SIEMENS

SINUMERIK

SINUMERIK ONE
基本功能

功能手册

适用于

控制系统
SINUMERIK ONE

CNC 软件　版本 6.21

01/2023
A5E48053578F AF

法律资讯

警告提示系统

为了您的人身安全以及避免财产损失，必须注意本手册中的提示。人身安全的提示用一个警告三角表示，仅与财产损失有关的提示不带警告三角。警告提示根据危险等级由高到低如下表示。

⚠ 危险
表示如果不采取相应的小心措施，**将会**导致死亡或者严重的人身伤害。

⚠ 警告
表示如果不采取相应的小心措施，**可能**导致死亡或者严重的人身伤害。

⚠ 小心
表示如果不采取相应的小心措施，可能导致轻微的人身伤害。

注意
表示如果不采取相应的小心措施，可能导致财产损失。

当出现多个危险等级的情况下，每次总是使用最高等级的警告提示。如果在某个警告提示中带有警告可能导致人身伤害的警告三角，则可能在该警告提示中另外还附带有可能导致财产损失的警告。

合格的专业人员

本文件所属的产品/系统只允许由符合各项工作要求的**合格人员**进行操作。其操作必须遵照各自附带的文件说明，特别是其中的安全及警告提示。 由于具备相关培训及经验，合格人员可以察觉本产品/系统的风险，并避免可能的危险。

按规定使用 Siemens 产品

请注意下列说明：

⚠ 警告
Siemens 产品只允许用于目录和相关技术文件中规定的使用情况。如果要使用其他公司的产品和组件，必须得到 Siemens 推荐和允许。正确的运输、储存、组装、装配、安装、调试、操作和维护是产品安全、正常运行的前提。必须保证允许的环境条件。必须注意相关文件中的提示。

商标

所有带有标记符号 ® 的都是 Siemens AG 的注册商标。本印刷品中的其他符号可能是一些其他商标。若第三方出于自身目的使用这些商标，将侵害其所有者的权利。

责任免除

我们已对印刷品中所述内容与硬件和软件的一致性作过检查。然而不排除存在偏差的可能性，因此我们不保证印刷品中所述内容与硬件和软件完全一致。印刷品中的数据都按规定经过检测，必要的修正值包含在下一版本中。

Siemens AG
Digital Industries
Postfach 48 48
90026 NÜRNBERG
德国

A5E48053578F AF
Ⓟ 01/2023 本公司保留更改的权利

目录

前言 1

1.1 关于 SINUMERIK

无论是普及型数控机床，还是标准型机床，或者是模块化高端机床，SINUMERIK 数控系统都能为不同类型的机床提供最佳解决方案。无论是单件生产还是批量生产、简单工件还是复杂工件，对于从样品和工具制造、模具制造乃至大批量生产的所有制造领域而言，SINUMERIK 自始至终都是高生产率的自动化解决方案。

详细信息请访问网页 SINUMERIK (https://www.siemens.com/sinumerik)。

1.2 关于本手册

本手册属于 SINUMERIK 功能手册系列。

SINUMERIK 功能手册

SINUMERIK 功能手册系列会介绍 SINUMERIK 控制系统的各种数控功能。

功能手册系列的目标读者是：机床配置选型人员、工艺工程师、调试人员和编程人员。

每本功能手册都涵盖了特定主题，介绍这些主题下的所有功能。

下表列出了 SINUMERIK 上的所有功能手册以及每本手册涵盖的主题：

功能手册	主题
基本功能	数控系统上的基本功能
进给轴和主轴	进给轴和主轴功能、轴耦合
坐标转换	坐标转换功能
监控和补偿	轴的监控和补偿功能、碰撞监控
刀具	刀具的选择、补偿和监控功能
同步动作	同步动作的功能
PLC	PLC 基本程序结构和功能
刀具管理	刀具管理的功能、调试和编程
工艺	工艺扩展功能

目录

功能手册的封面是一张目录，列出了手册的章节标题（即涉及的功能）。

适用性

手册封面还对本手册的适用性进行了说明，即该版本的编程手册适用于哪些 SINUMERIK 控制系统和哪些软件版本。

系统数据

在功能说明中，通常不会详细介绍功能涉及的系统数据（机床数据、设定数据、系统变量、接口信号和报警），除非是为了便于读者理解功能。这些数据的详细介绍可以查看对应的参数手册、报警查看诊断手册。

Basic Program Plus 和 Basic Program

在 TIA Portal 中为 PLC 功能的编程提供了两种不同的基本程序类型：

- Basis Program Plus 只能使用符号寻址进行编程。功能通过数据类型、PLC 程序块、PLC 函数和 PLC 指令来实现。Basic Program Plus 只可用于 SINUMERIK ONE。
- Basic Program 既可使用符号寻址，也可使用绝对寻址来编程。因此，大部分程序可与已有的 SIMATIC Manager 程序兼容。Basic Program 可用于 Steuerungen ONE 和 MC。

无法同时使用 Basic Program Plus 和 Basic Program。

PLC 程序块和数据类型

在功能说明中常会涉及 PLC 信号和 PLC 函数。在手册中主要是指 Basic Program Plus 的程序块和数据类型，这些只适用于 SINUMERIK ONE。

详细信息请参见各控制系统的 PLC 功能手册。

接口信号

进行接口信号寻址时，主要采用 Basic Program Plus 的符号寻址。对应的 Basic Program 信号，请参见各章节末尾的针对不同信号方向的 PLC 信号表，如以下示例所示：

NC → PLC

Basic Program Plus	Basic Program	
<Axis>.basic.in.enc1Synchronized	LBP_Axis*.E_RefSyn1	DB31, ...DBX60.4
...		

PLC → NC

Basic Program Plus	Basic Program	
<Axis>.basic.out.refPointApproachCam	LBP_Axis*.A_DelayRef	DB31, ...DBX12.7
...		

说明

数量结构

在 Basic Program Plus 中，通过相应的对象实例来映射轴、通道等。

在 Basic Program 中进行绝对寻址时，以下列组件的最大绝对数量为基础：运行方式组 (DB11)、通道 (DB21, ...)、轴/主轴 (DB31, ...)

标准功能范畴

本文档描述了标准功能范畴。该描述可能和交付的系统的功能有所不同。交付的系统的功能仅以订购资料为准。

在系统中也可能会运行本文档中未说明的功能，但这并不表示在交付系统时必须提供这些功能以及相关的维修服务。

为使文档简明清晰，本文档并不包含所有产品类型的所有详细信息，也无法对安装、运行和维护中可能出现的各种情况逐一进行说明。

机床制造商在产品上增添或者更改的功能，由机床制造商进行说明。

第三方网页

本文档可能包含第三方网页链接。西门子对此类网页的内容不承担任何责任，也不会声明或认可此类网页或其内容为西门子所有。西门子并不能控制此类网页上的信息，也不对上述网页的内容和信息负责。使用上述网页的风险由用户承担。

1.3 网上文档

1.3.1 SINUMERIK ONE 文档一览

有关 SINUMERIK ONE（自版本 6.13 起）功能的全部文档，请参见 SINUMERIK ONE 文档一览 (https://support.industry.siemens.com/cs/ww/en/view/109768483)。

您可以直接打开文档或者下载 PDF 和 HTML5 格式。

文档分为以下几个类别：

- 用户：操作
- 用户：编程
- 制造商/服务：功能
- 制造商/服务：硬件
- 制造商/服务：配置/调试
- 制造商/服务：Safety Integrated
- 介绍和培训
- 制造商/服务：SINAMICS

1.3.2 SINUMERIK 操作组件文档一览

有关 SINUMERIK 操作组件的全部文档，请参见 SINUMERIK 操作组件文档一览 (https://support.industry.siemens.com/cs/document/109783841/technische-dokumentation-zu-sinumerik-bedienkomponenten?dti=0&lc=en-WW)。

您可以直接打开文档或者下载 PDF 和 HTML5 格式。

文档分为以下几个类别：

- 操作面板
- 机床控制面板
- 机床按钮面板
- 手持单元/微型手持单元
- 其他操作组件

有关“SINUMERIK”的重要文档、文章和链接，请参见 SINUMERIK 专题页 (https://support.industry.siemens.com/cs/document/109766201/sinumerik-an-overview-of-the-most-important-documents-and-links?dti=0&lc=en-WW)。

1.4 技术文档反馈

对于西门子工业在线支持上发布的任何技术文档，如有疑问、建议或改进意见，请点击文章末尾的链接“发送反馈”。

1.5 mySupport 文档

使用网页版“mySupport 文档”可以自由组合西门子文档内容，创建自己的文档。

在 SiePortal 页面“mySupport 链接和工具” (https://support.industry.siemens.com/cs/cn/zh/my)上点击“我的文档”，便可启动应用：

配置的手册可以 RTF、PDF 或 XML 格式导出。

说明

在链接“配置”下可以查看网页版“mySupport 文档”支持的西门子文档内容。

1.6 服务与支持

产品支持

有关产品的详细信息请访问网址：

产品支持 (https://support.industry.siemens.com/cs/cn/zh/)

在该网址下可以提供：

- 最新产品信息（产品公告）
- FAQ（常见问题与解答）
- 手册
- 下载链接
- 持续提供产品最新信息的新闻。
- “技术论坛”，供全球用户和专家交流经验、分享信息
- “联系人”，提供全球联系人信息，方便查找本地联系人
- “售后服务”，提供现场服务、维修、备件等信息

技术支持

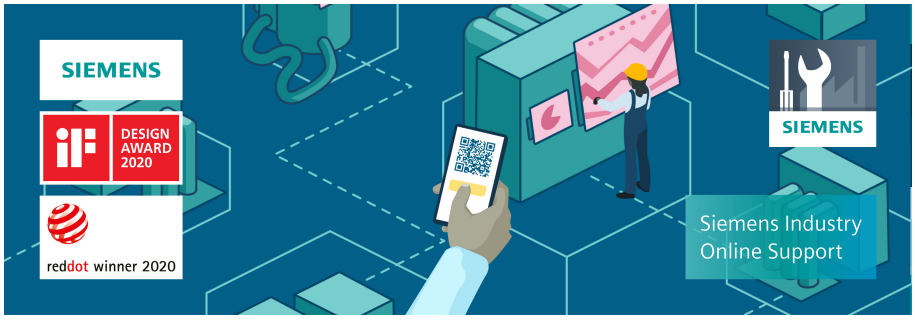
访问网址 (https://support.industry.siemens.com/cs/cn/zh/sc/4868)下的“联系方式”，便可以获取各个国家技术支持的电话号码。

如需咨询技术疑问，请使用“支持请求”一栏下的在线表格。

培训

访问网址 (https://www.siemens.com/sitrain)，可获取有关 SITRAIN 的相关信息。
SITRAIN 为西门子的驱动和自动化产品、系统和解决方案提供培训。

无论在何处都能得到最佳的支持



使用荣获大奖的“西门子工业在线支持”App，您可以随时随地查看超过 30 万份的西门子工业领域的产品文件。该应用也可帮助您：

- 解决项目实施中出现的问题
- 排除故障
- 进行设备扩展或重新规划

此外，您还可以登录技术论坛，查看我们的专家为您撰写的其他文章：

- 常见问题与解答
- 应用实例
- 手册
- 证书
- 产品公告等

“西门子工业在线支持”App 提供 Apple iOS 版和安卓版。

铭牌上的二维码

铭牌上的二维码包含了各设备的数据。使用任一智能手机通过“西门子工业在线支持”App 扫描该二维码，便可获取相应设备的技术信息。

1.7 OpenSSL 的使用

本产品可包含以下软件：

- 由 OpenSSL 项目开发并应用在 OpenSSL Toolkit 中的软件
- 由 Eric Young 开发的加密软件
- 由 Eric Young 开发的软件

详细信息请访问网址：

- OpenSSL (https://www.openssl.org)
- Cryptsoft (https://www.cryptsoft.com)

1.8 遵守通用数据保护条例

西门子遵守通用数据保护条例，特别是隐私保护设计（privacy by design）规定。

对于本产品意味着：

产品不会处理或保存个人相关数据，只会处理或保存技术功能数据（例如：时间戳）。用户如果将此类技术功能数据与其他数据（例如：排班表）关联或者将个人相关数据存储在同一介质（例如：硬盘）上而产生个人相关性，则应由用户自行确保遵循数据保护法规。

基本安全说明 2

2.1 一般安全说明

⚠ 警告

未遵循安全说明和遗留风险可引发生命危险

忽视随附硬件文档中的安全说明和遗留风险会导致重伤或死亡。

- 遵守硬件文档中的安全说明。
- 进行风险评估时应考虑到遗留风险。

⚠ 警告

因参数设置错误或修改参数设置引起机器故障

参数设置错误可导致机器出现故障，从而导致人员重伤或死亡。

- 采取保护措施，防止未经授权的参数设置。
- 采取适当措施（如驻停或急停）处理可能出现的故障。

2.2 应用示例的质保规定

应用示例在组态和配置以及各种突发事件方面对设备没有强制约束力，无需一一遵循。应用示例不会提供客户专用的解决方案，仅在典型任务设置中提供保护。

用户自行负责上述产品的规范运行事宜。应用示例并没有解除您在应用、安装、运行和维护时确保安全环境的责任。

2.3 安全性信息

Siemens 为其产品及解决方案提供了工业信息安全功能，以支持工厂、系统、机器和网络的安全运行。

为了防止工厂、系统、机器和网络受到网络攻击，需要实施并持续维护先进且全面的工业信息安全保护机制。Siemens 的产品和解决方案构成此类概念的其中一个要素。

客户负责防止其工厂、系统、机器和网络受到未经授权的访问。只有在有必要连接时并仅在采取适当安全措施（例如，防火墙和/或网络分段）的情况下，才能将该等系统、机器和组件连接到企业网络或 Internet。

关于可采取的工业信息安全措施的更多信息，请访问 https://www.siemens.com/industrialsecurity

Siemens 不断对产品和解决方案进行开发和完善以提高安全性。Siemens 强烈建议您及时更新产品并始终使用最新产品版本。如果使用的产品版本不再受支持，或者未能应用最新的更新程序，客户遭受网络攻击的风险会增加。

要及时了解有关产品更新的信息，请订阅 Siemens 工业信息安全 RSS 源，网址为 https://www.siemens.com/cert

其他信息请上网查找：

工业安全功能选型手册 (https://support.industry.siemens.com/cs/cn/zh/view/108862708/en)

⚠ 警告

篡改软件会引起不安全的驱动状态

篡改软件（如：病毒、木马、蠕虫等）可使设备处于不安全的运行状态，从而可能导致死亡、重伤和财产损失。

- 总是使用最新版本的软件。
- 将自动化和驱动组件集成到设备或机器上的整套先进工业信息安全方案中。
- 全面考虑整套工业信息安全方案中使用的所有产品。
- 采取相应的保护措施（如：使用杀毒软件）防止移动存储设备中的文件受到恶意软件的破坏。
- 在调试结束后，检查所有和安全相关的设置。

BAG、通道、程序运行、复位特性 3

3.1 简要说明

通道

手动运行轴和自动执行零件程序时，一个 NC 通道为最小单位。一个通道在某个时间点总是处于特定的运行方式下，例如 AUTOMATIC、MDI 或 JOG。可将一个通道视为一个独立的 NC。

运行方式组（BAG）

一个通道总是属于一个运行方式组。一个运行方式组也可能由多个通道组成。

运行方式组的标志是：同一运行方式组中的所有通道在一个时间点总是处于同一运行方式下，例如 AUTOMATIC、MDI 或 JOG。这通过 NC 内部运行方式逻辑确保。

可将一个运行方式组视为独立的多通道 NC。

通道空位

配置通道时可设置占位通道，从而使对同系列机床的配置尽可能一致，并只激活实际使用的通道。

程序测试

可通过以下方式对新的零件程序进行测试或试运行：

- 不输出设定值的程序执行
- 程序单段运行
- 采用空运行进给执行程序
- 跳过零件程序段
- 进行/不进行计算的程序段搜索

程序段搜索

通过程序段搜索可实现下列程序仿真，用于搜索特定的程序位置：

- 类型 1，不在轮廓处计算
- 类型 2，在轮廓处计算
- 类型 4，在程序段终点计算
- 类型 5，通过从以往的信息计算所需的所有数据来实现所选程序位置的自动启动
- 程序段搜索后自动启动 ASUB
- 级联程序段搜索
- 在“程序测试”模式下执行跨通道程序段搜索

程序运行

程序运行是指在 AUTOMATIC 或 MDI 运行方式下执行零件程序或零件程序段。执行期间可通过 PLC 接口信号和指令对程序运行进行调整。

通过通道专用机床数据可为每个通道设定初始设置。这些初始设置例如会影响 G 功能组合辅助功能输出。

只有在所涉及通道处于复位状态下时才能选择零件程序。

此外，所有其他程序运行都通过 PLC 接口信号和对应的指令处理。

- 启动零件程序或零件程序段
- 零件程序计算和程序控制
- RESET 指令，程序状态和通道状态
- 对操作和程序动作的响应
- 事件控制的程序调用

异步子程序（ASUB），中断程序

借助中断输入功能，NC 可中断当前的零件程序执行，并在中断程序/ ASUB 中对高优先级事件进行响应。

单程序段

单程序段功能可协助用户逐段执行零件程序。

单程序段功能有 3 种设置方式：

- SLB1: = 插补单程序段
- SLB2: = 解码单程序段
- SLB3: = 循环中停止

基本程序段显示

除现有的程序段显示外，还可通过另一个基本程序段显示功能对可引起机床上动作的所有程序段进行显示。

实际逼近的终点位置作为绝对位置显示。位置值可选择以工件坐标系（WCS）为基准，或以可设定零点坐标系（SZS）为基准。

外部执行程序

在加工复杂的工件时， NC 的存储空间可能不足以存放程序。使用“外部执行”或“EES（Execution from External Storage，从外部存储器执行）”功能可从一个外部存储器（例如硬盘）调用和执行零件程序。

上电/复位等情形后的特性

下列情形后的控制系统特性：

- 启动（上电）
- 复位/零件程序结束
- 零件程序开始

可通过机床数据修改以针对特定系统设置，例如用于 G 指令、刀具长度补偿、转换、耦合组、切向跟踪、可编程同步主轴等功能。

调用含 M 功能、T 功能或 D 功能的子程序

在某些应用中，通过子程序替代 M 功能、T 功能、D 功能以及个别 NC 语言指令（SPOS、SPOSA）更为适宜。这例如可用于调用换刀程序。

通过相应的机床数据可对含 M 功能、T 功能或 D 功能的子程序（例如用于齿轮档切换）进行定义和控制。

程序运行时间/工件计数器

控制系统可提供程序运行时间和工件计数的相关信息，以协助机床操作人员。

此处定义的功能**与刀具管理功能不同**，特别针对无刀具管理功能的 NC 系统。

3.2 运行方式组（BAG）

运行方式组

在运行方式组（BAG）中，NC 的多个通道组合成一个加工单元。原理上可将其视作 NC 内部的一个独立“NC”。

BAG 的主要特征：归属于同一 BAG 的所有通道在同一时间点始终处于相同的运行方式（AUTOMATIC、JOG、MDI）下。

说明

下面的功能说明以一个 BAG 和一条通道为例。

需要多条通道的功能（例如“轴交换”功能），请见：

更多信息：

更多信息参见 跨通道程序协调和逐通道试运行 (页 587)。

更多信息请见参数分配

通过以下数据将通道指定给 BAG：

MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP[<通道下标>] = BAG 编号

所有具有相同 BAG 编号的通道共同构成一个 BAG。

说明

默认情形下，NC 始终包含一个有一条通道的 BAG。无法参数设置无通道的 NC。

BAG 专用 NC/PLC 接口信号

BAG 专用 NC/PLC 接口信号位于数据块 DB11 中。

BAG 专用 NC/PLC 接口主要涵盖以下接口信号：

- 请求信号 PLC → NC
  - BAG 复位
  - BAG 停止，进给轴和主轴
  - BAG 停止
  - 运行方式切换禁止
  - 运行方式：JOG、MDI、AUTO
  - 单程序段：类型 A、类型 B
  - 机床功能：REF、REPOS、TEACH IN、INC x
- 状态信号 NC → PLC
  - 运行方式选通脉冲：JOG、MDI、AUTOMATIC
  - 机床功能选通脉冲：REF、REPOS、TEACH IN
  - 所有通道处于复位状态
  - BAG 复位
  - BAG 就绪
  - 运行方式选通脉冲：JOG、MDI、AUTOMATIC
  - 机床功能生效：REF、REPOS、TEACH、INC x

激活的和未激活的通道，通道空位

激活的通道

BAG 编号 ≠ 0 的通道为激活的通道。

未激活的通道

BAG 编号 为 0 的通道是未激活的通道。其不会占用控制系统内部的存储空间。

通道空位

BAG 编号 为 0 的通道不仅仅是未激活的通道。在通道序列中其即所谓的通道空位。

通道空位的优势在于：能够在一系列结构相似的机床中尽可能保持配置数据相同。特定调试中只会激活当前机床所需的通道。未占用的存储空间则可作为附加的用户存储器自由使用。

表格 3-1　示例

机床数据	含义
MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP[0] = 1	通道 1，BAG 1
MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP[1] = 2	通道 2，BAG 2
MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP[2] = 0	通道 3，不存在
MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP[3] = 1	通道 4，BAG 1
MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP[4] = 2	通道 5，BAG 2

3.2.1　BAG 停止

功能

通过以下 NC/PLC 接口信号在 BAG 的所有通道中停止进给轴的运行（或进给轴和主轴的运行），以及中断零件程序执行：

`<ModeGroup>.basic.out.stopAxesOnly`（运行方式组停止）

`<ModeGroup>.basic.out.stopAll` （进给轴和主轴 BAG 停止）

信号表

PLC → NC

Basic Program Plus	Basic Program	
<ModeGroup>.basic.out.stopAxesOnly	LBP_ModeGroup.A_MGStop	DB11.DBX0.5
<ModeGroup>.basic.out.stopAll	LBP_ModeGroup.A_MGStopASp	DB11.DBX0.6

3.2.2 BAG 复位

功能

BAG 复位请求通过以下 BAG 专用 NC/PLC 接口信号发出：

<ModeGroup>.basic.out.reset = 1（BAG 复位）

结果

对 BAG 中的通道生效：

- 零件程序准备（预处理）停止。
- 所有进给轴和主轴均根据加速度特性曲线，在不损坏轮廓的情况下制动至静止状态。
- 尚未输出至 PLC 的辅助功能将不再输出。
- 预处理指针会被设置至中断位置，程序段指针会被设置至各零件程序的起始处。
- 所有原始设置（例如 G 指令）均设置为参数设置的值。
- 所有删除标准为“通道复位”的报警均会被删除。

当 BAG 的所有通道都处于复位状态时：

- 所有删除标准为“BAG 复位”的报警均被删除。
- 会在 NC/PLC 接口上显示“BAG 复位完成”和“BAG 就绪”：
  <ModeGroup>.basic.in.resetAllChanDone = 1（所有通道处于复位状态）
  <ModeGroup>.basic.in.ready = 3（BAG 就绪）

信号表

NC → PLC

Basic Program Plus	Basic Program	
<ModeGroup>.basic.out.reset	LBP_ModeGroup.A_MGReset	DB11.DBX0.7

PLC → NC

Basic Program Plus	Basic Program	
<ModeGroup>.basic.in.ready	LBP_ModeGroup.E_MGOK	DB11.DBX6.3
<ModeGroup>.basic.in.resetAllChanDone	LBP_ModeGroup.E_ChanReset	DB11.DBX6.7

3.3 运行方式和运行方式切换

运行方式的唯一性

一个运行方式组（BAG）的所有通道总是处于同一运行方式下：

- AUTOMATIC
- JOG
- MDI

若各通道被指定给不同的运行方式组，那么通道切换时其同样会切换至相应的 BAG。借此可通过通道切换实现运行方式切换。

运行方式

可供使用的运行方式如下：

- **AUTOMATIC**
  自动执行零件程序：
  – 零件程序测试
  – BAG 的所有通道可同时生效。
- **JOG in AUTOMATIC**
  JOG in AUTOMATIC 是对 AUTOMATIC 运行方式的扩展，目的在于简化操作。在前提条件允许的情形下，可不退出 AUTOMATIC 模式执行 JOG。

- **JOG**
  通过机床控制面板上的运行键或机床控制面板上连接的手轮手动运行轴：
  – 在通过 ASUB 或静态同步动作执行的运行中，须注意通道专用信号和闭锁。
  – 同时须留意耦合关系。
  – BAG 中每条通道均可生效。
- **MDI**
  Manual Data Automatic（通过操作界面输入程序段）：
  – 在限制下执行零件程序和部分零件程序。
  – 零件程序测试
  – 每个 BAG 最多 1 条通道生效（仅在 TEACH IN 中）。
  – 只能在 JOG、REPOS 或 TEACHIN 等下级机床功能中手动运行轴。

适用于所有运行方式

跨运行方式同步动作

对于所有运行方式，可通过 IDS 并行于通道为以下功能执行模态同步动作：

- 指令轴功能
- 主轴功能
- 工艺循环

选择

用户可通过操作界面上的软键选择所需的运行方式。

此选择（AUTOMATIC、MDI 或 JOG）会被传输至 PLC 的 NC/PLC 接口，但尚不会被激活：

- "<HmiModeGroup>.basic.in.autoModeRequested"
- "<HmiModeGroup>.basic.in.mdaModeRequested"
- "<HmiModeGroup>.basic.in.jogModeRequested"（运行方式选通脉冲）

激活和优先级

BAG 的运行方式通过以下 NC/PLC 接口激活：

- "<HmiModeGroup>.basic.in.autoModeRequested"
- "<HmiModeGroup>.basic.in.mdaModeRequested"
- "<HmiModeGroup>.basic.in.jogModeRequested"（运行方式选通脉冲）

若在同一时间选择了多个运行方式，那么其优先级如下：

优先级	运行方式	BAG 信号（NC → PLC）
第 1 优先级，高	JOG	<ModeGroup>.basic.out.jogMod
第 2 优先级，中	MDA	<ModeGroup>.basic.out.mdaMode
第 3 优先级，低	AUTOMATIC	<ModeGroup>.basic.out.autoMode

显示

BAG 的当前运行方式通过以下 NC/PLC 接口显示：

- "<ModeGroup>.basic.in.autoModeActive"
- "<ModeGroup>.basic.in.mdaModeActive"
- "<ModeGroup>.basic.in.jogModeActive"

BAG 信号（NC → PLC）	生效的运行方式
<ModeGroup>.basic.in.jogModeActive	JOG
<ModeGroup>.basic.in.mdaModeActive	MDA
<ModeGroup>.basic.in.autoModeActive	AUTOMATIC

机床功能

可在运行方式内选择机床功能，其在 BAG 内同样适用：

- **JOG** 运行方式内的机床功能
  - REF（回参考点）
  - REPOS（重新定位）
  - JOG 回退（沿刀具方向回退）
- **MDI** 运行方式内的机床功能
  - REF（回参考点）
  - REPOS（重新定位）
  - TEACHIN（轴位置示教）

NC/PLC 接口信号

请求机床功能选通脉冲：

- "<HmiModeGroup>.basic.in.teachInRequested"
- "<HmiModeGroup>.basic.in.reposRequested"
- "<HmiModeGroup>.basic.in.refRequested"

激活机床功能：

- "<ModeGroup>.basic.out.teachIn"
- "<ModeGroup>.basic.out.repos"
- "<ModeGroup>.basic.out.refPoint"

机床功能生效反馈消息：

- "<ModeGroup>.basic.in.teachInActive"
- "<ModeGroup>.basic.in.reposActive"
- "<ModeGroup>.basic.in.refActive"

通道状态

- **通道复位**
  机床处于初始状态。这由机床制造商通过 PLC 程序定义，例如在接通或程序结束后。
- **通道生效**
  一个程序已启动，正在执行程序或回参考点。
- **通道中断**
  运行的程序或回参考点运行已被中断。

运行方式内的功能

可为运行方式补充用户自定义功能。这些功能与工艺和机床无关，可从“通道复位”、“通道生效”或“通道中断”这三种状态启动和/或执行。

适用于机床功能 TEACH IN 的前提条件

TEACH IN 不可用于生效轴组中的引导轴，例如：

- 龙门轴组或龙门轴对
- 引导轴和跟随轴组成的耦合轴组

JOG in AUTOMATIC

当 BAG 处于"RESET"状态下且轴可执行 JOG 时，允许在 AUTOMATIC 运行方式下使用 JOG。BAG 的"RESET"状态表示：

- 所有通道处于"RESET"状态
- 所有程序已终止
- 各通道中均无 DRF 生效

当轴不处于以下状态时，则表示**可执行 JOG**：

- PLC 轴作为并行定位轴（由 PLC 请求轴）
- 指令轴（轴已通过同步动作编程，且运动尚未结束）
- 旋转主轴（RESET 后仍旋转的主轴）
- 异步往复轴

提示："可执行 JOG"属性与"JOG in AUTOMATIC"功能无关。

激活

可通过以下机床数据激活"JOG in AUTOMATIC"功能：

MD10735 $MN_JOG_MODE_MASK

- 上电前必须设置以下机床数据：
  MD10735 $MN_JOG_MODE_MASK，位 0 = 1
- 用户切换至 AUTO（PLC 用户接口 `<ModeGroup>.basic.out.autoMode` = 0→1 脉冲沿）。若此前 BAG 所有通道中 NC 的通道状态均为"RESET"且程序状态均为"终止"，那么"JOG in AUTOMATIC"生效。此外所涉及的轴必须"可执行 JOG"。DRF 必须取消。
- 在所有通道状态不为"RESET"和程序状态不为"终止"的 BAG 通道中，系统会触发 RESET，或者通过 `M30`/`M2` 使运行中的程序终止。
- 所涉及的轴会自动变为"可执行 JOG"（例如跨通道取轴：PLC → NC）。

提示：在多数应用场合下，待运行轴均"可执行 JOG"，因此切换至 AUTOMATIC 运行方式后"JOG in AUTOMATIC"便生效。

特性

- 通过 +/–键可产生 JOG 运动，BAG 将**内部**切换至 JOG（即"内部 JOG"）。
- DRF 未生效时，通过运行手轮可产生 JOG 运动，BAG 将**内部**切换至 JOG（即"内部 JOG"）。

- JOG 运动开始后，达到增量的终点位置（若事先进行了设置）或通过“删除剩余行程”终止运行时运动才告结束。
  这样一来，可使用停止键暂停增量运行，并通过开始键重新运行至终点。在此期间 NC 均处于“内部 JOG”模式下。可运行部分增量，但不可通过停止键中断。另有一种模式，在该模式下松开运行键会使增量运行中断。
- 不执行 JOG 运动时，“JOG in AUTOMATIC”的特性与“AUTOMATIC”相同，特别是通过开始键启动所选择的零件程序，以及通过相应的 HMI 软键触发程序段搜索这些方面。
- JOG 运动生效时，NC 进入内部 JOG 模式，此时程序段搜索请求会被拒绝，也无法通过开始键启动零件程序。此时开始键用于启动可能剩余的增量运行，或者无作用。
- 只要 BAG 中的一根轴在 JOG 下运行，该 BAG 便处于内部 JOG 模式下。
  注释：此阶段可能随一根轴的 JOG 运动开始，在*另一根*轴的 JOG 运动完成时才告结束。
- 不允许对 JOG 运动生效的轴执行跨通道取轴。（该轴可能会切换 BAG）。此时 NC 会禁止跨通道取轴。
- PLC 用户接口显示“Automatic”运行方式：
  - “<ModeGroup>.basic.in.autoModeActive” = 1
  - “<ModeGroup>.basic.in.mdaModeActive” = 1
  - “<ModeGroup>.basic.in.jogModeActive” = 1
  - “<ModeGroup>.basic.in.teachInActive” = 0
  - “<ModeGroup>.basic.in.reposActive” = 0
  - “<ModeGroup>.basic.in.refActive” = 0
- 此 NC/PLC 接口显示，“JOG in AUTOMATIC”模式下 BAG 处于“BAG 复位”状态。
  - “<ModeGroup>.basic.in.MGreseted”（BAG 已复位，BAG x）
- 此 NC/PLC 接口显示，“JOG in AUTOMATIC”模式下 NC 已于内部切换至“内部 JOG”模式。
  - “<ModeGroup>.basic.in.resetDone”（NC 内部 JOG 激活，BAG x）

前提条件

仅当 BAG 处于“BAG 复位”状态时，“JOG in AUTOMATIC”才能于内部切换至 JOG。也就是说在一个暂停的程序中无法直接执行点动。在这种情形下，用户可在 BAG 的*所有通道中*按下 JOG 键或复位键来执行点动。

选择 AUTOMATIC 运行方式时，INC 键会被取消激活，用户可以/必须重新按下 INC 键来选择所需的增量。NC 切换至“内部 JOG”时，所选择的增量将得以保留。

若用户尝试对几何轴或定向轴执行点动，NC 同样会切换至“内部 JOG”，并会执行运动。此时可物理运行多根轴，这些轴必须均为“可执行 JOG”。

JOG 运动后 NC 会取消“内部 JOG”并重新选择 AUTO。内部模式切换会被延迟至运动结束。这样一来，很多开关步骤（例如在使用手轮时就可能出现）会受到不必要的避免。此时 PLC 只允许以 PLC 信号“内部 JOG 生效”为准。

在轴无使能的情形下，NC 仍会切换至“内部 JOG”。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<ModeGroup>.basic.in.autoModeActive	LBP_ModeGroup.autoModeActive	DB11.DBX6.0
<ModeGroup>.basic.in.mdaModeActive	LBP_ModeGroup.E_MDA	DB11.DBX6.1
<ModeGroup>.basic.in.jogModeActive	LBP_ModeGroup.E_JOG	DB11.DBX6.2
<ModeGroup>.basic.in.teachInActive	LBP_ModeGroup.E_TEACHIN	DB11.DBX7.0
<ModeGroup>.basic.in.reposActive	LBP_ModeGroup.E_REPOS	DB11.DBX7.1
<ModeGroup>.basic.in.resetDone	LBP_ModeGroup.E_Mgreseted	DB11.DBX6.4
<ModeGroup>.basic.in.nckInternalJogActive>	LBP_ModeGroup.E_NCKintJOG	DB11.DBX6.5

PLC → NC

Basic Program Plus	Basic Program	
<ModeGroup>.basic.out.autoMode	LBP_ModeGroup.A_AUTO	DB11.DBX0.0
<ModeGroup>.basic.out.mdaMode	LBP_ModeGroup.A_MDA	DB11.DBX0.1
<ModeGroup>.basic.out.jogMode	LBP_ModeGroup.A_JOG	DB11.DBX0.2
<ModeGroup>.basic.out.teachIn	LBP_ModeGroup.A_TEACHIN	DB11.DBX1.0
<ModeGroup>.basic.out.repos	LBP_ModeGroup.A_REPOS	DB11.DBX1.1
<ModeGroup>.basic.out.reset	LBP_ModeGroup.A_MGReset	DB11.DBX0.7

OP → PLC

Basic Program Plus	Basic Program	
<HmiModeGroup>.basic.in.autoModeRequested	LBP_HMI.E_MMC_AUTO	DB11.DBX4.0
<HmiModeGroup>.basic.in.mdaModeRequested	LBP_HMI.E_MMC_MDA	DB11.DBX4.1
<HmiModeGroup>.basic.in.jogModeRequested	LBP_HMI.E_MMC_JOG	DB11.DBX4.2

Basic Program Plus	Basic Program	
<HmiModeGroup>.basic.in.teachInRequested>	LBP_HMI.E_MMC_TEACHIN	DB11.DBX5.0
<HmiModeGroup>.basic.in.reposRequested	LBP_HMI.E_MMC_REPOS	DB11.DBX5.1
<HmiModeGroup>.basic.in.refRequested	LBP_HMI.E_MMC_REF	DB11.DBX5.2

3.3.1 对各运行方式的监控和锁定

通道状态决定监控功能

运行方式下的监控

各种运行方式下生效的监控不尽相同。 这些监控与工艺和机床无关。

在每种运行方式下，根据运行状态会有部分监控功能生效。 哪些监控在哪些运行方式下和哪些运行状态中生效，这由通道状态决定。

运行方式下的锁定

各种运行方式下可能会有不同的锁定功能生效。 这些锁定与工艺和机床无关。

在每种运行方式下，根据运行状态可激活几近所有锁定功能。

3.3.2 运行方式切换

引言

通过 BAG 接口信号（<ModeGroup>.basic.）请求和激活运行方式的转换。一个运行方式组只能处于 AUTOMATIC、JOG 或 MDI 这其中一种运行方式下，即一个运行方式组中的多个通道不能同时处于不同的运行方式下。

是否能打到所请求的运行方式，以及如何执行该运行方式，这些均通过 PLC 程序针对机床配置。

说明

仅当不处于“通道状态生效”情形时，才能于控制系统内部实现运行方式切换。但是，为了实现无故障的运行方式切换，所有通道都须处于允许采用的运行状态下。

可进行的运行方式切换

下表显示了可对一个通道进行的运行方式切换。

	AUTOMATIC		JOG				MDI		
				AUTO	MDI		手动运行		AUTO
	复位	中断	复位	中断	中断	复位	中断	生效	中断
AUTOMATIC			X	X		X			
JOG	X	X				X	X		X
MDI	X	X	X		X				

“x”标记表示可进行的运行方式切换。

特殊情况

- **运行方式切换时报错**
  若运行方式切换请求被系统拒绝，则会发出故障信息“NC 停止后才可进行运行方式切换”。无需修改通道状态便可清除此故障信息。
- **运行方式切换禁止**
  通过以下接口信号：
  `<ModeGroup>.basic.out.disableModeChangeover`（运行方式切换禁止）
  可禁止运行方式切换。
  此时运行方式请求即已被抑制。
  用户须自行配置一条消息，以提示操作人员禁止生效。系统不会为此提供任何信号。
- **运行方式从 MDI 切换至 JOG**
  在运行方式从 MDI 切换至 JOG 后，若 BAG 的所有通道均处于复位状态，那么 NC 将从 JOG 切换至 AUTO。在此状态下可执行零件程序指令 `START` 或 `INIT`。
  若运行方式切换后 BAG 的一个通道不再处于复位状态，那么此时零件程序指令 `START` 会被系统拒绝，并触发报警 16952。

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<ModeGroup>.basic.out.disableModeChangeover	LBP_ModeGroup.A_MCDisable	DB11.DBX0.4

3.4 通道

3.4.1 属性

NC 通道用于执行由用户设定的零件程序。

通道具有以下属性：

- 一条通道始终被指定给一个运行方式组 (页 32)（BAG）。
- 一条通道在一个时间点始终只能执行一个零件程序。
- 可将通道指定给机床轴、几何轴、副主轴以及主轴。只有这些轴可通过通道中执行的零件程序来运行。
- 一条通道包含以下内部单元
  - 预处理（程序解码和程序段预处理）
  - 主处理（轨迹插补和轴插补）
- 通道有一个 PLC 接口。借助此 NC/PLC 接口可通过 PLC 用户程序读取各种通道专用状态数据，以及写入对通道的请求。
- 通道中会启用通道专用的刀具补偿（参见*刀具*功能手册*刀具补偿*部分）。
- 通道中会启用通道专用的坐标系 (页 319)。
- 默认情形下，每条通道均具有一个独有的通道名称。设定的通道名称可通过以下机床数据修改：
  MD20000 $MC_CHAN_NAME = "<通道名称>"

多条通道可划归一个运行方式组（BAG）。一个 BAG 中的通道始终处于相同的运行方式（AUTOMATIC、JOG、MDI）下。

通道配置

通过以下机床数据可为通道设定名称：

MD20000 $MC_CHAN_NAME（通道名称）

各轴通过机床数据指定给现有通道。对于进给轴/主轴，同时只能有一个输出设定值的通道。进给轴/主轴的实际值可由多个通道同时读取。进给轴/主轴的信息必须已提供给相应通道。

此外还可通过机床数据定义以下通道专用设置：

- G 功能组的缺省设置或编程初始设置，通过以下机床数据设定：
  MD20150 $MC_GCODE_RESET_VALUES（G 功能组的初始设置）
- 辅助功能组，构成方式和输出时间点
- 机床轴和几何轴之间的转换条件
- 其他用于零件程序执行的设置

修改通道配置

通道配置无法以编程的方式在零件程序中或通过 PLC 用户程序在线修改。配置修改必须通过机床数据进行。重新上电后修改才生效。

接口信号

通道 1 的信号位于 DB21 的 NC/PLC 接口中，通道 2 的信号位于 DB22 中，可通过 PLC 或 NC 监控或控制通道。

通道专用工艺设定

可为每个通道设定使用的工艺：

MD27800 $MC_TECHNOLOGY_MODE

供货状态下，机床数据默认用于铣削工艺。

通过 PLC 使用主轴功能

除块 SINU_RotateSpindle / SINU_PositionSpindle（Basic Program FC18）外，也可通过轴专用 NC/PLC 接口信号并行于零件程序启动和停止主轴功能。

前提条件：

- 通道状态：“中断”或“RESET”
- 程序状态：“中断”或“终止”

以下功能可由 PLC 通过接口信号控制：

- 停止（相当于 M5）
- 开始顺时针旋转（相当于 M3）
- 开始逆时针旋转（相当于 M4）
- 选择齿轮档
- 定位（相当于 M19）

存在多个通道时，通过 PLC 启动的主轴在启动时所归属于的通道中生效。

关于特殊主轴接口的更多信息参见功能手册*进给轴和主轴*中的*主轴*部分。

PLC 控制的单轴进程

通过 PLC 不仅可以控制一个通道，也可实现对单轴的控制。为此，PLC 会通过 NC/PLC 接口从 NC 请求轴：

<Axis>.oscillation.out.plcCtrl = 1（PLC 控制轴）

以下功能可由 PLC 控制和调整：

- 终止进给轴/主轴的运行（相当于删除剩余行程）
- 停止或中断进给轴/主轴
- 继续运行进给轴/主轴（恢复运行）
- 将进给轴/主轴恢复至初始状态

通道专用信号交换（PLC → NC）的更多信息请见功能手册 *PLC*。

关于自控单轴进程的功能的更多信息参见功能手册*进给轴和主轴*中的*定位轴*部分。

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<Axis>.oscillation.out.plcCtrl	LBP_Axis*.A_PLCAxis	DB31.DBX28.7

3.4.2 启动禁止，全局和通道专用

功能

NC 程序的启动可以通过启动禁止（PI 服务）全局锁定或通道专用锁定。

全局启动禁止

若全局启动禁止置位，则所有通道中的 NC 程序启动均被禁止。已运行的 NC 程序不受影响。下一次 NC 程序启动才会被禁止。

通道专用启动禁止

通道专用启动禁止仅在 AUTO 运行方式以及通道状态“复位”下禁止 NC 程序的启动。

PI 服务

置位启动禁止（_N_STRTLK）

可通过 PI 服务 _N_STRTLK 以全局或针对特定通道的方式将启动禁止置位。

复位启动禁止（_N_STRTUL）

可通过 PI 服务 _N_STRTUL 将全局或通道专用启动禁止复位。

NC/PLC 接口信号

可通过 NC/PLC 接口信号取消启动禁止：

<Chan>.basic.out.suppressDisableNcStart （取消启动禁用）

机床数据

通过此机床数据进行以下设置：若在启动禁止置位的情况下请求启动，则显示报警。

MD11411 $MN_ENABLE_ALARM_MASK，位 6

BTSS 变量

变量	说明
startLockState	全局启动禁止的状态
chanStartLockState	通道专用启动禁止的状态
startRejectCounter	采用全局启动禁止时的启动计数器
startLockCounter	采用通道专用启动禁止时的启动计数器

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.suppressDisableNcStart	LBP_Axis*.A_SuppStartLock	DB21.DBX7.5

3.5 程序运行

3.5.1 程序运行

定义

程序运行是指在 AUTOMATIK 或 MDA 模式下执行 NC 程序或 NC 程序段。

NC/PLC 接口信号

程序运行可由 PLC 用户程序通过 BAG 和通道专用的 NC/PLC 接口信号进行控制或将相应的反馈信息反馈至 PLC 用户程序。

更多信息

NC/PLC 接口信号一览请见 PLC 功能手册；接口信号

3.5.2 缺省设置

通过通道专用机床数据可为每个通道设定初始设置。这些初始设置例如会影响 G 功能组合辅助功能输出。

辅助功能输出

可通过机床数据 AUXFU_x_SYNC_TYPE（MD22200、22210、22220、22230、22240、22250、22260；M、S、T、H、F、D、E 功能的输出时间点）预定义辅助功能的时序输出。更详细的说明请参见“PLC 辅助功能输出 (页 775)”章节。

G 功能组

对存在的每个 G 功能组均可通过
MD20150 $MC_GCODE_RESET_VALUES（G 组的缺省设置）设定一个编程缺省设置。在程序启动或复位状态中该编程缺省设置自动生效，直至其由于同 G 组的 G 指令被取消。

通过 MD22510 $MC_GCODE_GROUPS_TO_PLC（程序段切换/RESET 时向 NC-PLC 接口输出的 G 指令）可激活 G 指令向 PLC 接口的输出。

更多信息

G 指令组和对应 G 指令的列表请见编程手册“NC 编程”。

NC 语言范围的基本配置

可通过机床数据生成特定 NC 语言范围缺省配置，并对其进行设置。这样一来，用户可根据自身所需的选件和功能对 NC 语言范围进行定制。

3.5.2.1 机床数据

NC 语言范围

通过语言指令运行未生效选件和功能的方式可通过以下机床数据设置：

MD10711 $MN_NC_LANGUAGE_CONFIGURATION = <值>

值	含义
0	所有语言指令都可识别。执行时才识别所需功能是否激活。
1	所有语言指令都可识别。 未使能选件的语言指令在程序解释时会被识别 ⇒ 触发报警 12553 “选件/功能未生效”。
2	仅可识别已使能选件的语言指令。 未使能选件的语言指令在程序解释时会被识别 ⇒ 触发报警 12550 “名称未定义，或选件/功能不存在”。

值	含义
3	所有语言指令都可识别。 未激活功能的语言指令在程序解释时会被识别 ⇒ 触发报警 12553 “选件/功能未生效”。 **示例** 1. 柱面转换选件已设置 2. 柱面转换在机床数据 MD24100 $MC_TRAOF_TYPE_1 中**未**激活 3. 编程指令 TRACYL 会触发报警 12553。
4	仅可识别已激活功能的语言指令。 未激活功能的语言指令不会被识别 ⇒ 触发报警 12550 “名称未定义，或选件/功能不存在”。 **提示** 在此情形下无法分辨相关语言指令是不属于西门子语言范围，还是仅在相应设备上不存在。

3.5.2.2 编程

使用 "STRINGIS(...)" 功能可以检查，指定的字符串能否在当前语言集中作为 NC 编程语言元素使用。

可检查以下 NC 编程语言元素：

- 现有所有 G 功能组的 G 指令
- DIN 或 NC 地址
- 功能
- 步骤
- 关键字
- 系统数据，例如：机床数据 $M...、设定数据 $S... 或选项数据 $O...
- 系统变量 $A...、$V...、$P...
- 计算参数 R...
- 激活的循环的循环名称
- GUD 和 LUD 变量
- 宏名称
- 标签名称

定义

```
INT STRINGIS(STRING <名称>)
```

句法

```
<返回值> = STRINGIS(<名称>)
```

含义

`STRINGIS()`：	带返回值的校验函数
<名称>：	需要检查的字符串

<返回值>：	返回值在前三个小数位中进行编码 yxx	
	000	给定的字符串不是当前语言集的元素 [1)]
	100	给定的字符串是当前语言集的元素，但当前不可编程（选件/功能无效）
	2xx	给定的字符串是当前语言集的一个可编程元素（选项/功能生效）。 详细信息请见第 1 和第 2 个小数位 xx：
		xx **含义**
		01 DIN 地址或 NC 地址 [2)]
		02 G 指令（例如 G04、INVCW）
		03 带返回值的函数
		04 无返回值的函数
		05 关键字（例如 DEFINE）
		06 机床数据（$M...）、设定数据（$S...）或选项数据（$O...）
		07 系统参数，例如系统变量（$...）或计算参数（R...）
		08 循环（循环必须在 NC 中装载，并且循环程序生效 [3)]）
		09 GUD 变量（GUD 变量必须在 GUD 定义文件中定义，且必须被激活）
		10 宏名称（宏必须在宏定义文件中定义，且必须被激活）[4)]
		11 当前零件程序的 LUD 变量
		12 ISO G 指令（ISO 语言模式必须生效）
	400	给定的字符串是一个既未识别为 DIN 地址或 NC 地址 (xx==01)，又未识别为宏名称 (xx==10) 且不是 G 或者 R 的 NC 地址 [2)]
	y00	无专用分配

1) 某些控制系统可能只能识别西门子 NC 语言指令中的一部分，例如 SINUMERIK 802D sl。在这些控制系统上，对于这些原则上为西门子语言指令的字符串会返回值 0。可通过 MD10711 $MN_NC_LANGUAGE_CONFIGURATION 修改此特性。MD10711 = 1 时，对于西门子 NC 语言指令总是返回值 100。

2) NC 地址为以下字母：A, B, C, E, I, J, K, Q, U, V, W, X, Y, Z。NC 地址也可通过地址扩展编程。在使用 STRINGIS 进行检查时可设定地址扩展。示例：201 == STRINGIS("A1").
字母：D, F, H, L, M, N, O, P, S, T 为用户自定义的 NC 地址或辅助功能。对其总是返回值 400。示例：400 == STRINGIS("D").在使用 STRINGIS 检查这些 NC 地址时不可设定地址扩展。示例：000 == STRINGIS("M02")，但 400 == STRINGIS("M")。

3) 不可使用 STRINGIS 检查循环参数的名称。

4) 定义为宏的地址，例如 G, H, M, L，也被识别为宏

示例

在下面的示例中假设给定的 NC 语言元素在控制系统中可编程（若无特别说明）。

1. 字符串"T"定义为辅助功能：
   ```
   400 == STRINGIS("T")
   000 == STRINGIS ("T3")
   ```
2. 字符串"X"定义为进给轴：
   ```
   201 == STRINGIS("X")
   201 == STRINGIS("X1")
   ```
3. 字符串"A2"定义为带扩展的 NC 地址：
   ```
   201 == STRINGIS("A")
   201 == STRINGIS("A2")
   ```
4. 字符串"INVCW"定义为命名的 G 指令：
   ```
   202 == STRINGIS("INVCW")
   ```
5. 字符串"$MC_GCODE_RESET_VALUES"定义为机床数据：
   ```
   206 == STRINGIS("$MC_GCODE_RESET_VALUES")
   ```
6. 字符串"GETMDACT"定义为 NC 语言功能：
   ```
   203 == STRINGIS("GETMDACT ")
   ```
7. 字符串"DEFINE"定义为关键字：
   ```
   205 == STRINGIS("DEFINE")
   ```
8. 字符串"$TC_DP3"定义为系统参数（刀具长度分量）：
   ```
   207 == STRINGIS("$TC_DP3")
   ```
9. 字符串"$TC_TP4"为系统参数（刀具尺寸）：
   ```
   207 == STRINGIS("$TC_TP4")
   ```
10. 字符串"$TC_MPP4"为系统参数（刀库刀位状态）：
    - 刀具刀库管理生效：`207 == STRINGIS("$TC_MPP4")` ；
    - 刀具刀库管理未生效：`000 == STRINGIS("$TC_MPP4")`

    另见章节：刀具刀库管理。

11. 字符串"MACHINERY_NAME"定义为 GUD 变量：
```
209 == STRINGIS("MACHINERY_NAME")
```
12. 字符串"LONGMACRO"定义为宏：
```
210 == STRINGIS("LONGMACRO")
```
13. 字符串"MYVAR"定义为 LUD 变量：
```
211 == STRINGIS("MYVAR")
```
14. 字符串"XYZ"不是 NC 中已知的指令、GUD 变量、宏名称或循环名称：
```
000 == STRINGIS("XYZ")
```

边界条件

刀具刀库管理

如果刀具刀库管理功能未生效，则用于刀具刀库管理系统参数的 `STRINGIS()` 总是独立于以下机床数据

- MD10711 $MN_NC_LANGUAGE_CONFIGURATION

输出值 000。

ISO 模式

"ISO 模式"功能生效：

如果"ISO 模式"功能生效，`STRINGIS()` 会先将给定的字符串作为 SINUMERIK G 指令检查：

- MD18800 $MN_MM_EXTERN_LANGUAGE（激活外部 NC 语言）
- MD10880 $MN_ MM_EXTERN_CNC_SYSTEM（待匹配的控制系统）

"ISO 模式"生效时，`STRINGIS()` 会先检查给定的字符串与 SINUMERIK G 指令的归属关系。

如果该字符串不是 SINUMERIK G 指令，则随后检查其与 ISO G 指令的归属关系。

编程的切换 ISO 模式

编程的通过指令 `G290` （SINUMERIK 模式）和 `G291` （ISO 模式）进行的切换对 `STRINGIS()` 没有影响。

3.5.3 选择并启动 NC 程序

NC/PLC 接口信号

选择

仅当相关通道处于复位状态时，才可选择 NC 程序。

- <Chan>.basic.in.stateReset == 1（复位）

Start

NC 程序是通过两个不同的事件启动的：

1. <Chan>.basic.out.ncStart = 1（NC 启动），该信号通常是通过按下机床控制面板上的“NC 启动”键触发的。
2. 另一个生效通道的 NC 程序中的指令 START。该通道必须处于 AUTOMATIK 运行方式下以及“复位”或“中断”状态下。
   - <Chan>.basic.in.stateReset == 1（复位）
   - <Chan>.basic.in.stateInterrupted == 1（中断）

输出条件

仅当满足以下输出条件时，才能启动 NC 程序。

- <ModeGroup>.basic.in.ready == 1（BAG 就绪）
- <ModeGroup>.basic.out.reset == 0（BAG 复位）
- <Chan>.basic.out.progTest == 0（激活程序测试）
- <Chan>.basic.out.disableNcStart == 0（NC 启动禁止）
- <Chan>.basic.out.ncStopBlockEnd == 0（程序段交界处 NC 停止）
- <Chan>.basic.out.ncStop == 0（NC 停止）
- <Chan>.basic.out.ncStopAxesAndSpindles == 0（进给轴和主轴 NC 停止）
- <Chan>.basic.out.reset == 0（复位）
- <Nc>.basic.out.emergencyStop == 0（急停）
- 不可以存在轴或 NC 专用报警

执行指令

零件程序或零件程序段会自动执行，以下接口信号会被置位：

- <Chan>.basic.in.stateActive（通道状态激活）
- <Chan>.basic.in.progStateRunning（程序状态“运行”）

程序一直运行下去，直至到达程序结束，或者通道由于 STOP/RESET 指令中断或终止。

更多信息

对接口信号的详细说明请见 PLC 功能手册。

对 NC 变量的详细说明请参见 NC 变量参数手册

报警

在特定情形下，START 指令不生效，并会触发下列报警：

- 10200"报警生效时不允许进行 NC 启动"
- 10202"指令生效时不允许进行 NC 启动"
- 10203"对未回参考点的轴不允许进行 NC 启动"

更多信息

诊断手册之报警

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.stateReset	LBP_Chan*.E_ChanReset	DB21, ...DBX35.7
<Chan>.basic.in.stateInterrupted	LBP_Chan*.E_ChanInterrupt	DB21, ...DBX35.6
<Chan>.basic.in.stateActive	LBP_Chan*.E_ChanActive	DB21, ...DBX35.5
<Chan>.basic.in.progStateRunning	LBP_Chan*.E_ProgRunning	DB21, ...DBX35.0
<ModeGroup>.basic.in.ready	LBP_ModeGroup.E_MGOK	DB11.DBX6.3
<ModeGroup>.basic.out.reset	LBP_ModeGroup.A_MGReset	DB11.DBX0.7

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.ncStart	LBP_Chan*.A_NCStart	DB21, ...DBX7.1
<Chan>.basic.out.progTest	LBP_Chan*.A_ProgTest	DB21, ...DBX1.7
<Chan>.basic.out.disableNcStart	LBP_Chan*.A_NCStartDisabl	DB21, ...DBX7.0
<Chan>.basic.out.ncStopBlockEnd	LBP_Chan*.A_NCStopBlock	DB21, ...DBX7.2
<Chan>.basic.out.ncStop	LBP_Chan*.A_NCStop	DB21, ...DBX7.3
<Chan>.basic.out.ncStopAxesAndSpindles	LBP_Chan*.A_NCStopASp	DB21, ...DBX7.4

Basic Program Plus	Basic Program	
<Chan>.basic.out.reset	LBP_Chan*.A_Reset	DB21, ...DBX7.7
<Nc>.basic.out.emergencyStop	LBP_NC.A_EMERGENCY	DB10.DBX56.1

3.5.4 程序中断

NC/PLC 接口信号

前提条件

仅当通道和 NC 程序激活时，才能执行程序中断：

- <Chan>.basic.in.stateActive == 1（通道：生效）
- <Chan>.basic.in.progStateRunning == 1（程序：运行中）

程序中断

通过以下事件可以中断程序执行：

- <Chan>.basic.out.ncStopBlockEnd == 1（程序段交界处 NC 停止）
- <Chan>.basic.out.ncStop == 1（NC 停止）
- <Chan>.basic.out.ncStopAxesAndSpindles == 1（进给轴和主轴 NC 停止）
- <Chan>.basic.out.singleBlock == 1（激活单程序段）
- 已执行 NC 程序中的编程指令 M00 或 M01

之后通道和 NC 程序处于“中断”状态：

- <Chan>.basic.in.stateInterrupted == 1（通道中断）
- <Chan>.basic.in.progStateInterrupted == 1（程序中断）

更多信息

对接口信号的详细说明请见 PLC 功能手册

过程

识别出程序中断后会执行以下操作：

- 中断程序执行：
  - 在以下事件中会停止在下一个程序段边界上：“程序段交界处NC停止”、M00、M01或单程序段
  - 立即：所有其他时间
- 运行的通道轴通过制动斜坡停止。轴制动至静止状态可能需要持续多个程序段。
- 程序段指针显示中断位置的当前程序段。
- 到中断位置为止尚未输出的辅助功能将不再输出。

中断状态下可进行的操作

零件程序中断期间，可在通道中执行多种功能，例如：

- **溢出转存**
- **程序段搜索**
  更多信息请见 程序段搜索类型 1、2 和 4 (页 147)。
- **再次返回轮廓（REPOS）**
  更多信息请见 用于 RET 和 REPOS 的用户专用 ASUB (页 215)。
- **定向刀具回退**
  更多信息
  - NC 编程手册
  - 功能手册“刀具”
- **ASUB**（参见章节“异步子程序（ASUB） (页 199)”）
- **DRF 功能，工件零点的移动**
  更多信息
  功能手册“进给轴和主轴”；手动程序
- **继续执行中断的 NC 程序**
  - 来自另一个通道的指令 START
    更多信息
    编程手册“NC 编程”
  - NC/PLC 接口：
    <Chan>.basic.out.ncStart = 1（NC 重启）

参见

程序段搜索类型 5（SERUPRO） (页 163)

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.stateInterrupted	LBP_Chan*.E_ChanInterrupt	DB21, ...DBX35.6
<Chan>.basic.in.progStateInterrupted	LBP_ChanX.E_ProgInterrupt	DB21, ...DBX35.3
<Chan>.basic.in.stateActive	LBP_Chan*.E_ChanActive	DB21, ...DBX35.5
<Chan>.basic.in.progStateRunning	LBP_ChanX.E_ProgRunning	DB21, ...DBX35.0

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.ncStart	LBP_Chan*.A_NCStart	DB21, ...DBX7.1
<Chan>.basic.out.ncStop	LBP_Chan*.A_NCStop	DB21, ...DBX7.3
<Chan>.basic.out.ncStopAxesAndSpindles	LBP_Chan*.A_NCStopASp	DB21, ...DBX7.4
<Chan>.basic.out.singleBlock	LBP_Chan*.A_SBL	DB21, ...DBX0.4

3.5.5 通道复位

功能

在 AUTO 模式下，通过通道复位可以中断一个正在执行的 NC 程序；在 MDA 模式下，通过通道复位可以中断一个正在执行的程序段。

NC 程序或程序段无法再从中断位置继续执行。通道复位结束后，通道的所有进给轴和主轴都处于“准停”状态（例外：跟踪运行）。

过程

通过 NC 复位可在通道中执行以下操作：

- 立即停止程序准备。
- 进给轴和主轴通过其设置的制动斜坡减速至静止状态。
- 当前程序段尚未输出的辅助功能将不再输出到 PLC。
- 程序段指针返回到所选 NC 程序的起始处。
- 显示器中所有通道复位报警清除。

规则

- 通道复位可在各通道状态下执行。
- 通道复位不会由其他指令终止。

NC/PLC 接口信号

请求：通道复位

通过以下 NC/PLC 接口信号请求通道复位：

- `<Chan>.basic.out.reset` = 1（复位）

请求：BAG 复位

BAG 复位在所有 BAG 通道中都会触发通道复位。

通过以下 NC/PLC 接口信号请求 BAG 复位：

- `<ModeGroup>.basic.out.reset` = 1（BAG 复位）

反馈信息：通道复位结束

- `<Chan>.basic.in.stateReset` == 1（通道状态复位）

反馈信息：BAG 复位结束

- `<ModeGroup>.basic.in.resetDone` == 1（BAG 已复位）

更多信息

对接口信号的详细说明请见 PLC 功能手册。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.stateReset	LBP_Chan*.E_ChanReset	DB21, ...DBX35.7
<ModeGroup>.basic.in.resetDone	LBP_ModeGroup.E_MGreseted	DB11, ...DBX6.4

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.reset	LBP_Chan*.A_Reset	DB21, ...DBX7.7
<ModeGroup>.basic.out.reset	LBP_ModeGroup.A_MGReset	DB11, ...DBX0.7

3.5.6 程序状态

对于每个通道，所选 NC 程序的状态在接口中显示。

在 AUTOMATIC 和 MDA 运行方式下可以显示所有程序状态。在所有其他运行方式下或机床功能中，程序状态为终止或中断。

NC/PLC 接口信号

在 NC/PLC 接口（<Chan>.basic.in...）上会显示以下程序状态：

- <Chan>.basic.in.progStateAborted（“终止”）
- <Chan>.basic.in.progStateInterrupted（“中断”）
- <Chan>.basic.in.progStateStopped（“停止”）
- <Chan>.basic.in.progStateWaiting（“等待”）
- <Chan>.basic.in.progStateRunning（“运行”）

对接口信号的详细说明请参见参数手册，NC 变量和接口信号。

状态变化

程序状态受指令、NC/PLC 接口信号和报警影响。从程序状态“正在运行”出发，表格中显示了各个事件下的后续状态。

程序的输出状态：“正在运行”					
事件	程序的后续状态				
	终止	中断	停止	等待	运行
<Chan>.basic.out.reset（复位）	x				
<Chan>.basic.out.ncStop（NC 停止）			x		
<Chan>.basic.out.ncStopBlockEnd（程序段交界处 NC 停止）			x		
<Chan>.basic.out.ncStopAxesAndSpindles（进给轴和主轴 NC 停止）			x		
<Chan>.basic.out.disableReadIn（读取禁止）					x
<Chan>.basic.out.disableFeed（进给禁止）					x
<Chan>.geoAxis1.out.disableFeed （进给停止，几何轴 1 / 2 / 3）					x

程序的输出状态：“正在运行”					
事件	程序的后续状态				
	终止	中断	停止	等待	运行
<Chan>.basic.out.pathFeedrateOvrFactor（进给倍率 = 0%）					x
<Chan>.basic.out.ncStopAxesAndSpindles（进给停止/主轴停止）					x
<Chan>.auxMDynFuncs.in.m02Decoded（程序段中 M02 指令）	x				
<Chan>.auxMDynFuncs.in.m30Decoded（程序段中 M30 指令）	x				
<Chan>.auxMDynFuncs.in.m00Decoded（程序段中 M00 指令）			x		
<Chan>.auxMDynFuncs.in.m01Decoded（程序段中 M01 指令）			x		
<Chan>.basic.out.singleBlock（单程序段）			x		
<Chan>.basic.out.deleteDistanceToGo（删除剩余行程）					x
向 PLC 输出辅助功能，但尚未应答			x		
程序中的 WAIT 指令				x	
报警及系统响应“NOREADY”		x			

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.progStateRunning	LBP_Chan*.E_ProgRunning	DB21, ...DBX35.0
<Chan>.basic.in.progStateWaiting	LBP_Chan*.E_ProgWait	DB21, ...DBX35.1
<Chan>.basic.in.progStateStopped	LBP_Chan*.E_ProgStop	DB21, ...DBX35.2
<Chan>.basic.in.progStateInterrupted	LBP_Chan*.E_ProgInterrupt	DB21, ...DBX35.3
<Chan>.basic.in.progStateAborted	LBP_Chan*.E_ProgAborted	DB21, ...DBX35.4

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.singleBlock	LBP_Chan*.A_SBL	DB21, ...DBX0.4
<Chan>.basic.out.ncStopBlockEnd	LBP_Chan*.A_NCStopBlock	DB21, ...DBX7.2
<Chan>.basic.out.ncStop	LBP_Chan*.A_NCStop	DB21, ...DBX7.3
<Chan>.basic.out.ncStopAxesAndSpindles	LBP_Chan*.A_NCStopASp	DB21, ...DBX7.4
<Chan>.basic.out.reset	LBP_Chan*.A_Reset	DB21, ...DBX7.7
<Chan>.basic.out.disableFeed	LBP_Chan*.A_FDdisable	DB21, ...DBX6.0
<Chan>.basic.out.disableReadIn	LBP_Chan*.A_RIdisable	DB21, ...DBX6.1
<Chan>.basic.out.deleteDistanceToGo	LBP_Chan*.A_DeleteDTG	DB21, ...DBX6.2
<Chan>.geoAxis1.out.disableFeed	LBP_Chan*.FDStop	DB21, ...DBX12.3, DBX14.3, DBX16.3
<Chan>.basic.out.pathFeedrateOvrFactor	LBP_Chan*.A_FD_OR	DB21, ...DBB4
<Chan>.basic.out.ncStopAxesAndSpindles	LBP_Chan*.A_NCStopASp	DB21, ...DBX4.3
<Chan>.auxMDynFuncs.in.m02Decoded（程序段中 M02 指令）	LBP_Chan*.MDyn[2]	DB21, ...DBX194.2
<Chan>.auxMDynFuncs.in.m30Decoded（程序段中 M30 指令）	LBP_Chan*.MDyn[30]	DB21, ...DBX197.6
<Chan>.auxMDynFuncs.in.m00Decoded（程序段中 M00 指令）	LBP_Chan*.MDyn[2]	DB21, ...DBX194.0
<Chan>.auxMDynFuncs.in.m01Decoded（程序段中 M01 指令）	LBP_Chan*.MDyn[30]	DB21, ...DBX194.1
<Axis>.basic.out.feedStopSpindleStop	LBP_Axis*.A_ProgTest	DB31, ...DBX4.3

3.5.7 通道状态

在所有运行方式下，每个通道的通道状态均在 NC/PLC 接口（<Chan>.basic.out....）上显示。

NC/PLC 接口信号

在 NC/PLC 接口上通过以下信号显示通道状态：

- <Chan>.basic.in.stateReset（“复位”）
- <Chan>.basic.in.stateInterrupted（“中断”）
- <Chan>.basic.in.stateActive（“激活”）

对接口信号的详细说明请参见参数手册，NC 变量和接口信号。

状态变化

通道状态受指令和 NC/PLC 接口信号影响。从通道状态“生效”出发，表格中显示了各个事件下的后续状态。

通道的输出状态：“生效”			
事件	通道的后续状态		
	“复位”	“中断”	“生效”
<Chan>.basic.out.reset（复位）	x		
<Chan>.basic.out.ncStop（NC 停止）		x	
<Chan>.basic.out.ncStopBlockEnd（程序段交界处 NC 停止）		x	
<Chan>.basic.out.ncStopAxesAndSpindles（进给轴和主轴 NC 停止）		x	
<Chan>.basic.out.disableReadIn（读取禁止）			x
<Chan>.basic.out.disableFeed（进给停止）			x
<Chan>.geoAxis1.out.disableFeed（进给停止，几何轴 1/2/3）			x
<Chan>.basic.out.pathFeedrateOvrFactor 进给倍率 = 0%			x
<Chan>.basic.out.ncStopAxesAndSpindles（进给停止/主轴停止）			x
<Chan>.auxMDynFuncs.in.m02Decoded（程序段中 M02 指令）	x		
<Chan>.auxMDynFuncs.in.m30Decoded（程序段中 M30 指令）	x		

通道的输出状态：“生效”			
事件	通道的后续状态		
	“复位”	“中断”	“生效”
<Chan>.auxMDynFuncs.in.m00Decoded（程序段中 M00 指令）			x
<Chan>.auxMDynFuncs.in.m01Decoded（程序段中 M01 指令）			x
<Chan>.basic.out.singleBlock（单程序段）		x	
<Chan>.basic.out.deleteDistanceToGo（删除剩余行程）			x
向 PLC 输出辅助功能，但尚未应答			x
程序中的 WAIT 指令			x

执行 NC 程序或 NC 程序段，或者在 JOG 运行方式下运行轴时为“通道状态：生效”。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.stateActive	LBP_Chan*.E_ChanActive	DB21, ...DBX35.5
<Chan>.basic.in.stateInterrupted	LBP_Chan*.E_ChanInterrupt	DB21, ...DBX35.6
<Chan>.basic.in.stateReset	LBP_Chan*.E_ChanReset	DB21, ...DBX35.7

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.singleBlock	LBP_Chan*.A_SBL	DB21, ...DBX0.4
<Chan>.basic.out.ncStopBlockEnd	LBP_Chan*.A_NCStopBlock	DB21, ...DBX7.2
<Chan>.basic.out.ncStop	LBP_Chan*.A_NCStop	DB21, ...DBX7.3
<Chan>.basic.out.ncStopAxesAndSpindles	LBP_Chan*.A_NCStopASp	DB21, ...DBX7.4
<Chan>.basic.out.reset	LBP_Chan*.A_Reset	DB21, ...DBX7.7
<Chan>.basic.out.disableFeed	LBP_Chan*.A_FDdisable	DB21, ...DBX6.0
<Chan>.basic.out.disableReadIn	LBP_Chan*.A_RIdisable	DB21, ...DBX6.1

Basic Program Plus	Basic Program	
<Chan>.basic.out.deleteDistanceToGo	LBP_Chan*.A_DeleteDTG	DB21, ...DBX6.2
<Chan>.geoAxis1.out.disableFeed	LBP_Chan*.FDStop	DB21, ...DBX12.3, DBX14.3, DBX16.3
<Chan>.basic.out.pathFeedrateOvrFactor	LBP_Chan*.A_FD_OR	DB21, ...DBB4
<Chan>.basic.out.ncStopAxesAndSpindles	LBP_Chan*.A_NCStopASp	DB21, ...DBX4.3
<Chan>.auxMDynFuncs.in.m02Decoded	LBP_Chan*.MDyn[2]	DB21, ...DBX194.2
<Chan>.auxMDynFuncs.in.m30Decoded	LBP_Chan*.MDyn[30]	DB21, ...DBX197.6
<Chan>.auxMDynFuncs.in.m00Decoded（程序段中 M00 指令）	LBP_Chan*.MDyn[2]	DB21, ...DBX194.0
<Chan>.auxMDynFuncs.in.m01Decoded（程序段中 M01 指令）	LBP_Chan*.MDyn[30]	DB21, ...DBX194.1
<Axis>.basic.out.feedStopSpindleStop	LBP_Axis*.A_ProgTest	DB31, ...DBX4.3

3.5.8 对操作和程序动作的响应

状态过渡

下表列出了在某些操作或程序动作后所产生的通道和程序的状态。

表格左侧列出了通道和在通道中选择的程序的不同状态以及生效的运行方式。

表格右侧列出了操作/程序动作和后续状态。

状态	通道状态			程序状态					运行方式			操作或程序动作 => 后续状态
	R	U	A	R	U	S	W	A	A	M	J	
1		x						x	x			复位 ⇒ 4
2		x						x		x		复位 ⇒ 5
3		x						x			x	复位 ⇒ 6
4	x			x					x			NC 启动 ⇒ 13；运行方式切换 ⇒ 5 或 6
5	x			x						x		NC 启动 ⇒ 14；运行方式切换 ⇒ 4 或 6
6	x			x							x	NC 启动 ⇒ 15；运行方式切换 ⇒ 4 或 5
7		x		x						x		NC 启动 ⇒ 14
8		x		x							x	NC 启动 ⇒ 15
9		x			x				x			NC 启动 ⇒ 13；运行方式切换 ⇒ 10 或 11

状态	通道状态			程序状态					运行方式			操作或程序动作 => 后续状态
	R	U	A	R	U	S	W	A	A	M	J	
10		x			x					x		NC 启动 ⇒ 16；运行方式切换 ⇒ 9 或 11
11		x			x						x	方向键 ⇒ 17；运行方式切换 ⇒ 9 或 10
12		x				x			x			NC 启动 ⇒ 13；运行方式切换 ⇒ 10 或 11
13			x					x	x			NC 停止 ⇒ 12
14			x	x						x		NC 停止 ⇒ 7；在程序段末尾 ⇒ 5
15			x	x							x	NC 停止 ⇒ 8；在 JOG 末尾 ⇒ 6
16			x		x					x		NC 停止 ⇒ 10；在程序段末尾 ⇒ 10
17			x		x						x	NC 停止 ⇒ 11；在 JOG 末尾 ⇒ 11
18			x				x		x			复位 ⇒ 4；等待其他通道 ⇒ 18

通道状态

R → 终止

U → 中断

A → 正在运行

程序状态

R → 终止

U → 中断

S → 停止

W → 等待

A → 正在运行

运行方式

A → AUTO

M → MDA

J → JOG

3.5.9 程序运行的时间图示例

程序代码

```
N10 G01 G90 X100 M3 S1000 F1000 M170
N20 M0
```

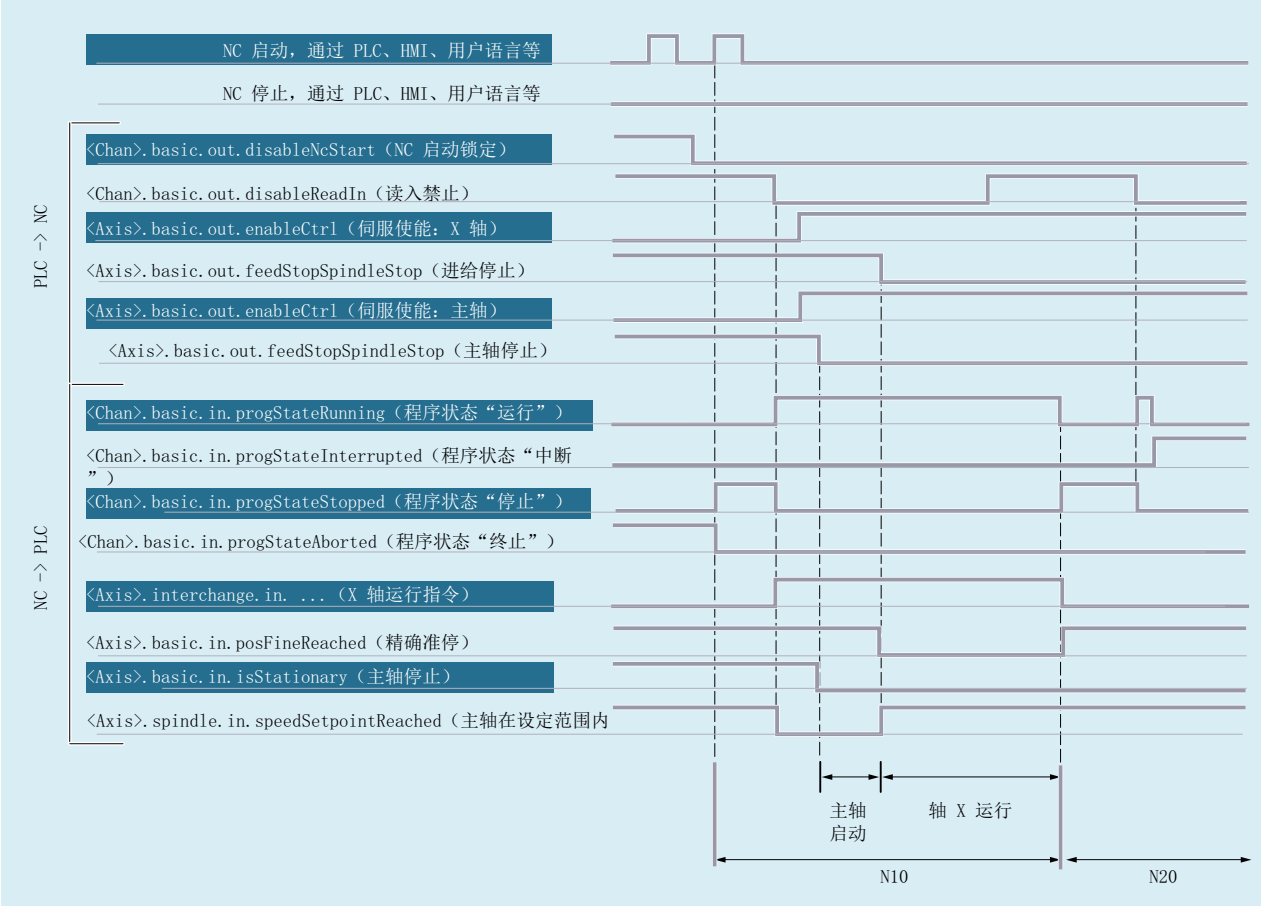


图 3-1　程序期间的信号特性曲线

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.progStateRunning	LBP_Chan*.E_ProgRunning	DB21, ...DBX35.0
<Chan>.basic.in.progStateStopped	LBP_Chan*.E_ProgStop	DB21, ...DBX35.2
<Chan>.basic.in.progStateInterrupted	LBP_Chan*.E_ProgInterrupt	DB21, ...DBX35.3
<Chan>.basic.in.progStateAborted	LBP_Chan*.E_ProgAborted	DB21, ...DBX35.4
<Axis>.basic.in.posFineReached	LBP_Axis*.E_ExactFine	DB31, ...DBX60.7
<Axis>.basic.in.isStationary	LBP_Axis*.E_Stat	DB31, ...DBX61.4
<Axis>.spindle.in.speedSetpointReached	LBP_Axis*.E_SetRange	DB31, ...DBX83.5

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.disableReadIn	LBP_Chan*.A_RIdisable	DB11.DBX6.1
<Axis>.basic.out.enableCtrl	LBP_Axis*.A_ContrEnable	DB31, ...DBX2.1
<Axis>.basic.out.feedStopSpindleStop	LBP_Axis*.A_FDSpStop	DB31, ...DBX4.3
<Chan>.basic.out.ncStart	LBP_Chan*.A_NCStart	DB21, ...DBX7.1

3.5.10 程序跳转

3.5.10.1 跳回至程序开始处（GOTOS）

功能

通过“跳回至程序开始处”功能可从零件程序跳转回程序开始处。之后该程序会被重新执行。

与同样可用于实现零件程序的重新执行的“程序跳转至跳转标记”功能相比，“跳回至程序开始处”功能具有以下优势：

- 不需要在程序开始处编写跳转标记。
- 程序重启可通过 NC/PLC 接口信号控制：
  `<Chan>.basic.out.enableGoToStart` （控制程序层级）
- 可在程序重启时将程序运行时间的定时器复位为“0”。
- 可在程序重启时将工件计数的定时器增加“1”。

应用示例

此功能适用于以下情形：需要通过自动程序重启加工后面的工件，例如对于配备棒式进料器/换料器的车床。

NC/PLC 接口信号

仅当以下 NC/PLC 接口信号置位时，才进行跳回：

`<Chan>.basic.out.enableGoToStart`（控制程序层级）= **1**

PLC 信号 NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.out.enableGoToStart	LBP_Chan*.A_ProgJump	DB21, ...DBX384.0

参数设置

程序运行时间

所选 NC 程序的运行时间保存在系统变量 $AC_CYCLE_TIME 中。启动新的程序时，该系统变量会自动复位为"0"（参见"程序运行时间 (页 283)"章节）

通过设置以下机床数据，可在通过"跳回至程序开始处"功能重启程序时也将系统变量 $AC_CYCLE_TIME 复位为"0"：

MD27860 $MC_PROCESSTIMER_MODE，位 8 = <值>（激活程序运行时间测量）

位	值	含义
8	0	"跳回至程序开始处"功能**不会**使 $AC_CYCLE_TIME 复位为"0"。
	1	"跳回至程序开始处"功能会使 $AC_CYCLE_TIME 复位为"0"。

说明

为了使位 8 的设置生效，当前程序运行时间的测量必须生效（MD27860 位 1 = 1）。

工件计数

在到达零件程序结束（M02 / M30）后，激活的工件计数器（$AC_TOTAL_PARTS / $AC_ACTUAL_PARTS / $AC_SPECIAL_PARTS）的数值加"1"（参见"工件计数器 (页 291)"章节）。

通过设置以下机床数据，可在通过"跳回至程序开始处"功能重启程序时也将激活的工件计数器向上计数：

MD27880 $MC_PART_COUNTER，位 <n> = <值>（激活工件计数器）

位	值	含义：通过"跳回至程序开始处"功能重启程序时，工件计数器：
7	0	$AC_**TOTAL**_PARTS **未**向上计数。
	1	$AC_**TOTAL**_PARTS 向上计数。
11	0	$AC_**ACTUAL**_PARTS **未**向上计数。
	1	$AC_**ACTUAL**_PARTS 向上计数。

位	值	含义：通过“跳回至程序开始处”功能重启程序时，工件计数器：
15	0	$AC_**SPECIAL**_PARTS 未向上计数。
	1	$AC_**SPECIAL**_PARTS 向上计数。

编程

句法

GOTOS

含义

GOTOS：	跳回到当前程序段的开头	
	预处理停止：	是
	有效性：	逐段式

示例

编程	注释
N10 ...	；程序开始
...	
IF ...	
N100 GOTOS	跳回到程序开头 (N10)
ENDIF	
...	
RET	

3.5.11 程序部分重复

3.5.11.1 编程

程序部分重复是指在一个 NC 程序中，可以重复已经编写的程序部分。

需要重复的程序行或程序段落带有跳转标记（标签）。

说明

跳转标记（标签）

跳转标记总是位于一个程序段的起始处。如果有程序号，则跳转标记紧跟在程序段号之后。

跳转标记名称有下列规定：

- 字符数：
  - 至少 2 个
  - 最多 32 个
- 允许使用的字符有：
  - 字母
  - 数字
  - 下划线
- 开始的两个字符必须是字母或者下划线。
- 在跳转标记名之后为一个冒号（“：”）。

句法

1. REPEATB：重复单个程序行

```
<跳转标记>: ...
...
REPEATB <跳转标记> P=<n>
```

2. REPEAT + 跳转标记：重复跳转标记和 REPEAT 指令之间的程序段落

```
<跳转标记>: ...
...
REPEAT <跳转标记> P=<n>
```

3. REPEAT + 跳转标记_1 + 跳转标记_2：重复两个跳转标记间的段落

```
<开始-跳转标记>: ...
...
<结束-跳转标记>: ...
...
REPEAT <起始跳转标记> <结束跳转标记> P=<n>
```

说明

REPEAT 指令不能被括在这两个跳转标记之间。如果在 REPEAT 指令前找到了<起始跳转标记>，但在 REPEAT 指令前没有找到<结束跳转标记>，则重复<起始跳转标记>和 REPEAT 指令之间的程序段落。

4. REPEAT + 跳转标记 + ENDLABEL：重复跳转标记和 ENDLABEL 间的段落：

```
<跳转标记>: ...
...
ENDLABEL: ...
...
REPEAT <跳转标记> P=<n>
```

说明

REPEAT 指令不能被括在<跳转标记>和 ENDLABEL 之间。如果在 REPEAT 指令前找到了<跳转标记>，但在 REPEAT 指令前没有找到 ENDLABEL，则重复<跳转标记>和 REPEAT 指令之间的程序段落。

含义

REPEATB:	重复程序行的指令
REPEAT:	重复程序段落的指令
<跳转标记>:	<跳转标记>标出了： • REPEATB：需要重复的程序行 • REPEAT：需要重复的程序段落的开始
	标有<跳转标记>的程序行可以位于 REPEAT-/REPEATB 的前面或后面。首先在向程序起始的方向搜索。如果在这个方向没有找到跳转标记，则向程序末尾方向搜索。 **例外：** 如果需要重复跳转标记和 REPEAT 指令之间的程序段落（参见句法下的第 2 点），带有<跳转标记>的程序行必须位于 REPEAT 指令**之前**，因为此时**只**向程序起始的方向搜索。 如果带<跳转标记>的程序行中还有其它的指令，在每次重复时都会重新执行这些指令。

ENDLABEL:	标出需要重复的程序段落结尾的关键字 如果带 ENDLABEL 的程序行中还有其它的指令，在每次重复时都会重新执行这些指令。 在程序中可以多次使用 ENDLABEL。
P:	指定重复数量的地址 **提示：** 如果没有指定 P=<n>，则程序部分仅重复一次。
<n>:	重复次数
	类型： INT
	程序部分会重复<n>次。在重复最后一次之后，继续执行 REPEAT-/REPEATB 指令之后的程序行。

示例

示例 1：重复单个程序行

程序代码	注释
N10 POSITION1:X10 Y20	
N20 POSITION2:CYCLE(0,,9,8)	；位置循环
N30 ...	
N40 REPEATB POSITION1 P=5	；执行程序段 N10 五次。
N50 REPEATB POSITION2	；执行程序段 N20 一次。
N60 ...	
N70 M30	

示例 2：重复跳转标记和 REPEAT 指令之间的程序段落

程序代码	注释
N5 R10=15	
N10 Begin:R10=R10+1	；程序段开头
N20 Z=10-R10	
N30 G1 X=R10 F200	
N40 Y=R10	
N50 X=-R10	
N60 Y=-R10	
N70 Z=10+R10	
N80 REPEAT BEGIN P=4	；执行 N10 到 N70 程序部分四次。
N90 Z10	
N100 M30	

示例 3：重复两个跳转标记间的段落

程序代码	注释
N5 R10=15	
N10 Begin:R10=R10+1	; 程序段开头
N20 Z=10-R10	
N30 G1 X=R10 F200	
N40 Y=R10	
N50 X=-R10	
N60 Y=-R10	
N70 END:Z=10	; 程序段结尾
N80 Z10	
N90 CYCLE(10,20,30)	
N100 REPEAT BEGIN END P=3	; 执行 N10 到 N70 程序部分三次。
N110 Z10	
N120 M30	

示例 4：重复跳转标记和 ENDLABEL 间的段落

程序代码	注释
N10 G1 F300 Z-10	
N20 BEGIN1:	; 程序段 1 的开头
N30 X10	
N40 Y10	
N50 BEGIN2:	; 程序段 2 的开头
N60 X20	
N70 Y30	
N80 ENDLABEL:Z10	
N90 X0 Y0 Z0	
N100 Z-10	
N110 BEGIN3:X20	; 程序段 3 的开头
N120 Y30	
N130 REPEAT BEGIN3 P=3	; 执行 N110 到 N120 程序部分三次。
N140 REPEAT BEGIN2 P=2	; 执行 N50 到 N80 之间的程序部分两次。
N150 M100	
N160 REPEAT BEGIN1 P=2	; 执行 N20 到 N80 之间的程序部分两次。
N170 Z10	
N180 X0 Y0	
N190 M30	

示例 5：铣削加工、采用不同的工艺加工钻孔位置

程序代码	注释
N10 ZENTRIERBOHRER()	; 换上定中钻头。
N20 POS_1:	; 程序段 1 的开头；钻孔位置 1
N30 X1 Y1	

程序代码	注释
N40 X2	
N50 Y2	
N60 X3 Y3	
N70 ENDLABEL:	
N80 POS_2:	; 程序段 2 的开头；钻孔位置 2
N90 X10 Y5	
N100 X9 Y-5	
N110 X3 Y3	
N120 ENDLABEL:	; 程序段 1 和 2 的结尾
N130 BOHRER()	; 钻削循环
N140 GEWINDE(6)	; 螺纹循环
N150 REPEAT POS_1	; 重复程序部分一次，自 POS_1 到 ENDLABEL。
N160 BOHRER()	; 钻削循环
N170 GEWINDE(8)	; 螺纹循环
N180 REPEAT POS_2	; 重复程序部分一次，自 POS_2 到 ENDLABEL。
N190 M30	

其它信息

- 程序部分重复可以嵌套调用。每次调用占用一个子程序级。
- 如果在执行程序重复过程中编程了 M17 或者 RET，则程序重复被停止。程序接着从 REPEAT 指令行之后的语句开始运行。
- 在当前的程序显示中，程序重复部分作为单独的子程序级显示。
- 如果在执行程序部分重复过程中取消该级别，则在调用程序部分执行之后，继续加工该程序。
  示例：

程序代码	注释
N5 R10=15	
N10 BEGIN:R10=R10+1	; 宽度
N20 Z=10-R10	
N30 G1 X=R10 F200	
N40 Y=R10	; 级别取消
N50 X=-R10	
N60 Y=-R10	
N70 END:Z10	
N80 Z10	
N90 CYCLE(10,20,30)	
N100 REPEAT BEGIN END P=3	
N120 Z10	; 继续程序加工。
N130 M30	

- 控制结构和程序部分重复可以组合使用。但是，两者之间不得产生重叠。一个程序部分重复应该位于一个控制结构分支之内，或者一个控制结构位于一个程序部分重复部分之内。
- 如果跳转和程序重复部分交织在一起，则程序段按次序执行。比如说，程序重复部分有一个跳跃，则一直进行加工，直至找到编程的程序结束部分。
  示例：

程序代码

```
N10 G1 F300 Z-10
N20 BEGIN1:
N30 X=10
N40 Y=10
N50 GOTOF BEGIN2
N60 ENDLABEL:
N70 BEGIN2:
N80 X20
N90 Y30
N100 ENDLABEL:Z10
N110 X0 Y0 Z0
N120 Z-10
N130 REPEAT BEGIN1 P=2
N140 Z10
N150 X0 Y0
N160 M30
```

说明

REPEAT 指令应位于运行程序段之后。

3.5.12 事件控制的程序调用(PROG_EVENT)

3.5.12.1 功能

功能

在出现一个所选事件时，“事件控制的程序调用”(PROG_EVENT) 功能会在 NC 中启动一个 NC 程序（PROG_EVENT 程序）。

应用示例

系统或用户变量的功能或初始化设置

事件

触发事件通过机床数据 MD20108 $MC_PROG_EVENT_MASK 选择（参见章节“参数设置 (页 83)”）。

PROG_EVENT 程序

通过机床数据 MD11620 $MN_PROG_EVENT_NAME 设置 PROG_EVENT 程序的名称（参见章节“参数设置 (页 83)”）。

PROG_EVENT 程序在发生对应事件的通道中执行。

PROG_EVENT 程序以最低的优先级执行，因此可由用户 ASUB 中断。

执行顺序

通过事件激活时的执行顺序：程序开始

初始状态：

通道：	复位状态
运行方式：	自动模式（可选：溢出转存生效）
	MDI
	TEACH IN

1. 在通道中执行程序开始
2. 初始化步骤，包含对以下机床数据的检测：
   - MD20112 $MC_START_MODE_MASK
3. 作为子程序隐含调用 PROG_EVENT 程序
4. 处理主程序的数据部分
5. 处理主程序的程序部分

通过事件激活时的执行顺序：程序结束

初始状态

通道：	激活
运行方式：	自动模式（可选：溢出转存生效）
	MDI
	TEACH IN

1. 通道中已执行的程序中将换为程序结束程序段。
2. 执行程序结束，检测以下机床数据：
   - MD20110 $MC_RESET_MODE_MASK
   - MD20150 $MC_GCODE_RESET_VALUES
   - MD20152 $MC_GCODE_RESET_MODE
3. 作为 ASUB 隐含调用 PROG_EVENT 程序
4. 在通道中执行复位，检测以下机床数据：
   - MD20110 $MC_RESET_MODE_MASK
   - MD20150 $MC_GCODE_RESET_VALUES
   - MD20152 $MC_GCODE_RESET_MODE

通过事件激活时的顺序：通道复位

初始状态：

通道： 任意

运行方式： 任意

1. 控制系统激活复位步骤，包含对以下机床数据的检测：
   - MD20110 $MC_RESET_MODE_MASK
   - MD20150 $MC_GCODE_RESET_VALUES
   - MD20152 $MC_GCODE_RESET_MODE
2. 作为 ASUB 隐含调用 PROG_EVENT 程序
3. 控制系统激活复位步骤，包含对以下机床数据的检测：
   - MD20110 $MC_RESET_MODE_MASK
   - MD20150 $MC_GCODE_RESET_VALUES
   - MD20152 $MC_GCODE_RESET_MODE

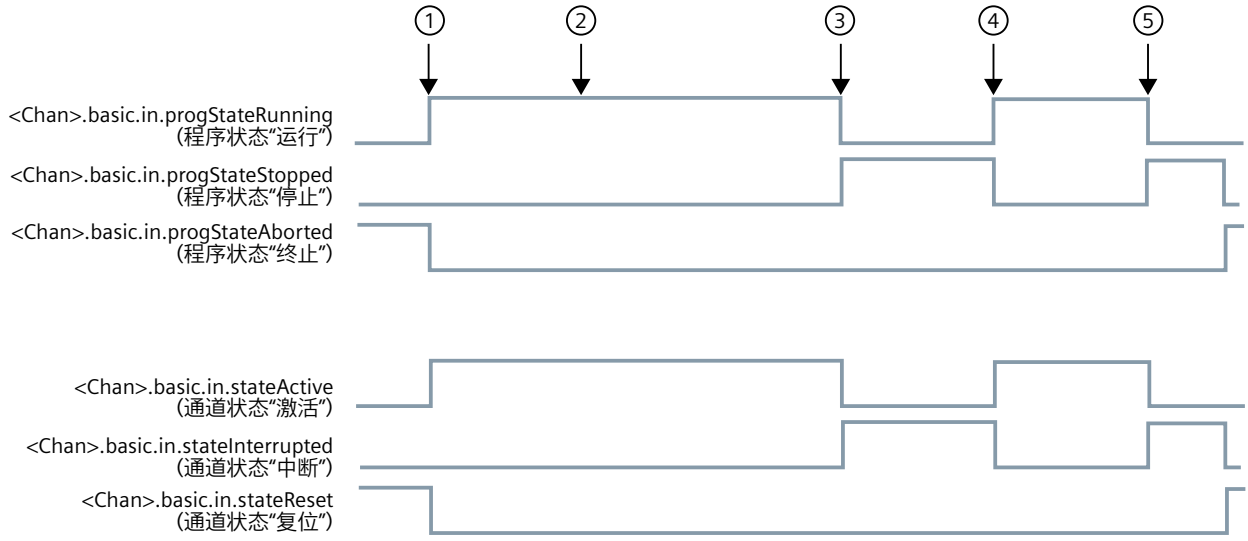
通过事件激活时的顺序：NC 启动

1. 控制系统在启动后激活复位步骤，包含对以下机床数据的检测：
   - MD20110 $MC_RESET_MODE_MASK
   - MD20150 $MC_GCODE_RESET_VALUES
   - MD20152 $MC_GCODE_RESET_MODE
2. 作为 ASUB 隐含调用 PROG_EVENT 程序
3. 控制系统激活复位步骤，包含对以下机床数据的检测：
   - MD20110 $MC_RESET_MODE_MASK
   - MD20150 $MC_GCODE_RESET_VALUES
   - MD20152 $MC_GCODE_RESET_MODE

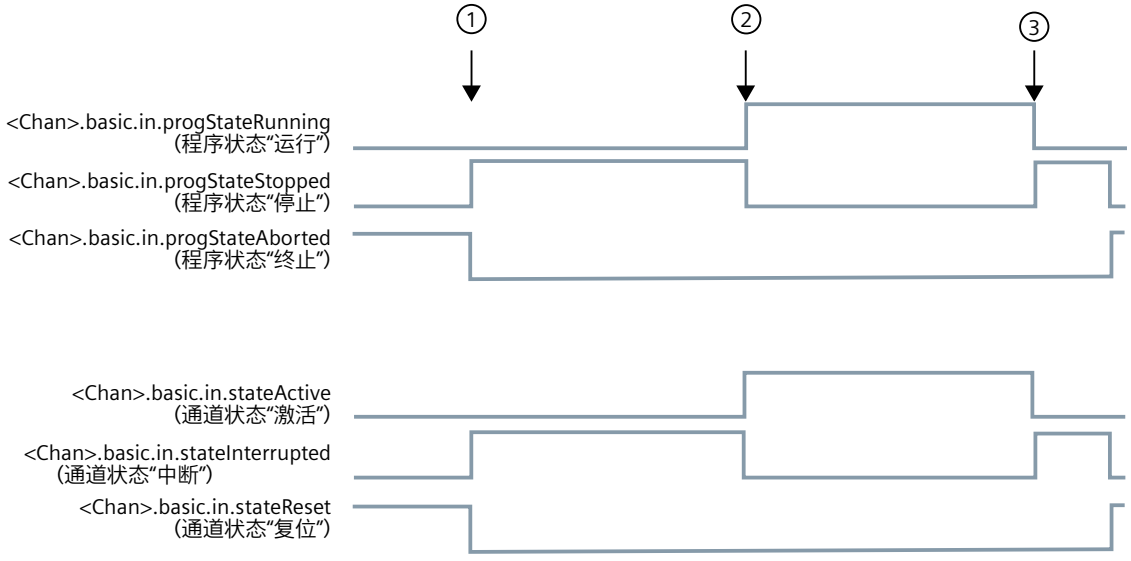
NC/PLC 接口信号：“程序状态”和“通道状态”的变化过程

下图展示了在事件控制的程序调用期间 NC/PLC 接口信号“程序状态”和“通道状态”的变化过程：

由程序开始和程序结束激活的信号变化图



由通道复位激活的信号变化图



NC/PLC 接口信号：<Chan>.basic.in.progStateAborted（程序状态“终止”）和 <Chan>.basic.in.stateReset（通道状态复位）

- 当激活的由 PROG_EVENT 程序再次结束时，接口信号才被置位。
- 在以下事件之间时接口信号不会被置位：
  - 程序结束和 ROG_EVENT 程序开始之间
  - 通道复位和 ROG_EVENT 程序开始之间

NC/PLC 接口信号：触发事件

触发事件通过接口信号 <Chan>.progEvent.in.ncStartActive、
<Chan>.progEvent.in.partProgramEndActive、
<Chan>.progEvent.in.opResetActive、
<Chan>.progEvent.in.powerOnActive
和 <Chan>.progEvent.in.blockSearchActive 显示：

位	值	事件
0	1	程序从“复位”通道状态开始
1	1	程序结束
2	1	通道复位
3	1	NC 启动
4	1	程序段搜索后的第 1 次程序启动（参见“程序段搜索后 ASUB 的自动启动 (页 155)”）

如果 PROG_EVENT 程序结束或被通道复位终止，接口信号会再次被清除。

接口信号至少应在 PLC 循环期间存在。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.progStateRunning	LBP_Chan*.E_ProgRunning	DB21, ...DBX35.0
<Chan>.basic.in.progStateStopped	LBP_Chan*.E_ProgStop	DB21, ...DBX35.2
<Chan>.basic.in.progStateInterrupted	LBP_Chan*.E_ProgInterrupt	DB21, ...DBX35.3
<Chan>.basic.in.progStateAborted	LBP_Chan*.E_ProgAborted	DB21, ...DBX35.4
<Chan>.basic.in.stateActive	LBP_Chan*.E_ChanActive	DB21, ...DBX35.5

Basic Program Plus	Basic Program	
<Chan>.basic.in.stateInterrupted	LBP_Chan*.E_ChanInterrupt	DB21, ...DBX35.6
<Chan>.basic.in.stateReset	LBP_Chan*.E_ChanReset	DB21, ...DBX35.7
<Chan>.progEvent.in.ncStartActive	LBP_Chan*.E_ProgEvent_Start	DB21, ...DBX376.0
<Chan>.progEvent.in.partProgramEndActive	LBP_Chan*.E_ProgEvent_M30	DB21, ...DBX376.1
<Chan>.progEvent.in.opResetActive	LBP_Chan*.E_ProgEvent_Reset	DB21, ...DBX376.2
<Chan>.progEvent.in.powerOnActive	LBP_Chan*.E_ProgEvent_PowerOn	DB21, ...DBX376.3
<Chan>.progEvent.in.blockSearchActive	LBP_Chan*.E_ProgEvent_SearchRun	DB21, ...DBX376.4

3.5.12.2 参数设置

事件选择

通过设置针对具体通道的以下机床数据，可定义用于激活 PROG_EVENT 程序的事件：

MD20108 $MC_PROG_EVENT_MASK，位 <n> = 1

位	<值>	含义：事件
0	1	零件程序开始
1	1	零件程序结束
2	1	通道复位
3	1	启动

说明

MD20108 $MC_PROG_EVENT_MASK 不在仿真中进行检测。

PROG_EVENT 程序

PROG_EVENT 程序（预设：_N_PROG_EVENT_SPF）必须已载入并使能。

缺省设置

事件发生时默认会执行用户程序_N_CMA_DIR/**_N_PROG_EVENT_SPF**。

PROG_EVENT 程序必须已载入并使能。

用户专用设置

如要在事件发生时执行另一个 PROG_EVENT 程序，则应在以下机床数据中输入 NC 程序的名称：

MD11620 $MN_PROG_EVENT_NAME = <程序名称>

查找路径

PROG_EVENT 程序必须位于以下其中一个循环目录中。设置的事件出现时，系统会按以下路径搜索：

1. /_N_CUS_DIR/（用户循环）
2. /_N_CMA_DIR/（制造商循环）
3. /_N_CST_DIR/ (标准循环)

最先找到的带有在 MD11620 $MN_PROG_EVENT_NAME 中指定的名称的程序将被执行。

说明

- 系统会像对子程序名称一样对设定的名称进行句法检查，即其前两个字符必须为字母或下划线（非数字）。若未为程序名称设定前缀（_N_）和后缀（_SPF），则会自动添加。
- 和适用于循环的情况类似，保护机制对该程序生效（读写的保护级等）。

启动用户 ASUB 时的特性

通过以下机床数据，可针对通道对从通道复位状态启动用户 ASUB 时“事件控制的程序调用”的特性进行设置：

MD20109 $MC_PROG_EVENT_MASK_PROPERTIES，位 0 = <值>

位	值	含义
0	0	出现通过 MD20108 设置的事件时，系统会启动 PROG_EVENT 程序。
	1	出现通过 MD20108 设置的事件时，系统不会启动 PROG_EVENT 程序。

启用单程序段执行时的特性

通过以下机床数据，可针对通道为每个触发事件设置启用单程序段执行时“事件控制的程序调用”功能的特性：

MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK，位 <n> = <值>

位	值	含义：在 PROG_EVENT 程序中，以下事件下系统会执行单程序段：
0	0	零件程序开始：**生效**
	1	零件程序开始：**抑制**
1	0	零件程序结束：**生效**
	1	零件程序结束：**抑制**
2	0	通道复位：**生效**
	1	通道复位：**抑制**
3	0	启动：**生效**
	1	启动：**抑制**

单程序段模式被抑制时，系统会继续执行 PROG_EVENT 程序。

说明

- MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK 对所有单程序段执行类型生效。
- 一般情况下可通过以下配置取消 PROG_EVENT 程序中的单程序段执行：
  MD10702 $MN_IGNORE_SINGLEBLOCK_MASK，位 0 = 1（忽略单程序段停止）
  此时 MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK 中的设置将无效。

读取禁止生效时的特性

通过以下机床数据，可针对通道为每个触发事件设置读取禁止生效（<Chan>.basic.out.disableReadIn = 1）时“事件控制的程序调用”功能的特性：

MD20107 $MC_PROG_EVENT_IGN_INHIBIT，位 <n> = <值>

位	值	含义：在 PROG_EVENT 程序中，以下事件下系统会执行读取禁止：
0	0	零件程序开始：**生效** 如果 PROG_EVENT 程序被指令 RET 结束，则**不**会再生成可执行的程序段
	1	零件程序开始：**抑制** 如果 PROG_EVENT 程序被指令 RET 结束，则会再生成**一个**与 M17 类似的可执行程序段。
1	0	零件程序结束：**生效**
	1	零件程序结束：**抑制**
2	0	通道复位：**生效**
	1	通道复位：**抑制**

位	值	含义：在 PROG_EVENT 程序中，以下事件下系统会执行读取禁止：
3	0	启动：**生效**
	1	启动：**抑制**

抑制程序状态和通道状态的显示更新

为了避免 HMI 操作界面中程序状态和通道状态的显示闪动，对于通常很短的 PROG_EVENT 程序执行可抑制显示更新。这样一来界面将显示激活 PROG_EVENT 程序前的程序状态和通道状态。

MD20192 $MC_PROG_EVENT_IGN_PROG_STATE，位 <n> = <值>

位	值	含义：在执行 PROG_EVENT 程序时，以下事件下系统会进行程序状态和通道状态的显示更新：
1	0	零件程序结束：**不抑制**
	1	零件程序结束：**抑制**
2	0	通道复位：**不抑制**
	1	通道复位：**抑制**
3	0	启动：**不抑制**
	1	启动：**抑制**

说明

系统变量 $AC_STAT 和 $AC_PROG 不受此功能影响，即事件控制用户程序运行时 $AC_STAT 将设置为“生效”， $AC_PROG 将设置为“运行”。

以下 NC/PLC 接口信号也不受影响：

```
<Chan>.basic.in.progStateRunning;
<Chan>.basic.in.progStateWaiting;
<Chan>.basic.in.progStateStopped;
<Chan>.basic.in.progStateInterrupted;
<Chan>.basic.in.progStateAborted; <Chan>.basic.in.stateActive;
<Chan>.basic.in.stateInterrupted; <Chan>.basic.in.stateReset
```

<Chan>.basic.out.ncStopBlockEnd、<Chan>.basic.out.ncStop 和 <Chan>.basic.out.ncStopAxesAndSpindles 中的变量

通过以下机床数据，可针对通道为零件程序结束、通道复位和引导启动这些触发事件设置 NC 停止（即“NC 停止”、“程序段交界处 NC 停止”和“进给轴和主轴 NC 停止”）时“事件控制的程序调用”功能的特性：

MD20193 $MC_PROG_EVENT_IGN_STOP，位 <n> = <值>

位	值	含义：PROG_EVENT 程序在 NC 停止和事件中
1	0	零件程序结束：**停止/避免**
	1	零件程序结束：**全部执行**
2	0	通道复位：**停止/避免**
	1	通道复位：**全部执行**
3	0	启动：**停止/避免**
	1	启动：**全部执行**

应用示例

借此，操作员可在通道复位或引导启动时按下 NC 停止键忽略执行 PROG_EVENT 程序时产生的接口信号 <Chan>.basic.out.ncStop（NC 停止）的脉冲沿切换，并避免机床上出现意外停止特性。

说明

通过 MD20193 设置的特性不支持在 PROG_EVENT 程序中编写 DELAYFSTON/DELAYFSTOF。执行 DELAYFSTON 指令前的 NC 停止仍会引起中断。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.progStateRunning	LBP_Chan*.E_ProgRunning	DB21, ...DBX35.0
<Chan>.basic.in.progStateWaiting	LBP_Chan*.E_ProgWait	DB21, ...DBX35.1
<Chan>.basic.in.progStateStopped	LBP_Chan*.E_ProgStop	DB21, ...DBX35.2
<Chan>.basic.in.progStateInterrupted	LBP_Chan*.E_ProgInterrupt	DB21, ...DBX35.3
<Chan>.basic.in.progStateAborted	LBP_Chan*.E_ProgAborted	DB21, ...DBX35.4
<Chan>.basic.in.stateActive	LBP_Chan*.E_ChanActive	DB21, ...DBX35.5
<Chan>.basic.in.stateInterrupted	LBP_Chan*.E_ChanInterrupt	DB21, ...DBX35.6
<Chan>.basic.in.stateReset	LBP_Chan*.E_ChanReset	DB21, ...DBX35.7

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.disableReadIn	LBP_Chan*.A_RIdisable	DB21, ...DBX6.1
<Chan>.basic.out.ncStopBlockEnd	LBP_Chan*.A_NCStopBlock	DB21, ...DBX7.2
<Chan>.basic.out.ncStop	LBP_Chan*.A_NCStop	DB21, ...DBX7.3
<Chan>.basic.out.ncStopAxesAndSpindles	LBP_Chan*.A_NCStopASp	DB21, ...DBX7.4

3.5.12.3 编程

PROG_EVENT 程序

程序结束

若需通过零件程序开始激活用户程序，则须注意以下几点：

- 该用户程序必须通过 M17 或 RET 结束。
- 不允许通过 REPOS 指令进行跳回。

程序段显示

可以通过 PROC 指令中 DISPLOF 属性抑制当前程序段中的显示。

与 PLC 用户程序的通讯

通过写入 PROG_EVENT 程序的用户 M 功能可知道 PLC 用户程序的信息，例如：PROG_EVENT 程序的执行状态。

系统变量

查询触发事件

可在 PROG_EVENT 程序中通过以下系统变量查询触发事件：

<值> = $P_PROG_EVENT（事件控制的程序调用生效）

值	含义：激活，通过
1	零件程序开始
2	零件程序结束
3	通道复位

值	含义：激活，通过
4	**启动**
5	**程序段搜索**后在输出最后一个动作程序段后激活（参见“程序段搜索后自动启动 ASUB (页 155)”）

查询当前通道

可通过以下机床数据确定执行 PROG_EVENT 程序的通道：

<值> = $P_CHANNO（查询当前通道编号）

说明

PROG_EVENT 程序在发生对应触发事件的通道中执行。
控制系统启动事件在所有通道中同步发生。

3.5.12.4 前提条件

急停/报警

通道复位时或系统启动后若存在急停或 BAG/NC 专用报警，那么仅当在所有相关通道中进行急停应答或故障应答后才会执行 PROG_EVENT 程序。

说明

控制系统启动事件在所有通道中同步发生。

3.5.12.5 示例

示例 1：在所有事件中调用 PROG_EVENT 程序

参数设置

MD20108 $MC_PROG_EVENT_MASK = 'H0F'

在以下事件时调用 _N_PROG_EVENT_SPF：

- 零件程序开始
- 零件程序结束
- 通道复位
- 控制系统启动

编程

程序代码	注释
PROC PROG_EVENT DISPLOF	
; 零件程序开始时执行	
IF ($P_PROG_EVENT==1)	
MY_GUD_VAR=0	; 初始化 GUD 变量。
RET	
ENDIF	
; 零件程序结束和操作面板复位时执行	
IF ($P_PROG_EVENT==2) OR ($P_PROG_EVENT==3)	
DRFOF	; 取消 DRF 偏移。
IF $MC_CHAN_NAME=="CHAN1"	
CANCEL(2)	; 删除模态同步动作 2。
ENDIF	
RET	
ENDIF	
; 控制系统启动时执行	
IF ($P_PROG_EVENT==4)	
IF $MC_CHAN_NAME=="CHAN1"	
IDS=1 EVERY $A_INA[1]>5.0 DO $A_OUT[1]=1	
ENDIF	
RET	
ENDIF	
RET	

示例 2：在通道复位时调用 PROG_EVENT 程序

参数设置

MD20108 $MC_PROG_EVENT_MASK = 'H04'

在以下事件时调用 _N_PROG_EVENT_SPF：

- 操作面板复位

编程

程序代码	注释
PROC PROG_EVENT DISPLOF	
N10 DRFOF	; 取消 DRF 偏移
N20 M17	

示例 3：功能初始化

参数设置

机床数据布局，调试文件（_N_INITIAL_INI）的节选

程序代码	注释
...	
CHANDATA(3)	；通道 3 初始化
$MC_PROG_EVENT_IGN_INHIBIT='H01F'	
$MC_PROG_EVENT_MASK='H04'	；事件：通道复位
...	

含义

执行通道复位时，在第 3 个通道中启动 PROG_EVENT 程序 _N_CMA_DIR / _N_PROG_EVENT_SPF 并一直执行到其结束，不考虑是否启用读取禁止。

3.5.13 通过停止延迟区控制停止事件

3.5.13.1 功能

停止延迟区

通过 NC 程序中可有条件中断的区域，可调整对停止事件的响应。这样的程序区域被称为停止延迟区。

在停止延迟区内不会发生停止，进给率也不会改变。该程序区域执行完毕后，可能存在的停止才会生效。

因此具有以下优点：

- 在没有速度干扰的情况下，处理程序段。
- 在停止之后，如果用户使用 RESET 中断程序，则被中断的程序段就在受到保护的程序块后面。这样的程序段适用于作为下一个程序段搜索的搜索目标。
- 只要一个停止延迟区被加工，则以下的主运行轴不会被停止。
  - 指令轴
  - 使用 POSA 运动的定位轴

应用

示例：加工螺纹。

定义

停止延迟区是在零件程序中通过预定义程序 DELAYFSTON 和 DELAYFSTOF（参见“编程 (页 96)”）定义的。

停止事件

会引发停止的事件一览：

事件	响应
<Chan>.basic.out.reset（复位）或 <ModeGroup>.basic.out.reset（BAG 复位）	立即
NC 程序：M30	报警 16954
用于启动 ASUB 的用户中断： FC 9 或 <Nc>.digitalInputs[1].out.set（由 PLC 置位 NC 数字量输入）	延迟
<Chan>.basic.out.deleteDistanceToGo / <Axis>.basic.out.resetMovement（删除剩余行程）通道/轴专用	立即
<Chan>.basic.out.deleteSubProgRepetition（删除子程序循环次数）	延迟
<Chan>.basic.out.abortProgLevel（程序级终止）	延迟
因单程序段停止 • 在停止延迟区： NC 在停止延迟区外的第 1 个程序段处停止。 • 单程序段在停止延迟区前已生效： NC 在每个程序段交界处停止，在停止延迟区中也是如此： <Chan>.basic.out.ncStopBlockEnd（程序段交界处 NC 停止） **停止延迟区已取消！**	延迟
<ModeGroup>.basic.out.singleBlockTypeA（启用单程序段类型 A）	延迟
<ModeGroup>.basic.out.singleBlockTypeB（启用单程序段类型 B）	延迟
<Chan>.basic.out.ncStopAxesAndSpindles（NC 停止） 和 <ModeGroup>.basic.out.stopAll（进给轴和主轴 BAG 停止）	立即
<Chan>.basic.out.ncStop（程序段交界处 NC 停止） 和 <ModeGroup>.basic.out.stopAxesOnly（BAG 停止）	延迟
NC 程序：因覆盖缓存为空而停止	报警 16954
NC 程序：编程或隐性 STOPRE	报警 16954
NC 程序：M0 或 M1	报警 16954
<Chan>.basic.out.ncStopBlockEnd（程序段交界处 NC 停止）	延迟
子程序结束应总是撤销停止延迟区。	系统故障

事件	响应
NC 程序：WAITM	报警 16954
NC 程序：WAITE	报警 16954
NC 程序：INIT，通过参数 “S”	报警 16954
NC 程序：MMC (STRING, CHAR)	报警 16954
<HmiChan>.basic.in.skipBlockLevel0（激活/切换程序段隐藏）	延迟
<HmiChan>.basic.in.skipBlockLevel0（关闭程序段隐藏）	延迟
<Chan>.basic.out.dryRunFeedrate（激活 DryRun）	延迟
<Chan>.basic.out.dryRunFeedrate（关闭 DryRun）	延迟
<Chan>.basic.out.disableReadIn（读取禁止）	延迟
报警：报警配置 STOPATENDBYALARM	立即
报警：报警配置 STOPBYALARM	立即
内部：插补缓存为空时在报警后停止	立即
内部：插补缓存为空时在报警后停止	立即
NC 程序：LFON 时报警 16954	报警 16954
NC 程序：WAITMC	报警 16954
NC 程序：NEWCONF	报警 16954
NC 程序：NEWCONF	报警 16954
BTSS:PI "_N_SETUDT"	延迟
扩展的停止和退回	延迟
<Axis>.basic.out.acceptExternalWorkOffset（接收外部零点偏移）	延迟
BTSS:PI "_N_FINDST" STOPRUN	报警 16955
响应为 NOREADY 的报警	立即
<Chan>.basic.out.ncStopBlockEnd == 1（程序段交界处 NC 停止） <Chan>.basic.out.ncStop == 1（NC 停止） <Chan>.basic.out.ncStopAxesAndSpindles == 1（进给轴和主轴 NC 停止）	延迟
<Chan>.basic.out.disableReadIn == 1（读取禁止）	延迟

事件	响应
<Chan>.basic.out.skipBlockLevel0 == 1（单程序段）	延迟

响应

- **立即**
  立即停止，即便在停止延迟区中也是如此。此类停止事件称为“硬停止事件”。
- **延迟**
  停止（也包括短时间的）只在停止延迟区后产生。此类停止事件称为“软停止事件”。
- **报警 16954**
  程序将被停止，因为在停止延迟区中使用了不被允许的程序命令。
- **报警 16955**
  程序继续执行，停止延迟区中进行一个不允许的动作。
- **报警 16957**
  通过 DELAYFSTON 和 DELAYFSTOF 括起来的程序区域（停机延迟区）未能激活。因此停止均会立即生效，无延迟！在停止延迟区进行制动时总是会出现此情形，即制动过程在停止延迟区前开始，但是在停止延迟区内才结束。若以倍率 0 进入停止延迟区，则同样无法激活停止延迟区（示例：借助停止延迟区前的 G4，用户可将倍率降至 0，此时停止延迟区中的下一个程序段将以倍率 0 启动，这样便出现了报警情形）。
  提示
  MD11411 $MN_ENABLE_ALARM_MASK（激活警告）位 7 会激活此报警。

说明

某些 NC 事件只会为执行切换而短暂停止，并于随后立即重新启动。例如：可短暂停止 NC 程序以启动对应 ASUB 的中断。此类事件同样可用于停止延迟区，但是其会被推移至区域的末尾并作为“软”停止事件。

说明

如果一个“硬”的停止事件碰到“停止延迟区”，那“停止延迟区”完全不再被选择。这就是说，如果在该程序段中出现另一个任意的停止，就会立即停住。只有重新编程 (更新的 DELAYFSTON) 才可开始一个新的停止延迟区。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<ModeGroup>.basic.out.reset	LBP_ModeGroup.A_MGReset	DB11.DBX0.7
<Chan>.basic.in.progStateRunning	LBP_Chan*.E_ProgRunning	DB21, ...DBX35.0
<Chan>.basic.in.progStateWaiting	LBP_Chan*.E_ProgWait	DB21, ...DBX35.1
<Chan>.basic.in.progStateStopped	LBP_Chan*.E_ProgStop	DB21, ...DBX35.2

Basic Program Plus	Basic Program	
<Chan>.basic.in.progStateInterrupted	LBP_Chan*.E_ProgInterrupt	DB21, ...DBX35.3
<Chan>.basic.in.progStateAborted	LBP_Chan*.E_ProgAborted	DB21, ...DBX35.4

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.dryRunFeedrate	LBP_Chan*.A_DRY	DB21, ...DBX0.6
<Chan>.basic.out.skipBlockLevel0	LBP_Chan*.A_SKP0	DB21, ...DBX2.0
<Chan>.basic.out.disableReadIn	LBP_Chan*.A_RIdisable	DB21, ...DBX6.1
<Chan>.basic.out.deleteDistanceToGo	LBP_Chan*.A_DeleteDTG	DB21, ...DBX6.2
<Chan>.basic.out.deleteSubProgRepetition	LBP_Chan*.A_SP_Clear	DB21, ...DBX6.3
<Chan>.basic.out.abortProgLevel	LBP_Chan*.A_ProgAbort	DB21, ...DBX6.4
<Chan>.basic.out.ncStart	LBP_Chan*.A_NCStart	DB21, ...DBX7.1
<Chan>.basic.out.ncStopBlockEnd	LBP_Chan*.A_NCStopBlock	DB21, ...DBX7.2
<Chan>.basic.out.ncStop	LBP_Chan*.A_NCStop	DB21, ...DBX7.3
<Chan>.basic.out.ncStopAxesAndSpindles	LBP_Chan*.A_NCStopASp	DB21, ...DBX7.4
<Chan>.basic.out.reset	LBP_Chan*.A_Reset	DB21, ...DBX7.7
<Chan>.basic.out.disableFeed	LBP_Chan*.A_FDdisable	DB21, ...DBX21.7
<Axis>.basic.out.acceptExternalWorkOffset	LBP_Axis*.A_ExtZO	DB31, ...DBX3.0
<Axis>.basic.out.resetMovement	LBP_Axis*.A_DelDTGSpReset	DB31, ...DBX2.2
<ModeGroup>.basic.out.stopAxesOnly	LBP_ModeGroup.A_MGStop	DB11.DBX0.5
<ModeGroup>.basic.out.stopAll	LBP_ModeGroup.A_MGStopASp	DB11.DBX0.6
<ModeGroup>.basic.out.singleBlockTypeA	LBP_ModeGroup.A_SingleBlock_A	DB11.DBX1.7
<ModeGroup>.basic.out.singleBlockTypeB	LBP_ModeGroup.A_SingleBlock_B	DB11.DBX1.6
<HmiChan>.basic.in.skipBlockLevel0	LBP_ModeGroup.E_MMC_SKP0	DB11.DBX26.0
<Nc>.digitalInputs[1].out.set	LBP_NC.A_Set_Inp1 - 40	DB10.DBB1

3.5.13.2 参数设置

机床数据

G331/G332 时的停止特性

对于刚性攻丝（G331，G332），停止特性可如下设置：

MD11550 $MN_STOP_MODE_MASK

位	值	含义
0	0（缺省）	若 G331/G332 生效并额外编写了轨迹运动或暂停时间（G4），则会生成一个隐性停止延迟区。
	1	在以下情形下，可在 G331/G332 生效时停止： • 在连续路径运行（G64）中断时。 • 在编写位于 G33x 程序段之间的停留时间（G4）时。 必须使用 NC 指令 DELAYFSTON 和 DELAYFSTOF 来定义停止延迟区。

3.5.13.3 编程

定义停止延迟区 (DELAYFSTON, DELAYFSTOF)

停止延迟区是在零件程序中通过预定义程序 DELAYFSTON 和 DELAYFSTOF 定义的。

说明

DELAYFSTON 和 DELAYFSTOF 不可用于同步动作！

句法

```
DELAYFSTON
...
DELAYFSTOF
```

含义

DELAYFSTON:	定义一个停止延迟区的开始	
	在单独程序段中编程：	选择
DELAYFSTOF:	定义一个停止延迟区的结尾	
	在单独程序段中编程：	选择

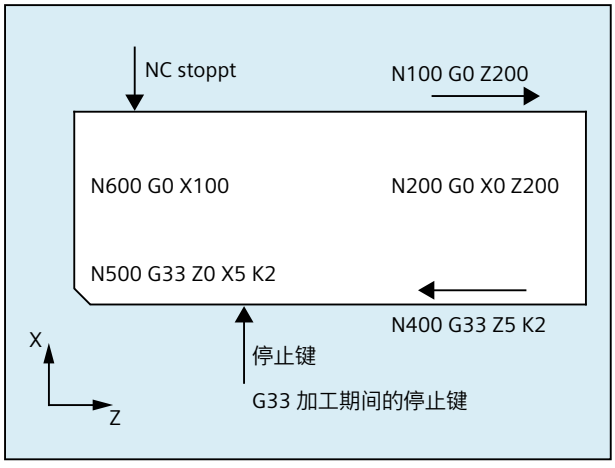
编程示例

在一个循环中重复下列程序块：

程序代码

```
...
N99 MY_LOOP:
N100 G0 Z200
N200 G0 X0 Z200
N300 DELAYFSTON
N400 G33 Z5 K2 M3 S1000
N500 G33 Z0 X5 K3
N600 G0 X100
N700 DELAYFSTOF
N800 GOTOB MY_LOOP
...
```

下图可见用户在停止延迟区中按下“停止”，NC 开始执行停止延迟区范围之外的制动过程，即在程序段 N100 中。这样 NC 就在 N100 的前端区域停住。



更多信息

子程序结束

当子程序结束时，DELAYFSTON 已经在其中被调用，DELAYFSTOF 就被隐式激活。

嵌套

如子程序 1 在停止延迟区中调用子程序 2，那么子程序 2 就是完整的停止延迟区。特别的是 DELAYFSTOF 在子程序 2 中无效。

示例：

程序代码	注释
N10010 **DELAYFSTON**	; 带 N10xxx 的程序级面 1 的程序段。
N10020 R1 = R1 + 1	
N10030 G4 F1	; 开始停止延迟区。
...	
N10040 子程序 2	
...	
...	; 子程序 2 的说明。
N20010 **DELAYFSTON**	; 无效，程序级 2 重复的开始。
...	
N20020 **DELAYFSTOF**	; 另一个程序级的结束，无效。
N20030 RET	
N10050 **DELAYFSTOF**	; 相同程序级中停止延时区的结束。
...	
N10060 R2 = R2 + 2	
N10070 G4 F1	; 结束停止延迟区。停止从现在起立即有效。

系统变量

可通过以下系统数据查询零件程序执行当前是否位于停止延迟区中：

- 在带 $P_DELAYFST 的零件程序中
- 在带 $AC_DELAYFST 的同步动作中

值	含义
0	停止延迟区未生效
1	停止延迟区生效

3.5.13.4 边界条件

重叠

如果两个停止延迟区相互交迭，一个来自 DELAYFSTON/DELAYFSTOF，另一个来自 MD11550 $MN_STOP_MODE_MASK，则会生成尽可能大的停止延迟区。

暂停时间（G4）

G4 可在停止延迟区中使用。其它会引起临时停止的零件程序指令（例如 WAITM）则不允许使用，并会触发报警 16954。

倍率

如果在停止延迟区**前**调整倍率，倍率会在停止延迟区**中**生效。

如果在停止延迟区**中**调整倍率，倍率会在停止延迟区**后**生效。

说明

倍率 = 0

如果在停止延迟区**前**将倍率置 0，则无法激活停止延迟区！

进给禁止

<Chan>.basic.out.disableFeed（进给禁止）在停止延时区中不起作用，只有在离开停止延时区后才会停住。

PLC 信号 PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.disableFeed	LBP_Chan*.A_FDdisable	DB21, ...DBX6.0

单程序段

如果在停止延迟区中激活单程序段，NC 会在停止延迟区外的第一个程序段处停止。

如果在停止延迟区前已经选择了单程序段，NC 会在每个程序段交界处停止，在停止延迟区中也是如此！**停止延迟区已取消。**

3.5.14 插补缓存的大小调整

MD28060

在零件程序运行中，通道专用插补器会从插补缓存执行准备的程序段。插补缓存中可同时保存的程序段的最大数量由存储器配置机床数据 MD28060 $MM_IPO_BUFFER_SIZE（插补缓存 DRAM 中 NC 程序段的数量）定义。在某些应用中可选择不占满该缓存，从而保证准备和插补间的较小“间隔”。

SD42990

通过设定数据 SD42990 $SC_MAX_BLOCKS_IN_IPOBUFFER（插补缓存中程序段的最大数量），可将插补缓存中的程序段数量动态限制为**小于** MD28060 $MC_MM_IPO_BUFFER_SIZE（插补缓存 DRAM 中 NC 程序段的数量）的值，最低为 2 个程序段。

设定数据 SD42990 $SC_MAX_BLOCKS_IN_IPOBUFFER 的值：

值	作用
< 0	对插补缓存无限制。 对应 MD 28060:MM_IPO_BUFFER_SIZE 激活最大的插补缓存。
或 1	允许的最小插补缓存生效，大小为 2 个程序段。
< MM_IPO_BUFFER_SIZE	插补缓存的激活上限为指定的程序段数量。
>= MM_IPO_BUFFER_SIZE	通过 MD 28060:MM_IPO_BUFFER_SIZE 中指定的程序段数量激活插补缓存。

说明

零件程序中设置了 SD42990 $SC_MAX_BLOCKS_IN_IPOBUFFER 时，解释器在准备中执行包含该 SD 的程序段时插补缓存限制将立即生效。

即插补缓存限制可能会较期望提前数个程序段生效（另见 MD28070 $MC_MM_NUM_BLOCKS_IN_PREP）。

若需避免此情形，使插补缓存限制同步于程序段生效，则须在零件程序中设置 SD 前编写 STOPRE（预处理停止）指令。

有效性

SD42990 $SC_MAX_BLOCK_IN_IPOBUFFER 针对通道全局生效，并可在零件程序中修改。修改过的值会在程序结束时保留。若需在定义的事件下复位该设定数据，则须为此创建一个所谓的事件控制程序。例如可在 RESET 时将该设定数据设置为预定义值。

应用

若需尽可能减少程序段准备和插补间的程序段数量，例如在零件程序中读取和进一步处理实际位置时，可启用插补缓存限制。

示例

```
N10 ...
N20 ...
..........
N100 $SC_MAX_BLOCKS_IN_IPOBUFFER = 5              ; 将插补缓存限制为 5 个 NC 程序段
N110 ...
N120 ...
............
N200 $SC_MAX_BLOCKS_IN_IPOBUFFER = -1             ; 取消插补缓存限制
```

```
N210 ...
.............
```

3.5.15 基本程序段显示（仅适用于 ShopMill/ShopTurn）

3.5.15.1 功能

除现有的程序段显示外，还可在 ShopMill/ShopTurn 中通过所谓的基本程序段显示功能来显示可引起**机床上动作**的所有程序段。

实际逼近的终点位置作为绝对位置显示。位置值可选择以工件坐标系（WCS）为基准，或以可设定零点坐标系（SZS）为基准。

显示缓存中预见性保存的显示程序段数量取决于控制器预处理中对应加工状态下准备的程序段的数量。处理预处理停止时，显示程序段的数量会降至零，并在应答预处理停止后重新提升。在 REORG 事件（例如运行方式切换，ASUB 启动）下，预见性保存的显示程序段会被删除并于之后重新预处理。

在基本程序段显示中预处理的值与以下因素相匹配：

- 选择的刀具
- 进给和主轴转速
- 实际逼近的位置值
  特例：
  刀具半径补偿生效时可能会存在偏差。
  对于模数轴，基本程序段显示中显示的编程值可能位于模数区域以外。

启用/关闭基本程序段显示

基本程序段显示可通过以下设定数据启用/关闭：

SD42750 $SC_ABSBLOCK_ENABLE（使能基本程序段显示）

3.5.15.2 参数设置

配置基本程序段显示

基本程序段显示可通过以下机床数据配置：

基本程序段显示相关的 NC 机床数据	含义：
MD28400 $MC_MM_ABSBLOCK	激活基本程序段显示

MD28402 $MC_MM_ABSBLOCK_BUFFER_CONF[2]	显示缓存的大小
显示机床数据	待设置的位置值：
MD9004 $MM_DISPLAY_RESOLUTION	用于公制尺寸设定
MD9011 $MM_DISPLAY_RESOLUTION_INCH	用于英制尺寸设定
MD9010 $MM_SPIND_DISPLAY_RESOLUTION	待设置坐标系的主轴显示精度
MD9424 $MM_MA_COORDINATE_SYSTEM	WCS 或 SZS 的实际值显示

这些显示机床数据会复制到 NC 机床数据
MD17200 $MN_GMMC_INFO_UNIT[0]（全局 HMI 信息）至 MD17200 $MN_GMMC_INFO_UNIT[3] 中。这样便可通过 NC 访问这些显示机床数据。

激活

基本程序段显示通过 MD28400 $MC_MM_ABSBLOCK（激活采用绝对值的程序段显示）及上电激活。若 MD28400 $MC_MM_ABSBLOCK 赋值为 1，则会在主处理中创建一个通道专用显示缓存（FIFO）。

显示缓存（FIFO）的大小 =（MD28060 $MC_MM_IPO_BUFFER_SIZE（插补缓存中 NC 程序段的数量） + MD28070 $MC_MM_NUM_BLOCKS_IN_PREP（用于程序段预处理的程序段数量））乘以 128 字节。采用缺省机床数据设置时，该缓存的大小为 6 KB。

优化显示缓存的大小：
可设定 128 和 512 之间的值来优化存储空间需求。显示缓存中预处理的显示程序段将通过一个可配置的上载缓存传输至 HMI。

上载缓存的大小上限由（MD28402 $MC_MM_ABSBLOCK_BUFFER_CONF[0] + MD28402 $MC_MM_ABSBLOCK_BUFFER_CONF[1] + 1）
与通过 MD28400 $MC_MM_ABSBLOCK 配置的程序段长度相乘得出。

当前程序段**之前**的程序段数量在 MD28402 $MC_MM_ABSBLOCK_BUFFER_CONF[0] 中配置，当前程序段**之后**的程序段数量在 MD28402 $MC_MM_ABSBLOCK_BUFFER_CONF[1] 中配置。

前提条件

在超出 MD28400 $MC_MM_ABSBLOCK 中配置的显示程序段长度时，系统会对该显示程序段进行相应裁切。此时会在程序段末尾显示"..."字符串作为提示。

对于预编译循环
（MD10700 $MN_PREPROCESSING_LEVEL > 1，程序预处理级），显示程序段**只**包含轴位置。

适用于基本程序段显示的更多前提条件：

- 包含绝对值的模态同步动作程序段不在考量范围内。
- 在进行/不进行计算的程序段搜索期间，基本程序段显示取消。
- 极坐标编程不以笛卡尔坐标显示。

半径/直径值

基本程序段显示和位置显示所显示的直径值在内部计算时可能需要作为半径。可通过以下方式按照 G 代码组 29 对用于设定半径/直径的值进行调整：

- G 指令 DIAMCYCOF（通道专用直径编程扩展）
  该 G 指令用于在循环执行期间取消通道专用直径编程。这样在循环中可始终采用半径进行计算。位置显示和基本程序段显示继续对应 DIAMCYCOF 前的直径编程状态。
  基本程序段显示中保留最后显示的值。
- G 指令 DIACYCOFA[AX]（轴专用直径编程）
  该 G 指令用于在循环执行期间针对轴取消直径编程。这样在循环中可始终采用半径进行计算。位置显示和基本程序段显示继续对应 DIACYCOFA[AX] 前的直径编程状态。
  基本程序段显示中保留最后显示的值。
- MD27100 $MC_ABSBLOCK_FUNCTION_MASK（参数设置采用绝对值的程序段显示）

位 0 = 1　　平面轴的设定轴在基本程序段显示中通常作为直径值显示。

压缩器生效时的特性

在压缩器生效且 G 代码组 30 不为 COMPOF 时，系统会生成两个显示程序段。

- 第一个包含生效压缩器的 G 指令。
- 第二个包含“...”字符串，提示缺少显示程序段。

示例：

```
G0 X10 Y10 Z10          ；尚需为基本程序段显示预处理的程序段
COMPCAD                 ；启用压缩器优化表面质量（CAD 编程）
...                     ；字符串提示缺少显示程序段
```

为了避免 NC 性能瓶颈，基本程序段显示被自动取消。此时生成含“...”字符串的显示程序段提示缺少显示程序段。
所有显示程序段均以单程序段生成。

3.5.15.3 DIN 程序段的结构

DIN 程序段的显示程序段的结构

DIN 程序段的显示程序段的基本结构

- 程序段编号/标签
- 第一个 G 组的 G 指令
  （仅在相对上一个机床功能数据组有变化时）。
- 轴位置
  （顺序对应 MD20070 $MC_AXCONF_MACHAX_USED，通道中生效的机床轴编号）。
- 其他模态生效的 G 指令
  （仅在相对上一个机床功能数据组有变化时）。
- 编写的其他地址。

用于基本程序段显示的显示程序段按照以下规则直接从编写的零件程序段导出：

- 对宏进行扩展。
- 省去跳转标识和注释。
- 程序段编号和标签取自原程序段，DISPLOF 生效时则省去。
- 小数点后位数由显示机床数据 MD9004、MD9010 和 MD9011 通过 HMI 定义。

HMI 显示机床数据	在 NC 机床数据中访问
MD9004 $MM_DISPLAY_RESOLUTION	MD17200 $MN_GMMC_INFO_NO_UNIT[0]
MD9011 $MM_DISPLAY_RESOLUTION_INCH	MD17200 $MN_GMMC_INFO_NO_UNIT[1]
MD9010 $MM_SPIND_DISPLAY_RESOLUTION	MD17200 $MN_GMMC_INFO_NO_UNIT[2]
MD9424 $MM_MA_COORDINATE_SYSTEM	MD17200 $MN_GMMC_INFO_NO_UNIT[3]

- 编写的轴位置作为绝对位置在通过 MD9424 $MM_MA_COORDINATE_SYSTEM（用于实际值显示的坐标系）设定的坐标系（WCS/SZS）中显示。

说明

对于模数轴，模数补偿会被省去。这样一来可能会显示模数区域外的位置，且必定与基本为模数转换的位置显示存在偏差。

示例

显示程序段（原程序段）和基本程序段显示的对比：

- **编写的位置**以绝对值显示。
  地址 AP/RP 以编写的值显示。

原程序段：	显示程序段：
N10 G90 X10.123	N10 X10.123
N20 G91 X1	N20 X11.123

- **地址指定**（非 DIN 地址）以 <地址> = <常量> 的格式显示。

原程序段：	显示程序段：
N110 R1 = -67.5 R2 = 7.5	
N130 Z = R1 RND = R2	N130 Z-67.5 RND = 7.5

- **地址索引**（地址扩展）作为常量显示，<地址> [<常量>] = <常量> 。

原程序段：	显示程序段：
N220 DEF AXIS AXIS_VAR = X	
N240 FA[AXIS_VAR] = R2	N240 FA[X] = 1000

- **无地址扩展的 DIN 地址**以 <din 地址> <常量> 的格式显示。

原程序段：	显示程序段：
N410 DEF REAL FEED = 1.5	
N420 F = FEED	N420 F1.5

对于 **H 功能**：与向 PLC 的输出类型
（MD22110 $MC_AUXFU_H_TYPE_INT，H 辅助功能的类型为整数）无关，显示每次编写的值。

- 对于**通过 T 指令进行的刀具选择**
  会生成 T<值> 或 T=<字符串> 格式的显示信息。若先前编写了地址扩展，则其也会被显示。
  若配置了多个主轴，或者“通过刀架换刀”（MD20124 $MC_TOOL_MANAGEMENT_TOOLHOLDER (刀架编号)）功能生效，那么 T 号输出时总是带有地址扩展。
  若之前未编写地址扩展，则会使用主主轴或主刀架的编号（T<主轴编号/刀架> = ）。
- 对于通过 S、M3、M4、M5、M19、M40 - M45 和 M70（或 MD 20094 $MC_SPIND_RIGID_TAPPING_M_NR，用于切换至受控轴模式的 M 功能）进行的**主轴编程**，以下规则适用：
  若先前编写了地址扩展，则其也会被显示。
  若之前配置了多个主轴，那么地址扩展也将一并输出。
  若之前未编写地址扩展，则使用主主轴编号（S<主轴编号>=）。
- G[<组>] = <表述> 格式的**间接 G 指令编程**由对应的 G 指令替代。

原程序段：	**显示程序段：**
N510 R1=2	
N520 G[8]= R1	N520 G54

- 不生成可执行程序段的**模态生效的 G 指令**会被收集，并在句法允许的前提下通过下一个可执行程序段的显示程序段显示（DIN 程序段）。若非此情形（例如预定义子程序调用 TRANSMIT），系统会将包含变更过 G 指令的独立显示程序段前置。

原程序段：	**显示程序段：**
N610 G64	G64
N620 TRANSMIT	N620 TRANSMIT

- 对于出现 **F 和 FA** 地址的**零件程序行**，系统总是会生成一个显示程序段（即便在 MD22240 $MC_ AUXFU_F_SYNC_TYPE = 3（F 功能的输出时间）的情形下）。

原程序段：	**显示程序段：**
N630 F1000	N630 F1000
N640 X100	N640 X100

- **为程序段显示生成的显示程序段直接**从编写的零件程序段推导。若通过轮廓预处理生成了中间程序段（例如刀具半径补偿 G41/G42，半径/倒角 RNDM，RND，CHF，CHR），那么这些程序段从运动所基于的零件程序段获取显示信息。

原程序段：	显示程序段：
N710 Y157.5 G42	N710 Y157.5 G42
N720 Z-67.5 RND=7.5	N720 Z-67.5 RND=7.5

- 使用 **EXECTAB 指令**（执行轮廓元素表）时，显示程序段中会显示通过 EXECTAB 生成的程序段。

原程序段：	显示程序段：
N810 EXECTAB (KTAB[5])	N810 G01 X46.147 Z-25.38

- 使用 **EXECSTRING 指令**时，显示程序段中会显示通过 EXECSTRING 生成的程序段。

原程序段：
N910 DEF STRING[40] PROGSTRING = "N905 M3 S1000 G94 Z100 F1000 G55"
N920 EXECSTRING(PROGSTRING)

原程序段：
N905 Z100 G55 G94 M3 S1000 F1000

3.6 程序控制

3.6.1 概述

控制系统提供各种功能来影响 NC 程序的执行过程。

这些功能主要用于测试或试运行新零件程序。通过使用可显著降低测试阶段机床受损的风险及时间消耗。

可同时激活多个程序控制功能，以实现更佳的测试效果。

功能

提供以下程序控制功能：

- 程序测试 (页 108)
- 空运行进给 (页 113)
- 降低快速运行 (页 115)

- 程序停止 (页 116)
- 手轮偏移 (页 117)
- 隐藏程序段 (页 119)
- 单程序段 (页 122)
- 配置停止（选项） (页 133)

选择和激活

通过操作界面

通过操作界面（例如 SINUMERIK Operate），为激活的通道选择程序控制功能：

1. 软键：操作区“加工”>“程序控制”
2. 菜单“程序控制”：选择功能的复选框。

然后 PLC 将选择信号传输给 NC/PLC 接口的相应激活信号。

直接通过 PLC 用户程序（适用于 Basic Program Plus）

也可以直接通过 PLC 用户程序对功能进行激活（置位 NC/PLC 接口信号中相关的激活信号）。

通过 FC1（适用于 Basic Program）

此功能可通过相应的 FC1 参数激活。

3.6.2 程序测试

功能

“程序测试”状态下，可运行一个零件程序但并不在机床轴真正运行。借此用户可对编写的轴位置以及零件程序的辅助功能输出进行检查。此外还可将此程序仿真作为扩展的句法校验。

说明

主轴的运行

与机床轴不同，缺省状态下，在处于“程序测试”状态下的通道中，主轴运行被使能（参见“基本设定”章节）。

若需要主轴在“程序测试”状态下**不**运行，则必须通过 PLC 用户程序进行明确禁止：

缺省设置

缺省状态下，在“程序测试”状态下的通道设置如下：

- 系统为轴生成与正常运行时相一致的设定值，但不将值输出至机床轴。
- 系统将主轴的设定值输出至机床轴。
- 为禁用轴/主轴显示的实际值在系统内部根据设定值生成。
- 正常执行通道同步指令。
- 正常处理 NC/PLC 接口信号。
- 程序执行时间与正常模式下一致。

可视需要针对具体通道或轴暂时退出“程序测试”状态。这样一来，系统会将设定值重新输出至机床轴，从而实现实际的轴运行。

选择和激活

通过操作界面

通过操作界面（例如 SINUMERIK Operate），为基本画面“机床”中显示的通道选择“程序测试”：

1. 软键：操作区“加工”>“程序控制”
2. 菜单“程序控制”：选择复选框“程序测试 (PRT)”。

然后 PLC 将选择信号传输给 NC/PLC 接口的相应激活信号。

直接通过 PLC 用户程序

也可以直接通过 PLC 用户程序对功能进行激活（置位 NC/PLC 接口信号中相关的激活信号）。

NC/PLC 接口信号

在选择程序测试（PRT）后，下列 NC/PLC 接口信号置位：

- 通道
  - <HmiChan>.basic.in.progTestSelected（由 HMI：程序测试已选择）== 1
  - <Chan>.basic.out.progTest（由 PLC：激活程序测试）== 1
  - <Chan>.basic.in.progTestSelected（由 NC：程序测试生效）== 1
- 轴
  - <HmiAxis>.basic.in.progTestSuppressionRequested（由 HMI：抑制程序测试）== 0
  - <HmiAxis>.basic.in.progTestRequested（由 HMI 激活程序测试）== 0
  - <Axis>.basic.out.progTestSuppression（由 PLC：抑制程序测试）== 0
  - <Axis>.basic.out.progTestRequest（由 PLC：激活程序测试）== 0
- 主轴
  - <HmiAxis>.basic.in.progTestSuppressionRequested（由 HMI：抑制程序测试）== 1
  - <HmiAxis>.basic.in.progTestRequested（由 HMI：激活程序测试）== 0
  - <Axis>.basic.out.progTestSuppression（由 PLC：抑制程序测试）== 1
  - <Axis>.basic.out.progTestRequest（由 PLC：激活程序测试）== 0

说明

运行的缺省状态

在通道中选择“程序测试”后，运行的缺省状态为：

- 进给轴 Disabled
- 主轴：已使能

禁止主轴的运行

若需要主轴在“程序测试”期间**不**运行，则必须通过 PLC 用户程序进行明确禁止：

- <HmiAxis>.basic.in.progTestSuppressionRequested（抑制程序测试）== 0
- <HmiAxis>.basic.in.progTestRequested（激活程序测试）== 1
- <Axis>.basic.out.progTestSuppression（由 PLC：抑制程序测试）== 0
- <Axis>.basic.out.progTestRequest（由 PLC：激活程序测试）== 1

使能轴的运行

若需要通道的轴在“程序测试”期间运行，则必须通过 PLC 用户程序进行明确使能：

- <Axis>.basic.out.progTestSuppression（由 PLC：抑制程序测试）== 1
- <Axis>.basic.out.progTestRequest（由 PLC：激活程序测试）== 0

允许的切换时间点

- 通道
  仅在通道状态为“复位”或“中断”的情况下，才允许对用于取消/启用通道专用“程序测试”状态的接口信号（<HmiChan>.basic.in.progTestSelected
  或 <Chan>.basic.out.progTest）进行切换。
- 轴/主轴
  始终可以对用于取消/启用轴专用“程序测试”状态的接口信号
  （<HmiAxis>.basic.in.progTestSuppressionRequested /
  <HmiAxis>.basic.in.progTestRequested
  或 <Axis>.basic.out.progTestSuppression /
  <Axis>.basic.out.progTestRequest）进行切换。

程序测试和程序运行

在程序测试功能生效的情况下，可通过以下 NC/PLC 接口信号启动和执行零件程序（包含辅助功能输出、等待时间、G 指令输出等）：

<Chan>.basic.out.ncStart（NC 启动）

软件限位开关、工作区域限制等安全功能继续生效。

编写的速度保持不变。这表示，操作界面上的位置和速度设定与通常执行零件程序时完全相同。此时位置控制不会中断，因此在测试功能关闭后无需使进给轴回参考点。

说明

准停信号 <Axis>.basic.in.posCoarseReached/
<Axis>.basic.in.posFineReached（粗/精准停）能够反映机床上的实际状态。
在程序测试中，仅当轴被推离设定位置时才需取消这些信号（程序测试期间设定位置保持恒定）。
对于信号 <Chan>.basic.in.progTestSelected（程序测试生效），PLC 程序和零件程序均可通过变量 $P_ISTEST 确定如何在测试期间响应或跳转至此信号。

说明

空运行进给

“程序测试”状态下的程序执行也可与“空运行进给”功能一同激活。这样一来能够缩短编写了较小进给值的零件程序的执行时间。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.progTestSelected	LBP_Chan*.E_ProgTest	DB21, ...DBX33.7
<Axis>.basic.in.posCoarseReached	LBP_Axis*.E_ExactCoarse	DB31, ...DBX60.6
<Axis>.basic.in.posFineReached	LBP_Axis*.E_ExactFine	DB31, ...DBX60.7

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.progTest	LBP_Chan*.A_ProgTest	DB21, ...DBX1.7
<Chan>.basic.out.ncStart	LBP_Chan*.A_NCStart	DB21, ...DBX7.1
<Axis>.basic.out.progTestSuppression	LBP_Axis*.A_ProgtestSuppress	DB31, ...DBX14.0
<Axis>.basic.out.progTestRequest	LBP_Axis*.A_ProgtestActivate	DB31, ...DBX14.1

OP → PLC

Basic Program Plus	Basic Program	
<HmiChan>.basic.in.progTestSelected	LBP_HMI.E_MMC_ProgTest /	DB21, ...DBX25.7
<HmiAxis>.basic.in.progTestSuppressionRequested	LBP_HMI.E_MMC_ProgtestSuppress	DB31, ...DBX128.0
<HmiAxis>.basic.in.progTestRequested	LBP_HMI.E_MMC_ProgtestActivate	DB31, ...DBX128.1

说明

刀具管理

由于轴禁用，刀库布局在程序测试时不会变化。必须通过一个 PLC 应用程序确保刀具管理数据和刀库之间的一致性。

3.6.3 空运行进给

功能

为在程序测试期间缩短加工时间，可通过激活空运行进给更快地执行运动。例如在采用 G01、G02、G03、G33、G34、G35、G95 的情况下，空运行进给会取代编写的进给率生效。

注意
过高的切削速度可致刀具或工件损坏 若在空运行进给生效的情况下执行工件加工，产生的切削速度可能会超出允许的范围并导致刀具和/或工件损坏。 因此，空运行进给应仅用于程序测试。

激活

在 AUTO 或 MDA 运行模式中在操作界面上选择空运行进给，随后可在自动中断时或在程序段末尾将其激活。

选择

通过操作界面（例如 SINUMERIK Operate），为基本画面“机床”中显示的通道选择“空运行进给”功能：

1. 软键：操作区“加工”>“程序控制”
2. 菜单“程序控制”：选择“空运行进给（DRY）”复选框。

作为发送至 PLC 用户程序的激活此功能的请求，在 HMI/PLC 接口中置位以下信号：

`<HmiChan>.basic.in.dryRunSelected`（空运行进给已选择）== 1

然后 PLC 将选择信号传输给 NC/PLC 接口的相应激活信号。

说明

也可以直接通过 PLC 用户程序对功能进行激活（置位 NC/PLC 接口信号中相关的激活信号）。

激活

作为发送至 NC 的激活此功能的请求，PLC 用户程序须将以下 NC/PLC 接口信号置位：

`<Chan>.basic.out.dryRunFeedrate`（激活空运行进给）== 1

反馈信息

此功能在 NC 中生效后，作为对 PLC 用户程序的反馈信息，置位以下 NC/PLC 接口信号：

<Chan>.basic.in.dryRunFeedrateActive（空运行进给生效）== 1

参数设置

空运行进给

通道中生效的空运行进给通过以下方式设置：

SD42100 $SC_DRY_RUN_FEED = <空运行进给>

说明

旋转进给率

空运行进给值也可替代通过 G95 编写的旋转进给率。

空运行进给的模式

空运行进给的生效方式设置为：

SD42101 $SC_DRY_RUN_FEED_MODE = <值>

<值>	含义
0	SD42100 和编写的进给率中的最大值作为空运行进给率生效。
1	SD42100 和编写的进给率中的最小值作为空运行进给率生效。
2	SD42100 作为空运行进给率生效。
10	除螺纹切削（G33、G34、G35）和攻丝（G331、G332、G63）外，与设置为“0”时相同：在这些功能中，编写的进给率生效。
11	除螺纹切削（G33、G34、G35）和攻丝（G331、G332、G63）外，与设置为“1”时相同：在这些功能中，编写的进给率生效。
12	除螺纹切削（G33、G34、G35）和攻丝（G331、G332、G63）外，与设置为“2”时相同：在这些功能中，编写的进给率生效。

更多信息

关于进给控制的更多信息参见功能手册“进给轴和主轴”。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.dryRunFeedrateActive	LBP_Chan*.E_DRY	DB21, ...DBX318.6

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.dryRunFeedrate	LBP_Chan*.A_DRY	DB21, ...DBX0.6

OP → PLC

Basic Program Plus	Basic Program	
<HmiChan>.basic.in.dryRunSelected	LBP_HMI.E_MMC_DRY	DB21, ...DBX24.6

3.6.4 降低快速运行

功能

在降低快速运行时，轴在快速运行模式中的运行速度将降低至当前设置的百分比值。

设置

降低快速运行的百分比值通过以下设定数据指定：

SD42122 $SC_OVR_RAPID_FACTOR（预先设定关于操作的额外快速运行倍率）

SD42122 仅在“降低快速运行”功能激活时有效。

激活

通过操作界面（例如 SINUMERIK Operate），为基本画面“机床”中显示的通道选择并激活“降低快速运行”功能：

1. 软键：操作区“加工”>“程序控制”
2. 菜单“程序控制”：选择“降低快速运行（RG0）”复选框。

3.6.5 程序停止

功能

如果“编程停止 1”功能激活，程序处理在所有包含辅助功能 M01 的程序段处停止。这允许用户在加工工件时检查各个加工步骤的结果。

激活

选择

通过操作界面（例如 SINUMERIK Operate），为基本画面“机床”中显示的通道选择“编程停止 1”功能：

1. 软键：操作区“加工”>“程序控制”
2. 菜单“程序控制”：选择“编程停止 1（M01）”复选框。

作为发送至 PLC 用户程序的激活此功能的请求，在 HMI/PLC 接口中置位以下信号：

<HmiChan>.basic.in.m01Selected（已请求 M01）== 1

说明

也可以直接通过 PLC 用户程序对功能进行激活（置位 NC/PLC 接口信号中相关的激活信号）。

激活

作为发送至 NC 的激活此功能的请求，PLC 用户程序须将以下 NC/PLC 接口信号置位：

<Chan>.basic.out.m01（激活 M01）== 1

反馈信息

此功能在 NC 中生效后，作为对 PLC 用户程序的反馈信息，置位以下 NC/PLC 接口信号：

<Chan>.basic.in.m00m01Active（M01 生效）== 1

如果 NC 程序由于该功能而在程序段结束处停止，则设置以下 NC/PLC 接口信号：

<Chan>.basic.in.progStateStopped（程序状态“停止”）== 1

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.m00m01Active	LBP_Chan*.E_M01	DB21, ...DBX32.5
<Chan>.basic.in.progStateStopped	LBP_Chan*.E_ProgStop	DB21, ...DBX35.2

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.m01	LBP_Chan*.A_M01	DB21, ...DBX0.5

OP → PLC

Basic Program Plus	Basic Program	
<HmiChan>.basic.in.m01Selected	LBP_HMI.E_MMC_M01	DB21, ...DBX24.5

3.6.6 手轮偏移

功能

通过“手轮偏移（DRF）”功能，可在 AUTO 运行方式下通过电子手轮在基准坐标系中设置几何轴和辅助轴的附加增量零偏。

典型应用：

- 在一个 NC 程序段内进行刀具磨损补偿
  对于加工时间非常长的 NC 程序段来说，有必要在一个程序段内对刀具磨损执行手动补偿（例如大型的平面铣床）。
- 磨削时的精补
- 简单的温度补偿（热误差补偿）

激活

选择

通过操作界面（例如 SINUMERIK Operate），为基本画面“机床”中显示的通道选择“手轮偏移”功能：

1. 软键：操作区“加工”>“程序控制”
2. 菜单“程序控制”：选择“手轮偏移（DRF）”复选框。

作为发送至 PLC 用户程序的激活此功能的请求，在 HMI/PLC 接口中置位以下信号：

<HmiChan>.basic.in.handwheelOffsetDrfSelected（已请求手轮偏移）== 1

说明

也可以直接通过 PLC 用户程序对功能进行激活（置位 NC/PLC 接口信号中相关的激活信号）。

激活

作为发送至 NC 的激活此功能的请求，PLC 用户程序须将以下 NC/PLC 接口信号置位：

<Chan>.basic.out.handwheelOffsetDrf（激活手轮偏移）== 1

反馈信息

此功能在 NC 中生效后，作为对 PLC 用户程序的反馈信息，置位以下 NC/PLC 接口信号：

<Chan>.basic.in.handwheelOvrActive（手轮超驰生效）== 1

更多信息

关于 DRF 偏移的详细信息参见功能手册“进给轴和主轴”。

3.6.6.1 程序测试 - ONE 信号表

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.handwheelOvrActive	LBP_Chan*.E_HWOverlay	DB21, ...DBX33.3

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.handwheelOffsetDrf	LBP_Chan*.A_DRF	DB21, ...DBX0.3

OP → PLC

Basic Program Plus	Basic Program	
<HmiChan>.basic.in.handwheelOffsetDrfSelected	LBP_HMI.E_MMC_DRF	DB21, ...DBX24.3

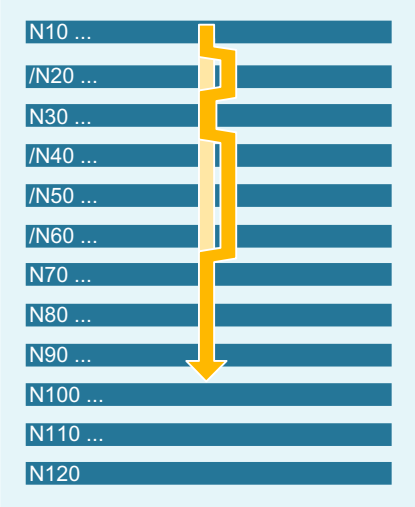
3.6.7 隐藏程序段

对于不需要在每次程序运行中都执行的 NC 程序段，可在执行时对其设置跳转。例如，可在测试或试运行新程序时使用该功能。

跳过程序段

在程序段编号前用"/"（斜线）标记要跳转的程序段。也可以几个程序段连续跳过。跳过的程序段中的指令不会被执行，系统会从下一未跳过的程序段开始继续执行程序。

示例：



程序代码	注释
N10 ...	; 执行
/N20 ...	; 跳过

程序代码	注释
N30 …	；执行
/N40 …	；跳过
/N50 …	；跳过
/N60 …	；跳过
N70 …	；执行
...	

跳转级

可以为程序段指定跳转级（最多为 10 级），可以通过操作界面或 PLC 用户程序将其激活。

在 NC 程序中，可以在前面插入斜线接着加入跳转级的数字来指定。每个程序段只能给定一个跳转级。

示例：

程序代码	注释
/ ...	；程序段跳过（第 1 跳转级）
/0 ...	；程序段跳过（第 1 跳转级）
/1 N010...	；程序段跳过（第 2 跳转级）
/2 N020...	；程序段跳过（第 3 跳转级）
...	
/7 N100...	；程序段跳过（第 8 跳转级）
/8 N080...	；程序段跳过（第 9 跳转级）
/9 N090...	；程序段跳过（第 10 跳转级）

说明

跳转级修改只能在控制系统停止/复位状态下进行。

说明

可以使用多少个跳转级取决于显示机床数据。

说明

“跳过程序段”功能在程序段搜索时同样生效。

说明

使用系统变量和用户变量，也可以改变程序运行过程，用于有条件跳转。

选择

通过操作界面（例如 SINUMERIK Operate），为基本画面“机床”中显示的通道选择跳转级：

1. 软键：操作区“加工”>“程序控制”
2. 菜单“程序控制”：选择“跳转级 ...（SKP）”复选框。

作为发送至 PLC 用户程序的激活所选跳转级的请求，在 HMI/PLC 接口中置位相关信号：

<HmiChan>.basic.in.skipBlockLevel0（隐藏第 1 个跳转级的程序段）== 1

<HmiChan>.basic.in.skipBlockLevel1（隐藏第 2 个跳转级的程序段）== 1

...

<HmiChan>.basic.in.skipBlockLevel7（隐藏第 8 个跳转级的程序段）== 1

<HmiChan>.basic.in.skipBlockLevel8（隐藏第 9 个跳转级的程序段）== 1

<HmiChan>.basic.in.skipBlockLevel9（隐藏第 10 个跳转级的程序段）== 1

说明

也可以直接通过 PLC 用户程序对功能进行激活（置位 NC/PLC 接口信号中相关的激活信号）。

激活

作为发送至 NC 的激活跳转级的请求，PLC 用户程序须相应的 NC/PLC 接口信号置位：

<Chan>.basic.out.skipBlockLevel0（隐藏第 1 个跳转级的程序段）== 1

<Chan>.basic.out.skipBlockLevel1（隐藏第 2 个跳转级的程序段）== 1

...

<Chan>.basic.out.skipBlockLevel7（隐藏第 8 个跳转级的程序段）== 1

<Chan>.basic.out.skipBlockLevel8（隐藏第 9 个跳转级的程序段）== 1

<Chan>.basic.out.skipBlockLevel9（隐藏第 10 个跳转级的程序段）== 1

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.skipBlockLevel0	LBP_Chan*.A_SKP0	DB21, ...DBX2.0
<Chan>.basic.out.skipBlockLevel1	LBP_Chan*.A_SKP1	DB21, ...DBX2.1
...		
<Chan>.basic.out.skipBlockLevel7	LBP_Chan*.A_SKP7	DB21, ...DBX2.7
<Chan>.basic.out.skipBlockLevel8	LBP_Chan*.A_SKP8	DB21, ...DBX31.6
<Chan>.basic.out.skipBlockLevel9	LBP_Chan*.A_SKP9	DB21, ...DBX31.7

OP → PLC

Basic Program Plus	Basic Program	
<HmiChan>.basic.in.skipBlockLevel0	LBP_HMI.E_MMC_SKP0	DB21, ...DBX26.0
<HmiChan>.basic.in.skipBlockLevel1	LBP_HMI.E_MMC_SKP1	DB21, ...DBX26.1
...		
<HmiChan>.basic.in.skipBlockLevel7	LBP_HMI.E_MMC_SKP7	DB21, ...DBX26.7
<HmiChan>.basic.in.skipBlockLevel8	LBP_HMI.E_MMC_SKP8	DB21, ...DBX27.0
<HmiChan>.basic.in.skipBlockLevel9	LBP_HMI.E_MMC_SKP9	DB21, ...DBX27.1

参数设置

跳转级的数量

跳转级的数量通过以下机床数据定义：

MD51029 $MM_MAX_SKP_LEVEL（NC 程序中跳转级的最大数量）

3.6.8 单程序段

3.6.8.1 功能

在单程序段执行中，零件程序处理会在每个程序段后停止。若选择了刀具半径补偿，那么会在每个由控制系统插入的中间程序段之后停止加工。程序状态切换至“程序状态：停止”。通道状态保持为激活。通过 NC 启动来执行下一个零件程序段。

应用

用户通过该功能可以一个程序段、一个程序段地执行零件程序并检查单独的加工步骤。如果已执行的零件程序段正确，可执行下一程序段。

单段方式

可供使用的单程序段类型如下：

- **“SB1：单程序段粗略”**（在具有机床功能的每个程序段之后停止 = 主运行程序段）
  在每个完整执行的具有机床功能的程序段后停止 NC 程序或加工。
- **“SB2：计算程序段”**（在每个包括计算程序段和注释程序段的程序段之后停止）
  在每个程序段后停止 NC 程序或加工。
- **“SB3：单程序段精确”**（在具有机床功能的每个程序段之后停止，即使在一个循环中）
  与 SB1 相似，但还会在每个具有机床功能的程序段后的循环中停止。

参见

前提条件 (页 131)

3.6.8.2 选择和激活

选择

可以选择单程序段加工：

- 通过机床控制面板（“Single Block”键）
- 例如使用 SINUMERIK Operate 操作界面：
  1. 操作区“机床”>“程序控制”
  2. 菜单：“程序控制”：在单程序段类型下拉列表中选择“SB1”、“SB2”或“SB3”。

更多信息：操作手册

激活

此功能由 PLC 基本程序通过 NC/PLC 接口信号为所选通道激活：

`<Chan>.basic.out.singleBlock` （激活单程序段）== 1

反馈信息

一旦程序执行在单程序段运行下处理了零件程序段，就会设置以下 NC/PLC 接口信号：

<Chan>.basic.in.singleBlockStopAtBlockEnd（由于单程序段在程序段末尾停止）== 1

<Chan>.basic.in.progStateStopped（程序状态“停止”）== 1

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.progStateStopped	LBP_Chan*.E_ProgStop	DB21, ...DBX35.2
<Chan>.basic.in.singleBlockStopAtBlockEnd	LBP_Chan*.E_SblStopAtBlockEnd	DB21, ...DBX39.4

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.singleBlock	LBP_Chan*.A_SBL	DB21, ...DBX0.4

3.6.8.3 参数设置

机床数据

关闭单程序段加工

针对特定加工情形和程序类型，可通过以下机床数据进行设置，使得在单程序段功能生效的情况下不发生停止：

MD10702 $MN_IGNORE_SINGLEBLOCK_MASK

说明

通过在 ASUB 或子程序内编写 (页 126) SBLON/SBLOF，能够显性激活/关闭单程序段执行。

说明

单程序段类型"SB2：计算程序段"中，机床数据仅在系统 ASUB、用户 ASUB 和属性为 DISPLOF 的子程序这些情形下生效。

事件控制的程序调用行为

事件控制的程序调用（Prog-Events）在单程序段方面的特性通过以下机床数据设置：

MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK

中断程序（ASUB）的特性

中断程序（ASUB）在单程序段方面的特性通过以下机床数据设置：

MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP

在单程序段模式下激活了一个 ASUB 时，该 ASUB 将完整执行。直到该 ASUB 结束或在首个未激活单程序段抑制的插补程序段中，单程序段才会重新生效。如果在从 ASUB 过渡到 NC 程序时轨迹速度过大，以至于无法在后续程序段中制动至静止状态，那么例如在连续路径运行 G64 生效时，视情况可通过再后面的多个程序段进行制动。

说明

在 ASUB 内编写 (页 126) SBLON 后，在此情形下**无法**重新激活单程序段执行。

设定数据

单程序段"SB2：计算程序段"（SD42200）的排故模式

由于程序段的提前解码，与主运行相关的当前程序段显示与显示的变量值之间的基准可能会在操作界面上丢失。此时可能无法可靠地显示变量值。

通过以下通道专用设定数据可设置单程序段"SB2：计算程序段"生效时通过每个程序段执行预处理停止。这样一来，零件程序段的预先执行会被抑制，当前程序段显示与变量值显示之间的关联得以保留。

SD42200 $SC_SINGLEBLOCK2_STOPRE（激活 SB2 的排故模式）

说明

轮廓偏差

在排故模式下处理具有单程序段类型"SB2：计算程序段"的运行程序段时，可能会出现轮廓偏差。

3.6.8.4 编程

即使在单程序段执行激活时，用户也能执行全部或部分的 NC 程序，无需停止执行。通过指令 SBLOF 抑制单程序段执行，通过指令 SBLON 可再次激活。

抑制全部 NC 程序的单程序段执行

若在**主程序**的第一行（PROC ...）中通过编程关闭单程序段执行（SBLOF），则其一直生效，直至该NC程序结束或中止为止。借此在采用单程序段的情况下以不停止的方式执行NC程序。

若在**子程序**的第一行（PROC ...）中通过编程关闭单程序段执行（SBLOF），则其一直生效，直至该 NC 程序结束或中止为止。编写的返回指令将决定是否在该子程序的末尾停止：

- 通过 M17 返回：停止于子程序末尾处
- 通过 RET 跳回：在子程序末尾处**不**停止

在 NC 程序内抑制单程序段执行

若在 NC 程序内的一个程序段中通过编程关闭单程序段执行（SBLOF），则自此程序段起，直至下一次编写激活单程序段执行（SBLON）或直至生效的子程序级结束为止，关闭单程序段执行。

句法

抑制全部 NC 程序的单程序段执行：

```
PROC ... SBLOF
...
```

在 NC 程序内抑制单程序段执行：

```
...
SBLOF
...
SBLON
...
```

含义

PROC:	一个程序的第一个指令	
SBLOF:	用于关闭单程序段执行的预定义步骤	
	在单独程序段中编程：	是，在 PROC 程序段中有效
	生效方式：	模态
SBLON:	用于激活单程序段执行的预定义步骤	
	在单独程序段中编程：	是
	生效方式：	模态

特殊性

- **单程序段执行被抑制时的程序段显示**
  可以在子程序中使用 DISPLOF 抑制当前的程序段显示。若一同编写 DISPLOF 和 SBLOF，则会在单程序段停止时在子程序内显示子程序调用。
- **异步子程序 (ASUB) 的单程序段执行抑制**
  为了在单程序段执行激活时一步执行 ASUB，必须在 ASUB 中编程一个带有 SBLOF 的 PROC 语句。这也适用于功能“可编辑的系统 ASUB”（MD11610 $MN_ASUP_EDITABLE）。
  如果系统或用户 ASUB 中的单程序段停止通过在 PROC 语句中编写 SBLOF 指令或者通过设置机床数据 MD10702 $MN_IGNORE_SINGLEBLOCK_MASK 来进行抑制（位 0 = 1 或位 1 = 1），则单程序段停止可以通过在 ASUB 中编程 SBLON 再次激活。
  如果用户 ASUB 中的单程序段停止通过设置机床数据 MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP 来进行抑制，则单程序段停止**无法**通过在 ASUB 中编程 SBLON 再次激活。
- **不同单程序段执行方式下的特性**
  – “SB2：计算程序段” 和 MD10702 $MN_IGNORE_SINGLEBLOCK_MASK，位 12 = 1：
  → 在 SBLON 程序段中不停止。
  – “SB3：精准单程序段”：
  → 指令 SBLOF 被抑制。
- **抑制已调用程序中的单程序段执行**
  如果在一个子程序的 PROC 语句中编程 SBLOF，则可通过 M17 执行子程序跳回来停止执行。由此防止在调用的程序中已开始执行下一个程序段。如果在一个子程序中使用 SBLOF（不在 PROC 语句中编写 SBLOF）激活单程序段执行的抑制，则只有在调用程序中的下一个机床功能程序段之后才会停止执行。如果不希望如此，则应在跳回之前（M17）在子程序中再次编程 SBLON。使用 RET 跳回上一级程序时，不会停止执行。

示例

示例 1：在 NC 程序内抑制单程序段执行

初始情况：单程序段执行激活。

程序代码	注释
N10 G1 X100 F1000	
N20 **SBLOF**	；关闭单程序段执行
N30 Y20	
N40 M100	
N50 R10=90	
N60 **SBLON**	；重新启用单程序段执行
N70 M110	
N80 ...	

