SIEMENS

SINUMERIK

SINUMERIK 828D
工作准备部分

编程手册

适用于：

CNC 系统软件 版本 4.3

07/2010

6FC5398-2BP40-0RA0

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
带有警告三角，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。

小心
不带警告三角，表示如果不采取相应的小心措施，可能导致财产损失。

注意
表示如果不注意相应的提示，可能会出现不希望的结果或状态。

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。

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 ® 的都是西门子股份有限公司的注册商标。标签中的其他符号可能是一些其他商标，这是出于保护所有者权利的 目地由第三方使用而特别标示的。

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
Industry Sector
Postfach 48 48
90026 NÜRNBERG
德国

文件订购号: 6FC5398-2BP40-0RA0
℗ 06/2010

前言

SINUMERIK 文献

SINUMERIK 文献分为 3 个类别：

- 一般文献
- 用户文献
- 制造商/维修文献

在网页 http://www.siemens.com/motioncontrol/docu 中可获取下列主题的相关信息：

- 订购资料
  这里您可以查阅到当前的印刷品一览。
- 下载资料
  更多用于从“服务与支持”下载文件的链接。
- 在线检索资料
  获取 DOConCD 的信息，以及直接访问 DOConWEB 中的印刷品。
- 以西门子文献的内容为基础，使用 My Documentation Manager（MDM）创建个人文献，请访问 http://www.siemens.com/mdm

  My Documentation Manager 提供了一系列功能用于创建用户自己的机床文献。
- 培训与 FAQ（常见问题解答）

  通过页面导航可以获取培训以及 FAQ（常见问题解答）的相关信息。

目标客户

该手册供以下人员使用：

- 编程人员
- 设计人员

使用

利用该编程手册目标用户可以设计程序和软件界面、写入、测试和消除故障。

标准功能范畴

在该编程说明中描述了标准的功能范畴。 机床制造商增添或者更改的功能，由机床制造商资料进行说明。

控制系统有可能执行本文献中未描述的某些功能。 但是这并不意味着在提供系统时必须带有这些功能，或者为其提供有关的维修服务。

同样，因为只是概要，所以该文献不包括全部类型产品的所有详细信息，也无法考虑到安装、运行和维修中可能出现的各种情况。

技术支持

请咨询下列热线：

	欧洲 / 非洲
电话	+49 (0) 911 895 7222
传真	+49 (0) 911 895 7223
网址	http://www.siemens.de/automation/support-request

	美洲
电话	+1 423 262 2522
传真	+1 423 262 2200
电子邮件	mailto:techsupport.sea@siemens.com

	亚洲 / 太平洋
电话	+86 1064 757 575
传真	+86 1064 747 474
电子邮件	mailto:support.asia.automation@siemens.com

说明

各个国家的技术支持电话请访问以下网址：
http://www.automation.siemens.com/partner

文献资料疑问

如果您对该文献有疑问（建议，修改），请发送传真或电子邮件到下列地址：

传真： +49 9131 98 2176
电子邮件： mailto:motioncontrol.docu@siemens.com

传真表格见本文献附录。

SINUMERIK 网址

http://www.siemens.com/sinumerik

编程手册“基本原理”和“工作准备”。

关于 NC 编程的说明分列在两本手册中：

1. **基本原理**

   编程手册“基本原理”供机床专业操作供使用，需要有相应的钻削、铣削和车削加工知识。 这里也利用一些简单的编程举例，说明常见的指令和语句（符合 DIN66025）。

2. **工作准备部分**

   编程手册“工作准备部分”供熟悉所有编程方法的工艺人员使用。 SINUMERIK 控制系统可利用一种专用编程语言对复杂的工件程序（例如自由成形曲面，通道坐标，.....）进行编程，并且可减轻工艺人员编程的负担。

NC 语言的可用性

本手册中所描述的全部 NC 语言都可用于 SINUMERIK 840D sl。 有关 SINUMERIK 828D 的可用性见表格“指令：在 SINUMERIK 828D 上的可用性 (页 808)”。

目录

灵活的 NC 编程 1

1.1 变量

1.1.1 变量的一般信息

通过使用变量，特别是计算功能和控制结构的相关变量，可以使零件程序和循环的编写更为灵活。 为此系统提供了三种不同类型的变量：

- 系统变量

  系统变量是系统中定义供用户使用的变量，它们具有固定的预设含义。 也可以通过系统软件读取和写入这些变量。 示例： 机床数据

  系统变量的含义中的大部分属性由系统固定预设。 用户只能小范围的对属性进行重新定义和匹配。 参见“系统变量，用户变量和 NC 语言指令的重新定义（REDEF） (页 32)”

- 用户变量

  用户变量是系统不确知其含义，也不对其进行分析的变量。 其含义只由用户定义。

  用户变量又可以分为：

  – 预定义用户变量

    预定义用户变量是在系统中已经定义的变量，但是用户还需通过专门的机床数据对其数量进行参数设置。 这些变量的属性大部分由用户进行匹配。 参见“系统变量，用户变量和 NC 语言指令的重新定义（REDEF） (页 32)”。

  – 用户定义变量

    用户定义变量是仅由用户定义的变量，直到运行时系统才会创建这些变量。 它们的数量，数据类型，可见性和所有其它属性都完全由用户定义。

    参见“定义用户变量（DEF） (页 25)”

参见

系统变量 (页 18)

预定义用户变量： 计算参数（R） (页 21)

预定义用户变量：链接变量 (页 23)

属性：初始化值 (页 35)

属性： 极限值(LLI, ULI) (页 38)

属性： 物理单位(PHU) (页 40)

属性： 存取权限(APR, APW, APRP, APWP, APRB, APWB) (页 44)

可定义和可重新定义的属性一览 (页 49)

定义和初始化数组变量（DEF，SET，REP） (页 51)

数据类型 (页 59)

1.1.2 系统变量

系统变量是在系统中预定义的变量，通过此变量可在零件程序与循环中存取当前控制系统的编程，以及机床、控制系统和加工步骤状态。

预处理变量

预处理变量是指在预处理程序状态中，即在对编程了系统变量的零件程序段进行编译时，读取和写入的系统变量。 预处理变量不会触发预处理停止。

主运行变量

主运行变量是指在主运行程序状态中，即在执行编程了系统变量的零件程序段时，读取和写入的系统变量。 主运行变量有：

- 可在同步动作中编程的系统变量（读取/写入）
- 可在零件程序中编程的系统变量（读取/写入）
- 可在零件程序中编程并在预处理中计算值，但是在主运行中才写入的系统变量（主运行同步：只写入）

前缀系统

系统变量的一个显著特点是其名称通常包含一个前缀，该前缀以 $ 字符之后跟随一个或两个字母以及一条下划线的形式构成。

$ + 第 1 个字母	含义： 数据类型
在预处理时读取/写入的系统变量	
$M	机床数据 1)
$S	设定数据，保护区域 1)
$T	刀具管理参数
$P	程序数值
$C	ISO 包络循环的循环变量
$O	选项数据
R	R 参数（计算参数）2)
在主运行时读取/写入的系统变量	
$$M	机床数据 1)
$$S	设定数据 1)
$A	当前主运行数据
$V	伺服数据
$R	R 参数（计算参数）2)
1) 在零件程序/循环中使用机床数据和设定数据作为预处理变量时，在前缀中写入一个 $ 字符。 在同步动作中用作主运行变量时，在前缀中写入两个 $ 字符。 2) 在零件程序/循环中使用 R 参数作为预处理变量时，不写入前缀，如 R10。 在同步动作中用作主运行变量时，在前缀中写入一个 $ 字符，如 $R10。	

第 2 个字母	含义： 变量显示
N	NCK 全局变量（NCK）
C	通道专用变量（Channel）
A	轴专用变量（Axis）

边界条件

前缀系统中的特殊情况

以下系统变量和上述前缀系统有偏差：

- $TC_...： 第 2 个字母 C 在这里表示的不是通道专用，而是刀架专用系统变量（TC = Tool Carrier）
- $P_ ...： 通道专用系统变量

在同步动作中使用机床数据和设定数据

在同步动作中使用机床数据和设定数据时，可通过前缀确定，机床数据或设定数据是预处理同步还是主运行同步读取/写入。

如果数据在同步期间保持不变，则可以和预处理同步地读取数据。 为此在机床数据或设定数据的前缀中写入一个 $ 字符：

程序代码

```
ID=1 WHENEVER G710 $AA_IM[z] < $SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0
```

如果数据在同步期间改变，则必须和主运行同步地读取/写入数据。 为此在机床数据或设定数据的前缀中写入两个 $ 字符：

程序代码

```
ID=1 WHENEVER $AA_IM[z] < $$SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0
```

说明

写入机床数据

在写入机床数据或设定数据时必须注意，在执行零件程序/循环时，生效的存取级允许写入操作，且数据的有效性为“IMMEDIATE”。

文献

全部系统变量的属性列表请参见：

/PGA1/ 参数手册 系统变量

参见

变量的一般信息 (页 17)

1.1.3 预定义用户变量： 计算参数（R）

功能

计算参数或 R 参数是名称为 R 的预定义用户变量，定义为 REAL 数据类型的数组。 由于历史原因，R 参数既可以带数组索引编写，如 R[10]，也可不带数组索引编写，如 R10。

在同步动作中使用计算参数时，必须写入 $ 字符作为前缀，如$R10。

句法

作为预处理变量使用时：
R<n>
R[<表达式>]

作为主运行变量使用时：
$R<n>
$R[<表达式>]

含义

R： 作为预处理变量使用时的名称，比如在零件程序中

$R： 作为主运行变量使用时的名称，比如在同步动作中

类型： REAL

取值范围： 非指数的写入方式：

± (0.000 0001 ... 9999 9999)

提示：

最多允许有 8 个小数位

指数写入方式：

± (1*10^{-300} ... 1*10^{+300})

提示：

- 写入方式：<尾数>EX<指数>，如：8.2EX-3
- 最多允许有 10 个字符（包括符号和小数点）。

<n>： R 参数编号

类型： INT

取值范围： 0 - MAX_INDEX

提示

MAX_INDEX 由 R 参数中设置的数量得出：
MAX_INDEX = (MD28050
$MN_MM_NUM_R_PARAM) - 1

<表达式>： 数组索引

只要可将表达式结果转换为数据类型 INT，则可设定任意表达式作为数组索引（INT，REAL，BOOL，CHAR）。

示例

算术功能中 R 参数的赋值和应用：

程序代码		注释
R0=3.5678	;	在预处理中赋值
R[1]=-37.3	;	在预处理中赋值
R3=-7	;	在预处理中赋值
$R4=-0.1EX-5	;	在主运行中赋值：R4 = -0.1 * 10^-5
$R[6]=1.874EX8	;	在主运行中赋值：R6 = 1.874 * 10^8
R7=SIN(25.3)	;	在预处理中赋值
R[R2]=R10	;	通过 R 参数间接定址
R[(R1+R2)*R3]=5	;	通过算术表达式间接定址
X=(R1+R2)	;	将 X 轴运行至由 R1 和 R2 的和确定的位置
Z=SQRT(R1*R1+R2*R2)	;	将 Z 轴运行至通过 (R1^2 + R2^2) 的平方根确定的位置

参见

变量的一般信息 (页 17)

1.1.4 预定义用户变量：链接变量

功能

通过链接变量，可在“NCU 链接”功能的范围内循环交换一个网络中相连的 NCU 之间的数据。 此时，可以访问链接变量存储器中特定格式的数据。 用户/机床制造商确定设备专用链接变量存储器时，既须考虑大小，也须考虑数据结构。

链接变量为系统全局用户变量，在设置了链接连接时这些变量能够从所有链接组的 NCU 中读取或写入到零件程序段和循环。 与全局用户数据（GUD）不同，链接变量也可用在同步动作中使用。

在无有效 NCU 链接的设备上，除了全局用户变量（GUD）外，可将链接变量作为控制系统本地全局用户变量使用。

句法

```
$A_DLB[<索引>]
$A_DLW[<索引>]
$A_DLD[<索引>]
$A_DLR[<索引>]
```

含义

$A_DLB:	数据格式 BYTE（1 字节）的链接变量	
	数据类型：	UINT
	取值范围：	0 ... 255
$A_DLW:	数据格式 WORD（2 字节）的链接变量	
	数据类型：	INT
	取值范围：	-32768 ... 32767
$A_DLD:	数据格式 DWORD（4 字节）的链接变量	
	数据类型：	INT
	取值范围：	-2147483648 ... 2147483647
$A_DLR:	数据格式 REAL（8 字节）的链接变量	
	数据类型：	REAL
	取值范围：	$\pm(2{,}2*10^{-308} \ldots 1{,}8*10^{+308})$

<索引>： 地址索引以字节，从链接变量存储器开始处计算

数据类型： INT

取值范围： 0 - MAX_INDEX

提示

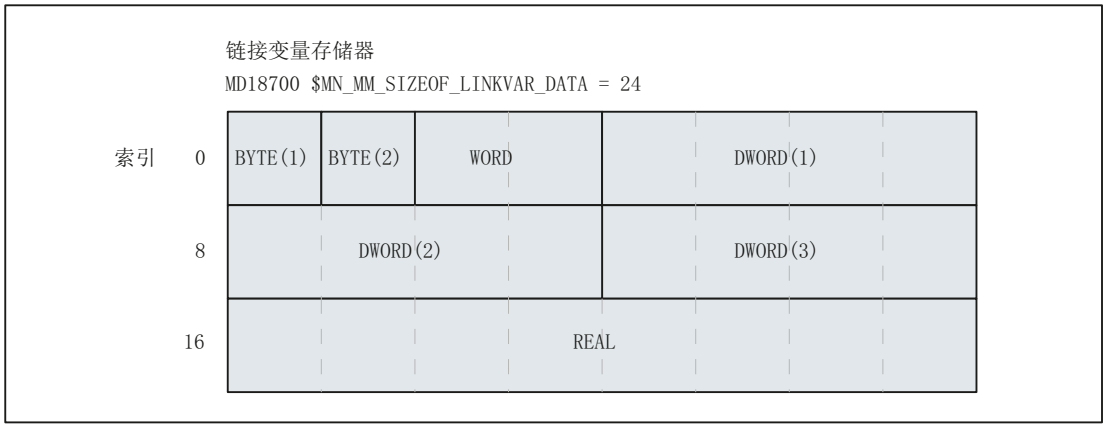
- MAX_INDEX 由参数设置的链接变量存储器大小得出： MAX_INDEX = (MD18700 $MN_MM_SIZEOF_LINKVAR_DATA) - 1
- 只可对索引进行编程，从而可以使链接变量存储器中定址的字节位于数据格式限制内 ⇒
  索引= n * 字节，其中 n = 0，1，2，...
  - $A_DLB[i]： i = 0，1，2，...
  - $A_DLW[i]： i = 0，2，4，...
  - $A_DLD[i]： i = 0，4，8，...
  - $A_DLR[i]： i = 0，8，16，...

示例

在自动化设备中有 2 个 NCU（NCU1 和 NCU2）。 在 NCU1 上连接了机床轴 AX2，该轴作为 NCU2 的链接轴运行。

NCU1 将轴 AX2 的电流实际值（$VA_CURR）循环写入链接变量存储器。 NCU2 循环读取通过链接通讯传输的电流实际值，并在超出限值时显示报警 61000。

链接变量中的数据结构在下图中显示。 电流实际值以 REAL 值传输。



NCU1

NCU1 在静态同步动作的 IPO 周期中将轴 AX2 的电流实际值通过链接变量 $A_DLR[16] 循环写入链接变量存储器。

程序代码

```
N111 IDS=1 WHENEVER TRUE DO $A_DLR[16]=$VA_CURR[AX2]
```

NCU2

NCU2 在静态同步动作的 IPO 周期中通过链接变量 $A_DLR[16] 从链接变量存储器循环读取轴 AX2 的电流实际值。 如果电流实际值大于 23.0 A，则显示报警 61000。

程序代码

```
N222 IDS=1 WHEN $A_DLR[16] > 23.0 DO SETAL(61000)
```

参见

变量的一般信息 (页 17)

1.1.5 定义用户变量（DEF）

功能

用户可通过 DEF 指令定义自己的变量并进行赋值。 在划分系统变量时，这些变量被称为用户定义变量或用户变量（User Data）。

根据变量的有效范围，即变量可见范围，用户变量可分为以下几个类别：

- 局部用户变量（LUD）

  局部用户变量（LUD）是在执行时不是主程序的零件程序中定义的变量。 此变量在调用零件程序时创建，并在零件程序结束或者 NC 复位时删除。 只能在定义 LUD 的零件程序中存取该 LUD。

- 程序全局用户变量（PUD）

  程序全局用户变量（PUD）是在作为主程序的零件程序中定义的变量。 此变量在零件程序开始时创建，在零件程序结束或 NC 复位时删除。 可在主程序及所有子程序中存取 PUD。

- 全局用户变量（GUD）

  全局用户变量（GUD）是在数据块（SGUD，MGUD，UGUD，GUD4 ... GUD9）中定义的 NC 或通道全局变量，此变量上电后依然保留。 可在所有零件程序中存取 GUD。

在使用（读/写）用户变量前必须对其进行定义 必须遵循以下规则：

- GUD 必须在定义文件如 _N_DEF_DIR/_M_SGUD_DEF 中定义 。
- PUD 和 LUD 必须在零件程序的定义段中定义。
- 必须在单独的程序段中进行数据定义。
- 每次数据定义只能使用一种数据类型。
- 每次数据定义可以定义多个相同数据类型的变量。

句法

```
DEF <范围> <类型> <预处理停止> <初始化时间> <物理单位> <限值> <存取权限>
<名称>[<值_1>,<值_2>,<值_3>]=<初始化值>
```

含义

DEF：	用于定义用户变量 GUD，PUD，LUD 的指令	
<范围>：	有效范围，只和 GUD 相关：	
	NCK：	NC 全局用户变量
	CHAN：	通道全局用户变量
<类型>：	数据类型：	
	INT：	带正负号的整数值

	REAL:	实数（根据 IEEE 为 LONG REAL）
	BOOL:	真值 TRUE (1) / FALSE (0)
	CHAR:	ASCII-字符
	STRING[<最大长度>]:	定义长度的字符串
	AXIS:	进给轴/主轴标识符
	FRAME:	静态坐标转换的几何设定

参见“数据类型 (页 59)”

<预处理停止>: 预处理停止，只 GUD 相关（可选）

SYNR: 在读取时执行预处理停止

SYNW: 在写入时执行预处理停止

SYNRW: 在读取/写入时执行预处理停止

<初始化时间>: 变量重新初始化的时间（可选）

INIPO: 上电

INIRE: 主程序结束，NC 复位或上电

INICF: 重新配置或主程序结束，NC 复位或上电

PRLOC: 主程序结束，本地更改后 NC 复位或上电

参见“属性：初始化值 (页 35)”

<物理单位>: 物理单位（可选）

PHU <单位>:

参见“属性： 物理单位(PHU) (页 40)”

<限值>: 上限或下限（可选）

LLI <限值>: 下限（lower limit）

ULI <限值>: 上限（upper limit）

参见“属性： 极限值(LLI, ULI) (页 38)”

<存取权限>: 通过零件程序或 BTSS 读取/写入 GUD 的权限（可选）

APRP <保护等级>: 读取： 零件程序

APWP <保护等级>: 写入： 零件程序

APRB <保护等级>: 读取： BTSS

APWB <保护等级>: 写入： BTSS

	保护等级　　取值范围： 0 ... 7
	参见“属性： 存取权限(APR, APW, APRP, APWP, APRB, APWB) (页 44)”
<名称>：	变量名称
	提示 • 最多 31 个字符 • 前两个字符必须为一个字母和/或一条下划线。 • “$”字符预留给系统变量，不可使用。
[<值_1>, <值_2>, <值_3>]：	设定 1 维至 3 维（最大）数组变量的数组长度（可选）
<初始化值>：	初始化值（可选） 参见“属性：初始化值 (页 35)” 用于初始化数组变量： 参见“定义和初始化数组变量（DEF，SET，REP） (页 51)”

示例

示例 1： 在机床制造商数据块中定义用户变量

程序代码

```
%_N_MGUD_DEF                                                    ; GUD 模块： 机床制造商
$PATH=/_N_DEF_DIR
DEF CHAN REAL PHU 24 LLI 0 ULI 10 STROM_1, STROM_2
; 描述
; 两个 GUD 的定义： STROM_1, STROM_2
; 有效范围： 整个通道
; 数据类型： REAL
; 预处理停止：未编程 => 缺省值 = 无预处理停止
; 物理单位： 24 = [A]
; 限值：Low = 0.0，High = 10.0
; 存取权限： 未编程 => 缺省值 = 7 = 钥匙开关位置 0
; 初始化值：未编程 => 缺省值 = 0.0

DEF NCK REAL PHU 13 LLI 10 APWP 3 APRP 3 APWB 0 APRB 2 ZEIT_1=12, ZEIT_2=45
; 说明
; 两个 GUD 的定义： ZEIT_1, ZEIT_2
```

程序代码
; 有效范围： 整个 NCK
; 数据类型：REAL
; 预处理停止：未编程 => 缺省值 = 无预处理停止
; 物理单位： 13 = [s]
; 限值：Low = 10.0，High = 未编程 => 定义范围上限
; 存取权限：
; 零件程序：写入/读取 = 3 = 最终用户
; BTSS:写入 = 0 = 西门子，读取 = 3 = 最终用户
; 初始化值：ZEIT_1 = 12.0，ZEIT_2 = 45.0
DEF NCK APWP 3 APRP 3 APWB 0 APRB 3 STRING[5] GUD5_NAME = "COUNTER"
; 说明
; 一个 GUD 的定义： GUD5_NAME
; 有效范围： 整个 NCK
; 数据类型： STRING，最大 5 个字符
; 预处理停止：未编程 => 缺省值 = 无预处理停止
; 物理单位：未编程 => 缺省值 = 0 = 无物理单位
; 限值：未编程 => 定义范围限值：Low = 0，High = 255
; 存取权限：
; 零件程序：写入/读取 = 3 = 最终用户
; BTSS:写入 = 0 = 西门子，读取 = 3 = 最终用户
; 初始化值：“COUNTER”
M30

示例 2： 程序全局和局部用户变量（PUD / LUD）

程序代码	注释
PROC MAIN	; 主程序
DEF INT VAR1	; PUD 定义
...	
SUB2	; 子程序调用
...	
M30	

程序代码	注释
PROC SUB2	; 子程序 SUB2
DEF INT VAR2	; LUD 定义

程序代码	注释
...	
IF(VAR1==1)	; PUD 读取
VAR1=VAR1+1	; PUD 读取和写入
VAR2=1	; LUD 写入
ENDIF	
SUB3	; 子程序调用
...	
M17	

程序代码	注释
PROC SUB3	; 子程序 SUB3
...	
IF(VAR1==1)	; PUD 读取
VAR1=VAR1+1	; PUD 读取和写入
VAR2=1	; 错误： SUB2 中的 LUD 未知
ENDIF	
...	
M17	

示例 3： 数据类型为 AXIS 的用户变量的定义和应用

程序代码	注释
DEF AXIS ABSZISSE	; 1. 几何轴
DEF AXIS SPINDLE	; 主轴
...	
IF ISAXIS(1) == FALSE GOTOF WEITER	
ABSZISSE = $P_AXN1	
继续:	
...	
SPINDLE=(S1)	1. 主轴
OVRA[SPINDLE]=80	; 主轴倍率 = 80%
SPINDLE=(S3)	3. 主轴

边界条件

全局用户变量（GUD）

在定义全局用户变量（GUD）时须考虑以下机床数据：

编号	名称： $MN_	含义
11140	GUD_AREA_ SAVE_TAB	GUD 模块的附加备份
18118 1)	MM_NUM_GUD_MODULES	当前主动文件系统中 GUD 文件的数量
18120 1)	MM_NUM_GUD_NAMES_NCK	全局 GUD 名称数量
18130 1)	MM_NUM_GUD_NAMES_CHAN	通道专用 GUD 名称数量
18140 1)	MM_NUM_GUD_NAMES_AXIS	轴专用 GUD 名称数量
18150 1)	MM_GUD_VALUES_MEM	全局 GUD 值的存储空间
18660 1)	MM_NUM_SYNACT_GUD_REAL	可设置的数据类型为 REAL 的 GUD 数量
18661 1)	MM_NUM_SYNACT_GUD_INT	可设置的数据类型为 INT 的 GUD 数量
18662 1)	MM_NUM_SYNACT_GUD_BOOL	可设置的数据类型为 BOOL 的 GUD 数量
18663 1)	MM_NUM_SYNACT_GUD_AXIS	可设置的数据类型为 AXIS 的 GUD 数量
18664 1)	MM_NUM_SYNACT_GUD_CHAR	可设置的数据类型为 CHAR 的 GUD 数量
18665 1)	MM_NUM_SYNACT_GUD_STRING	可设置的数据类型为 STRING 的 GUD 数量

1) 不适用于 SINUMERIK 828D。

程序全局用户变量（PUD）

注意
程序全局用户变量的可见性（PUD） 当设置了以下机床数据时，在主程序中定义的程序局部用户变量（PUD）同样在子程序中可见。 MD11120 $MN_LUD_EXTENDED_SCOPE = 1 设置 MD11120 = 0 时，在主程序中定义的程序局部用户变量只在主程序中可见。

数据类型为 AXIS 的 NCK 全局变量的跨通道应用

当通道中的轴的通道轴编号相同时，在数据块定义时使用轴名称初始化的，数据类型为 AXIS 的 NCK 全局用户变量才可在 NC 的不同通道中使用。

如果不是这种情况，必须在零件程序开始处载入变量，或者象下面的例子一样使用 AXNAME(...) 功能（参见：“”）。

程序代码	注释
DEF NCK STRING[5] ACHSE="X"	; 在数据块中定义
N100 AX[AXNAME(ACHSE)]=111 G00	; 在零件程序中使用。

参见

变量的一般信息 (页 17)

1.1.6 系统变量，用户变量和 NC 语言指令的重新定义（REDEF）

功能

使用 REDEF 指令可对系统变量，用户变量和 NC 语言指令的属性进行更改。 重新定义的前提条件是，必须在相应的定义后进行。

在重新定义中不能同时对多个属性进行更改。 必须为每个需要更改的属性编程单独的 REDEF 指令。

如果编程的多个属性更改之间有冲突，则最后进行的更改生效。

可重定义属性

参见“可定义和可重新定义的属性一览 (页 49)”

局部用户变量（PUD / LUD）

不能对局部用户变量（PUD / LUD）进行重新定义。

句法

```
REDEF <名称> <预处理停止>
REDEF <名称> <物理单位>
REDEF <名称> <限值>
REDEF <名称> <存取权限>
REDEF <名称> <初始化时间>
REDEF <名称> <初始化时间> <初始化值>
```

含义

REDEF： 用于重定义系统变量，用户变量和 NC 语言指令的特定属性的指令

<名称>： 已定义的变量或 NC 语言指令的名称

<预处理停止>： 预处理停止

SYNR: 在读取时执行预处理停止

SYNW： 在写入时执行预处理停止

SYNRW： 在读取/写入时执行预处理停止

<物理单位>： 物理单位

PHU <单位>：

参见“属性： 物理单位(PHU) (页 40)”

提示

不可进行重新定义：

- 系统变量
- 全局用户数据（GUD）
- 数据类型： BOOL，AXIS，STRING，FRAME

<限值>： 下限和/或上限

LLI <限值>： 下限（lower limit）

ULI <限值>： 上限（upper limit）

参见“属性： 极限值(LLI, ULI) (页 38)”

提示

不可进行重新定义：

- 系统变量
- 全局用户数据（GUD）
- 数据类型： BOOL，AXIS，STRING，FRAME

<存取权限>： 通过零件程序或 BTSS 读取/写入的权限

APX <保护等级>： 执行：NC 语言元素

APRP <保护等级>： 读取：零件程序

APWP <保护等级>： 写入：零件程序

APRB <保护等级>： 读取：BTSS

APWB <保护等级>： 写入：BTSS

保护等级 取值范围： 0 ... 7

参见“属性： 存取权限(APR, APW, APRP, APWP, APRB, APWB) (页 44)”

<初始化时间>： 变量重新初始化的时间

INIPO： 上电

INIRE： 主程序结束，NC 复位或上电

INICF： 重新配置或主程序结束，NC 复位或上电

PRLOC： 主程序结束，本地更改后 NC 复位或上电

参见“属性：初始化值 (页 35)”

<初始化值>： 初始化值
在定义初始化值时，必须设定初始化时间（参见<初始化时间>）。
参见“属性：初始化值 (页 35)”
用于初始化数组变量：
参见“定义和初始化数组变量（DEF，SET，REP） (页 51)”

提示

不可进行重新定义：

- 系统变量，除去设定数据

示例

重新定义系用于机床制造商的数据块的系统变量 $TC_DPC1

程序代码

```
%_N_MGUD_DEF                                                    ; GUD 模块：机床制造商
$PATH=/_N_DEF_DIR
REDEF $TC_DPC1 APWB 2 APWP 3
REDEF $TC_DPC1 PHU 21
REDEF $TC_DPC1 LLI 0 ULI 200
REDEF $TC_DPC1 INIPO (100, 101, 102, 103)
; 描述
; 写入权限：BTSS = 保护等级 2，零件程序 = 保护等级 3
; 提示
; 在使用 ACCESS 文件时
; 必须将对 _N_MGUD_DEF 的重定义权限转移到 _N_MACCESS_DEF
; 物理单位 = [ % ]
; 限值：下限 = 0，上限 = 200
; 在上电时使用四个值初始化数组变量
M30
```

边界条件

粒度

重定义总是针对所有通过名称明确标识的变量。 不能在数组变量中为单个的数组单元分配不同的属性值。

参见

变量的一般信息 (页 17)

1.1.7 属性：初始化值

用户变量的定义(DEF)

在进行定义时可为以下变量预设一个初始化值：

- 全局用户变量（GUD）
- 程序全局用户变量（PUD）
- 局部用户变量（LUD）

重新定义（REDEF）系统和用户变量

在进行重定义时可为以下变量预设一个初始化值：

- 系统数据
  - 设定数据
- 用户数据
  - R 参数
  - 同步动作变量（$AC_MARKER，$AC_PARAM，$AC_TIMER）
  - 同步动作 GUD (SYG_xy[ ], 其中 x=R, I, B, A, C, S;y=S, M, U, 4, ..., 9)
  - EPS 参数
  - 刀具数据 OEM
  - 刀库数据 OEM
  - 全局用户变量（GUD）

重新初始化时间

在进行重新定义时可设定变量重新初始化的时间，即重新设置为初始化值的时间：

- INIPO（上电）

  在上电时重新初始化变量。

- INIRE（复位）

  在 NC 复位，BAG 复位，零件程序结束（M02 / M30）或上电时重新初始化变量。

- INICF（新配置）

  在通过 HMI、零件程序指令 NEWCONFIG 请求重新配置时，或者 NC 复位，BAG 复位，零件程序结束（M02 / M30）或上电时重新初始化变量。

- PRLOC：（程序局部更改）

  只有在当前零件程序范围内进行修改变量时，才可在 NC 复位，BAG 复位或零件程序结束（M02 / M30）时进行重新初始化。

  PRLOC 属性必须与可编程设定数据（见下表）一起使用。

表格 1- 1　可编程的设定数据

序号	名称	G 指令 1)
42000	$SC_THREAD_START_ANGLE	SF
42010	$SC_THREAD_RAMP_DISP	DITS / DITE
42400	$SA_PUNCH_DWELLTIME	PDELAYON
42800	$SA_SPIND_ASSIGN_TAB	SETMS
43210	$SA_SPIND_MIN_VELO_G25	G25
43220	$SA_SPIND_MAX_VELO_G26	G26
43230	$SA_SPIND_MAX_VELO_LIMS	LIMS
43300	$SA_ASSIGN_FEED_PER_REV_SOURCE	FPRAON
43420	$SA_WORKAREA_LIMIT_PLUS	G26
43430	$SA_WORKAREA_LIMIT_MINUS	G25
43510	$SA_FIXED_STOP_TORQUE	FXST
43520	$SA_FIXED_STOP_WINDOW	FXSW
43700	$SA_OSCILL_REVERSE_POS1	OSP1
43710	$SA_OSCILL_REVERSE_POS2	OSP2

序号	名称	G 指令 1)
43720	$SA_OSCILL_DWELL_TIME1	OST1
43730	$SA_OSCILL_DWELL_TIME2	OST2
43740	$SA_OSCILL_VELO	FA
43750	$SA_OSCILL_NUM_SPARK_CYCLES	OSNSC
43760	$SA_OSCILL_END_POS	OSE
43770	$SA_OSCILL_CTRL_MASK	OSCTRL
43780	$SA_OSCILL_IS_ACTIVE	OS
43790	$SA_OSCILL_START_POS	OSB
1)使用此 G 指令对设定数据进行响应		

边界条件

初始化值： 全局用户变量（GUD）

- 对于有效范围为 NCK 的全局用户变量（GUD），只有 INIPO（上电）能预设为初始化时间。
- 对于有效范围为 CHAN 的全局用户变量（GUD），除了 INIPO（上电）外，INIRE（复位）或 INICF（新配置）也可以预设为初始化时间。
- 对于有效范围为 CHAN、初始化时间为 INIRE（复位）或 INICF（新配置）的全局用户变量(GUD)，只有在通道中触发了列举的事件时，才会在 NC 复位，BAG 复位和新配置时在该通道中重新初始化变量。

初始化值： 数据类型 FRAME

不能为数据类型为 FRAME 的变量设定初始化值。 数据类型为 FRAME 的变量总是通过缺省框架隐式初始化。

初始化值： 数据类型 CHAR

对于数据类型为 CHAR 的变量，也可通过带引号的 ASCII 符号编程，比如“A”，而不用 ASCII 码（0...255）。

初始化值： 数据类型 STRING

对于数据类型为 STRING 的变量，必须使用带引号的字符串进行编程，例如：...= "MASCHINE_1"

初始化值： 数据类型 AXIS

对于数据类型为 AXIS 的变量，必须在具有扩展地址时将轴名称编写在括号中，例如：...=(X3)

初始化值： 系统变量

对于系统变量，可通过重新定义来指定非用户专用的初始化值。 系统变量的初始化值由系统固定预设。 但是通过重新定义可以更改系统变量重新初始化的时间（INIRE，INICF）。

隐式初始化值： 数据类型 AXIS

对于数据类型为 AXIS 的变量，使用以下隐式初始化值：

- 系统数据：“第一几何轴”
- 同步动作 GUD（名称：SYG_A*），PUD，LUD：
  机床数据中的轴名称： MD20082 $MC_AXCONF_CHANAX_DEFAULT_NAME

隐式初始化值： 刀具和刀库数据

对于刀具和刀库数据，可通过以下机床数据预设隐式初始化值： MD17520 $MN_TOOL_DEFAULT_DATA_MASK

注意
同步 用户/机床制造商应全权负责实现同步，即触发全局数据重新初始化的事件和在别处读取该变量之间的同步。

参见

变量的一般信息 (页 17)

1.1.8 属性： 极限值(LLI, ULI)

只允许为以下类型的数据确定定义范围的上限值和下限值。

- INT
- REAL
- CHAR

用户变量的定义(DEF)： 极限值和隐式初始化值

在定义上述某个数据类型的一个用户变量时，如果没有定义显式初始化值，该变量会设为该数据类型的隐式初始化值。

- INT: 0
- REAL: 0.0
- CHAR: 0

如果隐式初始化值超出了编程极限值构成的定义范围，该变量会按照隐式初始化值临近的极限值进行初始化：

- 隐式初始化值 **<** 下限值(LLI)⇒
  初始化值 **=** 下限值
- 隐式初始化值 **>** 上限值(LLI)⇒
  初始化值 **=** 上限值

示例：

程序代码	注释
DEF REAL GUD1	; 下限值 = 定义范围极限
	; 上限值 = 定义范围极限
	; 未编程初始化值
	; => 隐式初始化值 = 0.0
DEF REAL LLI 5.0 GUD2	; 下限值 = 5.0
	; 上限值 = 定义范围极限
	; => 初始化值 = 5.0
DEF REAL ULI -5 GUD3	; 下限值 = 定义范围极限
	; 上限值 = -5.0
	; => 初始化值 = -5.0

用户变量的重新定义(DEF)： 极限值和当前实际值

如果在重新定义一个用户变量的极限值时，当前实际值超出了新的定义范围，系统会输出报警，而不接收该极限值。

说明

用户变量的重新定义(DEF)

在重新定义用户变量的极限值时，请注意以下值的修改应保持一致：

- 极限值
- 实际值
- 初始化值，在重新定义和由于 INIPO、INIRE 或 INICF 自动重新初始化时

参见

变量的一般信息 (页 17)

1.1.9 属性： 物理单位(PHU)

只允许为以下类型的变量设定物理单位：

- INT
- REAL

可编程的物理单位(PHU)

物理单位作为定点数设定： PHU <单位>

可编程以下物理单位：

<单位>	含义	物理单位
0	没有物理单位	-
1	线性位置或角度位置 1)2)	[mm], [inch], [deg]
2	线性位置 2)	[mm], [inch]
3	角度位置	[deg]
4	线性速度或角速度 1)2)	[mm/min], [inch/min], [rpm]
5	线性速度 2)	[mm/min]

<单位>	含义	物理单位
6	角速度	[rpm]
7	线性加速度或角加速度 1)2)	[m/s²], [inch/s²], [rev/s²]
8	线性加速度 2)	[m/s²], [inch/s²]
9	角加速度	[rev/s²]
10	线性急动度或角急动度 1)2)	[m/s³], [inch/s³], [rev/s³]
11	线性急动度 2)	[m/s³], [inch/s³]
12	角急动度	[rev/s³]
13	时间	[s]
14	位置控制器增益	[16.667/s]
15	旋转进给率 2)	[mm/rev], [inch/rev]
16	温度补偿 1)2)	[mm], [inch]
18	力	[N]
19	质量	[kg]
20	惯性矩 3)	[kgm²]
21	百分比	[%]
22	频率	[Hz]
23	电压	[V]
24	电流	[A]
25	温度	[°C]
26	角度	[deg]
27	KV	[1000/min]
28	线性位置或角度位置 3)	[mm], [inch], [deg]
29	切削速度 2	[m/min], [feet/min]
30	圆周速度 2)	[m/s], [feet/s]
31	电阻	[Ohm]
32	电感	[mH]
33	转矩 3)	[Nm]
34	转矩常量 3)	[Nm/A]
35	电流控制器增益	[V/A]

<单位>	含义	物理单位
36	转速控制器增益 3)	[Nm/(rad*s)]
37	转速	[rpm]
42	功率	[kW]
43	弱电流	[µA]
46	低转矩 3)	[µNm]
48	千分比	-
49	-	[Hz/s]
65	流量	[l/min]
66	压力	[bar]
67	体积 3)	[cm^3]
68	距离增益 3)	[mm/(V*min)]
69	力控制器距离增益	[N/V]
155	螺距 3)	[mm/rev], [inch/rev]
156	螺距改变 3)	[mm/rev / rev], [inch/rev / rev]
1)这些物理单位取决于轴的类型，即线性轴或回转轴		
2)单位制转换 G70/G71（英制/公制） 在使用 G70/G71 进行了基本单位制（MD10240 $MN_SCALING_SYSTEM_IS_METRIC）转换后， 在写入/读取长度相关系统变量和用户变量时**不换算**其数值（实际值、缺省值和限值） G700/G710（英制/公制） 在使用 G700/G710 进行了基本单位制（MD10240 $MN_SCALING_SYSTEM_IS_METRIC）转换后， 在写入/读取长度相关系统变量和用户变量时**换算**其数值（实际值、缺省值和限值）		
3)这些变量**不**会自动换算为 NC 的当前单位制（公制或英制） 换算功能只能由用户或机床制造商实现。		

说明

单位换算导致的平面溢出

对于带和长度相关的物理单位的所有用户变量(GUD / PUD / LUD)，其内部存储格式为公制数据。 如果在 NCK 主运行中（如同步运行）过多地使用这些变量，在转换单位制时可能会导致插补平面的运算时间溢出，即报警 4240。

注意
单位之间的兼容性 在变量应用中（如赋值、比较和计算等），系统不会检查附属单位是否兼容。 必要的换算只能由用户或机床制造商实现。

参见

变量的一般信息 (页 17)

1.1.10 属性： 存取权限(APR, APW, APRP, APWP, APRB, APWB)

存取权限对应了以下在编程时给定的保护等级：

存取权限	保护等级
系统口令	0
机床制造商口令	1
服务口令	2
最终用户口令	3
钥匙开关位置 3	4
钥匙开关位置 2	5
钥匙开关位置 1	6
钥匙开关位置 0	7

用户变量的定义(DEF)

可以定义以下变量的存取权限(APR... / APW...)：

- 全局用户数据（GUD）

重新定义（REDEF）系统和用户变量

可以重新定义以下变量的存取权限(APR... / APW...)：

- 系统数据
  - 机床数据
  - 设定数据
  - FRAME
  - 过程数据
  - 主轴螺距误差补偿（EEC）
  - 垂度补偿(CEC)
  - 象限误差补偿(QEC)
  - 刀库数据
  - 刀具数据
  - 保护区
  - 可定向刀架
  - 运动链
  - 3D 保护区
  - 工作区域限制
  - ISO 刀具数据
- 用户数据
  - R 参数
  - 同步动作变量（$AC_MARKER，$AC_PARAM，$AC_TIMER）
  - 同步动作 GUD (SYG_xy[ ], 其中 x=R, I, B, A, C, S；y=S, M, U, 4, ..., 9)
  - EPS 参数
  - 刀具数据 OEM
  - 刀库数据 OEM
  - 全局用户变量（GUD）

说明

在重新定义时可以自由确定变量的存取权限，这些变量处于最低保护等级 7 和自有保护等级如 1（机床制造商）之间。

NC 语言指令的重新定义(REDEF)

可以重新定义以下 NC 语言指令的存取权限或执行权限(APX)：

- G 功能 / 位移条件

  文献：

  /PG/ 编程指南 基本原理，章节： G 功能 / 位移条件

- 预定义功能

  文献：

  /PG/ 编程指南 基本原理，章节： 预定义功能

- 预定义子程序调用

  文献：

  /PG/ 编程指南 基本原理，章节： 预定义子程序调用

- 执行同步动作时的 DO 指令
- 循环的程序名称

  循环必须保存在某一个循环目录中并且含有一个 PROC 指令。

零件程序和循环的存取权限(APRP, APWP)

不同的存取权限会对零件程序或循环的存取产生以下影响：

- APRP 0 / APWP 0
  - 执行零件程序时必须输入系统口令
  - 循环必须保存在目录 _N_CST_DIR（系统）中
  - 为使用目录 _N_CST_DIR，必须在机床数据 MD11160 $MN_ACCESS_EXEC_CST 中将执行权限设为“系统”
- APRP 1 / APWP 1 或 APRP 2 / APWP 2
  - 执行零件程序时必须输入机床制造商口令或服务口令
  - 循环必须保存在目录 _N_CMA_DIR（机床制造商）或 _N_CST_DIR 中
  - 为使用目录 _N_CMA_DIR 或 _N_CST_DIR，必须在以下机床数据中将执行权限至少设为“机床制造商”： MD11161 $MN_ACCESS_EXEC_CMA 或 MD11160 $MN_ACCESS_EXEC_CST。

- APRP 3 / APWP 3
  - 执行零件程序时必须输入最终用户口令
  - 循环必须保存在目录 N_CUS_DIR（用户）、 __N_CMA_DIR 或 _N_CST_DIR 中
  - 为使用目录 _N_CUS_DIR、_N_CMA_DIR 或 _N_CST_DIR，必须在以下机床数据中将执行权限至少设为“最终用户”：MD11162 $MN_ACCESS_EXEC_CUS、MD11161 $MN_ACCESS_EXEC_CMA 或 MD11160 $MN_ACCESS_EXEC_CST。
- APRP 4...7 / APWP 4...7
  - 在执行零件程序时必须设为钥匙开关位置 3 ... 0。
  - 循环必须保存在目录 N_CUS_DIR、 __N_CMA_DIR 或 _N_CST_DIR 中
  - 为使用目录 _N_CUS_DIR、_N_CMA_DIR 或 _N_CST_DIR，必须在以下机床数据中将执行权限至少设为相应的钥匙开关位置：MD11162 $MN_ACCESS_EXEC_CUS、MD11161 $MN_ACCESS_EXEC_CMA 或 MD11160 $MN_ACCESS_EXEC_CST。

BTSS 的存取权限(APRB, APWB)

存取权限指令(APRB, APWB)以相同的方式限制所有系统组件（HMI、PLC、外部计算机、EPS 服务等）上、通过操作面板接口对系统变量和用户变量的存取。

说明

HMI 本地存取权限

在修改系统数据的存取权限时必须注意，该权限必须和 HMI 装置上定义的存取权限一致。

存取属性 APR / APW

由于兼容性的原因，属性 APR 和 APW 隐式映射至属性 APRP / APRB 和 APWP / APWB：

- APR x ⇒ APRP x APRB x
- APW y ⇒ APWP y APWB y

设置 ACCESS 文件分配的存取权限

如果使用 ACCES 文件分配存取权限，应仍只在该 ACCESS 文件中重新定义系统/用户数据和 NC 语言指令的存取权限。 全局用户数据除外(GUD)。 对于此类数据的存取权限，必要时应在相应的定义文件中继续重新定义。

为保证存取保护的一致性，设置执行权限的机床数据和设置相应目录存取保护的机床数据必须相匹配。

请按照以下几个基本步骤执行：

- 创建需要的定义文件：
  - _N_DEF_DIR/_N_SACCESS_DEF
  - _N_DEF_DIR/_N_MACCESS_DEF
  - _N_DEF_DIR/_N_UACCESS_DEF
- 将此定义文件的写权限设为重新定义所需的值：
  - MD11170 $MN_ACCESS_WRITE_SACCESS
  - MD11171 $MN_ACCESS_WRITE_MACCESS
  - MD11172 $MN_ACCESS_WRITE_UACCESS

- 如果需要在循环中访问受保护单元，必须修改循环目录 _N_CST_DIR、_N_CMA_DIR 和 _N_CST_DIR 的执行权限和写权限。

  执行权限

  – MD11160 $MN_ACCESS_EXEC_CST
  – MD11161 $MN_ACCESS_EXEC_CMA
  – MD11162 $MN_ACCESS_EXEC_CUS

  写权限

  – MD11165 $MN_ACCESS_WRITE_CST
  – MD11166 $MN_ACCESS_WRITE_CMA
  – MD11167 MN_ACCESS_WRITE_CUS

  执行权限必须至少设为所用单元的最高保护等级。

  写权限必须至少设为和执行权限相同的保护等级。

- HMI 本地循环目录的写权限必须设为和 NC 本地循环目录相同的保护等级。

  文献

  /BAD/ 操作手册 HMI 高级型，

  章节： 操作区通讯 > 管理数据 > 修改属性

ACCESS 文件中的子程序调用

也可以在 ACCESS 文件中调用子程序（SPF 或 MPF 标识），以继续分级存取保护。 此时，子程序会采用待调用 ACCESS 文件的执行权限。

说明

在 ACCESS 文件中只能重新定义存取权限。 所有其他属性必须继续在相应的定义文件中写入或重新定义。

参见

变量的一般信息 (页 17)

1.1.11 可定义和可重新定义的属性一览

下表展示了各个数据类型以及相应的可以定义(DEF)的属性或重新定义(REDEF)的属性。

系统数据

数据类型	初始化值	极限值	物理单位	存取权限
机床数据	---	---	---	REDEF
设定数据	REDEF	---	---	REDEF
FRAME 数据	---	---	---	REDEF
过程数据	---	---	---	REDEF
主轴螺距误差补偿 (EEC)	---	---	---	REDEF
垂度补偿(CEC)	---	---	---	REDEF
象限误差补偿(QEC)	---	---	---	REDEF
刀库数据	---	---	---	REDEF
刀具数据	---	---	---	REDEF
保护区	---	---	---	REDEF
可定向刀架	---	---	---	REDEF
运动链	---	---	---	REDEF
3D 保护区	---	---	---	REDEF
工作区域限制	---	---	---	REDEF
ISO 刀具数据	---	---	---	REDEF

用户数据

数据类型	初始化值	极限值	物理单位	存取权限
R 参数	REDEF	REDEF	REDEF	REDEF
同步变量($AC_...)	REDEF	REDEF	REDEF	REDEF
同步 GUD (SYG_...)	REDEF	REDEF	REDEF	REDEF
EPS 参数	REDEF	REDEF	REDEF	REDEF
刀具数据 OEM	REDEF	REDEF	REDEF	REDEF
刀库数据 OEM	REDEF	REDEF	REDEF	REDEF
全局用户变量（GUD）	DEF / REDEF	DEF	DEF	DEF / REDEF
本地用户变量(PUD / LUD)	DEF	DEF	DEF	---

参见

变量的一般信息 (页 17)

1.1.12 定义和初始化数组变量（DEF，SET，REP）

功能

一个用户变量可以定义为 1 维~3 维数组(Array)。

- 1 维： DEF <数据类型> <变量名称>[<n>]
- 2 维： DEF <数据类型> <变量名称>[<n>，<m>]
- 3 维： DEF <数据类型> <变量名称>[<n>,<m>,<o>]

说明

STRING 数据类型的用户变量可以最大定义为 2 维数组。

数据类型

用户变量可以定义为以下类型的数组： BOOL, CHAR, INT, REAL, STRING, AXIS, FRAME

数组元素的赋值

可以在以下时间为数组元素赋值：

- 数组定义时（初始化值）
- 在程序执行过程中

可以通过以下方法赋值：

- 显式指定一个数组元素
- 显式指定一个数组元素为起始元素并给出值列表 (SET)
- 显式指定一个数组元素为起始元素并给出值列表以及重复的频率 (REP)

说明

不能向 FRAME 数据类型的用户变量分配初始化值。

句法(DEF)

```
DEF <数据类型> <变量名称>[<n>,<m>,<o>]
DEF  STRING[<字符串长度>] <变量名称>[<n>,<m>]
```

句法(DEF...=SET...)

使用值列表：

- 定义时：

  DEF <数据类型> <变量名称>[<n>,<m>,<o>]=SET(<值 1>,<值 2>,...)

  相同地：

  DEF <数据类型> <变量名称>[<n>，<m>，<o>]=（<值 1>，<值 2>，...）

  说明
  在通过值列表进行初始化时，可以选择给定 SET。

- 赋值时：

  <变量名称>[<n>，<m>，<o>]=SET（<值 1>，<值 2>，...）

句法(DEF...=REP...)

使用重复值

- 定义时：

  DEF<数据类型> <变量名称>[<n>，<m>，<o>]=REP（<值>）

  DEF<数据类型> <变量名称>[<n>，<m>，<o>]=REP（<值>,<数量_数组元素>）

- 赋值时：

  <变量名称>[<n>，<m>，<o>]=REP（<值>）

  <变量名称>[<n>，<m>，<o>]=REP（<值>，<数量_数组元素>）

含义

DEF:	变量定义指令
<数据类型>：	变量数据类型 取值范围： • 对于系统变量： BOOL, CHAR, INT, REAL, STRING, AXIS • 对于 GUD 或 LUD 变量： BOOL, CHAR, INT, REAL, STRING, AXIS, FRAME
<字符串长度>	STRING 数据类型下允许的最大字符数
<变量名称>：	变量名称
[<n>,<m>,<o>]:	数组长度或数组索引
<n>:	1 维的数组长度或数组索引 类型： INT（对于系统变量也为 AXIS） 取值范围： 最大数组长度： 65535 数组索引： 0 ≤ n ≤ 65534
<m>:	2 维的数组长度或数组索引 类型： INT（对于系统变量也为 AXIS） 取值范围： 最大数组长度： 65535 数组索引： 0 ≤ m ≤ 65534
<o>:	3 维的数组长度或数组索引 类型： INT（对于系统变量也为 AXIS） 取值范围： 最大数组长度： 65535 数组索引： 0 ≤ o ≤ 65534
SET:	通过给出的值列表赋值
(<值 1>,<值 2>,...):	值列表
REP:	通过给出的<值>赋值

<值>： 数组元素在带 REP 的初始化时具有的数值。

<数量_数组元素>： 使用给定<值>的数组元素的数量。 其他的数组元素取决于不同时间点：

- 数组定义时初始化：
  → 剩下的数组元素赋值为零
- 在程序运行过程中赋值：
  → 数组元素的当前值保持不变。

如果没有编程该参数，所有的数组元素都会分配到<值>。

如果参数为零，则取决于不同的时间点：

- 数组定义时初始化：
  → 所有元素预定为零
- 在程序运行过程中赋值：
  → 数组元素的当前值保持不变。

数组索引

在如 SET 或 REP 的赋值中，通过数组索引从右向左的循环构成数组元素的隐式顺序。

示例： 3 维数组的初始化，数组具有 24 个元素：

```
DEF INT FELD[2,3,4] = REP(1,24)
  FELD[0,0,0] = 1     第 1 个数组元素
  FELD[0,0,1] = 1     第 2 个数组元素
  FELD[0,0,2] = 1     第 3 个数组元素
  FELD[0,0,3] = 1     第 4 个数组元素
  ...
  FELD[0,1,0] = 1     第 5 个数组元素
  FELD[0,1,1] = 1     第 6 个数组元素
  ...
  FELD[0,2,3] = 1     第 12 个数组元素
  FELD[1,0,0] = 1     第 13 个数组元素
  FELD[1,0,1] = 1     第 14 个数组元素
```

```
...
FELD[1,2,3] = 1      第 24 个数组元素
```

相应地：

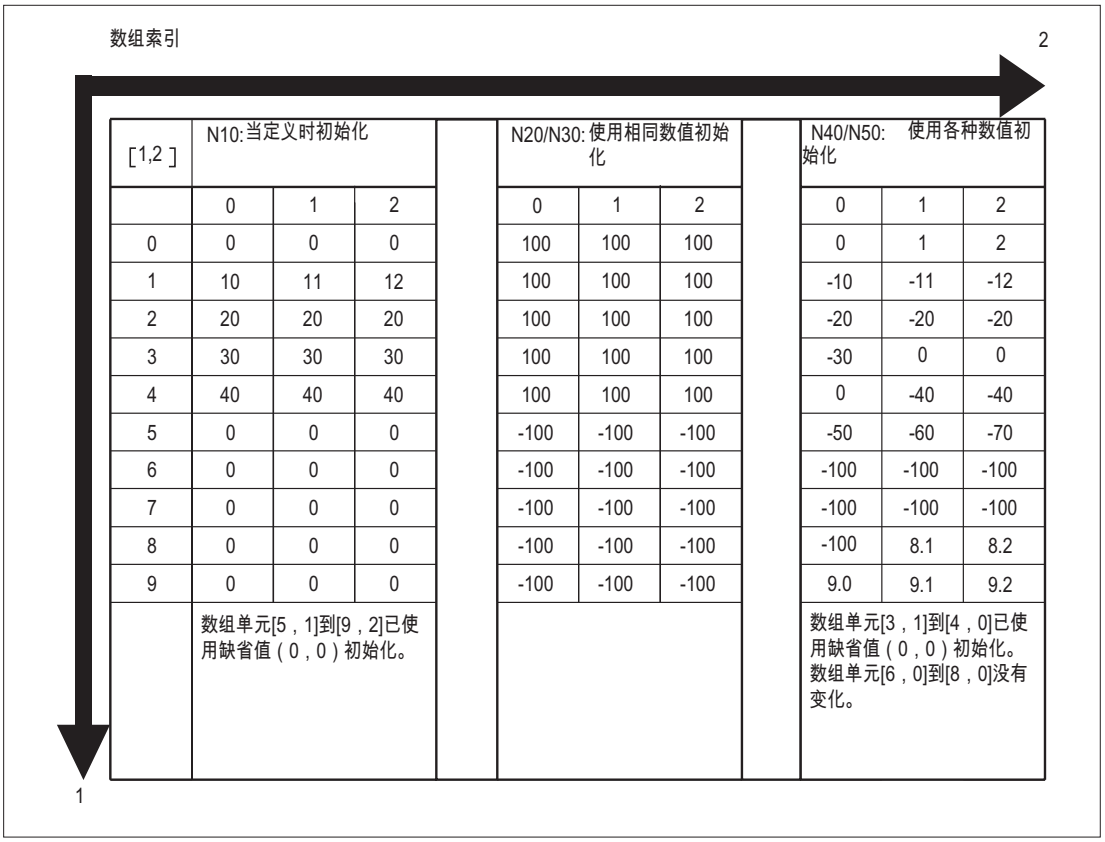
```
FOR n=0 TO 1
  FOR m=0 TO 2
    FOR o=0 TO 3
      FELD[n,m,o] = 1
    ENDFOR
  ENDFOR
ENDFOR
```

示例： 初始化整个变量数组

当前占用情况见插图。

程序代码

```
N10 DEF REAL FELD1[10,3]=SET(0,0,0,10,11,12,20,20,20,30,30,30,40,40,40,)
N20 FELD1[0,0]=REP(100)
N30 FELD1[5,0]=REP(-100)
N40 FELD1[0,0]=SET(0,1,2,-10,-11,-12,-20,-20,-20,-30, , , ,-40,-40,-50,-60,-70)
N50 FELD1[8,1]=SET(8.1,8.2,9.0,9.1,9.2)
```



[1,2]	N10:当定义时初始化				N20/N30:使用相同数值初始化				N40/N50: 使用各种数值初始化		
	0	1	2		0	1	2		0	1	2
0	0	0	0		100	100	100		0	1	2
1	10	11	12		100	100	100		-10	-11	-12
2	20	20	20		100	100	100		-20	-20	-20
3	30	30	30		100	100	100		-30	0	0
4	40	40	40		100	100	100		0	-40	-40
5	0	0	0		-100	-100	-100		-50	-60	-70
6	0	0	0		-100	-100	-100		-100	-100	-100
7	0	0	0		-100	-100	-100		-100	-100	-100
8	0	0	0		-100	-100	-100		-100	8.1	8.2
9	0	0	0		-100	-100	-100		9.0	9.1	9.2
	数组单元[5，1]到[9，2]已使用缺省值（0，0）初始化。								数组单元[3，1]到[4，0]已使用缺省值（0，0）初始化。 数组单元[6，0]到[8，0]没有变化。		

参见

定义和初始化数组变量（DEF，SET，REP）： 其它信息 (页 56)

变量的一般信息 (页 17)

1.1.13 定义和初始化数组变量（DEF，SET，REP）： 其它信息

更多信息(SET)

定义时进行初始化

- 从第 1 个数组元素开始，按照值列表中的值和写入的元素数量进行初始化。
- 值列表中没有显式指定值的数组元素(数值表中的空白)自动赋值 0。
- 对于 AXIS 数据类型的变量，值列表中不允许出现空白。
- 如果值列表包含的值大于数组元素的数量，则显示报警。

在程序执行中赋值

以上说明的定义规则同样适用于程序执行中的赋值。 此外，还有以下方法：

- 表达式也允许用作值列表的元素。
- 从编程的数组索引开始赋值。 从而根据需要赋值部分数组。

示例：

程序代码	注释
DEF INT FELD[5,5]	; 数组定义
FELD[0,0]=SET(1,2,3,4,5)	; 对前 5 个数组元素进行赋值[0,0] - [0,4]
FELD[0,0]=SET(1,2, , ,5)	; 对前 5 个数组元素[0,0] - [0,4]进行带空隙的赋值，数组元素 [0,2] 和 [0,3] = 0
FELD[2,3]=SET(VARIABLE,4*5.6)	; 带变量和表达式的赋值，自数组索引[2,3]起： [2,3] = VARIABLE [2,4] = 4 * 5.6 = 22.4

更多信息(REP)

定义时进行初始化

- 所有或指定数量的数组元素都会以给定值（常量）进行初始化。
- FRAME 数据类型的变量无法进行初始化。

示例：

程序代码	注释
DEF REAL varName[10]=REP(3.5,4)	; 数组定义，数组元素 0] ~[3] 以值 3.5 初始化

在程序执行中赋值

以上说明的定义规则同样适用于程序执行中的赋值。 此外，还有以下方法：

- 表达式也允许用作值列表的元素。
- 从编程的数组索引开始赋值。从而根据需要赋值部分数组。

示例：

程序代码	注释
DEF REAL varName[10]	; 数组定义
varName[5]=REP(4.5,3)	; 数组元素 [5] ~ [7] = 4.5
R10=REP(2.4,3)	; R-Parameter R10 ~ R12 = 2.4

程序代码	注释
DEF FRAME FRM[10]	; 数组定义
FRM[5]=REP(CTRANS(X,5))	; 数组元素[5] ~ [9] = CTRANS(X,5)

其它信息（概述）

轴机床数据的赋值

通常，轴机床数据的数组索引的数据类型为 AXIS。 在通过 SET 或 REP 进行的轴机床数据赋值中，这些数组索引会被忽略或跳过。

示例： 机床数据 MD36200 $MA_AX_VELO_LIMIT 的赋值

```
$MA_AX_VELO_LIMIT[1, AX1] = 设置(1.1, 2.2, 3.3)
```

相应的：

```
$MA_AX_VELO_LIMIT[1,AX1]=1.1
$MA_AX_VELO_LIMIT[2,AX1]=2.2
$MA_AX_VELO_LIMIT[3,AX1]=3.3
```

注意

轴机床数据的赋值

在通过 SET 或 REP 进行的轴机床数据赋值中，AXIS 数据类型的数组索引会被忽略或跳过。

内存需求

数据类型	每个元素的内存需求
BOOL	1 个字节
CHAR	1 个字节
INT	4 个字节
REAL	8 个字节
STRING	（字符串长度 + 1）个字节
FRAME	~ 400 个字节, 取决于轴数目
AXIS	4 个字节

参见

定义和初始化数组变量（DEF，SET，REP） (页 51)

1.1.14 数据类型

NC 中提供了以下数据类型：

数据类型	含义	取值范围
INT	带正负号的整数值	-2147483648 ... +2147483647
REAL	实数（根据 IEEE 为 LONG REAL）	±(~2.2*10^{-308} … ~1.8*10^{+308})
BOOL	真值 真（1）和 假（0）	1, 0
CHAR	ASCII-字符	ASCII 代码 0...255
STRING	定义长度的字符串	最多 200 个字符（无特殊字符）
AXIS	进给轴/主轴标识符	通道轴识别符
FRAME	用于静态坐标转换（平移、旋转、缩放和镜像）的几何参数	---

隐式数据类型转换

系统允许以下数据类型转换，并且在赋值和传递参数时隐式进行转换：

从 ↓/ 到 →	REAL	INT	BOOL
REAL	x	o	&
INT	x	x	&
BOOL	x	x	x
x： 无限制 o: 由于超出取值范围可能会丢失数据 ⇒ 报警； 取整： 小数值 ≥ 0.5 ⇒ 向上取值, 小数值 < 0,5 ⇒ 略去 &: 值 ≠ 0 ⇒ TRUE, 值 == 0 ⇒ FALSE			

参见

变量的一般信息 (页 17)

1.2 间接编程

1.2.1 间接编程地址

功能

在间接编程地址时，扩展的地址（索引）由一个合适的变量类型替代。

说明

在下列情况下，不能间接编程地址：

- N（程序段编号）
- L（子程序）
- 可调地址

  （例如，X[1] 代替 X1 是不允许的）

句法

<地址>[<索引>]

含义

<地址>[...]:	带扩展名（索引）的固定地址
<索引>:	变量，例如主轴编号，轴，....

示例

示例 1： 间接编程一个主轴编号

直接编程：

程序代码	注释
S1=300	; 主轴转速 300 转/分钟，编号为 1。

间接编程

程序代码	注释
DEF INT SPINU=1	; 定义 INT 型变量和赋值
S[SPINU]=300	; 主轴转速 300 转/分钟，其编号保存在变量 SPINU 中（在该示例中，主轴编号为 1）。

示例 2： 间接编程一个轴

直接编程：

程序代码	注释
FA[U]=300	; "U"轴的进给量为 300。

间接编程

程序代码	注释
DEF AXIS AXVAR2=U	; 定义一个 AXIS 型变量和赋值
FA[AXVAR2]=300	; 轴的进给量为 300，其地址名称保存在名称为 AXVAR2 的变量中。

示例 3： 间接编程一个轴

直接编程：

编程	注释
$AA_MM[X]	; 读取轴的测量探头一测量值（MKS）"X。

间接编程

程序代码	注释
DEF AXIS AXVAR3=X	; 定义一个 AXIS 型变量和赋值
$AA_MM[AXVAR3]	; 为轴读取测量探头一测量值（MKS），其名称保存在变量 AXVAR3 中。

示例 4： 间接编程一个轴

直接编程：

程序代码
X1=100 X2=200

间接编程

程序代码	注释
DEF AXIS AXVAR1 AXVAR2	; 定义两个 AXIS 型变量。
AXVAR1=(X1) AXVAR2=(X2)	; 分配轴名称。
AX[AXVAR1]=100 AX[AXVAR2]=200	; 运行轴，其地址名称保存在名为 AXVAR1 和 AXVAR2 的变量中。

示例 5： 间接编程一个轴

直接编程：

程序代码
G2 X100 I20

间接编程

程序代码	注释
DEF AXIS AXVAR1=X	; 定义一个 AXIS 型变量和赋值
G2 X100 IP[AXVAR1]=20	; 间接编程轴的中点数据，其地址名称保存在名为 AXVAR1 的变量中。

示例 6： 间接编程数组元素

直接编程：

程序代码	注释
DEF INT 数组1[4,5]	; 定义数组 1。

间接编程

程序代码	注释
DEFINE DIM1 AS 4	; 对于数组维，必须将数组大小设定为固定值。
DEFINE DIM2 AS 5	
DEF INT 数组[DIM1,DIM2]	
数组[DIM1-1,DIM2-1]=5	

示例 7： 间接调用子程序

程序代码	注释
CALL "L" << R10	; 调用其编号在 R10 中的程序（字符串级联）。

1.2.2 间接编程 G 代码

功能

通过间接编程 G 代码，可以进行有效的循环编程。

句法

G[<组]=<编号>

含义

G[...]:	带扩展名（索引）的 G 指令
<组>:	索引参数 G 功能组
	类型： INT
<编号>:	用于 G 代码编号的变量
	类型： INT 或 REAL

说明

通常只能间接编程非句法定义的 G 代码。

句法定义的 G 代码中只有 G 功能组 1 可采用间接编程。
而 G 功能组 2、3 和 4 中的句法定义 G 代码则不可以。

说明

在间接 G 代码编程中不允许进行算术计算。 必须在 G 代码间接编程前，在一个自身的零件程序行中进行必要的 G 代码编号计算。

示例

示例 1： 可设定的零点偏移（G 功能组 8）

程序代码	注释
N1010 DEF INT INT_VAR	
N1020 INT_VAR=2	
...	
N1090 G[8]=INT_VAR G1 X0 Y0	; G54
N1100 INT_VAR=INT_VAR+1	; G 代码计算
N1110 G[8]=INT_VAR G1 X0 Y0	; G55

示例 2： 平面选择（G 功能组 6）

程序代码	注释
N2010 R10=$P_GG[6]	; 读取 G 功能组 6 中的有效功能
...	
N2090 G[6]=R10	

文献

有关 G 功能组信息，见：
编程手册，基本原理：“G 功能组”章节

1.2.3 间接编程位置属性（GP）

功能

位置属性，例如轴位置的增量或绝对编程可以连同关键字 GP 一起间接编程为变量。

应用

位置属性的间接编程在 **替换循环**中有应用，因为，在这里，与将位置属性编程为关键字（例如 IC，AC，...）相比，有下列优点：

通过间接编程为变量，**无需**通过可能的位置属性转到的 CASE 指令。

句法

<定位指令>[<轴/主轴>]=
GP（<位置>，<位置属性）
<轴/主轴>=GP（<位置>，<位置属性）

含义

<定位指令>[]:	下列定位指令可以与关键字 GP 一起进行编程： POS, POSA, SPOS, SPOSA 此外，还可以编程： • 所有通道中现有的轴/主轴名称： <轴/主轴> • 可变的轴/主轴名称 AX
<轴/主轴>：	需要定位的轴/主轴
GP()：	用于定位的关键字
<位置>:	参数 1 轴/主轴作为常量或变量
<位置属性>：	参数 2 位置属性（例如位置开始模式）作为变量（例如 $P_SUB_SPOSMODE）或作为关键字（IC，AC ...）

由变量提供的值有下列含义：

值	含义	允许：
0	不改变位置属性	
1	AC	POS, POSA, SPOS, SPOSA, AX, 轴地址
2	IC	POS, POSA, SPOS, SPOSA, AX, 轴地址
3	DC	POS, POSA, SPOS, SPOSA, AX, 轴地址
4	ACP	POS, POSA, SPOS, SPOSA, AX, 轴地址
5	ACN	POS, POSA, SPOS, SPOSA, AX, 轴地址
6	OC	-
7	PC	-
8	DAC	POS, POSA, AX, 轴地址
9	DIC	POS, POSA, AX, 轴地址
10	RAC	POS, POSA, AX, 轴地址
11	RIC	POS, POSA, AX, 轴地址
12	CAC	POS, POSA
13	CIC	POS, POSA
14	CDC	POS, POSA
15	CACP	POS, POSA
16	CACN	POS, POSA

示例

对于一个有效的同步主轴耦合，在主动主轴 S1 和随动主轴 S2 之间通过 SPOS 指令在下面的用于主轴定位的替换循环的主程序中调用。

通过 N2230 中的指令进行定位：

```
SPOS[1]=GP($P_SUB_SPOSIT,$P_SUB_SPOSMODE)
SPOS[2]=GP($P_SUB_SPOSIT,$P_SUB_SPOSMODE)
```

从系统变量 $P_SUB_SPOSIT 中读取开始位置，从系统变量 $P_SUB_SPOSMODE 中读取位置开始模式。

程序代码	注释
N1000 PROC LANG_SUB DISPLOF SBLOF	
...	

程序代码	注释
N2100 IF($P_SUB_AXFCT==2)	
N2110	; 对于有效的同步主轴耦合，替换 SPOS / SPOSA / M19 指令
N2185 DELAYFSTON	; 开始停止延时段
N2190 COUPOF(S2,S1)	; 取消同步主轴耦合
N2200	; 定位引导主轴和随动主轴
N2210 IF($P_SUB_SPOS==TRUE) OR ($P_SUB_SPOSA==TRUE)	
N2220	; 用 SPOS 定位主轴
N2230 SPOS[1]=GP($P_SUB_SPOSIT,$P_SUB_SPOSMODE)	
SPOS[2]=GP($P_SUB_SPOSIT,$P_SUB_SPOSMODE)	
N2250 ELSE	
N2260	; 用 M19 定位主轴
N2270 M1=19 M2=19	; 定位引导主轴和随动主轴
N2280 ENDIF	
N2285 DELAYFSTOF	; 结束停止延时段
N2290 COUPON(S2,S1)	; 激活同步主轴耦合
N2410 ELSE	
N2420	; 查询其它替换
...	
N3300 ENDIF	
...	
N9999 RET	

边界条件

- 不能够在同步动作中间接编程位置属性。

文献

功能手册 基本功能；BAG，通道，程序运行，复位特性（K1），
章节： 由子程序替换 NC 功能

1.2.4 间接编程零件程序行（EXECSTRING）

功能

使用零件程序指令 EXECSTRING 可将之前生成的 String 变量作为零件程序行执行。

句法

EXECSTRING 须编程在一个单独的零件程序行中：
EXECSTRING (<字符串变量>)

含义

EXECSTRING:	将 String 变量作为零件程序行执行的指令
<字符串变量>：	包含了原本需要执行的零件程序的 STRING 变量

说明

通过 EXECSTRING 可取消所有可在零件程序段落中编程的零件程序结构。 PROC 和 DEF 指令除外，在 INI 和 DEF 文件中的应用通常也不可。

示例

程序代码	注释
N100 DEF STRING[100] BLOCK	; 定义包含需要执行的零件程序行的 String 变量。
N110 DEF STRING[10] MFCT1="M7"	
...	
N200 EXECSTRING(MFCT1 << "M4711")	; 执行零件程序行"M7 M4711"。
...	
N300 R10=1	
N310 BLOCK="M3"	
N320 IF(R10)	
N330 程序块 = 程序块 << MFCT1	
N340 ENDIF	
N350 EXECSTRING(BLOCK)	; 执行零件程序行"M3 M7"。

1.3 运算功能

功能

计算功能主要应用于 R 参数和实数型变量（或常量和功能）。 整数型和字符型也是允许的。

运算符/计算功能	含义
+	加法
-	减法
*	乘法
/	除法 **注意：** (（INT 型）/（INT 型）=（REAL 型）；示例：3/4 = 0.75 3/4 = 0.75
DIV	除法,用于变量类型整数型和实数型 **注意：** （INT 型）DIV（INT 型）=（INT 型）；示例： 3 DIV 4 = 0
MOD	取模除法（仅用于 INT 型）提供一个 INT 除法的余数 示例： 3 MOD 4 = 3
:	串运算符（在框架变量时）
Sin()	正弦
COS()	余弦
TAN()	正切
ASIN()	反正弦
ACOS( )	反余弦
ATAN2 (,)	反正切 2
SQRT()	平方根
ABS()	总计
POT()	2. 幂（平方）

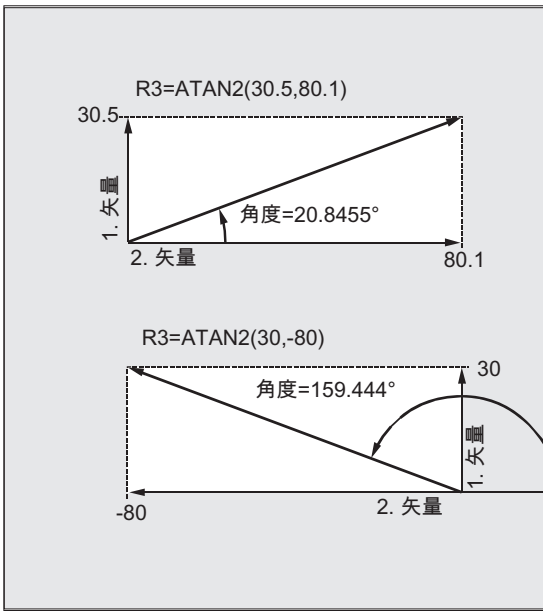
TRUNC()	整数部分 比较命令时的精确度，可使用 TRUNC 设置（参见“比较错误的精确度修正 (TRUNC) (页 75)”）
ROUND()	取整
LN()	自然对数
EXP()	指数函数
MINVAL ()	两变量中的较小值 （参见“参见“最大变量、最小变量和变量区域(MINVAL, MAXVAL, BOUND)” (页 77)”）
MAXVAL ()	两变量中的较大值 （参见“参见“最大变量、最小变量和变量区域(MINVAL, MAXVAL, BOUND)” (页 77)”）
BOUND ()	已定义值域中的变量值 （参见“参见“最大变量、最小变量和变量区域(MINVAL, MAXVAL, BOUND)” (页 77)”）
CTRANS()	偏移
CROT ()	旋转
CSCALE ()	比例修改
CMIRROR ()	镜像

编程

计算功能采用通常使用的数学运算法则。 在处理中需优先处理的用圆括号给出。 对于三角函数和它的反函数其单位是度(直角=90）。

示例

示例 1： ATAN2



ATAN2 计算功能可以从两个互相垂直的矢量计算出总矢量的角度。

结果位于四个象限的范围内 (-180°< 0 < +180°)。

角度是指在正方向的第 2 个数值。

示例 2： 初始化整个变量数组

程序代码	注释
R1=R1+1	; 新的 R1 = 旧的 R1 +1
R1=R2+R3 R4=R5-R6 R7=R8*R9	
R10=R11/R12 R13=SIN(25.3)	
R14=R1*R2+R3	; 四则运算法则
R14=(R1+R2)*R3	; 首先运算括号中的数字
R15=SQRT(POT(R1)+POT(R2))	; 首先计算内括号： R15 = （R1+R2）的平方根
RESFRAME=FRAME1:FRAME2 FRAME3=CTRANS(…):CROT(…)	; 使用链接算子将括号 与另一个得出结果的括号联系起来或者 给括号中的分量赋值。

1.4 比较运算和逻辑运算

功能

比较运算 例如可以用来表达某个跳转条件。 完整的表达式也可以进行比较。

比较函数可用于 CHAR、INT、 REAL 和 BOOL 型的变量。 对于 CHAR 型变量，比较代码值。
对于 STRING、AXIS 和 FRAME 可以为： **==** 和 **<>**，也可在同步动作中用于运算 STRING 型的变量。

比较运算的结果始终为 BOOL 型。

逻辑运算 用来将真值联系起来。

逻辑运算只能用于 BOOL 型变量。 通过内部类型转换也可将其用于 CHAR、INT 和 REAL 数据类型。

对于逻辑（布尔）运算而言，适用数据类型为 BOOL、CHAR、INT 和 REAL：

- 0 表示： FALSE
- 等于 0 相当于： TRUE

位逻辑运算符

使用 CHAR 和 INT 型变量也可进行逐位逻辑运算。 如果有这种情况，类型转换自动进行。

编程

比较运算符	含义
==	相同于
<>	不等
>	大于
<	小于
>=	大于等于
<=	小于等于

逻辑运算符	含义
AND	与
OR	或
NOT	非
XOR	异－或

逐位逻辑运算符	含义
B_AND	位方式“与”
B_OR	位方式“或”
B_NOT	位方式“非”
B_XOR	逐位式“异－或”

说明

在算术表达式中可以通过圆括号确定所有运算的顺序并且由此脱离原来普通的优先计算规则。

说明

在布尔的操作数和运算符之间必须加入空格。

说明

运算符 B_NOT 只与一个运算域有关。 它位于运算符之后。

示例

示例 1： 比较运算符

```
IF R10>=100 GOTOF 目标
```

或

```
R11=R10>=100
IF R11 GOTOF 目标
```

R10>=100 的比较结果首先存储在 R11 中。

示例 2： 逻辑运算符

```
IF (R10<50) AND ($AA_IM[X]>=17.5) GOTOF 目标
```

或

```
IF NOT R10 GOTOB START
```

NOT 只与一个运算域有关。

示例 3： 逐位逻辑运算符

```
IF $MC_RESET_MODE_MASK B_AND 'B10000' GOTOF ACT_PLANE
```

1.5 比较错误的精确度修正 (TRUNC)

功能

TRUNC 指令用来截断与一个精度系数相乘后的运算数。

比较操作时的可设定精度

实数型零件程序参数内部用 64 位的 IEEE 格式描述。 这种显示形式不能构成精确的十进制数，在与理想计算的数值进行比较时可能会带来不好的结果。

相对相等性

为了使这种描述所带来的不精确性不影响程序流程，在比较指令中不检测绝对奇偶性，而是检测一个相对相等性。

句法

比较错误时的精度补偿

```
TRUNC (R1*1000)
```

含义

TRUNC: 去除小数点后位数

所考虑的相对相等性为 10^{-12} 当

- 相等性： (==)
- 不相等性： (<>)
- 大于等于： (>=)
- 小于等于： (<=)
- 大于/小于： (><)绝对相等
- 大于： (>)
- 小于： (<)

兼容性

出于兼容性考虑，在 (>) 和 (<) 时通过设置机床数据 MD10280 $MN_PROG_FUNCTION_MASK Bit0 = 1 可以取消相对相等性的检测。

说明

与实数型数据比较时，由于以上原因一般会出现一定的误差。 当出现不可接受的偏差时，必须另选 INTEGER 型计算，方法是将运算数和一个精度系数相乘，然后再使用 TRUNC 截断。

同步动作

所描述的比较指令性能也适用于同步动作。

示例

示例 1： 精度检查

程序代码	注释
N40 R1=61.01 R2=61.02 R3=0.01	; 初始值分配
N41 IF ABS(R2-R1) > R3 GOTOF FEHLER	; 到此为止将可执行跳转
N42 M30	; 程序结束
N43 出错： SETAL(66000)	;
R1=61.01 R2=61.02 R3=0.01	; 初始值分配
R11=TRUNC(R1*1000) R12=TRUNC(R2*1000) R13=TRUNC(R3*1000)	; 精度补偿
IF ABS(R12-R11) > R13 GOTOF 出错	; 不再执行跳转
M30	; 程序结束
FEHLER: SETAL(66000)	;

示例 2： 得出并且分析两个运算数的商

程序代码	注释
R1=61.01 R2=61.02 R3=0.01	; 初始值分配
IF ABS((R2-R1)/R3)-1) > 10EX-5 GOTOF FEHLER	; 不执行跳转
M30	; 程序结束
FEHLER:SETAL(66000)	;

1.6 参见“最大变量、最小变量和变量区域(MINVAL, MAXVAL, BOUND)”

功能

使用指令 MINVAL 和 MAXVAL 可以比较两个变量的值。 其中的较小值（采用 MINVAL 时）或较大值（采用 MAXVAL 时）会作为结果返回。

使用指令功能 BOUND 可以检查，待检变量的值是否在定义的值域内。

句法

```
<较小值>=MINVAL(<变量 1>,<变量 2>)
<较大值>=MAXVAL(<变量 1>,<变量 2>)
<返回值>=<BOUND>(<最小>,<最大>,<待检变量>)
```

含义

MINVAL:	确定两个变量即(<变量 1>, <变量 2>)中的**较小**值
<较小值>:	指令 MINVAL 的结果变量 设为较小的变量值。
MAXVAL:	确定两个变量即(<变量 1>, <变量 2>)中的**较大**值
<较大值>：	指令 MAXVAL 的结果变量 设为较大的变量值。
BOUND:	检查（<待检变量>）是否在已定义的值域中。
<最小>:	用于定义取值范围中最小值的变量
<最大>:	用于定义取值范围中最大值的变量
<返回值>:	指令 BOUND 的结果变量 如果待检变量的值在定义的取值范围内，结果变量会设为此待检变量。 如果待检变量的值大于最大值，结果变量会设为取值范围的最大值。 如果待检变量的值小于最小值，结果变量会设为取值范围的最小值。

说明

MINVAL、MAXVAL 和 BOUND 也可以在同步动作中编程。

说明

值相等时的属性

两值相等时，MINVAL/MAXVAL 返回该等值。 BOUND 再次返回待检变量的值。

示例

程序代码	注释
DEF REAL rVar1=10.5, rVar2=33.7, rVar3, rVar4, rVar5, rValMin, rValMax, rRetVar	
rValMin=MINVAL(rVar1,rVar2)	; rValMin 设为值 10.5。
rValMax=MAXVAL(rVar1,rVar2)	; rValMax 设为值 33.7。
rVar3=19.7	
rRetVar=BOUND(rVar1,rVar2,rVar3)	; rVar3 在极限值范围内，rRetVar 设为 19.7。
rVar3=1.8	
rRetVar=BOUND(rVar1,rVar2,rVar3)	; rVar3 小于最小值，rRetVar 设为 10.5。
rVar3=45.2	
rRetVar=BOUND(rVar1,rVar2,rVar3)	; rVar3 大于最大值，rRetVar 设为 33.7。

1.7 运算的优先级

功能

每个运算符都被赋予一个优先级。 在计算一个表达式时，有高一级优先权的运算总是首先被执行。 在优先级相同的运算中，运算由左到右进行。

在算术表达式中可以通过圆括号确定所有运算的顺序并且由此脱离原来普通的优先计算规则。

运算的顺序

从最高到最低优先级

1.	NOT, B_NOT	非，位方式非
2.	*, /, DIV, MOD	乘除
3.	+, –	加减
4.	B_AND	位方式“与”
5.	B_XOR	位方式“异或”
6.	B_OR	位方式“或”
7.	AND	与
8.	XOR	异或
9.	OR	或
10.	<<	字符串的链接,结果类型字符串
11.	==, <>, >, <, >=, <=	比较运算符

说明

级联运算符“：”在表达式中不能与其它的运算符同时出现。 因此这种运算符不要求划分优先级。

如果-语句举例

```
If (otto==10) and (anna==20) gotof end
```

1.8 可能有的类型转换

功能

赋值时的类型转换

数值常量、变量或者给某个变量赋值的表达式必须与该变量的类型相容。一旦变量给出，在赋值时类型自动转换。

可能的类型转换

到	REAL	整数	BOOL	CHAR	字符串	AXIS	FRAME
从							
REAL	是	是*	是 1)	是*	–	–	–
整数	是	是	是 1)	是 2)	–	–	–
BOOL	是	是	是	是	是	–	–
CHAR	是	是	是 1)	是	是	–	–
字符串	–	–	是 4)	是 3)	是	–	–
AXIS	–	–	–	–	–	是	–
FRAME	–	–	–	–	–	–	是

说明

* 从实数型到整数型的转换中，小数值>= 0.5 时向上园整，否则舍去(ROUND 功能)。

1) 值<> 0 对应于 TRUE,值== 0 对应于 FALSE

2) 如果数值在允许的值范围内

3) 如果只有一个字符

4) 字符串长度 0 = >FALSE,否则 TRUE

说明

如果在转换中一个值大于目标范围，就会出现出错提示。

如果表达式中出现混合类型，系统会自动进行类型匹配。类型转换也可用于同步动作中，见章节“同步运行动作，隐式类型转换”。

1.9 字符串运算

字符串运算

除了典型的运算“赋值”和“比较”之外，可以有下列字符串运算：

- 类型转换到字符串 (AXSTRING)
- 从字符串 (NUMBER, ISNUMBER, AXNAME) 类型转换
- 字符串的链接 (<<)
- 大小写字母转换 (TOLOWER, TOUPPER)
- 确定一个字符串的长度 (STRLEN)
- 在字符串中查找字符/字符串 (INDEX, RINDEX, MINDEX, MATCH)
- 部分字符串的选择 (SUBSTR)
- 选择一个单字符 (STRINGVAR, STRINGFELD)

字符 0 的特别意义

在系统内部，字符 0 被视为一个字符串的结束标志。 如果一个字符被一个 0 字符代替，那么这个字符串就被缩短了。

示例：

程序代码	注释
`DEF STRING[20] STRG="存在轴"`	
`STRG[6] = "X"`	
`MSG(STRG)`	; 出现“X 轴停止”的提示。
`STRG[6] = 0`	
`MSG(STRG)`	; 出现“轴”的提示。

1.9.1 类型转换到字符串 (AXSTRING)

功能

通过功能“向字符串类型转换”，不同类型的变量可以用作一个信息（MSG）的组成部分。

使用运算符<<时，隐含适用于数据类型INT，REAL，CHAR和BOOL（参见“字符串的链接 (<<) (页 84)”）。

一个 INT 值会被转换为可读形式。 在显示实数值时会给出小数点后 10 位。

通过指令 AXSTRING 可以将变量由类型 AXIS 转换到 STRING。

句法

```
<STRING_ERG> = << <bel._Typ>
<STRING_ERG> = AXSTRING(<轴名称>)
```

含义

<STRING_ERG>:	用于类型转换结果的变量 类型： STRING
<bel._Typ>:	变量类型 INT, REAL, CHAR, STRING 和 BOOL
AXSTRING:	指令 AXSTRING 作为字符串提供所给出的轴名称。
<轴名称>:	轴名称变量 类型： AXIS

说明

FRAME 变量不能被转换。

示例

示例 1：

```
MSG("位置:"<<$AA_IM[X])
```

示例 2： AXSTRING

程序代码	注释
DEF STRING[32] STRING_ERG	
STRING_ERG=AXSTRING(X)	; STRING_ERG == "X"

1.9.2 从字符串 (NUMBER, ISNUMBER, AXNAME) 类型转换

功能

通过指令 NUMBER 可以实现从 STRING 到 REAL 的转换。 转换可行性可以通过指令 ISNUMBER 检测。

通过指令 AXNAME 转换一个字符串到数据类型 AXIS。

句法

```
<REAL_ERG>=NUMBER("<String>")
<BOOL_ERG>=ISNUMBER("<String>")
<AXIS_ERG>=AXNAME("<String>")
```

含义

NUMBER: 指令 NUMBER 将由<字符串>格式显示的数值作为实数值送回。

<字符串>: 要转换的类型 STRING 的变量

<REAL_ERG>: 通过 NUMBER 类型转换结果变量

类型： REAL

ISNUMBER: 通过指令 ISNUMBER 可以检测这个<字符串>是否能被转换为一个有效的数字。

<BOOL_ERG>: 通过 ISNUMBER 查询结果变量

类型： BOOL

值： TRUE 当<字符串>显示一个根据语言规则有效的实数时，ISNUMBER 显示值为 TRUE。

FALSE 当 ISNUMBER 给出值 FALSE 时，就会在调用带有相同<字符串> 的 NUMBER 时触发报警。

AXNAME: 指令 AXNAME 转换规定的 <字符串> 到一个轴名称内。

提示：
如果该 <字符串> 没有分配到设计的轴名称，则触发报警。

<AXIS_ERG>: 通过 AXNAME 类型转换结果变量

类型： AXIS

示例

程序代码	注释
DEF BOOL BOOL_ERG	
DEF REAL REAL_ERG	
DEF AXIS AXIS_ERG	
BOOL_ERG=ISNUMBER("1234.9876Ex-7")	; BOOL_ERG == TRUE
BOOL_ERG=ISNUMBER("1234XYZ")	; BOOL_ERG == FALSE
REAL_ERG=NUMBER("1234.9876Ex-7")	; REAL_ERG == 1234.9876Ex-7
AXIS_ERG=AXNAME("X")	; AXIS_ERG == X

1.9.3 字符串的链接 (<<)

功能

功能“字符串链接”使单个的字符串可以组合在一起。

通过运算符“**<<**”实现链接。 这个运算符适用于所有基本类型 CHAR，BOOL，INT，REAL 和 STRING 的组合，变成目标类型 STRING。 必需的数据类型转换按照现有的规则进行。

句法

```
<bel._Typ> << <bel._Typ>
```

含义

<bel._Typ>: 类型 CHAR, BOOL, INT, REAL 或 STRING 的变量

<< : 变量链接运算 (**<bel._Typ>**) 可以得到一个合并的字符串 (类型 STRING)。

该运算符也可单独作为“一元”变量使用。 这样可以执行一个明确的、到字符串类型的转换（FRAME 和 AXIS 不可用）：

```
<< <bel._Typ>
```

比如，自文本列表组成一个信息或一个命令，并且插入参数（如一个模块名）：

```
MSG(STRG_TAB[LOAD_IDX]<<BAUSTEIN_NAME)
```

小心
在字符串级联时，中间结果不可以超过最大字符串长度。

说明

类型 FRAME 和 AXIS 类型不能使与运算符“**<<**”一起使用。

示例

示例 1： 字符串的链接

程序代码	注释
DEF INT IDX=2	
DEF REAL VALUE=9.654	
DEF STRING[20] STRG="INDEX:2"	
IF STRG=="Index:"<<IDX GOTOF NO_MSG	
MSG("Index:"<<IDX<<"/Wert:"<<VALUE)	; Anzeige: "Index:2/Wert:9.654"
NO_MSG:	

示例 2： 明确的类型转换通过 <<

程序代码	注释
DEF REAL VALUE=3.5	
<<VALUE	; 将规定的变量从类型 REAL 转换到类型 STRING。

1.9.4 大小写字母转换 (TOLOWER, TOUPPER)

功能

功能“大小写字母转换”允许一个字符串的全部字母一起转换到另一种统一的表示方式。

句法

```
<STRING_ERG>=TOUPPER("<String>")
<STRING_ERG>=TOLOWER("<String>")
```

含义

TOUPPER:	通过指令 TOUPPER 将一个字符串的全部字母都转换为 **大写**字母。
TOLOWER:	通过指令 TOLOWER 将一个字符串的全部字母都转换为 **小写**字母。
<字符串>:	要转换的字符串 类型： STRING
<STRING_ERG>:	类型转换结果变量 类型： STRING

示例

因为也有可能与操作界面上的用户输入发生冲突，可以使用大写或者小写字母来统一显示结果：

程序代码

```
DEF STRING [29] STRG
...
IF "LEARN.CNC"==TOUPPER(STRG) GOTOF LOAD_LEARN
```

1.9.5 确定一个字符串的长度 (STRLEN)

功能

通过指令 STRLEN 可以确定字符串的长度。

句法

```
<INT_ERG>=STRLEN("<STRING>")
```

含义

STRLEN:	通过指令 STRLEN 可以确定指定字符串的长度。 它可以返回字符的数量，即从字符串开头数起，但不包括 0 的字符数量。
<字符串>:	要确定长度的字符串 类型： STRING
<INT_ERG>:	类型转换结果变量 类型： INT

示例

该功能连同单字符访问一起可以确定一个字符串的末尾：

程序代码

```
IF(STRLEN (模块名称) > 10) GOTOF 出错
```

1.9.6 在字符串中查找字符/字符串 (INDEX, RINDEX, MINDEX, MATCH)

功能

利用此功能，可以在后面一个字符串中查找单个字符或者一个字符串。 查找结果说明：在字符串的一个位置找到需要查找的字符/字符串。

句法

INT_ERG=INDEX(STRING,CHAR)；**结果类型： INT**

INT_ERG=RINDEX(STRING,CHAR)；**结果类型：INT**

INT_ERG=MINDEX(STRING,STRING)；**结果类型：INT**

INT_ERG=MATCH(STRING,STRING)；**结果类型：INT**

符号语义

查找功能： 它会把所查找字符串（第一个参数）中的位置送回。 如果找不到字符或字符串，就会送回数值-1。 第一个字符位置为 0。

含义

INDEX: 在第一个参数中寻找作为第二个参数的字符（从前面）。

RINDEX: 在第一个参数中寻找作为第二个参数的字符（从后面）。

MINDEX: 相当于 INDEX 功能，不同之处在于，它传送一个字符列表作为字符串，其中传送了第一个被发现的字符的索引。

MATCH: 在一个字符串中寻找一个字符串。

这样字符串可以按照一定的标准进行分解，大约是在空格或路径分隔符的位置("/")。

示例

将一个输入分解成路径名称和模块名称

程序代码	注释
DEF INT PFADIDX, PROGIDX	
DEF STRING[26] EINGABE	
DEF INT LISTIDX	
EINGABE = "/_N_MPF_DIR/_N_EXECUTE_MPF"	
LISTIDX = MINDEX (EINGABE, "M,N,O,P") + 1	; 将 3 作为 LISTIDX 中的值送回；因为 "N" 为参数 EINGABE 中的第一个字符（从选择列表前面数起）。
PFADIDX = INDEX (EINGABE, "/") +1	; 由此： PFADIDX = 1 有效
PROGIDX = RINDEX (EINGABE, "/") +1	; 由此： PROGIDX = 12 有效
	借助下一段落中引用的功能 SUBSTR 将
	;变量 EINGABE 分解成"路径"和"模块"：
VARIABLE = SUBSTR (EINGABE, PFADIDX, PROGIDX-PFADIDX-1)	; 然后给出"_N_MPF_DIR"
VARIABLE = SUBSTR (EINGABE, PROGIDX)	; 然后给出"_N_EXECUTE_MPF"

1.9.7 部分字符串的选择 (SUBSTR)

功能

这个功能允许一个部分字符串从一个字符串中生成。 为此需指定第一个字符的索引，有时还需指定所需长度。 如果没有说明长度，则指的就是剩余字符串。

句法

STRING_ERG = SUBSTR (STRING,INT) ; **结果类型：INT**

STRING_ERG = SUBSTR(STRING,INT, INT) ; **结果类型：INT**

符号语义

在第一种情况，部分字符串从通过第二个参数确定的位置起到结束都被返还。

在第二种情况，结果字符串的长度被限制在通过第三个参数给出最大值之内。

如果开始位置位于字符串结尾之后，空字符串“ ”被返还。

如果开始位置或长度为非，触发警报。

示例

程序代码	注释
DEF STRING [29] ERG	
ERG = SUBSTR ("QUITTUNG:10 至 99", 10, 2)	; 由此： ERG == "10"有效

1.9.8 选择一个单字符 (STRINGVAR, STRINGFELD)

功能

这个功能允许选择一个字符串的单个字符。 它不仅适用于正在读取的数据，也适用于正在写入的数据。

句法

CHAR_ERG = STRINGVAR [IDX] ; **结果类型： CHAR**

CHAR_ERG = STRINGFELD [IDX_FELD, IDX_CHAR] ; **结果类型：CHAR**

符号语义

从指定位置上的字符串读取/写入字符。 如果指定的位置为非或大于这个字符串，触发警报。

信息举例：

在一个已经完成的字符串中使用一个轴标识。

程序代码	注释
DEF STRING [50] MELDUNG = "轴 n 已经到达位置"	
MELDUNG [6] = "X"	
MSG (MELDUNG)	; 发送信息"轴 X 已经到达位置"

参数

单字符存取仅可针对用户自定义变量(LUD 数据、GUD 数据、PUD 数据)。

此外在调用一个子程序时只可以对 Call-By-Value 值调用型参数进行存取。

示例

示例 1： 对某个系统数据、机床数据进行单字符存取

程序代码	注释
DEF STRING [50] STRG	
DEF CHAR QUITTUNG	
…	
STRG = $P_MMCA	
QUITTUNG = STRG [0]	; 确认组件的运用

示例 2： 对 Call-By-Reference 引用调用参数进行单字符存取

程序代码	注释
DEF STRING [50] STRG	
DEF CHAR CHR1	
EXTERN UP_CALL (VAR CHAR1)	; Call-By-Reference 引用调用参数!
…	
CHR1 = STRG [5]	
UP_CALL (CHR1)	; Call-By-Reference 引用调用
STRG [5] = CHR1	

1.10 程序跳转和分支

1.10.1 跳回到程序开始 (GOTOS)

功能

通过指令 GOTOS 可以用于程序重复时跳回到某个主程序或者子程序的开始处。

通过机床数据可以设置在每次跳回时在程序开始处：

- 程序运行时间设置为“0”。
- 工件计数提高值“1”。

句法

GOTOS

含义

GOTOS: 带有程序开始跳转目标的跳转指令。

该执行通过 NC/PLC 接口信号控制：

DB21, ... DBX384.0 (调节程序分支)

值：	含义：
0	没有跳回到程序开始。 程序加工在 GOTOS 后继续进行下一个零件程序段。
1	跳回到程序开始。 重复零件程序。

边界条件

- GOTOS 触发内部一个 STOPRE（预处理停止）。
- 对于一个带有数据定义（LUD 变量）的零件程序，通过 GOTOS 根据定义段跳转到第一个程序段，即不重新执行数据定义。 为此定义的变量保持了 GOTOS 程序段中达到的值，并且不跳回到定义段中编程的标准值。
- 在同步动作和工艺周期中不提供指令 GOTOS 。

示例

程序代码	注释
N10 ...	; 程序开始。
...	
N90 GOTOS	; 跳转到程序开始。
...	

1.10.2 程序跳转到跳转标记处 (GOTOB, GOTOF, GOTO, GOTOC)

功能

在一个程序中可以设置跳转标记（标签），通过指令 GOTOF, GOTOB, GOTO 或 GOTOC 可以在同一个程序内从其他位置跳转到跳转标记处。 然后通过该指令继续程序加工，该指令直接跟随在跳转标记后。 因此可以在程序内实现分支。

除了跳转标记外，主程序段号码和旁支程序段号码也可以作为跳转目标。

如果在跳转指令前存在跳转条件（IF ...），则仅在满足跳转条件情况下才进行程序跳转。

句法

```
GOTOB <跳转目标>
IF <跳转条件> = TRUE GOTOB <跳转目标>
GOTOF <跳转目标>
IF <跳转条件> = TRUE GOTOF <跳转目标>
GOTO <跳转目标>
IF <跳转条件> = TRUE GOTO <跳转目标>
GOTOC <跳转目标>
IF <跳转条件> = TRUE GOTOC <跳转目标>
```

含义

GOTOB:	以程序开始方向的带跳转目标的跳转指令。
GOTOF:	以程序末尾方向的带跳转目标的跳转指令。
GOTO:	带跳转目标查找的跳转指令。 查找先向程序末尾方向进行，然后再从程序开始处进行查找。

GOTOC:	与 GOTO 有区别的是，报警 14080“跳转目标未找到”被抑制。 这表示，在跳转目标查找没有结果情况下不中断程序加工，而以指令 GOTOC 下面的程序行继续进行。
<跳转目标>:	跳转目标参数 允许的说明有：
	<跳转标记**>**: 跳转目标是程序中带有用户定义名称的跳转标记： **<**跳转标记>：
	<程序段号码**>**: 主程序段号或者分支程序段号作为跳转目标（比如： 200, N300)
	STRING 类型变量： 跳转目标变量。 变量提供一个跳转标记或者一个程序段号。
IF:	用于编制跳转条件的关键字。 跳转条件允许使用所有的比较运算和逻辑运算（结果： TRUE 或者 FALSE)。 如果这种运算的结果为 TRUE，则执行程序跳转。

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（“：”）。

边界条件

- 跳转目标可能仅仅是一个带跳转标记或者程序段号的程序段，它们位于程序**内**。
- 不带跳转条件的跳转指令必须在一个独立的程序段中编程。 带跳转条件的跳转指令不适用于这类限制。 在一个程序段中可以编制几个跳转指令。
- 在不带跳转条件的跳转指令的程序中，程序结束 M2/M30 并不一定必须位于程序结束处。

示例

示例 1： 跳转到跳转标记

程序代码	注释
N10 …	
N20 GOTOF Label_1	; 向程序末尾方向跳转到跳转标记"Label_1"。
N30 …	
N40 Label_0: R1=R2+R3	; 设置了跳转标记"Label_0"。
N50 …	
N60 Label_1:	; 设置了跳转标记"Label_1"。
N70 …	
N80 GOTOB Label_0	; 向程序开始方向跳转到跳转标记"Label_0"。
N90 …	

示例 2： 间接跳转到程序段号

程序代码	注释
N5 R10=100	
N10 GOTOF "N"<<R10	; 跳转到程序段号码为 R10 的程序段。
...	
N90 ...	
N100 ...	; 跳转目标
N110 ...	
...	

示例 3： 跳转到可变的跳转目标

程序代码	注释
DEF STRING[20] ZIEL	
ZIEL = "Marke2"	
GOTOF ZIEL	; 向程序末尾方向跳转到可变的跳转目标 ZIEL。
标记 1： T="孔 1"	
...	
标记 2： T="孔 2"	; 跳转目标
...	

示例 4： 带跳转条件的跳转

程序代码	注释
N40 R1=30 R2=60 R3=10 R4=11 R5=50 R6=20	; 初值赋值。
N41 LA1: G0 X=R2*COS(R1)+R5 Y=R2*SIN(R1)+R6	; 已设置跳转标记 LA1。
N42 R1=R1+R3 R4=R4-1	
N43 IF R4>0 GOTOB LA1	; 如果满足跳转条件，向程序开始方向跳转到跳转标记 LA1。
N44 M30	; 程序结束

1.10.3 程序分支(CASE ... OF ... DEFAULT ...)

功能

CASE 功能可以检测一个变量或者一个计算函数当前值 (类型： INT），根据结果跳转到程序中的不同位置。

句法

CASE(<表达式>) OF <常量_1> GOTOF <跳转目标_1> <常量_2> GOTOF <跳转目标_2> ... DEFAULT GOTOF <跳转目标_n>

含义

CASE:	跳转指令
<表达式>:	变量或计算函数
OF:	用于编制有条件程序分支的关键字

<常量_1>: 变量或者计算函数首先规定的恒定值

类型： INT

<常量_2>: 变量或者计算函数第二个规定的恒定值

类型： INT

DEFAULT: 对于变量或者计算函数没有采用规定值的情况,可以用 DEFAULT 指令确定跳转目标。

提示：
如果 DEFAULT 指令没有被编程，在这些情况中紧跟在 CASE 指令之后程序段将成为跳转目标。

GOTOF: 以程序末尾方向的带跳转目标的跳转指令。

代替 GOTOF 可编程所有其他的 GOTO 指令（参见主题“程序跳转到跳转标记”）。

<跳转目标_1>: 当变量值或者计算函数值符合第一个规定的常量，程序分支到跳转目标。

可以如下规定跳转目标：

<跳转标记**>**: 跳转目标是程序中带有用户定义名称的跳转标记： **<**跳转标记>：

<程序段号码**>**: 主程序段号或者分支程序段号作为跳转目标（比如： 200, N300)

STRING 类型变量： 跳转目标变量。变量提供一个跳转标记或者一个程序段号。

<跳转目标_2>: 当变量值或者计算函数值符合第二个规定的常量，程序分支到跳转目标。

<跳转目标_n>: 当变量值不符合规定的常量，程序分支到跳转目标。

示例

程序代码

```
...
N20 DEF INT VAR1 VAR2 VAR3
N30 CASE(VAR1+VAR2-VAR3) OF 7 GOTOF Label_1 9 GOTOF Label_2 DEFAULT GOTOF Label_3
N40 Label_1: G0 X1 Y1
N50 Label_2: G0 X2 Y2
N60 Label_3: G0 X3 Y3
...
```

CASE 指令由 N30 定义下列程序分支可行性：

1. 如果计算函数值 VAR1+VAR2-VAR3 = 7，则跳转到带有跳转标记定义的程序段 "Label_1" (→ N40)。
2. 如果计算函数值 VAR1+VAR2-VAR3 = 9，则跳转到带有跳转标记定义的程序段 "Label_2" (→ N50)。
3. 如果计算函数 VAR1+VAR2-VAR3 的值既不等于 7 也不等于 9，则跳转到带有跳转标记定义的程序段“Label_3”(→ N60)。

1.11 程序部分重复 (REPEAT, REPEATB, ENDLABEL, P)

功能

程序部分重复是指在一个程序中，可以任意组合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.重复单个程序行：

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

2.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.重复两个跳转标记间的段落：

```
<起始跳转标记>: ...
...
<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.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：

- 需要重复的程序行（REPEATB）

  或者

- 需要重复的程序段落的开始（REPEAT）

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

<n>: 程序部分重复的次数

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 行之后的语句。

提示：

如果没有指定 P=<n>，则程序部分仅重复一次。

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 SATZ N10 五次
N50 REPEATB POSITION2	; 执行程序段 N20 一次

程序代码	注释
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	; 宽度
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	; 宽度
N20 Z=10-R10	
N30 G1 X=R10 F200	
N40 Y=R10	
N50 X=-R10	
N60 Y=-R10	
N70 END: Z=10	
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:	
N30 X10	
N40 Y10	
N50 BEGIN2:	
N60 X20	
N70 Y30	
N80 ENDLABEL: Z10	
N90 X0 Y0 Z0	
N100 Z-10	
N110 BEGIN3: X20	
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:	; 钻孔位置 1
N30 X1 Y1	
N40 X2	
N50 Y2	
N60 X3 Y3	
N70 ENDLABEL:	
N80 POS_2:	; 钻孔位置 2
N90 X10 Y5	
N100 X9 Y-5	
N110 X3 Y3	
N120 ENDLABEL:	
N130 BOHRER()	; 更换钻头和钻孔循环。

程序代码	注释
N140 GEWINDE(6)	; 换上螺纹钻 M6 和螺纹循环。
N150 REPEAT POS_1	; 重复程序部分一次，自 POS_1 到 ENDLABEL,
N160 BOHRER()	; 更换钻头和钻孔循环。
N170 GEWINDE(8)	; 换上螺纹钻 M8 和螺纹循环。
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。

1.12 控制结构

功能

控制系统按照编制好的标准顺序处理 NC 程序段。

该顺序可以通过编程可选的程序块和程序循环改变。 控制结构编程通过控制结构单元（关键字） IF...ELSE, LOOP, FOR, WHILE 和 REPEAT 实现。

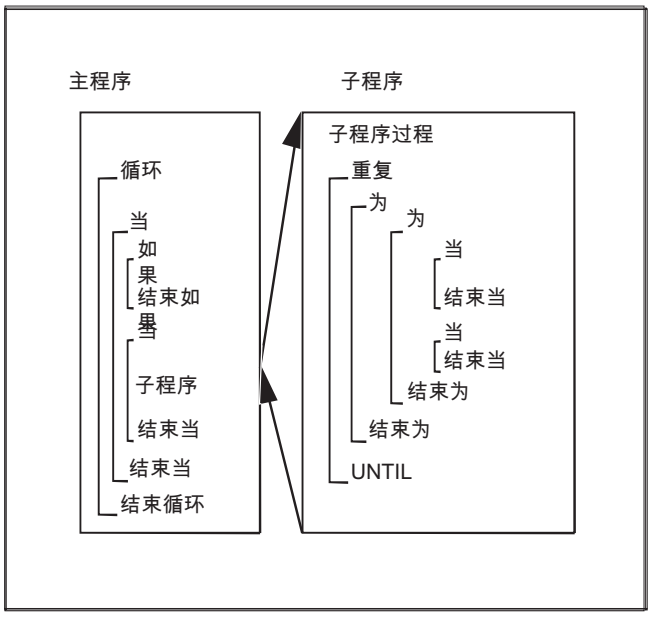
⚠小心
控制结构只有在一个程序的指令部分才可能。 程序头的定义不能有条件或重复执行。 标准控制结构的关键词和跳转目标一样不能和宏叠加。 宏定义时不能进行检测

有效性

控制结构对部分程序有效。

嵌套深度

在每个子程序之内,嵌套的层数可以达到 16 个标准控制结构。



操作时间的实现

在标准有效的翻译操作中,可以通过程序跳转的运用达到比标准控制结构快的程序操作。

在前面汇编的循环中,程序跳转和标准控制结构没有实际的区别。

边界条件

- 带有标准控制结构数组元的程序段不能被跳过。
- 跳转标记（标签）不允许在带控制结构单元的程序段中。
- 标准控制结构被翻译。 在识别一个循环结尾时,考虑到所找到的标准控制结构，会寻找循环开头。 之后在翻译过程中,模块结构不会完全被检测。
- 建议不要混合使用标准控制结构和程序跳转。
- 在循环的预处理中,会检查控制结构的正确嵌套。

1.12.1 带选项的程序循环 (IF, ELSE, ENDIF)

功能

当查询循环应包含一个可选的程序块时，可使用带 IF 和 ELSE 的结构: 如果满足 IF 条件，则执行 IF 内的程序块。 如果 IF 条件不满足，则执行 ELSE 内可选的程序块。

说明
如果不需要选择，则 IF 循环也可以不带 ELSE 指令，并对 ELSE 后的程序块编程。

句法

```
IF <条件>
...
ELSE
...
ENDIF
```

含义

IF:	导入 IF 循环。
ELSE:	导入可选的程序块。

ENDIF: 标记 IF 循环结束处并跳转到循环开头。

<条件>: 决定运行哪些程序块的条件。

示例

刀具更换子程序

程序代码	注释
PROC L6	; 刀具更换路线
N500 DEF INT TNR_AKTUELL	; 有效 T 号码变量
N510 DEF INT TNR_VORWAHL	; 预选 T 号码变量
	; 确定当前刀具
N520 STOPRE	
N530 IF $P_ISTEST	; 正在运行程序测试...
N540 TNR_AKTUELL = $P_TOOLNO	; ... 从程序文本中读取"当前"刀具。
N550 ELSE	; 否则...
N560 TNR_AKTUELL = $TC_MPP6[9998,1]	; ... 已读取主轴刀具。
N570 ENDIF	
N580 GETSELT(TNR_VORWAHL)	; 读取主轴上预选刀具的 T 号码。
N590 IF TNR_AKTUELL <> TNR_VORWAHL	; 如果预选刀具还不是当前刀具，则...
N600 G0 G40 G60 G90 SUPA X450 Y300 Z300 D0	; ... 回到刀具更换点...
N610 M206	; ... 并进行刀具更换。
N620 ENDIF	
N630 M17	

1.12.2 无限程序循环（LOOP，ENDLOOP）

功能

无限循环在无限程序中被应用。 在循环结尾总是跳转到循环开头重新进行。

句法

```
LOOP
...
ENDLOOP
```

含义

LOOP: 引入无限循环。

ENDLOOP: 标记循环结束处并跳转到循环开头。

示例

程序代码

```
...
LOOP
MSG（"无刀沿有效"）
M0
STOPRE
ENDLOOP
...
```

1.12.3 计数循环（FOR ... TO ...，ENDFOR）

功能

当一个带有一个确定值的操作程序被循环重复，计数循环就会被运行。

句法

```
FOR <变量> = <初值> TO <终值>
...
ENDFOR
```

含义

FOR: 引入计数循环。

ENDFOR: 一旦还没有得到计数终值，则标记循环结束处并跳转到循环开头。

<变量>: 计数变量从初值开始向上计数，直到终值且在每次运行时提高值“1”。

类型　INT 或 REAL

提示：

如果为计数循环编程了例如：R 参数， 则采用实数型变量。

如果计数变量为实数变量，则将四舍五入该变量值。

<初值>:　计数的初值

条件 初值必须小于终值。

<终值>:　计数的终值

示例

示例 1： 整数变量或 R 参数作为计数变量

整数变量作为计数变量：

程序代码	注释
DEF INT iVARIABLE1	
R10=R12-R20*R1 R11=6	
FOR iVARIABLE1 = R10 TO R11	; 计数变量 = 整数变量
R20=R21*R22+R33	
ENDFOR	
M30	

R 参数作为计数变量：

程序代码	注释
R11=6	
FOR R10=R12-R20*R1 TO R11	; 计数变量 = R 参数（实数变量）
R20=R21*R22+R33	
ENDFOR	
M30	

示例 2： 加工一个固定的零件数

程序代码	注释
DEF INT STUECKZAHL	; 用名称"STUECKZAHL"定义的 INT 型变量。
FOR STUECKZAHL = 0 TO 100	; 引入计数循环。 变量"STUECKZAHL"从初值"0"向上计数，直到终值"100"。

程序代码	注释
G01 …	
ENDFOR	; 计数循环结束
M30	

1.12.4 在循环开始处带有条件的程序循环（WHILE，ENDWHILE）

功能

WHILE 循环的开始是有条件的。 一旦满足条件，WHILE 循环即开始运行。

句法

```
WHILE <条件>
...
ENDWHILE
```

含义

WHILE:	引入程序循环。
ENDWHILE:	标记循环结束处并跳转到循环开头。
<条件>:	必须满足条件，只有这样 WHILE 循环才能运行。

示例

程序代码	注释
...	
WHILE $AA_IW[钻削轴] > -10	; 在下列条件下调用 WHILE 循环。 当前的钻削轴 WKS 额定值必须大于 -10。
G1 G91 F250 AX[钻削轴] = -1	
ENDWHILE	
...	

1.12.5 在循环结束处带有条件的程序循环（REPEAT，UNTIL）

功能

REPEAT 循环的结束是有条件的。 REPEAT 循环一旦被执行会不断重复,直到满足条件为止。

句法

```
REPEAT
...
UNTIL <条件>
```

含义

REPEAT:	引入程序循环。
UNTIL:	标记循环结束处并跳转到循环开头。
<条件>:	必须满足条件，只有这样 REPEAT 循环才能运行。

示例

程序代码	注释
...	
REPEAT	; 调用 REPEAT 循环。
...	
UNTIL ...	; 检查是否已满足条件。
...	

1.12.6 带层叠控制结构的程序示例

程序代码	注释
LOOP	
IF NOT $P_SEARCH	; 没有程序段搜索
G01 G90 X0 Z10 F1000	
WHILE $AA_IM[X] <= 100	

程序代码	注释
G1 G91 X10 F500	; 钻孔图
Z-F100	
Z5	
ENDWHILE	
Z10	
ELSE	
MSG（"在搜索过程中不钻孔"）	
ENDIF	
$A_OUT[1]=1	; 下一个钻孔板
G4 F2	
ENDLOOP	
M30	

1.13 程序协调（INIT，START，WAITM，WAITMC，WAITE，SETM，CLEARM）

功能

通道

一个通道可以独立地处理自己的程序，而与其它的通道无关。 因此那些它所赋值的轴和主轴可以通过程序控制。

在安装调试时，控制系统可以设定两个或多个通道。

程序协调

如果多个通道和一个工件的加工有关,那么就要求同步程序操作过程。

程序协调需要使用特殊的指令(命令)。 它们总是位于单独的程序段内。

说明

也可在自身通道中进行程序协调。

程序协调的指令

- 给出绝对路径

	在这里绝对路径根据以下规则构成
INIT (n,"/_HUGO_DIR/_N_名称_MPF") 或者	- 当前目录/_N_名称_MPF "当前目录" 表示所选择的工件目录或者默认目录 /_N_MPF_DIR.
INIT (n,"/_N_MPF_DIR/_N_名称_MPF")	- 选择某个程序以便在某个通道中执行： n: 通道的编号，数值视控制系统的配置情况而定 完整的程序名称 -

示例：	版本 SW3 以前
INIT(2,"/_N_WKS_DIR/_ABRICHT_MPF") G01F0.1 START	在一个**初始化**指令（没有同步动作）和一个 **NC 启动** 之间必须至少有一个可执行的程序段。 当调子用程序时，必须在指定路径时加上 "_SPF"
INIT (2,"/_N_WKS_DIR/_N_UNTER_1_SPF")	

- 给出相对路径

示例：	在给出相对路径时，子程序调用的规则同样有效。
INIT(2,"ABRICHT")	
INIT(3,"UNTER_1_SPF")	当调用子程序时，必须在程序名中加上 "_SPF" 。

参数

可以使用通道所共同具有的变量在程序之间交换数据（NCK 特有的全局变量）。 其它情况中每个通道的程序都是被分开建立的。

INIT（n，路径说明，确认模式）	用来在某个通道中执行的指令。 选择有绝对路径说明或者相对路径说明的特定程序。
START (n, n)	在其它的通道中启动所选的程序。 n,n：列举通道号：根据每个操作结构得出的值
WAITM（标记编号, n, n, ...）	在自身的通道里设定标记挨 以准停结束上述的程序段。 等待指定通道"n"中具有相同"标记编号"的标记（不必指定自有通道）。 标记会在同步之后被取消。 最多可以同时有 10 个通道被标记。
WAITMC（标记编号，n，n，	在自身的通道里设定标记挨 准停只有在其他通道没有达到标记时才会进行。 等待指定通道"n"中具有相同"标记编号"的标记（不必指定自有通道）。 一旦标记在给出的通道中达到挨
WAITE (n, n, ...)	等待给出的通道的程序结尾(自身通道不作说明)。 举例：对开始指令之后的停留时间进行编程。 N30 START(2) N31 G4 F0.01 N40 WAITE(2)
SETM（标记编号，标记编号，	在自有通道中给标记设定"标记编号，对正在执行的加工没有影响。 SETM（）跳过 RESET 和 NC-START 保存有效

	性。
CLEARM（标记编号，标记编号，	在自有通道中删除“标记编号”，对正在执行的加工没有影响。 通道中的所有标记都可以用 CLEARM（）取消。 CLEARM (0) 表示删除标记 "0". CLEARM（）跳过 RESET 和 NC-START 保存有效性。
n	相应的通道号或通道名

说明

以上所有的命令都必须在独立的程序段中。

标记的数量取决于装入的 CPU。

通道号

可以为待协调的通道规定最多 10 个通道作为通道号（整数值）。

通道名

通道名必须通过变量（见章节“变量和计算参数”）转换为编号，或者通道号的设立也可以通过 $MC_CHAN_NAME 定义的通道名（名称或者关键字）编程。 定义的名称必须符合 NC 语言规定（即前两个字符必须由字母或者由下划线组成）。

⚠小心

数字赋值应当在轻率的修改前被保存。

名称在 NC 中不能已含有不同的含义，如作为关键字、语言指令、轴名称等等。

SETM() 和 CLEARM()

SETM() 和 CLEARM() 也可从某个同步动作出发进行编程。 参见“设定/删除等候标记”一章： SETM CLEARM"

示例

名称为"MASCHINE"的通道应有通道编号 1，

名称为 "LADER" 的通道应有通道编号 2：

```
DEF INT MASCHINE=1, LADER=2
```

变量保存与通道相同的名称。

比如说指令显示 START：

```
START(MASCHINE)
```

编程协调举例

通道 1:

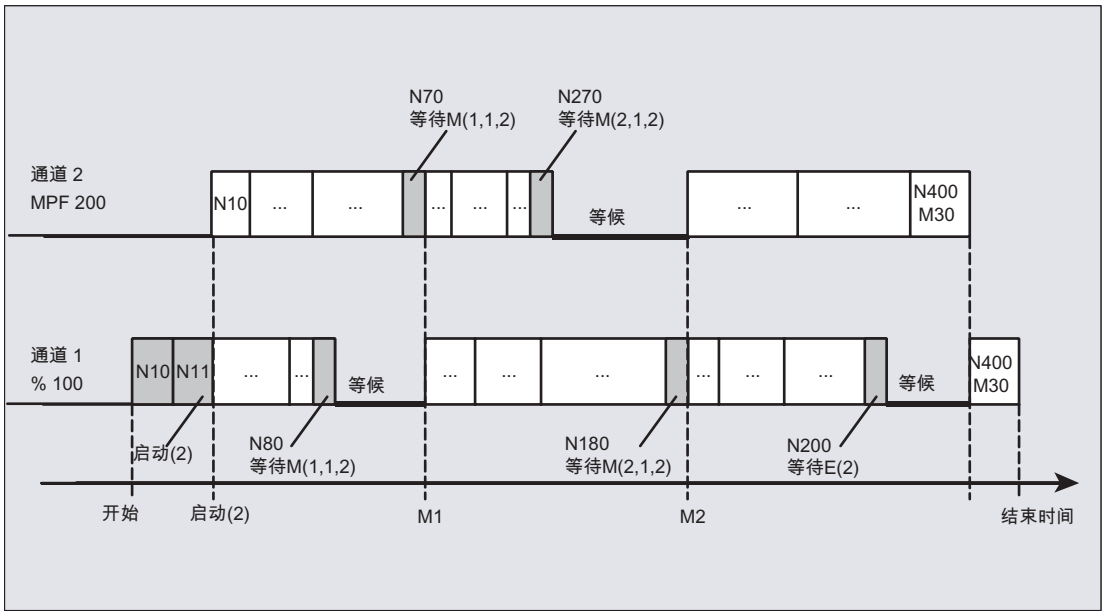
_N_MPF100_MPF

程序代码	注释
N10 INIT(2,"MPF200")	
N11 START(2)	; 在通道 2 中加工
...	
N80 WAITM(1,1,2)	; 在通道 1 和通道 2 中等待 WAIT 标记 1 在通道 1 中继续加工
...	
N180 WAITM(2,1,2)	; 在通道 2 和通道 2 中等待 WAIT 标记 1 在通道 1 中继续加工
...	
N200 WAITE(2)	; 等候通道 2 程序结束
N201 M30	; 通道 1 程序结束，全部结束
...	

通道 2：

_N_MPF200_MPF

程序代码	注释
;$PATH=/_N_MPF_DIR	
	; 在通道 2 中加工
N70 WAITM(1,1,2)	; 在通道 1 和通道 2 中等待 WAIT 标记 1，在通道 1 中继续加工
...	
N270 WAITM(2,1,2)	; 在通道 2 和通道 2 中等待 WAIT 标记 1，在通道 2 中继续加工
...	
N400 M30	; 通道 2 程序结束



示例： 工件中的程序

程序代码
N10 INIT(2,"/_N_WKS_DIR/_N_WELLE1_WPD/_N_ABSPAN1_MPF")

示例： 有相对路径说明的初始化指令

在通道 1 中选择程序/_N_MPF_DIR/_N_MAIN_MPF

程序代码	注释
N10 INIT(2,“MYPROG”)	; 程序/_N_MPF_DIR/_N_MYPROG_MPF，在通道 2 中选择。

示例： 带整数变量的通道名称和通道编号

$MC_CHAN_NAME[0]= "CHAN_X" ;第 1 通道的名称
$MC_CHAN_NAME[1]= "CHAN_Y" ;第 2 通道的名称

程序代码	注释
START(1, 2)	; 在第 1 和第 2 通道中执行启动

与带通道名称的编程类似：

程序代码	注释
START(CHAN_X, CHAN_Y)	; 在第 1 和第 2 通道中执行启动 ; 根据机床数据$MC_CHAN_NAME ，名称通道_X 和 通道_Y 在内部表示通道号 1 和 2 相应地在第 1 和第 2 通道中执行启动。

带整数变量的编程：

程序代码	注释
DEF INT chanNo1, chanNo2)	; 定义通道号
chanNo1=CHAN_X chanNo2=CHAN_Y	
START(chanNo1, chanNo2)	

1.14 中断程序（ASUP）

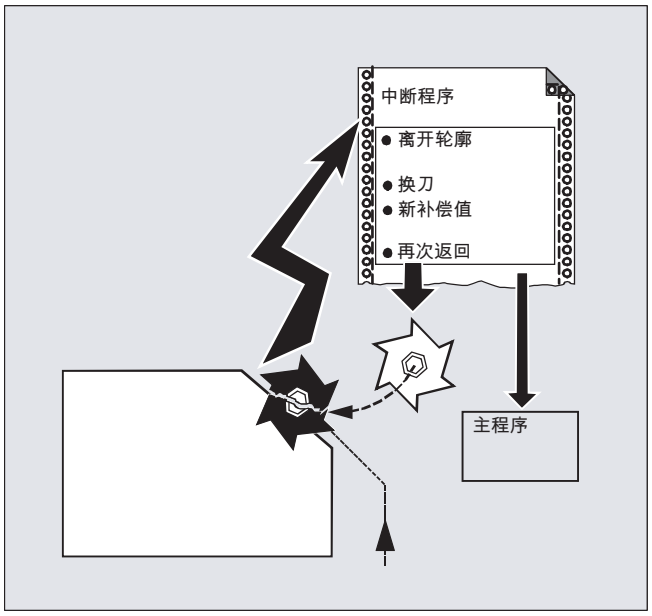
1.14.1 中断程序的功能

说明

在以下说明中交替出现的“异步子程序” (ASUP)和“中断程序”表示同一种功能。

功能

应该依据某个典型示例来阐述中断程序的功能：



在加工过程中工具折断。 由此触发一个信号，这个信号中止正在运行的处理过程并同时开始一个子程序，也就是那个所谓的中断程序。 在这个子程序中有所有在这种情况下应当被执行的指令。

如果子程序已执行完毕（并且因此而恢复运行就绪状态），控制系统就会跳回到主程序中，并且 － 根据 REPOS 指令－ 在中断点继续执行加工。（见“返回轮廓 (页 491)”）。

⚠小心

如果在子程序中没有编程任何 REPOS 指令，则向着程序段的结束点定位，该结束点跟随中断的程序段。

文献

功能手册 基本功能；BAG，通道，程序运行，复位特性（K1），章节：“异步子程序 (ASUP)、中断程序”

1.14.2 建立中断程序

建立作为子程序的中断程序

这个中断程序在定义时和一个子程序一样被标识。

示例：

程序代码	注释
PROC ABHEB_Z	; 程序名称“ABHEB_Z”
N10 ...	; 紧接着的是 NC 程序段。
...	
N50 M17	; 最后结束程序并返回到主程序

保存模态 G 功能（SAVE）

进行定义时，可使用 SAVE 来标识中断程序。

属性 SAVE 发挥下列作用：在调用中断程序之前保存有效的模态 G 功能，并在结束中断程序之后再次激活（见“带 SAVE 机制（SAVE）的子程序 (页 168)”）。

由此，可以在结束中断程序之后，在中断点继续进行加工。

示例：

程序代码
PROC ABHEB_Z SAVE
N10 ...
...
N50 M17

赋值其它中断程序（SETINT）

可以在中断程序内部编程SETINT指令（见 “赋值并启动中断程序（SETINT）” (页 121)），并由此立即接通其它的中断程序。 只有通过输入端才可以触发。

文献

有关建立子程序的其它信息可参阅“子程序技术，宏技术”一章。

1.14.3 中断程序赋值和启动(SETINT, PRIO, BLSYNC)

功能

控制系统使用信号（输入端 1...8），它能引起正在进行的程序的中断和启动相应的中断程序。

在零件程序中用指令 SETINT 分配，哪些程序启动哪些输入端。

如果在零件程序中有多个 SETINT 指令，并由此能够同时出现多个信号，则必须为那些赋值的中断程序分配优先级值，它用于确定加工时的顺序： PRIO=<值>

如果在中断处理期间有新的信号输入,有较高优先级的程序中断当时的中断程序。

句法

```
SETINT(<n>) PRIO=<值> <名称>
SETINT(<n>) PRIO=<值> <名称> BLSYNC
SETINT(<n>) PRIO=<值> <名称> LIFTFAST
```

含义

SETINT(<n>):	指令： 赋值中断程序输入端 **<n>**。 当接通输入端 **<n>**时，启动赋值的中断程序。	
	提示： 如果一个确定的输入端被一个新的程序赋值，旧的值自动失效。	
<n>:	参数： 输入端编号	
	类型：	INT
	取值范围：	1 ... 8
PRIO= ：	指令： 确定优先级	
<值>：	优先级值	
	类型：	INT
	取值范围：	1 ... 128
		优先级 1 相当于最高优先级。

<名称>:	需要处理的子程序（中断程序）名称。
BLSYNC:	如果共同编程了 SETINT 指令和 BLSYNC，在中断信号出现时仍会继续处理运行中的程序段，然后才启动中断程序。
LIFTFAST:	如果共同编程了SETINT指令和LIFTFAST，在中断信号出现时会首先使得“刀具快速离开工件轮廓”（参见 快速离开工件轮廓（SETINT LIFTFAST，ALF） (页 125)），然后才启动中断程序。

示例

示例 1： 赋值中断程序和确定优先级

程序代码	注释
...	
N20 SETINT(3) PRIO=1 ABHEB_Z	; 如果接通了输入端 3，则应该启动中断程序“ABHEB_Z”。
N30 SETINT(2) PRIO=2 ABHEB_X	; 如果接通了输入端 2，则应该启动中断程序“ABHEB_X”。
...	

如果多个输入端同时保留，则中断程序会根据级别数的顺序进行处理。 首先是“ABHEB_Z”，然后是“ABHEB_X”。

示例 2： 重新赋值中断程序

程序代码	注释
...	
N20 SETINT(3) PRIO=2 ABHEB_Z	; 如果接通了输入端 3，则应该启动中断程序“ABHEB_Z”。
…	
N120 SETINT(3) PRIO=1 ABHEB_X	; 给一个新的中断程序赋值输入端 3： 当接通输入端 3 时，应该启动“ABHEB_X”而不是“ABHEB_Z”。

1.14.4 取消/再激活一个中断程序的赋值（DISABLE，ENABLE）

功能

SETINT 指令可以通过 DISABLE 取消，并通过 ENABLE 再次激活，不会丢失输入端 → 中断程序的赋值。

句法

```
DISABLE(<n>)
ENABLE(<n>)
```

含义

DISABLE(<n>): 指令： **取消** 中断程序输入端的赋值 **<n>**

ENABLE(<n>): 指令： **再次激活** 中断程序输入端的赋值 **<n>**

<n>: 参数：输入端编号

类型： INT

取值范围： 1 ... 8

示例

程序代码	注释
...	
N20 SETINT(3) PRIO=1 ABHEB_Z	; 如果接通了输入端 3，则应该启动中断程序"ABHEB_Z"。
...	
N90 DISABLE(3)	; 取消 N20 中的 SETINT 指令。
...	
N130 ENABLE(3)	; 再次激活 N20 中的 SETINT 指令。
...	

1.14.5 删除中断程序的赋值（CLRINT）

功能

用 SETINT 定义的输入端 → 中断程序赋值可以用 CLRINT 删除。

句法

```
CLRINT(<n>)
```

含义

CLRINT(<n>):	指令：	删除中断程序输入端赋值 **<n>**
<n>:	参数：	输入端编号
	类型：	INT
	取值范围：	1 ... 8

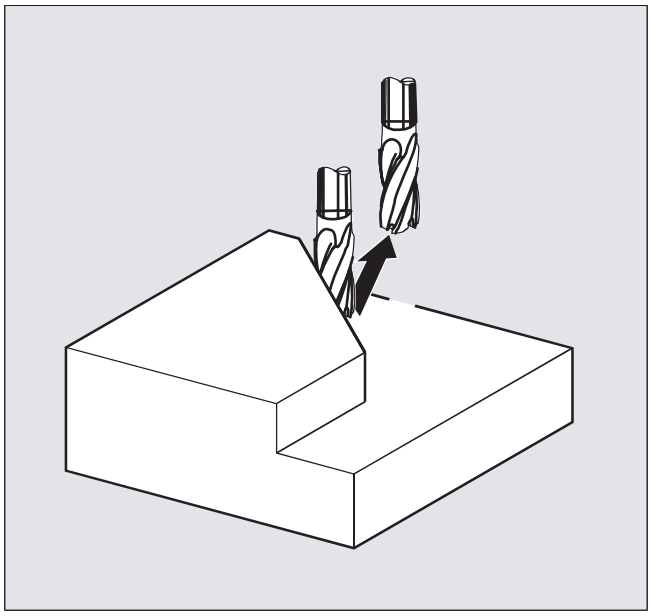
示例

程序代码	注释
...	
N20 SETINT(3) PRIO=2 ABHEB_Z	;
...	
N50 CLRINT(3)	; 输入端"3"和程序"ABHEB_Z"之间的赋值被删除。
...	

1.14.6 快速离开工件轮廓（SETINT LIFTFAST，ALF）

功能

如果 SETINT 指令带 LIFTFAST，在接通输入端时，通过快速从工件轮廓离开的方式使刀具离开。



其它的过程与 LIFTFAST 旁的 SETINT 指令是否包含一个中断程序有关：

带中断程序：	在快速离开**之后**执行中断程序。
不带中断程序：	在快速离开之后加工停止并发出报警。

句法

```
SETINT(<n>) PRIO=1 LIFTFAST
SETINT(<n>) PRIO=1 <名称> LIFTFAST
```

含义

SETINT(<n>):	指令： 赋值中断程序输入端 **<n>**。 当接通输入端 **<n>**时，启动赋值的中断程序。
<n>:	参数： 输入端编号
	类型： INT

	取值范围： 1 ... 8
PRIO= ：	确定优先级
<值>：	优先级值
	取值范围： 1 ... 128
	优先级 1 相当于最高优先级。
<名称>:	需要处理的子程序（中断程序）名称。
LIFTFAST:	指令： 快速离开工件轮廓
ALF=… :	指令： 可编程的运动方向（在运动程序段中）
	有关用 ALF编程的方法，见主题“快速离开工件轮廓时的运行方向 (页 127)”。

边界条件

带镜像的有效框架的性能

在确定离开方向时会检测,是否有一个框架带镜像被激活。 在这种情况下，刀具沿正切线离开，左右相间。 在刀具方向的方向分量没有镜像。 通过 MD 设置激活该性能：

MD21202 $MC_LIFTFAST_WITH_MIRROR = TRUE

示例

折断的刀具自动地被另一个刀具替代。 加工以新的刀具继续进行。

主程序：

主程序	注释
N10 SETINT(1) PRIO=1 W_WECHS LIFTFAST	; 如果接通输入端 1 ，刀具会立刻以快速离开（代码 7 对应工具半径补偿 G41）的方式离开工件轮廓。 然后中断程序“W_WECHS”被执行。
N20 G0 Z100 G17 T1 ALF=7 D1	
N30 G0 X-5 Y-22 Z2 M3 S300	
N40 Z-7	
N50 G41 G1 X16 Y16 F200	
N60 Y35	
N70 X53 Y65	
N90 X71.5 Y16	
N100 X16	
N110 G40 G0 Z100 M30	

子程序：

子程序	注释
PROC W_WECHS SAVE	; 带当前运行状态储存的子程序
N10 G0 Z100 M5	; 换刀位置，主轴停止
N20 T11 M6 D1 G41	; 更换刀具
N30 REPOSL RMB M3	; 返回轮廓并跳转到主程序中（在一个程序段中编程）

1.14.7 快速离开工件轮廓时的运行方向

后退运行

退回运动的平面由下列 G 代码确定：

- LFTXT

  由轨迹切线和刀具方向来确定退回运动的平面（标准设置）。

- LFWP

  退回运动的平面是用 G 代码 G17，G18 或 G19 选择的、已激活的工作平面。 撤回运动的方向不由轨道切线决定。 由此可以编程一个与轴并行的快速离开。

- LFPOS

  使通过 POLFMASK / POLFMLIN 指明的轴回到用 POLF 编程的绝对轴位置。

  ALF 在多个轴以及多个线性相关轴上时对退刀方向没有影响。

  文献:
  编程手册 基本原理；章节：“螺纹切削快速回程”

可编程的运行方向（ALF=...）

在退回平面中，用 ALF 以 45 度的不连续步骤对方向进行编程。

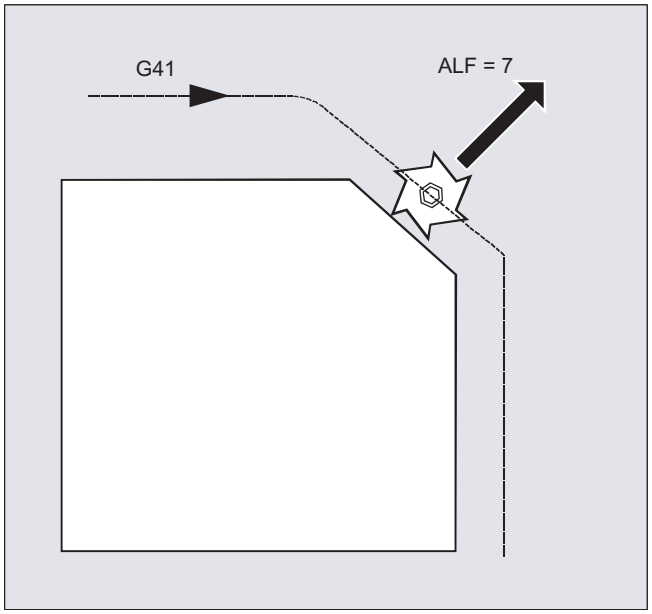
可能的运行方向存储在控制系统中，带专门的代码号，并可以在这个代码下调用。

示例：

程序代码
N10 SETINT(2) PRIO=1 ABHEB_Z LIFTFAST

程序代码
ALF=7

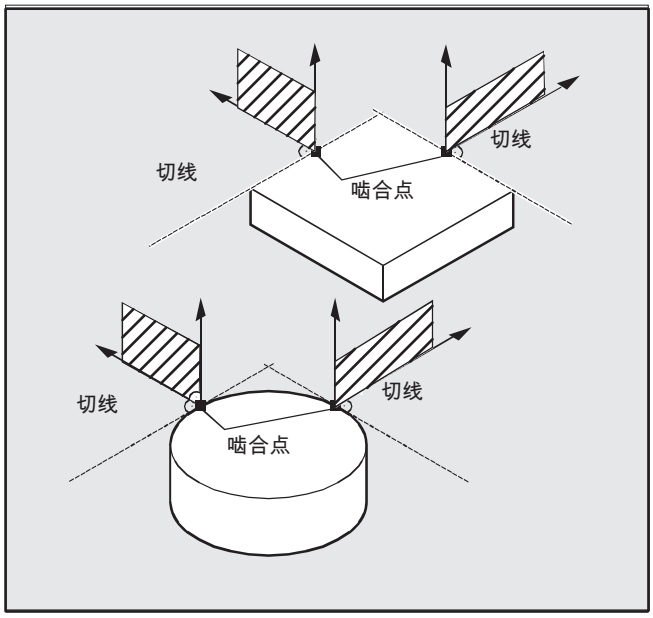
刀具在启用了 G41 的情况下（从轮廓左侧加工方向）垂直从轮廓上离开。



LFTXT 下用于描述运行方向的基准面

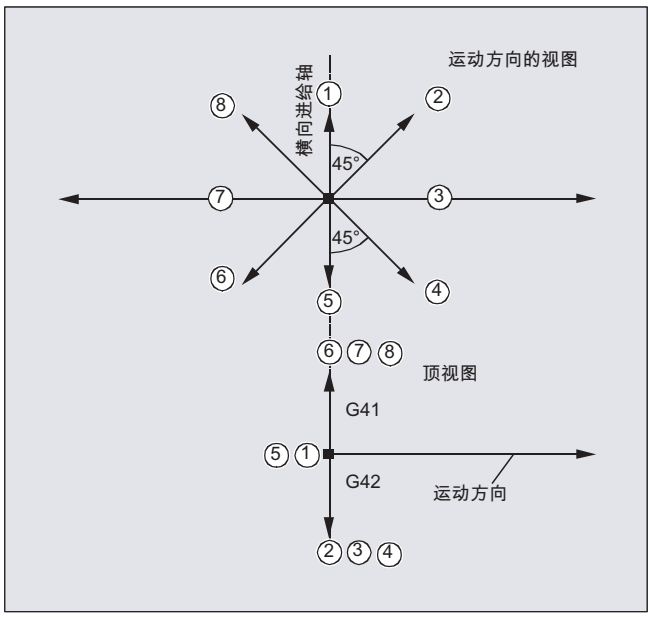
工具在编程的轮廓上的切入点有一个平面,它作为带相应代码离开运动的参数说明的基准面。

这个基准面由工具径向轴(进刀方向)和一个矢量组成,这个矢量与这个平面相对并与工具在轮廓上的切入点的切线垂直。



LFTXT 下带运行方向的代码编号

从这个基准面出发您可以在下面的插图里找到带运行方向的代码编号。



对于 ALF=1，后退在刀具方向中确定。

用 ALF=0 取消“快速离开”功能。

⚠小心
对于已接通的刀具半径补偿，应该： • 对于 G41，编码 2，3，4 • 对于 G42，编码 6，7，8 **不会** 被使用，因为在这些情况下刀具驶向轮廓并会与工件相撞。

LFWP 下带运行方向的代码编号

对于 LFWP，工作平面中的方向被分配如下：

- G17: X/Y 平面

  ALF=1： 在 X 方向后退

  ALF=3： 在 Y 方向后退

- G18: Z/X 平面

  ALF=1： 在 Z 方向后退

  ALF=3： 在 X 方向后退

- G19: Y/Z 平面

  ALF=1： 在 Y 方向后退

  ALF=3： 在 Z 方向后退

1.14.8 中断程序下的运动过程

没有 LIFTFAST 的中断程序

轴在轨迹上运动，直至在停止状态中停止。 接着启动中断程序。

停止状态位置被保存为中断位置，并且在 REPOS 下，用 RMI 在中断程序结束时向该位置逼近。

带 LIFTFAST 的中断程序

轴运动在轨迹上停止。 同时，LIFTFAST 运动作为叠加运动被执行。 如果轨迹运动和 LIFTFAST 运动停止，则启动中断程序。

轮廓上的位置作为中断位置被保存, 在这个位置上开始 LIFTFAST 运动并由此离开轨迹。

带有 LIFTFAST 和 ALF=0 的中断程序与没有 LIFTFAST 的中断程序有一样的特性。

说明

几何轴快速离开工件轮廓时所移动的距离，可以通过机床数据设定。

1.15 交换轴，交换主轴 (RELEASE, GET, GETD)

功能

一个或多个轴和主轴总是仅可以在一个通道中被插补。 如果某个轴必须在两个不同的通道中以交替方式工作（例如托盘更换器），则必须首先在当前通道中将其释放，然后将其接收到另一个通道中. 轴会在两个通道之间进行转换。

轴交换扩展

一个进给轴/主轴可以通过预处理停止和同步动作在预运行和主运行之间切换，或者也可以不通过预处理停止进行切换。 此外，也可以通过下列方式进行轴交换

- 轴容器旋转 AXCTSWE 或者 AXCTWED 借助包含的 GET/GETD。
- 如果该轴在此与其它轴连接在一起时请旋转框架。
- 同步动作，参见运动同步动作“轴交换 RELEASE，GET”。

机床制造商

请注意机床制造商说明。 通过可编程的机床数据，必须对某一轴在所有通道的交换进行明确的定义，且也可通过可修改的机床数据设定轴交换属性。

句法

RELEASE(轴名称，轴名称， ...) 或者 RELEASE(S1)

GET （轴名称，轴名称， ...） 或者 GET （S2）

GETD（轴名称，轴名称 ...）或者 GETD（S3）

用 GETD(GET Directly)将一个轴从另一个通道中直接取出。 这就是说，在另一个通道中不必给该 GETD 编程适当的 RELEASE。 不过这也意味着：现在必须建立另一个通道通讯（例如等待标记）。

含义

RELEASE （轴名称， 轴名称， …）:	轴使能（n）
GET （轴名称， 轴名称， …）:	轴接收（n）
GETD （轴名称， 轴名称， …）:	轴直接接收（n）
轴名称:	系统中的轴赋值： AX1, AX2, … 或者给出加工轴名称

RELEASE(S1)：	主轴 S1,S2,...释放
GET(S2)：	主轴 S1,S2,...接收
GETD(S3)：	主轴 S1,S2,直接接收...

无预处理停止的 GET 指令

如果在一个 **无**预处理停止的 GET 指令后，轴通过 RELEASE(轴) 或 WAITP(轴) 再次被使能，则接下来的 GET 会变为**带**预处理停止 GET。

⚠小心

即使在按键复位或者程序复位之后，某个使用 GET 接收的轴和主轴也会保持分配给该通道。

当重新启动程序时，如果在基本通道中需要轴的话，就必须以程序控制方式类分配所交换的轴或者主轴。

在 POWER ON（上电）后，它将给在机床数据中保存的通道赋值。

示例

示例 1： 在两个通道之间进行轴交换

6 个轴在通道 1 中用于加工的为： 1., 2., 3. 和第 4 个轴。
第 5 和第 6 个轴在通道 2 中被用来更换工件。

轴 2 应当在两个通道之间可以进行交换并在 POWER ON 之后给通道 1 赋值。

通道 1 中的程序“MAIN”：

程序代码	注释
INIT (2,"交换 2")	; 在通道 2 中选择程序交换 2
N… START (2)	; 在通道 2 中启动程序。
N… GET (AX2)	; 接受轴 AX2
...	
N… RELEASE (AX2)	; 释放轴 AX2。
N… WAITM (1,1,2)	; 等待通道 1 和通道 2 中的 Wait 标记，以便在两个通道中进行同步。
...	; 交换轴之后的其它流程。
N…M30	

通道 2 中的程序“交换 2”：

编程	注释
N… RELEASE(AX2)	
N160 WAITM (1,1,2)	; 等待通道 1 和通道 2 中的 Wait 标记，以便在两个通道中进行同步。
N150 GET (AX2)	; 接受轴 AX2
...	; 交换轴之后的其它流程。
N…M30	

示例 2： 没有同步的轴交换

如果不必对轴进行同步，则通过 GET 不会产生预处理停止。

编程	注释
N01 G0 X0	
N02 RELEASE(AX5)	
N03 G64 X10	
N04 X20	
N05 GET(AX5)	; 当不需要同步时，这就不会成为可执行的程序段。
N06 G01 F5000	; 不是可执行的程序段。
N07 X20	; 不是可执行的程序段，因为 X 位置与 N04 中的一样。
N08 X30	; 在 N05 之后第一个可执行的程序段。
...	

示例 3： 激活无预处理停止的轴交换

前提条件： 无预处理停止的轴交换必须通过机床数据设计。

编程	注释
N010 M4 S100	
N011 G4 F2	
N020 M5	
N021 SPOS=0	
N022 POS[B]=1	
N023 WAITP[B]	; 轴 B 变成中性轴。
N030 X1 F10	
N031 X100 F500	
N032 X200	
N040 M3 S500	; 轴不触发预处理停止/REORG。
N041 G4 F2	

编程	注释
N050 M5	
N099 M30	

如果主轴或者轴 B 直接按照程序段 N023 作为 **PLC 轴**，例如运行到 180 度且返回到 1 度,然后该轴重新变为中性轴而在程序段 N40 中不释放预处理停止。

前提条件

轴交换的前提

- 轴必须已经通过机床数据在所有要使用该轴的通道中定义好。
- 必须通过 **achs** 特定的机床数据确定在 POWER ON 之后将轴分配给哪个通道。

说明

释放轴：RELEASE

在轴使能时必须要注意:

1. 轴不可以参加转换。
2. 在轴耦合时(正切控制),所有相关轴都必须使能。
3. 一个参与的定位轴在这种状态下不能交换。
4. 在龙门架主轴机床中，所有跟随轴也被交换。
5. 在轴耦合时(联动,引导轴耦合,电子齿轮)只有相连的引导轴被使能。

接受轴：GET

用这个命令执行原来的轴交换。完全由已在其中编程了该指令的通道来负责轴。

GET 的作用

带同步的轴变换:

当某个轴临时处在另外一个通道中或者分配给了 PLC、且在 GET 之前没有通过 "WAITP", G74 或者删除剩余行程的方式进行同步时，才必须对该轴进行同步。

- 进给停止（与 STOPRE 相同）。
- 在交换完全执行之前，加工始终保持中断状态。

自动的"GET"

如果一个轴在通道中原则上可用,但是当时实际上不是作为挨如果这个（些）轴已经被同步，就不会产生进给停止。

设置可修改的轴交换属性。

轴的交换时刻可通过机床数据如下设置：

- 如果轴通过 WAITP 处于一个中性状态(与前面的性能一样),那么也可以在两个通道之间进行自动的轴变换。
- 当某个轴容器旋转请求可由执行的通道分配的轴容器所有轴通过隐式 GET 和 GETD 指令取出放入通道中。随后的轴交换仅允许在结束轴容器旋转后进行。
- 在主程序中插入一个临时程序段之后，检查是否已成功进行了重新编组。只有当该程序段的轴状态与当前的轴状态不 一致时，才有必要进行重新编组。
- 也可以在不停止进给的情况下进行轴交换，而无需带进给停止和进给与主程序同步的 GET 程序段。 然后只生成带 GET 指令的临时程序段。在主程序中处理该程序段时，检查程序段中的轴状态是否与当前轴状态一致。

轴或主轴交换功能的其它信息参见
/FB2/ 功能手册扩展功能；BAGs、通道、轴交换（K5）。

1.16 将轴移交到另一个通道中（AXTOCHAN）

功能

用语言指令 AXTOCHAN 可以把轴指定给一个特定通道，以此把轴移到另一个通道。 该轴可以从 NC 零件程序以及同步动作中移到相应的通道。

句法

AXTOCHAN（轴名称，通道名称，[，轴名称，通道名称[，...]]）

含义

AXTOCHAN:	指定轴为某一特定通道
轴名称:	系统中的轴赋值： X，Y，... 或者参与的机床轴名称的数据。 待执行的通道不必是其自身通道，也不必是当前具有该轴插补权的通道。
通道编号：	要给轴分配的通道号

说明

参与的定位轴和仅由 PLC 控制的轴

一个作为参与定位轴的 PLC 轴不能更换通道。 一个仅由 PLC 控制的轴不能分配给 NC 程序。

文献

功能手册 扩展功能；定位轴（P2）

示例

NC 程序中的 AXTOCHAN

轴 X 和 Y 在通道 1 和 2 中已知。 当前通道 1 具有插补权且将在通道 1 中启动下列程序：

程序代码	注释
N110 AXTOCHAN(Y,2)	; Y 轴移向通道 2。
N111 M0	
N120 AXTOCHAN(Y,1)	; 重新取回 Y 轴（中性）

程序代码	注释
N121 M0	
N130 AXTOCHAN(Y,2,X,2)	; Y 轴和 X 轴移到通道 2（轴中性）。
N131 M0	
N140 AXTOCHAN(Y,2)	; Y 轴移向通道 2（NC 程序）
N141 M0	

其它信息

NC 程序中的 AXTOCHAN

对于在自身通道中的 NC 程序，仅当轴请求时，执行 GET，并由此等待真正的状态改变。如果轴被要求用于另一个通道或者要变成自身通道中的中性轴时，取消相应指令。

同步动作的 AXTOCHAN

如果要求轴用于自身通道时，则将来自同步动作的 AXTOCHAN 映像到同步动作的 GET。在这种情况下，轴在首个用于自身通道的请求时成为中性轴。 第二个请求时，把轴分配给 NC 程序，与 NC 程序中的 GET 指令类似。 关于同步动作的 GET 指令参见章节“运动同步动作”。

1.17 有效设置机床数据（NEWCONF）

功能

使用指令 NEWCONF 可以使得所有生效级为“NEW_CONFIG”的机床数据生效。 也可在操作界面 HMI 中通过按下软键“激活机床数据”的方式来激活该功能。

当执行功能 NEWCONF 时，会出现隐式预处理停止，即轨迹运动会被中断。

句法

