编程说明 版本10/2004

sinumerik

SINUMERIK 840D/840Di/810D

工作准备部分

SIEMENS

SIEMENS

SINUMERIK 840D/840Di/810D

工作准备部分

编程手册

适用于控制系统
SINUMERIK 840D powerline
SINUMERIK 840DE powerline (出口型)
SINUMERIK 840Di
SINUMERIK 840DiE (出口型)
SINUMERIK 810D powerline
SINUMERIK 810DE powerline (出口型)
软件 NC 版本 7

2004年10月版
6FC5 298-7AB10-3RP1

安全技术提示

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

仅允许安装和驱动与本文件相关的附属设备或系统。设备或系统的调试和运行仅允许由**合格的专业人员**进行。本文件安全技术提示中的合格专业人员是指根据安全技术标准具有从事进行设备、系统和电路的运行，接地和标识资格的人员。

按规定使用

请注意下列说明：

警告

设备仅允许用在目录和技术说明中规定的使用情况下，并且仅允许使用西门子股份有限公司推荐的或指定的外部设备和部件。设备的正常和安全运行必须依赖于恰当的运输，合适的存储、安放和安装以及小心的操作和维修。

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。

西门子股份有限公司
自动化和驱动技术领域
Postfach 4848，D-90327 Nürnberg

前言

资料结构

SINUMERIK资料分为3种类型：

- 一般文献
- 用户文献
- 制造商/维修文献

读者对象

该资料面向机床用户。本手册详细描述了用户对 SINUMERIK 840D/810D 控制系统进行编程所需的实际内容。

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。

有关 SINUMERIK 840D/810D 的其它手册以及适用于所有 SINUMERIK 控制系统的手册（例如通用接口，测量循环.....）请向您的西门子分支机构索取。

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。

适用性

该编程说明适用于控制系统：

SINUMERIK 840D powerline	7
SINUMERIK 840DE powerline (出口版本)	7
SINUMERIK 840Di	3
SINUMERIK 840DiE (出口版本)	3
SINUMERIK 810D powerline	7
SINUMERIK 810DE powerline (出口版本) 带有操作面板 OP 010, OP 010C, OP 010S, OP 12 或者 OP 15 (PCU 20 或者 PCU 50)	7

出口版本

在出口版本中不含有以下功能：

功能	810DE	840DE
5轴加工软件包	–	–
操作转换软件包（5轴）	–	–
多轴插补（>4轴）	–	–
螺旋线插补2D+6	–	–
同步动作，级别2	–	O1)
测量，级别2	–	O1)
适配控制	–	O1)
连续修整	–	O1)
使用编译循环（OEM）	–	–
垂度补偿，多维	–	O1)
	– 功能不可用 1) 功能受到限制	

帮助热线与网址

有问题时请打以下热线电话：

自动化与驱动集团技术支持部门，
电话： +49 (0)180 5050-222

传真： +49 (0)180 5050-223

资料方面有疑问时（建议，更正）请发传真或电子邮件至：

传真： +49 (0)9131 98-2176

电子邮件：motioncontrol.docu@erlf.siemens.de

传真格式：参见手册封底所附带的表格。

说明

基础部分

编程说明“基础部分”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示例，说明常见的指令和语句（符合DIN66025）。

工作准备部分

编程说明“工作准备部分”供熟悉所有编程方法的工艺人员使用。SINUMERIK 840D/810D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。

本编程说明中所述的指令和语句与工艺无关。

例如可以用于：

- 磨削
- 循环加工（包装加工，木材加工）
- 激光功率控制

目录

灵活的NC编程

1

1.1 变量和计算参数（用户自定义变量，计算参数，系统变量）

功能

通过使用变量而非固定值就可以灵活地编制程序。您可以以此对信号（例如测量值）作出响应，或者通过将变量用作设定值的方式将同一个程序用于各种几何形状。

结合变量计算和程序跳转，就可使熟练编程人员极为灵活地建立程序文档并且可节省许多编程时间。

变量类型

控制系统区分出三种变量类型：

用户定义变量	由用户定义名称和类型的变量，例如计算参数。
计算参数	专门的、预定义的计算变量，给定地址R及随后的数字。预定义的计算变量类型为REAL。
系统变量	供控制系统使用的变量，它们可以在程序中进行处理（写，读）。系统变量可供访问零点位移、刀具补偿、实际值、轴的测量值、控制系统的状态等等 （系统变量的含义请参阅附录）。

变量类型

类型	意义	值范围
INT	整数值，带符号	$\pm(2^{31} - 1)$
实数	实数 (带小数点的分数, LONG 实数按照 IEEE)	$\pm(10^{-300} \ldots 10^{+300})$

BOOL	逻辑值：TRUE真 (1) 和 FALSE假 (0)	1.0
CHAR	ASCII字符，相应编码	0 … 255
字符串	字符串, 字符数，在 […]中, 最多200个字符	带0…255的数值
AXIS	仅为轴名称（轴地址）	所有在通道中出现的轴名
FRAME	位移、旋转、缩放、镜像的几何数据，参见“框架”章节	

计算变量

正常情况下，如果没有做进一步说明，则在地址R下有100个计算变量供使用，数据为实数型。

计算变量的具体个数（最大32535）由机床参数决定。

举例：R10=5

系统变量

控制系统提供系统变量，这些变量可以供所有程序运行并执行。

系统变量用来提供机床和控制系统的状态。它们有时不可以描述。

系统变量综述

为了专门标识，系统变量的名称总是以 "$"符号开始的。紧接着的是专门的名称。

1. 字母	意义
$M	机床数据
$S	设定数据
$T	刀具管理参数
$P	程序数值
$A	实际数值
$V	服务参数

2. 字母	意义
N	NCK-全局
C	通道专用
A	轴专用

举例：$AA_IM

表示：机床坐标系统中当前轴的实际值。

1.2 变量定义（DEF 用户自定义变量 LUD，GUD，PUD）

功能

除了预设的变量，编程者还可以确定自己的变量，并用数值加以注明。

局部变量（LUD）仅在其被定义的那个程序中才有效。

全局变量（GUD）在所有程序中都有效。

通过机床日期可以将主程序中已经定义过的局部自定义变量（LUD）重新定义为程序全局自定义变量（PUD），参见用户示例。

机床制造商

参见机床制造商说明。

编程

变量类型 INT

DEF INT 名称

或者

DEF INT 名称=值

变量类型 实数

DEF 实数 名称

或者

DEF 实数 名称1，名称2=3，名称4

或者

DEF 实数 名称[数组索引1，数组索引2]

变量类型 BOOL

DEF BOOL 名称

变量类型字符

DEF CHAR 名称

或者

DEF CHAR 名称[数组索引]=("A","B",…)

变量类型字符串

DEF STRING[字符串长度] 名称

变量类型轴

DEF AXIS 名称

或者

DEF AXIS 名称

变量类型框架

DEF FRAME 名称

注意

如果在定义时没有给变量赋值，那么系统将之预定为0。

变量必须在使用之前、在程序开始时定义。必须在一个自身的程序段中进行定义；每个程序段只能定义一种变量类型。

参数

INT	变量类型 Integer，即整数
实数	变量类型 实数，即有小数点断开的数字
BOOL	变量类型 Bool，即 1 或者 0（TRUE 或者 FALSE）
CHAR	变量类型 Char，即字符，相当于 ASCII 码（0 ~ 255）
字符串	变量类型 String，即字符串
AXIS	变量类型 Axis，即轴地址和主轴
FRAME	变量类型 Frame，即几何数据

举例

变量类型	意义
INT	
DEF INT 数目	将数目名称设定一个 Integer 型变量。 系统默认为零。
DEF INT ANZAHL=7	设定一个名为数目的整数型变量。变量的起始值为7。
实数	
DEF 实数 深度	设定一个名为深度的实数型变量。 系统预设为0。
DEF 实数 深度=6.25	设定一个名为深度的实数型变量。变量的起始值为6.25。
DEF 实数 深度=3.1,长度=2,数目	在一行中可以定义多个变量。
BOOL	
DEF BOOL WENN_ZUVIEL	设定一个名为WENN_ZUVIEL的布尔型变量。 系统预设为0（FALSE）。

DEF BOOL WENN_ZUVIEL=1 或者	设定一个名为WENN_ZUVIEL的布尔型变量。
CHAR	
DEF CHAR GUSTAV_1=65	可以直接为字符类型的变量赋值一个相应的ASCII-字符或
DEF CHAR GUSTAV_1="A"	ASCII-字符的代码值(代码值 65 相应于字母 "A")。
字符串	
DEF STRING[6] MUSTER_1="ANFANG"	字符串类型的变量可以接收一个字符链。字符个数的最大值列在变量类型后面的方括号中。
AXIS	
DEF AXIS ACHSNAME=(X1)	AXIS 类型的变量名为 ACHSNAME 并且含有一个通道的轴标识符 - 此处为 X1。 （带有扩展地址的轴名称在圆括号内。）
FRAME	
DEF FRAME SCHRAEG_1	框架类型的变量名称为 SCHRAEG_1。

注意

轴类型的一个变量有一个通道的轴名称和主轴名称。

注意

带扩展地址的轴名称必须写在圆括号中。

举例重新定义局部 (LUD) 和程序全局自定义变量 (PUD)

当它们在主程序中被定义之后，它们在被调用的子程序的所有级上都是有效的。它们随着零件程序起始而设置，随着零件程序结束或复位而被删除。

如果机床参数 $MN_LUD_EXTENDED_SCOPE
已经设定，就不能在主程序和子程序中用相同的名称再去定义一个变量。

```
$MN_LUD_EXTENDED_SCOPE=1
PROC MAIN                                   ;主程序
DEF INT VAR1                                ;PUD-定义
...                                         ；子程序调用，
SUB2
...
M30

PROC SUB2                                   ;子程序SUB2
DEF INT VAR2                                ;LUD-定义
...
IF (VAR1==1)                                ;PUD 读取
 VAR1=VAR1+1                                ;PUD 读取和写入
 VAR2=1                                     ;LUD 写入
ENDIF                                       ；子程序调用，
SUB3
...
M17

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:来自 SUB2 的LUD未知
ENDIF
...
M17
```

变量名称

一个变量名称最多由31个符号组成。前面两个符号必须是字母或下划线。

字符 "$" 不可以用于自定义变量，因为该字符为系统变量所使用。

程序局部变量举例

```
DEF INT ZAEHLER
SCHLEIFE:G0 X                               循环
ZAEHLER=ZAEHLER+1
IF ZAEHLER<50 GOTOB SCHLEIFE
M30
```

查询现有几何轴举例

```
DEF 轴 横坐标                                    ;1. 几何轴
IF ISAXIS(1) == FALSE GOTOF WEITER
横坐标 = $P_AXN1
继续:
```

间接主轴编程举例

```
DEF 轴 主轴
主轴=(S1)
OVRA[主轴]=80                                    ;主轴倍率 = 80%
主轴=(S3)
…
```

1.3 数组定义 (DEF, SET, REP)

功能

数组是通过有名称和大小的变量类型来定义的存储块。数组可以最多由2维尺寸定义。

数组的最大规格由机床参数设定。

机床制造商

参见机床制造商资料

一个数组可能比一个存储块大。选择存储块大小的MD值时，应只有在例外情况下才会出现数组分割。

数组的初始化

可以将初始化值赋给数组元素：

- 在程序执行过程中

或者

- 在定义数组时。

在两维的数组中,右边的数组变址首先增值。

编程

```
DEF CHAR NAME[n,m]
```

或者

```
DEF INT NAME[n,m]
```

或者

DEF 实数 NAME[n,m]

或者

DEF AXIS NAME[n,m]

或者

DEF FRAME NAME[n,m]

或者

DEF STRING[字符串长度] NAME[m]

或者

DEF BOOL[n,m]

- **使用值列表初始化；SET**

定义数组时的方法

DEF Typ VARIABLE = 设定(值)

DEF Typ 数组[n,m] = 设定(值, 值, …)

或者

DEF Typ VARIABLE = 值

DEF 类型 数组[n,m] = (值 , 值 …)

注意

在数组定义中您可以有选择的给出设置项。

程序运行方法

数组[n,m]= 设置(值, 值, 值,…)

数组[n,m]= 设置(表达式, 表达式, 表达式,…)

- **使用相同值初始化，REP**

定义数组时的方法

DEF 类型 数组[n,m] = REP(值)

注意

框架类型的变量不能被赋予初值。

程序运行方法

数组[n,m] = REP(值)

数组[n,m] = REP(表达式)

注意

允许框架类型的变量,因此可以很方便地初始赋值。

参数

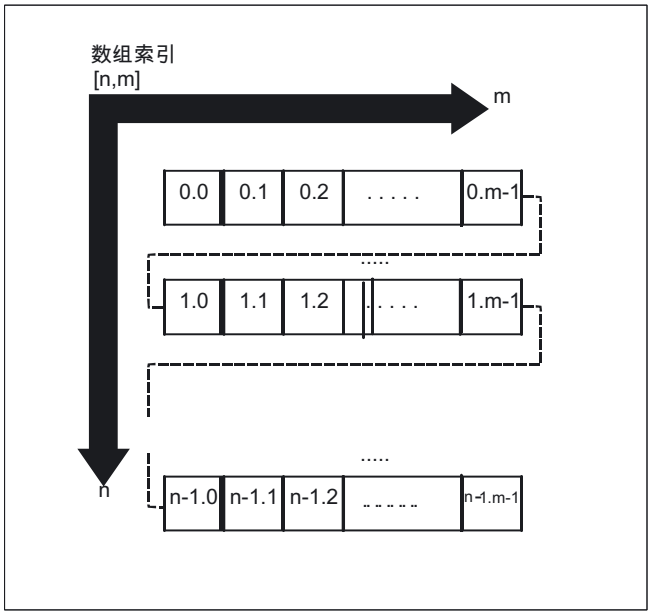
DEF 变量类型	数组定义
SET 值或者表达式	在定义数组或者程序执行时用值列表初始化
REP 值或者表达式	在定义数组或者程序执行时用相同的值初始化
CHAR 名称[n,m] INT 名称[n,m] 实数 名称[n,m] AXIS 名称[n,m] FRAME 名称[n,m] BOOL[n,m]	变量类型 (CHAR, INTEGER, 实数, AXIS, FRAME, BOOL)
STRING[字符串长度] 名称[m]	数据类型 STRING 只能用一维数组定义。 根据数据类型 STRING 规定字符串长度。
名称	变量名
变量类型	变量类型 (CHAR, INTEGER, 实数, AXIS, FRAME, BOOL)
TYP FELD[n,m] = SET(值，值，...)	在定义数组时使用列出的值初始化某个数组的所有元素
TYP FELD[n,m] = REP(值)	在定义数组时使用相同的值初始化某个数组的所有元素
FELD[n,m] = SET(值，值， ..) FELD[n,m] = SET(表达式， ...)	在执行程序时使用列出的值初始化某个数组的所有元素
FELD[n,m] = REP(值) FELD[n,m] = REP(表达式)	在执行程序时使用相同的值初始化某个数组的所有元素
FELD[n, m]	数组索引
n	第1维的数组大小
m	第2维的数组大小

带有字符串类型的变量的数组只允许是单尺寸。

数组索引 [n,m]

可以通过数组变址读取数组单元。通过该数组变址,数组单元设置数值,或者读取数组单元值。

第一个数组单元从变址[0,0]开始；在数组大小为[3,4]时，最大可能的数组变址为[2,3]。



内存需求

变量类型	**每个元素的内存需求**
BOOL	1 字节
CHAR	1 字节
INT	4 字节
实数	8 字节
字符串	字符串长 + 1
FRAME	~ 400 字节, 取决于轴数目
AXIS	4 字节
最大数组大小	标准：812 字节

注意

最大的数组大小决定了存储块的规格，在此对变量存储器进行管理。此规格不应大于所需大小。

标准：812 字节

如果没有定义大数组，就选择：256 字节

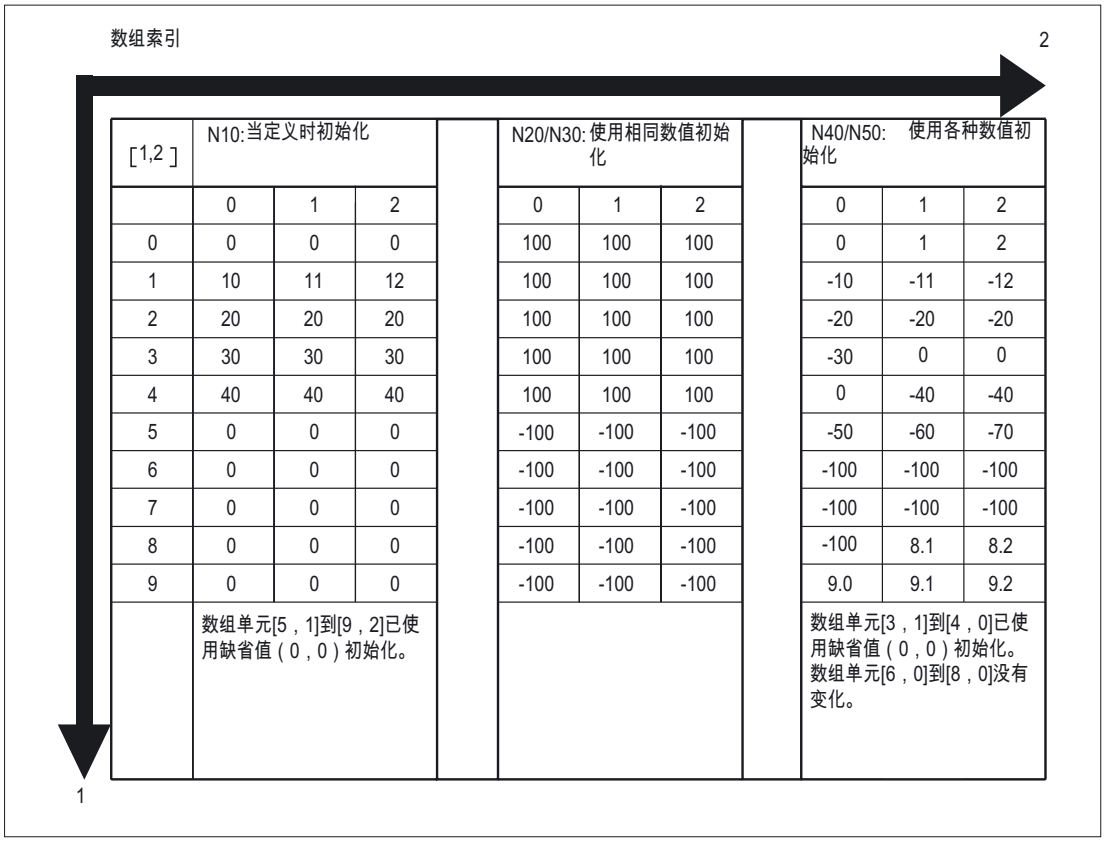
举例:定义 BOOL数组

全局用户数据应当含有接通/关闭控制系统的PLC机床数据。

初始化全部变量数组举例

实际的数值占用情况见图。

```
N10 DEF 实数 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		
	0	1	2
0	0	0	0
1	10	11	12
2	20	20	20
3	30	30	30
4	40	40	40
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
	数组单元[5，1]到[9，2]已使用缺省值（0，0）初始化。		

N20/N30: 使用相同数值初始化		
0	1	2
100	100	100
100	100	100
100	100	100
100	100	100
100	100	100
-100	-100	-100
-100	-100	-100
-100	-100	-100
-100	-100	-100
-100	-100	-100

N40/N50: 使用各种数值初始化		
0	1	2
0	1	2
-10	-11	-12
-20	-20	-20
-30	0	0
0	-40	-40
-50	-60	-70
-100	-100	-100
-100	-100	-100
-100	8.1	8.2
9.0	9.1	9.2
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。		

在定义数组时使用值列表初始化，SET

- 有多少初值被编程就有多少数组元被赋值。
- 没有值的数组元(数值表中的空白)会自动被填上0。
- 轴类型的变量是不允许有空白的。
- 如果被编程的值超过现有的剩余数组元,就会触发系统警报。

举例：

```
DEF 实数 FELD[2,3]=(10, 20, 30, 40)
```

在程序执行过程中使用值列表初始化，SET

- 赋予初值与定义时一样。
- 作为数值在此也可以使用表达式。
- 赋予初值开始于编程的数组变址。因此目标明确的部分数组也可以被数值占用。

举例：表达式赋值

```
DEF INT FELD[5, 5]
FELD[0,0] = 设置(1, 2, 3, 4, 5)
FELD[2,3] = 设置(变量, 4*5.6)
```

在使用轴变量时,轴变址不能运行：

举例：在一行中初始化

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

与之相应的:

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

在定义数组时使用相同的值初始化，REP

所有的数组元被相同的数值(常量)占用。

框架类型的变量不能被赋予初值。

举例：

```
DEF 实数 FELD5[10,3] = REP(9.9)
```

在程序执行过程中使用相同的值初始化

- 作为数值在此也可以使用表达式。
- 使用相同的值初始化所有数组元素。
- 赋予初值开始于编程的数组变址。因此目标明确的部分数组也可以被数值占用。

举例：使用某个值初始化所有元素

```
DEF FRAME FRM[10]
FRM[5] = REP(CTRANS (X,5))
```

1.4 间接编程

功能

通过间接编程可以使程序通用。同时扩展地址(变址)被合适类型的变量代替。

间接G代码编程

通过变量间接编程G代码，可以进行有效的循环编程。对此有两个参数

带有Integer型常量的G代码组

带有整数/实数型变量的G代码编号

供使用。

编程

```
ADRESSE[INDEX]
```

或者

```
G[<组索引>] = <整数/实数-型变量>
```

通过用于一个有效循环编程的变量对G代码进行间接编程

参数

所有的地址都是可以设定参数的,除了：

- N - 程序段代码
- L - 子程序

对于所有的可调整地址,间接编程是不可能的。

举例：X[1] 代替 X1 是不允许的。

ADRESSE	带有参数说明的地址，作为索引
[INDEX]	变量索引，例如主轴代码，轴 ....
G<组索引	G代码组：选择带有G代码组的Integer常量
<整数/实数-型变量>	G代码编号：选择带有G代码编号的Integer或者Real型变量

有效的G代码组

只有模态有效的G代码组可以间接编程。

程序段有效的G代码组被拒绝，并给出警报12470。

有效的G代码号

在G代码间接编程中不允许进行计算。

G代码编号必须保存在Integer或者Real型变量中。无效的G代码号被拒绝，并发出报警12475。

必须在对G代码进行间接编程之前，在一个自身的零件程序行中进行必要的G代码变化计算。

注意

所有有效的G代码均在 PG，“G功能/行程条件列表”章节中分组描述。参见 /PG/ 编程说明基础部分，“表格”

举例

主轴

```
S1=300                          ;直接编程
DEF INT SPINU=1                 ;间接编程
S[SPINU]=300                    ;主轴转速 300 转/分钟，
                                ;其在变量SPINU中的编号
                                ;被保存（在该示例1中）。
```

进给

```
FA[U]=300                       ;直接编程
DEF AXIS AXVAR2=U               ;间接编程
FA[AXVAR2]=300                  ;定位轴进给，其
                                ;在 AXIS 型变量中的地址名称，使用
                                ;变量名称 AXVAR2 保存。
```

测量值

```
$AA_MM[X]                       ;直接编程
DEF AXIS AXVAR3=X               ;间接编程
$AA_MM[AXVAR3]                  ;轴在机床坐标中的测量值，
                                ;其名称在变量 AXVAR3 中
                                ;保存。
```

数组单元

```
DEF INT 数组1[4,5]              ;直接编程
DEFINE DIM1 AS 4                ;间接编程
DEFINE DIM2 AS 5
DEF INT 数组[DIM1,DIM2]         ;对于数组维，必须将数组大小
                                ;设定为固定值。
数组[DIM1-1,DIM2-1]=5
```

带轴变量的轴指令

```
X1=100 X2=200                   ;直接编程
DEF AXIS AXVAR1 AXVAR2          ;间接编程
AXVAR1=(X1) AXVAR2=(X2)         ;变量定义
AX[AXVAR1]=100 AX[AXVAR2]=200   ;指定轴名称 将
                                ;保存在变量中的轴移动
                                ;到 100 或者 200。
```

```
带轴变量的插补参数
G2 X100 I20                          ;直接编程
DEF AXIS AXVAR1=X                    ;间接编程
G2 X100 IP[AXVAR1]=20                ;定义并且指定轴名称
                                     ;对
                                     ;中心点数据进行间接编程
间接子程序调用
CALL "L" << R10                      ;调用其编号在 R10 ;中的程序
```

注意

R参数也可以被理解为带有简略的记数法的一维数组（R10与R[10]相对应）。

间接G代码编程举例

可调零点位移 G代码组8

```
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
```

平面选择 G代码组6

```
N2010 R10 = $P_GG[6]                 ；读当前平面的G代码
...
N2090 G[6] = R10                     ;G17
```

1.4.1 将字符串作为零件程序行执行 (EXECSTRING)

功能

通过零件程序指令EXECSTRING可以把一个字符串作为一个参数来传送，该字符串含有一个需要执行的零件程序行。

编程

EXECSTRING (<字符串变量>)

参数

EXECSTRING	给出一个字符串变量，带待执行的零件程序行
(<字符串变量>)	带有需要执行的零件程序行的参数

注意

所有的零件程序结构都可以被取消，它们可以在零件程序的程序部分被编程。PROC-和DEF-指令被排除在外，在INI和DEF文件中的应用也不行。

间接零件程序行举例

```
N100 DEF 字符串[100] 程序块                    ;用来输入零件程序行 ;的字符串变量
N110 DEF 字符串[10] MFCT1 = "M7"

N200 EXECSTRING(MFCT1 << " M4711")          ;执行零件程序行 "M7 M4711"

N300 R10 = 1
N310 程序块 = "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 M4711"
```

1.5 赋值

功能

变量/计算参数可以在程序中被赋予一个合适类型的值。

编程

赋值总是要求一个独立的程序段；每个程序段可以有多个赋值语句。给轴地址赋值（运行指令）与变量赋值相反要求一个分开的程序段。

参数

给字符串-变量赋值

在字符或字符串中区分大小写。

如果'或者"为字符链中的组成部分，则它们就被括 在'...\'中。

举例：

```
MSG("Viene lavorata l'"ultima figura")
```

在显示屏上显示文本'Viene lavorata l'ultima figura'。

不能显示的字符会以二进制或十六进制的常量接收到字符串中。

举例

```
R1=10.518 R2=4 VARI1=45          ;数字值的赋值
X=47.11 Y=R2
R1=R3 VARI1=R4                   ;适当类型变量的赋值
R4=-R5 R7=-VARI8                 ;分配相反的前置符
                                 ;(仅当 INT 和 REAL 型时允许)
```

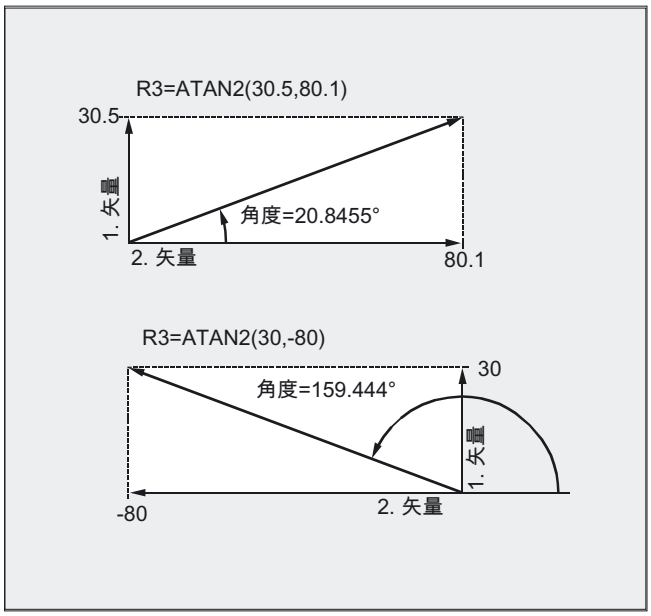
1.6 计算/计算功能

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。

计算功能ATAN2(,)

这个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 (–° < 0 < +180°)。角度是指在正方向的第二个数值。



比较命令的精确度，可使用 TRUNC( ) 设置

参见“比较命令的精确度补偿”

编程

算术运算子采用通常使用的数学写法。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 ）。

参数

运算子/运算功能

+	加法
-	减法
*	乘法
/	除法 **注意**：(INT型)/(INT型)=(REAL型) ;举例： 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) ;举例：3 DIV 4 = 0
MOD	模除运算 (INT 或者 REAL) 可使 INT-除法运算复位， 例如 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. 乘方（平方）
TRUNC()	整数部分
ROUND()	整数园整
LN()	自然对数
EXP()	指数函数
CTRANS()	偏移
CROT ()	旋转
CSCALE ()	比例转换
CMIRROR ()	镜像

初始化全部变量数组举例

```
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))                ;首先解决内部括号
                                         ;R15 = (R12+R22)的平方根
RESFRAME= FRAME1:FRAME2                  ;使用链接算子将括号
FRAME3=CTRANS(…):CROT(…)                 ;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
```

1.7 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。如果是CHAR型，就比较代码值。如果是STRING, AXIS和FRAME型，可以：== 和 <>.

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。通过隐式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

==

或者

<>

或者

>

或者

<

或者

>=

或者

<=

逻辑运算

AND

或者

OR

或者

NOT

或者

XOR

在布尔的操作数和运算符之间必须加入空格。

逐位逻辑运算

B_AND

或者

B_OR

或者

B_NOT

或者

B_XOR

参数

比较运算符的含义

==	相同于
<>	不等
>	大于
<	小于
>=	大于等于
<=	小于等于

逻辑运算的意义

AND	与
OR	或
NOT	非
XOR	异 - 或

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。

逐位逻辑运算符的意义

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"

注意

运算符 B_NOT 仅涉及一个运算对象；
该运算对象位于运算符之后。

比较运算符举例

```
IF R10>=100 GOTOF 目标
```

或者

```
R11=R10>=100
IF R11 GOTOF 目标
```

R10>=100比较的结果首先存储在R11中。

逻辑运算符举例

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

或者

```
IF NOT R10 GOTOB START
```

NOT只与一个操作数有关。

逐位逻辑运算符

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

1.7.1 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。

编程

比较错误时的精确度校正

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

参数

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

出于兼容性的原因，可以通过设置机床数据 MD 10280:PROG_FUNCTION_MASK Bit0 = 1 ，在 (>) 和 (<) 情况下取消检查相对相等性。

注意

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。

精度检查举例

```
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 误差：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)                               ;精度修正
IF ABS(R12-R11) > R13 GOTOF FEHLER               ;不再执行跳转
M30                                              ； 程序结束
FEHLER:SETAL(66000)
```

得出并且分析两个运算数的商举例

```
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)
```

1.8 运算的优先级

功能

每个运算操作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

注意

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

如果(otto==10)和 (anna==20) gotof 结束

1.9 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	INT	BOOL	CHAR	字符串	AXIS	FRAME
从							
REAL	是	是*	是1)	是*	–	–	–
NT	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

注意

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。

1.10 字符串运算

概述

除了在这一章描述的常规运算，如“赋值”和“比较”，还有其它的方法用于字符串的处理。

参数

类型转换到字符串：	
STRING_ERG = <<bel._类型[1)]	结果类型：字符串
STRING_ERG = AXSTRING (AXIS)	结果类型：字符串
从字符串转换类型：	
BOOL_ERG = ISNUMBER (STRING)	结果类型：BOOL
REAL_ERG = NUMBER (STRING)	结果类型：REAL
AXIS_ERG = AXNAME (STRING	结果类型：AXIS
字符串的级联：	
bel._类型[1)] << bel.类型[1)]	结果类型：字符串
大小写字母转换	
STRING_ERG = TOUPPER (STRING)	结果类型：字符串
STRING_ERG = TOLOWER (STRING)	结果类型：字符串
字符串长度	
INT_ERG = STRLEN (STRING)	结果类型：INT
在字符串中查找字符/字符串	
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
部分字符串的选择	
STRING_ERG = SUBSTR (STRING, INT)	结果类型：INT
STRING_ERG = SUBSTR (STRING, INT, INT)	结果类型：INT
单个字符的选择。	
CHAR_ERG = STRINGVAR [IDX]	结果类型：CHAR
CHAR_ERG = STRINGFELD [IDX_FELD, IDX_CHAR]	结果类型：CHAR

[1)] "bel._Typ" 表示变量类型 INT, REAL, CHAR, STRING 和 BOOL.

字符0的特别意义

在系统内部，字符0是被作为一个字符串的结束标识的。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。

举例

```
DEF STRING[20] STRG = " 轴有"."
STRG[6] = "X"                          出现"X轴停止"的提示
MSG(STRG)
STRG[6] = 0
MSG(STRG)                              出现"轴"的提示
```

1.10.1 类型转换到字符串：

功能

因此不同类型的变量可以作为一个信息（MSG）的组成部分来使用。

使用运算符<<时，隐含适用于数据类型INT，REAL，CHAR和BOOL（参见"字符串级联"）。

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。

编程

句法

STRING_ERG = AXSTRING (AXIS)	**结果类型：STRING**

符号语义：

AXSTRING（轴）作为字符串提供所给出的轴名称。

参数

AXIS类型变量可以通过AXSTRING功能转换为STRING。

FRAME变量不能被转换。

举例：

```
MSG("位置:"<<$AA_IM[X])
```

举例

```
DEF STRING[32] STRING_ERG
STRING_ERG = AXSTRING(X)                    现在STRING_ERG == "X"
```

1.10.2 从字符串转换类型：

功能

通过NUMBER功能可以实现从STRING到REAL的转换。

当 ISNUMBER 给出值 FALSE 时，就会在调用带有相同参数的 NUMBER 时触发报警。

使用函数 AXNAME 可以将某个字符串转换成数据类型
AXIS。如果该字符串没有分配到设计的轴名称，则给出报警。

编程

句法

REAL_ERG = NUMBER (STRING)	**结果类型：REAL**
BOOL_ERG = ISNUMBER (STRING)	**结果类型：BOOL**
AXIS_ERG = AXNAME (STRING)	**结果类型：AXIS**

符号语义：

NUMBER（字符串）送回一个通过字符串显示的数，作为实数值。

当字符串显示一个根据语言规则有效的实数时，ISNUMBER （字符串）显示TRUE。
由此可以检测这个字符串是否能被转换为一个有效的数字。

AXNAME（字符串）转换所给出的字符串为一个轴名称。

举例

```
DEF BOOL BOOL_ERG
DEF REAL REAL_ERG
DEF AXIS AXIS_ERG
```

```
BOOL_ERG = ISNUMBER ("1234.9876Ex-7")      现在BOOL_ERG == TRUE
BOOL_ERG = ISNUMBER ("1234XYZ")            现在BOOL_ERG == FALSE
REAL_ERG = NUMBER ("1234.9876Ex-7")        现在REAL_ERG == 1234.9876Ex-7
AXIS_ERG = AXNAME("X")                     现在AXIS_ERG == X
```

1.10.3 字符串的链接

功能

这个功能使单个的字符串组合在一起。

通过运算符实现级联：<<.
这个运算符适用于所有基本类型CHAR，BOOL，INT，REAL和STRING的组合，变成目标类型字符串。一个可能发生的必要的转换将根据现行的规则进行。

编程

句法

bel._Typ << bel._Typ	**结果类型：STRING**

符号语义

所给的字符串（有时隐含转换的其它类型）被相互级联在一起。

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）。

FRAME和AXIS类型不能使用这个算符。

句法

<< bel._类型	**结果类型：STRING**

符号语义

所给的类型被隐含转换为字符串类型。

比如，自文本列表组成一个信息或一个命令，并且插入参数（大约一个模块名）：

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。

字符串的链接举例

```
DEF INT IDX = 2
DEF REAL VALUE = 9.654
DEF STRING[20]STRG = "INDEX:2"
IF STRG == "索引:"<<IDX GOTOF NO_MSG
MSG ("索引:"<<IDX <<"/值:"<<VALUE)          显示索引2/值 9.654"
NO_MSG:
```

1.10.4 大小写字母转换

功能

这个功能允许一个字符串的所有字母以统一的标准显示。

句法

STRING_ERG = TOUPPER (STRING)	**结果类型：STRING**
STRING_ERG = TOLOWER (STRING)	**结果类型：STRING**

符号语义

所有小写字母都变成大写或小写字母。

举例

因为也有可能与HMI/MMC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：

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

1.10.5 字符串长度

功能

这个功能允许确定字符串长度

句法

INT_ERG = STRLEN (STRING)	**结果类型：INT**

符号语义

可以使一些不是0字符（从字符串开始处数起）的字符恢复。

举例

例如允许在与下述单字符存取有关的情况下确定字符串的末尾：

```
IF(STRLEN (BAUSTEIN_NAME) > 10) GOTOF FEHLER
```

1.10.6 在字符串中查找字符/字符串

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。

编程

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。

参数

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函数，除非传送一个字符列表（作为字符串），从这些字符中恢复第一个被找到字符的索引。
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("/")。

将一个输入分解成路径名称和模块名称示例

```
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函数 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"
```

1.10.7 部分字符串的选择

功能

这个功能允许一个部分字符串从一个字符串中生成。对此给出第一个字符的变址，有时还有所要求的长度。如果没有说明长度，则指的就是剩余字符串。

编程

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一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。

举例

```
DEF STRING [29] ERG
ERG = SUBSTR ("应答：10 ~ 99", 10, 2)        由此ERG == "10"有效
```

1.10.8 单个字符的选择

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。

编程

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

这个字符在这个字符串内被读取/写入，这个字符在给定的位置。如果给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。

```
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-,GUD- und PUD-数据)。

此外在调用一个子程序时只可以对“数值调用”数据进行存取。

对某个系统数据、机床数据, ...进行单字符存取示例

```
DEF STRING [50] STRG
DEF CHAR QUITTUNG
…
STRG = $P_MMCA
QUITTUNG = STRG [0]                    ; 应答部件运用
```

对参考调用参数进行单字符存取示例

```
DEF STRING [50] STRG
DEF CHAR CHR1
EXTERN UP_CALL (VAR CHAR1)             ;参考调用-参数!
…
CHR = STRG [5]
UP_CALL (CHR1)                         ;参考调用
STRG [5] = CHR1
```

1.11 CASE指令

功能

这个CASE指令提供根据INT类型实际值不同而进行转移的可能。

被CASE指令检测的常量采用什么值,程序就转移到所属跳转目标确定的位置。

编程

```
CASE (表达式) OF 常量1 GOTOF LABEL1 … DEFAULT GOTOF LABELn
CASE (表达式) OF 常量1 GOTOB LABEL1 … DEFAULT GOTOB LABELn
```

参数

CASE	跳转指令的关键词
GOTOB	以反向跳转为目标的跳转指令(方向为程序开头)
GOTOF	以正向跳转为目标的跳转指令(方向为程序结尾)
GOTO	跳转目标首先是正向然后是反向的跳转指令(方向先是程序结尾然后是程序开头)

GOTOC	抑制报警14080"没有找到跳转目标"。 跳转目标首先是正向然后是反向的跳转指令(方向先是程序结尾然后是程序开头)
LABEL	目标(在程序内标记)
标签：	在跳转目标名称后有一个冒号。
表达式	数学表达式
恒定的	INT类型常量
DEFAULT	程序路径;如果没有前面的常量相吻合

注意

有关 GOTO 指令的更多信息可参阅第10章中的计算参数和程序跳转。

对于常量没有采用前面确定的值的情况,可以用DEFAULT指令确定跳转目标。

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。

举例1

CASE(表达式) OF 1 GOTOF LABEL1 2 GOTOF LABEL2 … DEFAULT GOTOF LABELn

„1挨

当表达式的值 = 1 (INT-常量), 跳转到带LABEL1的程序段

当表达式值= 2 (INT-常量),跳转到带LABEL2的程序段

…

依此类推跳转到带LABELn的程序段

举例2

```
DEF INT VAR1 VAR2 VAR3
CASE(VAR1+VAR2-VAR3) OF 7 GOTOF MARKE1 9 GOTOF MARKE2 DEFAULT GOTOF MARKE3
MARKE1:G0 X1 Y1
MARKE2:G0 X2 Y2
MARKE3:G0 X3 Y3
```

1.12 控制结构

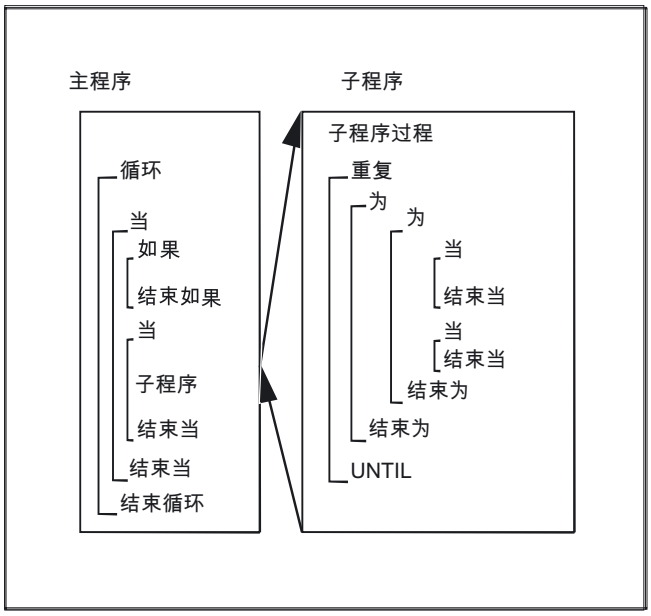
功能

控制系统按照编制好的标准顺序处理NC程序段。
使用这些指令（除了本章中所描述的程序跳转外），还可以确定其它选择和程序循环。
这些命令使编程具有某种结构并使程序具有较强的可读性。

编程

嵌套深度

控制结构对部分程序有效。在每个子程序之内,嵌套的层数可以达到8个标准控制结构。



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

参数

IF	二选一
LOOP	无限循环
FOR	计数循环
WHILE	在循环开头有条件的循环
REPEAT	在循环结尾有条件的循环

无限循环程序举例

```
%_N_LOOP_MPF
LOOP
 IF NOT $P_SEARCH                          ;没有程序段寻找操作
 G01 G90 X0 Z10 F1000
 WHILE $AA_IM[X] <= 100
 G1 G91 X10 F500                           ;钻孔图
 Z-F100
 Z5
 ENDWHILE
 Z10
 ELSE                                      ;程序段寻找过程
 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
```

加工一个固定的零件数举例

```
%_N_STUECKZAHL_MPF

DEF INT STUECKZAHL
FOR STUECKZAHL = 0 TO 100
 G01 …
ENDFOR
M30
```

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。在这些程序段中不允许有标签。

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。

工作流程

1. **IF**

一个 IF-ELSE-ENDIF-程序段用来在2个选择之间进行选择：

IF (表达式)

NC程序段

ELSE

NC程序段

ENDIF

如果表达式的值为 TRUE，也就是说满足了条件，就执行下列程序块。如果条件不满足,ELSE分支被执行。

这个ELSE分支可取消。

1. **无限程序循环LOOP**

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。

LOOP

NC程序段

ENDLOOP

1. **计数循环 FOR**

当一个带有一个确定值的操作程序被循环重复，FOR循环就会被运行。记数变量同时会从初始值到最后值增加数值初始值必须小于终值。变量必须为 INT 型。

FOR 变量 = 初值TO 终值

NC程序段

ENDFOR

1. **在循环开始处带有条件的程序循环 WHILE**

只要条件满足，WHILE循环就被执行。

WHILE 表达式

NC程序段

ENDWHILE

1. **在循环结束处带有条件的程序循环 REPEAT**

REPEAT循环一旦被执行会不断重复,直到条件被满足为止。

REPEAT

NC程序段

UNTIL (表达式)

1.13 程序协调

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。这样，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生成有关,那么就要求同步程序操作过程。

对于程序协调有特殊的指令(命令)。它们总是在自己的程序段内。

注意

也可在自有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

-

举例：

版本SW3以前

INIT(2,"/_N_WKS_DIR/_ABRICHT_MPF")
G01 F0.1
START

在一个**初始**-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

在给出相对路径时，子程序调用的规则同样有效。

举例：

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	用来在某个通道中执行的指令
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探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探
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

注意

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。

通道名

通道名称必须通过变量（参见"变量和计算参数"一章）转换成数字。

小心

数字赋值应当在轻率的修改前被保存。

SETM() 和 CLEARM()

SETM() 和 CLEARM() 也可以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"

举例

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。

由此比如说指令显示START:

```
START(MASCHINE)
```

编程协调示例

通道 1:

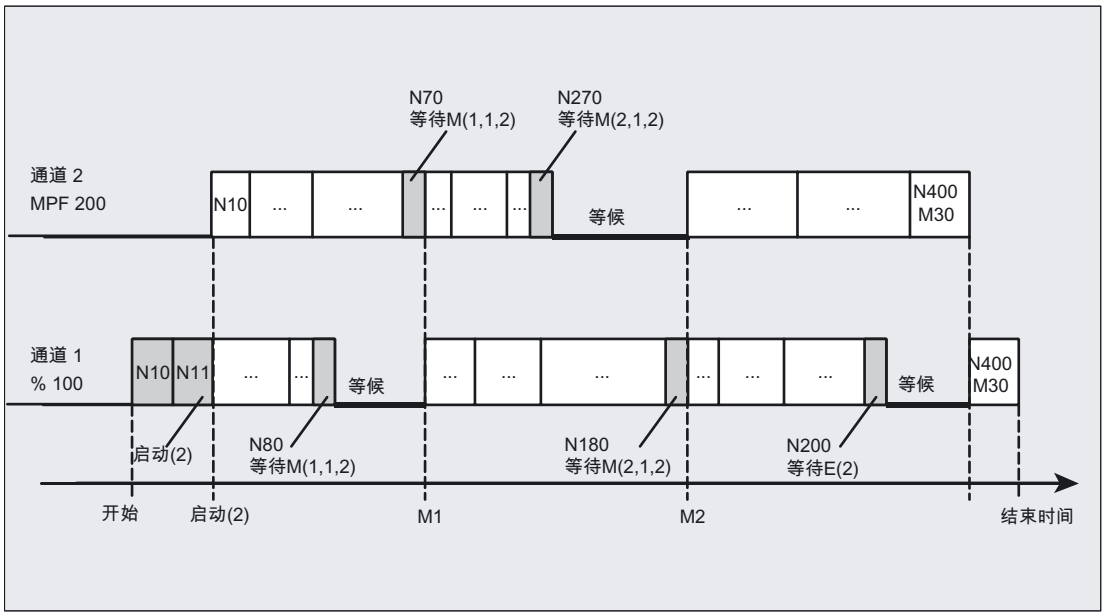
%_N_MPF100_MPF

```
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

```
;$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
```



来自工件的程序举例

```
N10 INIT(2,"/_N_WKS_DIR/_N_WELLE1_WPD/_N_ABSPAN1_MPF")
```

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 。
N10 INIT(2,"MYPROG") ; 在通道2中选择程序 /_N_MPF_DIR/_N_MYPROG_MPF
```

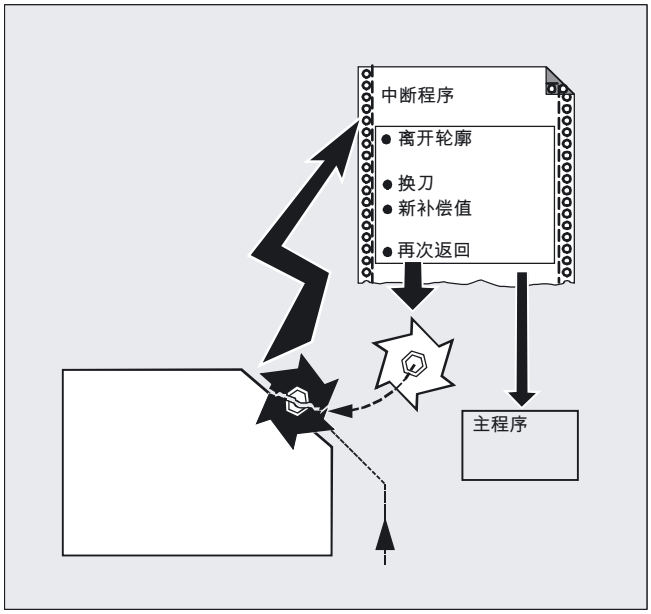
1.14 中断程序 (SETINT, DISABLE, ENABLE, CLRINT)

功能

依据某个典型示例阐述涉及某个中断程序编程的关系：

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。



有关 REPOS 的更多信息可在 "再次返回轮廓" 一章中查阅。

编程

```
SETINT(3) PRIO=1 NAME
SETINT(3) PRIO=1 LIFTFAST
SETINT(3) PRIO=1 NAME LIFTFAST
G… X… Y… ALF=…
DISABLE (3)
ENABLE (3)
CLRINT (3)
```

参数

SETINT(n)	启动中断程序，当输入端n接通时，n(1…8)是输入端序号。
PRIO=1	确定优先级别1到128(1最高)。
LIFTFAST	快速离开工件轮廓
NAME	这里是被执行的子程序的的名称
ALF=…	可编程的运动方向（在运动程序段中）
DISABLE(n)	切断中断程序，序号n
ENABLE(n)	再次接通中断程序，序号n
CLRINT(n)	取消中断程序序号n的中断赋值

后退运行

退回运动的方向通过G代码**LFTXT** 或者 **LFWP** 使用变量 **ALF** 进行编程。

- LFTXT
  从轨迹切线和刀具方向来确定退回运动的平面。通过这种G代码(标准设定)，编程快速离开时之前的性能。
- LFWP
  退回运动的平面是使用G代码 G17, G18 或者 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
  在用POLFMASK 标明的轴上回程到用 POLF 编程的绝对轴位置。也可参阅功能说明书 M3 中的NC控制的退回。
  ALF 对多个轴以及有线性关系的多个轴的抬起方向没有影响。

在退回平面中，与以往一样使用ALF 以45的不连续步骤对方向进行编程。在 LFTXT 时，回程在ALF=1的刀具方向中确定。

通过 LFWP ，工作平面的方向被分配如下：

- G17:X/Y-平面ALF=1沿着X方向退回
  ALF=3沿着Y方向退回
- G18:Z/X-平面ALF=1沿着Z方向退回
  ALF=3沿着X方向退回
- G19:Y/Z-平面ALF=1沿着Y方向退回
  ALF=3沿着Z方向退回

举例

在这个例子中，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

```
N10 SETINT(1) PRIO=1 W_WECHS ->
-> LIFTFAST

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
```

如果输入端1接通，刀具会立刻以快速离开(代码7对应工具半径补偿G41)的形式离开工件轮廓。然后中断程序W_WECHS被执行。

子程序

```
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                    ;重新返回轮廓并且跳回到
                                     ;主程序中
->在一个程序段中编程。
```

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。

生成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。

举例：

```
PROC ABHEB_Z
N10 …
N50 M17
```

程序名称 ABHEB_Z, 随后是NC程序段，最后是 `M17` 程序结束并且返回到主程序中。

注意

在中断程序内,SETINT指令可以被编程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。

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。

中断位置保存,SAVE

进行定义时，可使用 `SAVE` 来标识中断程序。

举例：

```
PROC ABHEB_Z SAVE
N10 …
N50 M17
```

通过SAVE定语,模态G功能会在中断程序结束后被调节到中断程序开始时的值。

此外对于可调节的零点偏置(模态G功能组8)，可编程的零点偏置和基准偏置再次被生成。当出现G功能组15（进给类型）的变化时，例如从 G94 变成 G95, 也会重新恢复相应的F值。

由此以后的处理可以从中断处继续进行。

中断程序赋值和开始,SETINT

控制系统支配信号

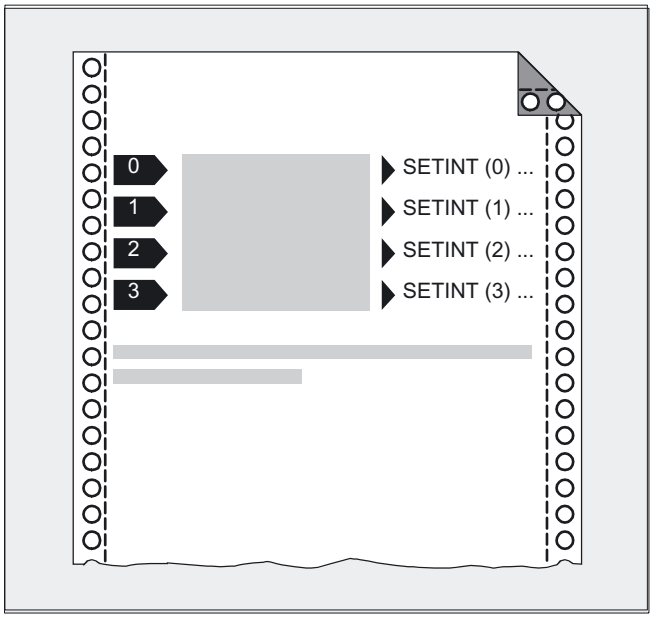
(输入端1…8)，它能引起正在进行的程序的中断和启动相应的中断程序。

哪一个输入端分配到哪一个程序，在执行程序中进行。

举例：

```
N10 SETINT(3) PRIO=1 ABHEB_Z
```

在接通输入端3时程序ABHEB_Z立刻被启动。



启动更多中断程序,确定级别,PRIO=

如果在您的NC-程序中有多个SETINT指令并且由此多个信号同时输入,您必须确定中断程序的级别,据此进行处理。优先级1到128,1最高。

举例：

```
N10 SETINT(3) PRIO=1 ABHEB_Z
N20 SETINT(2) PRIO=2 ABHEB_X
```

如果多个输入端同时保留,程序会根据级别数的顺序进行处理。首先 SETINT(3), 然后 SETINT(2).

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。

取消/重新启用中断程序, DISABLE, ENABLE

可以使用 DISABLE(n) 来取消NC程序中的中断程序，并且可用 ENABLE(n) 重新启用中断重新（n 表示输入编号）。

当 DISABLE 时保留输入/程序分配，并且可使用 ENABLE 重新激活。

中断程序重新赋值

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。

举例：

```
N20 SETINT(3) PRIO=2 ABHEB_Z
…
…
N120 SETINT(3) PRIO=1 ABHEB_X
```

取消赋值,CLRINT

使用 CLRINT(n) 可以删除分配。

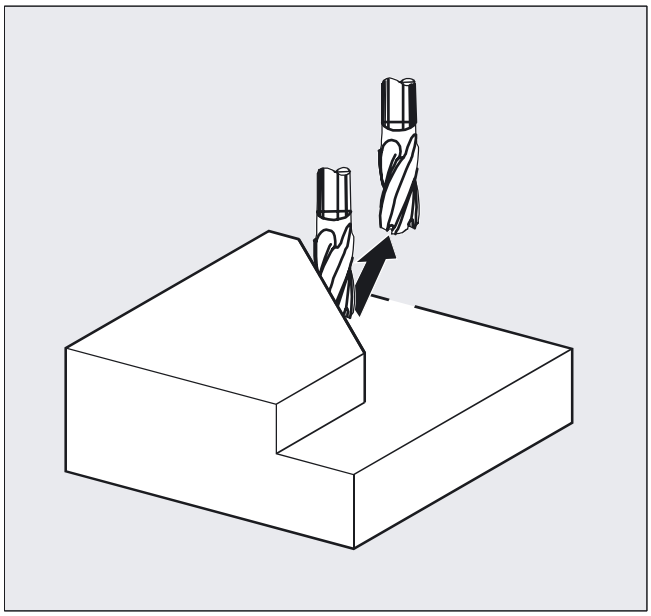
举例：

```
N20 SETINT(3) PRIO=2 ABHEB_Z
N50 CLRINT(3)
```

输入端3和程序ABHEB_Z之间的赋值被取消。

快速离开工件轮廓

使用 LIFTFAST 将在接通某个输入端时，通过快速从工件轮廓抬起的方式使刀具离开。



如果 SETINT-指令除了 LIFTFAST 之外还含有一个中断程序，就会在中断程序**之前**执行快速抬起动作。

举例：

```
N10 SETINT(2) PRIO=1 LIFTFAST
```

或者

```
N30 SETINT(2) PRIO=1 ABHEB_Z LIFTFAST
```

在两种情况中,在接通有最高优先权的输入端2时快速离开被执行。

- 如果是 N10，就会以报警 16010 来停止加工（因为没有指定异步子程序 ASUP）。
- 如果是N30，就执行 ASUP "ABHEB-Z"。

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。这种操作通过MD $MC_LIFTFAST_WITH_MIRROR=TRUE被激活

快速离开的运动过程

几何轴快速离开工件轮廓时所移动的距离，可以在机床数据中设定。

没有LIFTFAST的中断程序

一旦在轨道上的运行停止,就要在轨道上制动并启动中断程序。

该位置被保存为中断位置，并且当 REPOS 时，使用 RMI 在中断程序结束时向该位置逼近。

带LIFTFAST的中断程序

在轨道上刹车并同时进行FIFTFAST运动作为迭加运动。如果轨道运动和LIFTFAST运动停止,中断程序被启动。

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

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。

可编程的运行方向,ALF=...

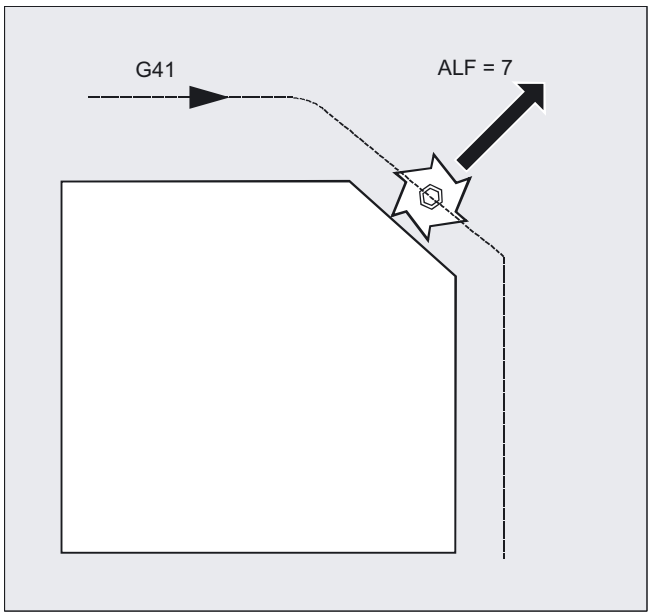
在NC程序中，给出工具快速离开时的方向。

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。

举例：

```
N10 SETINT(2) PRIO=1 ABHEB_Z LIFTFAST
ALF=7
```

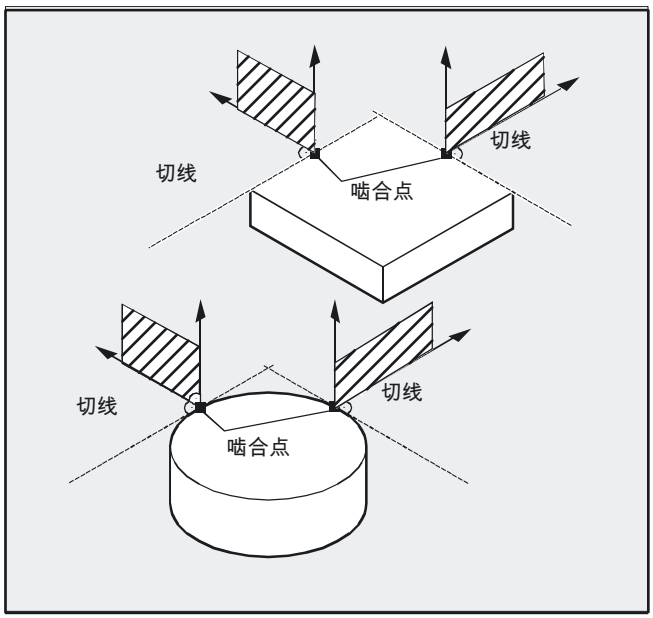
刀具在启用了 G41 的情况下（从轮廓左侧观察）以俯视垂直方向从轮廓上离开。



运行方向的基准面

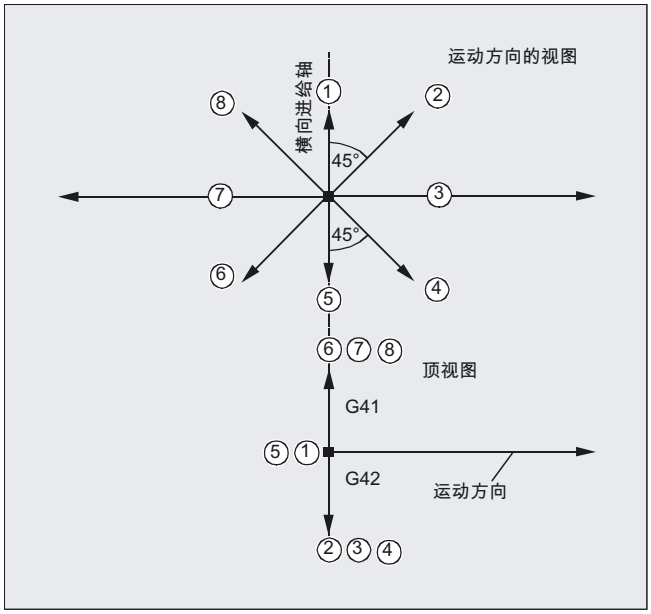
工具在编程的轮廓上的切入点有一个平面,它作为带相应代码离开运动的参数说明的基准面。

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。



代码序号概述，带运行方向

从这个基准面出发您可以在旁边的图象里找到带运行方向的代码序号。



使用 `ALF=0` 取消快速抬起功能。

小心

当启用了刀具半径补偿功能时，应当
在使用 G41 的情况下不使用编码2, 3, 4 并且
在使用 G42 的情况下
不 使用编码6，7，8。

在这些情况下工具驶向轮廓并会与工件相撞。

1.15 交换轴，交换主轴 (RELEASE, GET, GETD)

功能

一个或多个轴和主轴可以在一个通道中被使用。如果某个轴必须在两个不同的通道中以交替方式工作（例如集装架更换器），则必须首先在当前通道中将其释放，然后将其接受到另一个通道中：轴会在两个通道之间进行转换。

更多变换轴或主轴的功能的信息详见

/FB/, K5 BAGs, 通道，交换轴。

编程

RELEASE（轴名称，轴名称，...）或者 RELEASE (S1)

GET（轴名称，轴名称，...）或者 GET (S2)

或者

GETD（轴名称，轴名称 ...）或者 GETD (S3)

参数

RELEASE（轴名称，轴名称，…）	轴使能
GET（轴名称，轴名称，…）	轴接收
GETD（轴名称，轴名称，…）	轴直接接收
轴名称	在系统中轴赋值:AX1，AX2，… 或者给出加工轴名称
RELEASE (S1)	主轴S1,S2,匿
GET(S2)	主轴S1,S2,…接收
GETD(S3)	主轴S1,S2,直接接收...

举例

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应当在两个轴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"

```
%_N_MAIN_MPF
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
N …                                 ;交换轴之后的流程
N… M30
```

通道2中的程序 "交换2"

```
%_N_交换2_MPF
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
N …                                 ;交换轴之后的流程
N…M30
```

激活一个没有进给停止的轴交换举例

```
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
N041 G4 F2
N050 M5
N099 M30
```

主轴（轴B）直接在程序段N023之后成为**PLC轴** 例如向180度运动并且返回到1度且重新成为中性轴。则程序段N040没有引起进刀停止，并且无需重组。

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. 一个角逐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。

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

没有同步的交换

如果不必对轴进行同步，则通过GET不会产生进给停止。

举例：

```
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。
N09 …
```

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通道轴，则自动执行"GET"。如果这个(些)轴已经被同步,就不会产生进刀停止。

小心

即使在按键复位或者程序复位之后，某个使用GET接管的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。

轴直接接收:GETD

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）。

以可修改的方式设置轴交换属性

轴的交换时刻可通过 MD 10722:AXCHANGE_MASK 按照如下所述进行设置：

- 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。
- 所有使用 GET 或者 GETD 收入轴容器中的轴之后在轴容器旋转一次之后才能重新交换。
- 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。

1.16 NEWCONF:有效设置机床数据

功能

使用语言指令 NEWCONF 对有效等级"NEW_CONFIG" 的所有机床数据进行有效设置。
也可在操作界面HMI 中通过按下功能键“有效设置MD”的方式来激活该功能。

当执行功能 NEWCONF 时，会有隐式进给停止，即轨迹运动会被中断。

编程

```
NEWCONF
```

参数

NEWCONF	有效等级"NEW_CONFIG" 的所有机床数据被有效设置。

举例

铣削加工:用不同的工艺加工钻孔的位置。

```
N10 $MA_CONTOUR_TOL[AX]=1.0          ;更改机床数据
N20 NEWCONF                          ;有效设置机床数据
```

1.17 WRITE:写入文件

功能

使用 WRITE 指令可以将数据（例如执行测量循环时的测量结果）添加到指定文件的末尾。

所编制的文件可以：

- 由所有的使用者读、修改和删除，
- 写入到正在执行的零件程序中。

程序段在文件结束处插入，也就是说在M30之后。

当前所设置的保护级别必须等于或者大于文件的WRITE权限。如果不是这样，就会拒绝访问并且显示 错误=13

编程

WRITE(变量整数错误， 字符[160] 文件名， 字符[200] 字符串)

参数

机床制造商

用WRETE命令可以从零件程序中存放到文件中。记录文件(千字节)的大小在MD中确定。

通过 MD 11420:LEN_PROTOCOL_FILE 以 KByte 为单位来设置记录文件的最大长度。
这个长度对于所有用WRITE命令设定的文件都有效。

如果文件达到给定的长度,就会出现一个出错提示,字符串不会被保存。如果存储器够用,则可以编制一个新的文件。

WRITE	将数据添加到指定文件的末尾
错误	错误变量的返回
	0 没有错误
	1 路径非法
	2 找不到路径
	3 找不到文件
	4 错误的文件类型
	10 文件已满
	11 文件正被使用
	12 没有资源可用
	13 没有访问权限
	20 其它错误
filename	字符串写入的文件名称如果文件名称含有空格或者控制符（为十进制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"。 文件名可以用路径和文件标识给出。路径名称必须是绝对的，即以 "/" 开始。如果文件名不包含范围(_N_),它会被相应的填满。如果没有给出标记(_MPF 或 _SPF),就会被自动填上 _MPF。若给出的信息没有路径,文件会存放在当前的目录(选中程序的目录)中。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。 **举例：** PROTFILE _N_PROTFILE _N_PROTFILE_MPF /_N_MPF_DIR_/_N_PROTFILE_MPF/
字符串	正在写入的文本。内部还会附加上 LF，就是说文本会增加1个字符的长度。

注意

一个用WRETE命令描述的文件被重新编制,如果它不存在于NC中的话。

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。

消除方法：在操作面板下方通过功能键“属性”修改NC中的名称。

举例

```
N10 DEF INT ERROR
N20 WRITE(ERROR,"TEST1","PROTOKOLL VOM 7.2.97")          ;将文本从 PROTOKOLL VOM ;7.2.97 写入文件 TEST1 中
N30 IF ERROR
N40 MSG ("执行WRITE指令时出错:" <<ERROR)
N50 M0
N60 ENDIF
...
WRITE(ERROR,"/_N_WKS_DIR/_N_PROT_WPD/_N_PROT_MPF","PROTOKOLL VOM 7.2.97")          ;给出绝对路径
```

1.18 DELETE:删除文件

功能

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

编程

```
DELETE(var int 错误, char[160] 文件名)
```

参数

DELETE	删除所说明的文件。
错误	错误变量的返回
	0 没有错误

	1 路径非法 2 找不到路径 3 找不到文件 4 错误的文件类型 11 文件正被使用 12 没有资源可用 20 其它错误
文件名	应当被删除的文件名称。 文件名可以用路径和文件标识给出。路径名称必须是绝对的，即以 "/" 开始。如果文件名不包含范围(_N_),它会被相应的填满。文件标识符 ("_" 加上3个字符)， 例如 _SPF) 是可选项。如果实际上没有标识,文件名会自动被加上_MPF。若给出的信息没有路径,文件会存放在当前的目录(选中程序的目录)中。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。 **举例：** PROTFILE _N_PROTFILE _N_PROTFILE_MPF /_N_MPF_DIR/_N_PROTFILE_MPF/

举例

```
N10 DEF INT ERROR
N15 STOPRE                                          ;进刀停止
N20 DELETE (ERROR,                                  ;删除分支程序中的文件TEST1
 "/_N_SPF_DIR/_N_TEST1_SPF")
N30 IF ERROR
N40 MSG ("执行DELETE指令时出错:"
 <<ERROR)
N50 M0
N60 ENDIF
...
```

1.19 READ:读取文件中的行

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型字段中。每被读取的行都占用一个数组元。

当前所设置的保护级别必须等于或者大于文件的READ权限。如果不时这种情况,存取会被错误=13拒绝。

编程

READ(变量整数错误， 字符串[160]文件， 整数行,整数,变量字符串[255] 结果[])

参数

READ50	在指定文件中读取一个或者多个行并且保存在某个数组的数组元素中。 信息为字符串。
错误	错误变量的返回(参考调用-参数,类型INT)
	0 没有错误
	1 路径非法
	2 找不到路径
	3 找不到文件
	4 错误的文件类型
	13 访问级别不够
	21 行不存在(参数 "行" 或者 "数目" 大于文件行的数目)
	22 结果变量"结果" 的数组长度太小。
	23 行范围太大(参数 "文件" 选得太大，使得读取时超出了文件末尾)
数量	用于读取的文件的名称/路径(最大长度为160字节的STRING类型数值调用-参数)文件必须位于NCK的用户存储器中 (被动文件系统)。范围标识_N_可以反放在文件名前面。缺少的范围标识会被相应地填充。
	文件标识符 ("_" 加上三个字符，例如 _SPF) 为可选项。
	如果实际上没有标识,文件名会自动被加上_MPF。
	如果在"file"中没有指定路径，就在当前目录(= 选中程序的目录)查找文件。在"数量" 中已有的指定路径必须以一个 "/" 开始(绝对路径)。
行	需要读取的行范围的位置说明 (INT型的数值调用-参数)。
	0: 在文件末尾处读取用参数"文件" 说明的行个数。 1 ~ n: 第一个要读取行的编号。
文件	要读取的行的个数(INT类型的数值调用-参数)。
结果	字符串类型的数组，在这个数组中存放被读取的文本。 (长度为255字节的参考调用-参数)。

如果在参数“数目”中的行少于“结果”的数组长度，其余的数组元素不改变。

通过控制符 "LF" (换行) 或者 "CR LF" (托架折回换行) 表示的行结尾将不保存在目标变量 "结果" 中。如果被读取的行长于探不会出现错误提示。

注意

二进制文件不能被读入。

给出错误 错误=4:错误的文件类型。以下的文件类型不可读:_BIN, _EXE, _OBJ, _LIB, _BOT, _TRC, _ACC, _CYC, _NCK.