N20 和 N60 之间的区域，在单程序段运行时作为一步处理。

示例 2：循环对于用户而言就如同一个指令

初始情况：单程序段执行激活。

主程序：

程序代码
...
N100 G1 X10 G90 F200
N120 X-4 Y6
N130 **CYCLE1**
N140 G1 X0
N150 M30

循环 CYCLE1：

程序代码	注释
N100 **PROC** CYCLE1 DISPLOF **SBLOF**	；抑制全部程序的单程序段执行。
N110 R10=3*SIN(R20)+5	
N120 IF (R11 <= 0)	
N130 SETAL(61000)	
N140 ENDIF	
N150 G1 G91 Z=R10 F=R11	
N160 M17	

单程序段执行激活时执行循环 CYCLE1。即执行 CYCLE1 时，必须按一次“启动”按钮。

示例 3：需隐藏由 PLC 启动的、用于激活经过修改的零点偏移和刀具补偿的 ASUB

程序代码	注释
N100 **PROC** NV **SBLOF DISPLOF**	；抑制单程序段执行和程序段显示。
N110 CASE $P_UIFRNUM OF	
0 GOTOF _G500	
1 GOTOF _G54	
2 GOTOF _G55	
3 GOTOF _G56	
4 GOTOF _G57	
DEFAULT GOTOF END	
N120 _G54:G54 D=$P_TOOL T=$P_TOOLNO	
N130 RET	
N140 _G54:G55 D=$P_TOOL T=$P_TOOLNO	
N150 RET	
N160 _G56:G56 D=$P_TOOL T=$P_TOOLNO	
N170 RET	
N180 _G57:G57 D=$P_TOOL T=$P_TOOLNO	
N190 RET	
N200 END:D=$P_TOOL T=$P_TOOLNO	
N210 RET	

示例 4：子程序中的针对性停止

初始情况：

- 单程序段执行激活。
- MD10702 $MN_IGNORE_SINGLEBLOCK_MASK, 位 12 = 1

主程序：

程序代码	注释
N10 G0 X0	; 单程序段停止
N20 X10	; 单程序段停止
N30 **CYCLE**	；由循环产生的运行程序段。
N50 G90 X20	; 单程序段停止
M30	

循环 CYCLE：

程序代码	注释
PROC CYCLE **SBLOF**	；抑制单程序段执行
N100 R0=1	
N110 **SBLON**	；由于 MD10702 位 12 = 1，无单程序段停止
N120 X1	; 单程序段停止
N140 **SBLOF**	

程序代码	注释
N150 R0=2	
RET	

示例 5：程序调用时抑制单程序段执行

初始情况：

- 单程序段执行方式 2 生效。
- 在 SBLON 程序段中不停止执行（MD10702 $MN_IGNORE_SINGLEBLOCK_MASK，位 12 = 1）

程序代码	注释
N10 X0 F1000	; 单程序段停止
N20 UP1(0)	
PROC UP1(INT _NR) **SBLOF**	; 为 UP1 关闭单程序段执行
N100 X10	
N110 UP2(0)	
PROC UP2(INT _NR)	
N200 X20	
N210 **SBLON**	; 启用单程序段执行
N220 X22	; 单程序段停止
N230 UP3(0)	
PROC UP3(INT _NR)	
N300 **SBLOF**	; 关闭单程序段执行
N305 X30	
N310 **SBLON**	; 启用单程序段执行
N320 X32	; 单程序段停止
N330 **SBLOF**	; 关闭单程序段执行
N340 X34	
N350 M17	; 单程序段停止（M17）
N240 X24	; 单程序段停止（N210）
N250 M17	; 单程序段停止（M17）
N120 X12	
N130 M17	; 单程序段停止（M17）
N30 X0	; 单程序段停止
N40 M30	; 单程序段停止

3.6.8.5 前提条件

使用同步动作时的特性

单程序段类型“SB2：计算程序段”中，若启用逐段生效同步动作，则要在下一次主处理程序段（具有机床功能的程序段）后才会执行下一次停止。该同步动作与下一主处理程序段之间的预处理程序段不会停止。

G33/G34/G35 的特性

对于 G33/G34/G35 语句系列，只有选择了“空运行进给”功能之后单程序段 SB1 或 SB3 才有效。SB2 在 G33/G34/G35 中未生效。计算程序段不会被单段执行（仅针对译码单程序段）。

报警 16922

初始情况：在通道中，NC 程序因其中编写的 M0 停止，且该通道中激活了单程序段（<Chan>.basic.out.singleBlock == 1）

在此情形下，若在操作界面上多次在单程序段类型 SB1 或 SB3 与 SB2 之间切换，则会显示报警 16922“超出最大嵌套深度”。

原理

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.singleBlock	LBP_Chan*.A_SBL	DB21, ...DBX0.4

3.6.9 BAG 专用单程序段类型 A / B

在采用 BAG 专用单程序段时，在一个通道（控制通道）中通过单程序段逐段执行 NC 程序。在该控制通道中，单程序段必须通过 NC/PLC 接口信号 <Chan>.basic.out.singleBlock 激活。

在 BAG 的其余通道（“关联通道”）中，根据针对特定 BAG 通过 NC/PLC 接口信号 <ModeGroup>.basic.out.singleBlockTypeB / <ModeGroup>.basic.out.singleBlockTypeA 选择的单程序段类型 A 或 B 逐段执行相应的 NC 程序。在该关联通道中 NC/PLC 接口信号 <Chan>.basic.out.singleBlock 不允许置位。

单程序段类型 A / B

单程序段类型 A：若控制通道停止，关联通道也立即停止，与 NC 停止相似。

单程序段类型 B：若控制通道停止，则关联通道在相应的程序段末尾停止，与程序段交界处 NC 停止相似。

接口信号

控制通道

- <Chan>.basic.out.singleBlock（激活单程序段）

BAG 的所有通道

- <Chan>.basic.out.ncStart（NC 启动）

BAG

- <ModeGroup>.basic.out.singleBlockTypeB（单程序段类型 B）
- <ModeGroup>.basic.out.singleBlockTypeA（单程序段类型 A）

单程序段类型 A 的流程图

前提条件：BAG 的所有通道处于“复位”或“中断”状态下。

1. PLC 用户程序：在控制通道中选择单程序段，<Chan>.basic.out.singleBlock = 1
2. PLC 用户程序：为 BAG 选择单程序段类型 A，<ModeGroup>.basic.out.singleBlockTypeA = 1
3. PLC 用户程序：启动 BAG 的所有通道，<Chan>.basic.out.singleBlock = 1
4. 控制通道在程序段末尾处停止。
5. 所有关联通道获得立即停止执行的内部信号。
6. 在所有关联通道到达制动阶段末尾的情况下，BAG 的所有通道处于“中断”状态。

单程序段类型 B 的流程图

前提条件：BAG 的所有通道处于“复位”或“中断”状态下。

1. PLC 用户程序：在控制通道中选择单程序段，<Chan>.basic.out.singleBlock = 1
2. PLC 用户程序：为 BAG 选择单程序段类型 B，<ModeGroup>.basic.out.singleBlockTypeB = 1
3. PLC 用户程序：启动 BAG 的所有通道，<Chan>.basic.out.singleBlock = 1
4. 控制通道在程序段末尾处停止。
5. 所有关联通道获得在程序段末尾处停止执行的内部信号。
6. 在所有关联通道到达相应的程序段末尾的情况下，BAG 的所有通道处于“中断”状态。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<ModeGroup>.basic.out.singleBlockTypeB	LBP_ModeGroup.A_SingleBlock_B	DB11.DBX1.6
<ModeGroup>.basic.out.singleBlockTypeA	LBP_ModeGroup.A_SingleBlock_A	DB11.DBX1.7

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.singleBlock	LBP_Chan*.A_SBL	DB21, ...DBX0.4
<Chan>.basic.out.ncStart	LBP_Chan*.A_NCStart	DB21, ...DBX7.1

3.6.10 配置停止（选项）

说明

“配置停止”功能是一个需要许可证的选项。

订货号：6FC5800-0BS24-0YB0

3.6.10.1 功能

与“单程序段”功能（在每个程序段后停止 NC 程序）相比，“配置停止”功能允许机床操作员仅在程序中的关键点停止。在启动程序之前，他必须从机床制造商定义的选项中选择相关的停止情况。

停止情况的定义

以下 NC 功能调用或 NC 功能转换可指定为停止情况：

- M/H/T/D 功能调用
- 调用子程序（也包括用户 ASUB）

- WAIT 指令（WAITM / WAITMC / WAITE）
- G 功能转换：
  - G0 → 非 G0
  - 非 G0 → G0
  - G0 → G0

配置停止的停止情况在调试期间通过设定数据 (页 137)定义。选择和激活 (页 135)由机床操作员通过操作界面或直接由 PLC 用户程序执行。

程序运行过程

如果控制器在程序解释过程中识别到被定义为停止情况的功能调用/子程序调用或 G 功能转换，则 NC 程序在有功能调用/子程序调用或 G 功能转换的程序段**之前**的程序段结束处停止。与此相反，对于 WAIT 指令，它被认为是一种停止情况，停止发生在带有 WAIT 指令的程序段结束处。

程序状态切换至“程序状态：停止”。通道状态保持为激活。

通过 NC 启动，程序处理可以继续进行，直到下一个停止情况。

应用

模具及模型制造

刀具和模具制造中的零件程序可以包含大量描述工件上单个轨迹的短 G1 程序段。为了使程序不会在每个短 G1 程序段后停止，就像激活的单程序段处理一样，机床操作员可以使用“配置停止”功能代替单程序段处理，例如，您可以指定仅在 G0 进给运动到工件的下一个轨迹之前停止工件（停止情况“非 G0 → G0 过渡”）。对于较长的 G0 程序段，这些程序段对过程更为关键并以最大速度运行，则应在每个 G0 程序段后停止（停止情况“G0 → G0 过渡”）。

工件驶入

在机床中，许多机床功能通过辅助功能（M/H 功能）或循环（子程序）来实现（例如换刀、将中心架移动到工件等）。工件驶入时，机床操作员通常希望稍微慢一点地运行此类机床功能，以确保不会发生碰撞。为此，他可以使用“配置停止”功能在循环开始或辅助功能开始之前停止程序，然后将倍率开关调低至 0%。然后他通过 NC 启动继续程序处理。切换带有循环或辅助功能的程序段，并在拧开倍率开关时开始运行。

说明

也可以通过 PLC 用户程序将倍率值设置为 0%。

工件驶入时的另一个关键情况是开始具有大质量或物体的运动。如果在程序启动之前定义了合适的停止情况，也可以使用“配置停止”功能停止此类运动。

3.6.10.2 选择和激活

选择

在所有运行方式下，都可以通过用户界面为基本画面“机床”中显示的通道选择“配置停止”功能。

例如使用 SINUMERIK Operate：

1. 操作区“机床”>“程序控制”

2. “程序控制”窗口：选择“CST 配置停止”

通过“程序控制”窗口中的相应选择框选择相关的停止情况。调试期间定义的所有停止情况都在此处列出。

要选择功能并选择相关的停止情况，通道必须处于“复位”或“中断”状态：

<Chan>.basic.in.stateReset == 1（通道状态“复位”）

<Chan>.basic.in.stateInterrupted == 1（通道状态“中断”）

作为发送至 PLC 用户程序的激活此功能的请求，将在 HMI/PLC 接口中设置以下信号：

<Axis>.coupling.out.requestCtrlAxis（激活配置停止）== 1

通过操作软件激活配置停止：

<hmiChan>.basic.in.configuredStopFuncSelected == 1（HMI：通过操作软件激活配置停止 (CST)）

<hmiChan>.basic.in.atStopSetOvrZeroSelected == 1（HMI：在停止倍率时置位为 0%）

说明

如果多通道机床中的一个通道设置了倍率 0% 请求，则同一运行方式组中的所有其他通道也将停止。

说明

也可以直接通过 PLC 用户程序对功能进行激活（置位 NC/PLC 接口信号中相关的激活信号）。

激活

作为发送至 NC 的激活此功能的请求，PLC 用户程序须将以下 NC/PLC 接口信号置位：

<Chan>.basic.out.configuredStopFunc（激活配置停止）= 1

反馈信息

作为发送至 PLC 用户程序以表明功能在 NC 中生效的反馈消息，将以下 NC/PLC 接口信号置位：

<Chan>.basic.in.configuredStopFuncActive（激活配置停止）= 1

如果 NC 程序由于该功能而在程序段结束处停止，则设置以下 NC/PLC 接口信号：

<Chan>.basic.in.configuredStopAtBlockEnd（由于配置停止而在程序段的结束处停止）= 1

<Chan>.basic.in.progStateStopped（程序状态“停止”）= 1

取消

在所有程序和通道状态下都可以取消选择“配置停止”功能。

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<Axis>.plcCtrl.out.requestPlcCtrl	LBP_Axis*.A_PLCAxis	DB31, ...DBX28.7
<Chan>.basic.out.configuredStopFunc	LBP_Chan*.A_CfgStopFunc	DB21, ...DBX7.6

NC → PLC

Basic Program Plus	Basic Program	
<hmiChan>.basic.in.configuredStopFuncSelected	LBP_Chan*.E_MMC_CfgStopFunc	DB21, ...DBX24.1
<hmiChan>.basic.in.atStopSetOvrZeroSelected	LBP_Chan*.E_MMC_AtStopSetOvrZero	DB21, ...DBX24.2
<Chan>.basic.in.stateReset	LBP_Chan*.E_ChanReset	DB21, ...DBX35.7
<Chan>.basic.in.stateInterrupted	LBP_Chan*.E_ChanInterrupt	DB21, ...DBX35.6
<Chan>.basic.in.progStateStopped	LBP_Chan*.E_ProgStop	DB21, ...DBX35.2

Basic Program Plus	Basic Program	
<Chan>.basic.in.configuredStopFuncActive	LBP_Chan*.E_CfgStopFuncActive	DB21, ...DBX39.6
<Chan>.basic.in.configuredStopAtBlockEnd	LBP_Chan*.E_CfgStopAtBlockEnd	DB21, ...DBX39.7

3.6.10.3 参数设置

机床数据

防止配置停止

针对特定加工情形，可通过以下机床数据进行设置，使得在“配置停止”功能生效的情况下不发生停止：

MD10703 $MN_IGNORE_CFG_STOP_MASK

设定数据

停止情况的定义

配置停止的停止情况通过以下设定数据定义：

设定数据	含义		
SD42220 $SC_CFG_STOP_ARRAY[<n>] = "<Name>"	执行前应停止的 NC 功能和子程序（循环）		
	<名称>：	NC 功能/子程序的名称	
		数据类型：	STRING
		允许： • M/H/T/D 功能调用 必须遵守以下句法规则： – D 不支持地址扩展。 – 辅助功能的值始终为正整数值，包括 0。 – M 功能允许地址扩展的负整数值。 示例：“M[-5]=3” – 允许使用星号 (*) 作为通配符。 示例："M[*]=87" • 子程序名称 有效命名法：“_N_...”	
	<n>：	数组下标	
		数据类型：	INT
		取值范围：	0 ... 19
	示例： • $SC_CFG_STOP_ARRAY[0] = "M=87" • $SC_CFG_STOP_ARRAY[1] = "_N_CS_TOOL" • $SC_CFG_STOP_ARRAY[3] = "_N_MY_TOOL_CHANGE"		
SD42222 $SC_CFG_STOP_MASK	G 功能转换/WAIT 指令，在此基础上停止。		
	位 0	G0 → G0	
	位 1	G0 → 非 G0	
	位 2	非 G0 → G0	
	位 3	WAITM / WAITMC / WAITE	

设定数据	含义	
SD42224 $SC_CFG_STOP_ARRAY_MASK	使能 SD42220 $SC_CFG_STOP_ARRAY[<n>]	
	位 0	使能 SD42220 $SC_CFG_STOP_ARRAY[0]
	位 1	使能 SD42220 $SC_CFG_STOP_ARRAY[1]
	...	...
	位 19	使能 SD42220 $SC_CFG_STOP_ARRAY[19]

说明

指定 NC 功能或子程序名称时，必须遵守适用的命名法和句法规则。

出现错误时，会显示报警 16968“机床数据 $SC_CFG_STOP_ARRAY 包含无效句法”。

示例

$SC_CFG_STOP_MASK = 6	在选择和取消选择 G0 之前停止
$SC_CFG_STOP_ARRAY[0]="_N_MY_UP_0"	在调用子程序 MY_UP_0 之前停止
$SC_CFG_STOP_ARRAY[1]="_N_MY_UP_0"	在调用子程序 MY_UP_0 之前停止
SC_CFG_STOP_ARRAY[2]="_N_TC"	在执行循环“TC”之前停止
$SC_CFG_STOP_ARRAY_MASK='H0FFF'	使能 SD42220 $SC_CFG_STOP_ARRAY 位 0 ... 11

说明

取代子程序（循环）

如果要在定义文件中使用宏技术（DEFINE ... AS ...）被另一个子程序替换的子程序（循环）之前停止程序执行，则必须将替换子程序的名称指定为配置停止的停止情况。

示例：

在 MMAC_GM.DEF 文件中，将“TC”循环替换为“TC_096”循环：

```
DEFINE TC AS TC_096
```

为了使停止起作用，配置的停止必须位于“_N_TC_096”而不是“_N_TC”：

SD42220 $SC_CFG_STOP_ARRAY[<n>] = "_N_TC_096"

3.6.10.4 编程

与单程序段处理一样，也可以使用预定义的程序 SBLOF 和 SBLON 通过配置的停止来关闭和再次启用 NC 程序中的功能。

更多信息参见章节 “单程序段 (页 122)”。

3.6.10.5 前提条件

配置停止，同时进行单程序段处理

如果“单程序段”和“配置停止”功能同时激活，则配置的停止具有附加效果。这意味着由于配置停止而产生的额外停止将被保留，例如当 MD10702 $MN_IGNORE_SINGLEBLOCK_MASK 阻止单程序段停止时有效。

仅在可执行的 NC 程序段处停止

通过“配置停止”功能的停止只能在可执行程序段中生效。它们对通过内部程序段处理删除的 NC 程序段没有影响。

停止预防情况

在以下情况下，配置的停止会忽略停止：

- 无主轴停止的攻丝 G331/G63 已激活。
- 螺纹切削 G33 已激活。
- 刀具退刀运动已激活。
- 在引导启动中的 ASUB 期间。
- 当发生在有条件可中断的程序区域（停止延迟区域）时。

同步动作

同步动作的动作部分不会针对配置停止进行评估！因此，指定为停止情况的辅助功能在同步动作中发生时不会导致程序停止。

3.6.10.6 示例

停止情况：G0 → G0 过渡

参数设置

$SC_CFG_STOP_MASK = 1　　在 G0 到 G0 过渡时停止

程序示例

程序代码
N10 G0 X10
N20 G0 X20
N30 G1 X30 F1000
N40 G1 X40
N50 G0 X50
N60 G0 X70
N70 M30

程序运行过程

NC 启动 → 程序开始，在程序段 N10 中到达位置 X10 后停止。
NC 启动 → 程序继续，在程序段 N50 中到达位置 X50 后停止。
NC 启动 → 程序继续进行，直到程序结束。

停止情况：WAIT 指令

参数设置

$SC_CFG_STOP_MASK = 8　　在 WAIT 指令处停止（仅适用于通道 1）

程序示例

通道 1：

程序代码
N110 G0 X0 Y0 Z0 G64
N120 X10
N125 X15
N130 **WAITM(7,1,2)**
N140 Y10
N150 **WAITM(99,1,2)**
N160 Z10
N170 **WAITE(2)**
N180 X20
N200 M30

通道 2：

程序代码
N2010 G0 X0 G64
N2020 X10

程序代码

```
N2030 WAITM(7,1,2)
N2040 X20
N2060 X30
N2070 WAITM(99,1,2)
N2080 X40
N2099 M30
```

程序运行过程

NC 启动（通道 1+2） → 启动两个程序。到达 WAITM 指令并在程序段 N130 中设置等待标志后，通道 1 中的程序在程序段结束处停止。

NC 启动（通道 1） → 通道 1 中的程序继续进行。到达 WAITM 指令并在程序段 N150 中设置等待标志后，程序在程序段结束处停止。

NC 启动（通道 1） → 通道 1 中的程序继续进行。在程序段 N170 中到达 WAITE 指令后，程序在程序段结束处停止。在通道 2 中等待程序结束。

NC 启动（通道 1） → 通道 1 中的程序继续进行，直到程序结束。

停止情况：M 功能

参数设置

$SC_CFG_STOP_ARRAY[3]="M=88"	执行辅助功能 M88 前停止
$SC_CFG_STOP_ARRAY_MASK='H8'	使能 SD42220 $SC_CFG_STOP_ARRAY 位 3

程序示例 1

程序代码

```
N10 G0 X10
N20 G0 X20
N30 G1 X30 F1000
N35 M88
N40 G1 X40
N50 G0 X50
N60 M30
```

程序运行过程 1

NC 启动 → 程序启动。在程序段 N30 结束时，机床停在位置 X = 30。

NC 启动 → M 功能的输出，程序继续进行，直到程序结束。

程序示例 2

程序代码
N10 G0 X10
N20 G0 X20
N30 G1 X30 M88 F1000
N40 G1 X40
N50 G0 X50
N60 M30

程序运行过程 2

NC 启动 → 程序启动。在程序段 N20 结束时，机床停在位置 X = 20。

NC 启动 → M 功能在程序段 N30 中输出，程序继续进行，直到程序结束。

程序示例 3

程序代码
N10 G0 X10
N20 G0 X20
N30 G1 X30 F1000
N35 WHEN TRUE DO M88
N40 G1 X40
N50 G0 X50
N60 M30

程序运行过程 3

NC 启动 → 程序开始并运行到程序结束。同步动作的动作部分中的辅助功能调用不会导致程序停止。

停止情况：子程序

参数设置

$SC_CFG_STOP_ARRAY[0]="_N_CS_TOOL"	执行子程序 CS_TOOL 前停止
$SC_CFG_STOP_ARRAY_MASK='H1'	使能 SD42220 $SC_CFG_STOP_ARRAY 位 0

程序示例

程序代码
N10 G0 X10
N20 G0 X20

程序代码
N30 G1 X30 F1000
N35 CS_TOOL
N40 G1 X40
N50 G0 X50
N60 G0 X70
N70 M30

程序运行过程

NC 启动 → 程序开始，在程序段 N30 中到达位置 X30 后停止。
NC 启动 → 程序继续进行，直到程序结束。

停止情况：使用 T 功能换刀

在以下示例中，“配置停止”功能仅用于在带有刀具转塔和未激活刀具管理的车床上进行特定刀具更换之前停止。

参数设置

$MN_MM_TOOL_MANAGEMENT_MASK = 0	刀具管理没有内存预留
$MC_TOOL_MANAGEMENT_MASK = 0	刀具管理未激活
$MC_TOOL_CHANGE_MODE = 0	用刀具号换刀
$SC_CFG_STOP_ARRAY[0] = 'T=1'	换刀前停止 T=1
$SC_CFG_STOP_ARRAY[1] = 'T=3'	换入刀具前停止 T=3
$SC_CFG_STOP_ARRAY_MASK = 3	使能 SD42220 $SC_CFG_STOP_ARRAY 位 0 和 1

程序示例

程序代码	注释
...	
N50 G0 Z0	
N100 G0 Z10 X20	; 停在 Z10 X20
N200 **T1**	; 程序继续和换刀需要 NC 启动
N250 G1 X9	
N300 G1 Z100	
N350 G1 X20	
N400 G1 Z10	; 没有停止，因为 T2 不在 $SC_CFG_STOP_ARRAY 中
N500 **T2**	
N550 G1 X9	
N600 G1 Z100	

程序代码	注释
N650 G1 X20	
N700 G1 Z10	; 停在 Z10 X20
N800 **T3**	; 程序继续和换刀需要 NC 启动
N850 G1 X8	
N900 G1 Z100	
N950 G1 X20	
N1000 G1 Z10	
N2000 M30	; 由于程序结束而停止

说明

如果没有刀具管理，T 功能将被视为所有其他辅助功能。这可以在特定刀具更换之前启用选择性停止（如示例所示）。

这在刀具管理中是不可能的。此处只能在每次换刀前停止（停止情况“T[*]=*”）。

3.6.11 状态

当前选择的程序控制设置可以通过系统变量读取：

- 在零件程序中通过预处理变量：

系统变量	类型	含义
$P_ISTEST	BOOL	程序测试生效
$P_DRYRUN	BOOL	空运行进给有效
$P_ISRG0	BOOL	降低快速运行激活
$P_ISPROGSTOP	BOOL	编程停止 1（M01）激活
$P_ISDRF	BOOL	手轮偏移激活
$P_ISSKIP	BOOL	隐藏程序段激活

- 在零件程序和同步动作中通过主运行变量：

系统变量	类型	含义	
$AC_PROGINF	INT	生效的程序控制 该值是位编码的。各个位具有以下含义：	
		位 0 ... 9	跳转级 0 - 9 激活
		位 10	空运行进给有效
		位 11	编程停止 1（M01）激活
		位 12	手轮偏移激活
		位 13	单程序段激活
		位 14	降低快速运行激活
		位 15	进给停止激活
		位 16	程序测试生效
		位 17	关联辅助功能（M-1）激活
		位 18	配置停止激活

更多信息：参数手册之系统变量

3.7 程序段搜索类型 1、2 和 4

功能

通过程序段搜索功能可从近乎任意零件程序段开始执行零件程序。

使用该功能时，系统会在不触发运行的情况下快速运行零件程序，直至到达所选择的目标程序段。在此过程中会尽可能使目标程序段处的控制系统状态与正常执行零件程序时相同（例如轴位置、主轴转速、换入的刀具、NC/PLC 接口信号、变量值等方面），从而在尽可能减少手动调整的情况下从目标程序段起继续自动执行零件程序。

程序段搜索类型

- **类型 1：不带计算的程序段搜索**
  不带计算的程序段搜索能够最快地搜索到零件程序段。搜索中不执行任何计算。目标程序段处的控制系统状态与启动搜索前相同。
- **类型 2：在轮廓处计算的程序段搜索**
  通过使用在轮廓处计算的程序段搜索，可在任意情形下定位至编写的轮廓。使用 NC 启动移动到目标程序段的起始位置或者目标程序段之前的程序段的终点位置，再从该位置运行至终点位置。使用此搜索时的程序执行遵循轮廓。
- **类型 4：在程序段终点计算的程序段搜索**
  通过使用在程序段终点计算的程序段搜索，可在任意情形下定位至一个目标位置（例如换刀位置）。此时会使用目标程序段中生效的插补类型定位至目标程序段的终点或编写的下一个位置。此过程将不遵循轮廓。
  系统只会运行目标程序段中编写的轴。在启动其他零件程序的自动执行前，必要时须在机床上通过"JOG-REPOS"运行方式手动确保无碰撞的初始状态。
- **类型 5：在“程序测试”模式计算的程序段搜索（SERUPRO）**
  SERUPRO（**se**arch **ru**n by **pro**gramtest，程序测试时搜索）是一个带计算的跨通道程序段搜索功能。NC 在“程序测试”模式下启动所选择的零件程序。到达目标程序段后再取消程序测试。采用此程序段搜索类型时，还可实现执行程序段搜索的通道、同步动作以及 NC 的其它通道之间的交互。
  参见“程序段搜索类型 5（SERUPRO） (页 163)”章节。

说明

对程序段搜索的更多说明请见“程序段搜索时的特性 (页 826)”章节。

跟随动作

在程序段搜索完成后，可进行以下跟随动作：

- 类型 1 - 类型 5：自动启动一个 ASUB
  切换至最后一个动作程序段后，可将一个用户程序作为 ASUB 启动。
- 类型 1 - 类型 4：级联程序段搜索
  可从“找到搜索目标”状态启动另一个程序段搜索来查找另一个目标。

3.7.1 功能说明

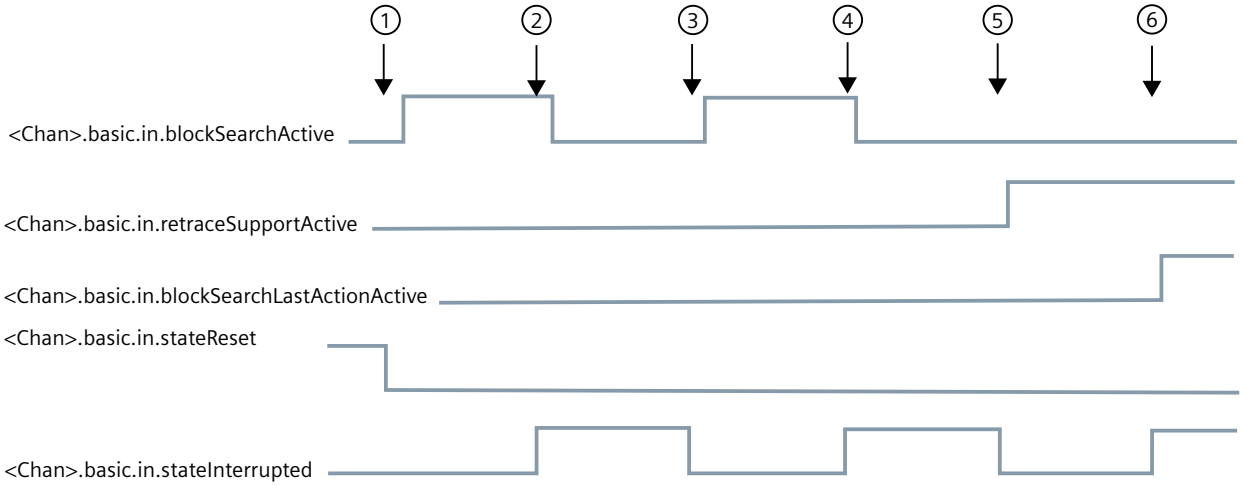
类型 2 或类型 4 的基本流程

1. 用户：通过操作界面激活程序段搜索**类型 2** 或**类型 4** （在 ... 处**计算**的程序段搜索）
2. 通过收集辅助功能搜索目标程序段
3. 在找到搜索目标 ⇒ 显示报警 10208"通过 NC 启动继续执行程序“后停止
4. 用户：通过 NC 启动执行动作程序段 `<Chan>.basic.out.ncStart` = 1（NC 启动）
5. 执行动作程序段
6. 切换至最后一个动作程序段 ⇒ 自动启动 /_N_CMA_DIR/_N_PROG_EVENT_SPF（缺省设置）作为 ASUB
7. 切换至最后一个 ASUB 程序段（`REPOSA`）
   ⇒ `<Chan>.basic.in.blockSearchLastActionActive` = 1（最后的动作程序段生效）
8. **可选**：通过 PLC 用户程序执行用户专用请求
9. 显示报警 10208"通过 NC 启动继续程序程序”
10. 用户：通过 NC 启动继续执行程序 <Chan>.basic.out.ncStart = 1（NC 启动）

未找到搜索目标

若未找到搜索目标，则会显示报警 15370"程序段搜索中未找到搜索目标”，并终止程序段搜索。

时序



① 开始搜索
② 搜索目标 1 已找到
③ 开始搜索
④ 搜索目标 2 已找到
⑤ NC 启动 - 输出动作程序段
⑥ 最后的响应程序段

动作程序段

在**类型 2** 或**类型 4**（在 ... 处**计算**的程序段搜索）程序段搜索期间，系统会收集动作，例如刀具编程（T、D）、主轴编程（S）、进给率编程或 M 功能输出。通过 NC 启动执行动作程序段时，收集的动作会输出至 PLC。

说明

为动作程序段进行 NC 启动时，在**类型 2** 或**类型 4**（在 ... 处**计算**的程序段搜索）程序段搜索期间收集的主轴编程（S 值、M3 / M4 / M5 / M19、SPOS）将生效。

用户须通过 PLC 用户程序确保刀具可运行，或确保主轴编程复位及不输出：

<Axis>.basic.out.resetMovement = 1（主轴复位）。

前提条件

程序段搜索类型 4 后的累加模式

程序段搜索类型 4（在程序段终点计算的程序段搜索）后，若首个轴编程采用增量方式，可将编写的增量值叠加至到搜索目标为止所收集的位置值，或叠加至轴的当前实际值。这通过以下机床数据设置：

SD42444 $SC_TARGET_BLOCK_INCR_PROG

NC 启动后系统会分析此设定数值，以输出动作程序段。

单程序段

在类型 2 或类型 4（在 ... 处**计算**的程序段搜索）程序段搜索中找到搜索目标，且“单程序段”功能激活（<Chan>.basic.out.singleBlock == 1（激活单程序段））后，若不希望在每个动作程序段后停止，可通过以下机床数据取消此特性：

MD10702 $MN_IGNORE_SINGLEBLOCK_MASK，位 3 = 1（在动作程序段中抑制单段模式）

接口信号“定位程序段生效”

此接口信号**仅**在程序段搜索**类型 2**（**在轮廓处计算**的程序段搜索）时置位：

- <Chan>.basic.in.blockSearchApproachActive = 1（定位程序段生效）

在程序段搜索**类型 4**（**在程序段终点计算**的程序段搜索）中，此接口信号不会置位，因为此时不会生成定位程序段（定位程序段等同于目标程序段）。

目标程序段的插补方式

在程序段搜索**类型 4**（**在程序段终点计算**的程序段搜索）中，定位运动以目标程序段中生效的插补方式执行。若插补方式不是线性插补（G0 或 G1），则可能引起定位终止并触发报警（例如圆弧插补 G2 或 G3 时出现圆弧终点错误）。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.blockSearchApproachActive	LBP_Chan*.E_BegBlock	DB21, ...DBX32.4
<Chan>.basic.in.blockSearchLastActionActive	LBP_Chan*.E_LastActBlock	DB21, ...DBX32.6
<Chan>.basic.in.blockSearchActive	LBP_Chan*.E_BlockSearch	DB21, ...DBX33.4
<Chan>.basic.in.stateReset	LBP_Chan*.E_ChanReset	DB21, ...DBX35.7
<Chan>.basic.in.stateInterrupted	LBP_Chan*.E_ChanInterrupt	DB21, ...DBX35.6

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.ncStart	LBP_Chan*.A_NCStart	DB21, ...DBX7.1
<Axis>.basic.out.resetMovement	LBP_Axis*.A_DelDTGSpReset	DB31, ...DBX2.2
<Chan>.basic.out.singleBlock	LBP_Chan*.A_SBL	DB21, ...DBX0.4

3.7.2 程序段搜索与其它 NCK 功能的组合使用

3.7.2.1 程序段搜索后/中的 ASUB

程序段搜索类型 2 和类型 4：通道轴的同步

若在类型 2 或类型 4（在 ... 处计算的程序段搜索）的程序段搜索后启动 ASUB，系统会将所有通道轴的实际位置同步。

影响

在 ASUB 中读取下列系统变量时，得到的值如下：

- $P_EP：通道轴的当前实际位置（WCS）
- $AC_RETPOINT：收集的通道轴的程序段搜索位置（WCS）

程序段搜索类型 2：结束 ASUB

在程序段搜索类型 2（在轮廓处计算的程序段搜索）中，必须编写后续指令 REPOSA（重新定位至轮廓；线性；所有通道轴）来结束 ASUB。

影响

- 所有通道轴均被运行至程序段搜索中收集的搜索位置。
- $P_EP ==“收集的通道轴的程序段搜索位置（WCS）”

程序段搜索类型 4：REPOS 特性

在程序段搜索类型 4（在程序段终点计算的程序段搜索）后，在通过**起点**和**终点**描述的时间范围内，REPOS 指令不会触发自动重定位：

- **起点**：NC/PLC 接口信号<Chan>.basic.in.blockSearchLastActionActive ==1（最后的动作程序段生效）
- **终点**：通过 NC 启动继续执行程序。

定位运行的起点是发出 NC 启动指令时通道轴的当前位置。终点通过零件程序中编写的其他运行得出。

在程序段搜索类型 4 中，NC 不会生成定位运行。

影响：

- 在退出 ASUB 后，系统变量 $P_EP（编写的终点位置）提供通道轴通过 ASUB 或手动（JOG 运行方式）运行定位至的实际位置。
  $P_EP == 通道轴的当前实际位置（WCS）

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.blockSearchLastActionActive	LBP_Chan*.E_LastActBlock	DB21, ...DBX32.6

3.7.2.2 程序段搜索后的 PLC 动作

若所有动作程序段均通过 NC 执行，且动作可通过 PLC 实现，例如启动 ASUB 来执行换刀（或通过操作人员，例如溢出转存），那么以下通道专用 NC/PLC 接口信号会置位：

- <Chan>.basic.in.blockSearchLastActionActive = 1（最后的动作程序段生效）

参数设置输出报警的时间点

为了向操作人员提示“需要在通道中进行 NC 启动方能继续执行程序”，系统会显示报警 10208“通过 NC 启动继续执行程序”。

该报警的显示时间点可通过以下机床数据设置：
MD11450 $MN_SEARCH_RUN_MODE，位 0 = <值>

<值>	含义
0	程序段搜索后切换至最后一个动作程序段时： • 零件程序执行停止 • <Chan>.basic.in.blockSearchLastActionActive = 1（最后的动作程序段生效） • 显示报警 10208
1	程序段搜索后切换至最后一个动作程序段时： • 零件程序执行停止 • <Chan>.basic.in.blockSearchLastActionActive = 1（最后的动作程序段生效） • <Chan>.basic.out.blockSearchAckPlcActionDone == 1（PLC 动作结束）时，才显示报警 10208

伴随报警，系统还会置位以下接口信号：

- <Chan>.basic.in.blockSearchLastActionActive = 1（导致加工停止的 NC 报警待处理）
- <Chan>.basic.in.blockSearchLastActionActive = 1（通道特定的 NC 报警待处理）

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.blockSearchLastActionActive	LBP_Chan*.E_LastActBlock	DB21, ...DBX32.6
<Chan>.basic.in.ncAlarmWithStopActive	LBP_Chan*.E_NCKalarmStop	DB21, ...DBX36.7
<Chan>.basic.in.ncAlarmActive	LBP_Chan*.E_NCKalarmChan	DB21, ...DBX36.6

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.blockSearchAckPlcActionDone	LBP_Chan*.A_PLC_ActCompl	DB21, ...DBX1.6

3.7.2.3 程序段搜索后的主轴功能

控制系统特性和输出

程序段搜索中收集的主轴专用辅助功能是在动作程序段中自动输出至 PLC，还是依据用户定义于随后输出，可通过以下机床数据定义：

MD11450 $MN_SEARCH_RUN_MODE，位 2 = <值>

<值>	含义
0	程序段搜索中收集的主轴专用辅助功能（M3、M4、M5、M19、M70）在动作程序段中输出。
1	程序段搜索中收集的主轴专用辅助功能在动作程序段中不输出。 可依据用户定义于随后输出，例如在一个 ASUB 中输出。为此，系统会将收集的主轴专用辅助功能保存在系统变量中：

系统变量

系统会将程序段搜索中收集的主轴专用辅助功能保存在以下系统变量中：

系统变量	说明
$P_SEARCH_S [<n>]	最近一次编写的主轴转速或切削速度
$P_SEARCH_SDIR [<n>]	最近一次编写的主轴旋转方向
$P_SEARCH_SGEAR[<n>]	最近一次编写的齿轮档 M 功能
$P_SEARCH_SMODE [<n>]	最近一次编写的主轴运行方式
$P_SEARCH_SPOS [<n>]	最近一次通过 M19、SPOS 或 SPOSA 编写的主轴位置或运行行程
$P_SEARCH_SPOSMODE [<n>]	最近一次通过 M19、SPOS 或 SPOSA 编写的位置定位模式
<n>：主轴编号	

随后需要输出主轴专用辅助功能时，例如可在一个 ASUB 中读取系统变量，并于动作程序段输出后进行输出：

<Chan>.basic.in.blockSearchLastActionActive==1（最后的动作程序段生效）

说明

系统变量 $P_S、$P_DIR 和 $P_SGEAR 的内容可能会在程序段搜索后由于同步而丢失。

ASUB、程序段搜索以及动作程序段的更多详细信息请见“抑制主轴专用辅助功能的输出 (页 831)”和“程序测试 (页 108)”章节。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.blockSearchLastActionActive	LBP_Chan*.E_LastActBlock	DB21, ...DBX32.6

3.7.2.4 程序段搜索时读取系统变量

在零件程序中，可通过系统变量从预处理、主处理或伺服/驱动区域读取值：

$P_...	预处理相关系统变量，提供编写的值
$A_...	主处理相关系统变量，提供当前值
$V_...	伺服/驱动相关系统变量，提供当前值

在类型 2 和类型 4 的程序段搜索（在 ... 处**计算**的程序段搜索）中，程序段不会进入主处理，因此程序段搜索期间主处理相关和伺服/驱动相关系统变量不会改变。 针对这些变量，必要时须在 NC 程序中通过 $P_SEARCH（程序段搜索生效）查询程序段搜索是否生效，从而实现对程序段搜索的特殊处理。

预处理相关系统变量在所有搜索类型中均能提供正确的值。

3.7.3 程序段搜索后自动启动 ASUB

参数设置

激活功能

程序段搜索后的自动 ASUB 启动通过以下机床数据设置激活：

MD11450 $MN_SEARCH_RUN_MODE，位 1 = 1

待激活的程序

采用缺省设置时，程序段搜索后切换至最后一个动作程序段时，系统会从 _N_CMA_DIR 目录激活 **_N_PROG_EVENT_SPF** 程序作为 ASUB。若需激活另一个程序，则须在以下机床数据中输入该用户程序的名称：

MD11620 $MN_PROG_EVENT_NAME

启用单程序段执行时的特性

启用了单程序段模式时，可通过以下通道专用机床数据设置是无中断地执行激活的 ASUB，还是使单程序段执行生效：

MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK，位 4 = <值>

<值>	含义
0	单程序段执行生效。
1	抑制单程序段执行。

启用读取禁止时的特性

设置了读取禁止时（<Chan>.basic.out.disableReadIn = 1），可通过以下通道专用机床数据设置是完整执行 ASUB，还是使读取禁止生效：

MD20107 $MC_PROG_EVENT_IGN_INHIBIT，位 4 = <值>

<值>	含义
0	读取禁止生效。
1	抑制读取禁止。

说明

MD11620、MD20108 和 MD20107 的更多参数设置信息请见“参数设置 (页 83)”章节。

编程

启动 ASUB 的事件保存在系统变量 $P_PROG_EVENT 中。在程序段搜索后自动激活的情形下，$P_PROG_EVENT 输出值“5”。

过程

程序段搜索后自动启动 ASUB 的过程

1. 用户：通过操作界面激活程序段搜索**类型 2** 或**类型 4** （在 ... 处**计算**的程序段搜索）
2. 通过收集辅助功能搜索目标程序段
3. 在找到搜索目标 ⇒ 显示报警 10208”通过 NC 启动继续执行程序 “后停止
4. 用户：通过 NC 启动执行动作程序段 <Chan>.basic.out.ncStart = 1（NC 启动）
5. 执行动作程序段
6. 切换至最后一个动作程序段 ⇒ 自动启动 /_N_CMA_DIR/_N_PROG_EVENT_SPF（缺省设置）作为 ASUB

7. 切换至最后一个 ASUB 程序段（REPOSA）
   ⇒ <Chan>.basic.in.blockSearchLastActionActive = 1（最后的动作程序段生效）
8. **可选**：通过 PLC 用户程序执行用户专用请求
9. 显示报警 10208"通过 NC 启动继续程序程序"

说明

MD11450 $MN_SEARCH_RUN_MODE，位 0 == 1 时，通过 PLC 用户程序使能（<Chan>.basic.out.blockSearchAckPlcActionDone = 1（PLC 动作结束））后才输出报警 10208。

10. 用户：通过 NC 启动继续执行程序 <Chan>.basic.out.ncStart = 1（NC 启动）

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.blockSearchLastActionActive	LBP_Chan*.E_LastActBlock	DB21, ...DBX32.6

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.blockSearchAckPlcActionDone	LBP_Chan*.A_PLC_ActCompl	DB21, ...DBX1.6
<Chan>.basic.out.disableReadIn	LBP_Chan*.A_RIdisable	DB21, ...DBX6.1
<Chan>.basic.out.ncStart	LBP_Chan*.A_NCStart	DB21, ...DBX7.1

3.7.4 级联程序段搜索

功能

通过“级联程序段搜索”功能可从“找到搜索目标”状态启动另一个程序段搜索。每次找到搜索目标后可任意级联搜索，其可用于以下程序段搜索功能：

- 程序段搜索类型 1（不带计算）
- 类型 2：在轮廓处计算的程序段搜索
- 程序段搜索类型 3（在程序段终点计算）

说明

只有当查找到搜索目标后，才可以从停止的程序执行开始启动另一个“级联程序段搜索”。

激活

“级联程序段搜索”在现有机床数据中配置：
MD11450 $MN_SEARCH_RUN_MODE

- 通过位 3 = 0（FALSE）使能级联程序段搜索
（即可设定多个搜索目标）。
- 考虑到兼容性，也可通过位 3 = 1（TRUE）禁用级联程序段搜索。预设为：级联程序段搜索位 3 = 0。

过程特性

找到搜索目标，重新启动程序段搜索

到达搜索目标后，程序执行停止，并将搜索目标作为当前程序段显示。每次找到搜索目标后，可任意次数地重复新的程序段搜索。

修改搜索目标设定

在每次开始程序段搜索前，可对搜索目标设定和程序段搜索功能进行修改。

示例：含级联程序段搜索的执行序列

- 复位
- 程序段搜索，找到搜索目标 1

- 程序段搜索，找到搜索目标 2 →“级联程序段搜索”
- 通过 NC 启动输出动作程序段 → 报警 10208
- NC 启动 → 继续执行程序

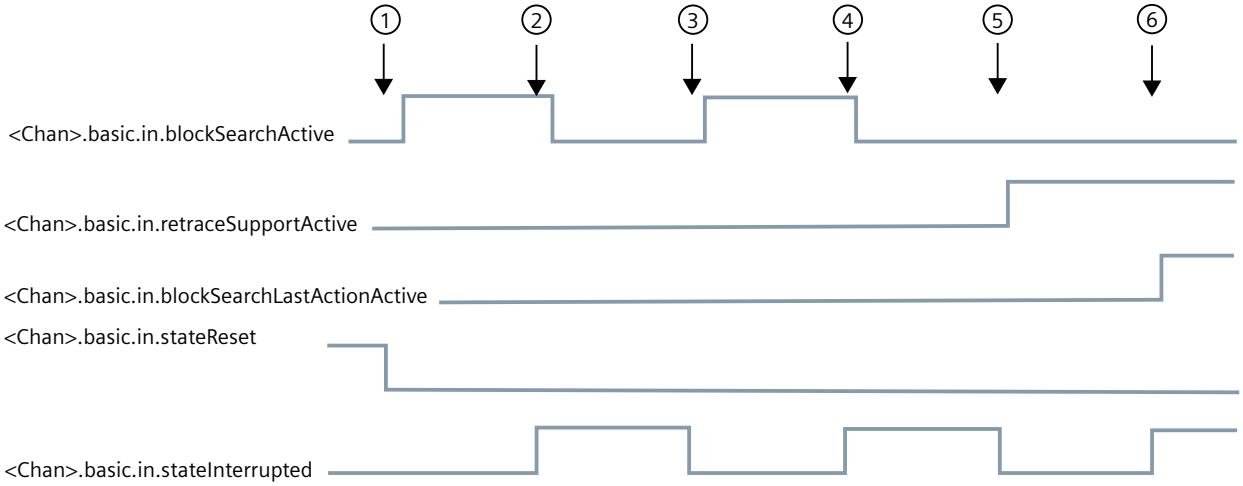


图 3-2　接口信号的时间顺序

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.blockSearchApproachActive	LBP_Chan*.E_BegBlock	DB21, ...DBX32.4
<Chan>.basic.in.blockSearchLastActionActive	LBP_Chan*.E_LastActBlock	DB21, ...DBX32.6
<Chan>.basic.in.blockSearchActive	LBP_Chan*.E_BlockSearch	DB21, ...DBX33.4
<Chan>.basic.in.stateReset	LBP_Chan*.E_ChanReset	DB21, ...DBX35.7
<Chan>.basic.in.stateInterrupted	LBP_Chan*.E_ChanInterrupt	DB21, ...DBX35.6

3.7.5　带计算的程序段搜索的示例

选择

参考以下示例，选择最能应对您的问题的程序段搜索类型。

类型 4：在程序段终点计算的程序段搜索

示例包含刀具管理生效时程序段搜索后的自动换刀：

1. 设置机床数据：
   MD11450 $MN_ SEARCH_RUN_MODE 设为 1
   MD11602 $MN_ASUP_START_MASK 位 0 = 1（从停止状态启动 ASUB）
2. 从 PLC 通过 FB4 选择 ASUB“SUCHLAUF_ENDE”（另见功能手册“PLC 和基本程序”，章节“PLC 主程序”）。
3. 载入和选择零件程序“WERKSTUECK_1”。
4. 程序段搜索编号为 N220 的程序段的终点。
5. HMI 发送“找到搜索目标”信息。
6. 通过 NC 启动输出动作程序段。
7. 使用 PLC 信号：
   <Chan>.basic.in.blockSearchLastActionActive （最后一个动作程序段生效）
   PLC 通过 FC9 启动 ASUB “SUCHLAUF_ENDE”（另见功能手册“PLC”）。
8. ASUB 结束（例如可通过待定义 M 功能 M90 分析，参见示例程序段 N1110）后，PLC 置位信号：
   <Chan>.basic.out.blockSearchAckPlcActionDone（PLC 动作结束）。
   或者可查询 NC/PLC 接口信号：
   <Chan>.basic.in.asupStopStateActive（ASUB 已停止）
   。
   这样一来会显示报警 10208，也就是说现在可由操作人员执行其它动作。
9. 操作人员进行手动调整（JOG、JOG-REPOS、溢出转存）。
10. 通过 NC 启动继续执行零件程序。

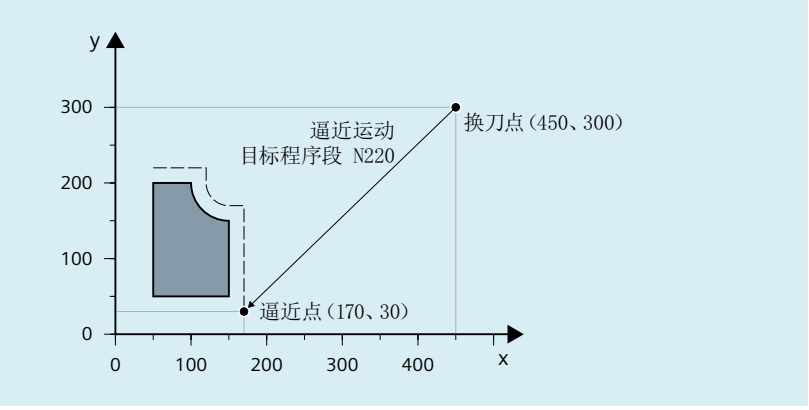


图 3-3　程序段搜索程序段终点时的定位运行（目标程序段 N220）

说明

采用目标程序段为 N220 的“在轮廓处计算的程序段搜索”时，会生成至换刀点（目标程序段的起点）的定位运行。

类型 2：在轮廓处计算的程序段搜索

示例包含刀具管理生效时程序段搜索后的自动换刀：

第 1 步至第 3 步	如程序段搜索类型 4 的示例
第 4 步	沿轮廓搜索程序段编号 N260
第 5 步至第 10 步	如程序段搜索类型 4 的示例

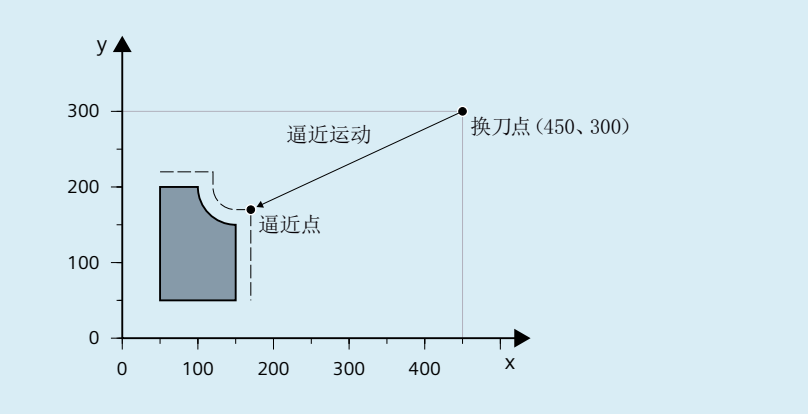


图 3-4　沿轮廓进行程序段搜索时的定位运行（目标程序段 N260）

说明

目标程序段为 N260 的“搜索程序段终点”会触发报警 14040（圆弧终点错误）。

类型 4 和类型 2 的零件程序

PROC WERKSTUECK_1

程序代码	注释
; 主程序	
...	
; 使用刀具**“FRAESER_1”**加工轮廓段 1	
...	
N100 G0 G40 X200 Y200	; 取消半径补偿
N110 Z100 D0	; 取消长度补偿
; 轮廓段 1 结束	
;	

程序代码	注释
; 使用刀具"**FRAESER_2**"加工轮廓段 2	
N200 T="FRAESER_2"	; 预选择刀具
N210 WZW	; 调用换刀程序
N220 G0 X170 Y30 Z10 S3000 M3 D1	; 轮廓段 2 的定位程序段
N230 Z-5	; 进刀
N240 G1 G64 G42 F500 X150 Y50	; 轮廓起始点
N250 Y150	
N260 G2 J50 X100 Y200	
N270 G1 X50	
N280 Y50	
N290 X150	
N300 G0 G40 G60 X170 Y30	; 取消半径补偿
N310 Z100 D0	; 取消长度补偿
轮廓段 2 结束	
...	
M30	
PROC WZW	
; **换刀程序**	
N500 DEF INT TNR_AKTIV	; 生效 T 号的变量
N510 DEF INT TNR_VORWAHL	; 预选 T 号的变量
N520 TNR_AKTIV = $TC_MPP6[9998,1]	; 读取生效刀具的 T 号
N530 GETSELT(TNR_VORWAHL)	; 读取预选刀具的 T 号
;	
; 仅当刀具尚未生效时才执行换刀	
N540 IF TNR_AKTIV == TNR_VORWAHL GOTOF ENDE	
N550 G0 G40 G60 G90 SUPA X450 Y300 Z300 D0	; 定位至换刀点
N560 M6	; 执行换刀
;	
ENDE:M17	
PROC SUCHLAUF_ENDE SAVE	
; **ASUB**，用于在程序段搜索后调用换刀程序	
N1000 DEF INT TNR_AKTIV	; 生效 T 号的变量
N1010 DEF INT TNR_VORWAHL	; 预选 T 号的变量
N1020 DEF INT TNR_SUCHLAUF	; 程序段搜索中获取的 T 号 ; 的变量
N1030 TNR_AKTIV = $TC_MPP6[9998,1]	; 读取生效刀具的 T 号
N1040 TNR_SUCHLAUF = $P_TOOLNO	; 读取通过程序段搜索获取的 T 号
N1050 GETSELT(TNR_VORWAHL)	; 读取预选刀具的 T 号
N1060 IF TNR_AKTIV ==TNR_SUCHLAUF GOTOF ASUP_ENDE	
N1070 T = $TC_TP2[TNR_SUCHLAUF]	; 通过刀具名称选择 T
N1080 WZW	; 调用换刀程序
N1090 IF TNR_VORWAHL == TNR_SUCHLAUF GOTOF ASUP_ENDE	
N1100 T = $TC_TP2[TNR_VORWAHL]	; 通过刀具名称恢复 T 预选
ASUP_ENDE:	

程序代码	注释
N1110 M90	; 发送给 PLC 的反馈信息
N1120 REPOSA	; ASUB 结束

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.blockSearchLastActionActive	LBP_Chan*.E_LastActBlock	DB21, ...DBX32.6
<Chan>.basic.in.asupStopStateActive	LBP_Chan*.E_ASUP_Stop	DB21, ...DBX318.0

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.blockSearchAckPlcActionDone	LBP_Chan*.A_PLC_ActCompl	DB21, ...DBX1.6

3.7.6 前提条件

压缩器功能生效时的特性

在程序段搜索**类型 2** 或**类型 4**（在 ... 处**计算**的程序段搜索）中，若目标程序段所处于的程序部分中激活了压缩器功能（COMP...），那么在重定位至轮廓时会逼近通过**压缩器**计算出的轨迹上的位置。这些位置不必与零件程序中编写的轨迹上的位置精确一致。

若压缩时省去了零件程序中编写的程序段，那么在程序段搜索中将无法将这些程序段作为目标程序段找到。会输出报警 15370“在程序段搜索中未找到搜索目标”。

3.8 程序段搜索类型 5（SERUPRO）

3.8.1 功能说明

通过程序段搜索类型 5，即“程序测试”模式下的带计算的程序段搜索（SERUPRO，**“Se**arch-**Ru**n by **Pro**gramtest”），可在选择的中断点实现带计算的跨通道程序段搜索。为此，在 SERUPRO 期间，根据当前的程序协调指令，系统会测定在中断通道中继续执行程序所需的所有状态数据，并于随后将 NC 和 PLC 调整为继续执行程序所需的状态。

在随后继续执行程序重新定位至轮廓前，可通过一个自动启动的用户专用 ASUB 生成可能需要的所有初始状态。

通道

配合 HMI 使用时，SERUPRO 可应用于以下通道范围：

- 仅用于当前的 SERUPRO 通道 (1)
- 用于所有与 SERUPRO 通道具有相同工件名称的通道 (2)
- 用于所有与 SERUPRO 通道归属于相同 BAG 的通道 (3)
- 用于 NCU 的所有通道 (4)

对 SERUPRO 的通道范围选择在 **maschine.ini** 文件中的 [BlockSearch] 部分进行：

[BlockSearch] 部分	**为 HMI 使能程序段搜索功能和选择程序段搜索配置**
SeruproEnabled=1	; SERUPRO 软键在 HMI 中可用。缺省值为 (1)
SeruproConfig=1	; 上文所述通道分组的编号 (1) 至 (4) 。缺省值为 (1)

通过 SERUPRO 启动的所有其他通道均在“自执行 Serupro”模式下运行。只有选择了目标程序段的通道才可通过 SERUPRO 模式下的程序段搜索启动。

激活

SERUPRO 通过 HMI 激活。通过“程序测试轮廓”软键操作 SERUPRO。

SERUPRO 使用 REPOS 定位至目标程序段。

SERUPRO 的时序过程

1. 通过 HMI 操作“程序测试轮廓”软键和搜索目标。
2. NC 在“程序测试”模式下自行启动所选择的程序。
   – 轴在此过程中不运行。
   – 输出辅助功能 $A_OUT 和直接 PLC-IO。
   – 目标程序段的辅助功能则不输出。
3. NC 在目标程序段开始处停止，于系统内部取消程序测试，并显示停止条件“等待：找到搜索目标”。
4. 若存在用户专用 ASUB“PROG_EVENT.SPF”，则其会自动启动。
5. 下一次 NC 启动时会重新定位至轮廓（REPOS）。
   REPOS 进程通过一个系统 ASUB 实现，并可通过“可编辑的 ASUB”功能扩展。

程序段搜索 SERUPRO 的边界条件

SERUPRO 功能只允许在"AUTOMATIC"运行方式下激活，以及在程序状态（通道状态 RESET）下终止。

若在普通模式下仅 PLC 同时启动多个通道，则可在每个通道中通过 SERUPO 对其进行仿真。

采用以下机床数据设置时：
MD10708 $MN_SERUPRO_MASK，位 1 = 0
零件程序指令 START 会使仿真终止，并触发报警 16942
"通道 %1 启动程序指令动作 %2<ALNX> 不可用"。

机床数据：
MD10707 $MN_PROG_TEST_MASK
支持在停止状态下取消程序测试，并且不会影响 SERUPRO 进程。采用缺省设置时，只允许在 RESET 状态下取消功能。

说明

取消程序测试后 REPOS 进程开始，此时 SERUPRO 定位时的限制条件同样适用。可通过一个 ASUB 来消除负面效应。

调整 SERUPRO 特性

对于下列功能示例，可针对 NC 设定 SERUPRO 特性：

- 编程的停止（M0）
- 程序协调指令 START
- 组 SERUPRO
- 跨通道结束 SERUPRO
- 倍率

MD10708 $MN_SERUPRO_MASK = <SERUPRO 中的特性>

SERUPRO 的通道专用初始设置

在一般情形下，可通过以下机床数据定义零件程序启动后的通道专用初始设置：

MD20112 $MC_START_MODE_MASK= <初始设置>

可为 SERUPRO 设定独立的初始设置，来代替 MD20112 中的初始设置：

MD22620 $MC_START_MODE_MASK_PRT = <SERUPRO 初始设置>

SERUPRO 初始设置必须通过以下机床数据显性使能：

MD22621 $MC_ENABLE_START_MODE_MASK_PRT = 1

NC/PLC 接口信号：“程序测试模式下的程序段搜索生效”

程序测试模式下的程序段搜索通过 NC/PLC 接口信号显示：
`<Chan>.basic.in.blockSearchSeruproActive` == 1

该接口信号在程序段搜索启动时置位，直至切换至目标程序段进入主处理。

对于 SERUPRO 进程后的用户自定义 ASUB

提示

若机床制造商决定在 SERUPRO 后按照上文介绍的第 7 点启动一个 ASUB，则须注意以下事项：

第 6 点后的停止状态：

机床数据：
MD11602 $MN_ASUP_START_MASK
和
MD11604 $MN_ASUP_START_PRIO_LEVEL
可支持 NC 从停止状态自行通过 FC9 模块启动 ASUB。

REPOS 程序段结束后再应答 FC9：

REPOS 程序段也已结束时，才可由 FC9 模块通过“ASUB Done”提示 ASUB 结束。

设定的 REPOS 进程在第 8 点后取消：

ASUB 启动时，设定的 REPOS 进程会被取消！

因此，若需保持 REPOS 进程，则应通过 REPOSA 完成 ASUB。

删除不需要的 REPOS 进程：

通过 `M17` 或 `RET` 结束 ASUB，从而删除不需要的 REPOS 进程。

对 ASUB 的特别处理：

对通过 REPOS 结束和从停止状态启动的 ASUB，原则上需要特别处理。

ASUB 会在 REPOS 程序段前自行停止，并通过以下信号予以显示：

`<Chan>.basic.in.asupStopStateActive`（ASUB 停止）

自动 ASUB 启动

采用以下机床数据设置时：
MD11450 $MN_SEARCH_RUN_MODE，位 1 = 1
系统会按照以下流程在 SERUPRO 定位运行中自动启动
/_N_CMA_DIR/_N_PROG_EVENT_SPF
路径下的 ASUB。

1. SERUPRO 进程已完全执行。
2. 用户触发“NC 启动”。
3. 启动 ASUB。
4. NC 在零件程序指令 REPOS **前**自动停止，并显示信息“需通过 NC 启动继续执行程序”。
5. 用户再次按下“NC 启动”。
6. NC 执行重定位运行，并从目标程序段起继续执行零件程序。

说明

通过 MD11450 进行的自动 ASUB 启动需要按下 **Starts** 用于程序的继续执行。
这样一来，其流程特性便与其它程序段搜索类型近似。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.asupStopStateActive	LBP_Chan*.E_ASUP_Stop	DB21, ...DBX318.0
<Chan>.basic.in.blockSearchSeruproActive	LBP_Chan*.E_SearchAct	DB21, ...DBX318.1

3.8.2 重新定位至轮廓（REPOS）

借助“重新定位至轮廓”（REPOS）功能可在中断位置上继续进行已中断的加工。与 REPOS 不同的是，SERUPRO 可“还原”或“重复”程序段。为此，在 SERUPRO 找到目标程序段后，系统会定位至可通过 REPOS 模式选择的位置处的轮廓并继续进行加工。

SERUPRO：设置 REPOS 特性

REPOS 特性（即重定位程序段中的特性）是通过以下机床数据设置的：
MD11470 $MN_REPOS_MODE_MASK = <REPOS 模式>

<REPOS 模式>		
位	**值**	**含义**
0	0	重复已中断的暂停时间
	1	继续已中断的暂停时间
1	-	预留
2	0	重定位程序段中**不会**考虑 <Axis>.basic.out.reposDelay（REPOS 延时）。
	1	重定位程序段中考虑 <Axis>.basic.out.reposDelay（REPOS 延时）。
3	0	**SERUPRO**：在重定位程序段中只运行轨迹轴
	1	**SERUPRO**：在重定位程序段中同时运行轨迹轴和定位轴
4	0	**REPOS**：在重定位程序段中只运行轨迹轴
	1	**REPOS**：在重定位程序段中同时运行轨迹轴和定位轴
5	0	中断期间，经过修改的进给率和主轴转速只会从中断位置后的第一个零件程序段起生效
	1	中断期间，经过修改的进给率和主轴转速从中断位置起生效，故其已在剩余程序段中生效并因此被优先使用。此特性针对每个 REPOS 进程。
6	0	SERUPRO：中性轴和定位主轴在重定位程序段中作为轨迹轴运行。
	1	SERUPRO：中性轴和定位主轴在重定位程序段中作为指令轴运行。 中性轴和定位主轴在 SERUPRO 后重定位。 对于不允许重定位的中性轴而言，必须将 REPOS 延时置位：<Axis>.basic.out.reposDelay = 1（REPOS 延时）
7	0	REPOS 延时**未**使能。
	1	REPOS 延时已使能。 激活了 REPOS 延时（<Axis>.basic.out.reposDelay == 1）且并非几何轴或定向轴的轴在重定位时**不**运行。

⚠ 小心
碰撞危险 MD11470 $MN_REPOS_MODE_MASK，位 3 或位 4 = 1 用户有责任确保在重定位程序段中同时进行轴重定位时，轴不会在机床上产生碰撞。

通过受控的 REPOS 进行重定位

可通过 NC/PLC 接口设定轨迹轴的 REPOS 模式：

<Chan>.basic.out.reposModeSelection（REPOS 模式）

REPOS 模式在 NC 程序中编写，并决定定位特性（参见章节“通过受控 REPOS 重定位至轮廓 (页 177)”）。

各轴的 REPOS 特性也可通过 NC/PLC 接口信号控制，并且必须通过以下机床数据使能：
MD11470 $MN_REPOS_MODE_MASK.位 2 = 1

轨迹轴无法单独控制。对于其他所有非几何轴的轴，可针对单个轴暂时抑制或推延重定位。对于 REPOS 准备运行的通道轴，可通过 NC/PLC 接口信号在稍后的时间点重新使能或继续禁用。

⚠ 危险
碰撞危险 在选择了“抑制单个轴的重定位”的情况下，通过信号 <Axis>.basic.out.resetMovement（删除剩余行程）会引发以下危险特性： MD11470 $MN_REPOS_MODE_MASK.位 2 == 1 若在中断后采用增量方式编写了轴，那么 NC 定位至的位置将与未发生中断时不同。 参见下例：以增量方式编写轴

示例：以增量方式编写回转轴 A

回转轴 A 为第四机床轴。

- 在 REPOS 进程前，回转轴 A 位于 11°
  在中断程序段中，即在 SERUPRO 的目标程序段中，需要将回转轴 A 运行至 27°。
  在任意数量的程序段后，将回转轴 A 增量运行 5°：
  `N1010 POS[A]=IC(5) FA[A]=1000`
  在接口信号 <Axis>.basic.out.reposDelay = 1（REPOS 延时）置位的情况下，回转轴 A 在 REPOS 进程中不运行，并通过 `N1010` 向 32° 运行。必要时用户须对从 11° 向 27° 的行程进行应答。

  ⚠ 危险

  碰撞危险
  由于在中断后以增量方式编写轴，其向 16° 运行，而非 32°。

- 单独启动轴
  包含多轴的 SERUPRO 定位的 REPOS 特性通过以下机床数据选择：
  MD11470 $MN_REPOS_MODE_MASK.位 3 = 1
  NC 通过一个程序段启动 SERUPRO 定位，其会将**所有**定位轴运行至编写的终点，并将轨迹轴运行至目标程序段。
  用户通过选择相应进给使能来启动单个轴。接下来便是目标程序段的运行。

- 在重定位程序段中对定位轴进行重定位
  定位轴不在剩余程序段中，而是在重定位程序段中进行重定位，并且不仅仅适用于程序测试下的程序段搜索中的 SERUPRO 定位：
  MD11470 $MN_REPOS_MODE_MASK.位 3 = 1：用于程序测试模式下的程序段搜索（SERUPRO）
  MD11470 $MN_REPOS_MODE_MASK.位 4 = 1:用于每个 REPOS

  说明
  若位 3 和位 4 均未置位，那么此阶段会在剩余程序段中对非轨迹轴的轴进行重定位。

通过 REPOS 偏移延迟轴定位

若轴专用接口信号 `<Axis>.basic.out.reposDelay`（REPOS 延时）置位，则在下一次编写时才会通过 `<Chan>.basic.out.reposModeActivationRequest`（REPOS 模式变更）的上升沿运行该轴的 REPOS 偏移。

可使用同步动作通过 $AA_REPOS_DELAY 确定该轴当前是否处于 REPOS 偏移下。

⚠ 小心

碰撞危险

<Axis>.basic.out.reposDelay（REPOS 延时）对形成轨迹的机床轴无效。
可通过 <Axis>.basic.in.isPathAxis（轨迹轴）确定轴是否为轨迹轴。

REPOS 信号的接收时间点

通过 <Chan>.basic.out.reposModeActivationRequest （REPOS 模式变更）的上升沿将以下 REPOS 信号接收至 NC：

- 通道专用：<Chan>.basic.out.reposModeSelection（REPOS 模式）
- 轴专用：<Axis>.basic.out.reposDelay（REPOS 延时）

REPOS 信号的电平基于当前的主处理程序段。此时可分为以下两种情形：

1. 主处理中存在**一个**当前生效 REPOS 进程的重定位程序段。
   运行中的 REPOS 进程会终止并重新启动，且 REPOS 偏移通过上述 REPOS 信号调整。
2. 主处理中**无**当前生效 REPOS 进程的重定位程序段。
   未来每一个需要重定位至当前主处理程序段的 REPOS 进程均通过上述 REPOS 信号调整。

说明

在运行的 ASUB 中，<Chan>.basic.out.reposModeActivationRequest（REPOS 模式变更）不对最后的 REPOS 起作用，在执行 REPOS 程序段的时间点上，该信号随机置位。

在第 1 种情形中，信号只适用于停止状态。

RESET 时的特性：

- NC 已应答 PLC 信号：
  <Chan>.basic.out.reposModeActivationRequest == 1（REPOS 模式变更）**和**
  <Chan>.basic.in.reposModeChangeAck == 1（REPOS 模式变更应答）
  若在此情形下发生通道复位，则系统将生效的 REPOS 模式删除：
  <Chan>.basic.in.reposModeSelectedNumber = 0（生效的 REPOS 模式）
- NC 尚未应答 PLC 信号：
  <Chan>.basic.out.reposModeActivationRequest == 1（REPOS 模式变更）**和**
  <Chan>.basic.in.reposModeChangeAck == 0（REPOS 模式变更应答）
  若在此情形下发生通道复位，则系统将 REPOS 模式变更应答和生效的 REPOS 模式删除：
  <Chan>.basic.in.reposModeChangeAck = 0（REPOS 模式变更应答）
  <Chan>.basic.in.reposModeSelectedNumber = 0（生效的 REPOS 模式）

通过 NC/PLC 接口信号控制 SERUPRO 定位

可通过 <Chan>.basic.out.reposModeActivationRequest（REPOS 模式变更）和对应信号在以下阶段使用 SERUPRO 定位：

- 从“找到搜索目标”到“SERUPRO ASUB 启动”
- 从“SERUPO ASUB 在 REPOS 前自动停止”到“执行目标程序段”

执行 SERUPRO ASUB（例如在 REPOS 前的程序部分）时，接口信号对 SERUPRO 定位不生效。

包含 NC/PLC 接口信号的 REPOS 进程

通过 NC/PLC 接口信号控制 REPOS

可通过以下 NC/PLC 接口信号控制 REPOS 偏移：

- <Chan>.basic.out.reposModeSelection（REPOS 模式）
- <Chan>.basic.out.reposModeActivationRequest（REPOS 模式变更）
- <Axis>.basic.out.reposDelay（REPOS 延时）
- <Axis>.basic.out.reposDelay（REPOS 延时）

REPOS 应答信号

使用以下 NC/PLC 接口信号**通过 NC** 对借助 PLC 调整 REPOS 特性的功能进行应答：

- <Chan>.basic.in.reposModeChangeAck （REPOS 模式变更应答）
- <Chan>.basic.in.reposModeSelectedNumber（生效的 REPOS 模式）
- <Chan>.basic.in.reposDelayActive（REPOS 延迟应答）
- <Axis>.basic.in.reposOffsetActive（REPOS 偏移）
- <Axis>.basic.in.reposOffsetValid （REPOS 偏移有效）
- <Axis>.basic.in.reposDelayAck（REPOS 延迟应答）
- <Axis>.basic.in.isPathAxis（轨迹轴）

更多相关信息请见“接口中的 REPOS 偏移”部分。

REPOS 应答进程

若 NC 识别出"REPOS 模式变更"（<Chan>.basic.out.reposModeActivationRequest == 1），则借助 PLC 通过 <Chan>.basic.in.reposModeChangeAck = 1 对其进行应答。

说明

若 NC 尚未通过接口信号 <Chan>.basic.out.reposModeActivationRequest（REPOS 模式变更应答）对接口信号 <Chan>.basic.in.reposModeChangeAck（REPOS 模式变更）进行应答，
那么在此情形下的通道复位会使程序终止，并不会执行 REPOS 来控制 REPOS 模式。

NC 通过以下接口信号应答由 PLC 设定的 REPOS 模式：

- <Chan>.basic.in.reposModeSelectedNumber（生效的 REPOS 模式）
- <Axis>.basic.out.reposDelay（REPOS 延时）
- <Axis>.basic.in.reposDelayAck（REPOS 延迟应答）

示例

- 时间点 ②：在程序段 N20 中通过 NC 停止将 NC 程序停止。所有轴通过为其设置的制动斜坡制动至静止状态。
- 时间点 ③：在 PLC 用户程序将"REPOS 模式"置位后，NC 通过"REPOS 模式变更"的 0/1 脉冲沿接收 REPOS 模式。
- 时间点 ④："REPOS 模式变更应答"始终保持置位，直至触发 ASUB。
- 时间点 ⑤：ASUB 中的 REPOS 进程开始。
- 时间点 ⑥：重新转向 ASUB 的剩余程序段。

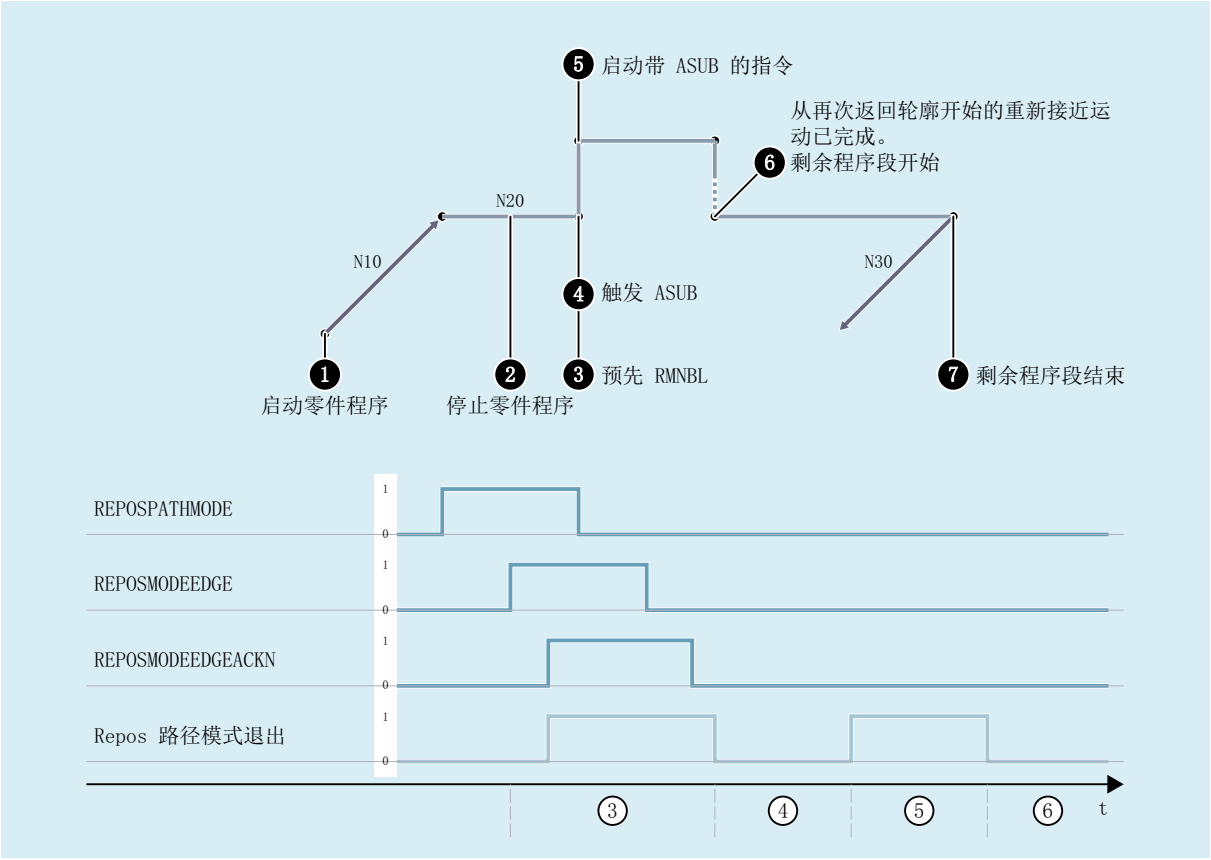


图 3-5　零件程序中的 REPOS 流程，以及按时序排列的 NC 应答信号

NC 重新设置应答

此阶段中 REPOSPATHMODE 继续生效（→ 时间点 (2) 停止的程序的剩余程序段尚未执行完毕）。

一旦执行 ASUB 的 REPOS 重定位运行，NC 便会立即重新设置 “Repos Path Mode Quitt”（→ 时间点 (5)）。若未预先通过 NC/PLC 接口信号选择 REPOSPATHMODE，则会显示编写的 REPOS 模式。

“Repos Path Mode Quit” 会在切换至剩余程序段时撤销（→ 时间点 (6)）。
→ 时间点 (2) 处后续的零件程序段 N30 将继续执行。

接口信号：
<Axis>.basic.in.reposDelayAck（REPOS 延时应答）以类似的方式定义。

<Axis>.basic.in.reposOffsetValid（REPOS 偏移生效）= 1，若：

<Chan>.basic.in.reposModeSelectedNumber （生效的 REPOS 模式）= 4 (RMNBL)。

有效的 REPOS 偏移

SERUPRO 进程结束时，用户可通过轴/主轴专用 NC/PLC 接口信号（NC→PLC）：
<Axis>.basic.in.reposOffsetActive（REPOS 偏移）读取 REPOS 偏移。

此信号对轴的作用如下：

值 0： 不启用 REPOS 偏移。
值 1： 启用 REPOS 偏移。

有效范围

接口信号：
<Axis>.basic.in.reposOffsetActive（REPOS 偏移）
在 SERUPRO 进程结束时提供。

启动 SERUPRO ASUB 或自动启动 ASUB 时 REPOS 偏移失效。

更新有效范围内的 REPOS 偏移

在 SERUPRO 结束和启动之间，可通过模式切换在 JOG 下运行轴。

用户需要在 JOG 模式下手动运行 REPOS 偏移，从而将 NST
<Axis>.basic.in.reposOffsetActive（REPOS 偏移）设为 0 值。

在有效范围内也可通过 SINU_TraversePosAxis（Basic Program FC18）运行轴，此时 NST
<Axis>.basic.in.reposOffsetActive（REPOS 偏移）会持续更新。

显示有效范围

REPOS 偏移的有效范围通过以下接口信号显示：

<Axis>.basic.in.reposOffsetValid （REPOS 偏移有效）

该信号可指示计算的有效性：

值 0： 该轴的 REPOS 偏移计算正确。
值 1： 该轴的 REPOS 无法计算，因为尚未进行 REPOS（例如处于 ASUB 结尾）或无 REPOS 生效。

跨通道取轴后的 REPOS 偏移

通过汇总信号 <Chan>.basic.in.reposDelayActive（REPOS 延时）可确定是否发生了有效的 REPOS 偏移：

值 0： 该通道当前控制的所有轴无 REPOS 偏移，或其 REPOS 偏移无效。

值 1： 其它。

经过同步的同步主轴耦合中的 REPOS 偏移

在通过 SERUPRO 重定位时，会先预处理至中断位置。若已对同步主轴耦合进行同步，那么将不存在跟随轴的 REPOS 偏移，也不会有同步行程。同步信号保持置位。

程序段切换时找到搜索目标

当轴归属于轨迹组时，轴专用 NC/PLC 接口信号 <Axis>.basic.in.isPathAxis（轨迹轴）为 1。

此信号用于显示程序段切换时当前待执行程序段的状态。之后的状态变化则不考虑。

SERUPRO 进程通过“找到搜索目标”结束时，<Axis>.basic.in.isPathAxis（轨迹轴）基于目标程序段。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Axis>.basic.in.isPathAxis	LBP_Axis*.E_PathAxis	DB31, ...DBX76.4
<Axis>.basic.in.reposOffsetActive	LBP_Axis*.E_ReposSh	DB31, ...DBX70.0
<Axis>.basic.in.reposOffsetValid	LBP_Axis*.E_ReposShValid	DB31, ...DBX70.1
<Axis>.basic.in.reposDelayAck	LBP_Axis*.E_ReposDelayQuit	DB31, ...DBX70.2
<Chan>.basic.in.reposModeChangeAck	LBP_Chan*.E_REPOS_EdgeAckn	DB21, ...DBX319.0
<Chan>.basic.in.reposModeSelectedNumber	LBP_Chan*.E_REPOS_PMode0	DB21, ...DBX319.1 ..3
<Chan>.basic.in.reposDelayActive	LBP_Chan*.E_REPOS_DEFERRA	DB21, ...DBX319.5

PLC → NC

Basic Program Plus	Basic Program	
<Axis>.basic.out.resetMovement	LBP_AxisX.A_DelDTGSpReset	DB31, ...DBX2.2
<Axis>.basic.out.reposDelay	LBP_AxisX.A_REPOSDelay	DB31, ...DBX10.0
<Chan>.basic.out.reposModeSelection	LBP_ChanX.A_REPOSPM_0	DB21, ...DBX31.0..2
<Chan>.basic.out.reposModeActivationRequest	LBP_Chan*.A_REPOSMode	DB21, ...DBX31.4

3.8.2.1 通过受控 REPOS 重定位至轮廓

在通过 SERUPRO 找到目标程序段后，系统会在继续执行中断的程序前先执行一个重新定位至轮廓的 REPOS 进程。缺省设置下，REPOS 模式“重新定位至目标程序段的程序段起点”(RMBBL) 生效。可通过以下 NC/PLC 接口针对特定用户设定 REPOS 模式：

`<Chan>.basic.out.reposModeSelection`（REPOS 模式）

更多信息

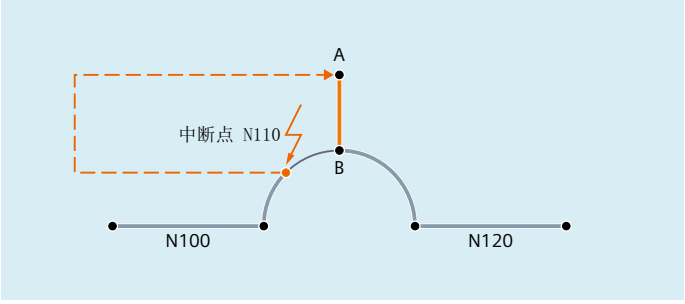
对接口信号的详细说明请见 PLC 功能手册。

REPOS 模式：重新定位至下一个轨迹点 (RMNBL)

在 REPOS 模式 `RMNBL` 下，系统会从 REPOS 起始位置开始定位至下一个轮廓点。

示例

程序在程序段 N110 中的任意位置中断。随后例如以手动方式将轴运行至位置（A）。在 SERUPRO 找到目标程序段 N110 后，以 REPOS 模式 `RMNBL` 执行 REPOS 进程。从 REPOS 起始位置（A）出发，下一轮廓点为点（B）。到达点（B）时，REPOS 进程完成。从点（B）起重新运行中断程序的编写的轮廓。



通过 NC/PLC 接口设定 REPOS 模式

REPOS 模式可通过以下 NC/PLC 接口信号设定：
<Chan>.basic.out.reposModeSelection（REPOS 模式）

说明

RMNBL 是一个通用的 REPOS 扩展功能，不仅限用于 SERUPRO。

RMIBL 和 RMBBL 的作用与 SERUPRO 的一致。

<Chan>.basic.out.reposModeSelection（REPOS 模式）只会影响轨迹轴的运行。

其它轴的特性可通过接口信号 <Axis>.basic.out.reposDelay（REPOSDELAY）单独修改。此 REPOS 偏移不会立即运行，而是在下一次编写时才运行。

重定位点编程的更多信息请见：

更多信息
编程手册“NC 编程”

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<Axis>.basic.out.reposDelay	LBP_Axis*.A_REPOSDelay	DB31, ...DBX10.0
<Chan>.basic.out.reposModeSelection	LBP_Chan*.A_REPOSPM_0	DB21, ...DBX31.0..2

3.8.3 搜索加速

机床数据设定

SERUPRO 进程的执行速度可通过以下机床数据调整

MD22600 $MC_SERUPRO_SPEED_MODE = <值>

<值>	含义
0	程序测试采用程序段搜索速度/空运行进给速度： • 进给轴 MD22601 $MC_SERUPRO_SPEED_FACTOR * 空运行进给率 • 主轴：MD22601 $MC_SERUPRO_SPEED_FACTOR * 编写的转速 不考虑轴/主轴的动态限制。
1	程序测试采用编写的速度： • 进给轴空运行进给 • 主轴：编写的转速 遵循轴/主轴的动态限制。
2	程序测试采用空运行进给率 程序测试下以编写的速度/转速运行。 遵循轴/主轴的动态限制。
3	程序测试采用程序段搜索速度 程序测试下采用以下速度运行： • 进给轴 MD22601 $MC_SERUPRO_SPEED_FACTOR * 编写的进给率 • 主轴：MD22601 $MC_SERUPRO_SPEED_FACTOR * 编写的转速 不考虑轴/主轴的动态限制。 **提示** 旋转进给率（例如 G95）生效时，编写的进给率不与 MD22601 $MC_SERUPRO_SPEED_FACTOR 相乘，只采用编写的主轴转速。这样一来有效轨迹速度会提升 MD22601 $MC_SERUPRO_SPEED_FACTOR。

前提条件

主处理轴

SERUPRO 中 MD22600 $MC_SERUPRO_SPEED_MODE 对下列主处理轴生效：

- PLC 轴
- 指令轴
- 定位轴
- 往复轴

同步动作

注意
SERUPRO 中有时不会执行同步动作的动作 SERUPRO 期间系统内部生成的实际值（例如轴位置）与常规程序运行中不同，因此系统可能会认定验证实际值（例如轴位置）的同步动作条件不再为真（TRUE），并因而不再执行同步动作的动作部分。

旋转进给率

DryRun 下 MD22601 $MC_SERUPRO_SPEED_FACTOR 的作用：

- 从 G95/G96/G961/G97/G971 切换至 G94
- 攻丝和螺纹切削：正常的 DryRun 速度

刚性攻丝

- “刚性攻丝”（G331/G332）时，主轴在位置闭环控制下以轨迹组插补。此时会设定钻孔深度（直线轴）、螺纹螺距和转速（主轴）。
  DryRun 期间会设定直线轴的速度，转速保持恒定，螺纹螺距会被相应调整。
  SERUPRO 后得到的主轴位置会与常规运行有偏差，因为 SERUPRO 中主轴旋转相对较少。

3.8.4 SERUPRO ASUB

SERUPRO ASUB 的特殊性

使用 SERUPRO ASUB 时须注意以下方面的特殊性：

- 回参考点运行：通过零件程序 G74
- 刀具管理：换刀和刀库数据
- 主轴加速：启动 SERUPRO ASUB 时

G74 回参考点运行

若程序起点和搜索目标之间有 G74（回参考点运行）指令，则会被 NC 忽略。

SERUPRO 定位不会考虑 G74 指令！

刀具管理

刀具管理生效时，建议采用以下设置：

MD20310 $MC_TOOL_MANAGEMENT_MASK 位 20 = 0

这样一来，SERUPRO 进程中产生的刀具管理指令便**不会**输出至 PLC！

刀具管理指令如下生效：

- NC 自动应答指令。
- 刀库数据不会被修改。
- 刀具数据不会被修改。
  特例：
  测试模式下激活的刀具可能会变为“生效”状态。因此 SERUPRO 进程后主轴上可能会有错误的刀具。
  解决办法：
  用户启动一个实际运行的 SERUPRO ASUB。启动前用户可启动一个 ASUB 来换入正确的刀具。

SERUPRO 进程：流程中第 2 点至第 6 点
SERUPRO ASUB：流程中第 7 点。

此外必须设置机床数据 MD20310 $MC_TOOL_MANAGEMENT_MASK 位 11 = 1，因为 ASUB 必要时须重复 T 选择。

配备刀具管理和副主轴的设备不支持 SERUPRO！

示例

换刀子程序

程序代码	**注释**
PROC L6	; 换刀程序
N500 DEF INT TNR_AKTUELL	; 生效 T 号的变量
N510 DEF INT TNR_VORWAHL	; 预选 T 号的变量
	; 确定当前刀具
N520 STOPRE	; 程序测试模式下
N530 IF $P_ISTEST	; 从程序上下文
N540 TNR_AKTUELL = $P_TOOLNO	; 读取“当前”刀具。
N550 ELSE	; 否则会读取主轴的刀具。
N560 TNR_AKTUELL = $TC_MPP6[9998,1]	; 读取主轴上刀具的 T 号
N570 ENDIF	
N580 GETSELT(TNR_VORWAHL)	; 读取预选的主主轴刀具的 T 号
	; 仅当刀具当前未生效时才执行换刀
N590 IF TNR_AKTUELL <> TNR_VORWAHL	; 返回刀具更换点

程序代码	注释
N600 G0 G40 G60 G90 SUPA X450 Y300 Z300 D0	
N610 M206	；执行刀具更换
N620 ENDIF	
N630 M17	

ASUB，用于在程序段搜索类型 5 后调用换刀程序

程序代码	注释
PROC ASUPWZV2	
N1000 DEF INT TNR_SPINDEL	；生效 T 号的变量
N1010 DEF INT TNR_VORWAHL	；预选 T 号的变量
N1020 DEF INT TNR_SUCHLAUF	；程序段搜索中获取的 T 号的变量
N1030 TNR_SPINDEL = $TC_MPP6[9998,1]	；读取主轴上刀具的 T 号
N1040 TNR_SUCHLAUF = $P_TOOLNO	；读取通过程序段搜索获取的 T 号，
	；即该刀具决定
	；当前刀具补偿
N1050 GETSELT(TNR_VORWAHL)	；读取预选刀具的 T 号
N1060 IF TNR_SPINDEL ==TNR_SUCHLAUF GOTOF ASUP_ENDE1	
N1070 T = $TC_TP2[TNR_SUCHLAUF]	；通过刀具名称选择 T
N1080 L6	；调用换刀程序
N1085 ASUP_ENDE1:	
N1090 IF TNR_VORWAHL == TNR_SUCHLAUF GOTOF ASUP_ENDE	
N1100 T = $TC_TP2[TNR_VORWAHL]	；通过刀具名称恢复 T 预选
N1110 ASUP_ENDE:	
N1110 M90	；发送给 PLC 的反馈信息
N1120 REPOSA	；ASUB 结束

在 PROC L6 和 PROC ASUPWZV2 这两个程序中，换刀均采用 M206 取代 M6 编写。
ASUB 程序“ASUPWZV2”使用了各种系统变量，其一方面能够识别程序进度（$P_TOOLNO），另一反面又能显示机床的当前状态（$TC_MPP6[9998,1] ）。

主轴启动

启动 SERUPRO ASUB 时，主轴不会加速至程序中设定的转速，因为 SERUPRO ASUB 用于在换刀后将新道具校准至正确的工件位置。

通过 SERUPRO ASUB 如下执行主轴加速：

- SERUPRO 进程已完全结束。
- 用户通过 FC-9 模块启动 SERUPRO ASUB，
  用于在必要时加速主轴。
- ASUB 中 M0 后的启动不会修改主轴状态。
- SERUPRO ASUB 在 REPOS 零件程序段前自动停止。
- 用户按下 START。
- 只要 ASUB 中未对主轴进行其它编程，主轴便会加速至目标程序段状态。

说明

调整主轴 REPOS：

在调整 SERUPRO 定位和主轴功能时，须将转速控制模式和定位模式的过渡纳入考量。主轴运行方式切换的更多信息请见“运行方式和运行方式切换 (页 36)”章节。

3.8.5 自动执行的 SERUPRO

自动执行的 SERUPRO

通过使用通道专用功能“自动执行的 SERUPRO”，**无需**在相关 SERUPRO 通道的程序中事先定义搜索目标便可实现 SERUPRO。

此外可为**每个**“自动执行的 SERUPRO”定义一个“serurpoMasterChan”特殊通道。可在该通道中定义一个搜索目标。

“自动执行的 SERUPRO”功能支持跨通道的 SERUPRO 程序段搜索。

功能

“自动执行的 SERUPRO”进程无法找到搜索目标。若未到达搜索目标，那么通道也不会停止。不过特定情形下通道会暂时停止。此时该通道通常会等待另一个通道。此类情形例如包括：等待标记、耦合或跨通道取轴。

等待阶段发生：

在此等段阶段中，NC 会检查“seruproMasterChan”通道是否已到达一个搜索目标。未到达搜索目标时会再次退出等待阶段。

若到达了搜索目标，
那么 SERUPRO 进程在该通道中亦结束。“serupro-MasterChan”通道须已在通常 SERUPRO 模式下启动。

无等待阶段发生：

“自动执行的 SERUPRO”通过零件程序的 M30 结束。

之后通道重新恢复复位状态。

不会发生 SERUPRO 运行。

启动一组通道

若只通过“自动执行的 SERUPRO”来启动一组通道，那么所有通道均通过“RESET”结束。

特例情形：
通道等待一个根本未启动的协作通道。

可如下执行跨通道程序段搜索：

- 用户通过 HMI 选择须共同运作的通道（通道组）。
- 用户从通道组中选取一个想为其选定搜索目标的重要通道（目标通道）。
- 这样一来 HMI 会在目标通道上启动 SERUPRO，并在通道组的其它通道上启动“自动执行的 SERUPRO”。

当涉及的**每条**通道均清除了“seruproActive”时，进程结束。

“自动执行的 SERUPRO”不接受另一个 NCU 上的主主轴。

激活

通过 HMI 激活“自动执行的 SERUPRO”，作为针对目标通道“seruproMasterChan”的程序段搜索类型 5 的程序段搜索启动。

对于通过目标通道启动的相关通道，无需为其设定搜索目标。

3.8.6 禁用程序部分用于恢复执行

编写的中断指示

由于生产或过程工艺的原因，若特定程序部分内发生程序中断时预计无法恢复执行，可为搜索的目标程序段禁用该程序部分。

在程序中断后，若在为恢复执行而禁用的程序部分中进行了对中断位置的程序段搜索，那么控制系统会将禁用区域前最后的可执行程序段（主处理程序段）用作目标程序段（停止程序段）。

编程

句法

```
IPTRLOCK()
```

功能

标记需要禁用恢复执行的程序部分的起点。对于“在中断位置恢复执行”的程序段搜索，下一个 IPTRLOCK 生效的可执行程序段（主处理程序段）将被用作目标程序段，直至通过 IPTRUNLOCK 使能。该程序段下文称作**停止程序段**。

有效性：模态有效

句法

```
IPTRUNLOCK()
```

标记为恢复执行禁用的程序部分的终点。对于“在中断位置恢复执行”的程序段搜索，从下一个 IPTRLOCK 生效的可执行程序段（主处理程序段）起，当前程序段重新用作目标程序段。该程序段下文称作**使能程序段**。

有效性：模态有效

示例

程序代码	**注释**
...	
N010 IPTRLOCK()	；禁用区域：起始
N020 R1=R1+1	
N030 G4 F1	**；停止程序段**
...	；禁用区域
N200 IPTRUNLOCK()	；禁用区域：结束
N220 R1=R1+1	
N230 G4 F1	**；使能程序段**
...	

前提条件

- 程序（*.MPF，*.SPF）中的 IPTRLOCK 最多生效至程序结束（M30、M17、RET）。程序结束时 IPTRUNLOCK 会隐性生效。
- 在一个程序内多次编写 IPTRLOCK 时，效果不会累加。在程序内首次编写 IPTRUNLOCK 或达到程序末尾时，之前的所有 IPTRLOCK 调用将结束。
- 若在禁用区域内进行了子程序调用，那么恢复执行禁用同样对该子程序级及其后可能存在的所有程序级生效。即使在调用的子程序中显性编写 IPTRUNLOCK，禁用仍不会被取消。

示例：两个程序级中禁用程序部分的套用

通过在 PROG_1 中激活恢复执行禁用，同时对 PROG_2 及其后可能存在的所有程序级禁用恢复执行。

程序代码	注释
PROC PROG_1	; 程序 1
...	
N010 IPTRLOCK()	
N020 R1=R1+1	
N030 G4 F1	**; 停止程序段**
...	; 禁用区域：起始
N040 PROG_2	; 禁用区域
...	; 禁用区域：结束
N050 IPTRUNLOCK()	
N060 R2=R2+2	
N070 G4 F1	**; 使能程序段**
...	

程序代码	注释
PROC PROG_2	; 程序 2
N210 IPTRLOCK()	; 由于程序 1 而不生效
...	
N250 IPTRUNLOCK()	; 由于程序 1 而不生效
...	
N280 RET	; 由于程序 1 而不生效

示例 3：多重编写 IPTRLOCK

程序代码	注释
PROC PROG_1	; 程序 1
...	
N010 IPTRLOCK()	

程序代码	注释
N020 R1=R1+1	
N030 G4 F1	**；停止程序段**
...	；禁用区域：起始
N150 IPTRLOCK()	；禁用区域
...	；禁用区域
N250 IPTRLOCK()	；禁用区域
...	；禁用区域：结束
N360 IPTRUNLOCK()	
N370 R2=R2+2	
N380 G4 F1	**；使能程序段**
...	

系统变量

系统变量 $P_IPTRLOCK 可用于获取当前程序段的状态：

$P_IPTRLOCK	含义
FALSE	当前程序段**不处于**为恢复执行禁用的程序部分中。
TRUE	当前程序段**处于**为恢复执行禁用的程序部分中。

自动功能专用恢复执行禁用

对于各种耦合，恢复执行禁用可自动随耦合的启用/撤销而激活或取消：

MD22680 $MC_AUTO_IPTR_LOCK，位 x

位	值	含义
0	电子齿轮箱（EGON / EGOF）	
	1	自动恢复执行禁用生效
	0	自动恢复执行禁用不生效
1	轴向引导值耦合 (LEADON / LEADOF)	
	1	自动恢复执行禁用生效
	0	自动恢复执行禁用不生效

该程序区域从激活**前**的最后一个可执行程序段开始，并在取消处结束。

自动中断指示对通过同步动作激活或取消的耦合不生效。

示例:轴向引导值耦合自动定义为不可搜索：

程序代码	注释
N100 G0 X100	
N200 EGON(Y,"NOC",X,1,1)	；不可搜索的程序部分开始。
N300 LEADON(A,B,1)	
...	
N400 EGOFS(Y)	
...	
N500 LEADOF(A,B)	；不可搜索的程序部分结束。
N600 G0 X200	

不可搜索的程序区域（N200 - N500）中的程序中断总是通过 N100 提供中断指示。

注意
功能同时启用可导致异常 通过机床数据同时启用了“可编程中断指示”和“自动中断指示”时，NC 会选择其中最大的不可搜索区域。 一个程序可能在整个运行时间内都需要启用耦合。此时自动中断指示将总是指向程序开始处，而 SERUPRO 功能事实上将无作用。

3.8.7 上电、运行方式切换和复位时的特性

SERUPRO 在上电时不生效。SERUPRO 期间允许进行运行方式切换。复位会使 SERUPRO 终止，内部选择的程序测试会被重新取消。SERUPRO 不可与其它程序段搜索类型组合使用。

3.8.8 前提条件

3.8.8.1 目标程序段中的 STOPRE

STOPRE 程序段从前一程序段获取所有跨程序段设置，因此可针对以下情形将原程序段前的条件纳入考量：

- 将当前执行的程序行与主处理同步。
- 为 SERUPRO 退刀跨程序段设置，例如用于在 SERUPRO 定位时调整 REPOS 运行。

示例：通过 X 轴的设定值给定定位 Z 轴。

系统解释程序段"G1 F100 Z=$AA_IM[X]"时，前一 STOPRE 程序段用于实现与主处理的同步。这样便可通过 $AA_IM 读取正确的 X 轴设定值，从而将 Z 轴运行至相同位置。

示例：读取并正确计算外部零点偏移。

```
N10 G1 X1000 F100
N20 G1 X1000 F500
N30 G1 X1000 F1000
N40 G1 X1000 F5000
N50 SUPA G1 F100 X200              ；将外部零点偏移向 200 运行
N60 G0 X1000
N70 ...
```

通过 N50 前的隐性 STOPRE，NC 可读取并正确计算当前零点偏移。
在以 N50 为目标的 SERUPRO 搜索中，系统会在 SERUPRO 定位中重新定位至隐性 STOPRE，并从 N40 通过 F5000 确定速度。

隐性预处理停止

取消隐性预处理停止的情形：

1. 在所有会发生以下变量访问的程序段中：
   - 编写以 $A... 开头的系统变量
   - 编程一个属性为 SYNR / SYNRW 的重定义变量
2. 对于以下指令：
   - 零件程序指令 MEACALC、MEASURE
   - 编程 SUPA（抑制框架和在线补偿）
   - 编程 CTABDEF（开始曲线图表定义）
   - 零件程序指令 WRITE/DELETE（写入/删除文件）
   - 在此类指令序列的第一个 WRITE/DELETE 指令前
   - 零件程序指令 EXTCALL
   - 零件程序指令 GETSELT、GETEXET
   - 在换刀和刀具精补偿 FTOCON 激活时
3. 对于以下指令编辑：
   - 结束程序段搜索，类型 1（"无计算搜索程序段"）
   - 结束程序段搜索，类型 2，带计算（"在轮廓终点上搜索程序段"）

3.8.8.2 目标程序段中的 SPOS

若通过 M3 / M4 编程了主轴，并在目标程序段中切换至 SPOS，那么主轴会在 SERUPRO 进程结束（找到搜索目标）时切换至 SPOS。

<Axis>.spindle.in.posModeActive = TRUE（有效主轴运行方式：定位模式）

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Axis>.spindle.in.posModeActive	LBP_Axis*.E_PosMode	DB21, ...DBX84.5

3.8.8.3 运行到固定点（FXS）

在重新运行至轮廓（REPOS）时，系统会自动重复“运行到固定点”（FXS）功能。此时每根轴都会被包含在内。力矩将采用搜索目标前最后编写的力矩。

系统变量

“运行到固定点”的系统变量在 SERUPRO 中的含义如下：

- $AA_FXS：程序仿真的进度
- $VA_FXS：实际机床状态

在 SERUPRO 功能**外**，这两个系统变量总是位相同的值。

ASUB

可为 SERUPRO 激活一个用户专用 ASUB。

更多信息

程序段搜索 SERUPRO 的详细信息请见功能手册“进给轴和主轴”；运行到固定挡块。

3.8.8.4 以限制的力矩/力运行（FOC）

在重新运行至轮廓（REPOS）时，系统会自动重复“以限制的力矩/力运行”（FOC）功能。此时每根轴都会被包含在内。力矩将采用搜索目标前最后编写的力矩。

系统变量

“以限制的力矩/力运行”的系统变量在 SERUPRO 中的含义如下：

- $AA_FOC：程序仿真的进度
- $VA_FOC：实际机床状态

前提条件

在重新运行至轮廓时，**无法**实现持续变化的力矩特性曲线。

示例

程序将 X 轴从位置 0 运行至 100，并对每 20 个增量中的 10 个增量启用“以限制的力矩/力运行”（FOC）。此力矩特性曲线通常通过逐段生效的 FOC 生成，并且无法在重新运行至轮廓（REPOS）时实现。此时会在限制力矩/力或无限制的情况下根据最后的编程将 X 轴从 0 运行至 100。

更多信息

程序段搜索 SERUPRO 的详细信息请见功能手册“进给轴和主轴”；运行到固定挡块。

3.8.8.5 同步主轴

可对同步主轴进行仿真。

在现有的所有通道中均可通过 SERUPRO 对包含一根引导主轴和任意根跟随主轴的同步主轴运行进行仿真。

更多信息

同步主轴的更多信息请见功能手册之扩展功能；同步主轴

3.8.8.6 轴耦合

SERUPRO 是程序测试模式下的程序仿真，可用于对设定值耦合和实际值耦合进行仿真。

通过 SERUPRO 对轴耦合进行仿真时，系统总是将其认定为设定值耦合。此时会为**所有**轴计算终点，这些点用作 SERUPRO 定位的目标点。“找到搜索目标”的同时耦合即已生效。从当前点到终点的行程在耦合生效的情形下在 SERUPRO 定位中执行。

引导值耦合

LEADON

相应的，对于轴向引导值耦合的仿真，以下定义适用：

1. 总是通过设定值耦合进行仿真。
2. SERUPRO 定位在耦合生效且跟随轴叠加运行的情形下进行，用于达到仿真的目标点。

只通过耦合运行的跟随轴不是总能到达目标点。在 SERUPRO 定位中会为跟随轴计算出一个叠加线性运动，以定位至仿真点！

通过 JOG 到达 LEAD 的仿真目标点

在“找到搜索目标”的时间点耦合已生效，特别是对于 JOG 运行。在未到达目标点时，SERUPRO 定位中可通过生效的耦合及叠加运动将跟随轴运行至目标点。

说明

轴耦合重新定位的更多信息参见“重新定位至轮廓（REPOS） (页 167)”章节。

电子齿轮箱 (EG)

对于 EG 仿真，以下定义适用：

1. 总是通过设定值耦合进行仿真。
2. 若仅个别引导轴，而不是所有引导轴均处于 SERUPRO 下，那么仿真会被终止，并触发报警 16952“ResetClear/NoStart”。在采用跨通道耦合时可能会出现此情形。
3. 对于从 NC 方面看来只有一个编码器且由外部运行的轴无法实现正确的仿真。特别需要注意的是，这些轴不能参与耦合。

⚠ 小心

仿真错误

为了能实现对耦合的正确仿真，必须事先取消耦合。

这可通过机床数据 MD10708 $MA_SERUPRO_MASK 进行。

转速/转矩耦合（主从耦合）

程序段搜索结束后，可自动启动一个系统 ASUB。在该 ASUB 中，用户可对耦合状态及对应的轴位置进行后续调整。为此所需的信息通过以下系统变量获取：

系统变量	说明
$P_SEARCH_MASLD[<从动轴>]	耦合关闭时间点从动轴和主动轴间的位置偏移。
$AA_MASL_STAT[<从动轴>]	主从耦合的当前状态
$P_SEARCH_MASLC[<从动轴>]	状态：耦合状态已在程序段搜索中变化
在通过 MASLON 激活耦合时，这些系统变量会被清除。	

说明

在程序段搜索的时间点，参与耦合的轴必须处于同一通道中。

更多信息

有关主从耦合的更多信息请见功能手册“轴和主轴”，“轴耦合”章节

示例

- **系统 ASUB**
  - 路径和名称：/_N_CMA_DIR/PROGEVENT.SPF
  - 主动轴：X
  - 从动轴：Y

程序代码	
PROG PROGEVENT	
N10 IF(($S_SEARCH_MASLC[Y] <> 0) AND ($AA_MASL_STAT[Y] <> 0))	
N20 MASLOF(Y)	
N30 SUPA Y = $AA_IM[X] - $P_SEARCH_MASLD[Y]	
N40 MASLON(Y)	
N50 ENDIF	
N60 REPOSA	
...	
RET	

- **机床数据**

  为了能自动启动 ASUB，必须设置以下机床数据：
  - NC 专用机床数据：
    - MD11604 $MN_ASUP_START_PRIO_LEVEL = 100
    - MD11450 $MN_SEARCH_RUN_MODE = 'H02'
  - 启动 ASUB 的通道专用，或适用于所有通道：
    - MD20105 $MC_PROG_EVENT_IGN_REFP_LOCK，位 <n> = TRUE
      n：用于全部所需事件控制的程序调用（程序事件）
    - MD20115 $MC_IGNORE_REFP_LOCK_ASUP，位 <n> = TRUE
      n：用于全部**所需用户**中断

注意
系统中断 通过 MD20115 $MC_IGNORE_REFP_LOCK_ASUP，位 8 ~ 31 使能系统中断。 通过位 8 / 中断 9 启动一个包含运行的 ASUB。

联动

SERUPRO 支持跟随运行（TRAILON）功能。

更多信息：

跟随运行（TRAILON，TRAILOF）的更多详细信息请见：

- “轴和主轴”功能手册，“轴耦合”章节
- “NC 编程”手册，“轴耦合”章节

龙门轴

SERUPRO 支持龙门轴功能。

更多信息

有关龙门轴功能的更多信息请见功能手册“轴和主轴”，“龙门轴”章节

切向控制

SERUPRO 支持单轴的切向跟踪。

更多信息

有关切向控制的更多信息请见功能手册“轴和主轴”，“切向控制”章节

加快执行速度

对于跟随轴处于另一个通道的引导轴，用于加快执行速度 (MD22601 (页 178)$MC_SERUPRO_SPEED_FACTOR) 的设置不生效。

3.8.8.7 轴功能

SERUPRO 条件

在轴使能、自控轴运动和跨通道取轴中，必须注意针对 SERUPRO 的特殊条件。

轴使能

若不宜或无法将闭环控制使能发送至机床，且程序测试或 SERUPRO 期间无使能生效，那么轴接口信号 <Axis>.basic.out.progTestEnableCtrl（“程序测试轴/主轴使能”）可用于控制轴使能。

可通过接口信号 PLC → NC
<Axis>.basic.out.progTestEnableCtrl （“程序测试轴/主轴使能”）进行使能。若程序测试或 SERUPRO 中缺少实际的闭环控制使能，则会对轴/主轴产生以下影响：

- 仿真的程序运行需要运行轴/主轴时，会立即显示信息“等待轴使能”或“等待主轴使能”，并停止仿真。
- 若在仿真运行期间重新取消 NC/PLC 接口信号 <Axis>.basic.out.progTestEnableCtrl（程序测试轴/主轴使能），则会触发报警 21612：“通道 %1 轴 %2 NC/PLC 接口信号‘闭环控制使能’在运行期间被取消”。

自控轴运动

自控单轴运动是由 PLC 控制的轴，其也可在 SERUPRO 中进行仿真。为此，SERUPRO 期间 PLC 会像一般运行时一样接收或交还轴的控制权。必要时也可通过 SINU_TraversePosAxis（Basic Program FC18）运行该轴。PLC 在逼近程序段**前**接收轴的控制权，并负责该轴的定位。这适用于所有程序段搜索类型。

更多信息
关于自控单轴进程的详细信息，请参见功能手册之进给轴和主轴；定位轴

跨通道取轴

问题：轴通过程序运行，而改程序在写入了 WAITP(X) 的目标程序段前交还控制权。这样一来 X 不归属于 REPOS，SERUPRO 定位中将不包含该轴。

通过机床数据 MD11470 $MN_REPOS_MODE_MASK 可为 SERUPRO-REPOS 设定以下特性：

中性轴作为“指令轴”在 SERUPRO-REPOS 中运行。该轴的插补无轨迹关联，即便其最近一次是被编程作为轨迹轴。在此情形下速度取决于 MD32060 $MA_POS_AX_VELO。SERUPRO 定位后该轴重新恢复为中性轴。

不允许重新定位的中性轴必须设定轴专用 NC/PLC 接口信号“REPOSDELAY”。这样便可删除 REPOS 运行。

示例：

SERUPRO 后通过同步动作中的工艺循环运行一根轴。指令轴总是在逼近程序段中运行，而从不在目标程序段中运行。所有指令轴都切换至终点时，才可切换至目标程序段。

⚠ 小心
PLC 控制的轴不进行重新定位 目标程序段前通过 RELEASE(X) 使能的轴不进行重新定位。

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<Axis>.basic.out.progTestEnableCtrl	LBP_Chan*.A_PrgtestAxRel	DB31, ...DBX3.7

3.8.8.8 齿轮档切换

步骤

齿轮档切换（GSW）从 NC 请求物理运动，用于切换至新的齿轮档。
SERUPRO 进程中不需要齿轮档切换，并如下执行：

某些齿轮箱只能在 NC 的控制下进行切换，因为轴会出现摆动，或者需要在切换前定位至特定位置。

可通过 MD35035 $MA_SPIND_FUNCTION_MASK 的位 0 至位 2 选择为 DryRun、程序测试和 SERUPRO 抑制齿轮档切换。

之后须在 REPOS 中追加执行 GSW；即使对应轴需要在目标程序段处于“转速控制运行”，追加的 GWS 同样生效。否则，若零件程序中 GWS 和目标程序段之间轴参与了转换或耦合，那么自动 GWS 将会被系统拒绝并触发报警。

说明

有关 DryRun、程序测试和 SERUPRO 时的齿轮档切换的更多信息请见功能手册之*进给轴和主轴*的*主轴*一章。

3.8.8.9 叠加运动

仅可使用 SERUPRO

若使用“叠加运动”，则只可采用程序测试下的程序段搜索（SERUPRO），因为此时叠加运动会在主处理中进行相应插补。 这尤其适用于 $AA_OFF。

速度特性图取代最大轴速度

在程序测试中必须使用速度特性图，其支持在主处理中插补“叠加运动”。 这样一来便无法通过最大轴速度进行插补。

轴速度在“空运行进给”模式下通过
SD42100 $SC_DRY_RUN_FEED 设置。

SERUPRO 进程的速度通过
MD22600 $MC_SERUPRO_SPEED_MODE 选择。

3.8.8.10 NC/PLC 接口信号

REPOS 偏移存在

若 SERUPRO 期间轴存在 REPOS 偏移，则会在 SERUPRO 进程结束时通过轴专用 NC/PLC 接口信号显示：

<Axis>.basic.in.reposOffsetActive == 1（REPOS 偏移存在）

REPOS 偏移的生效范围

SERUPRO ASUB 启动，或通过 NC 启动 继续执行时，REPOS 偏移失效：

<Axis>.basic.in.reposOffsetValid == 1（REPOS 偏移失效）

在 SERUPRO 进程结束和通过 NC 启动 继续执行之间，可在 JOG 运行方式下手动运行轴，或借助 PLC 用户程序通过 FC 18 运行轴。REPOS 偏移完全走完时，接口信号复位。

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<Axis>.basic.in.reposOffsetActive	LBP_Chan*.E_ReposSh	DB31, ...DBX70.0
<Axis>.basic.in.reposOffsetValid	LBP_Chan*.E_ReposShValid	DB31, ...DBX70.1

3.8.8.11 缺省设置的灵活设定

缺省设置 / SERUPRO 缺省设置

通过机床数据 MD20112 $MC_START_MODE_MASK 定义零件程序开始时控制系统在 G 代码（特别是当前层级和可设定的零点偏移）、刀具长度补偿、转换和轴耦合方面的缺省设置。特别是对于 SERUPRO 进程而言，还可通过 MD22620 $MC_ENABLE_START_MODE_MASK_PRT 选择与一般零件程序启动有别的缺省设置。其中新的设置必须保存在以下机床数据中：

MD22620 $MC_START_MODE_MASK_PRT

MD22620 各位的含义与 MD20112 $MC_START_MODE_MASK 相同。

示例：

零件程序启动时保留 SERUPRO 开始时的同步主轴耦合。

```
$MC_START_MODE_MASK = 'H400'                    ; 未配置的同步主轴耦合
                                                ; 被撤销
$MC_START_MODE_MASK_PRT = 'H00'                 ; 保持生效
$MC_ENABLE_START_MODE_MASK_PRT = 'H01'          ; SERUPRO 中会分析
                                                $MC_START_MODE_MASK_PRT，而不
                                                是 $MC_START_MODE_MASK
```

3.8.8.12 压缩功能

在程序段搜索**类型 5** 中，若目标程序段所处的程序部分中激活了压缩器功能（COMP...），那么在重定位至轮廓时会逼近通过**压缩器**计算出的轨迹上的位置。这些位置不必与零件程序中编写的轨迹上的位置精确一致。

若压缩时省去了零件程序中编写的程序段，那么在程序段搜索中将无法将这些程序段作为目标程序段找到。会输出报警 15370“在程序段搜索中未找到搜索目标”。

3.8.9 系统变量

SERUPRO 的相关系统变量概览：

系统变量	含义
$AC_ASUP，位 20	ASUB 激活原因： $AC_ASUB, 位 20 == 1 ⇒ 系统 ASUB 生效，原因：到达 SERUPRO 搜索目标
$AC_SERUPRO	SERUPRO 状态： $AC_SERUPRO == 1 ⇒ SERUPRO 生效
$P_ISTEST	程序测试状态： SERUPRO 生效 ⇒ $P_ISTEST == 1
$P_SEARCHL	最后生效的程序段搜索类型： $P_SEARCHL == 5 从 SERUPRO 开始至复位或程序结束
$AC_REPOS_PATH_MODE	REPOS 模式按照 SERUPRO 用于轮廓重新定位

3.9 异步子程序（ASUB）

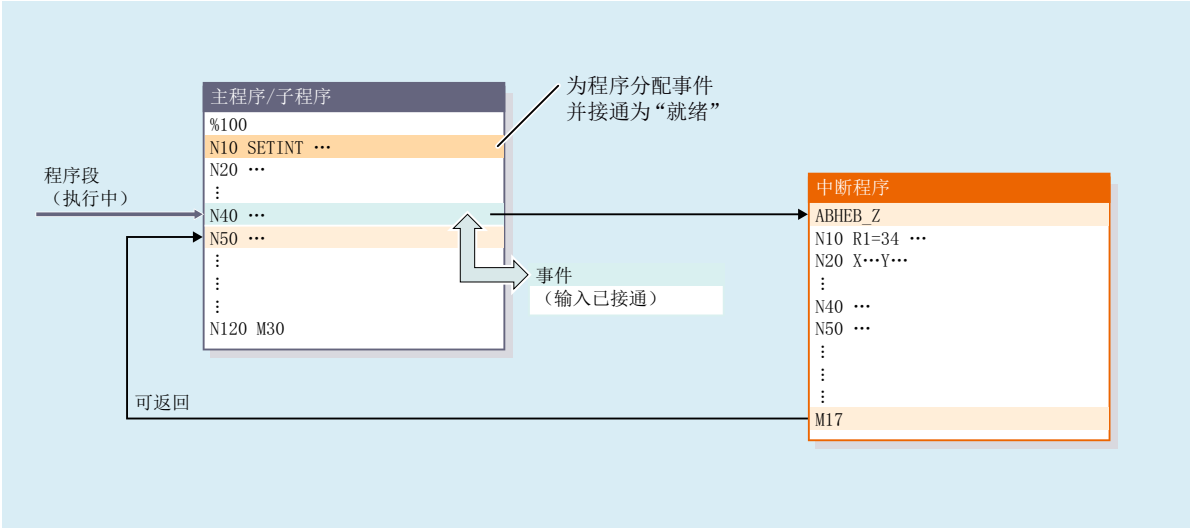
3.9.1 功能

说明

下文所用的术语“异步子程序”、“ASUB”和“中断程序”表示同一种功能。

通用

异步子程序（ASUB）是在 NC 通道中作为对异步事件（中断输入信号、过程或机床状态）的响应启动的 NC 程序。通过激活 ASUB 可以中断一个正在执行的 NC 程序。通过结束 ASUB 可在中断位置上继续执行 NC 程序。



正在通道中执行的 NC 程序在中断前可由 ASUB 进行整体或部分保护。参见章节“编程（SETINT、PRIO） (页 211)”，“灵活编程”。

定义

为了从 NC 程序中生成一个 ASUB（中断程序），必须通过指令 SETINT（参见章节“编程（SETINT、PRIO） (页 211)”）或 PI 服务 “ASUB”（参见功能手册“PLC”的*模块描述*）给 NC 程序分配一个中断信号。

中断信号

- 共有 8 个外设输入可用作中断信号。
- 可通过 PLC 用户程序控制外设输入信号。
- 前四个外设输入为 NCU 模块的 4 个快速输入。可通过数据块 DB10 中的 NC/PLC 接口读取信号状态。此外，通过数据块 DB10 中的 NC/PLC 接口还可禁用输入信号。

更多信息

- 更多信息请参见 NC 数字量和模拟量 I/O (页 859)。
- 通过 PLC 控制快速 NC 输入（中断信号）的更多信息请见 PLC 功能手册。

调用

在程序运行中

在程序运行中，即在 AUTO 或 MDI 模式下，原则上始终可以调用 ASUB。

在程序运行外

在程序运行外，在以下运行方式、机床功能和状态下可以调用 ASUB。

- JOG，JOG-REF
- MDI-Teach In，MDI-Teach In-REF，MDI-Teach In-JOG，MDI-REF，MDI-JOG
- AUTOMATIK，程序状态“已停止”、“就绪”
- 进给轴状态“未回参考点”

如果在 JOG 或 JOG-REF 期间启动 ASUB，系统会终止当前运行。

激活

可通过以下方式激活 ASUB：

- 中断信号的 0/1 脉冲沿，通过快速 NC 输入上的 0/1 脉冲沿触发
- 调用“功能调用 ASUB”（另见功能手册“PLC 和基本程序”，章节“PLC 主程序”）。
- 通过同步动作设置输入，该输入通过短路在中断输入上进行参数设置（参见“示例 (页 215)”）

  更多信息

  功能手册之同步动作分册

显示

激活 ASUB 通过以下 NC/PLC 接口信号显示：

`<Chan>.basic.in.asupActive == 1`（ASUB 生效）

PLC 信号

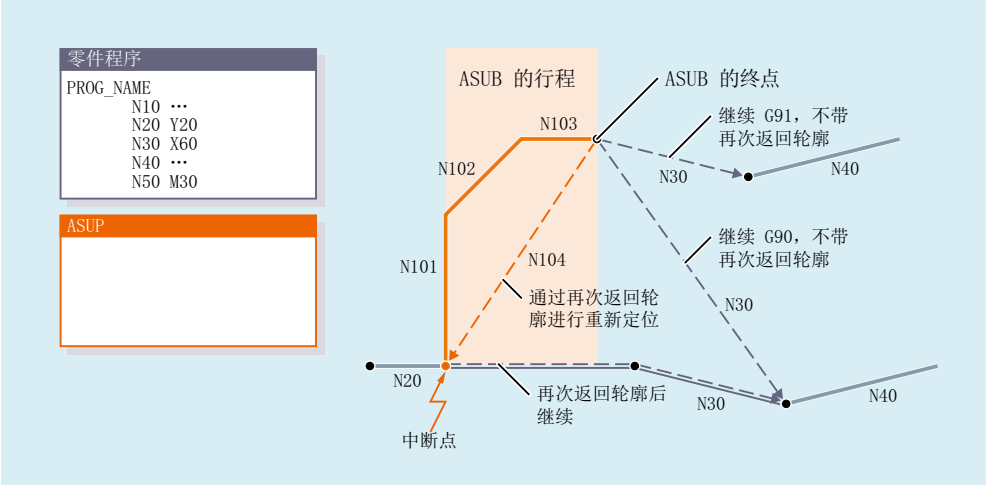
NC → PLC

Basic Program Plus	Basic Program
<Chan>.basic.in.asupActive	LBP_Chan*.E_AnyAsup / DB21, ...DBX378.0

3.9.1.1 程序运行中 ASUB 的执行过程

1. 轴的制动
   激活 ASUB 后，所有机床轴都会沿制动斜坡（MD32300 $MA_MAX_AX_ACCEL）被制动到静止状态，轴位置会被保存。
2. 重新整理
   除了轴的制动外，预解码计算程序段乃至中断程序段都会被反推计算，即确定中断位置无零件程序预解码情形下的值并指定给所有变量、框架和 G 指令。这些值也会保存在缓存中，以便在 ASUB 结束后对其进行访问。
   不可进行重整的特例情形：
   – 螺纹切割程序段内
   – 几何值复杂的情形下（例如样条或半径补偿）

3. 执行 ASUB
   重整结束后，ASUB 自动启动。
   像普通子程序一样执行 ASUB（嵌套深度等）。
4. 结束 ASUB
   在执行完 ASUB 的结束标识（M02、M30、M17）后，默认情形下会向中断程序段后的零件程序段的终点位置运行。
   若需定位回中断点，则必须在 ASUB 结束处编写 REPOS 指令，例如：N104 REPOSL M17



3.9.1.2 带 REPOSA 的 ASUB

在 NC 程序因 NC 停止或报警而停止，并且随后通过 PLC 用户程序借助 FC9 触发包含 REPOSA 的 ASUB 的情况下，通常会出现以下过程：

- ASUB 或其中编写的运行被执行：
  - 程序状态：“停止”
  - <Chan>.basic.in.asupStopStateActive（ASUB 已停止）= 1
- 在重新定位至轮廓（REPOS）前重新停止。

- 为重新定位至轮廓（REPOS），操作员触发 NC 启动：
  - <Chan>.basic.in.asupStopStateActive（ASUB 已停止）= 0
  - 系统执行重新定位运行。
- 重新定位运行结束后，系统会置位 FC9 应答信号"ASUB 完成"，并继续执行中断的 NC 程序。

说明

NC/PLC 接口信号 <Chan>.basic.in.asupStopStateActive（ASUB 已停止）只在以下情形下置位：
程序运行中“中断”通道状态下的中断

说明

对于无 REPOS 的 ASUB，FC9 应答信号"ASUB 完成"和 NC/PLC 接口信号 <Chan>.basic.in.asupStopStateActive（ASUB 已停止）的复位同时出现。

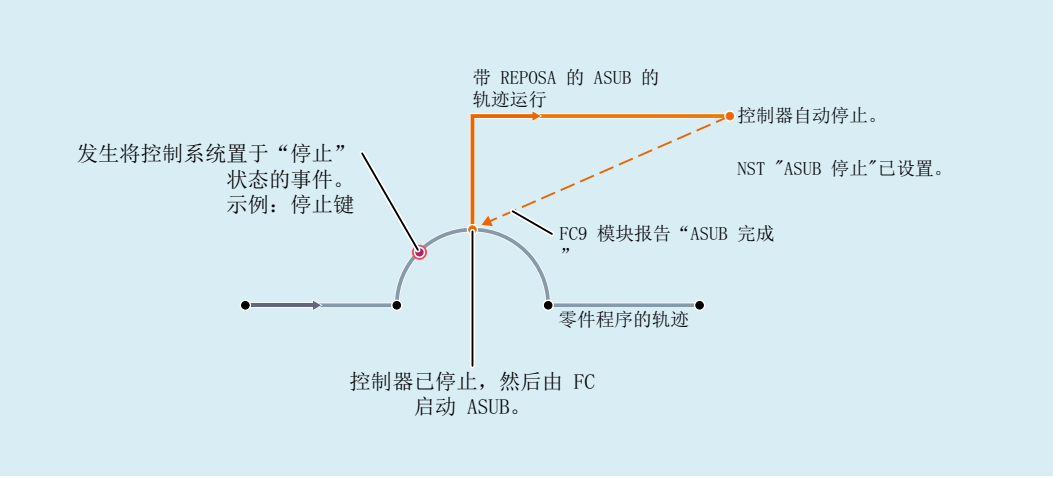


图 3-6　示意性流程图带 REPOSA 的 ASUB

PLC 信号

NC → PLC

Basic Program Plus	Basic Program
<Chan>.basic.in.asupStopStateActive	LBP_Chan*.E_ASUP_Stop / DB21, ...DBX318.0

3.9.1.3 NC 特性

不同状态下的通道、BAG 或 NC 对一个激活的 ASUB 的响应参见下表：

状态	ASUB 启动	控制系统的响应
程序生效	中断，（PLC）	1. 快速退刀或停止轴 2. 在 ASUB 执行期间中断程序 3. ASUB 中编写了 REPOS 时，定位至中断位置 4. 继续执行零件程序
复位	中断，（PLC）	ASUB 如主程序运行。在 ASUB 结束处重新执行复位（无 M30）。控制系统随后的状态取决于以下机床数据： MD20110 $MC_RESET_MODE_MASK MD20112 $MC_START_MODE_MASK **更多信息** 工件相关的实际值系统 (页 445)
程序运行（AUTOMATC 或 MDI） + 通道停止	中断，（PLC）	ASUB 运行。其运行完毕时，系统重新进入停止状态。 ASUB 中编写了 REPOS 时： • ASUB 执行在定位程序段前停止。 • 定位可通过启动键触发。
	启动键	ASUB 运行后，停止的程序继续运行。
手动模式 + 通道停止	中断，（PLC）	控制系统针对响应的通道启用“内部程序执行运行方式”（对外不显示），并激活 ASUB。所选择的运行方式将保留。ASUB 结束（M17）后控制系统重新恢复原来的状态。
JOG 自动示教 自动示教回参考点	中断，（PLC）	执行停止，分析以下机床数据： MD11602 $MN_ASUP_START_MASK MD11604 $MN_ASUP_START_PRIO_LEVEL 必要时于系统内部切换至“内部程序执行运行方式”，激活 ASUB，恢复为 ASUB 启动前的状态。 JOG 模式下，配合 SETINT 定义的 LIFTFAST 不会被激活。
MDI-JOG, MDI-Teach-In, MDI-Teach 参考点	中断，（PLC）	
手动模式 + 通道运行	中断，（PLC）	停止当前生效的运动。删除剩余行程。之后的过程与“手动模式，通道停止”相同。
用户报警 65500 - 65999， 通道处于复位状态	中断，（PLC）	设置了以下机床数据时，存在报警响应“NC 启动禁用”的情形下系统仍会执行用户 ASUB： MD20194 $MC_IGNORE_NONCSTART_ASUP

状态	ASUB 启动	控制系统的响应
执行 INITIAL.INI	**无法使用**	系统会生成“无法请求中断”信号。
程序段搜索		
无法通过 NC 启动消除的报警		
启用数字化		
通道处于故障状态		
其他用户报警 65500 - 65999， 通道处于复位状态， MD20194 未生效	**无法使用**	生成信号“请求已因报警而终止”。

3.9.2 调试：机床数据

3.9.2.1 NC 专用：BAG 专用 NC/PLC 接口信号和运行方式切换

通过机床数据定义 <ModeGroup>.basic 的 BAG 专用 NC/PLC 接口信号的生效方式，以及执行运行方式切换的通道：

MD11600 $MN_BAG_MASK = <值>

值	含义
0	<ModeGroup>.basic 的 BAG 专用 NC/PLC 接口信号**生效**。 在运行方式组的**所有**通道中进行内部运行方式切换。
1	<ModeGroup>.basic 的 BAG 专用 NC/PLC 接口信号**不**生效。 **仅**在 **ASUB** 生效的通道中进行内部运行方式切换。
2	<ModeGroup>.basic 的 BAG 专用 NC/PLC 接口信号**生效**。 **仅**在 **ASUB** 生效的通道中进行内部运行方式切换。
3	<ModeGroup>.basic 的 BAG 专用 NC/PLC 接口信号**不**生效。 **仅**在 **ASUB** 生效的通道中进行内部运行方式切换。

说明

多通道系统

若在多通道系统中可使用“JOG 运行方式下 ASUB 中断期间手动运行”功能（见下文），则必须将 MD11600 $MN_BAG_MASK 设置成值“2”或“3”。

参见

编程（SETINT、PRIO）(页 211)

3.9.2.2 NC 专用：ASUB 启动使能

通过机床数据定义在 ASUB 启动时忽略哪些停止原因：

MD11602 $MN_ASUP_START_MASK，<位> = <值>

位	值	含义
0	0	在因停止键、M0 或 M01 停止的情况下，**不**启动 ASUB。
	1	尽管因停止键、M0 或 M01 停止，仍**启动** ASUB。
2	0	在启动 ASUB 时，在通道中存在读取禁止的情况下，以下通道专用机床数据中的设置生效： • MD20107 $MC_PROG_EVENT_IGN_INHIBIT • MD20116 $MC_IGNORE_INHIBIT_ASUP
	1	即便在相应通道中启用读取禁止的情况下，仍在 NC 的所有通道中启动用户或系统 ASUB。 以下通道专用机床数据中的设置**不**生效： • MD20107 $MC_PROG_EVENT_IGN_INHIBIT • MD20116 $MC_IGNORE_INHIBIT_ASUP **注意** 尽管存在读取禁止，启动使能仍在 NC 的所有通道中生效。这既适用于用户 ASUB，也适用于系统 ASUB。因此**强烈建议**使用**通道专用**使能来替代 NC 专用使能。
3	0	若在 JOG 运行模式中启动 ASUB 并通过 NC 停止中断，在此状态下**不**可手动运行轴。
	1	若在 JOG 运行模式中启动 ASUB 并通过 NC 停止中断，在此状态下可以手动运行轴。 可通过 NC 启动继续执行 ASUB。其中自动进行 REPOS。 **提示** 若在控制系统中参数设置了超过一个通道，必须额外设置以下机床数据： MD11600 $MN_BAG_MASK，位 1 = 1

手动启动使能

若在参数设置了启动使能的情况下 ASUB 未**自动**启动，可通过 PLC 用户程序借助 NC/PLC 接口信号（<Chan>.basic.out.ncStart），或者通过手动操作 NC 启动来启动 ASUB。

说明

用于“从轮廓快速退刀”（LIFTFAST）的 ASUB 在任何情形下都会启动。

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.ncStart	LBP_Chan*.A_NCStart	DB21, ...DBX7.1

3.9.2.3 NC 专用：参数设置的启动使能的生效范围

通过以下机床数据设置：从最高优先级起至哪个 ASUB 优先级 (页 211)为止，MD11602 $MN_ASUP_START_MASK 中的设置生效。

MD11604 $MN_ASUP_START_PRIO_LEVEL = <ASUB 优先级>

示例

MD11604 $MN_ASUP_START_PRIO_LEVEL = 5

MD11602 $MN_ASUP_START_MASK 中的设置对优先级为 1 → 5 的 ASUB 生效。

3.9.2.4 通道专用：在轴未回参考点的情况下仍使能启动

通过以下机床数据设置：在哪些中断中，在参数设置了“不带参考点的 NC 启动禁用”（MD20700 $MC_REFP_NC_START_LOCK）的情况下仍启动对应的 ASUB：

MD20115 $MC_IGNORE_REFP_LOCK_ASUP，位 (1 - <中断>) = TRUE

注意
系统中断 通过 MD20115 $MC_IGNORE_REFP_LOCK_ASUP，位 8 - 31 使能与系统中断对应的 ASUB。 通过位 8 / 中断 9 启动一个包含运行的 ASUB。 **NC 专用 ASUB 启动使能** 若 MD11602 $MN_ASUP_START_MASK 位 2 == TRUE，那么尽管参数设置了通道专用功能“不带参考点的 NC 启动禁用”（MD20700 $MC_REFP_NC_START_LOCK），仍对 NC 的所有通道进行 ASUB 使能。因此**强烈建议**使用**通道专用**使能来替代 NC 专用使能。

3.9.2.5 通道专用：在读取禁止的情况下仍使能启动

通过以下机床数据设置：在哪些中断中，在通道中启用读取禁止（<Chan>.basic.out.disableReadIn）的情况下仍启动对应的 ASUB：

MD20116 $MC_IGNORE_INHIBIT_ASUP，位 (1 - <中断>) = TRUE

注意
系统中断 通过 MD20116 $MC_IGNORE_INHIBIT_ASUP，位 8 - 31 使能与系统中断对应的 ASUB。 通过位 8 / 中断 9 启动一个包含运行的 ASUB。 **NC 专用 ASUB 启动使能** 若 MD11602 $MN_ASUP_START_MASK 位 2 == TRUE，则在 NC 的所有通道中忽略 MD20116 $MC_IGNORE_INHIBIT_ASUP 中的通道专用设置。因此**强烈建议**使用**通道专用**使能来替代 NC 专用使能。

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.disableReadIn	LBP_Chan*.A_RIdisable	DB21, ...DBX6.1

3.9.2.6 通道专用：在采用单程序段的情况下仍连续执行

通过以下机床数据设置：在哪些中断中，在通道中启用单程序段处理（<Chan>.basic.out.singleBlock）的情况下仍**连续**地，即以无逐段中断的方式执行对应的 ASUB：

MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP，位 (1 - <中断>) = TRUE

边界条件

MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP 中的设置仅在单程序段 SBL1（主处理单程序段）中生效。

注意
系统中断 通过 MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP，位 8 - 31 使能与系统中断对应的 ASUB。 通过位 8 / 中断 9 启动一个包含运行的 ASUB。 **NC 专用 ASUB 启动使能** 若 MD10702 $MN_IGNORE_SINGLEBLOCK_MASK 位 1 == TRUE，则在 NC 的所有通道中忽略 MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP 中的通道专用设置。因此**强烈建议**使用**通道专用**使能来替代 NC 专用使能。

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.singleBlock	LBP_Chan*.A_SBL	DB21, ...DBX0.4

3.9.2.7 通道专用：显示更新

通过以下机床数据设置：在执行 ASUB 期间不更新显示，从而防止在执行非常短的 ASUB 时，操作界面上的程序状态和通道状态的显示闪动：

MD20191 $MC_IGN_PROG_STATE_ASUP，位 (1 - <中断>) = TRUE

说明

NC/PLC 接口信号

在抑制显示的情形下执行 ASUB 时，以下 NC/PLC 接口信号置位：

<Chan>.basic.in.asupSilentActive = 1（“静止”ASUB 生效）

系统变量和 NC/PLC 接口信号

在执行 ASUB 期间抑制显示时，用于程序状态和通道状态的系统变量和 NC/PLC 接口信号不会受影响：

- $AC_STAT（通道状态）
- $AC_PROG（程序状态）
- <Chan>.basic.in.stateActive（通道状态）
- <Chan>.basic.in.stateInterrupted
- <Chan>.basic.in.stateReset
- <Chan>.basic.in.progStateRunning（程序状态）
- <Chan>.basic.in.progStateWaiting
- <Chan>.basic.in.progStateStopped
- <Chan>.basic.in.progStateInterrupted
- <Chan>.basic.in.progStateAborted

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.progStateRunning	LBP_Chan*.E_ProgRunning	DB21, ...DBX35.0
<Chan>.basic.in.progStateWaiting	LBP_Chan*.E_ProgWait	DB21, ...DBX35.1
<Chan>.basic.in.progStateStopped	LBP_Chan*.E_ProgStop	DB21, ...DBX35.2
<Chan>.basic.in.progStateInterrupted	LBP_Chan*.E_ProgInterrupt	DB21, ...DBX35.3
<Chan>.basic.in.progStateAborted	LBP_Chan*.E_ProgAborted	DB21, ...DBX35.4
<Chan>.basic.in.stateActive	LBP_Chan*.E_ChanActive	DB21, ...DBX35.5
<Chan>.basic.in.stateInterrupted	LBP_Chan*.E_ChanInterrupt	DB21, ...DBX35.6

Basic Program Plus	Basic Program	
<Chan>.basic.in.stateReset	LBP_Chan*.E_ChanReset	DB21, ...DBX35.7
<Chan>.basic.in.asupSilentActive	LBP_Chan*.E_SilentAsup	DB21, ...DBX378.1

3.9.3 编程：系统变量

3.9.3.1 REPOS 可能性（$P_REPINF）

对于 ASUB，可能存在无法退回重新逼近轮廓（REPOS）的情形。

通过以下系统变量，能够在 ASUB 中读取是否能够进行 REPOS：

<值> = $P_REPINF

值	含义
0	不可通过 REPOS 重新定位，因为： • 未在 ASUB 中调用 • ASUB 从复位状态开始运行 • ASUB 从 JOG 模式开始运行
1	可在 ASUB 中通过 REPOS 重新定位

3.9.3.2 激活事件（$AC_ASUP）

通过系统变量 $AC_ASUP 能够读取有关导致 ASUB 激活的事件的下列信息：

- 为何激活 ASUB，例如位 0：用户中断“采用 Blsync 的 ASUB”
- 如何激活 ASUB，例如位 0：NC/PLC 接口信号，数字量模拟量接口
- 可采用哪些继续执行的方式，例如位 0：可自由选择 REORG 或 RET

3.9.4 编程（SETINT、PRIO）

指定对应关系：中断信号与 NC 程序

通过指令 SETINT 指定 NC 程序与中断信号的对应关系。进而确定 NC 程序与 ASUB 的对应关系。

句法

SETINT(<n>) <NC 程序>

含义

SETINT：	NC 程序与中断信号的对应关系	
<n>：	中断信号编号	
	值域：	0, 1, 2, ... 8
<NC 程序：	程序名	

示例

程序代码	注释
N20 SETINT(3) ABHEBEN_Z	; IF 输入 3 == 1 ; 则启动 ASUB“ABHEBEN_Z”

此外还可配合 SETINT 编写以下指令：

- LIFTFAST
  出现中断信号时，在 ASUB 启动前执行“从轮廓快速退刀”。快速退刀的运动方向通过程序指令 ALF 定义。
- BLSYNC
  出现中断信号时仍继续执行运行中的程序段，之后再启动 ASUB。

说明

在下列条件下，指定的“中断信号 ↔ 零件程序”对应关系会被清除：
- 通道处于复位状态
- 零件程序中编写 CLRINT 指令

优先级

如果在一个 NC 程序中通过 SETINT 激活了多个中断，则须为对应的 NC 程序或 ASUB 指定不同的优先级。

句法
PRIO=<值>

含义

PRIO：	用于确定中断优先级的关键字
<值>：	优先级：1, 2, 3 ... 128。1 对应最高优先级。

示例

程序代码	注释
N20 SETINT(3) PRIO=2 ABHEBEN_Z	; IF 输入 3 == 1
	; 则启动 ASUB"ABHEBEN_Z"
N30 SETINT(2) PRIO=3 ABHEBEN_X	; IF 输入 2 == 1
	; 则启动 ASUB"ABHEBEN_X"

如果输入端 2 和 3 同时保留，则 ASUB 会根据级别数的顺序进行处理：

1. "ABHEBEN_Z"
2. "ABHEBEN_X"。

其他中断专用指令

指令	含义
SAVE	如果在 ASUB 中使用指令 SAVE，那么 ASUB 结束时，中断之前在中断的 NC 程序中生效的 G 指令、框架和转换重新生效。
DISABLE ENABLE	通过指令对 DISABLE ENALBE 可保护程序段，防止其被 ASUB 中断。 通过 SETINT 指定的中断信号与 NC 程序之间的对应关系仍然保留。 在 ENALBE 之后通过中断信号的下一个 0/1 沿启动 ASUB。
CLRINT(<n>)	删除中断信号与通过 SETINT 指定的 NC 程序的对应关系。

更多信息

NC 编程手册

参见

编程 (页 217)

3.9.5 前提条件

跨运行方式的 ASUB 启动

需要检查的设置

- MD11600 $MN_BAG_MASK
- MD11604 $MN_ASUP_START_PRIO_LEVEL
- 中断指定的优先级

推荐设置

NC 专用机床数据：

- MD11600 $MN_BAG_MASK = 'H3'

  说明

  在采用此设置时需要注意，<ModeGroup>.basic 的 BAG 专用 NC/PLC 接口信号不再对执行 ASUB 的通道生效。若不期望此特性，则可使用设置 MD11600 $MN_BAG_MASK = 'H2' 作为替代（参见“调试：机床数据 (页 205)”）。

  MD11602 $MN_ASUP_START_MASK = 'H5'
- MD11604 $MN_ASUP_START_PRIO_LEVEL = 7

通道专用机床数据，用于启动 ASUB 的通道，或适用于所有通道：

- MD20105 $MC_PROG_EVENT_IGN_REFP_LOCK，位 <n> = TRUE
  <n>：用于全部所需事件控制的程序调用（程序事件）
- MD20115 $MC_IGNORE_REFP_LOCK_ASUP，位 <n> = TRUE
  <n>：用于全部**所需用户**中断

注意
系统中断 通过 MD20115 $MC_IGNORE_REFP_LOCK_ASUP，位 8 - 31 使能系统中断。 通过位 8 / 中断 9 启动一个包含运行的 ASUB。

3.9.6 示例

通过中断从同步动作中激活 ASUB

1. 设置两个生效的数字量输入/输出字节：
   – MD10350 $MN_FASTIO_DIG_NUM_INPUTS = 2
   – MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS = 2
2. 通过逻辑或运算将短路设为从输出 9 到输入 9：
   – 输入 1，输入字节 2 =（输出 1，输出字节 2）**或** 硬件输入信号 1，输入字节 2）：
   MD10361 $MN_FASTIO_DIG_SHORT_CIRCUIT[0] = 'H0102B102'
3. 将硬件输入字节指定给中断编程 SETINT：
   – 输入字节 2：
   MD21210 $MC_SETINT_ASSIGN_FASTIN = 2
4. 将输入定义为 ASUB 触发器：
   – 第二个输入字节中的输入 1，即绝对输入 9，启动程序 SYNCASUP
   ```
   SETINT(1) PRIO=1 SYNCASUP
   ```
5. 定义用于设置输出的同步动作：
   – 如果标准轨迹参数 >= 0.5，带 ID 1 的同步动作会将输出 9 始终设为 1。
   ```
   IDS=1 EVERY $$AC_PATHN >= 0.5 DO $A_OUT[9]=1
   ```
   从输出 9 到输入 9 的短路会触发中断 1 且 NC 程序 “SYNCASUP” 会作为 ASUB 启动。

3.10 用于 RET 和 REPOS 的用户专用 ASUB

3.10.1 功能

功能

控制系统软件包含一个用于实现 NC 程序结束 (RET) 和重新逼近轮廓 (REPOS) 功能的西门子专用 ASUB。机床制造商可用一个用户专用 ASUB 替换系统 ASUB。

⚠ 危险
编程错误
如果该 ASUB 导致编程错误，责任由机床制造商自行承担。

3.10.2 参数设置

安装

ASUB 名称

必须为用户专用 ASUB 给定以下名称：

- **_N_ASUP_SPF**

ASUB 目录

用户专用 ASUB "_N_ASUP_SPF" 必须保存在以下其中一个目录中：

- **_N_CMA_DIR**（制造商目录）
- **_N_CUS_DIR**（用户目录）

激活和查找路径

在以下机床数据中通过位 0 和 1 设置激活哪个事件中的用户专用 ASUB "_N_ASUP_SPF"。

通过位 2 设置在激活用户专用 ASUB "_N_ASUP_SPF" 时从哪个位置开始查找。

MD11610 $MN_ASUP_EDITABLE，位 0、1、2 = <值>

位	值	含义
0 和 1	0	在程序结束 (**RET**) 或重新逼近轮廓 (**REPOS**) 时不会激活用户专用 ASUB。
	1	执行 **RET** 时会激活**用户专用** ASUB。 执行 **REPOS** 时会激活**系统专用** ASUB。
	2	执行 **RET** 时会激活**系统专用** ASUB。 执行 **REPOS** 时会激活**用户专用** ASUB。
	3	执行 **RET** 和 **REPOS** 时会激活**用户专用** ASUB。
2	0	首先在用户目录 **_N_CUS_DIR** 中搜索用户专用 ASUB。
	1	首先在制造商目录 **_N_CMA_DIR** 中搜索用户专用 ASUB。

定义保护级

需要为 RET 和/或 REPOS 启用用户专用 ASUB 时（MD11610 $MN_ASUP_EDITABLE ≠ 0），可为用户专用程序 "_N_ASUP_SPF" 定义一个保护级。保护级的取值范围为 0 - 7。此时通过以下机床数据设置：

MD11612 $MN_ASUP_EDIT_PROTECTION_LEVEL = <用户专用 ASUB 的保护级>

保护等级的详细信息请见：
更多信息
调试手册之保护级方案

启用单程序段执行时的特性

通过设置以下机床数据，可在启用单程序段执行的情形下实现系统专用 ASUB 和用户专用 ASUB "_N_ASUP_SPF" 的无中断执行：

MD10702 $MN_IGNORE_SINGLEBLOCK_MASK，位 0 = <值>

位	值	含义
0	0	单程序段运行激活时，系统会在**每个 ASUB 程序段中停止**。
	1	单程序段运行激活时，系统会**无中断地执行 ASUB**。

3.10.3 编程

确定 ASUB 激活的原因

引起 ASUB 激活的原因可通过系统变量 $AC_ASUP 以位编码读取。

继续执行

使用系统 ASUB 时，在 ASUB 内执行动作后的继续执行特性为固定设置：

- 系统 ASUB1 → 通过 RET（子程序返回）继续执行
- 系统 ASUB2 → 通过 REPOS（重新定位）继续执行

在系统变量的说明中，在“通过...继续执行”下注明了针对每种原因的系统 ASUB 相关特性。

说明

使用用户专用 ASUB 时的继续执行

使用用户专用 ASUB 时，建议保留对应的系统 ASUB 继续执行特性。

原因：运行方式切换（$AC_ASUP，位 9 == 1）

运行方式组切换时，继续执行特性取决于以下机床数据：

MD20114 $MC_MODESWITCH_MASK（通过运行方式切换中断 MDI）

- 位 0 == 0：系统 ASUB1 → 通过 RET 继续执行
- 位 0 == 1：系统 ASUB2 → 通过 REPOS 继续执行

更多信息

对系统变量的详细说明请见系统变量参数手册

3.11 在存在用户报警的情形下启动 ASUB

3.11.1 功能

说明

ASUB 可在各种情形下由用户、系统或通过事件控制来触发。不过在某些情形下，报警响应为“NC 启动禁用”的用户报警会阻止 ASUB 启动。存在报警响应时，系统会通过报警 16906 应答复位触发的用户 ASUB。

通过设置特定的通道专用机床数据，在报警响应为“NC 启动禁用”的用户报警生效的情形下同样可启动和执行 ASUB。此时系统会为 ASUB 启动抑制该报警响应，从而可执行程序。

说明

报警响应为“NC 启动禁用”的 NC 报警则不会受此抑制影响。复位触发的 ASUB 仍无法启动，系统会通过报警 16906 将其拒绝。

可抑制的用户报警

可抑制的用户报警的序号带划分如下：

序号范围	影响	删除
65000 - 65499	显示，NC 启动禁用	复位
65500 - 65999	显示，NC 启动禁用（设置 MD20194 时不适用于 ASUP）	复位
66000 - 66999	显示，NC 启动禁用，执行预编码程序段后运动停止	复位
67000 - 67999	显示	Cancel
68000 - 68999	显示，NC 启动禁用，即刻插补停止	复位
69000 - 69999	显示，NC 启动禁用，在下一个程序段结束处停止	复位

说明

对于以下报警序号带，尽管可抑制报警响应"NC 启动禁用"，但相应报警的其他响应会构成停止条件。这些停止条件无法通过本章节中描述的功能抑制：

- 66000 – 66999
- 68000 – 68999
- 69000 – 69999

3.11.2 激活

设置

可通过以下通道专用机床数据独立设置每个 ASUB 通道：

MD20194 $MC_IGNORE_NONCSTART_ASUP（针对特定用户报警在报警应答"NC 启动禁用"存在的情形下支持 ASUB 启动）

对此机床数据设置的修改需通过零件程序指令 NEWCONF 或操作界面上的软键激活。

过程

ASUB 启动的一般过程如下：

- 相应设置机床数据 MD20194 $MC_IGNORE_NONCSTART_ASUP，并通过 NEWCONF 激活。
- 启动零件程序。
  出现可抑制序号带内的用户报警，例如报警 65500。同步动作或零件程序指令均可能触发该报警。
- 尽管存在报警，系统仍执行零件程序，直至到达程序结束 M02、M30 或 M17。
  通道状态"复位"生效。
- 系统执行复位触发的用户 ASUB。

说明

存在报警响应"NC 启动禁用"时，也可从运行中的或停止的零件程序启动 ASUB。这对响应为"NC 启动禁用"的用户报警和 NC 报警均适用。

说明

若执行用户 ASUB 期间零件程序停止，则无法再通过 NC 启动键继续执行 ASUB，例如 ASUB 中写入了 M0 或启用用户停止时。系统会通过报警 16906 拒绝报警响应"NC 启动禁用"。之前生成的用户报警仍可通过复位应答。

3.11.3 示例

3.11.3.1 通过复位触发的用户 ASUB - 示例 1

在此应用中 MD20194 不置位。

主程序

程序代码

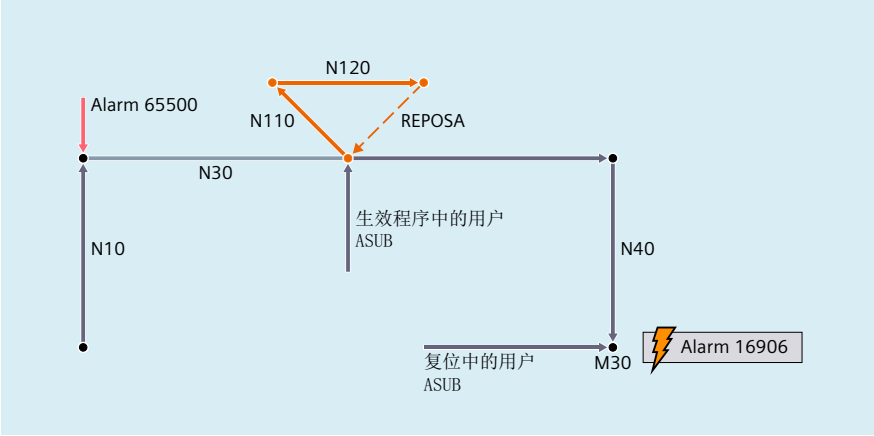
```
N10 G90 G0 Z10
N20 SETAL(65500)
N30 X100
N40 Z0
N50 M30
```

用户 ASUB

程序代码

```
N110 G91 G0 X-10 Z5
N120 X20
N130 REPOSA
```

过程



执行程序段 N10。 出现报警 65500，其响应为“显示”和“NC 启动禁用”。 接下来零件程序不停止。 切换至程序段 N30 并运行。 若在程序段中央启用用户 ASUB，尽管存在报警响应“NC 启动禁用”，系统仍会予以执行。 REPOSA 在程序中断处重新开始，并运行零件程序直至程序结束 M30。 若此时启用用户 ASUB，则会被系统拒绝，并触发报警 16906。 NC 处于复位状态。

3.11.3.2 通过复位触发的用户 ASUB - 示例 2

此应用为一般情形，MD20194 置位。

主程序

程序代码

```
N4 $MC_IGNORE_NONCSTART_ASUP=1
N6 NEWCONF
N10 G90 G0 Z10
N20 SETAL(65500)
N30 X100
N40 Z0
N50 M30
```

用户 ASUB

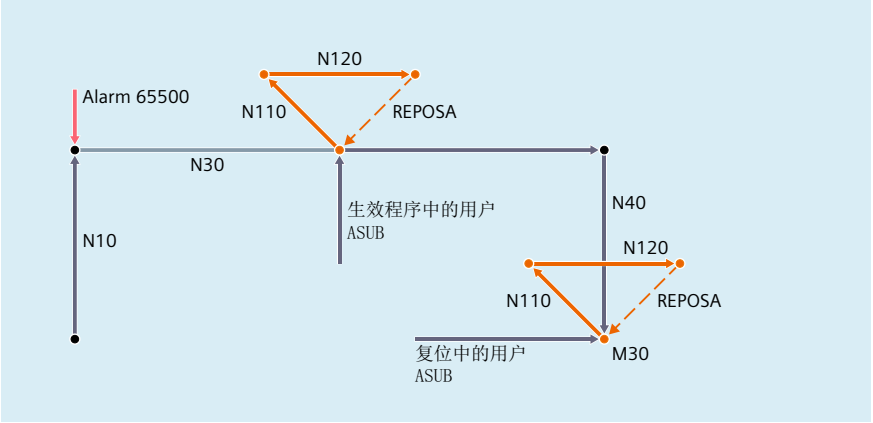
程序代码

```
N110 G91 G0 X-10 Z5
N120 X20
```

程序代码

```
N130 REPOSA
```

过程



为 ASUB 通道 1 设置机床数据 MD20194 $MC_IGNORE_NONCSTART_ASUP，并通过 NEWCONF 激活。执行程序段 N10。出现报警 65500，其响应为“显示”和“NC 启动禁用”。接下来零件程序不停止。切换至程序段 N30 并运行。若在程序段中央启用用户 ASUB，尽管存在报警响应“NC 启动禁用”，系统仍会予以执行。REPOSA 在程序中断处重新开始，并运行零件程序直至程序结束 M30。若此时启用用户 ASUB，系统会基于设置的机床数据 MD20194 重新执行。

3.11.3.3 写入 M0 的用户 ASUB

在此应用中 MD20194 置位。

主程序

程序代码

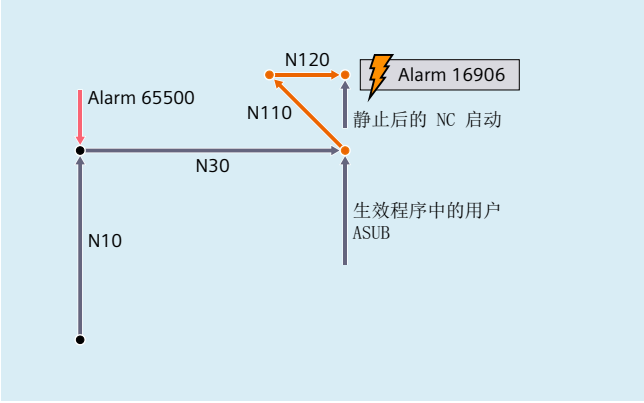
```
N4 $MC_IGNORE_NONCSTART_ASUP=1
N6 NEWCONF
N10 G90 G0 Z10
N20 SETAL(65500)
N30 X100
N40 Z0
N50 M30
```

用户 ASUB

程序代码

```
N110 G91 G0 X-10 Z5
N120 X10
N122 M0
N124 X10
N130 REPOSA
```

过程



为 ASUB 通道 1 设置机床数据 MD20194 $MC_IGNORE_NONCSTART_ASUP，并通过 NEWCONF 激活。执行程序段 N10。出现报警 65500，其响应为“显示”和“NC 启动禁用”。接下来零件程序不停止。切换至程序段 N30 并运行。若在程序段中央启用用户 ASUB，尽管存在报警响应“NC 启动禁用”，系统仍会予以执行。在程序段 N122 处，M0 使零件程序停止。无法再通过按下 NC 启动键继续执行 ASUB。系统会通过报警 16906 拒绝报警响应“NC 启动禁用”。之前生成的报警 65500 仍可通过复位应答。

3.11.3.4 带停止的用户 ASUB

在此应用中 MD20194 置位。

主程序

程序代码

```
N4 $MC_IGNORE_NONCSTART_ASUP=1
N6 NEWCONF
N10 G90 G0 Z10
N20 SETAL(65500)
```

程序代码

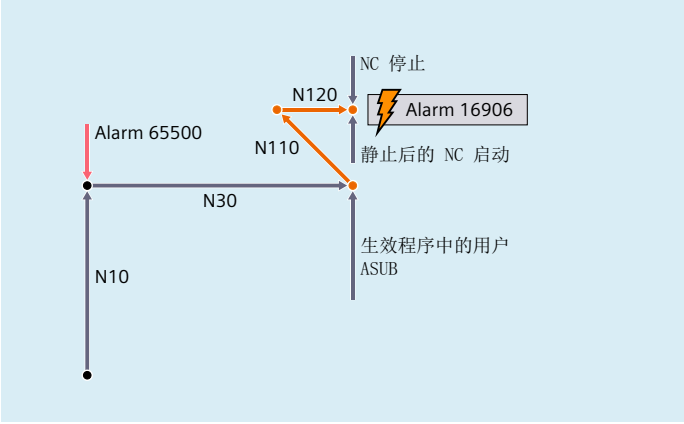
```
N30 X100
N40 Z0
N50 M30
```

用户 ASUB

程序代码

```
N110 G91 G0 X-10 Z5
N120 X20
N130 REPOSA
```

过程



为 ASUB 通道 1 设置机床数据 MD20194 $MC_IGNORE_NONCSTART_ASUP，并通过 NEWCONF 激活。执行程序段 N10。出现报警 65500，其响应为“显示”和“NC 启动禁用”。接下来零件程序不停止。切换至程序段 N30 并运行。若在程序段中央启用用户 ASUB，尽管存在报警响应“NC 启动禁用”，系统仍会予以执行。若在 N120 处在程序段中央按下 NC 停止键，ASUB 将停止。无法再通过按下 NC 启动键继续执行 ASUB。系统会通过报警 16906 拒绝报警响应“NC 启动禁用”。之前生成的报警 65500 仍可通过复位应答。

3.11.3.5 从停止状态触发的用户 ASUB

在此应用中 MD20194 可置位或不置位。

主程序

程序代码

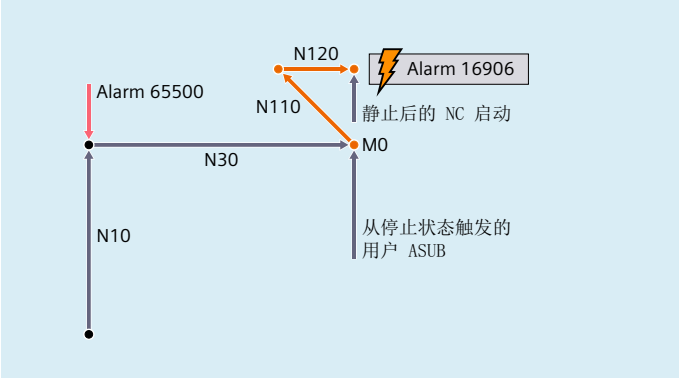
```
N10 G90 G0 Z10
N20 SETAL(65500)
N30 X50
N35 M0
N38 X100
N40 Z0
N50 M30
```

用户 ASUB

程序代码

```
N110 G91 G0 X-10 Z5
N120 X20
N130 REPOSA
```

过程



执行程序段 N10。 出现报警 65500，其响应为“显示”和“NC 启动禁用”。 接下来零件程序不停止。 切换至程序段 N30 并运行。 在程序段 N35 处，M0 使零件程序停止。 NC 处于停止状态。 若此时启用用户 ASUB，尽管 NC 处于停止状态且存在报警响应“NC 启动禁用”，系统仍会予以执行。 机床数据 MD20194 $MC_IGNORE_NONCSTART_ASUP 是否置位对此过程无影响。 程序段 N120 后，ASUB 在 REPOSA 前停止。 下一次 NC 启动时才可进行重新定位。 但是报警响应“NC 启动禁用”会使 NC 启动被拒绝，并触发报警 16906。 之前生成的报警 65500 仍可通过复位应答。

3.12 外部执行

功能

借助“外部执行”功能，可通过一个外部存储器执行由于其大小无法保存在 NC 存储器中的程序。

说明

受保护的循环（_CPF 文件） **无法**通过外部程序存储器执行。

外部程序存储器

外部程序储存器可以位于下列数据载体上：

- 本地驱动器
- 网络驱动器
- USB 驱动器

说明

执行 USB 驱动器上的外部程序仅可使用操作面板前端或 TCU 上的 USB 接口。

注意
USB 闪存可损坏刀具/工件 建议在执行外部子程序时不要使用 USB 闪存驱动器。如在执行零件程序的过程中由于接触不良、脱落以及因碰撞或误拔出而中断与 USB 闪存驱动器的通讯，会导致加工立即停止。这可能会损坏刀具或/和工件。

应用

- **直接执行外部程序**
  原则上可通过操作界面的目录结构在 HMI 模式“外部执行”下访问的所有程序均可选择和执行。
- **从零件程序执行外部子程序**
  “外部”子程序通过零件程序指令 EXTCALL 及指定的调用路径（可选）和子程序名称来调用（→ 参见“执行外部子程序（EXTCALL） (页 228)”）。

参数设置

为了在“外部执行”模式下执行程序，动态 NC 存储器中须预留一块缓存（FIFO 缓存）。

FIFO 缓存的大小

FIFO 缓存的大小通过以下机床数据设置：

MD18360 $MN_MM_EXT_PROG_BUFFER_SIZE（用于外部执行的 FIFO 缓存大小）

说明

带跳转语句的程序

对于含有跳转指令（GOTOF，GOTOB，CASE，FOR，LOOP，WHILE，REPEAT，IF，ELSE，ENDIF 等）的外部程序，在回装存储器中应存在跳转目标。

该条件主要会在跳转目标为程序开始处的跳转指令（GOTOS）中出现问题，因为程序通常都很巨大，无法完整存储在回装存储器中。首次回装后，程序开头会从回装存储器中删除。如果在后续的程序执行过程中执行回到程序开始处的跳转指令，该功能此时便无法找到跳转目标。这会导致程序中断并输出报警 14000。

如要在执行外部程序时正常使用所编写的跳转指令，建议使用“从外部存储器执行程序（EES）” (页 231)替代功能“从外部执行”。

说明

ShopMill/ShopTurn 程序

由于文件末尾附带了轮廓描述，ShopMill/ShopTurn 程序必须完整地保存在回装存储器中。

FIFO 缓存的数量

“外部执行”模式下同时执行的所有程序须分别配备一个 FIFO 缓存。

FIFO 缓存的数量通过以下机床数据设置：

MD18362 $MN_MM_EXT_PROG_NUM（可同时外部执行的程序级的数量）

复位、上电时的特性

复位和上电会使外部程序调用终止，并清除对应的 FIFO 缓存。

通过外部程序存储器选择主程序的情形下，若上电时该程序存储器继续可用，且 MD9106 $MM_SERVER_EXTCALL_PROGRAMS 中激活了通过 EXTCALL 调用加工，则系统会自动重新选择该程序。

3.13 执行外部子程序（EXTCALL）

功能

在加工复杂的工件时，某些加工步骤的程序串可能会由于存储空间需求的问题无法直接保存在 NC 存储器中。

在此情形下，用户可通过零件程序指令 EXTCALL 在“外部执行”模式下从外部存储器将程序串作为子程序调用并执行。

前提条件

执行外部子程序须满足以下前提条件：

- 子程序必须可通过操作界面的目录结构访问。
- 针对每个子程序，动态 NC 存储器中均须预留一块缓存（FIFO 缓存）。

说明

子程序，带跳转语句

在执行带有跳转指令（GOTOF，GOTOB，CASE，FOR，LOOP，WHILE，REPEAT，IF，ELSE，ENDIF 等）的外部子程序时，跳转目标必须位于回装存储器中。

该条件主要会在跳转目标为程序开始处的跳转指令（GOTOS）中出现问题，因为程序通常都很巨大，无法完整存储在回装存储器中。首次回装后，程序开头会从回装存储器中删除。如果在后续的程序执行过程中执行回到程序开始处的跳转指令，该功能此时便无法找到跳转目标。这会导致程序中断并输出报警 14000。

如要在执行外部子程序时正常使用所编写的跳转指令，建议使用“从外部存储器执行程序（EES）”替代选项“执行外部子程序（EXTCALL）”(页 231)。

回装存储器的大小

可以通过如下指令设置加载存储器的大小：

MD18360 MM_EXT_PROG_BUFFER_SIZE

ShopMill/ShopTurn 程序

由于文件末尾附带了轮廓描述，ShopMill/ShopTurn 程序必须完整地保存在回装存储器中。

参数设置

可通过以下设定数据预设到外部子程序目录的路径：

SD42700 $SC_EXT_PROG_PATH（外部子程序调用 EXTCALL 的程序路径）

此路径和编程时指定的子程序路径或名称共同构成待调用程序的整体路径。

说明

如果程序路径只通过编程给出，则 SD42700 必须空着！

编程

通过零件程序指令 EXTCALL 调用一个外部子程序。

语法： EXTCALL("<Pfad/><Programmname>")

参数：

<路径/>： 绝对或相对路径说明（**可选**）
类型： STRING

<程序名称>: 设定程序名称时不添加"_N_"前缀。
可使用字符"_"或"."将后缀名（"MPF"、"SPF"）添加在程序名上（**可选**）。
类型： STRING

说明

路径说明：缩写

在指定路径时可使用以下缩写：

- **LOCAL_DRIVE:** 本地驱动
- **CF_CARD:**用于存储卡
- **USB:** 前端 USB 接口

CF_CARD: 和 **LOCAL_DRIVE:**可以互换使用。

采用绝对路径设定的 EXTCALL 调用

如果在给定的路径下存在子程序，则在 EXTCALL 调用后执行子程序。如果不存在该子程序，则中断程序执行。

采用相对路径设定/无路径设定的 EXTCALL 调用

在进行采用相对路径设定/无路径设定的 EXTCALL 调用时，系统根据下列模式查找存在的程序存储器：

- 如果在 SD42700 $SC_EXT_PROG_PATH 中预设了路径设定，则首先从此路径出发查找 EXTCALL 调用中的设定（程序名或者相对路径设定）。而绝对路径由字符串组成：
  - SD42700 中预设的路径说明
  - "/" 为分隔符
  - EXTCALL 中指定的子程序路径或者子程序名称
- 如果没有在预设的路径下找到调用的子程序，接下来系统会从用户存储器的目录查找 EXTCALL 调用中的设定。
- 一旦找到子程序，查找结束。如果没有查找到子程序，则程序中断。

示例

从本地驱动执行

主程序：

程序代码

```
N010 PROC MAIN
N020 ...
N030 EXTCALL ("SCHRUPPEN")
N040 ...
N050 M30
```

外部子程序：

程序代码

```
N010 PROC SCHRUPPEN
N020 G1 F1000
N030 X= ... Y= ... Z= ...
N040 ...
...
...
N999999 M17
```

主程序“Main.mpf”位于 NC 存储器中，并已选择执行该程序：

需要下载的子程序“SCHRUPPEN.SPF”或“SCHRUPPEN.MPF”位于本地驱动器的目录“/user/sinumerik/data/prog/WKS.DIR/WST1.WPD”下。

在 SD42700 中，子程序路径的预设为：

SD42700 $SC_EXT_PROG_PATH = "LOCAL_DRIVE:WKS.DIR/WST1.WPD"

说明

未在 SD42700 中设定路径时，必须为此示例编程以下 EXTCALL 指令：

```
EXTCALL("LOCAL_DRIVE:WKS.DIR/WST1.WPD/SCHRUPPEN")
```

3.14 从外部存储器执行（选件）

3.14.1 功能

说明

使用该功能必须具有需要购买授权的选件"CNC 用户存储器扩展"或"从外部存储器执行 (EES)"！

功能

EES（Execution from External Storage，从外部存储器执行）功能可协助用户通过 NC **直接从一个外部存储器**执行程序。

以下驱动器可用作外部存储器：

- NC Extend（从前的"本地驱动器"）
  硬件组件：NCU 的存储卡**或** PCU/IPC 的本地硬盘
  符号名称：CF_CARD, LOCAL_DRIVE, SYS_DRIVE
  驱动器 NC Extend 被固定分配有符号名称 LOCAL_DRIVE、CF_CARD 和 SYS_DRIVE（⇒ NC Extend 可通过 LOCAL_DRIVE、CF_CARD 和 SYS_DRIVE 进行寻址）。
- 网络驱动器
- 静态引导 USB 驱动器

注意
USB 闪存可损坏刀具/工件 **不**建议使用 USB 闪存来执行外部程序。如在执行程序的过程中由于接触不良、脱落以及因碰撞或误拔出而中断与 USB 闪存的通讯，会导致加工意外停止。这可能会损坏刀具或/和工件。

前提条件

使用 EES 需要满足下列前提条件：

- 必须具有需要购买授权的选件"CNC 用户存储器扩展"或"从外部存储器执行（EES）"。
- 需要在控制系统上用作外部存储器的驱动器必须配置成逻辑驱动器（参见"调试 (页 233)"）。
- 可用 IPC 列表请参阅 可用的 IPC (页 927)。

工作模式

根据选件和驱动器配置，可采用不同的 EES 工作模式。控制系统的生效的工作模式通过机床数据 MD18045 $MN_EES_MODE_INFO 显示：

MD18045	工作模式	选件	外部存储器
= 0	EES 未生效	-	-
= 1	本地 EES 生效	6FC5800-0BP77-0YB0 CNC 用户存储器扩展	NCU 上 EES 的使用被限制于存储卡的扩展用户存储器（100 MB）。 使用 PCU/IPC 上的 EES 时，可以使用 NC Extend 中总的空白存储器。
			EES 不适用于网络或 USB。
		6FC5800-0BP12-0YB0 NCU/PPU 的 SD 卡上扩展的 HMI 用户存储器	使用 NCU 上的 EES 时，可通过附加选件将本地用户存储器扩展到最大 6 GB（取决于 MD9111）。 使用 PCU/IPC 上的 EES 时，不需要附加选件。
= 2	全局 EES 生效	6FC5800-0BP75-0YB0 从外部存储器执行（EES）	EES 适用于所有可用外部存储器。
		6FC5800-0BP12-0YB0 NCU/PPU 的 SD 卡上扩展的 HMI 用户存储器	使用 NCU 上的 EES 时，可通过附加选件将本地用户存储器扩展到最大 6 GB（取决于 MD9111）。 使用 PCU/IPC 上的 EES 时，不需要附加选件。

MD18045	工作模式	选件	外部存储器
= 5	本地 EES 生效 + 全局零件程序存储器	如 MD18045 = 1，区别仅在于在扩展用户存储器上增设了全局零件程序存储器 (页 235)。	
= 6	全局 EES 生效 + 全局零件程序存储器	如 MD18045 = 2，区别仅在于在外部存储器上增设了全局零件程序存储器 (页 235)。	

属性

EES 功能能够取代“外部执行 (页 226)”和“执行外部子程序（EXTCALL） (页 228)”功能。

EES 功能具有以下优点：

- 程序处理在系统内一致
- 可使用指令方面无限制
  使用“外部执行”和“执行外部子程序（EXTCALL）”时的限制，例如无反向跳转、跳转指令的跳转范围受缓存制约等，在 EES 功能中被取消。
- 可在各种程序存储器（NC、GDIR、外部驱动器）之间移动程序。
- 实际应用中对零件程序大小和程序数量无限制（其只受外部数据存储器容量制约）。
- 子程序调用的句法一致，与子程序的存储路径无关（不需要 EXTCALL 调用）。
- 网络驱动器可由多个工作站（PCU/IPC/NCU）共用。前提条件是这些工作站采用一致的驱动器配置。以获取统一程序视图。
- 所有工作站通过一致方式查看外部程序存储器，因此对外部存储器中保存的程序的修改持续对所有工作站生效。

3.14.2 调试

3.14.2.1 驱动器的配置

在使用 EES 功能前，必须确定控制系统上使用的驱动器。

更多信息

- 调试手册之基本软件和操作软件
- SINUMERIK Operate 操作手册

激活新的驱动器配置后，可将程序自由分配给可用驱动器。

说明

在新建的驱动器配置中，先前的驱动器可能并非均可用。因此，无法再访问驱动器中的程序。
解决方法：事先将该驱动器中的程序复制到另一个可访问驱动器中。

说明

受系统限制，受保护循环（CPF 文件）只能从 NC 零件程序存储器执行，因此**无法**保存用于外部存储器上的执行。

目前为止使用的、建有 MPF.DIR、SPF.DIR 和 WKS.DIR 目录的 NC 零件程序存储器对 EES 功能非强制必需。也可配置不采用 NC 零件程序存储器的系统。

注意

执行不可见程序

即便将 NC 零件程序存储器从驱动器配置中删除，其仍始终存在于系统内部。这特别是意味着，在执行程序时，该存储器的 SPF.DIR 目录下尚存在的程序也可能会被意外执行。
解决方法：为系统启用无 NC 零件程序存储器的组态时，手动删除该存储器可能尚存的程序。

如需继续使用 NC 零件程序存储器，建议不要将其从系统完全删除，必要时可为其启用较高的保护级。

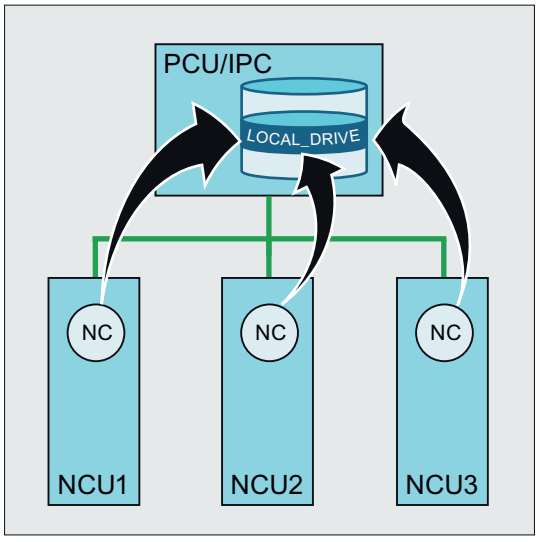
驱动器可由多个工作站（PCU/IPC//NCU）共用。为这些工作站启用一致的驱动器配置时，便可获取不受工作站制约的统一程序视图。

说明

NCU/PPU 的存储卡**不**可由多个工作站共享。

示例：

多个 NCU 共用一个位于 PCU/IPC 的本地硬盘（LOCAL_DRIVE）上的程序存储器。



说明

在 EES 运行中，若多个工作站共用外部程序存储器，则须遵循以下规则：

- 不可同时通过多个工作站编辑一个程序。
- 不可修改执行中的程序。

通过外部 HMI 运行时，必须启用外部 HMI 上的驱动器配置！必须从外部 HMI 将驱动器配置（logdrive.ini）载入至对应 NCU。驱动器配置对话框中有传输软键。

对于多 NC 协作的设备，所有 NC 的驱动器配置须相同。这通过将 logdrive.ini 文件分配给 mmc.ini 中列出的所有 NCU 来实现。这样便将该处的现有配置覆盖。

更多信息

PCU 基础软件的列表请参阅：

- SINUMERIK ONE PCU 基本软件调试手册
- 可用 IPC 列表请参阅 可用的 IPC (页 927)。

3.14.2.2 全局零件程序存储器（GDIR）

在确定驱动器时，可将其中一个驱动器定义为全局零件程序存储器（GDIR）。

更多信息

操作手册之通用/车削/铣削/磨削；管理程序 > 设置驱动器

系统会在该驱动器上自动创建 MPF.DIR、SPF.DIR 和 WKS.DIR 目录（GDIR 的标配目录）。这三个目录共同组成 GDIR。

GDIR 只用于 EES 功能。根据驱动器配置，GDIR 或取代 NC 零件程序存储器，或作为对其的扩展。GDIR 设置对于 EES 操作并不是必要的。

GDIR 目录和文件可以和在被动文件系统一样，在零件程序中以相同的方式写上地址。这样就可以将 NC 程序和路径信息从被动文件系统兼容转移到 GDIR。

对于不通过给定绝对路径调用的子程序，GDIR 将其的搜索路径扩展。

GDIR 取代 NC 零件程序存储器

当 NC 零件程序存储器的 MPF.DIR、SPF.DIR 和 WKS.DIR 目录完全为空时，GDIR 会取代 NC 零件程序存储器。目前的 NC 搜索路径通过 GDIR 1:1 映射。

在外部存储器上选择主程序：

子程序的查找顺序：

1. 外部存储器上的当前目录
2. GDIR 存储器中的 SPF.DIR
3. 通过 CALLPATH 指向的驱动器
4. 循环

GDIR 作为对 NC 零件程序存储器的扩展

若 NC 零件程序存储器的 MPF.DIR、SPF.DIR 和 WKS.DIR 目录中不为空，那么子程序的搜索序列取决于主程序的存储路径（生效的目录）。

在 NC 零件程序存储器中选择主程序（MPF.DIR 或 WKS.DIR 中的 xxx.WPD）

子程序的查找顺序：

1. NC 零件程序存储器中的当前目录
2. NC 零件程序存储器中的 SPF.DIR
3. 通过 CALLPATH 指向的驱动器
4. 循环

在 EES 共享外部存储器上选择主程序

子程序的查找顺序：

1. 外部存储器上的当前目录
2. NC 零件程序存储器中的 SPF.DIR
3. GDIR 中的 SPF.DIR
4. 通过 CALLPATH 指向的驱动器
5. 循环

说明

查找顺序的确定另见 MD11625 $MN_FILE_ONLY_WITH_EXTENSION 和 MD11626 $MN_CYCLES_ONLY_IN_CYCDIR！

说明

通过 CALLPATH 指令也可指向外部驱动器。

3.14.2.3 EES 功能下零件程序中的文件处理设置

设备内唯一的程序名称

在 EES 运行中，若多个工作站共用外部程序存储器，那么在各工作站上同时执行文件操作（WRITE、DELETE、…）可能会导致访问冲突。为了避免此类访问冲突，建议在每个工作站上为文件名称设置一个设备内唯一的命名空间。

设备内唯一的命名空间

文件名称的设备内唯一命名空间例如可如下实现：将文件名称与机床数据中可参数设置的 EES 专用 NC 名称，以及执行程序的通道的编号相连。例如，在执行程序时，以下程序会将 EES 专用的 NC 名称（NC1）和通道编号（通道 1）附加至程序名称，从而生成一个设备内唯一的文件名（MYFILE _NC1_1.SPF）。

```
$MN_EES_NC_NAME="NC1"
N10 DEF STRING[31] FILENAME
N20 FILENAME="MYFILE_" << $MN_EES_NC_NAME << "_" << $P_CHANNO <<
".SPF"
```

参数设置

NC 的 EES 专用名称通过以下 NC 专用机床数据设置：

MD10125 $MN_EES_NC_NAME = <NC 名称>

说明

设备内唯一的 NC 名称

为了避免访问冲突，NC 的 EES 专用名称须为设备内唯一。相关责任由机床制造商/用户自行担负。

调用程序时仅搜索有文件标识的文件

为了在调用子程序时加快程序搜索，建议只搜索**带有**文件标识（例如：SPF、MPF 等）的文件：

MD11625 $MN_FILE_ONLY_WITH_EXTENSION = **1**

说明

在通过 EXTCALL 执行外部子程序时，MD11625 对程序搜索没有影响。

更多信息

对子程序调用搜索路径的说明参见：编程手册"NC 编程"。

仅在循环目录下搜索含接口的程序

为了在调用子程序时加快程序搜索，建议在搜索创建了接口描述（借助 PROC 指令）、且从循环目录（CUS、CMA、CST）生成接口描述的子程序时，可将搜索范围限制为循环目录：

MD11626 $MN_CYCLES_ONLY_IN_CYCDIR = **1**

说明

MD11626 对通过 EXTERN 说明创建接口的子程序无影响。对此类子程序的搜索会在所有程序目录下执行。

注意
搜索不到循环目录外的循环 通过设置 MD11626 = 1 无法再找到当前目录和本地子程序目录中的循环！ 解决方法：将循环始终保存在循环目录中。

3.14.2.4 存储器配置

缩小被动文件系统中的用户程序存储器容量

EES 激活时，可缩小被动文件系统中的用户程序存储器容量：

MD18352 $MN_MM_U_FILE_MEM_SIZE（用于零件程序/循环/文件的用户存储器容量）

未占用的存储器容量可用于刀具数据或制造商循环 (MD18353 $MN_MM_M_FILE_MEM_SIZE)。

更多信息
有关存储器配置的详细信息参见 存储器配置 (页 899)。

共享缓存

EES 功能可代替“外部执行”和“执行外部子程序 (EXTCALL)”功能。

如通过 EXTCALL 调用（而非“执行外部子程序 (EXTCALL)”功能）执行零件程序中的子程序，必须将 EXTCALL 调用转换为 CALL 调用并修改路径信息。

转换完成后可共享“外部执行”和“执行外部子程序 (EXTCALL)”所需的缓存（FIFO 缓存）。

MD18362 $MN_MM_EXT_PROG_NUM（可同时外部执行的程序级的数量）= 0

3.14.3 边界条件

示教

在 EES 运行中，**不**可在 AUTO 运行模式下使用“示教”功能。

3.15 Process DataShare - 数据输出到外部设备/文件上

3.15.1 功能

通过“Process DataShare”功能，可以把数据从零件程序输出到外部设备/外部文件中，以便如记录生产数据，调控操作系统上的附加装置等。

可用性

该功能可用在：

- 只在真实的 NC 中（**不是**在模拟软件 SNC 和 VNC 中）。
- 仅限于零件程序（**不**用于同步动作）。
- 并行用于 NC 的所有处理通道，用于所有可用的输出设备。

外部设备/文件

外部设备/文件可以是：

- 本地 SD 卡上的文件
  本地 SD 卡是 HMI 上的标识符"LOCAL_DRIVE"对应的存储器。在 SINUMERIK ONE 上，该存储器就是本地驱动。

  说明

  在 SINUMERIK ONE 上向"LOCAL_DRIVE"输出数据时，需要使用选件"NCU 存储卡上的附加 xxx MB HMI 用户存储器"。

- 网络驱动上的文件
- V.24 接口

  说明

  在 SINUMERIK ONE 上向 V.24 接口输出数据时，需要使用 NCU 选件"RS232 接口"。

打开的外部设备的最大数量

在零件程序/通道中，也可以有多个外部设备/文件。除了所有 NC 通道外，可以同时最多打开 10 个输出设备。另外，还有两个备用的条目，用于西门子循环。

这些输出设备最多可以同时执行 5 个任务。

使用模式

在打开每个输出设备时都可以指定，该设备是仅被一个通道占用，还是被多个通道共同使用（即"分享"模式）。

零件程序结束和通道复位时的特性

程序段结束和通道的复位会关闭所有在该通道中打开的外部设备/文件。

使用数据传输（至控制系统）功能

注意
数据安全 在使用 Process DataShare 功能将数据从一个外部设备通过以太网 - 接口 X130 传输到控制系统时，可能会混入第三方数据，导致数据不一致。因此，在使用该功能时须确保网络能够阻止第三方访问。

3.15.2 调试

在文件"/oem/sinumerik/nck/extdev.in"或"/user/sinumerik/nck/extdev.ini."中可以配置需要使用的外部设备。如果这两个文件同时存在，后一个文件中的条目优先。可以在"调试"操作区的"系统数据/CF 卡"中编辑该文件。

说明

使用"LOCAL_DRIVE"和"CYC_DRIVE"时，无需在文件"extdev.ini"中开展配置。只要有用户 CF 卡，并且设置了对应的选件，就一直可以使用这两个设备。

在文件"extdev.ini"的段落[ExternalDevices]中，可以定义/计数需要使用的外部设备。一个串行设备(/dev/v24)和最多九个文件或目录(/dev/ext/1...9)可以指定为"设备"。输出的记法采用的是 Linux 记数法。以"；"开始的行都是注释，可以忽略。

以"/"符号结尾的设备是目录路径，以文件名称结尾的（无"/"）设备是文件路径，也就是全验证路径，但 /dev/v24 是特例。对于"目录路径"型设备，在零件程序中还必须一同输入文件名称（路径）。

设备由三个用逗号隔开的信息定义："服务器"、"路径"和可选的"写入模式"，
但 /dev/v24 是特例。

可以对文件或目录进行设置：在打开文件后，输出数据是覆盖文件（"O" = 覆盖）还是添加到文件中 ("A" = 添加）。对目录进行的设置会作用于目录下的所有文件。缺省值是"A"。在打开文件/目录时，请新建缺少的文件/目录。

设备"V.24 接口"请按照传输率、数据位、停止位、奇偶位、协议和结束位的顺序依次设置。

在需要将数据输出/保存到"LOCAL_DRIVE"上时，请通过"LOCAL_DRIVE_MAX_FILESIZE "设置一个最大文件大小，单位是字节，该设置对所有文件统一生效。在"添加模式"中执行

EXTOPEN 指令时，会检测文件的大小。也可以用“LOCAL_DRIVE_FILE_MODE”来确定写入模式（O= 覆盖，A= 添加）。缺省值是“A”。

说明

在目录“/siemens/sinumerik/nck”中有一份配置文件“extdev.ini”的备份。

说明

只有在重新启动 NC 后，文件“extdev.ini”上的修改才生效。

说明

USB 设备

在 SINUMERIK ONE 上，只能把 TCU 上稳态连接的 USB 接口定义为 USB 设备。它通过“SERVER:/PATH ”配置，其中“SERVER”是 TCU 的名称，“/PATH”是 USB 接口。TCU 上的 USB 接口名称为“dev0-0”、“dev0-1”、“dev1-0”。路径由“/Partition”引导，其中，分区由两位的分区号或分区名称指定，有时还补充有文件路径，例如：

/dev/ext/8 = "TCU4:/dev0-0, /01/, A"

/dev/ext/8 = "TCU4:/dev0-0, /01/mydir.dir/"

/dev/ext/8 = "TCU4:/dev0-0, /myfirstpartition/Mydir.dir/myfile.txt, O"

示例

```
[ExternalDevices]
；注释行
; V24 示例
; /dev/v24 = "9600, 8, 1, none, rts [, etx]"
；网络驱动示例
; /dev/ext/1 = "//[USERNAME[/DOMAIN][%PASSWORD]@]SERVER/SHARE/, /, A"
; /dev/ext/2 = "//[USERNAME[/DOMAIN][%PASSWORD]@]SERVER/SHARE, /myfile.txt, O"
; /dev/ext/3 = "//[USERNAME[/DOMAIN][%PASSWORD]@]SERVER/SHARE, /mydir/, A"
; /dev/ext/4 = "SERVER:/dev0-0, /01/, A"
; ...
；仅限西门子
; /dev/cyc/1= "//[USERNAME[/DOMAIN][%PASSWORD]@]SERVER/SHARE, /mydir/, A"
; /dev/cyc/2= "//[USERNAME[/DOMAIN][%PASSWORD]@]SERVER/SHARE/mydir, /, A"
```

LOCAL_DRIVE_MAX_FILESIZE = 50000（50 kByte）

LOCAL_DRIVE_FILE_MODE = “O”

EXTOPEN 参数的有效性 <写入模式>

由于写入模式既可以在文件 “extdev.ini” 中定义，又可以在 EXTOPEN 指令中定义，因此可能会出现设置冲突，导致 EXTOPEN 指令报错。

“extdev.ini”文件中的值	**EXTOPEN 参数值**		
	“OVR”	**“APP”**	**-**
“O”	O	错误	O
“A”	错误	A	A
-	O	A	A
	说明： O：“覆盖”模式有效。 A:“添加”模式有效。 错误：EXTOPEN 指令报错。		

LOCAL_DRIVE：文件属性

通过 EXTOPEN 在 “LOCAL_DRIVE” 上打开的文件具有以下属性：

- 所有人： "user" 设置了读写权限
- 组群： "operator" 设置了读写权限

3.15.3 编程

将数据从零件程序写入外部设备/文件需要三步：

1. 打开外部设备/文件
   通过 EXTOPEN 指令打开外部设备/文件。
2. 写入数据
   可以用 NC 语言的字符串函数（“”）来处理输出数据，例如 SPRINT 函数。而写入过程本身通过 WRITE 指令执行。
3. 关闭外部设备/文件
   通过指令 EXTCLOSE、达到程序结束 M30 或通道复位，再次关闭通道中的外部设备/文件。

句法

```
DEF INT <Result>
DEF STRING[<n>] <Output>
…
EXTOPEN(<Result>,<ExtDev>,<SyncMode>,<AccessMode>,<WriteMode>)
…
<Output>="输出数据"
WRITE(<Result>,<ExtDev>,<Output>)
…
EXTCLOSE(<Result>,<ExtDev>)
```

含义

EXTOPEN:	打开外部设备/文件的预定义步骤		
<Result>:	**参数 1：**结果变量 通过结果变量值可以检查程序中指令的执行情况。		
	类型：	INT	
	数值：	0	无错误
		1	外部设备无法打开
		2	没有配置外部设备
		3	外部设备配置了无效路径
		4	缺少对外部设备的存储权限
		5	使用模式：外部设备被设为“独占”
		6	使用模式：外部设备被设为“共享”
		7	文件长度大于“LOCAL_DRIVE_MAX_FILESIZE”
		8	超过允许的外部设备最大数量
		9	没有勾选选项“LOCAL_DRIVE"
		11	预留
		12	写入模式：输入和“extdev.ini”矛盾
		16	写入了无效的外部路径
		22	外部设备没有安装

<table>
<tr><td rowspan="8"><ExtDev>:</td><td colspan="2">参数 2：需要打开的外部设备/文件的标识符</td></tr>
<tr><td>类型：</td><td>STRING</td></tr>
<tr><td colspan="2">标识符由以下字符组成：
1. 逻辑设备名称
2. 必要时也包含文件路径（前面带“/”）。
定义了以下逻辑设备名称：</td></tr>
<tr><td>“LOCAL_DRIVE”:</td><td>本地 SD 卡（预定义）</td></tr>
<tr><td>“CYC_DRIVE”:</td><td>预留给西门子循环使用的驱动器（预定义）</td></tr>
<tr><td>"/dev/ext/1", ... "/dev/ext/9":</td><td>可用的网络驱动
注：
必须在文件 extdev.ini 中进行配置！</td></tr>
<tr><td>"/dev/cyc/1", "/dev/cyc/2":</td><td>预留给西门子循环使用的驱动器
注：
必须在文件 extdev.ini 中进行配置！</td></tr>
<tr><td colspan="2">文件路径：
• 必须为 “LOCAL_DRIVE” 和 “CYC_DRIVE” 指定文件路径，例如：“LOCAL_DRIVE/my_dir/my_file.txt”
• 逻辑设备名称 “/dev/ext/1...9” 和 “/dev/cyc/1...2” 可以通过配置：
– 链接到一个文件，但是只允许输入逻辑设备名称，如：“/dev/ext/4”
– 链接到一个目录，但是必须输入文件路径：“/dev/ext/5/my_dir/my_file.txt”</td></tr>
<tr><td colspan="2">注：
逻辑设备名称“/dev/ext/1...9”，“/dev/v24”和“/dev/cyc/1...2”不区分大小写，但文件路径区分大小写。“LOCAL_DRIVE” 和 “CYC_DRIVE” 只允许大写字母。</td></tr>
</table>

<SyncMode>:	**参数 3：**WRITE 指令的处理模式，向该设备/文件输入数据		
	类型：	STRING	
	数值：	"SYN"：	同步写入 数据写入结束后，才继续处理程序， 查看 WRITE 指令中的错误变量，可以检查同步写入是否顺利结束。
		"ASYN"：	异步写入 WRITE 指令不会中断程序处理。 **注：** WRITE 指令的结果变量在该模式下无效，而且一直显示为0（无错误）。因此，在该模式下无法确保 WRITE 指令成功执行。
<AccessMode>:	**参数 4：**设备/文件的使用模式		
	类型：	STRING	
	数值：	"SHARED":	设备/文件进入“分享”模式，其他通道也可以使用该设备，即同样也在该模式下打开。
		"EXCL"：	设备/文件在一个通道中单独使用，其他通道不可使用该设备。
<WriteMode>:	**参数 5：**WRITE 指令的写入模式，向设备/文件输出数据（可选）		
	类型：	STRING	
	数值：	"APP"：	添加 文件内容保持不变，输出的数据添加到结尾处。
		"OVR"：	覆盖 输出的数据覆盖旧文件内容。
	注： 通过该参数， 不能覆盖 extdev.ini 文件中的写入模式。在出现冲突的情况下，EXTOPEN 指令会报告错误。		

WRITE:	写入输出数据的预定义步骤

EXTCLOSE:	关闭已打开的外部设备/文件的预定义步骤		
<Result>:	**参数 1：**结果变量		
	类型：	INT	
	数值：	0	无错误
		16	写入了无效的外部路径
		21	关闭外部设备出错
<ExtDev>:	**参数 2：**需要关闭的外部设备/文件的标识符，详细信息参见 EXTOPEN！ **注：** 标识必须与 EXTOPEN 调用中输入的标识一致！		

示例

程序代码

```
N10       DEF INT RESULT
N20       DEF BOOL EXTDEVICE
N30       DEF STRING[80] OUTPUT
N40       DEF INT PHASE
N50       EXTOPEN(RESULT,"LOCAL_DRIVE/my_file.txt","SYN","SHARED")
N60       IF RESULT > 0
N70          MSG（“EXTOPEN 出错：” << RESULT）
N80       ELSE
N90          EXTDEVICE=TRUE
N100      ENDIF
…
N200      PHASE=4
N210      IF EXTDEVICE
N220         OUTPUT=SPRINT（“结束相位：%D”，PHASE）
N230         WRITE(RESULT,"LOCAL_DRIVE/my_file.txt",OUTPUT)
N240      ENDIF
…
```

3.15.4 边界条件

对连续路径运行的影响

EXTOPEN、WRITE 和 EXTCLOSE 每次都会触发预处理停止，从而中断连续路径运行。

程序段搜索动作

在“带计算的程序段搜索期间”，WRITE 指令不会输入任何结果。但是，在找到搜索目标，并按下“NC 启动”后，EXTOPEN 和 EXTCLOSE 会变为有效。后面的 WRITE 指令因此处于正常的程序环境下。

在“程序测试”模式下进行“带计算的程序段搜索”时，EXTOPEN、WRITE 和 EXTCLOSE 按照正常的程序处理过程执行。

3.16 启动、复位/零件程序结束和零件程序开始的系统设置

3.16.1 启动、复位/零件程序结束和零件程序开始的系统设置

方案

可通过机床数据针对以下事件设置控制系统特性：

- 启动（上电）
- 复位/零件程序结束
- 零件程序开始

下列情形后的控制系统特性：	可通过哪些机床数据设置：
启动（上电）*)	MD20110 $MC_RESET_MODE_MASK MD20144 $MC_TRAFO_MODE_MASK MD20150 $MC_GCODE_RESET_VALUES
复位/零件程序结束	MD20110 $MC_RESET_MODE_MASK MD20150 $MC_GCODE_RESET_VALUES MD20152 $MC_GCODE_RESET_MODE
零件程序开始	MD20112 $MC_START_MODE_MASK MD20110 $MC_RESET_MODE_MASK
*) 另见 上电 (页 438)	

启动后的系统设置

MD20110 $MC_RESET_MODE_MASK，位 0 = 0 或 1

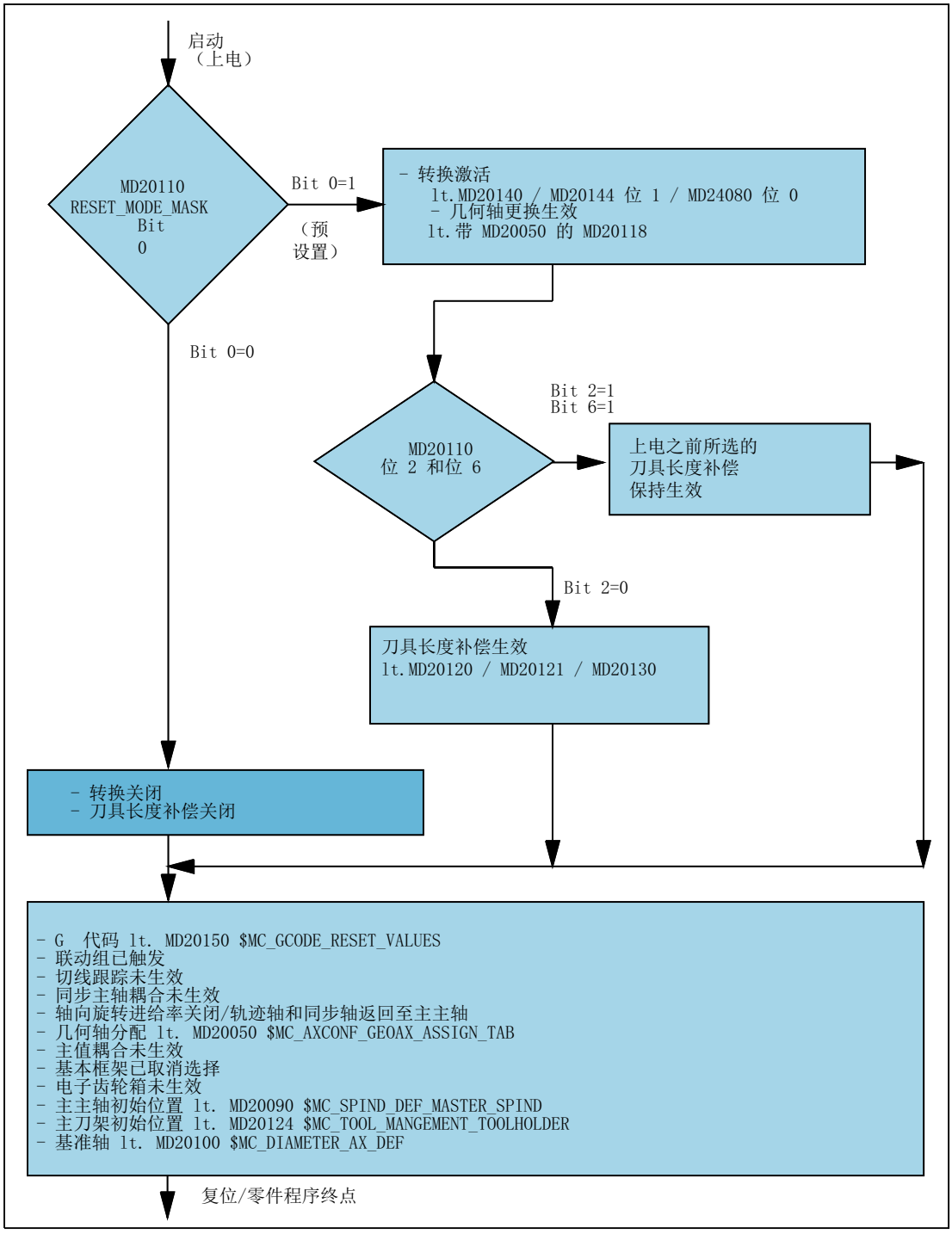


图 3-7　启动（上电）后的系统设置

复位/零件程序结束和零件程序开始后的系统设置

MD20110 $MC_RESET_MODE_MASK，位 0 = 0 或 1

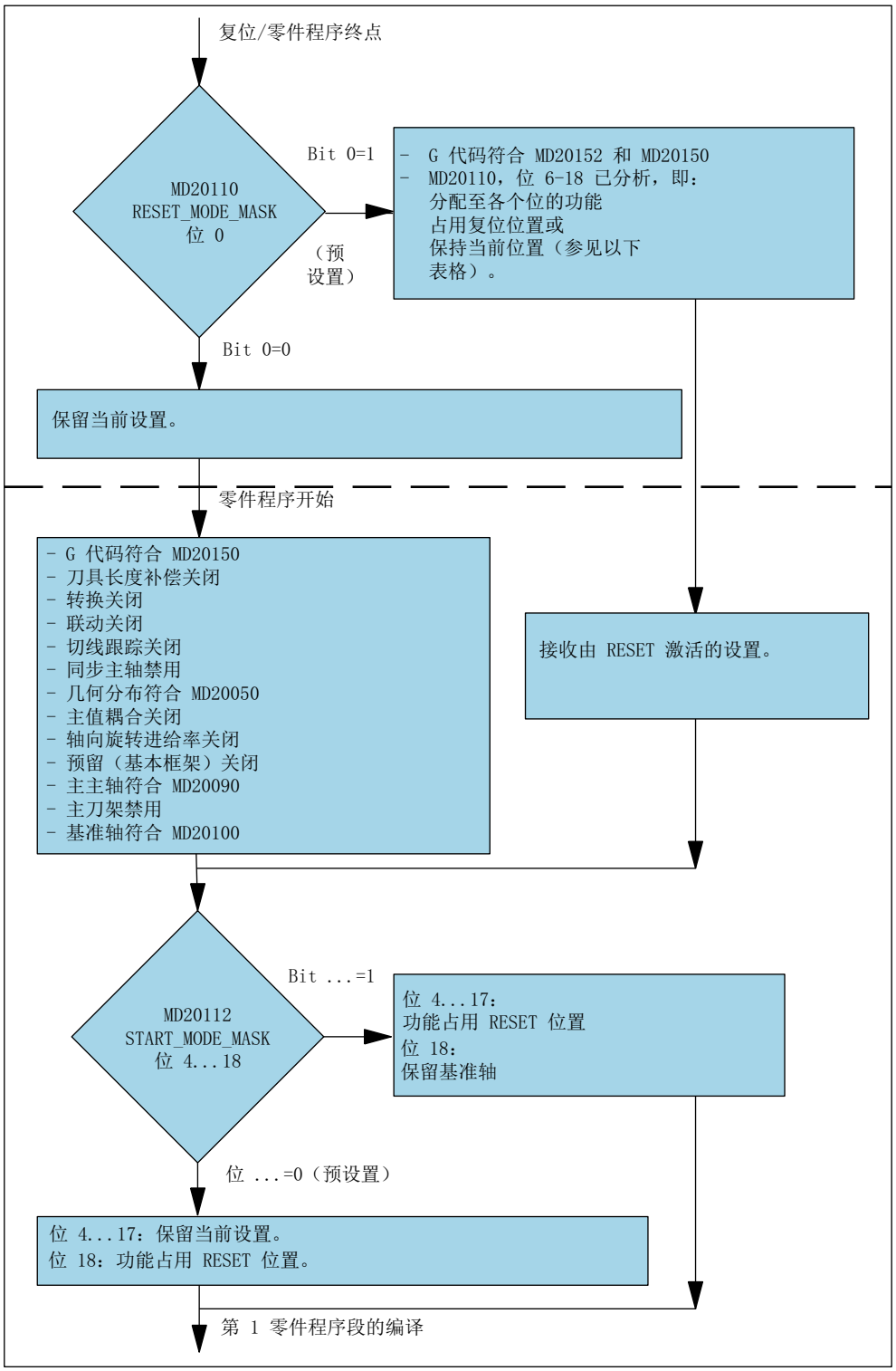


图 3-8　复位/零件程序结束和零件程序开始后的系统设置

启动和复位/零件程序结束后生效的 G 指令

启动（上电）和复位/零件程序结束后每个 G 功能组中生效的 G 代码通过以下机床数据设置：

MD20150 $MC_GCODE_RESET_VALUES[<**G 组**>] = <缺省 G 代码>

MD20152 $MC_GCODE_RESET_MODE[<**G 组**>] = <值>

值	说明：针对每个 G 功能组
0	MD20150 $MC_GCODE_RESET_VALUES 中的缺省 G 指令生效。
1	最后生效的/当前 G 指令生效

启动、复位/零件程序结束和零件程序开始后的系统初始设置

通过以下机床数据定义启动（上电）、复位/零件程序结束和零件程序开始后的系统初始设置

- MD20110 $MC_RESET_MODE_MASK（定义**启动**和**复位/零件程序结束**后的控制系统初始设置）
- MD20112 $MC_START_MODE_MASK（定义**零件程序开始**后的控制系统初始设置）

更多信息

详细机床数据说明

机床数据关联

机床数据	含义
MD20120 $MC_TOOL_RESET_VALUE	启动、复位/零件程序结束时的刀具长度补偿
MD20121 $MC_TOOL_PRESEL_RESET_VALUE	复位时的预选刀具
MD20130 $MC_CUTTING_EDGE_RESET_VALUE	启动时的刀沿长度补偿
MD20140 $MC_TRAFO_RESET_VALUE	启动转换数据组
MD20144 $MC_TRAFO_MODE_MASK	坐标转换功能的选择
MD20150 $MC_GCODE_RESET_VALUES	G 功能组的初始设置
MD20152 $MC_GCODE_RESET_MODE	G 功能组的复位特性
MD21330 $MC_COUPLE_RESET_MODE_1	耦合撤销特性
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB	几何轴指定为通道轴
MD20118 $MC_GEOAX_CHANGE_RESET	允许自动几何轴切换

示例

复位时激活复位设置：

- MD20110，位 0 = 1
- MD20112 = 0

复位/零件程序启动时保留转换：

- MD20110，位 0 = 1
- MD20110，位 7 = 1
- MD20112 = 0

复位/零件程序启动后保留刀具长度补偿：

- MD20110，位 4 = 1
- MD20110，位 6 = 1
- MD20112 = 0

复位后保留生效的级（位 4）和可设置的框架（位 5），零件程序开始时复位：

- MD20110，位 4 = 1
- MD20110，位 5 = 1
- MD20112，位 4 = 1
- MD20112，位 5 = 1

说明

MD20110 / MD20112，位 5 和位 6

在 AUTO 或 MDI 运行方式下启动零件程序时，若 MD20110 / MD20112 的参数设置使得刀具长度补偿或框架生效，第一个轴编程则须采用绝对尺寸，用于运行偏移。

例外：启用 MD42442 / MD42440 时 G91 下的偏移运行被抑制。

3.16.2 上电后通过定向转换回退刀具

功能

若包含刀具定向加工的零件程序由于掉电或复位而终止，那么在控制系统启动（上电）后可重新选择之前生效的转换并沿刀具轴的方向生成一个框架。之后可在 JOG 运行方式下沿刀具轴的方向回退刀具。

前提条件

对于参与转换的所有机床轴，生效的测量系统必须采用一个机床参考。参见功能手册“进给轴和主轴”，章节“自动恢复机床参考”。

参数设置

为了使最后生效的转换在上电后保留，必须设置以下机床数据：

- MD20144 $MC_TRAFO_MODE_MASK，位 1 = 1
- MD20110 $MC_RESET_MODE_MASK，位 0 = 1
- MD20110 $MC_RESET_MODE_MASK，位 7 = 1

另见章节“启动、复位/零件程序结束和零件程序开始的系统设置 (页 248)”。

编程

等待机床参考 WAITENC

通过 WAITENC 指令在程序中针对通道进行等待，直至参数设置的轴的所有测量系统均为同一有效的机床参考。参见上文的“前提条件”。轴的参数设置如下：

MD34800 $MA_WAIT_ENC_VALID = 1

应用

在引导启动期间（前提条件：MD20108 位 3 = 1）要调用的事件控制用户程序（.../_N_CMA_DIR/_N_PROG_EVENT_SPF）中，必须使用 WAITENC 指令等待有效的轴位置可用。

之后可通过 NC 语言指令 TOROTX/TOROTY/TOROTZ 生成一个框架，将刀具轴置于 X 轴、Y 轴或 Z 轴方向。

示例

定向转换和采用增量编码器的定向轴。

配置：	含义：
MD10720 $MN_OPERATING_MODE_DEFAULT [0] = 6	在 JOG 运行方式下启动
MD30240 $MA_ENC_TYPE [0, <轴>] = 1	增量测量系统
MD34210 $MA_ENC_REFP_STATE [0, <轴>] = 3	对增量编码器使能轴位置恢复。
MD20108 $MC_PROG_EVENT_MASK = ‘H9’	在启动和零件程序开始时激活事件控制的用户程序（PROG_EVENT）。

配置：	含义：
MD20152 $MC_GCODE_RESET_MODE [52] = 1	复位后保留 TOFRAME。
MD20110 $MC_RESET_MODE_MASK = 'HC1'	复位后保留转换和刀具补偿。
MD20144 $MC_TRAFO_MODE_MASK = 'H02'	下电后保留转换。

事件控制的用户程序（.../_N_CMA_DIR/_N_PROG_EVENT_SPF）

```
；示例：引导启动和通过零件程序开始回退时
；激活框架沿刀具方向对准 WCS。
  IF $P_PROG_EVENT == 4；引导启动
       IF $P_TRAFO <> 0；已选择转换。
         WAITENC；等待有效的方向轴位置。
         TOROTZ；将 WCS 的 Z 轴转到刀具轴方向。
       ENDIF
       M17
  ENDIF
  IF $P_PROG_EVENT == 1；零件程序启动。
       TOROTOF；复位刀具框架。
       RET
  ENDIF
```

WAITENC 指令基本对应以下程序串（采用 AB 运动系统的 5 轴机床）：

```
WHILE TRUE；等待测量系统。
      IF (($AA_ENC_ACTIVE[X]==TRUE) AND ($AA_ENC_ACTIVE[Y]
      ==TRUE) AND ($AA_ENC_ACTIVE[Z]==TRUE) AND ($AA_ENC_ACTIVE[A]
      ==TRUE) AND ($AA_ENC_ACTIVE[B]==TRUE)) GOTOF GET_LABEL
      ENDIF
      G4 F0.5；等待时间 0.5 s
ENDWHILE
：位置同步
GET_LABEL:GET(X,Y,Z,A,B,)
```

继续程序执行

AUTO 运行方式

为了在 AUTO 运行方式下自动执行程序，必须对所有生效测量系统实际位置恢复的机床轴进行回参考点。

MDI 运行方式和溢出转存

在 MDI 运行方式和溢出转存中，也可采用恢复后的位置进行加工，而无需执行轴回参考点。为此，必须针对特定通道明确使能系统恢复后的位置启动：

MD20700 $MC_REFP_NC_START_LOCK = 2

前提条件

采用增量编码器和无实际值缓存的轴

假设采用增量编码器和**无**实际值缓存的轴在掉电时能够足够快地夹紧，以防止最后设定位置偏移。

3.17 由子程序替换功能

3.17.1 概述

功能

用户专用辅助功能（例如 M101）不会触发系统功能。其只会被输出至 NC/PLC 接口。辅助功能的功能性须由用户/机床制造商在 PLC 用户程序中实现。 下面将介绍配置用户专用子程序（替换子程序）调用的方法，来取代缺省设置下的“输出至 NC/PLC 接口”特性。

零件程序中仍编写 M101 功能。 但是在执行零件程序时则会调用替换子程序。 这样一来，该功能便由 NC 替换为子程序调用。 这会带来以下优势：

- 在调整生产流程时，现有的、经过验证的可靠零件程序可不经修改继续使用。 需要进行的调整转移到用户专用子程序内部进行。
- 在完整的 NC 语言功能的支持下，替换子程序能内能实现优越的功能性。
- NC 和 PLC 间的通讯开销得以省去。

可替换的功能

下列功能可通过子程序替换：

辅助功能	
M	开关功能
T	刀具选择
TCA	和刀具状态无关的刀具选择
D	刀具补偿
DL	附加刀具补偿

同步主轴耦合生效时的主轴相关功能	
M40	自动齿轮档切换
M41 - M45	齿轮档选择 1 ... 5
SPOS	主轴定位
SPOSA	主轴定位
M19	主轴定位

3.17.2 替换 M、T/TCA 和 D/DL 功能

3.17.2.1 替换 M 功能

简介

下列条件适用于 M 功能替换：

- 每个程序段只替换一个 M 功能。
- 需要替换 M 功能的程序段**不允许**包含以下元素：
  - M98
  - 模态子程序调用
  - 子程序返回
  - 零件程序结束
- 会触发系统功能的 M 功能不允许通过子程序替换（参见“不可替换的 M 功能”部分）。

参数设置

M 功能和子程序

M 功能和替换子程序通过以下机床数据进行参数设置：

- MD10715 $MC_M_NO_FCT_CYCLE[<**索引**>] = <M 功能编号>
- MD10716 $MC_M_NO_FCT_CYCLE_NAME[<**索引**>] = "<子程序名称>"

M 功能和对应的替换子程序通过相同的索引相关联。

示例： M 功能 M101 通过子程序 SUB_M101 替代，M 功能 M102 通过 SUB_M102 替代：

MD10715 $MC_M_NO_FCT_CYCLE[**0**]	= 101
MD10716 $MC_M_NO_FCT_CYCLE_NAME[**0**]	= "SUB_M101"
MD10715 $MC_M_NO_FCT_CYCLE[**1**]	= 102
MD10716 $MC_M_NO_FCT_CYCLE_NAME[**1**]	= "SUB_M102"

用于信息传输的系统变量

对于可自由选择的 M 功能，可通过系统变量（参见“系统变量 (页 262)”章节）获取被替换的 M 功能及其他功能（T、TCA、D、DL）的相关信息，用于替换子程序中的分析。 系统变量中包含的数据基于编写了待替换 M 功能的程序段。

M 功能通过机床数据 MD10715 $MC_M_NO_FCT_CYCLE[<**索引**>] 的索引选择，待替换的 M 功能在该机床数据中进行参数设置。

MD10718 $MC_ M_NO_FCT_CYCLE_PAR = <**索引**>

说明

在通过系统变量传输信息的 M 功能替换中，M 功能的地址扩展和功能值必须编写为常量值。

允许的编程：

- `M<功能值>`
- `M=<功能值>`
- `M[<地址扩展>]=<功能值>`

不允许的编程：

- `M=<变量 1>`
- `M[<变量 2>]=<变量 1>`

编程

M 功能的替换规则：

- 替换子程序在程序段末尾调用
- 在替换子程序内 M 功能不被替换
- 在 ASUB 中，仅当 ASUB 是在替换子程序内启动时，M 功能才会被替换。

不可替换的 M 功能

下列 M 功能作为预定义辅助功能，会触发系统功能，不允许由子程序替换：

- M0 ... M5
- M17, M30
- M19
- M40 ... M45
- M98，M99（仅在 MD18800 $MN_MM_EXTERN_LANGUAGE ≠ 0 时）

通过机床数据参数设置的用户专用 M 功能也会触发系统功能，因此同样不允许由子程序替换。

机床数据	含义
MD10714 $MN_M_NO_FCT_EOP	复位后生效的主轴用 M 功能
MD10804 $MN_EXTERN_CHAN_M_NO_SET_INT	用于激活 ASUB 的 M 功能（外部模式）
MD10806 $MN_EXTERN_CHAN_M_NO_DISABLE_INT	用于取消 ASUB 的 M 功能（外部模式）
MD10814 $MN_EXTERN_M_NO_MAC_CYCLE	M 功能宏调用
MD20094 $MC_SPIND_RIGID_TAPPING_M_NR	用于切换至开环控制轴运行的 M 功能
MD20095 $MC_EXTERN_RIGID_TAPPING_M_NR	用于切换至开环控制轴运行的 M 功能（外部模式）
MD22254 $MC_AUXFU_ASSOC_M0_VALUE	用于程序停止的附加 M 功能
MD22256 $MC_AUXFU_ASSOC_M1_VALUE	用于有条件停止的附加 M 功能
MD26008 $MC_NIBBLE_PUNCH_CODE	M 功能定义（步冲专用）
MD26012 $MC_PUNCHNIB_ACTIVATION	激活冲裁和步冲功能