```
NEWCONF
```

含义

NEWCONF: 激活所有生效级为“NEW_CONFIG”的机床数据的指令

跨通道执行零件程序中的 NEWCONF

如果改变了零件程序的轴机床数据，并随即用 NEWCONF 激活，则 NEWCONF 仅激活会导致零件程序通道改变的机床数据。

说明

为了确保所有的更改有效，必须在每个通道中执行 NEWCONF 指令，在这些通道中和机床数据更改相关的轴或者功能也被即时计算。

对于 NEWCONF 无轴向机床数据设置为有效。

由 PLC 控制的轴必须执行轴复位。

示例

铣削加工: 用不同的工艺加工钻孔的位置。

程序代码	注释
N10 $MA_CONTOUR_TOL[AX]=1.0	; 更改机床数据。
N20 NEWCONF	; 激活机床数据。
...	

1.18 写入文件（WRITE）

功能

使用 WRITE 指令可以将零件程序中的段落或数据写入到指定文件（日志文件）的末尾，或写入到正在执行的零件程序中。 程序段/数据在文件末尾插入，也就是说在 M30 之后。

说明

如果需要使用 WRITE 指令的文件不存在 NC 中，应首先新建该文件。

然后将该文件存放在静态 NC 存储器中。 在 SINUMERIK 840D s 上，存储器为 CF 卡。因此它和 SINUMERIK 840D 相比，WRITE 指令的运行时间延长了 75 ms。

如果硬盘中有一个相同名称的文件,则文件关闭后(在 NC 中)被覆盖。 解决方法： 进入操作区“通讯”，按下软键“属性”，修改 NC 中的名称。

前提条件

当前所设置的保护级别必须等于或者大于文件的 WRITE 权限。 否则系统会拒绝访问并且显示出错提示（出错变量的返回值=13）。

句法

```
DEF INT <错误>
WRITE(<错误>,"<文件名称>","<程序段/数据>")
```

含义

WRITE:	将一个程序段/数据插入到指定文件末尾的指令		
<错误>:	返回错误值的变量		
	类型：	INT	
	值：	0	无错误
		1	非法路径
		2	路径未找到
		3	文件未找到
		4	错误的文件类型
		10	文件已满
		11	文件正被使用

12 没有空余的存储量

13 无访问权限

20 其它错误

<文件名称>: 插入指定的程序段/数据的文件名称

类型： STRING

在指定文件名称时应注意以下几点：

- 如果指定的文件名称含有空格或者控制符（ASCII 代码<= 32 的字符），就会中断 WRITE 指令并且显示出错标识 1“路径非法”。
- 文件名可以通过路径和文件标识指定。
  - 路径说明

    路径必须是绝对的，即以“/”开始。

    如果没有指定路径，文件会存放在当前的目录（=选中程序的目录）中。
  - 文件标识

    如果文件名不包含文件主标识(_N_),系统会自动补充。

    如果文件名中倒数第四个字符是一个下划线“_”，则后面的三个字符被视为文件标识。 只允许使用文件标识 _SPF 和_MPF，从而可以在执行所有的文件指令时使用相同的文件名称，如通过 STRING 类型的变量。

    如果没有指定标记(“_MPF”或“_SPF”)，系统会自动补充 _MPF。
- 文件名的长度最多可以有 32 个字节,路径的长度最多可以有 128 个字节。

示例：

```
"PROTFILE"
"_N_PROTFILE"
"_N_PROTFILE_MPF"
"/_N_MPF_DIR_/_N_PROTFILE_MPF/"
```

<程序段/数据>: 插入指定文件的程序段/数据。

类型： STRING

提示：

内部还会附加上 LF，就是说字符串会增加 1 个字符的长度。

边界条件

- **最大文件大小(→ 机床制造商!)**

  允许的最大日志文件大小在以下机床数据中定义：

  MD11420 $MN_LEN_PROTOCOL_FILE

  最大文件大小的限制适用于所有通过 WRITE 指令创建的文件。 一旦超出该限制，系统就会输出出错信息，并不再保存程序段或数据。 如果存储器够用,则可以编制一个新的文件。

示例

示例 1： WRITE 指令，无绝对路径

程序代码	注释
N10 DEF INT ERROR	; 出错变量的定义。
N20 WRITE(ERROR,"TEST1","PROTOKOLL VOM 7.2.97")	; 将文本 PROTOKOLL VOM 7.2.97 写入到文件 _N_TEST1_MPF.中
N30 IF ERROR	; 出错分析。
N40 MSG ("执行 WRITE 指令时出错:"<<ERROR)	
N50 M0	
N60 ENDIF	
...	

示例 2： WRITE 指令，带绝对路径

程序代码
...
WRITE(ERROR,"/_N_WKS_DIR/_N_PROT_WPD/_N_PROT_MPF","PROTOKOLL VOM 7.2.97")
...

1.19 删除文件（DELETE）

功能

用 DELETE 指令可以删除所有的文件，无论它是否通过 WRITE 指令产生。 通过更高存取级别产生的文件也可以用 DELETE 删除。

句法

```
DEF INT <错误>
DELETE(<错误>,"<文件名称>")
```

含义

DELETE:	删除指定文件的指令		
<错误>:	返回错误值的变量		
	类型：	INT	
	值：	0	无错误
		1	非法路径
		2	路径未找到
		3	文件未找到
		4	错误的文件类型
		11	文件正被使用
		12	没有空余的存储量
		20	其它错误
<文件名称>:	需要删除的文件的名称		
	类型：	STRING	

在指定文件名称时应注意以下几点：

- 如果指定的文件名称含有空格或者控制符（ASCII 代码<= 32 的字符），就会中断 DELETE 指令并且显示出错标识 1“路径非法”。
- 文件名可以通过路径和文件标识指定。
  - 路径说明

    路径必须是绝对的，即以“/”开始。

    如果没有指定路径，会在当前的目录（=选中程序的目录）中查找文件。
  - 文件标识

    如果文件名不包含文件主标识(_N_),系统会自动补充。

    如果文件名中倒数第四个字符是一个下划线“_”，则后面的三个字符被视为文件标识。 只允许使用文件标识 _SPF 和_MPF，从而可以在执行所有的文件指令时使用相同的文件名称，如通过 STRING 类型的变量。

    如果没有指定标记(“_MPF”或“_SPF”)，系统会自动补充 _MPF。
- 文件名的长度最多可以有 32 个字节,路径的长度最多可以有 128 个字节。

示例：

```
"PROTFILE"
"_N_PROTFILE"
"_N_PROTFILE_MPF"
"/_N_MPF_DIR_/_N_PROTFILE_MPF/"
```

示例

程序代码	注释
N10 DEF INT ERROR	; 出错变量的定义。
N15 STOPRE	; 预处理停止。
N20 DELETE(ERROR,"/_N_SPF_DIR/_N_TEST1_SPF")	; 删除子程序目录中的文件 TEST1。
N30 IF ERROR	; 出错分析。
N40 MSG（“执行 DELETE 指令时出错：” <<ERROR)	
N50 M0	
N60 ENDIF	

1.20 读取文件中的行（READ）

功能

READ 指令用来在指定文件中读取一个或者多个行，并且将所读取的信息保存在一个 STRING 型数组中。 每个读入的文件行都占用数组中的一个数组元素。

说明

文件必须位于 NCK 的静态用户存储器中 (被动文件系统)。

前提条件

当前所设置的保护级别必须等于或者大于文件的 READ 权限。 否则系统会拒绝访问并且显示出错提示（出错变量的返回值=13）。

句法

```
DEF INT <错误>
DEF  STRING[<字符串长度>] <结果>[<n>,<m>]
READ(<错误>,"<文件名称>",<起始行>,<行数>,<结果>)
```

含义

READ:	指令，用于读取指定文件中的某些数据行并将它保存到变量数组中。		
<错误>:	返回错误值的变量（Call-By-Reference 引用调用参数）		
	类型：	INT	
	值：	0	无错误
		1	非法路径
		2	路径未找到
		3	文件未找到
		4	错误的文件类型
		13	访问权限不够
		21	行不存在（参数 <起始行>或<行数>大于指定文件中的总行数）。

22 结果变量（<结果>）的数组长度太短。

23 行范围太大（选择的参数<行数>太大，已超出了文件末尾）。

<文件名称>: 待读取的文件的名称（Call-By-Value 值调用参数）

类型： STRING

在指定文件名称时应注意以下几点：

- 如果指定的文件名称含有空格或者控制符（ASCII 代码<= 32 的字符），就会中断 READ 指令并且显示出错标识 1“路径非法”。
- 文件名可以通过路径和文件标识指定。
  - 路径说明

    路径必须是绝对的，即以“/”开始。

    如果没有指定路径，会在当前的目录（=选中程序的目录）中查找文件。
  - 文件标识

    如果文件名不包含文件主标识(_N_),系统会自动补充。

    如果文件名中倒数第四个字符是一个下划线“_”，则后面的三个字符被视为文件标识。 只允许使用文件标识 _SPF 和 _MPF，从而可以在执行所有的文件指令时使用相同的文件名称，如通过 STRING 类型的变量。

    如果没有指定标记(“_MPF”或“_SPF”)，系统会自动补充 _MPF。
- 文件名的长度最多可以有 32 个字节,路径的长度最多可以有 128 个字节。

示例：

```
"PROTFILE"
"_N_PROTFILE"
"_N_PROTFILE_MPF"
"/_N_MPF_DIR_/_N_PROTFILE_MPF/"
```

<起始行>: 待读取的文件范围的起始行（Call-By-Value 值调用参数）

类型： INT

值： 0 根据参数<行数>指定的数量，读取文件末尾前的数据行。

1 … n 第一个要需读取行的编号。

<行数>: 待读取的文件行的数量（Call-By-Value 值调用参数）

类型： INT

<结果>: 结果变量（Call-By-Reference 引用调用参数）

存有读入的文本的变量数组。

类型： STRING（最大长度为 255）

如果参数<行数>中指定的数量小于结果变量中的数组长度 [<n>,<m>]，则剩余的数组元素保持不变。

通过控制符“LF”（换行）或者“CR LF”（回车换行）表示的行尾将不保存在结果变量中。

如果文件行比定义的字符串长度长，读入的文件行会被隔断。 不会出现错误提示。

说明

二进制文件不能被读入。 系统会输出错误提示“错误的文件类型”（出错变量的返回值 = 4）。 以下的文件类型不可读: _BIN, _EXE, _OBJ, _LIB, _BOT, _TRC, _ACC, _CYC, _NCK.

示例

程序代码	注释
N10 DEF INT ERROR	; 出错变量的定义。
N20 DEF STRING[255] RESULT[5]	; 结果变量的定义。
N30 READ(ERROR,"/_N_CST_DIR/_N_TESTFILE_MPF",1,5,RESULT)	; 有主标识符、文件标识符和路径说明的文件名称。
N40 IF ERROR <>0	; 出错分析。
N50 MSG("错误"<<ERROR<<"READ 指令")	
N60 M0	
N70 ENDIF	
...	

1.21 检查文件的存在性(ISFILE)

功能

使用 ISFILE 指令可检查某个文件是否位于 NCK 的静态用户存储器(被动文件系统)中。

句法

<结果>=ISFILE("<文件名称>")

含义

ISFILE:	用于检查指定的文件是否位于被动文件系统中的指令。
<文件名称>:	需要检查是否位于被动文件系统中的文件的名称。 类型： STRING

在指定文件名称时应注意以下几点：

- 指定的文件名称不允许包含空格或控制符（ASCII 码 ≤ 32 的字符）。
- 文件名可以通过路径和文件标识指定。
  - 路径说明

    路径必须是绝对的，即以“/”开始。

    如果没有指定路径，会在当前的目录（=选中程序的目录）中查找文件。
  - 文件标识

    如果文件名不包含文件主标识(_N_),系统会自动补充。

    如果文件名中倒数第四个字符是一个下划线“_”，则后面的三个字符被视为文件标识。 只允许使用文件标识 _SPF 和_MPF，从而可以在执行所有的文件指令时使用相同的文件名称，如通过 STRING 类型的变量。

    如果没有指定标记(“_MPF”或“_SPF”)，系统会自动补充 _MPF。
- 文件名的长度最多可以有 32 个字节,路径的长度最多可以有 128 个字节。

示例：

```
"PROTFILE"
"_N_PROTFILE"
"_N_PROTFILE_MPF"
"/_N_MPF_DIR_/_N_PROTFILE_MPF/"
```

<结果>: 用于接收检查结果的变量

类型： BOOL

值： TRUE 文件存在

FALSE 文件不存在

示例

程序代码	注释
N10 DEF BOOL RESULT	; 结果变量的定义。
N20 RESULT=ISFILE("TESTFILE")	
N30 IF(RESULT==FALSE)	
N40 MSG（“文件不存在”）	
N50 M0	

程序代码	注释
N60 ENDIF	
...	

或者

程序代码	注释
N10 DEF BOOL RESULT	; 结果变量的定义。
N20 RESULT=ISFILE("TESTFILE")	
N30 IF(NOT ISFILE("TESTFILE"))	
N40 MSG（"文件不存在"）	
N50 M0	
N60 ENDIF	
...	

1.22 读取文件信息(FILEDATE, FILETIME, FILESIZE, FILESTAT, FILEINFO)

功能

借助指令 FILEDATE、FILETIME、FILESIZE、FILESTAT 和 FILEINFO 可以读取特定文件的信息，如：上次读访问时的日期/时间、当前文件大小、文件状态或信息总和。

说明

文件必须位于 NCK 的静态用户存储器中 (被动文件系统)。

前提条件

当前所设置的保护级别必须等于或者大于上一级目录的 Show（显示）权限。 否则系统会拒绝访问并且显示出错提示（出错变量的返回值=13）。

句法

```
DEF INT <错误>
DEF STRING[<字符串长度>] <结果>
FILE....(<错误>,"<文件名称>",<结果>)
```

含义

FILEDATE:	FILEDATE 指令会提供上次读取指定文件的**日期**信息。
FILETIME:	FILETIME 指令会提供上次读取指定文件的**时间**信息。
FILESIZE:	FILESIZE 指令会提供指定文件的**当前大小**信息。
FILESTAT:	FILESTAT 指令会提供指定文件的**状态**信息，即读写权限和执行权限。
FILEINFO:	FILEINFO 指令会提供指定文件的**文件信息总和**，即可以通过 FILEDATE、FILETIME、FILESIZE 和 FILESTAT 读取的信息。
<错误>:	返回错误值的变量（Call-By-Reference 引用调用参数）
	类型： INT
	值： 0 无错误
	1 非法路径
	2 路径未找到

3 文件未找到

4 错误的文件类型

13 访问权限不够

22 结果变量（<结果>）的字符串长度太短。

<文件名称>: 需要读取信息的文件的名称。

类型： STRING

在指定文件名称时应注意以下几点：

- 如果指定的文件名称含有空格或者控制符（ASCII 代码<= 32 的字符），就会中断 FILE...指令并且显示出错标识 1“路径非法”。
- 文件名可以通过路径和文件标识指定。
  - 路径说明

    路径必须是绝对的，即以“/”开始。

    如果没有指定路径，会在当前的目录（=选中程序的目录）中查找文件。
  - 文件标识

    如果文件名不包含文件主标识(_N_),系统会自动补充。

    如果文件名中倒数第四个字符是一个下划线“_”，则后面的三个字符被视为文件标识。 只允许使用文件标识 _SPF 和 _MPF，从而可以在执行所有的文件指令时使用相同的文件名称，如通过 STRING 类型的变量。

    如果没有指定标记(“_MPF”或“_SPF”)，系统会自动补充 _MPF。
- 文件名的长度最多可以有 32 个字节,路径的长度最多可以有 128 个字节。

示例：

```
"PROTFILE"
"_N_PROTFILE"
"_N_PROTFILE_MPF"
"/_N_MPF_DIR_/_N_PROTFILE_MPF/"
```

<结果>: 结果变量（Call-By-Reference 引用调用参数）

保存所获取的文件信息的变量。

类型： STRING 当： FILEDATE

格式： “dd.mm.yy”

⇒ 字符串长度必须为 **8**。

			FILETIME 格式：“ hh:mm.ss” ⇒ 字符串长度必须为 **8**。 FILESTAT 格式：“rwxsd” (r: **r**ead, w: **w**rite, x: e**x**ecute, s: **s**how, d: **d**elete) ⇒ 字符串长度必须为 **5**。 FILEINFO 格式：“rwxsd nnnnnnnn dd.mm.yy hh:mm:ss” ⇒ 字符串长度必须为 **32**。
	INT	当：	FILESIZE 文件大小以字节表示。

示例

程序代码	注释
N10 DEF INT ERROR	; 出错变量的定义。
N20 STRING[32] RESULT	; 结果变量的定义。
N30 FILEINFO(ERROR,"/_N_MPF_DIR/_N_TESTFILE_MPF",RESULT)	; 有主标识符、文件标识符和路径说明的文件名称。
N40 IF ERROR <>0	; 错误计值
N50 MSG("错误"<<ERROR<<"FILEINFO 指令")	
N60 M0	
N70 ENDIF	
...	

在本示例中，结果变量 RESULT 会提供以下结果：

"77777 12345678 26.05.00 13:51:30"

1.23 通过数组计算校验和(CHECKSUM)

功能

借助指令 CHECKSUM 可以通过数组计算校验和。 将该校验和与之前计算出的校验和结果相比，可以确定此数组的数据是否发生改变。

应用

检查输入轮廓在切削时是否已修改过。

句法

```
DEF INT <错误>
DEF STRING[<字符串长度>] <校验和>
DEF ... <数组>[<n>,<m>,<o>]
<错误>=CHECKSUM(<校验和>,"<数组>"[,<起始栏>,<结束栏>])
```

含义

CHECKSUM: 通过数组计算校验和的指令

<错误>: 返回错误值的变量

类型： INT

值：

0	无错误
1	符号未找到
2	无数组
3	索引 1 太大
4	索引 2 太大
5	无效的数据类型
10	校验和溢出

<校验和>: 用于接收校验和计算结果的变量（Call-By-Reference 引用调用参数）

类型： STRING

要求的字符串长度： 16

校验和显示为 16 进制的字符串。
但是不带格式符。

示例： “A6FC3404E534047C”

<数组>: 构成校验和的数组的名称（Call-By-Value 值调用参数）

类型： STRING

最大字符串长度： 32

允许的数组为以下类型的 1 维到 3 维数组：

BOOL, CHAR, INT, REAL, STRING

提示：
不允许指定机床数据的数组。

<起始栏>: 计算校验和的数组的起始栏编号（可选参数）

<结束栏>: 计算校验和的数组的结束栏编号（可选参数）

说明

<起始栏>和<结束栏>为可选参数。 如果没有指定栏目索引，校验和通过整个数组构成。
校验和的结果总是唯一的。 数组元素的修改会产生另一个结果字符串。

示例

程序代码	注释
N10 DEF INT ERROR	; 出错变量的定义。
N20 DEF STRING[16] MY_CHECKSUM	; 结果变量的定义。
N30 DEF INT MY_VAR[4,4]	; 数组定义。
N40 MY_VAR=...	
N50 ERROR=CHECKSUM(MY_CHECKSUM,"MY_VAR",0,2)	
...	

在本示例中，结果变量 MY_CHECKSUM 会提供以下结果：

“A6FC3404E534047C”

1.24 取整 (ROUNDUP)

功能

通过功能“ROUNDUP”可以将 REAL 型的输入值（带小数点的数字）取整为一个较大的整数值。

句法

```
ROUNDUP(<值>)
```

含义

ROUNDUP:	用于取整输入值的指令
<值>：	REAL 型的输入值

说明

原样返回一个 INTEGER 型的输入值（一个整数）。

示例

示例 1： 不同的输入值及其取整结果

示例	取整结果
ROUNDUP(3.1)	4.0
ROUNDUP(3.6)	4.0
ROUNDUP(-3.1)	-3.0
ROUNDUP(-3.6)	-3.0
ROUNDUP(3.0)	3.0
ROUNDUP(3)	3.0

示例 2： NC 程序中的 ROUNDUP

程序代码

```
N10 X=ROUNDUP(3.5) Y=ROUNDUP(R2+2)
N15 R2=ROUNDUP($AA_IM[Y])
N20 WHEN X=100 DO Y=ROUNDUP($AA_IM[X])
...
```

1.25 子程序

1.25.1 概述

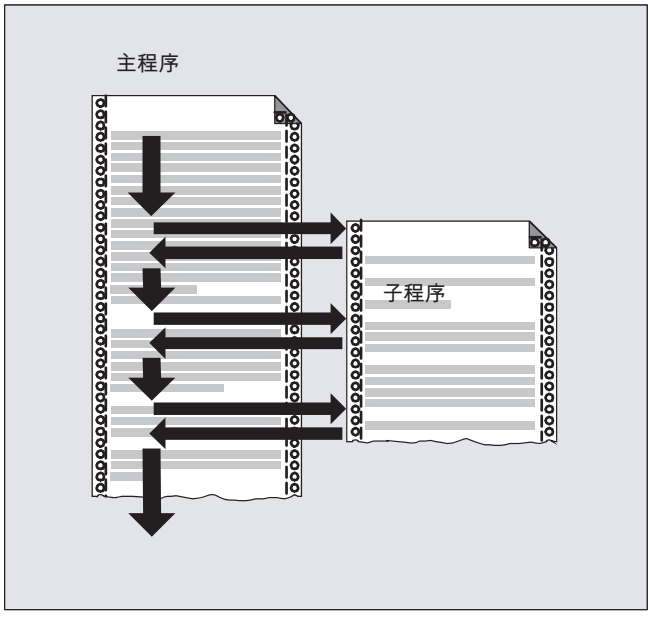
1.25.1.1 子程序

功能

在零件程序之前还固定区分为“主程序”和“子程序”的时候，就出现了“子程序”的概念。 其中，主程序指在控制系统上选择加以处理，随后启动的零件程序。 而子程序指由主程序调用的零件程序。

在目前的 SINUMERIK NC 语言中，这种固定的划分已不再存在。 原则上，每个零件程序既可以作为主程序选择并启动；也可以作为子程序由另一个零件程序调用。

因此，随着子程序定义的演变，零件程序指可以由另一个零件程序调用的程序。



应用

如同所有高级的编程语言一样，在 NC 语言中也使用了子程序，以便将一些多次应用的程序部分保存为独立、封闭的程序。

子程序具有以下优点：

- 提高了清晰性和程序可读性
- 通过重复使用测试的程序部分提高了质量
- 可以提供建立专门的加工库
- 节省了存储空间

1.25.1.2 子程序名称

命名规则

在命名子程序时应注意以下规定：

- 开始的两个字符必须是字母(A - Z, a - z)。
- 后面的字符可以是字母、数字(0 - 9)和下划线(“_”)的任意组合。
- 名称最多允许使用 31 个字符。

说明
在 SINUMERIK NC 语言中不区分大小写。

程序名称的扩展

在控制系统内部会为创建程序时给定的名称添加前缀名和后缀名：

- 前缀名： _N_
- 后缀名：
  - 主程序： _MPF
  - 子程序： _SPF

程序名称的使用

在使用程序名称时，如调用子程序时，可以组合所有的前缀名、程序名称和后缀名。

示例：

名为“SUB_PROG”的子程序可以通过以下调用方法启动：

1. `SUB_PROG`
2. `_N_SUB_PROG`

3. SUB_PROG_SPF

4. _N_SUB_PROG_SPF

说明

主程序和子程序的名称相同

如果主程序(.MPF)和子程序(.SPF)的名称相同，在零件程序中使用程序名时，必须给出相应的后缀名，以明确区分程序。

1.25.1.3 子程序的嵌套

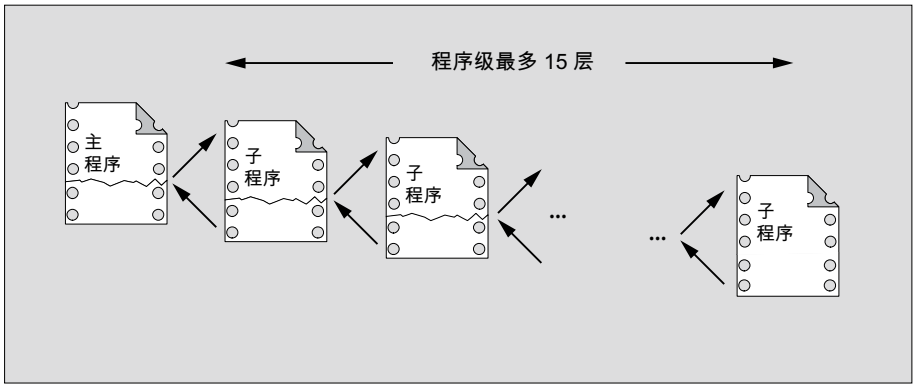
一个主程序可以调用子程序，而这个子程序又能继续调用一个子程序。 因此各个程序以相互嵌套的方式运行。 此时，每个程序都在各自的程序级上运行。

嵌套深度

NC 语言目前提供 16 个程序级。 主程序始终在最高的程序级上运行，即 0。而子程序始终在下一个更低级别的程序级上运行。 因此，程序级 1 是第一个子程序级。

程序级的划分：

- 程序级 0： 主程序级
- 程序级 1 - 15： 子程序级 1 - 15



中断程序（ASUP）

如果在中断程序的范围内调用了子程序，该程序将不会在通道中当前生效的程序级(n)上执行，而是在下一个更低级别的程序级(n+1)上执行。 考虑到中断程序，为了在最低的程序级上也能执行上述步骤，还另外提供了 2 个程序级（16 和 17）。

如果为此需要的程序级大于 2，必须在构建通道中处理的零件程序时加以考虑。即：应为中断程序的处理预留足够多的程序级。

如果中断程序处理需要 4 个程序级，那么零件程序最多只能占用 13 个程序级。 在进行中断时，这 4 个程序级（14~17）将发挥作用。

西门子循环

西门子循环为此需要使用 3 个程序级。 因此必须最迟在以下程序级中调用西门子循环：

- 零件程序处理： 程序级 12：
- 中断程序： 程序级 14：

1.25.1.4 查找路径

在调用没有指定路径的子程序时，控制系统会按照规定的顺序查找以下目录：

顺序	目录	描述
1.	当前目录	待调用程序的目录
2.	/_N_SPF_DIR /	全局子程序目录
3.	/_N_CUS_DIR /	用户循环
4.	/_N_CMA_DIR /	机床制造商循环
5.	/_N_CST_DIR /	标准循环

1.25.1.5 形式参数和实际参数

形式参数和实际参数通常与带参数传递的子程序的定义和调用相关。

形式参数

在定义子程序时必须定义需要传递给子程序的参数（即形式参数）的类型和名称。

形式参数由此定义了子程序的接口。

示例：

程序代码	注释
`PROC KONTUR (REAL X, REAL Y)`	`; 形式参数： X 和 Y，都是 REAL 型`
`N20 X1=X Y1=Y`	`; 将轴 X1 运行到位置 X 上；轴 Y1 运行到位置 Y 上`

程序代码	注释
...	
N100 RET	

实际参数

在调用子程序时，必须将绝对值或变量，即实际参数传递给子程序。

在调用时，实际参数由此为子程序接口填充实际值。

示例：

程序代码	注释
N10 DEF REAL BREITE	; 变量定义
N20 BREITE=20.0	; 变量赋值
N30 KONTUR(5.5, BREITE)	; 子程序调用，带实际参数： 5.5 和 BREITE
...	
N100 M30	

1.25.1.6 参数传递

定义一个带参数传递的子程序

通过关键字 PROC、一张包含了所有子程序需要的参数的完整列表可以定义一个带参数传递的子程序。

不完整的参数传递

在调用子程序时，不需要总是显式传递所有在子程序接口中定义的参数。 如果省略了一个参数，会传递缺省值“0”给该参数。

但为了明确区分参数的顺序，必须始终用逗号隔开参数。 最后一个参数后面不需要逗号。 如果在调用时略去该参数，最后一个逗号也可以省略。

示例：

子程序：

程序代码	注释
PROC SUB_PROG (REAL X, REAL Y, REAL Z)	; 形式参数：X, Y 和 Z
...	
N100 RET	

主程序：

程序代码	注释
PROC MAIN_PROG	
...	
N30 SUB_PROG(1.0,2.0,3.0)	; 子程序调用，带完整的参数传递： X=1.0, Y=2.0, Z=3.0
…	
N100 M30	

示例：在 N30 中调用子程序，带完整的参数传递：

```
N30 SUB_PROG( ,2.0,3.0)          ; X=0.0, Y=2.0, Z=3.0
N30 SUB_PROG(1.0, ,3.0)          ; X=1.0, Y=0.0, Z=3.0
N30 SUB_PROG(1.0,2.0)            ; X=1.0, Y=2.0, Z=0.0
N30 SUB_PROG( , ,3.0)            ; X=0.0, Y=0.0, Z=3.0
N30 SUB_PROG( , , )              ; X=0.0, Y=0.0, Z=0.0
```

小心
Call-by-Reference 引用调用式参数传递 在调用子程序时不应省略引用调用方式传递的参数。

小心
数据类型 AXIS 在调用子程序时不应省略 AXIS 数据类型的参数。

检查传递参数

借助系统变量“P_SUBPAR [n]”（其中 n = 1, 2, ...）可以检查子程序中是否显式传递或省略了某个参数。 索引 n 指形式参数的顺序。 索引 n = 1 表示第 1 个形式参数；索引 n = 2 表示第 2 个形式参数，依此类推。

下面的程序段落举例说明了如何检查第 1 个形式参数。

编程	注释
PROC SUB_PROG (REAL X, REAL Y, REAL Z)	; 形式参数：X, Y 和 Z
N20 IF $P_SUBPAR[1]==TRUE	; 检查第 1 个形式参数 X。

编程	注释
...	; 如果显式传递了形式参数 X，执行此动作。
N40 ELSE	
...	; 如果没有显式传递形式参数 X，执行此动作。
N60 ENDIF	
...	; 通用动作
N100 RET	

1.25.2 定义子程序

1.25.2.1 没有参数传递的子程序

功能

在定义没有参数传递的子程序时，可以省略程序头的定义行。

句法

```
[PROC <程序名称>]
...
```

含义

PROC:	程序开头的定义指令
<程序名称>:	程序的名称

示例

示例 1： 子程序，带 PROC 指令

程序代码	注释
PROC SUB_PROG	; 定义行
N10 G01 G90 G64 F1000	
N20 X10 Y20	
...	

程序代码	注释
N100 RET	; 子程序返回

示例 2： 子程序，不带 PROC 指令

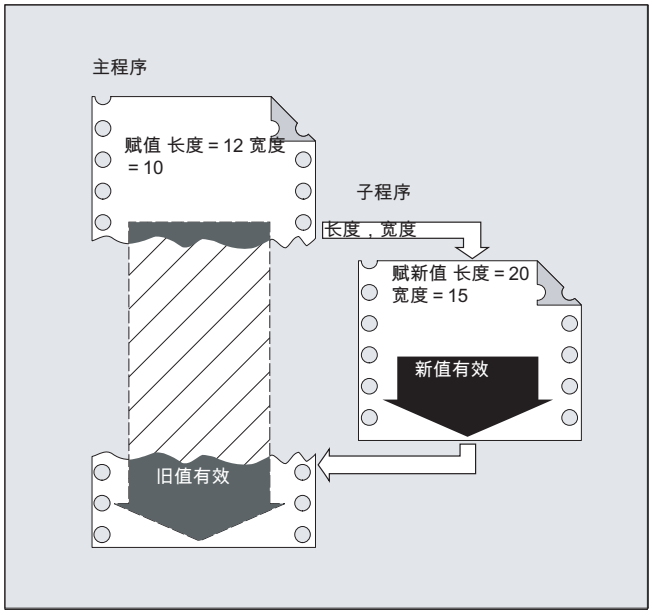
程序代码	注释
N10 G01 G90 G64 F1000	
N20 X10 Y20	
...	
N100 RET	; 子程序返回

1.25.2.2 子程序，带 Call-by-Value 值调用式参数传递(PROC)

功能

通过关键字 PROC、程序名称、一张包含了所有子程序需要的参数类型和名称的完整列表，就可以定义一个带 Call-by-Value 值调用式参数传递的子程序。 定义指令必须位于第一个程序行中。

值调用式参数传递不会对主调程序产生影响。 主调程序只向子程序传递实际参数的值。



说明

最多可以传递 127 个参数。

句法

```
PROC <程序名称> (<参数类型> <参数名称>, ...)
```

含义

PROC:	程序开头的定义指令
<程序名称>:	程序的名称
<参数类型>:	参数的数据类型，如 REAL, INT, BOOL
<参数名称>:	参数名称

注意
关键字 PROC 后指定的程序名称必须和操作界面上指定的程序名称一致。

示例

定义带 2 个 REAL 型参数的子程序：

程序代码	注释
PROC SUB_PROG (REAL LAENGE, REAL BREITE)	; 参数 1 类型： REAL，名称： LAENGE 参数 2 类型：REAL，名称： BREITE
...	
N100 RET	; 子程序返回

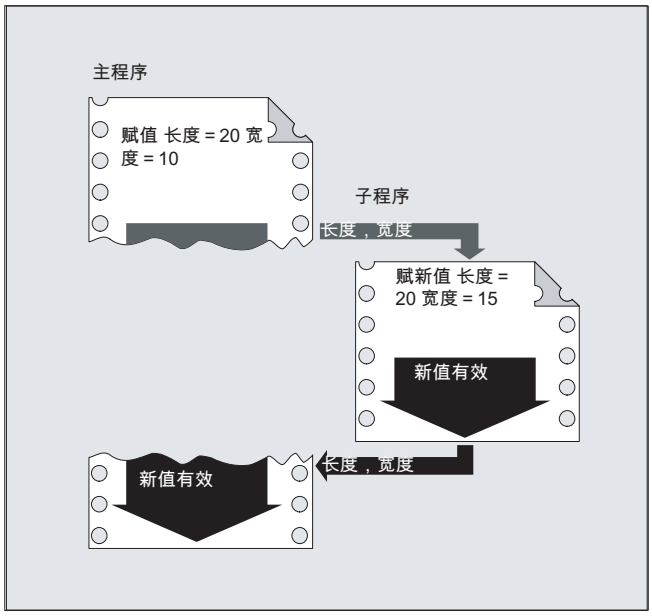
1.25.2.3 子程序，带 Call-by-Reference 引用调用式参数传递(PROC, VAR)

功能

通过关键字 PROC、程序名称、一张包含了所有子程序需要的参数关键字 VAR、类型和名称的完整列表，就可以定义一个带 Call-by-Reference 引用调用式参数传递的子程序。 定义指令必须位于第一个程序行中。

在引用调用式的参数传递中，也可以传递数组的引用。

引用调用式参数传递会对主调程序产生影响。 主调程序向子程序传递实际参数的引用，并由此使得子程序能够直接访问相关变量。



说明

最多可以传递 127 个参数。

说明

因此只有当在主调程序中定义了传递变量(LUD)时，才需要按照引用调用的方法传递参数。 而通道全局变量或 NC 全局变量无需传递，因为子程序也能够直接访问这些变量。

句法

```
PROC <程序名称> (VAR <参数类型> <参数名称>, ...)
PROC <程序名称> (VAR <数组类型> <数组名称> [<m>,<n>,<o>], ...)
```

含义

PROC:	程序开头的定义指令
VAR:	按照引用进行的参数传递的关键字
<程序名称>:	程序的名称
<参数类型>:	参数的数据类型，如 REAL, INT, BOOL
<参数名称>:	参数名称
<数组类型>:	数组元素的数据类型，如 REAL, INT, BOOL

<数组名称>:	数组的名称
[<m>,<n>,<o>]:	数组长度 目前最多可以为 3 维数组： <m>: 1 维数组长度 <n>: 2 维数组长度 <o>: 3 维数组长度

注意
关键字 PROC 后指定的程序名称必须和操作界面上指定的程序名称一致。

说明

子程序可以将不确定长度的数组用作形式参数，来处理可变长度的数组。 为此在定义一个形式参数的二维数组时，不规定 1 维的长度。但是必须写上逗号。

示例： PROC <程序名称> (VAR REAL FELD[,5])

示例

定义带 2 个参数（作为 REAL 型的引用）的子程序：

程序代码	注释
PROC SUB_PROG(VAR REAL LAENGE, VAR REAL BREITE)	; 参数 1 对 REAL 型的引用，名称：LAENGE 参数 2 对 REAL 型的引用，名称：BREITE
...	
N100 RET	

1.25.2.4 保存模态 G 功能（SAVE）

功能

属性 SAVE 用于保存子程序调用前激活的模态 G 功能，在子程序结束后再次激活。

小心
连续路径运行的中断 如果在连续路径运行生效时调用了含 SAVE 属性的子程序，则在此子程序结束（返回）时连续路径运行会中断。

句法

```
PROC <子程序名称> SAVE
```

含义

SAVE: 保存子程序调用前激活的模态 G 功能，并使功能在子程序结束后再次生效

示例

在子程序 KONTUR 中模态 G 功能 G91 有效（增量尺寸）。 在主程序中模态 G 功能 G90 有效（绝对尺寸）。 通过带 SAVE 的子程序定义，G90 在主程序中的子程序结束后再次生效。

子程序定义：

程序代码	注释
PROC KONTUR (REAL WERT1) SAVE	; 带参数 SAVE 的子程序定义
N10 G91 ...	; 模态 G 功能 G91: 增量尺寸
N100 M17	; 子程序结束

主程序：

程序代码	注释
N10 G0 X... Y... G90	; 模态 G 功能 G90: 绝对尺寸
N20 ...	
...	
N50 KONTUR (12.4)	; 子程序调用
N60 X... Y...	; 模态 G 功能 G90 通过 SAVE 再次激活

边界条件

框架

与带属性 SAVE 的子程序相关的框架特性取决于框架类型，并可以通过机床数据设置。

文献

功能手册 基本功能；轴、坐标系、框架（K2）、
章节“带 SAVE 的子程序跳转”。

1.25.2.5 抑制单程序段处理 (SBLOF, SBLON)

功能

全部程序的单程序段抑制

带有 SBLOF 标记的程序，在有效单程序段处理时如同一个程序段一样进行完整处理，即对于整个程序，抑制单程序段处理。

SBLOF 位于 PROC 行，并且一直有效，直至子程序结束或者中断。 使用返回指令判断在子程序结束处是否被停止：

通过 M17 跳回：	停止于子程序末尾处
通过 RET 跳回：	在子程序末尾处不停止

程序内的单程序段抑制

SBLOF 必须单独在程序段中。 从这个程序段起，关闭单段至：

- 下一个 SBLON

  或者

- 生效子程序级的结束处

句法

全部程序的单程序段抑制：
```
PROC ... SBLOF
```

程序内的单程序段抑制：

```
SBLOF
...
SBLON
```

含义

PROC:	一个程序的第一个指令
SBLOF:	用于关闭单程序段处理的指令 SBLOF 可以位于一个 PROC 程序段中，或者单独位于程序段中。
SBLON:	用于打开单程序段处理的指令 SBLON 必须位于一个独立的程序段中。

边界条件

- **单程序段抑制和程序段显示**

  可以在循环/子程序中使用 DISPLOF 抑制当前的程序段显示。 如果 DISPLOF 连同 SBLOF 一起编程，则在循环/子程序之内在单程序段停止时，如同在调用循环/子程序之前一样显示。

- **系统 ASUP 或用户 ASUP 中的单程序段抑制**

  如果系统或用户 ASUP 中的单程序段停止通过在机床数据 MD10702 $MN_IGNORE_SINGLEBLOCK_MASK 中设置进行抑制，（Bit0 = 1 或 Bit1 = 1），则单程序段停止可以通过在 ASUP 中编程 SBLON 再次激活。

  如果用户 ASUP 中的单程序段停止通过在机床数据 MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP 中设置进行抑制，则单程序段停止通过在 ASUP 中编程 SBLON **无法** 再次激活。

- **不同的单程序段处理类型，单程序段抑制的特性**

  在激活的单程序段处理 SBL2（在零件程序段后停止）时，当在 MD10702 $MN_IGNORE_SINGLEBLOCK_MASK（避免单程序段停止）设置 Bit 12 为“1”时，在 SBLON 程序段中 **不** 停止。

  在激活的单程序段处理 SBL3（也在循环中零件程序段后停止）时，指令 SBLOF 被抑制。

示例

示例 1： 某个程序内的单程序段抑制

程序代码	注释
N10 G1 X100 F1000	
N20 SBLOF	; 关闭单段

程序代码	注释
N30 Y20	
N40 M100	
N50 R10=90	
N60 SBLON	; 再次激活单程序段
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

主程序：

程序代码
N10 G1 X10 G90 F200
N20 X-4 Y6
N30 CYCLE1
N40 G1 X0
N50 M30

循环 CYCLE1：

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

当激活单程序段时执行循环 CYCLE1，即处理 CYCLE1 时，必须按一次“启动”按钮。

示例 3：

为激活已修改的零点偏移和刀具补偿而被 PLC 启动的 ASUP 应该被隐藏。

程序代码
N100 PROC NV SBLOF DISPLOF

程序代码

```
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： 通过 MD10702 Bit 12 = 1 不停止

初始情况：

- 单程序段处理激活。
- MD10702 $MN_IGNORE_SINGLEBLOCK_MASK Bit12 = 1

主程序：

程序代码	注释
N10 G0 X0	; 在该零件程序行中停止
N20 X10	; 在该零件程序行中停止
N30 CYCLE	; 由循环产生的运行程序段。
N50 G90 X20	; 在该零件程序行中停止
M30	

循环 CYCLE：

程序代码	注释
PROC CYCLE SBLOF	; 抑制单程序段停止
N100 R0 = 1	
N110 SBLON	; 由于 MD10702 Bit12=1，在该零件程序行中不停止。
N120 X1	; 在该零件程序行中停止
N140 SBLOF	
N150 R0 = 2	

程序代码	注释
RET	

示例 5： 程序嵌套时单程序段抑制

初始情况：

单程序段处理激活。

程序嵌套：

程序代码	注释
N10 X0 F1000	; 在该程序段中停止。
N20 UP1(0)	
PROC UP1(INT _NR) SBLOF	; 抑制单程序段停止。
N100 X10	
N110 UP2(0)	
PROC UP2(INT _NR)	
N200 X20	
N210 SBLON	; 激活单程序段停止。
N220 X22	; 在该程序段中停止。
N230 UP3(0)	
PROC UP3(INT _NR)	
N300 SBLOF	; 抑制单程序段停止。
N305 X30	
N310 SBLON	; 激活单程序段停止。
N320 X32	; 在该程序段中停止。
N330 SBLOF	; 抑制单程序段停止。
N340 X34	
N350 M17	; SBLOF 激活。
N240 X24	; 在该程序段中停止。 SBLON 激活。
N250 M17	; 在该程序段中停止。 SBLON 激活。
N120 X12	
N130 M17	; 在跳回程序段中停止。 PROC 语句的 SBLOF 激活。
N30 X0	; 在该程序段中停止。
N40 M30	; 在该程序段中停止。

其它信息

异步子程序单段禁止

为了在某个步骤中执行单程序段中的 ASUP，必须在 ASUP 中编程一个带有 SBLOF 的 PROC 指令。 这也适用于功能“可编辑的系统 ASUP”（MD11610 $MN_ASUP_EDITABLE）。

可编辑系统 ASUP 举例：

程序代码	注释
N10 PROC ASUP1 SBLOF DISPLOF	
N20 IF $AC_ASUP=='H200'	
N30 RET	; 当 BA 转换时没有 REPOS。
N40 ELSE	
N50 REPOSA	; 所有其它情况中的 REPOS。
N60 ENDIF	

在单段中的程序影响

在单程序段处理中，用户可以按程序段方式执行零件程序。 有下列设置类型：

- SBL1： 在每个机床功能程序段后面带有停止的 IPO 单程序段。
- SBL2: 单段，在每个程序段之后停顿
- SBL3: 在循环中停顿（通过选择 SBL 3 抑制 SBLOF 指令）。

程序嵌套时单段抑制

如果在一个子程序中编程 SBLOF 在 PROC 语句中，则用 M17 停止到子程序跳回。 由此防止在调用的程序中已经执行下一个程序段。 如果在某个子程序中使用 SBLOF（PROC 语句中没有 SBLOF）激活某个单程序段抑制，就只有在调用程序的下一个机床功能程序段之后停止。 如果不希望如此，则在子程序中在跳回之前（M17）必须再次编程 SBLON。 在一个上一级的程序中，在用 RET 跳回时，不停止。

1.25.2.6 抑制当前的程序段显示(DISPLOF, DISPLON, ACTBLOCNO)

功能

在标准情况下，程序段显示画面中会显示当前的程序段。 在循环或子程序中可以通过指令 DISPLOF 抑制当前程序段的显示。 显示循环的调用或者子程序的调用，而不显示当前的程序段。 借助指令 DISPLON 可以再次恢复程序段显示。

DISPLOF 或 DISPLON 应写入包含 PROC 指令的程序行中，它作用于整个子程序，并会隐式影响所有该子程序调用的其他子程序，这些子程序中不包含 DISPLON 或 DISPLOF 指令。 这个属性同样针对 ASUP。

句法

```
PROC … DISPLOF
PROC … DISPLOF ACTBLOCNO
PROC … DISPLON
```

含义

DISPLOF: 用于抑制当前程序段显示的指令。

位置： 在包含 PROC 指令的程序行的结尾

有效性： 直至从子程序返回或者程序结束。

提示：

如果从带 DISPLOF 指令的子程序调用其他子程序，而在这些子程序中没有显式编程 DISPLON，则也抑制其中的当前程序段显示。

DISPLON: 用于恢复当前程序段显示的指令

位置： 在包含 PROC 指令的程序行的结尾

有效性： 直至从子程序返回或者程序结束。

提示：

如果从带 DISPLON 指令的子程序调用其他子程序，而在这些子程序中没有显式编程 DISPLOF，则也激活其中的当前程序段显示。

ACTBLOCNO: DISPLOF 连同属性 ACTBLOCNO 一起作用，在报警情况下会输出出现报警的当前程序段的号码。 同样，如果在较低等级的程序级中只编程了 DISPLOF，也会输出相应号码。

与此相反，DISPLOF 不带 ACTBLOCNO 时，循环或子程序调用号码由上一个不带 DISPLOF 标记的程序级显示。

示例

示例 1： 在循环中抑制当前程序段显示

程序代码	注释
PROC CYCLE (AXIS TOMOV, REAL POSITION) SAVE DISPLOF	; 抑制当前的程序段显示。 换而言之，如果要显示循环调用，例如： CYCLE(X,100.0)
DEF REAL DIFF	; 循环内容
G01 ...	
...	
RET	; 子程序跳回。 在程序段显示中在循环调用上显示下列程序段。

示例 2： 发出报警时程序段显示

子程序 SUBPROG1（带有 ACTBLOCNO）：

```
PROC SUBPROG1 DISPLOF ACTBLOCNO
N8000 R10 = R33 + R44
...
N9040 R10 = 66 X100                    ; 触发报警 12080
...
N10000 M17
```

子程序 SUBPROG2（不带 ACTBLOCNO）：

```
PROC SUBPROG2 DISPLOF
N5000 R10 = R33 + R44
...
N6040 R10 = 66 X100            ; 触发报警 12080
...
N7000 M17
```

主程序：

```
N1000 G0 X0 Y0 Z0
N1010 ...
...
N2050 SUBPROG1                 ; 发出报警 = "12080 通道 K1 程序段 N9040 同步错误 对于文本
                                 R10="
N2060 ...
N2350 SUBPROG2                 ; 发出报警 = "12080 通道 K1 程序段 N2350 同步错误 对于文本
                                 R10="
...
N3000 M30
```

示例 3： 恢复当前程序段显示

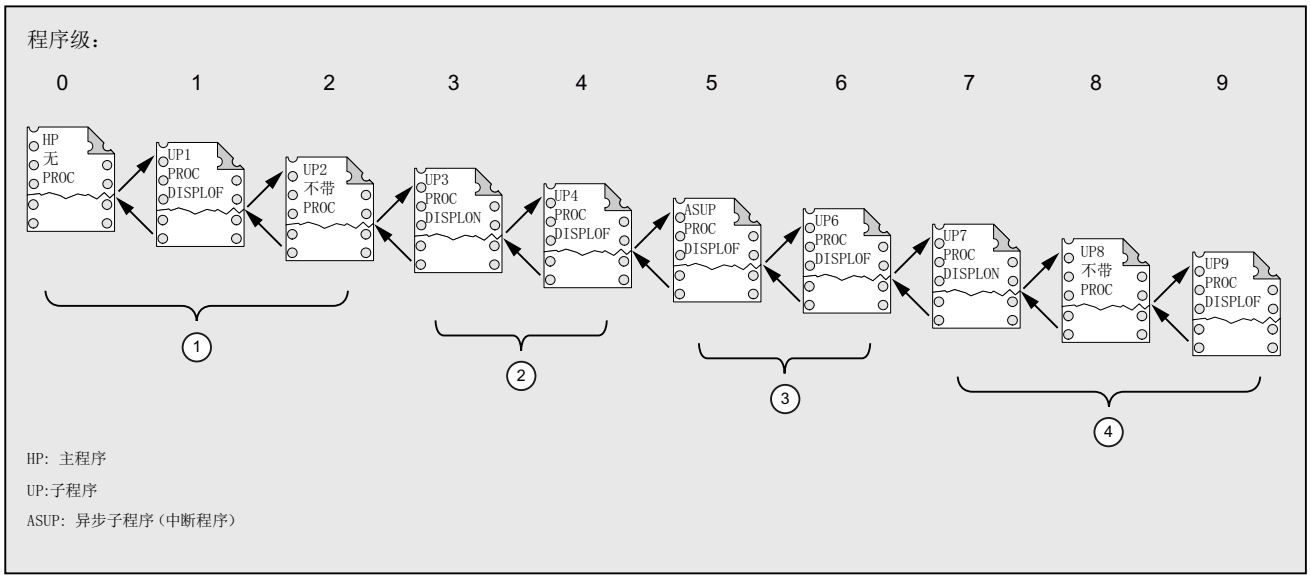
子程序 SUB1 带抑制：

程序代码	注释
PROC SUB1 DISPLOF	; 抑制子程序 SUB1 中当前程序段显示。 而应显示带 SUB1 调用的程序段。
...	
N300 SUB2	; 调用 子程序 SUB2。
...	
N500 M17	

子程序 SUB2 不带抑制：

程序代码	注释
PROC SUB2 DISPLON	; 激活子程序 SUB2 中的当前程序段显示。
...	
N200 M17	; 返回到子程序 SUB1。 在 SUB1 中再次抑制当前程序段显示。

示例 4： 不同的 DISPLON/DISPLOF 组合下的显示属性



① 在当前程序段显示中，会显示从程序级 0 开始的零件程序行。
② 在当前程序段显示中，会显示从程序级 3 开始的零件程序行。
③ 在当前程序段显示中，会显示从程序级 3 开始的零件程序行。
④ 在当前程序段显示中会显示从程序级 7/8 开始的零件程序行。

1.25.2.7 标记子程序“准备”(PREPRO)

功能

关键字 PREPRO 可以在引导启动中 PROC 指令行结尾处标记所有文件。

说明

程序预处理的方式取决于相应设置的机床数据。 参见机床制造商说明。

文献：

功能手册 特殊功能；预处理 (V2)

句法

```
PROC … PREPRO
```

含义

PREPRO: 关键字，用于标记引导启动中经过预处理的文件以及循环目录中保存的 NC 程序

读入经过预处理的子程序和子程序调用

不管是在启动中经过处理的、带参数的子程序，还是子程序调用，循环目录的处理顺序都相同：

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

如果带相同名称的 NC 程序有不同的特征，则首先激活找到的 PROC 指令而忽略其它 PROC 指令，而不输出报警提示。

1.25.2.8 子程序返回指令 M17

功能

返回指令 M17 或零件程序结束指令 M30 位于子程序的末尾。 它使得程序执行返回到主调程序中、子程序调用指令后的零件程序段上。

说明

M17 和 M30 在 NC 语言中视为同等的指令。

句法

```
PROC <程序名称>
...
M17/M30
```

边界条件

子程序返回对连续路径运行的影响

如果 M17 或 M30 位于单独的零件程序段中，则通道中激活的连续路径运行被中断。

为避免此类中断，应在最后一个运行程序段中写入 M17 或 M30。 此外，还必须将以下机床数据设为 0：

MD20800 $MC_SPF_END_TO_VDI = 0 （没有 M30/M17 输出给 NC/PLC 接口 ）

示例

1. M17 位于单独程序段中的子程序

程序代码	注释
N10 G64 F2000 G91 X10 Y10	
N20 X10 Z10	
N30 M17	; 返回，中断连续路径运行。

2. M17 位于最后一个运行程序段中的子程序

程序代码	注释
N10 G64 F2000 G91 X10 Y10	
N20 X10 Z10 M17	; 返回，不中断连续路径运行。

1.25.2.9 子程序返回指令 RET

功能

在子程序中也可以代替 M17 而编程指令 RET。 RET 必须在一个单独的零件程序段中编程。 和 M17 类似，RET 使得程序执行返回到主调程序中、子程序调用指令之后的零件程序段上。

说明

编程参数可以修改RET的返回属性（参见“可设定的子程序返回 (RET ...) (页 182)”）。

应用

如果不希望因为返回而中断 G64 连续路径运行（G641 ... G645），则必须使用 RET 指令。

前提条件

只能在未定义 SAVE 属性的子程序中使用 RET 指令。

句法

```
PROC <程序名称>
...
RET
```

示例

主程序：

程序代码	注释
PROC MAIN_PROGRAM	; 程序开始

程序代码	注释
...	
N50 SUB_PROG	; 子程序调用： SUB_PROG
N60 ...	
...	
N100 M30	; 程序结束

子程序：

程序代码	注释
PROC SUB_PROG	
...	
N100 RET	; 返回到主程序的程序段 N60。

1.25.2.10 可设定的子程序返回 (RET ...)

功能

通常如果子程序的末尾为 RET 或 M17，程序执行会返回到主调程序中，并接着从子程序调用指令后的程序行开始。

此外还有种使用情况，程序要在其他位置继续处理，例如：

- 在 ISO 语言方式下调用切削循环之后，会根据轮廓说明继续程序加工。
- 在故障处理时，从任意一个子程序级（也在 ASUP 之后）返回到主程序。
- 返回需越过几个程序级，用于在编译循环和 ISO 语言方式中的特殊应用。

在这些情况下指令 RET 与“返回参数”一起编程。

句法

RET("<目标程序段>")
RET("<目标程序段>",<目标程序段后的程序段>)
RET("<目标程序段>",<目标程序段后的程序段>,<返回级的数量>)
RET("<目标程序段>", ,<返回级的数量>)
RET("<目标程序段>",<目标程序段后的程序段>,<返回级的数量>,
<返回程序头>)
RET(, ,<返回级的数量>,<返回程序头>)

含义

RET: 子程序末尾（替代 M17 应用）

<目标程序段>: 返回参数 1

将要继续编程处理的程序段称作返回目标。

如果返回参数 3 未编程，则返回目标位于主调程序中。

允许的说明有：

"<程序段号码>" 目标程序段号码

"<跳转标记>" 跳转标记，该跳转标记必须设置在目标程序段中。

"<字符串>" 字符串，该字符串必须在程序中已知（例如程序或者变量名称）

对于目标程序段中的字符串编程，有下列规则：

- **末尾处空格** （与通过一个 ":" 在末尾处标记的跳转标记有区别 ）。
- **字符串前** 仅允许设置一个程序段号码和/或一个跳转标记，**没有程序指令**。

<目标程序段后的程序段>: 返回参数 2

与返回参数 1 有关。

类型： INT

值： 0 跳回到通过返回参数 1 规定的程序段上。

> 0 跳回到紧跟在通过返回参数 1 规定的程序段后面的程序段上。

<返回级的数量>: 返回参数 3

列出要越过的程序级数量，以便到达需要继续处理程序的程序级。

类型： INT

值： 1 程序就在“当前程序级 －1”中继续执行 (如同不带参数的 RET)。

	2	程序在“当前程序级－2”中继续执行，即越过一级。
	3	程序在“当前程序级－3”中继续执行，即越过两级。
	...	
取值范围：	1 ... 15	
<返回到程序头>:	返回参数 4	
类型：	BOOL	
值：	1	当跳回到主程序中且主程序中已有一个 **ISO-语言模式** 激活时，就回到程序头。

说明

如果一个子程序返回中指定了一个字符串用于目标程序段查找，则始终首先在主调程序中查找跳转标记。

如果要通过一个字符串明确定义返回目标，该字符串不允许与跳转标记同名，否则子程序总是返回该跳转标记，而不会返回到该字符串（参见示例 2）。

边界条件

在越过几个程序级返回时，会分析各个程序级的 SAVE 指令。

如果在越过几个程序级返回时已有一个模态子程序激活，且如果在某个被跳过的子程序中已经为该模态子程序编程了取消指令 MCALL，那么该模态子程序将继续保持激活状态。

⚠小心

编程人员必须注意，在越过几个程序级返回时使用正确的模态设置继续执行。 例如，通过编程一个相应的主程序段就可做到这一点。

示例

示例 1： 在 ASUP 处理之后，在主程序中继续

编程	注释
N10010 CALL "UP1"	; 程序级 0（主程序）

编程	注释
N11000 PROC UP1	; 程序级 1：
N11010 CALL "UP2"	
N12000 PROC UP2	; 程序级 2：
...	
N19000 PROC ASUP	; 程序级 3（ASUP 处理）
...	
N19100 RET("N10900", ,$P_STACK)	; 子程序返回
N10900	; 在主程序中继续。
N10910 MCALL	; 关闭模态子程序。
N10920 G0 G60 G40 M5	; 修改其它模态设置。

示例 2： 字符串 (<String>) 作为目标程序段查找数据

主程序：

程序代码	注释
PROC MAIN_PROGRAM	
N1000 DEF INT iVar1=1, iVar2=4	
N1010 ...	
N1200 subProg1	; 调用子程序"subProg1"
N1210 M2 S1000 X10 F1000	
N1220	
N1400 subProg2	; 调用子程序"subProg2"
N1410 M3 S500 Y20	
N1420 ..	
N1500 lab1: iVar1=R10*44	
N1510 F500 X5	
N1520 ...	
N1550 subprog1: G1 X30	; "subProg1" 这里定义为跳转标记。
N1560 ...	
N1600 subProg3	调用子程序"subProg3"
N1610 ...	
N1900 M30	

子程序 subProg1:

程序代码	注释
PROC subProg1	

程序代码	注释
N2000 R10=R20+100	
N2010 ...	
N2200 RET("subProg2")	; 返回到主程序中的程序段 N1400

子程序 subProg2:

程序代码	注释
PROC subProg2	
N2000 R10=R20+100	
N2010 ...	
N2200 RET("iVar1")	; 返回到主程序中的程序段 N1500

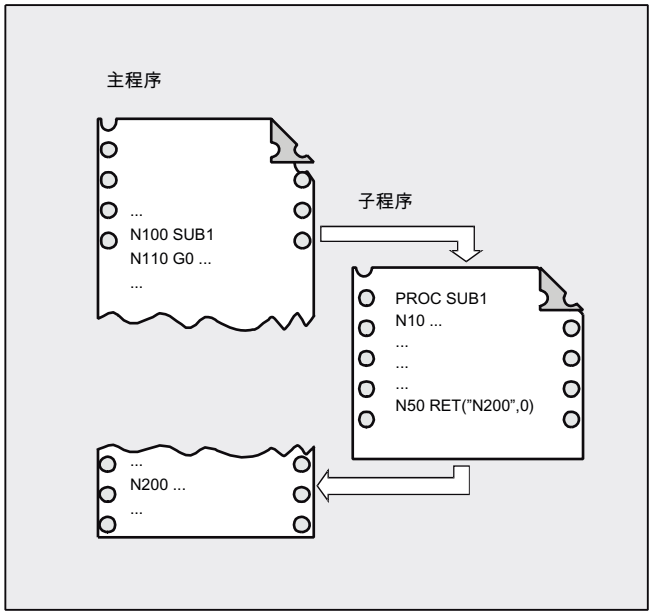
子程序 subProg3:

程序代码	注释
PROC subProg3	
N2000 R10=R20+100	
N2010 ...	
N2200 RET("subProg1")	; 返回到主程序中的程序段 N1550

其它信息

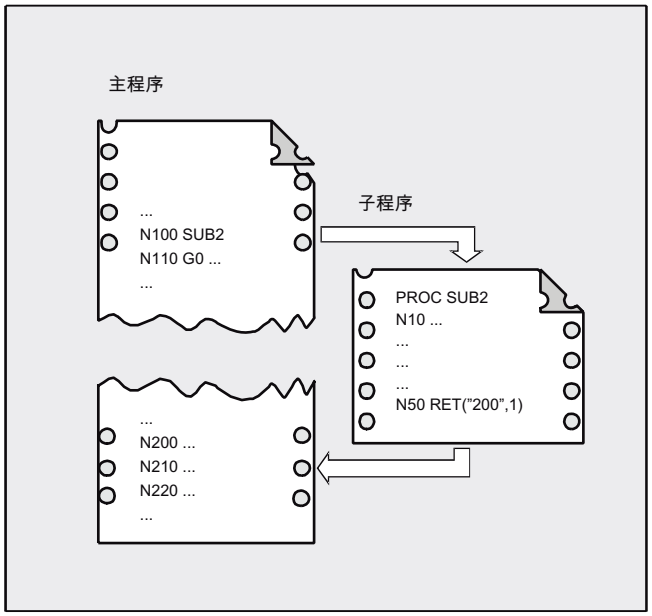
下列图形可以说明返回参数 1－3 的不同效用。

1. 返回参数 1 = "N200", 返回参数 2 = 0



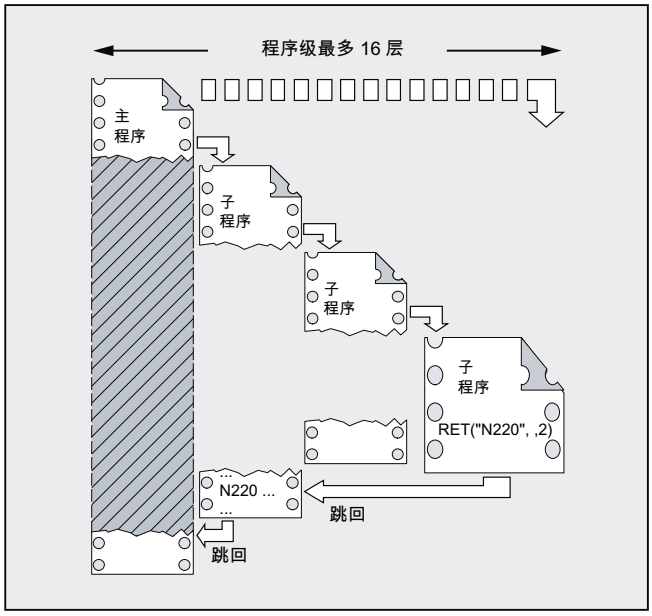
根据 RET 指令，程序处理继续通过主程序中程序段 N200 进行。

2. 返回参数 1 = "N200", 返回参数 2 = 1



根据 RET 指令，程序处理继续通过程序段 （N210 ）进行，该程序段紧跟在主程序中的程序段 N200 之后。

3. 返回参数 1 = "N220", 返回参数 3 = 2



根据 RET 指令，跳回到两个程序级，程序处理继续通过程序段 N220 进行。

1.25.3 子程序调用

1.25.3.1 没有参数传递的子程序调用

功能

调用子程序时，可以使用地址 L 加子程序号，或者直接使用程序名称。

一个主程序也可以作为子程序调用。 此时，主程序中设置的程序结束指令 M2 或 M30 视作 M17（返回到主调程序的程序结束）处理。

说明

同样，一个子程序也可以作为主程序启动。

控制系统的查找方法：

是否有 *_MPF ?

是否有 *_SPF ?

接着： 如果被调子程序的名称和主程序的名称相同，则再次调用主调主程序。 一般这种情况不应发生，所以主程序和子程序的名称必须相互区别，不得相同。

说明

从一个初始化文件中可以调用无需参数传递的子程序。

句法

L<编号>/<程序名称>

说明

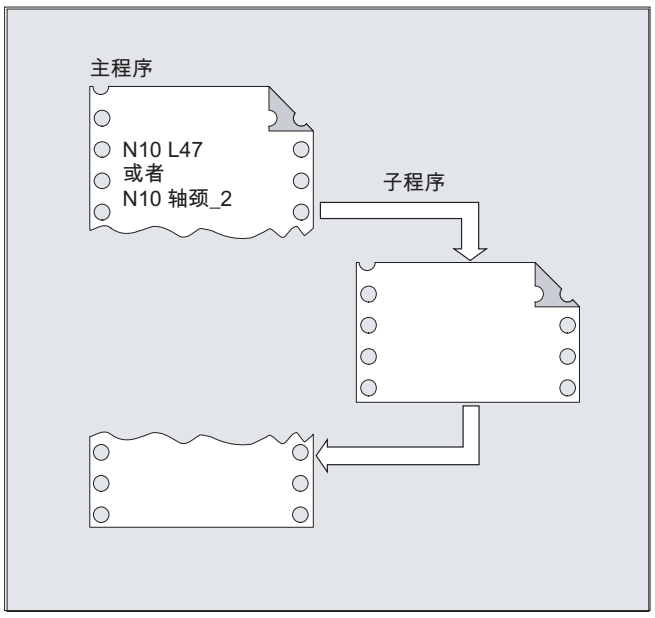
子程序调用必须在独立的 NC 程序段中编程。

含义

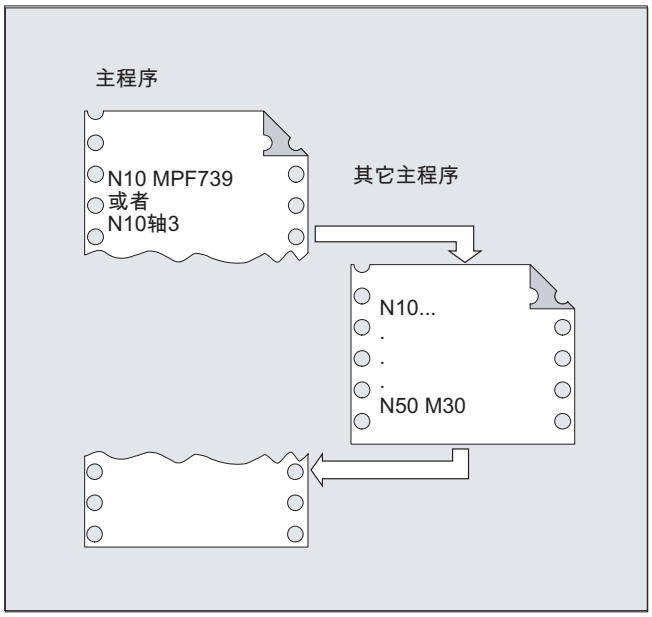
L:	子程序调用地址	
<编号>:	子程序号码	
	类型：	INT
	值：	最多 7 位数
		注意： 数值中开始的零在命名时具有不同的含义（⇒ L123, L0123 和 L00123 表示三个不同的子程序）。
<程序名称>:	子程序或主程序的名称	

示例

示例 1： 调用一个不带参数传递的子程序



示例 2： 作为子程序调用主程序



1.25.3.2 带参数传递的子程序调用(EXTERN)

功能

在带参数传递的子程序调用时，可以直接传递变量或者数值（不针对 VAR 参数）。

必须在调用之前在主程序中使用 EXTERN 声明带参数传递的子程序，例如，在程序头。其中应给出子程序的名称以及传递顺序中的变量类型。

⚠小心
不管是变量类型还是传递的顺序，均必须和子程序中 PROC 所约定的定义相符。参数名称可以在主程序和子程序中不一样。

句法

```
EXTERN <程序名称>(<类型_参数 1>,<类型_参数 2>,<类型_参数 3>)
...
<程序名称>(<数值_参数 1>,<数值_参数 2>,<数值_参数 3>)
```

⚠小心
子程序调用必须在独立的 NC 程序段中编程。

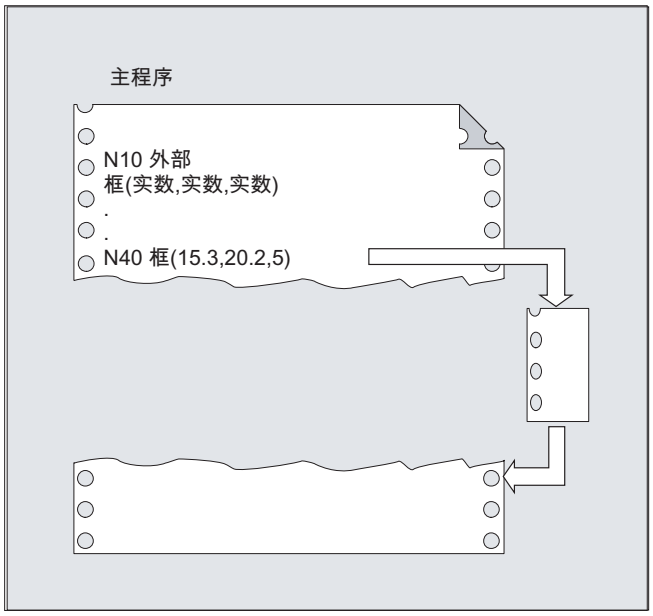
含义

<程序名称>:	子程序名称
EXTERN:	关键字，用于带有参数传递的子程序声明 **提示：** 必须仅当子程序在工件中或者在全局子程序目录中时，才可指定 EXTERN。循环不必声明为 EXTERN 。
<类型_参数 1>,<类型_参数 2>,<类型_参数 3>:	传递序列中要传递的参数变量类型
<数值_参数 1>,<数值_参数 2>,<数值_参数 3>:	要传递的参数变量值

示例

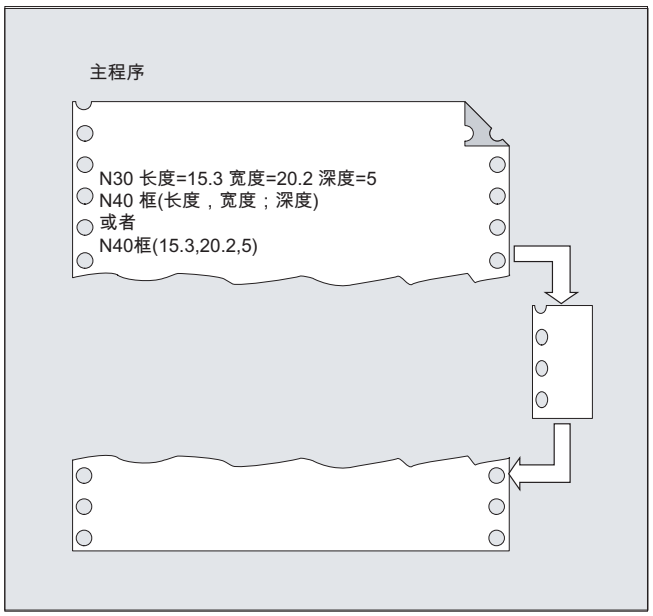
示例 1： 子程序调用，事先声明

程序代码	注释
N10 EXTERN RAHMEN(REAL,REAL,REAL)	; 子程序说明。
...	
N40 RAHMEN(15.3,20.2,5)	; 调用带参数传递的子程序。



示例 2： 子程序调用， 无声明

程序代码	注释
N10 DEF REAL LAENGE, BREITE, TIEFE	
N20 …	
N30 LAENGE=15.3 BREITE=20.2 TIEFE=5	
N40 RAHMEN(LAENGE,BREITE,TIEFE)	; 或者 N40 RAHMEN(15.3,20.2,5)



1.25.3.3 程序重复次数(P)

功能

如果一个子程序需要多次连续执行，则可以在该程序段中在地址 P 下编程重复调用的次数。

> ⚠小心
>
> **带程序重复和参数传递的子程序调用**
> 参数仅在程序调用时或者第一次执行时传送。 在后续重复过程中，这些参数保持不变。如果您在程序重复时要修改参数，则您必须在子程序中确定相应的协议。

句法

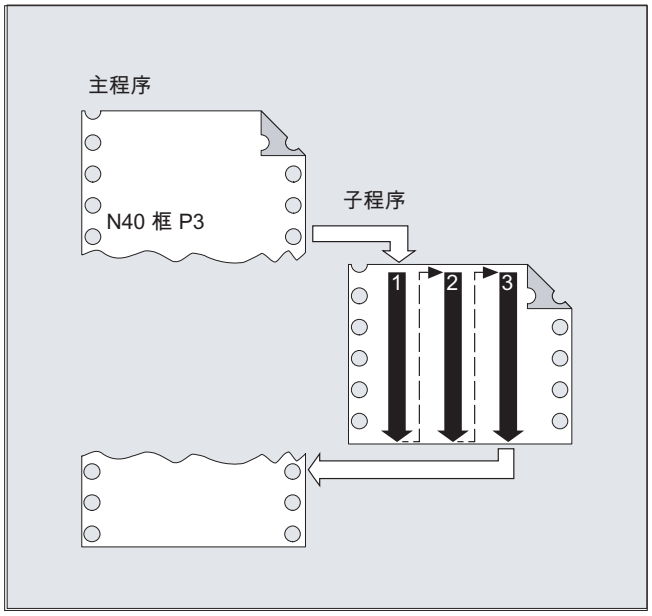
<程序名称> P<值>

含义

<程序名称>:	子程序调用	
P:	程序重复的编程地址	
<值>:	程序重复次数	
	类型：	INT
	取值范围：	1 … 9999 （不带正负号）

示例

程序代码	注释
...	
N40 RAHMEN P3	; 子程序"RAHMEN"应被连续执行三次。
...	



1.25.3.4 模态子程序调用 (MCALL)

功能

在通过 MCALL 进行模态子程序调用时，子程序可以在每个带轨迹运行的程序段之后自动调用和执行。 可自动调用要在不同工件位置执行的子程序，例如用于建立钻孔图时。

功能关闭通过 MCALL 实现，不调用子程序，或者通过编程一个新的模态子程序调用，用于一个新的子程序。

⚠小心

在某个程序执行过程中，同时只能有一个 MCALL 调用生效。 在 MCALL 调用中仅传送一次参数。

在下面的情况下也可以调用模态子程序，而不编程一个运动：

- 在编程地址 S 和 F，当 G0 或 G1 有效时。
- G0/G1 单独编程在程序段中，或者与其它的 G 代码一起编程。

句法

MCALL <程序名称>

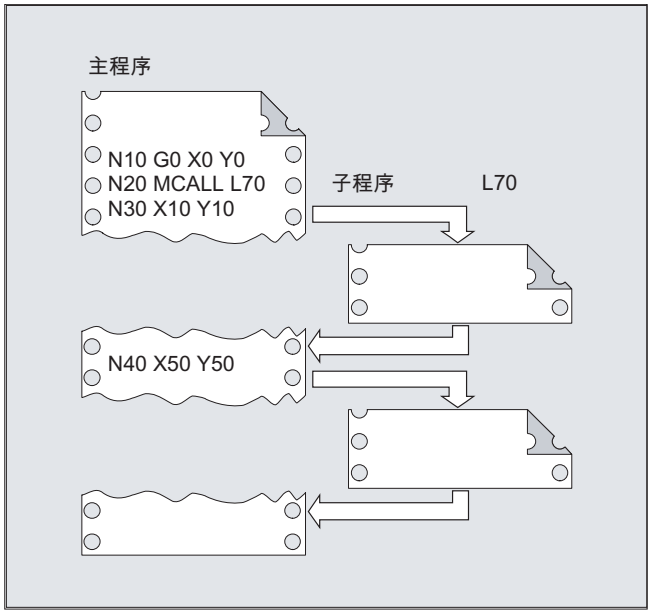
含义

MCALL:	用于模态子程序调用的指令
<程序名称>:	子程序名称

示例

示例 1：

程序代码	注释
N10 G0 X0 Y0	
N20 MCALL L70	; 模态子程序调用。
N30 X10 Y10	; 接近编程位置，并接着处理子程序 L70。
N40 X50 Y50	; 接近编程位置，并接着处理子程序 L70。



示例 2：

程序代码
N10 G0 X0 Y0
N20 MCALL L70
N30 L80

在这个例子中，子程序 L80 中有编程的轨迹轴和后续的 NC 程序段。 L70 通过 L80 调用。

1.25.3.5 间接子程序调用(CALL)

功能

根据所给定的条件，可以在一个地点调用不同的子程序。 这里子程序名称存放在一个字符串类型的变量中。 子程序调用通过 CALL 和变量名进行。

⚠小心
间接调用子程序仅可以用于没有参数传递的子程序。 直接调用某个子程序时，可将名称保存在一个字符串常量中

句法

CALL <程序名称>

含义

CALL:	用于间接子程序调用的指令
<程序名称>:	子程序的名称（变量或常量）
	类型： STRING

示例

使用字符串常量直接调用：

程序代码	注释
…	
CALL "/_N_WKS_DIR/_N_SUBPROG_WPD/_N_TEIL1_SPF"	; 使用 CALL 直接调用子程序 TEIL1。
…	

使用变量间接调用：

程序代码	注释
…	
DEF STRING[100] PROGNAME	; 定义变量。
PROGNAME="/_N_WKS_DIR/_N_SUBPROG_WPD/_N_TEIL1_SPF"	; 将变量 PROGNAME 指定给子程序 TEIL1。
CALL PROGNAME	; 通过 CALL 和变量 PROGNAME 间接调用子程序 TEIL1。
…	

1.25.3.6 指定待执行部分的间接子程序调用(CALL BLOCK ... TO ...)

功能

通过 CALL 和关键字组合 BLOCK ... TO 可以间接调用一个子程序，并执行用起始标签和结束标签标记的程序部分。

句法

```
CALL <程序名称> BLOCK <起始标签> TO <结束标签>
CALL BLOCK <起始标签> TO <结束标签>
```

含义

CALL:	用于间接子程序调用的指令
<程序名称>:	子程序名称（变量或常量），其中包含了要处理的程序部分**(指定 可选)**。 类型： STRING **提示：** 如果没有编程<程序名称>，则在当前程序中查找带有<起始标签>和<结束标签>标记的程序部分并执行此部分。
BLOCK ... TO ... :	用于间接执行程序部分的关键字组合
<起始标记>:	表明要处理的程序部分开头的变量 类型： STRING
<结束标记>:	表明要处理的程序部分末尾的变量。 类型： STRING

示例

主程序：

程序代码	注释
...	
DEF STRING[20] STARTLABEL, ENDLABEL	; 起始标记和结束标记的变量定义。
STARTLABEL="LABEL_1"	
ENDLABEL="LABEL_2"	
...	
CALL "CONTUR_1" BLOCK STARTLABEL TO ENDLABEL	; 间接的子程序调用并标记待执行的程序部分。
...	

子程序：

程序代码	注释
PROC CONTUR_1 ...	
LABEL_1	; 起始标记： 执行程序部分的开始
N1000 G1 ...	
...	
LABEL_2	; 结束标记： 执行程序部分的结束
...	

1.25.3.7 间接调用某个以 ISO 语言编程的程序 (ISOCALL)

功能

利用间接程序调用 ISOCALL，可以调用一个用 ISO 语言编程的程序。 由此激活机床数据中设定的 ISO 模式。 在程序结束处，原先的加工方式再次生效。 如果在机床数据中没有设定 ISO 方式，则子程序调用以西门子方式进行。

有关 ISO 模式的其它信息，参见：
文献：
功能说明 ISO 语言

句法

ISOCALL <程序名称>

含义

ISOCALL:	子程序调用关键字，由此激活机床数据中设定的 ISO 模式。
<程序名称>:	ISO 语言编程的程序名称（STRING 型的变量和常量）

示例： 使用 ISO 模式的循环编程调用轮廓

程序代码	注释
0122_SPF	; 以 ISO 模式描述轮廓
N1010 G1 X10 Z20	
N1020 X30 R5	
N1030 Z50 C10	
N1040 X50	
N1050 M99	
N0010 DEF STRING[5] PROGNAME = "0122"	; 西门子零件程序（循环）
...	
N2000 R11 = $AA_IW[X]	
N2010 ISOCALL PROGNAME	
N2020 R10 = R10+1	; 以 ISO 模式编辑程序 0122.spf
...	
N2400 M30	

1.25.3.8 调用带有路径说明和参数的子程序 (PCALL)

功能

利用 PCALL 可以调用带绝对路径说明和参数传送的子程序。

句法

PCALL <路径/程序名称>(<参数 1>,…,<参数 n>)

含义

PCALL:	关键字，用于带绝对路径说明的子程序调用
<路径/程序名称>:	绝对的路径说明，以“/”开始，包括子程序名。 如果没有说明绝对路径，则 PCALL 表现如同一个带程序名的标准子程序调用。 不使用前缀 _N_ 和后缀说明程序标识符。 如果要给程序名编写前缀和后缀，就必须通过指令 EXTERN 明确说明前缀和后缀。
<参数 1>, ...:	实际参数符合子程序的 PROC 指令。

示例

程序代码
PCALL/_N_WKS_DIR/_N_WELLE_WPD/WELLE(参数 1,参数 2,…)

1.25.3.9 扩展调用子程序时的路径查找 (CALLPATH)

功能

使用指令 CALLPATH 可以扩展查找路径用于子程序调用。

这样就可以从某个未选中的工件目录中调用子程序，而无需指定完整、绝对的子程序路径名。

在输入用户循环之前扩展查找路径 (_N_CUS_DIR)。

通过下列结果再次选择查找路径扩展：

- CALLPATH 带空格
- CALLPATH 不带参数
- 零件程序结束
- Reset

句法

```
CALLPATH("<路径名称>")
```

含义

CALLPATH: 关键字，用于可编程的查找路径扩展。
在一个自身的零件程序部分中编程。

<路径名称>: 字符串型常量或变量。 包含某个目录的绝对路径说明，以此来扩展查找路径。 路径说明以 "/" 开始。 路径必须使用前缀和后缀进行完整说明。 最大的路径长度达到 128 个字节。

如果<路径名>包含一个空字符串，或者调用不带参数的 CALLPATH，则查找路径指令被再次复位。

示例

程序代码

```
CALLPATH ("/_N_WKS_DIR/_N_MYWPD_WPD")
```

以此来设置下列查找路径（位置 5 是新建的）：

1. 当前的目录/子程序名称
2. 当前目录/子程序标识符_SPF
3. 当前目录/子程序标识符_MPF
4. /_N_SPF_DIR/子程序名称_SPF
5. **/_N_WKS_DIR/_N_MYWPD/子程序标识符_SPF**
6. /N_CUS_DIR/_N_MYWPD/子程序标识符_SPF
7. /_N_CMA_DIR/子程序名称_SPF
8. /_N_CST_DIR/子程序名称_SPF

边界条件

- CALLPATH 用来检查所编写的路径名是否存在。 在故障情况下，零件程序加工带补偿程序段报警 14009 中断。
- CALLPATH 也可以在 INI 文件中编程。 然后就会对 INI 文件的处理时间产生影响 (WPD-INI-文件或者用于 NC-活动文件的初始化程序，例如第 1 个通道中的框架 _N_CH1_UFR_INI). 然后再次复位查找路径。

1.25.3.10 执行外部子程序 (EXTCALL)

功能

使用 EXTCALL 指令可从外部程序存储器（本地驱动、网络驱动、USB 驱动）下载和执行子程序。

在以下设定数据中可以预设到外部子程序目录的路径：

SD42700 $SC_EXT_PROG_PATH

此路径和 EXTCALL 调用时给定的子程序路径或者子程序名一起组成了待调用程序的完整路径。

说明

外部子程序不允许包含任何跳转指令，例如：GOTOF、GOTOB、CASE、FOR、LOOP、WHILE 或者 REPEAT。

可以有 IF-ELSE-ENDIF 结构。

可以进行子程序调用和嵌套的 EXTCALL 调用。

句法

```
EXTCALL("<路径/><程序名称>")
```

含义

EXTCALL:	调用一个外部子程序的指令	
"<路径/><程序名称>":	字符串型常量/变量	
	<路径/>:	绝对或相对路径说明（**可选**）
	<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**）。 示例： "WELLE" 或者 "WELLE_SPF" 或 "WELLE.SPF"

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:** CF 卡
- **USB:** 前端 USB 接口

CF_CARD: 和 **LOCAL_DRIVE:** 可交替使用。

说明

通过 USB 驱动外部执行

如果需要通过 USB 接口从外部 USB 驱动器传输外部程序，则应只使用通过 X203 、名称为 “TCU_1” 的接口。

注意

通过 USB 闪存（前端 USB 接口上）外部执行

不推荐从 USB 闪存驱动器直接执行程序。

在持续运行中，USB 闪存驱动器可能会接触不良、掉落、由于碰撞或不小心拔出而折断。

如其在刀具加工期间断开，则将导致加工立即停止并且工件被损坏。

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 中预设到子程序的路径：

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")
```

其它信息

EXTCALL 调用，带绝对路径说明

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。

EXTCALL 调用，带相对路径说明/不带路径说明

在进行带相对路径说明/不带路径说明的 EXTCALL 调用时，根据下列模式查找存在的程序存储器：

- 如果在 SD42700 $SC_EXT_PROG_PATH 中预设了路径说明，则首先从此路径出发查找 EXTCALL 中的设定（程序名或者相对路径说明）。 而绝对路径由字符串组成：
  - SD42700 中预设的路径说明
  - "/" 为分隔符
  - 在 EXTCALL 所说明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，则继续从用户存储器的目录查找 EXTCALL 调用的说明。
- 一旦找到子程序，查找结束。 如果没有查找子程序，则程序中断。

可设定的加载存储器（FIFO 缓存器）

在“从外部执行”模式中编辑某个程序时(主程序或者子程序)， 在 NCK 中需要有一个加载内存。 后装载存储器的大小预设置为 30 KB，可如同其它存储器相关的机床数据那样，仅由机床制造商根据需求修改。

对于所有同时在“从外部执行”模式中被处理的程序而言，必须相应设置一个加载内存。

复位和上电

通过复位和上电，可以中断外部的子程序调用，并且清除各自的后装载存储器。

选择用于“从外部执行”的子程序在 RESET/零件程序结束后，选择仍生效。 然而上电后，选择失效。

文献

“外部执行”的更多相关信息请参见：

功能手册 基本功能；BAG、通道、程序运行、复位特性 (K1)

1.25.4 循环

1.25.4.1 循环：给用户循环设定参数

功能

使用文件 cov.com 和 uc.com 可以给自有循环设定参数。

文件 cov.com 以标准循环提供，并且相应地扩展。 文件 uc.com 由用户自己编制。

这两个文件要在被动式文件系统中加载到目录“用户循环”中（或者使用相应的路径说明）：

```
;$PATH=/_N_CUS_DIR
```

配置在程序中。

文件和路径

cov.com_COM	循环概述
uc.com	循环调用说明

适配 cov.com - 循环一览表

以标准循环提供的文件 cov.com 有以下的结构：

```
%_N_COV_COM                          文件名
;$PATH=/_N_CST_DIR                   路径说明
;Vxxx 11.12.95 Sca 循环概述          注释行
C1(CYCLE81) 钻削，定中               调用用于第 1 个循环
C2(CYCLE82) 钻削，锪面               调用用于第 2 个循环
...
C24(CYCLE98) 螺纹的序列              调用用于最后一个循环
M17                                  文件结束
```

句法

对于每个新来的循环，需要在下面的句法中添加一行：

C<号> (<循环名>) 注释文本

序号： 一个任意整数，到目前为止在该文件中还不允许使用；

循环名： 待捆绑循环的程序名

注释文本： 可以选择，用于循环的一个注释文本

示例：

C25 (MEIN_ZYKLUS_1) 用户循环_1

C26 (特殊循环)

举例，文件 uc.com - 用户循环说明

根据后续示例进行说明：

以下两个循环应当重新建立一个循环参数设定：

编程	注释
PROC MEIN_ZYKLUS_1 (REAL PAR1, INT PAR2, CHAR PAR3, STRING[10] PAR4)	
；循环有以下的传送参数：	
PAR1:	; 实数值，范围为 -1000.001 <= PAR2 <= 123.456，预设置为 100
PAR2:	; 在 0 <= PAR3 <= 999999 之间的正整数值，预设置为 0
PAR3:	; 1 个 ASCII-字符
PAR4:	; 长度 10 的字符串，用于一个子程序名
...	
M17	;

编程	注释
PROC SPEZIALZYKLUS (REAL 值 1, INT 值 2)	
；循环有以下的传送参数：	
值 1:	; 没有数值范围限制和预设置的实数值
值 2:	; 整数值，没有值范围限制和预置
...	
M17	;

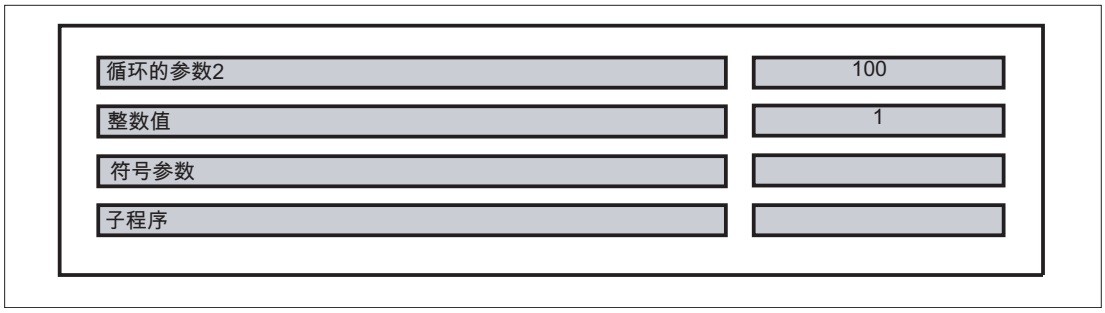
所属的文件 uc.com:

编程

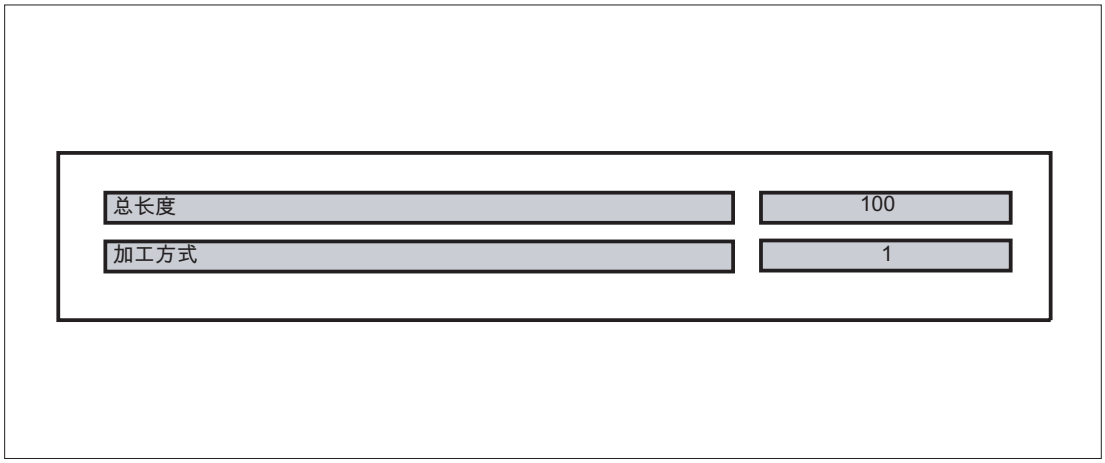
```
%_N_UC_COM
;$PATH=/_N_CUS_DIR
//C25(MEIN_ZYKLUS_1) 用户循环_1
(R/-1000.001 123.456 / 100 /该循环的参数_2)
(I/0 999999 / 1 / 整数值)
(C//"A" / 符号参数)
(S///子程序名)
//C26(SPEZIALZYKLUS)
(R///总长度)
(I/*123456/3/加工方式)
M17
```

两个循环举例

显示窗口，用于循环 MEIN_ZYKLUS_1



显示窗口，用于循环 SPEZIALZYKLUS



文件 uc.com 的语法说明 - 用户循环说明

每个循环的顶行：

如同文件 cov.com 中带有前置 “//”

//C <号> (<循环名>) 注释文本

例如：

//C25(MEIN_ZYKLUS_1) 用户循环_

每个参数的说明行：

(<数据类型标识符> / <最小值> <最大值>
/ <预置值> /<注释>)

数据类型标识符：

R	用于实数
I	用于整数
C	用于字符（1 个字符）
S	用于字符串

最小值，最大值(可以省略)

待输入值的极限，在输入时检测；超出该范围的值不可以输入。可以说明计数值，它们可以用触发键操作；这些值以“*”开始计数，其它的值不允许。

例如：

(I/*123456/1/加工方式)

如果式字符串和字符型就没有限制。

预置值(可以省略)

该值在调用循环时在相应的表征码中预置；它可以通过操作修改。

注释

文本，最多 50 个字符，在用于循环的调用表征码中、在该参数输入数组之前显示。

1.26 宏指令技术 (DEFINE ... AS)

⚠小心
使用宏指令技术可能会使控制系统的编程语言发生巨大变化！ 因此您必须要特别小心地使用宏指令技术！

功能

作为宏指令，是指单个的指令组合成一个新的总指令，带自己的名称。 G-,M-和 H-功能或者 L-子程序名也可以作为宏指令编制。 在程序运行中调用该宏指令时，可以在该宏指令名下一个接一个地执行编程的指令。

应用

总是反复的指令序列，人们仅编程一次，在一个自身的宏指令模块（宏文件）中作为宏指令，或者仅在程序开始处出现一次。 宏指令可以在任意一个主程序或者子程序中调用和执行。

激活

为了可以在 NC 程序中使用宏文件的宏指令，必须将宏文件装载到 NC 中。

句法

宏指令定义：
```
DEFINE <宏名称> AS <指令 1> <指令 2> ...
```
在 NC 程序中调用：
<宏名称>

含义

DEFINE ... AS：	关键字组合用于定义一个宏指令
<宏名称>:	宏名称 只有命名符才允许用作宏指令名称。 通过宏名称可以从 NC 程序中调用宏。
<指令>:	编程指令，应该包含在宏中。

宏定义规则

- 在宏中可以定义任意的命名符、G-/M-/H-功能和 L-程序名。
- 宏也可以在 NC 程序中约定。
- G 功能宏仅可以在宏指令模块中由系统全局约定。
- H 功能和 L 功能可以 2 位编程。
- M 功能和 G 功能可以 3 位编程。

⚠小心
不得使用宏指令对关键字和备用名称进行覆盖定义。

边界条件

不可以嵌套宏指令。

示例

示例 1： 程序开始处的宏定义

程序代码	注释
DEFINE LINIE AS G1 G94 F300	; 宏指令定义
...	
...	
N70 LINIE X10 Y20	; 宏调用
...	

示例 2： 一个宏文件中的宏定义

程序代码	注释
DEFINE M6 AS L6	; 当换刀时调用接收所需数据传送的某个子程序。 在子程序中输出实际的换刀 M 功能（例如 M106）.
DEFINE G81 AS DRILL(81)	; 模仿 DIN-G 功能。
DEFINE G33 AS M333 G333	; 在切削螺纹时要求与 PLC 的同步。 原来的 G 功能 G33 被 MD 改名为 G333，编程对于用户而言保持相同。

示例 3： 外部宏文件

在控制系统中读入外部宏文件后，必须将宏文件装载到 NC 中。 然后才可以使用 NC 程序中的宏。

程序代码	注释
%_N_UMAC_DEF	
;$PATH=/_N_DEF_DIR	; 用户特有的宏
DEFINE PI AS 3.14	
DEFINE TC1 AS M3 S1000	
DEFINE M13 AS M3 M7	; 主轴右转，冷却液开
DEFINE M14 AS M4 M7	; 主轴左转，冷却液开
DEFINE M15 AS M5 M9	; 主轴停止，冷却液关
DEFINE M6 AS L6	; 调用刀具更换程序
DEFINE G80 AS MCALL	; 撤销选择钻削循环
M30	

文件和程序管理 2

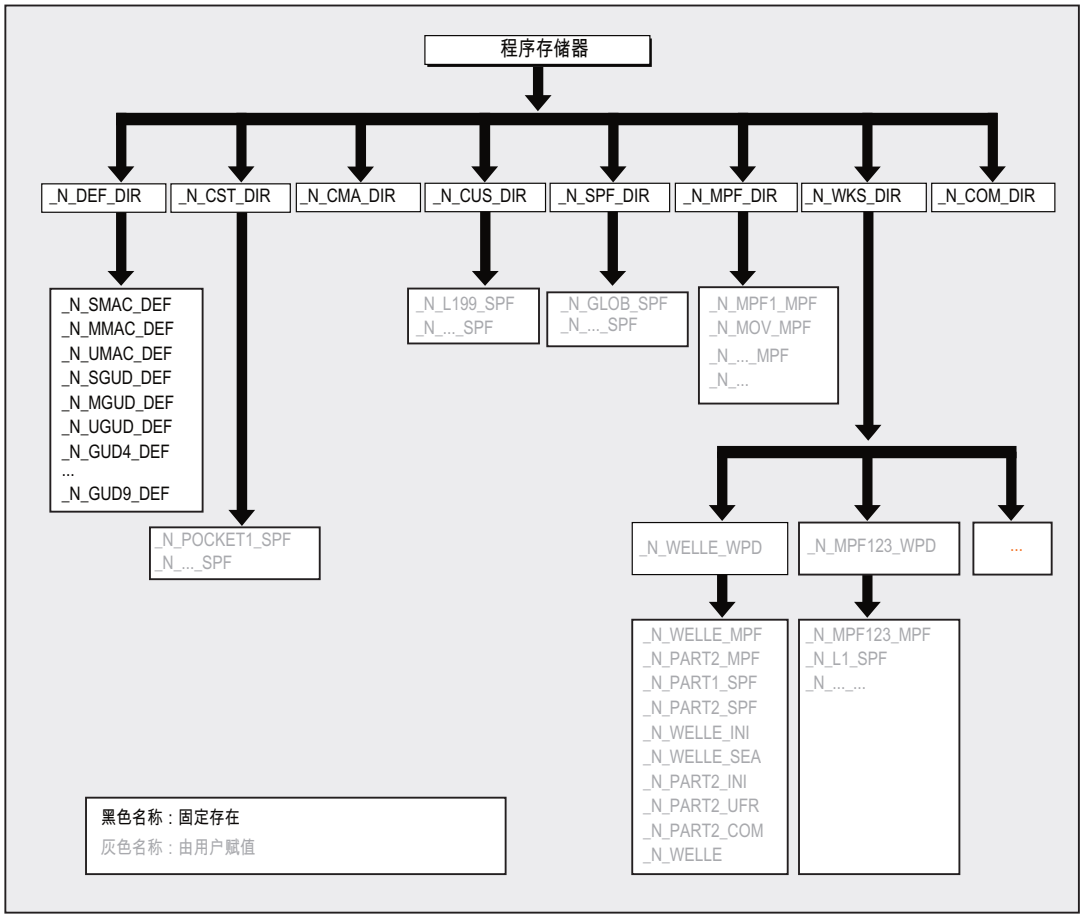
2.1 程序存储器

功能

文件和程序（例如：主程序和子程序、宏指令定义）将永久保存在程序存储器中（→ 被动文件系统）。

文献：
功能手册 扩展功能；存储器功能(S7)

此外，还有一些文件类型可以临时保存在这里并且在需要时（例如当加工某个特定的工件时）传送到工作存储器中（例如用于初始化目的）。



标准目录

正常情况下有以下目录：

目录	内容
_N_DEF_DIR	数据块和宏指令块
_N_CST_DIR	标准循环
_N_CMA_DIR	机床制造商循环
_N_CUS_DIR	用户循环
_N_WKS_DIR	工件
_N_SPF_DIR	全局子程序
_N_MPF_DIR	主程序
_N_COM_DIR	注释

文件类型

在程序存储器中可以有以下文件类型：

文件类型	描述
name_MPF	主程序
name_SPF	子程序
name_TEA	机床数据
name_SEA	设定数据
name_TOA	刀具补偿
name_UFR	零点偏移/框架
name_INI	初始化文件
name_GUD	全局用户数据
name_RPA	R 参数
name_COM	注释
name_DEF	全局用户数据和宏指令定义

工件主目录 (_N_WKS_DIR)

工件主目录以默认名称_N_WKS_DIR 建立在程序存储器中。 工件主目录包含所有编程工件的相应工件目录。

工件主目录(..._WPD)

为了可以灵活处理数据和程序，可以把某些数据和程序打包，或者存放在单独的工件目录下。

工件目录包含加工工件时所需的所有文件。 它可以是主程序，子程序，任意初始化程序和注释文件。

在选中程序后，第一次零件程序开始时一次性执行初始化程序（根据机床数据 MD11280 $MN_WPD_INI_MODE) 。

示例：

工件目录 _N_WELLE_WPD, 为工件 WELLE 所建立，包含有下列文件：

文件	说明
_N_WELLE_MPF	主程序
_N_PART2_MPF	主程序
_N_PART1_SPF	子程序
_N_PART2_SPF	子程序
_N_WELLE_INI	工件数据的常规初始化程序
_N_WELLE_SEA	设定数据初始化程序
_N_PART2_INI	程序第 2 部分数据的常规初始化程序
_N_PART2_UFR	用于程序第 2 部分框架文件的初始化程序
_N_WELLE_COM	注释文件

在外部计算机上建立工件目录

下面介绍如何在一个外部数据站中编制工件目录。 如何直接在控制系统中管理文件和程序（由 PC 到控制系统）参见操作说明。

建立带路径说明($PATH=...)的工件目录

在某个文件的第二行中，使用$PATH=... 指定目标路径。 文件存放在所说明的路径下。

示例：

程序代码

```
%_N_WELLE_MPF
;$PATH=/_N_WKS_DIR/_N_WELLE_WPD
N10 G0 X… Z…
...
M2
```

文件 _N_WELLE_MPF 保存在目 录/_N_WKS_DIR/_N_WELLE_WPD 中。

建立不带路径说明的工件目录

如果没有路径说明，则后缀名为 _SPF 的文件保存在目 录/_N_SPF_DIR 中；后缀名为 _INI 的文件保存在工作存储器中，其他文件保存在目录 /_N_MPF_DIR 中。

示例：

程序代码

```
%_N_WELLE_SPF
...
M17
```

文件 _N_WELLE_SPF 保存在目录 /_N_SPF_DIR 中。

选择用于加工的工件

可以为一个通道中的加工选择一个工件目录。 如果在该目录中有一个**同名** 主程序或者只有一个唯一的主程序(_MPF)，就自动选择该程序来执行。

文献:
/BAD/ HMI 高级操作手册；章节“工作表”以及“选择待执行程序”

在调用零件程序时编程查找路径

如果在调用某个子程序（或者初始化文件）时没有在零件程序中明确指定调用路径，则调用程序就会根据默认查找路径进行查找。

子程序调用，带绝对路径说明

示例：

程序代码

```
...
CALL"/_N_CST_DIR/_N_CYCLE1_SPF"
...
```

子程序调用，不带绝对路径说明

在正常情况下不用说明路径即可调用程序：

示例：

程序代码

```
...
CYCLE1
...
```

根据以下的顺序在目录中查找调用的程序：

编号	目录	说明
1	当前目录 / *名称*	工件主目录或者标准目录 _N_MPF_DIR
2	当前目录 / *名称_SPF*	
3	当前目录 / *名称_MPF*	
4	/_N_SPF_DIR / *名称_SPF*	全局子程序
5	/_N_CUS_DIR / *名称_SPF*	用户循环
6	/_N_CMA_DIR / *名称_SPF*	机床制造商循环
7	/_N_CST_DIR / *名称_SPF*	标准循环

编程子程序调用时的查找路径(CALLPATH)

在调用子程序时可以使用零件程序指令 CALLPATH 扩展查找路径。

示例：

程序代码

```
CALLPATH ("/_N_WKS_DIR/_N_MYWPD_WPD")
...
```

根据所指定的编程将查找路径保存在位置 5（用户循环）之前。

关于可使用 CALLPATH 对子程序调用的查找路径进行编程的其它信息可参阅“使用 CALLPATH 扩展子程序调用的查找路径”一章。

2.2 工作存储器 (CHANDATA, COMPLETE, INITIAL)

功能

工作存储器包含当前的系统数据和用户数据，控制系统以此数据运行（有源文件系统），例如

- 激活的机床数据
- 刀具补偿数据
- 零点偏移
- ...

初始化程序

这里讨论工作存储器数据可以预置（初始化）时如何编程。 可以使用以下的文件类型：

文件类型	描述
name_TEA	机床数据
name_SEA	设定数据
name_TOA	刀具补偿
name_UFR	零点偏移/框架
name_INI	初始化文件
name_GUD	全局用户数据
name_RPA	R 参数

所有文件类型的信息请参阅操作手册中的操作界面。

数据区

数据可以划分为不同的区。 例如，某个控制系统可以具有多个通道，通常也可拥有多个轴。

有：

标记	数据区
NCK	NCK 专用数据
CH<n>	通道特有的数据（<n> 用来指定通道号）
AX<n>	轴特有的数据（<n> 用来指定加工轴的编号）
TO	刀具数据
COMPLETE	所有数据

在外部计算机上生成初始化程序

利用数据区标志和数据类型标志，可以确定数据保护时视作数组的数据区：

_N_AX5_TEA_INI	用于第 5 轴的机床数据
_N_CH2_UFR_INI	通道 2 框架
_N_COMPLETE_TEA_INI	所有机床数据

在系统开机调试之后在工作存储器中有一个数据组，它保证控制系统正常运行。

多通道控制系统的工作步骤 (CHANDATA)

用于多个通道的 CHANDATA(<通道号>) 仅在文件 N_INITIAL_INI 中允许。 这是调试文件，用来初始化控制系统的所有数据。

程序代码	注释
%_N_INITIAL_INI	
CHANDATA(1)	
	; 加工轴分配通道 1：
$MC_AXCONF_MACHAX_USED[0]=1	
$MC_AXCONF_MACHAX_USED[1]=2	
$MC_AXCONF_MACHAX_USED[2]=3	
CHANDATA(2)	
	; 加工轴分配通道 2：
$MC_AXCONF_MACHAX_USED[0]=4	
$MC_AXCONF_MACHAX_USED[1]=5	
CHANDATA(1)	
	; 轴向机床数据：
	; 粗准停窗口：

程序代码	注释
$MA_STOP_LIMIT_COARSE[AX1]=0.2	; 轴 1
$MA_STOP_LIMIT_COARSE[AX2]=0.2	; 轴 2
	; 精准停窗口：
$MA_STOP_LIMIT_FINE[AX1]=0.01	; 轴 1
$MA_STOP_LIMIT_FINE[AX1]=0.01	; 轴 2

⚠小心

CHANDATA-语句

在零件程序中，只可以将 CHANDATA 指令设置给执行 NC 程序的通道，也就是说，可以将该语句用来防止 NC 程序在非配置的通道上执行。

在故障时停止程序执行。

说明

在工作表中的 INI 文件不含 CHANDATA 指令。

保存初始化程序 (COMPLETE, INITIAL)

工作存储器的文件可以保护到一个外部 PC 中，并可以从那儿再次读入。

- 使用 COMPLETE 备份文件。
- 使用 INITIAL 通过所有范围生成一个 INI(_N_INITIAL_INI) 文件。

读入初始化程序

注意

如果读入名称“INITIAL_INI”的文件，则对所有文件中未提供的数据用标准数据进行初始化。 只有机床数据除外。 也提供**设置数据，刀具数据，NPV, GUD 值, ...** 与标准数据（一般情况下“零”)。

为了读入单独的机床数据，例如适用于文件 COMPLETE_TEA_INI。 在该文件中控制系统仅等待机床数据。 为此在这种情况下其他数据范围保持不变。

加载初始化程序

如果 INI 程序仅使用一个通道的数据，则它也可以作为零件程序选择并调用。 因此也就可以初始化程序控制的文件。

2.3 步进编辑器中的结构化指令（SEFORM）

功能

结构化指令 SEFORM 在步进编辑器（基于编辑器的程序支持）中处理，从而从中生成步进画面，用于 HMI－高级。 步进画面用于改善 NV 子程序可读性。

句法

SEFORM(<段名称>,<级面>,<图标>)

含义

SEFORM()	通过参数 <段名称>, <级面> 和 <图标> 进行结构化指令功能调用		
<段名称>	工作步骤名称		
	类型：	字符串	
<级面>	主级面或者子级面索引		
	类型：	INT	
	值：	0	主级面
		1, ..., <n>	子级面 1, ... , 子级面 <n>
<图标>	图标名称，显示用于该文件。		
	类型：	字符串	

说明

SEFORM 指令在步进编辑器中产生。

用参数<段名称>传递的字符串与主运行同步，存放在 BTSS 变量中（与 MSG 指令类似）。 在到下一次 SEFORM 指令改写之前，该信息保持不变。 使用复位键，和零件程序结束时删除该内容。

参数<级面> 和 <图标>在执行零件程序时由 NCK 检查，但是不继续处理。

文献

基于编辑器的程序支持其他信息参见：
操作手册 HMI 高级

保护区

3

3.1 保护区的确定 (CPROTDEF, NPROTDEF)

功能

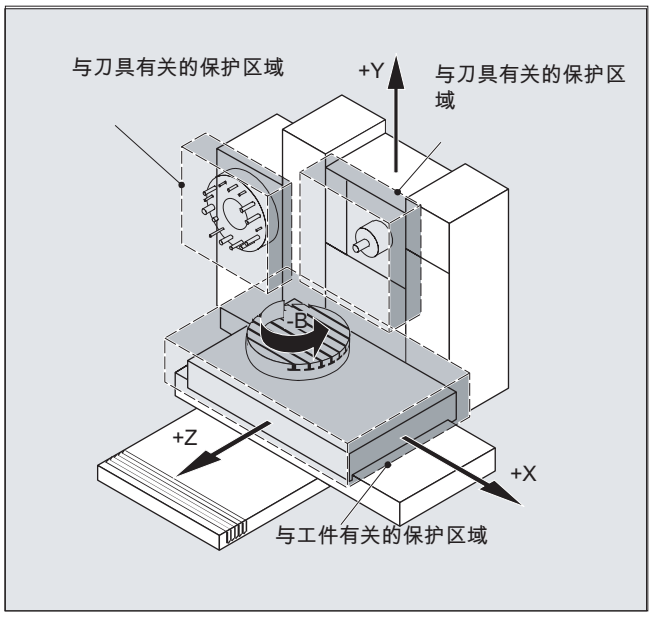
利用保护区，可以保护机床上各个不同的部件、夹具以及工件，防止误动作。

与刀具有关的 保护区：

用于属于刀具的零件（例如刀具，刀架）。

与工件有关的 保护区：

用于属于工件的零件（例如工件的零件，装夹台，夹爪，主轴卡盘，尾架）。



句法

```
DEF INT NOT_USED
G17/G18/G19
CPROTDEF/NPROTDEF(<n>,<t>,<applim>,<applus>,<appminus>)
G0/G1/... X/Y/Z...
```

```
...
EXECUTE (NOT_USED)
```

含义

DEF INT NOT_USED:	定义局部变量、INTEGER 数据类型 （参见"运动同步动作 (页 567)"章节）
G17/G18/G19：	在 CPROTDEF 或者 NPROTDEF 之前用 G17/G18/G19 选择所需平面，且不可在 EXECUTE 之前修改。在 CPROTDEF 或者 NPROTDEF 和 EXECUTE 之间，不允许编程应用。
CPROTDEF:	定义通道特有的保护区（仅用于 NCU 572/573)
NPROTDEF:	定义机床特有的保护区
G0/G1/... X/Y/Z... ...：	在所选平面中最多以 11 次运行设定保护区的轮廓。其中第一次运行为向轮廓的运动。轮廓左侧的区域作为保护区。 **提示：** 在 CPROTDEF 或者 NPROTDEF 和 EXECUTE 之间的运动不被执行，而是用于定义保护区。
EXECUTE:	结束定义
<n>：	定义的保护区序号
<t>:	保护区类型 TRUE: 与**刀具**有关的保护区 FALSE: 与**工件**有关的保护区
<applim>:	第 3 个维度的限制类型 0: 无限制 1: 正向限制 2: 负向限制 3: 正负方向的限制
<applus>:	第 3 个维度正向的限制值
<appminus>:	第 3 个维度负向的限制值
NOT_USED:	在有 EXECUTE 的保护区中故障变量无效

边界条件

在定义保护区时：

- 不允许激活铣刀半径补偿或者刀沿半径补偿。
- 不允许激活转换。
- 不允许激活框架。

也不允许编程回参考点（G74）、运动到固定点停止（G75）、程序段预处理停止或者程序结束。

其它信息

保护区的定义

以下部分属于保护区的定义：

- CPROTDEF 用于通道专用的保护区
- NPROTDEF 用于机床专用的保护区
- 保护区轮廓描述
- 使用 EXECUTE 结束定义

在 NC 零件程序中激活保护区时，可以相对地平移保护区基准点。

轮廓描述中的基准点

工件相关的保护区在基准坐标系中定义。

刀具相关的保护区以刀架基准点 F 为参考设定。

允许的轮廓单元

可用于定义保护区轮廓的轮廓单元：

- G0、G1 用于直线轮廓单元
- G2 用于顺时针圆弧段（仅用于工件相关的保护区）
- G3 用于逆时针圆弧段

说明

如果要求描述一个整圆作为保护区，则它必须分为两个半圆。 不允许使用顺序 G2、G3 或者 G3、G2。 有时必须要插进一个较短的 G1 程序段。

轮廓描述的最后一个点必须与第一个点重合。

外侧保护区

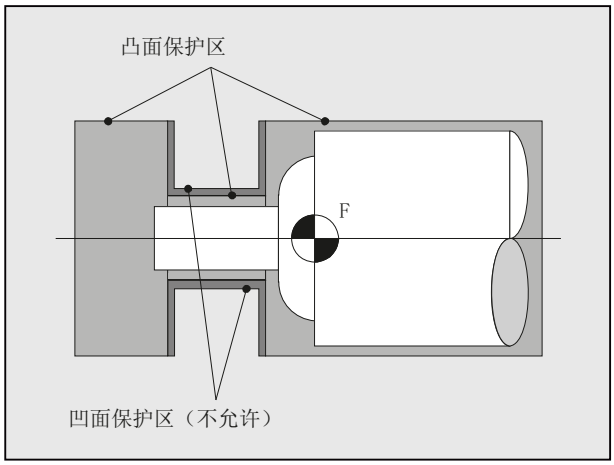
外侧保护区（仅当与工件有关的保护区才可以）应以逆时针方向定义。

旋转对称保护区

对于旋转对称的保护区（例如主轴卡盘），必须定义全部轮廓（不仅仅到旋转中心为止!）。

与刀具有关的保护区：

与刀具有关的保护区必须始终为凸面 。 如果希望有一个凹面保护区，则可以把它分成多个凸面保护区。



3.2 激活/取消激活保护区 (CPROT, NPROT)

功能

激活、预先激活之前定义好的保护区来监控碰撞，或者解除激活的保护区。

同时在一个通道中有效的保护区的最大数量通过机床数据确定。

如果没有刀具相关的保护区有效，则按照工件相关的保护区对刀具轨迹进行检查。

说明

如果没有工件相关的保护区有效，则不进行保护区监控。

句法

```
CPROT(<n>,<state>,<xMov>,<yMov>,<zMov>)
NPROT(<n>,<state>,<xMov>,<yMov>,<zMov>)
```

含义

CPROT:	调用通道专用保护区 (仅用于 NCU 572/573)
NPROT:	调用机床专用保护区
<n>:	保护区序号
<state>:	状态参数说明
	0: 取消激活保护区
	1: 预先激活保护区
	2: 激活保护区
	3: 预先激活保护区，有条件停止
<xMov>,<yMov>,<zMov>:	平移几何轴中已经定义的保护区

边界条件

刀具半径补偿激活时的保护区监控

在刀具半径补偿激活时，仅在刀具半径补偿的平面和保护区定义的平面相同时才能进行有效的保护区监控。

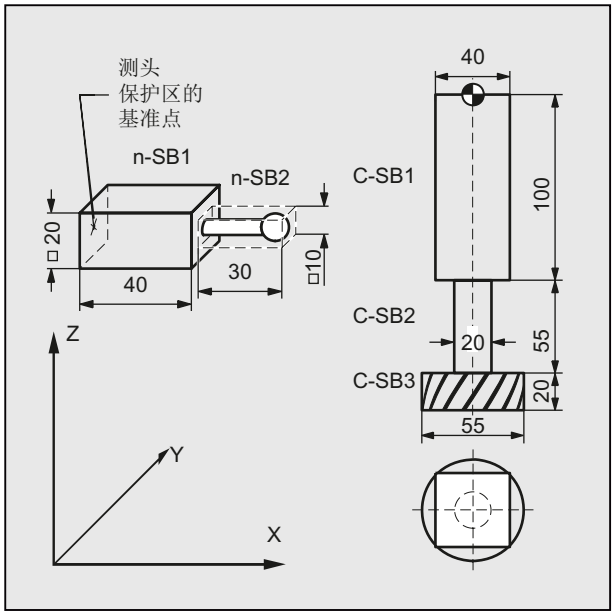
示例

对于铣床而言，应对铣刀与探头可能会有的碰撞进行监控。 探头的位置应在激活时通过位移来设定。 为此，定义以下的保护区：

- 探头支架 (n-SB1) 和探头自身 (n-SB2)各有一个机床专用的保护区、一个与工件有关的保护区。
- 铣刀刀夹(c-SB1)、铣刀柄(c-SB2)和铣刀自身(c-SB3)各有一个通道专用的保护区、一个和刀具相关的保护区。

所有保护区的定向均在 Z 方向中。

当激活时，探头的参考点位置应为 X = -120, Y = 60 和 Z = 80 。



程序代码	注释
DEF INT SCHUTZB	; 定义某个辅助变量
定义保护区 G17	; 设定方向
NPROTDEF(1,FALSE,3,10,-10)G01 X0 Y-10	; 保护区 n-SB1
X40	
Y10	
X0	
Y-10	
EXECUTE(SCHUTZB)	
NPROTDEF(2,FALSE,3,5,-5)	; 保护区 n-SB2
G01 X40 Y-5	
X70	
Y5	

程序代码	注释
X40	
Y-5	
EXECUTE(SCHUTZB)	
CPROTDEF(1,TRUE,3,0,-100)	; 保护区 c-SB1
G01 X-20 Y-20	
X20	
Y20	
X-20	
Y-20	
EXECUTE(SCHUTZB)	
CPROTDEF(2,TRUE,3,-100,-150)	; 保护区 c-SB2
G01 X0 Y-10	
G03 X0 Y10 J10	
X0 Y-10 J-10	
EXECUTE(SCHUTZB)	
CPROTDEF(3,TRUE,3,-150,-170)	; 保护区 c-SB3
G01 X0 Y-27,5	
G03 X0 Y27,5 J27,5	
X0 Y27,5 J-27,5	
EXECUTE(SCHUTZB)	
激活保护区:	
NPROT(1,2,-120,60,80)	; 激活带偏移的保护区 n-SB1
NPROT(2.2,-120,60,80)	; 激活带偏移的保护区 n-SB2
CPROT(1,2,0,0,0)	; 激活带偏移的保护区 c-SB1
CPROT(2,2,0,0,0)	; 激活带偏移的保护区 c-SB2
CPROT(3,2,0,0,0)	; 激活带偏移的保护区 c-SB3

其它信息

激活状态 (<state>)

- **<state>=2**

  在通常情况下，在零件程序中用 Status = 2 激活一个保护区。

  状态总是指通道专用的，机床相关的保护区也如此。

- **<state>=1**

  当打算通过 PLC 用户程序来使保护区通过 PLC 用户程序设置成有效时，则可通过 Status = 1 来进行所需的预先激活。

- **<state>=3**

  在使用有条件停止的预激活时，原则上不会在进入预激活的保护区之前停下。 当保护区设置为有效后才会停止。 当保护区只在特殊情况下设置为有效时，就可以实现不间断的加工。 需要注意的是，如果保护区在运行前才刚刚设置为有效，那么由于制动斜坡而有可能会驶入到保护区中。

  带有条件停止的预激活通过 Status = 3 进行设置。

- **<state>=0**

  通过 Status = 0 取消激活，即关闭保护区。不需要偏移。

在（预）激活时偏移保护区

可以用 1、2 或者 3 维尺寸偏移。 偏移的参数说明与以下相关：

- 工件专用的保护区中机床零点。
- 刀具专用的保护区中刀架基准点 F。

启动之后的状态

保护区可以在引导及回参考点之后就已经激活。 因此必须已将系统变量 $SN_PA_ACTIV_IMMED[<n>] 或者 $SC_PA_ACTIV_IMMED[<n>] 设置为 TRUE。 使用 Status = 2 来将其激活并且没有位移。

多次激活保护区

某个保护区同时也可以在多个通道中（例如两个相对滑板的顶尖套筒）。 只有当所有的几何轴都回参考点之后，才可以监控保护区。

这里：

- 在一个通道中，保护区不能同时多次激活不同的偏移。
- 机床相关的保护区必须在两个通道中指向相同的方向。

3.3 检查超出保护区的情况、工作区域限制和软件极限值(CALCPOSI)

功能

功能 CALCPOSI 用于检测几何轴从所给定的起始点起是否可以运行一段给定的位移，而不会损害轴界限（软件极限）、工作区域限制或者保护区。

如果无法以设定的行程进行运动，就恢复最大允许值。

功能 CALCPOSI 是一个预定义的子程序。该功能必须单独在某个程序段内。

句法

```
Status=CALCPOSI(_STARTPOS, _MOVDIST, _DLIMIT, _MAXDIST, _BASE_SYS,
_TESTLIM)
```

含义

状态	0: 功能正常， 所规定的行程可以完全执行完毕。 –: 在_DLIMIT 中至少有一个分量为负 –: 在一个转换计算中出现一个错误 如果给定的位移不能完全运行，则送回一个正的、十进制编码的数值： **个位（受损界限种类）：** 1: 软件极限限制运行行程。 2: 工作区域极限限制运行行程。 3: 保护区限制运行行程。 当同时有多个极限受到侵犯时（例如软件极限值和保护区），就在个位上显示可用来极为严格地限制规定运动行程的极限值。 **十位** 10: 起始值损害界限。 20: 规定的直线侵犯极限。 当终点自身没有侵犯极限，但是在从起始点到终点的行程中却有可能侵犯某个极限值时（例如穿过保护区，当进行非线性转换时 WKS 中的曲面软件极限值，例如传送），就会将该数值返回。 **百位** 100: 正极限值已被侵犯（仅当个位为 1 或者 2 时，即软件极限和工作范围极限） 100: 某个 NCK 保护区受到侵犯（仅当个位为 3 时）。 200: 负极限值已被侵犯（仅当个位为 1 或者 2 时，即软件极限和工作范围极限） 200: 某个通道转悠的保护区受到侵犯（仅当个位为 3 时）。

千位

1000:

用来与侵犯极限的轴的编号相乘的系数（仅当个位为 1 或者 2 时，即软件极限值和工作范围极限）。

这种轴从 1 开始计数，在软件极限受损时（个位＝1）与加工轴有关，在工作区域限制受损时（个位＝2）与几何轴有关。

1000:

系数，用此系数乘以受损的保护区的个数（仅当个位是 3 时）。

当多个保护区受到侵犯时，就在百位和千位中显示可极为严格地限制规定运动行程的保护区。

_STARTPOS

起始值，用于横坐标[0]、纵坐标[1]和工件坐标系中应用[2]

_MOVEDIST

增量式位移给定，用于横坐标[0]、纵坐标[1]和工件坐标系中应用[2]

_DLIMIT

[0] - [2]: 分配给几何轴的最小间距。

[3]: 最小间距，在非线性转换时分配此最小间距给一个直线加工轴，如果没有几何轴可以明确地分配时。

[4]: 最小间距，在非线性转换时此最小间距分配给一个旋转加工轴，如果没有几何轴可以明确地分配时。仅在特殊转换时，当软件极限应该受到监控时。

_MAXDIST

数组 [0] - [2] 用于返回值。所有三个几何轴中的增量行程，没有低于参与加工轴中的某个轴极限的规定最小间距。

如果运行位移没有限制，则该返回参数的内容等同于 _MOVDIST 的内容。

_BASE_SYS

FALSE 或者未指定参数：

在评价位置说明和长度说明时，使用组 13(G70, G71, G700, G710; 英制/公制)中的 G 代码。如果 G70 已激活且是公制基本系统（例如 G71 激活且为英寸），就在基本系统中发送 WKS 系统变量 $AA_IW[X] 和 $AA_MW[X]) 并且有可能必须经过换算以供功能 CALCPOSI 使用。

真：

在分析位置和长度说明时，始终使用控制系统的基本系统，与组 13 已激活的 G 的值无关。

_TESTLIM

待检测的界限（二进制编码）：

1: 监控软件极限

2: 监控工作区域限制

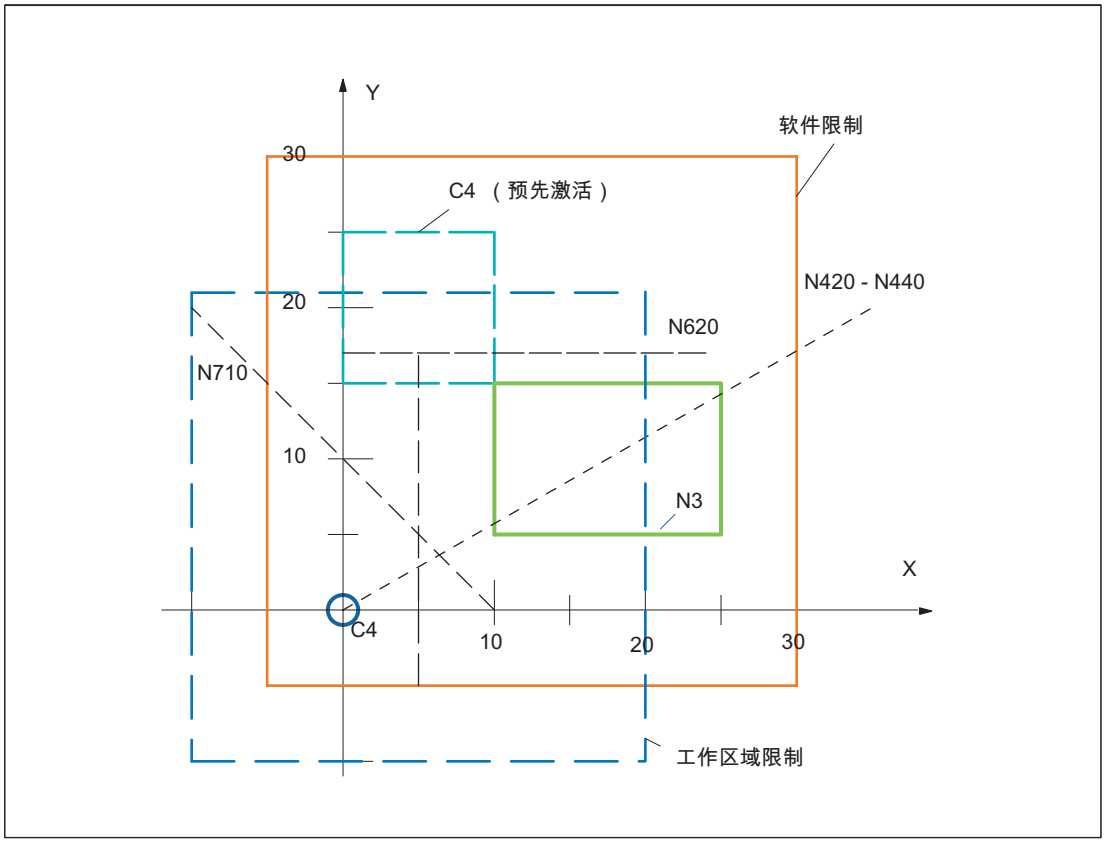
3: 监控激活的保护区

4: 监控预激活的保护区

通过数值的相加进行组合。缺省值：15; 检测所有限制。

示例

在举例中（见图示），在 X 中绘入了软件极限值和工作范围极限。此外，还定义了三个保护区，即两个通道特有的保护区 C2 和 C4 以及 NCK 保护区 N3。C2 是一个圆弧形状有效的、刀具相关的保护区，半径 2 毫米。C4 是一个正方形、已经预激活且与工件有关、侧面长度为 10mm 的保护区，N3 是一个矩形、已激活、侧面长度为 10mm 以及 15mm 的保护区。在下列 NC 中，首先如图所示定义保护区和工作范围极限，接着调用具有各种参数设置的功能 CALCPOSI。各次调用 CALCPOSI 的结构被汇总在示例结束处的表格中。



程序代码	注释
N10 DEF REAL KTAB[3]	
N20 def real _MOVDIST[3]	
N30 def real _DLIMIT[5]	
N40 def real _MAXDIST[3]	
N50 def int _SB	
N60 def int _STATUS	
N70 cprotdef(2, true, 0)	; 与刀具有关的保护区

程序代码	注释
N80 g17 g1 x-y0	
N90 g3 i2 x2	
N100 i-x-	
N110 execute(_SB)	
N120 cprotdef(4, false, 0)	; 工件相关的保护区
N130 g17 g1 x0 y15	
N140 x10	
N150 y25	
N160 x0	
N170 y15	
N180 execute(_SB)	
N190 nprotdef(3, false, 0)	; 与机床有关的保护区
N200 g17 g1 x10 y5	
N210 x25	
N220 y15	
N230 x10	
N240 y5	
N250 execute(_SB)	
N260 cprot(2,2,0, 0, 0)	; 激活或者预激活保护区
N270 cprot(4,1,0, 0, 0)	
N280 nprot(3,2,0, 0, 0)	
N290 g25 XX=-YY=-	; 定义工作区域界限
N300 g26 xx= 20 yy= 21	
N310 _STARTPOS[0] = 0.	
N320 _STARTPOS[1] = 0.	
N330 _STARTPOS[2] = 0.	
N340 _MOVDIST[0] = 35.	
N350 _MOVDIST[1] = 20.	
N360 _MOVDIST[2] = 0.	
N370 _DLIMIT[0] = 0.	
N380 _DLIMIT[1] = 0.	
N390 _DLIMIT[2] = 0.	
N400 _DLIMIT[3] = 0.	
N410 _DLIMIT[4] = 0.	
；不同的功能调用	; 其它起始点
N420 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST)	
N430 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST,,3)	
N440 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST,,1)	
N450 _STARTPOS[0] = 5.	; 其它目标
N460 _STARTPOS[1] = 17.	

程序代码	注释
N470 _STARTPOS[2] = 0.	
N480 _MOVDIST[0] = 0.	
N490 _MOVDIST[1] =-.	
N500 _MOVDIST[2] = 0.	
；不同的功能调用	
N510 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST,,14)	
N520 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST,, 6)	
N530 _DLIMIT[1] = 2.	
N540 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST,, 6)	
N550 _STARTPOS[0] = 27.	
N560 _STARTPOS[1] = 17.1	
N570 _STARTPOS[2] = 0.	
N580 _MOVDIST[0] =-.	
N590 _MOVDIST[1] = 0.	
N600 _MOVDIST[2] = 0.	
N610 _DLIMIT[3] = 2.	
N620 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST,, 12)	
N630 _STARTPOS[0] = 0.	
N640 _STARTPOS[1] = 0.	
N650 _STARTPOS[2] = 0.	
N660 _MOVDIST[0] = 0.	
N670 _MOVDIST[1] = 30.	
N680 _MOVDIST[2] = 0.	
N690 trans x10	
N700 arot z45	
N710 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST)	
N720 M30	

示例中的检查结果：

程序段号 N...	_STATUS	_MAXDIST [0] (= X)	_MAXDIST [1] (= Y)	注释
420	3123	8.040	4.594	保护区 SB N3 受到损害。
430	1122	20.000	11.429	没有 SB－监控，－工作区域界限受到损害。

440	1121	30.000	17.143	仅软极限监控仍有效。
510	4213	0.000	0.000	起始点损害 SB C4
520	0000	0.000	–.000	预激活的 SB C4 没有监控。给定的位移可以完全运行。
540	2222	0.000	–.000	因为 _DLIMIT[1]=2，所以通过工作范围极限来显示运动行程。
620	4223	–.000	0.000	由于 C2 和_DLIMIT[3]，到 C4 的间距共为 4 毫米。 0.1 毫米的 C2-C3 间距不会导致运行位移的限制。
710	1221	0.000	21.213	平移和旋转的框架有效。 _MOVDIST 中的允许运动行程在经过位移和旋转的坐标系中适用 (WKS).

特殊情况以及其它说明

所有行程说明均为半径尺寸，即使当某个平面轴带有激活的 G"DIAMON"时也是如此。如果某个参与轴的行程不能完全执行完毕，就会在返回值 _MAXDIST 中也相应减去其它轴的行程，使得出的终点在给定的轨迹上。

允许不给一个多个参与轴定义软件极限值以及工作范围极限或者保护区。仅当参与轴已经过找零时才监控所有极限值。如果参加插补的回转轴不是取模轴，则它们才会被监控。

与正常运动防止一样，对软件极限值和工作范围极限的监控取决于所激活的设置（用来选择软件极限值 1 以及软件极限值 2 的接口信号，GWALIMON/WALIMOF，用来个别激活工作范围极限和用来确定在监控工作范围极限时是否要考虑已激活刀具的半径的调整数据）。

对于某些运动转换而言（例如 TRANSMIT），从工件坐标系（WKS）中的位置不能唯一确定加工轴的位置（多义性）。在正常运动方式中，通常是从历史记录和某个加工轴的连续运动必须符合 WKS 重的某个连续运动的条件中得出单值性。因此在这一类情况下，在使用功能 CALCPOSI 监控软件极限时，当前的加工位置往往会引起多重含义。因此，有可能必须在 CALCPOSI 的前面编程一个 **STOPRE**，以便可以给函数提供有效的加工轴位置。

不能保证当所规定的运动行程上有某个运动时，在_DLIMIT[3] 中所规定的相对于保护区的距离能够处处得到遵守。所能保证的仅仅是：当以这段距离延长至 _MOVDIST 中所经过的终点时，保护区不会受到侵犯。 但是在其曲线过程中，该直线可以任意近地逼近一个保护区。

说明

有关工作区域限制的详细信息，请参见
/PG/ 编程手册基础部分，
有关软件限制的详细信息，参见
/FB1/ 功能手册基本功能：轴监控，保护区（A3）。

特殊的位移指令 4

4.1 逼近已经过编码处理的位置 (CAC, CIC, CDC, CACP, CACN)

功能

通过下列指令可以将直线轴和回转轴通过位置号码返回到机床数据表中设定的固定轴位置。 这种编程类型称作“返回到编码位置”。

句法

```
CAC(<n>)
CIC(<n>)
CACP(<n>)
CACN(<n>)
```

含义

CAC(<n>)	返回到位置号码 n 的编码位置
CIC(<n>)	编码位置，起自于当前的位置编号，向前 (+n) 或向后 (–n) 返回到 n 位置。
CDC(<n>)	以最短的行程返回到位置编号 n 的编码位置 （仅用于回转轴）
CACP(<n>)	按正方向返回到位置编号 n 的编码位置 （仅用于回转轴）
CACN(<n>)	按负方向返回到位置编号 n 的编码位置 （仅用于回转轴）
<n>	机床数据表中的位置编号 取值范围： 0, 1, ... (最大表位数 - 1)

示例： 返回到某个定位轴的编码位置

编程代码	注释
N10 FA[B]= 300	; 用于定位轴 B 的进给
N20 POS[B]=CAC(10)	; 返回到位置号码 10 的编码位置

编程代码	注释
N30 POS[B]=CIC(-4)	; 返回到"当前位置号码" －4 的编码位置

文献

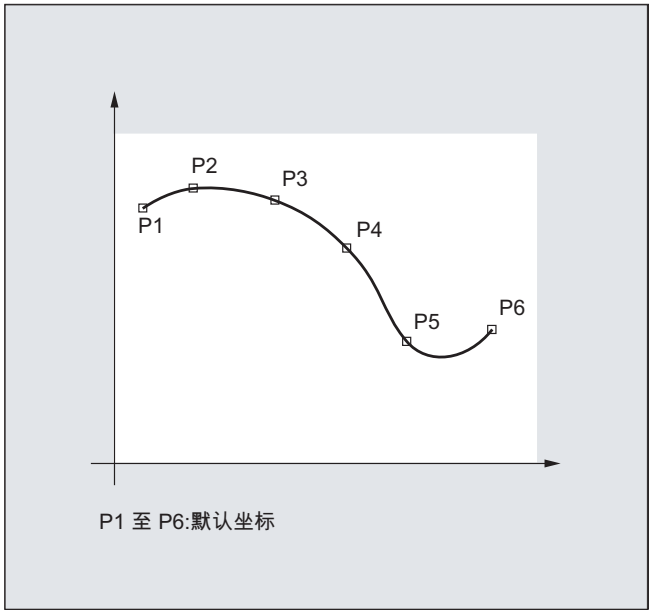
- 功能手册 扩展功能部分；分度轴（T1）
- 功能手册 同步动作

4.2 样条插补 (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL)

功能

无法精确分析描述工件上任意曲线轮廓。 因此这种类型的轮廓通过一个限定的支点数近似描述，例如表面数字化。 为了建立工件上的数字化表面，支点必须连接到一个轮廓描述。 这可以是样条插补。

样条定义一个由 2 阶或 3 阶多项式合并的曲线。 可定义样条支点上的特性，**取决于使用的样条类型** 。



SINUMERIK solution line 提供下列样条类型：

- A 样条
- B 样条
- C 样条

句法

通用：

```
ASPLINE X... Y... Z... A... B... C...
BSPLINE X... Y... Z... A... B... C...
CSPLINE X... Y... Z... A... B... C...
```

对于 B 样条可另外编程：

```
PW=<n>
SD=2
PL=<值>
```

对于 A 和 C 样条可另外编程：

```
BAUTO / BNAT / BTAN
```

```
EAUTO / ENAT / ETAN
```

含义

样条插补类型：

ASPLINE	用于打开 A 样条插补的指令
BSPLINE	用于打开 B 样条插补的指令
CSPLINE	用于打开 C 样条插补的指令
	指令 ASPLINE, BSPLINE 和 CSPLINE 是模态有效，并属于位移指令组。

支点或检查点：

X... Y... Z... A... B... C...	直角坐标的位置

点权重（仅 B 样条）：

PW	通过指令 PW 可对每个支点进行所谓的“点权重”编程。		
<n>	“点权重”		
	取值范围：	0 ≤ n ≤ 3	
	步进宽度：	0.0001	
	作用：	n > 1	曲线被检查点更为紧密地拉近。
		n < 1	曲线被检查点略微拉近。

样条阶（仅 B 样条）：

SD	正常情况下使用一个 3 级多项式。 通过编程 SD=2 也可以使用一个 2 阶多项式。

节点间距（仅 B 样条）：

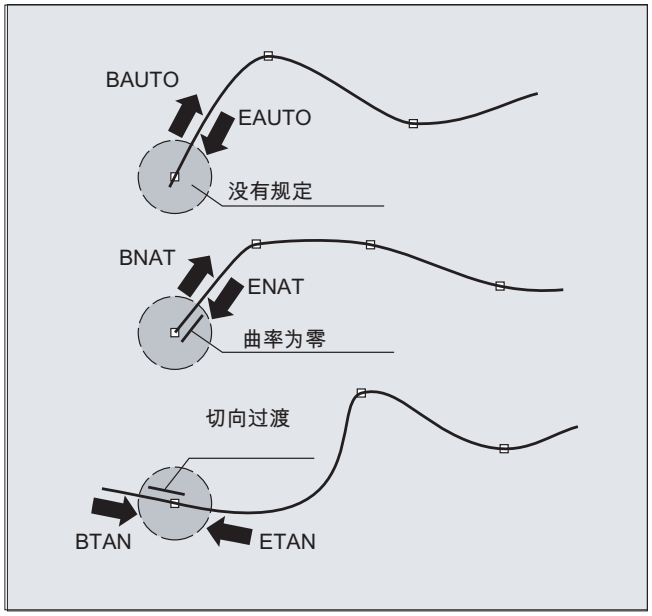
PL	节点间距适合于内部计算。 但是，控制系统也可以对规定的节点间距进行处理，在用指令 PL 规定作为所谓的参数间隔长度。
<值>	参数－间隔－长度 取值范围： 如位移尺寸

样条曲线开始处的过渡特性（仅 A 或 C 样条）：

BAUTO	没有给定过渡特性。 由第一个点的位置开始。
BNAT	曲率为零
BTAN	切线过渡到上一段（清除位置）

样条曲线末尾处的过渡特性（仅 A 或 C 样条）：

EAUTO	没有给定过渡特性。 从最后一个点的位置结束。
ENAT	曲率为零
ETAN	切线过渡到上一段（清除位置）



说明

可编程过渡特性不影响 B 样条。 在起始点和终点处 B 样条始终与控制多边形轮廓相切。

边界条件

- 可以使用刀具半径补偿。
- 以投影到平面上的方式来进行碰撞监控。

示例

示例 1： B 样条

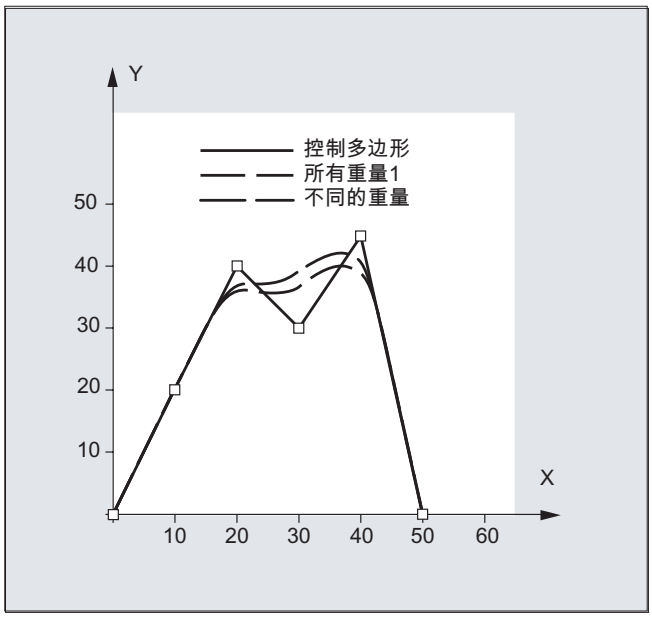
程序代码 1 (全部权重 1)

```
N10 G1 X0 Y0 F300 G64
N20 BSPLINE
N30 X10 Y20
N40 X20 Y40
N50 X30 Y30
N60 X40 Y45
N70 X50 Y0
```

程序代码 2 (不同的权重)

```
N10 G1 X0 Y0 F300 G64
N20 BSPLINE
N30 X10 Y20 PW=2
N40 X20 Y40
N50 X30 Y30 PW=0.5
N60 X40 Y45
N70 X50 Y0
```

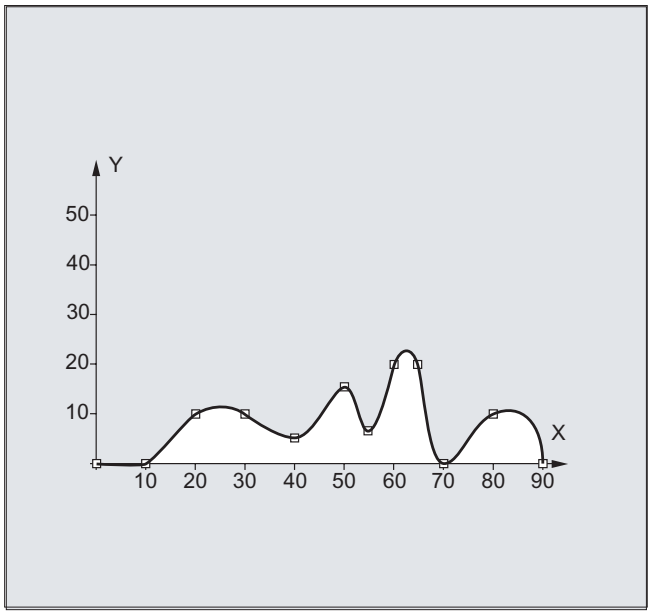
程序代码 3 (检查多项式)	注释
N10 G1 X0 Y0 F300 G64	
N20	; 已取消
N30 X10 Y20	
N40 X20 Y40	
N50 X30 Y30	
N60 X40 Y45	
N70 X50 Y0	



示例 2： C 样条，曲面开始和结束处为零

程序代码