举例

```
N10 DEF INT ERROR                              ;错误变量
N20 STRING[255] RESULT[5]                      ;结果变量
...
N30 READ(ERROR, "TESTFILE", 1, 5,              ;没有范围标识和文件标识的文件名
 RESULT)
...
N30 READ (ERROR, "TESTFILE_MPF", 1, 5,         ;没有域标识符且有文件标识符的文明名称
 RESULT)
...
N30 READ(ERROR,"_N_TESTFILE_MPF",1,5,          ;带范围标识和文件标识的文件名
 RESULT)
...
N30 READ(ERROR,"/_N_CST_DIR/N_TESTFILE         ;有域标识符、文件标识符
 _MPF", 1, 5 RESULT)                           和路径说明的文件名称
^...
N40 IF ERROR <>0                               错误计值
N50 MSG("FEHLER "<<ERROR<<" BEI READ-
BEFEHL")
N60 M0
N70 ENDIF
...
```

1.20 ISFILE:存在于用户存储器 NCK 中的文件

功能

使用 ISFILE 指令可检查某个文件是否存在于 NCK 的用户存储器中 (被动文件系统) 。
作为结果会出现TRUE(存在文件)或者FALSE(不存在文件)。

编程

结果=is文件(字符串[160]文件)

参数

ISFILE	检查某个文件是否存在于NCK用户存储器中。
数量	用于读取的文件的名称/路径(最大长度为160字节的STRING类型数值调用-参数)。 文件必须位于NCK的用户存储器中 (被动文件系统)。范围标识_N_可以反放在文件名前面。缺少的范围标识会被相应地填充。 文件标识符 (" " 加上三个字符，例如 _SPF) 为可选项。如果实际上没有标识,文件名会自动被加上_MPF。 如果在揆在"数量" 中已有的指定路径必须以一个 "/" 开始(绝对路径)。
结果	用于接收BOOL类型结果的变量(TRUE或者FALSE)

举例

```
N10 DEF BOOL RESULT
N20 RESULT=ISFILE("TESTFILE")
N30 IF(RESULT==FALSE)
N40 MSG("DATEI NICHT VORHANDEN")
N50 M0
N60 ENDIF
...
或者
N30 IF(NOT ISFILE("TESTFILE"))
N40 MSG("DATEI NICHT VORHANDEN")
N50 M0
N60 ENDIF
...
```

1.21 CHECKSUM:通过某个数组构成校验和

功能

使用CHECKSUM，通过数组构成检查和。

应用：

检查输入轮廓在切削时是否已修改过。

编程

错误=CHECKSUM(变量字符串[16] 校验合， 字符串[32]数组， 首个整数， 最后整数)

参数

CHECKSUM	通过数组构成校验和
错误	显示返回的错误变量 0 没有错误 1 找不到符号 2 没有数组 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	通过数组表示的检查和，作为字符串(字符串类型的参考调用-参数,确定长度16)。 检查和作为16进制的符号串被显示。但是不说明格式符号。 举例："A6FC3404E534047C"
数组	数组名称，通过数组构成检查和。(长度最多为32的字符串类型数值调用-参数)。 容许的错误 BOOL, CHAR, INT, REAL, STRING 类型的一维或者二维数组 机床数据的数组不允许。
首个	开始栏目的栏目号(可选)
最后	结束栏目的栏目号(可选)

注意

参数首个和最后为可选项。如果没有说明栏目变址,检查和通过整个数组构成。

检查和的结果总是明确的。修改数组元时也会产生另一个结果字符串。

举例

```
N10 DEF INT ERROR
N20 DEF STRING[16] MY_CHECKSUM
N30 DEF INT MY_VAR[4,4]
N40 MY_VAR=...
N50 ERROR=CHECKSUM (CHECKSUM;"MY_VAR", 0, 2)
...

在MY_CHECKSUM中提供值：
"A6FC3404E534047C"
```

1.22 ROUNDUP:向上舍入

函数

函数 ROUNDUP 用来在

- **正输入值**
  时输出最接近的较大的整数
- **负输入值**
  时输出最接近的较小的整数

如果输入值为Integer型（整数型），则输出的数值保持不变。

编程

ROUNDUP(变量实数)

参数

ROUNDUP	用来向上舍入成最接近的较大的整数并冠以适当的符号。
变量	实数型的输入值

实数	用于带有小数点的小数的变量类型

NC零件程序中的 ROUNDUP

```
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])
```

举例

ROUNDUP(3.1) 得出 4.0
ROUNDUP(3.6) 得出 4.0
ROUNDUP(-3.1) 得出 -4.0
ROUNDUP(-3.6) 得出 -4.0
ROUNDUP(3.0) 得出 3.0
ROUNDUP(3) 得出 3.0

子程序技术，宏技术

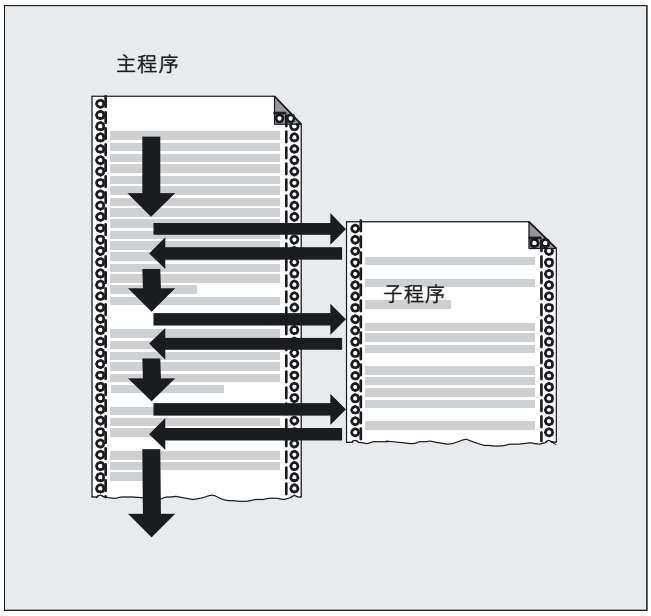
2

2.1 使用子程序

功能

原则上讲，一个子程序的结构与一个零件程序一样。它由带运行指令和开关指令的NC程序段组成。

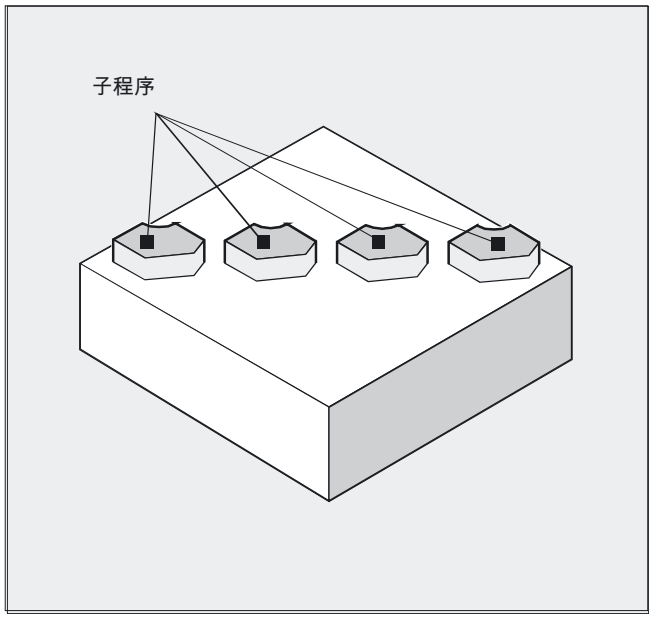
从本质上说，主程序与子程序没有区别。子程序中包含了要多次运行的工作过程或者工作步骤。



应用

总是反复出现的加工步骤在子程序中仅编程一次。比如说某个确定的轮廓，它们总是反复出现，或者是一个加工循环。

子程序可以在任意一个主程序中调用和执行。



子程序结构

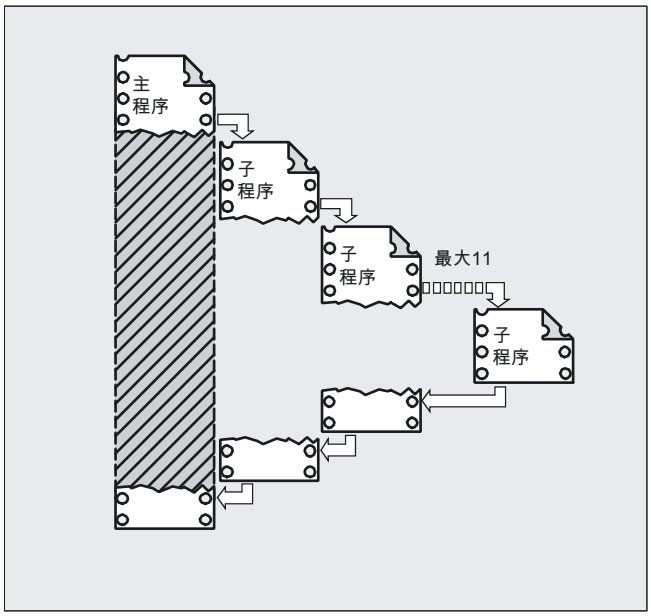
子程序结构与主程序结构相同。

此外，在子程序中可以编程一个程序头，带参数定义。

子程序的嵌套

在一个子程序中可以再次调用一个子程序。在该子程序中再调用一个子程序，依此类推。程序层的最大数量或者嵌套深度为12。

这表明：从一个主程序中可以调用11个嵌套的子程序。



中断程序和循环处理中子程序的限制条件

即使在中断程序中，也可以调用子程序。对于中断程序中的工作，您应该保留4个级面，或者仅仅嵌套7个子程序调用。

SIEMENS 的处理循环和测量循环需要三层。如果从一个子程序调用一个循环，则可以最多在级面5中进行（如果4个级面为中断程序保留）。

2.2 具有 SAVE 结构的子程序

功能

为此，对于带有 PROC 的定义语句，还要再设定指令 SAVE。

编程

在子程序中

PROC 子程序名称 SAVE

通过SAVE定语,模态G功能会在中断程序结束后被调节到中断程序开始时的值。如果因此而改变

G-函数组8（可设置的零点位移）

或者

G-函数组52（某个可旋转工件的框架旋转）

或者

G–函数组53（沿刀具方向中的框架旋转）

则相应的框架就会被恢复。

- 在子程序返回时，不会改变已激活的基本框架
- 可编程的零点位移被恢复

参数

可设置的零点位移和基本框架的特性可以通过机床数据MD10617：FRAME_SAVE_MASK来修改。

有关其它信息可参阅函数说明 /FB/ K1, 一般机床数据

举例

子程序定义

```
PROC KONTUR (REAL WERT1) SAVE
N10 G91 …
N100 M17
```

主程序

```
%123
N10 G0 X… Y… G90
N20…
N50 KONTUR (12.4)
N60 X… Y…
```

在子程序KONTUR中，G91增量尺寸生效。在返回到主程序中后，绝对尺寸再次生效，因为主程序的模态功能已经用SAVE存储。

2.3 带有参数传递功能的子程序 (PROC, VAR)

功能

子程序, PROC

一个子程序，它在调用程序的程序运行时应该接收参数，该子程序用关键字PROC标记。

子程序结束 M17, RET

用指令M17标记子程序结束，并且同时是返回到所调用主程序的指令。可替换M17的：关键字RET位于子程序结束处，没有中断轨迹控制运行，并且没有到PLC的功能输出。

编程

参数传送中重要的参数必须在子程序开始时说明名称和类型。

参数传送 数值调用

```
PROC PROGRAMMNAME(VARIABLENTYP1 VARIABLE1,VARIABLENTYP2 VARIABLE2,…)
```

举例：

```
PROC KONTUR(REAL LAENGE, REAL BREITE)
```

参数传递 参考调用, 使用关键字 VAR 标识

```
PROC PROGRAMMNAME(VAR VARIABLENTYP1 VARIABLE1,VAR VARIABLENTYP2 …,)
```

举例：

```
PROC KONTUR(VAR REAL LAENGE, VAR REAL BREITE)
```

传递数组 参考调用, 使用关键字 VAR 标识

```
PROC PROGRAMMNAME(VAR VARIABLENTYP1 FELDNAME1[Feldgröße],
VAR VARIABLENTYP2 FELDNAME2[数组大小],
VAR VARIABLENTYP3 FELDNAME3[数组大小1, 数组大小2],
VAR VARIABLENTYP4 FELDNAME4[ ],
VAR VARIABLENTYP5 FELDNAME5 [,数组大小])
```

举例：

```
PROC PALETTE(VAR INT FELD[,10])
```

参数

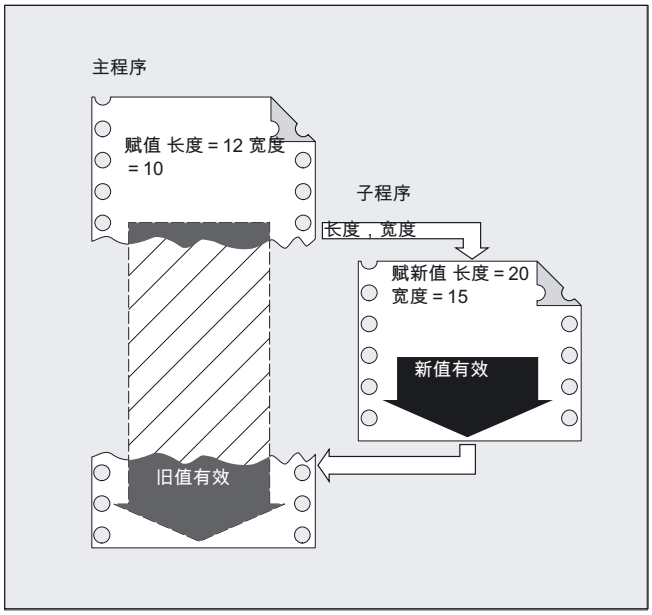
PROC	一个程序的第一个指令
PROGRAMMNAHME	应接收参数的重要值的子程序
VARIABLENTYP VARIABLE	带有变量值说明的变量类型 每次可以有多个说明。
VAR	用于参数传递类型的关键字
FELDNAME	某个数组的元素，定义数组时已赋值
数组大小1	用于1维数组
数组大小2	用于2维数组

注意

带有 PROC 的定义语句必须在一个自有NC程序段中编程。可以最多有127个参数用于参数传送。

主程序和子程序之间的参数传递举例

```
N10 DEF REAL LAENGE,BREITE
N20 LAENGE=12 BREITE=10
N30 RAHMEN(LAENGE,BREITE)
```



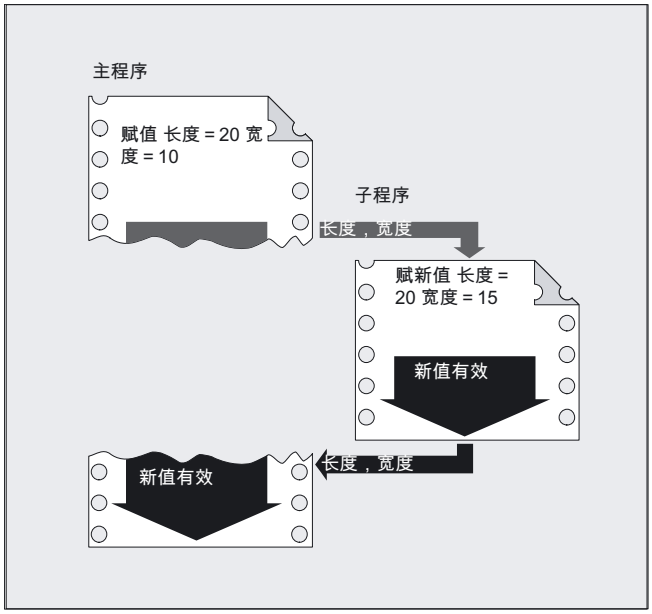
在主程序N20中赋值的数值传递到子程序N30中。参数传送按照所给定的顺序进行。

参数名称在主程序和子程序中不可一样。

参数传递有两种方法：

- **仅传送的值(数值调用)**

如果被传送的参数在子程序执行过程中被修改，这对主程序没有影响。这里参数保持不变（见图）。



- **带有数据交换的参数传递 (参考调用)**

在子程序中参数每次改变均同时影响主程序中该参数的改变（见图）。

数组长度变量举例

```
%_N_BOHRPLATTE_MPF                          主程序
DEF REAL TABELLE[100,2]                     ;定义位置表
EXTERN BOHRBILD (VAR REAL[,2],INT)
TABELLE[0,0]=-17.5                          ;确定位置
…
TABELLE[99.1]=45
BOHRBILD(TABELLE,100)                       ；子程序调用，
M30
```

根据一个已传送的可变长度位置表建立一个钻孔图举例

```
%_N_BOHRBILD_SPF                            子程序
PROC BOHRBILD(VAR REAL FELD[,2],->          ;参数传递
-> INT ANZAHL)
DEF INT ZAEHLER
STEP:G1 X=FELD[ZAEHLER,0]->                 ;加工顺序
-> Y=FELD[ZAEHLER,1] F100
Z=IC(-5)
Z=IC(5)
ZAEHLER=ZAEHLER+1
IF ZAEHLER<ANZAHL GOTOB STEP
RET                                         ;子程序结束
```

没有中断的轨迹控制运行

轨迹控制运行不被中断的前提条件是：

子程序**不得** 具有SAVE属性。有关SAVE机制的其它说明可参阅具有SAVE结构的子程序一章。

RET 必须在自有NC程序段中编程。

```
PROC KONTUR
N10…
…
N100 M17
```

主程序和子程序之间的参数传递

如果在主程序中带参数工作，则您也可以在子程序中使用相应计算的或者赋值的数值。在调用子程序时，主程序**当前参数**的值被传送给子程序的**形式参数** ，然后在执行子程序的过程中进行处理。

数组定义

对于形式参数的定义，适用于：如果是二维数组，则不必说明第一维的数组数量，但是逗号必须写。

举例：

```
VAR REAL FELD[,5]
```

子程序可以用不确定的数组长度加工处理可变长度的数组。当定义变量时，仍然必须确定要接收多少个元素。关于数组定义的说明可查阅数组定义同名章节中的“灵活NC编程”一项。

2.4 调用子程序 (L 以及 EXTERN)

功能

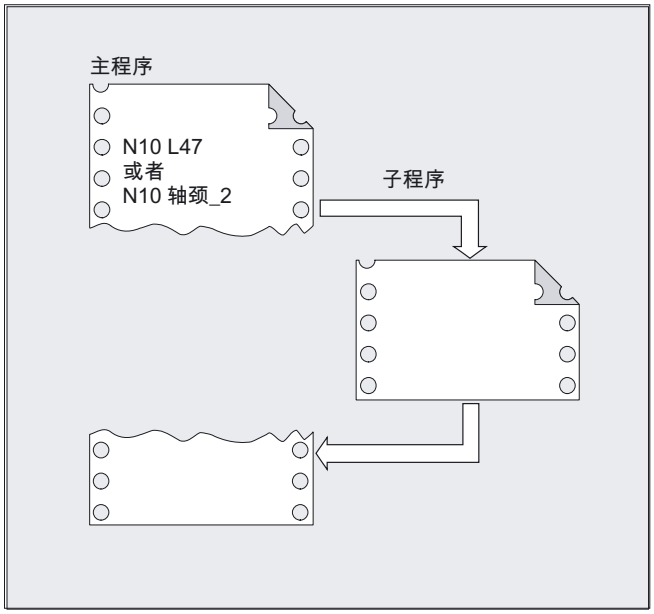
没有参数传递的子程序调用

在主程序中调用子程序时，要么

- 使用地址L和子程序编号，或者
- 通过指定程序名称来调用。

举例：

```
N10 L47 或者
N10 ZAPFEN_2
```



编程

带有参数传递的子程序，使用 EXTERN 说明

EXTERN

必须在调用之前在主程序中使用 EXTERN
对带有参数传递的子程序进行说明，例如在程序开始处。

说明子程序的名称和传递顺序中的变量类型，参见举例。

子程序，带参数转让

在主程序中，通过说明程序名和参数传送调用子程序。使用参数传递功能可以直接（当不是VAR参数时）发送变量或者值，参见举例。

参数

地址 L	子程序号
EXTERN	使用参数说明对某个子程序进行声明。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 。循环不必作为 EXTERN 解释。

不完整的参数传送

在子程序调用时，可以删除自身规定的值或者参数。在这种情况下，相应的参数在主程序中被赋零。

说明顺序时必须写入逗号。如果参数位于顺序的结束处，则可以同样取消该逗号。

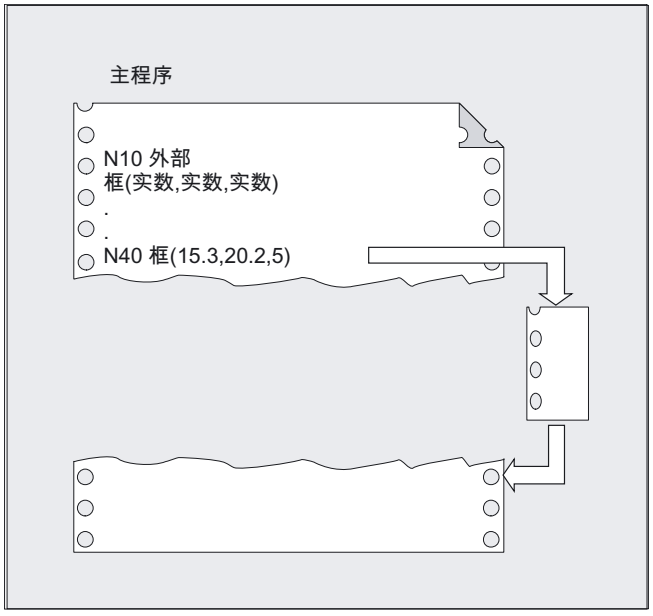
小心

类型AXIS的当前参数不允许删除。必须完整传送VAR参数。

带有参数传递的子程序举例，使用 EXTERN 说明

```
N10 EXTERN RAHMEN(REAL, REAL, REAL)
…
N40 RAHMEN(15.3,20.2,5)
```

N10 说明子程序，N40 调用带有参数传递的子程序。

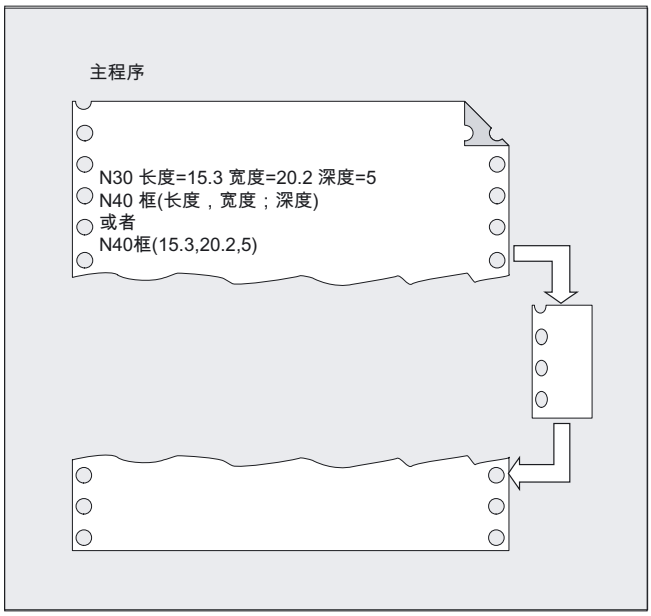


调用带有参数传递的子程序举例

```
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)
```



子程序举例

```
PROC SUB1 (INT VAR1, DOUBLE VAR2)
IF $P_SUBPAR[1]==TRUE
;参数 VAR1 已编程在子程序调用中
ELSE
;参数 VAR1 没有编程在子程序调用中
;且被系统预先赋予默认值0
ENDIF
IF $P_SUBPAR[2]==TRUE
;参数 VAR2 已编程在子程序调用中
ELSE
;参数 VAR2 没有编程在子程序调用中
;且被系统预先赋予默认值0.0
ENDIF
；参数3没有定义
IF $P_SUBPAR[3]==TRUE -> 报警 17020
M17
```

说明

小心

子程序定义相当于子程序调用

不管是变量类型还是传递的顺序，均必和主程序中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。

在子程序中定义：

```
PROC RAHMEN(REAL LAENGE, REAL BREITE, REAL TIEFE)
```

在主程序中调用：

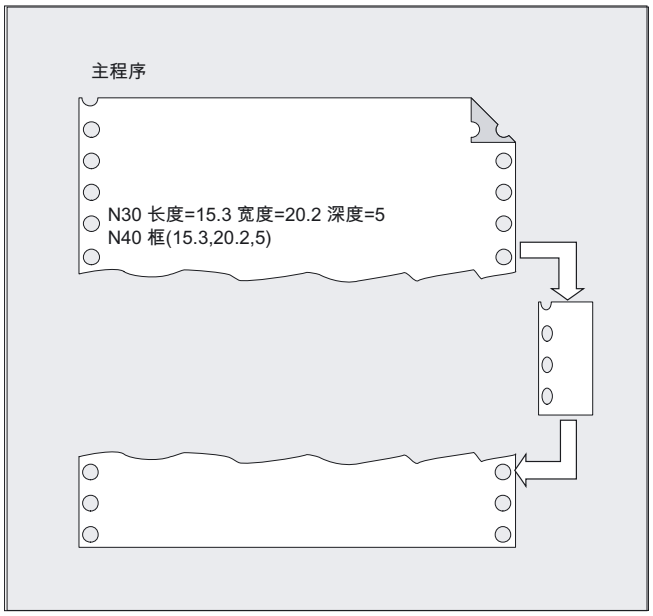
```
N30 RAHMEN(LAENGE, BREITE, TIEFE)
```

不完整的参数传送

返回到最后的示例：

```
N40 RAHMEN(15.3, ,5)
```

这里删除均值20.2。



如果参数传递不完整，可通过系统变量 `$P_SUBPAR[i]` 来识别是否已对子程序的传递参数进行了编程。作为自变量(i)，系统变量获得传送参数的号。

系统变量 `$P_SUBPAR` 发送

- TRUE, 表示传递参数已经编程
- FALSE, 表示没有参数被用作传递参数。

如果说明了一个不允许的参数号，则零件程序加工中断，并发出报警。

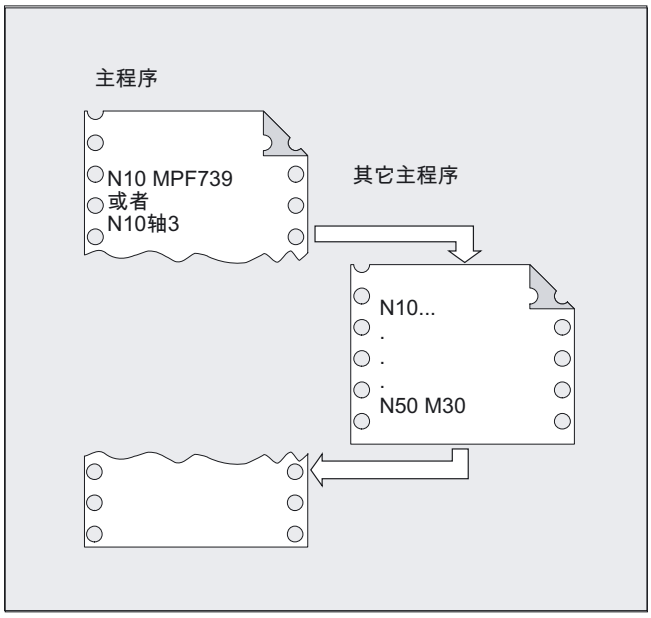
调用主程序作为子程序

一个主程序也可以作为子程序调用。在主程序中设置的程序结束M2或者M30在这种情况下如同M17（程序结束，返回到所调用的程序）处理。

通过给出程序名称编程此调用。

举例：

```
N10 MPF739 或者
N20 Welle3
```



注意

相应地也可以把一个子程序作为主程序启动。

2.5 可设定参数的子程序返回 (RET)

功能

通常情况下，从一个带程序结束RET或者M17的子程序返回到所调用的程序，并且零件程序的加工以在子程序调用之后的程序行继续。但是也有其它的应用情况，在此希望程序加工在另一处继续：

- 在ISO-语言-方式下调用切削循环之后，根据轮廓描述继续程序加工。
- 在任意一个子程序级面（也在ASUP之后），在故障处理时返回到主程序。
- 返回经过几个程序级面，用于在编译循环和ISO-语言-方式中的特殊应用。

编程

RET （<程序段号/标签>，<按程序段号/标签逐段执行>，
<返回层数>），<返回到程序开始处>）

或者

RET （<程序段号/标签>，< >，< >）

或者子程序通过多个层返回

（按照规定的子程序层数返回）

RET （，，<返回层数>，<返回到程序开始处>）

参数

使用可设定参数的指令 RET 可以利用四个参数满足继续执行或者返回的请求：

1. **<程序段号/标签>**
2. **<在带程序段号/标签程序段之后的程序段>**
3. **<返回级面个数>**
4. **<跳回到程序开始>**,

RET	子程序结束 （替代M17应用）
<程序段号/标签>	参数：程序段的程序段号或者作为字符串的标签（常量或者变量），程序加工应在此程序段处继续进行。 使用带有"程序段号/标签"的程序段在调用程序中继续进行程序处理。
<在带程序段号/标签程序段之后的程序段>，	类型INTEGER的参数 如果**值大于0，** 就使用"程序段号/标签"后的下一个程序段继续执行。如果**值等于0，** 子程序就返回到带有<程序段号/标签>的程序段。
<返回级面个数>，	带有允许**值1～11**的INTEGER型参数。 值 = 1:程序就在当前程序层中继续执行（如同没有参数的 RET ）。 值 = 2:程序在当前程序层中继续执行，同时越过某个层等等。
<跳回到程序开始>，	参数，BOOL类型 **值1或者0.** 值 = 1 当返回到主程序中且主程序中已有一个ISO-语言模式激活时，就分支到程序开始处。

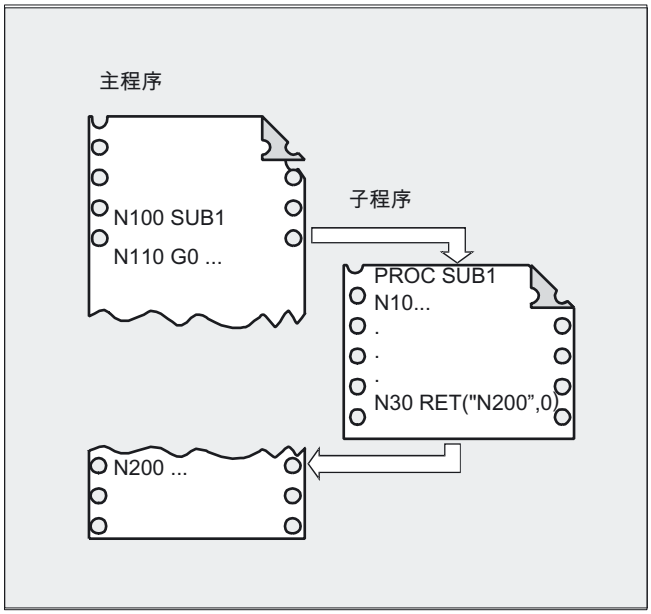
故障处理举例：在ASUP加工之后，在主程序中继续。

```
N10010 CALL "UP1"                          ; 程序级面0，主程序
  N11000 PROC UP1                          ; 程序级面1
  N11010 CALL "UP2
    N12000 PROC UP2                        ; 程序级面2
    N19000 PROC ASUP                       ; 程序级面2（ASUP加工）
      ... RET("N10900", , ...              ; 程序级面3
      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
```

说明

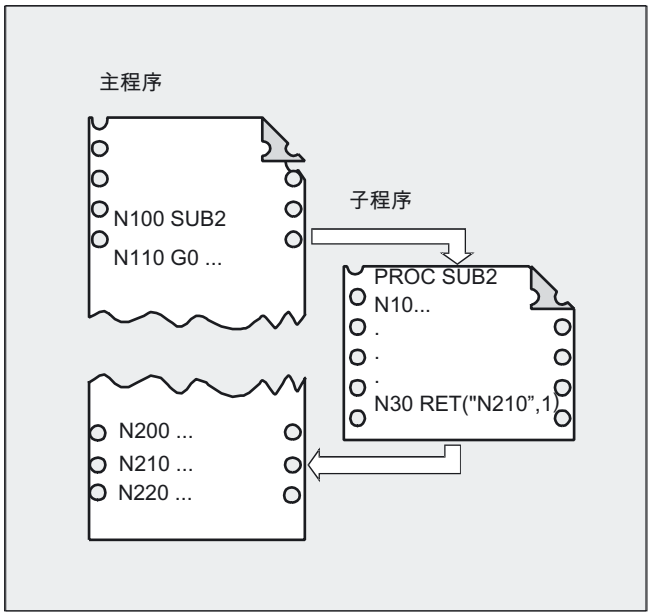
1.<程序段号/标签>

在所调用的程序中（主程序）用带“程序段号/标签”的程序段继续。



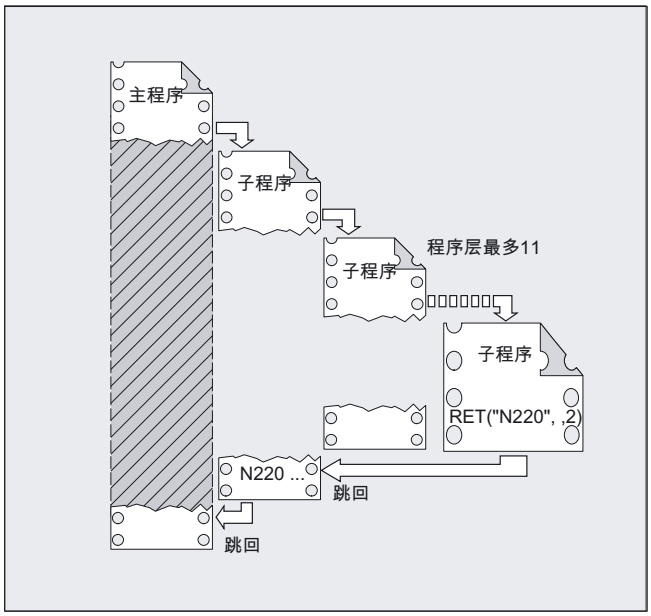
2.<在带程序段号/标签程序段之后的程序段>,

在带<程序段号/标签>的程序段之后的程序段中进行子程序返回。



3.<返回级面个数>

程序在当前程序层中减去 <返回层数> 继续执行。



不允许的返回级面

如果给返回级面的个数

- 编程一个负值，或者
- 某个值大于当前活动的程序层 –(最大11)

则用参数5给出报警14091。

返回，带SAVE指令

在跳回几个程序级面时，计算各个程序级面的SAVE指令。

在返回时模态子程序有效

如果在通过多个程序层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

用户必须**始终自行确保**在通过多个程序层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。

2.6 带有程序重复执行功能的子程序（P）

功能

如果要求连续多次执行一个子程序，则可以在程序段中调用子程序时，在地址P下编程程序重复的次数。

参数

小心

使用程序重复执行和参数传递功能调用子程序

参数仅在程序调用时或者第一次执行时传送。对于其它的重复，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。

P	子程序的执行次数
值范围：	1…9999 （整数型，没有前置符）

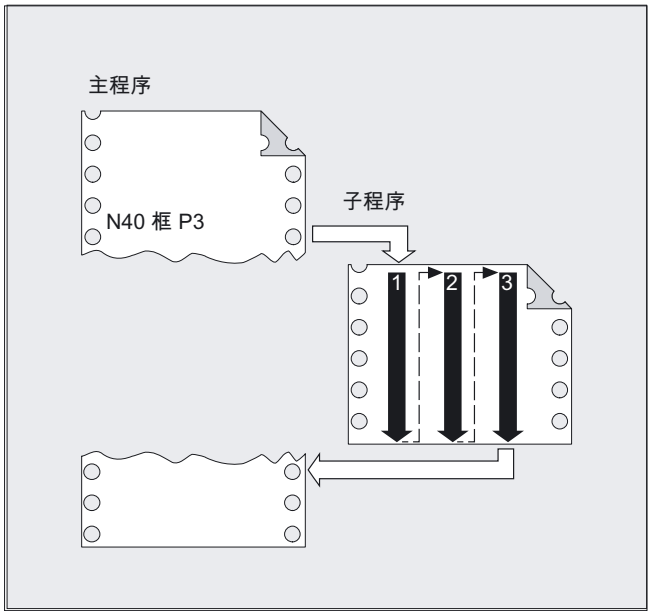
小心

对于每个子程序调用，要求：

子程序调用必须在独立的NC程序段中编程。

举例

```
N40 RAHMEN P3
```



子程序框架应被先后执行三次。

2.7 模态子程序 (MCALL)

功能

用此功能，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。

关闭模态子程序调用

使用MCALL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。

参数

MCALL	模态子程序调用
地址 L	子程序号

小心

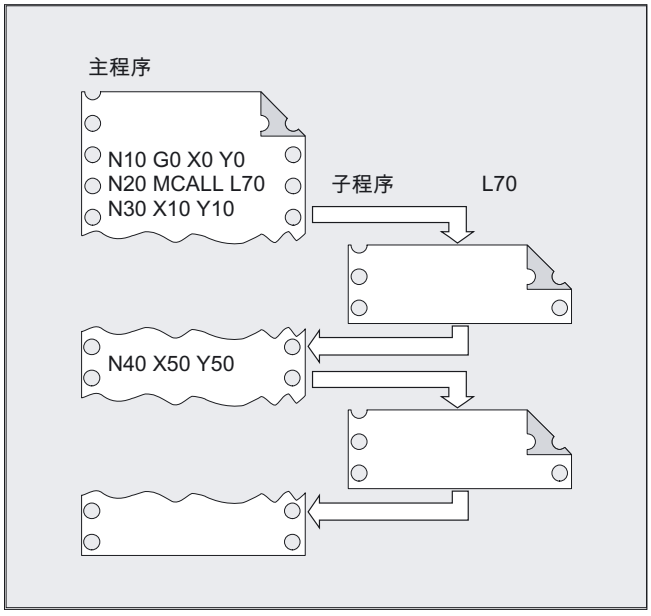
在某个程序执行过程中，同时**只能有一个** MCALL调用有效。在MCALL调用中仅传送一次参数。模态子程序在下面的情况下也可以不编程一个运动而调用：当G0或G1有效时，编程地址S和F。G0/G1单独编程在程序段中，或者与其它的G指令一起编程。

举例

```
N10 G0 X0 Y0
N20 MCALL L70
N30 X10 Y10
N40 X50 Y50
```

在程序段N30和N40中返回编程的位置，并接着执行子程序L70。

```
N10 G0 X0 Y0
N20 MCALL L70
N30 L80
```



在这个例子中，在子程序L80中有以下带编程轨迹轴的NC程序段。L70通过L80调用。

2.8 间接调用子程序 (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。

编程

CALL <程序名>

参数

CALL	间接调用子程序的关键字
<程序名>	变量或者字符串类型常量 程序名，包含待执行程序部分。

小心

间接调用子程序仅可以用于没有参数传送的子程序。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

举例

使用字符串常量直接调用

```
CALL "/_N_WKS_DIR/_N_SUBPROG_WPD/_N_TEIL1_SPF"
```

通过变量间接调用

```
DEF STRING[100] PROGNAME
PROGNAME="/_N_WKS_DIR/_N_SUBPROG_WPD/_N_TEIL1_SPF"
CALL PROGNAME
```

子程序程序段1被分配给变量PROGNAME 。使用CALL和路径说明，间接调用子程序。

2.9 使用间接编程重复执行程序段 (CALL)

功能

使用CALL可以间接调用子程序，在该子程序中用BLOCK定义的程序部分重复按照起始标签和结束标签进行。

编程

CALL <程序名> BLOCK <起始标签> TO <结束标签>

CALL BLOCK <起始标签> TO <结束标签>

参数

CALL	间接调用子程序的关键字
<程序名>（选件）	变量或者字符串类型常量，程序名，它包含待加工的程序部分。 如果没有编程 <程序名>，则在当前的程序中查找带<起始标签> <结束标签> 的程序部分，并且执行。
BLOCK ...TO ...	关键字用于 间接程序部分重复
<起始标签> <结束标签>	变量或者字符串类型常量 参阅待加工程序部分的开始处或者结束处。

举例

```
DEF STRING[20] STARTLABEL, ENDELABEL
STARTLABEL = "LABEL_1"
ENDELABEL = "LABEL_2"
...
CALL "CONTUR_1" BLOCK STARTLABEL TO ENDELABEL ...
M17
PROC CONTUR_1 ...
LABEL_1                                          ;程序部分重复开始
N1000 G1 ...
LABEL_2                                          ;程序部分重复结束
```

2.10 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 模式的其它信息可参阅 /FBFA/, "函数说明 ISO方言"

编程

ISOCALL <程序名>

参数

ISOCALL	子程序调用，由此激活机床数据中设定的ISO-模式。
<程序名>	变量或者字符串类型常量 以某个ISO语言编写的程序的名称

举例使用ISO模式的循环编程调用轮廓

```
%_N_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
N2300 ...                                     ;以ISO模式编辑程序0122.spf
N2400 M30
```

2.11 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。

编程

```
PCALL <路径/程序名>(参数 1, …, 参数 n)
```

参数

PCALL	关键字，用于带绝对路径说明的子程序调用
<路径名>	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，就必须明确使用外部前缀和后缀加以说明。
参数1到n	实际参数符合子程序的PROC指令。

举例

```
PCALL/_N_WKS_DIR/_N_WELLE_WPD/WELLE(参数1, 参数2, …)
```

2.12 使用CALLPATH扩展调用子程序时的路径查找

功能

使用指令CALLPATH可以扩展查找路径用于子程序调用。这样也可以从某个未选中的工件目录中调用子程序，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).

撤销选择查找路径扩展

查找路径扩展通过以下的事件撤销选择：

- CALLPATH 带空字符串
- CALLPATH 没有参数

- 零件程序结束
- 复位

编程

将保存在NCK的现有文件系统之外的子程序补充到现有NCK文件系统中。

CALLPATH <路径名>

参数

CALLPATH	关键字，用于可编程的查找路径扩展。指令CALLPATH在一个自身的零件程序行中编程。
<路径名>	字符串型常量或者变量。包含某个目录的绝对路径说明，以"/"开始，以此来扩展查找路径。路径必须使用前缀和后缀进行完整说明（例如：/_N_WKS_DIR/_N_WST_WPD）. 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。最大的路径长度达到128个字节。

注意

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。

举例

```
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/子程序标识符_SPF
8. /_N_CST_DIR/子程序标识符_SPF

注意

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). 随后将初始化程序重新复位。

2.13 抑制当前的程序段显示 (DISPLOF)

功能

使用 DISPLOF 可抑制子程序的当前程序段显示DISPLOF 位于PROC指令的结束处。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。

正常情况下打开程序段显示。用DISPLOF关闭程序段显示，直至从子程序返回或者程序结束。

编程

```
PROC … DISPLOF
```

如果从带DISPLOF属性的子程序中调用其它的子程序，则在这个子程序中也抑制当前的程序段显示。如果一个子程序带抑制的程序段显示，由一个异步的子程序中断，则当前子程序的程序段被显示。

参数

DISPLOF	抑制当前的程序段显示

在循环中抑制当前程序段显示举例

```
%_N_CYCLE_SPF
;$PATH=/_N_CUS_DIR
PROC CYCLE (AXIS TOMOV, REAL POSITION) SAVE DISPLOF
                                                    ;抑制当前的程序段显示
                                                    ;现在将循环的调用显示为当前程序段

                                                    ;例如：CYCLE(X, 100.0)
DEF REAL DIFF                                       ;循环内容

G01 …
…
```

```
RET                                    ;子程序返回，重新显示调用程序后面的程序段
```

2.14 单程序段抑制 (SBLOF, SBLON)

功能

程序专用的单段抑制

当遇到单程序段类型时，如同一个程序段一样完整执行带有SBLOF标识的程序。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。

使用M17返回:停止于子程序末尾处

使用RET返回:在子程序末尾处不停止

SBLOF也适用于所调用的子程序。

单程序段中没有停止的子程序举例：

```
PROC BEISPIEL SBLOF
G1 X10
RET
```

编程

```
PROC ...SBLOF          ；该指令可以位于一个PROC程序段中，或者单独位于程序段中
SBLON                  ;该指令必须在一个自有程序段中
```

在程序中单段抑制

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
- 有效子程序级面的结束。

参数

SBLOF	关闭单段
SBLON	再次接通单段

程序中的单程序段抑制举例

```
N10 G1 X100 F1000
N20 SBLOF
N30 Y20                                          ;关闭单程序段
N40 M100
N50 R10=90
N60 SBLON
N70 M110
N80 ...                                          ;再次启用单程序段
```

N20和N60之间的区域，在单段运行时作为一步处理。

示例循环对于用户而言就如同一个指令

主程序

```
N10 G1 X10 G90 F200
N20 X-4 Y6
N30 CYCLE1
N40 G1 X0
N50 M30
程序循环:1
N100 PROC CYCLE1 DISPLOF SBLOF                   ；抑制单段
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，必须按一次起动按钮。

举例，为激活已修改的零点位移和刀具补偿而被PLC所起动的ASUP应该不可见。

```
N100 PROC NV SBLOF DISPLOF
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
```

举例，使用 MD 10702 IGNORE_SINGLEBLOCK_MASK, Bit 12 = 1 不会停止

在单程序段类型中 SBL2 (在每个零件程序行中停止) 在SBLON-语句中：

```
；SBL2有效
;$MN_IGNORE_SINGLEBLOCK_MASK = 'H1000'        ;在MD 10702中:设定Bit 12 = 1
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
 PROC CYCLE SBLOF                             ；抑制单段停止
 N100 R0 = 1
 N110 SBLON                                   ;因为 MD 10702:Bit 12 = 1 所以不会
                                              ;停止
 N120 X1                                      ;在该零件程序行中停止
 N140 SBLOF
 N150 R0 = 2
 RET
N50 G90 X20                                   ;在该零件程序行中停止
M30
```

程序嵌套时抑制单程序段举例

```
                                              ；单段有效
N10 X0 F1000                                  ;在该程序段中被停止
N20 UP1(0)
     PROC UP1(INT _NR) SBLOF                  ；单段"关"
     N100 X10
     N110 UP2(0)
          PROC UP2(INT _NR)
          N200 X20
          N210 SBLON                          ；单段"开"
          N220 X22                            ;在该程序段中被停止
          N230 UP3(0)
               PROC UP3(INT _NR)
               N302 SBLOF                     ；单段"关"
               N300 X30
               N310 SBLON                     ；单段"开"
```

```
                N320 X32                    ;在该程序段中被停止
                N330 SBLOF                  ；单段"关"
                N340 X34
                N350 M17                    ；SBLOF 有效
          N240 X24                          ;在该程序段中被停止，
                                            ;SBLON 激活
          N250 M17                          ;在该程序段中被停止，
                                            ;SBLON 激活
     N120 X12
     N130 M17                               ;在该返回程序段中被
                                            ;停止，
                                            ;PRO语句的SBLOF激活
N30 X0                                      ;在该程序段中被停止
N40 M30                                     ;在该程序段中被停止
```

边界条件

- 可以在循环中使用DISPLOF抑制当前的程序段显示。
- 如果DISPLOF连同SBLOF一起编程，则在循环之内在单段停止时，如同在调用循环之前一样显示。
- 如果在系统ASUP或者用户ASUP中，单段停止用位0 = 1或者位1 = 1（机床数据MD10702：IGNORE_SINGLEBLOCK_MASK 被抑制，可以通过在ASUP中编程 SBLON 的方式来重新激活单程序段停止。
- 用户ASUP中的单程序段停止可使用 MD 20117:IGNORE_SINGLEBLOCK_ASUP 进行抑制，并且可以通过编程SBLON 不再激活。
- 通过选择SBL3，抑制指令SBLOF。
- 忽略单程序段类型2中的单程序段停止。在单程序段类型2中 (SBL2)，当Bit12 = 1 被 MD 10702: IGNORE_SINGLEBLOCK_MASK 设定时，则在SBLON-程序段中**不** 停止。

注意

关于有/没有单程序段抑制的程序段显示的其它信息可参阅 /FB/, K1 BAG, 通道，程序运行 “单程序段”。

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语句。这也适用于通过MD 11610:ASUP_EDITABLE 实现的功能“可编辑系统ASUP”。

“可编辑系统ASUP”举例：

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

在单段中的程序影响

在单段功能中，用户可以按程序段方式执行零件程序。单段有以下的设定方式：

- 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SBL3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指令中，则用M17停止到子程序返回。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，则在子程序中在返回之前（M17）必须再次编程SBLON。在一个上一级的程序中，在用RET返回时，不停止。

2.15 执行外部子程序 (EXTCALL)

功能

使用 EXTCALL 可以从HMI在“从外部执行”模式中加载一个程序。在此所有通过HMI的目录结构可以到达的程序可以后装载并执行。

编程

EXTCALL (<路径/程序名>)

参数

EXTCALL\	用于子程序调用的关键字
<路径/程序名>	字符串型常量/变量 可以说明一个绝对的路径名或者一个程序名。 使用/不使用前缀 N 和不使用后缀说明程序名称。一个扩展名可以用符号<_>添加到程序名中。

举例：

```
EXTCALL ("/_N_WKS_DIR/_N_WELLE_WPD/_N_WELLE_SPF") 或者 EXTCALL ("WELLE")
```

注意

外部子程序不得含有跳转语句，如 GOTOF, GOTOB, CASE, FOR, LOOP, WHILE 或者 REPEAT

可以有IF-ELSE-ENDIF常量

子程序调用可以嵌套 EXTCALL调用。

举例 HMI 高级

需要加载的程序位于 **HMI 高级 的本地硬盘**上。

在调整数据SD 42700:EXT_PROG_PATH
中保存有下列路径："/_N_WKS_DIR/_N_WST1".主程序_N_MAIN_MPF在工作存储器中，并且已经选择。

```
N10 PROC MAIN
N20 ...
N30 EXTCALL "粗加工"                              ;调用外部子程序
                                                  ;SCHRUPPEN
N40 ...
N50 M30
子程序 "SCHRUPPEN" (位于HMI 高级的目录结构中， 在工件 ->WST1项下):
N10 PROC SCHRUPPEN
N20 G1 F1000
N30 X=... Y=... Z=...
N40 ...
N90 M17
```

举例 HMI 内置

需要加载的程序位于**网络驱动器或者HMI的ATA卡**上。

EXTCALL Windows-路径说明

调用**ATA卡** (HMI 内置) 例如
EXTCALL C:\工件\轮廓2.spf

注意

如果是HMI 内置，始终必须指定一个绝对路径。

HMI 高级 或者 HMI 内置 举例

调用**网络驱动器** (HMI 内置 或者 高级)
EXTCALL \\R4711\工件\轮廓1.spf

有关操作的其它信息可参阅

/BEM/ 内置HMI

/BAD/ 高级HMI

一个外部程序路径的说明

通过 SD 42700:EXT_PROG_PATH 可以灵活设置调用路径。SD 42700 包含一个路径说明，与所编程的子程序标识符共同构成被调用程序的绝对路径名称。

调用一个外部子程序

通过零件程序指令**EXTCALL**调用一个外部子程序。
从

- 用EXTCALL编程的子程序和
- 设定数据SD42700:EXT_PROG_PATH
  得出外部子程序调用的程序路径，方法是将符号连写，即
- SD 42700的内容:EXT_PROG_PATH (例如 /_N_WKS_DIR/_N_WKST1_WPD)
- 分隔符"/"
  (当已使用 SD 42700:EXT_PROG_PATH 设定一个路径时)
- 在EXTCALL所说明的子程序路径或者子程序名称

SD 42700:EXT_PROG_PATH 被赋予一个空格符。如果调用外部子程序，不作绝对路径说明，则在高级HMI中运行同样的查找路径，如同从NCK存储器中调用子程序时一样：

1. 当前的目录/子程序名称
2. 当前的目录/子程序名称_SPF

3. 当前的目录/子程序名称_MPF
4. /_N_SPF_DIR/子程序名称_SPF
5. /_N_CUS_DIR/子程序名称_SPF
6. /_N_CMA_DIR/子程序名称_SPF
7. /_N_CST_DIR/子程序名称_SPF

"当前目录" 表示选中主程序的目录。"子程序标识符" 表示使用EXTCALL编写的子程序名称。

POWER ON, RESET

通过复位和POWER ON（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。

关于 "从外部执行" 的其它信息可参阅功能说明：/FB/ K1, BAG, 通道，程序运行。

可设定的后装载存储器（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。加载内存的大小默认值为 30 KByte。

使用MD 18360:MM_EXT_PROG_BUFFER_SIZE 可以设置加载内存的大小。加载内存的数量可使用MD18362:MM_EXT_PROG_BUFFER_NUM
进行设置。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。

2.16 子程序调用，带M/T功能

功能

通过机床数据可以设置：通过子程序调用替换T功能或者M功能。例如可用于调用换刀程序。
在程序段查找时，带M/T功能的子程序调用性能如同标准子程序调用。

使用M6换刀举例

M-功能 M6 被换刀程序**WZW_UP_M6** 替代。

```
N10 PROC SCHRUPPEN3
N20 G1 F1000
N30 X=... Y=... Z=...
N40 T1234 M6 ;                                    ; WZW_UP_M6 调用
```

```
M30
所属的子程序 WZW_UP_M6:
N110 PROC WZW_UP_M6
...
N130 G53 D0 G0 X=... Y=... Z=... ;                ; 返回刀具更换点
N140 M6 ;                                         ; 执行刀具更换
...
N190 M17
```

使用T功能编程换刀举例

T-功能 被换刀程序**WZW_UP_T** 替代。

```
N10 PROC SCHRUPPEN4
N20 G1 F1000
N30 X=... Y=... Z=...
N40 T1234 ;                                       ; WZW_UP_T 调用
M30
所属的子程序 WZW_UP_T:
N310 PROC WZW_UP_T
...
N330 IF $C_T_PROG == 1
N340 G53 D0 G0 X=...Y=...Z=...                    ; 返回刀具更换点
N350 T=$C_T                                       ; 执行刀具更换
N360 ENDIF
...
N390 M17
```

T功能替换的扩展

T功能替换可扩展成通过机床数据可以设置是否在同时编程：
D号或者DL号和T号时

- 在一个程序段D或者DL中根据默认设置作为参数被传送给T功能替换循环 (默认设置) **或者**
- 在调用T功能替换循环之前被执行。

通过 MD 10719:T_NO_FCT_CYCLE_MODE ，T功能替换的参数设定以

- 值 0:如迄今为止那样，将D或者DL号直接传递给循环（默认值）
- 值 1:直接在程序段中计算D或者DL号

仅当已经使用某个M功能设计了换刀时，该功能才可激活 (MD 22550: TOOL_CHANGE_MODE = 1), 否则就总是传递D或者DL值。

关于“使用M/T功能调用子程序”的其它信息可参阅功能说明：/FB/ K1, BAG, 通道，程序运行。

2.17 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	调用用于第一个循环
C2(CYCLE82) 钻削，锪面	调用用于第二个循环
...	
C24(CYCLE98) 螺纹的序列	调用用于最后一个循环
M17	文件结束

编程

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

举例：

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：

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	

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	

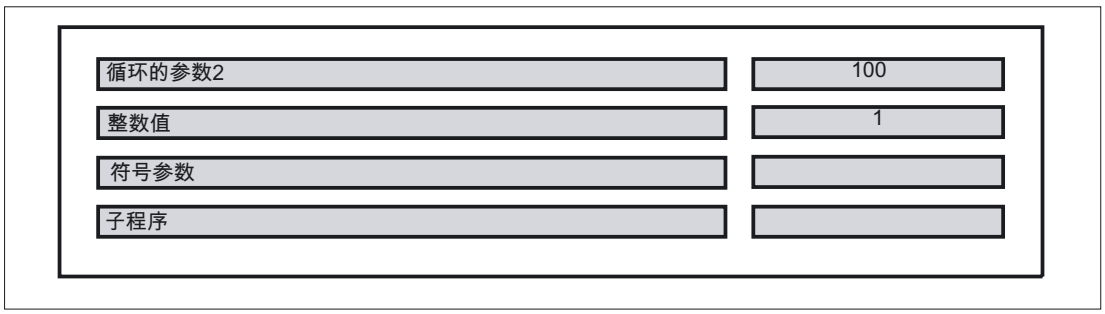
所属的文件 uc.com:

%_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)

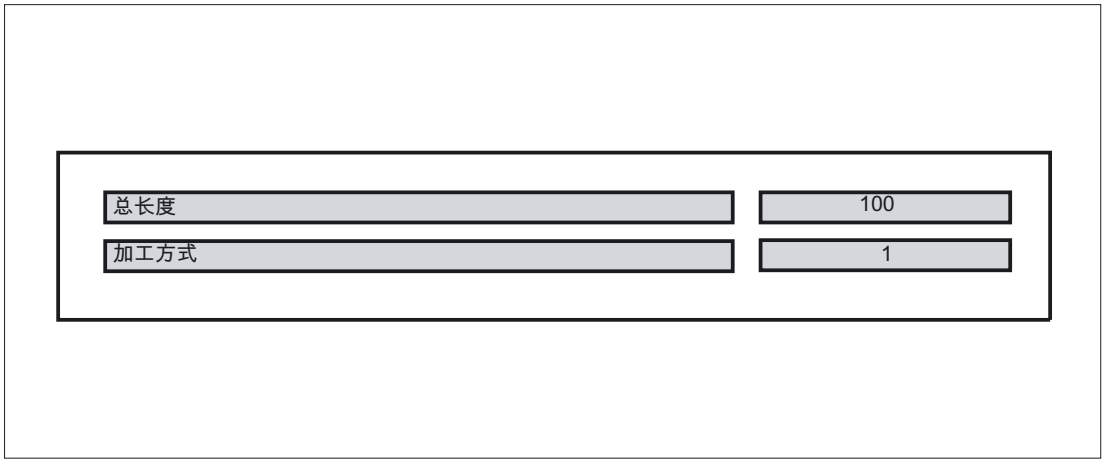
//C26(SPEZIALZYKLUS)
(R///总长度)
(I/*123456/3/加工方式)
M17

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

举例：

//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。

举例：

(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。

2.18 宏指令技术 (DEFINE...AS)

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。

宏指令使用

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。

编程

宏指令均有关键字 DEFINE…AS 作为标识。

宏指令定义为：

DEFINE NAME AS <语句>

举例：

宏指令定义：

```
DEFINE LINIE AS G1 G94 F300
```

在NC程序中调用：

```
N20 LINIE X10 Y20
```

激活宏指令

当宏指令被加载到NC中时（功能键“加载”），该宏指令就被激活。

参数

小心

不得使用宏指令对关键字和备用名称进行覆盖定义。

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

DEFINE	宏指令定义
NAME	这里是宏名称
AS	宏定义 STRING
语句	编程语句例如 G-, M- H- 和 L-功能

使用宏指令技术可以定义任意的命名符、G-/M-/H-功能和L-程序名。H功能和L功能可以两位编程。

三位 M-/G-功能

可以编写三位M功能和G功能。

举例：

```
N20 DEFINE M100 AS M6
N80 DEFINE M999 AS M6
```

注意

宏指令也可以在NC程序中约定。只有命名符才允许用作宏指令名称。G功能宏指令仅可以在宏指令模块中由系统全局约定。

不可以嵌套宏指令。

宏定义举例

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。

宏文件举例

在控制系统中读入该宏指令文件之后，激活宏指令（参见上面）。现在可以在零件程序中使用这些宏指令。

```
%_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

3

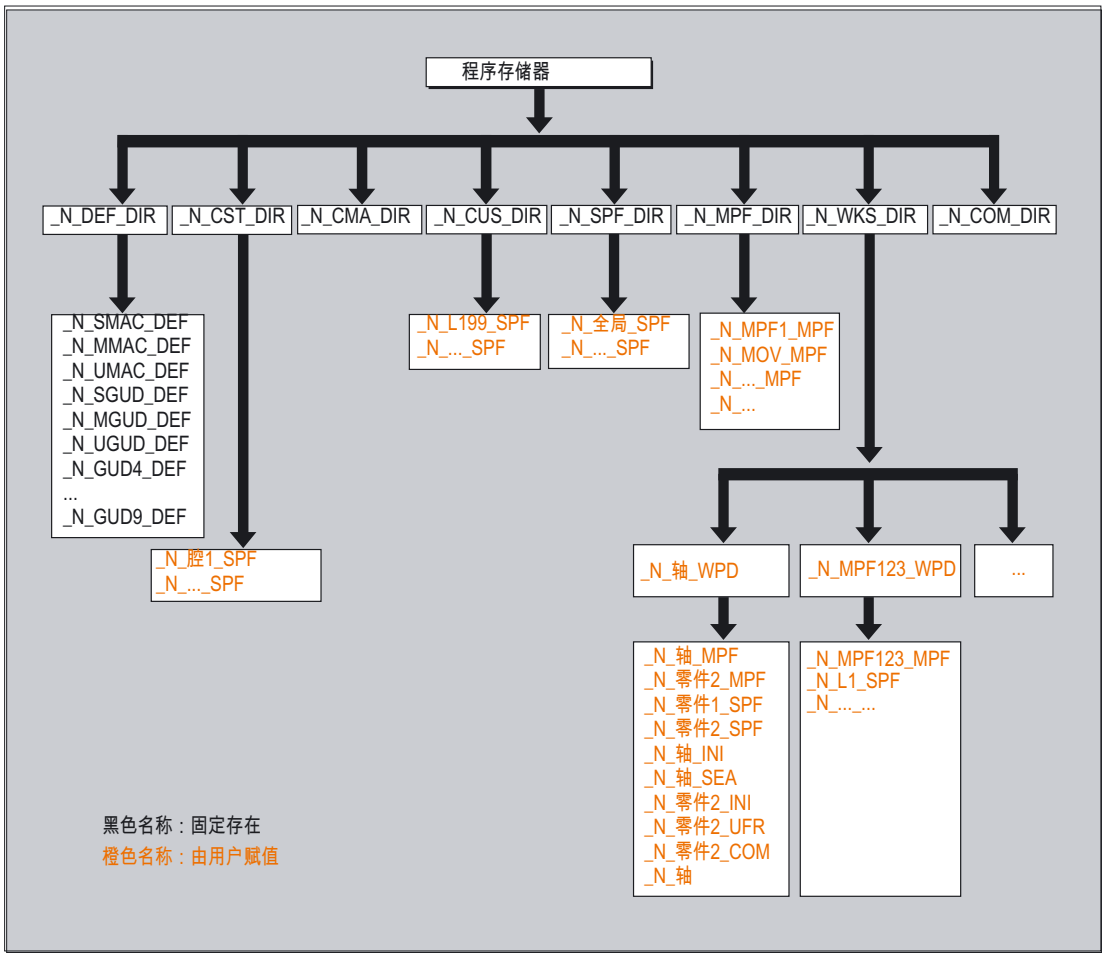
3.1 程序存储器

功能

在程序存储器中存储文件和程序，并且可以长期保存（无源文件系统）

举例：主程序和子程序，宏定义。

主程序和子程序存储在零件存储器中。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）。



目录

正常情况下有以下目录：

1. _N_DEF_DIR	数据块和宏指令块
2. _N_CST_DIR	标准循环
3. _N_CMA_DIR	机床制造商循环
4. _N_CUS_DIR	用户循环
5. _N_WKS_DIR	工件
6. _N_SPF_DIR	全局子程序
7. _N_MPF_DIR	主程序标准目录
8. _N_COM_DIR	注释标准目录

文件类型

在程序存储器中可以有以下文件类型：

*名称*_MPF	主程序
*名称*_SPF	子程序

*名称*_TEA	机床数据
*名称*_SEA	设定数据
*名称*_TOA	刀具补偿
*名称*_UFR	零点偏移/框架
*名称*_INI	初始化文件
*名称*_GUD	全局用户数据
*名称*_RPA	R参数
*名称*_COM	注释
*名称*_DEF	全局用户数据和宏指令定义

说明

工件目录，_N_WKS_DIR

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

工件目录，标识符 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。工件目录包含加工一个工件所需的所有文件。它可以是主程序，子程序，任意初始化程序和注释文件。

初始化程序根据机床数据MD 11280:WPD_INI_MODE 在选中程序之后第一次开始执行程序时进行一次。

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

_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。

;$PATH-指令

在某个文件的第二行中，使用$PATH=… 指定目标路径。

举例：

```
;$PATH=/_N_WKS_DIR/_N_WELLE_WPD
```

文件存放在所说明的路径下。

注意

如果缺少路径说明，就将文件类型为SPF /_N_SPF_DIR的文件、后缀为_INI的文件存放在工作存储器中，将所有其它文件存放在/_N_MPF_DIR 中。

为上一个示例指定路径的举例：

WELLE:_/N_WELLE_MPF 被存放在/_N_WKS_DIR/_N_WELLE_WPD中

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

WELLE:_/N_WELLE_SPF 被存放在/_N_SPF_DIR中

```
•
```

%_N_WELLE_SPF

```
•
M17
```

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。

举例：

工件目录/_N_WKS_DIR/_N_WELLE_WPD 含有文件_N_WELLE_SPF und _N_WELLE_MPF.

仅 MMC102/103:参见 "操作说明书" /BA/ 章节 "任务列表" 以及 "选择用来执行的程序".

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。

举例说明用绝对路径参数调用一个子程序。

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

在正常情况下不用说明路径调用程序：

举例：

```
CYCLE1
```

查找路径的顺序

1.	当前目录 / *名称*	工件目录或者默认目录 _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

对子程序调用时的查找路径进行编程

CALLPATH-指令

在调用子程序时可以使用零件程序指令CALLPATH扩展查找路径。

举例：

```
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。

3.2 工作存储器

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）。

举例：有效的机床数据，刀具补偿数据，零点偏移。

参数

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：

*名称*_TEA	机床数据
*名称*_SEA	设定数据
*名称*_TOA	刀具补偿
*名称*_UFR	零点偏移/框架

*名称*_INI	初始化文件
*名称*_GUD	全局用户数据
*名称*_RPA	R参数

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(不是指 810D CCU1, 840D NCU 571) 或者通常也可拥有多个轴。有：

标记	数据区
NCK	NCK专用数据
CHn	通道特有的数据（n 用来指定通道号）
AXn	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举例

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。

多通道控制方式举例

用于多个通道的CHANDATA (通道号)仅在文件N_INITIAL_INI 允许。

N_INITIAL_INI 是启动文件，用来初始化控制系统的所有数据。

```
% _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：

```
$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-语句设置给执行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。

注意

在工作表中的INI文件不含CHANDATA指令。

备份初始化程序

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文件：_N_INITIAL_INI 。

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。

所有文件类型说明均可参见操作编程说明。

3.3 定义用户数据

功能

注意事项

在启动时定义用户数据 (GUD)。所需要的机床数据必须做相应的设置。用户存储器必须已经配置。所有相关的机床数据均有名称组成GUD。

用户数据的定义(GUD) 可以在 HMI 操作界面中在服务操作范围内完成。这样就可免于耗费时间重新载入数据备份(%_N_INITIAL_INI).

适用于：

- 在硬盘上的定义文件无效。
- 在NC上的定义文件始终有效。

编程

单独的GUD变量用DEF指令编程：

DEF 范围 VL停止 类型 名称[.., ...]=值

参数

区	区标记该变量作为GUD变量，并确定应用范围： NCK NCK-全程 CHAN 通道全程
VL-停止	进刀停止选件特性： SYNR 读取时的进给停止 SYNW 写入时的进给停止 SYNRW 读写时的进给停止
类型	数据类型 BOOL REAL INT AXIS FRAME 字符串 CHAR
名称	变量名

[.., ...]	数组变量可选运行界限
值	可选择预置值，数组的多个值，用逗号隔开或者 REP (w1), SET(w1, W2, ...), (w1, w2, ...) 如果是框架型就不可能有初始化值。

定义文件举例，全局数据（西门子）

```
%_N_SGUD_DEF
;$PATH=/_N_DEF_DIR
DEF NCK REAL RTP                    ；退回平面
DEF CHAN INT SDIS                   ；安全距离
M30
```

定义文件举例，全局数据（机床制造商）

```
%_N_MGUD_DEF
;$PATH=/_N_DEF_DIR
；机床制造商的全局文件定义
DEF NCK SYNRW INT STUECKZAHL        ;读取/写入时的隐式进给停止
                                    ;存在于控制系统中的特定数据
                                    ；读写所有通道
DEF CHAN INT WERKZEUGTABELLE[100]   ;刀具表，用来根据特定的通道将
                                    ；刀具号映射到刀库位置上
M30                                 ;为每个通道单独建立表
```

预留的模块名称

在目录/_N_DEF_DIR 中可以存放下列模块：

_N_SMAC_DEF	含有宏定义（西门子系统应用）
_N_MMAC_DEF	包含宏指令定义（机床制造商）
_N_UMAC_DEF	包含宏指令定义（用户）
_N_SGUD_DEF	包含全局数据定义（西门子 - 系统应用）
_N_MGUD_DEF	包含全局数据定义（机床制造商）
_N_UGUD_DEF	包含全局数据定义（用户）
_N_GUD4_DEF	可以自由定义

_N_GUD5_DEF	含有测量循环的定义（西门子系统应用）
_N_GUD6_DEF	含有测量循环的定义（西门子系统应用）
_N_GUD7_DEF	含有默认循环的定义（西门子系统应用）
_N_GUD8_DEF	可以自由定义
_N_GUD9_DEF	可以自由定义

注意

如果不存在测量循环/默认循环，也可任意定义备用模块。

1. **定义用户数据（GUD）**
    1. 备份模块_N_INITIAL_INI
    2. 在HMI服务操作区中建立用户数据的定义文件
    3. 将定义文件加载到控制系统的程序存储器中
    4. 激活定义文件
    5. 数据保护
2. **给用户数据建立定义文件。**

   可以在外部计算机上或者在服务操作区中建立定义文件。也存在预定义的文件名（参见“备用模块名称”）：
   _N_SGUD_DEF
   _N_MGUD_DEF
   _N_UGUD_DEF
   _N_GUD4_DEF … _N_GUD9_DEF
   具有这些名称的文件可以含有GUD变量的定义。
3. **将定义文件加载到控制系统的程序存储器中**

   控制系统始终以默认方式创建一个目录_N_DEF_DIR。该名称作为路径登记到GUD定义文件头，并在由RS232接口读入时处理。
4. **激活定义文件**

   当GUD定义被加载到NC中时（功能键“加载”），该定义就被激活。参见“自动激活....”。
5. **数据保护**

   当从工作存储器中备份文件_N_COMPLETE_GUD时，仅备份数据内容。为全局用户变量建立的定义文件必须分开存放。
   给全局用户数据的变量赋值也可备份在_N_INITIAL_INI中，名称必须与定义文件中的名称相同。

注意

可设计的参数范围

通过机床数据可以按照通道特有的参数范围扩展各个GUD模块，也可以在同步动作中读写这些模块。由此GUD参数性能就像R参数。新建立的 GUD-范围可以使用数据类型 REAL, INT 和 BOOL 进行定义，参见预定义变量名称列表。

预定义变量名称列表			
Synact-GUD 的名称			
数据类型为**REAL**	数据类型为**INT**	数据类型为**BOOL**	在**模块**
SYG_RS[ ]	SYG_IS[ ]	SYG_BS[ ]	SGUD-模块
SYG_RM[ ]	SYG_IM[ ]	SYG_BM[ ]	MGUD-模块
SYG_RU[ ]	SYG_IU[ ]	SYG_BU[ ]	UGUD-模块
SYG_R4[ ]	SYG_I4[ ]	SYG_B4[ ]	GUD4-模块
SYG_R5[ ]	SYG_I5[ ]	SYG_B5[ ]	GUD5-模块
SYG_R6[ ]	SYG_I6[ ]	SYG_B6[ ]	GUD6-模块
SYG_R7[ ]	SYG_I7[ ]	SYG_B7[ ]	GUD7-模块
SYG_R8[ ]	SYG_I8[ ]	SYG_B8[ ]	GUD8-模块
SYG_R9[ ]	SYG_I9[ ]	SYG_B9[ ]	GUD9-模块

其它信息参见：/IAD/, “控制系统的参数设定”一章。

3.4 用户数据保护级别，MD, SD 和 NC语言指令

3.4.1 定义用户数据保护级别 (GUD)

功能

通过定义读写准则，使GUD模块免受处置。在循环中GUD变量可以被询问，它们不会通过HMI操作修改，也不会由程序进行修改。访问权限涉及在此模块中所定义的**所有** 变量。如果进行了不允许的存取，则控制系统会给出一个报警。

编程

在根据顶行指定整个模块的保护级别。

```
%_N_MGUD_DEF                                ; 模块种类
;$PATH=/_N_DEF_DIR                          ; 路径
APR 值 APW n                                ; 保护级在独立行中
```

在第一次定义一个变量之前，在GUD模块中按照所要求的保护级编程存取保护。关键字必须位于一个独立的程序段中。

参数

保护级：	访问级别 (**A**ccess **P**rotection)
APW n	用于写入 (**W**rite)
APR n	用于读取 (**R**ead)
n	使用级别 n
	从 0 或者 10 (最高级别)
	到7或者17 （最低级）
保护级n的含义：	
0或10	西门子
1或11	OEM_高
2或12	OEM_低
3或13	最终用户
4或14	钥匙开关3
...	...
7或17	钥匙开关0
APW 0-7, APR 0-7 读写保护在操作界面上**和** NC程序中或者 MDA-运行方式中起作用。	允许这些值在GUD模块中，以及用于REDEF指令中各个变量的保护级参数说明。
APW 10-17, APR 10-17: 读写保护在这里对操作界面有效。	这些值只允许在模块专用的GUD保护级中。

注意

如果某个完整的文件受到保护，那么这些指令必须位于文件的第一批定义之前，否则就在有关数据的**REDEF** - 语句之中，参见“NC语言指令的保护级别”。

带有写访问保护的定义文件距离

（机床制造商），读取（操作界面上的钥匙开关2）

```
%_N_GUD6_DEF
;$PATH=/_N_DEF_DIR
APR 15 APW 12                          ;适用于所有下列变量的保护级别
DEF CHAN REAL_CORRVAL
DEF NCK INT MYCOUNT
…
M30
```

首次激活GUD定义文件

当首次激活某个GUD定义文件时，可能会对其中所包含的已定义访问权限进行分析，并且自动向GUD定义文件的读写权限回送。

注意

在GUD定义文件中登记存取权，可以把存取权限制在GUD定义文件中，但不可进行扩展。

举例：

在定义文件 _N_GUD7_DEF中：APW2

1. 文件_N_GUD7_DEF有值3作为写保护。然后值3用值2改写。
2. 文件_N_GUD7_DEF有值0作为写保护。没有改变。

使用指令APW 可以自行反过来影响文件的写入权限。

使用指令APR 可以自行反过来影响文件的读取权限。

注意

如果因疏忽大意在GUD文件中登记了一个比自己的访问权限还要高的级别，就必须重新拷入文档文件。

3.4.2 自动激活 GUD 和 MAC

功能

在服务操作区中为 MMC 102/103 编辑GUD定义和宏定义的定义文件。

如果在NC中编辑一个定义文件，则在离开编辑器时出现一个询问：这些定义是否应该被设置为有效？

卸载GUD定义和宏定义

如果卸载某个定义文件，就会在出现一次询问之后删除相应的数据模块。

加载GUD定义和宏定义

如果装载一个定义文件，则出现一个询问：文件是否应该激活？或者这些文件是否应该保持不变？如果放弃激活，则文件不装载。

如果光标位于某个已加载的定义文件上，功能键上的标志符号就会从“加载”变成“激活”，以便使定义生效。当选择“激活”时，就会再次询问数据是否应当保留。

仅在变量定义文件中保护数据，在宏指令中没有。

注意

MMC 103

如果没有足够的内存可供激活定义文件，则必须在修改内存大小时候将文件从NC载入到MMC中，并且重新载入NC中以使其激活。

退出编辑器时的询问举例

“您要从文件GUD7.DEF中激活这些定义吗？”

"OK": 出现一个询问：是否应该保护当前有效的文件？
“这些定义的当前文件应该保持不变吗？”
OK": 将需要编辑的定义文件的GUD模块备份，激活新的定义文件并且将拯救出来的数据重新拷入。
“取消”： 将新定义激活，旧文件丢失。
“取消”： 取消在定义文件中所做的修改，有关数据模块没有被修改。

3.4.3 修改机床数据和调整数据的保护级别 (REDEF MD, SD)

功能

用户可以对保护级别进行**修改**。在机床数据中仅可以分配优先级较低的保护级，在设定数据中则要高一些。

编程

REDEF 机床数据/调整数据 保护级别

参数

REDEF	重新定义 (**REDEF**inition) 例如 设置机床数据和调整数据
机床数据/设定数据	应分配有某个保护级别的机床数据和调整数据。
保护级：	访问级别 (**A**ccess **P**rotection)
APW n	用于写入 (**W**rite)
APR n	用于读取 (**R**ead)
n	使用级别 n 从0（最高级） 到7 （最低级）

重新复位机床数据/调整数据

如果要求可以再次取消保护级的修改，则原来的保护级必须再次写回。

REDEF 扩展

关于零件程序中 REDEF 语句作用方式的其它信息可参阅“NC语言指令的保护级别”一章。

修改各个MD的权限举例

```
%_N_SGUD_DEF
;$PATH=/_N_DEF_DIR
REDEF $MA_CTRLOUT_SEGMENT_NR APR 2 APW 2
REDEF $MA_ENC_SEGMENT_NR APR 2 APW 2
REDEF $SN_JOG_CONT_MODE_LEVELTRIGGRD APR 2 APW 2
M30
```

将各个MD的权限复位成原始值举例

```
%_N_SGUD_DEF
;$PATH=/_N_DEF_DIR
REDEF $MA_CTRLOUT_SEGMENT_NR APR 7 APW 2
REDEF $MA_ENC_SEGMENT_NR APR 0 APW 0
REDEF $SN_JOG_CONT_MODE_LEVELTRIGGRD APR 7 APW 7
M30
```

3.4.4 NC语言指令的保护级别 (REDEF)

功能

用于读写机床数据/设定数据和GUD的保护级方案扩展到上面所说的零件程序指令。对此，可以用REDEF指令给一个零件程序分配一个保护级0到7。

注意

现在只有当相应的执行权限存在时，才会执行该指令。

编程

G代码称作“G功能/行程条件列表”

REDEF (NC-语言元素) **APX** 值

或者某个从零件程序或者同步动作到系统变量的写访问

REDEF (系统变量) **APW** 值

或者改变对目前为止的机床数据和调整数据的写访问或者读取访问

REDEF (机床数据/调整数据) **APW** 值

REDEF (机床数据/调整数据) **APR** 值

REDEF (机床数据/调整数据) **APR** 值 **APW** 值

参数

REDEF指令对所有通道和BAG全局有效

REDEF	**REDEF指令的作用和应用**
NC语言单元	语言单元，该语言单元应该分配一个保护级用于执行该指令： 1．预定义的子程序/功能 （参见同名列表） 2．用于同步动作的关键字 "DO"语句 3．G-功能 （G-功能/行程条件） 4．用于循环的程序标识符 循环必须已保存在某一个循环目录中并且含有一个PROC语句。
系统变量	应分配有某个写访问保护级别的系统变量。始终可以读存取。（参见系统变量列表）.
机床数据/设定数据	应分配有某个读写访问保护级别的机床数据或者调整数据。

APX	访问保护的关键字
APW, APR	执行
	读，写
值	保护级别的数值值（0～7）
	从0（最高级）
	到7 （最低级）
值7	钥匙开关位置0对应所有零件程序指令的预设定。

定义文件中的子程序调用举例

```
N10 REDEF GEOAX APX 3
N20 IF(ISFILE("/_N_CST_DIR/_N_SACCESS_SUB1_SPF"))
N30 PCALL /_N_CST_DIR/_N_SACCESS_SUB1_SPF
N40 ENDIF
N40 M17
```

说明

与GUD定义类似，有自身的定义文件供使用，它们在系统引导时处理：

最终用户：/_N_DEF_DIR/_N_UACCESS_DEF

制造商：/_N_DEF_DIR/_N_MACCESS_DEF

西门子：/_N_DEF_DIR/_N_SACCESS_DEF

定义文件中的子程序调用

在上面所述的定义文件中也可以调用包含REDEF指令的子程序。REDEF语句与DEF语句一样，基本上必须在数据块部分中的开始处。子程序必须有扩展名SPF或者MPF，并且继承用$MN_ACCESS_WRITE_xACCESS设定的定义文件的写保护。

注意

REDEF指令的扩展

只要“NC语言指令保护级别”功能已经激活，就必须将目前为止保存在GUD定义文件中的重新定义从机床数据/调整数据转移到用于进行保护级别赋值的新定义文件中，也就是说，对机床数据和调整数据保护级别的设置只有在上述保护级别定义文件中才会被允许，并且在GUD定义文件中被拒绝且发出报警15420。

注意

仅在GUD定义文件中可以设置初始化属性和同步动作属性。

系统变量保护级

系统变量的保护级仅适用于通过零件程序指令的赋值语句。在操作界面中，各个HMI高级的/内置的保护级方案生效。

关于“保护级别方案”的其它信息可参阅：

/BAD/, HMI 操作说明，在“钥匙开关”和“机床数据”一章中

/IAD/, 调试说明，“控制系统的参数设定”

3.5 REDEF:修改NC语言元素的属性

功能

通常利用可用的REDEF语句扩展，将在现有子章节中所描述的有关定义数据对象的功能和保护级别的确定，用来针对某个一般性接口进行属性和值的设置。

编程

REDEF NC语言元素 属性 值

参数

NC语言单元	以下适用：
	GUD50
	R参数
	机床数据/设定数据
	同步变量 ($AC_PARAM, $AC_MARKER, $AC_TIMER)
	可从零件程序写入的系统变量 (参见 PGA1)
	用户框架 (G500，等等)
	刀库/刀具配置
属性	**允许：**
初始化	
INIPO	GUD, R-参数, 同步变量
INIRE	GUD, R-参数, 同步变量
INICF	GUD, R-参数, 同步变量
PRLOC	设定数据

同步	允许：	给定一个缺省值：
SYNR	GUD50	在读时进刀停止
SYNW	GUD50	在写入时进刀停止
SYNRW	GUD50	读取和写入时停止进给
访问权限	允许：	
	机床数据和调整数据	
APW	机床数据和调整数据	写存取权限
APR		读存取权限
		对于机床数据和设定数据，可以后面再改写预设定的存取权。此时允许值从
		'0' (西门子密码) 到
		'7' (钥匙开关位置 0)

值（可选）

在属性INIPO、INIRE、INICF和PRLOC中可选的参数：补充性起始值

形式：

单一值	z. B. 5
值列表	例如 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) 用于有10个元素的变量
REP (w1)	带w1:需要重复的值列表 用于带有多个元素的变量，例如 REP(12)
SET(w1, w2, w3, ...) (w1, w2, w3, ...)	或者 值列表
n:	保护级别所需参数，当属性为APR 或者 APW时

对于**GUD** 而言，可以在定义时指定起始值 (DEF NCK INT _MYGUD=5).如果该起始值（比如在DEF NCK INT _MYINT时）没有说明，则在REDEF指令中可以后来再确定该起始值。

某个数组的初始化值适用于所有数组元素。各个元素可以使用一个初始化列表或者REP()来设定。举例：

```
REDEF MYGUD INIRE 5
REDEF MYGUD INIRE 0,1,2,3,4,5,6,7,8
REDEF MYGUD REP (12,14,16,18,20)
```

不可 用于R参数和系统变量。

仅可以赋值常量。
表达式不允许作为值。

属性的含义

INIPO	**INI**t 当 **P**ower **O**n时 再次启动NC时，数据被缺省值改写。
INIRE	**INI**t 当操作面板**Re**set 或者 TP-结束时 当某个主程序使用例如M2， M30等等结束时，或者使用复位取消时，就会用默认值来覆盖数据。 在INIPO时INIRE也生效。
INICF	**INI**t 当 **N**ew**C**onf-请求或者TP指令NEWCONF时 在NewConf请求时或者TP指令NEWCONF时，数据以缺省值覆盖。 在INIRE 和INIPO时INICF也生效。
PRLOC	仅**程**序**局**部修改 如果某个零件程序、子程序、循环或者 ASUP 中的数据被修改，就会在主程序结束之后（例如使用M2，M30等等结束，或者通过操作面板复位来取消）将该数据重新设定其原来的值。 该属性仅允许用于可编程的调整数据，参见可编程的调整数据。

用户必须实现触发初始化的事件的 **同步动作** ，也就是说，例如在两个**不同的通道**中执行一个零件程序结束，这样每个这样的过程的变量就会被初始化。这会影响全局数据或者轴向数据！

可编程的调整数据和可从零件程序写入的系统变量

下列SD 可以与 REDEF 语句一起初始化：

序号	名称	GCODE
42000	$SC_THREAD_START_ANGLE	SF
42010	$SC_THREAD_RAMP_DISP	DITS/DITE
43210	$SA_SPIND_MIN_VELO_G25	G25
43220	$SA_SPIND_MAX_VELO_G26	G26
43230	$SA_SPIND_MAX_VELO_LIMS	LIMS
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
43720	$SA_OSCILL_DWELL_TIME1	OST1
43730	$SA_OSCILL_DWELL_TIME2	OST2
43740	$SA_OSCILL_VELO	FA
43750	$SA_OSCILL_NUM_SPARK_CYCLES	OSNSC

序号	名称	GCODE
43760	$SA_OSCILL_END_POS	OSE
43770	$SA_OSCILL_CTRL_MASK	OSCTRL
43780	$SA_OSCILL_IS_ACTIVE	OS
43790	$SA_OSCILL_START_POS	OSB

在 PGA1 "系统变量列表" 中可找到系统变量的列表。所有在零件程序中用W（写）或者WS（写，进刀停止）标记的系统变量可以用RESET指令（复位指令）初始化。

举例

```
GUD的复位特性：
/_N_DEF_DIR/_N_SGUD_DEF
DEF NCK INT _MYGUD1                                    ；定义
DEF NCK INT _MYGUD2 = 2
DEF NCK INT _MYGUD3 = 3
在面板前方按复位键/零件程序结束时初始化：
DEF _MYGUD2 INIRE                                      ；初始化
M17
```

由此在面板前方复位键/零件程序结束"_MYGUD2"时再次设定到值“2”，此时"_MYGUD1" 和"_MYGUD3"保持它们的值。

零件程序中的模态转速限制举例（调整数据）

```
/_N_DEF_DIR/_N_SGUD_DEF
REDEF $SA_SPIND_MAX_VELO_LIMS PRLOC          ；极限转速的调整数据
M17
```

```
/_N_MPF_DIR/_N_MY_MPF
N10 SETMS(3)
N20 G96 S100 LIMS=2500
...
M30
```

在设定数据($SA_SPIND_MAX_VELO_LIMS)极限转速中确定的极限转速为1200转/分钟。因为人们完全可以在一个编排和测试完毕的零件程序中允许一个较高的转速，所以这里编程LIMS=2500。但在程序结束之后，通过调整数据所设计的值就会重新有效。

边界条件

- 对NC对象属性的**修改** 只可以在**定义**这个对象之后进行。特别是在GUD时必须要注意DEF.../ REDEF的顺序。（设定数据/系统变量包含在内，并且在处理定义文件之前就已经设定）。始终必须先定义符号（被系统或者通过DEF语句隐式定义)
并且只有在此之后才能使用REDEF进行修改。
- 如果编程了几次属性修改，则最后的修改有效。
- **数组的属性** 不能设定给各个元素，而始终只能设定给**整个数组** ：

```
DEF NCK INT _MYGUD[10,10]
REDEF _MYGUD INIRE                          // ok
REDEF _MYGUD[1,1] INIRE                     // 不可以，发出报警
                                            // (数组值)
```

- **GUD-数组的初始化** 本身保持不受影响。

```
DEF NCK INT _MYGUD[10] =(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
DEF NCK INT _MYGUD[100,100] = REP (12)
DEF NCK INT _MYGUD[100,100] ;
```

- 带有**R参数** 的REDEF语句必须用括号说明。

```
REDEF R[ ]INIRE
```

- **INI属性**
不过此时仍然要注意：设置这些变量的INI属性时，必须有相应大小的**Init值内存**可供使用，可通过MD 18150:MM_GUD_VAL_MEM进行设置。在机床数据 MD 11270:DEFAULT_VALUES_MEM_MASK 必须已设定Bit1 = 1 (初始化值内存已激活).内存太小时，就会发出报警12261 "不允许初始化"
- **R-参数和系统变量**
对于R参数和系统变量而言，不可以设定与编译值有差距的默认值。但是用INIPO、INIRE或者INICF可以复位到汇编值。
- 对于**GUD的数据类型FRAME**而言，同样不可以（如同已有的数据定义）设定与编译值有差距的默认值。

给定一个缺省值

当使用 REDEF <名称> INIRE, INIPO; INICF; PRLOC 修改某个系统变量的属性或者 GUD 时，机床数据MD 11270:DEFAULT_VALUES_MEM_MASK = 1 必须设置（用于初始化数值的存储器有效）。如果不是这种情况，就会导致报警12261 "不允许初始化".

3.6 步骤编辑器中的结构化语句 SEFORM

功能

指令SEFORM在步进编辑器中处理，从而从中生成步进画面，用于HMI - 高级。在HMI-高级中有步骤视图可供使用，且可用来改善NC子程序的可读性。使用结构化指令SEFORM，步进编辑器（以编辑器为基础的程序辅助）通过3个参数支持。

编程

SEFORM(STRING[128] 文件名称, INT 级, STRING[128] 图标)

参数

SEFORM	用以下参数调用结构化指令：文件名称，级和图标
文件名称	工作步骤名称
级	主级或者子级索引。 =0 相当于主平面 =1, ... 相当于子级1到n
图标	图标名称，显示用于该文件。

注意

SEFORM指令在步进编辑器中产生。

用参数<文件名>变址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SEFROM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。

参数级和图标在执行零件程序时由NCK检查，但是不继续处理。

有关编辑器基础上的编程辅助的详细信息，参见：
/BAD/ 操作说明书 HMI 高级。

保护区

4

4.1 保护区的确定 (CPROTDEF, NPROTDEF)

功能

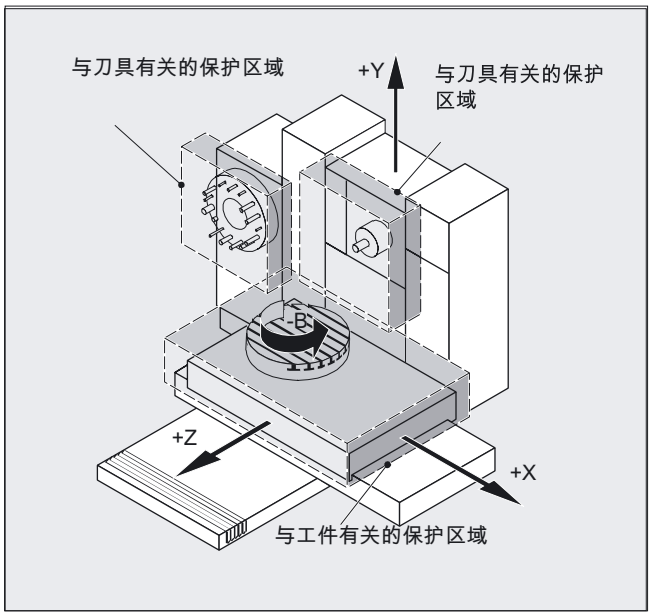
利用保护区，可以保护机床上各个不同的部件、夹具以及工件，防止误动作。

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）。



编程

```
DEF INT NOT_USED
CPROTDEF(n,t,applim,applus,appminus)
NPROTDEF(n,t,applim,applus,appminus)
EXECUTE (NOT_USED)
```

参数

DEF INT NOT_USED	定义局部变量、整数数据类型（可比较运动同步动作一章）
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
EXECUTE	结束定义
n	定义的保护区序号
t	TRUE = **与刀具有关**的保护区
	FALSE = **与工件有关**的保护区
applim	第3个尺寸的限制方式
	0 = 没有限制
	1 = 在正方向限制
	2 = 在负方向限制
	3 = 在正负方向限制
applus	第3个尺寸在正方向限制的值
appminus	第3个尺寸在负方向限制的值
NOT_USED	在有EXECUTE的保护区中故障变量无效

说明

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，可以相对偏移保护区基准点。

轮廓描述基准点

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。

保护区的轮廓描述

保护区的轮廓在所选择的平面中最多说明11个移动运行。这里，第一个移动运行指轮廓运行。轮廓左侧的区域作为保护区。在 CPROTDEF 或者 NPROTDEF 和 EXECUTE 之间的运动不被执行，而是定义保护区。

工作平面

在CPROTDEF 或者 NPROTDEF之前用G17、G18、G19选择所要求的平面，并且不允许在EXECUTE之前修改。在CPROTDEF 或者NPROTDEF和EXECUTE之间，不允许编程应 用。

轮廓单元

允许：

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

注意

对于 810D 而言，最多有4个轮廓元素可供用来定义相应的保护区(最多4个通道特有的和4个NCK-特有的保护区)。

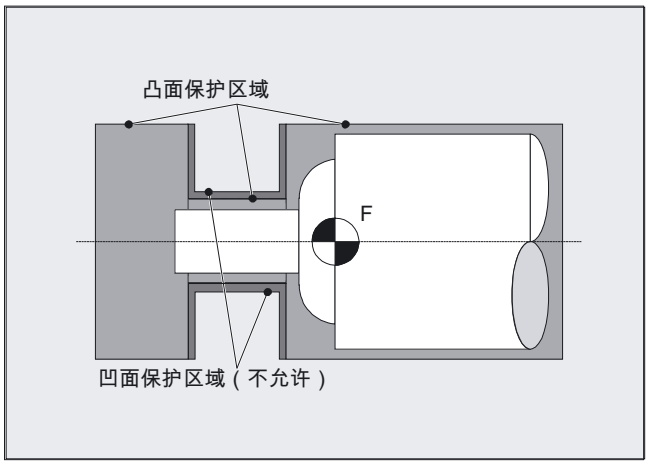
如果要求描述一个整圆作为保护区，则它必须分为两个分圆。不允许使用顺序G2、G3或者G3、G2。有时必须要插进一个较短的G1程序段。

轮廓描述的最后一个点必须与第一个点重合。

外侧保护区 (仅当与工件有关的保护区才可以) 应**以逆时针方向** 定义。

如果是**旋转对称的** 保护区（例如主轴卡盘），就必须描述**全部轮廓** (不仅仅到旋转中心为止!).