说明

特例

MD22560 $MC_TOOL_CHANGE_M_CODE（使用 M 功能换刀）参数设置的 M 功能允许通过子程序替换。

3.17.2.2 替换 T/TCA 和 D/DL 功能

前提条件

下列前提条件适用于 T、TCA、D 和 DL 功能的替换：

- 每个程序段中最多只能有一个功能替换生效。
- 编写了功能替换的程序段**不允许**包含以下元素：
  - M98
  - 模态子程序调用
  - 子程序返回
  - 零件程序结束
- 如果在多刀选择时用 T = 刀位编号通过 MTL 地址进行多刀刀位编号的编程，则由 T 功能替换代替 MTL 地址。在替换子程序中可查询带有系统变量 $C_MTL_PROG 和 $C_MTL 的编程的值。

参数设置：替换子程序

替换子程序在下列机床数据中针对功能设定：

功能	机床数据
T	MD10717 $MN_T_NO_FCT_CYCLE_NAME
TCA	MD15710 $MN_TCA_CYCLE_NAME
D/DL	MD11717 $MN_D_NO_FCT_CYCLE_NAME

说明

建议为 T、TCA 和 D/DL 功能的替换使用相同的子程序。

参数设置：T 功能同时存在时 D 或 DL 功能的相关特性

一个程序段中同时编写了 D/DL 和 T 功能时，或者将 D/DL 编号作为参数传输至替换子程序，或者在调用替换子程序前执行 D/DL 功能。此特性可通过以下机床数据设置：

MD10719 $MN_T_NO_FCT_CYCLE_MODE（T 功能替换的参数设置）

位	值	含义
0	0	D 或 DL 编号在子程序中以系统变量的形式提供（缺省设置）。
	1	D 或 DL 编号直接在程序段中计算。 **提示：** 仅当通过 M 功能配置了换刀时，此功能才生效： MD22550 $MC_TOOL_CHANGE_MODE = 1 否则就总是传输 D 或 DL 值。

用于信息传输的系统变量

程序段中编写的功能的所有相关信息均通过系统变量（参见“系统变量 (页 262)”章节）提供给替换子程序。

系统变量中包含的数据基于编写了待替换功能的程序段。

参数设置：替换子程序的调用时间点

替换子程序的调用时间点通过以下机床数据设置：

MD10719 $MN_T_NO_FCT_CYCLE_MODE，位 1 和位 2

位 2	位 1	替换子程序的调用时间点
0	0	程序段末尾处 执行替换子程序后，通过触发替换的程序段之后的程序行继续插补。
0	1	程序段开始处 执行替换子程序后，对引起替换子程序调用的程序行进行解译。此时将不再处理地址 T、D/DL 和用于换刀的 M 功能。
1	-	程序段开始处和程序段末尾处 调用两次替换子程序。

用于调用时间点的系统变量

通过系统变量 $P_SUB_STAT 可确定替换是否生效；若为是，则还可确定替换子程序的调用时间（以程序段为基准）：

值	含义
0	替换未生效
1	替换生效，在程序段开始处调用子程序
2	替换生效，在程序段末尾处调用子程序

示例：T 功能的替换

参数设置	含义
MD22550 $MC_TOOL_CHANGE_MODE = 0	使用 T 功能换刀
MD10717 $MN_T_NO_FCT_CYCLE_NAME = "MY_T_CYCLE"	用于替换 T 功能的子程序的名称
MD10719 $MN_T_NO_FCT_CYCLE_MODE = 0	调用时间点：程序段末尾

编程	注释
`N110 D1`	`; D1`
`N120 G90 G0 X100 Y100 Z50`	`; D1 生效。`
`N130 D2 X110 Z0 T5`	`; D1 保持生效。T 功能在程序段末尾处被子程序调用 MY_T_CYCLE 替换。D2 通过一个系统变量提供给 MY_T_CYCLE。`

T 功能替换的详细示例请见章节：“换刀时 M/T 功能替换的示例 (页 264)”。

3.17.2.3 系统变量

简介

程序段中编写的功能（T/TCA、D/DL、M）的所有相关信息均通过系统变量提供给替换子程序。

特例

下列情形下不传输 D/DL 编号：

- MD10719 $MN_T_NO_FCT_CYCLE_MODE，位 0 = 1
- MD22550 $MC_TOOL_CHANGE_MODE = 1

与

- D/DL 与 T 功能或 M 功能编写在同一程序段中。

⚠ 小心
值未生效 系统变量中提供给替换子程序的值尚未生效。 用户/机床制造商须自行在替换子程序中进行适宜的编程作为补充。

系统变量

系统变量	含义
$C_M_PROG	编写了 M 功能时为 TRUE
$C_M	$C_M_PROG == TRUE 时，提供地址 M 的值 此时须区分两种情形： • 若针对使用 M 功能的换刀通过参数传输配置了子程序， 那么 $C_M 提供以下机床数据的值：MD10715 MN_M_NO_FCT_CYCLE • 若只为地址 T 和/或 D/DL 配置了一个子程序，且在该程序中同时编写了用于换刀的 M 功能和待替换地址中的一个，那么 $C_M 提供以下机床数据的值：MD22560 $MN_TOOL_CHANGE_M_MODE
$C_AUX_VALUE[0]	替换的 M 功能的值
$C_ME	$C_M_PROG == TRUE 时，提供 M 功能地址扩展的值
$C_AUX_EXT[0]	M 功能的地址扩展（与 $C_ME 相同）
$C_AUX_IS_QUICK[0]	编写了快速输出至 PLC 的 M 功能时为 TRUE
$C_T_PROG	编写了 T 功能时为 TRUE
$C_T	$C_T_PROG == TRUE 时，提供 T 功能的值
$C_TE	在以下情形下： • $C_T_PROG == TRUE • $C_TS_PROG == TRUE 提供 T 功能地址扩展的值
$C_TS_PROG	在 T 或 TCA 替换中编写了刀具名称时为 TRUE
$C_TS	$C_TS_PROG == TRUE 时，提供 T/TCA 替换中编写的刀具名称
$C_TCA	TCA 替换生效时为 TRUE

系统变量	含义
$C_DUPLO_PROG	在 TCA 替换中编写了副编号时为 TRUE
$C_DUPLO	$C_DUPLO_PROG == TRUE 时，提供编写的副编号的值
$C_THNO_PROG	在 TCA 替换中编写了刀架/主轴编号时为 TRUE
$C_THNO	$C_THNO_PROG == TRUE 时，提供编写的刀架/主轴编号的值
$C_D_PROG	编写了 D 功能时为 TRUE
$C_D	$C_D_PROG == TRUE 时，提供 D 功能的值
$C_DL_PROG	编写了 DL 功能时为 TRUE
$C_DL	$C_DL_PROG == TRUE 时，提供 DL 功能的值
$P_SUB_STAT	替换子程序的调用时间点，以程序段为基准
$C_MTL_PROG	TRUE，如果写入了地址 MTL
$C_MTL	$C_MTL_PROG == TRUE 时，提供地址 MTL 的值

3.17.2.4 示例：M 功能的替换

示例 1

通过调用子程序“SUB_M6”替换 M6 功能。

换刀相关信息通过系统变量传输。

参数设置

机床数据	含义
MD10715 $MN_M_NO_FCT_CYCLE[**2**] = 6	使用 M6 换刀
MD10716 $MN_M_NO_FCT_CYCLE_NAME[**2**] = "SUB_M6"	M6 的替换子程序
MD10718 $MN_M_NO_FCT_CYCLE_PAR = **2**	通过系统变量传输信息

主程序

编程	注释
PROC MAIN	
...	;
N10 T1 D1 M6	; M6 由子程序“SUB_M6”
	; 替换
...	;

编程	注释
N90 M30	

子程序"SUB_M6"

编程	注释
PROC SUB_M6	
N110 IF $C_T_PROG==TRUE	; IF 编写了 T 地址
N120 T[$C_TE]=$C_T	; 执行 T 选择
N130 ENDIF	; ENDIF
N140 M[$C_ME]=6	; 执行换刀。
N150 IF $C_D_PROG==TRUE	; IF 编写了 D 地址
N160 D=$C_D	; 执行 D 选择
N170 ENDIF	; ENDIF
N190 M17	

示例 2

使用 T 功能为换刀准备新刀具。换刀则通过 M6 功能实现。T 功能由子程序"MY_T_CYCLE"替换。D/DL 编号传输至子程序。

参数设置

参数设置	含义
MD22550 $MC_TOOL_CHANGE_MODE = 1	使用 T 功能准备换刀
MD10717 $MN_T_NO_FCT_CYCLE_NAME = "MY_T_CYCLE"	替换子程序
MD10719 $MN_T_NO_FCT_CYCLE_MODE = 0	传输 D/DL 编号

主程序

程序代码	注释
N210 D1	;
N220 G90 G0 X100 Y100 Z50	; D1 生效。
N230 D2 X110 Z0 T5	; D1 保持生效，编写的 D2
	; 作为变量传输给子程序
N240 M6	; 执行换刀

示例 3

使用 T 功能为换刀准备新刀具。换刀则通过 M6 功能实现。T 功能由子程序"MY_T_CYCLE"替换。D/DL 编号不传输至子程序。

参数设置

参数设置	含义
MD22550 $MC_TOOL_CHANGE_MODE = 1	使用 T 功能准备换刀
MD10717 $MN_T_NO_FCT_CYCLE_NAME = "MY_T_CYCLE"	替换子程序
MD10719 $MN_T_NO_FCT_CYCLE_MODE = 1	不传输 D/DL 编号

主程序

程序代码	注释
N310 D1	
N320 G90 G0 X100 Y100 Z50	; D1 生效。
N330 D2 X110 Z0 T5	; D2 生效，不作为变量
	; 传输替换子程序。
N340 M6	; 执行换刀。

示例 4

通过子程序"MY_T_CYCLE"替换 T 功能和 M6 功能。

参数在 M6 替换时传输至子程序。

若程序段中同时编写了 M6 和 D/DL，那么即使未设置 D/DL 编号的传输，D 编号或 DL 编号也会作为参数传输至子程序。

MD10719 $MN_T_NO_FCT_CYCLE_MODE = 1

参数设置

配置	含义
MD22550 $MC_TOOL_CHANGE_MODE = 1	使用 M 功能换刀
MD22560 $MC_TOOL_CHANGE_M_CODE = 6	用于换刀的 M 代码
MD10715 $MC_M_NO_FCT_CYCLE[3] = 6	待替换 M 功能
MD10716 $MC_M_NO_FCT_CYCLE_NAME[3] = "MY_T_CYCLE"	M 功能的替换子程序
MD10717 $MN_T_NO_FCT_CYCLE_NAME = "MY_T_CYCLE"	T 功能的替换子程序
MD10718 $MN_M_NO_FCT_CYCLE_PAR = 3	参数传输至 M6 的替换子程序
MD10719 $MN_T_NO_FCT_CYCLE_MODE = 1	不传输 D/DL 编号

主程序

程序代码	注释
N410 D1	
N420 G90 G0 X100 Y100 Z50	; D1 生效。
N330 D2 X110 Z0 T5 M6	; D1 保持生效，D2 和 T5 作为变量传输至 M6 替换子程序。

3.17.2.5 示例：T 功能和 D 功能的替换

通过调用子程序"D_T_SUB_PROG"替换 T 功能和 D 功能。此示例还遵循以下条件：

- 换刀通过 T 地址进行。
- 子程序调用在程序段开始处进行。
- 刀具管理不生效。
- 轴 B 为配备端齿盘的分度轴。

参数设置

机床数据	含义
MD11717 $MN_**D**_NO_FCT_CYCLE_NAME = "**D_T_SUB_PROG**"	D 功能的替换子程序
MD10717 $MN_**T**_NO_FCT_CYCLE_NAME = "**D_T_SUB_PROG**"	M 功能的替换子程序
MD10719 $MN_T_NO_FCT_CYCLE_MODE = 'H2'	在程序段开始处调用
MD22550 $MC_TOOL_CHANGE_MODE = 0	使用 T 功能换刀

主程序

编程	注释
PROC MAIN	
...	;
N10 G01 F1000 X10 T1=5 D1	; 替换 T 功能和 D 功能，通过
	; 在程序段开始处调用"D_T_SUB_PROG"实现
...	;
N90 M30	

子程序"D_T_SUB_PROG"

编程	注释
N1000 PROC D_T_SUB_PROG DISPLOF SBLOF	

编程	注释
N4100 IF $C_T_PROG==TRUE	; IF 编写了 T 地址
N4120 POS[B]=CAC($C_T)	; 逼近分度位置
N4130 T[$C_TE]=$C_T	; 选择刀具（T 选择）
N4140 ENDIF	; ENDIF
N4300 IF $C_D_PROG==TRUE	; IF 编写了 D 地址
N4320 D=$C_D	; 选择补偿（D 选择）
N4330 ENDIF	; ENDIF
N4400 IF $C_DL_PROG==TRUE	; IF 编写了 DL 地址
N4420 D=$C_DL	; 选择附加补偿
N4430 ENDIF	; ENDIF
N9999 RET	

3.17.2.6 冲突情形下的特性

冲突情形

在一个程序段中编写了多个功能，而这些功能需要使用不同的子程序进行替换时，会出现冲突。

- D 和 DL 地址通过以下子程序替换：
  MD11717 $MN_FCT_CYCLE_NAME = "**D_SUB_PROG**"
- T 地址通过以下子程序替换：
  MD10717 $MN_FCT_CYCLE_NAME = "**T_SUB_PROG**"
- M6 功能通过以下子程序替换：
  MD10715 $MN_M_NO_FCT_CYCLE[0] = 6
  MD10716 $MN_M_NO_FCT_CYCLE_NAME[0] = "**M6_SUB_PROG**"
  MD10718 $MN_M_NO_FCT_CYCLE_PAR = 0
  MD22550 $MC_TOOL_CHANGE_MODE = 1
  MD22560 $MC_TOOL_CHANGE_M_CODE = 6

解决方案

冲突情形依据下表处理：

在一个程序行中编写了：			**调用的子程序：**
D 和/或 DL	**T 或 TCA**	**M6**	
–	–	x	**M6**_SUB_PROG
–	x	–	**T**_SUB_PROG
–	x	x	**M6**_SUB_PROG
x	–	–	**D**_SUB_PROG
x	–	x	**M6**_SUB_PROG
x	x	–	**T**_SUB_PROG
x	x	x	**M6**_SUB_PROG

3.17.3 替换主轴功能

3.17.3.1 简介

功能

耦合生效时，可为引导主轴替换以下主轴功能：

- M40：自动齿轮档切换
- M41 ... M45：编程的齿轮档切换
- SPOS, SPOSA 和 M19：主轴定位

前提条件

- 主轴功能的替换须满足以下条件：
  - 编写的主轴必须为生效耦合中的引导主轴。
  - 引导主轴和跟随主轴必须处于同一通道中。 这一点仅在引导主轴处于耦合关闭的通道中时才能识别出。 若引导主轴被交换至另一通道，那么该主轴的齿轮档切换或定位不会触发替换子程序的调用。
  - 编写的齿轮档切换必须能够产生实际的齿轮档切换。 为此须区分编写的齿轮档和生效的齿轮档。
- 在一个程序段中只能替换一个主轴功能。 多次替换会导致程序执行终止。 因此须将需要替换的主轴功能分别写入多个程序段。

参数设置

主轴功能

需要通过子程序替换的主轴功能在以下机床数据中选择：

MD30465 $MA_AXIS_LANG_SUB_MASK

位	含义	
0	自动齿轮档切换（M40）和直接齿轮档切换（M41-M45）	
	值	含义
	0	不替换
	1	由 MD15700 和 MD15702 中设置的子程序替换
1	通过 SPOS/SPOSA/M19 定位主轴	
	值	含义
	0	不替换
	1	由 MD15700 和 MD15702 中设置的子程序替换

子程序：名称

替换子程序的名称在以下机床数据中输入：

MD15700 $MN_LANG_SUB_NAME = "<子程序名称>"

子程序：路径

替换子程序的路径在以下机床数据中设置：

MD15702 $MN_LANG_SUB_PATH = <值>

值	含义
0	**制造商**循环目录：/_N_**CMA**_DIR
1	**用户**循环目录：/_N_**CUS**_DIR
2	**西门子**循环目录：/_N_**CST**_DIR

系统变量：替换子程序的调用时间点

替换子程序的调用时间点可通过系统变量 $P_SUB_STAT 读取：

值	含义
0	替换未生效
1	替换生效，在程序段开始处调用子程序
2	替换生效，在程序段末尾处调用子程序

程序段执行

若替换子程序在程序段开始处调用，那么系统会在执行完替换子程序后执行触发调用的程序段。 此时将不再执行被替换的指令。

若替换子程序在程序段末尾调用，那么系统会先执行引起替换子程序调用的程序段，而不执行待替换的指令。 之后系统会调用替换子程序。

3.17.3.2 M40 - M45（齿轮档切换）的替换

功能

在耦合生效时，通过调用用户专用子程序替换引导主轴的齿轮档切换指令（M40，M41 ... M45）。

参数设置

激活

- MD30465 $MA_AXIS_LANG_SUB_MASK，位 0 = 1

子程序的调用时间点

- M40
  无法设置调用时间点。 替换子程序总是在程序段开始处调用。
- M41 ... M45
  调用时间点取决于所配置的辅助功能向 PLC 的输出特性（参见下面的 MD22080）：
  – 运行**前**或运行**期间**输出：在**程序段开始处**调用子程序。
  – 运行**后**输出：在**程序段末尾**调用子程序
  MD22080 $MC_AUXFU_PREDEF_SPEC[12 ... 16]（M41 ... M45 的输出特性）

位	值	含义
5	1	辅助功能在运行**前**输出
6	1	辅助功能在运行**期间**输出
7	1	辅助功能在运行**后**输出

用于信息传输的系统变量

程序段中编写的功能的所有相关信息均通过系统变量（参见“系统变量 (页 273)”章节）提供给替换子程序。 这些数据只基于编写了待替换功能的程序段。

3.17.3.3 SPOS、SPOSA、M19（主轴定位）的替换

功能

在耦合生效时，通过调用用户专用子程序（替换子程序）替换引导主轴的定位指令（SPOS、SPOSA 或 M19）。

应用示例

在双主轴机床上对工件进行并行加工时，两根主轴以不为 1 的系数耦合。 为了执行换刀，必须将其定位至相同的位置。 为此，替换子程序将关闭耦合并将主轴分别定位至换刀位置，之后再重新激活耦合。

参数设置

激活

- MD30465 $MA_AXIS_LANG_SUB_MASK，位 1 = 1

替换子程序的调用时间点

- SPOS，SPOSA
  无法设置调用时间点。 替换子程序总是在程序段开始处调用。
- M19
  调用时间点取决于所配置的辅助功能向 PLC 的输出特性（参见下面的 MD22080）：
  - 运行**前**或运行**期间**输出：在**程序段开始处**调用子程序。
  - 运行**后**输出：在**程序段末尾**调用子程序

MD22080 $MC_AUXFU_PREDEF_SPEC[9]		
位	**值**	**含义**
5	1	辅助功能在运行**前**输出
6	1	辅助功能在运行**期间**输出
7	1	辅助功能在运行**后**输出

用于信息传输的系统变量

程序段中编写的功能的所有相关信息均通过系统变量（参见“系统变量 (页 273)”章节）提供给替换子程序。 这些数据只基于编写了待替换功能的程序段。

3.17.3.4 系统变量

系统变量	**含义**
$P_SUB_AXFCT	M40、M41 ... M45 的替换生效时为 TRUE
$P_SUB_GEAR	编写的或计算出的齿轮档 在替换子程序以外使用：主主轴的齿轮档
$P_SUB_AUTOGEAR	触发替换的程序段中 M40 生效时为 TRUE。 在替换子程序以外使用：解释器中的当前设置
$P_SUB_LA	提供触发了替换的生效耦合中引导主轴的轴名称。 **提示** 若在替换子程序以外使用此变量，程序执行将终止并发出报警。
$P_SUB_CA	提供触发了替换的生效耦合中跟随主轴的轴名称。 **提示** 若在替换子程序以外调用此变量，程序执行将终止并发出报警。
$P_SUB_AXFCT	提供生效的替换方式，对应 MD30465 $MA_AXIS_LANG_SUB_MASK

系统变量	含义	
$P_SUB_SPOS	SPOS 替换生效时为 TRUE	
$P_SUB_SPOSA	SPOSA 替换生效时为 TRUE	
$P_SUB_M19	M19 替换生效时为 TRUE	
$P_SUB_SPOSIT	提供编写的主轴位置 **提示** 若在替换子程序以外调用此变量，程序执行将终止并发出报警。	
$P_SUB_SPOSMODE	提供针对编写的主轴位置的逼近模式：	
	值	**含义**
	0	位置逼近模式无变化
	1	AC
	2	IC
	3	DC
	4	ACP
	5	ACN
	6	OC
	7	PC
	提示 若在替换子程序以外调用此变量，程序执行将终止并发出报警。	
$P_SUB_STAT	替换子程序的调用时间点，以程序段为基准	

3.17.3.5 示例：齿轮档切换

在子程序中替换所有齿轮档切换指令 M40、M41 ... M45。

参数设置

机床数据	含义
MD15700 $MN_LANG_SUB_NAME = "LANG_SUB"	子程序
MD15702 $MN_LANG_SUB_PATH = 0	制造商目录
MD22080 $MC_AUXFU_PREDEF_SPEC[12] = 'H21'	M41：运行前输出
MD22080 $MC_AUXFU_PREDEF_SPEC[13] = 'H21'	M42：运行前输出
MD22080 $MC_AUXFU_PREDEF_SPEC[13] = 'H21'	M43：运行前输出
MD22080 $MC_AUXFU_PREDEF_SPEC[15] = 'H21'	M44：运行前输出

MD22080 $MC_AUXFU_PREDEF_SPEC[16] = 'H21'　M45：运行前输出
MD30465 $MA_AXIS_LANG_SUB_MASK[AX5] = 'H0001'　替换齿轮档切换指令

主程序

编程	注释
PROC MAIN	
N110 COUPON(S2,S1)	; 激活同步主轴耦合
N120 G01 F100 X100 S5000 M3 M43	; 基于 M43 调用子程序
N130 M40	; 激活自动齿轮档切换
N140 M3 S1000	; 基于 S1000 以及
	; 因此触发的自动齿轮档切换
	; 调用子程序
N9999 M30	

替换子程序"LANG_SUB"，方案 1

通过直接定址简化编程和优化速度（S1：引导主轴，S2：跟随主轴）。

编程	注释
N1000 PROC LANG_SUB DISPLOF SBLOF	
N1100 IF($P_SUB_AXFCT ==1)	; 齿轮档切换触发替换
N1140 DELAYFSTON	; 开始停止延时段
N1150 COUPOF(S2,S1)	; 取消同步主轴耦合
N1160 ; 引导主轴和跟随主轴分别切换齿轮档	
N1170 M1=$P_SUB_GEAR M2=$P_SUB_GEAR	
N1180 DELAYFSTON	; 结束停止延时段
N1190 COUPON(S2,S1)	; 激活同步主轴耦合
N1200 ENDIF	
...	
N9999 RET	

替换子程序"LANG_SUB"，方案 2

通过系统变量间接定址，提升灵活度（引导主轴：$P_SUB_LA，跟随主轴：$P_SUB_CA）。

编程	注释
N1000 PROC LANG_SUB DISPLOF SBLOF	
N1010 DEF AXIS _LA	; 引导轴/主轴的标志位
N1020 DEF AXIS _CA	; 跟随轴/主轴的标志位
N1030 DEF INT _GEAR	; 齿轮档的标志位

编程	注释
N1100 IF($P_SUB_AXFCT==1)	; 齿轮档切换触发替换
N1110 _GEAR=$P_SUB_GEAR	; 待激活的齿轮档
N1120 _LA=$P_SUB_LA	; 引导主轴的轴名称
N1130 _CA=$P_SUB_CA	; 跟随主轴的轴名称
N1140 DELAYFSTON	; 开始停止延时段
N1150 COUPOF(_CA,_LA)	; 取消同步主轴耦合
N1160 ; 引导主轴和跟随主轴切换齿轮档	
N1170 M[AXTOSPI(_LA)]=_GEAR M[AXTOSPI(_CA)]=_GEAR	
N1180 DELAYFSTOF	; 结束停止延时段
N1190 COUPON(_CA,_LA)	; 激活同步主轴耦合
N1200 ENDIF	
...	
N9999 RET	

3.17.3.6 示例：主轴定位

在子程序中只显性执行 SPOS 和 SPOSA 指令的替换。其他替换须相应添加。

参数设置

机床数据	含义
MD30465 $MA_AXIS_LANG_SUB_MASK[AX5] = 'H0002'	替换定位指令
MD22080 $MC_AUXFU_PREDEF_SPEC[9] = 'H0021'	M19 在运行前输出至 PLC

设定数据	含义
SD43240 $SA_M19_SPOS[AX5] = 260	M19 下的主轴位置 = 260
SD43250 $SA_M19_SPOSMODE[AX5] = 4	M19 下的位置逼近模式：“正向趋近（ACP）”

主程序

编程	注释
PROC MAIN	
...	
N210 COUPON(S2,S1)	; 激活同步主轴耦合
N220 SPOS[1]=100	; 用 SPOS 定位引导主轴

编程	注释
...	
N310 G01 F1000 X100 M19	; 用 M19 定位引导主轴

替换子程序"LANG_SUB"，方案 1

通过直接定址简化编程和优化速度（S1：引导主轴，S2：跟随主轴）。

编程	注释
N1000 PROC LANG_SUB DISPLOF SBLOF	
N2100 IF($P_SUB_AXFCT==2)	
N2110 ; 在同步主轴耦合生效时替换 SPOS/SPOSA/M19	
N2185 DELAYFSTON	; 开始停止延时段
N2190 COUPOF(S2,S1)	; 取消同步主轴耦合
N2200	; 定位引导主轴和跟随主轴
N2210 IF($P_SUB_SPOS==TRUE) OR ($P_SUB_SPOSA==TRUE)	
N2220 ; SPOS 和 SPOSA 映射至 SPOS	
N2230 CASE $P_SUB_SPOSMODE OF \	
0 GOTOF LABEL1_DC \	
1 GOTOF LABEL1_IC \	
2 GOTOF LABEL1_AC \	
3 GOTOF LABEL1_DC \	
4 GOTOF LABEL1_ACP \	
5 GOTOF LABEL1_ACN \	
DEFAULT GOTOF LABEL_ERR	
LABEL1_DC:SPOS[1]=DC($P_SUB_SPOSIT) SPOS[2]=DC($P_SUB_SPOSIT)	
GOTOF LABEL1_CONT	
LABEL1_IC:DELAYFSTOF	
SPOS[1]=IC($P_SUB_SPOSIT) SPOS[2]=IC($P_SUB_SPOSIT)	
DELAYFSTON	
GOTOF LABEL1_CONT	
LABEL1_AC:SPOS[1]=AC($P_SUB_SPOSIT) SPOS[2]=AC($P_SUB_SPOSIT)	
GOTOF LABEL1_CONT	
LABEL1_ACP:SPOS[1]=ACP($P_SUB_SPOSIT) SPOS[2]=ACP($P_SUB_SPOSIT)	
GOTOF LABEL1_CONT	
LABEL1_ACN:SPOS[1]=ACN($P_SUB_SPOSIT) SPOS[2]=ACN($P_SUB_SPOSIT)	
LABEL1_CONT:	
N2250 ELSE	; 用 M19 定位主轴
N2270 M1=19 M2=19	; 引导主轴和跟随主轴
N2280 ENDIF	; SPOS、SPOSA 替换结束
N2285 DELAYFSTOF	; 结束停止延时段
N2290 COUPON(S2,S1)	; 激活同步主轴耦合
N2410 ELSE	
N2420 ; 自此开始进行其他替换	

编程	注释
...	
N3300 ENDIF	; 替换结束
...	
N9999 RET	; 一般程序结束
LABEL_ERR:SETAL(61000)	; 出错

替换子程序"LANG_SUB"，方案 2

通过系统变量间接定址，提升灵活度（引导主轴：$P_SUB_LA，跟随主轴：$P_SUB_CA）。

编程	注释
N1000 PROC LANG_SUB DISPLOF SBLOF	
N1010 DEF AXIS _LA	; 引导轴/主轴
N1020 DEF AXIS _CA	; 跟随轴/主轴
N1030 DEF INT _LSPI	; 引导主轴编号（编写的
	; 主轴）
N1040 DEF INT _CSPI	; 跟随主轴编号
...	
N2100 IF($P_SUB_AXFCT==2)	
N2110 ; 在同步主轴耦合生效时替换 SPOS/SPOSA/M19	
N2120 _LA=$P_SUB_LA	; 引导主轴的轴名称
N2130 _CA=$P_SUB_CA	; 跟随主轴的轴名称
N2140 _LSPI=AXTOSPI(_LA)	; 引导主轴的编号
N2180 _CSPI=AXTOSPI(_LA)	; 跟随主轴的编号
N2185 DELAYFSTON	; 开始停止延时段
N2190 COUPOF(_CA,_LA)	; 取消同步主轴耦合
N2200	; 定位引导主轴和跟随主轴：
N2210 IF($P_SUB_SPOS==TRUE) OR ($P_SUB_SPOSA==TRUE)	
N2220 ; SPOS 和 SPOSA 映射至 SPOS	
N2230 CASE $P_SUB_SPOSMODE OF	
0 GOTOF LABEL1_DC \	
1 GOTOF LABEL1_IC \	
2 GOTOF LABEL1_AC \	
3 GOTOF LABEL1_DC \	
4 GOTOF LABEL1_ACP \	
5 GOTOF LABEL1_ACN \	
DEFAULT GOTOF LABEL_ERR	
LABEL1_DC:SPOS[_LSPI]=DC($P_SUB_SPOSIT) SPOS[_CSPI]=DC($P_SUB_SPOSIT)	
GOTOF LABEL1_CONT	
LABEL1_IC:DELAYFSTOF	
SPOS[_LSPI]=IC($P_SUB_SPOSIT) SPOS[_CSPI]=IC($P_SUB_SPOSIT)	
DELAYFSTON	
GOTOF LABEL1_CONT	

编程	注释
LABEL1_AC:SPOS[_LSPI]=AC($P_SUB_SPOSIT) SPOS[_CSPI]=AC($P_SUB_SPOSIT)	
GOTOF LABEL1_CONT	
LABEL1_ACP:SPOS[_LSPI]=ACP($P_SUB_SPOSIT) POS[_CSPI]=ACP($P_SUB_SPOSIT)	
GOTOF LABEL1_CONT	
LABEL1_ACN:SPOS[_LSPI]=ACN($P_SUB_SPOSIT) POS[_CSPI]=ACN($P_SUB_SPOSIT)	
LABEL1_CONT:	
N2250 ELSE	; 用 M19 定位主轴
N2270 M[_LSPI]=19 M[_CSPI]=19	
N2280 ENDIF	
N2285 DELAYFSTOF	; 结束停止延时段
N2290 COUPON(_CA,_LA)	; 激活同步主轴耦合
N2410 ELSE	
N2420 ; 自此开始进行其他替换	
...	
N3300 ENDIF	
...	
N9999 RET	; 一般程序结束
LABEL_ERR:SETAL(61000)	; 出错

3.17.4 子程序的属性

一般规则

- 替换中调用的子程序可包含 PROC 指令，以及 SBLOF 和 DISPLOF 属性。
- 替换也可在 ISO 语言模式下进行。但是替换子程序只在标准语言模式（西门子）下执行。此时系统会隐性切换至标准语言模式。从替换子程序返回时，系统将重新切换回原先的语言模式。
- 对替换子程序的信息传输只通过系统变量进行。无法使用传输参数。
- 单程序段和 SBLOF 属性下的特性取决于以下机床数据中的设置：
  MD10702 IGNORE_SINGLEBLOCK_MASK，位 14（忽略单程序段停止）

值	含义
0	替换子程序的特性与“一般”子程序相同： • 通过 M17 返回：在子程序末尾停止 **提示** M 功能向 PLC 的输出取决于以下机床数据： MD20800 $MC_SPF_END_TO_VDI，位 0（子程序结束输出至 PLC） - 位 0 = 0：不输出 - 位 0 = 1：M17 输出至 PLC。 • 通过 RET 跳回：在替换子程序末尾处不停止
1	在调用替换子程序的程序段中**只停止一次**。并且这与以下因素无关： • 子程序是否是在程序段开始处和/或程序段末尾调用 • 子程序中是否调用了其他子程序 • 子程序是否通过 M17 或 RET 退出 对于 M 功能的替换，单程序段停止在替换子程序末尾处进行。 对于 T 和 D/DL 功能的替换，单程序段停止的时间点取决于子程序的调用时间点： • 在程序段开始处调用：在程序段末尾进行单程序段停止 • 在程序段末尾处调用：在替换子程序末尾进行单程序段停止

- 对于带 DISPLOF 属性的替换子程序，程序段显示中会显示引起子程序调用的程序行作为当前程序段。
- 在替换子程序中可使用 DELAYFSTON 和 DELAYFSTOF 指令指定区域或整个替换子程序，防止其受到中断（例如 NC 停止、读取禁止等）。
- 替换不会递归进行。也就是说，对于引起替换子程序调用的功能，若在替换子程序中再次编写，则其不再会被执行。

辅助功能向 PLC 的输出

替换辅助功能时，调用替换子程序不会引起辅助功能向 PLC 的输出。仅当在替换子程序中重新编写了辅助功能时，其才会被输出。

程序段搜索时的特性

替换子程序也可在“进行计算的程序段搜索”和“程序测试模式下进行计算的程序段搜索（SERUPRO）”这些程序段搜索模式下调用。可能存在的特殊性必须在替换子程序中使用系统变量 $P_SEARCH 和 $AC_SERUPRO 实现。

在“进行计算的程序段搜索”下的动作收集方面，替换子程序的特性与“一般”子程序相似。

3.17.5 前提条件

- 以下情形下不允许进行功能替换：
  - 同步动作
  - 工艺循环
- 若程序段开始处包含待替换的功能，那么该程序段之前不允许为逐段生效的同步动作。 参见下面的“示例：逐段生效的同步动作”部分。
- 在替换子程序中只允许执行相应替换所需的动作。
- 对于在程序段末尾调用替换子程序的程序段，须注意以下事项：
  - 不允许有模态生效的子程序调用生效。
  - 不允许编写子程序返回
  - 不允许编写程序结束

注意
控制系统不会监控替换子程序中是否实现了待替换功能。

示例：逐段生效的同步动作

MD30465 $MA_AXIS_LANG_SUB_MASK，位 0 = 1（齿轮级切换）

程序代码

```
...
N1000 WHENEVER $AA_IM[X2] <= $AA_IM[X1] + 0.5 DO $AA_OVR[X1]=0
N1010 G1 X100 M43
...
```

若程序段 N1010 中 M43 功能引起替换子程序的调用，那么加工将随之终止并显示报警。

3.18 重命名/禁用 NC 指令

功能

借助“重命名/禁用 NC 指令”功能，机床制造商或用户可以修改现有 NC 指令的名称。

应用

该功能可用于：

- 改善零件程序的可读性
- 禁用 NC 指令
- 扩展 NC 功能

参数设置

通过以下机床数据重命名/禁用 NC 指令：

MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[<n>]（重新配置后的 NC 代码列表）

偶数索引 [<n>] 包含指令的原始名称。奇数索引包含指令的新名称。

空字符串 ("") 表示没有给定指令名称。指令因此被禁用且无法再进行编程。

对 MD10712 的修改在控制器启动后生效。

示例

重命名指令 G00 并禁用指令 G01

MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[0] = "G00"

MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[1] = "EILGANG"

MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[2] = "G01"

MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[3] = ""

通过重新上电生效后，无法再通过指令 `G00`，而是通过指令 `EILGANG` 编程快速运行。`G01` 无法再进行编程。

3.19 程序运行时间/工件计数器

为了对机床操作人员提供支持，系统提供程序运行时间和工件数量的相关信息。

这些信息可以作为系统变量在 NC 和/或 PLC 程序中处理。同时这些信息也可在操作界面上显示。

3.19.1 程序运行时间

3.19.1.1 功能

功能“程序运行时间”提供了用于监控工艺过程，它可以通过系统变量在零件程序和同步动作中读取。有两种不同类型的计时器：

1. 标准计时器
   标准计时器始终生效。
2. 可激活的计时器
   可激活的计时器必须通过机床数据激活。

标准计时器

从上一次控制系统启动到现在的时间

系统变量	含义
$AN_SETUP_TIME	从上一次使用缺省值启动控制系统到现在的时间，单位分 在每次使用缺省值启动控制系统时，该变量都将自动复位为“0”。
$AN_POWERON_TIME	从上一次控制系统普通启动（热启动）到现在的时间，单位 min 在每次控制系统普通启动时，该变量都将自动复位为“0”。

程序运行时间

用于测量程序运行时间的计时器仅在 AUTO 模式下可用。

通道专用系统变量	说明
$AC_ACT_PROG_NET_TIME	当前程序的当前净运行时间，单位 s 净运行时间表示：程序停止所花费的时间已被减去。 若在 AUTO 运行方式下从 RESET 通道状态重新启动一个零件程序，那么 $AC_ACT_PROG_NET_TIME 会自动复位为“0”。 其他属性： • 复位键不会使 $AC_ACT_PROG_NET_TIME 复位为“0”，而只是使计时器停止。 • 启动 ASUB 时，系统会将 $AC_ACT_PROG_NET_TIME 设置为“0”，并对 ASUB 的运行时间进行计数。Am Ende eines ASUPs verhält es sich wie bei der RESET-Taste:der Timer wird nur angehalten, aber nicht auf "0" gesetzt. • 在启动事件控制的程序（PROG_EVENT）时，$AC_ACT_PROG_NET_TIME **不会**被复位。 仅当该程序为启动/M30/搜索 PROG_EVENT 时，程序运行时间才继续计数。 • GOTOS 和倍率 = 0% 条件下 $AC_ACT_PROG_NET_TIME 的特性可通过 MD27850 设置（参见“参数设置”部分） **提示：** 可通过 $AC_PROG_NET_TIME_TRIGGER 继续操控 $AC_ACT_PROG_NET_TIME。
$AC_OLD_PROG_NET_TIME	刚正确结束程序的净运行时间，单位 s “正确结束”表示程序不是通过 RESET 终止，而是通过常规的 M30 结束。 启动新程序时，$AC_OLD_PROG_NET_TIME 保持不变，直至重新到达 M30。 其他属性： • 当前选择的程序被编辑时，$AC_OLD_PROG_NET_TIME 会被设置为“0”。 • 在 ASUB 或事件控制程序（PROG_EVENT）的末尾 $AC_OLD_PROG_NET_TIME 不会改变。 **提示：** 仅当 $AC_PROG_NET_TIME_TRIGGER 未赋值时，才会发生从 $AC_ACT_PROG_NET_TIME 向 $AC_OLD_PROG_NET_TIME 的隐性复制。

<table>
<tr><th>通道专用系统变量</th><th colspan="2">说明</th></tr>
<tr><td>$AC_OLD_PROG_NET_TIME_COUNT</td><td colspan="2">$AC_OLD_PROG_NET_TIME 的变化
上电后 $AC_OLD_PROG_NET_TIME_COUNT 置“0”。
当控制系统重新写入 $AC_OLD_PROG_NET_TIME 时，$AC_OLD_PROG_NET_TIME_COUNT 总是随之提升。
用户通过 RESET 终止运行中的程序时，$AC_OLD_PROG_NET_TIME 和 $AC_OLD_PROG_NET_TIME_COUNT 保持不变。
因此，通过 $AC_OLD_PROG_NET_TIME_COUNT 可确定 $AC_OLD_PROG_NET_TIME 是否已被写入。
示例：
若两个连续运行的程序运行时间相同并已正确结束，那么用户可通过 $AC_OLD_PROG_NET_TIME_COUNT 中改变的值加以识别。</td></tr>
<tr><td rowspan="6">$AC_PROG_NET_TIME_TRIGGER</td><td colspan="2">用于选择性测量程序段的触发器
通过将值写入变量触发功能：</td></tr>
<tr><td>值</td><td>功能</td></tr>
<tr><td>0</td><td>无效</td></tr>
<tr><td>1</td><td>结束测量
• $AC_OLD_PROG_NET_TIME = $AC_ACT_PROG_NET_TIME
• $AC_ACT_PROG_NET_TIME = 0，随后继续运行</td></tr>
<tr><td>2</td><td>启动测量
• $AC_ACT_PROG_NET_TIME = 0
• $AC_OLD_PROG_NET_TIME 未改变</td></tr>
<tr><td>3</td><td>停止测量
• $AC_OLD_PROG_NET_TIME 未改变
• $AC_OLD_PROG_NET_TIME 未改变</td></tr>
<tr><td>4</td><td>继续停止的测量
• $AC_ACT_PROG_NET_TIME 继续运行
• $AC_OLD_PROG_NET_TIME 未改变</td></tr>
<tr><td colspan="3">上电时所有系统变量将复位为“0”！</td></tr>
</table>

说明

工件的剩余时间

如果需要依次加工相同的工件，可通过下列计时器值计算该工件的剩余时间：

- 上一个工件的加工时间（参见 $AC_OLD_PROG_NET_TIME）
- 当前的加工时间（参见 $AC_ACT_PROG_NET_TIME）

除当前加工时间外，剩余时间也会在操作界面上显示。

说明

使用 STOPRE

系统变量 $AC_OLD_PROG_NET_TIME 和 $AC_OLD_PROG_NET_TIME_COUNT 不会产生隐性的预处理停止。在零件程序中使用时，若系统变量的值来自于之前的程序运行，那么这一点影响不大。但若高频率地写入运行时间测量触发器（$AC_PROG_NET_TIME_TRIGGER），并由此导致 $AC_OLD_PROG_NET_TIME 频繁变更，则需在零件程序中写入一个显性的 STOPRE。

可激活的计时器

程序运行时间

用于测量程序运行时间的计时器仅在 AUTO 模式下可用。

通道专用系统变量	说明
$AC_OPERATING_TIME	AUTO 运行方式下 NC 程序的总运行时间，单位 s AUTO 运行方式下，NC 启动和程序结束/NC 复位间所有程序的运行时间的累加值。 缺省条件下，NC 停止和倍率 = 0% 时此变量不进行计数。可通过 MD27860 激活 0% 倍率下的继续计数。 在每次控制系统启动时，该变量都将自动复位为“0”。
$AC_CYCLE_TIME	所选择的 NC 程序的运行时间，单位 s 测量所选 NC 程序中 NC 启动和程序结束/NC 复位间的运行时间。 缺省条件下，NC 停止和倍率 = 0% 时此变量不进行计数。可通过 MD27860 激活 0% 倍率下的继续计数。 在每次启动新的 NC 程序时，该值都将自动复位为“0”。可通过机床数据 MD27860 设置，是否在使用 GOTOS 跳转至程序头或启动 ASUB 和 PROG_EVENT 时进行复位。
$AC_CUTTING_TIME	加工时间，单位 s 测量所有 NC 程序中 NC 启动和程序结束/NC 复位间无快进生效的轨迹轴（其中至少一根轴生效）的运行时间。可通过 MD27860 设置，是只通过生效的刀具进行测量，还是进行与刀具无关的测量。 此外测量会在暂停时间生效时中断。 在每次使用缺省值启动控制系统时该值都将自动复位为“0”。

3.19.1.2 调试

激活/取消激活

可激活的计时器可通过以下机床数据激活/取消：

MD27860 $MC_PROCESSTIMER_MODE，位 0 - 2 = <值>

位	值	含义
0	0	$AC_OPERATING_TIME 的计时器不生效。
	1	$AC_OPERATING_TIME 的计时器生效。

位	值	含义
1	0	$AC_CYCLE_TIME 的计时器不生效。
	1	$AC_CYCLE_TIME 的计时器生效。
2	0	$AC_CUTTING_TIME 的计时器不生效。
	1	$AC_CUTTING_TIME 的计时器生效。

参数设置

恒定生效的计时器的特性

恒定生效的计时器在 GOTOS 和倍率 = 0% 情形下的特性可通过以下机床数据设置：

MD27850 $MC_PROG_NET_TIMER_MODE

位	值	含义
0	0	通过 GOTOS 跳转至程序头时，$AC_ACT_PROG_NET_TIME 不复位为“0”（缺省设置）。
	1	通过 GOTOS 跳转至程序头时，$AC_ACT_PROG_NET_TIME 复位为“0”，$AC_OLD_PROG_NET_TIME 中之前的值被保存，程序计数器 $AC_OLD_PROG_NET_TIME_COUNT 提升。
1	0	在倍率 = 0% 时 $AC_ACT_PROG_NET_TIME 不提升。亦即，在倍率设置为“0”的时间内不计算程序运行时间（缺省设置）。
	1	倍率 = 0% 时 $AC_ACT_PROG_NET_TIME 同样提升。亦即，在倍率设置为“0”的时间内仍计算程序运行时间。

可激活计时器的特性

特定功能（例如空运行进给率、程序测试）下可激活计时器的特性通过以下机床数据设置：

MD27860 $MC_PROCESSTIMER_MODE

位	值	含义
4	0	空运行进给率生效时不进行测量。
	1	空运行进给率生效时仍进行测量。
5	0	程序测试模式下不进行测量。
	1	程序测试模式下仍进行测量。
6	仅针对位 1 = 1（$AC_CYCLE_TIME 的计时器生效）	
	0	通过 ASUB 和 PROG_EVENT 启动时，$AC_CYCLE_TIME 也复位为“0”。
	1	通过 ASUB 和 PROG_EVENT 启动时，$AC_CYCLE_TIME 不复位为“0”。

位	值	含义
7	仅针对位 2 = 1（$AC_CUTTING_TIME 的计时器生效）	
	0	$AC_CUTTING_TIME 的计时器仅在刀具生效时计数。
	1	$AC_CUTTING_TIME 的计时器计数与刀具无关。
8	仅针对位 1 = 1（$AC_CYCLE_TIME 的计时器生效）	
	0	通过 GOTOS 跳转至程序头时，$AC_CYCLE_TIME 不复位为"0"（缺省设置）。
	1	通过 GOTOS 跳转至程序头时，$AC_CYCLE_TIME 复位为"0"。
9	仅针对位 0，1 = 1（$AC_OPERATING_TIME 和 $AC_CYCLE_TIME 的计时器生效）	
	0	倍率 = 0% 时，程序运行时间不继续计数
	1	倍率 = 0% 时，程序运行时间继续计数

3.19.1.3 前提条件

程序段搜索

在程序段搜索时不会计算程序运行时间。

REPOS

REPOS 过程的时间会计入当前的加工时间（$AC_ACT_PROG_NET_TIME）。

3.19.1.4 示例

示例 1：Parametrierung der Laufzeitmessung über MD27860

为生效的 NC 程序激活运行时间测量，同时在试运行进给和程序测试生效时不进行测量：

MD27860 $MC_PROCESSTIMER_MODE = 'H2'

激活对刀具动作时间的测量，在试运行进给和程序测试生效时同样进行测量：

MD27860 $MC_PROCESSTIMER_MODE = 'H34'

激活刀具生效时对总运行时间和加工时间的测量，在程序测试模式下同样进行测量：

MD27860 $MC_PROCESSTIMER_MODE = 'H25'

激活对总运行时间和加工时间的测量（与刀具无关），在程序测试模式下同样进行测量：

MD27860 $MC_PROCESSTIMER_MODE = 'Ha5'

激活刀具生效时对加工时间的测量，在倍率 = 0% 时同样进行测量，但在空运行进给率生效时不进行测量：

MD27860 $MC_PROCESSTIMER_MODE = 'H22'

示例 2：测量“mySubProgrammA”的时间

程序代码

```
...
N50 DO $AC_PROG_NET_TIME_TRIGGER=2
N60 FOR ii= 0 TO 300
N70 mySubProgrammA
N80 DO $AC_PROG_NET_TIME_TRIGGER=1
N95 ENDFOR
N97 mySubProgrammB
N98 M30
```

程序处理 N80 行后，$AC_OLD_PROG_NET_TIME 中为“mySubProgrammA”的净运行时间。

$AC_OLD_PROG_NET_TIME 的值：

- 在 M30 后保持不变
- 在每次完整运行循环后更新

示例 3：测量“mySubProgrammA”和“mySubProgrammC”的时间

程序代码

```
N10 DO $AC_PROG_NET_TIME_TRIGGER=2
N20 mySubProgrammA
N30 DO $AC_PROG_NET_TIME_TRIGGER=3
N40 mySubProgrammB
N50 DO $AC_PROG_NET_TIME_TRIGGER=4
N60 mySubProgrammC
N70 DO $AC_PROG_NET_TIME_TRIGGER=1
N80 mySubProgrammD
N90 M30
```

3.19.2 工件计数器

3.19.2.1 功能

“工件计数器”功能以通道专用系统变量的形式提供多个值域为 0 至 999.999.999 的计数器。这些系统变量均可读写。

通过下列通道专用机床数据可以对计数器激活、归零时刻和计数算法进行调整。

用于工件计数的系统变量

系统变量	含义
$AC_REQUIRED_PARTS	待加工工件的数量（设定工件数量） 在此计数器中可以定义工件的个数，在到达这个数值之后，实际工件数（$AC_ACTUAL_PARTS）复位为“0”。 通过 MD27880 可生成显示报警： “通道 %1 工件设定数量 = %2 已达到” 和通道专用 NC/PLC 接口信号： <Chan>.basic.in.productionTargetReached（达到工件设定数量）。
$AC_TOTAL_PARTS	已加工工件的总数（实际工件总数） 该计数器给出自开始时刻起所加工的工件数量。只有在使用缺省值启动控制系统时该值才会自动复位为“0”。
$AC_ACTUAL_PARTS	已加工工件的数量（实际工件数） 此计数器会记录自开始时刻起加工的所有工件的数量。当达到设定工件数量（$AC_REQUIRED_PARTS）时，该计数器会自动归“0”（前提条件是 $AC_REQUIRED_PARTS > 0）。
$AC_SPECIAL_PARTS	用户计数的工件数量 该计数器可协助用户进行自定义的工件计数。可定义在到达设定工件数（$AC_REQUIRED_PARTS）时输出报警。用户必须自行将该计数器归零。

说明

在以缺省值启动控制系统时，所有的工件计数器都会归“0”，而且不管是否激活，都可以被读写。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.productionTargetReached	LBP_typeChanX.E_WS_Setp	DB21, ... DBX317.1

3.19.2.2 功能

“工件计数器”功能以通道专用系统变量的形式提供多个值域为 0 至 999.999.999 的计数器。这些系统变量均可读写。

通过下列通道专用机床数据可以对计数器激活、归零时刻和计数算法进行调整。

用于工件计数的系统变量

系统变量	含义
$AC_REQUIRED_PARTS	待加工工件的数量（设定工件数量） 在此计数器中可以定义工件的个数，在到达这个数值之后，实际工件数（$AC_ACTUAL_PARTS）复位为“0”。 通过 MD27880 可生成显示报警： “通道 %1 工件设定数量 = %2 已达到” 和通道专用 NC/PLC 接口信号： DB3300.DBX4001.1（达到工件设定数量）。
$AC_TOTAL_PARTS	已加工工件的总数（实际工件总数） 该计数器给出自开始时刻起所加工的工件数量。只有在使用缺省值启动控制系统时该值才会自动复位为“0”。
$AC_ACTUAL_PARTS	已加工工件的数量（实际工件数） 此计数器会记录自开始时刻起加工的所有工件的数量。当达到设定工件数量（$AC_REQUIRED_PARTS）时，该计数器会自动归“0”（前提条件是 $AC_REQUIRED_PARTS > 0）。
$AC_SPECIAL_PARTS	用户计数的工件数量 该计数器可协助用户进行自定义的工件计数。可定义在到达设定工件数（$AC_REQUIRED_PARTS）时输出报警。用户必须自行将该计数器归零。

说明

在以缺省值启动控制系统时，所有的工件计数器都会归"0"，而且不管是否激活，都可以被读写。

3.19.2.3 调试

激活

工件计数器通过以下机床数据激活：

MD27880 $MC_PART_COUNTER（激活工件计数器）

位	值	含义
0	1	$AC_REQUIRED_PARTS 生效。
1	0	以下情形时输出报警/信号：$AC_ACTUAL_PARTS = $AC_REQUIRED_PARTS
	1	以下情形时输出报警/信号：$AC_SPECIAL_PARTS = $AC_REQUIRED_PARTS
4	1	$AC_TOTAL_PARTS 生效。
5	0	M02 / M30 使 $AC_TOTAL_PARTS 的计数提升"1"。
	1	通过 MD27882[**0**] 定义的 M 指令使 $AC_TOTAL_PARTS 的计数提升"1"。
6	0	$AC_TOTAL_PARTS 在程序测试/程序段搜索时同样生效。
7	1	通过 GOTOS 返回时，$AC_TOTAL_PARTS 的计数提升"1"。
8	1	$AC_ACTUAL_PARTS 生效。
9	0	M02 / M30 使 $AC_ACTUAL_PARTS 的计数提升"1"。
	1	通过 MD27882[**1**] 定义的 M 指令使 $AC_ACTUAL_PARTS 的计数提升"1"。
10	0	$AC_ACTUAL_PARTS 在程序测试/程序段搜索时同样生效。
11	1	通过 GOTOS 返回时，$AC_ACTUAL_PARTS 的计数提升"1"。
12	1	$AC_SPECIAL_PARTS 生效。
13	0	M02 / M30 使 $AC_SPECIAL_PARTS 的计数提升"1"。
	1	通过 MD27882[**2**] 定义的 M 指令使 $AC_SPECIAL_PARTS 的计数提升"1"。
14	0	$AC_SPECIAL_PARTS 在程序测试/程序段搜索时同样生效。
15	1	通过 GOTOS 返回时，$AC_SPECIAL_PARTS 的计数提升"1"。

参数设置

通过用户定义的 M 指令触发工件计数

若 MD27880 中设置了相应位，可通过以下机床数据设置的 M 指令代替程序结束 M2/M30 来触发计数脉冲。

MD27882 $MC_PART_COUNTER_MCODE[<n>]（过用户定义的 M 指令触发工件计数）

<n>	含义
0	MD27882[**0**] 用于定义使 $AC_**TOTAL**_PARTS 计数提升的 M 指令。
1	MD27882[**1**] 用于定义使 $AC_**ACTUAL**_PARTS 计数提升的 M 指令。
2	MD27882[**2**] 用于定义使 $AC_**SPECIAL**_PARTS 计数提升的 M 指令。

在调用用户定义的 M 指令时，相关工件计数器的数值会提升“1”。

工件计数的保护等级

借助操作界面，通过以下机床数据为工件计数的激活/关闭设置保护等级：

MD51074 $MN_ACCESS_WRITE_WPC_COUNTER = <保护等级>

3.19.2.4 前提条件

运行方式切换/NC-RESET

运行方式切换和 NC-RESET 不会影响工件计数器。

$AC_REQUIRED_PARTS ≤ 0

$AC_REQUIRED_PARTS ≤ 0 且 MD27880 $MC_PART_COUNTER 位 0 == 1 时，所有生效的计数器**不**执行计数及通过 MD27880 设置的识别校验。

3.19.2.5 示例

激活工件计数器 $AC_REQUIRED_PARTS：

- MD27880 $MC_PART_COUNTER = 'H3'

$AC_REQUIRED_PARTS 生效。

以下情形时显示报警：$AC_REQUIRED_PARTS == $AC_SPECIAL_PARTS

激活工件计数器 $AC_TOTAL_PARTS:

- MD27880 $MC_PART_COUNTER = 'H10'
- MD27882 $MC_PART_COUNTER_MCODE[0] = 80

$AC_TOTAL_PARTS 生效，每个 M02 会使计数提升"1"。

$MC_PART_COUNTER_MCODE[0] 没有意义。

激活工件计数器 $AC_ACTUAL_PARTS：

- MD27880 $MC_PART_COUNTER = 'H300'
- MD27882 $MC_PART_COUNTER_MCODE[1] = 17

$AC_TOTAL_PARTS 生效，每个 M17 会使计数提升"1"。

激活工件计数器 $AC_SPECIAL_PARTS:

- MD27880 $MC_PART_COUNTER = 'H3000'
- MD27882 $MC_PART_COUNTER_MCODE[2] = 77

$AC_SPECIAL_PARTS 生效。

每个 M77 使 $AC_SPECIAL_PARTS 的计数提升 1

关闭工件计数器 $AC_ACTUAL_PARTS：

- MD27880 $MC_PART_COUNTER = 'H200'
- MD27882 $MC_PART_COUNTER_MCODE[1] = 50

$AC_ACTUAL_PARTS 不生效。

激活所有计数器：

- MD27880 $MC_PART_COUNTER = 'H3313'
- MD27882 $MC_PART_COUNTER_MCODE[0] = 80
- MD27882 $MC_PART_COUNTER_MCODE[1] = 17
- MD27882 $MC_PART_COUNTER_MCODE[2] = 77

$AC_REQUIRED_PARTS 生效。

以下情形时显示报警：$AC_REQUIRED_PARTS == $AC_SPECIAL_PARTS

$AC_TOTAL_PARTS 生效，每个 M02 会使计数提升"1"。

$MC_PART_COUNTER_MCODE[0] 没有意义。

$AC_ACTUAL_PARTS 生效，每个 M17 会使计数提升"1"。

$AC_SPECIAL_PARTS 生效，每个 M77 会使计数提升 1。

程序测试/程序段搜索时，工件计数器 $AC_ACTUAL_PARTS 无效：

- MD27880 $MC_PART_COUNTER = 'H700'
- MD27882 $MC_PART_COUNTER_MCODE[1] = 75

$AC_ACTUAL_PARTS 生效，每个 M75 会使计数提升"1"，程序测试和程序段搜索除外。

位 0 = 1 时取消 MD27880 $MC_PART_COUNTER 中涉及的计数模式：

- MD27882 $MC_PART_COUNTER_MCODE[0] = 41
- MD27882 $MC_PART_COUNTER_MCODE[1] = 42
- MD27882 $MC_PART_COUNTER_MCODE[2] = 43

程序代码	注释
...	
N100 $AC_REQUIRED_PARTS=-10	; 值 < 0：设置计数。
N200 M41 M43	; 不计数。
N300 M42	
...	
N500 $AC_REQUIRED_PARTS=52	; 值 > 0：计数依据 MD27880 生效。
N501 M43	; 计数。
N502 M42 M41	; 计数。
...	

3.20 程序模拟

3.20.1 功能

在程序模拟中，系统会对当前零件程序进行完整的计算，并在操作界面中以图形显示结果。这样一来无需运行机床轴便可检查程序的加工结果，并能提前发现编写错误的加工步骤，从而避免工件上的错误加工。

仿真 NC

仿真功能使用一个独立的 NC 实例（仿真 NC）。因此开始仿真前必须将真实 NC 调整为仿真 NC。调整时会从 NC 读取所有生效的机床数据，并导入仿真 NC。NC 机床数据和循环机床数据均包含在生效的机床数据中。

3.20.2 程序模拟与编译循环的组合使用

程序模拟时支持使用一些编译循环 (CC)。控制系统启动后会对支持的编译循环的机床数据进行一次性的调整。“仿真启动”时则不会进行调整！

说明

在零件程序中**不能**使用系统不支持的CC的专用语言指令及机床数据（参见“零件程序中的CC指令”部分）。

某些情形下，系统支持的 CC 的特殊运动（OEM 转换）会被错误显示。

零件程序中的 CC 指令

将系统不支持的编译循环（OMA1 ... OMA5，OEMIPO1/2，G810 ... G829，独立程序和功能）的语言指令写入零件程序时，若不特别处理，则会触发报警并使仿真终止。

解决方案：

在零件程序中单独处理不支持的 CC 专用语言指令（$P_SIM 查询）。

示例：

程序代码	注释
N1 G01 X200 F500	
IF (1==$P_SIM)	
N5 X300	; 仿真中 CC 不生效。
ELSE	
N5 X300 OMA1=10	
ENDIF	

4 轴、坐标系、框架

4.1 简要说明

4.1.1 轴

机床轴

机床轴指的是在机床上实际存在的轴。

通道轴

每根几何轴和辅助轴总是被指定给一个通道，从而被指定为通道轴。几何轴和辅助轴总是在所归属于的通道中运行。

几何轴

三根几何轴总是构成一个虚拟的直角坐标系，即基本坐标系（BCS）。

通过使用框架（偏移、旋转、比例缩放、镜像）可将工件坐标系（WCS）的几何轴映射至BCS。

辅助轴

与几何轴不同，辅助轴之间未定义几何关联。

轨迹轴

轨迹轴的特点是一同进行插补（一个通道的所有轨迹轴共用一个轨迹插补器）。

一个通道的所有轨迹轴具有共同的加速、恒速运行和减速阶段。

定位轴

定位轴的特点是独立进行插补（每根定位轴均有一个专属的轴插补器）。每根定位轴均具有独立的进给率和加速特性曲线。

同步轴

同步轴和轨迹轴一同进行插补（一个通道的所有轨迹轴和同步轴共用一个轨迹插补器）。

一个通道的所有轨迹轴和同步轴具有共同的加速、恒速运行和减速阶段。

轴配置

有关几何轴、辅助轴、通道轴和机床轴之间的分配以及各个轴类型的名称定义通过下列机床数据进行：

MD20050 $MC_AXCONF_GEOAX_ASIGN_TAB（几何轴指定为通道轴）

MD20060 $MC_AXCONF_GEOAX_NAME_TAB（通道中的几何轴名称）

MD20070 $MC_AXCONF_MACHAX_USED（通道中生效的机床轴编号）

MD20080 $MC_AXCONF_CHANAX_NAME_TAB（通道中的通道轴名称）

MD10000 $MN_AXCONF_MACHAX_NAME_TAB（机床轴名称）

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX（主轴指定为机床轴）

可切换的几何轴

借助“可切换的几何轴”功能，可通过零件程序将其他通道轴组成几何轴组。

通道中先前配置为同步辅助轴的轴也可通过程序指令替换任意几何轴。

4.1.2 坐标系

MCS

机床坐标系（MCS）具有以下属性：

- 由机床轴构成。
- 机床轴可相互垂直构成笛卡尔坐标系，也可采用任意布局。
- 机床轴的名称可设置。
- 机床轴可为线性轴或回转轴。

BCS

基本坐标系（BCS）具有以下属性：

- 由几何轴构成的笛卡尔直角坐标系。
- BCS 通过从 MCS 运动转换得到。

BZS

基本零点坐标系（BZS）是带基准偏移的基本坐标系。

SZS

从工件坐标系（WCS）角度看来，可设定的零点坐标系（SZS）是带可编程框架的 WCS。 工件零点通过可设定框架 G54 ... G599 定义。

WCS

工件坐标系（WCS）具有以下属性：

- 在 WCS 中，所有轴坐标均为编写（零件程序）。
- 由几何轴和辅助轴构成。
- 几何轴总是构成一个笛卡尔直角坐标系。
- 辅助轴构成一个坐标系，辅助轴之间无几何关联。
- 几何轴和辅助轴的名称可设置。
- 可通过框架实现对 WCS 的偏移、旋转、比例缩放和镜像（TRANS、ROT、SCALE、MIRROR）。
  也可实现多重偏移、多重旋转等。

外部零点偏移

外部零点偏移具有以下属性：

- 预先定义的基本坐标系和工件坐标系间的附加零点偏移，在一个由 PLC 决定的时间点激活。
- 对于涉及的每根轴，偏移量均可通过以下方式设置：
  - PLC
  - 操作面板
  - 零件程序

- 激活后，该偏移从执行轴的第一个运动程序段起开始生效。 这些偏移会被叠加至编写的路径（非插补）。
  运行外部零点偏移的速度为：
  编写的 F 值 + 1/2 JOG 速度
  在 G0 程序段中，零点偏移会在程序段末尾运行。
- RESET 和程序结束时，激活的偏移将被保留。
- 上电后，上一次生效的偏移继续保存在控制系统中，但是必须通过 PLC 重新激活。

4.1.3 框架

框架是一种既定的，用于实现笛卡尔坐标系转换的计算规定。

框架分量

一个框架由以下分量组成：

框架分量		编程时可使用：
偏移	粗偏	TRANS ATRANS（叠加偏移分量） CTRANS（对多根轴的零点偏移） G58（轴零点偏移）
	精偏	CFINE G59（轴零点偏移）
旋转		ROT / ROTS AROT / AROTS CROTS
比例缩放		SCALE ASCALE
镜像		MIRROR AMIRROR

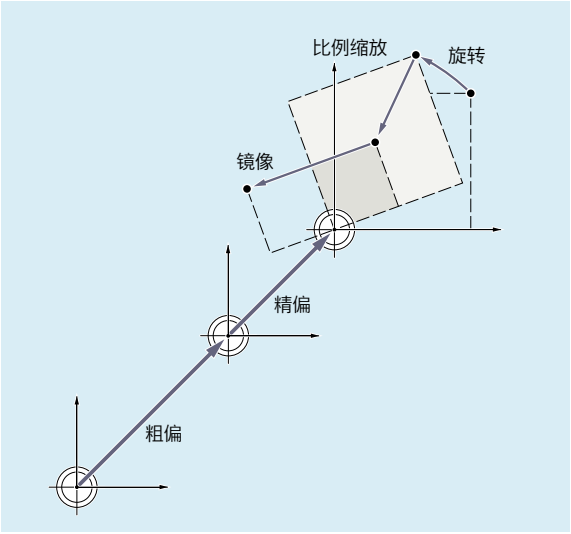


图 4-1　框架分量

粗偏和精偏

在所有通道中，机床轴对通道轴的指定关系，特别是机床轴对几何轴的指定关系可能不同，因此通道轴之间不存在几何关联。因此，NCU 全局框架中只可进行偏移、缩放和镜像。不可旋转。

G58：绝对轴专用可编程零点偏移（粗偏移）

通过 G58 编程平移（粗偏移）的绝对分量。保留平移（精偏移）的叠加分量。

G59：叠加轴专用可编程零点偏移（精偏移）

通过 G59 编程平移（精偏移）的叠加分量。保留平移（粗偏移）的绝对分量。

仅当精偏移使能时才可使用 G59：

- MD18600 $MN_MM_FRAME_FINE_TRANS = TRUE
- MD24000 $MC_FRAME_ADD_COMPONENTS = TRUE

旋转

空间中的定向通过框架旋转如下定义：

- 通过 ROT 定义针对所有几何轴的单次旋转。
- 通过 ROTS、AROTS、CROTS 定义空间角，确定平面在空间内的定向。
- 通过 TOFRAME 定义框架旋转，确定一个 Z 轴指向刀具方向的框架。

比例缩放

使用 SCALE 为所有几何轴和辅助轴编写可编程的比例缩放（标度系数）。

若需在其他比例缩放、旋转、偏移或镜像的基础上建立新的比例缩放，则须编写 ASCALE。

镜像

通过以下机床数据设置如何执行镜像：

MD10610 $MN_MIRROR_REF_AX

级联

框架和框架分量可通过级联运算符 ":" 进行级联。通道专用框架 $P_ACTFRAME 由所有通道生效框架级联得出：

$P_ACTFRAME = $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :
$P_ISO1FRAME : $P_ISO2FRAME : $P_ISO3FRAME :
$P_ACTBFRAME : $P_IFRAME : $P_GFRAME :
$P_TOOLFRAME : $P_WPFRAME : $P_TRAFRAME :
$P_PFRAME : $P_ISO4FRAME : $P_CYCFRAME

边界条件

增量尺寸设定（G91）

对于 G91 增量编程，选择零点偏移时，补偿值会被累加至增量编写的值一并运行。

此特性取决于以下设定数据中的设置：

SD42440 $SC_FRAME_OFFSET_INCR_PROG （框架中的零点偏移）

值	含义
1	在 FRAME 和轴采用增量编程的情形下运行零点偏移（= 缺省设置）。
0	只运行编写的行程。

一致性

在写入、读取和激活框架（例如通过通道协调）时，用户须自行确保通道内的一致性。无法跨通道激活框架。

4.2 轴

4.2.1 概述

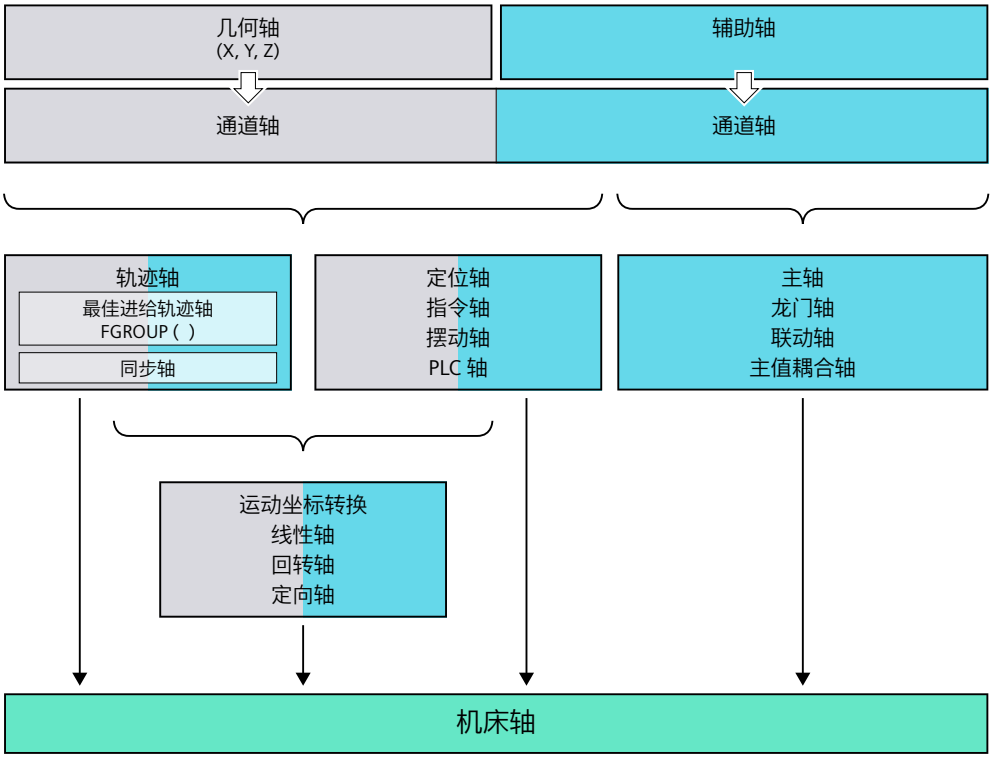


图 4-2 几何轴、辅助轴和机床轴之间的关联

4.2.2 机床轴

含义

机床轴指的是在机床上实际存在的轴。

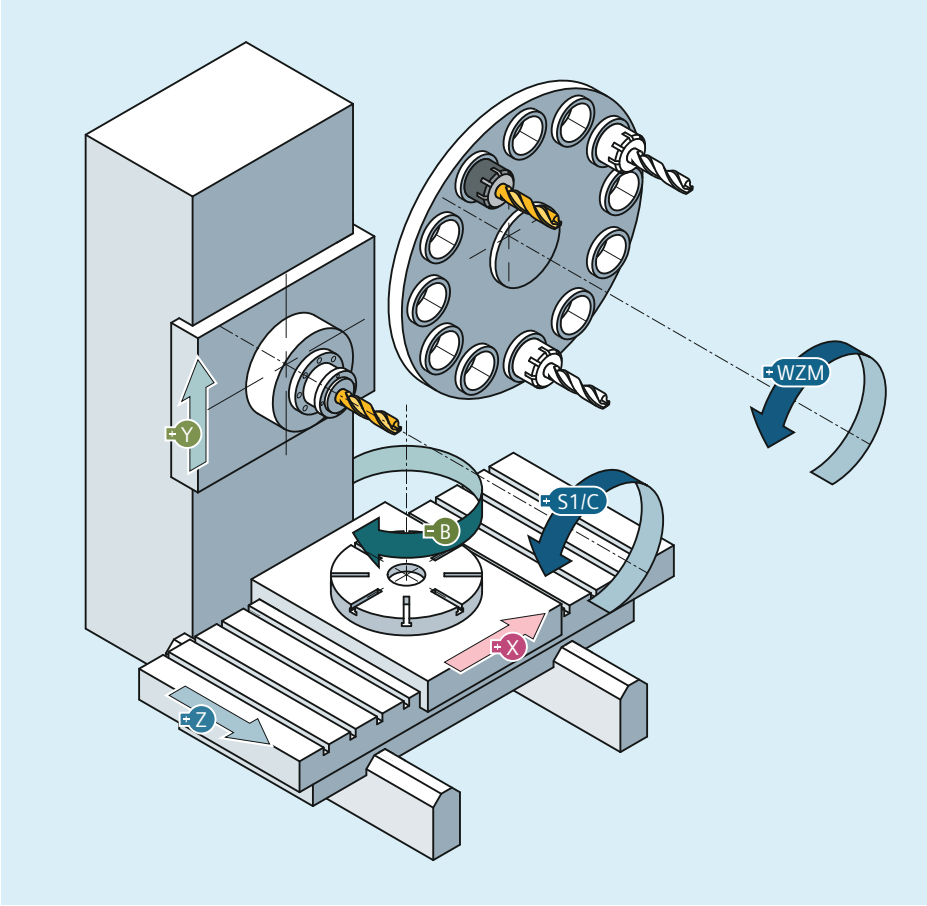


图 4-3　一台直角坐标机床的机床轴 X、Y、Z、B、S

应用

机床轴可以是：

- 几何轴 X、Y、Z
- 定向轴 A、B、C
- 进料轴
- 刀具转塔
- 用于刀库的轴

- 用于移动机械手的轴
- 顶尖套筒
- 用于盘式机械手的轴
- 等

4.2.3 通道轴

含义

每根几何轴和每根辅助轴均被指定给一个通道。 几何轴和辅助轴总是在所归属于的通道中运行。

4.2.4 几何轴

含义

三根几何轴总是构成一个虚拟的直角坐标系。

通过使用框架（偏移、旋转、比例缩放、镜像）可将工件坐标系（WCS）的几何轴映射至 BCS。

应用

几何轴用于编写工件几何数据（轮廓）。

平面选择 G17、G18 和 G19（DIN 66217）总是以三根几何轴为基准。 因此将三根几何轴命名为 X、Y 和 Z 会很便利。

4.2.5 辅助轴

含义

与几何轴不同，辅助轴之间未定义几何关联。

说明

几何轴之间的关联则为精确定义，以直角坐标系为形式。

辅助轴是基本坐标系（BCS）的组成部分。 通过使用框架（偏移、比例缩放、镜像）可将工件坐标系（WCS）的辅助轴映射至 BCS。

应用

典型的辅助轴有：

- 回转轴
- 刀具刀库轴
- 刀具转塔轴
- 进料轴

4.2.6 轨迹轴

含义

轨迹轴的特点是一同进行插补（一个通道的所有轨迹轴共用一个轨迹插补器）。

一个通道的所有轨迹轴具有共同的加速、恒速运行和减速阶段。

在地址 F 下编写的进给率（轨迹进给率）适用于所有在程序段中编程的轨迹轴，但有以下特例：

- 编写的轴通过 FGROUP 指令定义为不决定轨迹速度的轴。
- 通过 POS 或 POSA 指令编写的轴，具有独立的进给率（轴插补器）。

应用

轨迹轴用于以编写的轮廓加工工件。

4.2.7 定位轴

含义

定位轴的特点是独立进行插补（每根定位轴均有一个专属的轴插补器）。每根定位轴均具有独立的进给率和加速特性曲线。可编写定位轴作为对轨迹轴的补充（也可编写在同一程序段中）。轨迹轴的插补（轨迹插补器）不会受定位轴影响。轨迹轴和单独的定位轴不必同时到达程序段终点。

通过 POS 和 POSA 指令编写定位轴和定义程序段切换标准：

- POS
  轨迹轴和定位轴到达程序段终点时，进行程序段切换。
- POSA
  轨迹轴到达程序段终点时，进行程序段切换。定位轴超越程序段界限运行至程序段终点。

并行定位轴相比定位轴有以下区别：

- 只从 PLC 获取程序段终点。
- 可在任意时间点（非程序段交界处）启动。
- 对运行中的零件程序无影响。

应用

典型的定位轴有：

- 用于工件传输的进料轴
- 刀库/刀具转塔

更多信息

- 功能手册之进给轴和主轴
- 功能手册之同步动作分册
- 功能手册之 PLC 和基本程序

4.2.8 主处理轴

含义

主处理轴是指通过主处理进行插补的轴。

该插补可通过以下方式启动：

- 从同步动作启动
  （作为指令轴，基于一个逐段、模态或静态同步动作相关的事件）
- 由 PLC 通过特殊功能块在 PLC 基本程序中启动
  （作为并行定位轴，或称为 PLC 轴）
- 通过设定数据或从零件程序启动
  （作为异步或程序段同步往复轴）

控制

通过主处理插补的轴会对以下情形进行响应：

- NC 停止
- 报警处理
- 程序控制
- 程序结束
- 复位

说明

程序末尾的特性不同。 轴运动不必总是在程序末尾完成，因此可超出程序末尾进行。

应用

可在主处理中解除特定轴与由 NC 程序运行触发的通道特性的耦合，并通过 PLC 控制该轴。这些轴同样在主处理中插补，且其特性与通道运行及程序运行无关。

这样一来便可通过 NC 控制由 PLC 控制的轴。 这涉及到下列动作：

- 终止轴运行（等同于删除剩余行程）
- 停止或中断轴
- 继续运行轴（恢复运行）
- 将轴恢复至初始状态

4.2.9 同步轴

含义

同步轴为轨迹轴的一部分，在计算轨迹速度时不用作参考。其和轨迹轴一同进行插补（一个通道的所有轨迹轴和同步轴共用一个轨迹插补器）。

一个通道的所有轨迹轴和同步轴具有共同的加速、恒速运行和减速阶段。

在地址 F 下编写的进给率（轨迹进给率）适用于所有在程序段中编程的轨迹轴，但不适用于同步轴。

对于编写的行程，同步轴所需的运行时间与轨迹轴相同。

FGROUP 指令

通过 FGROUP 指令可定义将轴作为决定进给率的**轨迹轴**（计算轨迹速度时作为参考）或**同步轴**（计算轨迹速度时不作为参考）。

示例

程序代码	注释
N05 G00 G94 G90 M3 S1000 X0 Y0 Z0	;
N10 FGROUP(X,Y)	; 轴 X 和 Y 为轨迹轴 轴 Z 为同步轴
N20 G01 X100 Y100 F1000	; 编写的进给率 1000 mm/min 轴 X 的进给率 = 707 mm/min 轴 Y 的进给率 = 707 mm/min
N30 FGROUP (X)	; 轴 X 为轨迹轴 轴 Y 为同步轴
N20 X200 Y150	; 编写的进给率 1000 mm/min 轴 X 的进给率 = 1000 mm/min 轴 Y 的进给率设为 500 mm/min，因为只须运行一半行程。

说明

在 FGROUP 指令中必须使用通道名称。

这通过以下机床数据定义：

MD20080 $MC_AXCONF_CHANAX_NAME_TAB（通道中的通道轴名称）

应用

在螺旋线插补中，可选择通过 FGROUP 设置：

- 编写的进给率是否在轨迹上生效
  （编写的 3 根轴均为轨迹轴）
- 编写的进给率是否在圆弧上生效
  （2 根轴为轨迹轴，进给轴为同步轴）

4.2.10 轴配置

几何轴、辅助轴、通道轴、机床轴和驱动之间的对应关系

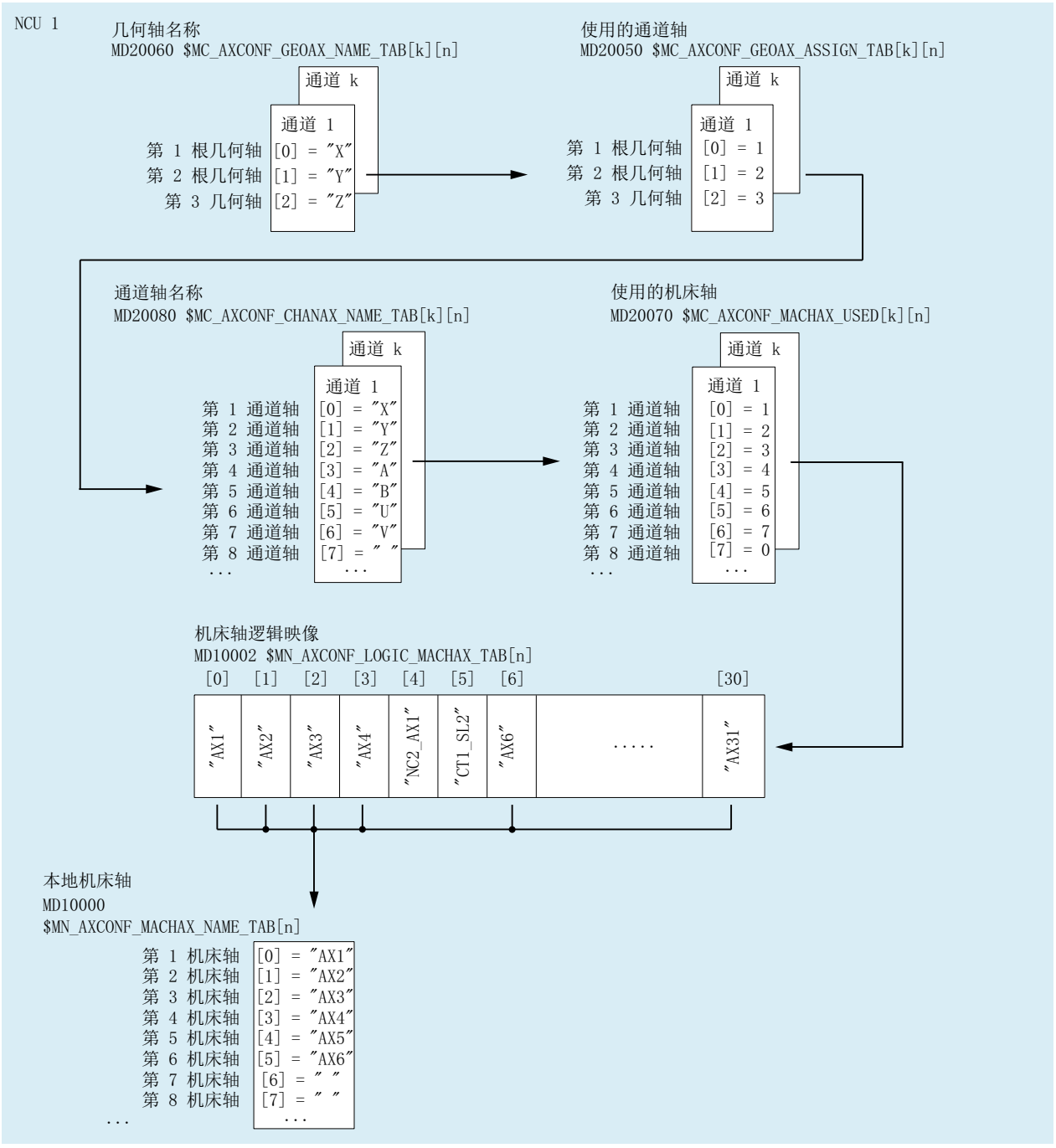


图 4-4 轴分配

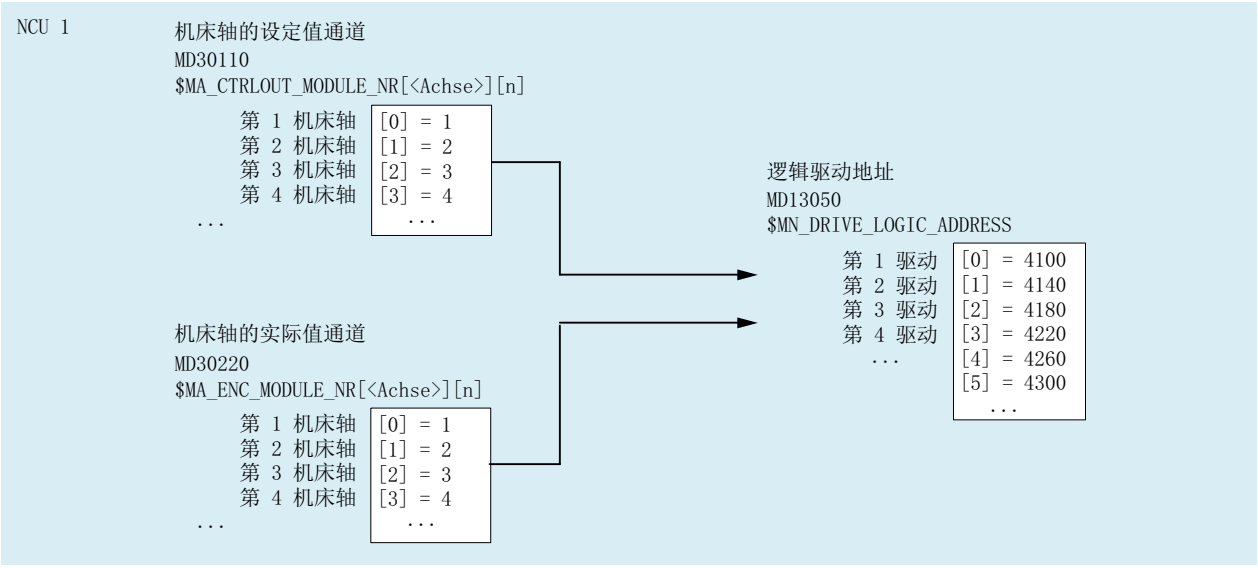


图 4-5 驱动分配

边界条件

- 用户定义轴名称中，开头的零会被忽略：
  MD10000 `$MN_AXCONF_MACHAX_NAME_TAB[0] = X01，等同于 X1
- 通道轴必须按升序无空隙地指定为几何轴。
- 所有非几何轴的通道轴均为辅助轴。

通道轴空隙

通常情形下，必须通过 MD20070 $MC_AXCONF_MACHAX_USED 为机床数据 MD20080 $MC_AXCONF_CHANAX_NAME_TAB 中定义的每根通道轴指定一根机床轴。

为了简化对机床轴数量不同的机床系列的调试，也可定义未指定机床轴的通道轴。这样一来，通道轴升序排列时可能会存在空隙。

对通道轴空隙的许可必须显性使能：

MD11640 $MN_ENABLE_CHAN_AX_GAP = 1

若未使能，那么机床数据 MD20070 $MC_AXCONF_MACHAX_USED 中的 0 值结束将其他通道轴指定给机床轴。

示例

未向通道轴 B 指定机床轴。

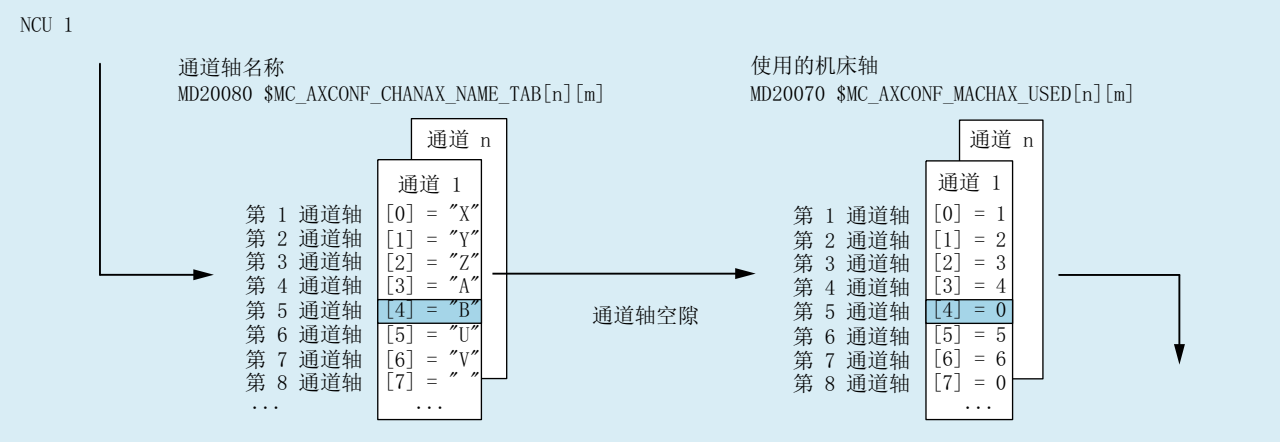


图 4-6　含通道轴空隙的轴配置（节选）

边界条件

- 系统会视数量和通道轴索引将未指定机床轴的通道轴（通道轴空隙）像指定了机床轴的普通通道轴一样处理。
- 若将未指定机床轴的通道轴（通道轴空隙）定义为几何轴，那么此操作会被拒绝，并发出报警。

4.3　零点和参考点

4.3.1　工作区域中的参考点

零点和参考点

机床的中性位置通过坐标轴和机床的构造特性得出。 坐标系零点通过在机床上的中性位置定义一个适宜的参考点来确定。

坐标系（MCS、BCS、BZS、SZS、WCS）的位置通过零点定义。

零点		参考点	
	**M** = 机床零点		**R** = 参考点
	**W** = 工件零点		**T** = 刀架参考点

机床零点 M

使用机床零点 M 可以确定机床坐标系 MCS。所有其他参考点都以机床零点为基准。

工件零点 W

以机床零点 W 为基准的工件零点可以用来确定工件坐标系。 编写的零件程序段在工件坐标系 WCS 中执行。

参考点 R

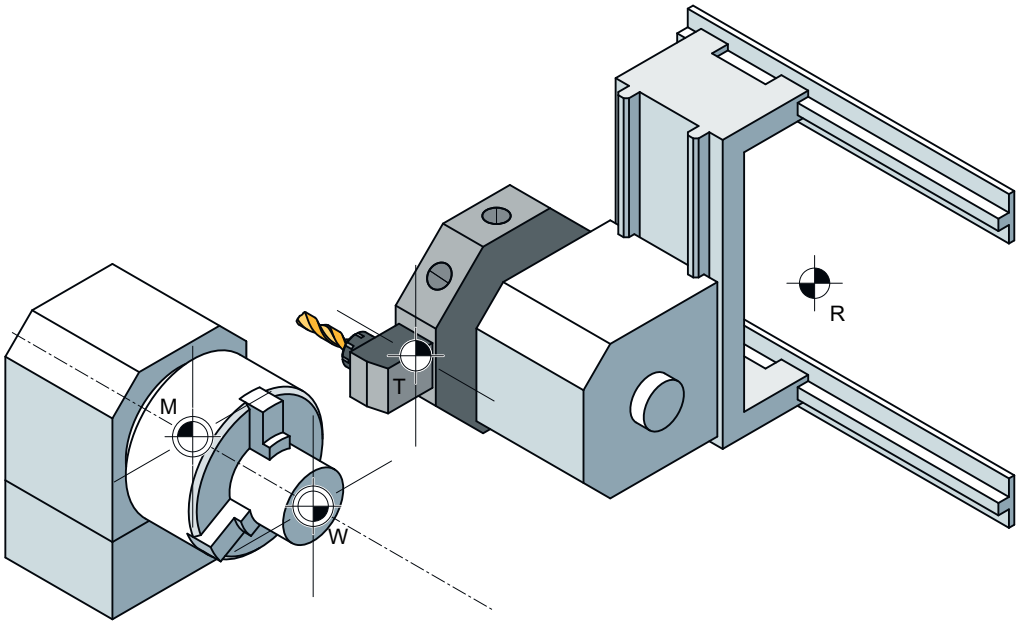
参考点 R 的位置通过凸轮开关设定。 其可用于校准路径测量系统。

使用增量测量编码器时，必须在每次接通控制系统后执行回参考点。 之后该控制系统才可使用测量系统运行，并将所有位置值传输至坐标系。

刀架参考点 T

刀架参考点 T 位于刀架安装处。 通过输入刀具长度，控制系统可以计算出刀尖（TCP，Tool Center Position，刀具中心位置）至刀架参考点的距离。

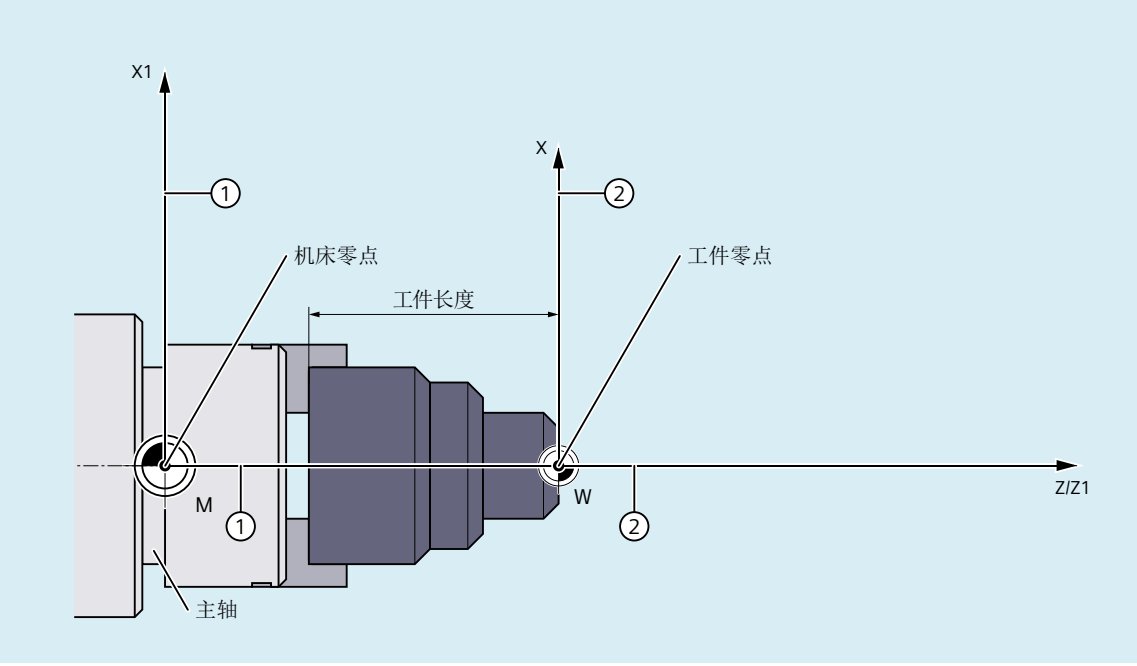
示例：车床上的零点和参考点



4.3.2 坐标系和参考点的位置

接通控制系统

使用增量测量编码器时，每次接通控制系统后必须执行回参考点，确保控制系统能将所有位置值传输至坐标系。



① 机床坐标系（MCS）

② 工件坐标系（WCS）

图 4-7 以机床零点 M 和工件零点 W 为基准的坐标系位置

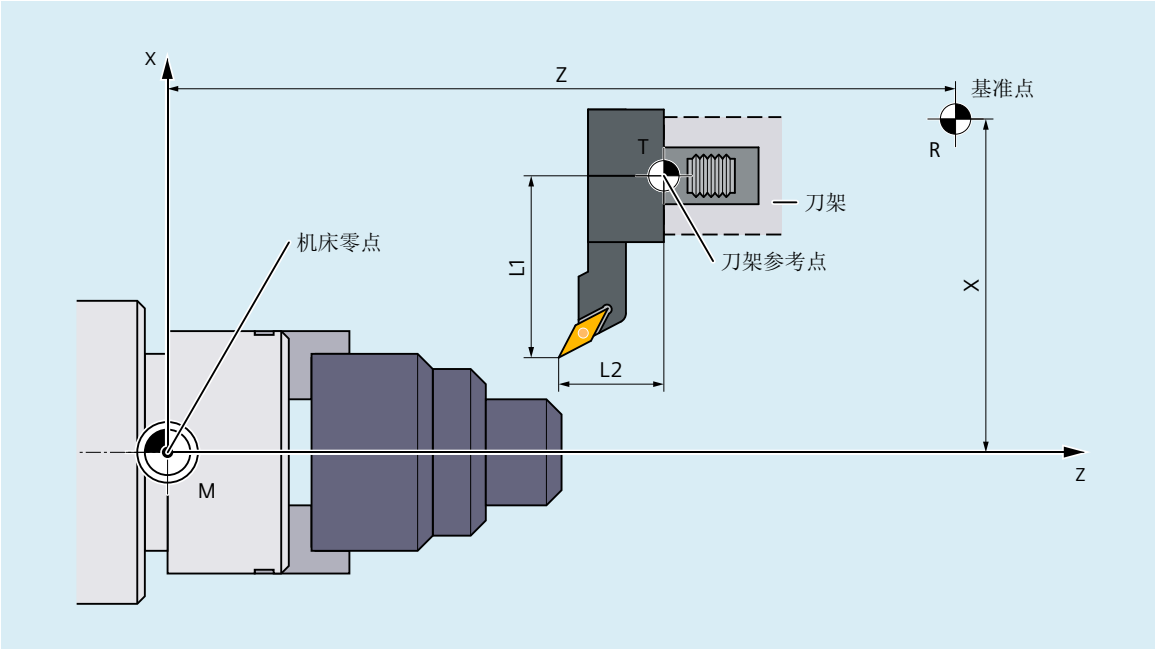


图 4-8　参考点相对机床零点的位置

4.4 坐标系

4.4.1 概述

定义

根据 DIN 66217，在机床的编程中使用直角（笛卡尔）坐标系。可通过“右手定则”确定坐标轴的正方向。

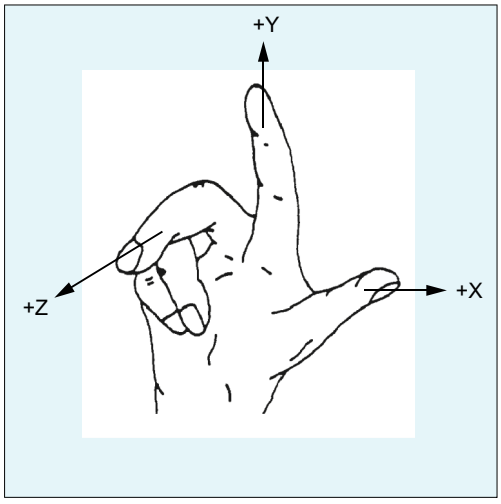


图 4-9 右手定则

编程采用的坐标系以工件为基准。编程不受刀具或者工件移动的影响。编程时始终假定工件静止，而刀具相对于工件坐标系发生位移。

通过“右手定则”也可确定回转轴的正（右旋）旋转方向。若右手拇指指向坐标轴（线性轴）的正方向，则手指指向相应回转轴的正旋转方向。

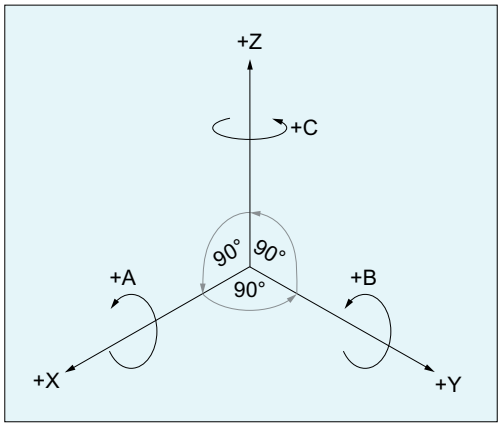


图 4-10　右旋的笛卡尔直角坐标系

坐标系

为机床定义的坐标系包括：

坐标系	缩写	注释
Workpiece **C**oordinate **S**ystem，工件坐标系	WCS	在 WCS 中编写几何轴的运行，用于加工工件。
Settable **Z**ero **S**ystem，可设定的零点坐标系	SZS	通过框架进行坐标转换： WCS ⇒ SZS
Basic **Z**ero **S**ystem，基本零点坐标系	BZS	通过框架进行坐标转换： SZS ⇒ BZS
Basic **C**oordinate **S**ystem，基本坐标系	BCS	通过框架进行坐标转换： BZS ⇒ BCS
Maschine **C**oordinate **S**ystem，机床坐标系	MCS	通过框架和运动转换进行坐标转换： BCS ⇒ MCS 在 WCS 中编写的几何轴运行通过框架和运动转换映射至 MCS 的机床轴。通过运动转换可将 MCS 转换为 BCS。因此一些诸如手动运行等功能不在 MCS 中而是在 BCS 中进行。

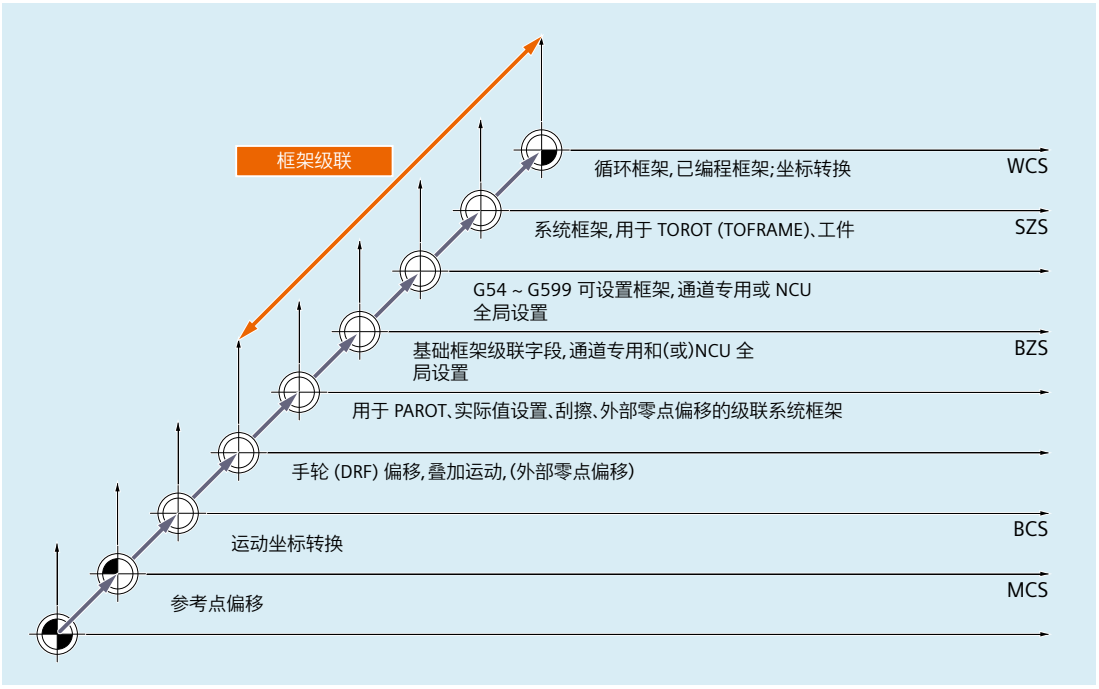


图 4-11　坐标系、框架和运动转换

4.4.2 机床坐标系（MCS）

机床坐标系（MCS）

机床坐标系（MCS）由所有实际存在的机床轴构成。

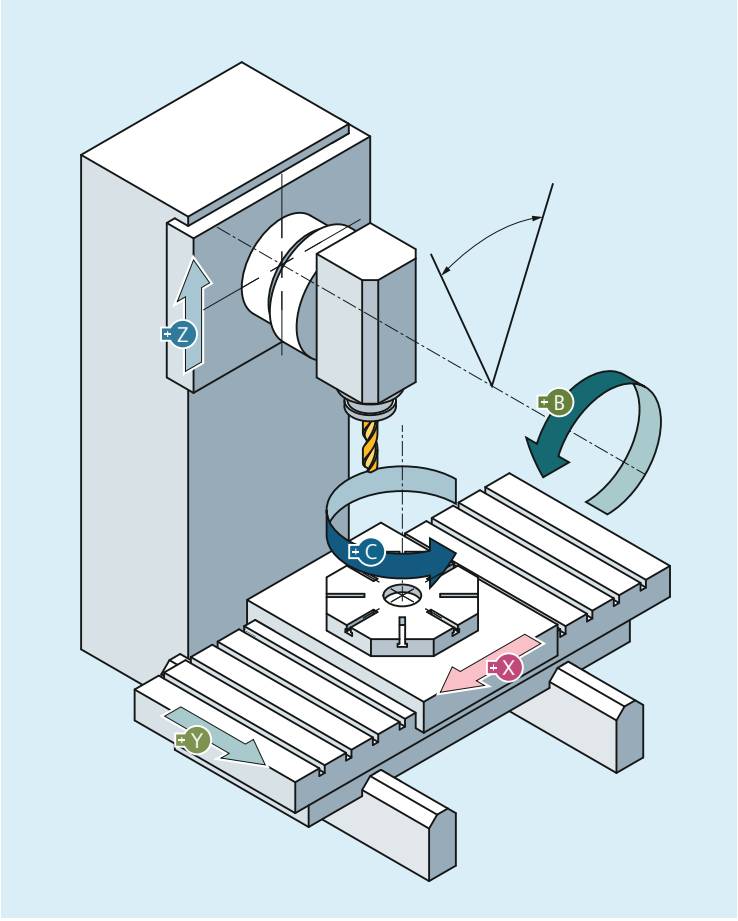


图 4-12　机床轴 X、Y、Z、B、C 构成的 MCS（5 轴铣床）

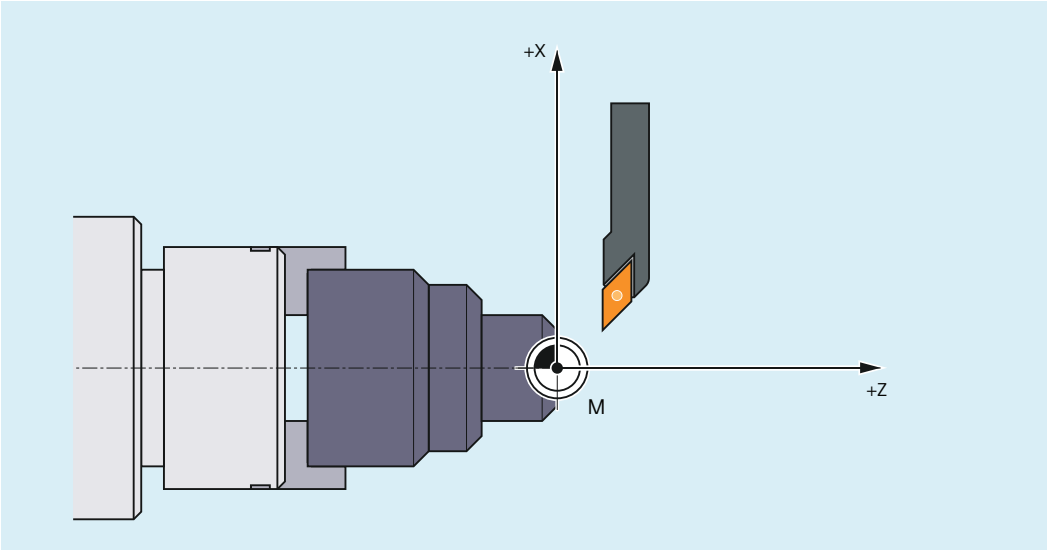


图 4-13　机床轴 X、Z 构成的 MCS（车床）

轴预设偏移

通过“预设定偏移（PRESETON）”功能，可以机床坐标系（机床零点）重新设置控制系统参考点。

小心

编码器校准丢失

在预设定偏移后，对应的机床轴切换至“未回参考点”状态！在使用绝对值编码器时，这意味着编码器校准将丢失并须重新执行（例如通过激光干涉仪测量）。因此不建议将 PRESETON 功能与绝对值编码器配合使用。

说明

建议只对不需回参考点的机床轴使用此功能。

若需恢复原始的机床坐标系，必须使机床轴重新回参考点，例如使用 G74（回参考点运行）。

启用预设定偏移时，机床轴不发生运动。

更多信息

- “NC 编程”手册，补充指令
- “NC 编程”手册，坐标转换 (FRAMES)

4.4.2.1 参考点状态的实际值设置和损失（PRESETON）

功能

程序 PRESETON() 在机床坐标系 (MCS) 为一个或多个轴设置新的实际值。该值等于轴的 MCS 零点偏移。因此不运行此轴。

通过 PRESETON 可触发一个带同步的预处理停止。实际位置在轴静止时分配。

如果轴在 PRESETON 指令下未分配到通道，随后的过程取决于以下机床数据中参数设置的跨通道取轴特性：

MD30552 $MA_AUTO_GET_TYPE

回参考点状态

通过在机床坐标系设置新的实际值使机床轴参考点状态复位：

<Axis>.basic.in.enc1Synchronized /<Axis>.basic.in.enc2Synchronized = 0（测量系统 1 / 2 已回参考点/已同步）

建议 PRESETON 只用于没有参考点义务的轴。

为了恢复原来的机床坐标系，机床轴的测量系统必须通过例如从零件程序接近参考点（G74）来重新返回到参考点。

⚠ 小心

参考点状态丢失

通过 PRESETON 在机床坐标系设置新的实际值，使机床轴的参考点状态重置到“未返回参考点/未同步”。

编程

句法

```
PRESETON(<轴_1>, <值_1> [, <轴_2>, <值_2>, ... <轴_8>, <值_8>])
```

含义

PRESETON:	参考点状态的实际值设置和损失	
	预处理停止：	是
	在单独程序段中编程：	是

<轴_x>: 机床进给轴名称
类型: AXIS
值域: 在通道定义的机床轴名称

<值_x>: 机床坐标系中机床轴的新实际值（MCS）
在当前有效的尺寸系统输入（英制/公制）
考虑到有效的直径编程 (DIAMON)
类型: REAL

系统变量

$AC_PRESET

轴专用系统变量 $AC_PRESET 提供了从当前移动的 MCS 的零点到机床轴回参考点后初始 MCS_0 的零点的矢量。

$AC_PRESET<轴> = $AC_PRESET<轴> + "轴在 MCS 中的当前实际位置" - "PRESETON 实际位置"

通过系统变量可再次取消零点偏移：

PRESETON(<轴>, $VA_IM + $AC_PRESET[<轴>]; "轴在 MCS 中的当前实际值" + "偏移"

示例

程序代码

```
N10 G1 X10 F5000
N20 PRESETON(X, $AA_IM[X]+70) ; 实际值 = 10 + 70 = 80 =>
                              ; $AC_PRESET = $AC_PRESET - 70
```

边界条件

不可以使用 PRESETON 的轴

- 正在运行的轨迹轴
- 正在运行的定位轴
- 主轴模式下正在运行的指令轴
- 正在运行的并行定位轴
- 参与切换的轴

- 摆动轴
- 端面齿轴
- 龙门轴组中的同步轴
- 用于从零件程序接近参考点 (G74) 的轴生效
- 转速/转矩耦合中的从动轴

几何轴

- PRESETON 可在静态几何轴上使用，只要此时通道中没有另一根几何轴在同步运行。
- PRESETON 可在静态几何轴上使用，即使此时通道中有另一根几何轴在“中性轴”状态下同步运行或作为指令轴同步运行。

程序代码	注释
N10 G0 X0 Y0	; X, Y:几何轴
N15 RELEASE(Y)	; 中性轴
N20 PRESETON(Y,20)	; Y 在 MCS 中的实际位置 = 20
N30 G0 X40	; 几何轴 X 运行
N40 M30	

示例：另一根几何轴 (X) 在“**中性轴**”状态下同步运行

程序代码	注释
N10 G0 X0 Y0	; X, Y:几何轴
N20 POS[X]=40 FA[X]=1000	; 指令轴 X
N30 DO PRESETON(Y,20)	; Y 在 MCS 中的实际位置 = 20
N40 M30	

示例：几何轴（X）同时作为**指令轴**运行

PLC 控制的轴

PRESETON 可在 PLC 控制的轴上使用（根据其当前类型）。

主轴状态

下表列出了 PRESETON 在同步动作中对主轴的响应：

NC 程序中的 PRESETON			
主轴运行方式	**运行状态**	**分配给 NC 程序**	**主运行轴**
转速控制模式	运动中	报警 22324	报警 22324
	静止	+	+
定位模式 SPOS	运动	-	+
	静止	+	+
通过 SPOSA 定位	运动中	报警 10610	-
进给轴模式	运动中	-	+
	静止	+	+
+：允许 -:不允许			

轴耦合

- 引导轴：由 PRESETON 引起的引导轴位置的骤变不会出现在跟随轴中。耦合保持不变。
- 跟随轴：通过 PRESETON 只能控制跟随轴的叠加位置分量。

龙门轴组

如果在龙门轴组中的引导轴上使用 PRESETON，系统则会在龙门轴组的所有同步轴中执行零点偏移。

分度轴

PRESETON 可在分度轴上使用。

软件限位开关、工作区域限制、保护区

如果在零点偏移后 PRESETON 使轴位置超出了规定的范围，那么只有在尝试运行轴时，系统才会显示报警。

带计算的程序段搜索

程序段搜索期间，系统会收集 PRESETON 指令并通过 NC 启动继续执行 NC 程序。

NC/PLC 接口信号

因实际位置发生变化，需重新确定 NC/PLC 接口信号的状态。

示例：固定点位置

- 设置的固定点位置：MD30600 $MA_FIX_POINT_POS[0...3] = <固定点位置 1...4>
- NC/PLC 接口信号：<Axis>.basic.in.jogFixedPointReachedNumber（JOG 逼近固定点：已到达）

如果轴位于固定点位置上（准停公差范围内），系统会置位对应的 NC/PLC 接口信号。如果通过 PRESETON 将实际值设为另一个准停公差范围外的值，系统会复位接口信号。

DRF 偏移

通过 PRESETON 删除轴的 DRF 偏移。

叠加运动 $AA_OFF

同步动作中的叠加运动 $AA_OFF 不受 PRESETON 影响。

在线刀具补偿 FTOC

同步动作中的生效在线刀具补偿 FTOC 在 PRESETON 后继续生效。

轴专用补偿

轴专用补偿在 PRESETON 后继续生效。

运行方式 JOG

PRESETON 只可在静态轴上使用。

运行方式 JOG，机床功能 REF

不可以使用 PRESETON。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Axis>.basic.in.enc1Synchronized	LBP_typeAxisX.E_RefSyn1	DB31, ...DBX60.4
<Axis>.basic.in.enc2Synchronized	LBP_typeAxisX.E_RefSyn2	DB31, ...DBX60.5
<Axis>.basic.in.jogFixedPointReachedNumber	LBP_typeAxisX.E_JogFixPPos0..2	DB31, ...DBX75.3..5

4.4.2.2 参考点状态的实际值设置，无损失（PRESETONS）

功能

程序 PRESETONS () 在机床坐标系 (MCS) 为一个或多个轴设置新的实际值。该值等于轴的 MCS 零点偏移。因此不运行此轴。

通过 PRESETON 可触发一个带同步的预处理停止。实际位置在轴静止时分配。

如果轴在 PRESETONS 指令下未分配到通道，随后的过程取决于跨通道取轴特性的轴专用配置：

MD30552 $MA_AUTO_GET_TYPE

回参考点状态

通过 PRESETONS 在机床坐标系 (MCS) 设置新的实际值，机床轴的参考点状态**不会**改变。

前提条件

- **编码器类型**
  PRESETONS 仅适用有效测量系统的以下编码器类型：
  - MD30240 $MA_ENC_TYPE[<测量系统>] = 0（仿真编码器）
  - MD30240 $MA_ENC_TYPE[<测量系统>] = 1（原始信号编码器）
- **回参考点模式**
  PRESETONS 仅适用有效测量系统的以下回参考点模式：
  - MD34200 $MA_ENC_REFP_MODE[<测量系统>] = 0（无法回参考点）
  - MD34200 $MA_ENC_REFP_MODE[<测量系统>] = 1（增量、旋转或线性测量系统的回参考点：编码器信号上的零脉冲）

调试

轴专用机床数据

必须针对具体轴激活无回参考点状态损失的实际值设定（PRESETONS）：

MD30455 $MA_MISC_FUNCTION_MASK, 位 9 = 1

说明

取消 PRESETON

激活“**无**回参考点状态损失的实际值设定 PRESETONS”功能会将“**有**回参考点状态损失的实际值设定 PRESETON”功能取消。两个功能相互排斥。

编程

句法

```
PRESETONS(<轴_1>, <值_1> [, <轴_2>, <值_2>, ... <轴_8>, <值_8>])
```

含义

PRESETONS：	参考点状态的实际值设置，无损失	
	预处理停止：	是
	在单独程序段中编程：	是
<轴_x>：	机床进给轴名称	
	类型：	AXIS
	值域：	在通道定义的机床轴名称
<值_x>：	机床坐标系中机床轴的新的当前实际值 (MCS)	
	该值以生效的单位制（英制/公制）为基准	
	考虑到有效的直径编程 (DIAMON)	
	类型：	REAL

系统变量

$AC_PRESET

轴专用系统变量 $AC_PRESET 提供了从当前移动的 MCS 的零点到机床轴回参考点后初始 MCS_0 的零点的矢量。

$AC_PRESET<轴> = $AC_PRESET<轴> + “轴在 MCS 中的当前实际位置” - “PRESETONS 实际位置”

通过系统变量可再次取消零点偏移：

PRESETONS(<轴>, $VA_IM + $AC_PRESET[<轴>])；“轴在 MCS 中的当前实际值” + “偏移”

示例

轴 X 的 MCS 的零点偏移有 70 个单位。

通过 PRESETONS 将编程的轴 X（指令轴）的最终位置转换到 MCS 中。

程序代码

```
N10 G1 X10 F5000
```

程序代码

```
N20 PRESETONS(X, $AA_IM[X]+70) ; 实际值 = 10 + 70 = 80 =>
                               ; $AC_PRESET = $AC_PRESET - 70
```

边界条件

不可以使用 PRESETONS 的轴

- 正在运行的定位轴
- 主轴模式下正在运行的指令轴
- 正在运行的并行定位轴
- 参与切换的轴
- 正在运行的轨迹轴
- 摆动轴
- 有以下一个或多个 Safety Integrated 功能生效的轴：
  - 使能“安全软件限位开关”
    MD36901 $MA_SAFE_FUNCTION_ENABLE[<SafeAxis>]，位 1 = 1
  - 使能“安全软件挡块”，对 1 ... 4，挡块 +/-
    MD36901 $MA_SAFE_FUNCTION_ENABLE[<SafeAxis>]，位 8 ... 15 = 1
    或者
    使能"安全软件凸轮", Nocke1...30
    MD36903 $MA_SAFE_CAM_ENABLE[<SafeAxis>], Bit 0 ... 29 = 1
- 端面齿轴
- 龙门轴组中的同步轴
- 用于从零件程序接近参考点 (G74) 的轴生效
- 转速/转矩耦合中的从动轴

几何轴

- PRESETONS 可在静态几何轴上使用，只要此时通道中没有另一根几何轴在同步运行。
- PRESETONS 可在静态几何轴上使用，即使此时通道中有另一根几何轴在“中性轴”状态下同步运行或作为指令轴同步运行。
  示例：另一根几何轴 (X) 在“**中性轴**”状态下同步运行

程序代码	注释
N10 G0 X0 Y0	; X, Y:几何轴
N15 RELEASE(Y) 1)	; 中性轴
N20 PRESETONS(Y,20)	; Y 在 MCS 中的实际位置 = 20
N30 G0 X40	; 几何轴 X 运行
N40 M30	
1) **提示** 在使能同步动作动作分量中的轴时不能确保及时使能。 N20 ID=1 WHEN 20.0 < $AA_IM[X] DO RELEASE(Y) PRESETONS(Y,20) ; 不建议！	

示例：另一几何轴（X）同时作为**指令轴**运行

程序代码	注释
N10 G0 X0 Y0	; X, Y:几何轴
N20 POS[X]=40 FA[X]=1000	; 指令轴 X
N30 PRESETONS(Y,20)	; Y 在 MCS 中的实际位置 = 20
N40 M30	

PLC 控制的轴

PRESETONS 可在 PLC 控制的轴上使用（根据其当前类型）。

主轴状态

下表列出了 PRESETONS 在同步动作中对主轴的响应：

NC 程序中的 PRESETONS			
主轴运行方式	**运行状态**	**分配给 NC 程序**	**主运行轴**
转速控制模式	运动中	报警 22324	报警 22324
	静止	+	+
定位模式 SPOS	运动	-	+
	静止	+	+

NC 程序中的 PRESETONS			
主轴运行方式	**运行状态**	**分配给 NC 程序**	**主运行轴**
通过 SPOSA 定位	运动中	报警 10610	-
进给轴模式	运动中	-	+
	静止	+	+
+：允许 -:不允许			

轴耦合

- 引导轴：由 PRESETONS 引起的引导轴位置的骤变不会出现在跟随轴中。耦合保持不变。
- 跟随轴：通过 PRESETONS 只能控制跟随轴的叠加位置分量。

龙门轴组

如果在龙门轴组中的引导轴上使用 PRESETONS，系统则会在龙门轴组的所有同步轴中执行零点偏移。

分度轴

PRESETONS 可在分度轴上使用。

软件限位开关、工作区域限制、保护区

如果在零点偏移后 PRESETONS 使轴位置超出了规定的范围，那么只有在尝试运行轴时，系统才会显示报警。

带计算的程序段搜索

程序段搜索期间，系统会收集 PRESETONS 指令并通过 NC 启动继续执行 NC 程序。

NC/PLC 接口信号

因实际位置发生变化，需重新确定 NC/PLC 接口信号的状态。

示例：固定点位置

- 设置的固定点位置：MD30600 $MA_FIX_POINT_POS[0...3] = <固定点位置 1...4>
- NC/PLC 接口信号：<Axis>.basic.in.jogFixedPointReachedNumber（JOG 逼近固定点：已到达）

如果轴位于固定点位置上（准停公差范围内），系统会置位对应的 NC/PLC 接口信号。如果通过 PRESETONS 将实际值设为另一个准停公差范围外的值，系统会复位接口信号。

DRF 偏移

通过 PRESETONS 删除轴的 DRF 偏移。

叠加运动 $AA_OFF

同步动作中的叠加运动 $AA_OFF 不受 PRESETONS 影响。

在线刀具补偿 FTOC

同步动作中的生效在线刀具补偿 FTOC 在 PRESETONS 后继续生效。

轴专用补偿

轴专用补偿在 PRESETONS 后继续生效。

运行方式 JOG

PRESETONS 只可在静态轴上使用。

运行方式 JOG，机床功能 REF

不可以使用 PRESETONS。

同步动作

若在一个针对轴的同步动作中包含多个 PRESETONS 指令，则系统只执行从左向右的序列中的最后一个。

示例：

程序代码

```
N10 ID=1 WHEN TRUE DO PRESETONS(X,40) PRESETONS(X,39) PRESETONS(X,38)
; 含义等同于：
N10 ID=1 WHEN TRUE DO PRESETONS(X,38)
```

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Axis>.basic.in.jogFixedPointReachedNumber	LBP_Axis*.E_JogFixPPos0..2	DB31, ...DBX75.3..5

4.4.3 基本坐标系（BCS）

基本坐标系（BCS）

基本坐标系（BCS）由三条相互垂直的轴（几何轴）、以及其他没有几何关联的轴（辅助轴）构成。

无运动转换的机床

将无运动转换（例如 TRANSMIT/端面转换，5 轴转换和最多三根机床轴）的 BCS 映射至 MCS 时，BCS 和 MCS 总是重合。

在此类机床上，机床轴与几何轴可以使用相同的名称。

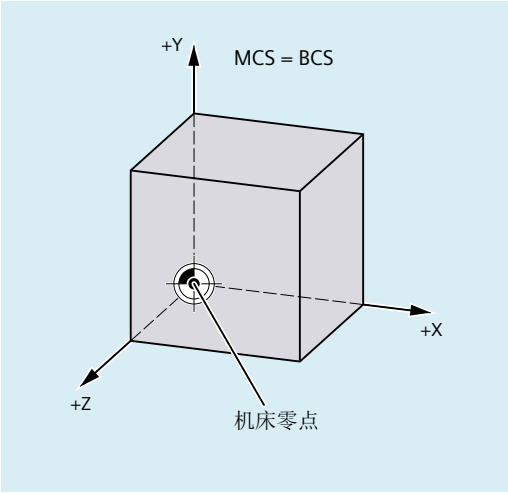


图 4-14　MCS=不进行运动转换的 BCS

带运动转换的机床

将带运动转换（例如 TRANSMIT/端面转换，5 轴转换或多于三根轴）的 BCS 映射至 MCS 时，BCS 和 MCS 不重合。

在此类机床上，机床轴与几何轴必须使用不同的名称。

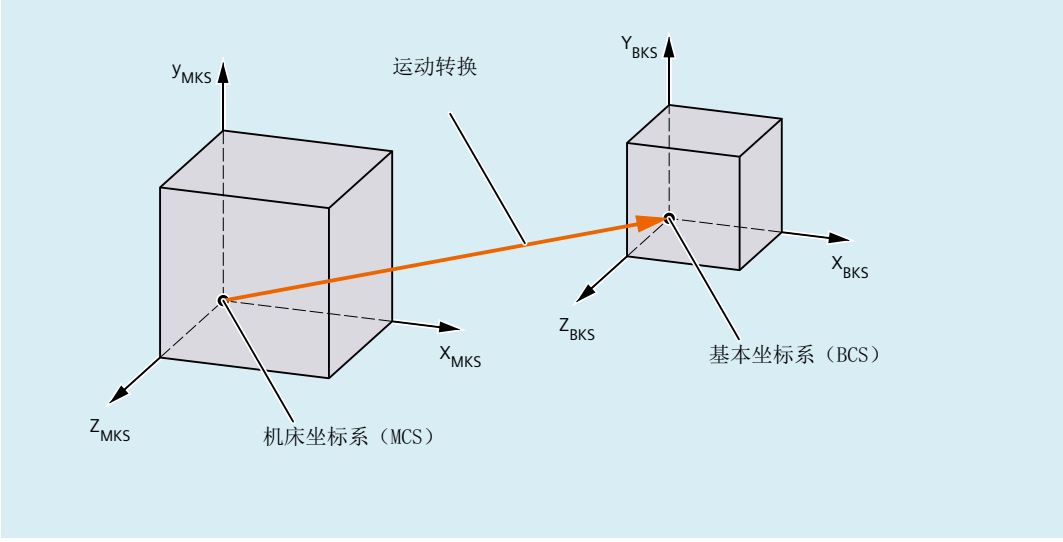


图 4-15　MCS 和 BCS 间的运动转换

机床运动系统

工件编程总是在一个二维或者三维的直角坐标系（WCS）中进行。但加工工件时经常需要使用带回转轴或非垂直布局的直线轴的机床。为了将 WCS 中编程的（直角）坐标投影到实际的加工轴运动中，需要用到运动转换。

更多信息

功能手册之转换分册；多轴转换

4.4.4 基本零点坐标系（BZS）

基本零点坐标系（BZS）

基本零点坐标系（BZS）由基本坐标系通过基本偏移后得到。

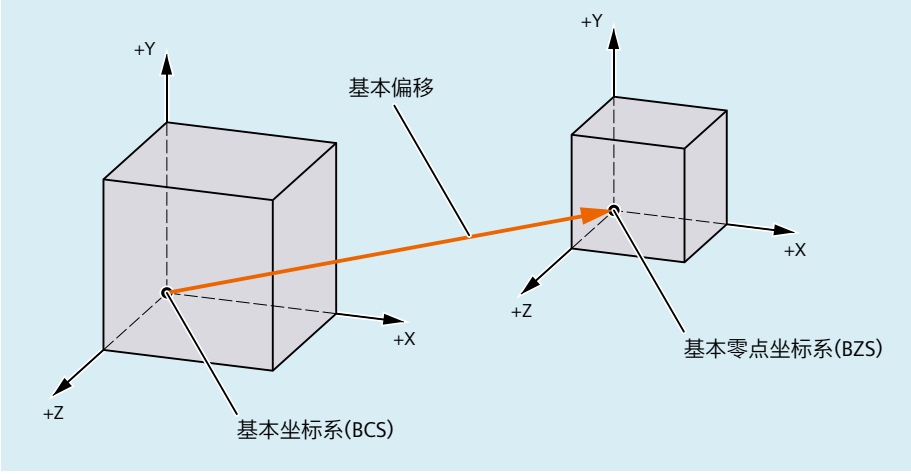


图 4-16 BCS 和 BZS 间的基本偏移

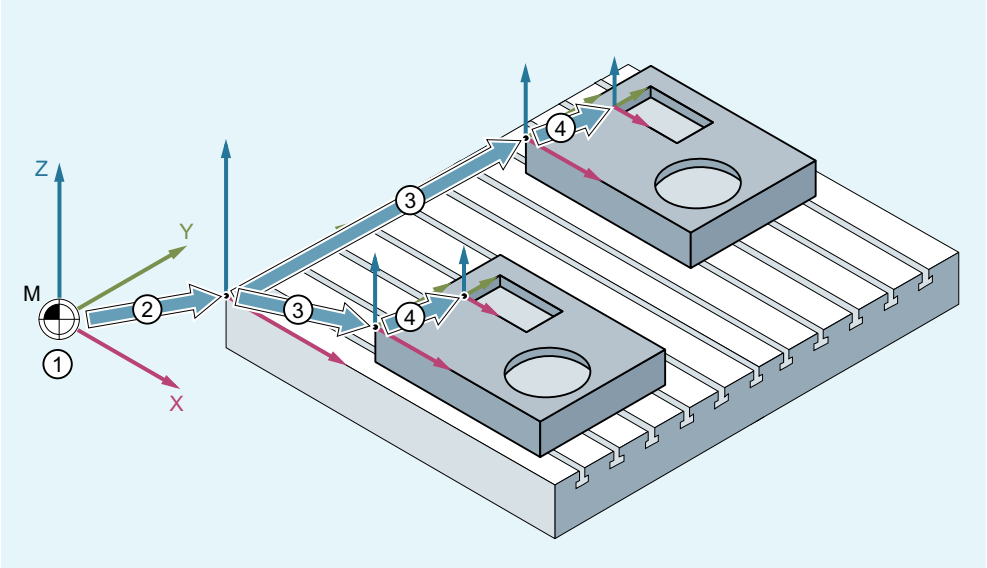
基本偏移

基本偏移表示 BCS 和 BZS 之间的坐标转换。其例如可用于确定托盘零点。

基本偏移由以下部分组成：

- 外部零点偏移
- DRF 偏移
- 叠加运动

- 级联的系统框架
- 级联的基本框架



① 运动转换未激活，即机床坐标系与基准坐标系重合。

② 通过基准偏移得到带有托盘零点的基准零点坐标系（BNS）。

③ 通过可设定的零点偏移 G54 或 G55 来确定用于工件 1 或工件 2 的“可设定零点坐标系”（ENS）。

④ 通过可编程的坐标转换确定工件坐标系（WCS）。

图 4-17 机床坐标系示例

可实现的调整：

- 用户可通过零件程序、操作和 PLC 修改基本偏移。
- 若需使基本偏移立即生效，可通过 PLC 使用 FC9 启动一个 ASUB，由该 ASUB 执行相应的 G 指令。

说明

对机床制造商的建议

针对自有应用，请使用自基本偏移 3 起的偏移。

基本偏移 1 和 2 设计用于“实际值设置”和“外部零点偏移”。

4.4.5 可设定的零点坐标系（SZS）

可设定的零点坐标系（SZS）

从工件坐标系（WCS）角度看来，“可设定的零点坐标系”（SZS）是带可编程框架的WCS。工件零点通过可设定框架 G54 ... G599 定义。

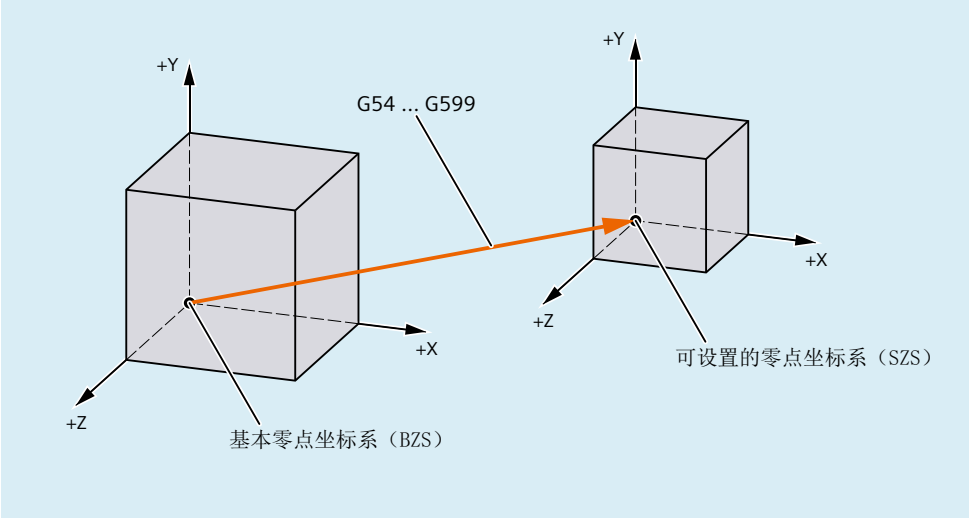


图 4-18 BZS 和 SZS 之间的可设定框架 G54 ... G599

可编程的偏移通过“可设定的零点坐标系”生效。 所有可编程偏移均以“可设定的零点坐标系”为基准。

以 WCS 或 SZS 显示 WCS 实际值

在 HMI 操作界面上，轴的实际值可以机床坐标系（MCS）或工件坐标系（WCS）显示。 以 WCS 显示时，也可基于 SZS 显示实际值。 相应设置通过以下机床数据进行：

MD9424 $MM_MA_COORDINATE_SYSTEM（用于实际值显示的坐标系）

值	含义
0	基于 WCS 显示实际值
1	基于 SZS 显示实际值

说明

当前坐标系的显示

“基于 SZS 显示实际值”生效时，HMI 操作界面上仍会将 WCS 作为实际值显示的基准坐标系加以显示。

示例

基于 WCS 或 SZS 显示实际值

代码（节选）：	实际值显示： 轴 X（WCS）	实际值显示： 轴 X（SZS）
`N10 X100`	100	100
`N20 X0`	0	0
`N30 $P_PFRAME = CTRANS(X,10)`	0	10
`N40 X100`	100	110

4.4.6 工件坐标系（WCS）

工件坐标系 WCS

工件坐标系（WCS）是编程的基础。

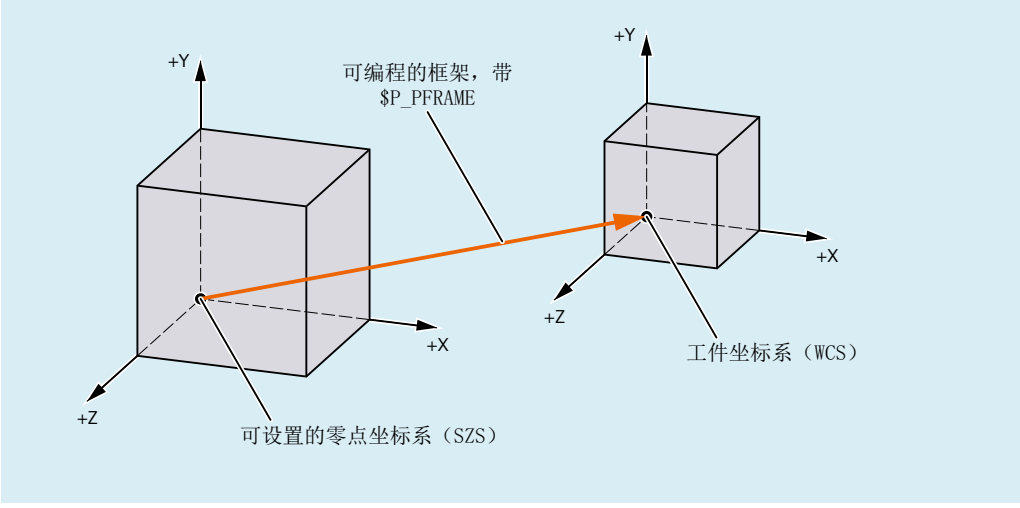
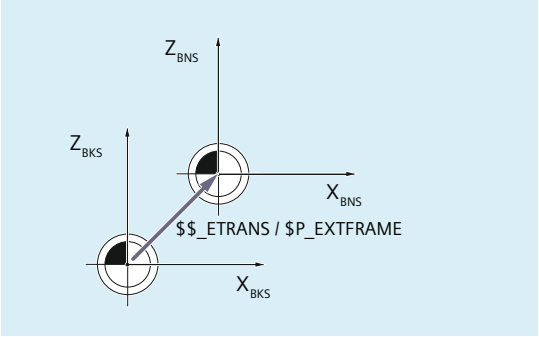


图 4-19　SZS 和 WCS 间的可编程框架

4.4.7 附加补偿

4.4.7.1 外部零点偏移

外部零点偏移是基本坐标系（BKS）和基础零点系统（BNS）之间的线性位移。



外部零点偏移通过 $AA_ETRANS 生效，取决于机床数据的参数设置，有两种方式：

1. 系统变量 $AA_ETRANS 在通过 NC/PLC 接口信号激活后直接作为偏移值生效
2. 系统变量 $AA_ETRANS 的值在通过 NC/PLC 接口信号激活后被接收成为有效的系统框架 $P:EXTFRAME 和数据管理框架 $P_EXTFR 的值。随后激活的总框架 $P_ACTFRAME 将被重新计算。

机床数据

与系统变量 $AA_ETRANS 相关，有两种方法可以区分，通过以下机床数据选择：

MD28082 $MC_MM_SYSTEM_FRAME_MASK，位 1 = <值>

<值>	含义
0	功能：通过 PLC、HMI 或 NC 程序直接写入 $AA_ETRANS[<轴>] 。 解锁，从 $AA_ETRANS[<轴>] 移出零点偏移至下一个可能的运行程序段： <Axis>.basic.out.acceptExternalWorkOffset
1	功能：激活有效的系统框架 $P:EXTFRAME 和数据管理框架 $P_EXTFR 解锁，从 $AA_ETRANS[<轴>] 移出零点偏移，通过： <Axis>.basic.out.acceptExternalWorkOffset。之后在通道内： • 停止通道中的所有运行移动（指令轴和 PLC 轴除外） • 预处理站，随后重组 (STOPRE) • 生效框架的粗偏移 $P_EXTFRAME[<轴>] = $AA_ETRANS[<轴>] • 数据管理框架的粗偏移 $P_EXTFR[<轴>] = $AA_ETRANS[<轴>] • 重新计算有效的总框架 $P_ACTFRAME • 偏移移出至编程轴 • 继续执行中断的运行或 NC 程序

编程

句法

```
$AA_ETRANS[<轴>] = <值>
```

含义

$AA_ETRANS:	系统变量，用于缓存外部零点偏移
<轴>:	通道轴
<值>:	偏移值

NC/PLC 接口信号

激活外部零点偏移：

<Axis>.basic.out.acceptExternalWorkOffset = 0 → 1 ⇒ $P_EXTFRAME[<轴>] = $P_EXTFR[<轴>] = $AA_ETRANS[<轴>]

抑制：外部零点偏移

- 通过指令 SUPA，可在程序段执行期间抑制“外部零点偏移”。
- 在回参考点运行期间，既可通过指令 G74（回参考点运行），也可通过回参考点运行方式下的相应操作抑制生效的外部零点偏移。
- 使用 G74 时，即 “AUTO” 或 “MDI” 操作方式下，之前生效的“外部零点偏移”会在程序段中的下一次运行时自动重新生效。
- 在从“回参考点”模式进行运行方式切换后，必须为回参考点的轴置位用于重新激活的 NC/PLC 接口信号。

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<Axis>.basic.out.acceptExternalWorkOffset	LBP_Axis*.A_ExtZO	DB31, ...DBX3.0

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Axis>.basic.in.enc1Synchronized	LBP_Axis*.E_RefSyn1	DB31, ...DBX60.4
<Axis>.basic.in.enc2Synchronized	LBP_Axis*.E_RefSyn2	DB31, ...DBX60.5
<Axis>.basic.in.jogFixedPointReachedNumber	LBP_Axis*.E_JogFixPPos0..2	DB31, ...DBX75.3..5

4.4.7.2 DRF 偏移

借助 DRF 偏移功能，可**通过手轮**在基本坐标系中为几何轴和辅助轴设置附加的增量零点偏移。

系统变量

DRF 偏移可通过轴专用系统变量读取：

$AC_DRF[<轴>]

更多信息

功能手册之进给轴和主轴；手动运行

4.4.7.3 复位特性

可通过机床数据设置生效框架 $P_EXTFRAME 和数据管理框架 $P_EXTFR 的复位和上电特性：

机床数据

- 通过以下机床数据设置通道中生效的外部零点偏移的系统框架 $P_EXTFRAME 的复位特性：
- MD24006 $MC_CHSFRAME_RESET_MASK，位 1 = <值>

值	含义 **生效的外部零点偏移的系统框架在通道/程序结束复位后：**
0	未生效
1	生效

- 通过以下机床数据设置数据管理外部零点偏移的系统框架 $P_EXTFR 的上电特性：MD24008 $MC_CHSFRAME_POWERON_MASK，位 1 = <值>

值	含义 **外部零点偏移的系统框架 $P_EXTFR 在上电时：**
0	未删除
1	删除

4.4.8 轴专用叠加（$AA_OFF）

4.4.8.1 功能

通过轴专用系统变量 $AA_OFF[<轴>] 可（例如：同步操作时）为规定的轴指定绝对位置或增量行程。由此产生的运行会与轴通道的运行同时进行。

$AA_OFF[<轴>] = <值>

4.4.8.2 调试

机床数据

设置叠加运动

通过轴专用机床数据可设置轴的叠加运动：

36750 $MA_AA_OFF_MODE，位<n> = <值>

位	值	含义
0	0	$AA_OFF 值编译为**绝对位置**
	1	$AA_OFF 值编译为**增量位置**
1	0	通道复位时撤销叠加运动。
	1	通道复位后保留叠加运动。
2	0	在 JOG 运行方式下，**不**移出叠加运动。
	1	在 JOG 运行方式下，移出叠加运动。
3	0	NC 停止时中断叠加运动。
	1	NC 停止时**不**中断叠加运动。

系统变量

轴叠加的集成行程

通过轴专用系统变量可以读取集成的叠加运动值。

<值> = $AA_OFF_VAL[<轴>]

4.4.8.3 编程：针对轴取消叠加（CORROF）

通过程序 CORROF 将下列轴专用叠加删除：

- 通过手轮运行设置的累加零点偏移（DRF 偏移）
- 通过系统变量 $AA_OFF 编写的位置偏移

删除叠加值会触发预处理停止，并将取消的叠加运动的位置分量接收到基准坐标系中的位置。在此情形下不运行轴。

可通过系统变量 $AA_IM（轴的当前 **MCS** 设定值）读取的**机床**坐标系中的位置值**不变**。

可通过系统变量 $AA_IW（轴的当前 **WCS** 设定值）读取的**工件**坐标系中的位置值**改变**，因为其包含叠加运动的被取消的分量。

说明

在 **NC 程序**中允许编写 CORROF。
在**同步动作**中**不**允许编写 CORROF。

句法

```
CORROF(<轴>,"<字符串>"[,<轴>,"<字符串>"])
```

含义

CORROF：	用于取消轴的下列偏移或叠加的程序： • DRF 偏移 • 位置偏移（$AA_OFF）	
	生效方式：	模态
<轴>：	轴名称（通道轴，几何轴或者加工轴名称）	
	数据类型：	AXIS
<字符串>：	用于定义叠加类型的字符串	
	数据类型：	BOOL
	值	**含义**
	DRF	DRF 偏移
	AA_OFF	位置偏移（$AA_OFF）

示例

示例 1：针对轴取消 DRF 偏移（1）

通过 DRF 手轮运行产生 X 轴上的 DRF 偏移。对于该通道中的所有其它轴，DRF 偏移不生效。

程序代码	注释
N10 CORROF(X,"DRF")	；此处，CORROF 作用如同 DRFOF。
...	

示例 2：针对轴取消 DRF 偏移（2）

通过 DRF 手轮运行产生了 X 轴和 Y 轴上的 DRF 偏移。对于该通道中的所有其它轴，DRF 偏移不生效。

程序代码	注释
；仅取消 X 轴的 DRF 偏移，保留 Y 轴的 DRF 偏移。	
；如果是 DRFOF，则两个偏移均被取消。	
N10 CORROF(X,"DRF")	
...	

示例 3：针对轴取消 $AA_OFF 位置偏移

程序代码	注释
; 为 X 轴插补一个位置偏移 == 10。	
N10 WHEN TRUE DO $AA_OFF[X] = 10 G4 F5	
...	
; 取消 X 轴的位置偏移：$AA_OFF[X]=0	
; 不运行 X 轴。	
; 位置偏移添加到 X 轴的当前位置。	
N80 CORROF(X,"AA_OFF")	
...	

示例 4：针对轴取消 DRF 偏移和 $AA_OFF 位置偏移（1）

通过 DRF 手轮运行产生 X 轴上的 DRF 偏移。对于该通道中的所有其它轴，DRF 偏移不生效。

程序代码	注释
; 为 X 轴插补为 10 的位置偏移。	
N10 WHEN TRUE DO $AA_OFF[X] = 10 G4 F5	
...	
; 仅取消 X 轴的 DRF 偏移和位置偏移。	
; 保留 Y 轴的 DRF 偏移。	
N70 CORROF(X,"DRF",X,"AA_OFF")	
...	

示例 5：针对轴取消 DRF 偏移和 $AA_OFF 位置偏移（2）

通过 DRF 手轮运行产生 X 轴和 Y 轴上的 DRF 偏移。对于该通道中的所有其它轴，DRF 偏移不生效。

程序代码	注释
; 为 X 轴插补一个位置偏移 == 10。	
N10 WHEN TRUE DO $AA_OFF[X] = 10 G4 F5	
...	
; 取消 Y 轴的 DRF 偏移和 X 轴的位置偏移。	
; 保留 X 轴的 DRF 偏移。	
N70 CORROF(Y,"DRF",X,"AA_OFF")	
...	

其它信息

$AA_OFF_VAL

通过 $AA_OFF 取消位置偏移后，相应轴的系统变量 $AA_OFF_VAL（轴叠加的积分行程）也归零。

运行方式 JOG 下的 $AA_OFF。

在运行方式 JOG 下，通过机床数据 MD36750 $MA_AA_OFF_MODE 使能了该功能后，更改 $AA_OFF 时位置偏移将作为叠加运行插补。

同步动作下的 $AA_OFF

在通过 CORROF(<轴>,"AA_OFF") 取消位置偏移时，若立即重新设置 $AA_OFF 的同步动作（DO $AA_OFF[<轴>]=<值>）生效，则 取消且不重新设置 $AA_OFF，并且显示报警 21660。如果同步动作在取消之后，比如在 CORROF 之后的程序段中才生效， 则 $AA_OFF 置位并插补位置偏移。

自动通道切换

如果一个 CORROF 编程的轴在另一个通道被激活，则在通道中可获得轴和轴交换（前提：MD30552 $MA_AUTO_GET_TYPE > 0），然后位置偏移和/或 DRF 偏移被取消。

4.5 框架

4.5.1 框架类型

框架是包含轴的偏移（TRANS）、精偏（FINE）、旋转（ROT）、镜像（MIRROR）和比例缩放（SCALE）数值的数据结构。

框架激活时，系统会依据框架值通过既定的计算规则对所涉及的轴执行静态坐标转换。

轴框架

轴框架中包含了一根轴的框架值。

几何轴 X 的轴框架数据结构示例：

轴	TRANS	FINE	ROT	MIRROR	SCALE
X	10.0	0.1	**0.0**	0	1

通道专用框架

通道专用框架包含了针对所有通道轴（几何轴、辅助轴和机床轴）的框架值。

旋转（ROT）只应用于几何轴。

通道专用框架仅在定义了框架的通道中生效。

通道专用框架的数据结构示例：

- 几何轴：X, Y, Z
- 辅助轴 A
- 机床轴：AX1

轴	TRANS	FINE	ROT	MIRROR	SCALE
X	10.0	0.1	**0.0**	0	1
Y	0.0	0.0	**0.0**	1	1
Z	0.0	0.0	**45.0**	0	1
A	2.0	0.1	0.0	0	2
AX1	0.0	0.0	0.0	0	0

全局框架

全局框架包含针对所有机床轴的框架值。

全局框架在 NC 的所有通道中生效。

全局框架的数据结构示例：

- 机床轴：AX1, ... AX5

轴	TRANS	FINE	ROT	MIRROR	SCALE
AX1	10.0	0.1	-	0	1
AX2	0.0	0.0	-	1	1
AX3	0.0	0.0	-	0	1
AX4	2.0	0.1	-	0	2
AX5	0.0	0.0	-	1	1

4.5.2 框架分量

4.5.2.1 偏移

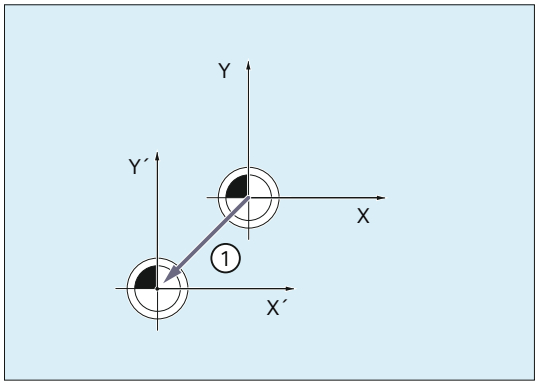
编程

偏移或粗偏可通过以下指令编写：

- 数据管理框架 $P_UIFR 的示例
  - 总框架：$P_UIFR[<n>] = CTRANS(<K1>,<V1>[,<K2>,<V2>][,<K3>,<V3>])
    其中 Km = x、y 或 z 坐标，Vm = 偏移 m
  - 框架分量：$P_UIFR[<n>,<K>,TR] = <V>
    其中 K = x、y 或 z 坐标，V = 偏移
- 可编程框架的示例
  - TRANS <K1> <V1> [<K2> <V2>][<K3> <V3>]
    其中 Km = x、y 或 z 坐标，Vm = 偏移 m

程序示例：

程序代码	注释
$P_UIFR[1] = CTRANS(X,10,Y,10)	总框架
$P_UIFR[1,X,TR] = 10	框架分量
TRANS X=10 Y=10	可编程的框架



① 翻译，精偏

4.5.2.2 精偏

参数设置

精偏通过以下机床数据使能：

MD18600 $MN_MM_FRAME_FINE_TRANS = <值>

值	含义
0	无法输入或编写精偏。
1	可通过操作或程序为可设定框架、基本框架和可编程框架启用精偏。

编程

偏移或粗偏可通过以下指令编写：

- 数据管理框架 $P_UIFR 的示例
  - 总框架：$P_UIFR[<n>] = CFINE(<K1>,<V1>[,<K2>,<V2>][,<K3>,<V3>])
    其中 Km = x、y 或 z 坐标，Vm = 偏移 m
  - 框架分量：$P_UIFR[<n>,<K>,FI] = <V>
    其中 K = x、y 或 z 坐标，V = 偏移
- 可编程框架的示例
  - TRANS <K1> <V1> [<K2> <V2>][<K3> <V3>]
    其中 Km = x、y 或 z 坐标，Vm = 偏移 m

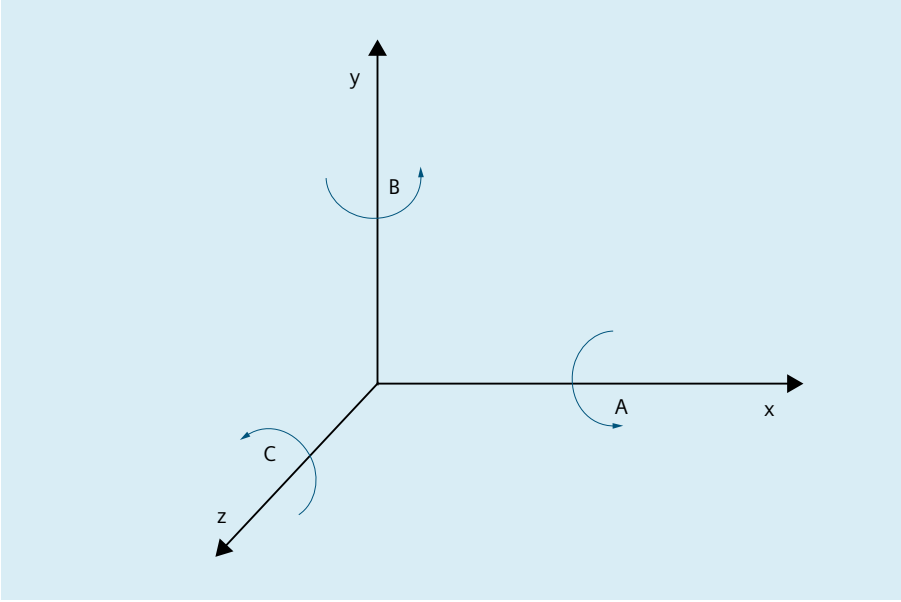
编程示例：

程序代码	注释
$P_UIFR[1] = CTRANS(X,10,Y,10)	总框架
$P_UIFR[1,X,TR] = 10	框架分量
TRANS X=10 Y=10	可编程的框架

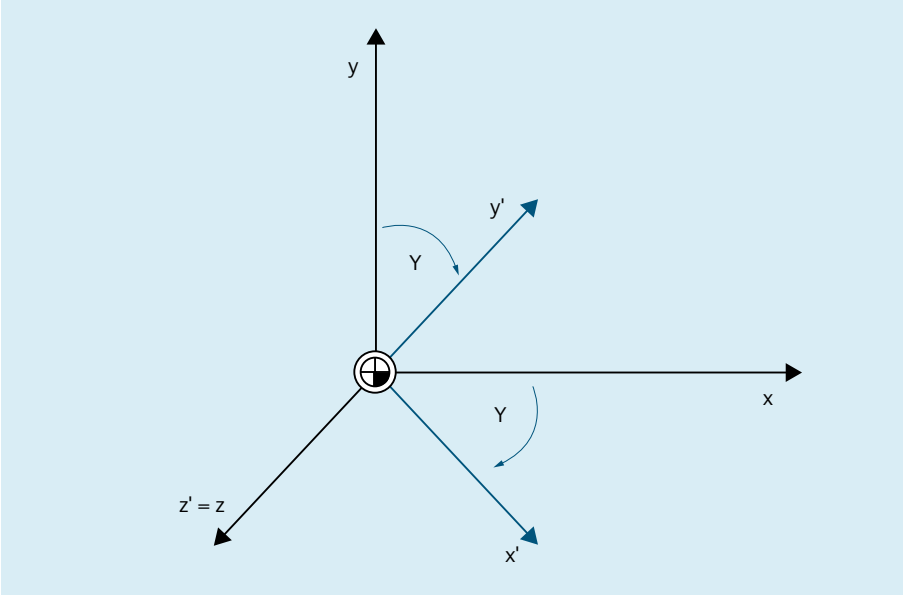
4.5.2.3 旋转：概述（只适用于几何轴）

功能

围绕坐标轴的旋转方向由通过 x、y 和 z 轴构成的右手直角坐标系决定。从坐标轴正方向观察，顺时针旋转时旋转方向为正。A、B 和 C 代表轴平行于坐标轴的旋转。



下图显示了围绕 z 旋转 γ = -45°后坐标系 x'、y' 和 z' 的新位置。



旋转顺序的参数设置

通过以下机床数据，可在编写了超过一个旋转角时设置围绕哪根坐标轴，以及以何种顺序执行旋转：

MD10600 $MN_FRAME_ANGLE_INPUT_MODE = <值>

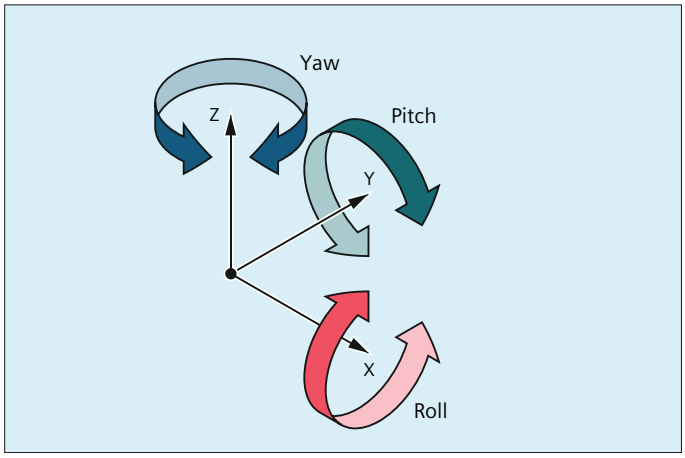
值	含义
1	zy'x" 转换中的欧拉角（RPY 角）
2	zx'z" 转换中的欧拉角

说明

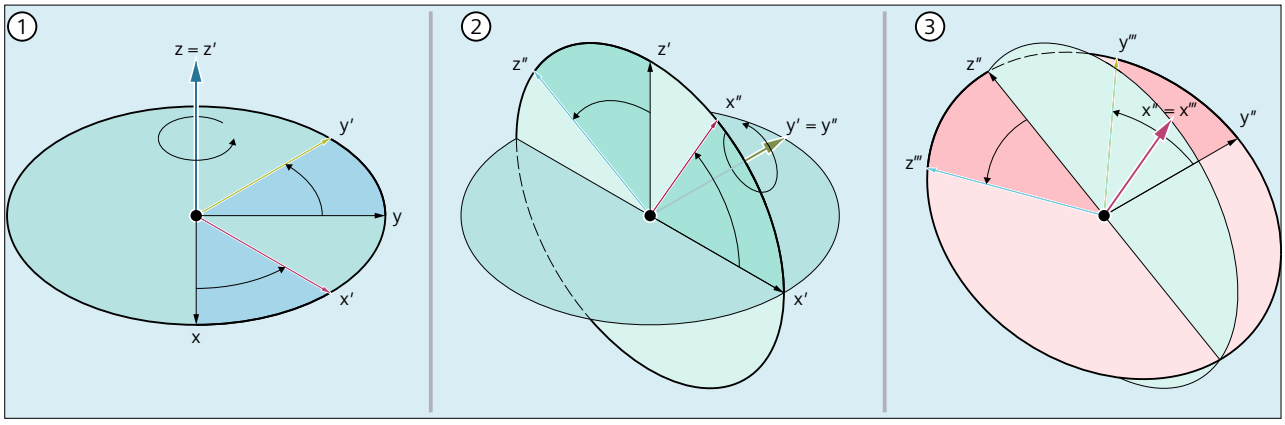
由于历史原因，系统支持 zx'z" 转换中的欧拉角。但是强烈建议只使用 zy'x" 转换中的欧拉角（RPY 角，参见 “通过欧拉角旋转：ZY'X" 转换（RPY 角） (页 352)”章节）。

4.5.2.4 通过欧拉角旋转：ZY'X" 转换（RPY 角）

ZY'X" 转换中的欧拉角也称作 RPY 角。RPY 来源于英语：Roll（横滚）、Pitch（俯仰）和 Yaw（航向）



在 ZY'X" 转换中以如下次序旋转：



① 绕 z 轴旋转

② 绕 y' 轴旋转

③ 绕 x" 轴旋转

取值范围

使用 PRY 角时，仅当编写的值位于以下取值范围内时，才能确保倒推值的唯一性：

$$-18 \le x \le 18$$
$$0 \qquad\qquad 0$$
$$-90 < y < 90$$

$$-18 \le z \le 18$$
$$0 \qquad\qquad 0$$

编程：写入所有旋转分量

在通过 CROT、ROT 或 AROT 编写框架的旋转分量时，总是将所有旋转分量写入。未明确编写的旋转分量会被赋值为 0°。

句法

```
<Frame> = CROT([<1.GeoAx>,<Angle>,][<2.GeoAx>,<Angle>,]
[<3.GeoAx>,<Angle>])
ROT [<1.GeoAx> <Angle>] [<2.GeoAx> <Angle>] [<3.GeoAx> <Angle>]
AROT [<1.GeoAx> <Angle>] [<2.GeoAx> <Angle>] [<3.GeoAx> <Angle>]
```

含义

CROT	绝对旋转
<Frame>	任意生效框架或数据管理框架

ROT	绝对旋转 基准框架：可编程框架 $P_PFRAME， 参考点：使用 G54 ... G57，G505 ... G599 设置的当前工件坐标系的零点
AROT	附加旋转 基准框架：可编程框架 $P_PFRAME， 参考点：使用 G54 ... G57，G505 ... G599 设置的当前工件坐标系的零点
<1.GeoAx> <2.GeoAx> <3.GeoAx>	几何轴 n 的名称，围绕该轴以给定的角度旋转 对于未编程的几何轴，旋转角度隐性设为 0°。 几何轴和旋转轴的指定关系：

几何轴	**旋转轴**
第 1 几何轴	x''
第 2 几何轴	y'
第 3 几何轴	z

<Angle>	角度设定，以度为单位
[...]	方括号中的内容选填。

编程：写入旋转分量

在明确编写框架的旋转分量时，仅编写的旋转分量会被写入。未编写的分量保持不变。

句法

```
<Frame>[<Index>,<GeoAx>,RT] = <Angle>
```

含义

<Frame>:	任意生效框架或数据管理框架
<Index>:	框架的数组下标，例如 $P_UIFR[0 ... n]
<GAx>:	几何轴的名称，围绕该轴以给定的角度旋转
RT:	旋转 (**RoT**ation) 的关键字
<Angle>	角度设定，以度为单位

回读旋转分量

通常情形下，回读的框架旋转分量与编写的值相同：

编程	读回时的值		
	x, RT	y, RT	z, RT
<Frame> = CROT(X,45,Y,30,Z,-20)	45	30	-20

值超出取值范围

若编写的值超出取值范围，则会映射至区域限值：

编程	读回时的值		
	x, RT	y, RT	z, RT
<Frame> = CROT(X,190,Y,0,Z,-200)	-170	0	160

说明

建议在写入框架旋转分量时遵循给定的取值范围，从而能在回读旋转分量时重新获取相同的值。

万向节死锁（Gimbal-Lock）

万向节死锁（Gimbal-Lock）指的是无法从位置矢量倒推出旋转分量的唯一值这一几何问题。对于 RPY 角，旋转分量 y 的角度设置 = 90°时便会出现万向节死锁。在此情形下，控制系统会在写入后对旋转分量进行换算，从而使：

- 旋转分量 z = 旋转分量 z - 旋转分量 x
- 旋转分量 x = 0°
- 旋转分量 y = 90°

编程	读回时的值		
	x, RT	y, RT	z, RT
<Frame> = CROT(X,30,Y,90,Z,40)	0	90	40 - 30 = 10

⚠ 小心

回读旋转分量 z 的不同数值

由于换算时间点不同，在写入**总框架**，或**数据管理框架**的**单个旋转分量**，或**生效框架**的**单个旋转分量**后，为旋转分量 z 回读的值有可能不同。

写入总框架和框架分量时的区别

在写入框架的旋转分量时，须区分两种情形：

1. 写入总框架：<Frame> = CROT(X,a,Y,b,Z,c)
   写入总框架时，系统会在写入时间点立即进行换算。
2. 写入各旋转分量，例如：围绕 X 旋转：<Frame>[0,X,RT]= <a>
   写入单个旋转分量时，换算取决于框架的存储位置：
   - 数据管理框架
     对于数据管理框架，换算基于目前为止写入的旋转分量在框架激活的时间点进行。就旋转分量换算方面而言，写入单个旋转分量后数据管理框架的特性与写入总框架时相似。
   - 生效框架
     对于生效的框架，换算在写入旋转分量的时间点立即进行。

示例

- 写入总框架
  写入**总框架**后，在**每个程序段中**进行**换算**。

编程	读回时的值		
	x, RT	**y, RT**	**z, RT**
N10 <Frame> = CROT(X,0,Y,90,Z,90)	0	90	90
N20 <Frame> = CROT(X,90,Y,90)	0	90	**-90** 1)
N30 <Frame> = CROT(X,90,Y,90,Z,90)	0	90	**0** 1)
1) 与写入**生效框架**的**各旋转分量**相比，数值不同			

- 写入**数据管理框架**的各旋转分量
  在数据管理架激活时进行**换算**。例如在 N30 后的任意一个时间点。

编程	读回时的值		
	x, RT	**y, RT**	**z, RT**
N10 <数据管理框架>[0,X,RT] = 0 N20 <数据管理框架>[0,Y,RT] = 90 N30 <数据管理框架>[0,Z,RT] = 90	0	90	90
N10 <数据管理框架>[0,X,RT] = 90 N20 <数据管理框架>[0,Y,RT] = 90 N30 <数据管理框架>[0,Z,RT] = 0	0	90	**-90** 1)
N10 <数据管理框架>[0,X,RT] = 90 N20 <数据管理框架>[0,Y,RT] = 90 N30 <数据管理框架>[0,Z,RT] = 90	0	90	**0** 1)
1) 与写入**生效框架**的**各旋转分量**相比，数值不同			

- 写入**生效框架**的各旋转分量
  写入相应旋转分量时立刻进行可能需要的**换算**。
  保存的生效框架的输出值为：x = 0, y = 0, z = 0。

编程	读回时的值		
	x, RT	y, RT	z, RT
N10 <生效框架>[0,X,RT] = 0 N20 <生效框架>[0,Y,RT] = 90 N30 <生效框架>[0,Z,RT] = 90	0	90	90
N10 <生效框架>[0,X,RT] = 90 N20 <生效框架>[0,Y,RT] = 90 N30 <生效框架>[0,Z,RT] = 0	0	90	**0** 1)
N10 <生效框架>[0,X,RT] = 90 N20 <生效框架>[0,Y,RT] = 90 N30 <生效框架>[0,Z,RT] = 90	0	90	**90** 1)
1) 与写入**总框架**或写入**数据管理框架**的**单个旋转分量**相比，数值不同			

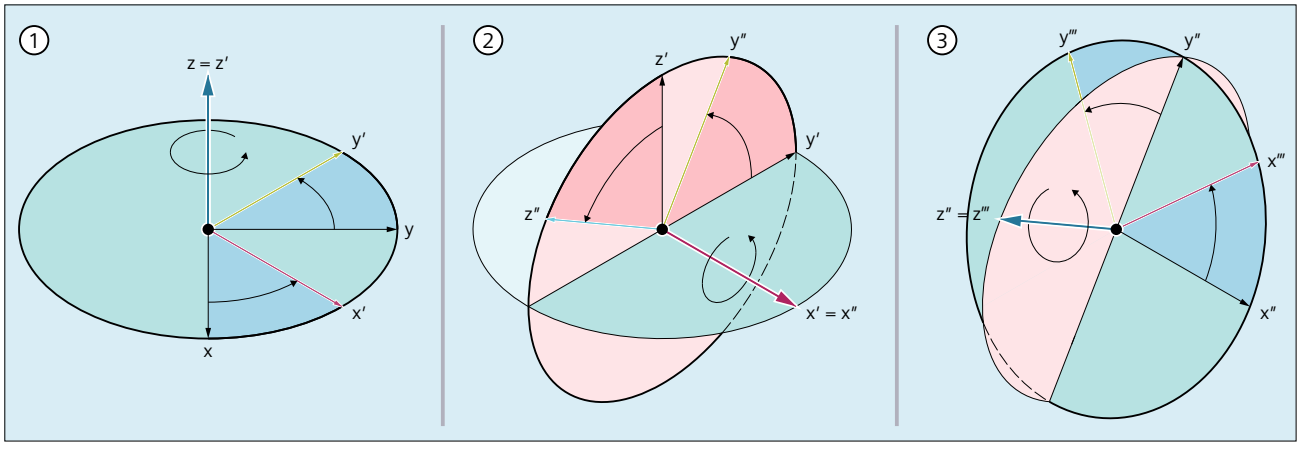
4.5.2.5 通过欧拉角旋转：ZX'Z" 转换

使用欧拉角以 z、x'、z" 的顺序进行旋转。

说明

使用建议

由于历史原因，系统支持 zx'z" 转换中的欧拉角。但是强烈建议只使用 zy'x" 转换中的欧拉角（RPY 角）（参见章节“通过欧拉角旋转：ZY'X" 转换（RPY 角） (页 352)”）。



① 绕 z 轴旋转

② 绕 x' 轴旋转

③ 绕 z" 轴旋转

旋转轴和几何轴的指定关系

旋转轴	通道中的几何轴
z	第 3 几何轴
x'	第 1 几何轴
z''	第 3 几何轴

取值范围

仅当欧拉角的设定处于以下取值范围内时，才能确保倒推出唯一值：

$$\begin{array}{ccccc} 0 & \leq & x & < & 180 \\ & & & & 0 \\ -180 & \leq & y & \leq & 180 \\ 0 & & & & 0 \\ -180 & \leq & z & \leq & 180 \\ 0 & & & & 0 \end{array}$$

设定值位于取值范围以外时，系统会基于限值进行模数换算。

说明

建议在写入框架旋转分量时遵循给定的取值范围，从而能在回读旋转分量时重新获取相同的值。

4.5.2.6 在任意平面中旋转

CRPL - Constant Rotation Plane（恒定旋转平面）

通过预定义功能“**C**onstant **R**otation **Pl**ane（恒定旋转平面）”，可为框架编写任意平面（G17、G18、G19）中的旋转，且无需指定几何轴的名称。 这样一来，在通道基于特殊机床配置只有两根几何轴的情形下，也可编写第三个平面中的旋转。

句法

```
CRPL(<旋转轴>,<旋转角度>)
```

含义

CRPL:	在任意平面中旋转
<旋转轴>:	围绕旋转的轴

类型： INT

值	含义
0	生效平面中的旋转
1	围绕 Z 轴旋转
2	围绕 Y 轴旋转
3	围绕 X 轴旋转

<旋转角度>： 旋转角度，以度为单位

类型： REAL

强烈建议遵循给出的取值范围。若未遵循限值，则无法倒推出唯一值。处于限值外的角度设定不会被系统拒绝。

RPY 角：	X	-180 <= <旋转角度> <= 180
	Y	-90 <= <旋转角度> <= 90
	Z	-180 <= <旋转角度> <= 180
ZX'Z" 转换：	X	-180 <= <旋转角度> <= 180
	Y	0 <= <旋转角度> <= 180
	Z	-180 <= <旋转角度> <= 180

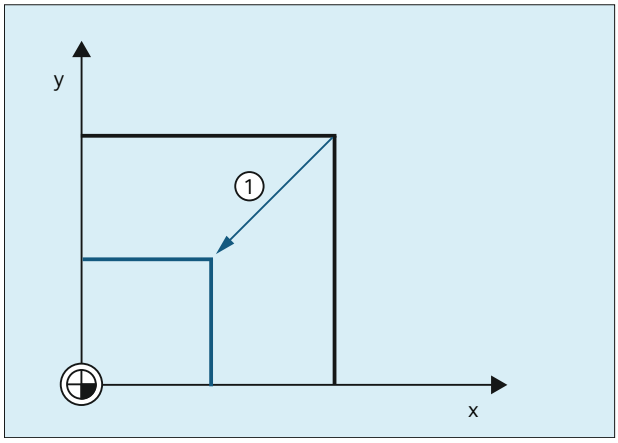
与框架级联

CRPL() 可与框架及 CTRANS()、CROT()、CMIRROR()、CSCALE()、CFINE() 等框架功能级联。

示例：

```
$P_PFRAME = $P_PFRAME : CRPL(0,30.0)
$P_PFRAME = CTRANS(X,10): CRPL(1.30.0)
$P_PFRAME = CROT(X,10): CRPL(2.30.0)
$P_PFRAME = CRPL(3,30.0) : CMIRROR(Y)
```

4.5.2.7 比例缩放



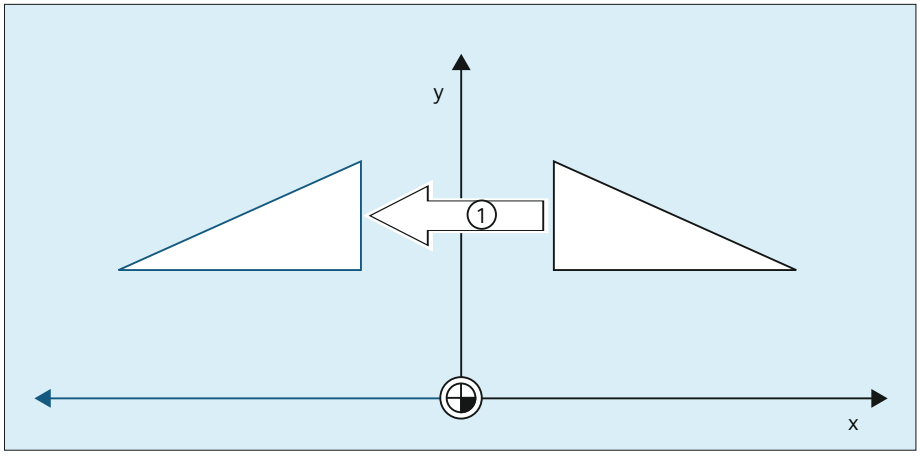
① 比例缩放

编程

通过以下程序指令编写比例缩放：

```
$P_UIFR[1] = CSCALE(x,1,y,1)

SCALE x = 1y = 1

$P_UIFR[1,x,sc] = 1
```

4.5.2.8 镜像



① 镜像

编程

通过以下程序指令编写镜像：

```
$P_UIFR[1] = CMIRROR(x,1,y,1)

MIRROR x = 1y = 1

$P_UIFR[1,x,mi] = 1
```

4.5.2.9 级联运算符

框架分量或总框架可通过级联运算符 (:) 组合成一个整体框架。

4.5.2.10 可编程的轴名称

在框架指令中可使用几何轴、通道轴和机床轴名称。 在通道专用框架中，编写的轴必须为系统已知。

SPI

编写框架指令时，轴名称的位置也可使用轴功能 SPI(<主轴编号>)。

此时 SPI(<主轴编号>) 为主轴相对通道轴的参考。
→ 参见 MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[ ]（主轴指定为机床轴）

下列框架指令可通过 SPI(主轴编号) 编写：

```
CTRANS()

CFINE()

CMIRROR ()

CSCALE ()
```

一根主轴只能指定给一根回转轴。因此CROT(..)功能不可通过SPI()编写，因为CROT()只支持几何轴。

在框架回译时，即便在零件程序中通过 SPI(..) 编写了轴名称，对于归属于主轴的轴，系统也总是输出其通道轴名称或机床轴名称。

例如主轴被指定给通道轴"A"，那么编程：

```
N10 $P_UIFR[1] =
CTRANS(SPI(1),33.33,X,1):CSCALE(SPI(1),33.33):CMIRROR(SPI(1))
```

回译时：

```
$P_UIFR[1]=CTRANS(X,1,A,33.33):CSCALE(A,33.33):CMIRROR(A)
```

若在一个框架指令中编写了主轴和指定的轴，则会触发报警 16420“轴 % 多次编写”。

示例：

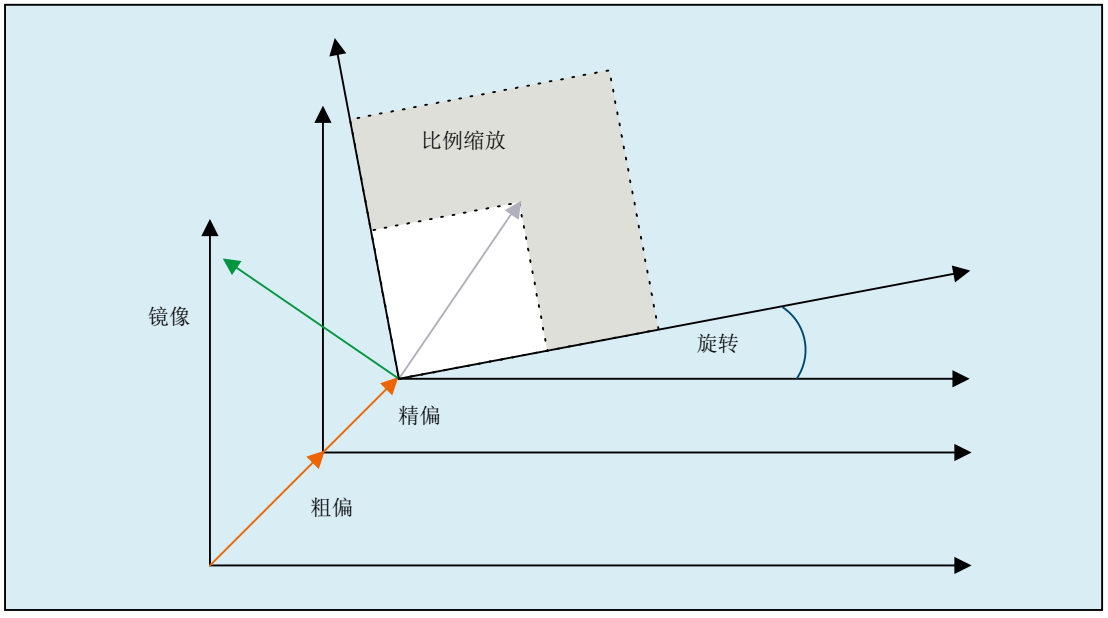
```
$P_UIFR[1] = CTRANS(SPI(1),33.33,X,1,A,44)
```

（主轴被指定给轴 A。）

编程示例

```
$P_PFRAME[SPI(1),TR]=22.22
$P_PFRAME=CTRANS(X,轴值,Y,轴值,SPI(1),轴值)
$P_PFRAME=CSCALE(X,标度,Y,标度,SPI(2),标度)
$P_PFRAME=CMIRROR(S1,Y,Z)
$P_UBFR=CTRANS(A,10):CFINE(SPI(1),0.1)
```

4.5.2.11 坐标转换



几何轴的坐标转换依据以下公式得出：

WCS -> BCS $\vec{v} = R * S * M * \vec{v'} + \vec{t}$

BCS -> WCS $\vec{v'} = inv(M) * inv(S) * inv(R) * (\vec{v} - \vec{t})$

V： BCS 中的位置矢量

V'： WCS 中的位置矢量

4.5.3 数据管理框架和生效框架

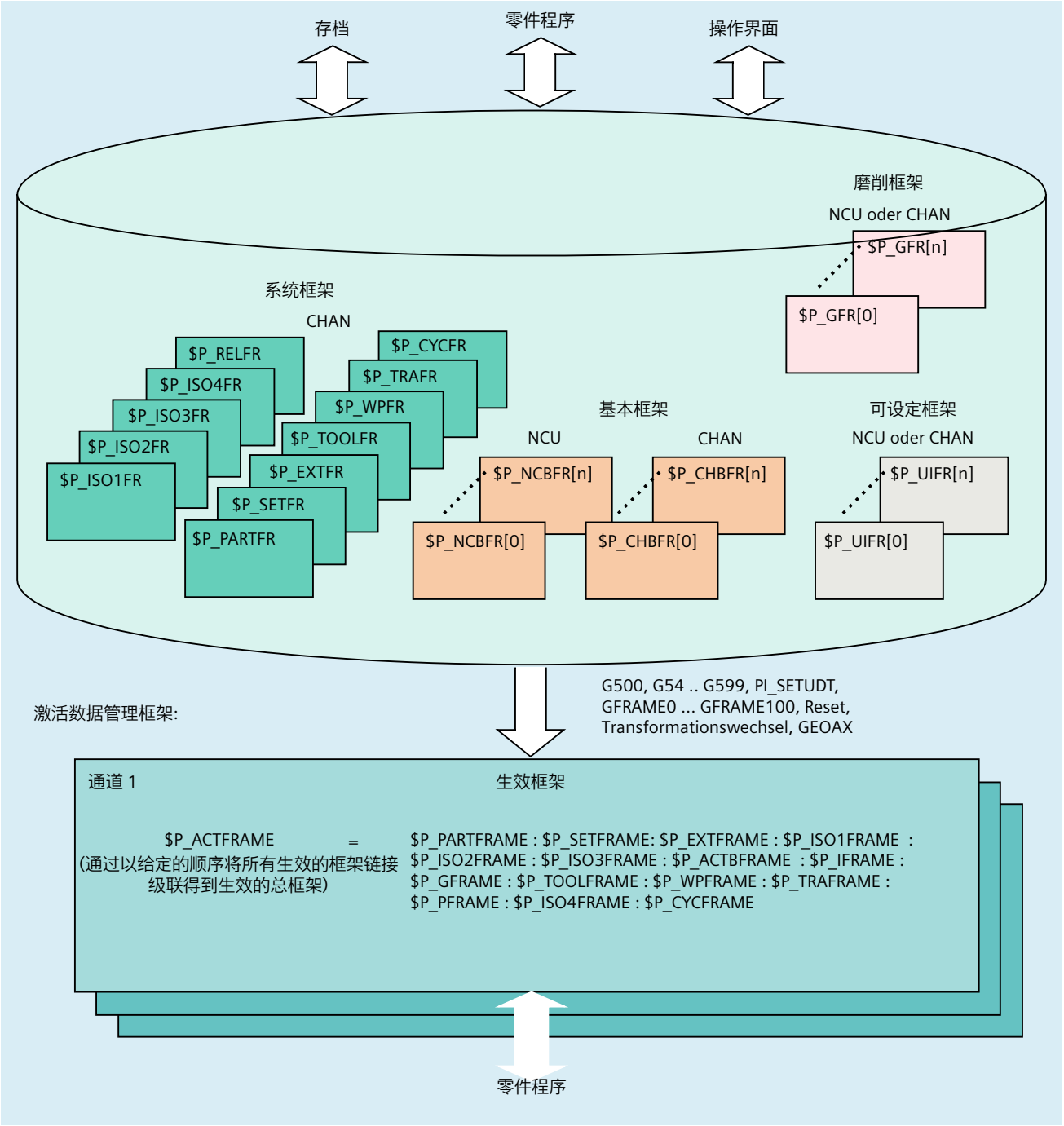
4.5.3.1 概述

框架类型

框架类型有：

- 系统框架 （$P_PARTFR, ... 见图）
- 基本框架 ($P_NCBFR[<n>], $P_CHBFR[<n>])
- 磨削框架 ($P_GFR[<n>])
- 可设定框架 ($P_UIFR[<n>])
- 可编程框架 ($P_PFRAME)

除去可编程框架，对所有框架，除通道中生效的框架外，均还在一个或多个数据管理中的框架（数据管理框架）。对于可编程框架，只存在一个在通道中生效的框架。



写入框架

可通过零件程序写入数据管理框架和生效框架。通过操作界面只可写入数据管理框架。

框架存档

只可存档数据管理框架。

4.5.3.2 激活数据管理框架

数据管理框架通过以下动作转换为生效框架：

- G 指令组“可设定框架”：G54 ... G57, G500, G505 ... G599
- G 指令组“磨削框架”：GFRAME0 ... GFRAME100
- 复位和 MD20110 $MC_RESET_MODE_MASK，位 14 == 1（保留基本框架的当前设置）
- 坐标转换切换
- 几何轴对应关系修改 GEOAX
- 由 HMI 通过 PI 服务 “_N_SETUDT”

由 HMI 激活

仅在当前 NC 程序恢复运行后，由 HMI 通过 PI 服务 “_N_SETUDT” 激活的数据管理框架才在通道中生效。

设置了以下机床数据时，激活在复位状态下立即生效：

MD9440 $MM_ACTIVATE_SEL_USER_DATA（使激活偏移立即生效）

激活系统框架

系统框架可通过以下方式激活：

- 在零件程序中编写对应的系统功能
- SINUMERIK Operate 上的操作

说明

修改数据管理的系统框架

原则上，数据管理的系统框架可由循环编程人员修改，并通过 G500、G54...G599 指令激活。但建议对此操作加以限制，在特定条件下使用。

激活数据管理框架

通过以下机床数据设置数据管理框架激活时的特性：

MD24050 $MC_FRAME_SAA_MODE（数据管理框架的保存和激活）

位	值	含义
0	0	@@@
	1	• 数据管理框架**只**通过编写系统变量 $P_CHBFRMASK、$P_NCBFRMASK 和 $P_CHSFRMASK 生效。 • G500...G599 只激活对应的可设定框架。
1	0	数据管理框架会因 TOROT、PAROT、外部零点偏移、坐标转换这些功能而被隐性赋值。
	1	数据管理框架**不**会因 TOROT、PAROT、外部零点偏移、坐标转换这些功能而被隐性赋值。

通过系统变量 $P_CHSFRMASK 激活系统框架

数据管理的系统框架可通过系统变量 **$P_CHSFRMASK** 激活。变量值以位编码设定，对应机床数据：
MD28082 $MC_MM_SYSTEM_FRAME_MASK（数据管理的系统框架）

通过将系统变量 $P_CHSFRMASK 的一个位设置为 1，使对应的数据管理系统框架在通道中生效。赋值为 0 时，通道中当前生效的系统框架继续生效。

复位后激活系统框架

依据下列机床数据中置位的位，复位后在通道中激活对应系统框架：

MD24006 $MC_CHSFRAME_RESET_MASK（复位后生效的系统框架）

激活 TCARR、PAROT 和 TOROT、TOFRAME 的系统框架

TCARR、PAROT 和 TOROT、TOFRAME 的系统框架根据以下机床数据中的设置激活：

MD20150 $MC_GCODE_RESET_VALUES（G 功能组的初始设置）

在选择/取消坐标转换或 GEOAX 指令使几何轴切换时，系统会删除当前总框架 $P_ACTFRAME，或依据新的几何轴配置重新计算和激活。系统框架和所有其他框架一样根据几何轴重新准备。

4.5.3.3 NCU 全局和通道专用框架

- 可设定框架和磨削框架只可配置为 NCU 全局型**或**通道专用型。
- 基本框架可配置为 NCU 全局型**或**通道专用型。
- NCU 全局框架在 NCU 的所有通道中生效。
- NCU 的所有通道都可读写 NCU 全局框架。
- 在所有通道中，机床轴对通道轴的指定关系，特别是机床轴对几何轴的指定关系可能不同，因此通道轴之间不存在几何关联。因此，NCU 全局框架中只可进行偏移、缩放和镜像。不可旋转。

说明

程序协调

用户自行负责协调 NCU 全局框架的访问。为此，建议使用程序协调指令。

更多信息

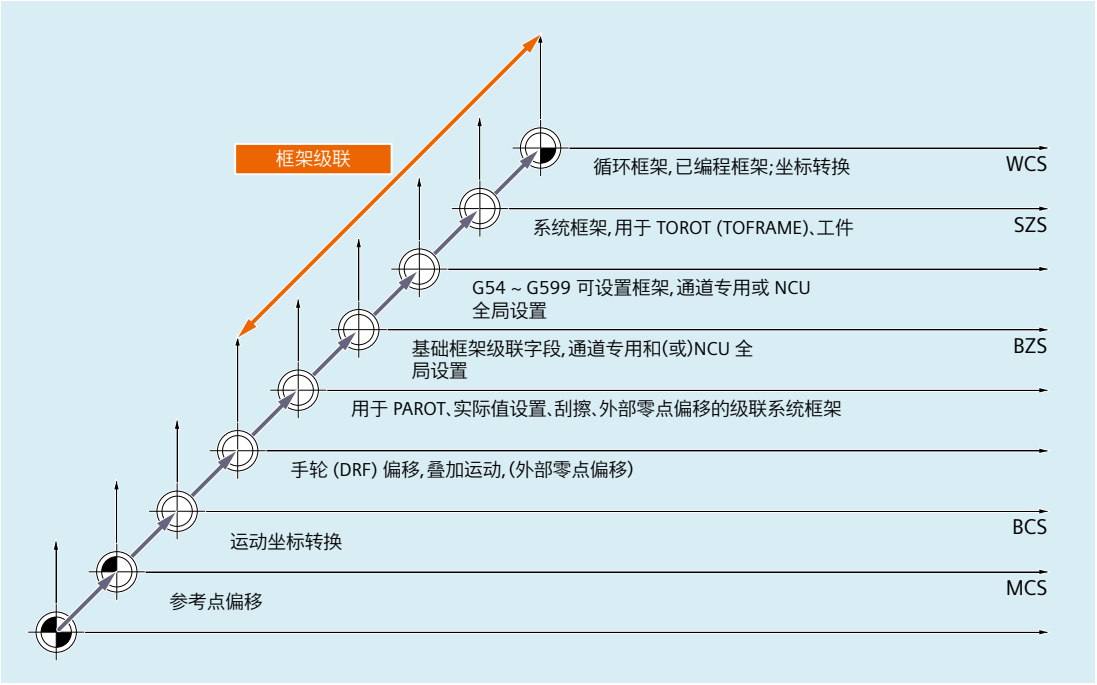
NC 编程手册

4.5.4 框架链和坐标系

4.5.4.1 概述

下图中显示了当前总框架的框架链。框架链位于基本坐标系（BCS）和工件坐标系（WCS）之间。

可设定的零点坐标系（SZS）等同于通过可编程框架转换的 WCS。基本零点坐标系（BZS）还包含当前的可设定框架。仅当进行了配置时，外部零点偏移的系统框架才存在，否则外部零点偏移会被作为轴的叠加运动运行。



WCS **W**orkpiece **C**oordinate **S**ystem，工件坐标系

:

SZS **S**ettable **Z**ero **S**ystem，可设定的零点坐标系

:

BZS **B**asic **Z**ero **S**ystem，基本零点坐标系

:

BCS **B**asic **C**oordinate **S**ystem，基本坐标系

:

MCS **M**aschine **C**oordinate **S**ystem，机床坐标系

:

总框架

当前的总框架 $P_ACTFRAME 由框架链的所有生效框架级联得出：

```
$P_ACTFRAME =   $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :
                $P_ISO1FRAME : $P_ISO2FRAME : $P_ISO3FRAME :
                $P_ACTBFRAME : $P_IFRAME : $P_GFRAME :
                $P_TOOLFRAME :$P_WPFRAME : $P_TRAFRAME :
                $P_PFRAME :$P_ISO4FRAME : $P_CYCFRAME
```

4.5.4.2 相对坐标系

相对坐标系的用处是：相对当前显示的坐标系中设定的参考点，显示当前的轴设定位置。相对坐标系方面无相关编程。只能在该系统中通过系统变量读取轴位置。

新的显示坐标系是相对于 WCS 和 SZS 的坐标系，其通过以下方式得到：使用生效的系统框架 $P_RELFRAME 对 WCS 或 SZS 轴位置进行坐标转换。相对坐标系不仅可以线性平移，还可旋转、镜像、压缩或扩展。

轴设定值的位置显示在 WCS 或 SZS 中进行。配置通过 HMI 机床数据进行。通道中始终只能有一个显示坐标系生效。因此只有一个相对框架可供使用，用于以相同的比例生成两个相对坐标系。HMI 依据配置显示相对坐标。

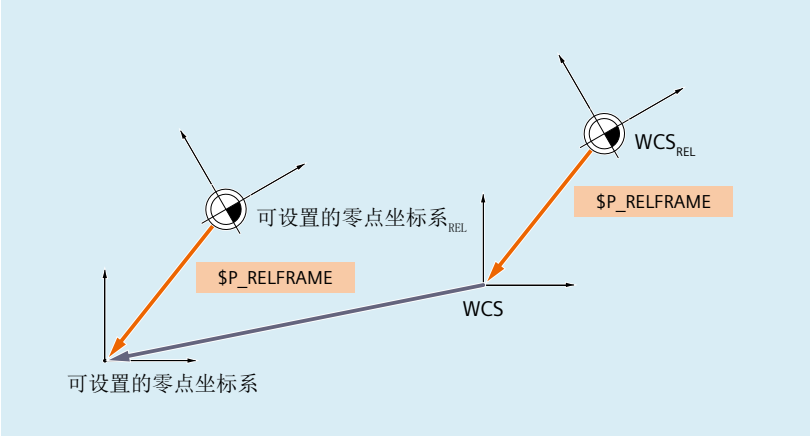


图 4-20　相对坐标系

通过 MD51036 $MNS_ENABLE_COORDINATE_REL=1 激活“相对坐标系”功能。

数据管理框架 $P_RELFR 可在零件程序中或通过 OPI 赋值。所有框架分量均可修改。

生效的系统框架 $P_RELFRAME 可在零件程序中赋值，并通过 OPI 读取。

系统框架 $P_RELFR 通过以下机床数据进行配置：

机床数据	位	含义
MD28082 $MC_MM_SYSTEM_FRAME_MASK	11	创建 $P_RELFR；相对坐标系将因此存在
MD28083 $MC_MM_SYSTEM_DATAFRAME_MASK	11	数据管理框架 $P_RELFR
MD24006 $MC_CHSFRAME_RESET_MASK.	11	$P_RELFR 在复位时生效
MD24007 $MC_CHSFRAME_RESET_CLEAR_MASK	11	$P_RELFR 在复位时删除
MD24008 $MC_CHSFRAME_POWERON_MASK	11	$P_RELFR 在上电时删除

相对坐标系 WCS$_{相对}$ 中的轴位置可通过变量 $AA_PCS_REL[轴] 读取。该变量可在零件程序中、OPI 上、以及通过同步动作读取。

相对坐标系 SZS $_{相对}$ 中的轴位置可通过变量 $AA_ACS_REL[轴] 读取。该变量可在零件程序中、OPI 上、以及通过同步动作读取。

通过操作界面设置相对参考点时，使用用于工件和刀具测量的通用指令接口进行设置。相对坐标系的系统变量 $P_RELFR 如下计算和激活：

- $AC_MEAS_TYPE = 14
- PI 通讯 _N_SETUDT(6, 7)

相对轴位置的设置示例请见：

更多信息

功能手册之工艺；测量（M5）

4.5.4.3 可选 SZS

在循环专用工件坐标系 (WCS) 中进行加工。此时，循环专用 WCS 是通过为循环编程的可编程框架 $P_PFRAME 和/或循环框架 $P_CYCFRAME 从 SZS 转换而来的。