```
N10 G1 X0 Y0 F300
N15 X10
N20 BNAT ENAT
N30 CSPLINE X20 Y10
N40 X30
N50 X40 Y5
N60 X50 Y15
N70 X55 Y7
N80 X60 Y20
N90 X65 Y20
N100 X70 Y0
N110 X80 Y10
N120 X90 Y0
N130 M30
```



示例 3： 样条插补 (A 样条) 和坐标转换 (ROT)

主程序：

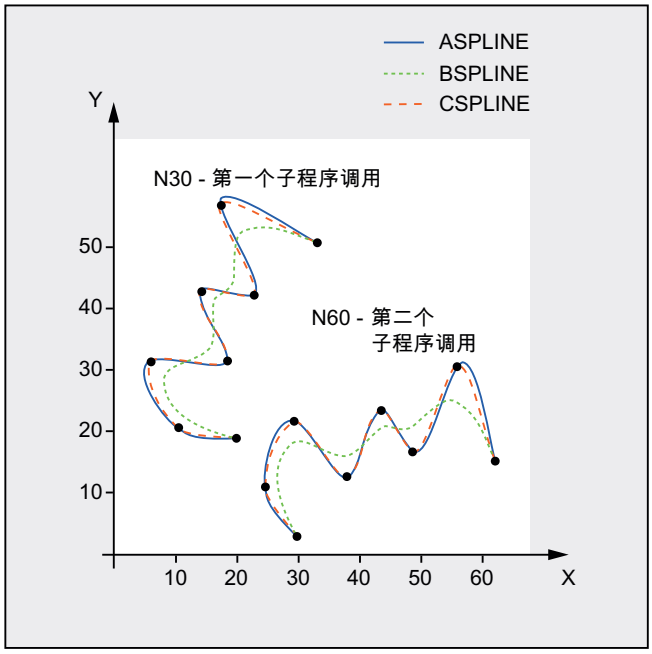
程序代码	注释
N10 G00 X20 Y18 F300 G64	; 返回起始点。
N20 ASPLINE	; 激活 A 样条插补类型。
N30 KONTUR	; 子程序首次调用。
N40 ROT Z-45	; 坐标转换： WKS 旋转 -45° 到 Z 轴。
N50 G00 X20 Y18	; 返回轮廓起始点。
N60 KONTUR	; 子程序第二次调用。
N70 M30	; 程序结束

子程序“轮廓” (包含支点坐标)：

程序代码
N10 X20 Y18
N20 X10 Y21
N30 X6 Y31
N40 X18 Y31
N50 X13 Y43
N60 X22 Y42
N70 X16 Y58

程序代码
N80 X33 Y51
N90 M1

在下列图形中除了由程序示例得出的样条曲线（ASPLINE），也包含激活 B 或 C 样条插补时得出的样条曲线（BSPLINE, CSPLINE）：



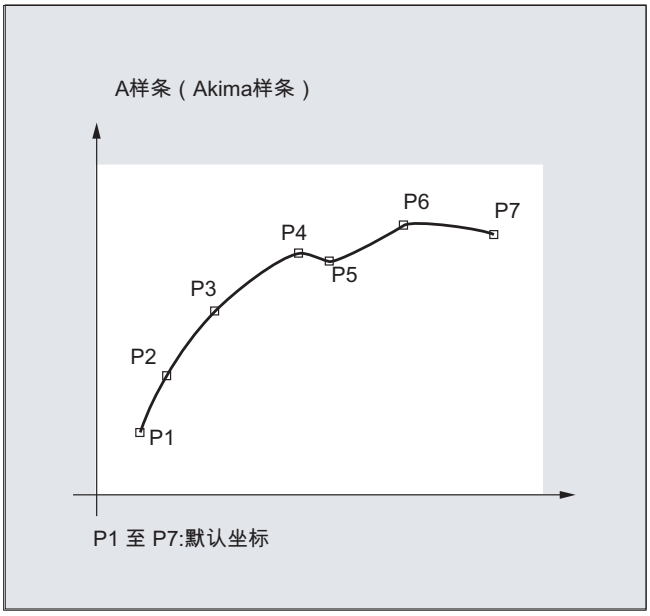
其它信息

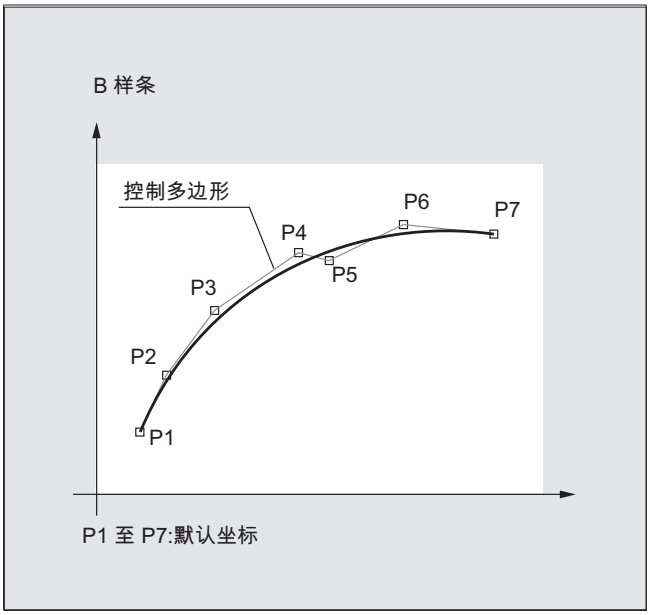
样条插补优点

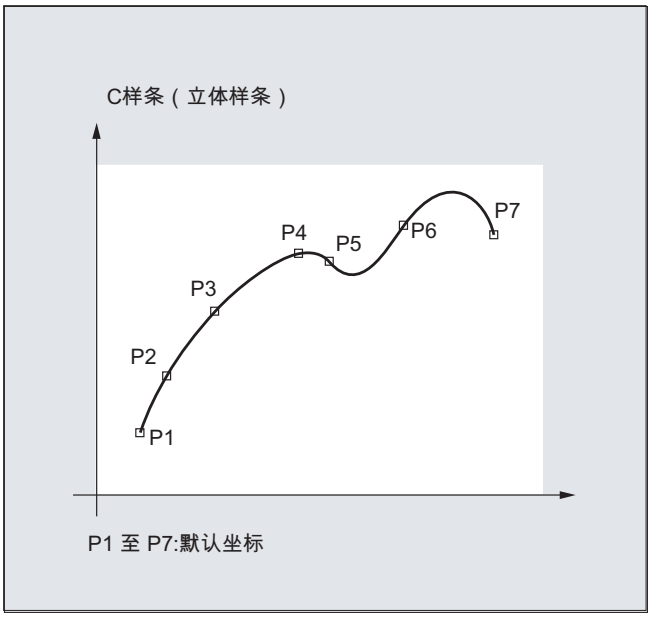
通过使用样条插补，与使用阶数组 G01 相比，有下列优点：

- 减少了用于描述轮廓所需零件程序段的数目
- 在零件程序段之间过渡时机械特性的曲线走向更为平滑

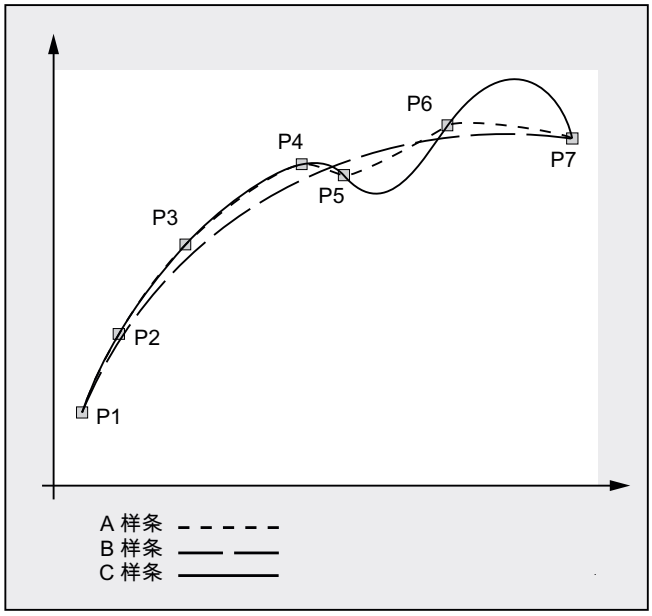
不同样条类型的特性和用途

样条类型	特性和用途
A 样条	 **特性：** • 通过规定的支点精确走向。 • 曲线走向为始终相切，但非曲面。 • 几乎不产生不期望的摆动。 • 支点改变影响范围是局部的，即支点的变化仅对 6 个以下相邻的支点起作用。 **应用：** A 样条首先适用于带有较大升幅改变的曲线插补（例如台阶式的曲线）。

样条类型	特性和用途
B 样条	 **特性：** • 走向不通过规定的支点，而是仅在其附近。 曲线通过支点拉近。通过支点权重及一个系数，可以另外影响曲线走向。 • 曲线走向为始终相切和曲面。 • 不产生不期望的摆动。 • 支点改变影响范围是局部的，即支点的变化仅对 6 个以下相邻的支点起作用。 **应用：** B 样条主要是考虑与 CAD 系统的接口。

样条类型	特性和用途
C 样条	 **特性：** • 通过规定的支点精确走向。 • 曲线走向为始终相切和曲面。 • 经常产生不期望的摆动，特别在带有较大升幅改变的位置。 • 支点改变影响范围是全局的，即一个支点的改变影响整个曲线走向。 **应用：** 当支点位于一个已知的曲线上时（圆，抛物线，双曲线），可以很好地采用 C 样条。

在相同的支点处对比三种类型的样条



样条程序段的最低数量

使用 G 指令 ASPLINE, BSPLINE 和 CSPLINE，使样条与程序段终点相联系。 对此必须同时计算一系列的程序段（终点）。 用于计算的缓冲器的大小一般情况下为 10 个程序段。 不是每条程序段信息都是一个样条终点。 然而，控制系统每 10 个程序段中必须有一定数量的样条终点程序段：

样条类型	样条程序段的最低数量
A 样条：	每 10 个程序段中必须至少有 **4** 个样条程序段。 注释程序段和参数计算不在此列。
B 样条：	每 10 个程序段中必须至少有 **6** 个样条程序段。 注释程序段和参数计算不在此列。
C 样条：	所需的样条程序段最低数量由下列求和得出： MD20160 $MC_CUBIC_SPLINE_BLOCKS 的值 + 1 在 MD20160 中设定通过样条段计算出的终点数量。 标准设置为 8。在标准情况下，每 10 个程序段中必须有 **9** 个样条程序段。

说明

如果低于允许值的范围，则会产生一个报警，同样当样条插补的轴当作定位轴编程时也会发出一个报警。

合并较短样条程序段

在样条插补中可能会存在较短的样条程序段，它们会降低轨迹速度。 通过功能“合并较短样条程序段”可以合并这些程序段，从而产生足够长的程序段并避免降低轨迹速度。

通过通道专用的机床数据：

MD20488 $MC_SPLINE_MODE （样条插补的设置）激活该功能。

文献:
功能手册 基本功能；轨迹控制运行，准停，预读(B1),
章节：合并较短样条程序段

4.3 样条组合(SPLINEPATH)

功能

使用指令 SPLINEPATH 选出样条组合中需要进行插补的轴。 样条插补中最多可以有 8 个轨迹轴。

说明

如果 SPLINEPATH 没有明确地编程，则通道中开始的 3 个轴作为样条组合运行。

句法

这需要在一个独立的程序段中确定样条组合：

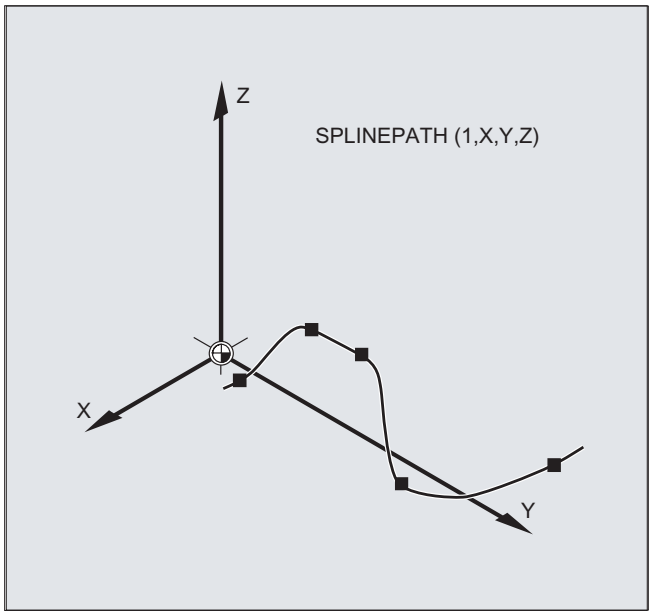
```
SPLINEPATH(n,X,Y,Z,…)
```

含义

SPLINEPATH	用于确定样条组合的指令
n	=1 (固定值)
X,Y,Z,…	样条组合中要插补的轨迹轴名称

示例： 带有三个轨迹轴的样条组合

程序代码	注释
N10 G1 X10 Y20 Z30 A40 B50 F350	
N11 SPLINEPATH(1,X,Y,Z)	; 样条组合
N13 CSPLINE BAUTO EAUTO X20 Y30 Z40 A50 B60	; C 样条
N14 X30 Y40 Z50 A60 B70	; 支点
…	
N100 G1 X… Y…	; 取消样条插补


Z
SPLINEPATH (1,X,Y,Z)
X
Y

4.4 NC 程序段压缩 (COMPON, COMPCURV, COMPCAD, COMPOF)

功能

CAD/CAM 系统通常提供线性程序段，它们按照设定的精度处理。 在轮廓比较复杂时这会导致数据量的大幅提高，并可能造成较短的路径段。 这种较短的路径段会限制加工速度。

使用压缩器功能，可以借助多项式程序段逼近由线形程序段设定的轮廓。 因此具有以下优点：

- 减少了用于描述工件轮廓所需零件程序段的数目
- 稳定的程序段过渡
- 提高了最大可行的路径速度

有下列压缩器功能可供使用：

- COMPON

  程序段过渡仅保持稳定的**速度**，轴的加速度可能会有跃变。

- COMPCURV

  程序段过渡保持稳定的**加速度**。 这样就可以保证程序段过渡时，所有轴的速度和加速度变化保持平稳。

- COMPCAD

  一种占用大量计算时间和内存空间的压缩器功能，优化了表面质量和速度。 只有当 CAD/CAM 的程序没有事先采取表面优化的措施时才使用 COMPCAD。

通过 COMPOF 退出压缩器功能。

句法

```
COMPON
COMPCURV
COMPCAD
COMPOF
```

含义

COMPON: 用于激活压缩器功能 COMPON 的指令。

有效性： 模态

COMPCURV:	用于激活压缩器功能 COMPCURV 的指令。
	有效性： 模态
COMPCAD:	用于激活压缩器功能 COMPCAD 的指令。
	有效性： 模态
COMPOF:	用于关闭当前激活的压缩器功能的指令。

说明

除此以外，改善表面质量还可以使用平滑功能 G642 和急动限制 SOFT。 这些指令应写在程序开始处。

边界条件

- 通常仅为线性程序段（G1）执行 NC 程序段压缩。
- 压缩功能只针对某个句法简单的程序段：

  N... G1X... Y... Z... F... ;注释

  所有其它的程序段按原样加工（没有压缩）。
- 带有扩展式地址如 C=100 或者 A=AC(100)的运动程序段也会被压缩。
- 位置值不必直接编程，也可以间接通过参数赋值，例如 X=R1*(R2+R3)。
- 如果提供选项“定向转换”，则也可以压缩 NC 程序段，在该 NC 程序段中刀具定向（也可能是刀具旋转）通过方向系数编程（参见“ 定向压缩 (页 368)”）。
- 可通过任意一个其它 NC 指令中断该压缩过程，例如辅助功能输出。

示例

示例 1： COMPON

程序代码	注释
N10 COMPON	; 激活压缩器功能 COMPON
N11 G1 X0.37 Y2.9 F600	; 终点前 G1 和进给。
N12 X16.87 Y-.698	
N13 X16.865 Y-.72	
N14 X16.91 Y-.799	
…	
N1037 COMPOF	; 关闭压缩功能。

程序代码	注释
…	

示例 2： COMPCAD

程序代码	注释
G00 X30 Y6 Z40	
G1 F10000 G642	; 激活平滑功能 G642。
SOFT	; 激活急动限制 SOFT。
COMPCAD	; 激活压缩器功能 COMPCAD。
STOPFIFO	
N24050 Z32.499	
N24051 X41.365 Z32.500	
N24052 X43.115 Z32.497	
N24053 X43.365 Z32.477	
N24054 X43.556 Z32.449	
N24055 X43.818 Z32.387	
N24056 X44.076 Z32.300	
…	
COMPOF	; 关闭压缩功能。
G00 Z50	
M30	

文献

功能手册 基本功能；连续路径运行，准停，预读(B1),
章节： “NC 程序段压缩器”

4.5 多项式插补 (POLY, POLYPATH)

功能

就本义来说，多项式插补（POLY）并不是一种样条插补。 首先它是用作编程外部生成的样条曲线的接口。 在此，样条区段可以直接编程。

使用此插补方式时，NC 不需计算多项式系数。 当系数直接来源于 CAD 系统或者后置处理程序时，此方式非常理想。

句法

3 阶多项式：

```
POLY PO[X]=(xe,a2,a3) PO[Y]=(ye,b2,b3) PO[Z]=(ze,c2,c3) PL=n
```

5 阶多项式和新多项式句法：

```
POLY X=PO(xe,a2,a3,a4,a5) Y=PO(ye,b2,b3,b4,b5)
Z=PO(ze,c2,c3,c4,c5) PL=n
POLYPATH("AXES","VECT")
```

说明

在一个 NC 程序段中编程的多项式系数和轴的总和不能超出每个程序段允许的最大轴数量。

含义

POLY：	程序段中写入 POLY，启用多项式插补。
POLYPATH：	可为两个轴组 AXIS 或 VECT 选择多项式插补
PO[轴名称/变量]：	终点和多项式系数
X, Y, Z：	轴名称
xe, ye, ze：	各个轴的终点位置说明；取值范围如同位移尺寸
a2, a3, a4, a5：	系数 $_2$, a_3, a_4, 和 a_5 使用其值写入；取值范围如同位移尺寸。 如果最后的系数值为零，则可以略去该值。
PL：	定义多项式的参数间隔长度（功能 f(p) 的定义范围）。 间隔从 0 开始，p 可以为 0 到 PL 之间的数值。 PL 的理论值范围： 0.0001 …99 999,9999 **提示：** PL 值用于它所在的程序段。 如果没有编程 PL，则 PL=1。

激活/取消多项式插补

在零件程序中通过 G 指令 POLY 激活多项式插补。

G 指令 POLY 和 G0、G1、G2、G3、ASPLINE、BSPLINE 和 CSPLINE 同属于 G 功能组 1。

仅使用其名称和终点编程的轴（例如 X10）以直线运行。 当一个 NC 程序段的所有轴都这样编程时，控制系统特性与采用 G1 时相同。

编程 G 功能组 1 中另一个指令（例如 G0、G1）会自动取消多项式插补。

多项式系数

PO 值(PO[]=) 或者 ...=PO(...) 用于设定轴的所有多项式系数。 根据多项式的阶数，可设定多个值并以逗号隔开。 在一个程序段中，不同的轴可以有不同的多项式阶数。

子程序 POLYPATH

使用 POLYPATH(...) 可选择性地为特定轴组使能多项式插补：

仅路径轴和辅助轴：	POLYPATH("AXES")
仅方向轴： （在包含方向转换的运行中）	POLYPATH ("VECT")

未使能的轴则以直线运行。

默认情况下会将两个轴组的多项式插补都使能。

编程时不写入参数 POLYPATH( ) 即可取消激活所有轴的多项式插补。

示例

程序代码	注释
N10 G1 X… Y… Z… F600	
N11 POLY PO[X]=(1,2.5,0.7) PO[Y]=(0.3,1,3.2) PL=1.5	; 启用多项式插补
N12 PO[X]=(0,2.5,1.7) PO[Y]=(2.3,1.7) PL=3	
...	
N20 M8 H126 …	
N25 X70 PO[Y]=(9.3,1,7.67) PL=5	; 轴的混合说明
N27 PO[X]=(10,2.5) PO[Y]=(2.3)	; 如果有编程 PL，则 PL=1
N30 G1 X… Y… Z.	; 关闭多项式插补
…	

示例： 新多项式句法

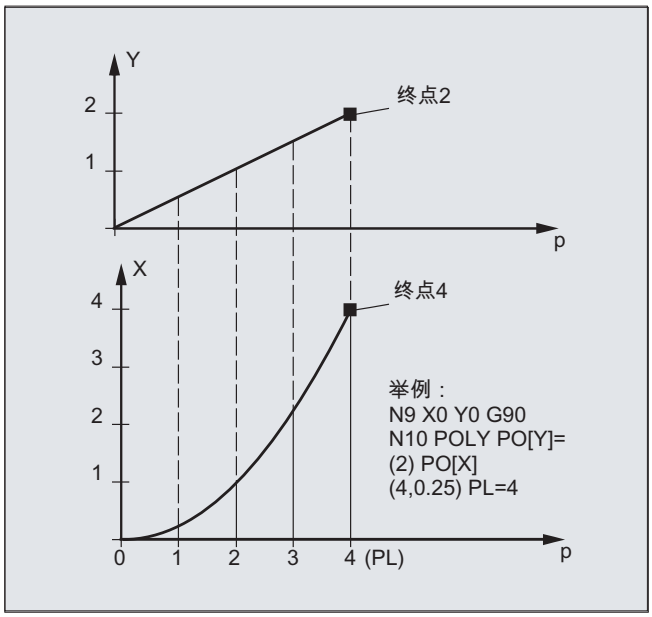
继续生效的多项式句法	新多项式句法
PO[轴名称]=(.. , ..)	轴名称=PO(.. , ..)
PO[PHI]=(.. , ..)	PHI=PO(.. , ..)
PO[PSI]=(.. , ..)	PSI=PO(.. , ..)
PO[THT]=(.. , ..)	THT=PO(.. , ..)
PO[]=(.. , ..)	PO(.. , ..)
PO[变量]=IC(.. , ..)	变量=PO IC(.. , ..)

示例： X/Y 平面中的曲线

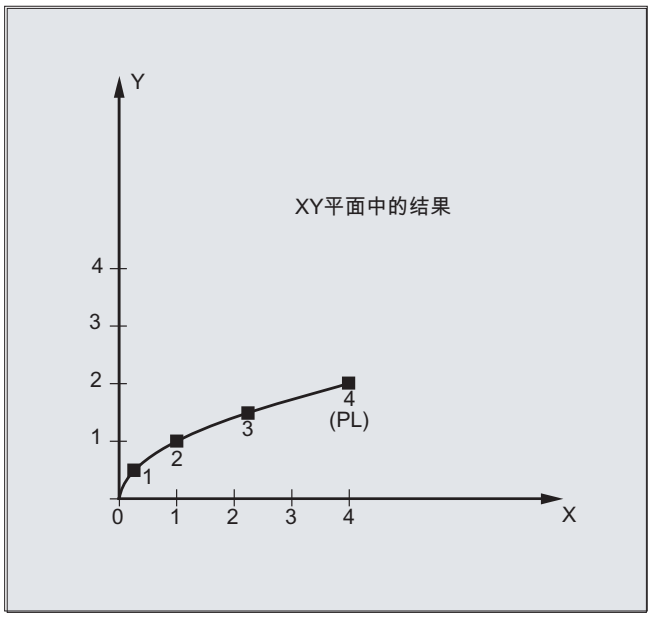
编程

程序代码
N9 X0 Y0 G90 F100
N10 POLY PO[Y]=(2) PO[X]=(4,0.25) PL=4

曲线 X(p) 和 Y(p)



XY 平面中的曲线



说明

多项式函数的一般形式为：

$$f(p)= a_0 + a_1p + a_2p^2 +. . . + a_np^n$$

其中： a_n: 固定系数

p： 参数

在控制系统中最多可编程 5 阶的多项式：

$$f(p)= a_0 + a_1p + a_2p^2 + a_3p^3 + a_4p^4 + a_5p^5$$

通过给系数设定具体的数值，可以产生各种曲线，如直线、抛物线和幂函数。

通过 $a_2 = a_3 = a_4 = a_5 = 0$ 生成直线：

$$f(p) = a_0 + a_1p$$

其中：

a_0：前一程序段结束时轴的位置

p = PL

$a_1 = (x_E - a_0 - a_2*p^2 - a_3*p^3) / p$

可以在**没有**使用 G 指令 POLY 激活多项式插补的情况下，对多项式进行编程。 在这种情况下不会插补编程的多项式，而是以直线（G1）逼近编程的轴终点。 只有在零件程序中以确定的方式激活多项式插补（POLY）后，才对编程的多项式进行插补。

特殊情况：除数多项式

对于几何轴，可使用 PO[]=(…) 不设定轴名称来编程一个共同的除数多项式，也就是说，将几何轴的运动作为两个多项式的商进行插补。

这样可精确描述例如圆锥面（圆，椭圆，抛物线，双曲线）等表面。

示例：

程序代码	注释
POLY G90 X10 Y0 F100	; 几何轴直线运行到位置 X10, Y0。
PO[X]=(0,-10) PO[Y]=(10) PO[]=(2,1)	; 几何轴沿四分圆运行到 X0 Y10。

除数多项式的固定系数（a_0）总是取 1。 编程的终点与 G90 或者 G91 无关。

X(p) 和 Y(p) 通过编程的值计算得出：

$$X(p) = (10 - 10 * p^2) / (1 + p^2)$$

$$Y(p) = 20 * p / (1 + p^2)$$

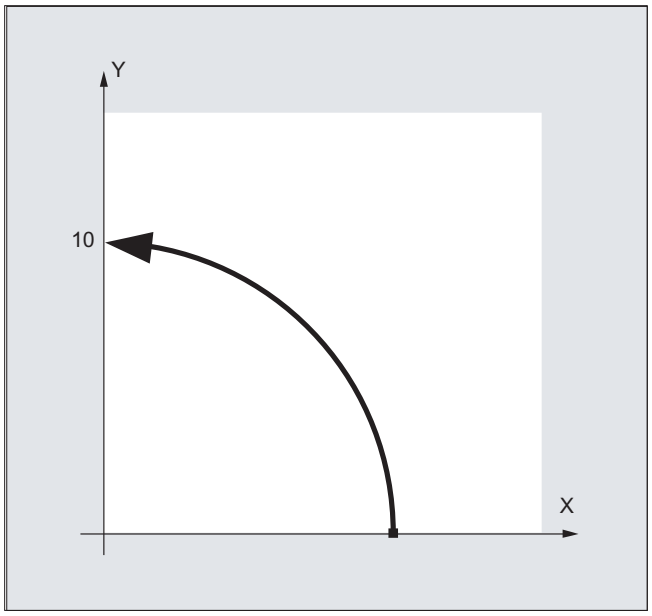
其中 $0 \le p \le 1$

根据已编程的起始点、终点、系数 a_2 和 PL=1，得出下列中间结果：

分子 (X) = $10 + 0 * p - 10 * p^2$

分子 (Y) = $0 + 20 * p + 0 * p^2$

分母＝ $1 + p^2$



多项式插补激活时，区间 [0,PL] 内以零编程的除数多项式会被拒绝，并输出报警。 除数多项式不会对辅助轴的运动产生影响。

说明

在多项式插补中可以通过 G41, G42 启用刀具半径补偿，使用方法同直线插补或者圆弧插补。

4.6 可设置的轨迹基准 (SPATH, UPATH)

功能

在多项式插补过程中，对于决定速度的 FGROUP 轴和其余路径轴之间的关系，用户可能会有两种不同的需求： 后者应与 FGROUP 轴的路径位移 S 同步，或者与其曲线参数 U 同步。

两种路径插补方式可以在不同的场合使用，并且可通过 G 代码组 45 中两个模态有效的指令 SPATH 和 UPATH 进行激活/切换。

句法

```
SPATH
UPATH
```

含义

SPATH:	FGROUP 轴的路径基准为弧长。
UPATH:	FGROUP 轴的路径基准为曲线参数。

说明

UPATH 和 SPATH 也用来确定 F 字多项式 (FPOLY, FCUB, FLIN) 与轨迹运动之间的关系。

边界条件

所设定的路径基准在以下情况下没有意义：

- 在直线插补和圆弧插补中
- 在螺纹程序段中
- 当所有的路径轴均包含在 FGROUP 中时。

示例

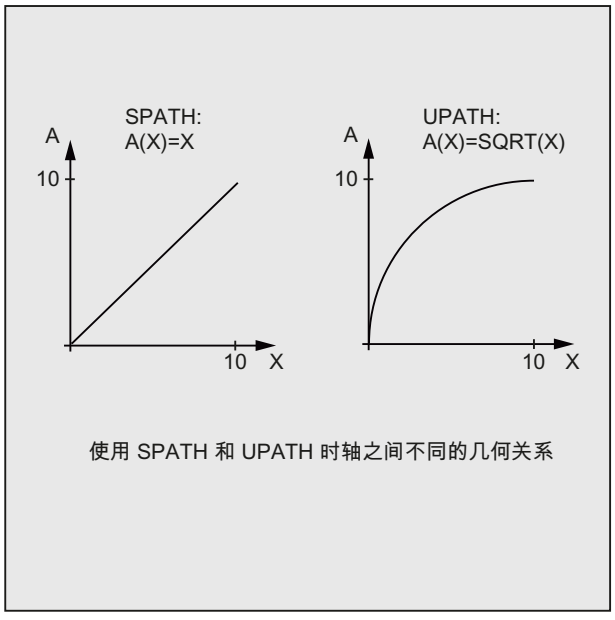
示例 1：

在下列示例中，使用 G643 对边长为 20mm 的正方形进行修光。 此时通过轴专用机床数据 MD33100 $MA_COMPRESS_POS_TOL[<n>] 为每根轴定义与精确轮廓的最大误差。

程序代码	注释
N10 G1 X… Y… Z… F500	
N20 G643	; G643，程序段内部平滑
N30 XO Y0	
N40 X20 Y0	; 边长（mm），用于轴
N50 X20 Y20	
N60 X0 Y20	
N70 X0 Y0	
N100 M30	

示例 2：

下列示例用来阐明两种运动控制之间的区别。 这两种情况下都保持缺省设置 FGROUP(X,Y,Z)。



使用 SPATH 和 UPATH 时轴之间不同的几何关系

程序代码
N10 G1 X0 A0 F1000 SPATH
N20 POLY PO[X]=(10,10) A10

或者：

程序代码
N10 G1 X0 F1000 UPATH

程序代码

```
N20 POLY PO[X]=(10,10) A10
```

在程序段 N20 中，FGROUP 轴的位移 S 与曲线参数 U 的平方有关。 因此根据激活的是 SPATH 还是 UPATH，沿着 X 轴的位移会得到同步轴 A 的不同位置。

其它信息

在多项式插补[其中包含狭义的多项式插补（POLY），所有样条插补（ASPLINE、BSPLINE、CSPLINE）和带压缩器功能 COMPON、 COMPCURV 的线性插补]，所有路径轴的位置 i 通过多项式 pi(U) 设定。 此时，曲线参数 U 在一个 NC 程序段之内从 0 到 1，即经过定标。

通过语言指令 FGROUP 可以在路径轴范围内选择某根轴，编写的路径进给率将针对该轴生效。 然而以固定的速度在该轴的位移 S 中进行插补，通常表示在多项式插补时曲线参数 U 的变化不稳定。

复位时的控制特性和机床数据/可选数据

在复位后，通过 MD20150 $MC_GCODE_RESET_VALUES[44] 确定的 G 代码生效（G 代码组 45）。 为了保持与当前设备的兼容性，此处 SPATH 为缺省值。

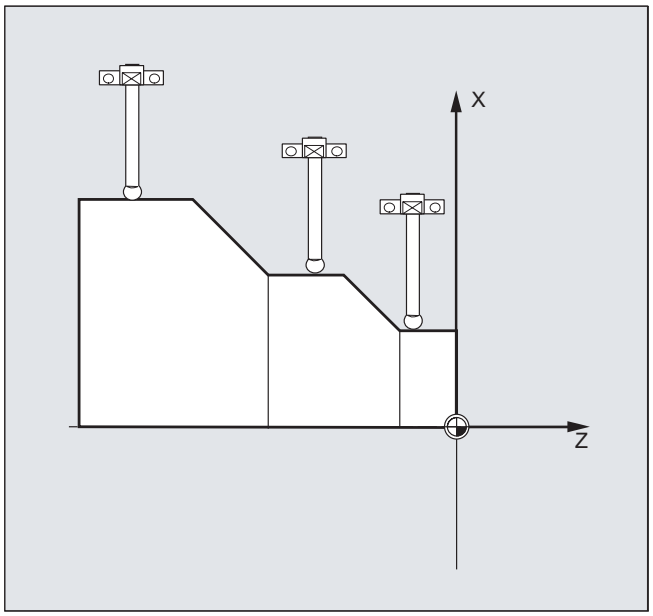
平滑方式的缺省值通过 MD20150 $MC_GCODE_RESET_VALUES[9] 定义（G 代码组 10）。

轴专用机床数据 MD33100 $MA_COMPRESS_POS_TOL[<n>] 具有另一层含义：它包含了压缩器功能的公差，以及 G642 平滑功能的公差。

4.7 用接触式探头测量 (MEAS, MEAW)

功能

通过功能“采用触发式探头测量”可以使轴移动到工件上的实际位置；在测量探头发出脉冲沿时，它可以测定所有测量程序段中写入的轴的位置，并将每个轴的位置写入到存储单元中。



编程测量程序段

在编程该功能，可以使用下列两个指令：

- MEAS

  通过指令 MEAS 可以删除实际位置和给定位置之间的剩余行程。

- MEAW

  而指令 MEAW 适用于一些测量任务，其中在任何情况下都必须移动轴到编程的位置。

MEAS 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 (G0, G1, ...) 以及轴的数量应与各自的测量任务相匹配。

读取测量结果

测量探头获取的轴的测量结果包含在以下变量中：

- $AA_MM[<轴>]

  机床坐标系中的测量结果

- $AA_MW[<轴>]

  工件坐标系中的测量结果

读取这些变量时不会在内部生成预处理停止。

说明

必须在 NC 程序中用 STOPRE 在适当的位置上编辑一个预处理停止。 否则会读入错误的值。

句法

```
MEAS=<TE> G... X... Y... Z...
MEAW=<TE> G... X... Y... Z...
```

含义

MEAS	指令：测量**带**剩余行程删除
	有效性：　逐段式
MEAW	指令：测量，**不带**剩余行程删除
	有效性：　逐段式
<TE>	触发测量的触发事件
	类型：　INT
	取值范围：　-2, -1, 1, 2
	提示： 最多可使用 2 个测量探头，视扩建阶段而定。
	含义：
	(+)1　测量探头 1（测量输入 1 上）的上升沿
	-1　测量探头 1（测量输入 1 上）的下降沿
	(+)2　测量探头 2（测量输入 2 上）的上升沿
	-2　测量探头 2（测量输入 2 上）的下降沿
	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 G3
X... Y... Z...	以直角坐标给定的终点

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量探头和直线插补的测量程序段。 自动产生预处理停止。
...	

其它信息

测量任务状态

如果需要在程序中分析测量探头是否已工作，可以查询状态变量 $AC_MEA[n] (n= 测量探头的编号)：

值	含义
0	测量任务未履行
1	测量任务已顺利结束（测量探头已工作）

说明

当探头在程序中偏转时，就将变量置为 1。 当某个测量程序段开始执行时，自动将变量设定成探头的初始状态。

记录测量值

采集程序段所有运动过的轨迹轴和定位轴的位置（轴上的最大数量要视控制系统配置而定）。 如果编程 MEAS，在测量探头触发之后按照定义使运动停止。

说明

如果在某个测量程序段中已经编程了某个几何轴，就保存所有当前几何轴的测量值。

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。

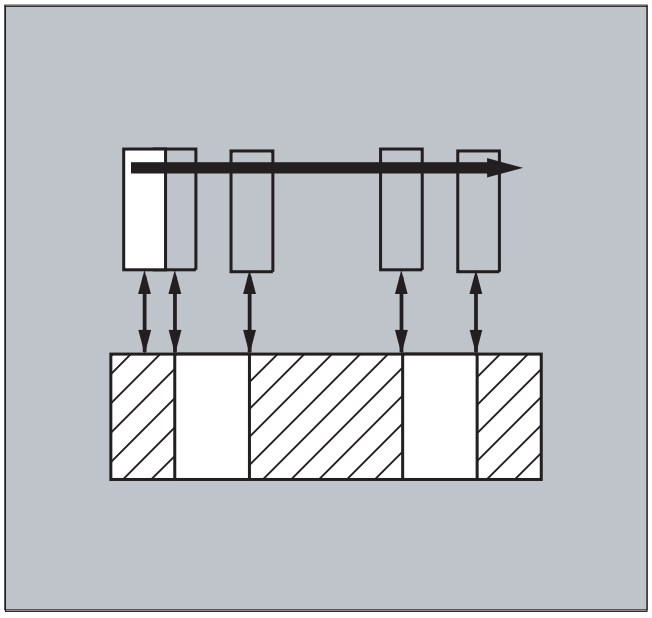
4.8 扩展测量函数 (MEASA, MEAWA, MEAC) (选项)

功能

如果是轴向测量，可以使用多个探头和多个测量系统。

通过指令 MEASA 或 MEAWA 可以在每次测量时为相应已编程的轴最多采集四个测量值，并且针对触发事件将其保存在系统变量中。

可以使用 MEAC 执行连续的测量任务。 在这种情况下，将测量结果保存在 FIFO 变量中。即使是 MEAC，每次测量最多也只能有四个测量值。



读取测量结果

测量结果包含在以下变量中：

- $AA_MM1...4[<轴>]

  机床坐标系中的测量结果

- $AA_MW1...4[<轴>]

  工件坐标系中的测量结果

句法

```
MEASA[<轴>]=(<模式>,<TE1>,...,<TE4>)
MEAWA[<轴>]=(<模式>,<TE1>,...,<TE4>)
MEAC[<轴>]=(<模式>,<测量存储器>,<TE1>,...,<TE4>)
```

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，就会发出出错信息。

含义

MEASA	指令： 轴测量，带剩余行程删除
	有效性： 逐段式
MEAWA	指令：轴测量，没有剩余行程删除
	有效性： 逐段式
MEAC	指令： 轴持续测量，没有剩余行程删除
	有效性： 逐段式
<轴>	名称，用于测量所使用的通道轴
<模式>	指定运行模式（测量模式和测量系统）的两位数

测量模式(十进制个位)：

0 中断测量任务。

1 4 个以下可以同时激活的触发事件。

2 4 个以下可以依次激活的触发事件。

3 4 个以下可以依次激活的触发事件，然而，在启动时没有对触发事件 1 的监控（报警 21700/21703 被抑制）。

提示：
该模式不适用于 MEAC。

测量系统 (十进制十位)：

0（或者没有说明）	已激活的测量系统
1	测量系统 1
2	测量系统 2
3	两个测量系统

<TE>	触发测量的触发事件
	类型： INT
	取值范围： -2, -1, 1, 2
	含义：

(+)1	测量探头 1 的上升沿
-1	测量探头 1 的下降沿
(+)2	测量探头 2 的上升沿
-2	测量探头 2 的下降沿
<测量存储器>	FIFO 号（循环存储器）

示例

示例 1： 在模式 1 中进行轴的测量，带剩余行程删除（按时间顺序分析）

a) 有 1 个测量系统

程序代码	注释
...	
N100 MEASA[X]=(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，测量探头 1 的上升沿/下降沿在 X = 100 后面的运动行程上。
N110 STOPRE	; 预处理停止
N120 IF $AC_MEA[1]==FALSE GOTOF ENDE	; 检查测量结果。
N130 R10=$AA_MM1[X]	; 保存属于第一个已编程触发事件 (上升沿)的测量值。
N140 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。
N150 ENDE:	

b) 有 2 个测量系统

程序代码	注释
...	
N200 MEASA[X]=(31,1,-1) G01 X100 F100	; 使用两个测量系统在模式 1 中测量。 等待测量信号，测量探头 1 的上升沿/下降沿在 X = 100 后面的运动行程上。
N210 STOPRE	; 预处理停止
N220 IF $AC_MEA[1]==FALSE GOTOF ENDE	; 检查测量结果。
N230 R10=$AA_MM1[X]	; 在出现上升沿时保存测量系统 1 的测量值。
N240 R11=$AA_MM2[X]	; 在出现上升沿时保存测量系统 2 的测量值。
N250 R12=$AA_MM3[X]	; 在出现下降沿时保存测量系统 1 的测量值。
N260 R13=$AA_MM4[X]	; 在出现下降沿时保存测量系统 2 的测量值。
N270 ENDE:	

示例 2： 在模式 2 中进行轴的测量，带剩余行程删除（按编程的顺序分析）

程序代码	注释
...	
N100 MEASA[X]=(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，顺序为测量探头 1 的上升沿，测量探头 1 的下降沿；测量探头 2 的上升沿，测量探头 2 的下降沿。
N110 STOPRE	; 预处理停止
N120 IF $AC_MEA[1]==FALSE GOTOF MESSTASTER2	; 使用测量探头 1 检查测量结果。
N130 R10=$AA_MM1[X]	; 保存属于第一个已编程触发事件 (测量探头 1 上升沿) 的测量值。
N140 R11=$AA_MM2[X]	; 保存属于第二个已编程触发事件 (测量探头 1 上升沿) 的测量值。
N150 MESSTASTER2:	
N160 IF $AC_MEA[2]==FALSE GOTOF ENDE	; 使用测量探头 2 检查测量结果。
N170 R12=$AA_MM3[X]	; 保存属于第三个已编程触发事件 (测量探头 2 上升沿) 的测量值。
N180 R13=$AA_MM4[X]	; 保存属于第四个已编程触发事件 (测量探头 2 上升沿) 的测量值。
N190 ENDE:	

示例 3： 在模式 1 中进行轴的持续测量（按时间顺序分析）

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) 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 $AC_FIFO1 中，等待测量探头 1 在 X = 1000 之后的运动行程上有下降沿的测量信号。
N135 STOPRE	
N140 MEAC[X]=(0)	; 在到达轴位置之后中断测量。
N150 R1=$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 $AC_FIFO1[4]>=10 DO MEAC[x]=(0) DELDTG(x)	; 删除剩余行程。
N20 MEAC[x]=(1,1,1,-1) G01 X100 F500	
N30 MEAC[X]=(0)	
N40 R1=$AC_FIFO1[4]	; 测量值的数量。
...	

其它信息

测量任务

可以在零件程序中或者从某个同步动作中（参见"运动同步"一章）编程一个测量任务。 某一时刻和同一时刻每个轴只能有一个测量任务激活。

说明

进给应和相应的测量问题适配。

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。

触发事件

一个触发事件由探头的编号和测量信号的触发条件（上升或者下降沿）组成。

每次测量时，可以分别处理 4 个以下的已响应探头的触发事件，即最多两个探头，每个探头分别有两个测量脉冲沿。 处理的顺序和触发事件的最大数量与所选的模式有关。

说明

针对测量模式 1： 在同一个测量任务中只能编程一个触发事件！

运行模式

使用运行模式的第一个数字（十位数）来选择所需的测量系统。 如果只有一个测量系统存在，但是仍然编程了第二个测量系统，就自动使用存在的测量系统。

使用第二个数字（个位数）可以选择所需的测量模式。 以此可以根据相应的控制方法调节测量过程：

- **模式 1:**

  按照触发事件出现的时间顺序对其进行分析。 在这种模态中使用六轴模块时仅可编程一个触发事件，或者在参数说明多个触发事件时自动切换到第 2 个模态（没有提示信息）。

- **模式 2:**

  按照编程顺序对触发事件进行分析。

- **模式 3:**

  按照编程顺序对触发事件进行分析，但是不在 START 时监控触发事件 1。

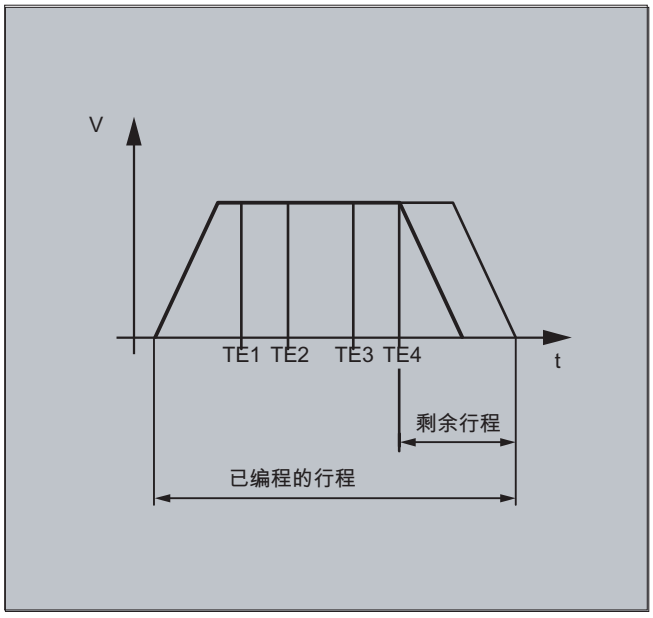
说明

如果使用 2 个测量系统，就只可编程两个触发事件。

有和没有剩余行程删除的测量

在编程 MEASA 时，只有在采集所有所要的测量值之后才会执行剩余行程删除。

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 不可以在同步动作中编程。取而代之的是可以将 MEAWA 加上剩余行程删除作为同步动作编程。

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。

MEASA, MEAWA 的测量结果

测量结果可在下列变量项下使用：

- 在机床坐标系中：

$AA_MM1[<轴>]	当出现触发事件 1 时已编程测量系统的测量值
...	...
$AA_MM4[<轴>]	当出现触发事件 4 时已编程测量系统的测量值

- 在工件坐标系中：

$AA_WM1[<轴>]	当出现触发事件 1 时已编程测量系统的测量值
...	...
$AA_WM4[<轴>]	当出现触发事件 4 时已编程测量系统的测量值

说明

读取这些变量时不会在内部生成预处理停止。 必须在 NC 程序中用 STOPRE 在适当的位置上编辑一个预处理停止。 否则会读入错误的值。

几何轴/坐标转换

如果要开始对某个几何轴进行轴向测量，就必须对所有剩余几何轴的相同测量任务进行显式编程。 这同样适用于进行转换的轴。

示例：
```
N10 MEASA[Z]=(1,1) MEASA[Y]=(1,1) MEASA[X]=(1,1) G0 Z100
```

或者
```
N10 MEASA[Z]=(1,1) POS[Z]=100
```

有 2 个测量系统的测量任务

当使用两个测量系统执行某个测量任务时，相应轴的两个测量系统可能有两个触发事件，应采集其中可能有的每个事件。 预定变量的配置规定为：

$AA_MM1[<轴>]	或者	$AA_MW1[<轴>]	当出现触发事件 1 时测量系统 1 的测量值
$AA_MM2[<轴>]	或者	$AA_MW2[<轴>]	当出现触发事件 1 时测量系统 2 的测量值

$AA_MM3[<轴>]	或者	$AA_MW3[<轴>]	当出现触发事件 2 时测量系统 1 的测量值
$AA_MM4[<轴>]	或者	$AA_MW4[<轴>]	当出现触发事件 2 时测量系统 2 的测量值

探头状态

测量探头状态包含在以下系统变量中：

$A_PROBE[<n>]

<n>=测量探头

值	含义
1	测量探头偏转
0	测量探头未偏转

当 MEASA, MEAWA 时的探头状态

如果有必要在程序中进行分析，可以通过“$AC_MEA[<n>]”（n = 探头的编号）来查询测量任务状态。 只要在某个程序段中已编程的探头<n>的所有触发事件已经出现，该变量就会给出值 1。其它情况下值为 0。

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，应查询新的 PLC 状态信号 DB31, … DBX62.3 或者等效变量 $AA_MEAACT[<轴>]。

含义：

$AA_MEAACT==1: 测量有效

$AA_MEAACT==0: 测量未激活

连续测量(MEAC)

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（循环存储器）。 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（可通过 MD 设置）。

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，参见“运动同步”一章。

说明

FIFO 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-存储器规定的最大数时，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。

已识别的错误编程

识别出下面的出错编程，并且显示一个出错：

- MEASA/MEAWA 与 MEAS/MEAW 被编写在一个程序段中

  示例：

  ```
  N01 MEAS=1 MEASA[X]=(1,1) G01 F100 POS[X]=100
  ```

- MEASA/MEAWA 参数个数 <2 或者 >5

  示例：

  ```
  N01 MEAWA[X]=(1) G01 F100 POS[X]=100
  ```

- MEASA/MEAWA 触发事件不等于 1/ -1/ 2/ -2

  示例：

  ```
  N01 MEASA[B]=(1,1,3) B100
  ```

- MEASA/MEAWA 模式错误

  示例：

  ```
  N01 MEAWA[B]=(4,1) B100
  ```

- MEASA/MEAWA 重复编程的触发事件

  示例：

  ```
  N01 MEASA[B]=(1,1,-1,2,-1) B100
  ```

- MEASA/MEAWA 和错误的几何轴

  示例：

  ```
  N01 MEASA[X]=(1,1) MEASA[Y]=(1,1) G01 X50 Y50 Z50 F100 ;几何轴 X/Y/Z
  ```

- 几何轴有不统一的测量任务

  示例：

  ```
  N01 MEASA[X]=(1,1) MEASA[Y]=(1,1) MEASA[Z]=(1,1,2) G01 X50 Y50 Z50 F100
  ```

4.9 适用于 OEM 用户的专用函数 (OEMIPO1, OEMIPO2, G810 bis G829)

功能

OEM 地址

OEM 用户确定 OEM 地址的含义。该功能通过编译循环带来。保留 5 个 OEM 地址。地址名称可以设定。在每个程序段中允许 OEM 地址。

参数

保留的 G 组

带有 OEMIPO1，OEMIPO2 的组 1

OEM 用户可以定义 G 函数 OEMIPO1，OEMIPO2 的两个辅助名称。该功能通过编译循环带来，保留给 OEM 用户。

- 带有 **G810 ～ G819** 的组 31
- 带有 **G820 ～ G829** 的组 32

可以保留 2 个 G 组给 OEM 用户，每个有 10 个 OEM－G－功能。**这样 OEM 用户带来的功能可以供外界使用。**

功能和子程序

此外，OEM 用户也可以通过参数传送设计预定义功能和子程序。

4.10 带有角部减速的进给减速 (FENDNORM, G62, G621)

功能

在自动拐角延迟时，在距离拐角很近处以钟形曲线降低进给速度。 除此之外，关系到加工的刀具性能的范围可以通过设定数据进行参数设定。 它们是：

- 开始和结束进给速度降低
- 用来减小进给速度的修调率
- 识别相关角

有些角部被视为重要的角部，即其内角小于通过调整数据所设定参数的角部。

使用 FENDNORM 缺省值关闭自动拐角倍率的功能。

文献：
/FBFA/ 功能说明 ISO 方言

句法

```
FENDNORM
G62 G41
G621
```

含义

FENDNORM	自动拐角延迟关
G62	激活刀具半径补偿时的内拐角减速
G621	激活刀具半径补偿时在所有角部减速

G62 仅作用于内角，带有

- 有效的刀具半径补偿 G41, G42 和
- 有效的轨迹控制运行 G64, G641

以降低后的进给速度逼近相应的角部，该进给速度来自于：

F * (用于降低进给速度的倍率) * 进给速度倍率

当刀具（以中心点轨迹为基准）在相应角应该变换方向时，表明已经到达了最大可能的进给减速。

G621 与 G62 相似作用于通过 FGROUP 所确定的轴的每个角

4.11 可编程的运动结束条件 (FINEA, COARSEA, IPOENDA, IPOBRKA, ADISPOSA)

功能

与轨迹插补 (G601, G602 和 G603) 的程序段转换条件相似，单轴插补的运动结束条件可以在一个零件程序或者适用于指令/PLC 的同步动作中进行编程。

要看设置了哪个运动结束条件的情况而定，以不同的速度来结束带有单轴运动的零件程序的程序段或者工艺循环程序段。 同样适用于 PLC，通过 FC15/16/18。

句法

```
FINEA[<轴>]
COARSEA[<轴>]
IPOENDA[<轴>]
IPOBRKA(<轴>[,<时间>])
ADISPOSA(<轴>[,<模式>,<窗口大小>])
```

含义

FINEA:	在到达“精准停”时运动结束
COARSEA:	在到达“粗准停”时运动结束
IPOENDA:	当到达插补器停止时结束运动
IPOBRKA:	可以在制动斜坡上进行程序段转换
ADISPOSA:	运动结束条件的公差窗口大小
<轴>:	通道轴名称 (X, Y, ...)
<时间>:	程序段转换时间点与制动斜坡百分比有关
<模式>:	模式
	类型： INT
	取值范围： 0 公差窗口无效
	1 公差窗口与给定位置相关
	2 公差窗口与实际位置相关

<窗口大小>: 公差窗口大小

该值以与主运行同步的方式输入到设置数据 SD43610 $SA_ADISPOSA_VALUE 中。

类型： REAL

示例

示例 1： 当到达插补器停止时结束运动

程序代码	注释
...	
N110 G01 POS[X]=100 FA[X]=1000 ACC[X]=90 IPOENDA[X]	以 1000 转/分钟的轨迹速度和 90% 的加速度值运行到位置 X100，在到达插补器停止时运动结束。
...	
N120 EVERY $A_IN[1] DO POS[X]=50 FA[X]=2000 ACC[X]=140 IPOENDA[X]	; 当输入端 1 有效时，以 2000 转/分钟的轨迹速度和 140% 的加速度值运行到位置 X50，到达插补器停止时运动结束。
...	

示例 2： 零件程序中制动斜坡程序段转换条件

程序代码	注释
	; 缺省设定生效
N40 POS[X]=100	; 如果 X 轴到达位置 100 和精准停，就开始进行程序段转换。
N20 IPOBRKA(X,100)	; 激活制动斜坡程序段转换条件。
N30 POS[X]=200	; 一旦 X 轴开始制动，就立即进行程序段转换。
N40 POS[X]=250	; X 轴不在位置 200 方向上制动，而是继续向位置 250 运动，只要 X 轴开始制动，就立即进行程序段转换
N50 POS[X]=0	; X 轴制动且向位置 0 返回，在到达位置 0 和精准停止时进行程序段转换。
N60 X10 F100	
N70 M30	
...	

示例 3： 同步动作中制动斜坡程序段转换条件

程序代码	注释
	; 在工艺循环中：
FINEA	; 运动结束条件精准停。
POS[X]=100	; 当 X 轴已到达位置 100 和精准停止时，就进行工艺循环程序段转换。
IPOBRKA(X,100)	; 激活制动斜坡程序段转换条件。
POS[X]=100	; POS[X]=100：只要 X 轴开始制动，就立即进行工艺循环程序段转换。
POS[X]=250	; X 轴不在位置 200 方向上制动，而是继续向位置 250 运动，只要 X 轴开始制动，就立即在工艺循环中进行程序段转换。
POS[X]=250	; X 轴制动并返回到位置 0，程序段转换在位置 0 和精准停时进行。
M17	

其它信息

读取运行结束条件

可以使用系统变量 $AA_MOTEND[轴] 查询设置好的运动结束条件：

值	含义
1	使用“精准停”结束运动
2	使用“粗准停”结束运动
3	使用“IPO-Stop”结束运动
4	轴运动的制动斜坡程序段转换条件
5	制动斜坡中的程序段转换，带有相对于“额定位置”的允差范围
6	制动斜坡中的程序段转换，带有相对于“实际位置”的允差范围

说明

在复位之后最后编程的值仍存在。

在制动斜坡中程序段转换条件

在执行主过程的同时百分比值被记录在以下设定数据中：

SD43600 $SA_IPOBRAKE_BLOCK_EXCHANGE

如果对数值不做说明，则设定数据中的当前值生效。

设定范围是 0 %到 100 %。

IPOBRKA 中附加的公差窗口

除了已有的制动斜坡中的程序段转换条件之外，还可以选择程序段转换条件公差范围。
在下面条件下才可以使能：

- 当轴到目前为止已经达到其制动斜坡的规定百分比值时

  以及

- 其当前实际位置或者额定位置与程序段中轴的终点位置的允差相差不大时。

文献

有关定位轴程序段转换条件的其它信息参见：

- 功能手册 扩展功能；定位轴（P2）
- 编程手册 基本原理，章节“进给控制”

4.12 可编程的伺服参数程序段 (SCPARA)

功能

使用 SCPARA 可以对零件程序和同步动作中的参数程序段（由 MD 组成）进行编程（迄今为止仅通过 PLC）。

DB3n DBB9 位 3

为使 PLC 和 NC 之间不出现冲突，在 PLC–>NCK 接口上再定义一个位：

DB3n DBB9 位 3 "参数程序段设定已被 SCPARA 锁止".

如果仍然对此进行编程，则在 SCPARA 禁止参数组给定时不会产生出错报警。

句法

```
SCPARA[<轴>]=<值>
```

含义

SCPARA	确定参数组
<轴>	通道轴名称(X, Y, ...)
<值>	所要求的参数组（1<= 值 <=6)

说明

可以使用系统变量 $AA_SCPAR[<轴>] 对当前参数程序段进行查询。

使用 G33, G331 或 G332 时最适合的参数组由控制系统选择。

如果**伺服参数程序段**要在某个零件程序中或者某个同步动作和 PLC 中**进行转换**，就必须对 PLC 用户程序进行扩展。

参考文献：

/FB1/ 功能手册 基本功能；进给率（V1），

章节“进给控制”。

示例

程序代码	注释
...	
N110 SCPARA[X]= 3	; 为轴 X 选择第 3 个参数程序段。
...	

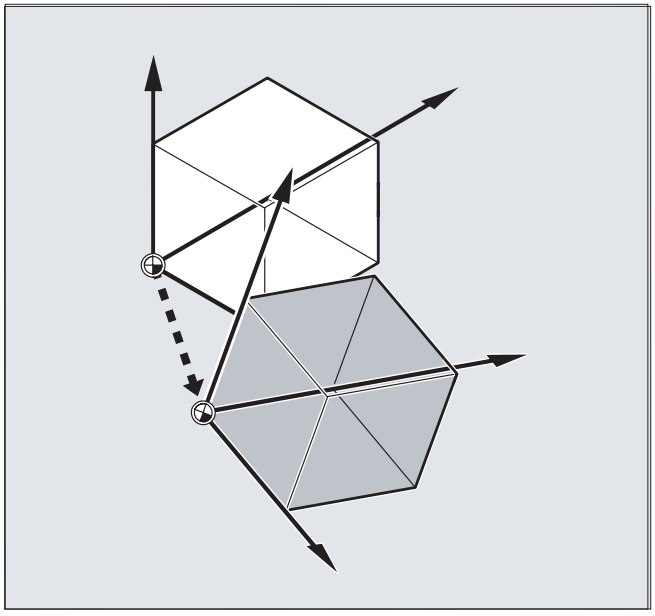
坐标转换 (FRAMES)

5

5.1 通过框架变量转换坐标

功能

除了在编程手册“基础部分”中所说明的编程方法之外，坐标系也可以用预定义的框架变量确定。



下列坐标系已经定义：

MKS:机床坐标系

BKS:基准坐标系

BNS:基准零点坐标系

ENS:可设定的零点坐标系

WKS:工件坐标系

什么是预定义框架变量？

预定义的框架变量已经在控制器的语言中规定了相应的含义，并可以在 NC 程序中进行处理。

可能的框架变量：

- 基准框架（基准偏移）
- 可设定的框架
- 可编程的框架

赋值和读取实际值

框架变量和框架之间的关系

坐标系转换可以通过框架给一个框架变量赋值而激活。

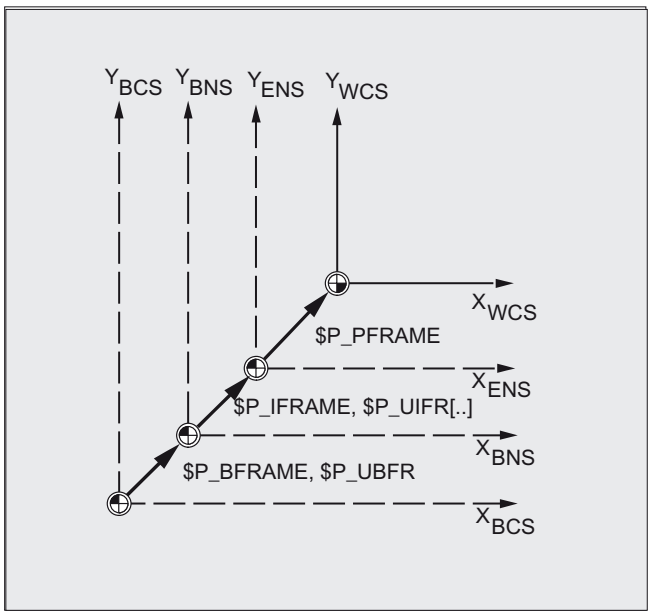
举例：`$P_PFRAME=CTRANS(X,10)`

框架变量

`$P_PFRAME` 表示：当前可编程的框架。

框架：

`CTRANS(X,10)` 表示：X 轴可编程的零点偏移为 10 毫米。



读取实际值

通过零件程序中的预定义变量可以读取坐标系的当前实际值：

$AA_IM[轴]:在 MKS 中读出实际值

$AA_IB[轴]:在 BKS 中读出实际值

$AA_IBN[轴]：在 BNS 中读出实际值

$AA_IEN[轴]:在 ENS 中读出实际值

$AA_IW[轴]:在 WKS 中读出实际值

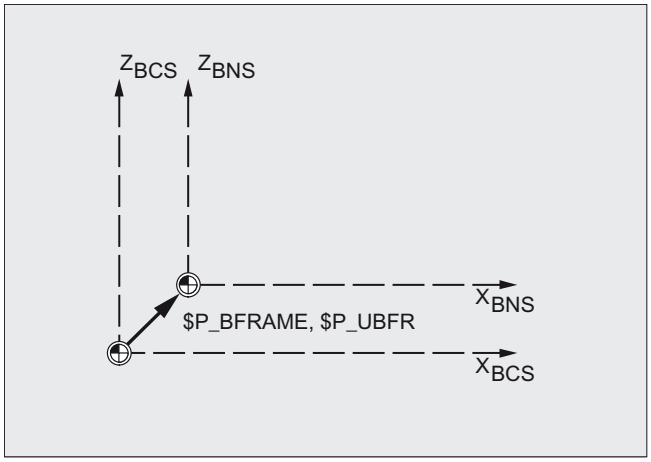
5.1.1 预定义框架变量 ($P_BFRAME, $P_IFRAME, $P_PFRAME, $P_ACTFRAME)

$P_BFRAME

当前的基准框架变量，建立基准坐标系(BKS)和基准零点坐标系(BNS)之间的关系。

如果要使通过 $P_UBFR 所写入的基本框架立即在程序中有效，就必须

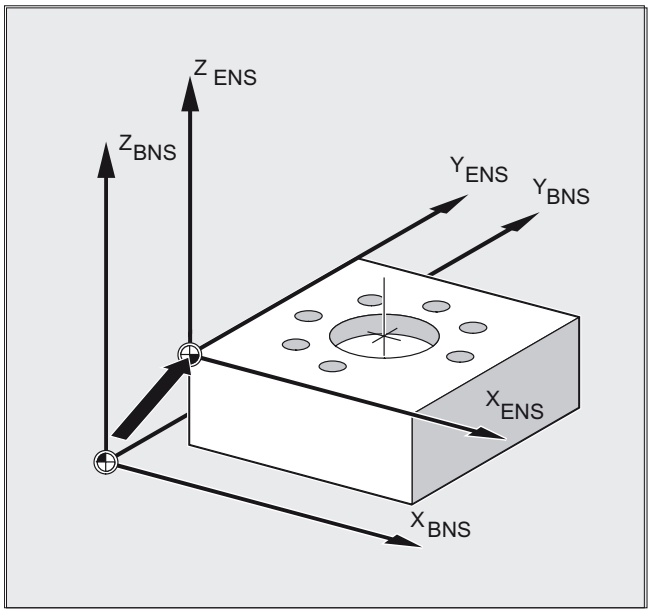
- 编程一个 G500, G54...G599 或者
- 写入$P_BFRAME 与$ $P_UBFR 。



$P_IFRAME

当前可设定的框架变量，建立基准零点坐标系(BNS)和可设定零点坐标系(ENS)之间的关系。

- $P_IFRAME 相当于$P_UIFR[$P_IFRNUM]
- 例如，在编程了 G54 之后，$P_IFRAME 就会含有通过 G54 所定义的转换、旋转、缩放和镜像。

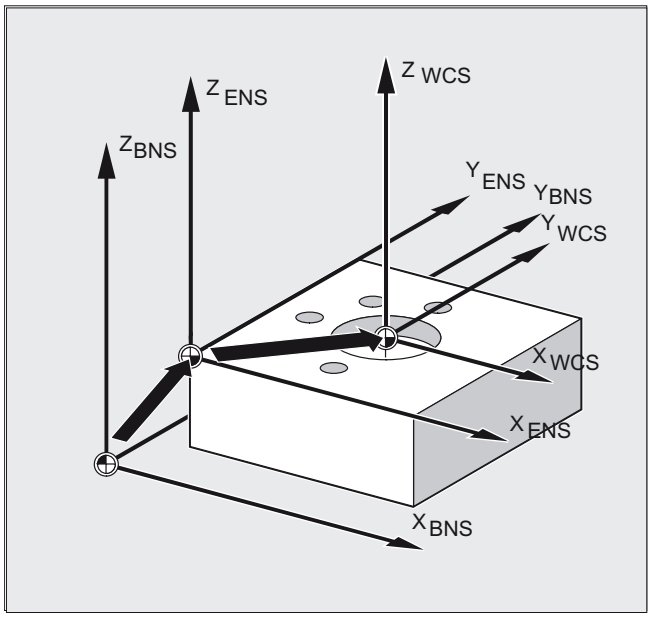


$P_PFRAME

当前可编程的框架变量，建立可设定零点坐标系(ENS)和工件坐标系(WKS)之间的关系。

$P_PFRAME 含有

- **从编程** TRANS/ATRANS, ROT/AROT, SCALE/ASCALE, MIRROR/AMIRROR 或者
- **从赋值** CTRANS, CROT, CMIRROR, CSCALE 给可编程的 FRAME 得出的合成框架。

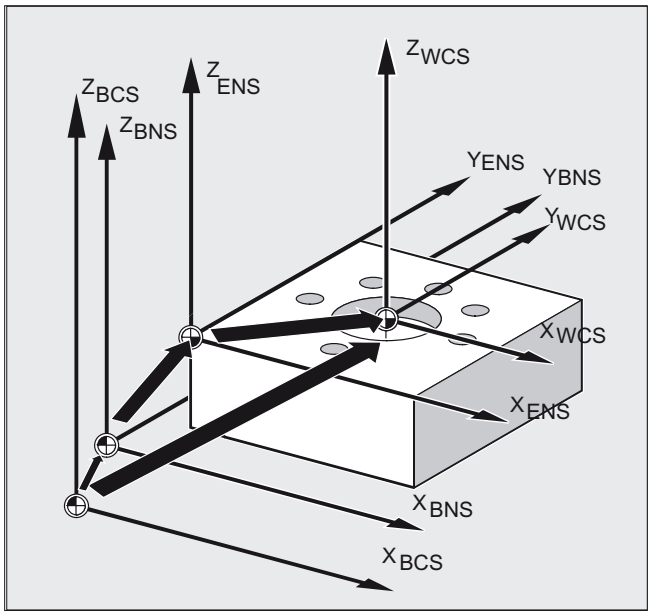


$P_ACTFRAME

通过级联从

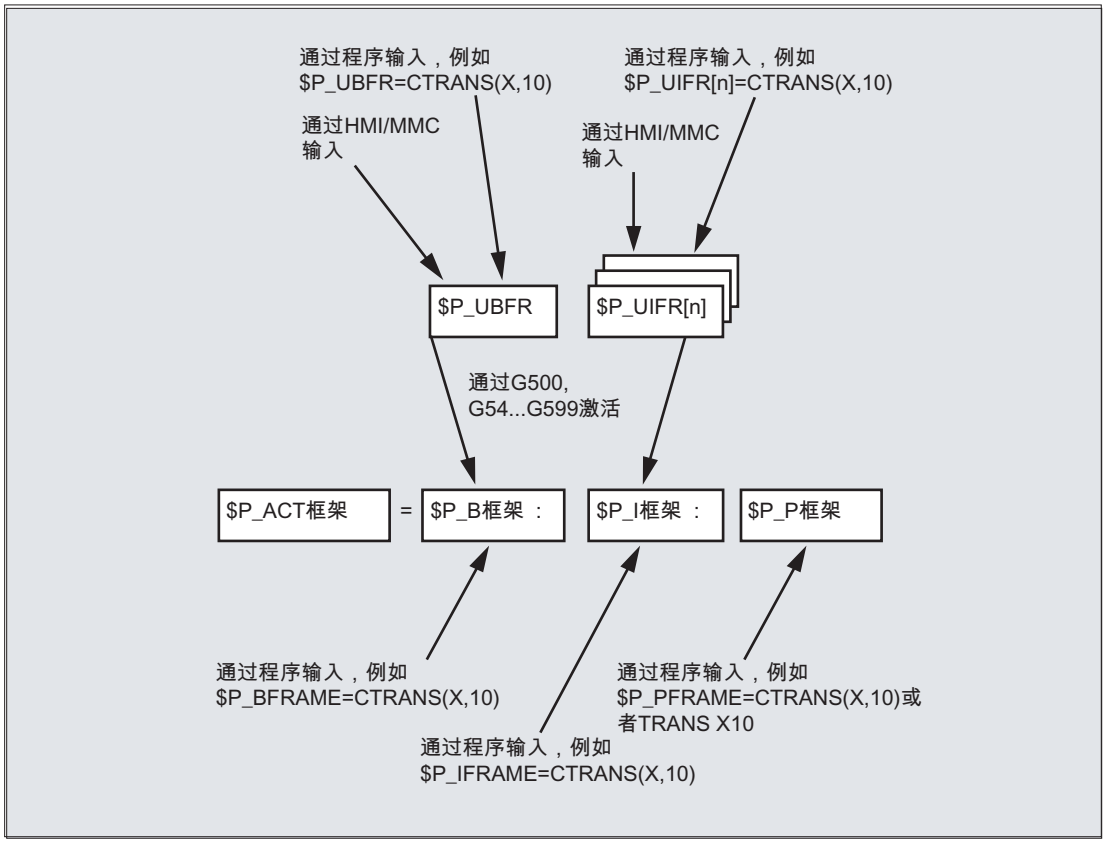
- 当前基本框架变量 $P_BFRAME,
- 当前的可设置框架变量 $P_IFRAME 与系统框架和
- 当前的可编程框架变量 $P_PFRAME 与系统框架

得出的当前的合成总框架。 系统框架，参见“在通道中有效的框架”一章 $P_ACTFRAME 所描述的是当前有效的工件零点。



如果 $P_BFRAME, $P_IFRAME 或者 $P_PFRAME 被改变，就重新计算 $P_ACTFRAME 。

$P_ACTFRAME 相当于 $P_BFRAME:$P_IFRAME:$P_PFRAME



如果 MD20110RESET_MODE_MASK 按照如下方式设定，则复位之后基准框架和可设定框架生效：

位 0=1, 位 14=1 --> $P_UBFR (基本框架)有效

位 0=1, 位 5=1 --> $P_UIFR[$P_UIFRNUM] (可设置的框架)有效

预定义可设定框架$P_UBFR

通过$P_UBFR 编程基准框架，但是不会在零件程序中同时生效。 在下面的情况下，用 $P_UBFR 编写的基准框架一并考虑：

- 接通复位，MD RESET_MODE_MASK 的位 0 和 14 设置。
- 语句 G500, G54...G599 已被执行。

预定义可设定框架$P_UIFR[n]

通过预定义框架变量 $P_UIFR[n] 可以从零件程序出发读取或者写入可设置的零点位移 G54 ～ G599 。

这些变量所表示的是名称为$P_UIFR[n] 的 FRAME 类型的一维数组结构。

G 指令的分配

按照标准有 5 个可调节框架 $P_UIFR[0]... $P_UIFR[4] 或者 5 个同样意义的 G-指令– G500 和 G54 到 G57 , 在这些地址下可以保存值。

$P_IFRAME=$P_UIFR[0] 相当于 G500

$P_IFRAME=$P_UIFR[1] 相当于 G54

$P_IFRAME=$P_UIFR[2] 相当于 G55

$P_IFRAME=$P_UIFR[3] 相当于 G56

$P_IFRAME=$P_UIFR[4] 相当于 G57

通过机床数据可以改变框架的个数：

$P_IFRAME=$P_UIFR[5] 相当于 G505

... ...

```
$P_IFRAME=$P_UIFR[99] 相当于 G599
```

说明

这样可以生成总计 100 个坐标系，例如可以超越程序范围将这些坐标系作为各种装置的零点来调用。

⚠小心

对框架变量和框架进行编程需要在 NC 程序中有一个自有 NC 程序段。 **特例：** 编程一个可设置的框架用 G54，G55，...

5.2 给框架变量/框架赋值

5.2.1 直接赋值（轴值，角度，尺寸）

功能

在 NC 程序中可以直接给框架或者框架变量赋值。

句法

```
$P_PFRAME=CTRANS (X, 轴值, Y, 轴值, Z, 轴值, …)
$P_PFRAME=CTOT (X, 角度, Y, 角度, Z, 角度, …)
$P_UIFR[..]=CROT（X，角度，Y，角度，Z，角度，…）
$P_PFRAME=CSCALE (X, 比例, Y, 比例, Z, 比例, …)
$P_PFRAME=CMIRROR (X, Y, Z)
```

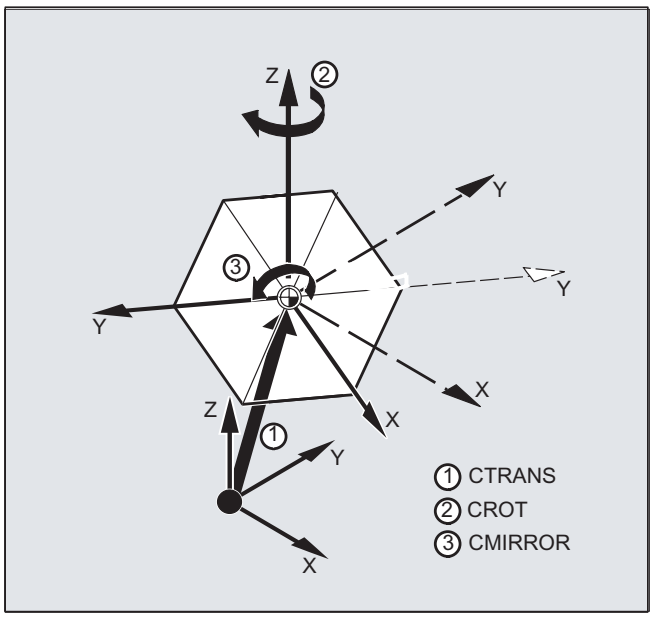
$P_BFRAME 的编程与 $P_PFRAME 相同。

含义

CTRANS	在给定轴上的偏移
CROT	围绕给定轴旋转
CSCALE	在给定轴上的比例改变
CMIRROR	在给定轴上的反向
X Y Z	在所给定的几何轴方向的偏移值
轴值	位移的轴值赋值
角度	围绕指定轴的旋转角赋值
标尺	改变比例尺

示例

通过在当前的可编程框架上赋值来激活转换、旋转和镜像。




```
N10 $P_PFRAME=CTRANS(X,10,Y,20,Z,5):CROT(Z,45):CMIRROR(Y)
```

给框架红色组件赋予其它的值

用 CROT 给 UIFR 的所有三个组件赋值

程序代码	注释
`$P_UIFR[5]=CROT (X, 0, Y, 0, Z, 0)`	
`N100 $P_UIFR[5, y, rt]=0`	
`N100 $P_UIFR[5, x, rt]=0`	
`N100 $P_UIFR[5, z, rt]=0`	

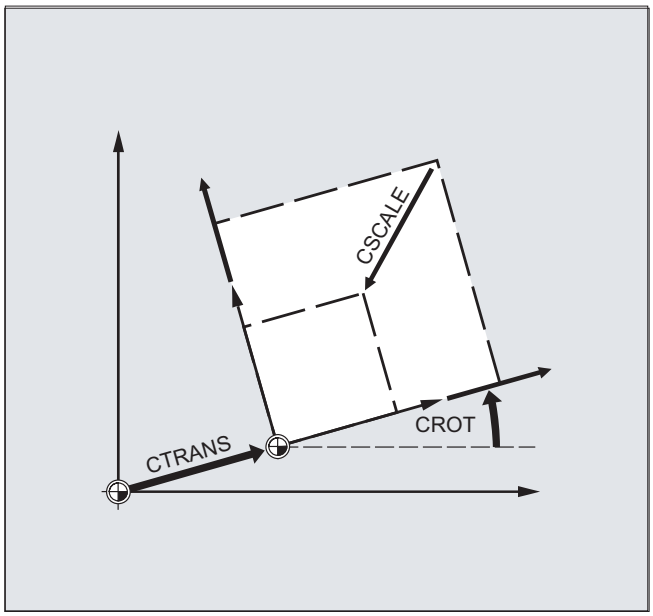
说明

您可以接连编程几个计算。

举例：

```
$P_PFRAME=CTRANS(...):CROT(...):CSCALE...
```

请注意：必须通过级联运算符冒号 (...):(...) 将这些指令相互联系起来。由此这些指令首先必须要相互逻辑联系，然后按照编程的顺序加法执行。



说明

用所给出的指令编程的值赋值给框架并存储。

只有当这些值被赋给某个激活的框架变量 $P_BFRAME 或者$P_PFRAME 的框架时才会激活。

5.2.2 读取和修改框架组件 (TR, FI, RT, SC, MI)

功能

可以对某个框架的**各个**数据进行访问，例如某个特定的位移值或者旋转角度。 这些值可以修改，或者赋值给另一个变量。

句法

R10=$P_UIFR[$P_UIFNUM,X,RT]	从当前的可设置零点位移 $P_UIFRNUM 得出的围绕 X 轴的旋转角度 RT 应当赋给变量 R10。
R12=$P_UIFR[25,Z,TR]	从已设置的编号为 25 的框架的数据集得出的 Z 轴中的位移值 TR 应当赋给变量 R12。
R15=$P_PFRAME[Y,TR]	给变量 R15 赋值 Y 轴的偏移值 TR，在当前可编程的框架中。
$P_PFRAME[X,TR] = 25	在当前可编程的框架中，改变 X 轴的偏移值 TR。 X25 立即适用。

含义

$P_UIFRNUM	使用该变量可以自动建立与当前可设定零点偏移坐标系的联系。
P_UIFR[n,…,…]	通过给出框架号 n，从而使用可设定框架 n。
	对需要读出或者修改的分量的说明：
TR	TR 转换
FI	FI 精细转换
RT	RT 旋转
SC	SC Scale 改变比例尺
MI	MI 镜像
X Y Z	此外（参见举例）还指定相应的轴 X，Y，Z。

RT 旋转的数值范围

围绕第 1 个几何轴旋转：	-180° ～ +180°
围绕第 2 个几何轴旋转：	-90° ～ +90°
围绕第 3 个几何轴旋转：	-180° ～ +180°

说明

调用框架

通过指定系统变量 $P_UIFRNUM 可以直接访问使用 $P_UIFR 或者 G54, G55, … 最新设置的零点位移
($P_UIFRNUM 含有最新设置的框架的编号)。

所有其它所保存的可设置框架 $P_UIFR 可通过指定相应的编号 $P_UIFR[n] 来调用。

可以为预定义框架变量和自定义框架指定名称，例如 $P_IFRAME.

数据调用

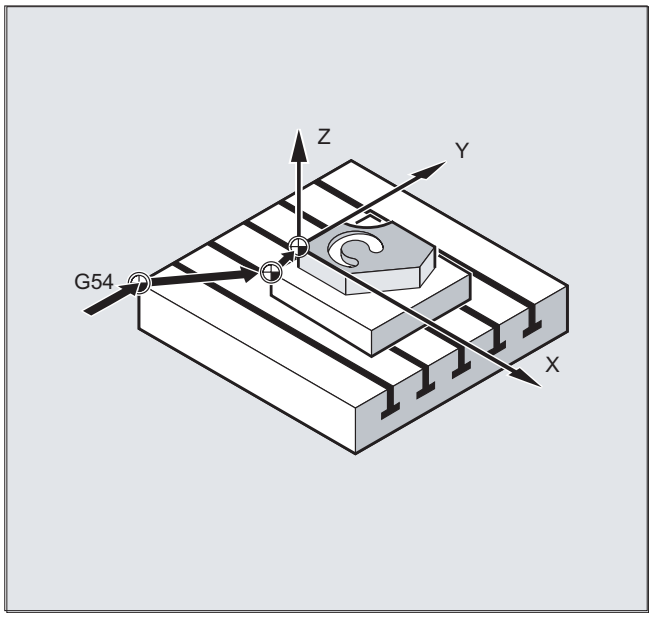
在方括号中的是要访问或者修改的轴名称和值的框架组件，例如 [X, RT] 或者 [Z, MI].

5.2.3 完整框架的逻辑联系

功能

在 NC 程序中，可以将某个完整的框架赋给另外一个框架或者使框架级联。

例如，框架级联适合用来描述排列在一个托盘上且应在一个加工流程中进行加工的多个工件。



描述托盘任务时，可以例如仅含有一些部分值，通过其级联来生成各种工件零点。

句法

给框架赋值

```
DEF FRAME EINSTELLUNG1
EINSTELLUNG1=CTRANS(X,10)
$P_PFRAME=EINSTELLUNG1
DEF FRAME EINSTELLUNG4
EINSTELLUNG4=$P_PFRAME
$P_PFRAME=EINSTELLUNG4
```

将自定义框架 EINSTELLUNG1 的值赋给当前的可编程框架。

当前可编程的框架存储在中间存储器中，在需要时再次返回。

框架级联

框架以所编程的顺序相互级联，框架组件例如位移、旋转等等按照先后次序累加执行。

$P_IFRAME=$P_UIFR[15]:$P_UIFR[16]	$P_UIFR[15] 含有例如零点位移的数据。 然后以此为基础，对 $P_UIFR[16] 的数据例如旋转的数据进行处理。
$P_UIFR[3]=$P_UIFR[4]:$P_UIFR[5]	可设定的框架 3 通过级联可设定的框架 4 和 5 产生。

说明

请注意：框架必须通过级联运算符冒号: 相互联系起来。

5.2.4 定义新框架 (DEF FRAME)

功能

除了前面所说的预定义的、可设定的框架之外，您也可以产生一些新框架。在此，与 FRAME 类型变量有关，您可以定义任意名称。

使用功能 CTRANS, CROT, CSCALE, CMIRROR 可以在 NC 程序中给您的框架赋值。

句法

```
DEF FRAME PALETTE1
PALETTE1=CTRANS(…):CROT(…)…
```

含义

`DEF FRAME`	生成新框架。
`PALETTE1`	新框架的名称
`=CTRANS(...):CROT(...) ...`	给可能有的功能赋值

5.3 粗偏移和精偏移 (CFINE，CTRANS)

功能

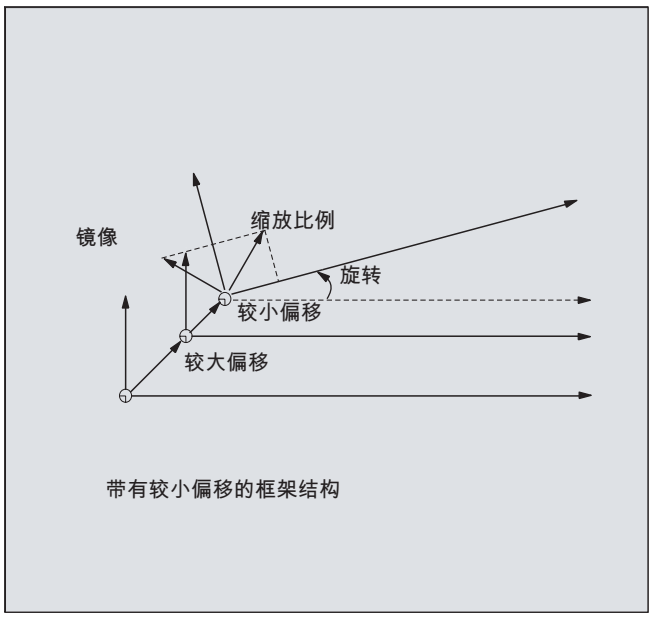
精偏移

使用指令 CFINE(X, ...,Y ...) 可以对基本框架和所有可设置框架的精位移进行编程。

精偏移只有在下面条件下才可以，即 MD18600 $MN_MM_FRAME_FINE_TRANS=1。

粗偏移

使用 CTRANS(...) 来确定粗位移。



粗偏移和精偏移相加成为最后的总偏移。

句法

```
$P_UBFR=CTRANS(x, 10) : CFINE(x,        ;位移的级联，
0.1) : CROT(x, 45)                      ;精位移和旋转
$P_UIFR[1]=CFINE(x, 0.5 y, 1.0, z,      ;整个框架可使用 CFINE
0.1)                                    ;包括粗位移
                                        ;来覆盖
```

通过组件说明 FI 来访问精位移的各个组件（精平移）。

```
DEF REAL FINEX                          ;定义变量 FINEX
FINEX=$P_UIFR[$P_UIFNUM, x, FI]         ;通过
                                        变量 FINEX 读取精位移
FINEX=$P_UIFR[3, x, FI]$P               ;通过变量 FINEX 读取
                                        第 3 个框架中 X 轴的精位移
```

含义

CFINE(x, 值, y, 值, z, 值)	多个轴的精位移。 累加式位移（平移）
CTRANS(x, 值, y, 值, z, 值)	多个轴的粗位移。 绝对位移（平移）。
x y z	轴的零点位移（最多 8）
值	平移部分

机床制造商

借助 MD18600 $MN_MM_FRAME_FINE_TRANS 可以用以下的变量设计精偏移：

0:
不可以输入或者不可以编程精位移。 不可以为 G58 和 G59。

1:
可以输入或者可以编程可设置框架、基本框架、可编程框架、G58 和 G59 的精位移。

说明

只有在激活相应的框架之后，某个通过 HMI 操作所改变的精位移才会激活，也就是说，通过 G500, G54...G599 激活。只要框架生效，激活的框架精偏移就一直有效。

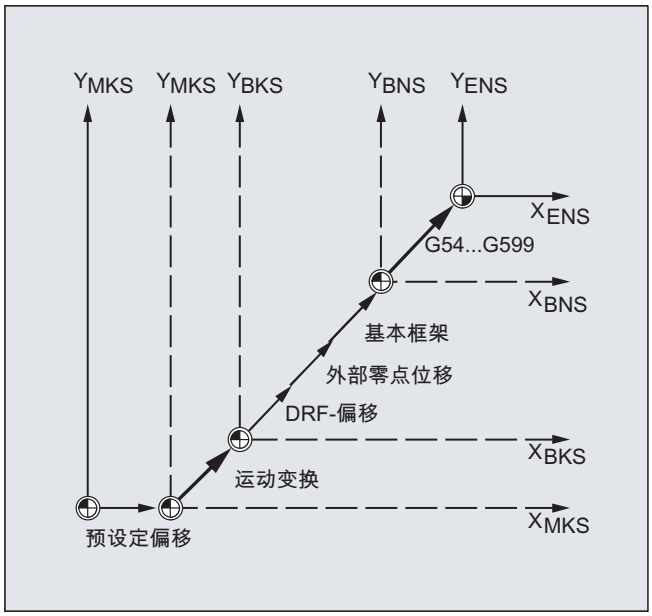
可编程的框架没有精偏移部分。如果带有精偏移的框架赋值给可编程的框架，则总偏移由粗偏移和精偏移的和构成。在读可编程框架时，精偏移始终为零。

5.4 外部零点偏移

功能

这就可以使您在基准坐标系和工件坐标系之间再次进行零点偏移。

在有外部零点偏移时，仅可以编程线性偏移。



编程

通过对特定轴的系统变量赋值来编程位移值$AA_ETRANS 。

偏移值赋值

```
$AA_ETRANS[Achse]=RI
```

RI 是含有新值的 REAL 型计算变量。

通常情况下外部偏移不在零件程序中说明，而是由 PLC 设置。

说明

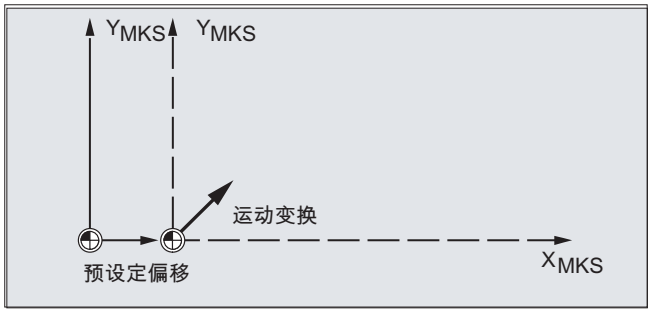
只有当 VDI-接口上
(NCU-PLC-接口) 设定有相应的信号时，在零件程序中写入的值才会有效。

5.5 预设定位移 (PRESETON)

功能

特殊应用时，可能需要使用 PRESETON 将一个新的实际值分配给已经回参考点的加工轴。 这类似于机床坐标系中的零点偏移。

> **⚠小心**
>
> 在执行 PRESETON 后加工轴的状态会变成"未回参考点"。 因此建议，该功能只在没有回参考点义务的加工轴上使用。 欲恢复原始的机床坐标系，必须使加工轴重新回参考点，例如可使用 G74（回参考点运行）。
>
> **资料：** 编程手册 基本原理，补充指令，回参考点运行(G74)



句法

```
PRESETON(<轴>,<值>, ...)
```

含义

PRESETON	设定实际值
<轴>	机床轴名称
<值>	机床坐标系中加工轴的新实际值

说明

只可使用关键字 WHEN 或者 EVERY 来进行带有同步动作的实际值设定。

示例

几何轴：A，对应的加工轴：X1

程序代码	注释
N10 G0 A100	; 轴 A 运行到位置 100
N20 PRESETON(X1,50)	; 加工轴 X1 在位置 100 处获得新的实际值 50 => 新实际值显示： - 轴 X1, MCS: 50 - 轴 A, WCS: 50
N30 A100	; 轴 A 运行 50mm 抵达位置 100

5.6 从空间中的三个测量点计算框架 (MEAFRAME)

功能

MEAFRAME 是 840D 语言的扩展，可支持测量循环。

使用功能 MEAFRAME 可以从三个理想的点及其相应的测量点计算出框架。

如果定位一个供加工的工件，则其位置相对于直角的机床坐标系及其理想位置可以偏移或者旋转。 用于精确加工或者测量时，要么需要进行成本高昂的物理调整，要么在零件程序中对运动进行修改。

通过在空间探测已知理想位置的三个点可以确定一个框架。 使用一个触碰标板上精确定位的专用孔或者测量球的接触式或者光电传感器进行探测。

句法

```
MEAFRAME IDEAL_POINT,MEAS_POINT,FIT_QUALITY)
```

含义

MEAFRAME	空间中 3 个测量点的框架计算 (MEAFRAME)
IDEAL_POINT	dim. 实数数组，它包含理想点的三个坐标。
MEAS_POINT	dim. 实数数组，它包含理想点的三个坐标。
FIT_QUALITY	实数型变量，以此来返回下列信息：
	-1: 这些理想点位于直线附近： 无法计算框架。 返回的框架变量含有一个中性框架。
	-1: 测量点几乎在一条直线上： 无法计算框架。 返回的框架变量含有一个中性框架。
	-2: 旋转矩阵的计算因另外一个原因而失败。
	-4: 将测定的三角形转换成一个与理想三角形一致的三角形所需的变形之和（点之间的距离）。
	正值：

说明

测量的质量

为了能够使用旋转/平移组合将所测定的坐标分配给理想的坐标，由测量点所确定的三角形必须与理想三角形一致。 应设法用一种可将偏差的平方之和减小到最小程度的补偿算法，将所测定的三角形转换成理想三角形。

测量点的有效所需变形可作为测量质量的指标，因此被 MEAFRAME 作为辅助变量输出。

说明

由 MEAFRAME 所生成的框架可以通过功能 ADDFRAME 转换成框架级联中的另一个框架。

参见示例： 框架的级联“带有 ADDFRAME 的级联”。

关于参数 ADDFRAME(FRAME, STRING) 的其它信息可参阅 /FB1/ 功能手册 基本功能；轴、坐标系、框架（K2），章节“FRAME 级联”。

示例

程序代码	注释
	; 零件程序 1
DEF FRAME CORR_FRAME	

设定测量点

编程	注释
DEF REAL IDEAL_POINT[3,3] = SET(10.0,0.0,0.0, 0.0,10.0,0.0, 0.0,0.0,10.0)	
DEF REAL MEAS_POINT[3,3] = SET (10.1,0.2,-0.2, -0.2,10.2,0.1, -0.2,0.2,9.8)	; 用于测试
DEF REAL FIT_QUALITY = 0	
DEF REAL ROT_FRAME_LIMIT = 5	; 最大允许有 5 度的零件位置旋转
DEF REAL FIT_QUALITY_LIMIT = 3	; 在理想和测量的三角之间 最大允许有 3 mm 的偏移
DEF REAL SHOW_MCS_POS1[3]	
DEF REAL SHOW_MCS_POS2[3]	
DEF REAL SHOW_MCS_POS3[3]	

程序代码	注释
N100 G01 G90 F5000	
N110 X0 Y0 Z0	
N200 CORR_FRAME=MEAFRAME(IDEAL_POINT,MEAS _POINT,FIT_QUALITY)	
N230 IF FIT_QUALITY < 0	
SETAL(65000)	
GOTOF NO_FRAME	
ENDIF	
N240 IF FIT_QUALITY > FIT_QUALITY_LIMIT	
SETAL(65010)	
GOTOF NO_FRAME	
ENDIF	
N250 IF CORR_FRAME[X,RT] > ROT_FRAME_LIMIT	; 第 1 个 RPY 角的极限值
SETAL(65020)	
GOTOF NO_FRAME	
ENDIF	
N260 IF CORR_FRAME[Y,RT] > ROT_FRAME_LIMIT	; 第 2 个 RPY 角的极限值
SETAL(65021)	
GOTOF NO_FRAME	
ENDIF	
N270 IF CORR_FRAME[Z,RT] > ROT_FRAME_LIMIT	; 第 3 个 RPY 角的极限值
SETAL(65022)	
GOTOF NO_FRAME	
ENDIF	
N300 $P_IFRAME=CORR_FRAME	; 激活带有一个可设置的框架的探测框架 ; 通过将几何轴向理想点定位的方式来检查框架
N400 X=IDEAL_POINT[0,0] Y=IDEAL_POINT[0,1] Z=IDEAL_POINT[0,2]	
N410 SHOW_MCS_POS1[0]=$AA_IM[X]	
N420 SHOW_MCS_POS1[1]=$AA_IM[Y]	
N430 SHOW_MCS_POS1[2]=$AA_IM[Z]	
N500 X=IDEAL_POINT[1,0] Y=IDEAL_POINT[1,1] Z=IDEAL_POINT[1,2]	
N510 SHOW_MCS_POS2[0]=$AA_IM[X]	
N520 SHOW_MCS_POS2[1]=$AA_IM[Y]	
N530 SHOW_MCS_POS2[2]=$AA_IM[Z]	
N600 X=IDEAL_POINT[2,0] Y=IDEAL_POINT[2,1] Z=IDEAL_POINT[2,2]	
N610 SHOW_MCS_POS3[0]=$AA_IM[X]	

程序代码	注释
N620 SHOW_MCS_POS3[1]=$AA_IM[Y]	
N630 SHOW_MCS_POS3[2]=$AA_IM[Z]	
N700 G500	; 取消可设定的框架，因为已使用零框架预置（没有事先输入值）
No_FRAME	; 取消可设定的框架，因为已使用零框架预置（没有输入值）
M0	
M30	

框架级联举例

级联 MEAFRAME 用于补偿

功能 MEAFRAME( ) 给出一个补偿框架。 当该补偿框架与调用功能时已激活的可设置框架 $P_UIFR[1] 级联时，例如 G54,就可得到一个可设置的框架，可继续换算成运动或者加工。

使用 ADDFRAME 级联

如果要让框架级联中的该补偿框架在另一个部位上发挥作用，或者在可设置框架之前尚有其它框架激活，则可将功能 ADDFRAME( ) 用来在其中一个通道基本框架或者某个系统框架中进行级联。

在这些框架中，以下功能不可生效：

- 使用 MIRROR 镜像
- 使用 SCALE 缩放

用于给定值和实际值的输入参数为工件坐标。 在控制器的基准系统中，这些坐标始终

- 为公制或者英制 (G71/G70) 作为
- 与半径有关的 (DIAMOF)

尺寸说明。

5.7 NCU 全局框架

功能

对于所有的通道，每个 NCU 仅有一个 NCU 全局框架。NCU 全局框架可以由所有的通道读写。分别在各个通道中激活 NCU 全局框架。

通过全局框架可以对带有位移的**通道轴和加工轴** 进行缩放和镜像。

几何关系与框架级联

在全局框架中各个轴之间没有几何关系。因此不可以进行旋转和编程几何轴名称。

- 全局框架中不可以使用旋转。编程旋转时会产生报警："18310 通道 %1 程序段 %2 框架:不可以旋转"
- 可以进行全局框架和通道专用框架的级联。最后生成的框架包含所有的框架分量，包括用于所有轴的旋转。如果带旋转分量的框架赋值于一个全局框架，则产生报警"框架：不可以旋转"。

NCU 全局框架

NCU-全局基本框架 $P_NCBFR[n]

可以设计 8 个以下的 NCU 全局基本框架：

通道专用的基本框架可以同时存在。

全局框架可以由一个 NCU 的所有通道读写。在写全局框架时，由用户考虑通道的协调。例如可以通过等候标记（WAITMC）来实现这一点。

机床制造商

全局基本框架的数量通过机床数据设计，参见
/FB1/ 功能手册基本功能；轴、坐标系、框架（K2）。

NCU-全局可设置框架 $P_UIFR[n]

所有可设置框架 G500, G54...G599 可以设计成 NCU-全局型或者通道专用型。

机床制造商

所有可设定的框架可借助一个机床数据 $MN_MM_NUM_GLOBAL_USER_FRAMES 再次设计为全局框架。

使用框架的编程指令时，可以使用通道轴名和加工轴名作为轴名称。编程几何轴名称时会出现报警，从而无法进行。

5.7.1 通道专用框架 ($P_CHBFR, $P_UBFR)

功能

可设定框架或者基准框架可以

- 通过零件程序和
- 通过机床控制面板

由操作装置例如 HMI Advanced 和 PLC 写入和读取。

精偏移也可以用于全局框架。和通道专用框架一样，也通过 G53，G153，SUPA 和 G500 来抑制全局框架。

机床制造商

通过 MD28081 MM_NUM_BASE_FRAMES 可以设定通道中基准框架的个数。默认配置被设计成每个通道至少有一个基本框架的形式。每个通道最多可以有 8 个基准框架。在通道中除了 8 个基准通道之外，还可以有另外 8 个 NCU 全局基准框架。

通道专用框架

$P_CHBFR[n]

通过系统变量 $P_CHBFR[n] 可以读取和写入基本框架。当写入某个基本框架时，级联的全部基本框架不会激活，而是在执行某个 G500，G54...G599-语句时才会激活。该变量主要在从 HMI 或者 PLC 写入到基本框架的过程中作为存储器使用。这些框架变量通过数据存储进行保护。

通道中的第一个基准框架

向预定义变量 $P_UBFR 写入时，不会同时激活数组索引为 0 的基本框架，而是在执行某个 G500，G54...G599-语句时才会激活。变量也可以在程序中读写。

$P_UBFR

$P_UBFR 和 $P_CHBFR[0]一样。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，则在读写时会产生报警“框架：“指令不允许”。

5.7.2 在通道中有效的框架

功能

在通道中有效的框架由零件程序通过这些框架的有关系统变量来输入。这里也包括系统变量。通过这些系统变量可以在零件程序中读写当前的系统框架。

当前在通道中有效的框架

一览

当前的系统框架	用于：
$P_PARTFRAME	TCARR 和 PAROT
$P_SETFRAME	实际值设定和刮削
$P_EXTFRAME	外部零点偏移
$P_NCBFRAME[n]	当前的 NCU 全局基准框架
$P_CHBFRAME[n]	当前的通道基本框架
$P_BFRAME	通道中当前的第一个基准框架
$P_ACTBFRAME	总的基准框架
$P_CHBFRMASK 和 $P_NCBFRMASK	总的基准框架
比如在编程 G54 之后，$P_IFRAME 包含由 G54 定义的平移、旋转、比例和镜像。	当前可设定的框架
当前的系统框架	用于：
$P_TOOLFRAME	TOROT 和 TOFRAME
$P_WPFRAME	工件基准点
$P_TRAFRAME	转换
$P_PFRAME	当前可编程的框架
当前的系统框架	用于：
$P_CYCFRAME	循环
P_ACTFRAME	当前的总框架
FRAME 级联	当前框架由全部基本框架组成

$P_NCBFRAME[n] 当前的 NCU-全局基本框架

通过系统变量 $P_NCBFRAME[n] 可以读取和写入当前的全局基本框架数组元素。在通道中写过程中，最后生成的总基准框架一起计算在内。

修改的框架仅在编程的通道中生效。如果要求修改一个 NCU 所有通道的框架，则必须同时说明 $P_NCBFR[n] 和 $P_NCBFRAME[n]。然后其它通道必须激活带有例如 G54 的框架。在写一个基准框架时，重新计算总的基准框架。

$P_CHBFRAME[n] 当前的通道基本框架

通过系统变量 $P_CHBFRAME[n] 可以读取和写入当前的通道基本框架数组元素。在通道中写过程中，最后生成的总基准框架一起计算在内。在写一个基准框架时，重新计算总的基准框架。

$P_BFRAME 通道中当前的第 1 个基本框架

通过预定义框架变量 $P_BFRAME 可以在零件程序中读取和写入带有在通道中有效的数组索引 0 的当前基本框架。写入的基准框架立即计算在内。

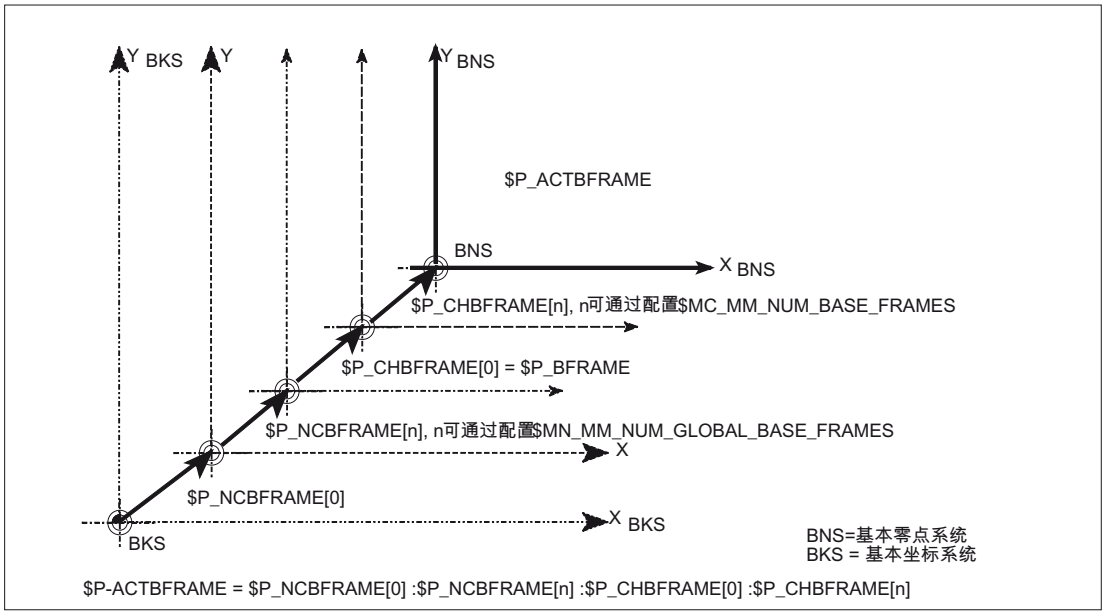
$P_BFRAME 和 $P_CHBFRAME[0]一样。在正常情况下，系统变量始终有一个有效值。如果没有通道专用基准框架，则在读写时会产生报警“框架：“指令不允许”。

$P_ACTBFRAME 全部基本框架

变量 $P_ACTFRAME 用来检查级联的全部基本框架。该变量仅可读。

$P_ACTFRAME 相当于

$P_NCBFRAME[0] : ... : $P_NCBFRAME[n] : $P_CHBFRAME[0] : ... : $P_CHBFRAME[n].



$P_CHBFRMASK 和 $P_NCBFRMASK 全部基本框架

用户可以通过系统变量 $P_CHBFRMASK 和 $P_NCBFRMASK 来选择要在计算“全部”基本框架时同时考虑哪些基本框架。变量仅在程序中编程，通过机床控制面板读入。将变量的值作为位掩码解释并且指定将 $P_ACTFRAME 的哪些基本框架数组元素考虑到计算中。

使用 $P_CHBFRMASK 可以设定将哪些通道专用基本框架考虑在内，且使用 $P_NCBFRMASK 来设定将哪些 NCU 全局基本框架考虑在内。

编程这些变量重新计算总的基准框架和总的框架。复位之后，在标准设置中有以下的数值：

$P_CHBFRMASK = $MC_CHBFRAME_RESET_MASK 和

$P_NCBFRMASK = $MC_CHBFRAME_RESET_MASK.

例如.

$P_NCBFRMASK = 'H81' ;$P_NCBFRAME[0] :$P_NCBFRAME[7]

$P_CHBFRMASK = 'H11' ;$P_CHBFRAME[0] :$P_CHBFRAME[4]

$P_IFRAME 当前的可设置框架

通过预定义框架变量 $P_IFRAME 可以在零件程序中读取和写入在通道中有效的当前可设置框架。写入的可设定框架立即计算在内。

在 NCU 全局的、可设定的框架中，修改的框架仅在编程的通道中生效。如果要修改某个 NCU 所有通道的框架，就必须同时写入 $P_UIFR[n] 和 $P_IFRAME 。然后其它通道必须激活带有例如 G54 的相应框架。

$P_PFRAME 当前的可编程框架

$P_PFRAME 是可编程框架，该框架从 TRANS/ATRANS, G58/G59, ROT/AROT, SCALE/ASCALE, MIRROR/AMIRROR 的编程中或者从赋值给可编程框架的 CTRANS, CROT, CMIRROR, CSCALE 得出。

当前的可编程框架变量，用来在可设置的

- 零点系统 (ENS) 和
- 工件坐标系（WCS）

之间建立关系。

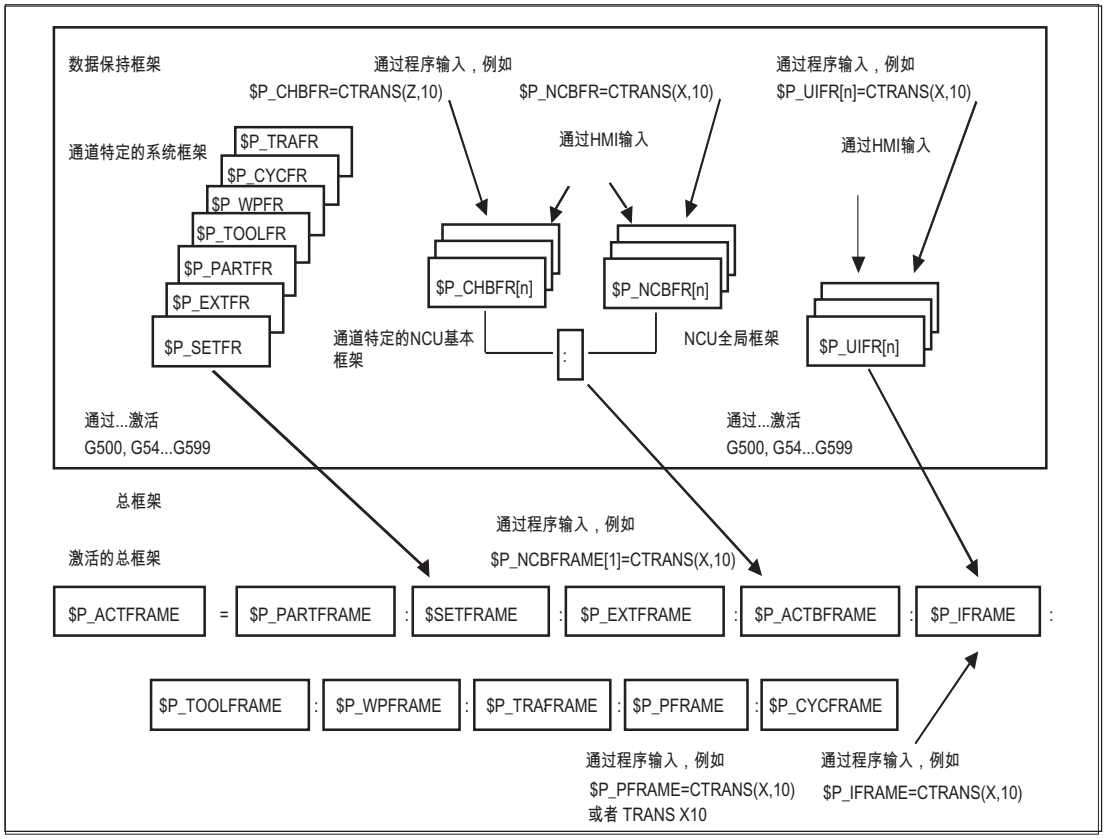
P_ACTFRAME 当前的总框架

当前的合成总框架 $P_ACTFRAME 现在作为级联受控于所有基本框架、当前的可设置框架和可编程框架。如果框架分量改变，则当前框架会更新。

$P_ACTFRAME 相当于

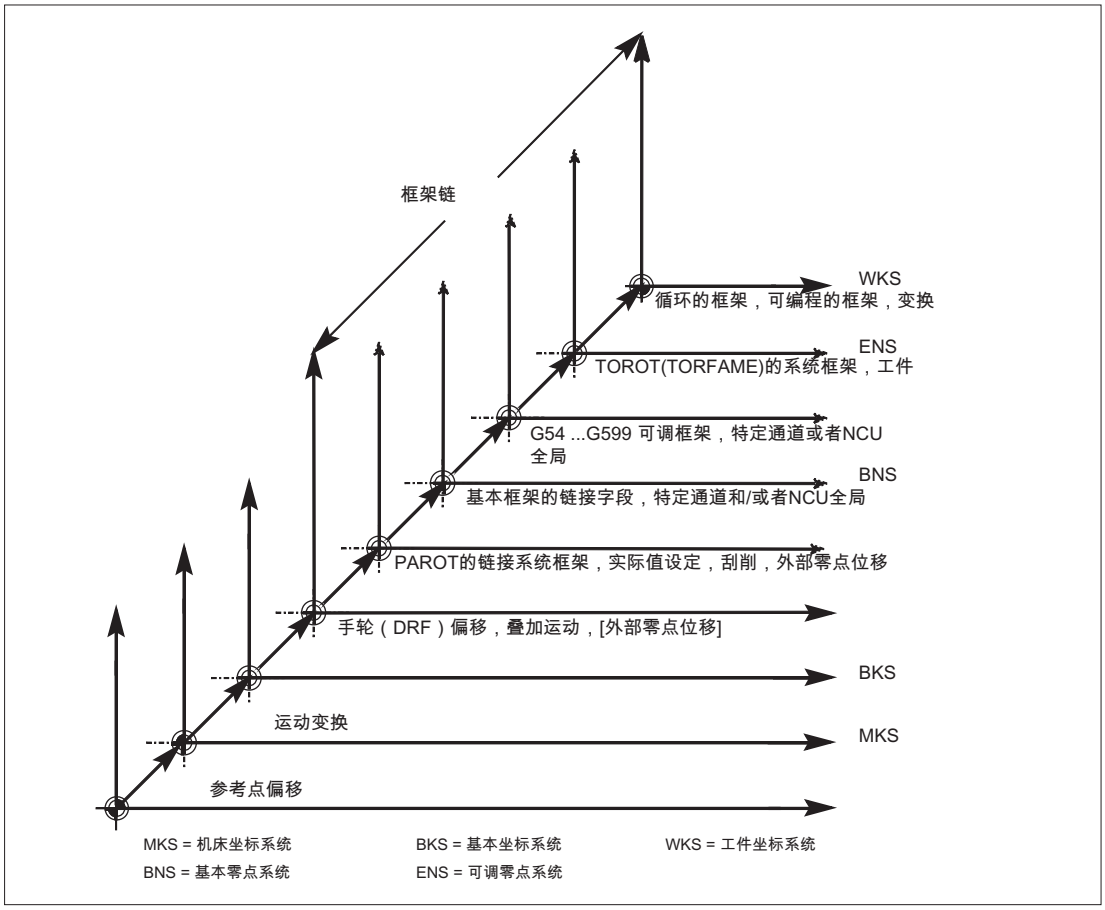
$P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME : $P_ACTBFRAME : $P_IFRAME :

$P_TOOLFRAME : $P_WPFRAME : $P_TRAFRAME : $P_PFRAME : $P_CYCFRAME



框架级联

根据以上所说的当前的总框架，当前的框架由总的基准框架、可设定的框架、系统框架和可编程的框架组成。



转换 6

6.1 转换方式的一般编程

一般功能

为了使控制系统能够匹配不同的机床运动，用合适的参数对所选的转换方式进行编程。通过这些参数，使刀具的空间定向和回转轴定向运动在所选的转换中达成一致。

在三轴、四轴和五轴转换中，进行编程的位置标注总是以刀尖为参照，它正交于空间加工平面。直角坐标系从基本坐标系转换到机床坐标系，并以几何轴为参照。对工作点进行描述。虚拟的回转轴描述了刀具的空间定向，用 TRAORI 对其进行编程。

运动转换时，在直角坐标系中对位置进行编程。控制系统把用 TRANSMIT,TRACYL 和 TRAANG 编程的直角坐标系过程运动转换成真实机床轴的过程运动。

编程

三轴、四轴和五轴转换（TRAORI）

约定的方位转换通过指令 TRAORI 和转换编号、方位矢量和回转轴偏移这三个可能的参数来激活。

TRAORI (转换编号、方位矢量、回转轴偏移)

运动转换

约定的转换 TRANSMIT (转换编号) 属于运动关系转换

TRACYL (加工直径、转换编号)

TRAANG (倾斜轴的角度、转换编号)

关闭有效转换

用 TRAFOOF 可以将正在激活中的转换关闭。

方位转换

三轴、四轴和五轴转换（TRAORI）

为了在机床加工空间中达到成形平面的最佳空间加工,机床除了 3 个线性轴 X、Y 和 Z 外还需要其他的轴。这些辅助轴用来描述空间中的定向，因此被称作定向轴。它们用作四种机床类型不同运动的旋转轴。

1. 两轴旋转头，例如：刀具台固定时，带有一个回转轴的万向刀具头平行于一个线性轴。
2. 两轴转台，例如固定式旋转头与可以围绕两个轴旋转的刀台
3. 单轴旋转头和单轴转台，例如一个可以旋转的旋转头，带有旋转式刀具，可围绕一个轴旋转的刀台。
4. 两轴旋转头和单轴转台，例如一个可围绕一个轴旋转的刀台和一个可绕自身旋转的带旋转刀具的旋转头。

3 轴和 4 轴转换是 5 轴转换的特殊形式，编程与 5 轴转换类似。

"生成的 3-/4-/5-/6-轴转换"可以用其用于垂直排列回转轴以及用于万向铣头转换的功能范围覆盖，可以象其它每个定向转换一样用 TRAORI 激活用于这四种机床类型。在生成 5/6 轴转换时，刀具方位有一个附加的第三自由度，有它可在空间中任意至刀具方向，刀具绕自身的轴可以任意旋转。

文献：/FB3/ 功能手册特殊功能；3 至 5 轴转换（F2）

刀具定向的初始位置与运动无关

ORIRESET

如果用 TRAORI 激活了一个定向转换，则可以用 ORIRESET 规定初始位置为最多 3 个带可选参数 A、B、C 的定向轴。根据通过转换确定的定向轴顺序分配已编程的参数到回转轴的顺序。编程 ORIRESET（A, B, C），使得线性和同步定向轴自其当前位置向给定的初始位置运行。

运动转换

TRANSMIT 和 TRACYL

车床进行铣削加工时，对于约定的转换，

1. 可以用 TRANSMIT 在车削夹装中编程一个端面加工或者
2. 用 TRACYL 在圆柱体上编程一个任意走向槽的加工

。

TRAANG

如果进给轴例如为了工艺磨削也可以倾斜进给，则可以用 TRAANG 为约定的转换编程一个可参数化的角。

直角坐标 PTP 运动

属于运动转换的还有“直角坐标系中的 PTP-运动”， 此种运动可以编程有 8 个以下的不同关节位置 STAT= 。这些位置在直角坐标系中进行编程，同时在机床坐标系中实现机床的运动。

文献：
/FB2/ 功能手册扩展功能；运动转换（M1）

级联的转换

每次可以前后进行两个转换。当对由此形成的级联进行第二个转换时，就会从第一个转换接管轴的运动分量。

作为第一个转换的可以是：

- 定向转换 TRAORI
- 极转换 TRANSMIT
- 圆柱转换 TRACYL
- 斜向轴转换 TRAANG

第二个转换必须是斜向轴 TRAANG

6.1.1 转换时的定向运动

过程运行和定向运动

可编程定向的过程运行首先取决于机床类型。 在用 TRAORI 进行三轴、四轴和五 轴转换时，旋转轴或者可摆动的线性轴体现了刀具的定向运动。

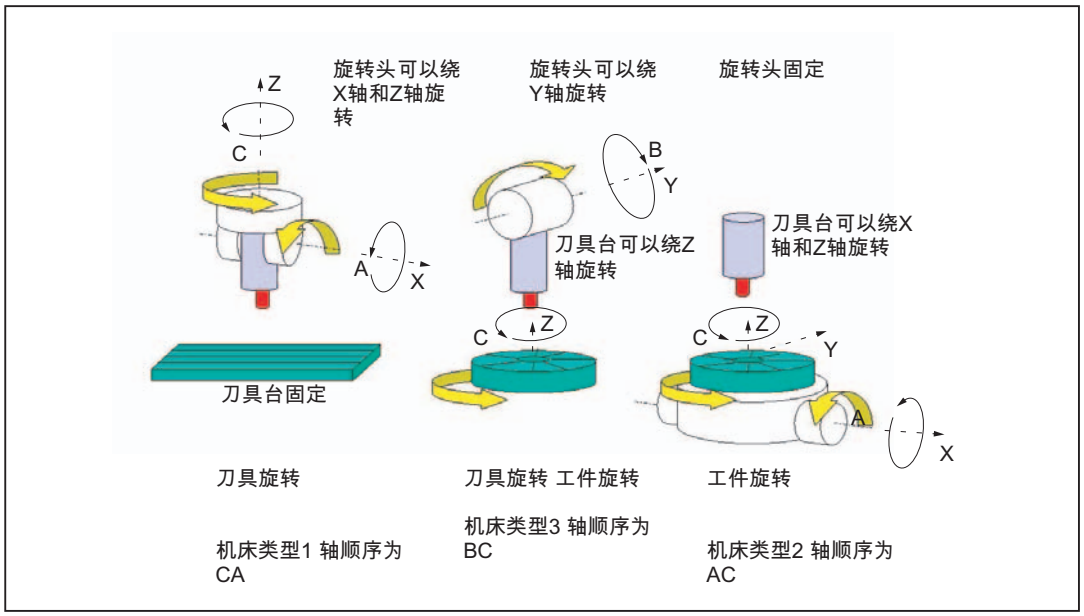
参与定向转换的回转轴的位置改变会导致其余加工轴的补偿运动。 因此刀尖的位置保持不变。

刀具的定向运动可以通过虚拟轴的回转轴标识符 A...,B...,C...分别按照用途进行编程，即通过标注欧拉角、RPY 角或者方向矢量或平面垂线矢量、用于圆锥旋转轴的标准矢量编程，或者在圆锥外表面为了暂时定向而编程。

当用 TRANSMIT,TRACYL 和 TRAANG 进行运动转换时，控制系统把编程的直角坐标系过程运动转换成真实机床轴的过程运动。

三轴、四轴和五轴转换（TRAORI）时的机床运动

刀具可旋转或者最多两个回转轴的刀台可旋转。 各单轴转动头和旋转台也可以组合在一起。



机床类型	对定向进行编程
机床类型 1 和 2 的三轴转换	仅在与旋转轴垂直的平面上对刀具定向进行编程。 有 **两个** 移动的轴(线性轴)和 **一个** 旋转的轴(回转轴)。
机床类型 1 和 2 的四轴转换	仅在与旋转轴垂直的平面上对刀具定向进行编程。 有 **三个** 移动的轴(线性轴)和 **一个** 旋转的轴(回转轴)。
机床类型 3 的 5 轴转换 单轴旋转头和单轴旋转台	对定向转换进行编程 带有 **三个** 线性轴和**两个** 正交旋转轴的运动。 旋转轴平行于三个线性轴中的其中两个线性轴。 第一个旋转轴被两个直角坐标线性轴所移动。 该旋转轴使第三个线性轴与刀具一起旋转。 第二个旋转轴使工件旋转。

生成 5/6 轴转换

机床类型	定向转换的编程
机床类型 4 的五/六轴转换 带可旋转刀具的两轴旋转头和单轴旋转台	对定向转换进行编程 带有 三个 线性轴和三个 正交旋转轴的运动。 旋转轴平行于三个线性轴中的其中两个线性轴。第一个旋转轴被两个直角坐标线性轴所移动。该旋转轴使第三个线性轴与刀具一起旋转。 第二个旋转轴使工件旋转。 可以通过一个以旋转角 THETA 围绕自身进行的额外旋转编程刀具的基本定向。

调用“生成的三轴、四轴、五轴和六轴转换”时，可以额外移交刀具基本定向。 这不再适用于回转轴方向的限制。 如果回转轴没有精确的彼此垂直或者现有的回转轴没有精确平行于线性轴，可能“生成的五/六轴转换”提供更好的刀具定向结果。

运动转换 TRANSMIT,TRACYL 和 TRAANG

对于车床上或者磨削时斜进给轴的铣削加工，取决于转换在标准情况下下列轴顺序有效：

TRANSMIT	激活极转换
卡装形式的端面加工	一个旋转轴 一个垂直于旋转轴的进给轴 一个平行于旋转轴的纵轴

TRACYL	激活圆柱面转换
在圆柱体上加工任意走向的槽口	一个旋转轴 一个垂直于旋转轴的进给轴 一个平行于旋转轴的纵轴

TRAANG	激活斜向轴转换
用斜置的进给轴加工	一个旋转轴 一个具有可设定参数的进给轴 一个平行于旋转轴的纵轴

直角坐标 PTP 运动

机床在机床坐标系中运动，并且编程时使用：

TRAORI	激活转换
PTP 点到点运行	在直角坐标系（MCS）中逼近位置
CP	直角轴的轨迹运动在（BKS）中
STAT	铰接的位置取决于转换。
TU	绕那个角度运行，轴的行程最短

生成 5/6 轴转换时的 PTP 运动

可以在机床坐标和刀具定向中通过回转轴位置和通过与运动无关的矢量欧拉以及 RPY 角或者方向矢量编程机床运动。

同时，可以沿一个圆锥表面进行回转轴插补、带大圆弧插补的矢量插补或者定向矢量插补。

举例：一个万向铣头的三到五轴转换

机床至少有 5 个轴，其中

- 用于直线运动的 3 个移动轴，在加工空间内把工作点向任意位置移动。
- 两个根据可设计的角度（通常为 45 度）排列的旋转运动轴，可以确定刀具在空间中的方位，当角度为 45 度时，这些方位局限到一个半球上。

6.1.2 定向转换 TRAORI 概述

与 TRAORI 相关的可能的编程方式

机床类型	当转换 TRAORI 有效时编程
机床类型 1、2 或者 3 两轴旋转头，两轴旋转台或者各单轴旋转头与旋转台的组合。	定向轴顺序和刀具定向方向 **以机床为参照** 时取决于机床运动结构， 可通过机床数据编程； **以工件为参照**时与机床运动结构无关， 可编程定向 编程定向轴的在参照系中的旋转方向时可以使用： - ORIMKS 参照系 = 机床坐标系 - ORIWKS 参照系 = 工件坐标系 缺省设置是 ORIWKS。 定向轴编程时使用： 直接的机床轴位置 A、B、C A2、B2、C2 虚拟轴角度编程 - ORIEULER 通过欧拉角（标准） - ORIRPY 通过 RPY 角 - ORIVIRT1 通过虚拟定向轴第 1 个定义 - ORIVIRT2 通过虚拟定向轴第 2 个定义 不同的插补方式： **直线插补** - 定向轴或机床轴的 ORIAXES **大圆插补**（定向矢量插补） - 定向轴的 ORIVECT 通过说明编程定向轴 矢量分量的 A3、B3、C3（方向/表面标准） 编程得出的刀具定向 程序段开始处表面标准矢量的 A4、B4、C4 程序段结束处表面标准矢量的 A5、B5、C5 用于刀具定向的提前角 LEAD 用于刀具定向的侧向角 TILT

机床类型	当转换 TRAORI 有效时编程
	定向矢量插补在一个圆锥表面上 定向变化在一个**在任意空间内**的 圆锥表面上，通过插补： - ORIPLANE 在平面中（大圆插补） - ORICONCW 在一个圆锥表面上，顺时针 - ORICONCCW 在一个圆锥表面上，逆时针 A6、B6、C6 方向矢量（圆锥的旋转轴） -OICONIO 在一个圆锥表面上插补： A7, B7, C7 中间矢量 (开始定向和结束定向)或者 - ORICONTO 在圆锥表面的切线过渡 改变定向参照 **一段轨迹** 用 - ORICURVE 两个触点运动的预设值通过 PO[XH]=(xe, x2, x3, x4, x5) 定向多项式至 5 次 PO[YH]=(ye, y2, y3, y4, y5) 定向多项式至 5 次 PO[ZH]=(ze, z2, z3, z4, z5) 定向多项式至 5 次 - ORIPATHS 用 A8, B8, C8 平滑 定向走向 刀具换向阶段适合: 退刀方向和行程长度

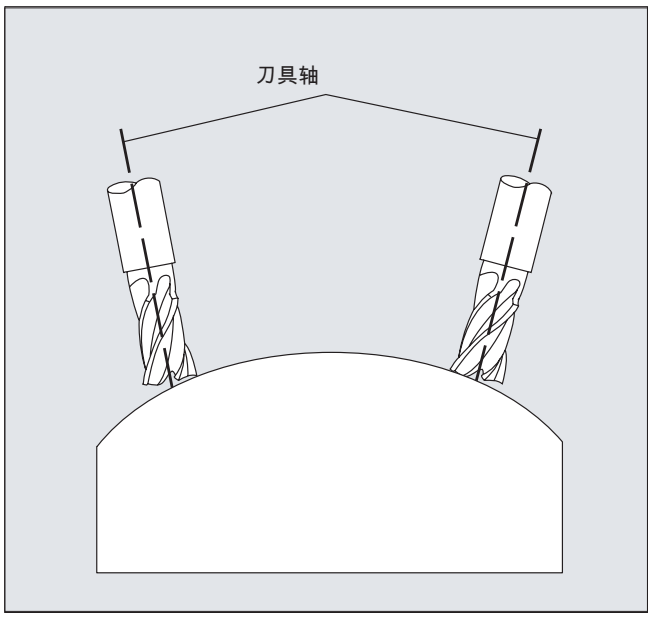
机床类型	当转换 TRAORI 有效时编程
机床类型 1 和 3	编程刀具定向**旋转**用 LEAD 超前角与平面垂线矢量相对 PO[PHI] 编程一个至 5 次的多项式 TILT 侧向角绕轨迹切线(Z-方向)旋转 PO[PSI] 编程一个至 5 次的多项式
其它带刀具绕自身额外旋转的机床类型要求第 3 个回转轴	THETA 旋转角(绕刀具在 Z 的方向旋转) THETA= 在程序段结尾达到的值 THETA=AC(...) 以程序段方式切换到绝对尺寸 THETA=IC(...) 以程序段方式切换到累接尺寸 THETA=Θ_e 插补编程的角度 G90/G91
定向转换，例如 6 轴转换。 方向矢量的旋转	PO[THT]=(..) 编程一个至 5 次的多项式 编程旋转矢量 - ORIROTA 绝对旋转 - ORIROTR 相对旋转矢量 - ORIROTT 切向旋转矢量
轨迹相关的定向：定向改变与轨迹相关或者旋转矢量的旋转正切于轨迹	定向改变**相对于轨迹** 用 - ORIPATH 刀具定向参照该轨迹 - ORIPATHS 额外在定向走势的一个拐点处 旋转矢量编程 - ORIROTC 与轨迹切线的切向矢量、旋转

6.2 三轴、四轴和五轴转换 (TRAORI)

6.2.1 万向切削头的一般关系

功能

当加工空间曲面时，为了获得最佳切削条件，刀具的定位角必须可以修改。

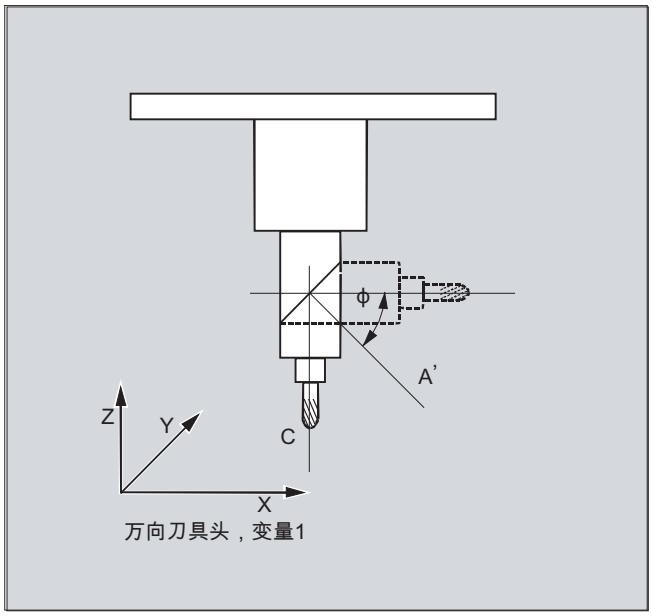


用哪一种机床结构达到这一点，这存储在轴数据中。

5 轴转换

万向切削头

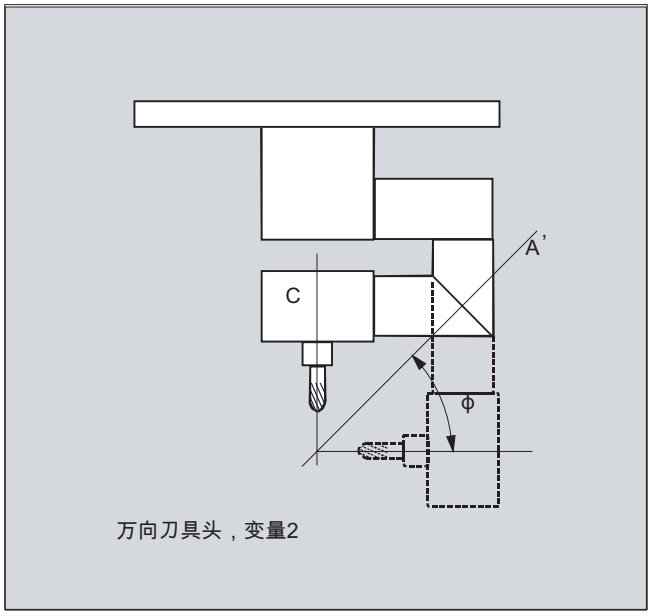
这里三个线性轴 (X, Y, Z) 和两个定向轴 (C, A) 用来确定刀具的定位角和工作点。两个定向轴的其中之一是作为斜置轴设置的，在这里为 A'（在很多情况下是 45°}£©¡£



在这里所举的示例中，可以看到带有机床运动系统的 CA 的万向组合刀盘的布置！

机床制造商

定向轴的轴顺序和刀具的运动方向取决于通过机床数据的机床类型来设置。



在该例中，A'位于与 X 轴成 φ 的角度下

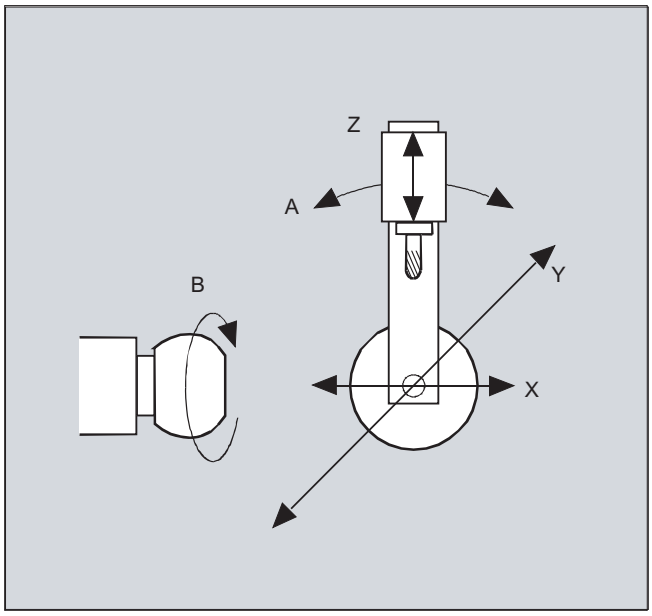
一般适用于下面的关系：

A'在角度 φ 下	X 轴
B' 在角度 φ 下	Y-轴
C' 在角度 φ 下	Z 轴

可以通过机床数据在 0° ～ +89° 的范围内设计角度 φ 。

带有可旋转的线性轴

这种情况与运动的工件和运动的刀具的布置有关。运动由三个线性轴 (X, Y, Z) 和两个呈直角的旋转轴合成。例如，通过一个有两个线性轴的滑板使第一个旋转轴运动，刀具平行于第三个线性轴。第二个旋转轴使工件旋转。第三个线性轴（转向轴）在十字滑板的平面上。



旋转轴的轴顺序和刀具的运动方向取决于通过机床数据的机床类型来设置。

适用于下面的关系：

进给轴	轴顺序：
1. 回转轴	A A B B C C
2. 回转轴	B C A C A B
转向的线性轴	Z Y Z X Y X

关于刀具定向方向可配置轴顺序的进一步说明参见

文献： /FB3/ 功能手册特殊功能；3 至 5 轴转换（F2），万向铣头一章中的“编程”。

6.2.2 三轴、四轴和五轴转换 (TRAORI)

功能

用户可以设计两个或者三个直线移动轴和一个旋转轴。 坐标转换的条件：旋转轴正交于定向平面上。

刀具只有定位在和旋转轴垂直的平面上。 坐标转换支持带有运动刀具和运动工件的机床类型。

三轴和四轴转换的设计和编程与五轴转换类似。

文献：
功能手册 特殊功能；多轴转换 (F2)

句法

```
TRAORI(<n>)
TRAORI(<n>,<X>,<Y>,<Z>,<A>,<B>)
TRAFOOF
```

含义

TRAORI:	激活第一个设定的方向转换
TRAORI(<n>):	激活用 n 个设定的方向转换
<n>:	转换编号
	值： 1 或 2
	示例：
	TRAORI(1) 激活方向转换 1
<X>,<Y>,<Z>:	刀具所指向的方向矢量分量
<A>,<B>:	旋转轴的可编程偏移
TRAFOOF:	取消转换

刀具定向

视所选的刀具定位方向而定，必须在 NC 程序中调整激活的工作平面 (G17, G18, G19)，使得刀具长度补偿在刀具定位的方向中有效。

说明

在启用坐标转换之后，位置数据（X，Y，Z）总是针对刀尖。 如果修改了参与坐标转换的旋转轴位置，其余机床轴也会开始补偿运动，使得刀尖的位置保持不变。

方向转换总是从刀尖指向刀夹。

方向轴的偏移

在激活方向转换时还可以直接编程一个方向轴的附加偏移。

如果在编程时确保了正确的顺序，参数可以不设。

示例：

```
TRAORI(, , , ,A,B) ；当只需输入一个唯一的偏移时
```

除了直接编程之外，方向轴的附加偏移还可以自动接收当前生效的零点偏移。 接收通过机床数据来设计。

示例

```
TRAORI(1,0,0,1)          ; 刀具的初始方向指向 Z 轴方向

TRAORI(1,0,1,0)          ; 刀具的初始方向指向 Z 轴方向

TRAORI(1,0,1,1)          ; 刀具的初始方向指向 Y/Z 轴方向（相当于 -45°）
```

6.2.3 定向编程变量和初始位置（ORIRESET）

TRAORI 时刀具定向的定向编程

使用可编程的定向转换 TRAORI 时，除了直线轴 X、Y、Z，还可以通过回转轴名称 A...,B...,C...来编程轴位置或者为虚拟轴编程角度或矢量分量。 定向轴和加工轴可用不同的插补方式。 与哪些定向多项式 PO[角度]和轴多项式 PO[轴]恰好激活无关，可以编程多个不同的多项式种类，例如 G1, G2, G3, CIP 或者 POLY

刀具定向的改变也可以通过定向矢量来编程。 由此，每个程序段的结束定位可通过直接矢量编程或者通过回转轴位置实现。

说明

三至五轴转换时定向编程的变量

三至五轴转换时，下列变量不能并存

1. A, B, C 机床位置直接标注
2. A2, B2, C2 通过欧拉角或者 RPY 角编程虚拟轴角度
3. A3 ,B3, C3 矢量分量标注
4. LEAD, TILT 以轨迹和表面为参照的超前角和侧向角标注
5. A4, B4, C4 和 A5, B5, C5 程序段开始和结束的平面垂线矢量
6. A6, B6, C6 和 A7, B7, C7 圆锥外面的定向矢量插补
7. A8, B8, C8 刀具重取向，退刀运动的方向和路径长度

是相互的。

混合编程的值通过报警信息被阻止。

刀具定向 ORIRESET 的初始位置

通过编程 ORIRESET（A, B, C），使得线性和同步定向轴自其当前位置向给定的初始位置运行。

如果未给轴编程初始位置，则使用所属机床数据 $MC_TRAFO5_ROT_AX_OFFSET_1/2 中定义的位置。 同时，不考虑回转轴当前有效的框架。

说明

仅在带 TRAORI(...) 的定向转换有效时，可以编程一个带 ORIRESET(...) 的与运动无关的刀具定向初始位置，不带报警 14101。

示例

```
1. 机床运动 CA（通道轴名称 C，A）举例
ORIRESET(90, 45)        ; C 为 90 度，A 为 45 度
ORIRESET(, 30)          ; C 为 $MC_TRAFO5_ROT_AX_OFFSET_1/2[0], A 为 30 度
ORIRESET( )             ; C 为 $MC_TRAFO5_ROT_AX_OFFSET_1/2[0],
                        ; A 为 $MC_TRAFO5_ROT_AX_OFFSET_1/2[1]
2. 机床运动 CAC（通道轴名称 C，A，B）举例
ORIRESET(90, 45, 90)    ; C 为 90 度，A 为 45 度，B 为 90 度
ORIRESET( )             ; C 为 $MC_TRAFO5_ROT_AX_OFFSET_1/2[0],
```

```
                                    ; A 为 $MC_TRAFO5_ROT_AX_OFFSET_1/2[1],
                                    ; B 为 $MC_TRAFO5_ROT_AX_OFFSET_1/2[2]
```

编程旋转 LEAD、TILT 和 THETA

三轴到五轴的转换时，刀具方位的旋转通过超前角 LEAD 和侧向角 TILT 编程。

用第三回转轴进行转换时，允许用矢量分量定向，也允许用角度标注 LEAD、TILT 对 C2（定向矢量旋转）进行额外编程。

用一个附加的第三回转轴可以编程以旋转角 THETA 使刀具绕自身旋转。

6.2.4 编程刀具定向 (A..., B..., C..., LEAD, TILT)

功能

对于刀具的定向编程有下列可能：

1. 直接编程回转轴运动。 定向改变总是在基准坐标系或者机床坐标系统中发生。 定向轴作为同步轴运动。
2. 通过 A2, B2, C2 用欧拉角或者 RPY 角编程。
3. 通过 A3, B3, C3 编程方向矢量。 方向矢量从刀尖指向刀夹。
4. 在程序段开始处，用 A4, B4, C4 编程平面垂线矢量，而在程序段结束处，则用 A5, B5, C5 编程（端面铣）。
5. 通过超前角 LEAD 和侧向角 TILT 编程
6. 通过 A6, B6, C6 编程圆锥旋转轴作为标准矢量或者编程圆锥表面的上的中间定向用 A7, B7, C7,
   ，参见章节“沿圆锥表面定向编程(ORIPLANE, ORICONxx)”。
7. 退刀运动时，通过 A8, B8, C8,
   编程刀具换向、方向和行程长度, 参见章节“平滑定向走势（ORIPATHS A8=, B8=, C8=）”

说明

在所有的情况下，只有当一个定向转换生效时，定向编程才是允许的。
优点： 这些程序在每个机床运动类型都是可传送的。

通过 G 代码定义刀具定向

说明

机床制造商

通过机床数据可以在欧拉角或 RPY 角之间转换。 对于相应的机床数据设置，可以根据也可以不根据组 50 有效的 G 代码进行切换。 可以选择下列设置方式：

1. 当用于定义定向轴和定向角的两个机床数据通过 G 代码置位 0 时：
   用 A2, B2, C2 编程的角 **根据机床数据** 将定向编程的角度定义编译为欧拉角或 RPY 角。
2. 如果用于定义定向轴的机床数据通过 G 代码置为 1，则
   根据 组 50 的有效 G 代码进行切换：
   用 A2, B2, C2 编程的角根据有效 G 代码 ORIEULER、ORIRPY、ORIVIRT1、ORIVIRT2、ORIAXPOS 和 ORIPY2 中的一个进行编译。 根据组 50 的有效 G 代码也可将用定向轴编程的值编译为定向角。
3. 如果用于定义定向角的机床数据通过 G 代码置为 1，而用于定义定向轴的机床数据通过 G 代码置为 0，则
   不根据 组 50 有效的 G 代码进行切换：
   用 A2, B2, C2 编程的角根据有效 G 代码 ORIEULER、ORIRPY、ORIVIRT1、ORIVIRT2 ORIAXPOS 和 ORIPY2 中的一个进行编译。 不根据组 50 的有效 G 代码总是将用定向轴编程的值编译为回转轴位置。

编程

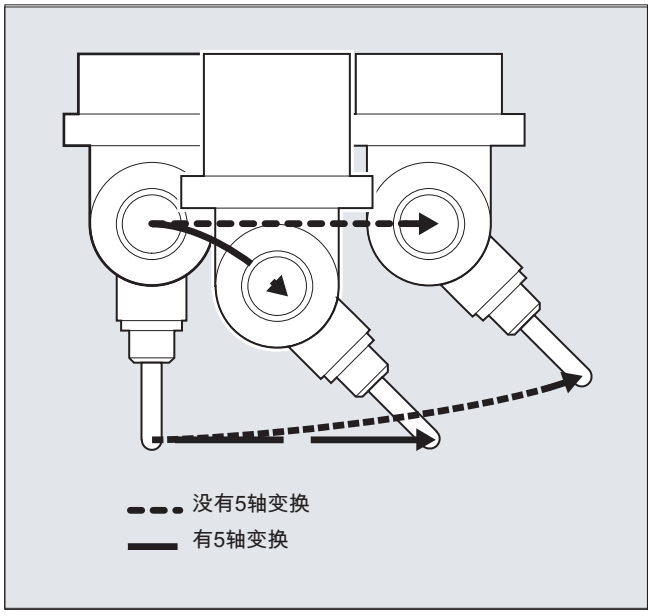
G1 X Y Z A B C	编程回转轴运动
G1 X Y Z A2= B2= C2=	编程欧拉角
G1 X Y Z A3== B3== C3==	编程方向矢量
G1 X Y Z A4== B4== C4==	在程序段开始时编程面正交矢量
G1 X Y Z A5== B5== C5==	在程序段结束时编程面正交矢量
LEAD=	刀具定向编程的超前角
TILT=	刀具定向编程的侧向角

参数

G....	指定旋转轴的运动方式
X Y Z	指定线性轴

A B C	指定旋转轴的加工轴位置
A2 B2 C2	虚拟轴或者定向轴的角度编程（欧拉角或者 RPY 角）
A3 B3 C3	指定方向矢量的矢量分量
A4 B4 C4	例如如果是端面铣削，指定程序段开始处面法向矢量的分量
A5 B5 C5	例如如果是端面铣削，指定程序段结束处面法向矢量的分量
LEAD	相对于面正交矢量的角度，在由轨迹切线和面正交矢量展开的平面上
TILT	平面上的角度，相对于面正交矢量垂直于轨迹切线

有/无 5 轴转换的对照举例



说明

通常情况下 5 轴程序由 CAD/CAM 系统生成并且没有输入到控制系统。因此，下列说明主要面向后处理器的编程人员。

在组 50 的 G 代码中确定定向编程的方式：

ORIEULER 通过欧拉角
ORIRPY 通过 RPY 角（旋转顺序 ZYX）
ORIVIRT1 通过虚拟的定向轴（定义 1）
ORIVIRT2 通过虚拟的定向轴（定义 2）
ORIAXPOS 通过带回转轴位置的虚拟定向轴
ORIPY2 通过 RPY 角（旋转顺序 XYZ）

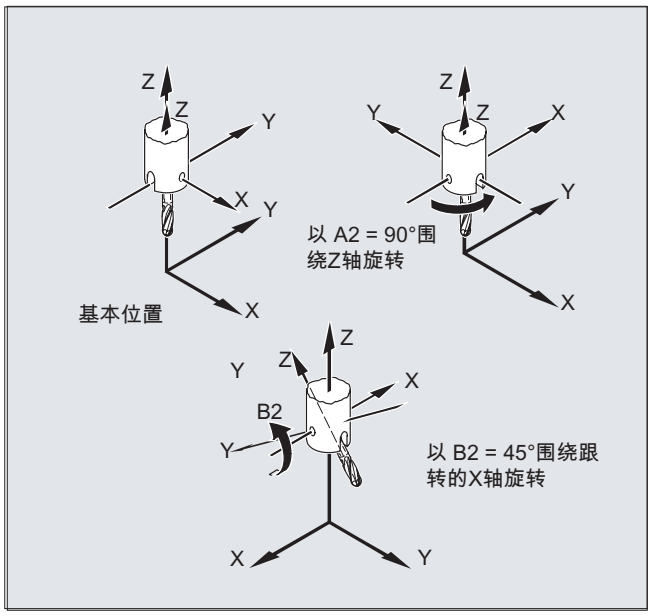
机床制造商

通过机床数据可以由机床制造商定义各种变量。请注意机床制造商说明。

编程欧拉角 ORIEULER

在使用 A2, B2, C2 进行定向编程时所编程的值被解释成为欧拉角（单位：度）。

得出定向矢量的方式：首先使用 A2 围绕 Z 轴、然后使用 B2 围绕新的 X 轴、最后使用 C2 围绕 Z 轴来使一个矢量在 Z 方向中旋转。



在这种情况下， C2 的值(围绕新 Z 轴的旋转) 没有意义，且不必进行编程。

在 RPY 角 ORIRPY 中编程

在使用 A2, B2, C2 进行定向编程时所编程的值被解释成为 RPY 角（单位：度）。

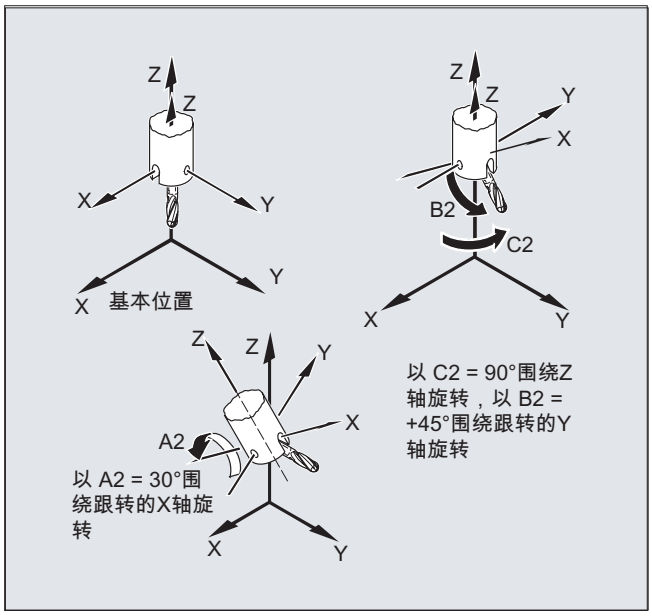
说明

与欧拉角编程相反，这里所有三个值均对定向矢量有影响。

机床制造商

在用定向角定义角时，通过 RPY 角适用于带 $MC_ORI_DEF_WITH_G_CODE = 0 的定向轴

得出定向矢量的方式：首先使用 C2 围绕 Z 轴、然后使用 B2 围绕新的 Y 轴、最后使用 A2 围绕新的 X 轴来使一个矢量在 Z 方向中旋转。

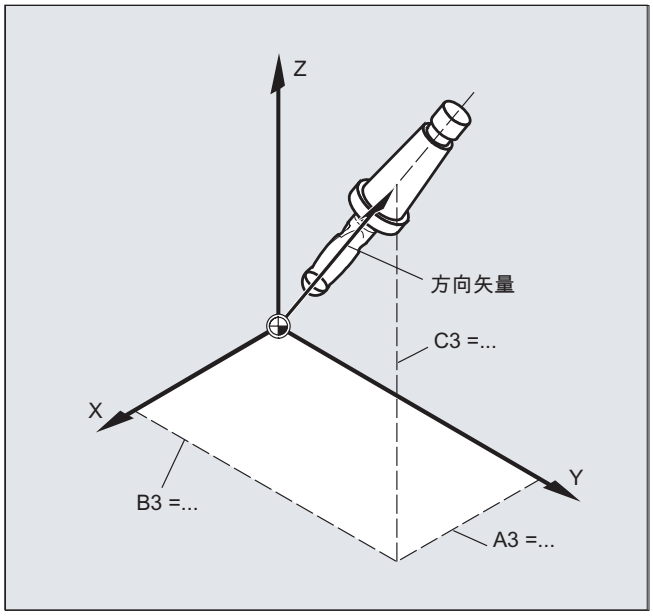


如果机床数据针对通过 G 代码 $MC_ORI_DEF_WITH_G_CODE = 1 定义定向轴，则适用于：
得出定向矢量的方式：首先使用 A2 围绕 Z 轴、然后使用 B2 围绕新的 Y 轴、最后使用 C2 围绕新的 X 轴来使一个矢量在 Z 方向中旋转。

编程方向矢量

方向矢量的分量使用 A3, B3, C3 进行编程。矢量显示在刀具安装方向；矢量的长度在此没有意义。

没有编程的矢量组成部分设置为零。



编程刀具定向，使用 LEAD= 和 TILT=

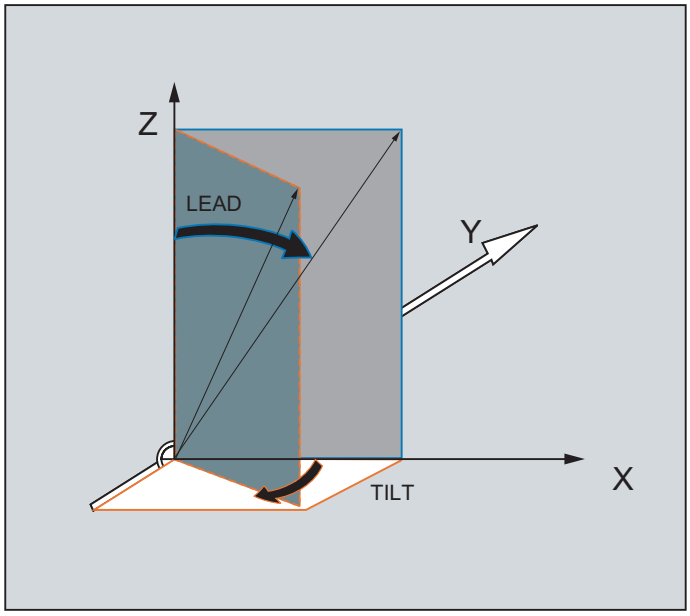
生成的刀具定向通过以下几项得出：

- 轨迹切线
- 平面法向矢量
  在程序段开始 A4, B4, C4 而在程序段结尾 A5, B6, C5
- 超前角 LEAD
  在轨迹切线和平面法线构成的平面中
- Seitwärtswinkel TILT 在程序段结尾
  垂直于轨迹切线且与平面垂线矢量相对

内角的特性（在 3D-WZK 时）

当某个内角上的程序段被缩短时，同样可在程序段结束处获得最终刀具定向。

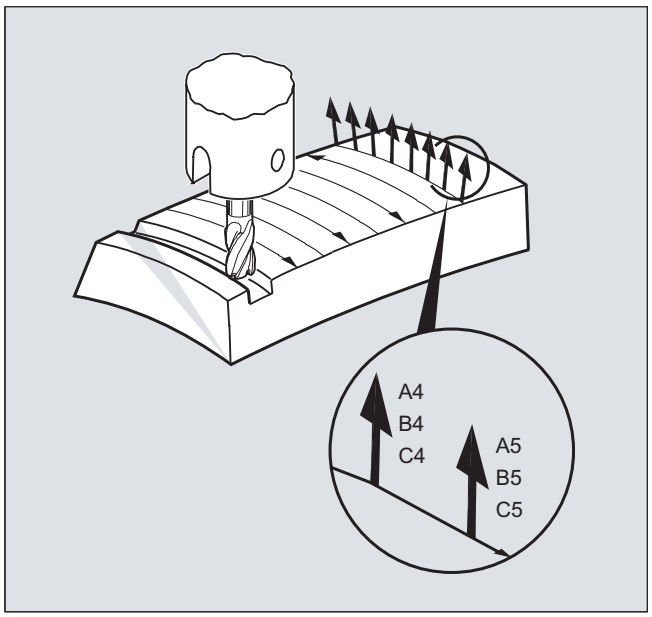
用 LEAD= 和 TILT=定义刀具定向



6.2.5 端面铣削 (3D-铣削 A4, B4, C4, A5, B5, C5)

功能

端面铣用来加工任意弯曲的表面。



对于这种 3D 铣削类型需要在工件表面积上按行描述 3D 轨迹。

计算需要考虑到刀具类型和刀具尺寸通常在 CAM 中执行。计算完成的 NC 程序段然后通过后置处理器读取到控制器。

轨迹曲率编程

面描述

轨迹弯曲通过面正交矢量用下列组成部分来描述：

A4, B4, C4 程序段开始处的起始矢量

A5, B5, C5 程序段结束处的结束矢量

如果在一个程序段中只有起始矢量，那么整个程序段的面正交矢量是恒定不变的。如果在某个程序段中仅有一个结束矢量，就会通过大圆插补从前一个程序段的终值插补到已编程的终值。

如果编程起始矢量和结束矢量，那么在这两个方向之间同样通过大圆插补来插补。因此生成连续的平滑的轨迹行程。

在基本位置中平面法向矢量指向 Z-方向，和激活的平面 G17 ～ G19 无关。

矢量的长度没有意义。

没有编程的矢量组成部分设置为零。

当激活 ORIWKS 时，参见“参照定向轴(ORIWKS, ORIMKS)”一章，平面法向矢量以激活的框架为参照并且当框架旋转时一起旋转。

机床制造商

面法向矢量必须在一个可通过机床数据设置的极限值范围内垂直于轨迹切线，否则就会发出报警。

6.2.6 定向轴的关系 (ORIWKS, ORIMKS)

功能

在通过下列方式在工件坐标系中进行方位编程时

- Euler- 以及 RPY-角，或者
- 方位矢量

可以通过 ORIMKS/ORIWKS 来设置旋转运动的轨迹。

说明

机床制造商

定向的插补类型可以由以下机床数据确定：

MD21104 $MC_ORI_IPO_WITH_G_CODE

= FALSE: 以 G 功能 ORIWKS 和 ORIMKS 为基准

= TRUE: 以第 51 组的 G 功能为基准(ORIAXES, ORIVECT, ORIPLANE, ...)

句法