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。



在定义保护区时，

- 不允许铣刀半径补偿或者刀沿半径补偿，

- 不允许有转换，
- 不允许框架有效。

也不允许编程回参考点运行（G74）、固定点返回（G75）、程序段进刀停止或者程序结束。

4.2 激活、解除保护区 (CPROT, NPROT)

功能

激活、预先激活预先定义好的保护区来进行碰撞监控，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。

注意

如果没有工件相关的保护区有效，则不进行保护区监控。

编程

```
CPROT (n,状态,xMov,yMov,zMov)
NPROT (n,状态,xMov,yMov,zMov)
```

参数

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
状态	状态参数说明
	0 = 取消保护区
	1 = 预激活保护区
	2 = 激活保护区
xMov,yMov,zMov	偏移几何轴中已经定义的保护区

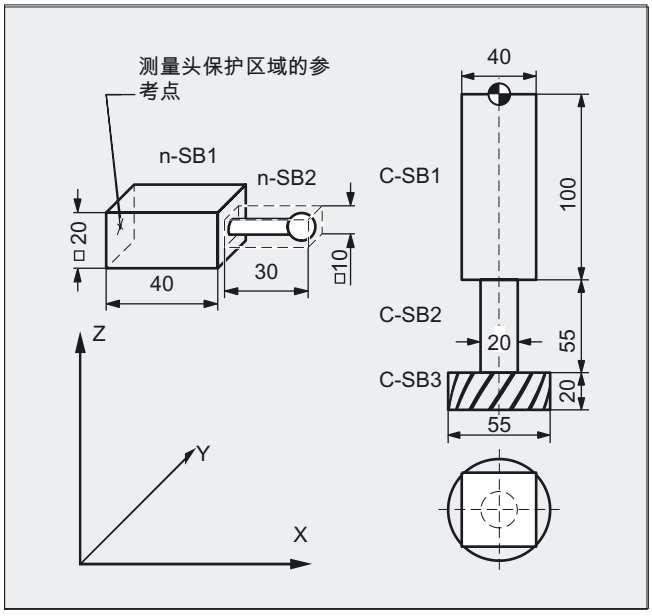
铣削示例

对于铣床而言，应对铣刀与测头可能会有的碰撞进行监控。测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：

- 各有一个机床专用的和与工件有关的保护区用于测头支架 (n-SB1) 和测头自身 (n-SB2).
- 各有一个通道专用的和刀具相关的保护区用于铣刀夹持架(c-SB1)、铣刀柄(c-SB2)和铣刀自身(c-SB3)。

所有保护区的定向均在Z方向中。

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




```
DEF INT SCHUTZB                          定义某个辅助变量
定义保护区                               设定方向
G17
NPROTDEF(1,FALSE,3,10,-10)               保护区n-SB1
G01 X0 Y-10
X40
Y10
X0
Y-10
EXECUTE(SCHUTZB)
```

```
NPROTDEF(2,FALSE,3,5,-5)                    保护区n-SB2
G01 X40 Y-5




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
```

激活状态

在通常情况下，在零件程序中用状态 = 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。

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，则可通过状态 = 1来进行所需的预先激活。

通过状态 = 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。为此必须已设置系统变量
`$SN_PA_ACTIV_IMMED [n]` 或者
`$SN_PA_ACTIV_IMMED[n] = TRUE`。
使用使用状态 = 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。

4.3 检查保护区侵犯情况、工作范围限制和软件极限值

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。该函数必须单独在某个程序段内。

编程

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

参数

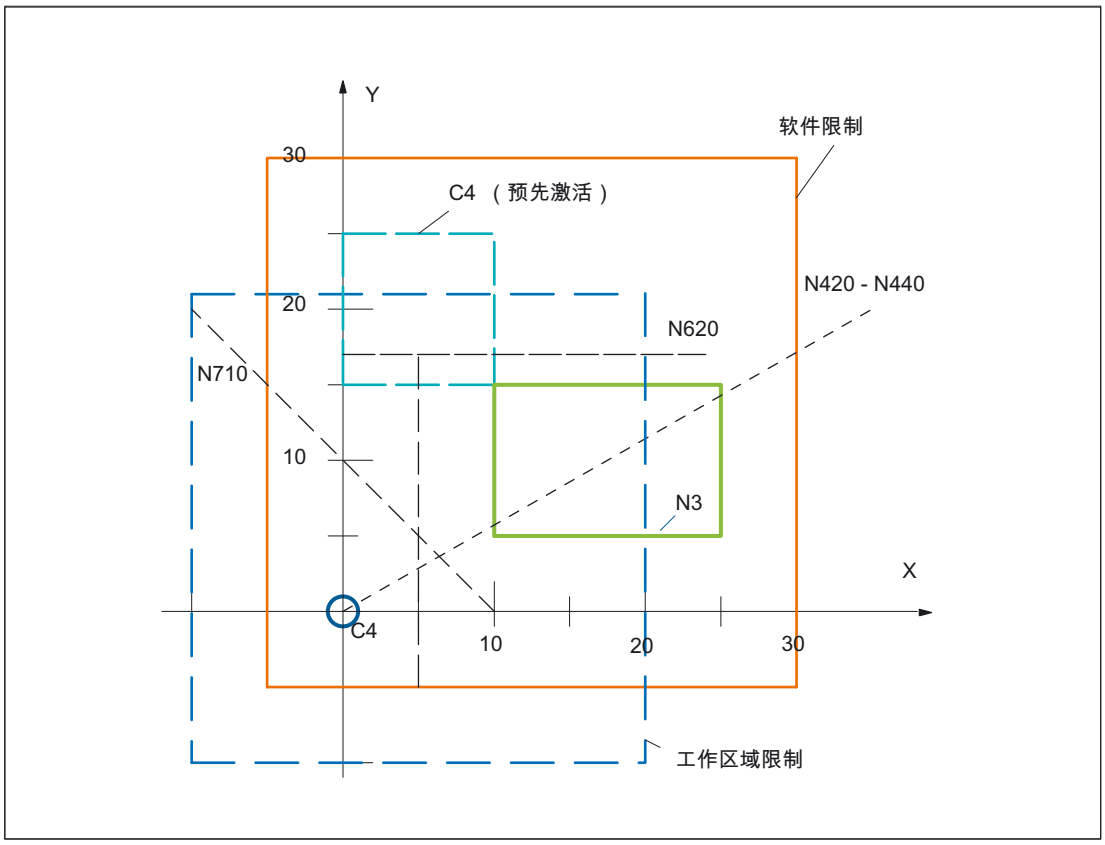
状态	0：功能正常， 所规定的行程可以完全执行完毕。 -：在_DLIMIT中至少有一个分量为负 -：在一个转换计算中出现一个错误 如果给定的位移不能完全运行，则送回一个正的、十进制编码的数值： **个位（受损界限种类）：** 1：软件极限限制运行行程。 2：工作区域极限限制运行行程。 3：保护区限制运行行程。 当同时有多个极限受到侵犯时（例如软件极限值和保护区），就在个位上显示可用来极为严格地限制规定运动行程的极限值。 **十位** 10: 起始值损害界限。 20: 规定的直线侵犯极限。当终点自身没有侵犯极限，但是在从起始点到终点的行程中却有可能侵犯某个极限值时（例如穿过保护区，当进行非线性转换时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函数 CALCPOSI 使用。 TRUE: 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。

举例

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函数CALCPOSI。各次调用CALCPOSI的结构被汇总在示例结束处的表格中。




```
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
N80 g17 g1 x-y0
N90 g3 i2 x2
N100 i-x-
N110 执行(_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 执行(_SB)
```

```
N190 nprotdef(3, false, 0)                ;机床相关的保护区
N200 g17 g1 x10 y5
N210 x25
N220 y15
N230 x10
N240 y5
N250 执行(_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
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.                    ;其它起始点
N460 _STARTPOS[1] = 17.
N470 _STARTPOS[2] = 0.
N480 _MOVDIST[0] = 0.                     ;其它目标
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	仅软件极限监控仍有效。
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（例如传送），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：当以这段距离来延长在 _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。

注意

有关工作范围极限的细节可在编程说明PG中查阅，有关软件极限值的细节可在函数说明A3中查阅。

特殊的位移指令

5

5.1 逼近已经过编码处理的位置 (CAC, CIC, CDC, CACP, CACN)

功能

通过机床数据可以在位置表中为2个轴各输入最多60个位置（0～59）。

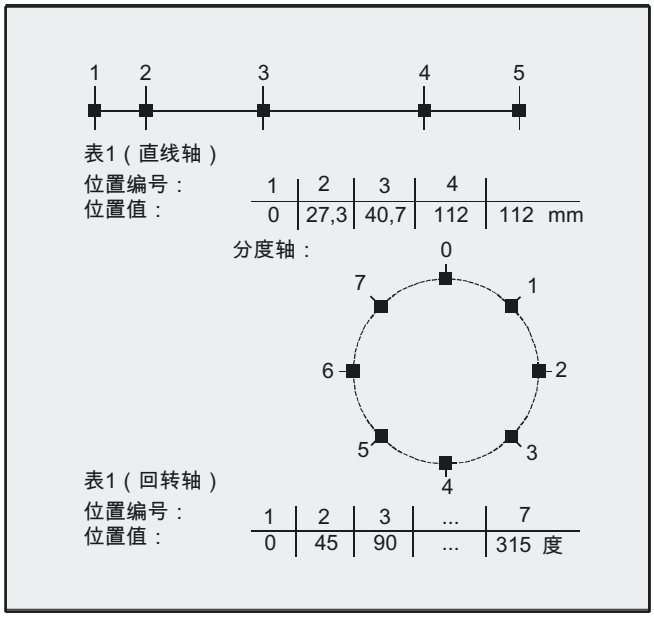
编程

CAC (n)
或者
CIC (n)
或者
CACP (n)
或者
CACN (n)

参数

CAC (n)	绝对返回编码的位置
CIC (n)	返回编码的位置，向前n位置（+）或者向后n位置（-）增量
CDC (n)	以最短的行程返回编码的位置（仅用于回转轴）
CACP (n)	编码的位置绝对在正方向（仅用于回转轴）
CACN (n)	编码的位置绝对在负方向（仅用于回转轴）
(n)	每个轴的位置号1,2,...最多60个

用于线性轴和旋转轴的定位表举例



注意

如果一个轴位于两个位置之间，则在带CIC(...)的增量数据时不运行。建议第一个运行指令始终以绝对位置参数编程。

举例2

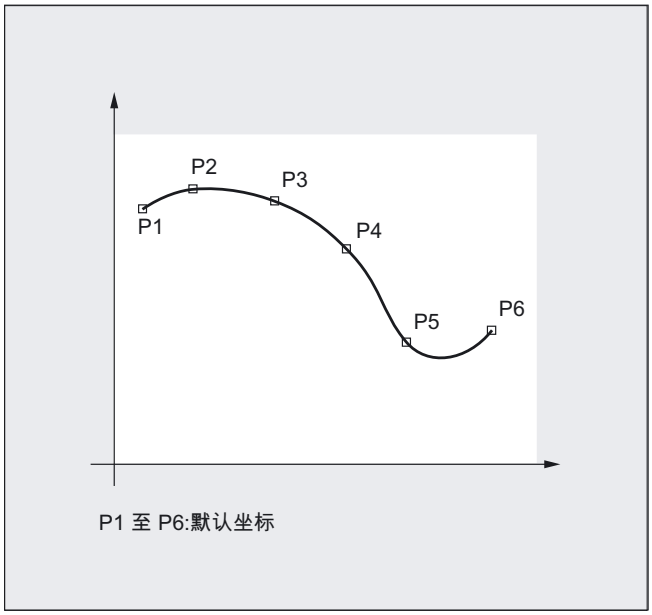
```
N10 FA[B]= 300                    ;用于位置轴B的进给
N20 POS[B]=    CAC     (10)       ;逼近已编码的位置10（绝对）
N30 POS[B]=    CIC     (-4)       ;从当前的位置返回4个工位
```

5.2 样条插补 (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN)

功能

通过样条插补可以用平滑的曲线连接各个分散的点。样条例如可用来将经过数字化处理的点与曲线连接。

不同的样条类型有不同的性能，它们也会产生不同的结果。除了可选择样条类型之外，用户也可对一系列参数施加影响。经常需要几次尝试才可以产生所要求的图形。



当要通过某个曲线连接一系列点时，就需要编程插补。可以有三种类型的样条：

- A样条（Akima样条）
- B样条（非统一、有理的基准样条，NURBS）
- C样条（立体样条）

编程

```
ASPLINE X Y Z A B C
```

或者

```
BSPLINE X Y Z A B C
```

或者

```
CSPLINE X Y Z A B C
```

参数

ASPLINE	Akima样条沿切线一侧经过所编程的节点。
BSPLINE	B样条不直接通过检查点，而只是在其附近经过。已编程的位置不是节点，而只是检查点。
CSPLINE	在切线一侧和曲面一侧节点上有过渡的三次样条。

A-,B-和C-样条为模态有效，属于行程指令组。
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。

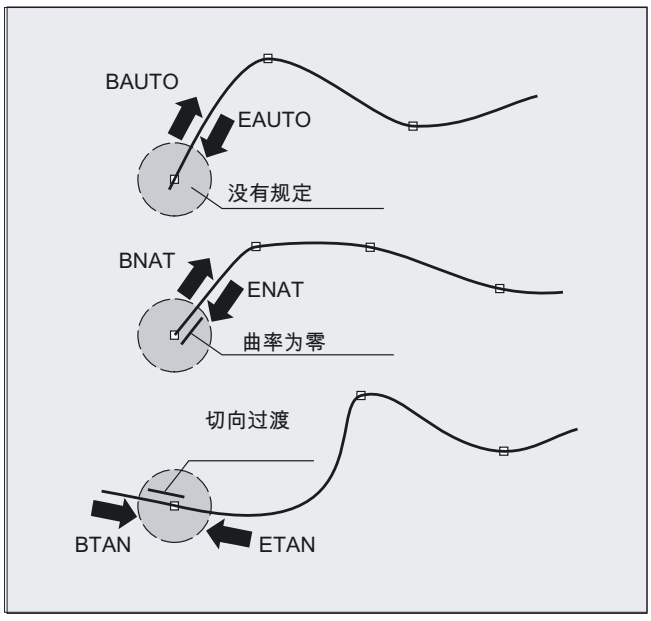
注意

A样条和C样条的参数

对于Akima样条（A样条）和三次样条（C样条）而言，可以对样条曲线开始和结束处过渡特定的边界条件进行编程。

过渡特性的这些边界条件可以划分为两个组，各以三个指令进行说明：

开始样条曲线：	
BAUTO	没有规定；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）
结束样条曲线：	
EAUTO	没有规定；从最后一个点的位置结束
ENAT	曲率为零
ETAN	切线过渡到下一段（清除位置）



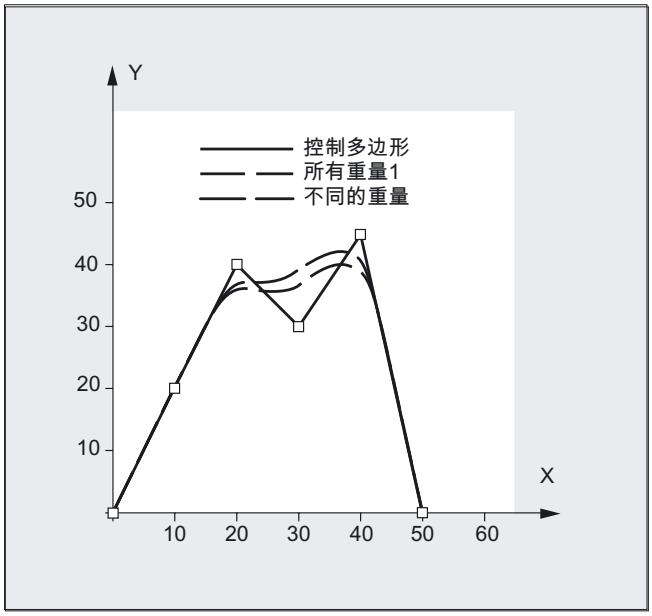
注意

B样条的参数

可编程的边界条件（参见A样条或者C样条）对B样条没有影响。在起始点和终点处B样条始终与控制多边形轮廓相切。

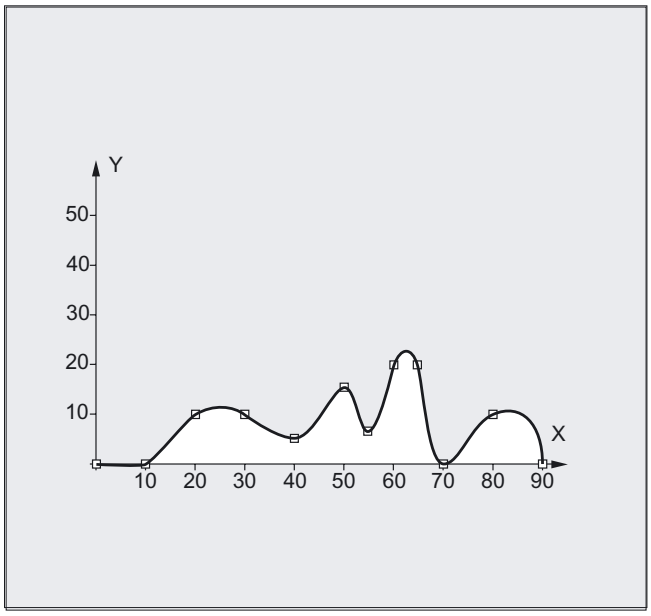
点权重：	
PW = n	对于每个节点，均可以将权重说明编程为所谓的点权重PW。
值范围：	
<= n <= 3	步距为 0.0001
作用：	
n > 1	曲线被检查点更为紧密地拉近
n < 1	曲线被检查点略微拉近
样条等级：	
SD = 2	正常情况下使用一个3级多项式。也可以使用一个2级多项式。
节点间距：	
PL = 值	节点间距适合于内部计算。但是，控制系统也可以对规定的节点间距进行处理，在所谓的参数间隔长度PL中对这些节点间距进行说明。
值	值范围如同位移

B 样条举例



所有重量1	不同的重量	控制多边形
N10 G1 X0 Y0 F300 G64	N10 G1 X0 Y0 F300 G64	N10 G1 X0 Y0 F300 G64
N20 BSPLINE	N20 BSPLINE	N20 ;省略
N30 X10 Y20	N30 X10 Y20 PW=2	N30 X10 Y20
N40 X20 Y40	N40 X20 Y40	N40 X20 Y40
N50 X30 Y30	N50 X30 Y30 PW=0.5	N50 X30 Y30
N60 X40 Y45	N60 X40 Y45	N60 X40 Y45
N70 X50 Y0	N70 X50 Y0	N70 X50 Y0

C样条举例，在曲面开始和结束处为零

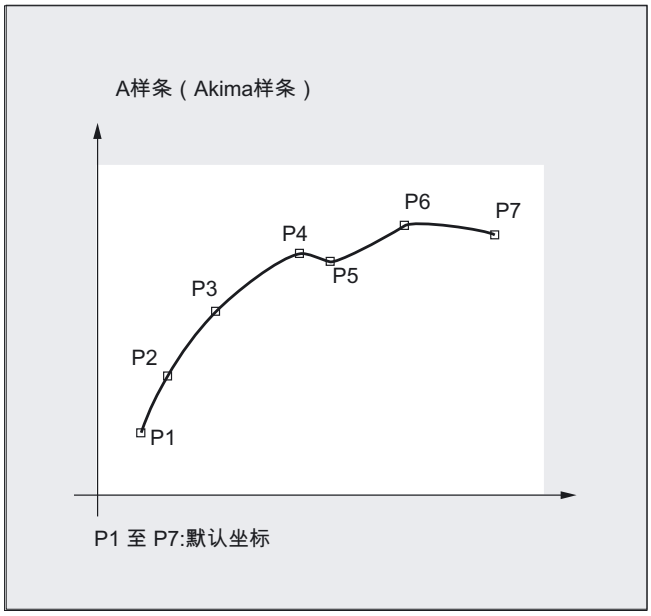



```
N10 G1 X0 Y0 F300
N15 X10
N20 BNAT ENAT
N30 CSPLINE X20 Y10
N40 X30
N50 X40 Y5
N60 X50 Y15
N70 X55 Y7
N80 X60 Y20
N90 X65 Y20
N100 X70 Y0
N110 X80 Y10
N120 X90 Y0
N130 M30
```

C样条举例，在曲面开始和结束处为零

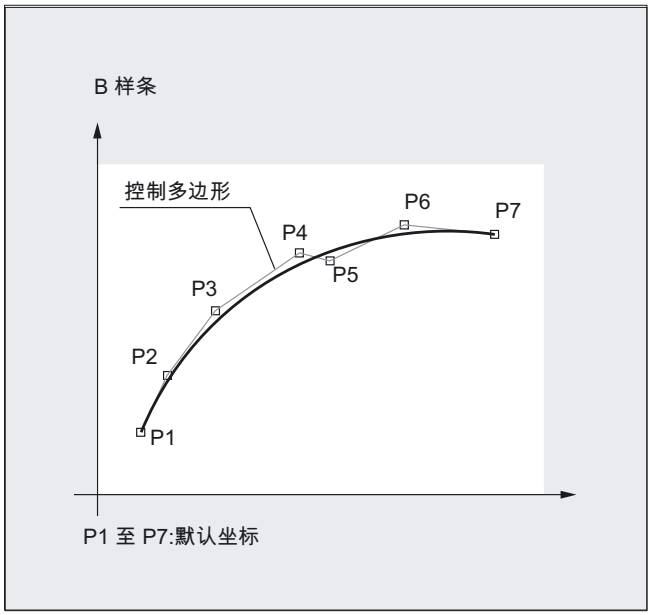
A样条

A样条（Akima样条）精确地通过各个支点。该样条几乎不会产生不好的摆动，但是在各个支点处也不会有曲线过渡。Akima样条为局部型，即节点的变化仅对6个以下相邻的节点起作用。因此它主要适用于数字化点的插补。插补时使用3级多项式。



B 样条

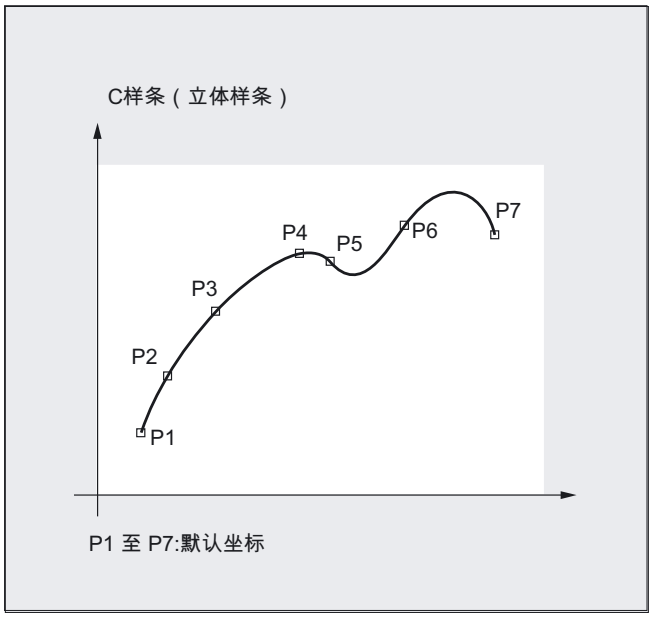
B样条的编程位置不是节点，而仅是样条的检查点，即曲线不直接经过这些点，而是被它们所“吸引”。通过直线连接这些点，从而构成样条的控制多边形。B样条最适用于描述自由加工面上的刀具位移。它主要是考虑与CAD系统的接口。3级B样条不会产生摆动，尽管它有曲线过渡。



C样条

与Akima样条不同的是，立体样条（C样条）在支点处有曲线过渡。但是仍有产生摆动的倾向。如果这些点位于一个已知的曲线上，则可以使用该样条。C样条使用3级多项式。

该样条为局部型，即某个节点的变化可能会在多个程序段中起作用（递减）。

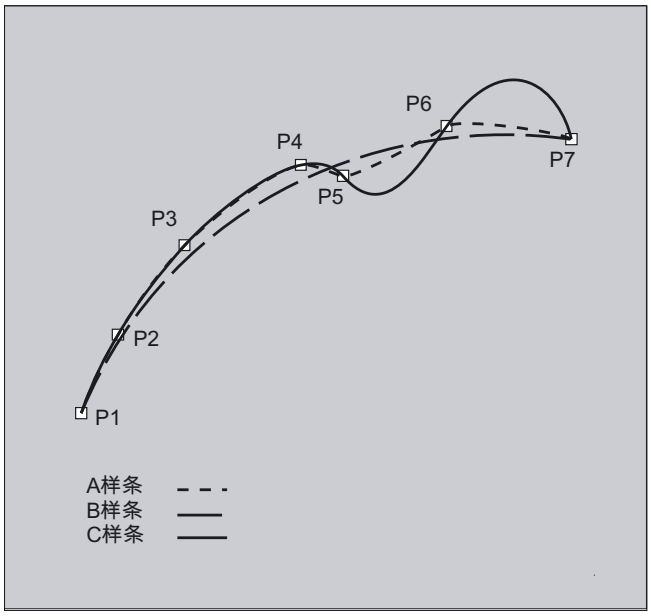


在相同的支点处对比三种类型的样条：

A样条（Akima样条）

样条（Bezier样条）

C样条（立体样条）



样条的调整

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。并非每个程序段信息均为一个样条终点，控制系统还需要从10个程序段中获得一定数量的样条终点程序段。

它们是：

A样条：	从每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：	从每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：	从每10个程序段中必须至少是机床数据$MC_CUBIC_SPLINE_BLOCKS+1的内容为样条程序段（默认情况为9）。 在机床数据$MC_CUBIC_SPLINE_BLOCKS（缺省值8)中输入点数，从而计算出样条区段。

注意

如果超出允许值的范围，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。

5.3 样条组合(SPLINEPATH)

功能

使用指令 Befehl 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。使用 SPLINEPATH-语句来确定样条上的参与轴。

编程

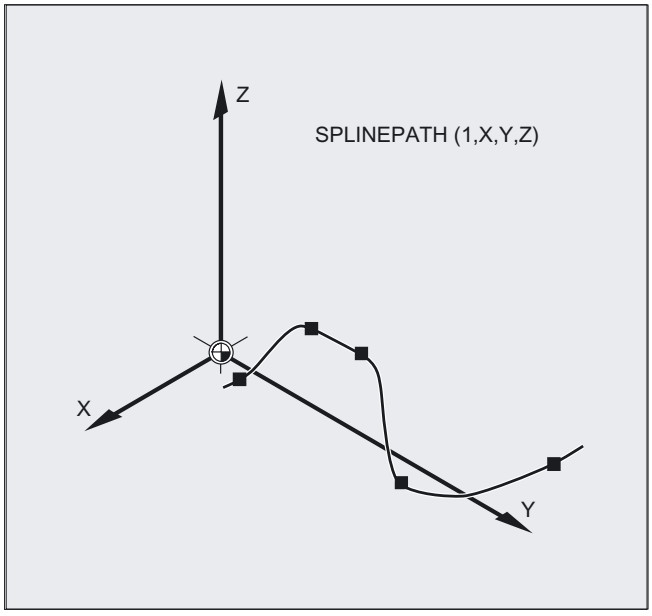
```
SPLINEPATH(n,X,Y,Z,…)
```

这需要在一个独立的程序段中确定。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。

参数

SPLINEPATH	确定样条连接
n = 1	固定值
X,Y,Z,…	说明轨迹轴

带有三个轨迹轴的样条组合举例




```
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                          ;节点
…
N100 G1 X… Y…                                    ;取消样条插补
```

5.4 压缩器 (COMPOF/ON, COMPCURV, COMPCAD)

功能

使用G代码 COMPON ，程序段过渡仅在**速度**方面稳定，同时参与程序段过渡的轴的加速度可能会有跃变。这可能会导致摆动的生成。

使用G代码 COMPCURV 来以**加速度稳定** 的方式设计程序段过渡。这样就可以保证程序段过渡时所有轴的速度和加速度的平滑。使用G代码COMPCAD
可以选择另外一个可以对**表面质量和速度** 进行优化的压缩函数。

机床制造商

压缩函数可以设计且和机床数据设置有关。

编程

COMPON

或者

COMPOF

或者

COMPCURV

或者

COMPCAD

参数

COMPON/ /	启用压缩器，在速度方面稳定
COMPOF	压缩器关
COMPCURV	启用压缩器，曲率稳定的多项式（加速度稳定）
COMPCAD	启用压缩器，优化表面质量（速度优化）

举例 COMPON

```
N10 COMPON                                ;或者COMPCURV，启用压缩器
N11 G1 X0.37 Y2.9 F600                    ;G1 必须在终点和进给之前

N12 X16.87 Y-.698
N13 X16.865 Y-.72
N14 X16.91 Y-.799…
N1037 COMPOF                              ;关闭压缩器
…
```

注意

压缩满足某个简单语法的所有程序段，例如

N19 X0.103 Y0. Z0.

N20 X0.102 Y-0.018

N21 X0.097 Y-0.036

N22 X0.089 Y-0.052

N23 X0.078 Y-0.067

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。

举例 COMPCAD

```
G00 X30 Y6 Z40
G1 F10000 G642
SOFT
COMPCAD                                   ;启用优化表面质量压缩器
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                                    ;关闭压缩器
G00 Z50
M30
```

“用于定向的压缩器”举例

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，但是压缩器却生成一个平滑的曲线。

```
DEF INT ANZAHL = 60                          ;最大偏差
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：.05 mm
$SC_COMPRESS_ORI_TOL = 5                     ;定向的最大偏差：
                                             ;5 度

TRAORI                                       ;跟踪一个从
COMPCURV                                     ;多边形形成的圆。
N100 X0 Y0 A3=0 B3==1                        ;此时在一个围绕Z轴、张角为45度的圆锥上作定向运动
N110 FOR COUNTER = 0 TO ANZAHL               。
N120 WINKEL= 360 * COUNTER / ANZAHL
N130 X=RADIUS*COS(WINKEL)Y=RADIUS*
 SIN(WINKEL) A3=SIN(WINKEL)
 B3=(WINKEL) C3=1
N140 ENDFOR
...
```

前提条件

机床制造商

有3个机床数据用于压缩器功能：

- $MC_COMPRESS_BLOCK_PATH_LIMIT
  设置最大行程长度，一直到可被压缩的程序段长度为止。更长的程序段不被压缩。
- $MA_COMPRESS_POS_TOL
  可以给每个轴设置一个允差。所形成的样条曲线与编程的终点偏离最多为该设定值。允许的公差越大，越多的程序段可以压缩。
- $MC_COMPRESS_VELO_TOL
  在有 FLIN 和 FCUB 配合的情况下，当压缩器激活时可以设定的最大允许偏差。

COMPCAD

- $MN_MM_EXT_PROG_BUFFER_SIZE 应选择较大的值，例如100 (kB)。
- $MC_COMPRESS_BLOCK_PATH_LIMIT 必须指定有明显较大的值，例如 50 (mm)。

- $MC_MM_NUM_BLOCKS_IN_PREP 必须设定成 >= 60，以便可以处理明显多于10个以上的点。
- FLIN 和 FCUB 可以不使用。

建议较大的程序段长度，优化的速度：

- MC_MM_MAX_AXISPOLY_PER_BLOCK = 5
  $MC_MM_PATH_VELO_SEGMENTS = 5
  $MC_MM_ARCLENGTH_SEGMENTS = 10

说明

CAD/CAM系统通常提供线性程序段，它们执行参数化的精度。在轮廓比较复杂时这会导致数据量的大幅提高，并可能造成较短的轨迹区段。这种较短的轨迹区段会限制加工速度。

压缩器会使一定数量的（最大10）的这种较短的轨迹区段合并为一个轨迹区段。

使用模态G代码 COMPON 以及 COMPCURV 可以启用一个 "NC程序段压缩器"。使用此功能，在线性插补时汇聚了一系列直线程序段（数量限制为10），并且在由机床数据设定的误差公差范围内近似于3级多项式(COMPON) 或者 5级多项式 (COMPCURV)。取代许多较小的程序段，而是加工一个较大的运行程序段。

使用条件

压缩过程仅在线性程序段（G1）中执行。可通过任意一个其它NC指令中断该过程，例如辅助函数输出，单并不能通过参数计算中断。仅包含程序段号、G1、轴地址、进给和注释的程序段才可以进行压缩。所有其它的程序段按原样加工（没有压缩）。变量不可以使用。

COMPCAD

COMPCAD可以压缩计算时间和存储空间。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。特征：

- COMPCAD指令产生加速度恒定的、过渡的多项式程序段。
- 相邻的轨迹中在同一个方向偏离。
- 使用调整数据 $SC_CRIT_SPLINE_ANGLE 可以确定一个极限角，COMPCAD 运行大于此角的角部存在。
- 压缩的程序段的个数没有限制为10。
- COMPCAD消除有缺陷的表面过渡。在此继续执行公差范围，但是没有考虑棱角临界角。
- 还可以附带使用修光函数 G642 。

COMPON, COMPCURV 和 COMPCAD 扩展

对压缩器 COMPON, COMPCURV 和 COMPCAD 进行扩展，使得在其中已借助方向矢量编程有定向的NC程序也能在遵守某个可设定允差的情况下被压缩。

定向转换TRAORI

对于“用于定向的压缩器”函数而言，必须有定向转换选项可供使用。前面在“使用条件”下所说的限制放宽，即位置值在此也可以通过参数赋值进行。

NC程序段的通常形式：

```
N10 G1 X=<...>Y=<...>Z=<...>A=<...>      ;轴位置，作为参数表达式，带有
     B=<...>F=<...>; 注释                 < ...> 参数表达式例如 X=R1*(R2+R3)
```

带有刀具定向的五轴机床

在方向转换有效时（TRAORI），在5轴的机床中可以给刀具方向进行如下编程（与运动无关）：

1. 通过以下参数编程方向矢量：A3=< ...>B3=< ...> C3=< ... >
2. 通过以下参数编程欧拉角或者RPY角：A2=< ...>B2=< ...> C2=< ... >

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：

1. MD 21104:ORI_IPO_WITH_G_CODE = FALSE, 当 ORIWKS 已激活，且已将定向作为矢量编程时 (使用 A3, B3, C3 bzw. A2, B2, C2).
2. MD 21104:ORI_IPO_WITH_G_CODE = TRUE, 当 ORIVECT 或者 ORIPLANE 有效时。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。如果已有某一个 GORICONxx 或者 ORICURVE 激活或者已编程有定向角 (PO[PHI] 和 PO[PSI]) 的多项式，就不会执行大圆插补，也就是说，此类程序段不会被压缩。

带有可旋转刀具的六轴机床

如果是6 机床，除了刀具定向之外，还可以对刀具的旋转进行编程。转角用名称THETA (THETA=<...>)编程。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：

```
N... X=<...> Y=<...> Z=<...> A3=<...>
     B3=<...> C3=<...> THETA=<...> F=<...>
```

或者

```
N... X=<...> Y=<...> Z=<...> A2=<...>
     B2=<...> C2=<...> THETA=<...> F=<...>
```

当通过旋转轴位置来说明刀具定向时，例如形式为：

```
N... X=<...> Y=<...> Z=<...> A=<...>
     B=<...> THETA=<...> F=<...>
```

则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，通过旋转轴的轴向多项式来描述已压缩的定向变化。

精度

只有当允许轮廓与所编程的轮廓有偏差时，才可对NC程序段进行压缩。最大的偏差可以作为压缩器公差在设定数据中设定。允许的公差越大，越多的程序段可以压缩。

轴精度

压缩器为每个轴生成一个样条，该样条最多以使用轴向MD所设置的误差值来偏离已编程的每个轴的终点。

轮廓精度

对轮廓（几何）和刀具定向的最大公制偏差进行检查。通过以下的设定参数进行：

1. 轮廓的最大允差
2. 刀具定向的最大角度偏差
3. 刀具旋转角的最大角偏差（仅当6机床可用）

使用通道专用的机床参数MD20482 COMPRESSOR_MODE可以设定公差值。

0: 轴精度：轴精度所有轴的轴向允差（几何轴和定向轴）。

1: 轮廓精度：规定轮廓允差 (1.), 定向允差大于轴向允差 (a.).

2: 规定刀具定向的最大角偏差 (2.), 轮廓允差大于轴向允差 (a.).

3: 用（1.）规定轮廓公差，用（2.）规定刀具方向的最大角度偏差。

只有当方向转换(TRAORI)有效时，才可以规定刀具方向的最大角度偏差。

激活

使用某个指令激活“定向压缩器”：COMPON, COMPCURV (COMPCAD 不可用).

文献： /FB3/, F2:“3 - 5轴转换”。

5.5 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产生的样条曲线的接口。在此样条区段可以直接编程。

这种插补方式使NC不用计算多项式系数。当系数直接从CAD系统或者后置处理程序中产生时，可以优化使用。

编程

POLY PO[X]=(x_e,a_2,a_3) PO[Y]=(y_e,b_2,b_3) PO[Z]=(z_e,c_2,c_3) PL=n 三阶多项式

或者扩展成五阶多项式和新的多项式语法

POLY X=PO(x_e,a_2,a_3,a_4,a_5) Y=PO(y_e,b_2,b_3,b_4,b_5) Z=PO(z_e,c_2,c_3,c_4,c_5) PL=n

POLYPATH ("AXES", VECT")

参数

POLY	接通带POLY的程序段的多项式插补。
POLYPATH	多项式插补，可选用于两个轴组AXIS或者VECT
PO [轴标识符/变量]=(…,…,…)	终点和多项式系数
X, Y, Z	轴名称
x_e, y_e, z_e	终点位置参数说明，用于各个轴；数值范围同位移尺寸
a_2, a_3, a_4, a_5	系数$_2$, a_3, a_4, 和 a_5 使用其值写入； 值范围如同位移。如果最后的系数值为零，则可以取消。
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。理论值范围 PL:0,0001 … 99 999,9999. PL值用于它所在的程序段。 如果没有编程PL，则 PL=1.

启用/关闭POLY

多项式插补与G0,G1,G2,G3,A样条,B样条和C样条一起在第一个G组中。当它有效时，不需要编程多项式语句：仅使用其名称和终点编程的轴以直线运行到其终点。

当所有的轴都如此编程时，控制系统如同G1时一样。

通过G组的另外一个指令（例如 G0, G1) 来关闭多项式插补。

多项式系数

PO-值 (PO[]=) 或者...=PO(...)
用来说明某个轴的所有多项式系数。几个值可以根据多项式等级通过逗号分开。在一个程序段中，可以有不同的多项式等级用于不同的轴。

带有PO的新多项式语法：以前的语法继续有效。

调用子程序 POLYPATH

使用POLYPATH可以选择轴组对多项式插补进行说明：

- POLYPATH ("AXES")
  所有轨迹轴和辅助轴。
- POLYPATH ("VECT") 定向轴
  (当定向转换时).

默认情况下两个轴组的已编程多项式也作为多项式插补。

举例：

```
POLYPATH ("VECT")
```

仅选择方向轴用于多项式插补，所有其它的轴以直线运行。

```
POLYPATH ( )
```

取消所有轴的多项式插补

举例

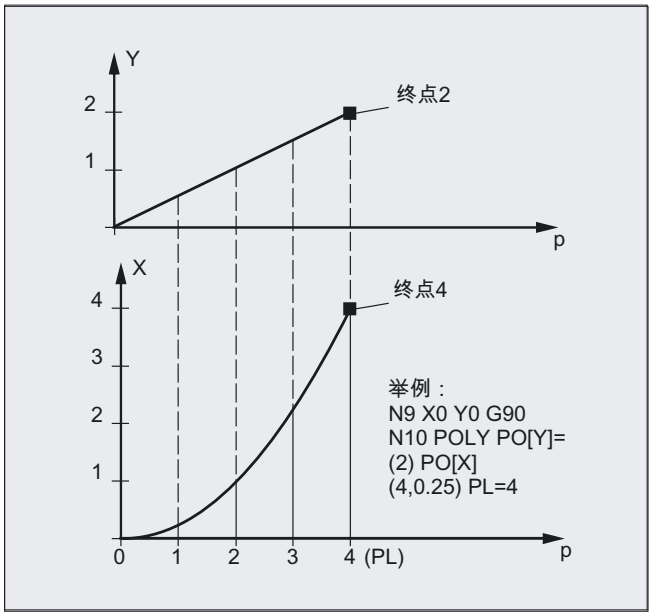
```
N10 G1 X… Y… Z… F600
N11 POLY PO[X]=(1,2.5,0.7) ->                ;启用多项式插补
-> PO[Y]=(0.3,1,3.2) PL=1.5
N12 PO[X]=(0,2.5,1.7) PO[Y]=(2.3,1.7) PL=3
…
N20 M8 H126 …
N25 X70 PO[Y]=(9.3,1,7.67) PL=5              ;轴的混合说明
N27 PO[X]=(10,2.5) PO[Y]=(2.3)               ;没有编程 PL ; 则 PL=1
N30 G1 X… Y… Z.                              ;关闭多项式插补
…
```

带有PO的有效多项式语法举例

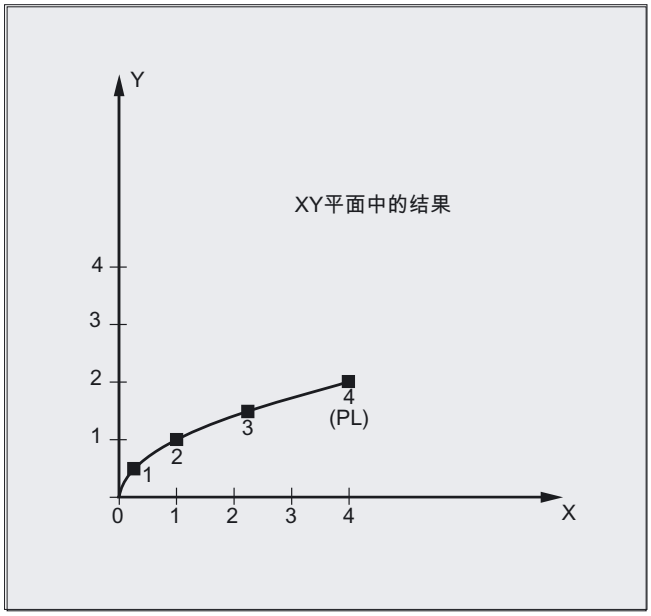
```
到目前为止有效的多项式句法仍然有效。          ;新多项式语法（SW 6 以上）
PO[轴名称]=(.. , ..)                         ;轴标识符=PO(.. , ..)
PO[PHI]=(.. , ..)                            ;PHI=PO(.. , ..)
```

```
PO[PSI]=(.. , ..)                    ;PSI=PO(.. , ..)
PO[THT]=(.. , ..)                    ;THT=PO(.. , ..)
PO[]=(.. , ..)                       ;PO(.. , ..)
PO[变量]=IC(.. , ..)                 ;变量=PO IC(.. , ..)
```

X/Y-平面中的曲线举例




```
N9 X0 Y0 G90 F100
N10 POLY PO[Y]=(2) PO[X]=(4,0.25) PL=4
```



说明

控制系统可以执行任何所选轨迹轴均跟踪某个函数（多项式，最大三阶）或者（多项式，最大五阶）的曲线运行（轨迹）。

多项式功能的一般形式为：

$f(p)= a_0 + a_1p + a_2p^2 + a_3p^3$

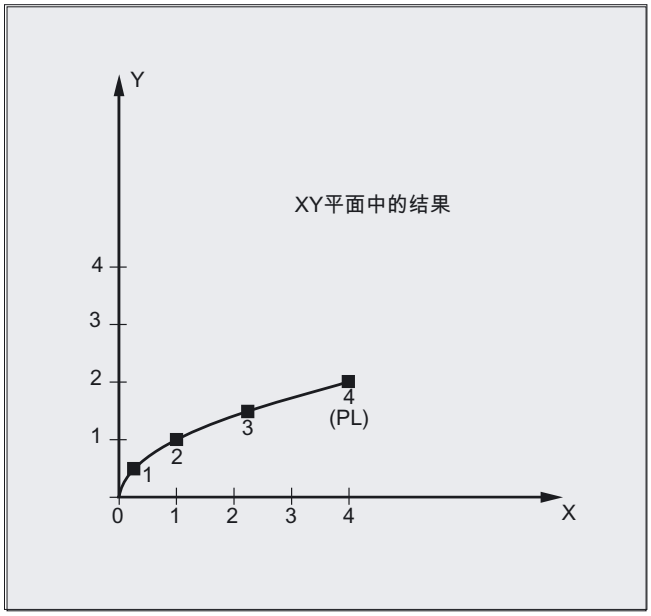
或者

$f(p)= a_0 + a_1p + a_2p^2 + a_3p^3 + a_4p^4 + a_5p^5$

这表明：

a_n:系数常量

p:参数



通过给系数设定具体的数值，可以产生不同的曲线如直线、抛物线和幂函数。

当设定系数 $a_2 = a_3 = 0$ 或者 $a_2 = a_3 = a_4 = a_5 = 0$ 时，例如可得出一条直线，

$f(p) = a_0 + a_1 p$

适用于：

a_0 = 正在执行的程序段末尾的轴位置

a_1 = 定义范围（PL）终点上的轴位置和起始位置之间的差

可以在**没有** 激活G代码POLY的情况下，对多项式进行编程。在这种情况下，并不是插补编程的多项式，而是每次以线性方式返回到每个轴的编程终点（G1）。然后通过编程POLY激活多项式插补。

此外，在激活G代码 POLY的情况下，还可以使用预定义的子程序 POLYPATH (...) 来选择要对哪个轴进行多项式插补。

分母多项式特点

对于几何轴而言，在没有说明某个轴名称的情况下，也可以使用 PO[]=(…) 来编程一个共同的分母多项式，也就是说，将几何轴的运动作为两个多项式的商进行插补。

例如，这样可精确描述圆锥截面（圆，椭圆，抛物线，双曲线）

举例：

```
POLY G90 X10 Y0 F100                    ;运动几何轴
                                        ;以线性方式朝向位置 X10, Y0
PO[X]=(0,-) PO[Y]=(10) PO[]=(2,1)       ;运动几何轴，;在四分之一圆中朝向 X0, Y10
```

分母多项式的常量系数 (a_0) 始终假设为1，所说明的终点与 G90/G91 无关。

由上面的示例可以得到下面的结果：

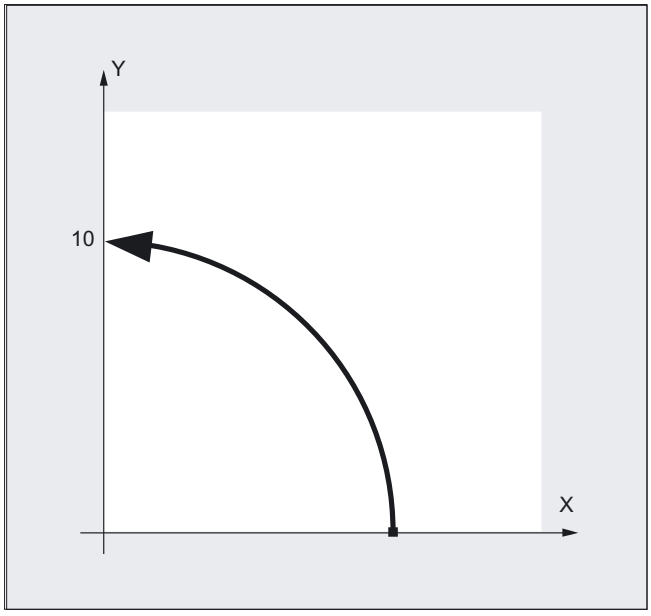
X(p)=10(1)/(1+p2) 和 Y(p)=20p/(1+p2) 且 0<=p<=1

根据已编程的起始点、终点、系数 a_2 和 PL=1，得出下列平均值：

分子 (X)=10+0*p–p2

分子 (Y)=0+20*p+0*p2

分母 = 1+2*p+1*p2



打开多项式插补时，在间隔[0,PL]内带零点的除数多项式的编程被拒绝，并发出报警。分母多项式多辅助轴的运动没有影响。

注意

在有G41，G42的多项式插补中可以接通刀具半径补偿，并且可以用于直线或者圆弧插补。

5.6 可设置的轨迹基准 (SPATH, UPATH)

功能

在多项式插补过程中，用户可能希望在决定速度的FGROUP轴和其余轨迹轴之间有两种不同的关系：剩余的轨迹轴应该是：

- 或者同步于FGROUP轴的轨迹位移，
- 或者同步于曲线参数

对于FGROUP中不包含的轴有两种方法跟随轨迹：

1. 可以与位移S（SPATH）同步
2. 或者与FGROUP轴（UPATH）的曲线参数U同步

两种轨迹插补方式可以在不同的场合使用，并且可以通过G指令SPATH和UPATH进行切换。

编程

SPATH

或者

UPATH

参数

SPATH	FGROUP轴的轨迹基准为弧长。
UPATH	FGROUP轴的轨迹基准为曲线参数。
FGROUP	确定带有轨迹进给的轴

SPATH, UPATH

通过两个G代码中的一个 (SPATH, UPATH) 可以选择和编程所需的特性。

这些指令模态有效。当SPATH有效时轴同步于位移，当UPATH有效时则同步于曲线参数。

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。

激活 FGROUP

不包含在 FGROUP 中的轴的轨迹基准可通过两个在 45. G-代码组中所包含的语言指令 SPATH 和 UPATH 进行设置。

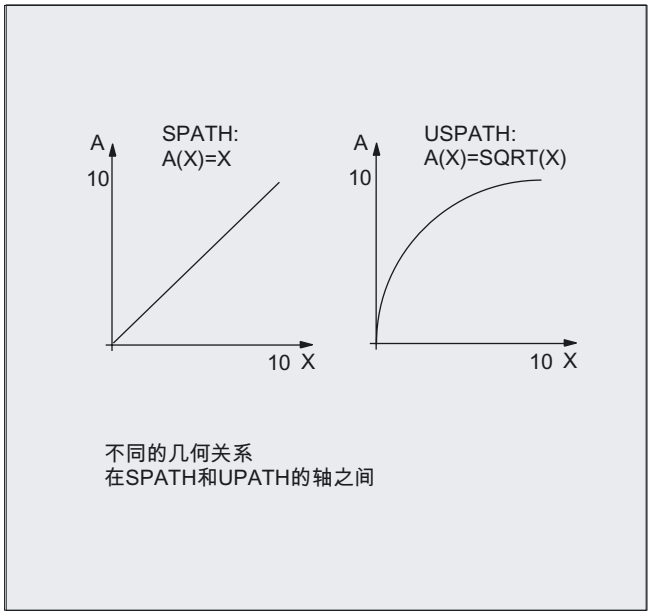
举例1

在下列示例中，使用G643对边长为20mm的正方形进行修光。此时通过机床数据 MD 33100:COMPRESS_POS_TOL[...] 为每个轴确定精确轮廓的最大偏差。

```
N10 G1 X… Y… Z… F500
N20 G643                                   ;程序段内部带有 G643 的修光
N30 XO Y0
N40 X20 Y0                                 ;mm 边长，用于轴
N50 X20 Y20
N60 X0 Y20
N70 X0 Y0
N100 M30
```

举例2

下列示例用来阐明两种运动控制之间的区别。预设定FGROUP(X,Y,Z)两次有效。



不同的几何关系
在SPATH和UPATH的轴之间

```
N10 G1 X0 A0 F1000 SPATH
N20 POLY PO[X]=(10, 10) A10
或者
N10 G1 X0 F1000 UPATH
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，沿着X轴的位移得出同步轴的不同位置，视情况而定，是否 SPATH 或者 UPATH 已激活。

边界条件

所设定的轨迹基准在下面情况下没有意义：

- 线性插补和圆插补
- 在螺纹程序段和
- 当所有的轨迹轴均包含在FGROUP中时

说明

在多项式插补过程中 - 且因此始终为多项式插补

- 严格来说 (POLY),
- 所有样条插补类型 (ASPLINE, BSPLINE, CSPLINE) 和
- 带有压缩器的线性插补 (COMPON, COMPCURV)

- 所有轨迹轴 i 的位置均通过多项式 pi(U) 来设定。曲线参数U在一个NC程序段之内从0运行到1，也经过标准化。

通过语言指令 FGROUP 可以在轨迹轴之内选择那些已编程轨迹进给以此为参照的轴。
用速度常数在该轴的位移S中进行插补，表示在多项式插补时进行一个非常数的曲线参数U的修改。

复位时的控制特性和机床数据/选型数据

在复位之后，通过MD 20150:GCODE_RESET_VALUES [44] 所决定的G代码有效 (45. G-代码组).

这类修光的基本位置值可使用MD 20150:GCODE_RESET_VALUES [9] 来确定 (10. G-代码组).

在复位之后激活的G代码组的值可通过机床数据MD 20150:GCODE_RESET_VALUES [44] 确定。为了保持与当前设备的兼容性，这里预先设定SPATH作为标准值。

轴向机床数据MD 33100:COMPRESS_POS_TOL具有一种扩展含义：它含有压缩器功能的公差和用G642进行精磨削的公差。

5.7 用接触式探头测量 (MEAS, MEAW)

功能

对于所有在NC程序段中编程的轴而言，其位置信号在出现测头的脉冲沿时采集，并且将每个轴的位置写入到相应的存储单元中。最多存在两个测头。

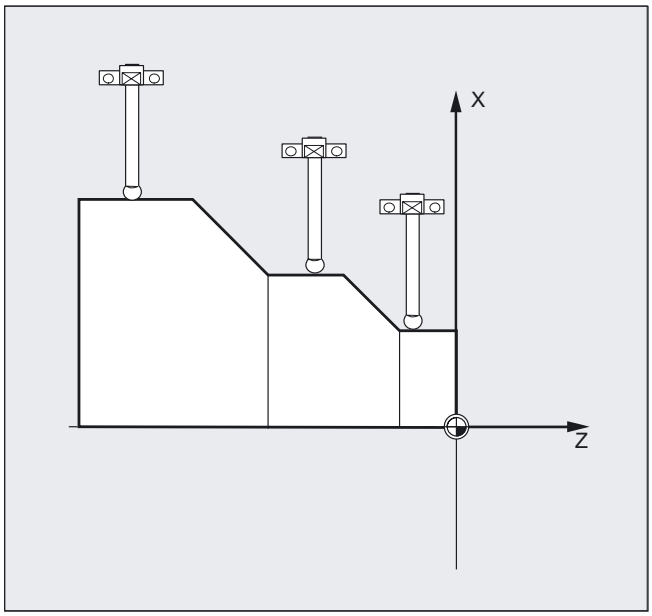
读取测量结果

使用测头所检测的轴的测量结果在下列变量项下可供使用：

- 在机床坐标系中在 $AA_MM[轴] 项下
- 在工件坐标系中在 $AA_MW[轴] 项下

在读这些变量时内部没有进刀停止。

必须在NC程序使用 STOPRE在适当的位置商编程一个进给停止。否则会读入错误的值。



编程

编程测量程序段, MEAS, MEAW

使用指令 MEAS **和某一种插补类型** 来逼近工件上的实际位置，且同时接收测量值。
在实际位置和给定位置之间的剩余行程删除。

对于在任何情况下均要逼近已编程的位置的特殊测量任务而言， 使用 MEAW MEAS 和 MEAW 为逐段有效。

```
MEAS=±1          G... X... Y... Z...
MEAS=±2          G... X... Y... Z...

MEAW=±1          G... X... Y... Z...
MEAW=±2          G... X... Y... Z...
```

(+1/+2 测量，带剩余行程删除和上升沿)
(–/–使用删除剩余行程和下降沿来测量)
(+1/+2 测量，不带剩余行程删除和上升沿)
(–/–不使用删除剩余行程和下降沿来测量)

参数

MEAS=±1	使用测量输入端1上的探头1测量
MEAS=±2*	使用测量输入端2上的探头2测量
MEAW=±1	使用测量输入端1上的探头1测量
MEAW=±2*	使用测量输入端2上的探头2测量
G...	插补类型，例如 G0, G1, G2 或者 G3
X... Y... Z...	直角坐标的终点

*视扩充等级而定，最多2个输入端

编程测量程序段举例

MEAS 和 MEAW 被编程在带有运动语句的程序段中。进给和插补类型 (G0, G1, ...) 要与相应的测量问题适配; 轴的数量也是如此。

```
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。进刀停止自动产生。

说明

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)。：

0 测量任务未履行

1 测量任务已顺利结束（测头探头已工作）

注意

当测头在程序中偏转时，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）。如果是MEAS，就在测头工作之后按照定义使运动停止。

注意

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。

5.8 扩展测量函数 (MEASA, MEAWA, MEAC) (选项)

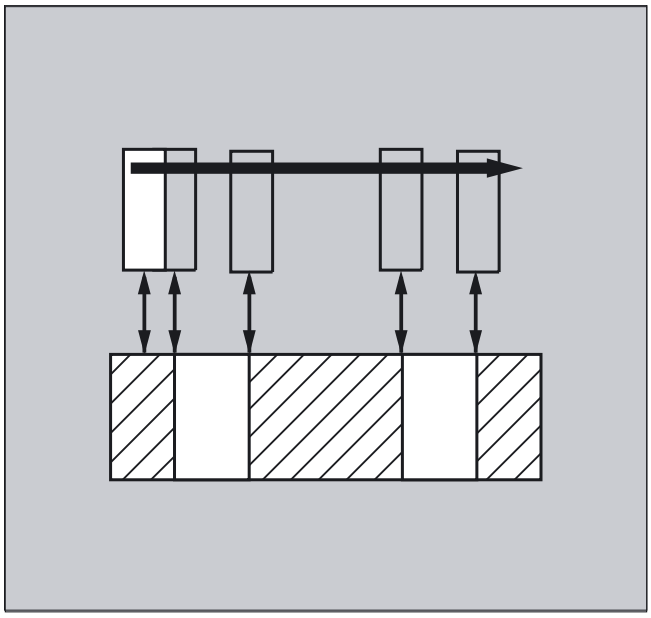
功能

如果是轴向测量，可以使用多个测头和多个测量系统。

如果是MEASA, MEAWA，每次测量时为相应已编程的轴采集四个以下的测量值，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：

- 在机床坐标系中，在 $AA_MM1 ~ 4[轴]项下
- 坐在工件坐标系中，在 $AA_WM1 ~ 4[轴]项下



编程

MEASA 和 MEAWA 为逐段有效且可以在某个程序段中编程。如果 MEASA/MEAWA 与 MEAS/MEAW 编程在一个程序段中，就会发出出错信息。

MEASA[轴]=(模式, TE1,..., TE4)

或者

MEAWA[轴]=(模式, TE1,..., TE4)

或者

MEAC[轴]=(模式, 测量存储器, TE1,..., TE4)

参数

MEASA	测量，带剩余行程删除
MEAWA	测量，不带剩余行程删除
MEAC	连续测量，不带剩余行程删除
轴	名称，用于测量所使用的通道轴

模型	运行模态的两位参数说明；由以下构成： **测量模式**（十进制个位） 和 0：模式**0**：取消测量任务 1：模式**1**：4个以下可以**同时** 激活的触发事件 2：模式**2**：4个以下可以**依次** 激活的触发事件 3：模式**3**：4个以下可以**依次** 激活的触发事件 但是不监控触发事件1 当START时 （报警 21700/21703 被抑制） 说明：在MEAC时不可能有模态 3 **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 1…4	**触发事件** 1：上升脉冲沿，测头1 -1：下降脉冲沿，测头1 2：上升脉冲沿，测头2 -2：下降脉冲沿，测头2
测量存储器	FIFO号（循环存储器）

模式1中带有剩余行程删除的测量举例

（按时间顺序进行处理）

a) 有1个测量系统

```
...
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
                                            ;用测头1的上升沿/下降沿，在
                                            ;X = 100 后面的运动行程上。
N110 STOPRE                                 ;进刀停止
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:
```

模式1中带有剩余行程删除的测量举例

b) 有2个测量系统

```
...
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
                                            ;用测头1的上升沿/下降沿，在
                                            ;X = 100 后面的运动行程上。
N210 STOPRE                                 ;进刀停止
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:
```

模式2中带有剩余行程删除的测量举例

（按编程顺序进行处理）

```
...
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。等候测量信
                                              号，顺序为测头1的上升沿，测头1的下降沿，
                                              ;测头2的上升沿，测头2的下降沿，
                                              ;在
                                              ;X = 100 后面的运动行程上。
N110 STOPRE                                   ;进刀停止
N120 IF $AC_MEA[1] == FALSE gotof             ;使用测头1检查测量结果
                                              。
测头2
N130 R10 = $AA_MM1[X]                         ;保存属于第一个已编程
                                              ;触发事件
                                              ;(测头1上升沿)
                                              ;的测量值。
N140 R11 = $AA_MM2[X]                         ;保存属于第二个已编程
                                              ;触发事件
                                              ;(测头1上升沿)
                                              ;的测量值。
N150 MESSTASTER2:
N160 IF $AC_MEA[2] == FALSE gotof ENDE        ;使用测头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
                                              ;(测头2上升沿)
                                              ;的测量值。
N190 ENDE:
```

在模式1中连续测量举例

（按时间顺序进行处理）

a) 测量100个以下的测量值

```
...
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, 等待测头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
```

在模式1中连续测量举例

（按时间顺序进行处理）

b) 在10个测量值之后使用剩余行程删除来测量

```
...
N10 WHEN $AC_FIFO1[4]>=10 DO                 ;删除剩余行程
 MEAC[x]=(0)_DELDTG (x)
N20 MEAC[x]=(1,1,1,-1) G01 X100 F500
N30 MEAC [X]=(0)
N40 R1=$AC_FIFO1[4]                          ;测量值的数量
...
```

说明

可以在零件程序中**或者** 从某个同步动作中（参见“运动同步动作”一章）进行编程。某一时刻和同一时刻每个轴只能有一个测量任务激活。

注意

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调节器节拍的四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，那么插补节拍和位置调节器节拍之间的比例不得大于8：1。

触发事件

一个触发事件由测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，可以分别处理四个以下的已响应测头的触发事件，即最多两个测头，每个测头分别有两个测量脉冲沿。处理的顺序和触发事件的最大个数与所选的模态有关。

注意

同样的触发事件仅可以在测量任务中编程一次（仅适用于模态1）。

工作模态

使用模态的第一个数字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、**测量模式，** 将测量过程与相应控制系统的可用测量方法进行适配：

- **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移植到第二个模态（没有通报）。
- **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。

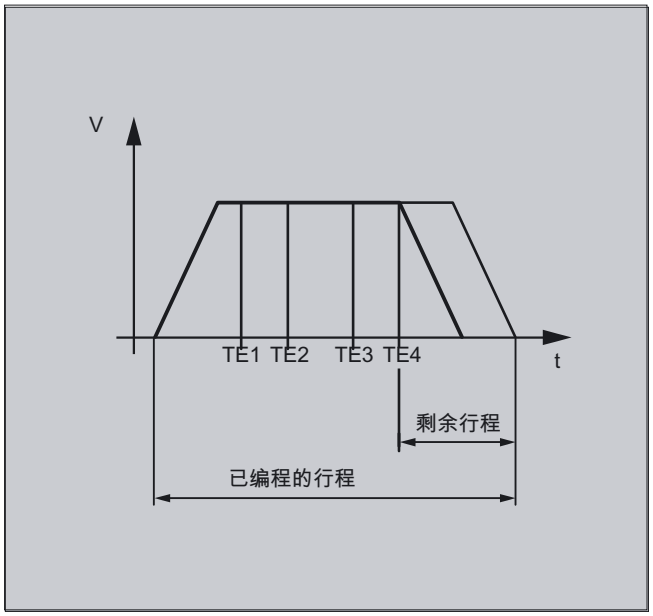
注意

如果使用两个测量系统，就只可编程两个触发事件。

有和没有剩余行程删除的测量，MEASA, MEAWA

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 。



- 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

注意

在读这些变量时内部没有进刀停止。使用 STOPRE (“指令列表”一章) 必须在适当的位置上编程一个进给停止。否则会读入错误的值。

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。

举例：

```
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2时测量系统1的测量值
$AA_MM3[轴]	或者	$AA_MW3[轴]	当出现触发事件1时测量系统2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

可通过 $A_PROBE[n] 读取测头状态

n=测头

1==测头已偏转

0==测头未偏转

当 MEASA, MEAWA 时的测头状态

如果有必要在程序中进行分析，可以通过 $AC_MEA[n]（n = 测头的编号）来查询测量任务状态。只要在某个程序段中已编程的测头“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。

注意

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DB(31-48) DBB62 位 3 或者等效变量 $AA_MEAACT["轴"]。

意义：

$AA_MEAACT==1:测量有效

$AA_MEAACT==0:测量未激活

文献：函数说明 /FB/ M5, 测量

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）。如果设计了两个测头用来进行测量，就会将第二个测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“运动同步动作”一章。

注意

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，
  例如：
  ```
  N01 MEAS=1 MEASA[X]=(1,1) G01 F100 POS[X]=100
  ```
- MEASA/MEAWA 参数个数 <2 或者 >5
  例如：
  ```
  N01 MEAWA[X]=(1) G01 F100 POS[X]=100
  ```
- MEASA/MEAWA 触发事件不等于 1/ -1/ 2/ -2
  例如：
  ```
  N01 MEASA[B]=(1,1,3) B100
  ```
- MEASA/MEAWA 模式错误
  例如：
  ```
  N01 MEAWA[B]=(4,1) B100
  ```

- MEASA/MEAWA 重复编程的触发事件
  例如：
  N01 MEASA[B]=(1,1,-1,2,-1) B100
- MEASA/MEAWA 且缺少几何轴
  例如：
  N01 MEASA[X]=(1,1) MESA[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
  例如：
  N01 MEASA[X]=(1,1) MEASA[Y]=(1,1) MEASA[Z]=(1,1,2)
  G01 X50 Y50 Z50 F100

5.9 适用于OEM用户的专用函数 (OEMIPO1, OEMIPO2, G810 至 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。

5.10 带有角部减速的进给减速 (FENDNORM, G62, G621)

功能

在自动拐角延迟时，在距离拐角很近时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。

注意

该函数功能不在 SINUMERK 的标准供货范围之内，必须激活有关软件版本才能使用。

文献：/FBA/ 函数说明 ISO-方言

编程

```
FENDNORM
G62 G41
```

或者

```
G621
```

参数

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所规定的轴的每个角部

5.11 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.

编程

FINEA[轴]

或者

COARSEA[轴]

或者

IPOENDA[轴]

或者

IPOBRKA(轴,[, [百分比值]]) 可以多次说明

或者

ADISPOSA(轴, [整数][, [实数]]) 可以多时说明

参数

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	在到达"插补器"时运动结束
IPOBRKA	在制动斜坡中可以更换程序段（自软件版本SW6.2起）
ADISPOSA	相对于运动结束条件的允差范围值（6.4以上版本的软件）
轴	通道轴名称 (X, Y, ...)

百分比数值	在与制动斜坡有关时，程序段转换应以%进行。
Int	Mode 0:公差窗口无效
	Mode 1:公差窗口与给定位置相关
	Mode 2:公差窗口与实际位置相关
实数	公差窗口大小。在执行主过程的同时将该值记录到调整数据 43610:ADISPOSA_VALUE 中

当到达插补器停止时结束运动举例

```
...
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X!00，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。
...
```

零件程序中制动斜坡程序段转换条件举例

```
          ；缺省设定生效
N40 POS[X]=100
          ;当X轴已到达位置100和精准停时，进行程序段转换

N20 IPOBRKA(X,100)      ；激活制动斜坡程序段转换准则
N30 POS[X]=200          ；一旦X轴开始制动，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，即开始进行程序段转换

N50 POS[X]=0            ;X轴制动且向位置0返回
                        ;在到达位置0和精准停时进行程序段转换
N60 X10 F100
N70 M30
...
```

同步动作中制动斜坡程序段转换条件举例

```
在工艺循环中：
FINEA                   ；运动结束准则精准停
```

```
POS[X]=100              ;当X轴已到达位置100和精准停时，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准则
POS[X]=100              ;POS[X]=100; 只要X轴开始制动，就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，就在工艺循环中进行程序段转换
POS[X]=250              ;X轴制动且向位置0返回
                        ;在到达位置0和精准停时进行程序段转换
M17
```

说明

系统变量 $AA_MOTEND

可以使用系统变量 $AA_MOTEND[轴] 对已经设置好的运动结束条件进行查询。

$AA_MOTEND[轴]=1	使用“精准停”结束运动
$AA_MOTEND[轴]=2	使用“粗准停”结束运动
$AA_MOTEND[轴]=3	使用“IPO-Stop”结束运动
$AA_MOTEND[轴]=4	轴运动的制动斜坡程序段转换条件
$AA_MOTEND[轴]=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
$AA_MOTEND[轴]=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

注意

在复位之后最后编程的值仍存在。

参考文献：
函数说明 /FB1/, V1 进给

在制动斜坡中程序段更换准则

百分比值在执行主过程的同时被记录在SD 43600:IPOBRAKE_BLOCK_EXCHANGE 中。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允差范围。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**且**

- 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。

有关定位轴程序段转换准则的其它信息参见：

参考文献：
函数说明 /FB2/, P2 定位轴
编程说明基本部分 /PG/, 进给调节和主轴运动

5.12 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。

编程

SCPARA[轴]=值

参数

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)

注意

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或者G332时最适合的参数组由控制系统选择。

如果要在某个零件程序中或者某个同步动作和PLC中**转换伺服参数程序段**，就必须对PLC用户程序进行扩展。

参考文献：
函数说明 /FB1/, V1 "进给干预".

举例

```
...
N110 SCPARA[X]= 3              ;将第三个参数程序段选择给轴X
...
```

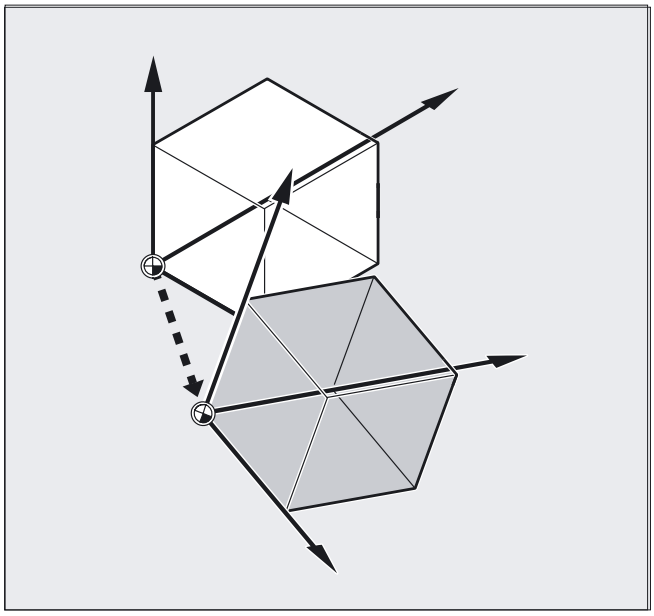
框架

6

6.1 通过框架变量转换坐标

功能

除了在编程说明“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。

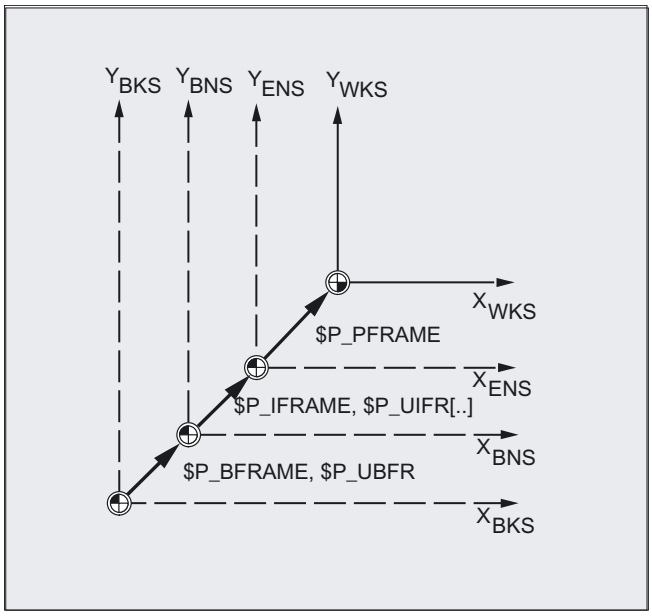
举例：`$P_PFRAME=CTRANS(X,10)`

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

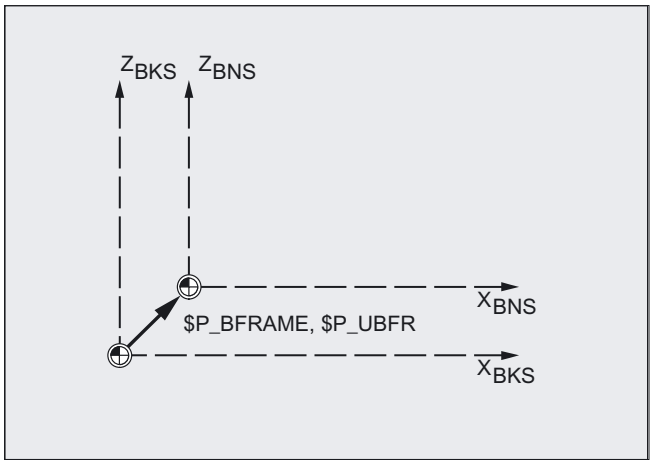
6.1.1 预定义框架变量 ($P_BFRAME, $P_IFRAME, $P_PFRAME, $P_ACTFRAME)

$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

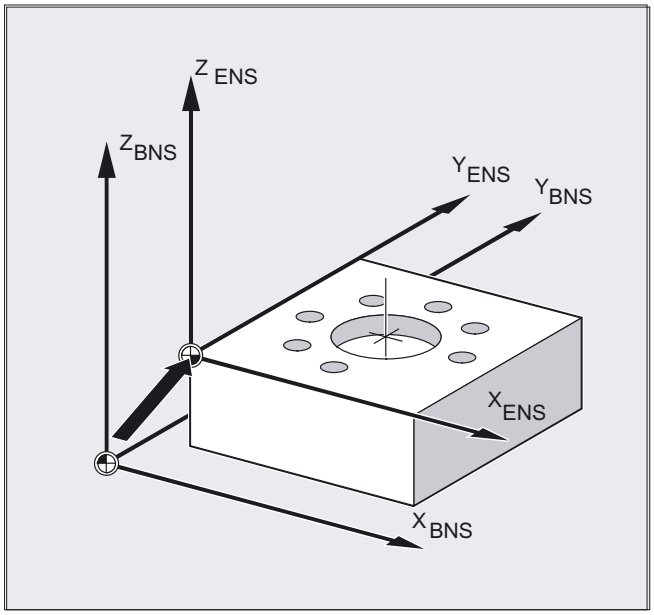
- 编程一个 G500, G54...G599 或者
- 写入$P_BFRAME 与$ $P_UBFR 。



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。

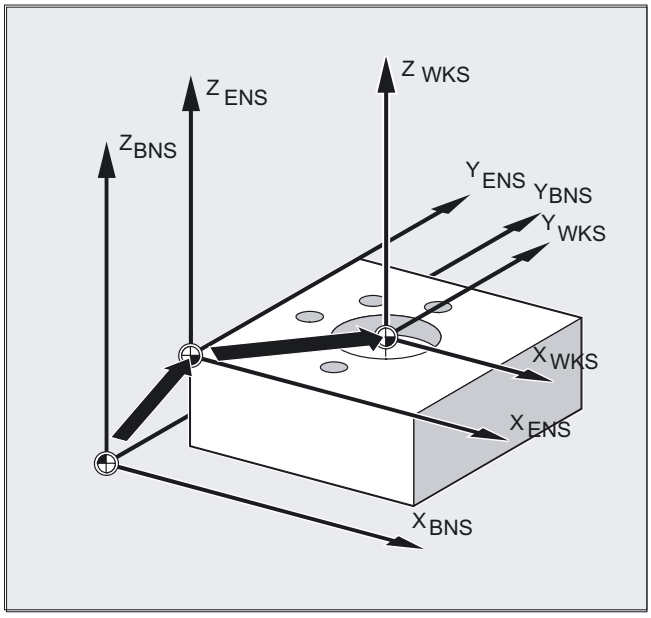


$P_PFRAME

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。

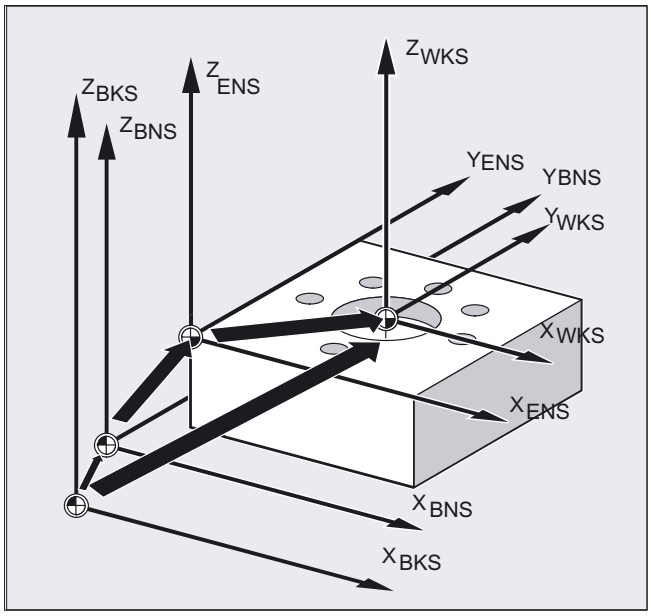


$P_ACTFRAME

通过级联从

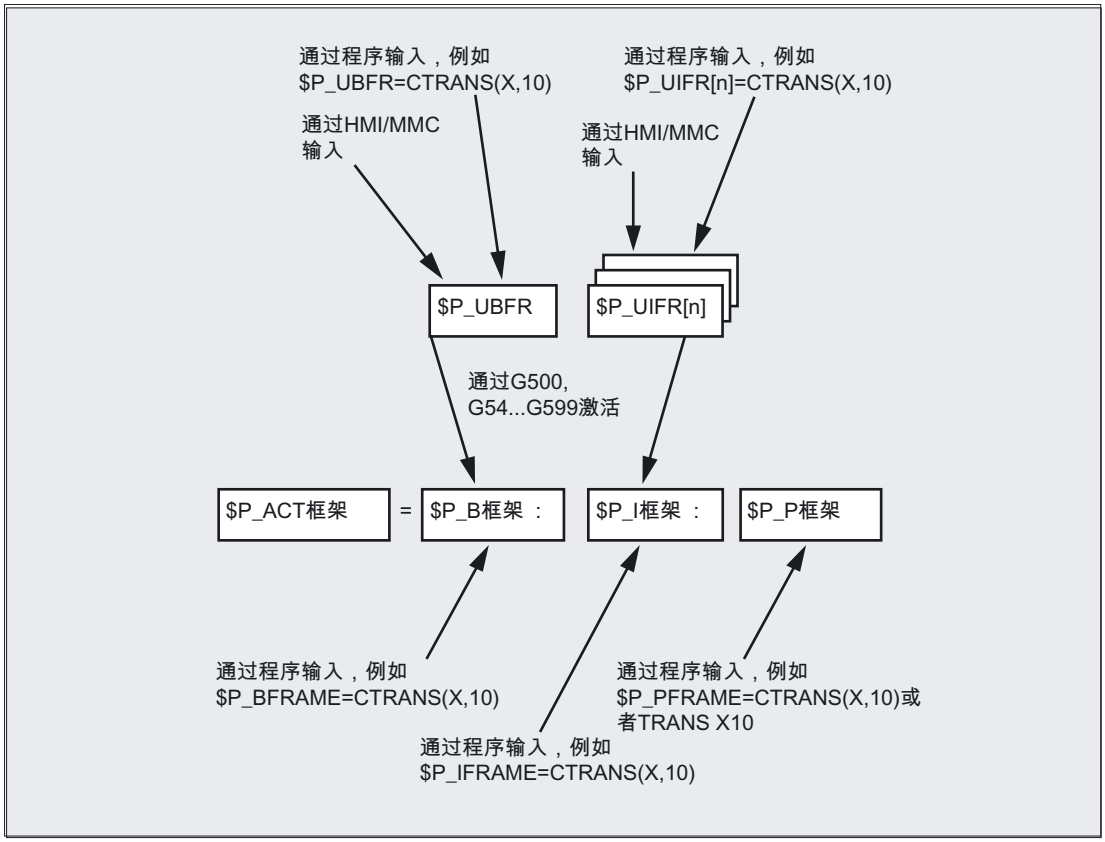
- 当前基本框架变量 $P_BFRAME,
- 当前的可设置框架变量 $P_IFRAME 与系统框架和
- 当前的可编程框架变量 $P_PFRAME 与系统框架

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

默认情况下有五个可设置框架 $P_UIFR[0]...$P_UIFR[4] 或者 5 个含义相同的G指令 – G500 和 G54 ~ G57 ,值可以保存在其地址下。

$P_IFRAME=$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

注意

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。**例外**：使用 G54, G55, ...编程一个可设置的框架