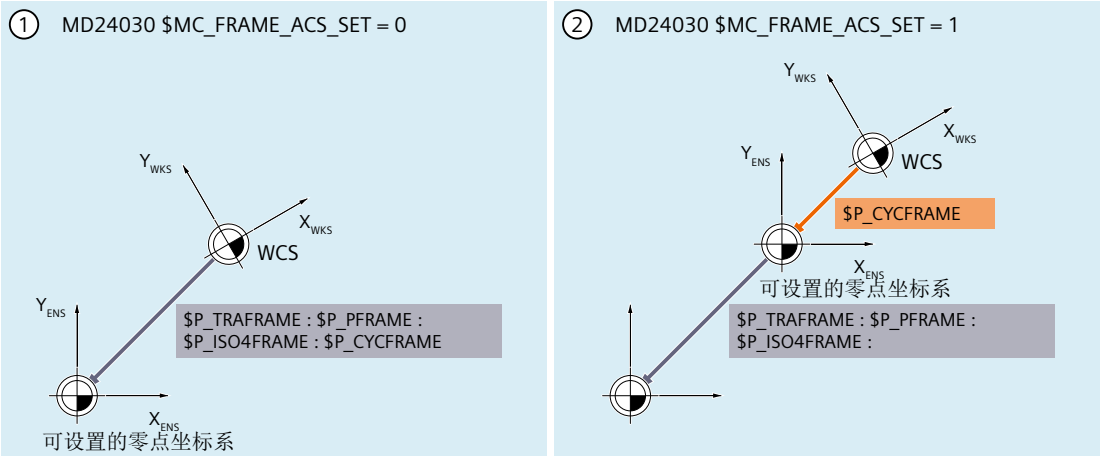
如果机床操作人员通过 NC 停止中断了循环，在激活循环前应运行有效的坐标系 SZS。

机床数据

可通过以下机床数据设置如何从循环专用 WCS 计算 SZS。

MD24030 $MC_FRAME_ACS_SET = <值>

<值>	含义
0	SZS = 通过 $P_TRAFRAME、$P_PFRAME、$P_ISO4FRAME 和 $P_CYCFRAME 转换的 WCS
1	SZS = 只通过 $P_CYCFRAME 转换的 WCS



① SZS = 通过 $P_TRAFRAME、$P_PFRAME、$P_ISO4FRAME 和 $P_CYCFRAME 转换的 WCS

② SZS = 只通过 $P_CYCFRAME 转换的 WCS

影响

SZS 的重新配置会影响：

- SZS 相关的实际值实际值显示，系统变量（例如： $AA_IEN 等）
- 在 SZS 中手动运行 (JOG) 几何轴

4.5.4.4 选择在 WCS 或 SZS 中手动运行几何轴 ($AC_JOG_COORD)

目前在 JOG 运行方式下执行手动运行时，几何轴是在 WCS 中运行。此外也可在 SZS 坐标系中执行手动运行。为此，可使用变量 $AC_JOG_COORD 在 WCS 中的手动运行和 SZS 中的手动运行之间进行切换。用户可选择在 SZS 或 WCS 中运行。

在 JOG 运行方式下，可选择在工件坐标系 (WCS) 或可设定零点坐标系 (SZS) 中手动运行几何轴。

系统变量

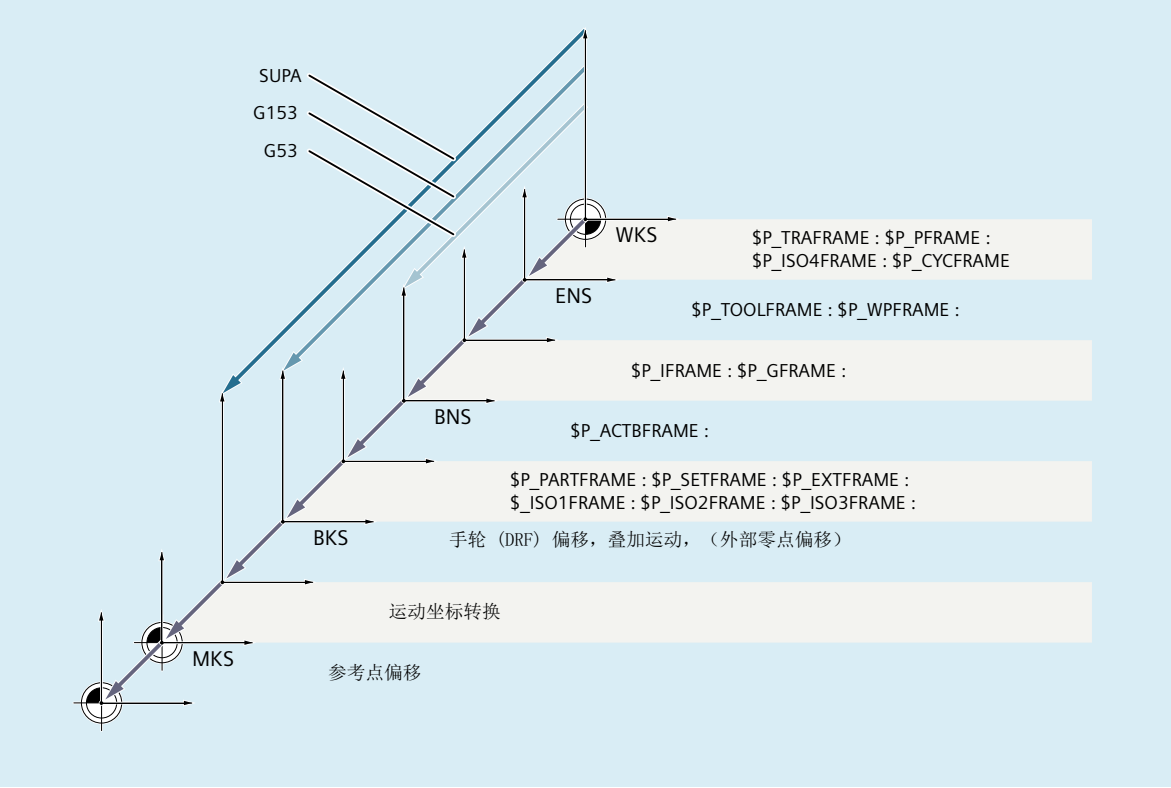
在 JOG 运行方式下，可选择在工件坐标系 (WCS) 或可设定零点坐标系 (SZS) 中手动运行几何轴。通过系统变量 $AC_JOG_COORD 选择：

$AC_JOG_COORD = <值>

<值>	含义
0	工件坐标系 (WCS)
1	可设定零点坐标系 (SZS)

4.5.4.5 抑制框架

通过下述指令 G53、G135 和 SUPA 抑制框架。激活框架抑制会导致与 WCS、SZS 或 BZS 相关的系统变量中的位置显示 (HMI) 及位置数据跳动。该特性可通过机床数据设置。



机床数据

通过以下机床数据确定系统变量中的位置显示 (HMI) 和位置数据的特性：

MD24020 $MC_FRAME_SUPPRESS_MODE，位 <n> = <值>（框架抑制时的位置）

位	值	含义
0	0	位置数据 (OPI)，**带**框架抑制 1)
	1	位置数据 (OPI)，**不带**框架抑制 2)
1	0	系统变量中的位置数据，**带**框架抑制 1)
	1	系统变量中的位置数据，**不带**框架抑制 2)
1) 位置值跳动 2) 位置值**不**跳动		

编程

指令	含义
G53：	逐段抑制下列框架： $P_TRAFRAME : $P_PFRAME : $P_ISO4FRAME : $P_CYCFRAME $P_IFRAME : $P_GFRAME : $P_TOOLFRAME : $P_WPFRAME :
G153：	逐段抑制 G53 时的框架及下列框架： $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME : $P_ACTBFRAME $P_ISO1FRAME : $P_ISO2FRAME : $P_ISO3FRAME :
SUPA：	隐性预处理停止并逐段抑制 G53 和 G135 时的框架及下列框架： • 手轮偏移（DRF） • 叠加运动 • 外部零点偏移
G500：	模态激活 G500 框架。G500 框架通常为零框架。
DRFOF：	关闭（取消）手轮偏移 (DRF)

4.5.5 框架链的框架

4.5.5.1 概述

可使用的框架有：

- 可设定框架 (G500, G54 ... G57, G505 ... G599)
- 磨削框架 (GFRAME0 ... GFRAME100)
- 基本框架
- 可编程的框架
- 系统框架

4.5.5.2 可设定框架 $P_UIFR[<n>]

机床数据

通道专用可设定框架

通道专用可设定框架的数量通过以下机床数据设置：

MD28080 $MC_MM_NUM_USER_FRAMES = <数量>

系统变量索引 n = 0, 1, 2, ... <数量> - 1

NCU 全局可设定框架

NCU 全局可设定框架的数量通过以下机床数据设置：

MD18601 $MN_MM_NUM_GLOBAL_USER_FRAMES = <数量>

系统变量索引 n = 0, 1, 2, ... <数量> - 1

如果机床数据的值 > 0，则**不**存在**通道专用**可设定框架。系统不会检测用于设置通道专用可设定框架的机床数据。

可设定框架的 G 功能组的初始设置

通过以下机床数据设置初始位置或通道复位或上电后可设定框架专用的**第 8 个 G 功能组**中的哪些 G 指令生效：

MD20150 $MC_GCODE_RESET_VALUES[**7**] = <值>

值	G 指令
1	G500
2	G54
3	G55
4	G56
5	G57
6	G505
...	...
100	G599

磨削专用的 G 功能组的复位特性

通过以下机床数据设置可设定框架专用的**第 8 个 G 功能组**的复位特性：

MD20152 $MC_GCODE_RESET_MODE[**7**] = <值>

值	含义 **通道复位或零件程序结束后：**
0	可设定框架专用的 G 指令生效，依据 **MD20150**。
1	当前生效的可设定框架专用的 G 指令继续生效。

说明

MD20110 $MC_RESET_MODE_MASK

机床数据 MD20152 $MC_GCODE_RESET_MODE 在以下条件时才被检测：
MD20110 $MC_RESET_MODE_MASK，位 0 == 1

系统变量

$P_UIFR[<n>]（数据管理的可设定框架）

通过系统变量 $P_UIFR[<n>] 可以读取和写入数据管理的可设定框架。在写入数据管理的可设定框架时，新值不会立即在通道中生效。只有在编程了零点偏移 G500,G54...G599 后，才可在通道中激活。

对于 NCU 全局框架，修改过的数据管理的可设定框架在 NCU 的每个执行 G500、G54..G599 指令的通道中生效。

在进行数据备份时会一并备份数据管理中的可设定框架。

$P_IFRAME（生效的可设定框架）

通过系统变量 $P_IFRAME 可以读取和写入通道中**生效的**可设定框架。在写入可设定框架时，新值不会立即在通道中生效。

在 NCU 全局可设定框架中，修改过的可设定框架仅在编程了新值的通道中生效。如果修改过的 NCU 全局可设定框架适用于 NCU 的所有通道，必须同时写入在通道中生效的可设定框架**和**数据管理中相应的可设定框架。

$P_UIFR[<n>] = $P_IFRAME = <新值>

- $P_UIFR[<n>]（数据管理中的可设定框架）
- $P_IFRAME（通道中生效的可设定框架）

要使修改过的可设定框架在另一个通道中生效，还须在该通道中通过相应的指令（例如：G54）进行激活。

$P_UIFRNUM（生效的可设定框架的编号）

通过系统变量 $P_UIFRNUM 可以读取通道中生效的可设定框架的索引 <n>：

通道中生效的可设定框架 $P_IFRAME == $P_UIFR[$P_UIFRNUM]

$P_UIFRNUM	$P_IFRAME == $P_UIFR[<n>]，其中 n =
0	0
1	1
2	2
...	...
99	99

编程

用于激活通道中的可设定框架的指令

通过编程指令 G500，G54...G599 可使通道中的数据管理可设定框架 $P_UIFR[<n>] 生效或者按照与数据管理可设定框架 $P_UIFR[<n>] 相同的方式设置生效的可设定框架 $P_IFRAME：

G<x> ⇒ $P_IFRAME = $P_UIFR[<n>]

指令	生效的可设定框架 $P_IFRAME =
G500	$P_UIFR[0]
G54	$P_UIFR[1]
G55	$P_UIFR[2]
G56	$P_UIFR[3]
G57	$P_UIFR[4]
G505	$P_UIFR[5]
...	...
G599	$P_UIFR[99]

边界条件

通过 HMI / PLC 写入可设定框架

通过 HMI 或 PLC 用户程序只能写入**数据管理**的可设定框架。

4.5.5.3 磨削框架 $P_GFR[<n>]

磨削框架专门用于磨削工艺，提供了附加零点偏移和补偿。还可作用于可设定框架 (页 375) 的零点偏移。

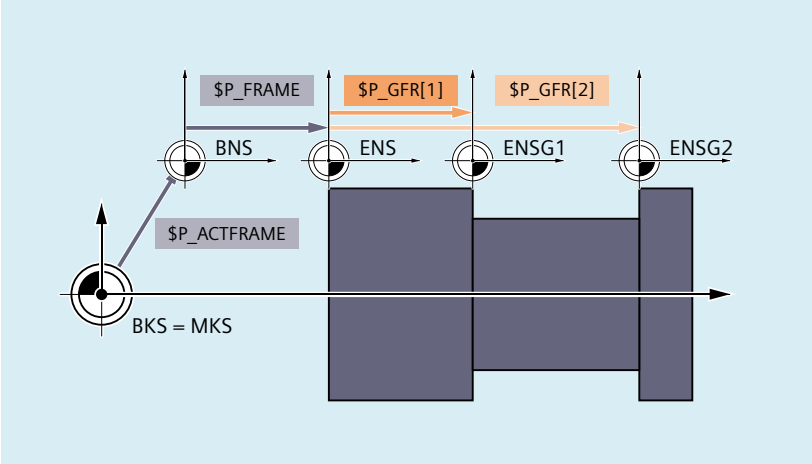


图 4-21 磨削框架

根据基本零点坐标系 (BZS)，由通道中生效的框架的级联得出可设定零点坐标系 (SZS)。

$P_IFRAME : **$P_GFRAME** : $P_TOOLFRAME :$P_WPFRAME

机床数据

通道专用磨削框架的数量

通过以下机床数据设置通道专用磨削框架的数量：

MD28079 $MN_MM_NUM_G_FRAMES = <数量>

其中 <数量> = 0, 1, 2, ... 最大数量

NCU 全局磨削框架的数量

通过以下机床数据设置 NCU 全局磨削框架的数量：

MD18603 $MN_MM_NUM_GLOBAL_G_FRAMES = <数量>

其中 <数量> = 0, 1, 2, ... 最大数量

如果机床数据的值 > 0，则**不**存在**通道专用**磨削框架。系统不会检测用于设置通道专用磨削框架的机床数据。

磨削专用 G 功能组的初始位置 (64)

通过以下机床数据设置初始位置或通道复位或上电后磨削专用的**第 64 个 G 功能组**中的哪些 G 指令生效：

MD20150 $MC_GCODE_RESET_VALUES[**63**] = <值>

值	G 指令
1	GFRAME0（零框架）
2	GFRAME1
3	GFRAME2
...	...
101	GFRAME100

磨削专用的 G 功能组的复位特性

通过以下机床数据设置磨削专用的**第 64 个 G 功能组**的复位特性：

MD20152 $MC_GCODE_RESET_MODE[**63**] = <值>

值	含义 **通道复位或零件程序结束后：**
0	可设定框架专用的 G 指令生效，依据 **MD20150**。
1	当前生效的磨削专用 G 指令继续生效。

说明

MD20110 $MC_RESET_MODE_MASK

机床数据 MD20152 $MC_GCODE_RESET_MODE 在以下条件时才被检测：
MD20110 $MC_RESET_MODE_MASK，位 0 == 1

系统变量

$P_GFR[<n>]（数据管理的磨削框架）

通过系统变量 $P_GFR[<n>] 可以读取和写入数据管理的磨削框架。在写入磨削框架时，新值不会立即在通道中生效。只有在编程了相应的指令 GFRAME0 ... GFRAME100 后，才可在通道中激活。对于 NCU 全局框架，修改过的框架在 NCU 的每个执行 GFRAME0 ... GFRAME100 指令的通道中生效。

在进行数据备份时会一并备份数据管理的磨削框架。

说明

显示（SINUMERIK Operate）

数据管理的磨削框架会显示在 SINUMERIK Operate 操作界面的一个单独的窗口中。

- 更多信息
  磨削版操作手册；章节“设置机床”>“零点偏移”>“显示并编辑基于位置的精偏移”

删除（SINUMERIK Operate）

数据管理的磨削框架可以单独或全部一起，例如杂更换工件后，在 SINUMERIK Operate 操作界面上进行删除。

- 更多信息
  磨削版操作手册；章节“设置机床”>“零点偏移”>“删除基于位置的精偏移”

$P_GFRAME（生效的磨削框架）

通过系统变量 $P_GFRAME 可以读取和写入通道中**生效的**磨削框架。在写入磨削框架时，新值不会立即在通道中生效。

对于 NCU 全局磨削框架，修改过的框架仅在编写了新框架值的通道中生效。

如果修改过的 NCU 全局磨削框架适用于 NCU 的所有通道，必须同时写入在通道中生效的磨削框架 $P_GFRAME 和数据管理中的磨削框架 $P_GFR[<n>]：

$P_GFRAME = $P_GFR[<n>] = <新值>

要使修改过的磨削框架 $P_GFR[<n>] 在另一个通道中生效，还须在该通道中通过相应的指令（例如：GFRAME<n>）进行激活。

$P_GFRNUM（生效的磨削框架的编号）

通过系统变量 $P_GFRNUM 可以读取通道中生效的磨削框架的索引 <n>：

通道中生效的磨削框架 $P_GFRAME == $P_GFR[$P_GFRNUM]

通过 G 指令 GFRAME<n> 激活的磨削框架 $P_GFRAME：	$P_GFRNUM
GFRAME0	0
GFRAME1	1
GFRAME2	2
...	...
GFRAME100	100

编程

用于激活通道中的磨削框架的指令

通过编程指令 GFRAME<n> 使通道中的相应磨削框架 $P_GFR[<n>] 生效。为此，必须按照与磨削框架 $P_GFR[<n>] 相同的方式设置生效的磨削框架 $P_GFRAME：

GFRAME<n> ⇒ $P_GFRAME = $P_GFR[<n>]

指令	通道中生效的磨削框架
GFRAME0	$P_GFR[0]（**零框架**）
GFRAME1	$P_GFR[1]
...	...
GFRAME100	$P_GFR[100]

句法

GFRAME<n>

含义

GFRAME<n>：	激活数据管理的磨削框架 <n>	
	G 功能组：	64
	初始设置：	MD20150 $MC_GCODE_RESET_VALUES[**63**]
	生效方式：	模态
<n>：	磨削框架的编号	
	值域：	0, 1, 2, ... 100

边界条件

通过 HMI / PLC 写入磨削框架

通过 HMI 或 PLC 用户程序只能写入**数据管理**的磨削框架。

4.5.5.4 通道专用基本框架[<n>]

机床数据

通道专用基本框架的数量

通道专用基本框架的数量通过以下机床数据设置：

MD28081 $MC_MM_NUM_BASE_FRAMES = <数量>

系统变量索引 n = 0, 1, 2, ... <数量> - 1

系统变量

$P_CHBFR[<n>]（数据管理的通道专用基本框架）

通过系统变量 $P_CHBFR[<n] 可以读取和写入数据管理的通道专用基本框架。在写入通道专用基本框架时，新值不会立即在通道中生效。只有在编程了相应的指令 G500，G54..G599 后，才可在通道中激活。

在进行数据备份时会一并备份数据管理的通道专用基本框架。

$P_CHBFRAME[<n>]（生效的通道专用基本框架）

通过系统变量 $P_CHBFRAME[<n] 可以读取和写入通道中**生效的**通道专用基本框架。在写入生效的通道专用基本框架时，通过重新计算生效的总基本框架 $P_ACTBFRAME 使新值在通道中立即生效。

出于兼容性原因的系统变量

$P_UBFR（第一个通道专用基本框架）

此变量是 $P_CHBFR[0] 的冗余变量，由于兼容性的原因得以保留。

向预定义变量 $P_UBFR 写入时，数组索引为 0 的基本框架不会被立即激活，而是在执行 G500、G54..G599 这些指令中的一个时才激活。对于 NCU 全局框架，修改过的框架在 NCU 的每个执行 G500、G54..G599 指令的通道中生效。该变量主要用作从 HMI 或 PLC 写入到基本框架时的存储器。此变量也可在程序中读写。

$P_UBFR 与 $P_CHBFR[0] 一致。默认情况下通道中始终有一个基本框架，使得这些系统变量可与较早的版本兼容。如果没有通道专用基准框架，那么在读写时会产生报警“框架：不允许使用的指令”。

$P_BFRAME（第一个生效的通道专用基本框架）

此变量是 $P_CHBFRAME[0] 的冗余变量，由于兼容性的原因得以保留。

通过预定义框架变量 $P_BFRAME，可在零件程序中读写通道中生效的、数组索引为 0 的基本框架。写入的基本框架会被立即计算在内。对于 NCU 全局可设定框架，修改的框架仅在编写了该框架的通道中生效。若需修改某个 NCU 的所有通道的框架，则须同时写入 $P_UBFR 和 $P_BFRAME。然后其它通道必须激活相应框架，例如通过 G54。

$P_BFRAME 与 $P_CHBFRAME[0] 一致。缺省情形下，系统变量始终有一个有效值。如果没有通道专用基准框架，那么在读写时会产生报警“框架：不允许使用的指令”。

边界条件

通过 HMI / PLC 写入基本框架

通过 HMI 或 PLC 用户程序只能写入**数据管理**的基本框架。

4.5.5.5 NCU 全局基本框架 $P_NCBFR[<n>]

机床数据

NCU 全局基本框架的数量

NCU 全局基本框架的数量通过以下机床数据设置：

MD18602 $MN_MM_NUM_GLOBAL_BASE_FRAMES = <数量>

系统变量索引 n = 0, 1, 2, ... <数量> - 1

系统变量

$P_NCBFR[<n>]（数据管理的 NCU 全局基本框架）

通过系统变量 $P_NCBFR[<n] 可以读取和写入数据管理的 NCU 全局基本框架。在写入 NCU 全局基本框架时，新值不会立即在通道中生效。只有在编程了相应的指令 G500,G54..G599 后，才可在通道中激活。

在进行数据备份时会一并备份数据管理的 NCU 全局基本框架。

$P_NCBFRAME[<n>]（当前 NCU 全局基本框架）

通过系统变量 $P_NCBFRAME[<n] 可以读取和写入通道中**生效的** NCU 全局基本框架。在写入生效的 NCU 全局基本框架时，通过重新计算生效的总基本框架 $P_ACTBFRAME 使新值在通道中立即生效。

如果修改过的 NCU 全局基本框架适用于 NCU 的所有通道，必须同时写入在通道中生效的 NCU 全局基本框架和数据管理的 NCU 全局基本框架：

$P_NCBFR[<n>] = $P_NCBFRAME = <新值>

- $P_NCBFR[<n>]（数据管理的 NCU 全局基本框架）
- $P_NCBFRAME（通道中生效的 NCU 全局基本框架）

要使修改过的 NCU 全局基本框架在另一个通道中生效，还须在该通道中通过相应的指令（例如：G500,G54..G599）进行激活。

编程

通过相应的指令 (G54 ... G57, G505 ... G599 und G500) 将数据管理的通道专用可设定框架 $P_UIFR[<n>] 转换为在通道中生效的可设定框架 $P_IFRAME。

指令	激活数据管理的 NCU 全局和通道专用基本框架
G500	$P_CHBFR[0] : $P_NCBFR[0]
G54	$P_CHBFR[1] : $P_NCBFR[1]
G55	$P_CHBFR[2] : $P_NCBFR[2]
G56	$P_CHBFR[3] : $P_NCBFR[3]
G57	$P_CHBFR[4] : $P_NCBFR[4]
G505	$P_CHBFR[5] : $P_NCBFR[5]
...	...
G599	$P_CHBFR[99] : $P_NCBFR[99]

4.5.5.6 生效的总基本框架 $P_ACTBFRAME

功能

总基本框架 $P_ACTBFRAME 中汇总了所有生效的 NCU 全局和通道专用基本框架：

```
$P_ACTBFRAME = $P_NCBFRAME[0] : ... : $P_NCBFRAME[<n>] :
               $P_CHBFRAME[0] : ... : $P_CHBFRAME[<n>]
```

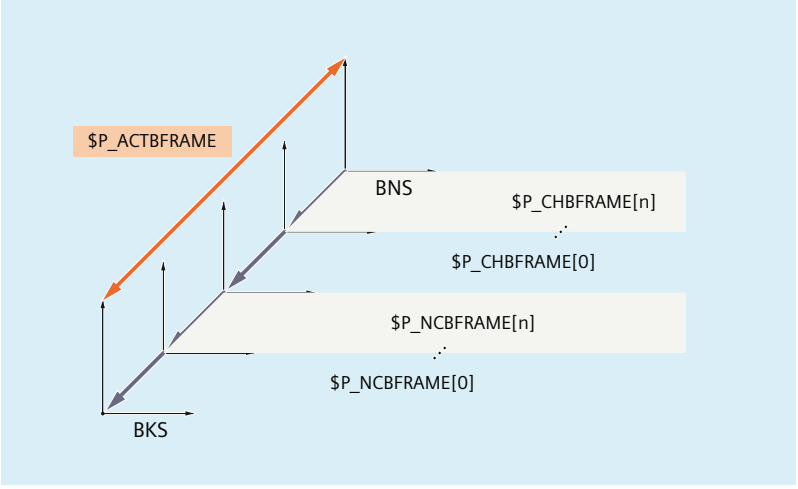


图 4-22 总基准框架

机床数据

复位特性

通过以下机床数据设置复位（通道复位、程序结束复位或上电）后生效的基本框架：

MD20110 $MC_RESET_MODE_MASK，位 0 = 1 和位 14 = 1

- 位 0 = 1：缺省值 ⇒ 设置其他位时的复位特性
- 位 14 = 0：通过复位彻底取消基本框架。
- 位 14 = 1：通过复位将机床数据设置接收到系统变量中且所选基本框架生效：
  - $P_NCBFRMASK = MD10613 $MN_NCBFRAME_RESET_MASK
  - $P_CHBFRMASK = MD24002 $MC_CHBFRAME_RESET_MASK

示例

机床数据设置	生效的基本框架
$P_NCBFRMASK = MD10613 $MN_NCBFRAME_RESET_MASK = 'H81'	$P_NCBFRAME[0] : $P_NCBFRAME[7]

编程

基本框架掩码

通过基本框架掩码 $P_NCBFRMASK 和 $P_CHBFRMASK 选择与总基本框架级联的基本框架。

通过设置基本框架**掩码**中的位选择相应的基本框架：

- $P_NCBFRMASK，位 **0, 1, 2, ... n** ⇒ $P_NCBFRAME[**0, 1, 2, ... n**]
- $P_CHBFRMASK，位 **0, 1, 2, ... n** ⇒ $P_NCHFRAME[**0, 1, 2, ... n**]

基本框架掩码 $P_NCBFRMASK 和 $P_CHBFRMASK 只能在 NC 程序中读取和写入。通过 OPI 可以读取基本框架掩码。

写入基本框架掩码后，系统会重新计算生效的总基本框架 $P_ACTBFRAME 和总框架 $P_ACTFRAME。

示例

程序代码	注释
$P_NCBFRMASK = 'H81'	; 生效的 NCU 全局基本框架：$P_NCBFRAME[0] : $P_NCBFRAME[7]

4.5.5.7 可编程框架 $P_PFRAME

可编程框架只存在生效框架。
此框架是为编程人员预留。

采用以下设置时，可编程框架在复位时保持生效：

MD24010 $MC_PFRAME_RESET_MODE（可编程框架的复位模式）= 1

在复位后需要从斜孔运行出来的情形下，此功能尤为重要。

MIRROR

到目前为止（至软件 P4），机床轴的镜像均以以下机床数据：
MD10610 $MN_MIRROR_REF_AX（镜像的基准轴）
定义的参考轴为基准。

从用户角度看来，此定义难以理解。在进行 z 轴的镜像时，显示中会提示 x 轴已执行镜像，y 轴已旋转 180°。若对两根轴进行镜像，情况将更为复杂，且无法轻易理解哪些轴已执行镜像、哪些轴没有。

从 SW-P5 开始，可通过一种新的方式明确显示轴的镜像。其不再投影为参考轴的镜像和其他轴的旋转。

这可通过以下机床数据设置：

MD10610 $MN_MIRROR_REF_AX = 0

通过 MIRROR 和 AMIRROR 进行的可编程框架编程得到扩展。在此之前，设定的坐标轴的值不会被控制系统检测，例如 MIRROR X0 中的值 0；而 AMIRROR 却是有切换功能，MIRROR X0 能够激活镜像，再次编写 AMIRROR X0 则又将其取消。MIRROR 总是绝对生效，而 AMIRROR 为附加生效。

通过以下机床数据设置：
MD10612 $MN_MIRROR_TOGGLE = 0（“镜像切换”）
可定义对编写的值进行检测。
在值为 0，例如 AMIRROR X0 时，取消该轴的镜像。值不为 0 时，若轴尚未镜像，则对其执行镜像。

对镜像的逐分量读写与以下机床设置无关：

MD10612 $MN_MIRROR_TOGGLE

值 = 0 表示之后轴未经过镜像；值 = 1 表示之后总是进行镜像，与该轴是否已经过镜像无关。

```
$P_NCBFR[0,x,mi]=1          ; x 轴总是执行镜像。
$P_NCBFR[0,x,mi]=0          ; x 轴镜像关闭。
```

轴专用替换 G58，G59

可编程框架的偏移分量划分为绝对分量，以及用于将所有叠加编写的偏移求和的分量。绝对分量可通过 TRANS、CTRANS 或写入偏移分量修改，此时叠加分量会被设置为零。G58 只会修改指定轴的绝对偏移分量，叠加编写的偏移的总和保持不变。

```
G58 X...Y...Z...A... ...
```

G59 用于针对指定轴改写通过 ATRANS 编写的叠加偏移。

```
G59 X...Y...Z...A... ...
```

示例

```
TRANS X10 Y10 Z10
ATRANS X5 Y5                ; 总偏移 X15 Y15 Z10
G58 X20                     ; 总偏移 X25 Y15 Z10
G59 X10 Y10                 ; 总偏移 X30 Y20 Z10
```

仅当满足以下条件时，才可使用 G58 和 G59：

MD24000 $MC_FRAME_ADD_COMPONENTS（G58 / G59 的框架分量）== TRUE

下表中介绍了各种编程指令对绝对偏移和附加偏移的影响。

	粗偏或绝对偏移	精偏或叠加偏移
TRANS X10	10	0
ATRANS X10	不变	alt_fine + 10
CTRANS(X,10)	10	0
CTRANS()	0	0
CFINE(X,10)	0	10
$P_PFRAME[X,TR] = 10	10	不变
$P_PFRAME[X,FI] = 10	不变	10
G58 X10	10	不变
G59 X10	不变	10

4.5.5.8 通道专用系统框架

通道专用系统框架只由系统功能描述，例如实际值设置、对刀、外部零点偏移和倾斜加工。

机床数据

通道专用系统框架的参数设置

由于存储空间原因，建议只配置系统功能所必需的通道专用系统框架。

每个系统框架会在每个通道中占用约 1 kB SRAM 和约 6 kB DRAM。

通过以下机床数据设置通道专用系统框架的参数：

MD28082 $MC_MM_SYSTEM_FRAME_MASK，位 <n>

位	值	现有系统框架：
0	1	$P_SETFR：实际值设置和对刀
1	1	$P_EXTFR：通过系统框架实现外部零点偏移
2	1	$P_PARTFR：使用可定向刀架时的 TCARR 和 PAROT
3	1	$P_TOOLFR：TOROT 和 TOFRAME
4	1	$P_WPFR：工件参考点的框架
5	1	$P_CYCFR：循环的框架
6	1	$P_TRAFR：用于坐标转换选择/取消的框架
7	1	$P_ISO1FRAME：G51.1 镜像（ISO）的框架
8	1	$P_ISO2FRAME：G68 2DROT（ISO）的框架
9	1	$P_ISO3FRAME：G68 3DROT（ISO）的框架
10	1	$P_ISO4FRAME：G51 比例缩放（ISO）的框架
11	1	$P_RELFR：相对坐标系的框架

ACS 坐标系的参数设置

通过以下机床数据确定 ACS 坐标系是由哪些系统框架构成的

MD24030 $MC_FRAME_ACS_SET = <值>

<值>	含义：ACS 坐标系由以下系统框架构成：
0	$P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :$P_ISO1FRAME : $P_ISO2FRAME : $P_ISO3FRAME :$P_ACTBFRAME : $P_IFRAME : $P_GFRAME : $P_TOOLFRAME : **$P_WPFRAME**
1	$P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :$P_ISO1FRAME : $P_ISO2FRAME : $P_ISO3FRAME :$P_ACTBFRAME : $P_IFRAME : $P_GFRAME : $P_TOOLFRAME : **$P_WPFRAME : $P_TRAFRAME :$P_PFRAME : $P_ISO4FRAME**

系统变量

数据管理的通道专用系统框架

数据管理的通道专用系统框架可通过以下框架变量读写：

系统变量	含义：数据管理的系统框架，用于
$P_SETFR	实际值设置和对刀（**Set**-**Fr**ame，设置框架）
$P_EXTFR	外部零点偏移（**Ext**-**Fr**ame，外部框架）
$P_PARTFR	使用可定向刀架时的 TCARR 和 PAROT（**Part**-**Fr**ame，零件框架）
$P_TOOLFR	TOROT 和 TOFRAME（**Tool**-**Fr**ame，刀具框架）
$P_WPFR	工件参考点（**W**ork-**P**iece-**Fr**ame，工件框架）
$P_CYCFR	循环（**Cyc**le-**Fr**ame，循环框架）
$P_TRAFR	坐标转换（(**Tra**nsformation-**Fr**ame，转换框架）
$P_ISO1FR	G51.1 镜像（ISO）
$P_ISO2FR	G68 2DROT (ISO)
$P_ISO3FR	G68 3DROT (ISO)
$P_ISO4FR	G51 比例缩放 (ISO)
$P_RELFR	相对坐标系

说明

循环编程

系统框架的系统变量仅可用于循环编程。因此，在 NC 程序中，系统变量不是由用户直接，而是通过系统功能（例如：TOROT、PAROT 等）写入的。

生效的通道专用系统框架

生效的通道专用系统框架的系统变量：

系统变量	含义：生效的系统框架，用于
$P_SETFRAME	实际值设置和对刀（**Set**-**Fr**ame，设置框架）
$P_EXTFRAME	外部零点偏移（**Ext**-**Fr**ame，外部框架）
$P_PARTFRAME	使用可定向刀架时的 TCARR 和 PAROT（**Part**-**Fr**ame，零件框架）
$P_TOOLFRAME	TOROT 和 TOFRAME（**Tool**-**Fr**ame，刀具框架）
$P_WPFRAME	工件参考点（**W**ork-**P**iece-**Fr**ame，工件框架）
$P_CYCFRAME	循环（**Cyc**le-**Fr**ame，循环框架）

系统变量	含义：生效的系统框架，用于
$P_TRAFRAME	坐标转换（**Tra**nsformation-**Fra**me，转换框架）
$P_ISO1FRAME	G51.1 镜像（ISO）
$P_ISO2FRAME	G68 2DROT (ISO)
$P_ISO3FRAME	G68 3DROT (ISO)
$P_ISO4FRAME	G51 比例缩放 (ISO)
$P_RELFRAME	相对坐标系

如果数据管理的通道专用系统框架未设置参数，系统框架"$P_<系统框架> == 零框架"自动生效。

通道专用的 ACS 总框架

通过系统变量 $P_ACSFRAME 可以读取构成 ACS 坐标系的系统框架。通过上述机床数据 MD24030 $MC_FRAME_ACS_SET 确定。参见章节"机床数据" > "ACS 坐标系的参数设置"

系统变量	含义：生效的系统框架，用于
$P_ACSFRAME	根据机床数据 MD24030 $MC_FRAME_ACS_SET 中的参数设置构成 ACS 坐标系的系统框架。

4.5.6 隐性框架修改

4.5.6.1 切换几何轴

可在激活/取消坐标转换或使用指令 GEOAX() 时切换几何轴。

通过以下机床数据可针对当前总框架 $P_ACTFRAME 的操作进行四种不同的设置：

MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = <值>

<值>	含义
0	删除 几何轴切换时，例如激活和取消坐标转换以及使用 GEOAX() 指令时，删除当前总框架。 激活新的框架后，变更过的几何轴配置才会被启用。
1	通过接收旋转重新计算 几何轴切换时重新计算当前总框架，同时新几何轴的框架分量生效。切换前编写的几何轴旋转同样对新几何轴保持生效。 TRANSMIT、TRACYL 和 TRAANG 的相关信息请见“选择/取消坐标转换：常规 (页 393)”一章。
2	只能在无生效的旋转时重新计算 几何轴切换时重新计算当前总框架，同时新几何轴的框架分量生效。若切换前当前基本框架、当前可设定框架或可编程框架中有生效的旋转，那么切换将被终止，并触发报警“框架：不允许的几何轴切换”。 TRANSMIT、TRACYL 和 TRAANG 的相关信息请见“选择/取消坐标转换：常规 (页 393)”一章。
3	转换：删除 / GEOAX()：通过接收旋转重新计算 • 转换：选择和取消坐标转换时，删除当前总框架。 • GEOAX()：GEOAX() 指令下重新计算总框架，同时新几何轴的偏移、比例缩放和镜像生效。切换前编写的几何轴旋转同样对新几何轴保持生效。

工件几何数据通过几何轴构成的坐标系描述。每根几何轴均被指定一根通道轴，每根通道轴均被指定一根机床轴。针对每个框架（系统框架、基本框架、可设定框架、可编程框架），每根机床轴都有一个轴框架。如果几何轴被指定了另一根机床轴，机床轴会算入其自有轴框架分量。这样一来，通道中的新几何配置将由新的轮廓框架构成，其包含最多三根新几何轴。

几何轴切换时，当前生效的框架会被重新计算，从而得到一个总框架。数据管理框架要在激活后才会被启用。

示例

需通过 GeoAx 切换将通道轴 A 设定为几何轴 X。切换后，可编程框架在 X 轴方向上有 10 的偏移分量。保留当前可设定框架。

MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1

程序代码	注释
`$P_UIFR[1] = CROT(X,10,Y,20,Z,30)`	; GeoAx 切换后保留框架。
`G54`	; 可设定框架生效。

程序代码	注释
TRANS A10	; 切换时附带轴 A 的轴偏移。
GEOAX(1, A)	; 将轴 A 设定为 X 轴
	; $P_ACTFRAME = CROT(X,10,Y,20,Z,30) :CTRANS(X10)

坐标转换切换时，可同时将多个通道轴设定为几何轴。

示例

通过 5 轴定向转换，通道轴 4、5、6 成为转换的几何轴。即在转换前替换所有几何轴。激活转换时，所有当前框架相应变化。为了计算新的 WCS 坐标系，将成为几何轴的通道轴的轴框架分量纳入计算。转换前编写的旋转保留。取消转换后，重新恢复为原先的 WCS。最常见应用就是使几何轴在坐标转换前后不发生变化，并保留转换前生效的框架。

机床数据

程序代码

```
$MC_AXCONF_CHANAX_NAME_TAB[0] = "CAX"
$MC_AXCONF_CHANAX_NAME_TAB[1] = "CAY"
$MC_AXCONF_CHANAX_NAME_TAB[2] = "CAZ"
$MC_AXCONF_CHANAX_NAME_TAB[3] = "A"
$MC_AXCONF_CHANAX_NAME_TAB[4] = "B"
$MC_AXCONF_CHANAX_NAME_TAB[5] = "C"

$MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1
$MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 2
$MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 3

$MC_AXCONF_GEOAX_NAME_TAB[0] = "X"
$MC_AXCONF_GEOAX_NAME_TAB[1] = "Y"
$MC_AXCONF_GEOAX_NAME_TAB[2] = "Z"

$MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=4
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]=5
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]=6

$MC_TRAFO_AXES_IN_1[0]=4
$MC_TRAFO_AXES_IN_1[1]=5
$MC_TRAFO_AXES_IN_1[2]=6
$MC_TRAFO_AXES_IN_1[3]=1
$MC_TRAFO_AXES_IN_1[4]=2
```

程序：

程序代码

```
$P_NCBFRAME[0] = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6)
$P_CHBFRAME[0] = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6)
$P_IFRAME = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6):CROT(Z,45)
$P_PFRAME = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6):CROT(X,10,Y,20,Z,30)
```

程序代码	注释
TRAORI	；坐标转换，设置 GEOAX(4,5,6)
	; $P_NCBFRAME[0] = CTRANS(X,4,Y,5,Z,6,CAX,1,CAY,2,CAZ,3)
	; $P_ACTBFRAME =CTRANS(X,8,Y,10,Z,12,CAX,2,CAY,4,CAZ,6)
	; $P_PFRAME = CTRANS(X,4,Y,5,Z,6,CAX,1,CAY,2,CAZ,3):CROT(X,10,Y,20,Z,30)
	; $P_IFRAME = CTRANS(X,4,Y,5,Z,6,CAX,1,CAY,2,CAZ,3):CROT(Z,45)
TRAFOOF	；取消转换，设置 GEOAX(1,2,3)
	; $P_NCBFRAME[0] = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6)
	; $P_CHBFRAME[0] = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6)
	; $P_IFRAME = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6):CROT(Z,45)
	; $P_PFRAME = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6):CROT(X,10,Y,20,Z,30)

4.5.6.2 选择/取消坐标转换：常规

选择和取消坐标转换时，几何轴对通道轴的指定关系通常会发生变化。在将回转轴切换为直线轴，以及将直线轴切换为回转轴的转换中，无法将轴框架分量明确指定为几何轮廓框架分量。在这些非线性转换中，必须采用特殊措施来处理轮廓框架。

通过 MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1 和 2 设置的模式需要进行扩展，从而将上述转换考虑在内。

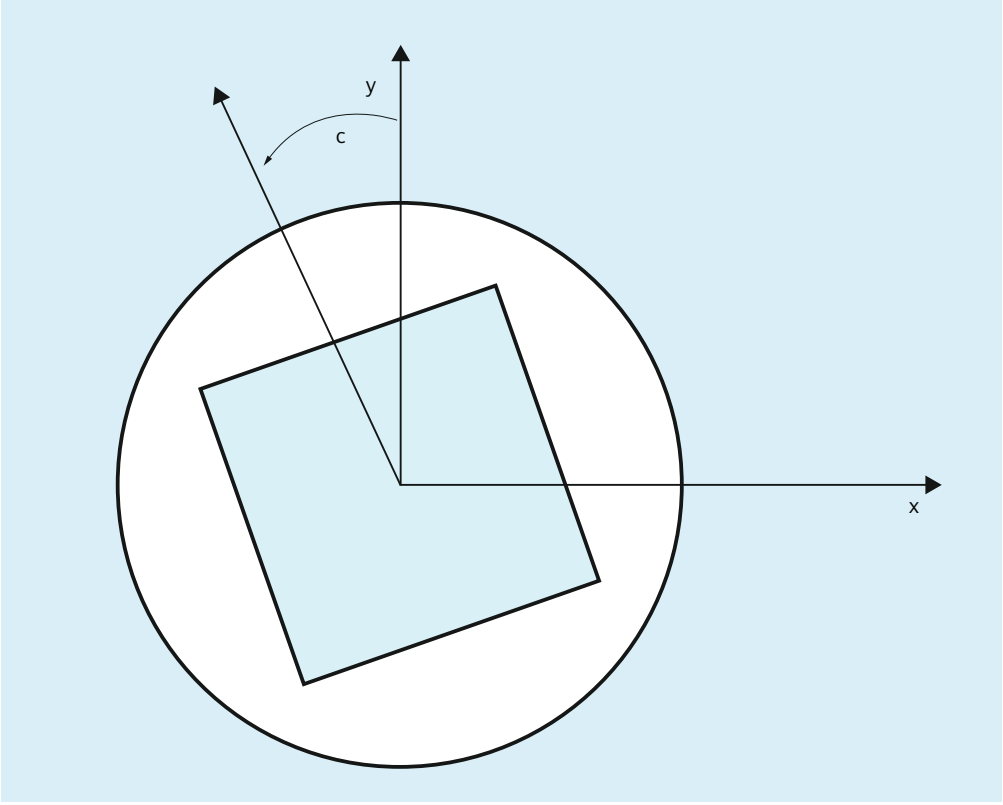
选择转换时，系统基于轴框架建立轮廓框架。此时 TRANSMIT、TRACYL 和 TRAANG 转换中的几何轴会受到特殊处理。

说明

带虚拟轴的转换

在选择 TRANSMIT 或 TRACYL 时，实际 Y 轴的偏移、比例缩放和镜像不接收到虚拟 Y 轴。TRAFOOF 时，虚拟 Y 轴的偏移、比例缩放和镜像会被删除。

4.5.6.3 选择/取消坐标转换：TRANSMIT



Transmit 扩展

可通过以下机床数据将 TRANSMIT 回转轴的轴总框架（即偏移、镜像和比例缩放）应用于转换：

- MD24905 $MC_TRANSMIT_ROT_AX_FRAME_1 = 1
- MD24955 $MC_TRANSMIT_ROT_AX_FRAME_2 = 1

例如可对框架链内一个框架中的工件倾斜角度进行补偿，从而输入回转轴的偏移。此偏移通常也用作坐标转换中的回转轴偏移。如上图所示的 C 轴偏移会产生对应的 X 值和 Y 值。

- MD24905 $MC_TRANSMIT_ROT_AX_FRAME_1 = 2
- MD24955 $MC_TRANSMIT_ROT_AX_FRAME_2 = 2

采用此设置时，回转轴的轴偏移乃至 SZS 都会在转换中被启用。SZS 框架中包含的回转轴轴偏移会作为旋转输入转换框架。仅当配置了转换框架时，此设置才生效。

框架扩展

下文介绍的扩展只适用于以下机床数据设置：

- MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1
- MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 2

选择 TRANSMIT 转换时，生成一根通过回转轴耦合的虚拟几何轴，其不参考轴框架，而只在轮廓框架中被启用。几何值通过回转轴旋转得出。所有其他几何轴在选择转换时接收其轴分量。

分量：

- 偏移
  选择 TRANSMIT 时，虚拟轴的偏移会被删除。回转轴的偏移可在转换中启用。
- 旋转
  转换前编写的旋转会被接收。
- 镜像
  虚拟轴的镜像会被删除。回转轴的镜像可在转换中启用。
- 比例缩放
  虚拟轴的比例缩放会被删除。回转轴的比例缩放可在转换中启用。

示例：机床数据

```
; 框架配置

$MC_MM_SYSTEM_FRAME_MASK='H41'              ; TRAFRAME, SETFRAME
$MC_CHSFRAME_RESET_MASK='H41'               ; 框架在复位后生效。
$MC_CHSFRAME_POWERON_MASK='H41'             ; 框架在上电时删除。

$MN_FRAME_GEOAX_CHANGE_MODE=1               ; 框架在几何轴切换后进行换算。
$MC_RESET_MODE_MASK='H4041'                 ; 基本框架在复位后不取消。
;$MC_RESET_MODE_MASK='H41'                  ; 基本框架在复位后取消。

;$MC_GCODE_RESET_VALUES[7]=2                ; G54 为缺省设置。
$MC_GCODE_RESET_VALUES[7]=1                 ; G500 为缺省设置。

$MN_MM_NUM_GLOBAL_USER_FRAMES=0
$MN_MM_NUM_GLOBAL_BASE_FRAMES=3

$MC_MM_NUM_USER_FRAMES=10                   ; 从 5 到 100
$MC_MM_NUM_BASE_FRAMES=3                    ; 从 0 到 8

$MN_NCBFRAME_RESET_MASK='HFF'
$MC_CHBFRAME_RESET_MASK='HFF'
```

```
$MN_MIRROR_REF_AX=0                          ; 镜像时无定标。
$MN_MIRROR_TOGGLE=0
$MN_MM_FRAME_FINE_TRANS=1                    ; 精偏
$MC_FRAME_ADD_COMPONENTS=TRUE                ; 可使用 G58、G59。

; TRANSMIT 为第 1 转换
$MC_TRAFO_TYPE_1=256

$MC_TRAFO_AXES_IN_1[0]=1
$MC_TRAFO_AXES_IN_1[1]=6
$MC_TRAFO_AXES_IN_1[2]=3
$MC_TRAFO_AXES_IN_1[3]=0
$MC_TRAFO_AXES_IN_1[4]=0

$MA_ROT_IS_MODULO[AX6]=TRUE;

$MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=1
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]=6
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]=3

$MC_TRANSMIT_BASE_TOOL_1[0]=0.0
$MC_TRANSMIT_BASE_TOOL_1[1]=0.0
$MC_TRANSMIT_BASE_TOOL_1[2]=0.0

$MC_TRANSMIT_ROT_AX_OFFSET_1=0.0
$MC_TRANSMIT_ROT_SIGN_IS_PLUS_1=TRUE

$MC_TRANSMIT_ROT_AX_FRAME_1=1


; TRANSMIT 为第 2 转换
$MC_TRAFO_TYPE_2=256

$MC_TRAFO_AXES_IN_2[0]=1
$MC_TRAFO_AXES_IN_2[1]=6
$MC_TRAFO_AXES_IN_2[2]=2
$MC_TRAFO_AXES_IN_2[3]=0
$MC_TRAFO_AXES_IN_2[4]=0

$MC_TRAFO_GEOAX_ASSIGN_TAB_2[0]=1
$MC_TRAFO_GEOAX_ASSIGN_TAB_2[1]=6
```

```
$MC_TRAFO_GEOAX_ASSIGN_TAB_2[2]=2

$MC_TRANSMIT_BASE_TOOL_2[0]=4.0
$MC_TRANSMIT_BASE_TOOL_2[1]=0.0
$MC_TRANSMIT_BASE_TOOL_2[2]=0.0

$MC_TRANSMIT_ROT_AX_OFFSET_2=19.0
$MC_TRANSMIT_ROT_SIGN_IS_PLUS_2=TRUE

$MC_TRANSMIT_ROT_AX_FRAME_2=1
```

示例：零件程序

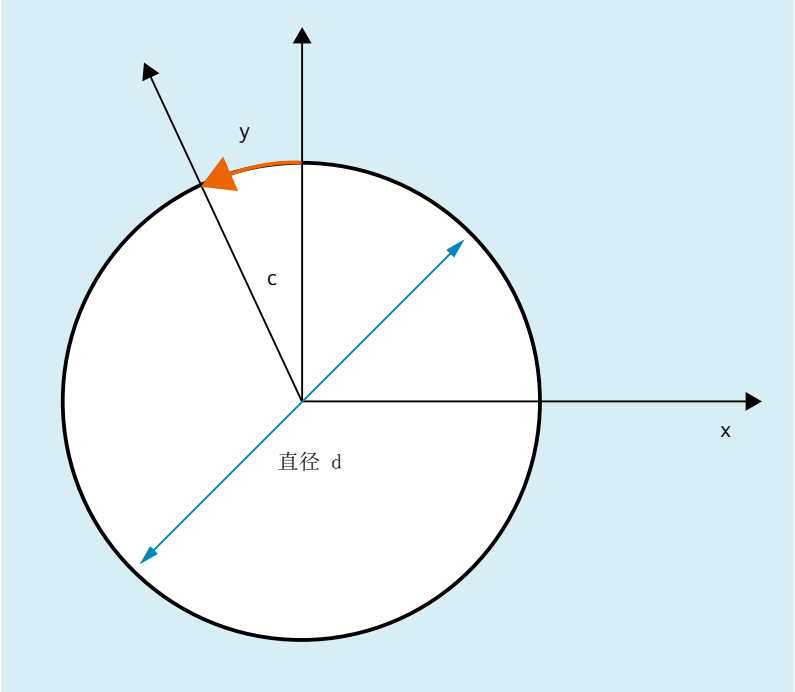
```
; 框架设置
N820 $P_UIFR[1] = ctrans(x,1,y,2,z,3,c,4)
N830 $P_UIFR[1] = $P_UIFR[1] : crot(x,10,y,20,z,30)
N840 $P_UIFR[1] = $P_UIFR[1] : cmirror(x,c)
N850
N860 $P_CHBFR[0] = ctrans(x,10,y,20,z,30,c,15)
N870


; 刀具选择、装夹补偿、平面选择
N890 T2 D1 G54 G17 G90 F5000 G64 SOFT
N900


; 逼近起始位置
N920 G0 X20 Z10
N930
N940 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,C,15)
N950 setal(61000)
N960 endif
N970 if $P_BFRAME <> $P_CHBFR[0]
N980 setal(61000)
N990 endif
N1000 if $P_IFRAME <>
CTRANS(X,1,Y,2,Z,3,C,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1010 setal(61000)
N1020 endif
N1030 if $P_IFRAME <> $P_UIFR[1]
N1040 setal(61000)
N1050 endif
```

```
N1060 if $P_ACTFRAME <>
CTRANS(X,11,Y,22,Z,33,C,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1070 setal(61000)
N1080 endif
N1090
N1100 TRANSMIT(2)
N1110
N1120 if $P_BFRAME <> CTRANS(X,10,Y,0,Z,20,CAZ,30,C,15)
N1130 setal(61000)
N1140 endif
N1180 if $P_IFRAME <>
CTRANS(X,1,Y,0,Z,2,CAZ,3,C,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1190 setal(61000)
N1200 endif
N1240 if $P_ACTFRAME <>
CTRANS(X,11,Y,0,Z,22,CAZ,33,C,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1250 setal(61001)
N1260 endif
N1270
N1280
N1290 $P_UIFR[1,x,tr] = 11
N1300 $P_UIFR[1,y,tr] = 14
N1310
N1320 g54
N1330


; 设置框架
N1350 ROT RPL=-45
N1360 ATRANS X-2 Y10
N1370


; 四边粗加工
N1390 G1 X10 Y-10 G41 OFFN=1; 余量 1mm
N1400 X-10
N1410 Y10
N1420 X10
N1430 Y-10
N1440


; 换刀
```

```
N1460 G0 Z20 G40 OFFN=0
N1470 T3 D1 X15 Y-15
N1480 Z10 G41
N1490


; 四边精加工
N1510 G1 X10 Y-10
N1520 X-10
N1530 Y10
N1540 X10
N1550 Y-10
N1560


; 取消框架
 N2950 m30 N1580 Z20 G40
N1590 TRANS
N1600
N1610 if $P_BFRAME <> CTRANS(X,10,Y,0,Z,20,CAZ,30,C,15)
N1620 setal(61000)
N1630 endif
N1640 if $P_BFRAME <> $P_CHBFR[0]
N1650 setal(61000)
N1660 endif
N1670 if $P_IFRAME <>
TRANS(X,11,Y,0,Z,2,CAZ,3,C,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1680 setal(61000)
N1690 endif
N1730 if $P_ACTFRAME <>
TRANS(X,21,Y,0,Z,22,CAZ,33,C,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1740 setal(61001)
N1750 endif
N1760
N1770 TRAFOOF
N1780
N1790 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,C,15)
N1800 setal(61000)
N1810 endif
N1820 if $P_BFRAME <> $P_CHBFR[0]
N1830 setal(61000)
N1840 endif
```

```
N1850 if $P_IFRAME <>
TRANS(X,11,Y,2,Z,3,C,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1860 setal(61000)
N1870 endif
N1880 if $P_IFRAME <> $P_UIFR[1]
N1890 setal(61000)
N1900 endif
N1910 if $P_ACTFRAME <>
TRANS(X,21,Y,22,Z,33,C,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1920 setal(61002)
N1930 endif
N1940
N2010 $P_UIFR[1] = ctrans()
N2011 $P_CHBFR[0] = ctrans()
N2020 $P_UIFR[1] = ctrans(x,1,y,2,z,3,c,0)
N2021 G54
N2021 G0 X20 Y0 Z10 C0
N2030 TRANSMIT(1)
N2040 TRANS x10 y20 z30
N2041 ATRANS y200
N2050 G0 X20 Y0 Z10
N2051 if $P_IFRAME <> CTRANS(X,1,Y,0,Z,3,CAY,2)
N2052 setal(61000)
N2053 endif
N2054 if $P_ACTFRAME <> CTRANS(X,11,Y,20,Z,33,CAY,2):CFINE(Y,200)
N2055 setal(61002)
N2056 endif
N2060 TRAFOOF
N2061 if $P_IFRAME <> $P_UIFR[1]
N2062 setal(61000)
N2063 endif
N2064 if $P_ACTFRAME <> CTRANS(X,11,Y,2,Z,33):CFINE(Y,0)
N2065 setal(61002)
N2066 endif
```

4.5.6.4 选择/取消坐标转换：TRACYL



TRACYL 扩展

可通过以下机床数据将 TRACYL 回转轴的轴总框架（即偏移、精偏、镜像和比例缩放）应用于转换：

- MD24805 $MC_TRACYL_ROT_AX_FRAME_1 = 1
- MD24855 $MC_TRACYL_ROT_AX_FRAME_2 = 1

例如可对框架链内一个框架中的工件倾斜角度进行补偿，从而输入回转轴的偏移。此偏移通常也用作坐标转换中的回转轴偏移或 y 偏移。如上图所示的 C 轴偏移会产生对应的 X 值和 Y 值。

- MD24805 $MC_TRACYL_ROT_AX_FRAME_1 = 2
- MD24855 $MC_TRACYL_ROT_AX_FRAME_2 = 2

采用此设置时，回转轴的轴偏移乃至 SZS 都会在转换中被启用。SZS 框架中包含的回转轴轴偏移会作为外表面上的偏移输入转换框架。仅当配置了转换框架时，此设置才生效。

框架扩展

下文介绍的扩展只适用于以下机床数据设置：

- MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1
- MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 2

选择 TRACYL 转换时，在外表面上生成一根通过回转轴耦合的虚拟几何轴，其不参考轴框架，而只在轮廓框架中被启用。虚拟几何轴的所有分量均会被删除。所有其他几何轴在选择转换时接收其轴分量。

分量：

- 偏移
  选择 TRACYL 时，虚拟轴的偏移会被删除。回转轴的偏移可在转换中启用。
- 旋转
  转换前编写的旋转会被接收。
- 镜像
  虚拟轴的镜像会被删除。回转轴的镜像可在转换中启用。
- 比例缩放
  虚拟轴的比例缩放会被删除。回转轴的比例缩放可在转换中启用。

示例：机床数据

```
; 框架配置

$MC_MM_SYSTEM_FRAME_MASK = 'H41'          ; TRAFRAME, SETFRAME
$MC_CHSFRAME_RESET_MASK = 'H41'           ; 框架在复位后生效。
$MC_CHSFRAME_POWERON_MASK='H41'           ; 框架在上电时删除。

$MN_FRAME_GEOAX_CHANGE_MODE = 1           ; 框架在几何轴切换后进行换算。

$MC_RESET_MODE_MASK='H4041'               ; 基本框架在复位后不取消。
;$MC_RESET_MODE_MASK = 'H41'              ; 基本框架在复位后取消。

;$MC_GCODE_RESET_VALUES[7]=2              ; G54 为缺省设置。
$MC_GCODE_RESET_VALUES[7] = 1             ; G500 为缺省设置。

$MN_MM_NUM_GLOBAL_USER_FRAMES = 0
$MN_MM_NUM_GLOBAL_BASE_FRAMES = 3

$MC_MM_NUM_USER_FRAMES = 10               ; 从 5 到 100
$MC_MM_NUM_BASE_FRAMES = 3                ; 从 0 到 8

$MN_NCBFRAME_RESET_MASK = 'HFF'
$MC_CHBFRAME_RESET_MASK = 'HFF'

$MN_MIRROR_REF_AX = 0                     ; 镜像时无定标。
$MN_MIRROR_TOGGLE = 0
$MN_MM_FRAME_FINE_TRANS = 1               ; 精偏
```

```
$MC_FRAME_ADD_COMPONENTS = TRUE            ; 可使用 G58、G59
```

```
; 带槽壁补偿的 TRACYL 为第 3 转换

$MC_TRAFO_TYPE_3 = 513; TRACYL

$MC_TRAFO_AXES_IN_3[0] = 1
$MC_TRAFO_AXES_IN_3[1] = 5
$MC_TRAFO_AXES_IN_3[2] = 3
$MC_TRAFO_AXES_IN_3[3] = 2

$MC_TRAFO_GEOAX_ASSIGN_TAB_3[0] = 1
$MC_TRAFO_GEOAX_ASSIGN_TAB_3[1] = 5
$MC_TRAFO_GEOAX_ASSIGN_TAB_3[2] = 3

$MC_TRACYL_BASE_TOOL_1[0] = 0.0
$MC_TRACYL_BASE_TOOL_1[1] = 0.0
$MC_TRACYL_BASE_TOOL_1[2] = 0.0

$MC_TRACYL_ROT_AX_OFFSET_1 = 0.0
$MC_TRACYL_ROT_SIGN_IS_PLUS_1 = TRUE

$MC_TRACYL_ROT_AX_FRAME_1 = 1
```

示例：零件程序

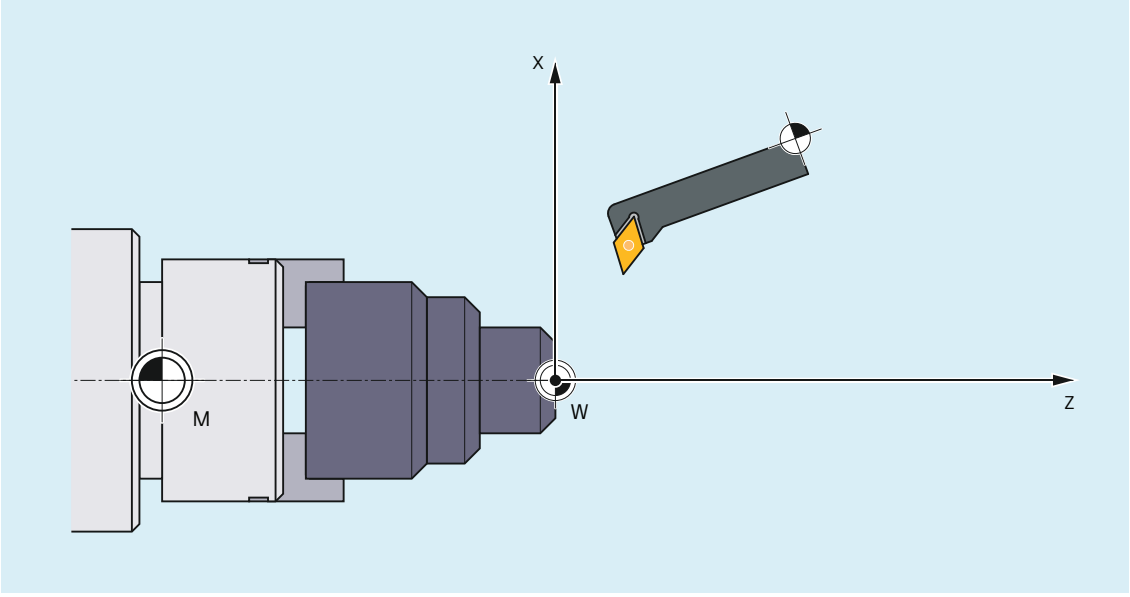
```
; 带槽壁补偿的简单运行测试
N450 G603
N460


; 框架设置
N500 $P_UIFR[1] = CTRANS(x,1,y,2,z,3,b,4)
N510 $P_UIFR[1] = $P_UIFR[1] : CROT(x,10,y,20,z,30)
N520 $P_UIFR[1] = $P_UIFR[1] : CMIRROR(x,b)
N530
N540 $P_CHBFR[0] = CTRANS(x,10,y,20,z,30,b,15)
N550
N560 G54
N570


; 连续路径运行，使用选择的精磨
```

```
N590 G0 x0 y0 z-10 b0 G90 F50000 T1 D1 G19 G641 ADIS=1 ADISPOS=5
N600
N610 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,B,15)
N620 setal(61000)
N630 endif
N640 if $P_BFRAME <> $P_CHBFR[0]
N650 setal(61000)
N660 endif
N670 if $P_IFRAME <>
TRANS(X,1,Y,2,Z,3,B,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N680 setal(61000)
N690 endif
N700 if $P_IFRAME <> $P_UIFR[1]
N710 setal(61000)
N720 endif
N730 if $P_ACTFRAME <>
TRANS(X,11,Y,22,Z,33,B,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N740 setal(61000)
N750 endif
N760


; 激活转换
N780 TRACYL(40.)
N790
N800 if $P_BFRAME <> CTRANS(X,10,Y,0,Z,30,CAY,20,B,15)
N810 setal(61000)
N820 endif
N830 if $P_CHBFR[0] <> CTRANS(X,10,Y,0,Z,30,CAY,20,B,15)
N840 setal(61000)
N850 endif
N860 if $P_IFRAME <>
TRANS(X,1,Y,0,Z,3,CAY,2,B,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N870 setal(61000)
N880 endif
N890 if $P_UIFR[1] <>
TRANS(X,1,Y,0,Z,3,CAY,2,B,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N900 setal(61000)
N910 endif
N920 if $P_ACTFRAME <>
TRANS(X,11,Y,0,Z,33,CAY,22,B,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N930 setal(61001)
```

```
N940 endif
N950
N960 $P_UIFR[1,x,tr] = 11
N970 $P_UIFR[1,y,tr] = 14
N980
N990 g54
N1000
N1010 if $P_BFRAME <> CTRANS(X,10,Y,0,Z,30,CAY,20,B,15)
N1020 setal(61000)
N1030 endif
N1040 if $P_BFRAME <> $P_CHBFR[0]
N1050 setal(61000)
N1060 endif
N1070 if $P_IFRAME <>
TRANS(X,11,Y,0,Z,3,CAY,2,B,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N1080 setal(61000)
N1090 endif
N1100 if $P_IFRAME <> $P_UIFR[1]
N1110 setal(61000)
N1120 endif
N1130 if $P_ACTFRAME <>
TRANS(X,21,Y,0,Z,33,CAY,22,B,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N1140 setal(61001)
N1150 endif
N1160


; 取消转换
N1180 TRAFOOF
N1190
N1200 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,B,15)
N1210 setal(61000)
N1220 endif
N1230 if $P_BFRAME <> $P_CHBFR[0]
N1240 setal(61000)
N1250 endif
N1260 if $P_IFRAME <>
TRANS(X,11,Y,2,Z,3,B,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N1270 setal(61000)
N1280 endif
N1290 if $P_IFRAME <> $P_UIFR[1]
N1300 setal(61000)
```

```
N1310 endif
N1320 if $P_ACTFRAME <>
TRANS(X,21,Y,22,Z,33,B,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N1330 setal(61002)
N1340 endif
N1350
N1360 G00 x0 y0 z0 G90
N1370
N1380 m30
```

4.5.6.5 选择/取消坐标转换：TRAANG



框架扩展：

下文介绍的扩展只适用于以下机床数据设置：

- MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1
- MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 2

分量：

- 偏移
  选择 TRAANG 时，虚拟轴的偏移会被保留。
- 旋转
  转换前编写的旋转会被接收。

- 镜像
  虚拟轴的镜像会被接收。
- 比例缩放
  虚拟轴的比例缩放会被接收。

示例：机床数据

```
; 框架配置

$MC_MM_SYSTEM_FRAME_MASK = 'H1'              ; SETFRAME
$MC_CHSFRAME_RESET_MASK = 'H41'              ; 框架在复位后生效。
$MC_CHSFRAME_POWERON_MASK='H41'              ; 框架在上电时删除。

$MN_FRAME_GEOAX_CHANGE_MODE = 1              ; 框架在几何轴切换后进行换算。

$MC_RESET_MODE_MASK='H4041'                  ; 基本框架在复位后不取消。
;$MC_RESET_MODE_MASK = 'H41'                 ; 基本框架在复位后取消。

;$MC_GCODE_RESET_VALUES[7]=2                 ; G54 为缺省设置。
$MC_GCODE_RESET_VALUES[7] = 1                ; G500 为缺省设置。

$MN_MM_NUM_GLOBAL_USER_FRAMES = 0
$MN_MM_NUM_GLOBAL_BASE_FRAMES = 3

$MC_MM_NUM_USER_FRAMES = 10                  ; 从 5 到 100
$MC_MM_NUM_BASE_FRAMES = 3                   ; 从 0 到 8

$MN_NCBFRAME_RESET_MASK = 'HFF'
$MC_CHBFRAME_RESET_MASK = 'HFF'

$MN_MIRROR_REF_AX = 0                        ; 镜像时无定标。
$MN_MIRROR_TOGGLE = 0
$MN_MM_FRAME_FINE_TRANS = 1                  ; 精偏
$MC_FRAME_ADD_COMPONENTS = TRUE              ; 可使用 G58、G59。
```

```
; TRAANG 为第 1 转换

$MC_TRAFO_TYPE_1 = 1024

$MC_TRAFO_AXES_IN_1[0] = 4                   ; 斜轴
$MC_TRAFO_AXES_IN_1[1] = 3                   ; 轴平行于 z 轴
$MC_TRAFO_AXES_IN_1[2] = 2
$MC_TRAFO_AXES_IN_1[3] = 0
```

```
$MC_TRAFO_AXES_IN_1[4] = 0

$MC_TRAFO_GEOAX_ASSIGN_TAB_1[0] = 4
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[1] = 2
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[2] = 3

$MC_TRAANG_ANGLE_1 = 85.
$MC_TRAANG_PARALLEL_VELO_RES_1 = 0.
$MC_TRAANG_PARALLEL_ACCEL_RES_1 = 0.

$MC_TRAANG_BASE_TOOL_1[0] = 0.0
$MC_TRAANG_BASE_TOOL_1[1] = 0.0
$MC_TRAANG_BASE_TOOL_1[2] = 0.0
```

；TRAANG 为第 2 转换

```
$MC_TRAFO_TYPE_2 = 1024

$MC_TRAFO_AXES_IN_2[0] = 4
$MC_TRAFO_AXES_IN_2[1] = 3
$MC_TRAFO_AXES_IN_2[2] = 0
$MC_TRAFO_AXES_IN_2[3] = 0
$MC_TRAFO_AXES_IN_2[4] = 0

$MC_TRAFO_GEOAX_ASSIGN_TAB_2[0] = 4
$MC_TRAFO_GEOAX_ASSIGN_TAB_2[1] = 0
$MC_TRAFO_GEOAX_ASSIGN_TAB_2[2] = 3

$MC_TRAANG_ANGLE_2 = -85.
$MC_TRAANG_PARALLEL_VELO_RES_2 = 0.2
$MC_TRAANG_PARALLEL_ACCEL_RES_2 = 0.2

$MC_TRAANG_BASE_TOOL_2[0] = 0.0
$MC_TRAANG_BASE_TOOL_2[1] = 0.0
$MC_TRAANG_BASE_TOOL_2[2] = 0.0
```

示例：零件程序

```
; 框架设置
N820 $P_UIFR[1] = ctrans(x,1,y,2,z,3,b,4,c,5)
N830 $P_UIFR[1] = $P_UIFR[1] : crot(x,10,y,20,z,30)
N840 $P_UIFR[1] = $P_UIFR[1] : cmirror(x,c)
N850
```

```
N860 $P_CHBFR[0] = ctrans(x,10,y,20,z,30,b,40,c,15)
N870


; 刀具选择、装夹补偿、平面选择
N890 T2 D1 G54 G17 G90 F5000 G64 SOFT
N900


; 逼近起始位置
N920 G0 X20 Z10
N930
N940 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,B,40,C,15)
N950 setal(61000)
N960 endif
N970 if $P_BFRAME <> $P_CHBFR[0]
N980 setal(61000)
N990 endif
N1000 if $P_IFRAME <>
TRANS(X,1,Y,2,Z,3,B,4,C,5):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1010 setal(61000)
N1020 endif
N1030 if $P_IFRAME <> $P_UIFR[1]
N1040 setal(61000)
N1050 endif
N1060 if $P_ACTFRAME <>
TRANS(X,11,Y,22,Z,33,B,44,C,20):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1070 setal(61000)
N1080 endif
N1090
N1100 TRAANG(,1)
N1110
N1120 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,CAX,10,B,40,C,15)
N1130 setal(61000)
N1140 endif
N1150 if $P_BFRAME <> $P_CHBFR[0]
N1160 setal(61000)
N1170 endif
N1180 if $P_IFRAME <>
CTRANS(X,1,Y,2,Z,3,CAX,1,B,4,C,5):CROT(X,10,Y,20,Z,30):CMIRROR(X,CAX,C)
N1190 setal(61000)
N1200 endif
```

```
N1210 if $P_IFRAME <> $P_UIFR[1]
N1220 setal(61000)
N1230 endif
N1240 if $P_ACTFRAME <>
TRANS(X,11,Y,22,Z,33,CAX,11,B,44,C,20):CROT(X,10,Y,20,Z,30):CMIRROR(X,CAX,
C)
N1250 setal(61001)
N1260 endif
N1270
N1280
N1290 $P_UIFR[1,x,tr] = 11
N1300 $P_UIFR[1,y,tr] = 14
N1310
N1320 g54
N1330


; 设置框架
N1350 ROT RPL=-45
N1360 ATRANS X-2 Y10
N1370


; 四边粗加工
N1390 G1 X10 Y-10 G41 OFFN=1; 余量 1mm
N1400 X-10
N1410 Y10
N1420 X10
N1430 Y-10
N1440


; 换刀
N1460 G0 Z20 G40 OFFN=0
N1470 T3 D1 X15 Y-15
N1480 Z10 G41
N1490


; 四边精加工
N1510 G1 X10 Y-10
N1520 X-10
N1530 Y10
```

```
N1540 X10
N1550 Y-10
N1560


; 取消框架
N1580 Z20 G40
N1590 TRANS
N1600
N1610 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,CAX,10,B,40,C,15)
N1620 setal(61000)
N1630 endif
N1640 if $P_BFRAME <> $P_CHBFR[0]
N1650 setal(61000)
N1660 endif
N1670 if $P_IFRAME <>
TRANS(X,11,Y,14,Z,3,CAX,1,B,4,C,5):CROT(X,10,Y,20,Z,30):CMIRROR(X,CAX,C)
N1680 setal(61000)
N1690 endif
N1700 if $P_IFRAME <> $P_UIFR[1]
N1710 setal(61000)
N1720 endif
N1730 if $P_ACTFRAME <>
TRANS(X,21,Y,34,Z,33,CAX,11,B,44,C,20):CROT(X,10,Y,20,Z,30):CMIRROR(X,CAX,
C)
N1740 setal(61001)
N1750 endif
N1760
N1770 TRAFOOF
N1780
N1790 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,B,40,C,15)
N1800 setal(61000)
N1810 endif
N1820 if $P_BFRAME <> $P_CHBFR[0]
N1830 setal(61000)
N1840 endif
N1850 if $P_IFRAME <>
TRANS(X,1,Y,14,Z,3,B,4,C,5):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1860 setal(61000)
N1870 endif
N1880 if $P_IFRAME <> $P_UIFR[1]
N1890 setal(61000)
```

```
N1900 endif
N1910 if $P_ACTFRAME <>
TRANS(X,11,Y,34,Z,33,B,44,C,20):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1920 setal(61002)
N1930 endif
N1940
N1950 m30
```

4.5.6.6 调整生效框架

程序执行期间或复位时，几何轴配置有可能发生变化。届时几何轴的数量可能为零到三不等。在无几何轴存在的情况下，生效框架中的分量（例如旋转）可能导致生效框架对该轴配置无效。通过报警“通道 %1 程序段 %2 为不存在的几何轴编写了旋转”显示。此报警将一直存在，直至框架被相应修改。

通过以下机床数据可启用对生效框架的自动调整：

MD24040 $MC_FRAME_ADAPT_MODE，位 <n> = <值>

位	<值>	含义
0	1	生效框架中的坐标轴旋转没有可用的几何轴时，将该旋转从生效框架删除。
1	1	生效框架中的剪切角变为垂直角。
2	1	将生效框架中所有几何轴的比例缩放设为 1。

通过以下机床数据可以删除生效框架中所有引起不存在几何轴的轴运动的旋转：

MD24040 $MC_FRAME_ADAPT_MODE = 1

数据管理框架此时不会改变。激活数据管理框架时，只接收可实现的旋转。

示例

不存在 Y 轴：

- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1
- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 0
- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 3
- $P_UIFR[1] = CROT(X,45,Y,45,Z,45)

程序代码	；注释
N390 G54 G0 X10 Z10 F10000	
IF $P_IFRAME <> CROT(Y,45)	；只接收围绕 Y 轴的旋转

程序代码	；注释
SETAL(61000) ENDIF	

4.5.6.7 映射框架

简介

“映射框架”功能用于在通道专用或全局数据管理框架内实现对轴专用框架的跨通道一致性修改。在哪些轴之间执行映射通过轴专用机床数据定义。

例如，为机床轴 AX1 和 AX4 启用了框架映射时，若在一个通道专用数据管理框架（例如：基本框架 $P_CHBFR[x]）中修改了 AX1 的轴专用框架（偏移、精偏、比例缩放、镜像），那么 AX1 和 AX4 的这些框架数据将传输至其作为通道轴的所有通道的所有通道专用数据管理框架（例如：基本框架 $P_CHBFR[x]）。

修改旋转的轴专用框架数据时不会进行框架映射。

前提条件

框架映射须满足下列前提条件：

- 用于映射的数据管理框架必须经过配置：
  MD28083 $MC_MM_SYSTEM_DATAFRAME_MASK（系统框架）
- **通道专用**数据管理框架须显性使能映射：
  MD10616 $MN_MAPPED_FRAME_MASK（框架映射使能）

  提示
  全局数据管理框架上总是会执行映射，无需使能。

参数设置

映射关系通过以下轴专用机床数据设置：

MD32075 $MA_MAPPED_FRAME[<AXn>] = "AXm"

AXn，AXm：机床轴名称，其中 n，m = 1, 2, ... 机床轴最大数量

映射规则

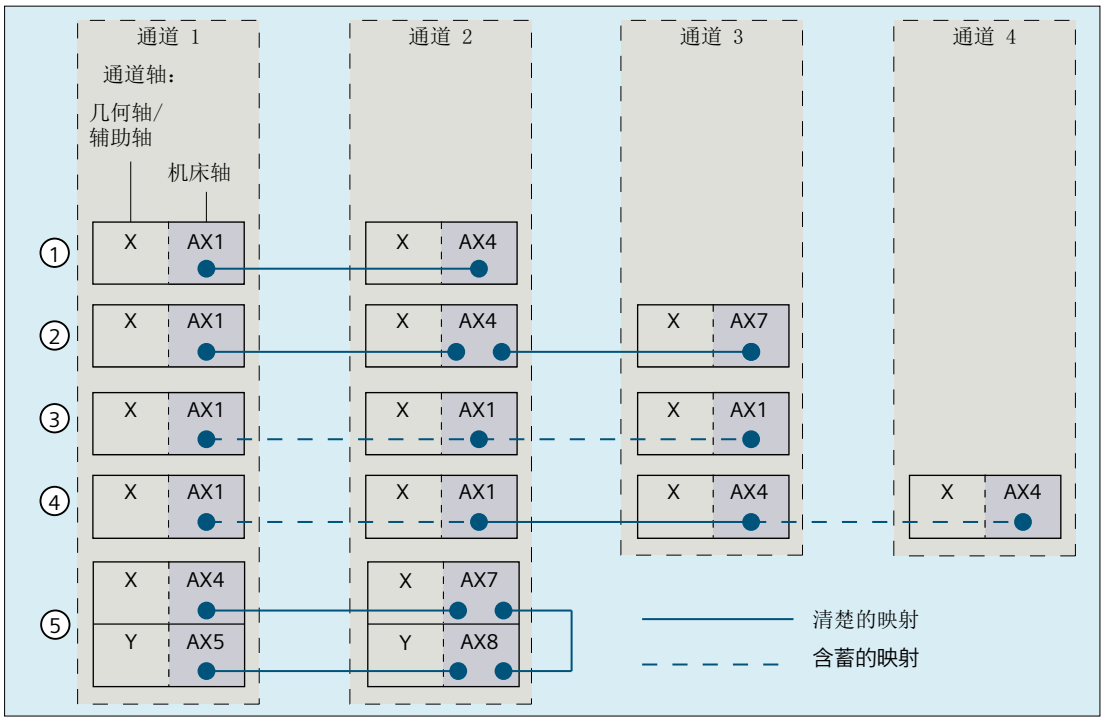
对于框架映射有以下规则：

- 映射为双向。
  可为轴 AXn 或 AXm 写入轴专用框架。框架数据总是为另一根轴接收。
- 系统总是会对设置的所有映射关系进行检测。
  在写入轴 AXn 的轴专用框架时，系统会检测其所有映射关系，并为所有直接或间接相关的轴接收框架数据。
- 映射为通道全局进行。
  在为一个通道专用框架写入轴 AXn 或 AXm 的轴专用框架时，系统会为 AXn 或 AXm 作为通道轴的所有通道接收框架数据。
- 在通过几何轴名称或辅助轴名称写入轴专用框架时，映射关系检测针对当前指定给该几何轴或辅助轴的机床轴进行。
- 映射针对特定框架。
  在写入轴专用框架时，框架数据映射只在相同的通道专用或全局数据管理框架内进行。

说明

数据一致性

用户/机床制造商须自行确保写入框架后所有通道中的框架数据一致，例如通过通道同步。



说明	参数设置：$MA_
① 简单映射关系： AX1(K1) ↔ AX4(K2)	MAPPED_FRAME[<AX1>] = "AX4"
② 级联映射关系： AX1(K1) ↔ AX4(K2) ↔ AX7(K3)	MAPPED_FRAME[<AX1>] = "AX4" MAPPED_FRAME[<AX4>] = "AX7"
③ 映射至自身，其中 AX1 作为通道 1、2、3 的通道轴： AX1(K1+K2+K3)	MAPPED_FRAME[<AX1>] = "AX1"
④ 两根轴之间的映射关系，分别为两个通道中的通道轴： AX1(K1+K2) ↔ AX4(K3+K4)	MAPPED_FRAME[<AX1>] = "AX4"
⑤ 级联映射关系，同个通道中写入多个通道轴： AX4(K1) ↔ AX7(K2) ↔ AX8(K2) ↔ AX5(K1)	MAPPED_FRAME[<AX4>] = "AX7" MAPPED_FRAME[<AX7>] = "AX8" MAPPED_FRAME[<AX8>] = "AX5"

图 4-23　映射示例

激活数据管理框架

数据管理框架可在零件程序中写入，或通过 SINUMERIK Operate 的操作界面写入。在通道中激活直接写入或通过框架映射写入的数据管理框架时，须注意以下几点：

- 在零件程序中写入
  数据管理框架必须在每个通道中显性激活（G500、G54 ... G599）。
- 通过操作界面写入
  数据管理框架通过操作界面写入，例如通过输入新的零点偏移。若该框架涉及的通道均不处于“通道生效”状态，那么修改过的数据管理框架将在这些通道中立即生效。若其中一个通道处于“通道生效”状态，那么数据管理框架不会在任何通道中生效。此时必须在每个通道中通过零件程序显性激活（G500、G54 ... G599）。或者其将在通道下一次切换至“通道复位”状态时生效。

示例

控制系统上设置了以下通道和通道轴：

- 通道 1
  - Z:几何轴
  - AX1:几何轴 Z 的机床轴
- 通道 2
  - Z:几何轴
  - AX4:几何轴 Z 的机床轴

两个通道中的 Z 轴零点总是相同：

- 映射关系：$MA_MAPPED_FRAME[AX1] = "AX4"

通道 1 和 2 中的零件程序

通道 1	通道 2
...	...
N100 WAIT (10,1,2)	N200 WAIT (10,1,2)
N110 $P_UIFR[1] = CTRANS(Z, 10)	
N120 WAIT (20,1,2)	N220 WAIT (20,1,2)
N130 G54	N230 G54
N140 IF ($P_IFRAME[0, Z, TR] <> 10)	N230 IF ($P_IFRAME[0, Z, TR] <> 10)
N150 SETAL(61000)	N250 SETAL(61000)
N160 ENDIF	N260 ENDIF
...	...

说明：

N100 / N200	通道同步，用于确保框架数据写入和映射的一致性
N110	写入可设定数据管理框架 $P_UIFR[1]： Z 轴零点偏移至 10 mm 轴专用框架数据的映射： 通道 1：Z ≙ AX1 ⇔ 通道 2：Z ≙ AX4
N120 / N220	通道同步，用于确保新框架数据的一致激活
N130 / N230	激活新框架数据
N140 / N240	检查 Z 轴零点：10 mm

4.5.6.8 映射框架

简介

“映射框架”功能用于在通道专用或全局数据管理框架内实现对轴专用框架的跨通道一致性修改。在哪些轴之间执行映射通过轴专用机床数据定义。

例如，为机床轴 AX1 和 AX4 启用了框架映射时，若在一个通道专用数据管理框架（例如：基本框架 $P_CHBFR[x]）中修改了 AX1 的轴专用框架（偏移、精偏、比例缩放、镜像），那么 AX1 和 AX4 的这些框架数据将传输至其作为通道轴的所有通道的所有通道专用数据管理框架（例如：基本框架 $P_CHBFR[x]）。

修改旋转的轴专用框架数据时不会进行框架映射。

前提条件

框架映射须满足下列前提条件：

- 用于映射的数据管理框架必须经过配置：
  MD28083 $MC_MM_SYSTEM_DATAFRAME_MASK（系统框架）
- **通道专用**数据管理框架须显性使能映射：
  MD10616 $MN_MAPPED_FRAME_MASK（框架映射使能）

  提示
  全局数据管理框架上总是会执行映射，无需使能。

参数设置

映射关系通过以下轴专用机床数据设置：

MD32075 $MA_MAPPED_FRAME[<AXn>] = "AXm"

AXn，AXm：机床轴名称，其中 n，m = 1, 2, ... 机床轴最大数量

映射规则

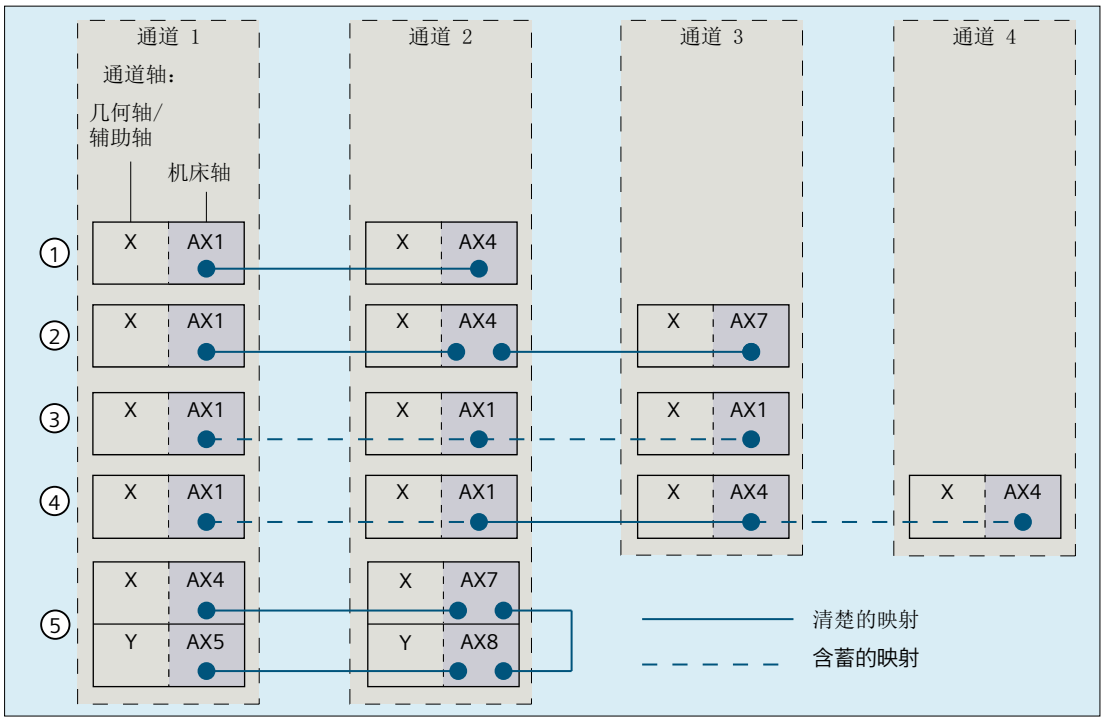
对于框架映射有以下规则：

- 映射为双向。
  可为轴 AXn 或 AXm 写入轴专用框架。框架数据总是为另一根轴接收。
- 系统总是会对设置的所有映射关系进行检测。
  在写入轴 AXn 的轴专用框架时，系统会检测其所有映射关系，并为所有直接或间接相关的轴接收框架数据。
- 映射为通道全局进行。
  在为一个通道专用框架写入轴 AXn 或 AXm 的轴专用框架时，系统会为 AXn 或 AXm 作为通道轴的所有通道接收框架数据。
- 在通过几何轴名称或辅助轴名称写入轴专用框架时，映射关系检测针对当前指定给该几何轴或辅助轴的机床轴进行。
- 映射针对特定框架。
  在写入轴专用框架时，框架数据映射只在相同的通道专用或全局数据管理框架内进行。

说明

数据一致性

用户/机床制造商须自行确保写入框架后所有通道中的框架数据一致，例如通过通道同步。



说明	参数设置：$MA_
① 简单映射关系： AX1(K1) ↔ AX4(K2)	MAPPED_FRAME[<AX1>] = "AX4"
② 级联映射关系： AX1(K1) ↔ AX4(K2) ↔ AX7(K3)	MAPPED_FRAME[<AX1>] = "AX4" MAPPED_FRAME[<AX4>] = "AX7"
③ 映射至自身，其中 AX1 作为通道 1、2、3 的通道轴： AX1(K1+K2+K3)	MAPPED_FRAME[<AX1>] = "AX1"
④ 两根轴之间的映射关系，分别为两个通道中的通道轴： AX1(K1+K2) ↔ AX4(K3+K4)	MAPPED_FRAME[<AX1>] = "AX4"
⑤ 级联映射关系，同个通道中写入多个通道轴： AX4(K1) ↔ AX7(K2) ↔ AX8(K2) ↔ AX5(K1)	MAPPED_FRAME[<AX4>] = "AX7" MAPPED_FRAME[<AX7>] = "AX8" MAPPED_FRAME[<AX8>] = "AX5"

图 4-24　映射示例

激活数据管理框架

数据管理框架可在零件程序中写入。在通道中激活直接写入或通过框架映射写入的数据管理框架时，须注意以下几点：

- 在零件程序中写入
  数据管理框架必须在每个通道中显性激活（G500、G54 ... G599）。

示例

控制系统上设置了以下通道和通道轴：

- 通道 1
  - Z:几何轴
  - AX1:几何轴 Z 的机床轴
- 通道 2
  - Z:几何轴
  - AX4:几何轴 Z 的机床轴

两个通道中的 Z 轴零点总是相同：

- 映射关系：$MA_MAPPED_FRAME[AX1] = "AX4"

通道 1 和 2 中的零件程序

通道 1	通道 2
...	...
N100 WAIT (10,1,2)	N200 WAIT (10,1,2)
N110 $P_UIFR[1] = CTRANS(Z, 10)	
N120 WAIT (20,1,2)	N220 WAIT (20,1,2)
N130 G54	N230 G54
N140 IF ($P_IFRAME[0, Z, TR] <> 10)	N230 IF ($P_IFRAME[0, Z, TR] <> 10)
N150 SETAL(61000)	N250 SETAL(61000)
N160 ENDIF	N260 ENDIF
...	...

说明：

N100 / N200　通道同步，用于确保框架数据写入和映射的一致性

N110	写入可设定数据管理框架 $P_UIFR[1]： Z 轴零点偏移至 10 mm 轴专用框架数据的映射： 通道 1：Z ≙ AX1 ⇔ 通道 2：Z ≙ AX4
N120 / N220	通道同步，用于确保新框架数据的一致激活
N130 / N230	激活新框架数据
N140 / N240	检查 Z 轴零点：10 mm

4.5.7 预定义框架功能

4.5.7.1 反转框架

INVFRAME()功能从一个框架计算出反框架。

功能说明

一个框架与其反转框架级联时总是得到零框架
框架：框架 `INVFRAME( FRAME )` ⇒ 零框架

框架反转是用于坐标转换的辅助功能。测量框架的计算通常在 WCS 中进行。若需将计算出的框架转换至另一坐标系，即将计算出的框架输入框架链内的一个任意框架，则可通过以下计算实现：

新的总框架由旧的总框架与计算出的框架级联得出：

```
$P_ACTFRAME = $P_ACTFRAME : $AC_MEAS_FRAME
```

框架链中的新框架如下得出：

- 目标框架为 $P_SETFRAME：
  ```
  $P_SETFRAME = $P_ACTFRAME :
  $AC_MEAS_FRAME :INVFRAME($P_ACTFRAME) : $P_SETFRAME
  ```
- 目标框架位第 n 个通道基本框架 $P_CHBFRAME[<n>]:
  ```
  k = $MN_MM_NUM_GLOBAL_BASE_FRAMES
  ```
  - n = 0 时 TMP 为：
    ```
    TMP = $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :
    $P_NCBFRAME[<0...k>]
    ```
  - n ≠ 0 时 TMP 为：
    ```
    TMP = $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :
    $P_NCBFRAME[<0..k>] : $P_CHBFRAME[<0...n> - 1]
    ```
  ```
  $P_CHBFRAME[<n>] = INVFRAME(TMP) : $P_ACTFRAME :
  $AC_MEAS_FRAME :INVFRAME($P_ACTFRAME) :TMP : $P_CHBFRAME[<n>]
  ```
- 目标框架为 $P_IFRAME：
  ```
  TMP = $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME : $P_BFRAME
  $P_IFRAME = INVFRAME(TMP) : $P_ACTFRAME :
  $AC_MEAS_FRAME :INVFRAME($P_ACTFRAME) :TMP : $P_IFRAME
  ```

示例

需要在当前 SETFRAME 中输入一个框架（例如通过测量功能测得），使新的总框架为旧的总框架和测量框架的级联。SETFRAME 通过框架反转进行相应换算。

程序代码	注释
DEF INT RETVAL	
DEF FRAME TMP	
$TC_DP1[1,1]=120	; 类型
$TC_DP2[1,1]=20.	; 0
$TC_DP3[1,1]= 10.	; (z) 长度补偿矢量
$TC_DP4[1,1]= 0.	; (y)
$TC_DP5[1,1]= 0.	; (x)
$TC_DP6[1,1]= 2.	; 半径
T1 D1	
g0 x0 y0 z0 f10000	
G54	
$P_CHBFRAME[0] = CROT(Z,45)	
$P_IFRAME[X,TR] = -SIN(45)	
$P_IFRAME[Y,TR] = -SIN(45)	
$P_PFRAME[Z,TR] = -45	
$AC_MEAS_VALID = 0	; 通过 4 个测量点测量拐角

程序代码	注释
G1 X-1 Y-3	; 第 1 返回测量点
$AC_MEAS_LATCH[0] = 1	; 第 1 保存测量点
G1 X5 Y-3	; 第 2 返回测量点
$AC_MEAS_LATCH[1] = 1	; 第 2 保存测量点
G1 X-4 Y4	; 第 3 返回测量点
$AC_MEAS_LATCH[2] = 1	; 第 3 保存测量点
G1 X-4 Y1	; 第 4 个返回测量点
$AC_MEAS_LATCH[3] = 1	; 第 4 个保存测量点
$AA_MEAS_SETPOINT[X] = 0	; 设置拐角的设定位置
$AA_MEAS_SETPOINT[Y] = 0	
$AA_MEAS_SETPOINT[Z] = 0	
$AC_MEAS_CORNER_SETANGLE = 90	; 设置交角设定值
$AC_MEAS_WP_SETANGLE = 30	
$AC_MEAS_ACT_PLANE = 0	; 测量平面为 G17
$AC_MEAS_T_NUMBER = 1	; 选择刀具
$AC_MEAS_D_NUMBER = 1	
$AC_MEAS_TYPE = 4	; 将测量类型设置为“拐角 1”
RETVAL = MEASURE()	; 开始测量
IF RETVAL <> 0	
SETAL(61043, << RETVAL)	
ENDIF	
IF $AC_MEAS_WP_ANGLE <> 30	
SETAL(61043, << $AC_MEAS_WP_ANGLE)	
ENDIF	
IF $AC_MEAS_CORNER_ANGLE <> 90	
SETAL(61043, << $AC_MEAS_CORNER_ANGLE)	
ENDIF	
; 对测得的框架进行转换并写入 $P_SETFRAME，使新的总框架	
; 为旧的总框架和测量框架的级联。	
$P_SETFRAME = $P_ACTFRAME : $AC_MEAS_FRAME :INVFRAME($P_ACTFRAME) : $P_SETFRAME	
$P_SETFR = $P_SETFRAME	; 写入数据管理中的系统框架
G1 X0 Y0	; 逼近拐角
G1 X10	; 沿旋转 30 度的矩形退回
Y10	
X0	
Y0	
M30	

4.5.7.2 框架链中的叠加框架

通过工件上的测量或零件程序或循环中的计算得到的框架通常是叠加于生效的总框架生效。因此，WCS 和编程的零点需要进行偏移和/或旋转。此时，测量的框架位于框架变量中且不算入框架链内。

功能说明

ADDFRAME() 功能从一个临时框架计算出通过参数 <STRING> 指定的目标框架，这样，新生效的总框架 $P_ACTFRAME 便可由旧生效总框架与临时框架的级联得出：

ERG = ADDFRAME(TMPFRAME,"$P_SETFRAME") ⇒ $P_SETFRAME$_{新}$ = $P_SETFRAME$_{旧}$
ADD TMPFRAME 和 $P_ACTFRAME$_{新}$ = $P_ACTFRAME$_{旧}$：TMPFRAME

如果指定了生效的框架作为目标框架，那么新的总框架**在预处理中**生效。

若目标框架为数据管理框架，那么必须在通道中（例如：零件程序、循环）显性激活才能使其生效。

该功能会反馈可在用户程序中对其作出响应的返回值。

编程

句法

```
INT ADDFRAME(<FRAME>,<STRING>)
```

含义

<FRAME>：	带需要额外算入的值的框架变量	
	类型	FRAME
<STRING>：	生效框架或数据管理框架的名称： • 生效框架 "$P_CYCFRAME", "$P_ISO4FRAME", "$P_PFRAME", "$P_WPFRAME", "$P_TOOLFRAME", "$P_IFRAME", "$P_GFRAME", "$P_CHBFRAME[<n>]", "$P_NCBFRAME[<n>]", "$P_ISO1FRAME", "$P_ISO2FRAME", "$P_ISO3FRAME", "$P_EXTFRAME", "$P_SETFRAME", "$P_PARTFRAME" • 数据管理框架 "$P_CYCFR", "$P_ISO4FR, "$P_TRAFR", "$P_WPFR", "$P_TOOLFR", "$P_UIFR[<n>]", "$P_GFR", "$P_CHBFR[<n>]", "$P_NCBFR[<n>]", "$P_ISO1FR, "$P_ISO2FR, "$P_ISO3FR, "$P_EXTFR", "$P_SETFR", "$P_PARTFR"	
	类型	STRING

返回值：	允许的返回值： • 0：正常 • 1：目标设定（字符串）错误 • 2：目标框架未配置 • 3：框架中的旋转不被允许	
	类型	INT

4.5.8 功能

4.5.8.1 设置零点、工件测量和刀具测量

实际值设定通过 HMI 或测量循环进行。计算出的框架会被写入系统框架 SETFRAME。设定实际值时，可对 WCS 中的轴设定位置进行修改。

“对刀”这一概念表示的是工件测量和刀具测量。进行工件测量时，可以测量工件一条边沿、一个拐角或一个钻孔的位置。为确定工件或钻孔的零点，可将实测位置和 WCS 中的设定位置相比。由此得出的偏移可输入到选中的框架中。进行刀具测量时，可根据一个经过测量的标准件测量刀具的长度或半径。

测量可通过操作或测量循环进行。与 NC 的通讯通过预定义系统变量进行。在 NC 中，通过 HMI 操作激活 PI 通讯，或通过测量循环中的零件程序指令来执行计算。可选择一把刀具或一个平面作为计算的基础。计算出的框架会被输入结果框架。

4.5.8.2 轴外部零点偏移

说明

机床数据

通过以下机床数据激活外部零点偏移或系统框架 $P_EXTFRAME：

MD28082 $MC_MM_SYSTEM_FRAME_MASK，位 1 = TRUE

外部零点偏移的值可使用 OPI 通过 HMI 操作界面和 PLC 用户程序手动设定，或在零件程序中通过轴专用系统变量 $AA_ETRANS [<轴>] 编写。

激活

外部零点偏移通过以下接口信号激活：
<Axis>.basic.out.progTestEnableCtrl（接收外部零点偏移）

特性

外部零点偏移激活时，系统会立即停止所有轴（除去指令轴和 PLC 轴）的运行并重组预处理。当前系统框架和数据管理中系统框架的粗偏会被设置为轴专用系统变量 $AA_ETRANS[<轴>] 的值。之后系统会先运行偏移，然后再恢复中断的运动。

采用增量尺寸设定时的特性

采用增量尺寸设定 G91 和以下机床数据设置时：

MD42440 $MC_FRAME_OFFSET_INCR_PROG（框架中的零点偏移）= 0

系统会与机床数据配置相反通过逼近程序段在“通过系统框架实现外部零点偏移”范围内运行偏移，尽管偏移通过框架设定。

说明

外部零点偏移总是作为绝对值生效。

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<Axis>.basic.out.progTestEnableCtrl	LBP_Chan*.A_PrgtestAxRel	DB31, ...DBX3.7

4.5.8.3 刀架

偏移

在“P”和“M”类型的运动系统中，选择刀架时会激活一个叠加框架（可定向刀架的工作台偏移），其将零点偏移作为工作台旋转的后果。零点偏移会被输入系统框架 $P_PARTFR。此时该框架的偏移分量被改写。其他框架分量则保持不变。

系统框架 $P_PARTFR 必须通过以下机床数据使能：

MD28082 $MC_MM_SYSTEM_FRAME_MASK，位 2 = 1（用于 TCARR 和 PAROT 的系统框架）

说明

也可通过以下机床数据来设置偏移，用于接收工作台偏移：

MD20184 $MC_TOCARR_BASE_FRAME_NUMBER = <基本框架的编号>

由于兼容性原因，只能在较老的软件版本中使用此方案。建议不采用此方案。

刀架切换引起的框架偏移会在通过 TCARR=... 进行选择时立即生效。而对刀具长度的修改仅在刀具生效时才会立即生效。

激活不会触发框架旋转，已生效的旋转也不会改变。和 T 情形（仅刀具可旋转）相似，用于计算的回转轴位置根据 TCOFR / TCOABS 指令从生效框架的旋转分量或 $TC_CARR 的记录得出。激活框架会使工件坐标系中的位置相应改变，而不会引起机床的补偿运动。

情形如下图所示：

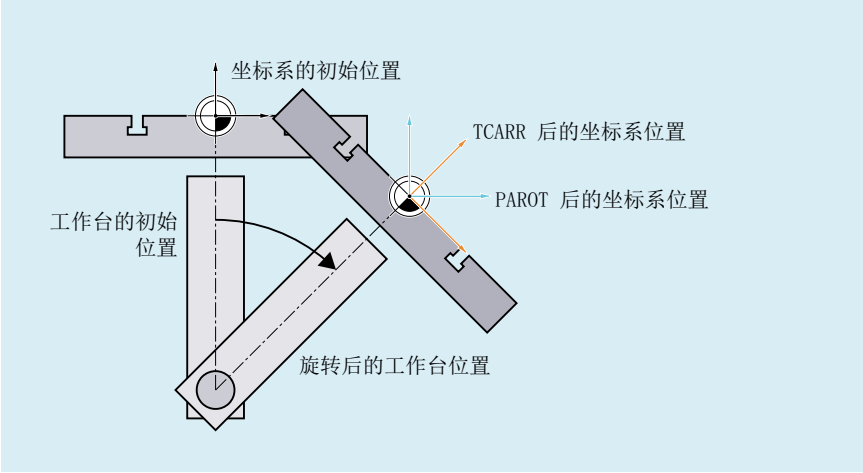


图 4-25　通过 TCARR 激活可旋转刀具工作台时的框架

在运动类型为 M 时（刀具和平台可以分别绕轴旋转），使用 TCARR 激活刀架会导致有效的刀具长度（如果刀具有效的话）和零点偏移同时发生变化。

旋转

根据加工任务，在使用可旋转刀架或工作台时，不仅需要考虑零点偏移（作为框架或作为刀具长度），还需考虑旋转。然而激活一个可定向刀架在任何情况下都不会直接导致坐标系旋转。

若只有刀具可旋转，可通过 TOFRAME 或 TOROT 为其定义一个框架。

对于可旋转工作台（运动系统类型 P 和 M），通过 TCARR 激活最先同样不会引起旋转，也就是说，尽管在以机床为基准时坐标系发生了偏移，但以工件零点为基准时其保持固定，定向也保持不变。

若需相对工件固定的坐标系，即相对原始位置不仅发生偏移，还根据工作台旋转进行旋转，则可类似采用可旋转刀具时的情形通过 PAROT 激活相应旋转。

使用 PAROT 指令时，生效框架中的偏移、比例缩放和镜像保留，旋转分量则由可定向刀架的旋转分量替代，该刀架依据工作台旋转。此时总体可编程框架保持不变，包括其旋转分量。

描述刀具工作台旋转的旋转分量可输入系统框架 $PARTFR，或输入通过 MD20184 $MC_TOCARR_BASE_FRAME_NUMBER 设置的基本框架。

$MC_MM_SYSTEM_FRAME_MASK，位 2 = <值>

值	含义
1	旋转分量 → $PARTFR
0	旋转分量 → MD20184 $MC_TOCARR_BASE_FRAME_NUMBER

依据描述工作台平移时的提示，建议不要为新设备采用第二种方案。

Partframe 的旋转分量可通过 PAROTOF 删除，与该框架在基本框架还是系统框架中无关。偏移分量通过激活不触发偏移的刀架来删除，或通过 TCARR=0 取消可能生效的可定向刀架来删除。

在回转台或刀具通过两根旋转轴定位的情形下，PAROT 或 TOROT 会考虑总体定向变化。在混合运动中只考虑由旋转轴引起的、相应部分的情况。这样一来，例如可在使用 TOROT 时旋转工件，使斜向平面平行于空间固定的 X-Y 平面；加工时则须考虑刀具旋转，例如须加工不垂直于该平面的钻孔时。

示例

机床中回转台的旋转轴指向 Y 轴正方向。工作台已旋转 +45 度。之后通过 PAROT 定义一个框架，同样为围绕 Y 轴的 45 度旋转。相对于外界未旋转的坐标系（在图中用" TCARR 之后的坐标系位置"进行标识）相对于一起运动的坐标系（PAROT 之后的位置）却旋转了 -45 度。若通过 ROT Y-45 定义此坐标系，之后在 TCOFR 生效时选择刀架，则会为刀架的旋转轴得到 +45 度的角度。

无可定向刀架生效时，语言指令 PAROT 不会被系统拒绝。但是此类调用不会引起框架变化。

在刀具方向上加工

首先，在配备可定向刀具的机床上，在未激活框架（例如通过 TOFRAME 或 TOROT）的情况下有时需要沿刀具方向运行（钻削时常见），并且其中一根轴指向刀具方向。当斜向加工中定义斜面的框架生效，但是索引刀架（端面齿）使刀具定向无法任意设置，因此无法完全垂直设置刀具时，也会出现问题。在这些情形下必须沿刀具方向钻孔，与原本要求的垂直于平面的运动有偏差，否则钻头无法向其纵轴方向引导（刀具断裂）。

增量运行

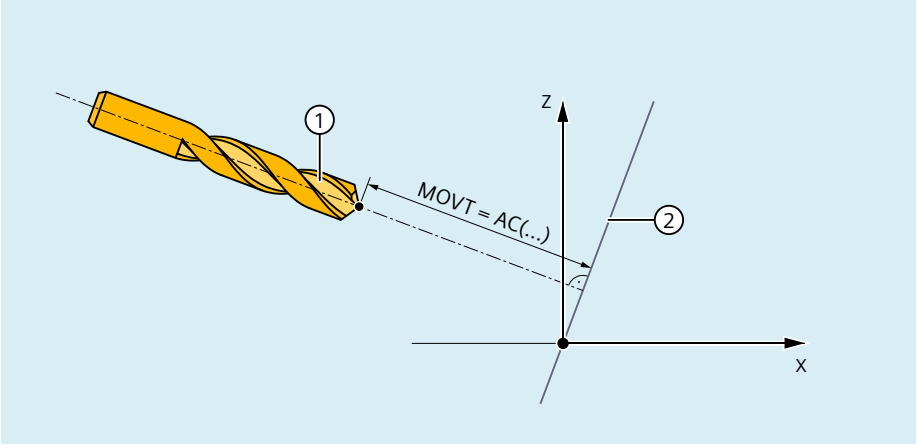
沿刀具方向的增量运行的终点通过 MOVT = <值> 或 MOVT=IC (<值>) 编写。

此时正运行方向定义为刀尖指向刀架的的方向。这对应平行于轴的加工，例如通过 G91 Z...。

绝对运行

沿刀具方向的绝对运行的终点通过 MOVT=AC(<值>)编写。

为此定义一个穿过当前零点的平面，其表面法线矢量平行于刀具定向。MOVT 指定的位置将基于该平面（见图）。参考轴用于计算最终位置。生效框架不受这些内部计算影响。



MOVT 编程与是够存在可设定刀架无关。运动方向取决于生效的平面。
其沿垂直坐标的方向运行，即 G17 中沿 Z 轴方向，G18 中沿 Y 轴方向，G19 中沿 X 轴方向。
这既适用于无可定向刀架生效的情形，也适用于可定向刀架无/有可旋转刀具的情形。

定向转换（3-4-5 轴转换）生效时，MOVT 以相同方式生效。

若在编写了 MOVT 的程序段中同时修改了刀具定向（例如在 5 轴转换生效时通过同时插补回转轴），那么程序段开始处的定向决定 MOVT 的运行方向。
5 轴转换生效时，刀具中心点不会因为定向改变受到影响，即轨迹保持为直线，方向由程序段开始处的刀具定向确定。

若编写了 MOVT，线性插补和样条插补（G0、G1、ASPLINE、BSPLINE、CSPLINE）必须生效。否则会输出报警。
若有样条插补方式生效，那么得到的轨迹通常非直线，因为由 MOVT 得到的终点作为通过 X、Y、Z 显性编写的终点处理。

在包含 MOVT 的程序段中不允许编写几何轴。

通过空间角定义框架旋转

若需定义围绕多于一根轴进行旋转的框架，可通过将单次旋转级联实现。其中在经旋转的新坐标系中进行后续旋转。

这既适用于一个程序段中的编程，也适用于在多个连续程序段中建立框架：

- 一个程序段：N10 ROT X...Y...Z...
- 多个连续的程序段：
  N10 ROT Y...
  N20 AROT X...
  N30 AROT Z...

空间角

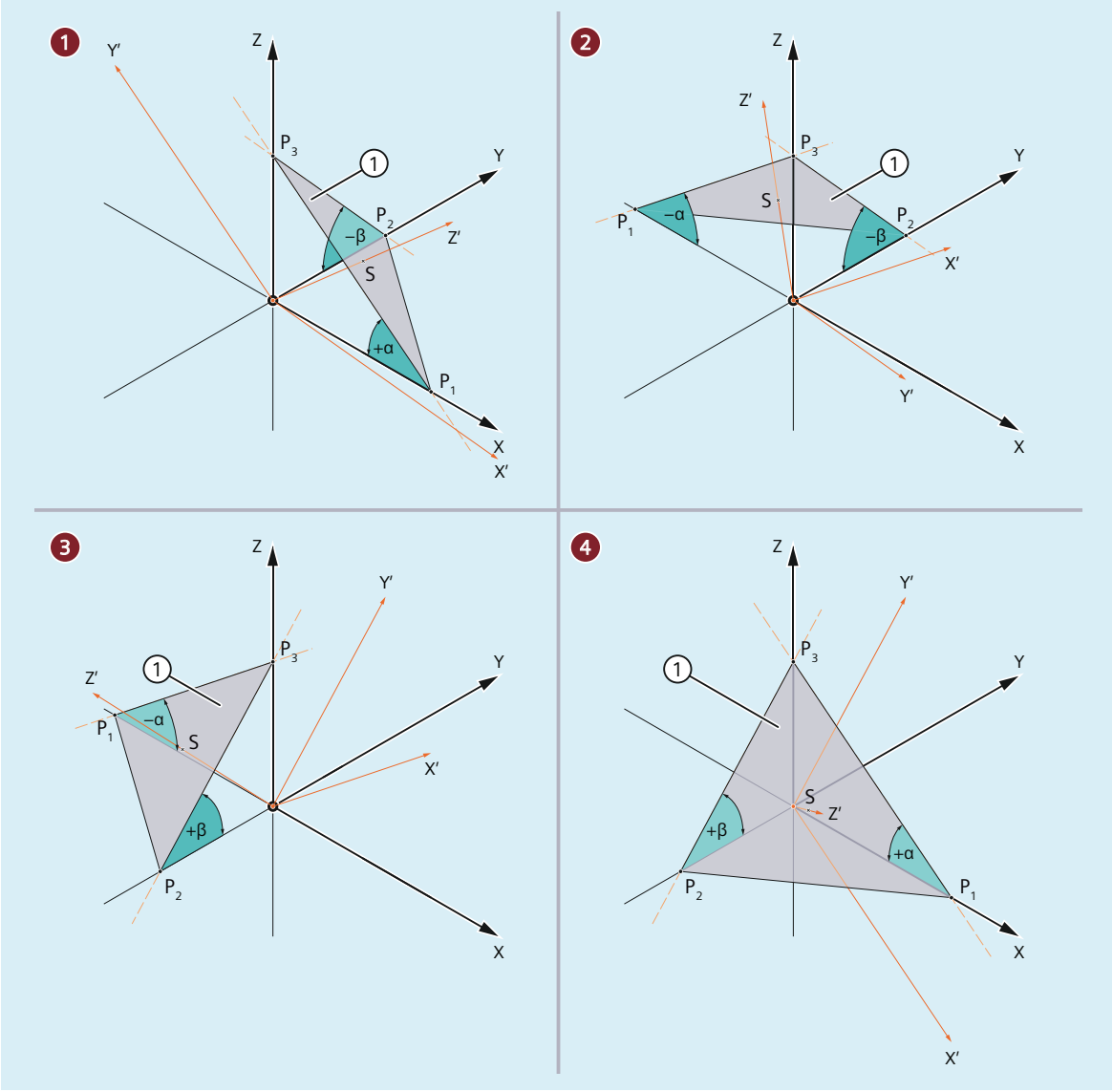
在工件图纸中，为描述斜面常会给出空间角。空间角是斜面与工件坐标系的主平面（X-Y、Y-Z、Z-X 平面）的交线形成的角度（见下图）。通过设定两个空间角在空间中对平面进行定位。第三个空间角由前两个得出。

借助 ROTS、AROTS 和 CROTS 指令，可将旋转直接描述为空间角。

允许设定单个空间角。在此情形下通过 ROTS 或 AROTS 执行的旋转与 ROT 或 AROT 时相同。

指令中编写的两个轴定义一个平面。未编写的轴则定义对应的笛卡尔坐标系第三轴。这样一来，对于编写的两个轴亦明确定义第一轴和第二轴。其中定义对应 G17/G18/G19 中的平面定义。

通过平面轴的轴字母编写的角度可指定一根轴，平面的另一根轴必须围绕该轴旋转从而转换为交线，交线与通过另一根轴和第三根轴组成的平面一同构成旋转过的平面。在编写两个角度中有一个趋近于零的情况下，通过此定义可确保以此方式定义的平面能够转换为只编写一根轴（例如通过 ROT 或 AROT）的平面。



I, ..., IV　象限 1 至 4

①　斜面作为针对新 G17 平面的设定

α, β　斜面的空间角

图 4-26　围绕空间角旋转

图中在象限 I 至 IV 中示出示例性平面的空间角。该斜面定义工件坐标系 WCS 旋转后 G17 平面的定向。空间角的符号给出了坐标系围绕相应轴旋转时所围绕的方向：

1. 围绕 y 轴旋转：
   工件坐标系 WCS 围绕 y 轴旋转符号相关的角度 α ⇒
   x' 轴平行于 xz 平面与该斜面的交线（共线）
2. 围绕 x' 轴旋转：
   工件坐标系 WCS' 围绕 x' 轴旋转符号相关的角度 β ⇒
   – y' 轴平行于 zy 平面与该斜面的交线（共线）
   – z' 轴垂直于该斜面
   – G17' 平行于该斜面

针对每个象限，用于编写工件坐标系 WCS 的 G17 平面朝斜面的定向的相应编程为：

- 象限 I：`ROTS X<+α> Y<–β>`
- 象限 II：`ROTS X<–α> Y<–β>`
- 象限 III：`ROTS X<–α> Y<+β>`
- 象限 IV：`ROTS X<+α> Y<+β>`

定向

设定空间角时，平面内的二维坐标系定向（即围绕表面法线矢量的旋转角）未定义。因此在定义坐标系的位置时，须使旋转过的第一轴位于未旋转的坐标系的第一轴和第三轴所构成的平面内。

这表示：

- 编写 X 和 Y 时，新的 X 轴位于原本的 Z-X 平面中。
- 编写 Z 和 X 时，新的 Z 轴位于原本的 Y-Z 平面中。
- 编写 Y 和 Z 时，新的 Y 轴位于原本的 X-Y 平面中。

如果需要与预设偏离的坐标系位置，必须通过 `AROT...` 执行叠加旋转。

参数设置：ZY'X"（RPY 角）或 ZX'Z" 转换

在输入时，系统根据以下机床数据将编写的空间角换算成 zy'x" 转换（RPY 角）或 zx'z"转换后的等效欧拉角：

MD10600 $MN_FRAME_ANGLE_INPUT_MODE

沿刀具方向进行框架旋转

通过旧有软件版本中即已提供的语言指令 TOFRAME，可定义一个 Z 轴指向刀具方向的框架。此时已编写的框架会被改写为一个纯粹描述旋转的框架。之前生效的框架中可能存在的零点偏移、镜像和比例缩放均会被删除。
此特性有时会造成干扰。特别是在经常需要保留通过工件中的参考点定义的零点偏移时。

因此系统又引入了语言指令 TOROT，在编写的框架中该指令只改写旋转分量，其他分量则保持不变。通过 TOROT 定义的旋转与使用 TOFRAME 时相同。
同 TOFRAME 一样，TOROT 也与是否存在可定向刀架无关。此语言指令还可在 5 轴转换中使用。

此外，在可定向刀架生效时，通过新的语言指令 TOROT 可为每种运动系统类型实现一致性的编程。

通过 TOFRAME 或 TOROT 定义 Z 方向指向刀具方向的框架。此定义适用于通常采用 G17 平面的铣削加工。对于车削加工或在 G18 或 G19 平面进行的加工，定义的框架能在 X 轴或 Y 轴对准将更有意义。为此 G 组 53 中存在以下指令：

- TOFRAMEX、TOFRAMEY、TOFRAMEZ
- TOROTX、TOROTY、TOROTZ

通过这些指令可定义相应的框架。此时 TOFRAME 和 TOFRAMEZ，以及 TOROT 和 TOROTZ 的功能相同。

通过 TOROT 或 TOFRAME 生成的框架可写入到独立的系统框架（$P_TOOLFR）中。可编程的框架保持不变。

- 前提条件：MD28082 $MC_MM_SYSTEM_FRAME_MASK，位 3 = 1

在编写 TOROT 或 TOFRAME 等指令时，有系统框架或无系统框架时的特性相同。继续处理可编程框架时才会出现差别。

说明

建议在新设备上对通过 G 功能组 53 的指令生成的框架只使用为其设计的系统框架。

示例

TOROT 后编写 TRANS。通过不设定参数的 TRANS 删除可编程框架。在无系统框架的方案中，通过 TOROT 产生的可编程框架的框架分量也会因此被删除；若 TOROT 分量处于系统框架中则会被保留。

TOROT 或 TOFRAME 等指令通过语言指令 TOROTOF 取消。TOROTOF 会删除总系统框架 $P_TOOLFR。若 TOFRAME 等指令描述的不是系统框架，而是可编程框架（旧方案），TOROT 只会删除旋转分量，其他框架分量保持不变。

若激活语言指令 TOFRAME 或 TOROT 前已有旋转框架生效，通常要求新定义的框架与旧框架的偏差尽可能小。例如，由于端面齿回转轴使刀具定向无法任意设置，必须微调框架定义时，便是此类情形。通过上述语言指令可将新框架的 Z 方向定义为唯一值。

参数设置：TOFRAME、TOROT 和 PAROT 中的框架定义（SD42980）

在借助 TOFRAME、TOROT（TOROTY、TOROTX）或 PAROT 定义框架时，通过以下设定数据定义当前加工平面的几何轴（G17：X 轴和 Y 轴）的方向。

在进行框架计算时，如此定义刀具方向，使得刀具方向平行于框架的垂直坐标（G17：Z 轴）并且垂直于加工平面。

围绕刀具轴的旋转最初是任意的。可通过此设定数据决定该自由旋转，使得新定义的框架与先前生效框架的偏差尽可能小。

在此设定数据不等于零的所有情形下，若新框架与旧框架的刀具方向一致，则生效的框架保持不变。

SD42980 $SC_TOFRAME_MODE

TCARR（请求刀架）和 PAROT（在工件上对准工件坐标系）

TCARR 使用通过以下机床数据标识的基本框架：
MD20184 $MC_TOCARR_BASE_FRAME_NUMBER

为了避免与已使用所有基本框架的系统冲突，可为 TCARR 和 PAROT 创建一个独立的系统框架。

PAROT、TOROT 和 TOFRAME 目前用于改写可编程框架的旋转分量。在此情形下无法单独取消 PAROT 或 TOROT。复位后可编程框架会被删除，这意味着运行方式切换至 JOG 后 PAROT 和 TOROT 的旋转分量将不再存在。此外用户必须能无限制地访问可编程框架。通过 PAROT 和 TOROT 生成的框架必须通过数据备份存档和重新载入。

TCARR 和 PAROT 的系统框架如下配置：
MD28082 $MC_MM_SYSTEM_FRAME_MASK，位 2 = 1

此时以下机床数据将不再被分析：
MD20184 $MC_TOCARR_BASE_FRAME_NUMBER

若为 TCARR 配置了系统框架，那么 TCARR 和 PAROT 将描述对应的系统框架，否则描述的是通过机床数据 MD20184 标识的基本框架。

使用 P 类型和 M 类型的运动系统时，TCARR 将可定向刀架的工作台偏移（工作台旋转引起的零点偏移）作为偏移输入系统框架。PAROT 对系统框架进行换算，从而得到基于工件的工件坐标系。

系统框架采用剩余存储，并在复位后保留。即便在运行方式切换时系统框架也保持生效。

PAROT 和 TOROT、TOFRAME 分别被指定给一个独立 G 组用于显示。

PAROTOF

PAROTOF 是 PAROT 的取消指令。该指令会删除 PAROT 的系统框架中的旋转。此时当前 $P_PARTFRAME 和数据管理框架 $P_PARTFR 中的旋转均会被删除。之后坐标系的位置会根据 TCARR 重新恢复。PAROTOF 和 PAROT 位于同一 G 功能组中，因此在 G 指令显示中出现。

TOROT（通过框架旋转平行于刀具定向对准 WCS 的 Z 轴）和 TOFRAME（dto.）

TOROT 和 TOFRAME 的系统框架通过以下机床数据激活：
MD28082 $MC_MM_SYSTEM_FRAME_MASK，位 3 = 1

此系统框架位于可编程框架前的框架链中。
SZS 坐标系相应地位于可编程框架前。

TOROTOF

TOROTOF 是 TOROT 和 TOFRAME 的取消指令。此指令会删除对应的系统框架。此时当前 $P_TOOLFRAME 和数据管理框架 $P_TOOLFR 均会被删除。TOROTOF 和 TOROT 及 TOFRAME 位于同一 G 功能组中，因此在 G 指令显示中出现。

示例

使用取消了运动系统的可定向刀架

```
N10 $TC_DP1[1,1]= 120
N20 $TC_DP3[1,1]= 13                    ; 刀具长度 13 mm
; 定义刀架 1 :
N30 $TC_CARR1[1] = 0                    ; 偏移矢量 1 的 X 分量偏移矢量
N40 $TC_CARR2[1] = 0                    ; 偏移矢量 1 的 Y 分量偏移矢量
N50 $TC_CARR3[1] = 0                    ; 偏移矢量 1 的 Z 分量偏移矢量
N60 $TC_CARR4[1] = 0                    ; 偏移矢量 2 的 X 分量偏移矢量
N70 $TC_CARR5[1] = 0                    ; 偏移矢量 2 的 Y 分量偏移矢量
N80 $TC_CARR6[1] = -15                  ; 偏移矢量 2 的 Z 分量偏移矢量
N90 $TC_CARR7[1] = 1                    ; 第 1 根轴的 X 轴
N100 $TC_CARR8[1] = 0                   ; 第 1 根轴的 Y 轴
N110 $TC_CARR9[1] = 0                   ; 第 1 根轴的 Z 轴
N120 $TC_CARR10[1] = 0                  ; 第 2 根轴的 X 轴
N130 $TC_CARR11[1] = 1                  ; 第 2 根轴的 Y 轴
N140 $TC_CARR12[1] = 0                  ; 第 2 根轴的 Z 轴
```

```
N150 $TC_CARR13[1] = 30                ; 第 1 根轴的轴
N160 $TC_CARR14[1] = -30               ; 第 2 根轴的轴
N170 $TC_CARR15[1] = 0                 ; 偏移矢量 3 的 X 分量偏移矢量
N180 $TC_CARR16[1] = 0                 ; 偏移矢量 3 的 Y 分量偏移矢量
N190 $TC_CARR17[1] = 0                 ; 偏移矢量 3 的 Z 分量偏移矢量
N200 $TC_CARR18[1] = 0                 ; 偏移矢量 4 的 X 分量偏移矢量
N210 $TC_CARR19[1] = 0                 ; 偏移矢量 4 的 Y 分量偏移矢量
N220 $TC_CARR20[1] = 15                ; 偏移矢量 4 的 Z 分量偏移矢量
N230 $TC_CARR21[1] = A                 ; 用于第 1 根轴的轴
N240 $TC_CARR22[1] = B                 ; 用于第 2 根轴的轴
N250 $TC_CARR23[1] = "M"               ; 刀架的类型
N260 X0 Y0 Z0 A0 B45 F2000
N270 TCARR=1 X0 Y10 Z0 T1 TCOABS       ; 选择定向刀架
N280 PAROT                             ; 工作台旋转
N290 TOROT                             ; 在刀具定向中旋转 z 轴
N290 X0 Y0 Z0
N300 G18 MOVT=AC(20)                   ; 在 G18 平面中加工
N310 G17 X10 Y0 Z0                     ; 在 G17 平面中加工
N320 MOVT=-10
N330 PAROTOF                           ; 取消工作台旋转
N340 TOROTOF                           ; 不再在刀具上对准 WCS
N400 M30
```

4.5.9 带 SAVE 属性的子程序

对于不同的框架，其子程序相关特性可通过 SAVE 属性设置。

可设定框架 G54 - G599

通过 MD10617 $MN_FRAME_SAVE_MASK.**位 0** 可对可设定框架的特性进行设置：

- 位 0 = 0
  若通过子程序使用系统变量 $P_IFRAME 只修改了生效的可设定框架的值，而 G 指令保留，那么修改在子程序结束后将保留。
- 位 0 = 1
  子程序结束时，调用子程序前生效的可设定框架、G 指令和数值重新激活。

基本框架 $P_CHBFR[ ] 和 $P_NCBFR[ ]

通过 MD10617 $MN_FRAME_SAVE_MASK.**位 1** 可设置基本框架的特性：

- 位 1 = 0
  若通过子程序修改生效的基本框架，那么修改在子程序结束后将保留。
- 位 1 = 1
  子程序结束时，调用子程序前生效的基本框架重新激活。

可编程的框架

子程序结束时。调用子程序前生效的可编程框架重新激活。

系统框架

若通过子程序修改系统框架，那么修改在子程序结束后将保留。

4.5.10 数据备份

数据块

在以下数据块中进行框架的数据备份：

- 通道专用框架 ⇒ 数据块 _N_CHAN<x>_UFR
- 全局框架 ⇒ 数据块 _N_NC_UFR

说明

- 强烈建议在备份与重置备份的系统框架期间不要修改以下机床数据。否则可能会导致备份的系统框架无法再次载入。
  MD28082 $MC_MM_SYSTEM_FRAME_MASK（在通道中计算的通道专用的系统框架设计）
- 数据备份总是按照当前生效的几何轴指定关系进行，而不是按照机床数据中设置的轴配置进行。

系统框架的数据备份

系统框架的数据备份仅针对创建在数据管理中的系统框架。这些系统框架在调试控制系统时通过以下机床数据选择：

MD28083 $MC_MM_SYSTEM_DATAFRAME_MASK

不备份未创建在数据管理中的框架。

NC 全局框架的数据备份

仅当在以下其中一个机床数据中设置了至少一个 NC 全局框架时，才能对 NC 全局框架进行数据备份。

- MD18601 $MN_MM_NUM_GLOBAL_USER_FRAMES
- MD18602 $MN_MM_NUM_GLOBAL_BASE_FRAMES
- MD18603 $MN_MM_NUM_GLOBAL_G_FRAMES

4.5.11 坐标系中的位置

坐标系中的当前设定值位置可通过以下系统变量读取。通过 PLC 可选择以 WCS、SZS、BZS 或 MCS 显示实际值。为此可使用软键“以 MCS/WCS 显示实际值”。机床制造商可通过 PLC 设置，在其机床中哪个坐标系等同于工件坐标系。HMI 会从 NC 请求对应的实际值。

$AA_IM[轴]

使用变量 $AA_IM[轴] 可为每根轴读取机床坐标系中的设定值。

$AA_IEN[轴]

使用变量 $AA_IEN[轴] 可为每根轴读取可设定的零点坐标系 SZS 中的设定值。

$AA_IBN[轴]

使用 $AA_IBN[轴] 可为每根轴读取基本零点坐标系 BZS 中的设定值。

$AA_IW[轴]

使用 $AA_IW[轴] 可为每根轴读取工件坐标系 WCS 中的设定值。

4.5.12 控制系统特性

4.5.12.1 上电

框架	上电后的状态
可编程框架 $P_PFRAME	删除
可设定框架 $P_IFRAME	保留，取决于： • MD24080 $MC_USER_FRAME_POWERON_MASK，位 0 • MD20152 $MC_GCODE_RESET_MODE[7]

框架	上电后的状态
磨削框架 $P_GFRAME	保留，取决于： • MD24080 $MC_USER_FRAME_POWERON_MASK，位 0 • MD20152 $MC_GCODE_RESET_MODE[63]
总基本框架 $P_ACTBFRAME	保留，取决于： • MD20110 $MC_RESET_MODE_MASK 位 0 和位 14 通过以下机床数据可以删除基本框架： • MD10615 $MN_NCBFRAME_POWERON_MASK • MD24004 $MC_CHBFRAME_POWERON_MASK
系统框架： $P_PARTFRAME, $P_SETFRAME, $P_ISO1FRAME, $P_ISO2FRAME, $P_ISO3FRAME, $P_TOOLFRAME, $P_WPFRAME, $P_TRAFRAME, $P_ISO4FRAME, $P_RELFRAME, $P_CACFRAME	保留 通过以下机床数据可以删除单个系统框架： • MD24008 $MC_CHSFRAME_POWERON_MASK 系统框架的删除优先在数据管理中进行。
外部零点偏移 $P_EXTFRAME	保留，但是必须重新激活。
DRF 偏移	删除

4.5.12.2 运行方式切换

系统框架

系统框架继续生效。

JOG 运行方式

在 JOG 运行方式下，旋转生效时只为几何轴将当前框架纳入计算。所有其他轴框架均不被考虑。

PLC 轴和指令轴

对于 PLC 轴和指令轴，其特性可通过以下机床数据设置：

MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED（不允许框架或 HL 补偿）

4.5.12.3 通道复位/零件程序结束

基本框架的复位特性

基本框架的复位特性通过以下机床数据设置：

MD20110 $MC_RESET_MODE_MASK（定义通道复位/零件程序结束后的控制系统初始设置）

基本框架的复位特性

通道复位/零件程序结束后，系统框架仍保留在数据管理中。

可通过以下机床数据配置单个系统框架的激活：

MD24006 $MC_CHSFRAME_RESET_MASK，位 <n> = <值>（通道复位/零件程序结束后生效的系统框架）

位	值	含义
0	1	通道复位/零件程序结束后实际值设置和对刀的系统框架生效。
1	1	通道复位/零件程序结束后外部零点偏移的系统框架生效。
2	---	不分析。
3	---	不分析。
4	1	通道复位/零件程序结束后工件参考点的系统框架生效。
5	1	通道复位/零件程序结束后循环的系统框架生效。
6	---	预留，复位特性取决于 MD20110 $MC_RESET_MODE_MASK。
7	1	通道复位/零件程序结束后系统框架 $P_ISO1FR 生效。
8	1	通道复位/零件程序结束后系统框架 $P_ISO2FR 生效。
9	1	通道复位/零件程序结束后系统框架 $P_ISO3FR 生效。
10	1	通道复位/零件程序结束后系统框架 $P_ISO4FR 生效。
11	1	通道复位/零件程序结束后系统框架 $P_RELFR 生效。
位 <n> = 0 时，相应的系统框架不生效。		

TCARR、PAROT、TOROT 和 TOFRAME 的系统框架的复位特性

通道复位/零件程序结束后 TCARR、PAROT、TOROT 和 TOFRAME 的系统框架的复位特性取决于 G 指令复位设置。

可通过以下机床数据进行设置：

- MD20110 $MC_RESET_MODE_MASK，位 <n> = <值>

<table>
<tr><th>位</th><th>值</th><th colspan="2">含义</th></tr>
<tr><td>0</td><td>0</td><td colspan="2">保留 TCARR 和 PAROT 的当前系统框架。</td></tr>
<tr><td rowspan="9">0</td><td rowspan="9">1</td><td>其他相关机床数据设置</td><td>作用</td></tr>
<tr><td>MD20152 $MC_GCODE_RESET_MODE[51] = 0 和
MD20150 $MC_GCODE_RESET_VALUES[51] = 1</td><td>PAROTOF</td></tr>
<tr><td>MD20152 $MC_GCODE_RESET_MODE[51] = 0 和
MD20150 $MC_GCODE_RESET_VALUES[51] = 2</td><td>PAROT</td></tr>
<tr><td>MD20152 $MC_GCODE_RESET_MODE[51] = 1</td><td>保留 TCARR 和 PAROT</td></tr>
<tr><td>MD20152 $MC_GCODE_RESET_MODE[52] = 0 和
MD20150 $MC_GCODE_RESET_VALUES[52] = 1</td><td>TOROTOF</td></tr>
<tr><td>MD20152 $MC_GCODE_RESET_MODE[52] = 0 和
MD20150 $MC_GCODE_RESET_VALUES[52] = 2</td><td>TOROT</td></tr>
<tr><td>MD20152 $MC_GCODE_RESET_MODE[52] = 0 和
MD20150 $MC_GCODE_RESET_VALUES[52] = 3</td><td>TOFRAME</td></tr>
<tr><td>MD20152 $MC_GCODE_RESET_MODE[52] = 1</td><td>保留 TOROT 和 TOFRAME</td></tr>
<tr><td colspan="4">位 0 == 1 时的其他位的说明</td></tr>
<tr><td rowspan="4">14</td><td>0</td><td colspan="2">级联总基本框架会被删除。</td></tr>
<tr><td rowspan="3">1</td><td colspan="2">总基本框架是由以下机床数据得出的：</td></tr>
<tr><td colspan="2">MD24002 $MC_CHBFRAME_RESET_MASK，位 <n> = 1
n：第 n 个通道专用基本框架被纳入级联的总基本框架内。</td></tr>
<tr><td colspan="2">MD10613 $MN_NCBFRAME_RESET_MASK，位 <n> = 1
n：第 n 个 NCU 全局基本框架被纳入级联的总基本框架内。</td></tr>
</table>

TCARR 为 PAROT 两个独立的功能，其定义的框架相同。使用 PAROTOF 时，TCARR 的分量同样不会在通道复位/零件程序结束时激活。

- MD20152 $MC_GCODE_RESET_MODE[ ]（G 功能组的复位特性）
- MD20150 $MC_GCODE_RESET_VALUES[ ]（G 功能组的初始设置）

通道复位/零件程序结束后的框架状态

框架	通道复位/零件程序结束后的状态
可编程框架：$P_PFRAME	删除
可设定框架 $P_IFRAME	保留，取决于： • MD20110 $MC_RESET_MODE_MASK • MD20152 $MC_GCODE_RESET_MODE[7]
磨削框架 $P_GFRAME	保留，取决于： • MD20110 $MC_RESET_MODE_MASK • MD20152 $MC_GCODE_RESET_MODE[63]
总基本框架 $P_ACTBFRAME	保留，取决于： • MD20110 $MC_RESET_MODE_MASK 位 0 和位 14 • MD10613 $MN_NCBFRAME_RESET_MASK • MD24002 $MC_CHBFRAME_RESET_MASK
系统框架： $P_PARTFRAME, $P_SETFRAME, $P_ISO1FRAME, $P_ISO2FRAME, $P_ISO3FRAME, $P_TOOLFRAME, $P_WPFRAME, $P_TRAFRAME, $P_ISO4FRAME, $P_RELFRAME, $P_CACFRAME	保留，取决于： • MD24006 $MC_CHSFRAME_RESET_MASK • MD20150 $MC_GCODE_RESET_VALUES[<n>]
外部零点偏移 $P_EXTFRAME	保留
DRF 偏移	保留

删除系统框架

在通道复位/零件程序结束时可通过以下机床数据删除数据管理中的系统框架：

MD24007 $MC_CHSFRAME_RESET_CLEAR_MASK，位 <n> = <值>

位	值	含义
0	1	通道复位/零件程序结束时删除实际值设置和对刀的系统框架。
1	1	通道复位/零件程序结束时删除外部零点偏移的系统框架。
2	---	预留，TCARR 和 PAROT 参见 MD20150 $MC_GCODE_RESET_VALUES[ ]。
3	---	预留，TOROT 和 TOFRAME 参见 MD20150 $MC_GCODE_RESET_VALUES[ ]。

位	值	含义
4	1	通道复位/零件程序结束时删除工件参考点的系统框架。
5	1	通道复位/零件程序结束时删除循环的系统框架。
6	---	预留，复位特性取决于 MD20110 $MC_RESET_MODE_MASK。
7	1	通道复位/零件程序结束时删除系统框架 $P_ISO1FR。
8	1	通道复位/零件程序结束时删除系统框架 $P_ISO2FR。
9	1	通道复位/零件程序结束时删除系统框架 $P_ISO3FR。
10	1	通道复位/零件程序结束时删除系统框架 $P_ISO4FR。
11	1	通道复位/零件程序结束时删除系统框架 $P_RELFR。
位 <n> = 0 时，不删除相应的系统框架。		

4.5.12.4 零件程序开始

零件程序开始后的框架状态

框架	零件程序开始后的状态
可编程框架 $P_PFRAME	删除
可设定框架 $P_IFRAME	保留，取决于： MD20112 $MC_START_MODE_MASK
磨削框架 $P_GFRAME	保留，取决于： MD20112 $MC_START_MODE_MASK
总基本框架 $P_ACTBFRAME	保留
系统框架： $P_PARTFRAME, $P_SETFRAME, $P_ISO1FRAME, $P_ISO2FRAME, $P_ISO3FRAME, $P_TOOLFRAME, $P_WPFRAME, $P_TRAFRAME, $P_ISO4FRAME, $P_RELFRAME, $P_CACFRAME	保留
外部零点偏移 $P_EXTFRAME	保留
DRF 偏移	保留

4.5.12.5 程序段搜索

带计算的程序段搜索

在“进行计算的程序段搜索”中，数据管理框架也会被修改。

程序段搜索终止

若程序段搜索由于通道复位终止，可通过以下机床数据将所有数据管理框架复位为程序段搜索前的值：

MD28560 $MC_MM_SEARCH_RUN_RESTORE_MODE，位 <n>

位	值	含义
0	1	恢复数据管理中的所有框架。

在级联程序段搜索中，框架会被设置为前一程序段搜索中的状态。

SERUPRO

不支持“SERUPRO”功能。

4.5.12.6 REPOS

对框架无特殊处理。 若在 ASUB 中修改了框架，则其保留在程序中。 通过 REPOS 重新定位时，只要在 ASUB 中激活了修改，经过修改的框架便会被启用。

4.6 工件相关的实际值系统

4.6.1 概述

定义

“工件相关实际值系统”涵盖了一系列功能，用于协助用户实现以下操作：

- 启动后，启用通过机床数据定义的工件坐标系。
  特性：
  - 无需额外操作
  - 在 JOG 和 AUTO 运行方式下生效
- 零件程序结束时生效的设置在下一个零件程序中保留，所涉及的设置包括：
  - 生效的平面
  - 可设定框架（G54-G57）
  - 运动转换
  - 生效的刀具补偿
- 通过操作界面在工件坐标系和机床坐标系间进行切换。
- 通过操作修改工件坐标系（例如修改可设定框架或刀具补偿）。

4.6.2 使用工件相关实际值系统

前提条件

已针对该系统执行上一章节中描述的设置。
HMI 软件启动后的初始设置为 MCS。

切换至 WCS

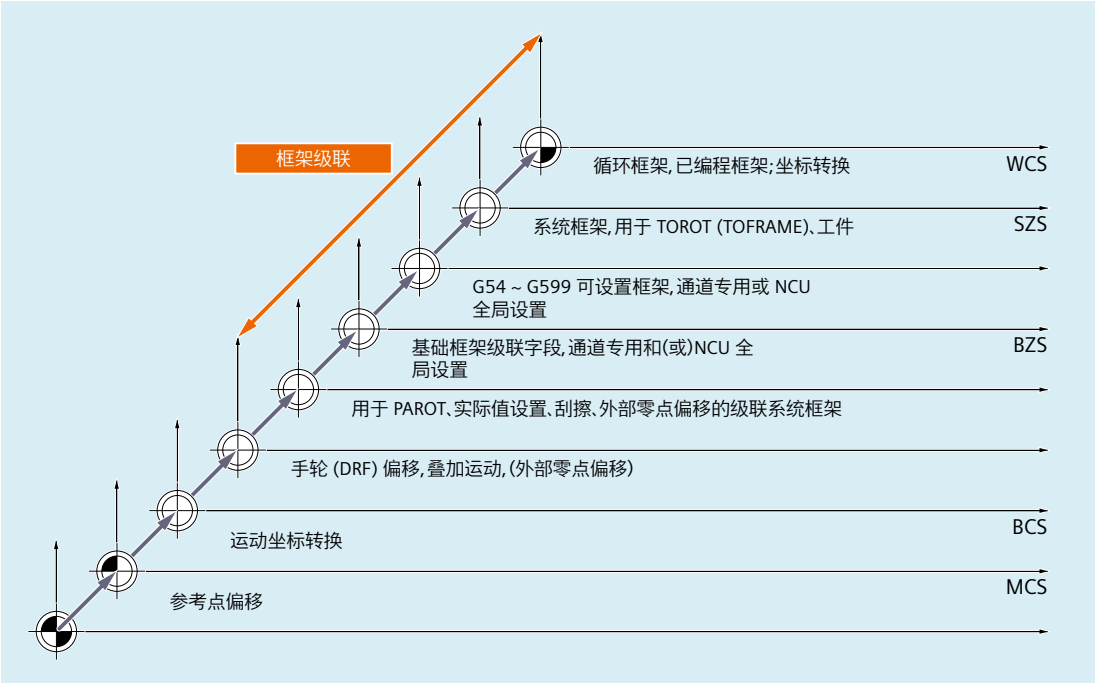
通过操作界面切换至 WCS 时，会触发以 WCS 原点为基准的轴位置显示。

切换至 MCS

通过操作界面切换至 MCS 时，会触发以 MCS 原点为基准的轴位置显示。

坐标系之间的关联

下图显示了从机床坐标系 MCS 到工作坐标系 WCS 之间的关联。



WCS　**W**orkpiece **C**oordinate **S**ystem，工件坐标系

:

SZS　**S**ettable **Z**ero **S**ystem，可设定的零点坐标系

:

BZS　**B**asic **Z**ero **S**ystem，基本零点坐标系

:

BCS　**B**asic **C**oordinate **S**ystem，基本坐标系

:

MCS　**M**aschine **C**oordinate **S**ystem，机床坐标系

:

图 4-27　框架级联

更多信息请参见“PLC 辅助功能输出 (页 775)”以及功能手册“*刀具*”一章中的“*刀具补偿*”。

更多信息

- NC 编程手册
- 功能手册之转换分册；运动转换

- 功能手册之进给轴和主轴；轴耦合
- 功能手册之工艺；切线控制

4.6.3 特殊响应

溢出转存

复位状态下对以下数据的溢出转存：

- 框架（零点偏移）
- 生效的平面
- 激活的转换
- 刀具补偿

立即作用于通道中所有轴的实际值显示。

通过操作面板前端输入

通过操作面板前端上的操作修改以下数值时：
“生效的框架”（零点偏移，“参数”操作区域）
和
“生效的刀具长度补偿”（“参数”操作区域）
则会通过下列措施之一在显示中激活修改：

- 按下 RESET 键
- 重新选择：
  - 零点偏移，通过零件程序
  - 刀具补偿，通过零件程序
- 重新设置：
  - 零点偏移，通过溢出转存
  - 刀具补偿，通过溢出转存
- 零件程序开始

MD9440

若设置了用于操作面板前端的 HMI 机床数据：
MD9440 ACTIVATE_SEL_USER_DATA
那么输入的值会在复位状态下立即生效。

在零件程序执行暂停的情形下进行输入时，数值会在继续运行开始时生效。

实际值读取

在激活框架（零点偏移）或刀具补偿后，若从 $AA_IW 读取 WCS 中的实际值，那么读取结果中将包含激活的修改，即便尚未通过激活的修改运行轴。

使用变量 $AA_IEN[轴] 可为每根轴从零件程序读取可设定零点坐标系 SZS 中的实际值。

使用 $AA_IBN[轴] 可为每根轴从零件程序读取基本零点坐标系 BZS 中的实际值。

实际值显示

WCS 中总是显示编写的轮廓。

以下偏移会被叠加至 MCS：

- 运动转换
- DRF 偏移/外部零点偏移
- 生效的框架
- 当前刀具的生效刀具补偿

通过 PLC 切换

通过 PLC 可选择以 WCS、SZS、BZS 或 MCS 显示实际值。可通过 PLC 设定，机床上的哪个坐标系等同于工件坐标系。

启动后的缺省坐标系为 MCS。
通过信号 `<Hmi>.basic.out.displayWcsValues` “MCS/WCS 切换”，也可通过 PLC 切换到 WCS。

传输至 PLC

通过设置以下机床数据：
MD20110 / MD20112，位 1
可设定在选择刀具长度补偿时是否将辅助功能（D、T、M）输出至 PLC。

说明

若通过 PLC 选择了 WCS，则可通过操作针对各运行方式在 WCS 和 WCS 间进行切换。
不过在运行方式和/或运行区域切换时，系统还是会检测和激活通过 PLC 选择的 WCS（参见“BAG、通道、程序运行、复位特性 (页 29)”章节）。

PLC 信号

OP → PLC

Basic Program Plus	Basic Program	
<Hmi>.basic.out.displayWcsValues	LBP_HMI.A_ActWCS	DB19.DBB0.7

4.7 边界条件

没有需要遵循的前提条件。

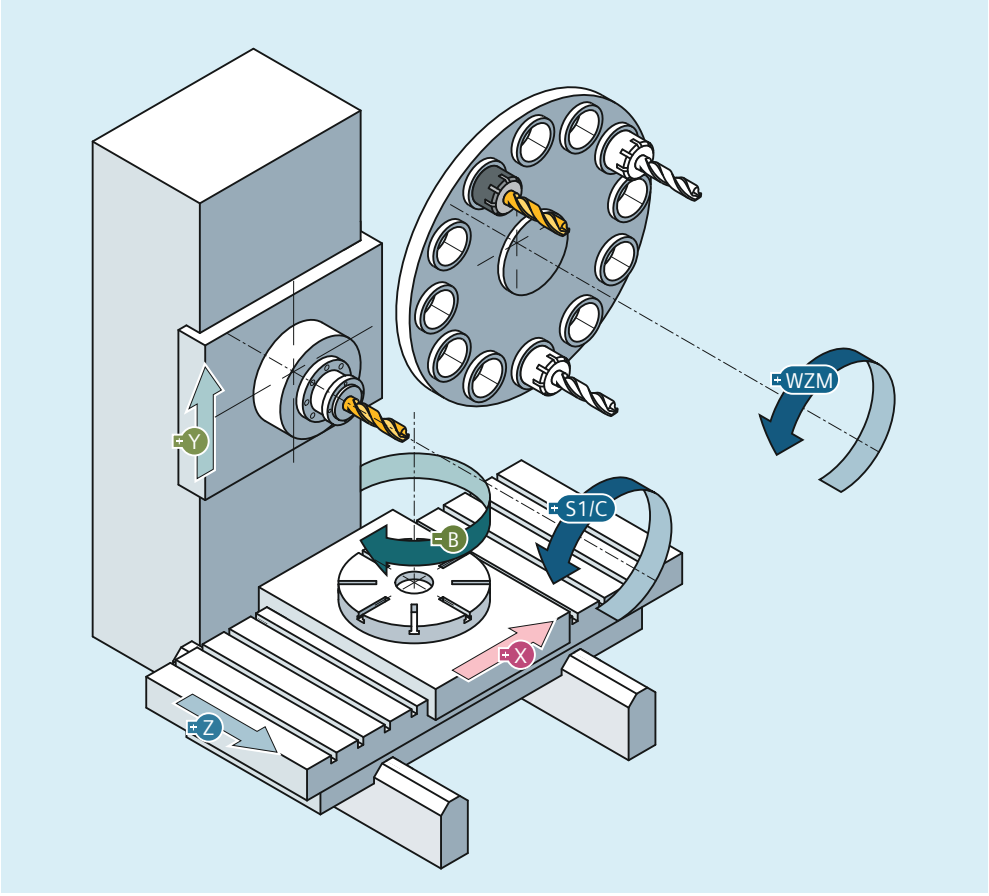
4.8 示例

4.8.1 轴

带回转台的 3 轴铣床的轴配置

机床轴 1：X1	线性轴
机床轴 2：Y1	线性轴
机床轴 3：Z1	线性轴
机床轴 4： B1	回转台（用于多面加工中的车削）
机床轴 5：W1	刀库的回转轴（刀具托盘）
机床轴 6： C1	（主轴）

几何轴 1：X	（通道 1）
几何轴 2：Y	（通道 1）
几何轴 3：Z	（通道 1）
辅助轴 1：B	（通道 1）
辅助轴 2： WZM	（通道 1）
主轴 1： S1/C	（通道 1）



机床数据的参数设置

机床数据	值
MD10000 AXCONF_MACHAX_NAME_TAB[0]	= X1
MD10000 AXCONF_MACHAX_NAME_TAB[1]	= Y1
MD10000 AXCONF_MACHAX_NAME_TAB[2]	= Z1

机床数据	值
MD10000 AXCONF_MACHAX_NAME_TAB[3]	= B1
MD10000 AXCONF_MACHAX_NAME_TAB[4]	= W1
MD10000 AXCONF_MACHAX_NAME_TAB[5]	= C1
MD20050 AXCONF_GEOAX_ASSIGN_TAB[0]	= 1
MD20050 AXCONF_GEOAX_ASSIGN_TAB[1]	= 2
MD20050 AXCONF_GEOAX_ASSIGN_TAB[2]	= 3
MD20060 AXCONF_GEOAX_NAME_TAB[0]	= X
MD20060 AXCONF_GEOAX_NAME_TAB[1]	= Y
MD20060 AXCONF_GEOAX_NAME_TAB[2]	= Z
MD20070 AXCONF_MACHAX_USED[0]	= 1
MD20070 AXCONF_MACHAX_USED[1]	= 2
MD20070 AXCONF_MACHAX_USED[2]	= 3
MD20070 AXCONF_MACHAX_USED[3]	= 4
MD20070 AXCONF_MACHAX_USED[4]	= 5
MD20070 AXCONF_MACHAX_USED[5]	= 6
MD20080 AXCONF_CHANAX_NAME_TAB[0]	= X
MD20080 AXCONF_CHANAX_NAME_TAB[1]	= Y
MD20080 AXCONF_CHANAX_NAME_TAB[2]	= Z
MD20080 AXCONF_CHANAX_NAME_TAB[3]	= B
MD20080 AXCONF_CHANAX_NAME_TAB[4]	= WZM
MD20080 AXCONF_CHANAX_NAME_TAB[5]	= S1
MD30300 IS_ROT_AX[3]	= 1
MD30300 IS_ROT_AX[4]	= 1
MD30300 IS_ROT_AX[5]	= 1
MD30310 ROT_IS_MODULO[3]	= 1

机床数据	值
MD30310 ROT_IS_MODULO[4]	= 1
MD30310 ROT_IS_MODULO[5]	= 1
MD30320 DISPLAY_IS_MODULO[3]	= 1
MD30320 DISPLAY_IS_MODULO[4]	= 1
MD20090 SPIND_DEF_MASTER_SPIND	= 1
MD35000 SPIND_ASSIGN_TO_MACHAX[AX1]	= 0
MD35000 SPIND_ASSIGN_TO_MACHAX[AX2]	= 0
MD35000 SPIND_ASSIGN_TO_MACHAX[AX3]	= 0
MD35000 SPIND_ASSIGN_TO_MACHAX[AX4]	= 0
MD35000 SPIND_ASSIGN_TO_MACHAX[AX5]	= 0
MD35000 SPIND_ASSIGN_TO_MACHAX[AX6]	= 1

4.8.2 坐标系

配置全局基本框架

需要一个有 2 个通道的 NC。其中：

- 两个通道均可写入全局基本框架。
- 重新激活全局基本框架后，另一通道能够识别出变化。
- 两个通道均可读取全局基本框架。
- 两个通道均可为自身激活全局基本框架。

机床数据

机床数据	值
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[0]	= X1
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[1]	= X2
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[2]	= X3
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[3]	= X4
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[4]	= X5
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[5]	= X6
MD18602 $MN_MM_NUM_GLOBAL_BASE_FRAMES	= 1
MD28081 $MC_MM_NUM_BASE_FRAMES	= 1

通道 1 的机床数据	值		通道 1 的机床数据	值
$MC_AXCONF_CHANAX_NAME_TAB[0]	= X		$MC_AXCONF_CHANAX_NAME_TAB[0]	= X
$MC_AXCONF_CHANAX_NAME_TAB[1]	= Y		$MC_AXCONF_CHANAX_NAME_TAB[1]	= Y
$MC_AXCONF_CHANAX_NAME_TAB[2]	= Z		$MC_AXCONF_CHANAX_NAME_TAB[2]	= Z
$MC_AXCONF_MACHAX_USED[0]	= 1		$MC_AXCONF_MACHAX_USED[0]	= 4
$MC_AXCONF_MACHAX_USED[1]	= 2		$MC_AXCONF_MACHAX_USED[1]	= 5
$MC_AXCONF_MACHAX_USED[2]	= 3		$MC_AXCONF_MACHAX_USED[2]	= 6
$MC_AXCONF_GEOAX_NAME_TAB[0]	= X		$MC_AXCONF_GEOAX_NAME_TAB[0]	= X
$MC_AXCONF_GEOAX_NAME_TAB[1]	= Y		$MC_AXCONF_GEOAX_NAME_TAB[1]	= Y
$MC_AXCONF_GEOAX_NAME_TAB[2]	= Z		$MC_AXCONF_GEOAX_NAME_TAB[2]	= Z
$MC_AXCONF_GEOAX_ASSIGN_TAB[0]	= 1		$MC_AXCONF_GEOAX_ASSIGN_TAB[0]	= 4
$MC_AXCONF_GEOAX_ASSIGN_TAB[1]	= 2		$MC_AXCONF_GEOAX_ASSIGN_TAB[1]	= 5
$MC_AXCONF_GEOAX_ASSIGN_TAB[2]	= 3		$MC_AXCONF_GEOAX_ASSIGN_TAB[2]	= 6

第 1 通道中的通道

代码（节选）：	; 注释
. . .	
N100 $P_NCBFR[0] = CTRANS(x, 10)	; 激活 NC 全局基本框架
. . .	

代码（节选）：	; 注释
N130 $P_NCBFRAME[0] = CROT(X, 45)	; 激活包含旋转的 NC 全局基本框架 => ; 报警 18310，因为 NC 全局框架中 ; 不允许有旋转
. . .	

第 2 通道中的通道

代码（节选）：	; 注释
. . .	
N100 $P_NCBFR[0] = CTRANS(x, 10)	; NC 全局基本框架也在通道 2 中生效
. . .	
N510 G500 X10	; 激活基本框架
N520 $P_CHBFRAME[0] = CTRANS(x, 10)	通道 2 的当前框架通过偏移激活
. . .	

4.8.3 框架

示例 1

需通过几何轴切换将通道轴设定为几何轴。

切换后，可编程框架包含 x 轴方向上 10 的偏移分量。

保留当前可设定框架。

FRAME_GEOX_CHANGE_MODE = 1

程序代码	注释
$P_UIFR[1] = CROT(X,10,Y,20,Z,30)	; 几何轴切换后保留框架
G54	; 可设定框架生效
TRANS A10	; 切换时附带轴 a 的轴偏移
GEOAX(1,A)	; 将轴 A 设定为 x 轴
	; $P_ACTFRAME = CROT(X,10,Y,20,Z,30) :CTRANS(X10)

坐标转换切换时，可同时将多个通道轴设定为几何轴。

示例 2

通过 5 轴定向转换，通道轴 4、5、6 成为转换的几何轴。即在转换前替换所有几何轴。

激活转换时，所有当前框架相应变化。

为了计算新的 WCS 坐标系，将成为几何轴的通道轴的轴框架分量纳入计算。转换前编写的旋转保留。取消转换后，重新恢复为原先的 WCS。

最常见应用就是使几何轴在坐标转换前后不发生变化，并保留转换前生效的框架。

机床数据：

$MN_FRAME_GEOAX_CHANGE_MODE = 1

$MC_AXCONF_CHANAX_NAME_TAB [0] = "CAX"
$MC_AXCONF_CHANAX_NAME_TAB [1] = "CAY"
$MC_AXCONF_CHANAX_NAME_TAB [2] = "CAZ"
$MC_AXCONF_CHANAX_NAME_TAB [3] = "A"
$MC_AXCONF_CHANAX_NAME_TAB [4] = "B"
$MC_AXCONF_CHANAX_NAME_TAB [5] = "C"

$MC_AXCONF_GEOAX_ASSIGN_TAB [0] = 1
$MC_AXCONF_GEOAX_ASSIGN_TAB [1] = 2
$MC_AXCONF_GEOAX_ASSIGN_TAB [2] = 3

$MC_AXCONF_GEOAX_NAME_TAB [0] = "X"
$MC_AXCONF_GEOAX_NAME_TAB [1] = "Y"
$MC_AXCONF_GEOAX_NAME_TAB [2] = "Z"

$MC_TRAFO_GEOAX_ASSIGN_TAB_1 [0] = 4
$MC_TRAFO_GEOAX_ASSIGN_TAB_1 [1] = 5
$MC_TRAFO_GEOAX_ASSIGN_TAB_1 [2] = 6

$MC_TRAFO_AXES_IN_1 [0] = 4
$MC_TRAFO_AXES_IN_1 [1] = 5
$MC_TRAFO_AXES_IN_1 [2] = 6
$MC_TRAFO_AXES_IN_1 [3] = 1
$MC_TRAFO_AXES_IN_1 [4] = 2

程序：

程序代码	注释
$P_NCBFRAME[0] = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6)	
$P_CHBFRAME[0] = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6)	
$P_IFRAME = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6) :CROT(Z,45)	
$P_PFRAME = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6) :CROT(X,10,Y,20,Z,30)	
TRAORI	; 坐标转换，设置 GeoAx(4,5,6)
	; $P_NCBFRAME[0] = CTRANS(X,4,Y,5,Z,6,CAX,1,CAY,2,CAZ,3)
	; $P_ACTBFRAME = CTRANS(X,8,Y,10,Z,12,CAX,2,CAY,4,CAZ,6)
	; $P_PFRAME = CTRANS(X,4,Y,5,Z,6,CAX,1,CAY,2,CAZ,3) :CROT(X,10,Y,20,Z,30)
	; $P_IFRAME = CTRANS(X,4,Y,5,Z,6,CAX,1,CAY,2,CAZ,3) :CROT(Z,45)
TRAFOOF	; 取消转换，设置 GEOAX (1,2,3)
	; $P_NCBFRAME[0] = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6)
	; $P_CHBFRAME[0] = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6)
	; $P_IFRAME = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6) :CROT(Z,45)
	; $P_PFRAME = CTRANS(X,1,Y,2,Z,3,A,4,B,5,C,6) :CROT(X,10,Y,20,Z,30)

运动链 5

5.1 功能说明

5.1.1 特性

本章介绍如何借助运动链为“防撞保护”或“运动坐标转换”等 NC 功能构建机床运动结构，以及如何通过系统变量在控制器上进行编程。

系统变量以掉电保持方式保存在 NC 中，可通过 SINUMERIK Operate 的调试归档功能将其归档为“NC 数据”，也可读取这些数据。

需要机床几何描述的“防撞保护”等功能，参见*监控和补偿*功能手册，*机床几何建模*章节。

说明

图形编辑器

作为在零件程序中描述系统变量这种方法的备选方案，可以通过 SINUMERIK Operate 操作界面进行机床建模：

操作区域：“调试” > “NC” > “机床模型”

更改机床模型

只有调用函数 PROTA() 或 PROTS()，发出重新计算机床模型的显式请求后，直接干预系统变量实现的机床模型更改才会显示在操作界面上。

通过操作界面实现的机床模型更改，会立刻应用在 NC 系统变量中。但是只有调用函数 PROTA() 或 PROTS()，发出重新计算机床模型的显式请求后，这些更改才会生效。

运动结构

机床运动结构由以下各部分构成：

- 机床轴的数量和类型：直线轴或回转轴
- 机床轴的布局：位置和定向
- 视具体机床轴而定，不尽相同：机床轴与其他哪个机床轴一起运行。

运动链

机床运动结构描述通过具有下列属性的运动链实现：

- 运动链由任意数量、相互连接的元素构成。
- 一个运动链可以衍生出多个平行的子链。

- 控制器中始终只有一个运动链处于活动状态。
- 活动的运动链以根元素作为开头。
- 未与根元素连接在一起，或者连接指向根元素的编程设置的元素或子链，不是当前有效运动链的构成部分。
- 运动链在空间确定的坐标系中定义。

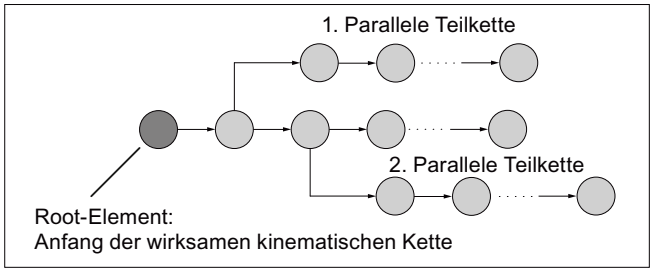


图 5-1　运动链示例

元素

从本质上来说，运动链元素描述的是上一个元素的本地坐标系映射到当前元素本地坐标系的方法，也就是坐标转换。

$K_{n-1} \Rightarrow T_n \Rightarrow K_n$

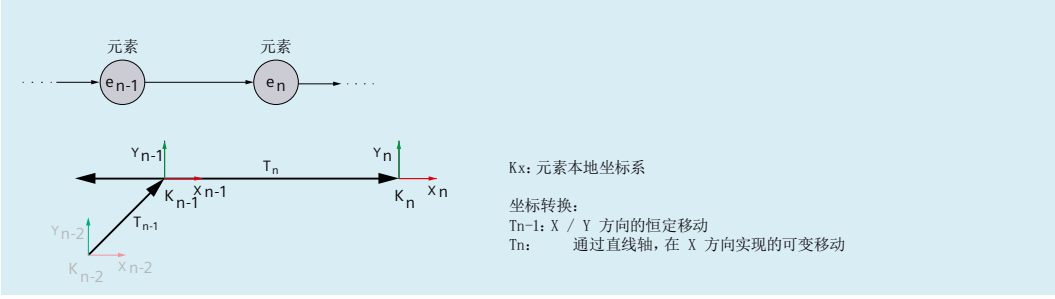


图 5-2　元素本地坐标系

支持以下恒定坐标转换方式：

- 平移（类型：OFFSET (页 478)）
- 旋转（类型：ROT_CONST (页 476)）

支持下列基于分配给元素的机床轴（直线轴或回转轴）的当前位置值且可变的坐标转换：

- 平移（类型：AXIS_LIN (页 469)）
- 旋转（类型：AXIS_ROT (页 472)）

如果某个元素因相关机床轴的位置发生变化而出现位置或定向改变，这种情况下运动链或平行子链上的所有**后续**元素都会受影响。

可通过机床数据 (页 462)编程设置可能的元素最大数量。

平行子链

如果从元素 e_n 分出一个平行子链，则在运动学方面，子链始终在元素**前面**分支。如果由于相关机床轴的位置发生变化等原因，导致元素 e_n 出现变动，这种情况下分出的子链**不**受影响。

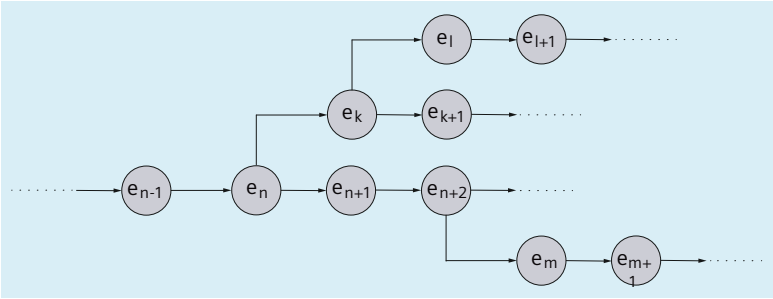


图 5-3　从元素 e_n 和 e_{n+2} 分出的平行子链

开关

开关是运动链中在运动学方面无效的特殊元素（类型：SWITCH (页 480)），具有“开”和“关”两种状态。

在**关**状态下，前一个元素和后一个元素之间的连接会被切断。

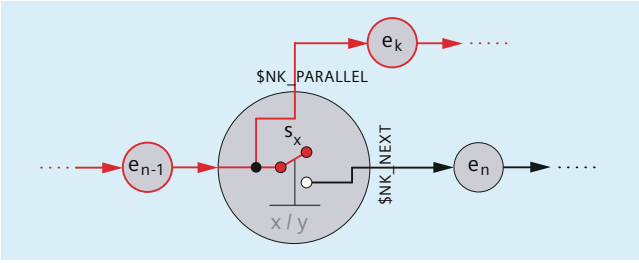


图 5-4　“关”状态

在**开**状态下，前一个元素与后一个元素连接在一起。

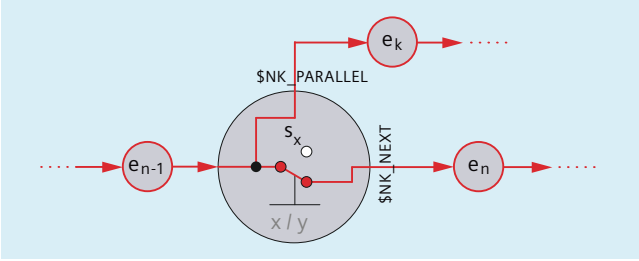


图 5-5　“开”状态

与平行元素的连接不受开关影响。

可通过机床数据 (页 463)编程设置可能的开关最大数量。

说明

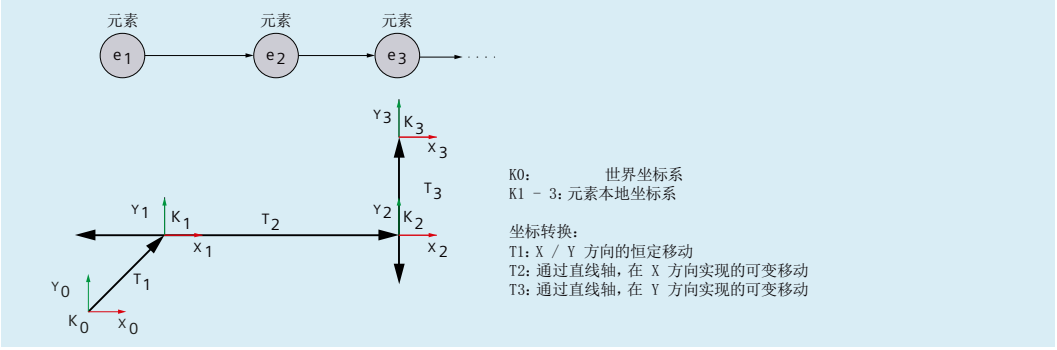
本地坐标系

与世界坐标系不同，开关的本地坐标系不旋转。

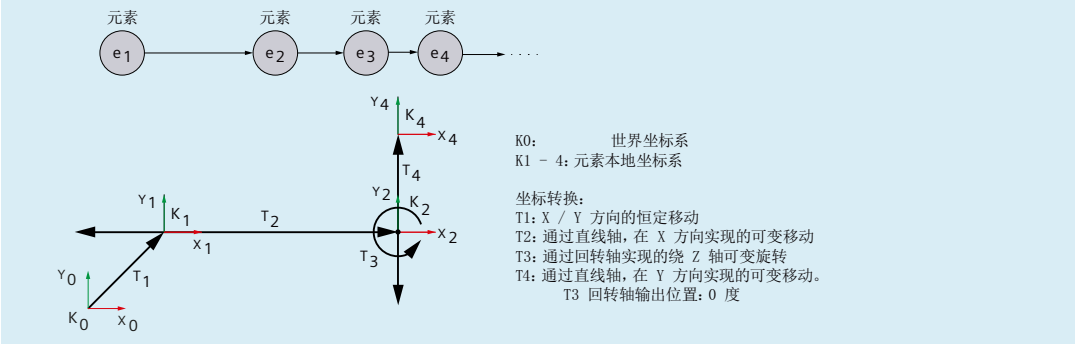
世界坐标系

为了明确描述机床的运动结构，运动链所有元素的机床轴定向矢量和偏移矢量以及恒定旋转或平移，都基于世界坐标系进行。

具有**直线轴**的元素不改变后续元素的定向。因此，在没有其他边界条件的情况下，后续元素的定向和偏移矢量均基于世界坐标系。



具有**回转轴**的元素改变后续元素的定向。因此，在前一个定向发生改变的元素的指定起始位置下，后续元素的定向和偏移矢量必须基于世界坐标系。



在根元素之前定义的运动链元素，可以自由选择世界坐标系的原点和定向。从根元素开始生效的坐标系，必须遵循下列规则：

- 世界坐标系的原点在机床零点
- 定向世界坐标系时，注意将正向坐标轴分配给机床的主要直线轴

方向矢量

在运动链中，用于规定机床轴定向的方向矢量必须以绝对形式指定，即基于世界坐标系指定。

5.2 调试

5.2.1 一般信息

5.2.1.1 简介

借助以下各项调试“运动链”这项功能：

- 机床数据
  - 规定数量框架
  - 规定运动链的第一个元素
- 系统变量
  - 规定元素的运动属性
  - 连接元素与运动链

5.2.1.2 构建系统变量

按照下列各式构建系统变量：

- **$NK_**<Name>[<Index_1>]
- **$NK_**<Name>[<Index_1>, <Index_2>]

一般信息

用于描述运动链各元素的系统变量具有以下属性：

- 运动链的所有系统变量，均以 **$NK_** 作为前缀（N 代表 NC，K 代表运动）。
- 系统变量可通过 NC 程序读取和写入。
- 系统变量可归档备份，以备后期重新读入 NC。

数据类型

STRING

所有数据类型为 STRING 的系统变量都具有下列属性：

- 最大字符串长度：31 个字符
- 不区分大小写
  示例：“Achse1”（轴 1）与“ACHSE1”（轴 1）意义相同
- 允许使用空格和特殊字符
  示例：“Achse1”（轴 1）与“Achse 1”（轴 1）意义不同
- 以**两**条下划线“__”**开头**的名称，是为系统预留的名称，**不**可用于用户自定义名称。

说明

以空格开头

由于空格是有效且可区分的字符，因此原则上以**空格开头**，后跟**两**条下划线“__”的名称可用于用户自定义名称。但是因为容易与系统名称混淆，因此**不**建议如此命名。

Index_1

各元素通过 Index_1 寻址。索引 0 → 第 1 个元素，索引 1 → 第 2 个元素，... n → 第 (n+1) 个元素，其中 n = ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)

同一元素的所有系统变量使用相同的索引。

Index_2

对于包含矢量的系统变量，其矢量坐标通过 Index_2 寻址。

- 0 → X 轴
- 1 → Y 轴
- 2 → Z 轴

5.2.2 机床数据

5.2.2.1 元素最大数量

通过机床数据为运动链设置元素最大数量：

MD18880 $MN_MM_MAXNUM_KIN_CHAIN_ELEM = <数量>

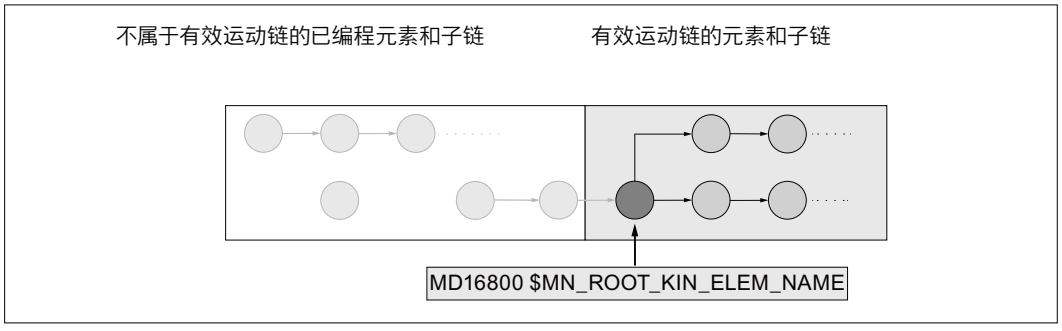
5.2.2.2 根元素

通过机床数据定义当前有效运动链的根元素，即第一个元素。

MD16800 $MN_ROOT_KIN_ELEM_NAME = "<Element_Name>"

编程设置的所有与根元素相连，且连接从根元素开始的元素和子链，都是当前有效运动链的组成部分。

未与根元素连接在一起，或者连接指向根元素的编程设置的元素或子链，不是当前有效运动链的构成部分。



5.2.2.3 开关最大数量

通过机床数据为运动链设置开关最大数量

MD18882 $MN_MM_MAXNUM_KIN_SWITCHES = <数量>

5.2.3 系统变量

5.2.3.1 简介

不受元素影响的系统变量

系统变量	含义
$NK_SWITCH	用于打开和关闭开关的开关变量

元素专用系统变量

元素专用系统变量氛围类型无关变量和类型相关变量两种：

- 类型无关变量

系统变量	含义
$NK_NAME	当前元素 e_n 的名称
$NK_NEXT	下一个元素 e_{n+1} 的名称
$NK_PARALLEL	在当前元素 e_n **前面**分支的平行元素 e_p 的名称
$NK_TYPE	元素的类型

- 类型相关变量

系统变量	含义
$NK_OFF_DIR	偏移矢量或方向矢量
$NK_AXIS	机床轴名称或对象名称
$NK_A_OFF	直线轴或回转轴的零点偏移
$NK_SWITCH_INDEX	开关索引
$NK_SWITCH_POS	开关位置“开”

类型相关变量在以下类型下进行处理：

系统变量	类型				
	OFFSET	AXIS_LIN	AXIS_ROT	ROT_CONST	SWITCH
$NK_OFF_DIR	x	x	x	x	-
$NK_AXIS	-	x	x	-	-
$NK_A_OFF	-	x	x	x	-
$NK_SWITCH_INDEX	-	-	-	-	x
$NK_SWITCH_POS	-	-	-	-	x

系统变量详细信息见下文。

说明

创建定义的输出状态

建议在开始编程设置运动链之前，先生成定义的输出状态。为此需使用 DELOBJ() 函数，将运动链的系统变量设为缺省值。

更改系统变量值

如更改上述系统变量的值，更改后的新值会立即显示在 SINUMERIK Operate 等操作界面上。但是只有调用函数 PROTA() 或 PROTS()，发出重新计算机床模型的显式请求后，才会更新 NC 的机床模型。

参见

删除组件 (DELOBJ) (页 483)

5.2.3.2 $NK_NAME

功能

在系统变量中输入元素名称，此名称在 NC 范围内具有唯一性。通过这个名称，在运动链等范围内对元素执行回参考点处理。名称也显示在 SINUMERIK Operate 的图形编辑器中。

句法

```
$NK_NAME[<n>] = "<Name>"
```

含义

$NK_NAME:	元素名称	
	数据类型：	STRING
	缺省值：	""（空字符串）
<n>:	系统变量索引或元素索引	
	数据类型：	INT
	取值范围：	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
<Name>:	元素名称，最大字符串长度：31 个字符	
	数据类型：	STRING

示例

为第 9 个运动元素分配名称“B 轴”：

程序代码	注释
`N100 $NK_NAME[8] = "B 轴"`	`; 第 9 个运动元素，` `; 名称 = "B 轴"`

5.2.3.3 $NK_NEXT

功能

如果元素是运动链的一部分，则在系统变量中输入后续元素的名称。

句法

```
$NK_NEXT[<n>] = "<Name>"
```

含义

NK_NEXT：	下一个元素的名称	
	数据类型：	STRING
	取值范围：	$NK_NAME (页 465) 中包含的所有名称
	缺省值：	""（空字符串）
<n>：	系统变量索引或元素索引	
	数据类型：	INT
	取值范围：	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
<Name>：	元素名称，最大字符串长度：31 个字符	
	数据类型：	STRING

示例

第 9 个运动元素无后续元素：

程序代码	注释
`N100 $NK_NEXT[8] = ""`	`; 第 9 个运动元素，` `; 后续元素 = ""`

5.2.3.4 $NK_PARALLEL

功能

在系统变量中输入在当前元素**前面**分支的元素名称。分支元素平行于当前元素。当前元素的变动，比如分配的机床轴发生位置改变等，对平行元素不产生影响。

句法

```
$NK_PARALLEL[<n>] = "<Name>"
```

含义

$NK_PARALLEL:	平行元素的名称	
	数据类型：	STRING
	取值范围：	$NK_NAME (页 465) 中包含的所有名称
	缺省值：	""（空字符串）
<n>：	系统变量索引或元素索引	
	数据类型：	INT
	取值范围：	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
<Name>：	元素名称，最大字符串长度：31 个字符	
	数据类型：	STRING

示例

元素“向回转台的偏移”平行于第 9 个运动元素：

程序代码	注释
N100 $NK_PARALLEL[8] = "回转台偏移"	; 平行于第 9 个运动元素， ; 名称 = "向回转台的偏移"

5.2.3.5 $NK_TYPE

功能

在系统变量中输入元素类型：

类型	说明
AXIS_LIN (页 469)	元素描述具有方向矢量 $NK_OFF_DIR 和零点偏移 $NK_A_OFF 的直线机床轴（直线轴）
AXIS_ROT (页 472)	元素描述具有方向矢量 $NK_OFF_DIR 和零点偏移 $NK_A_OFF 的旋转机床轴（回转轴），或者描述主轴。 **提示** 视运动链所用功能而定，主轴的位置各有不同： • 防撞保护：位置不明确 • 运动转换：取决于 $NT_CNTRL, Bit 1-3 的设置，位置对应 $NK_A_OFF 或转速设定值
ROT_CONST (页 476)	元素描述围绕方向矢量 $NK_OFF_DIR 旋转角度 $NK_A_OFF 的恒定旋转
OFFSET (页 478)	元素描述具有偏移矢量 $NK_OFF_DIR 的恒定直线偏移
SWITCH (页 480)	元素描述通过系统变量 $NK_SWITCH (页 482) 进行切换的开关。

句法

```
$NK_TYPE[<n>] = "<Typ>"
```

含义

$NK_TYPE：	元素的类型	
	数据类型：	STRING
	缺省值：	""（空字符串）
<n>：	系统变量索引或元素索引	
	数据类型：	INT
	取值范围：	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)

<Typ>:	元素名称，最大字符串长度：31 个字符	
	数据类型：	STRING
	取值范围：	"AXIS_LIN", "AXIS_ROT", "ROT_CONST", "OFFSET", "SWITCH"

示例

第 9 个运动元素是回转轴：

程序代码	注释
N100 $NK_TYPE[8] = "AXIS_ROT"	; 第 9 个元素 ; 类型 = 回转轴

5.2.3.6 $NK_TYPE = "AXIS_LIN" 时的类型相关变量

$NK_OFF_DIR

功能

在系统变量中输入分配给元素的直线轴 $NK_AXIS 沿着移动的方向矢量。从输入坐标系出发，根据直线轴的当前位置值和在 $NK_A_OFF 中指定的零点偏移进行移动，由此得到输出坐标系。

边界条件：

- 方向矢量以绝对值形式指定，即基于**世界坐标系**， 定义。
- 指定方向矢量时，注意矢量要指向机床轴的正运行方向。
- 方向矢量的大小必须超过 $1*10^{-6}$。

句法

```
$NK_OFF_DIR[<n>,<k>] = <值>
```

含义

$NK_OFF_DIR:	方向矢量 (X; Y; Z)	
	数据类型：	REAL
	取值范围：	方向矢量：$1*10^{-6} < |$矢量$| \leq$ 最大 REAL 值
	缺省值：	(0.0, 0.0, 0.0)

<n>：	系统变量索引或元素索引	
	数据类型：	INT
	取值范围：	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
<k>：	坐标索引	
	数据类型：	INT
	取值范围：	0:X 坐标（横坐标） 1:Y 坐标（纵坐标） 2：Z 坐标（第三坐标轴）
<值>：	坐标值	
	数据类型：	REAL
	取值范围：	- 最大 REAL 值 ≤ x ≤ + 最大 REAL 值

示例

第 9 个元素的直线轴沿方向矢量移动。方向矢量是相对于世界坐标系，在 X/Y 平面以 γ=90° 角度绕 Z 轴旋转，在 Y/Z 平面以 α=10° 角度绕 X 轴旋转的单位矢量 (1; 0; 0)。据此确定方向矢量各分量 (x, y, z) 的值：

- x = cos(γ) * cos(α) = cos(90) * cos(10) = 0.0
- y = sin(γ) * cos(α) = sin(90) * cos(10) ≈ 0.985
- z = sin(α) = sin(10) ≈ 0.174

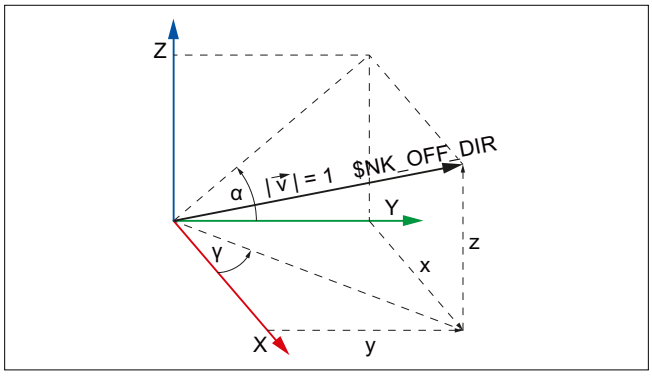


图 5-6　一般方向矢量

程序代码	注释
; 第 9 个运动元素	
N100 $NK_OFF_DIR[8,0] = COS(90)*COS(10)	; 0 = X 分量
N110 $NK_OFF_DIR[8,1] = SIN(90)*COS(10)	; 1 = Y 分量
N120 $NK_OFF_DIR[8,2] = SIN(10)	; 2 = Z 分量

$NK_AXIS

功能

在系统变量中输入分配给元素的机床轴 (MD10000 $MN_AXCONF_MACHAX_NAME_TAB) 的名称。

从输入坐标系出发，根据机床轴在机床坐标系中的当前额定位置和在 $NK_A_OFF 中指定的偏移进行移动，由此得到输出坐标系。机床轴额定位置中包含所有生效的零点偏移和叠加。

根据在 $NK_TYPE 中输入的类型 AXIS_LIN，机床轴必须是直线轴：

MD30300 $MA_IS_ROT_AX = 0

句法

```
$NK_AXIS[<n>] = <Name>
```

含义

$NK_AXIS:	机床轴名称	
	数据类型：	STRING
	取值范围：	机床轴名称
	缺省值：	""（空字符串）
<n>:	系统变量索引或元素索引	
	数据类型：	INT
	取值范围：	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
<值>:	机床轴名称	
	数据类型：	STRING
	取值范围：	机床轴名称

示例

第 9 个运动元素作为直线轴分配给了名为 V1 的机床轴

程序代码	注释
N100 $NK_AXIS[8] = "V1"	; 第 9 个元素 ; 轴 = 机床轴 V1

$NK_A_OFF

功能

可在系统变量中，为分配的机床轴 ($NK_AXIS) 输入一个附加的零点偏移。此零点偏移只在运动链内有效。

句法

```
$NK_A_OFF[<n>] = <值>
```

含义

$NK_A_OFF:	零点偏移	
	数据类型：	REAL
	取值范围：	- 最大 REAL 值 ≤ x ≤ ± 最大 REAL 值
	缺省值：	0.0
<n>:	系统变量索引或元素索引	
	数据类型：	INT
	取值范围：	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
<值>:	偏移值	
	数据类型：	REAL
	取值范围：	- 最大 REAL 值 ≤ x ≤ ± 最大 REAL 值

示例

第 9 个运动元素的直线轴零点相对于建模的运动偏移了 30.0 mm。

解释数值时采用哪个单位，取决于当前输入体系（英寸或公制）

程序代码	注释
`N100 $NK_A_OFF[8] = 30.0`	`; 第 9 个元素` `; 零点偏移 = 30.0 mm`

5.2.3.7 $NK_TYPE = "AXIS_ROT" 时的类型相关变量

$NK_OFF_DIR

功能

在系统变量中输入分配给元素的回转轴 $NK_AXIS 围绕其旋转的方向矢量。从输入坐标系出发，根据回转轴的当前位置值和在 $NK_A_OFF 中指定的零点偏移，围绕方向矢量 $NK_OFF_DIR 进行旋转，以此来计算输出坐标系。

边界条件：

- 方向矢量以绝对值形式指定，即基于**世界坐标系**， 定义。
- 必须根据“右手定则”规定方向矢量，当回转轴正向旋转时，拇指指向矢量方向。
- 方向矢量的大小必须超过 $1*10^{-6}$。

说明

主轴

如果分配的机床轴是主轴，视具体功能而定，位置不尽相同：

- 防撞保护：位置不明确
- 运动转换：取决于 $NT_CNTRL, Bit 1-3 的设置
  - Bit x == 0 → 位置不明确
  - Bit x == 1 → 当前位置值 + $NK_A_OFF

句法

```
$NK_OFF_DIR[<n>,<k>] = <值>
```

含义

$NK_OFF_DIR:	方向矢量 (X; Y; Z)	
	数据类型：	REAL
	取值范围：	方向矢量：$1*10^{-6}$ < |矢量| ≤ 最大 REAL 值
	缺省值：	(0.0, 0.0, 0.0)
<n>：	系统变量索引或元素索引	
	数据类型：	INT
	取值范围：	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
<k>：	坐标索引	
	数据类型：	INT
	取值范围：	0:X 坐标（横坐标） 1:Y 坐标（纵坐标） 2：Z 坐标（第三坐标轴）
<值>：	坐标值	
	数据类型：	REAL
	取值范围：	- 最大 REAL 值 ≤ x ≤ + 最大 REAL 值

示例

第 9 个元素的回转轴围绕方向矢量旋转。方向矢量是相对于世界坐标系，在 X/Y 平面以 γ=90° 角度绕 Z 轴旋转，在 Y/Z 平面以 α=10° 角度绕 X 轴旋转的单位矢量 (1; 0; 0)。据此确定方向矢量各分量 (x, y, z) 的值：

- x = cos(γ) * cos(α) = cos(90) * cos(10) = 0.0
- y = sin(γ) * cos(α) = sin(90) * cos(10) ≈ 0.985
- z = sin(α) = sin(10) ≈ 0.174

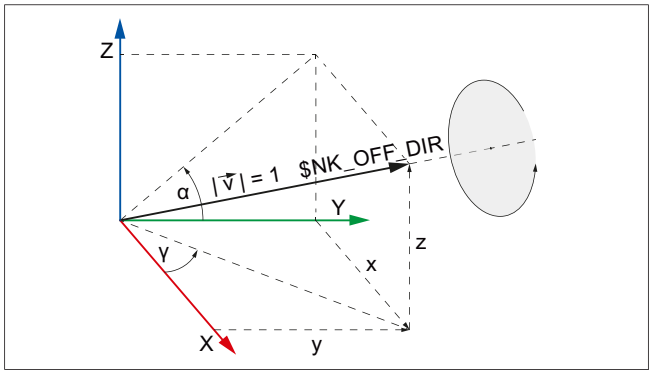


图 5-7　一般方向矢量

程序代码	注释
; 第 9 个运动元素	
N100 $NK_OFF_DIR[8,0] = COS(90)*COS(10)	; 0 = X 分量
N110 $NK_OFF_DIR[8,1] = SIN(90)*COS(10)	; 1 = Y 分量
N120 $NK_OFF_DIR[8,2] = SIN(10)	; 2 = Z 分量

$NK_AXIS

功能

在系统变量中输入分配给元素的机床轴 (MD10000 $MN_AXCONF_MACHAX_NAME_TAB) 的名称。

从输入坐标系出发，根据机床轴在机床坐标系中的当前额定位置和在 $NK_A_OFF 中指定的偏移进行旋转，由此得到元素的输出坐标系，机床轴额定位置中包含所有生效的零点偏移和叠加。

机床轴类型必须与在 $NK_TYPE 中输入的类型一致：

根据在 $NK_TYPE 中输入的类型 AXIS_ROT，机床轴必须是回转轴：

MD30300 $MA_IS_ROT_AX = 1

句法

```
$NK_AXIS[<n>] = <Name>
```

含义

$NK_AXIS:	机床轴名称	
	数据类型：	STRING
	取值范围：	机床轴名称
	缺省值：	""（空字符串）
<n>：	系统变量索引或元素索引	
	数据类型：	INT
	取值范围：	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
<值>：	机床轴名称	
	数据类型：	STRING
	取值范围：	机床轴名称

示例

第 9 个运动元素作为回转轴分配给了名为 B1 的机床轴。

程序代码	注释
N100 $NK_AXIS[8] = "B1"	; 第 9 个元素 ; 轴 = 机床轴 B1

$NK_A_OFF

功能

可在系统变量中，为分配的机床轴 ($NK_AXIS) 输入一个附加的零点偏移。此零点偏移只在运动链内有效。

句法

```
$NK_A_OFF[<n>] = <值>
```

含义

$NK_A_OFF:	零点偏移	
	数据类型：	REAL
	取值范围：	- 最大 REAL 值 ≤ x ≤ ± 最大 REAL 值
	缺省值：	0.0

<n>：	系统变量索引或元素索引	
	数据类型：	INT
	取值范围：	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
<值>：	偏移值	
	数据类型：	REAL
	取值范围：	- 最大 REAL 值 ≤ x ≤ ± 最大 REAL 值

示例

第 9 个运动元素的回转轴零点相对于建模的运动偏移了 30.0°。

程序代码	注释
`N100 $NK_A_OFF[8] = 30.0`	`; 第 9 个元素` `; 零点偏移 = 30.0°`

5.2.3.8 $NK_TYPE = "ROT_CONST" 时的类型相关变量

$NK_OFF_DIR

功能

在系统变量中输入围绕其进行恒定旋转的方向矢量。从输入坐标系出发，以在 $NK_A_OFF 中指定的角度围绕方向矢量 $NK_OFF_DIR 旋转，以此来计算输出坐标系。

边界条件：

- 方向矢量以绝对值形式指定，即基于**世界坐标系**， 定义。
- 必须根据“右手定则”规定方向矢量，当回转轴正向旋转时，拇指指向矢量方向。
- 方向矢量的大小必须超过 $1*10^{-6}$。

句法

```
$NK_OFF_DIR[<n>,<k>] = <值>
```

含义

$NK_OFF_DIR：	方向矢量 (X; Y; Z)	
	数据类型：	REAL
	取值范围：	方向矢量：$1*10^{-6}$ < |矢量| ≤ 最大 REAL 值
	缺省值：	(0.0, 0.0, 0.0)

<n>：	系统变量索引或元素索引	
	数据类型：	INT
	取值范围：	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
<k>：	坐标索引	
	数据类型：	INT
	取值范围：	0:X 坐标（横坐标） 1:Y 坐标（纵坐标） 2：Z 坐标（第三坐标轴）
<值>：	坐标值	
	数据类型：	REAL
	取值范围：	- 最大 REAL 值 ≤ x ≤ + 最大 REAL 值

示例

从输入坐标系出发，以在 $NK_A_OFF 中指定的角度围绕方向矢量旋转，得到第 9 个元素的输出坐标系。方向矢量是相对于世界坐标系，在 X/Y 平面以 γ=90° 角度绕 Z 轴旋转，在 Y/Z 平面以 α=10° 角度绕 X 轴旋转的单位矢量 (1; 0; 0)。据此确定方向矢量各分量 (x, y, z) 的值：

- x = cos(γ) * cos(α) = cos(90) * cos(10) = 0.0
- y = sin(γ) * cos(α) = sin(90) * cos(10) ≈ 0.985
- z = sin(α) = sin(10) ≈ 0.174

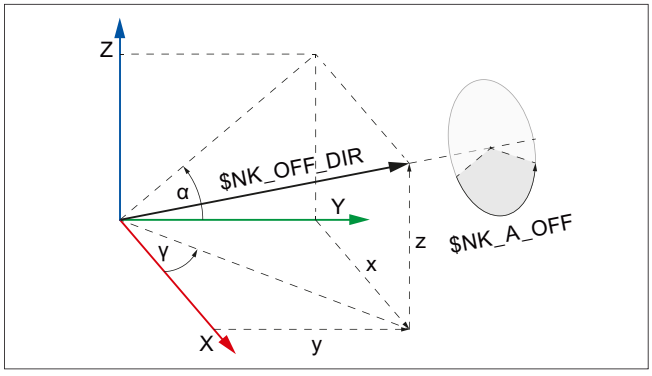


图 5-8　ROT_CONST

程序代码	注释
; 第 9 个运动元素	
N100 $NK_OFF_DIR[8,0] = COS(90)*COS(10)	; 0 = X 分量
N110 $NK_OFF_DIR[8,1] = SIN(90)*COS(10)	; 1 = Y 分量
N120 $NK_OFF_DIR[8,2] = SIN(10)	; 2 = Z 分量

$NK_A_OFF

功能

在系统变量中输入相对于输入坐标系，输出坐标系围绕方向矢量 $NK_OFF_DIR 旋转的角度。

句法

```
$NK_A_OFF[<n>] = <值>
```

含义

$NK_A_OFF:	旋转角度	
	数据类型：	REAL
	取值范围：	- 最大 REAL 值 ≤ x ≤ + 最大 REAL 值
	缺省值：	0.0
<n>:	系统变量索引或元素索引	
	数据类型：	INT
	取值范围：	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
<值>:	角	
	数据类型：	REAL
	取值范围：	- 最大 REAL 值 ≤ x ≤ + 最大 REAL 值

示例

第 9 个运动元素的旋转角度为 30.0°。

程序代码	注释
N100 $NK_A_OFF[8] = 30.0	; 第 9 个元素 ; 旋转角度 = 30.0°

5.2.3.9 $NK_TYPE = "OFFSET" 时的类型相关变量

$NK_OFF_DIR

功能

在系统变量中输入相对于输入坐标系，输出坐标系移动的偏移矢量。

偏移矢量以绝对值形式指定，即基于**世界坐标系**， 定义。

句法

```
$NK_OFF_DIR[<n>,<k>] = <值>
```

含义

$NK_OFF_DIR :	偏移矢量 (X; Y; Z)	
	数据类型：	REAL
	取值范围：	（- 最大 REAL 值）≤ x ≤（+ 最大 REAL 值）
	缺省值：	(0.0, 0.0, 0.0)
<n>：	系统变量索引或元素索引	
	数据类型：	INT
	取值范围：	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
<k>：	坐标索引	
	数据类型：	INT
	取值范围：	0:X 坐标（横坐标） 1:Y 坐标（纵坐标） 2：Z 坐标（第三坐标轴）
<值>：	坐标值	
	数据类型：	REAL
	取值范围：	（- 最大 REAL 值）≤ x ≤（+ 最大 REAL 值）

示例

从输入坐标系出发，按照偏移矢量 (x, y, z) 和下列基于世界坐标系的坐标进行移动，得到第 9 个元素的输出坐标系：

- x = 10.0
- y = 20.0
- z = 30.0

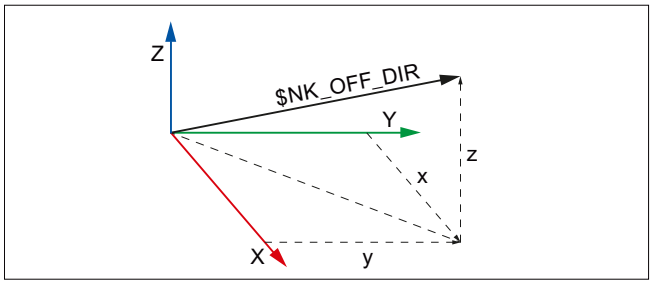


图 5-9　偏移矢量

程序代码	注释
; 第 9 个运动元素	

程序代码	注释
N100 $NK_OFF_DIR[8,0] = 10.0	; 0 = X 分量
N110 $NK_OFF_DIR[8,1] = 20.0	; 1 = Y 分量
N120 $NK_OFF_DIR[8,2] = 30.0	; 2 = Z 分量

5.2.3.10 $NK_TYPE = "SWITCH" 时的类型相关变量

$NK_SWITCH_INDEX

功能

开关由系统变量 $NK_SWITCH_INDEX 和 $NK_SWITCH_POS（见下文）构成。

在 $NK_SWITCH_INDEX 中输入索引 **i**，开关借助索引，通过系统变量 $NK_SWITCH[<**i**>] 接通和关闭。

句法

```
$NK_SWITCH_INDEX[<n>] = <i>
```

含义

$NK_SWITCH_INDEX:	索引 i，通过此索引和系统变量 $NK_SWITCH[<**i**>] 进行开关寻址	
	数据类型：	INT
	取值范围：	-1, 0, 1, 2, ... ($MN_MAXNUM_KIN_SWITCHES - 1)
		-1：开关状态恒定为“开”
	缺省值：	-1
<n>:	系统变量索引或元素索引	
	数据类型：	INT
	取值范围：	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
<i>:	索引值	
	数据类型：	INT

示例

运动链第 9 个元素是一个通过系统变量 $NK_SWITCH[**2**] 进行状态切换的开关

程序代码	注释
; 第 9 个元素	
N100 $NK_SWITCH_INDEX[8] = 2	; 索引 2

$NK_SWITCH_POS

功能

开关由系统变量 $NK_SWITCH_INDEX（见下文）和 $NK_SWITCH_POS 构成。

在 $NK_SWITCH_POS 中输入值 p（接通值），达到这个值时，开关通过系统变量 $NK_SWITCH[<i>] 切换到“开”位置。

根据 $NK_SWITCH[<i>]，开关具有下列状态：

- **开**：在 $NK_SWITCH_POS 中规定的开关接通值 p，与 $NK_SWITCH[<i>] 中的当前值相同。
  `$NK_SWITCH_POS[<n>] == $NK_SWITCH[<i>]`
  运动链前一个元素与输出（即在 $NK_NEXT 中规定的开关的下一个元素）连接在一起。
- **关**：在 $NK_SWITCH_POS 中规定的开关接通值，与 $NK_SWITCH[<i>] 中的当前值不同。
  `$NK_SWITCH_POS[<n>] ≠ $NK_SWITCH[<i>]`
  运动链前一个元素**未**与输出（即在 $NK_NEXT 中规定的开关的下一个元素）连接在一起。

可自由选择接通值

说明

平行元素 $NK_PARALLEL

在 $NK_PARALLEL 中规定的平行元素，其连接不受开关影响。也就是说，前一个元素始终与平行于开关的分支元素相连。

句法

```
$NK_SWITCH_POS[<n>] = <p>
```

含义

$NK_SWITCH_POS:	接通值	
	数据类型：	INT
	取值范围：	0 ≤ x ≤ 可能的最大 INT 值
	缺省值：	0
<n>：	系统变量索引或元素索引	
	数据类型：	INT
	取值范围：	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
<p>：	用于“开”这个开关位置的值	
	数据类型：	INT

示例

运动链第 9 个元素是一个开关，当 $NK_SWITCH[3] == 1 时，开关处于接通状态

程序代码	注释
; 第 9 个元素	
N110 $NK_SWITCH_POS[8] = 1	; 开关位置 = 1

另见

$NK_SWITCH (页 482)

$MN_MAXNUM_KIN_SWITCHES (页 463)

$MN_MM_MAXNUM_KIN_CHAIN_ELEM (页 462)

5.2.3.11 $NK_SWITCH

功能

开关变量由一个开关位置 i 的字段构成，在此字段内输入当前开关位置 p。

功能

为了对运动链中的开关 $_i$ 进行编程 (页 480)，必须将开关 $_i$ 与开关变量的索引 i 关联在一起，并为开关分配代表“开”状态的开关位置 p。

```
$NK_SWITCH_INDEX[<n>] = <i>
$NK_SWITCH_POS[<n>]   = <p>
```

之后可以通过开关变量的索引 i 接通和关闭开关 $_i$：

接通：`$NK_SWITCH[<i>] = <p>`

关闭：`$NK_SWITCH[<i>] ≠ <p>`

开关变量的索引可以与任意数量的开关关联在一起。

句法

```
$NK_SWITCH[<i>] = <p>
```

含义

$NK_SWITCH<i>:	具有开关位置 i 的开关变量	
	数据类型：	INT
	取值范围：	-1 ≤x ≤ 可能的最大 INT 值 提示 -1：初始状态“关”
<i>:	开关索引	
	数据类型：	INT
	取值范围：	0, 1, 2, ...(MD18882 $MN_MM_MAX_NUM_KIN_SWITCHES - 1)
<p>:	开关位置	
	数据类型：	INT
	取值范围：	-1 ≤x ≤ 可能的最大 INT 值
		-1：初始状态“关”

示例

为第 9 个运动元素分配名称“B 轴”：

程序代码	注释
N100 $NK_SWITCH[3] = 1	; 当前开关位置， ; 针对开关 [3] = 1

5.3 编程

5.3.1 删除组件 (DELOBJ)

函数 DELOBJ() 通过将分配的系统变量恢复为缺省值来“删除”组件：

- 运动链元素
- 保护区、保护区元素和防撞对
- 转换数据

句法

```
[<RetVal>=] DELOBJ(<CompType>[,,,<NoAlarm>)])
[<RetVal>=] DELOBJ(<CompType>,<Index1>[,,<NoAlarm>])
[<RetVal>=] DELOBJ(<CompType>[,<Index1>][,<Index2>][,<NoAlarm>])
```

含义

DELOBJ：	删除运动链元素、保护区元素、碰撞对元素和转换数据元素

<CompType>：	需删除组件的类型
	数据类型： STRING
	值：“KIN_CHAIN_ELEM” **含义**：所有运动元素的系统变量：$NK_...
	值：“KIN_CHAIN_SWITCH” **含义**：系统变量 $NK_SWITCH[<i>]
	值：“KIN_CHAIN_ALL” **含义**：所有运动元素和开关。 等同于通过 “KIN_CHAIN_ELEM” 和 “KIN_CHAIN_SWITCH” 逐步调用 DELOBJ
	值：“PROT_AREA” **含义**：保护区的系统变量： • $NP_PROT_NAME • $NP_CHAIN_NAME • $NP_CHAIN_ELEM • $NP_1ST_PROT
	值：“PROT_AREA_ELEM” **含义**：机床保护区和/或自动刀具保护区的保护区元素的系统变量： • $NP_NAME • $NP_NEXT • $NP_NEXTP • $NP_COLOR • $NP_D_LEVEL • $NP_USAGE • $NP_TYPE • $NP_FILENAME • $NP_PARA • $NP_OFF • $NP_DIR • $NP_ANG
	值：“PROT_AREA_COLL_PAIRS” **含义**：防撞对的系统变量： • $NP_COLL_PAIR • $NP_SAFETY_DIST
	值：“PROT_AREA_ALL” **含义**：所有保护区、保护区元素和防撞对（系统变量 $NP_...）

	等同于通过 “PROT_AREA”、“PROT_AREA_ELEM” 和 “PROT_AREA_COLL_PAIRS” 逐步调用 DELOBJ	
	值：“TRAFO_DATA” **含义**：所有转换的系统变量 $NT_...	
<Index1>：	第一个需删除组件的下标（**可选**）	
	数据类型：	INT
	缺省值：	-1
	取值范围：	-1 ≤ x ≤（所配置组件的最大数 -1）
	值	**含义**
	0, 1, 2, ...	需删除组件的下标。
	-1	所有指定类型的组件均已删除。<Index2> 不会被系统计算。
<Index2>：	最后需删除组件的下标（**可选**） 如果未编程 <Index2>，则只会删除 <Index1> 中指定的组件的系统变量。	
	数据类型：	INT
	缺省值：	仅 <Index1> 中指定组件的系统变量被删除。
	取值范围：	<Index1> < x ≤（所配置组件的最大数 -1）
<NoAlarm>：	报警抑制（**可选**）	
	数据类型：	BOOL
	缺省值：	FALSE
	值	**含义**
	FALSE	在故障时 (<RetVal> < 0) 停止程序执行并显示报警 .。
	TRUE	在故障时不停止程序执行且不显示报警。 使用场合：用户可根据返回值执行特定操作

<RetVal>:	函数的返回值	
	数据类型：	INT
	取值范围：	0, -1, -2, ... -7
	值	**含义**
	0	无故障
	-1	调用的函数不含参数。至少要指定参数 <CompType>。
	-2	<CompType> 表示未知组件
	-3	<下标 1> 小于 -1
	-4	<下标 1> 大于所配置的组件数
	-5	<下标 1> 在删除组件**组**时值不等于 -1
	-6	<下标 2> 小于 <下标 1>
	-7	<下标 2> 大于所配置的组件数
	-8	没有足够的写权限

5.3.2 通过名称确定下标 (NAMETOINT)

在 STRING 型的系统变量数组可以输入用户自定义的名称。借助系统变量名称，函数 NAMETOINT() 可以确定保存在系统变量数组中的名称所对应的下标值。

句法

```
<RetVal> = NAMETOINT(<SysVar>,<Name>[,<NoAlarm>])
```

含义

NAMETOINT:	确定系统变量下标	
<SysVar>:	STRING 型系统变量数组的名称	
	数据类型：	STRING
	值域：	NC 中所有 STRING 型系统变量数组的名称
<Name>:	需计算系统变量下标的字符串或名称。	
	数据类型：	STRING

<NoAlarm>:	报警抑制（**可选**）	
	数据类型：	BOOL
	缺省值：	FALSE
	值	**含义**
	TRUE	在故障时不停止程序执行且不显示报警。 使用场合：用户可根据返回值执行特定操作
	FALSE	在故障时 (<RetVal> < 0) 停止程序执行并显示报警。
<RetVal>:	系统变量下标或故障信息	
	数据类型：	INT
	值域：	-1 ≤ x ≤（所配置组件的最大数 -1）
	值	**含义**
	≥ 0	在指定的系统变量下标下找到了查找的名称。
	-1	名称未找到或出错。

示例

程序代码	注释
DEF INT INDEX	
$NP_PROT_NAME[27] = “隐藏”	
...	
INDEX = NAMETOINT（“$NP_PROT_NAME”, “隐藏”）	; INDEX == 27

5.4 示例

5.4.1 设定

简介

以配备三个交替使用的刀盘的五轴机床为例，介绍通过一个零件程序，对具有三个开关的运动链进行编程的基本步骤。在零件程序中写入所有与运动链有关的系统变量：

- 运动链 $NK_...