```
ORIMKS=...
ORIWKS=...
```

含义

ORIMKS	在机床坐标系中旋转
ORIWKS	在工件坐标系中旋转

说明

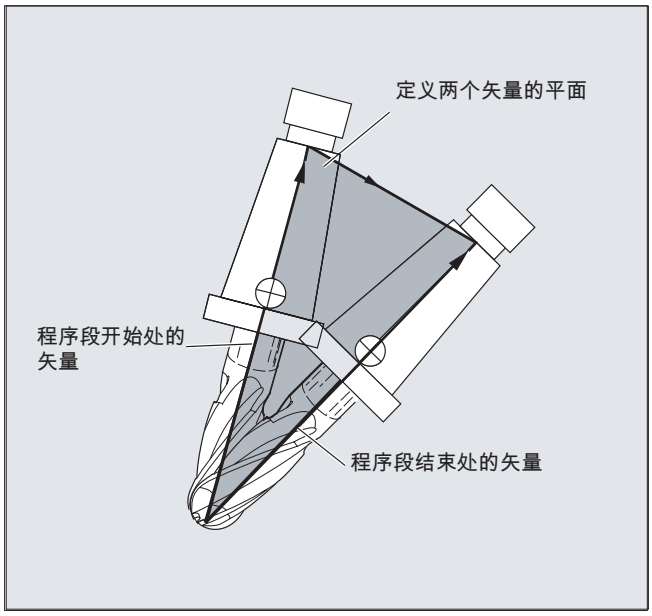
ORIWKS 是默认设置。 如果某个五轴程序从开始起就不清楚应在哪个机床上执行，原则上应选择 ORIWKS 。 机床实际上执行的运行取决于机床类型。

使用 ORIMKS 可以对实际的机床运动进行编程，以避免与装置等发生碰撞。

说明

对于 ORIMKS 而言，已执行的机床运动**取决于**机床运动机构。用空间固定的刀尖改变方向时在回转轴位置之间进行线性插补。

对于 ORIWKS 而言，已执行的机床运动**不取决于**机床运动机构。用空间固定的刀尖改变方向时刀具在由起始矢量和终点矢量展开的平面上运行。



单位置

说明

ORIWKS

在 5 轴机床单位置范围内的定向运行要求加工轴大幅度运行。(例如，当旋转式回转头将 C 轴作为旋转轴且将 A 轴作为回转轴时，所有位置均为单值 A=0。)

机床制造商

为了不使加工轴过载，速度导向将单个位置附近的轨迹速度大幅减小。

用机床数据

$MC_TRAFO5_NON_POLE_LIMIT

$MC_TRAFO5_POLE_LIMIT

能适当设定转换参数，使得极点附近的定向运动通过极点进行设定并且可以进行流畅的加工。

单位置仅使用 MD $MC_TRAFO5_POLE_LIMIT 进行处理。

文献：

/FB3/ 功能手册特殊功能；3 至 5 轴转换（F2），
章节“单位置及其处理”。

6.2.7 定位轴编程(ORIAXES, ORIVECT, ORIEULER, ORIRPY, ORIRPY2, ORIVIRT1, ORIVIRT2)

功能

定向功能描述的是刀具在空间内的定向，通过编程回转轴的偏移来实现。 可以通过刀具绕自身的额外旋转来达到较大的第三自由度。 这种刀具定向是在空间中任意通过一个第三回转轴实现，并需要 6 轴转换。 刀具绕自身的旋转取决于旋转矢量的插补方式，通过旋转角 THETA 来确定，参见章节“刀具定向旋转(ORIROTA/TR/TT, ORIROTC, THETA)”。

编程

通过轴标识符 A2, B2, C2 对定向轴编程。

`N... ORIAXES 或者 ORIVECT`	线性插补或者大圆弧插补
`N... G1 X Y Z A B C`	
或者	或者
`N... ORIPLANE`	平面的定向插补
或者	或者
`N... ORIEULER 或 ORIRPY 或 ORIRPY2`	欧拉角/RPY 角定向角
`N... G1 X Y Z A2= B2= C2=`	虚拟轴的角度编程
或者	或者
`N... ORIVIRT1 或者 ORIVIRT2`	虚拟定向轴定义 1 或 2 编程方向矢量
`N... G1 X Y Z A3= B3= C3=`	

为了沿着一个空间中的圆锥表面改变定向，可以编程定向轴的其它回转轴偏移，参见章节“沿圆锥表面编程定向(ORIPLANE, ORICONxx)”。

参数

`ORIAXES`	加工轴或者定向轴的线性插补
`ORIVECT`	大圆插补（和 `ORIPLANE` 一致）
`ORIMKS`	在机床坐标系中旋转
`ORIWKS`	在工件坐标系中旋转

	说明请参阅刀具定位的旋转章节
A= B= C=	加工轴位置编程
ORIEULER	通过欧拉角进行定向编程
ORIRPY	通过 RPY 角的定向编程 旋转顺序为 XYZ，此时： A2 为围绕 X 的旋转角 B2 为围绕 Y 的旋转角 C2 为围绕 Z 的旋转角
ORIRPY2	通过 RPY 角的定向编程 旋转顺序为 ZYX，此时： A2 为围绕 Z 的旋转角 B2 为围绕 Y 的旋转角 C2 为围绕 X 的旋转角
A2= B2= C2=	虚拟轴的角度编程
ORIVIRT1	通过虚拟定向轴的定向编程
ORIVIRT2	(定义 1)，根据 MD $MC_ORIAX_TURN_TAB_1 确定 (定义 2)，根据 MD $MC_ORIAX_TURN_TAB_2 确定
A3= B3= C3=	方向轴的方向矢量编程

说明

机床制造商

使用 MD $MC_ORI_DEF_WITH_G_CODE 可以确定如何对已编程的角 A2、B2、C2 进行定义：

根据 MD $MC_ORIENTATION_IS_EULER（默认）进行定义，或者根据 G 组 50 进行定义 (ORIEULER, ORIRPY, ORIVIRT1, ORIVIRT2)。

使用 MD $MC_ORI_IPO_WITH_G_CODE 来确定哪种插补方式有效： ORIWKS/ORIMKS 或者 ORIAXES/ORIVECT.

运行方式 JOG

定向角在这样的运行方式下总是线性插补。 在通过运行键操作的连续的，增量的运动时只能运行一个定向轴。 通过手轮可以同时运行定向轴。

对于定向轴的手动运行通道特有的进给补偿开关或者快速补偿开关在快速叠加时有效。

用下列的机床数据可以有一个单独的速度说明：

$MC_JOG_VELO_RAPID_GEO

$MC_JOG_VELO_GEO

$MC_JOG_VELO_RAPID_ORI

$MC_JOG_VELO_ORI

说明

SINUMERIK 840D “转换包处理”

使用功能“笛卡儿手动方式”可以在 JOG 运行方式下分别在参考系统 MCS、WCS 和 TCS 中以相互独立的方式分别设置几何轴转换。

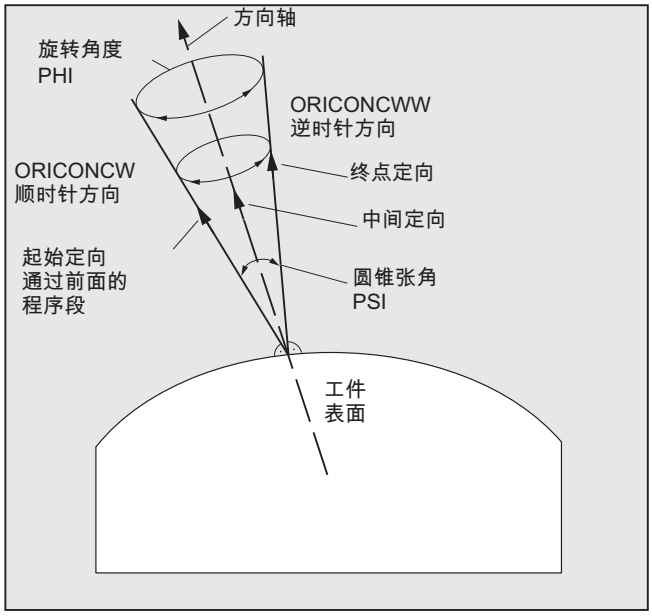
文献：

/FB2/ 功能手册 扩展功能；运动转换（M1）

6.2.8 沿一个圆锥表面定向编程(ORIPLANE, ORICONCW, ORICONCCW, ORICONTO, ORICONIO)

功能

用扩展的定向可以沿位于空间的圆锥表面执行方向改变。 在一个圆锥表面定向矢量插补可以用模态指令 ORICONxx 实现。 要在一个平面内插补，可以用 ORIPLANE 对结束定向编程。 通常起始定向通过前面的程序段确定。



编程

结束定向可以用 A2, B2, C2 以欧拉角或者 RPY 角编程角度数据，或者用 A,B,C 编程回转轴位置来确定。 对于沿圆锥表面的定向轴，必须有其它的编程数据：

- 锥体的旋转轴作为带有 A6, B6, C6 的矢量
- 张角 PSI 带有标识符 槽
- 用 A7, B7, C7 在圆锥面中间定向

说明

为圆锥的旋转轴编程方向矢量 A6, B6, C6

不一定需要对结束定向编程。 如果没有给出结束定向的数据，则整个圆锥面用 360 度插补。

用 NUT（槽）=角度编程圆锥张角

必须有结束定向的参数说明。

360 度的整圆锥面不能以这种方式插补。

在圆锥面中编程中间定向 A7,B7,C7

必须有结束定向的参数说明。 定向改变和旋转方向只能通过三个矢量来确定，即：起始定向、结束定向和中间定向。 因此三个矢量必须不同。 如果编程的中间定向平行于起始定向或结束定向，那么必须在起始矢量和结束矢量构成的平面上进行定向的直线大圆弧插补。

圆锥表面的扩展定向插补

`N... ORICONCW 或 ORICONCCW`	在圆锥面上**插补** 用
`N... A6= B6= C6= A3= B3= C3=`	圆锥的顺时针/逆时针方向矢量和结束定向或者
或者	
`N... ORICONTO`	切向过渡和
`N... G1 X Y Z A6= B6= C6=`	结束定向的参数说明
或者	或者
`N... ORICONIO`	结束定向的参数说明和
`N... G1 X Y Z A7= B7= C7=`	圆锥表面的中间定向用
`N... PO[PHI]=(a2, a3, a4, a5)`	旋转角的多项式和
`N... PO[PSI]=(b2, b3, b4, b5)`	张角多项式

参数

ORIPLANE	平面上的插补（大圆弧插补）
ORICONCW	顺时针方向圆锥表面上的插补
ORICONCCW	逆时针方向圆锥表面上的插补
ORICONTO	在切线过渡的圆锥表面上的插补
A6= B6= C6=	圆锥的旋转轴的编程（标准化的矢量）
NUT（槽）=角度	圆锥张角用度表示

NUT（槽）=+179	运行角度小于或等于 180 度
NUT（槽）=-181	运行角度大于或等于 180 度
ORICONIO	在圆锥表面插补
A7= B7= C7=	中间定向（作为标准矢量编程）
PHI	绕圆锥方向轴定向的旋转角
PSI	圆锥张角
可能的多项式 PO[PHI]=(a2, a3, a4, a5) PO[PSI]=(b2, b3, b4, b5)	除了每个角度外，多项式也最多只能编程为 5 次多项式。

定向改变的不同举例

```
…
N10 G1 X0 Y0 F5000
N20 TRAORI(1)                                ; 定向转换开。
N30 ORIVECT                                  ; 刀具定向插补为矢量。
…                                            ; 在平面中定向刀具。
N40 ORIPLANE                                 ; 选择大圆弧插补。
N50 A3=0 B3=0 C3=1
N60 A3=0 B3=1 C3=1                           ; 在 Y/Z 平面上定向绕 45 度旋转，在程序段结尾达
                                               到定向(0,1/√2,1/√2)。
…
N70 ORICONCW                                 ; 在圆锥表面定向编程：
N80 A6=0 B6=0 C6=1 A3=0 B3=0 C3=1            ; 在圆锥表面上以顺时针方向从 (0,0,1) 到定向
                                               (1/√2,0,1/√2)插补定向矢量，旋转角度达到 270
                                               度。
N90 A6=0 B6=0 C6=1                           ; 刀具定向在同一个圆锥表面上连续转了一整圈。
```

说明

如果要在空间任意存在的一个圆锥表面上描述定向改变，则刀具定向所绕的矢量必须已知。此外，还必须预先确定起始定向和结束定向。起始定向由前一程序段得出，而结束定向必须被编程或通过其它条件确定。

在平面 ORIPLANE 中按照 ORIVECT 编程

编程大圆弧插补和角度多项式必须符合轮廓的线性插补和多项式插补。刀具定向在起始定向和结束定向构成的平面中进行插补。如果附加编程多项式，则定向矢量也可以从平面中翻转。

在平面 G2/G3，CIP 和 CT 中编程圆

扩展定向符合一个平面中圆的插补。关于相应的用圆心数据或半径数据如 G2/G3 编程圆，圆通过中间点 CIP 和切向圆 CT 参见

文献： 编程手册基本原理，“编程行程指令”。

定向编程

在圆锥表面 ORICONxx 编程定向矢量插补

为了在圆锥表面定向插补，可以从 G 代码组 51 中选择四种不同的插补方式：

1. 在圆锥表面顺时针插补 ORICONCW 运用结束插补、圆锥方向或者张角数据。方向矢量用标识符 A6，B6，C6，圆锥张角用标识符 NUT（槽）= 取值范围在区间 0 到 180 度来编程。
2. 在圆锥表面逆时针插补 ORICONCWW 运用结束插补、圆锥方向或者张角数据。方向矢量用标识符 A6，B6，C6，圆锥张角用标识符 NUT（槽）= 取值范围在区间 0 到 180 度来编程。
3. 在圆锥表面插补 ORICONIO 用结束定向数据和以标识符 A7，B7，C7 编程的中间定向数据。
4. 在圆锥表面插补 ORICONTO 用切向过渡和结束定向数据。方向矢量用标识符 A6，B6，C6 来编程。

6.2.9 两个接触点的定向预设值(ORICURVE, PO[XH]=, PO[YH]=, PO[ZH]=)

功能

通过第二个空间曲线编程定向改变 ORICURVE

编程定向改变的另一个方法是，除了刀尖沿着一个空间曲线外，第二个刀具接触点的运动要用 ORICURVE 来编程。 因此如同编程刀具矢量本身，可以清楚地确定刀具定向改变。

机床制造商

请注意机床制造商对通过机床数据设置的用于刀具第 2 定向轨迹编程的轴标识符的提示。

编程

在这种插补方式下，对于这两个空间曲线用 G1 对点或者 POLY 对多项式进行编程。 圆和渐开线不允许。 此外，可以用 BSPLINE 激活样条插补和功能“合并较短样条程序段”。

文献:
/FB1/ 功能手册 基本功能；连续路径运行，准停，预读(B1),章节: 合并较短样条程序段

不允许其他的样条方式 ASPLINE 和 CSPLINE 以及用 COMPON, COMPCURV 或者 COMPCAD 激活压缩程序。

为坐标系进行定向多项式编程时，刀具两个接触点的运动可以最多可以设为 5 次。

用附加的空间曲线和多项式为坐标系扩展定向插补

```
N... ORICURVE
N... PO[XH]=(xe, x2, x3, x4, x5)
N... PO[YH]=(ye, y2, y3, y4, y5)
N... PO[ZH]=(ze, z2, z3, z4, z5)
```

刀具两个接触点的运动数据和各坐标系的附加多项式

参数

ORICURVE	带刀具两个接触点运行说明的定向插补
XH YH ZH	附加轮廓的刀具两个接触点坐标系的标识符作为空间曲线
可能的多项式 PO[XH]=(xe, x2, x3, x4, x5) PO[YH]=(ye, y2, y3, y4, y5) PO[ZH]=(ze, z2, z3, z4, z5)	除了各终点外，空间曲线还可以用多项式编程。
xe, ye, ze	空间曲线终点
xi, yi, zi	多项式系数最多 5 次

说明

标识符 XH YH ZH 用于第 2 定向轨迹的编程

选择的标识符必须与其它直线轴

X Y Z 轴

和回转轴如

A2 B2 C2 欧拉角或者 RPY 角

A3 B3 C3 方向矢量

A4 B4 C4 或者 A5 B5 C5 平面法向矢量

A6 B6 C6 旋转矢量或者 A7 B7 C7 中间点坐标

或者其它插补参数没有冲突。

6.3 定向多项式(PO[角度], PO[坐标])

功能

进行三轴到五轴转换时，与 G 代码组 1 的哪个多项式恰好激活无关，两种不同类型的定向多项式最多只能编程到 5 次。

1. 多项式用于**角度:** 超前角 LEAD、侧向角 TILT
   与起始定向与结束定向构成的平面有关。
2. 多项式用于**坐标:** 两个空间曲线的 XH, YH, ZH 用于刀具上参考点的刀具定向。

六轴转换时为了刀具定向，另外还可以用最多 5 次的多项式为刀具旋转编程旋转矢量 THT 的旋转。

句法

定向多项式类型 1 用于**角度**

```
N… PO[PHI]=(a2, a3, a4, a5)
N… PO[PSI]=(b2, b3, b4, b5)
```

三轴至五轴转换

三轴至五轴转换

定向多项式类型 2 用于**坐标**

```
N… PO[XH]=(xe, x2, x3, x4, x5)
N… PO[YH]=(ye, y2, y3, y4, y5)
N… PO[ZH]=(ze, z2, z3, z4, z5)
```

两个用于刀具定向的定向轨迹的坐标符

另外，在两种情况下可以编程一个用于**旋转**的多项式在六轴转换时用

```
N… PO[THT]=(c2, c3, c4, c5)
```

轨迹相对的旋转插补

或

```
N… PO[THT]=(d2, d3, d4, d5)
```

与定向矢量改变的绝对插补、

相对插补和切向插补。 如果该转换支持一个可用旋转角度 THETA 编程和插补的偏移，则上述编程可以实现。

含义

PO[PHI]	在起始定向和结束定向间的平面上的角度
PO[PSI]	描述起始定向和结束定向间的平面的定向倾斜角度
PO[THT]	用 THETA 编程的组 54 的 G 代码中的一个的旋转矢量旋转而成的旋转角
PHI	超前角 LEAD
PSI	侧面角 TILT
THETA	绕沿 Z 的刀具方向旋转
PO[XH]	刀具参考点的 X 坐标
PO[YH]	刀具参考点的 Y 坐标
PO[ZH]	刀具参考点的 Z 坐标

说明

不能编程定向多项式

- 如果样条插补 ASPLINE, BSPLINE, CSPLINE 有效。
  用于定向角的多项式类型 1 对于除样条外的每个插补方式都可能，即对于带快速行程 G00 或者带进给 G01 的线性插补
  对于带 POLY 的多项式插补和
  带 G02, G03, CIP, CT, INVCW 和 INCCCW 的圆插补或者
  渐开线插补。
  与次相反，用于定向坐标的多项式类型 2，仅当
  带快速行程 G00 或带进给 G01 的线性插补或者
  当带 POLY 的多项式插补有效时才可能。
- 当定向插补通过轴插补 ORIAXES。这种情况下，可以用 PO[A]和 PO[B]直接编程多项式用于轴定向 A 和 B。

带 ORIVECT、ORIPLANE 和 RICONxx 的定向多项式类型 1

对于带 ORIVECT、ORIPLANE 和 ORICONxx 的大圆弧插补和圆锥面插补，只有定向多项式类型 1 可以。

带 ORICURVE 的多项式类型 2

如果带额外空间曲线 ORICURVE 的插补有效，插补定向矢量的直角坐标分量且仅定向多项式类型 2 有效。

6.4 刀具定向旋转(ORIROTA, ORIROTR, ORIROTT, ORIROTC, THETA)

功能

如果在带运动刀具的机床类型中，要求刀具的方向也可以改变，那么每个程序段均要编程结束方向。 取决于机床类型可以编程定向轴的运动方向或者编程方向矢量 THETA 的旋转方向。 对于这些旋转矢量可以编程不同的插补类型：

- ORIROTA: 对于一个绝对规定的旋转方向的旋转角度。
- ORIROTR: 相对于起始方向和结束方向平面的旋转角度。
- ORIROTT: 相对于方向矢量改变的旋转角度。
- ORIROTC: 轨迹切线的切向旋转角。

句法

仅当插补类型 ORIROTA 已激活时，才可以按照下列四种方式对旋转角或者旋转矢量进行编程：

1. 直接旋转轴位置 A, B, C
2. 欧拉角（单位：度），通过 A2, B2, C2
3. RPY-角（单位：度），通过 A2, B2, C2
4. 方向矢量，通过 A3, B3, C3 (旋转角借助 THETA=值)

如果 ORIOTR 或者 ORIOTT 已激活，仅可直接使用 THETA 对旋转角进行编程。

在没有定向变化的情况下，也可以单独在某个程序段对旋转进行编程。 此时 ORIROTR 和 ORIROTT 没有意义。 在这种情况下，始终参照绝对方向对旋转角进行插补（ORIROTA).

N... ORIROTA	确定旋转矢量的插补
N... ORIROTR	
N... ORIROTT	
N... ORIROTC	
N... A3= B3= C3= THETA=值	确定定向矢量的旋转
N... PO[THT]=（d_2，d_3，d_4，d_5）	用 5 级多项式插补旋转角度

含义

ORIROTA	对于一个绝对规定的旋转方向的旋转角度
ORIROTR	相对于平面在起始方向和结束方向之间的旋转角度
ORIROTT	旋转角度作为到定向改变的切向旋转矢量
ORIROTC	旋转角度作为到轨迹切线的切向旋转矢量
THETA	方向矢量的旋转
THETA=值	以度表示的旋转角度，这个角度在程序段结束时达到
THETA=Θe	旋转角，带有旋转矢量的终止角 Θ_e
THETA=AC(…)	程序段方式转换到绝对尺寸说明
THETA=AC(…)	程序段方式转换到相对尺寸说明
Θe	旋转矢量的结束角度，绝对的用 G90，相对的用 G91（相对尺寸）均有效。
PO[THT]=(...)	旋转角度的多项式

定向旋转示例

程序代码	注释
N10 TRAORI	; ；激活方位转换
N20 G1 X0 Y0 Z0 F5000	; 刀具定向
N30 A3=0 B3=0 C3=1 THETA=0	; 在 Z 方向旋转角度 0
N40 A3=1 B3=0 C3=0 THETA=90	; 在 X 方向旋转 90 度
N50 A3=0 B3=1 C3=0 PO[THT]=(180,90)	; 定向
N60 A3=0 B3=1 C3=0 THETA=IC(-90)	; 在 Y 方向旋转到 180 度
N70 ORIROTT	; 保持不变且旋转到 90 度
N80 A3=1 B3=0 C3=0 THETA=30	; 旋转角度与定向改变相对
	; 旋转矢量与 X-Y 平面为 30 度

插补程序段
N40 时，旋转角度从起始值 0 度到结束值 90 度线性插补。 在程序段 N50 中，旋转角度从 90 度变为 180 度，按照
抛物线 $\Theta(u) = +90u^2$。 在 N60 中，无需改变定向也可以旋转。

当 N80 时，刀具定向从 Y 方向转到 X 方向。 从而定向改变在 X-Y 平面中，且旋转角度与该平面构成了 30 度。

说明

ORIROTA

旋转角 THETA 参照空间中的绝对设定方向进行插补。基本旋转方向通过机床数据生成。

ORIROTR

旋转角 THETA 相对于由起始和终点方位所定义的平面进行解析。

ORIROTT

旋转角 THETA 相对于方位变化进行解析。如果 THETA=0 ，旋转矢量以相对于方位变化的切向进行插补，并且只有当至少给方位编程了一个“PSI 倾斜角”多项式时，才会区别于 ORIROTR。因此生成一个不在平面上运行的方向改变。通过一个附加编程的旋转角 THETA ，就可以例如对旋转矢量进行适当插补，使其始终形成某个相对于方位变化的值。

ORIROTC

旋转矢量与轨迹切线相对用一个通过角度 THETA 编程的偏移来插补。对于角度偏移也可以编程一个多项式 PO[THT]=(c2, c3, c4, c5) ，最多 5 次。

6.5 与轨迹相对的定向

6.5.1 定向方式相对于轨迹

功能

使用扩展功能，不止在程序段结束也可以通过整个轨迹变化达到相对定向。 在前面程序段达到的定向通过大圆弧插补转到编程的结束定向。 原则上，有两种编程与轨迹相对的定向的方法：

1. 刀具定向和刀具旋转用 ORIPATH、ORPATHTS 相对于轨迹插补。
2. 按当前普遍的方式编程和插补定向矢量。 用 ORIROTC 创建与轨迹切线相对的定向矢量旋转。

句法

对定向和刀具旋转的插补方式进行编程时用：

N... ORIPATH	与轨迹相对的定向
N... ORIPATHS	轨迹相对的定向带有定向变化平整
N... ORIROTC	与轨迹相对的旋转矢量插补

由轨迹变化中的角产生的定向拐点可以用 ORIPATHS 来平整。 退刀运动的方向和行程长度可以通过带分量 A8=X, B8=Y C8=Z 的矢量来进行编程。

用 ORIPATH/ORIPATHS 能够使与轨迹切线相对的不同参照通过三个角度

- LEAD= 前置角参照于轨迹和表面的参数
- TILT= 侧向角参照于轨迹和表面的参数
- THETA= 旋转角度

为整个轨迹变换来编程。 对于旋转角度 THETA 可以用 PO[THT]=(...) 额外编程最多 5 次的多项式。

说明

机床制造商

请注意机床制造商说明。 通过设计的机床数据和设定数据，可以对与轨迹相对的定向方式进行其他设置。 进一步说明参见

文献：

/FB3/功能手册 特殊功能；3 至 5 轴转换（F2）
章节“定向”

含义

角度 LEAD 和 TILT 的插补可以通过机床数据进行不同设置：

- 在整个程序段中将始终保持以 LEAD 和 TILT 编程的刀具定向的参考量。
- 超前角 LEAD ： 绕与切线和平面垂线矢量 TILT 垂直的方向旋转: 绕平面垂线矢量旋转定向。
- 超前角 LEAD ： 绕与切线和平面垂线矢量侧向角 TILT 垂直的方向旋转: 绕轨迹切线方向旋转定向。
- 旋转角度 THETA=： 六轴转换时，刀具绕自身旋转且具有一个附加的第三回转轴作为定向轴。

说明

不允许轨迹相关的定向与 OSC、OSS、OSSE、OSD 和 OST 一起

与轨迹相对的定向插补 ORIPATH 或 ORIPATHS 与 ORIOTC 不能与用组 34 中某个 G 代码平整的定向变化一起编程。 为此 OSOF 必须激活。

6.5.2 轨迹相关的刀具定向旋转（ORIPATH、ORIPATHS、旋转角）

功能

六轴转换时为了在空间中任意定向刀具，也可以用一个第三回转轴使刀具绕自身旋转。 对于与轨迹相对的带 ORIPATH 或者 ORIPATHS 的刀具定向旋转，可以通过旋转角 THETA 编程额外的旋转。 也可以选择通过一个与刀具方向垂直的平面中的矢量来编程角度 LEAD 和 TILT。

机床制造商

请注意机床制造商说明。 通过机床数据可以设置角度 LEAD 和 TILT 的不同插补。

句法

刀具定向旋转和刀具旋转

用 ORIPATH 或者 ORIPATHS 激活与轨迹相对的刀具定向方式。

N... ORIPATH	激活参照于轨迹的定向方式
N... ORIPATHS	激活参照于轨迹的定向方式，带定向变化平滑
用旋转作用激活三种可能的角度：	
N... LEAD=	编程的与平面法线矢量相对的定向角度
N... TILT=	用于编程定向的角度，该定向在与平面法线矢量相对的轨迹切线垂直的平面中
N... THETA=	旋转角度相对于定向改变，绕第三回转轴的刀具方向

用 LEAD=值、 TILT=值或 THETA=值编程程序末端的角度值。 除恒定的角度外，可以对全部三个角度编程最多 5 次多项式。

N... PO[PHI]=(a2, a3, a4, a5)	提前角多项式 LEAD
N... PO[PSI]=(b2, b3, b4, b5)	侧向角多项式 TILT
N... PO[THT]=(d2, d3, d4, d5)	旋转角多项式 THETA

编程时，可以去掉为零的较高的多项式系数。 举例 PO[PHI]=a2 为超前角 LEAD 得出了一个抛物线。

含义

与轨迹相对的刀具定向

ORIPATH	刀具定向参照于轨迹
ORIPATHS	刀具定向参照这个轨迹，拐点在方向变化中被平滑
LEAD	相对于面正交矢量的角度，在由轨迹切线和面正交矢量展开的平面上

TILT	绕 Z 方向定向旋转或绕轨迹切线旋转
THETA	绕沿 Z 的刀具方向旋转
PO[PHI]	用于超前角 LEAD 的定向多项式
PO[PSI]	用于侧向角 TILT 的定向多项式
PO[THT]	用于旋转角 THETA 的定向多项式

说明

旋转角度 THETA

对于用第三回转轴作为定向轴的刀具绕自身的旋转，要求有一个六轴转换。

6.5.3 轨迹相关的刀具旋转插补（ORIROTC，THETA）

功能

用旋转矢量插补

对于用 ORIROTC 编程的相对于轨迹切线的刀具旋转,旋转矢量也可用一个可通过旋转角 THETA 编程的偏移量来插补。 为此，可以用 PO[THT]为该偏移角编程一个最多 5 次的多项式。

句法

`N... ORIROTC`	创建一个相对于轨迹切线的刀具旋转
`N... A3= B3= C3= THETA=值`	确定定向矢量的旋转
`N... A3= B3= C3= PO[THT]=(c2, c3, c4, c5)`	用最多 5 次的多项式插补偏移角

在没有定向变化的情况下，也可以单独在某个程序段对旋转进行编程。

含义

六轴转换时，与轨迹相对的刀具旋转插补

ORIROTC	创建到轨迹切线的切向旋转矢量
THETA=值	以度表示的旋转角度，这个角度在程序段结束时达到
THETA=θe	旋转角，带有旋转矢量的终止角 Θ_e
THETA=AC(…)	程序段方式转换到绝对尺寸说明
THETA=IC(…)	程序段方式转换到相对尺寸说明
PO[THT]=(c2, c3, c4, c5)	用 5 次多项式插补偏移角

说明

旋转矢量 ORIROTC 的插补

如果与刀具定向方向相反，也要创建一个相对于轨迹切线的刀具旋转，那么仅当六轴转换时可能。

激活 ORIROTC 时

不能编程旋转矢量 ORIROTA 。 在编程的情况下，输出报警 14128“ORIROTC 激活时刀具旋转绝对编程”。

三轴至五轴转换时刀具定位方向

可以象三轴至五轴转换时，通常用欧拉角或 RPY 角或者方向矢量来编程刀具定向方向。可以用大圆弧插补 ORIVECT、定向轴 ORIAXES 直线插补、所有圆锥面上的插补 ORICONxx 以及带刀具两个接触点空间曲线的额外插补来编程空间中的刀具定向改变。

G....	指定旋转轴的运动方式
X Y Z	指定线性轴
ORIAXES	加工轴或者定向轴的线性插补
ORIVECT	大圆插补（和 ORIPLANE 一致）
ORIMKS ORIWKS	在机床坐标系中旋转 在工件坐标系中旋转 说明请参阅刀具定位的旋转章节
A= B= C=	加工轴位置编程
ORIEULER	通过欧拉角进行定向编程

ORIRPY	通过 RPY 角的定向编程
A2= B2= C2=	虚拟轴的角度编程
ORIVIRT1 ORIVIRT2	通过虚拟定向轴的定向编程 (定义 1)，根据 MD $MC_ORIAX_TURN_TAB_1 确定 (定义 2), 根据 MD $MC_ORIAX_TURN_TAB_2 确定
A3= B3= C3=	方向轴的方向矢量编程
ORIPLANE	平面上的插补（大圆插补）
ORICONCW	顺时针方向圆锥表面上的插补
ORICONCCW	逆时针方向圆锥表面上的插补
ORICONTO	在切线过渡的圆锥表面上的插补
A6= B6= C6=	圆锥的旋转轴的编程（标准化的矢量）
NUT（槽）=角度	圆锥张角用度表示
NUT（槽）=+179	运行角度小于或等于 180 度
NUT（槽）=-181	运行角度大于或等于 180 度
ORICONIO	在圆锥表面插补
A7= B7= C7=	中间定向（作为标准矢量编程）
ORICURVE XH YH ZH 例如带有多项式 PO[XH]=(xe, x2, x3, x4, x5)	带刀具两个接触点运行说明的定向插补 除了相应的终点之外，其它空间曲线多项式也可以编程。

说明

如果用激活的 ORIAXES 通过定向轴对刀具定向进行插补，则只能在程序结束创建与轨迹相对的旋转角。

6.5.4 平滑定向变化(ORIPATHS A8=, B8=, C8=)

功能

当在轮廓上以恒定加速度改变定向时，不希望出现轨迹运动中断，尤其是在轮廓拐角处。可以通过插入一个自身中间程序段来平整在定向变化中由此而产生的拐点。 此后定向改变加速度恒定，如果在换向时 ORIPATHS 也激活了。 在该阶段执行一个刀具退刀。

机床制造商

请注意机床制造商关于激活该功能的预定义机床数据和设定数据的说明。

如何解释退刀矢量可以通过机床数据设置：

1. 在刀具坐标系中，通过刀具方向定义 Z 坐标。

2. 在工件坐标系中，通过有效平面定义 Z 坐标。

关于功能“轨迹相对的定向”的进一步说明参见
文献： /FB3/ 功能手册 特殊功能；3 至 5 轴转换（F2）

句法

对于以整个轨迹为参照的恒定的刀具定向，在轮廓拐角处要求有进一步的编程数据。 该运动的方向和行程长度可以通过带分量 A8=X、B8=Y、 C8=Z 的矢量来进行编程。

```
N... ORIPATHS A8=X B8=Y C8=Z
```

含义

ORIPATHS	刀具定向参照这个轨迹，拐点在方向变化中被平滑。
A8= B8= C8=	用于方向和行程长度的矢量分量
X, Y, Z	刀具方向上的退刀运动

说明

编程方向矢量 A8, B8, C8

如果该矢量长度为零，则无法进行退刀。

ORIPATHS

用 ORIPATHS 激活以轨迹为参照的刀具定向。 否则，定向通过线性大圆弧插补从起始定向变为结束定向。

6.6 定向压缩（COMPON，COMPCURV，COMPCAD）

功能

在遵守规定的容差前提下，可以对 NC 程序进行压缩，在 NC 程序中定向传输（TRAORI）激活且定向通过方向矢量编程。

说明

定向运动仅当大圆弧插补激活时可以压缩，且定向运动取决于定向插补的 G 代码。 这如同最大行程长度以及每个轴允许的容差或者压缩器功能的轨迹进给量允许的容差一样，可以通过机床数据设置。 请注意机床制造商说明。

编程

刀具定向

如果一个定向转换（TRAORI）处于激活状态，对于 5 轴机床可以如下措施（与运动无关）对刀具定向编程：

- 通过以下参数对方向**矢量**编程：

  A3=<...> B3=<...> C3=<...>

- 通过以下参数编程欧拉**角**或者 RPY **角**：

  A2=<...> B2=<...> C2=<...>

刀具旋转

如果是 **6 轴** 机床，除了刀具定向之外，还可以对刀具的旋转进行编程。

通过以下参数实现旋转角编程：

THETA=<...>

参见“刀具定向旋转 (页 358)”。

说明

在其中已经附带编程有一个旋转的 NC 程序段只有在旋转角以**线性**变化时才可以被压缩，也就是说，不允许使用 PO[THT]=(...) 给旋转角编程多项式。

NC 程序段的通常形式

因此可压缩的 NC 程序段的一般形式可以如下所述：

```
N... X=<...> Y=<...> Z=<...> A3=<...> B3=<...> C3=<...> THETA=<...> F=<...>
```

或者

```
N... X=<...> Y=<...> Z=<...> A2=<...> B2=<...> C2=<...> THETA=<...> F=<...>
```

说明

位置值可以直接标注（如：X90）或者通过参数赋值（例如 X=R1*(R2+R3)）间接标注。

通过回转轴位置进行刀具定向编程

刀具定向也可以通过回转轴位置来说明时，例如形式为：

```
N... X=<...> Y=<...> Z=<...> A=<...> B=<...> C=<...> THETA=<...> F=<...>
```

在这种情况下以两种不同的方式进行压缩，它由是否进行大圆插补而定。 在没有大圆插补的情况下，一般通过回转轴的轴向多项式来描述已压缩的定向变化。

轮廓精度

根据设定的压缩模式（MD20482 $MC_COMPRESSOR_MODE），对于几何轴和定向轴，在压缩时要么设计的轴专用的公差（MD33100 $MA_COMPRESS_POS_TOL）有效，要么下列可通过设置数据设定的通道专用的公差有效：

SD42475 $SC_COMPRESS_CONTUR_TOL（最大轮廓偏差）

SD42476 $SC_COMPRESS_ORI_TOL（刀具定向最大角度偏差）

SD42477 $SC_COMPRESS_ORI_ROT_TOL（用于刀具旋转角的最大角度偏差）（仅对于 6 轴机床可用）

文献：

功能手册 基本功能；3 至 5 轴转换（F2），
章节：“定向压缩”

激活/取消激活

通过模态 G 代码 COMPON、COMPCURV 或者 COMPCAD 接通压缩器功能。

通过 COMPOF 退出压缩器功能。

参见“NC 程序段压缩（COMPON，COMPCURV，COMPCAD） (页 257)”。

说明

仅当大圆插补已激活时，才会对定向运动进行压缩（也就是说，在从起点和终点展开的平面中改变刀具定向）。

大圆插补在以下条件下进行：

- MD21104 $MC_ORI_IPO_WITH_G_CODE = 0，
  ORIWKS 已激活且
  通过矢量对定向编程（通过 A3、B3、C3 或者 A2、B2、C2）。
- MD21104 $MC_ORI_IPO_WITH_G_CODE = 1 且
  ORIVECT 或 ORIPLANE 激活。
  刀具方向可以作为方向矢量编程，或者使用回转轴位置编程。 如果 G 代码 ORICONxx 或者 ORICURVE 中一个激活或者定向角多项式（PO[PHI] 和 PO[PSI]）已编程，则不进行大圆补插补。

示例

在下面的程序示例中，压缩一条用一个多边形轮廓逼近的圆弧。 这里刀具方向与一个圆锥面同步。 尽管几个编程的方向改变并不恒定，但是压缩器功能却生成一个平滑的曲线。

编程	注释
DEF INT ANZAHL=60	
DEF REAL RADIUS=20	
DEF INT COUNTER	
DEF REAL WINKEL	
N10 G1 X0 Y0 F5000 G64	
$SC_COMPRESS_CONTUR_TOL=0.05	; 轮廓的最大偏差= 0.05 mm
$SC_COMPRESS_ORI_TOL=5	; 最大定向偏差 = 5 度
TRAORI	
COMPCURV	
	; 跟踪一个从多边形形成的圆。 此时在一个围绕 Z、张角为 45 度的圆锥上作定向运动。
N100 X0 Y0 A3=0 B3=-1 C3=1	
N110 FOR COUNTER=0 TO ANZAHL	
N120 WINKEL=360*COUNTER/ANZAHL	
N130 X=RADIUS*cos(WINKEL) Y=RADIUS*sin(WINKEL)	
A3=sin(WINKEL) B3=-cos(WINKEL) C3=1	
N140 ENDFOR	

6.7 定向曲线的平滑(ORISON, ORISOF)

功能

使用功能“定向曲线的平滑(ORISON)”可以通过多个程序段对方向变化进行平滑。 从而获得一个比较平稳的定向曲线以及轮廓。

前提条件

只有在带 5/6 轴坐标转换的系统中才提供“定向曲线的平滑(ORISON)”功能。

句法

```
ORISON
...
ORISOF
```

含义

ORISON:	启用定向曲线的平滑	
	有效性：	模态
ORISOF:	取消定向曲线的平滑	
	有效性：	模态

设定数据

应在满足以下条件时平滑定向曲线：

- 在规定的最大公差内（刀具定向的最大角度偏差，单位度）

  以及

- 在规定的最大位移内。

这些规定值通过以下设定数据定义：

- SD42678 $SC_ORISON_TOL （用于定向曲线平滑的公差）
- SD42680 O$SC_ORISON_DIST （用于定向曲线平滑的位移）

示例

程序代码	注释
...	
TRAORI()	; 启用定向转换。
ORISON	; 启用定向平滑。
$SC_ORISON_TOL=1.0	; 定向平滑的公差 = 1.0 度。
G91	
X10 A3=1 B3=0 C3=1	
X10 A3=-1 B3=0 C3=1	
X10 A3=1 B3=0 C3=1	
X10 A3=-1 B3=0 C3=1	
X10 A3=1 B3=0 C3=1	
X10 A3=-1 B3=0 C3=1	
X10 A3=1 B3=0 C3=1	
X10 A3=-1 B3=0 C3=1	
X10 A3=1 B3=0 C3=1	
X10 A3=-1 B3=0 C3=1	
...	
ORISOF	; 关闭定向平滑。
...	

轴方向在 XZ 平面内偏转 90 度，即从 -45 度到 +45 度。 通过定向曲线的平滑，轴方向不再达到最大角度值 - 45 度或 +45 度。

其它信息

程序段的数量

通过定义数量的若干程序段可以对定向曲线进行平滑，该数量由 MD28590 $MC_MM_ORISON_BLOCKS 定义。

说明

如果以 ORISON 激活了定向曲线平滑，而没有为此设计充足的程序段存储器(MD28590 < 4)，则发出报警提示，并拒绝执行功能。

最大的程序段位移

只有当程序段中的运行位移小于定义的最大程序段位移长度时，即 MD20178 $MC_ORISON_BLOCK_PATH_LIMIT，才能在其中进行定向曲线的平滑。 超出位移限制的程序段会中断平滑过程，轴按照编程的指令退回。

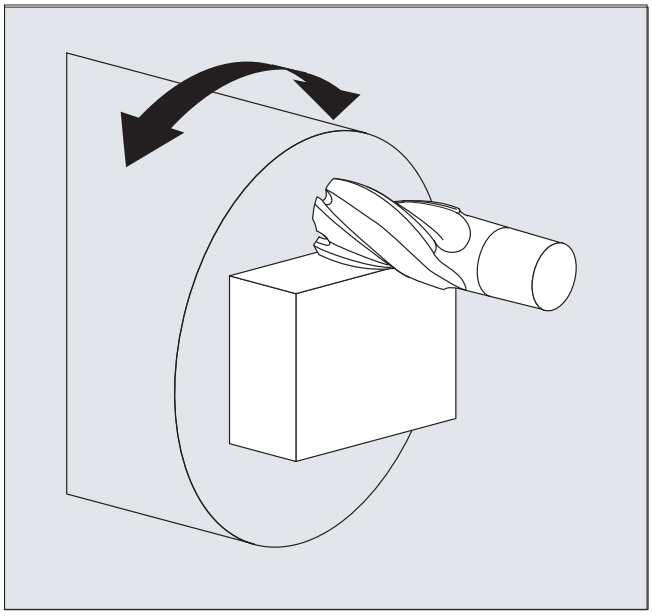
6.8 运动变换

6.8.1 铣削车削件(TRANSMIT)

功能

功能 TRANSMIT 可以用于以下操作：

- 在车削中的车削件的端面加工（钻孔，轮廓）。
- 对于加工编程可以使用一个直角坐标系。
- 控制系统将编程设计的直角坐标系的运行转换成实际的加工轴的运行（标准情况）：
  - 回转轴
  - 垂直于回转轴的横向进给轴
  - 纵轴平行于旋转轴
  - 线性轴互相垂直。
- 运行相对于旋转中心的刀具中心偏移。
- 速度导向考虑到为旋转运行所定义的限制。



TRANSMIT 转换类型

TRANSMIT 加工有下列可以设置的特征：

- TRANSMIT 在默认情况下带有 (TRAFO_TYPE_n = 256)
- TRANSMIT 带有辅助线性轴 Y (TRAFO_TYPE_n = 257)

扩展转换类型 257 可以用来使用实轴 Y 对刀具的装夹进行补偿。

句法

TRANSMIT 或者 TRANSMIT(n)

TRAFOOF

回转轴

不能编程回转轴，因为它们被一个几何轴覆盖并且因此作为通道轴不能直接编程。

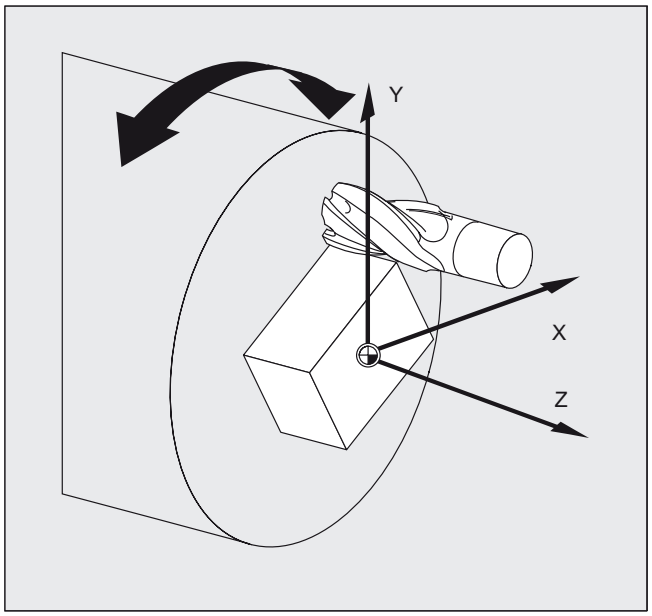
含义

TRANSMIT:	激活第一个约定的 TRANSMIT 功能。 该功能也被称为极转换。
TRANSMIT(n):	激活第 n 个约定的 TRANSMIT 功能；n 最大可以是 2(TRANSMIT(1)和 TRANSMIT 相符)。
TRAFOOF:	关闭一个当前有效的转换
OFFN:	偏移 轮廓－标准： 端面加工与已编程参考轮廓的间距

说明

如果在相应的通道中激活其余转换中的某一个转换，则同样会有一个转换 TRANSMIT 会被中断（例如 TRACYL, TRAANG, TRAORI).

示例



程序代码	注释
N10 T1 D1 G54 G17 G90 F5000 G94	; 刀具选择
N20 G0 X20 Z10 SPOS=45	; 接近起始位置
N30 TRANSMIT	; 激活 TRANSMIT-功能
N40 ROT RPL=-45	; 设置框架
N50 ATRANS X-2 Y10	
N60 G1 X10 Y-10 G41 OFFN=1OFFN	; 粗加工四方形；加工余量 1 mm
N70 X-10	
N80 Y10	
N90 X10	
N100 Y-10	
N110 G0 Z20 G40 OFFN=0	; 换刀
N120 T2 D1 X15 Y-15	
N130 Z10 G41	
N140 G1 X10 Y-10	; 精加工四边形
N150 X-10	
N160 Y10	
N170 X10	
N180 Y-10	
N190 Z20 G40	; 撤销框架选择
N200 TRANS	
N210 TRAFOOF	
N220 G0 X20 Z10 SPOS=45	; 接近起始位置
N230 M30	

说明

极点

有两种方式用于通过极点：

- 线性轴单独运行
- 以极点的回转轴旋转运行到极点并且再从极点运行出来

通过 MD 24911 和 24951 来选择。

TRANSMIT 带有辅助线性轴 Y（转换类型 257）：

如果是带有另一个线性轴的机床，极点转换的该转换型式可充分利用冗余度来进行较好的刀具补偿。对于第二个线性轴，则适用：

- 较小的工作范围，并且
- 第二个线性轴不应用于退出零件程序。

确定机床数据设置是零件程序和在 BKS 或者 MKS 中安排相应轴的前提条件，参见

参考文献

/FB2/ 功能手册扩展功能；运动转换（M1）

6.8.2 圆柱形外壳转换（TRACYL）

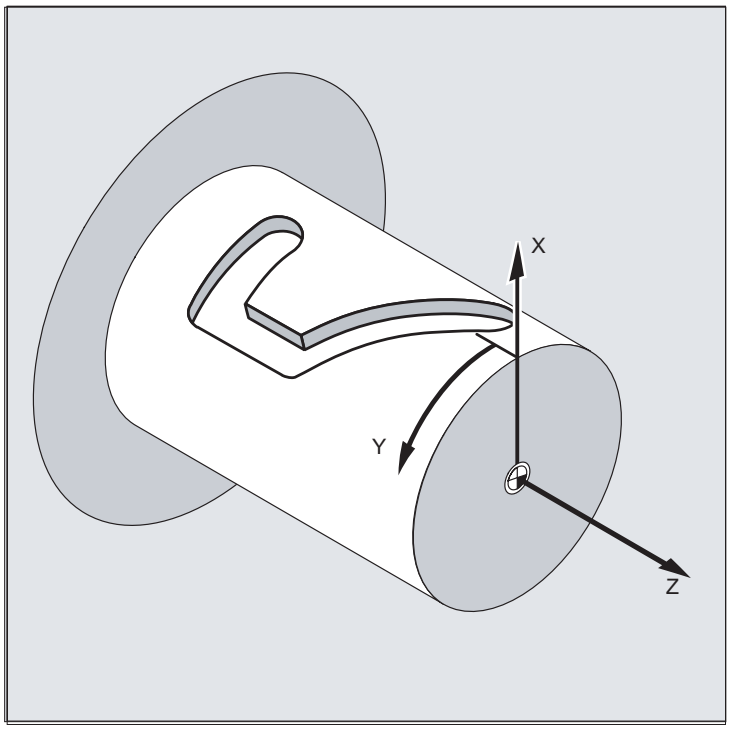
功能

圆柱表面曲线转换 TRACYL 可以用于以下操作：

加工：

- 圆柱体上的纵向槽，
- 圆柱体上的横向槽，
- 圆柱体上任意运行的槽。

槽的运行要根据展开的平面的圆柱表面来编程。



TRACYL 转换类型

圆柱面坐标转换有三个特征：

- TRACYL 没有槽壁补偿： (TRAFO_TYPE_n=512)
- TRACYL 有槽壁补偿： (TRAFO_TYPE_n=513)
- TRACYL 有辅助线性轴和槽壁补偿： (TRAFO_TYPE_n=514)
  使用 TRACYL 通过第三个参数对槽壁补偿进行参数设定。

当使用槽壁补偿进行圆柱面曲线转换时，用于补偿的轴应当位于零点 (y=0)，以便沿着已编程的槽中心线对槽进行加工。

轴使用

下列轴不可以作为定位轴或者摆动轴使用：

- 几何轴沿圆柱表面（Y 轴）的圆周方向
- 附加的线性轴在槽壁补偿时（Z 轴）

句法

TRACYL(d) 或者 TRACYL(d, n) 或者

用于转换类型 514

TRACYL(d, n, 槽壁补偿)

TRAFOOF

回转轴

不能编程回转轴，因为它们被一个几何轴覆盖并且因此作为通道轴不能直接编程。

含义

TRACYL (d)	激活第一个在通道机床数据中约定的 TRACYL 功能。d 参数用于加工直径。
TRACYL(d, n)	激活第 n 个在通道机床数据中约定的 TRACYL 功能。n 最大值可为 2，TRACYL(d,1) 相当于 TRACYL(d)。
D	加工直径的值。 加工直径是刀尖和旋转中心之间的两倍距离。 始终必须设定该直径且必须大于 1。
n	可选用的第 2 个参数，用于 TRACYL 数据记录 1 (临时选择) 或者 2。
槽壁补偿	可选用的第 3 个参数，其用于 TRACYL 的值临时从机床数据的模式中选择。 值范围： 0: 转换类型 514，没有如目前为止的槽壁补偿 1: 转换类型 514，有槽壁补偿
TRAFOOF	转换 关（BKS 和 MKS 再次一致）。
OFFN	偏移 轮廓－标准： 到编程设计的基准轮廓的槽壁的距离

说明

如果在相应的通道中激活其余转换中的某一个转换，则同样会有一个转换 TRACYL 会被中断（例如 TRANSMIT, TRAANG, TRAORI).

示例： 刀具定义

下列示例适用于对圆柱转换 TRACYL 的参数设定进行测试：

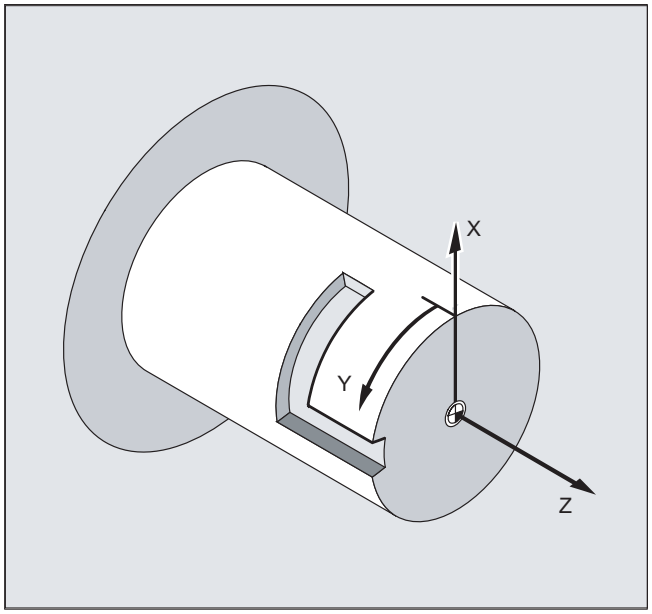
程序代码	注释	
刀具参数 编号(DP)	含义	附注
$TC_DP1[1,1]=120	刀具类型	铣刀
$TC_DP2[1,1]=0	刀沿位置	仅用于车刀

程序代码	注释	
几何尺寸	长度补偿	
$TC_DP3[1,1]=8.	长度补偿矢量	根据类型计算
$TC_DP4[1,1]=9.		和平面
$TC_DP5[1,1]=7.		

程序代码	注释	
几何尺寸	半径	
$TC_DP6[1,1]=6.	半径	刀具半径
$TC_DP7[1,1]=0	切槽锯片的槽宽 b，铣削刀具的倒圆半径	
$TC_DP8[1,1]=0	超出规定范围 k	只针对切槽锯片
$TC_DP9[1,1]=0		
$TC_DP10[1,1]=0		
$TC_DP11[1,1]=0	圆锥形铣削刀具角度	

程序代码	注释	
磨损	长度和半径补偿	
$TC_DP12[1,1]=0	剩余参数直到$TC_DP24=0	基本尺寸/适配器

示例： 加工一个钩形槽



打开圆柱体表面转换：

程序代码	注释
N10 T1 D1 G54 G90 F5000 G94	; 刀具选择，装夹补偿
N20 SPOS=0	; 接近起始位置
N30 G0 X25 Y0 Z105 CC=200	
N40 TRACYL (40)	; 启动：圆柱面曲线转换
N50 G19	; 选择平面

加工钩形槽

程序代码	注释
N60 G1 X20	; 将刀具向槽底进给
N70 OFFN=12	; 确定相对于槽中心线的槽壁间距 12 mm
N80 G1 Z100 G42	; 向右侧槽壁运动
N90 G1 Z50	; 槽截面平行于圆柱轴线
N100 G1 Y10	; 槽截面平行于周边
N110 OFFN=4 G42	; 向左侧槽壁运动；确定相对于槽中心线的槽壁间距 4 mm
N120 G1 Y70	; 槽截面平行于周边
N130 G1 Z100	; 槽截面平行于圆柱轴线
N140 G1 Z105 G40	; 离开槽壁
N150 G1 X25	; 空运行

程序代码	注释
N160 TRAFOOF	
N170 G0 X25 Y0 Z105 CC=200	; 接近起始位置
N180 M30	

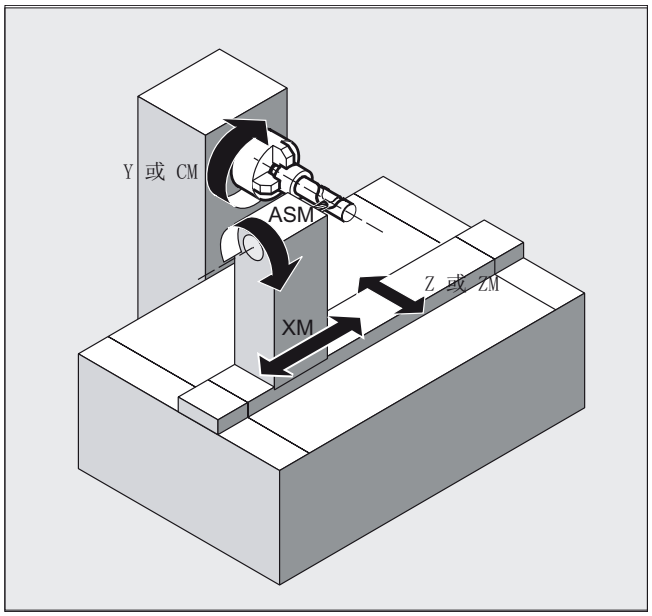
说明

没有槽壁补偿 (转换类型 512):

控制系统将编程设计的圆柱坐标系的运行转换成实际的加工轴的运行：

- 回转轴
- 垂直于回转轴的横向进给轴
- 纵轴平行于旋转轴

线性轴互相垂直。横向进给轴与回转轴相交。

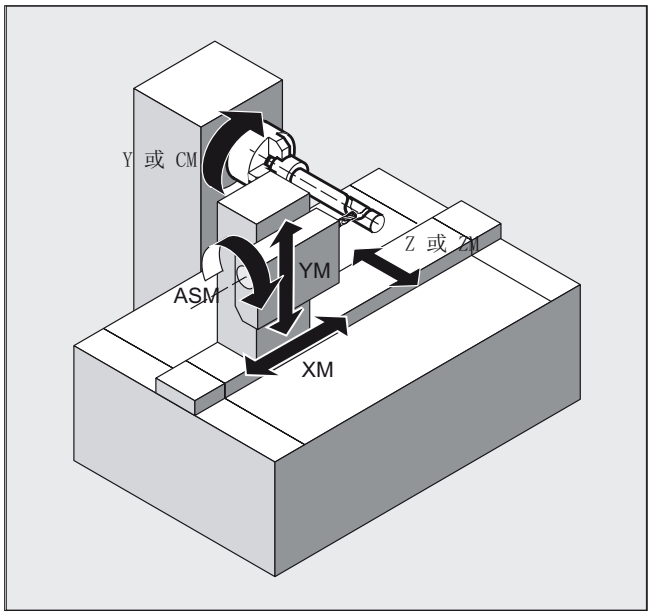


有槽壁补偿 (转换类型 513):

运动同上，但是纵轴平行于圆周方向

线性轴互相垂直。

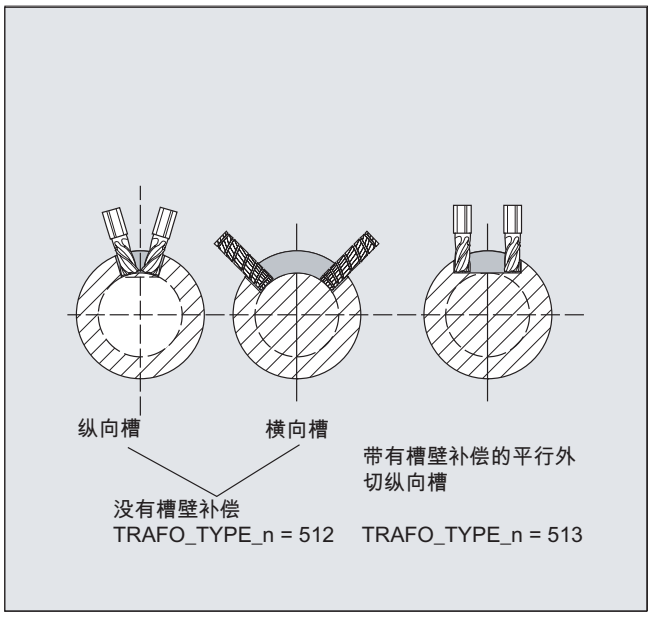
速度导向考虑到为旋转运行所定义的限制。



槽横截面

如果槽宽正好和刀具半径相符，那么在轴配置 1 时，与回转轴成纵向的槽只是平行的被限制。

与圆周平行的槽（横向槽）在开始和结束时不平行。



有辅助线性轴和槽壁补偿 (转换类型 514):

如果是带有另一个线性轴的机床，该转换方案可充分利用冗余度来进行较好的刀具补偿。对于第二个线性轴，则适用：

- 较小的工作范围，并且
- 第二个线性轴不应用于退出零件程序。

确定机床数据设置是零件程序和在 BKS 或者 MKS 中安排相应轴的前提条件，参见

参考文献

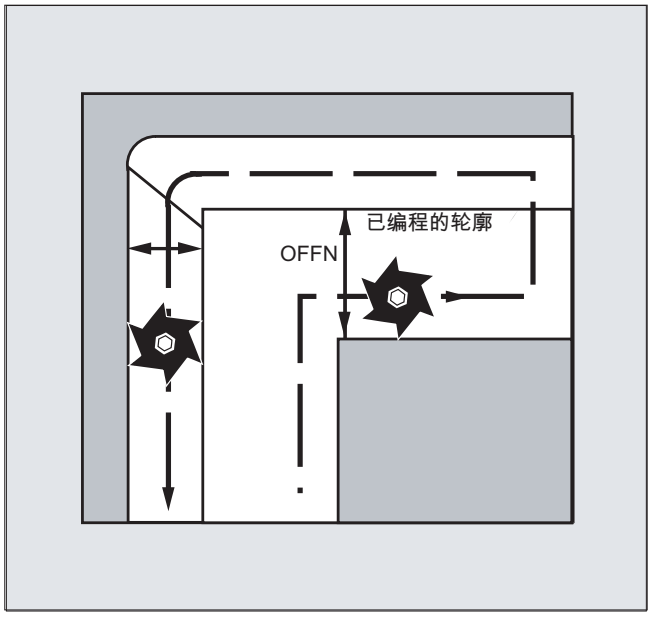
/FB2/ 功能手册扩展功能；运动转换（M1）

正常轮廓偏置 OFFN (转换类型 513)

为了使用 TRACYL 铣削槽，应在

- 零件程序中,
- 通过 OFFN **半个槽宽度** 对槽中心线进行编程。

OFFN 只有与选中的刀具半径补偿相配合才有效，以防止损伤槽壁).**此外，应使 OFFN>= 刀具半径，以防止损伤对面的槽壁。**



铣削一个槽的零件程序通常由以下几个步骤组成：

1. 选择刀具
2. 选择 TRACYL
3. 选择合适的坐标偏移（框架）
4. 定位
5. 编程 OFFN

6. 选择 WRK
7. 返回程序段（运行到 WRK 并且返回槽壁）
8. 槽中心线的轮廓
9. 取消 WRK
10. 退刀程序段（离开 WRK 并且离开槽壁）
11. 定位
12. TRAFOOF
13. 再次选择原始的坐标偏移（框架）

特点

- 选择 WRK：

  WRK 不是根据槽壁，而是相对于编程设计的槽中心线来编程。为了使刀具在槽壁左侧运行，要输入 G42（代替 G41）。如果在 OFFN 中输入带有符号的槽宽度，您就可避免这种情况。

- OFFN 与 TRACYL 配合使用的作用与没有 TRACYL 的作用不同。因为当 WRK 激活时， OFFN 也可在没有 TRACYL 的情况下被考虑在内，所以 OFFN 应在 TRAFOOF 之后重新置为零。
- 可以在零件程序内修改 OFFN 。因此槽中心线可以从中心偏移（见图）。
- 导向槽：

  如果是导向槽，使用 TRACYL 就不会生成如同使用刀具直径等于槽宽度的刀具所加工出来的槽。原则上不可能用一个较小的圆柱形刀具生成同一个槽壁几何尺寸，用较大的刀具也不行。TRACYL 可减少误差。为了不出现精确性问题，刀具半径只能略小于半槽宽。

说明

OFFN 和 WRK

当 TRAFO_TYPE_n = 512 时， OFFN 项下的值就作为相对于 WRK 的加工余量。

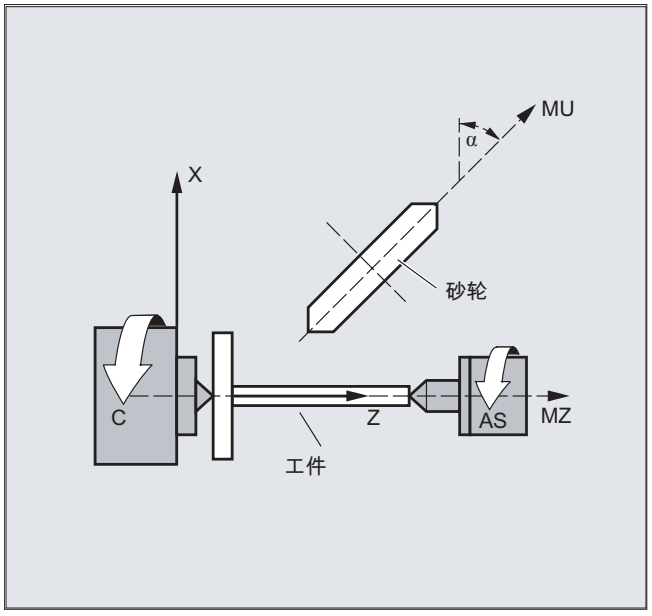
当 TRAFO_TYPE_n = 513 时，在 OFFN 中编程半个槽宽度。轮廓用 OFFN-WRK 开始运行。

6.8.3 斜置轴 (TRAANG)

功能

功能斜置轴是为磨削技术而考虑的并且适用于以下操作：

- 用斜置的进给轴加工
- 对于编程可以使用一个直角坐标系。
- 控制系统将编程设计的直角坐标系的运行转换成实际的加工轴的运行(标准情况): 斜置的进给轴。



句法

```
TRAANG(α) 或者 TRAANG(α, n)
TRAFOOF
```

含义

TRAANG( )或者 TRAANG(,n)	用前面选择的参数激活转换
TRAANG(α)	激活第一次约定的斜置轴转换
TRAANG(α,n)	激活第 n 个约定的斜置轴转换。n 最大可以是 2。 TRAANG(α,1) 相当于 TRAANG(α).

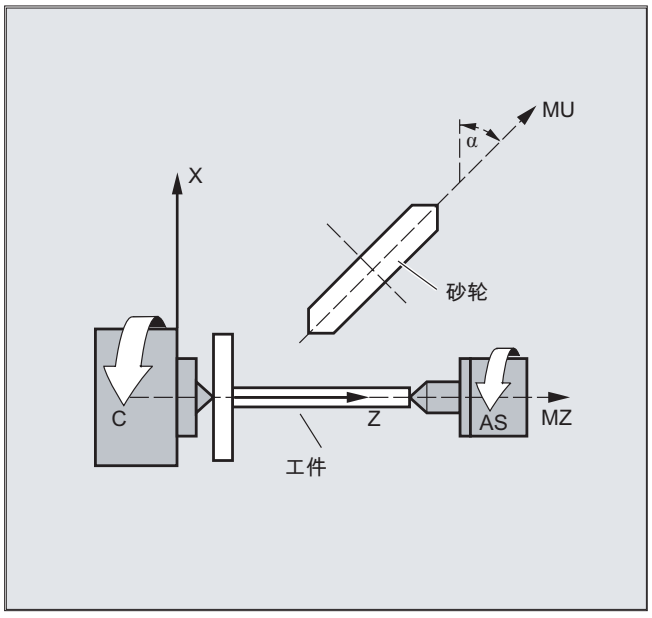
αA	斜置轴的角度 α 的允许值为： -90 度< α < + 90 度
TRAFOOF	转换 关
n	已约定转换的数量

角 α 被省略或者为零

如果角 α 被省略（例如 TRAANG()，TRAANG(,n)），就会使用之前选中的参数设定来激活转换。 在第一次选择的时预设置按照机床数据。

角 α = 0（例如 TRAANG(0)，TRAANG(0,n))是合法的参数设定，并且不再相当于旧版本中的省略参数。

示例



程序代码	注释
N10 G0 G90 Z0 MU=10 G54 F5000 ->	; 刀具选择，装夹补偿，
-> G18 G64 T1 D1	选择平面
N20 TRAANG(45)	; 启动倾斜轴转换
N30 G0 Z10 X5	; 接近起始位置
N40 WAITP(Z)	; 使轴摆动
N50 OSP[Z]=10 OSP2[Z]=5 OST1[Z]=-2 ->	; 摆动，直至达到尺寸为止

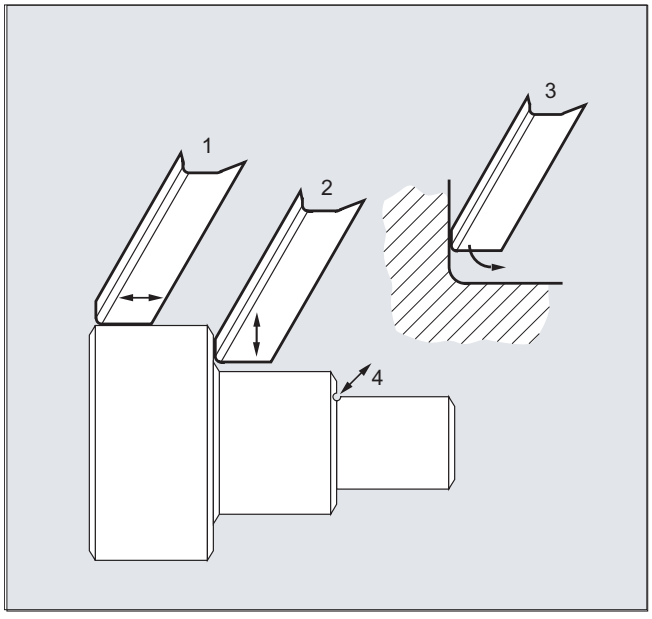
程序代码	注释
-> OST2[Z]=-2 FA[Z]=5000	（摆动请参阅"摆动"一章）
N60 OS[Z]=1	
N70 POS[X]=4.5 FA[X]=50	
N80 OS[Z]=0	
N90 WAITP(Z)	; 使摆动轴成为定位轴;
N100 TRAFOOF	; 取消转换
N110 G0 Z10 MU=10	; 空运行
N120 M30	;

-> 在一个程序段中编程

描述

可以有下列加工：

1. 纵向磨削
2. 平面磨削
3. 磨削一个特定的轮廓
4. 斜向切入式磨削



机床制造商

以下设置通过机床数据来确定：

- 一个加工轴和斜置轴之间的角度，
- 以"斜置轴"功能所确定的坐标系原点为基准的刀具零点位置，

- 在平行轴上为补偿运动预留的速度余量，
- 在平行轴上为补偿运动预留的加速度余量。

轴配置

为了能够在直角坐标系中进行编程，必须让控制系统知道该坐标系与实际存在的加工轴 (MU, MZ) 之间的关系：

- 几何轴的命名
- 几何轴分配给通道轴
  - 一般情况 (斜置轴没有激活)
  - 斜置轴激活
- 通道轴分配加工轴编号
- 主轴标记
- 加工轴名称的赋值

操作和标准轴配置的操作相符，“斜置轴有效”例外。

6.8.4 编程斜置轴 (G05, G07)

功能

在 JOG 运行方式中，砂轮可以在直角坐标系中运动，或者沿着倾斜轴的方向运动（仍然以直角坐标显示）。 只有实际的 U 轴在运动，更新 Z 轴的显示。

重新定位偏移必须以手动方式直角返回。

当激活
“PTP 运行”时，在 JOG 工作状态中对超过直角坐标系加工范围极限的运动进行监控，相应的轴会提前制动。 如果“PTP 运动”未激活，轴可以精确运动到加工范围极限处。

文献

/FB2/ 功能手册 扩展功能；运动转换（M1）

句法

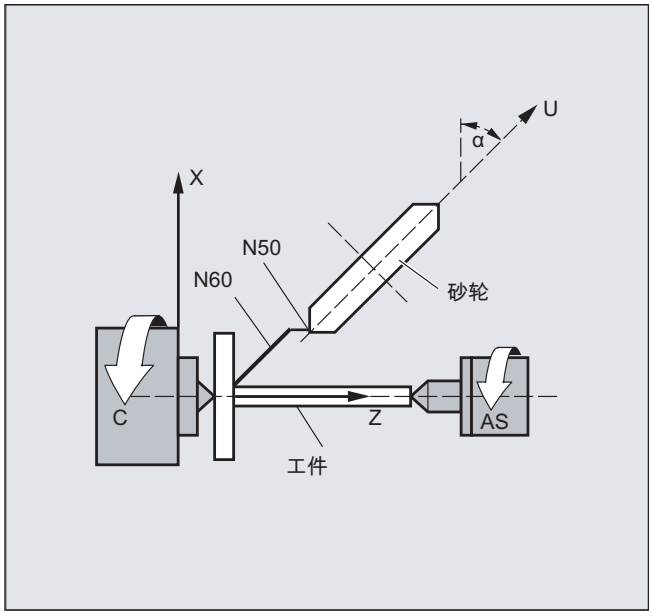
```
G07
G05
```

指令 G07/G05 可用来减轻对斜置轴的编程工作。 对此可以编程并且显示在直角坐标系中的位置。 刀具补偿和零点偏移进行直角计算。 在对 NC 程序中斜置轴的角度进行编程之后，可向起始位置运动 (G07) 且接着执行斜向切入 (G05) 。

含义

G07	返回起始位置
G05	激活斜置切入

示例



编程	注释
N.. G18	; 编程倾斜轴的角
N50 G07 X70 Z40 F4000	; 返回起始位置
N60 G05 X70 F100	; 斜向切入
N70 ...	;

6.9 直角坐标 PTP 运动

功能

用这个功能可以编程一个在直角坐标系上的位置，但是机床的运行在机床坐标系中完成。如果是通过单一性来运行的，这个功能可以在更换铰接位置时应用。

说明

这个功能只有与有效转换相联系时才有意义。 此外“PTP-运行”只有与 G0 和 G1 联系时才允许。

句法

```
N... TRAORI
N... STAT='B10' TU='B100' PTP
N... CP
```

生成 5/6 轴转换时的 PTP 运动

如果在用 PTP 有效生成 5/6 轴转换时，在机床坐标系 (ORIMKS) 中激活一个点对点运动，则刀具定向可以用回转轴位置

```
N... G1 X Y Z A B C
```

或者与运动无关的矢量欧拉角或 RPY 角

```
N... ORIEULER 或者 ORIRPY
N... G1 X Y Z A2 B2 C2
```

或者方向矢量

```
N... G1 X Y Z A3 B3 C3
```

编程。 同时，可以用大圆弧插补 ORIVECT 激活回转轴插补和矢量插补，或者沿一个圆锥表面 ORICONxx 激活定向矢量的插补。

带矢量的定向的多义性

在编程带矢量的定向时，在可能的回转轴位置上有多义性。 要返回的回转轴位置同时可以通过编程 STAT = <...> 来选择。 如果

编程（根据默认设置）STAT = 0，
则返回到与起始位置距离最短的位置。 如果

编程 STAT = 1，
则返回到与起始位置距离最长的位置。

含义

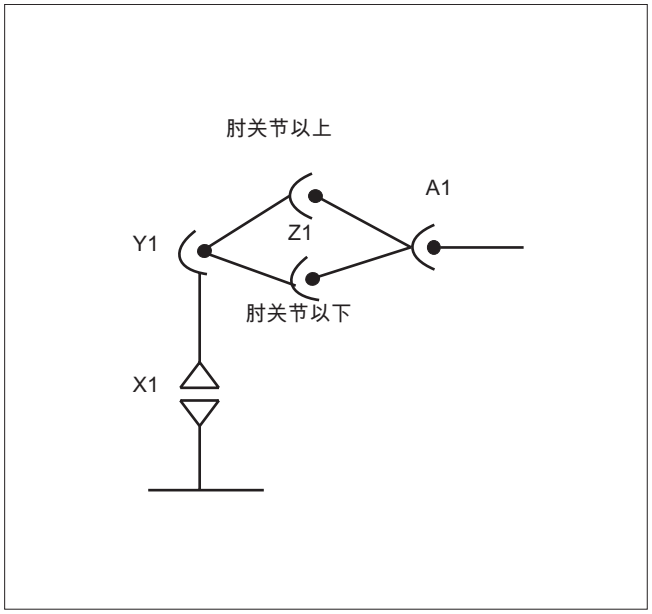
指令 PTP 和 CP 为模态有效。 CP 是默认设置。

在 STAT 值的编程模特有效期间，TU = <...> 的编程以逐段方式有效。

进一步区分，同时在 TU 的编程和有效回转轴插补情况下被评估时，STAT 值的编程仅在矢量插补时有效。

PTP	**P**oint **t**o **P**oint （点到点运动） 作为同步轴运动执行运行；参与运行的最慢的轴是速度主导轴。
CP	**c**ontinuous **p**ath (轨迹运动) 该运动为直角坐标系中的轨迹运动。
STAT=	铰接的位置；值取决于转换。
TU=	TURN 信息为逐段有效。 因此可以明确地返回到-360 度和+360 度之间的轴角度。

示例




```
N10 G0 X0 Y-30 Z60 A-30 F10000
```
起始位置
→ 肘部以上

```
N20 TRAORI(1)
```
转换 开

```
N30 X1000 Y0 Z400 A0
```

```
N40 X1000 Z500 A0 STAT='B10'
TU='B100' PTP
```
重新定向，没有转换
→ 肘部以下

```
N50 X1200 Z400 CP
```
转换重新激活

```
N60 X1000 Z500 A20
N70 M30
```

生成 5 轴转换时的 PTP 运动举例

接受： 以一个直角 CA 运动为基础。

程序代码	注释
TRAORI	; CA 运动转换启动
PTP	; PTP 运动启动
N10 A3 = 0 B3 = 0 C3 = 1	; 回转轴位置 C = 0 A = 0
N20 A3 = 1 B3 = 0 C3 = 1	; 回转轴位置 C = 90 A = 45

程序代码	注释
N30 A3 = 1 B3 = 0 C3 = 0	; 回转轴位置 C = 90 A = 90
N40 A3 = 1 B3 = 0 C3 = 1 STAT = 1	; 回转轴位置 C = 270 A = -45

选择唯一的回转轴返回位置：

同时在程序段 N40 中，回转轴通过编程 STAT = 1 以最长行程自其起点（C=90, A=90）运行到终点（C=270, A=–45）， 而当 STAT = 0 时，则以最短行程运行到终点（C=90, A=45）。

说明

通过指令 PTP 和 CP 在直角坐标运动和加工轴运动之间进行切换。

生成 5/6 轴转换时的 PTP 运动

在 PTP 运动时，与 5/6 轴转换不同，如果仅方向改变，TCP 一般不保持固定。线性返回所有转换轴（3 个线性轴和最多 3 个回转轴）的转换最终位置，同时转换在实际意义上还保持无效。

通过编程模态 G 代码 CP 结束 PTP 运动。

不同的转换包含在手册中：
/FB3/ 功能手册特殊功能； 转换包处理（TE4）。

位置编程(STAT=)

机床设置不只由带直角坐标的位置说明和刀具的方向决定。分别根据不同的机床类型，存在最多 8 种不同的或者有区别的铰接配置。这些是转换专用的。为了能够将直角坐标位置明确换算成轴间夹角，必须使用指令 STAT= 来规定接合点的位置。指令 "STAT" 为二进制值，每个可能有的位置均有一位。

表示位置的位，必须在使用 "STAT" 的情况下进行编程，参见：
/FB2/ 功能手册扩展功能；运动转换（M1），章节“直角坐标 PTP 运动”。

轴角度编程(TU=)

为了能够明确接近小于 < ±360 度的轴间夹角，必须使用指令 "TU= " 来对该信息进行编程。

轴以最短行程运行：

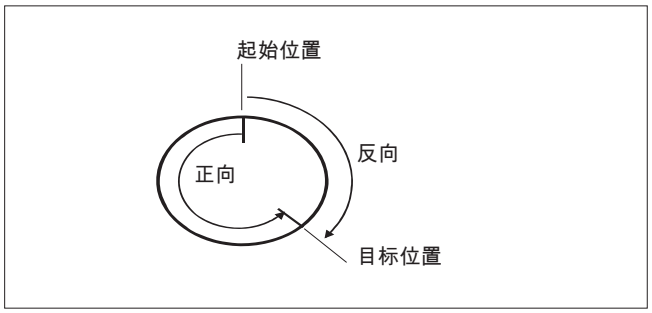
- 当某个位置没有编程 TU 时，
- 如果是运动范围大于 > ±360 度的轴。

举例：

图中给出的目标位置可以从正向或者负向返回运行。在地址 A1 下编程编程这个方向。

A1=225°, TU=Bit 0, → 正向

A1=−135°, TU=Bit 1, → 反向



评估用于生成 5/6 轴转换和目标位置的 TU 的举例

变量 TU 针对每个轴都包含一个进入转换的显示运行方向的位。根据回转轴的通道轴视图分配 TU 位。仅在最大可能有 3 个回转轴进入转换时评估 TU 信息：

位 0：轴 1，TU 位 = 0：0 度 <= 回转轴角 < 360 度

位 1：轴 2，TU 位 = 1：–360 度 < 回转轴角 < 0 度

某个回转轴的起点为 C = 0，通过编程 C = 270 将回转轴运行到下列目标位置上：

C = 270：TU 位 0，正向旋转方向

C = –90：TU 位 1，负向旋转方向

其它特性

运行方式转换

功能“直角的 PTP-运行”只在运行方式 AUTO 和 MDA 时有意义。在 JOG 之后转换运行方式时，当前设置保持不变。

如果 G 代码 PTP 已设置，轴将在 MKS 中运动。如果 G 代码 CP 已设置，轴将在 WKS 中运动。

上电/复位

在上电或者复位之后，设置取决于机床数据 $MC_GCODE_REST_VALUES[48]。默认设置的运动方式为 "CP"。

REPOS

如果在中断过程中已经设置了“直角坐标 PTP 运动”功能，也可用 PTP 复位。

叠加的运动

在直角坐标 PTP 运动情况下，只能进行有限度的 DRF 位移或者外部零点位移。在从 PTP－到 CP－运行的转换时在 BKS 中不允许有叠加。

在 CP 和 PTP 运行之间精磨

在程序段之间可以使用 G641 对过渡磨削进行编程。

磨削范围的大小是以毫米或者英寸表示的轨迹行程，在这个范围内进行程序段过渡磨削。大小说明如下：

- 用于 GO 程序段，带有 ADISPOS
- 用于所有其它行程指令，带有 ADIS

轨迹行程计算在非 G0 程序段时与对 F 地址的考虑相一致。朝向 FGROUP(...) 中所规定的轴进给。

进给计算

对于 CP 程序段，使用基准坐标系的直角轴来计算。

对于 PTP 程序段，机床坐标系的相应的轴用来计算。

6.9.1 PTP 当 TRANSMIT 时

功能

使用“PTP 当 TRANSMIT 时”，可以优化 G0- 和 G1 程序段的执行时间。 不是使基准坐标系的轴以线性运动（CP），而是使加工轴以线性运动（PTP）。 这样，极点附近加工轴的运动变化就可发挥作用，使得能够极为迅速地到达程序段末尾。

零件程序继续在工件直角坐标系中写入并且所有坐标位移、旋转和框架编程均保持有效。HMI 上的模拟也同样在工件直角坐标系中显示。

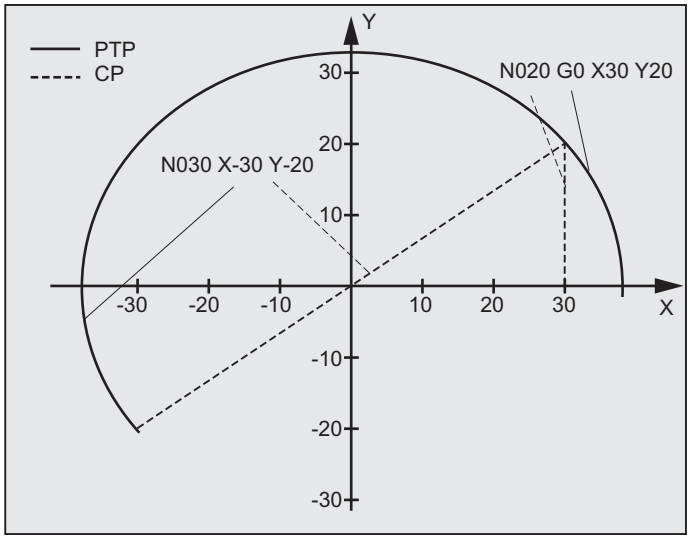
句法

```
N... TRANSMIT
N... PTPG0
N... G0 ...
...
N... G1 ...
```

含义

TRANSMIT	激活第一个约定的 TRANSMIT 功能 (参见章节“在车削件上进行洗削加工： TRANSMIT")
PTPG0	Point to Point G0 (点到点运动自动执行到每个 G0 程序段，然后重新设定 CP） 因为 STAT 和 TU 为模态，上一次编程的数值始终有效。
PTP	Point to Point （点到点运动） 对于 TRANSMIT 而言，PTP 表示在直角坐标系中，在阿基米德螺旋线上要么围绕极点或者从极点运动出来。 这里所得出的刀具运动明显不同于 CP，且已在相应的编程示例中描述过。
STAT=	解决关于极点的多义性。
TU=	TU 对于 TRANSMIT 的 PTP 没有多大意义。

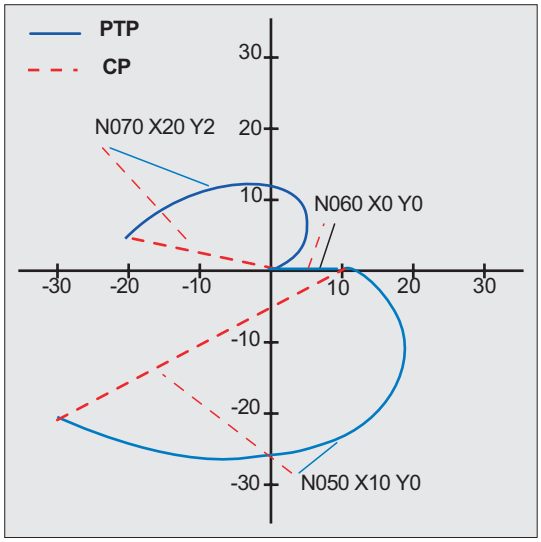
使用 PTP 和 TRANSMIT 围绕极点运动的示例



程序代码	注释
N001 G0 X30 Z0 F10000 T1 D1 G90	; 起始位置 绝对尺寸
N002 SPOS=0	
N003 TRANSMIT	; 转换 TRANSMIT
N010 PTPG0	; 自动到每个 G0 程序段 PTP 且然后重新 CP
N020 G0 X30 Y20	

程序代码	注释
N030 X-30 Y-20	
N120 G1 X30 Y20	
N110 X30 Y0	
M30	

使用 PTP 和 TRANSMIT 从极点运动出来



编程	注释
N001 G0 X90 Z0 F10000 T1 D1 G90	; 起始位置
N002 SPOS=0	
N003 TRANSMIT	; 转换 TRANSMIT
N010 PTPG0	; 自动到每个 G0 程序段 PTP 且然后重新 CP
N020 G0 X90 Y60	
N030 X-90 Y-60	
N040 X-30 Y-20	
N050 X10 Y0	
N060 X0 Y0	
N070 X-20 Y2	
N170 G1 X0 Y0	
N160 X10 Y0	
N150 X-30 Y-20	
M30	

说明

PTP 和 PTPG0

PTPG0 对于所有 PTP 可以处理的转换而言均要考虑。在所有其它情况下， PTPG0 没有多大意义。

G0 程序段在 CP 模式中执行完毕。

PTP 或者 PTPG0 的选择在零件程序中进行，或者在机床数据文件中通过取消 CP $MC_GCODE_RESET_VALUES[48].

⚠小心

边界条件

关于刀具运动和碰撞，适用多个边界条件和某些功能消除，如：

使用 PTP 不允许有刀具半径补偿（WRK）激活。

使用 PTPG0 可在激活刀具半径补偿（WRK）时以 CP 运动。

使用 PTP 无法柔性进入和退出（WAB）。

使用 PTPG0 可在柔性进入和退出（WAB）时以 CP 运动。

使用 PTP 不可以有切削循环 (CONTPRON, CONTDCON)。

使用 PTPG0 可在切削循环中 (CONTPRON, CONTDCON) 以 CP 运动。

倒棱 (CHF, CHR) 和倒圆 (RND, RNDM) 被忽略。

压缩器与 PTP 不相容并且自动在 PTP 程序段中被取消。

插补中的轴交迭在 PTP 段过程中不得改变。

当执行 G643 时，结束精磨之后自动用轴向精度转换 G642 。

当 PTP 激活时，转换的轴并不能同时为定位轴。

文献：

/FB2/功能手册扩展功能；运动转换（M1），
章节“直角坐标 PTP 运动”

PTP 当 TRACON 时:

PTP 也可以与 TRACON 一起使用，当第一个级联转换支持 PTP 时。

STAT= 和 TU= 的含义，当 TRANSMIT 时

如果圆轴旋转 180 度，或者当 CP 时轮廓穿过极点，圆轴可能会依据机床数据文件 $MC_TRANSMIT_POLE_SIDE_FIX_1/2 [48] 旋转 -/+ 180 度并且以顺时针或者逆时针方向运动。同样也能设置是否通过极点运动，或者围绕极点旋转。

6.10 在选择一个转换时的边界条件

功能

选择转换可以通过零件程序或者 MDA。对此要注意：

- 不添加一个运行中间程序段（棱角/半径）。
- 一个样条程序段顺序必须已经结束；如果没有，就会显示一个信号提示。
- 刀具精密补偿必须已经取消 (FTOCOF); 如果没有，就会显示一个信号提示。
- 刀具半径补偿必须已经取消 (G40); 如果没有，就会显示一个信号提示。
- 一个激活的刀具长度补偿由控制器接收到转换。
- 在转换之前有效的当前框架由控制器取消。
- 一个当前有效的工作范围限制对于和转换有关的轴由控制器取消（和 WALIMOF 相适应）。
- 取消保护范围监控。
- 轨迹控制运动和精磨被中断。
- 所有在机床数据中说明的轴必须程序段同步化。
- 将更换的轴换回来；如果不这样，会出现一个信号提示。
- 在不独立的轴时输出一个信号。

换刀

换刀只有在取消刀具半径补偿时才可以。

刀具长度补偿的转换和刀具半径补偿的选择/取消不可以在同一个程序段内编程。

框架转换

所有仅以基本坐标系为参照的语句均允许（FRAME，刀具半径补偿）。但是 G91 时（增量尺寸）的框架转换不作特别处理 － 与未激活转换时不同。待执行的增量在新框架的工具坐标系中予以分析 － 与前一个程序段中哪一个框架在起作用没有关系。

排除在外

涉及转换的轴不可以：

- 用作预置轴（报警），
- 用于向固定点返回（报警），
- 用于找零运行（报警）。

6.11 取消转换 (TRAFOOF)

功能

通过指令 TRAFOOF 可以取消所有激活的坐标转换和框架。

说明

之后所需的框架必须通过更新的编程生效。

对此要注意：

适用于取消转换的边界条件与选择的边界条件一样（参见“选择一个转换时的边界条件”一章）。

句法

```
TRAFOOF
```

含义

TRAFOOF	取消所有激活的坐标转换/框架的指令

6.12 级联转换 (TRACON, TRAFOOF)

功能

每次可前后连接**两个**转换（级联），使得源于第一个转换的轴的运动分量作为第二个级联转换的输入数据。 第二转换的运行分量影响加工轴。

级联允许包含**两个**转换。

说明

一个刀具总是分配到级联的第一个转换。 后来的转换会这样运行，就好像当前有效的刀具长度是零。 只有通过机床参数所设置的刀具基本长度 (_BASE_TOOL_) 才会对级联的第一个转换有效。

机床制造商

请注意机床制造商的说明，有可能通过机床数据在前面定义的转换。

转换和级联的转换是选项。 有关特定控制系统中级联转换的可用性，分别在各个“目录”中给出信息。

应用

- 使用一个斜置砂轮磨削已作为圆柱展开面包络线编程的轮廓 (TRACYL)，例如刀具磨削。
- 使用斜置砂轮对一个使用 TRANSMIT 生成的非圆形轮廓进行精加工。

句法

```
TRACON（trf，par）        一个级联的转换开通。
TRAFOOF
```

含义

TRACON	级联的转换开通。 另一个之前有效的转换通过 TRACON()隐含关闭。
TRAFOOF	上一次开通的（级联）转换被关闭。

trf	级联转换的编号： 0 或者 1 用于第一个/唯一的级联转换。 如果该位置上没有编程任何内容，则数值设定 0 或者 1 意义相同，即激活第一个/唯一的转换。 2 用于第二个级联转换。（值不等于 0－2 会发出一个错误报警）。
par	一个或者多个通过逗号分隔开的参数，用于级联中等候参数的转换，例如斜轴的角度。 如果是尚未设定的参数，则默认设置或者上一次所使用的参数有效。 通过置小数点必须考虑到：当要让前一个参数的基本设置有效时，对已知参数按照其等候的顺序进行分析。 特别是当声明至少一个参数时，在该参数前就必须有一个逗号，即使当 trf 的声明没有必要时也是如此，例如 TRACON(, 3.7).

前提条件

第二 个转换必须为**"斜轴"** (TRAANG) 。作为第一个转换可以：

- 定向转换（TRAORI），包括万向铣头
- TRANSMIT
- TRACYL
- TRAANG

使用一个关联转换启用指令的条件是：已通过机床参数定义了各个需要进行关联的转换和需要激活的关联转换。

转换详细描述中说明的边界条件和特殊情况在使用级联时也需要注意。

关于转换机床数据的设计，参见：

/FB2/ 功能手册扩展功能；运动转换（M1）和

/FB3/ 功能手册特殊功能：3 至 5 轴转换（F2）。

刀具补偿 7

7.1 补偿存储器

功能

建立补偿存储器

每个数据字段均可用 T 和 D 编号调用（除了“平面 D 编号”之外）并且除了刀具的几何数据之外，还包含有其它记录，例如刀具类型。

平面 D 号结构

如果刀具管理在 NCK 之外进行，那么使用“面积 D 编号结构”。 在这种情况下带从属刀具补偿程序段的 D 编号不对刀具进行赋值。

在零件程序中可以进一步编程 T。 但是这个 T 和编程设计的 D 编号没有关系。

用户刀沿数据

通过机床数据文件可以配置用户切削数据。请注意机床制造商说明。

刀具参数

说明

补偿存储器中的各个参数值

补偿存储器 P1～P25 的各个参数值可通过程序的系统变量读写。 所有其他的参数被保留。

刀具参数 $TC_DP6 至 $TC_DP8，$TC_DP10 和 $TC_DP11 以及 $TC_DP15 至 $TC_DP17，$TC_DP19 和 $TC_DP20 视刀具类型不同有另一个含义。

[1] 对于铣刀也适用于 3D 端面铣
[2] 对于切槽锯片刀具类型
[3] 预留： 不由 SINUMERIK 840D 使用

刀具参数编号(DP)	系统变量的意义	附注
$TC_DP1	刀具类型	概要参见清单
$TC_DP2	刀沿位置	仅用于车刀

刀具参数编号(DP)	系统变量的意义	附注
几何尺寸	**长度补偿**	
$TC_DP3	长度 1	计算
$TC_DP4	长度 2	类型和平面
$TC_DP5	长度 3	
几何尺寸	**半径**	
$TC_DP6[1] $TC_DP6[2]	半径 1 / 长度 1 直径 d	铣刀/车刀/磨削刀具 切槽锯片
$TC_DP7[1] $TC_DP7[2]	长度 2 / 圆锥形铣刀的拐角半径 切槽锯片拐角半径	铣刀 切槽锯片
$TC_DP8[1] $TC_DP8[2]	铣刀的倒圆半径 1 突出长度 k	铣刀 切槽锯片
$TC_DP9[1,3]	倒圆半径 2	备用
$TC_DP10[1]	刀具端面角度 1	圆锥形铣刀
$TC_DP11[1]	刀具纵轴角度 2	圆锥形铣刀
磨损	**长度和半径补偿**	
$TC_DP12	长度 1	
$TC_DP13	长度 2	
$TC_DP14	长度 3	
$TC_DP15[1] $TC_DP15[2]	半径 1 / 长度 1 直径 d	铣刀/车刀/磨削刀具 切槽锯片
$TC_DP16[1] $TC_DP16[3]	长度 2 / 圆锥形铣刀的拐角半径，拐角半径的槽宽 b	铣刀 切槽锯片
$TC_DP17[1] $TC_DP17[2]	铣刀的倒圆半径 1 超出长度 k	铣削 / 3D 端面铣 切槽锯片
$TC_DP18[1,3]	倒圆半径 2	备用
$TC_DP19[1]	刀具端面角度 1	圆锥形铣刀
$TC_DP20[1]	刀具纵轴角度 2	圆锥形铣刀
基本尺寸/适配器	**长度补偿**	
$TC_DP21	长度 1	
$TC_DP22	长度 2	

刀具参数编号(DP)	系统变量的意义	附注
$TC_DP23	长度 3	
工艺		
$TC_DP24	后角	仅用于车刀
$TC_DP25		备用

备注

几何尺寸（例如长度 1 或者半径）存在多个记录组成部分。这些部分经相加得出结果尺寸(例如长度总和 1，半径总和), 然后将成为有效尺寸。

不需要的补偿可以用值零来覆盖。

刀具参数 $TC-DP1 至 $TC-DP23，带轮廓刀具

说明

不评估表中未列出的刀具参数，例如 $TC_DP7，即其内容无意义。

刀具参数编号(DP)	含义	刀沿 Dn		附注
$TC_DP1	刀具类型			400 至 599
$TC_DP2	刀沿位置			
几何尺寸	**长度补偿**			
$TC_DP3	长度 1			
$TC_DP4	长度 2			
$TC_DP5	长度 3			
几何尺寸	**半径**			
$TC_DP6	半径			
几何尺寸	**极限角度**			
$TC_DP10	最小极限角度			
$TC_DP11	最大极限角度			
磨损	**长度和半径补偿**			

刀具参数编号(DP)	含义	刀沿 Dn		附注
$TC_DP12	长度 1 磨损量			
$TC_DP13	长度 2 磨损量			
$TC_DP14	长度 3 磨损量			
$TC_DP15	磨损半径			
磨损	**极限角度**			
$TC_DP19	最小极限角度的磨损量			
$TC_DP20	最大极限角度的磨损量			
基本尺寸/适配器	**长度补偿**			
$TC_DP21	长度 1			
$TC_DP22	长度 2			
$TC_DP23	长度 3			

基本值和磨损值

得出的尺寸分别作为总和由基本值和磨损值计算得出（例如用于半径的 $TC_DP6 + $TC_DP15）。 此外，将基本尺寸（$TC_DP21 – $TC_DP23）加到第一个刀沿的刀具长度。 此外，所有其他尺寸都影响该刀具长度，对于传统刀具，这些尺寸还可能影响有效刀具长度（适配器、可定向的刀架、设定数据）。

极限角度 1 和 2

极限角度 1 和 2 分别以刀沿终点到刀沿参考点的矢量为参照并以逆时针方向计数。

7.2 附加补偿

7.2.1 选择附加补偿（DL）

功能

附加补偿实际上就是一种可以在加工过程中编程的过程补偿。它们与一个切削刃的几何数据相关，因此是刀具切削刃数据的组成部分。

附加补偿数据通过一个 DL 号响应（DL: 与位置相关；根据使用位置补偿），通过操作界面输入。

应用

通过附加补偿，可以补偿使用地点条件下的尺寸误差。

句法

```
DL=<编号>
```

含义

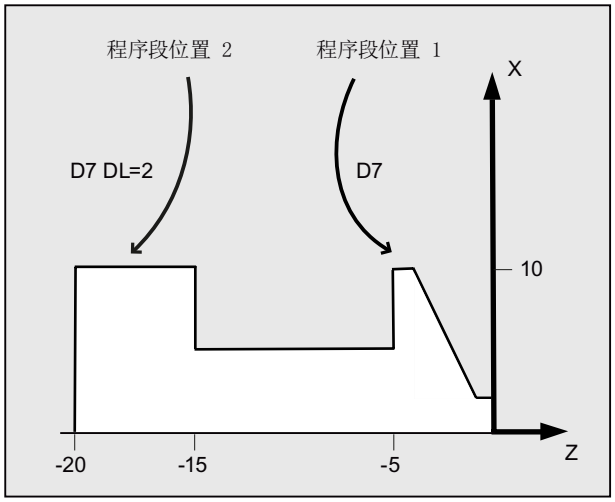
DL	用于激活附加补偿的指令
<编号>	通过参数 <编号> 可以指定待激活的附加刀具补偿数据段。

说明

如何确定附加补偿的数量和激活附加补偿，可以通过机床数据进行（→ 请注意机床制造商的说明！）。

示例

同一个切削刃可以用于 2 个轴颈。



程序代码	注释
N110 T7 D7	; 转塔位于刀位 7。 D7 和 DL=1 被激活，并在下一个程序段中退回。
N120 G0 X10 Z1	
N130 G1 Z-6	
N140 G0 DL=2 Z-14	; D7 和 DL=2 被激活，在下一个程序段中退回。
N150 G1 Z-21	
N160 G0 X200 Z200	; 回到刀具更换点。
...	

7.2.2 确定磨损值和设置值（$TC_SCPxy[t,d], $TC_ECPxy[t,d]）

功能

磨损量和设置值可以通过系统变量读取和写入。 这里的逻辑关系以刀具和刀沿相应的系统变量的逻辑为基准。

系统变量

系统变量	含义
$TC_SCPxy[<t>,<d>]	磨损量通过 xy 分配给各自的几何参数，x 代表磨损量的数字，y 则是建立与几何参数的关系。
$TC_ECPxy[<t>,<d>]	设置值通过 xy 分配给各自的几何参数，x 代表设置值的数字，y 则是建立与几何参数的关系。
<t>: 刀具 T 编号 **<d>**: 刀具切削刃的 D 号码	

说明

确定的磨损量和设置值附加到几何参数和其它的补偿参数（D 号码）。

示例

对于刀具（t）的切削刃（D 号码)，其长度 1 的磨损量确定为值 1.0。

参数： $TC_DP**3** （长度 1， 用于车刀）

磨损量： $TC_SCP1**3** bis $TC_SCP6**3**

设置值： $TC_ECP1**3** bis $TC_ECP6**3**

$TC_SCP4**3** [<t>,<d>] = 1.0

7.2.3 清除附加补偿（DELDL）

功能

用指令 DELDL 删除一个刀具切削刃的附加补偿（存储器使能）。 这时，不管是确定的磨损量还是设置值均清除。

句法

```
DELDL[<t>,<d>]
DELDL[<t>]
DELDL
<状态>=DELDL[<t>,<d>]
```

含义

DELDL	用于删除附加补偿的指令
<t>	刀具 T 编号
<d>	刀具切削刃的 D 号码
DELDL[<t>,<d>]	删除所有刀具 <t> 的切削刃 <d>附加补偿。
DELDL[<t>]	刀具<t> 的所有切削刃的附加补偿均被删除。
DELDL	TO 单元的所有刀具的所有附加切削刃补偿均被删除（对于编程了该指令的通道）
<状态>	删除状态

值：	含义：
0	已经成功地进行删除。
-	没有进行删除（如果参数设定规定的是一个刀沿），或者删除没有完全进行（如果参数设定多个刀沿）。

说明

有效刀具的磨损量和设置值不可以被删除（类似于 D 补偿或刀具数据的删除特性）。

7.3 刀具补偿 - 特殊操作

功能

设定数据 SD42900 - SD42960 可以用于控制刀具长度和磨损量符号的赋值。

这同样适用于几何轴镜像时的磨损量分量，或者在更换加工平面时的磨损量分量特性、以及在刀具方向上的进行温度补偿时。

磨损量

如果在磨损量之后给出一个参考基准，则表明是实际磨损量的和（$TC_DP12 到 $TC_DP20），以及磨损量（$SCPX3 到 $SCPX11）和设置值（$ECPX3 到 $ECPX11）的补偿值之和。

有关补偿值之和的完整信息，请见：
文献：
刀具管理功能手册

设定数据

设定数据	含义
SD42900 $SC_MIRROR_TOOL_LENGTH	镜像刀具长度分量和基准尺寸分量。
SD42910 $SC_MIRROR_TOOL_WEAR	镜像刀具长度分量的磨损量。
SD42920 $SC_WEAR_SIGN_CUTPOS	磨损量分量的符号赋值，与切削刃位置相关。
SD42930 $SC_WEAR_SIGN	反相磨损量尺寸的符号。
SD42935 $SC_WEAR_TRANSFORM	转换磨损量。
SD42940 $SC_TOOL_LENGTH_CONST	将刀具长度分量分配到几何轴。
SD42950 $SC_TOOL_LENGTH_TYPE	刀具长度分量的分配与刀具类型无关。
SD42960 $SC_TOOL_TEMP_COMP	刀具方向的温度补偿值 在前面已编程的刀具方向上也有效。

文献

功能手册 基本功能；刀具补偿（W1）

其它信息

使修改的设定数据生效

只有在下次选择了一个刀沿时，设定数据修改后的刀具分量其新的赋值才生效。 如果一个刀具已经生效，则只有重新选择该刀具后，修改后的刀具的赋值数据才生效。

如果发生这种情况：即因为一个轴的镜像状态改变，使所产生的刀具长度改变，则这种情况与上述相同。 也就是说，在镜像指令后必须重新选择刀具，这样修改后的刀具长度分量才会生效。

可定向的刀架和新的设定数据

设置数据 SD42900 － SD42940 对一个可能激活的可定向刀架不起作用。 但是，一个刀具总是把所有的长度（刀具长度＋磨损量＋基准尺寸）加入到可定向刀架的计算中。 在计算所生成的总长时，要考虑所有由设定数据引起的改变；也就是说可定向刀架的矢量与加工平面无关。

说明

在使用可定向刀架时经常要求定义所有的刀具（在没有镜像的基准系统中），包括那些仅在镜像加工中使用的刀具。 这样在加工镜像轴时给刀架旋转，使刀具的实际位置正确表述。 刀具长度分量自动在正确的方向生效，从而就没有必要由控制系统通过设定数据给每个分量赋值（取决于各个轴的镜像状态）。

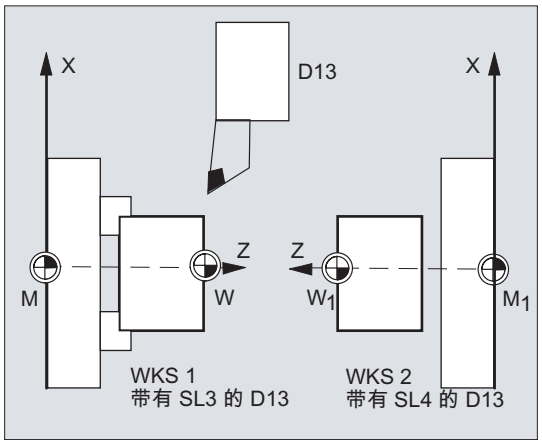
其它的应用可能

可定向刀架的这种功能非常有用，特别是在机床中，如果无法给刀具旋转，或者刀具在不同的方向已经固定安装。 这样刀具可以统一地在一个基准方向标注尺寸，然后通过一个虚拟地刀架的旋转产生加工时所需要的尺寸。

7.3.1 刀具长度镜像

功能

采用设置好不为零的设置数据 SD42900 $SC_MIRROR_TOOL_LENGTH 和 SD42910 $SC_MIRROR_TOOL_WEAR， 可以用相应轴的磨损量对刀具长度分量和基本尺寸分量进行镜像。



SD42900 $SC_MIRROR_TOOL_LENGTH

设置数据 **不等于** 零：

刀具长度分量（$TC_DP3, $TC_DP4 和 $TC_DP5）和基本尺寸分量（$TC_DP21, $TC_DP22 和 $TC_DP23）（其关联轴镜像）通过符号反向而镜像。

磨损量 **没有**一起镜像。 如果磨损量也必须镜像，则必须设定设置数据 SD42910 $SC_MIRROR_TOOL_WEAR。

SD42910 $SC_MIRROR_TOOL_WEAR

设置数据 **不等于** 零：

关联轴镜像的刀具长度分量，其磨损量通过符号反相也同样进行镜像。

7.3.2 磨损量的符号赋值

功能

用设定好不等于零的设置数据 SD42920 $SC_WEAR_SIGN_CUTPOS 和 SD42930 $SC_WEAR_SIGN，可以对磨损量的符号赋值进行反向。

SD42920 $SC_WEAR_SIGN_CUTPOS

设置数据 **不等于** 零：

设置数据不等于零：如果刀具带相应的切削刃方向（车刀和铣刀，刀具类型 400），则在加工平面中的磨损量分量的符号赋值取决于刀沿位置。 如果刀具类型不带相应刀沿方向，则该设定数据没有意义。

在下表中，尺寸通过 X 标记，其符号通过 SD42920（不等于 0）反向：

刀沿位置	长度 1	长度 2
1		
2		X
3	X	X
4	X	
5		
6		
7		X
8	X	
9		

说明

通过 SD42920 和 SD42910 进行后的符号赋值相互之间无关。 比如，一个尺寸参数的符号通过两个设定数据修改，则所产生的符号保持不变。

SD42930 $SC_WEAR_SIGN

设置数据 **不等于** 零：

所有磨损量尺寸的符号都反相。 这既作用于刀具长度上，也用于其他尺寸：比如刀具半径、倒圆半径等等。

如果输入一个正的磨损尺寸值，则借此使得刀具“变短”和“变薄”，请参见章节“刀具补偿，特殊操作”，更改的设置数据将生效“。

7.3.3 激活的加工的坐标系 (TOWSTD/TOWMCS/TOWWCS/TOWBCS/TOWTCS/TOWKCS)

功能

取决于机床的运动性能，或者是可定向刀架的当前状态，在一个这样的坐标系中所测得的磨损量被换算到或者变换到一个合适的坐标系中。

有效加工的坐标系

由下面的坐标系可以计算出刀具长度补偿，可以用此长度补偿，通过刀具组 56 的相应 G 代码，把刀具长度分量“磨损量”计算到有效的刀具中。

- 机床坐标系（MCS）
- 基准坐标系（BCS）
- 工件坐标系（WCS）
- 刀具坐标系（TCS）
- 运动转换的刀具坐标系（KCS）

句法

```
TOWSTD
TOWMCS
TOWWCS
TOWBCS
TOWTCS
TOWKCS
```

含义

TOWSTD	初始设定值，用于刀具长度磨损量补偿
TOWMCS	在 MKS 中刀具长度上的补偿
TOWWCS	在 WKS 中刀具长度上的补偿
TOWBCS	在 BKS 中刀具长度上的补偿
TOWTCS	在刀架参考点上的刀具长度补偿（可定向刀架）
TOWKCS	刀头上的刀具长度补偿（运动转换）

其它信息

区别标志

在下表中列出了最重要的区别特征：

G 代码	磨损量	有效的可定向刀架
TOWSTD	初始设定值，刀具长度	磨损量受旋转控制。
TOWMCS	在 MCS 中的磨损量 如果没有可定向的刀架被激活，则 TOWMCS 与 TOWSTD 一致。	仅旋转所生成的刀具长度的矢量，不考虑磨损量。
TOWWCS	强 WCS 中的磨损量换算到 MCS 中。	刀具矢量计算如同在 TOWMCS 中一样，不考虑磨损量。
TOWBCS	将 BCS 中的磨损量换算到 MCS 中。	刀具矢量计算如同在 TOWMCS 中一样，不考虑磨损量。
TOWTCS	将刀具坐标系中的磨损量换算到 MCS 中。	刀具矢量计算如同在 TOWMCS 中一样，不考虑磨损量。

TOWWCS , TOWBCS, TOWTCS: 磨损量矢量加到刀具矢量中。

线性转换

如果 MCS 是从 BCS 中通过一个线性平移而产生的，则在 MKS 中的刀具长度定义才有意义。

非线性转换

比如，如果用 TRANSMIT 激活一个非线性转换，则在 MCS 作为所要求的坐标系说明时，自动使用 BCS。

没有运动转换并且没有定向刀架

如果既没有运动变换生效，也没有可定向刀架生效，则所有 4 个坐标系（除 WCS 之外）均同时生效。 这样只有 WKS 与其它的坐标系相区别。 因为只有刀具长度需要求值，则坐标系之间的平移就没有意义。

文献：
刀具补偿的其它信息，参见：
功能手册基本功能；刀具补偿（W1）

把磨损量计算在内

设置数据 **SD42935 $SC_WEAR_TRANSFORM** 确定下面三个磨损量分量中：

- 磨损
- 精补偿总和
- 粗补偿总和

哪一个受控于适配变换的旋转，或者受控于一个可定向的刀架，如果下面 G 代码中的一个被激活：

- TOWSTD 基本设置

  刀具长度中的补偿

- TOWMCS 磨损量

  在机床坐标系（MCS）中

- TOWWCS 磨损量

  在工件坐标系（WCS）中

- TOWBCS 磨损量（BCS）

  在基准坐标系中

- TOWTCS 刀架装置（T 刀架参考系）上刀具坐标系中的磨损量
- TOWKCS 在运动转换时，刀头坐标系中的磨损量

说明

各个磨损量分量（分配到几何轴，符号赋值）的赋值受以下影响：

- 激活的平面
- 适配器转换
- 下列设置数据：
  - SD42910 $SC_MIRROW_TOOL_WEAR
  - SD42920 $SC_WEAR_SIGN_CUTPOS
  - SD42930 $SC_WEAR_SIGN
  - SD42940 $SC_TOOL_LENGTH_CONST
  - SD42950 $SC_TOOL_LENGTH_TYPE

7.3.4 刀具长度和平面更换

功能

采用设定好不为零的设置数据 SD42940 $SC_TOOL_LENGTH_CONST，可以在平面更换时将刀具长度分量例如长度、磨损和基本尺寸等分配到车刀和磨削工具的几何轴上。

SD42940 $SC_TOOL_LENGTH_CONST

设置数据 **不等于** 零：

在工作平面更换时（G17 - G19），刀具长度分量（长度、磨损量和基准尺寸）到几何轴的分配没有改变。

下表中说明在车刀和磨削工具（刀具类型 400 到 599）时刀具长度分量到几何轴的分配：

内容	长度 1	长度 2	长度 3
17	Y	X	Z
*)	X	Z	Y
19	Z	Y	X
-17	X	Y	Z
-18	Z	X	Y
-19	Y	Z	X

*) 每个不等于 0 的值，又不等同于六个值中的一个，则作为 18 求值。

下表中说明在其它的工具（刀具类型<400 或者>599）时刀具长度分量到几何轴的分配：

加工平面	长度 1	长度 2	长度 3
*)	Z	Y	X
18	Y	X	Z
19	X	Z	Y
-17	Z	X	Y
-18	Y	Z	X
-19	X	Y	Z

*) 每个不等于 0 的值，又不等同于六个值中的一个，则作为 17 求值。

说明

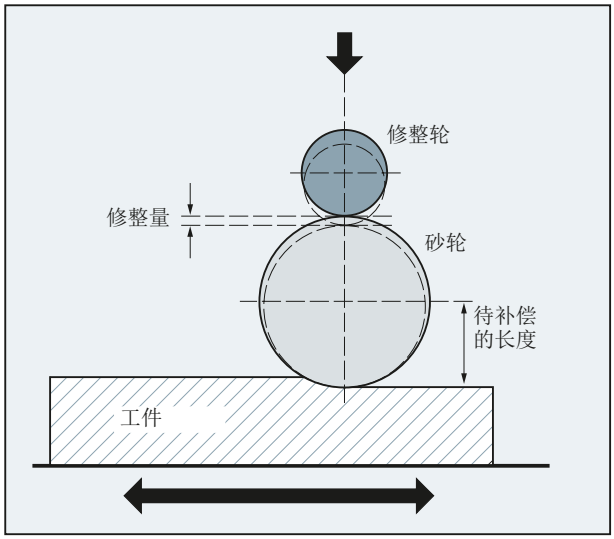
表中的说明以几何轴用 X、Y、Z 命名为前提。 补偿值到轴的分配不是由轴名称决定，而是由轴顺序来决定。

7.4 在线刀具补偿 (PUTFTOCF, FCTDEF, PUTFTOC, FTOCON, FTOCOF)

功能

激活功能“在线刀具补偿”后会立即计算由加工产生的刀具（磨具）长度补偿。

例如：它可以应用于 CD 修整，即在加工过程中同时修整磨削砂轮：



刀具长度补偿可以从加工通道或者一个平行的通道中（修整通道）来改变。

根据所需的修整时间点，使用不同的功能来写入在线刀具长度补偿：

- 逐段连续写入(PUTFTOCF)

  使用 PUTFTOCF 可以在加工时同时进行修整。

  在加工通道中连续根据第 1、2 或者 3 级的一个多项式函数来修改刀具补偿，必须使用 FCTDEF 预先定义该多项式函数。

  PUTFTOCF 始终逐段有效，即在紧接着的运动程序段中。

- 连续模态写入： ID=1 DO FTOC(参见“联机刀具补偿 (FTOC) (页 610)”）

- 不连续写入(PUTFTOC)

  使用 PUTFTOC 可以在加工之外的其他时间，从一个平行通道中进行修整。 PUTFTOC 指定的补偿值会在目标通道中立即有效。

说明

在线刀具补偿只能应用于磨具。

句法

启用/关闭目标通道中的在线刀具补偿：

```
FTOCON
...
FTOCOF
```

写入在线刀具长度补偿：

- 逐段连续：

```
FCTDEF(<函数>,<下限>,<上限>,<a0>,<a1>,<a2>,<a3>)
PUTFTOCF(<函数>,<参考值>,<刀具参数>,<通道>,<主轴>)
...
```

- 不连续：

```
PUTFTOC(<补偿值>,<刀具参数>,<通道>,<主轴>)
...
```

含义

FTOCON: 启用在线刀具补偿

FTOCON 必须在在线刀具补偿应起作用的通道中编程。

FTOCOF: 中断在线刀具补偿

使用 FTOCOF 不会继续获得补偿，但是在刀沿专用的补偿数据中，会完全按照 PUTFTOC/PUTFTOCF 写入的量值进行补偿。

提示：
为最终取消在线刀具补偿，还必须在 FTOCOF 后选择并取消该刀具(T...)。

FCTDEF: 通过 FCTDEF 可以定义 PUTFTOCF 的多项式函数。

参数：

<函数>:	多项式函数的编号	
	类型：	INT
<下限>:	值下限	
	类型：	REAL
<上限>:	值上限	

类型： REAL

<a0> ... <a3>: 多项式函数的系数

类型： REAL

PUTFTOCF: 调用功能“逐段连续式写入在线刀具补偿”

参数：

<函数>: 多项式函数的编号

类型： INT

提示：
它必须和 FCTDEF 的数据一致。

<参考值>: 可变的参考值，从该值可求出补偿，如：不断改变的实际值。

类型： VAR REAL

<刀具参数>: 磨削参数的编号，长度 1、2、3，补偿值将添加到该参数中。

类型： INT

<通道>: 通道号，在线刀具补偿在该通道中生效。

类型： INT

提示：
只有当需要补偿在未激活的通道中生效时，才需要给定该数据。

<主轴>: 主轴号，在该主轴上在线补偿将生效。

类型： INT

提示：
只有当需要补偿针对未激活的磨削砂轮，而不是针对激活的、使用中的刀具生效时，才需要给定该数据。

PUTFTOC: 调用功能“不连续地写入在线刀具补偿”

参数：

<补偿值>: 将会添加到磨削参数中的补偿值。

类型： REAL

<刀具参数>: 参见 PUTFTOCF

<通道>: 通道号，在线刀具补偿在该通道中生效。

类型： INT

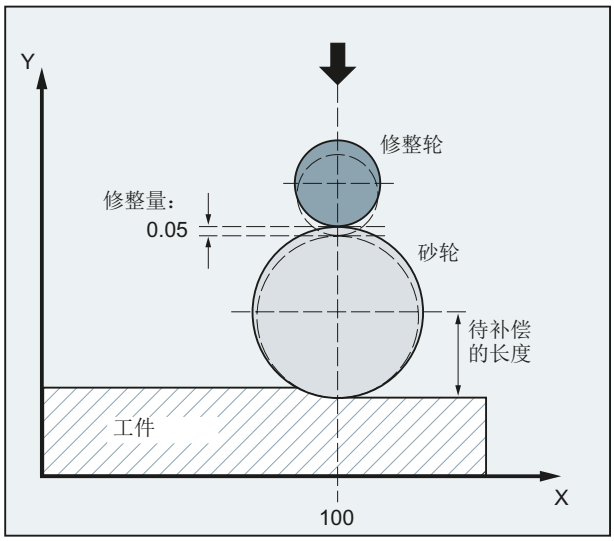
<主轴>: 参见 PUTFTOCF

示例

平面磨削机床：

- Y: 用于砂轮的进给轴
- V: 用于修整轮的进给轴
- 加工通道： 通道 1，用轴 X，Z，Y
- 修整通道： 通道 2，用轴 V

在磨削运行开始后，X100 时砂轮应被修整 0.05。 修整量应该用“连续在线刀具补偿”来编写，在使用磨具时生效。



在通道 1 中的加工程序：

程序代码	注释
…	
N110 G1 G18 F10 G90	; 初始位置。
N120 T1 D1	; 选择当前刀具。
N130 S100 M3 X100	; 启动主轴，运行到初始位置。
N140 INIT(2,"ABRICHT","S")	; 选择通道 2 中的修整程序。
N150 START(2)	; 启动通道 2 中的修整程序。
N160 X200	; 运行到目标位置。
N170 FTOCON	; 启用在线补偿。
N… G1 X100	; 继续加工。
N… M30	

在通道 2 中的修整程序：

程序代码	注释
…	
N40 FCTDEF(1,-1000,1000,-$AA_IW[V],1)	; 定义函数： 斜度 =1 的直线。
N50 PUTFTOCF(1,$AA_IW[V],3,1)	; 连续写入在线刀具补偿： 从 V 轴的运动得出；当前砂轮的长度 3 在通道 1 中补偿。
N60 V-0.05 G1 F0.01 G91	; 用于修整的进给运动，仅在该程序段中，PUTFTOCF 才有效。
…	
N… M30	

其它信息

在线刀具补偿概述

在使用连续写入时（每个 IPO 周期），为了避免给定值的跳跃，在计算功能启用之后每个改变都会添加到磨损存储器中。

以下总是有效的： 在线刀具补偿可以在每个通道中、对每个主轴和磨损参数的长度 1、2 **或者** 3 产生作用。

几何轴长度的赋值在当前加工平面中进行。

编程了 GWPSON 或 TMON 时，应通过刀具数据将刀具指定给主轴，只要它不是激活的砂轮。

当前砂轮面或者左侧砂轮面的磨损参数总是在没有激活的刀具时补偿。

说明

当对多个砂轮面进行相同的补偿时，通过关联规定会自动给第二个砂轮面输入这些参数值。

如果给一个加工通道规定在线补偿，就不得从加工程序从或者通过操作来修改该通道中当前工具的磨损值。

也可为恒定的砂轮圆周速度(SUG)以及刀具监控 TMON 定义在线刀具补偿。

7.5 激活 3D-刀具补偿 (CUT3DC..., CUT3DF...)

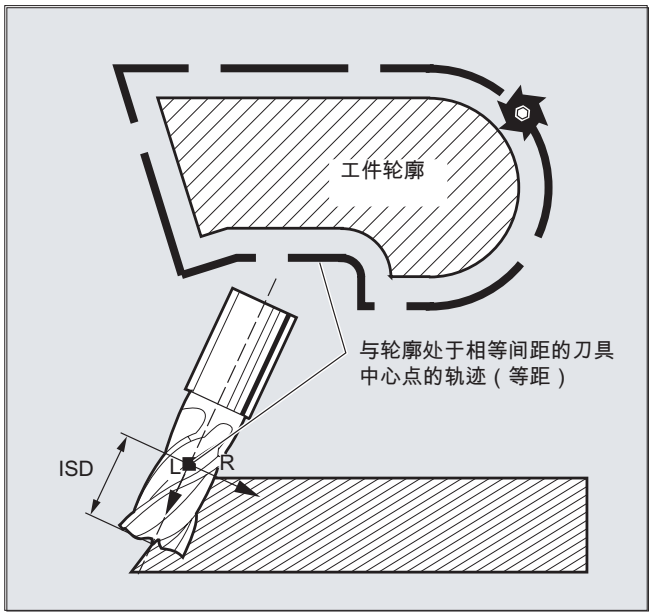
7.5.1 激活 3D 刀具补偿 (CUT3DC, CUT3DF, CUT3DFS, CUT3DFF, ISD)

功能

对圆柱形刀具进行刀具半径补偿时要考虑刀具定位的变化。

对于 3D 刀具半径补偿的选择适用和在 2D 刀具半径补偿时一样的程序指令。 用 G41/G42 说明在左/右运动方向的补偿。 起动特性始终为 NORM。 3D 刀具半径补偿仅在选中 5 轴转换时才有效。

3D 刀具半径补偿也可以作为 5D 补偿，因为在这种情况下在空间中刀具位置的 5 个自由度可供支配。



2 1/2D-和 3D-刀具半径补偿之间的区别

在 3D 刀具半径补偿时刀具方向是可以更改的。在 2 1/2D 刀具半径补偿时，只计算一个刀具的恒定方向。

句法

```
CUT3DC
CUT3DFS
CUT3DFF
CUT3DF
ISD=<值>
```

含义

CUT3DC	激活圆周铣削的 3D 半径补偿
CUT3DFS	带有恒定定向的端面铣削的 D 刀具补偿。 刀具定向已通过 G17 - G19 确定且不受框架影响。
CUT3DFF	带有恒定定向的端面铣削的 D 刀具补偿。 刀具定向通过 G17 - G19 来确定并且如果有可能是通过框架旋转的方向。
CUT3DF	带有定向变化的端面铣削的 D 刀具补偿（仅当激活 5 轴转换时）。
G40 X... Y... Z...	用于关闭： 带几何轴的线性程序段 G0/G1
ISD	插入深度

说明

这些指令为模态有效并且在相同的组中，如 CUT2D 和 CUT2DF. 在当前平面随着下一次运动才可以取消。 这始终适用于 G40 且和 CUT 指令无关。

中间程序段在有效的 3D 刀具半径补偿时是允许的。2 1/2D 刀具半径补偿的确定有效。

边界条件

- **G450/G451 和 DISC**

  在外角上始终插入一个圆程序段。 G450/G451 没有含义。 不分析指令 DISC。

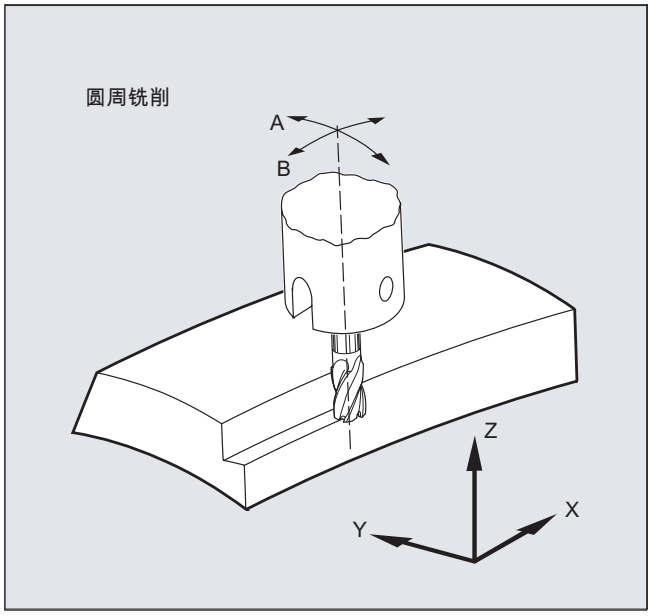
示例

程序代码	注释
N10 A0 B0 X0 Y0 Z0 F5000	
N20 T1 D1	; 刀具调用，调用刀具补偿值。
N30 TRAORI(1)	; 转换选择。
N40 CUT3DC	; 3D 刀具半径补偿选择
N50 G42 X10 Y10	; 刀具半径补偿选择
N60 X60	
N70 …	

7.5.2 3D 刀具半径补偿： 圆周铣削，端面铣削

圆周铣削

这里使用的圆周铣削的变量通过一条轨迹和其方向的说明来实现。在这种加工类型时，在轨迹上刀具类型没有意义。 起决定性作用的是刀具作用点上的半径。

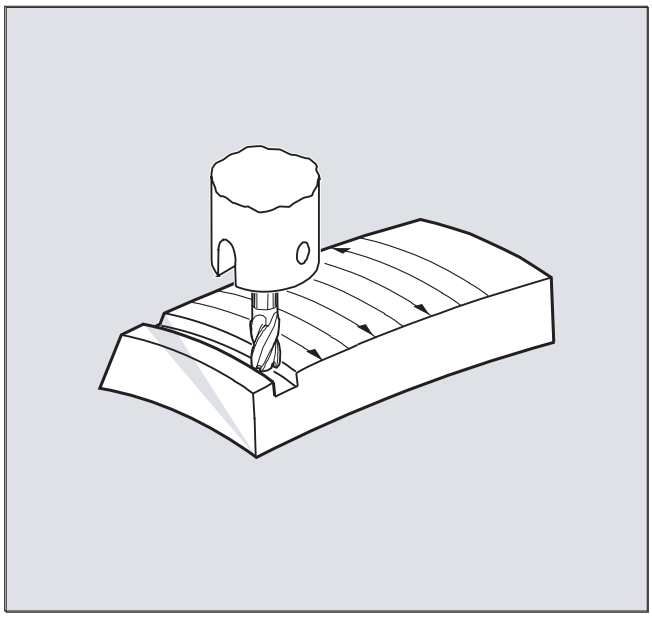


说明

功能 3D-WRK 限制使用在圆柱形刀具上。

端面铣

对于这种 3D 铣削类型，需要在工件表面上按行描述 3D 轨迹。在考虑刀具形状和刀具尺寸的情况下进行计算，通常在 CAM 中计算。后处理器写入零件程序（除了 NC 程序段外）中的是刀具定向（当激活 5 轴转换时）和所需 3D 刀具补偿的 G 代码。 这样机床操作人员就可以使用（与计算 NC 轨迹所使用的刀具不同的）略微小一些的刀具。



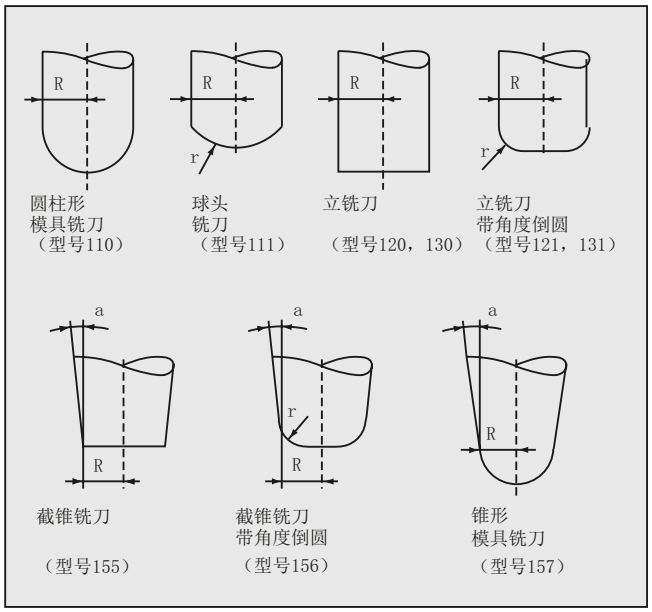
示例：

NC 程序段已使用 10 mm 铣刀计算过。这里也可以用直径为 9.9 毫米的铣刀来完成，对此可以用变化了的成型材料的粗糙度来计算。

7.5.3 3D 刀具半径补偿： 端面铣的刀具类型和刀具数据

铣刀形状，刀具数据

在下面的表格里汇集了允许用于端面铣的刀具类型和刀具数据的极限值。 刀柄的形状未考虑在内。 刀具类型 120 和 156 的作用相同。



如果在 NC 程序中指定了另一个类型编号，它和插图中显示的 T 编号不同，系统会自动使用刀具类型 110（圆柱形模具铣刀）。 超过刀具数据的极限值时会产生报警。

铣刀类型	类型号	R	r	a
圆柱形模具铣刀	110	> 0	-	-
球头铣刀	111	> 0	>R	-
立铣刀，角度铣刀	120.130	> 0	-	-
立铣刀，带倒角功能的角度铣刀	121.131	>r	> 0	-
截锥形铣刀	155	> 0	-	> 0
带有倒角功能的截锥形铣刀	156	> 0	> 0	> 0
锥形模具铣刀	157	> 0	-	> 0

R = 刀柄半径（刀具半径）
r ＝拐角半径

a = 刀具纵轴和环面上方终点之间的角度。
- ＝未求出

刀具数据	刀具参数	
刀具尺寸	几何尺寸	磨损
R	$TC_DP6	$TC_DP15
r	$TC_DP7	$TC_DP16
a	$TC_DP11	$TC_DP20

刀具长度补偿

刀具顶点作为长度补偿的参考点（纵轴/表面的交点）。

3D 刀具补偿，换刀

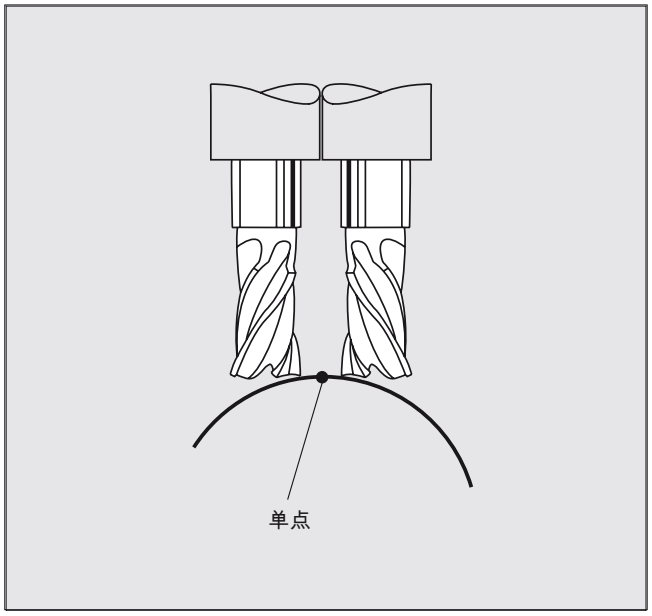
一个带改变后的尺寸（R，r，a）或者另一个类型的新刀具，仅允许通过编程 G41 或 G42 来说明（ G40 根据 G41 或 G42 过渡， 重新编程 G41 或 G42）。 所有其它刀具数据，例如刀具长度，均不为该规则考虑，使得此类刀具也可以在没有更新 G41 或者 G42 的情况下换入。

7.5.4 3D 刀具半径补偿： 轨迹、轨迹曲率和插入深度上的补偿 (CUT3DC，ISD)

功能

轨迹上的补偿

进行端面铣削时，必须对刀具表面上接触点的跳动情况进行观察。 就像在这个例子里用垂直的刀具在进行凸起面积的加工时。 图中所示的应用可以视为极限情况。



这个极限情况由控制器监控，通过在角度定位的基础上识别在刀具和面积标准矢量之间跳跃式的加工点的变化。 控制系统在这些部位上插入线性程序段，使得运动可以执行。

对于计算线性程序段，允许的角度范围储存在侧向角的机床数据中。 如果超过在机床数据中确定的允许角度范围的极限值，那么系统会报警。

轨迹曲线

不监控轨迹曲线。 这里只适用这样的刀具，用这样的刀具工作可以没有轮廓失真。

插入深度（IS）

插入深度 ISD 仅在激活 3D 刀具半径补偿时才会被分析。

使用程序指令 ISD（插入深度）对圆周铣削时刀具的插入深度进行编程。 因此，可以在刀具的表面上改变加工点的位置。

句法

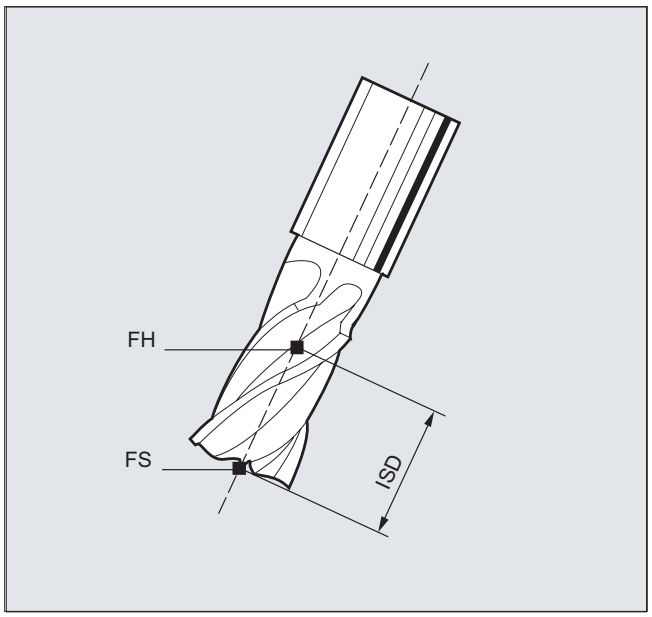
圆周铣削 3D 刀具补偿
```
CUT3DC
ISD=<值>
```

含义

CUT3DC	激活用于圆周铣削的 3D 刀具补偿，例如铣削带有斜壁的凹槽。
ISD	用指令 ISD 规定铣刀刀尖（FS）和铣刀辅助点（FH）直接的间距（<值>）。

铣刀辅助点

铣刀辅助点（FH）通过编程的加工点在刀具轴上的投影产生。



其它信息

带有斜壁的凹槽铣削，用于使用 CUT3DC 进行圆周铣削

通过在必须加工的面积的面积标准的方向上进行横向进给，来补偿在 3D 刀具半径补偿时的铣刀半径的偏差。当插入深度 ISD 保持相同时，铣刀端面所在的平面保持不变。 例如，一个比标准刀具半径小的铣刀有可能到达不了形成分界面的凹槽底部。 用于自动进给刀具时，系统必须已知该分界面，参见“带有分界面的 3D 圆周铣削”一章。

碰撞监控的其它信息，参见：

文献：

编程手册 基本原理；章节“刀具补偿”。

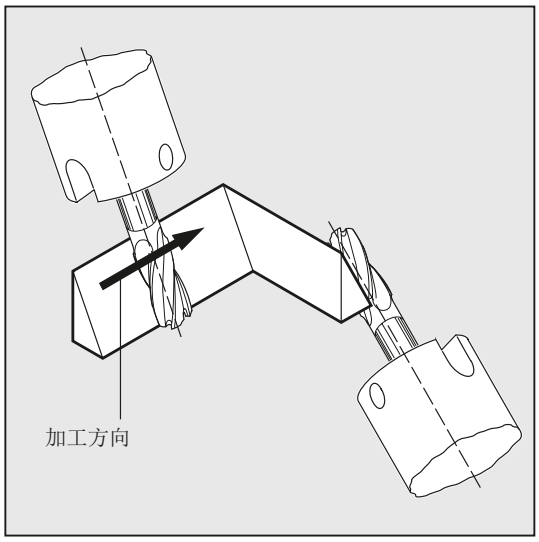
7.5.5 3D 刀具半径补偿：内角/外角和交点法 (G450/G451)

功能

内角/外角

外角和内角分开操作。 内角或者外角的名称取决于刀具方向。

当在某个角上改变定向时，可能会出现角类型在加工过程中产生变化的情况。 如果出现这样的情况，那么产生错误报告并且加工中断。



句法

```
G450
G451
```

含义

G450	过渡圆（刀具按圆形路径绕工件拐角运行）
G451	等距交点（刀具在工件角中切削）

其它信息

用于 3D 补偿的交点法

进行 3D 圆周铣削时，在外角上分析 G 代码 G450/G451 ，即可以向偏置曲线的交点运行。 至软件版本 SW4 总是在外角处插入一个圆弧。 可供使用的交点法对于 CAD 生成的典型 3D 程序特别有用。这些经常由短的直线程序段组成（用于平滑曲线的近似值），在这些程序段时相邻程序段之间的过渡几乎是切线的。

在轮廓外面进行刀具半径补偿时，插入用于外角绕行的迄今原则上的圆弧。 因为这些程序段在几乎切线的过渡时非常短，产生不受欢迎的速度干扰。

在这些情况下模拟 2 ½ D 半径补偿两个参与的曲线延长，并且返回两条延长曲线的交点。

通过延长两个参与程序段的偏移曲线，并且在垂直于角上刀具方向的平面上确定曲线的交点，以此来决定交点。如果不存在这样的交点，就按照现有的方法处理角，即插入一个圆。

交点法的其它信息，参见：
文献：
功能手册 特殊功能；3D 刀具半径补偿（W5）

7.5.6 3D 刀具半径补偿：带有限制面的 3D 圆周铣削

使 3D 圆周铣削与 CAD 程序的实际情况相匹配

通常情况下，由 CAD 系统产生的 NC 程序用大量较短线性程序段近似于一个标准刀具的中心点轨迹。为了使这些生成的很多零件轮廓的程序段尽可能准确地模拟原始轮廓，必需在零件程序中进行某个匹配。

对于最优补偿所需的，但是在零件程序种不可用的重要信息必须用合适的措施来替代。下面说明了典型的方法，用于在零件程序中直接补偿或在计算实际轮廓（例如通过刀具进给）时补偿临界过渡。

应用

除了以一个真实刀具替代标准刀具来描述中心点轨迹的典型应用情况之外，还可用 3D 刀具补偿来处理圆柱形刀具。这时,已编程的轨迹以加工面上的轮廓为参照。 这里所涉及的限制面与刀具无关。 和在习惯的刀具半径补偿时一样，总半径用于计算限制面积上的垂直偏移的计算。

7.5.7 3D 刀具半径补偿：考虑一个限制面 (CUT3DCC, CUT3DCCD)

功能

使用真实刀具进行 3D 圆周铣削

在带有连续或者恒定的刀具定向改变的 3D 圆周铣削时，经常编程一个定义的标准刀具的刀具中心点轨迹。 因为实际操作上合适的标准刀具常常不可以使用，可以使用一个和标准刀具偏差不是太多的刀具。

使用 CUT3DCCD 为实际上不同的刀具提供描述已编程标准刀具的分界面。 NC 程序所描述的是标准刀具的中心点轨迹。

在使用圆柱形刀具时，CUT3DCC 提供一个已编程标准刀具可能已经到达的分界面。 该 NC 程序所描述的是加工面上的轮廓。

句法