6.2 给框架变量/框架赋值

6.2.1 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。

编程

```
$P_PFRAME=CTRANS (X, 轴值, Y, 轴值, Z, 轴值, …)
```

或者

```
$P_PFRAME=CTOT (X, 角度, Y, 角度, Z, 角度, …)
```

或者

```
$P_PFRAME=CSCALE (X, 比例, Y, 比例, Z, 比例, …)
```

或者

```
$P_PFRAME=CMIRROR (X, Y, Z)
```

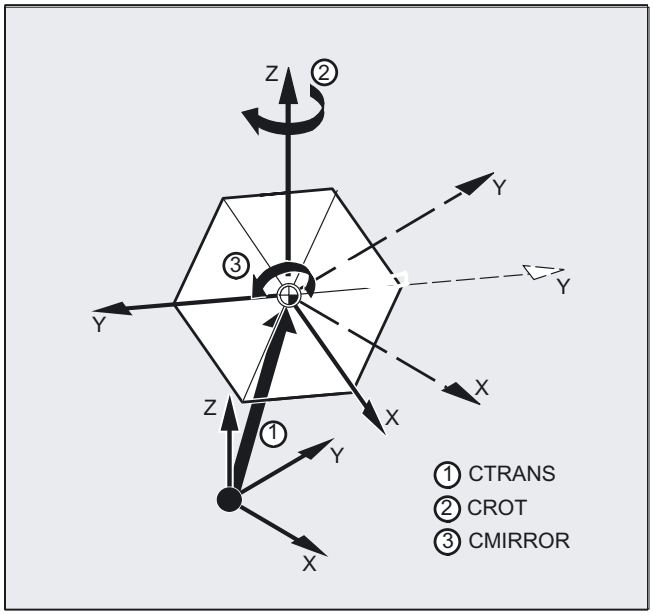
$P_BFRAME 的编程与 $P_PFRAME 相同。

参数

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

举例

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)
```

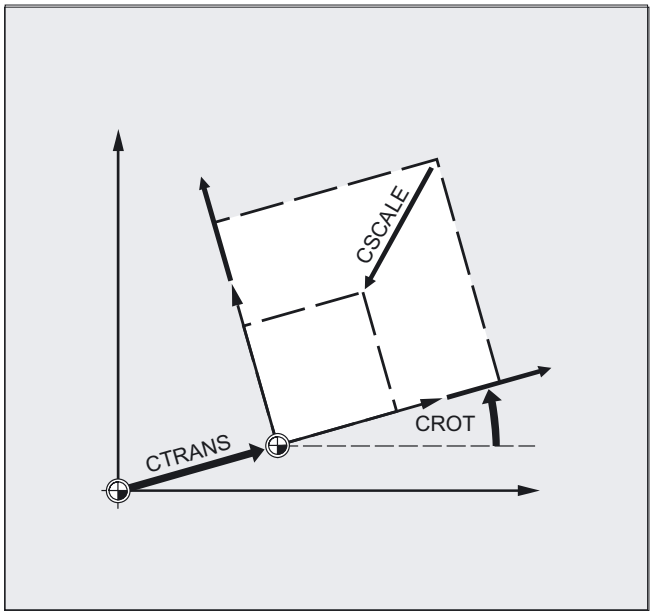
说明

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。



注意

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。

6.2.2 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。

编程

`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。

参数

`$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（参见示例）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：	-89.999° ~ +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

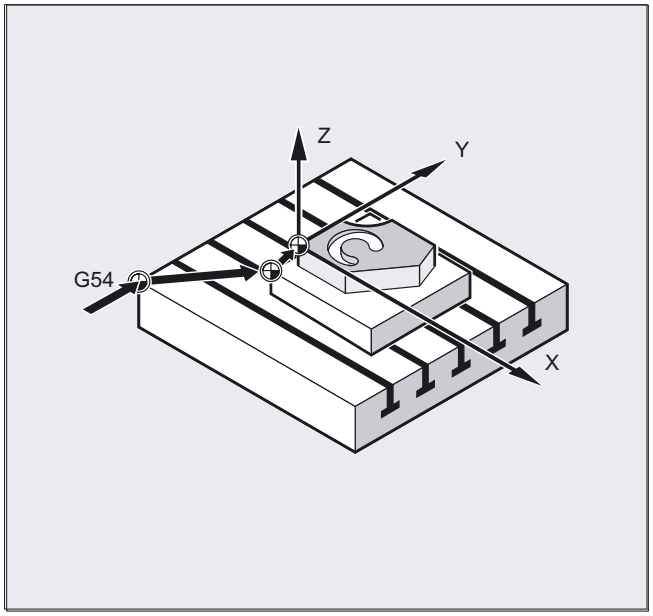
在方括号中的是要访问或者修改的轴名称和值的框架组件，例如 [X, RT] 或者 [Z, MI].

6.2.3 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。

编程

给框架赋值

`DEF FRAME EINSTELLUNG1`
`EINSTELLUNG1=CTRANS(X,10)`
`$P_PFRAME=EINSTELLUNG1`
	将自定义框架 `EINSTELLUNG1` 的值赋给当前的可编程框架。

`DEF FRAME EINSTELLUNG4`
`EINSTELLUNG4=$P_PFRAME`
`$P_PFRAME=EINSTELLUNG4`
	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。

注意

请注意：框架必须通过级联运算符冒号:相互联系起来。

6.2.4 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。

使用函数 CTRANS, CROT, CSCALE, CMIRROR 可以在NC程序中给您的框架赋值。

编程

```
DEF FRAME PALETTE1
```

或者

```
PALETTE1=CTRANS(…):CROT(…)…
```

参数

DEF FRAME	生成新框架。
PALETTE1	新框架的名称
=CTRANS(...):CROT(...)...	给可能有的函数赋值

6.2.5 确定框架旋转 (ROT, ROTS, TOFRAME, TOROT, PAROT)

功能

可以按照用户的特殊要求通过框架旋转来定义空间中的定向。

参数

ROT	所有几何轴一次旋转
ROTS, AROTS, CROTS	根据空间角度（最大2个）的参数旋转；参见/FB1/ K2中的说明：坐标系。
TOFRAME	通过其Z轴指向刀具方向的框架 "TOFRAME"进行旋转。
TOROT	通过仅覆盖已编程框架的旋转部分的框架 "TOROT" 进行旋转
PAROT	工件相关的框架旋转旋转部分通过一个可定向的刀架的旋转部分来决定。

6.3 粗位移和精位移 (CFINE; CTRANS)

功能

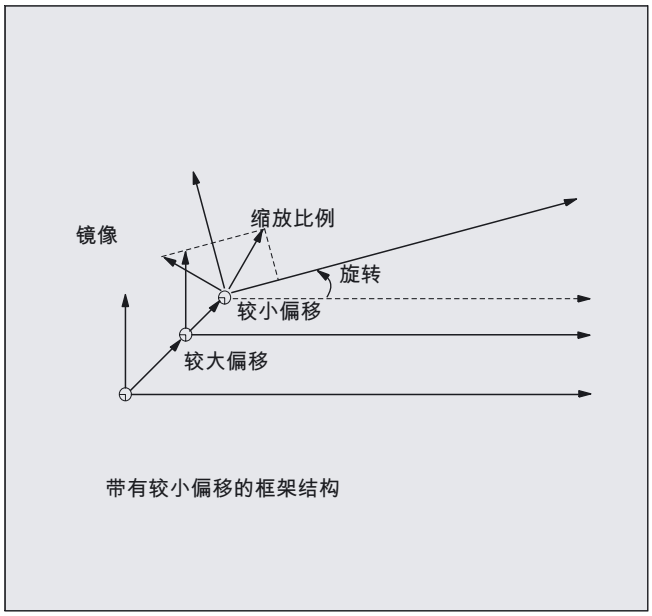
精细位移

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MM_FRAME_FINE_TRANS=1时。

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。

编程

```
$P_UBFR=CTRANS(x, 10) :CFINE(x, 0.1)
:CROT(x, 45)                          ;位移的级联，
                                      ;精位移和旋转
$P_UIFR[1]=CFINE(x, 0.5 y, 1.0, z, 0.1)
                                      ;整个框架可使用 CFINE
                                      ;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读取第三个框架中X轴的精
                                      位移
```

参数

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

使用MD18600：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。

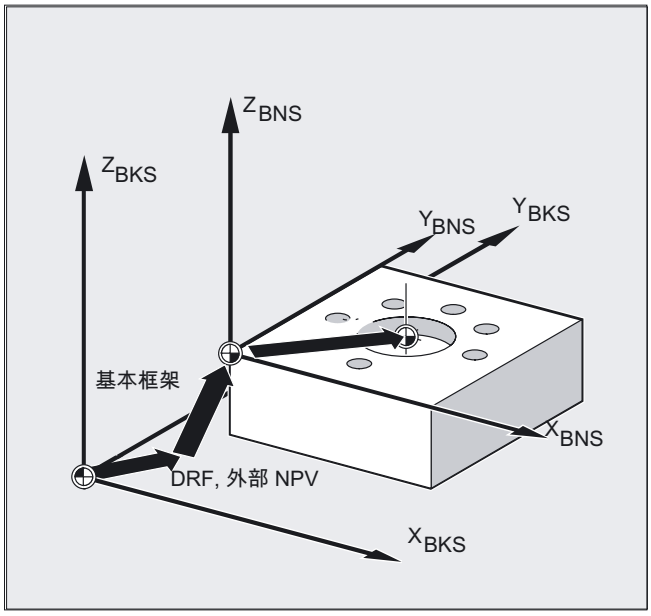
可编程的框架没有精偏移部分。如果带有精偏移的框架赋值给可编程的框架，则总偏移由粗偏移和精偏移的和构成。在读可编程框架时，精偏移始终为零。

6.4 DRF-偏移

使用手轮位移, DRF

除了本章中所说的偏移之外，还可以通过手轮（DRF偏移）另外确定一个零点偏移。

DRF 位移在基本坐标系中有效：



有关更多信息可查阅相应的操作说明书。

清除DRF偏移，DRFOF

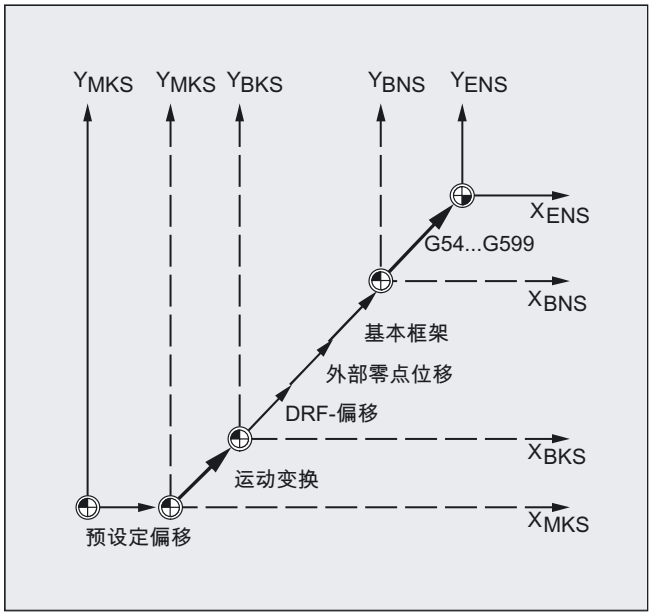
通过DRFOF可以取消该通道中所分配的所有轴的手轮偏移。DRFOF需有一个独立的程序段。

6.5 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。

注意

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。

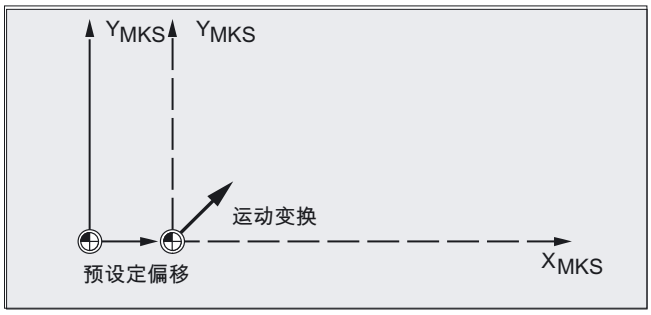
6.6 预设定位移 (PRESETON)

功能

对于一些特殊的应用场合，有时要求对一个或多个轴的当前位置（停止状态）赋值一个新的、编程的实际值。

小心

使用PRESETON功能后基准点变得无效。因此这种功能仅仅应用于不需回参考点的轴中。如果要恢复原来的系统，就必须使用G74来逼近基准点 - 参阅“文件和程序管理”。



编程

PRESETON(轴，值, ...)

参数

PRESETON	设定实际值
轴	加工轴说明
值	新的实际值，适用于所给定的轴

注意

只可使用关键字 "WHEN" 或者 "EVEREY" 来进行带有同步动作的实际值设定。

举例

在机床坐标系中进行实际值赋值 - 值以机床轴为参照。

```
N10 G0 A760
N20 PRESETON(A1,60)
```

轴A向位置760运动。加工轴A1在位置760上获得新的实际值。从现在起在新的实际值系统中进行定位。

6.7 取消框架 (DRFOF, G53, G153 和 SUPA)

功能

可通过将一个“零框架”（没有轴数据）赋给可编程框架的方式，来删除可编程的框架。

编程

DRFOF
或者
G53
或者
G153
或者
SUPA

参数

DRFOF	关闭（取消）手轮偏移（DRF）
G53	按照程序段方式取消可编程的和所有可设定的框架
G153	按照程序段方式取消可编程的框架、基准框架和所有可设定的框架
SUPA	按照程序段方式取消所有可编程的框架、基准框架、所有可设定的框架和手轮偏移（DRF）

零框架赋值举例

```
$P_PFRAME=TRANS( )
$P_PFRAME=ROT( )
$P_PFRAME=SCALE( )
$P_PFRAME=MIRROR( )
```

6.8 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。

编程

```
MEAFRAME IDEAL_POINT,MEAS_POINT,FIT_QUALITY)
```

参数

MEAFRAME	空间中三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。
	-2:	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。
	-4:	旋转矩阵的计算因另外一个原因而失败。
	正值：	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）。

注意

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。

注意

由 MEAFRAME 所生成的框架可以通过函数 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
函数说明 /FB1/ K2, 轴, 坐标系，框架“"FRAME-级联"

举例

```
；零件程序1
;
DEF FRAME CORR_FRAME
;
;设定测量点
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 ;允许零件位置最多旋转5度

DEF REAL FIT_QUALITY_LIMIT = 3 ;允许在理想三角形和测定三角形之间最多偏移 3
mm

DEF REAL SHOW_MCS_POS1[3]
DEF REAL SHOW_MCS_POS2[3]
DEF REAL SHOW_MCS_POS3[3]
;=======================================================
;
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 ;限制第一个RPY-
;角度
SETAL(65020)
GOTOF NO_FRAME
ENDIF
;
N260 IF CORR_FRAME[Y,RT] > ROT_FRAME_LIMIT ;限制第二个RPY-
;角度
SETAL(65021)
GOTOF NO_FRAME
ENDIF
;
N270 IF CORR_FRAME[Z,RT] > ROT_FRAME_LIMIT ;限制第三个RPY-
;角度
SETAL(65022)
GOTOF NO_FRAME
ENDIF
;
N300 $P_IFRAME=CORR_FRAME ;使用一个可设定的框架激活探测框架
;
;通过将几何轴向理想点定位的方式来检查框架
;
N400 X=IDEAL_POINT[0,0] Y=IDEAL_POINT[0,1] Z=IDEAL_POINT[0,2]
N410 SHOW_MCS_POS1[0]=$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]=$AA_IM[X]
N620 SHOW_MCS_POS3[1]=$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 (没有输入值)

;
NO_FRAME:
M0
M30
```

框架级联举例

级联MEAFRAME用于补偿

函数 MEAFRAME( ) 给出一个补偿框架。当该补偿框架与调用函数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，则可将函数 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。

6.9 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：

机床制造商

通过MD设计全局基本框架的数量，参见"通道专用框架" 一章和函数说明 /FB1/ K2, 轴, 坐标, 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。

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

所有可设置框架均可借助 MD 18601 MM_NUM_GLOBAL_USER_FRAMES 重新设计成全局框架，参见
函数说明 /FB1/ K2, 轴，坐标，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。

6.9.1 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/MMC 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”。

6.9.2 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。

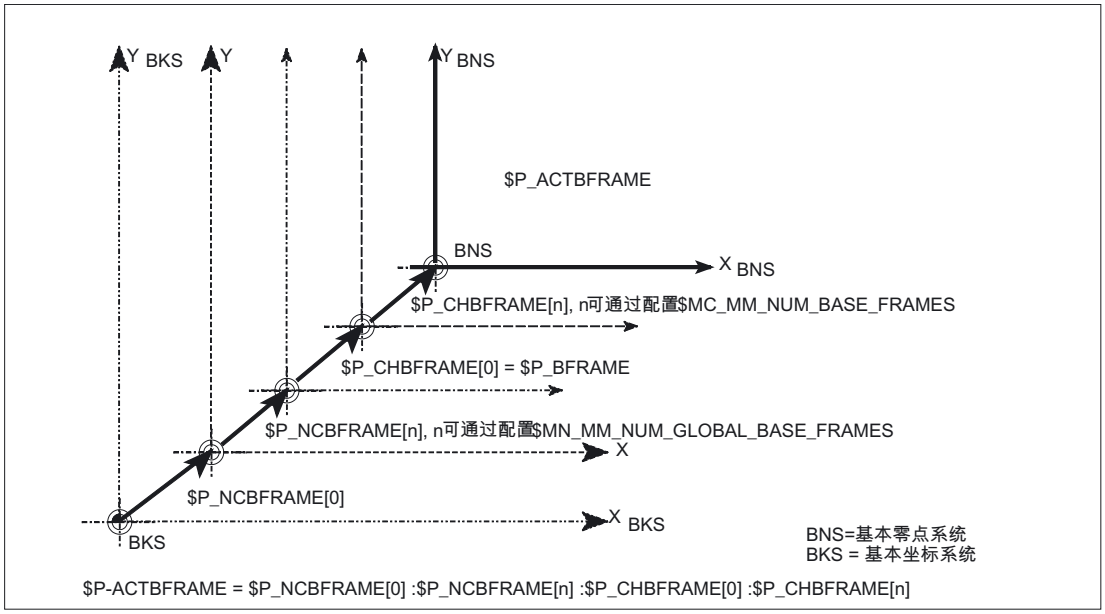
`$P_BFRAME` 和 `$P_CHBFRAME[0]`一样。在正常情况下，系统变量始终有一个有效值。如果没有通道专用基准框架，则在读写时会产生报警“框架：“指令不允许”。

$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。

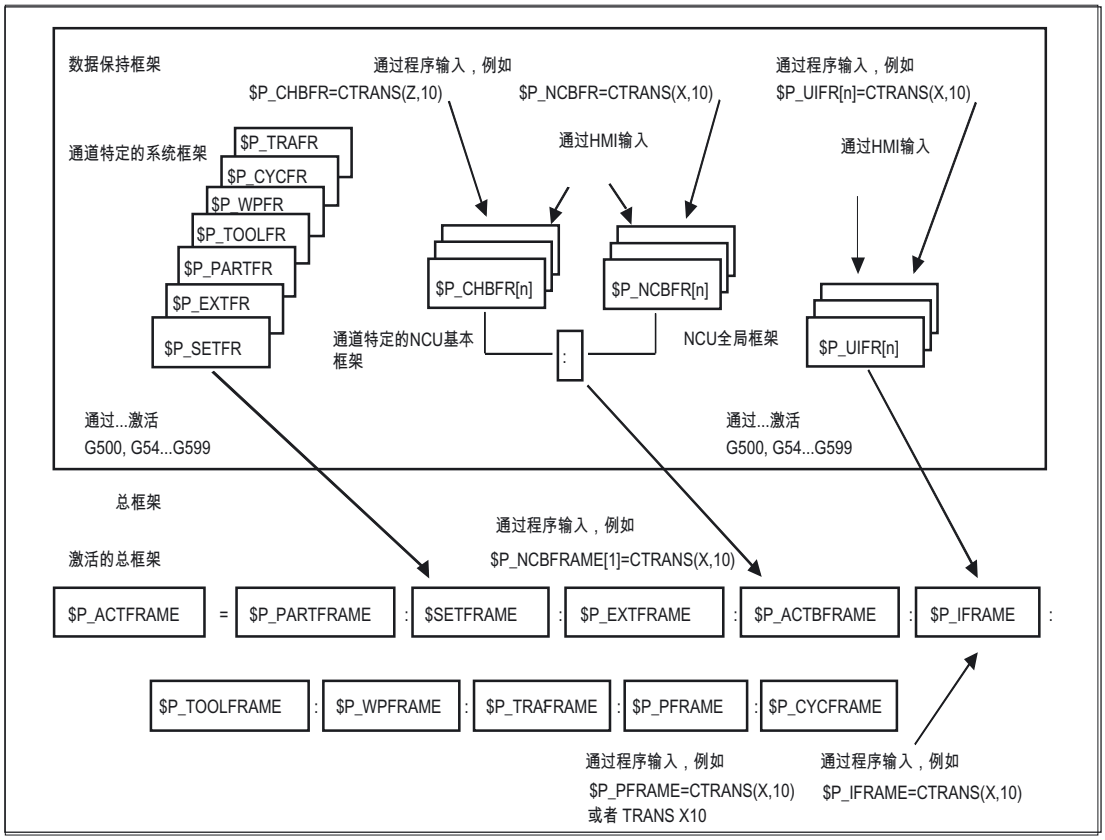
P_ACTFRAME 当前的总框架

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

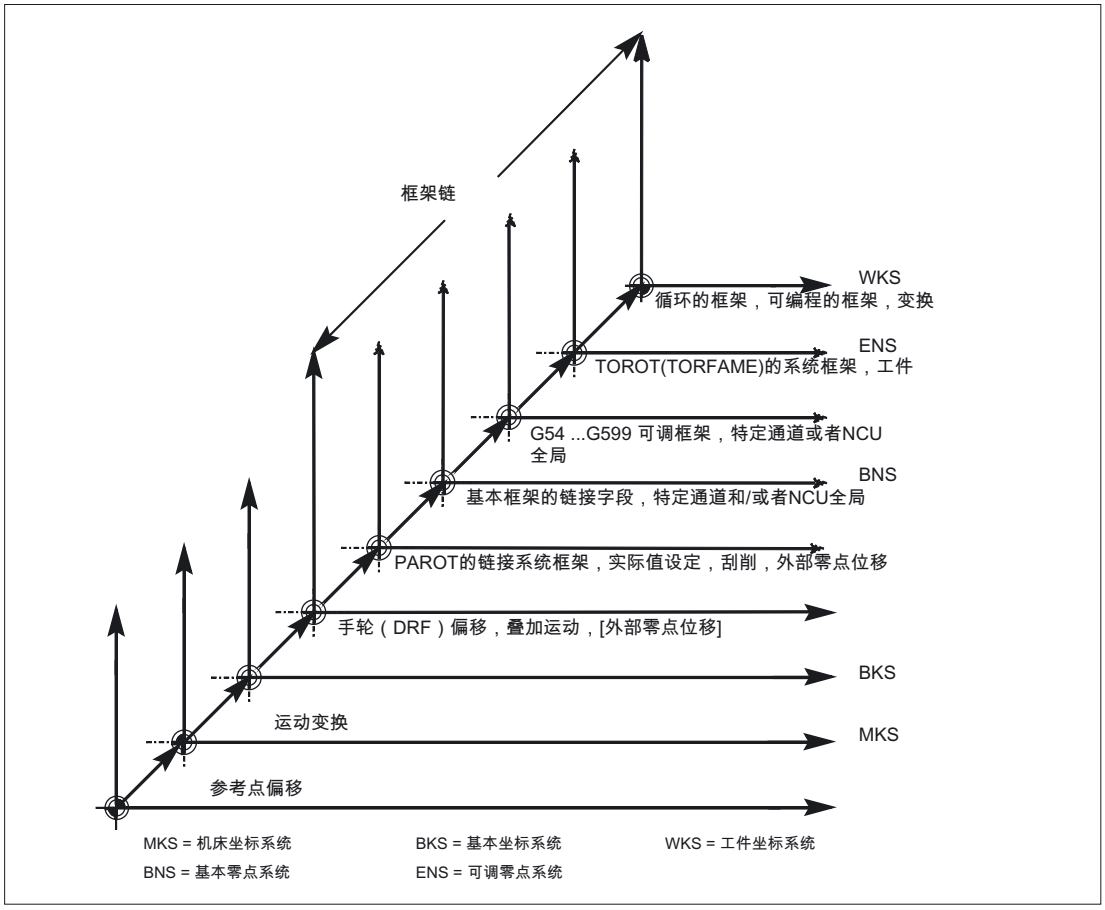
$P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME : $P_ACTBFRAME : $P_IFRAME :

$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

7

3～5轴转换

对立体空间面进行完美加工时，机床除了需要三个线性轴X、Y和Z之外，还需要有辅助轴。这些辅助轴用来描述空间中的定向，因此被称作定向轴。可用于具有不同运动特征的三种类型的机床：

1. 两轴旋转头，例如固定式工作台的万向组合阿刀盘
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

一览

TRAORI	激活3～5轴转换
3轴转换，机床类型1和2	带有**两个** 直线运动轴和**一个** 旋转运动轴（旋转轴）编程的刀具定向
4轴转换，机床类型1和2	带有**三个** 直线运动轴和**一个** 旋转运动轴（旋转轴）编程的刀具定向
5轴转换，机床类型3	带有可旋转线性轴的定向转换。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。
5轴转换，机床类型1、2或者3	定向轴的轴顺序和刀具的定向方向取决于机床运动系统，并且可以要么**以机床为参照** ，通过机床数据或者编程定向进行设计**以工件为参照** ，使用欧拉角或者RPY角或者矢量分量（方向矢量和面法向矢量）进行设计
5轴转换，机床类型1和3	对刀具定向的旋转进行编程使用 ORIROTA, ORIROTR, ORIROTT 进行旋转矢量插补
6轴转换，定向转换	定向轴的关系刀具运动与机床运动系统 **有关**，带有 ORIWKS 编程 **无关** ，带有 ORIMKS 编程

通过机床数据进行设计3个定向轴以此进行旋转的顺序刀具矢量在机床基本位置中的方向通道轴分配通过
轴标识符 A2, B2 和 C2 对定向轴进行编程
同时对定向加以区别：
ORIEULER 通过欧拉角（默认）
ORIRPY 通过 RPY-角
ORIVIRT1 通过第1个定义的虚拟定向轴
ORIVIRT2 通过第2个定义的虚拟定向轴
同时对插补类型加以区别：
ORIAXES 定向轴的线性插补
ORIVECT 定向轴的大圆插补

万向铣头举例

机床至少有5个轴，其中

- 三个直线运动轴用于直线运动，工作点在空间中的任意位置。
- 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。

如果是**一般5轴转换**，就不再存在方向限制，指迄今为止可供使用的5轴转换的旋转轴的方向。与迄今为止所实现的定向转换相比，刀具的基本定向不再通过机床数据设定成固定形式，而是可以通过系统变量任意编程。

文献： 函数说明 /FB3/, F2 "3 ~ 5轴转换"

运动转换

在车床上进行铣削加工时，可以要么

1. 是卡装形式的端面加工，或者
2. 在圆柱体上加工任意走向的槽口

。控制系统将所编程的直角坐标系中的运动转换成加工轴的实际运动。

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

例如，如果要让用来进行磨削加工的进给轴也可斜向进给，可以编程一个可设定参数的角度。

TRAANG	激活斜向轴转换
用斜置横向进给轴加工	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

属于运动转换的还有所谓的“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： 函数说明 /FB2/, M1 "运动转换"

级联的转换

可以依次嵌套两个转换。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

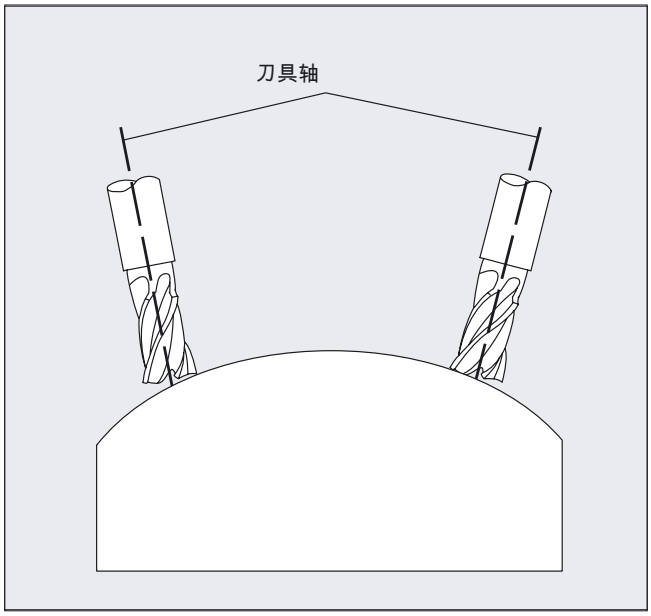
文献： 函数说明 /FB2/, M1 "运动转换"

7.1 三轴、四轴和五轴转换 (TRAORI)

7.1.1 三轴、四轴和五轴转换 (TRAORI)

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。

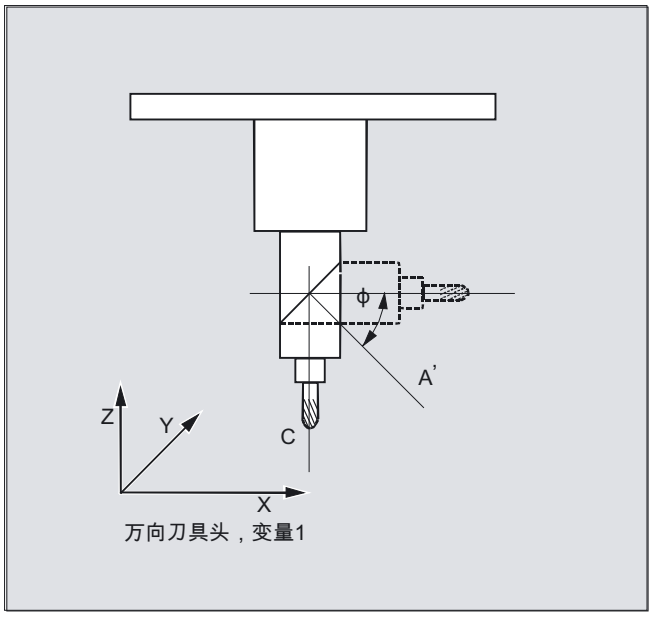


用哪一种机床结构达到这一点，这存储在轴数据中。

5轴转换

万向切削头

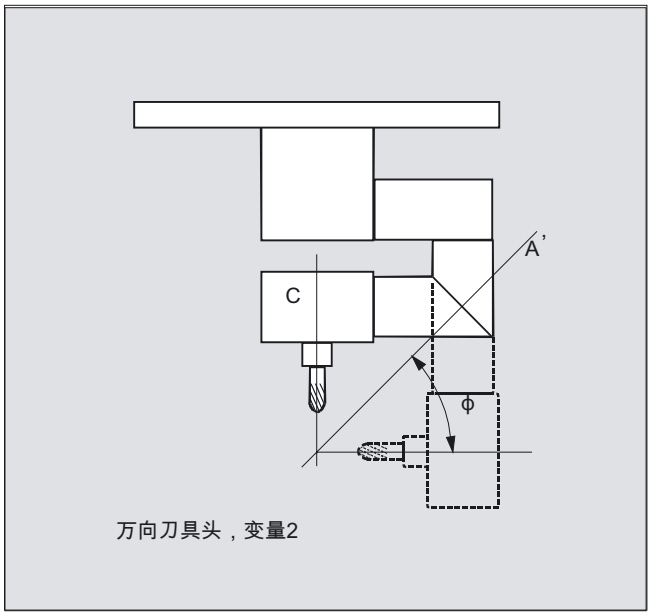
这里三个线性轴 (X, Y, Z) 和两个定向轴 (C, A)用来确定刀具的定位角和工作点。两个定向轴的其中之一是作为斜置轴设置的，在这里为A' (在很多情况下是45°。



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。



万向刀具头，变量2

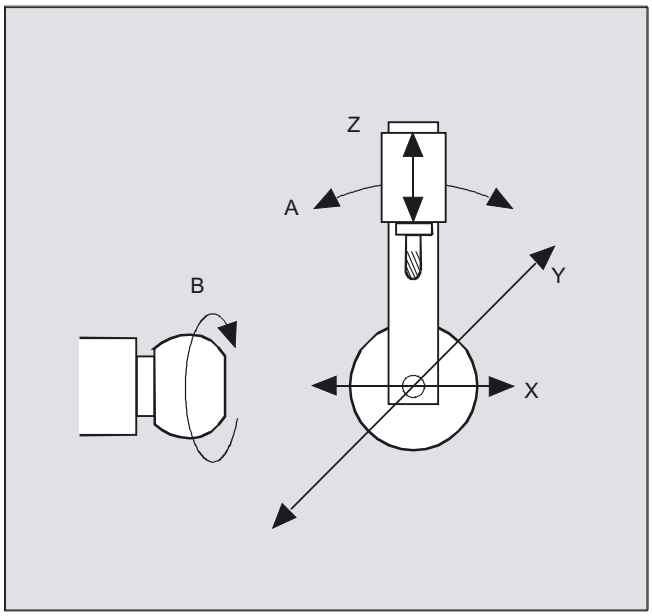
适用于下面的关系：

A' 以角度 φ 相对于	X轴
B' 以角度 φ 相对于	Y-轴
C' 以角度 φ 先对于	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。



回转轴的轴顺序和刀具的运动方向通过机床数据设置，与机床运动类型有关。

适用于下面的关系：

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

3～4轴转换

3轴和4轴的转换是5轴转换的特殊形式。用户可以设计两个或者三个移动的轴和一个回转的轴。转换的条件：旋转运动轴正交于定向平面上。

刀具只在与回转轴垂直的平面上才可以定向。转换支持带有运动刀具和运动工件的机床类型。

3轴和4轴转换的设计与编程和5轴转换相同。

参考文献

功能说明 /FB3/ F2，3～5轴转换

编程

```
TRAORI(n)
```

或者

```
TRAORI(n,X,Y,Z,A,B)
```

或者

```
TRAFOOF
```

参数

TRAORI	激活第一个约定的方向转换
TRAORI(n)	激活用n约定的方向转换
n	转换的编号（n = 1或者2），TRAORI（1）与方向转换一相同
X,Y,Z	刀具所指向的定向矢量分量
A,B	旋转轴的可编程偏移
TRAFOOF	关闭转换

刀具定向

视刀具的所选定向方向而定，必须在NC程序中对激活的工作平面 (G17, G18, G19) 进行调整，使得刀具长度补偿在刀具定向的方向中有效。

注意

在接通转换之后位置说明（X，Y，Z）总是和刀尖有关。修改参与转换的旋转轴的位置会导致其余加工轴也进行这样的补偿运动，使得刀尖的位置保持不变。

定向转换总是从刀尖指向刀具安装位置。

一般转换举例

刀具的基本定向指向：

TRAORI(1,0,0,1) Z-轴方向

TRAORI(1,0,1,0) Y-轴方向

TRAORI(1,0,1,1) Y/Z-轴方向 (相当于位置 -45°)

定向轴的偏移

在激活定向转换时还可以直接编程一个定向轴的附加偏移。

如果在编程时保持正确的顺序不变，参数可以不设。

举例

TRAORI(, , , ,A,B) 当只要输入一个唯一的偏移时。

除了直接编程之外，定向轴的附加偏移还可以自动接收目前有效的零点偏移。接收通过机床数据来设计。

7.1.2 编程刀具定向 (A..., B..., C..., LEAD, TILT)

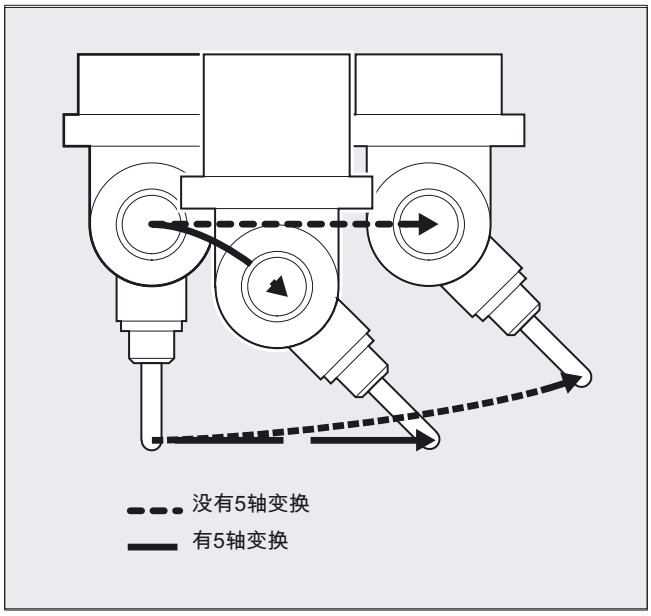
功能

对于刀具的定向编程有3个可能：

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. 在欧拉角或者RPY-角中的编程，通过 A2，B2，C2 或者方向矢量的编程，通过 A3，B3，C3.方向矢量从刀尖指向刀夹
3. 通过超前角 LEAD 和侧向角 TILT 编程(端面铣)

注意

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。

机床制造商

注意

通过机床数据可以在欧拉角和RPY角之间转换。

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

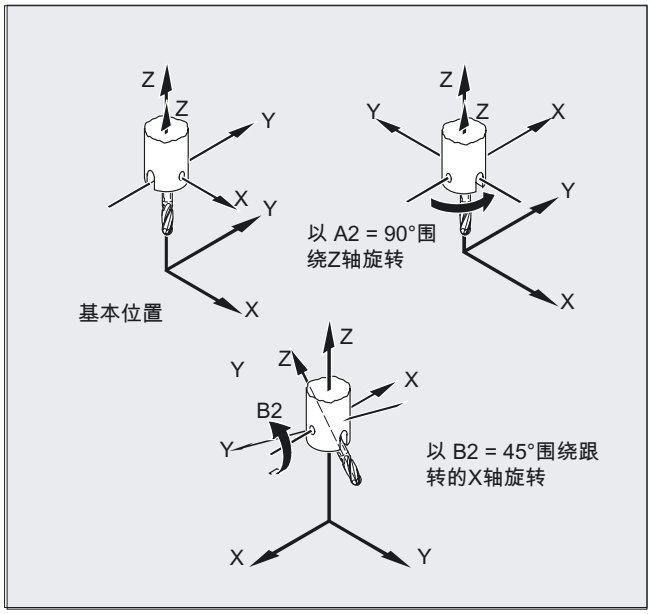
说明

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。

编程欧拉角

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。

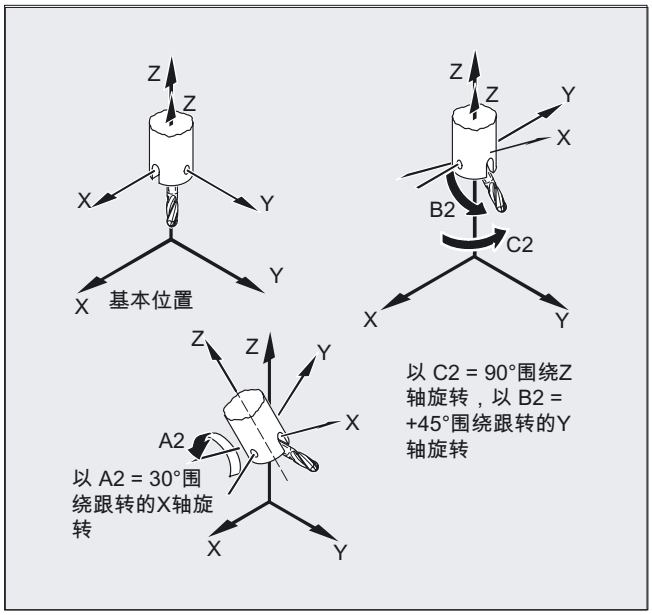


在这种情况下， C2 的值(围绕新Z轴的旋转) 没有意义，且不必进行编程。

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）。

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X轴、最后使用 A2 围绕X轴来使一个矢量在Z方向中旋转。



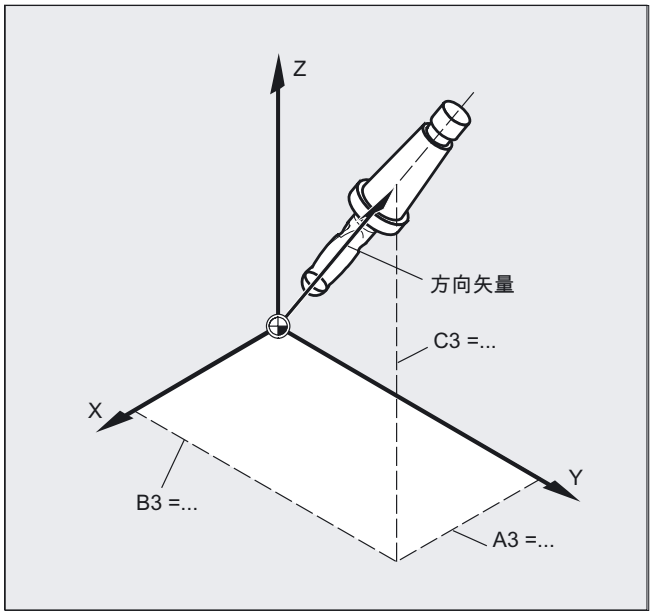
注意

与欧拉角编程相反，这里所有三个值均对定向矢量有影响。

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：

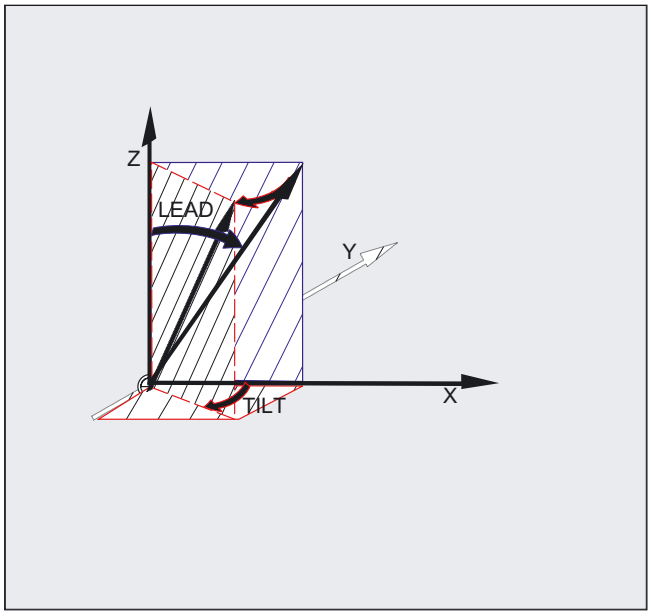
– 轨迹切线，

– 面法向矢量

– 超前角 LEAD

– 程序段结束处的侧向角 TILT

刀具定向，使用 LEAD= 和 TILT=



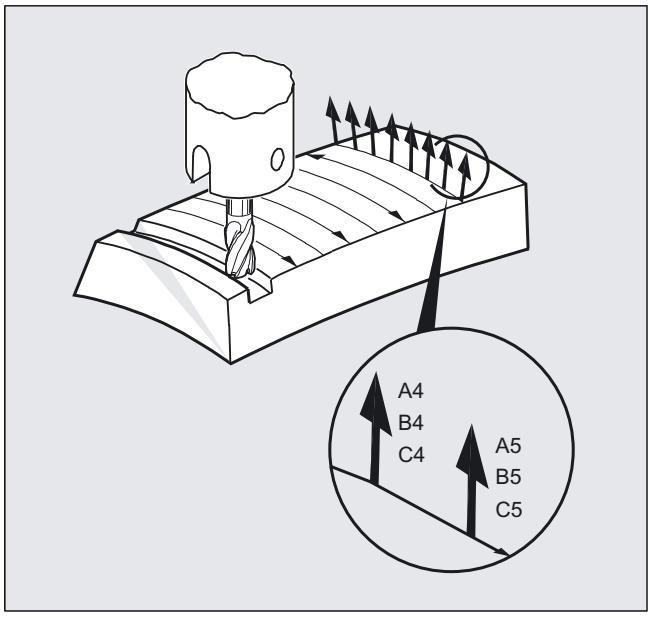
内角的特性（在3D-WZK时）

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

7.1.3 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。

在基本位置终，面法向矢量指向
– 和激活的平面 G17 ~ G19 无关– Z-方向。

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“定向轴的关系”一章)，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。

7.1.4 旋转刀具方位 (ORIROTA, ORIROTR, ORIROTT)

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。

编程

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 或者 N...ORIROTR 或者 N...ORIROTT	确定旋转矢量的插补
THETA	激活方向矢量的旋转
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=θ_e	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[THETA]=(d_2, d_3, d_4, d_5)	用5级多项式插补旋转角度

参数

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	相对于方向矢量改变的旋转角度
THETA	方向矢量的旋转
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[THETA]=(....)	旋转角度的多项式

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。

7.2 方位转换

7.2.1 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。

机床制造商

用机床数据$MC_ORI_IPO_WITH_G_CODE可以确定哪种插补类型有效：

ORIMKS/ORIWKS

或者

ORIMACHAX/ORIVIRTAX.

编程

```
N..ORIMKS=
```

或者

```
N..ORIWKS=
```

参数

ORIMKS	在机床坐标系中旋转
ORIWKS	在工件坐标系中旋转

注意

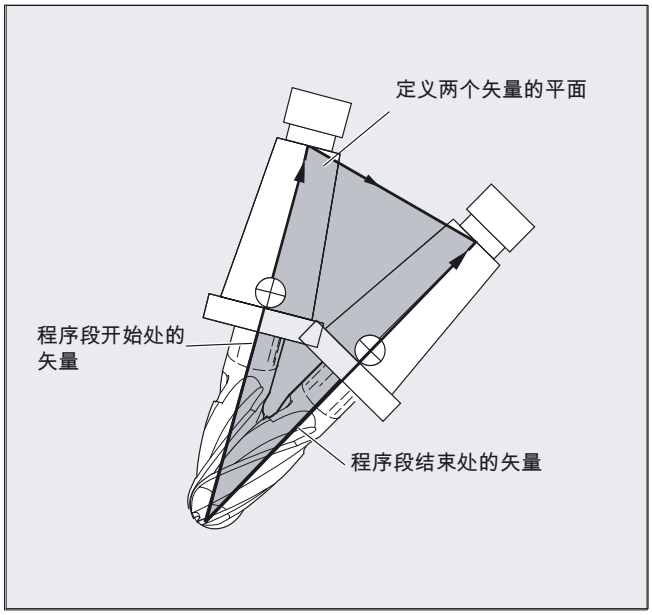
ORIWKS 是默认设置。如果某个5轴程序从开始起就不清楚应在哪个机床上执行，原则上应选择 ORIWKS 。机床实际上执行的运行取决于机床类型。

使用 ORIMKS 可以对实际的机床运动进行编程，以避免例如与装置等发生碰撞。

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，刀具运动与机床运动机构**无关**。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

注意

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。

功能说明文件资料
/FB3/, F2, 5-轴转换，“单位置及其处理”。

7.2.2 定向轴 (ORIAXES, ORIVECT, ORIEULER, ORIRPY)

功能

定向轴的功能描述了空间里刀具的定向。对此更进一步的引入了一个第三自由度，这个自由度描述了围绕自身的旋转。这个对6轴转换是必要的。

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	定向编程：
`N...ORIEULER` 或者 `ORIRPY`	定向角度通过 Euler/RPY
`N...G1 X Y Z A2= B2= C2=`	
或者	或者
`N...ORIVIRT1` 或者 `ORIVIRT2`	定向角度通过虚拟定向轴
`N...G1 X Y Z A3= B3= C3=`	
或者	或者
`N...ORICONCW` 或者 `ORICONCCW`	扩展式定向插补
`N...ORICONTO` 或者 `ORICURVE`	在一个锥体表面上
`N...G1 X Y Z A6= B6= C6=`	
`N...ORICONIO`	
`N...G1 X Y Z A7= B7= C7=`	

扩展式定向插补

用扩展的定向可以沿位于空间的圆锥表面执行方向改变。需要以下的编程说明：

- 通过前面程序段的起始方向
- 终点定向要么通过带有 `A3, B3, C3` 的方向矢量或者在 Euler 角以及RPY 角中通过 `A2, B2, C2`
- 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

带有 `A7, B7, C7` 的中间定向

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

G1 X Y Z 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
G1 X Y Z 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
G1 X Y Z A3= B3= C3=	方向轴的方向矢量编程

定向插补

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
G1 X Y Z A6= B6= C6=	圆锥的旋转轴的编程（标准化的矢量）
ORICONIO G1 X Y Z A7= B7= C7=	在圆锥表面上带中间定向说明的插补（作为标准化矢量编程）
ORICURVE XH YH ZH 带多项式 PO[XH]=(xe, x2, x3..)	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

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
```

直角坐标系中手动运动的功能可以在带有“转换程序包处理程序”的SINUMERIK 840D 和 Sinumerik 810D powerline 的JOG运行过程中，对 MKS, WKS 和 TKS 参照系中的几何轴单独进行设置。

文献提示

功能说明 /FB2/ M1，“运动转换”

7.3 在线式刀具长度补偿 (TOFFON, TOFFOF)

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。

三个几何轴命名符作为变址使用。与此同时当前有效的补偿方向的数量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：

- 定向转换TRAORI
- 可定向的刀架TCARR

机床制造商

在线式刀具长度补偿是一种**选项功能**，必须事先予以激活。只有在与一个激活的定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。

编程

```
N..TRAORI
N..TOFFON(X,25)
N..WHEN TRUE DO $AA_TOFF[X] 在同步动作中
```

参数

TOFFON	**刀具 关设置 开** （激活在线式刀具长度补偿） 在激活时可以对于相应的补偿方向说明一个偏移值，这个值立刻可以得出。
TOFFOF	**刀具 关设置 OF** （复位在线式刀具长度补偿） 相应的补偿值被复位并且释放进刀停。

X, Y, Z,	说明的偏移值的补偿方向

选择刀具长度补偿示例

```
MD 21190:TOFF_MODE =1                          ;返回绝对值
MD 21194:TOFF_VELO[0] =1000
MD 21196:TOFF_VELO[1] =1000
MD 21194:TOFF_VELO[2] =1000
MD 21196:TOFF_ACCEL[0] =1
MD 21196:TOFF_ACCEL[1] =1
MD 21196:TOFF_ACCEL[2] =1
N5 DEF 实数 X偏置
N10 TRAORI(1)                                  ;转换 开
N20 T关开(Z)                                   ;激活Z轴刀具方向的
                                               ;在线式刀具长度补偿
N30 WHEN TRUE DO $AA_TOFF[Z] = 10              ;给Z轴刀具方向插补一个
G4 F5                                          ;刀具长度补偿10
...
N40 T关开(X)                                   ;激活X轴刀具方向的
                                               ;在线式刀具长度补偿
N50 ID=1 DO $AA_TOFF[X] = $AA_IW[X2]           ;对X轴刀具方向进行补偿
G4 F5                                          ;，取决于
                                               ;轴X2 的位置
...
N100 XOFFSET = $AA_TOFF_VAL[X]                 ;给X轴方向中的当前补偿赋值
N120 T关开(X, -XOFFSET)                        ;对于x轴刀具方向，将
G4 F5                                          ;刀具补偿重新恢复为0
```

取消刀具长度补偿示例

```
N10 TRAORI(1)                                  ;转换 开
N20 T关开(X)                                   ;激活Z - 刀具方向
N30 WHEN TRUE DO $AA_TOFF[X] = 10              ;给X轴刀具方向插补一个
G4 F5                                          ;刀具长度补偿10
...
N80 TOFFOF(X)                                  ;X轴刀具方向的位置偏置量
                                               ;被取消：...$AA_TOFF[X] = 0
                                               ;没有轴运动
                                               ;根据当前的定向给WKS中的
                                               ;当前位置添加
                                               ;位置偏置量
```

说明

程序段预处理

在运行程序段预处理时要考虑到在主运行中有效的当前刀具长度偏移。为了进一步充分利用允许的最大轴速度，需要用一个进刀停STOPRE，在建立刀具偏移这段时间里来暂停程序段预处理。

如果刀具长度补偿在程序启动之后不再改变，或者如果在改变刀具长度补偿之后执行多个程序段，它多于进刀运行和主运行之间IPO - 缓冲器所能接收地程序段，那么刀具偏移在进刀时也始终已知。

变量$AA_TOFF_PREP_DIFF

在当前插补器中的有效补偿和程序段预处理时有效的补偿之间的差数程度可以在变量 `$AA_TOFF_PREP_DIFF[ ]` 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机床数据：

- `MD 20610:ADD_MOVE_ACCEL_RESERVE` 设计速度时的备用值
- `MD 21190:TOFF_MODE` 系统变量 `$AA_TOFF[ ]` 的内容可作为绝对值取得或者相加。
- `MD 21194:TOFF_VELO` 在线式刀具长度补偿的速度
- `MD 21196:TOFF_ACCEL` 在线式刀具长度补偿的加速度
- 用于设定极限值的设定日期
  `SD 42970:TOFF_LIMIT` 刀具长度补偿上限

参考文献

功能说明 /FB3/ F2，3～5轴转换

7.4 运动变换

7.4.1 铣削车削件(TRANSMIT)

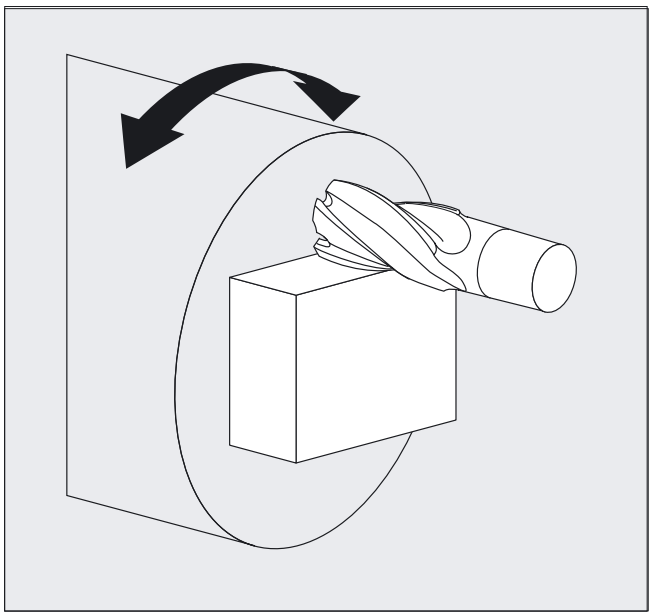
功能

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。

编程

TRANSMIT 或者 TRANSMIT(n)

或者

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。

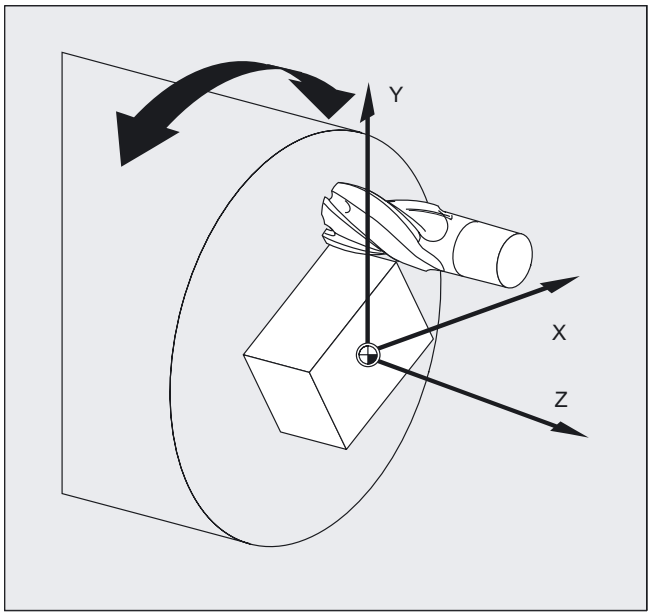
参数

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

注意

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).

举例




```
N10 T1 D1 G54 G17 G90 F5000 G94          ；刀具选择
N20 G0 X20 Z10 SPOS=45                   ；返回起始位置
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=1               ；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                                    ;取消框架
N200 TRANS
N210 TRAFOOF
N220 G0 X20 Z10 SPOS=45                         ；返回起始位置
N230 M30
```

说明

极点

有两种方法通过极点：

- 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

功能说明 /FB2/, M1 “运动转换”

7.4.2 圆柱形外壳转换（TRACYL）

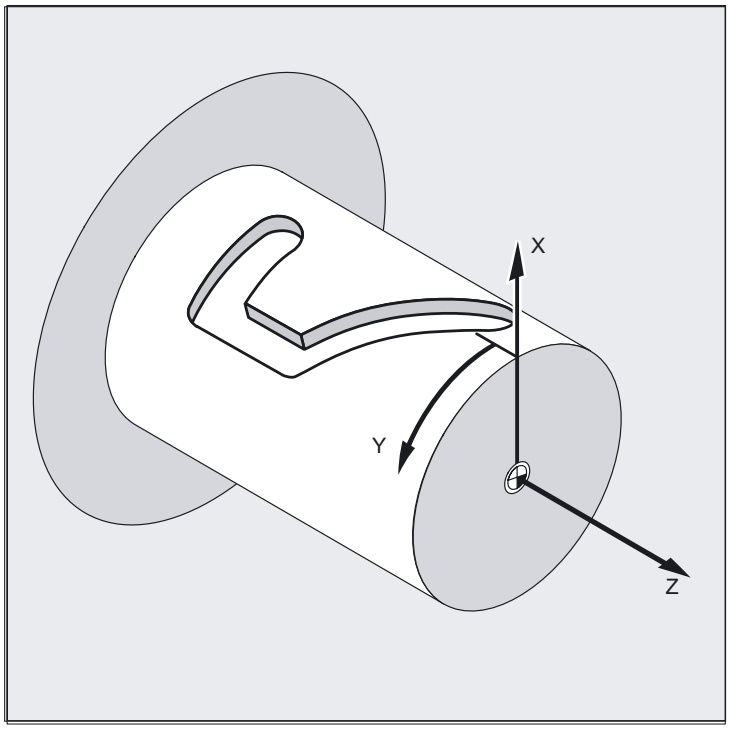
功能

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）

编程

TRACYL(d) 或者TRACYL(d,n) 或者

用于转换类型 514

TRACYL(d,n,槽壁补偿)

或者

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。

参数

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

注意

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).

刀具定义示例

下列示例适用于对圆柱转换 TRACYL 的参数设定进行测试：

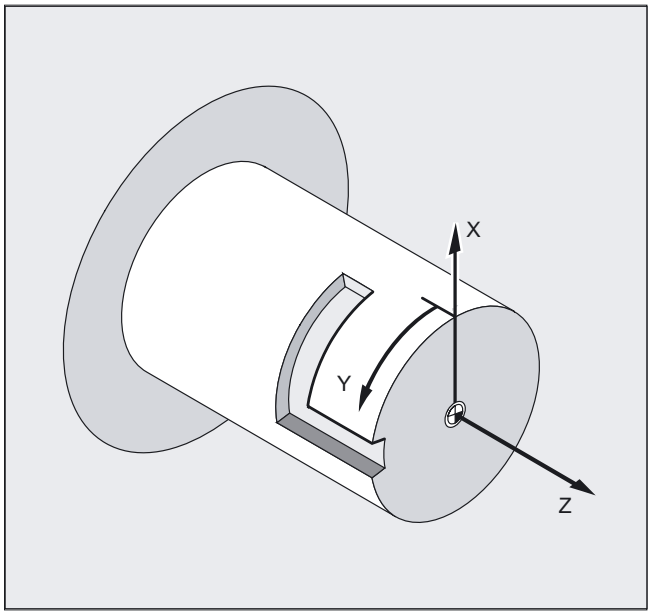
刀具参数 编号(DP)	**意义**	**附注**
$TC_DP1[1,1]=120	刀具类型	铣刀
$TC_DP2[1,1]=0	切削刃位置	只针对车刀

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.		

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	

磨损	**长度和半径补偿**	
$TC_DP12[1,1]=0	剩余参数直到$TC_DP24=0	基础尺寸/适配器

加工一个钩形槽的示例



启动圆柱面转换

```
N10 T1 D1 G54 G90 F5000 G94          ;刀具选择，装夹补偿
N20 SPOS=0                           ；返回起始位置
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

```
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 12mm
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 4mm
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                          ;空走刀
N160 TRAFOOF
N170 G0 X25 Y0 Z105 CC=200           ；返回起始位置
N180 M30
```

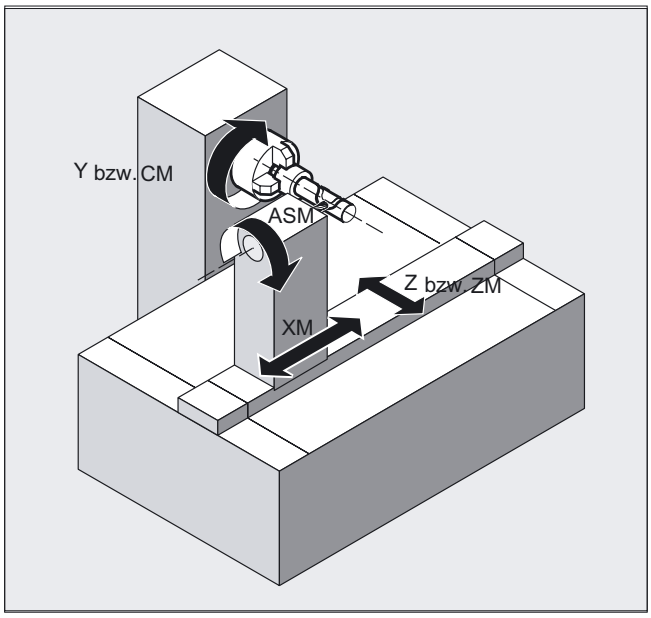
说明

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。

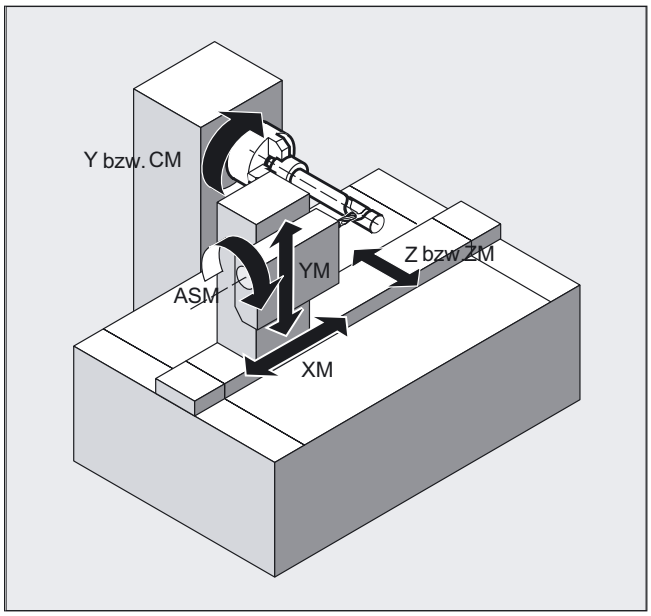


有槽壁补偿 (转换类型 513):

运动同上，但是纵轴平行于圆周方向

线性轴互相垂直。

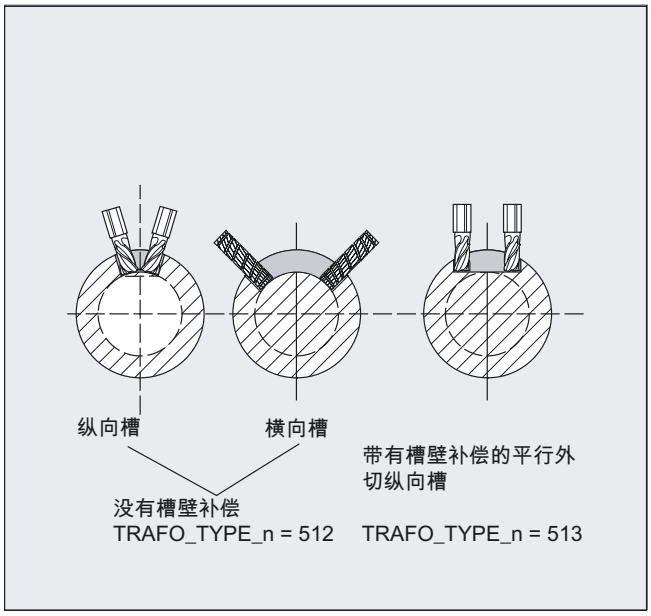
速度导向考虑到为旋转运行所定义的限制。



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

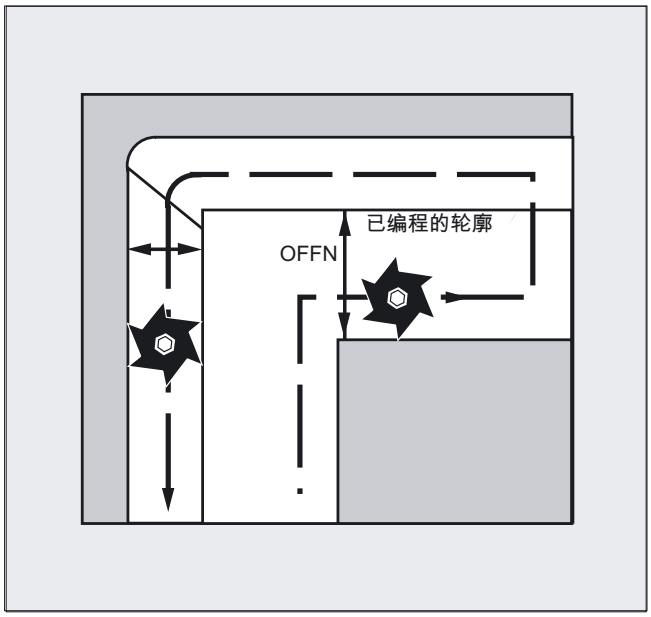
功能说明 /FB2/, M1 “运动转换”

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。

注意

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。

7.4.3 斜置轴

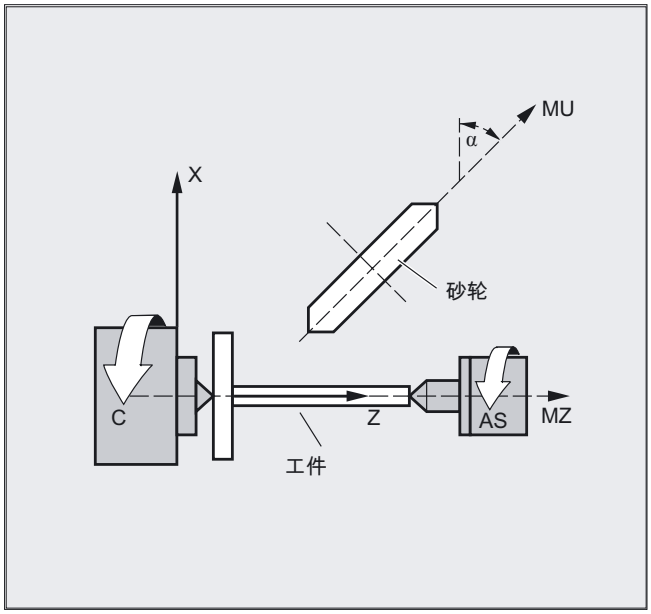
7.4.3.1 斜置轴 (TRAANG)

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：

- 用斜置横向进给轴加工
- 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）：斜置的横向进给轴。



编程

TRAANG(α) **或者** TRAANG(α,n)

或者

TRAFOOF

参数

TRAANG	如果省略角 α 或者输入零，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。
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).
α	斜置轴的角度
TRAFOOF	转换 关
n	已约定转换的数量

目前为止的特性 (角α 被省略或者为零)

如果省略角 α
或者输入零，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。(软件版本低于 6.4 时的特性，新版本请参阅扩展).

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 TRAANG 会被中断（例如TRACYL, TRANSMIT, TRAORI).

扩展：角α 被省略或者为零

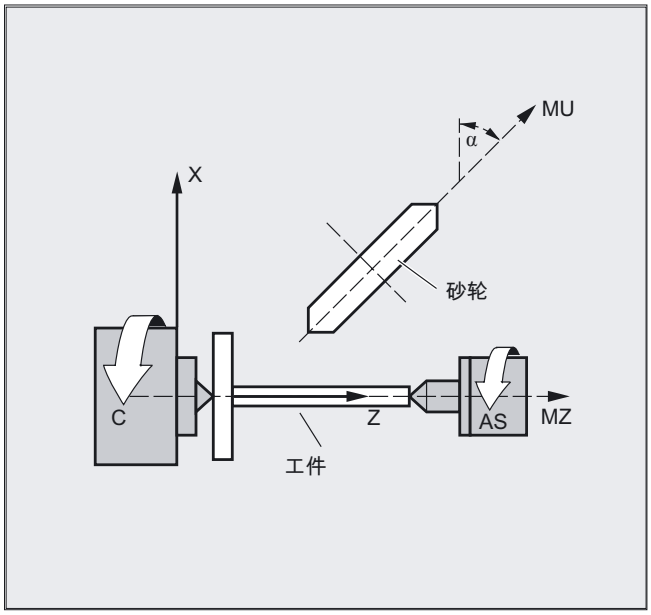
如果 α（角）被省略（例如TRAANG(), TRAANG(,n)），就会使用之前选中的参数设定来激活转换。在第一次选择的时预设置按照机床数据。

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。

α 的允许值为：

-90 度 < α < + 90 度

举例




```
N10 G0 G90 Z0 MU=10 G54 F5000 ->
-> G18 G64 T1 D1
N20 TRAANG(45)
N30 G0 Z10 X5
N40 WAITP(Z)
```

;刀具选择，;装夹补偿，
;选择平面
;启动;斜置轴转换
；返回起始位置
;使轴摆动

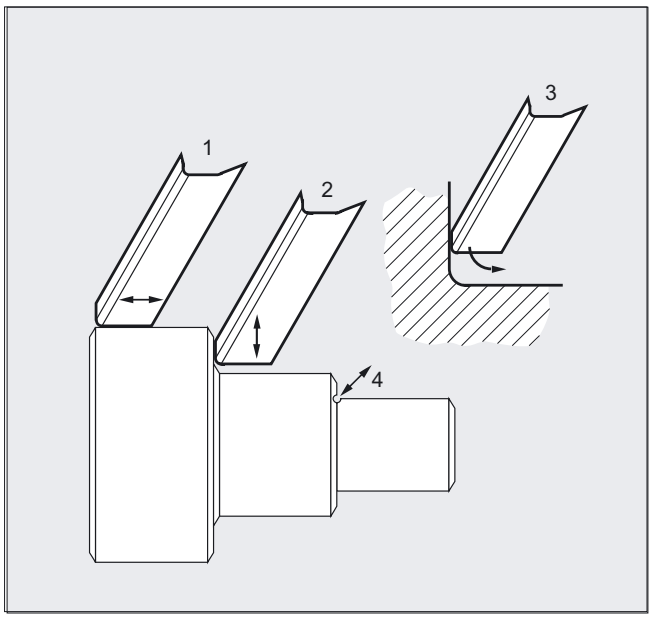
```
N50 OSP[Z]=10 OSP2[Z]=5 OST1[Z]=-2 ->
-> OST2[Z]=-2 FA[Z]=5000                ;摆动，直至达到尺寸为止
N60 OS[Z]=1                             ;（摆动请参阅"摆动"一章）
N70 POS[X]=4.5 FA[X]=50
N80 OS[Z]=0
N90 WAITP(Z)                            ;使摆动轴成为定位轴；
N100 TRAFOOF                            ;中断转换
N110 G0 Z10 MU=10                       ;空走刀
N120 M30
```

-> 在一个程序段中编程

说明

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. 斜置切入磨削



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储备，
- 在平行轴上为补偿运动预留的加速度储备。

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。

7.4.3.2 编程斜置轴 (G05, G07)

功能

在JOG工作状态中，砂轮可以选择在直角坐标系中运动，或者沿着斜置轴的方向运动（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。

参考文献

功能说明 /FB2 /, M1“运动转换”

编程

G07

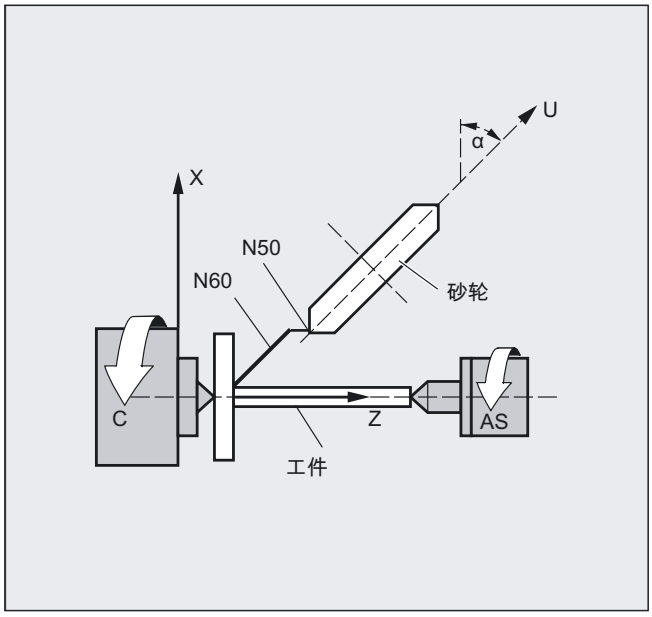
G05

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) 。

参数

G07	返回起始位置
G05	激活斜置切入

举例




```
N..                                    ;编程斜置轴的角;
N20 G07 X70 Z40 F4000                  ;返回起始位置
N30 G05 X70 F100                       ;斜向切入
N40 ...
```

7.4.4 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。

注意

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。

编程

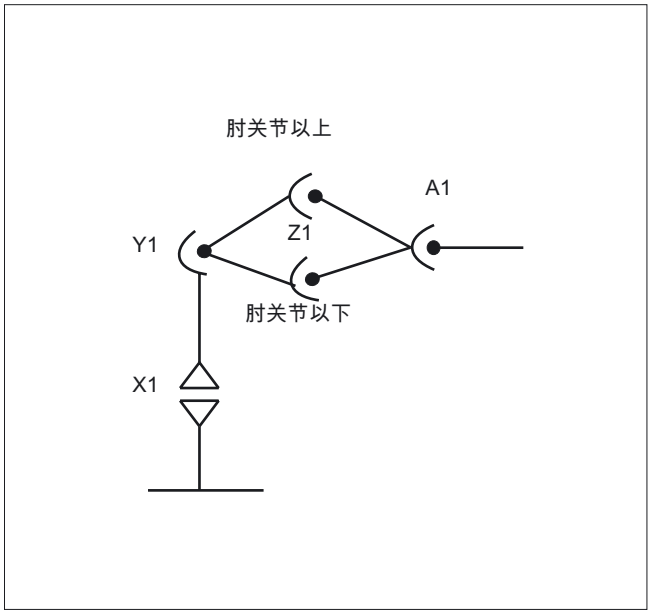
```
N...TRAORI
N...STAT='B10' TU='B100' PTP
N...CP
```

指令 PTP 和 CP 为模态有效。CP 是默认设置。

参数

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。

举例




```
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
```

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。

不同的转换包含在手册中：
功能说明 (第3部分), TE4, "转换软件包处理".

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。

参考文献

表示位置的位，必须在使用 "STAT" 的情况下进行编程，参见：功能描述，
/FB2/ M1, 运动转换“直角坐标PTP运动”一章

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：

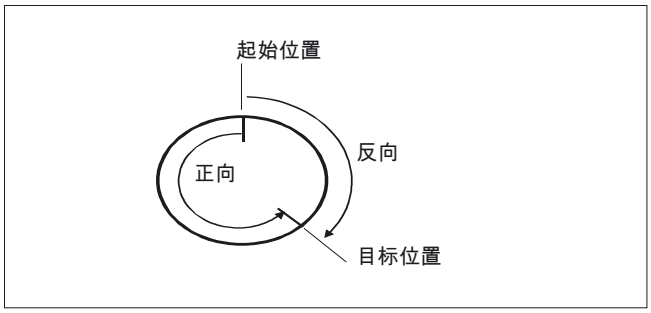
- 当某个位置没有编程 TU 时，
- 如果是运动范围大于 > ±360 度的轴。

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



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。

电源 开/复位

在接通电源或者复位之后，设置取决于加工日期 $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。

7.4.4.1 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。

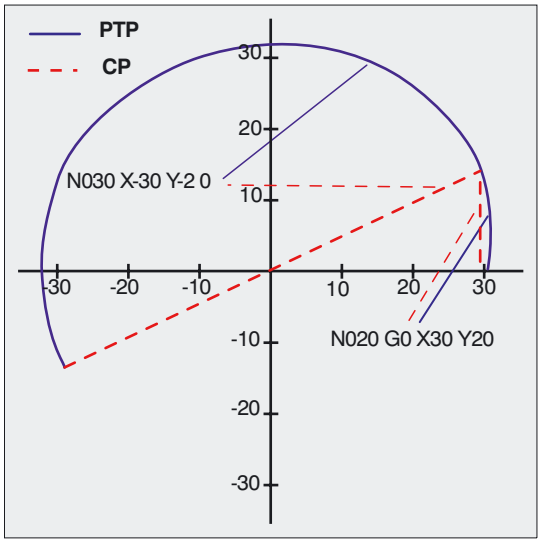
编程

```
N...TRANSMIT
N...PTPG0
N...G0 ...
...
N...G1 ...
```

参数

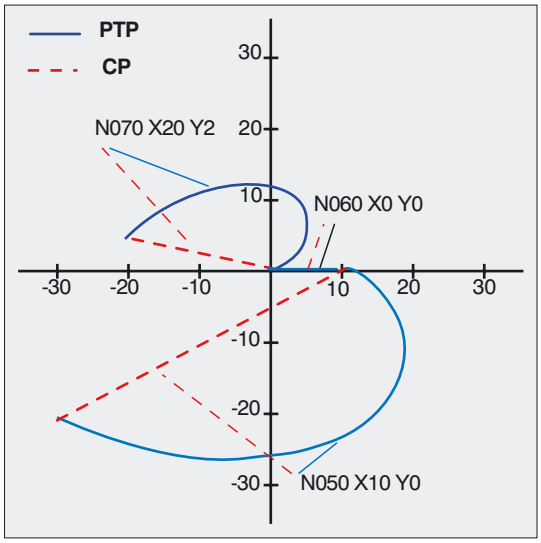
TRANSMIT	激活第一个约定的TRANSMIT功能 (参见章节"在车削件上进行洗削加工：TRANSMIT")
PTPG0	**P**oint **t**o **P**oint **G0** (点到点运动自动执行到每个G0程序段，然后重新设定CP) 因为 STAT 和 TU 为模态，上一次编程的数值始终有效。
PTP	**P**oint **t**o **P**oint (点到点运动) 对于 TRANSMIT而言，PTP表示在直角坐标系中，在阿基米德螺旋线上要么围绕极点或者从极点运动出来。这里所得出的刀具运动明显不同于CP，且已在相应的编程示例中描述过。
STAT=	解决关于极点的多义性。
TU=	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




```
N001 G0 X30 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 X30 Y20
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




```
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GO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：

功能描述，/FB2/ M1, 运动转换，“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。

7.5 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。
- 参与转换的轴上的DRF偏移在进刀和主运行加工之间不可以更改（至软件版本SW3）。
- 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）。

7.6 取消转换 (TRAFOOF)

功能

用指令TRAFOOF使所有当前有效的转换和框架关闭。

注意

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）。

编程

```
TRAFOOF
```

参数

TRAFOOF	关闭所有当前有效的转换/框架

7.7 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。

注意

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。

编程

TRACON(trf, par)	一个级联的转换开通。
TRAFOOF	

参数

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。通过设定分号必须做到：当要让前一个参数的基本设置有效时，对已声明的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可以在功能描述中找到：

/FB2/, M1 "运动转换"和

/FB3/, F2 "3- 至 5-轴转换".

7.8 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。

编程

GEOAX(n,通道轴,n,通道轴,…)

或者

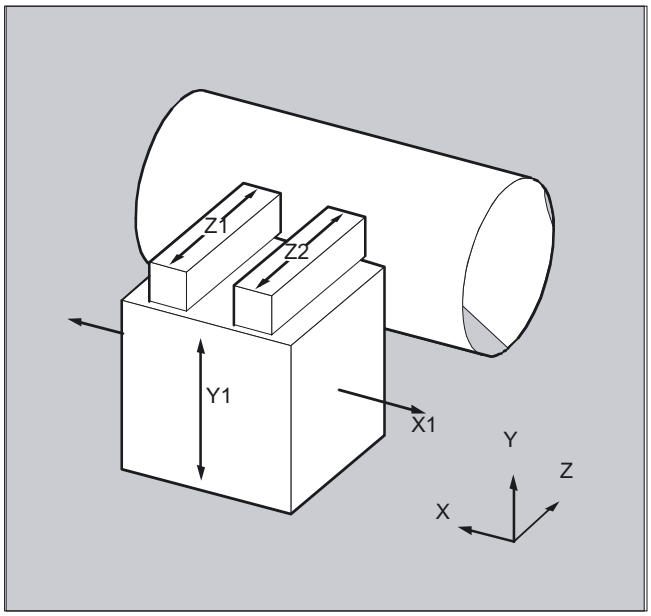
GEOAX()

参数

GEOAX(n,通道轴,n,通道轴,…)	几何轴转换。
GEOAX()	调入几何轴基本配置
n	几何轴编号（n=1，2或者3），另一个通道轴应分配到这个几何轴。 n=0：从几何轴组合中没有替换地清除说明的通道轴。
通道轴	通道轴的名称，这个通道轴应被纳入几何轴组合中去。

举例1

刀具溜板可以通过通道轴X1，Y1，Z1，Z2来运行。在零件程序中可以使用轴Z1和Z2交替地作为几何轴Z。轴之间的转换在零件程序中用GEOAX进行。



在启用之后， X1, Y1, Z1连接就会有效（可通过MD进行设置）。

```
N100 GEOAX (3,Z2)                    ;通道轴Z2作为Z轴
N110 G1 .....
N120 GEOAX (3,Z1)                    ;通道轴Z1作为Z轴
```

举例2

一个机床有6个通道轴，名称是XX，YY，ZZ，U，V，W。通过机床数据的几何轴配置的基本设置是：

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）

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.
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。

前提条件和限制

1. 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（参见PG基本部分“刀具补偿”一章）
   - 激活的刀具精密补偿（参见PG基本部分“刀具补偿”一章）
2. 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。
3. 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。
4. 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）。
5. 使用 GEOAX 在处理**轮廓表** 的过程中更换轴（CONTPRON, CONTDCON) 会导致报警。

说明

几何轴编号

在指令 GEOAX(n,通道轴...) 中，代码n表示几何轴，随后声明的通道轴应分配给该几何轴。

对于一个通道轴的转换几何轴编号1至3（X-, Y-, Z-轴）是允许的。

用n=0从几何轴组合中清除一个没有用几何轴重新覆盖的赋值的通道轴。

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。

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 (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在转换之后依旧有效。

交换轴位置

通过对已经赋值的通道轴的轴编号的新的指示可以在几何轴组合中进行一个位置转换。