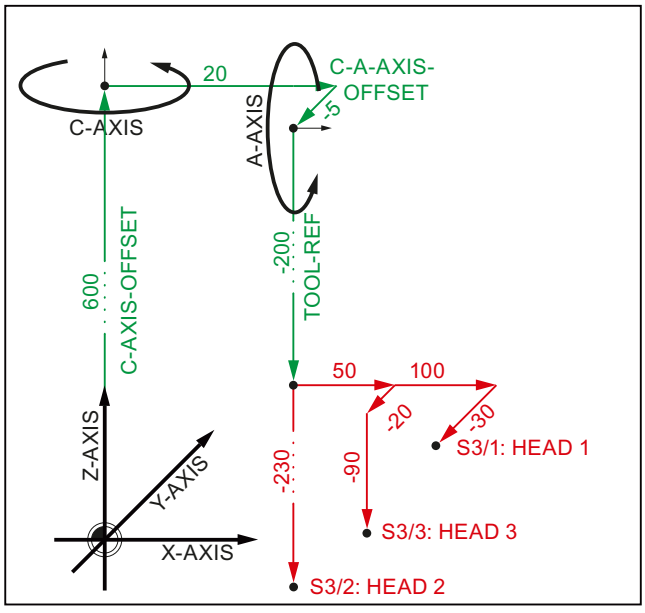
选项和机床数据

在本例中，需要设置下列选项和机床数据：

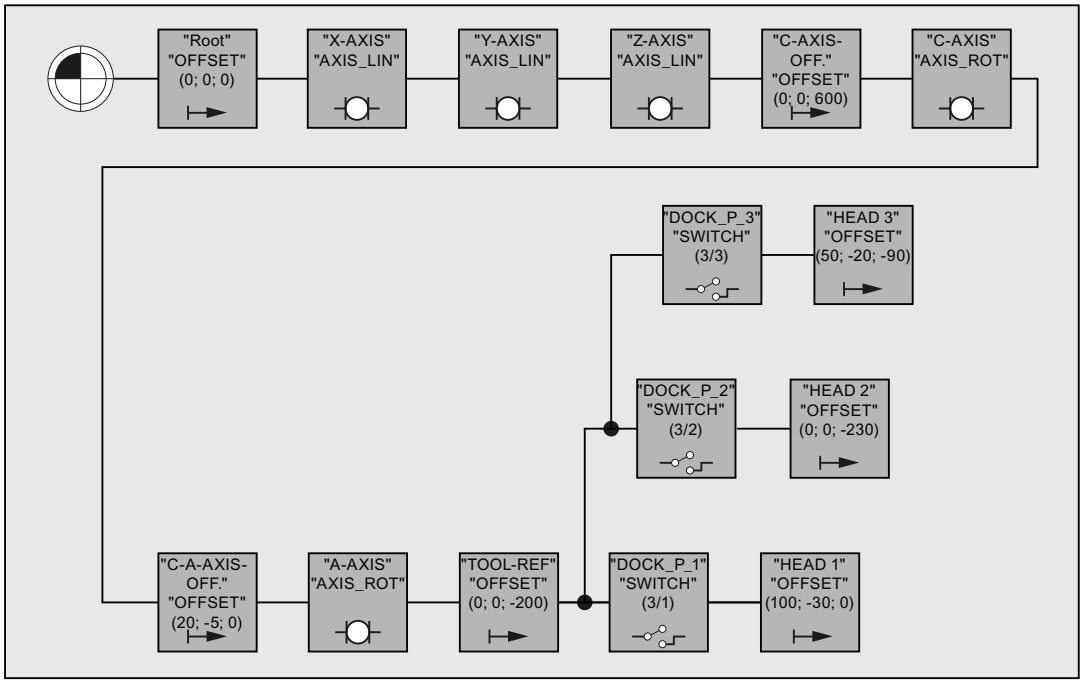
- MD19830 $ON_COLLISION_MASK.Bit 0 = 1
- MD18880 $MN_MM_MAXNUM_KIN_CHAIN_ELEM = 15

运动链

机床运动示意图



运动链元素



运动链以"Offset"（偏移）类型的一个元素作为开头。对防撞保护进行完整编程时，将机床的所有静态保护区域都分配给这个元素。

偏移元素后面是直线机床轴 X、Y、Z 的运动元素，以及回转轴 C 和 A 的偏移元素和运动元素。

在刀盘参考点的偏移元素后面，是用于开关刀盘的三个开关。所有三个开关都通过 $NK_INDEX = **3** 指向同一个开关变量 $NK_SWITCH[**3**]。各开关位置 (1, 2, 3) 只对处于活动状态的刀盘的子链有效。

5.4.2 机床模型的零件程序

程序代码

```
;============================================================
; 定义
;============================================================
N10 DEF INT KIE_CNTR ; 运动链元素计数器
N20 DEF INT RETVAL
;
;============================================================
; 碰撞数据初始化
;============================================================
; 将所有参数恢复初始值：
N30 RETVAL = DELOBJ("KIN_CHAIN_ELEM")
N40 KIE_CNTR = 0
;
;============================================================
; 运动链
;============================================================
; KE1:OFFSET: Root
; ------------------------------------------------------------
N50 $NK_TYPE[KIE_CNTR] = "OFFSET"
N60 $NK_NAME[KIE_CNTR] = "ROOT"
N70 $NK_NEXT[KIE_CNTR] = "X-AXIS"
N80 KIE_CNTR = KIE_CNTR + 1
;
; ------------------------------------------------------------
; 运动元素：直线轴：X 轴
; ------------------------------------------------------------
N90 $NK_TYPE[KIE_CNTR] = "AXIS_LIN"
N100 $NK_NAME[KIE_CNTR] = "X-AXIS"
N110 $NK_NEXT[KIE_CNTR] = "Y-AXIS"
N120 $NK_AXIS[KIE_CNTR] = "X1"
;
N130 $NK_OFF_DIR[KIE_CNTR,0] = 1.0    ; X
N140 KIE_CNTR = KIE_CNTR + 1
;
; ------------------------------------------------------------
; 运动元素：直线轴：Y 轴
; ------------------------------------------------------------
N150 $NK_TYPE[KIE_CNTR] = "AXIS_LIN"
N160 $NK_NAME[KIE_CNTR] = "Y-AXIS"
N170 $NK_NEXT[KIE_CNTR] = "Z-AXIS"
N180 $NK_AXIS[KIE_CNTR] = "Y1"
;
N190 $NK_OFF_DIR[KIE_CNTR,1] = 1.0    ; Y
N200 KIE_CNTR = KIE_CNTR + 1
;
; ------------------------------------------------------------
; 运动元素：直线轴：Z 轴
; ------------------------------------------------------------
```

程序代码

```
N210 $NK_TYPE[KIE_CNTR] = "AXIS_LIN"
N220 $NK_NAME[KIE_CNTR] = "Z-AXIS"
N230 $NK_NEXT[KIE_CNTR] = "C-AXIS-OFFSET"
N240 $NK_AXIS[KIE_CNTR] = "Z1"
;
N250 $NK_OFF_DIR[KIE_CNTR,2] = 1.0     ; Z
N260 KIE_CNTR = KIE_CNTR + 1
;
; ------------------------------------------------------------
; 运动元素：OFFSET：C 轴
; ------------------------------------------------------------
N270 $NK_TYPE[KIE_CNTR] = "OFFSET"
N280 $NK_NAME[KIE_CNTR] = "C-AXIS-OFFSET"
N290 $NK_NEXT[KIE_CNTR] = "C-AXIS"
;
N300 $NK_OFF_DIR[KIE_CNTR,2] = 600.0     ; Z 方向
N310 KIE_CNTR = KIE_CNTR + 1 ;
;
; ------------------------------------------------------------
; 运动元素：回转轴：C 轴
; ------------------------------------------------------------
N320 $NK_TYPE[KIE_CNTR] = "AXIS_ROT"
N330 $NK_NAME[KIE_CNTR] = "C-AXIS"
N340 $NK_NEXT[KIE_CNTR] = "C-A-OFFSET"
N350 $NK_AXIS[KIE_CNTR] = "C1"
;
N360 $NK_OFF_DIR[KIE_CNTR,2] = 1.0     ; Z 方向
N370 KIE_CNTR = KIE_CNTR + 1
;
; ------------------------------------------------------------
; 运动元素：OFFSET：C-A 轴
; ------------------------------------------------------------
N380 $NK_TYPE[KIE_CNTR] = "OFFSET"
N390 $NK_NAME[KIE_CNTR] = "C-A-OFFSET"
N400 $NK_NEXT[KIE_CNTR] = "A-AXIS"
;
N410 $NK_OFF_DIR[KIE_CNTR,0] = 20.0     ; X 方向
N420 $NK_OFF_DIR[KIE_CNTR,1] = -5.0     ; Y 方向
N430 KIE_CNTR = KIE_CNTR + 1
;
; ------------------------------------------------------------
; 运动元素：回转轴：A 轴
; ------------------------------------------------------------
N440 $NK_TYPE[KIE_CNTR] = "AXIS_ROT"
N450 $NK_NAME[KIE_CNTR] = "A-AXIS"
N460 $NK_NEXT[KIE_CNTR] = "TOOL-REF"
N470 $NK_AXIS[KIE_CNTR] = "A1"
;
N480 $NK_OFF_DIR[KIE_CNTR,0] = 1.0     ; X 方向
N490 KIE_CNTR = KIE_CNTR + 1
;
```

程序代码

```
; ------------------------------------------------------------
; 运动元素：OFFSET：刀具基准
; ------------------------------------------------------------
N500 $NK_TYPE[KIE_CNTR] = "OFFSET"
N510 $NK_NAME[KIE_CNTR] = "TOOL-REF"
N520 $NK_NEXT[KIE_CNTR] = "DOCKING_POINT 1"
;
N530 $NK_OFF_DIR[KIE_CNTR,2] = -200.0     ; Z 方向
N540 KIE_CNTR = KIE_CNTR + 1
;
; ------------------------------------------------------------
; 运动元素：开关 3/1
; ------------------------------------------------------------
N550 $NK_TYPE[KIE_CNTR] = "SWITCH"
N560 $NK_NAME[KIE_CNTR] = "DOCKING_POINT 1"
N570 $NK_NEXT[KIE_CNTR] = "HEAD 1"
N580 $NK_PARALLE[KIE_CNTR] = "DOCKING_POINT 2"
;
N590 $NK_SWITCH_INDEX[KIE_CNTR] = 3     ; 索引 3
N600 $NK_SWITCH_POS[KIE_CNTR] = 1       ; 开关位置 1
N610 KIE_CNTR = KIE_CNTR + 1
;
; ------------------------------------------------------------
; 运动元素：开关 3/2
; ------------------------------------------------------------
N620 $NK_TYPE[KIE_CNTR] = "SWITCH"
N630 $NK_NAME[KIE_CNTR] = "DOCKING_POINT 2"
N640 $NK_NEXT[KIE_CNTR] = "HEAD 2"
N650 $NK_PARALLE[KIE_CNTR] = "DOCKING_POINT 3"
;
N660 $NK_SWITCH_INDEX[KIE_CNTR] = 3     ; 索引 3
N670 $NK_SWITCH_POS[KIE_CNTR] = 2       ; 开关位置 2
N680 KIE_CNTR = KIE_CNTR + 1
;
; ------------------------------------------------------------
; 运动元素：开关 3/3
; ------------------------------------------------------------
N690 $NK_TYPE[KIE_CNTR] = "SWITCH"
N700 $NK_NAME[KIE_CNTR] = "DOCKING_POINT 3"
N710 $NK_NEXT[KIE_CNTR] = "HEAD 3"
N720 $NK_PARALLE[KIE_CNTR] = ""
N730 $NK_SWITCH_INDEX[KIE_CNTR] = 3     ; 索引 3
N740 $NK_SWITCH_POS[KIE_CNTR] = 3       ; 开关位置 3
N750 KIE_CNTR = KIE_CNTR + 1
;
; ------------------------------------------------------------
; 运动元素：OFFSET：HEAD 1
; ------------------------------------------------------------
N760 $NK_TYPE[KIE_CNTR] = "OFFSET"
N770 $NK_NAME[KIE_CNTR] = "HEAD 1"
N780 $NK_NEXT[KIE_CNTR] = ""
```

程序代码

```
;
N790 $NK_OFF_DIR[KIE_CNTR,0] = 100.     ; X 方向
N800 $NK_OFF_DIR[KIE_CNTR,1] = -30.     ; Y 方向
N810 KIE_CNTR = KIE_CNTR + 1
;
; ------------------------------------------------------------
; 运动元素：OFFSET：HEAD 2
; ------------------------------------------------------------
N820 $NK_TYPE[KIE_CNTR] = "OFFSET"
N830 $NK_NAME[KIE_CNTR] = "HEAD 2"
N840 $NK_NEXT[KIE_CNTR] = ""
;
N850 $NK_OFF_DIR[KIE_CNTR,2] = -230.     ; Z 方向
N860 KIE_CNTR = KIE_CNTR + 1
;
; ------------------------------------------------------------
; 运动元素：OFFSET：HEAD 3
; ------------------------------------------------------------
N870 $NK_TYPE[KIE_CNTR] = "OFFSET"
N880 $NK_NAME[KIE_CNTR] = "HEAD 3"
N890 $NK_NEXT[KIE_CNTR] = ""
;
N900 $NK_OFF_DIR[KIE_CNTR,0] =  50.     ; X 方向
N910 $NK_OFF_DIR[KIE_CNTR,1] = -20.     ; Y 方向
N920 $NK_OFF_DIR[KIE_CNTR,2] = -90.     ; Z 方向
N930 KIE_CNTR = KIE_CNTR + 1
;========================= 结束 ============================
```

连续路径运行、准停、预读 6

6.1 简要说明

准停或准停运行

准停是一种运行模式，在该模式下每个运行程序段结束时，所有参与运动的轴（除了跨程序段运行的轴）将制动至静止状态。只有当所有参与运动的轴都根据所选的准停标准到达编程的目标位置后，NC 才会切换至下一个程序段。

连续路径运行

连续路径运行是一种运行模式，在该模式下，NC 会尽可能保持编写的轨迹速度恒定不变。尤其是在需要避免轨迹轴在零件程序程序段交界处制动的情况下。

预读

预读是一个用于优化连续路径运行的功能。

高质量的工件表面要求进行均匀加工，因此在加工期间应尽量避免轨迹速度的波动。没有预读功能时，NC 只读取紧跟着当前程序段的后一条程序段，以确定可能的轨迹速度。如果该后续程序段只包含一段很短的行程，则 NC 必须降低轨迹速度，即在当前程序段内就进行制动，以便及时地在后续程序段结束时停止。

使用预读功能，NC 可以预先读取当前程序段后面一定数量的后续程序段，在一些条件下可以显著提高轨迹速度，因为现在 NC 可以提供更多的程序段或更长的行程供计算。

使用该功能的优点有：

- 平均轨迹速度更高
- 减少制动和加速过程，工件表面质量更佳

轨迹速度平滑

“轨迹速度平滑”是一项针对对均匀轨迹速度有高要求的应用而开发的功能，比如：模具制造上的高速铣削。为此，在平滑轨迹速度时会舍弃一些易引起机床高频共振的制动和减速过程。

使用该功能的优点有：

- 避免了机床共振，工件表面质量更佳，加工时间更短
- 避免了多余的加速过程，即对程序处理时间没有很大益处的加速过程，轨迹速度或切削速度更均匀稳定。

轨迹动态响应自适应

除了“轨迹速度平滑”外，“轨迹动态响应自适应”是另一项用于避免机床高频共振并可同时优化轨迹动态响应的功能。为此，相比于机床数据设置的原始动态响应极限值，轨迹速度的高频变化会自动采用更低的急动值或加速度值。

因此在轨迹速度低频变化时原始动态响应极限值生效，而在轨迹速度高频变化时 NC 会自动调整动态响应，低动态响应极限值生效。

轨迹插补的动态响应模式

在优化轨迹动态响应方面，工艺专用的动态响应设置也可产生很大影响，它针对不同的工况预设（如攻丝、粗加工、精加工等），可以通过在零件程序中调用对应的动态响应模式来激活。

任意形状表面模式

曲率或挠度的每次波动都会引起轨迹速度的变化，因此，通常在加工表面为任意形状的工件时，会产生一些多余的制动和加速过程，这些过程常常会降低工件的表面质量。

所以，针对该表面加工系统提供以下功能：

- 功能“任意形状表面模式：基本功能”
  该功能可以使轨迹速度的变化相对于曲率和挠度的变化变得“迟钝”。
- 功能“任意形状表面模式：扩展功能”
  该功能是预读标准功能的扩展，可以在加工任意形状表面时计算轨迹速度曲线。

任意形状表面模式的优点在于，首先工件表面更均匀，其次机床承受的负载更轻。

线性程序段的压缩

在用 CAD/CAM 系统设计完工件后，还需要使用该系统创建用于生成工件表面的零件程序。大多数 CAD/CAM 系统也采用线性程序段来描述弯曲的工件表面。为达到所需的轮廓精度，通常该过程需要大量的插补点，从而产生了大量短行程线性程序段。

通过使用“压缩器功能”，可以事后借助多项式程序段来近似由线性程序段设定的轮廓。因此，多条线性程序段被一条多项式程序段取代。

优点：

- 减少运行程序段的数量
- 提升轨迹速度
- 提升表面质量
- 程序段过渡稳定

压缩较短的样条程序段

样条定义一个由 2 阶或 3 阶多项式合并的曲线。通过样条插补，控制系统只需少数几个目标轮廓插补点便可生成一条光滑的曲线。

相比于线性插补而言，样条插补的优点有：

- 减少了描述曲面轮廓所需零件程序段的数目
- 曲线更加“软”，保护机械装置，程序段过渡处也是如此。

相比于线性插补而言，样条插补的缺点有：

- 对于样条曲线无法设置精确曲线形状，而只能设置样条曲线所在的公差带。

和线性插补一样，在处理样条时也会产生短程序段，插补这些短样条程序段时必须降低轨迹速度。就算样条是一条平滑的长曲线时也是如此。通过功能“压缩较短的样条程序段”可以合并样条程序段，从而产生足够长的程序段并避免降低轨迹速度。

6.2 准停运行

准停或准停运行

准停或准停运行是一种运行模式，在该模式下每个运行程序段结束时，所有参与运动的轨迹轴和辅助轴（除了跨程序段运行的轴）将制动至静止状态。只有当所有参与运动的轴都根据所选的准停条件到达编程的目标位置后，NC 才会切换至下一个程序段。

从而产生以下特性：

- 由于所有参与运动的机床轴都需要先制动，然后等待进入“准停”状态，所以相对于连续路径运行，程序运行时间会大大延长。
- 如果在加工期间进行准停运行，会在工件表面上留下加工灼痕。

准停条件

可设置下列准停条件：

- “粗准停”
- “精准停”
- “插补终点”

应用

如果需要精确地加工轮廓，则需要使用准停运行。

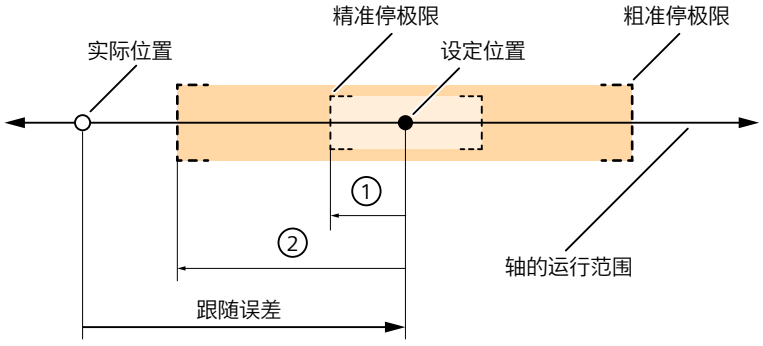
激活

在程序中，可通过下列指令模态或逐段设定准停运行：

指令	含义
G60	准停运行自当前程序段起**模态**生效。
G9	准停运行在当前程序段中生效。

准停条件“粗准停”或“精准停”

当机床轴的当前跟随误差小于等于以下机床数据确定的设定位置公差时，便满足准停条件“粗准停”或“精准停”。



① MD36010 $MA_STOP_LIMIT_FINE（精准停）

② MD36000 $MA_STOP_LIMIT_COARSE（粗准停）

图 6-1 准停条件的公差带

说明

在设置准停条件“粗准停”和“精准停”的公差带时，需满足以下条件：
粗准停 > 精准停

准停条件：“插补结束”

选择准停条件“插补结束”时，一旦所有参与运动的轨迹轴和辅助轴（除了跨程序段运行的轴外）都达到了在运行程序段中编写的位置，便立即切换到下一程序段。即插补器已经处理完该程序段。

该准停条件“插补结束”不考虑机床轴的实际位置或跟随误差。因此，与准停条件“粗准停”/“精准停”相比，程序段过渡处的轮廓更加平滑，平滑幅度取决于机床轴的动态响应。

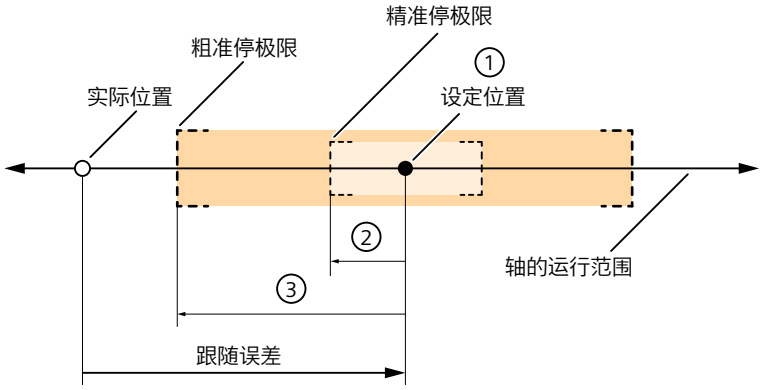
激活可编程的准停条件

通过下列指令激活可编程的准停条件：

指令	准停条件
G601	精准停
G602	粗准停
G603	插补结束

根据生效的准停条件进行程序段切换

下图示出取决于所选择的准停条件的程序段切换时间点。



① G603

② G601

③ G602

图 6-2 根据生效的准停条件进行程序段切换

准停条件的权重系数

可通过以下轴专用机床数据设定准停条件的取决于参数组的权重：

MD36012 $MA_STOP_LIMIT_FACTOR[<参数组>] = <值>

权重系数会影响下列机床数据的值：

- MD36000 $MA_STOP_LIMIT_COARSE
- MD36010 $MA_STOP_LIMIT_FINE
- MD36030 $MA_STANDSTILL_POS_TOL

应用示例

- 根据新的质量比来调整定位方式，比如：在切换齿轮档后。
- 根据不同的运行状态（粗加工或精加工）来缩短定位时间。

设定生效的准停条件

G 指令组 1 的准停条件可以固定设置：这使得编写的准停条件无效。

可**相互独立**地为下列指令设置准停条件：

- 快速移动 G0
- G 指令组 1 中所有其他指令

通过以下十进制编码机床数据针对特定通道设置准停条件：

MD20550 $MC_EXACT_POS_MODE = <Z><E>

Z	E	生效的准停条件
0	0	编写的准停条件
1	1	G601（精准停窗口）
2	2	G602（粗准停窗口）
3	3	G603（插补结束）
E（个位）：针对快速移动的准停条件设置。 Z（十位）：设置 G 指令组 1 中所有其他指令的准停条件。		

示例

MD20550 $MC_EXACT_POS_MODE = 02

- <E> = 2：执行快速移动时，不论如何编程，始终是准停条件"G602"（粗准停窗口）生效。
- <Z> = 0：在执行 G 指令组 1 中所有其他指令时，程序中写入的准停条件生效。

连续路径运行中 G0 ↔ 非 G0 程序段切换时的程序段切换特性

在连续路径运行中，可通过以下机床数据设置在快速移动与非快速移动（G0 ↔ 非 G0）程序段之间切换时的特性：

MD20552 $MC_EXACT_POS_MODE_G0_TO_G1 = <值>

<值>	含义
0	在程序段过渡处没有额外的停止。
1	在程序段过渡处停止。 特性对应 G601（精准停窗口）
2	在程序段过渡处停止。 特性对应 G602（粗准停窗口）
3	在程序段过渡处停止。 特性对应 G603（插补结束）

<值>	含义
4	在程序段过渡处**不**停止。 在连续路径运行中，在 **G0 ↔ 非 G0 程序段切换**时，在 G0 程序段中预先将随后之非 G0 程序段的**进给补偿**的当前值考虑在内。根据轴动态特性以及当前程序段的轨迹长度，以后续程序段的经精确或及尽可能最佳调整的速度执行程序段切换。
5	在程序段过渡处**不**停止。 在连续路径运行中，在 **G0 → 非 G0** 和 **非 G0 → G0 程序段切换**时，预先将后续程序段的**进给补偿**（G0 → 非 G0）或**快速移动补偿**（非 G0 → G0）的当前值考虑在内。根据轴动态特性以及当前程序段的轨迹长度，以后续程序段的经精确或及尽可能最佳调整的速度执行程序段切换。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Axis>.basic.in.posCoarseReached	LBP_Axis*.E_ExactCoarse	DB31, ...DBX60.6
<Axis>.basic.in.posFineReached	LBP_Axis*.E_ExactFine	DB31, ...DBX60.7

6.3 连续路径运行

6.3.1 一般功能

连续路径运行

在连续路径运行中，结束一条程序段而切换到另一条程序段时，轨迹速度**不会**为了达到精准停标准而降低到很小。目标是，在程序段切换点处避免轴停止加工，尽可能以相同的轨迹速度转到下一个程序段。为了实现此目标，激活连续路径运行时还应激活“预读”功能。

连续路径运行通过局部修改编程曲线，使原本突兀曲折的程序段过渡更加平滑、圆顺。对编程曲线的修改幅度可以由过载系数标准或平滑加以限制。

通过连续路径运行可以实现：

- 轮廓圆顺。
- 省去了达到准停标准所需的制动和加速过程， 从而缩短了加工时间。
- 平缓的速度变化，获得良好的切削质量

在下列情形下建议使用连续路径运行：

- 需要尽可能快速地加工轮廓（比如通过快速移动）。
- 实际运行与编程的运行允许有所偏差（在设定的公差内），以使运行保持连续、稳定。

要求精确加工轮廓时，不建议使用连续路径运行。

隐含的准停功能

在某些情形下，在连续路径加工方式中需要执行准停功能，以便进行后续操作。这时轨迹速度被降到零。

- 辅助功能指令如果在轴运行前给出，则前一个程序段只有在到达所选择的准停标准后才结束。
- 辅助功能指令如果在轴运行后给出，则在程序段插补结束后输出辅助功能。
- 如果可执行程序段（比如：启动一根定位轴）中不包含任何轨迹轴的运行信息，当符合所选择的准停标准时，前一个程序段结束。
- 如果定位轴被申明为“几何轴”并在程序中写入了该几何轴，则当插补结束时，前一个程序段结束。
- 如果程序中写入了一根同步轴，该轴最后又曾作为定位轴或主轴写入程序（辅助轴默认定义为定位轴），则当插补结束时，前一个程序段结束。
- 如果切换了坐标转换，则达到当前生效的准停标准后，切换前的程序段结束。
- 如果在下一个程序段中包含了加速特性 BRISK/SOFT 的转换，则当前程序段以插补结束结束（参见功能手册中“*进给轴和主轴*”中的“*加速度*”）。
- 如果在零件程序中写入了“清空缓冲存储器”功能，则前一个程序段在到达所选择的准停标准后才结束。

在连续路径加工方式下速度=0

在下列情形下，在程序段结束时加工速度被制动为零，不管是否隐含准停功能：

- 通过指令 POS 写入了定位轴，该轴的运行时间长于轨迹轴的运行时间。在此情况下，只有在定位轴达到“准停”后才进行程序段的切换。
- 通过指令 SPOS 写入的主轴的定位时间长于轨迹轴的运行时间。在此情况下，只有在定位主轴“精准停”后才进行程序段的切换。
- 在当前程序段中运行的是几何轴，但在下一程序段中运行的是同步轴；或在当前程序段中运行的是同步轴，而在下一个程序段中运行的是几何轴。
- 需要进行同步。

6.3.2 按过载系数降低速度

功能

此功能可以在连续路径方式下降低轨迹速度，以便非相切程序段过渡可以在一个插补周期中完成，并保持加速度极限和考虑过载系数。

如果程序段过渡处的轮廓不是相切轮廓，减速会导致轴速度突然改变。 该现象也可以从一同运行的同步轴上观察得出。 速度跃变可避免将进给速度降到零。如果当轴速度降低到某一速度值并从该速度值进行加速，从而可以到达新的速度设定值时，则发生速度跃变。 可以使用过载系数来控制速度跃变值。速度跃变值是与轴相关的，因此在程序段切换时考虑的是当前有效轴的速度跃变值。

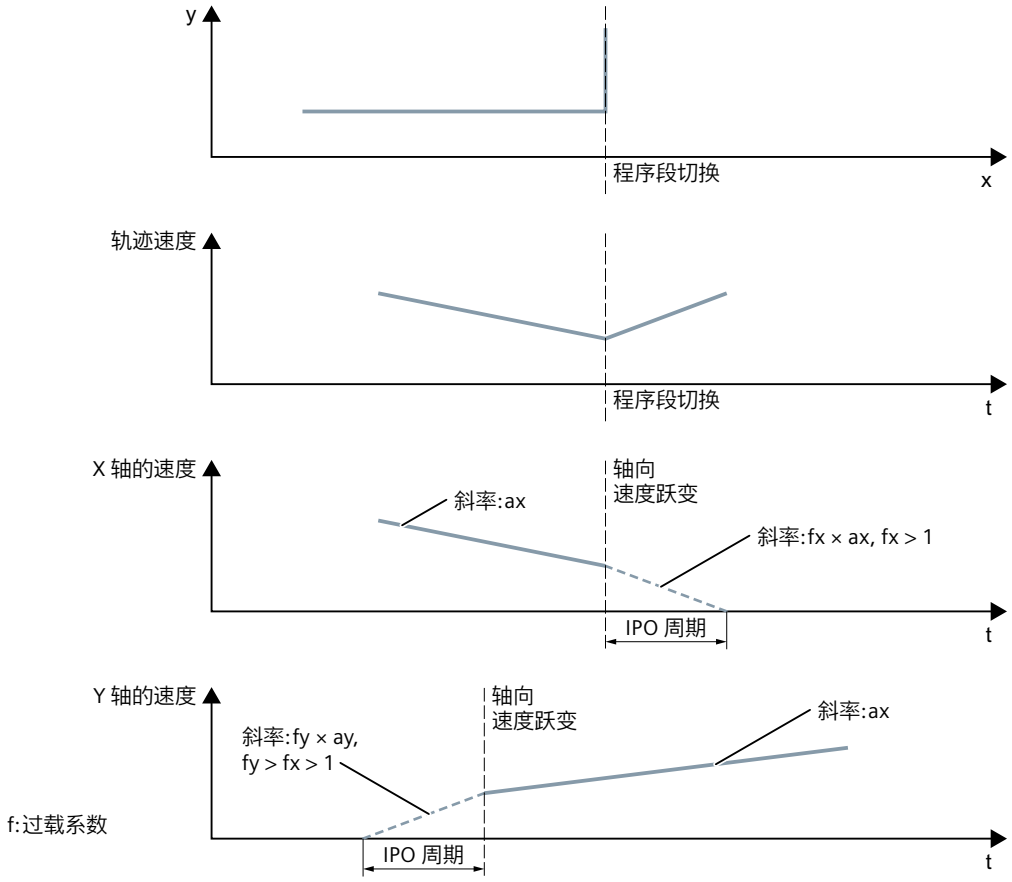


图 6-3　程序段切换处的轴速度变化

对于几乎是相切的程序段切换，如果未超过允许的轴加速度，则不降低轨迹速度。 因此可以对轮廓进行很小角度（比如：0.5°）的超程加工。

过载系数

过载系数限制机床轴在程序段切换时的速度跃变。为了确保速度跃变不超出轴负载，跃变值通过轴的加速度计算得到。

过载系数定义了机床轴的加速度在一个插补周期内可以超出极限的幅度，此加速度值在 MD32300 $MA_MAX_AX_ACCEL 中定义。

速度跃变值的计算公式为：

速度跃变值 = 轴加速度 * (过载系数 -1) * 插补周期。

在以下机床数据中可以对过载系数进行调整：

MD32310 $MA_MAX_ACCEL_OVL_FACTOR（轴速度跃变的过载系数）

系数 1.0 表示只能按限定系数进行相切程序段切换。 对于任何其它的切换，速度将根据设定值降为 0。 其特性相当于“准停标准：插补结束”。 由于该方式对于连续路径运行而言不理想，因此应设置大于 1.0 的系数。

说明

在调试时注意，当在突兀的程序段过渡处机床振动而不希望使用“平滑”功能时，应降低该系数。

设置以下机床数据，可以对程序段过渡进行平滑，不管当前设置的过载系数 G641 / G642：

MD20490 $MC_IGNORE_OVL_FACTOR_FOR_ADIS

激活/取消激活

在每个数控程序段中都可以通过模态指令 G64 激活“连续路径运行 + 按过载系数降低速度”。

可以选择非模态准停指令 G9 来中断连续路径运行。

或者选择以下某个指令来撤销连续路径运行 G64：

- 模态准停 G60
- 平滑 G641, G642, G643, G644 或 G645

隐含的连续路径运行

如果由于程序段行程过短（比如：零周期程序段）而导致无法在 G641 中插入平滑程序段，则切换到 G64。

6.3.3 平滑

功能

“平滑”功能会沿着编写的轮廓（轨迹轴）在非圆顺（弯折）程序段过渡处插入中间程序段（平滑程序段），由此便可产生圆顺（切向）的新程序段过渡。

同步轴

除几何轴外，平滑还会考虑所有的同步轴。但在同时运行轨迹轴和同步轴时，两种类型的轴可能无法同时生成一个圆顺的程序段过渡。由于轨迹轴的运行精度始终高于同步轴，因此，只能生成一个接近圆顺的程序段过渡。

G64 平滑

如果为了遵循程序段过渡时的动态限值，要求的速度低于 G64 允许的程序段过渡速度，则也可以使用平滑功能（参见 “按过载系数降低速度 (页 506)” 一章中的过载系数）。

对同步条件的影响

在进行平滑时，系统会缩短编写的程序段，这些程序段之间插入了一个或多个平滑轮廓。编写的程序段分界被取消，不再可用作同步条件的标准，比如：和运动、停止一同输出的辅助功能。

说明

在使用“平滑”功能时，同步条件针对的都是平滑位置前的程序段的末尾，而不是插入的平滑程序段的末尾。因此，下一条程序段没有开始，在停止在程序段末尾时，还可以手动修改下一条程序段的轮廓。

例外情况

在以下程序段过渡中，此处以从 N10 过渡带到 N20 为例，**无**平滑，即不会插入任何平滑程序段：

隐性停止运行

可能的原因：

- N20 运行前辅助功能输出生效
- N20 不包含轨迹轴运行动作
- 一个轴之前是定位轴，但在 N20 中首次作为轨迹轴运行
- 一个轴之前是轨迹轴，但在 N20 中首次作为定位轴运行
- N10 中运行几何轴，而 N20 中不运行
- N20 中运行几何轴，而 N10 中不运行
- 激活 N20 中的螺纹切削 G33
- 在 BRISK 和 SOFT 之间进行切换
- 对坐标转换非常重要的轴没有完全分配到轨迹运动 （比如摆动轴、定位轴）。

插入平滑程序段会使零件程序运行速度过渡减慢

可能的原因：

- 一个程序或程序段是由大量很短的运行程序段组成的（≈ 1 个插补周期 / 运行程序段；由于每个运行程序段至少需要一个插补周期，所以插入的中间程序段使运行时间加倍）
- 进行程序段切换时，不减速的 G64（不带平滑的连续路径运行）生效
- 编写的过载系数 (MD32310 $MA_MAX_ACCEL_OVL_FACTOR) 可实现在不减小轨迹速度的情况下运行出编写的轮廓。另见：MD20490 $MC_IGNORE_OVL_FACTOR_FOR_ADIS

轨迹参数忽略平滑

可能的原因：

- G641（符合行程标准的带平滑连续路径运行）生效，快速移动也生效 (G0) UND ADISPOS == 0（G0 时的平滑间距）
- G641（符合行程标准的带平滑连续路径运行）生效，但快速移动**未**生效 UND ADIS == 0（轨迹功能 G1、G2、G3 ... 的平滑间距）
- G642 或 G643（遵循指定公差的带平滑连续路径运行）生效，但所有公差 == 零

N10 或 N20 不包含运行动作（零程序段）

通常不会生成零程序段。例外情况：

- 同步动作激活时
- 程序跳转

对同步条件的影响

在进行平滑时，系统会缩短编写的程序段，这些程序段之间插入了平滑轮廓。最初编写的程序段分界被取消，不再可用于同步条件，比如：和运动、停止一同输出的辅助功能。

说明

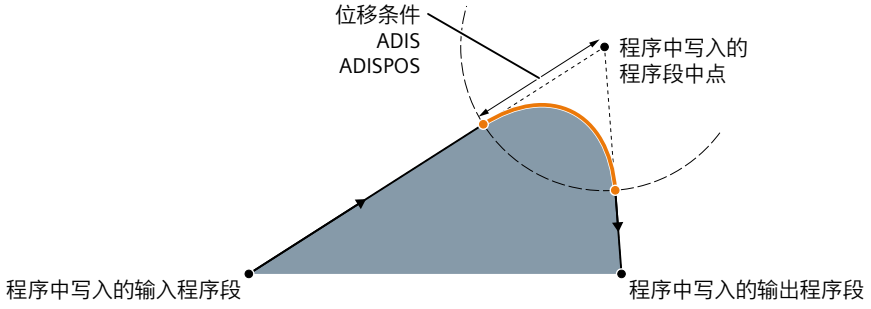
在使用“平滑”功能时，同步条件针对的都是平滑位置前的程序段的末尾，而不是插入的平滑程序段的末尾。因此，下一条程序段没有开始，在停止在程序段末尾时，还可以修改下一条程序段的轮廓。

6.3.3.1 按照位移条件开展平滑(G641)

功能

在按照位移条件开展平滑的连续路径运行中，平滑区域的大小是由位移条件 ADIS 和 ADISPOS 决定的。

位移条件 ADIS 或 ADISPOS 描述了程序段结束前平滑程序段最早可以开始的距离，或者程序段结束后必须结束的距离。



说明

尖角会产生特别弯曲的平滑曲线，从而导致减速。

说明

ADISPOS 和 ADIS 处理方式相同，但只能用于快速移动 G0。

位移条件的有效性

- 必须编程 ADIS 或 ADISPOS。 保持缺省值“零”时，G641 和 G64 的特性一样。
- 如果两个相连程序段不是快速移动 G0，则较短的平滑距离生效。

- 如果 ADIS 的值过小，要注意，控制系统会确保每个被插补的程序段（即使是一个平滑中间程序段）也至少具有一个插补点。 因此，最大的轨迹速度被限制在“ADIS/插补周期”以下。
- 不管是 ADIS 还是 ADISPOS，平滑区域都是通过程序段位移长度来限制的。
  在短行程程序段中（行程小于 4 倍的 ADIS 或 4 倍的 ADISPOS），平滑距离会被缩短，使原始程序段仍能保持一段可移动行程。 剩余长度由轴特性决定，大概是轴待移动行程的 60%。 ADIS 或 ADISPOS 因此被缩短为待移动行程的 40%! 该算法还可避免在非常细微的轮廓变化中插入平滑程序段。 此时系统会一直切换到连续路径运行 G64，直到可以再次插入平滑程序段。

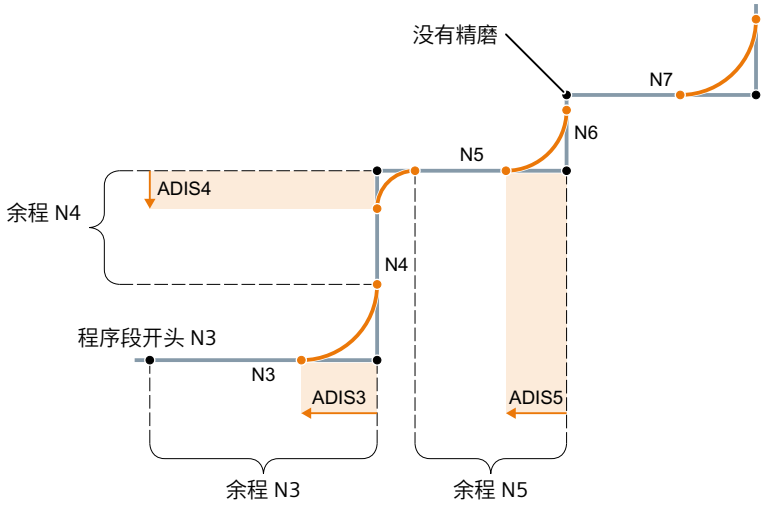


图 6-4　含 ADIS 限制的轨迹曲线

激活/取消激活

在每个数控程序段中都可以通过模态指令 G641 激活按照位移条件开展平滑的连续路径运行。在激活前或激活时应设定位移条件 ADIS/ADISPOS。

可以选择非模态准停指令 G9 来中断连续路径运行。

或者选择以下某个指令来撤销 G641：

- 模态准停 (G60)
- 连续路径运行 G64, G642, G643, G644 或 G645

程序示例

程序代码	注释
N1 G641 Y50 F10 ADIS=0.5	; 连续路径运行，根据位移条件进行平滑（平滑距离： 0.5 mm）。

程序代码	注释
N2 X50	
N3 X50.7	
N4 Y50.7	
N5 Y51.4	
N6 Y51.0	
N7 X52.1	

6.3.3.2 按照定义的公差开展平滑(G642/G643)

功能

在按照定义的公差开展平滑的连续轨迹运行中，通常平滑按照最大允许的轨迹公差进行。 该轴专用公差也可通过最大轮廓偏差（轮廓公差）或者刀具定向的最大角度偏差（定向公差）来取代。

激活

在每个数控程序段中都可以通过模态指令 G642 或 G643 激活按照定义的公差开展平滑的连续轨迹运行。

可以选择非模态准停指令 G9 来中断连续路径运行。

或者选择以下某个指令来撤销 G642/G643：

- 模态准停 (G60)
- 连续路径运行 G64, G641, G644 或 G645

G642 和 G643 之间的区别

G642 和 G643 在平滑特性上有以下区别：

G642	G643
G642 中，平滑距离由所有轴最短的平滑距离决定。在生成平滑程序段时会考虑该值。	G643 中，各轴的平滑距离可以有所不同。程序段内部会考虑各轴的平滑距离（(⇒ 不是各自的平滑距离）。
在 G642 中，平滑区域由最小的公差设定得出。	如果轮廓公差和刀具定向公差的设定数据区别很大，则只在 G643 中生效。

参数设置

最大轨迹公差

G642/G643 平滑中最大允许的轨迹公差在以下机床数据中为各个轴单独设定：

MD33100 $MA_COMPRESS_POS_TOL

轮廓公差和定向公差

轮廓公差和定向公差在通道专用设定数据中设置：

SD42465 $SC_SMOOTH_CONTUR_TOL（最大轮廓偏差）

SD42466 $SC_SMOOTH_ORI_TOL（刀具定向最大角度偏差）

设定数据可在程序中编程，因此为各程序段过渡分别设定。

说明

设定数据 SD42466 $SC_SMOOTH_ORI_TOL 只有在定向转换激活时生效。

平滑特性

G642/G643 的平滑特性由以下机床数据设置：

MD20480 $MC_SMOOTHING_MODE（G64x 平滑特性）

个位(**E**) 定义 G643 特性，十位(**Z**) 定义 G642 特性：

E 或 Z 的值	含义
0	**所有轴：** 按照允许的最大轨迹公差开展平滑： MD33100 $MA_COMPRESS_POS_TOL
1	**几何轴：** 按照轮廓公差开展平滑： SD42465 $SC_SMOOTH_CONTUR_TOL **其他轴：** 按照允许的最大轨迹公差开展平滑： MD33100 $MA_COMPRESS_POS_TOL

E 或 Z 的值	含义
2	**几何轴：** 按照定向公差开展平滑： SD42466 $SC_SMOOTH_ORI_TOL **其他轴：** 按照允许的最大轨迹公差开展平滑： MD33100 $MA_COMPRESS_POS_TOL
3	**几何轴：** 按照定向公差和轮廓公差开展平滑： SD42465 $SC_SMOOTH_CONTUR_TOL SD42466 $SC_SMOOTH_ORI_TOL **其他轴：** 按照允许的最大轨迹公差开展平滑： MD33100 $MA_COMPRESS_POS_TOL
4	**所有轴：** 使用 ADIS 或 ADISPOS 编程的平滑长度，和 G641 一样。 轴专用公差、轮廓公差或定向公差的设定会被忽略。

极限速度的特性

按照定义公差进行平滑时的速度特性是由 MD20480 的百位确定的：

值	含义	
< 100:	在平滑区域内，会计算极限速度的特性，它是由设定的各轴/轨迹最大加速度和最大急动度得出的。 这种计算有可能导致轨迹速度在平滑区域内提升，并由此导致参与轴加速。	
≥100:	在 G641/G642 的平滑程序段中，不计算极限速度的特性。 只确定恒定的极限速度。 它可以避免在 G641/G642 平滑中参与轴在平滑区域内加速。 但是在一些条件下，尤其是平滑区域很大时，该设置会导致平滑程序段的处理速度过慢。	
	1xx:	G641 无速度特性
	2xx:	G642 无速度特性

说明

参见 MD28530 $MC_MM_PATH_VELO_SEGMENTS（存储单元的数量，用于限制轨迹速度）

前提条件

在半径补偿和刀具定向激活时保护区的局限性：

对于不垂直于基本坐标系中某个基准面的刀具定向而言，它虽然考虑半径补偿，但是保护区不会转入对应的平面。

G643 中必须作如下设置：

MD28530 $MC_MM_PATH_VELO_SEGMENTS > 0（存储单元的数量，用于限制轨迹速度）

如果不满足该条件，则必须为所有轴设置：

MD35240 $MC_ACCEL_TYPE_DRIVE = FALSE（加速特性曲线 DRIVE 开/关）

6.3.3.3 按照允许的最大动态响应进行平滑(G644)

功能

在该带平滑的连续路径运行中，允许的最大动态响应是重要因素。

说明

只有在以下条件下，才允许 G644 平滑：

- 在这两个被读取的程序段中，所有参与轴只包含线性运动。
- 运动转换**没有**生效。

一根参与轴包含多项式（编程了多项式、样条或压缩器生效）或有一个运动转换生效（PTP 会暂时关闭运动转换）时，程序段过渡处使用 G642 进行平滑。

激活

在每个数控程序段中都可以通过模态指令 G644 激活以允许的最大轴动态响应进行平滑的连续路径运行。

可以选择非模态准停指令 G9 来中断连续路径运行。

或者选择以下某个指令来撤销 G644：

- 模态准停 (G60)
- 连续路径运行 G64, G641, G642, G643 或 G645

参数设置

G644 的平滑特性由以下机床数据的千位和万位设置：

MD20480 $MC_SMOOTHING_MODE（G64x 平滑特性）

值	含义
千位：	
0xxx:	G644 平滑按照以下数据设定的每轴最大公差进行： MD33100 $MA_COMPRESS_POS_TOL 如果轴的动态响应允许，有时不会完全利用设置的公差。
1xxx:	通过编程 ADIS=...或 ADISPOS=...设定最大的平滑距离，同 G641。
2xxx:	通过以下机床数据设置平滑区域内每轴的最大平滑频率： MD32440 $MA_LOOKAH_FREQUENCY （预读时的平滑频率） 确定合适的平滑范围，使平滑频率不会超出设置的最大频率。
3xxx:	在拐角处会出现速度跃变的每轴会以最大允许的动态响应（最大加速度和最大急动度）绕行拐角。 SOFT: 在使用 SOFT 时，会限制每轴的最大加速度**以及**最大急动度。 BRISK: 在使用 BRISK 时，只限制每轴的最大加速度，**不**限制最大急动度。 在该设置中，即不监控最大可能出现的公差也不监控最大平滑距离。公差或平滑距离仅仅由各轴的动态极限值和当前轨迹速度得出。
4xxx:	同 0xxx ，按照以下数据设定的每轴最大公差进行平滑： MD33100 $MA_COMPRESS_POS_TOL 和 0xxx 不同之处在于，在可能条件下会充分利用设置的公差。因此，轴不会达到最大可能的动态响应。
5xxx:	同 1xxx，通过编程 ADIS=...或 ADISPOS=...设定最大的平滑距离。 和 1xxx 不同之处在于，在可能条件下会充分利用设定的平滑距离。因此，参与轴不会达到最大可能的动态响应。
万位	
0xxxx	采用 BRISK 时，平滑区域内轴的速度特性不含急动度限制，采用 SOFT 时，含急动度限制，
1xxxx	不管采用 BRISK 或是 SOFT，平滑区域内轴的速度特性都会含急动度限制。

在设定了最大轴公差 (MD33100 $MA_COMPRESS_POS_TOL) 或最大平滑距离(ADIS/ADISPOS) 时，只要参与轴的动态响应允许，通常不会充分利用平滑距离。因此，平滑距离的长度由当前轨迹进给率决定。在轨迹速度较低时，和编程轮廓的偏差较小。但也可以作其

他设定，即此时尽可能充分利用设定的最大公差或最大平滑距离。这样和编程轮廓的公差便与编程的轨迹进给率毫无关联。

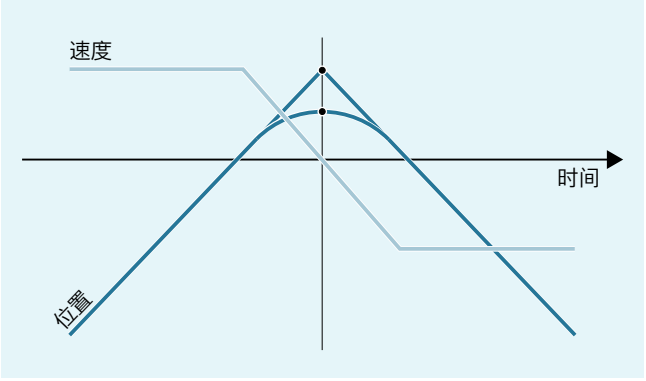
说明

另外还可以激活以下附加的限制：
平滑距离最大是原始程序段行程的一半。

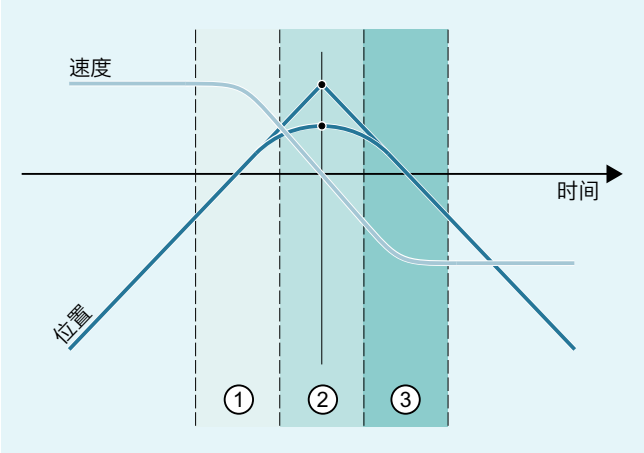
急动度限制

每轴速度跃变的平滑以及平滑距离的类型取决于插补是带急动度限制还是不带该限制。

不带该限制时，在整个平滑区域内每轴加速度都可以达到其最大值：



带该限制时，在整个平滑区域内每轴急动度都被限制在最大值以内。因此，平滑运动通常分为 3 个阶段（① ... ③）：



① 在第 1 阶段中，逐渐提升到每轴最大加速度。此时，急动度保持恒定，等于每轴允许的最大急动度。

② 在第 2 阶段中，一直以最大加速度运行。

③ 在第 3 阶段中，每轴加速度以最大急动度降到零。

6.3.3.4 相切程序段过渡的平滑(G645)

功能

在该带平滑的连续路径运行模式中，如果至少一根轴上原始轮廓的曲率出现跃变，即使是在相切程序段过渡处也会生成平滑程序段。

在平滑运动过程中，系统会确保所有参与轴不发生加速度跃变且不超出参数设置的、与原始轮廓的最大偏差（MD33120 $MA_PATH_TRANS_POS_TOL）。

对于折线式的、不相切的程序段过渡，平滑特性 G642，参见章节“按照定义的公差开展平滑 (G642/G643) (页 513)”。

激活/取消激活

在每个数控程序段中都可以通过模态指令 G645 激活带相切程序段过渡平滑的连续路径运行。

可以选择非模态准停指令 G9 来中断连续路径运行。

或者选择以下某个指令来撤销 G645：

- 模态准停 (G60)
- 连续路径运行 G64, G641, G642, G643 或 G644

G642 - G645 的区别

G642 只平滑有尖角的程序段过渡，也就是说：至少一根轴的速度会出现跃变。 然而如果程序段为相切过渡，但是曲率呈跃变，则 G642 不会插入平滑程序段。 如果该程序段过渡以限定速度进行，则轴会进行稍稍的加速度跃变，但该跃变在急动度限制激活时不会超出设置的极限值（MD32432 $MA_PATH_TRANS_JERK_LIM）。 取决于该极限值的大小，程序段过渡处的轨迹速度有可能大幅降低。 可使用 G645 避免该情况，因为在该模式中平滑运动不会导致加速度跃变。

参数设置

通过以下机床数据可以为每轴设置 G645 时最大允许的轨迹公差：

MD33120 $MA_PATH_TRANS_POS_TOL

该轴只针对加速度不恒定的相切程序段过渡。 在对弯折的、非相切程序段过渡进行平滑时，同 G642，在该模式中 MD33100 $MA_COMPRESS_POS_TOL 的公差值生效。

参见

任意形状表面模式：基本功能 (页 545)

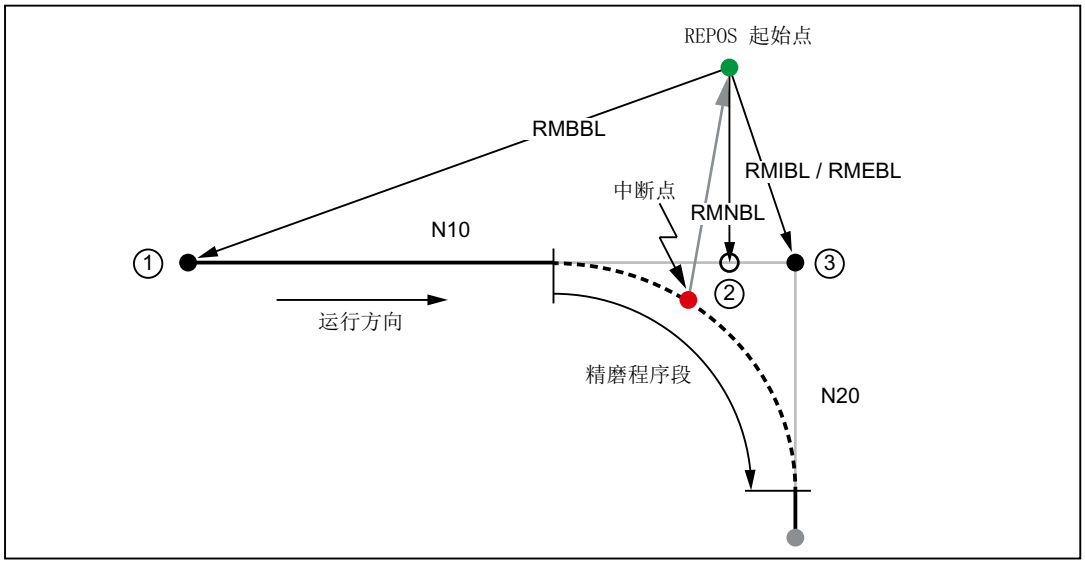
6.3.3.5 平滑和重新定位 (REPOS)

如果加工在平滑轮廓范围内中断，通过一个 REPOS 进程则**无法**再次直接定位到平滑轮廓上，而是只能定位到**编写的**轮廓上。

示例

编程：带编程平滑 G641 的两个运行程序段 N10 和 N20。

轴在平滑范围内中断运行。然后轴被手动运行到 REPOS 起始点上。根据所选 REPOS 模式重新定位至轮廓上的点 ①、② 或 ③。



RMBBL 重新定位至已中断运行程序段的开头

RMIBL 重新定位至中断点

RMEBL 重新定位至已中断运行程序段的末尾

RMNBL 重新定位至下一个轮廓点

① 程序段开头 N10

② 从 REPOS 起始点至下一个轮廓点

③ 程序段末尾 N10 / 程序段开头 N20

6.3.4 预读

6.3.4.1 标准功能

功能

预读是连续路径运行(G64、G64x)下的一个功能，它可以预先决定当前程序段以后的几个零件程序段的速度控制。

说明

预读功能仅限用于轨迹轴，不用于主轴和定位轴。

不使用预读功能时，如果相连程序段的行程很短，则处理每条程序段时只能达到某个速度，该速度值可以使轴在程序段终点制动并遵守加速度极限值。这说明，根本没有达到编程速度。

使用预读功能后，如果程序段过渡接近相切，则可以跨多条程序段进行加速和减速过程，对于短行程也可以获得高进给率。

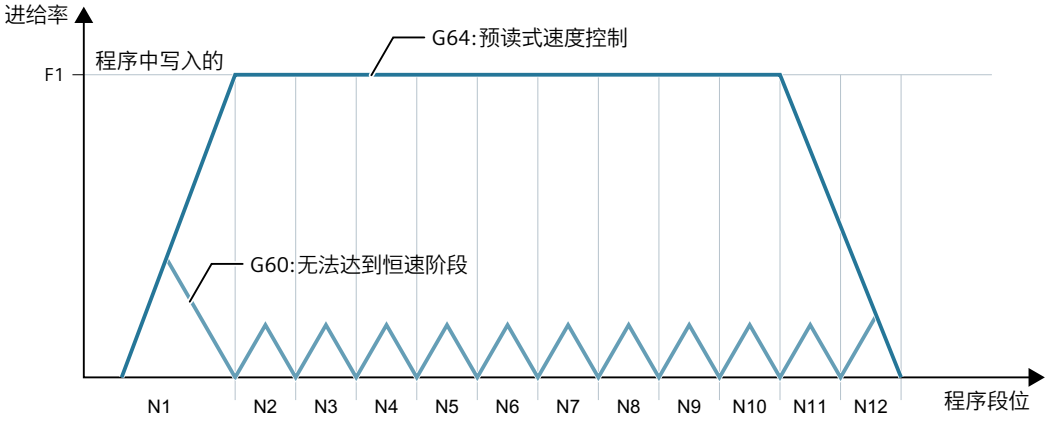


图 6-5　短行程程序段中 G60 准停和 G64 （带预读连续路径运行）之间的速度特性对比

此时，系统会根据速度极限提前对轴进行制动，避免超出加速度极限和速度极限。

预读考虑了可以预见的速度极限：

- 程序段结束时的准停
- 程序段中的速度极限
- 程序段中的加速度极限
- 程序段切换时的速度极限
- 程序段切换时同步

工作原理

预读功能预先分析各程序段中可预见的速度极限，然后确定对应的制动斜坡。预读功能会自动根据程序段长度、制动能力和允许的轨迹速度加以调整。

出于安全原因，最后一条预处理程序段终点处的速度被视为 0，因为下一条程序段可能很短或者可能是准停程序段，而轴必须在达到程序段终点处前已经停止，

在一些具有高设定速度和极短行程的多条程序段中，可以根据当前预读出的速度值提高速度，以达到目标设定速度，然后再次降低该速度，在最后一条预读程序段终点处速度降为 0。由此会获得锯齿形的速度特性曲线（见下图），锯齿可以通过降低设定速度或提高预读程序段数量来加以避免。

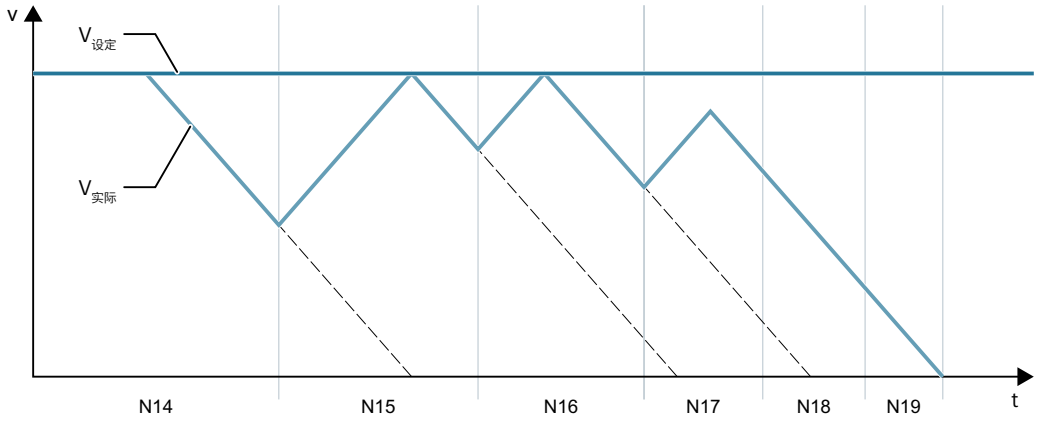


图 6-6　示例：跨程序段的速度控制（预读程序段数量为 2）

激活/禁用

预读通过选择连续路径运行 G64, G641, G642, G643, G644 或 G645 激活。

可以选择非模态指令准停 G09 来中断连续路径运行。

选择模态指令 G60 来取消预读。

参数设置

程序段数量

为确保连续路径运行中的安全运行，必须修改多条程序段的进给率。预读程序段的数量是系统内部自动确定的，但是也可以通过机床数据手动限制。该数据缺省值为 1，也就是说只预读后一条程序段，以进行速度控制。

预读主要用于短程序段（相对于制动行程而言），因此对于预读制动而言，程序段数量非常重要。将行程长度视为制动行程（即从最大速度到静止所需的行程），就已经足够。

如果一台机床的轴加速度比较低，为 $a = 1\ m/s^2$；而进给率比较高，为 $v_{轨迹} = 10\ m/min$，则在系统程序段处理周期 TB = 10 ms 时，得出的预读程序段数量 $n_{预读}$为：

$n_{预读}$= 制动行程/ 程序段长度= $(v_{轨迹}^2 / (2a)) / (v_{轨迹} * TB) = 9$

在该条件下，不建议预读超过 10 个程序段的进给率。指定的预读程序段数量会改变预读算法以及存储器占用。

在一个程序中加工速度通常是低于最大速度的，因此预读程序段数量比实际需要的多，从而不必要的占用计算性能。因此，实际所需的程序度数量是由速度和以下系数相乘后的结果得出的：

- 编程速度 * MD12100 $MN_OVR_FACTOR_LIMIT_BIN
  （使用**二进制**码的进给倍率开关时）
- 编程速度 * MD12030 $MN_OVR_FACTOR_FEEDRATE[**30**]
  （使用**格雷**码的进给倍率开关时）

MD12100 的值或者 MD12030 的第 31 个倍率值确定了动态响应裕量，它预留用于速度控制以提高轨迹进给率。

说明

MD12030 的第 31 个倍率值建议设为实际使用的倍率系数。

说明

预读程序段数量由受到插补缓冲器内允许的程序段数量的限制。

速度特性

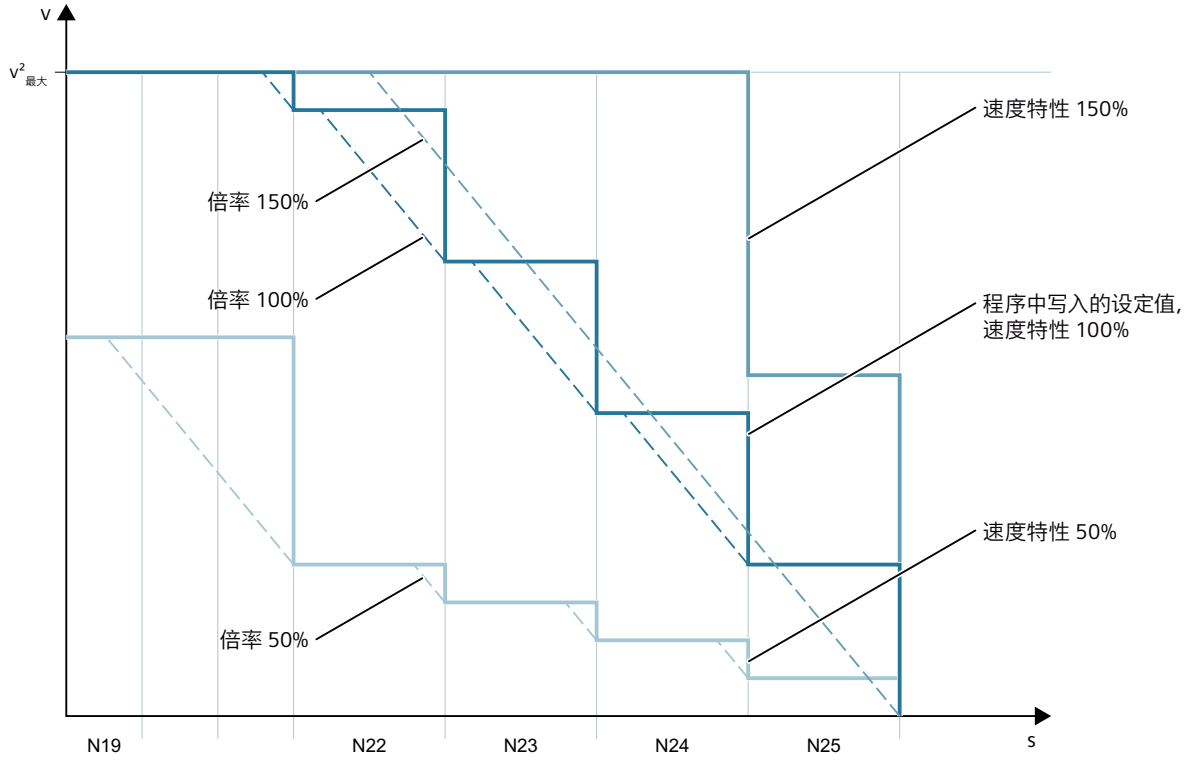
除了可以预见的固定速度极限，预读功能还可以处理编程的速度，这可以使当前程序段以外的程序段获得较小的速度。

- **确定后续程序段的速度**
  速度特性包括对后续程序段速度的检测。
  根据当前程序段和后续程序段的信息，预读功能计算出速度特性，并为当前超出值减去所需的速度修调量。
  速度特性的最大值由最大轨迹速度限制。
  此功能通过考虑修调值，可以在当前的程序段中降低速度，这样可以在下一个程序段开始时就到达较小的速度值。如果降低速度所需的时间比当前程序段中的进给时间，则在下一个程序段中继续降低速度。速度控制功能只用于后续程序段。
  该功能可由以下机床数据激活:
  MD20400 $MC_LOOKAH_USE_VELO_NEXT_BLOCK

值	含义
TRUE	功能生效
FALSE	功能**无效**

- **确定倍率基准值**
  如果后续程序段的速度特性不足够，比如：因为使用的倍率值过高（200%）或者因为激活了恒定切削速度 G96/G961，从而必须再次降低后续程序段中的速度，那么预读功能便会预读多个程序段，降低编程速度：
  通过定义的倍率基准值预读功能便可以为每个基准值计算出一个基准速度特性。从该特性中推导出当前倍率所需的速度降低幅度。
  速度特性的最大值由最大轨迹速度限制。
  最高基准值应该覆盖以下机床数据中的最大值能达到的速度范围：
  MD12030 $MN_OVR_FACTOR_FEEDRATE[n]（轨迹进给倍率开关的权重系数）
  以下机床数据值也可以达到该速度范围：
  MD12100 $MN_OVR_FACTOR_LIMIT_BIN（二进制编码的倍率开关的限制）
  因此可以避免速度指令所在的程序段出现减速情况。
  如果在 100%倍率条件下就需要达到明显的跨程序段减速，则建议在低倍率区内也设置一个基准值。
  使用的每通道倍率基准值的数量由以下机床数据确定：
  MD20430 $MC_LOOKAH_NUM_OVR_POINTS（预读中倍率开关基准值的数量）
  对应的基准值在以下机床数据中确定：
  MD20440 $MC_LOOKAH_OVR_POINTS（预读中倍率开关基准值）
  示例：
  基准速度特性，其中：
  – 倍率 = 50 %、100 % 或 150 %
  – 预读程序段数量 = 4
  – MD20430 $MC_LOOKAH_NUM_OVR_POINTS = 2
  – MD20440 $MC_LOOKAH_OVR_POINTS = 1.5, 0.5

– MD20400 $MC_LOOKAH_USE_VELO_NEXT_BLOCK = 1



可以组合这两种方法（即确定后续程序段速度和确定倍率基准值）来确定速度特性，通常我们也建议如此做，因为在该功能机床数据的缺省值中，已经覆盖了倍率相关速度限制的最大范围。

说明

如果不使用任何一种方法，则只有在当前程序段中才按设定速度移动。

说明

可预见的速度限制会限制倍率相关的速度限制。

出现程序段周期问题时的稳定系数

如果待处理程序段的行程过短，迫使预读功能降低机床速度，为程序段处理提供充足的时间，那么便会出现程序段处理周期问题。此时轨迹运动可能会出现持续的加速和减速制动过程。

这种速度波动可以通过稳定系数加以抑制：

MD20450 $MC_LOOKAH_RELIEVE_BLOCK_CYCLE（程序段处理周期的稳定系数）

边界条件

轴专用的进给停止/轴禁用

轴专用的进给停止/轴禁用不在预读功能的考虑范围内。

如果需要插补一根轴，而该轴又需要依据轴专用的进给停止/轴禁用指令停止，那么预读功能不会在该程序段前停止轨迹运动，而是**在该程序段中**停止运动。

如果不希望出现该特性，则可以将**轴专用**的进给停止改为**通道专用**的进给停止，这样轨迹运动就会立即停止了（参见（功能手册章节*“监控和补偿”*中的*“夹紧监控”*）。

6.3.4.2 任意形状表面模式：扩展功能

功能

使用“任意形状表面模式：扩展功能”是预读标准功能的一项扩展，用于计算任意形状表面加工中的轨迹速度特性，参见章节“任意形状表面模式：基本功能 (页 545)”。

该功能优化了连续路径运行，主要有：

- 对称的加速和制动特性
- 均匀的加速过程，即使是急动度限制或加速度限制变化时也是如此
- 均匀的设定速度加速过程，不管是否带设定的动态响应限制。
- 按更低的设定速度“预读式”制动

均匀的加速过程以及对动态响应限制的符合使设定速度特性更加平滑，它可以将跟随误差对表面质量的影响降到最低程度。

优点

- 工件表面质量更加均匀
- 机床负载更轻

应用

使用“任意形状表面模式：扩展功能”用于加工主要由任意形状表面构成的工件。

说明

在标准加工中，该功能不体现任何优势，因此在标准加工中建议使用预读标准功能。

有效性

该功能只在以下条件下生效：

- AUTO 运行方式生效
- “软加速模式 (SOFT)”生效

参数设置

工作存储器

通过以下机床数据配置了“任意形状表面模式：扩展功能”所需的存储空间：

MD28533 $MC_MM_LOOKAH_FFORM_UNITS = <值>

所需存储器取决于零件程序、程序段长度、轴动态响应以及激活的坐标转换。

针对任意形状表面加工的参考值为：MD28533 = 18

说明

由于该功能会占用内存，因此建议只为参与任意形状表面加工的通道设置 MD28533。

插补缓冲器中的 NC 程序段数量

通常建议在使用“任意形状表面模式：扩展功能”时大幅提高插补缓冲器中的程序段数量：

MD28060 $MC_MM_IPO_BUFFER_SIZE > 100

程序段内存太小可能会降低轨迹速度特性的均匀度。

激活/禁用

可以为每个动态响应模式单独地激活或取消该功能，详见章节“轨迹插补的动态响应模式 (页 542)”。

MD20443 $MC_LOOKAH_FFORM[<n>]= <值>

下标 <n>	动态响应模式	<值>	任意形状表面模式：扩展功能
0	标准动态响应设置 (DYNNORM)	0	关闭
		1	启用
1	定位运行，攻丝 (DYNPOS)	0	关闭
		1	启用
2	粗加工 (DYNROUGH)	0	关闭
		1	启用

下标 <n>	动态响应模式	<值>	任意形状表面模式：扩展功能
3	初精整（DYNSEMIFIN）	0	关闭
		1	启用
4	精加工（DYNFINISH）	0	关闭
		1	启用
5	精修整（DYNPREC）	0	关闭
		1	启用

通常只有在一同激活了“任意形状表面模式：基本功能”时，“任意形状表面模式：扩展功能”才生效。MD20443 $MC_LOOKAH_FFORM[<n>] 中的设置因此应该和 MD20606 $MC_PREPDYN_SMOOTHING_ON[<n>] 一致。

在“任意形状表面模式：扩展功能”关闭的动态响应模式中，预读标准功能生效。

编程

通常在程序中切换动态响应模式时，“任意形状表面模式：扩展功能”生效。

示例

已进行了以下参数设置：

MD20443 $MC_LOOKAH_FFORM[0] = 0

MD20443 $MC_LOOKAH_FFORM[1] = 0

MD20443 $MC_LOOKAH_FFORM[2] = 1

MD20443 $MC_LOOKAH_FFORM[3] = 1

MD20443 $MC_LOOKAH_FFORM[4] = 1

MD20443 $MC_LOOKAH_FFORM[5] = 1

在零件程序中切换动态响应模式：

程序代码	注释
N10 DYNPOS	; 激活动态响应模式 DYNPOS。 在该模式中，预读标准功能生效。
...	
N100 G17 G54 F10000	
N101 DYNFINISH	; 激活动态响应模式 DYNFINISH。 在该模式中，“任意形状表面模式：扩展功能”生效。
N102 SOFT G642	

程序代码	注释
N103 X-0.274 Y149.679 Z100.000 G0	
N104 COMPCAD	
...	
N1009 Z4.994 G01	
N10010 X.520 Y149.679 Z5.000	
N10011 X10.841 Y149.679 Z5.000	
N10012 X11.635 Y149.679 Z5.010	
N10013 X12.032 Y149.679 Z5.031	
M30	

说明

在预读标准功能和“任意形状表面模式：扩展功能”之间切换时，连续轨迹运行会因插补器停止而中断。

边界条件

自动切换

使用以下功能会自动切换到预读标准功能：

- 螺纹切削/攻丝(G33, G34, G35, G331, G332, G63)
- 轨迹主轴耦合
- 步冲/冲压
- 坐标 PTP 运动

随后“任意形状表面模式：扩展功能”再次自动激活。

使用 G 功能组 15 的指令（进给率类型）

使用“任意形状表面模式：扩展功能”时不建议使用以下进给率类型：

- 旋转进给率(G95, G96, G97, ...)
- 反比时间进给率（G93）

使用 FLIN

使用“任意形状表面模式：扩展功能”时不可以使用进给属性 FLIN。

进给补偿的影响

和预读标准功能相比，通过机床控制面板或 $AC_OVR 设定的进给倍率可能会大大延长运行时间。

通过快速运行 (G0) 切换

预读功能不会切换到任意形状表面模式中零散的 G0 程序段（在预读标准功能和“任意形状表面模式：扩展功能”之间切换时）。也就是说：虽然 G0 和标准动态响应模式生效（DYNNORM），但是为 DYNNORM 预设的预读标准功能（→ MD20443 $MC_LOOKAH_FFORM）不会一同自动生效。由于保留了当前激活的预读功能，因此通常 G0 程序段和多项式程序段经过“平滑”后会变得平整，从而可以达到更加均匀的速度特性。

6.4 动态响应自适应功能

6.4.1 轨迹速度平滑

引言

速度控制会充分利用所给定的轴动态响应。如果无法达到编程进给率，则系统会使轨迹速度逼近设置的轴极限值和轨迹极限值，这会导致在轨迹上频繁制动和加速。

在高轨迹速度条件下先进行短暂的加速，再进行制动时，无法明显地缩短加工时间。加速过程反而会导致异常，比如：引发机床共振。

在一些应用，比如：模具制造和高速铣削中，需要保持均匀的轨迹速度。 这种应用中就应该舍弃短暂的加速过程，以保持轨迹速度的平稳。

功能

在“轨迹速度平滑”生效时，一个平滑系数会生效，以保持平稳的轨迹速度。该系数会确定允许的最高生产率下降幅度。 如果一个加速过程缩短的程序运行时间比该系数短，则不执行该加速过程。 此处只考虑那些频率高于设定的参与轴极限频率的加速过程。

优点：

- 避免因为频繁的加速和制动过程（在几个插补周期内）导致机床共振。
- 避免对程序运行时间影响不大的加速过程导致切削速度持续变化。

说明

轨迹速度平滑功能对轮廓误差没有影响。

因此，以恒定轨迹速度加工时，轮廓曲率导致的轴速度波动还是无法通过该功能加以避免。同样，由于设定了新的进给率而导致的轨迹速度波动也无法抑制。避免该波动属于编程人员的责任。

前提条件

- 只有在 SOFT 和 BRISK 中激活了带预读的连续路径运行时，轨迹速度平滑才生效。 该平滑在 G0 时不生效。
- 控制系统的周期时间应合理设置，使预处理程序可以处理足够多的程序段，以分析加速过程。

激活/取消激活

轨迹速度平滑功能通过以下机床数据激活/取消：

MD20460 $MC_LOOKAH_SMOOTH_FACTOR（预读中的平滑系数）

值	含义
0.0	**不激活**轨迹速度平滑（缺省值）
> 0	激活轨迹速度平滑

只有在通过重新配置后该机床数据的修改才生效。

参数设置

平滑系数

平滑系数通过以下通道专用的机床数据设置：

MD20460 $MC_LOOKAH_SMOOTH_FACTOR（预读中的平滑系数）

该百分比值确定了和有加速/制动过程相比时，无加速/制动过程的工步可以最多延长多少时间。

该值应设为“最差”值，即假设一个程序内除了首个运动外的所有加速过程都被平滑。 实际上延长的时间肯定更短，不满足任何加速条件时，延长时间甚至为0。 当然，也可以输入 50% 到 100% 的值，以避免明显延长加工时间。

考虑编程进给率

轨迹速度的平滑可以考虑和不考虑编程进给率。 通过以下机床数据进行选择：

MD20462 $MC_LOOKAH_SMOOTH_WITH_FEED（轨迹平滑考虑编程进给率）

值	含义
0	不考虑编程进给率。
1	考虑编程进给率（缺省值）。

在考虑编程进给率时，如果倍率为 100 %，会更好地保持 MD20460 确定的平滑系数。

轴专用的极限频率

轴专用的极限频率通过以下机床数据确定：

MD32440 $MA_LOOKAH_FREQUENCY （预读时的平滑频率）

如果加速和制动过程的频率过高，则系统会根据以下机床数据的设置平滑加速和制动过程，或降低动态响应：

MD20460 $MC_LOOKAH_SMOOTH_FACTOR（预读中的平滑系数）

MD20465 $MC_ADAPT_PATH_DYNAMIC（轨迹动态响应自适应）

有关 MD20465 的更多信息参见章节“轨迹动态响应自适应 (页 535)”。

说明

如果某轴机械部件出现振动，而且其振动频率已知，则应该将 MD32440 设为低于该频率的值。
比如：共振频率可以通过内置的测量功能测出。

工作原理

从轨迹参与轴中自动计算出 MD32440 的最小值（= $f_{轨迹}$）。 平滑功能只考虑在以下时间内再次回到运动起始速度或结束速度的加速过程：

$t = t_2 - t_1 = 2 / f_{轨迹}$

如果加速过程导致的加工时间延长超出平滑系数（MD20460）设置的极限值，则舍弃该加速过程。

示例

已进行了以下参数设置：

MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 10 %

MD32440 $MA_LOOKAH_FREQUENCY[AX1] = 20 Hz

MD32440 $MA_LOOKAH_FREQUENCY[AX2] = 20 Hz

MD32440 $MA_LOOKAH_FREQUENCY[AX3] = 10 Hz

在该轨迹上有 3 根轴参与：X = AX1, Y = AX2, Z = AX3。

3 根轴上 MD32440 的最小值为 10 Hz， 因此只检查在时间段“ $t_2 - t_1 = 2 / 10$ Hz = 200 ms ”内执行的加速过程。 t_2 是从速度 v_1 开始进行加速后再次回到 v_1 的时间点。 只有该时间段才会一同考虑进延长的加工时间。

如果时间段“$t_2 - t_1$”超过 200 毫秒或者额外的程序运行时间“$t_3 - t_2$”超过“$t_2 - t_1$”的 10 % (= MD20460)，则进行以下动作：

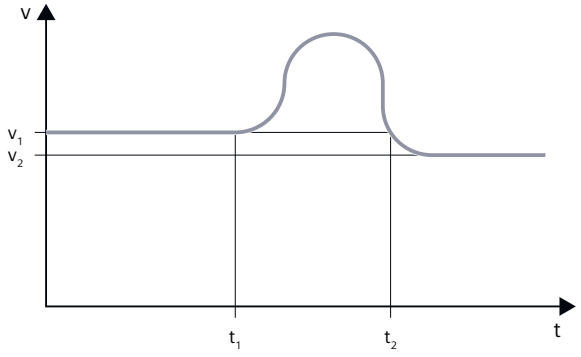


图 6-7　轨迹速度在时间上最优，无平滑

如果与此相反，时间段“$t_2 - t_1$”短于 200 毫秒或者额外的程序运行时间“$t_3 - t_2$”没有超过“$t_2 - t_1$”的 10 %)，则进行以下动作：

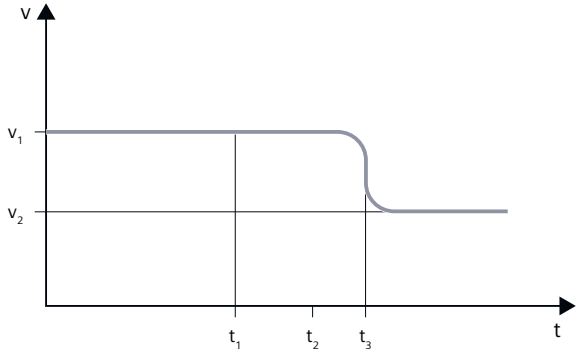


图 6-8　平滑轨迹速度

6.4.2 轨迹动态响应自适应

功能

加工期间高动态的加速和制动过程可能会引起机床机械振动，从而降低工件的表面质量。

利用功能“轨迹动态响应自适应”可以根据机床具体条件来调整加速和制动过程的动态响应。

说明

该功能不考虑参与轨迹的单轴的加速和制动过程，只考虑整个轨迹的加速和制动过程。 因此，即使在保持恒定轨迹速度条件下，也可能由于不均匀的轮廓变化或者坐标转换，导致一些引发机械振动的加速和制动过程。

有效性

“轨迹动态响应自适应”只在轨迹运动期间有效：

- 连续路径运行（G64, G64x）
  在连续路径运行中，当倍率为 100 % 时，可以达到最佳的动态响应自适应效果。 当倍率明显低于该值时或者有功能导致轨迹轴制动时（比如：输出给 PLC 的辅助功能），自适应效果会明显变差。
- 准停（G60）

在以下条件下，即使在轨迹运动中，“轨迹动态响应自适应”也**不生效**：

- 编程了快速运行（G0）
- 倍率值变化
- 在运动期间出现停止请求（如 NC 停止、NC 复位）
- “速度相关的轨迹加速度”（DRIVE）功能激活

激活/取消激活

该功能通过以下机床数据激活/取消：

MD20465 $MC_ADAPT_PATH_DYNAMIC（轨迹动态响应自适应）

值	含义
= 1.0	**不激活**轨迹动态响应自适应（缺省值）
> 1.0	激活轨迹动态响应自适应

在激活该功能后，连续路径运行内部会自动一**同激活“轨迹速度平滑”**，参见章节“轨迹速度平滑 (页 531)”。

如果 MD20460 $MC_LOOKAH_SMOOTH_FACTOR 平滑系数被设为 0（即取消功能，缺省值！），则使用 100 % 的平滑系数。 如果平滑系数不是 0 %，则按其原本设置值生效。

参数设置

轨迹动态响应调整系数

通过轨迹动态响应调整系数可以按下调后的动态响应限值执行一些短暂的轨迹速度变化。

调整系数可以针对各个通道单独设置：

- 如果是不带急动度限制的刚性加速 (**BRISK**)，则该系数通过以下机床数据设置：
  MD20465 $MC_ADAPT_PATH_DYNAMIC[**0**]
  → 调整系数作用于加速度。
- 如果是带急动度限制的软加速 (**SOFT**)，则该系数通过以下机床数据设置：
  MD20465 $MC_ADAPT_PATH_DYNAMIC[**1**]
  → 调整系数作用于急动度。

轴专用的极限频率

动态响应限制功能应只用于那些机械振动频率大于定义的极限频率，从而导致机床共振的加速和制动过程。

启用动态响应限制功能的极限频率可以通过以下机床数据针对各轴单独设置：

MD32440 $MA_LOOKAH_FREQUENCY （预读时的平滑频率）

更多信息参见章节“轨迹速度平滑 (页 531)”。

工作原理

在加工期间，控制系统会周期性地从所有轨迹参与轴中计算出所有极限频率中的最小值，将该值用作启用动态响应自适应的极限频率(f)，由此计算出对应的时间窗口（$t_{自适应}$）

$t_{自适应} = 1 / f$

$t_{自适应}$的大小确定了后续特性：

1. 速度变化所需时间短于 $t_{自适应}$：
   加速度按照大于 1 的系数降低，小于等于以下机床数据：
   MD20465 ADAPT_PATH_DYNAMIC（轨迹动态响应自适应）
   降低加速度后，速度变化时间会延长。
   有以下几种情况：
   – 加速度降到比 MD20465 小的值，以持续 $t_{自适应}$[s]。 不一定要充分利用允许的降低幅度。
   – 加速时间按照 MD20465 值缩短。 加速度虽然更低，但时间比 $t_{自适应}$ 短。 充分利用了允许的降低幅度。
2. 速度变化所需时间长于 $t_{自适应}$：
   无需进行动态响应自适应。

示例

下面的示例生动地地展示了“轨迹动态响应自适应”对刚性加速 BRISK 的影响：

已进行了以下参数设置：

MD20465 $MC_ADAPT_PATH_DYNAMIC[0] = 1.5

MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 1.0

MD32440 $MA_LOOKAH_FREQUENCY[AX1] = 20 Hz	T_{AX1} = 1/20 Hz = 50 ms
MD32440 $MA_LOOKAH_FREQUENCY[AX2] = 10 Hz	T_{AX2} = 1/10 Hz = 100 ms
MD32440 $MA_LOOKAH_FREQUENCY[AX3] = 20 Hz	T_{AX3} = 1/20 Hz = 50 ms

说明

为了清晰地展示动态响应自适应的效果，平滑系数 MD20460 被设为 1，因此实际上一同激活的“轨迹速度平滑”功能并未起效。

在该轨迹上有 3 根轴参与：X = AX1, Y = AX2, Z = AX3。

在 AX2 参与的轨迹运动中，所有短于 T_{AX2} 的加速和制动过程都被调整。

在只有 AX1 和/或 AX3 参与的轨迹运动中，所有短于 $T_{AX1} = T_{AX3}$ 的加速和制动过程都被调整。

相关的时间窗口在下图中标为 $t_{自适应...}$。

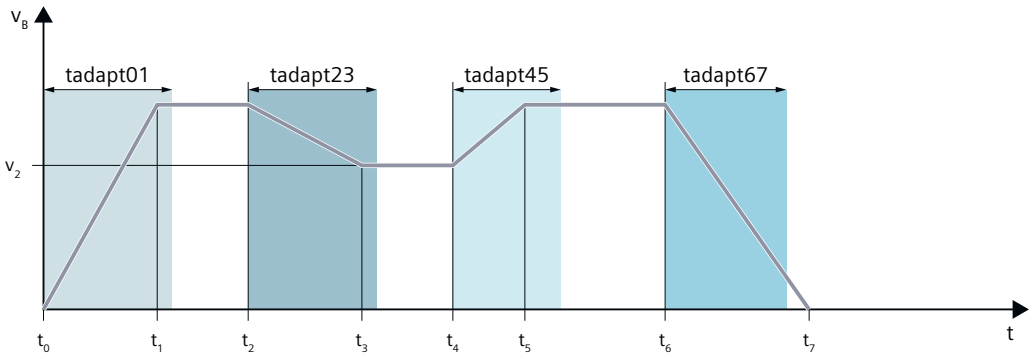


图 6-9　时间最优的轨迹速度曲线，无平滑，无动态响应自适应

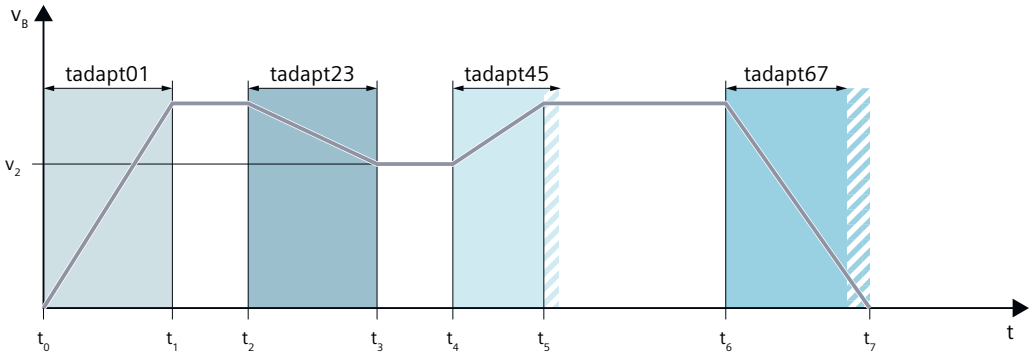


图 6-10　轨迹速度曲线，有动态响应自适应

间隔“t_0 - t_1”和“t_2 - t_3”:　由于对加速度的调整，“t_0 - t_1”之间的加速过程和“t_2 - t_3”之间的制动过程延长到 $t_{自适应\ 01}$ 或 $t_{自适应\ 23}$。

间隔“t_4 - t_5”:　“t_4 - t_5”之间的加速过程以按最大调整系数 1.5 下调后的加速度进行。 但是，加速过程在 $t_{自适应\ 45}$ 前结束。

间隔“t_6 - t_7”:　“t_6 - t_7”之间的制动过程保持不变，因为它持续的时间比 $t_{自适应\ 67}$ 长。

6.4.3　确定动态响应极限值

调试功能“轨迹动态响应自适应”时除了要确定轨迹轴固有频率，以确定轴专用的极限频率（MD32440 $MA_LOOKAH_FREQUENCY）外，还需要确定速度、加速度和急动度的动态响应极限值。

步骤

下文将介绍如何通过软加速 SOFT 来确定轨迹轴的动态响应极限值。 这些步骤同样适用于刚性加速 BRISK。

1. 取消“轨迹动态响应自适应”：
   MD20465 $MC_ADAPT_PATH_DYNAMIC [1] = 1
2. 检查在不同运行速度下各轨迹轴的定位特性。 设置合适的急动度，使运行保持理想的定位公差。

   说明

   定位开始时的运行速度越高，通常急动度就可以设置得越高。

3. 采用安全运行速度允许的最大急动度：
   MD32431 $MA_MAX_AX_JERK（最大急动度）
4. 确定所有轨迹轴的系数 F_{APD}，其中：
   F_{APD} = 得出的最大急动度/得出的最小急动度

   说明

   得出的最小急动度是在临界运行速度条件下的急动度。

5. 输入所有轨迹轴上得出的最大系数 F_{APD}，它将用作动态响应调整系数：
   MD20465 $MC_ADAPT_PATH_DYNAMIC [1] = F_{APD}

6.4.4 “轨迹速度平滑”和“轨迹动态响应自适应”的共同作用

下面的示例生动地说明了在连续路径运行中，“轨迹速度平滑”和“轨迹动态响应自适应”的共同作用效果。

示例 1

加速模式：BRISK

在该轨迹上有 3 根轴参与：X = AX1, Y = AX2, Z = AX3。

已进行了以下参数设置：

MD20465 $MC_ADAPT_PATH_DYNAMIC [0] = 3

MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 80.0

MD32440 $MA_LOOKAH_FREQUENCY[AX1] = 20　　T_{AX1} = 1/20 Hz = 50 ms

MD32440 $MA_LOOKAH_FREQUENCY[AX2] = 20　　T_{AX2} = 1/20 Hz = 50 ms

MD32440 $MA_LOOKAH_FREQUENCY[AX3] = 20　　T_{AX3} = 1/20 Hz = 50 ms

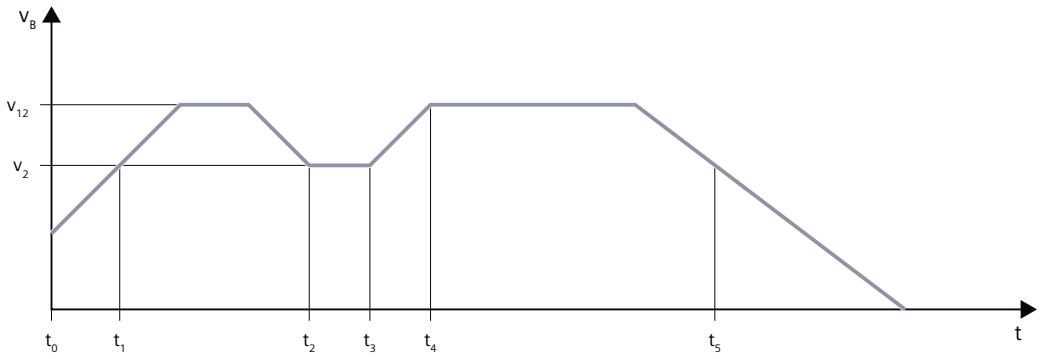


图 6-11　时间最优的轨迹速度曲线，无平滑，无动态响应自适应

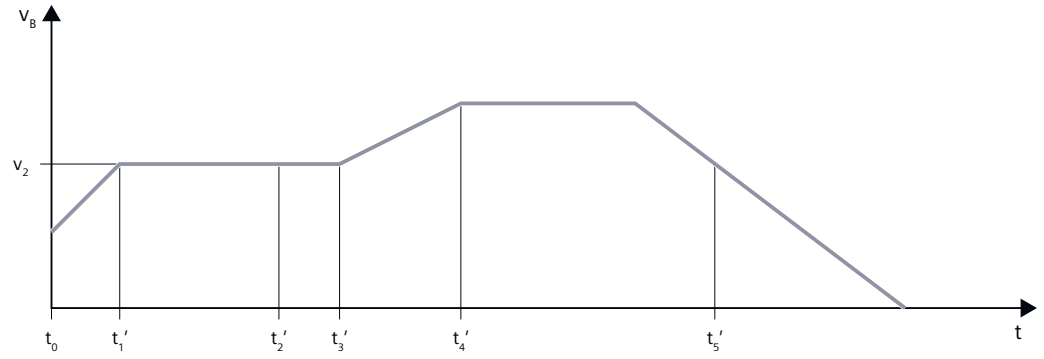


图 6-12　轨迹速度曲线，轨迹速度平滑和轨迹动态响应自适应共同作用

轨迹速度平滑的作用：

间隔“t_1 - t_2”:　由于不带加速到 v_{12} 这一过程时，延长的加工时间比平滑系数 80 % 得出的时间短，因此跳过了“t_1 - t_2”之间的加速和制动过程。

间隔“t_3 - t_5”:　“t_3 - t_5”之间的加速和制动过程不满足这一条件，或者长于设置的平滑时间 T_{Axn} = 2/20 Hz = 100 ms。

动态响应自适应的作用：

间隔“t_3 - t_4”:　“t_3 - t_4”的加速过程短于 MIN(T_{AXn}) = 1/20 Hz = 50 ms，因此采用按调整系数 3 下调后的加速度。

间隔“开始到t_1”：　由于动态响应自适应，t_1 前进行轨迹平滑后剩下的加速过程被延长到 t_1'。

说明

该示例表明，即使轨迹速度平滑功能没有消除一些加速或制动过程，这些过程也会接着由动态响应自适应功能加以优化。因此，应尽量同时激活这两个功能。

示例 2

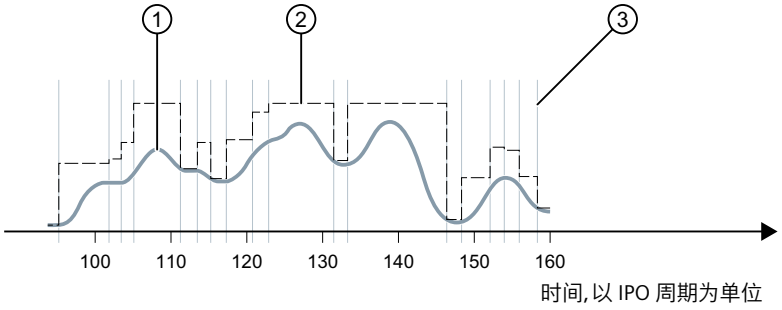
加速模式：SOFT

在该轨迹上有 3 根轴参与：X = AX1, Y = AX2, Z = AX3。

已进行了以下参数设置：

MD20465 $MC_ADAPT_PATH_DYNAMIC[1] = 1
MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 0.0
MD32440 $MA_LOOKAH_FREQUENCY[AX1] = 10　　T_{AX1} = 1/20 Hz = 100 ms
MD32440 $MA_LOOKAH_FREQUENCY[AX2] = 10　　T_{AX2} = 1/20 Hz = 100 ms
MD32440 $MA_LOOKAH_FREQUENCY[AX3] = 20　　T_{AX3} = 1/20 Hz = 50 ms

这些设置会产生时间最优的轨迹速度曲线，没有轨迹速度平滑和动态响应自适应：

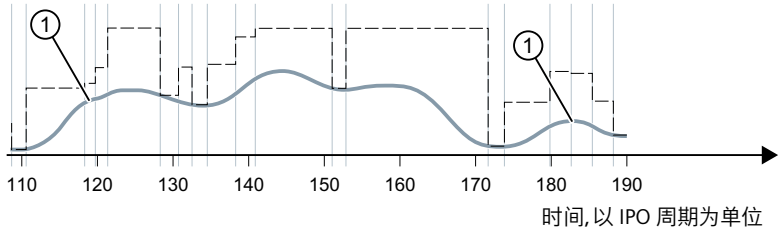


① 轨迹速度曲线
② 根据轴动态响应或程序写入的进给率来限制轨迹速度
③ 程序段切换标记

进行以下参数设置：

```
MD20465 $MC_ADAPT_PATH_DYNAMIC[1] = 4
MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 1.0
```

这些设置会产生带动态响应自适应、最低程度（几乎关闭）轨迹速度平滑的轨迹速度曲线：



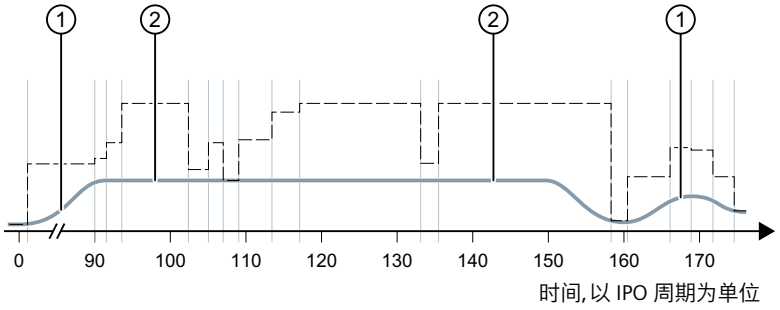
① 低轨迹急动度使轨迹速度更平稳

平滑系数从 1 % 改为 0 %（即缺省值！）：

```
MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 0.0
```

该设置使 100 % 的平滑系数生效。

由此得出了一条轨迹速度平滑和轨迹动态响应自适应共同作用的轨迹速度曲线：



① 低轨迹急动度使轨迹速度更平稳

② 放弃加速和减速，轨迹平滑的影响

6.4.5 轨迹插补的动态响应模式

功能

针对不同工艺的动态响应设置可以通过机床数据确定，然后在程序中通过第 59 G 功能组指令（轨迹插补的动态响应模式）激活。

指令	激活动态响应设置
DYNNORM	标准动态响应设置
DYNPOS	定位模式，攻丝
DYNROUGH	粗加工
DYNSEMIFIN	初精整
DYNFINISH	精加工
DYNPREC	精修整

说明

第 59 G 功能组指令（轨迹插补的动态响应模式）只能激活**轨迹轴**的动态响应。它对以下轴没有影响：

- 定位轴
- PLC 轴
- 指令轴
- 基于轴耦合的运动
- 手轮叠加运动
- JOG 运动
- 参考点运行 (G74)
- 运行至固定点 (G75)
- 快速运行 （G0）

对于这些运动而言，始终是标准动态响应设置 (DYNNORM) 生效。

应用

比如：通过不同的动态响应设置，工件的粗加工可以达到时间最优，而精加工可以获得最佳工件表面。

参数设置

轴专用的动态响应设置：

- MD32300 $MA_MAX_AX_ACCEL[<n>]（轴加速度）
- MD32310 $MA_MAX_ACCEL_OVL_FACTOR[<n>]（用于限制速度跳跃的过载系数）
- MD32431 $MA_MAX_AX_JERK[<n>]（轨迹运动时的最大轴急动度）
- MD32432 $MA_PATH_TRANS_JERK_LIM[<n>]（连续路径运行中程序段过渡处的最大轴急动度）
- MD32433 $MA_SOFT_ACCEL_FACTOR[<n>]（SOFT 指令时的最大加速度比例）

通道专用的动态响应设置：

- MD22450 $MC_DYN_LIM_MODE[<n>]（单轴速度限制或几何尺寸上的速度限制）
- MD20600 $MC_MAX_PATH_JERK[<n>]（最大轨迹急动度）
- MD20602 $MC_CURV_EFFECT_ON_PATH_ACCEL[<n>]（轨迹曲率对轨迹加速度的影响）
- MD20603 $MC_CURV_EFFECT_ON_PATH_JERK[<n>]（轨迹曲率对轨迹急动度的影响）

其中下标 <n> = 0 用于 DYNNORM
1 用于 DYNPOS
2 用于 DYNROUGH
3 用于 DYNSEMIFIN
4 用于 DYNFINISH
5 适用于 DYNPREC

说明

在**写入无下标机床数据**时，系统将向该机床数据的所有数组单元内写入相同值。
在**读无下标机床数据**时，系统始终下标 0 的数组单元的值。

封锁 G 指令

我们建议通过以下机床数据封锁不使用的第 59 G 功能组指令（轨迹插补的动态响应模式）：

MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[<n>]（重新配置后的指令列表）

使用一个封锁的 G 指令时，系统会输出报警。以避免未经设置的机床数据生效。

示例

以下设置可以封锁 G 指令 DYNPOS 和 DYNSEMIFIN：

- MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[0]="DYNPOS"
- MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[1]=" "
- MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[2]="DYNSEMIFIN"
- MD10712 $MN_ NC_USER_CODE_CONF_NAME_TAB[3]=" "

更多信息

关于 G 指令组 59 的编程说明请参见
编程手册“数控编程”

使用选件“Top Speed”（位置设定值滤波器）

一根或多根轴上使用选件“Top Speed”时，系统也会针对每种动态响应模式切换 FIR 滤波器设置。

更多关于“Top Speed”的信息参见功能手册中的章节“进给轴和主轴”。

使用选件“智能负载自适应”

使用选件“智能负载自适应”时，系统会针对每种动态响应模式根据负载来单独设置加速度和急动度的比例，尤其是比例的限值。

更多关于“Top Speed”的信息参见功能手册中的章节“进给轴和主轴”。

6.4.6 任意形状表面模式：基本功能

引言

在刀具和模具制造中，工件表面的均匀度非常重要，其重要性甚至高于工件表面的精度。

不均匀的工件表面可能是由以下原因引起的：

- 工件加工程序内包含的尺寸不均匀，这主要是一些曲率和挠率的变化。

  说明

  轮廓的曲率 k 是针对轮廓上某个点的切线方向角对弧长的转动率，是半径的倒数 (k = 1/r)。挠率是曲率的变化率（1 阶导数）。

  执行程序时，如果几何尺寸不均匀，会达到机床的动态响应极限，导致多余的加速和制动过程。由此会导致轮廓误差，误差水平取决于轴的有效滞后量。

- 多余的加速和制动过程还可导致机床共振，在工件上留下加工痕迹。

为排除这些异常，可以采取以下方法：

- 使用 CAD/CAM
  用 CAD/CAM 系统创建程序，这种程序会包含非常均匀的曲率和挠率曲线，因此不会导致轨迹速度的异常减速。
- 确定合理的最大轨迹速度，避免曲率和挠率的波动产生影响。

功能

使用“任意形状表面模式：基本功能”，**对于动态的几何轴，以及定向坐标转换时参与转换的回转轴**，可以使定义的轨迹速度极限对细微的曲率或挠率变化更加“迟钝”，同时还不会超出机床关于加速度和急动度方面的动态响应限制。

使用该功能的优点有：

- 均匀的轨迹速度
- 工件表面质量更加均匀
- 缩短加工时间，条件是机床的动态响应允许

应用

该功能用于加工主要由任意形状表面构成的工件。

前提条件

只有在预留了所需内存后，才能激活该功能：

MD28610 $MC_MM_PREPDYN_BLOCKS = **10**

输入值确定了在确定轨迹速度时（速度处理）需要计算多少条程序段。
推荐值为 10。

如果 MD28610 的值为 0，则在确定一条程序段的最大轨迹速度时，只考虑该程序段中的运动。如果该机床数据值大于 0，则在确定轨迹速度时会一同考虑相邻程序段中的尺寸，从而使轨迹速度更加均匀。

激活/禁用

可以为每个动态响应模式单独地激活或取消该功能，详见章节“轨迹插补的动态响应模式 (页 542)”。

MD20606 $MC_PREPDYN_SMOOTHING_ON[<n>] = <值>

下标 <n>	动态响应模式	<值>	任意形状表面模式：基本功能
0	标准动态响应设置 (DYNNORM)	0	关闭
		1	启用
1	定位运行，攻丝 (DYNPOS)	0	关闭
		1	启用
2	粗加工 (DYNROUGH)	0	关闭
		1	启用
3	初精整（DYNSEMIFIN）	0	关闭
		1	启用
4	精加工（DYNFINISH）	0	关闭
		1	启用
5	精修整（DYNPREC）	0	关闭
		1	启用

说明

由于该功能会占用内存，因此建议只在对应加工通道中激活该功能。

参数设置

轮廓采样系数的变化

在插补曲面轮廓时产生的割线误差取决于以下因素：

- 曲率
- 插补周期（在 MD10071 $MN_IPO_CYCLE_TIME 中显示）
- 加工对应轮廓的速度

每根轴上允许的切割误差在以下机床数据中确定：

MD33100 $MA_COMPRESS_POS_TOL（压缩时的最大偏差）

如果设置的插补周期不够短，在特别弯曲的轮廓上最大轨迹速度有可能会下降。减速是为了保证工件表面的加工精度。

通过修改轮廓采样系数，可以将插补器对曲面轮廓进行采样的周期（即轮廓采样周期）设为和插补周期不一样的值。如果轮廓采样周期短于插补周期，在特别弯曲的轮廓上最大轨迹速度不会降低。

轮廓采样系数通过以下机床数据设置：

MD10682 $MN_CONTOUR_SAMPLING_FACTOR

有效的轮廓采样周期的计算公式为：

$T_s = f * T_1$

其中： T_s = 有效的轮廓采样时间
T_1 = 插补周期
f = 轮廓采样系数（MD10682 的值）

轮廓采样系数的缺省值为 1，即等于插补周期。

该系数可大于也可小于 1。

如果小于 1，则对轮廓采样精度的检查失效。

设置的采样时间不能短于设置的最短轮廓采样时间：

MD10680 $MN_MIN_CONTOUR_SAMPLING_TIME

说明

MD10680 是根据具体控制系统型号固定设置的，无法修改。

速度限制

每种动态响应模式中因曲率产生的速度限制模式在以下机床数据中确定：

MD22450 $MC_DYN_LIM_MODE [<n>] = <值>

其中下标 <n> = 0 用于 DYNNORM
1 用于 DYNPOS
2 用于 DYNROUGH
3 用于 DYNSEMIFIN
4 用于 DYNFINISH
5 适用于 DYNPREC

<值>	含义
= 0	根据参与轨迹的几何轴的加速度来限制速度 该设置会导致轨迹速度过高，因为在对角线方向上可能会移动更快。
= 1	根据参与轨迹的几何轴的曲率来限制速度 该设置会在对角线切向和轴向切向上以相同方式限制轨迹速度。轨迹速度变化会因此更加平缓。在精加工中，建议采用该设置。

MD22450 通过以下数据影响轨迹速度的限值：

- “可编程轮廓精度”
- 最大轴加速度（MD32300 $MA_MAX_AX_ACCEL）
- 轮廓采样系数（MD10682 $MN CONTOUR_SAMPLING_FACTOR）

编程

根据机床数据设置：MD20606 $MC_PREPDYN_SMOOTHING_ON，“任意形状表面模式：基本功能”可以通过切换生效的动态响应模式在程序中激活或取消。

示例：

设置 MD20606 $MC_PREPDYN_SMOOTHING_ON[2-5] = 1 且
MD20606 $MC_PREPDYN_SMOOTHING_ON[0-1] = 0 后，便可以通过指令 DYNROUGH、DYNSEMIFIN 、DYNFINISH 和 DYNPREC 激活该功能，通过指令 DYNNORM 和 DYNPOS 关闭该功能。

参见

相切程序段过渡的平滑(G645) (页 519)

任意形状表面模式：扩展功能 (页 527)

6.5 压缩功能

6.5.1 压缩线性程序段、圆弧程序段和快速运行程序段

6.5.1.1 功能

CAD/CAM 为描述复杂轮廓而生成了大量的线性和圆弧程序段，其中一些的轨迹长度很短。

允许的最大轨迹速度常常会受到程序段切换时间的限制。自某个轨迹速度开始，预处理不再快速处理新的运行程序段并切换至主处理。

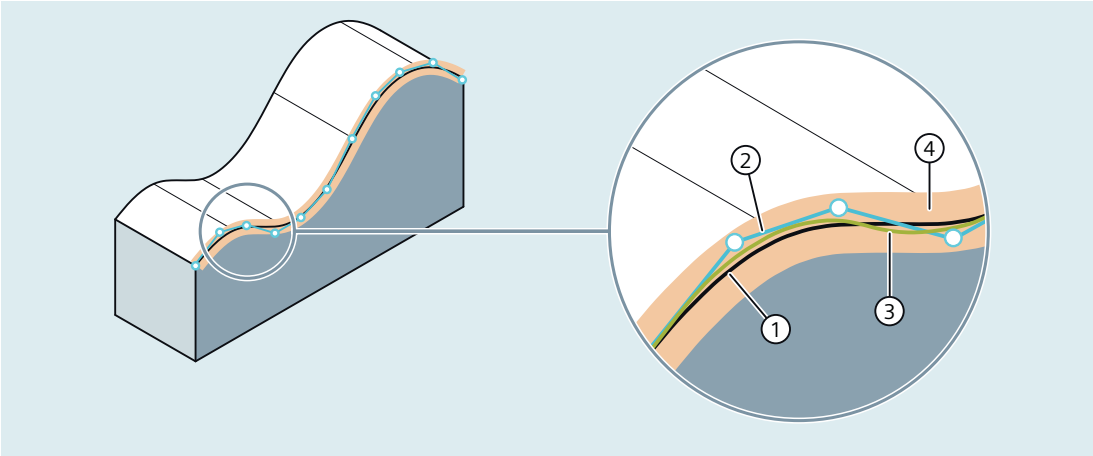
另外，程序段过渡处的加速度跃变会激发机床共振，最后降低表面质量。

甚至含 G0 的位移运动有时也可能非常短，导致在时间要求比较高的运动中，需要压缩或平滑该运动。

一种解决方案是采用所谓的“压缩器”功能。

线性程序段的压缩

压缩器功能能够将连续的线性程序段替换为轨迹长度尽可能长的多项式程序段，同时维持可参数设置的轮廓精度。



① 工件轮廓
② 压缩前：大量简短的线性程序段
③ 压缩后：一条长的多项式程序段
④ 可设定压缩公差

使用该功能的优点有：

- 提升轨迹速度
  运行程序段越少，运行速度也就越快；程序段过渡处的速度和加速度保持恒定，运行速度也就越稳定。
- 提升表面质量
  避免程序段过渡处的加速度跃变，可以使机床轴平稳运行，而不激发共振。

下表列出了可用的压缩器功能及其重要的特性：

压缩器	功能	使用说明
COMPCAD	COMPCAD 可以将**任意数量的**连续的线性程序段压缩成一条多项式程序段。	COMPCAD 非常占用内存和计算能。因此我们建议只有在 CAD/CAM 程序中的措施无法改进工件表面质量时，才使用 COMPCAD。
COMPSURF	与 COMPCAD 一样	使用 COMPSURF 比使用 COMPCAD 达到的效果更好。与 COMPCAD 一样，COMPSURF 非常占用内存和计算能。 使用 COMPSURF 可以在 CAD/CAM 程序中明显提高工件表面质量，例如在倾斜划分的简单程序上、不好的数据质量和/或不规则的点分布时。 此外，COMPSURF 还能进行相同的、可切断的铣削轨迹平滑（与方向无关），从而明显提高了两个方向上的铣削表面质量。
COMPPATH	同 COMPSURF，只是只适用于 G0 运动	COMPPATH 的作用和 COMPSURF 相同，只是它适用于 G0 运动。使用 COMPPATH 用于可以实现更快的 G0 运动。 COMPPATH 要求 ≥ 2mm 的轮廓公差（CTOL）！

在待压缩的运行程序段中、这些程序段之间含非运行指令（例如：辅助功能输出，激活绝对尺寸或增量尺寸的指令等）时，压缩进程会中断。为保证程序中的程序段是可压缩的，程序段中允许写入 G0、G1、G2、G3、轴位置、方向矢量和进给率值。所有其他指令或 G 代码都会中断压缩，轨迹位移很短时，会导致程序执行缓慢。

计算出的轨迹和程序写入轨迹之间的最大允许偏差可以通过机床数据设定（参见“参数设置 (页 552)”）。在临近轨迹内会充分利用该设定的偏差。

压缩圆弧程序段和快速运行程序段

除压缩线性程序段以外，所有压缩器功能也可用于压缩快速运行程序段（G0 程序段）。而圆弧程序段只能通过压缩器功能 COMPCAD 压缩。

圆弧和/或快速运行程序段的压缩通过以下机床数据的**百位**针对具体通道设置：

MD20482 $MC_COMPRESSOR_MODE = <值>

值	含义
0xx	不压缩圆弧程序段和 G0 程序段。这与之前的软件版本兼容。
1xx	通过 COMPCAD 将圆弧程序段线性化并压缩。 优点： 压缩器功能更加精确，并且通常由此产生更佳的表面。 缺点： 压缩器功能对 NC 程序中的缺陷更加敏感。出于兼容性原因，可能需要保留设置 0xx。
2xx	压缩 G0 程序段，其中另一公差视具体情况生效（参见“快速移动中的公差 (页 573)”）。 优点： 由于可设置更大的公差和压缩 G0 进给运动，能够更加快速流畅地运行。
3xx	上述两种方案的组合：既压缩圆弧程序段，也压缩 G0 程序段。

激活/取消

压缩器功能 COMPCAD、COMPSURF 和 COMPPATH 通过 G 指令组 30 激活或关闭（参见“编程 (页 554)”）。

6.5.1.2 参数设置

轴专用机床数据

编号	名称 $MA_	含义
MD33100	COMPRESS_POS_TOL	压缩时允许的特定轴的最大轨迹偏差

通道专用机床数据

编号	名称 $MC_	含义
MD20170	COMPRESS_BLOCK_PATH_LIMIT	可压缩 NC 程序段的最大运行长度
MD20171	SURF_BLOCK_PATH_LIMIT	使用 COMPSURF/COMPPATH 压缩时可压缩 NC 程序段的最大运行长度
MD20172	COMPRESS_VELO_TOL	压缩时允许的最大轨迹进给率偏差
MD20173	SURF_VELO_TOL	使用 COMPSURF/COMPPATH 压缩时允许的最大轨迹进给率偏差
MD20482	COMPRESSOR_MODE	压缩器的工作方式
MD20485	COMPRESS_SMOOTH_FACTOR	使用 COMPCAD 压缩用于各个动态响应模式时的平滑系数
MD20486	COMPRESS_SPLINE_DEGREE	使用 COMPCAD 压缩用于各个动态响应模式时的样条度数
MD20487	COMPRESS_SMOOTH_FACTOR_2	使用 COMPCAD 压缩用于各个动态响应模式时的回转轴平滑系数
MD28071	MM_NUM_SURF_LEVELS	COMPSURF/COMPPATH (DRAM) 功能的定义
MD28072	MM_MAXNUM_SURF_GROUPS	COMPSURF/COMPPATH 功能中关于轴组的定义 (DRAM)

通道专用设定数据

编号	名称 $SC_	含义
SD42470	CRIT_SPLINE_ANGLE	使用 COMPCAD 压缩时的角点极限角度
SD42471	MIN_CURV_RADIUS	使用 COMPCAD 压缩时的最小曲率半径
SD42472	MIN_SURF_RADIUS	使用 COMPSURF/COMPPATH 压缩时的最小曲率半径
SD42473	ACTNUM_SURF_GROUPS	COMPSURF/COMPPATH 中的最大轴组数

编号	名称 $SC_	含义
SD42475	COMPRESS_CONTUR_TOL	压缩时允许的最大轮廓偏差
SD42476	COMPRESS_ORI_TOL	压缩时刀具定向的最大偏差 **注意：** 只在定向转换有效时！
SD42477	COMPRESS_ORI_ROT_TOL	压缩时刀具旋转的最大偏差 **注意：** 只在带可旋转刀具的 6 轴机床上！

说明

角点极限角度和压缩器 COMPCAD

通过设定数据 SD42470 $SC_CRIT_SPLINE_ANGLE 设置的压缩器功能 COMPCAD 角点极限角度只是一个用于识别角点的大致角度。压缩器也会根据合理性检查将平坦的程序段过渡判定为“角点”，将“大角度”判定为“异常值”。

精优曲面（Advanced Surface）/臻优曲面（Top Surface）刀具制造和模具制造和 COMPPATH 的设置建议

压缩器功能在刀具制造和模具制造中的曲面铣削中很重要。在使用需要授权方可使用的选件“精优曲面”或“臻优曲面”时，一定要采取此处给出的设置建议！西门子 SIOS 门户网站上为您提供专门的测试程序，以便检查设置的机床数据和设定数据：

→ 精优曲面/臻优曲面测试程序 (https://support.industry.siemens.com/cs/cn/zh/view/109738423/en)

6.5.1.3 编程

直线程序段的压缩功能（取决于参数设置也包括圆弧和/或快速运行程序段）可使用 G 指令组 30 的 G 指令激活/关闭。指令模态生效。

句法

```
COMPCAD / COMPSURF / COMPPATH
...
COMPOF
```

含义

COMPCAD	激活压缩器功能 COMPCAD
COMPSURF	激活压缩器功能 COMPSURF
COMPPATH	激活压缩器功能 COMPPATH
COMPOF	关闭当前激活的压缩器功能

说明

除此以外，改善表面质量还可以使用平滑功能 G642 和急动度限制 SOFT。这些指令应写在程序开始处。

示例

示例 1：COMPSURF

程序代码	注释
N10 G00 X30 Y6 Z40	
N20 G1 F10000 G642	; 激活：平滑功能 G642
N30 SOFT CTOL=0.01	; 激活：SOFT 急动限制，轮廓公差 = 0.01
N40 **COMPSURF**	; 激活：压缩器功能 COMPSURF
N50 FIFOCTRL	
N24050 Z32.499	; 第 1 运行程序段
N24051 X41.365 Z32.500	; 第 2 运行程序段
...	
N99999 X...Z...	; 上一个运行程序段
COMPOF	; 关闭压缩器功能
...	

示例 2：COMPPATH

程序代码	注释
N100 G64	
N110 CTOL=10	; 轮廓公差 = 10
N120 **COMPPATH**	; 激活：压缩器功能 COMPPATH
N130 G1 X0.Y0.F10000	
N140 G1 X10	
N150 G0 X100	; COMPPATH 生效
N160 G0 Y300	; COMPPATH 生效
N170 G0 X10	; COMPPATH 生效
N180 G1 X0	
N190 **COMPOF**	; 关闭压缩器功能

程序代码	注释
...	

6.5.1.4 前提条件

定向转换（TRAORI）

定向转换激活时，压缩器可以压缩用于刀具定向和刀具旋转的运动程序段。

更多信息

功能手册之坐标转换；多轴转换> 方向> 刀具方向压缩

带计算的程序段搜索

在程序段搜索**类型 2** 或**类型 4**（在 ... 处**计算**的程序段搜索）中，若目标程序段所处于的程序部分中激活了压缩器功能，那么在重定位至轮廓时会逼近通过压缩器计算出的轨迹上的位置。这些位置不必与零件程序中编写的轨迹上的位置精确一致。

在程序段搜索时不能将通过压缩替换的零件程序程序段作为目标程序段。系统输出报警 15370“程序段搜索未找到搜索目标”。

6.5.2 压缩较短的样条程序段

功能

在通过 CAD/CAM 系统生成样条程序段来描述复杂轮廓时，轨迹长度较长的程序段间总是会存在轨迹长度非常短的程序段。这会迫使控制系统显著降低轨迹速度。压缩较短的样条程序段功能会生成轨迹长度尽可能长的新样条程序段。

调试

激活

“压缩较短的样条程序段”功能可以用于以下样条类型：

- BSPLINE
- BSPLINE / ORICURVE
- CSPLINE

通过以下通道专用机床数据激活：

MD20488 $MC_SPLINE_MODE，位 <n> = <值>

位	<值>	样条类型	压缩较短的样条程序段
0	0	BSPLINE	未生效
	1	BSPLINE	生效
1	0	BSPLINE / ORICURVE	未生效
	1	BSPLINE / ORICURVE	生效
2	0	CSPLINE	未生效
	1	CSPLINE	生效

边界条件

- 在待压缩运行程序段中，以及在这些程序段之间编写了非运行指令（例如辅助功能输出）时，可能会无法合并样条程序段。
- 可以合并成一个程序块的连续样条程序段的数量取决于程序段预处理内存的大小：MD28070 $MC_MM_NUM_BLOCKS_IN_PREP（预处理的程序段数量）

示例

为了能在执行运行程序段时达到较高的轨迹速度，针对 BSPLINE 插补激活“压缩较短的样条程序段”功能：

MD20488 $MC_SPLINE_MODE, 位 0 = 1

程序代码	注释
`N10 G1 G64 X0 Y0 Z0 F1000`	`; 默认设置`
`N20 G91 F10000 BSPLINE`	`; 激活：B 样条`
`N30 X0.001 Y0.001 Z0.001`	`; 从此处开始：合并较短样条程序段`
`N40 X0.001 Y0.001 Z0.001`	
`...`	

6.6 压缩、精磨和方向平滑中的轮廓/方向误差

6.6.1 功能

可以专门针对压缩、精磨和方向平滑功能设置允许的轮廓精度误差和刀具方向精度误差。

这些功能专用的轮廓/刀具方向误差在调试时通过机床数据和设定数据确定。

更多信息：章节“参数设置 (页 558)”。

在程序中，可以通过指定地址来临时性修改之前在机床数据和设定数据中确定的轮廓/刀具方向误差。

更多信息：章节“编程 (页 559)”。

在通道复位或程序结束复位后，机床数据和设定数据中确定的轮廓/刀具方向误差再次生效。

6.6.2 调试

6.6.2.1 参数设置

机床数据

单轴轮廓/刀具方向误差

MD33100 $MA_COMPRESS_POS_TOL[<轴>] = <值>（压缩时的最大偏差）

通过该机床数据可以确定每根轴上允许的最大轮廓/刀具方向误差。

此机床数据在以下功能中生效：

- 平滑：G642，G643，G644，G645

  说明

  在 G641 中，MD33100 不生效。此时通过 ADIS 或 ADISPOS 编写的到程序段过渡处的距离生效。

- 压缩器功能：COMP...
  MD33100 的值越大，表示有更多的短程序段可压缩成一条长程序段。

平滑模式

MD20480 $MC_SMOOTHING_MODE（G64x 平滑特性）

压缩器模式

MD20482 $MC_COMPRESSOR_MODE（压缩模式）

平滑 G645

MD33120 $MA_PATH_TRANS_POS_TOL（G645 平滑中的最大轮廓偏差）

在平滑相切但曲率非恒定的程序段过渡（例如圆弧 - 直线）时生效

设定数据

通道专用轮廓公差

SD42465 $SC_SMOOTH_CONTUR_TOL（最大轮廓偏差）

通道专用定向公差

SD42466 $SC_SMOOTH_ORI_TOL（刀具定向最大角度偏差）

通过 OST 平滑时的通道专用定向公差

SD42676 $SC_ORI_SMOOTH_TOL（平滑时的定向平滑公差）

通过 ORISON 进行定向平滑时的通道专用定向公差

SD42678 $SC_ORISON_TOL（定向平滑公差）

6.6.3 编程

6.6.3.1 轮廓公差/定向公差的编程 (CTOL、OTOL、ATOL)

使用地址 CTOL、OTOL 和 ATOL，可以调整通过机床数据和设定数据设置的加工公差，以便用于零件程序中的压缩器功能、精磨和定向平滑。

这些编程的公差值会持续生效，直至被新的编程值取代，或由于分配了一个负值而被删除。此外，在编程结束或复位时也会被删除。删除后，编程设置的公差值会再次生效。

句法

```
CTOL=<Value>
OTOL=<Value>
ATOL[<Axis>]=<Value>
```

含义

CTOL:	用于编程**轮廓公差**的地址			
	应用范围：	• 所有的压缩器功能 • 所有的精磨方式，除了 G641 和 G644		
	预处理停止：	不选择		
	生效方式：	模态		
	<Value>:	轮廓公差值是长度数据。		
		类型：	REAL	
		单位：	英寸/毫米（根据当前单位系统的设置）	
		值域：	≥ 0:	公差值
			< 0:	清除编写的公差值 ⇒ 机床数据或设定数据中设置的公差值重新生效。
OTOL:	用于编程**定向公差**的地址			
	应用范围：	• 所有的压缩器功能 • 定向平滑 ORISON • 所有的精磨方式，除了 G641, G644 和 OSD		
	预处理停止：	不选择		
	生效方式：	模态		
	<Value>:	定向公差值是角度数据。		
		类型：	REAL	
		单位：	度	
		值域：	≥ 0:	公差值
			< 0:	清除编写的公差值 ⇒ 机床数据或设定数据中设置的公差值重新生效。

ATOL:	用于编程**轴专用公差**的地址			
	应用范围：	• 所有的压缩器功能 • 定向平滑 ORISON • 所有的精磨方式，除了 G641, G644 和 OSD		
	预处理停止：	不选择		
	生效方式：	模态		
	<Axis>:	编写的公差所生效于的通道轴的名称		
	<Value>:	取决于轴的类型（线性轴或回转轴），轴公差的值为长度数据或角度数据。		
		类型：	REAL	
		单位：	用于线性轴：	英寸/毫米（根据当前单位系统的设置）
			用于回转轴：	度
		值域：	≥ 0:	公差值
			< 0:	清除编写的公差值 ⇒ 机床数据或设定数据中设置的公差值重新生效。

说明

通过 CTOL 和 OTOL 编写的轴专用公差值较通过 ATOL 编写的轴专用公差值具有更高优先级。

说明

缩放框架

缩放框架对编程公差的影响和对轴位置的影响一样，即：相对公差保持不变。

示例

程序代码	注释
COMPCAD G645 G1 F10000	；激活压缩器功能 COMPCAD。
X...Y...Z...	；此处机床数据和设定数据生效。
X...Y...Z...	
X...Y...Z...	
CTOL=0.02	；从此处开始，0.02 毫米的轮廓公差生效。
X...Y...Z...	
X...Y...Z...	
X...Y...Z...	

程序代码	注释
ASCALE X0.25 Y0.25 Z0.25	；从此处开始，0.005 毫米的轮廓公差生效。
X...Y...Z...	
X...Y...Z...	
X...Y...Z...	
CTOL=-1	；从此处开始，机床数据和设定数据再次生效。
X...Y...Z...	
X...Y...Z...	
X...Y...Z...	

更多信息

系统变量

可通过以下系统变量读取当前生效的公差：

- 带预处理停止的读取（在零件程序和同步动作中）
  - $AC_CTOL
    处理当前主运行程序段时生效的通道专用轮廓公差。
    若无轮廓公差生效，$AC_CTOL 会返回将各几何轴公差的平方相加后求得的平方根值。
  - $AC_OTOL
    处理当前主运行程序段时生效的通道专用定向公差。
    若无定向公差生效，定向转换生效期间 $AC_OTOL 会返回由各定向轴公差的平方相加后求得的平方根值，否则返回“-1”值。
  - $AA_ATOL[<轴>]
    处理当前主运行程序段时生效的轴专用轮廓公差。
    如果轮廓公差生效，$AA_ATOL[<几何轴>] 会返回由该轮廓公差除以几何轴数量的平方根所得到的值。
    如果定向公差和定向转换生效，$AA_ATOL[<定向轴>] 会返回由定向公差除以定向轴数量的平方根所得到的值。

说明

若未编写公差值，那么 $A 变量将无法区分各功能的公差。

当机床数据和设定数据中确定了不同的公差值时，即压缩器功能、精磨和定向平滑的公差，会出现上述情况。此时系统变量会返回一个出现在当前生效功能中的最大值。例如，如果压缩器功能的定向公差为 0.1，而定向平滑 ORISON 的定向公差为 1°，那么 $AC_OTOL 会返回值“1”。如果关闭了定向平滑功能，$AC_OTOL 将返回值“0.1”。

- 无预处理停止的读取（仅在零件程序中）
  - $P_CTOL
    当前生效的通道专用轮廓公差。
  - $P_OTOL
    当前生效的通道专用定向公差。
  - $PA_ATOL
    当前生效的轴专用轮廓公差。

边界条件

通过 CTOL、OTOL 和 ATOL 编写的公差同样对间接关联的功能生效：

- 设定值计算中的切线误差限制
- 任意形状表面模式：基本功能

CTOL、OTOL 和 ATOL 的编程不影响以下平滑功能：

- OSD 定向精磨
  OSD 不使用公差，而是使用到程序段过渡处的间距。
- G644 精磨
  G644 不用于加工，而是用于优化换刀和其他无加工运动。
- G645 精磨
  G645 的特性和 G642 几乎一样，也使用编程公差。只有在曲率变化的相切程序段过渡中（比如：圆弧到直线的相切过渡），才使用机床数据 MD33120 $MA_PATH_TRANS_POS_TOL 的公差值。因为在这种条件下平滑距离也可以位于编程轮廓的外侧，而大多数应用不太允许。另外通常一个很小的固定设置的公差足以平衡曲率变化，编程员无需再加以考虑。

6.7 可编程轮廓精度

6.7.1 功能

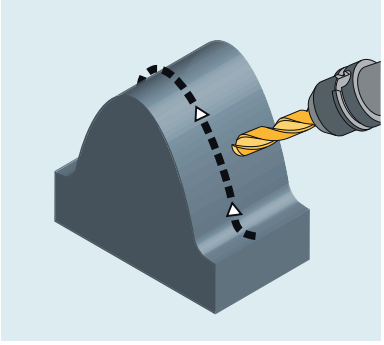


图 6-13 曲面轮廓上的刀具轨迹

在向曲面轮廓移动时，由于系统控制特性和急动滤波器的作用，实际轨迹可能会偏离设定轨迹，偏离程度受速度影响。

“可编程轮廓精度”功能可以根据曲率变化来降低轨迹速度，从而使轮廓误差保持在上限以内。该功能因此兼顾了加工精度和加工效率。

“预读 (页 521)”（LookAhead）功能可以保证在轨迹上的任何一点上轴都不会过快移动，超出指定轮廓精度允许的速度。

说明

“可编程轮廓精度”功能仅涉及轨迹的几何轴。此功能不会对定位轴的速度产生影响。

调试

该功能通过机床数据和设定数据设置。

更多信息：章节“参数设置 (页 565)”。

激活/禁用

该功能通过程序中的 G 指令激活/关闭。

更多信息：章节“编程 (页 569)”。

6.7.2 调试

6.7.2.1 参数设置

机床数据

设置功能和其他功能组合使用时的生效方式

设置功能和其他功能组合使用时的生效方式通过以下机床数据设置：

MD20470 $MC_CPREC_WITH_FFW（可编程轮廓精度）

值	含义
0	当前馈同时激活时，“可编程轮廓精度”功能失效。
1	当前馈同时激活时，“可编程轮廓精度”功能仍生效效。 当前馈激活时，前馈根据有效的 K_V 系数计算出轨迹速度的降低量。

值	含义
2	与 1 一样，只是该功能通过 MD32415 $MA_EQUIV_CPREC_TIME （可编程轮廓精度的时间常数）设置。 设定数据 SD42450 $SC_CONTPREC 确定了允许的最大轮廓误差（参见“参数设置”）。 急动滤波器会以合适的方式进入轮廓误差的计算。系统假定有一个合适的急动滤波器时间常数（MD32410 $MA_AX_JERK_TIME），该常数可以使采用前馈的受控系统产生一个忽略不计的微小轮廓误差。该时间常数要在机床数据 MD32415 $MA_EQUIV_CPREC_TIME 中设置（参见“参数设置”）。 系统会使用以下数值，来根据设置的急动度滤波器类型（MD32402 $MA_AX_JERK_MODE）计算出轮廓误差： • 在前馈激活时，这两个机床数据的差值：MD32410 $MA_AX_JERK_TIME - MD32415 $MA_$MA_EQUIV_CPREC_TIME • 当前馈没有激活时，单独的 MD32410 $MA_AX_JERK_TIME 的值 通过这种方法，调试人员便可以一开始获得准确、但可能偏刚性的控制效果，然后可以通过延长急动滤波器时间常数，来获得偏柔性、但损失了一定精度的控制效果。
3	与 2 一样，但是如果通过 CTOL 写入轮廓精度，该轮廓精度优先于 SD42450 $SC_CONTPREC
4	“可编程轮廓精度”功能的生效不受前馈、急动滤波器的影响。 系统在计算轮廓误差时只考虑 MD32415 $MA_EQUIV_CPREC_TIME 的值。所有对轮廓误差产生影响的时间常数要相加，作为总和输入到 MD32415 $MA_EQUIV_CPREC_TIME 中。
5	与 4 一样，但是如果通过 CTOL 写入轮廓精度，该轮廓精度优先于 SD42450 $SC_CONTPREC

说明

“可编程轮廓精度”功能不可以和“带阻型”急动滤波器（MD32402 $MA_AX_JERK_MODE = 3）组合使用。

说明

功能生效方式 MD20470 = 2 或 3 主要用于前馈。如果在关闭前馈时激活了上述其中一种功能生效方式，系统会加上从 K_V 系数得出的轮廓误差。此时，轨迹速度会显著下降。

说明

我们不再推荐使用功能生效方式 MD20470 = 0 或 1。这些方式只是为了和旧软件版本兼容而保留在系统中。

功能的初始设置

控制系统启动后、通道复位或程序结束复位后、程序启动后功能的初始设置通过以下机床数据确定：

MD20150 $MC_GCODE_RESET_VALUES[<n>] = <m>

其中：
<n> = G 指令组编号 - 1（此处：38)

<m> = G 指令组内指令的编号

轨迹速度限制方式

每种动态响应模式下如何根据曲率来限制几何轴的速度，可通过以下机床数据设置：

MD22450 $MC_DYN_LIM_MODE[<n>]（单轴速度限制或几何尺寸上的速度限制）

其中下标 <n> =
0 用于 DYNNORM
1 用于 DYNPOS
2 用于 DYNROUGH
3 用于 DYNSEMIFIN
4 用于 DYNFINISH
5 用于 DYNPREC

值	含义
0	根据参与轨迹的几何轴的加速度来限制速度。
1	根据参与轨迹的几何轴的几何曲率来限制速度

设置为 0 会导致轨迹速度过高，因为在对角线方向上可能会移动更快。

设置为 1 会在对角线切向和轴向切向上以相同方式限制轨迹速度。轨迹速度变化会因此更加平缓。因此，MD22450[4] = 1 和 MD22450[5] = 1 是针对精加工和精修整的推荐设置。

程序写入的轮廓精度的时间常数

可编程轮廓精度的时间常数在以下机床数据中输入：

MD32415 $MA_EQUIV_CPREC_TIME（可编程轮廓精度的时间常数）

要根据 MD20470 中设置的功能生效方式来输入合适值：

- MD20470 = 0/1　MD32415 不相关。
- MD20470 = 2/3　MD32415 中要输入一个合适的急动滤波器时间常数（MD32410 $MA_AX_JERK_TIME），当前馈激活时，在该时间常数下轮廓误差非常细微，可以忽略不计。
- MD20470 = 4/5　MD32415 中要输入所有会影响轮廓误差的时间常数的总和。

为 FIR 滤波器的特性曲线拟合预留的存储空间

需要组合使用“可编程轮廓精度”和“FIR 低通”型急动滤波器（MD32402 $MA_AX_JERK_MODE = 5）时，要设置以下用于配置存储空间的机床数据：

MD38020 $MA_MM_CPREC_FIR_POINTS（CPRECON 用于拟合 CPRECON 的 FIR 滤波器特性曲线的点数）

MD38020 确定了“可编程轮廓精度”功能为拟合 FIR 滤波器的反时限特性曲线而使用的表格值的数量。该值越大，拟合的精度也就越高。

建议设置 = 100

说明

如果没有预留专门的存储空间，即 MD38020 = 0，系统便无法执行该功能，并输出报警 10990。

设定数据

轮廓精度

如何确定曲面轮廓的几何轴轨迹上的最大轮廓误差：

- 当 MD20470 $MC_CPREC_WITH_FFW = 2/4 时，针对每种动态响应模式确定的设定数据决定了最大轮廓误差：

  SD42450 $SC_CONTPREC[<n>]（轮廓精度）

  其中下标 <n> = 0　用于 DYNNORM
  1　用于 DYNPOS
  2　用于 DYNROUGH
  3　用于 DYNSEMIFIN
  4　用于 DYNFINISH
  5　用于 DYNPREC

- 当 MD20470 $MC_CPREC_WITH_FFW = 3/5 时，由程序中用 CTOL 写入的轮廓误差决定了最大轮廓误差。
  程序中没有写入 CTOL 时，SD42450 $SC_CONTPREC[<n>] 生效，即使 MD20470 $MC_CPREC_WITH_FFW = 3/5。

此数值和几何轴的 K_V 系数越小，曲面轮廓上的轨迹进给率的下降幅度也就越大。

快速移动时的轮廓精度

在加工有曲面轮廓的工件时，如果激活了“可编程轮廓精度”功能，为了符合规定的轮廓精度，刀具在快速移动时的轨迹速度也会下降，例如：在绕行拐角时和绕行工件的精磨程序段中。为了尽量减轻快速移动时轨迹速度的下降幅度，可为“可编程轮廓精度”功能设置一个与正常工件加工不同的“快移轮廓精度”：

SD42451 $SC_CONTPREC_G00_ABS（快移时的轮廓精度）

设置 SD42451 = 0 时，$SC_CONTPREC[DYNNORM] 中设置的轮廓精度会在快移时生效。

最小轨迹进给

通过以下设定数据，可为“可编程轮廓精度”功能设定一个最小轨迹进给率：

SD42460 $SC_MINFEED（CPRECON 时的最小轨迹进给）

当程序内写入的 F 值低于该值或轴的动态响应能力强制要求采取更低的轨迹速度时，系统才会将进给率下调到该限值以下。

更多信息

关于 MD32402 $MA_AX_JERK_TIME（单轴急动滤波器类型）和 MD32410 $MA_AX_JERK_TIME（单轴急动滤波器的时间常数）的说明：

→ 功能手册之进给轴和主轴

关于 CTOL 的说明：

→ 章节“轮廓公差/定向公差的编程 (CTOL、OTOL、ATOL) (页 559)”

6.7.3 编程

6.7.3.1 激活/关闭可编程轮廓精度 (CPRECON, CPRECOF)

“可编程轮廓精度”功能通过自动调整速度减小弯曲轮廓上的轨迹误差。

该功能可在 NC 程序中使用以模态方式生效的指令“G 功能组 39”（可编程轮廓精度）来激活或关闭。

句法

```
CPRECON
...
CPRECOF
```

含义

CPRECON：	**激活**“可编程轮廓精度”功能
CPRECOF：	**关闭**“可编程轮廓精度”功能

示例

程序代码	注释
N10 G0 X0 Y0	
N20 CPRECON	；激活“可编程轮廓精度”。
N30 G1 G64 X100 F10000	；在连续路径运行中以 10 m/min 的速度加工。
N40 G3 Y20 J10	；在圆弧程序段中自动的进给限制。
N50 G1 X0	；无限制进给率（10 m/min）。
...	
N100 CPRECOF	；关闭“可编程轮廓精度”。
N110 G0 ...	

6.8 快速移动

6.8.1 功能

6.8.1.1 快速移动

在快速移动中，已编程的刀具以尽可能大的运行速度运行。

每个轴的快速运行速度都是单独定义的（参见“参数设置 (页 574)”）。

应用

快速移动可用于以下任务：

- 刀具快速定位
- 工件绕行
- 逼近换刀点
- 退刀

说明

快速移动不适用于工件加工！

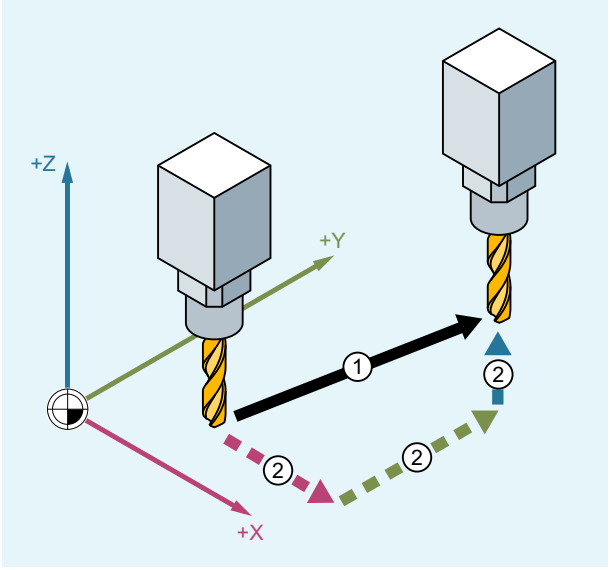
激活

通过 G0 编程在零件程序中激活快速移动（参见“编程 (页 575)”）。

6.8.1.2 快速移动下轨迹轴的插补特性

线性/非线性插补

快速移动中可选择通过线性插补或非线性插补运行轨迹轴。



① 快速移动中通过线性插补的轨迹

② 快速移动中通过非线性插补的单轴运动

线性插补

特性：

- 轨迹轴一同进行插补。
- 使用 G0 编写的刀具运动会以最快的运行速度执行（快速移动）。
- 每个轴的快速移动速度都是单独定义的。
- 如果同时在多根轴上执行快速移动，那么快速移动速度由所需时间最长的轴来决定。

以下情形中总是执行线性插补：

- 在采用包含 G0 的、不支持定位轴运动的 G 指令组合时，例如：
  G40、G41、G42、G96、G961 和 MD20750 $MC_ALLOW_G0_IN_G96 == FALSE
- 在 G0 和 G64 组合的情况下
- 压缩器生效，或转换生效时
- 点对点（PTP）运行中
- 选择了轮廓手轮时（FD=0）
- 包含几何轴旋转的框架生效时
- 几何轴步冲生效时

非线性插补

特性：

- 每根轨迹轴作为单轴（定位轴）独立于其他轴通过轴特定的快速移动速度插补：
- 通过 PLC 和同步动作对所有编写作为轨迹轴的定位轴启用通道特定的剩余行程删除。

在非线性插补时基于轴专用加加速度，可适用以下两种设置之一：

- 定位轴指令 BRISKA, SOFTA, DRIVEA
- 机床数据：
  - MD32420 $MA_JOG_AND_POS_JERK_ENABLE
  - MD32430 $MA_JOG_AND_POS_MAX_JERK

不支持基于轨迹剩余行程的系统变量（$AC_PATH、$AC_PLTBB 和 $AC_PLTEB）。

⚠ 小心
碰撞危险 由于非线性插补时的刀具运行与线性插补时的刀具运行不同，因此基于轨迹运行坐标的同步有时可能无效。

选择插补类型

快速移动时应生效的插补类型通过机床数据预设（参见“参数设置 (页 574)”）。

此外还可在零件程序中设置所需的插补特性，且不考虑预设（参见“激活/取消快速移动的线性插补（RTLION, RTLIOF） (页 577)”）。

6.8.1.3 快速移动中的公差

可通过 G0 公差设置一个不同于工件加工公差的快速移动公差。

优点

G0 公差越大，G0 程序段处理速度也就越快。

前提条件

仅在满足下列条件时，G0 公差才会生效：

- 下列功能中有一个生效：
  - 压缩器功能 COMP...
  - 平滑功能 G642 或 G645
  - 定向精磨 OST
  - 定向平滑 ORISON
  - 路径相关的定向平滑 ORIPATH
- 零件程序中有多个（≥ 2）相连的 G0 程序段。
  在只有一个 G0 程序段时 G0 公差不会生效，因为在从非 G0 运动过渡至 G0 运动（以及反之）时，原则上是“**较小的公差**”（工件加工公差）生效！

设置 G0 公差

G0 公差可针对不同通道通过机床数据预设（参见“参数设置 (页 574)”）。

预设的 G0 公差可以在程序中通过指令暂时更改（参见“调整快速移动的公差 (STOLF, CTOLG0, OTOLG0) (页 579)”）。

6.8.1.4 快速移动倍率

操作人员可使用机床控制面板上的快速移动倍率开关在设备现场降低快速移动速度，调整将立即生效。

激活的快速移动倍率对当前指定给通道的、且启用线性插补或非线性插补运行的所有轨迹轴生效。

更多信息：功能手册之进给轴和主轴

6.8.2 调试

6.8.2.1 参数设置

快速移动速度

快速移动速度是通过以下机床数据为各个轴确定的最大允许的轴速度：

MD32000 $MA_MAX_AX_VELO（最大轴速度）

快速移动时的插补特性

快速移动时的插补特性通过以下机床数据针对各个通道设置：

MD20730 $MC_G0_LINEAR_MODE = <值>（G0 下的插补特性）

<值>	含义
0	快速移动（G0）中**非线性**插补生效。 轨迹轴作为定位轴运行。
1	快速移动（G0）中**线性**插补生效。 轨迹轴一同进行插补。

G0 公差

相对 G0 公差

该公差是一个针对不同通道、基于工件加工公差的相对值：

MD20560 $MC_G0_TOLERANCE_FACTOR（G0 公差系数）

公差系数可大于也可小于 1.0。系数等于 1.0（缺省值）时，快速移动启用和非快速移动相同的公差。通常情况下公差系数设为大于 1.0 的值。

绝对 G0 公差

该公差也可是一个针对不同通道的、独立于轮廓公差和刀具方向公差的绝对值：

- MD20561 $MC_G0_TOLERANCE_CTOL_ABS（G0 运动中的轮廓公差）= <值>

<值>	含义
= 0（缺省值）	G0 运动中的轮廓误差由“相对 G0 公差”确定。
> 0	G0 运动中的轮廓误差由 MD20651 确定。

- MD20562 $MC_G0_TOLERANCE_OTOL_ABS（G0 运动中的刀具方向公差）=<值>

<值>	含义
= 0（缺省值）	G0 运动中的刀具方向误差由“相对 G0 公差”确定。
> 0	G0 运动中的刀具方向误差由 MD20652 确定。

6.8.3 编程

6.8.3.1 激活快速移动（G0）

通过 G 指令 G0 激活以快速移动速度运行的轨迹轴。

句法

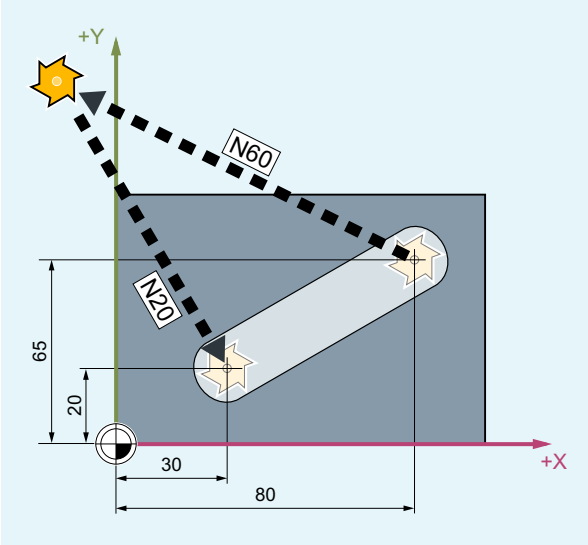
```
G0 X… Y… Z…
G0 RP=… AP=…
```

含义

G0:	以快速运行速度运行轴	
	生效方式：	模态
X...Y...Z...:	在直角坐标系中指定终点	
RP=...AP=…:	在极坐标中指定终点	

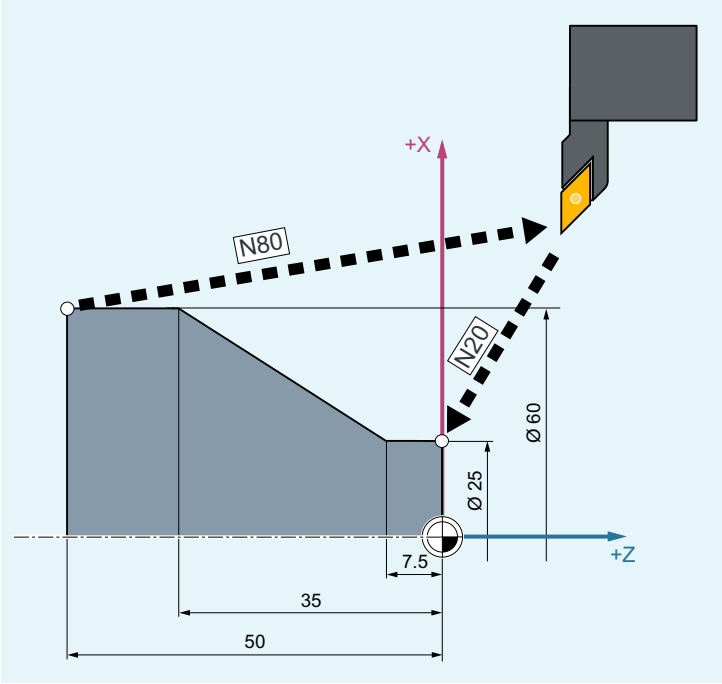
示例

示例 1：铣削



程序代码	注释
N10 G90 S400 M3	；绝对尺寸，主轴顺时针
N20 G0 X30 Y20 Z2	；回到起始位置
N30 G1 Z-5 F1000	；刀具横向进给
N40 X80 Y65	；直线运行
N50 G0 Z2	
N60 G0 X-20 Y100 Z100 M30	；退刀，程序结束

示例 2：车削



程序代码	注释
N10 G90 S400 M3	；绝对尺寸，主轴顺时针
N20 G0 X25 Z5	；回到起始位置
N30 G1 G94 Z0 F1000	；刀具横向进给
N40 G95 Z-7.5 F0.2	
N50 X60 Z-35	；直线运行
N60 Z-50	
N70 G0 X62	
N80 G0 X80 Z20 M30	；退刀，程序结束

6.8.3.2 激活/取消快速移动的线性插补（RTLION, RTLIOF）

快速移动时，可通过 G 功能组 55 的指令或在零件程序中设置插补特性，而与预设置无关（MD20730 $MC_G0_LINEAR_MODE）。

句法

```
RTLIOF
...
RTLION
```

含义

RTLIOF:	取消线性插补的 G 指令 ⇒ 快速移动（G0）中**非线性**插补生效。所有轨迹轴相互独立地运行至它们的终点。	
	生效方式：	模态
RTLION:	激活线性插补的 G 指令 ⇒ 快速移动（G0）中**线性**插补生效。所有轨迹轴同时运行至它们的终点。	
	生效方式：	模态

说明

RTLIOF 的前提条件

如要在 RTLIOF 时进行**非线性**插补，为此必须满足下列前提条件：

- 无转换生效 (TRAORI, TRANSMIT 等)。
- G60 生效 (在程序段末尾停止)。
- 无压缩器生效 (COMPOF)。
- 无刀具半径补偿生效 (G40)。
- 未选择轮廓手轮。
- 无步冲生效。

如果其中一个前提条件不满足，就会与 RTLION 一样进行直线插补。

示例

程序代码	注释
	; 线性插补已预设：
	; MD20730 $MC_GO_LINEAR_MODE == TRUE
...	
N30 **RTLIOF**	; 线性插补已取消。
N40 G0 X0 Y10	; G0 程序段以非线性插补运行。
N50 G41 X20 Y20	; 刀具半径补偿生效⇒ G0 程序段以线性插补运行。
N60 G40 X30 Y30	; 刀具半径补偿未生效⇒ G0 程序段以非线性插补运行。
N70 **RTLION**	; 线性插补激活。
...	

其它信息

读取当前插补特性

通过系统变量 $AA_G0MODE 可以读取当前的插补特性：

6.8.3.3 调整快速移动的公差 (STOLF, CTOLG0, OTOLG0)

通过机床数据设置的快速移动公差（G0 公差）可在零件程序中进行临时调整。机床数据中的设置此时不会被更改。在通道复位或程序结束复位后，所设置的公差会重新生效。

前提条件

仅在满足下列条件时，G0 公差才会生效：

- 下列功能中有一个生效：
  - 压缩器功能 COMP...
  - 平滑功能 G642 或 G645
  - 定向精磨 OST
  - 定向平滑 ORISON
  - 路径相关的定向平滑 ORIPATH
- 零件程序中有多个（≥ 2）相连的 G0 程序段。
  在只有一个 G0 程序段时 G0 公差不会生效，因为在从非 G0 运动过渡至 G0 运动（以及反之）时，原则上是“**较小的公差**”（工件加工公差）生效！

句法

调整 G0 相对公差

```
STOLF=<值>
```

调整 G0 绝对公差

```
CTOLG0=<值>
```

```
OTOLG0=<值>
```

含义

STOLF:	用于编写临时生效的快速移动公差系数的地址			
	<Value>:	G0 公差系数		
		类型：	REAL	
		值：	≥ 0:	G0 公差系数可大于也可小于 1.0。系数等于 1.0（缺省值）时，快速移动启用和非快速移动相同的公差。通常情况下公差系数设为大于 1.0 的值。 编写的 G0 公差系数一直生效，直至其再一次被 STOLF 指令覆盖或被 CTOLG0/OTOLG0 指令替换，或者因通道复位或程序结束复位而被清除。
			< 0:	清除编写的公差系数 ⇒ 机床数据中预设的公差系数重新生效。
CTOLG0:	用于编写临时生效的快速移动轮廓公差的地址			
	<值>:	轮廓公差绝对值		
		类型：	REAL	
		值：	≥ 0:	编写的轮廓公差绝对值一直生效，直至其再一次被 CTOLG0 指令覆盖或被 STOLF 指令替换，或者因通道复位或程序结束复位而被清除。
			< 0:	清除编写的公差值 ⇒ 机床数据中预设的公差值重新生效。
OTOLG0:	用于编写临时生效的快速移动定向公差的地址			
	<值>:	定向公差绝对值		
		类型：	REAL	
		值：	≥ 0:	编写的定向公差绝对值一直生效，直至其再一次被 OTOLG0 指令覆盖或被 STOLF 指令替换，或者因通道复位或程序结束复位而被清除。
			< 0:	清除编写的公差值 ⇒ 机床数据中预设的公差值重新生效。

说明

最后编写的地址的优先级始终最高，如以下示例所示：

- 当编写了 STOLF 时又编写 CTOLG0 时，则会为轮廓平滑使用由 CTOLG0 设置的公差值。
- 同样，当编写了 STOLF 时又编写 OTOLG0 时，则会为定向精磨使用由 OTOLG0 设置的公差值。
- 在重新编写 STOLF 指令后，又会再次使用轮廓公差和定向公差的公差系数。

示例

示例 1：调整 G0 相对公差

程序代码	注释
COMPCAD G645 G1 F10000	；压缩器功能 COMPCAD
X...Y...Z...	；此处机床数据和设定数据生效。
X...Y...Z...	
X...Y...Z...	
G0 X...Y...Z...	
G0 X...Y...Z...	；此处机床数据 $MC_G0_TOLERANCE_FACTOR（例如 =3）生效， 即 $MC_G0_TOLERANCE_FACTOR * $MA_COMPRESS_POS_TOL 的平滑公差生效。
CTOL=0.02	
STOLF=4	
G1 X...Y...Z...	；从此处开始，0.02 毫米的轮廓公差生效。
X...Y...Z...	
X...Y...Z...	
G0 X...Y...Z...	
X...Y...Z...	；从此处开始 G0 公差系数 4 生效，即 0.08 mm 的轮廓公差生效。
...	

示例 2：调整 G0 绝对公差

在机床数据中应预设置以下 G0 绝对公差：

- G0 轮廓公差：0.1
- G0 定向公差：1.0

在零件程序中应对这些公差进行临时调整：

程序代码	注释
COMPCAD G645 G1 F10000	；压缩器功能 COMPCAD
X...Y...Z...	；从此处开始，所设置的工件加工公差生效。
X...Y...Z...	
X...Y...Z...	
G0 X...Y...Z...	
G0 X...Y...Z...	；此处所设置的 G0 绝对公差生效。
CTOLG0=0.2 OTOLG0=2.0	；G0 绝对公差的编程。
G1 X...Y...Z...	
X...Y...Z...	
X...Y...Z...	
G0 X...Y...Z...	
X...Y...Z...	；从此处开始，所编写的 G0 公差生效。
...	

更多信息

读取 G0 公差系数

当前生效的快速移动公差系数可通过系统变量读取：

- 在同步动作或在带预处理停止的零件程序中，通过系统变量：
  $AC_STOLF　　生效的 G0 公差系数
  当前主程序段预处理时生效的 G0 公差系数。

- 在不带预处理停止的零件程序中，通过系统变量：
  $P_STOLF　　编程的 G0 公差系数

如果在生效的零件程序中未使用 STOLF 赋值，则两个系统变量会输出在机床数据中配置的值。

如果在程序段中无快速移动（G0），则这些系统变量总是输出值 1。

读取绝对 G0 公差

当前生效的快速移动绝对公差可通过系统变量读取：

- 在同步中或带预处理停止的零件程序中，通过系统变量：
  $AC_CTOL_G0_ABS　　G0 运动中生效的轮廓公差
  G0 轮廓公差，在处理当前主运行程序段时生效。
  $AC_OTOL_ G0_ABS　　G0 运动中生效的定向公差
  G0 定向公差，在处理当前主运行程序段时生效。

- 在不带预处理停止的零件程序中，通过系统变量：
  $P_CTOL_ G0_ABS　　G0 运动中编写的轮廓公差
  $P_OTOL_ G0_ABS　　G0 运动中编写的定向公差

如果在生效的零件程序中未使用 CTOLG0 和 OTOLG0 编写 G0 绝对公差，则这些系统变量会提供机床数据中所设置的值。

6.9 “低动态响应”模式（选件）

说明

“低动态响应”模式是一个选件，需要购买许可证。

订货号：6FC5800-0BS65-0YB0

6.9.1 功能

功能

当机床经常在最大功率附近工作时，用户可以暂时使机床进入“低动态响应”模式，以保护机床的电机和其他组件。激活该模式后，各个轴的动态响应参数和当前程序中的轨迹进给率会按照预设值下降，最大下降幅度为一半。轨迹轴因此减慢移动速度，保护机床的机械结构。

该功能只对轨迹运动有影响。以下运动不受该功能的影响：

- 主轴旋转（例外：插补车削参见章节“边界条件 (页 585)”
- 单轴运动 POS, POSA），即使该旋转是从同步动作触发的
- 碰撞监控激活时的制动运动
  程序段预读期间会考虑该功能。在轴停止时，会按照原有动态响应能力制动。
- JOG 方式下的移动运动

参数设置

“低动态响应”模式中，下降后的动态响应参数生效。此时系统会在原动态响应参数上乘以一个比例系数，使动态响应下降。该比例系数是通过该功能专用的机床数据设置的。

更多信息：→“参数设置 (页 584)”一章

激活/取消

激活

PLC 置位以下 NC/PLC 接口信号，便可以激活“低动态响应”模式：

`<Nc>.basic.out.enableReducedDynamics`（激活“低动态响应”模式）

在预处理期间，系统会针对每条通道单独计算并处理该信号。也就是说，只有在对应的程序段进入主处理后，该功能才生效。这也意味着，通过接口信号激活该功能到轨迹运动采用降低后的动态响应值之间有一段延时。

此功能一旦生效，NC 便会通过以下 NC/PLC 接口信号确认激活：

`<Chan>.basic.in.reducedDynamicsActive`（“低动态响应”模式生效）

禁用

PLC 复位 NC/PLC 接口信号后，便可以关闭“低动态响应”模式。

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<Nc>.basic.out.enableReducedDynamics	LBP_NC.A_EnableReducedDynamics	DB10.DBX57.6

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.reducedDynamicsActive	LBP_Chan*.E_ReducedDynamicsActive	DB21, ...DBX39.3

6.9.2 参数设置

通道专用机床数据

“低动态响应”模式中轨迹进给率的系数

MD20740 $MC_FEED_FACT_REDUCED_DYN

该机床数据可以确定一个系数，在“低动态响应”模式中，该系数和当前程序中的轨迹进给率相乘。

MD20740 的取值范围：0.5 ... 1.0

轴专用机床数据

“低动态响应”模式中最大轴速度的系数

MD32311 $MA_VEL_FACT_REDUCED_DYN

该机床数据可以确定一个系数，在“低动态响应”模式中，该系数和轨迹插补轴的最大轴速度（MD32000 $MA_MAX_AX_VELO）相乘。

因此，在“低动态响应”模式中，轨迹插补轴的最大轴速度（$VELO_{max(RD)}$）如下计算得出：

$$VELO_{max(RD)} = MD32000 * MD32311$$

MD32311 的取值范围：0.5 ... 1.0

“低动态响应”模式中最大轴加速度的系数

MD32312 $MA_ACC_FACT_REDUCED_DYN

该机床数据可以确定一个系数，在“低动态响应”模式中，该系数和轨迹插补轴的最大轴加速度（MD32300 $MA_MAX_AX_ACCEL）相乘。

因此，在“低动态响应”模式中，轨迹插补轴的最大轴速度（$ACC_{max(RD)}$）如下计算得出：

$$ACC_{max(RD)} = MD32300 * MD32312$$

MD32312 的取值范围：0.5 ... 1.0

“低动态响应”模式中最大轴急动度的系数

MD32313 $MA_JERK_FACT_REDUCED_DYN

该机床数据可以确定一个系数，在“低动态响应”模式中，该系数和轨迹插补轴的最大轴急动度（MD32431 $MA_MAX_AX_JERK）相乘。

因此，在“低动态响应”模式中，轨迹插补轴的最大轴加速度（$JERK_{max(RD)}$）如下计算得出：

$$JERK_{max(RD)} = MD32431 * MD32313$$

MD32313 的取值范围：0.5 ... 1.0

6.9.3 边界条件

插补车削

在插补车削中，直线轴（x, y, z）作圆弧运动，相当于车削加工中的主轴旋转运动。但和常规主轴旋转运动不同的是，在插补车削的圆弧运动的“低动态响应”模式中，每次总是下调后的动态响应值生效。

6.10 复位特性

MD20150

复位（通道复位或 BAG 复位）时，所有 G 功能组启用通道专用初始设置：

MD20150 $MC_GCODE_RESET_VALUES（G 功能组的初始设置）

和“连续路径运行、准停、预读”相关的 G 功能组有：

- 组 10：准停－连续路径模式
- 组 12：准停时的程序段切换条件
- 组 21：加速模式
- 组 30：数控程序段压缩器
- 组 59：轨迹插补的动态响应模式

有关初始设置的详细说明请参见章节“BAG、通道、程序运行、复位特性 (页 29)”。

6.11 前提条件

6.11.1 程序段切换和定位轴

在一个零件程序中轨迹轴处于连续路径运行时，同时运行的定位轴可能会影响轨迹轴的特性和程序段切换。

有关定位轴的详细说明请见:
更多信息
功能手册之进给轴和主轴；定位轴

6.11.2 程序段切换延时

即使所有在零件程序段中运行的轨迹轴和辅助轴都满足各自特殊的程序段切换标准，其他条件不满足和/或生效的功能也可能导致程序段切换延时。

示例：

- 没有通过 PLC 应答辅助功能
- 后续程序段不存在
- 功能“清空缓冲存储器”生效

影响

如果在连续路径运行期间无法执行程序段切换，那么所有在该零件程序段中编程的轴（除了跨程序段运行的辅助轴）都会停止运行。此时不会出现轮廓误差。

在加工期间停止轨迹轴运行可导致工件表面上出现切削痕迹。

跨通道程序协调和逐通道试运行 7

7.1 跨通道程序协调

7.1.1 跨通道程序协调（INIT，START，WAITM，WAITMC，WAITE，SETM，CLEARM）

一个 NC 通道原则上能够独立于同一运行方式组（BAG）中之其他通道执行在这个通道中启动的程序。但若一个 BAG 的多个通道中的多个程序同时参与工件加工，则必须在不同的通道中通过以下协调指令实现程序过程的协调。

前提条件

参与程序协调的所有通道必须属于**同一**运行方式组（BAG）。

MD10010 $MC_ASSIGN_CHAN_TO_MODE_GROUP[<通道>] = <BAG 编号>

通道名称替代通道编号

亦可用在 MD20000 $MC_CHAN_NAME[<通道索引>] 中记录的通道名称来替代通道编号，作为程序协调的预定义程序的参数。必须首先使能“在 NC 程序中使用通道编号”：

MD10280 $MN_PROG_FUNCTION_MASK，位 1 = TRUE

说明

指令之间的最短间距

指令 WAITMC 和以下指令必须至少间隔两条运动程序段：INIT、START、WAITE、WAITM、SETM、CLEARM。WAITMC 是一条可执行的程序段，系统会将它挪到上一条程序段中以便优化执行，并随后将它作为程序段删除。而 SETM 是一条不可执行的程序段，系统会将它挪到下一条程序段中，当两者之间只间隔一条程序段时，两个指令便位于中间的这条程序段中。因为只有一条程序段，当间隔一条程序段时便不会为 WAITMC 进行优化。

程序执行因此中断，加工短暂停止。

句法

```
INIT(<ChanNr>, <Prog>, <AckMode>)
START(<ChanNr>, <ChanNr>, ...)
WAITM(<MarkNr>, <ChanNr>, <ChanNr>, ...)
WAITE(<ChanNr>, <ChanNr>, ...)
WAITMC(<MarkNr>, <ChanNr>, <ChanNr>, ...)
SETM(<MarkNr>, <MarkNr>, ...)
```

```
CLEARM(<MarkNr>, <MarkNr>, ...)
```

含义

INIT():	预定义步骤，用于选择需要在所给定通道中执行的 NC 程序	
START():	预定义步骤，用于启动在各个通道中选择的程序	
WAITM():	预定义步骤，用于等待到达给定通道中的等待标记 在自身通道中会通过 WAITM 设置给定的等待标记。前一个程序段通过准停结束。等待标记会在同步后被清除。 每个通道最多可同时设置 10 个标记。	
WAITE():	预定义步骤，用于等待一个或多个其它通道中的程序结束。	
WAITMC(): [1)]	预定义步骤，用于等待到达给定通道中的等待标记 与 WAITM 的不同之处在于，在其它通道尚未到达等待标记的情况下，才会将轴制动至准停状态。	
SETM(): [1)]	预定义步骤，用于针对通道协调设置一个或多个等待标记 自身通道中的程序执行不受此影响。 SETM 在通道复位和 NC 启动后仍保持其有效性。	
CLEARM(): [1)]	预定义步骤，用于清除通道协调的一个或多个等待标记 自身通道中的程序执行不受此影响。 CLEARM() 清除通道中的所有等待标记。 CLEARM(0) 仅清除等待标记“0”。 CLEARM 在通道复位和 NC 启动后仍保持其有效性。	
<ChanNr>:	通道编号 不必给定自身通道的编号。	
	类型：	INT
<Prog>:	绝对或相对路径说明（**可选**）+ 程序名称	
	类型：	STRING
	有关路径指定的相关信息参见 NC 编程手册，章节“程序存储器文件的定址”。	

<AckMode>:	应答模式（**可选**）		
	类型:	CHAR	
	数值:	"N"	无应答 指令发送后，程序将继续执行。若无法成功执行指令，系统不会通知发送者。
		"S"	同步应答 在接收组件对指令进行应答前，程序将暂停执行。应答为正时会执行下一个指令。应答为负时会输出故障信息。
<MarkNr>:	等待标记编号 **提示** 在一个多通道系统中最多可设置 100 个等待标记（等待标记 0 ... 99）。 在单通道系统中则只有等待标记 0 可供使用。		
1) 为进行用户专用通讯和/或通道协调，借助 SETM / CLEARM 可在不使用有条件等待指令 WAITMC 的情况下使用等待标记。此时，等待标记保留通道复位和 NC 启动及其值。			

示例

通过 MD20000 中的通道名称启动

- 参数设置

```
MD10280 $MN_PROG_FUNCTION_MASK，位 1 = TRUE
$MC_CHAN_NAME[ 0 ] = "BEARBEITUNG"  ；通道 1 的名称
$MC_CHAN_NAME[ 1 ] = "ZUFUEHRUNG"   ；通道 2 的名称
```

- 编程

程序代码	注释
START(BEARBEITUNG)	；通道 1 启动
START(ZUFUEHRUNG)	；通道 2 启动

通过本地“通道名称”和用户变量启动

程序代码	注释
DEF INT MASCHINE = 1	；定义通道 1 的用户变量
DEF INT LADER = 2	；定义通道 2 的用户变量
...	
START(MASCHINE)	；通道 1 启动
START(LADER)	；通道 2 启动

通过本地“通道名称”、用户变量和设置的通道名称启动

程序代码	注释
DEF INT chanNo1	; 定义通道 1 的用户变量
DEF INT chanNo2	; 定义通道 2 的用户变量
chanNo1 = CHAN_1	; 将设置的通道名称分配给通道 1
chanNo2 = CHAN_2	; 将设置的通道名称分配给通道 2
...	
START(chanNo1)	; 通道 1 启动
START(chanNo2)	; 通道 2 启动

带绝对路径说明的 INIT 指令

在通道 2 中选择程序 /_N_MPF_DIR/_N_ABSPAN1_MPF。

程序代码

```
INIT(2,"/_N_WKS_DIR/_N_WELLE1_WPD/_N_ABSPAN1_MPF")
```

带程序名的 INIT 指令

通过名称“MYPROG”选择程序。控制系统通过搜索路径搜索程序。

程序代码

```
INIT(2,"MYPROG")
```

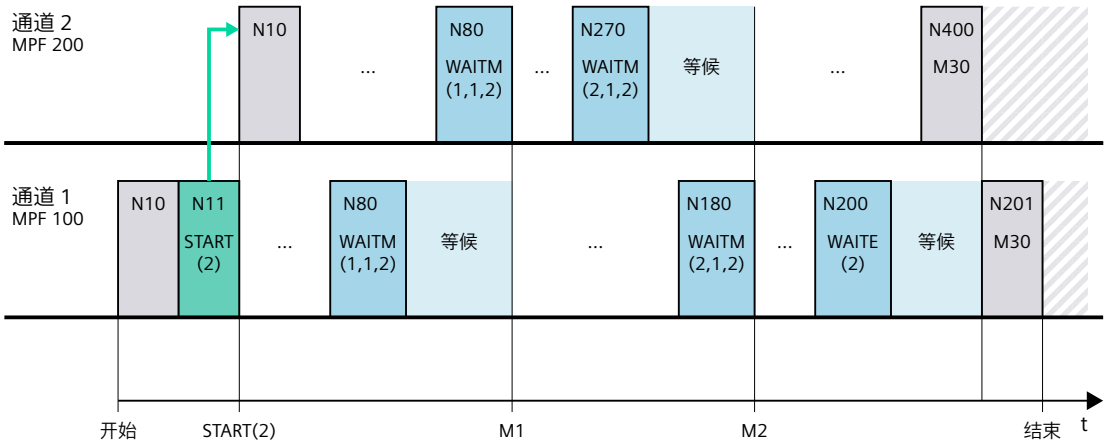
带 WAITM 的程序协调

- 通道 1：已选择并启动程序 /_N_MPF_DIR/_N_MPF100_MPF。

程序代码	注释
	; 程序 MPF100
N10 INIT(2,"MPF200","N")	; 选择程序 MPF200，通道 2
N11 START(2)	; 通道 2 启动
...	
N80 WAITM(1,1,2)	; 等待通道 1 和 2 中的 WAIT 标记 1
N81 ...	; 通道 1，N81 和通道 2，N71 ; 同步开始
...	
N180 WAITM(2,1,2)	; 等待通道 1 和 2 中的 WAIT 标记 2
N181 ...	; 通道 1，N181 和通道 2，N271 ; 同步开始
...	
N200 WAITE(2)	; 等待通道 2 中的程序结束
N201 ...	; N201 在 ; 通道 2 中的 MPF200 程序结束后才开始
N201 M30	; 通道 1 中的程序结束

- 通道 2：在通道 1 中通过程序段 N10 和 N20 选择并启动通道 2 中的程序 MPF200_MPF。

程序代码	注释
;$PATH=/_N_MPF_DIR	; 程序 MPF200
...	
N70 WAITM(1,1,2)	等待通道 1 和 2 中的 WAIT 标记 1
N71 ...	; 通道 1，N81 和通道 2，N71 ; 同步开始
...	
N270 WAITM(2,1,2)	等待通道 1 和 2 中的 WAIT 标记 2
N271 ...	; 通道 1，N181 和通道 2，N271 ; 同步开始
...	
N400 M30	通道 2 中的程序结束



边界条件

在 WAIT 标记后非同步开始执行后续程序段

在通道协调时，通过 WAIT 标记可导致非同步开始执行后续程序段。在待同步之通道中的一个中到达共同的 WAIT 标记前，触发在此通道中导致带有隐性重新定位（REPOSA）的剩余行程删除的动作的情况下，便会出现此特性。

假设：通道 1 和 2 中的当前轴分配

- 通道 1：轴 X1 和 U
- 通道 2：轴 X2

表格 7-1　通道 1 和 2 中的时间顺序

通道 1	通道 2	说明
. . .	. . .	通道 1 和 2 中的任意执行
N100 WAITM(20,1,2)		通道 1：到达 WAIT 标记并等待与通道 2 同步
开始执行 GETD(U)： • *跨通道取轴* • *剩余行程删除* • *REPOSA* *结束*	N200 GETD(U)	通道 2：请求通道 1 中的轴 U 通道 1：在后台执行 GET(U)
	N210 WAITM(20,1,2)	通道 2：到达 WAIT 标记。⇒ 为此，通道 1 和 2 的同步已结束
	N220 G0 X2=100	通道 2：开始执行 N220
N110 G0 X1=100		通道 1：时间错开地开始执行 N110

参见

连续路径运行中的有条件停止 (WAITMC) (页 593)

编程：取轴（GET、GETD） (页 611)

7.1.2 连续路径运行中的有条件停止 (WAITMC)

如果在一条通道中执行 WAITMC 之前已经在其他通道中出现了所有需要的等待标记，那么在该通道中，运行不会减速，程序不会停止，而是继续执行后续程序段。

WAITMC 作为功能调用采用单独的程序段。为了能在不制动的情况下正确执行 WAITMC，WAITMC 前方需要直接具有一个移动程序段。对于使用 G64x 的移动程序段，则带 WAITMC 的程序段将添加到之前的程序段中，从而避免速度大幅下降 。如 WAITMC 前方无直接的移动程序段，例如 "IF 环", 则命令的归类无法保持一致性，并会出现制动。

前提条件

必须在通道中激活以下功能：

- 连续路径运行(G64, G641, G642, G643, G644 或 G645)
- “预读”功能

制动方式

无制动

从调用 WAITMC 前的一条运动程序段起，NC 就开始检查其他待同步的通道中的等待标记。如果是，轴会不减速制动：

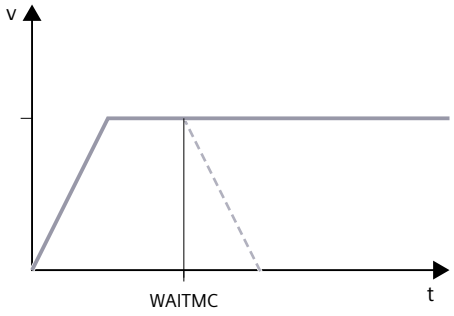


图 7-1 有条件等待 WAITCM 下轨迹速度的特性：所有通道均已执行到等待标记

短时制动

如果还有通道没有执行到等待标记，轴会开始减速。在减速期间，NC 会按插补周期检查该通道是否执行到等待标记。如果是，NC 会使轴再次加速到编程的速度：

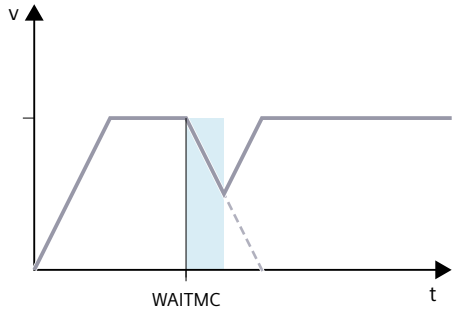


图 7-2　有条件等待 WAITCM 下轨迹速度的特性：最后的等待标记在制动期间出现

制动直至停车

如果在制动阶段没有到达等待标记，则减速至静止状态并等待。如果到达了等待标记，轴会再次加速到编程的速度：

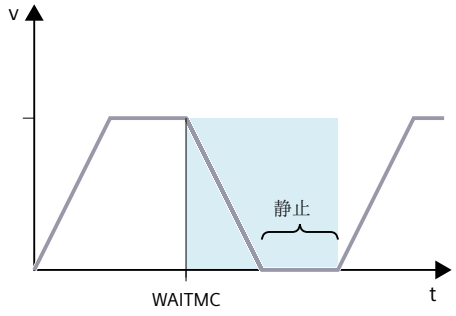


图 7-3　有条件等待 WAITCM 下轨迹速度的特性：最后的等待标记在制动后出现

制动期间的程序段切换 (IPOBRKA)

运动结束条件 IPOBRKA（程序段切换条件“制动斜坡”）生效时，到达最后缺少的等待标记时切换至下一个程序段，并启动进给轴。

示例 1：连续路径运行中的有条件停止

此示例只包含和通道同步相关的指令，仅作示意：

通道 1

程序代码	注释
N10 INIT(2,"_N_200_MPF","n")	; 选择通道 2 同步程序。
N11 INIT(3,"_N_300_MPF","n")	; 选择通道 3 同步程序。
N15 START(2,3)	; 启动通道 2 和通道 3 中的程序。
...	; 通道 1 中执行加工。
N20 WAITMC(7,2,3)	; 有条件等待通道 2 和通道 3 执行到标记 7。
...	; 通道 1 中继续加工。
N40 WAITMC(8,2)	; 有条件等待通道 2 执行到标记 8。

程序代码	注释
...	; 通道 1 中继续加工。
N70 M30	; 通道 1 结束。

通道 2

程序代码	注释
N200 ...	; 通道 2 中执行加工。
N210 SETM(7)	; 通道 2 设置等待标记 7。
...	; 通道 2 中继续加工。
N250 SETM(8)	; 通道 2 设置等待标记 8。
N260 M30	; 通道 2 结束。

通道 3

程序代码	注释
N300 ...	; 通道 3 中执行加工。
...	
N350 WHEN <条件> DO SETM(7)	; 在同步动作中设置等待标记。
...	; 通道 3 中继续加工。
N360 M30	; 通道 3 结束。

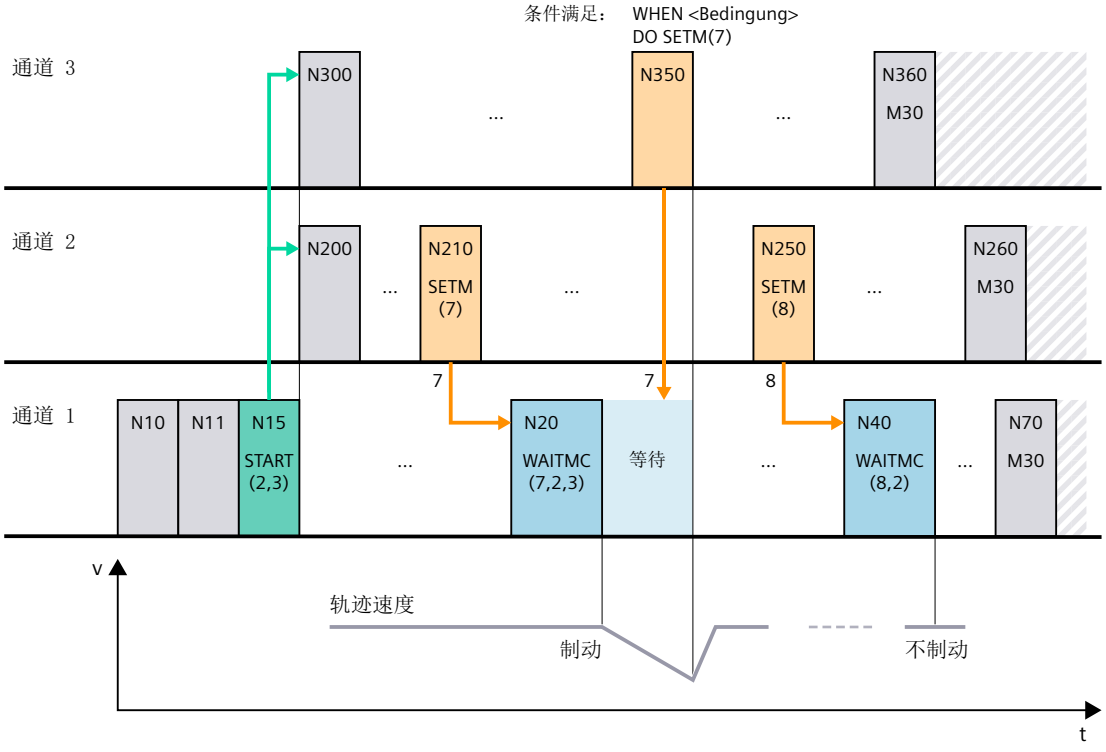


图 7-4　连续路径运行中有条件停止，涉及三个通道（示意图）

示例 2：WAITMC 和读取禁止

运行期间在通道 3 中输出辅助功能 M555 并生成读取禁止。由于 WAITMC 被分配给程序段 N312，因此设置等待标记，通道 2 继续运行。通道 3 中的程序处理则由于读取禁止而停止。

说明

G64 生效时，WAITMC 程序段**不生成单独的程序段**，而是添加到前一程序段中。连续路径运行生效时应不会出现速度跃变。这样一来，在前一程序段因读取禁止停止时，实际上已实现 WAITMC。

通道 2

程序代码	注释
N112 G18 G64 X200 Z200 F567	; 在通道 2 中加工
N120 WAITMC(1,2,3)	; 在通道 2 和通道 3 中的等待标记 1 上
	; 有条件等待
...	; 在通道 2 中继续加工，由于
	; WAITMC 被添加至程序段 N312
...	; 在通道 2 中继续加工
N170 M30	; 通道 2 结束

通道 3

	注释
	; 运行期间读取禁止 M555。
N300 ...	; 在通道 3 中加工
N312 G18 G64 D1 X180 Z300 M555	
N320 WAITMC(1,2,3)	; 等待，由于读取禁止

7.2 单通道式试车

7.2.1 功能

“单通道式试车”功能用于测试或试运行多个相互同步的通道的其中一个通道的 NC 程序。在处于“正常运行”状态下的通道中，编写的通道轴或主轴的运行导致机床轴的运行。对于处于“程序测试”状态下的通道：

- 系统默认为**轴**生成与正常运行时相一致的设定值，但**不**将值**输出**至机床轴。
- 系统默认将**主轴**的设定值**输出**至机床轴。
- 为禁用轴/主轴显示的实际值在系统内部根据设定值生成。

- 正常执行通道同步指令。
- 正常处理 NC/PLC 接口信号。
- 程序执行时间与正常模式下一致。

可视需要针对具体通道或轴暂时退出“程序测试”状态。这样一来，系统会将设定值重新输出至机床轴，从而实现实际的轴运行。

7.2.2 过程

正常情况下，通道在工作区域中移动刀具。如果多个通道在同一个工作区域中移动一个刀具，则须对刀具运动进行同步。可通过以下方式进行同步：

- 通过程序协调指令 WAITM、WAITMC、WAITE、START 进行通道同步。
- 通过 PLC 用户和 NC/PLC 接口信号进行通道同步。例如：通过将 M 功能从通道输出到 PLC，禁止从 PLC 到通道的读取。
- 跨通道取轴：一个通道等待直至另一个通道交出轴。
- NC 程序中通过全局变量进行同步。
- 跨通道同步
- 程序测试，包括主处理中的并行同步动作，以及同步动作与通道的同步。

在这些边界条件下，几乎不可能只启动一个通道，否则通道会停留在第一个同步位置。

借助“单通道式试车”功能能够只启动所需的所有通道，使得仅这些通道获得运行使能，以便对编写的运行的相互配合进行测试。其他通道则处于“程序测试”状态。

为此，机床操作人员必须在启动通道前定义需要在“程序测试”状态下运行的通道。这通过操作界面中的“程序控制”菜单实现。

7.2.3 单通道视图

选择

通过操作界面（例如 SINUMERIK Operate），为基本画面“机床”中显示的通道选择“程序测试”（PRT）：

1. 软键：操作区“加工”>“程序控制”
2. 菜单“程序控制”：选择复选框“程序测试 (PRT)”。

NC/PLC 接口信号

在选择程序测试（PRT）后，下列 NC/PLC 接口信号置位（参见下文提示“接口信号的自动传输”）：

- 通道
  - <HmiChan>.basic.in.progTestSelected == 1（来自 HMI：程序测试已选择）
  - <Chan>.basic.out.progTest == 1（来自 PLC：激活程序测试）
  - <Chan>.basic.in.progTestSelected == 1（来自 NC：程序测试生效）
- 轴
  - <HmiAxis>.basic.in.progTestSuppressionRequested == 0（来自 HMI：抑制程序测试）
  - <HmiAxis>.basic.in.progTestRequested == 0（来自 HMI：激活程序测试）
  - <Axis>.basic.out.progTestSuppression == 0（来自 PLC：抑制程序测试）
  - <Axis>.basic.out.progTestRequest == 0（来自 PLC：激活程序测试）
- 主轴
  - <HmiAxis>.basic.in.progTestSuppressionRequested == 1（来自 HMI：抑制程序测试）
  - <HmiAxis>.basic.in.progTestRequested == 0（来自 HMI：激活程序测试）
  - <Axis>.basic.out.progTestSuppression == 1（来自 PLC：抑制程序测试）
  - <Axis>.basic.out.progTestRequest == 0（来自 PLC：激活程序测试）

说明

运行的缺省状态

在通道中选择“程序测试”后，运行的缺省状态为：

- 轴：Disabled
- 主轴：许可

说明

接口信号的自动传输

当 FC1 参数 **MMCToIF** 设为 **TRUE** 时，HMI 请求信号 <HmiAxis>.basic.in.progTestSuppressionRequested / <HmiAxis>.basic.in.progTestRequested 才会从 PLC 基本程序传输至 PLC 请求信号 <Axis>.basic.out.progTestSuppression / <Axis>.basic.out.progTestRequest。若该参数未置位，则必须通过 PLC 用户程序将 PLC 请求信号置位。

禁止主轴的运行

若需要主轴在“程序测试”期间不运行，则必须通过 PLC 用户程序进行明确禁止：

- <HmiAxis>.basic.in.progTestSuppressionRequested = 0（抑制程序测试）
- <HmiAxis>.basic.in.progTestRequested = 1（激活程序测试）

- <Axis>.basic.out.progTestSuppression = 0（来自 PLC：抑制程序测试）
- <Axis>.basic.out.progTestRequest = 1（来自 PLC：激活程序测试）

注意
主轴的运行 缺省状态下，在处于“程序测试”状态下的通道中，主轴运行被使能。

使能轴的运行

若需要轴在“程序测试”期间运行，则必须通过 PLC 用户程序进行明确使能：

- <Axis>.basic.out.progTestSuppression = 1（来自 PLC：抑制程序测试）
- <Axis>.basic.out.progTestRequest = 0（来自 PLC：激活程序测试）

允许的切换时间点

- 通道
  仅在通道状态为“复位”或“中断”时，才允许对用于启用/禁用通道专用状态“程序测试”的接口信号（<HmiChan>.basic.in.progTestSelected 或 <Chan>.basic.out.progTest）进行切换。
- 轴/主轴
  始终可以对用于启用/禁用通道专用状态“程序测试”的接口信号（<HmiAxis>.basic.in.progTestSuppressionRequested / <HmiAxis>.basic.in.progTestRequested 或 <Axis>.basic.out.progTestSuppression / <Axis>.basic.out.progTestRequest）进行切换。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.progTestSelected	LBP_Chan*.E_ProgTest	DB21, ...DBX33.7

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.progTest	LBP_Chan*.A_ProgTest	DB21, ...DBX1.7
<Axis>.basic.out.progTestSuppression	LBP_Axis*.A_ProgtestSuppress	DB31, ...DBX14.0
<Axis>.basic.out.progTestRequest	LBP_Axis*.A_ProgtestActivate	DB31, ...DBX14.1

OP → PLC

Basic Program Plus	Basic Program	
<HmiChan>.basic.in.progTestSelected	LBP_HMI.E_MMC_ProgTest	DB21, ...DBX25.7
<HmiAxis>.basic.in.progTestSuppressionRequested	LBP_HMI.E_MMC_ProgtestSuppress	DB31, ...DBX128.0
<HmiAxis>.basic.in.progTestRequested	LBP_HMI.E_MMC_ProgtestActivate	DB31, ...DBX128.1

7.2.4 多通道视图

前提条件

为在菜单“程序控制”中在多通道视图中显示通道和主轴，必须满足下列前提条件：

- 通道
  - 参数设置了超过一个通道（MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP[<通道>] ≠ 0）
  - 选件“programSYNC”（6FC5800-0AP05-0YB0）经置位/许可
- 主轴
  - 通道中设置的主轴的含义经过定义（MD52206 $MC_AXIS_USAGE[<主轴>] ≠ 0）

选择

通过操作界面（例如 SINUMERIK Operate），为控制系统中参数设置的一或多个通道选择“程序测试”（PRT）：

1. 软键：操作区“加工”>“程序控制”
2. 软键：“试车”
3. 菜单“程序控制”：“试车”栏 > **取消**选择复选框“通道 x”。

通道/主轴复选框

在菜单“程序控制”：“试车”栏中，可通过复选框“通道 x”或“主轴 x”设置以下属性：

复选框	选择	撤销选择
“通道 x”	正常运行	只对程序进行计算 （通道处于试运行中）
“主轴 x”	正常运行	主轴 x 上不执行 （主轴处于试运行中）

NC/PLC 接口信号

在撤销选择“通道 x”的复选框后，通道处于“程序测试（PRT）”状态。下列 NC/PLC 接口信号置位（参见下文提示“接口信号的自动传输”）：

- 通道
  - <HmiChan>.basic.in.progTestSelected == 1（来自 HMI：程序测试已选择）
  - <Chan>.basic.out.progTest == 1（来自 PLC：激活程序测试）
  - <Chan>.basic.in.progTestSelected == 1（来自 NC：程序测试生效）
- 轴
  - <HmiAxis>.basic.in.progTestSuppressionRequested == 0（来自 HMI：抑制程序测试）
  - <HmiAxis>.basic.in.progTestRequested == 0（来自 HMI：激活程序测试）
  - <Axis>.basic.out.progTestSuppression == 0（来自 PLC：抑制程序测试）
  - <Axis>.basic.out.progTestRequest == 0（来自 PLC：激活程序测试）

缺省状态下，主轴的复选框被选中，即主轴**正常运行**：

- 主轴
  - <HmiAxis>.basic.in.progTestSuppressionRequested == 1（来自 HMI：抑制程序测试）
  - <HmiAxis>.basic.in.progTestRequested == 0（来自 HMI：激活程序测试）
  - <Axis>.basic.out.progTestSuppression == 1（来自 PLC：抑制程序测试）
  - <Axis>.basic.out.progTestRequest == 0（来自 PLC：激活程序测试）

说明

运行的缺省状态

在通道中选择“程序测试”后，运行的缺省状态为：

- 轴：Disabled
- 主轴：许可

说明

接口信号的自动传输

当 FC1 参数 **MMCToIF** 设为 **TRUE** 时，HMI 请求信号 <HmiAxis>.basic.in.progTestSuppressionRequested / <HmiAxis>.basic.in.progTestRequested 才会从 PLC 基本程序传输至 PLC 请求信号 <Axis>.basic.out.progTestSuppression / <Axis>.basic.out.progTestRequest。若该参数未置位，则必须通过 PLC 用户程序将 PLC 请求信号置位。

禁止主轴的运行

若需要通道的主轴在“程序测试”期间**不**运行，则必须通过 PLC 用户程序进行明确禁止：

- <HmiAxis>.basic.in.progTestSuppressionRequested = 0（抑制程序测试）
- <HmiAxis>.basic.in.progTestRequested = 1（激活程序测试）
- <Axis>.basic.out.progTestSuppression = 0（来自 PLC：抑制程序测试）
- <Axis>.basic.out.progTestRequest = 1（来自 PLC：激活程序测试）

注意
主轴的运行 缺省状态下，在处于“程序测试”状态下的通道中，主轴运行被使能。

使能轴的运行

若需要轴在“程序测试”期间运行，则必须通过 PLC 用户程序进行明确使能：

- <Axis>.basic.out.progTestSuppression = 1（来自 PLC：抑制程序测试）
- <Axis>.basic.out.progTestRequest = 0（来自 PLC：激活程序测试）

允许的切换时间点

- 通道
  仅在通道状态为“复位”或“中断”时，才允许对用于启用/禁用通道专用状态“程序测试”的接口信号（<HmiChan>.basic.in.progTestSelected 或 <Chan>.basic.out.progTest）进行切换。
- 轴/主轴
  始终可以对用于启用/禁用通道专用状态“程序测试”的接口信号（<HmiAxis>.basic.in.progTestSuppressionRequested / <HmiAxis>.basic.in.progTestRequested 或 <Axis>.basic.out.progTestSuppression / <Axis>.basic.out.progTestRequest）进行切换。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.progTestSelected	LBP_Chan*.E_ProgTest	DB21, ... DBX33.7

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.progTest	LBP_Chan*.A_ProgTest	DB21, ... DBX1.7
<Axis>.basic.out.progTestSuppression	LBP_Axis*.A_ProgtestSuppress	DB31, ... DBX14.0
<Axis>.basic.out.progTestRequest	LBP_Axis*.A_ProgtestActivate	DB31, ... DBX14.1

OP → PLC

Basic Program Plus	Basic Program	
<HmiChan>.basic.in.progTestSelected	LBP_HMI.E_MMC_ProgTest	DB21, ... DBX25.7
<HmiAxis>.basic.in.progTestSuppressionRequested	LBP_HMI.E_MMC_ProgtestSuppress	DB31, ... DBX128.0
<HmiAxis>.basic.in.progTestRequested	LBP_HMI.E_MMC_ProgtestActivate	DB31, ... DBX128.1

7.2.5 系统变量

“程序测试”状态可通过系统变量查询：

- 在操作界面、同步动作或带有预处理停止的零件程序中显示，通过系统变量：

$AC_ISTEST	通道的“程序测试”状态 通道处于“程序测试”状态下时，输出 TRUE (1)。
$AA_ISTEST[<n>]	轴 <n> 的“程序测试”状态 轴 <n> 处于“程序测试”状态下时，输出 TRUE (1)。

- 在不带预处理停止的零件程序中，通过系统变量：

$P_ISTEST	通道处于“程序测试”状态下时，输出 TRUE (1)。

示例

通道处于“程序测试”状态下，且轴“C”已通过“抑制程序测试”脱离该状态。通过系统变量查询时会得到以下结果：

- `$AC_ISTEST == TRUE`
- `$P_ISTEST == TRUE`
- `$AA_ISTEST[C] == FALSE`

7.2.6 边界条件

跨通道取轴

“跨通道取轴”功能能够将进给轴/主轴的信息提供给多个通道，并可由这些通道交替运行（参见“跨通道取轴 (页 609)”章节）。

在与“程序测试”和“单通道式试车”功能一起使用时，跨通道取轴时须注意：

- 如果只有一个通道处于“程序测试”状态，则从该通道取出轴，然后放入到不处于该状态的通道中。如果需要取出的轴处于“禁用状态”，在放入另一个通道后，不管该通道是否处于“程序测试”状态，该轴本身的状态不变（参见示例 3）。
- 程序测试中，零件程序结束/复位时所有未插补的进给轴/主轴将同步至当前的实际位置。在一些跨通道取轴功能中，如果轴只有在程序结束时才能从原通道中释放，那接收通道只能在程序结束后才能获得该轴。这会导致经过仿真后到达的位置不会传送给接收通道。

说明

程序末尾处应设置一个 WAIT 标记，以确保同时结束。

7.2.7 示例

示例 1：In einer 3-kanaligen Anlage soll Kanal 2 erprobt werden.