```
CUT3DCCD
CUT3DCC
```

含义

CUT3DCCD	激活 3D 刀具补偿，用于使用差分刀具在刀具中心点轨迹上进行带有分界面的圆周铣削： 向限制面进给。
CUT3DCC	激活 3D 刀具补偿，用于使用 3D 半径补偿进行带有限制面的圆周铣削：加工面上的轮廓

说明

带有 G41, G42 的刀具半径补偿

对于带有 G41, G42 的刀具半径补偿而言，当 CUT3DCCD 或者 CUT3DCC 激活时，必须存在“定向转换”选项。

带刀尖圆弧的标准刀具

标准刀具的圆角功能通过刀具参数 $TC_DP7 描述。 从刀具参数 $TC_DP16 可得出真实刀具相对于标准刀具的圆角偏差。

示例

相对于标准刀具，减小半径的圆环铣刀的刀具尺寸

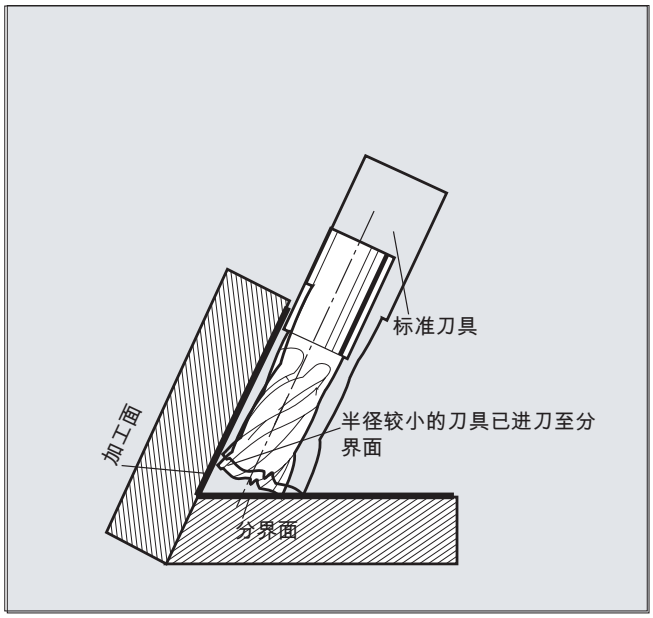
刀具类型	R = 刀柄半径	r = 角半径
带圆角功能的标准刀具	R = $TC_DP6	r = $TC_DP7

刀具类型	R = 刀柄半径	r = 角半径
带有圆角功能的真实刀具： 刀具类型 121 和 131 环形铣刀（立铣刀）	R' = $TC_DP6 + $TC_DP15 + OFFN	r' = $TC_DP7 + $TC_DP16
在这个例子中，$TC_DP15 + OFFN 和 $TC_DP16 都为负。 对刀具类型 ($TC_DP1) 进行分析。		
仅允许带有圆柱形刀柄的铣刀类型(圆柱形铣刀或者立铣刀) 以及环形铣刀（类型 121 和 131) 并且在极限情况下允许圆柱形模具铣刀（类型 110)。	这些允许使用的铣刀类型的角半径 r 等于刀柄半径 R。所有其它允许使用的刀具类型均解释为圆柱形铣刀，且不对可能已规定的圆角尺寸进行分析。	
所有编号 1 －399 的刀具类型都是允许的，但是编号 111 和 155 至 157 例外。		

其它信息

进给至限制面的刀具中心点轨迹 CUT3DCCD

如果使用一个和合适的标准刀具相比半径较小的刀具，那么一个纵向进给的铣刀会继续运行，直到再次碰到槽底。只要刀具允许，这样就可将由加工面和限制面所构成的角清除掉。在此，关系到圆周铣削和端面铣的混合加工方式。 类似于缩小半径的刀具，在使用扩大半径的刀具时，沿相反方向进行相应的进给。



相对于 G 代码组 22 的所有其它刀具补偿，一个给 CUT3DCCD 指定的刀具参数 $TC_DP6 对于刀具半径没有意义，且不影响补偿结果。

补偿偏移从以下几个和中得出：

- 刀具半径的磨损量（刀具参数 $TC_DP15）
- 和一个用于相对限制面计算垂直偏移量的编程的刀具偏移量 OFFN。

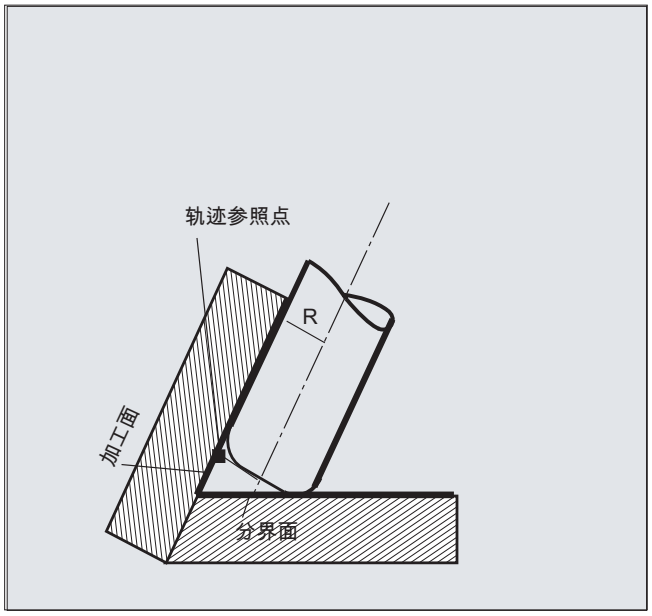
需要加工的表面在轨迹的左边或者右边，不可以从生成的零件程序中得知。因此从原始刀具的一个正半径和一个负磨损值得出。一个负向的磨损值总是描述一个缩小直径的刀具。

圆柱形刀具的使用

在使用圆柱形刀具时，只有当加工面和限制面构成一个锐角（小于 90 度）时，才需要横向进给。如果使用圆环铣刀（带刀尖圆弧的圆柱），那么铣刀不仅在锐角时而且在钝角时需要一个在刀具纵向的进给。

带有 CUT3DCC 的 3D 半径补偿，加工面上的轮廓

如果 CUT3DCC 与一个环形铣刀已激活，则已编程的轨迹以一个等于直径的虚拟圆柱形铣刀为参照。当使用一个环形铣刀时，由此得出的轨迹参考点显示于下图中。



加工面和限制面之间的夹角也允许在一个程序段中从锐角过渡刀钝角，反之亦然。

相对于标准刀具，使用的实际刀具既可以大些，也可以小些。 得出的角半径不可为负，且得出的刀具半径前置符号必须保留。

如果是 CUT3DCC ，NC 零件程序以加工面上的轮廓为参照。 此时，和常轨刀具半径补偿一样，要考虑由下列项目之和所构成的半径总和：

- 刀具半径 (刀具参数 $TC_DP6)
- 磨损量 (刀具参数 $TC_DP15)
- 和一个用于相对限制面计算垂直偏移量的编程的刀具偏移量 OFFN。

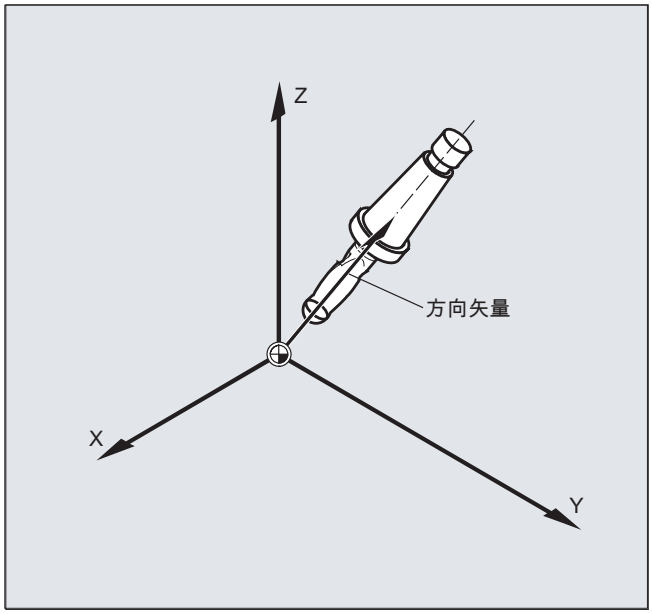
限制面的位置由两个值的分差决定：

- 测量标准刀具
- 刀具半径 (刀具参数 $TC_DP6)

7.6 刀具定向(ORIC, ORID, OSOF, OSC, OSS, OSSE, ORIS, OSD, OST)

功能

刀具定向可以理解为在空间中刀具的几何取向。 使用一个 5 轴加工机床时刀具定向可以通过程序指令来设置。



根据不同刀具定向的插补类型会构成用 OSD 和 OST 激活的定向平滑。

矢量插补生效时，也可通过矢量插补来插补经过平滑的定向曲线。 与此相反，回转轴插补生效时，则直接通过回转轴运动平滑定向。

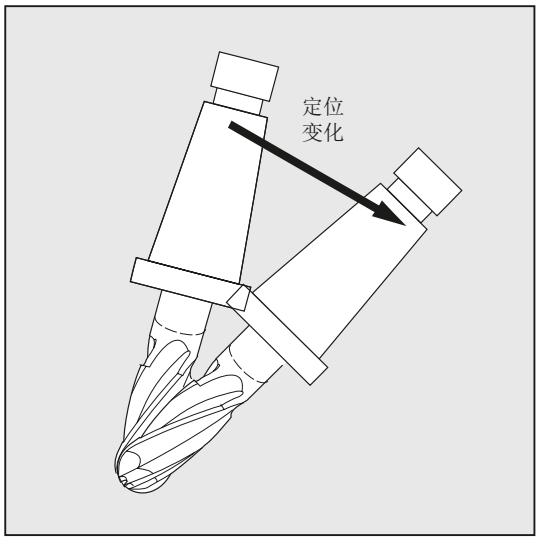
编程

定向变化的编程：

刀具定向的变化可以通过以下方式来编程：

- 直接编程回转轴 A, B, C（回转轴插补）
- 欧拉角或者 RPY 角
- 方向矢量（通过数据 A3 或 B3 或 C3 进行矢量插补）
- LEAD/TILT(端面铣)

参考坐标系既可以是机床坐标系 (ORIMKS) 也可以是当前的工件坐标系 (ORIWKS)。



刀具定向编程：

指令	含义
ORIC:	定向和轨迹运动并行
ORID:	定向和轨迹运动先后进行
OSOF:	没有定向平滑
OSC:	定向恒定
OSS:	只在程序段开始处进行定向平滑
OSSE:	在程序段开始和结束处进行定向平滑
ORIS:	启用定向平滑后方向的改变速度，单位为度或毫米（适用于 OSS 和 OSSE）
OSD:	平滑定向，通过以下设定数据来规定平滑长度： SD42674 $SC_ORI_SMOOTH_DIST
OST:	平滑定向，矢量插补中在以下设定数据中规定角度公差（度）： SD42676 $SC_ORI_SMOOTH_TOL 在回转轴插补中，此处规定的公差被视为定向轴的最大公差。

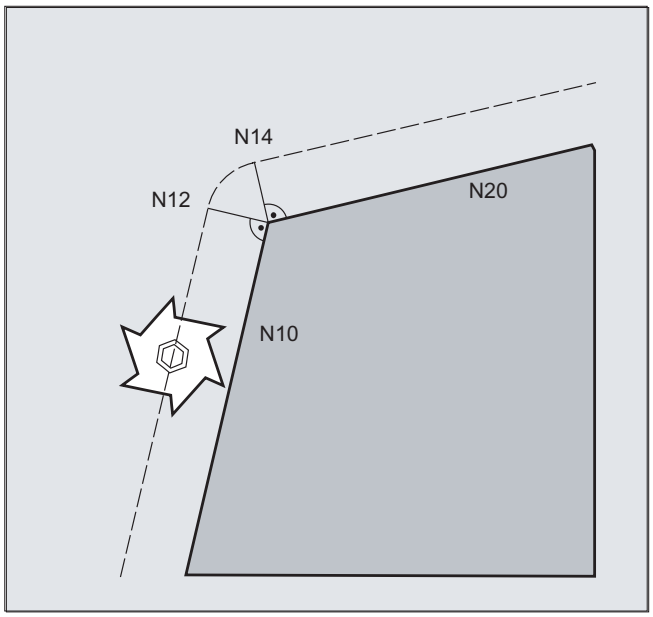
说明

所有用于平滑刀具定向的指令 (OSOF, OSC, OSS, OSSE, OSD 和 OST)都包含在 G 功能组 34 中。 这些指令都是模态生效指令，即只有其中一个指令生效。

示例

示例 1： ORIC

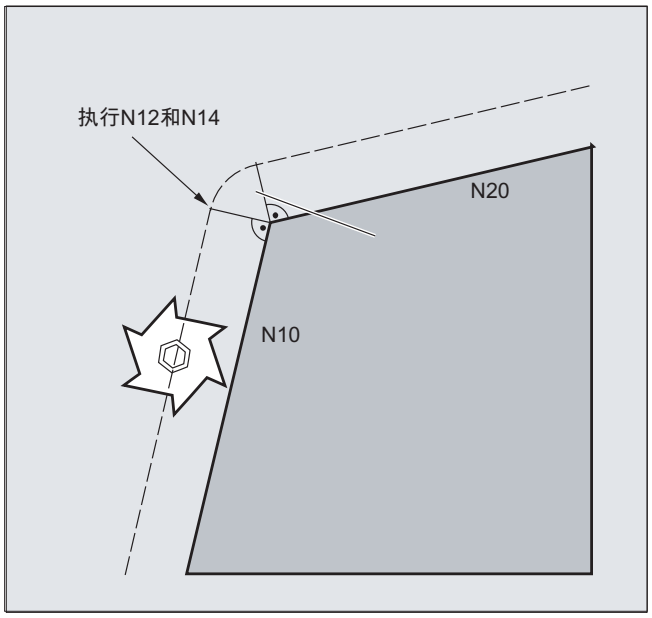
如果运行程序段 N10 和 N20 之间编程了两个或者多个带定向改变的程序段（例如 A2=... B2=... C2=...）且 ORIC 已被激活，则已插入的圆弧程序段按照角度变化量分布在其中的程序段上。



程序代码	注释
ORIC	
N8 A2=… B2=… C2=…	
N10 X… Y… Z…	
N12 C2=… B2=…	; 在外角上插入的圆弧程序段会根据定向改变量分配到 N12 和 N14 上。 圆周运动和定向变化此时会同时执行。
N14 C2=… B2=…	
N20 X =…Y=… Z=… G1 F200	

示例 2： ORID

如果 ORID 已激活，则两个运动程序段之间的所有程序段将在第一个运动程序段结束时执行。 带有恒定定向的圆弧程序段将直接在第二个运动程序段之前执行。



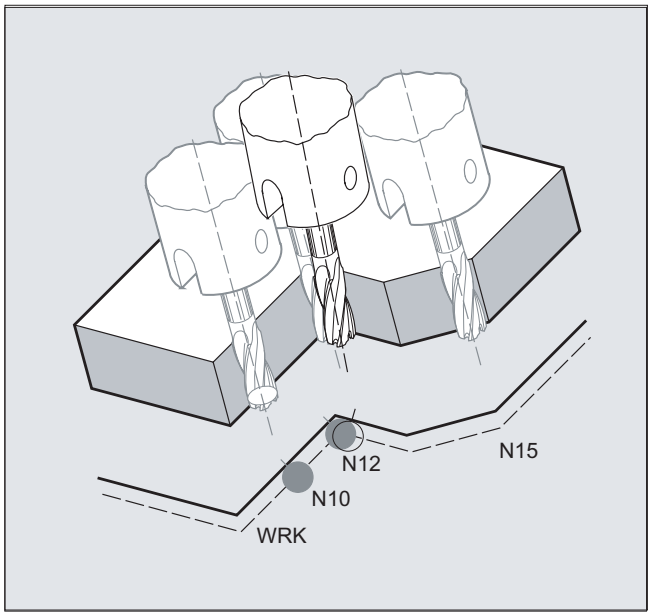
程序代码	注释
ORID	
N8 A2=… B2=… C2=…	
N10 X… Y… Z…	
N12 A2=… B2=… C2=…	; 程序段 N12 和 N14 在 N10 结束处执行。 然后圆弧程序段将以当前的方向运行。
N14 M20	; 辅助功能等等。
N20 X… Y… Z…	

说明

对于某个外角上的定向变化类型而言，起决定性作用的是在外角的第一个运动程序段中激活的程序指令。

没有定向变化： 如果程序段极限上的定向没有变化，则刀具截面是一个接触两个轮廓的圆。

示例 3： 内角上的定向变化



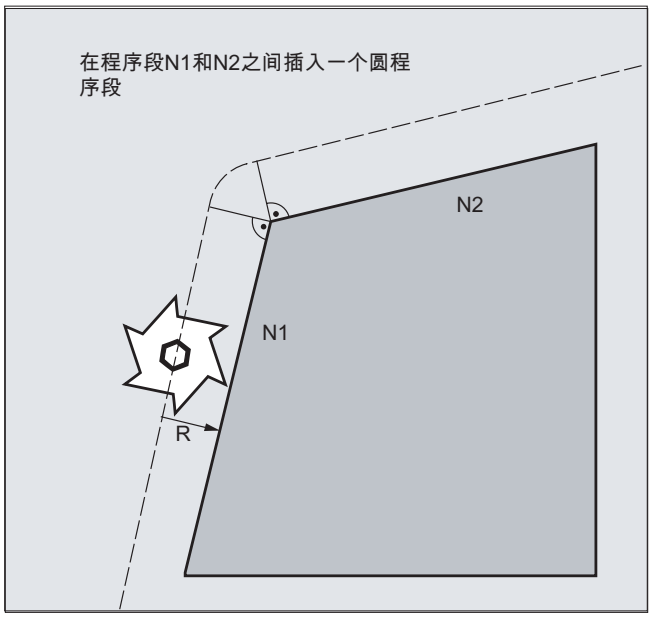
程序代码
ORIC
N10 X …Y… Z… G1 F500
N12 X …Y… Z… A2=… B2=… C2=…
N15 X …Y… Z… A2=… B2=… C2=…

其它信息

外角处的特性

在外角处总是插入一个带铣刀半径的圆弧程序段。

用程序指令 ORIC 或者 ORID 可以确定，在程序段 N1 和 N2 之间已编程的定向变化是否在插入的圆程序段之前还是与该程序段同时执行。



如果在外角处需要一个定向改变，那么定向改变可以选择和插补并行或者和轨迹运动分离来进行。

如果是 ORID ，首先执行已插入的、没有轨迹运动的程序段。 直接在两个运行程序段的第二个之前插入圆弧程序段，通过这两个程序段构成拐角。

如果在外角上插入了多个定向程序段并且 ORIC 已被选中，则圆周运动将根据各个插入程序段的定向变化分配给这些程序段。

用 OSD 或 OST 平滑定向

在用 G642 精磨时，轮廓轴和定向轴的最大偏差区别不会很大。 这两者之间较小的公差确定了平滑运动的形式或者角度公差，相对较强地平滑定向曲线，而避免出现较大的轮廓偏差。

通过激活 OSD 或 OST 可以用一个预设的平滑长度和角度公差，来平滑不明显的轮廓偏差、极小的定向曲线偏差。

说明

与用 G642 精磨轮廓（和定向曲线）不同，在用 OSD 或者 OST 平滑定向时，不再构成自身的程序段，而是直接在编程的原始程序段中插入平滑运动。

用 OSD 或者 OST 不能够平滑程序段过渡，因为在程序段过渡中用于刀具定向（矢量 –> 回转轴，回转轴 –> 矢量）的插补方式会发生变化。 必要时，这些程序段过渡可以用传统的平滑功能 G641、G642 或者 G643 进行平滑。

7.7 任意 D 编号赋值，切削刃编号

7.7.1 任意 D 编号赋值，切削刃编号 (地址 CE)

D 号码

D 编号可以作为补偿编号使用。 除此之外可以通过地址 CE 给切削刃的编号定址。 通过系统变量$TC_DPCE 可以描述切削刃编号。

预设置： 补偿编号 == 切削刃编号

通过机床数据确定 D 号码的最大数量（切削刃编号）和每个刀具的最大切削刃数量（→ 机床制造商）。 只有在确定最大切削刃编号（MD18105）大于每个刀具（MD18106）的切削刃数量时，下面的指令才有意义。 请注意机床制造商的说明。

说明

除了相对 D 号码分配之外，D 号码也可作为“平面”或者“绝对”D 号码（1-32000）在没有参照的情况下分配一个 T 编号（在“平面 D 号码结构”功能范围内）。

文献

功能手册 基本功能;刀具补偿(W1)

7.7.2 任意 D 编号赋值:检查 D 号码(CHKDNO)

功能

用指令 CKKDNO 可以检查现有的 D 号码是否是单一分配。 所有在一个 TO 单元内定义的刀具的 D 编号只能出现一次。 替代刀具在此不考虑。

句法

状态=CHKDNO(Tno1,Tno2,Dno)

含义

状态	=真：	对于检查范围单一的分配 D 编号。
	=假：	产生一个 D 编号的重合或者给定参数无效。通过 Tno1，Tno2 和 D 编号来过渡导致重合的参数。 这些数据可以在零件程序中计算。
CHKDNO(Tno1,Tno2)	检查命名刀具的所有 D 编号。	
CHKDNO(Tno1)	检查 Tno1 的所有 D 编号和其他所有的刀具比较。	
CHKDNO	检查所有刀具的所有 D 编号和其他所有刀具比较。	

7.7.3 任意 D 编号赋值：重命名 D 编号(GETDNO, SETDNO)

功能

D 编号必须单一分配。 一个刀具的两个不同的切削刃不可以有同一个 D 编号。

GETDNO

该指令用来给某个带有 T 编号刀具的特定切削刃提供 D 编号。如果不存在 D 编号给已经输入的参数，则设定 d=0。 如果该 D 编号无效，将返回一个大于 32000 的值。

SETDNO

用这个指令可以给刀具 t 的切削刃 ce 的 D 编号值 d 赋值。 通过 状态 可返回该指令的结果 (正确 或者 错误). 如果没有输入参数的数据程序段，那么给回错误。 句法错误会导致报警。 不可以明显将 D 编号设置为 0。

句法

```
d = GETDNO (t,ce)
状态 = SETDNO (t,ce,d)
```

含义

d	刀具刀刃的 D 编号
t	刀具的 T 编号
ce	刀具的刀刃编号（CE 编号）
状态	说明指令是否可以无误地执行（正确或者错误）。

重命名一个 D 编号举例

编程	注释
$TC_DP2[1.2]=120	;
$TC_DP3[1,2] = 5.5	;
$TC_DPCE[1,2] = 3	; 切削刃编号 CE
...	;
N10 def int D 旧编号, D 新编号 = 17	;
N20 D 旧编号 = GETDNO(1,3)	;
N30 SETDNO(1,3,D 新编号)	;

在此给切削刃 CE=3 赋值新的 D 值 17。 现在通过 D 编号 17 来响应该切削刃的数据; 既可以通过系统变量也可以在编程中使用 NC 地址。

7.7.4 任意 D 编号赋值：求得预先给出 D 编号刀具的 T 编号（GETACTTD）

功能

用指令 GETACTTD 可以相对于一个绝对 D 编号，得出所属的 T 编号。 不检查单一性。 如果在一个 TO 单元内有多个相同的 D 编号，就会返回第一个被发现刀具的 T 编号。 如果使用“平面”D 编号，则使用该指令就没有意义，因为此时始终返回数值 1（没有 T 编号在数据管理器中）。

句法

```
status=GETACTTD(Tnr,Dnr)
```

含义

D 号	D 编号，针对 D 编号应寻找对应的 T 编号。	
T 号	已发现的 T 编号	
状态	值：	含义：
	0	找到 T 编号。 T 号包含 T 编号的值。
	-1	对于说明的 D 编号不存在 T 编号；T 号=0。

-2 D 编号不是绝对的。 Tnr 包含第一个找到的刀具的值，这个刀具包含带有值 Dnr 的 D 编号。

-5 该功能可以由于一个其他的原因不执行。

7.7.5 任意 D 编号赋值：设定无效的 D 编号 (DZERO)

功能

指令 DZERO 用于在重新调整期间提供支持。 这样标记的补偿数据程序段不再由指令 CHKDNO 来检查。 为了重新对其进行访问，必须重新使用 SETDNO 设定 D 编号。

句法

```
DZERO
```

含义

DZERO 将 TO 单元的所有 D 编号标识为无效。

7.8 刀架的运动关系

前提条件

刀架可以给一个刀具在所有可能的空间方向上定向，只有当

- 两个旋转轴 V_1 和 V_2 均存在。
- 旋转轴相互垂直。
- 刀具纵轴垂直于第二个旋转轴 V_2。

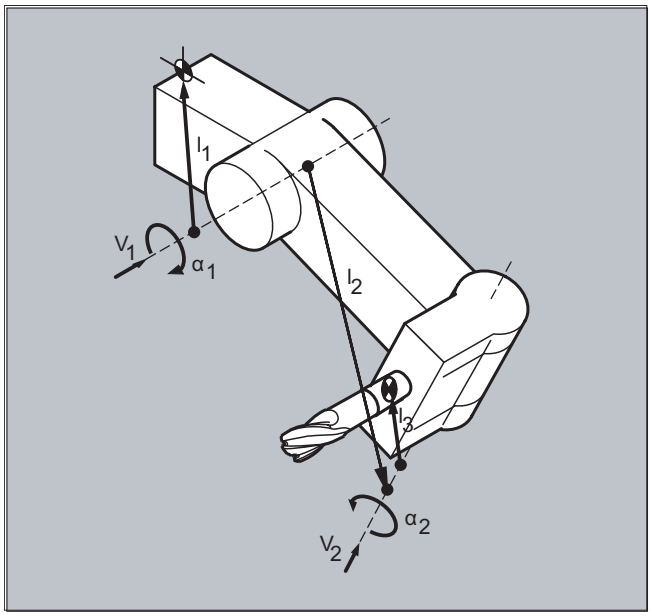
除此之外在使用机床时，在使用这些机床时所有可能的定向必须是可调节的，有下列要求：

- 刀具定向必须垂直于第一个旋转轴 V_1。

功能

带有最多两个旋转轴的刀架运动 v_1 或者 v_2 可通过 17 个系统变量 $TC_CARR1[m] ～ $TC_CARR17[m] 进行描述。 刀架的描述由以下几部分组成：

- 从第一个旋转轴到刀架参照点的矢量距离 I_1，从第一个旋转轴到第二个旋转轴的矢量距离 I_2，从第二个旋转轴到刀具参照点的矢量距离 I_3.
- 两个旋转轴的方向矢量 V_1，V_2.
- 围绕两个轴的旋转角 α_1，α_2。 旋转角以视角方向沿旋转轴矢量的方向顺时针正向来计算。



对于具有**分解运动** 的机床(刀具以及工件均可以转动)，系统变量已经以

- `$TC_CARR18[m]` ～ `$TC_CARR23[m]` 得到扩充。

参数

可定向刀架系统变量的功能			
名称	**x-分量**	**y-分量**	**z-分量**
l_1 偏移矢量	$TC_CARR1[m]	$TC_CARR2[m]	$TC_CARR3[m]
l_2 偏移矢量	$TC_CARR4[m]	$TC_CARR5[m]	$TC_CARR6[m]
v_1 旋转轴	$TC_CARR7[m]	$TC_CARR8[m]	$TC_CARR9[m]
v_2 旋转轴	$TC_CARR10[m]	$TC_CARR11[m]	$TC_CARR12[m]
α_1 旋转角 α_2 旋转角	$TC_CARR13[m] $TC_CARR14[m]		
l_3 偏移矢量	$TC_CARR15[m]	$TC_CARR16[m]	$TC_CARR17[m]

可定向刀架系统变量的扩展			
名称	**x-分量**	**y-分量**	**z-分量**
l_4 偏移矢量	$TC_CARR18[m]	$TC_CARR19[m]	$TC_CARR20[m]
轴标识符 旋转轴 v_1 旋转轴 v_2	旋转轴 v_1 和 v_2 的轴标识符 (默认设置为零) $TC_CARR21[m] $TC_CARR22[m]		
运动关系类型 **T**ool **P**art **M**ixed mode	$TC_CARR23[m]		
	运动类型-T ->	运动类型-P ->	运动关系类型-M
	仅刀具可以旋转（默认设置）	仅工件可以旋转	工件和刀具均可以旋转
偏移，旋转轴 v_1 旋转轴 v_2	旋转轴 v_1 和 v_2 的角度（单位：度），当收到基本位置时 $TC_CARR24[m] $TC_CARR25[m]		
角偏移，旋转轴 v_1 旋转轴 v_2	旋转轴 v_1 和 v_2 的圆锥齿圈的偏移量（单位：度） $TC_CARR26[m] $TC_CARR27[m]		

可定向刀架系统变量的扩展			
角度增量 v_1 旋转轴 v_2 旋转轴	旋转轴 v_1 和 v_2 圆锥齿圈的增量 $TC_CARR28[m] $TC_CARR29[m]		
最小位置 旋转轴 v_1 旋转轴 v_2	旋转轴 v_1 和 v_2 的最小位置软件极限值 $TC_CARR30[m] $TC_CARR31[m]		
最大位置 旋转轴 v_1 旋转轴 v_2	旋转轴 v_1 和 v_2 最大位置的软件极限值 $TC_CARR32[m] $TC_CARR33[m]		
刀架名称	代替一个数字，刀架可以获得一个名称。 $TC_CARR34[m]		
用户： 轴名称 1 轴名称 2 特征 **定位**	在用户的测量循环之内有意使用 $TC_CARR35[m] $TC_CARR36[m] $TC_CARR37[m]		
	$TC_CARR38[m]	$TC_CARR39[m]	$TC_CARR40[m]
精密 **位移**	可以添加给基本参数中的**数值** 的参数。		
l_1 偏移矢量	$TC_CARR41[m]	$TC_CARR42[m]	$TC_CARR43[m]
l_2 偏移矢量	$TC_CARR44[m]	$TC_CARR45[m]	$TC_CARR46[m]
l_3 偏移矢量	$TC_CARR55[m]	$TC_CARR56[m]	$TC_CARR57[m]
l_4 偏移矢量	$TC_CARR58[m]	$TC_CARR59[m]	$TC_CARR60[m]
v_1 旋转轴	$TC_CARR64[m]		
v_2 旋转轴	$TC_CARR65[m]		

说明

有关参数的解释

使用 "m" 可相应说明需要描述的刀架的编号。

$TC_CARR47 ～ $TC_CARR54 以及 $TC_CARR61 ～ $TC_CARR63 未定义且当尝试对其进行读写访问时会导致报警。

轴上距离矢量的起始点和终点可以自由选择。围绕两个轴的旋转角 α_1，α_2 在刀架的基本状态中被定义为 0° 。 刀架的运动关系可以有任意多种可能性来描述。

只有一个旋转轴或者没有旋转轴的刀架可以通过一个或者两个旋转轴方向矢量的零设置来描述。

如果是一个没有旋转轴的刀架，距离矢量的作用如同附加的刀具补偿，其分量在转换加工平面时(G17 ～ G19)° 不受影响。

参数的扩展

旋转轴的参数

系统变量已经按照输入记录 $TC_CARR24[m] ～ $TC_CARR33[m] 扩展且描述如下：

旋转轴**偏移** v_1, v_2	当可定向的刀具处于基本位置时，旋转轴 v_1 或者 v_2 的位置变化。
角度偏移/角度增量 旋转轴 v_1, v_2	旋转轴 v_1 和 v_2 的圆周齿圈的偏移量或者角度增量。 倒圆编程设计的或者计算出的角度到下一个值，这个值从来自公式 phi = s + n * d 的整数的 n 得出。
最小和最大位置 旋转轴 v_1, v_2	旋转轴的最小位置/最大位置 旋转轴 v1 和 v2 的极限角度（软件极限值）。

用户参数

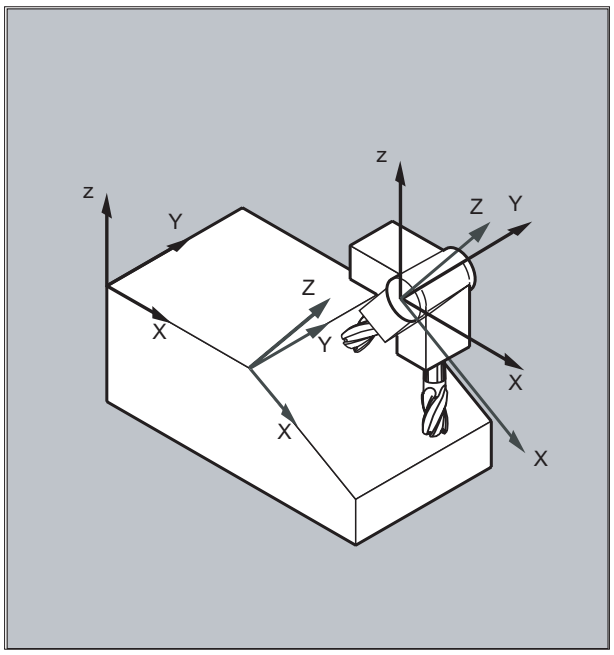
$TC_CARR34 至 $TC_CARR40 包含参数，这些参数可供用户任意使用并且软件版本 6.4 以下时默认在 NCK 之内不会被继续分析或者没有含义。

精偏移参数

$TC_CARR41 至 $TC_CARR65 包含精偏移参数，这些参数以数值方式可以添加到基本参数中。 当将数值 40 添加给参数编号时，得出分配给某个基本参数的精密位移值。

示例

下列举例中所使用的刀架可通过围绕 Y 轴旋转来完整描述。



程序代码	注释
N10 $TC_CARR8[1]=1	; 定义刀架 1 第一个旋转轴的 Y 分量。
N20 $TC_DP1[1,1]= 120	; 定义某个立铣刀。
N30 $TC_DP3[1,1]=20	; 定义一个长度为 20 mm 的立铣刀。
N40 $TC_DP6[1,1]=5	; 定义一个半径为 5 mm 的立铣刀。
N50 ROT Y37	; 以围绕 Y 轴旋转 37°来定义框架。
N60 X0 Y0 Z0 F10000	; 返回运行到出发位置。
N70 G42 CUT2DF TCOFR TCARR=1 T1 D1 X10	; 在旋转后的框架中设置半径补偿、刀架长度补偿，选择刀架 1，刀具 1。
N80 X40	; 在旋转 37°的情况进行加工。
N90 Y40	
N100 X0	
N110 Y0	
N120 M30	

其它信息

分解运动

对于具有分解运动的机床（刀具和工件均可旋转），系统变量均已经以输入记录 $TC_CARR18[m] ～ $TC_CARR23[m] 得到扩展且描如下：

可旋转的刀具台由以下几部分组成：

- 第二个旋转轴 V_2 相对于第三个旋转轴的一个可旋转刀具台参考点的矢量距离 I_4。

回转轴由以下几项组成：

- 两个用于旋转轴 V_1 和 V_2 参照点的通道标识符，在确定可定位刀架的定位时有可能要访问这些旋转轴的位置。

带有值 T，P 或者 M 其中之一的运动关系类型：

- 运动类型 T： 只有刀具是可旋转的。
- 运动类型 P： 只有工件是可旋转的。
- 运动类型 M： 工件和刀具均可以旋转

删除刀架数据

使用 $TC_CARR1[0] = 0$ 可以删除所有刀架数据记录的数据。

运动类型 $TC_CARR23[T] = T$ 必须配置三个允许大写或者小写字母 (T,P,M) 中的一个字母且处于这个原因不得被删除。

修改刀架数据

每个描述的值都可以通过零件程序中一个新的值的分配来改变。 每个其他不是 T，P 或者 M 的标志，在尝试激活可定向刀架时，会产生报警。

读取刀架数据

每个被描述的值可以通过对零件程序中变量的赋值来读取。

精密位移

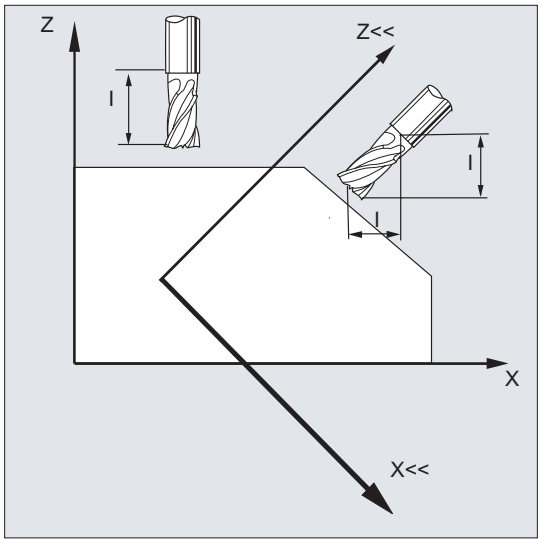
当激活一个可定向的刀架时，才会识别一个非法的精密位移值，该刀架含有一个此类数值同时还有设置数据 SD42974 $SC_TOCARR_FINE_CORRECTION = TRUE。

允许的精偏移的量可以通过机床数据限制在一个最大允许值上。

7.9 用于可定向刀架的刀具长度补偿(TCARR, TCOABS, TCOFR, TCOFRX, TCOFRY, TCOFRZ)

功能

刀具的空间方向改变后，刀具长度分量也一起变化。



重新安装后，比如使用固定的空间定向手动设定或者更换刀架，则必须重新计算刀具长度分量。 这可以通过位移指令 TCOABS 和 TCOFR 进行。

在有效框架中，对于可以定向的刀架，在用 TCOFRZ、TCOFRY 和 TCOFRX 选择刀具时，可以确定刀具的方向。

句法

```
TCARR=[<m>]
TCOABS
TCOFR
TCOFRZ
TCOFRY
TCOFRX
```

含义

TCARR=[<m>]:	要求带“m”的刀架
TCOABS:	从当前刀架的方向计算刀具长度分量。
TCOFR:	从当前框架的方向确定刀具长度分量
TCOFRZ:	有效的框架中可定向的刀架，其刀具指向 Z 方向
TCOFRY:	有效的框架中可定向的刀架，其刀具指向 Y 方向
TCOFRX:	有效的框架中可定向的刀架，其刀具指向 X 方向

其它信息

从刀架方向确定刀具长度补偿（TCOABS）

TCOABS 从刀架当前的方向角计算刀具长度补偿；存储在系统变量 $TC_CARR13 和 $TC_CARR14 中。

有关使用系统变量定义刀架的运动性能，请参见 “刀架运动关系 (页 449)”。

在框架更换后，为了重新计算刀具长度补偿，必须再次选择刀具。

当前框架的刀具方向

可以对可定向刀架进行设置，使得刀具指向下列方向：

- 通过 TCOFR 或 TCOFRZ 在 Z 方向
- 通过 TCOFRY 在 Y 方向
- 通过 TCOFRX 在 X 方向

在 TCOFR 和 TCOABS 之间进行转换，会引起刀具长度补偿的重新计算。

刀架要求（TCARR）

使用 TCARR，要求刀架号 m 连同其几何数据（补偿存储器）。

当 m=0 时，撤销选择当前的刀架。

只有在调用一个刀具之后，刀架的几何数据才有效。 在更换一个刀架后，所选择的刀具仍然有效。

刀架当前的几何数据也可以在零件程序中通过相应的系统变量进行定义。

在框架更换时重新计算刀具长度补偿 (TCOABS)

在框架更换后，为了重新计算刀具长度补偿，必须再次选择刀具。

说明

刀具方向必须手动匹配到当前的框架。

在计算刀具长度补偿时，可以在中间步骤计算刀架的转角。 带两个旋转轴的刀架通常有两个旋转角组，它们可以使刀具方向与当前的框架相适应。系统变量中存储的转角值至少必须与机械设定的转角相近似。

说明

刀具定向

控制系统不可以检查机床上旋转角的设定，旋转角可以通过框架定向进行计算。

如果刀架的旋转轴由于结构的原因，不能达到通过框架定向所计算得到的刀具方向，则发出一个报警。

刀具精确补偿不可以与运动刀架的刀具长度补偿功能相结合。 如果试图同时调用两个功能，则发出一个报警。

使用 TOFRAME 可以根据所选刀架的方向定义一个框架。 详细的信息参见章节“框架”。

在方向变换生效时（3、4、5 轴变换）可以选择一个与零位置方向偏离的刀架，而不产生一个报警。

标准循环和测量循环的传递参数

已定义的值域适用于标准循环和测量循环的传递参数。

角度值域如下：

- 围绕第 1 个几何轴旋转： -180 度 到 +180 度
- 围绕第 2 个几何轴旋转： -90 度 到 +90 度
- 围绕第 3 个几何轴旋转： -180 度 到 +180 度

参见章节框架，“可编程的旋转 （ROT，AROT，RPL）”

说明

将角度值传递到标准循环或测量循环时要注意：

小于 NC 计算精度的值将取整为零！

角度位置的 NC 计算精度在机床数据中确定：

MD10210 $MN_INT_INCR_PER_DEG

7.10 在线式刀具长度补偿 (TOFFON, TOFFOF)

功能

通过系统变量 $AA_TOFF[<n>] 可根据三个刀具方向实时三维叠加有效的刀具长度。

三个几何轴标识符作为索引 **<n>** 使用。 这样，当前有效的补偿方向的数量便可以由同时有效的几何轴来确定。

所有的补偿可以同时有效。

功能“在线刀具长度补偿”可以在以下情况应用：

- 方向转换 TRAORI
- 可定向的刀架 TCARR

说明

在线刀具长度补偿是一个**选项**，必须事先已经激活。 只有在与一个激活的方向转换功能或者一个激活的可定向刀架配合使用时，该功能才会有效。

句法

```
TRAORI
TOFFON(<补偿方向>[,<偏移值>])
WHEN TRUE DO $AA_TOFF[<补偿方向>]                    ; 在同步动作中。
...
TOFFOF(<补偿方向>)
```

运动同步动作中的在线刀具长度补偿编程的更多相关信息请参见“在线刀具长度补偿（$AA_TOFF[刀具方向]） (页 612)”。

含义

TOFFON:	**激活**在线刀具长度补偿	
	<补偿方向>:	在线刀具长度补偿生效的刀具方向（X、Y、Z）。
	<偏移值>:	在激活时可以针对相应的补偿方向指定一个立即执行的偏移值。

TOFFOF: **取消**在线刀具长度补偿

针对指定补偿方向上的补偿值被取消，并触发预处理停止。

示例

示例 1：选择刀具长度补偿

程序代码	注释
MD21190 $MC_TOFF_MODE =1	; 逼近绝对值。
MD21194 $MC_TOFF_VELO[0] =1000	
MD21196 $MC_TOFF_VELO[1] =1000	
MD21194 $MC_TOFF_VELO[2] =1000	
MD21196 $MC_TOFF_ACCEL[0] =1	
MD21196 $MC_TOFF_ACCEL[1] =1	
MD21196 $MC_TOFF_ACCEL[2] =1	
N5 DEF REAL XOFFSET	
N10 TRAORI(1)	; 坐标转换激活。
N20 TOFFON(Z)	; 激活用于 Z 轴刀具方向的在线刀具长度补偿。
N30 WHEN TRUE DO $AA_TOFF[Z]=10 G4 F5	; 为 Z 轴刀具方向插补一个大小为 10 的刀具长度补偿。
...	
N100 XOFFSET=$AA_TOFF_VAL[X]	; 在 X 方向分配当前补偿。
N120 TOFFON(X,-XOFFSET) G4 F5	; 将 X 轴刀具方向的刀具长度补偿复位为 0。

示例 2：取消刀具长度补偿

程序代码	注释
N10 TRAORI(1)	; 坐标转换激活。
N20 TOFFON(X)	; 激活用于 X 轴刀具方向的在线刀具长度补偿。
N30 WHEN TRUE DO $AA_TOFF[X] = 10 G4 F5	; 为 X 轴刀具方向插补一个大小为 10 的刀具长度补偿。
...	
N80 TOFFOF(X)	; 删除 X 轴刀具方向的位置偏移： ...$AA_TOFF[X]=0 不运行轴。 根据当前的定位将位置偏移添加至 WCS 中的当前位置。

其它信息

程序段预处理

在程序段预处理时，会一同考虑主处理中生效的刀具长度偏移。 为了进一步充分利用允许的最大轴速度，需要用预处理停止 STOPRE 在产生刀具偏移的这段时间里来暂停预处理。

如果刀具长度补偿在程序启动之后不再改变，或者此之后又执行多个程序段，而这些程序段的数目已经超出了预处理和主处理之间 IPO 缓冲器的空间，在预处理时该偏移量始终会纳入计算。

变量$AA_TOFF_PREP_DIFF

当前插补器中生效的补偿，和程序段预处理时生效的补偿之间的差值可在变量 $AA_TOFF_PREP_DIFF[<n>] 中查询。

设置机床数据和设定数据

对于在线刀具长度补偿可以使用以下系统数据：

- MD20610 $MC_ADD_MOVE_ACCEL_RESERVE（叠加运行的加速度预留量）
- MD21190 $MC_TOFF_MODE

  系统变量 $AA_TOFF[<n>] 的内容作为绝对值处理或者相加。
- MD21194 $MC_TOFF_VELO（在线刀具长度补偿的速度）
- MD21196 $MC_TOFF_ACCEL（在线刀具长度补偿的加速度）
- 用于设定限值的设定数据：
  SD42970 $SC_TOFF_LIMIT（刀具长度补偿上限）

文献：
功能手册 特殊功能；F2：多轴转换

7.11 可旋转刀具的切削刃数据修改 (CUTMOD)

功能

通过功能“可旋转刀具的切削刃数据修改”，可以得出刀具（主要是车刀，但是也包括钻头和铣刀）旋转后相对于待加工的工件发生的几何数据变化，这些变化会纳入刀具补偿范围内。

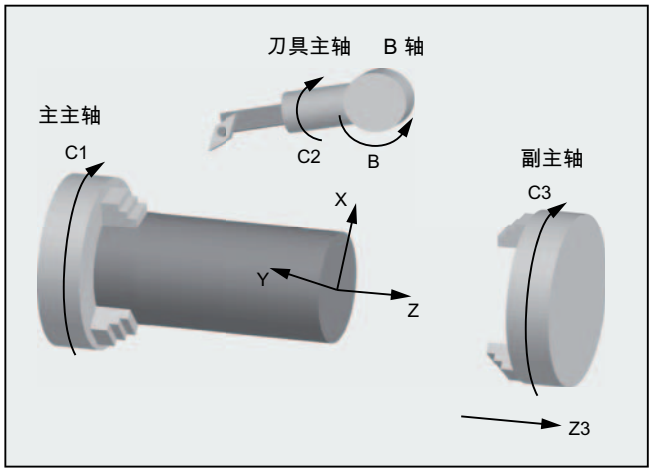


图 7-1 车床上可旋转刀具

此时当前的刀具旋转总是由一个当前激活的可定向的刀架（参见“可定向刀架刀具补偿 (页 455)”）确定。

该功能通过指令 CUTMOD 激活。

句法

```
CUTMOD=<值>
```

含义

CUTMOD	用于打开功能“可旋转刀具的切削刃数据修改”的指令	
<值>	可以为 CUTMOD 指令赋予下列值：	
	0	取消激活该功能。 由系统变量 $P_AD... 提供的值和相应的刀具参数相同。

> 0　如果一个指定的可定向刀架激活，则该功能也激活，即该功能的启用和特定的可定向刀架绑定。

由系统变量 $P_AD... 提供的值不和相应的刀具参数相同，而是取决于当前有效的旋转发生变化。

禁用某个指定的可定向刀架会暂时锁定该功能，启用另一个刀架会持续锁定该功能。 因此，在第一种情况下，再次选中指定刀架即可再次激活功能，而在第二种情况下，即使再次选中指定刀架，还需要重新选择。

该功能不受复位操作的影响。

-1　如果一个可定向的刀架激活，该功能总是激活。

在更换刀架，或取消选择并接着再次选中时，CUTMOD 不必重新设置。

-2　如果一个可定向刀架激活，而它的号码与当前的可定向刀架相同，该功能总是激活。

如果没有可定向刀架激活，则该设置相当于 CUTMOD=0。 如果一个可定向刀架激活，则该设置相当于直接给定当前刀架号。

< -2　小于 -2 的值被忽略，即该值相当于没有编程 CUTMOD。

提示：
该取值范围预留用于将来的功能扩展，请勿使用。

说明

SD42984 $SC_CUTDIRMOD

指令 CUTMOD 可以激活的功能替代了原来由设定数据 SD42984 $SC_CUTDIRMOD 激活的功能。 然而，原先的功能仍被保留。 但是同时使用两个功能又没有意义，因此，只有当 CUTMOD 等于零时，它才被激活。

示例

下面的例子采用的是一个带有切削位置 3 的刀具和一个可定向的刀架，该刀架可以使刀具绕着 B 轴旋转。

括号中的数值总是按顺序 X, Y, Z 规定了机床坐标系 (MCS) 中的程序段位置。

程序代码	注释
N10 $TC_DP1[1,1]=500	
N20 $TC_DP2[1,1]=3	; 刀沿位置

程序代码	注释		
N30 $TC_DP3[1,1]=12			
N40 $TC_DP4[1,1]=1			
N50 $TC_DP6[1,1]=6			
N60 $TC_DP10[1,1]=110	; 夹角		
N70 $TC_DP11[1,1]=3	; 切削方向		
N80 $TC_DP24[1,1]=25	; 后角		
N90 $TC_CARR7[2]=0 $TC_CARR8[2]=1 $TC_CARR9[2]=0	; B 轴		
N100 $TC_CARR10[2]=0 $TC_CARR11[2]=0 $TC_CARR12[2]=1	; C 轴		
N110 $TC_CARR13[2]=0			
N120 $TC_CARR14[2]=0			
N130 $TC_CARR21[2]=X			
N140 $TC_CARR22[2]=X			
N150 $TC_CARR23[2]="M"			
N160 TCOABS CUTMOD=0			
N170 G18 T1 D1 TCARR=2	X	Y	Z
N180 X0 Y0 Z0 F10000	; 12.000	0.000	1.000
N190 $TC_CARR13[2]=30			
N200 TCARR=2			
N210 X0 Y0 Z0	; 10.892	0.000	-5.134
N220 G42 Z-10	; 8.696	0.000	-17.330
N230 Z-20	; 8.696	0.000	-21.330
N240 X10	; 12.696	0.000	-21.330
N250 G40 X20 Z0	; 30.892	0.000	-5.134
N260 CUTMOD=2 X0 Y0 Z0	; 8.696	0.000	-7.330
N270 G42 Z-10	; 8.696	0.000	-17.330
N280 Z-20	; 8.696	0.000	-21.330
N290 X10	; 12.696	0.000	-21.330
N300 G40 X20 Z0	; 28.696	0.000	-7.330
N310 M30			

说明：

CUTMOD=0：程序段 N180 中，先选择刀具，没有选择可定向的刀架。 由于可定向刀架的所有偏移坐标为 0，所以轴逼近某个符合 $TC_DP3[1,1] 和 $TC_DP4[1,1] 中规定刀具长度的位置。

在程序段 N200 中，激活绕着 B 轴旋转 30°可定向的刀架。 因为 CUTMOD=0，切削刃长度没有被修改，所以还是参照以前的切削参考点。 因此在程序段 N210 中，轴逼近某个零点中保留了旧切削参考点的位置，即矢量 (1, 12) 在 Z/X 平面内旋转 30°。

和程序段 N200 不同，在 N260 中，CUTMOD=2。 由于可定向刀架的旋转，到达经过修改的切削刃位置 8。由此出现偏差的轴位置。

在程序段 N220 或 N270 中总是激活刀具半径补偿 (WRC)。 两个程序块中不同的切削刃位置对 WCR 作用的程序段的最终位置没有影响，对应的位置因此相同。 仅在程序段 N260 或 N300 中，不同的切削刃位置才有影响。

其它信息

修改的切削刃数据有效性

修改的切削刃位置和切削刃参考点在编程时也立即生效于已经激活的刀具。 为此无需选择刀具。

影响激活的工作平面

为了确定修改的切削刃位置，切削方向和夹角和后角，有必要注意各个激活级面 (G17 - G19) 中的切削刃。

如果设置数据 SD42940 $SC_TOOL_LENGTH_CONST（在级面更换时更换刀具长度分量）还包括一个有效值不等于零（正或负 17、18 或 19），则该级面确定其内容，在该级面中注意相关的尺寸。

系统变量

可提供下列系统变量：

系统变量	含义
$P_CUTMOD_ANG / $AC_CUTMOD_ANG	在激活的加工层面中提供（未倒圆）角，其对于修改切削刃数据（切削刃位置，切削方向，后角和夹角）在用 CUTMOD 或 $SC_CUTDIRMOD 激活的功能时作为基础。 $P_CUTMOD_ANG 与进给时的当前状态有关， $AC_CUTMOD_ANG 与当前主运行程序段有关。
$P_CUTMOD / $AC_CUTMOD	读取当前有效值，该值最近通过指令 CUTMOD 已编程（要激活切削刃数据修改的刀架号码）。 如果最近编程的 CUTMOD 值 = -2（用当前激活的可定向的刀架激活），则在 $P_CUTMOD 中值不返回 -2，而返回用于编程时间点的激活的可定向刀架号码。 $P_CUTMOD 与进给时的当前状态有关，$AC_CUTMOD 与当前主运行程序段有关。
$P_CUT_INV / $AC_CUT_INV	如果刀具旋转，主轴旋转方向必须反转，则值为真。 为此在相关各个读取操作的程序段中必须满足下列 4 个条件： 1. 车刀或铣刀激活 （刀具类型 400 至 599 和/或 r SD42950 $SC_TOOL_LENGTH_TYPE = 2）。 2. 切削刃影响通过语言指令 CUTMOD 激活。 3. 一个可定向的刀架激活，该刀架通过数值由 CUTMOD 标出。 4. 可定向的刀架使刀具绕着加工级面中的轴旋转（典型方式为 C 轴），得出的正常刀具切削刃相对于输出位置旋转超过 90°（典型方式为 180°）。 如果四个所述条件中至少有一个不满足，则变量内容为假。 对于未定义切削刃位置的刀具，变量值总是为假。 $P_CUT_INV 与进给时的当前状态有关，$AC_CUT_INV 与当前主运行程序段有关。

所有主运行变量 ($AC_CUTMOD_ANG, $AC_CUTMOD und $AC_CUT_INV) 可以读取在同步动作中。 从进给读入存取生成一个进给停止。

修改的切削刃数据：

如果刀具旋转激活，修改的数据可提供下列系统变量：

系统变量	含义
$P_AD[2]	刀沿位置
$P_AD[10]	夹角
$P_AD[11]	切削方向
$P_AD[24]	后角

说明

当功能“可旋转刀具的切削刃数据修改”通过指令 CUTMOD 激活且一个可定向的影响刀具旋转的刀架激活时，数据总是根据相应的刀具参数（$TC_DP2[..., ...] 等）修改。

文献

有关功能“可旋转刀具的切削刃数据修改”的其他信息参见：

功能手册 基本功能;刀具补偿(W1)

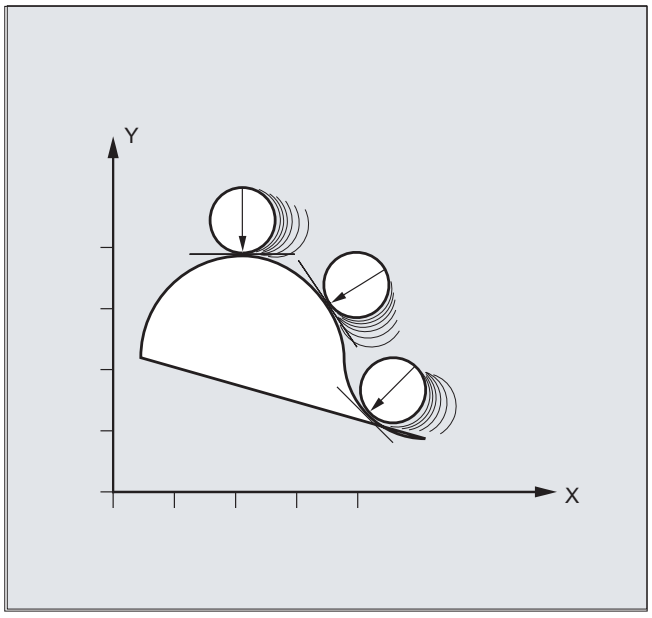
轨迹特性

8

8.1 切向控制 (TANG, TANGON, TANGOF, TLIFT, TANGDEL)

功能

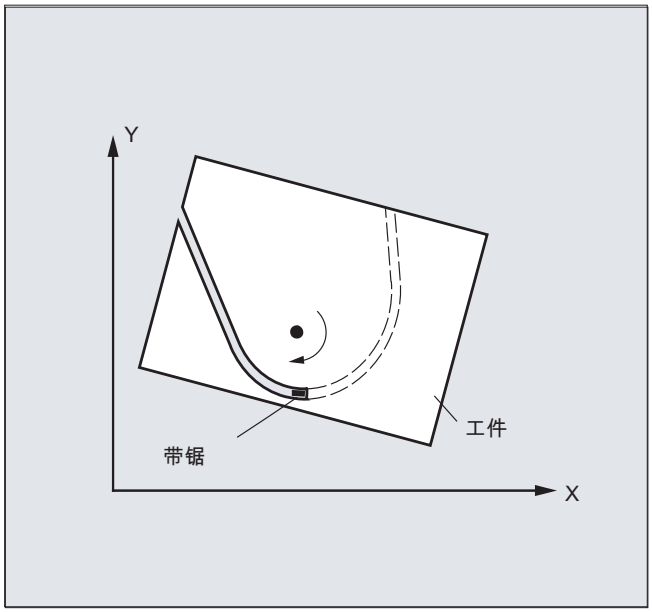
按照引导轴确定的轨迹的切线，跟随轴跟踪运行。 由此一个刀具可以调整到与轮廓平行。 通过 TANGON 指令中编程的角度，刀具可以相对于切线定位。



应用

切向控制例如可用于：

- 步冲时可旋转刀具的切向调整
- 使用带锯时对工件校准的跟踪（见下图）
- 按照砂轮调整修整刀具
- 玻璃或纸张加工用的小切削轮的调整
- 当进行 5 轴焊接时以切向送入一根棒材



句法

定义切向跟踪：
TANG(<跟随轴>,<引导轴 1>,<引导轴 2>,<耦合系数>,<坐标系>,<优化>)

激活切向控制：
TANGON(<跟随轴>,<角度>,<距离>,<角度公差>)

取消切向控制：
TANGOF(<跟随轴>)

激活“在轮廓拐角处插入中间程序段”功能：
TLIFT(<跟随轴>)

TLIFT 指令在使用 TANG(…) 设定轴分配后使用。

取消“在轮廓拐角处插入中间程序段”功能：

重复 TANG(...) 指令，之后不编程 TLIFT(<跟随轴>)。

删除一个切向跟踪的定义：
TANGDEL(<跟随轴>)

在用准备指令 TANG 定义一个新的切向跟踪时，如果需要使用相同的跟随轴，必须删除现有的用户自定义切向跟踪。 只有在使用 TANGOF(<跟随轴>) 取消了耦合后才能执行删除。

含义

TANG:	定义切向跟踪的准备指令
TANGON:	激活指定跟随轴的切向控制
TANGOF:	取消指定跟随轴的切向控制
TLIFT:	激活“在轮廓拐角处插入中间程序段”功能
TANGDEL:	删除一个切向跟踪的定义
<跟随轴>:	跟随轴 跟踪切向的附加回转轴
<引导轴 1>,<引导轴 2>:	引导轴 路径轴，切向被跟踪的轴
<耦合系数>:	耦合系数 切线角度的变化与跟随轴之间的关系 缺省设置： 1 **提示：** 不必明确写入耦合系数 1。
<坐标系>:	坐标系的标识符 “B”: 基本坐标系（缺省设置） **提示:** 不必明确编程 **<**坐标系> = "B"。 “W”: 工件坐标系（不可使用）
<优化>:	优化 “S”: 标准（缺省设置） **提示:** 不必明确编程 **<**优化> = "S"。 “P”: 自动调整切向轴的时间曲线和轮廓 **提示** 编程 **<**优化> = "P" 时，跟随轴的动态特性受引导轴的速度限制影响。 尤其是在使用坐标转换时，建议采用此设置。
<角度>:	跟随轴的偏移角
<距离>:	跟随轴的平滑距离（<优化> **=** "P"时需要此设置）
<角度公差>:	跟随轴的角度公差（可选，仅在 <优化> **=** "P"时分析） **提示：** 参数 **<**距离> 和 <角度公差> 用于限制跟随轴和引导轴切线之间的误差。

示例

示例 1： 定义并激活切向跟踪

程序代码	注释
N10 TANG(C,X,Y,1,"B","P")	; 定义切向跟踪：需要使回转轴 C 跟踪几何轴 X 和 Y。
N20 TANGON(C,90)	; C 轴为跟随轴。 在路径轴的每个运动中，它都转到和路径切线垂直 90°的位置。
...	

说明

简化编程

TANG(C,X,Y,1,"B","P") 可简化编程为 TANG(C,X,Y,,,"P")。

示例 2：平面切换

程序代码	注释
N10 TANG(A,X,Y,1)	; 切向跟踪的第 1 个定义。
N20 TANGON(A)	; 激活耦合。
N30 X10 Y20	; 半径
...	
N80 TANGOF(A)	; 取消第 1 个耦合。
N90 TANGDEL(A)	; 删除第 1 个定义。
...	
TANG(A,X,Z)	; 切向跟踪的第 2 个定义。
TANGON(A)	; 激活新的耦合。
...	
N200 M30	

示例 3： 几何轴切换和 TANGDEL

不产生报警。

程序代码	注释
N10 GEOAX(2,Y1)	; Y1 为几何轴 2。
N20 TANG(A,X,Y)	; 切向跟踪的第 1 个定义。
N30 TANGON(A,90)	; 激活带有 Y1 的跟踪。

程序代码	注释
N40 G2 F8000 X0 Y0 I0 J50	
N50 TANGOF(A)	; 取消带有 Y1 的跟踪。
N60 TANGDEL(A)	; 删除第 1 个定义。
N70 GEOAX(2, Y2)	; Y2 为新的几何轴 2。
N80 TANG(A,X,Y)	; 切向跟踪的第 2 个定义。
N90 TANGON(A,90)	; 激活带有 Y2 的跟踪。
...	

示例 4： 带有自动优化的切向跟踪

Y1 为几何轴 2。

程序代码	注释
...	
N80 G0 C0	
N100 F=50000	
N110 G1 X1000 Y500	
N120 TRAORI	
N130 G642	; 按照允许的最大路径偏差开展平滑
N171 TRANS X50 Y50	
N180 TANG(C,X,Y,1,,"P")	; 切向跟踪的定义，包含路径速度的自动优化
N190 TANGON(C,0,5.0,2.0)	; 激活带自动优化的切向跟踪：平滑距离 5 mm，角度公差 2°。
N210 G1 X1310 Y500	
N215 G1 X1420 Y500	
N220 G3 X1500 Y580 I=AC(1420) J=AC(580)	
N230 G1 X1500 Y760	
N240 G3 X1360 Y900 I=AC(1360) J=AC(760)	
N250 G1 X1000 Y900	
N280 TANGOF(C)	
N290 TRAFOOF	
N300 M02	

其它信息

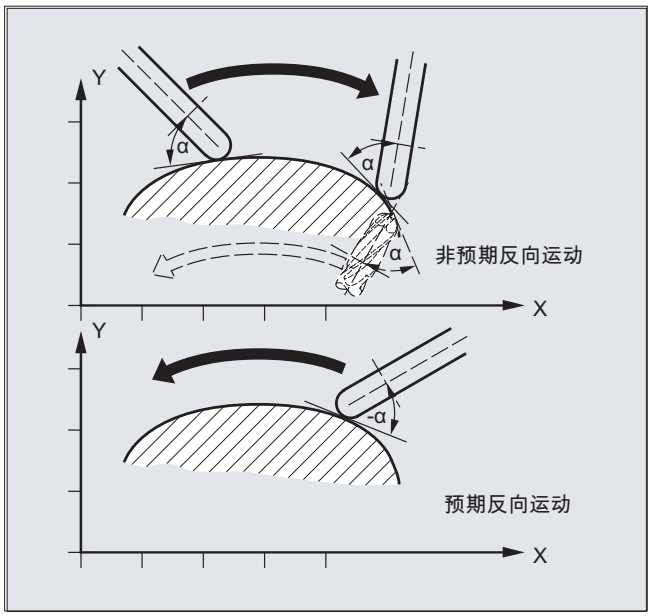
定义跟随轴和引导轴

使用 TANG 定义引导轴和跟随轴。

耦合系数表明切线角度变化和跟随轴之间的关系。 通常其数值为 1（缺省设置）。

通过工作范围限制极限角度

在往复的路径运动中，切线在折返点上反转 180 度， 相应地跟随轴的定位发生变化。 通常该特性无效： 往复运动中的偏移角度应当相同：



为此必须对跟随轴的工作区域进行限制（G25、G26）。 工作区域限制必须在换向时生效（WALIMON）。 当偏移角在工作范围限制之外时，会尝试将负偏移角重新带入容许的工作范围内。

在轮廓拐角处插入中间程序段（TLIFT）

在轮廓拐角处切线改变，跟随轴的目标位置变得不稳定。 在通常情况下，轴会尝试以其可能的最大速度平衡这种变化。 然而，此时在轮廓拐角后的一段距离内会产生误差。如果处于技术上的原因不允许有这样的误差，可使用指令 TLIFT 来命令控制系统停在拐角上，并且在一个自动生成的中间程序段中将跟随轴旋转到新的切向中。

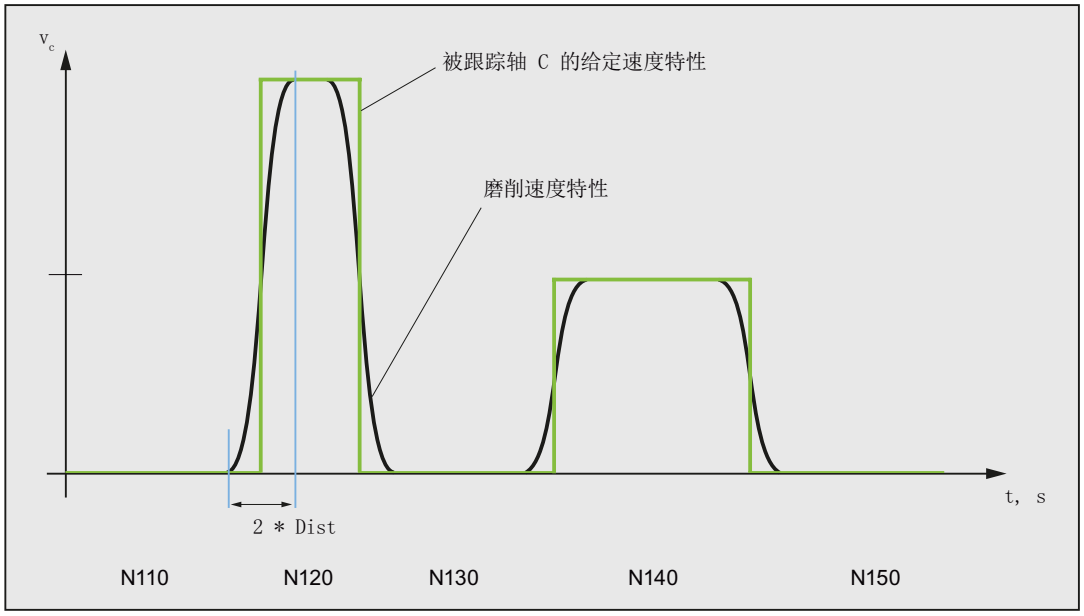
如果跟随轴曾被用作路径轴，则编程的路径轴旋转。 此处，通过设置功能 TFGREF[<轴>]=0.001 可以达到跟随轴的最大速度。

如果跟随轴还没有被用作路径轴，则该轴作为定位轴移动。 速度因此依赖于由机床数据确定的定位速度。

跟随轴以最大速度转动

优化方法

如果选择了自动优化，即<优化>="P"，并且为跟随轴设定了<平滑距离>和<角度公差>参数，那么在切向跟踪中，会平滑由引导轴路径变化而引起的跟随轴速度跃变。此时跟随轴采用预读控制（见图），尽可能减小偏差。



定义角度变化

从何种角度变化起自动插入中间程序段，可通过以下机床数据定义：

MD37400 $MA_EPS_TLIFT_TANG_STEP（拐角识别的切线角度）

对坐标转换的影响

参与跟踪的回转轴位置可以成为坐标转换的输入值。

跟随轴的明确定位

如果明确定位了随引导轴运行的跟随轴，这些位置数据会累加在已写入的偏移角上。

允许所有的位移设定（路径轴和定位轴运行）。

耦合的状态

在 NC 零件程序中，可使用系统变量 $AA_COUP_ACT[<轴>] 查询耦合的状态：

值	含义
0	无耦合有效
1,2,3	切向跟踪运行有效

8.2 进给曲线 (FNORM, FLIN, FCUB, FPO)

功能

为了较为灵活的设定进给曲线，根据
DIN 66025 的规定，进给编程增加了线性曲线和三次曲线。

三次曲线可以直接编程或作为插补样条编程。 从而可以根据待加工工件的曲度持续编程平滑的速度曲线。

这种速度曲线实现了平滑、没有急动的加速度变化，并进而完成了均匀的工件表面加工。

句法

```
F… FNORM
F… FLIN
F… FCUB
F=FPO(…,…,…)
```

含义

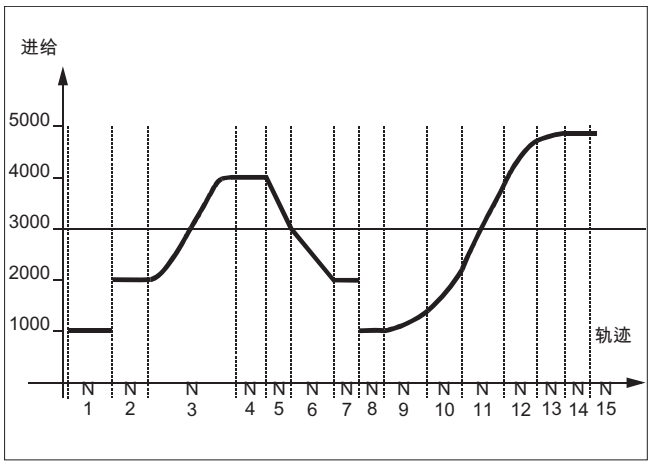
FNORM	初始设置。 进给值通过程序段中的位移来规定，并且作为模态值有效。
FLIN	轨迹速度曲线图**线性：** 从程序段开头到程序段末尾的范围内，轴从当前值开始以进给值进行线性位移，然后模态值生效。 这种属性可以和 G93 和 G94 组合使用。
FCUB	轨迹速度曲线图**三次曲线：** 以程序段方式编程的 F 值－与程序段结束处有关－通过一个样条连接。样条的开始和前一个进给值相切，结束和后一个进给值相切，并与 G93 和 G94 一起作用。 如在一个程序段中缺少 F 地址，在这里将使用最后编程的 F 值。
F=FPO…	轨迹速度曲线图**通过多项式：** F 地址表示由多项式定义的、从当前值开始到程序段结尾的进给曲线。此后终值将作为模态值有效。

在弯曲的轨迹中优化进给

进给多项式 F=FPO 和进给样条 FCUB 应当始终以恒定切削速度 CFC 执行完毕。 这样就可生成加速度恒定的设定进给曲线图。

示例： 不同的进给曲线图

在这个例子中列出了不同的进给曲线图的编程和图示。

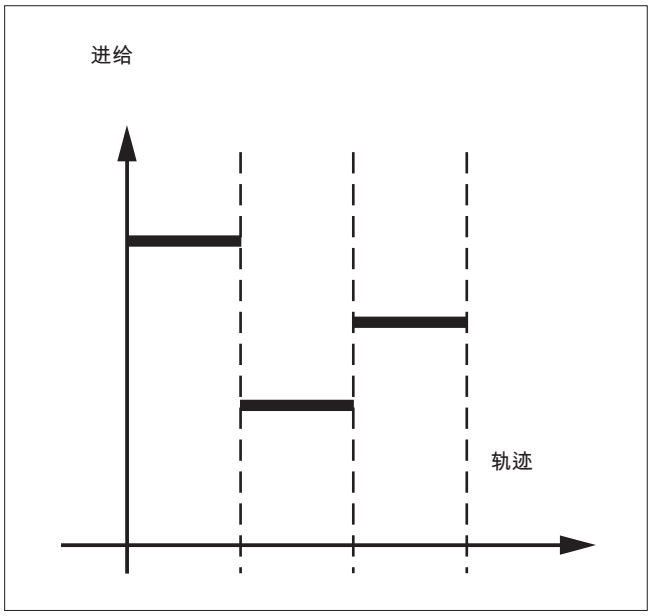


程序代码	注释
N1 F1000 FNORM G1 X8 G91 G64	; 恒定进给曲线图，增量数据
N2 F2000 X7	; 设定速度急动
N3 F=FPO(4000, 6000, -4000)	; 多项式进给曲线图，程序段结束处的进给为 4000
N4 X6	; 多项式进给 4000 作为模态值
N5 F3000 FLIN X5	; 线性进给曲线图
N6 F2000 X8	; 线性进给曲线图
N7 X5	线性进给作为模态值
N8 F1000 FNORM X5	; 恒定的进给曲线图，带有加速度急动
N9 F1400 FCUB X8	; 所有下列逐段编程的 F 值均使用样条联系在一起
N10 F2200 X6	
N11 F3900 X7	
N12 F4600 X7	
N13 F4900 X5	; 关闭样条曲线图
N14 FNORM X5	
N15 X20	

FNORM

进给地址 F 把轨迹进给表示为符合 DIN66025 的恒定值。

更多的信息，请参见编程手册“基础部分”。

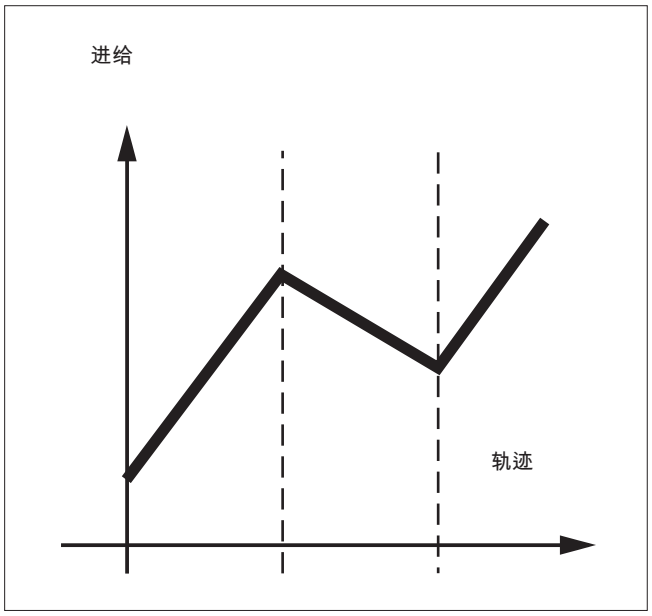


FLIN

进给进程通过实际的进给值到被编程的 F 值，线性运行到程序段末尾。

举例：

```
N30 F1400 FLIN X50
```

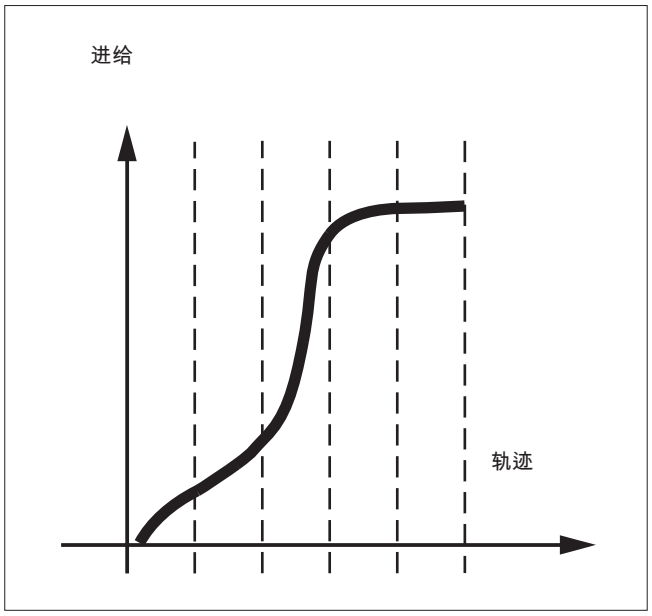


FCUB

从实际的进给值到编程的 F 值到程序段结尾，进给以空间的行程运行。控制系统通过样条连接所有有效 FCUB 程序方式编程的进给值。进给值作为计算样条插补的支点。

举例：

```
N50 F1400 FCUB X50
N60 F2000 X47
N70 F3800 X52
```



F=FPO(…,…,…)

进给进程通过一个多项式直接编程。多项式系数 的参数类似于多项式插补。

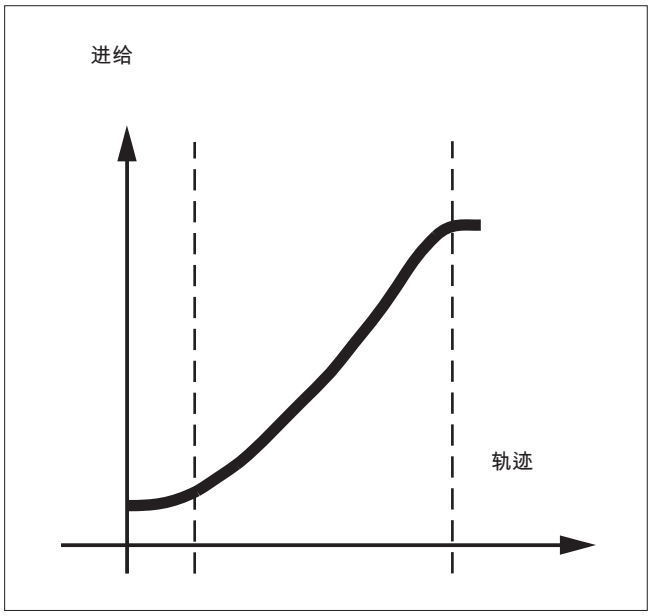
举例：

```
F=FPO(endfeed, quadf, cubf)
```

endfeed, quadf 和 cubf 为之前定义的变量。

endfeed:	程序段结束时的进给
quadf:	二次方的多项式系数
cubf:	立方的多项式系数

当 FCUB 激活时，程序段开始和结束处的样条与通过 FPO 设定的曲线相切。



边界条件

与编程的进给进程无关，轨迹运行方式的编程的功能有效。

可编程的进给曲线基本上绝对独立于 G90 或者 G91.

进给进程 FLIN 和 FCUB 与 G93 和 G94

G93 和 G94.

FLIN 和 FCUB 不在使用

G95, G96/G961 和 G97/G971 时起作用。

激活的压缩器 COMPON

当压缩器 COMPON 激活时，将多个程序段合并成一个样条线段时适用：

FNORM:

对于样条段，最后的程序段 F 字有效。

FLIN;

对于样条段，最后的程序段 F 字有效。
编程的 F 值在程序段的结束处有效，然后线性返回。

FCUB:

生成的进给样条最多按照机床数据 C $MC_COMPRESS_VELO_TOL 中所定义的值偏离以编程的终点。

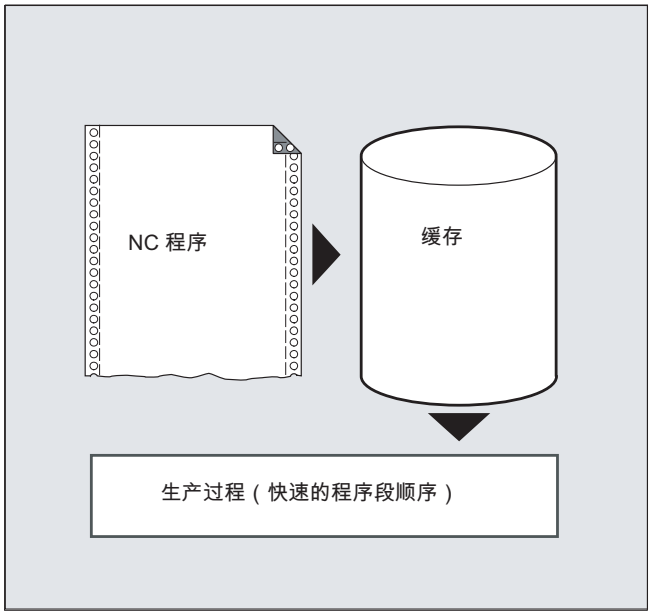
F=FPO(...,...,...)

程序段不压缩。

8.3 带有缓存的程序运行过程 (STOPFIFO, STARTFIFO, FIFOCTRL, STOPRE)

功能

根据扩展级，控制系统具有一定数量的缓存存储器，它们会在加工前存储预处理的程序段，并在加工过程中作为快速程序段输出。 借此可以以较高速度运行较短的行程。 只要控制系统的剩余时间允许，原则上缓存都会被载满。



标记缓存段

零件程序中需要在缓存中存储的程序段会标有 STOPFIFO（开始）以及 STARTFIFO（结束）。 当缓存已满或者执行指令 STARTFIFO 之后，才会开始执行经过预处理并处于缓存中的程序段。

缓存的自动控制

缓存的自动控制通过指令 FIFOCTRL 调用。 FIFOCTRL 的作用如同 STOPFIFO。 在每次编程时都会等待缓存被载满，然后才开始执行程序。 只是缓存在空运行时的属性有所不同： 编程 FIFOCTRL 后，当缓存容量达到 2/3 时，轨迹速度会不断降低，从而避免出现完全的空运行和急剧的停止过程。

预处理停止

如果在程序段中编程了 STOPRE 指令，程序段预处理和缓存过程将被终止。 只有当全部执行了所有预处理并缓存的程序段后，才开始执行后面的程序段。 之前的程序段会以准停方式停止（如 G9）。

句法

表格 8- 1　标记缓存段：

```
STOPFIFO
...
STARTFIFO
```

表格 8- 2　缓存的自动控制：

```
...
FIFOCTRL
...
```

表格 8- 3　预处理停止：

```
...
STOPRE
...
```

说明

指令 STOPFIFO、STARTFIFO、FIFOCTRL 和 STOPRE 必须在单独的程序段中编程。

含义

STOPFIFO:	STOPFIFO 标出了需要存储在缓存中的程序段的开始。 STOPFIFO 指令会停止处理并载满缓存，直至： • 识别到 STARTFIFO 或 STOPRE 或者 • 缓存已满 或者 • 达到程序末尾。
STARTFIFO:	STARTFIFO 会启动程序段的快速处理，同时会加载缓存
FIFOCTRL:	启用缓存的自动控制
STOPRE:	停止预处理

说明

当缓存程序段中包含的指令需要强制进行非缓存运行，如回参考点、测量功能等，系统就不会执行或者中断缓存加载。

说明

当访问机床的状态数据时（$SA...），控制系统生成会内部预处理停止。

⚠小心

当刀具补偿已启用且进行样条插补时，不应编程 STOPRE，否则会中断相关的程序段顺序。

示例： 停止预处理

程序代码	注释
...	
N30 MEAW=1 G1 F1000 X100 Y100 Z50	; 带有第一个测量输入端的测量探头和直线插补的测量程序段。
N40 STOPRE	; 预处理停止。
...	

8.4 可以有条件中断的程序段 (DELAYFSTON, DELAYFSTOF)

功能

可以有条件中断的零件程序段被称为（Stopp-Delay）停机延时段。在某些程序段内不应当**停住** 且**进给** 也不应当改变。基本上较短的程序段，例如用来制作螺纹的程序段，几乎都有防止停止事件的保护。 在程序段处理完后，一个可能的停止才能产生作用。

句法

```
DELAYFSTON
DELAYFSTOF
```

命令单独在一个零件程序行中。

两个指令都只允许在零件程序中，而不可以在同步动作中。

含义

DELAYFSTON	定义一个域的开始，在这个范围中“软”停止被延后，直到到达停止延迟区的末尾。
DELAYFSTOF	定义一个停止延迟区的结尾

说明

对于机床数据 MD11550 $MN_STOP_MODE_MASK Bit 0 = 0 (默认)，就会隐性定义停止延时段，当 G331/G332 已激活且轨迹运动或者 G4 已编程时。

示例： 停止事件

在停止延时段中，会忽略**进给和进给锁止** 的改变。 它们在停止延迟区之后才产生作用。

停止事件将被区分为：

“软”停止事件	反应： 延迟
“硬”停止事件	反应： 立即

选择几个停止事件，至少暂时停止：

事件名称	反应	中断参数
RESET	立即	NST: DB21,… DBX7.7 和 DB11, … DBX20.7
PROG_END	报警 16954	NC 程序： M30
INTERRUPT	延迟	NST: FC-9 和 ASUP DB10, ... DBB1
SINGLEBLOCKSTOP	延迟	停止延时段中的单程序段运行被启用： NC 在停止延时段之外的第 1 个程序段的末尾处停止。 单程序段已在停止延时段之前被选中： NST: "NC 在程序段结束处停止" DB21, ... DBX7.2
STOPPROG	延迟	NST: DB21,… DBX7.3 und DB11, … DBX20.5
PROG_STOP	报警 16954	NC 程序： M0 和 M1
WAITM	报警 16954	NC 程序： WAITM
WAITE	报警 16954	NC 程序： WAITE
STOP_ALARM	直接	报警： 报警设计 STOPBYALARM
RETREAT_MOVE_THREAD	报警 16954	NC 程序： 报警 16954，当 LFON (不可以在 G33 中停止和快速升高)
WAITMC	报警 16954	NC 程序： WAITMC
NEWCONF_PREP_STOP	报警 16954	NC 程序： NEWCONF
SYSTEM_SHUTDOWN	立即	在 840Di 时系统关闭
ESR	延迟	扩展的停止和退回
EXT_ZERO_POINT	延迟	外部零点偏移
STOPRUN	报警 16955	BTSS: PI "_N_FINDST" STOPRUN

反应的解释

立即("硬"停止事件)	立即在停止延迟区中停止
延迟 ("软"停止事件)	停止（也包括短时间的）只在停止延迟区后产生。
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	在停止延迟区中进行一个不被允许的动作，程序继续。
警报 16957	通过 DELAYFSTON 和 DELAYFSTOF 括起来的程序段（停机延时段）未能激活。因此每次停机在段内立即生效而不延迟。

关于停止事件其它反应的总结，请参见

文献：

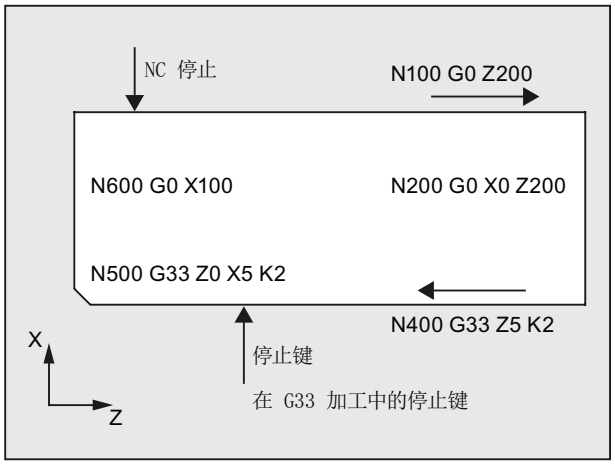
功能手册 基本功能；BAG、通道、程序运行、（K1），章节“控制和影响停机事件”

示例： 两个程序层中停止延时段的嵌套

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()	; 程序级面 2 重复的开始，无效。
...	
N20020 DELAYFSTOF()	; 无效，在另一个平面中结束。
N20030 RET	
N10050 DELAYFSTOF()	; 相同层中停止延时段末尾。
...	
N10060 R2 = R2 + 2	
N10070 G4 F1	; 结束停止延迟区。 停止从现在起立即有效。

示例： 程序摘录

在一个循环中重复下列程序块：



图中可见用户在停止延时段中按下“停止”，NC 开始执行停止延时段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。

程序代码	注释
...	
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	

有关类型 SERUPRO 的程序段查找和与 G331/G332 一起不用补偿夹具丝锥进给时的进给率的详细信息，参见

文献：

功能手册 基本功能；BAG、通道、程序运行 (K1)

功能手册 基本功能；进给 (V1)

停止延时段的有点

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一个跟踪搜索的搜索目标。

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。

- 指令轴和
- 使用 POSA 运动的定位轴

零件程序指令 G4 在停止延时段中是允许的，与此相反，其它执行临时停止的零件程序指令则不允许 (例如 WAITM) 。

G4 使停止延时段有效（就像一个轨迹运动）或者使其保持有效。

举例：进给干预

如果在停止延时段前将修调率降低到 6%，则修调率就会在停止延时段中有效。

如果在停止延时段中将修调率从 100% 降低到 6%，就会以 100% 使停止延时段执行结束，然后在以 6% 继续执行。

进给锁止在停止延时段中不起作用，只有在离开停止延时段后才会停住。

叠加/嵌套：

如果两个停止延时段相互交迭，一个来自于语言指令，另一个来自于机床数据 MD 11550:STOP_MODE_MASK, 就会形成最有可能的停止延时段。

下列各项用来调节语言指令 DELAYFSTON 和 DELAYFSTOF 与嵌套和子程序结束之间的相互作用。

1. 当子程序结束时， DELAYFSTON 已经在其中被调用， DELAYFSTOF 就被隐式激活。
2. DELAYFSTON 停止延时段保持无效。
3. 如子程序 1 在停止延迟区中调用子程序 2，那子程序 2 就是完整的停止延迟区。特别是 DELAYFSTOF 在子程序 2 中不起作用。

说明

REPOSA 是一个子程序结束且 DELAYFSTON 在任何情况下均会被取消。

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时段。

如果停止键在停止延时段之前被按下且 NCK 必须进入停止延时段以进行制动，则 NCK 就会在停止延时段停住，并且停止延时段保持被取消！

如果倍率为 0%时出现一个停机延时段，则停机延时段不 被接受!

这适用于所有“软”停止事件。

使用 STOPALL 可以在停止延时段中制动。使用一个 STOPALL 可立即激活其它所有目前为止被延迟的停止事件。

系统变量

可使用 $P_DELAYFST 在零件程序中识别一个停止延时段。如果系统变量中的位 0 值为 1 的话，零件程序的加工在这个时候处于一个停止延迟区内。

使用 $AC_DELAYFST 可在同步动作中识别一个停止延时段。如果系统变量中的位 0 值为 1 的话，零件程序的加工在这个时候处于一个停止延迟区内。

兼容性

预设置机床数据 MD 11550:STOP_MODE_MASK Bit 0 = 0 可在 G 代码组 G331/G332 已编程且当一个轨迹运动或者 G4 已编程时，使隐式停止延时段有效。

Bit 0 = 1 可在 G 代码组 G331/G332 已编程且当一个轨迹运动或者 G4 已编程时（软件版本 6 以下的特性），实现停止。定义一个停止延时段时，必须使用指令 DELAYFSTON/DELAYFSTOF 。

8.5 阻止 SERUPRO 的程序位置 (IPTRLOCK, IPTRUNLOCK)

功能

对于某些机械复杂的机床配置，要求阻止程序段查找 SERUPRO。

使用一个可编程的中断指示，可以使“查找中断点”时在不可查找的位置之前停止。

也可以在零件程序范围中定义不可查找的区域，在其中 NCK 不可以再次进入。 使用程序中断，NCK 记下最后加工的程序段，通过操作界面 HMI 可以查找到该程序段。

句法

```
IPTRLOCK
IPTRUNLOCK
```

这些示例单独处于某个零件程序行中并且可实现一个可编程的中断向量。

含义

IPTRLOCK	开始不可查找的程序段
IPTRUNLOCK	结束不可查找的程序段

两个指令仅允许在零件程序中，但**不可** 在同步动作中。

示例

在两个带有隐式 IPTRUNLOCK 的程序层中嵌套不可查找的程序段。 子程序 1 中的隐式 IPTRUNLOCK 结束不可查找的程序段。

程序代码	注释
N10010 IPTRLOCK()	
N10020 R1 = R1 + 1	
N10030 G4 F1	; 停止程序段，开始不可查找的程序段。
...	
N10040 子程序 2	
...	; 子程序 2 的说明。
N20010 IPTRLOCK ()	; 无效，重复的开始。
...	
N20020 IPTRUNLOCK ()	; 无效，在另一个平面中结束。
N20030 RET	

程序代码	注释
...	
N10060 R2 = R2 + 2	
N10070 RET	; 结束不可查找的程序段。
N100 G4 F2	; 继续主程序。

中断到 100，重新提供了中断指示。

采集和查找不可查找的区域

不可查找的程序段使用语言指令 IPTRLOCK 和 IPTRUNLOCK 进行标识。

指令 IPTRLOCK 将中断向量冻结成一个在主过程中可以执行的单程序段 (SBL1) 。该程序段在以下所述中被作为停止程序段。如果在 IPTRLOCK 之后出现一个程序中断，就可以在操作界面 HMI 上查找该停止程序段。

再次停止在当前的程序段

使用后续程序段的 IPTRUNLOCK 将中断向量设置给中断点的当前程序段。

在找到一个查找目标后，可以用该停止程序段重复一个新的查询目标。

被用户编辑过的中断向量必须通过 HMI 重新删除。

嵌套时调节

下列各项用来调节语言指令 IPTRLOCK 和 IPTRUNLOCK 与嵌套和子程序结束之间的相互作用。

1. 当子程序结束时， IPTRLOCK 已经在其中被调用， IPTRUNLOCK 就被隐式激活。
2. IPTRLOCK 在一个不可查找的程序段中保持无效。
3. 如果子程序 1 在一个不可查找的区域调用子程序 2，则子程序 2 保持不可查找。特别是 IPTRUNLOCK 在子程序 2 中不起作用。

其它信息，参见
/FB1/ 功能手册基本功能；BAG、通道、程序运行（K1）。

系统变量

可使用 $P_IPTRLOCK 在零件程序中识别一个不可查找的程序段。

自动的中断指示

自动的中断指示的功能自动将一个先前确定的耦合方式确定为无法搜索。借助机床数据

- 电子控制式变速器，当 EGON
- 当 LEADON 轴向主值偶合

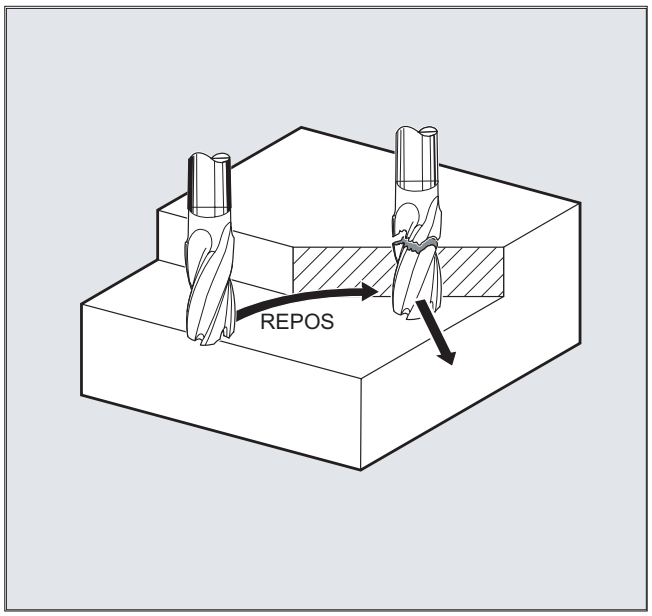
激活自动中断指针。如果已编程的中断向量和可通过机床数据激活的自动中断向量相互交迭，就会形成最有可能的不可查找程序段。

8.6 返回轮廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN）

功能

如果您在加工过程中必须中断正在运行的程序，使刀具移开，比如由于刀具断裂或者需要测量尺寸，则您可以通过程序控制再次返回轮廓到一个可选择的点。

REPOS 指令的作用如同一个子程序返回（例如通过 M17）。 中断程序中的下列程序段将不再执行。



关于程序过程的中断，另见该“编程手册”中“中断程序”一章中的“灵活的 NC 编程”段落。

句法

```
REPOSA RMI DISPR=…
REPOSA RMB
REPOSA RME
REPOSA RMN
REPOSL RMI DISPR=…
REPOSL RMB
REPOSL RME
REPOSL RMN
REPOSQ RMI DISPR=… DISR=…
REPOSQ RMB DISR=…
REPOSQ RME DISR=…
REPOSQA DISR=…
REPOSH RMI DISPR=… DISR=…
```

```
REPOSH RMB DISR=…
REPOSH RME DISR=…
REPOSHA DISR=…
```

含义

返回行程

`REPOSA`	所有轴线性返回
`REPOSL`	以线性返回
`REPOSQ DISR=…`	以四分之一圆弧返回，半径 DISR
`REPOSQA DISR=…`	所有轴以四分之一圆弧返回，半径 DISR
`REPOSH DISR=…`	以半个圆弧返回，直径 DISR
`REPOSHA DISR=…`	所有轴以半个圆弧返回，半径 DISR

再次返回点

`RMI`	返回中断点
`RMI DISPR=…`	间距为 DISPR 的进入点，单位为 mm/inch，在中断点前
`RMB`	返回程序段起始点
`RME`	返回程序段终点
`RME DISPR=…`	返回程序段终点，间距 DISPR，在终点之前
`RMN`	返回到下一个轨迹点
`A0 B0 C0`	应该返回的轴

示例： 返回，在一条直线上返回，REPOSA，REPOSL

刀具以一条直线直接回到再次返回点。

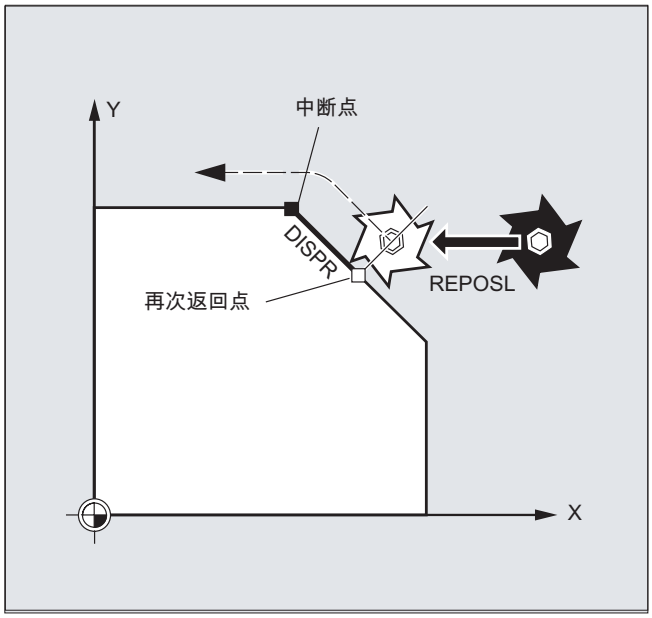
使用 REPOSA，所有轴自动处理。 使用 REPOSL 时，您可以指定待运行的轴。

示例：

```
REPOSL RMI DISPR=6 F400
```

或者

```
REPOSA RMI DISPR=6 F400
```

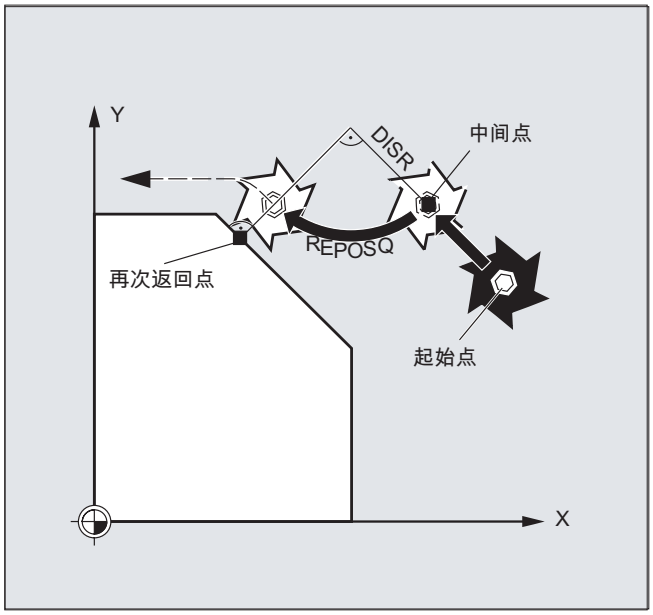


示例： 返回，在四分之一圆中返回，REPOSQ，REPOSQA

刀具向半径为 DISR=...的四分之一圆上的重新起动点运动。 控制系统自动计算起始点和再次返回点之间所必需的中间点。

示例：

```
REPOSQ RMI DISR=10 F400
```

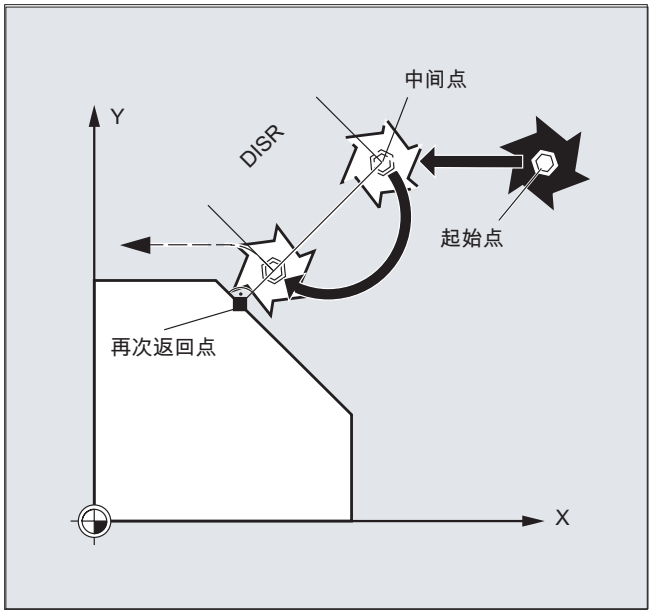


示例： 刀具在半圆中返回，REPOSH，REPOSHA

刀具向半径为 DISR=...的半圆上的重新起动点运动。 控制系统自动计算起始点和再次返回点之间所必需的中间点。

示例：

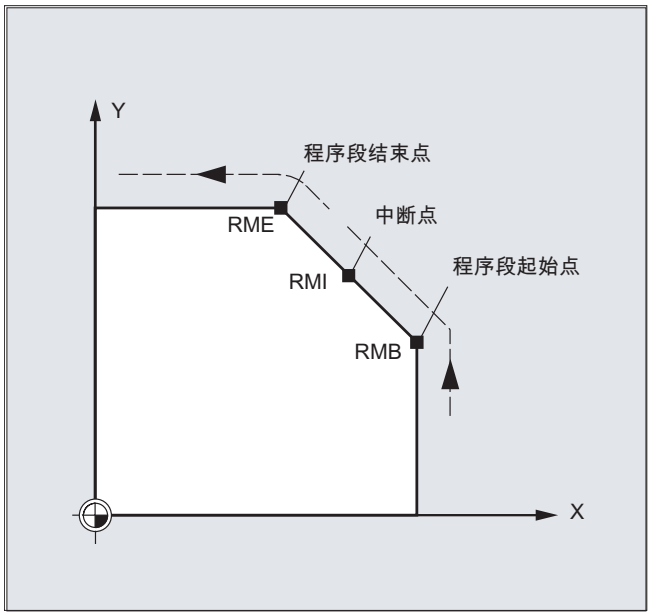
```
REPOSH RMI DISR=20 F400
```



确定重新起动点 (不适用于带有 RMN 的 SERUPRO 起动)

考虑到程序中断的 NC 程序段，您可以在三个再次返回点之间选择：

- RMI，中断点
- RMB，程序段起始点或者最后的终点
- RME，程序段终点



使用 RMI DISPR=... 或者使用 RME DISPR=... 可以确定位于中断点前或者程序段结束点前的重新起动点。

使用 DISPR=... 可围绕中断点或者结束点**前** 重新起动点的轮廓轨迹。 该点可以（也适用于较大数值）处于程序段开始点。

如果没有编程 DISPR=...，则 DISPR=0 且中断点（当 RMI) 以及程序段结束点 (当 RME) 因此有效。

DISPR 的前置符

分析 DISPR 的前置符。在正号时，特性同前。

当前置符为负时，就在中断点之后或者当 RMB 时在开始点之后重新起动。

中断点和重新起动点之间的间距从 DISPR 的值中得出。对于总量较大的值，该点最大可以位于程序段终点处。

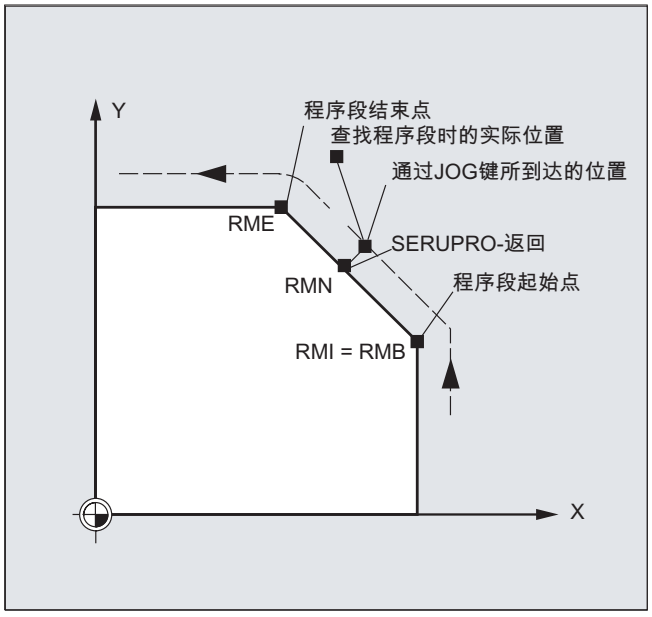
应用示例：

通过一个传感器可以识别出到夹板的接近。 将触发一个 ASUP，以此来围绕夹板运动。

然后使用负的 DISPR 重新向夹板后面的某个点定位并且继续执行程序。

SERUPRO-使用 RMN 起动

如果在任意一个位置上加工时强迫中断，然后就可在使用 RMN 的情况下，用 SERUPRO 起动从中断位置以最短行程起动，以便接着只执行完剩余行程。对此用户启动一个 SERUPRO 过程到中断程序段，并用 JOG 键定位到目标程序段损坏位置之前。



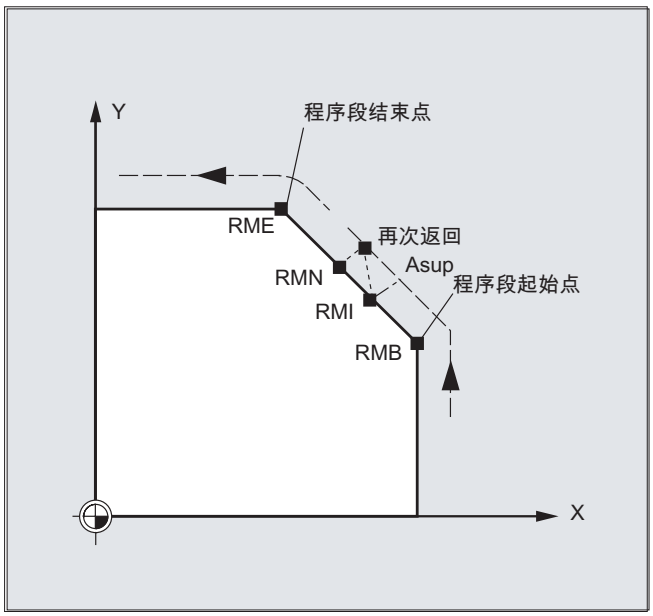
说明

SERUPRO

对于 SERUPRO 而言， RMI 和 RMB 是一样的。 RMN 不仅仅被限制到 SERUPRO ，而且普遍有效。

从下一个轨迹点起动 RMN

到了 REPOSA 的解释时刻时，在中断之后不会使用 RMN 再次完全开始重新起动程序段，而是仅执行完剩余行程。返回运行到中断程序段的最近轨迹点。



有效 REPOS-模式的状态

已中断程序段的有效 REPOS 模式可以通过带有变量 $AC_REPOS_PATH_MODE 的同步动作读取：

0: 不定义返回运行

1 RMB:返回到开始

2 RMI:返回到中断点

3 RME:返回到程序结束点

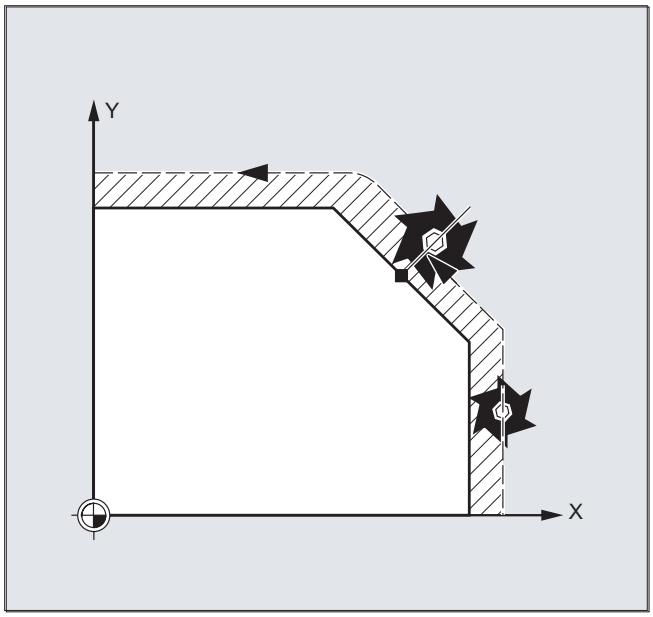
4 RMN: 向已中断程序段的下一个轨迹点运动。

使用新刀具起动

如果程序运行由于刀具损坏而停止：

通过编程新的 D 号，该程序自再次返回点起以修改后的刀具补偿值继续进行。

如果刀具补偿值修改，则中断点可能不再返回。在这种情况下，返回到新轮廓上与该中断点最近的点（有时更改 DISPR）。



返回轮廓

刀具的这种再次返回轮廓的运动可以编程。用值零说明待运行轴的地址。

使用指令 REPOSA, REPOSQA 和 REPOSHA 自动重新定位所有轴。不需要进行轴说明。

当编程 REPOSL, REPOSQ 和 REPOSH 时，所有几何轴自动起动，即使没有在指令中指定也会起动。 所有其它轴必须在指令中指定。

REPOSH 和 REPOSQ 适用于圆弧运动：

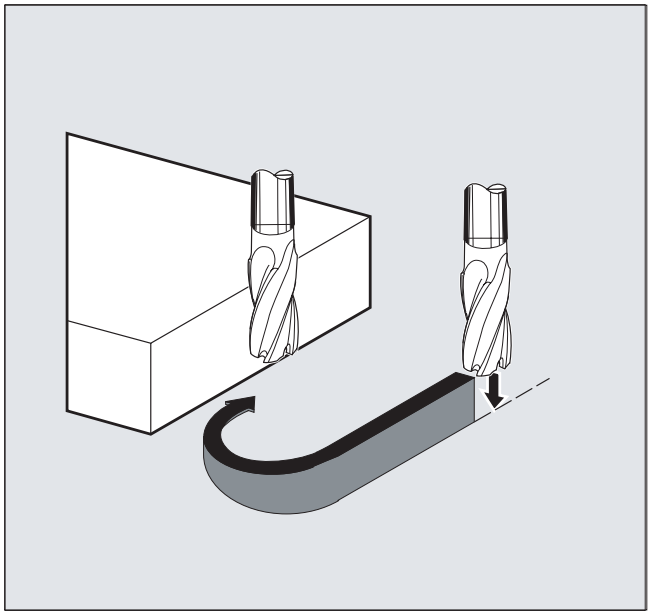
在指定的工作平面 G17 ～ G19 中作圆运动。

如果在起动程序段中指定第三个几何轴（进给方向），在刀具位置和已编程的位置在进给方向中不一致的情况下，就会在一个螺旋线上向重新起动点运动。

在下列情况下自动转换成

现行起动 REPOSL ：

- 没有为 DISR 指定值。
- 没有定义的返回运行方向（在一个程序段中程序中断，没有运行信息）。
- 当返回运行方向垂直于当前工作平面时。



8.7 对运动控制的影响

8.7.1 百分比式急冲修正 (JERKLIM)

功能

使用 NC 指令 JERKLIM，可在重要程序段落中降低或升高原先由机床数据设置的、路径运行允许的最大轴急动。

前提条件

加速模式 SOFT 必须已激活。

有效性

此功能在以下情况下生效：

- 在 AUTO 运行方式中。
- 仅对于路径轴生效。

句法

```
JERKLIM[<轴>]=<值>
```

含义

JERKLIM:	急动补偿指令
<轴>：	需要调整急动限值的机床轴。
<值>：	百分比的补偿值，以设置的路径运行中最大轴急动为基准（MD32431 $MA_MAX_AX_JERK）。 取值范围： 1 ... 200 取值为 100 时对急动没有影响。

说明

零件程序结束和通道复位时 JERKLIM 的特性通过机床数据 MD32320 $MA_DYN_LIMIT_RESET_MASK 的位 0 配置。

- 位 0 = 0：

  编程的 JERKLIM 的值在通道复位/M30 时复位为 100 %。

- 位 0 = 1：

  编程的 JERKLIM 的值在通道复位/M30 后保持不变。

示例

程序代码	注释
...	
N60 JERKLIM[X]=75	; 轴溜板在 X 方向以最大 75% 的轴急动进行加速/减速。
...	

8.7.2 百分比式速度修正 (VELOLIM)

功能

为降低机床负荷或改善加工质量，可在重要程序段落中编写 NC 指令 VELOLIM，降低原先由机床数据设置的最大速度：进给轴/主轴在进给轴模式中的最大速度；主轴在主轴模式中受齿轮级影响的最大转速，其中包括：转速控制模式 M3、M4、M5 和定位模式 SPOS、SPOSA、M19。

有效性

此功能在以下情况下生效：

- 在 AUTO 运行方式中。
- 对路径轴和定位轴生效。
- 对主轴模式/进给轴模式中的主轴生效

句法

VELOLIM[<进给轴/主轴>]=<值>

含义

VELOLIM:	速度补偿指令
<进给轴/主轴>：	需要调整速度或转速限值的机床轴或主轴。 **VELOLIM 用于主轴** 通过机床数据（MD30455 $MA_MISC_FUNCTION_MASK，位 6）可为零件程序中的编程设置，VELOLIM 针对进给轴/主轴作用相同（位 6 = 1），还是作用不同而需要分别编程（位 6 = 0）。 如果设置了分别编程，则在编程时通过标识符进行选择轴： • 主轴标识符 S<n> 用于主轴运模式 • 进给轴标识符，例如“C”，用于进给轴模式
<值>：	百分比补偿值 补偿值的基准： • 进给轴模式中的进给轴/主轴（当 MD30455 位 6 = 0 时）： 以配置的最大轴速度为基准（MD32000 $MA_MAX_AX_VELO）。 • 主轴模式或进给轴模式中的主轴（当 MD30455 位 6 = 1 时）： 以生效的齿轮级的最大转速为基准（MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT[<n>]） 取值范围： 1 ... 100 取值为 100 时对速度或转速没有影响。

说明

零件程序结束和通道复位时的特性

零件程序结束和通道复位时 VELOLIM 的特性通过机床数据 MD32320 $MA_DYN_LIMIT_RESET_MASK 的位 0 配置。

- 位 0 = 0：
  编程的 VELOLIM 的值在通道复位/M30 时复位为 100 %。
- 位 0 = 1：
  编程的 VELOLIM 的值在通道复位/M30 后保持不变。

说明

VELOLIM 用于同步动作中的主轴

在同步动作中编程 VELOLIM 时，不会区分主轴模式和进给轴模式。 与编程时使用的标识符无关，会对主轴模式中的转速和进给轴模式中的速度进行同等程度的限制。

诊断

主轴模式中 VELOLIM 的诊断

在主轴模式中，可通过读取系统变量 $AC_SMAXVELO 和 $AC_SMAXVELO_INFO 确定由 VELOLIM 生成的转速限制（小于 100 %）。

在存在限制时，$AC_SMAXVELO 会输出 VELOLIM 生成的转速限制。 此时变量 $AC_SMAXVELO_INFO 会反馈值“16”以表示限制原因是 VELOLIM。

示例

示例 1： 机床轴的速度限制

程序代码	注释
...	
N70 VELOLIM[X] = 80	; 轴溜板在 X 方向应以 80% 轴允许的最大速度运行。
...	

示例 2： 主轴的转速限制

程序代码	注释
N05 VELOLIM[S1]=90	; 将主轴 1 的最大转速限制为 1000 rpm 的 90%。
...	
N50 VELOLIM[C]=45	; 将转速限制为 1000 rpm 的 45%，C 为 S1 的进给轴标识符。
...	

主轴 1 的配置数据（AX5）：

MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT[1,AX5]=1000	; 齿轮级 1 的最大转速 = 1000 rpm
MD30455 $MA_MISC_FUNCTION_MASK[AX5] = 64	; 位 6 = 1： VELOLIM 的编程与编程的标识符无关，而是对主轴模式和进给轴模式共同生效。

8.7.3 JERKLIM 和 VELOLIM 的程序举例

下面的程序说明了百分比情况下突变和速度极限值的应用示例：

程序代码	注释
N1000 G0 X0 Y0 F10000 SOFT G64	
N1100 G1 X20 RNDM=5 ACC[X]=20	
ACC[Y]=30	
N1200 G1 Y20 VELOLIM[X]=5	; 轴溜板在 X 方向应以最大 5% 轴的允许速度进行运行。
JERKLIM[Y]=200	; 轴溜板在 X 方向应以最大 200% 轴的允许突变值进行加速/延迟。
N1300 G1 X0 JERKLIM[X]=2	; 轴溜板在 X 方向应以最大 2% 轴的允许突变值进行加速/延迟。
N1400 G1 Y0	
M30	

8.8 可编程的轮廓公差/定向公差(CTOL, OTOL, ATOL)

功能

通过指令 CTOL、OTOL 和 ATOL 可以在 NC 程序中修改以下参数：通过机床数据和设定数据确定的、用于压缩器功能(COMPON, COMPCURV, COMPCAD)、精磨方式 G642、G643、G645、OST 和定向平滑 ORISON 的加工公差。

这些编程的值会持续生效，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，在程序结束、通道复位、工作方式复位、NCK 复位（热启动）和上电（冷启动）时也会删除这些值。 删除后机床数据和设定数据中的值恢复生效。

句法

```
CTOL=<值>
OTOL=<值>
ATOL[<轴>]=<值>
```

含义

CTOL　用于编程**轮廓公差**的指令

CTOL 适用于：

- 所有的压缩器功能
- 所有的精磨方式，除了 G641 和 G644

<值>：　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）

OTOL　用于编程**定向公差**的指令

OTOL 适用于：

- 所有的压缩器功能
- 定向平滑 ORISON
- 所有的精磨方式，除了 G641, G644, OSD

<值>：　定向公差值是角度数据。

类型：　REAL

单位：　度

ATOL　用于编程**轴专用公差**的指令

ATOL 适用于：

- 所有的压缩器功能
- 定向平滑 ORISON
- 所有的精磨方式，除了 G641, G644, OSD

<轴>:　编程的轴公差针对的轴的名称

<值>:　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

说明

CTOL 和 OTOL 优先于 ATOL。

边界条件

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...	
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...	

程序代码	注释
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...	

其它信息

读取公差值

考虑到后续应用和诊断目的，不管在何种状态下，当前生效的公差值始终可以通过系统变量读取，即压缩器功能(COMPON, COMPCURV, COMPCAD)、精磨方式 G642, G643, G645, OST 和定向平滑 ORISON 的公差。

- 在同步中或带预处理停止的零件程序中，通过系统变量：

$AC_CTOL	轮廓公差，在处理当前主运行程序段时生效 如果没有轮廓公差生效，$AC_CTOL 会返回一个由各个几何轴公差的平方相加后计算得出的平方根值。
$AC_OTOL	定向公差，在处理当前主运行程序段时生效 如果没有定向公差生效，在定向转换生效期间，$AC_OTOL 会返回一个由各个定向轴公差的平方相加后计算得出的平方根值，否则为“-1”。
$AA_ATOL[<轴>]	轴公差，在处理当前主运行程序段时生效 如果轮廓公差生效，$AA_ATOL[<几何轴>]会返回一个由该轮廓公差除以几何轴数量的平方根得出值。 如果定向公差和定向转换生效，$AA_ATOL[<几何轴>]会返回一个由该定向公差除以定向轴数量的平方根得出的值。

说明

如果没有编程任何公差值，$A 变量将无法区分单个功能的不同公差，因为它只能返回一个值。

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，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°，变量 $AC_OTOL 会返回值“1”。 如果关闭了定向平滑功能，则只返回值“0.1”。

- 在不带预处理停止的零件程序中，通过系统变量：

$P_CTOL	编程的轮廓公差
$P_OTOL	编程的定向公差
$PA_ATOL	编程的轴公差

说明

如果没有编程任何公差值，$P 变量将返回值“-1”。

8.9 G0 运动的公差（STOLF）

G0 公差系数

与加工工件不同，在 G0 运动中（快速移动，进给运动）可允许较大的公差。 优点是缩短了 G0 的返回时间。

通过 G0 公差系数的机床数据（MD20560 $MC_G0_TOLERANCE_FACTOR）可设置该 G0 公差。

G0 公差系数仅在以下情况下才生效：

- 下列功能中有一个生效：
  - 压缩器功能：COMPON、COMPCURV 和 COMPCAD
  - 平滑功能：G642 和 G645
  - 方向圆滑：OST
  - 方向平滑：ORISON
  - 路径相关的方向平滑：ORIPATH
- 存在连续多个（≥ 2）G0 程序段。

  在只有一个 G0 程序段时 G0 公差系数不会生效，因为在从非 G0 运动**过渡至** G0 运动（并反向）时，通常**"较小的公差"**（工件加工公差）会生效！

功能

通过在零件程序中编程 STOLF 可临时覆盖设置的 G0 公差系数（MD20560）。 此时不会修改 MD20560 中的值。 在复位或零件程序结束后，配置的公差系数会重新生效。

句法

```
STOLF=<公差系数>
```

含义

STOLF:	用于编程 G0 公差系数的指令
<公差系数>:	G0 公差系数 系数可大于 1 也可小于 1。 但是通常可为 G0 运动设置较大的公差。 在 STOLF=1.0（等于配置的缺省值）时，G0 运动时生效的公差与非 G0 运动时相同。

系统变量

零件程序中或当前插补程序中生效的 G0 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通过 MD20560 $MC_G0_TOLERANCE_FACTOR 设置的值。

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。

示例

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

程序代码	注释
	$MC_G0_TOLERANCE_FACTOR*$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 mm 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。

轴耦合

9

9.1 联动 (TRAILON, TRAILOF)

功能

当一个已定义的引导轴运动时，指定给该轴的耦合轴（＝跟随轴）会在参照某个耦合系数的情况下，开始运行引导轴所引导的位移。

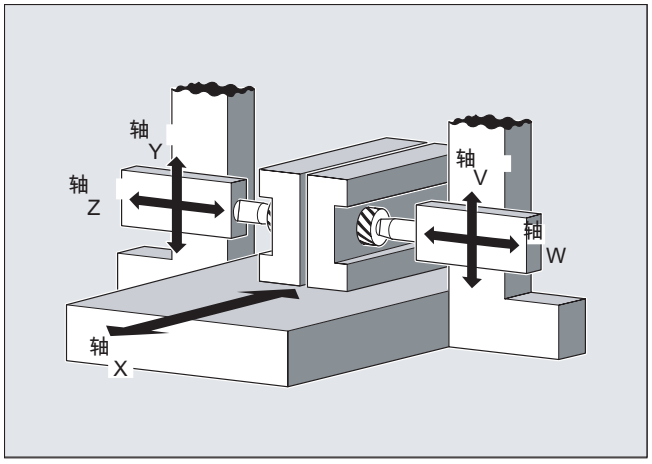
引导轴和跟随轴共同组成耦合组合。

应用范围

- 通过一个模拟轴进行轴运行。 引导轴是一个模拟轴，而耦合轴是一个真正的轴。 从而可以使得真实轴可以参照耦合系数运行。
- 用 2 个耦合组合进行两面加工：

  第 1 引导轴 Y，耦合轴 V

  第 2 引导轴 Z，耦合轴 W



句法

```
TRAILON(<跟随轴>,<引导轴>,<耦合系数>)
TRAILOF(<跟随轴>,<引导轴>,<引导轴 2>)
TRAILOF(<跟随轴>)
```

含义

TRAILON	用于启用和定义耦合轴组合的指令
	有效性： 模态
<跟随轴>	参数 1 耦合轴的名称
	提示： 一个耦合轴也可以是其余耦合轴的引导轴。 以这种方式可以建立不同的耦合组合。
<引导轴>	参数 2 引导轴的名称
<耦合系数>	参数 3 耦合系数
	耦合系数说明了耦合轴和引导轴位移之间的关系。
	<耦合系数**>**= 耦合轴位移/ 引导轴位移
	类型： REAL
	预设置： 1
	负值表明引导轴和耦合轴在相反方向运行。
	如在编程中未指定耦合系数，则耦合系数 1 自动生效。
TRAILOF	关闭耦合组合的指令
	有效性： 模态
	有 2 个参数的 TRAILOF 只会关闭指定引导轴的耦合： TRAILOF(<跟随轴>,<引导轴>)
	如果一个耦合轴拥有 2 个引导轴，可调用带有 3 个参数的 TRAILOF 来关闭这两个耦合： TRAILOF(<跟随轴>,<引导轴>,<引导轴 2>)
	如果编程了 TRAILOF，而没有指定引导轴，也会给出相同的结果： TRAILOF(<跟随轴>)

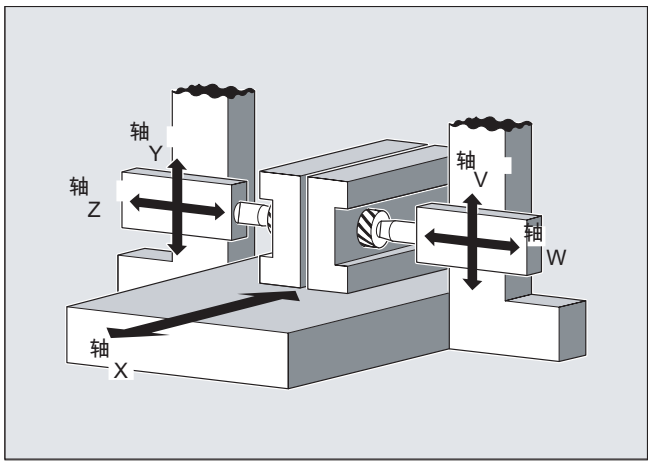
说明

耦合运动始终在基准坐标系（BCS)中进行。

可同时激活的耦合组合的数量只由机床上现有的轴的组合方法限制。

示例

须根据展示的轴结构加工工件两面。 据此应构成 2 个耦合组合。



程序代码	注释
…	
N100 TRAILON(V,Y)	; 启用第 1 个耦合组合
N110 TRAILON(W,Z,-1)	; 启用第 2 个耦合组合。 耦合系数为负： 耦合轴以与引导轴相反的方向作相应运动。
N120 G0 Z10	; Z 轴和 W 轴以相反的轴向进给。
N130 G0 Y20	; Y 轴和 V 轴以相同的轴向进给。
…	
N200 G1 Y22 V25 F200	; 叠加耦合轴"V"轴的某个相关和不相关的运动。
…	
TRAILOF(V,Y)	; 关闭第 1 个耦合组合。
TRAILOF(W,Z)	; 关闭第 2 个耦合组合。

其它信息

轴类型

一个耦合组合可以由线性轴和回转轴的任意组合构成。 一个模拟轴也可在此被定义为引导轴。

耦合轴

一个耦合轴最多可同时指定 2 个引导轴。 在不同的耦合组合中指定引导轴。

可以为耦合轴编程所有系统提供的运行指令(G0, G1, G2, G3, ...).。 除了单独定义的位移，耦合轴还会按照耦合系数运行从引导轴导出的位移。

动态性能限制

动态性能的限制取决于激活耦合组合的方式：

- 在零件程序中激活

  如果在零件程序中激活耦合，而所有的引导轴被用作当前生效的编程轴，那么在引导轴运行时会考虑所有耦合轴的动态性能，避免出现过载。

  如果在零件程序中激活了耦合，而其中的引导轴没有被用作当前生效通道中的编程轴($AA_TYP ≠ 1)，那么在引导轴运行时不会考虑耦合轴的动态性能。 因此，如果某个耦合轴的动态性能稍稍低于耦合要求的水平，会使该轴出现过载。

- 在同步中激活

  如果在同步中激活耦合，那么在引导轴运行时不会考虑耦合轴的动态性能。 因此，如果某个耦合轴的动态性能稍稍低于耦合要求的水平，会使该轴出现过载。

⚠小心
如果一个耦合组合 • 在同步中 • 或在零件程序中被激活，其中的引导轴不是耦合轴通道中的编程轴， 那么用户和机床制造商应负责采取相应的措施，避免引导轴的运行导致耦合轴出现过载。

耦合状态

在零件程序中可以采用以下系统变量查询轴的耦合状态：

$AA_COUP_ACT[<轴>]

值	含义
0	无耦合有效
8	耦合运行生效

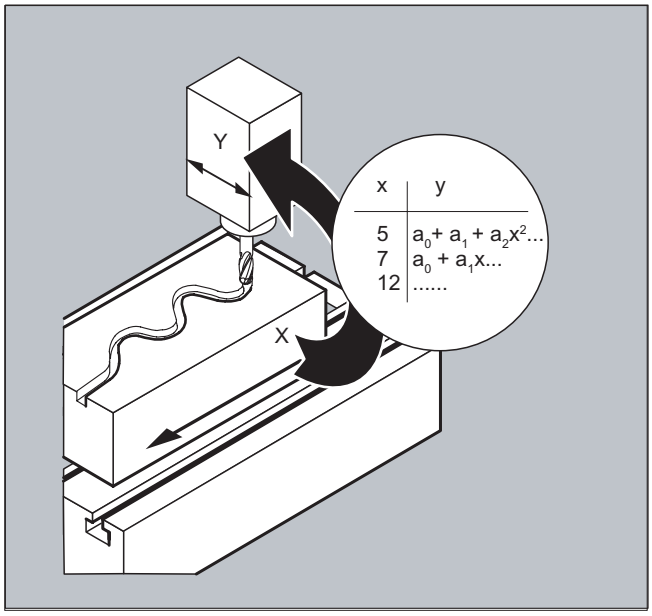
9.2 曲线图表 (CTAB)

功能

借助曲线图表可以编程两个轴（引导轴和跟随轴）之间的位置关系和速度关系。 曲线图表的定义在零件程序中进行。

应用

曲线图表替代了机械凸轮。 通过实现引导值和跟随值之间的函数关联，曲线图表构成了轴向引导值耦合的基础。 在相应的编程中，控制系统从相互所属的引导轴和跟随轴的位置中计算出一个与凸轮相应的多项式。

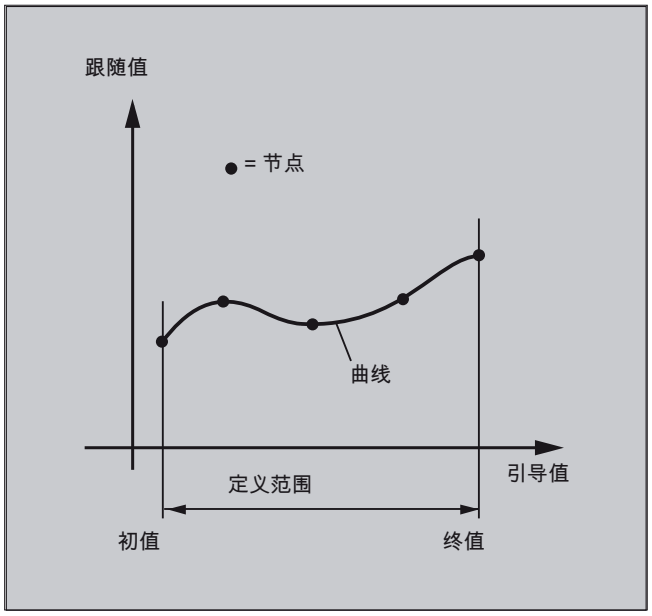


9.2.1 定义曲线图表(CTABDEF, CATBEND)

功能

一个曲线图表所描述的是一个零件程序或者一个零件程序段，其特点是前面插入 CTABDEF 且使用指令 CTABEND 结束。

在该程序段范围内，通过运动指令将引导轴的各个位置一一指定给跟随轴的位置，这些跟随值位置用来作为计算曲线的节点，曲线的形式至多为 5 阶多项式。



前提条件

在定义曲线图表前，必须通过相应的机床数据定义来预留足够的存储容量(→ 机床制造商！）。

句法

```
CTABDEF(<跟随轴>,<引导轴>,<n>,<周期性>[,<存储地点>])
...
CTABEND
```

含义

CTABDEF ( )	曲线图表定义的开始
CTABEND	曲线图表定义的结束
<跟随轴>	需要通过曲线图表计算其运行的轴
<引导轴>	提供引导值以计算跟随轴运行的轴
<n>	曲线图表的编号(**ID**) 曲线图表的编号是唯一的，和存储地点无关。 在静态和动态 NC 存储器中不能出现带有相同编号的图表。
<周期性>	图表周期性 0 图表不具有周期性，即使是回转轴也只执行一次

	1　引导轴上图表具有周期性
	2　引导轴和跟随轴上，图表具有周期性
<存储地点>	存储地点的说明（可选）
	"SRAM"　曲线图表保存在**静态** NC 存储器中。
	"DRAM"　曲线图表保存在**动态** NC 存储器中。
	提示： 如果没有为该参数编程任何值，机床数据 MD20905 $MC_CTAB_DEFAULT_MEMORY_TYPE 中设置的默认存储地点会生效。

说明

覆盖

只要一个新曲线图表定义时其编号(<n>)被使用，这个曲线图表将被覆盖。（特例：曲线图表在某个轴耦合中被激活或已被 CTABLOCK 禁用）。 **在覆盖曲线图表时不会给出相应的警告！**

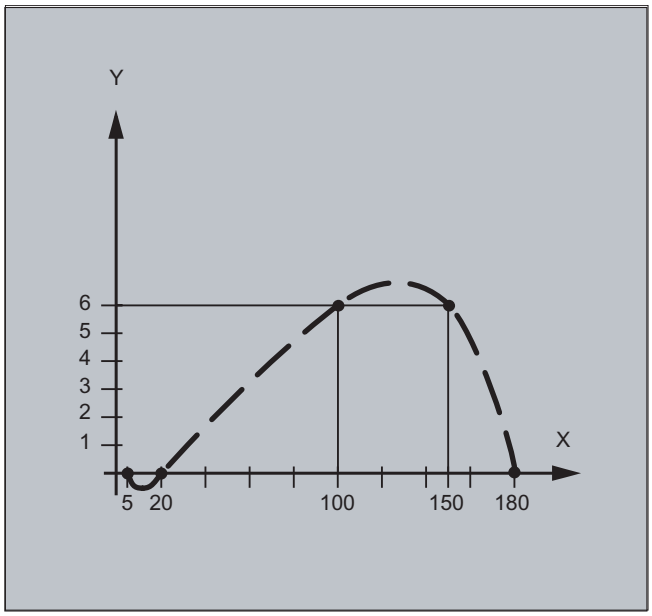
示例

示例 1： 程序段，作为曲线图表定义

不改变一个程序段，用于定义一个曲线图表。 其中所出现的预处理指令 STOPRE 会被保留，在该程序段不再用作图表定义，而 CTABDEF 和 CTABEND 也被删除后，立即恢复生效。

程序代码	注释
…	
CTABDEF(Y,X,1,1)	; 定义一个曲线图表。
…	
IF NOT ($P_CTABDEF)	
STOPRE	
ENDIF	
…	
CTABEND	

示例 2： 定义一个非周期性的曲线图表



程序代码	注释
N100 CTABDEF(Y,X,3,0)	; 开始定义一个带有编号 3 的非周期性曲线图表。
N110 X0 Y0	; 第 1 个运动指令，确定起始值和第 1 个节点： 引导值： 0，跟随值： 0
N120 X20 Y0	; 第 2 个节点： 引导值： 0…20，跟随值： 起始值…0
N130 X100 Y6	; 第 3 个节点： 引导值： 20…100，跟随值： 0…6
N140 X150 Y6	; 第 4 个节点： 引导值： 100…150，跟随值： 6…6
N150 X180 Y0	; 第 5 个节点： 引导值： 150…180，跟随值： 6…0
N200 CTABEND	; 结束定义。 曲线图表在其内部示意图中会生成为最大 5 阶多项式。 视模态选择的零件程序状态插补方式（圆弧插补、直线插补、样条插补）而定，用规定的节点对曲线段再次进行计算。 再次恢复到定义开始前的零件程序状态。

示例 3： 定义一个周期性曲线图表

定义一个周期性曲线图表，带有编号 2，引导值范围 0～360，跟随轴运动从 0 到 45 并且返回到 0：

程序代码	注释
N10 DEF REAL DEPPOS	
N20 DEF REAL GRADIENT	
N30 CTABDEF(Y,X,2,1)	; 开始定义。
N40 G1 X=0 Y=0	
N50 POLY	
N60 PO[X]=(45.0)	
N70 PO[X]=(90.0) PO[Y]=(45.0,135.0,-90)	
N80 PO[X]=(270.0)	
N90 PO[X]=(315.0) PO[Y]=(0.0,-135.0,90)	
N100 PO[X]=(360.0)	
N110 CTABEND	; 结束定义。
; 通过耦合 Y 到 X 对曲线进行测试	
N120 G1 F1000 X0	
N130 LEADON(Y,X,2)	
N140 X360	
N150 X0	
N160 LEADOF(Y,X)	
N170 DEPPOS=CTAB(75.0,2,GRADIENT)	; 在引导值为 75.0 时读图表功能。
N180 G0 X75 Y=DEPPOS	; 引导轴和跟随轴的定位。
; 启用耦合之后，无需对跟随轴进行同步。	
N190 LEADON(Y,X,2)	
N200 G1 X110 F1000	
N210 LEADOF(Y,X)	
N220 M30	

其它信息

曲线图表的初值和终值

曲线图表定义范围开始的初值是曲线图表定义之内相关轴位置的说明（第一个运动指令）。 曲线图表的定义范围的终值相应地由最后的运行指令决定。

可用的语言范围

在曲线图表的定义内，可使用整个 NC 语言范围。

说明

在曲线图表定义中不允许以下指令：

- 预处理停止
- 引导轴运动过程中的跳转（例如当切换坐标转换时）
- 单独的跟随轴的运动指令
- 引导轴的反向运动，即引导轴的位置必须始终唯一
- 不同程序级的 CTABDEF 和 CTABEND 指令。

模态指令的有效性

所有在曲线图表定义之内激活的模态指令均在曲线图表定义结束处失效。 因此，图表定义所位于的零件程序，在图表定义的前后处于相同的状态。

R 参数赋值

编程 CTABEND 后，图表定义范围内的 R 参数赋值被复位。

示例：

程序代码	注释
...	
R10=5 R11=20	; R10=5
...	
CTABDEF	
G1 X=10 Y=20 F1000	
R10=R11+5	; R10=25
X=R10	
CTABEND	
...	; R10=5

ASPLINE, BSPLINE, CSPLINE 的激活

如果在一个曲线图表定义 CTABDEF ... CTABEND 内激活一个 ASPLINE, BSPLINE 或 CSPLINE，则在激活该样条前至少应当编程一个起始点。要避免在 CTABDEF 之后立即激活，否则样条会依赖于曲线图表定义之前的当前轴位置。

示例：

程序代码

```
...
CTABDEF(Y,X,1,0)
X0 Y0
ASPLINE
X=5 Y=10
X10 Y40
...
CTABEND
```

重复使用曲线图表

如果图表存储在 NC 静态存储器（SRAM）中，则通过曲线图表计算出的、主动轴和跟随轴的函数关系会保留在所选择的图表号之下，即使零件程序结束或断电。

保存在动态存储器（DRAM）中的图表会在上电时被删除，必须再次创建。

已建立的曲线图表可用到引导轴和跟随轴的任意轴组合上，而和建立曲线图表时使用的轴没有关系。

曲线图表的覆盖

只要一个新曲线图表定义时其编号被使用，这个曲线图表将被覆盖。

特例： 曲线图表已在某个轴耦合中激活或者已被 CTABLOCK 禁用。

说明

在覆盖曲线图表时中不给出相应的警告。

曲线图表定义生效？

使用系统变量$P_CTABDEF 可随时从零件程序中查询曲线图表定义是否已激活。

取消曲线图表定义

将定义曲线图表的语句用括号括起来后，零件程序段就可重新作为真实的零件程序使用。

通过“从外部执行”载入曲线图表

通过“从外部执行”载入曲线图表时，必须通过机床数据 MD18360 $MN_MM_EXT_PROG_BUFFER_SIZE 正确选择加载缓存器(DRAM)的容量，从而可以同时在加载缓存器中保存完整的曲线表定义。 否则将发出报警，停止零件程序的处理。

跟随轴的跳转

根据机床数据
MD20900 $MC_CTAB_ENABLE_NO_LEADMOTION
的设置，在缺少引导轴运动时允许跟随值跳转。

9.2.2 检查曲线图表的存在性(CTABEXISTS)

功能

通过指令 CTABEXISTS 可以检查，NC 存储器中是否存在某个曲线图表号。

句法

```
CTABEXISTS(<n>)
```

含义

CTABEXISTS	检查，在静态或动态 NC 存储器中是否存在编号为<n>的曲线图表
	0 图表不存在
	1 图表存在
<n>	曲线图表的编号(ID)

9.2.3 删除曲线图表(CTABDEL)

功能

使用 CTABDEL 可以删除曲线图表。

说明
在轴耦合中生效的曲线图表不能被删除。

句法

```
CTABDEL(<n>)
CTABDEL(<n>,<m>)
```

```
CTABDEL(<n>,<m>,<存储地点>)
CTABDEL( )
CTABDEL(,,<存储地点>)
```

含义

CTABDEL	用于删除曲线图表的指令
<n>	待删除的曲线图表的编号(**ID**)
	在删除曲线图表范围 CTABDEL(<n>,<m>)时，用<n>指定范围内第一个曲线图表的编号。
<m>	在删除曲线图表范围 CTABDEL(<n>,<m>)时，用<m>指定范围内最后一个曲线图表的编号。
	<m>必须大于<n>!
<存储地点>	存储地点的说明（可选）
	如果删除时**没有指定**存储地点，则删除静态和动态 NC 存储器中指定的曲线图表。
	如果删除时**指定了**存储地点，则只删除指定存储器中指定的曲线图表。 其他曲线图表保持不变。
	"SRAM" 删除**静态** NC 存储器中的曲线图表
	"DRAM" 删除**动态** NC 存储器中的曲线图表

如果编程 CTABDEL 时没有指定需要删除的曲线图表，则删除**所有**或指定存储器中的所有曲线图表。

CTABDEL( )	删除静态和动态 NC 存储器中的所有曲线图表
CTABDEL(,,"SRAM")	删除静态存储器中的所有曲线图表
CTABDEL(,,"DRAM")	删除动态存储器中的所有曲线图表

说明

需要删除多个曲线图表 CTABDEL(<n>,<m>)或 CTABDEL()时，如果至少其中有一个在耦合运动中生效，则不执行删除指令，即：**不**删除指定的曲线图表。

9.2.4 禁止删除和覆盖曲线图表(CTABLOCK, CTABUNLOCK)

功能

可以设置“禁止删除和覆盖曲线图表”来保护曲线图表。 该禁止可以随时被取消。

句法

激活锁定功能：

```
CTABLOCK(<n>)
CTABLOCK(<n>,<m>)
CTABLOCK(<n>,<m>,<存储地点>)
CTABLOCK()
CTABLOCK(,,<存储地点>)
```

取消锁定功能：

```
CTABUNLOCK(<n>)
CTABUNLOCK(<n>,<m>)
CTABUNLOCK(<n>,<m>,<存储地点>)
CTABUNLOCK()
CTABUNLOCK(,,<存储地点>)
```

含义

CTABLOCK	**激活**禁止删除/覆盖的指令
CTABUNLOCK	**取消**禁止删除/覆盖的指令 CTABUNLOCK 会再次激活被 CTABLOCK 锁定的曲线图表。 在激活的耦合中生效的图表继续保持锁定状态并不能被删除。 一旦由于耦合失效而取消了禁止，CTABLOCK 的锁定功能就被取消。 从而可以删除此图表。 而无需再次调用 CTABUNLOCK。
<n>	待锁定/解除锁定的曲线图表的编号(ID) 在锁定/解除锁定曲线图表范围 CTABLOCK(<n>,<m>)/CTABUNLOCK(<n>,<m>)时，用<n>指定范围内第一个曲线图表的编号。

<m>	在锁定/解除锁定曲线图表范围 CTABLOCK(<n>,<m>)/CTABUNLOCK(<n>,<m>)时，用<m>指定范围内最后一个曲线图表的编号。 <m>必须大于<n>！
<存储地点>	存储地点的说明（可选） 如果在锁定/解除锁定时**没有指定**存储地点，则该指令对静态和动态 NC 存储器中指定的曲线图表生效。 如果在锁定/解除锁定时**指定了**存储地点，则该指令只对指定存储器中指定的曲线图表生效。 其他的曲线图表不会受影响。 "SRAM" 锁定/解除锁定**静态** NC 存储器中的曲线图表 "DRAM" 锁定/解除锁定**动态** NC 存储器中的曲线图表

如果编程 CTABLOCK/CTABUNLOCK 时没有指定需要锁定或解除锁定的曲线图表，则该指令对**所有**或指定存储器中的所有曲线图表生效。

CTABLOCK()	锁定静态和动态 NC 存储器中的所有曲线图表
CTABLOCK(,,"SRAM")	锁定静态存储器中的所有曲线图表
CTABLOCK(,,"DRAM")	锁定动态存储器中的所有曲线图表

CTABUNLOCK()	解除锁定静态和动态 NC 存储器中的所有曲线图表
CTABUNLOCK(,,"SRAM")	解除锁定静态存储器中的所有曲线图表
CTABUNLOCK(,,"DRAM")	解除锁定动态存储器中的所有曲线图表

9.2.5 曲线图表： 确定图表属性(CTABID, CTABISLOCK, CTABMEMTYP, CTABPERIOD)

功能

通过该指令可以查询曲线图表的重要属性，如图表编号、锁定状态、存储地点和周期性。

句法

```
CTABID(<p>)
CTABID(<p>,<存储地点>)
CTABISLOCK(<n>)
CTABMEMTYP(<n>)
```

TABPERIOD(<n>)

含义

CTABID	返回一个**图表号**，该图表号在指定的存储器中被存为第<p>个曲线图表。 示例： CTABID(1,"SRAM")返回静态 NC 存储器中第一个曲线图表的编号。 此处，第一个曲线图表相当于最高编号的图表。 **提示：** 如果在前后两个 CTABID 的调用期间，存储器中的图表顺序发生变化，例如，由于用 CTABDEL 删除了某个图表，则 CTABID(<p>,...)会在 <p>号相同时返回另一个图表，而不是之前的一个。
CTABISLOCK	返回编号为<n>的曲线图表的**锁定状态**： 0 图表未锁定 1 图表被 CTABLOCK 锁定 2 图表被激活的耦合锁定 3 图表被 CTABLOCK 和激活的耦合锁定 -1 图表不存在
CTABMEMTYP	返回编号为<n>的曲线图表的**存储地点**： 0 静态 NC 存储器中的曲线图表 1 动态 NC 存储器中的曲线图表 -1 图表不存在
CTABPERIOD	返回编号为<n>的曲线图表的**周期性**： 0 图表为非周期性 1 引导轴上图表具有周期性 2 引导轴和跟随轴上，图表具有周期性 -1 图表不存在
<p>	存储器中的输入号
<n>	曲线图表的编号(**ID**)
<存储地点>	存储地点的说明（可选） "SRAM" **静态** NC 存储器

"DRAM" **动态** NC 存储器

提示：
如果没有为该参数编程任何值，机床数据 MD20905 $MC_CTAB_DEFAULT_MEMORY_TYPE 中设置的默认存储地点会生效。

9.2.6 读取曲线图表值 (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX)

功能

在零件程序中可以读取以下曲线图表值：

- 曲线图表开头和结尾上的引导轴值和跟随轴值
- 曲线段开头和结尾的跟随轴值
- 一个引导轴值的跟随轴值
- 一个跟随轴值的引导轴值
- 跟随轴的最大值和最小值
  - 在图表的整个定义范围内

    或者
  - 在定义的图表间隔内

句法

```
CTABTSV(<n>,<斜率>[,<跟随轴>])
CTABTEV(<n>,<斜率>[,<跟随轴>])
CTABTSP(<n>,<斜率>[,<引导轴>])
CTABTEP(<n>,<斜率>[,<引导轴>])
CTABSSV(<引导值>,<n>,<斜率>[,<跟随轴>])
CTABSEV(<引导值>,<n>,<斜率>[,<跟随轴>])
CTAB(<引导值>,<n>,<斜率>[,<跟随轴>,<引导轴>]
CTABINV(<跟随值>,<近似值>,<n>,<斜率>[,<跟随轴>,<引导轴>]
CTABTMIN(<n>[,<跟随轴>])
CTABTMAX(<n>[,<跟随轴>])
CTABTMIN(<n>,<a>,<b>[,<跟随轴>,<引导轴>])
CTABTMAX(<n>,<a>,<b>[,<跟随轴>,<引导轴>])
```

含义

CTABTSV:	读取编号为<n>的图表**开头**的跟随轴值
CTABTEV:	读取编号为<n>的图表**结尾**的跟随轴值
CTABTSP:	读取编号为<n>的图表**开头**的引导轴值
CTABTEP:	读取编号为<n>的图表**结尾**的引导轴值
CTABSSV:	指定的引导轴值，即<引导值>对应的曲线段**开头**的跟随轴值
CTABSEV:	指定的引导轴值，即<引导值>对应的曲线段**结尾**的跟随轴值
CTAB:	读取指定的引导轴值，即<引导值>对应的跟随轴值
CTABINV:	读取指定的跟随轴值，即<跟随值>对应的引导轴值
CTABTMIN:	确定跟随轴的**最小值**： • 在图表的整个定义范围内 或者 • 在一个定义的间隔<a> ... <b>内
CTABTMAX:	确定跟随轴的**最大值**： • 在图表的整个定义范围内 或者 • 在一个定义的间隔<a> ... <b>内
<n>：	曲线图表的编号(**ID**)
<斜率>:	在参数<斜率>内返回了测定位置上曲线图表函数的**斜率**。
<跟随轴>:	需要通过曲线图表计算其运行的轴（可选）
<引导轴>:	提供引导值以计算跟随轴运行的轴（可选）
<跟随值>:	CTABINV 中，用于读取相应引导值的跟随值
<引导值>:	引导值： • CTAB 中，用于读取相应跟随值 或者 • CTABSSV/CTABSEV 中，用于选择曲线段
<近似值>:	CTABINV 中，引导值轴和跟随轴值的对应关系并不强制是唯一的。 CTABINV 为此需要一个参数，即用于所求引导轴值的近似值。
<a>:	CTABTMIN/CTABTMAX 中引导值间隔的**下限**

<b>: CTABTMIN/CTABTMAX 中引导值间隔的**上限**

提示：
引导值间隔<a> ... <b>必须在曲线图表的定义范围内。

示例

示例 1：

确定曲线图表开头和末尾的跟随轴值和引导轴值，以及整个定义范围内跟随轴的最大值和最小值。

程序代码	注释
N10 DEF REAL STARTPOS	
N20 DEF REAL ENDPOS	
N30 DEF REAL STARTPARA	
N40 DEF REAL ENDPARA	
N50 DEF REAL MINVAL	
N60 DEF REAL MAXVAL	
N70 DEF REAL GRADIENT	
...	
N100 CTABDEF(Y,X,1,0)	; 开始图表定义
N110 X0 Y10	; 第 1 个图表曲线段的开始位置
N120 X30 Y40	; 第 1 个图表曲线段的结束位置 = 第 2 个图表曲线段的开始位置
N130 X60 Y5	; 第 2 个图表曲线段的结束位置＝...
N140 X70 Y30	
N150 X80 Y20	
N160 CTABEND	; 结束图表定义。
...	
N200 STARTPOS=CTABTSV(1,GRADIENT)	; 曲线图表开头的跟随轴值 = 10
N210 ENDPOS=CTABTEV(1,GRADIENT)	; 曲线图表结尾的跟随轴值 = 20
N220 STARTPARA=CTABTSP(1,GRADIENT)	; 曲线图表开头的引导轴值 = 0
N230 ENDPARA=CTABTEP(1,GRADIENT)	; 曲线图表结尾的引导轴值 = 80
N240 MINVAL=CTABTMIN(1)	; 确定 Y=5 时跟随轴的最小值
N250 MAXVAL=CTABTMAX(1)	; 确定 Y=40 时跟随轴的最大值

示例 2：

确定引导轴值 X=30 所属的曲线段开头和结尾的跟随轴值。

```
程序代码                                    注释
N10 DEF REAL STARTPOS
N20 DEF REAL ENDPOS
N30 DEF REAL GRADIENT
...
N100 CTABDEF(Y,X,1,0)                       ; 开始图表定义。
N110 X0 Y0                                  ; 第 1 个图表曲线段的开始位置
N120 X20 Y10                                ; 第 1 个图表曲线段的结束位置 = 第 2 个图表曲线段的开始位置
N130 X40 Y40                                  第 2 个图表曲线段的结束位置=...
N140 X60 Y10
N150 X80 Y0
N160 CTABEND                                ; 结束图表定义。
...
N200 STARTPOS=CTABSSV(30.0,1,GRADIENT)      ; 曲线段 2 的开始位置 Y = 10
N210 ENDPOS=CTABSEV(30.0,1,GRADIENT)        ; 曲线段 2 的结束位置 Y = 40
```

其它信息

同步中的应用

所有用于读取曲线图表值的指令也可以应用在同步中，参见章节“运行同步”。

在使用指令 CTABINV, CTABTMIN 和 CTABTMAX 时应注意：

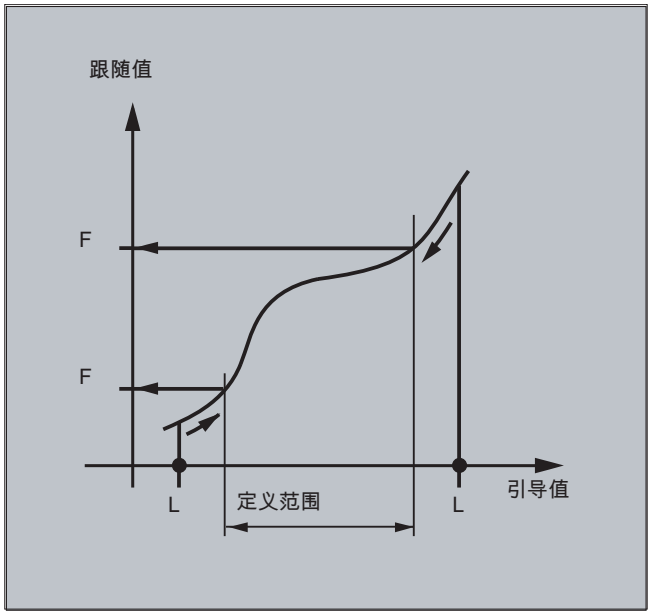
- 在执行指令时应具有足够的 NC 性能

  或者

- 在调用前应询问曲线图表段的数量，从而可以划分相关的图表。

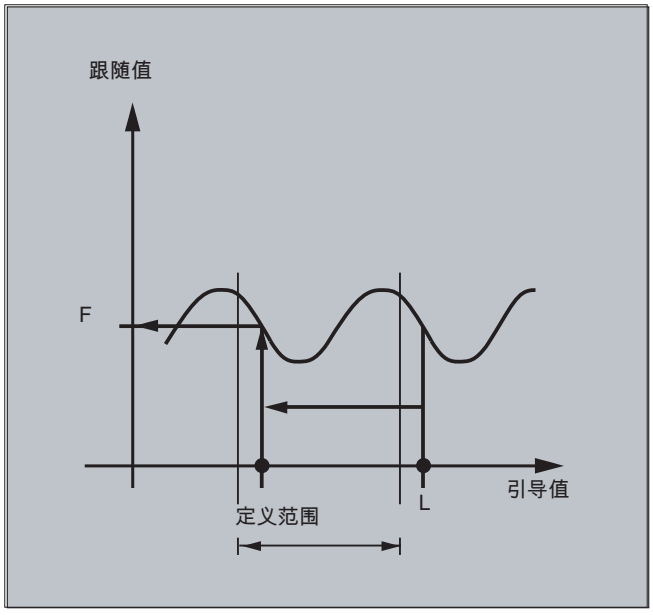
CTAB，非周期性曲线图表

如果指定的<引导值>在定义范围外，则输出上限值或下限值作为跟随值：



CTAB，周期性曲线图表

如果指定的<引导值>在定义范围之外，则取模计算定义范围内的引导值，并输出相应的跟随值。



用于 CTABINV 的近似值

CTABINV 指令需要一个用于所求引导值的近似值。 CTABINV 返回一个和近似值最接近的引导值。例如，近似值可以是上一个插补周期中的引导值。

曲线图表函数的斜率

斜率的输出值(<斜率>)可以计算相应位置上的引导轴或跟随轴的速度。

引导轴或跟随轴的说明

如果以不同的长度单位定义了引导轴和跟随轴，引导轴和/或跟随轴的可选说明就比较重要。

CTABSSV, CTABSEV

指令 CTABSSV 和 CTABSEV 在下列情况下**不适合**查询对已编程的曲线段：

- 编程了圆弧或者渐开线。
- 倒角或倒圆已使用 CHF, RND 激活。
- 使用 G643 进行精磨的功能已激活。
- NC 程序段压缩器已使用例如 COMPON/COMPCURV/COMPCAD 激活。

9.2.7 曲线图表： 检查资源使用率(CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL)

功能

使用该指令，编程人员可以了解目前被曲线图表、图表段和多项式占用的资源情况。

句法