```
N...GEOAX (1, XX, 2, YY, 3, ZZ)          ;通道轴XX 为第一个, YY 为第二个
N...GEOAX (1, U, 2, V, 3, W)             ;且ZZ 为第三个几何轴,
                                         ;通道轴U为第一个, V 为第二个
                                         ;且 W 为第三个几何轴。
```

使转换无效

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。

转换过程和刀具长度补偿

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。

几何轴配置和转换切换

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。

举例：

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。

刀具补偿

8

8.1 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。

机床制造商

通过机床数据文件可以配置用户切削数据。

参数

注意

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 编号(DP)	系统变量的意义	附注
$TC_DP 1	刀具类型	概要参见清单
$TC_DP 2	切削刃位置	只针对车刀
几何尺寸	**长度补偿**	

$TC_DP 3	长度1	根据类型计算
$TC_DP 4	长度2	和平面
$TC_DP 5	长度3	
几何尺寸	**半径**	
$TC_DP 6	半径	
$TC_DP 7	切槽锯片的槽宽b， 铣削刀具的倒圆半径	
$TC_DP 8	超出规定范围k	只针对切槽锯片
$TC_DP 10	圆锥形铣削刀具的角度1	刀具的端面
$TC_DP 11	圆锥形铣削刀具的角度2	刀具纵向轴
磨损	**长度和半径补偿**	
$TC_DP 12	长度1	
$TC_DP 13	长度2	
$TC_DP 14	长度3	
$TC_DP 15	半径	
$TC_DP 16	切槽锯片的槽宽b， 铣削刀具的倒圆半径	
$TC_DP 17	超出规定范围k	只针对切槽锯片
$TC_DP 19	圆锥形铣削刀具的角度1	刀具的端面
$TC_DP 20	圆锥形铣削刀具的角度2	刀具纵向轴
基础尺寸/适配器	**长度补偿**	
$TC_DP 21	长度1	
$TC_DP 22	长度2	
$TC_DP 23	长度3	
工艺		
$TC_DP 24	后角	针对车刀

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。

8.2 刀具管理的语言指令

功能

当使用刀具管理功能时，可以修改和更新刀具数据。通过预先定义的功能，可以在NC程序中：

- 赋给刀具名称和调用刀具。
- 添加一个新刀具或者删除一个现有的刀具。
- 将一个必要的T编号分配给一个有已知名称的刀具。
- 更新件数监控数据。

- 读取主轴预选刀具的T编号。

编程

T="钻头" 或者 T="123 有名称的刀具

或者

返回参数r=NEWT("WZ", DUPLO_NR)

或者

DELT("WZ", DUPLO_NR)

或者

返回参数=GETT("WZ", DUPLO_NR)

或者

SETSPIECE(x,y)

或者

GETSELT (x)

参数

T="WZ"	带名称的刀具的选择
NEWT ("WZ",DUPLO_NR)	设置新的刀具，可选择双编号
DELT ("WZ",DUPLO_NR)	删除刀具，可选择双编号
GETT ("WZ",DUPLO_NR)	决定T编号
设置工件 (x,y)	设置件数
GETSELT (x)	读取前面选出的刀具编号（T编号）
"WZ"	刀具标记
DUPLO_NR	件数
x, y	主轴编号，可选择说明

NEWT-功能举例

用NEWT功能可以在NC程序段中设置一个新刀具的名称。功能作为返回参数自动提供了生成的T编号，用这个编号可以接下来给刀具定地址。

如果没有第二编号的说明，那么这个在刀具管理中生成。

```
DEF INT DUPLO_NR
DEF INT T_NR
DUPLO_NR = 7
T_NR=NEWT("钻头", DUPLO_NR)          ;添加带有Duplo编号7的新刀具"钻头";
                                     ;所生成的T编号保存在T_NR中。
```

DELT-功能举例

用DELT功能可以不考虑T编号删除一个刀具。

GETT-功能举例

GETT功能给一个刀具提供了设置刀具数据必需的T编号，这个刀具通过名称是已知的。

如果有给出名称的几个刀具，那么送回最有可能的刀具的T编号。

返回 = –1:可以不给刀具分配刀具名称或者第二编号。

```
T="钻头"
R10=GETT("钻头", DUPLO_NR)                    ;为钻头算得的T编号;Duplo编号= DUPLO_NR
```

"钻头" 必须事先使用NEWT或者 $TC_TP1[ ] 声明。

```
$TC_DP1[GETT("钻头", DUPLO_NR),1]=100         ;写入带有刀具名称的一个刀具参数;(系统变量)
```

设置工件-功能举例

这个功能用于件数监控数据的更新。该功能用来捕捉自上次激活 设置工件 以来在已指定主轴编号上换入的所有切削刀。

设置工件 (x,y)

```
x                                             ;已加工工件的数量
y                                             ;y 主轴编号, 0 表示;主动轴(默认设置)
```

GETSELT-功能举例

该功能为主轴预选刀具提供T编号。因此在M6之前就可以存取刀具的补偿数据并且可以早一些生成和主运行的同步。

使用刀具管理器更换刀具举例

T1:
预选刀具，即可以在加工的同时将刀库置于刀具位置中。

M6:
换入预选刀具(视机床数据中的默认设置而定，也可以不用M6编程)。

```
T1 M6                          ;换入刀具1
D1                             ;选择刀具长度补偿
G1 X10 …                       ;使用T1加工
T="钻头"                       ;预选刀具 钻头
D2 Y20 …                       ;更换切削刀 T1
X10 …                          ;使用T1加工
M6                             ;换入钻头刀具
设置工件(4)                    ;已加工工件的数量
D1 G1 X10 …                    ;用钻头加工
```

注意

用于刀具管理的所有变量的完整列表可查阅

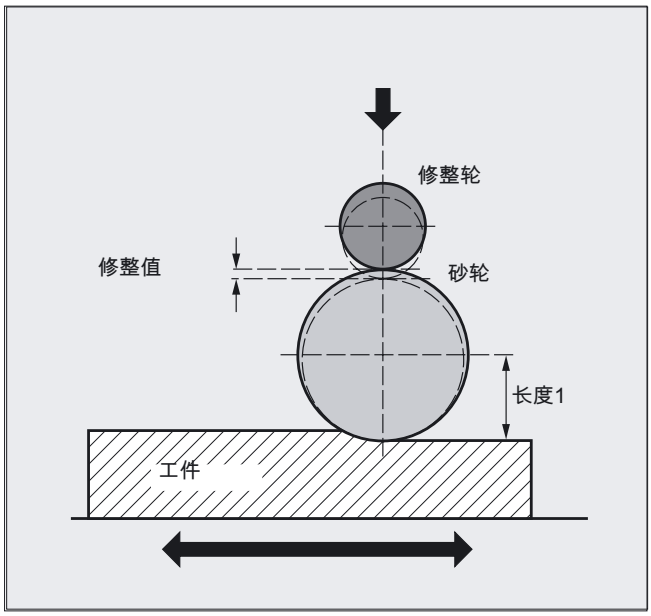
文献：

/PGA1/ 系统变量列表

8.3 在线刀具补偿 (PUTFTOCF, PUTFTOC, FTOCON, FTOCOF)

功能

使用该功能可通过在线有效的刀具长度补偿立即将从加工中得出的刀具补偿计算在内（例如修整：并行地用于加工来修整砂轮）。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。



注意

在线 - WZK只能用在磨具上。

编程

FCTDEF(Polynom-Nr., L极限, U极限,a0,a1,a2,a3)

或者

PUTFTOCF(多项式编号, 参考值, 长度1_2_3, 通道, 主轴)

或者

PUTFTOC(值, 长度1_2_3, 通道, 主轴)

或者

FTOCON

或者

FTOCOF

参数

PUTFTOCF	连续写入在线 - 刀具补偿
FCTDEF	功能PUTFTOCF的参数化
PUTFTOC	不连续写入在线 - 刀具补偿
FTOCON	在线 - 刀具补偿开通
FTOCOF	在线 - 刀具补偿关闭
多项式编号	值1至3：最多可同时有三个多项式；小于第3级的多项式

参考值	参考值，从中可以导出补偿值
长度1_2_3	磨损参数，在其中添加刀具补偿值
通道	通道的编号，在通道中刀具补偿有效；只有当不止涉及自身通道时才给出说明
主轴	主轴的编号，对于主轴在线刀具补偿有效；在没有当前有效的砂轮的情况下需要给出说明
L极限	上限值
U极限	下限值
a_0, a_1, a_2, a_3	多项式功能的系数
值	在磨损参数中添加的值

举例

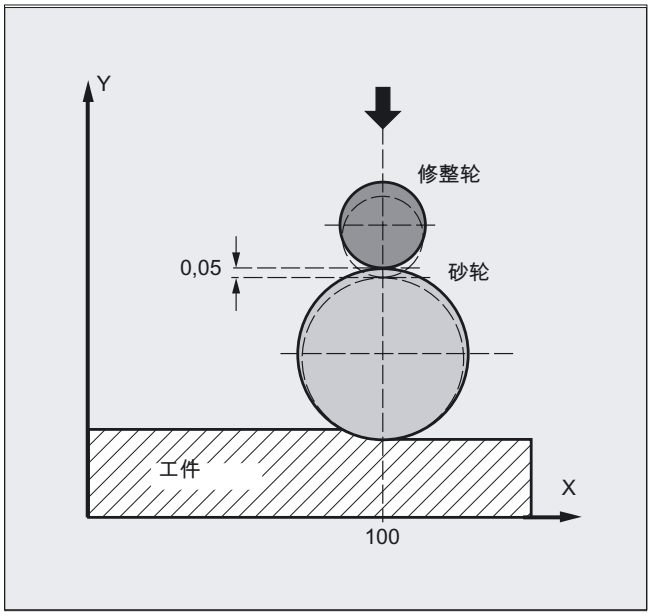
在使用带有后来确定的平面磨削机床时，在磨削运动开始之后X100时将砂轮的值修整0.05。修整量应该用在线WZK来编写，在使用磨削工具时连续有效。

Y:砂轮的横向进给轴

V:修整轮的横向进给轴

加工：通道1用轴X，Z，Y

修整：通道2用轴V



在通道1中的加工程序：

```
%_N_BEARB_MPF
…
N110 G1 G18 F10 G90                          ;基本位置
N120 T1 D1                                   ;选中当前刀具
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                                    ;向目标位置运动
N170 FTOCON                                  ;启用在线补偿
N… G1 X100                                   ;进一步加工
N…M30
```

在通道2中的修整程序：

```
%_N_ABRICHT_MPF
…
N40 FCTDEF (1, -1000, 1000, -$AA_IW[V],      ;定义功能：直线
1)
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                      ;用于修整的横向进给运动，仅在;该程序段中PUTFTOCF
                                             才有效
…
N… M30
```

修整程序模态有效：

```
%_N_ABRICHT_MPF
FCTDEF(1,-1000,1000,-$AA_IW[V],1)            ;定义功能。
ID=1 DO FTOC(1,$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                         ;用于修整的横向进给运动
G1 V-0.05 F0.02
...
取消(1)                                      ;取消在线补偿
...
```

说明

在线刀具补偿概述

分别根据修整进程的时间点，对于在线刀具补偿的写入使用不同的功能：

- 逐段连续写入：PUTFTOCF
- 连续模态写入：ID=1 DO FTOC (参见同步一章)
- 不连续写入：PUTFTOC

在使用连续写入时（每个IPO-节拍），为了避免额定值的跳跃，在计算功能开通之后每个改变在磨损存储器中是加法计算的。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。

几何轴长度的赋值根据当前平面来进行。

通过刀具数据使用 GWPSON 或者TMON将主轴分配给刀具，只要不是已激活的砂轮即可（参见编程说明书“基本部分”）。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。

注意

当对多个砂轮面进行相同的补偿时，通过关联规定（参见操作说明）可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参数值。

也可为恒定的砂轮圆周速度 (SUG) 以及刀具监控 TMON 考虑在线刀具补偿。

PUTFTOCF = 连续写入

修整进程和加工同时进行：通过砂轮总宽度用修整轮或者用修整金刚钻从砂轮的一面向另一面修整。

加工和修整可以在不同的通道中进行。如果没有编程通道，那么补偿在当前有效的通道中有效。

PUTFTOCF(多项式编号， 参考值， 长度1_2_3， 通道， 主轴)

在加工通道中连续根据第1、2或者3级的一个多项式功能来修改刀具补偿，必须使用 FCTDEF 预先定义该多项式功能。从“参考值”变量得出补偿，例如可修改的实际值。如果没有编程主轴编号，那么修正当前有效的、使用中的刀具。

确定功能FCTDEF的参数

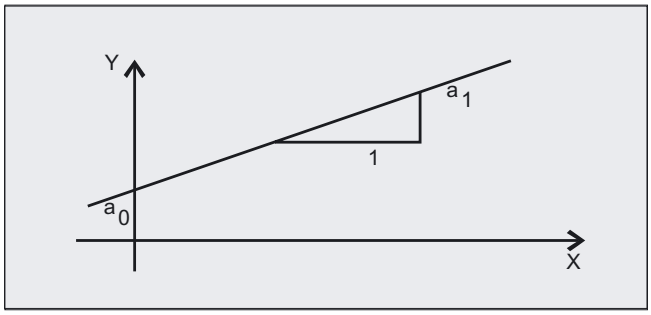
需要在一个自身的程序段中给定参数。

FCTDEF （多项式编号， L极限， U极限， a0,a1,a2,a3)

多项式可以是第1，第2或者第3级。极限表示极限值（L极限 = 下限，U极限 = 上限）

举例：带导程1的直线(y = a0 + a1x)

FCTDEF(1, -1000, 1000, -$AA_IW[X], 1)



不连续写入在线刀具补偿：PUTFTOC

使用该指令可以**一次** 写入补偿值。补偿在目标通道中立即有效。

应用 PUTFTOC：砂轮从一个并行的通道中来修整，尽管如此它和加工是不同时进行的。

PUTFTOC(值， 长度1_2_3， 通道， 主轴)

指定长度1, 2 **或者**3的在线刀具补偿可按照指定的值修改，即将数值加在磨损参数中。

考虑在线刀具补偿：FTOCON, FTOCOF

仅当 FTOCON 激活时，目标通道才能接收在线刀具补偿。

- FTOCON 必须在补偿应起作用的通道中写入。使用 FTOCOF 不会继续获得补偿，在切削刀专用补偿数据中，但是使用 PUTFTOC 完整写入的量值已经被补偿。
- FTOCOF 始终为复位位置。
- PUTFTOCF 始终逐段有效，即在紧接着的运动程序段中。
- 也可以使用 FTOC 模态选择在线刀具补偿。对此更多的信息参见“运动同步动作”章节。

8.4 恒定保持刀具半径补偿 (CUTCONON)

功能

“保持恒定刀具半径补偿”功能用来抑制一定数量程序段的刀具半径补偿，但同时也会将一个在之前程序段中通过刀具半径补偿所构成的差数作为刀具中心点已编程轨迹和实际运动轨迹之间的位移予以保留。例如，当在反向点中进行逐行铣削需要多个运动程序段、且这些运动程序段不是由刀具半径补偿生成的轮廓（环绕运动策略）所需要时，使用该功能就会有所裨益。该功能可独立于刀具半径补偿方式 ($2^1/_2$D, 3D-端面铣削, 3D-圆周铣削) 进行使用。

编程

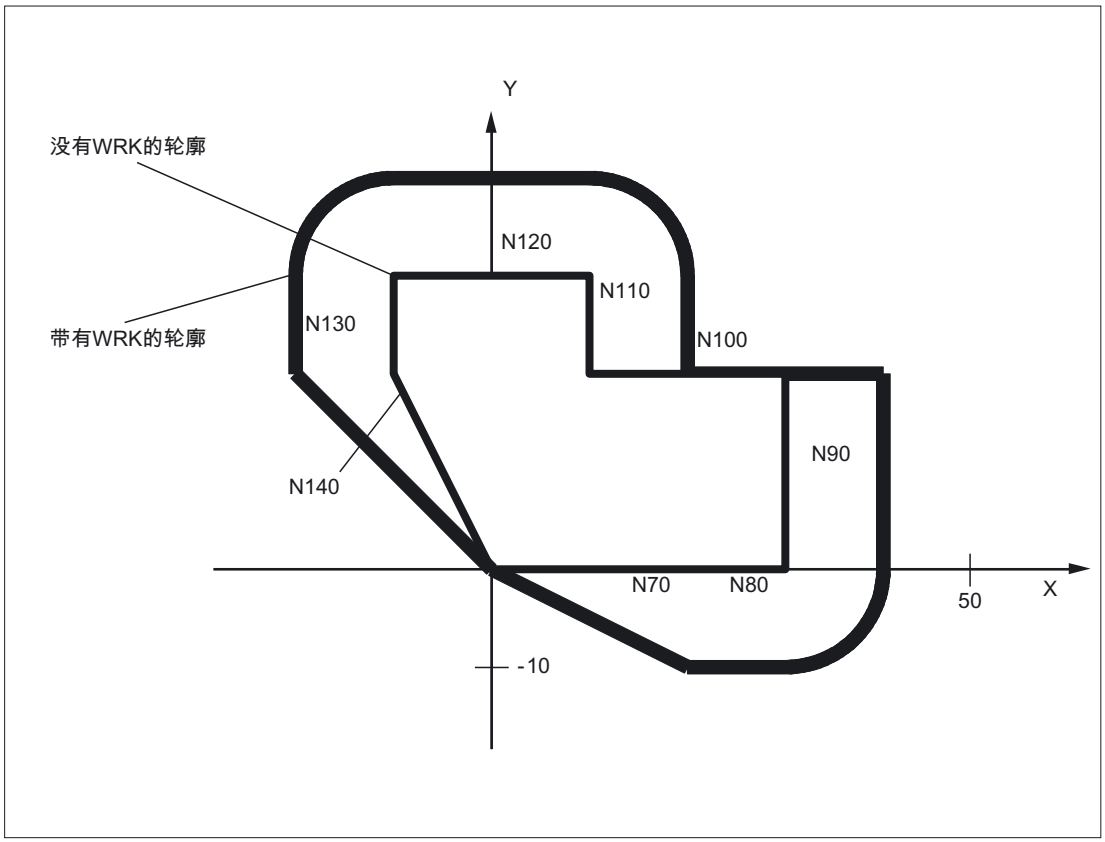
```
CUTCONON
CUTCONOF
```

参数

CUTCONON	功能刀具半径补偿开通保持恒定
CUTCONOF	功能刀具半径补偿关闭保持恒定（标准设置）

举例

```
N10                                    ;定义刀具d1
N20 $TC_DP1[1,1]= 110                  ;类型
N30 $TC_DP6[1,1]= 10.                  ; 半径
N40
N50 X0 Y0 Z0 G1 G17 T1 D1 F10000
N60
N70 X20 G42 NORM
N80 X30
N90 Y20
N100 X10 CUTCONON                      ;启用补偿抑制
N110 Y30 KONT                          ;当关闭补偿抑制时可能要;插入环绕运动圆
N120 X-10 CUTCONOF
N130 Y20 NORM                          ;关闭刀具半径补偿时没有环绕运动圆
N140 X0 Y0 G40
N150 M30
```



说明

在通常情况下在激活补偿抑制之前，刀具半径补偿已经是有效的，并且如果取消补偿抑制的话依旧有效。在 CUTCONON 之前的上一个运动程序段中向程序段终点中的偏移点运动。所有后续的，并且在其中补偿抑制当前有效的程序段可以在没有补偿的情况下运行。然而它们在运行时会从最后补偿程序段的终点偏移一定的矢量到它的偏移点。这些程序段的插补类型（线性、圆周形、多项式）为任意类型。

取消补偿抑制的程序段（即含有 CUTCONOF 的程序段）被正常修改; 该程序段在起始点的偏移点中开始。在上一个程序段的终点（即带有激活的 CUTCONON的上一个已编程的运动程序段的终点）和该点之间插入一个线性程序段。

那些圆平面垂直于补偿平面的圆形程序段（垂直的圆）被处理成就像是在其中已经编程了 CUTCONON 的形式。补偿抑制的隐含的激活在第一运行程序段中自动清除，这个运行程序段在补偿平面中包含一个运行程序段并且不是这样的圆弧。在这个意义上垂直的圆弧只可能在圆周铣削时出现。

8.5 激活 3D-刀具补偿 (CUT3DC..., CUT3DF...)

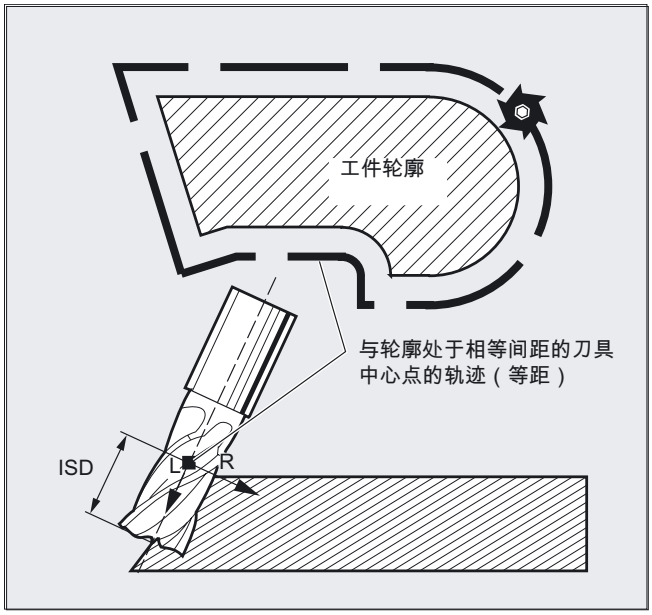
8.5.1 激活 3D-刀具补偿 (CUT3DC, CUT3DF, CUT3DFS, CUT3DFF)

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-刀具半径补偿仅当选中5轴转换时有效。

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。

编程

```
CUT3DC
```

或者

```
CUT3DFS
```

或者

```
CUT3DFF
```

或者

CUT3DF

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。

参数

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=数值	浸没深度

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 。

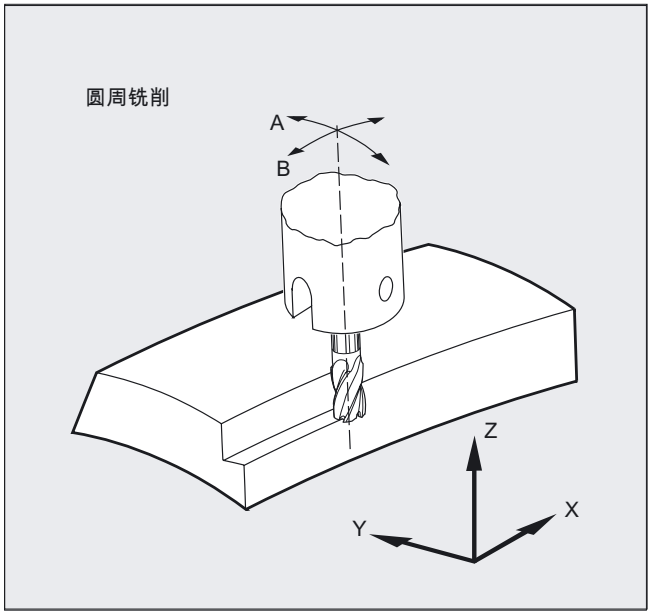
举例

```
N10 A0 B0 X0 Y0 Z0 F5000
N20 T1 D1                          ;调用刀具， 调用;刀具补偿值
N30 TRAORI(1)                      ;选择转换
N40 CUT3DC                         ;选择3D-刀具半径补偿
N50 G42 X10 Y10                    ;选择刀具半径补偿
N60 X60
N70 …
```

8.5.2 3D-刀具半径补偿：圆周铣削，端面铣削

圆周切削

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。

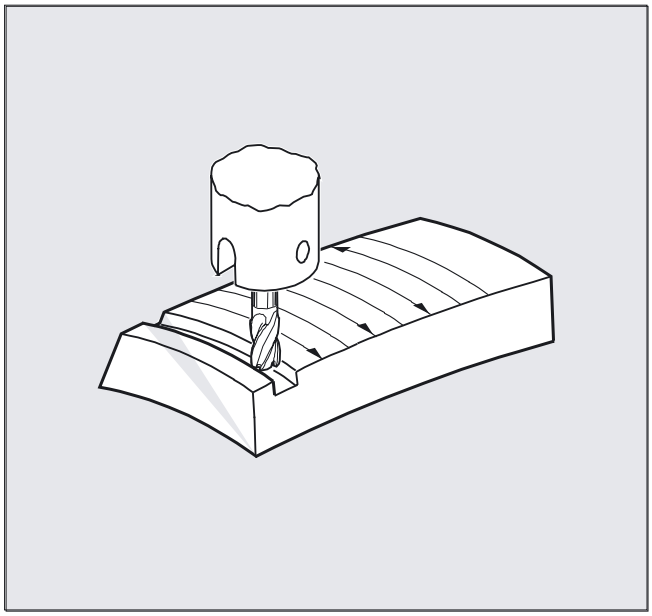


注意

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积上按行描述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。这样机床操作工就可以使用（与计算NC轨迹所使用的刀具不同的）略微小一些的刀具。



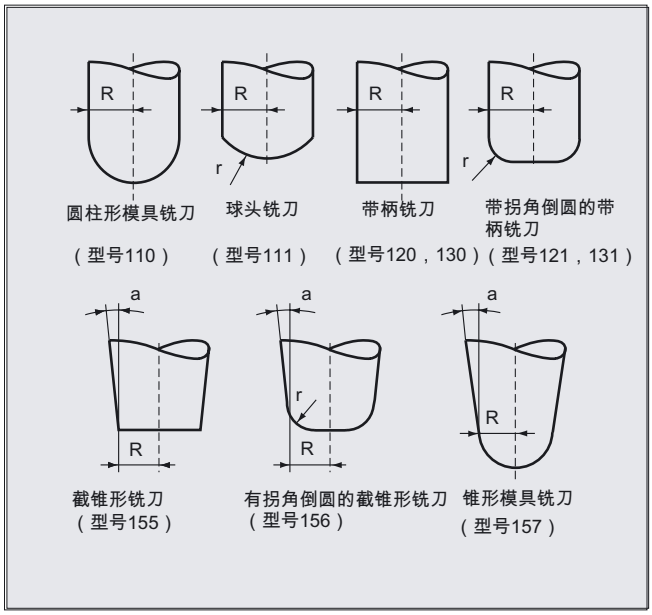
举例：
NC程序段已使用10 mm 铣刀计算过。这里也可以用直径9.9毫米的铣刀来完成，对此可以用变化了的成型材料的粗糙度来计算。

8.5.3 带有修改尺寸的刀具类型/换刀 (G40, G41, G42)

功能

铣刀形状，刀具数据

在这张表格里汇集了对于端铣有可能的刀具类型和刀具数据的极限值。不考虑刀柄的类型，刀具类型120和156作用是相同的。



如果在NC程序中说明了另一个类型编号，和表格中提供的编号不同，系统自动使用圆柱形模具铣刀的刀具类型110。超过刀具数据的极限值时产生报警。

参数

铣刀类型	类型号	R	r	a
圆柱形模具铣刀	110	>0	X	X
圆锥头铣刀	111	>0	>R	X
带柄铣刀，角度铣刀	120, 130	>0	X	X
立铣刀，带圆角功能的斜角铣刀	121, 131	>r	>0	X
截锥形铣刀	155	>0	X	>0
带有圆角功能的截锥形铣刀	156	>0	>0	>0
锥形模具铣刀	157	>0	X	>0

刀具数据	刀具参数		X = 未计算
刀具尺寸	几何尺寸	磨损	
R	$TC_DP6	$TC_DP15	R = 刀柄半径（刀具半径）
r	$TC_DP7	$TC_DP16	r = 角半径
a	$TC_DP11	$TC_DP20	a = 刀具纵轴和环面上方终点之间的角度

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

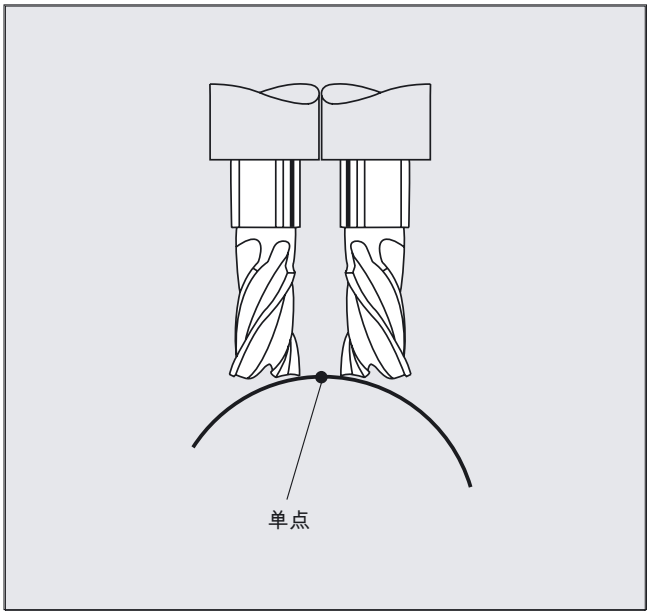
带有修改尺寸 (R, r, a) 或者其它形状的新刀具只能在编程 G41 或者G42 时说明（G40 过渡到 G41 或者G42, 重新编程 G41 或者G42).所有其它刀具数据，例如刀具长度，均不为该规则考虑，使得此类刀具也可以在没有更新 G41 或者G42 的情况下换入。

8.5.4 轨迹、轨迹曲率、浸没深度ISD和刀具进给上的补偿 (CUT3DC)

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。

编程

浸没深度 ISD

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。因此由可能在刀具的表面积上改变加工点的位置。

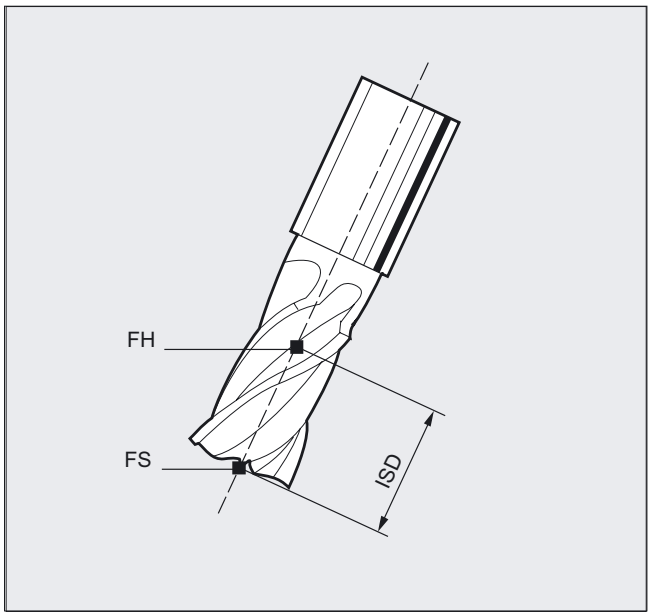
圆周铣削 3D刀具补偿

CUT3DC

参数

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	ISD 用来规定铣刀刀尖 (FS) 和铣刀辅助点 (FH) 之间的间距。

点FH通过编程设计的加工点在刀具轴上的投影产生。



说明

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。

有关碰撞监控的其它信息可参阅
文献资料：/PG/ 基本部分，“刀具补偿

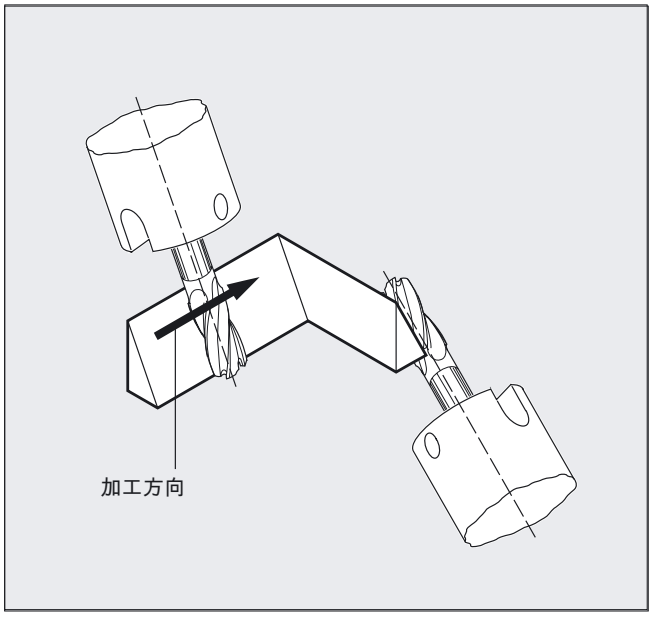
8.5.5 内角/外角和交点法 (G450/G451)

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。



编程

G450

或者

G451

参数

G450	过渡圆（刀具在一个圆形轨迹上绕工件角部运动）
G451	等距交点（刀具工具角中切削）。

说明

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动。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。

文献：

有关交点法的其它信息 /FB/ W5, 3D-刀具半径补偿。

8.5.6 带有分界面的3D圆周铣削，一般应用

功能

使3D圆周铣削适应CAD程序的实际情况

由CAD系统产生的NC程序通常情况下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。后面表示出独特的方法，为了平衡临界的过渡，要么

- 直接在零件程序中，或者
- 当计算真实轮廓时，例如通过刀具进给

补偿。

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。这里所涉及的分界面与刀具无关。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。

8.5.7 考虑一个分界面 (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 给一个真实差分刀具考虑一个可能要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。

使用 CUT3DCC 可在使用圆柱形刀具的情况下考虑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。

编程

```
CUT3DCCD
```

或者

```
CUT3DCC
```

参数

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： 向分界面进给。
CUT3DCC	激活3D刀具补偿，用于使用3D半径补偿进行带有分界面的圆周铣削： 加工面上的轮廓

注意

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。

举例

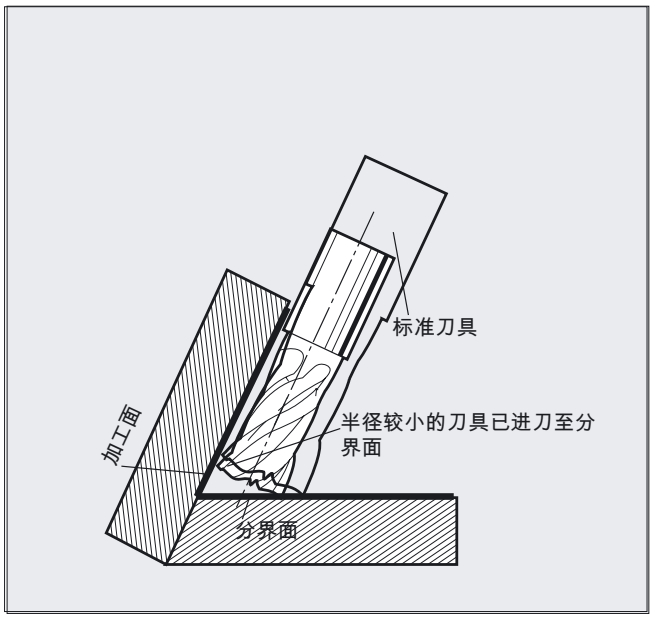
相对于标准刀具，减小半径的圆环铣刀的刀具尺寸

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_TP6	r = $TC_TP7
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_TP6 + $TC_TP15 + OFFN	r' = $TC_TP7 + $TC_TP6
在这个例子中 对刀具类型 ($TC_DP1) 进行分析。	既有 也有	$TC_TP15 + OFFN $TC_TP16 为负。
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Typ 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。		

说明

进给至分界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到再次碰到槽底。只要刀具允许，这样就可将由加工面和分界面所构成的角清除掉。在此关系到圆周铣削和端铣的加工方式。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量中得出

- 刀具半径的磨损值 (刀具参数 $TC_DP15)

和在限制面积上用于计算垂直偏移量的

- 已编程刀具偏移量 OFFN.

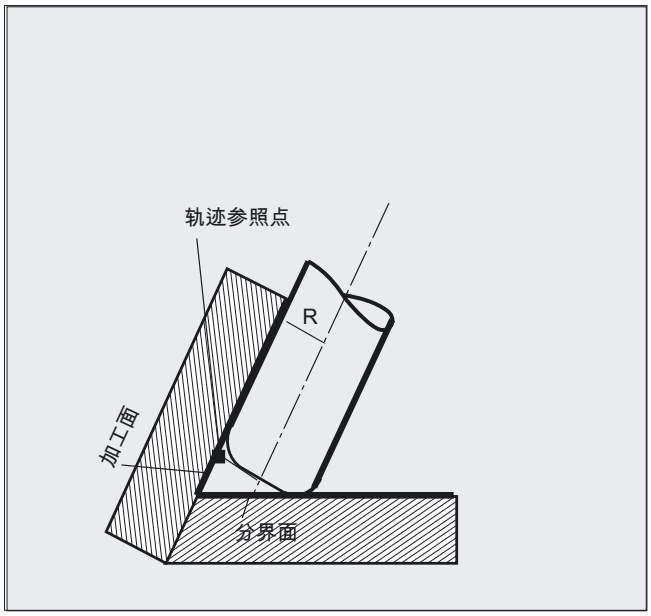
需要加工的面积在轨迹的左边或者右边，不可以从生成的零件程序中得知。因此从原始刀具的一个正半径和一个负磨损值得出。一个负向的磨损值总是描述一个缩小直径的刀具。

圆柱形刀具的使用

在使用圆柱形刀具时，只有当加工面积和限制面积构成一个锐角（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。



加工面和分界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)
- 磨损值 (刀具参数 $TC_DP15)

和在限制面积上用于计算垂直偏移量的

- 已编程刀具偏移量 OFFN.

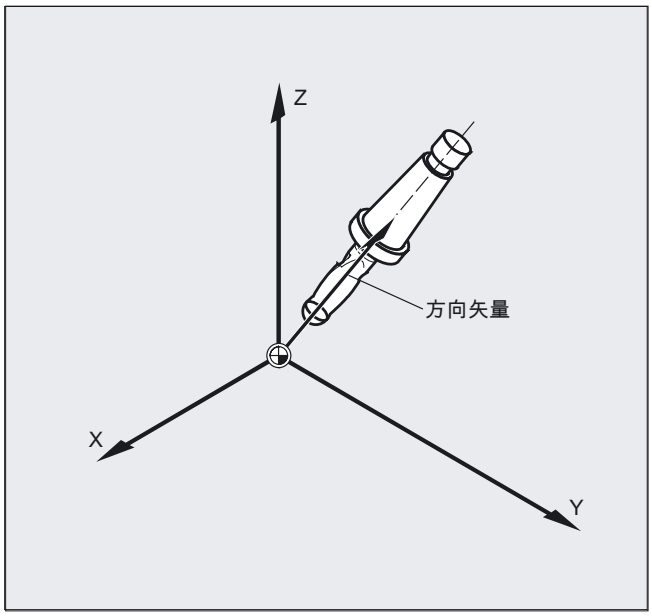
限制面积的位置由两个值的分差决定

- 标准刀具的尺寸和
- 刀具半径 (刀具参数 $TC_DP6).

8.6 刀具定向 (ORIC, ORID, OSOF, OSC, OSS, OSSE)

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。

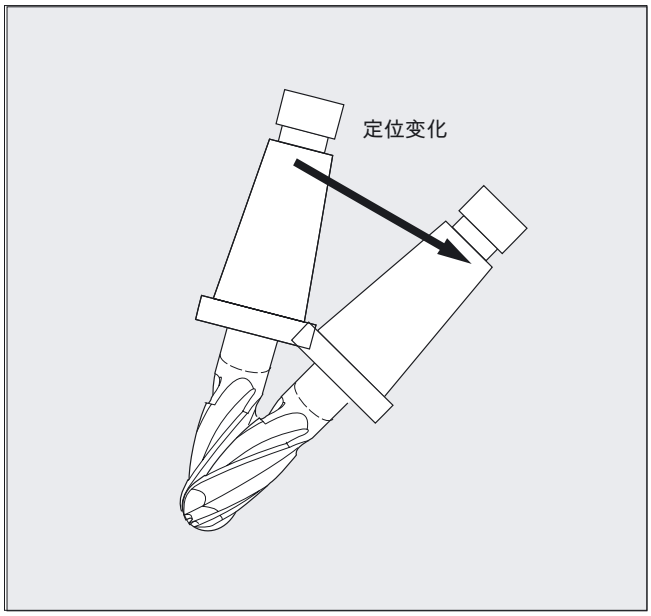


编程

刀具定向的改变可以通过以下方式来编程：

- 回转轴的直接编程
- 欧拉- 或者 RPY-角
- 方向矢量
- 导程/倾斜 (端面铣削)

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).

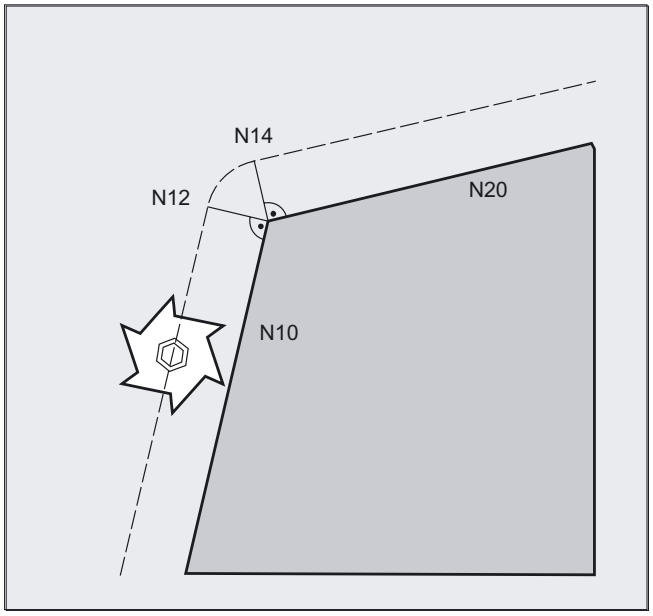


参数

ORIC	定向和轨迹运动并行
ORID	定向和轨迹运动先后进行
OSOF	没有定向平整
OSC	定向恒定
OSS	只在程序段开始处的定向平整
OSSE	在程序段开始和结束处的定向平整
ORIS	在开通的定向平整时以度每毫米表示的定向改变的速度；适用于OSS和OSSE

举例 ORIC

如果在运动程序段 N10 和 N20 之间已经编程了两个或者多个带有定向变化的程序段 (例如A2= B2= C2=) 并且 ORIC 已激活，则插入的圆程序段将根据角度变化值划分到这些中间程序段上。

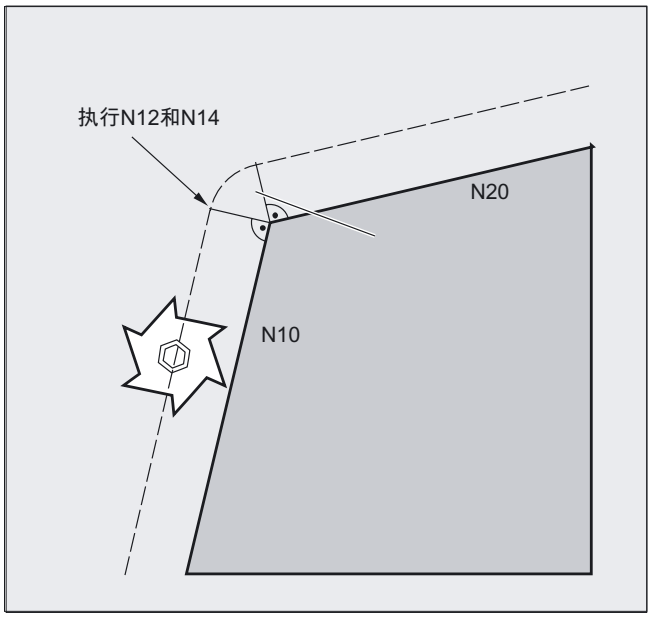



```
ORIC
N8 A2=… B2=… C2=…
N10 X… Y… Z…
N12 C2=… B2=…
N14 C2=… B2=…
N20 X =…Y=… Z=… G1 F200
```

;在外角上插入的圆程序段； 会根据定向变化分配到 N12 和 N14上 ;圆周运动和;定向变化此时会同时执行;

举例 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。带有恒定定向的圆程序段将直接在第二个运动程序段之前执行。



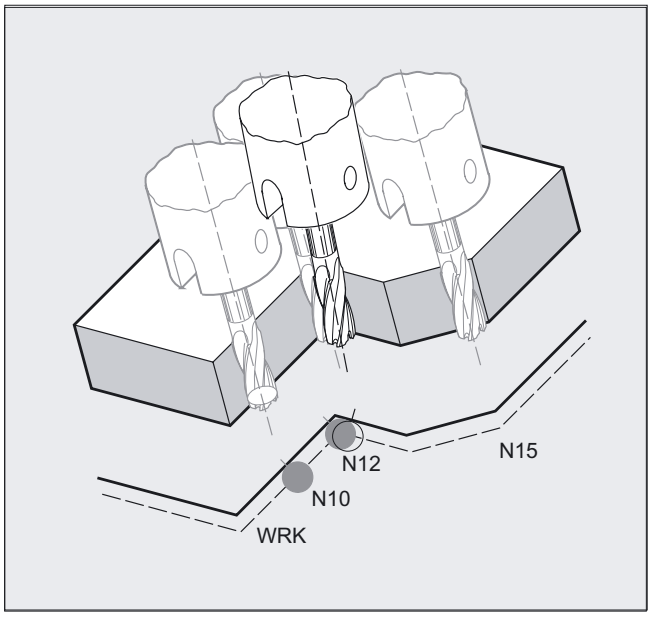

```
ORID
N8 A2=… B2=… C2=…
N10 X… Y… Z…
N12 A2=… B2=… C2=…                ; 程序段 N12 和 N14 在 N10
                                  结束时;执行。然后圆程序段将以;当前的定向执行。
N14 M20                           ;辅助功能等等
N20 X… Y… Z…
```

注意

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。

内角上定向变化举例

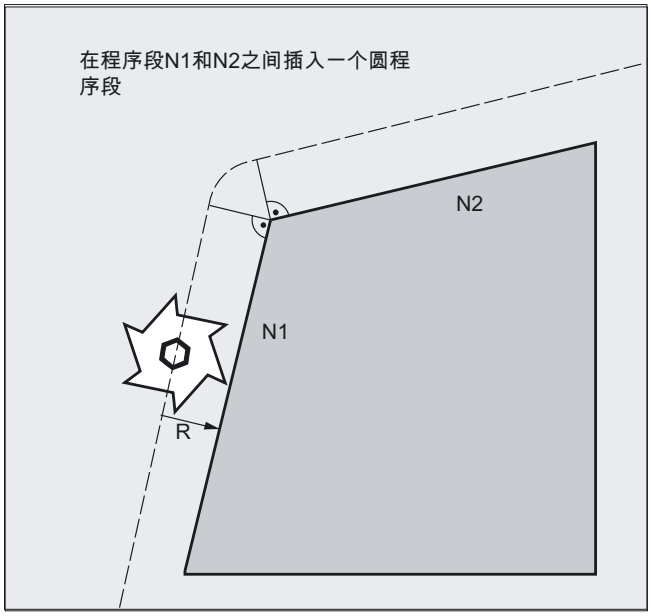



```
ORIC
N10 X …Y… Z… G1 F500
N12 X …Y… Z… A2=… B2=…, C2=…
N15 X Y Z A2 B2 C2
```

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。

使用程序指令 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，角是通过这两个程序段构成的。

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。

8.7 任意D编号赋值，切削刃编号

8.7.1 任意D编号赋值，切削刃编号 (地址 CE)

功能

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

文献：功能说明/FB1/, W1 (刀具补偿)

机床制造商

通过机床数据确定D编号的最大数量（切削刃编号）和每个刀具的最大切削刃数量。只有当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。

注意

除了相对D编号赋值之外，D编号也可作为“平面”或者“绝对”D编号 (1-32000) 在没有参照的情况下赋给一个T编号(在“平面D编号结构”功能范围内).

8.7.2 检查D编号(CHKDNO)

功能

用CKKDNO可以检查现有的D编号是否是单一分配。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。

编程

状态=CHKDNO(Tno1,Tno2,Dno)

参数

状态	正确:对于检查范围单一的分配D编号。 错误: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。

8.7.3 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D编号非法，则返回大于 32000 的某个数值。

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。

编程

```
d = GETDNO (t,ce)
状态 = SETDNO (t,ce,d)
```

参数

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

```
$TC_DP2[1.2]=120
$TC_DP3[1,2] = 5.5
$TC_DPCE[1,2] = 3; 切削编号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。

8.7.4 求得预先给出D编号刀具的T编号（GETACTTD）

功能

用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）。

编程

状态 = GETACTTD (T 号, D 号)

参数

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	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：功能可以由于一个其他的原因不执行。

8.7.5 设定无效的D编号 (DZERO)

功能

这个指令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语言指令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。

编程

DZERO

参数

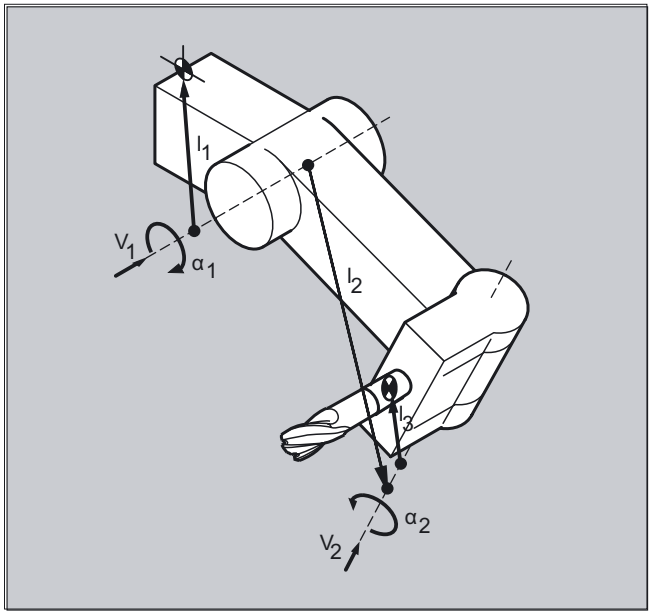
DZERO	将TO单元的所有D编号标识为无效

8.8 刀架的运动关系

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]

可定向刀架系统变量的功能			
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]		
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]		
使用者： 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]		

注意

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 $TC_CARR24 ~ $TC_CARR33

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：

偏移 ， 旋转轴 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）。

用户的参数 $TC_CARR34 ~ $TC_CARR40 包含参数，

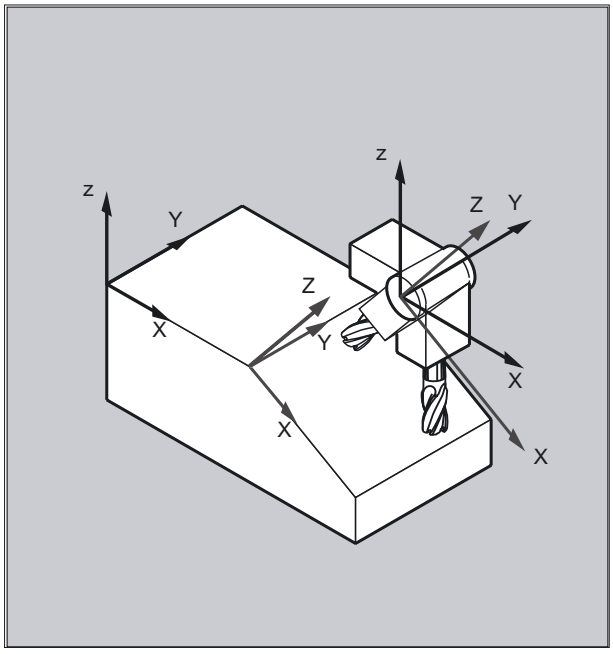
用户	可供用户任意使用并且软件版本6.4以下时默认在NCK之内不会被继续分析或者没有含义。

精密位移的参数 $TC_CARR41 ~ $TC_CARR65 包含

精细位移	可以添加给基本参数中的数值的精密位移参数。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。

举例

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。




```
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，刀架1
N80 X40                                     ;在旋转37°的情况进行加工
                                            ;
N90 Y40
N100 X0
N110 Y0
N120 M30
```

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。

说明

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] 得到扩展且描述如下：

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定数据 SD 42974：即 TOCARR_FINE_CORRECTION = 正确 。

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。

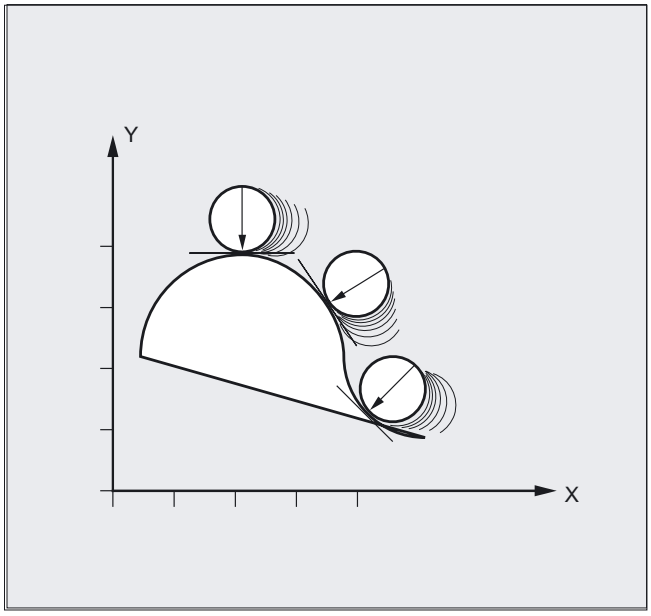
轨迹特性

9

9.1 切向控制 (TANG, TANGON, TANGOF, TANGDEL)

功能

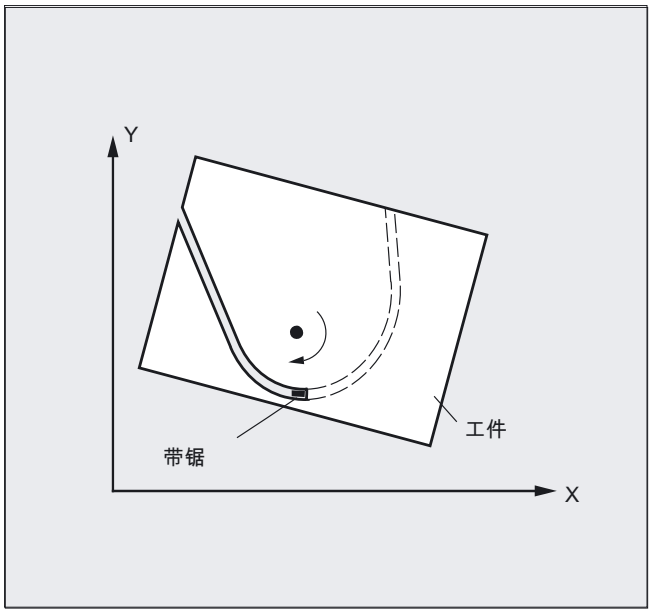
按照引导轴确定的轨迹的切线，跟随轴跟踪运行。由此一个刀具可以调整到与轮廓平行。通过TANGON指令中编程的角度，刀具可以与切线相关调用。



应用范围

切向控制此外可用于：

- 步冲时可转动的刀具切向调整
- 在使用锯条时工件矫正的跟踪（见图）
- 将一个校正刀具向砂轮调整
- 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



编程

TANG (F轴,L轴1,L轴2,偶合,KS,Opt)

或者

TANGON (F轴,角度, 间距, 角误差)

或者

TANGOF (F轴)

或者

TLIFT (F轴)

或者

TANGDEL (F轴)

简化编程：

不必精确编程偶合系数1。

TANG(C, X, Y, 1, "B", "P") 可以简写成 TANG(C, X, Y, , , , "P").与以往一样，可以将 TANG(C, X, Y, 1, "B", "S") 写成 TANG(C, X, Y).

TLIFT(...)-指令在接着使用TANG(...)分配轴时说明。举例：

```
TANG(C, X, Y...)
TLIFT(C)
```

TLIFT关断

为此可重复轴分配 TANG(...) 不使用下列 TLIFT(...).

TANGDEL 删除一个切向跟踪的定义

现有的用户自定义切向跟踪必须删除，当要用相同的跟随轴在准备调用 TANG 的过程中定义一个新的切向跟踪时。只有当使用 TANGOF(F轴) 取消偶合后，才能删除。

参数

TANG	定义一个切向跟踪的准备指令；默认设置： 1
	TANG(C,X,Y,1,"B") 表示： 圆轴 C 跟随几何轴 X 和 Y。关闭TLIFT
TANGON	以指定跟随轴和所需的跟随轴偏移角度来启用切向控制，可能还要有修整行程，
	TANGON(C,90) 表示： C轴为跟随轴。它在每次轨迹轴的运动中向轨迹切线转动90度位置。
TANGOF	以指定跟随轴来关闭切向控制。
	切向控制系统的关断，说明跟随轴： **TANGOF(C)**
TLIFT	在轮廓角上插入中间程序段。
TANGDEL	删除一个切向跟踪的定义 **举例：TANGDEL(F轴)**
F轴	跟随轴切向跟踪运行的附加回转轴。
L轴1, L轴2	引导轴轨迹轴，确定跟踪的切向。
偶合	耦合系数切线角度的变化与跟踪轴之间的关系 可选说明；默认设置： 1
KS	坐标系的标记字母 "B" = 基本坐标系；可选说明；默认设置
Opt	优化： "S" 标准，缺省
	"P" 自动调整切向轴的时间曲线和轮廓
角度	跟随轴的偏移角
Dist	跟随轴的磨削行程，在Opt揭揆
角度误差	跟随轴的角度公差，（选件）， 仅在Opt=揭揆

优化方法 Opt, Dist 和角度误差

使用 Opt="P" 可在限制引导轴速度时一并考虑跟随轴的运动，且特别当插入运动转换时推荐使用。

参数 (Dist 和 Winkeltol) 可有针对性地限制被跟踪轴和引导轴切线之间的误差。

平面转换举例

```
N10 TANG(A, X, Y,1)          ;1. Tang的定义跟随运行
N20 TANGON(A)                ;激活偶合
N30 X10 Y20                  ；半径
```

```
...
N80 TANGOF(A)                    ;关闭第一个偶合
N90 TANGDEL(A)                   ;删除第1个定义
...
TANG(A, X, Z)                    ;2. Tang的定义跟随运行
TANGON(A)                        ;激活新的偶合
...
N200 M30
```

几何轴转换和TANGDEL举例

不产生报警。

```
N10 GEOAX(2,Y1)                  ;Y1 为几何轴2
N20 TANG(A, X, Y)
N30 TANGON(A, 90)
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)                ;2. Tang的定义跟随运行
N90 TANGON(A, 90)                ;激活带有第2个定义的跟踪
...
```

带有自动优化的切向跟踪举例

自动优化 通过 Dist 和角度误差

```
N80 G0 C0                        ;Y1 为几何轴2
N100 F=50000
N110 G1 X1000 Y500
N120 TRAORI                      ; 带轴向公差的精磨削
N130 G642
N171 TRANS X-Y-                  ;自动优化轨迹速度
N180 TANG(C,X,Y, 1,,"P")         ;修整行程 5 mm,
N190 TANGON(C, 0, 5.0, 2.0)      ;角度误差2度
N210 G1 X1310 Y500               ;激活带有第2个定义的跟踪
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
```

定义引导轴和跟随轴

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被跟踪轴之间的关系。通常其数值为1（预调）。

在基本坐标系 "B"中跟踪。在切向控制中，选项参数 KS 仅可供 "B" 坐标系使用。
工件坐标系的说明 "W" 使用报警16795 "STRING不可解释" 来应答。

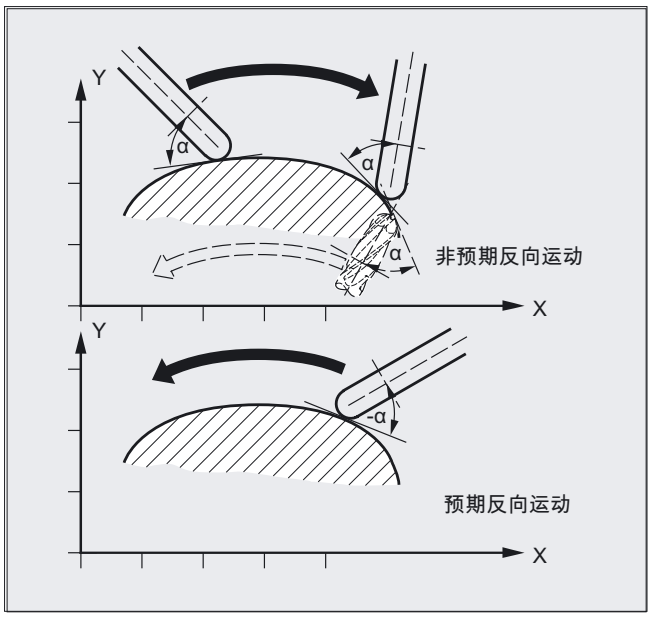
通过工作范围限制极限角度

在来回导入的轨迹运动中，切线在轨迹换向点上转向180度， 相应地跟随轴的矫正发生改变。

通常这样的行为无效应在与前进运动相等的反向偏移角度执行后退运动。

对此您可限制跟随轴的工作范围 (G25, G26).工作范围限制必须当轨迹反转时激活 (WALIMON) 。

当偏移角在工作范围限制之外时，尝试将负偏移角重新带入容许的工作范围内。



在轮廓拐角处添加中间程序段，TLIFT

在轮廓拐角处切线改变，从而被跟踪的轴的额定位置变得不稳定。轴在通常情况下，尝试以其可能的最大速度平衡这种变化在此情况下在轮廓上拐角下，相对于所期望的切向定位，出现超过一定距离的误差。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` 来命令控制装置停在拐角上并且在一个自动生成的中间程序段中将被跟踪的轴旋转到新的切向方向中。

如跟随轴某一次被作为轨迹轴运行，则编程的轨迹轴旋转。通过功能 `TFGREF[ax] = 0.001` 可以在此达到被跟踪轴的最大轴速度。

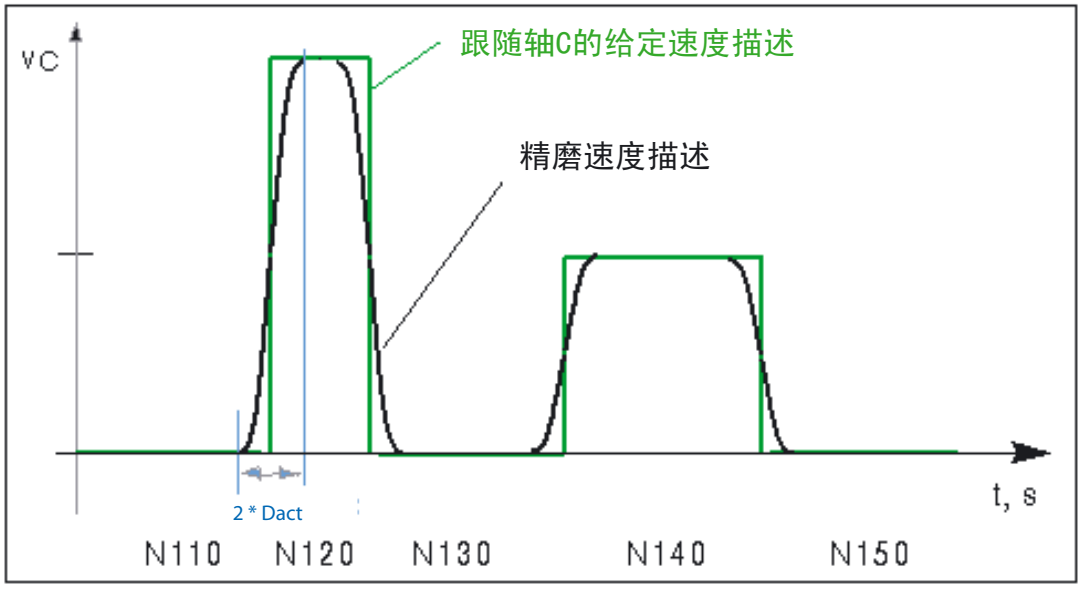
到现在为止被跟踪轴没有被作为轨迹轴，而是作为定位轴处理。这样，速度依赖于由机床数据确定的定位速度。

被跟踪轴进行最大速度转动

优化方法

由于引导轴轮廓的变化而引起的跟随轴的速度变化，通过（距离和角度公差）平滑。

在此将跟随轴带预见性地导入（见图表），以使误差尽可能减到最小。



定义角度变化

通过机床数据文件 $MA_EPS_TLIFT_TANG_STEP 来定义角度变化，从角度开始变化起自动插入一个中间程序段。

对转换进行干预

被跟踪的回转轴位置可以成为用于转换的输入值。

跟随轴的明确定位

如一跟随引导轴运行的跟随轴明确定位，则定位参数附加影响编程的偏移角。

允许所有的位移规定：轨迹轴和定位轴运行。

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: 无耦合有效

1,2,3: 切向跟踪运行有效

9.2 联动 (TRAILON, TRAILOF)

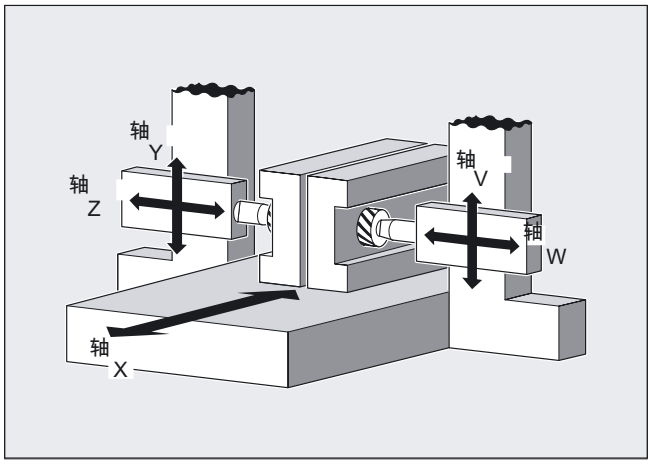
功能

当一个定义过的主动轴运动时，分配给该轴的联动轴(= 从动轴) 会在考虑到某个偶合系数的情况下，经过引导轴所留下的运动行程。

引导轴和跟随轴共同组成联动轴组合。

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，联动轴是一个真正的轴。真实轴运行，考虑到耦合系数。
- 用两个联动轴组合加工两个面：
  第1个引导轴Y, 联动轴V
  第2个引导轴Z, 联动轴W



编程

TRAILON(F轴,L轴,偶合)

或者

TRAILOF(F轴,L轴, 轴2)

TRAILON 和 TRAILOF 为模态有效。

参数

TRAILON	激活并且定义联动轴 **举例：**V = 联动轴, Y = 引导轴 TRAILON(V,Y)
TRAILOF	关断联动组合 **举例：**V = 联动轴, Y = 引导轴 TRAILOF(V,Y) 有两个参数的TRAILOF只关闭第一个引导轴的耦合。如果一个联动轴拥有两个引导轴，例如 V=联动轴和 X,Y=引导轴, 则关闭偶合时可调用带有三个参数的TRAILOF： TRAILOF(V,X,Y)

F轴	联动轴的轴名称 一个联动轴也可以是其余联动轴的引导轴。以这种方式可以建立不同的联动组合。
L轴	引导轴的名称
偶合	耦合系数 = 联动轴行程/ 引导轴行程 预设置 = 1

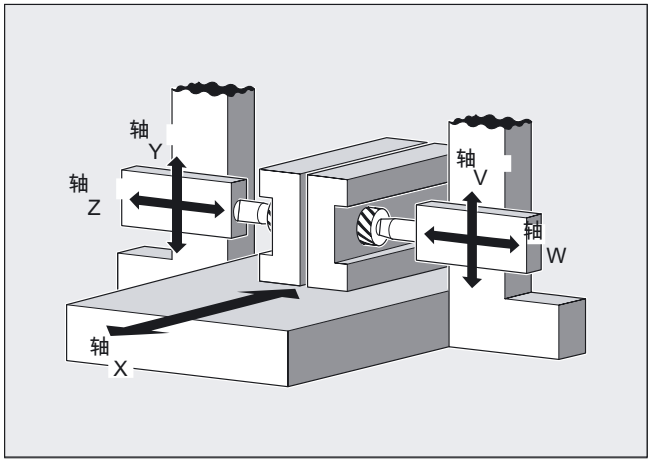
注意

联动始终在基准坐标系 (BKS)中进行。

可同时激活的联动组合的数量只由机床上所存在的轴的组合可能性限制。

举例

工件两面须根据展示的轴的情况加工。据此构成两个联动组合。




```
…
N100 TRAILON(V,Y)          ;启用第1个联动组合
N110 TRAILON(W,Z,-1)       ;启用第2个联动组合 偶合系数为负：;联动轴以与引导轴相反 ;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    ;叠加一个相关和不相关的联动轴"V"的运动
…
TRAILOF(V,Y)            ;关闭第1个联动组合
TRAILOF(W,Z)            ;关闭第2个联动组合
```

关联的轴的类型

联动组合由线性轴和回转轴任意组合构成。一个模拟轴也可在此被定义为引导轴。

联动轴

一个联动轴最多可同时被两个引导轴分配。这种分配通过不同的联动组合实现。

一个联动轴可以用所有可支配的运动指令编程（G0,G1,G2,G3,…)。此外与已定义位移无关的是，联动轴运行其耦合系数从引导轴导出的位移。

耦合系数

耦合系数说明了联动轴和引导轴位移之间的要求关系。

公式： 偶合系数 = 联动轴的行程 / 引导轴的行程

如在编程中耦合系数未被说明，则耦合系数1自动有效。

带小数点的数据将作为分数输入（类型 实数）。输入负数值，表明引导轴和联动轴在相反方向运行。

加速度和速度

属于耦合的轴的加速度极限和速度极限由联动组合中"最弱的轴"决定。

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: 无耦合有效

8: 联动轴有效

9.3 曲线图表 (CTAB)

9.3.1 曲线图表：一般关系

功能

在曲线图表一章中，您可查阅可用来对两个轴（引导轴和跟随轴）之间的关系进行编程的编程指令。

在引导值的一个定义好的数值范围内，可以明确给每个引导值分配跟随值。如果引导值在定义范围之外，就可为周期性和非周期性曲线图表编程曲线图表边缘上的特性。

说明

机械凸轮可通过曲线图表替代，用这些图表可以定义

- 某个定义范围内的特定曲线
- 各个曲线段，也就是凸轮段
- 周期性和非周期性曲线图表的曲线边缘
- 有关的凸轮段位置

在一个定义好的数值范围内，可以从

- 有关的图表位置以及
- 一个图表曲线段的起点和终点值

中读取相应的跟随值给一个引导值，同样也可将引导值读取给跟随值。

显示所有其它形状，将可选参数分配给相应的编程指令。由此得出的控制单个或者多个曲线图表进入相应存储器类型的方法可为其它应用进行灵活的编程。这样也可为诊断轴偶合规定大量的编程方法。

关于曲线图表的定义和对曲线图表位置的访问，有相应的典型编程举例说明。

9.3.2 曲线图表 重点功能 (CTABDEF, CATBEND, CTABDEL)

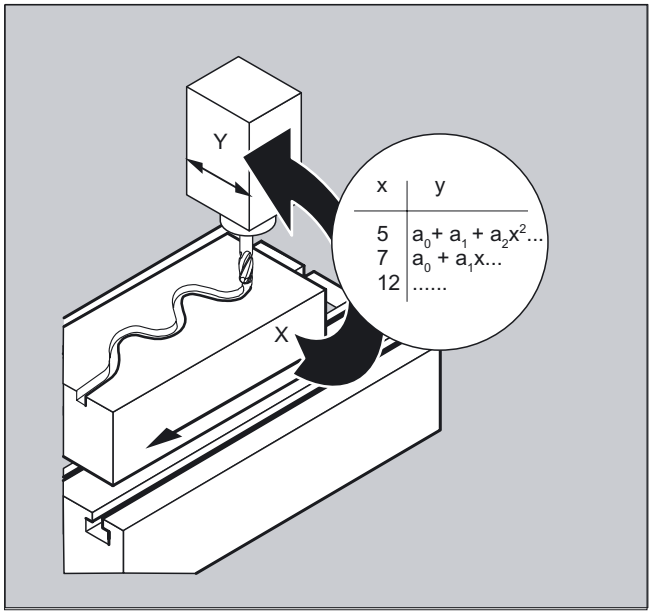
功能

可以借助曲线图表编程两轴之间的位置和速度关系。在一个零件程序中定义曲线图表。

替换机械凸轮的**举例** ：

通过实现引导值和跟随值之间的 功能性关联，曲线图表由此构成轴向引导值耦合的基础。

在相应的编程时，从相互所属的引导轴和跟随轴的位置中，控制系统计算出一个与曲线字盘相应的多项式。



编程

模态有效的语言指令与曲线图表

CTABDEF (F轴,L轴,n,applim, mem类型)

或者

CTABEND ()

或者

CTABDEL ()

参数

重点功能

CTABDEF (F轴,L轴, n,applim, mem类型)	定义曲线图表开始。
CTABEND ()	定义曲线图表末尾。
CTABDEL ()	删除所有曲线图表，**不考虑存储器类型**
F轴	跟随轴 跟随轴是通过曲线图表编程的轴。
L轴	引导轴 通过引导轴编程引导值。

n, m	曲线图表的编号； n < m，例如 CTABDEL(n, m) 曲线图表的序号是明确的，并且与存储器类型无关。在SRAM和DRAM中不能存在相同序号的图表。
applim	图表周期性的标志： 图表为非周期性 图表涉及引导轴时为周期性 图表涉及引导轴和跟随轴时为周期性
存储器类型	NC的存储器类型的选择性说明："DRAM" / "SRAM" 如对这些参数没有编程数值，则使用通过MD 20905: CTAB_DEFAULT_MEMORY_TYPE设置的标准存储器类型。

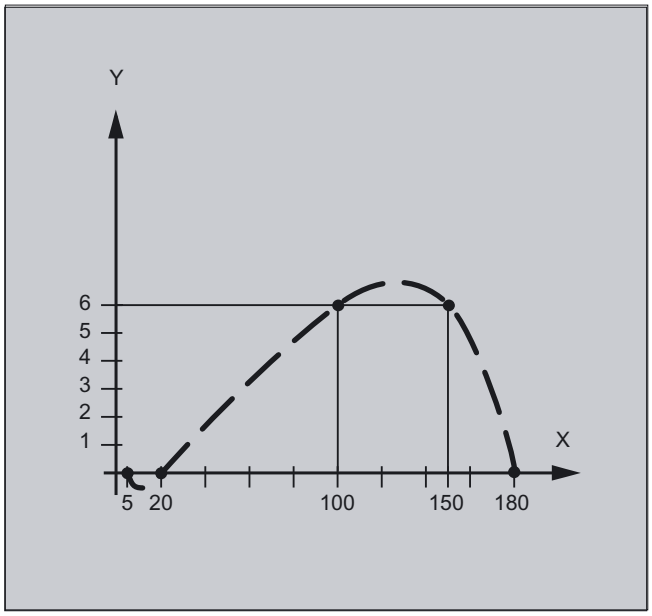
机床制造商
为了建立曲线图表，必须通过对机床数据进行相应的设定来预留存储位置。

CTABDEF 和 CTABEND 的使用举例

不改变一个程序段，用于定义一个曲线图表。其中所出现的进给停止指令STOPRE可能停止执行且会立即重新激活，只要不将该程序段用作图表定义、 CTABDEF 和 CTABEND 未曾删除。

```
…                                                    ;定义一个曲线图表
CTABDEF(Y,X,1,1)
…
…
IF NOT ($P_CTABDEF)
STOPRE
ENDIF
…
…
CTABEND
```

定义一个曲线图表举例




```
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
                               ;定义一个曲线图表
N110 X0 Y0                     ;1. 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                    ;2. 支点：引导值：0…20
                               ;跟随值：起始值…0
N130 X100 Y6                   ;3. 支点：引导值：20…100
                               ;跟随值： 0…6
N140 X150 Y6                   ;4. 支点：引导值：100…150
                               ;跟随值： 6…6
N150 X180 Y0                   ;5. 支点：引导值：150…180
                               ;跟随值： 6…0
N200 CTABEND                   ;定义结束; 曲线图表
                               ;在其内部表达中被生成为
                               ;多项式，至多为三阶多项式;
                               ;使用指定的节点计算曲线时
                               ;取决于
                               ;模态选择的插补方式
                               ;(圆形插补，线性插补，样条插补)
                               ;开始定义之前的零件程序状态
                               ;被重新恢复。
```

一个周期性曲线图表的定义举例

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：

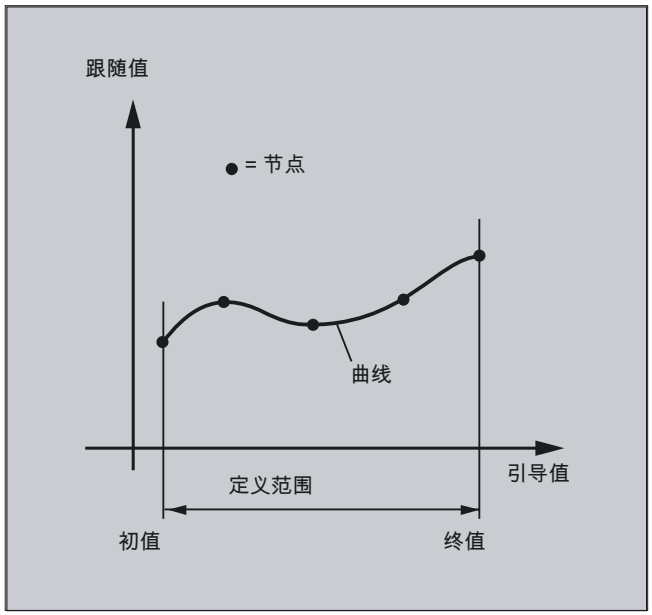
```
N10 DEF REAL DEPPOS
N20 DEF REAL GRADIENT
N30 CTABDEF(Y,X,2,1)                          ;开始定义
N40 G1 X=0 Y=0
N50 POLY
N60 PO[X]=(45.0)
N70 PO[X]=(90.0) PO[Y]=(45.0,135.0,-90)
N80 PO[X]=(270.0)
N90 PO[X]=(315.0) PO[Y]=(0.0,-135.0,90)
N100 PO[X]=(360.0)
N110 CTABEND                                  ;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)

在引导值75.0时读图表功能：
N170 DEPPOS=CTAB(75.0,2,GRADIENT)

引导轴和跟随轴的定位：
N180 G0 X75 Y=DEPPOS

启用偶合之后，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
```

定义曲线图表

CTABDEF, CTABEND

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，通过运动指令将唯一的跟随轴位置分配给引导轴的各个位置，这些跟随值位置用来作为计算曲线的节点，曲线的形式至多为五阶多项式。



曲线图表的初值和终值

作为曲线图表定义范围开始处初值的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。

在曲线图表的定义内，可使用整个NC语言范围。

所有在曲线图表定义之内所作出的模态有效指令均在曲线图表定义结束处失效。进行图表定义的零件程序，在图表定义的前后处于相同的状态。

注意

不允许：

进刀停止

引导轴运动过程中的跳转（例如当交替转换时）

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

ASPLINE, BSPLINE, CSPLINE的激活

如果在曲线图表内 CTABDEF( ) ...CTABEND 激活 ASPLINE, BSPLINE 或者 CSPLINE，则应在这种样条激活之前至少编程一个起始点。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。

举例：

```
...
CATBDEF(Y, X, 1, 0)
X0 Y0
ASPLINE
X=5 Y=10
X10 Y40
...
CTABEND
```

取决于机床数据 MD 20900:CATB_ENABLE_NO_LEADMOTION 可以在引导轴运动中断时允许跟随值跳转。其他在说明中提到的限制继续有效。

在图表的设置和删除中可以使用NC存储器类型的参数。

删除曲线图表, CTABDEL

使用 CTABDEL
可以删除曲线图表。在轴耦合中有效的曲线图表不能被删除。如果在某个偶合中至少有一个曲线图表从多重删除指令 CTABDEL( ) 或者 CTABDEL(n, m) 中激活，则**不** 删除已设定地址的曲线图表。一个特定存储器类型的曲线图表可以选择指定存储器类型来删除，参见“曲线图表形式 (CTABDEL, ...CTABUNLOCK”一章。

9.3.3 曲线图表形式 (CTABDEL, CTABNOMEM, CTABFNO, CTABID, CTABLOCK, CTABUNLOCK)

功能

曲线图表的其它应用为：

- 在某个特定的存储器类型 SRAM 或者 DRAM 中删除
- 确定存储器类型中**已定义** 和**可能有的** 曲线图表的数量
- **阻止** 删除或者覆盖曲线图表或者重新**取消**阻止
- 可指定选择，如删除
  一个 曲线图表，删除指定存储器中**所有** 曲线图表的
  一个 曲线图表范围，
  和阻止被覆盖以及重新**取消阻止**.
- 提供、返回和检查轴偶合的诊断说明，如某些曲线图表属性确定曲线图表、曲线段和曲线多项式的数量

编程

模态有效的语言指令与曲线图表

CTABDEL(n, m, mem类型)

或者

CTABNOMEM (mem**类型**)

或者

CTABFNO(mem**类型**)

或者

CTABID(n, mem**类型**)

或者

CTABLOCK(n, m, mem**类型**) **或者** CTABUNLOCK(n, m, mem**类型**)

或者

CTABDEL(n) **或者** CTABDEL(n, m)

或者

CTABLOCK(n) **或者** CTABLOCK(n, m) **或者** CTABLOCK() **或者** CTABLOCK(, , mem**类型**)

或者

CTABUNLOCK(n) **或者** CTABUNLOCK(n, m) **或者** CTABUNLOCK() **或者** CTABUNLOCK(, , mem**类型**)

或者

CTABID(n, mem**类型**)CTABID(p, mem**类型**) **或者** CTABID(n)

或者

CTABISLOCK(n)

或者

CTABEXIST(n)

或者

CTABMEMTYP(n)

或者

CTABPERIOD(n)

或者

CTABSEG(mem**类型**) **或者** CTABSEGID(n) **或者** CTABFSEG(mem**类型**) **或者** CTABMSEG(mem**类型**)

或者

CTABPOLID(n) **或者** CTABMPOL(mem**类型**)

参数

一般形式，在存储器类型SRAM或者DRAM中：

CTABDEL(n, m, mem**类型**)	删除存储在mem类型中的曲线图表范围中的曲线图表。
CTABNOMEM (mem**类型**)	**已定义** 曲线图表的数量。
CTABFNO(mem**类型**)	**可能** 有的图表数量
CTABID(n, mem**类型**)	提供在存储器类型中作为第n个 曲线图表登记的图表序号。
CTABLOCK(n, m, mem**类型**)	**设定** 防止删除和覆盖的**阻止**功能。

mem类型)	
CTABUNLOCK(n, m, mem类型)	**取消** 防止删除和覆盖的**阻止**功能。 CTABUNLOCK使能用CTABLOCK禁止的图表。在激活的耦合中有效的图表继续保持禁止状态并不能被删除。只要通过使激活的耦合失效取消禁止，CTABLOCK的禁止就失效。从而可以删除此图表。再次CTABUNLOCK调用是不必要的。

其它形式的应用可指定选择：

CTABDEL(n)	删除**一个** 曲线图表。 删除**一个** 曲线图表范围。
CTABDEL(, , mem类型)	删除指定存储器中的**所有** 曲线图表。
CTABLOCK(n)	删除**和 覆盖 阻止**功能： n号的曲线图表
CTABLOCK(n, m)	禁止从n到m范围的曲线图表。
CTABLOCK()	所有已存在的曲线图表。
CTABLOCK(, , mem类型)	**在**存储器类型中的所有曲线图表。
CTABUNLOCK(n)	删除**和覆盖取消**功能：n号的曲线图表。
CTABUNLOCK(n, m)	将从 n 到 m 数字范围内的曲线图表重新启用。
CTABUNLOCK()	所有已存在的曲线图表。
CTABUNLOCK(, , mem类型)	**在**存储器类型中的所有曲线图表。

其它形式的应用用于诊断轴偶合：

CTABID(n, mem类型)	提供 n-/p-ten 曲线图表的图表编号，**存储器类型** mem类型.
CTABID(p, mem类型)	
CTABID(n)	提供第n号的曲线图表，通过MD 20905:CTAB_DEFAULT_MEMORY_TYPE **已确定的** 存储器类型。
CTABISLOCK(n)	返回 **带有编号 n**的曲线图表的锁定状态。
CTABEXIST(n)	**检查** 编号为n 的曲线图表
CTABMEMTYP(n)	返回在其中建立了编号为n 的曲线图表的**存储器** 。
CTABPERIOD(n)	返回**图表周期性**.
CTABSEG(mem类型)	相关存储器类型中已经**使用的** 曲线段的数量。
CTABSEGID(n)	使用n号的曲线图表的数量。曲线段。
CTABFSEG(mem类型)	**可能有的** 曲线段的数量。
CTABMSEG(mem类型)	**最多** 可能有的曲线段的数量。
CTABPOLID(n)	**编号为n** 的曲线图表的已使用数量。曲线多项式。
CTABFPOL(mem类型)	相关存储器类型中**可能还有的** 曲线多项式的数量。
CTABMPOL(mem类型)	相关存储器类型中最多可能还有的**曲线多项式** 的**数量**。

n, m	曲线图表的编号; n < m，例如 CTABDEL(n, m) 曲线图表的序号是明确的，并且与存储器类型无关。在SRAM和DRAM中不能存在相同序号的图表。
存储器类型	NC存储器类型的可选择性参数："DRAM" / "SRAM" 如对这些参数没有编程数值，则使用通过MD 20905: CTAB_DEFAULT_MEMORY_TYPE设置的标准存储器类型。

说明

通过"外部处理"载入曲线图表

对曲线图表进行外部处理时，必须通过 MD 18360:MM_EXT_PROG_BUFFER_SIZE 来选择加载缓存（DRAM）的容量，只得所有曲线图表定义能同时保存在加载缓存中。否则警报15050将停止零件程序的加工。

重复使用曲线图表

如果表格存储在静止存储器（SRAM)中，则通过曲线图表计算出的、引导轴和跟随轴的功能性关系保持在所选择的图表号之下，与零件程序结束和关机无关。

保存在动态存储器（DRAM）中的图表会在通电时被删除，可能要再次加载。

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，并且和哪些轴被用来建立曲线图表没有关系。

覆盖曲线图表

只要一个新曲线图表定义时其序号被使用，这个曲线图表将被改写。

例外：曲线图表已在某个轴偶合中激活或者已使用 CTABLOCK( ) 锁定。

注意

在曲线图表的改写中不给出相应的警告。

使用系统变量 $P_CTABDEF 可随时从信息帧中查询曲线图表是否已激活。

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。

9.3.4 曲线图表边缘上的特性 (CTABTSV, CATBTSP, CTABMIN, CTABMAX)

功能

如果引导值在定义范围之外，则可以由跟随轴读取曲线图表开始和结束处的数值。

使用 CTABTSV 可以由**跟随轴** 读取曲线图表 **开始** 处的数值。使用 CTABTEV 可以由**跟随轴** 读取曲线图表**结束** 处的数值。

一个曲线图表的始值和终值与其用上升或是下降的引导值定义无关。始值总由范围下限，终值总由范围上限确定。

使用 CTABTMIN 和 CTABTMAX 可以在整个范围内或者在一个定义好的区间内确定曲线图表的**最小** 或者 **最大值** 。对引导值的相应的范围将给出两个界限。

编程

跟随值 初值和终值 跟随轴：

```
CTABTSV(n, grad, F轴), CTABTEV(n, grad, F轴)
```

引导值 初值和终值 引导轴：

```
CTABTSP(n, 度, F轴), CTABTEP(n, 度, F轴)
```

值范围，最小和最大：

```
CTABTMIN(n, F轴)
```

或者

```
CTABTMAX(n, F轴)
```

参数

CTABTSV	由一个跟随轴读取曲线图表的初值。
CTABTEV ()	由一个跟随轴读取曲线图表的终值。
CTABTSP ()	由一个引导轴读取曲线图表的初值。
CTABTEP ()	由一个引导轴读取曲线图表的终值。
CTABMIN ()	在整个范围或者某个定义好的区间内确定一个曲线图表的最小值。
CTABMAX ()	在整个范围或者某个定义好的区间内确定一个曲线图表的最大值。
F轴	跟随轴 跟随轴是通过曲线图表编程的轴。
L轴	引导轴 通过引导轴编程引导值。
n, m	曲线图表的编号 可以任意给曲线图表的编号赋值。这些编号仅用作进行唯一识别。
grad	图表函数的梯度在其中被恢复的参数：

数值和数值范围

在一个曲线图表的起始和终端点，跟随轴和引导轴的数值CTABTSV,CTABTEV,CTABTSP,CTABTEP.