Testmöglichkeit 1:Programmtest ohne SERUPRO

1. 用户考虑需要实际运行哪些进给轴/主轴。为这些轴置位“抑制程序测试”。
2. 为通道 1 和通道 3 选择“程序测试”状态。
3. 通过 PLC 启动通道 1、2、3。
4. 程序结束后可重新取消“程序测试”。
5. 若当前的“抑制程序测试”设置也适用于其他情形（需要对通道 1 或通道 3 进行试车），可保持此信号的置位状态。很多场合都会用到此设置。

Testmöglichkeit 2:Programmtest mit SERUPRO

1. 用户考虑需要实际运行哪些进给轴/主轴。为这些轴置位“抑制程序测试”。
2. 为通道 1 和通道 3 选择“程序测试”状态。
3. 通过 PLC 启动通道 1、2、3。
4. 出现故障或报警时，按下 RESET 终止。
5. 对 3 个通道的断点执行 SERUPRO。
6. 3 个通道中均到达搜索目标。
7. 启动全部 3 个通道。
8. 通道 1 和通道 3 重新进入“程序测试”状态，“单通道式试车”继续进行。

示例 2：Einschalten von "Programmtest unterdrücken"

一个通道处于“程序测试”状态下。需要在运行中为“Y”轴触发“抑制程序测试”（程序段 N1010 处）。

程序代码	注释
N1000 G0 Y1000	
N1010 G4 F10	
N1020 G0 G91 Y=10	；增量运行
N1030 M30	

程序运行至位置 1010，即在“抑制程序测试”启用后运行该轴的仿真分量“1000”。

示例 3：Programmtest und Achstausch

Achse X1 aus Kanal 1 und Achse X2 aus Kanal 2 sind der 1.Maschinenachse AX1 der NC zugeordnet.

通道 1 进行“程序测试”	通道 2 不进行“程序测试”
N10010 G0 G90 X0	
N10020 X1=100	
N10030 WAITM(91,1,2)	
N10040 WAITM(92,1,2)	
N10050 M0	
N10060 M30	
	N20010 WAITM(91,1,2)
	N20020 G91 G0 X2=10
	N20030 WAITM(92,1,2)
	N20040 M0
	N20050 M30

在程序段 N20040 中，将机床轴 AX1 换至通道 2 中，接收通道 1 中的轴的最后一个位置并随后运行至位置 110。

7.3 边界条件

7.3.1 运行方式 MDI：连续路径运行和 WAITMC

运行方式 MDI：连续路径运行和 WAITMC

在 MDI 运行方式下，开始执行 MDI 程序段缓存时，配合连续路径运行 (G64 / G604) 的 WAITMC 指令不得处于 MDI 程序段缓存的最后一个程序段。否则程序处理会在倒数第二个运行程序段处停止，且须通过复位终止。

示例

在 NC 启动前，MDI 程序段缓存中包含以下程序段：

程序代码	注释
N10 G64 G1 G94 F5000 X100	; MDI 程序段缓存的
	; 第一个程序段的连续路径运行
N20 X200	
N30 X300	; 倒数第二个运行程序段
N40 X400	; 程序加工在此处
N50 WAITMC(...)	; 因 N50 中的 WAITMC 停止。
	; 需用 RESET 键终止!

7.3.2 根据 WAIT 指令非同步开始运行

在通道协调时，通过 WAIT 指令可导致非同步开始执行后续程序段。在同步通道中到达共同同步位置之前触发动作，并在删除剩余行程中导致隐性重新定位 (REPOSA) 时发生这种行为。

接受：通道 1 和 2 中的当前轴分配

- 通道 1：轴 Z1 和 W
- 通道 2：轴 Z2

示例 1：延时启动运行

通道 1 和 2 中的以下指令顺序会导致同步位置 N110 / N210 后的延时启动出现问题。

表格 7-2　通道 1 和 2 中的时间顺序

通道 1	通道 2	说明
...	...	在通道 1 和 2 中随意执行
N100 W100		通道 1：运行轴 W

通道 1	通道 2	说明
...	...	在通道 1 和 2 中任意执行
N110 WAITM(2,1,2)	N210 WAITM(2,1,2)	通道 1 和 2：等待同步
开始 *"AUTOGET(W)"：* • *轴交换* • *剩余行程删除* • *REPOSA* *结束* N120 Z1=100	*"AUTOGET(W)" 并等待通道 1 的轴 W* N220 Z2=100 W100	通道 2：延时启动运行 通道 1：启动运行

示例 2：同步启动运行

示例 1 中的延时启动运行可通过明确使能和请求需要交换的轴 W 和插入另一个同步位置避免。通道 1 和 2 中的以下指令顺序会触发同步位置 N120 / N220 后的同步启动运行：

表格 7-3　通道 1 和 2 中的时间顺序

通道 1	通道 2	说明
...	...	在通道 1 和 2 中任意执行
N100 W100		通道 1：运行轴 W
...	...	在通道 1 和 2 中任意执行
RELEASE(W)		通道 1：使能轴 W ⇒ 通道 2 中的取轴不再影响通道 1
N110 WAITM(2,1,2)	N210 WAITM(2,1,2)	通道 1 和 2：等待同步
	N215 GET(W)	通道 2：请求轴 W。由于轴 W 已使能，取轴时间得以优化。
N120 WAITM(9,1,2)	N220 WAITM(9,1,2)	通道 1 和 2：等待同步
N130 Z1=100	N230 Z2=100 W100	通道 1 和 2：同步启动运行

跨通道取轴 8

8.1 概述

说明

主轴

下文针对**轴**的“跨通道取轴”的描述和功能说明同样适用于**主轴**。

在控制系统调试中，每根轴必须指定给一个通道。这根轴只能由该通道运行，例如通过零件程序或同步动作。使用“跨通道取轴”功能可释放轴并将其指定给另一个通道，即跨通道取轴。这样该轴才能由其他通道运行。

轴状态

在“跨通道取轴”功能范畴下，轴可能的状态如下：

- “通道轴”
  通道轴是指被指定给一个通道的轴。其可通过零件程序运行，或手动运行。
- “PLC 轴”
  PLC 轴是指被指定给 PLC 的轴。该轴只能由 PLC 用户程序运行。
- “中立轴”
  中立轴是指当前尚未指定给通道或 PLC 的轴。运行前必须先由通道或 PLC 请求该轴。
- “另一个通道中的轴”
  通道请求轴时，该轴即处于此状态。但其尚归属于另一通道，因此不能指定给请求它的通道。

8.2 调试

参数设置

NC 专用机床数据

- 跨通道取轴特性的常规参数设置
  MD10722 $MN_AXCHANGE_MASK

通道专用机床数据

- 参数设置哪根轴归属于通道，即为通道轴：
  MD20070 $MC_AXCONF_MACHAX_USED[<通道轴>] = <机床轴>
  提示：任何情形下均须将 NC 中使用的所有轴指定为一个或多个通道的通道轴，而不单单是使用“跨通道取轴”功能时。

轴专用机床数据

- 若一根轴为多个通道中的通道轴，则通过以下机床数据定义控制系统启动（上电复位）后该轴指定给哪一个通道：
  MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN[<通道>] = <通道编号>
  参数设置的通道是该轴的主通道。
- 在零件程序中编写了轴，而该轴当前未指定给那个通道时，参数设置响应：
  – 显示报警，且不运行轴。
  – 根据 GET (页 611) 指令自动请求轴
  – 根据 GETD (页 611) 指令自动请求轴
  MD30552 $MA_AUTO_GET_TYPE[<轴>] = <响应>

系统变量

轴专用系统变量

- 跨通道取轴相关的轴状态：
  $AA_AXCHANGE_TYP[<轴>]
- 跨通道取轴相关的轴统计：
  $AA_AXCHANGE_STAT[<轴>]

8.3 编程：使能轴（RELEASE）

功能

将归属于当前通道的轴通过预定义程序 RELEASE() 使能以用于跨通道取轴，为为此使其切换至“中立轴”状态。

句法

```
RELEASE（<轴 1>[、轴 2 ... 轴 15]）
```

含义

RELEASE：	使能轴，以用于跨通道取轴	
	预处理停止：	是
	在单独程序段中编程：	是
<轴>：	轴：使能的轴的通道轴名称 主轴：使能的主轴的通道轴名称，或通过 SPI（<主轴编号>）将主轴编号转换为通道轴名称	
	类型：	AXIS

边界条件

无法使能

- 轴参与了转换。
- 轴位于轴组中。

龙门轴组中的主动轴

使能龙门轴组中的主动轴时也会同时使能所有的从动轴。

8.4 编程：取轴（GET、GETD）

请求跨通道取轴（GET）

功能

为此，使用预定义程序 GET() 为该通道请求该轴。

在该轴所归属于的通道中，必须借助零件程序或同步动作使用 RELEASE() 使能该轴，以将其用于跨通道取轴。

跨通道取轴后，该轴状态为“通道轴”。

句法

GET（<轴 1>[、轴 2 ... 轴 15]）

含义

GET：	为当前通道指定一根轴	
	预处理停止：	是
	在单独程序段中编程：	是
<轴>：	轴：所请求的轴的通道轴名称 主轴：所请求的主轴的通道轴名称，或通过 SPI（<主轴编号>）将主轴编号转换为通道轴名称	
	类型：	AXIS

边界条件

以下情形下，跨通道取轴会被延迟：

- 轴尚未由其当前归属于的通道通过 RELEASE 使能。
- 测量系统切换尚未完成
- 控制器使能状态更改尚未完成（从控制过渡至跟踪/停止，或反之）
- NC/PLC 接口信号“轴或主轴禁用”存在（<Axis>.basic.out.fixedStopEnable == 1）
- 当前轴运行（插补）尚未完成。

直接取轴（GETD）

功能

零件程序的下一个加工段中需要用到未指定给当前通道的轴。使用预定义程序 GETD() 从该轴所归属于的通道直接取轴。待取轴**无需**由其所归属于的通道通过 RELEASE() 使能。

跨通道取轴后，该轴状态为“通道轴”。

根据轴在让渡通道中的状态，可能会在该通道中触发预处理停止（STOPRE）：

- “通道轴”状态 ⇒ 预处理停止
- “中立轴”状态 ⇒ **无**预处理停止

为了协调通过 GETD() 进行的跨通道取轴，建议启用请求通道和让渡通道间的通道同步 (页 587)。

注意
让渡通道中的预处理停止 轴在让渡通道中处于“通道轴”状态时，会在该通道中触发预处理停止（STOPRE）：

句法

GETD(<轴>)

含义

GETD：	将轴直接取至当前通道	
	预处理停止：	是
	在单独程序段中编程：	是
<轴>：	轴：所请求的轴的通道轴名称 主轴：所请求的主轴的通道轴名称，或通过 SPI（<主轴编号>）将主轴编号转换为通道轴名称	

边界条件

轴在让渡通道中的状态为“PLC 轴”时，必须通过 PLC 用户程序使能该轴以用于跨通道取轴。

边界条件

通道复位

- 若请求轴的通道中触发了通道复位，那么跨通道取轴会终止。
- 通道复位后，所取轴仍归属于最后请求该轴的通道。

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<Axis>.basic.out.fixedStopEnable	LBP_Axis*.A_EnabTravFixedStop	DB31, ...DBX3.1

8.5 自动取轴

功能

在零件程序或同步动作中编写了轴，而该轴未指定给当前通道时，自动执行跨通道取轴，或自动将该轴取至当前通道。

前提条件

通过 GET() 或 GETD() 设置自动跨通道取轴：

MD30552 $MA_AUTO_GET_TYPE (页 609)

边界条件

参见对 GETD() 的描述 (页 611)。

示例

示例 1

程序代码	注释
N1 M3 S1000	; 运行主主轴
N2 RELEASE (SPI(1))	; 使能，将状态切换为中立轴
N3 S3000	; 编写主主轴 => 自动取轴
	; 特性取决于 MD30552 $MA_AUTO_GET_TYPE:
	; 0 => 报警“轴类型错误”
	; 1 => 隐性 GET(SPI(1))
	; 2 => 隐性 GETD(SPI(1))

示例 2

程序代码	注释
	; 机床轴 AX1 ≙ 通道轴 X
N1 RELEASE (AX1)	; 使能，将状态切换为中立轴
N2 G04 F2	; 暂停时间
N3 G0 X100 Y100	; 编写 X 轴作为轨迹轴
	; 特性取决于 MD30552 $MA_AUTO_GET_TYPE:
	; 0 => 报警“轴类型错误”
	; 1 => 隐性 GET(AX1)
	; 2 => 隐性 GETD(AX1)

示例 3

程序代码	注释
	; 机床轴 AX1 ≙ 通道轴 X
N1 RELEASE (AX1)	; 使能，将状态切换为中立轴
N2 G04 F2	; 暂停时间
N3 POS (X) = 100	; 编写 X 轴作为定位轴
	; 特性取决于 MD30552 $MA_AUTO_GET_TYPE:
	; 0 => 报警“轴类型错误”
	; 1 => 隐性 GET(AX1) *)
	; 2 => 隐性 GETD(AX1) *)

*) 轴尚为同步时，系统不会为通过 GET() 或 GETD() 进行的自动取轴生成独立程序段。

8.6 通过 PLC 取轴

功能

可借助 PLC 用户程序通过 NC/PLC 接口请求跨通道取轴：

- 从一个 NC 通道取至 PLC
- 从 PLC 取至 NC 通道
- 从一个 NC 通道取至另一个 NC 通道

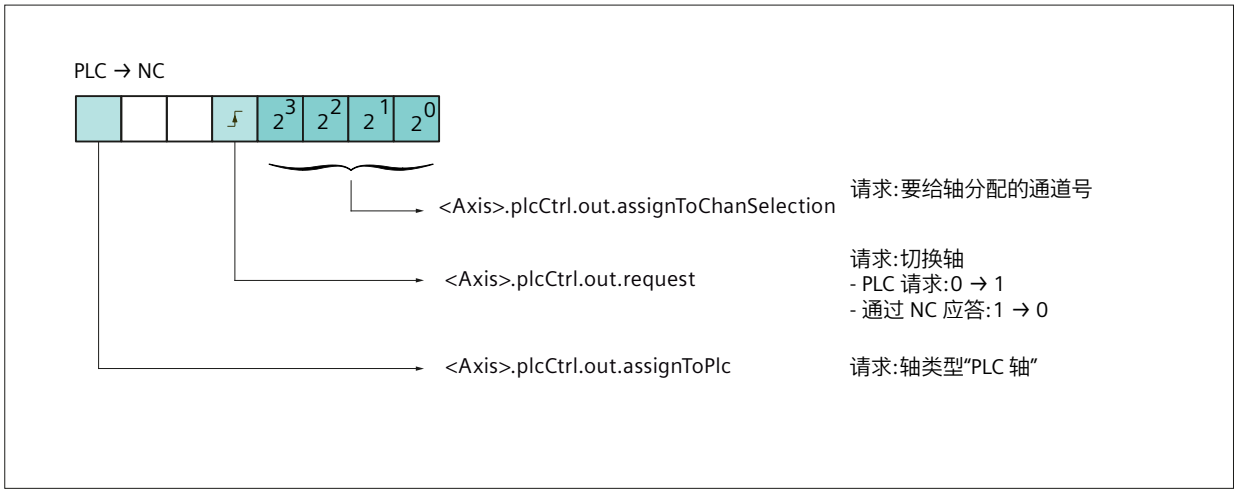


图 8-1 跨通道取轴请求：PLC → NC

跨通道取轴状态

可借助 PLC 用户程序通过 NC/PLC 接口读取轴的跨通道取轴状态。

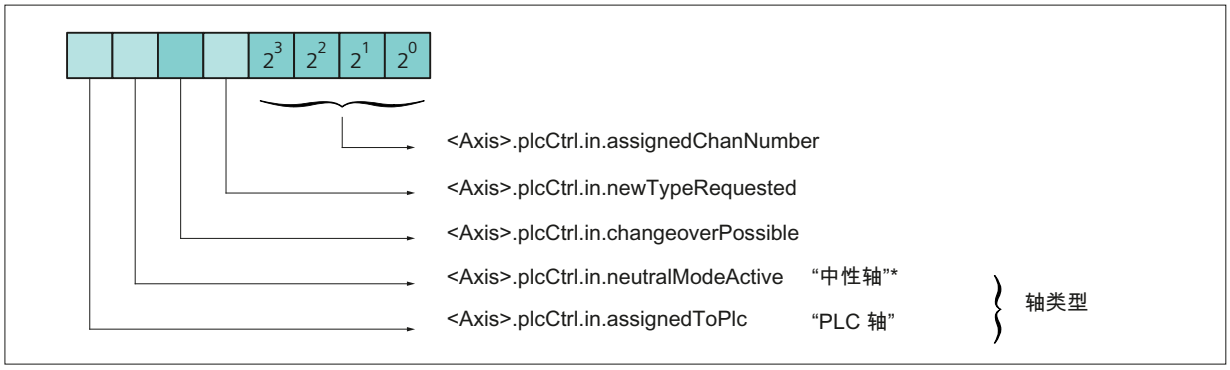


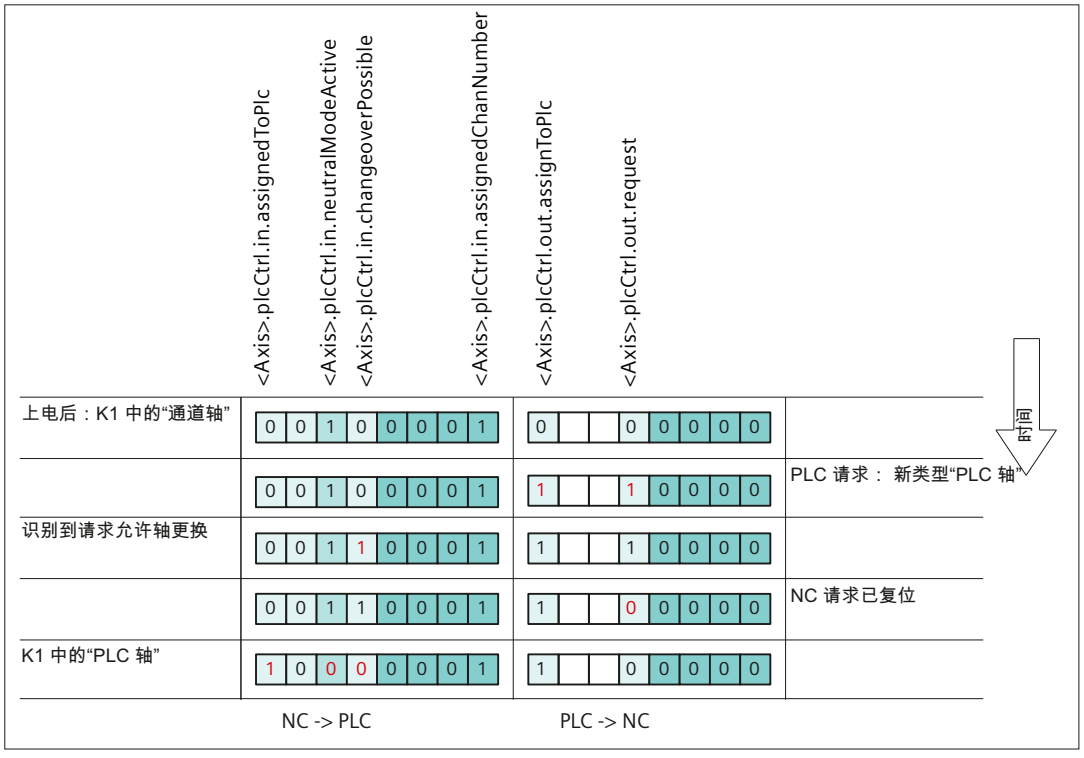
图 8-2　跨通道取轴状态：NC → PLC

示例

示例 1

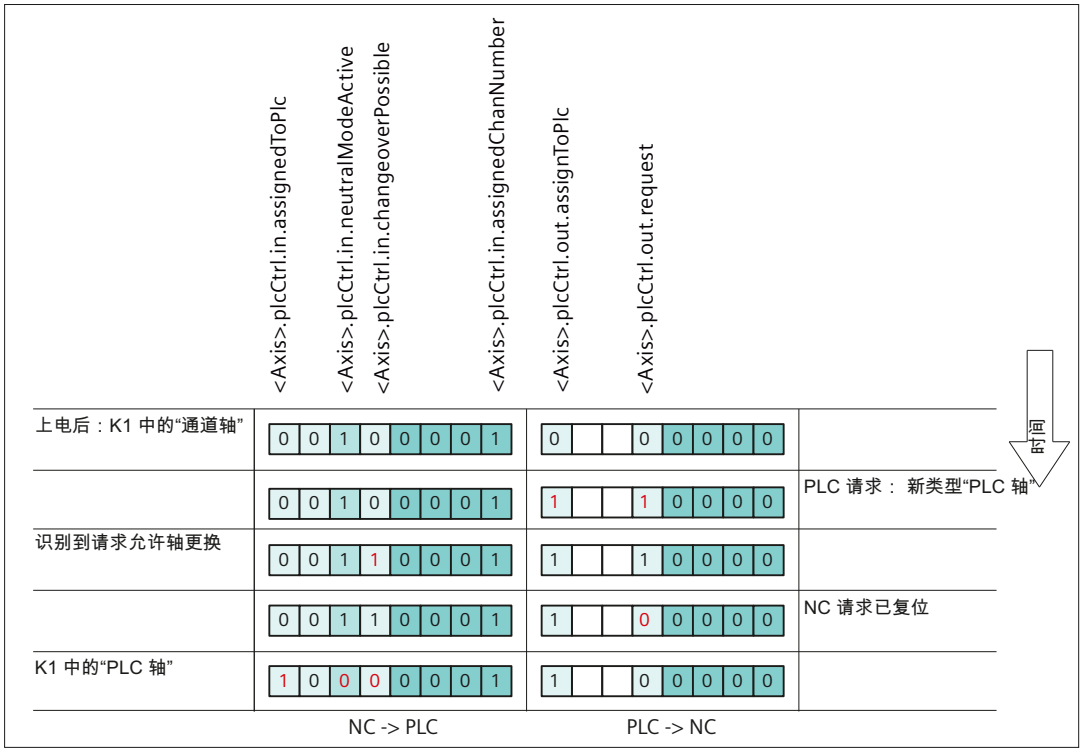
通过在零件程序编写 RELEASE() 和 GET() 将轴从通道 1 取至通道 2，零件程序在相应通道中执行：

- 通道 1：RELEASE(<轴>)
- 通道 2：GET(<轴>)



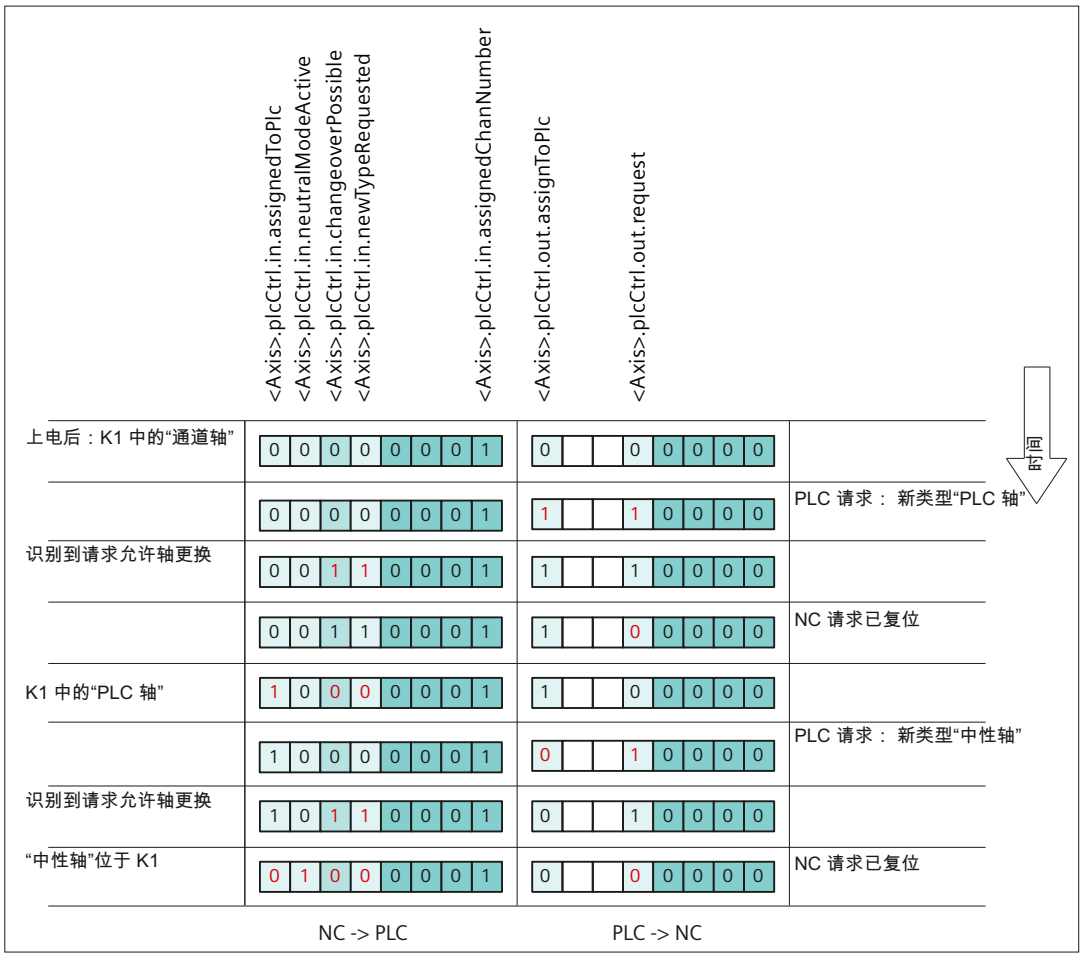
示例 2

通过 PLC 用户程序将指定给通道 1 的“NC 轴”的状态切换至“PLC 轴”。



示例 3

通过 PLC 用户程序将指定给通道 1 的“NC 轴”的状态通过“PLC 轴”切换至“中性轴”。



8.7 通过 PLC ONE 进行跨通道取轴时的信号表

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Axis>.plcCtrl.in.assignedChanNumber	LBP_Axis*.E_NCASpChanA..D	DB31, ...DBX068.0 ..3
<Axis>.plcCtrl.in.newTypeRequested	LBP_Axis*.E_PLCType	DB31, ...DBX68.4
<Axis>.plcCtrl.in.changeoverPossible	LBP_Axis*.E_ChPoss	DB31, ...DBX68.5
<Axis>.plcCtrl.in.neutralModeActive	LBP_Axis*.E_NeutrASp	DB31, ...DBX68.6
<Axis>.plcCtrl.in.assignedToPlc	LBP_Axis*.E_PLCASp	DB31, ...DBX68.7

PLC → NC

Basic Program Plus	Basic Program	
<Axis>.plcCtrl.out.assignToChanSelection	LBP_Chan*.A_NCASpChanA..D	DB31, ...DBX08.0..3
<Axis>.plcCtrl.out.request	LBP_Chan*.A_AxResume	DB31, ...DBX08.4
<Axis>.plcCtrl.out.assignToPlc	LBP_Chan*.A_PLCASp	DB31, ...DBX08.7

8.8 触发/不触发预处理停止的取轴

跨通道取轴功能的新增特性：不触发预处理停止

使用此 GET 指令只会生成一个中间程序段，而不是触发预处理停止的 GET 程序段。在主处理中执行该程序段时，系统会检查程序段中的轴状态是否与当前轴状态一致。 若不一致，则可能会触发强制重组。

系统检查**进给轴或定位主轴**的以下状态：

- 是否是进给轴或定位主轴
- 设定位置

检查**处于转速模式的主轴**的以下状态：

- 主轴运行方式：转速模式
- 主轴转速 S
- 旋转方向 M3, M4
- 齿轮档 M40, M41, M42, M43, M44, M45
- 主主轴是否为恒定切削速度

必要时可执行强制重组。从动轴则必须进行强制重组。

激活

不触发预处理停止的取轴以及对当前状态的检查通过以下机床数据激活：MD10722 $MN_AXCHANGE_MASK，位 2=1。

示例

激活取轴，不触发预处理停止

表格 8-1

```
N010 M4 S1000
N011 G4 F2
N020 M5
N021 SPOS=0
N022 POS[B]=1
N023 WAITP[B]                                   ; 轴 b 转换为中立轴
N030 X1 F10
N031 X100 F500
N032 X200
N040 M3 S500
N041 G4 F2
N050 M5
N099 M30
```

若在程序段 N023 后将主轴（轴 B）直接转换为 PLC 轴，使轴运行至 180°再返回 1°，然后再转换为中立轴，那么程序段 N040 不会触发预处理停止和重组。

特殊情况：触发预处理停止的取轴

只要 GET 或 GETD 指令还没有进入主处理，就可以用 RELEASE(轴) 或 WAIT(轴) 再次释放轴。接着的 GET 指令会**导致**预处理停止。

8.9 仅由 PLC 控制的轴

功能

控制系统启动后，轴处于“中立轴”状态，并由 PLC 控制。为了将其作为并行定位轴运行（由 PLC 通过 SINU_TraversePosAxis（Basic Program FC18）运行），必须先由 PLC 显式请求这些轴。

说明

可通过机床数据将 PLC 取轴范围限制为“仅由 PLC 控制的轴”。MD10722 $MN_AXCHANGE_MASK，位 3 = 1

该轴**无法**通过 NC 零件程序运行。

参数设置

通过以下轴专用机床数据将轴设置为仅由 PLC 控制的轴：

MD30460 $MA_BASE_FUNCTION_MASK，位 4 = 1

由 PLC 控制

仅由 PLC 控制的轴的运行特性只可通过轴 NC/PLC 接口信号调整：

- <Axis>.spindle.out.reset（复位）
- <Axis>.spindle.out.resume（继续）
- <Axis>.spindle.out.stop（沿制动斜坡停止）

可采用的运行功能

仅由 PLC 控制的轴可使用以下运行功能：

1. 通过运行键和手轮在 JOG 模式下运行
2. 轴回参考点
3. 通过静态同步动作作为指令轴运行
4. 作为异步往复轴运行
5. 由 PLC 通过 `SINU_TraversePosAxis`（Basic Program FC18）作为并行定位轴运行

运行功能 1 至 4 结束后，该轴自动恢复为“中立轴”状态。由 PLC 结束功能 5 后，轴仍为 PLC 轴。只有由 PLC 明确释放后，轴才变为中立轴。

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<Axis>.spindle.out.reset	LBP_Axis*.A_AxReset	DB31, ...DBX28.1
<Axis>.spindle.out.resume	LBP_Axis*.A_AxResume	DB31, ...DBX28.2
<Axis>.spindle.out.stop	LBP_Axis*.A_Stop	DB31, ...DBX30.0

8.10 固定指定给 PLC 的轴

功能

控制系统启动后，轴处于“中立轴”状态，且由 NC 通道控制。为了将其作为并行定位轴运行（由 PLC 通过 SINU_TraversePosAxis（Basic Program FC18）运行），**无需**先由 PLC 显式请求这些轴。通过 SINU_TraversePosAxis（Basic Program FC18）发出运行请求时，PLC 会自动取轴。通过 SINU_TraversePosAxis（Basic Program FC18）请求的运行结束后，轴自动恢复为“中立轴”状态。

取轴完成后，通过 PLC 请求也可由 PLC 控制该轴：“PLC 轴”状态。

说明

可通过机床数据将 PLC 取轴范围限制为“固定指定给 PLC 的轴”。MD10722 $MN_AXCHANGE_MASK，位 3 = 1

参数设置

通过以下轴专用机床数据将轴设置为固定指定给 PLC 的轴：

MD30460 $MA_BASE_FUNCTION_MASK，位 5 = 1

由 PLC 或 NC 通道控制

固定指定给 PLC 的轴的运行特性可通过 NC 通道或 PLC 调整：

NC 通道：通道专用 NC/PLC 接口信号（选择）

- <Chan>.basic.out.ncStart（NC 启动）
- <Chan>.basic.out.ncStop（NC 停止）
- <Chan>.basic.out.reset（复位）

PLC：轴专用 NC/PLC 接口信号

- <Axis>.spindle.out.reset（复位）
- <Axis>.spindle.out.resume（继续）
- <Axis>.spindle.out.stop（沿制动斜坡停止）

可采用的运行功能

固定指定给 PLC 的轴可使用以下运行功能：

1. 通过运行键和手轮在 JOG 模式下运行
2. 轴回参考点
3. 由 PLC 通过 SINU_TraversePosAxis（Basic Program FC18）作为并行定位轴运行

运行功能 1 至 3 结束后，该轴自动恢复为“中立轴”状态。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.out.ncStart	LBP_Chan*.A_NCStart /	DB21, ...DBX7.1
<Chan>.basic.out.ncStop	LBP_Chan*.A_NCStop /	DB21, ...DBX7.3
<Chan>.basic.out.reset	LBP_Chan*.A_Reset /	DB21, ...DBX7.7

PLC → NC

Basic Program Plus	Basic Program	
<Axis>.spindle.out.reset	LBP_Axis*.A_AxReset	DB31, ...DBX28.1
<Axis>.spindle.out.resume	LBP_Axis*.A_AxResume	DB31, ...DBX28.2
<Axis>.spindle.out.stop	LBP_Axis*.A_Stop	DB31, ...DBX30.0

8.11 跨通道取轴和经过旋转的 WCS 中的几何轴的综合应用

跨通道取轴功能的新增特性：支持含旋转分量的框架指令

在 JOG 模式下，可通过静态同步动作将经过旋转的 WCS 中的几何轴作为 PLC 轴或指令轴运行。为此必须设置以下机床数据：

MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED，位 10 == 1

说明

JOG 模式下切换运行方式

在 JOG 模式下，经过旋转的 WCS 中**所有**作为几何轴相互关联的 **PLC 轴和指令轴**必须结束运行，方可切换至其他运行方式。这些轴至少需要恢复为中立轴状态，否则运行方式切换时会发出报警 16908。此外，若在经过旋转的坐标系中只有一根几何轴作为 PLC 轴或指令轴运行，也会触发此报警。

此类轴只能在通道内转换为 PLC 轴或指令轴。不允许由其他通道取轴。

从 JOG 切换至 AUTO 的前提条件

在从 JOG 切换至 AUTO 运行方式时，仅在设置了以下机床数据时，“程序中断”时几何轴运动的终点才会被接收。

MD 32074：FRAME_OR_CORRPOS_NOTALLOWED，位 11 == 1

此时会根据 WCS 中的旋转量定位 PLC 轴或指令轴。

受 WCS 旋转影响的**所有**轴均被归入一个几何轴轴组，并一同处理。因此，该组的所有轴均

- 被指定给 NC 程序，或者
- 所有轴为中立轴，或者
- 所有轴为主处理轴（PLC 轴、指令轴或往复轴）。

例如，几何轴组中的一根轴写入了 WAITP 时，系统会等待该组中的所有其他轴，从而将其一同转换为中立轴。若其中一根轴在主处理中转换为 PLC 轴，那么该组中的所有其他轴将转换为中立轴。

边界条件

若 MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED 位 10 == 0，且在 NC 程序中编写了 `ROT Z45`，则**无法**跨通道取 X 轴和 Y 轴。对于 Z 轴也存在类似情形，例如编写了 `ROT X45` 或 `ROT Y45`，或者 JOG 模式下程序段由于此类编程中断时。此时虽然会为 X 轴和 Y 轴置位 NC/PLC 接口信号“可以取轴”（`<Axis>.plcCtrl.in.changeoverPossible`），但之后会复位。

仅在 MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED 位 10 == 1，且当前执行的程序段不包含此类指令时，才可在 JOG 模式下跨通道取这些轴。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Axis>.plcCtrl.in.changeoverPossible	LBP_Axis*.E_ChPoss	DB31, ...DBX68.5

8.12 通过同步动作取轴

功能

通过同步动作，可使用（GET(<轴>)）请求一根轴，并使用（RELEASE(<轴>)）释放该轴用于取轴。

说明

相关轴必须通过机床数据作为通道轴指定给通道。

使用指令 AXTOCHAN，可通过同步动作或零件程序将轴直接传递给特定通道。此时不必是其自身通道，该通道也无需具备该轴的当前插补权限。

当前状态和轴的插补权限

跨通道取轴所涉及的轴类型和插补权限可通过以下系统变量读取：

<值> = $AA_AXCHANGE_TYP[<轴>]

<值>	含义
0	轴被指定给 NC 程序。
1	轴被指定给 PLC ，或者作为指令轴/往复轴生效。
2	另一个通道具有插补权限。
3	轴为中立轴。
4	中立轴由 PLC 控制。
5	另一个通道具有插补权限，已为 NC 程序请求该轴。
6	另一个通道具有插补权限，已请求该轴作为中立轴。
7	轴为 PLC 轴，或作为指令轴/往复轴生效，已为 NC 程序请求该轴。
8	轴为 PLC 轴，或作为指令轴/往复轴生效，已请求该轴作为中立轴。

<值>	含义
9	固定指定的 PLC 轴，中立轴状态。
10	固定指定的 PLC 轴，由 PLC 控制，中立轴状态。

“处于中立轴状态下固定指定的 PLC 轴” (9) 和 “处于中立轴状态下由 PLC 控制的固定指定的 PLC 轴” (10) 固定分配给 PLC，**与 GET 和 RELEASE** 无关。能否对该轴执行取轴通过系统变量 $AA_AXCHANGE_STAT[轴] 显示。

同步动作中执行 GET、RELEASE 的过渡状态以及 GET 完成时的情形

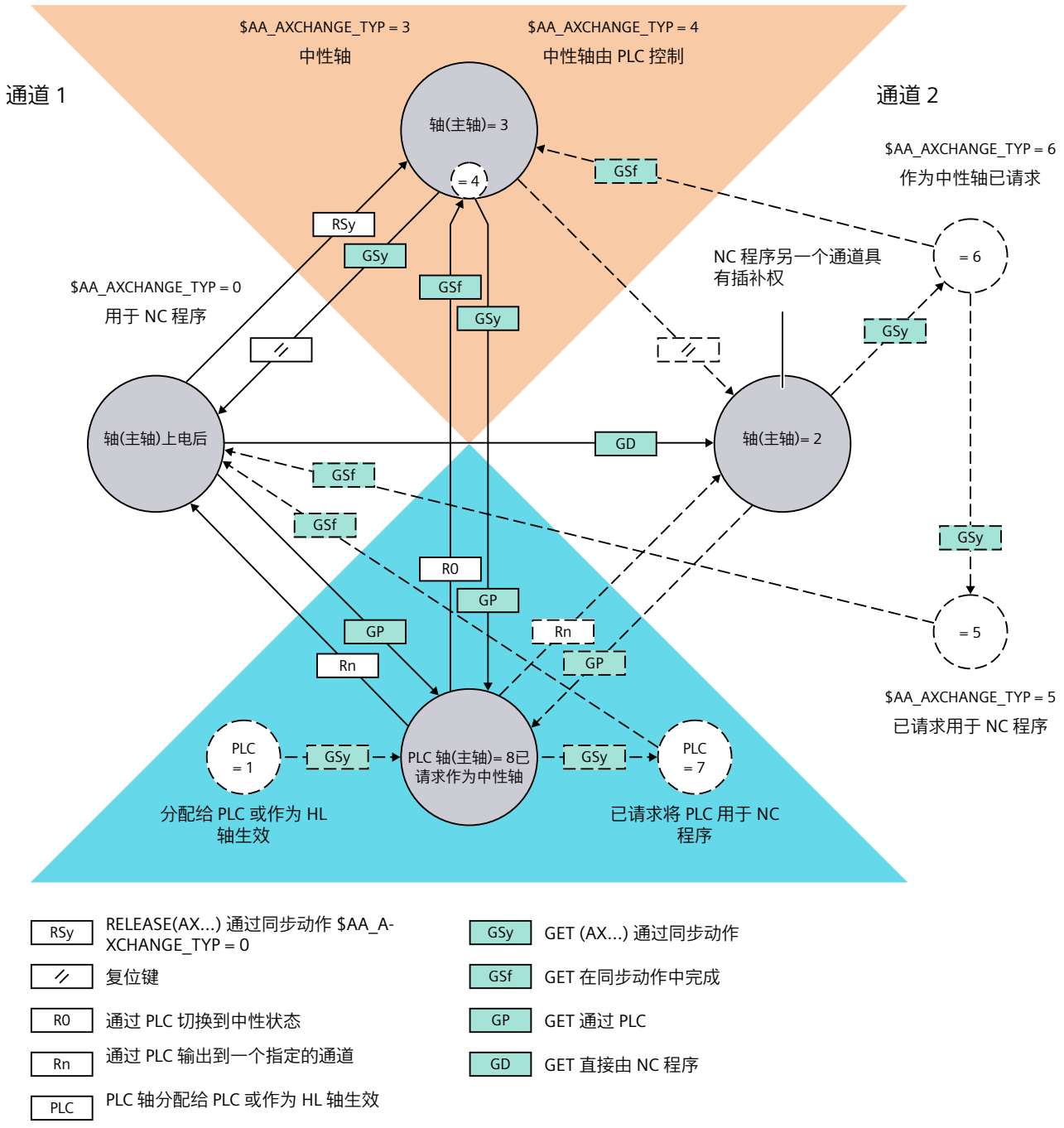


图 8-3　同步动作中的过渡状态

更多信息

参见功能手册之同步动作；同步动作中可执行的动作

8.13 跨通道取轴和龙门轴的综合应用

功能

跨通道取轴时，一个封闭的龙门轴组被当作一个单元处理。因此跨通道取龙门轴组中主动轴的操作也会同时获取轴组中的从动轴。此时除了要满足之前的章节中描述的适用于主动轴的前提条件外，还须满足该组所有从动轴的前提条件。

轴机床数据

在取龙门轴时，必须将该组所有轴的以下机床数据设为相同的值：

- MD30460 $MA_BASE_FUNCTION_MASK，位 4（控制分量）
- MD30460 $MA_BASE_FUNCTION_MASK，位 5（指定分量）

轴专用 NC/PLC 接口信号

在跨通道取轴功能范畴内，龙门轴组内所有轴的以下 NC/PLC 接口信号总是显示相同的值：

- <Axis>.plcCtrl.in.resetDone（复位已执行）
- <Axis>.plcCtrl.in.active（PLC 控制轴）
- <Axis>.plcCtrl.in.stopped（轴停止生效）

轴专用系统变量

在跨通道取轴功能范畴内，龙门轴组内所有轴的以下以下系统变量总是显示相同的值：

- $AA_AXCHANGE_TYP（跨通道取轴相关的轴类型）
- $AA_AXCHANGE_STAT（跨通道取轴相关的轴状态）
- $AA_SNGLAX_STAT（单轴类型）

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Axis>.plcCtrl.in.resetDone	LBP_Axis*.E_AxReset	DB31, ...DBX63.0
<Axis>.plcCtrl.in.active	LBP_Axis*.E_PLCCtrlAx	DB31, ...DBX63.1
<Axis>.plcCtrl.in.stopped	LBP_Axis*.E_AxStop	DB31, ...DBX63.2

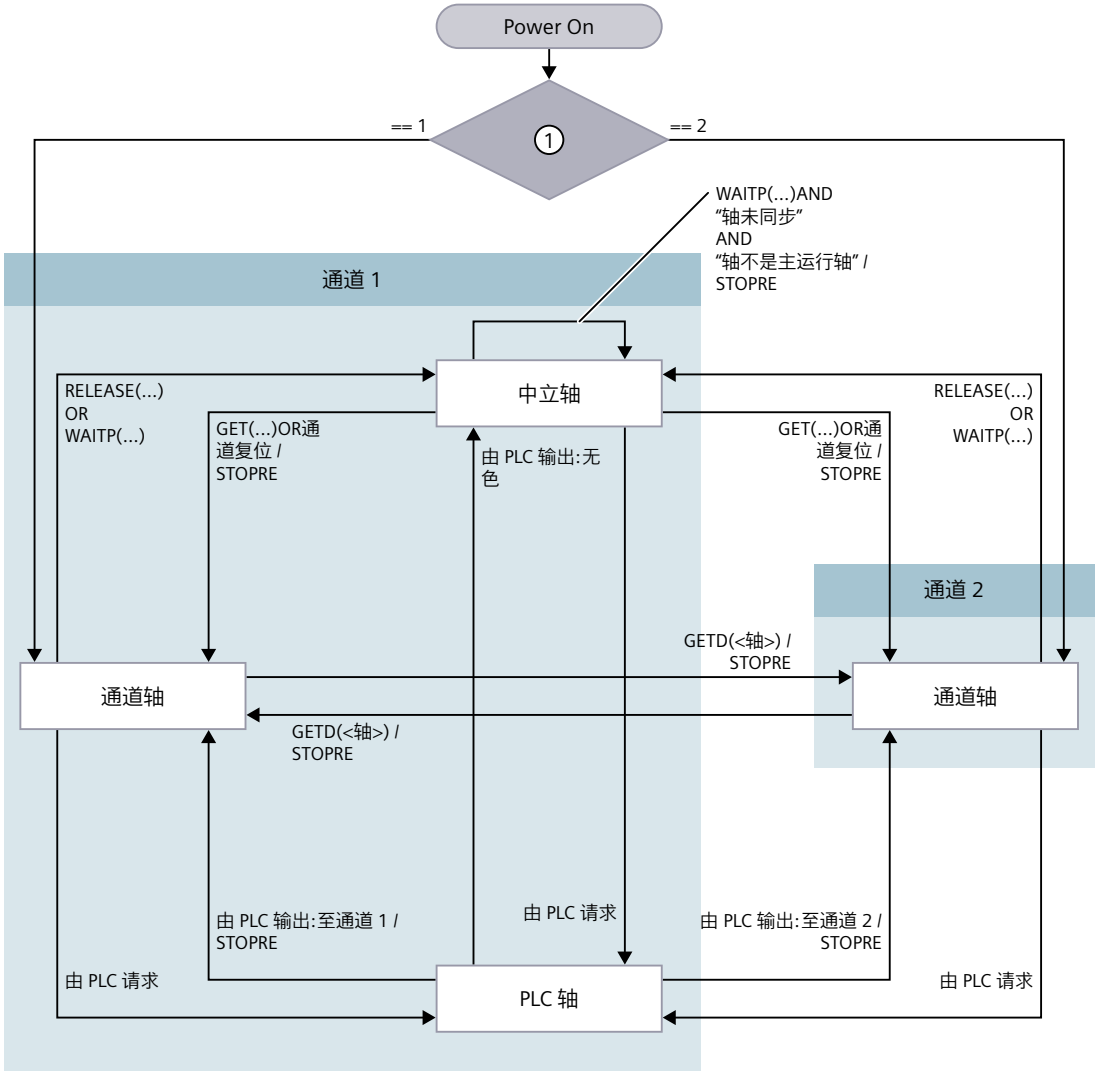
8.14 状态图

下图显示的是通道 1 中“跨通道取轴”功能相关的轴的状态、事件、动作和状态过渡。

对于通道 2，为了便于概览，子状态“中立轴”和“PLC 轴”不予显示。

假设：缺省状态下轴被指定给通道 1：

MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN[<轴>] = 1



① MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN[<轴>]

图 8-4 状态图：跨通道取轴

8.15 边界条件

在轴从“PLC 轴”、“中立轴”或“另一个通道中的轴”状态切换为“通道轴”状态时，系统会通过预处理停止进行同步，以在取轴通道中实现同步。此时：

- 轴：接收当前轴位置
- 主轴：接收当前转速和当前齿轮级

8.15.1 通过 PLC 跨通道取轴

通过 PLC 请求跨通道取轴（PLC → 通道，或通道 → PLC）时，若通道的零件程序处于下列加工段其中之一，那么要待离开加工段后才会执行跨通道取轴。

- 连续路径运行（G64/G640）
- 螺纹切削/攻丝（G33/G331/G332）

8.15.2 带计算的程序段搜索

在带计算的程序段搜索中，动作程序段中只会输出相互取消的指令 GET、GETD 和 RELEASE。

示例

带计算的程序段搜索，至目标程序段 N700：

程序代码	注释
...	
N100 RELEASE(AX1)	; 记录 RELEASE(AX1)
N110 GET(AX2)	; 记录 GET(AX2)
...	
N400 GET(AX1)	; GET(AX1) 取消 RELEASE(AX1) =>
	; RELEASE(AX1) 和 GET(AX1) 不予以保存
...	
N700 ...	; 目标程序段 ⇒ 输出：GET(AX2)
N710 RELEASE(AX2)	
...	

更多信息

更多信息请参见 程序段搜索类型 1、2 和 4 (页 147)。

8.16 示例

假设

- 通道 1：通道轴为下列轴：1、2、3、**4**
- 通道 2：通道轴为下列轴：**4**、5、6
- 缺省对应关系：缺省状态下轴 4 被指定给通道 1

参数设置

通道 1

通道中的轴名称 MD20080

- $MC_AXCONF_CHANAX_NAME_TAB[<通道 1>][0] = "X" ; 第 1 通道轴
- $MC_AXCONF_CHANAX_NAME_TAB[<通道 1>][1] = "Y" ; 第 1 通道轴
- $MC_AXCONF_CHANAX_NAME_TAB[<通道 1>][2] = "Z" ; 第 1 通道轴
- $MC_AXCONF_CHANAX_NAME_TAB[<通道 1>][3] = "U" ; 第 1 通道轴

所使用的机床轴： MD20070

- $MC_AXCONF_MACHAX_USED[<通道 1>][0] = 1 ; 第 1 通道轴 → 轴 1
- $MC_AXCONF_MACHAX_USED[<通道 1>][1] = 2 ; 第 2 通道轴 → 轴 2
- $MC_AXCONF_MACHAX_USED[<通道 1>][2] = 3 ; 第 3 通道轴 → 轴 3
- $MC_AXCONF_MACHAX_USED[<通道 1>][3] = **4** ; 第 4 通道轴 → 轴 4

通道 2

通道中的轴名称 MD20080

- $MC_AXCONF_CHANAX_NAME_TAB[<通道 2>][0] = "X" ; 第 1 通道轴
- $MC_AXCONF_CHANAX_NAME_TAB[<通道 2>][1] = "Y" ; 第 1 通道轴
- $MC_AXCONF_CHANAX_NAME_TAB[<通道 2>][2] = "U" ; 第 1 通道轴

所使用的机床轴：MD20070

- $MC_AXCONF_MACHAX_USED[<通道 2>][0] = 5 ; 第 1 通道轴 → 轴 5
- $MC_AXCONF_MACHAX_USED[<通道 2>][1] = 6 ; 第 2 通道轴 → 轴 6
- $MC_AXCONF_MACHAX_USED[<通道 2>][2] = **4** ; 第 3 通道轴 → 轴 4

缺省对应关系

轴 4（AX4）的主通道 → 通道 1

AX4 是对应 MD10000 $MN_AXCONF_MACHAX_NAME_TAB[3] 的第 4 机床轴的缺省名称

- MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN[AX4] = 1

程序示例

通道 1 中的程序	通道 2 中的程序
...	...

通道 1 中的程序	通道 2 中的程序
; 运行轴 4（AX4）	; 与通道 1 的同步点
G01 F1000 U100	; WAITM(1,1,2)
; 使能 AX4	; 请求 AX4
RELEASE(AX4)	GET(AX4)
; 在通道 2 中选择程序 TAUSH2	; 运行轴 4（AX4）
INIT (2,"_N_MPF_DIR_N_TAUSH2_MPF","S")	G0 U0
; 在通道 2 中启动程序 TAUSH2	...
START(2)	
; 与通道 2 的同步点	; 使能 AX4
WAITM (1,1,2)	RELEASE(AX4)
...	...
M30	M30

程序预处理 9

9.1 功能

前提条件

“程序预处理”是一个需要购买许可证的软件选件。必须通过“许可证管理”将许可证分配给硬件后，才能使用该选件。

功能

程序预处理的目的是缩短循环/子程序的运行时间，避免影响控制系统功能。预处理的实现方式是，在启动循环/子程序时控制系统将其编译为方便执行的二进制中间代码。在调用循环/子程序时，不会加载原始的 ASCII 程序，而是加载可更快执行的编译后的程序。

编译后的程序与原始程序的名称相同，但会带有一个文件扩展名“_CYC”。此程序只能执行，不可读写。同时，也不能更改和存档。每次启动控制系统时，都会重新生成此程序。原始程序保持不变。

所有可通过更正程序段修正的程序错误都会在预处理时进行识别。此外在使用跳转和控制结构时会检查，跳转目标是否存在，以及控制结构的嵌套是否正确。这样在程序执行时便可直接跳转到目标程序段，无需在所有程序段中进行查找，如同在编译未经预处理的程序时所作的操作。

预处理时识别到的错误可由用户在原始程序中进行更正。编译程序此时不会发生变化。为使错误更正或其他程序更改能在编译程序中生效，必须重启控制系统，以重新生成编译程序。

为避免执行老版本的编译程序，在调用程序时会检查，编译程序的版本是否早于原始程序。如果版本早于原始程序，编译程序会被删除并输出报警，提示必须重新执行程序预处理。

应用

程序预处理主要适用于以下程序：

- 含有高级数控编程语言元素（跳转、控制结构、同步动作运行）的循环/子程序
- 计算密集型循环/子程序（例如：轮廓车削循环）
- 含有关键运行时间点的循环/子程序（例如：删除剩余行程后继续执行或循环中预处理停止）
  示例：
  如果存在作为预处理循环的中断程序，那么在程序中断后可更快地恢复程序执行。

激活

程序预处理通过机床数据激活。

通过该机床数据还可以设置，是在标准循环目录和用户循环目录 _N_CST_DIR、_N_CUS_DIR 和 _N_CMA_DIR 下的所有程序中使用该功能，还是在这些目录中的部分选定程序中使用该功能。此外还可以设置，只对带有程序属性 PREPRO 的程序进行预处理。

更多信息：章节“参数设置 (页 638)”

程序调用

程序预处理生效时，在调用循环/子程序时会调用编译程序，而不是原始 ASCII 程序。

具体执行取决于程序调用的编写方式：

程序调用	结果
循环名称不带文件扩展名 示例：CYCLE	如果程序预处理生效并调用不带文件扩展名的循环（CYCLE），那么会加载编译循环（CYCLE_CYC），而不是 ASCII 循环（CYCLE_SPF）。 如果根据参数设置，因缺少 PREPRO 标记而无法进行调用时，则会加载 ASCII 循环（CYCLE_SPF）。
循环名称带文件扩展名"CYC" 示例：CYCLE_CYC	如果调用明确带有文件扩展名"_CYC"的循环，则会调用预处理循环。 如果编译循环不存在，则会报告出错。 根据参数设置，如果循环中缺少 PREPRO 标记，则也会报错。
循环名称带文件扩展名"SPF" 示例：CYCLE_SPF	如果调用明确带有文件扩展名"_SPF"的循环，那么即使编译循环存在，也会调用 ASCII 循环。

说明

调用带有文件扩展名"_SPF"或"_CYC"的循环只允许用于不带传输参数的循环/子程序。

所需存储空间

程序预处理功能需要使用额外的存储空间。在定义存储器大小时需要考虑这一点。

更多信息：章节 "参数设置 (页 638)"

为了减少占用的存储空间，可采取以下方式：

- 将运行时间长的循环集中保存在一个目录下，只对该目录下的程序进行预处理。
- 使用程序属性 PREPRO，只对选定程序进行预处理。
- 使用 DISPLOF（关闭显示）。

9.2 参数设置

激活/禁用

程序预处理功能通过以下机床数据激活/禁用：

MD10700 $MN_PREPROCESSING_LEVEL（程序预处理等级）

通过该机床数据还可以对程序预处理的执行方式进行设置：

位	值	含义
0		程序预处理无效
	0	循环的调用说明未知 循环应与普通子程序一样在调用前使用关键字 EXTERN 进行声明。 **提示** 该设置只能用于不含传输参数的循环。
	1	控制系统启动时会生成循环的调用说明。目录 _N_CST_DIR、_N_CUS_DIR 和 _N_CMA_DIR 下所有带有传输参数的标准循环和用户循环都可以在不使用 EXTERN 声明的零件程序中进行调用。 如果循环的参数接口发生变化，那么在控制系统重启后变化才会生效。
1	1	控制系统启动时，目录 N_CST_DIR、_N_CUS_DIR 和 _N_CMA_DIR 下的所有标准循环和用户循环都会通过一次可优化执行的编译进行预处理。
2	1	控制系统启动时，目录 N_CST_DIR 下的标准循环会通过一次可优化执行的编译进行预处理。
3	1	控制系统启动时，目录 _N_CUS_DIR 下的用户循环会通过一次可优化执行的编译进行预处理。
4	1	控制系统启动时，目录 _N_CMA_DIR 下的用户循环会通过一次可优化执行的编译进行预处理。
5	0	控制系统启动时，通过位 1 ... 4 指定的目录下的所有循环都会进行预处理。这也会对不带程序属性 PREPRO 的循环生效。
	1	控制系统启动时，带有程序属性 PREPRO 的所有循环都会进行预处理。不带属性标记的程序不进行预处理。
6	0	只要有空余的存储空间，编译程序都会保存在动态用户存储器中。 如果没有空余空间，程序预处理则会中断。
	1	只要有空余的存储空间，编译程序都会保存在动态用户存储器中。 如果没有空余空间，编译程序会保存在静态 NC 存储器中。

示例配置		结果
MD10700 $MN_PREPROCESSING_LEVEL = 63	⇒ 位 0, 2, 3, 4 和 5 置位。	只对部分选定程序（带 PREPRO 标记的程序）进行预处理。 编译程序只保存在动态用户存储器中。
MD10700 $MN_PREPROCESSING_LEVEL = 127	⇒ 位 0, 2, 3, 4, 5 和 6 置位。	只对部分选定程序（带 PREPRO 标记的程序）进行预处理。 编译程序保存在动态用户存储器中，也可选择使用静态用户存储器。

所需存储空间

执行预处理时的所需存储空间应能满足调用第一子程序级中的预处理程序的需要。每个跳转标记/跳转目标以及每个控制结构元素在预处理时都会被作为一个变量。这些在为零件程序中定义的变量确定存储器的大小时都应加以考虑：

MD28020 $MC_MM_NUM_LUD_NAMES_TOTAL（局部用户变量的数量）

MD28010 $MC_MM_NUM_REORG_LUD_MODULES（重组时局部用户变量程序块的数量）

MD28040 $MC_MM_LUD_VALUES_MEM（局部用户变量的存储器大小）

MD18242 $MC_MM_MAX_SIZE_OF_LUD_VALUE（LUD/GUD 值的存储区域大小）

每个程序占用的空间大小都应进行检测。操作步骤如下例所示：

程序代码	注释
PROC NAMES	
DEF INT VARIABLE, FELD[2]	; 2 个名称
ANFANG:	; 1 个名称，只用于预处理
FOR VARIABLE = 1 TO 9	; 1 个名称，只用于预处理
G1 F10 X=VARIABLE*10-56/86EX4+4*SIN(VARIABLE/3)	
ENDFOR	; 1 个名称，只用于预处理
M17	

为了正常执行该程序，MD28020 $MC_MM_NUM_LUD_NAMES_TOTAL 至少应指定 3 个名称。

为了在控制系统启动时对该程序进行预处理，必须指定 6 个名称。

通道相关性

程序预处理始终在第一个通道中执行。也就是说，必须在第一个通道中激活与执行加工的通道相关的全部功能。

如果要在预编译循环中直接使用几何轴名称和通道轴名称，则还必须通过以下机床数据在所有通道中将可设置的几何轴名称和通道轴名称设为相同的：

MD20060 $MC_AXCONF_GEOAX_NAME_TAB（通道中的几何轴名称）

MD20080 $MC_AXCONF_CHANAX_NAME_TAB（通道中的通道轴名称）

通常在加工循环中不会直接使用轴名称，因为循环在编写时与通道和机床上定义的轴名称不相关。

待运行的轴只是间接通过机床数据进行响应，或者作为参数传输，如下例所示：

示例 1：间接轴编程

程序代码	注释
...	
IF $AA_IM[AXNAME($MC_AXCONF_CHANAX_NAME_TAB[4])] > 5	; 如果第 5 通道轴在 MCS 中的实际值大于 5，则会执行该程序段。
G1 AX[AXNAME($MC-AXCONF-GEOAX-NAME-TAB[0])] = 10	; 将第一根几何轴运行到数值 10。
F1000 G90	
ENDIF	
...	

示例 2：从主程序中传输待运行的轴

程序代码	注释
PROC BOHRE(AXIS BOHRACHSE)	; 循环定义
WHILE $AA_IW[BOHRACHSE] > -10	
G1 G91 F250 AX[BOHRACHSE] = -1	
ENDWHILE	
...	

参见

存储器配置 (页 899)

9.3 边界条件

编程语言范围

在零件程序中可以使用所有类型的 NC 语言。

所测量过程变量的计算以及过程中和其他通道中的信号（倍率、删除剩余行程、同步动作运行）响应都不受任何限制。

ISO 语言模式

在预处理程序中不能使用 G291 切换到外部 ISO 语言模式，此时会触发报警，如下例所示：

程序代码	注释
PROC SUB1 PREPRO	
G291	; 编译时报警（不能使用 G291）
G0 X0 Y0 Z0	
M17	

调用预处理循环时，会显式切换到西门子语言模式。

使用程序全局用户变量

在要进行预处理的程序中，不允许使用程序全局用户变量（PUD），因为这些在执行调用的主程序中创建的变量在编译和控制系统启动后会无法识别。

9.4 示例

下面的示例展示了如何使用 MD10700 $MN_PREPROCESSING_LEVEL 更改程序的预处理方式。

程序代码
PROC UP1 PREPRO
N1000 DEF INT ZAEHLER
N1010 ZIEL:G1 G91 COMPCAD
N1020 G1 X0.001 Y0.001 Z0.001 F100000
N1030 ZAEHLER=ZAEHLER+1
N1040 ZAEHLER=ZAEHLER-1
N1050 ZAEHLER=ZAEHLER+1
N1060 IF ZAEHLER <=10 GOTOB ZIEL
N1070 M30

程序代码

```
PROC UP2
N2000 DEF INT VARIABLE, FELD[2]
N2010 IF $AN_NCK_Version < 3.4
N2020 SETAL(61000)
N2030 ENDIF
N2040 ANFANG:
N2050 FOR VARIABLE = 1 TO 5
N2060 G1 F1000 X=VARIABLE*10-56/86EX4+4*SIN(VARIABLE/3)
N2070 ENDFOR
N2080 M17
PROC MAIN
N10 G0 X0 Y0 Z0
N20 UP1
N30 G0 X10 Y10 Z10
N40 UP2
N50 G0 X100 Y100
N60 UP3
N70 G0 X10 Y10
N80 M30
```

示例配置		结果
MD10700 $MN_PREPROCESSING_LEVEL = 45	⇒ 位 0, 2, 3 和 5 置位。	子程序 UP1 进行预处理，生成调用说明。 子程序 UP2 不进行预处理，但也生成调用说明。
MD10700 $MN_PREPROCESSING_LEVEL = 13	⇒ 位 0, 2 和 3 置位。	两个子程序都进行预处理，生成调用说明。

测量 10

10.1 简要说明

第 1 级测量：单通道测量

功能

可以在 CNC 控制系统上同时连接最多 2 个切换式测头。在对特定通道进行测量时，总是由在相关通道中运行的零件程序来激活用于 CNC 通道的测量过程。测量程序段中编写的所有轴均参与测量过程。可分别对触发事件（上升沿或下降沿）和一个带或不不带剩余行程的测量模式进行编程。可借助零件程序或同步动作，在机床坐标系或工件坐标系中读取测量结果。可通过查询系统变量和输出至 PLC 接口来检查测头的偏转，并在零件程序中将响应导出。第 2 级测量提供了另一个功能。

优点

- 处理时的进程影响
- 确保处理精确性

第 2 级测量：单轴测量（可选项；短代码 M32）

订货号：6FC5800-0BM32-0YB0

功能

在应用于零件程序中的运动程序段中时，测量功能被限制在一个程序段内，而同步动作中的测量功能则可以独立于零件程序的方式随时激活。可在 NC 程序段中将测量事件指定给轴（单轴测量）。在进行同时测量时，在每个位置控制循环内最多可对 4 个触发事件进行分析。测量值通过 3 个参数读取：测头、轴和测量边沿。

采用持续（循环）测量时，系统将测量结果写入一 FIFO 变量。可通过循环读取 FIFO 值实现不间断测量。可以文件的形式对测量结果进行记录。

优点

- 处理时的进程影响
- 确保处理精确性
- 记录测量结果

工件和刀具的测量功能

功能

通过工件和刀具的测量功能，可顺利解决铣床或车床上的所有常规测量任务。测量可通过 HMI 操作或零件程序中的测量循环来执行。既可在测量程序段中所编写的轴均参与时针对特定通道执行测量功能，也可以在零件程序或在跨 NC 程序段的同步动作中针对特定轴执行测量功能。

其他属性：

- 在 JOG 方式下测量
- 可同时连接 2 个测头
- 采用框架的空间测量
- 可调节的测量方式，删除/不删除剩余行程
- 显示并记录测量参数/测量结果（使用测量循环）
- 可在机床坐标系或工件坐标系中读取测量结果
- 借助同步动作在加工工件的同时进行循环测量

优点

- 确保处理精确性
- 缩短辅助时间
- 关闭故障源
- 制造过程自动化。

西门子测量循环（选件；代码 P28）

订货号：6FC5800-0BP28-0YB0

功能

西门子作为选件提供的测量循环是用于解决特定刀具或工件测量任务的预定义子程序，可通过相应参数针对具体测量条件加以匹配。

优点

- 可解决车床/铣床上几乎所有待解决的测量任务。
- 自动直接在机床上进行测量使零件的生产质量稳定
- 使用带有图形支持的输入屏幕，即使在复杂的测量任务时也能快速完成编程
- 工件高精度以及经优化的质量数据透明度

更多信息：编程手册之测量循环

10.2 测头

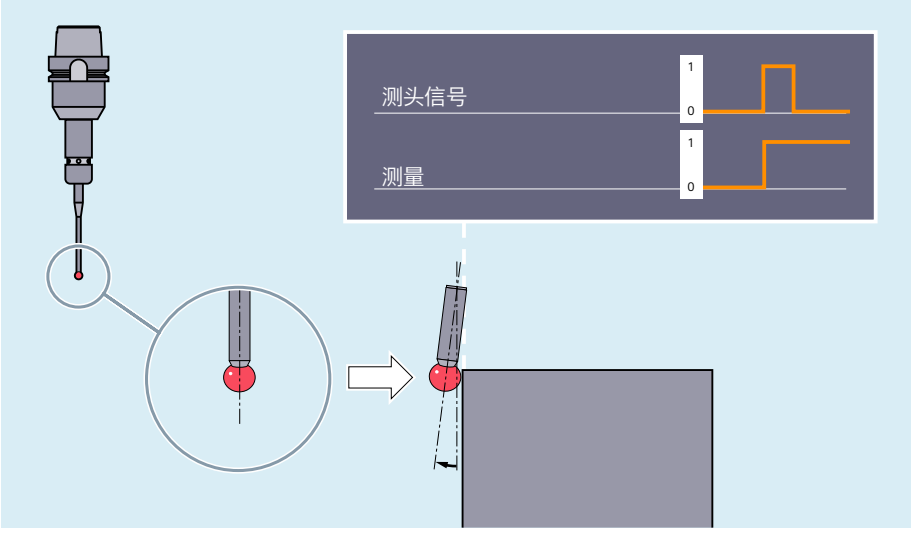
根据使用目的，可以使用不同的测头。

10.2.1 触发式测头

采集刀具和工件尺寸时，需要使用一个在偏转时可提供稳定的无跳跃信号（非脉冲）的触发式测头。

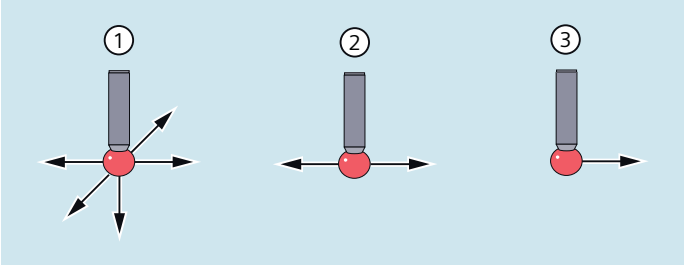
工作原理

当探针球接触到被测对象时，会产生一个开关量信号。该开关量信号会作为读取测量值的触发事件。



类型

根据可能的偏转方向数量，可分为以下测头类型：



① 多向测头 (3D)
适用于车床和铣床上的所有刀具测量及工件测量。

② 双向测头
适用于铣削和加工中心（例如加工单向测头）和车床上的工件测量。
不适用于刀具测量。

③ 单向测头
限制较少，适合用于在铣削和加工中心上进行工件测量。
不适合用于车床上的工件测量。
不适用于刀具测量。

单向测头的使用前提

单向测头用于铣削和加工中心需符合以下前提条件：

- 主轴必须可通过 SPOS 功能进行定位。
- 测头开关信号可传输范围必须大于 360° 。

此外，测头必须在主轴中经过适当的机械校准， 使主轴处于 0°时，可以在以下方向进行测量：

平面	测量方向
G17 (X-Y)	X 轴正向
G18 (Z-X)	Z 轴正向
G19 (Y-Z)	Y 轴正向

说明

由于需要在测量循环中多次通过 SPOS 定位主轴，整个测量过程持续时间较长。

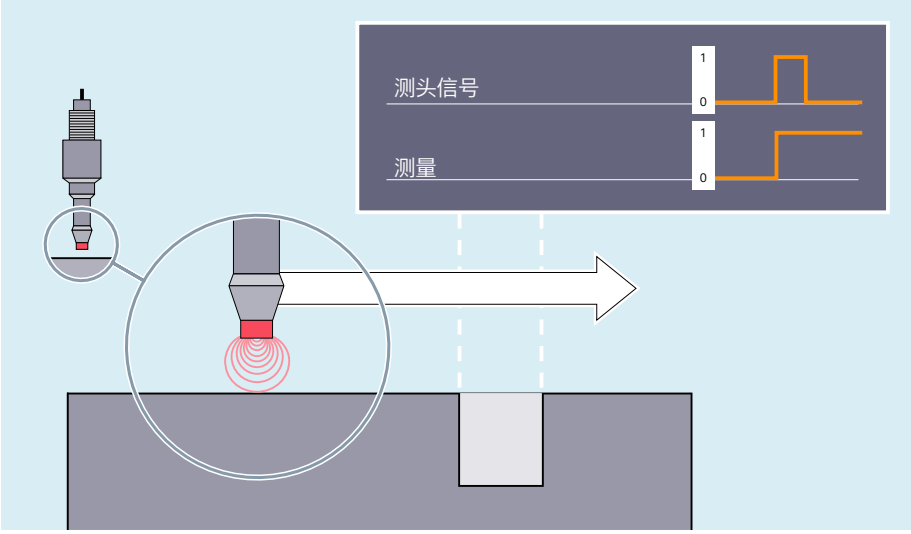
10.2.2 非触发式测头

除了触发式测头，还可以在机床上使用非触发式测头，例如：在测量特定轴时。

非触发式测头比如可作为感应测头使用。

工作原理

感应测头可在其传感面周围产生一个高频电磁场，该磁场可穿过传感器与被测对象之间的间距发生作用。如果与传感器的间距发生变化，电磁场也会变化。传感器会检测这些变化并将其转换为开关量信号。



10.3 第 1 级测量：单通道测量

在对特定通道进行测量时，总是由在相关通道中运行的零件程序来激活用于 NC 通道的测量过程。

一旦测量程序段激活，测头便向工件移动，并且程序段中编程的轨迹和定位轴运动开始。

当测量程序段中编程的触发事件发生时，所有涉及的轨迹和定位轴的当前位置被记录并存储在系统变量中。

根据功能的不同（MEAS/MEASF 或 MEAW），在触发事件发生后，行程运动会以规定的方式减慢（带删除剩余行程的测量）或继续到结束处（不带删除剩余行程的测量）。

前提条件

MEASF

使用 MEASF 功能类型必须具有需要购买授权的选件“测量级别 2”。

句法

```
MEAS=<TE> G...F...X...Y...Z...
MEASF=<TE> G...F...X...Y...Z...
MEAW=<TE> G...F...X...Y...Z...
```

说明

MEAS、MEASF 和 MEAW 为逐段生效且可以和运行指令一同编程。进给率、插补类型以及轴的数量须根据相应的测量问题进行调整。

含义

MEAS	通道专用测量，**带**删除剩余行程		
MEASF	通道专用**快速**测量，**带**删除剩余行程 如果设置了“测量级别 2”选件，MEASF 类型也可用于通道专用测量带删除剩余行程。 通过 MEASF，为测量做准备的特殊内部控制措施可确保优化测量过程并尽快提供测量结果。 因此，MEASF 用于时间要求严格的测量任务。		
MEAW	通道专用测量，**不带**删除剩余行程		
<TE>	触发测量的触发事件		
	类型：	INT	
	取值范围：		-2, -1, 1, 2
	值：	(+)1	测头 1 的上升沿（测量输入 1 上）
		-1	测头 1 的下降沿（测量输入 1 上）
		(+)2	测头 2 的上升沿（测量输入 2 上）
		-2	测头 2 的下降沿（测量输入 2 上）
	提示： 最多可使用 2 个测量探头，视扩建阶段而定。		
G...:	插补类型（例如 G0、G1、G2 或者 G3）		

F...:	进给率
X...Y...Z... :	直角坐标的终点

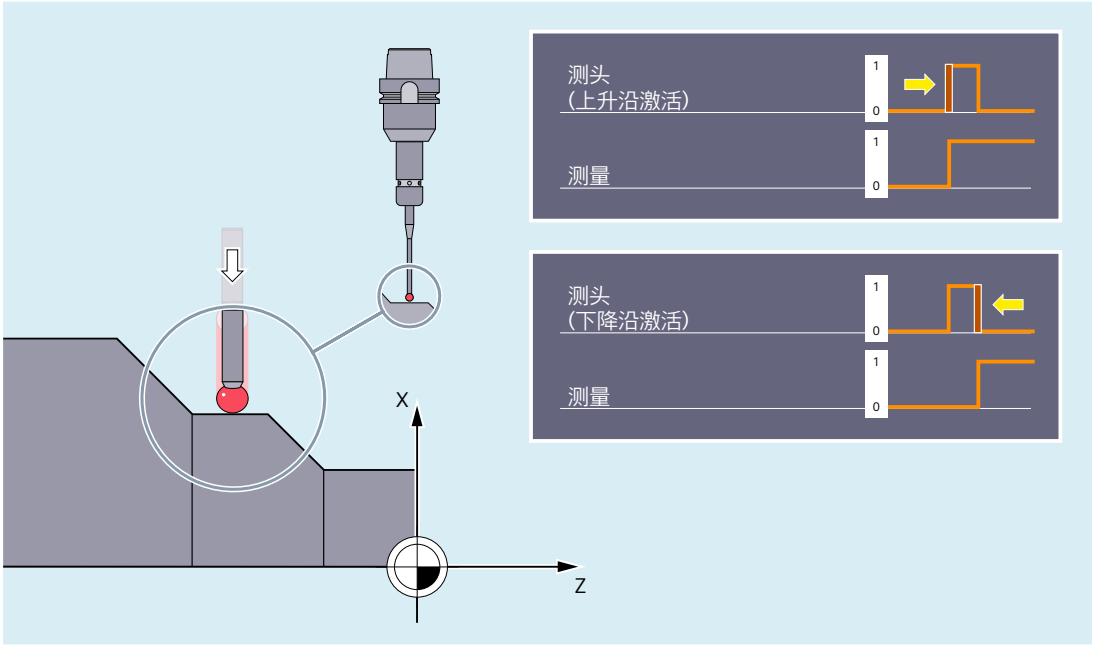
示例

程序代码	注释
... N10 **MEAS=1** G1 F1000 X100 Y730 Z40 ...	; 带有第一个测量输入端的测头和直线插补的测量程序段。 ; 测头信号的上升沿触发测量。 ; 自动产生预处理程序停止。

其它信息

测量过程

测量的起因是在测量程序段中编程的触发事件，即测头 1 或 2 的上升沿（0 → 1）或下降沿（1 → 0）：



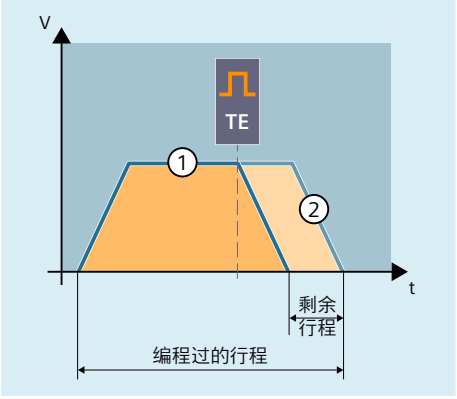
当触发事件发生时，程序段的所有运行轨迹和定位轴的位置被记录并存储在系统变量中。

说明

如果在测量程序段中编程几何轴，所有几何轴的测量值将被存储。
如果在某个测量程序段中编程了某个参与转换的轴，就保存所有参与该转换的轴的测量值。

测量带或者不带删除剩余行程

使用 MEAS 或 MEASF，在触发事件发生后，行程运动会以规定的方式减慢（带删除剩余行程的测量）；使用 MEAW 则继续到结束处（不带删除剩余行程的测量）：



① 测量**带**删除剩余行程（MEAS/MEASF）
② 测量，**不带**删除剩余行程（MEAW）

读取测量结果

被测轴的测头测量值可在零件程序及同步动作中通过以下系统变量读取：

系统变量	含义
$AA_MM[<Axis>]	机床坐标系中的测头测量值
$AA_MW[<Axis>]	工件坐标系中的测头测量值

查询状态

如果需要在需要在程序中了解测头是否偏转以及是否切换，可以通过以下系统变量查询该状态；

系统变量	含义	数据类型	值	
$A_PROBE[<n>]	测头的偏差状态	INT	0	测头未偏转。
			1	测头偏转。

系统变量	含义	数据类型	值	
$AC_MEA[<n>]	测头切换状态 测量开始时 $AC_MEA[<n>] 会被自动复位。	INT	0	测头未切换。
			1	测头已切换

<n> = 测头编号

10.4 第 2 级测量：单轴测量（可选项）

针对特定轴的测量可以通过零件程序**或同步动作**激活测量过程。特定时刻每个轴只能有一个测量任务激活。

测量的起因是在测量程序段中编程的触发事件。每个轴最多可编程 4 个触发事件。触发事件由测头或测量输入的编号（1 或 2）和触发标准（测头信号的上升沿或下降沿）确定。

通过测量模式可设置触发事件是在位置控制器周期中按照它们发生的顺序同时评估，还是按照编程的顺序接连评估。

轴上配备两个测量系统时，便可以使用这两个系统进行测量。

根据功能类型，测量任务在最后一个触发事件发生时完成（轴特定的测量，带或不带删除剩余行程）或触发器事件在每次发生后再次激活并重复评估（轴特定连续测量，不删除剩余行程）。

测量结果存储在系统变量中，或者在连续测量的情况下，存储在 FIFO 变量中。

句法

```
MEASA[<Axis>]=(<Mode>,<TE1>,...,<TE4>)
MEAWA[<Axis>]=(<Mode>,<TE1>,...,<TE4>)
MEAC[<Axis>]=(<Mode>,<MeasMem>,<TE1>,...,<TE4>)
```

定位轴

测量指定轴的定位轴时，带有测量任务的程序行采用以下通用格式：

```
MEAx[<Axis>]=(<Mode>,<TE1>,...,<TE4>)
POS[<Axis>]=...FA[<Axis>]=...
```

几何轴/坐标转换

如果要开始对某个几何轴或某个涉及转换的轴进行单轴测量，就必须也为所有剩余几何轴或所有其他涉及转换的轴编写相同的测量任务。

示例：x 轴的指定轴测量。x、y 和 z 轴是几何轴。

```
MEAx[X]=(<Mode>,<TE1>,...,<TE4>)
MEAx[Y]=(<Mode>,<TE1>,...,<TE4>)
MEAx[Z]=(<Mode>,<TE1>,...,<TE4>) G...X...F...
```

说明

MEASA 和 MEAWA 为逐段有效且可以在某个程序段中共同编程。但如果 MEASA/MEAWA 与 MEAS/MEAW 编程在一个程序段中，就会发出故障信息。

含义

MEASA	轴测量，**带**删除剩余行程
MEAWA	轴测量，**不带**删除剩余行程
MEAC	不带剩余行程删除的轴连续测量
<Axis>	名称，用于测量所使用的通道轴

<Mode>	指定运行模式（测量模式和测量系统）的两位数（xx）	
	十进制个位：测量模式	
	x**0**	中断测量任务。
	x**1**	在一个位置控制器周期中同时最多激活 4 个**不同**的触发事件 按照触发事件出现的顺序对其进行分析（= 时间顺序）。 **提示：** 在此模式下，编程的触发事件必须不同，否则测量任务将中止并显示故障信息。
	x**2**	最多 4 个依次激活的触发事件。 按照编程顺序对触发事件进行分析： <TE1> → <TE2> → <TE3> → <TE4> 该模式用于监控测头。如果测头的偏差状态与测量任务开始时第一个编程触发事件的开关沿相同，则测量任务将中止并输出报警 21703。
	x**3**	与 **2** 一样，但是起动时不监控触发事件 1（报警 21703 被抑制）。 **提示：** 该模式不适用于 MEAC。
	十进制十位：测量系统 确定使用哪个测量系统进行测量。	
	0x（或者不指定）	已激活的测量系统
	1x	测量系统 1
	2x	测量系统 2
	3x	两个测量系统
	提示： 如果只有一个测量系统可用，则自动使用该测量系统（与编程无关）。	

<TE1>,...,<TE4>	触发测量的触发事件	
	类型：	INT
	取值范围：	-2, -1, 1, 2
	(+)1	测头 1 的上升沿
	-1	测头 1 的下降沿
	(+)2	测头 2 的上升沿
	-2	测头 2 的下降沿
	说明： 使用两个测量系统执行测量过程时，最多可编程两个触发事件（上升沿或下降沿）。每个触发事件都会触发两个测量系统的测量值采集。	
<MeasMem>	FIFO 号（循环存储器）	
	类型：	INT

示例

示例 1：

在模式 1 中进行指定轴的定位轴测量，带剩余行程删除（按时间顺序分析）

程序代码	注释
...	
N100 **MEASA[Q]=(1,1,-1) POS[Q]=100 FA[Q]=1000**	
	；使用激活的测量系统在模式 1 中测量。等待测量信号，测头 1 的上升沿/下降沿在 Q = 100 后面的运动行程上。
N110 IF $AC_MEA[1]==FALSE GOTOF ENDE	;检查测量结果。
N120 R10=$AA_MM1[Q]	；保存属于第一个已编程触发事件（上升沿）的测量值。
N130 R11=$AA_MM2[Q]	；保存属于第二个已编程触发事件（下降沿）的测量值。
N140 ENDE:	

示例 2：

在模式 1 中进行指定轴的几何轴测量，带剩余行程删除（按时间顺序分析）

x 轴的指定轴测量。x、y 和 z 轴是几何轴。

a)使用一个测量系统测量

程序代码	注释
...	

程序代码	注释
N100 **MEASA[X]=(1,1,-1) MEASA[Y]=(1,1,-1) MEASA[Z]=(1,1,-1)** G01 X100 F100	
	；使用激活的测量系统在模式 1 中测量。等待测量信号，测头 1 的上升沿/下降沿在 X = 100 后面的运动行程上。
N110 IF $AC_MEA[1]==FALSE GOTOF ENDE	;检查测量结果。
N120 R10=$AA_MM1[X]	；保存属于第一个已编程触发事件（上升沿）的测量值。
N130 R11=$AA_MM2[X]	；保存属于第二个已编程触发事件（下降沿）的测量值。
N140 ENDE:	

b)使用两个测量系统测量

程序代码	注释
...	
N200 **MEASA[X]=(31,1,-1) MEASA[Y]=(31,1,-1) MEASA[Z]=(31,1,-1)** G01 X100 F100	
	；使用激活的测量系统在模式 1 中测量。等待测量信号，测头 1 的上升沿/下降沿在 X = 100 后面的运动行程上。
N210 IF $AC_MEA[1]==FALSE GOTOF ENDE	;检查测量结果。
N220 R10=$AA_MM1[X]	；在出现上升沿时保存测量系统 1 的测量值。
N230 R11=$AA_MM2[X]	；在出现上升沿时保存测量系统 2 的测量值。
N240 R12=$AA_MM3[X]	；在出现下降沿时保存测量系统 1 的测量值。
N250 R13=$AA_MM4[X]	；在出现下降沿时保存测量系统 2 的测量值。
N260 ENDE:	

示例 3：

在模式 2 中进行指定轴的几何轴测量，带剩余行程删除（按编程顺序分析）

x 轴的指定轴测量。x、y 和 z 轴是几何轴。

程序代码	注释
...	
N100 **MEASA[X]=(2,1,-1,2,-2) MEASA[Y]=(2,1,-1,2,-2) MEASA[Z]=(2,1,-1,2,-2)** G01 X100 F100	
	；使用激活的测量系统在模式 2 中测量。在 X = 100 后面的运动行程上，等候测量信号，顺序为测头 1 上升沿，测头 1 下降沿；测头 2 上升沿，测头 2 下降沿。
N110 IF $AC_MEA[1]==FALSE GOTOF MESSTASTER2	；使用测头 1 检查测量结果。
N120 R10=$AA_MM1[X]	；保存属于第一个已编程触发事件（测头 1 上升沿）的测量值。
N130 R11=$AA_MM2[X]	；保存属于第二个已编程触发事件（测头 1 下降沿）的测量值。
N140 MESSTASTER2:	
N150 IF $AC_MEA[2]==FALSE GOTOF ENDE	；使用测头 2 检查测量结果。

程序代码	注释
N160 R12=$AA_MM3[X]	；保存属于第三个已编程触发事件（测头 2 上升沿）的测量值。
N170 R13=$AA_MM4[X]	；保存属于第四个已编程触发事件（测头 2 下降沿）的测量值。
N180 ENDE:	

示例 4：

在模式 1 中进行指定轴的几何轴的持续测量（按时间顺序分析）

x 轴的指定轴测量。x、y 和 z 轴是几何轴。

a) 测量 100 个以下的测量值

程序代码	注释
...	
N110 DEF REAL MESSWERT[100]	
N120 DEF INT 循环=0	
N130 **MEAC[X]=(1,1,-1) MEAC[Y]=(1,1,-1) MEAC[Z]=(1,1,-1)** G01 X1000 F100	
	；使用激活的测量系统在模式 1 中测量。将测量值保存在 $AC_FIFO1 下。等待测量信号，测头 1 的下降沿在 X = 1000 后面的运动行程上。
N135 STOPRE	
N140 MEAC[X]=(0) MEAC[Y]=(0) MEAC[Z]=(0)	；在到达轴位置之后中断测量。
N150 R1=$AC_FIFO1[4]	；将累计测量值的数量保存在参数 R1 中。
N160 FOR 循环=0 TO R1-1	
N170 测量值[循环]=$AC_FIFO1[0]	；从 $AC_FIFO1 中读取并且保存测量值。
N180 ENDFOR	

b) 在 10 个测量值之后使用剩余行程删除来测量

程序代码	注释
...	
N10 WHEN $AC_FIFO1[4]>=10 DO MEAC[x]=(0) DELDTG(x)	
	；10 个测量值之后删除剩余行程。
N20 **MEAC[X]=(1,1,1,-1) MEAC[Y]=(1,1,1,-1) MEAC[Z]=(1,1,1,-1)** G01 X100 F500	
	；使用激活的测量系统在模式 1 中测量。将测量值保存在 $AC_FIFO1 下。等待测量信号，测头 1 的上升沿/下降沿在 X = 100 后面的运动行程上。
N30 MEAC[X]=(0) MEAC[Y]=(0) MEAC[Z]=(0)	
N40 R1=$AC_FIFO1[4]	；测量值的数量。
...	

c) 通过 2 个测头测量上升沿/下降沿

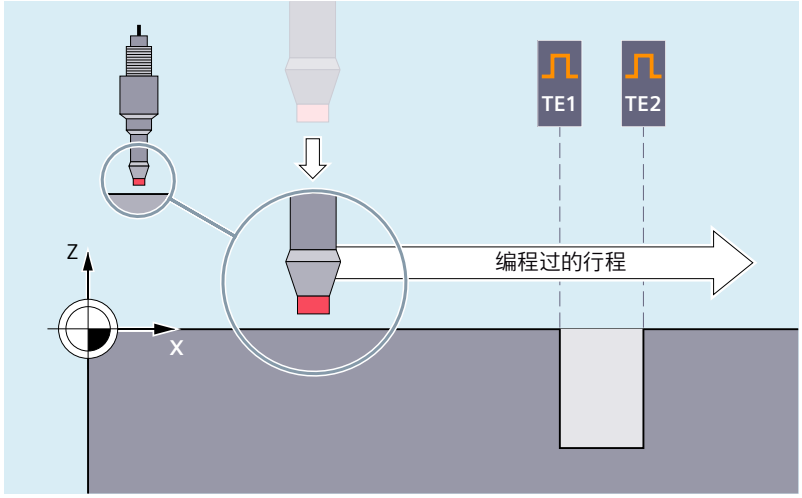
程序代码	注释
...	
N110 DEF REAL MESSWERT[16]	
N120 DEF INT 循环=0	
N130 **MEAC[X]=(1,1,-1,2) MEAC[Y]=(1,1,-1,2) MEAC[Z]=(1,1,-1,2)** G01 X100 F100	
	; 使用激活的测量系统在模式 1 中测量。将测量值保存在 $AC_FIFO1 下。等待测量信号，测头 1 的下降沿/测头 2 的上升沿在 X = 100 后面的运动行程上。
N140 STOPRE	; 预处理停止。
N150 MEAC[X]=(0) MEAC[Y]=(0) MEAC[Z]=(0)	
	; 在到达轴位置之后中断测量。
N160 R1=$AC_FIFO1[4]	; 将累计测量值的数量保存在参数 R1 中。
N170 FOR Schleife=0 TO R1-1	
N180 MESSWERT[Schleife]=$AC_FIFO1[0]	; 从 $AC_FIFO1 中读取并且保存测量值。
N190 ENDFOR	

更多信息

MEASA 或 MEAWA

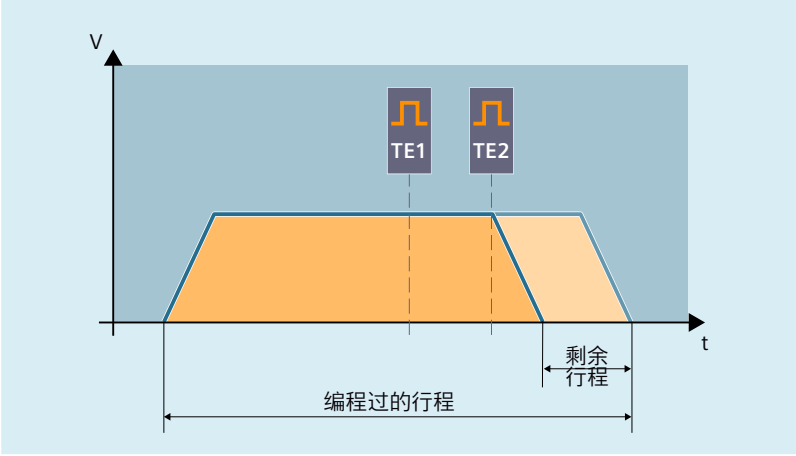
使用 MEASA 或 MEAWA 时，每次测量最多可为相应已编程的轴采集四个测量值，并将测量结果保存在系统变量中。

MEASA/MEAWA 的生效方式如下图所示：



本例中使用 MEASA 或 MEAWA 来测量沿编程的运动行程的 x 轴的钻孔的位置。为此所需的两个触发事件 TE1 到 TE2 在“按照编程的顺序接连”模式下进行分析。将非接触式开关测头用作测头（例如感应测头）。

当使用“测量带删除剩余行程（MEASA）”功能类型时，轴运动在**所有**编程触发事件发生后被制动：



对于在任何情况下均要逼近已编程位置的特殊测量任务而言，使用 MEAWA。

说明

MEASA 在同步动作中不可编程。取而代之的是可以将 MEAWA 加上删除剩余行程作为同步动作编程。

当使用 MEAWA 从同步动作中开始测量任务时，仅机床坐标系中的测量值可供使用。

读取测量结果（MEASA/MEAWA）

MEASA/MEAWA 的测头测量值可在零件程序及同步动作中通过以下系统变量读取：

系统变量	含义
$AA_MM1[<Axis>] ... $AA_MM4[<Axis>]	发生触发事件 1 时 MCS 中的测头测量值 ... 发生触发事件 4 时 MCS 中的测头测量值
$AA_MW1[<Axis>] ... $AA_MW4[<Axis>]	发生触发事件 1 时 WCS 中的测头测量值 ... 发生触发事件 4 时 WCS 中的测头测量值

<Axis> = 测量轴

当使用两个测量系统执行某个测量任务时，系统会采集这两个测量系统上每一个可能发生的这两种触发事件。

系统变量的分配如下：

系统变量	含义
$AA_MM1[<Axis>] 或 $AA_MW1[<Axis>]	当出现触发事件 1 时测量系统 1 的测量值
$AA_MM2[<Axis>] 或 $AA_MW2[<Axis>]	当出现触发事件 1 时测量系统 2 的测量值
$AA_MM3[<Axis>] 或 $AA_MW3[<Axis>]	当出现触发事件 2 时测量系统 1 的测量值
$AA_MM4[<Axis>] 或 $AA_MW4[<Axis>]	当出现触发事件 2 时测量系统 2 的测量值

<Axis> = 测量轴

MEAC

对于连续测量（MEAC），每次发生后重新激活已编程的触发事件。结果是循环重复的开关沿编程和评估。

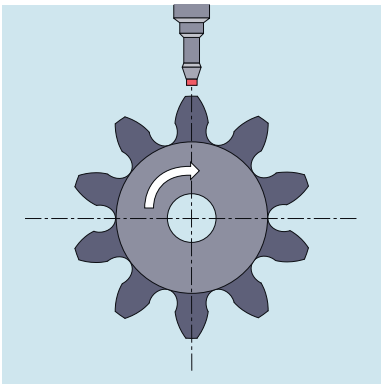
测量值在执行 MEAC 时存在于机床坐标系中并且被保存在指定的 FIFO[<n>] 存储器中。如果设计了两个探头用来进行测量，就会将第二个探头的测量值单独保存在额外为此而设计的 FIFO[<n>+1] 存储器中。

FIFO 是一种循环存储器，按照循环原理将 $AC_FIFO 变量中的测量值记录在该存储器中。内容仅能从循环存储器中读出一次。如果要多次使用测量数据，就必须将其临时保存在用户数据中。

当测量值的数量超过机床数据中为 FIFO 存储器规定的最大值时，就会自动结束测量。

可通过循环读取测量值的方式来实现连续测量。此时必须至少以和新测量值的输入频率相同的频率来进行读取。

MEAC 的一个典型应用示例是相互啮合的工件的测量：



详细信息：功能手册之同步动作

进给率

进给应和相应的测量问题适配。允许的进给率取决于编程触发事件的数量以及插补节拍与位置控制器周期的比率。

如果是 MEASA 和 MEAWA，那么只有当进给不再作为一个相同的触发事件、且不再作为每个位置控制器周期的 4 个不同的触发事件出现时，才能保证结果正确。

如果是带有 MEAC 的连续测量，那么插补周期和位置控制器周期之间的比例不得大于 8:1。

优化

采用缺省设置 PROFIBUS 通讯中的 PROFIBUS 报文 391 时，每个触发事件、每个位置控制器周期只可以采集一个测量值。

在测量模式 1 下执行 MEAC 时，可使用 PROFIBUS 报文 395 将每个触发事件和位置控制器周期的测量值数量按照如下提高：

- 一个测头：每个测头的上升沿 8 个测量值，下降沿 8 个测量值
- 两个测头：每个测头的上升沿 4 个测量值，下降沿 4 个测量值

由此可实现更高的进给率或转速。

查询状态

如果需要在需要在程序中了解测头是否偏转以及是否切换，可以通过以下系统变量查询该状态；

系统变量	含义	数据类型	值	含义
$A_PROBE[<n>]	测头的偏差状态	INT	0	测头未偏转。
			1	测头偏转。
$AC_MEA[<n>]	测头切换状态 测量开始时 $AC_MEA[<n>] 会被自动复位。	INT	0	测头未切换。
			1	测头已切换（发生了测量程序段中编程的所有触发事件）

<n> = 测头编号

说明

当从同步动作中开始测量时，就不再更新 $AC_MEA 。在这种情况下，需要查询 NC/PLC 接口信号 `<Axis>.basic.in.measurementActive` 或者等效变量 $AA_MEAACT[<Axis>]：

$AA_MEAACT==1:测量有效

$AA_MEAACT==0:测量未激活

测头限制

在使用 PROFIBUS 报文 395 时可通过系统变量 $A_PROBE_LIMITED 在 NC 程序或同步动作中读取测头限制的状态：

$A_PROBE_LIMITED[<n>] == 0:测头限制无效/复位

$A_PROBE_LIMITED[<n>] == 1:测头限制生效

<n> = 测头编号

防止错误编程

识别出下面的出错编程，并且显示一个出错：

说明	示例
MEASA/MEAWA 和 MEAS/ MEAW 位于同一条程序段中	`N01 MEAS=1 MEASA[X]=(1,1) G01 F100 POS[X]=100`
MEASA/MEAWA 参数个数 <2 或者 >5	`N01 MEAWA[X]=(1) G01 F100 POS[X]=100`
MEASA/MEAWA 触发事件不等于 1/ -1/ 2/ -2	`N01 MEASA[B]=(1,1,3) B100`
MEASA/MEAWA 模式错误	`N01 MEAWA[B]=(4,1) B100`
MEASA/MEAWA 测量模式 1 重复编写了触发事件	`N01 MEASA[B]=(1,1,-1,2,-1) B100`
MEASA/MEAWA 缺少几何轴	`N01 MEASA[X]=(1,1) MEASA[Y]=(1,1) G01 X50 Y50 Z50 F100` ；几何轴 X/Y/Z
几何轴中测量任务不一致	`N01 MEASA[X]=(1,1) MEASA[Y]=(1,1) MEASA[Z]=(1,1,2) G01 X50 Y50 Z50 F100`

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Axis>.basic.in.measurementActive	LBP_Axis*.E_MeasAct	DB31, ...DBX62.3

报文选择

缺省设置下，单轴测量通过 PROFIBUS 报文 391 实现。如测量需要每个触发事件、每个位置控制周期获取更多测量值时，可使用 PROFIBUS 报文 395。

此外还需进行以下设置：

- 驱动参数：
  CU：p0922 = 395; 报文选择设置
  CU：p0684 = 16; 测量方案设置
  CU：p0680; 中央测头端子的配置
- PROFIBUS 连接：
  MD13211 $MN_MEAS_CENTRAL_SOURCE = 2（不使用握手的报文连接）

10.5 工件或刀具测量

10.5.1 基本原理

10.5.1.1 引言

在 **测量工件**时，测头如同一把刀具向夹紧的工件移动以检测测量值。通过多样化选择不同的测量类型，可顺利解决在铣床或车床上所有通常需解决的测量任务。例如可以测量工件一条边沿、一个拐角或一个钻孔的位置。为确定工件或钻孔的零点，可将实测位置和 WCS 中的设定位置相比。由此得出的偏移可输入到选中的框架中。

设定实际值时，可对 WCS 中的轴设定位置进行修改。计算出的框架将写入系统框架 SETFRAME。

进行**刀具测量**时，可根据一个经过测量的标准件测量刀具的长度或半径。

测量可通过 **HMI 操作或测量循环** 进行。

在 NC 中的计算，将通过 HMI 操作激活 PI 通讯，或通过测量循环中的预定义函数 MEASURE() 来执行。可选择一把刀具或一个平面作为计算的基础。计算出的框架将输入结果框架。

与 NC 的通讯通过预定义系统变量进行。**输入变量**必须通过 HMI 或由循环写入。**输出变量**包含计算结果。

10.5.1.2 输入变量

输入变量的有效位

在进行定义时，每个输入变量将隐性设置 $AC_MEAS_VALID 中的有效性位。若未复位这些有效位，则这些变量值在下一次计算时仍保持生效。因此将在每次测量过程前，将所有输入变量的有效位定义为无效，其中$AC_MEAS_VALID 将设定为数值 0：

$AC_MEAS_VALID = 0

说明

通道重置或程序末端重置不会使有效位复原。

位	系统变量	含义
0	$AA_MEAS_POINT1[<Axis>]	用于所有通道轴的第一个测量点
1	$AA_MEAS_POINT2[<Axis>]	用于所有通道轴的第二个测量点
2	$AA_MEAS_POINT3[<Axis>]	用于所有通道轴的第三个测量点
3	$AA_MEAS_POINT4[<Axis>]	用于所有通道轴的第四个测量点
4	$AA_MEAS_SETPOINT[<Axis>]	轴的设定位置
5	$AC_MEAS_WP_SETANGLE	工件位置的设定角度; -90 < α < 180
6	$AC_MEAS_CORNER_SETANGLE	工件拐角的设定交角 φ：0 < φ < 180
7	$AC_MEAS_T_NUMBER	刀具选择
7	$AC_MEAS_D_NUMBER	刀沿选择
9	$AC_MEAS_DIR_APPROACH	移动到工件上
10	$AC_MEAS_ACT_PLANE	操作平面和进给方向规定
11	$AC_MEAS_FRAME_SELECT	工件测量时的框架选择
12	$AC_MEAS_TYPE	选测测量类型
13	$AC_MEAS_FINE_TRANS	精推移校正
14	$AA_MEAS_SETANGLE[<轴>]	轴的设定角度
15	$AA_MEAS_SCALEUNIT	输入值和输出值的单位
16	$AA_MEAS_TOOL_MASK	刀具位置以及考虑到刀具长度
17	$AA_MEAS_P1_COORD	第 1 测量点的坐标系
18	$AA_MEAS_P2_COORD	第 2 测量点的坐标系
19	$AA_MEAS_P3_COORD	第 3 测量点的坐标系
20	$AA_MEAS_P4_COORD	第 4 测量点的坐标系
21	$AA_MEAS_SET_COORD	设定点的坐标系
22	$AA_MEAS_CHSFR	系统框架的框架选择
23	$AA_MEAS_NCBFR	全局基本框架的框架选择
24	$AA_MEAS_CHBFR	通道基本框架的框架选择
25	$AA_MEAS_UIFR	可设定框架的框架选择

位	系统变量	含义
26	$AA_MEAS_PFRAME	可编程框架的框架选择
27	$AC_MEAS_INPUT[<n>]	测量输入参数
28	$AC_MEAS_GFR	磨削框架的框架选择
29	$AC_MEAS_ORIWKS	测量接口的转型特性

说明

通过以下设置将使相应测量点的所有轴实际值无效：

$AC_MEAS_LATCH = 0

测量点

所有通道轴最多有四个测量点可供使用：

系统变量	含义	数据类型
$AA_MEAS_POINT1[<Axis>]	用于所有通道轴的第一个测量点	REAL
$AA_MEAS_POINT2[<Axis>]	用于所有通道轴的第二个测量点	REAL
$AA_MEAS_POINT3[<Axis>]	用于所有通道轴的第三个测量点	REAL
$AA_MEAS_POINT4[<Axis>]	用于所有通道轴的第四个测量点	REAL

测定的点位置通常作为WCS中的实际值。一旦为测量点写入轴值，该测量点立即标记为“有效”。每个单独的测量点均可直接定义或解锁。

有几种测量类型也支持其他坐标系（BCS、MCS）中的测量点。可通过以下系统变量输入在哪个坐标系中测量测量点：

系统变量	含义	数据类型	值
$AA_MEAS_P1_COORD	第 1 测量点的坐标系	INT	0：WCS (默认) 1:BCS 2:MCS 3：可设置的零点坐标系 4：WKS_REL 5：ENS_REL
$AA_MEAS_P2_COORD	第 2 测量点的坐标系	INT	
$AA_MEAS_P3_COORD	第 3 测量点的坐标系	INT	
$AA_MEAS_P4_COORD	第 4 测量点的坐标系	INT	
$AA_MEAS_SET_COORD	设定点的坐标系	INT	

获取测量点

测量点可通过当前轴实际值进行定义。这些位置基于所选择的坐标系解锁。如未设定坐标系，则会锁存 WCS 中的位置。解锁的测量点将保存在
$AA_MEAS_POINT1[<Axis>] ...$AA_MEAS_POINT4[<Axis>]中。

获取所有轴的测量点

用于测量点的**所有**最新轴实际值将通过系统变量 $AC_MEAS_LATCH[0 ...3] 设定:

系统变量	含义	数据类型	值
$AA_MEAS_LATCH[0]	获取所有轴的第 1 测量点	INT	0：不接受用于测量点的轴实际值 1:接受用于测量点的轴实际值。
$AA_MEAS_LATCH[1]	获取所有轴的第 2 测量点	INT	
$AA_MEAS_LATCH[2]	获取所有轴的第 3 测量点	INT	
$AA_MEAS_LATCH[3]	获取所有轴的第 4 测量点	INT	

获取一个轴的测量点

单个轴实际值的获取将通过系统变量指定 $AA_MEAS_P1_VALID[<Axis>] ... $AA_MEAS_P4_VALID[<Axis>]：

系统变量	含义	数据类型	值
$AA_MEAS_P1_VALID[<Axis>]	获取一个轴的第 1 测量点	INT	0：不接受用于测量点的轴实际值 1:接受用于测量点的轴实际值
$AA_MEAS_P2_VALID[<Axis>]	获取一个轴的第 2 测量点	INT	
$AA_MEAS_P3_VALID[<Axis>]	获取一个轴的第 3 测量点	INT	
$AA_MEAS_P4_VALID[<Axis>]	获取一个轴的第 4 测量点	INT	

设定值

框架的计算会遵循用户给出的设定值。

系统变量	含义	数据类型	值	
$AA_MEAS_SETPOINT[<Axis>]	轴的设定位置	REAL		
$AA_MEAS_SETANGLE[<轴>]	轴的设定角度	REAL		
$AA_MEAS_SP_VALID[<Axis>]	轴设定位置无效/有效	INT	0	轴设定位置无效
			1	轴设定位置有效
$AC_MEAS_WP_SETANGLE	工件位置设定角度	REAL	-90 < α < 180	
$AC_MEAS_CORNER_SETANGLE	拐角的设定交角	REAL	0 < φ < 180	

系统变量	含义	数据类型	值	
$AC_MEAS_DIR_APPROACH *)	移动到工件上	INT	0	+ X
			1	- X
			2	+ Y
			3	- Y
			4	+ Z
			5	- Z

*) 仅在边沿测量、槽测量、隔断测量和刀具测量中需要逼近方向。

以下测量点不相关且不会被分析：

- 用于工件位置设定角输入的第 2 个测量点。
- 设定交角输入的第 4 个测量点。

平面设置

用于确定刀具方向的工作平面通过系统变量$AC_MEAS_ACT_PLANE 给出：

系统变量	含义	数据类型	值	
$AC_MEAS_ACT_PLANE	操作平面和进给方向规定	INT	0	• G17 - 工作平面 X/Y • 横向进给方向 Z
			1	• G18 工作平面 Z/X • 横向进给方向 Y
			2	• G17 工作平面 Y/Z • 横向进给方向 X

如未设定工作平面，则会采用生效的平面。

平移

测量工件时，可将平移输入所选框架的精偏分量。为此需要使用系统变量 $AC_MEAS_FINE_TRANS：

系统变量	含义	数据类型	值	
$AC_MEAS_FINE_TRANS	精推移校正	INT	0	平移补偿输入粗偏。
			1	平移补偿输入精偏。

如变量$AC_MEAS_FINE_TRANS 未定义，则将校正值输入粗移量，并转换为目标框架。通过转换也可能得到平移中的精分量。

如出现以下情况下，则在不受$AC_MEAS_FINE_TRANS 影响的情况下，校正值将始终输入到粗移量中：

MD18600 $MN_MM_FRAME_FINE_TRANS = 0

工件测量时的框架选择

进行工件测量时，计算出的框架会被输入指定框架。通过系统变量 $AC_MEAS_FRAME_SELECT 选择框架。

系统变量	含义	数据类型
$AC_MEAS_FRAME_SELECT	工件测量时的框架选择	INT

系统变量 $AC_MEAS_FRAME_SELECT 可取以下值：

值	框架	含义
0	$P_SETFRAME	生效的系统框架
1	$P_PARTFRAME	生效的系统框架
2	$P_EXTFRAME	生效的系统框架
10 ...25	$P_CHBFRAME[0 ...15]	生效的通道特定基本框架
50 ...65	$P_NCBFRAME[0 ...15]	生效的 NCU 全局基本框架
100 ...199	$P_IFRAME	选择了相应框架时，通过生效的可设定框架进行计算。若选择的框架未生效，则将对应的数据管理框架纳入计算。
500	$P_TOOLFRAME	生效的系统框架
501	$P_WPFRAME	生效的系统框架
502	$P_TRAFRAME	生效的系统框架
503	$P_PFRAME	生效的当前可编程框架
504	$P_CYCFRAME	生效的系统框架
505	$P_RELFRAME（WCS）	生效的系统框架
506	$P_RELFRAME（SZS）	生效的系统框架
1010 ...1025	$P_CHBFRAME[0 ...15]	生效的通道特定基本框架，包含生效的 G500
1050 ...1065	$P_NCBFRAME[0 ...15]	生效的 NCU 全局基本框架，包含生效的 G500
2000	$P_SETFR	数据管理中的系统框架
2001	$P_PARTFR	数据管理中的系统框架

值	框架	含义
2002	$P_EXTFR	数据管理中的系统框架
2010 ... 2025	$P_CHBFR[0 ...15]	数据管理中的通道特定框架
2050 ... 2065	$P_NCBFR[0 ...15]	数据管理中的 NCU 全局基本框架
2100 ... 2199	$P_UIFR[0 ...99]	数据管理中的可设定框架
2500	$P_TOOLFR	数据管理中的系统框架
2501	$P_WPFR	数据管理中的系统框架
2502	$P_TRAFR	数据管理中的系统框架
2504	$P_CYCFR	数据管理中的系统框架
2505	$P_RELFR（WCS）	数据管理中的系统框架
2506	$P_RELFR（SZS）	数据管理中的系统框架
3010 ...3025	$P_CHBFR[0 ...15]	数据管理中带生效 G500 的通道专用基本框架
3050 ...3065	$P_NCBFR[0 ...15]	数据管理中带生效 G500 的 NCU 全局基本框架

使用 MEASURE( ) 功能可根据指定的框架计算框架 $AC_MEAS_FRAME。

对于 **0 至 1065** 的设置 ，可借助生效的框架执行计算。

如 **2000 至 3065** 时，基于所选择的数据管理中的框架执行计算。测量类型 14 和 15 不支持选择数据管理中的框架。选择数据管理中的框架时，此框架必须未生效。在此情形下内部计算会假定其为生效状态。

测量点会被转换至所选择的系统，所选择的框架借助整体框架 （包括所选择的框架）确定。补偿和激活框架后，实际值设定才生效。

对于具有生效的 **G500** (1010 ...1025, 1050 ...1065, 3010 ...3025, 3050 ...3065）的数值，计算框架时会确保 G500 在选择此框架后保持生效，从而能到达设定位置。

换算至另一坐标系

如需将一个位置换算为另一个坐标系中的位置，可通过以下变量设定所需框架链的组合：

系统变量	含义	数据类型	值	
$AC_MEAS_CHSFR	系统框架的框架选择	INT	位标记对应 MD28082 $MC_MM_SYSTEM_FRAME_MASK	
$AC_MEAS_NCBFR	全局基本框架的框架选择	INT	位标记（0 ... FFFF）	
$AC_MEAS_CHBFR	通道基本框架的框架选择	INT	位标记（0 ... FFFF）	
$AC_MEAS_UIFR	可设定框架的框架选择	INT	0 ... 99	
$AC_MEAS_PFRAME	可编程框架的框架选择	INT	0	将包含在计算中。
			1	将不包含在计算中。

针对变量中的相应值，系统会读取数据管理框架和建立新的框架链。

如这些系统变量未设定，则会保留生效的框架。

说明

仅须对需要将数据管理框架加入新框架链的系统变量进行赋值。对于基本框架**只可更换所有**框架，而不是一个特定框架。通过 $P_NCBFRMASK 和 $P_CHBFRMASK 进行的有效修改不计入。

测量输入参数

对于输入用于各种测量循环中的其他输入参数，将采用字段系统变量 $AC_MEAS_INPUT[<n>]:

系统变量	含义	数据类型	值
$AC_MEAS_INPUT[<n>]	测量输入参数	INT	<n> = 0 ...9

测量输入参数的控制作用请见测量方案部分。

刀具和切割选择

生效刀具的刀具号和刀沿号必须与所选择的刀具一致。选择 T0 和 D0 时会采用当前生效的刀具。如无刀具生效，则采用通过 T 和 D 选择的刀具。但是只允许选中的刀具为生效状态。

系统变量	含义	数据类型
$AC_MEAS_T_NUMBER	刀具选择	INT
$AC_MEAS_D_NUMBER	刀沿选择	INT

使用 3D 测头测量

使用 3D 测头进行测量时，刀具半径已通过测量点补偿，因此不允许将此半径再次引入各种测量的计算。此特性可通过系统变量$AC_MEAS_TOOL_MASK 进行指定。

系统变量	含义	数据类型
$AC_MEAS_TOOL_MASK	刀具位置以及考虑到刀具长度	INT

系统变量 $AC_MEAS_TOOL_MASK 可取以下值：

值	含义
0x0：	所有刀具长度均计入（缺省设置）。
0x1	计算时不计入刀具半径。
0x2	刀具位置处于 x 轴方向（G19）。
0x4	刀具位置处于 y 轴方向（G18）。
0x8	刀具位置处于 Z 轴方向（G17）。
0x10	计算时不计入刀具 长度。
0x20	生效刀具的长度计入位置的坐标系转换
0x40	刀具位置处于 x 轴方向。
0x80	刀具位置处于 y 轴方向。
0x100	刀具位置处于 z 轴方向。
0x200	刀具长度差值将采用负值计算。

通过刀具位置和逼近方向可识别出铣刀半径是否计入。若未显性设定逼近方向，可通过选择的平面得出：

工作平面	逼近方向
G17	-Z
G18	-Y
G19	-X

10.5.1.3 选择测量类型

测量类型的选择采用系统变量 $AC_MEAS_TYPE。

系统变量	含义	数据类型
$AC_MEAS_TYPE	选测测量类型	INT

系统变量 $AC_MEAS_TYPE 可取以下值：

值	含义
0	预设
1	测量类型 1：测量 X 边沿
2	测量类型 2：测量 Y 边沿
3	测量类型 3：测量 Z 边沿
4	测量类型 4：测量拐角 C1
5	测量类型 5：测量拐角 C2
6	测量类型 6：测量拐角 C3
7	测量类型 7：测量拐角 C4
8	测量类型 8：测量钻孔
9	测量类型 9：测量曲轴
10 *)	测量类型 10：测量刀具长度
11 *)	测量类型 11：测量刀具直径
12	测量类型 12：测量开槽
13	测量类型 13：隔片的测量
14	测量类型 14：用于几何轴和辅助轴的实际值设定
15	测量类型 15：只针对辅助轴的实际值设定
16	测量类型 16：测量斜边

值	含义
17	测量类型 17：测量倾斜平面中的角度
18	测量类型 18：在倾斜平面上重新定义 WCS 坐标系
19	测量类型 19：1 维设定值给定
20	测量类型 20：2 维设定值给定
21	测量类型 21：3 维设定值给定
22 *)	测量类型 22：ShopTurn-通过放大镜功能测量刀具长度
23 *)	测量类型 23：ShopTurn-通过标记位置或当前位置测量刀具长度
24	测量类型 24：测量点坐标轴转换
25	测量类型 25：测量矩形
26	测量类型 26：备份数据管理框架
27	测量类型 27：恢复已备份的数据管理框架
28	测量类型 28：用于锥形体车削的附加车削定义

*) 刀具测量类型

单个测量类型详细说明参见章节 "工件测量类型 (页 680)" 或 "刀具测量的测量类型 (页 721)" 并通过合适的编程示例进行详细解释。

10.5.1.4 激活计算

通过 HMI 操作界面激活

计算通过 HMI 的 PI 服务 _N_SETUDT 激活。此 PI 服务支持以下参数类型：

参数类型	含义
1	生效的刀具补偿
2	生效的基本框架
3	被激活的可设定框架
4	全局基本框架
5	全局可设定框架
6	计算工件零点或刀具长度
7	激活工件零点（写入对刀）
8	激活外部零点偏移
9	生效的刀架，激活 TCOABS 和 PAROT

在重置状态下，更改将立即可见。在停止状态下，框架将在下一次启动时移出。

说明

PI 服务只可在复位或停止状态下执行。

在进行工件测量时，计算出的框架将通过参数类型 7 立即激活。

测量刀具时，不允许通过参数类型 7 发出 PI 服务，因为不必激活零点。

在停止状态下激活

新的 WCS 位置会在停止状态下更新。通过继续部件程序，将删除中断程序段的剩余路径。将从当前位置移动到下一个程序段的末端位置。由此在停止状态下也可在 MDA 模式或部件程序中启动主轴，并通过 M0 执行实际值设定和对刀。但也可实施另一个测量。

测量循环

测量循环中的计算通过预定义功能 MEASURE 进行激活（）。

MEASURE() 会输出一个结果框架，其可通过 $AC_MEAS_FRAME 读取：结果是由设定值得出的平移和旋转，并换算为所选择的框架。通过计算结果框架，使交联的总和框架等同于通过计算出的平移和旋转量对整个框架进行交联（测量前）。

注意
无预处理限制 MEASURE() 不会触发隐性的预处理停止。MEASURE( ) 和预运行程序段框架一起使用。用户因此必须决定是否需要在计算前停止预运行。

说明

如未选择框架，则将不对计算出的框架进行转换。即平移和旋转量通过给出的设定值和计算出的边沿、拐角、开槽等位置得出。功能的多次使用将始终增加到结果框架。必要时须提前删除结果框架。

测量过程的同步

测量变量在每个通道中仅出现一次。可通过停止状态和复位状态下的操作执行测量。在停止状态下，操作过程可与测量循环重叠。为了保护相互重叠，将采用系统变量 $AC_MEAS_SEMA（Semaphore 用于测量界面）。

Semaphor 变量 $AC_MEAS_SEMA 将由测量循环在循环开始时赋值为 1，并在循环结束时重新复位到 0。

当$AC_MEAS_SEMA 赋值为 1 时，HMI 不使用测量接口。

10.5.1.5 输出变量

结果计算

如设定位置已指定，得出的框架会输出到结果框架 $AC_MEAS_FRAME 中。可在部件程序中读取和写入此框架。结果框架将根据选择的框架进行计算。

如未选择框架，则通过结果框架得出 WCS 中产生的平移和旋转。通过 PI 服务 _N_SETUDT 和参数类型 7， 可将此框架输入到选择的框架中。输入此框架后，结果框架将被清除。

以下输出变量可用于计算结果：

系统变量	含义	数据类型
$AC_MEAS_FRAME	结果框架	FRAME
$AC_MEAS_WP_ANGLE	计算出的工件位置角 α	REAL
$AC_MEAS_CORNER_ANGLE	计算出的交角 ϕ	REAL
$AC_MEAS_DIAMETER	计算出的直径	REAL
$AC_MEAS_TOOL_LENGTH	计算出的刀具长度	REAL
$AC_MEAS_RESULTS[<n>]	计算结果 对哪些字段元素进行赋值将取决于测量类型 ($AC_MEAS_TYPE) 。	DOUBLE

10.5.1.6 单位制

英制/公制测量系统

下列输入和输出变量通过英制（INCH）或公制（METRISCH）单位进行评估：

系统变量	含义
$AA_MEAS_POINT1[<Axis>]	输入变量：1.用于所有通道轴的测量点
$AA_MEAS_POINT2[<Axis>]	输入变量：2.用于所有通道轴的测量点
$AA_MEAS_POINT3[<Axis>]	输入变量：3.用于所有通道轴的测量点
$AA_MEAS_POINT4[<Axis>]	输入变量：4.用于所有通道轴的测量点
$AA_MEAS_SETPOINT[<Axis>]	输入变量：轴的设定位置
$AC_MEAS_DIAMETER	输出变量：计算出的直径
$AC_MEAS_TOOL_LENGTH	输出变量：计算出的刀具长度
$AC_MEAS_RESULTS[<n>]	输出变量：计算结果

读取或写入输入值和输出值时采用的单位可通过以下系统变量$AC_MEAS_SCALEUNIT 进行设置。

系统变量	含义	数据类型	值	
$AC_MEAS_SCALEUNIT	输入值和输出值的单位	INT	0	尺寸单位将参照组 13 的生效 G 功能。 • 对于生效的 G70/G700: 英制 • 对于生效的 G71/G710: 公制
			1	尺寸单位符合设定（标准设置）

如$AC_MEAS_SCALEUNIT 不写入，则数值 1 总是作为缺省设置。

示例：

此示例应遵循以下条件：

- $AC_MEAS_SCALEUNIT = 0
- 基本单位制：公制

程序代码	注释
G70	
$AC_MEAS_POINT1[X]=$AA_IW[X]	; $AA_IW[x] 提供基本单位制
$AC_MEAS_POINT1[X]=10	; 10 mm
G71	
$AC_MEAS_POINT1[X]=$AA_IW[X]	; $AA_IW[x] 提供基本单位制
$AC_MEAS_POINT1[X]=10	; 10 mm
G700	
$AC_MEAS_POINT1[X]=$AA_IW[X]	; $AA_IW[x] 提供英制值
$AC_MEAS_POINT1[X]=10	; 10 英寸
G710	
$AC_MEAS_POINT1[X]=$AA_IW[X]	; $AA_IW[x] 提供公制值
$AC_MEAS_POINT1[X]=10	; 10 mm

直径编程

直径编程通过以下机床数据设置：

- MD20100 $MC_DIAMETER_AX_DEF（带端面轴功能的几何轴）
- MD20150 $MC_GCODE_RESET_VALUES（G 功能组的初始设置）
- MD20360 $MC_TOOL_PARAMETER_DEF_MASK（定义刀具参数）

示例：

```
MD20100 $MC_DIAMETER_AX_DEF = "X"          ；平面轴为 X。
MD20150 $MC_GCODE_RESET_VALUES[28] = 2     ; DIAMON
MD20360 $MC_TOOL_PARAMETER_DEF_MASK =      ；刀具长度、框架和 直径实际值
'B1001010'
```

注意以下事项：

- MCS 中的轴位置不会被作为直径值。
- 计算出的刀具补偿和框架分量与生效的 G 指令 DIAMON 或 DIAMOF 无关。
- 测量位置和设定位置根据 DIAMON 进行 读写。
- 框架中的平移被作为平面轴中的直径值。

计算和显示精度

单位为毫米、英寸或度的位置值以小数点后 6 位精确计算和显示。

10.5.1.7 诊断

故障信息

PI 服务 _N_SETUDT

如客户端通过 DIAGN:errCodeSetNrGent 或 DIAGN:errCodeSetNrPi 已登录，则 PI_SETUDT 将提供以下错误代码：

故障代码	含义
EX_ERR_PI_REJ_**MEASOK**	计算正确。
EX_ERR_PI_REJ_**MEASNOTYPE**	测量类型未指定。
EX_ERR_PI_REJ_**MEASTOOLERROR**	刀具测定故障。

故障代码	含义
EX_ERR_PI_REJ_**MEASNOPOINT1**	测量点 1 不存在。
EX_ERR_PI_REJ_**MEASNOPOINT2**	测量点 2 不存在。
EX_ERR_PI_REJ_**MEASNOPOINT3**	测量点 3 不存在。
EX_ERR_PI_REJ_**MEASNOPOINT4**	测量点 4 不存在。
EX_ERR_PI_REJ_**MEASNOSPECPOINT**	无参考点存在。
EX_ERR_PI_REJ_**MEASNODIR**	无移动靠近方向。
EX_ERR_PI_REJ_**MEASEQUALPOINTS**	测量点相同。
EX_ERR_PI_REJ_**MEASWRONGALPHA**	Alpha (α) 错误。
EX_ERR_PI_REJ_**MEASWRONGPHI**	Phi ϕ 错误。
EX_ERR_PI_REJ_**MEASWRONGDIR**	移动靠近方向错误。
EX_ERR_PI_REJ_**MEASNOCROSSING**	直线不相交。
EX_ERR_PI_REJ_**MEASNOPLANE**	平面不存在。
EX_ERR_PI_REJ_**MEASWRONGFRAME**	未选择框架或选择了错误框架。
EX_ERR_PI_REJ_**MEASNOMEMORY**	存储空间不足。
EX_ERR_PI_REJ_**MEASINTERNALERROR**	内部错误。

若客户未登录，则总是生成汇总故障编号 0xD003。

MEASURE( )

通过预定义的功能 MEASURE( ) 将输出以下返回值：

编号	返回值	含义
0	MEAS_OK	计算正确。
1	MEAS_NO_TYPE	测量类型未指定。
2	MEAS_TOOL_ERROR	刀具测定故障。
3	MEAS_NO_POINT1	测量点 1 不存在。
4	MEAS_NO_POINT2	测量点 2 不存在。
5	MEAS_NO_POINT3	测量点 3 不存在。
6	MEAS_NO_POINT4	测量点 4 不存在。
7	MEAS_NO_SPECPOINT	无参考点存在。
8	MEAS_NO_DIR	无移动靠近方向。
9	MEAS_EQUAL_POINTS	测量点相同。
10	MEAS_WRONG_ALPHA	Alpha (α) 错误。

编号	返回值	含义
11	MEAS_WRONG_PHI	Phi ϕ 错误。
12	MEAS_WRONG_DIR	移动靠近方向错误。
13	MEAS_NO_CROSSING	直线不相交。
14	MEAS_NO_PLANE	平面不存在。
15	MEAS_WRONG_FRAME	未选择框架或选择了错误框架。
16	MEAS_NO_MEMORY	存储空间不足。
17	MEAS_INTERNAL_ERROR	内部错误。

刀具测定故障

出现故障代码 EX_ERR_PI_REJ_MEASTOOLERROR 或 MEAS_TOOL_ERROR 时，系统会将更精确的故障信息连同以下值保存至输出变量 $AC_MEAS_TOOL_LENGTH：

编号	返回值	含义
1	TOOL_NO_BLOCK	无程序段可用于刀具计算。
2	TOOL_WRONG_T_NUMBER	T 号错误。
3	TOOL_WRONG_D_NUMBER	D 号错误。
4	TOOL_EVAL_WRONG_TYPE	刀具不存在。
5	TOOL_NO_TOOLCORR_BODY	内存问题。
6	TOOL_DATA_READ_ERROR	读取刀具数据时出错。
7	TOOL_NO_TOOL_WITH_TRAFO	转换生效时未选择刀具。

其他诊断方式

协议

对于现有的数据"/_N_MPF_DIR/_N_MEAS_DUMP_MPF"，将一条记录写入数据，以便能复制问题。为了使用记录功能，还必须事先使用名称"_N_MEAS_DUMP_MPF" 在目录"/_N_MPF_DIR"中创建一个空数据。文件的内容将一直保留，直至通过 $AC_MEAS_VALID = 0 清除。

跟踪

出于运行时间方面的原因，应只对已出现过一次的问题启用跟踪功能。

10.5.2 工件测量类型

10.5.2.1 测量类型 1, 2, 3：测量边沿

功能

夹紧工件的棱边如下测量：逼近此棱边，然后通过一把已知的刀具执行测量。

测量类型 1：测量 X 边沿

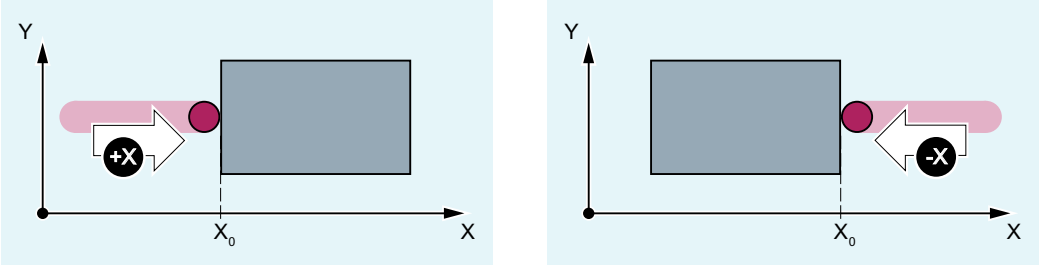


图 10-1　测量 X 边沿

用于测量类型 1，将对以下系统变量数值进行评估：

输入变量	含义	
$AC_MEAS_VALID	输入变量的有效位	
$AA_MEAS_POINT1[<Axis>]	所有通道轴的第 1 测量点	
$AA_MEAS_SETPOINT[<Axis>] *)	X 边沿设定位置	
$AC_MEAS_DIR_APPROACH	= 0	+ X
	= 1	- X
$AC_MEAS_ACT_PLANE *)	未设定时将采用当前生效的平面。 刀具半径仅用于 G17 和 G18。	
$AC_MEAS_FINE_TRANS *)	= 0	粗偏
	= 1	精偏
$AC_MEAS_FRAME_SELECT *)	未设定时将计算叠加框架 。	
$AC_MEAS_T_NUMBER *)	未设定时将采用当前生效的 T（T0）进行计算。	
$AC_MEAS_D_NUMBER *)	未设定时将采用当前生效的 D（D0）进行计算。	
$AC_MEAS_TYPE	= 1	测量类型 1：测量 X 边沿

*) 可选配

对于测量类型 1，将写入以下输出变量：

输出变量	含义
$AC_MEAS_FRAME	结果框架，包含平移
$AC_MEAS_RESULTS[0]	被测棱边的位置

测量类型 2：测量 Y 边沿

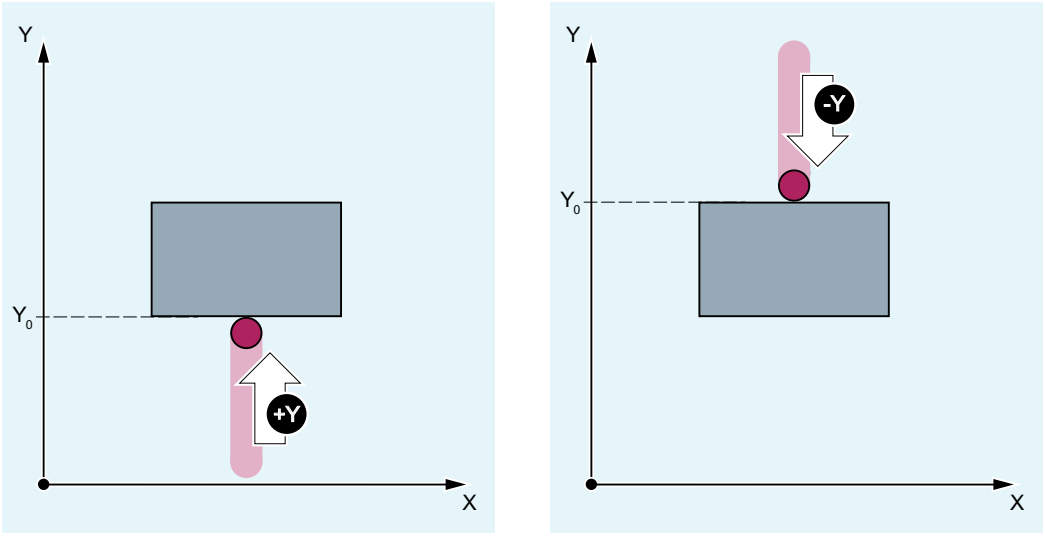


图 10-2　测量 Y 边沿

用于测量类型 2，将对以下系统变量数值进行评估：

输入变量	含义	
$AC_MEAS_VALID	输入变量的有效位	
$AA_MEAS_POINT1[<Axis>]	所有通道轴的第 1 测量点	
$AA_MEAS_SETPOINT[<Axis>] *)	Y 边沿的设定位置	
$AC_MEAS_DIR_APPROACH	= 2	+ Y
	= 3	- Y
$AC_MEAS_ACT_PLANE *)	未设定时将采用当前生效的平面。 刀具半径仅用于 G17 和 G19。	
$AC_MEAS_FINE_TRANS *)	= 0	粗偏
	= 1	精偏
$AC_MEAS_FRAME_SELECT *)	未设定时将计算叠加框架。	
$AC_MEAS_T_NUMBER *)	未设定时将采用当前生效的 T（T0）进行计算。	

输入变量	含义	
$AC_MEAS_D_NUMBER *)	未设定时将采用当前生效的 D（D0）进行计算。	
$AC_MEAS_TYPE	= 2	测量类型 2：测量 Y 边沿

*) 可选配

对于测量类型 2，将写入以下输出变量：

输出变量	含义
$AC_MEAS_FRAME	结果框架，包含平移
$AC_MEAS_RESULTS[0]	被测棱边的位置

测量类型 3：测量 Z 边沿

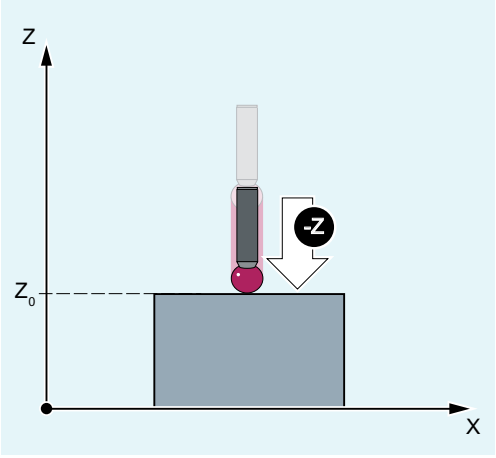


图 10-3　测量 Z 边沿

用于测量类型 3，将对以下系统变量数值进行评估：

输入变量	含义	
$AC_MEAS_VALID	输入变量的有效位	
$AA_MEAS_POINT1[<Axis>]	所有通道轴的第 1 测量点	
$AA_MEAS_SETPOINT[<Axis>] *)	Z 边沿的设定位置	
$AC_MEAS_DIR_APPROACH	= 4	+ Z
	= 5	- Z
$AC_MEAS_ACT_PLANE *)	未设定时将采用当前生效的平面。 刀具半径仅用于 G18 和 G19。	
$AC_MEAS_FINE_TRANS *)	= 0	粗偏
	= 1	精偏

输入变量	含义	
$AC_MEAS_FRAME_SELECT *)	未设定时将计算叠加框架。	
$AC_MEAS_T_NUMBER *)	未设定时将采用当前生效的 T（T0）进行计算。	
$AC_MEAS_D_NUMBER *)	未设定时将采用当前生效的 D（D0）进行计算。	
$AC_MEAS_TYPE	= 3	测量类型 3：测量 Z 边沿

*) 可选配

对于测量类型 3，将写入以下输出变量：

输出变量	含义
$AC_MEAS_FRAME	结果框架，包含平移
$AC_MEAS_RESULTS[0]	被测棱边的位置

示例

测量类型 1：测量 X 边沿

程序代码	注释
DEF INT RETVAL	
DEF FRAME TMP	
$TC_DP1[1,1]=120	；类型
$TC_DP2[1,1]=20	; 0
$TC_DP3[1,1]= 10	；(Z) 长度补偿矢量
$TC_DP4[1,1]=0	; (Y)
$TC_DP5[1,1]= 0	; (X)
$TC_DP6[1,1]=2	；半径
T1 D1	
G0 X0 Y0 Z0 F10000	
G54	
	；测量 x 边沿。
$AC_MEAS_VALID=0	；将所有输入值设置为无效。
G1 X-1 Y-3	；移动到第 1 测量点
$AA_MEAS_POINT1[X]=$AA_IW[X]	
$AA_MEAS_POINT1[Y]=$AA_IW[Y]	
$AA_MEAS_POINT1[Z]=$AA_IW[Z]	

程序代码	注释
$AC_MEAS_DIR_APPROACH=0	; 设置移动方向 +X。
$AA_MEAS_SETPOINT[X]=0 $AA_MEAS_SETPOINT[Y]=0 $AA_MEAS_SETPOINT[Z]=0	; 设置边沿的设定位置。
$AC_MEAS_ACT_PLANE=0	; 测量平面为 G17。
$AC_MEAS_FRAME_SELECT=101	; 选择框架：IFRAME
$AC_MEAS_T_NUMBER=1 $AC_MEAS_D_NUMBER=1	; 选择刀具。
$AC_MEAS_TYPE=1	; 设置 X 边沿测量类型。
RETVAL=MEASURE()	; 开始测量。
IF RETVAL <> 0 SETAL(61043, << RETVAL) ENDIF	
$P_IFRAME=$AC_MEAS_FRAME	
$P_UIFR[1]=$P_IFRAME	; 将系统框架写入数据存储装置。
G1 X0 Y0	**; 移动靠近边沿。**
M30	

10.5.2.2 测量类型 4、 5、 6、 7：测量拐角

功能

通过逼近 4 个测量点 P1 至 P4 来测定唯一的拐角。交角 ϕ 已知时只需要 3 个测量点。

若交角 ϕ 和工件位置角 α 已知，则只需要 P1 和 P3 这 2 个测量点。

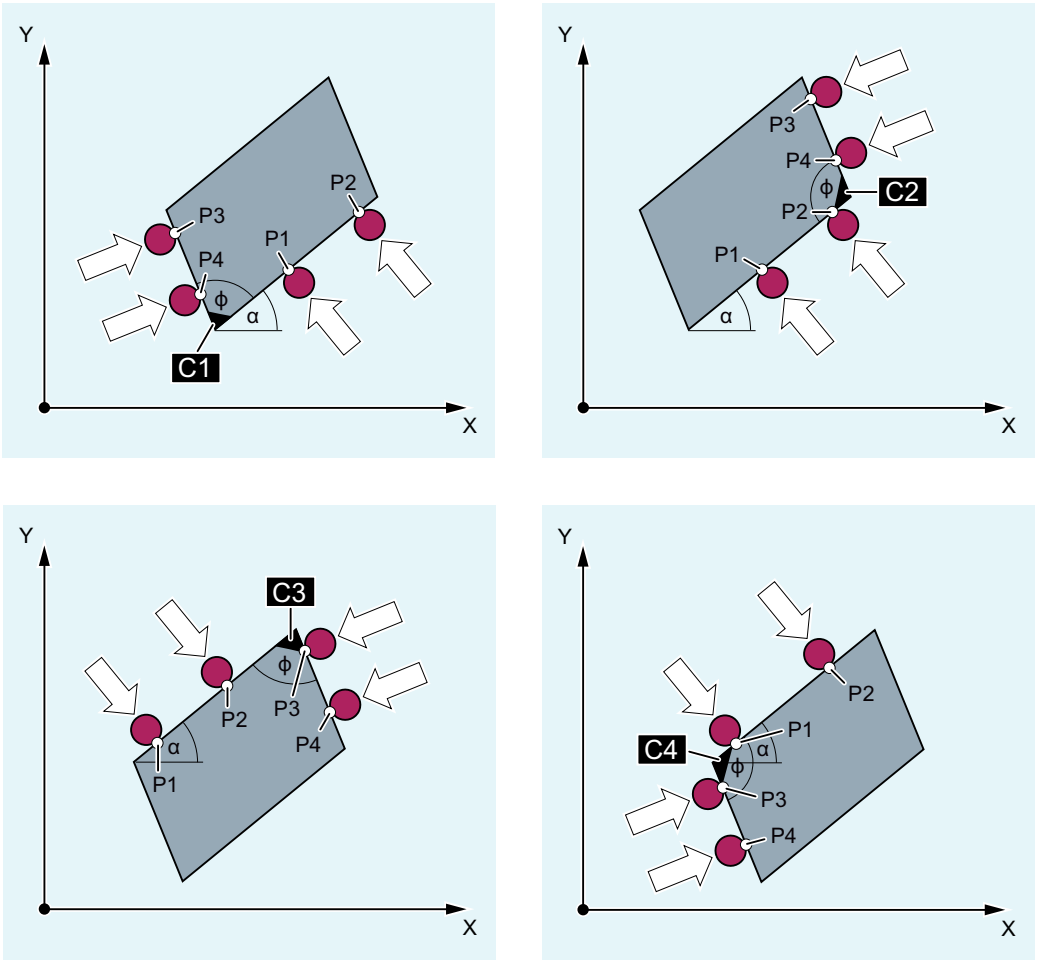


图 10-4　测量拐角 (C1、C2、C3 或 C4)

对于测量类型 4 至 7，将对以下系统变量进行分析：

输入变量	含义
$AC_MEAS_VALID	输入变量的有效位
$AA_MEAS_POINT1[<Axis>]	测量点 1
$AA_MEAS_POINT2[<Axis>]	测量点 2 （不适用于 $AC_MEAS_WP_SETANGLE ）
$AA_MEAS_POINT3[<Axis>]	测量点 3
$AA_MEAS_POINT4[<Axis>]	测量点 4（不适用于 $AC_MEAS_CORNER_SETANGLE）
$AC_MEAS_WP_SETANGLE *)	设定工件位置角
$AC_MEAS_CORNER_SETANGLE *)	设定交角
$AA_MEAS_SETPOINT[<Axis>] *)	拐角设定位置
$AC_MEAS_ACT_PLANE *)	未设定时将采用当前生效的平面。

输入变量	含义	
$AC_MEAS_FINE_TRANS *)	= 0	粗偏
	= 1	精偏
$AC_MEAS_FRAME_SELECT *)	未设定时将计算叠加框架。	
$AC_MEAS_T_NUMBER *)	未设定时将采用当前生效的 T（T0）进行计算。	
$AC_MEAS_D_NUMBER *)	未设定时将采用当前生效的 D（D0）进行计算。	
$AC_MEAS_INPUT[0] *)	= 0	外角测量
	= 1	内角测量
	未定义时将适用以下说明：外角测量	
$AC_MEAS_TYPE	= 4	测量类型 4：测量拐角 C1
	= 5	测量类型 5：测量拐角 C2
	= 6	测量类型 6：测量拐角 C3
	= 7	测量类型 7：测量拐角 C4

*) 可选配

对于测量类型 4 至 7，将写入以下输出变量：

输出变量	含义
$AC_MEAS_FRAME	结果框架，包含平移和旋转
$AC_MEAS_WP_ANGLE	计算出的工件位置角
$AC_MEAS_CORNER_ANGLE	计算出的交角
$AC_MEAS_RESULTS[0]	计算出的拐角点的横坐标
$AC_MEAS_RESULTS[1]	计算出的拐角点的纵坐标
$AC_MEAS_RESULTS[2]	计算出的拐角点的垂直坐标

示例

测量类型 4：测量拐角 C1

拐角具有 3 个测量点（P1、P3 和 P4），已知交角 ϕ（90°），未知工件位置角 α。

程序代码	注释
DEF INT RETVAL	
DEF FRAME TMP	
$TC_DP1[1,1]=120	; 类型
$TC_DP2[1,1]=20	; 0

程序代码	注释
$TC_DP3[1,1]= 10	; (Z) 长度补偿矢量
$TC_DP4[1,1]=0	; (Y)
$TC_DP5[1,1]= 0	; (X)
$TC_DP6[1,1]=2	; 半径
T1 D1	
G0 X0 Y0 Z0 F10000	
G54	
$P_CHBFRAME[0]=CROT(Z,45)	
$P_IFRAME[x,tr]= -SIN(45)	
$P_IFRAME[y,tr]= -SIN(45)	
$P_PFRAME[z,tr]= -45	
	; 通过 3 个测量点测量拐角：
$AC_MEAS_VALID=0	; 将所有输入值设置为无效。
G1 X-1 Y-3	; 移动到第 1 测量点
$AC_MEAS_LATCH[0]=1	; 获取测量点 P1。
G1 X-4 Y4	; 移动到第 3 测量点
$AC_MEAS_LATCH[2]=1	; 获取测量点 P3。
G1 X-4 Y1	; 移动到第 4 个测量点
$AC_MEAS_LATCH[3]=1	; 获取测量点 P4。
$AA_MEAS_SETPOINT[X]=0	; 将拐角设定位置设为 (0,0,0)。
$AA_MEAS_SETPOINT[Y]=0	
$AA_MEAS_SETPOINT[Z]=0	
$AC_MEAS_CORNER_SETANGLE=90	; 给定设定交角 ϕ。
$AC_MEAS_ACT_PLANE=0	; 测量平面为 G17。
$AC_MEAS_FRAME_SELECT=0	; 选择框架：SETFRAME
$AC_MEAS_T_NUMBER=1	; 选择刀具。
$AC_MEAS_D_NUMBER=1	
$AC_MEAS_TYPE=4	; 将测量类型设置为拐角 1。
RETVAL=MEASURE()	; 开始测量。
IF RETVAL <> 0	
SETAL(61043, << RETVAL)	
ENDIF	

程序代码	注释
IF $AC_MEAS_CORNER_ANGLE <> 90 SETAL(61043, << $AC_MEAS_CORNER_ANGLE) ENDIF	；查询已知设定交角 ϕ。
$P_SETFRAME=$AC_MEAS_FRAME	
$P_SETFR=$P_SETFRAME	；将系统框架写入数据存储装置。
G1 X0 Y0	**；逼近拐角。**
G1 X10 Y10 X0 Y0	；离开直角运行。
M30	

10.5.2.3 测量类型 8：测量钻孔

功能

确定圆心和直径需要 3 个测量点。这三个点必须为不同的点。设定了 4 个点时，系统会根据最小误差平方法对圆弧进行调整。圆弧的确定依据以下原则：使圆弧的点距离平方总和最小化。可读取调整质量。

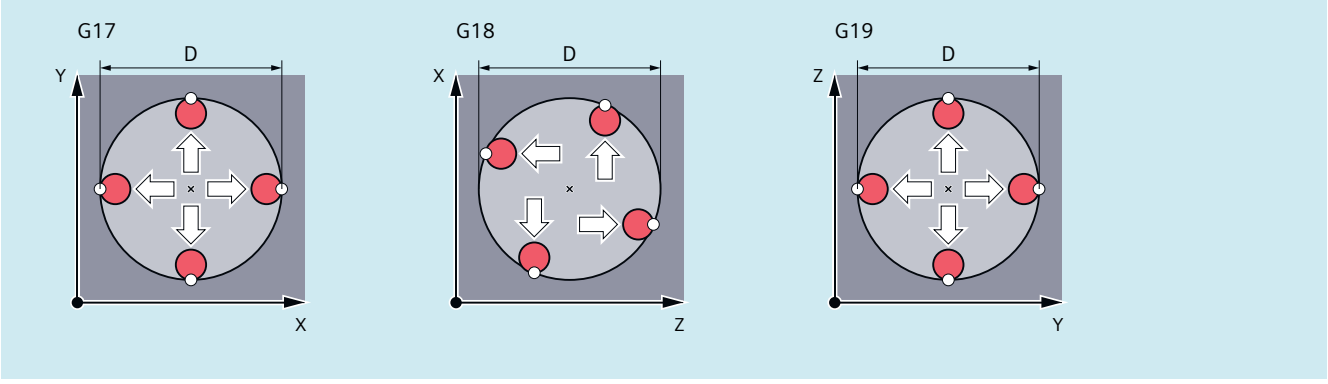


图 10-5 测量钻孔

用于测量类型 8，将对以下系统变量数值进行评估：

输入变量	含义	
$AC_MEAS_VALID	输入变量的有效位	
$AA_MEAS_POINT1[<Axis>]	测量点 1	
$AA_MEAS_POINT2[<Axis>]	测量点 2	
$AA_MEAS_POINT3[<Axis>]	测量点 3	
$AA_MEAS_POINT4[<Axis>] *)	设定此变量时将通过 4 个点确定中心。	
$AA_MEAS_SETPOINT[<Axis>] *)	钻孔中心点设定位置	
$AC_MEAS_ACT_PLANE *)	未设定时将采用当前生效的平面。	
$AC_MEAS_FINE_TRANS *)	= 0	粗偏
	= 1	精偏
$AC_MEAS_FRAME_SELECT *)	未设定时将计算叠加框架 。	
$AC_MEAS_T_NUMBER *)	未设定时将采用当前生效的 T（T0）进行计算。	
$AC_MEAS_D_NUMBER *)	未设定时将采用当前生效的 D（D0）进行计算。	
$AC_MEAS_TYPE	= 8	测量类型 8：测量钻孔

*) 可选配

对于测量类型 8，将写入以下输出变量：

输出变量	含义
$AC_MEAS_FRAME	结果框架，包含平移
$AC_MEAS_DIAMETER	钻孔直径
$AC_MEAS_RESULTS[0]	计算出的中心的横坐标
$AC_MEAS_RESULTS[1]	计算出的中心的纵坐标
$AC_MEAS_RESULTS[2]	计算出的中心的垂直坐标
$AC_MEAS_RESULTS[3]	用于圆形调整的量化尺寸间隔平方数总和

示例

程序代码	注释
DEF INT RETVAL	
DEF FRAME TMP	
$TC_DP1[1,1]=120	；类型

程序代码	注释
$TC_DP2[1,1]=20	; 0
$TC_DP3[1,1]= 10	; (Z) 长度补偿矢量
$TC_DP4[1,1]=0	; (Y)
$TC_DP5[1,1]= 0	; (X)
$TC_DP6[1,1]=2	; 半径
T1 D1	
G0 X0 Y0 Z0 F10000	
G54	
	; 测量钻孔
$AC_MEAS_VALID=0	; 将所有输入值设置为无效。
G1 X-3 Y0	; 移动到第 1 测量点
$AA_MEAS_POINT1[X]=$AA_IW[X]	
$AA_MEAS_POINT1[Y]=$AA_IW[Y]	
$AA_MEAS_POINT1[Z]=$AA_IW[Z]	
G1 X0 Y3	; 第 2 测量点
$AA_MEAS_POINT2[X]=$AA_IW[X]	
$AA_MEAS_POINT2[Y]=$AA_IW[Y]	
$AA_MEAS_POINT2[Z]=$AA_IW[Z]	
G1 X3 Y0	; 移动到第 3 测量点
$AA_MEAS_POINT3[X]=$AA_IW[X]	
$AA_MEAS_POINT3[Y]=$AA_IW[Y]	
$AA_MEAS_POINT3[Z]=$AA_IW[Z]	
$AA_MEAS_SETPOINT[X]=0	; 设置中心的设定位置。
$AA_MEAS_SETPOINT[Y]=0	
$AA_MEAS_SETPOINT[Z]=0	
$AC_MEAS_ACT_PLANE=0	; 测量平面为 G17。
$AC_MEAS_FRAME_SELECT=0	; 选择框架 - SETFRAME
$AC_MEAS_T_NUMBER=1	; 选择刀具。
$AC_MEAS_D_NUMBER=1	
$AC_MEAS_TYPE=8	**; 将测量类型设置为“钻孔”。**
RETVAL=MEASURE()	; 开始测量。
If RETVAL <> 0	
SETAL(61043, << RETVAL)	
ENDIF	

程序代码	注释
IF $AC_MEAS_DIAMETER <> 10 SETAL(61043, << $AC_MEAS_WP_ANGLE) ENDIF	; 查询已知直径。
$P_SETFRAME=$AC_MEAS_FRAME	
$P_SETFR=$P_SETFRAME	; 写入数据管理中的系统框架
G1 X-3 Y0	**; 移动靠近 P1。**
G2 I=$AC_MEAS_DIAMETER/2	; 基于圆心沿钻孔运行。
M30	

10.5.2.4 测量类型 9：测量曲轴

功能

确定圆心和直径需要 3 个测量点。这三个点必须为不同的点。设定了 4 个点时，系统会根据最小误差平方法对圆弧进行调整。圆弧的确定依据以下原则：使圆弧的点距离平方总和最小化。可读取调整质量。

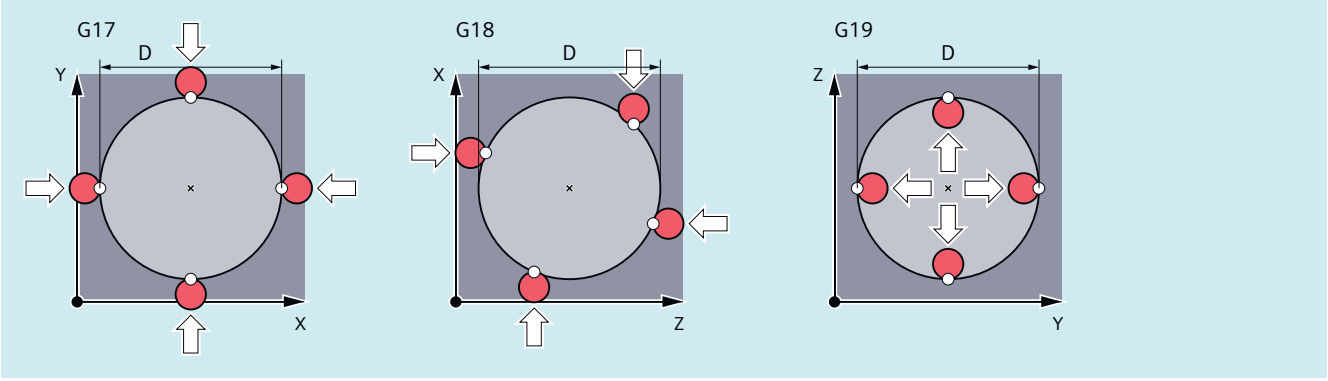


图 10-6 测量曲轴

用于测量类型 9，将对以下系统变量数值进行评估：

输入变量	含义
$AC_MEAS_VALID	输入变量的有效位
$AA_MEAS_POINT1[<Axis>]	测量点 1

输入变量	含义	
$AA_MEAS_POINT2[<Axis>]	测量点 2	
$AA_MEAS_POINT3[<Axis>]	测量点 3	
$AA_MEAS_POINT4[<Axis>] *)	设定此变量时将通过 4 个点确定中心。	
$AA_MEAS_SETPOINT[<Axis>] *)	曲轴中心点设定位置	
$AC_MEAS_ACT_PLANE *)	未设定时将采用当前生效的平面。	
$AC_MEAS_FINE_TRANS *)	= 0	粗偏
	= 1	精偏
$AC_MEAS_FRAME_SELECT *)	未设定时将计算叠加框架 。	
$AC_MEAS_T_NUMBER *)	未设定时将采用当前生效的 T（T0）进行计算。	
$AC_MEAS_D_NUMBER *)	未设定时将采用当前生效的 D（D0）进行计算。	
$AC_MEAS_TYPE	= 9	测量类型 9：测量曲轴

*) 可选配

对于测量类型 9，将写入以下输出变量：

输出变量	含义
$AC_MEAS_FRAME	结果框架，包含平移
$AC_MEAS_DIAMETER	轴直径
$AC_MEAS_RESULTS[0]	计算出的中心的横坐标
$AC_MEAS_RESULTS[1]	计算出的中心的纵坐标
$AC_MEAS_RESULTS[2]	计算出的中心的垂直坐标
$AC_MEAS_RESULTS[3]	用于圆形调整的量化尺寸间隔平方数总和

10.5.2.5 测量类型 12：测量开槽

功能

槽通过逼近两个外边沿或内边沿测量。可将开槽中心设置到设定位置。通过移动方向分量确定隔断位置。

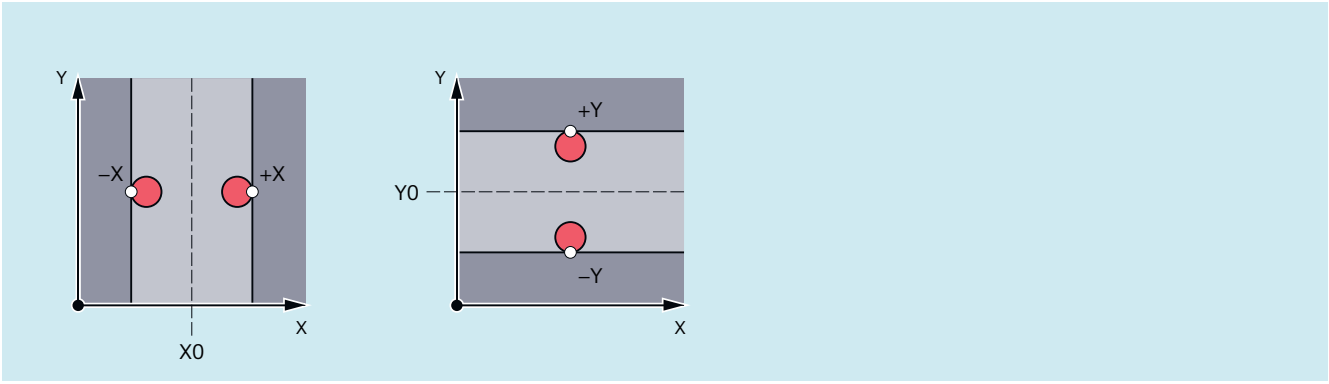


图 10-7 测量开槽

用于测量类型 12，将对以下系统变量数值进行评估：

输入变量	含义	
$AC_MEAS_VALID	输入变量的有效位	
$AA_MEAS_POINT1[<Axis>]	测量点 1	
$AA_MEAS_POINT2[<Axis>]	测量点 2	
$AA_MEAS_SETPOINT[<Axis>] *)	开槽中心设定位置	
$AC_MEAS_DIR_APPROACH	移动到工件上	
	= 0	+ X
	= 1	- X
	= 2	+ Y
	= 3	- Y
	= 4	+ Z
	= 5	- Z
$AC_MEAS_ACT_PLANE *)	未设定时将采用当前生效的平面。	
$AC_MEAS_FINE_TRANS *)	= 0	粗偏
	= 1	精偏
$AC_MEAS_FRAME_SELECT *)	未设定时将计算叠加框架。	

输入变量	含义	
$AC_MEAS_T_NUMBER *)	未设定时将采用当前生效的 T（T0）进行计算。	
$AC_MEAS_D_NUMBER *)	未设定时将采用当前生效的 D（D0）进行计算。	
$AC_MEAS_INPUT[0] *)	测量凹陷时第 2 测量点的逼近方向。必须具有与第 1 测量点的逼近方向相同的。坐标。	
$AC_MEAS_TYPE	= 12	测量类型 12：测量开槽

*) 可选配

对于测量类型 12，将写入以下输出变量：

输出变量	含义
$AC_MEAS_FRAME	结果框架，包含平移
$AC_MEAS_RESULTS[0]	计算出的开槽中心位置 (X0、Y0 或 Z0)
$AC_MEAS_RESULTS[1]	逼近方向上的槽宽

示例

测量槽，移动方向 x。

程序代码	注释
DEF INT RETVAL	
DEF FRAME TMP	
$TC_DP1[1,1]=120	; 类型
$TC_DP2[1,1]=20	; 0
$TC_DP3[1,1]= 10	; (Z) 长度补偿矢量
$TC_DP4[1,1]=0	; (Y)
$TC_DP5[1,1]= 0	; (X)
$TC_DP6[1,1]=2	; 半径
T1 D1	
G0 X0 Y0 Z0 F10000	
G54	
$P_CHBFRAME[0]= CROT(Z,45)	
$P_IFRAME[X,TR]= -sin(45)	
$P_IFRAME[Y,TR]= -sin(45)	
$P_PFRAME[Z,RT]= -45	
	; 测量槽。
$AC_MEAS_VALID=0	; 将所有输入值设置为无效。

程序代码	注释
G1 X-2	; 移动到第 1 测量点
$AA_MEAS_POINT1[X]=$AA_IW[X]	
$AA_MEAS_POINT1[Y]=$AA_IW[Y]	
$AA_MEAS_POINT1[Z]=$AA_IW[Z]	
G1 X4	; 第 2 测量点
$AA_MEAS_POINT2[X]=$AA_IW[X]	
$AA_MEAS_POINT2[Y]=$AA_IW[Y]	
$AA_MEAS_POINT2[Z]=$AA_IW[Z]	
$AA_MEAS_SETPOINT[X]=0	; 设置开槽中心的设定位置。
$AA_MEAS_SETPOINT[Y]=0	
$AA_MEAS_SETPOINT[Z]=0	
$AC_MEAS_DIR_APPROACH=0	; 设置移动方向 +X。
$AC_MEAS_ACT_PLANE=0	; 测量平面为 G17。
$AC_MEAS_FRAME_SELECT=0	; 选择框架 - SETFRAME
$AC_MEAS_T_NUMBER=1	; 选择刀具。
$AC_MEAS_D_NUMBER=1	
$AC_MEAS_TYPE=12	**; 将测量类型设置为开槽。**
RETVAL=MEASURE()	; 开始测量。
IF RETVAL <> 0 SETAL(61043, << RETVAL)	
ENDIF	
$P_SETFRAME=$AC_MEAS_FRAME	
$P_SETFR=$P_SETFRAME	; 将系统框架写入数据存储装置。
G1 X0 Y0	**; 移动靠近开槽中心。**
M30	

10.5.2.6 测量类型 13：隔片的测量

功能

隔断通过逼近两个外边沿或内边沿测量。可将隔片中心设置到设定位置。通过移动方向的分量确定隔断位置。

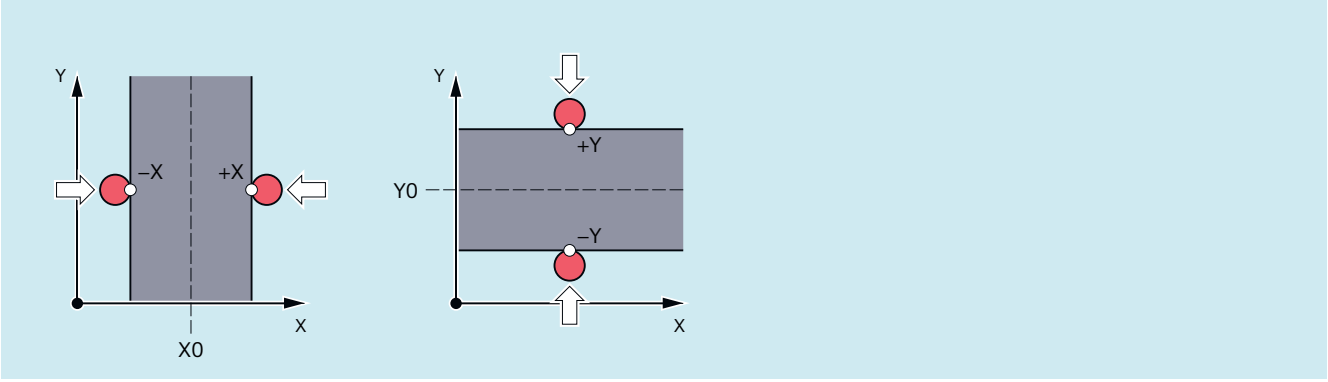


图 10-8 隔片的测量

用于测量类型 13，将对以下系统变量数值进行评估：

输入变量	含义	
$AC_MEAS_VALID	输入变量的有效位	
$AA_MEAS_POINT1[<Axis>]	测量点 1	
$AA_MEAS_POINT2[<Axis>]	测量点 2	
$AA_MEAS_SETPOINT[<Axis>] *)	隔片中心设定位置	
$AC_MEAS_DIR_APPROACH	移动到工件上	
	= 0	+ X
	= 1	- X
	= 2	+ Y
	= 3	- Y
	= 4	+ Z
	= 5	- Z
$AC_MEAS_ACT_PLANE *)	未设定时将采用当前生效的平面。	
$AC_MEAS_FINE_TRANS *)	= 0	粗偏
	= 1	精偏
$AC_MEAS_FRAME_SELECT *)	未设定时将计算叠加框架。	

输入变量	含义	
$AC_MEAS_T_NUMBER *)	未设定时将采用当前生效的 T（T0）进行计算。	
$AC_MEAS_D_NUMBER *)	未设定时将采用当前生效的 D（D0）进行计算。	
$AC_MEAS_INPUT[0] *)	测量凹陷时第 2 测量点的逼近方向。必须具有与第 1 测量点的逼近方向相同的。坐标。	
$AC_MEAS_TYPE	= 13	测量类型 13：隔片的测量

*) 可选配

对于测量类型 13，将写入以下输出变量：

输出变量	含义
$AC_MEAS_FRAME	结果框架，包含平移
$AC_MEAS_RESULTS[0]	计算出的隔断中心位置 (X0、Y0 或 Z0)
$AC_MEAS_RESULTS[1]	逼近方向上的隔断宽度

10.5.2.7 测量类型 14：用于几何轴和辅助轴的实际值设定

功能

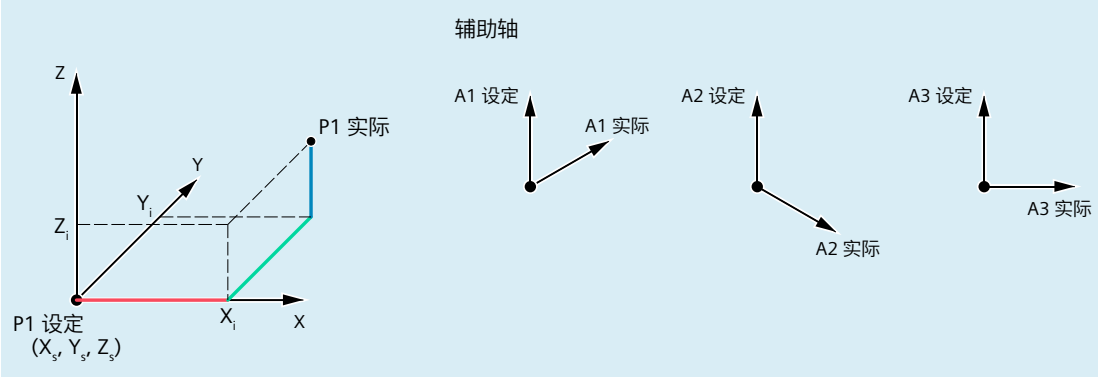


图 10-9 用于几何轴和辅助轴的实际值设定

用于测量类型 14，将对以下系统变量数值进行评估：

输入变量	含义
$AC_MEAS_VALID	输入变量的有效位
$AA_MEAS_POINT1[<Axis>]	轴的实际值
$AA_MEAS_SETPOINT[<Axis>] *)	单个轴的设定位置

输入变量	含义	
$AC_MEAS_ACT_PLANE *)	未设定时将采用当前生效的平面。	
$AC_MEAS_FINE_TRANS *)	= 0	粗偏
	= 1	精偏
$AC_MEAS_FRAME_SELECT *)	未设定时将计算叠加框架。	
$AC_MEAS_T_NUMBER *)	未设定时将采用当前生效的 T（T0）进行计算。	
$AC_MEAS_D_NUMBER *)	未设定时将采用当前生效的 D（D0）进行计算。	
$AC_MEAS_TYPE	= 14	测量类型 14：用于几何轴和辅助轴的实际值设定

*) 可选配

对于测量类型 14，将写入以下输出变量：

输出变量	含义
$AC_MEAS_FRAME	结果框架，包含平移

示例

相对坐标系中的参考点设置。

程序代码	注释
DEF INT RETVAL	
T1 D1	; 激活测头。
G54	; 激活所有框架和 G54。
TRANS X=10	; WCS 和 SZS 间的平移
G0 X0 F10000	; WCS(X)=0; SZS(X)=10
$AC_MEAS_VALID=0	; 将所有输入变量设置为无效。
$AC_MEAS_TYPE=14	**; 用于实际值设定的测量类型**
$AC_MEAS_ACT_PLANE=0	; 测量平面为 G17。
$AC_MEAS_P1_COORD=5	; ENS_REL 用于第 1 个测量点
$AC_MEAS_LATCH[0]=1	; 获取所有轴位置。
$AC_MEAS_SET_COORD=5	; 设定位置相对于 ENS。
$AA_MEAS_SETPOINT[X]=0	; 相对 ENS 坐标系统内的设定位置
$AC_MEAS_FRAME_SELECT=2505	; $P_RELFR

程序代码	注释
RETVAL=MEASURE()	; 计算 $P_RELFR; PI SETUDT(6)
IF RETVAL<>0 GOTOF ERROR	
ENDIF $P_RELFR=$AC_MEAS_FRAME	; 激活; PI SETUDT(7)

10.5.2.8 测量类型 15：只针对辅助轴的实际值设定

功能

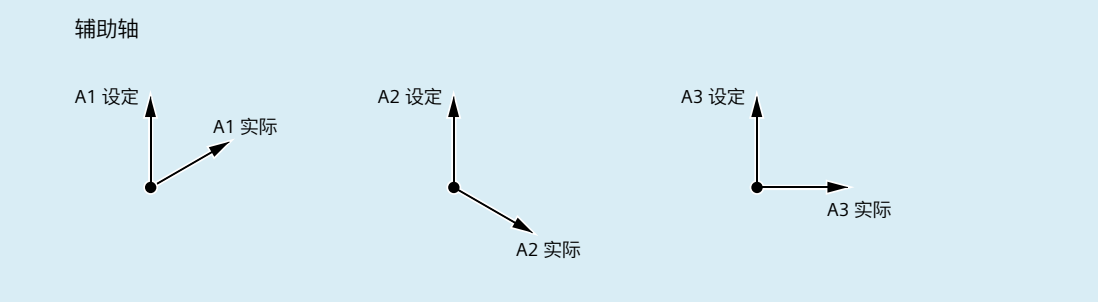


图 10-10 只针对辅助轴的实际值设定

用于测量类型 15，将对以下系统变量数值进行评估：

输入变量	含义	
$AC_MEAS_VALID	输入变量的有效位	
$AA_MEAS_POINT1[<Axis>]	轴的实际值	
$AA_MEAS_SETPOINT[<Axis>] *)	单个轴的设定位置	
$AC_MEAS_FINE_TRANS *)	= 0	粗偏
	= 1	精偏
$AC_MEAS_FRAME_SELECT *)	未设定时将计算叠加框架。	
$AC_MEAS_TYPE	= 15	测量类型 14：只针对辅助轴的实际值设定

*) 可选配

对于测量类型 15，将写入以下输出变量：

输出变量	含义
$AC_MEAS_FRAME	结果框架，包含平移

10.5.2.9 测量类型 16：测量斜边

功能

通过此测量可确定工件位置角，并将其输入框架。可输入一个范围为 ± 90 度的设定角度，其可作为激活相对生效 WCS 的结果框架后产生的旋转。

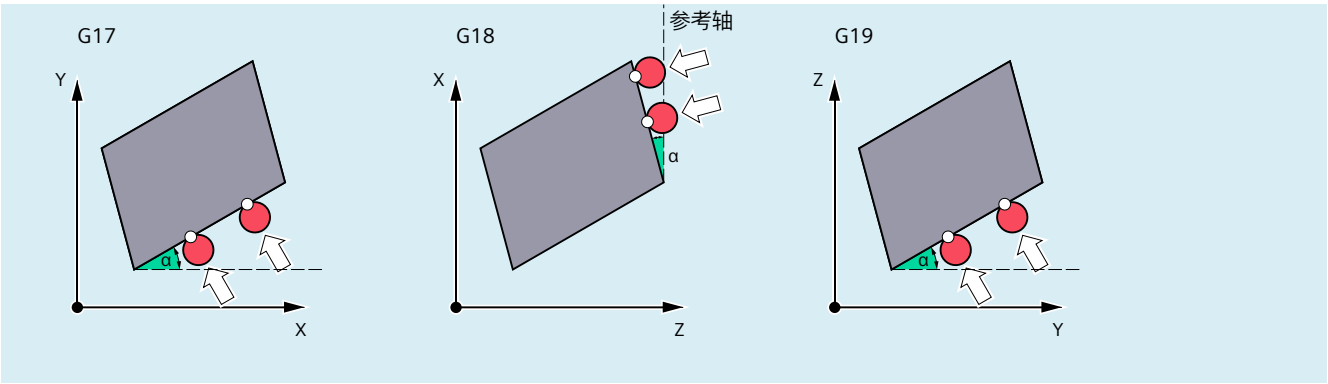


图 10-11　测量 G17/G18/ G19 平面中的斜边

用于测量类型 16，将对以下系统变量数值进行评估：

输入变量	含义	
$AC_MEAS_VALID	输入变量的有效位	
$AA_MEAS_POINT1[<Axis>]	测量点 1	
$AA_MEAS_POINT2[<Axis>]	测量点 2	
$AA_MEAS_SETANGLE *)	设定角度	
$AC_MEAS_ACT_PLANE *)	未设定时将采用当前生效的平面。	
$AC_MEAS_FINE_TRANS *)	= 0	粗偏
	= 1	精偏
$AC_MEAS_FRAME_SELECT *)	未设定时将计算叠加框架。	
$AC_MEAS_T_NUMBER *)	未设定时将采用当前生效的 T（T0）进行计算。	
$AC_MEAS_D_NUMBER *)	未设定时将采用当前生效的 D（D0）进行计算。	
$AC_MEAS_INPUT[0] *)	若未设定，则总是将工件应对准的参考坐标作为所选平面的横坐标。	
	= 0	参考坐标为横坐标
	= 1	参考坐标为纵坐标

输入变量	含义	
$AC_MEAS_INPUT[1] *)	若未设定，则将工件位置角作为旋转输入框架。否则可为回转轴指定一个通道轴索引，将其平移与计算出的旋转一起设置到当前的回转轴位置中。这样工件便在回转轴位置 = 0 时对准。当前回转轴值必须位于 $AA_MEAS_POINT[<Axis>] 。	
$AC_MEAS_TYPE	= 16	测量类型 16：测量斜边

* 可选

对于测量类型 16，将写入以下输出变量：

输出变量	含义
$AC_MEAS_FRAME	结果框架，包含旋转
$AC_MEAS_WP_ANGLE	计算出的工件位置角

10.5.2.10 测量类型 17：测量倾斜平面中的角度

功能

倾斜平面通过 P1、P2 和 P3 三个测量点确定。

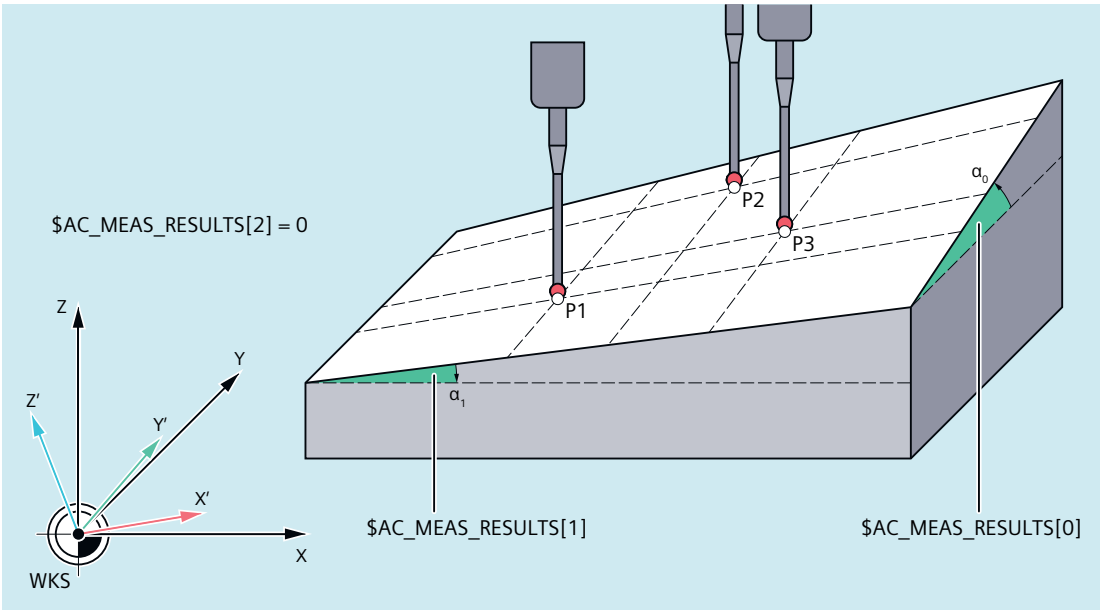


图 10-12 测量 G17 中倾斜平面中的角度

通过测量类型 17 确定两个用于平面倾斜位置的角度 α_0 和 α_1，并输入 $AC_MEAS_RESULTS[0 ...1]：

- $AC_MEAS_RESULTS[0] → 绕横坐标旋转
- $AC_MEAS_RESULTS[1] → 绕纵坐标旋转

借助三个测量点 P1、P2 和 P3 计算这些角度。

采用此测量类型 17 时，垂直坐标的角度（$AC_MEAS_RESULTS[2]）总是预设为 0。

可为横坐标和/或纵坐标设定一个设定旋转，此旋转输入至结果框架中。若只通过设定值设定了一个角度，计算第二个角度时会假定三个测量点在倾斜平面上构成设定角度。结果框架中只输入旋转，WCS 参考点保留。WCS 会被旋转，从而使 z' 垂直于倾斜平面。