```
CTABNO
CTABNOMEM(<存储地点>)
CTABFNO(<存储地点>)
CTABSEGID(<n>,<存储地点>)
CTABSEG(<存储地点>,<段类型>)
CTABFSEG(<存储地点>,<段类型>)
CTABMSEG(<存储地点>,<段类型>)
CTABPOLID(<n>)
CTABPOL(<存储地点>)
CTABFPOL(<存储地点>)
CTABMPOL(<存储地点>)
```

含义

CTABNO	确定**已定义**的曲线图表的总数量（在静态和动态 NC 存储器中）
CTABNOMEM	确定指定的<存储地点>中**已定义**的曲线图表的数量。
CTABFNO	确定指定的<存储地点>中**还可以定义**的曲线图表的数量。
CTABSEGID	确定编号为<n>的曲线图表、指定<段类型>的段数量。
CTABSEG	确定指定的<存储地点>中、指定<段类型>的**已使用**的段数量
CTABFSEG	确定指定的<存储地点>中、指定<段类型>的**还可使用**的段数量
CTABMSEG	确定指定的<存储地点>中、指定<段类型>的**允许的最大**曲线段数量
CTABPOLID	确定编号为<n>的曲线图表的多项式数量。
CTABPOL	确定指定的<存储地点>中**已使用**的曲线多项式的数量。
CTABFPOL	确定指定的<存储地点>中**还可以使用**的曲线多项式的数量。
CTABMPOL	确定指定的<存储地点>中**允许的最大**曲线多项式的数量。
<n>	曲线图表的编号(**ID**)
<存储地点>	存储地点的说明（可选） "SRAM" **静态** NC 存储器 "DRAM" **动态** NC 存储器 **提示：** 如果没有为该参数编程任何值，机床数据 MD20905 $MC_CTAB_DEFAULT_MEMORY_TYPE 中设置的默认存储地点会生效。
<段类型>	段类型的说明（可选） "L" 线性段 "P" 多项式段 **提示：** 如果没有为该参数编程任何值，则输出所有线性段和多项式段。

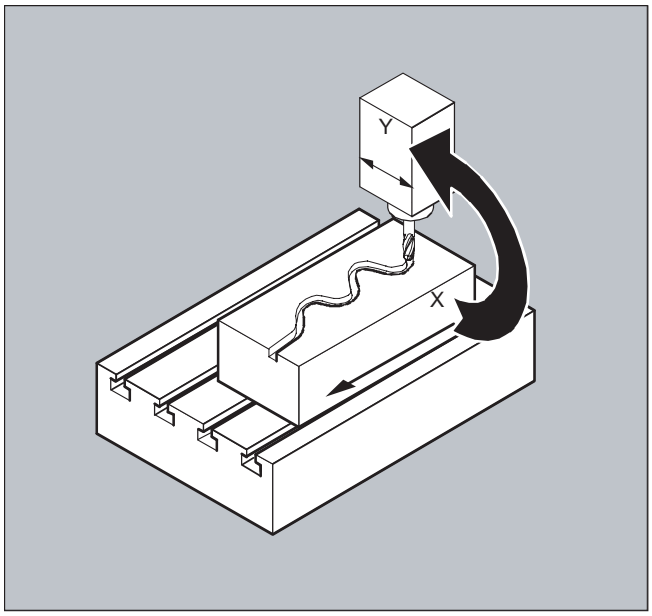
9.3 轴向引导值偶合 (LEADON, LEADOF)

说明

该功能不能用于 SINUMERIK 828D。

功能

在轴的引导值耦合时同步运行一个引导轴和一个跟随轴。 同时，跟随轴的相应位置通过一个曲线图表或者通过一个从该图表算得的多项式已明确分配给引导轴的一个（有可能是模拟的）位置。



引导轴 是给曲线图表发送输入值的那个轴。 **跟随轴** 是接收通过曲线图表算得的位置的那个轴。

实际值和设定值耦合

作为引导值，即用于计算位置的输出值可以使用：

- 引导轴位置的实际值： 实际值耦合
- 引导轴位置的设定值： 设定值耦合

引导值耦合一直在基准坐标系中有效。

关于曲线图表的建立可参阅“曲线图表”一章。
关于引导值耦合可参阅 /FB/, M3, 耦合运行和引导值耦合。

句法

```
LEADON(F 轴,L 轴,n)
LEADOF（F 轴，L 轴）
```

或者在不给定引导轴的情况下关闭：

```
LEADOF（F 轴）
```

引导值耦合既可以从零件程序、也可以在运动过程中从同步动作开始启用和关闭，参阅“运动同步动作”一章。

含义

LEADON	启用引导值耦合
LEADOF	关断引导值耦合
F 轴	跟随轴
L 轴	引导轴
n	曲线图表编号
$SA_LEAD_TYPE	在设定值和实际值耦合之间转换

关闭引导值耦合， LEADOF

如关闭引导轴耦合，跟随轴可重新成为正常的指令轴！

轴向引导值耦合和各种运行状态，RESET

与机床参数的设定有关，引导值耦合由 RESET 关断。

同步动作所构成的引导值耦合举例

在冲压设备中，在一个引导轴（冲杆轴）和由传输轴与辅助轴构成的传输系统的轴之间，这种传统地机械耦合应由一个电子的耦合系统代替。

这表明了，在一个冲压设备中一个机械的传输系统如何被一个电子的传输系统所代替。耦合和去耦合过程作为**静态同步动作** 实现。

传输轴和辅助轴可通过曲线图表作为跟随轴被引导轴 LW(冲杆轴)控制。

跟随轴

X 进给轴或者纵向轴
YL 闭合轴或者横向轴
ZL 升降轴
U 辊进给，辅助轴

V 校正头，辅助轴
W 润滑装置，辅助轴

作用

作为动作出现在同步动作中的例如有：

- 耦合， LEADON(跟随轴，引导轴，曲线图表编号)
- 去耦合， LEADOF(跟随轴，引导轴)
- 设定实际值， PRESETON(轴，数值)
- 设定标记， $AC_MARKER[i] = 数值
- 耦合方式： 实际/虚拟的引导值
- 轴位置的起动， POS[轴] = 数值

条件

作为条件，对数字快速输入、实时变量 $AC_MARKER 和与逻辑运算子 AND 有关联的位置对比进行分析。

说明

下列举例中将行交错变化、行首空格和**粗体**字仅用于提高编程的可读性。 为了控制，所有在标志行数下的都占一行。

注释

程序代码	注释
	; 定义所有静态同步动作。
	; ****复位标记
N2 $AC_MARKER[0]=0 $AC_MARKER[1]=0 $AC_MARKER[2]=0 $AC_MARKER[3]=0 $AC_MARKER[4]=0 $AC_MARKER[5]=0 $AC_MARKER[6]=0 $AC_MARKER[7]=0	
	; **** E1 0=>1 启用传送耦合
N10 IDS=1 EVERY ($A_IN[1]==1) AND ($A_IN[16]==1) AND ($AC_MARKER[0]==0) DO LEADON(X,LW,1) LEADON(YL,LW,2) LEADON(ZL,LW,3) $AC_MARKER[0]=1	
	; **** E1 0=>1 启用辊进给耦合
N20 IDS=11 EVERY ($A_IN[1]==1) AND ($A_IN[5]==0) AND ($AC_MARKER[5]==0) DO LEADON(U,LW,4) PRESETON(U,0) $AC_MARKER[5]=1	
	; **** E1 0->1 启用校正头耦合
N21 IDS=12 EVERY ($A_IN[1]==1) AND ($A_IN[5]==0) AND ($AC_MARKER[6]==0) DO LEADON(V,LW,4) PRESETON(V,0) $AC_MARKER[6]=1	
	; **** E1 0->1 启用润滑装置耦合
N22 IDS=13 EVERY ($A_IN[1]==1) AND ($A_IN[5]==0) AND ($AC_MARKER[7]==0) DO LEADON(W,LW,4) PRESETON(W,0) $AC_MARKER[7]=1	

程序代码	注释
	; **** E2 0=>1 断开耦合

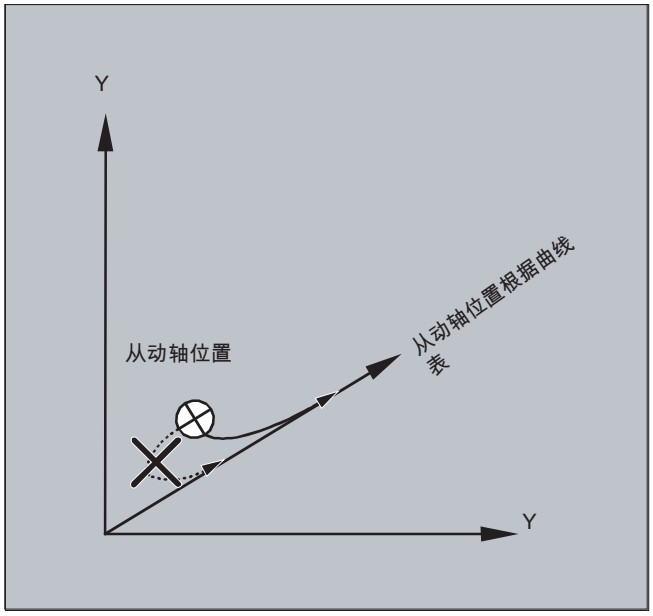
```
N30 IDS=3 EVERY ($A_IN[2]==1)
DO LEADOF(X,LW) LEADOF(YL,LW) LEADOF(ZL,LW) LEADOF(U,LW) LEADOF(V,LW) LEADOF(W,LW) $AC_MARKER[0]=0
$AC_MARKER[1]=0 $AC_MARKER[3]=0 $AC_MARKER[4]=0 $AC_MARKER[5]=0 $AC_MARKER[6]=0 $AC_MARKER[7]=0
....
N110 G04 F01
N120 M30
```

说明

引导值耦合要求引导轴和跟随轴的同步动作。只有跟随轴在根据曲线图表计算出的曲线的公差范围内，引导值耦合打开的情况下，才能达到同步动作。

跟随轴位置的允差范围通过机床数据 MD 37200:COUPLE_POS_POL_COARSE A_LEAD_TYPE 定义。

如跟随轴在引导值耦合接通时还不位于相应的位置上，同步运动自动产生，只要计算所得的跟随轴位置的额定值与其实际的跟随轴位置接近。跟随轴在同步过程中沿着通过跟随轴的额定速度（从引导轴速度中且根据曲线图表 CTAB 算得）所定义的方向运动。

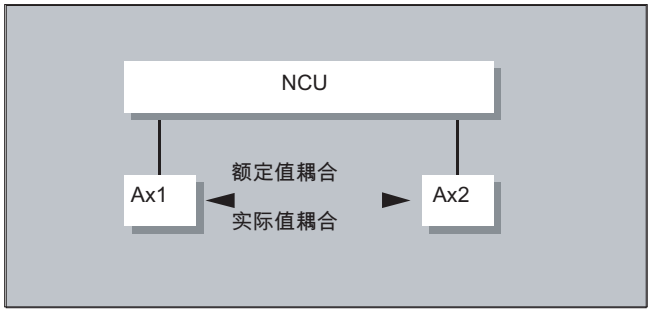


没有同步运动

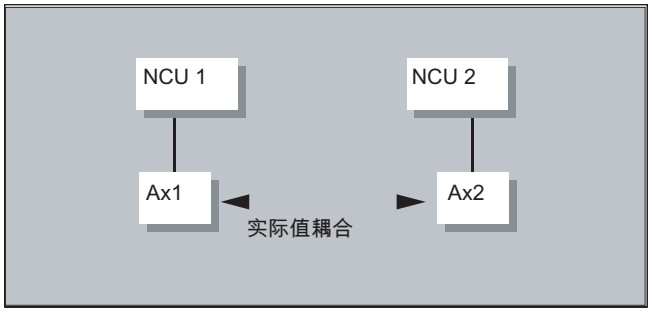
如计算所得的、在引导值耦合打开时跟随轴的额定位置与其实际跟随轴位置相距很远，则不产生同步运动。

实际值和额定值耦合

与实际值耦合相比，额定值耦合提供更好的引导轴和跟随轴之间的同步运动，因此符合预置符合标准。



只有当引导轴和跟随轴插补由相同的 NCU 进行时，才可能进行额定值耦合。一个外部的引导轴上仅可以通过实际值，使跟随轴与引导轴耦合。



一个**转换** 可以通过设置数据 $SA_LEAD_TYPE 进行。

实际值和额定值间的转换应一直在跟随轴静止状态下完成。因为只有在静止状态时，转换后才能重新同步。

应用举例

额定值的读取在大的机床振荡时不能完成。在启动引导值耦合时，如在压力传输时有大的振荡，则有必要从实际值耦合切换到额定值耦合。

引导值模拟，当进行额定值偶合时

通过机床数据可将引导轴插补器与伺服器分开。从而额定值耦合时额定值可以在引导轴不运动的情况下得到。

例如用在同步动作中的通过额定值偶合生成的引导值可以从下列变量中读取：

- $AA_LEAD_P	引导值位置
- $AA_LEAD_V	引导值速度

生成引导值

引导值可有选择性地在其他自动编程的程序中产生。这样产生的引导值将写入变量

- $AA_LEAD_SP 引导值位置
- $AA_LEAD_SV 引导值速度

并从中读取。使用这些变量时，必须将设置数据 $SA_LEAD_TYPE = 2 设定。

耦合的状态

在 NC 零件程序中可以用以下系统变量询问耦合的状态：

$AA_COUP_ACT[轴]
0:没有偶合激活
16:引导值耦合有效

同步动作的状态管理

开关和耦合过程通过实时变量

$AC_MARKER[i] = n
进行管理，使用:
i 标记编号
n 状态参数

9.4 电子齿轮箱 (EG)

功能

使用辅助功能“电子齿轮箱” 可以控制 **跟随轴**运动，使之按照线性运动偏移与最多与五个**引导轴**相关联运动。引导轴和跟随轴之间的关联按照每个引导轴通过耦合系数进行定义。

算出的跟随轴运动分量是由单个引导轴运动分量乘各自的耦合系数通过加法构成的。激活一个 EG 轴组时，可以使跟随轴在某定义的位置上同步。一个齿轮组可以由零件程序：

- 定义，
- 接通，
- 关闭，
- 删除

。

跟随轴的运动可以有选择的被

- 引导轴的额定值以及
- 引导轴的实际值来引导。

作为扩展功能，引导轴和跟随轴之间的非线性关联也可以通过 **曲线列表** (参见章节轨迹特性)来实现。电子齿轮可以串联，即：电子齿轮的跟随轴可以成为另一个电子齿轮的引导轴。

9.4.1 定义电子齿轮箱 (EGDEF)

功能

一副 EG 轴组可以通过跟随轴数据和最少 1 个最多 5 个引导轴带各自耦合类型来确定。

前提条件

EG 轴组定义的前提：

对于跟随轴还不允许定义轴耦合（有的话，必须提前用 EGDEL 删除现有的）。

句法

```
EGDEF（跟随轴，引导轴 1，耦合类型 1，引导轴 2，耦合类型 2，...）
```

含义

EGDEF	电子齿轮定义
跟随轴	由引导轴影响的轴
引导轴 1	影响跟随轴的轴
,..., 引导轴 5	
耦合类型 1	耦合类型
,..., 耦合类型 5	耦合类型不必对所有引导轴都相同，因此必须为每个引导轴单独标注。

值：	含义：
0	跟随轴受相应引导轴的 **实际值** 影响。
1	跟随轴受相应引导轴的 **额定值** 影响。

说明

对 EG 耦合组进行定义时，必须预设耦合系数为零。

说明

EGDEF 触发进给停止。用 EGDEF 进行齿轮定义，在使用时必须保持不变，如果系统中有一个或者多个引导轴通过**曲线图表**影响跟随轴。

示例

程序代码	注释
EGDEF(C,B,1,Z,1,Y,1)	; 定义一个 EG 轴组。 引导轴 B，Z，Y 通过额定值影跟随轴 C。

9.4.2 接通电子齿轮（EGON，EGONSYN，EGONSYNE）

功能

有 3 种型式用于接通 EG 轴组。

句法

型式 1：

在无同步的情况下选择性接通 EG 轴组，通过：
EGON（FA，"程序段转换模式"，LA1，Z1，N1，LA2，Z2，N2，...，LA5，Z5，N5）

型式 2：

在同步的情况下选择性接通 EG 轴组，通过：
EGONSYN（FA，"程序段转换模式"，SynPosFA，[，LAi，SynPosLAi，Zi，Ni]）

型式 3：

在同步的情况下选择性接通 EG 轴组并规定返回模式，通过：
EGONSYNE（FA，"程序段转换模式"，SynPosFA，返回模式[，LAi，SynPosLAi，Zi，Ni]）

含义

型式 1：

FA	跟随轴	
程序段转换模式	可以用下列模式：	
	"NOC"	立即进行程序段转换
	"FINE"	在"精确同步运行"时进行程序段转换
	"COARSE"	在"近似同步运行"时进行程序段转换
	"IPOSTOP"	当额定值同步运行时进行程序段转换
LA1, ... LA5	引导轴	
Z1, ... Z5	耦合系数 i 的分子	
N1, ... N5	耦合系数 i 的分母	
	耦合系数 i = 分子 i /分母 i	

只允许对先前用 EGDEF 进行详细说明的引导轴编程。 必须至少编程一个引导轴。

型式 2：

FA	跟随轴
程序段转换模式	可以用下列模式：
	"NOC" 立即进行程序段转换
	"FINE" 在“精确同步运行”时进行程序段转换
	"COARSE" 在“近似同步运行”时进行程序段转换
	"IPOSTOP" 当额定值同步运行时进行程序段转换
[,LAi,SynPosLAi,Zi,Ni]	(不写方括号)
	最少 1 个，最多 5 个跟随由：
LA1, ... LA5	引导轴
SynPosLAi	i.引导轴的同步位置
Z1, ... Z5	耦合系数 i 的分子
N1, ... N5	耦合系数 i 的分母
	耦合系数 i = 分子 i /分母 i

只允许对先前用 EGDEF 进行详细说明的引导轴编程。 通过为跟随轴(SynPosFA) 和引导轴 (SynPosLA) 编程的“同步定位”，对位置进行定义，其中耦合组当作*同步* 有效。 一旦接通时电子齿轮不处于同步状态，跟随轴就运行到它定义的同步位置。

型式 3：

该参数符合型式 2 中的参数，包括：

返回模式	可以用下列模式：
	"NTGT" 在最佳时间返回下一个齿间隙
	"NTGP" 以最佳路径返回下一个齿间隙
	"ACN" 在负旋转方向上绝对运行回转轴
	"ACP" 在正旋转方向上绝对运行回转轴
	"DCT" 编程同步位置时间最佳
	"DCP" 编程同步位置路径最佳

方案 3 只对与模数引导轴耦合的模数跟随轴有影响。 最佳时间考虑了跟随轴的速度极限。

其它信息

描述接通型式

型式 1：

将引导轴以及跟随轴启动时刻的位置作为“同步位置”保存。 可以使用系统变量 $AA_EG_SYN 读入“同步位置”。

型式 2：

如果取模轴处于耦合关联状态，则其位置值取模降低。 这样就保证了向可能最接近的同步位置运动（所谓的*相对同步*： 例如最接近的齿隙）。 如果没有将“释放跟随轴叠加”共生面信号 DB(30 +轴编号), DBX 26 位 4 发送给跟随轴，就不会向同步位置运动。 与此相反，程序在 EGONSYN－程序段处停止，并且只要上面的信号设置，就给出自删除的报警 16771。

型式 3：

齿间距（度）由下面公式产生： 360 * Zi/Ni. 对于跟随轴处于调用时刻的情况而言，行程优化与时间优化一样，可提供相同的特性。

如果跟随轴已经运动，可使用 NTGP 向下一个齿隙同步运动，不受跟随轴当前速度的影响。 如果跟随轴已经运动，可使用 NTGT 依据跟随轴当前的速度向下一个齿隙同步运动。 有时轴也会制动。

曲线图表

要将某个**曲线图表** 用于引导轴中的某一个引导轴时，就必须：

Ni	线性耦合耦合系数的分母设置为 0。（分母 0 对于线性耦合是不允许的）。 分母零对于控制系统而言表示
Zi	应解释成待使用的曲线表的编号。 带有给定编号的曲线图表在启用时刻必须已经定义。
LAi	引导轴的参数与通过耦合系数耦合时的引导轴参数一样（线性耦合）。

有关使用曲线图表和电子齿轮的级联及其同步的其它提示，请参见

文献：

功能手册 特殊功能；轴耦合和 ESR（M3），章节“联动和引导值耦合”。

当上电、RESET、运行方式转换、搜索时的电子齿轮箱的特性

- 在上电之后**没有** 偶合激活。
- 在复位和运行方式转换之后有效的耦合仍保持。
- 在程序段搜索时，有关开关、删除、定义电子齿轮的指令不予执行和考虑，而是直接跳过。

电子齿轮箱的系统变量

利用电子齿轮的系统变量，零件程序可以求值一个 EG 轴关联的当前状态，有时并做出反应。

电子齿轮的系统变量标记如下：

$AA_EG_ ...

或

$VA_EG_ ...

文献：
系统变量手册

9.4.3 关闭电子齿轮（EGOFS, EGOFC）

功能

有 3 种型式用于关闭 EG 轴组。

编程

型式 1：

句法	含义
EGOFS(跟随轴)	关闭电子齿轮。 跟随轴制动到停止。 此调用删除进给停止。

型式 2：

句法	含义
EGOFS（跟随轴，引导轴 1，…，引导轴 5）	指令的这种参数设定允许**有选择性地**排除各个引导轴对跟随轴的运动的影响。

必须至少指定一个引导轴。 有针对性地中止指定引导轴对跟随轴的影响。 此调用删除进给停止。 如果尚有引导轴保持激活状态，则跟随轴将在其影响下继续运行。 如果所有的引导轴影响都以这种方式关闭，则跟随轴被制动到停止。

型式 3：

句法	含义
EGOFC（跟随主轴 1）	关闭电子齿轮。 跟随主轴以关闭时刻有效的转速/速度继续运行。 此调用删除进给停止。

说明
该型式仅允许用于主轴。

9.4.4 删除某个电子齿轮箱的定义（EGDEL）

功能

在可以删除电子齿轮箱轴组合的定义之前，必须先将其关闭。

编程

句法	含义
EGDEL（跟随轴）	轴关联的耦合定义被删除。 在到达同时激活的轴关联最大个数之前，又可以用 EGDEF 重新定义其它的轴关联。 此调用删除进刀停止。

9.4.5 旋转进给（G95）/电子齿轮箱（FPR）

功能

使用 FPR 指令也可以将一个电子齿轮箱的跟随轴设定成旋转进给的轴。 对于这种情况，以下的特性适用：

- 进给取决于电子齿轮跟随轴的给定速度。
- 给定速度可以由引导轴和取模－引导轴（不是轨迹轴）的速度和相应的耦合系数计算出来。
- 线性或者非模量引导轴的速度比例和跟随轴的叠加运动不予考虑。

9.5 同步主轴

功能

在同步运行中有一个引导主轴（LS）和一个跟随主轴（FS），即**同步主轴对**。 跟随主轴根据确定的功能关系在激活耦合（同步运行）时跟随引导主轴。

可以借助通道专用的机床数据为每个机床固定同步主轴对，或通过用户专用的 CNC 零件程序定义同步主轴对。 根据 NC 通道，可以最多同时运行 2 个同步主轴对。

耦合可以在零件程序中

- 定义以及修改
- 打开
- 关闭
- 删除

。

由此，可以根据软件版本

- 等待同步运行条件
- 修改程序段转换特性
- 选择给定值耦合或实际值耦合的耦合方式，或者规定引导主轴和跟随主轴之间的角度偏差
- 在打开耦合时接受上一个跟随主轴的编程
- 修改一个测量的或一个已知的同步运行偏差

。

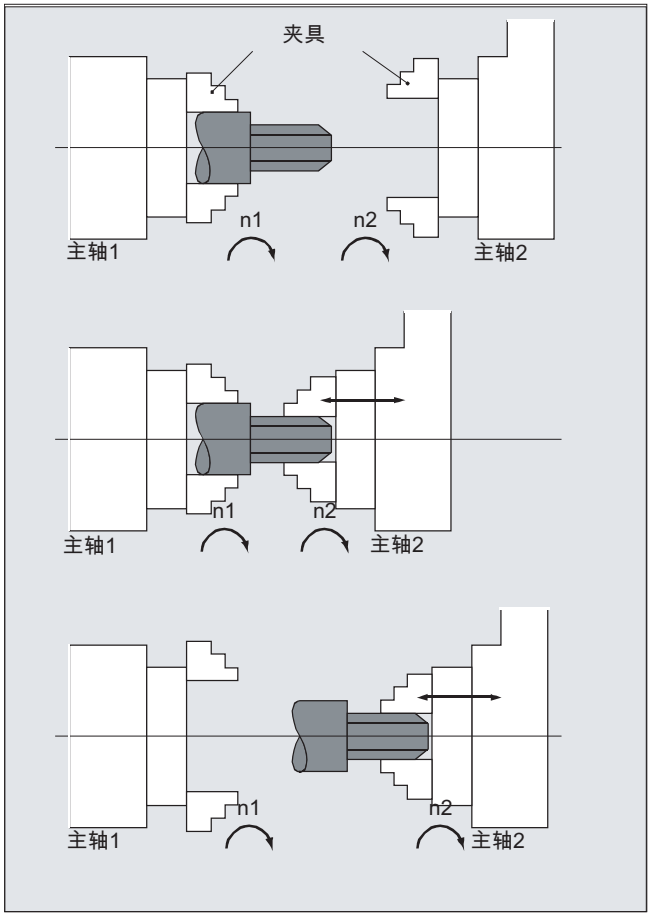
9.5.1 同步主轴： 编程（COUPDEF, COUPDEL, COUPON, COUPONC, COUPOF, COUPOFS, COUPRES, WAITC）

功能

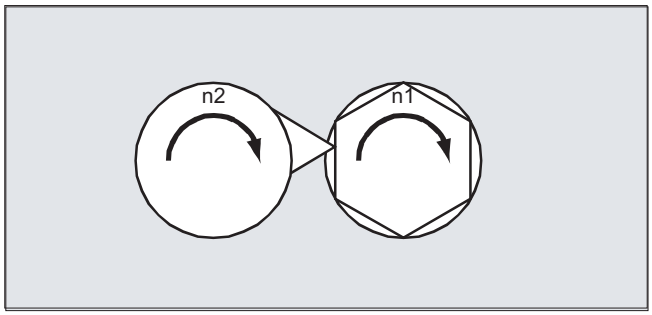
功能“同步主轴”可以使得两个主轴（跟随主轴 FS 和引导主轴 LS）实现同步运行，例如用于飞速递交工件。

该功能具有以下模式：

- 转速同步性($n_{FS} = n_{LS}$)
- 位置同步性（$\phi_{FS} = \phi_{LS}$）
- 位置同步及角度偏移（$\phi_{FS} = \phi_{LS} + \Delta\phi$）



给定一个不等于 1 的、引导主轴和跟随主轴之间的传动比后，可以进行多棱加工（多角车削）。



句法

```
COUPDEF(<跟随主轴>,<引导主轴>,<传动比分子>,<传动比分母>,<程序段切换>,<耦合方式>)
COUPON(<跟随主轴>,<引导主轴>,<跟随主轴位置>)
COUPONC(<跟随主轴>,<引导主轴>)
COUPOF(<跟随主轴>,<引导主轴>,<跟随主轴位置>,<引导主轴位置>)
COUPOFS(<跟随主轴>,<引导主轴>)
COUPOFS(<跟随主轴>,<引导主轴>,<跟随主轴位置>)
COUPRES(<跟随主轴>,<引导主轴>)
COUPDEL(<跟随主轴>,<引导主轴>)
WAITC(<跟随主轴>,<程序段切换>,<引导主轴>,<程序段切换>)
```

说明

简化的写入方法

在使用 COUPOF、COUPOFS、COUPRES 和 COUPDEL 时，可以省略引导主轴，简化写入。

含义

COUPDEF: 定义/更改用户自定义的耦合

COUPON: 激活耦合。 从当前转速开始，跟随主轴与引导主轴同步

COUPONC: 在之前通过编程 M3 S... 或者 M4 S... 激活时接受耦合。

立即接受跟随主轴的转速差。

COUPOF: 解除耦合。

- 立即开始程序段切换：
  COUPOF(<S2>,<S1>)
- 在越过解耦位置后才进行程序段切换 **<**跟随主轴位置> 或 **<**引导主轴位置>:
  COUPOF(<主轴 2>,<主轴 1>,<跟随主轴位置>)
  COUPOF(<主轴 2>,<主轴 1>,<跟随主轴位置>,<引导主轴位置>)

COUPOFS: 通过停止跟随主轴来解除耦合。

程序段切换以最快程序段切换速度进行：

COUPOFS(<主轴 2>,<主轴 1>)

越过解耦位置后才进行程序段切换：

COUPOFS(<主轴 2>,<主轴 1>,<跟随主轴位置>)

COUPRES:	将耦合参数复位到设计的 MD 和 SD
COUPDEL:	删除用户定义的耦合
WAITC:	等待同步运行条件 （更换程序段时取消 IPO 上的 NOC）
<跟随主轴>:	跟随主轴名称

可选参数：

<引导主轴>:	引导主轴名称 带主轴号的说明： 例如 主轴 2, 主轴 1
<传动比分子>, <传动比分母>:	跟随主轴和引导主轴之间的传动比。 <传动比分子> = 分子, <传动比分母> = 分母 缺省设置： <传动比分子> / <传动比分母> = 1.0 ; 可选择设定分母
<程序段切换>:	程序段切换特性 实现程序段切换： "NOC" 立即 "FINE" 达到“精同步” "COARSE" 达到“粗同步” "IPOSTOP" 达到 IPOSTOP，即在设定值同步后（缺省设置） 程序段切换性能模态有效。
<耦合方式>:	耦合方式： 跟随主轴和引导主轴之间的耦合 “DV” 给定值耦合（预设置） “AV” 实际值耦合 “VV” 速度耦合 耦合方式模态有效。
<跟随主轴位置>:	引导主轴和跟随主轴之间的角度偏差 取值范围： 0°… 359,999°
<跟随主轴位置>,<引导主轴位置>:	跟随主轴和引导主轴的解耦位置 “在越过 POS_{FS}（跟随主轴位置）, POS_{LS}（引导主轴位置）后使能程序段切换” 取值范围： 0°… 359,999°

示例

示例 1： 采用引导主轴和跟随主轴工作

编程	注释
	; 引导主轴 = 主主轴 = 主轴 1
	; 跟随主轴 = 主轴 2
N05 M3 S3000 M2=4 S2=500	; 引导主轴转速为 3000 rpm, 跟随主轴转速为 500 rpm。
N10 COUPDEF(S2,S1,1,1,"NOC","Dv")	; 定义耦合（也可以配置）。
...	
N70 SPCON	; 引导主轴采用位置闭环控制（设定值耦合）。
N75 SPCON(2)	; 跟随主轴采用位置闭环控制
N80 COUPON(S2,S1,45)	; 飞速耦合至偏移位置 = 45 度。
...	
N200 FA[S2]=100	; 定位速度 = 100 deg/min
N205 SPOS[2]=IC(-90)	; 负方向上 90 度增量运行。
N210 WAITC(S2,"Fine")	; 等待"精同步"。
N212 G1 X... Y... F...	; 加工
...	
N215 SPOS[2]=IC(180)	; 正方向上 180 度增量运行。
N220 G4 S50	; 停留时间 = 主主轴 50 转
N225 FA[S2]=0	; 激活配置的速度（MD）。
N230 SPOS[2]=IC(-7200)	; 20 转。 在负方向上以配置的速度运行。
...	
N350 COUPOF(S2,S1)	; 飞速脱开耦合，S=S2=3000
N355 SPOSA[2]=0	; 在 0 度时停止跟随主轴。
N360 G0 X0 Y0	
N365 WAITS(2)	; 等待主轴 2。
N370 M5	; 停止跟随主轴。
N375 M30	

示例 2： 转速差编程

编程	注释
	; 引导主轴 = 主主轴 = 主轴 1
	; 跟随主轴 = 主轴 2
N01 M3 S500	; 引导主轴转速为 500 rpm。
N02 M2=3 S2=300	; 跟随主轴转速为 300 rpm。
...	

编程	注释
N10 G4 F1	; 主主轴停留时间。
N15 COUPDEF (S2,S1,-1)	; 耦合系数以传动比-1:1
N20 COUPON(S2,S1)	; 激活耦合。 跟随主轴转速由引导主轴转速和耦合系数得出。
...	
N26 M2=3 S2=100	; 转速差编程。

示例 3： 示例：传送转速差运动

1. 在之前的跟随主轴编程中，使用 COUPON 激活耦合

编程	注释
	; 引导主轴 = 主主轴 = 主轴 1
	; 跟随主轴 = 主轴 2
N05 M3 S100 M2=3 S2=200	; 引导主轴转速为 100 rpm，跟随主轴转速为 200 rpm。
N10 G4 F5	; 停留时间 = 主主轴 5 秒
N15 COUPDEF(S2,S1,1)	; 跟随主轴和引导主轴之间的传动比为 1.0（缺省设置）。
N20 COUPON(S2,S1)	; 飞速和引导主轴耦合。
N10 G4 F5	; 跟随主轴转速为 100 rpm。

2. 在之前的跟随主轴编程中，使用 COUPONC 激活耦合

编程	注释
	; 引导主轴 = 主主轴 = 主轴 1
	; 跟随主轴 = 主轴 2
N05 M3 S100 M2=3 S2=200	; 引导主轴转速为 100 rpm，跟随主轴转速为 200 rpm。
N10 G4 F5	; 停留时间 = 主主轴 5 秒
N15 COUPDEF(S2,S1,1)	; 跟随主轴和引导主轴之间的传动比为 1.0（缺省设置）。
N20 COUPONC(S2,S1)	; 飞速和引导主轴耦合，并接受之前的转速用于 S2。
N10 G4 F5	; S2 以 100 rpm + 200 rpm = 300 rpm 的转速转动

3. 当跟随主轴静止时，使用 COUPON 激活耦合

编程	注释
	; 引导主轴 = 主主轴 = 主轴 1
	; 跟随主轴 = 主轴 2
N05 SPOS=10 SPOS[2]=20	; 跟随主轴 S2 处于定位模式。
N15 COUPDEF(S2,S1,1)	; 跟随主轴和引导主轴之间的传动比为 1.0（缺省设置）。

编程	注释
N20 COUPON(S2,S1)	; 飞速和引导主轴耦合。
N10 G4 F1	; 耦合已关闭，S2 停留在 20 度。

4. 当跟随主轴静止时，使用 COUPONC 激活耦合

说明

定位模式或进给轴模式

如果跟随主轴在耦合前处于定位模式或者进给轴模式，则 COUPON(<跟随轴>,<引导轴>) 和 COUPONC(<跟随轴>,<引导轴>) 中跟随轴的属性是相同的。

注意

引导主轴和进给轴模式

如果引导主轴在定义耦合前处于进给轴模式中，在启用耦合后，以下机床数据中的速度极限值也会生效：

MD32000 $MA_MAX_AX_VELO（最大轴速度）

为避免出现此特性，必须在定义耦合前将进给轴切换到主轴模式中(M3 S...或 M4 S...)。

其它信息

定义同步主轴对

配置的耦合：

在配置耦合时，引导主轴和跟随主轴通过机床数据定义。 在零件程序中无法修改配置的主轴。 但可以通过 COUPDEF 在零件程序中设定其参数（前提条件： 未写保护）。

用户定义的耦合：

通过 COUPDEF 可以在零件程序中重新定义或修改耦合。 如果已有一个耦合运行生效，则在重新定义前必须先通过 COUPDEL 删除该耦合。

定义耦合：COUPDEF

通过以下指令可以完全重新定义一个耦合：

COUPDEF(<跟随主轴>,<引导主轴>,<传动比分子>,<传动比分母>, 程序段切换特性, 耦合方式)

跟随主轴（FS）和引导主轴（LS）

通过跟随主轴（FS）和引导主轴（LS）的轴名称可以明确地定义一个耦合。 在每个 COUPDEF 指令后都必须写入轴名称。 其他的耦合参数都模态生效，只有当需要进行修改时，才重新编程。

示例：

```
COUPDEF(S2,S1)
```

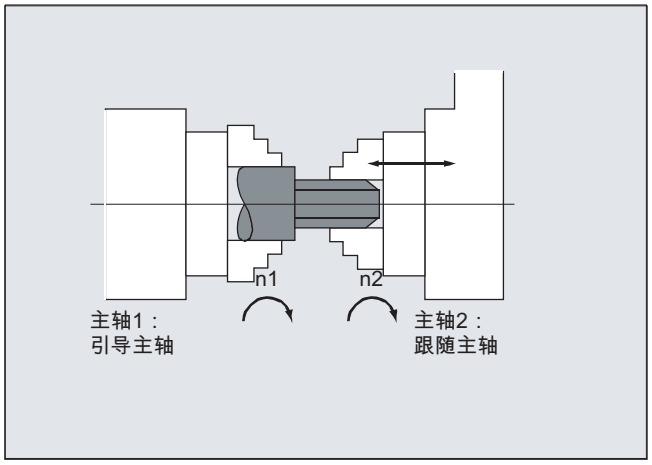
传动比分子/分母

该传动比是跟随主轴（分子）和引导主轴（分母）之间的转速比。 必须写入分子。 如果没有写入分母，则分母为 1.0。

示例：

跟随主轴 S2 和引导主轴 S1，传动比 = 1 / 4 = 0.25。

```
COUPDEF(S2,S1,1.0, 4.0)
```



说明

在启用了耦合并且主轴旋转时也可以修改传动比。

程序段切换特性 NOC，FINE，COARSE，IPOSTOP

在编程程序段切换特性时，可以采用简化的写入方式：

- "NO": 立即（缺省设置）
- "FI": 达到“精同步”

- "CO": 达到“粗同步”
- "IP":达到 IPOSTOP，即在设定值同步后

耦合方式 DV，AV

⚠小心
仅当耦合解除时才能更改耦合方式！

激活同步运行 COUPON，POSFS

- 激活采用任意角度基准的引导主轴和跟随主轴之间的耦合
  - COUPON(S2,S1)
  - COUPON(S2,S1,<跟随主轴位置>)
  - COUPON(S2)
- 激活带角度偏移<跟随主轴位置>的耦合

  用于成型工件中位置同步的耦合
  <跟随主轴位置>以正方向旋转的引导主轴的 0°位置为基准
  取值范围 <跟随主轴位置>:0°… 359.999°
  - COUPON(S2,S1,30)

  即使耦合已经激活，也可以用这种方式改变角度偏移。

跟随主轴的定位

同步主轴耦合激活时，跟随主轴也可以在 ±180° 的范围内定位，而不受引导主轴触发的运行的影响。

定位 SPOS

可以用 SPOS=... 来插补跟随主轴。

示例：

```
SPOS[2]=IC(-90)
```

SPOS 的更多相关信息请参见：
文献：
编程手册 基本原理

转速差 M3 S... 或者 M4 S...

转速差由两个带正负号的转速源叠加产生；在转速控制运行中，当同步主轴耦合生效时，应重新为跟随主轴编程此值，例如通过 S<n>=... 或 M<n>=3, M<n>=4。 同时，通过引导主轴的耦合系数推断得出该转速分量，并按照正确的符号添加到跟随主轴上。

说明

除了旋转方向 M3 或 M4，还必须重新编程转速 S...，否则会发出报警，提示缺少编程。

转速差的更多相关信息请参见：

文献：

功能手册 扩展功能；同步主轴 (S3)

COUPONC 时的转速差

接受一个用于转速差的运动

通过 COUPONC 激活同步主轴耦合后，跟随主轴当前生效的转速（M3 S... 或 M4 S...）被叠加。

说明

使能叠加

只有在使能了叠加时，通过同步主轴耦合 COUPONC 进行的主轴转速叠加（M3 S... 或 M4 S...）才会生效。

引导主轴的动态性能限制

引导主轴的动态性能应限制在一定范围内，从而避免在跟随主轴叠加时，超出跟随主轴的动态性能极限。

速度，加速度：FA, ACC, OVRA, VELOLIMA

可以采用以下指令来编写跟随主轴的轴向速度和加速度：

- FA[SPI(S<n>)] 或 FA[S<n>]（轴向速度）
- ACC[SPI(S<n>)] 或 ACC[S<n>]（轴向加速度）
- OVRA[SPI(S<n>)] 或 OVRA[S<n>]（轴向倍率）
- VELOLIMA[SPI(S<n>)] 或 VELOLIMA[S<n>]（轴向速度提升或降低）

其中，<n> = 1, 2, 3, ...（跟随主轴的编号）

文献：
编程手册 基本原理

说明

加速度分量 JERKLIMA[S<n>]

目前，在主轴上速度递增或递减的编程还没有生效。

轴动态性能配置的更多相关信息请参见：

文献：
功能手册 扩展功能；回转轴（R2）

可编程的程序段切换特性 WAITC

通过 WAITC 可以借助不同的同步运行条件，如粗同步、精同步、IPOSTOP 来规定程序段切换的特性，如：在修改了耦合参数或定位过程后。 如果没有规定同步运行条件，则 COUPDEF 定义时指定的程序段切换特性生效。

示例：

等待达到 COUPDEF 中指定的同步运行条件

```
WAITC( )
```

等待达到跟随主轴 S2 上的同步运行条件 FINE，和跟随主轴 S4 上的 COARSE：

```
WAITC(S2,"FINE",S4,"COARSE")
```

解除耦合 COUPOF

通过 COUPOF 可以设定耦合的解耦特性：

- 解除耦合，并立即切换程序段：
  - COUPOF(S2,S1)（指定引导主轴）
  - COUPOF(S2)（未指定引导主轴）
- 在越过解耦位置后解除耦合。越过解耦位置后切换程序段。
  - COUPOF(S2, S1, 150)（跟随主轴解耦位置：150°）
  - COUPOF(S2,S1,150,30)（跟随主轴解耦位置：150°，引导主轴解耦位置：30°）

解除耦合并停止跟随主轴 COUPOFS

通过 COUPOFS 可将耦合的解耦特性设置为跟随主轴停止：

- 解除耦合，并停止跟随主轴，立即切换程序段：
  - COUPOFS(S2,S1)（指定引导主轴）
  - COUPOFS(S2)（未指定引导主轴）
- 越过解耦位置后解除耦合，并停止跟随主轴。越过解耦位置后切换程序段。
  - COUPOFS(S2,S1,150)（跟随主轴解耦位置：150°）

删除耦合 COUPDEL

通过 COUPDEL 可以删除耦合：

- COUPDEL(S2,S1)（指定引导主轴）
- COUPDEL(S2)（未指定引导主轴）

复位耦合参数 COUPRES

通过 COUPRES 可以激活机床数据和设定数据中设置的耦合值：

- COUPRES(S2,S1)（指定引导主轴）
- COUPRES(S2)（未指定引导主轴）

系统变量

跟随主轴当前的耦合状态

通过以下系统变量可以读取跟随主轴当前的耦合状态：

$AA_COUP_ACT[<跟随主轴>]

值	含义
0	无耦合有效
4	同步主轴耦合有效
提示： 系统变量的其他值都是针对进给轴运行 **文献：** 参数手册 系统变量	

当前角度偏移

通过以下系统变量可以读取跟随主轴相对于引导主轴的当前角度偏移：

- $AA_COUP_OFFS[<跟随主轴>]（设定值角度偏移）
- $VA_COUP_OFFS[<跟随主轴>]（实际值角度偏移）

说明

撤销控制器使能后，对于已经激活的耦合和跟踪运行，在控制器使能重新分配后会设置另一个位置偏移作为初始的编程值。 此时可以读取更改过的位置偏移，必要时可在 NC 零件程序中进行修改。

9.6 主/从组合 (MASLDEF, MASLDEL, MASLON, MASLOF, MASLOFS)

功能

6.4 以下软件版本的主/从偶合仅允许在参与轴处于停止状态中将从动轴偶合到其引导轴上。

6.5 版本软件的扩展功能则允许偶合和分离**正在旋转的**、受到转速控制的主轴，并且允许动态设计。

句法

`MASLON(Slv1,Slv2,..., )`	
`MASLOF(Slv1,Slv2,..., )`	
`MASLDEF（Slv1,Slv2,..., 主主轴）`	用于动态设计的扩展功能
`MASLDEL(Slv1,Slv2,..., )`	用于动态设计的扩展功能
`MASLOFS(Slv1, Slv2, ..., )`	从动轴扩展

说明

当 MASLOF/MASLOFS 时隐式进给停止会消失。 受缺少进刀停止的限制，用于从动轴的 $P-系统变量不提供更新值，直至重新编程为止。

含义

概述

MASLON	启动一个临时偶合
MASLOF	断开一个已激活的偶合。 如果是主轴应注意扩展功能。

动态设计扩展功能

MASLDEF	以用户自定义方式通过机床数据或者也可从零件程序中添加/修改偶合。
MASLOFS	以与 MASLOF 相似的方式断开偶合并且自动将从动轴制动。
MASLDEL	分离主/从轴连接，删除连接定义。
Slv1, Slv2, ...	受某个引导轴引导的从动轴。
引导轴	对某个主/从组合中所定义的从动轴进行引导的轴。

示例

示例 1： 动态设计某个主/从偶合的举例

动态设计一个主/从耦合，由零件程序出发：

在轴容器旋转以后，重要的轴应该成为引导轴。

程序代码	注释
MASLDEF(AUX,S3)	; S3 主动，用于 AUX
MASLON(AUX)	; 耦合开，用于 AUX
M3=3 S3=4000	; 右旋转方向
MASLDEL(AUX)	; 删除设计并脱开耦合
AXCTSWE(CT1)	; 容器旋转

示例

示例 2： 某个从动轴的实际值偶合

通过 PRESETON 将某个从动轴的实际值偶合设置成引导轴的相同值。

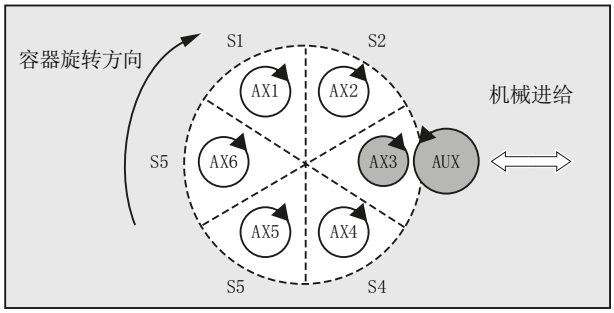
如果是某个永久性的主/从偶合，应当在从动轴上通过 PRESETON 来改变实际值。

程序代码	注释
N37262 $MA_MS_COUPLING_ALWAYS_ACTIVE[AX2]=0	; 持久的耦合短时间切断。
N37263 NEWCONF	
N37264 STOPRE	
MASLOF(Y1)	; 临时耦合关。
N5 PRESETON(Y1,0,Z1,0,B1,0,C1,0,U1,0)	; 给尚未经过找零的从动轴设定实际值，因为这些轴已使用上电激活了。
N37262 $MA_MS_COUPLING_ALWAYS_ACTIVE[AX2]=1	; 激活持久的耦合。
N37263 NEWCONF	

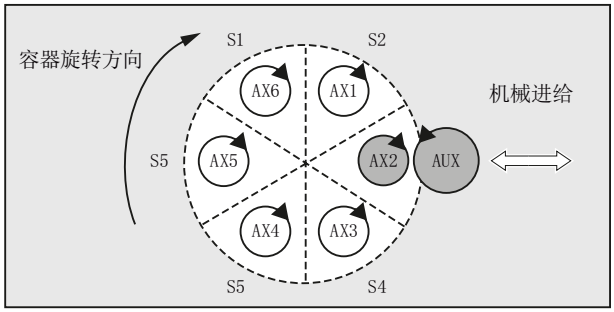
示例 3： 偶合顺序 位置 3/容器 CT1

为了能在容器旋转之后使用另一个主轴来闭合偶合，原有的偶合必须实现断开、删除设计并且设计新的偶合。

初始情况：



在以某个槽旋转之后：



文献：

功能手册 扩展功能；多个操作面板和 NCU(B3)，章节 “轴容器”

其它信息

概述

MASLOF	主轴在转速控制运行时，直接执行该指令。 此时旋转的从动主轴维持其转速，直至重新编程转速。

动态设计扩展功能

MASLDEF	从零件程序出发定义某个主/从组合。之前只通过机床数据定义。
MASLDEL	该指令取消了对主主轴分配从动轴，同时脱开耦合，类似于 MASLOF。 保留机床数据中约定的主/从定义。
MASLOFS	MASLOFS 可以用来在断开偶合时自动使从动轴停止。 如果是定位运行方式中的轴和主轴，仅在停止状态中闭合和断开偶合。

说明

对于从动轴而言，可通过 PRESETON 使实际值同步到与引导轴的值相同。为此必须瞬间断开持久的主/从偶合，以便在上电时将尚未找零的从动轴的实际值设定成引导轴的值。然后该持续的耦合再次恢复。

通过 MD 设置 MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE = 1 激活持久的主/从耦合，这对于临时耦合的语言命令没有效用。

主轴耦合特性

如果主轴在转速控制的运行方式中，则 MASLON, MASLOF, MASLOFS 和 MASLDEL 的耦合特性通过机床数据 MD 37263:

在 MD37263 = 0 的默认设置中，仅在参与轴的停止状态中进行从动轴的耦合和脱离。MASLOFS 相当于 MASLOF。

当 MD37263 = 1 时，就会直接执行偶合指令并且也在运动中执行。偶合会在 MASLON 时立即闭合并且当 MASLOFS 或者 MASLOF 立即脱离。 此时正在转动的从动轴会在 MASLOFS 时自动制动，并且当 MASLOF 时将其转速一直保持到重新编程转速时为止。

运动同步动作 10

10.1 基础部分

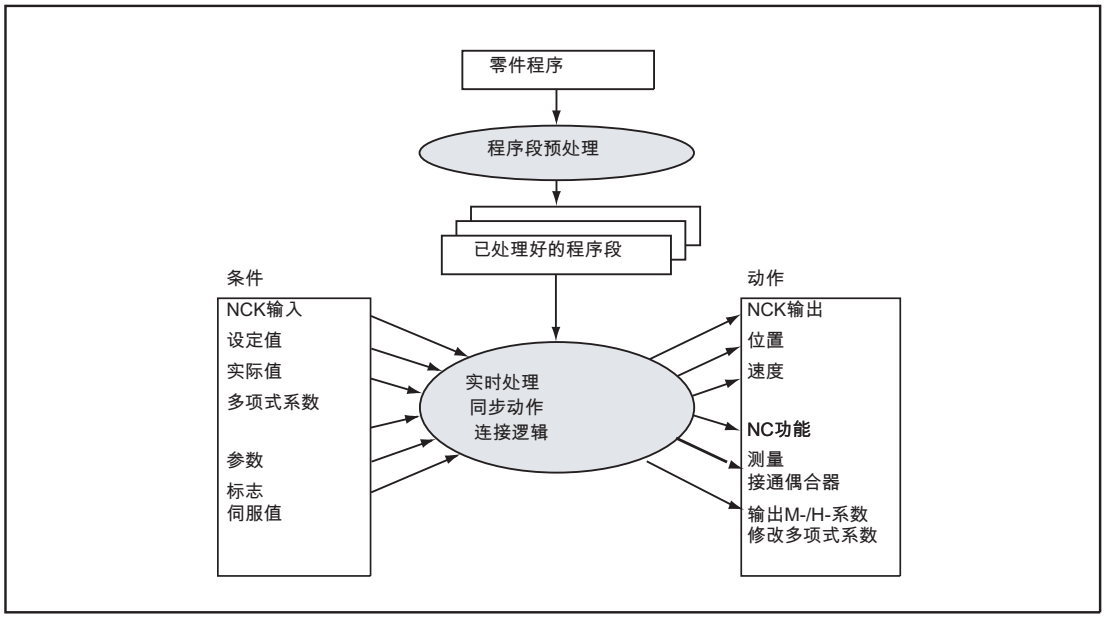
功能

同步动作提供可以同步执行处理程序段的可能性。

动作的执行时间可以通过各个条件定义。 这些条件在插补节拍中得以监控。 这些动作是对实时事件的反应；执行并不是在程序段交接处进行。

此外，同步动作还包含对其使用寿命的说明和对编程主运行变量的询问频率，以及对启动动作的执行频率说明。 由此，一个动作可以一次或者也可以循环（插补节拍）方式进行触发。

可能的应用



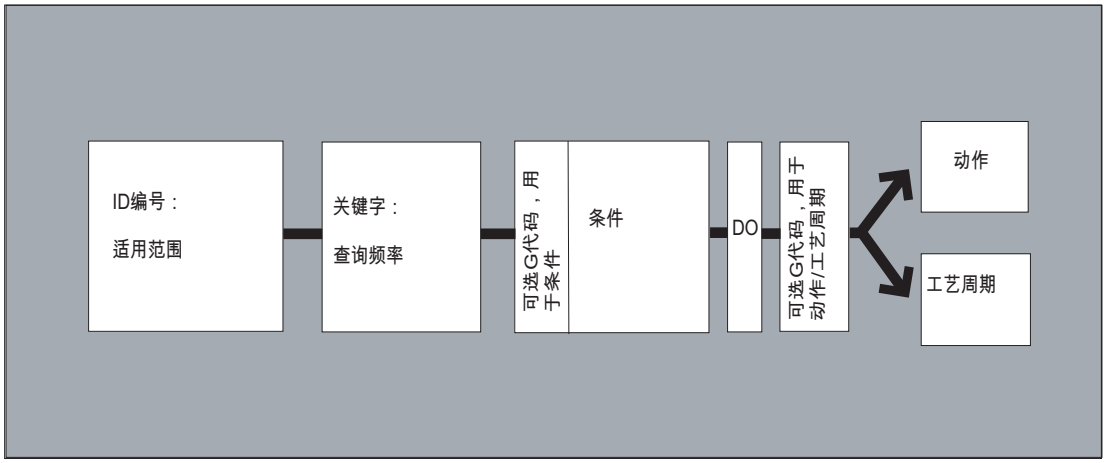
- 对运行时间紧张的应用进行优化（例如换刀）
- 对外部事件的快速反应
- 编程 AC 调节

- 调节安全功能
- ....

编程

一个同步动作在程序段中是单独的，并且从机床功能的下一个可执行程序段起生效（例如带有 G0, G1, G2, G3 的横切运动）。

同步动作由多达 5 个具有不同任务的指令单元组成：



句法：

```
DO <动作 1> <动作 2> …
<关键字> <条件> DO <动作 1> <动作 2> …
ID=<n> <关键字> <条件> DO <动作 1> <动作 2> …
IDS=<n> <关键字> <条件> DO <动作 1> <动作 2> …
```

含义：

DO	触发编程动作的指令 仅在满足 <条件> 时有效（如果已编程）。 → 参见“ 动作”
<动作 1> <动作 2> …	要启动的动作 示例： • 分配变量 • 启动工艺循环

<关键字>	通过关键字（WHEN, WHENEVER, FROM 或者 EVERY），定义一个同步动作 <条件> 的循环检查。 → 参见“ 条件的循环检查 ”
<条件>	主运行变量的链接逻辑 条件在 IPO 节拍中检查。
ID=<n> 或 IDS=<n>	识别号 通过识别号确定加工顺序中的适用范围和位置。 → 参见“ 有效范围和加工顺序 ”

同步动作/工艺周期的协调

提供下列命令用于协调同步动作/工艺周期：

指令	含义
CANCEL(<n>)	删除同步动作 → 参见“删除同步动作”
LOCK(<n>)	阻止同步动作
UNLOCK(<n>)	释放同步动作
RESET	复位工艺循环 有关 LOCK, UNLOCK 和 RESET: → 参见“ 锁止，释放，中断 ”

示例

程序代码	注释
WHEN $AA_IW[Q1]>5 DO M172 H510	; 如果轴 Q1 的实际值超过 5 毫米，则辅助功能 M172 和 H510 输出到 PLC 接口。

10.1.1 适用范围和加工顺序（ID, IDS）

功能

适用范围

同步动作适用范围通过标识 ID 或 IDS 确定：

没有模态 ID：	自动运行方式中的逐段有效同步动作
ID:	程序结束时在自动运行方式中模态有效的同步动作
IDS:	静态同步动作，在各个工作方式中模态有效，也通过程序结束

应用

- 在 JOG 方式下的 AC 循环
- 用于安全集成的连接逻辑
- 监控功能，对所有运行方式中机床状态的反应

加工顺序

模态和静态有效的同步动作以插补节拍中 ID- 或 IDS-编号 (ID=<n> 或 IDS=<n>) 的顺序加工。

程序段方式有效的同步动作（没有 ID 号）在加工模态有效的同步动作结束之后，按照编程的顺序进行处理。

说明

通过机床数据设置可以保护模态有效的同步动作不会被改变或删除（→ 机床制造商！）。

编程

句法	含义
没有 模态-ID	同步动作仅在**自动运行方式**中有效。 仅适用于下列可执行的程序段（带有运动指令或者其它机床动作的程序段），为**逐段** 有效。 示例： `WHEN $A_IN[3]==TRUE DO $A_OUTA[4]=10`
ID=<n> ...	同步动作在下列 **模态** 程序段中有效且可通过 CANCEL(<n>) 关闭或者通过编程一个带有相同 ID 的新同步动作来覆盖。 M30 程序段中有效的同步动作延迟程序结束。 ID-同步动作仅在**自动运行方式**中有效。 值范围 **<n>**： 1 ... 255

示例：

```
ID=2 EVERY $A_IN[1]==1 DO POS[X]=0
```

IDS=<n>

静态同步动作在**所有工作方式中模态** 有效。它们也可通过程序结束保持有效并能够直接在上电后用一个 ASUP 激活。 因此可以激活动作，它们与 NC 中所选择的运行方式无关，直接运行。

值范围 **<n>**： 1 ... 255

示例：

```
IDS=1 EVERY $A_IN[1]==1 DO POS[X]=100
```

10.1.2 条件循环检查（WHEN, WHENEVER, FROM, EVERY）

功能

通过一个关键字定义一个同步动作的条件循环检查。 如果未编程关键字，则在每个 IPO 节拍中执行同步动作的动作。

关键字

没有关键字	动作执行不受条件制约。 在每个插补节拍循环执行动作。
WHEN	在每个插补节拍中对条件进行查询，直到该条件被满足时为止；然后将相应的动作准确执行一次。
WHENEVER	在每个插补节拍中对条件进行循环检查。 只要条件被满足，就在每个插补节拍中执行相应的动作。
FROM	在每个插补节拍中对条件进行检查，直到条件满足时为止。然后就执行动作，同步动作激活时间有多久，该动作就会执行多久，也就是说，即使条件不再满足时，也会执行继续执行该动作。
EVERY	在每个插补节拍中对条件进行查询。 只有当条件满足后，才执行一次动作。 脉冲沿控制： 当条件从状态 FALSE 变成 TRUE 时，就会再次执行动作。

主运行变量

在插补节拍(IPO-节拍)中分析所使用的变量。 同步动作中的主运行变量不触犯进给停止。

分析：

如果在某个零件程序中出现主运行变量（例如实际值，某个数字输入或者输出端的位置等等），就会停止进给运动，直到上一个程序段执行完毕并且主运行变量的值存在时为止。

示例

示例 1： 没有关键字

程序代码	注释
DO $A_OUTA[1]=$AA_IN[X]	; 发送实际值到模拟输出端。

示例 2： WHENEVER

程序代码	注释
WHENEVER $AA_IM[X] > 10.5*SIN(45) DO …	; 和预先算出的表达式进行比较。
WHENEVER $AA_IM[X] > $AA_IM[X1] DO …	; 和其它主运行变量进行比较。
WHENEVER ($A_IN[1]==1) OR ($A_IN[3]==0) DO ...	; 两个相互关联的比较。

示例 3： EVERY

程序代码	注释
ID=1 EVERY $AA_IM[B]>75 DO POS[U]=IC(10) FA[U]=900	; 当 MKS 中轴 B 的实际值总是超过值 75 时，U 轴应以最大轴向进给量 10 继续定位。

其它信息

条件

条件表示一个可以由布尔运算符任意建立的逻辑表达式。 布尔表达式必须总是在括号中加以说明。

在插补节拍中检查该条件。

在条件前可以用一个 G 代码说明。 这样就能做到为分析条件和需要执行的动作/工艺循环而定义的设置与正处于激活状态的零件程序无关。 要求同步动作去除与程序外围的耦合，因为在任意时间根据所满足的释放条件，在定义输出状态执行同步动作。

应用情况

通过 G 代码 G70，G71，G700，G710 来确定条件分析和动作的测量单位制。

条件的某个规定 G 代码适用于条件分析，并且当动作没有规定 G 代码时，也适用于动作。

每个条件只可编程 G 代码组的**一个 G 代码** 。

可能有的条件

- 比较主运行变量（模拟/数字输入/输出，以及其它）
- 比较结果之间的布尔关系
- 计算实时表达式
- 时间/距离程序段开始
- 距离程序段结束
- 测量值，测量结果
- 伺服值
- 速度，轴状态

10.1.3 动作 (DO)

功能

在同步动作中可以编程一个或者多个动作。 所有在一个程序段中编程的动作以相同的插补节拍激活。

句法

```
DO <动作 1> <动作 2> …
```

含义

DO	当满足条件时执行一个动作或者工艺循环。
<动作>	当满足条件时，给已开始的动作（例如变量）赋值，接通轴偶合，设定 NCK 输出，输出 M, S 和 H 功能，规定已编程的 G 代码，...

G 代码 可在动作/工艺循环的同步动作中编程。 有时，在程序段中和工艺循环中所有的动作给定一个另外的 G 代码，与在条件中所设置的不同。 如果工艺循环在动作部分中，则 G 代码在结束工艺循环后也适用于随后的动作直至下一个 G 代码模态继续。

每个动作部分仅允许编程 G 代码组的 **一个 G 代码**（G70， G71， G700， G710）。

示例： 带有两个动作的同步动作

程序代码	注释
WHEN $AA_IM[Y] >= 35.7 DO M135 $AC_PARAM=50	; 如果条件已满足，就会将 M135 发送给 PLC 并且将倍率设定为 50%。

10.2 条件和动作的运算符

比较 (==, <>, <, >, <=, >=)	在条件中可以比较变量或者部分表达式。 结果始终为数据类型 BOOL。 允许所有已知的比较运算符。
布尔运算符 (NOT, AND, OR, XOR)	可以使用布尔运算符将变量、常量或者比较相互联系起来。
逐位运算符 (B_NOT, B_AND, B_OR, B_XOR)	可以使用的逐位运算符 B_NOT, B_AND, B_OR, B_XOR.
基本计算类型 (+, -, *, /, DIV, MOD)	主运行变量可以通过基本计算类型相互连接或者与常量连接。
数学函数 (SIN, COS, TAN, ASIN, ACOS, ABS, TRUNC, ROUND, LN, EXP, ATAN2, POT, SQRT, CTAB, CTABINV).	在数据类型为 REAL 的变量上可以使用数学函数。
索引	可以用主运行表达式进行定位。

示例

- **关联基本计算类型**

适用四则运算，允许表达式有括号。 也允许将运算符用于数据类型 REAL

编程	注释
DO $AC_PARAM[3] = $A_INA[1]-$AA_IM[Z1]	; ;减法，两个 ; 主运行变量
WHENEVER $AA_IM[x2] < $AA_IM[x1]-1.9 DO $A_OUT[5] = 1	; ; 从变量中减去一个常量
DO $AC_PARAM[3] = $INA[1]-4*SIN(45.7 $P_EP[Y])*R4	; 常量表达式，在进刀时计算

- **数学函数**

编程	注释
DO $AC_PARAM[3] = COS($AC_PARAM[1])	; ;

- **实时表达式**

编程	注释
ID=1 WHENEVER ($AA_IM[Y]>30) AND ($AA_IM[Y]<40) DO $AA_OVR[S1]=80	; 选择一个位置范围

编程	注释
ID=67 DO $A_OUT[1]=$A_IN[2] XOR $AN_MARKER[1]	; 分析 2 个布尔信号
ID=89 DO $A_OUT[4]=$A_IN[1] OR ($AA_IM[Y]>10)	; 发送某个比较结果

- **定位主运行变量**

编程	注释
WHEN…DO $AC_PARAM[$AC_MARKER[1]] = 3	;
不允许的是	;
$AC_PARAM[1] = $P_EP[$AC_MARKER]	;

10.3 同步动作的主运行变量

10.3.1 系统变量

功能

借助系统变量可以读取和写入 NC 数据。 系统变量在进给变量和主运行变量中是有区别的。 进给变量总是在进给时刻执行。 主运行变量总是根据当前主运行状态来计算其值。

名称

系统变量名称大多都是以 $ 字符开始：

进给变量：

$M...	机床数据
$S...	设定数据，保护区
$T...	刀具管理参数
$P...	编程的值，进给数据
$C...	ISO 包络循环的循环变量
$O...	选项数据
R ...	R 参数

主运行变量：

$$A...	当前主运行数据
$$V...	伺服数据
$R...	R 参数

第 2 个字母说明变量的访问方法：

N...	NCK 全局值（一般有效的值）
C...	通道专用的值
A...	轴专用的值

第 2 个字母一般仅用于主运行变量。 进给变量，如 $P_，一般在没有 2 个字母的情况下执行。

前缀（$ 后面跟着一个或两个字母）后面总是跟着一个下划线和后缀变量名称（一般都作为英文标记或缩写）。

数据类型

主运行变量不能有下列数据类型：

INT	整数值的整数，带符号
REAL	有理数的实数
BOOL	布尔 TRUE 和 FALSE
CHAR	ASCII 字符
字符串	用数字字母字符组成的字符串
AXIS	轴地址和主轴

进给变量还可以有下列数据类型：

FRAME	坐标转换

变量数组

系统变量可以设定为 1 至 3 维。

可以支持以下数据类型： BOOL, CHAR, INT, REAL, STRING, AXIS

索引的数据类型可以是类型 INT 和 AXIS，在此可以对其任意分类。

字符串变量只能设定为 2 维的。

数组定义举例：

```
DEF BOOL $AA_NEWVAR[x,y,2]
DEF CHAR $AC_NEWVAR[2,2,2]
```

```
DEF INT $AC_NEWVAR[2,10,3]
DEF REAL $AA_VECTOR[x,y,z]
DEF STRING $AC_NEWSTRING[3,3]
DEF AXIS $AA_NEWAX[x,3,y]
```

说明

如果有针对系统变量的一个 BTSS 变量，显示 3 维系统变量可以不受限制。

10.3.2 隐式类型转换

功能

在赋值和参数传输时可以给变量分配或传输不同的数据类型。

隐式类型转换触发值的内部类型转换。

可能的类型转换

遵循	REAL	整数	BOOL	CHAR	字符串	AXIS	FRAME
从							
REAL	是	是*	是 1)	–	–	–	–
整数	是	是	是 1)	–	–	–	–
BOOL	是	是	是	–	–	–	–

说明

* 从实数型到整数型的转换中，小数值>= 0.5 时向上凑整，否则舍去（ROUND 功能）。
值超过时会触发报警。

1) 值<> 0 对应于 TRUE，值== 0 对应于 FALSE

结果

```
 REAL 或 INTEGER 类型转换为 BOOL
结果 BOOL = TRUE                如果 REAL 或 INTEGER 的值不等于 0
```

```
结果 BOOL = FALSE                如果 REAL 或 INTEGER 的值等于 0
BOOL 类型转换为 REAL 或 INTEGER
结果 REAL TRUE                   如果 BOOL 的值= TRUE (1)
结果 INTEGER = TRUE              如果 BOOL 的值= TRUE （1）
BOOL 类型转换为 REAL 或 INTEGER
结果 REAL FALSE）                如果 BOOL 的值= FALSE （0）
结果 INTEGER = FALSE             如果 BOOL 的值= FALSE （0）
```

隐式类型转换举例

```
INTEGER 类型转换为 BOOL
$AC_MARKER[1]=561
ID=1 WHEN $A_IN[1] == TRUE DO $A_OUT[0]=$AC_MARKER[1]

REAL 类型转换为 BOOL
R401 = 100.542
WHEN $A_IN[0] == TRUE DO $A_OUT[2]=$R401

BOOL 类型转换为 INTEGER
ID=1 WHEN $A_IN[2] == TRUE DO $AC_MARKER[4] = $A_OUT[1]]

BOOL 类型转换为 REAL
R401 = 100.542
WHEN $A_IN[3] == TRUE DO $R10 = $A_OUT[3]
```

10.3.3 GUD 变量值

同步动作允许的 GUD 变量

除了特定的系统变量，在同步动作中还可以使用一些预定义的全局同步用户变量（同步 GUD）。 通过以下机床数据可以定义不同数据类型、不同访问等级下用户可以使用的同步动作 GUD 数量：

- MD18660 $MM_NUM_SYNACT_GUD_**REAL**[<x>] = <数量>
- MD18661 $MM_NUM_SYNACT_GUD_**INT**[<x>] = <数量>
- MD18662 $MM_NUM_SYNACT_GUD_**BOOL**[<x>] = <数量>
- MD18663 $MM_NUM_SYNACT_GUD_**AXIS**[<x>] = <数量>

- MD18664 $MM_NUM_SYNACT_GUD_**CHAR**[<x>] = <数量>
- MD18665 $MM_NUM_SYNACT_GUD_**STRING**[<x>] = <数量>

索引<x>下可以指定模块（访问权限）；<数量> 下可以指定相应数据类型 (REAL, INT, ...) 的同步动作 GUD 数量。 在各个模块中会继续为每个数据类型建立一个 1 维数组变量，变量名称和数据类型的对应关系如下： SYG_<数据类型><存取权限>[<索引>]:

索引 <x>		数据类型 (MD18660 ... MD18665)					
	模块	REAL	INT	BOOL	AXIS	CHAR	STRING
0	SGUD	SYG_RS[i]	SYG_IS[i]	SYG_BS[i]	SYG_AS[i]	SYG_CS[i]	SYG_SS[i]
1	MGUD	SYG_RM[i]	SYG_IM[i]	SYG_BM[i]	SYG_AM[i]	SYG_CM[i]	SYG_SM[i]
2	UGUD	SYG_RU[i]	SYG_IU[i]	SYG_BU[i]	SYG_AU[i]	SYG_CU[i]	SYG_SU[i]
3	GUD4	SYG_R4[i]	SYG_I4[i]	SYG_B4[i]	SYG_A4[i]	SYG_C4[i]	SYG_S4[i]
4	GUD5	SYG_R5[i]	SYG_I5[i]	SYG_B5[i]	SYG_A5[i]	SYG_C5[i]	SYG_S5[i]
5	GUD6	SYG_R6[i]	SYG_I6[i]	SYG_B6[i]	SYG_A6[i]	SYG_C6[i]	SYG_S6[i]
6	GUD7	SYG_R7[i]	SYG_I7[i]	SYG_B7[i]	SYG_A7[i]	SYG_C7[i]	SYG_S7[i]
7	GUD8	SYG_R8[i]	SYG_I8[i]	SYG_B8[i]	SYG_A8[i]	SYG_C8[i]	SYG_S8[i]
8	GUD9	SYG_R9[i]	SYG_I9[i]	SYG_B9[i]	SYG_A9[i]	SYG_C9[i]	SYG_S9[i]
其中 i = 0 到(<数量> - 1) 模块： _N_DEF_DIR/_N_ ... _DEF,例如，用于 SGUD ⇒ _N_DEF_DIR/_N_**SGUD**_DEF							

属性

同步动作 GUD 具有以下属性：

- 同步动作 GUD 可以在同步运动和零件程序/循环中读写
- 同步动作 GUD 可以通过操作界面接口访问
- 同步动作 GUD 会显示在 HMI 操作界面的操作区“参数”下
- 同步动作 GUD 可以在 HMI 的向导程序中、变量视图和变量日志中使用

- 同步动作 GUD 中，字符串类型的数组长度固定为 32 个字符，即 31 个字符+ \0。
- 即使没有手动建立全局用户数据(GUD)的定义文件，也可以通过机床数据中定义的同步动作 GUD，在 HMI 上从相应的 GUD 模块读取。

注意
只有当没有同步 GUD 定义了(MD18660 - MD18665)相同的名称时，用户变量(GUD, PUD, LUD)才可以和同步动作 GUD 有相同的名称(DEF ... SYG_xy)。 在同步动作中**不能**使用这些由用户定义的 GUD。

存取权限

GUD 定义文件中写入的存储权限继续生效，但它只针对该 GUD 定义文件中写入的 GUD 变量。

删除属性

如果重新激活某个指定 GUD 定义文件的内容，则首先删除主动文件系统中旧的 GUD 数据模块。 同样，系统定义的同步动作 GUD 也被复位。 该过程也可以通过 HMI 在操作界面“服务”“定义和激活用户数据（GUD）”中进行。

10.3.4 缺省轴标识符（NO_AXIS）

功能

未用某个值初始化的 AXIS 类型变量或参数可以用定义的缺省轴标识符标明。 用该缺省值初始化未定义的轴变量。

初始化无效的轴名称通过询问一个同步动作中的“NO_AXIS”变量来识别。 这一未初始化的轴标识符通过机床数据设计的缺省轴标识符分配。

机床制造商

必须通过机床数据至少定义和预占用一个有效的现有轴标识符。 也可以预占用所有现有的有效轴标识符。 请注意机床制造商说明。

说明

现在，新设立的变量在定义时自动分配给机床数据中保存的缺省轴名称的值。

有关机床数据适用定义的其他信息参见:

文献：

/FBSY/ 功能手册 同步动作

句法

```
PROC UP(AXIS PAR1=NO_AXIS, AXIS PAR2=NO_AXIS)
IF PAR1 <>NO_AXIS...
```

含义

PROC	子程序定义
UP	要识别的子程序名称
PARn	参数 n
NO_AXIS	用缺省轴标识符初始化形式参数

示例： 定义主程序中一个轴变量

程序代码

```
DEF AXIS AXVAR
UP( , AXVAR)
```

10.3.5 同步动作标记（$AC_MARKER[n]）

功能

可以在同步动作中读取、写入数组变量 $AC_MARKER[n]。 这些变量可能存储在主动或被动文件系统的存储器中。

同步动作变量： 数据类型 INT

$AC_MARKER[n]	通道专用的标记/数据类型 INTEGER 的计数器
$MC_MM_NUM_AC_MARKER	机床数据，用于设置运动同步动作的通道专用标记个数
n	变量的数组索引 0-n

读取和写入标记变量举例

程序代码

```
WHEN ... DO $AC_MARKER[0] = 2
WHEN ... DO $AC_MARKER[0] = 3
WHENEVER $AC_MARKER[0] == 3 DO $AC_OVR=50
```

10.3.6 同步动作参数（$AC_PARAM[n]）

功能

同步动作参数 $AC_PARAM[n] 用于进行计算，并且作为同步动作中的缓存。 这些变量可能存储在主动或被动文件系统的存储器中。

同步动作变量： 数据类型 REAL

每个通道在相同的名称下参数只可以出现一次。

$AC_PARAM[n]	运动同步动作的计算变量（REAL）
$MC_MM_NUM_AC_PARAM	机床数据，用于将运动同步动作的参数个数设置为最大 20000。
n	参数数组索引 0n

同步动作参数 $AC_PARAM[n] 举例

程序代码

```
$AC_PARAM[0]=1.5
$AC_MARKER[0]=1
ID=1 WHEN $AA_IW[X]>100 DO $AC_PARAM[1]=$AA_IW[X]
ID=2 WHEN $AA_IW[X]>100 DO $AC_MARKER[1]=$AC_MARKER[2]
```

10.3.7 计算参数（$R[n]）

功能

该静态数组变量用于在零件程序和同步动作中进行计算。

句法

在零件程序中编程：

```
REAL R[n]
REAL Rn
```

在同步动作中编程：

```
REAL $R[n]
REAL $Rn
```

计算参数

使用计算参数可以：

- 存储数值，它们在程序结束、NC 复位和上电时保持不变。
- 在 R 参数图中显示所存储的值。

示例

程序代码	注释
WHEN $AA_IM[X]>=40.5 DO $R10=$AA_MM[Y]	; 在同步动作中使用 R10
G01 X500 Y70 F1000	
STOPRE	; 进给停止
IF R10>20	; 分析计算变量。

程序代码
SYG_AS[2]=X
SYG_IS[1]=1
WHEN $AA_IM[SGY_AS[2]]>10 DO $R3=$AA_EG_DENOM[SYG_AS[1]], SYG_AS[2]]
WHEN $AA_IM[SGY_AS[2]]>12 DO $AA_SCTRACE[SYG_AS[2]]=1
SYG_AS[1]=X
SYG_IS[0]=1
WHEN $AA_IM[SGY_AS[1]]>10 DO $R3=$$MA_POSCTRL_GAIN[SYG_IS[0]],SYG_AS[1]]
WHEN $AA_IM[SGY_AS[1]]>10 DO $R3=$$MA_POSCTRL_GAIN[SYG_AS[1]]
WHEN $AA_IM[SGY_AS[1]]>15 DO $$MA_POSCTRL_GAIN[SYG_AS[0]], SYG_AS[1]]=$R3

10.3.8 读取和写入 NC 机床数据和 NC 设定数据

功能

也可从同步动作中读取和写入 NC 机床数据/设定数据。 编程在读取和写入机床数据数组元时可以省略一个索引。 如果这一过程在零件程序中进行，则在读取 **第一个** 数组元时读取值并在写入所有数组元时写入值。

在这种情况下，在同步动作中读取或写入**第一个** 元素。

确定

MD，SD 带

$：读取同步动作编译时间的值

$$：读取主运行中的值

读取进给时间的 MD 和 SD 值

使用 $ 符号从同步动作导入这些数据后对其寻址并且在进给时进行分析。

```
ID=2 WHENEVER $AA_IM[z]<$SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0
;为摆动而假设的回转范围 2 在这里响应，不可以修改
```

读取主运行时间的 MD 和 SD 值

使用 $$ 符号从同步动作导入这些数据后对其寻址并且在主运行时进行分析。

```
ID=1 WHENEVER $AA_IM[z]<$$SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0
;这里的出发点是：可以通过加工过程中的操作来改变回转位置。
```

写入主运行时间的 MD 和 SD 值

当前已设置的访问权限必须允许写入访问。 所有 MD 和 SD 的有效性，见 **文献;** /LIS/，规定列表 （卷 1）。

需要写入的 MD 和 SD 应使用 $$ 导入后进行寻址。

示例

程序代码	注释
ID=1 WHEN $AA_IW[X]>10 DO $$SN_SW_CAM_PLUS_POS_TAB_1[0]=20 $$SN_SW_CAM_MINUS_POS_TAB_1[0]=30	; 改变 SW 凸轮的开关位置。 说明： 开关位置必须在到达位置之前修改 2-3 个插补节拍。

10.3.9 计时变量（$AC_Timer[n]）

功能

系统变量 $AC_TIMER[n] 可以在定义等候时间结束后起动动作。

定时器变量： 数据类型 REAL

$AC_TIMER[n]	数据类型 REAL 的通道专用定时器
秒	单位为秒
n	定时器变量索引

设定定时器

通过赋值来启动定时器变量相加：
$AC_TIMER[n] = 值

n:	时间变量号码
值：	开始值（一般为“0”）

使定时器停止

赋给一个负值就可使定时器变量停止相加：
$AC_TIMER[n]=-1

读取定时器

可以在定时器变量正在计数或者停止时读取当前的时间值。 在通过赋给数值 -1 使定时器变量停止之后，最后的当前值就会停住并且可以继续读取。

示例

通过模拟输出端发送某个实际值在识别某个数字输入之后的 500 ms

程序代码	注释
WHEN $A_IN[1]==1 DO $AC_TIMER[1]=0	; 复位定时器并启动
WHEN $AC_TIMER[1]>=0.5 DO $A_OUTA[3]=$AA_IM[X] $AC_TIMER[1]=-1	

10.3.10 FIFO 变量 ($AC_FIFO1[n] ... $AC_FIFO10[n])

功能

有 10 个 FIFO 变量（循环存储器）可用来保存相关的数据顺序。
数据类型： REAL

应用：

- 循环测量
- 循环加工

可以对每个单元进行读写存取。

FIFO 变量

可使用的 FIFO-变量的数量通过机床数据 MD28260 $MC_NUM_AC_FIFO 确定。

FIFO 变量中可写入值的数目通过机床数据 MD28264 $MC_LEN_AC_FIFO 定义。 所有 FIFO 变量有相同的长度。

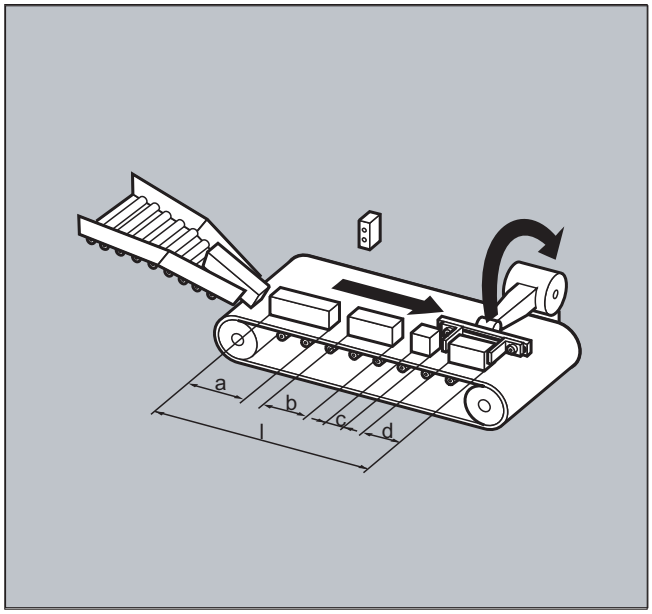
当在 MD28266 $MC_MODE_AC_FIFO 中设定 Bit0 时，仅形成 FIFO 单元的和。

索引 **0** ～ **5** 具有特殊含义：

索引	含义	
0	当写入时：	新值设立在 FIFO 中。
	在读取时：	读最早的单元并从 FIFO 中去除。
1	访问最早保存的单元	
2	访问最新保存的单元	
3	所有 FIFO 单元之和	
4	在 FIFO 中可供使用的单元的数量。 可以对 FIFO 的每个单元进行读写访问。将单元数量复位就可使 FIFO 变量复位，例如用于第一个 FIFO 变量： $AC_FIFO1[4] = 0	
5	相对于 FIFO 开始的当前写索引	
6 至 n_{max}	访问第 n 个 FIFO-单元	

示例： 循环存储器

在生产过程中，使用一个传送带用于传送不同长度(a, b, c, d)的产品。 因此，在输送长度的输送带上，将视相应产品的长度而定，同时输送不同数量的产品。 当输送速度相同时，必须将从输送带上取出产品的动作调整到与产品的可变到达时间相适应。



程序代码	注释
DEF REAL ZWI=2.5	; 已放置产品之间的恒定间距。
DEF REAL GESAMT=270	; 纵向测量位置与取件位置之间的间距。
EVERY $A_IN[1]==1 DO $AC_FIFO1[4]=0	; 在过程开始时将 FIFO 复位。
EVERY $A_IN[2]==1 DO $AC_TIMER[0]=0	; 某个产品中断光栅，开始时间测定。
EVERY $A_IN[2]==0 DO $AC_FIFO1[0]=$AC_TIMER[0]*$AA_VACTM[B]	; 如果光栅没有被挡住，从测得的时间和输送速度中算出产品长度并且保存在 FIFO 中。
EVERY $AC_FIFO1[3]+$AC_FIFO1[4]*ZWI>=GESAMT DO POS[Y]=-30 $R1=$AC_FIFO1[0]	; 只要所有产品长度和间隔长度之和大于/等于放入和取件位置之间的长度，就将取件位置上的产品从输送带上取出，从 FIFO 中读取相应的产品长度。

10.3.11 通过插补器中的程序段类型询问（$AC_BLOCKTYPE, $AC_BLOCKTYPEINFO, $AC_SPLITBLOCK）

功能

下面的系统变量供同步动作使用，从而得到在主运行中当前程序段的信息：

- $AC_BLOCKTYPE
- $AC_BLOCKTYPEINFO
- $AC_SPLITBLOCK

程序块类型变量和程序块类型信息变量

$AC_BLOCKTYPE		$AC_BLOCKTYPEINFO				
值：		值：				
0	不等于 0	T	H	Z	E	含义：
原始程序段	中间程序段					中间程序段触发器：
	1	1	0	0	0	内部生成的程序段，没有其它信息
	2	2	0	0	1	倒棱/倒圆： 直线
	2	2	0	0	2	倒棱/倒圆： 圆弧
	3	3	0	0	1	WAB: 以直线返回
	3	3	0	0	2	WAB: 以四分之一圆弧逼近
	3	3	0	0	3	WAB: 以半圆逼近
						刀具补偿：
	4	4	0	0	1	STOPRE 之后的逼近程序段
	4	4	0	0	2	在找不到交点时的连接程序段
	4	4	0	0	3	内角上的点状圆弧 (仅当 TRACYL)
	4	4	0	0	4	外角处绕行圆弧（或者锥形截面）

$AC_BLOCKTYPE		$AC_BLOCKTYPEINFO				
值：		值：				
0	不等于 0	T	H	Z	E	含义：
原始程序段	中间程序段					中间程序段触发器：
	4	4	0	0	5	去除补偿时的逼近程序段
	4	4	0	0	6	重新激活 WRC 时的逼近程序段
	4	4	0	0	7	由于曲率太大，分解程序段
	4	4	0	0	8	3D 面铣削时补偿程序段（刀具矢量 II 平面矢量）
						平滑：
	5	5	0	0	1	G641
	5	5	0	0	2	G642
	5	5	0	0	3	G643
	5	5	0	0	4	G644
						TLIFT 程序段，包含：
	6	6	0	0	1	切向轴线性运动，无提刀运动
	6	6	0	0	2	切向轴非线性运动（多项式），无提刀运动
	6	6	0	0	3	提刀运动，切向轴运动和提刀运动同时启动
	6	6	0	0	4	提刀运动，到达了特定提刀位置时先启动切向轴。
						位移划分：
	7	7	0	0	1	编程的位移划分有效，没有冲裁或者步冲
	7	7	0	0	2	编程的位移划分，带有效的冲裁或者步冲
	7	7	0	0	3	内部自动生成的位移划分

$AC_BLOCKTYPE		$AC_BLOCKTYPEINFO				
值：		值：				
0	不等于 0	T	H	Z	E	含义：
原始程序段	中间程序段					中间程序段触发器：
						编译循环：
	8	ID 应用				编译循环应用的 ID，它产生该程序段

T: 千位
H: 百位
Z: 十位
E: 个位

说明

$AC_BLOCKTYPEINFO 在千位中始终也含有块类型的数值，适用于存在中间程序段的情况。 在 $AC_BLOCKTYPE 不为 0 时，千位不被传送。

$AC_SPLITBLOCK	
值：	含义：
0	没有修改的编程的程序段（通过压缩器生成的程序段也作为编程的程序段处理）。
1	有一个内部生成的程序段或者一个缩短的原始程序段。
3	内部生成的程序段或者缩短的原始程序段链中最后的程序段。

示例： 修整程序段的计数

程序代码	注释
$AC_MARKER[0]=0	
$AC_MARKER[1]=0	
$AC_MARKER[2]=0	
...	
	; 对用来计数修整程序段的同步动作进行定义

程序代码	注释

```
                        ; 所有的平滑程序段在 $AC_MARKER[0] 中计数：
ID=1 WHENEVER ($AC_TIMEC==0) AND ($AC_BLOCKTYPE==5) DO $AC_MARKER[0]=$AC_MARKER[0]+1
...
                        ; 用 G641 产生的平滑程序段在 $AC_MARKER[1] 中计数：
ID=2 WHENEVER ($AC_TIMEC==0) AND ($AC_BLOCKTYPEINFO==5001) DO $AC_MARKER[1]=$AC_MARKER[1]+1
                        ; 用 G642 产生的平滑程序段在 $AC_MARKER[2] 中计数：
ID=3 WHENEVER ($AC_TIMEC==0) AND ($AC_BLOCKTYPEINFO==5002) DO $AC_MARKER[2]=$AC_MARKER[2]+1
...
```

10.4 同步进行的动作

10.4.1 同步动作中的可能动作一览

同步动作中的动作由赋值、功能调用或参数调用、关键字或工艺循环组成。 通过运算符不能进行复杂的操作。

可能的应用有：

- 在 IPO 节拍中计算复杂的表达式
- 轴运动和主轴控制
- 在线修改和分析同步动作中的设定数据（例如将软件凸轮的位置和时间发送给 PLC 或者 NC 外围设备）
- 将辅助功能发送给 PLC
- 设立其它安全功能
- 设置叠加运动、在线刀具补偿和距离调节
- 在所有运行方式中执行动作
- 从 PLC 对同步动作进行干预
- 执行工艺循环
- 输出数字和模拟量信号
- 利用插补节拍采集同步动作的性能集合并且采集用来评估利用率的位置调节器的计算时间
- 操作界面中的诊断方法

同步动作	说明
DO $V…= DO $A...=	分配（伺服值） 分配变量（主运行变量）
DO $AC…[n]= DO $AC_MARKER[n]= DO $AC_PARAM[n]=	专门的主运行变量 读取或写入同步动作标记 读取或写入同步动作参数
DO $R[n]=	读取或写入计算变量
DO $MD...= DO $$SD...=	读取用于插补时间点的 MD 值 写入主运行中的 SD 值

同步动作	说明
DO $AC_TIMER[n]=始值	定时器
DO $AC_FIFO1[n] …FIFO10[n]=	FIFO 变量
DO $AC_BLOCKTYPE= DO $AC_BLOCKTYPEINFO= DO $AC_SPLITBLOCK=	解释当前的程序段（主运行变量）
DO M-，S 和 H，例如 M07	输出 M-,S-和 H 辅助功能
DO RDISABLE	设置读入禁止
DO STOPREOF	取消进给停止
DO DELDTG	在不停止进给的情况下快速删除剩余行程
FTCDEF(Polyn., LL, UL , Koeffiz.) DO SYNFCT(Polyn., Output, Input)	定义多项式 激活同步功能： AC 调节
DO FTOC	在线刀具补偿
DO G70/G71/G700/G710	确定定位任务的尺寸系统（英制或者公制尺寸）
DO POS[轴]= / DO MOV[轴]= DO SPOS[主轴]=	启动/定位/停止指令轴 起动/定位/停止主轴
DO MOV[轴] = 值	起动/停止指令轴的无限运动
DO POS[轴]= FA [轴]=	轴向进给 FA
ID=1 ... DO POS[轴]= FA [轴]= ID=2 ... DO POS[轴]= $AA_IM[轴] FA [轴]=	从同步动作中定位
DO PRESETON(轴，值)	设定实际值（从同步动作预先设定）
ID=1 EVERY $A_IN[1]=1 DO M3 S… ID=2 EVERY $A_IN[2]=1 DO SPOS=	启动/定位/停止主轴
DO TRAILON（FA，LA，耦合系数） DO LEADON(FA,LA,NRCTAB,OVW)	接通联动 接通引导值耦合
DO MEAWA（轴）= DO MEAC（轴）=	启用轴向测量 启用连续测量
DO [Feld n, m]=SET（值，值，...） DO [Feld n, m]=REP（值，值，...）	用值列表初始化数组变量 用相同的值初始化数组变量

同步动作	说明
DO SETM（标记编号） DO CLEARM（标记编号）	设置等待标记 删除等待标记
DO SETAL(报警编号)	设定循环报警（附加安全功能）
DO FXS[轴]= DO FXST[轴]= DO FXSW[轴]= DO FOCON[轴]= DO FOCOF[轴]=	选择运行到固定挡块 改变夹紧力矩 改变监控窗口 激活带极限力矩/力的运行（模态）FOC 取消带极限力矩/力的运行 （同步动作根据程序段生效）
ID=2 EVERY $AC_BLOCKTYPE==0 DO $R1=$AC_TANEB	当前程序段终点中的轨迹切线和已编程下一程序段起点中的轨迹切线之间的夹角
DO $AA_OVR= DO $AC_OVR= DO $AA_PLC_OVR DO $AC_PLC_OVR DO $AA_TOTAL_OVR DO $AC_TOTAL_OVR	轴向倍率 轨迹倍率 由 PLC 规定的轴向倍率 由 PLC 规定的轨迹倍率 得出的轴向倍率 得出的轨迹倍率
$AN_IPO_ACT_LOAD= $AN_IPO_MAX_LOAD= $AN_IPO_MIN_LOAD= $AN_IPO_LOAD_PERCENT= $AN_SYNC_ACT_LOAD= $AN_SYNC_MAX_LOAD= $AN_SYNC_TO_IPO=	当前 IPO 计算时间 最长 IPO 计算时间 最短 IPO 计算时间 当前的 IPO 计算时间与 IPO 节拍之间的比例 当前的计算时间，用于所有通道的同步动作 最长的计算时间，用于所有通道的同步动作 所有同步动作的百分比部分
DO TECCYCLE	执行工艺循环
DO LOCK(n, n, ...) DO UNLOCK(n, n, ...) DO RESET(n, n, ...)	禁用 释放 复位一个工艺循环
CANCEL(n, n, ...)	带有标识 ID(S)的模态同步动作在零件程序中删除

10.4.2 辅助功能输出

功能

输出时间

辅助功能的输出在同步动作中可间接用于动作的输出时间。 通过机床数据定义的辅助功能输出时间变无效。

当条件满足后，给出输出时间。

示例：

在某个轴位置时打开冷却液：
```
WHEN $AA_IM[X]>=15 DO M07 POS[X]=20 FA[X]=250
```

逐段方式同步动作（不带模态 ID）中允许的关键字

只能使用关键字 WHEN 或者 EVERY 才能在逐段有效的同步动作中（没有模态 ID）编程辅助功能。

说明

在同步动作中不允许以下辅助功能：

- M0, M1, M2, M17, M30: 程序停止/结束（在工艺循环时为 M2, M17, M30）。
- M6 或者通过机床数据设置的用于换刀的 M 功能

示例

程序代码	注释
`WHEN $AA_IW[Q1]>5 DO M172 H510`	`; 当 Q1 轴的实际值大于 5 mm 时，将辅助功能 M172 和 H510 发送给 PLC 。`

10.4.3 设定读入禁止 (RDISABLE)

功能

当满足条件时，使用 RDISABLE 停止主程序中的后续程序段处理。 编程的运动同步动作继续执行，后面的程序段也继续处理。

在有 RDISABLE 的程序段的结尾处，始终触发准停，而与读入禁止是否有效无关。 当控制系统处于连续路径运行(G64, G641 ... G645)中时，也会触发准停。

应用

例如根据外部输入端，使用 RDISABLE 可以在插补周期中启动程序。

示例

程序代码	注释
WHENEVER $A_INA[2]<7000 DO RDISABLE	; 如果输入端 2 上的电压低于 7V，程序会中止执行（假设：值 1000 对应 1 V）。
...	
N10 G01 X10	; 如果在执行中满足条件，在 N10 结尾处 RDISABLE 生效。
N20 Y20	
...	

边界条件

轴切换时 RDISABLE 的作用

如果 RDISABLE 在也同时执行轴切换的程序段中生效，那么 RDISABLE 在由轴切换触发的 REPOSA 程序段中也有效。

程序示例：

```
程序代码
N100 G0 G60 X300 Y300
N105 WHEN TRUE DO POS[X]=20 FA[X]=20000
N110 WHENEVER $AA_IM[X]<>20 DO RDISABLE
N115 G0 Y20
N120 Y-20
N125 M30
```

通过同步动作使 X 轴离开路径，会执行 REORG (REPOSA)。 RDISABLE 功能在 REPOSA 过程中生效。 由此 X 轴首先运行到其位置，接着在 N115 中运行到 Y20。

如果在 N101 中编写了 RELEASE(X) 或 WAITP(X)，则可阻止 REORG，因为这时 X 轴被使能为，例如，指令轴：

```
程序代码
N100 G0 G60 X300 Y300
N101 RELEASE(X)
N105 WHEN TRUE DO POS[X]=20 FA[X]=20000
...
```

10.4.4 取消进给停止 (STOPREOF)

功能

如果是显式编程的预处理停止 STOPRE 或者是通过一个激活的同步动作隐式激活的预处理停止，只要满足了条件， STOPREOF 就会在结束下一个加工程序段后取消预处理停止。

说明

STOPREOF 必须使用关键字 WHEN 并且逐段（没有 ID 号）进行编程。

示例

在程序段结束处快速进行程序分支。

程序代码	注释
WHEN $AC_DTEB<5 DO STOPREOF	; 当与程序段结束处的距离超过 5 mm 时，就取消预处理停止。
G01 X100	; 在线性插补执行完毕后取消预处理停止。
IF $A_INA[7]>500 GOTOF MARKE1=X100	; 当输入端 7 上的电压超过 5V 时跳转到标签 1。

10.4.5 删除剩余行程（DELDTG）

功能

轨迹和指定轴的剩余行程删除可以按照某个条件启动。

可以使用：

- 快速、经过预处理的剩余行程删除
- 剩余行程删除，未经预处理

经过预处理的剩余行程删除 DELDTG 会对触发事件作出极为迅速的反应，因此可在时间紧张的情况下使用，例如当

- 删除剩余行程和启动后续程序段之间的时间很短。
- 很可能会满足剩余行程删除的条件。

说明

置于 DELDTG 后面的括号中的轴名称仅对一个 定位轴有效。

句法

轨迹的剩余行程删除

```
DO DELDTG
```

轴的剩余行程删除

```
DO DELDTG（轴 1） DELDTG（轴 2） ...
```

轨迹快速剩余行程删除举例

程序代码	注释
WHEN $A_IN[1]==1 DO DELDTG	
N100 G01 X100 Y100 F1000	; 当输入端已设定时，中断运动。
N110 G01 X…	
IF $AA_DELT>50…	

快速轴剩余行程删除举例

程序代码	注释
中断定位运动：	
ID=1 WHEN $A_IN[1]==1 DO MOV[V]=3 FA[V]=700	; 启动轴
WHEN $A_IN[2]==1 DO DELDTG(V)	; 剩余行程删除，使用 MOV=0 使轴停止
取决于输入端电压，删除剩余行程：	
WHEN $A_INA[5]>8000 DO DELDTG(X1)	; 只要在输入端 5 上电压超过 8V，就删除轴 X1 的剩余行程。 轨迹运动会继续。
POS[X1]=100 FA[X1]=10 G1 Z100 F1000	

其它信息

在预置式剩余行程删除已被释放的运动程序段末尾处隐性激活进给停止。

因此在程序段结束处，用快速的剩余行程删除中断或者停止轨迹控制运行或定位轴运动。

说明

预置的剩余行程删除：

- 在有效的刀具半径补偿时不可以使用。
- 仅在程序段方式有效的同步动作中（没有 ID 号）编程动作。

10.4.6 多项式定义（FCTDEF）

功能

使用 FCTDEF 可以定义 3 阶多项式，形式为 $y=a_0+a_1x+a_2x^2+a_3x^3$。 该多项式由在线刀具补偿 FTOC 和计算功能 SYNFCT 使用。

句法

```
FCTDEF（多项式编号，LLIMIT，ULIMIT，a0，a1，a2，a3）
```

含义

多项式编号	3 阶多项式序号
LLIMIT	函数值下限
ULIMIT	函数值上限
a0, a1, a2, a3	多项式系数

该值也可通过系统变量存取

$AC_FCTLL[n]	函数值下限
$AC_FCTUL[n]	函数值上限
$AC_FCT0[n]	a_0
$AC_FCT1[n]	a_1

$AC_FCT2[n]	a_2
$AC_FCT3[n]	a_3

说明

写入系统变量

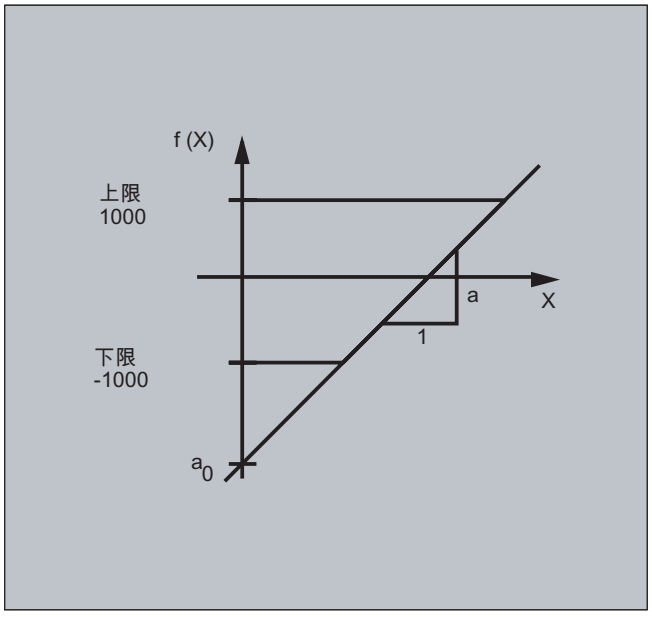
- 系统变量可以由零件程序或者从一个同步动作写入。 当从零件程序写入时，必须通过编程 STOPRE 来进行逐段同步写入。
- 系统变量 $AC_FCTLL[n]，$AC_FCTUL[n]，$AC_FCT0[n] 直到 $AC_FCTn[n] 可从同步动作中修改

在从同步动作中写入时，多项式系数和函数值界限立即生效。

直线段多项式举例

上限为 1000、下限为 -1000、纵坐标线段为 a_0=$AA_IM[X] 且斜率为 1 的多项式定义：

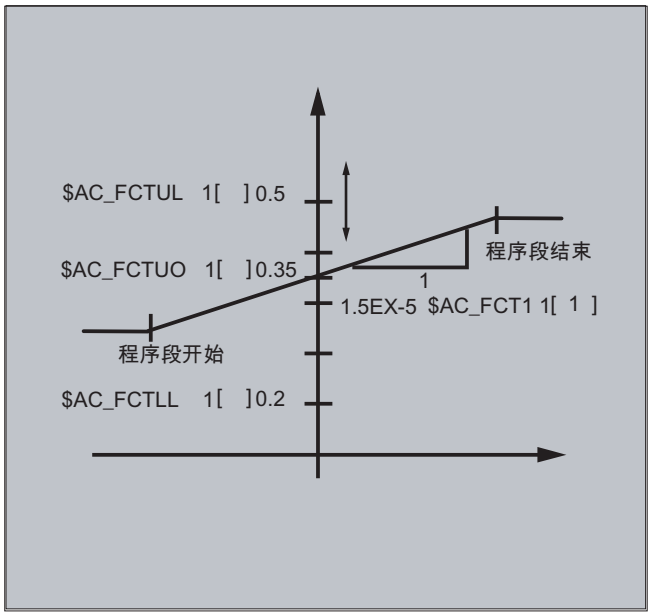
```
FCTDEF(1, -1000,1000,$AA_IM[X],1)
```



激光功率控制举例

激光器功率控制系统是可能的多项式定义应用。

激光器功率控制系统是指：
例如以轨迹速度为依据来干预某个模拟输出。



程序代码	注释
`$AC_FCTLL[1]=0.2`	; 定义多项式系数
`$AC_FCTUL[1]=0.5`	
`$AC_FCT0[1]=0.35`	
`$AC_FCT1[1]=1.5EX-5`	
`STOPRE`	
`ID=1 DO $AC_FCTUL[1]=$A_INA[2]*0.1 +0.35`	; 在线修改上限。
`ID=2 DO SYNFCT(1,$A_OUTA[1],$AC_VACTW)`	; 视轨迹速度而定（保存在 $AC_VACTW 中）， 通过模拟输出端 1 来控制激光功率控制系统

说明

通过 SYNFCT 使用上面定义的多项式。

10.4.7 同步功能（SYNFCT）

功能

SYNFCT 计算用输入变量加权的第 3 级多项式的初值。 结果在输出变量中并有上限和下限。

求值功能应用于：

- AC-调节（自适应控制），
- 激光功率控制，
- 位置前馈。

句法

SYNFCT（多项式编号，主运行变量输出，主运行变量输入）

含义

作为输出变量，可以选择变量，

- 它们对处理过程有加法影响
- 它们对处理过程有乘法影响
- 它们作为位置补偿
- 直接

进入加工。

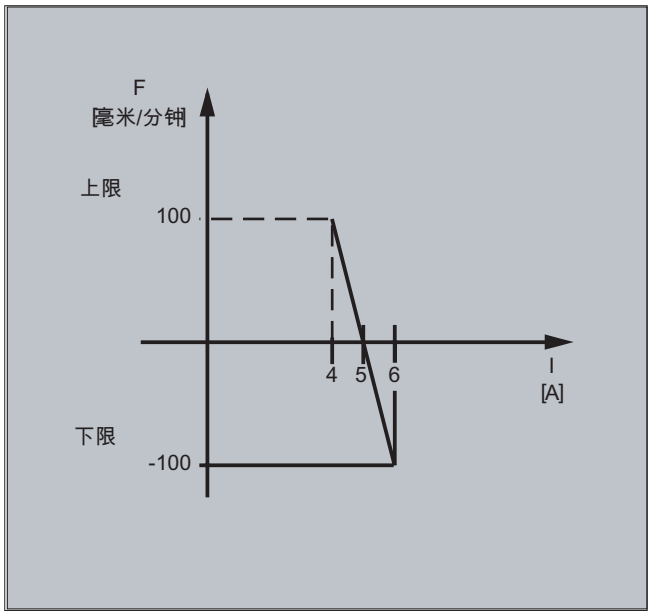
DO SYNFCT	激活分析功能
多项式编号	用 FCTDEF 定义的多项式（参见章节“多项式定义”）
主运行变量输出	写入主运行变量
主运行变量输入	读取主运行变量

AC 调节举例（相加式）

编程进给，加法影响

通过 X 轴（横向进给轴）附加调节一个编程的进给：

进给应在+/- 100 mm/min 上下改变，电流则在 5A 的工作点波动+/-1A。



1. 多项式定义

确定系数

$y = f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$

a_1 = -100mm/1 min A

a_0 = -(-100)*5 =500

$a_2 = a_3 = 0$ (不是二次项和三次项)

上限＝100

下限＝-100

由此产生：

```
FCTDEF(1,-100,100,500,-100,0,0)
```

启用第 2 个 AC 闭环控制

```
ID=1 DO SYNFCT(1,$AC_VC,$AA_LOAD[x])
```

;通过 $AA_LOAD[x] 读取当前轴负荷 (% 最大驱动电流),
;使用上述定义的多项式计算轨迹进给补偿。

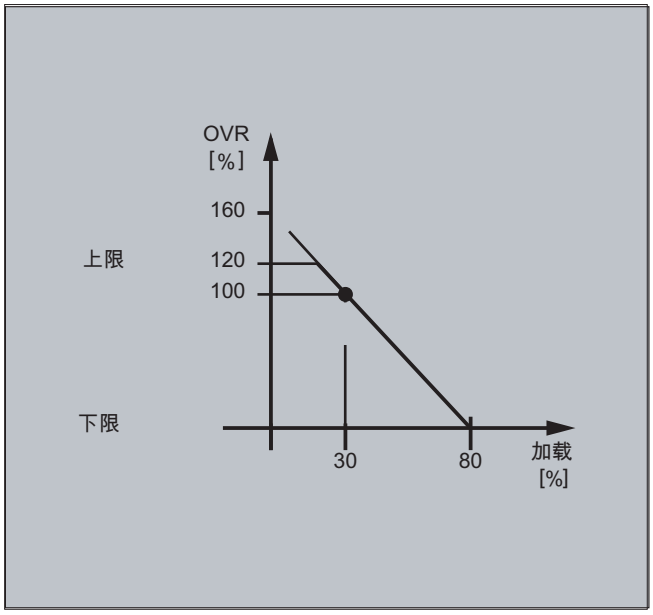
AC 调节举例（倍增式）

编程的进给，乘法影响

应当以倍增式干预已编程的进给，同时进给不应超过一定的极限（取决于驱动装置的负荷）：

- 当驱动负载为 80%时，应停止进给： 倍率＝0。
- 当驱动负载为 30%时，允许按照编程的进给运行：
  修调率 = 100%.

进给速度允许超出最大 20%:
最大修调率 = 120%.



1. 多项式定义

确定系数

$y = f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$

$a_1 = -100\%/(80-30)\% = -2$

$a_0 = 100 + (2*30) = 160$

$a_2 = a_3 = 0$（不是二次项和三次项）

上限＝120

下限＝0

由此产生：

```
FCTDEF(2,0,120,160,-2,0,0)
```

启用第 2 个 AC 闭环控制

```
ID=1 DO SYNFCT(2,$AC_OVR,$AA_LOAD[x])
```

;通过 $AA_LOAD[x] 读取当前轴负荷 (% 最大驱动电流),
;使用上述定义的多项式计算进给修调率。

10.4.8 带限定的补偿系数的调节间距（$AA_OFF_MODE）

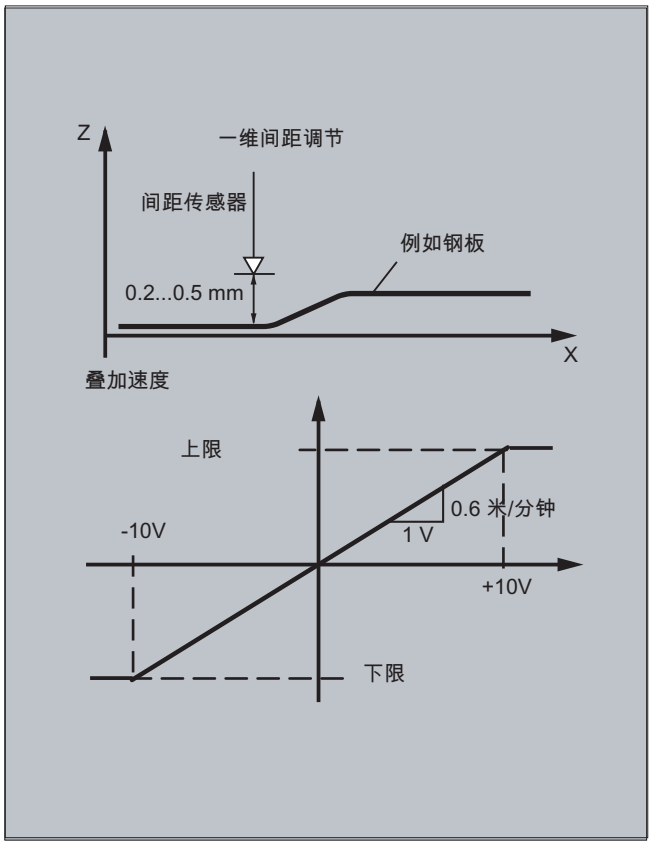
说明

该功能不能用于 SINUMERIK 828D。

功能

以检查极限范围的方式来对间距值进行积分计算：

$AA_OFF_MODE = 1



注意
叠加的控制环的增益与插补周期的设定相关。 解决方法： 读取并计算用于插补周期的机床数据。

说明

插补周期为 12 ms 时通过 MD32020 JOG_VELO 设置的叠加插补器的速度限制。

速度公式：

$$\frac{0.120\text{mm}}{0.6\text{ms}} / \text{mV} = 0.6 \frac{\text{m}}{\text{min}} / \text{V}$$

示例

子程序"AON"： 启用间距调节

程序代码	注释
PROC AON	
$AA_OFF_LIMIT[Z]=1	; 确定极限值。
FCTDEF(1, -10, +10, 0, 0.6, 0.12)	; 多项式定义
ID=1 DO SYNFCT(1,$AA_OFF[Z],$A_INA[3])	; 间隙控制已激活。
ID=2 WHENEVER $AA_OFF_LIMIT[Z]<>0 DO $AA_OVR[X] = 0	; 当超过极限范围时锁止轴 X。
RET	
ENDPROC	

子程序"AOFF"： 关闭间距调节

程序代码	注释
PROC AOFF	
CANCEL(1)	; 删除间距调节同步动作
CANCEL(2)	; 删除极限范围检查
RET	
ENDPROC	

主程序"MAIN"

程序代码	注释
AON	; 启用间距调节
...	
G1 X100 F1000	
AOFF	; 关闭间距调节
M30	

其它信息

在基准坐标系中的位置偏移

使用系统变量 $AA_OFF[轴] 可以给通道中的每个轴叠加一个运动。 它作为基准坐标系中的位置偏移。

如此编程的位置偏移立即叠加到相应的轴，而与该轴是否编程运行无关。

限制主运行变量输出范围：

可以给绝对要补偿的值（主运行变量输出）限制为设定数据 SD43350 $SA_AA_OFF_LIMIT 中存储的值。

通过机床数据 MD36750 $MA_AA_OFF_MODE 确定间距叠加的方式：

值	含义
0	比例加权
1	积分加权

使用系统变量 $AA_OFF_LIMIT[轴] 可以根据方向来查询补偿值是否在极限范围内。 可以从同步动作中查询该系统变量，并且当达到某个极限值时，（例如）使轴停止或者发出报警。

0: 补偿值不在极限范围之内

1 到达正向补偿值极限

-1: 在负方向达到补偿值的极限

10.4.9 联机刀具补偿 (FTOC)

功能

FTOC 可以根据一个基准值，如轴的实际值，按照 FCTDEF 所编程的某个多项式来实现某个几何轴的叠加运动。

FCTDEF(...)函数定义中的系数 a_0 会在 FTOC 中运行。 上下限都取决于 a_0。

FTOC 可以编程模态的在线刀具补偿，或者将距离调节作为同步动作编程。

该功能可用于加工工件、修整同一个通道中或者不同通道（加工通道和修整通道）中的砂轮。

砂轮修整的边界条件和规定适用于FTOC，类似于用PUTFTOCF的在线刀具补偿（参见"在线刀具补偿 (PUTFTOCF, FCTDEF, PUTFTOC, FTOCON, FTOCOF) (页 420)"）

句法

```
FCTDEF(<函数>,<下限>,<上限>,<a0>,<a1>,<a2>,<a3>)
FTOC(<函数>,<参考值>,<刀具参数>,<通道>,<主轴>)
...
```

含义

FCTDEF: 通过 FCTDEF 可以定义 FTOC 的多项式函数。

参数：

<函数>: 多项式函数的编号
类型： INT
取值范围： 1 ... 3

<下限>: 值下限
类型： REAL

<上限>: 值上限
类型： REAL

<a0> ... <a3>: 多项式函数的系数
类型： REAL

DO FTOC: 执行功能“持续模态写入在线刀具补偿”

参数：

<函数>: 多项式函数的编号

类型： INT

取值范围： 1 ... 3

提示：
它必须和 FCTDEF 的数据一致。

<参考值>: 主运行变量，需要通过 FCTDEF 定义的多项式函数来计算一个函数值。

类型： VAR REAL

<刀具参数>: 磨削参数的编号，长度 1、2、3，补偿值将添加到该参数中。

类型： INT

<通道>: 通道号，在线刀具补偿在该通道中生效。

类型： INT

提示：
只有当需要补偿在未激活的通道中生效时，才需要给定该数据。

<主轴>: 主轴号，在该主轴上在线补偿将生效。

类型： INT

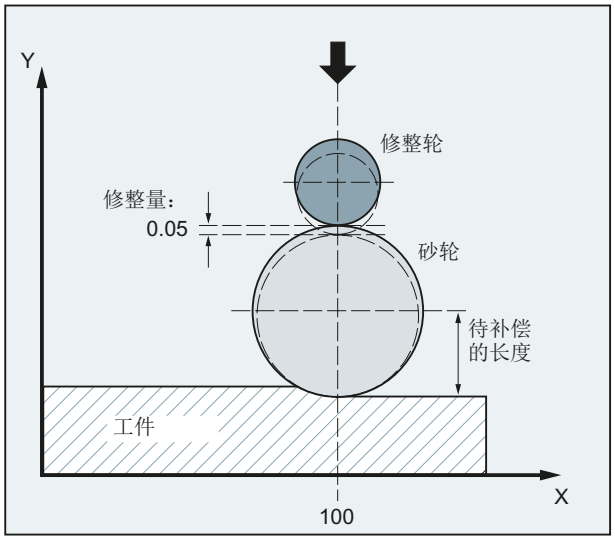
提示：
只有当需要补偿针对未激活的磨削砂轮，而不是针对激活的、使用中的刀具生效时，才需要给定该数据。

说明

在目标通道中必须已启用 FTOCON。

示例

应该修整当前有效的、正在使用的砂轮的长度。



程序代码	注释
FCTDEF(1,-1000,1000,-$AA_IW[V],1)	; 定义函数。
ID=1 DO FTOC(1,$AA_IW[V],3,1)	; 选择在线刀具补偿： V 轴的实际值为多项式 1 的输入值，结果在通道 1 中作为补偿值添加给已激活砂轮的长度 3。
WAITM(1,1,2)	; 与加工通道同步。
G1 V-0.05 F0.01 G91	; 用于修整的进给运动。
G1 V-0.05 F0.02	
...	
CANCEL(1)	; 取消在线补偿。
...	

10.4.10 在线刀具长度补偿（$AA_TOFF[刀具方向]）

功能

通过这个系统变量$AA_TOFF[ ]可以实时三维叠加和三个刀具方向相符的有效的刀具长度。

三个几何轴标识符作为索引使用。 这样，当前有效的补偿方向的数量便可以由同时有效的几何轴来确定。

所有的补偿可以同时有效。

句法

```
N... TRAORI
N... TOFFON(X,<偏移值>)
N... WHEN TRUE DO $AA_TOFF[X]
N... TOFFON(Y,<偏移值>)
N... WHEN TRUE DO $AA_TOFF[Y]
N... TOFFON(Z,<偏移值>)
N... WHEN TRUE DO $AA_TOFF[Z]
```

含义

TOFFON:	**激活**在线刀具长度补偿	
	X, Y, Z:	在线刀具长度补偿作用的刀具方向。
	<偏移值>:	在激活时可以针对相应的补偿方向指定一个立即执行的偏移值。
TOFFOF:	**取消**在线刀具长度补偿	
	针对指定补偿方向上的补偿值被取消，并触发预处理停止。	
$AA_TOFF[X]=<值>:	X 方向的叠加	
$AA_TOFF[Y]=<值>:	Y 方向的叠加	
$AA_TOFF[Z]=<值>:	Z 方向的叠加	

示例

示例 1： 选择刀具长度补偿

程序代码	注释
N10 TRAORI(1)	; 坐标转换激活。
N20 TOFFON(Z)	; 激活用于 Z 轴刀具方向的在线刀具长度补偿。
N30 WHEN TRUE DO $AA_TOFF[Z]=10 G4 F5	; 为 Z 轴刀具方向插补一个大小为 10 的刀具长度补偿。
N40 TOFFON(X)	; 激活用于 X 轴刀具方向的在线刀具长度补偿
N50 ID=1 DO $AA_TOFF[X]=$AA_IW[X2] G4 F5	; 根据轴 X2 的位置，对 X 轴刀具方向进行补偿。
...	
	; 在 X 方向分配当前补偿。 对于 X 刀具方向，将刀具长度补偿重新恢复为 0：
N100 XOFFSET=$AA_TOFF_VAL[X] N120 TOFFON(X,-XOFFSET) G4 F5	

示例 2： 取消刀具长度补偿

程序代码	注释
N10 TRAORI(1)	; 坐标转换激活。
N20 TOFFON(X)	; 激活用于 X 轴刀具方向的在线刀具长度补偿
N30 WHEN TRUE DO $AA_TOFF[X] = 10 G4 F5	; 为 X 轴刀具方向插补一个大小为 10 的刀具长度补偿。
...	
N80 TOFFOF(X)	; 删除 X 轴刀具方向的位置偏移： ...$AA_TOFF[X]=0 不运行轴。 根据当前的定位将位置偏移添加至 WCS 中的当前位置。

10.4.11 定位运动

功能

轴可以与零件程序完全异步，由同步动作定位。由同步动作编程定位轴，建议用于循环过程或者由事件控制的过程。从同步运动中编程的轴叫做**指令轴**。

编程

文献：
/PG/ 编程手册基础部分；章节“位移说明”
/FBSY/ 同步动作功能描述；“起动指令轴”

参数

同步动作中定位任务的测量单位制用 G 代码 G70/G71/G700/G710 来确定。

通过编程同步动作中的 G 功能，可以确定同步动作的 INCH/METRIC 求值系统，而与零件程序文本无关。

10.4.12 定位轴 (POS)

功能

与对零件程序进行编程不同，定位轴运动对零件程序的执行没有影响。

句法

POS[轴]=值

含义

DO POS	起动/定位指令轴
轴	应当运行的轴名称。
值	待运行值的说明（根据运行方式）

示例

示例 1：

程序代码	注释
ID=1 EVERY $AA_IM[B]>75 DO POS[U]=100	; 轴 U 根据运动模式以 100（inch/mm）的增量运动或者向控制零点的位置 100（inch/mm）运动。
	; 轴 U 以由主运行变量计算的位移向前移动：
ID=1 EVERY $AA_IM[B]>75 DO POS[U]=$AA_MW[V]-$AA_IM[W]+13.5	

示例 2：

编程环境影响定位轴的定位行程
（同步动作的动作程序段中没有 G 功能）：

程序代码	注释
N100 R1=0	
N110 G0 X0 Z0	
N120 WAITP(X)	
N130 ID=1 WHENEVER $R==1 DO POS[X]=10	
N140 R1=1	
N150 G71 Z10 F10	; Z=10mm X=10mm
N160 G70 Z10 F10	; Z=254mm X=254mm
N170 G71 Z10 F10	; Z=10mm X=10mm
N180 M30	

同步动作的动作程序段中的 G71 用来唯一确定定位轴的定位行程（公制），与编程环境无关。

程序代码	注释
N100 R1=0	
N110 G0 X0 Z0	
N120 WAITP(X)	
N130 ID=1 WHENEVER $R==1 DO G71 POS[X]=10	
N140 R1=1	
N150 G71 Z10 F10	; Z=10mm X=10mm
N160 G70 Z10 F10	; Z=254mm X=10mm （X 总是定位到 10mm 处）
N170 G71 Z10 F10	; Z=10mm X=10mm
N180 M30	

如果不要在程序段开始处启动轴运动，可将从某个同步动作到所需开始时间点的轴倍率保持为 0：

程序代码	注释
WHENEVER $A_IN[1]==0 DO $AA_OVR[W]=0 G01 X10 Y25 F750 POS[W]=1500 FA=1000	; 一旦数字输入端 1＝0，则定位轴被一直停止。

10.4.13 规定的参考区域中的位置（POSRANGE）

功能

使用功能 POSRANGE（ ）可以确定，某个轴的当前插补给定位置是否在某个规定参考位置的某个窗口中。 位置参数可能以可规定的坐标系为参考。

在询问某个模态轴的轴实际位置时考虑取模补偿。

说明

仅可从同步动作中调用该功能。 在从零件程序中调用时，不允许执行报警 14091 %1 程序段 %2 功能，索引： %3 使用索引 5 调用。

句法

```
BOOL POSRANGE（轴，Refpos, Winlimit, [Coord]）
```

含义

BOOL POSRANGE	指令轴的当前位置在规定参考位置的窗口中。
AXIS <轴>	加工轴、通道轴或几何轴的轴标识符
REAL Refpos	Coord 坐标系中的参考位置
REAL Winlimit	量值，用于得出位置窗口的极限值
INT Coord	可以选择的是 MCS 有效。 允许的值有： 0 用于 MCS（机床坐标系） 1 用于 BCS（基准坐标系） 2 用于 ENS（可设置的零点系统） 3 用于 WCS（工件坐标系）

函数值

规定的坐标系中根据位置数据的当前给定位置

函数值： TRUE	如果 Refpos(Coord) - abs(Winlimit) ≤ Actpos(Coord) ≤ Refpos(Coord) + abs(Winlimit)
函数值： FALSE	其它

10.4.14 起动/停止轴 (MOV)

功能

通过 MOV[轴]=值，可以启动一个指令轴，而不对终点位置说明。 轴在编程的方向运行，直至通过一个新的运动指令或者定位指令规定另一个运动，或者该轴通过一个停止指令停止。

句法

```
MOV[轴] = 值
```

含义

DO MOV	起动指令轴运动
轴	应当启动的轴名称。
值	运动/停止运动的开始指令 前置符用来确定运动方向 该值的数据类型是 INTEGER。
值 >0 (通常为 +1)	正方向
值 <0 (通常为 -1)	负方向
值 ==0	停止轴运动

说明

当使用 MOV[轴] = 0 使分度轴停止时，就会在下一个分度位置将轴停住。

示例

程序代码	注释
... DO MOV[U]=0	; U 轴被停止

10.4.15 轴交换(RELEASE, GET)

功能

可以请求相应的指令轴作为某个带有 GET（轴）的同步动作的动作用于刀具更换。 该通道分配的轴类型和由此与该时间相连的插补权限可以通过系统变量 $AA_AXCHANGE_TYP 询问。 根据自身的状态和由通道占用的该轴的当前插补权限可以进行不同的过程。

如果已执行刀具更换，则可以为通道释放该指令轴作为某个带有 RELEASE（轴）的同步动作。

机床制造商

相关的轴必须通过机床数据分配到通道。 请注意机床制造商说明。

句法

GET(轴[,轴{,...}]) 请求轴

RELAESE(轴[,轴{,...}]) 释放轴

含义

DO RELEASE	轴被作为中性轴释放
DO GET	取轴用于轴交换
轴	应当启动的轴名称。

例如：为两个通道的一次轴交换运行程序

轴 Z 在通道 1 和 2 中已知。

在通道 1 中的程序运行

程序代码	注释
WHEN TRUE DO RELEASE(Z)	; Z 轴成为中性轴
WHENEVER($AA_TYP[Z]==1) DO RDISABLE	; 只要 Z 轴为程序轴就禁止读取
N110 G4 F0.1	
WHEN TRUE DO GET(Z)	; Z 轴重新成为 NC 程序轴
WHENEVER($AA_TYP[Z]<>1) DO RDISABLE	; 禁止读取，直到 Z 轴成为程序轴
N120 G4 F0.1	
WHEN TRUE DO RELEASE(Z)	; Z 轴成为中性轴
WHENEVER($AA_TYP[Z]==1) DO RDISABLE	; 只要 Z 轴为程序轴就禁止读取
N130 G4 F0.1	;
N140 START(2)	; 启动第 2 个通道

在通道 2 中的程序运行

程序代码	注释
WHEN TRUE DO GET(Z)	; ; 取 Z 轴至通道 2
WHENEVER($AA_TYP[Z]==0) DO RDISABLE	; ; 禁止读取，只要 Z 轴在另一个 ; 通道
N210 G4 F0.1	
WHEN TRUE DO GET(Z)	; ; Z 轴成为 NC 程序轴
WHENEVER($AA_TYP[Z]<>1) DO RDISABLE	; ; 禁止读取直到 Z 轴成为程序轴

程序代码	注释
N220 G4 F0.1	
WHEN TRUE DO RELEASE(Z)	; ；Z 轴在第 2 通道为中性轴
WHENEVER($AA_TYP[Z]==1) DO RDISABLE	; ；只要 Z 轴为程序轴禁止读取
N230 G4 F0.1	
N250 WAITM(10, 1, 2)	; 与通道 1 同步

通道 1 中的其它程序运行

程序代码	注释
N150 WAIM(10, 1, 2)	; 与通道 2 同步
WHEN TRUE DO GET(Z)	; 取 Z 轴至该通道
WHENEVER($AA_TYP[Z]==0) DO RDISABLE	; 禁止读取，只要 Z 轴在另一个通道
N160 G4 F0.1	
N199 WAITE(2)	
N999 M30	; 在通道 2 中等待程序结束

举例在工艺循环中交换轴

轴 U ($MA_AUTO_GET_TYPE=2)在通道 1 和 2 中已知，当前通道 1 具有插补权。 在通道 2 中启动下面的工艺循环：

程序代码	注释
GET(U)	; 在通道中取 U 轴
POS[U]=100	; 运行 U 轴到位置 100

如果 U 轴被取至通道 2，才执行指令运动行 POS[U]。

工作流程

在指令 GET（轴）激活的时间点请求的轴，可以按照用于轴交换的轴类型用系统变量（$AA_AXCHANGE_TYP[<Achse>]进行读取:

- 0: 轴分配给 NC 程序
- 1: 轴分配给 PLC 或者作为指令轴或者摆动轴激活
- 2: 另一个通道具有插补权

- 3: 轴是中性轴
- 4: 中性轴由 PLC 控制
- 5: 另一个通道具有插补权，轴被请求用于 NC 程序
- 6: 另一个通道具有插补权，轴被请求用作中性轴
- 7: PLC 轴或者作为指令轴或者摆动轴激活，轴被请求用于 NC 程序
- 8: PLC 轴或者作为指令轴或者摆动轴激活，轴被请求用作中性轴

边界条件

相关的轴必须通过机床数据分配到通道。

一个仅仅由 PLC 控制的轴不能分配给 NC 程序。

文献：
/FB2/功能手册扩展功能；定位轴（P2）

用 GET 指令请求另一个通道的轴

在激活指令 GET 的时间点，另**一个通道具有书写权**即对轴($AA_AXCHANGE_TYP[<轴>] == 2)的(插补权),则通过该通道的轴交换对轴请求 ($AA_AXCHANGE_TYP[<轴>]==6)，且一有可能就分配给请求的通道。

它接受中性轴状态($AA_AXCHANGE_TYP[<轴>]==3)。

在请求的通道中无法重新分组。

分配**作为带重新分组的 NC 程序轴：**

如果在激活指令 GET 的时间点，轴已经被请求作为中性轴($AA_AXCHANGE_TYP[<轴>]==6),则轴被请求用于 NC 程序($AA_AXCHANGE_TYP[<轴>]==5)，且一有可能就分配给通道的 NC 程序 ($AA_AXCHANGE_TYP[<轴>]==0)。

轴已经分配给请求的通道

分配**作为带重新分组的 NC 程序轴：**

如果请求的轴在激活的时间点已经分配给了请求的通道，且为中性轴状态-不受 PLC 控制-($AA_AXCHANGE_TYP[<轴>]==3),则将其分配给 NC 程序 ($AA_AXCHANGE_TYP[<轴>]==0)。

轴在中性轴状态时由 PLC 控制

如果该轴在中性轴状态由 PLC 控制($AA_AXCHANGE_TYP[<轴>]==4), 则请求该轴作为中性轴($AA_AXCHANGE_TYP[<轴>] == 8), 因此该轴在机床数据 MD 10722 中与位 0 相关:禁止 AXCHANGE_MASK 用于通道间的自动轴交换(位 0 == 0)。这个符合($AA_AXCHANGE_STAT[<轴>] == 1)。

轴作为中性指令轴或者回转轴激活或者分配给 PLC。

如果该轴作为指令轴或者回转轴激活或者分配给 PLC 运行，PLC 轴==竞争定位轴($AA_AXCHANGE_TYP[<轴>]==1)，则请求该轴作为中性轴($AA_AXCHANGE_TYP[<轴>] == 8)，因此因此该轴在机床数据 MD 10722 中与位 0 相关：禁止 AXCHANGE_MASK 用于通道间的自动轴交换(位 0 == 0)。这个符合($AA_AXCHANGE_STAT[<轴>] == 1)。

一个新的 GET 指令要求轴用于 NC 程序($AA_AXCHANGE_TYP[<轴>]将 == 7)。

轴已经分配给 NC 程序

如果该轴已经分配给了通道的 NC 程序($AA_AXCHANGE_TYP[<轴>]==0)或者已经请求分配，例如：释放 NC 程序的轴交换($AA_AXCHANGE_TYP[<轴>]==5 或者 $AA_AXCHANGE_TYP[<轴>] == 7)，则不出现状态改变。

10.4.16 轴向进给 (FA)

功能

指令轴的轴向进给为模态有效。

句法

```
FA[<轴>]=<值>
```

示例

程序代码	注释
`ID=1 EVERY $AA_IM[B]>75 DO POS[U]=100 FA[U]=990`	; 固定规定进给值。
	; 由主运行变量构成进给值：
`ID=1 EVERY $AA_IM[B]>75 DO POS[U]=100 FA[U]=$AA_VACTM[W]+100`	

10.4.17 SW 限位开关

功能

与设定数据$SA_WORKAREA_PLUS_ENABLE 相关，考虑指令轴的、G25/G26 编程的工作区域限制。

通过零件程序中 G 功能 WALIMON/WALIMOF 开关工作区域限制，这对指令轴不起作用。

10.4.18 轴协调

功能

标准情况下，一个轴或者由零件程序或者作为定位轴由同步动作运动。

如果同一个轴交替地由零件程序作为轨迹轴或者定位轴运行，或者由同步动作运行，则在两个轴运动之间进行一次协调传送。

如果一个指令轴紧接着由零件程序运行，则它要求一个预处理的重组。 它再次决定零件程序加工的中断，与进刀停止类似。

可选择从零件程序和同步动作中运动的 X 轴举例

程序代码	注释
N10 G01 **X100** Y200 F1000	; 在零件程序中编程 X 轴
…	
N20 ID=1 WHEN $A_IN[1]==1 DO **POS[X]=150** FA[X]=200	; 当存在数字输入端时，从同步动作中开始定位
…	
取消（1）	; 取消同步动作
…	
N100 G01 **X240** Y200 F1000	; X 成为轨迹轴；如果数字输入端为 1 且已从同步运动中定位了 X，就会由于轴变换而出现等候时间 。

改变同一轴的运动指令举例

程序代码	注释
ID=1 EVERY $A_IN[1]>=1 DO POS[V]=100 FA[V]=560	; 当数字输入端 >= 1 时，从同步运动中开始定位
ID=2 EVERY $A_IN[2]>=1 DO POS[V]=$AA_IM[V] FA[V]=790	; 轴跟随运动，第 2 个输入端被设定，即在连续运动的情况下，当有两个同时激活的同步动作时，V 轴的终点位置和进给 将被连续跟踪 。

10.4.19 设定实际值 (PRESETON)

功能

当执行 PRESETON (轴，值) 时，不改变当前轴位置，给其赋一个新值。

出自同步动作的 PRESETON 可以用于：

- 取模回转轴，由零件程序启动
- 已从同步动作中起动的所有指令轴

句法

DO PRESETON（轴，值）

含义

DO PRESETON	同步动作中的实际值设定
轴	要改变其控制零点的轴
值	以此来改变控制零点的值

轴的限制

PRESETON 不可用于已参与转换的轴。

该轴仅可以错开时间由零件程序或者一个同步动作运动，因此如果该轴事先在一个同步动作中编程，则在由零件程序编程一个轴时可能会出现等待时间。

如果交替使用相同的轴，则在两个轴运动之间协调传送一次。 必须为此中断零件程序处理。

示例

移动某个轴的控制零点

程序代码	注释
`WHEN $AA_IM[a] >= 89.5 DO PRESETON(a4,10.5)`	`;` 沿轴的正向将轴 a 的控制零点移动 10.5 个长度单位（inch 或者 mm）

10.4.20 主轴运动

功能

主轴可以与零件程序完全异步，由同步动作定位。 这种编程方式建议用于循环过程或者由事件控制的过程。

当通过同时激活的同步动作给某个主轴规定补偿指令时，则上一次的主轴指令有效。

启动/停止/定位主轴举例

程序代码	注释
`ID=1 EVERY $A_IN[1]==1 DO M3 S1000`	`;` 设置旋转方向和转速
`ID=2 EVERY $A_IN[2]==1 DO SPOS=270`	`;` 定位主轴

设置旋转方向、转速/定位主轴举例

程序代码	注释
`ID=1 EVERY $A_IN[1]==1 DO M3 S300`	`;` 设置旋转方向和转速
`ID=2 EVERY $A_IN[2]==1 DO M4 S500`	`;` 规定新的旋转方向和新的转速
`ID=3 EVERY $A_IN[3]==1 DO S1000`	`;` 规定新的转速
`ID=4 EVERY ($A_IN[4]==1) AND ($A_IN[1]==0) DO SPOS=0`	`;` 定位主轴

10.4.21 联动 (TRAILON, TRAILOF)

功能

在由同步动作接通耦合时，可能是引导轴处于运动当中。 在这种情况下，跟随轴加速到给定速度。 在速度同步时引导轴的位置是联动的起始位置。 联动功能在“轨迹运动特性”一章中有所描述。

句法

接通联动

```
DO TRAILON(跟随轴, 引导轴, 耦合系数)
```

关闭联动

```
DO TRAILOF（跟随轴，引导轴，引导轴 2）
```

含义

激活异步联动：

`... DO TRAILON(FA, LA, Kf)`	其中： FA： 跟随轴 LA: 引导轴 Kf: 耦合系数

取消异步联动：

`... DO TRAILOF(FA, LA, LA2)`	其中： FA： 跟随轴 LA: 引导轴，选件 LA2： 引导轴 2，选件
`... DO TRAILOF(FA)`	断开所有与跟随轴的耦合。

示例

程序代码	注释
`$A_IN[1]==0 DO TRAILON(Y,V,1)`	`;` 当数字输入端为 1 时，启用第 1 个联动组合
`$A_IN[2]==0 DO TRAILON(Z,W,-1)`	`;` 启用第 2 个联动组合

程序代码	注释
G0 Z10	; Z 轴和 W 轴以相反的轴向进给
G0 Y20	; Y 轴和 V 轴以相同的轴向进给
...	
G1 Y22 V25	; 叠加"V"轴的某个相关和不相关的运动
...	
TRAILOF(Y,V)	; 关闭第 1 个联动组合
TRAILOF(Z,W)	; 关闭第 2 个联动组合

使用 TRAILOF 避免冲突举例

为了使某个已经耦合的轴断开以便可以作为通道轴重新存取，必须预先调用功能 TRAILOF 。 在通道请求相应的轴之前，必须保证已经执行了 TRAILOF 。 在下面的示例中并非这种情况

…

```
N50 WHEN TRUE DO TRAILOF(Y,X)
N60 Y100
```

…

在这种情况下，不会及时释放轴，因为带有 TRAILOF 的逐段起作用的同步动作同时以 N60 激活，参阅运动同步动作一章"结构，一般基础部分"。
为了避免出现冲突情况，应以

下列方式运动

…

```
N50 WHEN TRUE DO TRAILOF(Y,X)
N55 WAITP(Y)
N60 Y100
```

10.4.22 引导值偶合 (LEADON, LEADOF)

说明
该功能不能用于 SINUMERIK 828D。

功能

轴向的引导值耦合可以在同步动作中编程，不受限制。 只有在同步动作中才可以选择在之前没有某个新同步动作的情况下修改现有耦合的某个曲线图表。

句法

启用引导值耦合

```
DO LEADON(跟随轴，引导轴，曲线图表 编号，OVW)
```

关断引导值耦合

```
DO LEADOF(跟随轴，引导轴，引导轴 2)
```

含义

接通轴向引导值耦合：

`...DO LEADON(FA, LA, NR, OVW)`	其中： FA： 跟随轴 LA: 引导轴 编号： 已保存曲线图表的编号 OVW: 允许使用修改后的曲线图表覆盖某个现有的耦合

关断轴向引导值耦合：

`...DO LEADOF(FA, LA)`	使用： FA:跟随轴 LA: 引导轴，选件
`... DO LEADOF(FA)`	缩短的形式，无引导轴说明

通过同步动作释放存取 RELEASE

为了释放某个需要耦合的轴以便通过同步动作进行存取，必须预先为需要耦合的跟随轴调用功能 RELEAS。

示例：

```
RELEASE(XKAN)

ID=1 every SR1==1 to LEADON(CACH,XKAN,1)
```

OVW=0 (默认值)

可以在没有新同步动作的情况下，不给某个现有的耦合规定新的曲线图表。 必须事先关断现有的耦合并且使用修改后的曲线图表编号再次启用，才可修改曲线图表。 这样来影响耦合的某个新同步动作。

使用 OVW=1 来修改现有耦合的曲线图表

使用 OVW=1 可以给某个现有的耦合规定一个新的曲线图表。 不会有新同步动作。 跟随轴尝试以尽可能快的方式跟踪通过新曲线图表设定的位置值。

飞剪举例

始终通过一个分割工具的工作区运动的带材，应该分割为相同长度的部分。

X 轴： 带材运动的轴。 WKS
X1-轴： 带材的加工轴，MKS
Y-轴： 剪断装置与带材在其中“共同运动”的轴

假设通过 PLC 来控制剪断工具的进给及其控制系统。 为了确定带材与分割工具的同步性，可以求算 PLC 接口的信号。

动作
启用耦合， LEADON
断开耦合， LEADOF
设定实际值， PRESETON

程序代码	注释
N100 R3=1500	; 一个待分割部件的长度
N200 R2=100000 R13=R2/300	
N300 R4=100000	
N400 R6=30	; Y 轴起始位置
N500 R1=1	; 带材轴的起始条件
N600 LEADOF(Y,X)	; 删除某个可能存在的耦合
N700 CTABDEF(Y,X,1,0)	; 图表定义
N800 X=30 Y=30	; 值组对
N900 X=R13 Y=R13	
N1000 X=2*R13 Y=30	
N1100 CTABEND	; 结束图表定义
N1200 PRESETON(X1,0)	; 用于开始的 PRESET
N1300 Y=R6 G0	; Y 轴起始位置，轴为线性轴
N1400 ID=1 WHENEVER $AA_IW[X]>$R3 DO PESETON(X1,0)	; 根据长度 R3 PRESET，分割之后新开始

程序代码	注释
N1500 RELEASE(Y)	
N1800 ID=6 EVERY $AA_IM[X]<10 DO LEADON(Y,X,1)	; 在 X<10 时，Y 通过表 1 耦合到 X
N1900 ID=10 EVERY $AA_IM[X]>$R3-30 DO EADOF(Y,X)	; >30 在运行的分割长度之前去除耦合
N2000 WAITP(X)	
N2100 ID=7 WHEN $R1==1 DO MOV[X]=1 FA[X]=$R4	; 带材轴始终处于运动状态
N2200 M30	

10.4.23 测量 (MEAWA, MEAC)

功能

与零件程序运动程序段中的应用相比，可以从同步动作中任意启用和关断测量功能。

关于测量的其它信息请参阅特殊行程指令“扩展测量功能”

句法

轴向测量，没有剩余行程删除

MEAWA[轴] = (模式，触发事件_1, ..._4)

连续测量，不带剩余行程删除

MEAC[轴] = (模式，测量存储器，触发事件_1, ..._4))

含义

程序代码	注释		
DO MEAWA	;	启用轴向测量	
DO MEAC	;	启用连续测量	
轴	;	为其进行测量的轴的名称	
模型	;	**十位**的数据 0：已激活的测量系统 测量系统的数量（视模式而定） 1：1．测量系统 2：2．测量系统 3：两个测量系统	**个位**的数据 0：取消测量任务 4 个以下可激活的触发事件 1：同时 2：先后 3：与 2 一样，但是起动时不监控触发事件 1
触发事件_1 到 _4	;	：上升脉冲沿，测头 1 -1：下降脉冲沿，测头 1，可选	

程序代码	注释
	2：上升脉冲沿，测头 2，可选 -2：下降脉冲沿，测头 2，可选
测量存储器	; FIFO-循环存储器的编号

10.4.24 初始化数组变量（SET，REP）

功能

在同步动作中可以初始化数组变量或者写入指定的值。

说明

仅在同步动作中可描述的变量才能如此。 因此，不得初始化机床数据。 可以规定轴变量的值为 NO_AXIS。

句法

```
DO FELD[n,m]=SET（<值 1>，<值 2>，...）
DO FELD[n,m]=REP（<值>）
```

含义

FELD[n,m]	编程的数组变址 赋予初值开始于编程的数组变址。对于 2 维数组，首先增加第 2 个变址。 对于轴索引，不进行该过程。
SET（<值 1>，<值 2>，...）	使用值列表初始化 将数组从编程的数组变址开始使用 SET 参数描述。有多少值被编程就有多少数组单元被赋值。 如果被编程的值超过现有的剩余数组单元，就会触发系统警报。
REP（<值>）	使用相同的值初始化 将数组从编程的数组变址开始直到数组末尾使用参数 REP 的（<值>）重复描述。

示例

程序代码	注释
WHEN TRUE DO SYG_IS[0]=REP(0)	; 结果：
WHEN TRUE DO SYG_IS[1]=SET(3,4,5)	; SYG_IS[0]=0
	SYG_IS[1]=3
	SYG_IS[2]=4
	SYG_IS[3]=5
	SYG_IS[4]=0

10.4.25 设置/删除等候标记（SETM，CLEARM）

功能

在同步动作中可以设置或者删除等候标记，以便（例如）在通道之间进行协调。

句法

DO SETM（<标记编号>）
DO CLEARM（<标记编号>）

含义

SETM	设置通道等候标记的指令 指令 SETM 可以在零件程序中和某个同步动作的动作程序段中写入。 该指令设置指令运行通道的（<标记编号>）。
CLEARM	删除通道等候标记的指令 指令 CLEARM 可以在零件程序中和某个同步动作的动作程序段中写入。 该指令删除指令运行通道的 （<标记编号>）。
<标记编号>	等候标记

10.4.26 故障应答 (SETAL)

功能

通过同步动作可以对故障应答编程。 此时查询状态变量并触发相应动作。

对故障的可能应答：

- 停止轴 (倍率=0)
- 设置报警

  使用 SETAL 可以由同步动作设置循环报警。
- 设置输出端
- 同步动作中所有可能的动作

句法

设置循环报警：

```
DO SETAL(<报警编号>)
```

含义

SETAL	用于设置循环报警的指令	
<报警编号>	报警编号	
	用户循环报警范围：	65000 至 69999

示例

程序代码	注释
ID=67 WHENEVER ($AA_IM[X1]-$AA_IM[X2])<4.567 DO $AA_OVR[X2]=0	; 如果轴 X1 和 X2 之间的安全距离太小，则轴 X2 停止。
ID=67 WHENEVER ($AA_IM[X1]-$AA_IM[X2])<4.567 DO SETAL(65000)	; 如果轴 X1 和 X2 之间的安全距离太小，则输出报警 65000 。

10.4.27 运行到固定挡块（FXS, FXST, FXSW, FOCON, FOCOF）

功能

功能"运行到固定挡块"的指令通过零件程序指令 FXS, FXST 和 FXSW 在同步动作/工艺循环中编程。

不用运动就可以激活这些指令，扭矩立即被限制。 一旦轴运动通过设定点，就会激活限制停止监视器。

以限制的力矩/力（FOC）运行

该功能允许通过同步动作随时改变力矩/力，并且能以模态方式或者根据程序段激活。

句法

```
FXS[<轴>]
FXST[<轴>]
FXSW[<轴>]
FOCON[<轴>]
FOCOF[<轴>]
```

含义

FXS	仅在带数字驱动 (VSA, HSA, HLA)的系统中选择
FXST	改变夹紧扭矩 FXST
FXSW	改变监控窗口 FXSW
FOCON	激活模态有效的扭矩/力－极限
FOCOF	取消扭矩/力－极限
<轴>	轴名称 允许： • 几何轴名称 • 通道轴名称 • 机床轴名称

说明

一个选择仅允许进行一次。

示例

示例 1： 运行到固定挡块 (FXS)，通过同步动作触发

程序代码	注释
Y-轴：	; 静态同步动作
激活：	
N10 IDS=1 WHENEVER (($R1==1) AND $AA_FXS[y]==0)) D $R1=0 FXS[Y]=1 FXST[Y]=10 FA[Y]=200 POS[Y]=150	
	; 通过设置 $R1=1 激活轴 Y 的 FXS，这将有效减少扭矩到 10% 并向挡块方向开始运动。
N11 IDS=2 WHENEVER ($AA_FXS[Y]==4) DO FXST[Y]=30	; 只要识别到挡块 ($AA_FXS[Y]==4)，就将力矩增大到 30%。
N12 IDS=3 WHENEVER ($AA_FXS[Y]==1) DO FXST[Y]=$R0	; 在达到挡块后根据 R0 控制扭矩。
N13 IDS=4 WHENEVER (($R3==1) AND $AA_FXS[Y]==1)) DO FXS[Y]=0 FA[Y]=1000 POS[Y]=0	
	; 取消选择和 R3 有关并返回。
N20 FXS[Y]=0 G0 G90 X0 Y0	; 正常的程序运行：
N30 RELEASE(Y)	; 释放轴 Y 用于同步动作中的运动。
N40 G1 F1000 X100	; 运动另一个轴。
N50 ...	
N60 GET(Y)	; 将轴 Y 重新纳入轨迹组合中

示例 2： 激活力矩/力限制 (FOC)

程序代码	注释
N10 FOCON[X]	; 模态方式激活极限
N20 X100 Y200 FXST[X]=15	; X 以降低的力矩（15%）运行
N30 FXST[X]=75 X20	; 将力矩改变成 75%，X 以这个受到限制的力矩运动
N40 FOCOF[X]	; 关断力矩极限。

其它信息

多次选择

如果由于错误的编程，再次在激活后调用该功能（FXS[<轴>]=1），将触发如下报警：

轴 20092“运行到固定挡块仍然激活”

要么以条件方式询问 $AA_FXS[] 或者询问某个自身的标记 (这里是 R1) 的编程可避免多次激活“零件程序碎片”功能。

程序代码