R10=CTABTSV(n, 度, F轴).	曲线图表起始的跟随值
R10=CTABTEV(n, grad, F轴)	曲线图表起始的跟随值
R10=CTABTSP(n, grad, L轴)	曲线图表起始的引导值
R10=CTABTEP(n, grad, L轴)	曲线图表末端的引导值

跟随值的曲线图表的值范围CTABTMIN, CTABTMAX

R10=CTABTMIN(n, F轴)	整个周期中曲线图表最小的跟随值
R10=CTABTMAX(n, F轴)	整个周期中曲线图表最大的跟随值
R10=CTABTMIN(n, a, b, F轴, L轴)	曲线图表上、在a...b引导值范围内的最小的跟随值
R10=CTABTMAX(n, a, b, F轴, L轴)	在曲线图上、在a...b引导值范围内的最大的跟随值

注意

取消在图表定义范围内的R参数的赋值。

给R参数赋值举例

```
...
R10=5 R11=20
...
CTABDEF
G1 X=10 Y=20 F1000
R10=R11+5              ;R10=25
X=R10
CTABEND
...                    ;R10=5
```

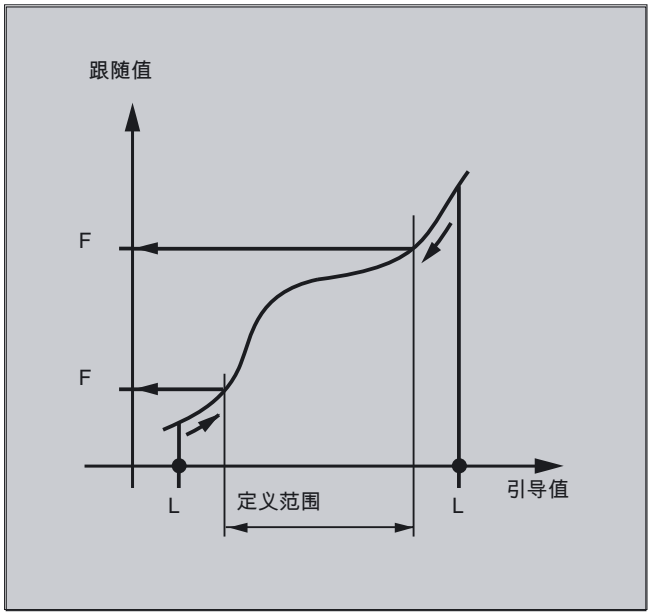
使用CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABTMIN, CTABMAX 举例

曲线图表的最小和最大值的确定。

```
N10 DEF REAL STARTVAL
N20 DEF REAL ENDVAL
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)                          ;图表定义开始
N110 X0 Y10                                    ;第1个图表曲线段的初值
N120 X30 Y40                                   ;第1个图表曲线段的终值 =
N130 X60 Y5                                    ;第2个图表曲线段的初值 ...
N140 X70 Y30
N150 X80 Y20
N160 CTABEND                                   ;图表定义结束
...
N200 STARTPOS = CTABTSV(1, GRADIENT)           ;开始位置 STARTPOS = 10,
N210 ENDPOS = CTABTEV(1, GRADIENT)             ;结束位置 ENDPOS = 图表的20，以及
N220 SRARTPARA = CTABTSP(1, GRADIENT)          ;STARTPARA = 10,
N230 ENDPARA = CTABTEP(1, GRADIENT)            ;ENDPARA = 读取跟随轴数值范围的80
...
N240 MINVAL = CTABTMIN(1)                      ;当Y = 5时的最小值和
N250 MAXVAL = CTABTMAX(1)                      ;当 Y = 40 时的最大值
```

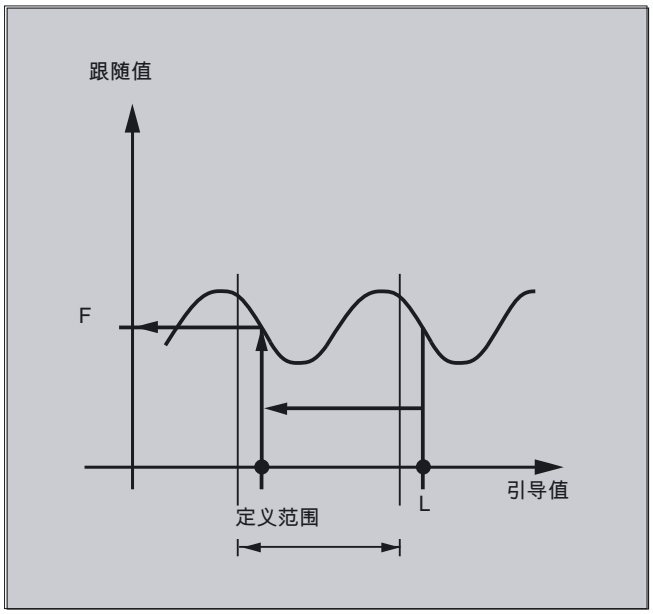
非周期的曲线图表

如引导值在定义范围外，上限或下限将作为跟随值给出。



周期性的曲线图表

如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。



注意

CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABTMIN, CTABTMAX

这些语言指令可以

- 由零件程序或者
- 从同步动作起直接使用。

功能

- CTABINV()P 的内部执行时间**取决于**
- CTABTSV, CTABTEV, CTABTSP, CTABTEP,(CTABTMIN, CTABTMAX 仅在没有规定引导值的区间时)
  是独立于

图表曲线段的数量。

在同步动作中读取

在同步动作中使用指令 CTABINV() 或者 CTABTMIN() 和CTABTMAX() 时，用户要注意：执行时

- 有足够的NC功率可以使用，或者
- 在调用前询问曲线图表的段的数量，从而可以划分相关的图表。

与同步动作编程相关的其它关系在“运动同步动作”一章中有所描述。

9.3.5 访问曲线图表位置和图表曲线段 (CTAB, CTABINV, CTABSSV, CATBSEV)

功能

读取图表位置, CTAB, CTABINV

使用 CTAB 可以从零件程序或者同步动作中直接读取相对于引导值的跟随值。

使用CTABINV可以读取跟随值的引导值。该分配不必始终唯一。CTABINV 因此需要预期值的一个近似值。

编程

读取相对于引导值的跟随值

CTAB(引导值, n, grad, [跟随轴，引导轴])

读取相对于跟随值的引导值

CTABINV(跟随值, 近似引导值, n, grad, [跟随轴，引导轴])

读取一个图表曲线段的起点和终点值

CTABSSV(引导值, n, grad, [F轴]),
CTABSEV(引导值, n, grad,[F轴])

参数

CTAB	直接从一个引导值读取跟随值。
CTABIN	读取相对于跟随值的引导值。
CTABSSV	从跟随轴读取曲线段的初值。
CTABSEV	从跟随轴读取曲线段的终值。
F轴	跟随轴
	跟随轴是通过曲线图表编程的轴。
L轴	引导轴
	通过引导轴编程引导值。
n, m	曲线图表的编号
	可以任意给曲线图表的编号赋值。这些编号仅用作进行唯一识别。
grad	图表函数的梯度在其中被恢复的参数：
近似引导值	预期近似值的位置值，使用该值可确定一个唯一的引导值。

- **CTABSSV, CTABSEV**

使用 CTABSSV 可以读取曲线段的**初值** ，该曲线段属于指定的引导值。属于指定引导值的曲线段的 **终值** 可以使用 CTABSEV 读取。

- **从带有CTAB, CTABINV 的曲线图表中得出的跟随位置或者引导位置**

R10=CTAB(LW, n, grad, F轴, L轴)	跟随值对一个引导值
R10=CTABINV(FW, aproxLW, n, grad, F轴, L轴)	引导值对一个跟随值

- **使用 CTABSSV, CTABSEV 通过指定引导值来确定曲线图表的曲线段**

R10=CTABSSV(LW, n, 度, F轴, L轴)	属于LW段的跟随轴的起始值
R10=CTABSEV(LW, n, 度, F轴, L轴)	属于LW段的跟随轴的终值

CTABSSV 和 CTABSEV 的使用举例

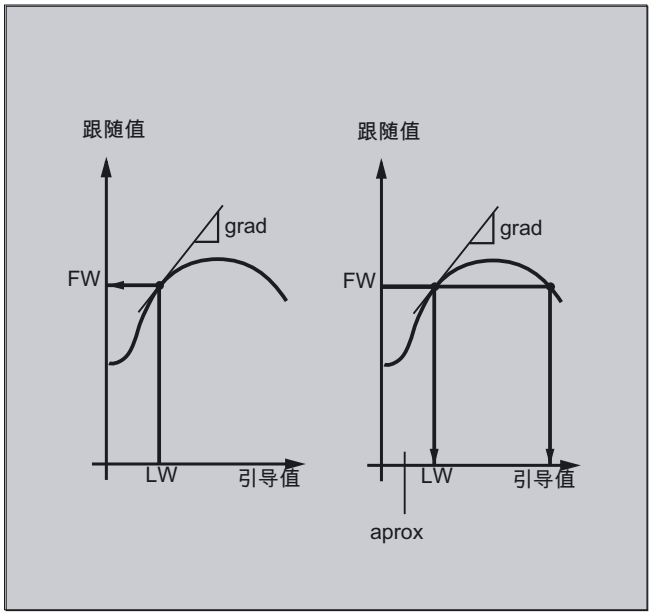
确定属于引导值X=30的曲线段。

```
N10 DEF REAL STARTPOS
N20 DEF REAL ENDPOS
N30 DEF REAL GRADIENT
...
N100 CTABDEF(Y,X,1,0)                ;图表定义开始
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 =
N130 X40 Y40                         ;第2个图表曲线段的开始位置 ...
N140 X60 Y10
```

```
N150 X80 Y0
N160 CTABEND                                    ;图表定义结束
...
N200 STARTPOS =
 CTABSSV(30.0, 1, GRADIENT)                     ;曲线段2中的开始位置 Y = 10
...                                             ;曲线段2中的结束位置 Y = 40
N210 ENDPOS =
 CTABSEV(30.0, 1, GRADIENT)                     ;属于引导值 X = 30.0 的是曲线段2。
```

读取图表位置, CTAB, CTABINV

CTABINV 需要一个预期引导值的近似值 (aproxLeitwert) 。CTABINV 返回与近似值最接近的引导值。例如，近似值可以是上一个插补节拍中的引导值。



此外，这两个功能还会将相应位置上图表函数的梯度发送给梯度参数 (gad) 。这样就可计算引导轴或者跟随轴在相应位置上的速度。

注意

CTAB, CTABINV, CTABSSV 和 CTABSEV

语言指令 CTAB, CTABINV 和

CTABSSV, CTABSEV 可以

- 由零件程序或者
- 由同步动作直接

使用。与同步动作编程相关的所有关系在“运动同步动作”一章中有所描述。

如果引导轴和跟随轴投影在不同的长度单位中，就必须在使用 CTAB/CTABINV/CTABSSV/CTABSEV 时对引导轴或者跟随轴进行选择说明。

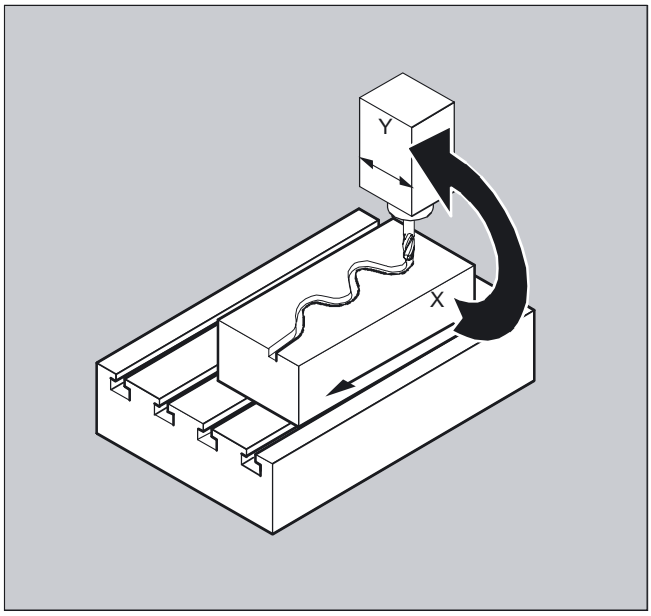
语言指令 CTABSSV 和 CTABSEV 在下列情况下**不适合** 对已编程的曲线段进行查询：

1. 圆弧或者渐开线编程。
2. 倒棱或者圆角已使用 CHF, RND 激活。
3. 使用 G643 进行修整的功能已激活。
4. 压缩器已使用例如 COMPON, COMPCURV, COMPCAD 激活。

9.4 轴向引导值偶合 (LEADON, LEADOF)

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。

实际值和额定值偶合

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

- 引导轴位置的额定值：额定值耦合

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。
关于引导值偶合可参阅 /FB/, M3, 联动和引导值偶合。

编程

LEADON(F轴,L轴,n)

或者

LEADOF(F轴,L轴,n)

引导值偶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。

参数

LEADON	接通引导值耦合
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	在额定值和实际值偶合之间转换

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！

轴向引导值偶合和各种运行状态，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偶合举例

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。偶合和去偶合过程作为**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：

- 偶合， LEADON(跟随轴，引导轴，曲线图表编号)
- 去偶合， 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。

注意

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

```
; 用来定义所有静态同步动作.
; **** 复位标记器
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 $AC_MARKER[7]=0
                                          ; **** E1 0=>1 启用传送偶合
N10 IDS=1 EVERY ($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) $AC_MARKER[0]=1
                                          ;**** E1 0=>1 启用辊进给偶合
N20 IDS=11 EVERY ($A_IN[1]==1) AND
($A_IN[5]==0) AND ($AC_MARKER[5]==0)
DO LEADON(U,LW,4) PRESETON(U,0)
$AC_MARKER[5]=1
                                          ; **** E1 0->1 启用校正头偶合
N21 IDS=12 EVERY ($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 启用润滑装置偶合
```

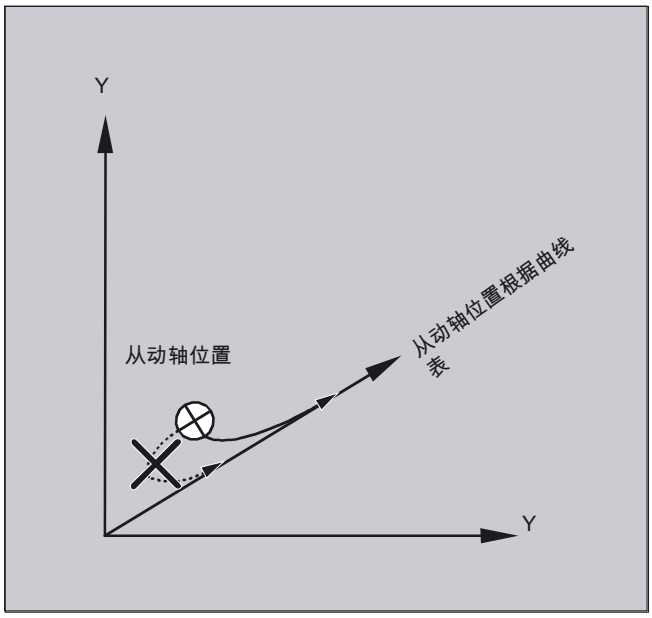
```
N22 IDS=13 EVERY ($A_IN[1]==1) AND
($A_IN[5]==0) AND ($AC_MARKER[7]==0)
DO LEADON(W,LW,4) PRESETON(W,0)
$AC_MARKER[7]=1
                                                ; **** E2 0=>1 断开偶合
N30 IDS=3 EVERY ($A_IN[2]==1)
DO LEADOF(X,LW) LEADOF(YL,LW)
LEADOF(ZL,LW) LEADOF(U,LW) LEADOF(V,LW)
LEADOF(W,LW) $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。

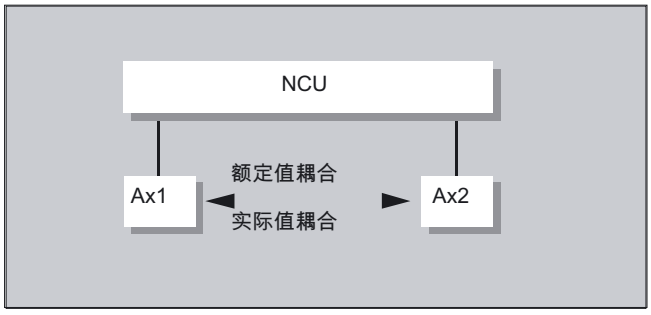


没有同步运动

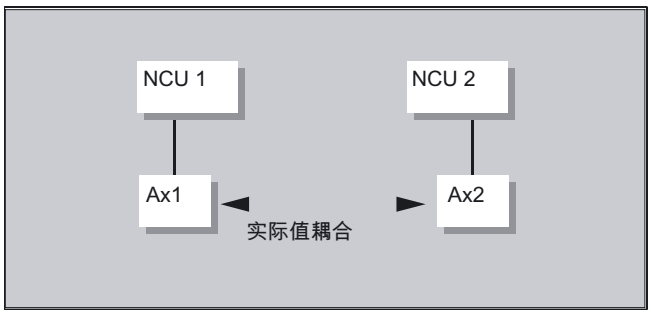
如计算所得的、在引导值耦合打开时跟随轴的额定位置与其实际跟随轴位置相距很远，则不产生同步运动。

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

9.5 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的规定对进给编程增加线性曲线和三次曲线。

空间的进程可以直接或者作为插补的样条被编程。由此，与须加工的工件的曲线有关，平滑的速度进程被连续地编程。

速度的过程使可没有冲击的速度变化成为可能，并通过这完成均匀的工件表面加工。

编程

F… FNORM

或者

F… FLIN

或者

F… FCUB

或者

F=FPO(…,…,…)

参数

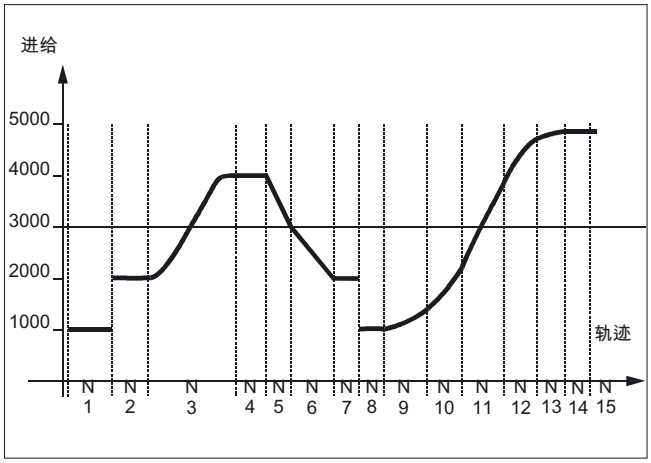
FNORM	基本设置。进给值通过程序段的轨迹行程来规定，并且作为模态值有效。
FLIN	轨迹速度曲线图**线性：** 进给值由实际值通过轨迹行程线性的、由程序开始到程序末尾运行，其后作为模态值有效。这种性能可与G93和G94联系在一起。
FCUB	轨迹速度曲线图**三次曲线：** 以程序段方式编程的F值 - 与程序段结束处有关 - 通过一个样条连接。样条以切线形式与前一个或者后一个进给说明开始和结束，与G93和G94一起作用。 如在一个程序段中缺少F地址，在这里将使用最后编程的F值。
F=FPO…	轨迹速度曲线图**通过多项式：** F地址通过从一个实际值到程序段末尾的的多项式说明进给过程。此后终值作为模态值有效。

在弯曲的轨迹中的进给优化

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额定进给曲线图。

各种进给曲线图举例

在这个例子中可找到不同进给剖面图的编程和图示。

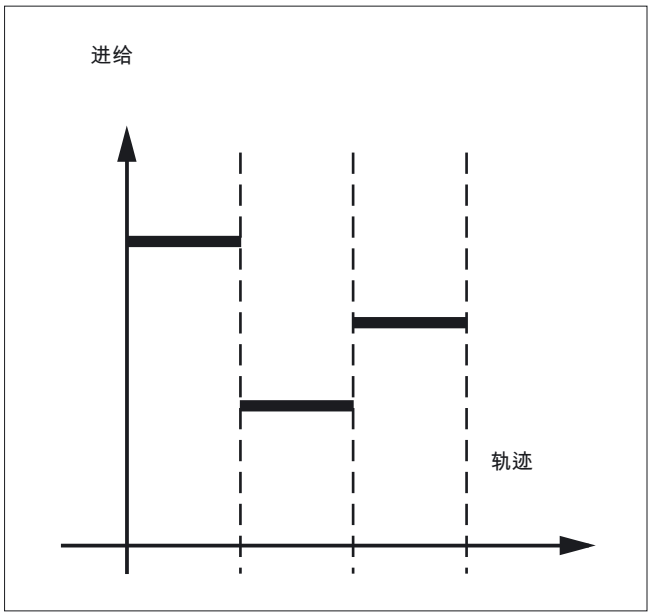



```
N1 F1000 FNORM G1 X8 G91 G64          ;恒定进给曲线图，增量说明
N2 F2000 X7                           ;额定速度突变
N3 F=FPO(4000, 6000, -4000)           ;通过带有进给4000的多项式在程序段末尾生成进给曲线图
                                      ;
N4 X6                                 ;多项式进给4000作为模态值
N5 F3000 FLIN X5                      ;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，带有加速度突变
                                      ;
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。

更多的信息可以在编程说明“基础部分”中找到。

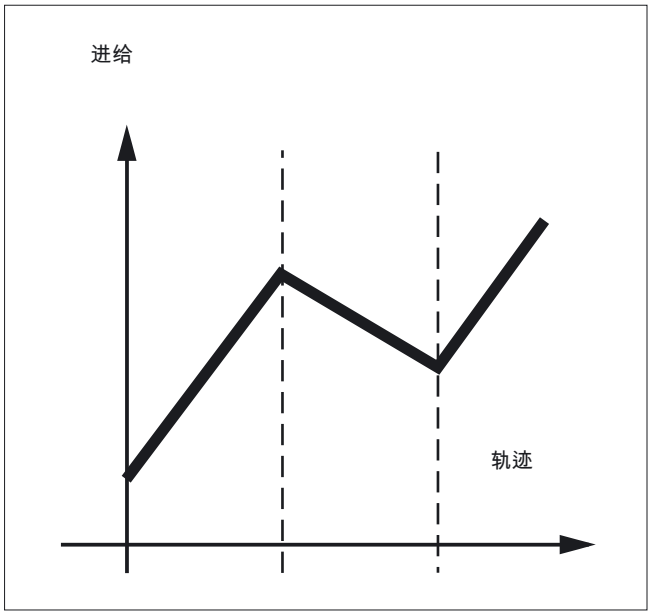


FLIN

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
```

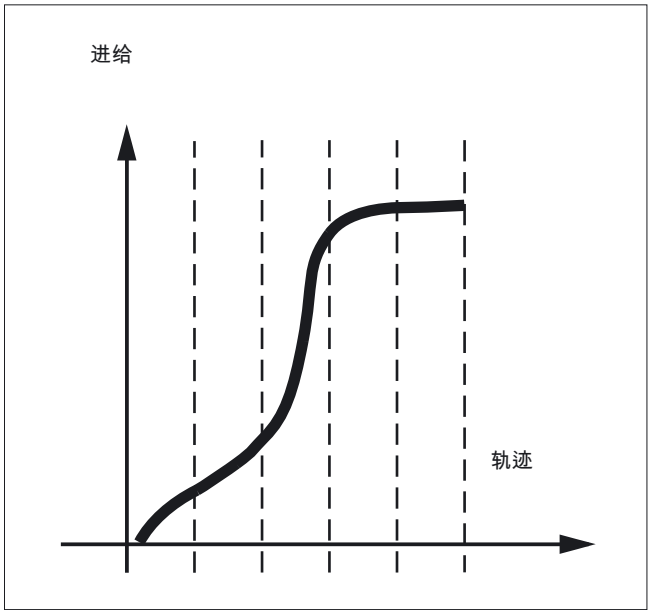


FCUB

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。

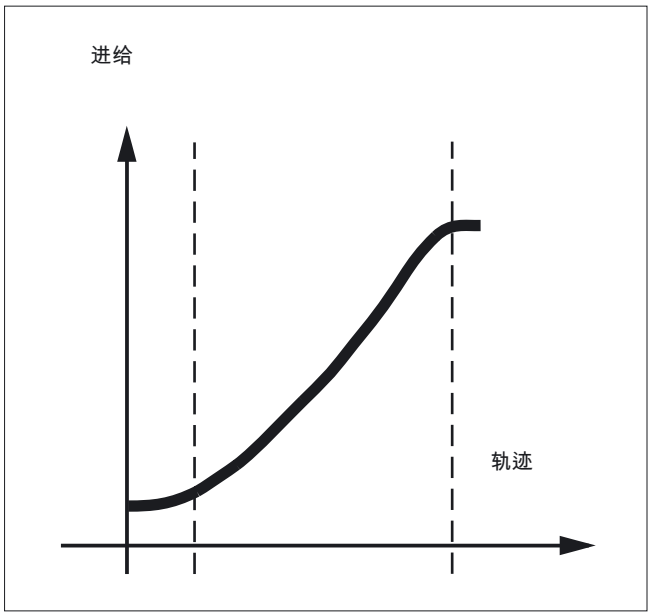
举例：

```
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;

对于样条曲线段而言，适用上一个所属程序段的F字。
以编程的F值在曲线段结束时有效且开始以线性运动。

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。

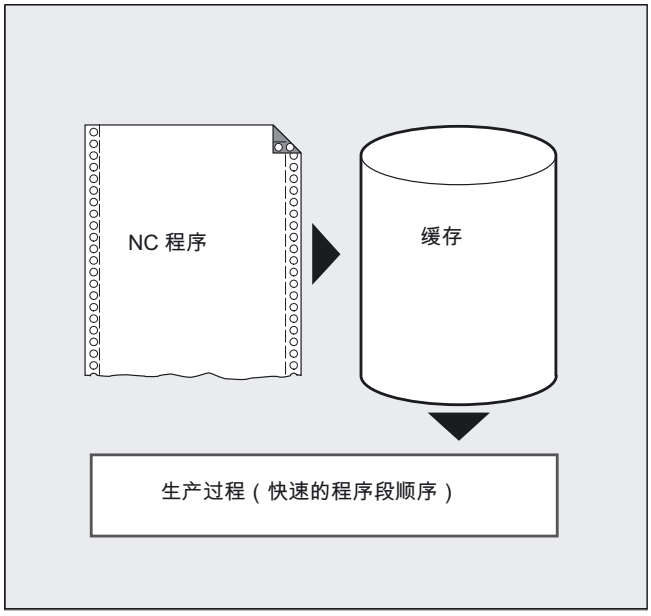
9.6 带有缓存的程序运行过程 (STARTFIFO, STOPFIFO, STOPRE)

功能

根据扩展级，控制系统具有一定数量的进刀存储器，它们在加工完成前已经存储预处理的程序段，并在加工过程中给出快速的程序段序列。

由此，可以以高速度运行较短的位移。

只要控制系统的剩余时间允许，原则上都载入缓存中。



编程

STARTFIFO

或者

STOPFIFO

或者

STOPRE

参数

STOPFIFO	停止快速加工程序段，载入缓存中，直到STARTFIFO，"缓存已满" 或者 "程序末尾" 被识别时为止。
STARTFIFO	快速加工段的开始，与进刀缓冲器的装载并行。
STOPRE	进刀停

注意

STOPFIFO 使加工动作停止，直到缓存已满或者 STARTFIFO 或 STOPRE 被识别时为止。

标识加工程序段举例

要临时保存在缓存中的加工程序段的开始和结束处用 STOPFIFO 以及 STARTFIFO 进行标识。

```
N10 STOPFIFO
N20
N100
N110 STARTFIFO
```

当缓存已满或者执行指令 STARTFIFO之后，才会开始执行这些程序段。

例外：

当加工程序段包含有强制执行一个未经缓存的操作指令时，就不会执行或者中断缓存加载（找零运行，测量功能，...）

停止前进运动举例 STOPRE

当编程**STOPRE** 时，只有在所有预先处理和保存后的程序段完全执行完毕的情况下，才会执行下一个程序段。 上一个程序段被停在精确停止位置中（如同 G9）。

举例：

```
N10 …
N30 MEAW=1 G1 F1000 X100 Y100 Z50
N40 STOPRE
```

当访问机床的状态数据时 ($SA...)，控制系统生成内部的前进运动停止。

举例：

```
R10 = $AA_IM[X]          ; 读取X轴的实际值
```

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。

9.7 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。

编程

N... DELAYFSTON N... DELAYFSTOF	命令单独在一个零件程序行中。 **DELAYF**eed **ST**op **ON**/**OF**

两个指令都只允许在零件程序中，而不可以在同步动作中。

参数

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

注意

当机床数据为 MD 11550：STOP_MODE_MASK Bit 0 = 0 (默认t)，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。见下说明

停止事件举例

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

事件名称	反应	中断参数
RESET	立即	NST: DB21,… DBX7.7 和 DB11, … DBX20.7
PROG_END	警报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	警报16954	NC程序：M0 和 M1
WAITM	警报16954	NC程序：WAITM
WAITE	警报16954	NC程序：WAITE
STOP_ALARM	直接	报警：报警设计 STOPBYALARM
RETREAT_MOVE_THREAD	警报16954	NC程序： 报警16954，当LFON (不可以在G33中停止和快速升高)
WAITMC	警报16954	NC程序：WAITMC
NEWCONF_PREP_STOP	警报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	警报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。
警报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。
警报16955	在停止延迟区中进行一个不被允许的动作，程序继续。

对不同事件的其它反应可参阅功能说明中的表格 /FB1/, BAG, 通道, 程序运行，复位特性 (K1).

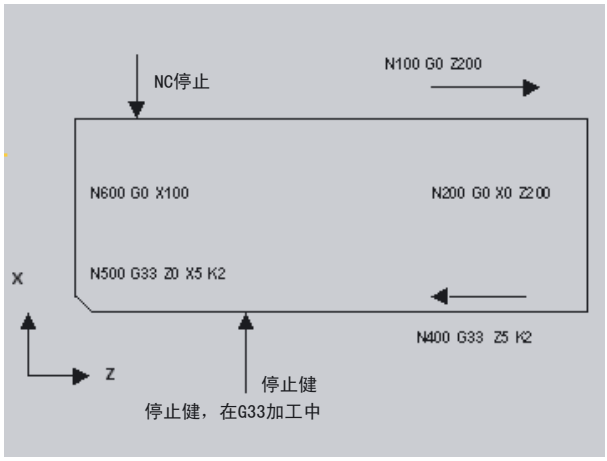
两个程序层中停止延时段的嵌套举例

```
N10010 DELAYFSTON()              ; 带N10xxx的程序级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2重复的开始，无效
...
```

```
N20020 DELAYFSTOF()                ；另一个程序级的结束，无效
N20030 RET

N10050 DELAYFSTOF()                ;停止延时段 相同层中的末尾
...
N10060 R2 = R2 + 2
N10070 G4 F1                       ;结束停止延迟区 停止
                                   ;从现在起立即有效
```

程序节选举例

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 "Stopp"，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。

```
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 SERUPRO 类型的程序段查找的详细资料可参阅

/FB1/, K1, 通道，程序运行，复位特性。
/FB1/, V1,配合使用 G331/G332 进给，没有平衡卡盘时的螺纹切削进给。

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。

注意

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！

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 。

9.8 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。

编程

N... IPTRLOCK 或者 N... 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。

参数

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。

举例

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。

```
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 ()                              ; 重复的开始，无效
...
N20020 IPTRUNLOCK ()                            ; 另一个程序级的结束，无效
N20030 RET
...
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。

与此有关的其它信息可查阅功能说明
/FB1/, K1, 通道，程序运行，复位特性。

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
- 轴向引导值偶合，当 LEADON

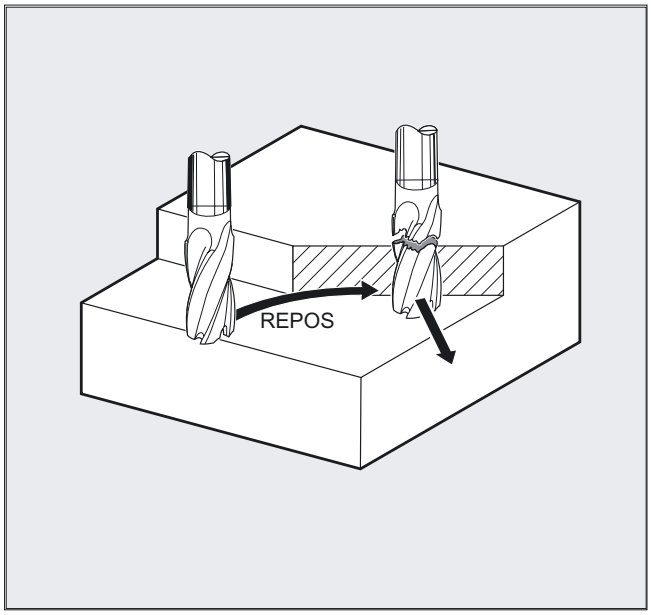
激活自动中断指针。
如果已编程的中断向量和可通过机床数据激活的自动中断向量相互交迭，就会形成最有可能的不可查找程序段。

9.9 重新向轮廓运动 (REPOSA/L, REPOSQ/H, RMI, RMN, RMB, RME)

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。



对于程序过程的中断，参见编程说明“工作准备”中的“中断程序”段落。

编程

REPOSA RMI DISPR=… **或者** REPOSA RMB **或者** REPOSA RME **或者** REPOSA RMN

或者

REPOSL RMI DISPR=… **或者** REPOSL RMB **或者** REPOSL RME **或者** REPOSL RMN

或者

REPOSQ RMI DISPR=…DISR=… **或者** REPOSQ RMBDISR=… **或者** REPOSQ RME DISR=… **或者** REPOSQA DISR=…

或者

REPOSH RMI DISPR=… DISR=…**或者** REPOSH RMB DISR=… **或者** REPOSH RME DISR=… **或者**

REPOSHA DISR=…

参数

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

起动举例，在一条直线上起动，REPOSA, REPOSL

刀具以一条直线直接回到再次返回点。

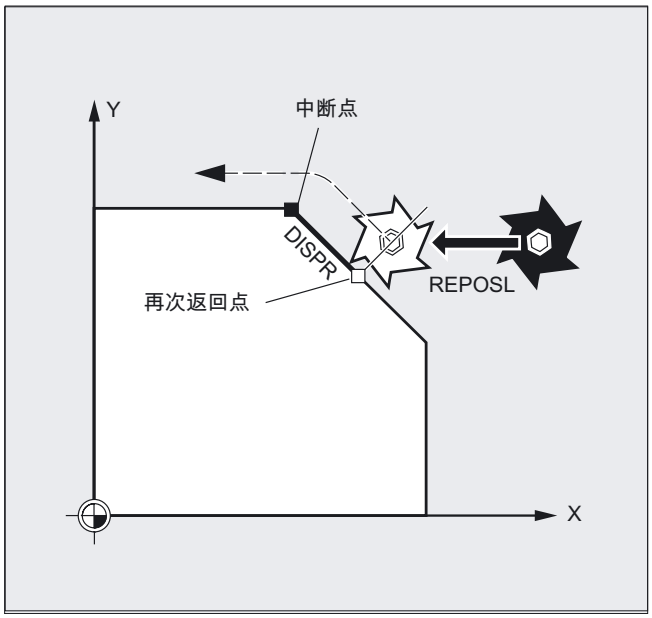
使用REPOSA，所有轴自动处理。使用REPOSL时，您可以指定待运行的轴。

举例：

```
REPOSL RMI DISPR=6 F400
```

或者

```
REPOSA RMI DISPR=6 F400
```

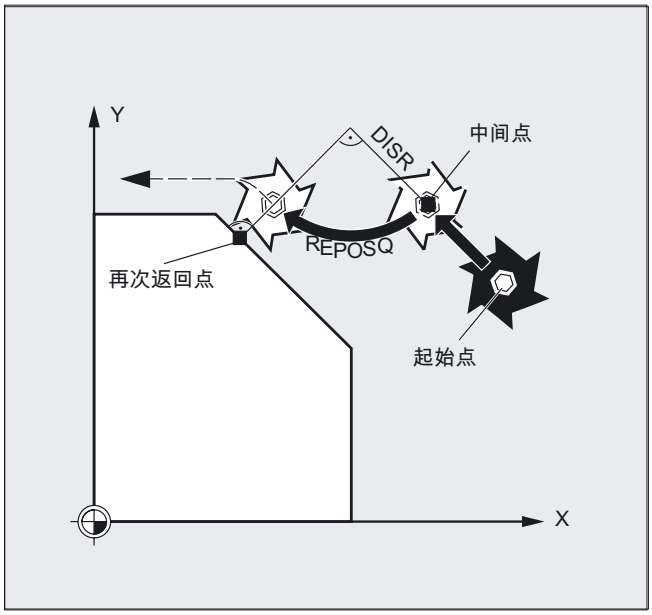


起动举例，在四分之一圆中起动，REPOSQ, REPOSQA

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。

举例：

```
REPOSQ RMI DISR=10 F400
```

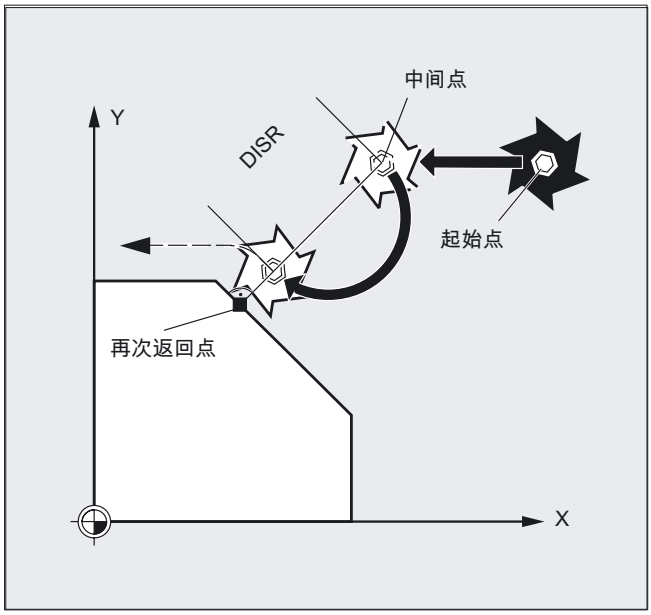


在半圆中起动刀具举例，REPOSH, REPOSHA

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。

举例：

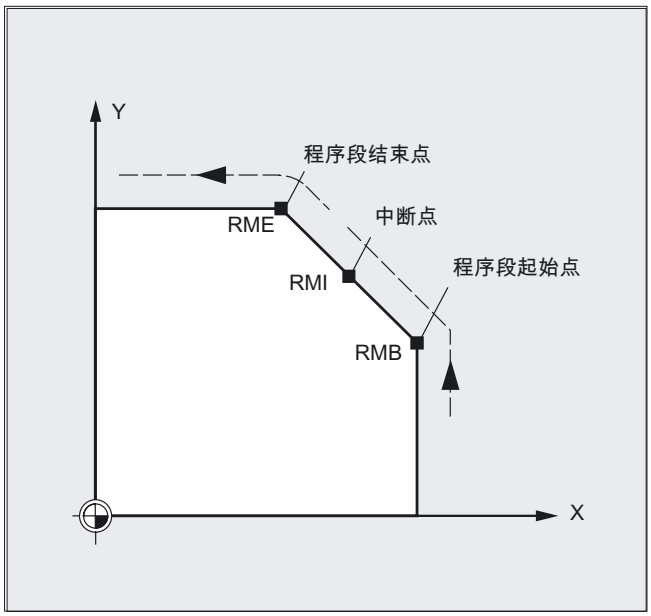
```
REPOSH RMI DISR=20 F400
```



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。

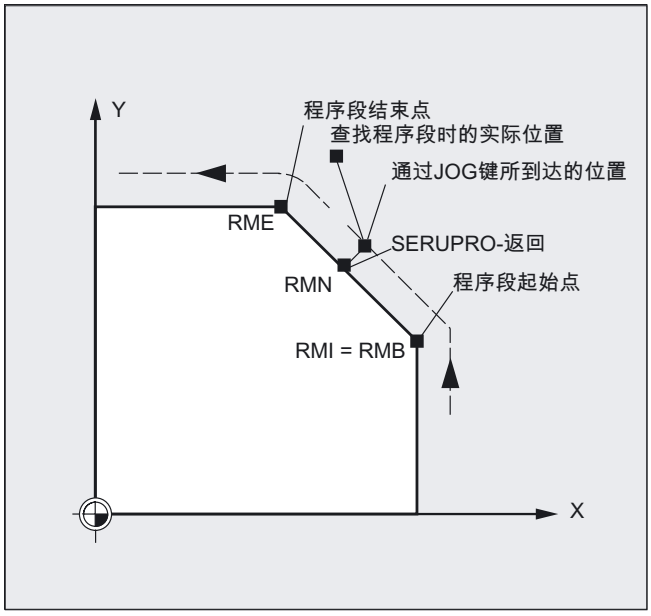
应用示例：

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。



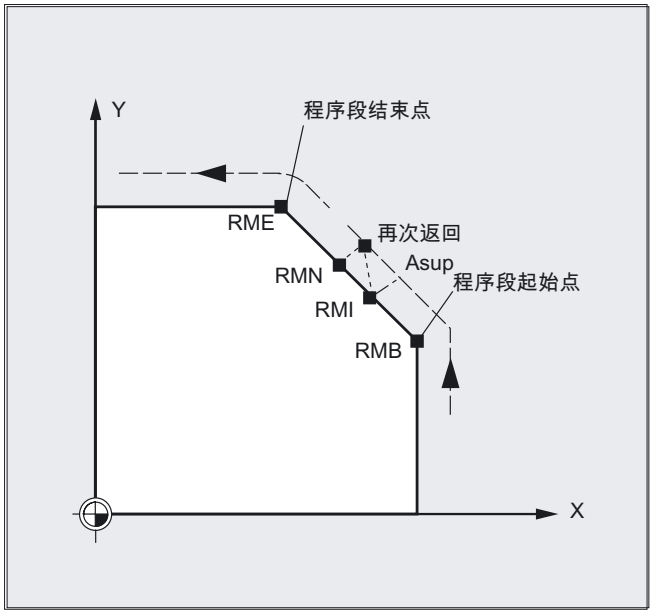
注意

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

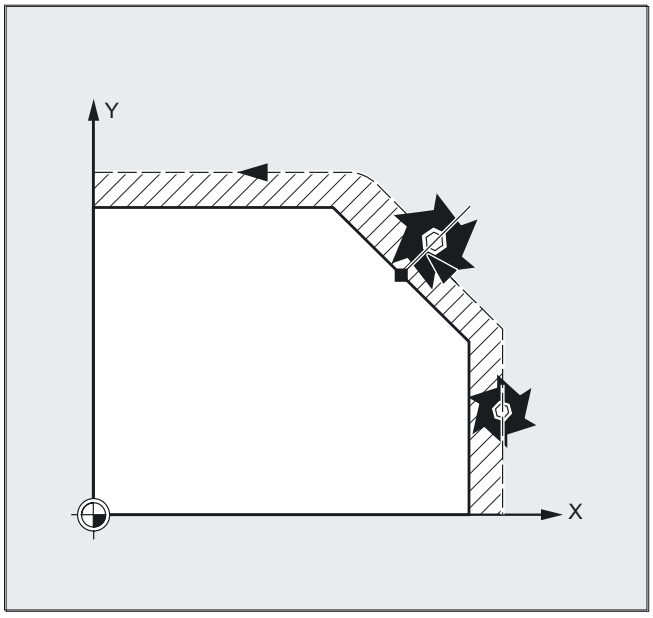
4 RMN: 向已中断程序段的下一个轨迹点运动。

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：

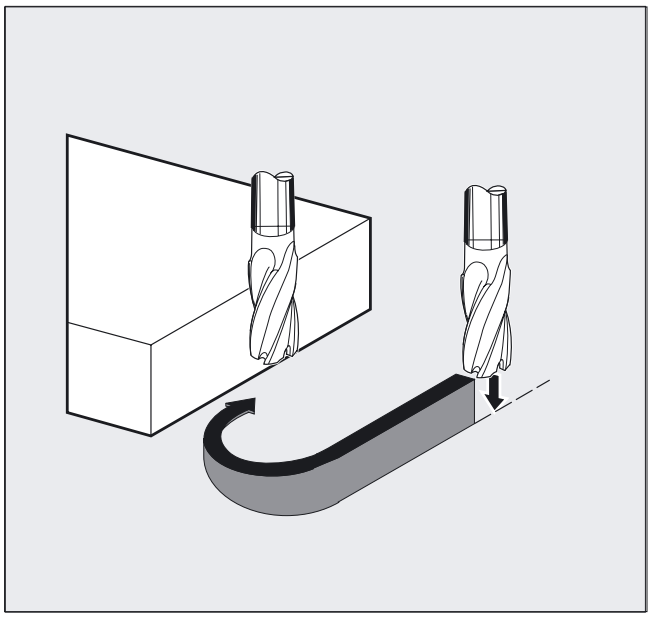
在指定的工作平面 G17 ~ G19 中作圆运动。

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。

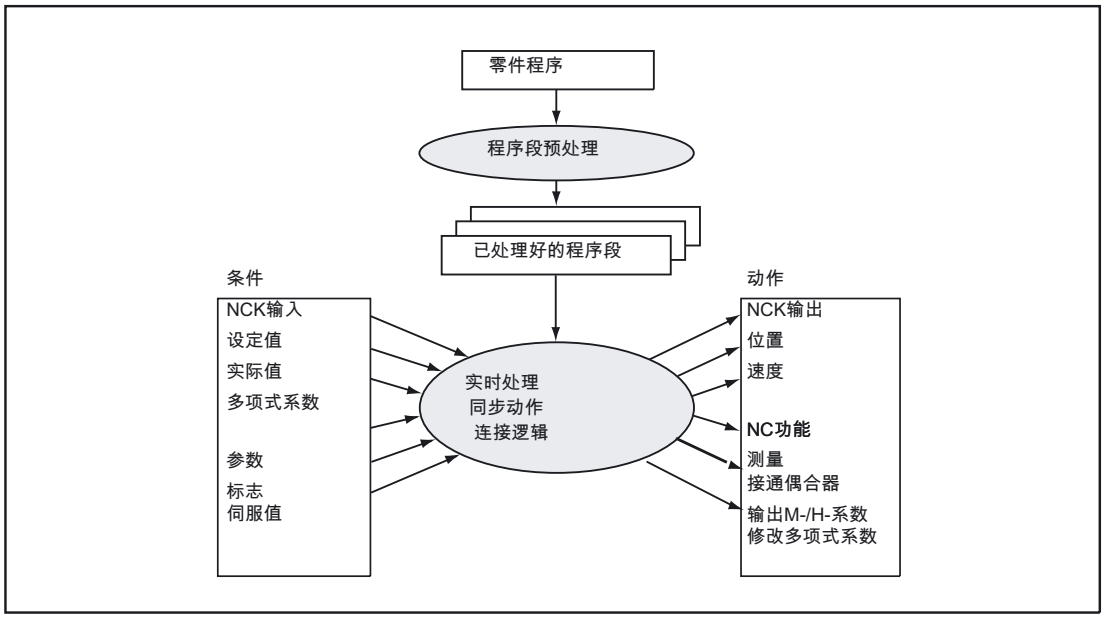


运动同步动作 10

10.1 结构，一般基础

功能

同步动作提供如下可能性：从当前的零件程序出发推动几个不同的动作，并使它们同步执行。



同步动作如何使用由条件定义，其求值运算以实时（插补节拍）方式进行。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，同步动作还包含动作有效级的说明和对编程实时变量的询问频率，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

DO 动作1 动作2 匝

关键字条件 DO 动作1 动作2 匝

ID=n 关键字条件 DO 动作1 动作2 匝

IDS=n 关键字条件 DO 动作1 动作2 匝

指令单元

识别代码 ID/IDS:

ID=n	自动运行方式中的**模态** 有效同步动作，**局部程序**；n = 1... 255
IDS=n	各种运行方式中的**模态**有效同步动作， **静态**；n = 1... 255
没有 ID/IDS	自动运行方式中的**逐段** 有效同步动作

关键字：

没有关键字	动作执行不受条件制约。
WHEN, WHENEVER, FROM, EVERY,	需要开始的动作的询问次数

条件

实时变量的逻辑联系。
已使用的实时变量在解释节拍中 (IPO-节拍) 进行分析。在同步动作时的优点：这里没有出现进刀停止。

如果在某个零件程序中出现实时变量（例如实际值，某个数字输入或者输出端的位置等等），就会停止向前运动，直到上一个程序段执行完毕并且实时变量的值存在时为止。

DO:

触发动作

同步动作/工艺周期的协调：

CANCEL[n]	删除同步动作
LOCK[n]	禁止工艺循环
UNLOCK[n]	释放工艺循环
RESET	复位工艺循环

举例

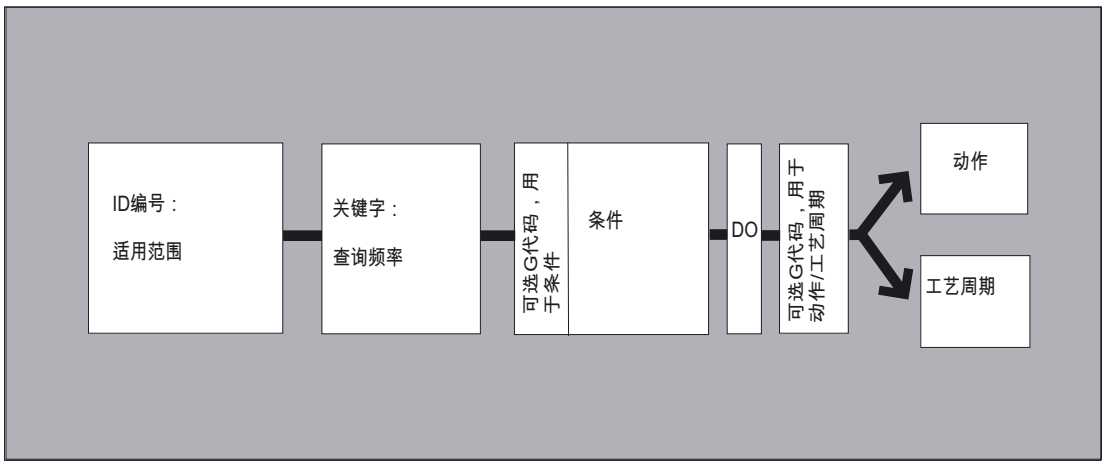
WHEN $AA_IW[Q1]>5 DO M172 H510	；如果轴Q1的实际值超过5毫米，则辅助功能M172和H510输出到PLC接口。

10.1.1 编程与指令单元

功能

一个同步动作在程序段中是单独的，并且在某个机床功能的下一个可执行程序段中有效（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：



编程

```
ID=n 关键字 条件 DO 动作 1 动作 2 ...
```

指令单元

识别号 ID/IDS	自动运行方式或者任何运行方式中**模态** 有效同步动作的有效范围
关键字	没有询问次数，WHEN, WHENEVER, FROM, EVERY
条件	实时变量的连接逻辑，在IPO节拍中检查该条件。
DO	当满足条件时，触发动作或者工艺周期。
动作	当满足条件时给已开始的动作或者变量赋值。
工艺周期	当满足条件时将某个程序当作动作调用。

举例

ID=1	WHENEVER	$A_IN[1]==1	DO	$A_OUT[1]=1
同步动作号1：	如果	有输入端1	则	设置输出端1

10.1.2 有效性范围： 识别代码ID

功能

同步动作的有效范围已通过识别代码确定：

- **没有模态ID** 自动运行方式中逐段有效的同步动作
- **ID=n** 局部程序任何运行方式中模态有效的同步动作
- **IDS=n** 任何静态运行方式中模态有效的同步动作

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其 ID(S)-代码的顺序进行加工(在插补节拍中)。

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。

识别代码ID

- **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。

  举例：

```
WHEN $A_IN[3]==TRUE DO $A_OUTA[4]=10
G1 X20                                          ; 可执行的程序段
```

- **ID=n; n=1...255**

  同步动作在下列程序段中**模态** 有效且可通过 CANCEL(n) 或者通过编程一个带有相同ID的新同步动作来关闭。 在M30程序段中有效的同步动作会继续有效（有可能要用 CANCEL 指令来删除）。 ID-同步动作**仅** 在**自动运行方式**中有效。

  举例：

```
ID=2 EVERY $A_IN[1]==1 DO POS[X]=0
```

- **IDS=n; n=1...255**

  这种**静态** 同步动作在**所有运行方式** 中**模态**有效。它不仅可以在零件程序中定义，而且也可以在上电之后直接从一个由PLC启动的异步子程序（ASUP）中定义。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。

举例：

```
IDS=1 EVERY $A_IN[1]==1 DO POS[X]=100
```

10.1.3 关键字

功能

关键字用来确定查询条件的次数和执行相应动作的次数。

如果没有编程关键字，则该条件一直被看作满足。同步指令循环执行。可以查询的条件有 WHEN, WHENEVER, FROM 或者 EVERY。

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查询，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。
条件	实时变量的连接逻辑，在IPO节拍中检查该条件。当满足该条件时，就会执行相应的动作。 通过比较两个实时变量或者将某个实时变量与某个预先算出的表达式进行比较，确定是否要执行某个动作。 G代码可以在同步动作中编程，用于分析条件。

举例

没有关键字

```
DO $A_OUTA[1]=$AA_IN[X] ;发送实际值到模拟输出端
```

EVERY

```
ID=1 EVERY $AA_IM[B]>75 DO POS[U]=IC(10) FA[U]=900
```

;当MKS中轴B的实际值超过值75时，U轴应以最大轴向进给量10继续定位。

WHENEVER

```
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] > $AA_IM[X1] DO …             ;和其它实时变量进行比较
WHENEVER ($A_IN[1]==1) OR ($A_IN[3]==0) DO ...   ;两个相互关联的比较
```

条件

比较两个实时变量，或者比较一个实时变量与前面计算的表达式，确定是否应该执行一个动作。

在条件中也可以通过布尔运算子将比较结果联系起来 ()。

在插补节拍中检查该条件。如果该条件满足，则执行相关的动作。

条件可以用一个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

- 比较实时变量（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

10.1.4 作用

功能

在每个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启动。

可以使用一个G代码将动作用于动作，或者用于被子程序当作工艺循环的加工。

其它动作（包括迄今为止的所有补充）可参阅“同步动作一览表”一章。

指令单元

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程序段中，那么即使在结束工艺循环之后，G代码对于所有随后的动作而言将继续模态有效，一直到下一个G代码时为止。每个动作程序段只可编程G代码组的**一个G代码** (G70，G71，G700，G710) 。

注意

动作可以与运行方式无关执行。

在有效程序中，仅在自动方式下以下动作生效

- STOPREOF,
- DELDTG.

带有两个动作的同步动作举例

```
WHEN $AA_IM[Y] >= 35.7 DO M135                ;如果条件已满足，就会将
$AC_PARAM=50                                  ;M135 发送给PLC并且将修调设定为50%。
```

10.1.5 可能有的同步动作一览表

编程同步动作

一个程序（单轴程序，工艺循环）也可以作为动作说明。该程序由可以单独在同步动作中编程的动作构成。这样一个程序中的各个动作按插补节拍中顺序执行。

同步动作已连续在多个软件版本中更新过，更新内容有表达式、可使用的主要过程变量和同步动作中的复杂条件。

应用方法

可能有：

- 将实时变量与IPO节拍中的基本计算类型和功能关联起来或者对主要过程变量的间接选址与联机索引进行修改
- 通过可设计的机床数据来设置同时激活的同步动作的数量
- 联机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
- 设置其它安全功能
- 设置叠加运动、联机刀具补偿和距离调节
- 让JOG运行方式中的同步功能超出程序段范围之外发挥作用
- 从PLC对同步动作进行干预
- 调用单轴程序或者工艺循环
- 在同步动作、工艺循环、零件程序和PLC之间进行独立于二进制信号或者模拟信号的协调
- 输出数字和模拟量信号
- 利用插补节拍采集同步动作的时间比例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

一览

同步动作的应用	说明
DO $$V…= DO $$A...=	给变量赋值（主要过程变量） 给变量赋值（伺服值）
DO $AC…[n]= DO $AC_MARKER[n]= DO $AC_TIIMER[n]= DO $AC_PARAM[n]=	特殊的实时变量 读取或者写入特定通道的标记/计数器，设定定时器 设定同步动作参数
DO $Rxx= IF Rxx>20	将计算变量作为实时变量读取 分析正常的计算变量
DO $$S...=	写入设定参数（主要过程变量）
DO $AC_FIFO1[n] …FIFO10[n]=	设置系统参数
DO $AC_BLOCKTYPE= DO $AC_BLOCKTYPEINFO= DO $AC_SPLITBLOCK=	解释当前的程序段（主要过程变量）
DO 辅助功能 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)	分析功能的定义（多项式） 激活分析功能：AC调节
DO FTOC	在线刀具补偿
G-代码 DO G70/G71/G700/G710	确定定位任务的测量单位制 英制或者公制尺寸
将轴定位 DO POS[轴]= / DO MOV[轴]= DO SPOS[主轴]=	确定定位轴运动 起动/定位/停止指令轴 起动/定位/停止主轴
起动/停止轴 DO MOV[轴]=值	起动同步动作中的定位轴运动和主轴（指令轴）
进给补偿 DO POS[轴]= FA [轴]=	确定轴向进给FA 设定不变的进给值
软件终端开关 DO $A_WORAREA_PLUS_ENABLE]=	考虑使用 G25/G26 编程的 工作范围极限
轴协调，进给补偿 ID=1 ...DO POS[轴]= FA [轴]= ID=2 ...DO POS[轴]= $AA_IM[轴] FA [轴]=	修改该轴的运动指令 从同步动作起动定位 当同时进行两个同步动作时，跟踪轴的终点位置和进给
软件终端开关 DO $A_WORAREA_PLUS_ENABLE]=	考虑使用 G25/G26 编程的 工作范围极限
DO PRESETON(轴，值)	设定实际值（从同步动作预先设定）
主轴运动 ID=1 EVERY $A_IN[1]=1 DO M3 S…. ID=2 EVERY $A_IN[2]=1 DO SPOS=	起动/定位/停止主轴 设置旋转方向和转速 定位主轴
轴偶合 DO TRAILON(FA, LA, 偶合系数) DO LEADON(FA, LA, NRCTAB, OVW)	开启/关闭轴偶合，设定参数 开启联动 开启引导值偶合 (OVW:修改后的表格)

DO MEAWA(轴)= DO MEAC(轴)=	开启轴向测量 开启连续测量
DO SETM(标记编号) DO CLEARM(标记编号)	设定等候标记 删除等候标记
DO SETAL(报警编号)	设定循环报警（附加安全功能）
向固定止档运动 DO FXS[轴]= DO FXST[轴]= DO FXSW[轴]= DO FOCON[轴]= DO FOCOF[轴]=	释放固定止挡和紧固力矩： 选择向固定止挡运动 修改紧固力矩 修改监控范围 激活（模态）以限定的力矩/力运动，解除FOC（同步动作与程序段有关）
确定同步动作中的 s ID=2 EVEREY $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
确定当前的修调 DO $AA_OVR= DO $AC_OVR= DO $AA_PLC_OVR DO $AC_PLC_OVR DO $AA_TOTAL_OVR DO $AC_TOTAL_OVR	读取或者写入当前的修调 轴向修调 轨迹修调 PLC所规定的轴向修调 PLC所规定的轨迹修调 总的轴向修调 总轨迹修调
利用率分析 $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=	包括所有通道同步动作在内的时间需求 当前的 IPO-计算时间 最长的 IPO-计算时间 最短的 IPO-计算时间 与IPO节拍成比例的当前 IPO-计算时间 通过所有通道的同步动作的当前计算时间 通过所有通道的同步动作的最长计算时间 整个同步动作的百分比率
调用工艺循环 POS[X]= FA[X]= POS[Y]= FA[Y]= POS[Z]= FA[Z]= 锁止，释放，中断 DO LOCK(n, n, ...) DO UNLOCK(n, n, ...) DO RESET(n, n, ...)	作为同步动作起动轴程序，例如： 满足条件时起动轴程序 X，满足条件时起动轴程序 Y，满足条件时起动轴程序 Z 工艺周期 锁止 释放 中断
删除同步动作 CANCEL(n, n, ...)	带有标识ID(S)的模态同步动作 仅直接在零件程序中删除

10.2 条件和动作的基本模块

实时变量

在插补节拍中分析和写入实时变量。

实时变量为

- $A..., 主要过程变量

- $V..., 伺服变量。

用作特殊标识时，可以在同步动作中使用**$$** 对这些变量进行编程：

$AA_IM[X] 等同于 $$AA_IM[X].

当在IPO节拍中进行分析/赋值时，设定数据和机床数据必须标识有 $$ 。

文献：系统变量列表。

系统变量列表有同步动作的标识SA。

实时计算

实时计算限制为数据类型INT, REAL 和 BOOL。

实时表达式是插补节拍中可以执行的计算，可以在赋值给NC地址和变量的条件与动作中使用这些计算。

比较 (==, <>, <, >, <=, >=)	在条件中可以比较变量或者同一数据类型的部分表达式。结果始终为数据类型BOOL。所有已知的比较运算符均允许
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.运算对象为变量或者INTERGER型常量。
基本计算类型 (+, -, *, /, DIV, MOD)	INTEGER 型和 REAL 型实时变量可以通过基本计算类型相互关联或者与常量关联起来。
数学函数 (SIN, COS, TAN, ASIN, ACOS, ABS, TRUNC, ROUND, LN, EXP, ATAN2, POT, SQRT, CTAB, CTABINV).	数据类型为 REAL 的实时变量上可以使用数学函数。
索引	可以使用实时变量对实时变量进行索引。

注意

只能将同一类型的变量相互关联起来。
正确写法为：$R10=$AC_PARAM[1]

错误写法为：$R10=$AC_MARKER[1]

不是实时形成的变量不得使用实时变量进行索引。

举例

- **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

```
DO $AC_PARAM[3] = $A_INA[1]-$AA_IM[Z1]                    ;减法，两个
                                                          ;实时变量
WHENEVER $AA_IM[x2] < $AA_IM[x1]-1.9 DO $A_OUT[5] = 1
                                        ;从实时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**

```
DO $AC_PARAM[3] = COS($AC_PARAM[1])
```

- **实时表达式**

```
ID=1 WHENEVER ($AA_IM[Y]>30) AND ($AA_IM[Y]<40)      ;选择一个位置范围
DO $AA_OVR[S1]=80
ID=67 DO $A_OUT[1]=$A_IN[2] XOR $AN_MARKER[1]        ;分析两个布尔信号
ID=89 DO $A_OUT[4]=$A_IN[1] OR ($AA_IM[Y]>10)        ;发送某个比较结果
```

- **索引实时变量**

```
WHEN…DO $AC_PARAM[$AC_MARKER[1]] = 3
不允许的是
$AC_PARAM[1] = $P_EP[$AC_MARKER]
```

10.3 同步动作的特殊实时变量

10.3.1 标记/计数器 $AC_MARKER[n]

功能

可以在同步动作中读取和写入标记变量。

特定通道的标记/计数器 $AC_MARKER[n]
数据类型：INTEGER
每个通道已在相同名称下有一个特定通道的标记变量。

举例

```
WHEN ...DO $AC_MARKER[0] = 2
WHEN ...DO $AC_MARKER[0] = 3
WHEN $AC_MARKER == 3 DO $AC_OVR=50
```

10.3.2 计时变量 $AC_定时器[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。
数据类型：REAL
单位：s
n:定时器变量序号

定时器变量

设定定时器

通过赋值来时定时器变量开始相加
$AC_TIMER[n] = 值
n:时间变量的编号
值:初值 (i.d.R 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。
举例

举例

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

```
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
```

10.3.3 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。
数据类型：REAL
n:参数的编号 0-n

每个通道可供使用的AC参数变量的数量通过机床数据MD 28254:MM_NUM_AC_PARAM来确定。

每个通道在相同的名称下参数只可以出现一次。标志位$AC_PARAM保持在动态存储器中。

10.3.4 访问R参数 $Rxx

功能

这些静态参数通常用来在零件程序中进行计算。在 IPO 节拍中可以通过添加**$** 的方式来对其进行访问。
数据类型：REAL

应用

同步动作中R参数的应用，可以提供：

- 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
- 存档同步动作时所求得的值。

参数

R参数要么被用作

Rxx	"正常"计算变量

或者

$Rxx	实时变量

注意

如果R参数在某个同步动作中使用结束之后被重新作为"正常"计算变量使用，则必须用STOPRE给进给过程和主要过程的同步动作编程进给停止。

举例

```
WHEN $AA_IM[X]>=40.5 DO $R10=$AA_MM[Y]      ;在同步动作中私用R10
G01 X500 Y70 F1000
STOPRE                                      ;进刀停止
IF R10>20                                   ;分析计算变量
```

读取R参数

```
WHEN $AA_IM[X]>=40.5 DO $R10=$AA_MM[Y]      ;读取R参数10。
WHEN $AA_IM[X]>=6.7 DO
$R[$AC_MARKER[1]]=30.6                      ;读取编号在标记1中的R参数
```

10.3.5 读取/写入机床数据和设定数据

功能

也可从同步动作中读取和写入机床数据和设定数据（MD，SD）。

读取不可修改的MD，SD举例

就像在标准零件程序指令中一样，这些数据从同步动作中进行寻址，并且使用一个$符号导入。

```
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
```

读取可修改的MD，SD举例

使用 $$ 符号从同步动作导入这些数据后对其寻址并且在 IPO 节拍中进行分析。

```
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MD，SD举例

前提条件：

当前已设置的访问权限必须允许写入访问。仅当修改立即有效时，从同步动作改变MD和SD才有意义。所有MD和SD的有效性均在文献中规定：/LIS/, 列表

寻址：

需要修改的 MD 和 SD 应使用 $$ 导入后进行寻址。

举例

```
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。
```

10.3.6 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。

参数

可使用的 FIFO-变量的数量通过机床数据
MD 28260:NUM_AC_FIFO 来确定。

可写入某个 FIFO-变量的数值的数量通过机床数据
MD 28264:LEN_AC_FIFO 来定义。所有FIFO变量有相同的长度。

所有 FIFO-单元之和只有当在 MD 28266:MODE_AC_FIFO 中已设定Bit0时才会形成。

索引**0 ~ 5** 具有特殊含义：

n=0： 当写入时：新数值被保存在 FIFO 中
当读取时：读最早的单元并从FIFO中去除

n=1: 访问最早保存的单元

n=2: 访问最新保存的单元

n=3: 所有 FIFO 单元之和

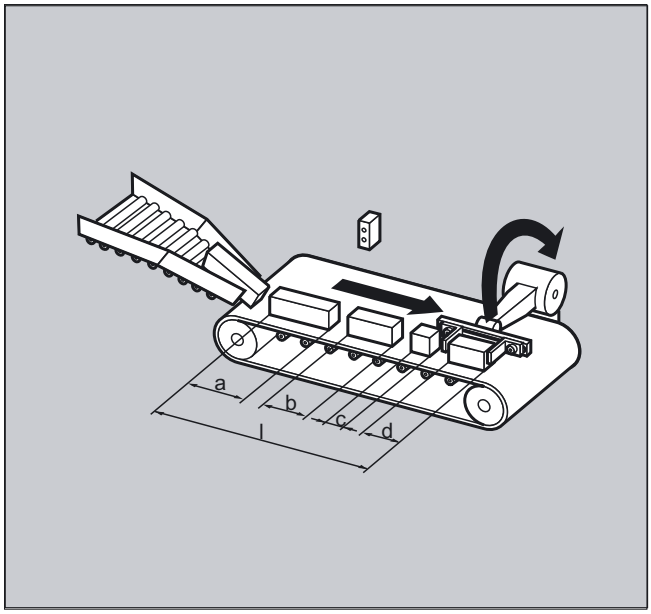
n=4: 在 FIFO 中可供使用的单元的数量。
可以对 FIFO 的每个单元进行读写访问。将单元数量复位就可使FIFO-变量复位，例如用于第一个 FIFO-变量：`$AC_FIFO1[4] = 0`

n=5: 相对于FIFO-开始的当前写索引

n=6 bis 6+nmax: 访问第 n 个 FIFO-单元

循环存储器举例

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，在输送长度为“l”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。




```
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，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。
```

10.3.7 关于插补器中记录类型的信息

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

参数

$AC_BLOCKTYPE		$AC_BLOCKTYPEINFO				
值：		值：				
0	不等于0	T	H	Z	E	意义：
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1/4个圆弧返回
	3	3	0	0	3	WAB:以半圆返回
						刀具补偿：
	4	4	0	0	1	STOPRE之后的返回程序段
	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）
	4	4	0	0	5	去除补偿时返回程序段
	4	4	0	0	6	重新激活WRK时返回程序段
	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）

$AC_BLOCKTYPE		$AC_BLOCKTYPEINFO				
值：		值：				
0	不等于0	T	H	Z	E	意义：
原始程序段	中间程序段					中间程序段触发器：
						精磨削：
	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，带：
	6	6	0	0	1	切向轴线性运动，没有提刀运动
	6	6	0	0	2	切向轴非线性运动（多项式），没有提刀运动
	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	当到达确定的提刀位置时，首先启动提刀运动，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
						编译循环：
	8	ID应用				编译 - 循环 - 应用的ID，它产生该程序段

注意

$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 中，千位不可不等于 0 。

T:千位

H:百位

Z:十位

E:个位

$AC_SPLITBLOCK	
值：	**意义：**
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。

修整程序段的计数举例

```
$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

；所有的精磨削程序段在$AC_MARKER[0]中计数
ID = 1 WHENEVER ($AC_TIMEC ==0) AND ($AC_BLOCKTYPE==5) DO _
 $AC_MARKER[0]= $AC_MARKER[0] + 1
...
；用G641产生的精磨削程序段在$AC_MARKER[1]中计数
ID = 2 WHENEVER ($AC_TIMEC ==0) AND ($AC_BLOCKTYPEINFO==5001) DO _
 $AC_MARKER[1]= $AC_MARKER[1] + 1
...
；用G642产生的精磨削程序段在$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 输出辅助功能

功能

在满足条件的情况下，每个加工程序段可以最多输出10个M-,H-和S-功能。

使用动作识别字 "DO" 启动辅助功能输出。

在插补节拍中**立即** 输出辅助功能。通过机床数据定义的辅助功能输出时间无效。

当条件满足后，确定输出时间。

辅助功能的有效性由PLC决定，例如使用NC起动。

举例：

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
```

允许的关键字

只能使用关键字 WHEN 或者 EVERY在逐段有效的同步动作中（没有模态ID）对辅助功能进行编程。

注意

从一个运动同步动作中不可以：

- M0, M1, M2, M17, M30:程序停止/结束（在工艺循环时可以为M2,M17,M30）。
- M70:主轴功能
- M6 或者通过机床数据设置的用于换刀的M功能
- M40, M41, M42, M43, M44, M45:齿轮级切换

举例

```
WHEN $AA_IW[Q1]>5 DO M172 H510          ;当Q1轴的实际值高于 5 mm 时，将辅助功能 M172 和
                                        H510 发送给 PLC
                                        。
```

10.4.2 设定读入禁止 (RDISABLE)

功能

当满足田间时，使用 RDISABLE 将主程序中的其它程序段处理停住。编程的运动同步动作继续执行，后面的程序段继续处理。

在有RDISABLE的程序段的起始处，始终触发准停，而与RDISABLE是否有效无关。

举例

与外部的输入端无关，以插补节拍启动程序。

```
...
WHENEVER $A_INA[2]<7000 DO RDISABLE        ;如果在输入端2上电压低于
                                           ;7V，就停止继续执行程序 (1000= 1V).
N10 G1 X10                                 ;当条件满足时，
                                           ;读入禁止就会在N10的结束位置发挥作用
N20 G1 X10 Y20
...
```

10.4.3 取消进给停止 (STOPREOF)

功能

如果是显式编程的进给停止STOPRE或者是通过一个激活的同步动作隐式激活的进给停止，只要满足了条件， STOPREOF 就会在结束下一个加工程序段后取消进给停止。

注意

STOPREOF 必须使用关键字 WHEN 并且逐段（没有ID号）进行编程。

举例

在程序段结束处快速的程序分支。

```
WHEN $AC_DTEB<5 DO STOPREOF                ;当与程序段结束处的距离超过 5 mm时，就取消进给停止。
G01 X100                                   ;在线性插补执行完毕后取消进给停止
                                           。
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。
```

10.4.4 删除剩余行程 (DELDTG)

功能

与一个条件相关，剩余行程删除可以释放，用于轨迹和所说明的轴。

可以使用：

- 快速、预备的剩余行程删除
- 剩余行程删除，不带预置

10.4.5 剩余行程删除，带预置 (DELDTG, DELDTG("轴 1 ~ x"))

功能

预置式剩余行程删除DELDTG允许对释放结果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。
- 剩余行程删除的条件有很大的可能性满足。

注意

置于 DELDTG 后面的括号中的轴名称仅对一个 定位轴有效。

编程

轨迹的剩余行程删除
```
DO DELDTG
```
或者

轴向剩余行程删除
```
DO DELDTG(轴1, 轴2, ...)
```

轨迹快速剩余行程删除举例

```
WHEN $A_IN[1]==1 DO DELDTG
N100 G01 X100 Y100 F1000          ;当输入端已设定时，取消运动
                                  。
N110 G01 X…
IF $AA_DELT>50…
```

快速轴向剩余行程删除举例

```
停止一个编程的定位运动：
ID=1 WHEN $A_IN[1]==1 DO MOV[V]=3 FA[V]=700          ;起动轴
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
```

说明

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。

注意

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) 逐段同步和激光功率控制

功能

使用 FCTDEF 可以定义三阶多项式，形式为 $y=a_0+a_1x+a_2x^2+a_3x^3$。这些多项式被联机刀具补偿FTOC和分析功能SYNFCT用来从主过程变量（实时变量）中计算函数值。

编程

```
FCTDEF(多项式编号，LLIMIT,ULIMIT,a0,a1,a2,a3)
```

参数

要么以逐段同步方式使用函数 FCTDEF 定义多项式

FCTDEF	定义分析函数
n	多项式序号
多项式编号	3阶多项式序号
LLIMIT	功能值下限
ULIMIT	功能值上限
a_0, a_1, a_2, a_3	多项式系数

或者通过系统变量

$AC_FCTLL[n]	功能值下限
$AC_FCTUL[n]	功能值上限
$AC_FCT0[n]	a_0
$AC_FCT1[n]	a_1
$AC_FCT2[n]	a_2
$AC_FCT3[n]	a_3

注意

写入系统变量

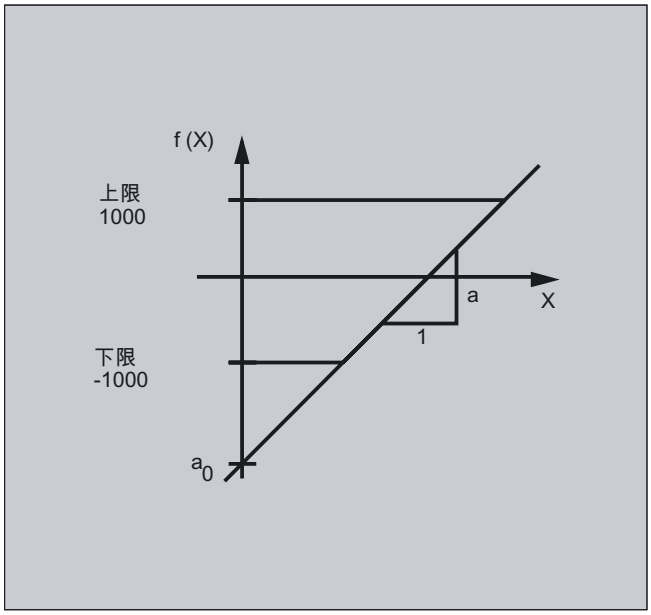
- 系统变量可以由零件程序，或者从一个同步动作写入。当从零件程序写入时，必须通过编程STOPRE来进行逐段同步写入。
- 系统变量 $AC_FCTLL[n], $AC_FCTUL[n], $AC_FCT0[n] 直到 $AC_FCTn[n] 可从同步动作中修改

在从同步动作中写入时，多项式系数和功能值界限立即生效。

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)
```

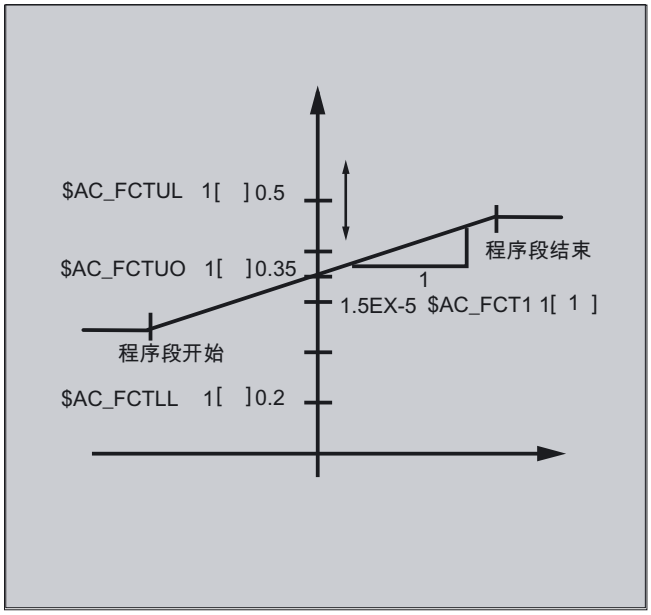


激光功率控制举例

通过变量地多项式定义

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。