对于测量类型 17，将确定以下平面设置：

轴名称	**G17**	**G18**	**G19**
横坐标	X 轴	Z 轴	Y 轴
纵坐标	Y 轴	X 轴	Z 轴
垂直坐标（进给轴）	Z 轴	Y 轴	X 轴

用于测量类型 17，将对以下系统变量数值进行评估：

输入变量	**含义**	
$AC_MEAS_VALID	输入变量的有效位	
$AA_MEAS_POINT1[<Axis>]	测量点 1	
$AA_MEAS_POINT2[<Axis>]	测量点 2	
$AA_MEAS_POINT3[<Axis>]	测量点 3	
$AA_MEAS_SETANGLE[<Axis>] *)	围绕横坐标和纵坐标的设定旋转。	
$AC_MEAS_ACT_PLANE *)	未设定时将采用当前生效的平面。	
$AC_MEAS_FRAME_SELECT *)	未设定时将计算叠加框架 。	
$AC_MEAS_T_NUMBER *)	未设定时将采用当前生效的 T（T0）进行计算。	
$AC_MEAS_D_NUMBER *)	未设定时将采用当前生效的 D（D0）进行计算。	
$AC_MEAS_INPUT[0] *)	如无规定要求，则将位置点置于平面内。	
	= 0	位置点将不置于平面上。
	= 1	位置点将置于活动平面或选择的平面内。
$AC_MEAS_TYPE	= 17	

*) 可选配

对于测量类型 17，将写入以下输出变量：

输出变量	含义
$AC_MEAS_FRAME	结果框架
$AC_MEAS_RESULTS[0]	通过三个测量点计算出的、围绕横坐标的角度
$AC_MEAS_RESULTS[1]	通过三个测量点计算出的、围绕纵坐标的角度
$AC_MEAS_RESULTS[2]	通过三个测量点计算出的、围绕垂直坐标的角度
$AC_MEAS_RESULTS[3]	结果框架中输入的、围绕横坐标的角度
$AC_MEAS_RESULTS[4]	结果框架中输入的、围绕纵坐标的角度
$AC_MEAS_RESULTS[5]	结果框架中输入的、围绕垂直坐标的角度

示例

```
DEF INT RETVAL
DEF AXIS _XX, _YY, _ZZ

T1 D1                                   ; 激活测头。
G54                                     ; 激活所有框架和 G54。

$AC_MEAS_VALID=0                        ; 将所有输入值设置为无效。

$AC_MEAS_TYPE=17                        ; 将测量类型设置为倾斜平面。
$AC_MEAS_ACT_PLANE=0                    ; 测量平面为 G17。

_XX=$P_AXN1                             ; 根据平面定义轴。
_YY=$P_AXN2
_ZZ=$P_AXN3

G17 G1 _XX=10 _YY=10 F1000              ; 移动靠近第 1 测量点。
MEAS=1 _ZZ=...
$AA_MEAS_POINT1[_XX]=$AA_MW[_XX]        ; 将测量值指定给横坐标。
$AA_MEAS_POINT1[_YY]=$AA_MW[_YY]        ; 将测量值指定给纵坐标。
$AA_MEAS_POINT1[_ZZ]=$AA_MW[_ZZ]        ; 将测量值指定给垂直坐标。

G1 _XX=20 _YY=10 F1000                  ; 移动靠近第 2 测量点。
MEAS=1 _ZZ=...
$AA_MEAS_POINT2[_XX]=$AA_MW[_XX]        ; 将测量值指定给横坐标。
$AA_MEAS_POINT2[_YY]=$AA_MW[_YY]        ; 将测量值指定给纵坐标。
$AA_MEAS_POINT2[_ZZ]=$AA_MW[_ZZ]        ; 将测量值指定给垂直坐标。
```

程序代码	注释
G1 _XX=20 _YY=20 F1000	; 移动靠近第 3 测量点。
MEAS=1 _ZZ=...	
$AA_MEAS_POINT3[_XX]=$AA_MW[_XX]	; 将测量值指定给横坐标。
$AA_MEAS_POINT3[_YY]=$AA_MW[_YY]	; 将测量值指定给纵坐标。
$AA_MEAS_POINT3[_ZZ]=$AA_MW[_ZZ]	; 将测量值指定给垂直坐标。
	; 设定角度设定值。
$AA_MEAS_SETANGLE[_XX]=12	; 绕横坐标旋转
$AA_MEAS_SETANGLE[_YY]=4	; 绕纵坐标旋转
$AC_MEAS_FRAME_SELECT=102	; 选择目标框架 - G55
$AC_MEAS_T_NUMBER=1	; 选择刀具。
$AC_MEAS_D_NUMBER=1	
RETVAL=MEASURE()	; 开始测量。
IF RETVAL <> 0	
SETAL(61043, << RETVAL)	
ENDIF	
IF $AC_MEAS_RESULTS[0] <> 12	
SETAL(61043, << $AC_MEAS_RESULTS[0])	
ENDIF	
IF $AC_MEAS_RESULTS[1] <> 4	
SETAL(61043, << $AC_MEAS_RESULTS[1])	
ENDIF	
$P_UIFR[2]=$AC_MEAS_FRAME	; 将测量框架写入数据管理（G55）。
G55 G0 AX[_XX]=10 AX[_YY]=10	; 激活框架并运行。
M30	

10.5.2.11 测量类型 18：在倾斜平面上重新定义 WCS 坐标系

功能

新 WCS' 坐标系的零点通过倾斜平面法线上的测量点 P1 确定。

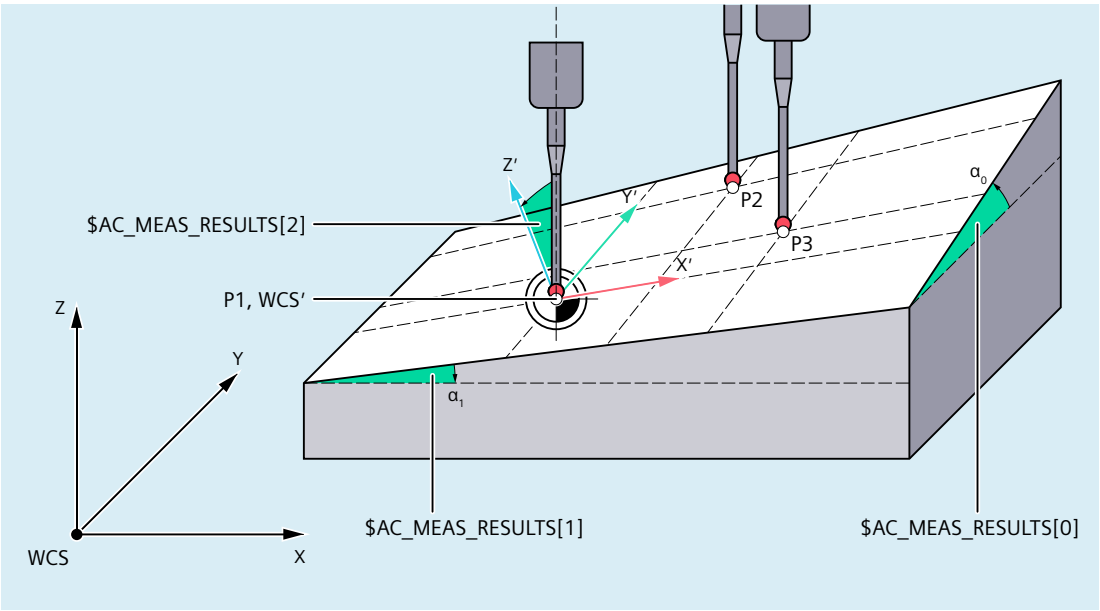


图 10-13　在 G17 中重新定义倾斜平面上的 WCS

平面测量

平面测量在测量循环中执行。此循环记录三个测量点，并向测量接口提供必要的数值。

借助 MEASURE() 功能可依据输入值计算出空间角以及新 WCS' 的平移。

测量框架转换

计算的结果（即空间角和平移）会被输入至测量框架 $AC_MEAS_FRAME 中。根据框架链中选择的框架，系统会对测量框架进行转换。此外空间角还会另外保存至输出值 $AC_MEAS_RESULTS[0 ...2] 中。

- $AC_MEAS_RESULTS[0]:围绕旧 WCS 横坐标的角度
- $AC_MEAS_RESULTS[1]:围绕纵坐标的角度
- $AC_MEAS_RESULTS[2]:围绕垂直坐标的角度

确定新 WCS' 的零点

执行完测量后，测量循环可通过测量框架描述和激活框架链中选择的框架。激活后，新的 WCS' 坐标系位于倾斜平面的法线上，测量点 P1 作为新 WCS' 的零点。

之后编写的位置将以倾斜平面为基准。

应用

CAD 系统常通过设定一个平面上的 P1、P2 和 P3 三个点来定义空间中的斜置平面。其中将使用第 1 个测量点 P1 作为新的 WCS'参照点，第 2 个测量点 P2 给出了新的旋转 WCS'坐标轴的横坐标方向，第 3 个测量定 P3 需用于测定空间角度。

输入/输出变量

用于测量类型 18，将对以下系统变量数值进行评估：

输入变量	含义	
$AC_MEAS_VALID	输入变量的有效位	
$AA_MEAS_POINT1[<Axis>]	测量点 1	
$AA_MEAS_POINT2[<Axis>]	测量点 2	
$AA_MEAS_POINT3[<Axis>]	测量点 3	
$AA_MEAS_SETPOINT[<Axis>] *)	P1 设定位置	
$AC_MEAS_ACT_PLANE *)	未设定时将采用当前生效的平面。	
$AC_MEAS_FRAME_SELECT *)	未设定时将计算叠加框架。	
$AC_MEAS_T_NUMBER *)	未设定时将采用当前生效的 T（T0）进行计算。	
$AC_MEAS_D_NUMBER *)	未设定时将采用当前生效的 D（D0）进行计算。	
$AC_MEAS_INPUT[0] *)	如无规定要求，则将位置点置于平面内。	
	= 0	位置点将不置于平面上。
	= 1	位置点将置于活动平面或选择的平面内。
$AC_MEAS_TYPE	= 18	测量类型 18：在倾斜平面上重新定义 WCS 坐标系

* 可选

对于测量类型 18，将写入以下输出变量：

输出变量	含义
$AC_MEAS_FRAME	结果框架，包含旋转和转换
$AC_MEAS_RESULTS[0]	计算出的围绕横坐标的角度
$AC_MEAS_RESULTS[1]	计算出的围绕纵坐标的角度
$AC_MEAS_RESULTS[2]	计算出的围绕垂直坐标的角度

示例

程序代码	注释
DEF INT RETVAL	
DEF AXIS _XX, _YY, _ZZ	
T1 D1	; 激活测头。
G54	; 激活所有框架和 G54。
$AC_MEAS_VALID=0	; 将所有输入值设置为无效。
$AC_MEAS_TYPE=18	**; 将测量类型设置为倾斜平面。**
$AC_MEAS_ACT_PLANE=0	; 测量平面为 G17。
_XX=$P_AXN1	; 根据平面定义轴。
_YY=$P_AXN2	
_ZZ=$P_AXN3	
G17 G1 _XX=10 _YY=10 F1000	; 移动靠近第 1 测量点。
MEAS=1 _ZZ=...	
$AA_MEAS_POINT1[_XX]=$AA_MW[_XX]	; 将测量值指定给横坐标。
$AA_MEAS_POINT1[_YY]=$AA_MW[_YY]	; 将测量值指定给纵坐标。
$AA_MEAS_POINT1[_ZZ]=$AA_MW[_ZZ]	; 将测量值指定给垂直坐标。
G1 _XX=20 _YY=10 F1000	; 移动靠近第 2 测量点。
MEAS=1 _ZZ=...	
$AA_MEAS_POINT2[_XX]=$AA_MW[_XX]	; 将测量值指定给横坐标。
$AA_MEAS_POINT2[_YY]=$AA_MW[_YY]	; 将测量值指定给纵坐标。
$AA_MEAS_POINT2[_ZZ]=$AA_MW[_ZZ]	; 将测量值指定给垂直坐标。
G1 _XX=20 _YY=20 F1000	; 移动靠近第 3 测量点。
MEAS=1 _ZZ=...	
$AA_MEAS_POINT3[_XX]=$AA_MW[_XX]	; 将测量值指定给横坐标。
$AA_MEAS_POINT3[_YY]=$AA_MW[_YY]	; 将测量值指定给纵坐标。
$AA_MEAS_POINT3[_ZZ]=$AA_MW[_ZZ]	; 将测量值指定给垂直坐标。
$AA_MEAS_SETPOINT[_XX]=10	; 设定 P1 的设定值。
$AA_MEAS_SETPOINT[_YY]=10	
$AA_MEAS_SETPOINT[_ZZ]=10	
$AC_MEAS_FRAME_SELECT=102	; 选择目标框架 - G55
$AC_MEAS_T_NUMBER=1	; 选择刀具。
$AC_MEAS_D_NUMBER=1	

程序代码	注释
RETVAL=MEASURE()	；开始测量。
IF RETVAL <> 0	
SETAL(61043, << RETVAL)	
ENDIF	
	；空间角的计算结果。
R0=$AC_MEAS_RESULTS[0]	；围绕旧 WCS 横坐标的角度
R1=$AC_MEAS_RESULTS[1]	；围绕纵坐标的角度
R2=$AC_MEAS_RESULTS[2]	；围绕垂直坐标的角度
$P_UIFR[2]=$AC_MEAS_FRAME	；将测量框架写入数据管理（G55）。
G55 G0 AX[_XX]=10 AX[_YY]=10	；激活框架并运行。
M30	

10.5.2.12 测量类型 19：1 维设定值给定

功能

使用此测量方法可为横坐标、纵坐标或垂直坐标给定一个设定值。若给定了两个或三个设定值，则只会按照“横坐标 - 纵坐标 - 垂直坐标”的顺序接收第一个值。此时刀具不被考虑在内。

这是一个完全意义上的用于横坐标、纵坐标或垂直坐标的实际值设定功能。

用于测量类型 19，将对以下系统变量数值进行评估：

输入变量	含义
$AC_MEAS_VALID	输入变量的有效位
$AA_MEAS_POINT1[<Axis>]	横坐标的测量点 1
$AA_MEAS_POINT1[<Axis>]	纵坐标的测量点 1
$AA_MEAS_POINT1[<Axis>]	垂直坐标的测量点 1
$AA_MEAS_SETPOINT[<Axis>]	横坐标、纵坐标或垂直坐标的设定位置
$AC_MEAS_ACT_PLANE *)	未设定时将采用当前生效的平面。
$AC_MEAS_FRAME_SELECT *)	未设定时将计算叠加框架。

输入变量	含义	
$AC_MEAS_FINE_TRANS *)	未设定时将写入粗偏移 。	
	= 0	粗偏
	= 1	精偏
$AC_MEAS_TYPE	= 19	测量类型 19：1 维设定值给定

*) 可选配

对于测量类型 19，将写入以下输出变量：

输出变量	含义
$AC_MEAS_FRAME	结果框架，包含旋转和平移

示例

```
DEF INT RETVAL
DEF REAL _CORMW_XX, CORMW_YY, _CORMW_ZZ
DEF AXIS _XX, _YY, _ZZ
T1 D1                                   ; 激活测头。
G54                                     ; 激活所有框架和 G54。
$AC_MEAS_VALID=0                        ; 将所有输入值设置为无效。
$AC_MEAS_TYPE=19                        ; 将测量类型设为"1 维设定值给定”。
$AC_MEAS_ACT_PLANE=0                    ; 测量平面为 G17。
_XX=$P_AXN1                             ; 根据平面定义轴。
_YY=$P_AXN2
_ZZ=$P_AXN3
                                        ; 指定测量值。
$AA_MEAS_POINT1[_XX]=$AA_MW[_XX]        ; 将测量值指定给横坐标。
$AA_MEAS_POINT1[_YY]=$AA_MW[_YY]        ; 将测量值指定给纵坐标。
$AA_MEAS_POINT1[_ZZ]=$AA_MW[_ZZ]        ; 将测量值指定给垂直坐标。
$AA_MEAS_SETPOINT[_XX]=10               ; 设定横坐标的设定值。
$AC_MEAS_FRAME_SELECT=102               ; 选择目标框架 - G55
RETVAL=MEASURE()                        ; 开始测量。
IF RETVAL <> 0
SETAL(61043, << RETVAL)
ENDIF
$P_UIFR[2]=$AC_MEAS_FRAME               ; 将测量框架写入数据管理（G55）。
G55 G0 AX[_XX]=10 AX[_YY]=10            ; 激活框架并运行。
M30
```

10.5.2.13 测量类型 20：2 维设定值给定

功能

采用此测量方法时可设定两个维度的设定值。此时可从 3 个维度中任意选择 2 个。若给定了三个设定值，则只会接收横坐标和纵坐标的值。此时刀具不被考虑在内。

这是一个完全意义上的实际值设定功能。

用于测量类型 20，将对以下系统变量数值进行评估：

输入变量	含义	
$AC_MEAS_VALID	输入变量的有效位	
$AA_MEAS_POINT1[<Axis>]	横坐标的测量点 1	
$AA_MEAS_POINT1[<Axis>]	纵坐标的测量点 1	
$AA_MEAS_POINT1[<Axis>]	垂直坐标的测量点 1	
$AA_MEAS_SETPOINT[<Axis>]	第 1 维的设定位置	
$AA_MEAS_SETPOINT[<Axis>]	第 2 维的设定位置	
$AC_MEAS_ACT_PLANE *)	未设定时将采用当前生效的平面。	
$AC_MEAS_FRAME_SELECT *)	未设定时将计算叠加框架。	
$AC_MEAS_FINE_TRANS *)	未设定时将写入粗偏移 。	
	= 0	粗偏
	= 1	精偏
$AC_MEAS_TYPE	= 20	测量类型 20：2 维设定值给定

*) 可选配

对于测量类型 20，将写入以下输出变量：

输出变量	含义
$AC_MEAS_FRAME	结果框架，包含旋转和平移

示例

程序代码	注释
DEF INT RETVAL	
DEF REAL _CORMW_XX, CORMW_YY, _CORMW_ZZ	
DEF AXIS _XX, _YY, _ZZ	
T1 D1	；激活测头。

程序代码	注释
G54	；激活所有框架和 G54。
$AC_MEAS_VALID=0	；将所有输入值设置为无效。
$AC_MEAS_TYPE=20	**；将测量类型设为"2 维设定值给定"。**
$AC_MEAS_ACT_PLANE=0	；测量平面为 G17。
_XX=$P_AXN1	；根据平面定义轴。
_YY=$P_AXN2	
_ZZ=$P_AXN3	
	；指定测量值。
$AA_MEAS_POINT1[_XX]=$AA_MW[_XX]	；将测量值指定给横坐标。
$AA_MEAS_POINT1[_YY]=$AA_MW[_YY]	；将测量值指定给纵坐标。
$AA_MEAS_POINT1[_ZZ]=$AA_MW[_ZZ]	；将测量值指定给垂直坐标。
$AA_MEAS_SETPOINT[_XX]=10	；设定横坐标和纵坐标的设定值。
$AA_MEAS_SETPOINT[_YY]=10	
$AC_MEAS_FRAME_SELECT=102	；选择目标框架 - G55
RETVAL=MEASURE()	；开始测量。
IF RETVAL <> 0	
SETAL(61043, << RETVAL)	
ENDIF	
$P_UIFR[2]=$AC_MEAS_FRAME	；将测量框架写入数据管理（G55）。
G55 G0 AX[_XX]=10 AX[_YY]=10	；激活框架并运行。
M30	

10.5.2.14 测量类型 21：3 维设定值给定

功能

使用此测量方法可为横坐标、纵坐标和垂直坐标分别给定一个设定值。此时刀具不被考虑在内。

这是一个完全意义上的用于横坐标、纵坐标和垂直坐标的实际值设定功能。

用于测量类型 21，将对以下系统变量数值进行评估：

输入变量	含义
$AC_MEAS_VALID	输入变量的有效位
$AA_MEAS_POINT1[<Axis>]	横坐标的测量点 1
$AA_MEAS_POINT1[<Axis>]	纵坐标的测量点 1
$AA_MEAS_POINT1[<Axis>]	垂直坐标的测量点 1
$AA_MEAS_SETPOINT[<Axis>]	横坐标的设定位置
$AA_MEAS_SETPOINT[<Axis>]	纵坐标的设定位置

输入变量	含义	
$AA_MEAS_SETPOINT[<Axis>]	垂直坐标的设定位置	
$AC_MEAS_ACT_PLANE *)	未设定时将采用当前生效的平面。	
$AC_MEAS_FRAME_SELECT *)	未设定时将计算叠加框架。	
$AC_MEAS_FINE_TRANS *)	未设定时将写入粗偏移 。	
	= 0	粗偏
	= 1	精偏
$AC_MEAS_TYPE	= 21	测量类型 21：3 维设定值给定

*) 可选配

对于测量类型 21，将写入以下输出变量：

输出变量	含义
$AC_MEAS_FRAME	结果框架，包含旋转和平移

示例

程序代码	注释
DEF INT RETVAL	
DEF REAL _CORMW_XX, CORMW_YY, _CORMW_ZZ	
DEF AXIS _XX, _YY, _ZZ	
T1 D1	; 激活测头。
G54	; 激活所有框架和 G54。
$AC_MEAS_VALID=0	; 将所有输入值设置为无效。
$AC_MEAS_TYPE=21	**; 将测量类型设为"3 维设定值给定"。**
$AC_MEAS_ACT_PLANE=0	; 测量平面为 G17。
_XX=$P_AXN1	; 根据平面定义轴。
_YY=$P_AXN2	
_ZZ=$P_AXN3	
	; 指定测量值。
$AA_MEAS_POINT1[_XX]=$AA_MW[_XX]	; 将测量值指定给横坐标。
$AA_MEAS_POINT1[_YY]=$AA_MW[_YY]	; 将测量值指定给纵坐标。
$AA_MEAS_POINT1[_ZZ]=$AA_MW[_ZZ]	; 将测量值指定给垂直坐标。
$AA_MEAS_SETPOINT[_XX]=10	; 给定横坐标、纵坐标和垂直坐标的设定值。
$AA_MEAS_SETPOINT[_YY]=10	
$AA_MEAS_SETPOINT[_ZZ]=10	
$AC_MEAS_FRAME_SELECT=102	; 选择目标框架 - G55
RETVAL=MEASURE()	; 开始测量。
IF RETVAL <> 0	
SETAL(61043, << RETVAL)	

程序代码	注释
ENDIF	
$P_UIFR[2]=$AC_MEAS_FRAME	；将测量框架写入数据管理（G55）。
G55 G0 AX[_XX]=10 AX[_YY]=10	；激活框架并运行。
M30	

10.5.2.15 测量类型 24：测量点坐标轴转换

功能

使用此测量方法可将任意坐标系（WCS、BCS、MCS）中的测量点通过坐标转换的方式换算为新的坐标系。

新坐标系通过指定所需的框架链生成。

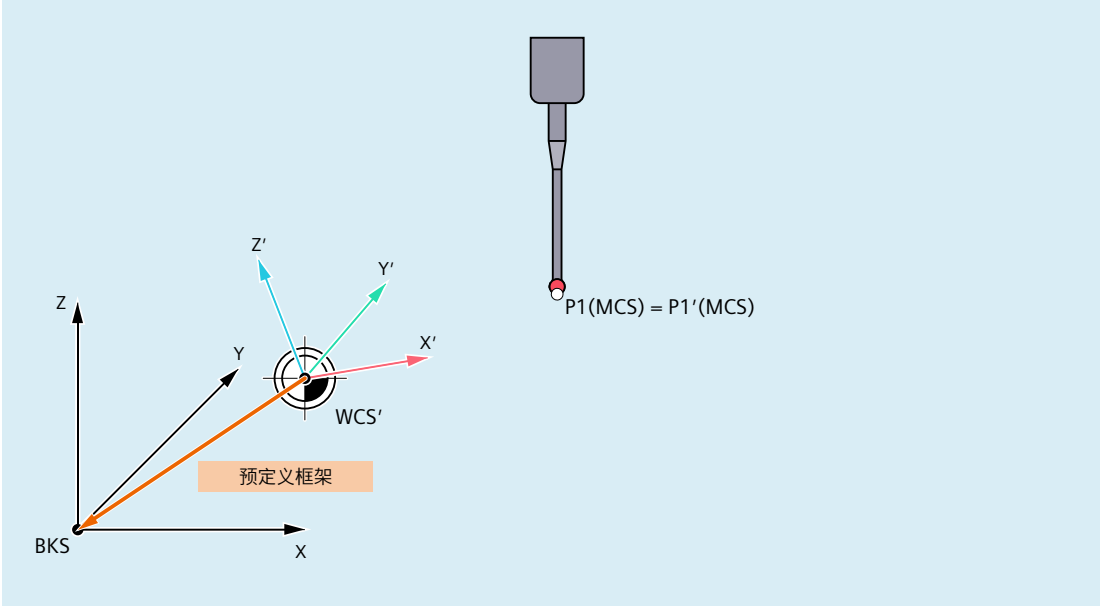


图 10-14 测量点坐标轴转换

用于测量类型 24，将对以下系统变量数值进行评估：

输入变量	含义	
$AC_MEAS_VALID	输入变量的有效位	
$AA_MEAS_POINT1[<Axis>]	需要转换的位置	
$AC_MEAS_P1:COORD *)	= 0	WCS (默认)
	= 1	BCS
	= 2	MCS

输入变量	含义	
$AC_MEAS_P2_COORD *)	目标坐标系	
$AC_MEAS_TOOL_MASK *)	0x20	生效刀具的长度计入位置的坐标系转换
$AC_MEAS_CHSFR *)	数据管理中的系统框架	
$AC_MEAS_NCBFR *)	数据管理中的全局基本框架	
$AC_MEAS_CHBFR *)	数据管理中的通道基本框架	
$AC_MEAS_UIFR *)	数据管理中的可设定框架	
$AC_MEAS_PFRAME *)	= 1	不结算可编程框架。
$AC_MEAS_TYPE	= 24	测量类型 24：

*) 可选配

对于测量类型 24，将写入以下输出变量：

输出变量	含义
$AC_MEAS_POINT2[轴]	经过换算的轴位置

示例

被测位置的 WCS 坐标转换。

程序代码	注释
DEF INT RETVAL	
DEF INT LAUF	
DEF REAL _CORMW_XX, _CORMW_YY, _CORMW_ZZ	
DEF AXIS _XX, _YY, _ZZ	
$TC_DP1[1,1]=120	; 刀具类型为立铣刀
$TC_DP2[1,1]=20	
$TC_DP3[1,1]=0	(Z) 长度补偿矢量
$TC_DP4[1,1]=0	(Y) 长度补偿矢量
$TC_DP5[1,1]= 0	(X) 长度补偿矢量
$TC_DP6[1,1]=2	; 半径
T1 D1	; 激活测头。
G17	; 倾斜平面 G17
_XX=$P_AXN1 _YY=$P_AXN2 _ZZ=$P_AXN3	**; 根据平面定义轴。**
	; 整个框架根据以下条件得出
	; CTRANS(_XX,10,_YY,-1,_ZZ,5,A,6,B,7)
$P_CHBFR[0]=CTRANS(_ZZ,5,A,6):CROT(_ZZ,45)	

程序代码	注释
$P_UIFR[1]=CTRANS( )	
$P_UIFR[1,_XX,TR]= -SIN(45)	
$P_UIFR[1,_YY,TR]= -SIN(45)	
$P_UIFR[2]=CTRANS( )	
$P_PFRAME=CROT(_ZZ,-45)	
$P_CYCFR=CTRANS(_XX,10,B,7)	
G54	; 激活所有框架和 G54。
G0 X0 Y0 Z0 A0 B0 F1000	
$AC_MEAS_VALID=0	; 将所有输入值设置为无效。
$AC_MEAS_TYPE=24	**; 设置测量类型设置用于坐标转换。**
$AC_MEAS_ACT_PLANE=0	; 测量平面为 G17。
	; 指定测量值。
$AA_MEAS_POINT1[_XX]=$AA_IW[_XX]	; 将测量值指定给横坐标。
$AA_MEAS_POINT1[_YY]=$AA_IW[_YY]	; 将测量值指定给纵坐标。
$AA_MEAS_POINT1[_ZZ]=$AA_IW[_ZZ]	; 将测量值指定给垂直坐标。
$AA_MEAS_POINT1[A]=$AA_IW[A]	
$AA_MEAS_POINT1[B]=$AA_IW[B]	
$AC_MEAS_P1_COORD=0	; WCS 位置换算至 WCS'
$AC_MEAS_P2_COORD=0	
	; 设置 WCS。
	; 总框架由以下构成：
	; CTRANS(_XX,0,_YY,0,_ZZ,5,A,6,B,0)
	; 取消循环框架
$AC_MEAS_CHSER=$MC_MM_SYSTEM_FRAME_MASK B_AND 'B1011111'	
$AC_MEAS_NCBFR='B0'	; 取消全局基本框架。
$AC_MEAS_CHBFR='B1'	; 数据管理中的通道基本框架 1
$AC_MEAS_UIFR=2	; 数据管理中的可设定框架 G55
$AA_MEAS_PFRAME=1	; 可设定框架不计入。
RETVAL=MEASURE()	; 开始测量。
IF RETVAL <> 0	
SETAL(61043, << RETVAL)	
ENDIF	
IF $AA_MEAS_PIONT2[_XX] <> 10	
SETAL(61043)	
M0	
STOPRE	
ENDIF	
IF $AA_MEAS_PIONT2[_YY] <> -1	
SETAL(61043)	

程序代码	注释
M0	
STOPRE	
IF $AA_MEAS_PIONT2[_ZZ] <> 0	
SETAL(61043)	
M0	
STOPRE	
IF $AA_MEAS_PIONT2[A] <> 0	
SETAL(61043)	
M0	
STOPRE	
IF $AA_MEAS_PIONT2[B] <> 7	
SETAL(61043)	
M0	
STOPRE	
M30	

10.5.2.16 测量类型 25：测量矩形

功能

在工作平面 G17（工作平面 X/Y、进给方向 Z）、G18（工作平面 Z/X、进给方向 Y）或 G19（工作平面 Y/Z、进给方向 X）内测定矩形时（刀具尺寸），每个矩形需要 4 个测量点。

这些测量点可按任意顺序设定。纵坐标间距最大的测量点为 P3 和 P4。

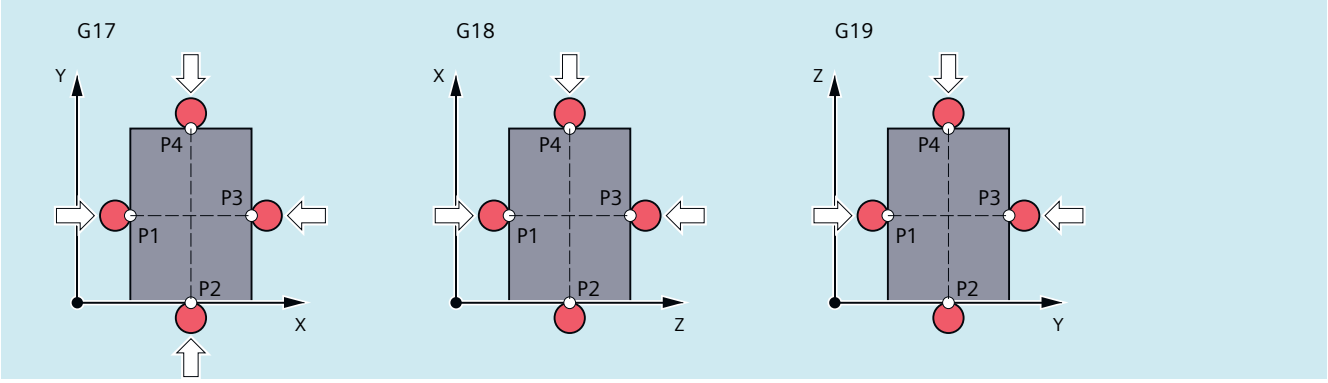


图 10-15 测量矩形

用于测量类型 25，将对以下系统变量数值进行评估：

输入变量	含义	
$AC_MEAS_VALID	输入变量的有效位	
$AA_MEAS_POINT1[<Axis>]	测量点 1	
$AA_MEAS_POINT2[<Axis>]	测量点 2	
$AA_MEAS_POINT3[<Axis>]	测量点 3	
$AA_MEAS_POINT4[<Axis>]	测量点 4	
$AC_MEAS_ACT_PLANE *)	未设定时将采用当前生效的平面。	
$AC_MEAS_FINE_TRANS *)	= 0	粗偏
	= 1	精偏
$AC_MEAS_FRAME_SELECT *)	未设定时将计算叠加框架。	
$AC_MEAS_T_NUMBER *)	未设定时将采用当前生效的 T（T0）进行计算。	
$AC_MEAS_D_NUMBER *)	未设定时将采用当前生效的 D（D0）进行计算。	
$AC_MEAS_INPUT[0] *)	未设定时测量外角。	
	= 0	外角测量
	= 1	内角测量
$AC_MEAS_TYPE	= 25	测量类型 25：测量矩形

*) 可选配

对于测量类型 25，将写入以下输出变量：

输出变量	含义
$AC_MEAS_FRAME	结果框架，包含平移
$AC_MEAS_RESULTS[0]	计算出的中心的横坐标
$AC_MEAS_RESULTS[1]	计算出的中心的纵坐标
$AC_MEAS_RESULTS[2]	计算出的中心的垂直坐标
$AC_MEAS_RESULTS[3]	矩形宽度 P1/P2
$AC_MEAS_RESULTS[4]	矩形长度 P3/P4

10.5.2.17 测量类型 26：备份数据管理框架

功能

使用此测量类型时，可将全部或选定的部分数据管理框架以当前的赋值备份至一个文件。这可通过执行命令或通过零件程序进行。此功能也可从不同的通道激活。文件将创建在目录"_N_SYF_DIR" 内。

恢复操作会删除数据备份，重新备份会覆盖此前的备份。

最后备份的数据可通过使用以下设置进行的重新备份，从数据管理系统中删除：

- $AC_MEAS_CHSFR = 0 ; 系统框架
- $AC_MEAS_NCBFR = 0 ; 全局框架
- $AC_MEAS_CHBFR = 0； 通道基本框架
- $AC_MEAS_UIFR = 0 ; 可设定框架数量

说明

备份所有数据管理框架时须注意，每个框架要占用约 1 kB 的存储空间。如存储空间不足，则故障信息过程将中断。

解决方法：调整静态用户存储器的配置。

更多信息：功能手册之基本功能，存储器配置

用于测量类型 26，将对以下系统变量数值进行评估：

输入变量	含义	
$AC_MEAS_VALID	输入变量的有效位	
$AC_MEAS_CHSFR *)	数据管理中的系统框架的位标记 若未写入此变量，则会备份所有系统框架。	
$AC_MEAS_NCBFR *)	数据管理中的全局基本框架的位标记 若未写入此变量，则会备份所有全局基本框架。	
$AC_MEAS_CHBFR *)	数据管理中的通道基本框架的位标记 若未写入此变量，则会备份所有通道基本框架。	
$AC_MEAS_UIFR *)	数据管理中的可设定框架的数量 若未写入此变量，则会备份所有可设定框架。	
$AC_MEAS_TYPE	= 26	测量类型 26：备份数据管理框架

*) 可选配

10.5.2.18 测量类型 27：恢复已备份的数据管理框架

功能

使用此测量类型，可将通过测量类型 26 备份的数据管理框架重新写入到动态用户内存。

可恢复所有最近备份的框架，或只恢复其中选择的框架。若选择了未备份的框架，此设定会被忽略，且不会使恢复进程终止。

用于测量类型 27，将对以下系统变量数值进行评估：

输入变量	含义	
$AC_MEAS_VALID	输入变量的有效位	
$AC_MEAS_CHSFR *)	数据管理中的系统框架的位标记 若未写入此变量，则将恢复所有系统框架。	
$AC_MEAS_NCBFR *)	数据管理中的全局基本框架的位标记 若未写入此变量，则将恢复所有全局基本框架。	
$AC_MEAS_CHBFR *)	数据管理中的通道基本框架的位标记 若未写入此变量，则将恢复所有通道基本框架。	
$AC_MEAS_UIFR *)	数据管理中的可设定框架的数量 若未写入此变量，则将恢复所有可设定框架。	
$AC_MEAS_TYPE	= 27	测量类型 27：恢复已备份的数据管理框架

*) 可选配

10.5.2.19 测量类型 28：用于锥形体车削的附加车削定义

功能

使用测量类型 28 可为生效平面或特定平面设定一个角度为 α 的叠加旋转。

数值范围：$-90° \le \alpha \le +90°$

旋转围绕与平面相垂直的坐标轴进行。

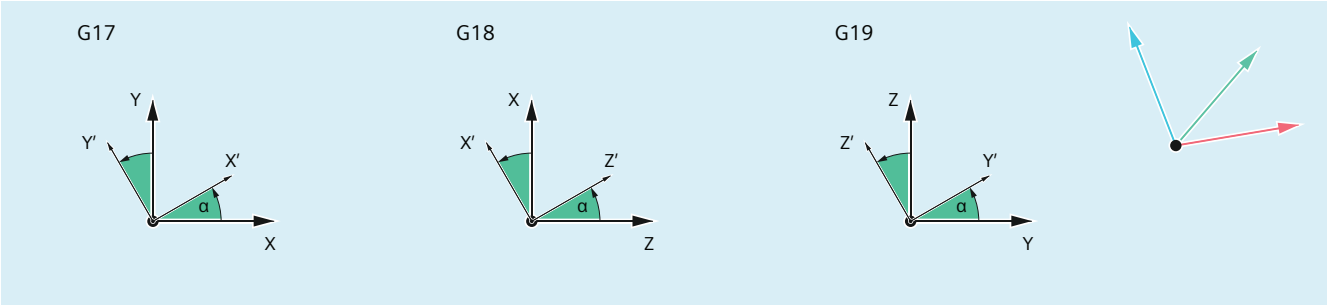


图 10-16　在平面 G17 / G18 / G19 内指定一个用于锥形体旋转的叠加旋转(α = +30°)

应用

在锥面车削中，生效平面会旋转特定锥角，此时旋转会被写入生效的循环框架。RESET 时循环框架会被清除。必要时可重新激活。循环框架的选择取决于 SZS 位置显示。如旋转激活后，例如在生效平面 G18 内在方向 Z 上移动，则用于 X 和 Z 的相应轴实际位置将同时更改。生效平面 G17 和 G19 的旋转将具有类似特性，并在上图中显示。

输入/输出变量

用于测量类型 28，将对以下系统变量数值进行评估：

输入变量	含义	
$AC_MEAS_VALID	输入变量的有效位	
$AC_MEAS_WP_SETANGLE	设定角度	
$AC_MEAS_ACT_PLANE *)	未设定时将围绕生效的平面旋转。	
$AC_MEAS_FRAME_SELECT *)	未设定时将计算叠加框架 。	
$AC_MEAS_INPUT[0] *)	= 1	锥面车削生效。
$AC_MEAS_TYPE	= 28	测量类型 28：用于锥形体车削的附加车削定义

*) 可选配

对于测量类型 28，将写入以下输出变量：

输出变量	含义
$AC_MEAS_FRAME	结果，包含旋转

10.5.3 刀具测量的测量类型

10.5.3.1 概述

为了在车床或铣床上对换入的刀具执行测量，可使用以下测量类型：

测量类型 $AC_MEAS_TYPE =	测量类型
10	测量刀具长度 (页 721)
11	测量刀具直径 (页 724)
22	通过放大镜功能测量刀具长度 (页 726)
23	通过标记位置或当前位置测量刀具长度 (页 727) 通过定向测量两把刀具的长度 (页 728)

10.5.3.2 测量类型 10：测量刀具长度

功能

可在一个已经过测量的参考件上进行刀具长度测量。

平面选择取决于刀具位置：

- G17 用于 Z 轴方向的刀具位置
- G18 用于 Y 轴方向的刀具位置
- G19 用于 X 轴方向的刀具位置

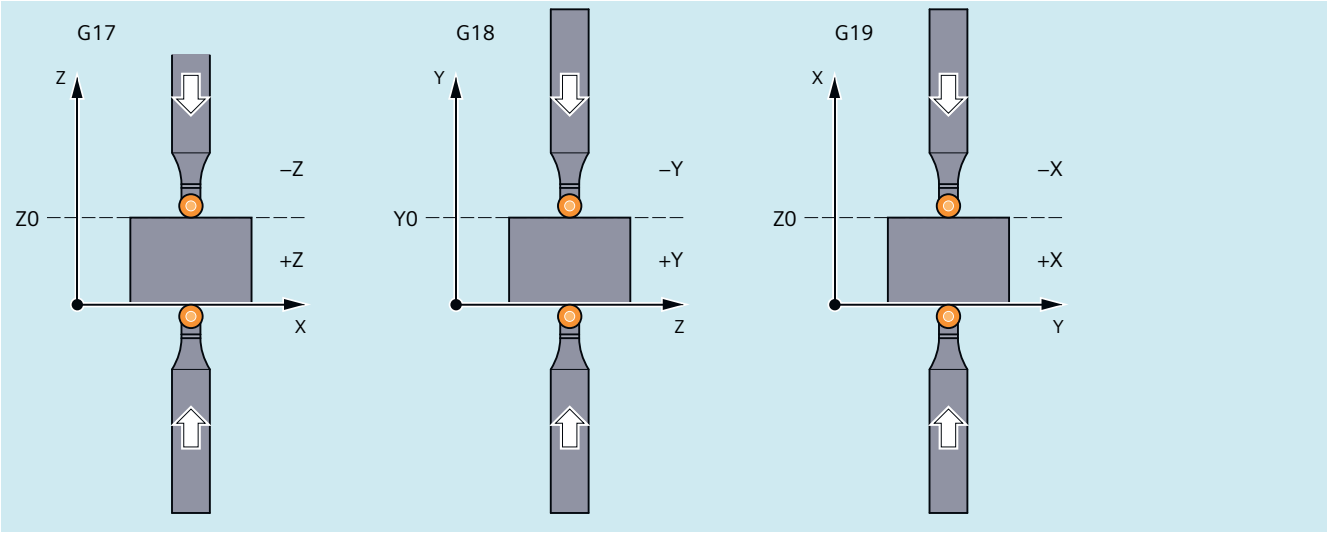


图 10-17　测量 G17/G18/ G19 平面中的刀具长度

用于测量类型 10，将对以下系统变量数值进行评估：

输入变量	含义	
$AC_MEAS_VALID	输入变量的有效位	
$AA_MEAS_POINT1[<Axis>]	测量点 1	
$AC_MEAS_P1_COORD *)	测量点的坐标系	
$AA_MEAS_SETPOINT[<Axis>]	设定位置 Z0	
$AC_MEAS_SET_COORD *)	设定点的坐标系	
$AC_MEAS_DIR_APPROACH	移动到工件上	
	= 0	+ X
	= 1	- X
	= 2	+ Y
	= 3	- Y
	= 4	+ Z
	= 5	- Z
$AC_MEAS_ACT_PLANE *)	未设定时将采用当前生效的平面。	
$AC_MEAS_TYPE	= 10	测量类型 10：测量刀具长度

*) 可选配

对于测量类型 10，将写入以下输出变量：

输出变量	含义
$AC_MEAS_TOOL_LENGTH	刀具长度
$AC_MEAS_RESULTS[0]	刀具长度，X 方向
$AC_MEAS_RESULTS[1]	刀具长度，Y 方向
$AC_MEAS_RESULTS[2]	刀具长度，Z 方向
$AC_MEAS_RESULTS[3]	刀具长度 L1
$AC_MEAS_RESULTS[4]	刀具长度 L2
$AC_MEAS_RESULTS[5]	刀具长度 L3

示例

```
DEF INT RETVAL
T0 D0
G0 X0 Y0 Z0 F10000
                                    ; 测量刀具长度。
$AC_MEAS_VALID=0                    ; 将所有输入值设置为无效。
G1 Z10                              ; 将刀具移动至参考件。

$AC_MEAS_LATCH[0]=1                 ; 获取测量点 1。
$AC_MEAS_DIR_APPROACH=5             ; 设置移动靠近方向 -Z。
$AA_MEAS_SETPOINT[X]=0              ; 设置参考位置。
$AA_MEAS_SETPOINT[Y]=0
$AA_MEAS_SETPOINT[Z]=0
$AC_MEAS_ACT_PLANE=0                ; 测量平面为 G17。
$AC_MEAS_T_NUMBER=0                 ; 未选择刀具。
$AC_MEAS_D_NUMBER=0
$AC_MEAS_TYPE=10                    ; 将测量类型设置为刀具长度。

RETVAL=MEASURE()                    ; 开始测量。
IF RETVAL <> 0
SETAL(61043, << RETVAL)
ENDIF
IF $AC_MEAS_TOOL_LENGTH <> 10       ; 查询已知刀具长度。
SETAL(61043, << $AC_MEAS_TOOL_LENGTH)
ENDIF
M30
```

10.5.3.3 测量类型 11：测量刀具直径

功能

可在一个已测量的参考件上进行刀具长度测量。

平面选择取决于刀具位置：

- G17 用于 Z 轴方向的刀具位置
- G18 用于 Y 轴方向的刀具位置
- G19 用于 X 轴方向的刀具位置

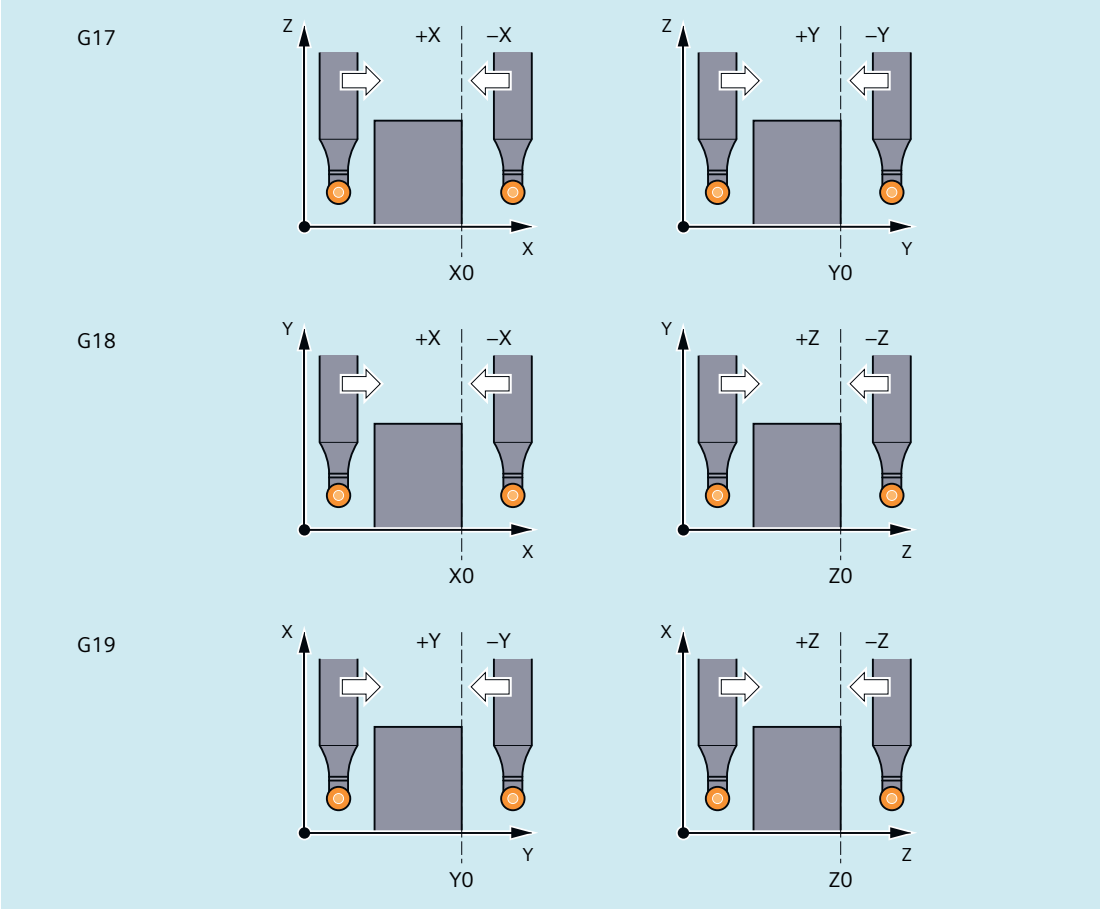


图 10-18　测量 G17/G18/ G19 平面中的刀具直径

用于测量类型 11，将对以下系统变量数值进行评估：

输入变量	含义	
$AC_MEAS_VALID	输入变量的有效位	
$AA_MEAS_POINT1[<Axis>]	测量点 1	
$AA_MEAS_SETPOINT[<Axis>]	设定位置 X0	
$AC_MEAS_DIR_APPROACH	移动到工件上	
	= 0	+ X
	= 1	- X
	= 2	+ Y
	= 3	- Y
	= 4	+ Z
	= 5	- Z
$AC_MEAS_ACT_PLANE *)	未设定时将采用当前生效的平面。	
$AC_MEAS_TYPE	= 11	测量类型 11：测量刀具直径

*) 可选配

对于测量类型 11，将写入以下输出变量：

输出变量	含义
$AC_MEAS_TOOL_DIAMETER	刀具直径

10.5.3.4 测量类型 22：通过放大镜功能测量刀具长度

功能

如果机床配备了放大镜功能，可通过其测定刀具长度。

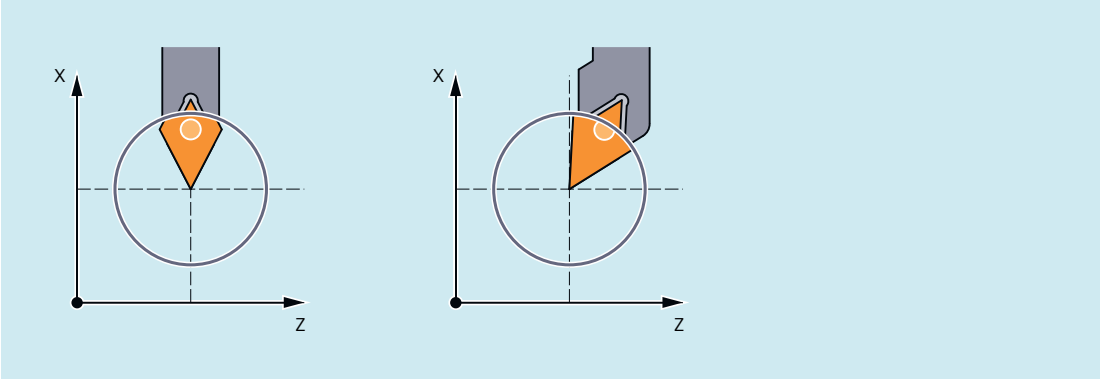


图 10-19　通过放大镜功能测量刀具长度

用于测量类型 22，将对以下系统变量数值进行评估：

输入变量	含义	
$AC_MEAS_VALID	输入变量的有效位	
$AA_MEAS_POINT1[<Axis>]	所有通道轴的第 1 测量点	
$AC_MEAS_P1_COORD *)	测量点的坐标系	
$AA_MEAS_SETPOINT[<Axis>]	必须设定放大镜位置 X 和 Z。	
$AC_MEAS_SET_COORD *)	设定点的坐标系	
$AC_MEAS_ACT_PLANE *)	未设定时将采用当前生效的平面。	
$AC_MEAS_FRAME_SELECT *)	未设定时将计算叠加框架 。	
$AC_MEAS_T_NUMBER *)	未设定时将采用当前生效的 T（T0）进行计算。	
$AC_MEAS_D_NUMBER *)	未设定时将采用当前生效的 D（D0）进行计算。	
$AC_MEAS_TYPE	= 22	测量类型 22：通过放大镜功能测量刀具长度

*) 可选配

对于测量类型 22，将写入以下输出变量：

输出变量	含义
$AC_MEAS_RESULT[0]	刀具长度，X 方向
$AC_MEAS_RESULT[1]	刀具长度，Y 方向

输出变量	含义
$AC_MEAS_RESULT[2]	刀具长度，Z 方向
$AC_MEAS_RESULT[3]	刀具长度 L1
$AC_MEAS_RESULT[4]	刀具长度 L2
$AC_MEAS_RESULT[5]	刀具长度 L3

10.5.3.5 测量类型 23：通过标记位置或当前位置测量刀具长度

功能

手动测量中可测定 X 轴方向和 Z 轴方向的刀具尺寸。ShopTurn 通过已知的刀架参考点位置以及测量头位置计算刀具校正数据。

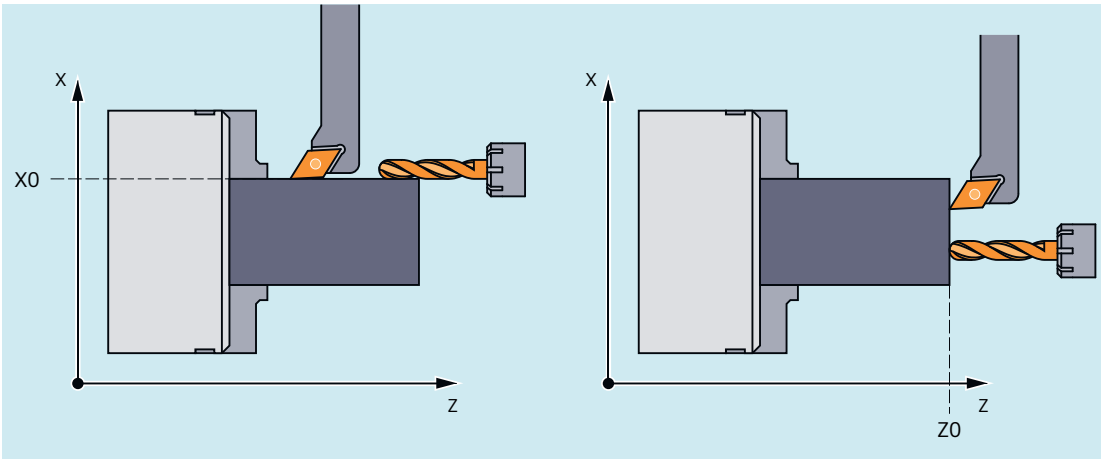


图 10-20 通过标记位置或当前位置测量刀具长度

用于测量类型 23，将对以下系统变量数值进行评估：

输入变量	含义
$AC_MEAS_VALID	输入变量的有效位
$AA_MEAS_POINT1[<Axis>]	当前位置或标记位置
$AC_MEAS_P1_COORD *)	测量点的坐标系
$AA_MEAS_SETPOINT[<Axis>]	设定位置 （至少须设定一根几何轴）
$AC_MEAS_SET_COORD *)	设定点的坐标系
$AC_MEAS_ACT_PLANE *)	未设定时将采用当前生效的平面。
$AC_MEAS_T_NUMBER *)	未设定时将采用当前生效的 T（T0）进行计算。
$AC_MEAS_D_NUMBER *)	未设定时将采用当前生效的 D（D0）进行计算。

输入变量	含义	
$AC_MEAS_TOOL_MASK *)	刀具长度，半径	
$AC_MEAS_DIR_APPROACH *)	移动到工件上	
	= 0	+ X
	= 1	- X
	= 2	+ Y
	= 3	- Y
	= 4	+ Z
	= 5	- Z
$AC_MEAS_INPUT[0] *)	= 1	计算出的刀具长度写入数据管理。
$AC_MEAS_TYPE	= 23	测量类型 23：

*) 可选配

对于测量类型 23，将写入以下输出变量：

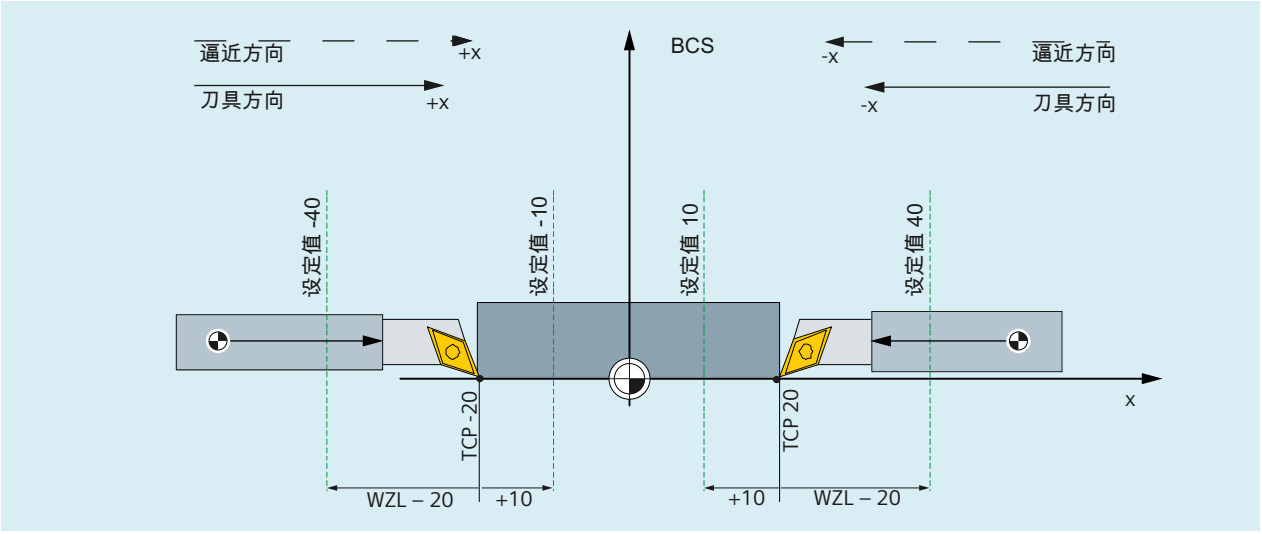
输出变量	含义
$AC_MEAS_RESULT[0]	刀具长度，X 方向
$AC_MEAS_RESULT[1]	刀具长度，Y 方向
$AC_MEAS_RESULT[2]	刀具长度，Z 方向
$AC_MEAS_RESULT[3]	刀具长度 L1
$AC_MEAS_RESULT[4]	刀具长度 L2
$AC_MEAS_RESULT[5]	刀具长度 L3

10.5.3.6 测量类型 23：通过定向测量两把刀具的长度

刀具定向

对于向刀架定向的刀具，系统变量 $AC_MEAS_TOOL_MASK 位 9 必须置 1（0x200）。之后计算出的刀具长度将取反计入。

启用各自参考点的两把车刀，刀具定向沿逼近方向



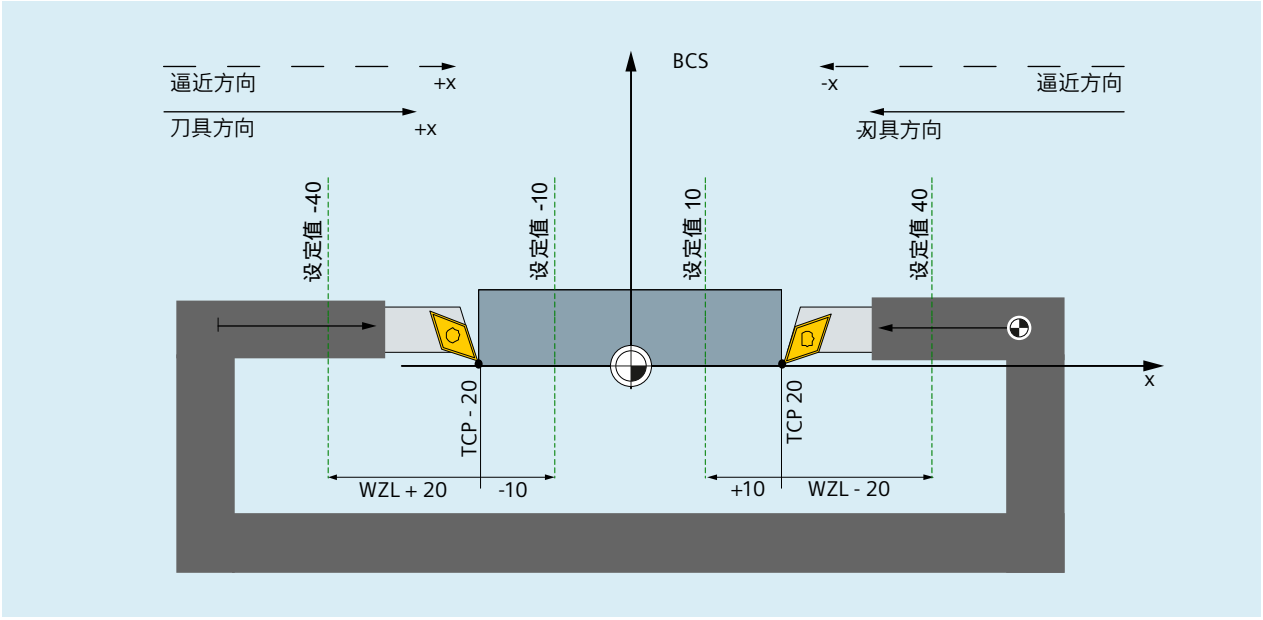
系统数据中的设置：

左侧刀具：逼近方向和刀具定向 +x	
系统变量	**含义**
$AC_MEAS_TOOL_MASK = 0x2	刀具位置处于 x 轴正向（G19）
$AC_MEAS_DIR_APPROACH = 0	逼近方向为 x 轴正向

右侧刀具：逼近方向和刀具定向 -x	
$AC_MEAS_TOOL_MASK = 0x40	刀具位置处于 x 轴负向
$AC_MEAS_DIR_APPROACH = 1	逼近方向为 x 轴负向

针对两把刀具	
$AC_MEAS_Px_COORD = 1	第 x 个测量点的坐标系 = BCS
$AC_MEAS_SET_COORD = 1	设定点的坐标系 = BCS

启用一个参考点的两把车刀，刀具定向与逼近方向相反

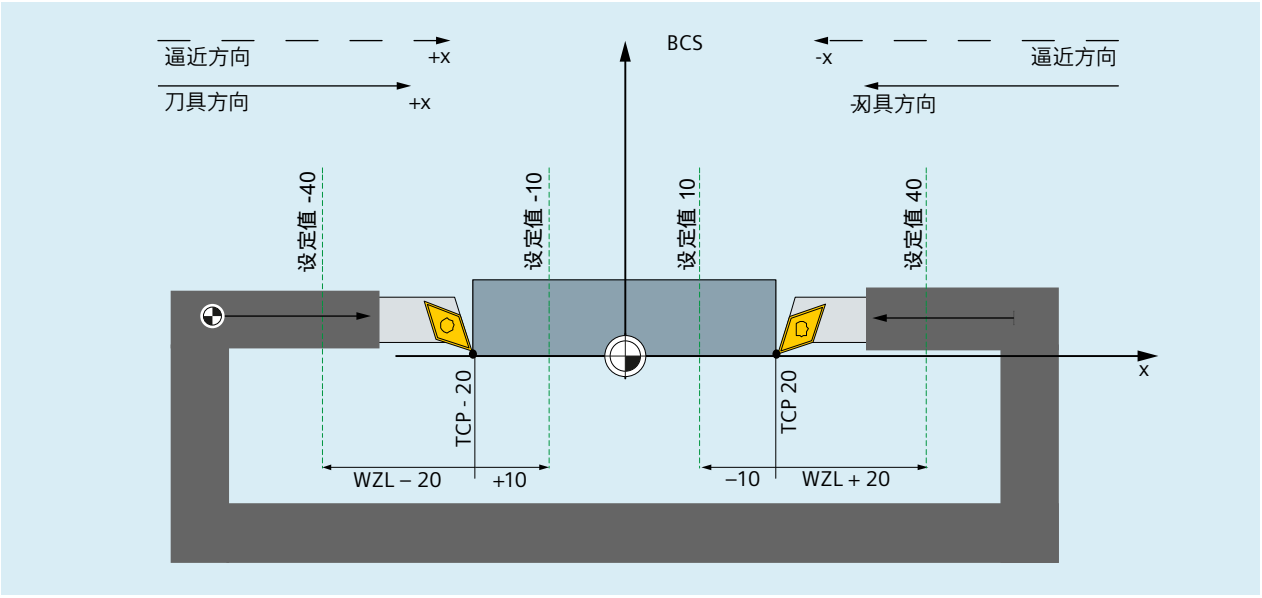


系统数据中的设置：

左侧刀具：逼近方向和刀具定向 +x	
系统变量	**含义**
$AC_MEAS_TOOL_MASK = 0x2 + 0x200	刀具位置处于 x 轴正向（G19）+ 刀具长度差值取反计入
$AC_MEAS_DIR_APPROACH = 0	逼近方向为 x 轴正向

右侧刀具：逼近方向和刀具定向 -x	
$AC_MEAS_TOOL_MASK = 0x40	刀具位置处于 x 轴负向
$AC_MEAS_DIR_APPROACH = 1	逼近方向为 x 轴负向

针对两把刀具	
$AC_MEAS_Px_COORD = 1	第 x 个测量点的坐标系 = BCS
$AC_MEAS_SET_COORD = 1	设定点的坐标系 = BCS



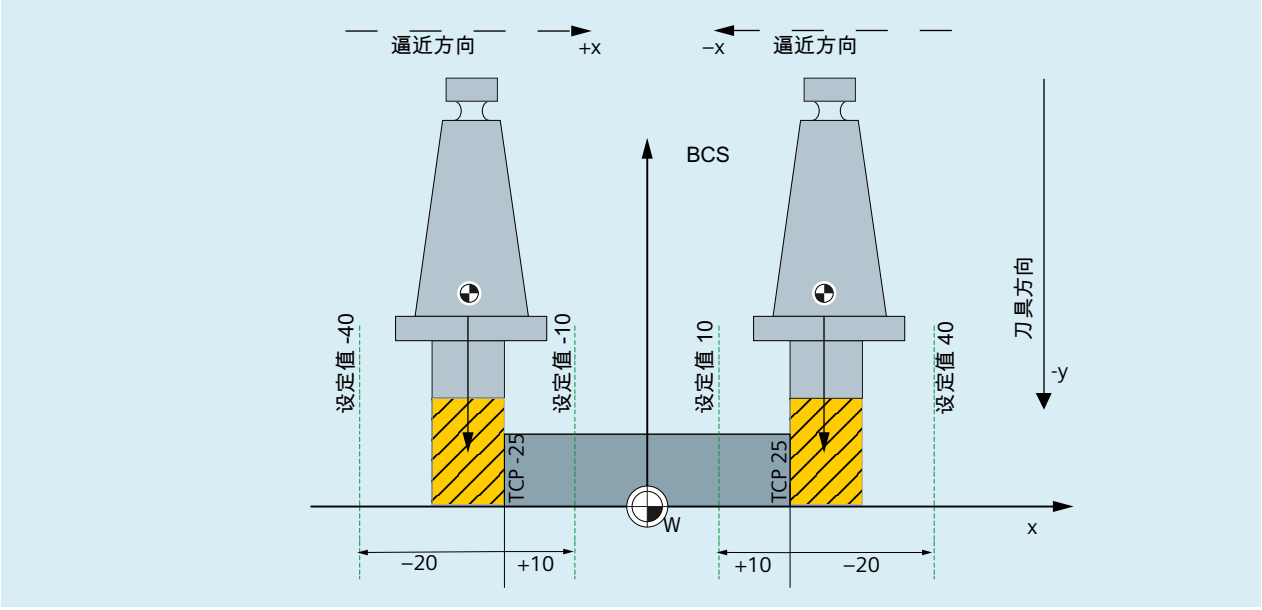
系统数据中的设置：

左侧刀具：逼近方向和刀具定向 +x	
系统变量	**含义**
$AC_MEAS_TOOL_MASK = 0x2	刀具位置处于 x 轴正向（G19）
$AC_MEAS_DIR_APPROACH = 0	逼近方向为 x 轴正向

右侧刀具：逼近方向和刀具定向 -x	
$AC_MEAS_TOOL_MASK = 0x40 + 0x200	刀具位置处于 x 轴负向 + 刀具长度差值取反计入
$AC_MEAS_DIR_APPROACH = 1	逼近方向为 x 轴负向

针对两把刀具	
$AC_MEAS_Px_COORD = 1	第 x 个测量点的坐标系 = BCS
$AC_MEAS_SET_COORD = 1	设定点的坐标系 = BCS

启用各自参考点的两把铣刀，刀具定向垂直于逼近方向



系统数据中的设置：

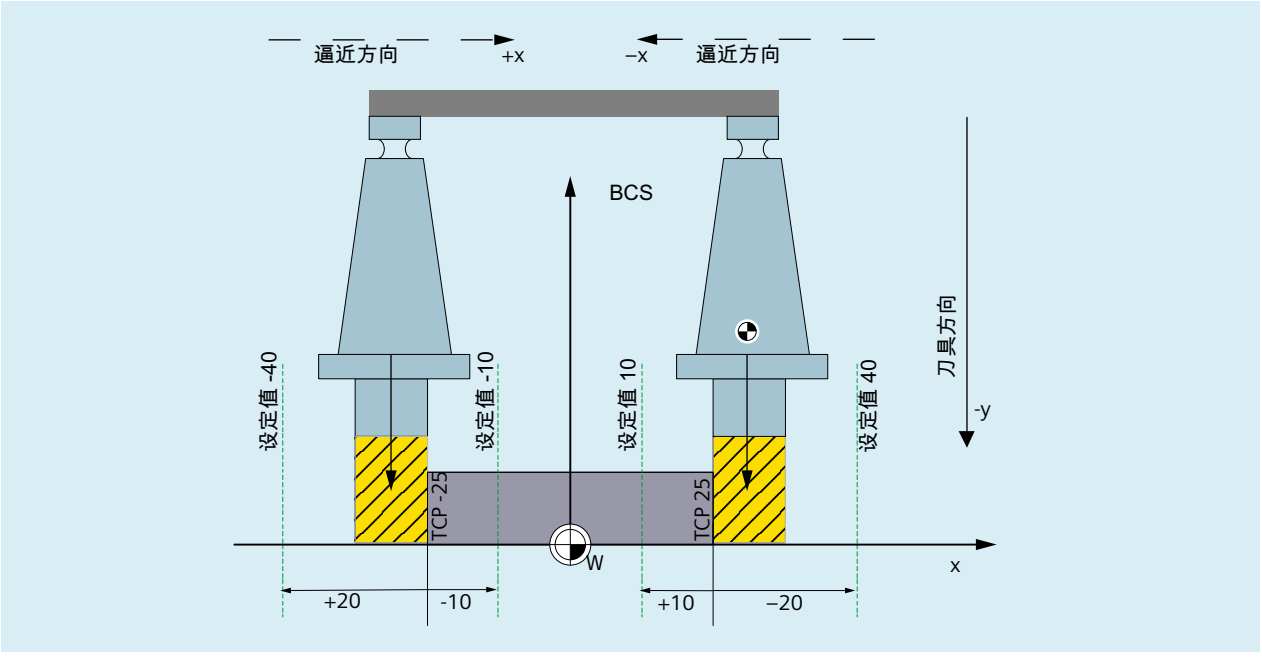
左侧刀具：逼近方向为 x 轴正向和刀具定向为 y 轴负向	
系统变量	**含义**
$AC_MEAS_TOOL_MASK = 0x80	刀具位置处于 y 轴负向
$AC_MEAS_DIR_APPROACH = 0	逼近方向为 x 轴正向

右侧刀具：逼近方向为 x 轴负向和刀具定向为 y 轴负向	
$AC_MEAS_TOOL_MASK = 0x80	刀具位置处于 y 轴负向
$AC_MEAS_DIR_APPROACH = 1	逼近方向为 x 轴负向

针对两把刀具	
$AC_MEAS_Px_COORD = 1	第 x 个测量点的坐标系 = BCS
$AC_MEAS_SET_COORD = 1	设定点的坐标系 = BCS

启用一个参考点的两把铣刀，刀具定向垂直于逼近方向

启用一个参考点的两把铣刀，刀具定向为 -y

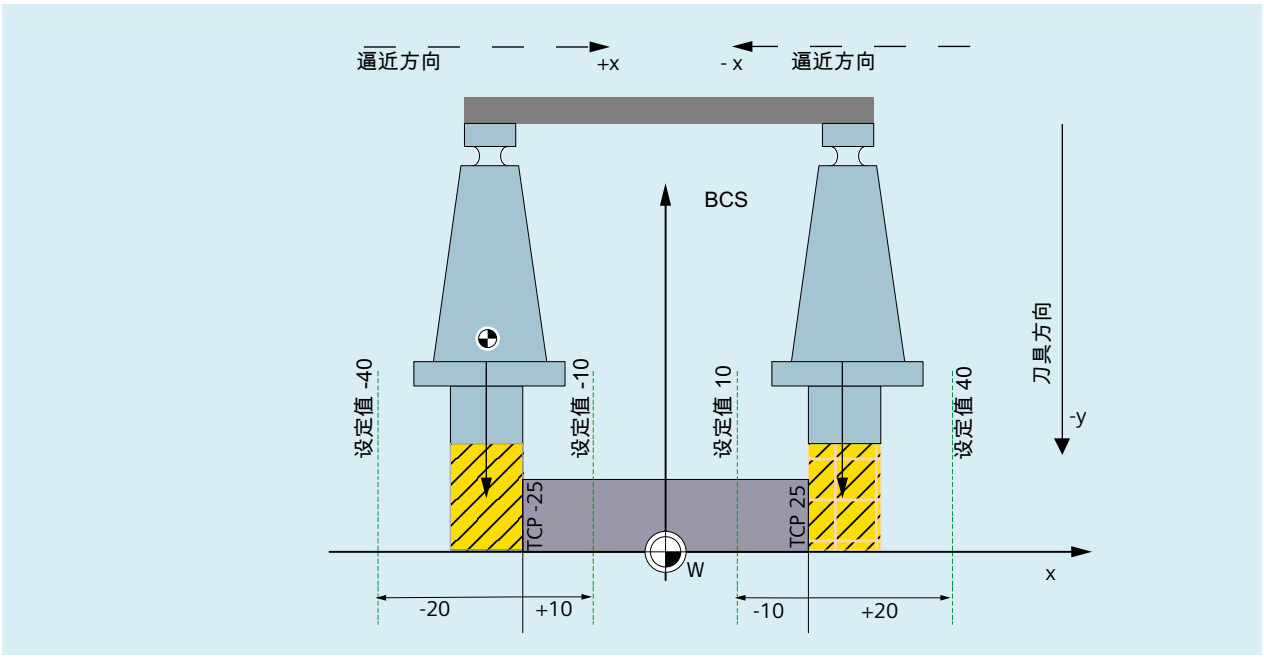


使用现有配置时，刀具位置 $AC_MEAS_TOOL_MASK 和向工件的逼近方向 $AC_MEAS_DIR_APPROACH 必须如下设置：

左侧刀具：逼近方向为 x 轴正向和刀具定向为 y 轴负向	
系统变量	**含义**
$AC_MEAS_TOOL_MASK = 0x80 + 0x200	刀具位置处于 y 轴负向 + 刀具长度差值取反计入
$AC_MEAS_DIR_APPROACH = 0	逼近方向为 x 轴正向

右侧刀具：逼近方向为 x 轴负向和刀具定向为 y 轴负向	
$AC_MEAS_TOOL_MASK = 0x80	刀具位置处于 y 轴负向
$AC_MEAS_DIR_APPROACH = 1	逼近方向为 x 轴负向

针对两把刀具	
$AC_MEAS_Px_COORD = 1	第 x 个测量点的坐标系 = BCS
$AC_MEAS_SET_COORD = 1	设定点的坐标系 = BCS



使用现有配置时，刀具位置 $AC_MEAS_TOOL_MASK 和向工件的逼近方向 $AC_MEAS_DIR_APPROACH 必须如下设置：

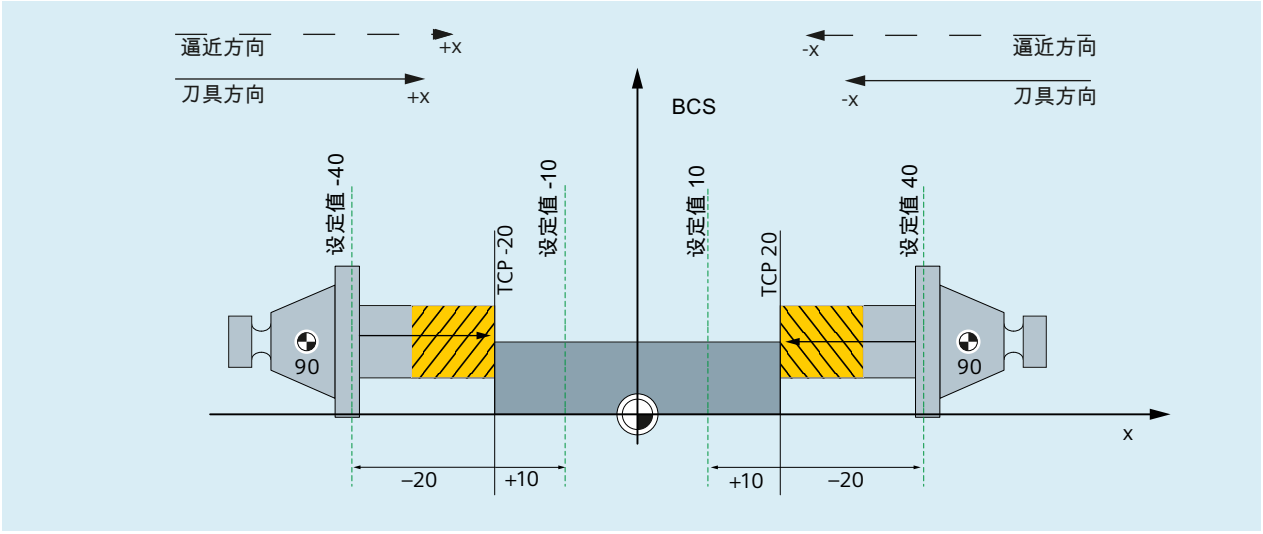
左侧刀具：逼近方向为 x 轴正向和刀具定向为 y 轴负向	
系统变量	**含义**
$AC_MEAS_TOOL_MASK = 0x80	刀具位置处于 y 轴负向
$AC_MEAS_DIR_APPROACH = 0	逼近方向为 x 轴正向

右侧刀具：逼近方向为 x 轴负向和刀具定向为 y 轴负向	
$AC_MEAS_TOOL_MASK = 0x80 + 0x200	刀具位置处于 y 轴负向 + 刀具长度差值取反计入
$AC_MEAS_DIR_APPROACH = 1	逼近方向为 x 轴负向

针对两把刀具	
$AC_MEAS_Px_COORD = 1	第 x 个测量点的坐标系 = BCS
$AC_MEAS_SET_COORD = 1	设定点的坐标系 = BCS

启用各自参考点的两把铣刀，刀具定向与逼近方向相反

启用各自参考点的两把铣刀，刀具定向沿逼近方向



使用现有配置时，刀具位置 $AC_MEAS_TOOL_MASK 和向工件的逼近方向 $AC_MEAS_DIR_APPROACH 必须如下设置：

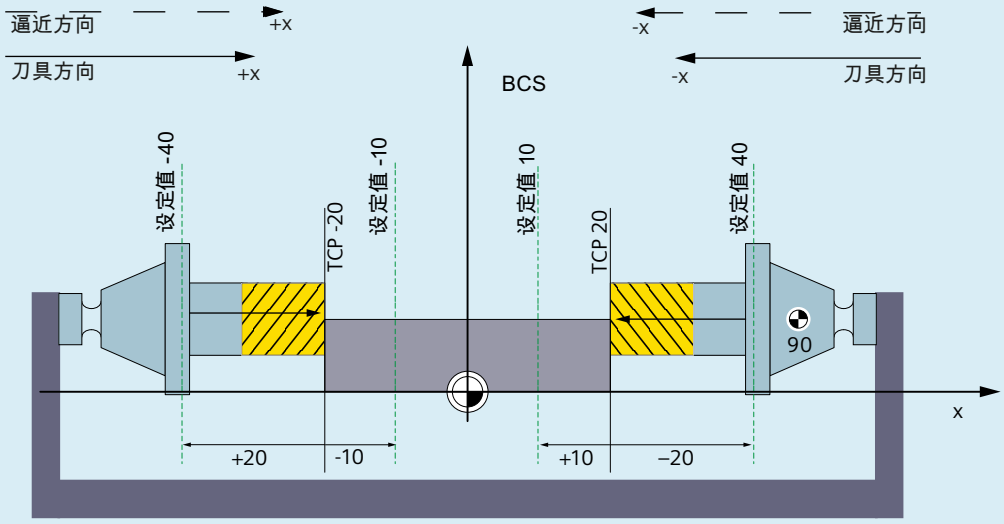
左侧刀具：逼近方向和刀具定向 +x	
系统变量	**含义**
$AC_MEAS_TOOL_MASK = 0x2	刀具位置处于 x 轴正向（G19）
$AC_MEAS_DIR_APPROACH = 0	逼近方向为 x 轴正向

右侧刀具：逼近方向和刀具定向 -x	
$AC_MEAS_TOOL_MASK = 0x40	刀具位置处于 x 轴负向
$AC_MEAS_DIR_APPROACH = 1	逼近方向为 x 轴负向

针对两把刀具	
$AC_MEAS_Px_COORD = 1	第 x 个测量点的坐标系 = BCS
$AC_MEAS_SET_COORD = 1	设定点的坐标系 = BCS

启用一个参考点的两把铣刀，刀具定向与逼近方向相反

启用一个参考点的两把铣刀，刀具位置与定向相反

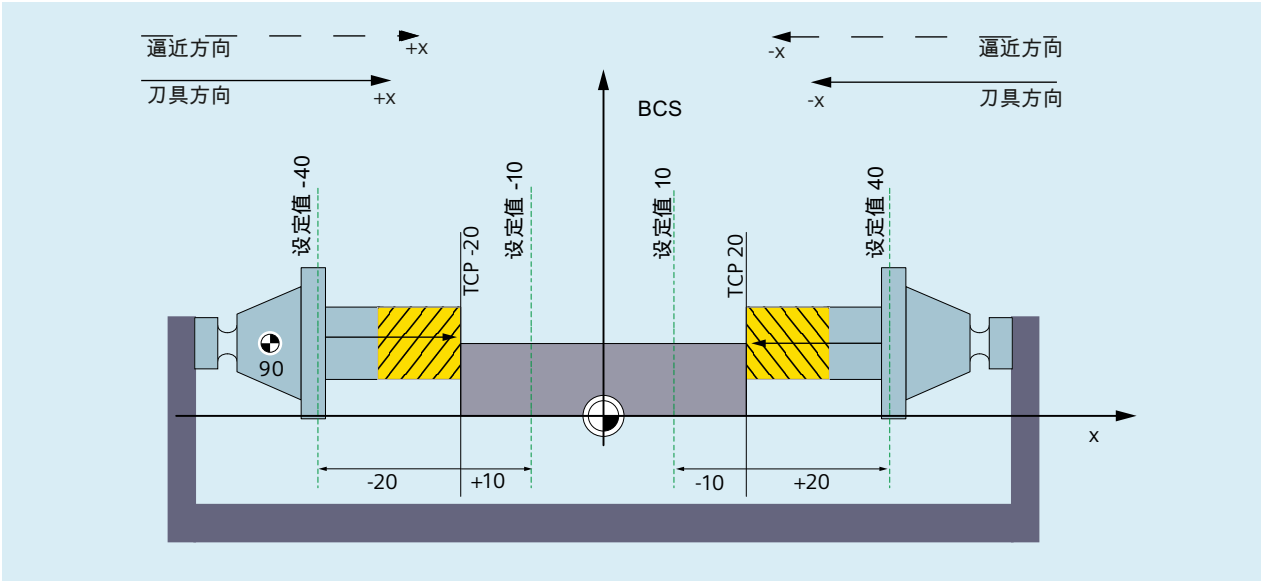


使用现有配置时，刀具位置 $AC_MEAS_TOOL_MASK 和向工件的逼近方向 $AC_MEAS_DIR_APPROACH 必须如下设置：

左侧刀具：逼近方向和刀具定向 +x	
系统变量	**含义**
$AC_MEAS_TOOL_MASK = 0x2 + 0x200	刀具位置处于 x 轴正向（G19）+ 刀具长度差值取反计入
$AC_MEAS_DIR_APPROACH = 0	逼近方向为 x 轴正向

右侧刀具：逼近方向和刀具定向为 x 轴负向	
$AC_MEAS_TOOL_MASK = 0x40	刀具位置处于 x 轴负向
$AC_MEAS_DIR_APPROACH = 1	逼近方向为 x 轴负向

针对两把刀具	
$AC_MEAS_Px_COORD = 1	第 x 个测量点的坐标系 = BCS
$AC_MEAS_SET_COORD = 1	设定点的坐标系 = BCS



使用现有配置时，刀具位置 $AC_MEAS_TOOL_MASK 和向工件的逼近方向 $AC_MEAS_DIR_APPROACH 必须如下设置：

左侧刀具：逼近方向和刀具定向 +x	
系统变量	**含义**
$AC_MEAS_TOOL_MASK = 0x2	刀具位置处于 x 轴正向（G19）
$AC_MEAS_DIR_APPROACH = 0	逼近方向为 x 轴正向

右侧刀具：逼近方向和刀具定向为 x 轴负向	
$AC_MEAS_TOOL_MASK = 0x40 + 0x200	刀具位置处于 x 轴负向 + 刀具长度差值取反计入
$AC_MEAS_DIR_APPROACH = 1	逼近方向为 x 轴负向

针对两把刀具	
$AC_MEAS_Px_COORD = 1	第 x 个测量点的坐标系 = BCS
$AC_MEAS_SET_COORD = 1	设定点的坐标系 = BCS

WCS 中的各种刀具

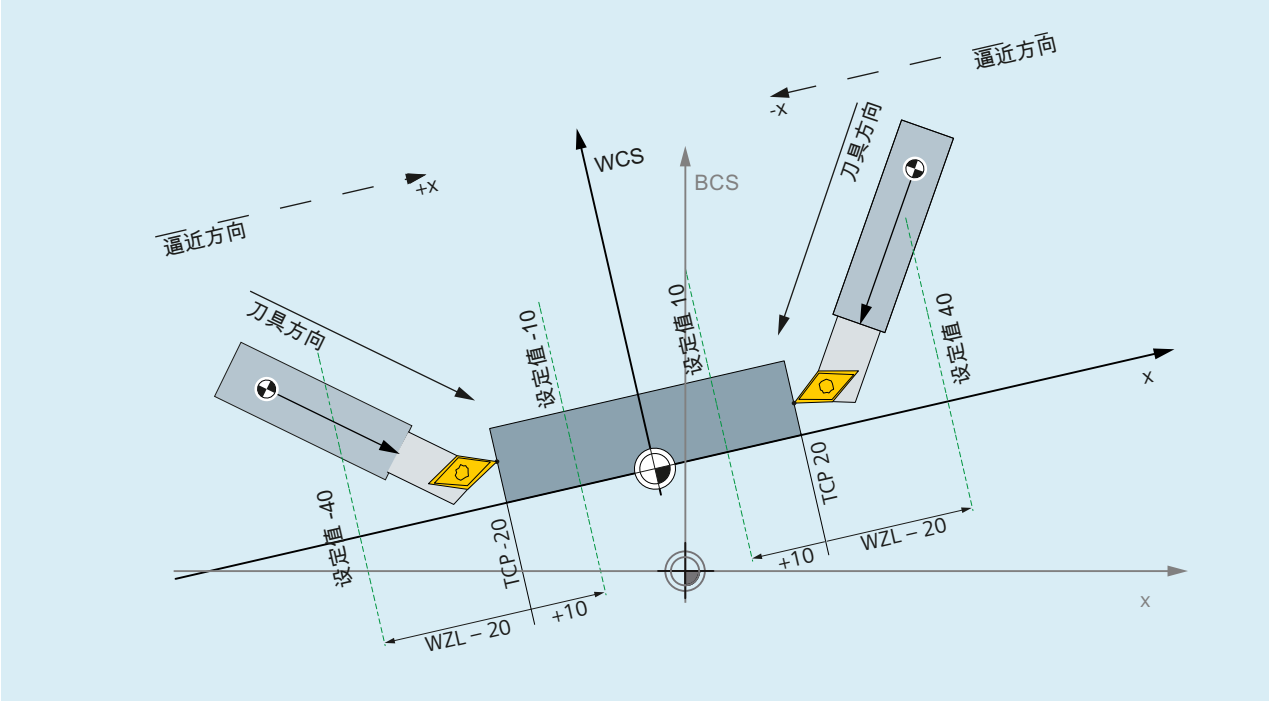


图 10-21　启用各自参考点的两把车刀

系统数据中的设置：

左侧刀具：逼近方向为 x 轴正向和刀具定向为 y 轴负向	
系统变量	**含义**
$AC_MEAS_TOOL_MASK = 0x0	所有刀具长度均计入 （缺省设置）
$AC_MEAS_DIR_APPROACH = 0	逼近方向为 x 轴正向

右侧刀具：逼近方向为 x 轴负向和刀具定向为 y 轴负向	
$AC_MEAS_TOOL_MASK = 0x0	所有刀具长度均计入 （缺省设置）
$AC_MEAS_DIR_APPROACH = 1	逼近方向为 x 轴负向

针对两把刀具	
$AC_MEAS_Px_COORD = 0	第 x 个测量点的坐标系 = MKS（缺省设置）
$AC_MEAS_SET_COORD = 0	设定点的坐标系 = WCS（缺省设置）

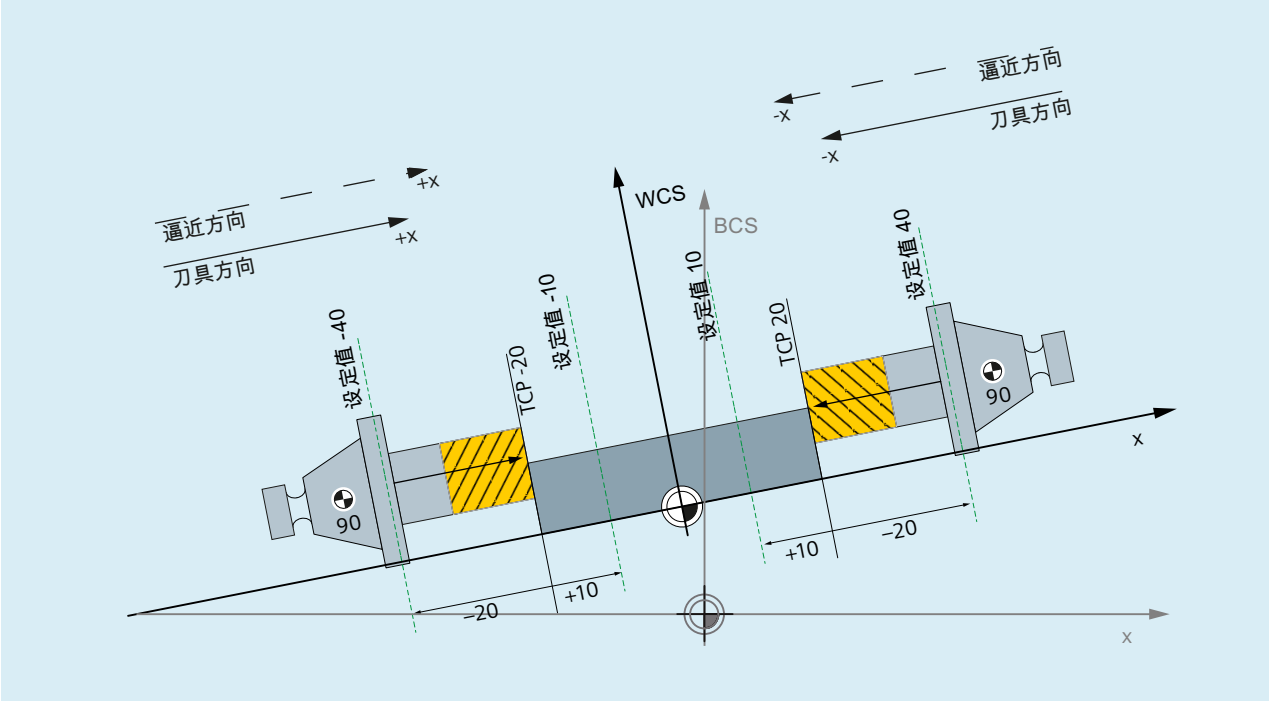


图 10-22　启用各自参考点的两把铣刀

系统数据中的设置：

左侧刀具：逼近方向为 x 轴正向和刀具定向为 y 轴负向	
系统变量	**含义**
$AC_MEAS_TOOL_MASK = 0x80	刀具位置处于 y 轴负向
$AC_MEAS_DIR_APPROACH = 0	逼近方向为 x 轴正向

右侧刀具：逼近方向为 x 轴负向和刀具定向为 y 轴负向	
$AC_MEAS_TOOL_MASK = 0x80	刀具位置处于 y 轴负向
$AC_MEAS_DIR_APPROACH = 1	逼近方向为 x 轴负向

针对两把刀具	
$AC_MEAS_Px_COORD = 0	第 x 个测量点的坐标系 = MKS（缺省设置）
$AC_MEAS_SET_COORD = 0	设定点的坐标系 = WCS（缺省设置）

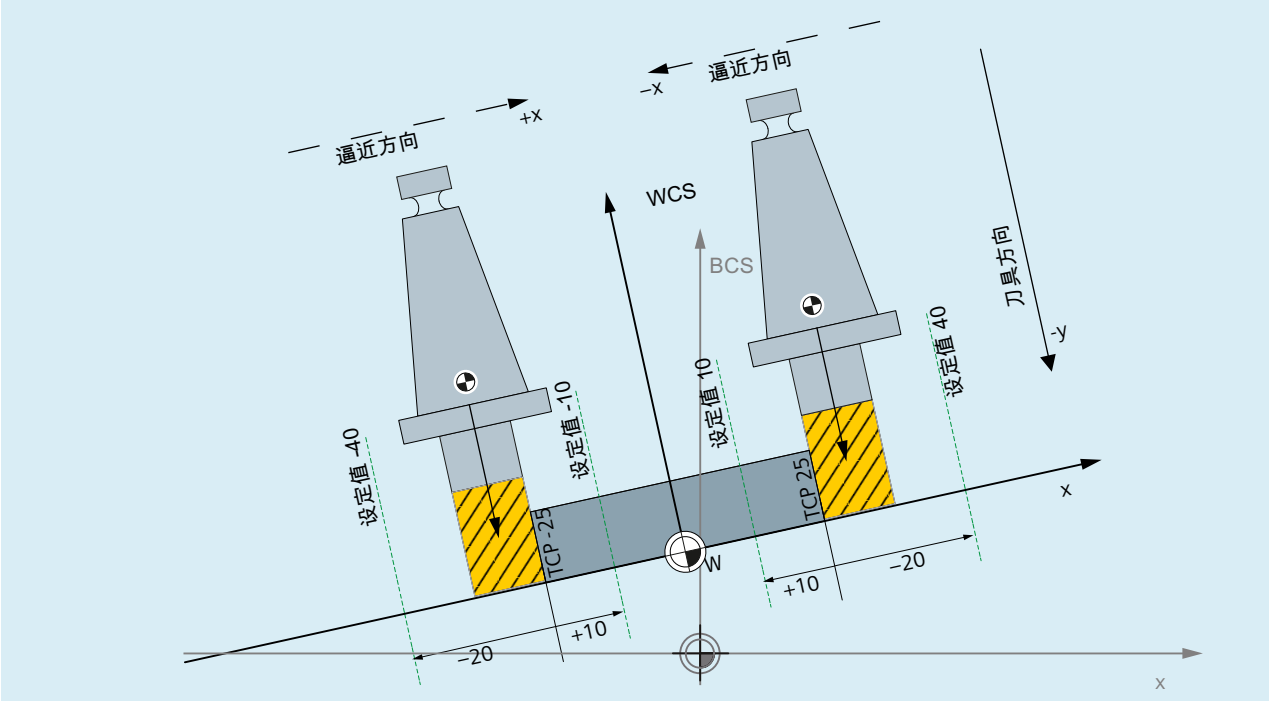


图 10-23　旋转 90 度、启用各自参考点的两把铣刀

系统数据中的设置：

左侧刀具：逼近方向为 x 轴正向和刀具定向为 y 轴负向	
系统变量	**含义**
$AC_MEAS_TOOL_MASK = 0x2	刀具位置处于 x 轴正向（G19）
$AC_MEAS_DIR_APPROACH = 0	逼近方向为 x 轴正向

右侧刀具：逼近方向为 x 轴负向和刀具定向为 y 轴负向	
$AC_MEAS_TOOL_MASK = 0x40	刀具位置处于 x 轴负向
$AC_MEAS_DIR_APPROACH = 1	逼近方向为 x 轴负向

针对两把刀具	
$AC_MEAS_Px_COORD = 0	第 x 个测量点的坐标系 = MKS（缺省设置）
$AC_MEAS_SET_COORD = 0	设定点的坐标系 = WCS（缺省设置）

10.6 测量精度和测试

10.6.1 测量精度

测量精度受下列参数影响：

- 测量信号延时 (T_{Delay})
- 测量过程中的运行速度 (v_M)

测量信号延时补偿 (T_{Delay})

测量信号延时（即从触发测头到将测量值保存到控制系统中的时间）取决于测头的响应时间及控制系统硬件的信号响应时间。测量时会对控制系统延时进行补偿。为此，须算出延时并将结果输入到以下机床数据中：

MD13220 $MN_MEAS_PROBE_DELAY_TIME = <算出的延时>

说明

最大补偿延时 $T_{MaxDelay}$

$T_{MaxDelay}$ = 31 * 位置控制周期或 DP 周期
对超过 $T_{MaxDelay}$ 的延时进行补偿没有任何的意义。因此，超限的延时会被限制为 $T_{MaxDelay}$。

测量过程中的最大运行速度 (v_M)

测量过程中允许的最大运行速度取决于编写的测量脉冲沿的数量以及位置控制周期与 DP 周期之间的比例。

为得到正确的结果，测量过程中必须选择满足以下条件（每**两**个位置控制周期或 DP 周期）的运行速度：

- 最多**一个相同**的触发信号，即：**一个**测头有一个正脉冲沿**或**一个负脉冲沿。
- 最多**四个不同**的触发信号，即：**两个**测头各有一个正脉冲沿和一个负脉冲沿。

10.6.2 测头功能测试

可通过一个 NC 程序根据以下模板对测头的功能进行测试。

程序代码	注释
%_N_PRUEF_MESSTASTER_MPF	
;$PATH=/_N_MPF_DIR	
; 测头连接的检查程序	
N05 DEF INT MTSIGNAL	; 控制状态的标记。
N10 DEF INT ME_NR=1	; 测量输入号码。
N20 DEF REAL MESSWERT_IN_X	
N30 G17 T1 D1	; 预选择用于测量头的刀具校正。
N40 _ANF:G0 G90 X0 F150	; 起始位置和测量速度。
N50 MEAS=ME_NR G1 X100	; X 中的测量输入端 1 处的测量。
N60 STOPRE	
N70 MTSIGNAL=$AC_MEA[1]	; 读取测量输入 1 上的。; 软件切换信号。
N80 IF MTSIGNAL==0 GOTOF _FEHL1	; 信号分析
N90 MESSWERT_IN_X=$AA_MW[X]	; 读入工件坐标中的测量值 。
N95 M0	
N100 M02	
N110 _FEHL1: MSG（“测头未开关！”）	
N120 M0	
N130 M02	

通过诊断菜单“PLC 状态”可在程序结束后检查测量信号。

- <Nc>.basic.in.measuringProbe1Actuated（测头 1 偏转）
- <Nc>.basic.in.measuringProbe2Actuated（测头 2 偏转）

轴的当前测量状态将通过以下 NC/PLC 接口信号显示：

- <Axis>.basic.in.measurementActive（测量生效）

此信号可在所有测量功能中进行评估，并可通过系统变量$AA_MEAACT[<Axis>]在同步动作中可用。

更多信息
功能手册之同步动作分册

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Nc>.basic.in.measuringProbe1Actuated	LBP_NC.E_InspProbe1	DB10.DBX107.0
<Nc>.basic.in.measuringProbe2Actuated	LBP_NC.E_InspProbe2	DB10.DBX107.1
<Axis>.basic.in.measurementActive	LBP_Axis*.E_MeasAct	DB31, ...DBX62.3

10.6.3 重复精度

整个测量系统（机器测量传感器到 NC 的信号传输）测量控制（重复精度）可通过 NC 程序根据以下模板进行测定。

在此程序示例中，将在 X 轴上进行 10 次测量，并将测量值接收到工件坐标轴。

这样可确定非趋势性的、偶然条件下的尺寸偏差。

程序代码	注释
%_N_PRUEF_GENAU_MPF;	
$PATH=/_N_MPF_DIR	
DEF INT SIGNAL, II	; 变量定义
DEF REAL MESSWERT_IN_X[10]	
G17 T1 D1	; 初始条件，
	; 刀具补偿
	; 为测头预先选择
_ANF:G0 X0 F150	; 在测量轴上预定位
MEAS=+1 G1 X100	; 在测量输入 1 上进行测量，此时
	; 切换信号未偏转，
	; 在 X 轴中偏转
STOPRE	; 在之后的结果分析后
	; 停止解码
SIGNAL= $AC_MEA[1]	; 读取测量
	; 输入 1 上的软件切换信号
IF SIGNAL == 0 GOTOF_FEHL1	; 检查接通信号
MESSWERT_IN_X[II]=$AA_MW[X]	; 以工件坐标读取测量值
II=II+1	
IF II<10 GOTOB_ANF	; 重复 10 次
M0	
M02	
_FEHL1:MSG（“测头未切换！”）	
M0	

程序代码	注释
M02	

选择了参数显示（用户自定义变量）后，程序执行过程中都可在 MESSWERT_IN_X[10] 区域读取测量结果。

10.7 仿真测量

10.7.1 一般功能

简要说明

在实际机床上进行测量时，需要连接测头以在特定位置提供切换信号。仿真环境中的测量则不需要使用测头，此时切换位置通过其他方式给定。

仿真测量支持两种切换位置设定方式：

- 位置相关的切换请求：切换位置通过测量程序段中编写的轴结束位置推导得出。
- 外部切换请求：切换位置通过控制数字量输出来确定。

前提条件

若需执行仿真测量，系统中的所有机床轴都必须编写为仿真轴：

- MD30130 $MA_CTRLOUT_TYPE[轴] = 0（仿真设定值）
- MD30240 $MA_ENC_TYPE[轴] = 0（仿真编码器）

10.7.2 位置相关切换请求

功能

“位置相关切换请求”通过以下 NC 专用机床数据选择：

- MD13230 $MN_MEAS_PROBE_SOURCE = 0
- MD13231 $MN_MEAS_PROBE_OFFSET = <位置偏移>

轴切换位置通过测量程序段中编写的轴结束位置和参数设置的位置偏移计算得出：

切换位置[轴] = 结束位置[轴] - 位置偏移

测量程序段中会循环检查是否到达轴的切换位置：

设定位置[轴] ≥ 切换位置[轴]

到达切换位置时，系统会为测头 1 和测头 2 生成切换信号上升沿。一个位置控制周期后将生成下降沿。

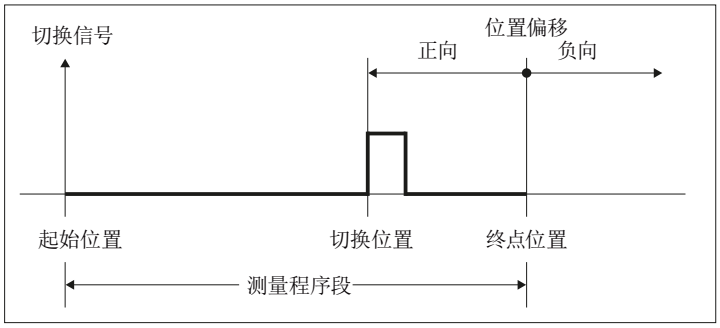


图 10-24　位置相关切换请求

测量值为测量程序段中编写的切换信号（上升沿/ 下降沿）出现时的轴实际值。

若在一个测量程序段中编写了多根轴，则会通过针对特定轴的位置偏移为每根轴生成一个独立的切换位置。测头信号在到达第一个轴切换位置时生成。

说明

测头信号

为测头 1 和测头 2 生成的测头信号总是相同。

负偏移值

位置偏移输入负值时，切换位置将推移到结束位置后。在此情形下不会生成测头信号。

示例

位置偏移设为 0.1 mm：MD13231 $MN_MEAS_PROBE_OFFSET = 0.1

示例 1：2 根轴中的通道专用测量

程序代码	注释
N10 G01 G90	
N20 MEAS=1 X100 Y10 F100	; 上升沿，测头 1
	; 切换位置[X] = 99.9
	; 切换位置[Y] = 9.9

示例 2：通过同步动作进行轴测量

程序代码	注释
N10 G01 G90	
N15 WHEN TRUE DO MEASA[X]=(1,1)	; 上升沿，测头 1
N20 X10 F100	; 切换位置[X] = 9.9

10.7.3 外部切换请求

功能

在以下 NC 专用机床数据中输入使用的数字量输出的编号（1...8），从而选择“外部切换请求”：

- MD13230 $MN_MEAS_PROBE_SOURCE = <数字量输出的编号>

测头信号通过控制配置过的数字量输出来触发。测量输入和数字量输出之间不需要硬件接线。

系统会通过置位数字量输出来为测头 1 和测头 2 生成切换信号上升沿。下降沿则通过数字量输出复位生成。

测量值为测量程序段中编写的切换信号（上升沿/ 下降沿）出现时的轴实际值。

数字量输出：设计

为了能将数字量输出用于仿真测量，必须设置以下机床数据：

- MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS = 1（生效的 NC 数字量输出字节的数量）
- MD13120 $MN_CONTROL_UNIT_LOGIC_ADDRESS = 0（SINAMICS-CU 逻辑地址）

数字量输出：设置

配置的数字量输出可在同步动作中置位：

```
WHEN <条件> DO $A_OUT[<数字量输出的编号>] = 1
```

示例

所使用的数字量输出：MD13230 $MN_MEAS_PROBE_SOURCE = 1

示例 1：2 根轴中的通道专用测量

程序代码	注释
N10 G01 G90 $A_OUT[1]=0	; 预占用数字量输出 1
N15 WHEN $AC_DETW<=10 DO $A_OUT[1]=1	; 轨迹剩余行程 <= 10 => 数字量输出 1 = 1
N20 MEAS=1 X100 Y10 F100	; 上升沿，测头 1

示例 2：轴测量

程序代码	注释
N10 G01 G90 $A_OUT[1]=0	; 预占用数字量输出 1
N15 WHEN $AA_IW[X]>=80 DO $A_OUT[1]=1	; 轴设定值 >= 80 => 数字量输出 1 = 1
N20 MEASA[X]=(1,1) X100 F100	; 上升沿，测头 1

10.7.4 系统变量

使用仿真测量时，下列系统变量的功能与实际测量时相同：

- $AC_MEA（测头已切换）
- $AA_MEAACT（轴测量生效）
- $AA_MM（获取的测头位置（MCS））
- $AA_MM1...4（测头位置，第 1 次 - 第 4 次触发（MCS））
- $AA_MW（获取的测头位置（WCS））
- $AA_MW1...4（测头位置，第 1 次触发（WCS））

以下系统变量提供的值无意义：

- $A_PROBE（测头状态）

急停 11

11.1 简要说明

功能

控制系统通过以下功能协助机床制造商实现急停：

- 所有 SINUMERIK 机床控制面板上均配备了一个急停键，操作人员可方便地触及。 急停键的功能包括强制打开电子开关触点，以及机械自动闭锁。
- 急停请求由 PLC 通过 NC/PLC 接口发送至 NC。
- 急停功能必须实现 0 类停机或 1 类停机（EN 60204）。
- 所有通过 PLC 控制的机床功能可在急停时进入安全状态，此状态可由机床制造商设置。
- 急停键的解锁不会使急停状态取消，也不会触发重启。
- 在取消急停状态后，不需要对机床轴执行回参考点或执行主轴同步。 系统会在急停期间持续跟踪机床轴的实际位置。

11.2 标准

相关标准

对于急停功能，须遵循下列标准：

- EN ISO 12000-1
- EN ISO 12000-2
- EN 418
- EN 60204

急停

根据 EN 418，急停功能的特点是：

- 用于防止人员遭受潜在或现存的危险，避免机械或生产材料受损。
- 当普通的停止功能不适用时，可由人员通过单次操作触发。

危险

根据 EN 418，危险可能来源于：

- 功能不规则（机械失灵、待加工材料的属性不当、人员操作失误等）。
- 一般运行。

EN ISO 12000-2 标准

根据欧盟机械指令中对急停的基本安全要求，机械必须配备急停装置。

特例

以下机械不需要急停装置：

- 急停装置无法降低风险，因为无法减少停止时间，或因所需采取的风险控制措施不适宜。
- 手持和手动控制的机械。

注意
机床制造商会接收到书面提示，敦促其遵循国家和国际标准。 SINUMERIK 控制系统能够协助机床制造商按照下文的功能说明实现急停功能。 但机床制造商须自行对急停功能（触发、运行以及应答）负责。

11.3 急停配置

急停控制元件

根据 EN 418，急停控制元件必须能够实现机械自动闭锁，且在紧急状况下能由操作人员和其他人员方便地操作。

例如可使用以下类型的控制元件：

- 蘑菇头开关（按键开关）
- 导线/线缆、索线、棒杆
- 手柄
- 特殊情形下：无保护帽的足控开关

急停和控制系统

有关 SINUMERIK 急停配置和参数设置的详细说明，请参见：

- Safety Integrated 调试手册
- SIMATIC Safety 功能手册

有关 Safety Integrated 的详细说明，请参见：

- Safety Integrated SINAMICS S120 功能手册

连接条件

急停键的连接请参见：
更多信息：
设备手册之操作组件

不同的 NC/PLC 接口信号与功能 12

12.1 简要说明

内容

PLC/NC 接口由数据接口和功能接口构成。数据接口包含有状态信号、控制信号、辅助指令和 G 指令等，PLC 至 NC 的任务传输则通过功能接口实现。

本说明介绍了一些常用的接口信号的功能，在特殊功能的说明中不会再介绍这些常用信号：

- 异步事件
- 状态信号
- PLC 变量（读取和写入）

12.2 功能

12.2.1 过载时的图像更新特性

在某些零件程序中，主处理须等待，直至预处理提供新程序段。

预处理和显示更新会争夺 NC 计算时间。

可通过以下机床数据设置预处理过慢时的 NC 特性：

MD10131 $MN_SUPPRESS_SCREEN_REFRESH

值	含义
0	当通道的预处理过慢时，抑制所有通道中的显示更新。
1	当通道的预处理过慢时，仅在时间关键通道中抑制抑制显示更新，以获取计算时间用于预处理。
2	基本不抑制显示更新。

12.2.2 激活缺省存储器

GUD 起始值

通过语言指令 DEF... / REDEF... 可以分配全局用户变量（GUD）缺省值。如果要使该缺省值在设置的初始化时间点可供使用，例如上电后通过 INIPO 属性，就必须将该缺省值永久保存在系统中。该缺省值的存储空间必须由以下机床数据进行使能：

MD18150 $MM_GUD_VALUES_MEM （用于 GUD 值的可记忆存储空间）

更多信息

- 存储器配置 (页 899)
- NC 编程手册

12.2.3 读写 PLC 变量

高速数据通道

在该模块 (DPR) 的通讯缓冲器中预留一个存储区以实现 PLC 和 NC 之间的高速信息交换。在该存储区中可以交换任意多个 PLC 变量（I/O、DB、DW、标记）。

PLC 通过'FunctionCalls' (FC) 访问该存储器而 NC 通过系统变量访问该存储器。

划分存储区

用户程序的编程人员（NC 和 PLC）自行负责存储区的划分。

此时可任意划分存储区，但是必须根据数据格式选择限制（"DWORD" 限制为 4 个字节，"WORD" 限制为 2 个字节，等等）。

存储区是通过指定数据类型和存储区内的位置偏移来访问的。

从 NC 访问

NC 系统中提供变量以便从零件程序或同步动作快速访问 PLC 变量。可通过 NC 直接读写数据。数据类型由系统变量的名称得出。存储区内的位置作为索引指定，单位为字节。

系统变量	数据类型	取值范围
$A_DBB（<索引>）	字节（8 位）	0 <= x <= 255
$A_DBW（<索引>）	字（16 位）	-32768 <= x <= 32767

系统变量	数据类型	取值范围
$A_DBD（<索引>）	双字（32 位）	-2147483648 <= x <= 2147483647
$A_DBR（<索引>）	浮点（32 位）	$\pm(1.5 \cdot 10^{-45} <= x <= 3.4 \cdot 10^{38})$

从 PLC 访问

PLC 借助"FunctionCall"(FC) 来访问存储器。在 FC 中，立即（而非在 PLC 循环开始时才）在 DPR 中读写数据。数据类型和存储区中的位置被作为参数传输至 FC。

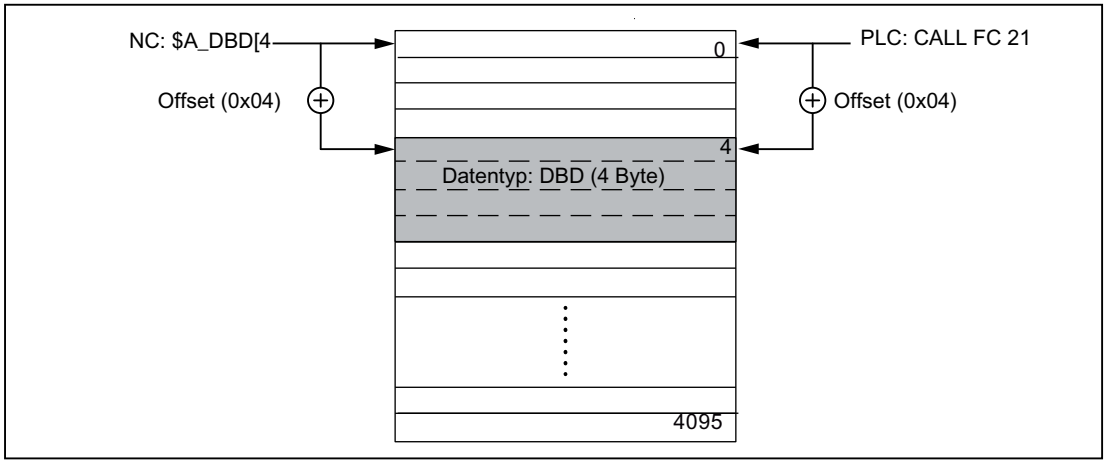


图 12-1　用于 NC/PLC 通讯的通信缓冲器 DPR

边界条件

- DPR 存储区的划分完全由用户负责。系统不会对配置的一致性进行检查。
- 在输入和输出方向上共有 4096 个字节可供使用。
- 系统不支持单个位指令，用户必须使用字节指令。
- 由于变量的内容是直接在通讯缓冲器中被修改的，用户程序的编程人员应注意：在多次分析一个变量或多个变量互联时，值可能被中途修改（即有必要将值暂时保存在局部变量或 R 参数中或安装信号装置）。
- 用户编程人员应负责协调不同通道对通讯缓冲器的访问。
- 在数据访问中，系统可保证 16 位以内（字节型和字型）数据的一致性。32 位数据（双字型和实数型）的一致性由用户自行负责。为此在 PLC 侧提供了一个简单的信号装置。

- PLC 将数据以 'Little Endian' 格式保存在 DPR 中。
- 通过 $A_DBR 传输的值需要经过转换，因此会损失一定的精度。浮点型数据在 NC 上为 DOUBLE（64 位），而在 PLC 上却只是 FLOAT（32 位）。DPR 中的保存格式为 FLOAT。转换在保存前后各进行一次。
  例如：如果是从 NC 写入然后再次读取 DPR 中的变量，则会进行两次转换。由于数据是以两种格式保存的，也就无法避免读写的值有偏差。
  示例
  通过比较"EPSILON" 来绕开该问题（细微偏差）

程序代码

```
N10          DEF REAL DBR
N12          DEF REAL EPSILON = 0.00001
N20          $A_DBR[0]=145.145
N30          G4 F2
N40          STOPRE
N50          DBR=$A_DBR[0]
N60          IF ( ABS(DBR/145.145-1.0) < EPSILON ) GOTOF ENDE
N70          MSG（“故障”）
N80          M0
N90          ENDE:
N99          M30
```

激活

可通过 MD28150 $MC_MM_NUM_VDIVAR_ELEMENTS（用于写入 PLC 变量的单元数）来设置
可同时写入的最大输出变量数

示例

一个单字型变量需从 PLC 传送至 NC。

NC 输入端内的位置偏移应为第四个字节（PLC 输出区）。位置偏移必须为数据宽度的整数倍。

由 PLC 写入：

程序代码	注释
. . .	
CALL FC21 (	
Enable :=M10.0,	; 若为 TRUE，则 FC21 生效
Funct :=B#16#4,	
S7Var :=P#M 104.0 WORD1,	
IVAR1 :=04,	

程序代码	注释
IVAR2 :=-1, Error :=M10.1, ErrCode :=MW12); . . .)	

在零件程序中读取

程序代码	注释
. . . PLCDATA = $A_DBW[4];	; 读取一个单字
. . .	

重新上电后的特性，程序段搜索

在上电时通讯缓冲器 DPR 初始化。

在进行“程序段搜索”时系统会收集输出的 PLC 变量，并通过移动程序段将其传送给通讯缓冲器 DPR（这一过程类似于写入模拟量和数字量输入）。

其他状态过渡在这儿无效。

更多信息

有关 PLC 通过 FC 21 进行数据交换的详细信息请参见：

功能手册之 PLC 和基本程序

12.2.4 通过口令和钥匙开关实现访问保护

访问权限

根据用户对功能、程序和数据的访问权限，系统划分了 7 个保护等级。它们分为：

- 多个口令等级，分别用于机床制造商和用户
- 多个钥匙开关位置，用于用户

多级安全方案

系统提供一种由多个口令级别以及钥匙开关位置组成的多级安全方案来控制访问权限。

保护等级	方式	用户	访问权限
1	口令	机床制造商： 开发人员	访问指定的的功能、程序和数据。 示例：输入选件
2	口令	机床制造商： 调试人员	访问指定的的功能、程序和数据。 示例：访问多个机床数据
3	口令	用户： 服务	访问指定的功能、程序和数据。
4	钥匙开关位置 3	用户： 编程人员、装配人员	比保护等级 0 - 3 的访问权限少。 由机床制造商或用户确定。
5	钥匙开关位置 2	用户： 有资质无需编程的操作员	比保护等级 0 - 3 的访问权限少。 由用户确定。
6	钥匙开关位置 1	用户： 经过培训、无需编程的操作员	示例： 只能选择程序、输入刀具磨损数据以及零点偏移
7	钥匙开关位置 0	用户： 刚刚入门的操作员	示例： 不可以输入和选择程序，只能操作机床控制面板

访问特性

- 保护等级 1 具有最高访问权限，保护等级 7 具有最低访问权限。
- 如果给某个保护等级设定了某个访问权限，则比它更高的保护等级会自动纳入该访问权限。
- 反之，如果需要修改某保护等级的访问权限，只能在比它更高的保护等级中进行修改。
- 保护等级 1 - 3 的访问权限由西门子设置（缺省设置）。
- 通过询问当前钥匙开关位置和比较输入的口令来设置访问权限。此时，输入的口令会覆盖钥匙开关位置的访问权限。
- 在每个保护等级中都可以保存选项数据。但是只有保护等级 1 才可以输入选件数据。
- 保护等级 4 - 7 的访问权限仅是推荐设置，可由机床制造商或者用户进行更改。

12.2.4.1 口令

设置口令

通过操作界面输入保护等级（1 - 3）的口令：

操作区“调试”>“口令”>“设置口令”

删除口令

通过输入设置的口令获得的访问权限一直保持生效，在明确删除口令后失效：

操作区“调试”>“口令”>“删除口令”

修改口令

通过操作界面更改保护等级（1 - 3）的口令：

操作区“调试”>“口令”>“修改口令”

说明

热启动

热启动不会影响访问权限或口令状态（已设置/已删除）！

缺省设置

缺省状态下，下列口令对保护等级 1 - 3 生效：

- 保护等级 1：SUNRISE
- 保护等级 2：EVENING
- 保护等级 3：CUSTOMER

注意
需要修改默认口令 保护等级 1-3 的默认口令稍后必须更改为您自己/自定义的口令以供操作使用。

更改口令

保护等级 1 - 3 的口令最迟应在调试完成时进行更改。

此时必须注意下列条件：

- 口令至少应由八个字符组成，且最多 32 个字符。
- 允许使用的字符有：
  - 大写字母
  - 小写字母
  - 数字
  - 特殊字符（ASCII 0x20 - 0x7E）
- 口令的字符组合中必须包含：
  - 至少一个数字
  - 至少一个大写字母
  - 至少一个小写字母
- 口令不得含有与以下类似的字符组合：
  - 用户名（制造商、服务、用户）
  - 计算机名称
  - 在用户的 Linux 口令文件中保存的附加信息（即 GECOS 数据）

设置口令时建议只使用操作面板上的字符。

注意
需要修改默认口令 保护等级 1-3 的默认口令稍后必须更改为您自己/自定义的口令以供操作使用。

注意
无法通过西门子重置口令 西门子无法将 SINUMERIK 上的密码重置为默认密码。因此，请密切注意您更改的口令。只能通过删除存储卡并重新安装软件版本 (restore -full) 来重置口令。通过更新 (restore -update) 无法将口令重置为默认口令。

注意
口令不包含在存档中 口令不包含在 SINUMERIK 存档中。因此在更换存储卡时，如必要应重新更改口令。

说明

设定安全密码

在设定新密码时请遵循以下规则：

- 在设定新密码时，请勿设定易于猜测的密码，例如简单的词、键盘上的易于猜测的按键序列等。
- 密码必须包含大小写字母的组合，以及数字和特殊字符。密码最少须包含 8 个字符。PINS 必须是无规律的数字序列。
- 在可行且 IT 系统和软件支持的情况下，在选择密码时必须采用尽可能复杂的字符。

有关安全密码设定的其他规则参见德国联邦信息安全办公室（BSI）。 (https://www.bsi.bund.de/EN/Themen/Verbraucherinnen-und-Verbraucher/Informationen-und-Empfehlungen/Cyber-Sicherheitsempfehlungen/Accountschutz/Sichere-Passwoerter-erstellen/sichere-passwoerter-erstellen_node.html)

在采用密码的情况下，可以使用用于密码管理的程序来加以协助。借助该程序能够对密码和口令进行加密保存、管理以及生成安全密码。

故障消息

由于在一些国家中对默认口令的修改存在强制性规定，因此如未将保护等级 1 - 3 的口令更改为自定义口令，则会重复（约每小时 1 次）输出以下报警：

报警 2130“提示：在访问等级“制造商、服务或用户”中，至少仍有一个尚在使用默认口令。”

该报警不会阻止 NC 启动。

Linux 密码

在 SINUMERIK 控制系统上同时使用 NC 口令和 Linux 密码。NC 上用于访问等级**制造商 (OEM)**、**服务**和**用户**的口令与 Linux 中用于对应 Linux 用户 **manufact**、**service** 和 **user** 的密码是一致的。这也就是说，通过 HMI 操作界面更改了 NC 口令后，Linux 用户的密码也随之更改。反之，Linux 密码更改也同时会导致 NC 口令变化。

两者的对应关系如下：

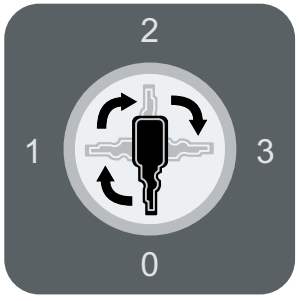
NC 访问等级	Linux 用户
制造商 (OEM)	manufact
服务	service
用户	user

在软件升级后，NC 会立即使用当前在 Linux 系统中保存的密码。交付时，Linux 密码为已知的默认口令。如在交付后更改了该密码，NC 在软件升级后会立即使用更改后的密码。

12.2.4.2 钥匙开关

钥匙开关位置与保护等级的对应关系

钥匙开关提供了**四个开关位置（0 到 3）**：



每个开关位置都分配了一个**指定的保护等级**：

开关位置	保护等级
0	7
1	6
2	5
3	4

钥匙开关有**三把不同颜色的钥匙**，钥匙可以在不同的开关位置插拔：

颜色	可插拔的开关位置
黑色	0 或 1
绿色	0、1 或 2
红色	0、1、2 或 3

根据钥匙开关的位置，可禁止或使能对 NC 中特定单元的访问：

- 钥匙开关位置 0 具有最低访问权限。
- 钥匙开关位置 3 具有最高访问权限。

接口信号

钥匙开关位置 1 到 3 的接口信号可直接通过机床控制面板的钥匙开关或 PLC 用户程序设定。

- <Nc>.basic.out.keyOperatedSwitchPos0（开关位置 0）
- <Nc>.basic.out.keyOperatedSwitchPos1（开关位置 1）

- <Nc>.basic.out.keyOperatedSwitchPos2（开关位置 2）
- <Nc>.basic.out.keyOperatedSwitchPos3（开关位置 3）

这其中只可设置一个位。若同时有多个位置位，那么系统内部会激活开关位置 3。

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<Nc>.basic.out.keyOperatedSwitchPos0	LBP_NC.A_Keyswitch0	DB10.DBX56.4
<Nc>.basic.out.keyOperatedSwitchPos1	LBP_NC.A_Keyswitch1	DB10.DBX56.5
<Nc>.basic.out.keyOperatedSwitchPos2	LBP_NC.A_Keyswitch2	DB10.DBX56.6
<Nc>.basic.out.keyOperatedSwitchPos3	LBP_NC.A_Keyswitch3	DB10.DBX56.7

12.2.4.3 可设置的保护等级

可自由设置不同的功能和数据区的保护等级。

可通过符合以下命名机制的操作面板机床数据来设置保护等级：

$MM_USER_CLASS_<*功能_数据区*>

示例

$MM_USER_CLASS_READ_TOA	读取刀具补偿
$MM_USER_CLASS_WRITE_TOA	写入刀具补偿
$MM_USER_CLASS_READ_PROGRAM	读取零件程序
$MM_USER_CLASS_WRITE_PROGRAM	写入/编辑零件程序

缺省值

除了少数个别情况，在出厂时或进行标准调试后，保护等级通常默认为 7（最低保护等级）。

12.2.5 切换电机数据组和驱动数据组

12.2.5.1 简介

电机数据组和驱动数据组

为了最佳地符合不同加工条件或不同机床配置的需求，一台驱动中有时需要具备多个不同的数据组用于电机、驱动参数和编码器。调试期间可借助“驱动向导”来设置驱动对象的基本数据组。

说明

更多信息

调试手册：CNC:NC，PLC - 驱动，调试 NC 控制的驱动

下列对数据组的修改和管理是通过操作界面实现的：

SINUMERIK Operate:操作区“调试”>“驱动系统”>“驱动”>“数据组”

具体加工条件中机床轴所需的电机数据组 (MDS) 或驱动数据组 (DDS)必须由 PLC 用户程序通过下文说明的接口来激活。

轴 NC/PLC 接口

用于切换电机数据组和驱动数据组的轴 NC/PLC 接口可分为三个区域：

- 格式接口 (页 764)
- 请求接口 (页 766)
- 显示接口 (页 767)

12.2.5.2 格式接口

格式

通过格式接口可以设置，请求接口和显示接口的哪些位用于电机数据组(MDS)的定址以及哪些位用于驱动数据组(DDS)的定址：

<轴>.drive.in.numberOfDriveDataSets，位 x = <值>

<值>	含义
0	电机数据组(MDS)的位的位置或无效位的位置
1	驱动数据组(DDS)的位的位置

驱动中的电机数据组和驱动数据组

格式取决于驱动中当前电机数据组(MDS)和驱动数据组(DDS)的数量。各个数量可通过以下驱动参数测定：

- p0130（电机数据组数量）
- p0180（驱动数据组数量）

接口有效性

一旦在控制系统启动时驱动传送了所有需要的信息，控制系统分析完这些信息，请求 (页 766)和显示接口 (页 767)就显示为有效：

<轴>.drive.in.numberOfDriveDataSetsValid == 1（请求接口和显示接口有效）

如果驱动没有传送任何的信息或是传送了不兼容的信息，那么请求和显示接口便显示为无效。

说明

当请求和显示接口无效时，用户/机床制造商若通过无效接口进行数据组切换，则所有责任由其自行承担。

原理

NC → PLC

Basic Program Plus	Basic Program	
<Axis>.drive.in.numberOfDriveDataSets	-	DB31, ...DBX130.0 ..4
<Axis>.drive.in.numberOfDriveDataSetsValid	-	DB31, ...DBX130.7

参见

示例 (页 767)

接口一览 (页 769)

12.2.5.3 请求接口

切换到新的电机数据组(MDS)和/或驱动数据组(DDS)的请求通过以下接口进行：

<轴>.drive.out.driveDataSetSelection = <DDS 下标>

<轴>.drive.out.motorDataSetSelection = <MDS 下标>

值域

电机数据组或驱动数据组的定址 n（n = 1, 2, 3, ...）根据其下标 i 进行，其中 i = n - 1 = 0, 1, 2, ...。

- 电机数据组：MDS[0, 1, 2, ... 15]
- 驱动数据组：DDS[0, 1, 2, ... 31]

接口格式

请求接口的格式，即哪些位用于电机数据组(MDS)的定址以及哪些位用于驱动数据组(DDS)的定址，通过格式接口 (页 764)进行设置。

驱动中的电机数据组和驱动数据组

驱动中现有的电机数据组(MDS)和驱动数据组(DDS)的数量可通过以下驱动参数测定：

- p0130（电机数据组数量）
- p0180（驱动数据组数量）

主主轴驱动的电机数据组(MDS)

主主轴驱动适用以下对应关系：

- MDS[0] → 星形接线方式
- MDS[1] → 三角形接线方式

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<Axis>.drive.out.driveDataSetSelection	LBP_typeAxisX.A_ParA..C; A_MotA..B	DB31, ...DBX21.0..2
<Axis>.drive.out.motorDataSetSelection	-	DB31, ...DBX21.3..4

12.2.5.4 显示接口

生效电机数据组(MDS)和驱动数据组(DDS)通过以下接口进行显示：

<轴>.drive.in.driveDataSetNumber == <DDS 下标>

<轴>.drive.in.motorDataSetNumber == <MDS 下标>

取值范围和格式与请求接口 (页 766)一致。

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<Axis>.drive.in.driveDataSetNumber	LBP_typeAxisX.E_ParA..C	DB31, ...DBX93.0..2
<Axis>.drive.in.motorDataSetNumber	LBP_typeAxisX.E_MotA..B	DB31, ...DBX93.3..4

12.2.5.5 示例

驱动中有两个电机数据组(MDS)，每个电机数据组有两个驱动数据组(DDS)。这与图 12-2 电机数据组/驱动数据组切换的原理 (页 770) 中所列出的编号为 9 的数据组组合一致。

格式

驱动数据组切换 (DDS) 的位位置：

- <轴>.drive.in.numberOfDriveDataSets == 1

电机数据组切换 (MDS) 的位位置：

- <轴>.drive.in.numberOfDriveDataSets.1 == 0

无效的位位置：

- <轴>.drive.in.numberOfDriveDataSets.2 == 0
- <轴>.drive.in.numberOfDriveDataSets.3 == 0
- <轴>.drive.in.numberOfDriveDataSets.4 == 0

驱动数据组接口 (DDS)

请求和显示接口相关的位位置：

- <轴>.drive.out.driveDataSetSelection.0 / <轴>.drive.in.driveDataSetNumber.0
  - <轴>.drive.out.driveDataSetSelection.0 / <轴>.drive.in.driveDataSetNumber.0 == 0 ⇒ 1.驱动数据组 DDS[0]
  - <轴>.drive.out.driveDataSetSelection.0 / <轴>.drive.out.motorDataSetSelection.0 == 1 ⇒ 2.驱动数据组 DDS[1]

电机数据组接口 (MDS)

请求和显示接口相关的位位置：

- <轴>.drive.out.driveDataSetSelection.1 / <轴>.drive.out.motorDataSetSelection.1
  - <轴>.drive.out.driveDataSetSelection.1 / <轴>.drive.in.driveDataSetNumber.1 == 0 ⇒ 1.电机数据组 MDS[0]
  - <轴>.drive.out.driveDataSetSelection.1 / <轴>.drive.in.driveDataSetNumber.1 == 1 ⇒ 2.电机数据组 MDS[1]

无效的位位置 (MDS / DDS)

请求和显示接口无效的位位置：

- <轴>.drive.out.driveDataSetSelection.1 / <轴>.drive.in.driveDataSetNumber.2 == 0
- <轴>.drive.out.driveDataSetSelection.1 / <轴>.drive.in.driveDataSetNumber.3 == 0
- <轴>.drive.out.driveDataSetSelection.1 / <轴>.drive.in.driveDataSetNumber.4 == 0

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<Axis>.drive.in.numberOfDriveDataSets	-	DB31, ...DBX130.0 ..4
<Axis>.drive.out.driveDataSetSelection	LBP_typeAxisX.A_ParA..C; A_MotA..B	DB31, ...DBX21.0.. 2
<Axis>.drive.out.motorDataSetSelection	-	DB31, ...DBX21.3.. 4
<Axis>.drive.in.driveDataSetNumber	LBP_typeAxisX.E_ParA..C	DB31, ...DBX93.0.. 2
<Axis>.drive.in.motorDataSetNumber	LBP_typeAxisX.E_MotA..B	DB31, ...DBX93.3.. 4

参见

接口一览 (页 769)

12.2.5.6 接口一览

表格 12-1 可配置的 MDS / DDS 组合

MDS（电机）数量	每个 MDS 对应的 DDS（驱动）数量
1	1 ... 32
2	1, 2, 4, 8, 16
3	1, 2, 4, 8
4	1, 2, 4, 8
5	1, 2, 4
6	1, 2, 4
7	1, 2, 4
8	1, 2, 4
9	1, 2
10	1, 2
11	1, 2
12	1, 2
13	1, 2
14	1, 2
15	1, 2
16	1, 2

每个 MDS 中的 DDS

<Axis>.drive.out.driveDataSetSelection
<Axis>.drive.out.motorDataSetSelection
<Axis>.drive.in.driveDataSetNumber
<Axis>.drive.in.motorDataSetNumber

<Axis>.drive.in.numberOfDriveDataSets

编号	MDS		4	3	2	1	0		4	3	2	1	0
1	1	1							0	0	0	0	0
2	2	1							0	0	0	0	0
3	3	1							0	0	0	0	0
4	4	1							0	0	0	0	0
5	8	1							0	0	0	0	0
6	16	1							0	0	0	0	0
7	32	1							0	0	0	0	0
8	1	2							0	0	0	0	1
9	2	2							0	0	0	0	1
10	3	2							0	0	0	0	1
11	4	2							0	0	0	0	1
12	8	2							0	0	0	0	1
13	16	2							0	0	0	0	1
14	1	4							0	0	0	1	1
15	2	4							0	0	0	1	1
16	3	4							0	0	0	1	1
17	4	4							0	0	0	1	1
18	8	4							0	0	0	1	1
19	1	8							0	0	1	1	1
20	2	8							0	0	1	1	1
21	3	8							0	0	1	1	1
22	4	8							0	0	1	1	1
23	1	16							0	1	1	1	1
24	2	16							0	1	1	1	1
25	1	32							1	1	1	1	1

不支持组合

重要的位:
电机数据组(MDS)
驱动数据组(DDS)
无效的位

MDS	电机数据组的数量
每个 MDS 中的 DDS	每个电机数据组中的驱动数据组的数量
<Axis>.drive.out.driveDataSetSelection	请求接口
<Axis>.drive.out.motorDataSetSelection	
<Axis>.drive.in.driveDataSetNumber	显示接口
<Axis>.drive.in.motorDataSetNumber	
<Axis>.drive.in.numberOfDriveDataSets	格式接口

图 12-2　电机数据组/驱动数据组切换的原理

PLC 信号

PLC → NC

Basic Program Plus	Basic Program	
<Axis>.drive.in.numberOfDriveDataSets	-	DB31, ...DBX130.0..4
<Axis>.drive.out.driveDataSetSelection	LBP_typeAxisX.A_ParA..C; A_MotA..B	DB31, ...DBX21.0..2
<Axis>.drive.out.motorDataSetSelection	-	DB31, ...DBX21.3..4
<Axis>.drive.in.driveDataSetNumber	LBP_typeAxisX.E_ParA..C	DB31, ...DBX93.0..2
<Axis>.drive.in.motorDataSetNumber	LBP_typeAxisX.E_MotA..B	DB31, ...DBX93.3..4

12.2.5.7 前提条件

“最后”一个电机数据组的驱动数据组的数量

“最后”一个电机数据组带有最高编号或下标。

通常，在驱动中会为每个电机数据组创建相同数量的驱动数据组（“每个 MDS 中的 DDS”数）。 但允许“最后”一个电机数据组有所不同，可以设置任意数量的驱动数据组：

1 ≤ a ≤（“每个 MDS 中的 DDS”数）

示例

现在要设置 4 个电机数据组、每个电机数据组有 8 个驱动数据组（每个 MDS 中的 DDS）。这与图 12-2 电机数据组/驱动数据组切换的原理 (页 770)中所列出的编号为 22 的数据组组合一致：

- 电机数据组：MDS[0], MDS[1], ... MDS[3]（“最后”一个电机数据组）
- 每个电机数据组中的驱动数据组：DDS[0] ... DDS[7]

因此，各个电机数据组中的驱动数据组的数量为：

电机数据组（MDS）	每个电机数据组(MDS)对应的驱动数据组(DDS)的数量
MDS[0] ... MDS[2]	8
MDS[3]	1 - 8

切换点：驱动参数组

原则上驱动参数组的切换可在任意时间执行。 在轴运行期间，特别是从转速控制器参数切换到电机转速定标时，可能会出现转矩跃变。 因此建议只在静止状态下切换驱动参数组，特别是轴处于静止时。

参见

接口一览 (页 769)

12.3 示例

12.3.1 参数组切换

参数组切换

通过参数组切换针对机床轴 X1 将位置闭环控制增益系数（K_V 系数）从 $K_V = 4.0$ 切换至 $K_V = 0.5$。

前提条件

必须通过以下机床数据来使能参数组切换：

MD35590 $MA_PARAMSET_CHANGE_ENABLE [AX1] = 1 或 2（可进行参数组切换）

根据下标 “0” 的机床数据，选择的是机床轴 X1 的第 1 个参数组。NC/PLC 接口：

<轴>.basic.out.posCtrlParameterSetSelection = 0（控制器参数组）

取决于参数组的机床数据

取决于参数组的机床数据是按照以下方式设置的：

机床数据	注释
MD32200 $MA_POSCTRL_GAIN [0, AX1] = 4.0	参数组 1 的 K_V 设置
MD32200 $MA_POSCTRL_GAIN [1, AX1] = 2.0	参数组 2 的 K_V 设置
MD32200 $MA_POSCTRL_GAIN [2, AX1] = 1.0	参数组 3 的 K_V 设置
MD32200 $MA_POSCTRL_GAIN [3, AX1] = 0.5	参数组 4 的 K_V 设置

机床数据	注释
MD32200 $MA_POSCTRL_GAIN [4, AX1] = 0.25	参数组 5 的 K_V 设置
MD32200 $MA_POSCTRL_GAIN [5, AX1] = 0.125	参数组 6 的 K_V 设置
MD31050 $MA_DRIVE_AX_RATIO_DENOM [0, AX1] = 3	参数组 1 的负载传动级分母
MD31050 $MA_DRIVE_AX_RATIO_DENOM [1, AX1] = 3	参数组 2 的负载传动级分母
MD31050 $MA_DRIVE_AX_RATIO_DENOM [2, AX1] = 3	参数组 3 的负载传动级分母
MD31050 $MA_DRIVE_AX_RATIO_DENOM [3, AX1] = 3	参数组 4 的负载传动级分母
MD31050 $MA_DRIVE_AX_RATIO_DENOM [4, AX1] = 3	参数组 5 的负载传动级分母
MD31050 $MA_DRIVE_AX_RATIO_DENOM [5, AX1] = 3	参数组 6 的负载传动级分母
MD31060 $MA_DRIVE_AX_RATIO_NUMERA [0, AX1] = 5	参数组 1 的负载传动级分子
MD31060 $MA_DRIVE_AX_RATIO_NUMERA [1, AX1] = 5	参数组 2 的负载传动级分子
MD31060 $MA_DRIVE_AX_RATIO_NUMERA [2, AX1] = 5	参数组 3 的负载传动级分子
MD31060 $MA_DRIVE_AX_RATIO_NUMERA [3, AX1] = 5	参数组 4 的负载传动级分子
MD31060 $MA_DRIVE_AX_RATIO_NUMERA [4, AX1] = 5	参数组 5 的负载传动级分子
MD31060 $MA_DRIVE_AX_RATIO_NUMERA [5, AX1] = 5	参数组 6 的负载传动级分子
MD35130 $MA_AX_VELO_LIMIT [0...5, AX1]	每个参数组的设置*)
MD32800 $MA_EQUIV_CURRCTRL_TIME [0..5, AX1]	每个参数组的设置*)
MD32810 $MA_EQUIV_SPEEDCTRL_TIME [0..5, AX1]	每个参数组的设置*)
MD32910 $MA_DYN_MATCH_TIME [0...5, AX1]	每个参数组的设置*)
*) 每个参数组所对应的行都需要根据句法规则单独指定。	

切换

PLC 用户程序选择机床轴 X1 的第 4 个参数组来切换位置闭环的增益系数。

- 通过 PLC 用户程序请求：
  <轴>.basic.out.posCtrlParameterSetSelection = 3（伺服参数组）
  - 向机床轴 AX1 发送请求，请求切换至第 4 个参数组。
  - 延迟时间之后会切换参数组。
  - 参数组 4 现已激活，与带有下标 “3” 的机床数据相一致。
- NC 发出反馈：
  <轴>.drive.in.posCtrlParameterSetNumber = 3（伺服参数组）
  - 参数组的切换由 NC 来确认/应答。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Axis>.drive.in.posCtrlParameterSetNumber	LBP_Axis*.E_ParS_A	DB31, ...DBX69.0..2

PLC → NC

Basic Program Plus	Basic Program	
<Axis>.basic.out.posCtrlParameterSetSelection	LBP_Axis*.A_Para_A..C	DB31, ...DBX9.0..2

PLC 辅助功能输出 13

13.1 简要说明

13.1.1 功能

辅助功能可用于激活 NC 的系统功能和 PLC 用户功能。可在以下程序或动作中编程辅助功能：

- 零件程序
- 同步动作
- 用户循环

有关在同步动作中使用辅助功能输出的详细信息请参考：

更多信息
功能手册之同步动作

预定义的辅助功能

预定义的辅助功能可激活系统功能。此外， 辅助功能还会输出到 NC/PLC 接口上。

预定义的辅助功能有：

类型	功能	示例	含义
M	附加功能	M30	程序结束
S	主轴功能	S100	主轴转速 100（如 rpm）
T	刀具号	T3	刀具号 3
D, DL	刀具补偿	D1	刀沿号 1
F	进给率	F1000	进给率 1000（如毫米/分）

用户自定义辅助功能

用户定义的辅助功能要么是经过扩展的预定义辅助功能，要么是用户专用的辅助功能。

预定义辅助功能的扩展

预定义辅助功能的扩展基于“地址扩展”参数。通过地址扩展定义辅助功能涉及的主轴的编号。例如，针对主主轴预定义了主轴功能 M3（主轴正转）。若为通道分配了第 2 根主轴，则需要定义相应的用户自定义辅助功能，以作为对预定义辅助功能的扩展。

类型	功能	示例	含义
M	附加功能	M2=3	第 2 根主轴：主轴正转
S	主轴功能	S2=100	第 2 根主轴：主轴转速 = 100（如 rpm）
T	刀具号	T2=3	

用户专用辅助功能

用户专用辅助功能不会激活系统功能。用户专用辅助功能只会被输出至 NC/PLC 接口。辅助功能的功能性须由机床制造商/用户在 PLC 用户程序中实现。

类型	功能	示例	含义
H[1)]	辅助功能	H2=5	用户专用的功能

1) 推荐

13.1.2 辅助功能的定义

辅助功能是通过以下参数来定义的：

- **类型、地址扩展和值**
  这 3 个参数会输出到 NC/PLC 接口上。
- **输出特性**
  通过辅助功能特定的输出特性可以确定：辅助功能输出到 NC/PLC 接口上所需的时间以及相对于零件程序中编程的运动何时进行输出。
- **分组指定**
  辅助功能可划分到指定的辅助功能组中。每个辅助功能组都可定义自己的输出特性。如果辅助功能没有特定的输出特性，所属功能组的输出特性便生效。此外，分组也会影响辅助功能在程序段搜索时的输出。

更多信息

功能手册之 PLC 和基本程序，章节“基本程序”

13.1.3 辅助功能一览

M 功能

M（附加功能）					
地址扩展		**值**			
取值范围	**含义**	**取值范围** [10]	**类型**	**含义**	**数量** [8]
0（固有的）	---	0 ... 99	INT	功能	5
取值范围	**含义**	**取值范围** [11]	**类型**	**含义**	**数量** [8]
1 ... 20	主轴号	1 ... 99	INT	功能	5
取值范围	**含义**	**取值范围** [12]	**类型**	**含义**	**数量** [8]
0 ... 99	任意	100 ... 2147483647	INT	功能	5

[8] 参见概览末尾处的“脚注含义”

[10] 对于 0 至 99 的取值范围地址扩展为 0。强制无地址扩展：M0，M1，M2，M17，M30

[11] M3，M4，M5，M19，M70：地址扩展为主轴编号，例如 M2=5 ⇒ 主轴 2 的主轴停止（M5）
在无地址扩展的情况下，该 M 功能作用于主主轴。

[12] 用户专用的 M 功能。

使用

同步控制机床功能和零件程序。

更多信息

- 下列 M 功能具有预定义的含义：M0，M1，M2，M17，M30，M3，M4，M5，M6，M19，M70，M40，M41，M42，M43，M44，M45。
- 针对 M 功能（M0 - M99）设有对应的用于显示有效性的动态 NC/PLC 接口信号。此外，还可为自有 M 功能分配 64 个附加信号（参见功能手册之 *PLC 和基本程序*，章节“*基本程序*”）。
- 对于子程序，可通过以下机床数据来设置是否应将用于零件程序结束的 M 功能 M17、M2 或 M30 输出到 PLC 上：
  MD20800 $MC_SPF_END_TO_VDI（子程序结束输出给 PLC）
- 对于预定的 M 功能 M40 ... M45，对输出特性的重新定义受到限制。

- 预定义的辅助功能 M0、M1、M17、M30、M6、M4、M5 无法进行重新定义。
- M 功能专用的机床数据：
  - MD10800 $MN_EXTERN_CHAN_SYNC_M_NO_MIN
  - MD10802 $MN_EXTERN_CHAN_SYNC_M_NO_MAX
  - MD10804 $MN_EXTERN_M_NO_SET_INT
  - MD10806 $MN_EXTERN_M_NO_DISABLE_INT
  - MD10814 $MN_EXTERN_M_NO_MAC_CYCLE
  - MD10815 $MN_EXTERN_M_NO_MAC_CYCLE_NAME
  - MD20094 $MC_SPIND_RIGID_TAPPING_M_NR
  - MD20095 $MC_EXTERN_RIGID_TAPPING_M_NR
  - MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO
  - MD22200 $MC_AUXFU_M_SYNC_TYPE
  - MD22530 $MC_TOCARR_CHANGE_M_CODE
  - MD22532 $MC_GEOAX_CHANGE_M_CODE
  - MD22534 $MC_TRAFO_CHANGE_M_CODE
  - MD22560 $MC_TOOL_CHANGE_M_CODE

S 功能

S（主轴功能）					
地址扩展 [10]		**值**			
取值范围	**含义**	**取值范围**	**类型**	**含义**	**数量** [8]
0 ... 20	主轴编号 [5]	0 ... ± 3.4028 exp38 [3]	REAL	主轴转速	3

[3] [5] [8] 参见概览末尾处的“脚注含义”。

[10] 如果没有指定地址扩展，系统会对通道的主主轴进行编址。

使用

主轴转速。

更多信息

- 缺省设置中，S 功能分配给第 3 个辅助功能组。
- 如果没有地址扩展，S 功能用于通道的主主轴。
- S 功能专用的机床数据：
  MD22210 $MC_AUXFU_S_SYNC_TYPE（S 功能的输出时间点）

H 功能

H 功能需要在 PLC 用户程序中实现。

H（辅助功能）[10]					
地址扩展		值			
取值范围	含义	取值范围	类型	含义	数量 [8]
0 ... 99	任意	- 2147483648 ... + 2147483647	INT	任意	3
		0 ... ± 3.4028 exp38 [2] [3] [4]	REAL		

[2] [3] [4] [8] 参见概览末尾处的“脚注含义”。

使用

用户专用的辅助功能。

更多信息

H 功能专用的机床数据：

- MD22110 $MC_AUXFU_H_TYPE_INT（H 辅助功能的类型为整数）
- MD22230 $MC_AUXFU_H_SYNC_TYPE（H 功能的输出时间点）

T 功能

刀具名称不输出至 NC/PLC 接口。

T（刀具编号）[1] [5] [6]					
地址扩展		值			
取值范围	含义	取值范围	类型	含义	数量 [8]
1 ... 12	主轴号（在刀具管理生效时）	0 ... 32000（在刀具管理生效时也可以为符号刀具名称）	INT	选择刀具	1

[1] [5] [6] [8] 参见概览末尾处的“脚注含义”。

使用

刀具选择。

更多信息

- 选择通过刀具编号或刀位编号来识别刀具（参见功能手册之*刀具*，章节“*刀具补偿*”）。
  功能手册之刀具管理
- 用 T0 从刀架中取下刀具，但不换上新的刀具（初始设置）。
- T 功能专用的机床数据：
  MD22220 $MC_AUXFU_T_SYNC_TYPE（T 功能的输出时间点）

D 功能

用 D0 取消刀具补偿。预赋值为 D1。

D（刀具补偿）					
地址扩展		**值**			
取值范围	**含义**	**取值范围**	**类型**	**含义**	**数量** [8]
---	---	0 ... 9	INT	选择刀具补偿	1

[8] 参见概览末尾处的“脚注含义”。

使用

选择刀具补偿。

更多信息

- 初始设置：D1
- 可通过以下机床数据对换刀后的缺省刀沿进行设置：
  MD20270 $MC_CUTTING_EDGE_DEFAULT （未编程的刀沿初始设置）
- 取消刀具补偿：D0
- D 功能专用的机床数据：
  MD22250 $MC_AUXFU_D_SYNC_TYPE（D 功能的输出时间点）

DL 功能

通过 DL 选中的刀具总补偿针对的是生效的 D 号。

DL（刀具总补偿）					
地址扩展		值			
取值范围	含义	取值范围	类型	含义	数量 [8]
---	---	0 ... 6	INT	选择刀具总补偿	1

[8] 参见概览末尾处的“脚注含义”。

使用

选择与生效的刀具补偿相关的刀具总补偿。

更多信息

- 初始设置：DL = 0
- 无法通过同步动作将 DL 值输出到 PLC 上。
- DL 功能未生效时的刀具总补偿的初始设置：
  MD20272 $MC_SUMCORR_DEFAULT （未编程总补偿初始设置）
- 取消刀具总补偿：DL = 0
- DL 功能专用的机床数据：
  MD22252 $MC_AUXFU_DL_SYNC_TYPE（DL 功能的输出时间）

F 功能

F（轨迹进给率）					
地址扩展		值			
取值范围	含义	取值范围	类型	含义	数量 [8]
---	---	0.001 ... 999 999.999	REAL	轨迹进给率	6

[8] 参见概览末尾处的“脚注含义”。

使用

轨迹速度。

更多信息

F 功能专用的机床数据：

- MD22240 $MC_AUXFU_F_SYNC_TYP（F 功能的输出时间）

FA 功能

FA（轴专用进给率）					
地址扩展		值			
取值范围	含义	取值范围	类型	含义	数量 [8]
1 - 31	轴编号	0.001 ... 999 999.999	REAL	轴进给	6

[8] 参见概览末尾处的“脚注含义”。

使用

轴专用速度

更多信息

F 功能专用的机床数据：

- MD22240 $MC_AUXFU_F_SYNC_TYP（F 功能的输出时间）

脚注的含义

[1] 当刀具管理激活时， 系统既不会将 T 修改信号，也不会将 T 字发送至通道专用 NC/PLC 接口。

[2] 用户可通过机床数据 MD22110 $MC_AUXFU_H_TYPE_INT 选择值的类型。

[3] 由于操作面板屏幕上的显示方式受限，REAL 型的值被限制在以下范围内：

–999 999 999.9999 至 999 999 999.9999

NC 会在内部使用完整的精度进行计算。

[4] 在设置以下机床数据时，NC 会对 REAL 值进行取整，然后将其输出到 PLC 上：

MD22110 $MC_AUXFU_H_TYPE_INT = 1（H 辅助功能的类型为整数）

PLC 用户程序须根据机床数据的设置来编译传递值。

[5] 当刀具管理激活时，可对地址扩展的含义进行参数设置。地址扩展 = 0 表示该值须由主主轴号替换，即与没有对地址扩展进行编程时的含义相同。

在进行程序段搜索时，收集的辅助功能 M19 “定位主轴”不会输出到 PLC 上。

[6] M6：地址扩展的取值范围：

- 无刀具管理：0 ... 99

- 带刀具管理：0 ... 最大主轴编号

0：由主主轴号或主刀架的值替换

7) 当刀具管理激活时，无论是否编程了地址扩展，辅助功能 M6“换刀”在零件程序段中只能编程一次。

8) 每个零件程序段的最大辅助功能数。

13.2 预定义的辅助功能

功能

每个预定义的辅助功能都会分配一个系统功能且不可更改。如果已经在一个零件程序/循环中编程了一个预定义的辅助功能，该功能会通过 NC/PLC 接口输出到 PLC 上并在 NC 中执行相应的系统功能。

预定义辅助功能的定义

预定于辅助功能的参数在机床数据中定义，且可对其进行部分修改。所有属于一个辅助功能的机床数据都具有相同的下标 <n>。

- MD22040 $MC_AUXFU_PREDEF_GROUP[]（预定义辅助功能的分组指定）
- MD22050 $MC_AUXFU_PREDEF_TYPE[<n>]（预定义辅助功能的类型）
- MD22060 $MC_AUXFU_PREDEF_EXTENSION[<n>]（预定义辅助功能的地址扩展）
- MD22070 $MC_AUXFU_PREDEF_VALUE[<n>]（预定义辅助功能的数值）
- MD22080 $MC_AUXFU_PREDEF_SPEC[<n>]（预定义辅助功能的输出特性）

13.2.1 一览：预定义的辅助功能

下表中所列参数的含义：

参数	含义
下标 <n>	辅助功能参数的机床数据下标
类型	MD22050 $MC_AUXFU_PREDEF_TYPE[<n>]
地址扩展	MD22060 $MC_AUXFU_PREDEF_EXTENSION[<n>]
值	MD22070 $MC_AUXFU_PREDEF_VALUE[<n>]
组	MD22040 $MC_AUXFU_PREDEF_GROUP[<n>]

预定义的辅助功能

通用辅助功能：第 1 部分					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
停止	0	M	0	0	1
有条件停止	1	M	0	1	1
子程序结束	2	M	0	2	1
	3	M	0	17	1
	4	M	0	30	1
换刀	5	M	(0)	6 [1)]	(1)

主轴专用的辅助功能，主轴 1					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
主轴正转	6	M	1	3	(2)
主轴反转	7	M	1	4	(2)
主轴停止	8	M	1	5	(2)
主轴定位	9	M	1	19	(2)
进给轴模式	10	M	1	70 [2)]	(2)
自动换档	11	M	1	40	(4)
齿轮档 1	12	M	1	41	(4)
齿轮档 2	13	M	1	42	(4)
齿轮档 3	14	M	1	43	(4)
齿轮档 4	15	M	1	44	(4)
齿轮档 5	16	M	1	45	(4)
主轴转速	17	S	1	-1	(3)

通用辅助功能：第 2 部分					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
进给率	18	F	0	-1	(1)
刀沿选择	19	D	0	-1	(1)
DL	20	L	0	-1	(1)

通用辅助功能：第 2 部分					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
刀具选择	21	T	(0)	-1	(1)
停止（组合式）	22	M	0	-1 [3)]	1
有条件停止（组合式）	23	M	0	-1 [4)]	1
子程序结束	24	M	0	-1 [5)]	1
步冲	25	M	0	20 [6)]	(10)
步冲	26	M	0	23 [6)]	(10)
步冲	27	M	0	22 [6)]	(11)
步冲	28	M	0	25 [6)]	(11)
步冲	29	M	0	26 [6)]	(12)
步冲	30	M	0	122 [6)]	(11)
步冲	31	M	0	125 [6)]	(11)
步冲	32	M	0	27 [6)]	(12)

主轴专用的辅助功能，主轴 2					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
主轴正转	33	M	2	3	(72)
主轴反转	34	M	2	4	(72)
主轴停止	35	M	2	5	(72)
主轴定位	36	M	2	19	(72)
进给轴模式	37	M	2	70 [2)]	(72)
自动换档	38	M	2	40	(74)
齿轮档 1	39	M	2	41	(74)
齿轮档 2	40	M	2	42	(74)
齿轮档 3	41	M	2	43	(74)
齿轮档 4	42	M	2	44	(74)
齿轮档 5	43	M	2	45	(74)
主轴转速	44	S	2	-1	(73)

主轴专用的辅助功能，主轴 3					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
主轴正转	45	M	3	3	(75)
主轴反转	46	M	3	4	(75)
主轴停止	47	M	3	5	(75)
主轴定位	48	M	3	19	(75)
进给轴模式	49	M	3	70 [2)]	(75)
自动换档	50	M	3	40	(77)
齿轮档 1	51	M	3	41	(77)
齿轮档 2	52	M	3	42	(77)
齿轮档 3	53	M	3	43	(77)
齿轮档 4	54	M	3	44	(77)
齿轮档 5	55	M	3	45	(77)
主轴转速	56	S	3	-1	(76)

主轴专用的辅助功能，主轴 4					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
主轴正转	57	M	4	3	(78)
主轴反转	58	M	4	4	(78)
主轴停止	59	M	4	5	(78)
主轴定位	60	M	4	19	(78)
进给轴模式	61	M	4	70 [2)]	(78)
自动换档	62	M	4	40	(80)
齿轮档 1	63	M	4	41	(80)
齿轮档 2	64	M	4	42	(80)
齿轮档 3	65	M	4	43	(80)
齿轮档 4	66	M	4	44	(80)
齿轮档 5	67	M	4	45	(80)
主轴转速	68	S	4	-1	(79)

主轴专用的辅助功能，主轴 5					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
主轴正转	69	M	5	3	(81)
主轴反转	70	M	5	4	(81)
主轴停止	71	M	5	5	(81)
主轴定位	72	M	5	19	(81)
进给轴模式	73	M	5	70 [2)]	(81)
自动换档	74	M	5	40	(83)
齿轮档 1	75	M	5	41	(83)
齿轮档 2	76	M	5	42	(83)
齿轮档 3	77	M	5	43	(83)
齿轮档 4	78	M	5	44	(83)
齿轮档 5	79	M	5	45	(83)
主轴转速	80	S	5	-1	(82)

主轴专用的辅助功能，主轴 6					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
主轴正转	81	M	6	3	(84)
主轴反转	82	M	6	4	(84)
主轴停止	83	M	6	5	(84)
主轴定位	84	M	6	19	(84)
进给轴模式	85	M	6	70 [2)]	(84)
自动换档	86	M	6	40	(86)
齿轮档 1	87	M	6	41	(86)
齿轮档 2	88	M	6	42	(86)
齿轮档 3	89	M	6	43	(86)
齿轮档 4	90	M	6	44	(86)
齿轮档 5	91	M	6	45	(86)
主轴转速	92	S	6	-1	(85)

主轴专用的辅助功能，主轴 7					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
主轴正转	93	M	7	3	(87)
主轴反转	94	M	7	4	(87)
主轴停止	95	M	7	5	(87)
主轴定位	96	M	7	19	(87)
进给轴模式	97	M	7	70 [2)]	(87)
自动换档	98	M	7	40	(89)
齿轮档 1	99	M	7	41	(89)
齿轮档 2	100	M	7	42	(89)
齿轮档 3	101	M	7	43	(89)
齿轮档 4	102	M	7	44	(89)
齿轮档 5	103	M	7	45	(89)
主轴转速	104	S	7	-1	(88)

主轴专用的辅助功能，主轴 8					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
主轴正转	105	M	8	3	(90)
主轴反转	106	M	8	4	(90)
主轴停止	107	M	8	5	(90)
主轴定位	108	M	8	19	(90)
进给轴模式	109	M	8	70 [2)]	(90)
自动换档	110	M	8	40	(92)
齿轮档 1	111	M	8	41	(92)
齿轮档 2	112	M	8	42	(92)
齿轮档 3	113	M	8	43	(92)
齿轮档 4	114	M	8	44	(92)
齿轮档 5	115	M	8	45	(92)
主轴转速	116	S	8	-1	(91)

主轴专用的辅助功能，主轴 9					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
主轴正转	117	M	9	3	(93)
主轴反转	118	M	9	4	(93)
主轴停止	119	M	9	5	(93)
主轴定位	120	M	9	19	(93)
进给轴模式	121	M	9	70 [2)]	(93)
自动换档	122	M	9	40	(95)
齿轮档 1	123	M	9	41	(95)
齿轮档 2	124	M	9	42	(95)
齿轮档 3	125	M	9	43	(95)
齿轮档 4	126	M	9	44	(95)
齿轮档 5	127	M	9	45	(95)
主轴转速	128	S	9	-1	(94)

主轴专用的辅助功能，主轴 10					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
主轴正转	129	M	10	3	(96)
主轴反转	130	M	10	4	(96)
主轴停止	131	M	10	5	(96)
主轴定位	132	M	10	19	(96)
进给轴模式	133	M	10	70 [2)]	(96)
自动换档	134	M	10	40	(98)
齿轮档 1	135	M	10	41	(98)
齿轮档 2	136	M	10	42	(98)
齿轮档 3	137	M	10	43	(98)
齿轮档 4	138	M	10	44	(98)
齿轮档 5	139	M	10	45	(98)
主轴转速	140	S	10	-1	(97)

主轴专用的辅助功能，主轴 11					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
主轴正转	141	M	11	3	(99)
主轴反转	142	M	11	4	(99)
主轴停止	143	M	11	5	(99)
主轴定位	144	M	11	19	(99)
进给轴模式	145	M	11	70 [2)]	(99)
自动换档	146	M	11	40	(101)
齿轮档 1	147	M	11	41	(101)
齿轮档 2	148	M	11	42	(101)
齿轮档 3	149	M	11	43	(101)
齿轮档 4	150	M	11	44	(101)
齿轮档 5	151	M	11	45	(101)
主轴转速	152	S	11	-1	(100)

主轴专用的辅助功能，主轴 12					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
主轴正转	153	M	11	3	(102)
主轴反转	154	M	12	4	(102)
主轴停止	155	M	12	5	(102)
主轴定位	156	M	12	19	(102)
进给轴模式	157	M	12	70 [2)]	(102)
自动换档	158	M	12	40	(104)
齿轮档 1	159	M	12	41	(104)
齿轮档 2	160	M	12	42	(104)
齿轮档 3	161	M	12	43	(104)
齿轮档 4	162	M	12	44	(104)
齿轮档 5	163	M	12	45	(104)
主轴转速	164	S	12	-1	(103)

主轴专用的辅助功能，主轴 13					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
主轴正转	165	M	13	3	(105)
主轴反转	166	M	13	4	(105)
主轴停止	167	M	13	5	(105)
主轴定位	168	M	13	19	(105)
进给轴模式	169	M	13	70 [2)]	(105)
自动换档	170	M	13	40	(107)
齿轮档 1	171	M	13	41	(107)
齿轮档 2	172	M	13	42	(107)
齿轮档 3	173	M	13	43	(107)
齿轮档 4	174	M	13	44	(107)
齿轮档 5	175	M	13	45	(107)
主轴转速	176	S	13	-1	(106)

主轴专用的辅助功能，主轴 14					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
主轴正转	177	M	14	3	(108)
主轴反转	178	M	14	4	(108)
主轴停止	179	M	14	5	(108)
主轴定位	180	M	14	19	(108)
进给轴模式	181	M	14	70 [2)]	(108)
自动换档	182	M	14	40	(110)
齿轮档 1	183	M	14	41	(110)
齿轮档 2	184	M	14	42	(110)
齿轮档 3	185	M	14	43	(110)
齿轮档 4	186	M	14	44	(110)
齿轮档 5	187	M	14	45	(110)
主轴转速	188	S	14	-1	(109)

主轴专用的辅助功能，主轴 15					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
主轴正转	189	M	15	3	(111)
主轴反转	190	M	15	4	(111)
主轴停止	191	M	15	5	(111)
主轴定位	192	M	15	19	(111)
进给轴模式	193	M	15	70 [2)]	(111)
自动换档	194	M	15	40	(113)
齿轮档 1	195	M	15	41	(113)
齿轮档 2	196	M	15	42	(113)
齿轮档 3	197	M	15	43	(113)
齿轮档 4	198	M	15	44	(113)
齿轮档 5	199	M	15	45	(113)
主轴转速	200	S	15	-1	(112)

主轴专用的辅助功能，主轴 16					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
主轴正转	201	M	16	3	(114)
主轴反转	202	M	16	4	(114)
主轴停止	203	M	16	5	(114)
主轴定位	204	M	16	19	(114)
进给轴模式	205	M	16	70 [2)]	(114)
自动换档	206	M	16	40	(116)
齿轮档 1	207	M	16	41	(116)
齿轮档 2	208	M	16	42	(116)
齿轮档 3	209	M	16	43	(116)
齿轮档 4	210	M	16	44	(116)
齿轮档 5	211	M	16	45	(116)
主轴转速	212	S	16	-1	(115)

主轴专用的辅助功能，主轴 17					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
主轴正转	213	M	17	3	(117)
主轴反转	214	M	17	4	(117)
主轴停止	215	M	17	5	(117)
主轴定位	216	M	17	19	(117)
进给轴模式	217	M	17	70 [2)]	(117)
自动换档	218	M	17	40	(119)
齿轮档 1	219	M	17	41	(119)
齿轮档 2	220	M	17	42	(119)
齿轮档 3	221	M	17	43	(119)
齿轮档 4	222	M	17	44	(119)
齿轮档 5	223	M	17	45	(119)
主轴转速	224	S	17	-1	(118)

主轴专用的辅助功能，主轴 18					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
主轴正转	225	M	18	3	(120)
主轴反转	226	M	18	4	(120)
主轴停止	227	M	18	5	(120)
主轴定位	228	M	18	19	(120)
进给轴模式	229	M	18	70 [2)]	(120)
自动换档	230	M	18	40	(122)
齿轮档 1	231	M	18	41	(122)
齿轮档 2	232	M	18	42	(122)
齿轮档 3	233	M	18	43	(122)
齿轮档 4	234	M	18	44	(122)
齿轮档 5	235	M	18	45	(122)
主轴转速	236	S	18	-1	(121)

主轴专用的辅助功能，主轴 19					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
主轴正转	237	M	19	3	(123)
主轴反转	238	M	19	4	(123)
主轴停止	239	M	19	5	(123)
主轴定位	240	M	19	19	(123)
进给轴模式	241	M	19	70 [2)]	(123)
自动换档	242	M	19	40	(125)
齿轮档 1	243	M	19	41	(125)
齿轮档 2	244	M	19	42	(125)
齿轮档 3	245	M	19	43	(125)
齿轮档 4	246	M	19	44	(125)
齿轮档 5	247	M	19	45	(125)
主轴转速	248	S	19	-1	(124)

主轴专用的辅助功能，主轴 20					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
主轴正转	249	M	20	3	(126)
主轴反转	250	M	20	4	(126)
主轴停止	251	M	20	5	(126)
主轴定位	252	M	20	19	(126)
进给轴模式	253	M	20	70 [2)]	(126)
自动换档	254	M	20	40	(128)
齿轮档 1	255	M	20	41	(128)
齿轮档 2	256	M	20	42	(128)
齿轮档 3	257	M	20	43	(128)
齿轮档 4	258	M	20	44	(128)
齿轮档 5	259	M	20	45	(128)
主轴转速	260	S	20	-1	(127)

刀架专用的辅助功能、T 辅助功能					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
刀具选择	261	T	1	-1	129
刀具选择	262	T	2	-1	130
刀具选择	263	T	3	-1	131
刀具选择	264	T	4	-1	132
刀具选择	265	T	5	-1	133
刀具选择	266	T	6	-1	134
刀具选择	267	T	7	-1	135
刀具选择	268	T	8	-1	136
刀具选择	269	T	9	-1	137
刀具选择	270	T	10	-1	138
刀具选择	271	T	11	-1	139
刀具选择	272	T	12	-1	140
刀具选择	273	T	13	-1	141
刀具选择	274	T	14	-1	142
刀具选择	275	T	15	-1	143
刀具选择	276	T	16	-1	144
刀具选择	277	T	17	-1	145
刀具选择	278	T	18	-1	146
刀具选择	279	T	19	-1	147
刀具选择	280	T	20	-1	148

刀架专用的辅助功能、M6 辅助功能					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
换刀	281	M	1	6 [1)]	149
换刀	282	M	2	6 [1)]	150
换刀	283	M	3	6 [1)]	151
换刀	284	M	4	6 [1)]	152
换刀	285	M	5	6 [1)]	153

刀架专用的辅助功能、M6 辅助功能					
系统功能	**下标 <n>**	**类型**	**地址扩展**	**值**	**组**
换刀	286	M	6	6 1)	154
换刀	287	M	7	6 1)	155
换刀	288	M	8	6 1)	156
换刀	289	M	9	6 1)	157
换刀	290	M	10	6 1)	158
换刀	291	M	11	6 1)	159
换刀	292	M	12	6 1)	160
换刀	293	M	13	6 1)	161
换刀	294	M	14	6 1)	162
换刀	295	M	15	6 1)	163
换刀	296	M	16	6 1)	164
换刀	297	M	17	6 1)	165
换刀	298	M	18	6 1)	166
换刀	299	M	19	6 1)	167
换刀	300	M	20	6 1)	168

图例：

( ) 可对该值进行修改。

1) 该值取决于机床数据：

MD22560 $MC_TOOL_CHANGE_M_MODE（用于换刀的 M 功能）

2) 可通过以下机床数据为该值预设另一个值：

MD20095 $MC_EXTERN_RIGID_TAPPING_M_NR（用于切换到开环控制轴运行的 M 功能（外部模式））

MD20094 $MC_SPIND_RIGID_TAPPING_M_NR（用于切换到开环控制轴运行的 M 功能）

提示

输出到 PLC 上的值始终为值 70。

3) 该值是通过以下机床数据设置的：

MD22254 $MC_AUXFU_ASSOC_M0_VALUE（用于程序停止的附加 M 功能）

4) 该值是通过以下机床数据设置的：
MD22256 $MC_AUXFU_ASSOC_M1_VALUE（用于有条件停止的附加 M 功能）

5) 该值是通过以下机床数据设置的：
MD10714 $MN_M_NO_FCT_EOP（复位后，主轴的 M 功能生效）

6) 该值是通过以下机床数据设置的：
MD26008 $MC_NIBBLE_PUNCH_CODE（确定 M 功能）

13.2.2 一览：输出特性

下表中列出的参数的含义：

参数	含义
下标 <n>	辅助功能参数的机床数据下标
输出特性	MD22080 $MC_AUXFU_PREDEF_SPEC[<n>]，位 0 ... 18 位 19 ... 31：预留

预定义辅助功能的输出特性

系统功能	下标 <n>																			
		输出特性，位																		
18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
停止	0	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
有条件停止	1	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
子程序结束	2	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
	3	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
	4	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
换刀	5	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	(1)

系统功能	下标 <n>																			
		输出特性，位																		
主轴正转	6	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	(1)
主轴反转	7	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	(1)
主轴停止	8	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	(1)
主轴定位	9	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	(1)
进给轴模式	10	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	(1)
自动换档	11	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	0	0	1	(0)	(0)	(0)	(0)	(1)
齿轮档 1	12	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	0	0	1	(0)	(0)	(0)	(0)	(1)
齿轮档 2	13	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	0	0	1	(0)	(0)	(0)	(0)	(1)
齿轮档 3	14	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	0	0	1	(0)	(0)	(0)	(0)	(1)
齿轮档 4	15	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	0	0	1	(0)	(0)	(0)	(0)	(1)
齿轮档 5	16	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	0	0	1	(0)	(0)	(0)	(0)	(1)
主轴转速	17	0	0	0	(0)	0	0	0	(0)	(0)	(0)	0	(0)	(1)	(0)	(0)	(0)	0	(0)	(1)
进给率	18	0	0	0	(0)	0	0	0	(0)	(0)	(0)	0	(0)	(1)	(0)	0	(1)	0	(0)	(1)
刀沿选择	19	0	0	0	(0)	0	0	0	(0)	(0)	(0)	0	(0)	(0)	(1)	0	(0)	0	(0)	(1)
DL	20	0	0	0	(0)	0	0	0	(0)	(0)	(0)	0	(0)	(0)	(1)	0	(0)	0	(0)	(1)
刀具选择	21	0	0	0	(0)	0	0	0	(0)	(0)	(0)	0	(0)	(0)	(1)	0	(0)	0	(0)	(1)

系统功能	下标 <n>																			
	输出特性，位																			
停止（组合式）	22	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
有条件停止（组合式）	23	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
子程序结束	24	0	0	0	(0)	0	0	0	0	(0)	0	0	1	0	0	0	0	0	(0)	(1)
步冲	25	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)
步冲	26	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)
步冲	27	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)
步冲	28	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)
步冲	29	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)
步冲	30	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)
步冲	31	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)
步冲	32	0	0	0	(0)	0	0	0	(0)	(0)	(0)	(0)	(0)	(1)	(0)	0	(0)	(0)	(0)	(1)

( ) 可对该值进行修改。

位的含义

位	含义
0	一个 OB1 周期后"normal（普通）"应答
	采用普通应答的辅助功能会在 OB1 周期开始时输出至 NC/PLC 接口。 此时系统通过辅助功能专用变更信号向 PLC 用户程序指示辅助功能有效。 组织块 OB1 完全运行一次后便会立即对辅助功能进行应答。 这相当于一个完整的 PLC 用户循环。 采用普通应答的辅助功能与写入了该功能的零件程序段同步输出。 如果零件程序段的执行（例如轨迹轴和/或定位轴运动）在应答辅助功能前结束，那么程序段切换将延迟，直至通过 PLC 应答。 在连续路径运行中，若与采用普通应答的辅助功能组合使用，那么保持恒定的轨迹速度需满足以下条件：在运行中输出辅助功能，并在到达程序段末尾前通过 PLC 进行应答。
1	通过 OB40"quick（快速）"应答
	采用快速应答的辅助功能会在下一个 OB1 周期前输出至 NC/PLC 接口。 此时系统通过辅助功能专用变更信号向 PLC 用户程序指示辅助功能有效。 PLC 基本程序会在下一个 OB40 周期中立即对辅助功能进行应答。 因此应答辅助功能并不能保证执行了对应的 PLC 用户功能。 辅助功能仍然在 OB1 周期中执行。 因此只有在该 OB1 周期结束后辅助功能才会接着输出给 PLC。 连续路径运行中在多个连续零件程序段中输出采用快速应答的辅助功能时，这一点尤为明显（轨迹速度降低）。 对于采用快速应答的辅助功能，无法确保 PLC 用户程序中的响应与程序段同步。 提示 只有在与用户自定义辅助功能一同使用时，才可将辅助功能的输出特性设置为"快速辅助功能"。
2	非预定义辅助功能
	采用此设置时，预定义辅助功能会被作为用户自定义辅助功能处理。 此时辅助功能不再会触发对应的系统功能，而只是输出至 PLC。 示例： 将辅助功能"主轴定位"（下标 9）重设为采用普通应答的用户自定义辅助功能，并在运行前输出。 MD22080 $MC_AUXFU_PREDEF_SPEC [9] = 'H25' (100101B)
3	不输出至 PLC
	辅助功能不输出至 PLC。
4	通过 PLC 应答后执行主轴响应
	在通过 PLC 应答后再执行对应的系统功能。
5	运行前输出
	在零件程序段中编写的运行（轨迹轴和/或程序段相关定位轴运动）前将辅助功能输出至 PLC。

位	含义
6	运行中输出
	在零件程序段中编写的运行（轨迹轴和/或程序段相关定位轴运动）期间将辅助功能输出至 PLC。
7	在程序段末尾输出
	在零件程序段中编写的运行（轨迹轴和/或程序段相关定位轴运动）完成后将辅助功能输出至 PLC。
8	程序段搜索类型 1、2、4 后不输出
	程序段搜索类型 1、2、4：程序段搜索中收集的辅助功能不输出。
9	在带程序测试的程序段搜索（类型 5，SERUPRO）中收集
	在带程序测试的程序段搜索中，辅助功能按照各组别收集在以下系统变量中： • $AC_AUXFU_M_VALUE[<n>] • $AC_AUXFU_M_EXT[<n>] • $AC_AUXFU_M_STATE[<n>]
10	在带程序测试的程序段搜索（类型 5，SERUPRO）中不输出
	在带程序测试的程序段搜索中，辅助功能不输出至 PLC。
11	跨通道辅助功能（SERUPRO）
	在带程序测试的程序段搜索（SERUPRO）中，辅助功能跨通道收集到全局辅助功能列表中。 提示 每个辅助功能组只会收集该组的最后一个辅助功能。
12	已通过同步动作输出（只读）
	若辅助功能已通过同步动作输出至 PLC，则该位置位。
13	隐性辅助功能（只读）
	若辅助功能已隐性输出至 PLC，则该位置位。
14	M01 生效（只读）
	若 M01 生效时辅助功能已输出至 PLC，则该位置位。
15	定位测试运行期间不输出
	在定位测试运行期间，辅助功能不输出至 PLC。
16	关闭步冲
17	激活步冲
18	步冲

说明

对于未定义输出特性的辅助功能，以下缺省输出特性生效：

- 位 0 = 1：输出持续一个 OB1 周期
- 位 7 = 1：在程序段末尾输出

13.2.3 参数设置

13.2.3.1 分组指定

程序段搜索时，辅助功能的处理方式通过其分组确定。 168 个可供使用的辅助功能组又划分为预定义组和用户自定义组：

预定义组：	1 ... 4	10 ... 12	72 ... 168
用户自定义组：	5 ... 9	13 ... 71	

默认状况下，每个预定义辅助功能属于一个辅助功能组。 通过以下机床数据可修改多数预定义辅助功能的分组。

MD22040 $MC_AUXFU_PREDEF_GROUP[<n>]（预定义辅助功能的分组指定）

若不将辅助功能指定给任何组，则须在机床数据中输入“0”值。

下标 <n> 为以下数字的预定义辅助功能不可修改分组：0、1、2、3、4、22、23、24

说明

第 1 辅助功能组和程序段搜索

程序段搜索中，第 1 辅助功能组的辅助功能只会被收集，而不会被输出。

13.2.3.2 类型、地址扩展和数值

辅助功能通过“类型”、“地址扩展”和“值”这几个参数进行编程（参见“辅助功能编程 (页 817)”章节）。

类型

“类型”参数用于定义辅助功能的标识符，例如：

"M"	代表附加功能
"S"	代表主轴功能
"F"	代表进给率

此时通过以下机床数据设置：

MD22050 $MC_AUXFU_PREDEF_TYPE[<n>]（预定义辅助功能的类型）

说明

预定义辅助功能的“类型”无法修改。

地址扩展

辅助功能的“地址扩展”参数用于确定同类型不同分量的地址。对于预定义辅助功能，“地址扩展”的值为辅助功能所关联的主轴编号。

此时通过以下机床数据设置：

MD22060 $MC_AUXFU_PREDEF_EXTENSION[<n>]（预定义辅助功能的地址扩展）

辅助功能汇总

若要将一个通道中所有主轴的某个辅助功能指定给同一个辅助功能组，那么须将“地址扩展”参数的值设为“-1”。

示例：

针对通道的所有主轴，辅助功能 M3（机床数据下标 = 6）指定给第 2 辅助功能组。

```
MD22040 $MC_AUXFU_PREDEF_GROUP[ 6 ]          = 2
MD22050 $MC_AUXFU_PREDEF_TYPE[ 6 ]           = "M"
MD22060 $MC_AUXFU_PREDEF_EXTENSION[ 6 ]      = -1
MD22070 $MC_AUXFU_PREDEF_VALUE[ 6 ]          = 3
```

值

“值”和“类型”参数用于定义辅助功能的含义，即基于此辅助功能激活的系统功能。

辅助功能的“值”在以下机床数据中定义：

MD22070 $MC_AUXFU_PREDEF_VALUE[<n>]（预定义辅助功能的数值）

说明

预定义辅助功能的“值”无法修改。个别预定义辅助功能的“值”可通过附加的机床数据重新配置（参见“关联辅助功能 (页 811)”章节）。

13.2.3.3 输出特性

“输出特性”参数用于定义预定义辅助功能输出至 NC/PLC 接口的时间，以及由 PLC 对其进行应答的时间。

此时通过以下机床数据设置：

MD22080 $MC_AUXFU_PREDEF_SPEC[<n>]（预定义辅助功能的输出特性）

以运行为基准的输出特性

运行前输出

- 前一零件程序段的运行（轨迹轴和/或程序段相关定位轴运行）通过准停结束。
- 辅助功能在当前零件程序段起始处输出。
- PLC 对辅助功能进行过应答后，当前零件程序段才开始运行（轨迹轴和/或定位轴运行）。
  - 输出持续一个 OB1 周期（普通应答）：一个 OB1 周期后输出
  - 输出持续一个 OB40 周期（快速应答）：一个 OB40 周期后输出

运行中输出

- 辅助功能在运行（轨迹轴和/或定位轴运行）开始时输出。
- 此时系统会降低当前零件程序段的轨迹速度，使到达程序段末尾所花费的时间大于 PLC 应答辅助功能所用的时间：
  - 输出持续一个 OB1 周期（普通应答）：一个 OB1 周期
  - 输出持续一个 OB40 周期（快速应答）：一个 OB40 周期

运行后输出

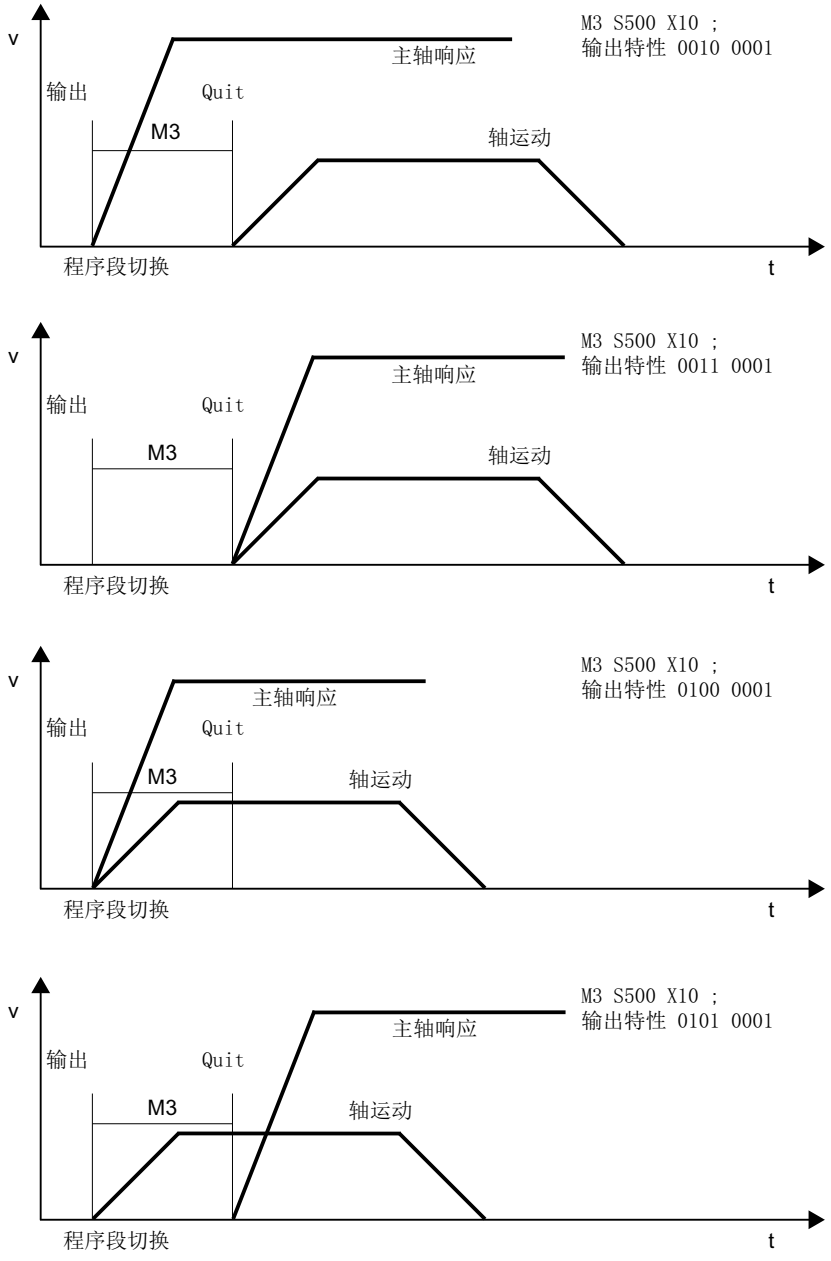
- 当前零件程序段的运行（轨迹轴和/或程序段相关定位轴运行）通过准停结束。
- 辅助功能在运行结束后输出。
- PLC 对辅助功能进行应答后，系统执行程序段切换。
  - 输出持续一个 OB1 周期（普通应答）：一个 OB1 周期后输出
  - 输出持续一个 OB40 周期（快速应答）：一个 OB40 周期后输出

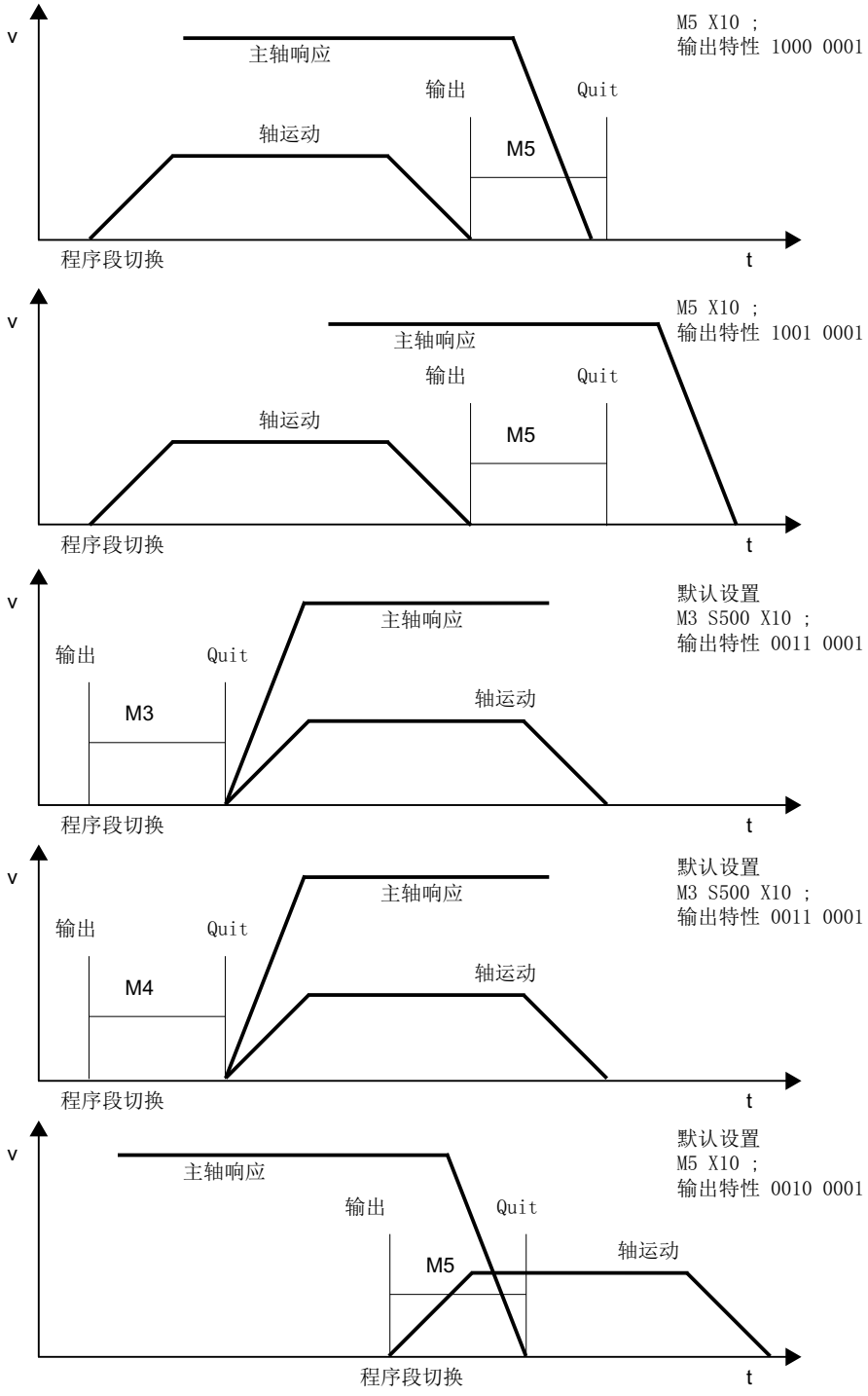
不同输出特性的示例

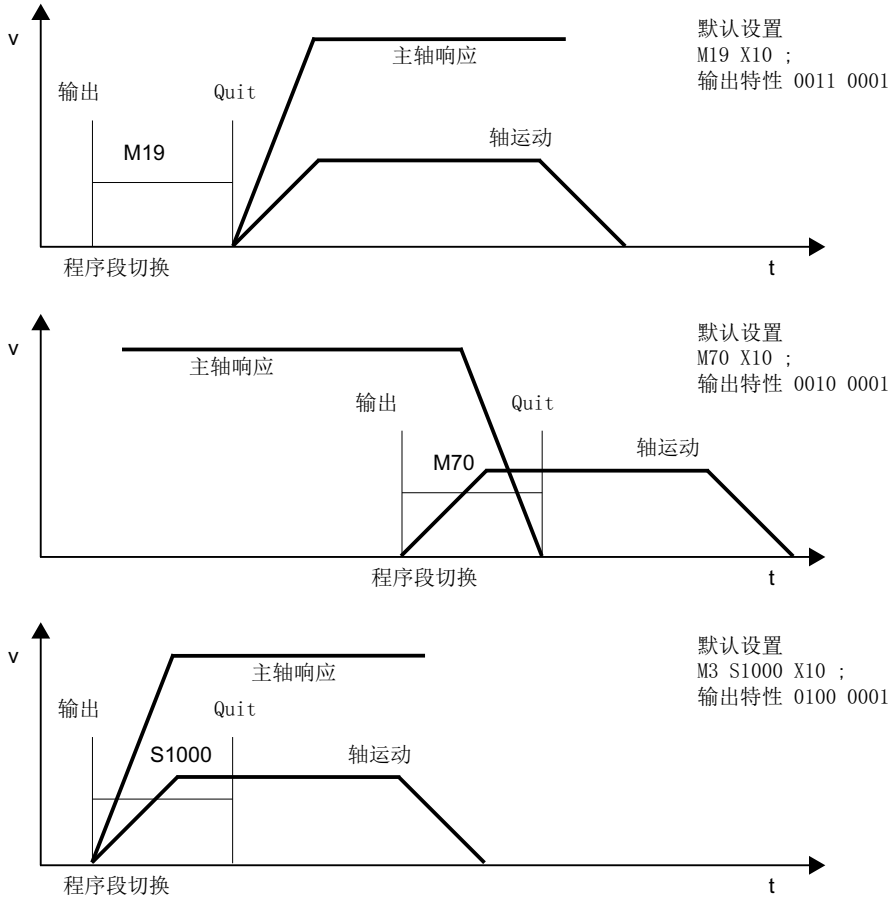
下图直观地展示了基于不同因素时的输出特性：

- 辅助功能的输出和应答
- 主轴响应（转速变化）
- 运行（速度变化）

图中“输出特性”下标注的二进制值以设置的输出特性（MD22080）为基准。




v
主轴响应
输出
Quit
轴运动
M5
程序段切换
t
M5 X10 ;
输出特性 1000 0001
v
主轴响应
输出
Quit
轴运动
M5
程序段切换
t
M5 X10 ;
输出特性 1001 0001
v
主轴响应
输出
Quit
M3
轴运动
程序段切换
t
默认设置
M3 S500 X10 ;
输出特性 0011 0001
v
主轴响应
输出
Quit
M4
轴运动
程序段切换
t
默认设置
M3 S500 X10 ;
输出特性 0011 0001
v
主轴响应
输出
Quit
M5
轴运动
程序段切换
t
默认设置
M5 X10 ;
输出特性 0010 0001



13.3 用户自定义辅助功能

用户自定义辅助功能的使用分为两个部分：

- 预定义辅助功能的扩展
- 用户专用辅助功能

预定义辅助功能的扩展

预定义辅助功能的机床数据只能存在一次，因此只能通过其确定通道中一根主轴的地址。 为了确定其他主轴的地址，必须参设置用户自定义辅助功能，作为对预定义辅助功能的扩展。

预定义辅助功能的扩展只基于“地址扩展”参数。 在“地址扩展”参数中输入辅助功能对应的主轴的编号。

可对以下系统功能所对应的预定义辅助功能进行扩展：

系统功能	类型		
		地址扩展 [1]	
			值
换刀	M	1	6
主轴正转	M	1	3
主轴反转	M	1	4
主轴停止	M	1	5
主轴定位	M	1	19
进给轴模式	M	1	70
自动换档	M	1	40
齿轮档 1	M	1	41
齿轮档 2	M	1	42
齿轮档 3	M	1	43
齿轮档 4	M	1	44
齿轮档 5	M	1	45
主轴转速	S	1	-1
刀具选择	T	1	-1

[1] 地址扩展 = 1 为预定义辅助功能的机床数据中使用的缺省值

示例：

系统功能“主轴正转”的预定义辅助功能扩展，针对通道中第 2 主轴和第 3 主轴。

辅助功能“主轴正转”，针对通道第 2 主轴：

MD22010 $MC_AUXFU_ASSIGN_TYPE [n] = "M"

MD22020 $MC_AUXFU_ASSIGN_EXTENSION [n] = 2

MD22030 $MC_AUXFU_ ASSIGN_VALUE [n] = 3

辅助功能“主轴正转”，针对通道第 3 主轴：

MD22010 $MC_AUXFU_ ASSIGN_TYPE [m] = "M"

MD22020 $MC_AUXFU_ASSIGN_EXTENSION [m] = 3

MD22030 $MC_AUXFU_ ASSIGN_VALUE [m] = 3

用户专用辅助功能

用户专用辅助功能有以下属性：

- 用户专用辅助功能只能用于激活用户功能。
- 用户专用辅助功能无法用于激活系统功能。
- 用户专用辅助功能会根据设置的输出特性输出至 PLC。
- 用户专用辅助功能的功能性由机床制造商/用户在 PLC 用户程序中实现。

13.3.1 参数设置

13.3.1.1 用户自定义辅助功能的最大数量

每个通道中用户自定义辅助功能的最大数量可通过以下机床数据设置：

MD11100 $MN_AUXFU_MAXNUM_GROUP_ASSIGN（用户自定义辅助功能的最大数量）

13.3.1.2 分组指定

程序段搜索时，辅助功能的处理方式通过其分组确定。 168 个可供使用的辅助功能组又划分为预定义组和用户自定义组：

预定义组：	1 ... 4	10 ... 12	72 ... 168
用户自定义组：	5 ... 9	13 ... 71	

默认状况下，每个用户自定义辅助功能被指定给第 1 个辅助功能组。 分组指定可通过以下机床数据修改：

MD22000 $MC_AUXFU_ASSIGN_GROUP[<n>]（用户自定义辅助功能的分组指定）

若不将辅助功能指定给任何组，则须在机床数据中输入“0”值。

说明

第 1 辅助功能组和程序段搜索

程序段搜索中，第 1 辅助功能组的辅助功能只会被收集，而不会被输出。

13.3.1.3 类型、地址扩展和数值

辅助功能通过类型、地址扩展和数值这几个参数进行编程（参见“辅助功能编程 (页 817)”章节）。

类型

“类型”参数用于定义辅助功能的标识符。

用户自定义辅助功能的标识符有：

类型	名称	含义
"H"	辅助功能	用户专用辅助功能
"M"	附加功能	预定义辅助功能的扩展
"S"	主轴功能	
"T"	刀具号	

此时通过以下机床数据设置：

MD22010 $MC_AUXFU_ASSIGN_TYPE[<n>]（用户自定义辅助功能的类型）

地址扩展

MD22020 $MC_AUXFU_ASSIGN_EXTENSION[<n>]（用户自定义辅助功能的地址扩展）

对于用户专用辅助功能，地址扩展参数的功能不确定。 其通常用于区分“值”相同的辅助功能。

辅助功能汇总

若需将所有类型和数值相同的辅助功能指定给同一个辅助功能组，那么须将“地址扩展”参数的值设为“-1”。

示例：

将所有数值 = 8 的用户专用辅助功能指定给第 10 辅助功能组。

```
MD22000 $MC_AUXFU_ASSIGN_GROUP [ 1 ]          = 10
MD22010 $MC_AUXFU_ ASSIGN_TYPE [ 1 ]          = "H"
MD22020 $MC_AUXFU_ ASSIGN_EXTENSION [ 1 ]     = -1
MD22030 $MC_AUXFU_ ASSIGN_VALUE [ 1 ]         = 8
```

值

MD22030 $MC_AUXFU_ASSIGN_VALUE[<n>]（用户自定义辅助功能的数值）

对于用户专用辅助功能，“值”参数的功能不确定。 通常其用于激活对应的 PLC 用户功能。

辅助功能汇总

若需将所有类型和地址扩展相同的辅助功能指定给同一个辅助功能组，那么须将“值”参数的值设为"-1"。

示例：

将所有地址扩展 = 2 的用户专用辅助功能指定给第 11 辅助功能组。

MD22000 $MC_AUXFU_ASSIGN_GROUP [2]	= 11
MD22010 $MC_AUXFU_ ASSIGN_TYPE [2]	= "H"
MD22020 $MC_AUXFU_ ASSIGN_EXTENSION [2]	= 2
MD22030 $MC_AUXFU_ ASSIGN_VALUE [2]	= -1

13.3.1.4 输出特性

用户自定义辅助功能的“输出特性”可通过以下机床数据设置：

MD22035 $MC_AUXFU_ASSIGN_SPEC[<n>]（用户自定义辅助功能的输出特性）

对各输出参数的说明请见预定义辅助功能的“输出特性 (页 804)”章节。 该章节的内容大体上也适用于用户自定义辅助功能的输出特性。

13.4 关联辅助功能

功能

关联辅助功能为用户自定义辅助功能，其与对应的预定义辅助功能有相同的作用。可为以下预定义辅助功能关联用户自定义辅助功能：

- M0（编程停止）
- M1（可选停止）

G 指令组

为关联辅助功能分配一个与预定义辅助功能相符的 G 功能组。

参数设置

通过以下机床数据将一个用户自定义辅助功能关联至上述预定义辅助功能。

- MD22254 $MC_AUXFU_ASSOC_M0_VALUE（用于“编程停止”的关联 M 功能）
- MD22256 $MC_AUXFU_ASSOC_M1_VALUE（用于“可选停止”的关联 M 功能）

选择

通过在 SINUMERIK Operate 操作界面的“自动”>“程序控制”操作区域将 HMI/PLC 接口信号 <HmiChan>.basic.in.m01AssociatedAuxFuncSelected 置位，便可选择“关联辅助功能” (M-1)。

接口信号根据 FC1 参数 MMCToIf 的值，从 PLC 基本程序传输至 NC/PLC 接口信号 <Chan>.basic.out.m01AssociatedAuxFunc：

- "TRUE":传输
- "FALSE":不传输

参数的缺省设置为 "TRUE"。

说明

通过 SINUMERIK Operate 操作界面的选择方法

如果机床数据中编程有关联辅助功能，则其选择只会在操作区域“AUTOMATIC”>“程序控制”下显示。

应用

关联辅助功能可用于：

- 主程序
- 子程序
- 循环

说明

关联辅助功能**不可**在同步动作中使用。

NC/PLC 接口信号

使用关联用户自定义辅助功能时，输出至 NC/PLC 接口的信号与使用对应预定义辅助功能时相同；不过输出的数值还是用户自定义辅助功能的数值（“值”参数），用于区分实际编写了哪个辅助功能，这样一来便可在 PLC 用户程序中区分预定义和用户自定义辅助功能。

关联辅助功能的 NC/PLC 接口信号：

- <HmiChan>.basic.in.m01AssociatedAuxFuncSelected（关联辅助功能已选择）
- <Chan>.basic.out.m01AssociatedAuxFunc（激活关联辅助功能）
- <Chan>.basic.in.m00m01AssociatedAuxFuncActive（关联辅助功能生效）

边界条件

注意下列边界条件：

- 一个用户自定义辅助功能不允许被多次关联。
- 预定义辅助功能（例如 M3、M4、M5 等）不可被关联。

示例

1. 将用户自定义辅助功能 M111 关联至 M0：
   MD22254 $MC_AUXFU_ASSOC_M0_VALUE = 111
   现在用户自定义辅助功能 M111 便具有和 M0 相同的功能。
2. 将用户自定义辅助功能 M222 关联至 M1：
   MD22256 $MC_AUXFU_ASSOC_M1_VALUE = 222
   现在用户自定义辅助功能 M222 便具有和 M1 相同的功能。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.m00m01AssociatedAuxFuncActive	LBP_Chan*.E_AssM01	DB21, ...DBX318.5

PLC → NC

Basic Program Plus	Basic Program	
<Chan>.basic.out.m01AssociatedAuxFunc	LBP_Chan*.A_NCKrelatedM01	DB21, ...DBX30.5

OP → PLC

Basic Program Plus	Basic Program	
<HmiChan>.basic.in.m01AssociatedAuxFuncSelected	LBP_HMI.E_MMC_M01AssocNC	DB21, ...DBX24.4

13.5 类型专用输出特性

功能

基于零件程序段中编写的运行的辅助功能输出特性可针对类型定义。

参数设置

类型专用输出特性通过以下机床数据设置：

MD22200 $MC_AUXFU_**M**_SYNC_TYPE（M 功能输出时间点）

MD22210 $MC_AUXFU_**S**_SYNC_TYPE（S 功能输出时间点）

MD22220 $MC_AUXFU_**T**_SYNC_TYPE（T 功能输出时间点）

MD22230 $MC_AUXFU_**H**_SYNC_TYPE（H 功能输出时间点）

MD22240 $MC_AUXFU_**F**_SYNC_TYPE（F 功能输出时间点）

MD22250 $MC_AUXFU_**D**_SYNC_TYPE（D 功能输出时间点）

MD22252 $MC_AUXFU_**DL**_SYNC_TYPE（DL 功能输出时间点）

可设置以下输出特性：

MD $MC_AUXFU_**xx**_SYNC_TYPE = <数值>

值	输出特性
0	运行前输出
1	运行中输出
2	在程序段末尾输出
3	不输出至 PLC
4	根据通过 MD22080 定义的输出特性进行输出

对各种输出特性的说明请见“输出特性 (页 804)”章节。

说明

各辅助功能类型可设置的输出特性请见参数手册中的“详细机床数据说明”。

示例

在一个包含运行的零件程序段中输出具有不同输出特性的辅助功能。

设置的输出特性：

MD22200 $MC_AUXFU_M_SYNC_TYPE = 1 ⇒ M 功能：运行**中**输出

MD22220 $MC_AUXFU_T_SYNC_TYPE = 0 ⇒ T 功能：运行**前**输出

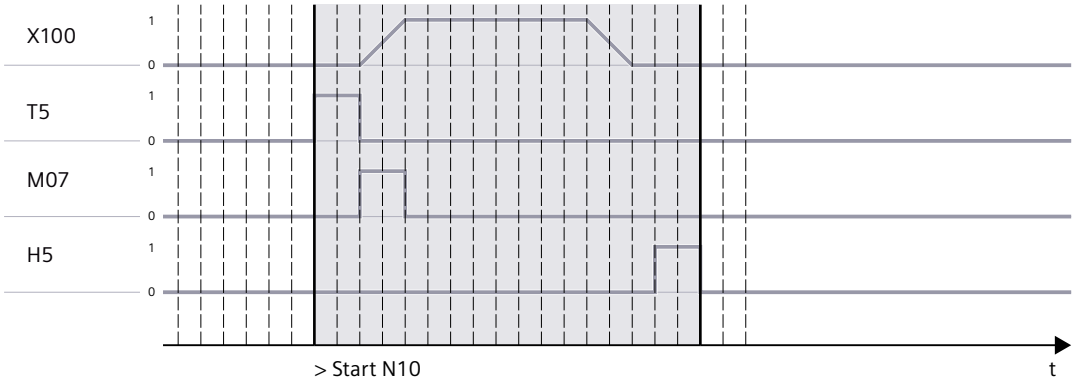
MD22230 $MC_AUXFU_H_SYNC_TYPE = 2 ⇒ H 功能：**在程序段末尾**输出

零件程序段：

程序代码

```
...
N10 G01 X100 M07 H5 T5
...
```

辅助功能输出的时间顺序：



13.6 设置的输出特性的优先级

设置的输出特性的优先级须参考以下标准分别加以考量：

- 输出时长（普通应答/快速应答）
- 以运行为基准的输出（运行前/中/后输出）

通常情况下，未设置高优先级的输出特性时，优先级低一级的输出特性生效。

输出时长

以下优先级适用于输出时长：

优先级	输出特性	定义方式：
最高	辅助功能专用	零件程序指令：QU(...) （参见“可编程的输出时长 (页 818)”章节）
↓	辅助功能专用	MD22035 $MC_AUXFU_ASSIGN_SYNC[<n>] MD22080 $MC_AUXFU_PREDEF_SYNC[<n>]
↓	组专用	MD11110 $MC_AUXFU_GROUP_SPEC[<n>]
最低	未定义	缺省输出特性：输出持续一个 OB1 周期

以运行为基准的输出

以下优先级适用于以运行为基准的输出：

优先级	输出特性	定义方式：
最高	辅助功能专用	MD22035 $MC_AUXFU_ASSIGN_SYNC[<n>] MD22080 $MC_AUXFU_PREDEF_SYNC[<n>]
↓	组专用	MD11110 $MC_AUXFU_GROUP_SPEC[<n>]
↓	类型专用	MD22200 $MC_AUXFU_M_SYNC_TYPE MD22210 $MC_AUXFU_S_SYNC_TYPE MD22220 $MC_AUXFU_T_SYNC_TYPE MD22230 $MC_AUXFU_H_SYNC_TYPE MD22240 $MC_AUXFU_F_SYNC_TYPE MD22250 $MC_AUXFU_D_SYNC_TYPE MD22252 $MC_AUXFU_DL_SYNC_TYPE
最低	未定义	缺省输出特性：在程序段末尾输出

说明

不含轨迹运行的零件程序段

在不含轨迹运行的零件程序段中（同样针对定位轴和主轴），辅助功能立即在一个块中输出。

13.7 辅助功能编程

句法

辅助功能在零件程序段中通过以下句法编写：
<类型>[<地址扩展>=]<数值>

说明

若未编写地址扩展，则会隐性设置地址扩展 = 0。

地址扩展 = 0 的预定义辅助功能总是基于通道的主主轴。

符号定址

“地址扩展”和“值”参数的值也可通过符号设定。地址扩展的符号名称必须在方括号中设定。

示例：

针对第 1 主轴的辅助功能 M3（主轴正转）的符号编程：

程序代码	注释
DEF SPINDEL_NR=1	; 通道中的第 1 主轴
DEF 旋转方向=3	; 旋转方向向右
N100 M[主轴号]=旋转方向	; 相应地：M1=3

说明

使用符号名称编写辅助功能的情形下，该辅助功能输出至 PLC 时不传输符号名称，而是对应的数值。

示例

示例 1：预定义辅助功能的编程

程序代码	注释
N10 M3	; “主轴正转”，针对通道主主轴。
N20 M0=3	; “主轴正转”，针对通道主主轴。
N30 M1=3	; “主轴正转”，针对通道第 1 主轴。
N40 M2=3	; “主轴正转”，针对通道第 2 主轴。

示例 2：辅助功能编程示例，对应值输出至 PLC

程序代码	注释
DEF 冷却剂=12	; 输出至 PLC：- - -
DEF 润滑剂=130	; 输出至 PLC：- - -
H[冷却剂]=润滑剂	; 输出至 PLC：H12=130
H=冷却剂	; 输出至 PLC：H0=12
H5	; 输出至 PLC：H0=5
H=5.379	; 输出至 PLC：H0=5.379
H17=3.5	; 输出至 PLC：H17=3.5
H[冷却剂]=13.8	; 输出至 PLC：H12=13.8
H='HFF13'	; 输出至 PLC：H0=65299
H='B1110'	; 输出至 PLC：H0=14
H5.3=21	; 错误

13.8 可编程的输出时长

功能

对于设置了“输出持续一个 OB1 周期（慢速应答）”输出特性的用户专用辅助功能，可通过零件程序指令 QU（Quick）将其定义为采用快速应答的辅助功能，用于单次输出。

句法

可在零件程序段中通过以下句法定义采用快速应答的辅助功能：
<类型>[<地址扩展>]=QU(<数值>)

示例

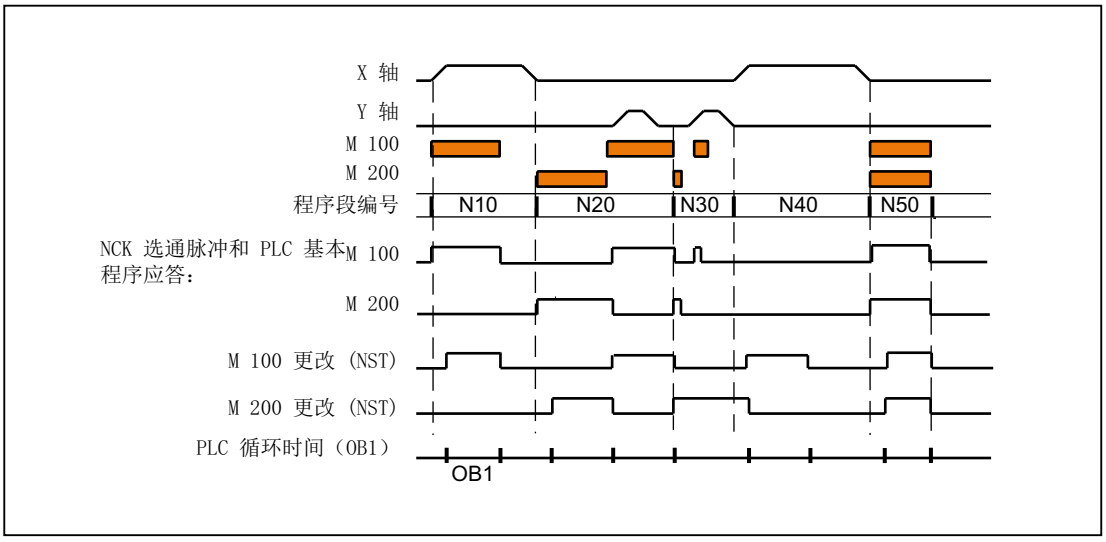
在一个零件程序段中辅助功能 M100 和 M200 有不同输出特性。辅助功能的输出特性如下设置：

- M100
  - 输出持续一个 OB1 周期（慢速应答）
  - 运行中输出
- M200
  - 输出持续一个 OB1 周期（慢速应答）
  - 运行前输出

程序代码	注释
N10 G94 G01 X50 M100	; 输出 M100：运动期间
	; 应答：慢
N20 Y5 M100 M200	; 输出 M200：运动前
	; 输出 M100：运动期间
	; 应答：慢
N30 Y0 M=QU(100) M=QU(200)	; 输出 M200：运动前
	; 输出 M100：运动期间
	; 应答：快
N40 X0	
N50 M100 M200	; 输出 M200：立即 1)
	; 输出 M100：立即 1)
	; 应答：慢
M17	

1) 不包含运行时，辅助功能总是即时输出至 PLC。

下图显示了零件程序的时间顺序。特别需要注意的是执行零件程序段 N20 和 N30 时的时间差。



13.9 输出至 PLC 的辅助功能

功能

在将辅助功能输出至 PLC 时，下列信号和数值会传输至 NC/PLC 接口：

- 变更信号
- “地址扩展”参数
- “值”参数

NC/PLC 接口中的数据区域

变更信号和辅助功能的数值位于 NC/PLC 接口的以下数据区域中：

- 从 NC 通道传输辅助功能时的变更信号：
  <Chan>.auxMFuncs[1].in.hasChanged
  <Chan>.auxMFuncs[1].in.notDecoded
  <Chan>.auxMFuncs[1].in.fastAckActive
  <Chan>.auxFFuncs[1].in.fastAckActive
- 传输的 M 功能和 S 功能：
  <Chan>.auxMFuncs[1].in.extendedAddress
  <Chan>.auxMFuncs[1].in.value
  <Chan>.auxSFuncs[1].in.extendedAddress
  <Chan>.auxSFuncs[1].in.value
  <Chan>.auxSFuncs[1].in.extendedAddress
- 传输的 T 功能、D 功能和 DL 功能：
  <Chan>.auxTFuncs[1].in.extendedAddresss
  <Chan>.auxTFuncs[1].in.value
  <Chan>.auxDFuncs[1].in.extendedAddress
  <Chan>.auxDFuncs[1].in.value
  <Chan>.auxDLFuncs[1].in.value
- 传输的 H 功能和 F 功能：
  <Chan>.auxHFuncs[1].in.extendedAddress
  <Chan>.auxHFuncs[1].in.value
  <Chan>.auxFFuncs[1].in.extendedAddress
  <Chan>.auxFFuncs[1].in.value
- 解译的 M 信号（M0 - M99）：
  <Chan>.auxMDynFuncs.in.m00Decoded

更多信息

- 功能手册之 PLC 和基本程序，PLC 基本程序
- 功能手册之 PLC 和基本程序，PLC 基本程序参考

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.auxMFuncs[1].in.hasChanged	LBP_Chan*.M1Change - M5Change	DB21, ...DBX058.1 ..4
<Chan>.auxMFuncs[1].in.notDecoded	LBP_Chan*.M1NDec- M5NDec	DB21, ...DBX059.1 ..4
<Chan>.auxMFuncs[1].in.fastAckActive	LBP_Chan*.M1Quick - M5Quick	DB21, ...DBX066.1 ..4
<Chan>.auxFFuncs[1].in.fastAckActive	LBP_Chan*.F1Quick - F6Quick	DB21, ...DBX067.0 ..5
<Chan>.auxMFuncs[1].in.extendedAddress	LBP_Chan*.ExtM1 - ExtM5	DB21, ...DBW068, .DBW074, .DBW080, .DBW086, .DBW092
<Chan>.auxMFuncs[1].in.value	LBP_Chan*.ExtS1 - ExtS3	DB21, ...DBD070, .DBD076, .DBD082, .DBD088, .DBD094
<Chan>.auxSFuncs[1].in.extendedAddress	LBP_Chan*.S1 - S3	DB21, ...DBW098, .DBW104, .DBW110
<Chan>.auxSFuncs[1].in.value	LBP_Chan*.M1 - M5	DB21, ...DBW100, .DBW106, .DBW112
<Chan>.auxTFuncs[1].in.extendedAddress	LBP_Chan*.ExtT1 - ExtT3	DB21, ...DBW116, .DBW120, .DBW124
<Chan>.auxTFuncs[1].in.value	LBP_Chan*.T1 - T3	DB21, ...DBW118, .DBW122, .DBW126
<Chan>.auxDFuncs[1].in.extendedAddress	LBP_Chan*.ExtD1 - ExtD3	DB21, ...DBW128, .DBW130, .DBW132
<Chan>.auxDFuncs[1].in.value	LBP_Chan*.D1 - D3	DB21, ...DBB129, .DBB131, .DBB133
<Chan>.auxDLFuncs[1].in.value	LBP_Chan*.E	DB21, ...DBD136
<Chan>.auxHFuncs[1].in.extendedAddress	LBP_Chan*.ExtH1 - ExtH3	DB21, ...DBW140, .DBW146, .DBW152

Basic Program Plus	Basic Program	
<Chan>.auxHFuncs[1].in.value	LBP_Chan*.H1 - H3	DB21, ...DBD142, .DBD148, .DBD154
<Chan>.auxFFuncs[1].in.extendedAddress	LBP_Chan*.ExtF1 - ExtF6	DB21, ...DBB158, .DBB164, .DBB170, .DBB176, .DBB182, .DBB188
<Chan>.auxFFuncs[1].in.value	LBP_Chan*.F1 - F6	DB21, ...DBB160, .DBB166, .DBB172, .DBB178, .DBB184, .DBB190
<Chan>.auxMDynFuncs.in.m00Decoded	LBP_Chan*.MDyn[x]	DB21, ...DBX194.0..206.3

13.10 无程序段切换延迟的辅助功能

功能

即便是对于设置和/或编写了以下输出特性的辅助功能：

- “输出持续一个 OB40 周期（快速应答）”
- “运行前输出”或“运行中输出”

在连续路径运行（行程短，速度高）中仍可能出现速度跃变，因为在到达程序段末尾前须等待 PLC 应答辅助功能。 为了避免此类速度跃变，可独立于辅助功能应答进行程序段切换。

参数设置

使用快速辅助功能时对程序段切换延迟的抑制通过以下机床数据设置：

MD22100 $MC_AUXFU_QUICK_BLOCKCHANGE（使用快速辅助功能时的程序段切换延迟）

值	含义
0	在辅助功能快速输出至 PLC 时，程序段切换延迟直至 PLC 应答（OB40）。
1	在辅助功能快速输出至 PLC 时，程序段切换不延迟。

前提条件

对于输出时无程序段延迟的辅助功能，无法保证该功能与其程序段之间的同步性。 在“最差”情形下，切换至下一零件程序段后应答还需一个 OB40 周期，辅助功能执行还需一个 OB1 周期。

13.11 带隐性预处理停止的 M 功能

功能

若需配合辅助功能触发预处理停止，可通过零件程序指令 STOPRE 显性编程。若需在 M 功能的编程中默认触发预处理停止，可通过以下机床数据设置此 M 功能专用特性：

MD10713 $MN_M_NO_FCT_STOPRE[<n>]（带预处理停止的 M 功能）

示例

用户自定义 M 功能 M88 需要触发预处理停止。

参数设置：

MD10713 $MN_M_NO_FCT_STOPRE [0] = 88

应用：

零件程序（节选）

程序代码	注释
...	
N100 G0 X10 M88	; 运行，并通过 M88 隐性触发预处理停止。
N110 Y=R1	; 在运行完成和 M 功能被应答后才会解译 N110。
...	