```
N10 R1=0
```

程序代码

```
N20 IDS=1 WHENEVER ($R1==0 AND
$AA_IW[AX3] > 7) DO R1=1 FXST[AX1]=12
```

与程序段有关的同步动作

在接通返回运行期间，通过编程一个程序段相关的同步动作可以运行到固定挡块。

示例：

程序代码	注释
N10 G0 G90 X0 Y0	
N20 WHEN $AA_IW[X] > 17 DO FXS[X]=1	; 当 X 到达某个大于 17 mm 的位置时激活 FXS。
N30 G1 F200 X100 Y110	

静态同步动作和与程序段有关的同步动作

在静态同步动作和与程序段有关的同步动作中，可以使用同样的指令 FXS, FXST 和 FXSW ，如同在标准零件程序执行过程中一样。所分配的值可以通过一个计算产生。

10.4.28 确定同步动作中的轨迹切线角

功能

在同步动作中可以读取的系统变量 $AC_TANEB (正切角，在程序段结束处) 用来计算当前程序段结束处中的轨迹切线和已编程跟随程序段开始处的轨迹切线之间的夹角。

参数

切线角始终在范围 0.0 到 180.0 度范围内给出正值。如果在主过程中不存在跟随程序段，就输出
角度 -180.0 度。

不应当给系统所生成的程序段读取系统变量 $AC_TANEB 。系统变量 $AC_BLOCKTYPE 可用来区别是否与某个已编程的程序段（主程序段）有关。

示例

```
ID=2 EVERY $AC_BLOCKTYPE==0 DO $SR1 = $AC_TANEB
```

10.4.29 确定当前的倍率

功能

当前的倍率

（NC 部分）可以用系统变量：

$AA_OVR 轴向修调率

$AC_OVR 轨迹修调率

在同步动作中读写。

由 PLC 给定的倍率用于读出：

$AA_PLC_OVR 轴向修调率

$AC_PLC_OVR 轨迹修调率

。

最后生成的倍率

用于读出系统变量中同步动作：

$AA_TOTAL_OVR 轴向修调率

$AC_TOTAL_OVR 轨迹修调率

。

得出的合成修调率为：

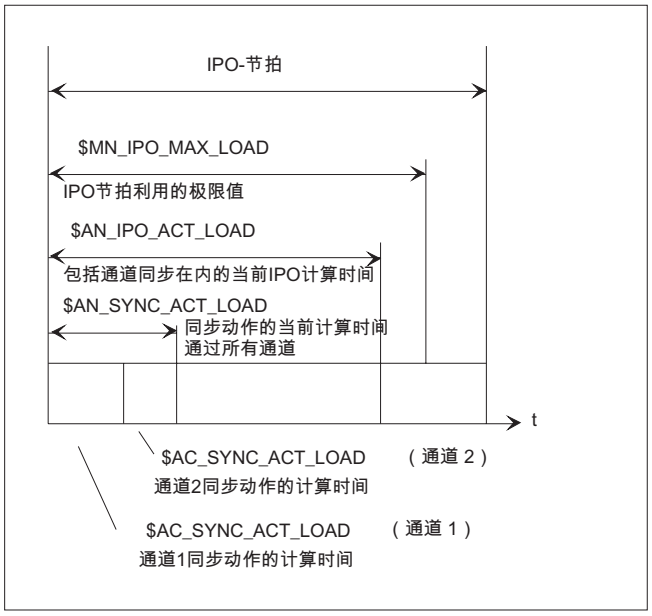
$AA_OVR * $AA_PLC_OVR 或者

$AC_OVR * $AC_PLC_OVR

10.4.30 通过同步动作的时间占用计算负荷

功能

在一个插补节拍中不仅要译出同步动作，而且也必须要由 NC 计算出运动。 利用后面介绍的系统变量，同步动作可以通过插补节拍中同步动作的实际时间分量和位置调节器的计算时间进行了解。



含义

仅当机床数据 $MN_IPO_MAX_LOAD 大于 0 时，这些变量才会有有效值。 其它情况下，总是净计算时间规定 SINUMERIK powerline 和 solution line Systeme 的变量，此时，将不再考虑通过 HMI 生成的中断。 净计算时间由下列各项计算得出：

- 同步动作时间，
- 位置调节时间，以及
- 剩余 IPO 计算时间，不带 HMI 限制的中断

这些系统变量始终含有上一个 IPO 节拍的值

$AN_IPO_ACT_LOAD 当前的 IPO 计算时间（包括所有通道中的同步动作）。

$AN_IPO_MAX_LOAD 最长的 IPO 计算时间（包括所有通道中的同步动作）。

$AN_IPO_MIN_LOAD 最短的 IPO 计算时间（包括所有通道中的同步动作）。

$AN_IPO_LOAD_PERCENT	当前的 IPO 计算时间，与 IPO 节拍比较（%）。
$AN_SYNC_ACT_LOAD	当前的计算时间，用于所有通道的同步动作
$AN_SYNC_MAX_LOAD	最长的计算时间，用于所有通道的同步动作
$AN_SYNC_TO_IPO	在总的 IPO 计算时间中（通过所有通道）总的同步动作的百分比。
$AC_SYNC_ACT_LOAD	通道中同步动作的当前计算时间
$AC_SYNC_MAX_LOAD	通道中同步动作的最长的计算时间
$AC_SYNC_AVERAGE_LOAD	通道中同步动作的平均的计算时间
$AN_SERVO_ACT_LOAD	位置调节器的当前的计算时间
$AN_SERVO_MAX_LOAD	位置调节器的最长的计算时间
$AN_SERVO_MIN_LOAD	位置调节器的最短的计算时间

传达过载信息的变量：

通过机床数据 $MN_IPO_MAX_LOAD 来设置应从哪一个 IPO 净计算时间（IPO 节拍的百分比）起将系统变量 $AN_IPO_LOAD_LIMIT 设定成 TRUE。 如果当前的负载又再次低于极限，则该变量再次置为 FALSE（假)。 如果机床数据为 0，就解除所有的诊断功能。

通过分析 $AN_IPO_LOAD_LIMIT 用户可以自己确定一种避免超越平面的方法。

10.5 工艺循环

功能

作为同步动作中的动作，也可以调用仅由功能组成的程序，这些功能也可以允许作为同步动作中的动作。 由此构成的程序称作工艺循环。

工艺循环可以作为子程序存储在控制系统中。

在一个通道中可以并行处理几个工艺循环或者动作。

编程

工艺循环编程有下列规定：

- 程序末尾编程有 M02/M17/M30/RET。
- 在一个程序级内，在某个周期中没有等待循环的情况下可以处理所有在 ICYCOF 中规定的动作。
- 每个同步动作可以最多连续询问 8 个工艺循环。
- 也可以在逐段有效同步动作中进行工艺循环。
- 可以编程 IF 控制结构和跳转指令 GOTO, GOTOF 和 GOTOB。
- 对于带有 DEF 和 DEFINE 指令得程序段适用于：
  - DEF 和 DEFINE 指令在工艺循环中省略。
  - 这会导致在句法不正确或不完整时发出报警提示。
  - 没有省略报警提示的情况下无需自行设立。
  - 通过赋值作为完整的零件程序循环考虑。

参数传递

可以在工艺循环上传输数据。 考虑作为形式参数“数值调用”分配的简单数据类型和在调用工艺循环时有效的标准设置。 它们是：

- 没有编程传输参数情况下的编程标准值。
- 执行带有初始值得标准参数。
- 用一个标准值分配未初始化的实际参数。

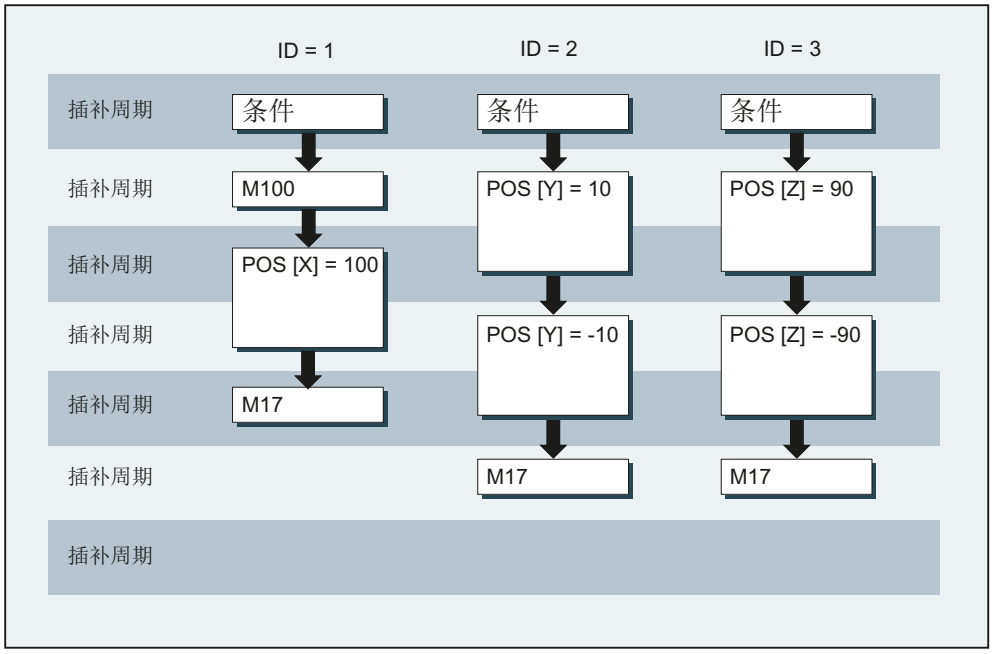
工作流程

一旦条件满足，则启动工艺循环。 在一个单独的 IPO 周期中处理工艺循环的每一行。 在定位轴中，需要几个 IPO 周期用于执行。 其它的功能执行一个周期。 在工艺循环中，程序段按顺序执行。

如果在同一个插补周期中调用几个动作，这几个动作相互之间排斥，则激活同步动作中用更高 ID 号调用的动作。

示例

示例 1： 通过置位数字输入来启动轴程序。



主程序：

程序代码	注释
`ID=1 EVERY $A_IN[1]==1 DO ACHSE_X`	`; 输入 1 为 1，启动轴程序 ACHSE_X。`
`ID=2 EVERY $A_IN[2]==1 DO ACHSE_Y`	`; 输入 2 为 1，启动轴程序 ACHSE_Y。`
`ID=3 EVERY $A_IN[3]==1 DO ACHSE_Z`	`; 输入 3 为 1，启动轴程序 ACHSE_Z。`
`M30`	

轴程序 ACHSE_X:

程序代码

```
M100
POS[X]=100 FA[X]=300
M17
```

轴程序 ACHSE_Y:

程序代码

```
POS[Y]=10 FA[Y]=200
POS [Y] = -10
M17
```

轴程序 ACHSE_Z:

程序代码

```
POS[Z]=90 FA[Z]=250
POS [Z] = -90
M17
```

示例 2： 工艺循环中不同的程序序列

程序代码

```
PROC CYCLE
N10 DEF REAL 值 =12.3
N15 DEFINE ABC AS G01
```

在不发出报警和不设立变量和宏的情况下忽略两个程序段。

程序代码

```
PROC CYCLE
N10 DEF REAL
N15 DEFINE ABC G01
```

两个程序段将导致 NC 报警，因为未正确写入句法。

程序代码

```
PROC CYCLE
N10 DEF AXIS 轴 1=XX2
```

如果未识别到轴 XX2，则发出报警 12080。 否则将在不发出报警和不设立变量情况下忽略该程序段。

程序代码

```
PROC CYCLE
N10 DEF AXIS 轴 1
N15 G01 X100 F1000
N20 DEF REAL 值 1
```

程序段 N20 始终会导致报警 14500，因为在第 1 个程序指令后不应跟随任何定义指令。

10.5.1 上下文变量（$P_TECCYCLE）

功能

借助变量 $P_TECCYCLE 可以区分同步动作程序和预处理程序中的程序。 由此可以处理句法正确写入的程序段或程序序列，也可以选择性地作为零件程序循环处理。

解释上下文变量

系统变量 $P_TECCYCLE 使得上下文专用的编译程序在工艺循环中能够由程序部分控制，如果：

```
IF $P_TECCYCLE==TRUE
...                          ; 同步动作中用于工艺循环的程序序列。
ELSE
...                          ; 用于零件程序循环的程序序列。
ENDIF
```

说明

一个带错误或者未许可的程序句法、带无法识别的赋值的程序段也会在零件程序循环中导致一个报警提示。

示例

工艺循环中带询问 $P_TECCYCLE 的程序序列：

程序代码	注释
PROC CYCLE	
N10 DEF REAL 值 1	; 在工艺循环中省略。
N15 G01 X100 F1000	
N20 IF $P_TECCYCLE==TRUE	
...	; 工艺循环的程序序列（不带变量值 1）。
N30 ELSE	
...	; 零件程序循环的程序序列（存在变量值 1）。
N40 ENDIF	

10.5.2 Call-by-Value 值调用参数

功能

工艺循环可以用 Call-by-Valu 值调用参数定义。 简单的数据类型，如 INT、REAL、CHAR、STRING、AXIS 和 BOOL 可以作为参数。

说明

采用值调用方法传递的形式参数可以不是数组。

实际参数也可以由默认参数组成（参见“缺省参数初始化 (页 644)”）。

句法

```
ID=1 WHEN $AA_IW[X]>50 DO TEC(IVAL,RVAL,,SVAL,AVAL)
```

在未初始化的实际参数中分配一个缺省值：

```
ID=1 WHEN $AA_IW[X]>50 DO TEC(IVAL,RVAL,,SYG_SS[0],AVAL)
```

10.5.3 缺省参数初始化

功能

也可以在 PROC 指令中用一个参数值规定缺省参数。

句法

在工艺循环中分配缺省参数：

```
PROC TEC（INT IVAL=1，REAL RVAL=1.0，CHAR CVAL='A'，STRING[10]
SVAL="ABC"，AXIS AVAL=X，BOOL BVAL=TRUE）
```

如果实际参数由一个缺省参数组成，则从 PROC 指令中分配初始值。 这不仅适用于零件程序中，也适用于同步动作中。

示例

程序代码	注释
TEC (IVAL, RVAL, SVAL, AVAL)	; 初始值适用于 CVAL 和 BVAL 中

10.5.4 控制工艺循环的处理工作（ICYCOF，ICYCON）

功能

ICYCOF 和 ICYCON 语言指令用于控制工艺循环的时间处理。

使用 ICYCOF 仅在一个插补周期中处理工艺循环的所有程序段。 所有执行时需要多个周期的动作，在 ICYCOF 下产生平行的处理过程。

应用

在 ICYCON 下，指令轴运动延迟工艺循环处理。 如果无需如此，则可以用 ICYCOF 在无等待时间的情况下，在一个插补周期中处理所有的动作。

句法

以下适用工艺循环处理：

ICYCON，根据 ICYCON 在一个单独的 IPO 周期中处理工艺循环的每个程序段

ICYCOF 根据 ICYCOF 在一个 IPO 周期中处理所有工艺循环的连续程序段

说明

语言指令 ICYCON 和 ICYCOF 仅在程序级内部有效。 在零件程序中简单省略无反应的这两个指令。

ICYCOF 处理方式举例

程序代码	注释
插补周期	; PROC TECHNOCYC
1.	; $R1=1
2.25	; POS[X]=100
26.	; ICYCOF
26.	; $R1=2
26.	; $R2=$R1+1
26.	; POS[X]=110
26.	; $R3=3
26.	; RET

10.5.5 工艺循环级联

功能

可以最多处理串联的 8 个工艺循环。 由此可以在一个同步动作中编程多个工艺循环。

句法

```
ID=1 WHEN $AA_IW[X]>50 DO TEC1（$R1）TEC2 TEC3（X）
```

加工顺序

根据上面规定的编程，由左向右（级联）处理工艺循环。 如果在 ICYCON 模式中处理循环，则延迟所有随后的处理工作。 出现的报警不会中断随后的动作。

10.5.6 逐段同步动作中的工艺循环

功能

也可以在逐段同步动作中进行工艺循环。

如果工艺循环的处理时间长于所属程序段的处理时间，则工艺循环在程序段切换时中断。

说明
工艺循环不会阻止程序段切换。

10.5.7 控制结构 (IF)

功能

同步动作中的 IF 控制结构可以用于工艺循环过程顺序中的分支。

句法

```
IF <条件>
$R1=1
[ELSE] 可选
$R1=0
ENDIF
```

10.5.8 跳转指令（GOTO、GOTOF、GOTOB）

功能

在工艺循环中可以有跳转指令 GOTO、GOTOF、GOTOB。 规定的标签必须在子程序中，这样就不会中断报警。

说明
标签和程序段号仅允许为常量。

句法

无条件的跳转

GOTO 标签，程序段号
GOTOF 标签，程序段号
GOTOB 标签，程序段号

跳转指令和跳转行

GOTO	首先向前跳转，接着向后跳转
GOTOF	向前跳转
GOTOB	向后跳转
标签：	跳过标记
程序段号码	到该程序段的跳转目标
N100	程序段号为副程序段
:100	程序段号为主程序段

10.5.9 禁用，释放，复位 (LOCK, UNLOCK, RESET)

功能

工艺循环过程可以通过一个模态同步动作禁用，再次释放或复位。

句法

```
LOCK(<n1>,<n2>,...)
UNLOCK(<n1>,<n2>,...)
RESET(<n1>,<n2>,...)
```

含义

LOCK	用于禁用同步动作的指令 中断激活的动作。
UNLOCK	用于释放同步动作的指令

RESET	用于复位工艺循环的指令
<n1>,<n2>,...	要禁用，释放或复位的同步动作或工艺循环识别号码。

联锁同步动作

具有识别号 **n=1 ... 64** 模态同步动作可以由 PLC 联锁。 因此相关的条件不再计算，相应的功能在 NCK 中禁止执行。

使用一个 PLC 的接口信号可以禁止所有同步动作。

说明

一个编程的同步动作按照标准激活，可以通过机床数据防止改写/禁止。
用于不得对机床制造商所设定的同步动作进行干预。

示例

示例 1： 禁用同步动作 (LOCK)

程序代码

```
N100 ID=1 WHENEVER $A_IN[1]==1 DO M130
...
N200 ID=2 WHENEVER $A_IN[2]==1 DO LOCK(1)
```

示例 2： 释放同步动作 (UNLOCK)

程序代码

```
N100 ID=1 WHENEVER $A_IN[1]==1 DO M130
...
N200 ID=2 WHENEVER $A_IN[2]==1 DO LOCK(1)
...
N250 ID=3 WHENEVER $A_IN[3]==1 DO UNLOCK(1)
```

示例 3： 中断工艺循环(RESET)

程序代码

```
N100 ID=1 WHENEVER $A_IN[1]==1 DO M130
...
N200 ID=2 WHENEVER $A_IN[2]==1 DO RESET(1)
```

10.6 删除同步动作 (CANCEL)

功能

使用指令 CANCEL 可以从零件程序中取消某个模态或者静态有效的同步动作。

如果一个同步动作停止，但是同时由此激活的定位轴运动却仍有效，则结束定位轴运行。如果这不是所期望的，可以在执行 CANCEL 指令之前使用轴剩余行程删除来使轴运动停止。

句法

```
CANCEL(<n1>,<n2>,...)
```

含义

CANCEL:	用于删除编程的同步动作指令
<n1>,<n2>,...:	要删除同步动作的识别号 **提示：** 没有指定识别号时，会删除**所有**模态/静态同步动作。

示例

示例 1： 中断同步动作

程序代码	注释
N100 ID=2 WHENEVER $A_IN[1]==1 DO M130	
...	
N200 CANCEL(2)	; 删除模态同步动作编号 2。

示例 2： 在中断同步动作前删除剩余行程

程序代码	注释
N100 ID=17 EVERY $A_IN[3]==1 DO POS[X]=15 FA[X]=1500	; 启动定位轴运行。
...	
N190 WHEN ... DO DELDTG(X)	; 结束定位轴运行。
N200 CANCEL(17)	; 删除模态同步动作编号 17。

10.7 特定运行状态下的控制属性

上电

当执行上电时，原则上没有同步动作激活。 可以用某个被 PLC 启动的异步子程序（ASUP）激活静态同步动作。

运行方式切换

在运行方式切换后，使用关键字 IDS 所激活的同步动作仍保持生效。所有其余同步动作在切换运行方式时生效（例如定位轴），当重新定位和恢复到自动运行方式时再次生效。

RESET

使用“NC 复位”会结束所有逐段生效和模态生效的同步动作。 静态的同步动作仍然有效。由它们可以启动新的动作。 如果在复位时有一个指令轴运动有效，则中断该动作。 已经执行的 WHEN 类型的同步动作在复位之后将不再被处理。

复位之后的属性		
同步动作/ 工艺循环	**模态/逐段方式**	**静态 (IDS)**
	生效的动作被停止，同步动作被删除	生效的动作被停止，工艺循环被复位
进给轴/ 定位中的主轴	运动被停止	运动被停止
转速控制的主轴	$MA_SPIND_ACTIVE_AFTER_RESET==1: 主轴保持有效 $MA_SPIND_ACTIVE_AFTER_RESET==0: 主轴停止。	
引导值耦合	$MC_RESET_MODE_MASK, Bit13 == 1: 引导值耦合保持有效 $MC_RESET_MODE_MASK, Bit13 == 0: 触发引导值耦合	
测量过程	由同步动作启动的测量过程被停止。	由静态同步动作启动的测量过程被停止。

NC 停止

静态 同步动作在"NC 停止"时保持生效。 由静态同步动作启动的运动没有被停止。 属于激活的程序段的**程序局部** 同步动作保持激活，由此而起动的运动被中断。

程序结束

程序结束和同步动作相互之间没有影响。 运行的同步动作在程序结束之后也被结束。 在 M30 程序段中有效的同步动作在 M30 程序段中仍然有效。 如果这不是所期望的，就必须在程序结束之前使用 CANCEL 中断同步动作。

程序结束之后的属性		
同步动作/ 工艺循环	**模态和逐段方式 →被中断**	**静态 (IDS) → 保持不变**
进给轴/ 定位中的主轴	M30 被延迟，直至进给轴/主轴停止	运动继续进行。
转速控制的主轴	程序结束： $MA_SPIND_ACTIVE_AFTER_RESET==1: 主轴保持有效 $MA_SPIND_ACTIVE_AFTER_RESET==0: 主轴停止 在运行方式切换时主轴保持有效。	主轴保持有效。
引导值耦合	$MC_RESET_MODE_MASK, Bit13 == 1: 引导值耦合保持有效 $MC_RESET_MODE_MASK，Bit13 == 0： 触发引导值耦合	由静态同步动作启动的耦合保持不变。
测量过程	由同步动作启动的测量过程被停止。	由静态同步动作启动的测量过程保持有效。

程序段搜索

程序段搜索过程中会集合同步动作并且在“NC 启动”时进行分析，同样相应的动作也会被启动。 在程序段搜索期间，静态同步动作也有效。如果在搜索程序段时找到使用 FCTDEF 编程的多项式系数，则这些系数就会立即有效。

通过异步子程序 ASUP 中断程序

ASUP 起始：

模态和静态运动同步动作保持不变且在异步子程序中也保持有效。

ASUP 结束：

如果没有用 REPOS 继续异步子程序，则在异步子程序中修改的模态和静态运动同步动作在主程序中继续有效。

重新定位（REPOS）

在结束重新定位（REPOS）之后，被中断程序段中的有效同步动作会重新激活。 在异步子程序中修改的模态同步动作在 REPOS 后，处理其他程序段时不再生效。

用 FCTDEF 编程的多项式系数不受异步子程序和 REPOS 影响。 不管它是在哪里编程的，即使在执行 REPOS 之后，也可随时在异步子程序和主程序中使用。

报警时的属性

如果一个包含运动停止的报警被激活，则由同步动作启动的进给轴运行和主轴运行将被停止。 所有其他动作如置位输出等，将继续执行。

如果一个同步动作自身引发了报警，将会停止执行，不会继续执行该同步动作的其他动作。 如果同步动作模态生效，在下一个插补周期中不再执行该动作。 该报警只会触发一次。 所有其他的同步动作继续执行。

会导致编译器停止的报警在执行完预编码的程序段后才生效。

如果一个工艺循环引发了一个带运行停止的报警，则不再处理该工艺循环。

11 摆动

11.1 异步摆动(OS, OSP1, OSP2, OST1, OST2, OSCTRL, OSNSC, OSE, OSB)

功能

一个摆动轴在两个换向点 1 和 2 之间以给定的进给来回摆动，直至摆动运动关断。

在摆动运行期间其它的轴可以任意插补。 通过一个轨迹运动或者用一个定位轴，可以达到一个连续进给。 此时在摆动运动和进给运动之间**不存在关系** 。

异步摆动特性

- 在特定的轴上，异步摆动会超出程序段范围生效。
- 通过零件程序可以确保摆动运动和程序段同时启动。
- 几个轴的共同插补和摆动距离的叠加无法进行。

编程

通过以下指令可以从零件程序中启用并控制异步摆动，使它和 NC 程序的执行一致。

编程的值会在主运行中、和程序段同步地输入到相应的设定数据中，在下一次修改前一直保持有效。

句法

```
OSP1[<轴>]=<值> OSP2[<轴>]=<值>
OST1[<轴>]=<值> OST2[<轴>]=<值>
FA[<轴>]=<值>
OSCTRL[<轴>]=(<设置选项>,<复位选项>)
OSNSC[<轴>]=<值>
OSE[<轴>]=<值>
OSB[<轴>]=<值>
OS[<轴>]=1
OS[<轴>]=0
```

含义

<轴>　　摆动轴的名称

OS　　启动/关闭摆动

值：　1　**启动**摆动

　　0　**关闭**摆动

OSP1　　确定换向点 1 的位置

OSP2　　确定换向点 2 的位置

提示：
如果处于一个增量运行中，则按照上次 NC 程序中写入的相应换向点增量计算出该位置。

OST1　　确定换向点 1 的 停止时间，单位秒

OST2　　确定换向点 2 的 停止时间，单位秒

<值>：　-2　继续插补，无需等待准停

　　-1　等待粗准停

　　0　等待精准停

　　>0　等待精准停，并且接着等候指定的停止时间

提示：
停止时间的单位与通过 G4 编程的停止时间相同。

FA　　确定进给速度

进给速度指定位轴的定义进给速度。 如果没有定义进给速度，则机床数据中存储的值生效。

OSCTRL　　指定设置选项和复位选项

选项 0 - 3 规定了取消摆动时换向点上的属性。 可以选择 0 - 3 其中的一个类型。 可以根据需要结合所选类型进行其他设置。 通过正号(+)将多个选项相加在一起。

<值>：　0　取消摆动运动时停止在下一个换向点处（预设置）

提示：
只允许通过值 1 和 2 复位。

　　1　取消摆动运动时停止在换向点 1 处

　　2　取消摆动运动时停止在换向点 2 处

　　3　如果没有编程修光行程，则在取消摆动运动时不返回换向点。

4 在修光后运行到终点位置

8 由于删除剩余行程而中断了摆动运动后，会紧接着执行修光，并运行到终点位置。

16 由于删除剩余行程而中断了摆动运动后，会如同取消摆动一样，运行到相应的换向位置。

32 修改的进给率从下一个换向点开始才有效

64 FA 等于 0, FA = 0: 位移叠加有效

FA 不等于 0, FA <> 0: 速度叠加有效

128 在回转轴 DC 时（最短的行程）

256 两次修光。（标准） 1 = 单次修光。

512 首先运行到起始位置

OSNSC 确定修光次数

OSE 确定 WCS 中的终点位置，在取消摆动时将运行到该位置

提示：

当编程 OSE 时，OSCTRL 的选项 4 会隐式有效。

OSB 确定 WCS 中的起始位置，在开始摆动前轴会运行到该位置

首先到达起始位置，然后是换向点 1。 如果起始位置和换向点 1 相同，则接着运行到换向点 2。 即使起始位置和换向点 1 相同，在达到起始位置时也不会等待“停止时间”，而是等待精准停。 即满足设置的准停条件。

提示：

必须在设定数据 SD43770 $SA_OSCILL_CTRL_MASK 中置位数位 9，才能够运行到起始位置。

示例

示例 1： 摆动轴在两个换向点之间摆动

摆动轴 Z 应该在位置 10 和 100 之间摆动。 换向点 1 应以精准停逼近，换向点 2 以粗准停逼近。 摆动轴的进给为 250。 在加工结束处应当执行 3 次修光行程，并且使得摆动轴达到终点位置 200。 进给轴的进给是 1，在 X 方向的进给在位置 15 处结束。

程序代码	注释
WAITP(X,Y,Z)	; 起始位置。
G0 X100 Y100 Z100	; 转换到定位轴运行方式。
WAITP(X,Z)	
OSP1[Z]=10 OSP2[Z]=100	; 换向点 1，换向点 2。

程序代码	注释
OSE[Z]=200	; 终点位置。
OST1[Z]=0 OST2[Z]=-1	; 换向点 1 的停止时间：精准停 ; 换向点 2 的停止时间： 粗准停
FA[Z]=250 FA[X]=1	; 摆动轴进给，进给轴进给。
OSCTRL[Z]=(4,0)	; 设置选项。
OSNSC[Z]=3	; 3 次修光行程。
OS[Z]=1	; 启动摆动。
WHEN $A_IN[3]==TRUE DO DELDTG(X)	; 剩余行程删除。
POS[X]=15	; X 轴初始位置。
POS[X]=50	X 轴终点位置。
OS[Z]=0	; 停止摆动。
M30	

说明

也可以在一个程序段中编程指令串 OSP1[Z]=...~OSNCS[Z]=...。

示例 2： 带换向位置在线修改的摆动

异步摆动所需的设定数据可以在零件程序中设置。

如果在零件程序中直接定义设定数据，则修改在预处理时就已经生效。 可以通过预处理停止 STOPRE 来实现同步特性。

程序代码	注释
$SA_OSCILL_REVERSE_POS1[Z]=-10	
$SA_OSCILL_REVERSE_POS2[Z]=10	
G0 X0 Z0	
WAITP(Z)	
ID=1 WHENEVER $AA_IM[Z] < $$AA_OSCILL_REVERSE_POS1[Z] DO $AA_OVR[X]=0	; 如果摆动轴的实际值超出换向点，则进给轴停止。
ID=2 WHENEVER $AA_IM[Z] < $$AA_OSCILL_REVERSE_POS2[Z] DO $AA_OVR[X]=0	
OS[Z]=1 FA[X]=1000 POS[X]=40	; 激活摆动。
OS[Z]=0	; 取消摆动。
M30	

其它信息

摆动轴

对于摆动轴而言：

- 每个轴都可以作为摆动轴使用。
- 而且可以同时有几个摆动轴有效（最大为 定位轴个数）。
- 对于摆动轴而言，直线插补 G1 始终有效，而与程序中当前有效的 G 指令无关。

摆动轴可以作为：

- 动态坐标转换的输入轴
- 使用龙门轴和耦合轴时的引导轴
- 运行：
  - 没有急动限制 (BRISK)

    或者
  - 有急动限制 (SOFT)

    或者
  - 有曲折的加速特性曲线（如同定位轴）

摆动换向点

在确定摆动位置时必须考虑当前的偏移：

- 绝对尺寸

  OSP1[Z]=<值>

  换向点位置 = 偏移 + 编程值之和
- 相对尺寸

  OSP1[Z]=IC(<值>)

  换向点位置 = 换向点 1 + 编程值

示例：

程序代码

```
N10 OSP1[Z] = 100 OSP2[Z] = 110
...
...
N40 OSP1[Z] = IC(3)
```

WAITP

如果要用几何轴进行摆动，则必须使用 WAITP 使能该轴进行摆动。

在摆动结束之后，用 WAITP 指令把摆动轴再次定义为定位轴，恢复正常使用。

带有运动同步动作和停止时间的摆动

在已设置的停止时间届满之后，就会在摆动时发生内部程序段转换（可在轴的新剩余行程上看出来）。 在转换程序段时检查关闭功能。 此时会根据已设置的运动过程控制设置(OSCTRL)来确定关闭功能。 *可通过进给修调率来调节这种时间特性。*

在启动修光行程或者向终点位置运动之前，有时还会执行一次摆动行程。*此时会产生关闭特性发生变化的印象。但实际上不是这种情况。*

11.2 由同步动作控制的摆动(OSCILL)

功能

就这种摆动方式而言，仅允许在返回点上或者在定义的返回范围内有进给运动。

根据具体的要求，可以在进给期间

- 继续执行摆动运动，或者
- 停止摆动运动，直至进给完全执行。

句法

1. **确定摆动参数**
2. **定义运动同步动作**
3. **分配轴，确定进给**

含义

OSP1[<摆动轴>]=	换向点 1 的位置
OSP2[<摆动轴>]=	换向点 2 的位置
OST1[<摆动轴>]=	在换向点 1 处的保持时间，单位秒
OST2[<摆动轴>]=	在换向点 2 处的保持时间，单位秒
FA[<摆动轴>]=	摆动轴的进给率
OSCTRL[<摆动轴>]=	设定或取消选件
OSNSC[<摆动轴>]=	修光次数
OSE[<摆动轴>]=	终点位置
WAITP(<摆动轴>)	使能用于摆动的轴

轴分配，进给

```
OSCILL[<摆动轴>]=(<进给轴 1>,<进给轴 2>,<进给轴 3>)
POSP[<进给轴>]=(<终点位置>,<分段长度>,<方式>)
```

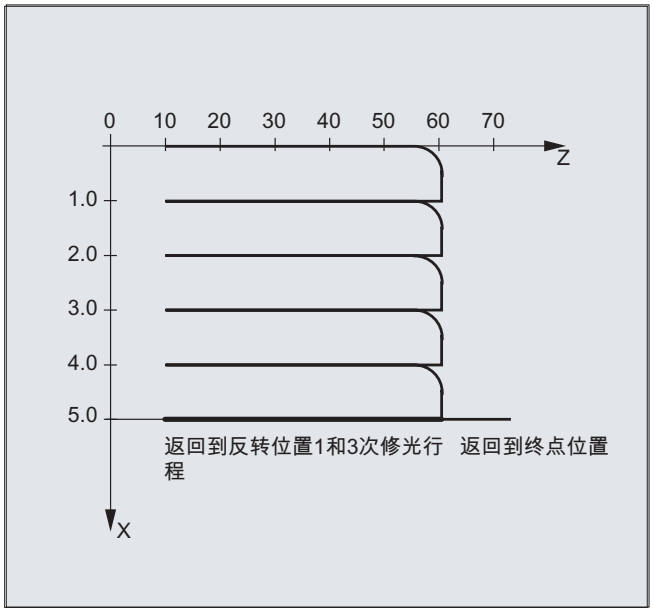
OSCILL:	给摆动轴分配进给轴
POSP：	确定总进给和分段进给（参见文件和程序管理一章）
终点位置:	在所有分段进给结束之后的进给轴的最终位置
分段长度:	换向点/换向区处的分段长度
方式：	总进给运动分为若干个分段进给
	= 两个同样大小的剩余步距（缺省设置）;
	= 所有分段进给同样大

运动同步动作

WHEN… … DO	当…, 则…
WHENEVER … DO	始终当…, 则…

示例

在换向点 1 上，无进给运动。 在换向点 2 上，在间隔换向点 2 ii2 处便应开始进给 ，并且在换向点处摆动轴不等待分段进给的结束。 轴 Z 为摆动轴且轴 X 为进给轴。



1. 摆动的参数

程序代码		注释
DEF INT ii2	;	定义变量，用于换向区 2
OSP1[Z]=10 OSP2[Z]=60	;	定义换向点 1 和 2
OST1[Z]=0 OST2[Z]=0	;	换向点 1： 精准停； 换向点 2： 精准停；
FA[Z]=150 FA[X]=0.5	;	摆动轴 Z 的进给率，进给轴 X 的进给率
OSCTRL[Z]=(2+8+16,1)	;	在换向点 2 处取消摆动运动；在 RWL 之后修光并返回终点位置；在 RWL 之后返回相应的换向位置
OSNC[Z]=3	;	修光行程
OSE[Z]=70	;	终点位置＝70
ii2=2	;	设定换向区
WAITP(Z)	;	允许的摆动，用于 Z 轴

2. 运动同步动作

程序代码		注释
WHENEVER $AA_IM[Z]<$SA_OSCILL_REVERSE_POS2[Z] DO -> $AA_OVR[X]=0 $AC_MARKER[0]=0	;	总是当 MCS 中摆动轴 Z 的当前位置小于换向区 2 的起点时，进给轴 X 的轴向倍率设为 0%，带有索引 0 的标记器设为 0。
WHENEVER $AA_IM[Z]>=$SA_OSCILL_REVERSE_POS2[Z] DO $AA_OVR[Z]=0	;	总是当 MCS 中摆动轴 Z 的当前位置大于等于换向位置 2 时，摆动轴 Z 的轴向倍率设为 0%。
WHENEVER $AA_DTEPW[X]==0 DO $AC_MARKER[0]=1	;	总是当分段进给剩余行程等于 0 时，索引为 0 的标志器设为 1。
WHENEVER $AC_MARKER[0]==1 DO $AA_OVR[X]=0 $AA_OVR[Z]=100	;	总是当索引为 0 的标志器等于 1 时，进给轴 X 的轴向倍率设为 0%。 从而防止过早进给（摆动轴 Z 尚未再次离开换向区 2，而进给轴 X 已准备就绪）。 将摆动轴 Z 的轴向倍率从 0%（第 2 个同步动作）重新设为 100%。

-> 必须在一个程序段中编程

3. 启动摆动

程序代码		注释
OSCILL[Z]=(X) POSP[X]=(5,1,1)	;	启动轴 轴 X 被作为进给轴分配给摆动轴。 轴 X 应以 1 为步距运动至终点位置 5。
M30	;	程序结束

说明

1. **确定摆动参数**
   在包含进给轴和摆动轴以及确定进给的运动程序段之前应确定摆动所需的参数（参见“异步摆动”）。
2. **确定运动同步动作**
   通过同步条件实现：
   抑制进给, 直到摆动轴在某个返回范围之内
   (ii1, ii2) 或者在某个返回点 (U1, U2) 上时为止。
   摆动运动 在进给过程中**停止**在返回点内。
   摆动运动 在结束部分进给之后重新**启动**. 确定
   启动下一个部分进给。
3. 确定**配置摆动轴和进给轴**以及 **最大进给和部分进给**。

确定摆动参数

配置摆动轴和进给轴： OSCILL

`OSCILL[摆动轴] = (横向进给轴 1, 横向进给轴 2, 横向进给轴 3)`

使用指令 `OSCILL` 实现轴配置和启动摆动运动。

一个摆动轴最多分配 3 个横向进给轴。

说明
在启动摆动之前必须已经确定轴性能的同步条件。

确定进给： POSP

`POSP[横向进给轴] = (终点位置, 分度长度, 方式)`

使用指令 `POSP` 向控制系统发送信息：

- 总的横向进给（通过终点位置）
- 在换向点或者在换向区处其分度横向进给的大小。
- 在到达终点位置时（通过方式）分度横向进给特性

方式＝0	对于两个最后的分度横向进给，划分剩余的位移，直至目标点在 2 个相同大小的剩余步（预设定）。
方式＝1	所有分度横向进给大小相同。 它们由总的横向进给计算。

确定运动同步动作

后面所执行的运动同步动作完全用于摆动。

您可以找到方案举例，用于满足您作为编制用户专用的摆动运动的模块而提出的各个要求。

说明
在单个情况下，同步条件也可以另外编程。

关键字

WHEN … DO …	当..., 则...
WHENEVER … DO	始终当..., 则...

功能

使用下列详细描述的语言手段可以实现下列功能

:

1. 在换向点处的横向进给。
2. 在换向区的横向进给。
3. 在两个换向点处的横向进给。
4. 在换向点处停止摆动运动。
5. 再次启动摆动运动。
6. 分度横向进给不要启动太早。

所有这里举例说明的同步动作适用于以下设定：

- 换向点 1 < 换向点 2
- Z = 摆动轴
- X = 横向进给轴

说明
更详细的解释可参阅运动同步动作一章。

确定配置摆动轴和进给轴以及确定最大进给和部分进给。

在换向区的横向进给。

在到达返回点之前，进给运动应当在某个返回范围内开始。

该同步动作阻止横向进给运动，直至摆动轴位于一个换向区之内。

在所给定的假设条件下（见上面）产生以下的指令：

换向区 1：

```
WHENEVER
$AA_IM[Z]>$SA_OSCIL
L_RESERVE_POS1[Z]+i
i1 DO $AA_OVR[X] =
0
```

总是当 MCS 中的摆动轴的当前位置大于等于换向区 1 开始时，设置摆动轴的轴向倍率为 0%。

换向区 2：

```
WHENEVER
$AA_IM[Z]<$SA_OSCIL
L_RESERVE_POS2[Z]+i
i2 DO $AA_OVR[X] =
0
```

总是当 MCS 中的摆动轴的当前位置小于换向区 2 开始时，设置进给轴的轴向倍率为 0%。

在换向点处的横向进给

只要摆动轴没有到达换向点，就不进行横向进给轴的运动。

在所给定的假设条件下（见上面）产生以下的指令：

换向区 1：

```
WHENEVER
$AA_IM[Z]<>$SA_OSCI
LL_RESERVE_POS1[Z]
DO $AA_OVR[X] = 0 →
→ $AA_OVR[Z] = 100
```

总是当 MCS 中的摆动轴 Z 的当前位置大于或小于换向点 1 的位置时，进给轴 X 的轴向倍率设置为 0%，摆动轴 Z 的轴向倍率设置为 100%。

换向区 2：

用于换向点 2：

```
WHENEVER
$AA_IM[Z]<>$SA_OSCI
LL_RESERVE_POS2[Z]
DO $AA_OVR[X] = 0 →
→ $AA_OVR[Z] = 100
```

总是当 MCS 中的摆动轴 Zu 的当前位置大于或小于换向点 2 的位置时，进给轴 X 的轴向倍率设置为 0%，摆动轴 Z 的轴向倍率设置为 100%。

在换向点处停止摆动运动

摆动轴在返回点上停住，同时开始进给运动。 如果横向进给运动完全执行，则继续执行摆动运动。

如果横向进给运动通过一个事先进行的、仍然有效的同步动作停止，则可以同时使用该同步动作，启动横向进给运动。

在所给定的假设条件下（见上面）产生以下的指令：

换向区 1：

```
WHENEVER
$SA_IM[Z]==$SA_OSCI
LL_RESERVE_POS1[Z]
DO $AA_OVR[X] = 0 →
→ $AA_OVR[Z] = 100
```

总是当 MCS 中的摆动轴当前位置等于换向位置 1 时，摆动轴的轴向倍率设置为 0%，进给轴的轴向倍率设置为 100%。

换向区 2：

```
WHENEVER
$SA_IM[Z]==$SA_OSCI
LL_RESERVE_POS2[Z]
DO $AA_OVR[X] = 0 →
→ $AA_OVR[Z] = 100
```

总是当 MCS 中的摆动轴 Zu 当前位置等于换向位置 2 时，摆动轴的轴向倍率设置为 0%，进给轴的轴向倍率设置为 100%。

换向点的在线计算

如果在对比的右侧有一个使用 $$ 标识的主过程变量，那么就会在 IPO 节拍中对这两个变量进行连续分析和相互比较。

说明

对此更多的信息参见“运动同步动作”章节。

再次启动摆动运动

如果分度横向进给运动已经结束，则使用同步动作，继续摆动轴的运动。

在所给定的假设条件下（见上面）产生以下的指令：

```
WHENEVER $AA_DTEPW[X]==0 DO $AA_OVR[Z] = 100
```

总是当 WCS 中进给轴 X 的零件进给的剩余行程等于零时，摆动轴的轴向倍率设置为 100%。

下一个分度横向进给

在结束进给之后，必须防止过早启动下一个部分进给。

为此可使用特定通道的标记 ($AC_MARKER[Index])，该标记在部分进给结束处 (部分剩余行程 ≡ 0) 被设定并且在离开返回范围时被删除。 然后用一个同步动作阻止下一个横向进给运动。

在所给定的假设条件下（见上面）产生给返回点 1 的以下指令：

1. 设定标记：

```
WHENEVER
$AA_DTEPW[X]==0 DO
$AC_MARKER[1]=1
```

总是当 WCS 中进给轴 X 的零件进给的剩余行程等于零时，带索引 1 的标记设置为 1。

2. 删除标记

```
WHENEVER
$AA_IM[Z]<>
$SA_OSCILL_RESERVE_
POS1[Z] DO
$AC_MARKER[1] = 0
```

总是当 MCS 中摆动轴 Z 的当前位置大于或小于换向点 1 的位置时，标记 1 设置为 0。

3. 阻止进给

```
WHENEVER
$AC_MARKER[1]==1 DO
$AA_OVR[X] = 0
```

总是当标记 1 相等时，进给轴 X 的轴向倍率设置为 0%。

冲裁和步冲 12

12.1 激活, 非激活

12.1.1 激活或取消冲压和步冲(SPOF, SON, PON, SONS, PONS, PDELAYON, PDELAYOF, PUNCHACC)

功能

激活/取消冲压或步冲

通过 PON 和 SON 可以激活冲压或步冲功能。 而 SPOF 会结束所有与此相关的功能。 模态有效的指令 PON 和 SON 相互关闭，即：PON 取消 SON，且 SON 取消 PON。

带预应力的冲压/步冲

SONS 和 PONS 功能同样启用冲压和步冲功能。

与 SON/PON 生效时插补平面上冲程控制相反，这两个功能采用了信号技术来控制伺服平面上的冲程释放。 由此可以用更高的冲程频率和更高的冲压功率加工。

在分析预应力中的信号时，所有会引起冲压/步冲轴位置改变的功能都被锁定，如手轮运行、通过 PLC 修改框架、测量功能等。

冲压延迟

PDELAYON 会延迟冲压冲程的输出。 模态有效的指令具有经过预处理的功能，一般在 PON 之前。 在 PDELAYOF 之后，继续正常冲压。

说明

延迟时间在设定数据 SD42400 $SC_PUNCH_DWELLTIME 中设置。

位移决定的加速度

通过 PUNCHACC 可以确定一条加速度特性曲线，它根据孔距定义了不同的加速度。

第二个冲压接口

如果机床需要换用第二个冲压接口（第二个冲压单元或类似的媒介），可以切换到控制系统上第二对高速数字输入输出端（I/O 对）。 所有的冲压功能可用于这两个接口。 通过指令 SPIF1 和 SPIF2 可以在第一个冲压接口和第二个冲压接口之间转换。

说明

前提条件：必须已经通过机床数据定义了第二个用于冲压功能的 I/O 对（(→ 参见机床制造商的说明！）。

句法

```
PON G... X... Y... Z...
SON G... X... Y... Z...
SONS G... X... Y... Z...
PONS G... X... Y... Z...
PDELAYON
PDELAYOF
PUNCHACC(<Smin>,<Amin>,<Smax>,<Amax>)
SPIF1/SPIF2
SPOF
```

含义

PON	激活冲压。
SON	激活步冲
PONS	激活带预应力的冲压
SONS	激活带预应力的步冲
SPOF	取消冲压/步冲
PDELAYON	激活冲压延迟
PDELAYOF	取消冲压延迟
PUNCHACC	激活行程控制式加速度
	参数：
	<Smin> 最小孔距
	<Amin> 开始加速度
	<Amin> 可以大于 <Amax>。
	<Smax> 最大孔距

<Amax> 结束加速度

<Amax> 可以大于 <Amin>。

SPIF1 激活**第一个**冲压接口

冲程控制由第一对高速 I/O 实现。

SPIF2 激活**第二个**冲压接口

冲程控制由第二对高速 I/O 实现。

提示：

复位或控制系统启动后，首个冲压接口始终有效。 如果只使用一个冲压接口，则无需编程。

示例

示例 1： 激活步冲

程序代码	注释
...	
N70 X50 SPOF	; 定位，不释放冲压。
N80 X100 SON	; 激活步冲，在运行前(X=50)和编程的运行结束(X=100)后释放冲程
...	

示例 2：冲压延迟

程序代码	注释
...	
N170 PDELAYON X100 SPOF	; 不带冲程释放的定位，激活延迟的冲程释放
N180 X800 PON	; 激活冲压。 到达终点位置后，冲程延迟释放。
N190 PDELAYOF X700	; 取消延迟冲裁，标准冲程释放生效
...	

示例 3： 带两个冲压接口的冲压

程序代码	注释
...	
N170 SPIF1 X100 PON	; 程序段结束时释放第一个高速输出端上的冲程。监控第一个输入端上的信号"冲程有效"。
N180 X800 SPIF2	; 第二次冲程释放在第二个高速输出端上进行。监控第二个输入端上的信号"冲程有效"。
N190 SPIF1 X700	; 所有后续的冲程控制都由第一个接口实现。
...	

其它信息

冲压和步冲，带预应力(PONS/SONS)

不可以同时在几个通道中进行带预应力的冲压和步冲。 PONS 或 SONS 仅可以在各自的通道中激活。

位移控制式加速度(PUNCHACC)

示例：
PUNCHACC(2,50,10,100)

2 毫米以下的孔距：

以最大加速度的 50% 运行。

孔距在 2 毫米到 10 毫米之间：

该加速度与距离成正比提高到 100%。

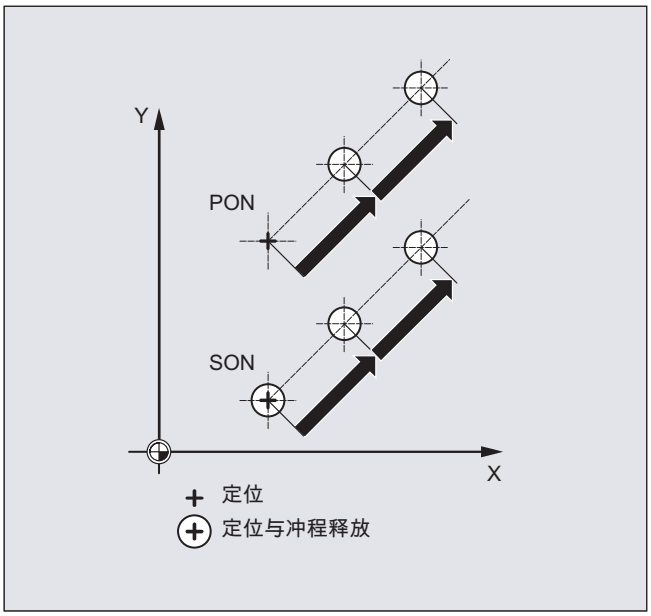
孔距大于 10 毫米：

以 100%的加速度运行。

释放第一个冲程

在步冲和冲压时，在激活该功能后释放第一个冲程在时间上不同。

- PON/PONS:
  - 所有的冲程 — 即使在激活之后第一个程序段的冲程 — 在程序段结束处发生。
- SON/SONS:
  - 在激活步冲之后，在程序段开始处已经发生第一个冲程。
  - 所有的其它冲程在程序段结束处释放。



立即进行冲压和步冲

只有当程序段中包含冲压轴或者步冲轴（有效平面的轴）的运行信息时，才释放一个冲程。

为了要在相同的地点释放一个冲程，用运行位移 0 编程一个冲压轴/步冲轴。

用可旋转的刀具工作

说明

为了可以在编程的轨迹处使用可旋转的刀具，请使用切线控制。

M 指令的应用

利用宏指令技术，仍可以用专用 M 功能替代语言指令使用（兼容性）。 其中，和旧系统的对应关系如下：

```
M20, M23  ≙  SPOF
M22       ≙  SON
M25       ≙  PON
M26       ≙  PDELAYON
```

宏指令文件的示例：

程序代码	注释
DEFINE M25 AS PON	; 激活冲压
DEFINE M125 AS PONS	; 激活带预应力的冲压
DEFINE M22 AS SON	; 激活步冲
DEFINE M122 AS SONS	; 激活带预应力的步冲
DEFINE M26 AS PDELAYON	; 激活延迟冲压
DEFINE M20 AS SPOF	; 关闭冲压,步冲
DEFINE M23 AS SPOF	; 关闭冲压,步冲

编程示例：

程序代码	注释
...	
N100 X100 M20	; 定位，不释放冲压。
N110 X120 M22	; 激活步冲，在运行前后释放冲程。
N120 X150 Y150 M25	; 激活步冲，在运行结束后释放冲程。
...	

12.2 自动划分位移

功能

部分区间再分

冲裁或步冲激活时，SPP 和 SPN 都会导致将轨迹轴的总运行区间划分为等长的部分区间（等长位移划分）。在内部每个部分区间都对应一个程序段。

冲程数量

冲裁时首个冲程在首个部分区间的终点时执行，相反步冲时在首个部分区间的起始点。从总运行区间会得到以下数量：

冲裁：冲程数 = 部分区间数

步冲：冲程数 = 部分区间数 + 1

辅助功能

辅助功能在生成的首个程序段中执行。

句法

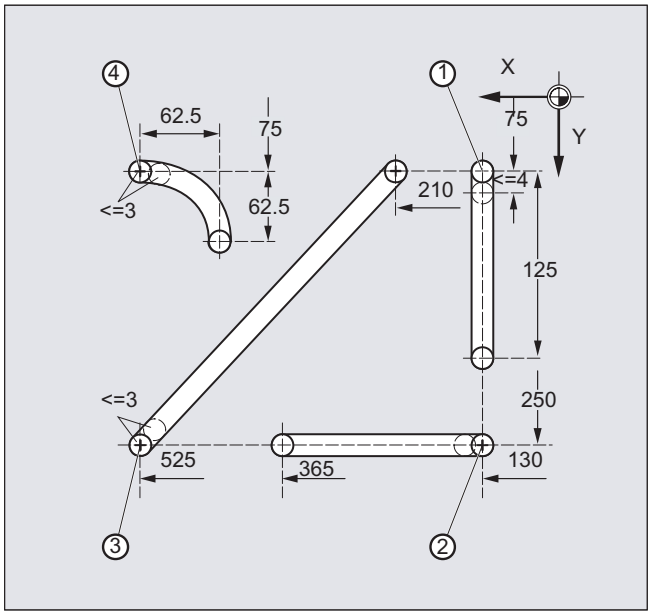
```
SPP=
SPN=
```

含义

SPP	部分区间大小（最大冲程间距）；模态有效
SPN	每个程序段的部分区间数；程序段方式有效

示例 1

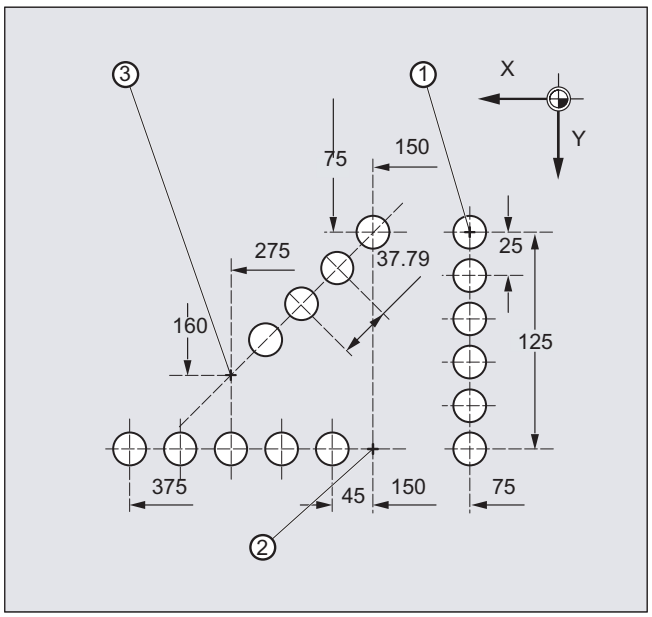
已编程的步冲区间要自动划分为等长的部分区间。



程序代码	注释
N100 G90 X130 Y75 F60 SPOF	; 定位在起始点 1
N110 G91 Y125 SPP=4 SON	; 打开步冲：自动位移分配最大部分区间位移：4 mm
N120 G90 Y250 SPOF	; 关闭步冲：定位在起始点 2
N130 X365 SON	; 打开步冲：自动位移分配最大部分区间位移：4 mm
N140 X525 SPOF	; 关闭步冲：定位在起始点 3
N150 X210 Y75 SPP=3 SON	; 打开步冲：自动位移分配最大部分区间位移：3 mm
N160 X525 SPOF	; 关闭步冲：定位在起始点 4
N170 G02 X-62.5 Y62.5 I J62.5 SPP=3 SON	; 打开步冲：自动位移分配最大部分区间位移：3 mm
N180 G00 G90 Y300 SPOF	; 关闭步冲

示例 2

对于独立的排孔要进行自动位移划分。划分时要分别定义最大部分区间长度 (SPP 值）。



程序代码		注释
N100 G90 X75 Y75 F60 PON	;	定位在起始点 1； 打开冲裁，冲压单个孔
N110 G91 Y125 SPP=25	;	自动位移分配最大部分区间长度 ： 25 mm
N120 G90 X150 SPOF	;	关闭冲裁；定位在 起始点 2
N130 X375 SPP=45 PON	;	打开冲裁；自动位移分配最大部分区间位移 ： 45 mm
N140 X275 Y160 SPOF	;	关闭冲裁；定位在 起始点 3
N150 X150 Y75 SPP=40 PON	;	打开冲裁；使用计算的部分区间长度 37.79 mm 替代编程的部分区间长度 40 mm。
N160 G00 Y300 SPOF	;	关闭冲裁；定位

12.2.1 在轨迹轴时的位移划分

分度距离 SPP 的长度

使用 SPP 说明最大的冲程距离和分度距离的最大长度，按照该分度距离对总的运行距离进行划分。 通过 SPOF 或者 SPP=0 关断该指令。

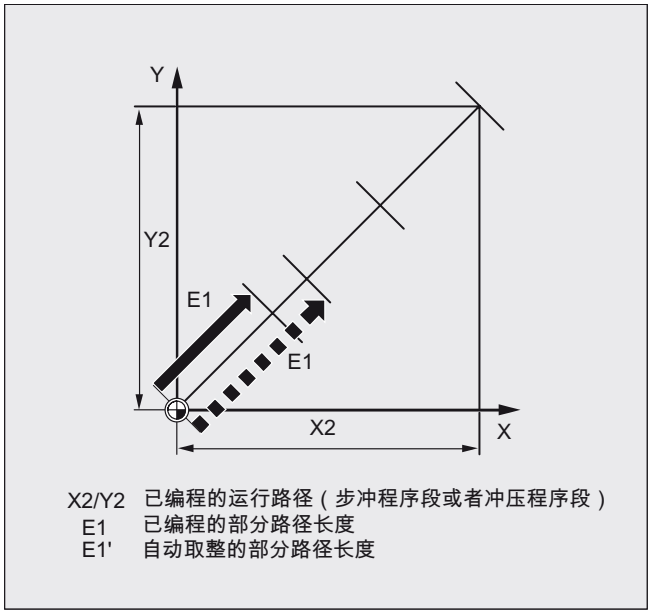
举例：

```
N10 SON X0 Y0
N20 SPP=2 X10
```

10 毫米的总的运行距离划分为 5 个分度距离，每段 2 毫米(SPP=2) 。

说明

用 SPP 划分位移总是进行等距离地划分：所有的分度距离长度相同。 也就是说，只有当总的运行距离与 SPP 值的商为整数时，编程的分度距离大小（ SPP 值）才有效。 如果不是这种情况，则分度距离的大小在内部减少，从而产生一个整数的商。



X2/Y2 已编程的运行路径（步冲程序段或者冲压程序段）
E1 已编程的部分路径长度
E1' 自动取整的部分路径长度

举例：

```
N10 G1 G91 SON X10 Y10
N20 SPP=3.5 X15 Y15
```

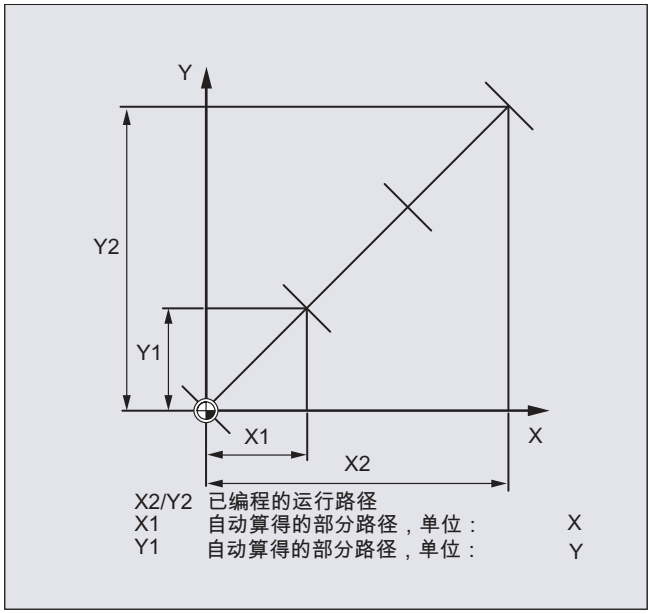
如果总的运行距离为 15 毫米，并且分度距离的一个长度为 3.5 毫米，则不会产生一个整数商（4.28）。因此降低 SPP 值，直至下一个可能的整数商。在这种情况下产生一个 3 毫米的分度距离长度。

分度距离 SPN 的个数

使用 SPN 您可以定义分度距离的个数，它应该由总的运行距离产生。分度距离的长度会自动计算出来。 因为 SPN 为程序段方式生效，所以在事先必须激活冲压或者步冲，使用 PON 或者 SON 。

SPP 和 SPN 在同一个程序段中

如果您在一个程序段中不仅编程了分度距离长度(SPP 而且也编程了分度距离(SPN）个数(SPN), 则 SPN 适用于该程序段， SPP 适用于所有其它的程序段。 如果 SPP 已经在 SPN 之前激活，则它在 SPN 程序段之后再次生效。



说明

只要在控制系统中原则上有冲裁/步冲可供使用，就可进行自动行程划分编程，带 SPN 或者 SPP ,也与该工艺无关。

12.2.2 在单个轴时的位移划分

如果除了轨迹轴之外也有单个轴作为冲压－步冲－轴定义，则它们可能也会进行自动位移划分。

SPP 时单个轴的特性

分度距离（SPP）的编程长度与轨迹轴相关。因此在除了单个轴运动和 SPP 值之外没有编程轨迹轴的程序段中，拒绝 SPP 值。

如果不仅编程了单个轴，而且在程序段中也编程了轨迹轴，则单个轴的特性取决于相应机床数据的设定。

1. 标准设置

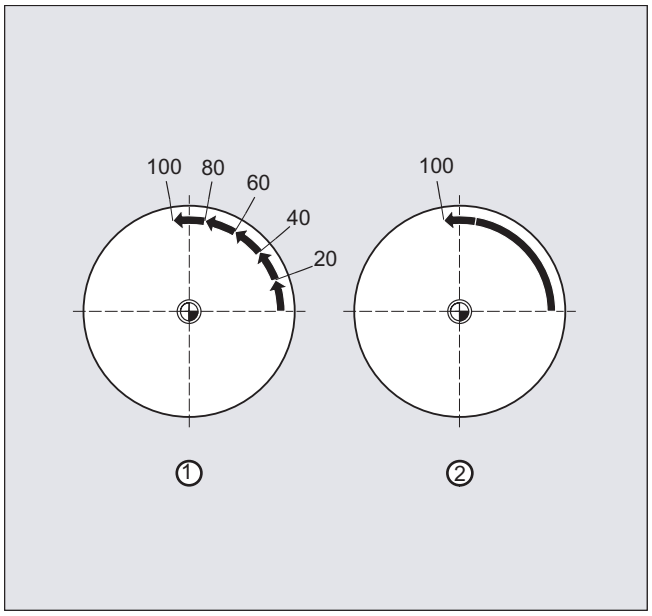
   单个轴的位移均匀地分布到用 SPP 产生的中间程序段中。

例如：

```
N10 G1 SON X10 A0
N20 SPP=3 X25 A100
```

通过 3 毫米的冲程距离，在 X 轴（轨迹轴）15 毫米的总运行距离中生成 5 个程序段。

因此 A 轴在每个程序段中转动 20 度。



1. 没有位移划分的单个轴

   单个轴在产生的第一个程序段中运行总位移。

2. 不同的位移划分

   单个轴的特性与轨迹轴的插补相关：

- 圆弧插补：位移划分
- 线性插补：没有位移划分

SPN 时的特性

如果不是同时编程一个轨迹轴，则编程的分度距离的个数也有效。

前提条件：单个轴作为冲压－步冲－轴定义。

磨削

13

13.1 在零件程序中磨削专用的刀具监控（TMON、TMOF）

功能

使用指令 TMON 可以在磨削刀具（类型 400-499）时激活 NC 零件程序的几何监控和转速监控。 监控一直有效，直至在零件程序中通过指令 TMOF 取消。

说明

请注意机床制造商的说明！

前提条件

必须设置磨削专用的刀具参数 $TC_TPG1 bis $TC_TPG9。

句法

```
TMON(<T 号>)
TMOF(<T 号>)
```

含义

TMON	用于**打开**磨削专用的刀具监控指令
TMOF	用于**关闭**磨削专用的刀具监控指令
<T 号>	规定 T 号 **提示：** 只有在带有 T 号的刀具未激活时才需要。
TMOF(0)	取消所有刀具的监控

其它信息

磨削专用的刀具参数

参数	含义	数据类型
$TC_TPG1	主轴号	INT
$TC_TPG2	级联规则 这些参数在砂轮的左右侧自动保持一致。	INT
$TC_TPG3	最小的砂轮半径	REAL
$TC_TPG4	最小的砂轮宽度	REAL
$TC_TPG5	实际的砂轮宽度	REAL
$TC_TPG6	最大的转速	REAL
$TC_TPG7	最大周边速度	REAL
$TC_TPG8	斜砂轮的角度	REAL
$TC_TPG9	用于半径计算的参数号	INT

文献：
功能手册 基本功能；刀具补偿 (W1)

通过刀具选择打开刀具监控

当刀具选择激活后，磨削刀具（类型 400－499）的刀具监控自动生效，当然这取决于机床数据的设定。

对于主轴，在任何时候只能有 一个 监控生效。

几何监控

监控当前的砂轮半径和宽度。

转速给定值对极限值的监控周期性地考虑主轴倍率。

由最大的砂轮圆周速度和当前的砂轮半径所计算得到的转速与最大的转速进行比较，其较小值就作为转速极限值。

加工，无 T 号和 D 号

通过机床数据可以设置一个标准 **T** 号和标准 **D** 号，

不允许对它再进行编程，在开机/重启后生效。

示例： 所有的加工使用相同的砂轮。

通过机床数据可以设定刀具在复位后仍然保持有效（参见“自由 D 号规定，切削刃号码 (页 445)”）。

其它功能

14

14.1 轴功能 (AXNAME, AX, SPI, AXTOSPI, ISAXIS, AXSTRING, MODAXVAL)

功能

当未识别轴的名称时，例如，在设置一般有效循环时使用 AXNAME。

AX 用于几何轴和同步轴的间接编程。 此时轴名称存放在一个类型 AXIS 的变量中或者由指令如 AXNAME 或 SPI 提供。

当轴功能用于一个主轴，例如同步主轴编程时，使用 SPI。

使用 AXTOSPI，可将一个轴名称转换到另一个主轴索引中 (转换功能针对 SPI)。

使用 AXSTRING，可将一个轴名称(数据类型 AXIS) 转换到一个字符串中 (转换功能针对 AXNAME)。

在一般有效循环中使用 ISAXIS，以确保某个指定的几何轴存在，并由 $P_AXNX 安全中断随后的调用。

使用 MODAXVAL，可以在模数回转轴时确定模数位置。

句法

```
AXNAME("字符串")
AX[AXNAME("字符串")]
SPI（n）

AXTOSPI(A) 或 AXTOSPI(B) 或 AXTOSPI(C)
AXSTRING（SPI（n））
ISAXIS(<几何轴号>)
<模数位置>=MODAXVAL(<轴>,<轴位置>)
```

含义

AXNAME	如果将输入字符串转换为轴标识符；输入字符串必须包含一个有效的轴名称。
AX	变量轴名称

SPI	将主轴编号转换为轴名称；转换参数必须包含一个有效的主轴编号。
n	主轴号
AXTOSPI	将轴标识符转换为一个整数型主轴索引。 AXTOSPI 相当于 SPI 的转换功能。
X, Y, Z	AXIS 型的轴标识符作为变量或常量
AXSTRING	输出带所分配主轴号的字符串。
ISAXIS	检查是否存在规定的几何轴。
MODAXVAL	在模数回转轴时确定模数位置：这符合模数余数，与参数化的模数范围有关（在标准设置下为 0 至 360 度；通过 MD30340 MODULO_RANGE_START 和 MD30330 $MA_MODULO_RANGE 可以改变模数范围的起始值和大小）。

说明

SPI 扩展

轴功能 SPI（n）也可用于读取和写入框架组件。 为此框架可以例如通过句法 $P_PFRAME[SPI(1),TR]=2.22 写入。

通过附加编程轴位置，通过地址 AX[SPI(1)]=<轴位置> 可以运行一根轴。 前提是主轴位于定位运行或者轴运行。

示例

示例 1： AXNAME, AX, ISAXIS

程序代码	注释
OVRA[AXNAME（“横向轴”）]=10	; 横行轴倍率
AX[AXNAME（“横向轴”）]=50.2	; 横向轴的终点位置
OVRA[SPI（1）]=70	; 主轴 1 的倍率
AX[SPI(1)]=180	; 主轴 1 的终点位置
IF ISAXIS(1) == FALSE GOTOF WEITER	; 有横坐标?
AX[$P_AXN1]=100	; 运行横坐标
继续:	

示例 2： AXSTRING

在使用 AXSTRING[SPI(n)] 编程时，不再将分配给主轴的轴索引作为主轴号输出，而是输出字符串“Sn”。

程序代码	注释
AXSTRING[SPI(2)]	; 输出字符串“S2”。

示例 3： MODAXVAL

应该确定模数回转轴的模数位置 A。

计算输出值是轴位置 372.55。

参数化的模数范围为 0 至 360 度：

MD30340 MODULO_RANGE_START = 0

MD30330 $MA_MODULO_RANGE = 360

程序代码	注释
R10=MODAXVAL(A,372.55)	; 计算的模数位置 R10 = 12.55。

示例 4： MODAXVAL

如果编程的轴名称不涉及到模数回转轴，则保持不变返回要转换的值 (<轴位置>)。

程序代码	注释
R11=MODAXVAL(X,372.55)	; X 是直线轴；R11 = 372.55。

14.2 可转换的几何轴 (GEOAX)

功能

使用“可转换的几何轴”功能，通过机床数据文件所配置的几何轴组合可从零件程序开始修改。 对此一个定义为同步附加轴的通道轴可以替换任意一个几何轴。

句法

```
GEOAX(<n>,<通道轴>,<n>,<通道轴>,<n>,<通道轴>)
GEOAX()
```

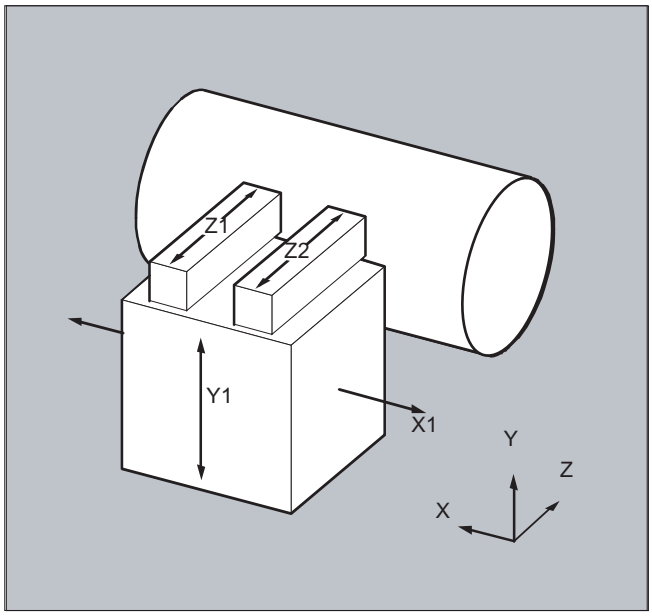
含义

GEOAX(...)	用于切换几何轴的指令 **提示：** GEOAX()（不带参数规定）调用几何轴的基本配置。
<n>	通过该参数指定要分配下列规定通道轴的几何轴号码。 取值范围： 1、2 或者 3 **提示：** Mit <n>=0 可以将下列规定的通道轴不用替代的从几何轴组合中去除。
<通道轴>	通过该参数可以规定要接受几何轴组合的通道轴名称。

示例

示例 1： 以切换方式切换两根轴作为几何轴

刀具溜板可以通过通道轴 X1，Y1，Z1，Z2 来运行：



这样设计几何轴，在打开后首先使 Z1 作为第 3 几何轴以几何轴名称“Z”有效，并与 X1 和 Y1 形成几何轴组合。

在零件程序中可以使用轴 Z1 和 Z2 交替地作为几何轴 Z：

程序代码	注释
...	
N100 GEOAX (3,Z2)	; 通道轴 Z2 作为第 3 几何轴 (Z) 起作用。
N110 G1 ...	
N120 GEOAX (3,Z1)	; 通道轴 Z1 作为第 3 几何轴 (Z) 起作用。
...	

示例 2： 在 6 通道轴时切换几何轴

机床具有 6 通道轴，名称分别是 XX, YY, ZZ, U, V, W。

几何轴配置基本设置通过机床数据实现：

通道轴 XX = 第 1 几何轴（X 轴）

通道轴 YY = 第 2 几何轴（Y 轴）

通道轴 ZZ = 第 3 几何轴（Z 轴）

程序代码	注释
N10 GEOAX()	; 几何轴的基本配置有效。
N20 G0 X0 Y0 Z0 U0 V0 W0	; 所有轴快速运动到位置 0。

程序代码	注释
N30 GEOAX(1,U,2,V,3,W)	; 通道轴成为第一个 (X), V 成为第二个 (Y), W 成为第三个几何轴 (Z)。
N40 GEOAX(1,XX,3,ZZ)	; 通道轴 XX 成为第一个 (X), ZZ 成为第三个几何轴 (Z)。 通道轴 V 保留为第二个几何轴 (Y)。
N50 G17 G2 X20 I10 F1000	; 在 X/Y 平面中完整的圆。 运行通道轴 XX 和 V。
N60 GEOAX(2,W)	; 通道轴 W 成为第二个几何轴 (Y)。
N80 G17 G2 X20 I10 F1000	; 在 X/Y 平面中完整的圆。 运行通道轴 XX 和 W。
N90 GEOAX()	; 返回到基本状态。
N100 GEOAX(1,U,2,V,3,W)	; 通道轴成为第一个 (X), V 成为第二个 (Y), W 成为第三个几何轴 (Z)。
N110 G1 X10 Y10 Z10 XX=25	; 通道轴 U, V, W 分别运行到位置 10, XX 作为辅助轴运行到位置 25。
N120 GEOAX(0,V)	; 从几何轴组合中去掉 V。 U 和 W 仍然为第一个 (X) 和第三个几何轴 (Z)。 第二个几何轴 (Y) 保持未配置状态。
N130 GEOAX(1,U,2,V,3,W)	; 通道轴 U 保持为第一个(X), V 成为第二个(Y), W 保持为第三个几何轴(Z)。
N140 GEOAX(3,V)	; V 成为第三个几何轴(Z), 同时 W 被覆盖且被从几何轴组合中去掉。第二个几何轴(Y)仍然保持未配置状态。

说明

轴配置

有关几何轴、辅助轴、通道轴和加工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 (分配几何轴到通道轴）

MD20060 $MC_AXCONF_GEOAX_NAME_TAB (通道中的几何轴名称)

MD20070 $MC_AXCONF_MACHAX_USED (加工轴号码在通道中有效)

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 (分配主轴到机床轴)

资料：

功能手册 基本功能；轴、坐标系、框架(K2)

限制

- 几何轴的切换在下列情况时不适用：
  - 激活的转换
  - 激活的样条插补
  - 激活的刀具半径补偿
  - 激活的刀具微补偿
- 如果几何轴和通道轴显示相同的名称，那么不可以转换每个几何轴。
- 参与转换的轴不得参与可能会超出程序段范围的动作，例如类型 A 的定位轴或者从动轴可能会有这样的情况。
- 使用指令 GEOAX 只能替换启用时已经存在的几何轴（即不会再定义新的轴）。
- 使用 GEOAX 在处理轮廓表的过程中更换轴（CONTPRON, CONTDCON) 会导致报警。

边界条件

更换后轴状态

一个由几何轴组合中的转换替代的轴在转换过程之后，通过它们通道轴名称作为附加轴可编程。

框架，保护范围，工作区域限制

所有的框架，保护范围和工作范围限制都可以用几何轴转换来删除。

极坐标

使用 GEOAX 交换几何轴会向一个平面转换一样 (用 G17-G19) 将模态极坐标设定成数值 0。

DRF, NPV

一个可能发生的手轮偏移（DRF）或者一个外部的零点偏移（NPV）在转换之后依旧有效。

几何轴基本配置

指令 GEOAX() 用来调用几何轴组合的基本配置。

在上电后并且在转换到“参考点运行方式”时将自动转换回基本配置。

刀具长度补偿

一个当前有效的刀具长度补偿在转换过程之后也是有效的。尽管如此，它对新接纳或者交换位置的几何轴仍然有效，当它们还没有运行时。使用第一个针对这些几何轴的运行指令时，生成的运动行程相应的由刀具长度补偿的总和与编程设计的运动行程组成。

在转换时在轴组合中保持自身位置的几何轴，也保持其状态，包括刀具长度补偿。

激活转换时几何轴配置

在一个有效的转换中所适用的几何轴配置（通过机床数据确定），不可以通过功能“可转换的几何轴”来更改。

如果需要改变和转换相关联的几何轴配置，那么这只有通过其它的转换才可以。

一个通过 GEOAX 修改的几何轴配置可通过激活一个转换来删除。

如果所设置的机床数据对于转换和对于几何轴的转换相互矛盾，那么在转换中的设置有优先权。

示例：

一个转换有效。根据机床数据转换在复位时保持不变，同时在复位时还生成几何轴的基本配置。 在这种情况下几何轴配置和其随转换而确定的配置一样保持不变。

14.3 轴容器 (AXCTSWE, AXCTSWED)

功能

在回转台机床/多主轴机床中，夹装了工件的轴从一个加工单元运行到下一个加工单元。因为加工单元隶属于不同的 NCU 通道，当更换工位/位置时，必须将载有工件的轴重新动态分配给相应的 NCU 通道。 轴容器正是用于这个目的。

就某一时刻而言，在局部加工单元上始终只有一个工件装夹轴/主轴处于激活状态。 轴容器可以与所有装夹轴/主轴一起合并连接，这些装夹轴/主轴对于加工单元激活总是只需要一个。

通过轴容器定义的可使用轴的切换通过移动轴容器中的条目（“轴容器旋转”）以实现一个通过设置数据可规定的步距（槽数目）。

针对轴容器旋转的调用由零件程序通过指令 AXCTSWE 或 AXCTSWED 实现。

句法

```
AXCTSWE(<轴容器>)
AXCTSWED(<轴容器>)
```

含义

AXCTSWE	用于旋转轴容器的指令 当控制系统中出现了为容器的轴释放所有通道的指令时，就会通过 SD41700 $SN_AXCT_SWWIDTH[<容器号码>] 中保存的容器专用步距实现。
AXCTSWED	用于在激活通道唯一有效情况下旋转轴容器的指令（调试指令系列！） **提示：** 当其余容器中的轴所具有的通道处于 RESET 状态时，才释放容器中所输入的轴。
<轴容器>	要继续切换的轴容器名称。 允许的说明有： CT<容器号码> 在字母组合 CT 上附加轴容器号码。 示例： CT3

<容器名称>	通过 MD12750 $MN_AXCT_NAME_TAB 设置的个性化的轴容器名称。 示例： A_CONT3

其它信息

轴容器

通过轴容器可以分配。

- 局部轴和/或者
- 链接轴

带链接轴的轴容器是一个 NCU 可以搭接的运行工具（NCU 全局）,它通过控制系统进行协调。 只管理局部轴的轴容器是可以的。

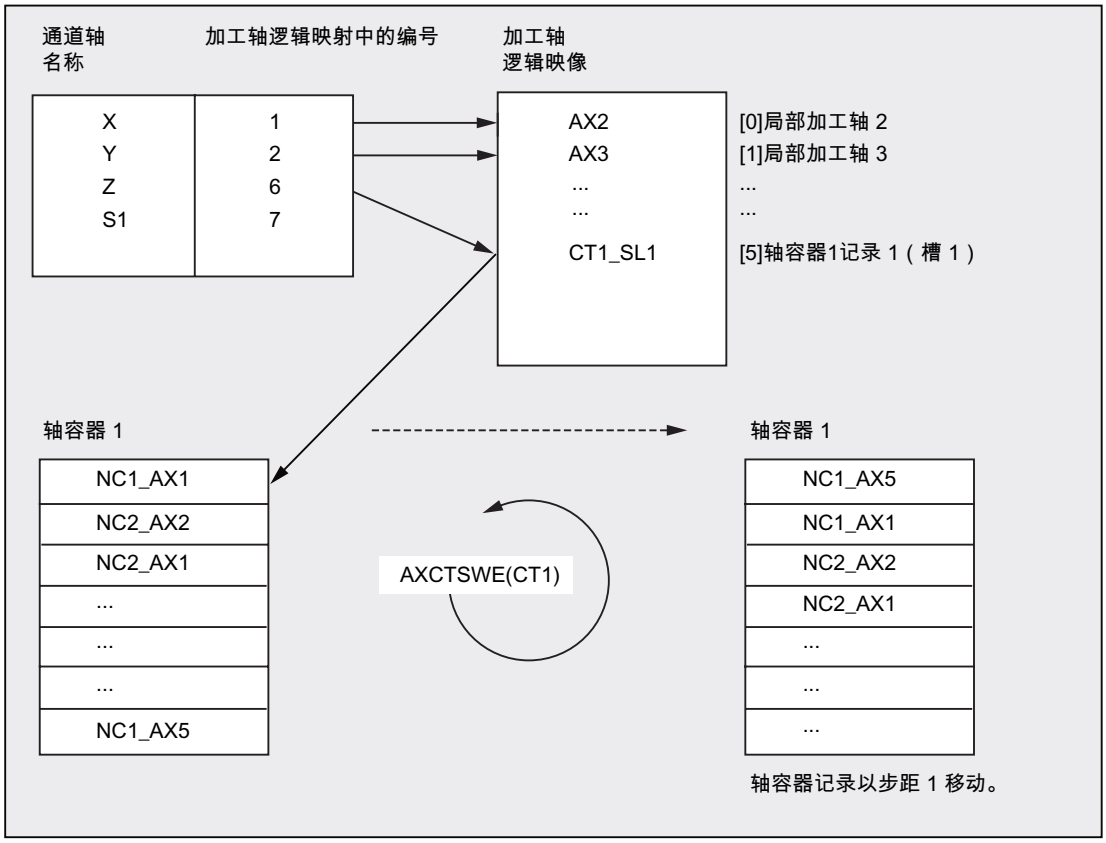
文献：
有关轴容器设计的详细提示可参见：
功能手册 扩展功能：多个 NCU 上多个操作面板，中央系统 (B3)

释放条件

AXCTSWE ()

轴已经输入在指定容器中的每个通道发出旋转容器的释放(enable)指令，当该通道结束处理位置/工位时。 当控制系统中出现了为容器的轴释放**所有**通道的指令时，就会通过 SD41700 $SN_AXCT_SWWIDTH[<容器号码>] 中保存的容器专用步距实现。

示例：



在轴容器旋转 1 之后，通道轴 Z 代替轴 AX1 分配到 NCU1，轴 AX5 分配到 NCU1。

AXCTSWED ()

指令系列 AXCTSWED() 可以用来简化调试。 轴容器在有效通道单独作用下旋转 SD41700 $SN_AXCT_SWWIDTH[<容器号码>] 中所存储的步距。 只有在容器中拥有轴的的其余通道处于**复位**状态时，才能使用该调用。

有效性

从新的轴配置到一个旋转轴容器的过程中，其通道通过逻辑机床轴映象指向被旋转的轴容器的所有 NCU 均会被涉及。

轴容器旋转包含 GET/GETD

当轴容器旋转使能时，所有分配给通道的轴容器轴通过 GET 或者 GETD 分配给通道。 仅当轴容器旋转后，才再次允许轴输出。

说明

这些特性可以通过机床数据设置。 请注意机床制造商说明。

说明

对于处于主运行轴状态的轴（例如对于 PLC 轴）， 包含 GET / GETD 的轴容器旋转不 使用，因为该轴为了轴容器旋转必须离开现有状态。

14.4 等待有效的轴位置（WAITENC）

功能

在 NC 程序中可编写 WAITENC 指令等待，由 MD34800 $MA_WAIT_ENC_VALID = 1 配置的轴获得经过同步或补偿的位置。

在等待状态下可执行中断，例如启动一个异步子程序，或切换到 JOG 模式。 必要时可通过继续执行程序重新进入等待状态。

说明

等待状态在操作界面中通过停止状态“等待测量系统”显示。

句法

可在任意 NC 程序部分编程 WAITENC 指令。

必须在单独的程序段中进行编程。

```
...
WAITENC
...
```

示例

WAITENC 例如可用于事件控制的用户程序 .../_N_CMA_DIR/_N_PROG_EVENT_SPF，如下面的例子所示。

应用示例：断电后的刀具回程，带方向转换

带刀具定向的加工已由于电源故障中断。
在之后的启动中调用事件控制用户程序 .../_N_CMA_DIR/_N_PROG_EVENT_SPF。

在事件控制用户程序中使用 WAITENC 等待经过同步或补偿的轴位置，从而计算出框架，按刀具方向校准 WCS。

程序代码	注释
...	
IF $P_PROG_EVENT == 4	; 启动
IF $P_TRAFO <> 0	; 已选择坐标转换。
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	
...	

之后可在运行方式 JOG 中，通过回程运行将刀具沿刀具轴的方向退回。

14.5 检查现有的 NC 语言范围（STRINGIS）

功能

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。

定义

```
INT STRINGIS(STRING <名称>)
```

句法

```
STRINGIS(<名称>)
```

含义

STRINGIS:	带返回值的功能
<名称>：	待检查的 NC 编程语言元素的名称
返回值：	返回值的格式为 yxx（十进制）。

NC 编程语言元素

可检查以下 NC 编程语言元素：

- 所有 G 功能组的 G 代码，例如 G0、INVCW、POLY、ROT、KONT、SOFT、CUT2D、CDON、RMB、SPATH
- DIN 或 NC 地址，例如 ADIS、RNDM、SPN、SR、MEAS
- 功能，例如 TANG(...) 或 GETMDACT
- 步骤，例如 SBLOF。
- 关键字，例如 ACN、DEFINE 或 SETMS
- 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

返回值

返回值仅与前 3 个十进制位相关。 返回值的格式为 yxx，其中 y 为基本信息，xx 为详细信息。

返回值	含义	
000	字符串“名称”在现有系统中未知 1)	
100	字符串“名称”是 NC 编程语言元素，但是当前不可编程（选项/功能未生效）	
2xx	字符串“名称”是可编程的 NC 编程语言元素（选项/功能生效）。 详细信息 xx 包含了更多元素类型的相关信息：	
	xx	**含义**
	01	DIN 地址或 NC 地址 2)
	02	G 代码（例如 G04、INVCW）
	03	带返回值的功能
	04	无返回值的功能
	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 NCK 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 代码（ISO 语言模式必须生效）
400	字符串“名称”是未识别为 xx == 01 或 xx == 10，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。 也可使用地址扩展编程 NC 地址。 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，也识别为宏

示例

在下面的示例中假设设定为字符串的 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 代码：

   ```
   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")`

    另见下面的段落：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”定义为轴：

```
210 == STRINGIS("LONGMACRO")
```

13. 字符串“MYVAR”定义为 LU 变量：

```
211 == STRINGIS("MYVAR")
```

14. 字符串“XYZ”不是 NCK 中已知的指令、GUD 变量、宏名称或循环名称：

```
000 == STRINGIS("XYZ")
```

刀具刀库管理

如果刀具刀库管理功能未生效，则与机床数据

- MD10711 $MN_NC_LANGUAGE_CONFIGURATION 无关，

STRINGIS 总是对刀具刀库管理的系统参数输出值 000。

ISO 模式

若“ISO 模式”功能生效：

- 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）

STRINGIS 会首先将指定字符串作为 SINUMERIK G 代码检查。 如果字符串不是 SINUMERIK G 代码，则之后会将其作为 ISO G 代码检查。

编程的切换（G290 （SINUMERIK 模式），或 G291 （ISO 模式））对 STRINGIS 没有影响。

示例

STRINGIS(...) 功能相关的机床数据有以下值：

- MD10711 $MN_NC_LANGUAGE_CONFIGURATION = 2（只有设置了选件的 NC 语言指令才能被识别）
- MD19410 $ON_TRAFO_TYPE_MASK = 'H0'（选件：坐标转换）
- MD10700 $MN_PREPROCESSING_LEVEL='H43'（循环的预处理生效）

执行以下示例程序时不输出故障信息：

程序代码	注释
N1 R1=STRINGIS("TRACYL")	; R1 == 0，由于缺少坐标转换选件，
	; TRACYL 被视为
	; “无法识别”
N2 IF STRINGIS("TRACYL") == 204	;
N3 TRACYL(1,2,3)	; 跳过 N3
N4 ELSE	
N5 G00	; 而是执行 N5
N6 ENDIF	
N7 M30	

14.6 读取功能调用 ISVAR 和机床数据队列索引

功能

根据 NC 语言 ISVAR 指令的功能带有：

- BOOL 型功能值
- STRING 型传送参数

当传送参数包含一个在 NC 中已知的变量时，ISVAR 指令提供 TRUE（机床数据、设定数据、系统变量、一般变量，如 GUD's）。

句法

```
ISVAR(<变量名称>)
ISVAR(<名称>,[<值>,<值>])
```

含义

<变量名称>	String 型的传送参数可以是无维、一维或二维的。
<名称>	带一个 NC 已知变量的标识符，此变量带或不带数组索引，作为机床数据、设定数据、系统变量或一般变量。 **扩展：** 对于一般和通道专用的机床数据，在缺失索引的情况下也读取数组的第一个单元。
<值>	BOOL 型功能值

检查

根据传送参数进行下列检查：

- 如果有标识符
- 则涉及到的是一维或二维数组
- 如果允许数组索引

仅当确定进行所有检查时，则反馈 TRUE。 如果仅有一个检查未完成或句法出现错误，则确认该检查为 FALSE 。 轴变量将作为轴名称的变址而不会对其进行检查。

扩展： 读取不带索引的机床数据和设定数据数组。

在缺失**一般和通道专用的** 机床数据索引时，**不再**发出报警 12400“通道 % 1 程序段 % 2 数组 % 3 单元不存在”。

此外 **至少轴索引** 要在 **轴专用的** 机床数据下编程。 若不在范围内，则发出报警 12400。

示例： 功能调用 ISVAR

程序代码	注释
DEF INT VAR1	
DEF BOOL IS_VAR=FALSE	; 传递参数是一个普通变量
N10 IS_VAR=ISVAR（“VAR1”）	; IS_VAR 在这种情况下为真
DEF REAL VARARRAY[10,10]	
DEF BOOL IS_VAR=FALSE	; 不同的句法变量
N20 IS_VAR=ISVAR（“VARARRAY[,]”）	; IS_VAR 为真，带有一个 二维队列
N30 IS_VAR=ISVAR（“VARARRAY”）	; IS_VAR 为真，存在变量
N40 IS_VAR=ISVAR("VARARRAY[8,11]")	; IS_VAR 为假，不允许队列索引
N50 IS_VAR=ISVAR（“VARARRAY[8,8”）	; IS_VAR 为假，句法错误 缺失 "]"
N60 IS_VAR=ISVAR（“VARARRAY[,8]”）	; IS_VAR 为真，允许队列索引
N70 IS_VAR=ISVAR（“VARARRAY[8,]”）	; IS_VAR 为真
DEF BOOL IS_VAR=FALSE	; 传递参数是一个机床数据
N100 IS_VAR=ISVAR("$MC_GCODE_RESET_VALUES[1]"	; IS_VAR 为真
DEF BOOL IS_VAR=FALSE	; 传递参数是一个系统变量
N10 IS_VAR=ISVAR（“$P_EP”）	; IS_VAR 在这种情况下为真
N10 IS_VAR=ISVAR（“$P_EP[X]”）	; IS_VAR 在这种情况下为真

示例： 读入带或不带索引的机床数据队列

在下面的情况下读取第一个单元

```
R1=$MC_EXTERN_GCODE_RESET_VALUES
```

这和以前相符

```
R1=$MC_EXTERN_GCODE_RESET_VALUES[0]
```

或读取第一个单元

```
R1=$MA_POSTCTRL_GAIN[X1]
```

这和以前相符

```
R1=$MA_POSTCTRL_GAIN[0, X1]
```

在下面的情况下也同步读取第一个单元

```
WHEN TRUE DO $R1 = $MC_EXTERN_GCODE_RESET_VALUES
```

这和以前相符

```
WHEN TRUE DO $R1 = $MC_EXTERN_GCODE_RESET_VALUES[0]
```

迄今为止报警 12400 **未读取。**

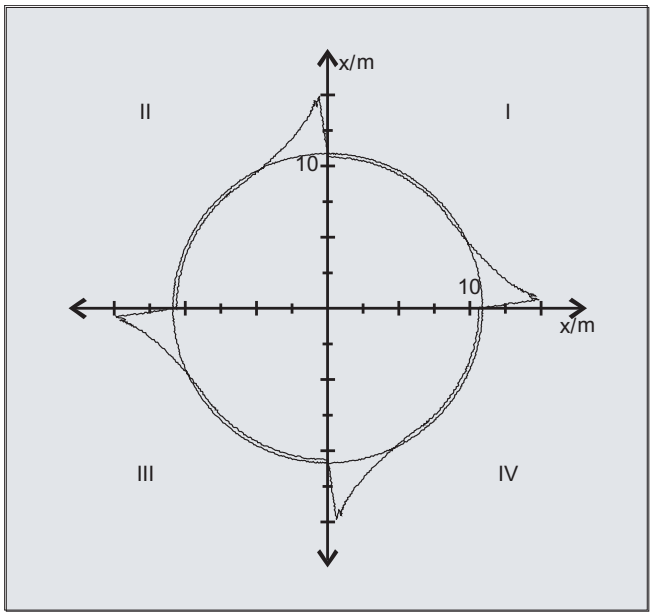
在下面的情况下继续发出报警 12400

```
R1=$MA_POSTCTRL_GAIN
```

14.7 学习补偿特性曲线 (QECLRNON, QECLRNOF)

功能

象限误差补偿可减少转换运动方向时因机械非线性（例如摩擦，松动）或者扭矩而产生的轮廓误差。 优化的补偿数据可以在学习期间由控制系统根据一个神经原网络进行适配，并且可以自动计算出补偿特征曲线。 这种学习可以最多 4 个轴同时进行。



句法

```
QECLRNON
QECLRNOF
```

激活学习过程： QECLRNON

自身的学习过程在 NC 程序中用指令 QECLRNON 在规定轴下激活：

```
QECLRNON (X1, Y1, Z1, Q)
```

只有当该指令激活时，改变特性线。

关闭学习： QECLRNOF

在关闭所需轴的学习运动后，学习过程通过 QECLRNOF 对于所有轴同时关闭。

含义

QECLRNON (轴 1,…4)	激活功能“学习象限误差补偿”
QECLRNO	关闭功能“学习象限误差补偿”
QECLRN.SPF	学习循环
QECDAT.MPF	样本 NC 程序用于系统变量占用和学习循环参数化
QECTEST.MPF	圆形测试样本 NC 程序

说明

学习所要求的轴的位移运动可以利用 NC 程序产生。这里学习运行以一个学习循环的形式出现。

第一次学习

在开机调试第一次学习时，在 PLC 基本程序的磁盘中包含样本 NC 程序，用于学习运行的学习，以及用于学习 QFK 系统变量的占用：

重新学习

重新优化已经学习的特征曲线可以用“补充学习”进行。在到目前为止用户存储器中的数据上进行。为了补充学习，您可以把样本 NC 程序适配于您的要求。

学习循环的参数 (例如 QECLRN.SPF) 有可能要进行修改以便“重新学习”：

- 设置"学习方式" = 1
- 有时减少“学习过程次数”
- 有时激活“分段方式学习”，并确定相应的范围极限

14.8 交互式调用零件程序 (MMC) 窗口

功能

通过 MMC 指令可以在 HMI 上从零件程序中显示用户自定义对话窗口（对话显示屏幕）。

通过纯文本设计来确定对话窗口的外形（循环目录中的 COM 文件），HMI 系统软件此时保持不变。

用户定义的会话窗口不可以同时在几个不同的通道中调用。

句法

```
MMC(CYCLES,PICTURE_ON,T_SK.COM,BILD,MGUD.DEF,BILD_3.AWB,TEST_1,A1
","S")
```

含义

MMC	从零件程序中交互式调用对话窗口 HMI。
CYCLES	操作区，在此执行所设计的用户会话。
PICTURE_ON 或 PICTURE_OFF	指令： 屏幕选择或者屏幕撤销选择
T_SK.COM	Com 文件： 会话屏幕文件名称（用户循环）。 在此确定会话屏幕的外观。 在会话屏幕中可以显示用户变量和/或注释文本。
BILD	会话屏幕名称： 单个的屏幕通过会话屏幕名称选择。
MGUD.DEF	用户数据定义文件，在读写变量时可以对此进行存取。
BILD_3.AWB	图形文件
TEST_1	显示时间或者应答变量
A1	文本变量...",
"S"	应答方式： 同步，通过软键¡°OK¡±进行应答

文献

有关编程 MMC 指令的详细提示（包括编程举例）参见“调试手册”。

14.9 程序执行时间/工件计数器

14.9.1 程序运行时间/工件计数器（概述）

为了对机床操作人员提供支持，提供了程序运行时间和工件计数的相关信息。

这些信息可以作为系统变量在 NC 和/或 PLC 程序中处理。 同时这些信息提供用于操作面板上的显示。

14.9.2 程序运行时间

功能

功能“程序运行时间”提供了 NC 内部计时器用于监控工艺过程，它可以通过 NC 和通道专用的系统变量在零件程序和同步动作中读取。

用于运行时间测量的触发器（$AC_PROG_NET_TIME_TRIGGER）是一个唯一可写的功能系统变量，用于选择性测量程序步骤。即通过在 NC 程序中触发器写入可以激活并再次关闭时间测量。

系统变量	含义	活动
NC 专用		
$AN_SETUP_TIME	从上一次使用缺省值启动控制系统（冷启动）到现在的时间，单位分 在每次使用缺省值启动控制系统时都将自动复位为“0”。	• 总是激活
$AN_POWERON_TIME	从上一次控制系统正常启动（热启动）到现在的时间，单位分 在每次正常启动控制系统时都将自动复位为“0”。	
通道专用		
$AC_OPERATING_TIME	在自动方式时 NC 程序的总运行时间，单位秒 在每次控制系统启动时都将自动复位为“0”。	• 通过 MD27860 激活

系统变量	含义		活动
$AC_CYCLE_TIME	所选择的 NC 程序的运行时间，单位秒 在每次启动一个新的 NC 程序时都将自动复位为“0”。		• 仅自动运行方式
$AC_CUTTING_TIME	加工时间，单位秒 即测得的 NC 启动和程序结束/NC 复位之间、所有 NC 程序中轨迹轴（至少一条）的运行时间，不包含快速移动 当暂停时间生效时，计算被中断。 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”。		
$AC_ACT_PROG_NET_TIME	当前 NC 程序的当前净运行时间，单位秒 在每次启动一个 NC 程序时都将自动复位为“0”。		• 总是激活 • 仅自动运行方式
$AC_OLD_PROG_NET_TIME	正确用 M30 结束程序的净运行时间，以秒为单位		
$AC_OLD_PROG_NET_TIME_COUNT	更改到 $AC_OLD_PROG_NET_TIME 在接通电源后 $AC_OLD_PROG_NET_TIME_COUNT 置“0”。 当控制系统 $AC_OLD_PROG_NET_TIME 重新写入时， $AC_OLD_PROG_NET_TIME_COUNT 总是升高。		
$AC_PROG_NET_TIME_TRIGGER	触发器用于运行时间测量：		• 仅自动运行方式
	0	中央状态 触发器未激活。	
	1	结束 结束测量并从 $AC_ACT_PROG_NET_TIME 复制值到 $AC_OLD_PROG_NET_TIME。 $AC_ACT_PROG_NET_TIME 置“0”并继续运行。	

系统变量	含义		活动
	2	Start 启动测量并设置 $AC_ACT_PROG_NET_TIME 为“0”。 $AC_OLD_PROG_NET_TIME 未改变。	
	3	停止 停止测量。 不改变 $AC_OLD_PROG_NET_TIME 并保持 $AC_ACT_PROG_NET_TIME 直至继续。	
	4	继续 继续测量，即再次接受一个以前停止的测量。 $AC_ACT_PROG_NET_TIME 继续运行。 $AC_OLD_PROG_NET_TIME 未改变。	
通过上电将所有系统变量复位为“0”！			
文献： 列出的系统变量的详细说明请参见： 功能手册，基本功能；BAG，通道，程序运行，复位特性（K1），章节：程序运行时间			

说明

机床制造商

可激活的定时器通过机床数据 MD27860 $MC_PROCESSTIMER_MODE 激活。

使用特定功能（例如 GOTOS，倍率 = 0%，生效的空运行进给，程序测试，ASSP，PROG_EVENT 等）时生效的时间测量特性通过机床数据 MD27850 $MC_PROG_NET_TIMER_MODE 和 MD27860 $MC_PROCESSTIMER_MODE 设置。

文献：

功能手册 基本功能；BAG，通道，程序运行，复位特性（K1），章节： 程序运行时间

说明

工件的剩余时间

如果需要依次加工相同的工件，可以由计时器值：

- 上次加工该工件的时间，参见 $AC_OLD_PROG_NET_TIME

以及

- 当前的加工时间，参见 $AC_ACT_PROG_NET_TIME

求得该工件的剩余时间。

除了当前加工时间，还会在操作界面上显示剩余时间。

注意

STOPRE 应用

系统变量 $AC_OLD_PROG_NET_TIME 和 $AC_OLD_PROG_NET_TIME_COUNT 不会产生隐含的预处理停止。 当系统变量值来自于预定的程序运行时，预处理停止在零件程序中无关紧要。 但是如果用于运行时间测量的触发器 ($AC_PROG_NET_TIME_TRIGGER) 高频写入，并且由此导致 $AC_OLD_PROG_NET_TIME 改变频繁，则零件程序中应使用一个明确定义的 STOPRE。

边界条件

- **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)。

示例

示例 1： 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
```

程序代码

```
N98 M30
```

在程序处理行 N80 后，在 $AC_OLD_PROG_NET_TIME 中有“mySubProgrammA”的净运行时间。

$AC_OLD_PROG_NET_TIME 值：

- 在 M30 后保持不变。
- 在每次完整运行循环后更新。

示例 2： 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
```

14.9.3 工件计数器

功能

使用“工件计数器”功能可提供各种不同的计数器，它们专用于在控制系统内部计算工件数量。

这些计数器作为通道专用的系统变量存在，带读写存取，值范围为 0 到 999 999 999。

系统变量	含义
$AC_REQUIRED_PARTS	待加工工件的数量（设定工件数量） 在此计数器中可以定义工件的个数，在到达这个数值之后，实际工件的个数($AC_ACTUAL_PARTS)复位为“0”。
$AC_TOTAL_PARTS	所有已加工工件的数量（总工件数量实际值） 该计数器给出所有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”。
$AC_ACTUAL_PARTS	所有已加工工件的数量（工件数量实际值） 在这种计数器中记录自开始时刻起所加工的所有工件数量。当达到设定工件数量时($AC_REQUIRED_PARTS)，该计数器就会自动归“0” ($AC_REQUIRED_PARTS > 0 是前提条件)。
$AC_SPECIAL_PARTS	用户计算的工件数量 该计数器允许用户根据自定义来对工件计数。 达到设定工件数量($AC_REQUIRED_PARTS)时可以定义一个报警输出。用户必须自行将该计数器归零。

说明

在控制系统按照缺省值启动时，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。

说明

使用通道专用的机床数据可以对计数器激活、归零时刻和计数算法进行设置。

说明

带用户定义 M 指令的工件计数

通过机床数据可以确定，通过用户定义的 M 指令来触发用于不同工件计数器的计数脉冲，而不是通过程序结束指令 M2/M30。

文献

关于“工件计数器”功能的其它信息，参见：

- 功能手册 基本功能；BAG，通道，程序运行，复位特性（K1），章节： 工件计数器

14.10 报警（SETAL）

功能

在一个 NC 程序中可以设置报警。 报警在操作界面中特殊区域内显示。 相应于一个报警，系统均有一个报警消除应答。

文献：
有关报警反应的其他信息请参见“调试手册”。

句法

```
SETAL（<报警号>）
SETAL(<报警号>,<字符串>)
```

含义

SETAL	用于报警编程的关键字。 SETAL 必须在一个 NC 程序段中编程。
<报警号>	INT 型变量。包括报警号。 报警号适用的范围 在 60000 和 69999 之间，其中 60000 到 64999 用于西门子循环，65000 到 69999 供用户使用。
<字符串>	对于用户循环报警编程可以另外规定一个字符串，最多 4 个参数。 在这些参数中，可以定义可更改的用户文本。 但还提供下列预定义参数：

参数	含义
%1	通道号
%2	程序段号，标签
%3	用于循环报警的文本索引
%4	补充的报警参数

说明
报警文本必须在操作界面中设计。

示例

程序代码	注释
...	
N100 SETAL (65000)	; 设置报警号 65000
...	

15 自有切割程序

15.1 用于切割的支持性功能

功能

您可以获得一个完整的加工循环用于切削。 由此您可以用以下所叙述的功能编制自身的切削程序：

- 设置轮廓表（CONTPRON）
- 设置轮廓表（CONTDCON）
- 断开轮廓预处理（EXECUTE）
- 计算两个轮廓元素之间的交点（INTERSEC）。

  （仅用于通过 CONTPRON 建立的表格。）
- 逐段执行某个图表的轮廓元素（EXECTAB）

  （仅用于通过 CONTPRON 建立的表格。）
- 计算圆的数据 (CALCDAT)

说明

您不仅可以在切削时用这些功能，而且也可以用于其它场合。

前提条件

在调用功能 CONTDCON 或 CONTDCON 之前必须：

- 返回到一个可以无轮廓冲突进行加工的起始点。
- 关断带 G40 的刀尖半径补偿。

15.2 设置轮廓表（CONTPRON）

功能

通过命令 CONTPRON 启用轮廓预处理。 不处理下列调用的 NC 程序段，而分布在各个运动中并存放在轮廓表格内。 每个轮廓单元相当于轮廓表格中二维数组的一个表格行。 所计算出的咬边个数送回。

句法

启用轮廓预处理：
CONTPRON(<轮廓表>,<处理类型>,<底切>,
<加工方向>)

断开轮廓预处理并且在正常处理模式中重新接通：
EXECUTE(<FEHLER>)

参见“ 断开轮廓预处理（EXECUTE）”

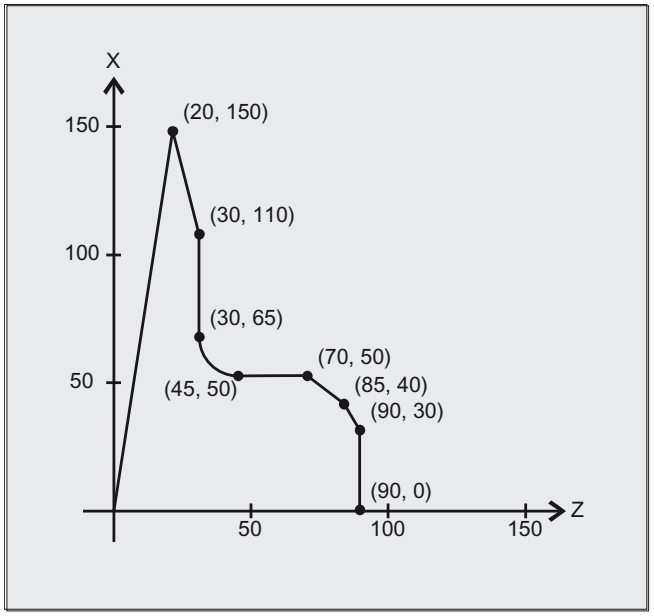
含义

CONTPRON	启用轮廓预处理命令用于创建轮廓表
<轮廓表>	轮廓表名称
<加工方式>	加工方式参数
	类型： CHAR
	值： "G" 纵向车削： 内部加工
	"L" 纵向车削： 外部加工
	"N" 端面车削： 内部加工
	"P" 端面车削： 外部加工
<底切>	出现的底切元素数目结果变量
	类型： INT
<加工方向>	加工方向参数
	类型： INT
	值： 0 向前轮廓预处理（标准值）
	1 轮廓预处理在两个方向上

示例 1

编制一个轮廓表格，包括：

- 名称“KTAB”
- 最多 30 个轮廓单元（圆弧，直线）
- 一个变量，表明所出现的底切元素数量
- 用于故障信息的一个变量



NC 程序：

程序代码	注释
N10 DEF REAL KTAB[30,11]	; 轮廓表包括名称 KTAB 和最多 30 个轮廓元素，参数值 11 (表格列数) 是一个固定值。
N20 DEF INT ANZHINT	; 名称为 ANZHINT 用于底切元素数量的变量。
N30 DEF INT FEHLER	; 故障反馈信息变量（0=没有故障，1=故障）。
N40 G18	
N50 CONTPRON (KTAB,"G",ANZHINT)	; 启用轮廓预处理。
N60 G1 X150 Z20	; N60 至 N120： 轮廓说明
N70 X110 Z30	
N80 X50 RND=15	
N90 Z70	
N100 X40 Z85	
N110 X30 Z90	
N120 X0	
N130 EXECUTE(FEHLER)	; 结束填写轮廓表，转换到正常程序运行方式。
N140	; 图表的其它处理。

轮廓表 KTAB：

索引 行	列									
(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
7	7	11	0	0	20	150	0	82.40535663	0	0
0	2	11	20	150	30	110	-1111	104.0362435	0	0
1	3	11	30	110	30	65	0	90	0	0
2	4	13	30	65	45	50	0	180	45	65
3	5	11	45	50	70	50	0	0	0	0
4	6	11	70	50	85	40	0	146.3099325	0	0
5	7	11	85	40	90	30	0	116.5650512	0	0
6	0	11	90	30	90	0	0	90	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

栏内容说明：

(0) 指针到下一个轮廓单元（同一个行号）

(1) 指针到前一个轮廓单元

(2) 编码用于运动的轮廓模式

X = abc 可能值

a = 10^2 G90 = 0 G91 = 1

b = 10^1 G70 = 0 G71 = 1

c = 10^0 G0 = 0 G1 = 1 G2 = 2 G3 = 3

(3), (4) 轮廓元素的始点

(3) = 横坐标， (4) = 当前平面中的纵坐标

(5), (6) 轮廓单元终点

(5) = 横坐标， (6) = 当前平面中的纵坐标

(7) 最大/最小指针： 标记轮廓中局部的最大和最小

(8) 轮廓元素和横坐标（当纵向加工时）或者纵坐标（当端面加工时）之间的最大值。 角度取决于所编程的加工方式。

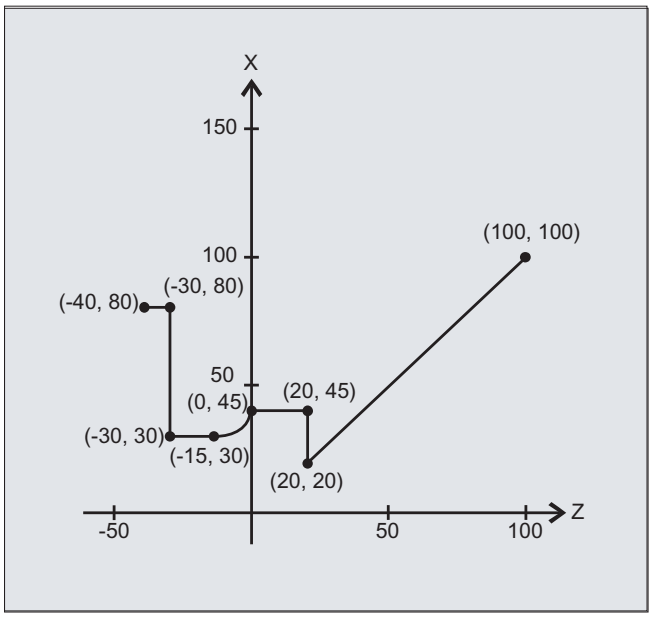
(9), (10) 如果是圆弧段，则轮廓单元的圆心坐标

(9) = 横坐标， (10) = 纵坐标

示例 2

编制一个轮廓表格，包括

- 名称 KTAB
- 最多 92 个轮廓单元（圆弧，直线）
- 工作方式 纵向车削，外侧加工
- 预处理，前进和后退



NC 程序：

程序代码	注释
N10 DEF REAL KTAB[92.11]	; 轮廓表包括名称 KTAB 和最多 92 个轮廓元素，参数 11 是一个固定量。
N20 DEF CHAR BT="L"	; CONTPRON 运行方式： 纵向车削，外侧加工
N30 DEF INT HE=0	; 底切元素数量=0
N40 DEF INT MODE=1	; 预处理，前进和后退
N50 DEF INT ERR=0	; 报警应答

程序代码	注释
...	
N100 G18 X100 Z100 F1000	
N105 CONTPRON(KTAB,BT,HE,MODE)	; 启用轮廓预处理。
N110 G1 G90 Z20 X20	
N120 X45	
N130 Z0	
N140 G2 Z-15 X30 K=AC(-15) I=AC(45)	
N150 G1 Z-30	
N160 X80	
N170 Z-40	
N180 EXECUTE(ERR)	; 结束填写轮廓表，转换到正常程序运行方式。
...	

轮廓表 KTAB：

在结束轮廓预处理之后，可以在两个方向使用轮廓。

索引	列										
行	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
0	6[1)]	7[2)]	11	100	100	20	20	0	45	0	0
1	0[3)]	2	11	20	20	20	45	-3	90	0	0
2	1	3	11	20	45	0	45	0	0	0	0
3	2	4	12	0	45	-15	30	5	90	-15	45
4	3	5	11	-15	30	-30	30	0	0	0	0
5	4	7	11	-30	30	-30	45	-1111	90	0	0
6	7	0[4)]	11	-30	80	-40	80	0	0	0	0
7	5	6	11	-30	45	-30	80	0	90	0	0
8	1[5)]	2[6)]	0	0	0	0	0	0	0	0	0
	...										
83	84	0[7)]	11	20	45	20	80	0	90	0	0
84	90	83	11	20	20	20	45	-1111	90	0	0
85	0[8)]	86	11	-40	80	-30	80	0	0	0	0
86	85	87	11	-30	80	-30	30	88	90	0	0
87	86	88	11	-30	30	-15	30	0	0	0	0

88	87	89	13	-15	30	0	45	-90	90	-15	45
89	88	90	11	0	45	20	45	0	0	0	0
90	89	84	11	20	45	20	20	84	90	0	0
91	83[9)]	85[10)]	11	20	20	100	100	0	45	0	0

栏内容说明和行 0、1、6、8、83、85 和 91 的备注

示例 1 中所述的栏内容说明有效。

始终在表格行 0：

1) 上一个： 行 n 包含向前的轮廓结束

2) 下一个： 行 n 是向前的轮廓表格结束

每一次在轮廓单元之内向前：

3) 上一个： 轮廓开始（向前）

4) 下一个： 轮廓结束（向前）

始终在轮廓表格行（向前）＋1：

5) 上一个： 咬边向前个数

6) 下一个： 咬边向后个数

每一次在轮廓单元之内向后：

7) 下一个： 轮廓结束（向后）

8) 上一个： 轮廓开始（向后）

始终在最后的表格行：

9) 上一个： 行 n 是轮廓表格起始（向后）

10) 下一个： 行 n 包含轮廓起始（向后）

其它信息

允许的运行指令，坐标系

下列 G 指令允许用于轮廓编程：

- G-组 1: G0, G1, G2, G3

最大可以是：

- 倒圆和倒角
- 圆编程通过 CIP 和 CT

样条、多项式和螺纹功能会导致出错。

不允许通过接通框架在 CONTPRON 和 EXECUTE 之间改变坐标系。 同时用于在 G70 和 G71 或 G700 和 G710 之间切换。

在预处理轮廓表格期间如果用 GEOAX 更换几何轴会导致报警。

咬边单元

单个的咬边单元的轮廓描述既可以在一个子程序中进行，也可以在单个程序段中进行。

与已编程的轮廓方向没有关系的切削

轮廓预处理通过 CONTPRON 已被扩展成在调用之后有独立于已编程方向的轮廓图表可供使用的型式。

15.3 设置轮廓表（CONTDCON）

功能

对于通过 CONTDCON 启用的轮廓预处理，下列调用的 NC 程序段以编码方式有效存放在一个 6 栏轮廓表中。 每个轮廓单元相当于轮廓表格中的一个表格行。 基于对下述编码规则的认识，例如来自图表行中的循环，可以组成 DIN 代码程序。 在号码 0 的表格行中，存储输出点的数据。

句法

启用轮廓预处理：
CONTDCON(<轮廓表>,<加工方向>)

断开轮廓预处理并且在正常处理模式中重新接通：
EXECUTE(<FEHLER>)

参见“ 断开轮廓预处理（EXECUTE）”

含义

CONTDCON	启用轮廓预处理命令用于创建一个编码的轮廓表		
<轮廓表>	轮廓表名称		
<加工方向>	加工方向参数		
	类型：	INT	
	值：	0	轮廓预处理根据轮廓程序段结果（标准值）
		1	不允许

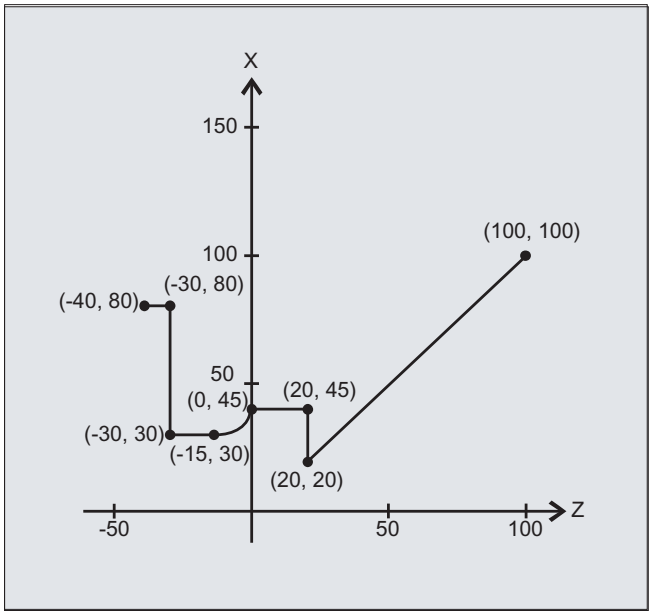
说明

在待列表的程序块中，允许用于 CONTPRON 的 G 代码比功能 CONTPRON 中的范围更广。除此之外，还将同时保存每个轮廓的进给和进给类型。

示例

编制一个轮廓表格，包括：

- 名称“KTAB”
- 轮廓单元（圆弧，直线）
- 工作方式车削
- 加工方向： 向前



NC 程序：

程序代码	注释
N10 DEF REAL KTAB[9,6]	; 名称为 KTAB 且有 9 个图表行的轮廓表。这些行允许 8 个轮廓程序段。 参数值 6（表的栏数）为固定值。
N20 DEF INT MODE = 0	; 加工方向变量。 默认值 0：仅在已编程的轮廓方向中。
N30 DEF INT ERROR = 0	; 反馈信息变量。
...	
N100 G18 G64 G90 G94 G710	
N101 G1 Z100 X100 F1000	
N105 CONTDCON (KTAB, MODE)	; 调用轮廓预处理（MODE 允许省略）。
N110 G1 Z20 X20 F200	; 轮廓说明。
N120 G9 X45 F300	
N130 Z0 F400	

程序代码	注释
N140 G2 Z-15 X30 K=AC(-15) I=AC(45)F100	
N150 G64 Z-30 F600	
N160 X80 F700	
N170 Z-40 F800	
N180 EXECUTE(ERROR)	; 结束填写轮廓表，转换到正常程序运行方式。
...	

轮廓表 KTAB：

	栏索引					
	0	1	2	3	4	5
行索引	轮廓模式	终点 横坐标	终点 纵坐标	中点 横坐标	中点 纵坐标	进给率
0	30	100	100	0	0	7
1	11031	20	20	0	0	200
2	111031	20	45	0	0	300
3	11031	0	45	0	0	400
4	11032	-15	30	-15	45	100
5	11031	-30	30	0	0	600
6	11031	-30	80	0	0	700
7	11031	-40	80	0	0	800
8	0	0	0	0	0	0

栏内容说明：

行 0： 编码**始点：**

列 0： 10^0 (个位): G0 = 0

10^1 (十位):G70 = 0, G71 = 1, G700 = 2, G710 = 3

列 1： 起始点横坐标

列 2： 起始点纵坐标

列 3－4： 0

列 5： 表格中最后轮廓段的行索引

行 1-n: **轮廓段**的记录

列 0： 10^0 (个位): G0 = 0, G1 = 1, G2 = 2, G3 = 3

10^1 (十位):G70 = 0, G71 = 1, G700 = 2, G710 = 3

10^2 (百位): G90 = 0, G91 = 1

10^3 (千位): G93 = 0, G94 = 1, G95 = 2, G96 = 3

10^4 (万位): G60 = 0, G44 = 1, G641 = 2, G642 = 3

10^5 (十万位): G9 = 1

列 1： 终点横坐标

列 2： 终点纵坐标

列 3： 圆心横坐标，在圆弧插补时

列 4： 圆心纵坐标，在圆弧插补时

列 5： 进给率

其它信息

允许的运行指令，坐标系

下列 G 组和 G 指令允许用于轮廓编程：

G-组 1: G0, G1, G2, G3

G-组 10: G60, G64, G641, G642

G-组 11: G9

G-组 13: G70, G71, G700, G710

G-组 14: G90, G91

G-组 15: G93, G94, G95, G96, G961

最大可以是：

- 倒圆和倒角
- 圆编程通过 CIP 和 CT

样条、多项式和螺纹功能会导致出错。

不允许通过接通框架在 CONTDCON 和 EXECUTE 之间改变坐标系。 同时用于在 G70 和 G71 或 G700 和 G710 之间切换。

在预处理轮廓表格期间如果用 GEOAX 更换几何轴会导致报警。

加工方向

使用 CONTDCON 生成的轮廓表可用于在已编程的轮廓方向中进行切削。

15.4 计算两个轮廓元素之间的交点（INTERSEC）

功能

INTERSEC 用来从使用 CONTPRON 生成的轮廓表中计算出两个已经过标准化处理的轮廓元素的交点。

句法

```
<Status>=INTERSEC(<轮廓表_1>[<轮廓元素_1>],
<轮廓表_2>[<轮廓元素_2>],<交点>,<加工方式>)
```

含义

INTERSEC	关键字用于从通过 CONTPRON 生成的轮廓表中确定第二个轮廓元素交点
<状态>	用于交点状态的变量 类型： BOOL 值： TRUE 找到交点 FALSE 没有找到交点
<轮廓表_1>	第一个轮廓表名称
<轮廓元素_1>	第一个轮廓表轮廓元素编号
<轮廓表_2>	第二个轮廓表名称
<轮廓元素_2>	第二个轮廓表轮廓元素编号
<交点>	激活平面中的交点坐标(G17 / G18 / G19) 类型： REAL
<加工方式>	加工方式参数 类型： INT 值： 0 在通过参数 2 激活的平面中计算交点（标准值） 1 交点计算与分配的平面无关

说明

请注意：变量必须在使用之前已经定义。

轮廓转换要求遵守用 CONTPRON 定义的值：

参数	含义
2	编码轮廓方式，用于运动
3	轮廓起始点横坐标
4	轮廓起始点纵坐标
5	轮廓终点横坐标
6	轮廓终点纵坐标
9	横坐标的中点坐标（仅对于圆弧轮廓）
10	纵坐标的中点坐标（仅对于圆弧轮廓）

示例

计算表格 TABNAME1 中的轮廓元素 3 与表格 TABNAME2 中轮廓元素 7 的交点。 活动平面中的交点坐标保存在变量 ISCOORD（第 1 项 = 横坐标，第 2 项= 纵坐标）中。 如果没有交点，则跳跃到 KEINSCH（没有找到交点）。

程序代码	注释
DEF REAL TABNAME1[12,11]	; 轮廓表 1
DEF REAL TABNAME2[10,11]	; 轮廓表 2
DEF REAL ISCOORD [2]	; 交点坐标变量。
DEF BOOL ISPOINT	; 交点状态变量。
DEF INT MODE	; 加工方式变量。
…	
MODE=1	; 计算与激活平面无关。
N10 ISPOINT=INTERSEC(TABNAME1[3],TABNAME2[7],ISCOORD,MODE)	; 调用轮廓元素的交点。
N20 IF ISPOINT==FALSE GOTOF KEINSCH	; 跳转到 KEINSCH。
…	

15.5 逐段执行某个图表的轮廓元素（EXECTAB）

功能

使用指令 EXECTAB 可以逐段执行某个图表（例如用指令 CONTPRON 生成的图表）的轮廓元素。

句法

EXECTAB(<轮廓表>[<轮廓元素>])

含义

EXECTAB	用于执行轮廓元素的指令
<轮廓表>	轮廓表名称
<轮廓元素>	轮廓元素编号

示例

表格 KTAB 的轮廓元素 0 至 2 应该逐段执行。

程序代码	注释
N10 EXECTAB (KTAB[0])	; 执行表格 KTAB 的元素 0
N20 EXECTAB (KTAB[1])	; 执行表格 KTAB 的元素 1
N30 EXECTAB (KTAB[2])	; 执行表格 KTAB 的元素 2

15.6 计算圆的数据 (CALCDAT)

功能

通过指令 CALCDAT 可以从三个或者四个已知的圆弧点计算半径和圆心坐标。 所给出的点必须不同。 如果是 4 个点，它们不是精确的在圆弧上，则选择一个平均值用于圆心和半径。

句法

```
<Status>=CALCDAT(<圆弧点>[<数目>,<类型>],<数目>,<结果>)
```

含义

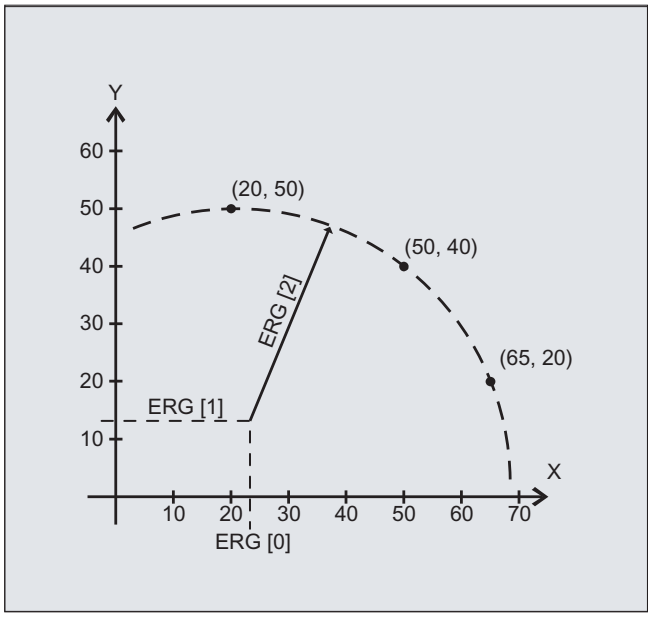
CALCDAT	用于从 3 个或 4 个点计算一个圆弧的半径和圆心坐标的指令
<状态>	用于圆弧计算状态的变量 类型： BOOL 值： TRUE 规定的点都位于圆弧上。 FALSE 规定的点都不在圆弧上。
<圆弧点>[]	变量用于 通过下列参数规定圆弧点： <数量> 圆弧点数目（3 或 4） <类型> 坐标数据类型， 例如 2 用于 2 点坐标系
<数量>	参数用于计算所使用点的数目（3 或 4）
<结果>[3]	用于结果的变量： 规定圆心坐标和半径 0 圆心： 横坐标值 1 圆心： 纵坐标值 2 半径

说明

请注意：变量必须在使用之前已经定义。

示例

由 3 个点计算出它们是否位于一个圆弧段。



程序代码	注释
N10 DEF REAL PKT[3,2]=(20,50,50,40,65,20)	; 变量用于规定圆弧点
N20 DEF REAL ERG[3]	; 用于结果的变量
N30 DEF BOOL STATUS	; 变量，用于状态
N40 STATUS=CALCDAT(PKT,3,ERG)	; 调用确定的圆弧数据。
N50 IF STATUS == FALSE GOTOF ERROR	; 跳转到故障

15.7 断开轮廓预处理（EXECUTE）

功能

使用指令 EXECUTE 来断开轮廓预处理并且同时在正常处理模式中重新接通。

句法

```
EXECUTE(<FEHLER>)
```

含义

EXECUTE	指令用于结束轮廓预处理
<故障>	故障反馈信息变量
	类型： INT
	变量值表示轮廓是否可以无故障加工：
	0 错误
	1 无故障

示例

程序代码

```
...
N30 CONTPRON(...)
N40 G1 X... Z...
...
N100 EXECUTE(...)
...
```

表 16

16.1 指令

图例:

1) 指令的有效性：
   m 模态
   s 逐段式

2) 资料参考，即包含指令详细说明的资料：
   PGsl 编程手册 基本原理
   PGAsl 编程手册 工作准备
   BNMsl 编程手册 测量循环
   BHDsl 操作手册 车床版
   BHFsl 操作手册 铣床版
   FB1 ( ) 功能手册 基本功能（括号中是相应功能的字母数字缩写）
   FB2 ( ) 功能手册 扩展功能（括号中是相应功能的字母数字缩写）
   FB3 ( ) 功能手册 特殊功能（括号中是相应功能的字母数字缩写）
   FBSIsl 功能手册 Safety Integrated
   FBSY 功能手册 同步动作
   FBW 功能手册 刀具管理

3) 程序初始的默认设置（若没有另行编程，即为控制系统的出厂设置）。

指令	含义	W 1)	说明参见 2)
:	NC 主程序段号、跳转标记结束、连接运算符		*PGAsl* 运算功能 (页 69)
*	乘法运算符		*PGAsl* 运算功能 (页 69)
+	加法运算符		*PGAsl* 运算功能 (页 69)
-	减法运算符		*PGAsl* 运算功能 (页 69)
<	关系运算符，小于号		*PGAsl* 运算功能 (页 69)

指令	含义	W [1]	说明参见 [2]
<<	字符串的连接运算符		*PGAsl* 运算功能 (页 69)
<=	关系运算符，小于等于		*PGAsl* 运算功能 (页 69)
=	赋值运算符		*PGAsl* 运算功能 (页 69)
>=	关系运算符，大于等于		*PGAsl* 运算功能 (页 69)
/	除法运算符		*PGAsl* 运算功能 (页 69)
/0 … … /7	程序段跳转（第 1 跳转级） 程序段跳转（第 8 跳转级）		*PGsl*
A	轴名称	m/s	*PGAsl* 编程刀具定向 (A..., B..., C..., LEAD, TILT) (页 338)
A2	刀具定向： RPY 角或欧拉角	s	*PGAsl* 编程刀具定向 (A..., B..., C..., LEAD, TILT) (页 338)
A3	刀具定向： 方向法线/平面法线的矢量分量	s	*PGAsl* 编程刀具定向 (A..., B..., C..., LEAD, TILT) (页 338)
A4	刀具定向： 程序段开头的平面法线矢量	s	*PGAsl* 端面铣削 (3D-铣削 A4, B4, C4, A5, B5, C5) (页 344)
A5	刀具定向： 程序段结尾的平面法线矢量	s	*PGAsl* 端面铣削 (3D-铣削 A4, B4, C4, A5, B5, C5) (页 344)
ABS	绝对值		*PGAsl* 运算功能 (页 69)

指令	含义	W [1]	说明参见 [2]
AC	坐标/位置的绝对尺寸值	s	*PGsl*
ACC	当前轴向加速度的控制	m	*PGsl*
ACCLIMA	当前最大轴向加速度的控制	m	*PGsl*
ACN	绝对尺寸值，用于回转轴运行到负向上的某个位置	s	*PGsl*
ACOS	反余弦 （三角 函数）		*PGAsl* 运算功能 (页 69)
ACP	绝对尺寸值，用于回转轴运行到正向上的某个位置	s	*PGsl*
ACTBLOCNO	输出报警程序段的当前编号，即使当前程序段显示被抑制 (DISPLOF)！		*PGAsl* 抑制当前的程序段显示(DISPLOF, DISPLON, ACTBLOCNO) (页 175)
ADDFRAME	计算并激活测得的框架		*PGAsl, FB1(K2)* 从空间中的三个测量点计算框架 (MEAFRAME) (页 311)
ADIS	用于路径功能 G1, G2, G3, ... 的平滑距离	m	*PGsl*
ADISPOS	用于快速移动 G0 的平滑距离	m	*PGsl*
ADISPOSA	用于 IPOBRKA 的公差窗口尺寸	m	*PGAsl* 可编程的运动结束条件 (FINEA, COARSEA, IPOENDA, IPOBRKA, ADISPOSA) (页 285)
ALF	快速退刀角度	m	*PGAsl* 快速离开工件轮廓（SETINT LIFTFAST，ALF） (页 125)
AMIRROR	可编程镜像	s	*PGsl*

指令	含义	W [1]	说明参见 [2]
AND	逻辑与		*PGAs/* 比较运算和逻辑运算 (页 72)
ANG	轮廓线角度	s	*PGs/*
AP	极角	m/s	*PGs/*
APR	读取/显示的存取保护		*PGAs/* 属性： 存取权限(APR, APW, APRP, APWP, APRB, APWB) (页 44)
APRB	读取 BTSS 的权限		*PGAs/* 属性： 存取权限(APR, APW, APRP, APWP, APRB, APWB) (页 44)
APRP	读取零件程序的权限		*PGAs/* 属性： 存取权限(APR, APW, APRP, APWP, APRB, APWB) (页 44)
APW	写存取保护		*PGAs/* 属性： 存取权限(APR, APW, APRP, APWP, APRB, APWB) (页 44)
APWB	写入 BTSS 的权限		*PGAs/* 属性： 存取权限(APR, APW, APRP, APWP, APRB, APWB) (页 44)
APWP	写入零件程序的权限		*PGAs/* 属性： 存取权限(APR, APW, APRP, APWP, APRB, APWB) (页 44)
APX	存取保护定义，用于执行指定的语言单元		*PGAs/* 系统变量，用户变量和 NC 语言指令的重新定义（REDEF） (页 32)
AR	张角	m/s	*PGs/*
AROT	可编程旋转	s	*PGs/*

指令	含义	W [1]	说明参见 [2]
AROTS	可编程的框架旋转，带立体角	s	*PGsl*
AS	宏指令定义		*PGAsl* 宏指令技术 (DEFINE ... AS) (页 210)
ASCALE	可编程缩放	s	*PGsl*
ASIN	反正弦算术函数		*PGAsl* 运算功能 (页 69)
ASPLINE	Akima 样条	m	*PGAsl* 样条插补 (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (页 243)
ATAN2	反正切 2		*PGAsl* 运算功能 (页 69)
ATOL	用于压缩器功能、方向平滑以及平滑方式的轴专有公差		*PGAsl* 可编程的轮廓公差/定向公差(CTOL, OTOL, ATOL) (页 505)
ATRANS	可编程附加偏移	s	*PGsl*
AX	可变的轴标识符	m/s	*PGAsl* 轴功能 (AXNAME, AX, SPI, AXTOSPI, ISAXIS, AXSTRING, MODAXVAL) (页 685)
AXCTSWE	容器轴相连		*PGAsl* 轴容器 (AXCTSWE, AXCTSWED) (页 693)
AXCTSWED	轴容器旋转		*PGAsl* 轴容器 (AXCTSWE, AXCTSWED) (页 693)
AXIS	轴标识符、轴地址		*PGAsl* 定义用户变量（DEF） (页 25)
AXNAME	把输入字符串转换为一个轴标识符		*PGAsl* 轴功能 (AXNAME, AX, SPI, AXTOSPI, ISAXIS, AXSTRING, MODAXVAL) (页 685)

指令	含义	W [1]	说明参见 [2]
AXSTRING	把字符串转换为主轴号		*PGAsl* 轴功能 (AXNAME, AX, SPI, AXTOSPI, ISAXIS, AXSTRING, MODAXVAL) (页 685)
AXTOCHAN	指定轴为某一特定通道。 由 NC 程序和从同步动作都可以。		*PGAsl* 将轴移交到另一个通道中（AXTOCHAN） (页 137)
AXTOSPI	将轴标识符转换为一个主轴索引		*PGAsl* 轴功能 (AXNAME, AX, SPI, AXTOSPI, ISAXIS, AXSTRING, MODAXVAL) (页 685)
B	轴名称	m/s	*PGAsl* 编程刀具定向 (A..., B..., C..., LEAD, TILT) (页 338)
B2	刀具定向： RPY 角或欧拉角	s	*PGAsl* 编程刀具定向 (A..., B..., C..., LEAD, TILT) (页 338)
B3	刀具定向： 方向法线/平面法线的矢量分量	s	*PGAsl* 编程刀具定向 (A..., B..., C..., LEAD, TILT) (页 338)
B4	刀具定向： 程序段开头的平面法线矢量	s	*PGAsl* 端面铣削 (3D-铣削 A4, B4, C4, A5, B5, C5) (页 344)
B5	刀具定向： 程序段结尾的平面法线矢量	s	*PGAsl* 端面铣削 (3D-铣削 A4, B4, C4, A5, B5, C5) (页 344)
B_AND	位方式“与”		*PGAsl* 比较运算和逻辑运算 (页 72)
B_OR	位方式“或”		*PGAsl* 比较运算和逻辑运算 (页 72)
B_NOT	位方式“非”		*PGAsl* 比较运算和逻辑运算 (页 72)
B_XOR	位方式“异－或”		*PGAsl* 比较运算和逻辑运算 (页 72)

指令	含义	W [1]	说明参见 [2]
BAUTO	通过后面的 3 个点定义样条段	m	*PGAsl* 样条插补 (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (页 243)
BLOCK	和关键字 TO 一起，在一个间接的子程序调用中定义一个需要处理的程序部分		*PGAsl* 指定待执行部分的间接子程序调用(CALL BLOCK ... TO ...) (页 197)
BLSYNC	在下一个程序段转换时才开始处理中断程序		*PGAsl* 中断程序赋值和启动(SETINT, PRIO, BLSYNC) (页 121)
BNAT [3]	自然过渡到第一个样条程序段	m	*PGAsl* 样条插补 (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (页 243)
BOOL	数据类型： 真值 TRUE/FALSE 或者 1/0		*PGAsl* 定义用户变量（DEF） (页 25)
BOUND	检查，值是否在已定义的值域中。同时返回检验值。		*PGAsl* 参见“最大变量、最小变量和变量区域(MINVAL, MAXVAL, BOUND)” (页 77)
BRISK [3]	跃变式的轨迹加速度	m	*PGsl*
BRISKA	激活编程轴跃变式的轨迹加速度		*PGsl*
BSPLINE	B 样条	m	*PGAsl* 样条插补 (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (页 243)
BTAN	切线过渡到第一个样条程序段	m	*PGAsl* 样条插补 (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (页 243)

指令	含义	W 1)	说明参见 2)
C	轴名称	m/s	*PGAsl* 编程刀具定向 (A..., B..., C..., LEAD, TILT) (页 338)
C2	刀具定向：RPY 角或欧拉角	s	*PGAsl* 编程刀具定向 (A..., B..., C..., LEAD, TILT) (页 338)
C3	刀具定向： 方向法线/平面法线的矢量分量	s	*PGAsl* 编程刀具定向 (A..., B..., C..., LEAD, TILT) (页 338)
C4	刀具定向：程序段开头的平面法线矢量	s	*PGAsl* 端面铣削 (3D-铣削 A4, B4, C4, A5, B5, C5) (页 344)
C5	刀具定向：程序段结尾的平面法线矢量	s	*PGAsl* 端面铣削 (3D-铣削 A4, B4, C4, A5, B5, C5) (页 344)
CAC	运行至某个绝对位置		*PGAsl* 逼近已经过编码处理的位置 (CAC, CIC, CDC, CACP, CACN) (页 241)
CACN	从负方向运行至表中某数值所在位置（绝对值）		*PGAsl* 逼近已经过编码处理的位置 (CAC, CIC, CDC, CACP, CACN) (页 241)
CACP	从正方向运行至表中某数值所在位置（绝对值）		*PGAsl* 逼近已经过编码处理的位置 (CAC, CIC, CDC, CACP, CACN) (页 241)
CALCDAT	从 3 个或者 4 个点中计算某个圆的半径和中点。		*PGAsl* 计算圆的数据 (CALCDAT) (页 735)
CALCPOSI	检查是否超出保护区、工作范围限制和软件限位开关		*PGAsl* 检查超出保护区的情况、工作区域限制和软件极限值(CALCPOSI) (页 231)
CALL	间接子程序调用		*PGAsl* 间接子程序调用(CALL) (页 196)

指令	含义	W [1]	说明参见 [2]
CALLPATH	子程序调用时可编程的查找路径		*PGAsl* 扩展调用子程序时的路径查找 (CALLPATH) (页 200)
CANCEL	中止模态同步动作		*PGAsl* 删除同步动作 (CANCEL) (页 650)
CASE	有条件程序跳转		*PGAsl* 程序分支(CASE ... OF ... DEFAULT ...) (页 95)
CDC	直接运行至某位置		*PGAsl* 逼近已经过编码处理的位置 (CAC, CIC, CDC, CACP, CACN) (页 241)
CDOF [3]	关闭碰撞监控	m	*PGsl*
CDOF2	关闭碰撞监控，3D 圆周铣削时	m	*PGsl*
CDON	启用碰撞监控	m	*PGsl*
CFC [3]	轮廓处的恒定进给	m	*PGsl*
CFIN	仅内曲面上的恒定进给，而不是外曲面上	m	*PGsl*
CFINE	为 FRAME 变量赋值一个精偏		*PGAsl* 粗偏移和精偏移 (CFINE，CTRANS) (页 306)
CFTCP	刀尖基准点、中心轨迹上的恒定进给	m	*PGsl*
CHAN	规定数据有效区。		*PGAsl* 定义用户变量（DEF） (页 25)
CHANDATA	设定通道号，用于通道数据存取		*PGAsl* 工作存储器 (CHANDATA, COMPLETE, INITIAL) (页 219)

指令	含义	W 1)	说明参见 2)
CHAR	数据类型： ASCII-字符		*PGAsl* 定义用户变量（DEF） (页 25)
CHECKSUM	通过一个字符串数组构成校验和，带一个确定的长度		*PGAsl* 通过数组计算校验和(CHECKSUM) (页 154)
CHF	倒角； 值＝ 倒角长度	s	*PGsl*
CHKDM	刀库的单一性检查		*FBW*
CHKDNO	D 号的单一性检查		*PGAsl* 任意 D 编号赋值:检查 D 号码(CHKDNO) (页 445)
CHR	倒角； 值＝倒角长度，在运动方向		*PGsl*
CIC	运行至表中某数值位置（增量值）		*PGAsl* 逼近已经过编码处理的位置 (CAC, CIC, CDC, CACP, CACN) (页 241)
CIP	通过中间点进行圆弧插补	m	*PGsl*
CLEARM	复位一个/多个标号，用于通道协调		*PGAsl* 程序协调（INIT，START，WAITM，WAITMC，WAITE，SETM，CLEARM） (页 113)
CLRINT	撤销选择中断		*PGAsl* 删除中断程序的赋值（CLRINT） (页 123)
CMIRROR	对一个坐标轴的镜像		*PGAsl* 运算功能 (页 69)
COARSEA	在到达“粗准停”时运行结束	m	*PGAsl* 可编程的运动结束条件 (FINEA, COARSEA, IPOENDA, IPOBRKA, ADISPOSA) (页 285)
COMPCAD	激活压缩器： 优化 CAD 程序的表面质量	m	*PGAsl* NC 程序段压缩 (COMPON, COMPCURV, COMPCAD, COMPOF) (页 257)

指令	含义	W [1]	说明参见 [2]
COMPCURV	激活压缩器： 曲率不变的多项式	m	*PGAsl* NC 程序段压缩 (COMPON, COMPCURV, COMPCAD, COMPOF) (页 257)
COMPLETE	读入和读出数据的控制指令		PGAsl 工作存储器 (CHANDATA, COMPLETE, INITIAL) (页 219)
COMPOF [3]	关闭压缩器	m	*PGAsl* NC 程序段压缩 (COMPON, COMPCURV, COMPCAD, COMPOF) (页 257)
COMPON	激活压缩器		*PGAsl* NC 程序段压缩 (COMPON, COMPCURV, COMPCAD, COMPOF) (页 257)
CONTDCON	启用表格形式的轮廓解码		*PGAsl* 设置轮廓表（CONTDCON） (页 727)
CONTPRON	激活参考点处理		*PGAsl* 设置轮廓表（CONTPRON） (页 720)
CORROF	取消所有有效的叠加运动。		*PGsl*
COS	余弦 （三角 函数）		*PGAsl* 运算功能 (页 69)
COUPDEF	定义 ELG 组合/同步主轴组合		*PGAsl* 同步主轴： 编程（COUPDEF, COUPDEL, COUPON, COUPONC, COUPOF, COUPOFS, COUPRES, WAITC） (页 550)
COUPDEL	删除 ELG 组合		*PGAsl* 同步主轴： 编程（COUPDEF, COUPDEL, COUPON, COUPONC, COUPOF, COUPOFS, COUPRES, WAITC） (页 550)
COUPOF	激活 ELG 组合/同步主轴对。		*PGAsl* 同步主轴： 编程（COUPDEF, COUPDEL, COUPON, COUPONC, COUPOF, COUPOFS, COUPRES, WAITC） (页 550)

指令	含义	W [1]	说明参见 [2]
COUPOFS	关闭 ELG 组合/同步主轴对，停止跟随主轴		*PGAsl* 同步主轴： 编程（COUPDEF, COUPDEL, COUPON, COUPONC, COUPOF, COUPOFS, COUPRES, WAITC） (页 550)
COUPON	激活 ELG 组合/同步主轴对。		*PGAsl* 同步主轴： 编程（COUPDEF, COUPDEL, COUPON, COUPONC, COUPOF, COUPOFS, COUPRES, WAITC） (页 550)
COUPONC	激活 ELG 组合/同步主轴对，并采用上一次编程		*PGAsl* 同步主轴： 编程（COUPDEF, COUPDEL, COUPON, COUPONC, COUPOF, COUPOFS, COUPRES, WAITC） (页 550)
COUPRES	复位 ELG 组合		*PGAsl* 同步主轴： 编程（COUPDEF, COUPDEL, COUPON, COUPONC, COUPOF, COUPOFS, COUPRES, WAITC） (页 550)
CP	轨迹运行	m	*PGAsl* 直角坐标 PTP运动 (页 390)
CPRECOF [3]	取消可编程轮廓精度	m	*PGsl*
CPRECON	启用可编程轮廓精度	m	*PGsl*
CPROT	激活/取消通道专用的保护区		*PGAsl* 激活/取消激活保护区 (CPROT, NPROT) (页 227)
CPROTDEF	定义通道专用的保护区		*PGAsl* 保护区的确定 (CPROTDEF, NPROTDEF) (页 223)
CR	圆弧半径	s	*PGsl*
CROT	旋转当前坐标系		*PGAsl* 运算功能 (页 69)

指令	含义	W [1]	说明参见 [2]
CROTS	以立体角进行可编程旋转（在指定轴旋转）	s	*PGsl*
CRPL	在任意平面内旋转框架		*FB1(K2)*
CSCALE	比例系数，用于多个轴		*PGAsl* 运算功能 (页 69)
CSPLINE	立方样条	m	*PGAsl* 样条插补 (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (页 243)
CT	切线过渡的圆弧	m	*PGsl*
CTAB	依据曲线表中的引导轴位置计算跟轴位置		*PGAsl* 读取曲线图表值 (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX) (页 529)
CTABDEF	激活表格定义		*PGAsl* 定义曲线图表(CTABDEF, CATBEND) (页 517)
CTABDEL	删除曲线表		*PGAsl* 删除曲线图表(CTABDEL) (页 524)
CTABEND	取消表格定义		*PGAsl* 定义曲线图表(CTABDEF, CATBEND) (页 517)
CTABEXISTS	检查带号 n 的曲线表		*PGAsl* 检查曲线图表的存在性(CTABEXISTS) (页 524)

指令	含义	W [1]	说明参见 [2]
CTABFNO	存储器中尚可使用的曲线表个数		*PGAs/* 曲线图表： 检查资源使用率(CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL) (页 534)
CTABFPOL	存储器中尚可使用的多项式个数		*PGAs/* 曲线图表： 检查资源使用率(CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL) (页 534)
CTABFSEG	存储器中尚可使用的曲线段个数		*PGAs/* 曲线图表： 检查资源使用率(CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL) (页 534)
CTABID	提供曲线表的表编号		*PGAs/* 曲线图表： 确定图表属性(CTABID, CTABISLOCK, CTABMEMTYP, CTABPERIOD) (页 527)
CTABINV	依据曲线表中的跟随轴位置计算引导轴位置		*PGAs/* 读取曲线图表值 (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX) (页 529)
CTABISLOCK	返回 n 号的曲线表的禁止状态		*PGAs/* 曲线图表： 确定图表属性(CTABID, CTABISLOCK, CTABMEMTYP, CTABPERIOD) (页 527)
CTABLOCK	禁止删除和改写		*PGAs/* 禁止删除和覆盖曲线图表(CTABLOCK, CTABUNLOCK) (页 526)