```
$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            ;联机修改上限。
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1来控制激光功率控制系统
```

注意

通过SYNFCT使用上面定义的多项式。

10.4.7 分析功能 (SYNFCT) 以及AC调节 (相加式和倍增式)

功能

SYNFCT 以和加工同步的方式读取实时变量 (例如模拟输出，实际值等等)
并且利用某个最多三阶的分析多项式（FCTDEF）来计算功能值（例如修调量，速度，轴位置等等）。计算结果被发送给实时变量并且使用FCTDEF来限制上下限。

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。

编程

SYNFCT(多项式编号，实时变量输出，实时变量输入)

参数

作为实时输出变量，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
- 它们作为位置补偿进入处理过程
- 它们直接进入处理过程

。

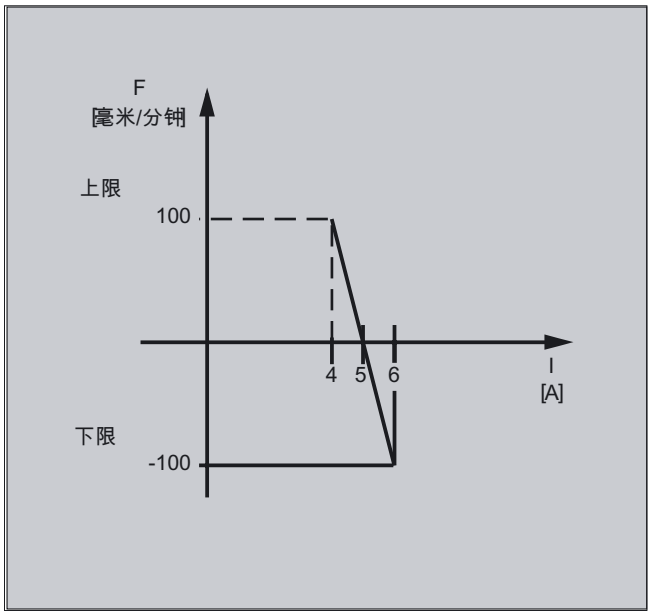
DO SYNFCT	激活分析功能
多项式编号	用FCTDEF定义的多项式（参见章节"多项式定义"）
实时变量输出端	写实时变量
实时变量输入端	读实时变量

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)
```

2. 启用AC调节

```
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。

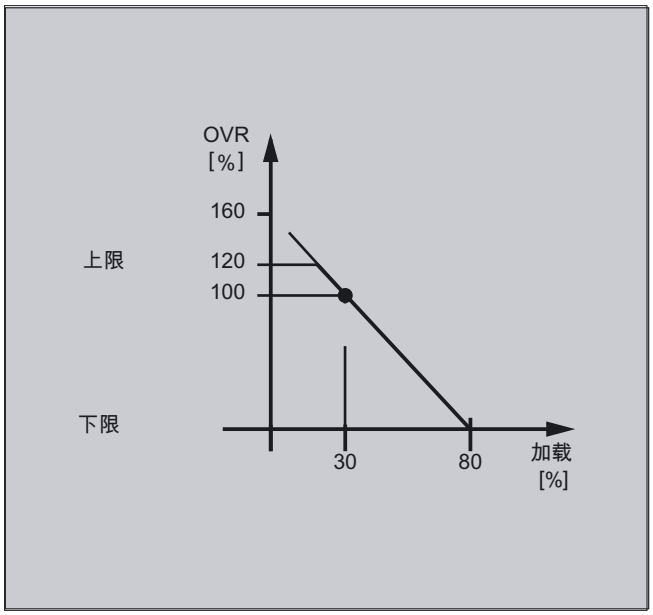
AC调节举例（倍增式）

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)
```

2. 启用AC调节

```
ID=1 DO SYNFCT(2,$AC_OVR,$AA_LOAD[x])
```

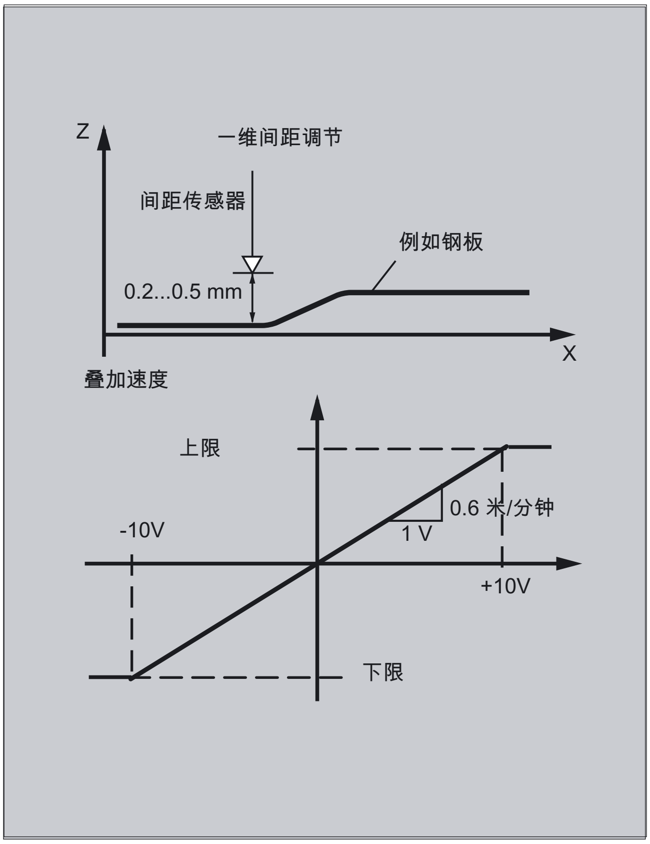
;通过 `$AA_LOAD[x]` 读取当前轴负荷 (% 最大驱动电流)，
;使用上述定义的多项式计算进给修调率。

10.4.8 使用限定的补偿系数调节间距 $AA_OFF_MODE

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



注意事项

超调的调节回路的回路放大与插补节拍的设定相关。

消除方法：读并计算MD，用于插补节拍。

注意

通过MD 32020限制叠加式插补器的速度：JOG_VELO 当Ipo节拍为12 ms时。速度公式：

$$\frac{0.120mm}{0.6ms}/mK = 0.6\frac{m}{\min}/K$$

举例

子程序：启用间距调节

```
%_N_AON_SPF                                    ;启用间距调节的子程序
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])         ;间距调节已激活
ID=2 WHENEVER $AA_OFF_LIMIT[Z]<>0              ;当超过极限范围时锁止轴X
 DO $AA_OVR[X] = 0
RET
ENDPROC
```

子程序：关闭间距调节

```
%_N_AOFF_SPF
PROC AOFF                                      ;间距调节子程序关
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

```
%_N_MAIN_MPF
AON                                            ;启用间距调节
...
G1 X100 F1000
AOFF                                           ;关闭间距调节
M30
```

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。

限制实时变量输出范围：

可以绝对需要补偿的值（实时变量输出）限制到保存在设定数据SD 43350:AA_OFF_LIMIT中的值。

通过机床数据 MD 36750:AA_OFF_MODE 确定叠加间距的方式：

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 可以根据使用FCTDEF依据某个基准值（例如该值可以是某个轴的实际值）所编程的某个多项式实现某个几何轴的叠加运动。

由此您也可以编程模态的在线刀具补偿，或者距离调节作为同步动作。

该功能可用于加工工件、校正同一个通道中或者拨不同通道（加工通道和校正通道）中的砂轮。

边界条件和确定砂轮的修整适用于FTOC，类似于用PUTFTOCF的在线刀具补偿。有关信息可在“刀具补偿”一章中查阅。

编程

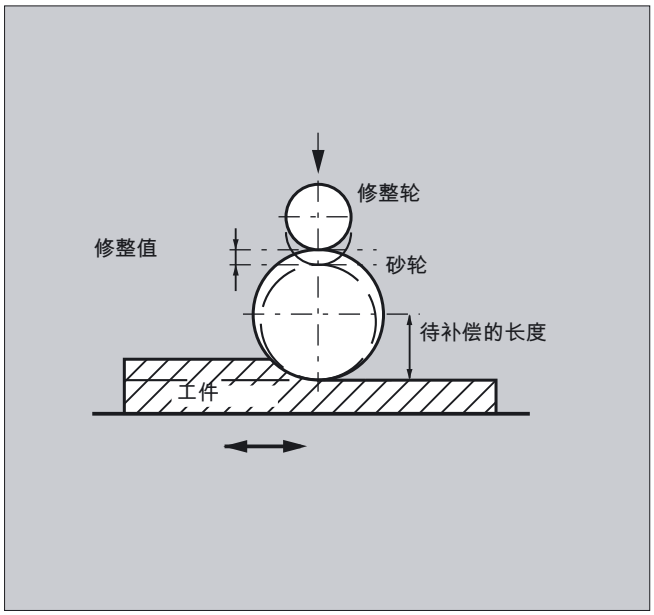
FTOC(多项式编号，EV，长度1_2_3 或者 半径4，通道，主轴)

参数

DO FTOC	执行联机刀具补偿
多项式编号	使用FCTDEF定义多项式，参见本章中"多项式定义"节。
EV	实时变量，通过说明的多项式计算一个功能值。
长度1_2_3	长度补偿 ($TC_DP1 至 3)或者半径补偿，计算的功能值加到该值。
半径4	
通道	通道号，补偿在该通道中生效。在有效通道中的一个补偿在此没有说明。在目标通道中必须已启用FTOCON。
主轴	仅在不需要修正有效主轴时予以说明。

举例

在该示例中，应该修调当前有效的、正在使用的砂轮的长度。




```
%_N_ABRICHT_MPF
FCTDEF(1,-1000,1000,-$AA_IW[V],1)          ;定义功能。
ID=1 DO FTOC(1,$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中被作为
                                           补偿值添加到已激活砂轮的长度3上
                                           。
WAITM(1,1,2)                               ;与加工通道同步
G1 V-0.05 F0.01 G91                        ;用于修整的横向进给运动
G1 V-0.05 F0.02
...
CANCEL(1)                                  ;取消在线补偿
...
```

10.4.10 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/ “行程参数”一章

/FBSY/ "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。

举例1

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

```
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=10 mm X=10 mm
N160 G70 Z10 F10                                ;Z=254 mm X=254 mm
N170 G71 Z10 F10                                ;Z=10 mm X=10 mm
N180 M30
```

举例2

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。

```
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=10 mm X=10 mm
N160 G70 Z10 F10                                ;Z=254 mm X=10 mm (X 始终定位到10 mm)
N170 G71 Z10 F10                                ;Z=10 mm X=10 mm
N180 M30
```

锁止某个已编程轴运动举例

如果不要在程序段开始处起动轴运动，可将从某个同步动作到所需开始时间点的轴修调率保持为0。

```
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。
```

10.4.11 定位轴 (POS)

功能

与从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。

编程

POS[轴]=值

参数

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）

举例

```
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。
ID=1 EVERY $AA_IM[B]>75 DO POS[U]=$AA_MW[V]-$AA_IM[W]+13.5
；轴U运行由实时变量计算的位移。
```

10.4.12 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。

编程

MOV[轴] = 值

参数

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

注意

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。

举例

```
... DO MOV[U]=0                    ;U轴被停止
```

10.4.13 轴向进给 (FA)

功能

指令轴的轴向进给为模态有效。进给值要么设定为固定值，要么以实时方式从实时变量中形成。

编程

FA[轴]=进给

举例

```
ID=1 EVERY $AA_IM[B]>75 DO POS[U]=100 FA[U]=990
；固定规定进给值
ID=1 EVERY $AA_IM[B]>75 DO POS[U]=100 FA[U]=$AA_VACTM[W]+100
；由实时变量构成进给值
```

10.4.14 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。

10.4.15 轴协调

功能

标准情况下，一个轴或者由零件程序运动程序段运动，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

```
N10 G01 X100 Y200 F1000                    ;X轴，在零件程序中编程
…
N20 ID=1 WHEN $A_IN[1]==1 DO               ;当存在数字输入时，从同步动作中开始定位
POS[X]=150 FA[X]=200
…
CANCEL(1)                                  ;取消同步动作
```

```
…
N100 G01 X240 Y200 F1000
;X成为轨迹轴；如果数字输入为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

```
ID=1 EVERY $A_IN[1]>=1 DO POS[V]=100 FA[V]=560
;当数字输入 >= 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。
```

10.4.16 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

编程

DO PRESETON(轴，值)

参数

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。

举例

移动某个轴的控制零点

```
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)
```

10.4.17 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。

起动/停止/定位主轴举例

```
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

```
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)  ;定位主轴
DO SPOS=0
```

10.4.18 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。

编程

DO TRAILON(跟随轴, 引导轴, 耦合系数)	接通联动
DO TRAILOF(跟随轴, 引导轴, 引导轴 2)	关闭联动

参数

激活异步联动：	
... DO **TRAILON**(FA, LA, Kf)	使用： FA:跟随轴 LA:引导轴 Kf:耦合系数
取消异步联动：	
... DO **TRAILOF**(FA, LA, LA2)	使用： FA:跟随轴 LA:引导轴 LA2:引导轴2，选件

举例

```
$A_IN[1]==0 DO TRAILON(Y,V,1)     ;当数字输入为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
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

为了使某个已经偶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
…
```

10.4.19 引导值偶合 (LEADON, LEADOF)

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偶合的某个曲线图表。

编程

DO LEADON(跟随轴，引导轴，曲线图表编号，OVW)	接通引导值耦合
DO LEADOF(跟随轴，引导轴，引导轴2)	关断引导值耦合

参数

接通轴向引导值耦合： ...DO **LEADON**(FA,LA,NR,OVW)	使用： 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偶合

关断轴向引导值耦合：

```
...DO LEADOF(FA,LA)                    使用：
                                       FA:跟随轴
                                       LA:引导轴
```

通过同步动作释放存取 RELEASE

为了释放某个需要偶合的轴以便通过同步动作进行存取，必须预先为需要偶合的跟随轴调用功能RELEAS。

举例：

```
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，不给某个现有的偶合规定新的曲线图表。必须事先关断现有的偶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。这样来影响偶合的某个新同步动作。

使用OVW=1来修改现有偶合的曲线图表

使用 OVW=1 可以给某个现有的偶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：线材的加工轴，MKS
Y-轴：剪断装置与线材在其中“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
启用偶合， LEADON
断开偶合， LEADOF
设定实际值， PRESETON

```
% _N_SCHERE1_MPF
;$PATH=/_N_WKS_DIR/_N_DEMOFBE_WPD
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偶合
N700 CTABDEF(Y,X,1,0)                  ;表定义
N800 X=30 Y=30                         ;值组对
N900 X=R13 Y=R13
```

```
N1000 X=2*R13 Y=30
N1100 CTABEND                                   ;表定义结束
N1200 PRESETON(X1,0)                            ; 用于开始的PRESET
N1300 Y=R6 G0                                   ; 起始位置Y轴，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
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 $R1==1 DO MOV[X]=1              ; 带材轴始终处于运动状态
FA[X]=$R4
N2200 M30
```

10.4.20 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”

编程

轴向测量，没有剩余行程删除

MEAWA[轴] = (模式，触发事件_1, ..._4)

或者

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))

参数

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 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

10.4.21 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。

编程

SETM(标记序号)

或者

CLEARM(标记序号)

参数

DO SETM(标记编号)	设置等待标记，用于通道
DO CLEARM(标记编号)	删除等待标记，用于通道

SETM

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序号。

CLEARM

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序号。

10.4.22 故障应答 (SETAL)

功能

故障应答可以用同步动作编程，通过询问状态变量和释放相应的动作。

对故障的可能应答：

- 轴停止：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

编程

SETAL(报警编号)

参数

DO SETAL(报警编号)	设置循环报警

举例

```
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(61000)
；如果轴X1和X2之间的安全距离太小，则设置报警。
```

10.4.23 向固定止挡运动 (FXS 和 FOCON/FOCOF)

功能

用于**向固定止挡运动** 的指令可使用零件程序指令 FXS, FXST 和 FXSW 在同步动作/工艺循环中进行编程。
可在没有运动的情况下激活，力矩会立即受到限制。一旦轴由给定值运动，则就对挡块进行监控。

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。

参数

FXS[轴]	仅在带数字驱动 (VSA, HSA, HLA)的系统中选择
FXST[轴]	改变夹紧扭矩FXST
FXSW[轴]	改变监控窗口FXSW
FOCON[轴]	激活模态有效的扭矩/力 - 极限
FOCOF[轴]	取消扭矩/力 - 极限
FOCON/FOCOF	轴的编程在方括号中。允许的有： – 几何轴 – 标识符 – 通道轴 – 标识符 – 加工轴 – 标识符

注意

一个选择仅允许进行一次。

向固定止挡运动举例 (FXS)

通过一个同步动作释放

```
Y-轴：                                          ;静态同步动作
激活：
N10 IDS=1 WHENEVER (($R1==1) AND                ;通过设置 $R1=1，为
             ($AA_FXS[y]==0)) DO
             $R1=0 FXS[Y]=1 FXST[Y]=10          ;轴Y激活FXS，将有效力矩
             FA[Y]=200 POS[Y]=150               ;降低到 10% 并且开始向着
                                                ;止挡方向运动
                                                ;
N11 IDS=2 WHENEVER ($AA_FXS[Y]==4) DO           ;是要止挡已被识别
             FXST[Y]=30
                                                ;($AA_FXS[Y]==4)，就将力矩增大到
                                                ; 30%
N12 IDS=3 WHENEVER ($AA_FXS[Y]==1) DO           ;在到达止挡之后，就
             FXST[Y]=$R0
                                                ;依据R0对力矩进行控制
N13 IDS=4 WHENEVER (($R3==1) AND                ;依据
             ($AA_FXS[Y]==1)) DO
             FXS[Y]=0                           ;R3来取消并且
             FA[Y]=1000 POS[Y]=0                ;返回
N20 FXS[Y]=0 G0 G90 X0 Y0                       ；正常的程序运行：
                                                ;轴Y，用于
N30 RELEASE(Y)                                  ；使能同步动作中运动
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
```

激活力矩/力限制举例 (FOC)

```
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
```

多次选择

当通过一个有缺陷的编程在激活 FXS[轴]=1) 之后再次调用该功能时，就会发出报警20092 "向固定止挡运动上处于激活状态“。

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。

```
N10 R1=0
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。

举例：

```
N10 G0 G90 X0 Y0
N20 WHEN $AA_IW[X] > 17 DO FXS[X]=1         ;当 X 到达某个大于17mm的位置时，
                                            ;就
N30 G1 F200 X100 Y110                       ;激活FXS
```

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。

10.4.24 在同步动作中确定

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。

举例

```
ID=2 EVERY $AC_BLOCKTYPE==0 DO $SR1 = $AC_TANEB
```

10.4.25 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

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：

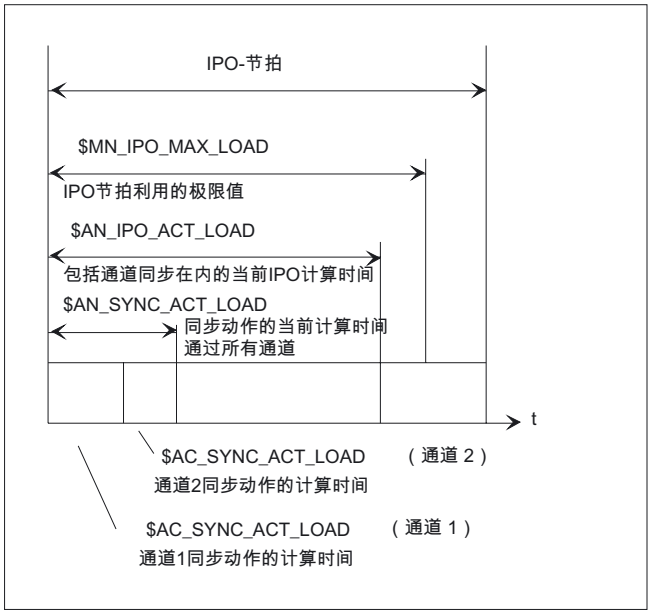
$AA_OVR * $AA_PLC_OVR 或者

$AC_OVR * $AC_PLC_OVR

10.4.26 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。



参数

仅当机床数据 $MN_IPO_MAX_LOAD 大于0时，这些变量才会有有效值。在其它情况下，变量始终说明总的计算时间。总的计算时间由以下构成：

- 同步动作时间，
- 位置调节时间，以及
- 其余IPO计算时间

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总计计算时间(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。

同步动作中的系统变量

这些系统变量可从同步动作中写入/读取：	
$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
$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_SYNC_MAX_LOAD	最长的计算时间，用于所有通道的同步动作
$AC_SYNC_MAX_LOAD	通道中同步动作的最长的计算时间
这些系统变量在每次写入访问时被复位成当前的负荷，与所写入的值无关。	

举例

```
$MN_IPO_MAX_LOAD = 80                                   MD:IPO节拍的使用极限
                                                        只要 $AN_IPO_LOAD_PERCENT > 80 %，则
                                                        $AN_IPO_LOAD_LIMIT 就置为 TRUE
                                                        （真）。
N01 $AN_SERVO_MAX_LOAD=0
N02 $AN_SERVO_MIN_LOAD=0
N03 $AN_IPO_MAX_LOAD=0
N04 $AN_IPO_MIN_LOAD=0
N05 $AN_SYNC_MAX_LOAD=0
N06 $AC_SYNC_MAX_LOAD=0
N10 IDS=1 WHENEVER $AN_IPO_LOAD_LIMIT == TRUE DO M4711 SETAL(63111)
N20 IDS=2 WHENEVER $AN_SYNC_TO_IPO > 30 DO SETAL(63222)
N30 G0 X0 Y0 Z0
...
N999 M30
```

如果超出总的使用极限，则第一个同步动作产生一个辅助功能输出，并且有一个报警。

如果同步动作在IPO计算时间中的分量（通过所有通道）超出30%，则第二个同步动作产生一个报警。

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。对于用户，调用如同子程序。

在一个通道中可以并行处理几个工艺循环或者动作。

应用

工艺循环作为轴程序：每个工艺循环仅控制一个轴。因此不同的轴运行可以在同一个插补节拍中由事件控制启动。零件程序在极端情况下只可用来管理同步动作。

编程

程序末尾编程有 M02/M17/M30/RET 。每个程序段最多可编程一个轴运动。

参数传送

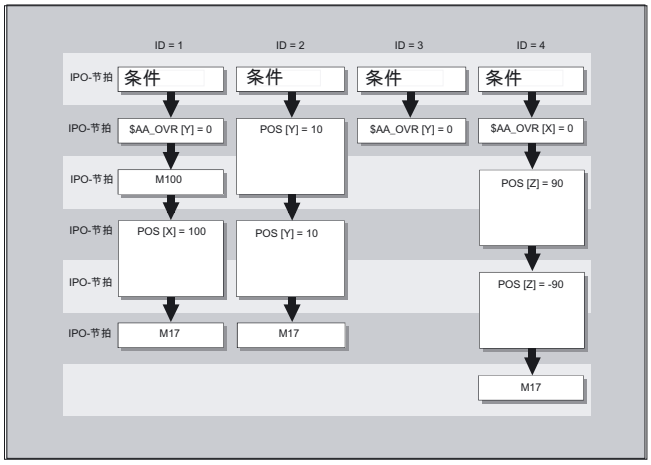
不可以进行参数传送。

一旦条件满足，则启动工艺循环。在定位轴中，需要几个IPO节拍用于执行。其它的功能（OVR）执行一个节拍。在工艺循环中，程序段按顺序执行。

如果在同一个插补节拍中调用几个动作，这几个动作相互之间排斥，则启动同步动作中用更高ID号调用的动作。

举例

通过设置数字输入来起动轴程序。



主程序：

```
ID=1 EVERY $A_IN[1]==1 DO ACHSE_X
ID=2 EVERY $A_IN[2]==1 DO ACHSE_Y
ID=3 EVERY $A_IN[3]==1 DO $AA_OVR[Y]=0
ID=4 EVERY $A_IN[4]==1 DO ACHSE_Z
M30
```

当
;输入1为1，启动轴程序 X
;输入2为1，启动轴程序 Y
;输入3为1，将轴Y的修调率设置成 0
;输入4为1，启动轴程序 Z

工艺循环 ACHSE_X:

```
$AA_OVR [Y] = 0
M100
POS[X]=100 FA[X]=300
M17
```

工艺循环 ACHSE_Y:

```
POS[Y]=10 FA[Y]=200
POS [Y] = -10
M17
```

```
工艺循环 ACHSE_Z:
$AA_OVR [X] = 0
POS[Z]=90 FA[Z]=250
POS [Z] = -90
M17
```

10.5.1 锁止，释放，中断 (LOCK, UNLOCK, RESET)

功能

工艺循环的过程可以由同步动作或者由一个工艺循环禁止、释放和中断。

编程

LOCK(n, n, ...)	禁止工艺循环，其有效动作被中断
UNLOCK(n, n, ...)	释放工艺循环
RESET(n, n, ...)	中断工艺循环，其有效动作被中断
n	同步动作的识别号

参数

锁止工艺循环，LOCK

可使用 LOCK 从另一个同步动作或者从某个工艺循环中来锁止工艺循环。

释放工艺循环，UNLOCK

可使用 UNLOCK 从另一个同步动作/工艺循环中重新释放被锁止的工艺循环。使用 UNLOCK 在当前位置继续该工艺循环，同样还有某个被中断的定位过程。

中断工艺循环，RESET

使用 RESET 从另一个同步动作或者从某个工艺循环中中断工艺循环。

PLC一侧闭锁

ID编号为**n=1 ...的模态同步动作64**可以被PLC锁止。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。

注意

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。

举例

锁止工艺循环，LOCK

```
N100 ID=1 WHENEVER $A_IN[1]==1 DO M130
...
N200 ID=2 WHENEVER $A_IN[2]==1 DO LOCK(1)
```

释放工艺循环，UNLOCK

```
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)
```

中断工艺循环，RESET

```
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

带名称ID(S)=n的模态有效的同步动作仅可以用CANCEL直接由零件程序删除。

注意

由一个删除的同步动作启动的、仍然在运行的动作根据编程结束。

编程

CANCEL(n, n, ...)	删除同步动作
n	同步动作的识别号

举例

```
N100 ID=2 WHENEVER $A_IN[1]==1 DO M130
...
N200 CANCEL(2)                                    ;删除同步动作编号2
```

10.7 边界条件

功能

在边界条件中对同步动作在出现重要事件时如何作出反映进行处理。它们可以是：

- **上电**
- **运行方式转换**
- **复位**
- **NC-停止**
- **程序结束**
- **程序段搜索**
- **通过异步子程序中断程序 ASUP**
- **重新定位 REPOS**
- **使用 CANCEL 取消**

事件

- **上电**

  当执行上电时，原则上没有同步动作激活。静态同步动作仍然可以在执行上电时与某个被 PLC 启动的异步子程序 (ASUP) 一起被激活。

- **运行方式转换**

  除了变换操作方式之外，使用关键字 IDS 所激活的同步动作保持激活状态。所有其余同步动作在变换操作方式时为非激活状态（例如定位轴）并且当重新定位和恢复到自动操作方式时重新进入激活状态。

- **复位**

  使用NC-复位可中断所有通过同步动作起动的动作。静态的同步动作仍然有效。
  由它们可以启动新的动作。使用同步动作或者某个工艺循环中的指令 RESET
  可以将一个模态有效的同步动作复位。如果一个同步动作复位，但是同时同步动作激活的定位轴运动却仍有效，则停止定位轴运行。已经执行的 WHEN 类型的同步动作在 RESET 之后将不再被处理。

RESET之后的性能		
同步动作/ 工艺循环	**模态有效/程序段方式**	**静态 (IDS)**
	有效的动作停止，同步动作删除	有效动作停止，工艺循环复位
轴/ 定位的主轴	运动被停止	运动被停止
转速控制的主轴	$MA_SPIND_ACTIVE_AFTER_RESET==1: 主轴保持激活 $MA_SPIND_ACTIVE_AFTER_RESET==0: 主轴停止。	
引导值耦合	$MC_RESET_MODE_MASK, 位13 == 1: 引导值偶合保持激活 $MC_RESET_MODE_MASK, 位13 == 0: 引导值偶合被解除	
测量过程	由同步动作启动的测量过程停止	由静态同步动作启动的测量过程停止

- **NC-停止**

  静态 同步动作在NC-Stop时保持激活。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 (参见上一个子章节) 中断同步动作。

程序结束之后的性能		
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**
轴/ 定位的主轴	M30延迟，直至轴/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 : 主轴保持激活 $MA_SPIND_ACTIVE_AFTER_RESET==0 : 主轴停止 当转换运行方式时，主轴保持激活	主轴保持有效
引导值耦合	$MC_RESET_MODE_MASK, 位13 == 1: 引导值保持激活 $MC_RESET_MODE_MASK, 位13 == 0: 引导值偶合被解除	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停止	由静态同步动作启动的测量过程保持激活

- **程序段搜索**

  程序段搜索过程中所找到的同步动作被汇总并且在NC-Start时被分析，相应的动作也同样会被启动。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。

- **通过异步子程序中断程序 ASUP**

  ASUP-开始：
  模态和静态运动同步动作保持不变并且也会在异步子程序中有效。
  ASUP-结束：
  如果没有使用 REPOS 继续执行异步子程序，则在异步子程序中所修改的模态和静态运动同步动作将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。从异步子程序中被修改的模态同步动作在执行 REPOS 之后不会在处理剩余程序段时有效。
  使用 FCTDEF 编程的多项式系数不受异步子程序和 REPOS 的影响。不管其是在那里编程的，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。

- **使用 CANCEL 取消**

  当使用 CANCEL 取消某个已激活的同步动作时，不会影响已激活的动作。定位运动将按照所编的程序结束。
  使用指令 CANCEL 可以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 指令之前使用轴向剩余行程删除来使轴运动停止：

使用 CANCEL 取消举例

```
ID=17 EVERY $A_IN[3]==1 DO POS[X]=15 FA[X]=1500      ; 启动定位轴运行
...
WHEN ...DO DELDTG(X)                                 ; 结束定位轴运行
CANCEL(1)
```

11 摆动

11.1 异步摆动

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，可以达到一个连续的横向进给。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

- 异步摆动在特定的轴上超越程序段范围有效。
- 通过零件程序，保证一个程序段同步的摆动运动开启。
- 几个轴的共同插补和摆动距离的叠加是不可能的。

编程

通过后面的地址，可以接通执行相应的NC程序，并且由零件程序影响异步摆动。

编程的值在主运行中与程序段同步登记到相应的设定数据，并且直至下一次修改一直保持有效。

启动/关闭摆动：OS

OS[轴] = 1:启动

OS[轴] = 0:关闭

参数

OSP1 [轴]=	返回点1的位置（摆动：左侧返回点）
OSP2 [轴]=	返回点2的位置（摆动：右侧返回点）
OST1 [轴]=	在换向点处的保持时间，单位秒
OST2 [轴]=	
FA [轴]=	摆动轴进给
OSCTRL [轴]=	（设置选件，复位选件）

OSNSC [轴]=	修光冲程数
OSE [轴]=	终点位置
OS [轴]=	1 = 接通摆动轴；0 = 关断摆动轴

在返回点中的停止时间：OST1, OST2

停止时间	在精确停止范围内、返回点上的运动特性
-2	继续插补，无需等待精确停止
-1	粗等待精确停止
0	精等待精确停止
>0	精等待精确停止，并且接着等候停留时间

停止时间的单位与通过 G4 编程的停止时间的单位相同。

摆动轴应当在两个返回点之间摆动的举例

摆动轴Z应该在10和100之间摆动。换向点1以精准停返回，换向点2以粗准停返回。应该用进给250进行摆动轴的加工。在加工结束处应当执行三次修光行程，并且使用摆动轴控制终点位置200。横向进给轴的进给是1，在X方向的横向进给在15处结束。

```
WAITP(X,Y,Z)                          ；起始位置
G0 X100 Y100 Z100                     ;转换到定位轴运行方式
N40 WAITP(X,Z)
N50 OSP1[Z]=10 OSP2[Z]=100 ->         ;返回点1，返回点2
-> OSE[Z]=200 ->                      ;终点位置
-> OST1[Z]=0 OST2[Z]=-1 ->            ;返回点1上的停止时间：精准停；
                                      ;返回点2上的停止时间：粗准停
-> FA[Z]=250 FA[X]=1 ->               ;摆动轴进给，进给轴
-> OSCTRL[Z]=(4,0) ->                ;调整选项
-> OSNSC[Z]=3 ->                     ;三次修光行程
N60 OS[Z]=1                           ;启动摆动
N70 WHEN $A_IN[3]==TRUE ->            ;剩余行程删除
-> DO DELDTG(X)
N80 POS[X]=15                         ;X轴初始位置
N90 POS[X]=50
N100 OS[Z]=0                          ;停止摆动
M30
```

-> 可以在某个程序段中编程。

使用联机修改返回位置进行摆动的举例

设定数据

异步摆动所需的调整数据可以在零件程序中设置。

如果在零件程序中直接描述设定数据，则修改在进刀时就已经生效。可以通过进给停止 STOPRE 来实现同步特性。

```
$SA_OSCILL_REVERSE_POS1[Z]=-10
$SA_OSCILL_REVERSE_POS2[Z]=10

G0 X0 Z0
WAITP(Z)

ID=1 WHENEVER $AA_IM[Z] < $$AA_OSCILL_REVERSE_POS1[Z] DO $AA_OVR[X]=0
ID=2 WHENEVER $AA_IM[Z] < $$AA_OSCILL_REVERSE_POS2[Z] DO $AA_OVR[X]=0
                                                    ; 如果摆动轴的实际值
                                                    ; 已经超过换向点
                                                    ; 横向进给轴保持。
OS[Z]=1 FA[X]=1000 POS[X]=40                        ; 接通摆动
OS[Z]=0                                             ; 关断摆动
M30
```

说明

适用于摆动轴的有：

- 每个轴可以作为摆动轴使用。
- 可以同时有几个摆动轴有效（最大：定位轴个数）。
- 对于摆动轴而言，线性插补 G1 始终处于激活状态，与程序中当前有效的G指令无关。

摆动轴可以：

- 是动态转换的输入轴，
- 是龙门轴和联动轴时的引导轴，
- 运行
  - 没有急冲限制 (BRISK) 或者
  - 有急冲限制 (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 1

换向点位置 = 偏移之和+编程的值

- 相对尺寸

OSP1[Z]=IC(值)

换向点位置 = 换向点1+编程值

举例：

```
N10 OSP1[Z] = 100 OSP2[Z] = 110
.
.
N40 OSP1[Z] = IC(3)
```

注意

WAITP(轴)：

- 要使用几何轴进行摆动，就必须使用 WAITP 将其释放来进行摆动。
- 在摆动结束之后，用该指令把摆动轴再次作为定位轴登记，并且可以再次正常使用。

带有运动同步动作和停止时间的摆动， OST1/OST2

在已设置的停止时间结束之后，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影响这种时间特性。**

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。**

设置进给，FA

作为进给速度，适用于定义的定位轴的进给速度。如果没有定义进给速度，则机床数据中存储的值适用。

定义运动过程，OSCTRL

该运动过程的控制装置位置用设定选件和复位选件设置。

OSCTRL[摆动轴] = （调整选项，复位选项）

调整选项安装如下方式定义（复位选项会取消设置）：

复位选件

该选件被关断（仅在事先作为设定选件开通时才可以）。

设定选件

转换该选件。当编程 OSE (终点位置)时，选项4会隐式有效。

选件值	意义
0	在下一个换向点处关断摆动运动时停止（预设定）；仅可以复位值1和2
1	在换向点1处关断摆动运动时停止
2	在换向点2处关断摆动运动时停止
3	如果没有编程修光冲程，则在关断摆动运动时不返回换向点运行。
4	在修光后返回终点位置
8	如果通过剩余行程删除停止摆动运行：则紧接着执行修光冲程，有时要返回终点运行。
16	如果通过剩余行程删除停止摆动运行：则如同在关断时一样返回相应的换向点。
32	修改的冲程仅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	=修光行程被作为双行程执行（默认） 1=修光行程被作为单行程执行。

通过正号将多个选项相加在一起。

举例：

轴Z的摆动运动应当在关断时停止在返回点1中。此时应当

- 向某个终点位置运动，
- 某个被修改的进给应立即有效并且在删除剩余行程之后该轴立即停止。

```
OSCTRL[Z] = (1+4,16+32+64)
```

11.2 通过同步动作控制的摆动

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，可以在横向进给期间

- 继续执行摆动运动，或者
- 停止摆动运动，直至横向进给完全执行。

编程

1. **确定摆动参数**
2. **定义运动同步动作**
3. **分配轴，确定横向进给**

参数

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]=	摆动轴进给
OSCTRL[摆动轴]=	设定选件或者
OSNSC[摆动轴]=	修光冲程数
OSE[摆动轴]=	终点位置
WAITP(摆动轴)	使能用于摆动的轴

轴分配，横向进给

OSCILL[摆动轴] = (横向进给轴1, 横向进给轴2, 横向进给轴3)

POSP[横向进给轴] = (终点位置, 分度长度, 方式)

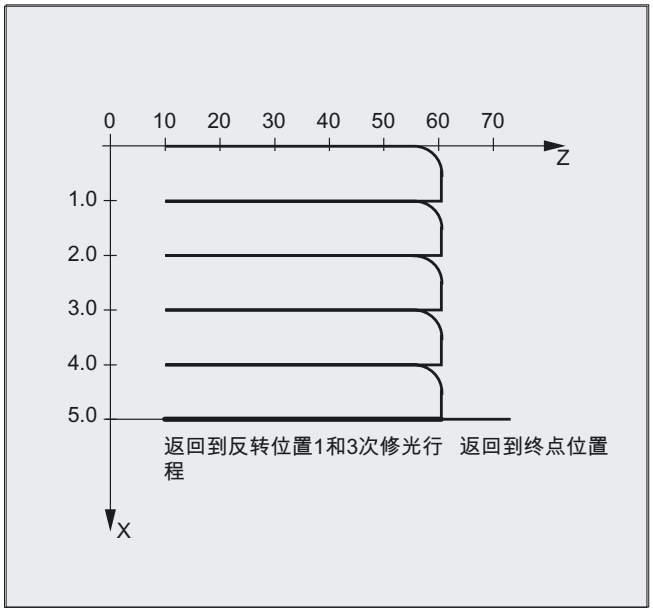
OSCILL	给摆动轴分配横向进给轴
POSP	确定最大进给和部分进给（参见文件和程序管理一章）
Endpos	横向进给轴最终位置，在所有分度横向进给开始运行之后。
部分长度	在换向点/换向区处的分度横向进给大小
模型	总的横向进给划分为几个分度横向进给 = 两个同样大小的剩余步距（预设置）； = 所有部分进给同样大

运动同步动作

WHEN… … DO	当…，则…
WHENEVER … DO	始终当…，则…

举例

在换向点1不应进行横向进给。在换向点2横向进给应该已经在换向点2之前距离ii2处进行，并且在换向点处摆动轴不等待分度横向进给结束。轴Z为摆动轴且轴X为进给轴。



1. 摆动的参数

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进给，横向进给轴X进给
OSCTRL[Z]=(2+8+16,1)	在换向点2处关断摆动运动；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

```
WHENEVER $AA_IM[Z]<$SA_OSCILL_REVERSE_POS2[Z]DO ->
-> $AA_OVR[X]=0 $AC_MARKER[0]=0
```

```
始终当                          当前的、在MKS中的摆动轴Z的位置
小于                            换向区2的开始，
则                              设定横向进给轴的轴向倍率到0%。
并且                            标记器（带变址0）设定到值0。
WHENEVER $AA_IM[Z]>=$SA_OSCILL_REVERSE_POS2[Z] DO $AA_OVR[Z]=0
始终当                          当前的、在MKS中的摆动轴Z的位置
大于等于                        换向位置2，
则                              设定摆动轴Z的轴向倍率到0%。
WHENEVER $AA_DTEPW[X]==0 DO $AC_MARKER[0]=1
始终当                          分度横向进给的剩余行程
相同于                          ，
则                              标记器（带变址0）设定到值1。
WHENEVER $AC_MARKER[0]==1 DO $AA_OVR[X]=0 $AA_OVR[Z]=100
始终当                          标记器带变址0
相同于                          ，
则                              设定横向进给轴X的轴向倍率到0%，由此防止一个过早的横向进给（摆动轴Z还没
                                有再次离开换向区2，但是横向进给轴X已经用于一个新的横向进给）。
                                设定摆动轴Z的轴向倍率到100%（由此取消第二个同步动作）。
```

-> 必须在一个程序段中编程

3. 启动摆动

```
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。

注意

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。

注意

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

注意

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_OSCILL_RESERVE_POS1[Z]+ii1 DO $AA_OVR[X] = 0
```

始终当	当前的、在MKS中的摆动轴的位置
大于	换向区1的开始，
则	设定横向进给轴的轴向倍率到0%。

换向区2：

```
WHENEVER $AA_IM[Z]<$SA_OSCILL_RESERVE_POS2[Z]+ii2 DO $AA_OVR[X] = 0
```

始终当	当前的、在MKS中的摆动轴的位置
小于	换向区2的开始，
则	设定横向进给轴的轴向倍率到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_OSCILL_RESERVE_POS1[Z] DO $AA_OVR[X] = 0 →
→ $AA_OVR[Z] = 100
```

始终当	当前的、在MKS中的摆动轴Z的位置
大于或者小于	换向点1的位置，
则	设定横向进给轴的轴向倍率到0%。
并且	摆动轴Z的轴向倍率到100%。

换向区2：

用于换向点2：

```
WHENEVER $AA_IM[Z]<>$SA_OSCILL_RESERVE_POS2[Z] DO $AA_OVR[X] = 0 →
→ $AA_OVR[Z] = 100
```

始终当	当前的、在MKS中的摆动轴Z的位置
大于或者小于	换向点2的位置，
则	设定横向进给轴的轴向倍率到0%。
并且	摆动轴Z的轴向倍率到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_OSCILL_RESERVE_POS1[Z] DO $AA_OVR[X] = 0 →
→ $AA_OVR[Z] = 100
```

始终当	当前的、在MKS中的摆动轴的位置
相同于	换向位置1，
则	设定摆动轴的轴向倍率到0%。
并且	横向进给轴的轴向倍率到100%。

换向区2：

```
WHENEVER $SA_IM[Z]==$SA_OSCILL_RESERVE_POS2[Z] DO $AA_OVR[X] = 0 →
→ $AA_OVR[Z] = 100
```

始终当	当前的、在MKS中的摆动轴的位置
相同于	换向位置2，
则	设定摆动轴的轴向倍率到0%。
并且	横向进给轴的轴向倍率到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。

注意

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
```

始终当	在WKS中横向进给轴X的分度横向进给剩余行程
相同于	为零，
则	将摆动轴的轴向修调率设定为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
```

始终当	在WKS中横向进给轴X的分度横向进给剩余行程
相同于	为零，
则	设置标记器（带变址1）到1。

2. 删除标记

```
WHENEVER $AA_IM[Z]<> $SA_OSCILL_RESERVE_POS1[Z] DO $AC_MARKER[1] = 0
```

始终当	当前的、在MKS中的摆动轴Z的位置
大于或者小于	换向点1的位置，
则	设定标记器1到0。

3. 阻止进给

```
WHENEVER $AC_MARKER[1]==1 DO $AA_OVR[X] = 0
```

始终当	标记器1
相同于	，
则	设定横向进给轴的轴向倍率到0%。

冲裁和步冲

12

12.1 激活, 非激活

12.1.1 启用或者关闭冲裁与步冲 (SPOF, SON, PON, SONS, PONS, PDELAYON/OF)

功能

激活/取消冲裁和步冲，PON/SON

使用PON和SON激活冲压或者步冲功能。SPOF结束所有的冲压和步冲专用的功能。
模态有效的指令PON和SON相互排斥，也就是说PON非激活SON，反之亦然。

冲压和步冲，带预应力，PONS/SONS

功能SONS和PONS同样打开冲压功能或者步冲功能。

与SON/PON相反 - 插补平面上的冲程控制 - 在这些功能中，在伺服级面上对冲程释放进行信号控制。由此您可以使用更高的冲程频率，以及更高的冲压功率。

在预应力信号计算期间所有导致步冲轴或者冲压轴位置改变的功能被闭锁。

举例： 手轮运行，通过PLC改变框架，测量功能。

带有延时的冲裁，PDELAYON/PDELAYOF

PDELAYON使冲压冲程延迟输出。模态有效的指令有预置的功能，并且通常位于PON之前。
在PDELAYOF之后又恢复到正常冲压。

编程

```
PONS G... X... Y... Z...
```

或者

```
SON G... X... Y... Z...
```

或者

```
SONS G... X... Y... Z...
```

或者

SPOF

或者

PDELAYON

或者

PDELAYOF

或者

PUNCHACC(Smin,Amin, Smax, Amax)

参数

PON	冲压开
PONS	冲压带预应力开
SON	步冲开
SONS	步冲带预应力开
SPOF	冲压，步冲关
PDELAYON	冲压带延迟开
PDELAYOF	冲压带延迟关
PUNCHACC	Wegabhängige Beschleunigung PUNCHACC (S_{min}, A_{min}, S_{max}, A_{max})
"S_{min}"	最小的孔距
"S_{max}"	最大的孔距
"A_{min}"	初始加速度 A_{min} 可以大于 A_{max}
"A_{max}"	最终加速度 A_{max} 可以小于 A_{min}

使用M指令

在使用宏指令技术时 ，您可以不用语言指令而是使用M指令 ：

DEFINE M25 AS PON	冲压开
DEFINE M125 AS PONS	冲压带预应力开
DEFINE M22 AS SON	步冲开
DEFINE M122 AS SONS	步冲带预应力开
DEFINE M26 AS PDELAYON	冲压带延迟开
DEFINE M20 AS SPOF	冲压，步冲关
DEFINE M23 AS SPOF	冲压，步冲关

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 或者 SONS 仅可以在各自的通道中激活。

如果 PONS 或者 SONS 同时在几个通道中激活，则马上会引起报警2200“通道%1在几个通道中不可以进行快 速冲压/步冲”。

否则功能 PONS 和 SONS 与功能 PON 和 SON一样。

行程控制式加速 PUNCHACC

语言指令 PUNCHACC(S_{min},A_{min}, S_{max}, A_{max}) 用来确定加速度特性曲线，该曲线视孔间距的情况定义不同的加速度 (A)。

举例 PUNCHACC(2, 50, 10, 100):

2毫米以下的孔距：

使用50%的最大加速度的加速度运行。

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%。

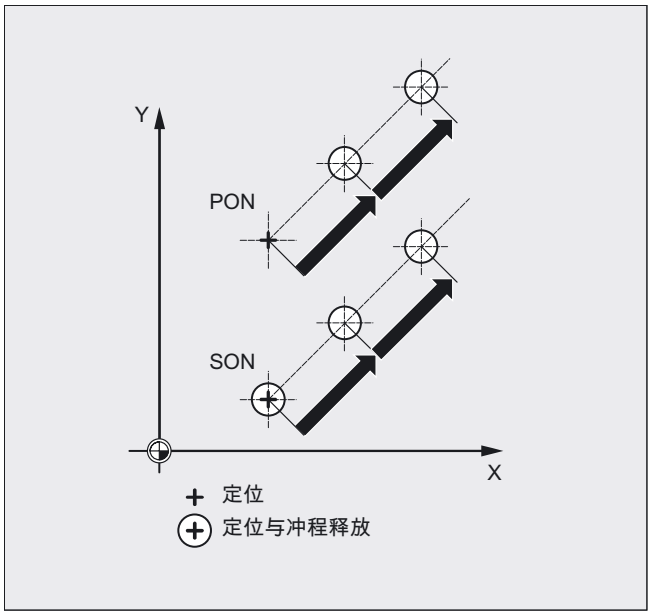
孔距大于10毫米：

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。

注意

用可旋转的刀具工作

为了可以在编程的轨迹处使用可旋转的刀具，请使用切线控制装置。

12.2 自动划分位移

功能

分度距离的细分

在有效的冲压时 或者步冲时，不仅SPP而且SPN会产生一个划分，使给轨迹轴编程的总的运行距离划分为一定量的相同长度的分度距离 (等距离位移划分)。在内部每个分度距离相当于一个程序段。

冲程数

在冲压时第一个冲程发生在第一个分度距离的终点处，相反，在步冲时则在第一个分度距离的起始处。对于总的运行位移，因此就有以下数量：

冲压冲程数 = 分度距离数

步冲冲程数 = 分度距离数+1

辅助功能

辅助功能在第一个所产生的程序段处执行。

编程

SPP=

或者

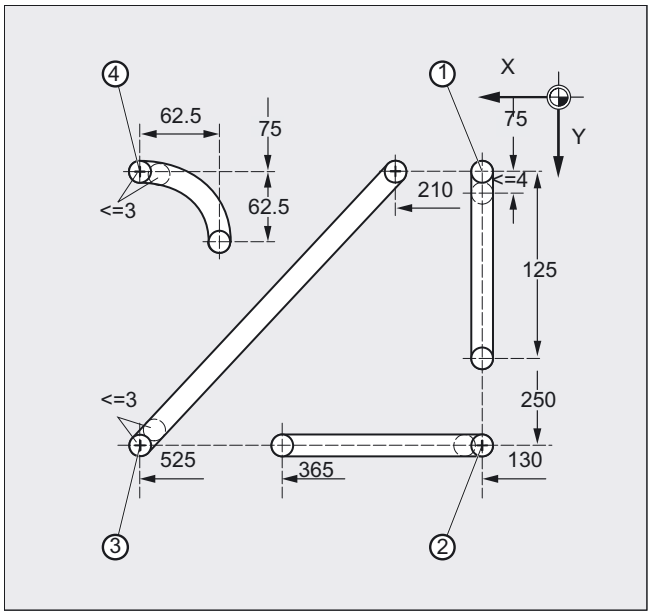
SPN=

参数

SPP	分度距离的大小 （最大冲程距离）； 模态方式有效
SPN	每个程序段分度距离的个数；程序段方式有效

举例1

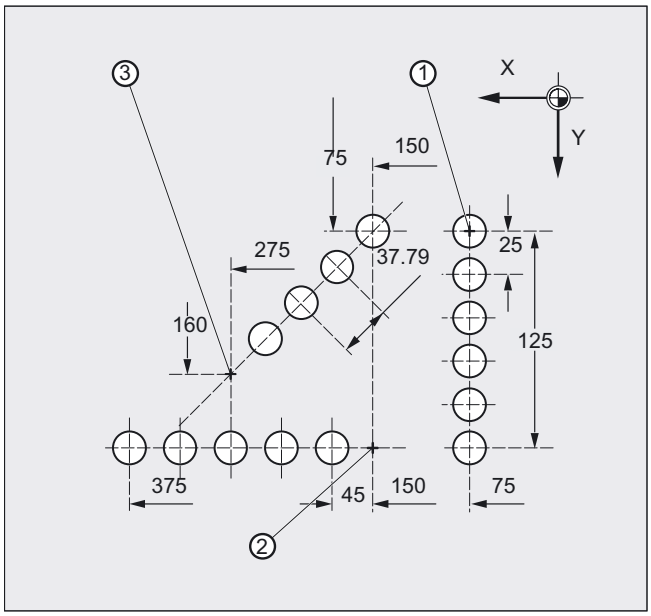
编程的步冲距离应该自动地分为相同大小的分度距离。




```
N100 G90 X130 Y75 F60 SPOF                ;定位到起始点1
N110 G91 Y125 SPP=4 SON                   ;步冲开；最大分度距离长度
                                          ;用于自动位移划分： 4 mm
N120 G90 Y250 SPOF                        ;步冲关；定位到
                                          ;起始点 2
N130 X365 SON                             ;步冲开；最大分度距离长度
                                          ;用于自动位移划分： 4 mm
N140 X525 SPOF                            ;步冲关；定位到
                                          ;起始点 3
N150 X210 Y75 SPP=3 SON                   ;步冲开；最大分度距离长度
                                          ;用于自动位移划分： 3 mm
N160 X525 SPOF                            ;步冲关；定位到
                                          ;起始点 4
N170 G02 X-62.5 Y62.5 I J62.5 SPP=3 SON   ;步冲开；最大分度距离长度
                                          ;用于自动位移划分： 3 mm
N180 G00 G90 Y300 SPOF                    ;步冲关
```

举例2

对于各个孔排列，应进行一次自动位移划分。划分时每次说明最大的分度距离长度（SPP值）。




```
N100 G90 X75 Y75 F60 PON          ; 定位到起始点1；
                                  ; 冲压开；冲压单个孔
N110 G91 Y125 SPP=25              ; 最大分度距离长度：
                                  ; 自动划分位移时： 25 mm
N120 G90 X150 SPOF                ; 冲压关；定位到
                                  ; 起始点2
N130 X375 SPP=45 PON              ; 冲压开；最大的分度距离长度
                                  ;用于自动位移划分： 45 mm
N140 X275 Y160 SPOF               ; 冲压关；定位到
                                  ; 起始点3
N150 X150 Y75 SPP=40 PON          ; 冲压开；不是使用编程的40毫米的分度距离长度，而
                                  是使用计算的分度距离长度
                                  ;37,79 mm
N160 G00 Y300 SPOF                ; 冲压关；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。

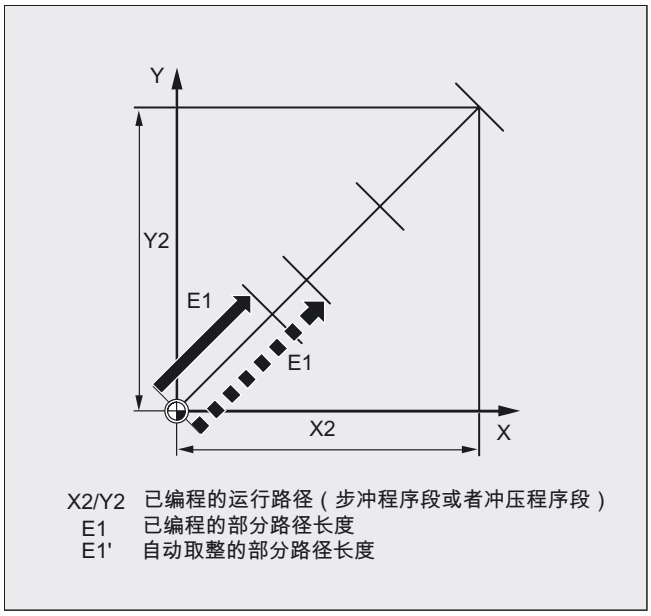
举例：

```
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)。

注意

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
```

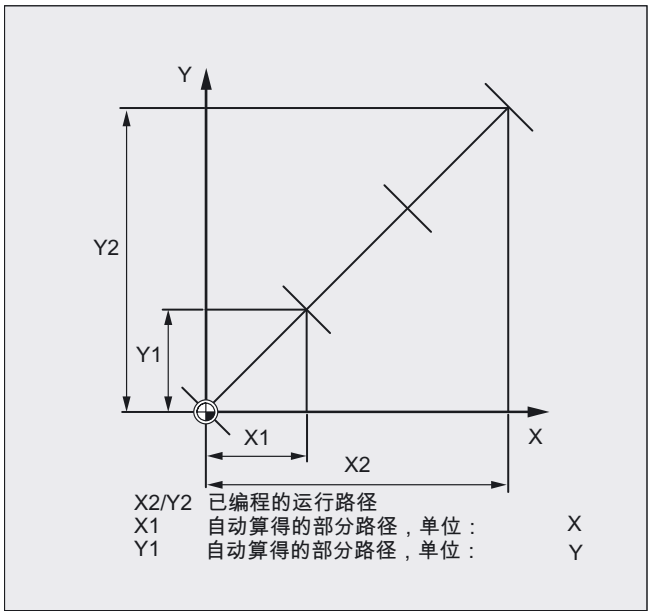
如果总的运行距离为15毫米，并且分度距离的一个长度为3.5毫米，则不会产生一个整数商(4.28)。因此降低SPP值，直至下一个可能的整数商。在这种情况下产生一个3毫米的分度距离长度。

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。



注意

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。 因此在一个除了单个轴运动 和SPP值之外没有编程轨迹轴的程序段中，SPP值被拒绝。

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. 缺省设定

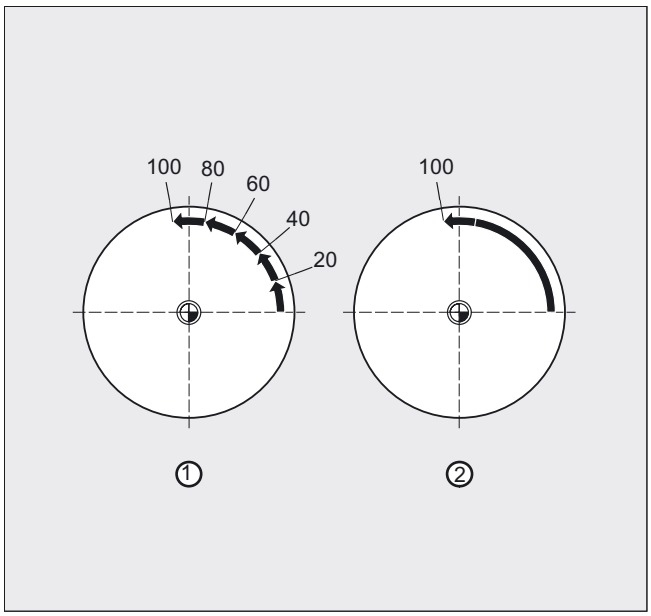
   单个轴的位移均匀地分布到用 SPP 产生的中间程序段中。

举例：

```
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毫米的总的运行距离中生成5个程序段。

因此A轴在每个程序段中转动20癮



1. 没有位移划分的单个轴

   单个轴在第一个所产生的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适用。

前提条件：单个轴作为冲压 - 步冲 - 轴定义。

其它功能 13

13.1 轴功能 (AXNAME, SPI, ISAXIS, AXSTRING)

功能

使用AXNAME，比如在编制通用的循环时，当轴名称并不知晓时（也可参见章节“字符串功能”）。

使用SPI，当轴功能用于一个主轴时，比如同步主轴。

在通用循环中使用ISAXIS，保证具有一个确定的几何轴，并且后面的$P_AXNX-调用不会被出错停止。

编程

AXNAME(平面轴)

或者

AX[AXNAME(String)]

或者

AXSTRING(SPI(n))

或者

SPI(n)

或者

ISAXIS(几何轴号)

参数

AXNAME	将某个输入字符串转换成轴标识符；输入字符串必须包含有效的轴名称。
SPI	将主轴编号转换为轴识别符；转换参数必须包含一个有效的主轴编号。
n	主轴号
AXSTRING	至软件版本SW5为止，主轴所分配的轴的轴变址作为主轴号输出。 自软件版本SW6起，该字符串连同所分配的主轴号输出。

AX	变量轴名称
ISAXIS	检查是否有所说明的几何轴。

SPI 扩展

轴功能 SPI(n) 现在也用于读写框架部件。
由此可以写框架，比如用句法$P_PFRAME[SPI(1),TR]=2.22 。 D通过地址AX[SPI(1)] = <轴位置> 附加编程轴位置，可以运行一个轴。

故障消除， 在AXSTRING(SPI(n))时

当使用 AXSTRING[SPI(n)] 进行编程时，已分配有主轴的轴的轴索引就不再作为主轴编号输出，而是输出字符串 "Sn" 。

举例：

AXSTRING(SPI(2)) 提供字符串 "S2"

举例

作为端面轴定义的轴应该运动。

```
OVRA[AXNAME("端面轴")]=10                    ; 端面轴
AX[AXNAME("端面轴")]=50.2                    ; 端面轴的终点位置
OVRA[SPI(1)]=70                              ; 主轴1的倍率
IF ISAXIS(1)==FALSE GOTOF WEITER             ; 有横坐标吗？
AX[$P_AXN1]=100                              ; 横坐标运行
继续:
```

13.2 功能调用 ISVAR ( ) 和读取带有或者没有 Array-Index 的机床数据

功能

ISVAR指令是一个具有NC语言含义的功能，带一个：

- BOOL类型的功能值
- 传送参数，类型STRING

当传送参数含有一个在NC中已知变量时，ISVAR-指令就会发送 TRUE（机床数据，调整数据，系统变量，常规变量如 GUD）。

编程

ISVAR(变量标识符)

或者

ISVAR (名称, [值, 值])

参数

变量名称	字符串类型的传送参数既可以没有尺寸，也可以单尺寸或者二维尺寸。
名称	名称，带一个NC知晓的变量，带或者不带行变址作为机床数据、设定数据、系统变量或者通用的变量。 **扩展：** 对于常规和特定通道的机床数据而言，即使在缺少索引的情况下，也会读取数组的第一个元素。
值	BOOL类型的功能值

机床数据和调整数据扩展，读取没有索引的数组

报警12400 "通道 % 1 程序段 % 2 字段 % 3 元素不存在" 在缺少索引的情况下**不再** 输出，从**常规和特定通道的** 机床数据中。

此外， **至少轴索引** 在**轴专用的** 机床数据中必须被编程。 否则将发出报警12400。

读取有和没有索引的机床数据数组举例

第一个元素被读取，当

```
R1=$MC_EXTERN_GCODE_RESET_VALUES
```

这相当于以前的

```
R1=$MC_EXTERN_GCODE_RESET_VALUES[0]
```

或者读取第一个元素

```
R1=$MA_POSTCTRL_GAIN[X1]
```

这相当于以前的

```
R1=$MA_POSTCTRL_GAIN[0, X1]
```

同步动作中的第一个元素也会被读取，当

```
WHEN TRUE DO $R1 = $MC_EXTERN_GCODE_RESET_VALUES
```

这相当于以前的

```
WHEN TRUE DO $R1 = $MC_EXTERN_GCODE_RESET_VALUES[0]
```

且之前没有使用报警12400 **读取过**

报警12400 会继续输出，当

```
R1=$MA_POSTCTRL_GAIN
```

功能调用举例 ISVAR

```
DEF INT VAR1
DEF BOOL IS_VAR=FALSE                    ;传送参数是 常规变量
N10 IS_VAR=ISVAR("VAR1")                 ; IS_VAR 在这种情况下是 TRUE(真)
DEF REAL VARARRAY[10,10]
DEF BOOL IS_VAR=FALSE                    ;不同的语法变量
N20 IS_VAR=ISVAR("VARARRAY[,]")          ;IS_VAR 为 TRUE,带有一个
                                         ;二维数组
N30 IS_VAR=ISVAR("VARARRAY")             ; IS_VAR 是TRUE(真), 变量存在
N40 IS_VAR=ISVAR
("VARARRAY[8,11]")                       ; IS_VAR 是 FALSE(假), 行变址不允许
N50 IS_VAR=ISVAR("VARARRAY[8,8")         ;IS_VAR 为 FALSE,
                                         的句法错误;缺少 "]"
N60 IS_VAR=ISVAR("VARARRAY[,8]")         ; IS_VAR 是 TRUE(真), 行变址允许
N70 IS_VAR=ISVAR("VARARRAY[8,]")         ; IS_VAR 是 TRUE(真)
DEF BOOL IS_VAR=FALSE                    ;传送参数为一个机床数据
N100 IS_VAR=ISVAR
("$MC_GCODE_RESET_VALUES[1]"             ; IS_VAR 是 TRUE(真)
DEF BOOL IS_VAR=FALSE                    ;传送参数为一个系统变量
N10 IS_VAR=ISVAR("$P_EP")                ; IS_VAR 在这种情况下是 TRUE(真)
N10 IS_VAR=ISVAR("$P_EP[X]")             ; IS_VAR 在这种情况下是 TRUE(真)
```

检查

根据传送参数进行以下的检查：

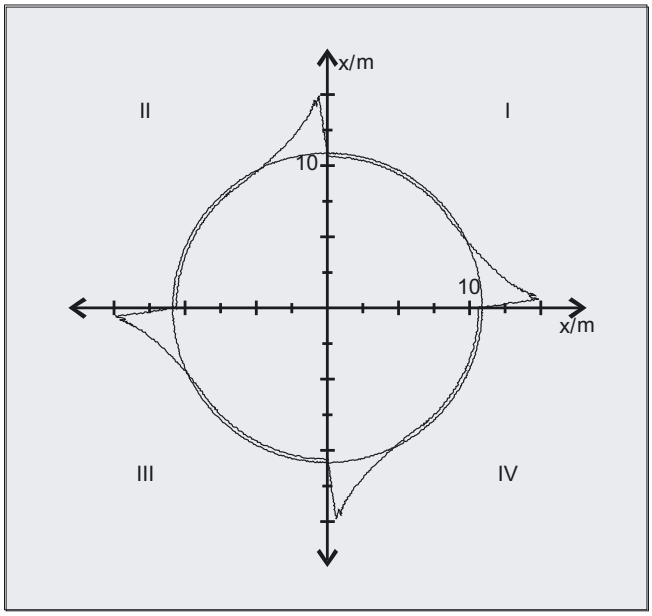
- 是否有名称
- 是否与一个尺寸或者二维的数组有关
- 是否允许一个行变址

只有当所有这些检查为正时，才送回TRUE(真)。如果仅有一个检查不能满足，或者仅出现一个语法错误，则它用FALSE(假)应答。轴向的变量作为轴名称的变址接收，但是没有进一步检查。

13.3 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。



编程

QECLRNON

或者

QECLRNOF

激活学习过程： QECLRNON

在NC程序中使用指令QECLRNON 在给定轴的情况下来激活实际学习过程：

QECLRNON (X1, Y1, Z1, Q)

只有当该指令有效时，才改变此特征曲线。

关闭学习： QECLRNOF

在所期望的轴结束学习运动之后，使用QECLRNOF同时结束所有轴的学习过程。

参数

QECLRNON (Achse.1,…4)	打开功能"学习象限缺陷补偿"
QECLRNOF	关闭功能"学习象限缺陷补偿"
QECLRN.SPF	学习循环
QECDAT.MPF	样本NC程序，用于系统变量的占用以及学习循环的参数设定。
QECTEST.MPF	样本NC程序，用于圆弧测试

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

13.4 同步主轴

功能

在同步运行方式中，有一个引导主轴 (LS) 和一个跟随主轴 (FS)，也就是所谓的 **同步主轴对**。跟随主轴在激活偶合的情况下（同步运行方式）根据所确定的函数关系来跟踪引导主轴的运动。

同步主轴副可以利用通道专用的机床数据固定设计，用于每个机床，或者通过CNC零件程序根据应用场合定义。每个NC通道可以同时运行最多2个同步主轴副。
可以从零件程序中

- 定义或者修改偶合。
- 启用
- 关闭
- 删除

。
除此之外，还可以

- 等候同步运行条件
- 修改程序段转换特性
- 选择要么是额定值偶合或者是实际值偶合的偶合方式
- 或者规定引导主轴和跟随主轴之间的角位移。

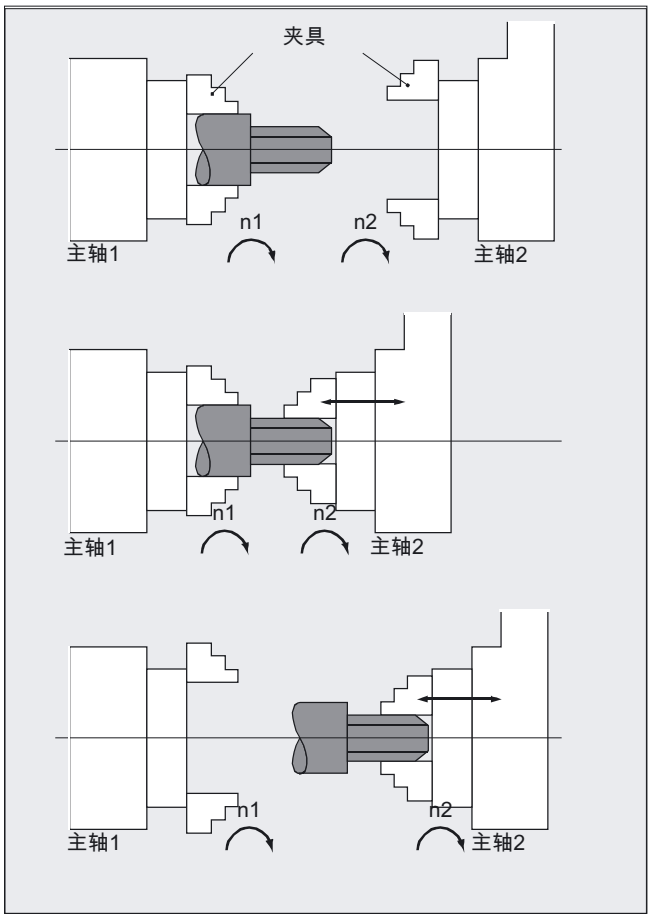
13.4.1 同步主轴 (COUPDEF, COUPDEL, COUPON, COUPOF, COUPRES)

功能

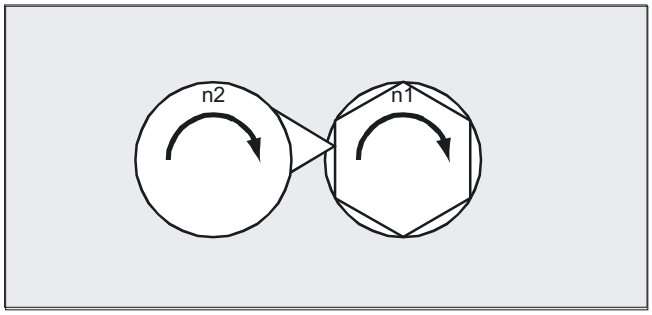
同步主轴功能够使车床在从主轴1向主轴2运动的过程中（例如用于最终加工），实现连续的工件传送，避免因重新装夹而导致的时间浪费。

工件传送可以进行：

- 转速同步度 ($n_{FS} = n_{LS}$)
- 位置同步度 ($\varphi_{FS} = \varphi_{LS}$)
- 位置同步度带角度偏移 ($\varphi_{FS} = \varphi_{LS} + \Delta\varphi$)



此外，可在主轴和某个“刀具主轴”之间设定一个传动比 $k_Ü$ ，使得可以进行多边加工（车削多边形）。



编程

COUPDEF(FS, LS, $Ü_{FS}$, $Ü_{LS}$, 程序段特性, 偶合)
COUPDEL(FS, LS)
COUPRES(FS, LS)
COUPON(FS, LS, PS_{FS})
COUPOF(FS, LS, POS_{FS} POS_{LS})
WAITC(FS, 程序段特性, LS, 程序段特性)

参数

COUPDEF	编制/修改用户定义的耦合
COUPON	打开耦合
COUPOF	关断耦合
COUPRES	复位耦合参数
COUPDEL	删除用户定义的耦合
WAITC	等待同步运行条件
FS, LS	跟随主轴和引导主轴的名称；用主轴号说明 例如S2
$Ü_{FS}$, $Ü_{LS}$	跟随主轴和引导主轴的传动参数
	预设置 = 1.0；分母优化说明
程序段特性：	程序段转换特性；进行程序段转换：
"NOC"	立即（预设置）
"FINE"	在"精同步运行"
"COARSE"	
"IPOSTOP"	在 "粗同步运行"
	当 IPOSTOP （即在额定值端的同步运行之后）
	程序段转换特性为模态有效
偶合	耦合方式：FS和LS之间的耦合
"DV"	额定值偶合（预设置）
"AV"	实际值耦合
	耦合方式模态有效。
PS_{FS}	在引导主轴和跟随主轴之间的角度偏差
POS_{FS}, POS_{LS}	跟随主轴和引导主轴的关断位置

使用引导主轴和跟随主轴进行加工的举例

```
                                        ;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/min,
                                        ;跟随主轴转速为 500/min
N10 COUPDEF (S2, S1, 1, 1, "NOC", "Dv") ;也可以设计偶合的定义
…
N70 SPCON                               ;调节引导主轴的位置(额定值补偿)
N75 SPCON(2)                            ; 跟随主轴接收到位置调节回路
N80 COUPON (S2, S1, 45)                 ;连续向着偏置位置 = 45 度的方向偶合
…
N200 FA [S2] = 100                      ; 定位速度=100度/分钟
N205 SPOS[2] = IC(-90)                  ;叠加90度沿着反方向运动
N210 WAITC(S2, "精")                    ; 等待同步运行"精"
N212 G1 X… Y… F…                        ; 加工
…
N215 SPOS[2] = IC(180)                  ;叠加180度沿着正方向运动
N220 G4 S50                             ;停留时间 = 主主轴旋转50转
N225 FA [S2] = 0                        ; 激活设计的速度(MD)
N230 SPOS[2] = IC (-7200)               ; 20转 使用设计 速度沿反方向
…
N350 COUPOF (S2, S1)                    ; 快速去除耦合, S=S2=3000
N355 SPOSA[2] = 0                       ; 在零度时停止FS
N360 G0 X0 Y0
N365 WAITS(2)                           ; 等待主轴2
N370 M5                                 ; 停止FS
N375 M30
```

确定同步主轴对

固定设计的耦合：

引导主轴和跟随主轴通过机床数据确定。 就这种偶合而言，NC零件程序不能修改为 LS 和 FS 规定的机床数据。 但仍然可以在NC零件程序中使用 COUPDEF 对偶合进行参数设定（前提条件： 确定没有写保护）。

用户定义的耦合：

使用语言指令 COUPDEF 可以在NC零件程序中重新建立和修改偶合。

当要定义某个新的偶合关系时，有可能必须事先使用 COUPDEL 将现有的某个用户自定义偶合删除。

COUPDEF定义新的耦合

在下面说明预定义的子程序参数设定：

COUPDEF(FS, LS, ÜFS, ÜLS, 程序段特性, 偶合)

跟随主轴和引导主轴，FS 和 LS

使用轴标识 FS 和 LS 可以明确地确定偶合。它们必须在每个COUP指令中编程。其它的耦合参数只有在要求改变时（模态有效）才进行编程。

举例：

N ... COUPDEF(S2,S1,ÜFS,ÜLS)

表示：

S2 = 跟随主轴，S1 = 引导主轴

跟随主轴的定位

当同步主轴耦合接通后，跟随主轴也与引导主轴引起的运动无关，在±180°。

SPOS定位

可以使用 SPOS = ... 对跟随主轴进行插补。 有关 SPOS 的更多信息可参阅编程说明书基本部分。

举例：

N30 SPOS[2]=IC(-90)

FA, ACC, OVRA:速度，加速度

使用 FA[SPI] (Sn)] 或者 FA[Sn], ACC[SPI(Sn)] 或者 ACC[Sn] 和 OVRA[SPI(n)]或者 OVRA[Sn] 可以对跟随主轴的定位速度和加速度值进行编程（参阅编程说明书基本部分）。 "n" 表示主轴编号 1...n。

可编程的程序段转换WAITC

使用 WAITC
l例如可在修改偶合参数或者定位过程之后继续执行程序时，确定具有不同的同步运行条件的程序段转换特性（例如粗，精密， IPOSTOP）。由此，延迟新程序段的换入，直至达到同步运行条件，从而可以更快地执行同步运行。如果没有说明同步运行条件，则给各个耦合编程/设计的程序段转换特性适用。

举例：

```
N200 WAITC
```

等待同步运行条件，用于所有有效的跟随主轴，没有说明同步运行条件。

```
N300 WAITC(S2,"FINE",S4,"COARSE")
```

等候为跟随轴 S2 和 S4所设定的同步运行条件“粗”。

传动比 $k_ü$

使用 FS 分子) 和 LS (分母) 的比值来设定传动比。

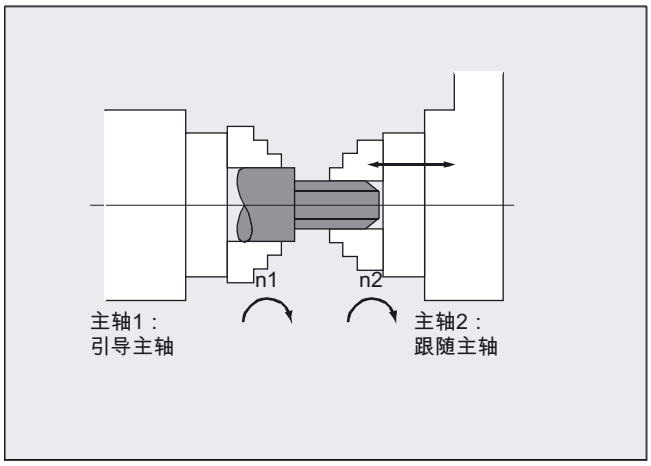
方法：

- 跟随主轴和引导主轴以相同的转速转动 ($n_{FS} = n_{LS}$; $k_ü$为正)
- ($k_ü$为负) LS 和 FS之间的通向或者反向运动
- 跟随主轴和引导装置以不同的转速转动
  ($n_{FS} = k_ü \bullet n_{LS}$;$k_ü \neq 1$)
  应用：多边形车削

举例：

```
N ... COUPDEF (S2,S1,1.0,4.0)
```

表示： 跟随主轴 S2 和引导主轴 S1 以 0.25 的传动比转动。



注意

至少必须对分子进行编程。如果没有说明分母，则分母设定为“1”。

在接通耦合期间也可以在运动中改变传动比。

程序段转换特性

因此在定义耦合时可以在下面的方法之间进行选择，何时进行程序段转换：

"NOC" 立即 (预设置)

"FINE" 当 "精密同步运行"时

"FIARSE" 当 "粗同步运行"时

"IPOSTOP" 当 IPOSTOP 时(即在额定值端的同步运行之后)

为了说明程序段转换特性，写粗写体字母就足够了。

耦合方式

方法：

"DV" FS 和 LS 之间的额定值偶合(预设置)

"AV" FS 和 LS 之间的实际值偶合

小心

耦合方式仅在关断耦合时才可以修改。

启动同步运行方式

- 以 LS 和 FS之间的任意角度关系尽可能迅速地启动偶合：

  ```
  N ... COUPON(S2,S1)
  ```

- 使用角位移 POS_{FS}启动

  用于成型工件的位置同步耦合。
  POSFS 以正旋转方向中引导主轴的 0°-位置为参照
  数值范围 POSFS: 0°… 359,999°:

  ```
  COUPON(S2,S1,30)
  ```

  以这种方式您也可以在有效的耦合时修改角度偏差。

关闭同步运行方式 COUPOF

下面的变量可能：

- 最大可能快速地关断耦合，程序段转换立即使能：

  ```
  COUPOF(S2,S1)
  ```

- 在超过关闭位置之后; 只有在超过关闭位置 POS_{FS} 且可能还有 POS_{LS} 之后才会释放程序段转换。
  值范围 0°… 359,999°:

  ```
  COUPOF(S2,S1,150)
  COUPOF(S2,S1,150,30)
  ```

使用停止跟随主轴来关闭某个偶合 COUPOFS

可以有两个变量：

- 尽可能快地关闭偶合并且在没有位置说明的情况下停止，立即释放程序段转换：
  COUPOFS(S2,S1)
- 在超过已编程的、以机床坐标系为参照的跟随轴关闭位置之后，只有在超过关闭位置 POS_{FS}之后才会释放程序段转换。
  值范围 0°… 359,999°:
  COUPOFS(S2,S1,POS_{FS})

删除偶合 COUPDEL，6.3以下软件版本的特性

当要定义某个新的偶合关系且所有可任意配置的偶合（1或者2）已确定时，必须删除某个现有的用户自定义同步主轴偶合。

N ... COUPON(S2,S1)

SPI(2) = 跟随主轴, SPI(1) = 引导主轴

注意事项

只有当偶合已取消时，才可删除 (COUPOF)。 6.3以下的软件版本不能使用 COUPDEL 来删除某个固定设计的偶合。

小心

删除偶合 COUPDEL，6.4以上版本的软件

在一个有效的同步主轴耦合中，现在也可以关断耦合，并删除耦合数据。
跟随主轴采用上一次的转速并且相当于 COUPOF(FS,LS) 目前为止的特性。

复位耦合参数 COUPRES

使用语言指令 "COUPRES"

- 激活保存在机床数据和调整数据中的参数（固定设计的偶合），
- 激活预设置（用户自定义偶合）。

同时，将丢失使用 COUPDEF 所编程的参数（包括传动比）。

N ... COUPRES(S2,S1)

S2 = 跟随主轴，S1 = 引导主轴

系统变量

跟随主轴当前的耦合状态

对于跟随主轴，可以读出在NC零件程序中带下面轴向系统变量的、当前的耦合状态：

```
$AA_COUP_ACT[FS]
```

FS = 带有主轴编号的跟随主轴的轴标识符，例如S2。

所读出的值对于跟随主轴有以下的含义：

0: 没有耦合有效

4: 同步主轴耦合有效

当前的角度偏差

可以在NC零件程序中使用下列轴向系统变量读入 `FS` 相对于 `LS` 的当前额定值端的位置偏移量：

```
$AA_COUP_OFFS[S2]
```

实际值一侧的位置偏差可以由以下参数读出：

```
$VA_COUP_OFFS[S2]
```

FS = 带有主轴编号的跟随主轴的轴标识符，例如 `S2`。

注意

在开启的耦合和跟随运行中取消调节器使能，并且在重新给予一个调节器使能之后，出现一个新的的位置偏置，它与原来编程的值不同。在这种情况下可以读出修改的位置偏差，该修改的位置偏差有时可以在NC零件程序中进行校准。

13.5 电子齿轮箱 (EG)

功能

借助“电子齿轮箱”功能可以根据取决于多达五个**引导轴** 的线性运动组对某个**跟随轴** 的运动进行控制。每个引导轴可以通过耦合系数定义引导轴和跟随轴之间的关系。

通过使单个引导轴运动分量乘以其耦合系数，由加法构成所计算的跟随轴运动分量。
在激活某个电子齿轮箱轴组合的情况下，可以向着某个定义位置启动跟随轴的同步动作。一个齿轮传动可以由零件程序：

- 定义，
- 接通，

- 关断，
- 删除。

跟随轴运动可以从以下值中导出（可选择）：

- 引导轴给定值，以及
- 引导轴的实际值。

作为扩展功能，也可以通过 **曲线图表** ((参见轨迹特性一章) 在引导轴和跟随轴之间实现非线性关系。
可以对电子齿轮箱进行级联，即某个电子齿轮箱的跟随轴可以是另一个电子齿轮箱的引导轴。

13.5.1 定义电子齿轮箱 (EGDEF)

功能

一个EG轴关联可以通过说明跟随轴，和至少一个、至多五个引导轴与其耦合类型确定。

EGDEF(跟随轴，引导轴1，偶合类型1，引导轴2，偶合类型2,..).