前提条件

若在零件程序中借由以下两种方式之一通过 M 功能调用了一个子程序，则不会触发预处理停止：

- MD10715 $MN_M_NO_FCT_CYCLE（须由子程序替换的 M 功能）
- M98（ISO 车铣编程语言）

13.12 溢出转存时的特性

溢出转存

通过 SINUMERIK 操作界面，可在以下功能开始前：

- 零件程序启动
- 恢复执行中断的零件程序

通过“溢出转存”功能修改启动时输出的辅助功能。

此功能的应用场合例如有：

- 在程序段搜索后添加辅助功能
- 恢复初始状态用于零件程序试运行

可溢出转存的辅助功能类型

下列辅助功能类型可被溢出转存：

- M（附加功能）
- S（主轴转速）
- T（刀具号）
- H（辅助功能）
- D（刀具补偿号）
- DL（总补偿）
- F（进给率）

生效时间

经过溢出转存的辅助功能（例如 M3，主轴正转）保持生效，直至其被同一辅助功能组的另一个辅助功能覆盖，或由于重新溢出转存或零件程序段中的编程被覆盖。

13.13 程序段搜索时的特性

13.13.1 程序段搜索类型 1、2 和 4 时的辅助功能输出

输出特性

在程序段搜索类型 1、2 和 4 中，系统会根据组收集辅助功能。系统启动后，每个辅助功能组的最后一个辅助功能会在真正的重入程序段前在一个单独的零件程序段中输出，特性如下：

- 输出持续一个 OB1 周期（普通应答）
- 运行前输出

输出控制

通过以下机床数据的位 8，可设置在程序段搜索后是否向 PLC 输出辅助功能：

- MD22080 $MC_AUXFU_PREDEF_SPEC[<n>]
  （预定义辅助功能的输出特性）
  其中 <n> = 系统功能下标（0 ... 32）
- MD22035 $MC_AUXFU_ASSIGN_SPEC[<n>]
  （用户自定义辅助功能的输出特性）
  其中 <n> = 辅助功能下标（0 ... 254）
- MD11110 $MN_AUXFU_GROUP_SPEC[<n>]
  （组中辅助功能的输出特性）
  其中 <n> = 组下标（0 ... 63）

位	值	含义
10	0	在程序段搜索类型 1、2、4 中输出
	1	在程序段搜索类型 1、2、4 中不输出

此特性对显示或变量 $AC_AUXFU_STATE[<n>]、$AC_AUXFU_VALUE[<n>]、$AC_AUXFU_EXT[<n>] 无影响。
程序段搜索后辅助功能仍被作为“已收集”，尽管其未被输出至 PLC。

收集时，程序段搜索后未输出的辅助功能仍会覆盖位 8 未置位的辅助功能。

用户可在程序段搜索后查询收集的辅助功能，某些情形下还可通过零件程序或同步动作将其再次输出。

说明

下列辅助功能不会被收集：

- 未指定分组的辅助功能
- 被指定给第 1 辅助功能组的辅助功能

辅助功能的溢出转存

程序段搜索完成后，下一次系统启动时会输出收集的辅助功能。若还需输出额外的辅助功能，可通过“溢出转存”功能添加（参见“溢出转存时的特性 (页 825)”章节）。

M19（主轴定位）相关特性

程序段预处理后，系统总是会执行最后使用 M19 编写的主轴定位，即便是在 M19 零件程序段和目标程序段间仍编写了其他主轴专用辅助功能。因此必须在 PLC 用户程序中通过运行指令的接口信号导出所需主轴使能：

<Axis>.basic.in.traversMinusCommandActive /
<Axis>.basic.in.traversPlusCommandActive（运行指令“负” / “正”）

主轴专用辅助功能 M3、M4、M5 不适用于此情形，因为有时在主轴定位后其才被输出至 PLC。

程序段搜索的详细信息请见“BAG、通道、程序运行、复位特性 (页 29)”章节。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Axis>.basic.in.traversMinusCommandActive	LBP_Axis*.E_TCMinus	DB31, ...DBX64.6
<Axis>.basic.in.traversPlusCommandActive	LBP_Axis*.E_TCPlus	DB31, ...DBX64.7

13.13.2 将一个辅助功能指定给多个组

功能

通过分组指定（MD22000 $MC_AUXFU_ASSIGN_GROUP）也可将用户自定义辅助功能指定给多个组。 程序段搜索时，系统会基于所涉及的所有功能组收集这些辅助功能。

说明

预定义辅助功能只能指定给一个功能组。

示例

在 DIN 标准中，下列 M 指令设计用于冷却剂输出：

- M7：冷却剂 2 ON
- M8：冷却剂 1 ON
- M9：冷却剂 1 和 2 OFF

若需使两组冷却剂能够同时生效：

- 必须在两个独立的功能组中收集 M7 和 M8（例如组 5 和组 6）
- 必须将 M9 同时指定给这两个组，例如：
  - 组 5：M7、M9
  - 组 6：M8、M9

参数设置：

MD11100 $MN_AUXFU_MAXNUM_GROUP_ASSIGN = 4

MD22000 $MC_AUXFU_ASSIGN_GROUP [0] = 5

MD22000 $MC_AUXFU_ASSIGN_GROUP [1] = 5

MD22000 $MC_AUXFU_ASSIGN_GROUP [2] = 6

MD22000 $MC_AUXFU_ASSIGN_GROUP [3] = 6

MD22010 $MC_AUXFU_ASSIGN_TYPE [0] = M

MD22010 $MC_AUXFU_ASSIGN_TYPE [1] = M

MD22010 $MC_AUXFU_ASSIGN_TYPE [2] = M

MD22010 $MC_AUXFU_ASSIGN_TYPE [3] = M

MD22020 $MC_AUXFU_ASSIGN_EXTENSION [0] = 0

MD22020 $MC_AUXFU_ASSIGN_EXTENSION [1] = 0

MD22020 $MC_AUXFU_ASSIGN_EXTENSION [2] = 0

MD22020 $MC_AUXFU_ASSIGN_EXTENSION [3] = 0

MD22030 $MC_AUXFU_ASSIGN_VALUE [0] = 7

MD22030 $MC_AUXFU_ASSIGN_VALUE [1] = 9

MD22030 $MC_AUXFU_ASSIGN_VALUE [2] = 8

MD22030 $MC_AUXFU_ASSIGN_VALUE [3] = 9

MD22035 $MC_AUXFU_ASSIGN_SPEC [0] = 'H121'

MD22035 $MC_AUXFU_ASSIGN_SPEC [1] = 'H121'

MD22035 $MC_AUXFU_ASSIGN_SPEC [2] = 'H121'

MD22035 $MC_AUXFU_ASSIGN_SPEC [3] = 'H121'

零件程序（节选）：

程序代码
... N10 ... M8 N20 ... M9 N30 ... M7 ...

程序段搜索时，系统会基于组 5 和组 6 收集辅助功能 M9。

查询收集到的 M 辅助功能：

第 5 组的 M 功能：$AC_AUXFU_M_VALUE [4] = 7

第 6 组的 M 功能：$AC_AUXFU_M_VALUE [5] = 9

13.13.3 生效的 M 辅助功能的时间戳

在程序段搜索后输出收集的辅助功能时，必须遵循收集时的顺序。 因此每个组都配有一个时间戳，其可通过以下系统变量针对组查询：

$AC_AUXFU_M_TICK[<n>]（生效的 M 辅助功能的时间戳）

13.13.4 确定输出顺序

功能

为了使编程人员方便地确定 M 功能输出顺序，特提供以下预定义程序：

```
AUXFUMSEQ(VAR INT _NUM_IN, VAR INT _M_IN[], VAR INT _EXT_IN[],
VAR INT _NUM_OUT, VAR INT _M_OUT[], VAR INT _EXT_OUT[])
```

输入参数：

`VAR INT _NUM_IN:`	相关 M 指令的数量
`VAR INT _M_IN[]:`	相关 M 代码的字段
`VAR INT _EXT_IN[]:`	相关 M 地址扩展的字段

输出参数：

`VAR INT _NUM_OUT:`	确定的 M 代码的数量
`VAR INT _M_OUT[]:`	确定的 M 代码的字段
`VAR INT _EXT_OUT[]:`	确定的 M 地址扩展的字段

此功能用于为给定的 M 代码确定针对组收集的 M 辅助功能的输出顺序。 顺序取决于收集时间点 $AC_AUXFU_M_TICK[<n>]（参见 “生效的 M 辅助功能的时间戳 (页 829)”章节）。

一个特定的 M 代码只会被考虑一次，即便其归属于多个组。 若相关 M 指令的数量小于或等于 0，则会输出收集的所有 M 代码。 相关 M 指令的数量上限为 64。

示例

用于冷却剂输出的 M 指令：

- M7：冷却剂 2 ON
- M8：冷却剂 1 ON
- M9：冷却剂 1 和 2 OFF

分组指定：

- 组 5：M7、M9
- 组 6：M8、M9

零件程序（节选）：

程序代码

```
...
N10 ... M8
N20 ... M9
N30 ... M7
...
```

程序段搜索时，系统会基于组收集辅助功能。 每个辅助功能组的最后一个辅助功能会在程序段搜索结束后输出至 PLC：

- 组 5：M7
- 组 6：M9

若以 M7 → M9 的顺序输出，那么之后将无冷却剂生效。 但是冷却剂 2 需要在程序运行中生效。 因此 M 辅助功能正确的输出顺序需通过一个含预定义程序 AUXFUMSEQ(...) 的 ASUB 确定：

程序代码

```
DEF INT _I, _M_IN[3], _EXT_IN[3], _NUM_OUT, _M_OUT[2], _EXT_OUT[2]
_M_IN[0]=7 _EXT_IN[0]=0
_M_IN[1]=8 _EXT_IN[1]=0
_M_IN[2]=9 _EXT_IN[2]=0
AUXFUMSEQ(3,_M_IN,_EXT_IN,_NUM_OUT,_M_OUT,_EXT_OUT)
FOR _I = 0 TO _NUM_OUT-1
   M[_EXT_OUT[_I]]=_M_OUT[_I]
ENDFOR
```

13.13.5 抑制主轴专用辅助功能的输出

功能

在特定情形下（例如换刀时），程序段搜索时收集的主轴专用辅助功能可能不需要在动作程序段中输出，而是在随后的某个时间点上（例如换刀后）再输出。为此可抑制程序段搜索后主轴专用辅助功能的自动输出。此时可于稍后的时间点通过溢出转存功能或 ASUB 手动输出。

参数设置

通过以下机床数据抑制程序段搜索后主轴专用辅助功能的自动输出：

MD11450 $MN_SEARCH_RUN_MODE（程序段搜索后的特性）

位	值	含义
2	0	主轴专用辅助功能在动作程序段中输出
	1	抑制辅助功能在动作程序段中的输出。

系统变量

程序段搜索时，主轴专用辅助功能总是存储在下列系统变量中，与上文所述设置无关：

系统变量	说明	
$P_SEARCH_S [<n>]	收集的主轴转速	
	值域：	0 ... Smax
$P_SEARCH_SDIR [<n>]	收集的主轴旋转方向	
	值域：	3, 4, 5, -5, -19, 70
$P_SEARCH_SGEAR [<n>]	收集的主轴齿轮档 M 功能	
	值域：	40 ... 45
$P_SEARCH_SPOS [<n>]	收集的主轴位置	
	值域：	0 ... MD30330 $MA_MODULO_RANGE（模数区域的大小）
	或者	
	收集的行程	
	值域：	-100000000 ... 100000000
$P_SEARCH_SPOSMODE [<n>]	收集的位置逼近模式	
	值域：	0 ... 5

随后需要输出主轴专用辅助功能时，可在一个 ASUB 中读取系统变量，并于动作程序段输出后进行输出：

```
<Chan>.basic.in.blockSearchLastActionActive = 1
```
（最后的动作程序段生效）

说明

系统变量 $P_S、$P_DIR 和 $P_SGEAR 的内容可能会在程序段搜索后由于同步而丢失。

ASUB、程序段搜索以及动作程序段的更多详细信息请见“BAG、通道、程序运行、复位特性 (页 29)”章节。

示例

进行程序段搜索，抑制主轴专用辅助功能的输出，并在动作程序段输出后启动一个 ASUB。

参数设置：MD11450 $MN_SEARCH_RUN_MODE，位 2 = 1

在 N55 程序段搜索后启动 ASUB。

零件程序：

程序代码	注释
N05 M3 S200	; 主轴 1
N10 G4 F3	
N15 SPOS=111	; 在 ASUB 中将主轴 1 定位至 111 度
N20 M2=4 S2=300	; 主轴 2
N25 G4 F3	
N30 SPOS[2]=IC(77)	; 主轴 2 增量运行 77 度
N55 X10 G0	; 目标程序段
N60 G4 F10	
N99 M30	

ASUB：

程序代码	注释
PROC ASUP_SAVE	
MSG (“输出主轴功能”)	
DEF INT SNR=1	
AUSG_SPI:	
M[SNR]=$P_SEARCH_SGEAR[SNR]	; 输出齿轮档。
S[SNR]=$P_SEARCH_S[SNR]	; 输出转速（M40 时确定匹配的齿轮档）。
M[SNR]=$P_SEARCH_SDIR[SNR]	; 输出旋转方向、定位或进给轴模式。
SNR=SNR+1	; 下一主轴。
REPEAT AUSG_SPI P=$P_NUM_SPINDLES-1	; 针对所有主轴。
MSG("")	
REPOSA	
RET	

示例说明

若主轴数已知，可将相同类型的输出写入至一个零件程序段，从而减少程序处理时间。

$P_SEARCH_SDIR 的输出须在单独的零件程序段中进行，因为主轴定位或切换至进给轴模式与齿轮档切换一起时可能会触发报警。

若通过 REPOSA 使启动的 ASUB 结束，那么主轴 1 将留在 111 度的位置，而主轴 2 则将重新定位至 77 度的位置。

如需实现另一种特性，则须特别调整用于程序段搜索的程序序列，例如“N05 M3 S...”和“N30 SPOS[2]=IC(...)”。

可在 ASUB 中通过系统变量 $P_SEARCH 确定程序段搜索是否生效：
$P_SEARCH==1；程序段搜索生效

在转速控制运行后的增量定位中，虽然目标行程已定义，但是某些情形下要在定位过程中才能确定达到的最终位置。例如，在越过零标记时进行位置校准，或启用位置闭环控制时均属于此类情形。因此，从零位置开始的行程被用作 REPOS 位置（ASUB 中为 REPOSA）。

边界条件

收集的 S 值

零件程序中 S 值的含义取决于当前生效的进给率类型：

G93、G94、G95、G97、G971：	S 值为转速
G96、G961：	S 值为恒定切削速度

输出系统变量 $P_SEARCH_S 前，若进给率类型发生变化（例如用于换刀），必须将进给率类型恢复为零件程序目标程序段中的原始设置，以避免采用错误的进给率类型。

收集的旋转方向

输出旋转方向时，程序段搜索开始的时间点系统变量 $P_SEARCH_SDIR 的预赋值为“-5”。该数值在输出时无效。

因此，在对没有主轴编写主轴旋转方向、主轴定位或进给轴模式的的程序段进行搜索时，能保持最后生效的主轴运行模式。

M19、SPOS 和 SPOSA 编程在系统变量 $P_SEARCH_SDIR 中作为“M-19”（内部 M19）收集，以替代 M3、M4、M5 和 M70。

输出“M-19”时，系统会从系统变量 $P_SEARCH_SPOS 和 $P_SEACH_SPOSMODE 读取定位数据。这两个系统变量均可写入，例如用于进行补偿。

说明

对于上文介绍的指令（例如 M[<n>] = $P_SEARCH_SDIR[<n>]），值“–5”和“19”通常为用户不可见，仅在 ASUB 中对系统变量进行特殊分析时才需查看。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Chan>.basic.in.blockSearchLastActionActive	LBP_Chan*.E_LastActBlock	DB21, ...DBX32.6

13.13.6 使用程序段搜索类型 5（SERUPRO）时的辅助功能输出

输出特性

使用程序段搜索类型 5（SERUPRO）时，辅助功能可在程序段搜索期间输出至 PLC，并/或按分组被收集到以下系统变量中：

- $AC_AUXFU_PREDEF_INDEX[<n>]（预定义辅助功能的下标）
- $AC_AUXFU_TYPE[<n>]（辅助功能的类型）
- $AC_AUXFU_STATE[<n>]（辅助功能的输出状态）
- $AC_AUXFU_EXT[<n>]（辅助功能的地址扩展）
- $AC_AUXFU_VALUE[<n>]（辅助功能的数值）

有关系统变量的描述请见 "查询系统变量 (页 848)" 一章。

输出控制

通过配置以下机床数据的位 9 和位 10，可定义是否在程序段搜索类型 5（SERUPRO）期间将辅助功能输出至 PLC，按分组被收集到上述系统变量中：

- MD22080 $MC_AUXFU_PREDEF_SPEC[<n>]
  （预定义辅助功能的输出特性）
  其中 <n> = 系统功能下标（0 ... 32）
- MD22035 $MC_AUXFU_ASSIGN_SPEC[<n>]
  （用户自定义辅助功能的输出特性）
  其中 <n> = 辅助功能下标（0 ... 254）
- MD11110 $MN_AUXFU_GROUP_SPEC[<n>]
  （组中辅助功能的输出特性）
  其中 <n> = 组下标（0 ... 63）

位	值	含义
9	0	在程序段搜索类型 5（SERUPRO）期间不收集
	1	在程序段搜索类型 5（SERUPRO）期间收集
10	0	在程序段搜索类型 5（SERUPRO）期间输出
	1	在程序段搜索类型 5（SERUPRO）期间不输出

输出计数器

用户可逐个通道地在程序段搜索 ASUB 中将收集的辅助功能输出至 PLC。为了实现多个通道的连续输出，每次输出辅助功能时三个输出计数器均会跨通道变化：

系统变量	含义		
$AC_AUXFU_TICK[<n>,<m>]	生效辅助功能的输出计数器		
	下标	含义	
	<n>	组下标（0 ... 63）	
	<m>	输出计数器（0 ... 2）	
		值	含义
		0	输出序列计数器（所有输出在一个插补周期内进行）
		1	插补周期中一个输出序列内的包计数器
		2	一个包内的辅助功能计数器

说明

- 一个辅助功能包可包含最多 10 个辅助功能。
- SERUPRO 期间每个通道在每个插补周期中可执行两个包，因为同步动作按该周期处理。
- 在一个插补周期内可跨所有通道执行一个最多含 20 个包的输出序列（每个通道 2 个包 * 10 通道）。

通过此编码可显示在同一个插补周期内执行了多少个辅助功能包和多少个辅助功能：

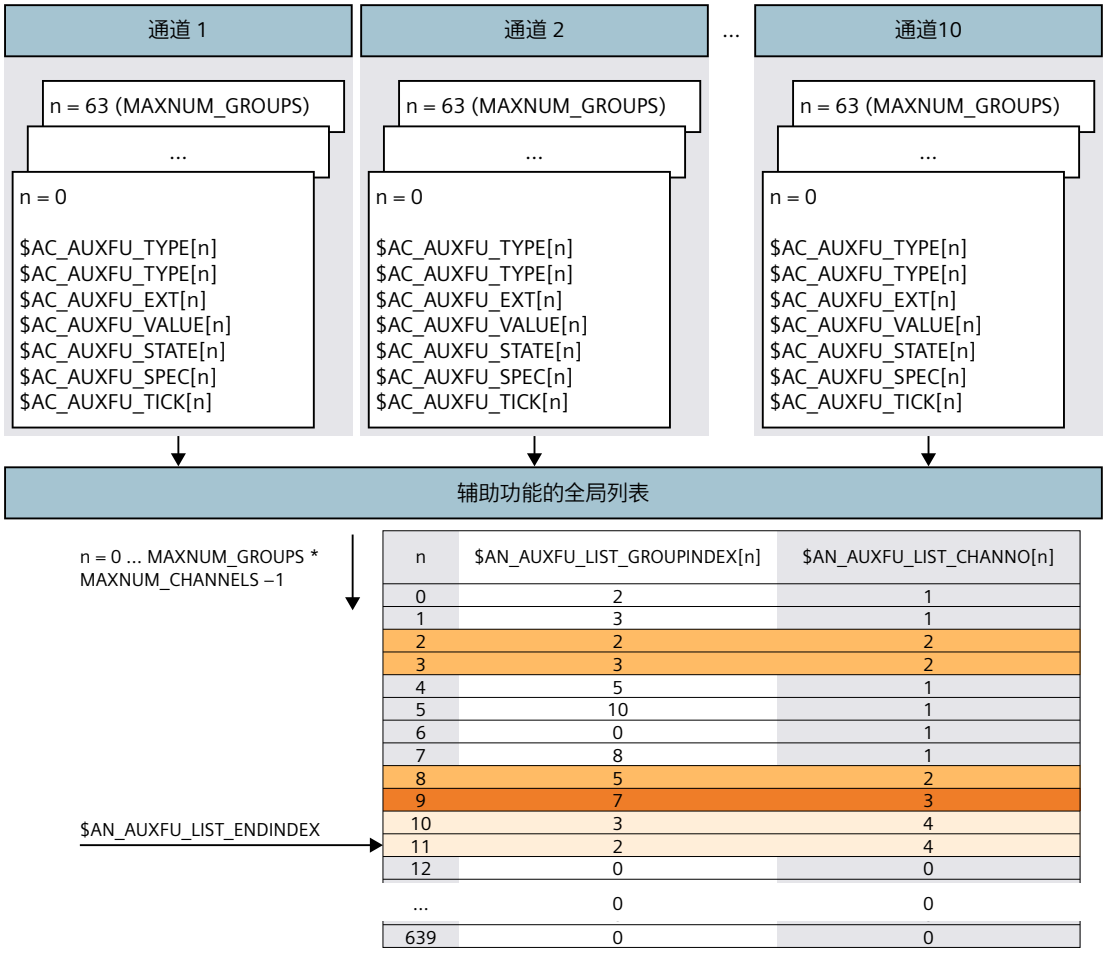
- 在一个插补周期内收集的辅助功能均有相同的序列计数器。
- 在一个包（程序段或同步动作）内收集的辅助功能均有相同的包计数器。

每收集到一个辅助功能，辅助功能计数器依次递增。

辅助功能的全局列表

SERUPRO 结束时，各通道中针对组收集的辅助功能会依据其计数器状态记录至一个跨通道列表中，并附注通道编号和组下标。

系统变量 1)	含义
$AC_AUXFU_TICK[<n>,<m>]	计数器状态
$AN_AUXFU_LIST_CHANNO[<n>] 2)	通道号
$AN_AUXFU_LIST_GROUPINDEX[<n>] 2)	组下标
1) 下标 <n> 的取值范围：0 ... MAXNUM_GROUPS * MAXNUM_CHANNELS - 1 2) 系统变量均可读写。	



找到搜索目标后，系统便会建立全局列表。此列表作为对下一个 SERUPRO 末尾 ASUB 中待输出辅助功能的系统建议。若辅助功能不需要输出，那么必须将对应的组下标设置为“0”。

主轴辅助功能相关特性

程序段搜索启动后，所有通道都会在通道变量中针对组收集辅助功能。为了在 SERUPRO 目标程序段中通过收集的辅助功能尽可能恢复主轴状态，任意主轴辅助功能组中最后生效的辅助功能必须能代表目标程序段中的主轴状态。在主轴状态过渡时，失效的辅助功能会从列表中清除，或者会在必要时输入隐性辅助功能。

全局辅助功能列表的所有主轴辅助功能必须在目标程序段中对应所达到的主轴状态，确保列表输出时这些辅助功能可执行，且不会引起报警或不期望的主轴状态使零件程序的继续执行受到影响。

相应地，对于系统中配置的每根主轴的辅助功能组，其主轴编号须对应辅助功能的地址扩展。

组 a： M3、M4、M5、M19、M70

组 b： M40、M41、M42、M43、M44、M45

组 c： S

清除失效的辅助功能

使用以下功能时，系统会为相关主轴清除组 a 的辅助功能：

- 对跟随主轴激活通用耦合时，如 COUPON、TRAILON、EGON 等

生成隐性组 a 辅助功能

使用以下功能时，系统会为相关主轴隐性生成组 a 的辅助功能：

- 对跟随主轴取消同步主轴耦合时
  - 根据耦合情形，COUPOF 会在主处理中生成 M3、M4、S 或 M5。
  - COUPOF(S<n>, S<m>, POS) 和 COUPOFS(S<n>, S<m>, POS, POS) 会生成 M3、M4 和 S。
  - COUPOFS 会在主处理中生成 M5。
  - COUPOFS(S<n>, S<m>, POS) 会在主处理中生成 M19。
    隐性 M19（ASUB 中"SPOS[<地址扩展>] = IC(0)"）会激活无运行的定位模式。
- 在主轴作为进给轴运行，或者通过选择转换（其中主轴作为进给轴）过渡至进给轴模式时，系统会生成 M70。
- 使用 SPCOF 时会生成 M5。

跨通道辅助功能

在程序段搜索类型 5（SERUPRO）中，辅助功能也可跨通道在全局辅助功能列表中收集。此时只会将该组最后收集的辅助功能（计数最高）记录至全局列表。

通过下列机床数据进行相应的配置：

- MD22080 $MC_AUXFU_PREDEF_SPEC[<n>]，位 11
  （预定义辅助功能的输出特性）
  其中 <n> = 系统功能下标
- MD22035 $MC_AUXFU_ASSIGN_SPEC[<n>]，位 11
  （用户自定义辅助功能的输出特性）
  其中 <n> = 辅助功能下标
- MD11110 $MN_AUXFU_GROUP_SPEC[<n>]，位 11
  （组中辅助功能的输出特性）
  其中 <n> = 组下标

位	值	含义
11	0	针对特定通道收集
	1	跨通道收集

根据主轴状态，主轴辅助功能会在程序段搜索结束时提前滤除。通道数据会相应更新。全局辅助功能列表可在 SERUPRO 末尾 ASUB 中按顺序执行，经过排序的辅助功能可同步于通道输出。

查询最后收集的辅助功能

通过系统变量 $AN_AUXFU_LIST_ENDINDEX 可查询全局列表中最后收集的辅助功能的下标。

13.13.7 SERUPRO 末尾 ASUB

功能

带程序测试的程序段搜索（SERUPRO）完成后，在下一步处理开始前必须将程序段搜索中收集的辅助功能输出。为此，程序段搜索期间系统会在一个全局列表中收集辅助功能。
SERUPRO 末尾的 ASUB 从此列表针对通道生成对应的零件程序段。这样可确保收集的辅助功能针对特定通道或跨通道输出时的时间顺序正确。完整功能的 SERUPRO 末尾 ASUB 是 NC 软件的组成部分。

SERUPRO 末尾 ASUB 可由用户/机床制造商修改。下面介绍的功能用于支持对辅助功能全局列表的处理，以及生成同步辅助功能输出所需的零件程序段。

AUXFUSYNC(...) 功能

功能：

AUXFUSYNC 功能能够在每次调用时从辅助功能全局列表以字符串形式生成一个完整的零件程序段。该零件程序段包含辅助功能，或者用于对辅助功能输出进行同步的指令（WAITM、G4 等）。

此功能会触发预处理停止。

句法：

```
PROC AUXFUSYNC(VAR INT <NUM>, VAR INT <GROUPINDEX>[10], VAR
STRING[400] <ASSEMBLED>)
```

参数：

<NUM>： <ASSEMBLED> 参数中提供的零件程序段的信息，或程序段中包含的辅助功能的信息。

取值范围：-1, 0, 1 ... 10

值	含义
≥1	零件程序段中包含的辅助功能的数量
0	无辅助功能的零件程序段，例如 WAITM、G4
-1	结束标识。针对当前通道，辅助功能全局列表已完全处理。

<GROUPINDEX>： 零件程序段中包含的辅助功能组的下标。其中下标 = 辅助功能组编号 - 1

<ASSEMBLED>： 字符串形式的完整零件程序段，用于针对特定通道的 SERUPRO 结束处 ASUB。

更多详细信息：

若辅助功能通过一个同步动作收集，则会生成两个 NC 程序段。一个 NC 程序段用于输出辅助功能。另一个可执行 NC 程序段，用于将输出辅助功能的 NC 程序段传输至主处理：

1. 通过同步动作输出辅助功能，例如：WHEN TRUE DO M100 M102
2. 可执行 NC 程序段，例如：G4 F0.001

AUXFUDEL(...) 功能

功能：

AUXFUDEL 功能可为调用通道将指定的辅助功能从全局列表删除。删除通过将对应的组索引 ...GROUPINDEX[n] 置 0 来实现。

此功能必须在调用 AUXFUSYNC 前调用。

此功能会触发预处理停止。

句法：

```
PROC AUXFUDEL(CHAR <TYPE>, INT <EXTENSION>, REAL <VALUE>, INT
<GROUP>)
```

参数：

<TYPE>:	待删除的辅助功能的类型
<EXTENSION>:	待删除的辅助功能的地址扩展
<VALUE>:	待删除的辅助功能的数值
<GROUP>:	辅助功能组的编号

AUXFUDELG(...) 功能

功能：

AUXFUDELG 功能可为调用通道将指定辅助功能组中的所有辅助功能从全局列表删除。删除通过将对应的组索引 ...GROUPINDEX[n] 置 0 来实现。

此功能必须在调用 AUXFUSYNC 前调用。

此功能会触发预处理停止。

句法：

```
PROC AUXFUDELG(INT <GROUP>)
```

参数：

<GROUP>:	辅助功能组的编号

多通道程序段搜索

注意
多通道程序段搜索和 AUXFUDEL / AUXFUDELG 在进行多通道程序段搜索时，若在 SERUPRO 末尾 ASUB 中通过 AUXFUDEL / AUXFUDELG 将辅助功能从全局列表中删除，那么在调用 AUXFUSYNC 功能前必须对所涉及的通道进行同步。通过同步可确保调用 AUXFUSYNC 功能前所有删除任务均被处理，并确保列表的一致性。

示例

以下为两个建立用户专用 SERUPRO 末尾 ASUB 的示例。

示例 1：删除辅助功能，并通过 AUXFUSYNC(...) 生成辅助功能输出

程序代码	注释
N10 DEF STRING[400] ASSEMBLED=""	
N20 DEF STRING[31] FILENAME="/_N_CST_DIR/_N_AUXFU_SPF"	
N30 DEF INT GROUPINDEX[10]	
N40 DEF INT NUM	
N60 DEF INT ERROR	
N90	
N140 AUXFUDEL("M",2,3,5)	; 删除 M2=3（第 5 辅助功能组）
N150	
N170 AUXFUDELG(6)	; 删除收集的
	; 删除第 6 组。
N180	
N190 IF ISFILE(FILENAME)	
N210 DELETE(ERROR,FILENAME)	; 删除 FILENAME 文件
N220 IF (ERROR<>0)	; 错误分析
N230 SETAL(61000+ERROR)	
N240 ENDIF	
N250 ENDIF	
; 注意!	
; 在多通道程序段搜索时，若通过 AUXFUDEL/AUXFUDELG	
; 从全局列表删除辅助功能，那么必须在	
; 通过 AUXFUSYNC 生成子程序 FILENAME 的 LOOP 前进行通道同步，	
; 通过同步可确保所有通道中的所有删除任务	
; 均被处理，并确保列表的一致性。	
; 示例：WAITM(99,1,2,3)	
N270 LOOP	
N300 AUXFUSYNC(NUM,GROUPINDEX,ASSEMBLED)	; 生成零件程序段
N310	
N320 IF (NUM==-1)	; 通道的所有辅助功能
	; 均已执行。
N340 GOTOF LABEL1	
N350 ENDIF	
N380 WRITE(ERROR,FILENAME,ASSEMBLED)	; 将零件程序段写入 FILENAME 文件。
N390 IF (ERROR<>0)	; 错误分析
N400 SETAL(61000+ERROR)	
N410 ENDIF	
N430 ENDLOOP	
N440	
N450 LABEL1:	
N460	

程序代码	注释
N480 CALL FILENAME	; 执行生成的子程序。
N490	
N510 DELETE(ERROR,FILENAME)	; 执行后再次删除文件。
N520 IF (ERROR<>0)	
N530 SETAL(61000+ERROR)	
N540 ENDIF	
N550	
N560 M17	

示例 2：删除辅助功能，不通过 AUXFUSYNC(...) 生成辅助功能输出

程序代码	注释
N0610 DEF STRING[400] ASSEMBLED=""	
N0620 DEF STRING[31] FILENAME="/_N_CST_DIR/_N_AUXFU_SPF"	
N0630 DEF INT GROUPINDEX[10]	
N0640 DEF INT NUM	
N0650 DEF INT LAUF	
N0660 DEF INT ERROR	
N0670 DEF BOOL ISQUICK	
N0680 DEF BOOL ISSYNACT	
N0690 DEF BOOL ISIMPL	
...	
N0760 AUXFUDEL("M",2,3,5)	; 删除 M2=3（第 5 辅助功能组）
N0770	
N0790 AUXFUDELG(6)	; 删除收集的 ; 删除第 6 组。
N0800	
N0810 IF ISFILE(FILENAME)	
N0830 DELETE(ERROR,FILENAME)	; 文件已存在，必须 ; 删除。
N0840 IF (ERROR<>0)	
N0850 SETAL(61000+ERROR)	
N0860 ENDIF	
N0870 ENDIF	
N0880	
; 注意!	
; 在多通道程序段搜索时，若通过 AUXFUDEL/AUXFUDELG	
; 从全局列表删除辅助功能，那么必须在	
; 通过 AUXFUSYNC 生成子程序 FILENAME 的 LOOP 前进行通道同步，	
; 通过同步可确保所有通道中的所有删除任务	
; 均被处理，并确保列表的一致性。	
; 示例：WAITM(99,1,2,3)	
N0890 LOOP	

程序代码	注释
N0920 AUXFUSYNC(NUM,GROUPINDEX,ASSEMBLED)	; 从全局辅助功能列表
	; 生成辅助功能程序段
	; 的步骤。
N0930	
N0940 IF (NUM==-1)	; 通道的所有辅助功能
	; 均已执行。
N0960 GOTOF LABEL1	
N0970 ENDIF	
N0980	
N1000 IF (NUM>0)	; 输出辅助功能时
	; 生成程序段。
N1010	
N1020 ASSEMBLED=""	
N1030	
N1050 FOR LAUF=0 TO NUM-1	; 对一个程序段收集的
	; 辅助功能。
N1060	
N1080 IF GROUPINDEX[LAUF]<>0	; 从全局列表删除的
	; 辅助功能的组下标为 0。
N1090	
N1100 ISQUICK=$AC_AUXFU_SPEC[GROUPINDEX[LAUF]] BAND'H2'	
N1110	
N1120 ISSYNACT=$AC_AUXFU_SPEC[GROUPINDEX[LAUF]] BAND'H1000'	
N1130	
N1140 ISIMPL=$AC_AUXFU_SPEC[GROUPINDEX[LAUF]] BAND'H2000'	
N1150	
N1180 IF ISSYNACT	; 编写用于输出 M 辅助功能
	; 的程序段
N1190 ASSEMBLED= ASSEMBLED << "WHEN TRUE DO "	
N1200 ENDIF	
N1210 ; 隐性生成的 M19 映射至 SPOS[SPI(<主轴编号>)] = IC(0) 。	
N1230 IF (ISIMPL AND ($AC_AUXFU_VALUE[GROUPINDEX[LAUF]==19))	
N1240 ASSEMBLED= ASSEMBLED << "SPOS[SPI(" <<	
$AC_AUXFU_EXT[GROUPINDEX[LAUF]] << ")=IC(0)"	
N1260 ELSE	
N1270 ASSEMBLED= ASSEMBLED << "M[" << $AC_AUXFU_EXT[GROUPINDEX[LAUF]] << "]="	
N1280	
N1290 IF ISQUICK	
N1300 ASSEMBLED= ASSEMBLED << "QU("	
N1310 ENDIF	
N1320	
N1330 ASSEMBLED= ASSEMBLED << $AC_AUXFU_VALUE[GROUPINDEX[LAUF]]	
N1340	
N1350 IF ISQUICK	

程序代码	注释
N1360 ASSEMBLED= ASSEMBLED << ")"	
N1370 ENDIF	
N1380 ENDIF	
N1400 ENDIF	
N1420 ENDFOR	
N1430	
N1450 WRITE(ERROR,FILENAME,ASSEMBLED)	; 将辅助功能程序段写入文件。
N1460	
N1470 IF ISSYNACT	
N1480 ASSEMBLED="G4 F0.001"	
N1490 WRITE(ERROR,FILENAME,ASSEMBLED)	
N1500 ENDIF	
N1510	
N1520 ELSE	
N1540 WRITE(ERROR,FILENAME,ASSEMBLED)	; 将辅助功能程序段写入文件。
N1550 ENDIF	
N1560	
N1570 ENDLOOP	
N1580	
N1590 LABEL1:	
N1600	
N1620 CALL FILENAME	; 执行生成的子程序。
N1630	
N1650 DELETE(ERROR,FILENAME)	; 执行后再次删除文件。
N1660 IF (ERROR<>0)	
N1670 SETAL(61000+ERROR)	
N1680 ENDIF	
N1690	
N1700 M17	

13.14 隐性输出的辅助功能

功能

隐性输出的辅助功能是指未显性编写，而通过其他系统功能（例如转换选择、刀具选择等）输出的辅助功能。此类隐性辅助功能不会引起系统功能；根据为其参数设置的输出特性，系统会收集 M 代码并/或输出至 PLC。

参数设置

隐性输出的辅助功能的 M 代码通过以下机床数据确定：

- MD22530 $MC_TOCARR_CHANGE_M_CODE（刀架切换时的 M 代码）
  此机床数据的值指示的是激活刀架时 NC/PLC 接口上输出的 M 代码的编号。
  若该值为正值，那么输出的是未经变化的 M 代码。
  若改值为负值，则会将刀架编号累加至机床数据的绝对值，并输出此编号。
- MD22532 $MC_GEOAX_CHANGE_M_CODE（几何轴切换时的 M 代码）
  几何轴切换时，NC/PLC 接口上输出的 M 代码的编号。
- MD22534 $MC_TRAFO_CHANGE_M_CODE（转换切换时的 M 代码）
  几何轴的转换切换时，NC/PLC 接口上输出的 M 代码的编号。

说明

若待输出的 M 代码的编号或 MD22530 / MD22532 / MD22534 的值本身为 0 至 6、17 或 30，则不输出 M 代码。对于所生成的 M 代码是否会与其它功能冲突，系统不进行监控。

输出特性

对于隐性输出的辅助功能，机床数据 MD22080 或 MD22035（预定义或用户自定义辅助功能的输出特性）的位 13 置位。

可通过系统变量 $AC_AUXFU_SPEC[<n>] 查询此位。

隐性输出的辅助功能 M19

为了使 M19 和 SPOS/SPOSA 在 NC/PLC 接口上的特性方面保持一致，可在使用 SPOS 和 SPOSA 时将辅助功能 M19 输出至 NC/PLC 接口（参见功能手册之*轴和主轴*，章节“*主轴*”）。

程序段搜索时会收集隐性输出的辅助功能 M19。

13.15 获取信息

辅助功能的相关信息（例如输出状态）可通过下列方式获取：

- 操作界面上针对组的模态 M 辅助功能显示。
- 在零件程序和同步动作中查询系统变量。

13.15.1 针对组的模态 M 辅助功能显示

功能

在操作界面上可针对组显示 M 辅助功能的输出状态和应答状态。

前提条件

对 M 辅助功能实现以功能为取向的应答和显示的前提条件是：对辅助功能的管理在 PLC 中进行，也就是在用户程序中进行。因此 PLC 编程人员需负责对这些辅助功能进行应答。其必须了解需要应答的辅助功能在哪些组中。

标准

不通过 PLC 管理的 M 辅助功能由 NC 输出至 PLC，并标记为“已传输”。对此类辅助功能无功能应答。此外程序段搜索后收集的所有 M 辅助功能均会被显示，从而告知操作人员启动后会输出哪些辅助功能。

PLC 相关事项

对于由 PLC 自身管理的辅助功能组，**接收**和**功能结束**时 PLC 用户程序必须对这些组的所有辅助功能进行应答。PLC 编程人员必须知晓这些组的所有辅助功能。

其它

M 辅助功能只**针对组**显示。此外也保留逐段显示。最多可显示 15 个组，其中每个组总是显示该组**最后收集或输出至 PLC 的 M 功能**。根据其状态，M 功能会以多种方式显示：

状态	显示方式
辅助功能已被收集	反色，黄色字体
辅助功能已由 NC 输出至 PLC	反色
辅助功能已由 NC 传输至 PLC，并进行传输应答	黑色字体，灰色背景
辅助功能由 PLC 管理，且已由 PLC 直接接收。	黑色字体，灰色背景
辅助功能由 PLC 管理，且已进行功能应答。	黑色字体，灰色背景

显示更新

显示总是先输出收集到的辅助功能，其次是由 PLC 管理和由 NC 管理的辅助功能。收集到的辅助功能始终标记为“已收集”，直至 NC 将其输出至 PLC。PLC 管理的辅助功能一直保留，直至其被其他辅助功能取代。复位时只会删除收集的和 NC 管理的辅助功能。

13.15.2 查询系统变量

功能

在零件程序和同步动作中，可通过系统变量针对组查询辅助功能：

$AC_AUXFU_... [<n>] = <数值>

系统变量	含义		
$AC_AUXFU_PREDEF_INDEX[<n>]	<值>：	一个辅助功能组中收集（程序段搜索）或输出的最后一个预定义辅助功能的下标	
		类型：	INT
		若指定组中尚未输出辅助功能，或者辅助功能为用户自定义辅助功能，那么该变量会输出“-1”值。	
	<n>：	组下标（0 ... 63）	
	提示： 通过此系统变量可确定唯一的预定义辅助功能。		
$AC_AUXFU_TYPE[<n>]	<值>：	一个辅助功能组中收集（程序段搜索）或输出的最后一个辅助功能的类型	
		类型：	CHAR
	<n>：	组下标（0 ... 63）	
$AC_AUXFU_EXT[<n>] 或者 M 功能专用： $AC_AUXFU_M_EXT[<n>]	<值>：	一个辅助功能组中收集（程序段搜索）或输出的最后一个辅助功能的地址扩展	
		类型：	INT
	<n>：	组下标（0 ... 63）	

系统变量	含义		
$AC_AUXFU_VALUE[<n>] 或者 M 功能专用： $AC_AUXFU_M_VALUE[<n>]	<值>：	一个辅助功能组中收集（程序段搜索）或输出的最后一个辅助功能的数值	
		类型：	REAL
	<n>：	组下标（0 ... 63）	
$AC_AUXFU_SPEC[<n>]	值：	一个辅助功能组中收集（程序段搜索）或输出的最后一个辅助功能的位编码输出特性，对应 MD22080/MD22035（或 QU 编程）	
		类型：	INT
	<n>：	组下标（0 ... 63）	
	提示： 通过此变量可确定是否进行采用快速应答的辅助功能输出。		
$AC_AUXFU_STATE[<n>] 或者 M 功能专用： $AC_AUXFU_M_STATE[<n>]	<值>：	一个辅助功能组中收集（程序段搜索）或输出的最后一个辅助功能的输出状态	
		类型：	INT
		取值范围：	0 ... 5
		0：	辅助功能不存在
		1：	M 辅助功能已通过程序段搜索收集
		2：	M 辅助功能已输出至 PLC
		3：	M 辅助功能已输出至 PLC，且已进行传输应答
		4：	M 辅助功能由 PLC 管理，且已由 PLC 接收
		5：	M 辅助功能由 PLC 管理，且已进行功能应答
	<n>：	组下标（0 ... 63）	

示例

需要按照输出顺序对第 1 组的所有 M 辅助功能进行存储：

```
id=1 every $AC_AUXFU_M_STATE[0]==2 do
$AC_FIFO[0,0]=$AC_AUXFU_M_VALUE[0]
```

更多信息

参数手册之系统变量

13.16 前提条件

13.16.1 一般前提条件

跨通道取主轴

对辅助功能的参数设置针对通道进行，因此在使用“跨通道取主轴”功能时，所有使用主轴的通道中主轴专用辅助功能的设置必须相同。

刀具管理

刀具管理生效时，以下前提条件适用：

- T 功能和 M<k> 功能不输出至 PLC。
  提示
  k 是参数设置的换刀辅助功能的值（缺省设置：6）：
  MD22560 $MC_TOOL_CHANGE_M_CODE（换刀辅助功能）
- 若未编写地址扩展，那么辅助功能基于通道的主主轴或主刀架。
  定义主主轴：
  - MD20090 $MC_SPIND_DEF_MASTER_SPIND
  - 零件程序指令：SETMS

  定义主刀架：
  - MD20124 $MC_TOOL_MANAGEMENT_TOOLHOLDER
  - 零件程序指令：SETMTH

每个零件程序段中辅助功能的最大数量

一个零件程序段中最多可以编写 10 个辅助功能。

DL（总补偿）

对 DL 功能有以下限制：

- 每个零件程序段中只可编写一个 DL 功能。
- 在同步动作中使用 DL 功能时，“值”参数不输出至 PLC。

13.16.2 输出特性

螺纹切削

螺纹切削 G33、G34 和 G35 生效时，主轴专用辅助功能

- M3（主轴顺时针旋转）
- M4（主轴逆时针旋转）

总是采用以下输出特性：

- 输出持续一个 OB40 周期（快速应答）
- 运行中输出

主轴专用辅助功能 M5（主轴停止）总是在程序段末尾输出。 也就是说，包含 M5 的零件程序段总是以准停结束，即便是在连续路径运行生效的情况下。

同步动作

在从同步动作输出辅助功能时，除以下参数外，设置的输出特性均忽略：

- 位 0：输出持续一个 OB1 周期（普通应答）
- 位 1：输出持续一个 OB40 周期（快速应答）

辅助功能：M17 或 M2 / M30（子程序结束）

编写在单独的零件程序段中

若辅助功能 M17、M2 或 M30 被编写在单独的程序段中，且尚有轴在运动，那么在轴停止后辅助功能才会输出至 PLC。

参数设置的输出特性的叠加

为辅助功能 M17 或 M2/M30 设置的输出特性会与以下机床数据中定义的输出特性叠加：

MD20800 $MC_SPF_END_TO_VDI，位 0（子程序结束/停止输出至 PLC）

位	值	含义
0	0	辅助功能 M17 或 M2/M30（子程序结束）不输出至 PLC。 子程序结束处连续路径运行不中断。
	1	辅助功能 M17 或 M2/M30（子程序结束）输出至 PLC。

辅助功能：M1（有条件停止）

参数设置的输出特性的叠加

为辅助功能 M1 设置的输出特性会与以下机床数据中定义的输出特性叠加：

MD20800 $MC_SPF_END_TO_VDI，位 1（子程序结束/停止输出至 PLC）

位	值	含义
1	0	辅助功能 M01（有条件停止）总是输出至 PLC。 快速应答无效，因为 M01 固定归属于第 1 辅助功能组并始终在程序段末尾输出。
	1	仅在通过 HMI 操作界面激活了“编程的停止”时，辅助功能 M01（有条件停止）才输出至 PLC。 采用快速应答时，M1 在运行中输出至 PLC。 只要此功能不生效，连续路径运行便不会中断。

不含运行的零件程序段

在不含运行的零件程序段中，所有辅助功能均立即在一个块中输出，不考虑参数设置的输出特性。

主轴专用辅助功能输出只作为提供给 PLC 用户程序的信息

在特定控制情形下，例如为了完成程序段搜索，系统会将收集到的主轴专用辅助功能（例如 M3、M4、M5、M19、M40...M45、M70）输出至 NC/PLC 接口，而只是作为提供给 PLC 用户程序的信息。 为此控制系统会生成一个零件程序段（动作程序段），其中记录收集的辅助功能，其地址扩展**取负值**。 这样一来便**不会**执行对应的系统功能。

示例：M(-2) = 41，为第 2 主轴请求齿轮档切换

13.17 示例

13.17.1 预定义辅助功能的扩展

任务

为通道的第 2 主轴设置辅助功能 M3、M4 和 M5。

参数设置：M3

要求：

- 机床数据下标：0（第一个用户自定义辅助功能）
- 辅助功能组：5
- 类型和数值：M3（主轴顺时针旋转）
- 地址扩展：2，对应通道第 2 主轴
- 输出特性：
  - 输出持续一个 OB1 周期（普通应答）
  - 运行前输出

参数设置：

```
MD22000 $MC_AUXFU_ASSIGN_GROUP [ 0 ]          = 5
MD22010 $MC_AUXFU_ASSIGN_TYPE [ 0 ]           = "M"
MD22020 $MC_AUXFU_ASSIGN_EXTENSION [ 0 ]      = 2
MD22030 $MC_AUXFU_ASSIGN_VALUE [ 0 ]          = 3
MD22035 $MC_AUXFU_ASSIGN_SPEC [ 0 ]           = 'H21'
```

参数设置：M4

要求：

- 机床数据下标：1（第二个用户自定义辅助功能）
- 辅助功能组：5
- 类型和数值：M4（主轴逆时针旋转）

- 地址扩展：2，对应通道第 2 主轴
- 输出特性：
  - 输出持续一个 OB1 周期（普通应答）
  - 应答后进行主轴响应
  - 运行中输出

参数设置：

```
MD22000 $MC_AUXFU_ASSIGN_GROUP [ 1 ]          = 5
MD22010 $MC_AUXFU_ASSIGN_TYPE [ 1 ]           = "M"
MD22020 $MC_AUXFU_ASSIGN_EXTENSION [ 1 ]      = 2
MD22030 $MC_AUXFU_ASSIGN_VALUE [ 1 ]          = 4
MD22035 $MC_AUXFU_ASSIGN_SPEC [ 1 ]           = 'H51'
```

参数设置：M5

要求：

- 机床数据下标：2（第三个用户自定义辅助功能）
- 辅助功能组：5
- 类型和数值：M5（主轴停止）
- 地址扩展：2，对应通道第 2 主轴
- 输出特性：
  - 输出持续一个 OB1 周期（普通应答）
  - 应答后进行主轴响应
  - 在程序段末尾输出

参数设置：

```
MD22000 $MC_AUXFU_ASSIGN_GROUP [ 2 ]          = 5
MD22010 $MC_AUXFU_ASSIGN_TYPE [ 2 ]           = "M"
MD22020 $MC_AUXFU_ASSIGN_EXTENSION [ 2 ]      = 2
MD22030 $MC_AUXFU_ASSIGN_VALUE [ 2 ]          = 5
MD22035 $MC_AUXFU_ASSIGN_SPEC [ 2 ]           = 'H91'
```

13.17.2 定义辅助功能

任务

针对采用以下配置的机床，对辅助功能专用机床数据进行参数设置：

主轴

- 主轴 1：主主轴
- 主轴 2：第二主轴

齿轮档

- 主轴 1：5 个齿轮档
- 主轴 2：无齿轮档

用于冷却水 ON/OFF 的开关功能

- 主轴 1
  - “ON”= M50
  - “OFF”= M51
- 主轴 2
  - “ON”= M52
  - “OFF”= M53

要求

主轴 1（主主轴）

说明

默认指定关系

- 主轴 1（主主轴）的辅助功能 M3、M4、M5、M70 和 M1=3、M1=4、M1=5、M1=70 默认指定给第 2 辅助功能组。
- 主轴 1（主主轴）的所有 S 值和 S1 值默认指定给第 3 辅助功能组。

- 程序段搜索后输出最后编写的齿轮档。为此将下列辅助功能指定给第 9 辅助功能组：
  - M40、M41、M42、M43、M44、M45
  - M1=40、M1=41、M1=42、M1=43、M1=44、M1=45
- 辅助功能 M3、M4、M5、M70 和 M1=3、M1=4、M1=5、M1=70 （第 2 辅助功能组）及 S 值和 S1 值（第 3 辅助功能组）采用以下输出特性：
  - 输出持续一个 OB40 周期（快速应答）
  - 运行前输出
- 用于齿轮档切换的辅助功能 M40 至 M45，M1=40 至 M1=45（第 9 辅助功能组）采用以下输出特性：
  - 输出持续一个 OB1 周期（普通应答）
  - 运行前输出

主轴 2

- 一个程序段中只允许编写一个用于切换旋转方向的 M 功能。程序段搜索后输出最后编写的旋转方向。为此将下列辅助功能指定给第 10 辅助功能组：
  - M2=3、M2=4、M2=5、M2=70
- 将所有 S2 值指定给第 11 辅助功能组。
- 辅助功能 M2=3、M2=4、M2=5、M2=70（第 10 辅助功能组）及 S2 值（第 11 辅助功能组）采用以下输出特性：
  - 输出持续一个 OB40 周期（快速应答）
  - 运行前输出

冷却水

- 不允许在一个零件程序段中接通和关闭。程序段搜索后将冷却水接通或关闭。为此将以下辅助功能指定给第 12 和第 13 辅助功能组：
  - 12.辅助功能组：M50, M51
  - 13.辅助功能组：M52, M53
- 辅助功能 M50、M51（第 12 辅助功能组）和 M52、M53（第 13 辅助功能组）采用以下输出特性：
  - 输出持续一个 OB1 周期（普通应答）
  - 运行前输出

机床数据的参数设置

对机床数据的参数设置通过零件程序段中的相应编程进行：

程序代码	注释
$MN_AUXFU_MAXNUM_GROUP_ASSIGN=21	; 每个通道用户自定义辅助功能的数量
$MN_AUXFU_GROUP_SPEC[1]='H22'	; 第 2 辅助功能组的输出特性
$MN_AUXFU_GROUP_SPEC[2]='H22'	; 第 3 辅助功能组的输出特性
$MN_AUXFU_GROUP_SPEC[8]='H21'	; 第 9 辅助功能组的输出特性
$MC_AUXFU_ASSIGN_TYPE[0]="M"	; 对辅助功能 1 的描述：M40
$MC_AUXFU_ASSIGN_EXTENSION[0]=0	
$MC_AUXFU_ASSIGN_VALUE[0]=40	
$MC_AUXFU_ASSIGN_GROUP[0]=9	
	; ...（辅助功能 2 至 5 以此类推）
$MC_AUXFU_ASSIGN_TYPE[5]="M"	; 对辅助功能 6 的描述：M45
$MC_AUXFU_ASSIGN_EXTENSION[5]=0	
$MC_AUXFU_ASSIGN_VALUE[5]=45	
$MC_AUXFU_ASSIGN_GROUP[5]=9	
$MC_AUXFU_ASSIGN_TYPE[6]="M"	; 对辅助功能 7 的描述：M1=40
$MC_AUXFU_ASSIGN_EXTENSION[6]=1	
$MC_AUXFU_ASSIGN_VALUE[6]=40	
$MC_AUXFU_ASSIGN_GROUP[6]=9	
	; . . .（辅助功能 8 至 11 以此类推）
$MC_AUXFU_ASSIGN_TYPE[11]="M"	; 对辅助功能 12 的描述：M1=45
$MC_AUXFU_ASSIGN_EXTENSION[11]=1	
$MC_AUXFU_ASSIGN_VALUE[11]=45	
$MC_AUXFU_ASSIGN_GROUP[11]=9	
$MN_AUXFU_GROUP_SPEC[9] = 'H22'	; 第 10 辅助功能组的输出特性
$MC_AUXFU_ASSIGN_TYPE[12]="M"	; 对辅助功能 13 的描述：M2=3
$MC_AUXFU_ASSIGN_EXTENSION[12]=2	
$MC_AUXFU_ASSIGN_VALUE[12]=3	
$MC_AUXFU_ASSIGN_GROUP[12]=10	
$MC_AUXFU_ASSIGN_TYPE[13]="M"	; 对辅助功能 14 的描述：M2=4
$MC_AUXFU_ASSIGN_EXTENSION[13]=2	
$MC_AUXFU_ASSIGN_VALUE[13]=4	
$MC_AUXFU_ASSIGN_GROUP[13]=10	
$MC_AUXFU_ASSIGN_TYPE[14]="M"	; 对辅助功能 15 的描述：M2=5
$MC_AUXFU_ASSIGN_EXTENSION[14]=2	

程序代码	注释
$MC_AUXFU_ASSIGN_VALUE[14]=5	
$MC_AUXFU_ASSIGN_GROUP[14]=10	
$MC_AUXFU_ASSIGN_TYPE[15]="M"	; 对辅助功能 16 的描述：M2=70
$MC_AUXFU_ASSIGN_EXTENSION[15]=2	
$MC_AUXFU_ASSIGN_VALUE[15]=70	
$MC_AUXFU_ASSIGN_GROUP[15]=10	
$MN_AUXFU_GROUP_SPEC[10]='H22'	; 第 11 辅助功能组的设定
$MC_AUXFU_ASSIGN_TYPE[16]="S"	; 对辅助功能 17 的描述：S2=<所有值>
$MC_AUXFU_ASSIGN_EXTENSION[16]=2	
$MC_AUXFU_ASSIGN_VALUE[16]=-1	
$MC_AUXFU_ASSIGN_GROUP[16]=11	
$MN_AUXFU_GROUP_SPEC[11]='H21'	; 第 12 辅助功能组的设定
$MC_AUXFU_ASSIGN_TYPE[17]="M"	; 对辅助功能 18 的描述：M50
$MC_AUXFU_ASSIGN_EXTENSION[17]=0	
$MC_AUXFU_ASSIGN_VALUE[17]=50	
$MC_AUXFU_ASSIGN_GROUP[17]=12	
$MC_AUXFU_ASSIGN_TYPE[18]="M"	; 对辅助功能 19 的描述：M51
$MC_AUXFU_ASSIGN_EXTENSION[18]=0	
$MC_AUXFU_ASSIGN_VALUE[18]=51	
$MC_AUXFU_ASSIGN_GROUP[18]=12	
$MN_AUXFU_GROUP_SPEC[12]='H21'	; 第 13 辅助功能组的设定
$MC_AUXFU_ASSIGN_TYPE[19]="M"	; 对辅助功能 20 的描述：M52
$MC_AUXFU_ASSIGN_EXTENSION[19]=0	
$MC_AUXFU_ASSIGN_VALUE[19]=52	
$MC_AUXFU_ASSIGN_GROUP[19]=13	
$MC_AUXFU_ASSIGN_TYPE[20]="M"	; 对辅助功能 21 的描述：M53
$MC_AUXFU_ASSIGN_EXTENSION[20]=0	
$MC_AUXFU_ASSIGN_VALUE[20]=53	
$MC_AUXFU_ASSIGN_GROUP[20]=13	

NC 数字量和模拟量 I/O

14

14.1 引言

功能

在 SINUMERIK 17x0 上可通过 PROFINET 连接 I/O 模块。通常情况下通过 PLC 用户程序访问相应的数字量和模拟量输入/输出。功能"SINUMERIK 17x0 的 NC 数字量和模拟量输入/输出"能够通过系统变量或编译循环直接从 NC（零件程序、同步动作和编译循环）访问 I/O 模块的输入/输出端。该 I/O 模块在下文称作 **NC 输入/输出**。

出于兼容性原因，有两种不同的功能可供使用：

1. 通过 PLC 间接访问输入/输出 (页 860)
   通过 NC 写入在连接至 PLC 的接口中读取和写入 I/O 模块的输入/输出的请求。这些请求以 OB1 周期循环执行。
   无论从组态范围还是响应时间上看，该功能的性能都是最低的。
2. 不通过 PLC 直接访问输入/输出 (页 885)
   通过 NC 直接访问控制系统内部的 I/O 模块输入/输出映像，无需经过 PLC。
   这从组态范围或响应时间来说都是现阶段性能测试功能。

前提条件

- NC 输入/输出的 PROFINET I/O 模块必须已连接且运行就绪。
- NC 输入/输出的 PROFINET I/O 模块的硬件配置已通过 TIA Portal 执行并已载入 PLC 中。

监控

下列监控对 NC 输入/输出生效：

- 启动：
  - 检查 PLC 侧识别的输入/输出与机床数据中设置的 NC 输入/输出是否一致。
- 循环运行：
  - 按插补周期进行生命符号监控
  - 按插补周期进行模块监控
  - 温度监控

出现故障时，系统会复位信号 `<Nc>.basic.in.ncReady` "NC-CPU 就绪" 并输出报警。

故障时的特性

出现故障（例如："NC-CPU 就绪" = 0）、NCU 出错以及掉电时，NC 输入/输出的数字量和模拟量输出会切换至安全状态 (0 V)。

应用

下列 NC 功能会使用 NC 输入/输出：

- 每个程序段内有多个进给率值或辅助功能
- 达到成品尺寸后快速退刀
- 针对特定轴的剩余行程删除
- 程序跳转
- 快速 NC 启动
- 模拟量测量钳
- 行程开关信号
- 冲裁/步冲功能
- 模拟量值控制

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Nc>.basic.in.ncReady	LBP_NCName.E_NCready	DB10.DBX108.7

14.2 通过 PLC 间接访问输入/输出

14.2.1 简要说明

SINUMERIK 17x0 **NCU** 上有三个数字量输入/输出接口（X122、X132 和 X142）。

接口 X142 的四个数字量输入/输出用作所谓的快速 NC 输入/输出。可通过首个地址字节或通过系统变量 $A_IN[1...4] 和 $A_OUT[1...4] 进行读写。

在 PROFINET IO 接口 X150 上可连接 I/O 模块。因此，NC 数字量输入/输出的组态范围可增加 **32** 路，NC 模拟量输入/输出的组态范围可增加 **8** 路。这些 NC 输入/输出在下文称作**外部** NC 输入/输出。

表格 14-1　NC 数字量和模拟量输入/输出的最大数量

	板载	NC 输入/输出	总计
数字量输入	4	32	36
数字量输出	4	32	36
模拟量输入	-	8	8
模拟量输出	-	8	8

更多信息

有关硬件的详细信息请见：

- SINUMERIK 17x0 设备手册
- SIMATIC ET 200S FC 操作说明

14.2.2　参数设置

机床数据

生效 NC 输入/输出的数量

在以下机床数据中设置生效或可由 NC 使用的**数字量**输入/输出**字节**的数量：

- NC 数字量输入**字节**的数量
  MD10350 $MN_FASTIO_DIG_NUM_INPUTS = <数量>
- NC 数字量输出**字节**的数量
  MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS = <数量>

在以下机床数据中设置生效或可由 NC 使用的**模拟量**输入/输出的数量：

- NC 模拟量输入的数量
  MD10300 $MN_FASTIO_ANA_NUM_INPUTS = <数量>
- NC 模拟量输出的数量
  MD10310 $MN_FASTIO_ANA_NUM_OUTPUTS = <数量>

输入/输出不必是实际存在的硬件口。此时，NC 内部会将信号状态或模拟量值置“零”。

通过 PLC 用户程序可在 NC 读取前修改输入端上的值。

槽地址

数字量输入/输出的定址

- 外部数字量输入对应的硬件口
  MD10366 $MN_HW_ASSIGN_DIG_FASTIN[<n>] = <地址 $_H$>
- 外部数字量输出对应的硬件口
  MD10368 $MN_HW_ASSIGN_DIG_FASTOUT[<n>] = <地址 $_H$>

模拟量输入/输出的定址

- 外部模拟量输入对应的硬件口
  MD10362 $MN_HW_ASSIGN_ANA_FASTIN[<n>] = <地址 $_H$>
- 外部模拟量输出对应的硬件口
  MD10364 $MN_HW_ASSIGN_ANA_FASTOUT[<n>] = <地址 $_H$>

<n>: 用于外部**数字量**输入/输出**字节** (0 ... 3) 或外部**模拟量**输入/输出 (0 ... 7) 定址的下标

<地址>: PROFINET 模块的槽地址，格式为 05 00 xxxx

05 PROFINET 模块的标识

00 固定设置

$xxxx_H$ 槽的十六进制逻辑起始地址

0000 = 无生效槽

对于 PLC 流程图中的逻辑起始地址须注意以下几点：

- 输入槽：可通过 NC 读取
- 输出槽：禁止通过 NC 写入 ⇒ NC 启动后报警

更多信息参见 NCU 17x0 PN 设备手册。

模拟量输入/输出的权重系数

通过权重系数可对各个 NC **模拟量**输入/输出进行调整，使其与所使用的模拟量 I/O 模块的模数或数模转换器相匹配：

- NC 模拟量输入的权重系数（参见“NC 模拟量输入 (页 873)”）：
  MD10320 $MN_FASTIO_ANA_INPUT_WEIGHT[<n>]
- NC 模拟量输出的权重系数（参见“NC 模拟量输出 (页 875)”）：
  MD10330 $MN_FASTIO_ANA_OUTPUT_WEIGHT[<n>]

<n>: 用于外部**模拟量**输入/输出 (0 ... 7) 定址的下标

指定 NC 功能

输入/输出是多个 NC 功能生效的前提条件。根据功能通过机床数据指定所用的输入/输出，例如：功能“一个程序段中编写多个仅给率”便是通过以下机床数据指定输入/输出的：

MD21220 $MC_MULTFEED_ASSIGN_FASTIN = <字节地址>

<字节地址>	NC 数字量输入/输出	
0	无	
1	1 ... 4	板载输入/输出
	5 ... 8	非硬件 NC 输出
2	9 ... 16	外部 NC 输入/输出
3	17 ... 24	外部 NC 输入/输出
4	25 ... 32	外部 NC 输入/输出
5	33 ... 40	外部 NC 输入/输出
128	比较器字节 1 的输入 1 至 8	
129	比较器字节 2 的输入 9 至 16	

说明

重复指定

输入的重复指定不被系统视为错误设置。

输出的重复指定会被系统视为错误设置，如果在启动时发现这一错误，则会显示报警。

示例：外部输入输出的硬件指定

通过 NC 读/写 PROFINET 模块的数字量输入/输出需设置两个输入字节和一个输出字节。

生效 NC 输入/输出字节的数量

MD10350 $MN_FASTIO_DIG_NUM_INPUTS ；2 个数字量输入字节 + 1 个板载字节 = 2 + 1 = 3

MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS ；1 个数字量输出字节 + 1 个板载字节 = 1 + 1 = 2

硬件指定

MD10366 $MN_HW_ASSIGN_DIG_FASTIN[0] = 'H5000**200**' ；第 1 个输入字节中的 $A_IN[9] ... [16]

MD10366 $MN_HW_ASSIGN_DIG_FASTIN[1] = 'H5000**201**' ；第 2 个输入字节中的 $A_IN[17] ... [24]

MD10368 $MN_HW_ASSIGN_DIG_FASTOUT[0] = 'H5000**200**' ；第 1 个输出字节中的 $A_OUT[9] ... [16]

机床数据中输入的十六进制地址 200_H 和 201_H 与在 TIA Portal 中配置时给定的十进制逻辑基准地址 512_D 和 513_D 一致

14.2.3 系统变量

输入数据

系统变量	下标或输入编号 <n>
$A_IN[<n>]	1 ... 4 和 9 ... 40，参见 NC 数字量输入 (页 865)
$A_INA[<n>]	1 ... 8，参见 NC 模拟量输入 (页 873)

从零件程序中读取输入数据时，通道中会触发预处理停止。

输出数据

系统变量	下标或输出编号 <n>
$A_OUT[<n>]	1 ... 4 和 9 ... 40，参见 NC 数字量输出 (页 867)
$A_OUTA[<n>]	1 ... 8，参见 NC 模拟量输出 (页 875)

从零件程序中读取输出数据时，通道中会触发预处理停止。

14.2.4 比较器输入

除 I/O 输入外，系统还提供了 16 个控制系统内部比较器输入。

通过对比模拟量输入与设定数据中规定的阈值得出比较器输入的当前信号状态。

参见“比较器输入 (页 881)”。

14.2.5 NC 数字量输入/输出

14.2.5.1 NC 数字量输入

功能

通过系统变量 $A_IN 可在 NC 程序或同步动作中读取数字量 NC 输入的值。通过 NC/PLC 接口信号可影响读取的值。

应用示例

NC 数字量输入可用于以下 NC 功能：

- 删除定位轴的剩余行程
- 在程序段末尾进行快速程序跳转
- 编写读取禁止
- 在一个程序段内编写多个进给率值

更多信息

功能手册之同步动作

NC/PLC 接口信号

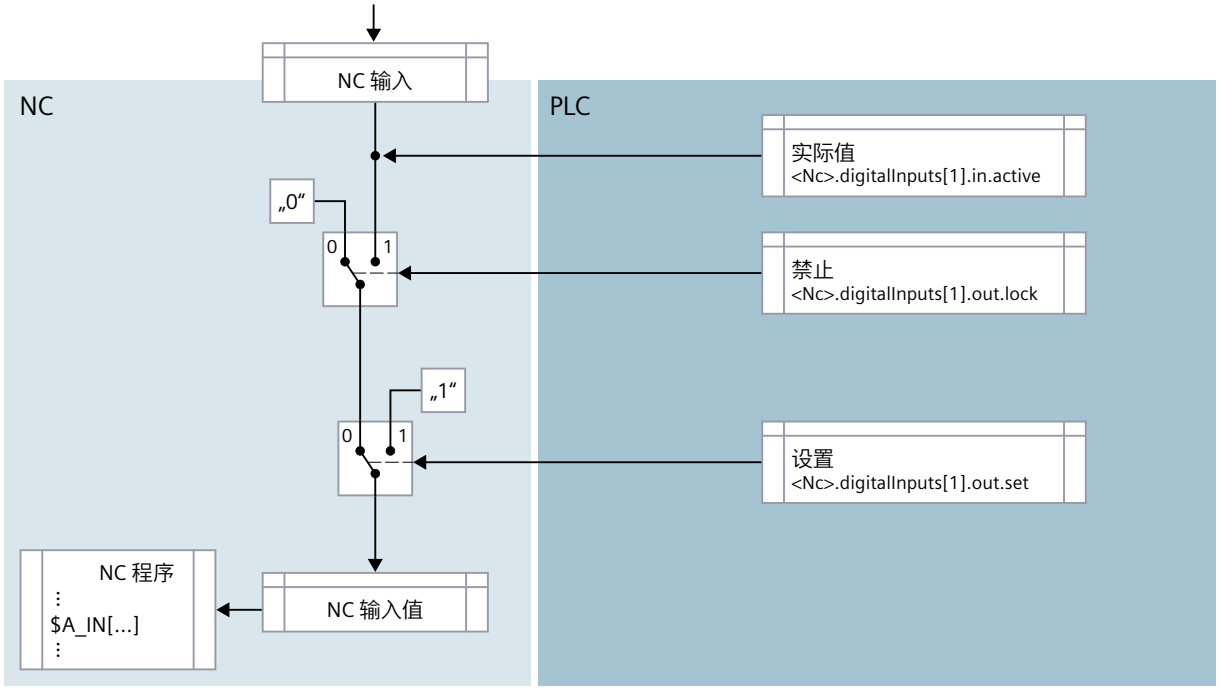


图 14-1　NCK 数字量输入的信号流

实际值

通过实际值接口可在 PLC 用户程序中读取 NC 输入的当前值。

说明

不同的值

接口“实际值”中的值可能因下列不同的影响方法而不同于通过系统变量 $A_IN 读取的 NC 输入值。

禁止

如果在该接口中设置了一个位，则将值 0 传送给相应的输入。

设置

如果在该接口中设置了一个位，则将值 1 传送给相应的输入。

概述

禁止	设置	实际值	NC 输入
<Nc>.digitalInputs[1].out.lock	<Nc>.digitalInputs[1].out.set	<Nc>.digitalInputs[1].in.active	1 - 40

边界条件

热启动和通道复位后的特性

在热启动和通道复位后继续传输硬件输入上的值。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Nc>.digitalInputs[1].in.active	LBP_NC.E_ActVal_In1 - 40	DB10, ...DBX060.0..7, .DBX186.0..DBX189.7

PLC → NC

Basic Program Plus	Basic Program	
<Nc>.digitalInputs[1].out.set	LBP_NC.A_Set_Inp1 - 40	DB10, ...DBX001.0..7, .DBX123.0..7, .DBX125.0..7, .DBX127.0..7, .DBX129.0..7
<Nc>.digitalInputs[1].out.lock	LBP_NC.A_Disabl_Inp1 - 40	DB10, ...DBX000.0..7, .DBX122.0..7, .DBX124.0..7, .DBX126.0..7, .DBX128.0..7

14.2.5.2 NC 数字量输出

功能

通过系统变量 $A_OUT 可在 NC 程序或同步动作中写入数字量 NC 输出的值。通过 NC/PLC 接口信号可影响写入的值。

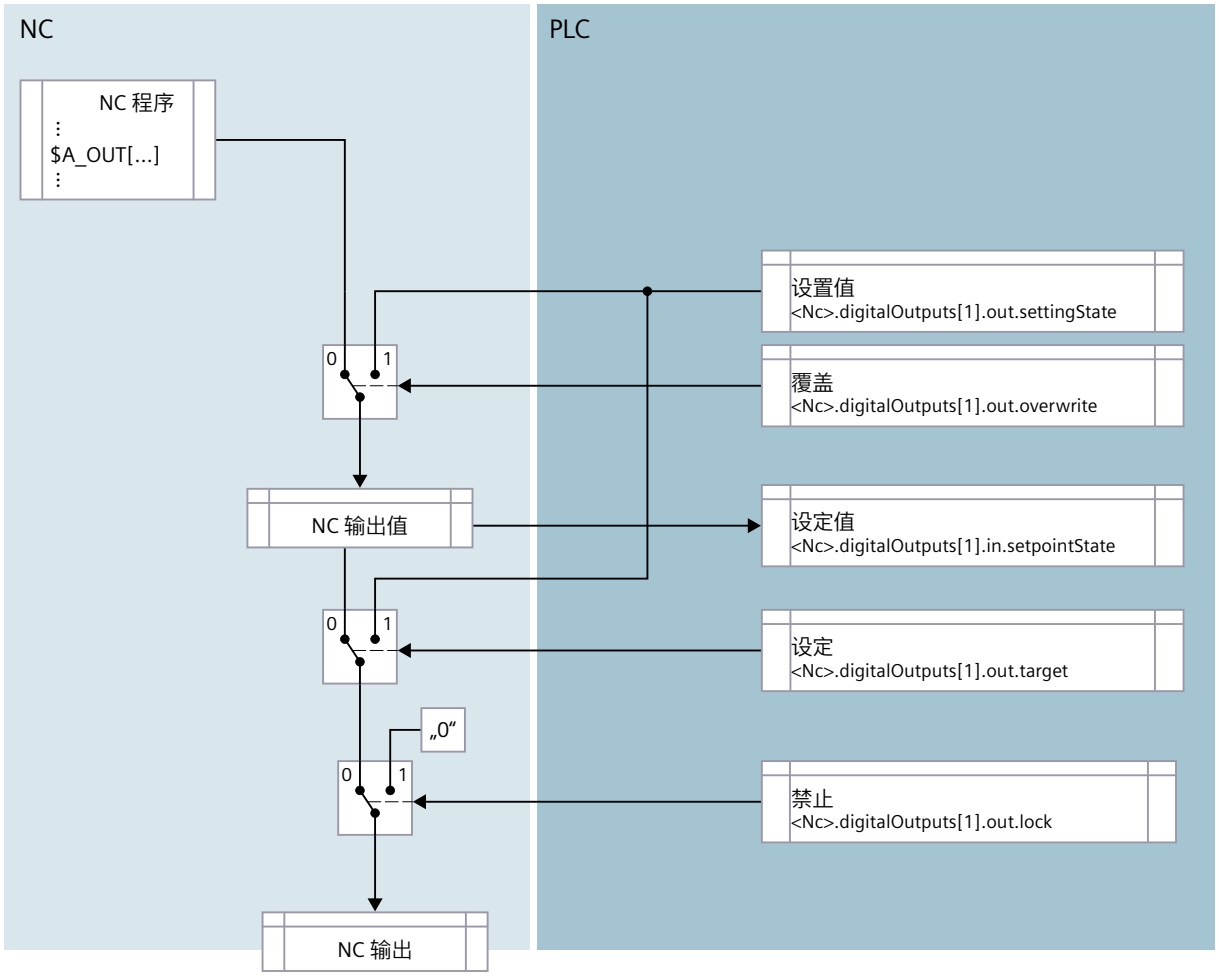
应用示例

- 行程开关信号（参见功能手册之*轴和主轴*，章节“软件凸轮”）
- 比较器信号的输出

更多信息

功能手册之同步动作

NC/PLC 接口信号



覆盖

如果在该接口中设置了一个位，相应的输出不会使用通过系统变量 $A_OUT 写入的值，而是使用由 PLC 用户程序指定的设置值。此时，通过系统变量 $A_OUT 写入的值会丢失。

如果在该接口中将一个位复位，则保留硬件输出上的当前值。

设置值

通过设置值可由 PLC 用户程序指定一个输出值。为了使设置值生效，必须通过针对“覆盖”或“设定”的接口将其激活。

设定值

通过设定值可在 PLC 用户程序中读取当前 NC 输出值。

说明

不同的值

接口“设定值”中的值可能因下列不同的影响方法而不同于 NC 输出上的值。

设定

如果在该接口中设置了一个位，相应的输出不会使用 NC 输出值，而是使用由 PLC 用户程序指定的设置值。此时保留当前 NC 输出值。

如果在该接口中将一个位复位，则最后的 NC 输出值重新生效。

禁止

如果在该接口中设置了一个位，则将值 0 输出给相应的输出。

概述

禁止	覆盖	设置值	设定	设定值	NC 输出
<Nc>.digitalOutputs[1].out.lock	<Nc>.digitalOutputs[1].out.overwrite	<Nc>.digitalInputs[1].out.set	<Nc>.digitalOutputs[1].out.settingState	<Nc>.digitalOutputs[1].out.target	1 - 40

边界条件

不带硬件的 NC 输出

如要说明指定的 NC 输出 (MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS)，但不存在相关硬件，则不显示报警。设定值可通过 PLC 用户程序读取。

程序结束复位/通道复位时的特性

从 PLC 用户程序出发，可通过针对“覆盖”、“设定”或“禁止”的接口，在程序结束复位或通道复位时针对具体应用设置每个 NC 输出。

热启动时的特性

热启动后，所有 NC 输出都会置“0”。从 PLC 用户程序出发，可通过改值位或禁用位设置每个 NC 输出。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Nc>.digitalOutputs[1].out.settingState	LBP_NC.E_Setpoint_Out1 - 40	DB10, ...DBX060.0..7, .DBX186.0..189.7

PLC → NC

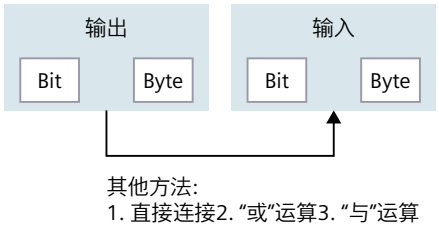
Basic Program Plus	Basic Program	
<Nc>.digitalOutputs[1].out.lock	LBP_NC.A_Disabl_Out1 - 40	DB10, ...DBX004.0..7, .DBX130.0..7, .DBX134.0..7, .DBX138.0..7, .DBX142.0..7
<Nc>.digitalOutputs[1].out.overwrite	LBP_NC.A_OvMask_Out1 - 40	DB10, ...DBX005.0..7, .DBX131.0..7, .DBX135.0..7, .DBX139.0..7, .DBX143.0..7
<Nc>.digitalOutputs[1].out.settingState	LBP_NC.A_Set_Out1 - 40	DB10, ...DBX006.0..7, .DBX132.0..7, .DBX136.0..7, .DBX140.0..7, .DBX144.0..7
<Nc>.digitalOutputs[1].out.target	LBP_NC.A_InMask_Out1 - 40	DB10, ...DBX007.0..7, .DBX133.0..7, .DBX137.0..7, .DBX141.0..7, .DBX145.0..7

14.2.5.3 高速数字量输入/输出之间的互联（直连和逻辑运算）

功能

NC 输入/输出中的高速输入可根据高速输出的信号状态通过软件加以设置：

一览：



直连

NC 输入/输出中的高速输入直接设为高速输出的信号状态。

逻辑或运算

NC 输入/输出中高速输入的信号状态是输出信号和对应输出信号经过逻辑或运算后的结果。

逻辑和运算

NC 输入/输出中高速输入的信号状态是输出信号和对应输出信号经过逻辑和运算后的结果。

特殊情况

- 若向同一个输入位指定了多个输出位，则机床数据下标最高的设定生效。
- 若指定了不存在或未激活的输入/输出，系统会忽略此指定且不发出报警。可查看下面两个机床数据的值来检查有效 NC 输入/输出字节的数量。
  MD10350 $MN_FASTIO_DIG_NUM_INPUTS
  MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS。

定义输入/输出之间的互联

输入/输出之间的互联是通过以下机床数据定义的：
MD10361 $MN_FASTIO_DIG_SHORT_CIRCUIT[n]

n：取值范围为 0 至 9，因此最多可指定 **10** 组互联。

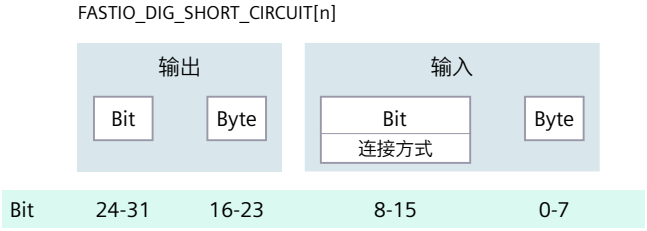
有 2 个十六进制字符用于指定一个输入的字节和位以及一个输出的字节和位。

互联方式在输入的位 12 至 15 中指定，选项有：0、A 和 B：

0　直连

A　逻辑和运算

B　逻辑或运算



示例

直连：

MD10361 $MN_FASTIO_DIG_SHORT_CIRCUIT = '04010302H'

输出 4，字节 1 和

输入 3，字节 2 直连

逻辑和运算：

MD10361 $MN_FASTIO_DIG_SHORT_CIRCUIT = '0705A201H'

输出 7，字节 5 和

输入 2，字节 1 逻辑和运算

逻辑或运算：

MD10361 $MN_FASTIO_DIG_SHORT_CIRCUIT = '0103B502H'

输出 1，字节 3 和

输入 5，字节 2 逻辑或运算

14.2.6 NC 模拟量输入/输出

14.2.6.1 NC 模拟量输入

功能

通过系统变量 $A_INA 可在 NC 程序或同步动作中读取模拟量 NC 输入的值。通过 NC/PLC 接口信号可影响读取的值。

二进制模拟量值显示

NC/PLC 接口信号

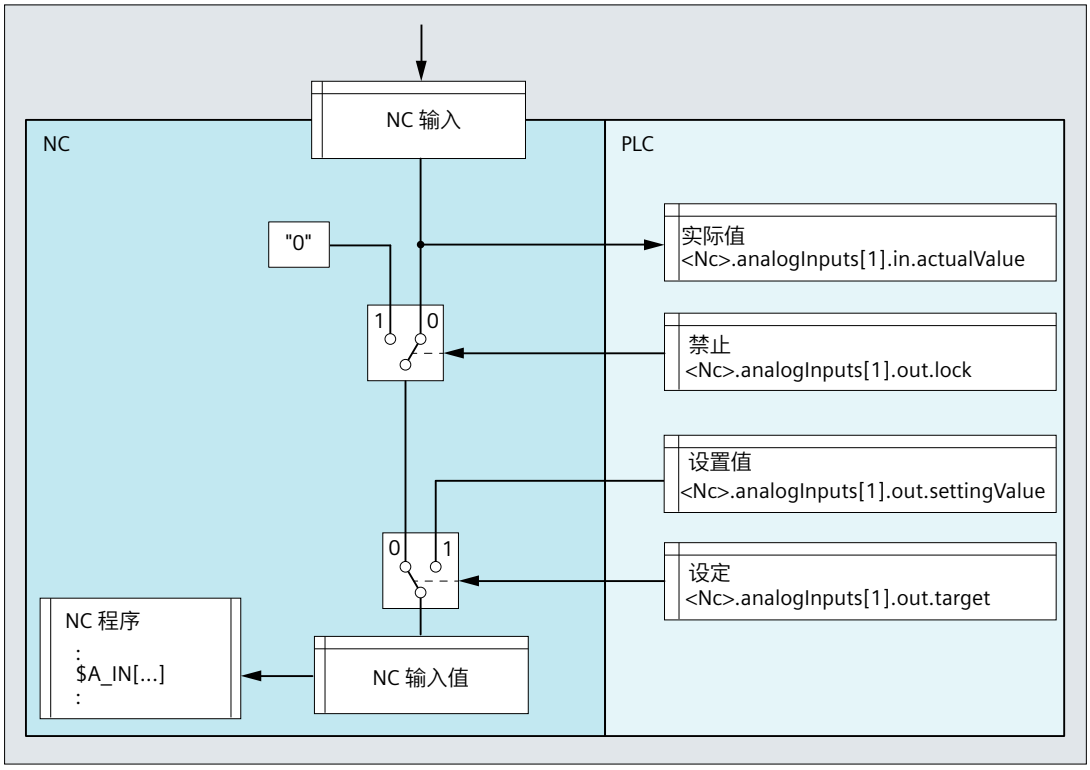


图 14-2 NCK 模拟量输入的信号流

实际值

通过实际值接口可在 PLC 用户程序中读取 NC 输入的当前值。

实际值作为定点数（16 位，含符号）以补码表示。

参见“模拟量输入/输出值的表示法 (页 879)”。

说明

不同的值

接口“实际值”中的值可能因下列不同的影响方法而不同于通过系统变量 $A_INA 读取的 NC 输入值。

禁止

如果在该接口中设置了一个位，则将值 0 传送给相应的输入。

设置值

通过设置值可由 PLC 用户程序指定一个输入值。为了使设置值生效，必须通过接口“设定”将其激活。

设置值必须作为定点数（16 位，含符号）以补码设定。

参见“模拟量输入/输出值的表示法 (页 879)”。

设定

如果在该接口中设置了一个位，则将对应的“设置值”传送给相应的输入。

概述

禁止	设定	设置值	实际值	NC 输入
<Nc>.analogInputs[1].out.lock	<Nc>.analogInputs[1].out.target	<Nc>.analogInputs[1].out.settingValue	<Nc>.analogInputs[1].in.actualValue	1 - 8

机床数据

权重系数

通过权重系数可对每个 NC 模拟量输入进行调整，使其与所使用的 I/O 模块的数模转换器相匹配：

MD10320 $MN_FASTIO_ANAINTPUT_WEIGHT[<输出>]

说明

不带硬件的 NC 模拟量输入

权重系数为 32767 时，对于 NC 程序和 PLC 用户程序，模拟量值经过模数处理后得出的数值是相同的。这样便能将 NC 输出用于 NC 程序与 PLC 用户程序之间的 1:1 通讯。

边界条件

不带硬件的 NC 模拟量输入

如要说明指定的 NC 输出 (MD10300 $MN_FASTIO_ANA_NUM_INPUTS)，但不存在相关硬件，则不显示报警。实际值可通过 PLC 用户程序读取。

热启动、程序结束复位和通道复位时的特性

在热启动、程序结束复位或通道复位后，为所有 NC 输入传送模拟量值。从 PLC 用户程序出发，可针对具体应用设置每个 NC 输入。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Nc>.analogInputs[1].in.actualValue	LBP_NC.E_ActVal_anaIn1 - 8	DB10, ...DBW194..208

PLC → NC

Basic Program Plus	Basic Program	
<Nc>.analogInputs[1].out.lock	LBP_NC.A_Disabl_anaIn1 - 8	DB10, ...DBX146.0..7
<Nc>.analogInputs[1].out.settingValue	LBP_NC.A_Setval_anaIn1 - 8	DB10, ...DBW148..162
<Nc>.analogInputs[1].out.target	LBP_NC.A_InMask_anaIn1 - 8	DB10, ...DBX147.0..7

14.2.6.2 NC 模拟量输出

功能

通过系统变量 $A_OUTA 可在 NC 程序或同步动作中写入模拟量 NC 输出的值。可通过机床数据和 NC/PLC 接口信号影响写入的值。

更多信息

功能手册之同步动作

NC/PLC 接口信号

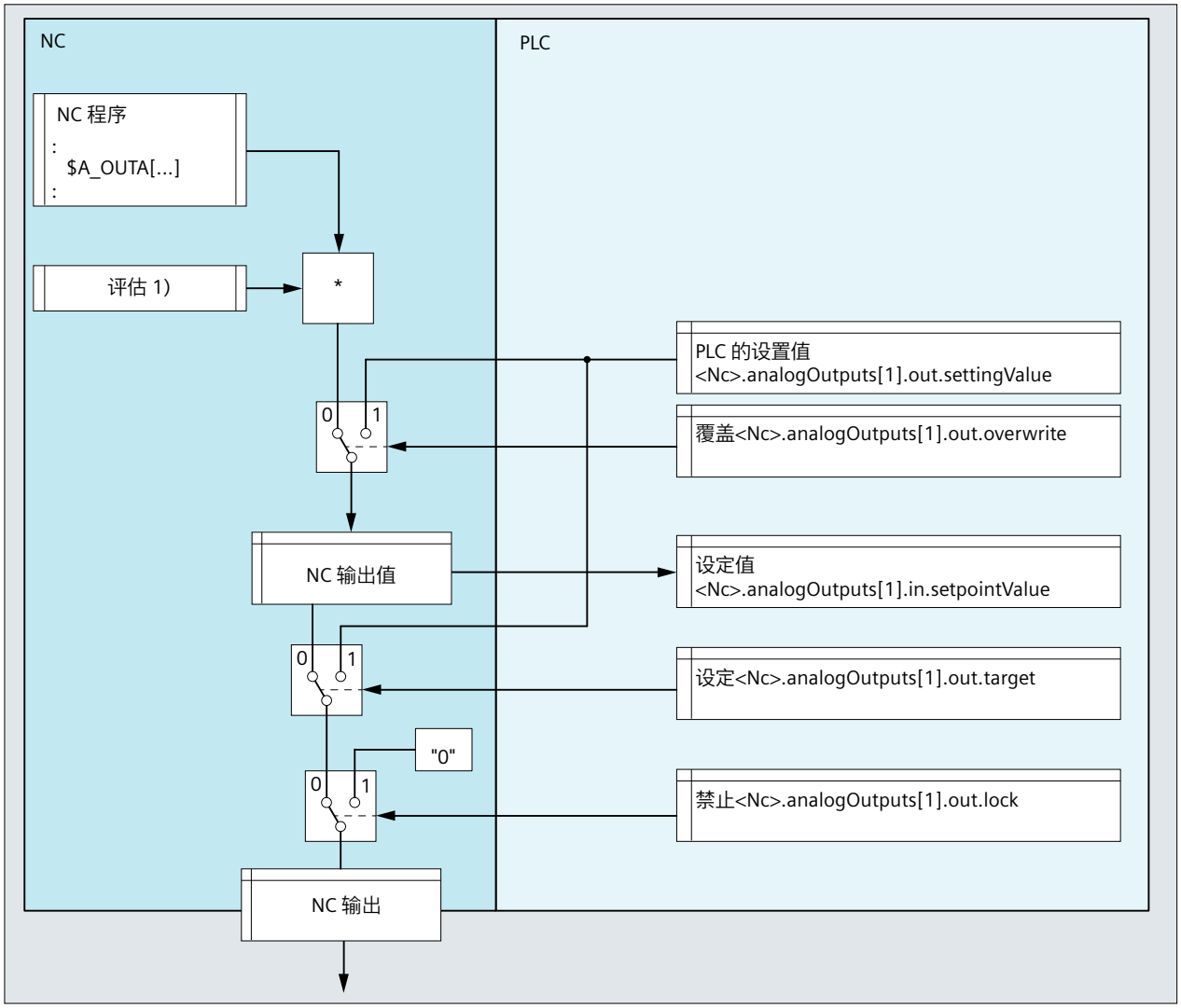


图 14-3　NCK 模拟量输出的信号流

设置值

通过设置值可由 PLC 用户程序指定一个输出值。为了使设置值生效，必须通过针对“覆盖”或“设定”的接口将其激活。

设置值必须作为定点数（16 位，含符号）以补码设定。

参见“模拟量输入/输出值的表示法 (页 879)”。

覆盖

如果设置了一个位，相应的输出不会使用通过系统变量 $A_OUTA 写入的值，而是使用由 PLC 用户程序指定的设置值。此时，通过系统变量 $A_OUTA 写入的值会丢失。

如果将一个位复位，则为相应输出保留硬件输出上的当前值。

设定

如果设置了一个位，相应的输出不会使用 NC 输出值，而是使用由 PLC 用户程序指定的设置值。此时保留当前 NC 输出值。

如果将一个位复位，则对于相应的输出而言，最后的 NC 输出值重新生效。

禁止

如果在该接口中设置了一个位，则在相应的 NC 输出上输出值 0 Volt。

设定值

通过设定值可在 PLC 用户程序中读取当前 NC 输出值。

设定值作为定点数（16 位，含符号）以补码表示。

参见“模拟量输入/输出值的表示法 (页 879)”。

说明

不同的值

接口“设定值”中的值可能因下列不同的影响方法而不同于 NC 输出上的值。

概述

禁止	覆盖	设置值	设定	设定值	NC 输出
<Nc>.analogOutputs[1].out.lock	<Nc>.analogOutputs[1].out.overwrite	<Nc>.analogOutputs[1].out.settingValue	<Nc>.analogOutputs[1].out.target	<Nc>.analogOutputs[1].in.setpointValue	1 - 8

机床数据

权重系数

通过权重系数可对每个 NC 模拟量输出进行调整，使其与所使用的 I/O 模块的数模转换器相匹配：

MD10330 $MN_FASTIO_ANA_OUTPUT_WEIGHT[<输出>]

程序代码	注释
; 假设：模拟值范围 ±10 V	
; 参数设置：MD10330 $MN_FASTIO_ANA_OUTPUT_WEIGHT[<Ausgang>] = 10000	
$A_OUTA[<输出>] = 9500	; NC 输出：9.5 V
$A_OUTA[<输出>] = -4120	; NC 输出 : -4.12 V

说明

不带硬件的 NC 模拟量输出

权重系数为 32767 时，对于 NC 程序和 PLC 用户程序，模拟量值经过模数处理后得出的数值是相同的。这样便能将 NC 输出用于 NC 程序与 PLC 用户程序之间的 1:1 通讯。

边界条件

非硬件 NC 模拟量输出

如要说明指定的 NC 输出 (MD10310 $MN_FASTIO_ANA_NUM_OUTPUTS)，但不存在相关硬件，则不显示报警。设定值可通过 PLC 用户程序读取。

热启动时的特性

热启动后，所有 NC 输出都会置“0”。从 PLC 用户程序出发，可针对具体应用设置每个 NC 输出。

程序结束复位/通道复位时的特性

从 PLC 用户程序出发，可通过针对“覆盖”、“设定”或“禁止”的接口，在程序结束复位或通道复位时针对具体应用设置每个 NC 输出。

PLC 信号

NC → PLC

Basic Program Plus	Basic Program	
<Nc>.analogOutputs[1].in.setpointValue	LBP_NC.E_Setval_anaOut1 - 8	DB10, ...DBW210..224

PLC → NC

Basic Program Plus	Basic Program	
<Nc>.analogOutputs[1].out.overwrite	LBP_NC.A_OvMask_anaOut1 - 8	DB10, ...DBX166.0..7
<Nc>.analogOutputs[1].out.target	LBP_NC.A_InMask_anaOut1 - 8	DB10, ...DBX167.0..7

Basic Program Plus	Basic Program	
<Nc>.analogOutputs[1].out.lock	LBP_NC.A_Disabl_anaOut1 - 8	DB10, ...DBX168.0 ..7
<Nc>.analogOutputs[1].out.settingValue	LBP_NC.A_Setval_anaOut1 - 8	DB10, ...DBW170.. 184

14.2.6.3 模拟量输入/输出值的表示法

模拟量值经过模数转换后在 NC/PLC 接口上作为定点数（16 位，含符号）以补码表示。

位编号	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
含义	VZ	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

VZ：符号

最小值	最大值
-32768_D	32767_D
8000_H	$7FFF_H$

步长

模数转换精度为 16 位、额定范围为 ±10 V 的情况下，步长为：

$20\ V / 2^{16} = 20\ V / 65536 \approx 0.305\ mV$

模数转换精度 < 16 位

若模拟量模块的模数转换精度小于 16 位（包含符号），那么模拟量值经过模数转换后的数值将会从位 14 起进入接口。未占用的低位将填 0。

14 位模数转换精度

模数转换精度为 14 位（包含符号），额定范围为 ±10 V 的情况下，步长为：

$20\ V / 2^{14} = 20\ V / 16384 \approx 1.22\ mV$

位 0 和位 1 总是为 0。

12 位模数转换精度

模数转换精度为 12 位（包含符号），额定范围为 ±10 V 的情况下，步长为：

$20\ V / 2^{12} = 20\ V / 4096 \approx 4.88\ mV$

位 0 到 3 总是为 0。

不同模数转换精度下最大值的表示

位编号	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
含义	VZ	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
16 位模数转换精度：$32767_D = 7FFF_H$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
14 位模数转换精度：$8191_D = 1FFF_H$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**	**0**
12 位模数转换精度：$2047_D = 7FF_H$	0	1	1	1	1	1	1	1	1	1	1	1	**0**	**0**	**0**	**0**

说明

所使用的模拟量输入/输出模块的数据（模数转换精度、额定范围）请见相应模块的文档。

示例

模数转换精度为 14 位（包含符号），额定范围为 ±10 V 时模拟量值经过模数转换后的数值表法方法：

示例 1：模拟量值 = 9.5 V

经过模数转换后的十进制表示：	9.5 V / 20 V * 16384 = 7782
经过 14 位模数处理后的二进制表示：	01 1110 0110 0110
经过 16 位模数处理后的二进制表示：	0111 1001 1001 10**00**
经过 16 位模数处理后的十六进制表示：	7998_H

示例 2：模拟量值 = -4.12 V

经过模数转换后的十进制表示：	-4.12 V / 20 V * 16384 = -3375
经过 14 位模数处理后的二进制表示：	11 0010 1101 0001
经过 16 位模数处理后的二进制表示：	1100 1011 0100 01**00**
经过 16 位模数处理后的十六进制表示：	$CB44_H$

14.2.7 比较器输入

功能

除了 NC 数字量/模拟量输入外，系统还提供 2 个内部比较器输入字节，每个字节各带有 8 个比较器输入。比较器输入的信号状态是通过对比高速模拟量输入信号与设定数据中可调阈值得出的。

比较器输入 <n> 的信号状态（即比较结果）可直接在零件程序中通过系统变量 $A_INCO[<n>] 查询。

下标 <n>：

<n> = 1 ... 8　　用于比较器字节 1
<n> = 9 ... 16　　用于比较器字节 2

术语

In der vorliegenden Beschreibung werden die Begriffe **Komparator-Eingänge** (mit Index <n>; Wertebereich von <n>:1 ... 8 bzw. 9 ... 16) und **Komparator-Eingangsbits** (mit Index <b>; Wertebereich von <b>:0 ... 7) verwendet.

这两个术语有以下关联：

<n> = 1 ... 8：　　比较器输入 <n> 与比较器输入位的对应关系：<b> = <n> - 1
<n> = 9 ... 16：　　比较器输入 <n> 与比较器输入位的对应关系：<b> = <n> - 9

示例：比较器输入 1 对应比较器输入位 0。

指定模拟量输入

通过以下机床数据为比较器字节 1 的输入位 <b> 指定一个模拟量输入：

MD10530 $MN_COMPAR_ASSIGN_ANA_INPUT_1[<b>]

示例：

MD10530 $MN_COMPAR_ASSIGN_ANA_INPUT_1[0] = 1

MD10530 $MN_COMPAR_ASSIGN_ANA_INPUT_1[1] = 1

MD10530 $MN_COMPAR_ASSIGN_ANA_INPUT_1[7] = 7

模拟量输入 1 作用于比较器字节 1 的输入位 0 和 1。

模拟量输入 7 作用于比较器字节 1 的输入位 7。

为比较器字节 2 指定模拟量输入则通过以下机床数据进行：

MD10531 $MN_COMPAR_ASSIGN_ANA_INPUT_2[<b>]

比较器设置

比较器字节 1 或 2 的各个位（0 到 7）通过以下机床数据设置：

MD10540 $MN_COMPAR_TYPE_1（比较器字节 1 的设置）

或者

MD10541 $MN_COMPAR_TYPE_2（比较器字节 2 的参数）

可以进行下列设置：

- 比较类型位（位 0 ... 7）
  为每个比较器输入位定义比较条件的类型：

  位 = 1： 满足以下条件时，对应的比较器输入位置 1：
  模拟量值 ≥ 阈值
  位 = 0： 满足以下条件时，对应的比较器输入位置 0：
  模拟量值 < 阈值

- 通过 NC 数字量输出输出比较器输入字节（位 16 ... 23）
  比较器位还可通过 NC 数字量输出以字节方式直接输出。为此，必须在此字节中（位 16 ... 23）设定通过哪个 NC 数字量输出字节输出。

  字节 = 0： 不通过 NC 数字量输出输出
  字节 = 1： 通过 NC 板载的数字量输出 9 ... 16 输出
  字节 = 2： 通过外部 NC 数字量输出 17 ... 24 输出
  字节 = 3： 通过外部 NC 数字量输出 25 ... 32 输出
  字节 = 4： 通过外部 NC 数字量输出 33 ... 40 输出

- 用于输出比较器输入字节的取反位（位 24 ... 31）
  对每个比较器信号可额外定义，是否对在 NC 数字量输出上输出的信号状态进行取反。

  位 = 1： 不取反对应的比较器输入位。
  位 = 0： 取反对应的比较器输入位。

阈值

在比较器字节 1 或 2 上用于比较的阈值必须作为设定数据保存。必须为每个比较器输入位 <b>（其中 <b> = 0 ... 7）输入一个独立的阈值：

SD41600 $SN_COMPAR_THRESHOLD_1[<b>]

或者

SD41601 $SN_COMPAR_THRESHOLD_2[<b>]

比较器信号作为 NC 数字量输入

所有执行顺序取决于 NC 数字量输入的 NC 功能均可通过比较器的信号状态进行控制。此时须在 NC 功能对应的机床数据（“指定使用的硬件字节”）中为比较器字节 1（硬件字节 128）或 2（硬件字节 129）输入字节地址。

示例：

NC 功能“在一个程序段内编写多个进给率”。

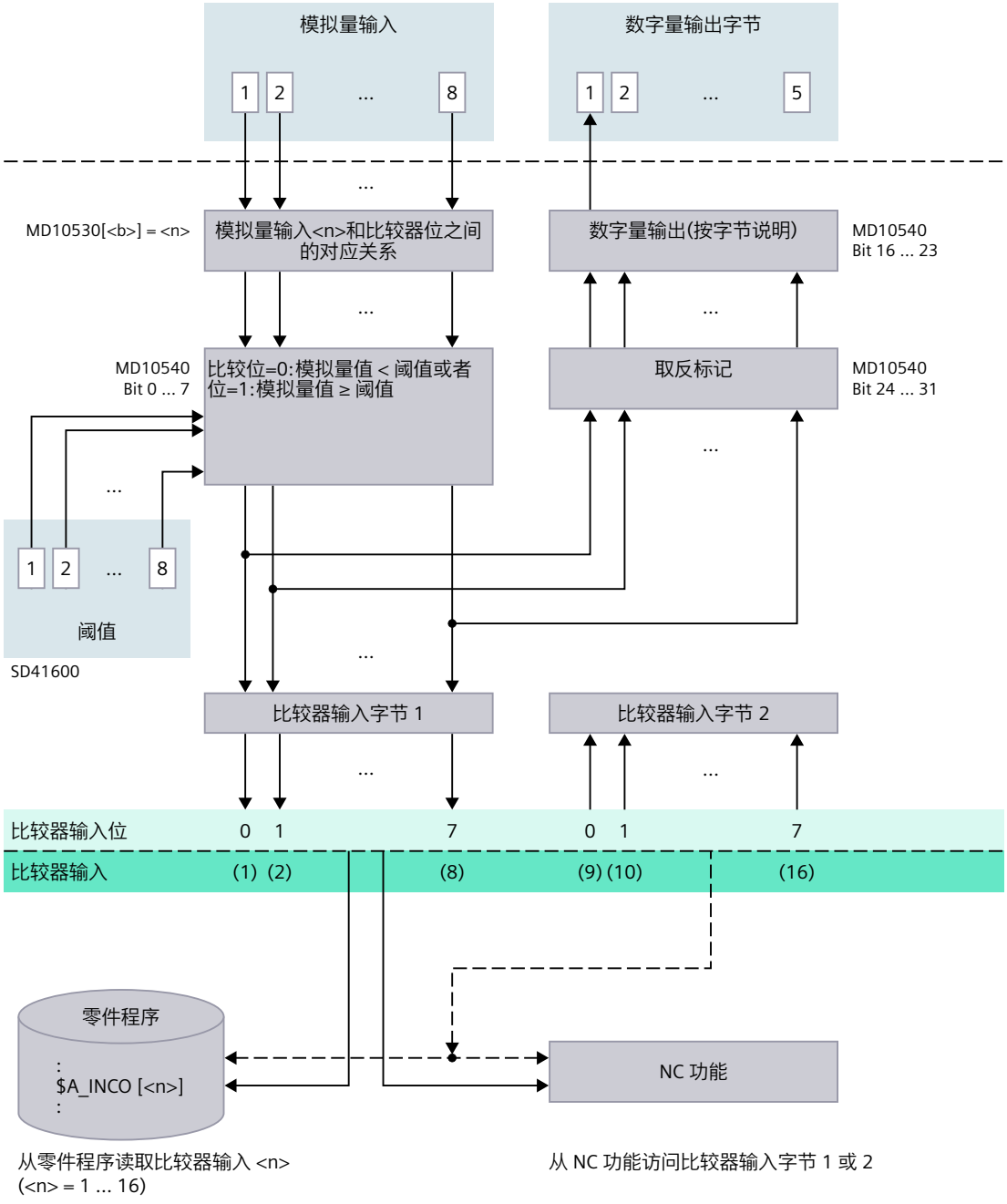
通道专用机床数据中的输入：

MD21220 $MC_MULTFEED_ASSIGN_FASTIN = 129

这样便会根据比较器字节 2 的状态激活不同的进给率值。

工作流程

下图显示了比较器输入字节 1 的工作流程。



14.3 不通过 PLC 直接访问输入/输出

14.3.1 简要说明

等时同步和非等时同步 PROFINET

在配置等时同步和非等时同步 PROFINET 时都可读/写 PROFINET 输入/输出。

I/O 区域

用于 NC 输入/输出的 RPOFINET IO 设备的槽按照升序**连续**排列时，则在下文称作 I/O 区域。

I/O 区域的特征如下：

- 逻辑初始地址：PROFINET IO 设备的第一个槽的逻辑初始地址
- 长度：PROFINET IO 设备使用的所有槽的总长度

在 NC 中通过机床数据设置 I/O 区域的逻辑初始地址和长度（参见“参数设置 (页 886)”）。

读/写

零件程序/同步动作：系统变量

在 NC 中通过机床数据按插补周期读写 NC 输入/输出。按插补周期写入 NC 输入/输出。

数据一致性：插补周期

参见“系统变量 (页 888)”。

并行读/写

读取

通过编译循环和零件程序/同步动作**可以**并行读取同一个 I/O 区域的输入数据。以不同的方式对 NC 输入/输出进行读访问。

- **确保**了**数据一致性**。
- 插补周期期间**无法确保数据一致性**。

写入

通过编译循环和零件程序/同步动作**无法**并行写入同一个 I/O 区域的输出数据。

14.3.2 参数设置

机床数据

I/O 区域的逻辑初始地址

通过以下机床数据设置所用 I/O 区域的逻辑初始地址

- 输入区域 1, 2, ... m 的逻辑初始地址：
  MD10500 $MN_DPIO_LOGIC_ADDRESS_**IN**[<n>] = <逻辑初始地址>；其中 <n> = 0, 1, 2, ... (m - 1)
- 输出区域 1, 2, ... m 的逻辑初始地址：
  MD10510 $MN_DPIO_LOGIC_ADDRESS_**OUT**[<n>] = <逻辑初始地址>；其中 <n> = 0, 1, 2, ... (m - 1)
  对于 PLC 流程图中的逻辑起始地址须注意以下几点：
  – 输入槽：可通过 NC 读取
  – 输出槽：禁止通过 NC 写入 ⇒ NC 启动后报警

更多信息

NCU 17x0. PN 设备手册，技术数据 > PLC > 流程图大小
详细信息请见 地址 (https://support.industry.siemens.com/cs/cn/zh/view/54058408)

说明

用于对 PROFINET I/O 进行**写**访问 (MD10510) 的 I/O 区域不可位于流程图区域（例如：PLC 1500）内。

对于 PROFINET I/O，必须将流程图中的 I/O 区域指定给 “PIP NCK”，通过周期同步报警 “NC” 进行设置。

I/O 区域的长度

通过以下机床数据设置所用 I/O 区域的长度：

- 输入区域 1, 2, ... m 的长度：
  MD10501 $MN_DPIO_RANGE_LENGTH_**IN**[<n>] = <长度>；其中 <n> = 0, 1, 2, ... (m - 1)
- 输出区域 1, 2, ... m 的长度：
  MD10511 $MN_DPIO_RANGE_LENGTH_**OUT**[<n>] = <长度>；其中 <n> = 0, 1, 2, ... (m - 1)

如果输入值 0，系统内部会将在给定的逻辑初始地址 (MD10500 / MD10510) 下找到的首个有效数据槽的长度设为 I/O 区域的长度。

I/O 区域的属性

- 输入区域 1, 2, ... m 的属性：
  MD10502 $MN_DPIO_RANGE_ATTRIBUTE_**IN**[<n>]；其中 <n> = 0, 1, 2, ... (m - 1)

位	值	含义
0	系统变量 $A_DPx_IN[<n>,<m>] 的表示方式	
	0	Little-Endian 格式
	1	Big-Endian 格式
2	读取输入数据	
	0	可通过系统变量和 CC-Binding 读取
	1	只可通过 CC-Binding 读取
3	输出槽生命符号报警	
	0	输出槽生命符号报警
	1	抑制槽生命符号报警

- 输出区域 1, 2, ... m 的属性：
  MD10512 $MN_DPIO_RANGE_ATTRIBUTE_**OUT**[<n>]；其中 <n> = 0, 1, 2, ... (m - 1)

位	值	含义
0	系统变量 $A_DPx_OUT[<n>,<m>] 的表示方式	
	0	Little-Endian 格式
	1	Big-Endian 格式
1	写入输出数据	
	0	仅通过系统变量写入
	1	仅通过 CC-Binding 写入
3	输出槽生命符号报警	
	0	输出槽生命符号报警
	1	抑制槽生命符号报警

14.3.3 读/写

14.3.3.1 系统变量

输入数据

系统变量	类型	含义
$A_DPB_IN[<n>,<b>]	8 位，无符号（unsigned）	读取一个数据字节（8 位）
$A_DPW_IN[<n>,<b>]	16 位，无符号（unsigned）	读取一个数据字（16 位）
$A_DPSB_IN[<n>,<b>]	8 位，有符号（signed）	读取一个数据字节（8 位）
$A_DPSW_IN[<n>,<b>]	16 位，有符号（signed）	读取一个数据字（16 位）
$A_DPSD_IN[<n>,<b>]	32 位，有符号（signed）	读取一个数据双字（32 位）
$A_DPR_IN[<n>,<b>]	32 位实数	读取输入数据（32 位实数）

<n> = 输入区域 1, 2, ... m；<b> = 输入区域内的字节下标：0, 1, ...（长度 - 1）

输出数据

系统变量	类型	含义
$A_DPB_OUT[<n>,<b>]	8 位，无符号（unsigned）	写入一个数据字节（8 位）
$A_DPW_OUT[<n>,<b>]	16 位，无符号（unsigned）	写入一个数据字（16 位）
$A_DPSB_OUT[<n>,<b>]	8 位，有符号（signed）	写入一个数据字节（8 位）
$A_DPSW_OUT[<n>,<b>]	16 位，有符号（signed）	写入一个数据字（16 位）
$A_DPSD_OUT[<n>,<b>]	32 位，有符号（signed）	写入一个数据双字（32 位）
$A_DPR_OUT[<n>,<b>]	32 位实数	写入输出数据（32 位实数）

<n> = 输出区域 1, 2, ... m；<b> = 输出区域内的字节下标：0, 1, ...（长度 - 1）

配置和设置的用于零件程序/同步动作的 I/O 区域

每个系统变量都是一个 32 位位域。每个位指定给一个 I/O 区域。

位 <n> ≙ 机床数据下标 <n> ≙ I/O 区域 <n+1>

位 <n> == 1 ⇒ I/O 区域 <n+1> 已配置/设置。

系统变量	类型	含义
$A_DP_IN_CONF	32 位位域	读取所有配置的输入区域
$A_DP_OUT_CONF	32 位位域	读取所有配置的输出区域

用于零件程序/同步动作的有效 I/O 区域

每个系统变量都是一个 32 位位域。每个位指定给一个 I/O 区域。

位 <n> ≙ 机床数据下标 <n> ≙ I/O 区域 <n+1>

位 <n> == 1 ⇒ I/O 区域 <n+1> 有效。可通过零件程序/同步动作进行读写。

系统变量	类型	含义
$A_DP_IN_VALID	32 位位域	读取所有有效的输入区域
$A_DP_OUT_VALID	32 位位域	读取所有有效的输出区域

I/O 区域的状态

通过下列系统变量可以读取 I/O 区域的状态。

系统变量	类型	含义
$A_DP_IN_STATE[<n>]	INT	读取输入区域的状态
$A_DP_OUT_STATE[<n>]		读取输出区域的状态
<n> = I/O 区域的下标		
状态 0：I/O 区域未经过配置 1：I/O 区域尚且无法激活 2：I/O 区域可用 3：I/O 区域区域当前不可用		

I/O 区域的长度

通过下列系统变量可以读取 I/O 区域配置的长度。

系统变量	含义
$A_DP_IN_LENGTH[<n>]	读取输入数据区域的长度
$A_DP_OUT_LENGTH[<n>]	读取输出数据区域的长度
<n> = I/O 区域的下标	

边界条件

- 从零件程序中读取/写入时，通道中会触发预处理停止。
- 通过 <b>（区域偏移）可指定在一个 I/O 区域中从哪个位置开始访问数据（字节偏移）。数据类型可在 I/O 区域内的任意字节偏移上读/写。若读/写访问超出了对应 I/O 区域的界限，则会被系统拒绝并发出报警（17030）。
- 通过机床数据 MD10502 $MN_DPIO_RANGE_ATTRIBUTE_IN 和 MD10512 $MN_DPIO_RANGE_ATTRIBUTE_OUT（参见“参数设置 (页 886)”），可针对单独的读/写方向或任意一个 I/O 区域确定系统变量 $A_DPx_IN[<n>,<b>] 或 $A_DPx_OUT[<n>,<b>] 的表示方式 (Little-/Big-Endian)。

14.3.4 边界条件

并行写入 NC 和 PLC

通过 NC 或从 PLC 用户程序中**并行写入输出**会导致输出值意外相互改写。因此，该应用是**不允许的**，但在控制系统侧可能无法避免。

生命符号监控

在插补周期开头，系统会为每个 I/O 区域检查对应的槽或 I/O 区域的生命符号是否故障。如果出现故障，系统会显示报警 9050 或 9052。

影响：

- 零件程序执行**不会**停止
- 恢复生命符号后，系统会自动删除报警。

14.3.5 示例

14.3.5.1 写入 NC 输入/输出

前提条件

有效配置已载入 PLC 中。

零件程序/同步动作的参数设置

设定

- 第 6 个数据组的参数设置：机床数据/系统变量下标 = 5
- 配置数据：
  - 逻辑初始地址 = 334
  - 槽长度 = 8 个字节
- 显示：Little-Endian 格式

机床数据中的参数设置

- MD10510 $MN_DPIO_LOGIC_ADDRESS_OUT[5] = 334（逻辑初始地址）
- MD10511 $MN_DPIO_LENGTH_OUT[5] = 8（I/O 区域的长度，单位：字节）
- MD10512 $MN_DPIO_ATTRIBUTE_OUT[5]
  - 位 0 = 0（Little-Endian 格式）
  - 位 1 = 0（仅通过系统变量写入）
  - 位 3 = 0（输出槽生命符号报警）

示例

程序代码	注释
`$A_DPB_OUT[5,6]=128`	`; 字节 ≙ 8 位，下标 = 5，偏移 = 6`
`$A_DPW_OUT[5,5]='B0110'`	`; 单字 ≙ 16 位，下标 = 5，偏移 = 5`
	`; 注意：偏移 6 上的数据被改写`
`$A_DPSD_OUT[5,3]='H8F'`	`; 双字 ≙ 32 位，下标 = 5，偏移 = 3`
	`; 注意：偏移 4、5、6 上的数据被改写`
`$AC_MARKER[0]=5`	`; 下标 = 5`
`$AC_MARKER[1]=3`	`; 偏移 = 3`
`$A_DPSD_OUT[$AC_MARKER[0],$AC_MARKER[1]]='H8F'`	`; 间接定址`
	`; 双字 ≙ 32 位，下标 = 5，偏移 = 3`
`R1 = $A_DPB_OUT[5,6]`	`; 给 R 参数赋值，字节 ≙ 8 位，下标 = 5，偏移 = 6`
	`; 结果：R1 == 'HFF'`

程序代码	注释
ID=1 WHENEVER TRUE DO $A_DPB_OUT[5,0]=123	；每个插补周期循环写入
	；字节 ≙ 8 位，下标 = 5，偏移 = 0
；编程错误	
$A_DPB_OUT[5.255]=128	；⇒ 报警 17030：偏移 255 > I/O 区域
$A_DPB_OUT[6.10]=128	；⇒ 报警 17020：下标 6 预留给编译循环（见下）
$A_DPB_OUT[7.10]=128	；⇒ 报警 17020：机床数据中未定义下标 7
$A_DPB_OUT[16.6]=128	；⇒ 报警 17020：下标 16 超出了取值范围

通过编译循环编程的配置

设定

- 第 7 个数据组的参数设置：机床数据/系统变量下标 = 6
- 配置数据：
  - 逻辑初始地址 = 444
  - 槽长度 = 10 个字节
- 显示：Little-Endian 格式

机床数据中的参数设置

- MD10510 $MN_DPIO_LOGIC_ADDRESS_OUT[6] = 444（逻辑初始地址）
- MD10511 $MN_DPIO_LENGTH_OUT[6] = 0（仅使用第一个有效数据槽）
- MD10512 $MN_DPIO_ATTRIBUTE_OUT[6]
  - 位 0 = 0（Little-Endian 格式）
  - 位 1 = 1（仅通过 CC-Binding 写入）
  - 位 3 = 1（抑制槽生命符号报警）

14.3.5.2 读取 NC 输入/输出

前提条件

有效配置已载入 PLC 中。

零件程序/同步动作的参数设置

设定

- 第 1 个数据组的参数设置：机床数据/系统变量下标 = 0
- 配置数据：
  - 逻辑初始地址 = 456
  - 槽长度 = 32 个字节
- 显示：Big-Endian 格式

机床数据中的参数设置

- MD10500 $MN_DPIO_LOGIC_ADDRESS_IN[0] = 456（逻辑初始地址）
- MD10501 $MN_DPIO_LENGTH_IN[0] = 32（I/O 区域的长度，单位：字节）
- MD1050ß2 $MN_DPIO_ATTRIBUTE_IN[0]
  - 位 0 = 1（Big-Endian 格式）
  - 位 2 = 0（可通过系统变量和 CC-Binding 读取）
  - 位 3 = 0（输出槽生命符号报警）

示例

程序代码	注释
$AC_MARKER[0]=$A_DPW_IN[0,0]	; 字节 ≙ 8 位，下标 = 0，偏移 = 0
$AC_MARKER[1]=$A_DPSD_IN[0,1]	; 有符号双字 ≙ 32 位，下标 = 0，偏移 = 1
$AC_MARKER[1]=$A_DPSD_IN[0,8]	; 有符号双字 ≙ 32 位，下标 = 0，偏移 = 8
$AC_MARKER[2]=0	; 下标 = 0
$AC_MARKER[3]=8	; 偏移 = 8
$AC_MARKER[1]=$A_DPSD_IN[$AC_MARKER[2],$AC_MARKER[3]]	; 间接定址
	; 有符号双字 ≙ 32 位，下标 = 0，偏移 = 8
ID=2 WHEN $A_DPB_IN[0,11]>=5 DO $AC_MARKER[2]=$A_DPSD_IN[0,8]	
	; IF 下标 0，偏移 11 >= 5
	; THEN 有符号双字 ≙ 32 位，下标 = 0，偏移 = 8
; 编程错误	
R1=$A_DPB_IN[0,255]	; ⇒ 报警 17030：偏移 255 > I/O 区域
R1=$A_DPB_IN[2.6]	; ⇒ 报警 17020：机床数据中未定义下标 2
R1=$A_DPB_IN[1.10]	; ⇒ 报警 17020：下标 1 预留给编译循环（见下）
R1=$A_DPB_IN[16.6]	; ⇒ 报警 17020：下标 16 超出了取值范围

通过编译循环编程的参数设置

设定

- 第 2 个数据组的参数设置：机床数据/系统变量下标 = 1
- 配置数据：
  - 逻辑初始地址 = 312
  - 槽长度 = 32 个字节
- 显示：Big-Endian 格式

机床数据中的参数设置

- MD10500 $MN_DPIO_LOGIC_ADDRESS_IN[01] = 312 （逻辑初始地址）
- MD10501 $MN_DPIO_LENGTH_IN[1] = 32（I/O 区域的长度，单位：字节）
- MD10502 $MN_DPIO_ATTRIBUTE_IN[1]
  - 位 0 = 1（Big-Endian 格式）
  - 位 2 = 1（只可通过 CC-Binding 读取）
  - 位 3 = 1（抑制槽生命符号报警）

14.3.5.3 通过状态查询写入 NC 输入/输出

前提条件

有效配置已载入 PLC 中。

零件程序/同步动作的参数设置

设定

- 第 6 个数据组的参数设置：机床数据/系统变量下标 = 5
- 配置数据：
  - 逻辑初始地址 = 1200
  - 槽长度 = 32 个字节
- 显示：Big-Endian 格式

机床数据中的参数设置

- MD10510 $MN_DPIO_LOGIC_ADDRESS_OUT[5] = 1200（逻辑初始地址）
- MD10511 $MN_DPIO_LENGTH_OUT[5] = 0（I/O 区域的长度，单位：字节）
- MD10512 $MN_DPIO_ATTRIBUTE_OUT[5]
  - 位 0 = 1（Little-Endian 格式）
  - 位 1 = 0（仅通过系统变量写入）
  - 位 3 = 0（输出槽生命符号报警）

示例

```
check:                                          ; 跳转标记
IF $A_DP_OUT_STATE[5]==2 GOTOF write            ; 数据区域有效时；=> 跳转至 N15
GOTOB check                                     ; 返回 check
write:                                          ; 跳转标记
$A_DPB_OUT[5,6]=128                             ; 写入数据字节
```

```
check:                                          ; 跳转标记
IF $A_DP_OUT_CONF==$A_DP_OUT_VALID GOTOF write  ; 数据区域有效时
                                                ; => 跳转至 N15
SETAL(61000)                                    ; 设置报警号 61000
write:                                          ; 跳转标记
$A_DPB_OUT[5,6]=128                             ; 写入数据字节
```

```
check:                                          ; 跳转标记
IF $A_DP_OUT_VALID B_AND 'B100000' GOTOF write  ; 数据区域有效时
                                                ; => 跳转至 N15
SETAL(61000)                                    ; 设置报警号 61000
write:                                          ; 跳转标记
$A_DPB_OUT[5,6]=128                             ; 写入数据字节
```

```
R1=$A_DP_OUT_LENGTH[5]          ; I/O 区域（槽）的长度，以字节为单位
                                ; 结果：R1 = 32
```

14.4 数据表

14.4.1 机床数据

14.4.1.1 通用机床数据

编号	名称：$MN_	说明
10300	FASTIO_ANA_NUM_INPUTS	有效 NC 模拟量输入的数量
10310	FASTIO_ANA_NUM_OUTPUTS	有效 NC 模拟量输出的数量
10320	FASTIO_ANA_INPUT_WEIGHT	NC 模拟量输入的权重系数
10330	FASTIO_ANA_OUTPUT_WEIGHT	NC 模拟量输出的权重系数
10350	FASTIO_DIG_NUM_INPUTS	有效 NC 数字量输入字节的数量
10360	FASTIO_DIG_NUM_OUTPUTS	有效 NC 数字量输出字节的数量
10362	HW_ASSIGN_ANA_FASTIN	外部 NC 模拟量输入的硬件指定
10364	HW_ASSIGN_ANA_FASTOUT	外部 NC 模拟量输出的硬件指定
10366	HW_ASSIGN_DIG_FASTIN	外部 NC 数字量输入的硬件指定
10368	HW_ASSIGN_DIG_FASTOUT	外部 NC 数字量输出的硬件指定
10500	DPIO_LOGIC_ADDRESS_IN	PROFIBUS I/O 区域的逻辑起始地址
10501	DPIO_RANGE_LENGTH_IN	PROFIBUS I/O 区域的长度
10502	DPIO_RANGE_ATTRIBUTE_IN	PROFIBUS I/O 的属性
10510	DPIO_LOGIC_ADDRESS_OUT	PROFIBUS I/O 区域的逻辑起始地址
10511	DPIO_RANGE_LENGTH_OUT	PROFIBUS I/O 区域的长度
10512	DPIO_RANGE_ATTRIBUTE_OUT	PROFIBUS I/O 的属性
10530	COMPAR_ASSIGN_ANA_INPUT_1	针对比较器字节 1 的 NC 模拟量输入的硬件指定
10531	COMPAR_ASSIGN_ANA_INPUT_2	针对比较器字节 2 的 NC 模拟量输入的硬件指定
10540	COMPAR_TYPE_1	比较器字节 1 的设置
10541	COMPAR_TYPE_2	比较器字节 2 的设置

14.4.1.2 通道专用机床数据

编号	名称：$MC_	描述
21220	MULTFEED_ASSIGN_FASTIN	“在一个程序段内编写多个进给率值”功能的 NC I/O 输入字节指定

14.4.2 设定数据

14.4.2.1 通用设定数据

编号	名称：$SN_	描述
41600	COMPAR_THRESHOLD_1	比较器字节 1 的阈值
41601	COMPAR_THRESHOLD_2	比较器字节 2 的阈值

14.4.3 系统变量

名称	说明
$A_IN	NC 数字量输入
$A_OUT	NC 数字量输出
$A_INA	NC 模拟量输入
$A_OUTA	NC 模拟量输出
$A_DPB_IN	读取一个数据字节（8 位）
$A_DPW_IN	读取一个数据字（16 位）
$A_DPSB_IN	读取一个数据字节（8 位）
$A_DPSW_IN	读取一个数据字（16 位）
$A_DPSD_IN	读取一个数据双字（32 位）
$A_DPR_IN	读取输入数据（32 位实数）
$A_DPB_OUT	写入一个数据字节（8 位）
$A_DPW_OUT	写入一个数据字（16 位）
$A_DPSB_OUT	写入一个数据字节（8 位）

名称	说明
$A_DPSW_OUT	写入一个数据字（16 位）
$A_DPSD_OUT	写入一个数据双字（32 位）
$A_DP_IN_VALID	读取所有有效的输入区域
$A_DP_OUT_VALID	读取所有有效的输出区域
$A_DP_IN_STATE[<n>]	读取输入区域的状态
$A_DP_OUT_STATE[<n>]	读取输出区域的状态
$A_DP_IN_LENGTH[<n>]	读取输入数据区域的长度
$A_DP_OUT_LENGTH[<n>]	读取输出数据区域的长度

15 存储器配置

15.1 引言

存储区

NCU 上有两个存储区，用于存储和管理**本地**持久性及非持久性 NC 数据。

- **静态 NC 存储器**
  静态 NC 存储器包含主动和被动文件系统 (页 899)的**持久性** NC 数据。持久性数据在控制系统因断电或掉电而重新上电后仍会保持其最后写入的数值。
- **动态 NC 存储器**
  动态 NC 存储器包含由 NC 动态生成、**非持久性**或易失性的 NC 数据，例如：宏、本地用户数据、中间缓存器等。非持久性数据在控制系统因断电或掉电而重新上电后不再存在。

确定性

为了确保 NC 在所有加工情形下的确定性，本地静态和动态 NC 存储器的所有存储区空间均可设置，但为固定大小，即在控制系统运行期间空间大小不变。

存储器配置

系统首次启动时，本地静态和动态 NC 存储器的存储区会基于存储器配置机床数据的缺省值设置。通常情形下这些设置足够使用。

重新配置

根据需要，可依照用户特定需求调整存储器配置。为此所需的存储器重新配置可能会导致静态 NC 存储器的所有用户数据丢失。因此，在修改过存储器配置机床数据后，系统会显示报警 4400：“机床数据的修改使得缓冲存储器重组（数据丢失！）”。在通过 NC 复位激活机床数据更改或重新配置存储器前，应先创建一个调试存档。

经过修改的存储器配置会在下一次 NC 启动时生效。然后可以读取创建的调试存档，以恢复用户数据。

15.2 主动和被动文件系统

本地静态 NC 存储器中保存了 NC 的用户专有数据。 该存储区中包含主动和被动文件系统的数据。

主动文件系统

主动文件系统包含了用于设置 NC 的系统数据。 其中主要包括：

- 机床数据
- 设定数据
- 选件数据
- 全局用户数据(GUD)
- 刀具补偿数据/刀库数据
- 保护区
- R 参数
- 零点偏移/FRAME
- 垂度补偿
- 象限误差补偿
- 丝杠螺距误差补偿

主动文件系统的数据表示的是 NC 的当前工作数据。

对主动文件系统的视角为数据趋向。

被动文件系统

被动文件系统包含所有从本地载入至 NC 的文件：

- 主程序
- 子程序
- 工件
- 全局用户数据和宏指令的定义文件(*.DEF)
- 标准循环
- 用户循环
- 注释
- 二进制文件（例如图片、PDF 文档）

对被动文件系统的视角为文件趋向。

参见

边界条件：持久性用户数据 (页 908)

15.3 调试

15.3.1 配置

本地静态和动态 NC 存储器的配置可通过以下机床数据设置或调整：

- 存储器配置机床数据：
  - $MN_**MM**_...（NC 专用存储器配置机床数据）
  - $MC_**MM**_...（通道专用存储器配置机床数据）
  - $MA_**MM**_...（轴专用存储器配置机床数据）
- 参数设置的通道的数量：
  - MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP（运行方式组中生效的通道）

15.3.2 重新配置

控制系统首次启动时，本地 NC 存储器的存储区会基于存储器配置机床数据的缺省值进行设置。通常情形下这些设置足够运行控制系统。如需根据用户的特定需求调整本地 NC 存储器的存储区，则会引起本地 NC 存储器的重新配置。如果此时涉及**静态** NC 存储器的主动或被动文件系统，系统会显示报警 4400。用户应通过创建调试存档将相关数据备份到一个外部存储器上。

下列情况下会显示报警 4400：

- 被动文件系统的存储器配置机床数据被更改
- 主动文件系统的存储器配置机床数据被更改且功能“自动存储器重配置” (AMR) **未生效**
- 主动文件系统的存储器配置机床数据被更改，AMR 功能**生效**，但主动文件系统的数据无法保存到本地

注意
重新配置造成的数据丢失 重新配置本地**静态** NC 存储器可导致主动和/或被动文件系统中的用户数据丢失。因此，在通过 NC 复位激活经过修改的存储器配置**前**，显示报警 4400 后强烈建议创建一个批量调试文件备份所有相关数据。

“自动存储器重配置”（Automatic Memory Reconfiguration，AMR）功能

主动文件系统

AMR 功能用于重新配置主动文件系统 (页 899)的存储区，同时创建调试存档并读取，防止用户数据丢失。

如果该功能生效，那么在对存储器配置机床数据的修改涉及主动文件系统时，控制系统会先检查能否将主动文件系统的所有数据放入本地剪贴板。若可以，那么在通过热启动激活修改的存储器配置时，控制系统会将主动文件系统的数据写入本地中间缓冲器。之后，系统会重新配置主动文件系统的存储器。之后重新读取新配置的主动文件系统中的缓冲数据。

如果无法缓冲存储主动文件系统中的数据，系统会显示报警 4400。在激活机床数据更改前，应创建一个含相关用户数据的调试存档。参见下面的“更多信息”。

被动文件系统

如果重新配置被动文件系统所需的机床数据被修改，无论 AMR 功能是否激活，系统始终会显示报警 4400。在激活机床数据更改前，应创建一个含相关用户数据的调试存档。

如果重新配置主动文件系统所需的机床数据被修改，系统会保留被动文件系统中的数据。

激活

“自动存储器重配置”（AMR）功能通过以下机床数据激活：

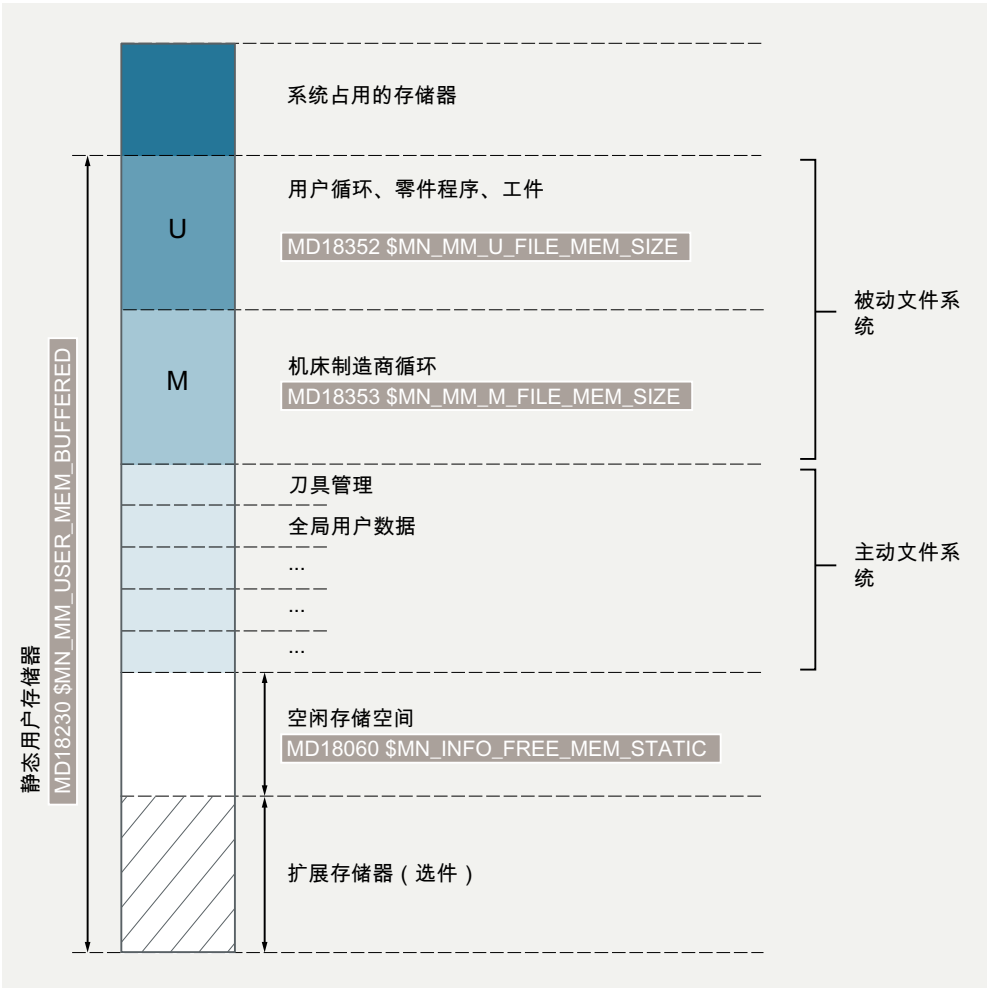
MD17950 $MN_IS_AUTOMATIC_MEM_RECONFIG = TRUE

15.4 配置静态用户存储器

15.4.1 静态 NC 存储器的结构

静态 NC 存储器可供系统和用户一同使用。供用户使用的存储区域称为“静态用户存储器”。它包含了主动和被动文件系统的数据。

下图展示了静态 NC 存储器的结构：



静态用户存储器容量

静态用户存储器的容量在以下机床数据中显示：

MD18230 $MN_MM_USER_MEM_BUFFERED

静态用户存储器的组成部分

被动文件系统

被动文件系统的下列分区位于静态用户存储器中：

分区	存有：
U（**U**ser = 用户）	• _N_CUS_DIR（用户循环）目录下的文件 • 零件程序 • 工件
M（**M**anufacturer = 机床制造商）	_N_CMA_DIR（机床制造商循环）目录下的文件

MD18060 $MN_INFO_FREE_MEM_STATIC 中显示的可用被动文件系统自由存储空间可任意划归 **U** 和 **M** 分区。这通过以下机床数据设置：

- MD18352 $MN_MM_U_FILE_MEM_SIZE = <用户数据的存储容量>
- MD18353 $MN_MM_M_FILE_MEM_SIZE = <机床制造商数据的存储容量>

说明

分区 S

被动文件系统的分区 S（**西**门子 = 系统制造商）位于动态 NC 存储器 (页 905)中。

主动文件系统

主动文件系统的存储器划分为不同区域（刀具管理、全局用户数据等）。通过存储器配置机床数据（...**MM_**...）可为各区域设置存储容量。

空闲的静态用户存储器

空闲的静态用户存储器空间显示在以下机床数据中：

MD18060 $MN_INFO_FREE_MEM_STATIC

说明

在操作界面上的“调试”操作区中会显示修改存储区所需占用的空间。借助该数据调试人员就可以了解计划的修改实际需要占用多少存储器。

扩展存储器（选件）

需要额外的静态用户存储空间时，可购买存储器扩展选件。

额外空间可根据需要用于扩大 U 分区和/或 M 分区，或用于扩大主动文件系统的存储区。

15.4.2 调试

用户可根据需求扩大/缩小各个存储区来修改默认的存储器分区。

基本步骤：

1. 载入标准机床数据。
   更多信息：
   CNC 调试手册：NC、PLC、驱动；调试前提条件
2. 测定静态用户存储器的最大容量（包括存储器扩展选件）：
   MD18230 $MN_MM_USER_MEM_BUFFERED
3. 可选：更改静态用户存储器的大小：
   – MD19250 $ON_USER_MEM_BUFFERED
   – 执行上电复位
4. 可选：设置 U 分区和 M 分区的大小：
   – MD18352 $MN_MM_U_FILE_MEM_SIZE
   – MD18353 $MN_MM_M_FILE_MEM_SIZE
5. 可选：设置控制系统的附加通道：
   MD10010 $MC_ASSIGN_CHAN_TO_MODE_GROUP
6. 可选：设置主动文件系统的存储区（刀具管理、全局用户数据等）的大小：
   – 确定空闲的静态用户存储器空间：
     MD18060 $MN_INFO_FREE_MEM_STATIC
   – 通过存储器配置机床数据（...MM_...）设置主动文件系统存储区的容量。
7. 执行上电复位。
   下一次启动控制系统时，存储器会被重新配置。

更多信息

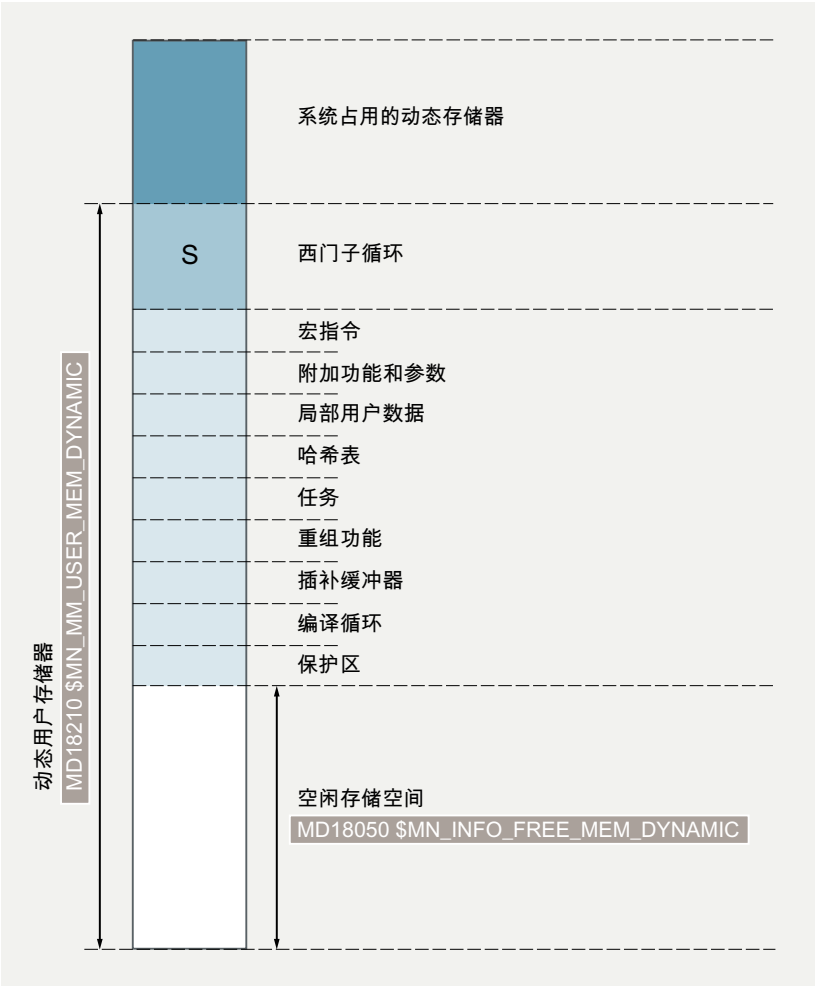
- 参数手册之机床数据和参数

15.5 配置动态用户存储器

15.5.1 动态 NC 存储器的结构

动态 NC 存储器可供系统和用户一同使用。供用户使用的存储区域称为“动态用户存储器”。

下图展示的是动态 NC 存储器的结构：



动态用户存储器的容量

动态用户存储器的容量显示在以下机床数据中：

MD18210 $MN_MM_USER_MEM_DYNAMIC

动态用户存储器的组成部分

被动文件系统

被动文件系统的分区 S 位于动态用户存储器中：

分区	存有：
S（**西**门子 = 系统制造商）	_N_CST_DIR（西门子循环）目录下的文件

分区 **S** 的大小是预先设置好的，不可更改。

用户数据区

动态用户存储器包含由 NC 动态生成、非持久性的 NC 数据，例如：宏、本地用户数据、中间缓存器等。

空闲的动态用户存储器

空闲的动态用户存储器空间显示在以下机床数据中：

MD18050 $MN_INFO_FREE_MEM_DYNAMIC

该机床数据的值指出了每条通道还剩余多少存储器空间可用于扩大用户数据区。

15.5.2 调试

用户可根据需求扩大/缩小各个存储区来修改默认的存储器分区。

基本步骤：

1. 确定空闲的动态用户存储器空间大小：
   MD18050 $MN_INFO_FREE_MEM_DYNAMIC
2. 可选：通过存储器配置机床数据（...MM_...）设置动态用户存储器的存储区的大小：
3. 执行上电复位。
   下一次启动控制系统时，存储器会被重新配置。

更多信息参见参数手册之机床数据和参数。

15.6 边界条件：通用

15.6.1 通道数和刀架数之间的关联

增加通道数时，也要相应地增加刀架数。

请根据以下规定来设置系统内提供的刀架数：

MD18088 $MN_NUM_TOOL_CARRIER =
<**一条通道**中的最大刀架数> *
<设置的通道的数量>

系统内部会将刀架均匀地分给现有通道。

示例

系统中设置了 3 条通道。通道中需要使用的刀架数是：

1. 通道：3 ⇒ 最大值
2. 通道：2
3. 通道：1

MD18088 = <一条通道中的最大刀架数> * 通道数= 3 * 3 = 9

因此，系统内部会分给每条通道 3 个刀架。

15.6.2 事后减少通道数和/或轴数

通道数和/或轴数一旦提高，便会在系统热启动后生效。需要从系统生成一份调试存档时，不允许撤销该修改。

将一个通道数/轴数再次减少的调试存档载入系统时，系统会报警，中断载入。

NC 动态存储器和静态存储器之间的区别

- **动态**存储器同时提高**通道数**和**轴数**才会占用更多存储空间。
- **静态**存储器单独提高**通道数**就会占用更多存储空间。

15.7 边界条件：持久性用户数据

“内部”和“外部”静态存储器

主动和被动文件系统 (页 899)的持久性用户数据保存在控制系统模块的静态存储器中。该存储器分为一个“内部”存储器和一个“外部”存储器：

- “内部”静态存储器由控制系统模块上的存储器单元组成。
- “外部”静态存储器目前是由插入在控制系统模块上插槽中的存储卡构成。

“外部”静态存储器的空间远远大于“内部”静态存储器。

运行时间特性

持久性用户数据在控制系统启动时从“外部”静态存储器载入到动态工作存储器中，以用于之后的加工，例如：零件程序、循环或同步动作。如果要写入一个持久性数据，则必须将其回写至“外部”静态存储器中，以确保其持久保存。

“外部”静态存储器的优点和缺点

“外部”静态存储器的优点是存储空间大小更为灵活。但相对于内部存储器，它的缺点是写数据的操作耗时明显更长。此外，写访问的次数还会影响存储介质的寿命。

中间缓存器

为了避免在每次写入持久性数据时都直接写入到“外部”静态存储器中，在 NC 的“内部”静态存储器中设置了中间缓存器。在该缓存器中保存写入的持久性数据。如果中间缓存器达到了所定义的存储容量，所有数据会原样回写到“外部”静态存储器中。

如果在回写前关闭了控制系统或发生掉电，那么数据会在下次启动控制系统时进行回写。这样便保证了数据一致性。

中间缓存器除了具有上述优点外，也有以下功能限制：

- 溢出
  持久性数据的循环写入，特别是在同步动作的主运行过程中，可能会导致中间缓存器溢出。也就是说，新数据无法再保存到中间缓存器中，因为已有数据还未回写到“外部”静态存储器中。
- 时间延迟
  从中间缓存器同步回写至“外部”静态存储器时，程序预处理在写入期间会停止。因为在此期间不会再预处理 NC 程序段，这会导致加工时间延长或者轨迹控制运行中断。

15.7.1 使用易失性数据替代持久性数据

易失性数据替代持久性数据

基于上一章节中说明的功能限制，在进行零件程序、循环和同步动作的编程时，**强烈**建议检查每一个待写入的变量，确认其是否需要进行持久性保存。

如果上一个写入的变量值在控制系统重启后不再需要用于下一个执行过程，那么该数据属于易失性存储类别。因此不应将该数据作为持久性用户数据使用，例如：R 参数。

非持久性系统变量

参数（默认为非持久性）

- `$AC_PARAM[<n>]` **通道专用**字段变量，实型

说明

检查持久性

如果将相应的机床数据置位，那么对应的系统变量 $AC_PARAM[<**n**>] 便是持久性数据。
MD28255 $MC_MM_BUFFERED_AC_PARAM[<**n**>] = 1

指定通道中的参数数量可通过以下机床数据设置：

MD28254 $MC_MM_NUM_AC_PARAM

标记（默认为非持久性）

- `$AC_MARKER[<n>]` **通道专用**字段变量，整型

说明

检查持久性

如果将相应的机床数据置位，那么对应的系统变量 $AC_MARKER[<**n**>] 便是持久性数据。
MD28257 $MC_MM_BUFFERED_AC_MARKER[<**n**>] = 1

指定通道中的标记数量可通过以下机床数据设置：

MD2826 $MC_MM_NUM_AC_MARKER

NCU-Link 变量

如不使用"NCU-Link"功能，则可任意使用 NCU-Link 变量。

- `$A_DLB[<n>]` Link 变量，字节型
- `$A_DLW[<n>]` Link 变量，单字型
- `$A_DLD[<n>]` Link 变量，整型
- `$A_DLR[<n>]` Link 变量，实型

非持久性用户变量

非持久性用户变量在 NC 程序的定义区域中使用关键字 `DEF` 进行定义。

- 程序局部用户变量（LUD）
- 程序全局用户变量（PUD）

更多信息

编程手册之 NC 编程

章节：工作准备 > 灵活的 NC 编程 > 变量 > 定义用户变量（DEF）

15.7.2 机床数据

15.7.2.1 概述

通过以下机床数据可以设置持久性属性：

机床数据		含义
MD17610	$MN_DEPTH_OF_LOGFILE_OPT_PF	中间缓存器的查找深度 (页 911)
MD18232	$MN_MM_ACTFILESYS_LOG_FILE_MEM	中间缓存器的大小 (页 912)
MD18233	$MN_MM_IS_CONTINOUS_DATA_SAVE_ON	持久性数据的连续备份 (页 913)
MD18234	$MN_MM_MEMORY_CONFIG_MASK	持久性数据的备份方式 (页 914)

15.7.2.2 MD17610:中间缓存器的查找深度

通过该机床数据可对中间缓存器的各缓存区中的查找深度进行设置。

中间缓存器含有三个不同的缓存区：

1. 在**预处理**时写入的持久性数据区
2. 在**换刀**和**回退**的**主运行**过程中写入的持久性数据区
3. 在**同步动作**的**主运行**过程中写入的持久性数据区

如要将一个持久性数据写入中间缓存器，则会根据作为查找深度所指定的已有数据记录的数量进行查找，确认该数据是否已在中间缓存器中创建。如果未找到该数据，则会新建该数据。如果找到了该数据，则只会接收新值。

MD17610 $MN_DEPTH_OF_LOGFILE_OPT_PF[<**下标**>] = <**值**>

下标	**含义** 持久性数据缓存区用于以下过程时写入：
0	预处理
1	**换刀**和**回退**的主运行过程
2	**同步动作**的主运行过程

值	含义
0	未在中间缓存器中找到的数据会立即新建。 影响： • 时间较快，但存储空间消耗大 • 缓存器很快占满，因此需要频繁回写至“外部”静态存储器中。因此，对“外部”静态存储器的写访问次数增加。
> 0	根据指定的值，在中间缓存器中的已有数据记录里查找数据。 • 找到：仅接收新值 • 未找到：新建数据 影响： • 时间较慢，但节省存储空间 • 缓存器较慢占满，因此不需要频繁回写至“外部”静态存储器中。因此，对“外部”静态存储器的写访问次数减少。

通过以下机床数据设置各个中间缓存器的大小：

MD18232 $MN_MM_ACTFILESYS_LOG_FILE_MEM (页 912)

15.7.2.3 MD18232:中间缓存器的大小

中间缓存器含有三个不同的缓存区：

1. 在**预处理**时写入数据的中间缓存器
2. 在**换刀**和**回退**的**主运行**过程中写入数据的中间缓存器
3. 在同步动作的**主运行**过程中写入数据的中间缓存器

通过机床数据设置各个缓存区的大小，单位：kB。1000 条数据记录大约需要 80 kB。

MD18232 $MN_MM_ACTFILESYS_LOG_FILE_MEM[<**下标**>] = <**值**>

下标	**含义**：用于以下数据的中间缓存器
0	在**预处理**时写入的数据
1	在**换刀**和**回退**的**主运行**过程中写入的数据
2	在**同步动作**的**主运行**过程中写入的数据

值	含义
0	缓存器大小为 0，无法为相应数据提供持久性存储
> 0	缓存器大小，单位：kB

缓存器溢出

如果持久性用户数据写入中间缓存器的速度比回写至“外部”静态存储器的速度更快，就会导致缓存器溢出。自该时间点起，如发生掉电，会导致数据丢失。

中间缓存器溢出会触发报警 15120“如现在发生掉电：上次更改的数据会丢失；缓存器大小 = %1”。

说明

报警抑制

当以下机床数据置位时，则会不显示报警 15120：
MD11415 $MN_SUPPRESS_ALARM_MASK_2，位 3 = 1

15.7.2.4 MD18233:持久性数据的连续备份

通过该机床数据可在具体的文件系统中启用或关闭持久性数据的连续备份。

连续备份可在**诊断**（检查对系统性能的影响）时暂时关闭。这样在断电后便无法再保证持久性用户数据的一致性。除非，控制系统的运行环境已排除了发生意外关机或掉电的可能。

MD18233 $MN_MM_IS_CONTINOUS_DATA_SAVE_ON[<**下标**>] = <**值**>

下标	**含义**
0	保留
1	**主动**文件系统
2	**被动**文件系统

值	**含义** 持久性用户数据的连续备份：
FALSE	关闭
TRUE	启用

15.7.2.5 MD18234:持久性数据的备份方式

通过该机床数据可设置持久性数据的备份方式。

- 同步
  如果中间缓存器即将占满，则会停止进行预处理以及程序段处理。中间缓存器的数据将回写至“外部”静态存储器中。当所有数据都回写完成后，才会继续执行预处理。
- 异步
  如果中间缓存器占满约一半时，会在进行程序段处理的同时将中间缓存器的数据回写至“外部”静态存储器中。
  优点：在向“外部”静态存储器写入数据时，不会停止执行预处理。这样通常便不会中断零件程序的执行。
  缺点：由于除主运行过程外，并未停止的预处理过程还会继续向中间缓存器中写入数据，此时回写已经自中间缓存器约占满一半时开始执行。因此，对“外部”静态存储器的写入操作次数几乎是翻倍的。

MD18234 $MN_MM_MEMORY_CONFIG_MASK = <**值**>

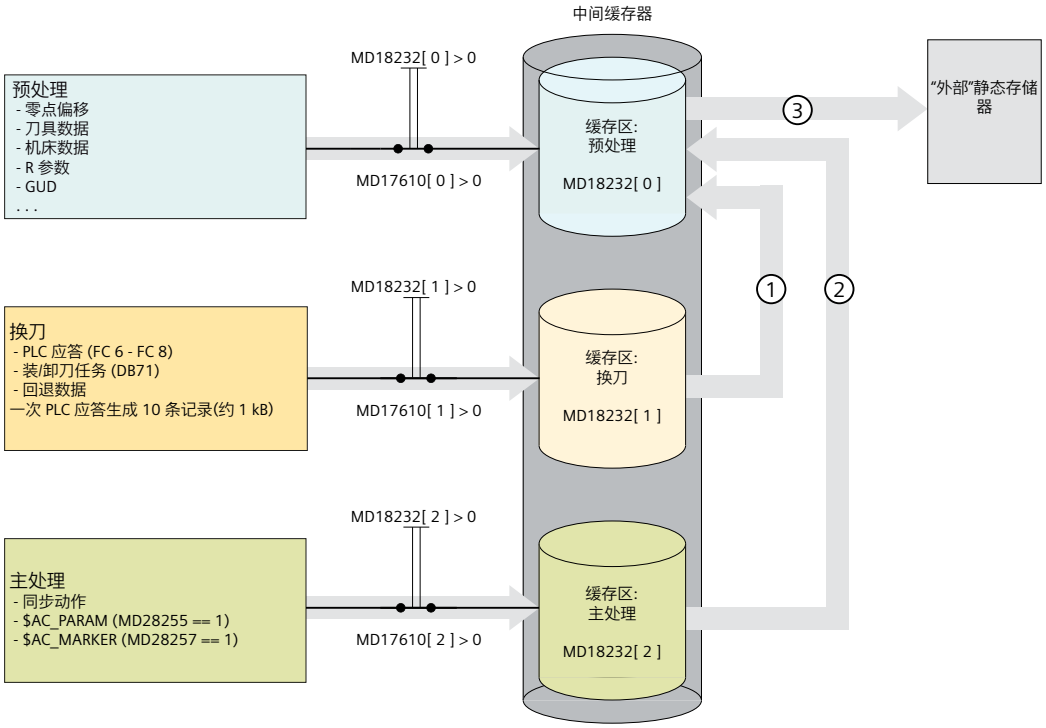
值	**含义** 中间缓存器的回写方式：
0	同步
1	异步

15.7.3 优化方式

下面将介绍与数据持久性的不同次级属性相关的优化方式。由于这与各个应用场景有很大的相关性，因此优化只能分步骤进行。

概述

下图为 NC 中持久性数据存储过程的示意图。



① 每次开始预处理时，数据都会复制到预处理区域中

②

③ 回写数据：

- 同步 / 异步：当预处理缓存区充分占满时
- 热启动时（NCK 复位）

图 15-1 持久性数据存储

系统变量 $AN_PERSDIAG

通过该系统变量可以查看数据持久性的当前状态。例如，每次从中间缓存器向“外部”静态存储器回写持久性数据时，写操作计数就会增加 1：

- 同步数据存储 (MD18234 == 0) 时
  $AN_PERSDIAG[**0**, **0**] += 1
- 异步数据存储 (MD18234 ==10) 时
  $AN_PERSDIAG[**0**, **20**] += 1

更多信息

参数手册之系统变量

章节：系统变量列表 > 系统数据

15.7.3.1 非持久性数据的使用

如前所述，避免数据持久性出现问题的首选且最高效的方法是，只在工艺上十分必要时，才在 NC 程序、循环和同步动作中使用持久性用户数据。这也就是说，只要对数据持久性没有强制要求，就不使用持久性用户数据。

参见章节“使用易失性数据替代持久性数据 (页 909)”

15.7.3.2 日后关闭同步动作中持久性数据的使用

避免同步动作中的数据持久性出现问题的最高效的方法是，只在工艺上十分必要时，才在同步动作中使用持久性用户数据。如果在同步动作中使用了持久性用户数据，并且不能更改这些同步动作，则可以通过以下措施在日后关闭持久性数据的使用。

在同步动作的主运行过程中写入数据的中间缓存器的参数设置，缓存器大小为 0：

MD18232 (页 912)[2] = 0

说明

缺点

将中间缓存器的大小设为 0 后，会统一将同步动作中所使用的全部数据更改为非持久性数据。可能出现的意外影响应具体情况具体处理。

15.7.3.3 写操作次数

借助系统变量 $AN_PERSDIAG，并根据在 MD18234 (页 914) 中设置的备份方式（同步/异步），可测定向“外部”静态存储器进行的写操作次数：

$AN_PERSDIAG[<**下标_1**>, <**下标_2**>]

下标_1	**下标_2**	**含义**：所有子功能（被动和主动文件系统以及机床数据）的总和：
0	0	所有**同步**写操作的次数
0	20	所有**异步**写操作的次数

如果写操作次数持续快速增加，那么不断写入的持久性用户变量首先会占满中间缓存器，然后会回写至“外部”静态存储器中。

优化方式

以下优化方式可以单独使用或组合使用：

- 从持久性用户数据切换为非持久性用户数据
  参见章节“使用易失性数据替代持久性数据 (页 909)”
- 增大中间缓存器的空间
  参见章节“MD18232:中间缓存器的大小 (页 912)”
- 加深查找深度
  参见章节“MD17610:中间缓存器的查找深度 (页 911)”

15.7.3.4 缓存器溢出

如果持久性用户数据写入中间缓存器的速度比回写至“外部”静态存储器的速度更快，就会导致缓存器溢出。自该时间点起，如发生掉电，会导致持久性用户数据出现数据不一致。

中间缓存器溢出会触发

报警 15120“如现在发生掉电：上次更改的数据会丢失；缓存器大小 = %1”。

说明

报警抑制

当以下机床数据置位时，则会不显示报警 15120：

MD11415 $MN_SUPPRESS_ALARM_MASK_2，位 3 = 1

详细诊断

可通过以下系统变量对详细诊断进行设置，查找发生溢出的中间缓存器区域以及设置诊断频率：

```
$AN_PERSDIAG[<下标 1>, <下标 2>]
```

下标 1	下标 2	含义：自 NC 启动后在以下过程中写入数据时缓存器发生溢出的次数：
2	6	**预处理**
2	7	**换刀**和**回退**的**主运行**过程
2	8	**同步动作**的**主运行**过程

下标 1	下标 2	含义：当掉电/断电时，在以下过程中写入数据时缓存器发生溢出：
2	9	**预处理**
2	10	**换刀**和**回退**的**主运行**过程
2	11	**同步动作**的**主运行**过程

优化方式

以下优化方式可以单独使用或组合使用：

- 从持久性用户数据切换为非持久性用户数据
  参见章节“使用易失性数据替代持久性数据 (页 909)”
- 增大中间缓存器的空间
  参见章节“MD18232:中间缓存器的大小 (页 912)”
- 加深查找深度
  参见章节“MD17610:中间缓存器的查找深度 (页 911)”

15.7.3.5 处理时间

从中间缓存器向“外部”静态存储器进行数据的同步回写时，在写操作期间会停止执行预处理。由于在此期间不处理新的程序段，因此会导致 NC 程序中断执行。

优化方式

以下优化方式可以单独使用或组合使用：

- 从持久性用户数据切换为非持久性用户数据
  参见章节“使用易失性数据替代持久性数据 (页 909)”
- 持久性数据的备份方式从同步切换为异步
  参见章节“MD18234:持久性数据的备份方式 (页 914)”
- 增大中间缓存器的空间
  参见章节“MD18232:中间缓存器的大小 (页 912)”
- 加深查找深度
  参见章节“MD17610:中间缓存器的查找深度 (页 911)”

15.7.3.6 中间缓存器的详细诊断

出现性能问题（处理时间、缓存器溢出、向“外部”静态存储器的写操作次数过多）时，需要对中间缓存器各区域的负载情况进行详细诊断，从而采取有针对性的处理措施。

检测缓存器区域

通过关闭各个中间缓存器区域中的持久性数据存储，来检测导致了最大写入负载的具体区域。

通过在机床数据 MD18232 (页 912) 中将特定区域的缓存器大小设为 0，便可关闭持久性数据存储。

优化方式

以下优化方式可针对测定的区域单独使用或组合使用：

- 从持久性用户数据切换为非持久性用户数据
  参见章节“使用易失性数据替代持久性数据 (页 909)”
- 增大测定区域的空间
  参见章节“MD18232:中间缓存器的大小 (页 912)”
- 加深测定区域的查找深度
  参见章节“MD17610:中间缓存器的查找深度 (页 911)”

附录 A

A.1 缩略语列表

A	
A	输出
AFIS	Automatic Filter Switch: 自动切换滤波器
ASCII	American Standard Code for Information Interchange:美国信息互换标准码
ASIC	Application Specific Integrated Circuit:用户自行开发的专用集成电路
ASUP	异步子程序
AUTO	AUTOMATIC，一种运行方式
AUXFU	Auxiliary Function:辅助功能
AWL	指令列表

B	
BAG	运行方式组
BCD	Binary Coded Decimals:用二进制代码编码的十进制数
BICO	Binector Connector
BIN	Binary Files:二进制文件
BKS	基准坐标系
BM	运行信息
BO	Binector Output
BTSS	操作面板接口

C	
CLC	间隙控制
CNC	Computerized Numerical Control: Computerunterstützte numerische Steuerung
COM	Communication
CP	Communication Processor

C	
CPU	Central Processing Unit:中央处理器
CST	Configured Stop:配置停止

D	
DDS	Drive Data Set:驱动数据组
DIR	Directory:目录
DO	Drive Object
DRF	Differential Resolver Function:微分旋转变压器功能（手轮）
DRY	Dry Run:空运行进给
DWORD	双字（当前 32 位）

E	
E	输入
EES	Execution from External Storage
E/A	输入/输出
ESR	扩展的停止和退回
ETC	ETC 键">"，用于在同一层菜单中扩展软键栏

F	
FDD	Feed Disable:进给禁止
FdStop	Feed Stop:进给停止
FIFO	First In First Out:存储器，工作无需地址说明，数据按存储的顺序读入
FM	故障信息
FUP	功能图（一种 PLC 编程方法）
FW	Firmware

G	
GEO	几何属性，例如几何值
GP	基本程序 (PLC)
GUD	Global User Data:全局用户数据

H	
HEX	十六进制数代号
HiFu	辅助功能
HMI	Human Machine Interface:SINUMERIK 操作介面
HSA	主轴驱动
HT	Handheld Terminal
HW	硬件

I	
IBN	调试
INC	Increment:增量尺寸
INI	Initializing Data:初始化数据
IPO	Interpolator

J	
JOG	Jogging:点动模式

K	
KOP	梯形图（一种 PLC 编程方法）

L	
LED	Light Emitting Diode:发光二极管
LMS	位置测量系统
LR	位置控制器

M	
Main	Main program:主程序 (OB1，PLC)
MCP	Machine Control Panel:机床控制面板
MD	机床数据

M	
MDA	Manual Data Automatic:手动数据输入
MDS	Motor Data Set:电机数据组
MELDW	信息字
MKS	机床坐标系
MM	Motor Module
MPF	Main Program File:主程序（NC）
MPI	Multi Point Interface
MSTT	机床控制面板

N	
NC	Numerical Control:带有程序段处理、运行范围等等的数控系统
NCU	Numerical Control Unit:NC 硬件单元
NCK	Numerical Control Kernel
NCSD	NC Start Disable
NST	接口信号
NV	零点偏移
NX	Numerical Extension:轴扩展模块

O	
OB	PLC 中组织块
OEM	Original Equipment Manufacturer
OP	Operation Panel:操作面板

P	
PCU	PC Unit:PC 主机（计算机元件）
PG	编程器
PI	程序实例
PLC	Programmable Logic Control:可编程逻辑控制器
PN	PROFINET
PO	Power On

P	
POS	位置/定位
PPO	Parameter Prozessdaten Objekt：通过 PROFIBUS–DP 和“转速可变驱动”传输时的循环数据报文
PPU	Panel Processing Unit ：基于面板的 CNC 控制系统（如 SINUMERIK 828D）的核心硬件
PROFIBUS	Process Field Bus:串行数据总线
PRT	程序测试
PTP	Point to Point:点到点
PZD	过程数据：PPO 的过程数据部分

R	
REF	返回参考点功能
REPOS	再定位功能
RESU	Retrace support
RID	Read In Disable
RP	R 参数，计算参数，预定义用户变量

S	
SA	同步动作
SBL	Single Block:单程序段
SBT	Safe Brake Test
SCC	Safety Control Channel
SCL	Structured Control Language
SD	Settingdatum：设定数据
SDI	Safe Direction
SERUPRO	Search-Run by Program Test:通过程序测试的程序段查找
SIC	Safety Info Channel
SKP	Skip:跳至零件程序段末尾
SLP	Safe Limited Position
SLS	Safely Limited Speed
SMI	Sensor Module Integrated

S	
SOS	Safe Operating Stop
SPF	Sub Program File:子程序（NC）
SS1	Safe Stop 1
SS2	Safe Stop 2
STO	Safe Torque Off
STW	控制字
SUG	砂轮外缘速度
SW	软件

T	
TCU	Thin Client Unit
TIA	Totally Integrated Automation
TM	Terminal Module (SINAMICS)
TO	Tool Offset:刀具补偿
TOA	Tool Offset Active:刀具补偿标识（文件类型）
TOFF	在线－刀具长度补偿
TRANSMIT	Transform Milling Into Turning:车床上铣削加工的坐标转换

U	
UP	子程序
USB	Universal Serial Bus

V	
VDI	NC 和 PLC 间的内部通讯接口

W	
WKS	工件坐标系
WPD	Work Piece Directory:工件目录
WZ	刀具
WZV	刀具管理

Z	
ZSW	（驱动）状态字

A.2 可用的 IPC

推荐用于 SINUMERIK 的 IPC

面板型 IPC	
IPC 477E 22" Win 7	6AV7241-3YA04-0FA0
IPC 477E 24" Win 7	6AV7241-5SB04-0FA0
IPC 477E 15" Win10	6AV7241-1WA07-0FA0
IPC 477E 19" Win10	6AV7241-3XB07-0FA0
IPC 477E 22" Win10	6AV7241-3YA07-0FA0
IPC 477E 24" Win10	6AV7241-5SB07-0FA0

箱式 IPC	
IPC 427E (Standard) Win7	6AG4141-1AA14-0FA0
IPC 427E (High) Win7	6AG4141-5AB14-0FA0
IPC 427E (Standard) Win10	6AG4141-1AA17-0FA0
IPC 427E (High) Win10	6AG4141-5AB17-0FA0

索引

_

A

B

C

N

O

P

R

S

T

W

X

包

保

被

比

标

参

测

插

程

粗

带

单

刀

地

点

定

动