指令	含义	W [1]	说明参见 [2]
CTABMEMTYP	返回存储器，在该存储器中编制了 n 号的曲线表。		*PGAsl* 曲线图表： 确定图表属性(CTABID, CTABISLOCK, CTABMEMTYP, CTABPERIOD) (页 527)
CTABMPOL	存储器中最大可用的多项式个数		*PGAsl* 曲线图表： 检查资源使用率(CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL) (页 534)
CTABMSEG	存储器中最大可用的曲线段个数		*PGAsl* 曲线图表： 检查资源使用率(CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL) (页 534)
CTABNO	在 SRAM 或者 DRAM 中定义的曲线表的个数		*FB3(M3)*
CTABNOMEM	在 SRAM 或者 DRAM 中定义的曲线表的个数		*PGAsl* 曲线图表： 检查资源使用率(CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL) (页 534)
CTABPERIOD	返回编号为 n 的曲线表的周期性数据		*PGAsl* 曲线图表： 确定图表属性(CTABID, CTABISLOCK, CTABMEMTYP, CTABPERIOD) (页 527)
CTABPOL	存储器中已经使用的多项式个数		*PGAsl* 曲线图表： 检查资源使用率(CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL) (页 534)

指令	含义	W 1)	说明参见 2)
CTABPOLID	n 号曲线表所使用的曲线多项式个数		*PGAsl* 曲线图表： 检查资源使用率(CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL) (页 534)
CTABSEG	存储器中已经使用的曲线段的个数		*PGAsl* 曲线图表： 检查资源使用率(CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL) (页 534)
CTABSEGID	n 号曲线表所使用的曲线段个数		*PGAsl* 曲线图表： 检查资源使用率(CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL) (页 534)
CTABSEV	提供曲线表一个分段的跟随轴终值		*PGAsl* 读取曲线图表值 (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX) (页 529)
CTABSSV	提供曲线表一个分段的跟随轴起始值		*PGAsl* 读取曲线图表值 (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX) (页 529)
CTABTEP	提供曲线表结束处提供引导轴的值		*PGAsl* 读取曲线图表值 (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX) (页 529)

指令	含义	W [1]	说明参见 [2]
CTABTEV	提供曲线表结束处跟随轴的值		*PGAsl* 读取曲线图表值 (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX) (页 529)
CTABTMAX	提供曲线表跟随轴的最大值		*PGAsl* 读取曲线图表值 (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX) (页 529)
CTABTMIN	提供曲线表跟随轴的最小值		*PGAsl* 读取曲线图表值 (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX) (页 529)
CTABTSP	提供曲线表开始处引导轴的值		*PGAsl* 读取曲线图表值 (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX) (页 529)
CTABTSV	提供曲线表开始处跟随轴的值		*PGAsl* 读取曲线图表值 (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX) (页 529)
CTABUNLOCK	取消禁止删除和改写		*PGAsl* 禁止删除和覆盖曲线图表(CTABLOCK, CTABUNLOCK) (页 526)
CTOL	用于压缩器功能、定向平滑和平滑方式的轮廓公差		*PGAsl* 可编程的轮廓公差/定向公差(CTOL, OTOL, ATOL) (页 505)
CTRANS	零点偏移，用于多个轴		*PGAsl* 粗偏移和精偏移 (CFINE，CTRANS) (页 306)

指令	含义	W [1]	说明参见 [2]
CUT2D [3]	2D 刀具补偿	m	*PGsl*
CUT2DF	2D 刀具补偿 刀具补偿相对于当前框架有效（倾斜平面）。	m	*PGsl*
CUT3DC	圆周铣削 3D 刀具补偿	m	*PGAsl* 激活 3D-刀具补偿 (CUT3DC..., CUT3DF...) (页 425)
CUT3DCC	3D 圆周铣削刀具补偿，带限制表面	m	*PGAsl* 3D 刀具半径补偿：考虑一个限制面 (CUT3DCC, CUT3DCCD) (页 434)
CUT3DCCD	3D 圆周铣削刀具补偿，带不同刀具的限制表面	m	*PGAsl* 3D 刀具半径补偿：考虑一个限制面 (CUT3DCC, CUT3DCCD) (页 434)
CUT3DF	端面铣 3D 刀具半径补偿	m	*PGAsl* 激活 3D-刀具补偿 (CUT3DC..., CUT3DF...) (页 425)
CUT3DFF	3D 刀具补偿端面铣削，带恒定的刀具定向，与有效框架有关	m	*PGAsl* 激活 3D-刀具补偿 (CUT3DC..., CUT3DF...) (页 425)
CUT3DFS	3D 刀具补偿端面铣削，带恒定的刀具定向，与有效框架无关	m	*PGAsl* 激活 3D-刀具补偿 (CUT3DC..., CUT3DF...) (页 425)
CUTCONOF [3]	取消恒定半径补偿	m	*PGsl*
CUTCONON	启用恒定半径补偿	m	*PGsl*
CUTMOD	开启“可旋转刀具的补偿数据修改”功能		*PGAsl* 可旋转刀具的切削刃数据修改 (CUTMOD) (页 461)
CYCLE...	工艺循环		*BNMsl*

指令	含义	W [1]	说明参见 [2]
D	刀具补偿号		*PGsl*
D0	如果编程 D0，则刀具的补偿无效。		*PGsl*
DAC	绝对、逐段、轴专用的直径编程	s	*PGsl*
DC	绝对尺寸参数，用于回转轴直接运行到某个位置	s	*PGsl*
DEF	变量定义		*PGAsl* 定义用户变量（DEF） (页 25)
DEFINE	用于宏指令定义的关键字		*PGAsl* 宏指令技术 (DEFINE ... AS) (页 210)
DEFAULT	跳转到 CASE 回路		*PGAsl* 程序分支(CASE ... OF ... DEFAULT ...) (页 95)
DELAYFSTON	定义一个停止延迟区的开始	m	*PGAsl* 程序分支(CASE ... OF ... DEFAULT ...) (页 95)
DELAYFSTOF	定义一个停止延迟区的结尾	m	*PGAsl* 可以有条件中断的程序段 (DELAYFSTON, DELAYFSTOF) (页 482)
DELDL	清除附加补偿		*PGAsl* 清除附加补偿（DELDL） (页 409)
DELDTG	剩余行程删除		*PGAsl* 删除剩余行程（DELDTG） (页 599)
DELETE	删除指定的文件。 文件名可以用路径和文件标识给出。		*PGAsl* 删除文件（DELETE） (页 143)
DELTOOLENV	删除用于说明刀具环境的数据段		*FB1(W1)*

指令	含义	W [1]	说明参见 [2]
DIACYCOFA	轴专用、模态的直径编程： 循环中的“关”	m	*FB1(P1)*
DIAM90	G90：直径编程；G91：半径编程	m	*PGAsl*
DIAM90A	G90 和 AC：轴专用、模态的直径编程；G91 和 IC：半径编程	m	*PGsl*
DIAMCHAN	MD 轴功能中的所有轴将接收直径编程的通道状态		*PGsl*
DIAMCHANA	接收直径编程的通道状态		*PGsl*
DIAMCYCOF	通道专用的直径编程： 在循环中关闭	m	*FB1(P1)*
DIAMOF [3]	直径编程： 关 参见机床制造商的初始设置	m	*PGsl*
DIAMOFA	轴专用的模态直径编程： 关 参见机床制造商的初始设置	m	*PGsl*
DIAMON	直径编程： 激活	m	*PGsl*
DIAMONA	轴专用的模态直径编程： 开启 参见机床制造商的定义	m	*PGsl*
DIC	相对、逐段、轴专用的直径编程	s	*PGsl*
DILF	返回行程（长度）	m	*PGsl*
DISABLE	中断“关”		*PGAsl* 取消/再激活一个中断程序的赋值（DISABLE，ENABLE） (页 122)
DISC	过渡圆弧刀具半径补偿加强	m	*PGsl*
DISCL	快速进刀终点和加工平面的间距		*PGsl*

指令	含义	W [1]	说明参见 [2]
DISPLOF	抑制当前的程序段显示		*PGAsl* 抑制当前的程序段显示(DISPLOF, DISPLON, ACTBLOCNO) (页 175)
DISPLON	恢复当前程序段显示		*PGAsl* 抑制当前的程序段显示(DISPLOF, DISPLON, ACTBLOCNO) (页 175)
DISPR	Repos(再定位)-轨迹差值	s	*PGAsl* 返回轮廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (页 491)
DISR	Repos（再定位)距离	s	*PGAsl* 返回轮廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (页 491)
DITE	螺纹导出行程	m	*PGsl*
DITS	螺纹导入行程	m	*PGsl*
DIV	整除		*PGAsl* 运算功能 (页 69)
DL	选择和地点无关的附加刀具补偿（DL、总调整补偿）	m	*PGAsl* 选择附加补偿（DL） (页 407)
DO	同步动作的关键字，在满足条件时动作触发		*PGAsl* 动作 (DO) (页 573)
DRFOF	取消手轮偏移（DRF）	m	*PGsl*
DRIVE	与速度相关的轨迹加速度	m	*PGsl*
DRIVEA	激活编程轴的折弯型加速度特征曲线		*PGsl*

指令	含义	W [1]	说明参见 [2]
DYNFINISH	精加工动态响应	m	*PGsl*
DYNNORM	常规动态响应	m	*PGsl*
DYNPOS	定位模式、攻丝的动态响应	m	*PGsl*
DYNROUGH	粗加工动态响应	m	*PGsl*
DYNSEMIFIN	精加工动态响应	m	*PGsl*
DZERO	将 TO 单元的所有 D 编号标识为无效		*PGAsl* 任意 D 编号赋值：设定无效的 D 编号 (DZERO) (页 448)
EAUTO	通过前面的 3 个点定义前一个样条段	m	*PGAsl* 样条插补 (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (页 243)
EGDEF	电子齿轮定义		*PGAsl* 定义电子齿轮箱 (EGDEF) (页 542)
EGDEL	删除跟随轴的耦合定义		*PGAsl* 删除某个电子齿轮箱的定义（EGDEL） (页 548)
EGOFC	持续取消电子齿轮		*PGAsl* 关闭电子齿轮（EGOFS, EGOFC） (页 547)
EGOFS	选择性取消电子齿轮		*PGAsl* 关闭电子齿轮（EGOFS, EGOFC） (页 547)
EGON	启用电子齿轮		*PGAsl* 关闭电子齿轮（EGOFS, EGOFC） (页 547)
EGONSYN	启用电子齿轮		*PGAsl* 关闭电子齿轮（EGOFS, EGOFC） (页 547)

指令	含义	W [1]	说明参见 [2]
EGONSYNE	启用电子齿轮，按照预定的起动模式		*PGAsl* 关闭电子齿轮（EGOFS, EGOFC） (页 547)
ELSE	当 IF 条件不满足时，程序跳转		*PGAsl* 带选项的程序循环 (IF, ELSE, ENDIF) (页 106)
ENABLE	中断“开”		*PGAsl* 取消/再激活一个中断程序的赋值（DISABLE，ENABLE） (页 122)
ENAT [3]	自然过渡到下一个运行程序段	m	*PGAsl* 样条插补 (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (页 243)
ENDFOR	FOR 计数循环的结束行		*PGAsl* 计数循环（FOR ... TO ...，ENDFOR） (页 108)
ENDIF	IF 跳转的结束行		*PGAsl* 带选项的程序循环 (IF, ELSE, ENDIF) (页 106)
ENDLABEL	零件程序通过 REPEAT 重复的结束标记		*PGAsl, FB1(K1)* 程序部分重复 (REPEAT, REPEATB, ENDLABEL, P) (页 98)
ENDLOOP	无限程序循环 LOOP 结束行		*PGAsl* 无限程序循环（LOOP，ENDLOOP） (页 107)
ENDPROC	带起始行 PROC 的一个程序的结束行		
ENDWHILE	WHILE-循环的结束行		*PGAsl* 在循环开始处带有条件的程序循环（WHILE，ENDWHILE） (页 110)
ETAN	在样条开始以切线过渡到下一个运行程序段	m	*PGAsl* 样条插补 (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (页 243)

指令	含义	W [1]	说明参见 [2]
EVERY	执行同步动作，当条件从 FALSE 过渡到 TRUE 时		*PGAsl* 条件循环检查（WHEN, WHENEVER, FROM, EVERY） (页 571)
EX	用于冥数运算法则中赋值的关键字		*PGAsl* 预定义用户变量： 计算参数（R） (页 21)
EXECSTRING	传递一个字符串变量，包含待执行的零件程序行		*PGAsl* 间接编程零件程序行（EXECSTRING） (页 67)
EXECTAB	执行来自运动表中的元素		*PGAsl* 间接编程零件程序行（EXECSTRING） (页 67)
EXECUTE	激活程序执行		*PGAsl* 断开轮廓预处理（EXECUTE） (页 737)
EXP	指数函数 ex		*PGAsl* 运算功能 (页 69)
EXTCALL	执行外部子程序		*PGAsl* 执行外部子程序 (EXTCALL) (页 202)
EXTERN	申明一个子程序，带参数传递		*PGAsl* 没有参数传递的子程序调用 (页 188)
F	进给值 （和 G4 一起，同时在 F 中编程暂停时间）		*PGsl*
FA	轴向进给	m	*PGsl*
FAD	横向进给，用于平滑逼近和离开		*PGsl*
FALSE	逻辑常量： 假		*PGAsl* 定义用户变量（DEF） (页 25)
FB	逐段式进给率		*PGsl*

指令	含义	W [1]	说明参见 [2]
FCTDEF	定义多项式函数		*PGAsl* 在线刀具补偿 (PUTFTOCF, FCTDEF, PUTFTOC, FTOCON, FTOCOF) (页 420)
FCUB	按照立方样条改变进给率	m	*PGAsl* 进给曲线 (FNORM, FLIN, FCUB, FPO) (页 474)
FD	用于手轮叠加的轨迹进给率	s	*PGsl*
FDA	用于手轮叠加的轴向进给率	s	*PGsl*
FENDNORM	取消拐角减速	m	*PGAsl* 带有角部减速的进给减速 (FENDNORM, G62, G621) (页 284)
FFWOF [3]	取消前馈控制	m	*PGsl*
FFWON	启用前馈控制	m	*PGsl*
FGREF	回转轴时为参考半径；定向轴时为轨迹参考系数（矢量插补）	m	*PGsl*
FGROUP	确定轴和轨迹进给率		*PGsl*
FI	用于存取框架数据的参数： 精偏		*PGAsl* 读取和修改框架组件 (TR, FI, RT, SC, MI) (页 301)
FIFOCTRL	缓存控制	m	*PGAsl* 带有缓存的程序运行过程 (STOPFIFO, STARTFIFO, FIFOCTRL, STOPRE) (页 479)
FILEDATE	提供最后一次写入文件的日期		*PGAsl* 读取文件信息(FILEDATE, FILETIME, FILESIZE, FILESTAT, FILEINFO) (页 151)

指令	含义	W [1]	说明参见 [2]
FILEINFO	提供 FILEDATE、FILESIZE、FILESTAT 和 FILETIME 的总和信息		*PGAsl* 读取文件信息(FILEDATE, FILETIME, FILESIZE, FILESTAT, FILEINFO) (页 151)
FILESIZE	提供当前文件大小		*PGAsl* 读取文件信息(FILEDATE, FILETIME, FILESIZE, FILESTAT, FILEINFO) (页 151)
FILESTAT	提供的文件状态，如读取、写入、执行、显示、删除（rwxsd）的权限		*PGAsl* 读取文件信息(FILEDATE, FILETIME, FILESIZE, FILESTAT, FILEINFO) (页 151)
FILETIME	提供最后一次写入文件的时间		*PGAsl* 读取文件信息(FILEDATE, FILETIME, FILESIZE, FILESTAT, FILEINFO) (页 151)
FINEA	在到达“精准停”时运行结束	m	*PGAsl* 可编程的运动结束条件 (FINEA, COARSEA, IPOENDA, IPOBRKA, ADISPOSA) (页 285)
FL	同步轴的极限速度	m	*PGsl*
FLIN	线性可变进给率	m	*PGAsl* 进给曲线 (FNORM, FLIN, FCUB, FPO) (页 474)
FMA	轴向多个进给率	m	*PGsl*
FNORM [3]	标准进给率符合 DIN66025	m	*PGAsl* 进给曲线 (FNORM, FLIN, FCUB, FPO) (页 474)
FOCOF	取消带受限力矩/力的运行	m	*PGAsl* 运行到固定挡块（FXS, FXST, FXSW, FOCON, FOCOF） (页 634)
FOCON	启用带受限力矩/力的运行	m	*PGAsl* 运行到固定挡块（FXS, FXST, FXSW, FOCON, FOCOF） (页 634)

指令	含义	W [1]	说明参见 [2]
FOR	带固定运行次数的计数循环		*PGAsl* 计数循环（FOR ... TO ...，ENDFOR） (页 108)
FP	固定点： 将运行到的固定点的编号	s	*PGsl*
FPO	通过一个多项式编程的进给曲线		*PGAsl* 进给曲线 (FNORM, FLIN, FCUB, FPO) (页 474)
FPR	回转轴标记		*PGsl*
FPRAOF	取消旋转进给率		*PGsl*
FPRAON	激活旋转进给率		*PGsl*
FRAME	用于确定坐标系的数据类型		*PGAsl* 定义新框架 (DEF FRAME) (页 304)
FRC	用于倒角和倒圆的进给率	s	*PGsl*
FRCM	用于倒角和倒圆的模态进给率	m	*PGsl*
FROM	一旦满足条件，并且同步动作激活，则执行动作		*PGAsl* 条件循环检查（WHEN, WHENEVER, FROM, EVERY） (页 571)
FTOC	修改刀具精补		*PGAsl* 联机刀具补偿 (FTOC) (页 610)
FTOCOF [3]	取消在线刀具精补	m	*PGAsl* 在线刀具补偿 (PUTFTOCF, FCTDEF, PUTFTOC, FTOCON, FTOCOF) (页 420)
FTOCON	激活在线刀具精补	m	*PGAsl* 在线刀具补偿 (PUTFTOCF, FCTDEF, PUTFTOC, FTOCON, FTOCOF) (页 420)

指令	含义	W [1]	说明参见 [2]
FXS	激活“运动到固定点停止”	m	*PGsl*
FXST	“运动到固定点停止”的力矩极限	m	*PGsl*
FXSW	“运动到固定点停止”的监控窗口		*PGsl*
FZ	每齿进给量	m	*PGsl*

指令	含义	W [1]	说明参见 [2]
G0	线性插补，带快速移动（快进运行）	m	*PGsl*
G1 [3]	线性插补，带进给（直线插补）	m	*PGsl*
G2	顺时针圆弧插补	m	*PGsl*
G3	逆时针圆弧插补	m	*PGsl*
G4	暂停时间，给定时间	s	*PGsl*
G5	斜向切入式磨削	s	*PGAsl* 斜置轴 (TRAANG) (页 385)
G7	斜向切入式磨削时的补偿运动	s	*PGAsl* 斜置轴 (TRAANG) (页 385)
G9	准停 - 速度减少	s	*PGsl*
G17 [3]	选择工作平面 X/Y	m	*PGsl*
G18	选择工作平面 Z/X	m	*PGsl*

指令	含义	W [1]	说明参见 [2]
G19	选择工作平面 Y/Z	m	*PGsl*
G25	工作范围下限	s	*PGsl*
G26	工作范围上限	s	*PGsl*
G33	螺纹切削，等螺距	m	*PGsl*
G34	螺纹切削,增螺距	m	*PGsl*
G35	螺纹切削,减螺距	m	*PGsl*
G40 [3]	取消刀具半径补偿	m	*PGsl*
G41	刀具半径补偿，轮廓左边	m	*PGsl*
G42	刀具半径补偿，轮廓右边	m	*PGsl*
G53	抑制当前零点偏移（逐段）	s	*PGsl*
G54	第 1 个可设定的零点偏移	m	*PGsl*
G55	2. 可设定的零点偏移	m	*PGsl*
G56	3. 可设定的零点偏移	m	*PGsl*
G57	4. 可设定的零点偏移	m	*PGsl*
G58	可编程的轴向绝对零点粗偏	s	*PGsl*

指令	含义	W [1]	说明参见 [2]
G59	可编程的轴向增量零点精偏	s	*PGsl*
G60 [3]	准停 - 速度减少	m	*PGsl*
G62	激活刀具半径补偿（G41、G42）时，内角上的减速度	m	*PGAsl* 带有角部减速的进给减速 (FENDNORM, G62, G621) (页 284)
G63	带弹性卡头的攻丝	s	*PGsl*
G64	连续路径运行	m	*PGsl*
G70	英制尺寸，用于几何数据（长度）	m	*PGsl*
G71 [3]	公制尺寸，用于几何数据（长度）	m	*PGsl*
G74	回参考点运行	s	*PGsl*
G75	回固定点运行	s	*PGsl*
G90 [3]	绝对尺寸	m/s	*PGsl*
G91	增量尺寸	m/s	*PGsl*
G93	时间倒数进给率 rpm	m	*PGsl*
G94 [3]	直线进给率 F，单位：毫米/分钟、英寸/分钟、度/分钟	m	*PGsl*
G95	旋转进给率 F，单位毫米/转、英寸/转	m	*PGsl*

指令	含义	W [1]	说明参见 [2]
G96	激活恒定切削速度（同 G95 时）	m	*PGsl*
G97	取消恒定切削速度（同 G95 时）	m	*PGsl*
G110	极点编程，相对于最后编程的给定位置	s	*PGsl*
G111	极点编程，相对于当前工件坐标系的零点	s	*PGsl*
G112	极点编程，相对于最后有效的极点	s	*PGsl*
G140 [3]	由 G41/G42 确定的逼近方向 WAB	m	*PGsl*
G141	逼近方向 WAB，轮廓左边	m	*PGsl*
G142	逼近方向 WAB，轮廓右边	m	*PGsl*
G143	逼近方向 WAB，切线相关	m	*PGsl*
G147	以直线平滑逼近	s	*PGsl*
G148	以直线平滑返回	s	*PGsl*
G153	取消当前框架，包括基准框架	s	*PGsl*
G247	沿四分圆平滑逼近	s	*PGsl*
G248	沿四分圆平滑返回	s	*PGsl*
G290	转换到 SINUMERIK 模式 ON	m	*FBW*
G291	转换到 ISO2/3 模式 ON	m	*FBW*

指令	含义	W [1]	说明参见 [2]
G331	不带弹性卡头的螺纹切削，正向螺距，右旋螺纹	m	*PGsl*
G332	不带弹性卡头的螺纹切削，负向螺距，左旋螺纹	m	*PGsl*
G340 [3]	空间逼近程序段（深度和平面上相等（螺旋线））	m	*PGsl*
G341	首先在垂直轴上进给(z)，然后在平面中运动	m	*PGsl*
G347	以半圆平滑逼近	s	*PGsl*
G348	以半圆平滑返回	s	*PGsl*
G450 [3]	过渡圆弧	m	*PGsl*
G451	等距离交点	m	*PGsl*
G460 [3]	启用轮廓碰撞监控，用于逼近程序段和退回程序段	m	*PGsl*
G461	在 TRC 程序段中插入一个圆弧	m	*PGsl*
G462	在 TRC 程序段中插入一条直线	m	*PGsl*
G500 [3]	取消所有可设定的框架，基本框架激活	m	*PGsl*
G505...G599	5 ... 99. 可设定的零点偏移	m	*PGsl*
G601 [3]	在精准停时切换程序段	m	*PGsl*
G602	在粗准停时切换程序段	m	*PGsl*

指令	含义	W [1]	说明参见 [2]
G603	在 IPO 程序段结束处切换程序段	m	*PGsl*
G621	所有拐角处都减速	m	*PGAsl* 带有角部减速的进给减速 (FENDNORM, G62, G621) (页 284)
G641	连续路径运行，根据位移标准开展平滑（= 可编程的平滑距离）	m	*PGsl*
G642	连续路径运行，按照定义的公差开展平滑	m	*PGsl*
G643	连续路径运行，按照定义的公差开展平滑（程序段内部）	m	*PGsl*
G644	连续路径运行，采用允许的最大动态响应开展平滑	m	*PGsl*
G645	连续路径运行，按照定义的公差对拐角和程序段切线过渡开展平滑	m	*PGsl*
G700	英制尺寸，用于几何数据和工艺数据（长度、进给率）	m	*PGsl*
G710 [3]	公制尺寸，用于几何数据和工艺数据（长度、进给率）	m	*PGsl*
G751	通过中间点返回固定点	s	PGsl
G810 [3], ..., G819	给 OEM 用户保留的 G 代码组		*PGAsl* 适用于OEM用户的专用函数 (OEMIPO1, OEMIPO2, G810 bis G829) (页 283)
G820 [3], ..., G829	给 OEM 用户保留的 G 代码组		*PGAsl* 适用于OEM用户的专用函数 (OEMIPO1, OEMIPO2, G810 bis G829) (页 283)
G931	进给由运行时间给定	m	
G942	取消线性进给、恒定切削速度或者主轴转速	m	

指令	含义	W [1]	说明参见 [2]
G952	取消旋转进给、恒定切削速度或者主轴转速	m	
G961	恒定切削速度和直线进给	m	*PGsl*
G962	线性进给、旋转进给和恒定切削速度	m	*PGsl*
G971	取消主轴转速和直线进给	m	*PGsl*
G972	取消线性进给、旋转进给和恒定主轴转速	m	*PGsl*
G973	无主轴转速限制的旋转进给	m	*PGsl*
GEOAX	给几何轴 1－3 分配新的通道轴		*PGAsl* 可转换的几何轴 (GEOAX) (页 688)
GET	更换通道间已经使能的轴		*PGAsl* 交换轴，交换主轴 (RELEASE, GET, GETD) (页 132)
GETACTT	从具有相同名称的刀具组中获取有效的刀具。		*FBW*
GETACTTD	确定绝对 D 号所属的 T 号		*PGAsl* 任意 D 编号赋值：求得预先给出 D 编号刀具的 T 编号（GETACTTD） (页 447)
GETD	直接更换通道间的轴		*PGAsl* 交换轴，交换主轴 (RELEASE, GET, GETD) (页 132)
GETDNO	提供某个刀具(T)某个刀沿(CE)的 D 号		*PGAsl* 任意 D 编号赋值：重命名 D 编号(GETDNO, SETDNO) (页 446)
GETEXET	读取换入的 T 号		*FBW*
GETFREELOC	为指定的刀具查找刀库中的空位		*FBW*

指令	含义	W [1]	说明参见 [2]
GETSELT	提供一个预选的 T 号		*FBW*
GETT	给刀具名确定 T 号		*FBW*
GETTCOR	读取刀具长度或刀具长度分量		*FB1(W1)*
GETTENV	读取 T 号、D 号和 DL 号		*FB1(W1)*
GOTO	跳转指令首先向前，然后向后（方向首先向程序结束处，然后向程序开始）		*PGAsl* 程序跳转到跳转标记处 (GOTOB, GOTOF, GOTO, GOTOC) (页 92)
GOTOB	跳转指令，向后（程序起始方向）		*PGAsl* 程序跳转到跳转标记处 (GOTOB, GOTOF, GOTO, GOTOC) (页 92)
GOTOC	和 GOTO 一样，报警 14080 “没有找到跳转目标”会被抑制		*PGAsl* 程序跳转到跳转标记处 (GOTOB, GOTOF, GOTO, GOTOC) (页 92)
GOTOF	跳转指令，向前（程序结束方向）		*PGAsl* 程序跳转到跳转标记处 (GOTOB, GOTOF, GOTO, GOTOC) (页 92)
GOTOS	跳回到程序开始		*PGAsl* 跳回到程序开始 (GOTOS) (页 91)
GP	位置属性间接编程的关键字		*PGAsl* 间接编程位置属性（GP） (页 65)
GWPSOF	撤销选择恒定砂轮圆周速度 (GWPS)	s	*PGsl*
GWPSON	选择恒定砂轮圆周速度(GWPS)	s	*PGsl*
H...	输出到 PLC 的辅助功能		*PGsl/FB1(H2)*
HOLES1	钻削图循环，成排孔		*BHDsl/BHFsl*
HOLES2	钻削图循环，圆弧孔		*BHDsl/BHFsl*
I	插补参数	s	*PGsl*

指令	含义	W [1]	说明参见 [2]
I1	中间点坐标	s	*PGsI*
IC	增量尺寸	s	*PGsI*
ICYCOF	根据 ICYCOF，一个工艺循环的所有程序段会在一个插补周期中执行		*PGAsI* 控制工艺循环的处理工作（ICYCOF，ICYCON） (页 645)
ICYCON	根据 ICYCON，一个工艺循环的每个程序段都会在一个单独的插补周期中执行		*PGAsI* 控制工艺循环的处理工作（ICYCOF，ICYCON） (页 645)
ID	表示模态同步动作	m	*PGAsI* 适用范围和加工顺序（ID, IDS） (页 569)
IDS	表示模态静态同步动作		*PGAsI* 适用范围和加工顺序（ID, IDS） (页 569)
IF	在零件程序/工艺循环中引入一个有条件的跳转		*PGAsI* 带选项的程序循环 (IF, ELSE, ENDIF) (页 106)
INDEX	确定输入字符串中一个字符的索引		*PGAsI* 在字符串中查找字符/字符串 (INDEX, RINDEX, MINDEX, MATCH) (页 87)
INIPO	上电时初始化变量		*PGAsI* 定义用户变量（DEF） (页 25)
INIRE	复位时初始化变量		*PGAsI* 定义用户变量（DEF） (页 25)
INICF	重新配置时初始化变量		*PGAsI* 定义用户变量（DEF） (页 25)
INIT	选择某个 NC 程序，然后在某个通道中执行该程序		*PGAsI* 程序协调（INIT，START，WAITM，WAITMC，WAITE，SETM，CLEARM） (页 113)

指令	含义	W [1]	说明参见 [2]
INITIAL	生成所有区域的 INI 文件		*PGAsl* 工作存储器 (CHANDATA, COMPLETE, INITIAL) (页 219)
INT	数据类型： 带正负号的整数值		*PGAsl* 定义用户变量（DEF） (页 25)
INTERSEC	计算两个轮廓单元之间的交点		*PGAsl* 计算两个轮廓元素之间的交点（INTERSEC）。 (页 732)
INVCCW	逆时针方向渐开线运行	m	*PGsl*
INVCW	顺时针方向渐开线运行	m	*PGsl*
INVFRAME	从一个框架计算出逆转框架		*FB1(K2)*
IP	可变的插补参数		*PGAsl* 间接编程 (页 60)
IPOBRKA	运动条件，自制动斜坡开始点	m	*PGAsl* 可编程的运动结束条件 (FINEA, COARSEA, IPOENDA, IPOBRKA, ADISPOSA) (页 285)
IPOENDA	在到达“插补停止”时运行结束	m	*PGAsl* 可编程的运动结束条件 (FINEA, COARSEA, IPOENDA, IPOBRKA, ADISPOSA) (页 285)
IPTRLOCK	不支持搜索功能的程序部分开始，中断指针位于下一个机床功能程序段上。	m	*PGAsl* 阻止SERUPRO的程序位置 (IPTRLOCK, IPTRUNLOCK) (页 488)
IPTRUNLOCK	不支持搜索功能的程序部分结束，中断指针位于中断时处理的当前程序段上。	m	*PGAsl* 阻止SERUPRO的程序位置 (IPTRLOCK, IPTRUNLOCK) (页 488)
ISAXIS	检查被设为参数的几何轴是否为 1		*PGAsl* 轴功能 (AXNAME, AX, SPI, AXTOSPI, ISAXIS, AXSTRING, MODAXVAL) (页 685)

指令	含义	W 1)	说明参见 2)
ISD	插入深度	m	*PGAsl* 激活 3D 刀具补偿 (CUT3DC, CUT3DF, CUT3DFS, CUT3DFF, ISD) (页 425)
ISFILE	检查在 NCK 用户存储器中是否有一个文件		*PGAsl* 检查文件的存在性(ISFILE) (页 148)
ISNUMBER	检查是否可以把输入字符串转换成数字		*PGAsl* 从字符串 (NUMBER, ISNUMBER, AXNAME) 类型转换 (页 83)
ISOCALL	间接调用某个以 ISO 语言编程的程序		*PGAsl* 间接调用某个以ISO语言编程的程序 (ISOCALL) (页 198)
ISVAR	检查传送参数是否包含一个 NC 知晓的变量		*PGAsl* 读取功能调用 ISVAR 和机床数据队列索引 (页 705)
J	插补参数	s	*PGsl*
J1	中间点坐标	s	*PGsl*
JERKA	激活借助机床数据设定的、编程轴的加速度属性		
JERKLIM	最大轴向急动的递增或递减	m	*PGAsl* 百分比式急冲修正 (JERKLIM) (页 500)
JERKLIMA	最大轴向急动的递增或递减	m	*PGsl*
K	插补参数	s	*PGsl*
K1	中间点坐标	s	*PGsl*
KONT	在刀具补偿时绕行轮廓	m	*PGsl*

指令	含义	W [1]	说明参见 [2]
KONTC	以曲率恒定的多项式逼近/后退	m	*PGsl*
KONTT	以切线恒定的多项式逼近/后退	m	*PGsl*
L	子程序号	s	*PGAsl* 没有参数传递的子程序调用 (页 188)
LEAD	导角 1. 刀具定向 2. 定向多项式	m	*PGAsl* 编程刀具定向 (A..., B..., C..., LEAD, TILT) (页 338)
LEADOF	取消引导值耦合		*PGAsl* 轴向引导值偶合 (LEADON, LEADOF) (页 536)
LEADON	激活引导值耦合		*PGAsl* 轴向引导值偶合 (LEADON, LEADOF) (页 536)
LENTOAX	提供生效刀具 L1、L2 和 L3 长度赋值给纵坐标、横坐标和垂直坐标的信息。		*FB1(W1)*
LFOF [3]	取消“螺纹切削时快速退刀”	m	*PGsl*
LFON	激活“螺纹切削时快速退刀”	m	*PGsl*
LFPOS	使由 POLFMASK 或 POLFMLIN 指明的轴退回到由 POLF 编写的绝对位置上	m	*PGsl*
LFTXT	快速退刀时的退回平面由轨迹切线和当前的刀具方向确定	m	*PGsl*
LFWP	快速退刀时的退回平面由当前的加工平面确定(G17/G18/G19)	m	*PGsl*
LIFTFAST	快速退刀		*PGsl* 快速离开工件轮廓（SETINT LIFTFAST，ALF） (页 125)

指令	含义	W [1]	说明参见 [2]
LIMS	转速限制 用于 G96/G961 和 G97	m	*PGsl*
LLI	变量的下限值		*PGAsl* 属性： 极限值(LLI, ULI) (页 38)
LN	自然对数		*PGAsl* 运算功能 (页 69)
LOCK	使用 ID 锁止同步动作 (停止工艺循环)		*PGAsl* 禁用，释放，复位 (LOCK, UNLOCK, RESET) (页 648)
LONGHOLE	铣削图循环，圆弧排列型长孔		*BHDsl/BHFsl*
LOOP	引入一个无限循环		*PGAsl* 无限程序循环（LOOP，ENDLOOP） (页 107)

指令	含义	W [1]	说明参见 [2]
M0	程序停止		*PGsl*
M1	可选停止		*PGsl*
M2	主程序程序结束，复位到程序开始		*PGsl*
M3	主轴顺时针旋转		*PGsl*
M4	主轴逆时针旋转		*PGsl*
M5	主轴停止		*PGsl*
M6	换刀		*PGsl*
M17	子程序结束		*PGsl*

指令	含义	W [1]	说明参见 [2]
M19	主轴运动到 SD43240 指定的位置		*PGs/*
M30	程序结束，同 M2		*PGs/*
M40	自动齿轮换档		*PGs/*
M41 ... M45	齿轮级 1 ... 5		*PGs/*
M70	过渡到进给轴运行		*PGs/*
MASLDEF	定义主/从轴连接		*PGAs/* 主/从组合 (MASLDEF, MASLDEL, MASLON, MASLOF, MASLOFS) (页 563)
MASLDEL	分离主/从轴连接，删除连接定义		*PGAs/* 主/从组合 (MASLDEF, MASLDEL, MASLON, MASLOF, MASLOFS) (页 563)
MASLOF	关闭一个临时耦合		*PGAs/* 主/从组合 (MASLDEF, MASLDEL, MASLON, MASLOF, MASLOFS) (页 563)
MASLOFS	取消临时的耦合，自动停止从动轴		*PGAs/* 主/从组合 (MASLDEF, MASLDEL, MASLON, MASLOF, MASLOFS) (页 563)
MASLON	激活一个临时耦合		*PGAs/* 主/从组合 (MASLDEF, MASLDEL, MASLON, MASLOF, MASLOFS) (页 563)
MATCH	在字符串中查找一个字符串		*PGAs/* 在字符串中查找字符/字符串 (INDEX, RINDEX, MINDEX, MATCH) (页 87)
MAXVAL	两个变量中的较大值（算术 函数）		*PGAs/* 参见“最大变量、最小变量和变量区域(MINVAL, MAXVAL, BOUND)” (页 77)

指令	含义	W 1)	说明参见 2)
MCALL	模态子程序调用		*PGAsl* 模态子程序调用 (MCALL) (页 194)
MEAC	连续测量，不带剩余行程删除	s	*PGAsl* 扩展测量函数 (MEASA, MEAWA, MEAC) (选项) (页 273)
MEAFRAME	从测量点中计算框架		*PGAsl* 从空间中的三个测量点计算框架 (MEAFRAME) (页 311)
MEAS	用触发探头进行测量	s	*PGAsl* 用接触式探头测量 (MEAS, MEAW) (页 270)
MEASA	测量，带剩余行程删除	s	*PGAsl* 扩展测量函数 (MEASA, MEAWA, MEAC) (选项) (页 273)
MEASURE	工件和刀具测量的计算方法		*FB2(M5)* 用接触式探头测量 (MEAS, MEAW) (页 270)
MEAW	用触发探头进行测量，不删除剩余行程	s	*PGAsl* 用接触式探头测量 (MEAS, MEAW) (页 270)
MEAWA	测量，不带剩余行程删除	s	*PGAsl* 扩展测量函数 (MEASA, MEAWA, MEAC) (选项) (页 273)
MI	存取框架数据： 镜像		*PGAsl* 读取和修改框架组件 (TR, FI, RT, SC, MI) (页 301)
MINDEX	确定输入字符串中一个字符的索引		*PGAsl* 在字符串中查找字符/字符串 (INDEX, RINDEX, MINDEX, MATCH) (页 87)
MINVAL	两个变量中的较小值（算数 函数）		*PGAsl* 参见“最大变量、最小变量和变量区域(MINVAL, MAXVAL, BOUND)” (页 77)
MIRROR	可编程镜像	s	*PGAsl*

指令	含义	W [1]	说明参见 [2]
MMC	在 HMI 上从零件程序中调用互动式对话窗口		*PGAsl* 交互式调用零件程序 (MMC) 窗口 (页 710)
MOD	取模除法		*PGAsl* 运算功能 (页 69)
MODAXVAL	得到模数回转轴的取模位置		*PGAsl* 轴功能 (AXNAME, AX, SPI, AXTOSPI, ISAXIS, AXSTRING, MODAXVAL) (页 685)
MOV	启动定位轴		*PGAsl* 起动/停止轴 (MOV) (页 617)
MSG	可编程的信息	m	*PGsl*
MVTOOL	用于移动刀具的语言指令		*FBW*
N	NC 分程序段号		*PGsl*
NCK	规定数据有效区。		*PGAsl* 定义用户变量（DEF） (页 25)
NEWCONF	采用已经修改的机床数据，相当于激活机床数据		*PGAsl* 有效设置机床数据（NEWCONF） (页 139)
NEWT	创建新的刀具		*PGAsl* 读取和修改框架组件 (TR, FI, RT, SC, MI) (页 301)
NORM [3]	在刀具补偿时，在起始点和终点处的标准设置	m	*PGsl*
NOT	逻辑“非”		*PGAsl* 比较运算和逻辑运算 (页 72)
NPROT	机床专用的保护区“激活/取消”		*PGAsl* 激活/取消激活保护区 (CPROT, NPROT) (页 227)

指令	含义	W 1)	说明参见 2)
NPROTDEF	定义机床专用的保护区		*PGAsl* 保护区的确定 (CPROTDEF, NPROTDEF) (页 223)
NUMBER	转换输入字符串为数字		*PGAsl* 从字符串 (NUMBER, ISNUMBER, AXNAME) 类型转换 (页 83)
OEMIPO1	OEM 插补 1	m	*PGAsl* 适用于OEM用户的专用函数 (OEMIPO1, OEMIPO2, G810 bis G829) (页 283)
OEMIPO2	OEM 插补 2	m	*PGAsl* 适用于OEM用户的专用函数 (OEMIPO1, OEMIPO2, G810 bis G829) (页 283)
OF	CASE 回路中的关键字		*PGAsl* 程序分支(CASE ... OF ... DEFAULT ...) (页 95)
OFFN	编程轮廓的加工余量	m	*PGsl*
OMA1	OEM 地址 1	m	
OMA2	OEM 地址 2	m	
OMA3	OEM 地址 3	m	
OMA4	OEM 地址 4	m	
OMA5	OEM 地址 5	m	
OR	逻辑运算符，“或”连接		*PGAsl* 比较运算和逻辑运算 (页 72)
ORIAXES	线性插补机床轴或者方向轴	m	*PGAsl* 定位轴编程(ORIAXES, ORIVECT, ORIEULER, ORIRPY, ORIRPY2, ORIVIRT1, ORIVIRT2) (页 348)
ORIAXPOS	虚拟的方向轴与回转轴位置的方向角	m	

指令	含义	W [1]	说明参见 [2]
ORIC [3]	外拐角定向变化叠加在将要插入的圆弧程序段上	m	*PGAsl* 刀具定向(ORIC, ORID, OSOF, OSC, OSS, OSSE, ORIS, OSD, OST) (页 439)
ORICONCCW	逆时针方向圆弧表面插补	m	*PGAsl/FB3(F3)* 沿一个圆锥表面定向编程(ORIPLANE, ORICONCW, ORICONCCW, ORICONTO, ORICONIO) (页 351)
ORICONCW	顺时针方向圆弧表面插补	m	*PGAsl/FB3(F4)* 沿一个圆锥表面定向编程(ORIPLANE, ORICONCW, ORICONCCW, ORICONTO, ORICONIO) (页 351)
ORICONIO	圆弧表面插补，指定了一个中间方向	m	*PGAsl/FB3(F4)* 沿一个圆锥表面定向编程(ORIPLANE, ORICONCW, ORICONCCW, ORICONTO, ORICONIO) (页 351)
ORICONTO	在切向过渡中的某个圆侧面上的插补 (最终定向说明)	m	*PGAsl/FB3(F5)* 沿一个圆锥表面定向编程(ORIPLANE, ORICONCW, ORICONCCW, ORICONTO, ORICONIO) (页 351)
ORICURVE	定向插补，其中指定了刀具的两个接触点的运动	m	*PGAsl/FB3(F6)* 两个接触点的定向预设值(ORICURVE, PO[XH]=, PO[YH]=, PO[ZH]=) (页 354)
ORID	在圆弧程序段之前执行定向变化	m	*PGAsl* 刀具定向(ORIC, ORID, OSOF, OSC, OSS, OSSE, ORIS, OSD, OST) (页 439)
ORIEULER	欧拉角方向角	m	*PGAsl* 定位轴编程(ORIAXES, ORIVECT, ORIEULER, ORIRPY, ORIRPY2, ORIVIRT1, ORIVIRT2) (页 348)
ORIMKS	在机床坐标系中的刀具定向	m	*PGAsl* 定向轴的关系 (ORIWKS, ORIMKS) (页 345)

指令	含义	W [1]	说明参见 [2]
ORIPATH	刀具定向参照于轨迹	m	*PGAsl* 轨迹相关的刀具定向旋转（ORIPATH、ORIPATHS、旋转角） (页 362)
ORIPATHS	刀具定向参照这个轨迹，拐点在方向变化中被平滑	m	*PGAsl* 轨迹相关的刀具定向旋转（ORIPATH、ORIPATHS、旋转角） (页 362)
ORIPLANE	平面插补 （相应于 ORIVECT） 大半径圆插补	m	*PGAsl* 沿一个圆锥表面定向编程(ORIPLANE, ORICONCW, ORICONCCW, ORICONTO, ORICONIO) (页 351)
ORIRESET	刀具定向的基本设置，最多带 3 个定向轴		*PGAsl* 定向编程变量和初始位置（ORIRESET） (页 336)
ORIROTA	规定的绝对旋转方向的旋转角度	m	*PGAsl* 刀具定向旋转(ORIROTA, ORIROTR, ORIROTT, ORIROTC, THETA) (页 358)
ORIROTC	轨迹切线的切向旋转矢量	m	*PGAsl* 刀具定向旋转(ORIROTA, ORIROTR, ORIROTT, ORIROTC, THETA) (页 358)
ORIROTR	相对于平面在起始方向和结束方向之间的旋转角度	m	*PGAsl* 刀具定向旋转(ORIROTA, ORIROTR, ORIROTT, ORIROTC, THETA) (页 358)
ORIROTT	相对于方向矢量改变的旋转角度	m	*PGAsl* 刀具定向旋转(ORIROTA, ORIROTR, ORIROTT, ORIROTC, THETA) (页 358)
ORIRPY	通过 RPY 角的定向角（XYZ）	m	*PGAsl* 定位轴编程(ORIAXES, ORIVECT, ORIEULER, ORIRPY, ORIRPY2, ORIVIRT1, ORIVIRT2) (页 348)
ORIRPY2	通过 RPY 角的定向角（ZYX）	m	*PGAsl* 定位轴编程(ORIAXES, ORIVECT, ORIEULER, ORIRPY, ORIRPY2, ORIVIRT1, ORIVIRT2) (页 348)

指令	含义	W [1]	说明参见 [2]
ORIS	定向改变	m	*PGAs/* 刀具定向(ORIC, ORID, OSOF, OSC, OSS, OSSE, ORIS, OSD, OST) (页 439)
ORISOF [3]	取消定向曲线的平滑	m	*PGAs/* 定向曲线的平滑(ORISON, ORISOF) (页 371)
ORISON	启用定向曲线的平滑	m	*PGAs/* 定向曲线的平滑(ORISON, ORISOF) (页 371)
ORIVECT	大圆插补（和 ORIPLANE 一致）	m	*PGAs/* 定位轴编程(ORIAXES, ORIVECT, ORIEULER, ORIRPY, ORIRPY2, ORIVIRT1, ORIVIRT2) (页 348)
ORIVIRT1	通过虚拟定向轴的定向角（定义 1）	m	*PGAs/* 定位轴编程(ORIAXES, ORIVECT, ORIEULER, ORIRPY, ORIRPY2, ORIVIRT1, ORIVIRT2) (页 348)
ORIVIRT2	通过虚拟定向轴的定向角（定义 1）	m	*PGAs/* 定位轴编程(ORIAXES, ORIVECT, ORIEULER, ORIRPY, ORIRPY2, ORIVIRT1, ORIVIRT2) (页 348)
ORIWKS [3]	在工件坐标系中的刀具定向	m	*PGAs/* 定向轴的关系 (ORIWKS, ORIMKS) (页 345)
OS	激活/取消摆动		*PGAs/* 异步摆动(OS, OSP1, OSP2, OST1, OST2, OSCTRL, OSNSC, OSE, OSB) (页 655)
OSB	摆动： 起始点	m	*FB2(P5)*
OSC	恒定平滑刀具定向	m	*PGAs/* 刀具定向(ORIC, ORID, OSOF, OSC, OSS, OSSE, ORIS, OSD, OST) (页 439)
OSCILL	Axis: 1-3 进给轴	m	*PGAs/* 由同步动作控制的摆动(OSCILL) (页 661)

指令	含义	W [1]	说明参见 [2]
OSCTRL	选件摆动	m	*PGAsl* 异步摆动(OS, OSP1, OSP2, OST1, OST2, OSCTRL, OSNSC, OSE, OSB) (页 655)
OSD	通过设定数据指定平滑长度来平滑刀具定向	m	*PGAsl* 刀具定向(ORIC, ORID, OSOF, OSC, OSS, OSSE, ORIS, OSD, OST) (页 439)
OSE	摆动结束位置	m	*PGAsl* 异步摆动(OS, OSP1, OSP2, OST1, OST2, OSCTRL, OSNSC, OSE, OSB) (页 655)
OSNSC	摆动： 无火花磨削次数	m	*PGAsl* 异步摆动(OS, OSP1, OSP2, OST1, OST2, OSCTRL, OSNSC, OSE, OSB) (页 655)
OSOF [3]	取消刀具定向平滑	m	*PGAsl* 异步摆动(OS, OSP1, OSP2, OST1, OST2, OSCTRL, OSNSC, OSE, OSB) (页 655)
OSP1	摆动： 左侧换向点	m	*PGAsl* 异步摆动(OS, OSP1, OSP2, OST1, OST2, OSCTRL, OSNSC, OSE, OSB) (页 655)
OSP2	右侧换向点的摆动	m	*PGAsl* 异步摆动(OS, OSP1, OSP2, OST1, OST2, OSCTRL, OSNSC, OSE, OSB) (页 655)
OSS	在程序段结束处平滑刀具方向	m	*PGAsl* 刀具定向(ORIC, ORID, OSOF, OSC, OSS, OSSE, ORIS, OSD, OST) (页 439)
OSSE	程序段开始和结束的刀具平滑定向	m	*PGAsl* 刀具定向(ORIC, ORID, OSOF, OSC, OSS, OSSE, ORIS, OSD, OST) (页 439)
OST	通过设定数据“相比编制的定向曲线的最大公差”来指定角度公差（度），平滑刀具定向	m	*PGAsl* 刀具定向(ORIC, ORID, OSOF, OSC, OSS, OSSE, ORIS, OSD, OST) (页 439)

指令	含义	W [1]	说明参见 [2]
OST1	摆动： 在右换向点停止	m	*PGAs/* 异步摆动(OS, OSP1, OSP2, OST1, OST2, OSCTRL, OSNSC, OSE, OSB) (页 655)
OST2	摆动： 在右换向点停止	m	*PGAs/* 异步摆动(OS, OSP1, OSP2, OST1, OST2, OSCTRL, OSNSC, OSE, OSB) (页 655)
OTOL	定向公差，用于压缩器功能、定向平滑和精磨方式		*PGAs/* 可编程的轮廓公差/定向公差(CTOL, OTOL, ATOL) (页 505)
OVR	转速补偿	m	*PGAs/*
OVRA	轴的转速补偿	m	*PGAs/*
OVRRAP	快进补偿	m	*PGAs/*
P	零件程序运行次数		*PGAs/* 程序重复次数(P) (页 193)
PAROT	工件坐标系和工件对准	m	*PGs/*
PAROTOF	取消和工件相关的框架旋转	m	*PGs/*
PCALL	调用子程序，带绝对的路径参数和参数传递		*PGAs/* 调用带有路径说明和参数的子程序 (PCALL) (页 199)
PDELAYOF	取消冲压延迟	m	*PGAs/* 激活或取消冲压和步冲(SPOF, SON, PON, SONS, PONS, PDELAYON, PDELAYOF, PUNCHACC) (页 671)
PDELAYON [3]	激活冲压延迟	m	*PGAs/* 激活或取消冲压和步冲(SPOF, SON, PON, SONS, PONS, PDELAYON, PDELAYOF, PUNCHACC) (页 671)

指令	含义	W [1]	说明参见 [2]
PHU	变量的物理单位		*PGAsl* 定义用户变量（DEF） (页 25)
PL	1. B 样条： 节点间距 2. 多项式插补： 多项式插补中参数间隔的长度	s	*PGAsl* 1. 样条插补 (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (页 243) 2. 多项式插补 (POLY, POLYPATH) (页 260)
PM	每分钟		*PGsl*
PO	多项式插补的多项式系数	s	*PGAsl* 多项式插补 (POLY, POLYPATH) (页 260)
POCKET3	铣削循环，矩形腔（任意铣刀）		*BHDsl/BHFsl*
POCKET4	铣削循环，圆形腔（任意铣刀）		*BHDsl/BHFsl*
POLF	返回位置 LIFTFAST	m	*PGsl/PGAsl*
POLFA	用 $AA_ESR_TRIGGER 启动单个轴的退回位置	m	*PGsl*
POLFMASK	激活轴间无关联的退回运动	m	*PGsl*
POLFMLIN	激活轴间有线性关联的退回运动	m	*PGsl*
POLY	多项式插补	m	*PGAsl* 多项式插补 (POLY, POLYPATH) (页 260)
POLYPATH	多项式插补可选择，用于轴组 AXIS 或者 VECT	m	*PGAsl* 多项式插补 (POLY, POLYPATH) (页 260)
PON	激活冲压	m	*PGAsl* 激活或取消冲压和步冲(SPOF, SON, PON, SONS, PONS, PDELAYON, PDELAYOF, PUNCHACC) (页 671)

指令	含义	W [1]	说明参见 [2]
PONS	在插补周期中激活冲压	m	*PGAsl* 激活或取消冲压和步冲(SPOF, SON, PON, SONS, PONS, PDELAYON, PDELAYOF, PUNCHACC) (页 671)
POS	轴定位		*PGsl*
POSA	轴定位，超出程序段界限		*PGsl*
POSM	刀库定位		*FBW*
POSP	轴分段定位（摆动）		*PGsl*
POSRANGE	确定，某个轴当前插补的给定位置是否在规定的参考位置的窗口中。		*PGAsl* 规定的参考区域中的位置（POSRANGE） (页 616)
POT	平方 （算术函数）		*PGAsl* 运算功能 (页 69)
PR	每转		*PGsl*
PREPRO	表示经过预处理的子程序		*PGAsl* 标记子程序“准备”(PREPRO) (页 179)
PRESETON	实际值设定，用于编程的轴		*PGAsl* 预设定位移 (PRESETON) (页 309)
PRIO	在处理中断时设置优先级的关键字		*PGAsl* 中断程序赋值和启动(SETINT, PRIO, BLSYNC) (页 121)
PROC	一个程序的第一个指令		*PGAsl* 调用带有路径说明和参数的子程序 (PCALL) (页 199)
PTP	点对点运行	m	*PGAsl* 直角坐标 PTP运动 (页 390)

指令	含义	W 1)	说明参见 2)
PTPG0	在 G0 时为点对点运动，其余为 CP	m	*PGAsl* PTP 当 TRANSMIT 时 (页 395)
PUNCHACC	步冲时的位移控制式加速度		*PGAsl* 激活或取消冲压和步冲(SPOF, SON, PON, SONS, PONS, PDELAYON, PDELAYOF, PUNCHACC) (页 671)
PUTFTOC	用于并行修整的刀具精补		*PGAsl* 在线刀具补偿 (PUTFTOCF, FCTDEF, PUTFTOC, FTOCON, FTOCOF) (页 420)
PUTFTOCF	根据 FCTDEF 定义的功能、用于并行修整的刀具精补		*PGAsl* 在线刀具补偿 (PUTFTOCF, FCTDEF, PUTFTOC, FTOCON, FTOCOF) (页 420)
PW	B 样条，点加权	s	*PGAsl* 样条插补 (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (页 243)
QECLRNOF	取消象限误差补偿学习		*PGAsl* 学习补偿特性曲线 (QECLRNON, QECLRNOF) (页 708)
QECLRNON	激活象限误差补偿学习		*PGAsl* 学习补偿特性曲线 (QECLRNON, QECLRNOF) (页 708)
QU	快速 辅助功能输出		*PGsl*
R...	计算参数也作为可设定的地址标识符，并带有数字扩展		*PGAsl* 预定义用户变量： 计算参数（R） (页 21)
RAC	绝对、逐段、轴专用的半径编程	s	*PGsl*
RDISABLE	读入禁止		*PGAsl* 设定读入禁止 (RDISABLE) (页 597)

指令	含义	W [1]	说明参见 [2]
READ	在所说明的文件中读入一个或者多个行，并且在数组中存放所读入的信息。		*PGAsl* 读取文件中的行（READ） (页 145)
REAL	数据类型： 带有正负号的浮点变量（实数）		*PGAsl* 定义用户变量（DEF） (页 25)
REDEF	机床数据、NC 语言单元和系统变量的设定，即在哪个用户组中显示		*PGAsl* 系统变量，用户变量和 NC 语言指令的重新定义（REDEF） (页 32)
RELEASE	使能机床轴，用于轴交换		*PGAsl* 交换轴，交换主轴 (RELEASE, GET, GETD) (页 132)
REP	关键字，用同一个值初始化一个数组的所有元素		*PGAsl* 定义和初始化数组变量（DEF，SET，REP） (页 51)
REPEAT	重复一个程序循环		*PGAsl* 程序部分重复 (REPEAT, REPEATB, ENDLABEL, P) (页 98)
REPEATB	重复一个程序行		*PGAsl* 程序部分重复 (REPEAT, REPEATB, ENDLABEL, P) (页 98)
REPOSA	所有轴再次逼近轮廓	s	*PGAsl* 返回轮廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (页 491)
REPOSH	以半圆再次逼近轮廓	s	*PGAsl* 返回轮廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (页 491)
REPOSHA	所有轴再次逼近轮廓，几何轴以半圆逼近	s	*PGAsl* 返回轮廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (页 491)

指令	含义	W [1]	说明参见 [2]
REPOSL	沿直线再次逼近轮廓	s	*PGAsl* 返回轮廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (页 491)
REPOSQ	沿四分圆再次逼近轮廓	s	*PGAsl* 返回轮廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (页 491)
REPOSQA	所有轴再次沿直线逼近轮廓，几何轴沿四分圆逼近	s	*PGAsl* 返回轮廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (页 491)
RESET	复位工艺循环		*PGAsl* 禁用，释放，复位 (LOCK, UNLOCK, RESET) (页 648)
RESETMON	用于激活设定值的语言指令		*FBW*
RET	子程序结束		*PGAsl* 可设定的子程序返回 (RET ...) (页 182)
RIC	相对、逐段、轴专用的半径编程	s	*PGsl* 交换轴，交换主轴 (RELEASE, GET, GETD) (页 132)
RINDEX	确定输入字符串中一个字符的索引		*PGAsl* 在字符串中查找字符/字符串 (INDEX, RINDEX, MINDEX, MATCH) (页 87)
RMB	再次逼近程序段开始的位置	m	*PGAsl* 返回轮廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (页 491)
RME	再次逼近程序段结束的位置	m	*PGAsl* 返回轮廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (页 491)

指令	含义	W [1]	说明参见 [2]
RMI [3]	再次逼近中断点	m	*PGAsl* 返回轮廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (页 491)
RMN	再次逼近最近的路径点	m	*PGAsl* 返回轮廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (页 491)
RND	轮廓角倒圆	s	*PGsl*
RNDM	模态倒圆	m	*PGsl*
ROT	可编程旋转	s	*PGsl*
ROTS	可编程的框架旋转，带立体角	s	*PGsl*
ROUND	小数位四舍五入		*PGAsl* 运算功能 (页 69)
ROUNDUP	向上取整输入值		*PGAsl* 取整 (ROUNDUP) (页 156)
RP	极半径	m/s	*PGsl*
RPL	平面中旋转	s	*PGsl*
RT	用于存取框架数据的参数： 旋转		*PGAsl* 读取和修改框架组件 (TR, FI, RT, SC, MI) (页 301)
RTLIOF	G0，不带直线插补（单轴插补）	m	*PGsl*
RTLION	带直线插补的 G0	m	*PGsl*

指令	含义	W 1)	说明参见 2)
S	主轴转速或 （G4，G96/G961 中含义不同）	m/s	*PGsl*
SAVE	在子程序调用时保护信息		*PGAsl* 保存模态 G 功能（SAVE） (页 168)
SBLOF	抑制单程序段		*PGAsl* 抑制单程序段处理 (SBLOF, SBLON) (页 170)
SBLON	取消单程序段抑制		*PGAsl* 抑制单程序段处理 (SBLOF, SBLON) (页 170)
SC	用于存取框架数据的参数： 比例		*PGAsl* 读取和修改框架组件 (TR, FI, RT, SC, MI) (页 301)
SCALE	可编程缩放	s	*PGsl*
SCC	选择端面轴进行 G96/G961/G962 设置 轴名称可以为几何轴、通道轴或者加工轴。		*PGsl*
SCPARA	编程伺服参数段		*PGAsl* 可编程的伺服参数程序段 (SCPARA) (页 289)
SD	样条度数	s	*PGAsl* 样条插补 (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (页 243)
SEFORM	工步编辑器中的结构化指令，用于生成 HMI 高级版上的工步图		*PGAsl* 步进编辑器中的结构化指令（SEFORM） (页 222)
SET	关键字，用列表值初始化一个数组的所有元素		*PGAsl* 定义和初始化数组变量（DEF，SET，REP） (页 51)
SETAL	设置报警		*PGAsl* 报警（SETAL） (页 717)

指令	含义	W [1]	说明参见 [2]
SETDNO	指定某个刀具(T)某个刀沿(CE)的 D 号		*PGAsl* 任意 D 编号赋值：重命名 D 编号(GETDNO, SETDNO) (页 446)
SETINT	确定在出现一个 NCK 输入时应该激活哪一个中断程序		*PGAsl* 中断程序赋值和启动(SETINT, PRIO, BLSYNC) (页 121)
SETM	设置自有通道中的标记位		PGAsl 程序协调（INIT，START，WAITM，WAITMC，WAITE，SETM，CLEARM）(页 113)
SETMS	机床数据中的主主轴复位		
SETMS (n)	主轴 n 应该作为主主轴		*PGsl*
SETMTH	设置主刀架号		*FBW*
SETPIECE	考虑所有刀具的数量，它们将分配到主轴		*FBW*
SETTA	激活磨损组中的刀具		*FBW*
SETTCOR	考虑到所有标准条件，修改刀具分量		*FB1(W1)*
SETTIA	取消磨损组中的刀具		*FBW*
SF	用于螺纹切削的起始点偏移	m	*PGsl*
SIN	正弦（三角 函数）		*PGAsl* 运算功能 (页 69)
SIRELAY	激活由 SIRELIN、SIRELOUT 和 SIRELTIM 设定的安全功能		*FBSIsl*
SIRELIN	初始化功能块的输入值		*FBSIsl*
SIRELOUT	初始化功能块的输出值		*FBSIsl*
SIRELTIME	初始化功能块的计时器		*FBSIsl*
SLOT1	铣削图循环，圆弧排列型键槽		*BHDsl/BHFsl*
SLOT2	铣削图循环，圆弧键槽		*BHDsl/BHFsl*

指令	含义	W [1]	说明参见 [2]
SOFT	限制急动的轨迹加速度	m	*PGsl*
SOFTA	激活编程的轴上、限制急动的轴加速度		*PGsl*
SON	激活步冲	m	*PGAsl* 激活或取消冲压和步冲(SPOF, SON, PON, SONS, PONS, PDELAYON, PDELAYOF, PUNCHACC) (页 671)
SONS	在插补周期内激活步冲	m	*PGAsl* 激活或取消冲压和步冲(SPOF, SON, PON, SONS, PONS, PDELAYON, PDELAYOF, PUNCHACC) (页 671)
SPATH [3]	FGROUP 轴的轨迹基准为弧长。	m	*PGAsl* 可设置的轨迹基准 (SPATH, UPATH) (页 267)
SPCOF	主主轴或者主轴(n)从位置控制转换到转速控制	m	*PGsl*
SPCON	主主轴或者主轴从转速控制转换到位置控制	m	*PGAsl*
SPI	把主轴编号转换为一个轴名称		*PGAsl* 轴功能 (AXNAME, AX, SPI, AXTOSPI, ISAXIS, AXSTRING, MODAXVAL) (页 685)
SPIF1 [3]	用于冲压/步冲的高速 NCK 输入/输出端 字节 1	m	*FB2(N4)*
SPIF2	用于冲压/步冲的高速 NCK 输入/输出端 字节 2	m	*FB2(N4)*
SPLINEPATH	确定样条连接		*PGAsl* 样条组合(SPLINEPATH) (页 255)
SPN	每个程序段中分段行程的数量	s	*PGAsl* 自动划分位移 (页 677)

指令	含义	W [1]	说明参见 [2]
SPOF [3]	关闭分段行程， 关闭冲压、步冲	m	*PGAs/* 激活或取消冲压和步冲(SPOF, SON, PON, SONS, PONS, PDELAYON, PDELAYOF, PUNCHACC) (页 671)
SPOS	主轴位置	m	*PGs/*
SPOSA	主轴位置超过程序段界限	m	*PGs/*
SPP	分段行程长度	m	*PGAs/* 自动划分位移 (页 677)
SQRT	平方根 (算术函数) (square root)		*PGAs/* 运算功能 (页 69)
SR	用于同步动作的摆动退回行程	s	*PGs/*
SRA	外部输入上，用于同步动作的轴摆动退回行程	m	*PGs/*
ST	用于同步动作的摆动无火花磨削时间	s	*PGs/*
STA	用于同步动作的、轴向摆动无火花磨削时间	m	*PGs/*
START	从运行的程序中，在几个通道中同时启动所选择的程序		*PGAs/* 程序协调（INIT，START，WAITM，WAITMC，WAITE，SETM，CLEARM）(页 113)
STARTFIFO [3]	执行加工；并同时载满缓存	m	*PGAs/* 带有缓存的程序运行过程 (STOPFIFO, STARTFIFO, FIFOCTRL, STOPRE) (页 479)
STAT	铰接位置	s	*PGAs/* 直角坐标 PTP运动 (页 390)
STOLF	G0 公差系数	m	*PGAs/* G0 运动的公差（STOLF） (页 509)

指令	含义	W 1)	说明参见 2)
STOPFIFO	停止执行，载满缓存，直至识别出 STARTFIFO、缓存已满或者程序结束	m	*PGAsl* 带有缓存的程序运行过程 (STOPFIFO, STARTFIFO, FIFOCTRL, STOPRE) (页 479)
STOPRE	预处理停止，直到所有预处理的程序段完成主运行		*PGAsl* 带有缓存的程序运行过程 (STOPFIFO, STARTFIFO, FIFOCTRL, STOPRE) (页 479)
STOPREOF	取消预处理停止		*PGAsl* 取消进给停止 (STOPREOF) (页 599)
STRING	数据类型： 字符串		*PGAsl* 定义用户变量（DEF） (页 25)
STRINGFELD	从编制的字符串数组中选择单个字符		*PGAsl* 选择一个单字符 (STRINGVAR, STRINGFELD) (页 89)
STRINGIS	检查现有的 NC 语言范围，检查专用于该命令所属的 NC 循环名称、用户变量、宏和标签名称是否存在、有效、已定义或激活。		*PGAsl* 检查现有的 NC 语言范围（STRINGIS） (页 699)
STRINGVAR	从编制的字符串中选择单个字符		*PGAsl* 选择一个单字符 (STRINGVAR, STRINGFELD) (页 89)
STRLEN	确定一个字符串的长度		*PGAsl* 确定一个字符串的长度 (STRLEN) (页 86)
SUBSTR	确定输入字符串中一个字符的索引		*PGAsl* 部分字符串的选择 (SUBSTR) (页 88)
SUPA	取消当前零点偏移，包括编程的偏移，系统框架，手轮偏移（DRF），外部零点偏移和叠加运动	s	*PGsl*
SVC	刀具切削速度	m	*PGsl*
SYNFCT	计算一个多项式，取决于运动同步动作中的一个条件		*PGAsl* 同步功能（SYNFCT） (页 604)

指令	含义	W 1)	说明参见 2)
SYNR	在执行时同步读取变量		*PGAsl* 定义用户变量（DEF） (页 25)
SYNRW	在执行时同步读取和写入变量		*PGAsl* 定义用户变量（DEF） (页 25)
SYNW	在执行时同步写入变量		*PGAsl* 定义用户变量（DEF） (页 25)
T	调用刀具 （只有通过机床数据才能加以改变；否则需要使用 M6 指令）		*PGsl*
TAN	正切（三角 函数）		*PGAsl* 运算功能 (页 69)
TANG	定义切线跟踪的轴组合		*PGAsl* 切向控制 (TANG, TANGON, TANGOF, TLIFT, TANGDEL) (页 467)
TANGDEL	删除切线跟踪的轴组合定义		*PGAsl* 切向控制 (TANG, TANGON, TANGOF, TLIFT, TANGDEL) (页 467)
TANGOF	取消切线跟踪		*PGAsl* 切向控制 (TANG, TANGON, TANGOF, TLIFT, TANGDEL) (页 467)
TANGON	激活切线跟踪		*PGAsl* 切向控制 (TANG, TANGON, TANGOF, TLIFT, TANGDEL) (页 467)
TCA	和刀具状态无关的刀具选择/刀具切换		*FBW*
TCARR	指定刀架，编号“m”		*PGAsl* 用于可定向刀架的刀具长度补偿(TCARR, TCOABS, TCOFR, TCOFRX, TCOFRY, TCOFRZ) (页 455)
TCI	将刀具从周转箱换入刀库		*FBW*

指令	含义	W [1]	说明参见 [2]
TCOABS[3]	从当前刀具定向中确定刀具长度分量	m	*PGAsl* 用于可定向刀架的刀具长度补偿(TCARR, TCOABS, TCOFR, TCOFRX, TCOFRY, TCOFRZ) (页 455)
TCOFR	从当前框架的方向确定刀具长度分量	m	*PGAsl* 用于可定向刀架的刀具长度补偿(TCARR, TCOABS, TCOFR, TCOFRX, TCOFRY, TCOFRZ) (页 455)
TCOFRX	选择 X 方向的刀具、刀具点，以确定有效框架的刀具定向	m	*PGAsl* 用于可定向刀架的刀具长度补偿(TCARR, TCOABS, TCOFR, TCOFRX, TCOFRY, TCOFRZ) (页 455)
TCOFRY	选择 Y 方向的刀具、刀具点，以确定有效框架的刀具定向	m	*PGAsl* 用于可定向刀架的刀具长度补偿(TCARR, TCOABS, TCOFR, TCOFRX, TCOFRY, TCOFRZ) (页 455)
TCOFRZ	选择 Z 方向的刀具、刀具点，以确定有效框架的刀具定向	m	*PGAsl* 用于可定向刀架的刀具长度补偿(TCARR, TCOABS, TCOFR, TCOFRX, TCOFRY, TCOFRZ) (页 455)
THETA	旋转角度	s	*PGAsl* 刀具定向旋转(ORIROTA, ORIROTR, ORIROTT, ORIROTC, THETA) (页 358)
TILT	侧向角	m	*PGAsl* 编程刀具定向 (A..., B..., C..., LEAD, TILT) (页 338)
TLIFT	在切线控制中，将中间程序段插入到轮廓角上		*PGAsl* 切向控制 (TANG, TANGON, TANGOF, TLIFT, TANGDEL) (页 467)
TMOF	撤销选择刀具监控		*PGAsl* 在零件程序中磨削专用的刀具监控（TMON、TMOF） (页 683)

指令	含义	W [1]	说明参见 [2]
TMON	选择刀具监控		*PGAsl* 在零件程序中磨削专用的刀具监控（TMON、TMOF） (页 683)
TO	表示 FOR 计数循环中的终点值		*PGAsl* 计数循环（FOR ... TO ...，ENDFOR） (页 108)
TOFF	刀具长度分量方向上的刀具长度偏移，它和索引中指定的几何轴同时生效	m	*PGsl*
TOFFL	刀具长度分量 L1、L2 或 L3 方向上的刀具长度偏移	m	*PGsl*
TOFFOF	复位在线刀具长度补偿		*PGAsl* 在线式刀具长度补偿 (TOFFON, TOFFOF) (页 458)
TOFFON	激活在线刀具长度补偿		*PGAsl* 在线式刀具长度补偿 (TOFFON, TOFFOF) (页 458)
TOFFR	刀具半径偏移	m	*PGsl*
TOFRAME	WCS 的 Z 轴通过框架旋转和刀具方向平行	m	*PGsl*
TOFRAMEX	WCS 的 X 轴通过框架旋转和刀具方向平行	m	*PGsl*
TOFRAMEY	WCS 的 Y 轴通过框架旋转和刀具方向平行	m	*PGsl*
TOFRAMEZ	同 TOFRAME	m	*PGsl*
TOLOWER	将一个字符串的字母转换成小写字母		*PGAsl* 大小写字母转换 (TOLOWER, TOUPPER) (页 85)

指令	含义	W [1]	说明参见 [2]
TOOLENV	保存所有当前状态，这些状态对于分析存储器中保存的刀具数据非常重要		*FB1(W1)*
TOROT	WCS 的 Z 轴通过框架旋转和刀具方向平行	m	*PGsl*
TOROTOF	取消刀具方向框架旋转	m	*PGsl*
TOROTX	WCS 的 X 轴通过框架旋转和刀具方向平行	m	*PGsl*
TOROTY	WCS 的 Y 轴通过框架旋转和刀具方向平行	m	*PGsl*
TOROTZ	同 TOROT	m	*PGsl*
TOUPPER	将一个字符串的字母转换成大写字母		*PGAsl* 大小写字母转换 (TOLOWER, TOUPPER) (页 85)
TOWBCS	基本坐标系中的磨损值（BCS）	m	*PGAsl* 激活的加工的坐标系 (TOWSTD/TOWMCS/TOWWCS/TOWBCS/TOWTCS/TOWKCS) (页 415)
TOWKCS	用于运动转换的刀头坐标系中的磨损值（与刀具旋转 MCS 不同）	m	*PGAsl* 激活的加工的坐标系 (TOWSTD/TOWMCS/TOWWCS/TOWBCS/TOWTCS/TOWKCS) (页 415)
TOWMCS	机床坐标系(MCS)中的磨损值	m	*PGAsl* 激活的加工的坐标系 (TOWSTD/TOWMCS/TOWWCS/TOWBCS/TOWTCS/TOWKCS) (页 415)
TOWSTD	刀具长度中偏移的初始设定值	m	*PGAsl* 激活的加工的坐标系 (TOWSTD/TOWMCS/TOWWCS/TOWBCS/TOWTCS/TOWKCS) (页 415)

指令	含义	W [1]	说明参见 [2]
TOWTCS	刀具坐标系中的磨损值（刀架基准点 T 位于刀具夹持装置中）	m	*PGAsl* 激活的加工的坐标系 (TOWSTD/TOWMCS/TOWWCS/TOWBCS/TOWTCS/TOWKCS) (页 415)
TOWWCS	工件坐标系（WCS）中的磨损值	m	*PGAsl* 激活的加工的坐标系 (TOWSTD/TOWMCS/TOWWCS/TOWBCS/TOWTCS/TOWKCS) (页 415)
TR	框架变量的偏移分量		*PGAsl* 读取和修改框架组件 (TR, FI, RT, SC, MI) (页 301)
TRAANG	倾斜轴转换		*PGAsl* 斜置轴 (TRAANG) (页 385)
TRACON	级联转换		*PGAsl* 级联转换 (TRACON, TRAFOOF) (页 401)
TRACYL	圆柱： 表面转换		*PGAsl* 圆柱形外壳转换（TRACYL） (页 376)
TRAFOOF	取消通道中激活的转换		*PGAsl* 级联转换 (TRACON, TRAFOOF) (页 401)
TRAILOF	取消异步耦合运动		*PGAsl* 联动 (TRAILON, TRAILOF) (页 513)
TRAILON	激活异步耦合运动		*PGAsl* 联动 (TRAILON, TRAILOF) (页 513)
TRANS	可编程的偏移	s	*PGsl*
TRANSMIT	极坐标转换（端面加工）		*PGAsl* 铣削车削件(TRANSMIT) (页 373)
TRAORI	4 轴转换、5 轴转换，同类转换		*PGAsl* 三轴、四轴和五轴转换 (TRAORI) (页 335)
TRUE	逻辑常量： 真		*PGAsl* 定义用户变量（DEF） (页 25)

指令	含义	W 1)	说明参见 2)
TRUNC	去除小数点后位数		*PGAsl* 比较错误的精确度修正 (TRUNC) (页 75)
TU	轴交角	s	*PGAsl* 直角坐标 PTP运动 (页 390)
TURN	螺旋线圈数	s	*PGsl*
ULI	变量的上限值		*PGAsl* 属性： 极限值(LLI, ULI) (页 38)
UNLOCK	使能带 ID 的同步动作（继续工艺循环）		*PGAsl* 禁用，释放，复位 (LOCK, UNLOCK, RESET) (页 648)
UNTIL	结束一个 REPEAT 循环的条件		*PGAsl* 在循环开始处带有条件的程序循环（WHILE，ENDWHILE） (页 110)
UPATH	FGROUP 轴的轨迹基准为曲线参数。	m	*PGAsl* 可设置的轨迹基准 (SPATH, UPATH) (页 267)
VAR	关键字： 参数传递方式		*PGAsl* 带参数传递的子程序调用(EXTERN) (页 190)
VELOLIM	降低最大轴速度	m	*PGAsl* 百分比式速度修正 (VELOLIM) (页 501)
VELOLIMA	降低或提高跟随轴的最大轴速度	m	*PGsl*
WAITC	等待，直到主轴对程序段变化条件已经能满足轴/主轴的要求。		*PGAsl* 程序协调（INIT，START，WAITM，WAITMC，WAITE，SETM，CLEARM） (页 113)
WAITE	等待另一个通道上的程序结束。		*PGAsl* 程序协调（INIT，START，WAITM，WAITMC，WAITE，SETM，CLEARM） (页 113)

指令	含义	W [1]	说明参见 [2]
WAITENC	等待轴位置经过同步或补偿		*PGAs/* 等待有效的轴位置（WAITENC） (页 697)
WAITM	等待指定通道中的标记；以准停结束前一个程序段。		*PGAs/* 程序协调（INIT，START，WAITM，WAITMC，WAITE，SETM，CLEARM） (页 113)
WAITMC	等待设定通道中的标记 ;仅当其它通道尚未到达标记时精确停止		*PGAs/* 程序协调（INIT，START，WAITM，WAITMC，WAITE，SETM，CLEARM） (页 113)
WAITP	等待定位轴运动结束		*PGs/*
WAITS	等待到达主轴位置		*PGs/*
WALCS0	取消选择工件坐标系工作区界限	m	*PGs/*
WALCS1	WCS 工作区域限制组 1 生效	m	*PGs/*
WALCS2	WCS 工作区域限制组 2 生效	m	*PGs/*
WALCS3	WCS 工作区域限制组 3 生效	m	*PGs/*
WALCS4	WCS 工作区域限制组 4 生效	m	*PGs/*
WALCS5	WCS 工作区域限制组 5 生效	m	*PGs/*
WALCS6	WCS 工作区域限制组 6 生效	m	*PGs/*
WALCS7	WCS 工作区域限制组 7 生效	m	*PGs/*

指令	含义	W 1)	说明参见 2)
WALCS8	WCS 工作区域限制组 8 生效	m	*PGsl*
WALCS9	WCS 工作区域限制组 9 生效	m	*PGsl*
WALCS10	WCS 工作区域限制组 10 生效	m	*PGsl*
WALIMOF	取消 BCS 工作区域限制	m	*PGsl*
WALIMON 3)	激活 BCS 工作区域限制	m	*PGsl*
WHEN	当条件满足后，循环执行该动作。		*PGAsl* 条件循环检查（WHEN, WHENEVER, FROM, EVERY） (页 571)
WHENEVER	当条件满足后，执行该动作一次。		*PGAsl* 条件循环检查（WHEN, WHENEVER, FROM, EVERY） (页 571)
WHILE	WHILE 程序循环开始		*PGAsl* 在循环开始处带有条件的程序循环（WHILE，ENDWHILE） (页 110)
WRITE	文本写入到文件系统。 在指定文件的结束处插入一个程序段。		*PGAsl* 写入文件（WRITE） (页 140)
WRTPR	延迟加工任务，而不中断连续路径运行		*PGAsl*
X	轴名称	m/s	*PGsl*
XOR	逻辑“异－或”		*PGAsl* 比较运算和逻辑运算 (页 72)

指令	含义	W [1]	说明参见 [2]
Y	轴名称	m/s	*PGsl*
Z	轴名称	m/s	*PGsl*

16.2 指令：在 SINUMERIK 828D 上的可用性

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
:	●	●	●	●	●	●
*	●	●	●	●	●	●
+	●	●	●	●	●	●
-	●	●	●	●	●	●
<	●	●	●	●	●	●
<<	●	●	●	●	●	●
<=	●	●	●	●	●	●
=	●	●	●	●	●	●
>=	●	●	●	●	●	●
/	●	●	●	●	●	●
/0 … … /7	● ○	● ○	● ○	● ○	● ○	● ○
A	●	●	●	●	●	●
A2	●	●	●	●	●	●
A3	●	●	●	●	●	●
A4	●	●	●	●	●	●
A5	●	●	●	●	●	●
ABS	●	●	●	●	●	●
AC	●	●	●	●	●	●
ACC	●	●	●	●	●	●
ACCLIMA	●	●	●	●	●	●
ACN	●	●	●	●	●	●
ACOS	●	●	●	●	●	●
ACP	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
ACTBLOCNO	●	●	●	●	●	●
ADDFRAME	●	●	●	●	●	●
ADIS	●	●	●	●	●	●
ADISPOS	●	●	●	●	●	●
ADISPOSA	●	●	●	●	●	●
ALF	●	●	●	●	●	●
AMIRROR	●	●	●	●	●	●
AND	●	●	●	●	●	●
ANG	●	●	●	●	●	●
AP	●	●	●	●	●	●
APR	●	●	●	●	●	●
APRB	●	●	●	●	●	●
APRP	●	●	●	●	●	●
APW	●	●	●	●	●	●
APWB	●	●	●	●	●	●
APWP	●	●	●	●	●	●
APX	●	●	●	●	●	●
AR	●	●	●	●	●	●
AROT	●	●	●	●	●	●
AROTS	●	●	●	●	●	●
AS	●	●	●	●	●	●
ASCALE	●	●	●	●	●	●
ASIN	●	●	●	●	●	●
ASPLINE	-	○	-	○	-	○
ATAN2	●	●	●	●	●	●
ATOL	-	●	-	●	-	●
ATRANS	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
AX	●	●	●	●	●	●
AXCTSWE	-	-	-	-	-	-
AXCTSWED	-	-	-	-	-	-
AXIS	●	●	●	●	●	●
AXNAME	●	●	●	●	●	●
AXSTRING	●	●	●	●	●	●
AXTOCHAN	●	●	●	●	●	●
AXTOSPI	●	●	●	●	●	●
B	●	●	●	●	●	●
B2	●	●	●	●	●	●
B3	●	●	●	●	●	●
B4	●	●	●	●	●	●
B5	●	●	●	●	●	●
B_AND	●	●	●	●	●	●
B_OR	●	●	●	●	●	●
B_NOT	●	●	●	●	●	●
B_XOR	●	●	●	●	●	●
BAUTO	-	○	-	○	-	○
BLOCK	●	●	●	●	●	●
BLSYNC	●	●	●	●	●	●
BNAT	-	○	-	○	-	○
BOOL	●	●	●	●	●	●
BOUND	●	●	●	●	●	●
BRISK	●	●	●	●	●	●
BRISKA	●	●	●	●	●	●
BSPLINE	-	○	-	○	-	○
BTAN	-	○	-	○	-	○

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
C	●	●	●	●	●	●
C2	●	●	●	●	●	●
C3	●	●	●	●	●	●
C4	●	●	●	●	●	●
C5	●	●	●	●	●	●
CAC	●	●	●	●	●	●
CACN	●	●	●	●	●	●
CACP	●	●	●	●	●	●
CALCDAT	●	●	●	●	●	●
CALCPOSI	●	●	●	●	●	●
CALL	●	●	●	●	●	●
CALLPATH	●	●	●	●	●	●
CANCEL	●	●	●	●	●	●
CASE	●	●	●	●	●	●
CDC	●	●	●	●	●	●
CDOF	●	●	●	●	●	●
CDOF2	●	●	●	●	●	●
CDON	●	●	●	●	●	●
CFC	●	●	●	●	●	●
CFIN	●	●	●	●	●	●
CFINE	●	●	●	●	●	●
CFTCP	●	●	●	●	●	●
CHAN	●	●	●	●	●	●
CHANDATA	●	●	●	●	●	●
CHAR	●	●	●	●	●	●
CHECKSUM	●	●	●	●	●	●
CHF	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
CHKDM	●	●	●	●	●	●
CHKDNO	●	●	●	●	●	●
CHR	●	●	●	●	●	●
CIC	●	●	●	●	●	●
CIP	●	●	●	●	●	●
CLEARM	-	-	-	-	-	-
CLRINT	●	●	●	●	●	●
CMIRROR	●	●	●	●	●	●
COARSEA	●	●	●	●	●	●
COMPCAD	-	○	-	○	-	○
COMPCURV	-	○	-	○	-	○
COMPLETE	●	●	●	●	●	●
COMPOF	-	○	-	○	-	○
COMPON	-	○	-	○	-	○
CONTDCON	●	●	●	●	●	●
CONTPRON	●	●	●	●	●	●
CORROF	●	●	●	●	●	●
COS	●	●	●	●	●	●
COUPDEF	○	-	○	-	○	-
COUPDEL	○	-	○	-	○	-
COUPOF	○	-	○	-	○	-
COUPOFS	○	-	○	-	○	-
COUPON	○	-	○	-	○	-
COUPONC	○	-	○	-	○	-
COUPRES	○	-	○	-	○	-
CP	●	●	●	●	●	●
CPRECOF	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
CPRECON	●	●	●	●	●	●
CPROT	●	●	●	●	●	●
CPROTDEF	●	●	●	●	●	●
CR	●	●	●	●	●	●
CROT	●	●	●	●	●	●
CROTS	●	●	●	●	●	●
CRPL	●	●	●	●	●	●
CSCALE	●	●	●	●	●	●
CSPLINE	-	○	-	○	-	○
CT	●	●	●	●	●	●
CTAB	-	-	-	-	-	-
CTABDEF	-	-	-	-	-	-
CTABDEL	-	-	-	-	-	-
CTABEND	-	-	-	-	-	-
CTABEXISTS	-	-	-	-	-	-
CTABFNO	-	-	-	-	-	-
CTABFPOL	-	-	-	-	-	-
CTABFSEG	-	-	-	-	-	-
CTABID	-	-	-	-	-	-
CTABINV	-	-	-	-	-	-
CTABISLOCK	-	-	-	-	-	-
CTABLOCK	-	-	-	-	-	-
CTABMEMTYP	-	-	-	-	-	-
CTABMPOL	-	-	-	-	-	-
CTABMSEG	-	-	-	-	-	-
CTABNO	-	-	-	-	-	-
CTABNOMEM	-	-	-	-	-	-

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
CTABPERIOD	-	-	-	-	-	-
CTABPOL	-	-	-	-	-	-
CTABPOLID	-	-	-	-	-	-
CTABSEG	-	-	-	-	-	-
CTABSEGID	-	-	-	-	-	-
CTABSEV	-	-	-	-	-	-
CTABSSV	-	-	-	-	-	-
CTABTEP	-	-	-	-	-	-
CTABTEV	-	-	-	-	-	-
CTABTMAX	-	-	-	-	-	-
CTABTMIN	-	-	-	-	-	-
CTABTSP	-	-	-	-	-	-
CTABTSV	-	-	-	-	-	-
CTABUNLOCK	-	-	-	-	-	-
CTOL	-	○	-	○	-	○
CTRANS	●	●	●	●	●	●
CUT2D	●	●	●	●	●	●
CUT2DF	●	●	●	●	●	●
CUT3DC	-	-	-	-	-	-
CUT3DCC	-	-	-	-	-	-
CUT3DCCD	-	-	-	-	-	-
CUT3DF	-	-	-	-	-	-
CUT3DFF	-	-	-	-	-	-
CUT3DFS	-	-	-	-	-	-
CUTCONOF	●	●	●	●	●	●
CUTCONON	●	●	●	●	●	●
CUTMOD	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
CYCLE...	●	●	●	●	●	●
D	●	●	●	●	●	●
D0	●	●	●	●	●	●
DAC	●	●	●	●	●	●
DC	●	●	●	●	●	●
DEF	●	●	●	●	●	●
DEFINE	●	●	●	●	●	●
DEFAULT	●	●	●	●	●	●
DELAYFSTON	●	●	●	●	●	●
DELAYFSTOF	●	●	●	●	●	●
DELDL	●	●	●	●	●	●
DELDTG	●	●	●	●	●	●
DELETE	●	●	●	●	●	●
DELTOOLENV	●	●	●	●	●	●
DIACYCOFA	●	●	●	●	●	●
DIAM90	●	●	●	●	●	●
DIAM90A	●	●	●	●	●	●
DIAMCHAN	●	●	●	●	●	●
DIAMCHANA	●	●	●	●	●	●
DIAMCYCOF	●	●	●	●	●	●
DIAMOF	●	●	●	●	●	●
DIAMOFA	●	●	●	●	●	●
DIAMON	●	●	●	●	●	●
DIAMONA	●	●	●	●	●	●
DIC	●	●	●	●	●	●
DILF	●	●	●	●	●	●
DISABLE	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
DISC	●	●	●	●	●	●
DISCL	●	●	●	●	●	●
DISPLOF	●	●	●	●	●	●
DISPLON	●	●	●	●	●	●
DISPR	●	●	●	●	●	●
DISR	●	●	●	●	●	●
DITE	●	●	●	●	●	●
DITS	●	●	●	●	●	●
DIV	●	●	●	●	●	●
DL	-	-	-	-	-	-
DO	●	●	●	●	●	●
DRFOF	●	●	●	●	●	●
DRIVE	●	●	●	●	●	●
DRIVEA	●	●	●	●	●	●
DYNFINISH	●	●	●	●	●	●
DYNNORM	●	●	●	●	●	●
DYNPOS	●	●	●	●	●	●
DYNROUGH	●	●	●	●	●	●
DYNSEMIFIN	●	●	●	●	●	●
DZERO	●	●	●	●	●	●
EAUTO	-	○	-	○	-	○
EGDEF	-	-	-	-	-	-
EGDEL	-	-	-	-	-	-
EGOFC	-	-	-	-	-	-
EGOFS	-	-	-	-	-	-
EGON	-	-	-	-	-	-
EGONSYN	-	-	-	-	-	-

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
EGONSYNE	-	-	-	-	-	-
ELSE	●	●	●	●	●	●
ENABLE	●	●	●	●	●	●
ENAT	-	○	-	○	-	○
ENDFOR	●	●	●	●	●	●
ENDIF	●	●	●	●	●	●
ENDLABEL	●	●	●	●	●	●
ENDLOOP	●	●	●	●	●	●
ENDPROC	●	●	●	●	●	●
ENDWHILE	●	●	●	●	●	●
ETAN	-	○	-	○	-	○
EVERY	●	●	●	●	●	●
EX	●	●	●	●	●	●
EXECSTRING	●	●	●	●	●	●
EXECTAB	●	●	●	●	●	●
EXECUTE	●	●	●	●	●	●
EXP	●	●	●	●	●	●
EXTCALL	●	●	●	●	●	●
EXTERN	●	●	●	●	●	●
F	●	●	●	●	●	●
FA	●	●	●	●	●	●
FAD	●	●	●	●	●	●
FALSE	●	●	●	●	●	●
FB	●	●	●	●	●	●
FCTDEF	-	-	-	-	-	-
FCUB	●	●	●	●	●	●
FD	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
FDA	●	●	●	●	●	●
FENDNORM	●	●	●	●	●	●
FFWOF	●	●	●	●	●	●
FFWON	●	●	●	●	●	●
FGREF	●	●	●	●	●	●
FGROUP	●	●	●	●	●	●
FI	●	●	●	●	●	●
FIFOCTRL	●	●	●	●	●	●
FILEDATE	●	●	●	●	●	●
FILEINFO	●	●	●	●	●	●
FILESIZE	●	●	●	●	●	●
FILESTAT	●	●	●	●	●	●
FILETIME	●	●	●	●	●	●
FINEA	●	●	●	●	●	●
FL	●	●	●	●	●	●
FLIN	●	●	●	●	●	●
FMA	-	-	-	-	-	-
FNORM	●	●	●	●	●	●
FOCOF	○	-	○	-	○	-
FOCON	○	-	○	-	○	-
FOR	●	●	●	●	●	●
FP	●	●	●	●	●	●
FPO	-	-	-	-	-	-
FPR	●	●	●	●	●	●
FPRAOF	●	●	●	●	●	●
FPRAON	●	●	●	●	●	●
FRAME	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
FRC	●	●	●	●	●	●
FRCM	●	●	●	●	●	●
FROM	●	●	●	●	●	●
FTOC	●	●	●	●	●	●
FTOCOF	●	●	●	●	●	●
FTOCON	●	●	●	●	●	●
FXS	●	●	●	●	●	●
FXST	●	●	●	●	●	●
FXSW	●	●	●	●	●	●
FZ	●	●	●	●	●	●
G0	●	●	●	●	●	●
G1	●	●	●	●	●	●
G2	●	●	●	●	●	●
G3	●	●	●	●	●	●
G4	●	●	●	●	●	●
G5	●	●	●	●	●	●
G7	●	●	●	●	●	●
G9	●	●	●	●	●	●
G17	●	●	●	●	●	●
G18	●	●	●	●	●	●
G19	●	●	●	●	●	●
G25	●	●	●	●	●	●
G26	●	●	●	●	●	●
G33	●	●	●	●	●	●
G34	●	●	●	●	●	●
G35	●	●	●	●	●	●
G40	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
G41	●	●	●	●	●	●
G42	●	●	●	●	●	●
G53	●	●	●	●	●	●
G54	●	●	●	●	●	●
G55	●	●	●	●	●	●
G56	●	●	●	●	●	●
G57	●	●	●	●	●	●
G58	●	●	●	●	●	●
G59	●	●	●	●	●	●
G60	●	●	●	●	●	●
G62	●	●	●	●	●	●
G63	●	●	●	●	●	●
G64	●	●	●	●	●	●
G70	●	●	●	●	●	●
G71	●	●	●	●	●	●
G74	●	●	●	●	●	●
G75	●	●	●	●	●	●
G90	●	●	●	●	●	●
G91	●	●	●	●	●	●
G93	●	●	●	●	●	●
G94	●	●	●	●	●	●
G95	●	●	●	●	●	●
G96	●	●	●	●	●	●
G97	●	●	●	●	●	●
G110	●	●	●	●	●	●
G111	●	●	●	●	●	●
G112	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
G140	●	●	●	●	●	●
G141	●	●	●	●	●	●
G142	●	●	●	●	●	●
G143	●	●	●	●	●	●
G147	●	●	●	●	●	●
G148	●	●	●	●	●	●
G153	●	●	●	●	●	●
G247	●	●	●	●	●	●
G248	●	●	●	●	●	●
G290	●	●	●	●	●	●
G291	●	●	●	●	●	●
G331	●	●	●	●	●	●
G332	●	●	●	●	●	●
G340	●	●	●	●	●	●
G341	●	●	●	●	●	●
G347	●	●	●	●	●	●
G348	●	●	●	●	●	●
G450	●	●	●	●	●	●
G451	●	●	●	●	●	●
G460	●	●	●	●	●	●
G461	●	●	●	●	●	●
G462	●	●	●	●	●	●
G500	●	●	●	●	●	●
G505...G599	●	●	●	●	●	●
G601	●	●	●	●	●	●
G602	●	●	●	●	●	●
G603	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
G621	●	●	●	●	●	●
G641	●	●	●	●	●	●
G642	●	●	●	●	●	●
G643	●	●	●	●	●	●
G644	●	●	●	●	●	●
G645	●	●	●	●	●	●
G700	●	●	●	●	●	●
G710	●	●	●	●	●	●
G751	●	●	●	●	●	●
G810 , ..., G819	●	●	●	●	●	●
G820 , ..., G829	●	●	●	●	●	●
G931	●	●	●	●	●	●
G942	●	●	●	●	●	●
G952	●	●	●	●	●	●
G961	●	●	●	●	●	●
G962	●	●	●	●	●	●
G971	●	●	●	●	●	●
G972	●	●	●	●	●	●
G973	●	●	●	●	●	●
GEOAX	●	●	●	●	●	●
GET	●	●	●	●	●	●
GETACTT	●	●	●	●	●	●
GETACTTD	●	●	●	●	●	●
GETD	●	●	●	●	●	●
GETDNO	●	●	●	●	●	●
GETEXET	●	●	●	●	●	●
GETFREELOC	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
GETSELT	●	●	●	●	●	●
GETT	●	●	●	●	●	●
GETTCOR	●	●	●	●	●	●
GETTENV	●	●	●	●	●	●
GOTO	●	●	●	●	●	●
GOTOB	●	●	●	●	●	●
GOTOC	●	●	●	●	●	●
GOTOF	●	●	●	●	●	●
GOTOS	●	●	●	●	●	●
GP	●	●	●	●	●	●
GWPSOF	●	●	●	●	●	●
GWPSON	●	●	●	●	●	●
H...	●	●	●	●	●	●
HOLES1	●	●	●	●	●	●
HOLES2	●	●	●	●	●	●
I	●	●	●	●	●	●
I1	●	●	●	●	●	●
IC	●	●	●	●	●	●
ICYCOF	●	●	●	●	●	●
ICYCON	●	●	●	●	●	●
ID	●	●	●	●	●	●
IDS	●	●	●	●	●	●
IF	●	●	●	●	●	●
INDEX	●	●	●	●	●	●
INIPO	●	●	●	●	●	●
INIRE	●	●	●	●	●	●
INICF	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
INIT	-	-	-	-	-	-
INITIAL	●	●	●	●	●	●
INT	●	●	●	●	●	●
INTERSEC	●	●	●	●	●	●
INVCCW	-	-	-	-	-	-
INVCW	-	-	-	-	-	-
INVFRAME	●	●	●	●	●	●
IP	●	●	●	●	●	●
IPOBRKA	●	●	●	●	●	●
IPOENDA	●	●	●	●	●	●
IPTRLOCK	●	●	●	●	●	●
IPTRUNLOCK	●	●	●	●	●	●
ISAXIS	●	●	●	●	●	●
ISD	●	●	●	●	●	●
ISFILE	●	●	●	●	●	●
ISNUMBER	●	●	●	●	●	●
ISOCALL	●	●	●	●	●	●
ISVAR	●	●	●	●	●	●
J	●	●	●	●	●	●
J1	●	●	●	●	●	●
JERKA	●	●	●	●	●	●
JERKLIM	●	●	●	●	●	●
JERKLIMA	●	●	●	●	●	●
K	●	●	●	●	●	●
K1	●	●	●	●	●	●
KONT	●	●	●	●	●	●
KONTC	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
KONTT	●	●	●	●	●	●
L	●	●	●	●	●	●
LEAD 刀具定向 定向多项式	 ● -	 ● -	 ● -	 ● -	 ● -	 ● -
LEADOF	-	-	-	-	-	-
LEADON	-	-	-	-	-	-
LENTOAX	●	●	●	●	●	●
LFOF	●	●	●	●	●	●
LFON	●	●	●	●	●	●
LFPOS	●	●	●	●	●	●
LFTXT	●	●	●	●	●	●
LFWP	●	●	●	●	●	●
LIFTFAST	●	●	●	●	●	●
LIMS	●	●	●	●	●	●
LLI	●	●	●	●	●	●
LN	●	●	●	●	●	●
LOCK	●	●	●	●	●	●
LONGHOLE	-	-	-	-	-	-
LOOP	●	●	●	●	●	●
M0	●	●	●	●	●	●
M1	●	●	●	●	●	●
M2	●	●	●	●	●	●
M3	●	●	●	●	●	●
M4	●	●	●	●	●	●
M5	●	●	●	●	●	●
M6	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
M17	●	●	●	●	●	●
M19	●	●	●	●	●	●
M30	●	●	●	●	●	●
M40	●	●	●	●	●	●
M41 ... M45	●	●	●	●	●	●
M70	●	●	●	●	●	●
MASLDEF	●	●	●	●	●	●
MASLDEL	●	●	●	●	●	●
MASLOF	●	●	●	●	●	●
MASLOFS	●	●	●	●	●	●
MASLON	●	●	●	●	●	●
MATCH	●	●	●	●	●	●
MAXVAL	●	●	●	●	●	●
MCALL	●	●	●	●	●	●
MEAC	-	-	-	-	-	-
MEAFRAME	●	●	●	●	●	●
MEAS	●	●	●	●	●	●
MEASA	-	-	-	-	-	-
MEASURE	●	●	●	●	●	●
MEAW	●	●	●	●	●	●
MEAWA	-	-	-	-	-	-
MI	●	●	●	●	●	●
MINDEX	●	●	●	●	●	●
MINVAL	●	●	●	●	●	●
MIRROR	●	●	●	●	●	●
MMC	●	●	●	●	●	●
MOD	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
MODAXVAL	●	●	●	●	●	●
MOV	●	●	●	●	●	●
MSG	●	●	●	●	●	●
MVTOOL	●	●	●	●	●	●
N	●	●	●	●	●	●
NCK	●	●	●	●	●	●
NEWCONF	●	●	●	●	●	●
NEWT	●	●	●	●	●	●
NORM	●	●	●	●	●	●
NOT	●	●	●	●	●	●
NPROT	●	●	●	●	●	●
NPROTDEF	●	●	●	●	●	●
NUMBER	●	●	●	●	●	●
OEMIPO1	●	●	●	●	●	●
OEMIPO2	●	●	●	●	●	●
OF	●	●	●	●	●	●
OFFN	●	●	●	●	●	●
OMA1	●	●	●	●	●	●
OMA2	●	●	●	●	●	●
OMA3	●	●	●	●	●	●
OMA4	●	●	●	●	●	●
OMA5	●	●	●	●	●	●
OR	●	●	●	●	●	●
ORIAXES	●	●	●	●	●	●
ORIAXPOS	●	●	●	●	●	●
ORIC	●	●	●	●	●	●
ORICONCCW	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
ORICONCW	●	●	●	●	●	●
ORICONIO	●	●	●	●	●	●
ORICONTO	●	●	●	●	●	●
ORICURVE	●	●	●	●	●	●
ORID	●	●	●	●	●	●
ORIEULER	●	●	●	●	●	●
ORIMKS	●	●	●	●	●	●
ORIPATH	●	●	●	●	●	●
ORIPATHS	●	●	●	●	●	●
ORIPLANE	●	●	●	●	●	●
ORIRESET	●	●	●	●	●	●
ORIROTA	●	●	●	●	●	●
ORIROTC	●	●	●	●	●	●
ORIROTR	●	●	●	●	●	●
ORIROTT	●	●	●	●	●	●
ORIRPY	●	●	●	●	●	●
ORIRPY2	●	●	●	●	●	●
ORIS	●	●	●	●	●	●
ORISOF	●	●	●	●	●	●
ORISON	●	●	●	●	●	●
ORIVECT	●	●	●	●	●	●
ORIVIRT1	●	●	●	●	●	●
ORIVIRT2	●	●	●	●	●	●
ORIWKS	●	●	●	●	●	●
OS	-	-	-	-	-	-
OSB	-	-	-	-	-	-
OSC	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
OSCILL	-	-	-	-	-	-
OSCTRL	-	-	-	-	-	-
OSD	●	●	●	●	●	●
OSE	-	-	-	-	-	-
OSNSC	-	-	-	-	-	-
OSOF	●	●	●	●	●	●
OSP1	-	-	-	-	-	-
OSP2	-	-	-	-	-	-
OSS	●	●	●	●	●	●
OSSE	●	●	●	●	●	●
OST	●	●	●	●	●	●
OST1	-	-	-	-	-	-
OST2	-	-	-	-	-	-
OTOL	-	●	-	●	-	●
OVR	●	●	●	●	●	●
OVRA	●	●	●	●	●	●
OVRRAP	●	●	●	●	●	●
P	●	●	●	●	●	●
PAROT	●	●	●	●	●	●
PAROTOF	●	●	●	●	●	●
PCALL	●	●	●	●	●	●
PDELAYOF	-	-	-	-	-	-
PDELAYON	-	-	-	-	-	-
PHU	●	●	●	●	●	●
PL	-	○	-	○	-	○
	-	-	-	-	-	-
PM	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
PO	-	-	-	-	-	-
POCKET3	●	●	●	●	●	●
POCKET4	●	●	●	●	●	●
POLF	●	●	●	●	●	●
POLFA	●	●	●	●	●	●
POLFMASK	●	●	●	●	●	●
POLFMLIN	●	●	●	●	●	●
POLY	-	-	-	-	-	-
POLYPATH	-	-	-	-	-	-
PON	-	-	-	-	-	-
PONS	-	-	-	-	-	-
POS	●	●	●	●	●	●
POSA	●	●	●	●	●	●
POSM	●	●	●	●	●	●
POSP	●	●	●	●	●	●
POSRANGE	●	●	●	●	●	●
POT	●	●	●	●	●	●
PR	●	●	●	●	●	●
PREPRO	●	●	●	●	●	●
PRESETON	●	●	●	●	●	●
PRIO	●	●	●	●	●	●
PROC	●	●	●	●	●	●
PTP	●	●	●	●	●	●
PTPG0	●	●	●	●	●	●
PUNCHACC	-	-	-	-	-	-
PUTFTOC	●	●	●	●	●	●
PUTFTOCF	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
PW	-	○	-	○	-	○
QECLRNOF	●	●	●	●	●	●
QECLRNON	●	●	●	●	●	●
QU	●	●	●	●	●	●
R...	●	●	●	●	●	●
RAC	●	●	●	●	●	●
RDISABLE	●	●	●	●	●	●
READ	●	●	●	●	●	●
REAL	●	●	●	●	●	●
REDEF	●	●	●	●	●	●
RELEASE	●	●	●	●	●	●
REP	●	●	●	●	●	●
REPEAT	●	●	●	●	●	●
REPEATB	●	●	●	●	●	●
REPOSA	●	●	●	●	●	●
REPOSH	●	●	●	●	●	●
REPOSHA	●	●	●	●	●	●
REPOSL	●	●	●	●	●	●
REPOSQ	●	●	●	●	●	●
REPOSQA	●	●	●	●	●	●
RESET	●	●	●	●	●	●
RESETMON	●	●	●	●	●	●
RET	●	●	●	●	●	●
RIC	●	●	●	●	●	●
RINDEX	●	●	●	●	●	●
RMB	●	●	●	●	●	●
RME	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
RMI	●	●	●	●	●	●
RMN	●	●	●	●	●	●
RND	●	●	●	●	●	●
RNDM	●	●	●	●	●	●
ROT	●	●	●	●	●	●
ROTS	●	●	●	●	●	●
ROUND	●	●	●	●	●	●
ROUNDUP	●	●	●	●	●	●
RP	●	●	●	●	●	●
RPL	●	●	●	●	●	●
RT	●	●	●	●	●	●
RTLIOF	●	●	●	●	●	●
RTLION	●	●	●	●	●	●
S	●	●	●	●	●	●
SAVE	●	●	●	●	●	●
SBLOF	●	●	●	●	●	●
SBLON	●	●	●	●	●	●
SC	●	●	●	●	●	●
SCALE	●	●	●	●	●	●
SCC	●	●	●	●	●	●
SCPARA	●	●	●	●	●	●
SD	-	○	-	○	-	○
SEFORM	●	●	●	●	●	●
SET	●	●	●	●	●	●
SETAL	●	●	●	●	●	●
SETDNO	●	●	●	●	●	●
SETINT	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
SETM	-	-	-	-	-	-
SETMS	●	●	●	●	●	●
SETMS (n)	●	●	●	●	●	●
SETMTH	●	●	●	●	●	●
SETPIECE	●	●	●	●	●	●
SETTA	●	●	●	●	●	●
SETTCOR	●	●	●	●	●	●
SETTIA	●	●	●	●	●	●
SF	●	●	●	●	●	●
SIN	●	●	●	●	●	●
SIRELAY	-	-	-	-	-	-
SIRELIN	-	-	-	-	-	-
SIRELOUT	-	-	-	-	-	-
SIRELTIME	-	-	-	-	-	-
SLOT1	●	●	●	●	●	●
SLOT2	●	●	●	●	●	●
SOFT	●	●	●	●	●	●
SOFTA	●	●	●	●	●	●
SON	-	-	-	-	-	-
SONS	-	-	-	-	-	-
SPATH	●	●	●	●	●	●
SPCOF	●	●	●	●	●	●
SPCON	●	●	●	●	●	●
SPI	●	●	●	●	●	●
SPIF1	-	-	-	-	-	-
SPIF2	-	-	-	-	-	-
SPLINEPATH	-	○	-	○	-	○

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
SPN	-	-	-	-	-	-
SPOF	-	-	-	-	-	-
SPOS	●	●	●	●	●	●
SPOSA	●	●	●	●	●	●
SPP	-	-	-	-	-	-
SQRT	●	●	●	●	●	●
SR	-	-	-	-	-	-
SRA	-	-	-	-	-	-
ST	-	-	-	-	-	-
STA	-	-	-	-	-	-
START	-	-	-	-	-	-
STARTFIFO	●	●	●	●	●	●
STAT	●	●	●	●	●	●
STOLF	-	-	-	-	-	-
STOPFIFO	●	●	●	●	●	●
STOPRE	●	●	●	●	●	●
STOPREOF	●	●	●	●	●	●
STRING	●	●	●	●	●	●
STRINGFELD	●	●	●	●	●	●
STRINGIS	●	●	●	●	●	●
STRINGVAR	-	-	-	-	-	-
STRLEN	●	●	●	●	●	●
SUBSTR	●	●	●	●	●	●
SUPA	●	●	●	●	●	●
SVC	●	●	●	●	●	●
SYNFCT	●	●	●	●	●	●
SYNR	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
SYNRW	●	●	●	●	●	●
SYNW	●	●	●	●	●	●
T	●	●	●	●	●	●
TAN	●	●	●	●	●	●
TANG	-	-	-	-	-	-
TANGDEL	-	-	-	-	-	-
TANGOF	-	-	-	-	-	-
TANGON	-	-	-	-	-	-
TCA	●	●	●	●	●	●
TCARR	-	●	-	●	-	●
TCI	●	●	●	●	●	●
TCOABS	-	●	-	●	-	●
TCOFR	-	●	-	●	-	●
TCOFRX	-	●	-	●	-	●
TCOFRY	-	●	-	●	-	●
TCOFRZ	-	●	-	●	-	●
THETA	●	●	●	●	●	●
TILT	●	●	●	●	●	●
TLIFT	-	-	-	-	-	-
TMOF	●	●	●	●	●	●
TMON	●	●	●	●	●	●
TO	●	●	●	●	●	●
TOFF	●	●	●	●	●	●
TOFFL	●	●	●	●	●	●
TOFFOF	●	●	●	●	●	●
TOFFON	●	●	●	●	●	●
TOFFR	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
TOFRAME	●	●	●	●	●	●
TOFRAMEX	●	●	●	●	●	●
TOFRAMEY	●	●	●	●	●	●
TOFRAMEZ	●	●	●	●	●	●
TOLOWER	●	●	●	●	●	●
TOOLENV	●	●	●	●	●	●
TOROT	●	●	●	●	●	●
TOROTOF	●	●	●	●	●	●
TOROTX	●	●	●	●	●	●
TOROTY	●	●	●	●	●	●
TOROTZ	●	●	●	●	●	●
TOUPPER	●	●	●	●	●	●
TOWBCS	-	●	-	●	-	●
TOWKCS	-	●	-	●	-	●
TOWMCS	-	●	-	●	-	●
TOWSTD	-	●	-	●	-	●
TOWTCS	-	●	-	●	-	●
TOWWCS	-	●	-	●	-	●
TR	●	●	●	●	●	●
TRAANG	-	-	-	-	○	-
TRACON	-	-	-	-	○	-
TRACYL	○	○	○	○	○	○
TRAFOOF	●	●	●	●	●	●
TRAILOF	●	●	●	●	●	●
TRAILON	●	●	●	●	●	●
TRANS	●	●	●	●	●	●
TRANSMIT	○	○	○	○	○	○

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
TRAORI	-	●	-	●	-	●
TRUE	●	●	●	●	●	●
TRUNC	●	●	●	●	●	●
TU	●	●	●	●	●	●
TURN	●	●	●	●	●	●
ULI	●	●	●	●	●	●
UNLOCK	●	●	●	●	●	●
UNTIL	●	●	●	●	●	●
UPATH	●	●	●	●	●	●
VAR	●	●	●	●	●	●
VELOLIM	●	●	●	●	●	●
VELOLIMA	●	●	●	●	●	●
WAITC	-	-	-	-	○	-
WAITE	-	-	-	-	-	-
WAITENC	-	-	-	-	-	-
WAITM	-	-	-	-	-	-
WAITMC	-	-	-	-	-	-
WAITP	●	●	●	●	●	●
WAITS	●	●	●	●	●	●
WALCS0	●	●	●	●	●	●
WALCS1	●	●	●	●	●	●
WALCS2	●	●	●	●	●	●
WALCS3	●	●	●	●	●	●
WALCS4	●	●	●	●	●	●
WALCS5	●	●	●	●	●	●
WALCS6	●	●	●	●	●	●
WALCS7	●	●	●	●	●	●

指令	828D 控制系统类型					
	PPU240.2 / 241.2		PPU260.2 / 261.2		PPU280.2 / 281.2	
	basic T	basic M	车削	铣削	车削	铣削
WALCS8	●	●	●	●	●	●
WALCS9	●	●	●	●	●	●
WALCS10	●	●	●	●	●	●
WALIMOF	●	●	●	●	●	●
WALIMON	●	●	●	●	●	●
WHEN	●	●	●	●	●	●
WHENEVER	●	●	●	●	●	●
WHILE	●	●	●	●	●	●
WRITE	●	●	●	●	●	●
WRTPR	●	●	●	●	●	●
X	●	●	●	●	●	●
XOR	●	●	●	●	●	●
Y	●	●	●	●	●	●
Z	●	●	●	●	●	●

- ● 标准
- ○ 可选
- - 不可用

附录 A

A.1 缩略符列表

A	输出
AS	自动化系统
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
AV	工作计划
AWL	语句表
BA	运行方式
BAG	工作方式组
BB	运行就绪
BuB, B&B	操作与观测
BCD	Binary Coded Decimals： 用二进制代码编码的十进制数
BHG	手动操作装置
BIN	二进制文件 (**Bin**ary Files)
BIOS	基本输入输出系统
BCS	基本坐标系
BOF	操作界面
BOT	Boot Files: SIMODRIVE 611D 的引导文件
BT	操作面板
BTSS	操作面板接口
CAD	计算机辅助设计
CAM	计算机辅助制造
CNC	Computerized Numerical Control： 计算机数字控制
COM	通讯
CP	通讯处理器

CPU	Central Processing Unit： 中央处理器
CR	回车键
CRT	Cathode Ray Tube： 阴极射线管
CSB	Central Service Board: PLC 模块
CTS	Clear To Send： 串行接口发送就绪状态
CUTCOM	Cutter radius compensation: 刀具半径补偿
DAU	数模转换器
DB	PLC 中的数据模块
DBB	PLC 中的数据模块字节
DBW	PLC 中的数据模块字
DBX	PLC 中的数据模块位
DC	Direct Control: 在一转内回转轴以最短距离移动到绝对位置
DCD	载波检测
DDE	动态数据交换
DEE	数据结束调试
DIN	德国工业标准
DIO	Data Input/Output: 数据传送显示
DIR	Directory： 目录
DLL	动态连接程序库
DOE	数据传输设备
DOS	磁盘操作系统
DPM	双端口存储器
DPR	双端口存储器
DRAM	动态随机存取存储器
DRF	Differential Resolver Function： 直接测量功能（手轮）
DRY	Dry Run： 空运行进给率
DSB	Decoding Single Block： 解码的程序段
DW	数据字
E	输入
I/O	输入/输出

E/R	SIMODRIVE 611D digital 的 馈电/回馈(电源)模块
EIA 代码	专用穿孔代码，每个符号的孔数总是奇数
ENC	Encoder: 实际值编码器
EPROM	Erasable Programmable Read Only Memory：可擦除可编程只读存储器
ERROR	打印机错误
FB	功能块
FBS	超薄显示屏
FC	Function Call: PLC 中的功能块
FDB	产品数据库
FDD	软盘驱动器
FEPROM	Flash-EPROM: 可读可写存储器
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
FIPO	精插补器
FM	功能模块
FPU	Floating Point Unit： 浮点单位
FRA	FRAME 块
FRAME	数据段（FRAME）
FRK	铣削半径补偿
FST	Feed Stop： 进给停止
FBD	功能块图（PLC 编程方法）
GP	主程序
GUD	Global User Data： 全局用户数据
HD	Hard Disk: 硬盘
HEX	十六进制数代号
HiFu	辅助功能
HMI	人机界面： 用于操作，编程和模拟的 SINUMERIK 操作功能。
HMS	高分辨率测量系统
HSA	主轴驱动
HW	硬件

IBN	调试
IF	驱动模块脉冲使能
IK (GD)	隐含通讯（全局数据）
IKA	Interpolative Compensation: 可插补补偿
IM	Interface-Modul： 接口模块
IMR	Interface-Modul Receive: 接收方接口模块
IMS	Interface-Modul Send: 发送方接口模块
INC	Increment： 增量
INI	Initializing Data: 初始化数据
IPO	插补器
ISA	国际标准体系
ISO	国际标准组织
ISO 代码	专用穿孔代码，每个符号的孔数总是偶数
JOG	Jogging: 调试模式
K1 .. K4	通道 1 至通道 4
K-Bus	通讯总线
KD	坐标旋转
LAD	梯形图（PLC 编程方法）
K_v	闭环增益系数
K_v	传动比
LCD	Liquid-Crystal Display: 液晶显示器
LED	Light Emitting Diode： 发光二极管
LF	进线电源
LMS	位置测量系统
LR	位置调节器
LUD	本地用户数据
MB	兆字节
MD	机床数据
MDA	Manual Data Automatic： 手动数据输入，自动执行
MK	测量回路

MCS	机床坐标系
MLFB	产品订货号
MPF	Main Program File： NC 零件程序（主程序）
MPI	Multi Port Interface: 多端口接口
MS-	微软（软件制造商）
MSTT	机床控制面板
NC	Numerical Control： 数字控制装置
NCK	Numerical Control Kernel： 带有程序段处理，运行范围等等的数字内核
NCU	Numerical Control Unit： NCK 硬件单元
NRK	NCK 操作系统名称
NST	接口信号
NURBS	非一致性数理 B 样条
NV	零点偏移
OB	PLC 中组织块
OEM	原装设备制造商
OP	Operation Panel： 操作面板
OPI	Operation Panel Interface： 操作面板接口
OPT	Options: 选项
OSI	开放式互联系统： 计算机通讯标准
P-Bus	外设总线
PC	个人计算机
PCIN	与控制系统进行数据更换的软件名称
PCMCIA	个人计算机内存卡国际协会： 存储器插卡标准
PCU	PC Unit: PC 主机（计算机元件）
PG	编程器
PLC	Programmable Logic Control： 可编程逻辑控制器
POS	定位
RAM	Random Access Memory: 可读可写程序存储器
REF	回参考点
REPOS	再定位功能

RISC	Reduced Instruction Set Computer： 精简指令集计算机：带有小命令集和快速命令处理的处理器类型
ROV	Rapid Override： 快速倍率
RPA	R-Parameter Active： NCK 中的存储范围 用于 R 参数编号的 R- NCK
RPY	旋转定位移动： 一种坐标系旋转方式
RTS	Request To Send： 开启发送方，控制信号自串行数据接口
SBL	Single Block： 单程序段
SD	设定数据
SDB	系统数据块
SEA	Setting Data Active： 设定数据符号（文件类型）
SFB	系统功能块
SFC	系统功能调用
SK	软键
SKP	Skip： 跳过程序段
SM	步进电机
SPF	Sub Program File： 子程序
SPS	可编程存储器控制
SRAM	静态存储器（缓存）
SRK	刀尖半径补偿
SSFK	主轴丝杆螺距误差补偿
SSI	Serial Synchron Interface： 串行同步接口
SW	软件
SYF	System Files： 系统文件
TEA	Testing Data Active： 机床数据标志
TO	Tool Offset： 刀具补偿
TOA	Tool Offset Active： 刀具补偿符号（文件类型）
TRANSMIT	Transform Milling into Turning： 在车床上用于铣削的坐标转换
UFR	用户框架： 零点偏移
UP	子程序
VSA	进给驱动

RS232	串行接口（DEE 和 DÜE 之间的数据交换线定义）
WCS	工件坐标系
WKZ	刀具
WLK	刀具长度补偿
WOP	现场编程
WPD	Work Piece Directory： 工件目录
WRK	刀具半径补偿
WZK	刀具补偿
WZW	换刀
ZOA	Zero Offset Active： 零点偏移数据符号（文件类型）
µC	微型控制器

A.2 资料反馈

本资料将在质量及易用性上持续改进。 您的建议和意见可以帮助我们完善，请发邮件或传真至：

电子邮件： mailto:docu.motioncontrol@siemens.com

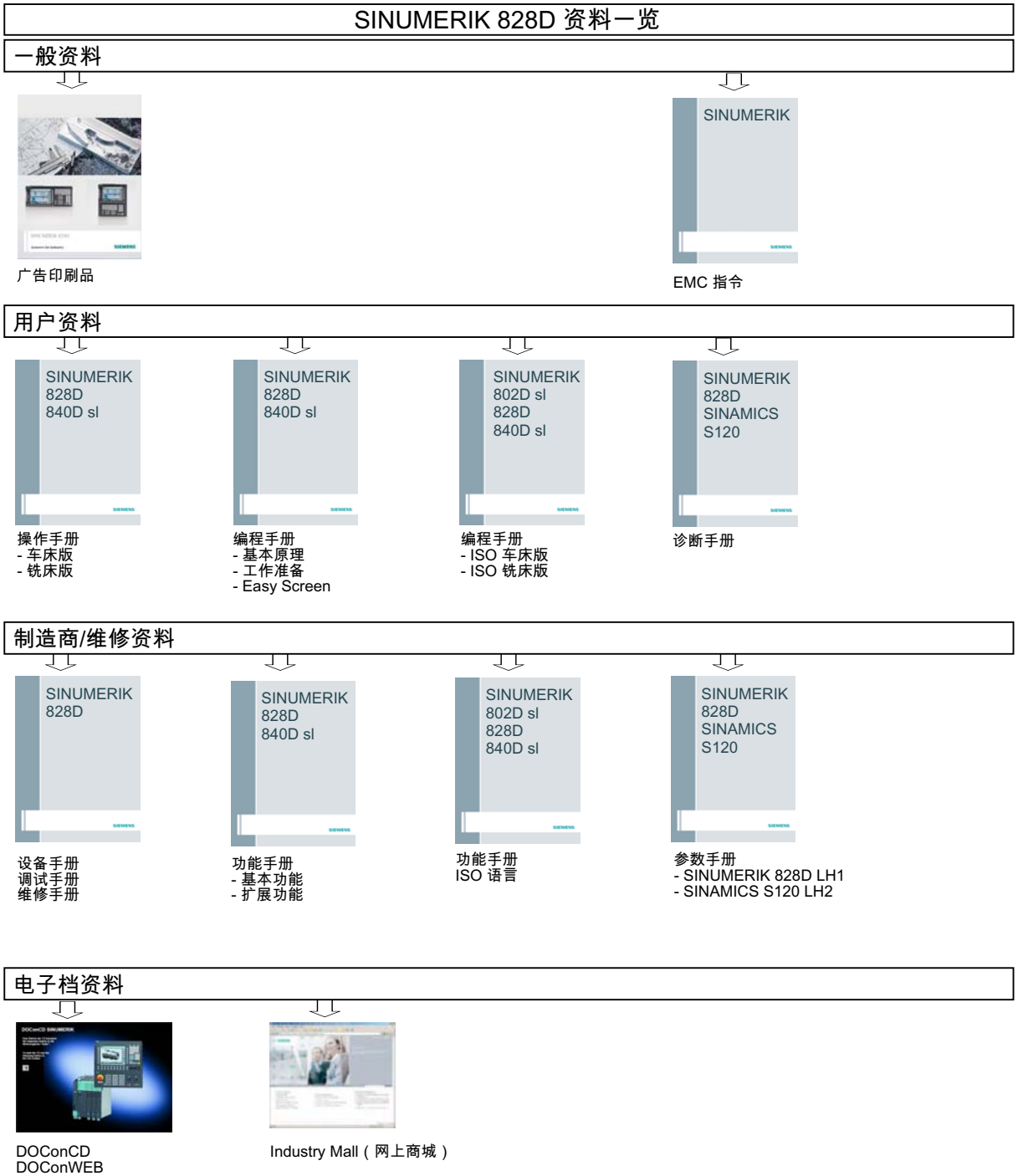
传真： +49 9131 - 98 2176
请使用手册末页的传真样表。

寄 SIEMENS AG I DT MC MS1 Postfach 3180（邮箱 3180） D-91050 Erlangen（爱尔兰根） 传真： +49 9131 - 98 2176 (文献资料)	寄信人 姓名： 公司/部门通信地址 街道： 邮编： 城镇： 电话： / 传真： /

建议及/或更正

A.3 资料概览

A.3.1 828D 的资料分类图



词汇表

C 轴

围绕 C 轴产生一个受控的旋转运动，并用工件主轴定位。

CNC

参见 → NC

CNC 标准语言

标准语言提供： → 用户定义变量、→ 系统变量、→ 宏技术。

COM

NC 控制系统部件，用于执行和和协调通讯。

CPU

中央处理单元，参见 → 可编程存储器控制器

C 样条

C 样条最出名，是一种最常用的样条。 支点处以切线过渡，弯曲平缓。 使用 3 级多项式。

DRF

Differential Resolver Function: NC 功能，在自动方式下利用电子手轮产生增量式零点偏移。

Frame

框架定义一种运算规范，它把一种直角坐标系转换到另一种直角坐标系。 框架中包含几个部分 → 零点偏移、 → 旋转、 → 缩放、 → 镜像。

HIGHSTEP

编程方法汇编，用于。 → 系统 AS300/AS400 中的 PLC。

JOG

控制系统的一种运行方式（调试运行）： 在 Jog 运行方式下，机床可以进行调试。 各个进给轴和主轴可以通过方向键点动运行。 在 JOG 手动运行方式中还有其它的一些功能，如 → 回参考点运行，→ 重新定位以及预设定（设定实际值）。

Kp

传动比

KV

回路放大系数，调节回路中可调节的物理量。

MDA

控制系统的一种运行方式： 手动输入，自动运行 在 MDA 方式下，可以输入单个程序段或者几个程序段，它们与主程序或者子程序无关，使用 NC 启动键可以立即执行。

NC

Numerical Control: NC 控制装置包括所有机床控制装置的组件： → NCK, → PLC, HMI, → COM.

说明

对于控制系统 SINUMERIK 840D，CNC 控制系统应改为： Computerized Numerical Control。

NCK

Numerical Control Kernel: NC 控制系统部件，执行 → 零件程序，并控制机床的运动过程。

NRK

数字机器人内核（NCK →的运行系统）

NURBS

控制系统内部的动作执行和轨迹插补是基于 NURBS（**N**on **U**niform **R**ational **B**-**S**plines 非均匀有理 B 样条）进行的。 这样，在 SINUMERIK 840D 系统内部可为所有插补提供统一的方法。

OEM

SINUMERIK 840D 给机床制造商提供各种不同应用的使用空间（OEM 应用），制造商可以自己设计操作界面或者在系统中开发专用的应用功能。

PLC

可**编程逻辑控制**： → 可编程存储器控制。 -> NC 控制系统组件： 用于执行机床控制逻辑的转接控制。

PLC 编程

PLC 用软件 **STEP 7** 编程。 编程软件 STEP7 基于 **WINDOWS** 标准操作系统并包含创新技术继续开发的 STEP5 编程功能。

PLC-编程存储器

SINUMERIK 840D: 在 PLC 用户存储器中，PLC 用户程序和用户数据与 PLC 主程序一起存储。

R 参数

计算参数，可以由 → 零件程序编程人员在程序中进行任意设定或者询问。

TOA 单元

每个→ TOA 范围可以包含多个 TOA 单元。 TOA 单元的数量以最大有效 →通道数量为极限。 一个 TOA 单元包括一个刀具数据模块和一个刀库数据模块。 另外，也可包含一个刀架数据模块（选件）。

TOA 范围

TOA 范围包含所有刀具和刀库数据。 正常情况下,数据有效范围即→ 通道范围。 但通过机床数据可以确定，多个通道分割一个→ TOA 单元，以使这些通道也可使用通用的刀具管理数据。

WinSCP

WinSCP 是一个自由使用的开放式源程序，用于 Windows 的文件传输。

丝杠螺距误差补偿

滚珠丝杠在进给时产生机械误差，由控制系统通过存储的误差测量值进行补偿。

中断程序

中断程序是专门的 → 子程序，它们可以通过加工过程中的外部事件（外部信号）启动。加工过程中零件程序的程序段被中断，进给轴的中断位置被自动存储。

中间程序段

带→ 刀具补偿（G41/G42）的运行可以由一定数量的中间程序段（在补偿级的程序段，没有轴运动）中断，这样刀具补偿还可以进行正确地计算。 先于控制系统读出所允许的中间程序段数量，可以通过系统参数设定。

串行接口 V.24

PCU 20 上有一个串行 V.24 接口(RS232)用于数据输入/输出；PCU 50/70 上则存在两个 V.24 接口。 通过该接口可以装载和保护加工程序以及制造商和用户数据。

主程序

用序号或者名称标志的 → 零件程序，在主程序中可以调用其它的主程序、子程序或者 → 循环。

主程序段

通过“：”引导的程序段，包含在 → 零件程序中启动操作顺序所需要的所有数据。

保护区

在 → 加工区之内的一个三维空间，刀尖不可以进入此区域。

信息

零件程序中可编程的所有信息，以及系统可识别的 → 报警均在操作面板上显示，带日期和时间，并有相应的清除标准符号。 报警和信息单独显示。

倍率

可以手动或者编程进行工作，允许操作人员覆盖编程的进给或者转速，使加工速度与具体的工件和材料相适应。

倒圆轴

倒圆轴指工件或者刀具旋转到一个分度头给定的角度位置。 到达分度头刻度后，倒圆轴“到达位置”。

公制测量系统

单位均为公制： 用于长度，例如 mm（毫米），m（米）。

关键字

有确定写法的字，它们在 → 零件程序编程语言中具有所定义的含义。

准停

使用编程的准停指令，可以准确地、有时必须较慢地回到程序段中所设定的位置。 为了减少准停时的逼近时间，对于快速移动和进给需定义 → 准停极限。

准停界限

如果所有的轨迹轴均到达准停界限，则控制系统会认为已经精确到达目标。 进行 → 零件程序的程序段转换。

几何尺寸

→ 工件在 → 工件坐标系中的描述。

几何轴

几何轴用于描述工件坐标系中 2 维或者 3 维区域。

刀具

机床中进行加工的有效零件（诸如车刀、铣刀、钻头、激光...）

刀具半径补偿

为了可以直接编程一个所需的 → 工件轮廓，控制系统必须考虑所使用刀具的半径，与编程的轮廓等距离轨迹运行。（G41/G42）。

刀具补偿

计算轨迹时，考虑到刀具尺寸。

刀沿半径补偿

在编程一个轮廓时，往往从刀具的尖端计算。 但是，这在实际加工过程中并不可以实现，因为所使用的刀具会有一个弯曲半径，控制系统必须要考虑这个值。 在此计算的加工点就位于其中心点，距离为半径的长度。

加工区

用加工区定义一个三维空间，在此空间内刀尖可以移动。 参见 → 保护区

加工轴

在机床中表示实际存在的轴。

加工通道

通过通道结构可以进行并行处理，缩短辅助时间，比如在装载的同时可以进行加工。 在此，一个 CNC 通道可以看作为一个独力的 CNC 控制系统，可以译码、程序段预处理并进行插补。

加速度，带过冲限制

为了在机床上获得优化的加速性能，同时又要保护机械部分，在加工程序中可以在跃变式加速度和平缓式加速度之间进行转换。

参考点

机床中的点，→ 加工轴的测量系统以该点为基准。

反比时间进给

在 SINUMERIK 840D 中，可以编程一个程序段的轨迹行程所需要的时间（G93），而不用编程轴的进给速度。

变量定义

定义变量时，包括确定数据类型和变量名。 使用该变量名，也就是调用该变量值。

可插补补偿

借助于补偿插补，加工操作中的 主轴上升故障和测量系统故障可以得到补偿（SSFK，MSFK）。

可编程存储器控制系统

可编程存储器控制系统（SPS）是电子控制系统，它们的功能以程序的形式存储到控制器中。 因此，控制器的结构和布线与控制系统的功能无关。 可编程存储器控制系统具有计算机的结构，它由带存储器的 CPU（中央模块）、输入/输出模块和内部总线系统构成。外设和编程语言以控制技术为准。

可编程的工作区域限制

将刀具的运动空间限制到一个通过编程的限制定义的空间范围。

可编程的框架

使用编程的 → 框架可以在零件程序加工过程中，动态地定义新的坐标系原点。 根据当前的原点，利用一个新框架和附加的确定值，与绝对的确定值加以区分。

同步

→ 零件程序中的指令，用于协调同一加工地点时不同→ 通道中的加工过程。

同步动作

1. 辅助功能输出

   在工件加工期间，可以把工艺功能（→ 辅助功能）从 CNC 程序中输出到 PLC 中。 通过这些辅助功能可以控制机床的附加设备，比如顶尖套筒、夹持器、卡盘等等。

2. 快速辅助功能输出

   对于时间较紧的开关功能，可以减少 → 辅助功能的应答时间，避免加工过程不必要的停顿。

同步轴

同步轴为 → 龙门架轴，其额定位置始终由运行的 → 引导轴引导并与之同步运行。 从操作员和编程者的视角观察，同步轴是“不存在的”。

同步轴

同步轴运行时间与几何轴相同。

名称

根据 DIN 66025 标准，字需要补充变量名（计算变量，系统变量和用户变量）、子程序名、关键字名和带多个地址字母的字。 这些补充的字在意义上与构成程序段的字一样。 名称必须意义明确。 同一个名称不可以用于不同的对象。

回转轴

回转轴指工件或者刀具旋转到一个给定的角度位置。

圆弧插补

在轮廓上两个固定点之间，→ 刀具以给定的进给量按圆弧运行，从而加工出工件。

地址

地址是一个确定的运算数或者运算范围的标志，比如输入、输出等等。

坐标系

参见 → 机床坐标系, → 工件坐标系

基准坐标系

是一个直角坐标系，它通过转换到机床坐标系而形成。

在 → 零件程序中使用准坐标系编程的轴名称。 如果没有 → 有效的坐标系转换，则它平行于 → 机床坐标系。 不同点在于 → 轴名称。

基准轴

计算补偿值时必须考虑该轴的给定值或者实际值，这个轴就称为基准轴。

增量尺寸

也称为相对尺寸： 表示一个进给轴待运行的行程和方向，以已经到达的点为基准。 参见 → 绝对尺寸

增量尺寸

通过相对尺寸说明加工行程。 增量尺寸可以作为 → 设定数据存储，或者通过相应的增量键 10、100、1000 和 10000 进行选取。

备份电池

利用备份电池保证在电网掉电时，→ 用户程序可以安全地存放在 → CPU 中，并且确定的数据区以及剩余的标志位、定时器和计数器可以保持。

外设模块

用外设模块建立 CPU 和过程之间的联系。

外设模块是：

- → 数字量输入/输出模块
- → 模拟量输入/输出模块
- → 模拟器模块

外部零点偏移

由 → PLC 给定的零点偏移。

多项式插补

用多项式插补功能可以产生不同的曲线，如**线性函数、抛物线函数和幂函数**（SINUMERIK 840D）。

子程序

一个→ 零件程序的连续指令，它们可以通过设定不同的参数反复调用。 子程序从主程序中调用。 针对没有授权的读取和显示，子程序会被锁定。 → 循环是子程序的一种形式。

存档

读出文件和/或目录，存储到 **外部** 存储器设备中。

安全功能

系统所具有的始终处于激活的监控功能，可以及早识别出 → CNC 中，匹配控制系统 (→ PLC) 中和机床中的故障，从而排除一切对工件、刀具或者机床可能造成的损害。 在故障发生时，加工过程会中断，驱动停止，故障原因被存储并作为报警显示。 同时通知 PLC 数控系统有一报警。

宏指令技术

一个指令名称下汇编一串指令。 在程序中，该指令名就代表这一串汇编的指令。

定位轴

在机床中执行辅助运动的轴。（例如刀库，托盘运输）。 定位轴不与 → 轨迹轴进行插补。

定向主轴准停

比如工件主轴在一给定角度位置停止，从而可以在某一确定的位置进行其它的加工工作。

定向刀具退回

RETTOOL: 当加工过程被停止时（比如刀具折断），刀具可以根据编程指令按照事先给定的方向后撤一段距离。

尺寸系统：公制和英制

在加工程序中，位置值和螺距值可以用英制编程。 控制器设定一个基准系统，它与编程的尺寸说明(G70/G71)无关。

工件

需由机床制造/加工的零件。

工件坐标系

工件坐标系的原点为→ 工件零点。 在工件坐标系中编程时，尺寸和方向以工件坐标系为基准。

工件轮廓

待建立/加工→ 工件的给定轮廓。

工件零点

工件零点构成了→ 工件坐标系的原点。 它由与→ 机床零点的距离定义。

工作区极限

除行程开关之外，还可以使用工作区域限制功能对进给轴的行程范围进行限制。 对于每个进给轴，可以使用两个数值对保护加工区进行设定。

工作存储器

工作存储器是一个 RAM 存储器，程序加工期间在 → CPU 中可以对用户程序进行存取。

异步子程序

指可以通过一个中断信号（比如信号“快速 NC 输入”）启动的、与当前程序状态异步（无关）的零件程序。

引导

上电后装载系统程序。

引导轴

引导轴为 → 龙门架轴，以操作员和编程者的视角呈现并与普通的 NC 轴一样运行。

循环

受保护的子程序，用于执行 → 工件上反复出现的加工过程。

快速数字输入/输出

通过数字输入端可以启动快速 CNC 程序（中断程序）。 通过数字输出端可以触发快速、程序控制的开关功能（SINUMERIK840d）。

快速离开工件轮廓

当中断加工时，可以通过 CNC 加工程序引入一个动作，使刀具从所加工的工件轮廓快速离开。 此外还可以设定退刀的角度和位移的参数。 在快速提刀以后可以另外执行一个中断程序(SINUMERIK 840D)。

快速移动

轴运行最快速度。 比如，当刀具由静止状态运行到 → 工件轮廓或者由工件轮廓返回时使用快速移动速度。 快速移动速度可以根据不同机床在机床数据中设置。

成品轮廓

成品工件的轮廓。 参见 →毛坯件。

报警

所有→ 信息和报警均在操作面板上显示带有日期和时间的文本，并有相应的用于删除标准的符号。 报警和信息单独显示。

1. 零件程序中的报警和提示信息

   报警和提示信息可以直接以明码文本的形式从零件程序中显示。

2. PLC 报警和提信息

   机床报警和提示信息可以直接以明码文本的形式从 PLC 程序中显示。 在此无需另外的功能块软件包。

接地

接地是指在设备中连接到一起的所有无源器件，它们即使在出现故障时也不会有危险电压。

插补器

→ NCK 的逻辑单元，根据零件程序中目标位置的参数确定进给轴待运行的中间值。

操作界面

操作界面（BOF）是 CNC 控制系统的显示形式，带屏幕。 它带有水平软键和垂直软键。

攻丝，不带补偿衬套

用此功能可以不带补偿衬套攻丝螺纹。 通过插补运行，主轴作为回转轴和钻削轴将螺纹精确切削至终点钻削深度，比如盲孔螺纹（前提条件： 主轴作为进给轴运行）。

数据传输程序 PCIN

PCIN 是一种辅助程序，通过串行接口发送和接收 CNC 用户数据，如零件程序、刀具补偿等等。PCIN 程序可以在标准工业 PC 中的 MSDOS 下运行。

数据块

1. 数据单元， → PLC 可以对 → HIGHSTEP 程序进行存取。
2. →NC 的数据单元： 数据块包含全局用户数据的数据定义。 数据可以在定义时直接初始化。

数据字

在 → 文件中两个字节大小的数据单位。

文本编辑器

参见 → 编辑器

斜面加工

在工件表面进行钻削和铣削加工，它们不在机床坐标平面，但是可以通过¡°斜面加工¡±功能很方便地实现。

曲率

轮廓的曲率 k 是轮廓点上该段圆弧半径 r 的倒数 (k = 1/r)。

机床固定点

由机床明确定义的点，比如参机床参考点。

机床坐标系

以机床轴为基准的坐标系。

机床控制面板

机床的控制面板有各个操作按键、旋钮开关等，以及各个显示单元如 LED。 它们直接通过 PLC 对机床进行控制。

机床零点

机床固定点，所有测量系统均可以以此点为出发点。

极坐标

极坐标系指在一个平面中确定一个点的位置，它由到零点的距离与半径矢量和一个轴之间的夹角确定。

极限转速

最大/最小（主轴）转速： 通过在机床数据、→ PLC 数据或者 → 设定数据中的规定，可以限制主轴的最大速度。

标准循环

对于经常出现的加工情形，可以使用标准循环：

- 适用于钻削/铣削
- 用于旋转技术

在“程序”操作区“循环辅助”菜单下，列出所有供使用的循环清单。 选择了所要求的加工循环后，屏幕上会显示参数赋值指令中必须设定的参数。

样条插补

通过样条插补，控制系统可以由理论轮廓上较少的、给定的支点生成一条光滑的曲线。

模块

模块是指编程和程序执行时所需要的所有文件。

比例尺

是构成 → 框架的一个部分，可以改变某个轴的比例尺。

毛坯

毛坯指用于工件加工的原材料。

波特率

数据传输时的速度（位/秒）。

清零

在清零时，→ CPU 中以下的存储器将被清零：

- → 工作存储器
- → 装载存储器的读写区
- → 系统存储器
- → 备份存储器

用户存储器

所有的程序和数据，比如零件程序、子程序、注释、刀具补偿、零点偏移、框架以及通道和程序用户数据均可以存储到共同的 CNC 用户存储器中。

用户定义变量

用户可以定义用户变量，从而可以在 → 零件程序或者数据块（全局用户数据）中任意使用。 一个定义通常含有数据类型和变量名称。 参见 → 系统变量。

用户程序

可编程控制器 S7-300 中的用户程序用 STEP7 语言编写。 用户程序为模块化结构，由各个模块构成。

基本的模块类型有：

- 代码模块

  该模块含有 STEP 7 指令。

- 数据模块

  该模块包含有用于 STEP7 程序的常量和变量。

直线轴

与回转轴相反，直线轴指按直线运行的轴。

程序段

程序块包含 → 零件程序的主程序和子程序。

程序段查找

在进行零件程序测试时或者在中断一个加工后，可以通过“程序段搜索”功能找到程序中的任意位置，在此位置加工可以启动或者继续。

系统变量

无需 → 零件程序编程员的工作，已经存在的变量。 它由数据类型和变量名称定义，变量名称由符号**$** 引入。 参见 → 用户定义的变量。

系统存储器

系统存储器是 CPU 中的一个存储器，其内容为：

- 操作系统所需要的数据
- 运算的定时器、计数器和标志位

线性插补

刀具以直线运行到目标点，同时进行工件的加工。

绝对尺寸

进给轴在某一方向上移动说明，表明在当前坐标系中离开零点的距离。 参见 → 增量尺寸。

编程码

编程码是一种字符和字符串，它们在用于 编程码是一种字符和字符串，它们在用于 → 零件程序的编程语言中具有确定的含义。

编辑器

利用编辑器可以进行程序/文本/程序段的创建、修改、补充、合并和插入。

网络

网络指通过 → 连接电缆连接多个 S7-300 和其它终端设备，比如一台编程器。 通过网络进行相连设备之间的数据交换。

翻转

→ 框架的一个部分，定义坐标系按照一定的角度进行旋转。

自动方式

控制系统的运行方式（程序段连续运行，符合 DIN 标准）： NC 系统中的运行方式，这种方式下选择 → 零件程序并连续加工执行。

英寸尺寸系统

定义长度为“英寸”及其下级小数单位的尺寸系统。

螺旋线插补

螺旋线插补特别适用于利用成形铣刀简单地加工内螺纹和外螺纹，以及铣削润滑槽。

在这里螺旋线由两个运动组成：

- 平面中的回转运动
- 与此平面垂直的直线运动

补偿值

测量传感器所测得的轴位置与所要的、编程的轴位置之间的差值。

补偿存储器

控制系统中的一个数据区，刀具补偿数据存储在其中。

补偿表

支点表。 补偿表给基准轴所选择的位置提供补偿轴的补偿值。

补偿轴

设定值或者实际值可以通过补偿值进行修改的轴。

装载存储器

在 → SPS 的 CPU314 中，装载存储器就等同于 → 工作存储器。

设定数据

设定数据确定机床的性能，按照系统软件定义的方法在系统中设定。

诊断

1. 系统操作区
2. 控制系统不仅有自诊断程序，而且还可以进行维修时辅助测试。 状态、报警和服务信息

象限误差补偿

在象限过渡时，由于在导轨面上出现不同的摩擦而引起的轮廓误差，可以通过象限误差补偿予以消除。 象限误差补偿的参数可以通过圆弧形状测试确定。

轨迹轴

轨迹轴指所有通道的加工轴 → 通道由 → 插补器控制，它们可以同时启动、加速、停止直至到达终点。

轨迹进给

轨迹进给影响 → 轨迹轴。 表明相关 → 几何轴其进给量的几何量总和。

轨迹速度

最大可编程轨迹速度与进给精度有关。 比如精度为 0.1 毫米，则可编程的最大轨迹速度为 1000 米/分钟。

转换

附加或者绝对的轴零点偏移

轮廓

工件 → 轮廓

轮廓监控

作为轮廓监控的尺寸，滞后量误差控制在一个可定义的公差带之内。 比如，当驱动负载过大时就可能产生一个不允许的、过高的滞后量误差。 在这种情况下会产生一个报警，从而轴停止运行。

软件限位开关

软件限位开关限制一个轴的移动范围，阻止滑枕冲撞硬件限位开关。 每个轴可以给定 2 组数值，它们可以由 → PLC 分别激活。

软键

软键在屏幕上显示，具有对应的区域，可以动态地与当前的操作情形相对应。 这些功能键（软键）可以自由分配，它们由软件按照定义的功能进行分配。

轴名

参见→ 轴名称

轴名称

按照 DIN 66217 标准，垂直右旋 →坐标系中轴的名称为 X,Y,Z。

围绕 X、Y、Z 旋转的 → 回转轴名称为 A、B、C。其它平行的进给轴可以用其它地址字母标识。

轴地址

参见 →轴名称

辅助功能

在 → 零件程序中，使用辅助功能可以把机床制造商定义的 → 参数传送到 → PLC 中，并释放其所定义功能。

辅助程序段

由“N”引导的程序段，包含一个加工步骤的信息，比如一个位置数据。

运行方式

SINUMERIK 控制系统的运行过程控制方式。 定义的运行方式有 → Jog, → MDA, → 自动。

运行方式组

工艺相关的轴和主轴可以总结为一个工作方式组（BAG）。 一个 BAG 的加工轴和主轴可以由 1 个或多个 → 通道控制。 同一个工作方式组中的通道均有相同的 → 工作方式。

运行范围

线性轴中最大允许的运行范围可以达到±9 位。 绝对值取决于所选择的输入精细度和位置控制精细度，以及单位制（英制或者公制）。

返回固定点

机床中可以定义一些固定点，比如刀具更换点、装料点、托盘更换点等等，并可返回。 这些点的坐标存储到控制系统中。 控制系统控制相关轴运行，如果可能，以 → 快速方式运行。

进给倍率

通过机床控制面板或者→ PLC 可以调节实际速度，并覆盖编程的速度（0－200%）。 另外，进给速度也可以在加工程序中，通过一个编程的百分比（1－200%）进行修改。

进给轴

数控系统中的进给轴根据其功能可以分为：

- 轴： 可插补的轨迹轴
- 辅助轴： 不可插补的横向进给和定位轴，具有轴向进给功能。 辅助轴不参与加工，比如刀具供料器、刀具库。

连接电缆

连接电缆指预制的或者由用户自己定制的 2 芯电缆，带 2 个插头。 连接电缆通过多端口接口（MPI）把→ CPU 与→ PG 或者其它 CPU 相连。

连续路径运行

连续路径运行的目的在于，可以在零件程序段转换点处避免 → 轨迹轴停止加工，尽可能以相同的速度转到下一个程序段。

通道

一个通道是指可以单独处理一个 → 零件程序，而与其它的通道无关。 一个通道仅控制其所分配的进给轴和主轴。 不同通道的零件程序其加工过程可以通过 →同步功能进行协调。

速度控制

在轴移动时，为了可以使每个较小行程的程序段达到一个可以承受的运行速度，可以使用处理多个程序段的预读功能（→ Look Ahead）。

钥匙开关

→ 机床控制面板上的钥匙开关占据 4 个位置，它们由控制系统的操作系统分配相应的功能。 钥匙开关有 3 个不同颜色的钥匙，它们可以在所给定的位置插拔。

镜像

使用镜像功能，使加工轮廓相关轴的坐标值符号相反。 可以同时对多个轴进行镜像。

间隙补偿

对机械机床间隙进行补偿，比如滚珠丝杠的反向间隙。 对于每个轴，可以分别输入间隙补偿。

零件程序

NC 控制系统中的连续指令，它们一起制造出指定的 → 工件。 也就是说，在一个所提供的 → 毛坯上进行指定的加工。

零件程序段

→ 零件程序的一个部分，换行后结束。 分为→ 主程序段和→ 辅助程序段。

零件程序管理

零件程序可以按照→ 工件管理。 用户存储器的尺寸确定所管理的程序和数据的数量。 每个文件（程序和数据）可以命名最多 24 个字母数据字符的名称。

零点偏移

在一个坐标系中，相对于目前的零点和 → 框架规定一个新的基准点。

1. 可设定

   SINUMERIK 840D: 对于每个 CNC 轴，可以设定不同数量的零点偏移。 通过 G 功能可选择的偏移可以选择性地使用。

2. Extern

   另外，对于用于确定工件零点位置的所有偏移值，可以通过手轮（DRF 偏移）或者由 PLC 叠加一个外部零点偏移。

3. 可编程

   使用 TRANS 指令可以给所有的轨迹轴和定位轴编程零点偏移。

预控制，动态

滞后量误差所决定的→ 轮廓误差，几乎可以通过动态的、由加速度决定的预控制消除。由此可以获得一个非常好的加工精度，即使是在→ 轨迹速度很高的情况下。 预控制可以通过→ 零件程序根据相应的轴选择或者撤销选择。

预符合

如果轨迹路径接近设定三角形的终端位置，则进行程序段转换。

预读功能

利用功能**预读** 可以通过“预读”几个可参数 化的程序段而获取加工速度的最优化。

驱动

属于 CNC 的组件，它执行 NC 预设的转速和扭矩控制。

索引

符号

B

C

D

G

I

P

Q

R

S

T

U

V

W

X

A

一划

二划

三划

四划

五划

六划

七划

八划

九划

八划

九划

十划

十一划

十二划

十三划

十四划

十五划

十六划