前提条件

定义一个EG轴关联的前提条件：
对于跟随轴而言，尚不允许有轴偶合已被定义（有可能必须事先使用 EGDEL 来删除现有的轴偶合）。

编程

```
EGDEF(C, B, 1, Z, 1, Y, 1)
```

B, Z, Y 通过给定值影响 C

所有引导轴的偶合类型不必相同，因此为每个引导轴单独设定偶合类型。在定义EG耦合关联时，用零预置耦合系数。

注意

EGDEF 触发进给停止。 带有 EGDEF 的电子齿轮箱定义也可不加修改地使用，当系统的一个或者多个引导 轴通过**曲线图表** 影响跟随轴时。

参数

EGDEF	定义一个电子齿轮箱
跟随轴	被引导轴影响的轴
引导轴1, ...引导轴5	影响跟随轴的轴
偶合类型1, ...偶合类型5	影响跟随轴的有： 0：实际值 1：给定值 影响。

13.5.2 启用电子齿轮箱 (EGON)

功能

有三种启用指令。

编程

型式1：

不使用同步选择来启用电子齿轮箱轴组合，使用
EGON(FA, "程序段转换模式", LA1, Z1 ,N1, LA2, Z2, N2, ..LA5, Z5,N5)

型式2：

使用同步选择来启用电子齿轮箱轴组合，使用
EGONSYN(FA, "Satzwechselmodus", SynPosFA,[, LAi, SynPosLAi, Zi, Ni])

型式3：

使用同步选择来启用电子齿轮箱轴组合。规定 **起动模式**，使用：
EGONSYNE(FA, "程序段转换模式", SynPosFA, 启动模式 [, LAi, SynPosLAi, Zi, Ni])

参数

型式1：

FA	跟随轴
程序段转换方式	可以使用以下的方式： "NOC" 立即执行程序段转换 "FINE" 当"精密同步运行"时执行程序段转换 "COARSE" 当"粗同步运行"时执行程序段转换 "IPOSTOP" 当额定值端同步运行时执行程序段转换
LA1, ... LA5	引导轴

Z1, ... Z5	分子，用于耦合系数i
N1, ... N5	分母，用于耦合系数i
	耦合系数 i = 分子 i / 分母 i

仅允许对事先已使用 EGDEF 指定的引导轴进行编程。 必须至少编程一个引导轴。

型式2：

FA	跟随轴
程序段转换方式	可以使用以下的方式： "NOC" 立即执行程序段转换 "FINE" 当"精密同步运行"时执行程序段转换 "COARSE" 当"粗同步运行"时执行程序段转换 "IPOSTOP" 当额定值端同步运行时执行程序段转换
[, LAi, SynPosLAi, Zi, Ni]	（不写方括号）
	最少1，最多5个跟随：
LA1, ... LA5	引导轴
SynPosLAi	用于第I个引导轴的同步位置
Z1, ... Z5	分子，用于耦合系数i
N1, ... N5	分母，用于耦合系数i
	耦合系数 i = 分子 i / 分母 i

仅允许对事先已使用 EGDEF 指定的引导轴进行编程。 通过为跟随轴 (SynPosFA) 和引导轴 (SynPosLA编程的"同步位置"来定义偶合组在其中作为*synchron* 的位置。如果电子齿轮在接通时不在同步状态，则跟随轴运行到其定义的同步位置。

型式3：

参数相当于型式2的参数**加上：**

起动模式	可以使用以下的方式： "NTGT"：经过优化时间后向下一个齿隙运动 "NTGP"：经过优化时间后向下一个齿隙运动 "ACN"：以绝对方式经过负旋转方向中的圆轴 "ACP"：以绝对方式经过正旋转方向中的圆轴 "DCT"：相对于已编程的同步位置优化时间 "DCP"：相对于已编程的同步位置优化行程

变量3仅对耦合到取模 - 引导轴的取模 - 跟随轴产生影响。以时间最优方式考虑跟随轴的速度极限。

说明

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箱的级联及其同步的**其它提示** 请在功能说明中查阅：
/FB2/ M3，联动，引导值偶合

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。

您可以在附录中找到电子齿轮的系统变量。这些变量通过以下列字符开始的名称进行标识：

```
$AA_EG_ ...
```

或者

```
$VA_EG_ ...
```

13.5.3 关闭电子齿轮箱 (EGOFS)

功能

对于关闭一个有效的EG轴关联，有以下三种方法：

编程

型式1：

`EGOFS(跟随轴)`	关断电子齿轮。跟随轴制动到停止。此调用删除进刀停止。

型式2：

`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。此调用删除进刀停止。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：

`EGOFC(跟随主轴1)`	关断电子齿轮。跟随主轴以关闭时刻有效的转速/速度继续运行。此调用删除进刀停止。

注意

该功能仅允许用于主轴。

删除某个电子齿轮箱的定义

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。

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。

13.5.4 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。

13.6 扩展的停止和退回

功能

功能“扩展的停止和退回”ESR(Extended Stopping and Retract)提供一种可能性，即对可选择的故障源可以进行灵活的、保护工件的反应。

可使用的部分反应

“扩展的停止和退回”可以使用以下的部分反应：

- **"扩展式停止"** (自主驱动) 是一种经过定义的、延时式停止。
- **"退回"** (自主驱动)
  表示从加工平面中“逃出”，进入一个安全的退回位置中。由此应该避免一个刀具和工件之间的冲撞危险。

- **"发电运行"** (自主驱动)
  如果中间电路的电力不足以满足安全的退回动作，在这种情况下就可以采用发电运行方式。在电网故障时，作为自身的驱动运行方式，可以使用一个必要的能量，用于驱动中间回路，它用于一个有秩序的“停止”和“退回”。

附加扩展功能

- **扩展式停止** (NC控制)
  是一种经过定义的延时停止动作，可保护轮廓不至于受损，该动作受NC控制。
- **退回"** (自主驱动)
  表示从加工平面中“逃出”，进入一个安全的退回位置中。这样可避免在刀具和工件之间的碰撞危险。例如，在加工齿轮时，可理解成是从刚刚加工好的齿槽中退出。

所有的反应可以相互无关地加以利用。

其它信息在/FB/ M3, 轴偶合和ESR中

可能有的触发源

启用“高级停止和退回”功能的故障源有如下几种：**一般源** (NC-外部/局部 或者 BAG-/特定通道):

- 数字输入（例如指向 NCU-组件以及接线板）或者控制过程内部的、可回读的数字输出的映象 $A_IN, $A_OUT
- 通道状态 $AC_STAT
- VDI-信号$A_DBB)
- 一些报警的成组信号 ($AC_ALARM_STAT)

轴向源

- 跟随轴的紧急退回极限 (电子偶合的同步运动, $VC_EG_SYNCDIFF[跟随轴])
- 驱动：中间电路报警极限（危险的欠压）， $AA_ESR_STAT[轴]
- 驱动：发电机最低转速极限（不再有可回馈的转动能量）， $AA_ESR_STAT[轴].

静止同步动作的逻辑关系：源/反应关系

使用静态同步动作的柔性逻辑方法，从而可以根据源触发一定的反应。

由用户借助静态同步动作将所有有关的源联系起来。用户可以作为整体逻辑联系源 - 系统变量，或者利用位掩码也可以选择计算和联系所要求的反应。静态同步动作可以在所有运行方式生效。

有关应用同步动作的详细描述可查阅

参考文献：/FBSY/ 同步动作功能描述。

激活

功能释放：

$AA_ESR_ENABLE
通过设置相应的控制信号 ($AA_ESR_ENABLE) 来启用发电机运行、停止、退回功能。该控制信号可以由同步动作修改。

功能触发（同时触发所有已被释放的轴）

$AN_ESR_TRIGGER

在识别出危险的中间回路欠压时，发生器运行“自动地”有效。

在识别出一个通讯故障时（在NC和驱动之间），以及在识别出运行时一个中间回路欠压时（前提条件配置和使能），激活驱动自给停止和/或退回。

此外，通过设置相应的控制信号 $AN_ESR_TRIGGER (向所有驱动主轴播发指令)，也可以从NC端来触发自主驱动的停止和/或者退回。

13.6.1 对 ESR 的自主驱动反应

功能

轴向定义自主驱动反应，即每个驱动装置在激活情况下自行处理其停止/退回请求没有给出停止时或者退回时轴的插补耦合或者轨迹定义的耦合，轴的基准由时间控制运行。

在执行驱动自给的反应期间和之后，其驱动不再受NC使能或者NC运行指令控制。要求一次关机再开机。报警 "26110:已触发自主驱动的停止/退回”是对此的提示。

参数

发生器运行

发生器运行

- 配置：通过 MD 37500: **10**
- 使能：系统变量 $AA_ESR_ENABLE
- 激活：取决于低于中间回路电压时驱动 - 机床数据的设定。

退回（驱动自给）

自主驱动式退回被

- 配置：通过 MD 37500:**11**; 时间设定和退回速度通过 MD 实现，参见“举例：驱动自给反应的使用”，在本章结束处。
- 使能：系统变量 $AA_ESR_ENABLE
- 释放：系统变量 $AN_ESR_TRIGGER.

停止（驱动自给）

驱动自给的停止

- 配置：通过 MD 37500:**12** 以及时间设定通过 MD;
- 释放 ($AA_ESR_ENABLE) 和
- 启动：系统变量 $AN_ESR_TRIGGER.

应用自主驱动式反应举例

配置举例

- 轴A应作为发生器驱动工作，
- 轴X在故障情况下以最大速度退回10毫米，并且
- 轴Y和Z延迟100毫秒停止，从而退回轴有时间去除机械耦合。

过程举例

1. 释放“高级停止和退回”与“交叉动作运行方式”选项功能（包括静态同步动作 IDS ...)”。
2. 功能分配：
   ```
   $MA_ESR_REACTION[X] = 11,
   $MA_ESR_REACTION[Y] = 12,
   $MA_ESR_REACTION[Z] = 12,
   $MA_ESR_REACTION[A] = 10;
   ```
3. 驱动配置：
   MD 1639:RETRACT_SPEED[X] = 400000H 在正方向中（最大速度），
   = FFC00000H 在负方向中，
   MD 1638:RETRACT_TIME[X] = 10ms (退回时间),
   MD 1637:GEN_STOP_DELAY[Y] = 100ms,
   MD 1637:GEN_STOP_DELAY[Z] = 100ms,
   MD 1635:GEN_AXIS_MIN_SPEED[A] = 发电机最低转速（转/分钟）.
4. 功能释放（由零件程序或者同步动作）：
   ```
   $AA_ESR_ENABLE[X] = 1,
   $AA_ESR_ENABLE[Y] = 1,
   $AA_ESR_ENABLE[Z] = 1,
   $AA_ESR_ENABLE[A] = 1.
   ```
5. 将发电机驱动主轴置于“推进”转速（例如在主轴运行过程中 M03 S1000)
6. 将触发条件表达成静态同步动作，例如：

- 取决于发电机轴的啮合：
  ```
  IDS = 01 WHENEVER $AA_ESR_STAT[A]>0 DO
  $AN_ESR_TRIGGER = 1
  ```

- 和/或者取决于随动运动方式所触发的报警 (位13=2000H):
```
IDS = 02 WHENEVER ($AC_ALARM_STAT_B AND 'H2000'>0
DO $AN_ESR_TRIGGER = 1
```
- 以及取决于 EG-同步运动监控 (例如当 Y 被定义成 EG-跟随轴且最大允许同步偏差为 100 µm 时) :
```
IDS = 03 WHENEVER ABS($VA_E_SYNCDIFF[Y])>0.1
DO $AN_ESR_TRIGGER = 1
```

13.6.2 NC控制的对退回的反应

功能

对于NC控制的退回而言，必须有一些初始条件，这些条件在下述前提之下有所描述。当这些退回的前提条件被满足时，就激活快速抬起动作。

在零件程序中，必须已经编程有退回位置 POLF。对于退回运动必须已经设置使能信号，并且保持设置。

编程

POLF[geo |mach]=,= 值	退回轴目标位置
POLFA(轴, 类型, 值)	单个轴退回位置
	允许以下的字母简式：
POLFA(轴, 类型)	字母简式，用于单个轴退回
POLFA(轴, 0/1/2)类型)	快速非激活或者激活
POLFA(轴, 0, $AA_POLFA[轴])	引起**一个** 进给停止
POLFA(轴, 0)	**不** 引起进给停止
POLFMASK(轴名称1，轴名称2, ...)	轴选择，用于退回
	轴，没有关联
POLFMLIN(轴名1, 轴名2, ...)	轴选择，用于退回
	轴，带线性关联

注意事项

如果在使用缩写型式 POLFA 时仅改变类型，则用户就必须保证退回位置或者退回行程含有一个有效的值。特别是在上电以后必须重新设置退回位置和退回位移。

参数

geo | mach	退回的几何轴或者通道/加工轴。
轴	有效单轴的轴标识符
类型	单轴的位置值，类型：
	位置值使无效
	位置值为绝对的
	位置值为增量的（距离）
值	退回位置，WKS适用于几何轴，MKS适用于其它轴。当几何轴和通道轴/加工轴有**相同的** 标识符时，就在工件坐标系中退回。
	允许增量式编程。
	有类型=1 的退回位置，用于单轴
	有类型=2 的退回行程，用于单轴
	也可使用类型 = 0来接受数值。然后才会将该值标记成无效，且必须重新编程退回。
POLF	指令POLF模态有效。
POLFA	如果一个轴不是单个轴，或者缺少类型或类型 = 0，则会发出相应的报警26080和报警26081。
POLFMASK,	使用指令**POLFMASK** 来释放已指定的退回轴 - 不使用轴之间的关系。
	指令**POLFMASK()** 在没有指定某个轴的情况下，将不使用轴之间的关系所退回的所有轴的快速抬起动作取消。
POLFMLIN,	使用指令**POLFMLIN** 来释放已指定的退回轴 - 使用轴之间的线性关系。
	指令**POLFMLIN()** 在没有指定某个轴的情况下，将使用线性关系被退回的所有轴的快速抬起动作取消。
轴名称 i	轴名称，这些轴在LIFTFAST时应该运行到用POLF定义的位置。所有说明的轴必须处于同一个坐标系中。在通过**POLFMASK** 或者 **POLFMLIN**可以向某个固定位置释放快速抬起动作之前，必须使用**POLF** 给选定的轴编程一个位置。没有用来预设置**POLF** 值的机床数据。
	当解释**POLFMASK** 或者 **POLFMLIN** 时，如果**POLF**尚未编程，就会发出报警16016。

注意

当使用 POLFMASK, POLFMLIN 或者 POLFMLIN, POLFMASK 将轴依次释放时，相应轴的上一次的设置有效。

小心

使用 POLF 编程的位置和通过 POLFMASK 或者 POLFMLIN 编程的激活指令在零件程序开始执行时被删除。这就是说，用户必须在每个零件程序中对 POLF 的值和在 POLFMASK 或者POLFMLIN 中所选择的轴重新编程。

有关坐标系的修改、模量圆轴的作用等等的其它提示可在功能说明中查阅 /FB/ M3, 轴偶合和 ESR

某个单轴的退回举例

```
MD 37500:ESR_REACTION[AX1] = 21                ; NC控制的退回
...
$AA_ESR_ENABLE[AX1] = 1
POLFA(AX1,1, 20.0)                             ;轴向退回位置
                                               ;20.0 (绝对值) 被分配给AX1。
$AA_ESR_TRIGGER[AX1] = 1                       ; 从这里开始退回运动。
```

前提条件

退回

- 使用 POLFMASK 或者 POLFMLIN 所选中的轴，
- 使用 POLF 所定义的、特定轴的位置，
- 使用 POLFA 所定义的某个单轴的退回位置，
- 时帧，在
  MD 21380:ESR_DELAY_TIME1 和
  MD 21381:ESR_DELAY_TIME2,
- 通过系统变量 $AC_ESR_TRIGGER
  $AA_ESR_TRIGGER 触发，用于单轴，
- 已约定的 ESR
  MD 37500:ESR_REACTION = 21,
- LFPOS 来自于模态 46. G代码组。

释放和起动NC控制的退回

当系统变量 $AC_ESR_TRIGGER = 1 被设定，且当在该通道中某个退回轴已被设置（即 MD 37500:ESR_REACTION = **21**） 并且已设定给该轴 $AA_ESR_ENABLE = 1 ，那么就会在这 个通道中**LIFTFAST** 被激活。

在零件程序中，必须已经编程有退回位置 POLF 。在使用 POLFA(轴，类型，值) 退回单轴时，值必须已经编程并且遵守下列条件：

- $AA_ESR_ENABLE = 1 已设定。
- POLFA(轴) 在触发时刻必须是一个单轴。
- POLFA(类型) 要么是 轴名称=1 或者 轴名称=2

对于退回运动必须已经设置使能信号，并且保持设置。

- 使用 LFPOS, POLF 配置的、用 POLFMASK 或者 POLFMLIN 所选择的轴的抬起运动用来替代在零件程序中为这些轴所规定的轨迹运动。
- 高级退回运动（即通过 $AC_ESR_TRIGGER 所触发的 LIFTFAST/LFPOS）**无法中断**且只能通过 NOTAUS（紧急停机）提前结束。

可用于退回的最大时间之和 MD 21380:ESR_DELAY_TIME1 和 MD 21381:ESR_DELAY_TIME2 。在该段时间结束之后，对于退回轴也引入快速制动，接着跟随运行。

在激活快速退刀时刻考虑有效的框架。

注意

带有旋转的框架也会通过 POLF 来影响抬起的方向。NC控制的退回

- 配置：通过 MD 37500:**21** 以及 2 时间设定通过 MD 见上，
- 释放 ($AA_ESR_ENABLE) 和
- 启动：系统变量 $AC_ESR_TRIGGER 用于带有 $AA_ESR_TRIGGER的单轴。

13.6.3 NC控制的对停止的反应

功能

停止

扩展停止的执行（NC控制）通过两个机床数据

MD 21380:ESR_DELAY_TIME1 和
MD 21381:ESR_DELAY_TIME2 说明。

轴不受干扰地继续插补在MD21380中的时间长度，如同编程一样。在MD21380中时间段结束之后，引入插补控制的制动（斜坡停止）。对于插补控制的制动，在此可以使用MD21381中的时间段，在此时间段之后引入带跟随运行的快速制动。

释放和起动NC控制的停止

NC控制的停止

配置：通过 MD 37500:**22** 以及 2 时间设定通过两个 MD 见上;

使能 ($AA_ESR_ENABLE) 并且

启动：系统变量 $AC_ESR_TRIGGER，用于带有 $AA_ESR_TRIGGER 的单轴。

某个单轴的停止举例

```
MD 37500:ESR_REACTION[AX1] = 22                    ; NC控制的停止
MD 21380:ESR_DELAY_TIME1[AX1] = 0.3
MD 21381:ESR_DELAY_TIME2[AX1] = 0.06
...
$AA_ESR_ENABLE[AX1] = 1
$AA_ESR_TRIGGER[AX1] = 1                           ; 从这里开始停止。
```

13.6.4 发电机运行/中间电路支持

功能

通过对静态同步动作($AA_ESR_ENABLE) 设计驱动MD和相应的编程，可以短时间地补偿中间回路电压跌落。消除的时间取决于发生器存储的能量，它用于支持中间回路，并取决于保持当前运动所需要的能量（中间回路辅助和监控发生器转速极限）。

在低于中间电路电压下限时，所涉及的轴/主轴就会从位置或者转速控制式运行方式过渡到发电机运行方式中。通过制动驱动（给定转速额定值 = 0）能量反馈到中间回路。

有关信息可参阅功能说明 /FB/ M3, 联动，引导值偶合。

13.6.5 驱动自给停止

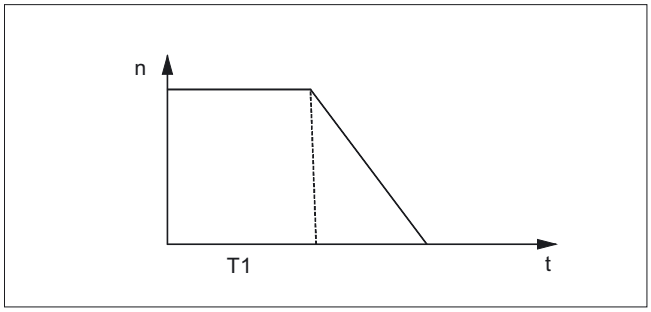
功能

一个事先耦合的关联驱动可以通过时间控制的关断延迟，以尽可能小的偏差相继停止，如果这在控制系统一侧不能实现。

驱动自给的停止通过MD进行配置和使能（在MD中的延迟时间T1），并且通过系统变量$AA_ESR_ENABLE使能，通过$AN_ESR_TRIGGER启动。

反应

在故障情况下正有效的转速给定值继续输出，时间T1。以此来尝试在断电之前维持有效的运动，直到强制连接被取消或者同时在其它驱动装置中已开始的退回运动已结束时为止。这对于所有的引导驱动/跟随驱动，或者对于耦合中或关联中的驱动具有意义。



在时间T1之后，所有轴以转速给定值零 - 接通在电流极限处停止，并且在到达停止状态或在时间结束之后（ + 驱动MD）删除脉冲。

13.6.6 驱动自给退回

功能

带有数字驱动装置 SIMODRIVE 611数字型 的轴可以（如果有此设计且已释放）

- 即使在控制系统失灵时（生命标志丢失识别）
- 当中间电路电压下降到警告极限以下时，
- 在通过系统变量$AN_ESR_TRIGGER触发时

自己执行一个退回运动。退回运动以自主方式通过驱动装置SIMODRIVE 611数字型实现。自退回阶段开始时驱动就使其使能保持到先前有效的值。

其它信息参阅 /FB/ M3, 联动，引导值偶合。

13.7 链接通讯

功能

在一台设备中各个NCU单元之间的NCU链接，用于带分散式系统结构的设备中。当对轴和通道的需求量较大时，例如旋转式连续自动工作机床和多主轴机床，则使用唯一一个NCU时的计算性能、配置方法和内存可能会碰到极限。

使用一个 NCU 链接模块联网的多个 NCU 可提供一种开放式解决方案，可满足此类机床的所有加工任务。NCU-链接模块 (HW) 可通过对系统变量进行读写访问的方式来实现快速的 NCU-NCU-通讯。

前提条件

这种功能已经与需要订购的选项功能相关联。

链接变量

链接变量是**系统全局参数**, 这些参数可以被联网的NCU作为**系统变量** 来访问。

通过用户（在这种情况下通常是机床制造商）来设定：

- 这些变量的**内容** ，
- 其 **数据类型**,
- 其 **应用**,
- 其在联网存储器中的位置**(访问索引)** 。

链接变量的应用

- 全局机床状态
- 工件夹具 打开/闭合
- 等等

访问应用的时间特性

各个NCU同时 **在某个时刻**对联网存储器进行访问的应用必须**同一** 利用联网存储器。对于在时间上完全去除耦合的过程，该链接存储器可以不同地进行占用。

警告

只有当其它的NCUs也出现所写入的信息时，才结束一个链接变量的写过程。为此，需要大约两个插补节拍。为了保持稠度，局部地写入到链接存储器中延迟相同的时间。

其它提示可参阅功能说明 /FB/ B3.多个操作面板和 NCU。

13.7.1 访问某个NCU全局存储器

功能

通过链接模块联网的多个NCU可以借助下述系统变量的访问来对某个NCU全局存储器进行读写访问。

- 每个通过链接模块联网的NCU可以使用所有联网NCU可统一访问的**全局链接变量** 。
- 链接变量可作为系统变量编程。该变量的含义通常通过机床制造商确定和说明。
- 链接变量的应用
- 文件卷相对来说较小
- 传输速度非常快，因此：这种使用为时间紧迫的信息设置
- 可以从**零件程序** 和**同步动作** 中对这些系统变量进行访问。可以对 NCU-全局系统变量的存储器大小进行设计。

在结束某个插补节拍之后，所有参与工作的NCU均可以一致读取某个全局系统变量中新写入的值。

参数

链接变量保存在联网存储器中。在引导之后链接存储器用0初始化。

在联网存储器范围内可以访问下列链接变量：

- INT `$A_DLB[i]` ;数据字节 (8 位)
- INT `$A_DLW[i]` ;数据字 (16 位)
- INT `$A_DLD[i]` ;双数据字 (32 位)
- REAL `$A_DLR[i]` ;实时数据 (64 位)

在读写链接变量时，根据相应类型触发1、2、4、8字节。

索引i 表示相应变量的开始处，与所设计的联网存储器的开始处有关。索引自0开始计数。

数值范围

下面的值范围与数据类型相联系：

BYTE:0 到 255

WORD:-32768 到 32767

DWORD:-2147483648 到 2147483647

REAL:-4.19e-308 到 4.19e-307

举例

$A_DLB[5]=21	在公共链接存储器中第5个字节包含值21。

13.8 轴容器 (AXCTWE, AXCTWED)

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。轴容器将所有装夹轴/主轴的连接可能性汇集在一起，从中始终只有**一个** 为加工单元而**激活**。

编程

在轴容器中登记时推移步距n可以用指令：

AXCTSWE(CT_i)	AXIS CONTAINER SWITCH ENABLE
AXCTSWED(CT_i)	AXIS CONTAINER SWITCH ENABLE DIRECT

参数

AXCTSWE	从每个通道释放容器中已输入的轴来旋转容器。
AXCTSWED	在已激活轴的单独作用下将轴容器按照所保存的步距旋转。当其余容器中的轴所具有的通道处于RESET状态时，就释放容器中所输入的轴。
CT_i	轴容器号，其内容应进行推移，或者
或者	通过MD设置的轴容器的各个名称。
例如 A_CONT1	

轴容器

通过轴容器可以分配：

- 局部轴和/或者
- 链接轴（参见基础部分）

通过轴容器定义的、可以使用的轴，在转换时通过推移到轴容器的登记来进行。

可通过**零件程序** 来触发位移。

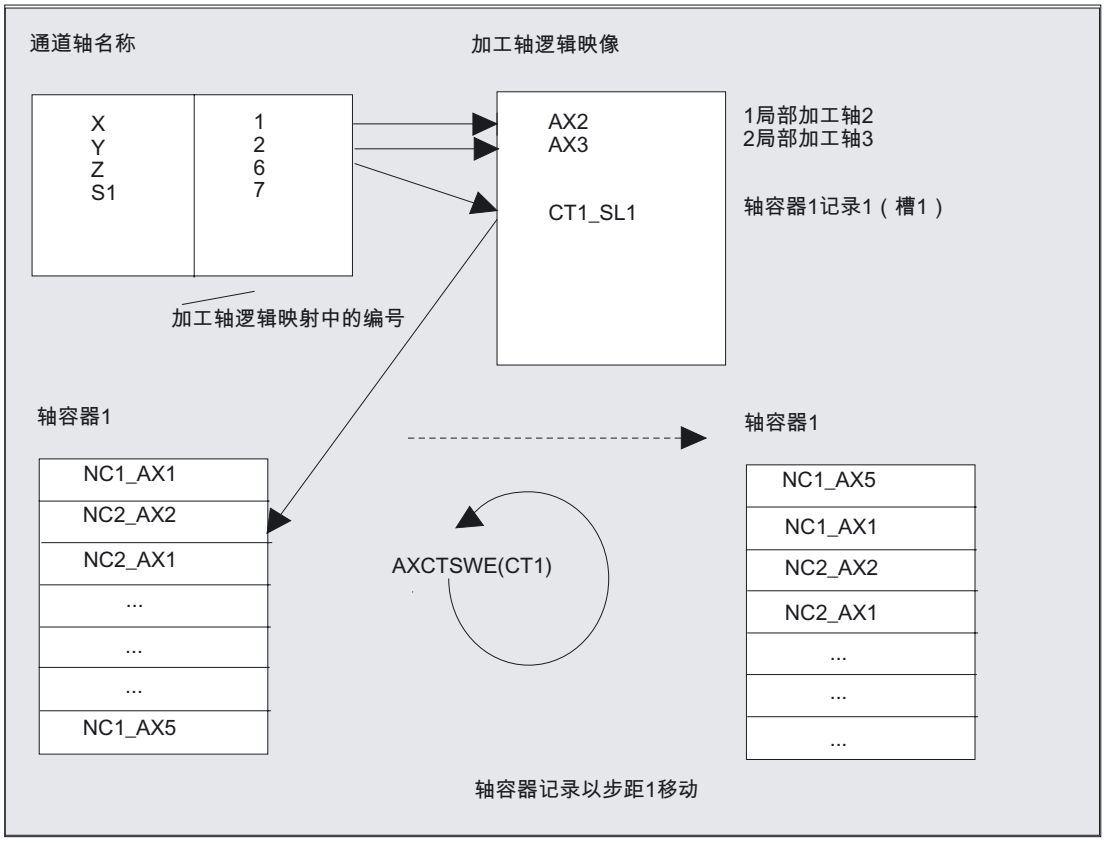
带链接轴的轴容器是一个NCU可以搭接的运行工具（NCU全局）,它通过控制系统进行协调。只管理局部轴的轴容器是可以的。

有关设计轴容器的详细提示可参阅功能说明 /FB/ B3, 多个操作面板和 NCU，“轴容器”

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，就会以保存在SD中的步距来旋转轴。



在轴容器旋转1之后，在前面的举例中通道轴Z代替轴AX1分配到NCU1，轴AX5分配到NCU1。

AXCTSWED ()

指令类型 AXCTSWED(CT_i)
可以用来简化起动。轴容器在有效通道单独作用下旋转SD中所存储的步距。只有在容器中拥有轴的的其余通道处于**RESET**-状态时，才能使用该调用。

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。

13.9 程序执行时间/工件计数器

13.9.1 概述

为了对机床操作者给予帮助，有关于程序执行时间的信息可供使用。

这些信息在相应的机床数据中进行规定，并且可以作为NC或者PLC程序中的系统变量进行处理。这些信息也可工操作面板证明接口中的HMI/MMC使用。

13.9.2 程序运行时间

功能

在程序执行时间功能项下，定时器被作为可以用来监控工艺过程的系统变量。

对于该定时器仅有读存取存在。HMI/MMC 可以随时对这些定时器进行读取访问。

参数

下面两个定时器作为NCK专用的系统变量定义，并且一直有效。

系统变量

$AN_SETUP_TIME	从最后的设置开始的时间，单位分钟。 在设置时复位。
$AN_POWERON_TIME	从最后的PowerOn（上电）开始的时间，单位分钟。 在PowerOn(上电）时复位。

下面的三个定时器作为通道专用的系统变量定义，并且可以通过机床数据激活。

$AC_OPERATING_TIME	在自动方式时NC程序的总的运行时间，单位秒。
$AC_CYCLE_TIME	所选择的NC程序的运行时间，单位秒。
$AC_CUTTING_TIME	刀具啮合时间，单位秒。
$MC_RUNTIMER_MODE	刀具啮合时间，单位秒。

注意

所有的定时器在系统引导时均用缺省值归零，并且可以与其激活无关而读出。

举例

```
1. 激活已激活NC程序的执行时间测定功能，同时不在
激活试运行进给和程序测试时进行测量：
$MC_PROCESSTIMER_MODE = 'H2'
2. 激活刀具作用时间的测定功能，同时也在
激活试运行进给和程序测试时进行测量：
$MC_PROCESSTIMER_MODE = 'H34'
3. 激活全部执行时间和刀具作用时间的测定功能，
同时也在程序测试时进行测量：
$MC_PROCESSTIMER_MODE = 'H25'
```

13.9.3 工件计数器

功能

使用“工件计数器”功能可提供计数器，例如可以用来在控制系统内部对工件进行计数。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。

通过机床数据可以对计数器激活、归零时刻和计数算法产生影响。

参数

准备以下的计数器：

系统变量

$AC_REQUIRED_PARTS	所需工件的个数（工件给定值）
	在此计数器中可以定义工件的个数，在到达这个数值之后，实际工件的个数$AC_ACTUAL_PARTS归零。通过一个MD可以激活生成显示报警“工件额定值已达到”和通道VDI信号“工件额定值已达到”。
$AC_TOTAL_PARTS	全部已生产工件的数量（总实际值）
	计数器给出所有自开始时刻起所生产的工件数量。该计数器仅在系统引导时自动地用缺省值归零。
$AC_ACTUAL_PARTS	当前工件的数量（当前实际值）
	在这种计数器中记录自开始时刻起所生产的所有工件数量。当达到工件额定数时($AC_REQUIRED_PARTS)，计数器就会自动归零 ($AC_REQUIRED_PARTS 不等于 0 是前提条件)。
$AC_SPECIAL_PARTS	用户指定工件的数量
	该计数器允许用户根据自定义来对工件计数。在与$AC_REQUIRED_PARTS (工件给定值)一致时可以定义一个报警输出。用户必须自行将该计数器归零。

注意

“工件计数器”功能与刀具管理器的功能无关。所有计数器均可以从 HMI/MMC 读取或者写入。

所有的计数器在系统引导时均用缺省值归零，并且可以与其激活无关而读写。

举例

激活工件计数器 $AC_REQUIRED_PARTS:	
$MC_PART_COUNTER='H3'	$AC_REQUIRED_PARTS 已激活，当 $AC_REQUIRED_PARTS == $AC_SPECIAL_PARTS 时显示报警
激活工件计数器 $AC_TOTAL_PARTS:	
$MC_PART_COUNTER='H10' $MC_PART_COUNTER_MCODE[0]=80	$AC_TOTAL_PARTS 有效，使用M02 可以使计数器增加1， $MC_PART_COUNTER_MCODE[0] 没有含义。
激活工件计数器 $AC_ACTUAL_PARTS:	
$MC_PART_COUNTER='H300' $MC_PART_COUNTER_MCODE[1]=17	$AC_TOTAL_PARTS 有效，使用每个M17可以使计数器增加值1。
激活工件计数器 $AC_SPECIAL_PARTS:	
$MC_PART_COUNTER='H3000' $MC_PART_COUNTER_MCODE[2]=77	$AC_SPECIAL_PARTS 有效，使用每个M77可以使计数器增加值1。
关闭工件计数器 $AC_ACTUAL_PARTS:	
$MC_PART_COUNTER='H200' $MC_PART_COUNTER_MCODE[1]=50	$AC_TOTAL_PARTS 无效，其余没有意义。
激活所有计数器举例 1～4：	
$MC_PART_COUNTER = 'H3313' $MC_PART_COUNTER_MCODE[0] = 80 $MC_PART_COUNTER_MCODE[1] = 17 $MC_PART_COUNTER_MCODE[2] = 77	$AC_REQUIRED_PARTS 已激活 当$AC_REQUIRED_PARTS == $AC_SPECIAL_PARTS时显示报警 $AC_TOTAL_PARTS 已激活，每次使用 M02 可将计数器读数提高1 $MC_PART_COUNTER_MCODE[0] 没有含义 $AC_ACTUAL_PARTS 已激活，每次使用 M17 可将计数器读数提高1 $AC_SPECIAL_PARTS 已激活，每次使用 M77 可将计数器读数提高1

13.10 交互式调用零件程序指令 (MMC) 的窗口

功能

通过MMC指令可以在HMI/MMC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/MMC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。

编程

```
MMC(CYCLES, PICTURE_ON, T_SK.COM, BILD, MGUD.DEF, BILD_3.AWB,
TEST_1, A1", "S")
```

有关MMC指令编程的详细提示（包括编程示例）可根据所使用的HMI/MMC软件在 /IAM/ 中的手册AE1, BE1, HE1, IM2, IM4 和 IM5 中查阅。

参数

MMC	在 HMI/MMC 上从零件程序中调用交互式对话窗口。
CYCLES	操作区，在此执行所设计的用户会话。
PICTURE_ON bzw. 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

13.11 对运动控制的影响

13.11.1 百分比式急冲修正 (JERKLIM)

功能

在条件极为不利的程序段执行过程中，可能有必要将急冲限制在最大可能值以下，以便例如减小机床的负荷。加速模式SOFT必须已激活。该功能仅对轨迹轴有影响。

编程

JERKLIM[轴] =...

参数

JERKLIM	按照百分比修改所允许的最大冲击，它与机床数据中对该轴的设定值有关。
轴	应对其急冲极限值进行调整的加工轴
值范围： 1 ... 200	100 表示：对急冲没有影响。 100 在 RESET 和零件程序开始之后有效。

举例

在自动方式下，对于编程的轴，冲击极限值限制到机床数据中所存储冲击极限值的百分比值。

N60 JERKLIM[X] = 75

表示：轴溜板在X方向应以75%轴的允许冲击值进行加速/延迟。

注意

另一个示例可在“百分比式速度修正 (VELOLIM)”一章中查阅。

13.11.2 百分比式速度修正 (VELOLIM)

功能

在条件极为不利的程序段执行过程中，可能有必要将速度限制在最大可能值以下，以便例如减小机床的负荷或者改善加工品质。该功能仅对轨迹轴和定位轴有作用。

编程

VELOLIM[轴] = ...

参数

VELOLIM	按照百分比修改所允许的最大速度，它与机床数据中对该轴的设定值有关。
轴	应对其速度极限值进行调整的加工轴
值范围： 1 ... 100	100 表示：对速度没有影响。 100 在 RESET 和零件程序开始之后有效。

举例 VELOLIM

在自动方式下，对于编程的轴，速度极限值限制到机床数据中所存储速度极限值的百分比值。

```
N70 VELOLIM[X] = 80
```

表示：轴溜板在X方向应以80%轴的允许速度值进行运行。

举例 VELOLIM 和 JERKLIM

```
N1000 G0 X0 Y0 F10000 SOFT G64
N1100 G1 X20 RNDM = 5 ACC[X] = 20
ACC[Y] = 30
N1200 G1 Y20 VELOLIM[X] = 5
JERKLIM[Y] = 200
N1300 G1 X0 JERKLIM[X] = 2
N1400 G1 Y0
M30
```

13.12 主/从组合 (MASLDEF, MASLDEL, MASLOF, 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主动轴上。

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。

编程

MASLON(Slv1, Slv2, ...,)	
MASLOF(Slv1, Slv2, ...,)	
MASLDEF(Slv1, Slv2, ..., 主动轴)	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, ...,)	用于从动轴的扩展功能

注意

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。

参数

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	分开主/从组合并且删除组合的定义。
Slv1, Slv2, ...	受某个主动轴引导的从动轴。
主动轴	对某个主/从组合中所定义的从动轴进行引导的轴。

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，重要的轴应该成为主动轴。

```
MASLDEF(AUX,S3)          ;S3主动，用于AUX
MASLON(AUX)              耦合开，用于AUX
M3=3 S3=4000             ;转动方向向右
```

```
MASLDEL(AUX)                                ;删除设计并且
                                            ;断开偶合
AXCTSWE(CT1)                                ；容器旋转
```

某个从动轴的实际值偶合举例

通过 PRESETON将某个从动轴的实际值偶合设置成主动轴的相同值。

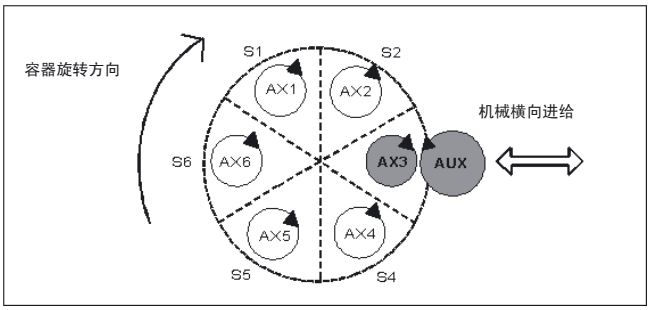
如果是某个永久性的主/从偶合，应当在从动轴上通过 PRESETON 来改变实际值。

```
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,     ;给尚未经过找零的从动轴设定实际值，
U1, 0)                                      ;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
```

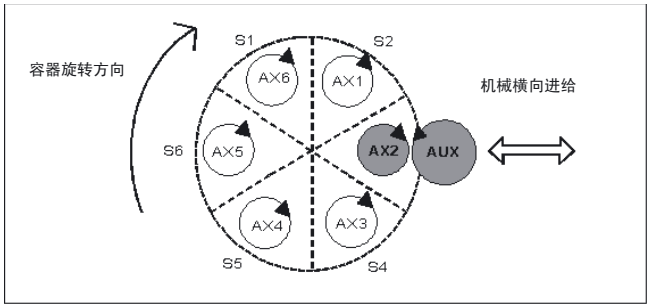
举例 偶合顺序 位置 3/容器 CT1

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：



参见 /FB/ B3, 多个操作面板和NCU，“轴容器”一章

说明

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	该指令用来取消将从动轴分配给主动轴，并且同时断开偶合，与 MASLOF 相同。已经在机床数据中约定的主/从定义保持不变。
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。

注意

对于从动轴而言，可通过 PRESETON 使实际值同步到与主动轴的值相同。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主动轴的值。然后该持续的耦合再次恢复。

持久的主/从偶合使用MD 37262:MS_COUPLING_ALWAYS_ACTIVE = 1 激活并且对临时偶合的语言指令没有影响。

6.5以上软件版本主轴的偶合特性

如果主轴在转速控制的运行方式中，则 MASLON, MASLOF, MASLOFS 和 MASLDEL 的偶合特性通过MD 37263:MS_SPIND_COUPLING_MODE 来确定。

在 MD 37263 = 0 的默认设置中，仅在参与轴的停止状态中进行从动轴的偶合和脱离。MASLOFS 相当于 MASLOF.

当 MD 37263 = 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。

自有切割程序

14

14.1 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。

编程

CONTPRON

或者

CONTDCON

或者

INTERSEC

或者

EXECTAB

或者

CALCDAT

参数

CONTPRON	打开以表格形式整理的轮廓（11栏）
CONTDCON	打开以表格形式译码的轮廓（6栏）
INTERSEC	计算两个轮廓元素之间的交点。 （仅用于通过 CONTPRON 建立的表格）.
EXECTAB	逐段处理某个图表的轮廓元素 （仅用于通过 CONTPRON 建立的图表）.
CALCDAT	从三个或者四个点中计算某个圆的半径和中点。

注意

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。

14.2 轮廓预处理 (CONTPRON)

功能

在CONTPRON之后运行的程序段描述预处理的轮廓。这些程序段没有执行，而是存放在轮廓表格中。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。

注意

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。

编程

```
CONTPRON (TABNAME, BEARBART, NN, MODE)
CEXECUTE (FEHLER)
```

参数

CONTPRON	启用轮廓预处理
TABNAME	轮廓图表的名称
BEARBART	加工方式参数： "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
NN	INT型结果变量中的根切数量
MODE	加工方向，INT型 0 = 轮廓预处理 向前（默认值） 1 = 两个方向中的轮廓预处理
EXECUTE	结束轮廓预处理
FEHLER	故障反馈变量，INT型 1 = 故障；0 = 没有故障

使用**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。

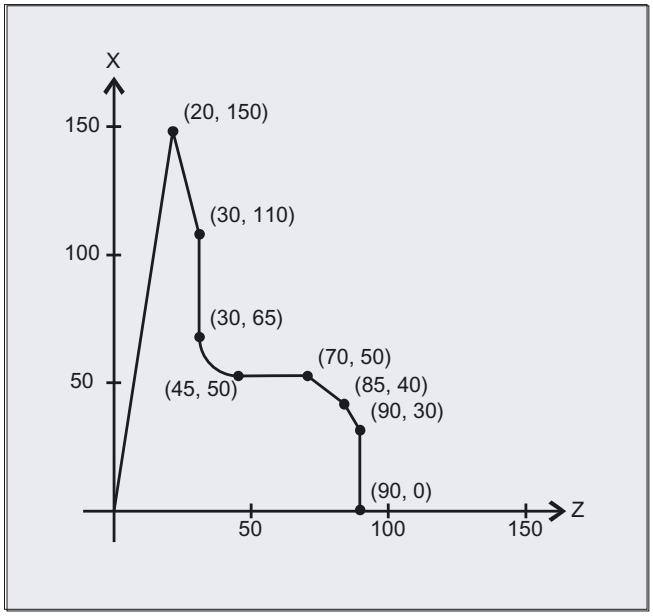
举例：

```
N30 CONTPRON(...)
N40 G1 X...Z...
N50 ...
N100 EXECUTE(...)
```

建立曲线图表举例1

编制一个轮廓表格，带

- 名称 KTAB,
- 最多30个轮廓单元（圆弧，直线），
- 一个变量，表明所出现的咬边单元的个数
- 用于故障信息的一个变量。



NC-零件程序

```
N10 DEF REAL KTAB[30,11]                ;名称为KTAB且
                                        ;例如最多有30个轮廓元素的曲线图表
                                        ;参数值11是一个固定量
N20 DEF INT ANZHINT                     ;名称为
                                        ;ANZHINT用于根切元素数量的变量
N30 DEF INT FEHLER                      ;用于应答的变量
                                        ;0=没有故障， 1=故障
N40 G18
N50 CONTPRON (KTAB,"G",ANZHINT)         ;调用轮廓预处理
N60 G1 X150 Z20                         ;N60 ~ 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)                    ;结束填写轮廓图表，
                                        ;转换到正常程序运行方式
N140                                    ;图表的其它处理
```

相关的表格KTAB

(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) 编码轮廓方式，用于运动

X = abc 时可能有的数值
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。

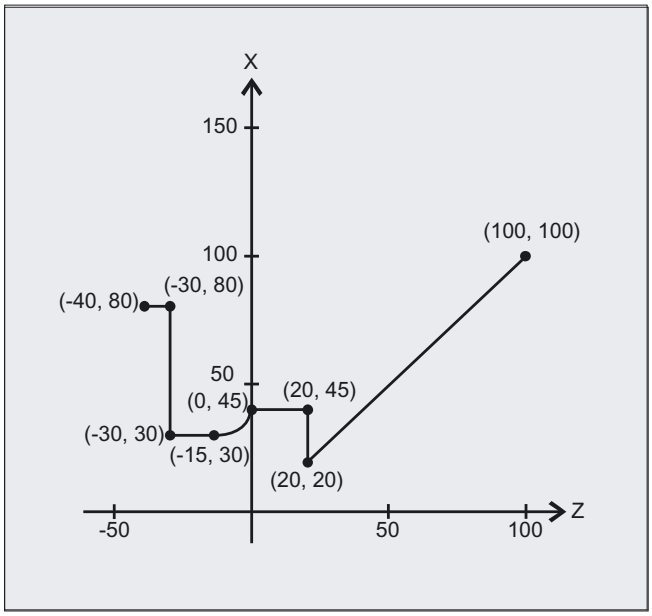
(9), (10) 如果是圆弧段，则轮廓单元的圆心坐标

(9) = 横坐标， (10) = 纵坐标

建立曲线图表举例2

编制一个轮廓表格，带

- 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-零件程序

```
N10 DEF REAL KTAB[92.11]                ;名称为KTAB且;例如最多有92个轮廓元素的曲线图表
                                        ;参数11是一个固定量
N20 CHAR BT="L"                         ;用于CONTPRON的运行方式：
                                        ;纵向切削，外侧加工
N30 DEF INT HE=0                        ;根切元素的数量=0
N40 DEF INT MODE=1                      预处理，前进和后退
N50 DEF INT ERR=0                       ;故障反馈
...
N100 G18 X100 Z100 F1000
N105 CONTPRON (KTAB, BT, HE, MODE)      ;调用轮廓预处理
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)                       ;结束填写轮廓图表，
                                        ;转换到正常程序运行方式
...
```

相关的表格KTAB

在结束轮廓预处理之后，可以在两个方向使用轮廓。

行	列（栏）										
	(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

(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) 编码轮廓方式，用于运动

X = abc 时可能有的数值
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

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）

前提条件

在调用 CONTPRON 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。

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

另外还有倒圆和倒角。

可以通过 CIP 和 CT 编程圆。

样条、多项式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。G70和G71/G700和G710之间的转换也同样适用。

在预处理轮廓表格期间如果用GEOAX更换几何轴会导致报警。

预处理结束

通过调用EXECUTE（变量）可以在写轮廓之后切换回正常的程序运行，并且结束轮廓预处理。变量显示：

1 = 故障

0 = 没有故障（可以无故障预处理轮廓）。

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 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。

14.3 轮廓解码 (CONTDCON)

功能

在CONTDCON之后运行的程序段描述译码的轮廓。这些程序段没有处理，而是译码后以占用存储器最小的方式存放在一个6栏的轮廓表格中。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，来自图表行中的应用（例如循环）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。在待列表的程序块中，允许用于CONTDCON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。

编程

CONTDCON (TABNAME,MODE)

EXECUTE (FEHLER)

参数

CONTDCON	启用轮廓预处理
TABNAME	轮廓图表的名称
MODE	加工方向，类型 INT 0 = 根据轮廓程序段的顺序预处理轮廓（默认值）
EXECUTE	结束轮廓预处理
FEHLER	故障反馈变量，INT型 1 = 故障；0 = 没有故障

用EXECUTE关断轮廓预处理，同时切换回通常的加工方式。

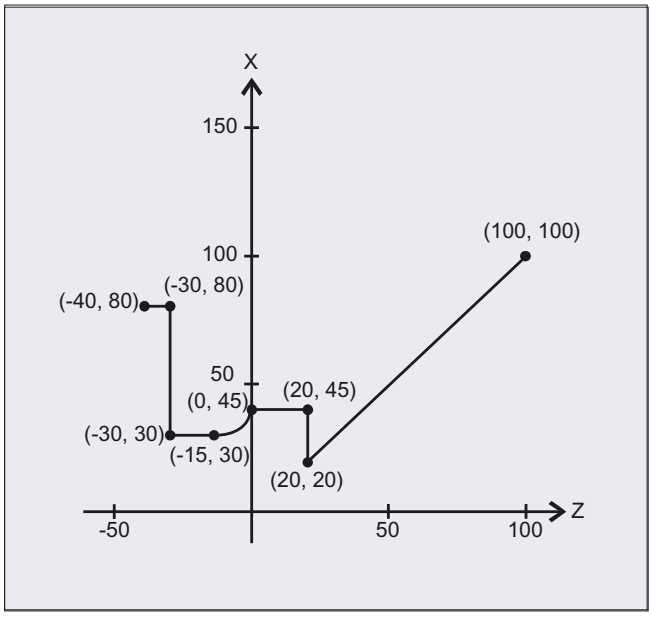
举例：

```
N30 CONTDCON(...)
N40 G1 X...Z...
N50 ...
N100 EXECUTE(...)
```

建立轮廓图表举例

编制一个轮廓表格，带

- 名称 KTAB,
- 轮廓单元（圆弧，直线），
- 工作方式车削，
- 向前准备。



NC-零件程序

```
N10 DEF REAL KTAB[9,6]          ;名称为 KTAB
                                且有9个图表行的轮廓图表。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            ;默认值 0:仅在已编程的轮廓方向中
                                。值1不允许。
N30 DEF INT ERROR = 0           ;故障反馈
...
N100 G18 G64 G90 G94 G710
```

```
N101 G1 Z100 X100 F1000
N105 CONTDCON (KTAB, MODE)                    ;调用轮廓解码
                                              ;MODE 允许省略，见上
N110 G1 Z20 X20 F200                          ;轮廓说明
N120 G9 X45 F300
N130 Z0 F400
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)                           ;结束填写轮廓图表，
                                              ;转换到正常程序运行方式
...
```

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
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

前提条件

在调用 CONTDCON 之前必须

1. 返回到一个可以无轮廓冲突进行加工的起始点
2. 关断带G40的刀尖半径补偿。

允许的运行指令，坐标系

下面的G组允许用于轮廓编程，带所说明的指令：

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

另外还有倒圆和倒角。

可以通过 CIP 和 CT 编程圆。

样条、多项式螺纹功能会导致出错。

不允许通过接通一个框架在CONTDCON 和 EXECUTE之间改变坐标系。G70和G71/G700和G710之间的转换也同样适用。

在预处理轮廓表格期间如果用GEOAX更换几何轴会导致报警。

预处理结束

通过调用EXECUTE（故障），可以根据所写轮廓表格切换回正常的程序运行，并且结束轮廓预处理。在相应的变量FEHLER中进行应答：

0 = 没有故障（可以正常预处理轮廓）
1 = 故障
非法指令，错误的初始条件，在没有EXECUTE( )的情况下重复调用 CONTDCON，轮廓程序段太少或者定义太小的图表导致报警。

在编程的轮廓方向切削

使用CONTDCON生成的轮廓图表可用于在已编程的轮廓方向中进行切削。

14.4 两个轮廓元素的交点 (INTERSEC)

功能

INTERSEC 用来从使用 CONTPRON 生成的轮廓图表中计算出两个已经过标准化处理的轮廓元素的交点。所说明的状态用来提示是否存在交点（TRUE = 有交点）（FALSE = 没有交点）。

注意

请注意：变量必须在使用之前已经定义。

编程

```
VARIB=INTERSEC (TABNAME1[n1], TABNAME2[n2], TABNAME3)
```

参数

VARIB	用于状态的变量： TRUE:找到交点 假：没有找到交点
TABNAME1[n1]	表格名称和n1。第一个表格的轮廓单元
TABNAME2[n2]	表格名称和n2。第二个表格的轮廓单元
TABNAME3	用于交点坐标的表格名称，在有效的平面G17-G19中。

举例

计算表格KTAB1中的轮廓单元3与表格KTAB2中的轮廓单元7的交点。活动平面中的交点坐标保存在交点(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）。

```
DEF REAL KTAB1 [12, 11]                ;轮廓图表 1
DEF REAL KTAB2 [10, 11]                ;轮廓图表 2
DEF REAL 交点 [2]                      ;交点图表
DEF BOOL ISPOINT                       ;用于状态的变量
…
```

```
N10 ISPOINT=INTERSEC (KTAB1[3],KTAB2[7],交点)
                                                ;调用轮廓元素的交点
N20 IF ISPOINT==FALSE GOTOF KEINSCH             ;跳转到 KEINSCH
…
```

14.5 从表格中执行某个轮廓元素 (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。

编程

```
EXECTAB (TABNAME[n])
```

参数

TABNAME[n]	表格名称，带单元号n

举例

使用子程序 EXECTAB 可逐段执行完表格 KTAB 的轮廓元素。在调用时可以一个接一个把单元0到2交付使用。

```
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
```

14.6 计算圆的数据 (CALCDAT)

功能

半径和圆心坐标由三个或者四个已知的圆弧点计算。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。

编程

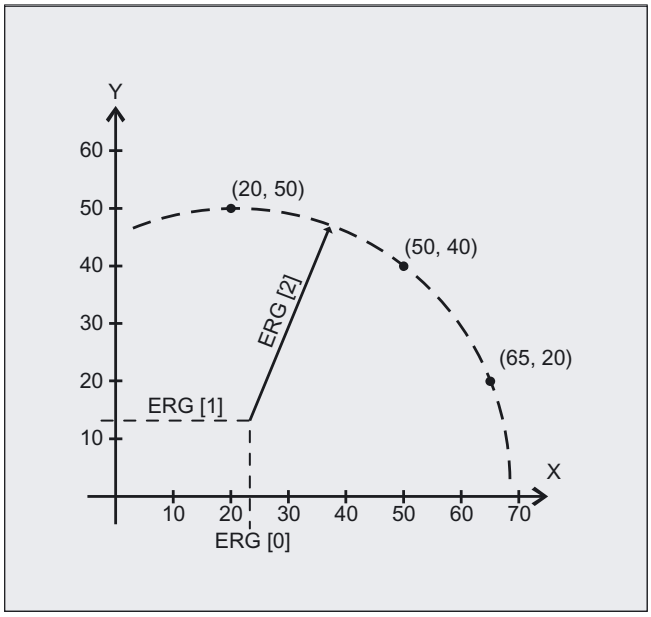
```
VARIB=CLCDAT (PKT[n,2], ANZ, ERG)
```

参数

VARIB	变量，用于状态
	TRUE = 圆弧，FALSE = 不是圆弧
PKT [n,2]	用于计算的点
	n = 点的数量（3或者4）； 2 = 2个点坐标的说明
ANZ	用于计算的点的个数：3或4
ERG [3]	用于结果的变量：圆心坐标和半径的说明：
	0 = 横坐标，1 = 圆心纵坐标；2 = 半径

举例

由3个点计算出它们是否位于一个圆弧段。




```
N10 DEF REAL PKT[3,2]=(20,50,50,40,65,20)        ;定义点
N20 DEF REAL ERG[3]                              ;结果
N30 DEF BOOL STATUS                              ;用于状态的变量
N40 STATUS = CALCDAT(PKT,3,ERG)                  ;调用算得的圆数据
N50 IF STATUS == FALSE GOTOF ERROR               ;跳转到出错
```

表 15

15.1 指令列表_1

在指令列表中可查阅工作准备过程中现有的所有编程指令以及所有G代码汇总。

图例说明：

[1] 程序初始的默认设置（若没有另行编程，即为控制器的出厂设置）。

[2] 组编号相当于“G功能/位移条件列表”段落中的表格。

[3] 绝对终点：模态方式；增量终点：逐段方式；否则依据G功能语法规定为模态/逐段方式。

[4] IPO参数作为圆中心起增量作用。使用AC它们可以绝对编程。如果是其它含义（例如螺距）就忽略地址修改。

[5] 关键字对SINUMERIK 810D无效。

[6] 关键字对SINUMERIK 810D/NCU571无效。

[7] 关键字仅适用于SINUMERIK FM-NC

[8] OEM用户可以增加两个附加的插补类型。名称可以由OEM用户改变。

[9] 不允许将扩展式地址写入方式用于这些功能。

名称	意义	赋值	说明，注释	句法	模态/非模态	组[2]
:	程序段号 – 主程序段 (参见 N)	0 ... 9999 9999 仅整数，没有前置符	程序段的特殊标记 - 替代 N...;该程序段应当含有用于下列某个完整处理程序段的所有指令。	例如. :20		
A	轴	实数			m,s [3]	
A2 [5]	刀具定向：欧拉角	实数			s	
A3 [5]	刀具定向：方向矢量	实数			s	

A4 [5]	刀具定向，用于程序段起始	实数			s	
A5 [5]	程序段尾的刀具定位； 标准矢量分量	实数			s	
ABS	绝对值	实数				
AC	绝对尺寸输入	0 ..., 359.9999°		X=AC(100)	s	
ACC [5]	轴向加速度	实数，无符号			m	
ACN	回转轴绝对尺寸参数，在负方向返回位置			A=ACN(...) B=ACN(...) C=ACN(...)	s	
ACP	回转轴绝对尺寸参数，在正方向返回位置			A=ACP(...) B=ACP(...) C=ACP(...)	s	
ACOS	反余弦（三角函数）	实数				
ADIS	用于轨迹功能的精磨距离 G1, G2, G3, ...	实数，无符号			m	
ADISPOS	快速移动G0的精磨距离	实数，无符号			m	
ADISPOSA	用于IPOBRKA的公差窗口尺寸	整数，实数		ADISPOSA=.. 或者 ADISPOSA(<轴>[,REAL])	m	
ALF	快速退刀角度	整数，无符号			m	
AMIRROR	可编程镜像			AMIRROR X0 Y0 Z0 ；自身程序段	s	3
AND	逻辑与					
ANG	轮廓线角度	实数				
AP	极角	0,..., ± 360°			m,s [3]	
APR	读取/显示访问保护 (access protection read)	整数，无符号				
APW	写入访问保护 (access protection write)	整数，无符号				
AR	圆弧张角	0, ..., 360°			m,s [3]	
AROT	可编程旋转 （附加旋转）	围绕 第1个几何轴旋转： -180° ..180° 第2个几何轴： -89.999° ..90° 第三几何轴： -180° .. 180°		AROT X...Y...Z... AROT RPL= ;自身的程序段	s	3

AROTS	可编程的框架旋转，带立体角			AROT X...Y... AROT Z...X... AROT Y...Z... AROT RPL= ;自身的程序段	s	3
AS	宏指令定义	字符串				
ASCALE	可编程的比例			ASCALE X...Y...Z... ;自身的程序段	s	3
ASIN	反余弦 (三角函数)	实数				
ASPLINE	Akima样条				m	1
ATAN2	反正切2	实数				
ATRANS	附加的可编程偏置 （附加转换）			ATRANS X...Y...Z... ;自身的程序段	s	3
AX	变量轴名称	实数			m,s [3]	
AXCSWAP	容器轴转接			AXCSWAP(CTn, CTn+1,..)		25
AXIS	数据类型：轴名称		可以接收一个文件的名称			
AXNAME	把输入端字符串转换为轴名称	字符串	如果输入端字符串没有有效的轴名称，则设置一个报警			
AXSTRING	转换主轴编号字符串 (get string)	字符串	可以接收一个文件的名称	AXSTRING[SPI(n)]		
B	轴	实数			m,s [3]	
B_AND	位方式“与”					
B_NOT	位方式“非”					
B_OR	位方式“或”					
B_XOR	位方式“异 - 或”					
B2 [5]	刀具定位： 欧拉角	实数			s	
B3 [5]	刀具定位： 方向矢量分量	实数			s	
B4 [5]	刀具定向，用于程序段起始	实数			s	
B5 [5]	程序段尾的刀具定位； 标准矢量分量	实数			s	

BAUTO	通过后面的3点确定第一个样条段				m	19
BLSYNC	在下一个程序段转换时才开始中断程序的加工。					
BNAT [1]	自然过渡到第一个样条程序段 (begin natural)				m	19
BOOL	数据类型：真值TRUE/FALSE或者0/1					
BRISK [1]	突变型的轨迹加速度				m	21
BRISKA	接通突变型轨迹加速度，用于编程的轴					
BSPLINE	B 样条				m	1
BTAN	切向过渡到第一个样条程序段 (begin tangential)				m	19
C	轴	实数			m,s [3]	
C2 [5]	刀具定向：欧拉角	实数			s	
C3 [5]	刀具定位： 方向矢量分量	实数			s	
C4 [5]	刀具定向，用于程序段起始	实数			s	
C5 [5]	程序段尾的刀具定位； 标准矢量分量	实数			s	
CAC	以绝对方式向某个位置逼近 (coded position:absolute coordinate)		编码的值是表索引；返回表数值			
CACN	以绝对负方向逼近保存在表中的值。 (coded position absolute negative)		允许用于将旋转轴编程为定位轴			
CACP	以绝对正方向逼近保存在表中的值。 (coded position absolute positive)					
CALCDAT	从3或者4个点中计算某个圆的半径和中心 (calculate circle data)	VAR Real [3]	必须区分这些点。			
CALL	间接子程序调用			CALL PROGVAR		
CALLPATH	子程序调用时可编程的查找路径		可以编程一个带CALLPATH的路径，用于现在的NCK文件系统。	CALLPATH(/_N_WKS_DIR/ _N_MYWPD/子程序标识符_SPF)		

CANCEL	中止模态同步动作	INT	用给定的ID来取消。 没有参数：所有模态同步动作被取消。			
CASE	有条件程序跳转					
CDC	直接逼近某个位置 (coded position:direct coordinate)		参见 CAC			
CDOF [1]	碰撞监控关闭 (collision detection OFF)				m	23
CDON	轮廓冲突检测“开”				m	23
CDOF2	碰撞监控关闭 (collision detection OFF)		仅用于 CUT3DC		m	23
CFC [1]	轮廓上恒定进给 (constant feed at contour)				m	16
CFIN	仅当是内曲率 而不是外曲率时恒定进给 (constant feed at internal radius)				m	16
CFTCP	刀沿参考点恒定进给（圆心路径） (constant feed in tool-center-point)				m	16
CHAN	规定数据有效区。		每个通道一次			
CHANDATA	设定通道号，用于通道数据存取	INT	仅在初始化模块中允许			
CHAR	数据类型：ASCII-字符	0, ..., 255				
CHECKSUM	通过一个字符串数组构成检查和，带一个确定的长度	最大长度为 32	发送16进制数字的字符串	ERROR=CHECKSUM		
CHF CHR	棱边；值 = 棱边长度，在运动方向（倒棱） 棱边；值 = 棱边长度	实数，无符号			S	
CHKDNO	D号的单一性检查					
CIC	以增量方式逼近某个位置 (coded position:incremental coordinate)		参见 CAC			
CIP	通过中间点进行圆弧插补			CIP X...Y...Z... I1=...J1=...K1=...	m	1
CLEARM	复位一个/多个标号，用于通道协调	INT, 1 - n	对自身通道中的加工没有影响			

CLRINT	撤销选择中断	INT	参数：中断编号			
CMIRROR	对一个坐标轴的镜像	FRAME				
COARSEA	在到达“粗准停”时运行结束			COARSEA=.. 或者 COARSEA[n]=..	m	
COMPOF [1,6]	压缩器“关”				m	30
COMPON [6]	压缩器“开”				m	30
COMPCURV	压缩器“开”：曲率恒定的多项式				m	30
COMPCAD	压缩器“开”：优化的表面品质				m	30
CONTDCON	启用表格形式的轮廓解码					
CONTPRON	启用找零准备 (contour preparation ON)					
COS	余弦（三角函数）	实数				
COUPDEF	定义 ELG-组合/同步主轴组合 (couple definition)	字符串	程序段转换（SW）性能： NOC:没有SW控制， FINE/COARSE:当“同步运行精/粗”时程序段转换， IPOSTOP:当额定值端结束叠加运动时程序段转换			
COUPDEL	删除ELG连接					
COUPOF	ELG-组合/同步主轴对 断开 (couple OFF)					
COUPON	ELG-组合/同步主轴对 接合 (couple ON)					
COUPRES	复位ELG-组合 (couple reset)		已编程的值无效；机床数据值有效			
CP	轨迹运行（连续路径）				m	49
CPRECOF [1,6]	关闭可编程轮廓精度 (contour precision OFF)				m	39
CPRECON [6]	开启可编程轮廓精度 (contour precision ON)				m	39
CPROT	通道专用的保护区“开/关”					
CPROTDEF	定义某个特定通道的保护范围 (channel specific protection area definition)					

CR	圆弧半径	实数，无符号			S	
CROT	旋转当前坐标系	FRAME	参数的最大数目： 6			
CROTS	可编程的立体角框架旋转 (在指定的轴中旋转)			CROT X...Y... CROT Z...X... CROT Y...Z... CROT RPL= ;自身的程序段	S	
CSCALE	比例系数，用于多个轴	FRAME	参数的最大数目： 2 * 轴数量$_{max}$			
CSPLINE	立体样条				m	1
CT	圆弧，带切线过渡			CT X...Y....Z...	m	1
CTAB	依据曲线表中的引导轴位置计算跟随轴位置	实数	参数4/5不可编程时：标准刻度			
CTABDEF	表格定义“开”					
CTABDEL	删除曲线表					
CTABEND	表格定义“关”					
CTABEXIST	检查带号n的曲线表		参数n			
CTABFNO	存储器中还可能的曲线表个数		存储器类型			
CTABFPOL	存储器中还可能的多项式个数		存储器类型			
CTABFSEG	存储器中还可能的曲线段个数		存储器类型			
CTABID	提供曲线表的表编号		参数 n 和 mem类型			
CTABINV	依据曲线表中的跟随轴位置计算引导轴位置	实数	参见 CTAB			
CTABISLOCK	送回n号的曲线表的禁止状态		参数n			
CTABLOCK	设置禁止删除和改写		参数 n, m, 和 mem类型			
CTABMEMTYP	送回存储器，在该存储器中编制了n号的曲线表		参数n			
CTABMPOL	存储器中最大可能的多项式个数		存储器类型			
CTABMSEG	存储器中最大可能的曲线段个数		存储器类型			
CTABNO	所定义的曲线表的个数，不管存储器类型		没有说明参数			
CTABNOMEM	在存储器SRAM或者DRAM中定义的曲线表的个数		存储器类型			

CTABPERIOD	送回n号的表格周期性		参数n			
CTABPOL	存储器中已经使用的多项式个数		存储器类型			
CTABPOLID	n号曲线表所使用的曲线多项式个数		参数n			
CTABSEG	已经使用的曲线段的个数		存储器类型			
CTABSEGID	n号曲线表所使用的曲线段个数		参数n			
CTABSEV	提供曲线表一个分段的跟随轴终值		通过LW确定段	R10 = CTABSEV(LW, n, 度, F轴, L轴)		
CTABSSV	提供曲线表一个分段的跟随轴起始值		通过LW确定段	R10 = CTABSSV(LW, n, 度, F轴, L轴)		
CTABTEP	在曲线表结束处提供引导轴的值		曲线图表末端的引导值	R10 = CTABTEP(n, 度, L轴)		
CTABTEV	在曲线表结束处提供跟随轴的值		在曲线表结束处的跟随值	R10 = CTABTEV(n, 度, F轴)		
CTABTMAX	提供曲线表跟随轴的最大值		曲线表的跟随轴	R10 = CTABTMAX(n, F轴)		
CTABTMIN	提供曲线表跟随轴的最小值		曲线表的跟随轴	R10 = CTABTMIN(n, F轴)		
CTABTSP	在曲线表开始处提供引导轴的值		曲线图表起始的引导值	R10 = CTABTSP(n, 度, L轴)		
CTABTSV	在曲线表开始处提供跟随轴的值		在曲线表开始处的跟随轴	R10 = CTABTSV(n, 度, F轴)		
CTABUNLOCK	取消禁止删除和改写		参数 n, m, 和 mem类型			
CTRANS	零点偏移，用于多个轴	FRAME	最多8个轴			
CUT2D [1]	2½D 刀具补偿 (cutter compensation type 2dimensional)				m	22
CUT2DF	2½D 刀具补偿 (cutter compensation type 2dimensional frame); 刀具补偿相对于当前框架有效（斜面）				m	22
CUT3DC [5]	3D 圆周铣削刀具补偿				m	22

CUT3DCC [5]	3D 圆周铣削刀具补偿，带限制面积				m	22
CUT3DCCD [5]	3D 圆周铣削刀具补偿，带差分刀具限制面积				m	22
CUT3DF [5]	3D 面铣削刀具补偿				m	22
CUT3DFF [5]	3D 面铣削刀具补偿，带恒定的刀具定向，与有效框架相关。				m	22
CUT3DFS [5]	3D 面铣削刀具补偿，带恒定的刀具定向，与有效框架无关。				m	22
CUTCONO[1]	恒定半径补偿“关”				m	40
CUTCONON	恒定半径补偿“开”				m	40
D	刀具补偿号	1, ... 32 000	包含某个刀具的补偿数据 T...; D0 → 刀具的补偿值	D...		
DC	用于回转轴的绝对尺寸参数，直接返回位置			A=DC(...) B=DC(...) C=DC(...) SPOS=DC(...)	s	
DEF	变量定义	整数，无符号				
DEFAULT	跳转到CASE回路		如果表达式没有满足所说明的值，则跳转。			
DEFINE	宏指令定义					
DELAYFSTON	定义一个停止-延时范围的开始 (DELAY Feed Stop ON)		隐式，当G331/G332激活时		m	
DELAYFSTOF	定义一个停止-延时范围的结束 (DELAY Feed Stop OF)				m	
DELDTG	删除剩余行程 (Delete distance to go)					
DELETE	删除所说明的文件。文件名可以用路径和文件标识给出。		可以删除所有文件			
DELT	删除刀具		可以删除Duplo号			

DIAMCYOF	为 G90/91 编程半径：启用。显示时该组中最后有效的G代码有效。		半径编程，最终激活的G代码		m	29
DIAMOF[1]	直径编程：关闭 (Diametral programming OFF)		为 G90/G91 编程半径		m	29
DIAMON	进行直径编程：开启 (Diametral programming ON)		为G90/G91编程直径		m	29
DIAM90	G90的直径编程，G91的半径编程				m	29
DILF	快速退刀长度				m	
DISABLE	中断“关”					
DISC	过渡圆弧刀具半径补偿加强	0, ..., 100			m	
DISPLOF	抑制当前的程序段显示 (Display OFF)					
DISPR	Repos(再定位)-轨迹差值	实数，无符号			S	
DISR	Repos（再定位)-距离	实数，无符号			S	
DITE	螺纹导出位移	实数			m	
DITS	螺纹导入位移	实数			m	
DIV	整数 - 分割					
DL	刀具补偿和	INT			m	
DRFOF	关闭（取消）手轮偏移（DRF）				m	
DRIVE[9]	与速度相关的轨迹加速度				m	21
DRIVEA	开启折弯型加速度特征曲线，用于编程的轴					
DYNFINISCH	用于精加工的动态功能		G组工艺	DYNFINISH G1 X10 Y20 Z30 F1000	m	59
DYNNORM	和至今为止一样，标准动态			DYNNORM G1 X10	m	59
DYNPOS	用于定位运行的动态功能，攻丝			DYNPOS G1 X10 Y20 Z30 F...	m	59
DYNROUGH	用于粗加工的动态功能			DYNROUGH G1 X10 Y20 Z30 F10000	m	59
DYNSEMIFIN	用于修整的动态功能			DYNSEMIFIN G1 X10 Y20 Z30 F2000	m	59
DZERO	分配到通道的T0单元的所有刀具D号被设置无效。					
EAUTO	通过后面的3点确定最后一个样条段				m	20
EGDEF	定义一个电子齿轮箱 (Electronic gear define)		用于具有5个以下引导轴的1个跟随轴			

EGDEL	删除跟随轴的偶合定义 (Electronic gear delete)	触发前进停止			
EGOFC	连续断开电子齿轮箱 (Electronic gear OFF continuous)				
EGOFS	有选择性断开电子齿轮箱 (Electronic gear OFF selectiv)				
EGON	启用电子齿轮箱 (electronic gear ON)	没有同步动作			
EGONSYN	启用电子齿轮箱 (electronic gear ON synchronized)	有同步动作			
EGONSYNE	启用电子齿轮箱，有起动模式设定 (electronic gear ON synchronized)	有同步动作			
ELSE	当IF条件不满足时，程序跳转				
ENABLE	中断“开”				
ENAT [1,7]	曲线自然过渡到下一个运动程序段 (end natural)			m	20
ENDFOR	FOR-计数循环的结束行				
ENDIF	IF跳转的结束行				
ENDLOOP	程序死循环 LOOP 结束行				
ENDPROC	带起始行 PROC 的一个程序的结束行				
ENDWHILE	WHILE-循环的结束行				
ETAN	曲线切线过渡到下一个样条开始的运行程序段			m	20
EVERY	当条件从FALSE转换到TRUE时，执行同步动作				
EXECSTRING	给出一个字符串变量，带待执行的零件程序行	间接零件程序行	EXECSTRING(MFCT1 << M4711)		
EXECTAB	执行一个运动表中的某个项 (Execute table)				
EXECUTE	程序执行“开”	从找零准备模式或者根据保护区的结构恢复到正常程序处理			

EXP	指数函数 e^x	实数				
EXTCALL	执行外部子程序		在“从外部执行”模式中加载HMI的程序			
EXTERN	通告一个子程序，给出参数					
F	进给值 （和G4一起，同时在F中编程停留时间）	0.001, ..., 99 999. 999	刀具/工件轨迹速度； 尺寸单位：mm/分钟 或者 mm/转，取决于 G94 或者 G95	F=100 G1 ...		
FA	轴向进给	0.001, ..., 999999.999 mm/分钟，度/分钟；0.001, ..., 39999.9999 英寸/分钟		FA[X]=100	m	
FAD	进给，用于平滑逼近和离开 (Feed approach/depart)	实数，无符号				
FALSE	逻辑常量：错	BOOL	可通过 Integer型常量0 替换			
FCTDEF	定义多项式函数		在 SYNFCT 或者 PUTFTOCF 中求值			
FCUB [6]	根据立方样条的进给率变量 (feed cubic)		作用于带G93和G94的进给		m	37
FD	手轮修调的路径进给 (feed DRF)	实数，无符号			S	
FDA	手轮修调的轴向进给率 (feed DRF axial)	实数，无符号			S	
FENDNORM	拐角延迟“关”				m	57
FFWOF [1]	预控制“关”				m	24
FFWON	预控制“开”				m	24
FIFOCTRL	进刀运行缓冲控制				m	4
FIFOLEN	可编程的进刀深度					

FINEA	在到达“精准停”时运行结束			FINEA=... 或者 FINEA[n]=..	m	
FL	同步轴的速度极限 (feed limit)	实数，无符号	适用以 G93, G94, G95 所设置的单位 (最大快速行程)	FL [轴] =...	m	
FLIN [6]	进给线性改变		作用于带G93和G94的进给		m	37
FMA	多重轴向进给率 (feed multiple axial)	实数，无符号			m	
FNORM [1,6]	正常进给，符合DIN66025				m	37
FOCOF	关闭带极限力矩/力的运行				m	
FOCON	开通带极限力矩/力的运行				m	
FOR	带固定运行次数的计数循环					
FP	固定点：待返回运行固定点的号	整数，无符号		G75 FP=1	S	
FPO	通过一个多项式编程的进给曲线 (Feed polynomial)	实数	二次、三次多项式系数			
FPR	回转轴标记	0.001, ..., 999999.999		FPR (回转轴)		
FRAME	数据类型，用于确定坐标系		每个几何轴包含： 位移、旋转、剪切角、缩放、镜像； 每个辅助轴： 位移、缩放、镜像			

FRC,	进给，用于半径和棱边				s	
FRCM,	进给，用于半径和棱边模态				m	
FTOC	修改刀具精补偿		取决于一个用FCTDEF确定的3次多项式			

FTOCOF [1,6]	关闭在线精刀具补偿 (fine tool offset OFF)				m	33
FTOCON [6]	开启在线精刀具补偿 (fine tool offset ON)				m	33
FXS	开启运行到固定挡块 (fixed stop)	整数，无符号	1 = 选定， 0 = 不选择		m	
FXST	用于运行到固定挡块的力矩极限 (fixed stop torque)	%	可选参数		m	
FXSW	运行碰到固定挡块的监控窗口	毫米，英寸或者度	可选参数			

G功能						
G	G功能（准备功能） G功能分为各个G组。在一个程序段中只能编写某组的一个G功能。 G功能可以是模态的（直到被同组中其他功能替代），或者是非模态的（只在写入的程序段中有效）。	仅为整数，预设值		G...		
G0	线性插补，带快速移动		运行指令	G0 X...Z...	m	1
G1 [1]	线性插补，带进给			G1 X...Z...F...	m	1
G2	顺时针圆弧插补			G2 X...Z...I...K...F... ;中点和终点 G2 X...Z...CR=...F... ;半径和终点 G2 AR=...I...K...F... ;张角和 ;中点 G2 AR=...X...Z...F... ;张角和 ;终点	m	1
G3	逆时针圆弧插补			G3 ...;否则就如同 G2	m	1
G4	停留时间，给定时间		特殊运行	G4 F... ;停留时间 in soder G4 S... ;在 ;主轴旋转过程中的停留时间 ;自身的程序段	s	2
G9	准停速度取消				s	11
G17 [1]	选择工作平面X/Y		横向进给方向Z		m	6

G18	选择工作平面Z/X		横向进给方向Y		m	6
G19	选择工作平面Y/Z		横向进给方向X		m	6
G25	工作范围下限		赋值，	G25 X..Y..Z..;自身的程序段	s	3
G26	工作范围上限		在通道轴中	G26 X..Y..Z..;自身的程序段	s	3
G33	螺纹插补，带恒定螺距	0.001, ..., 2000.00 毫米/转	运行指令	G33 Z...K...SF=... ;圆柱螺纹 G33 X...I...SF=... ;平螺纹 G33 Z...X...K...SF=... ;锥形螺纹（在Z轴中；行程比在X轴中的大） G33 Z...X...I...SF=... ;锥形螺纹（在X轴中，行程比在Z轴中的大）	m	1
G34	螺距增大 (递增变化）		运行指令	G34 Z...K...F_{ZU}=...	m	1
G35	螺距增大 (递减变化）		运行指令	G35 Z...K...F_{AB}=...	m	1
G40 [1]	刀具半径补偿“关”				M...	7
G41	刀具半径补偿，轮廓左边				M...	7
G42	刀具半径补偿，轮廓右边				M...	7
G53	抑制当前零点偏移（程序段方式）		包括 已编程的位移		s	9
G54	1. 可设定的零点偏移				M...	8
G55	2. 可设定的零点偏移				M...	8
G56	3. 可设定的零点偏移				M...	8
G57	4. 可设定的零点偏移				M...	8
G58	可编程的偏移		轴向替换		s	3
G59	可编程的偏移		添加式轴向 替换		s	3
G60 [1]	准停速度取消				M...	10
G62	当激活刀具半径补偿时内角上的 角减速 (G41, G42)		仅与轨迹控制运行方式共同使用	G62 Z...G1	M...	57
G63	攻丝，带补偿夹具			G63 Z...G1	s	2
G64	准停 - 轨迹控制运行				M...	10
G70	英制尺寸（长度）				M...	13
G71 [1]	公制尺寸（长度）				M...	13
G74	回参考点运行			G74 X...Z..;自身的程序段	s	2
G75	回固定点运行		加工轴	G75 FP=..X1=...Z1=... ;自身的程序段	s	2
G90 [1]	绝对尺寸输入			G90 X...Y...Z...(...)Y=AC(...) 或者 X=AC Z=AC(...)	m s	14

G91	增量尺寸输入			G91 X...Y...Z... 或者 X=IC(...) Y=IC(...) Z=IC(...)	m s	14
G93	时间倒数进给1/分钟		一个程序段的开始运行：时间长度	G93 G01 X...F...	M...	15
G94 [1]	直线进给率F，单位：毫米/分钟或英寸/分钟和度/分钟				M...	15
G95	转速进给F，单位毫米/转或者英寸/转				M...	15
G96	恒定切削速度“开”			G96 S...LIMS=...F...	M...	15
G97	恒定切削速度“关”				M...	15
G110	极点编程，相对于最后编程的给定位置			G110 X..Y..Z..	s	3
G111	极点编程，相对于当前工件坐标系的零点			G110 X..Y..Z..	s	3
G112	极点编程，相对于最后有效的极点			G110 X..Y..Z..	s	3
G140 [1]	确定返回运行方向WAB，通过G41/G42				M...	43
G141	返回运行方向WAB，轮廓左边				M...	43
G142	返回运行方向WAB，轮廓右边				M...	43
G143	返回运行方向WAB，切线相关				M...	43
G147	转换以直线返回运行				s	2
G148	转换以直线开始运行				s	2
G153	抑制当前框架，包括基准框架				s	9
G247	转换以1/4个圆弧返回运行				s	2
G248	转换以1/4个圆弧开始运行				s	2
G290	转换到SINUMERIK模式 ON				M...	47
G291	转换到ISO2/3模式 开启				M...	47
G331	攻丝	± 0.001,..., 2000.00 毫米/转	运行指令		M...	1
G332	退回（攻丝）				M...	1
G340 [1]	空间返回运行程序段（深度和平面中同时）（螺旋）		当平滑逼近或者离开时起作用		M...	44
G341	首先在垂直轴上横向进给(z)，然后在平面中返回运行		当平滑逼近或者离开时起作用		M...	44
G347	转换以半圆返回运行				s	2
G348	转换以半圆开始运行				s	2
G450 [1]	过渡圆弧		刀具半径补偿时的拐角性能		M...	18
G451	等距离交点				M...	18

G460 [1]	在WRK时返回运行/开始运行特性			M...	48
G461	在WRK时返回运行/开始运行特性			M...	48
G462	在WRK时返回运行/开始运行特性			M...	48
G500 [1]	如果在G500中没有值，则关断所有可设定框架。			M...	8
G505 ....G599	5. ... 99. 可设定的零点偏移			M...	8
G601 [1]	在精准停时程序段转换	有效，仅当G60有效时，或者G9带有可编程的过渡磨削时		M...	12
G602	在粗准停时程序段转换			M...	12
G603	在IPO程序段结束处程序段转换			M...	12
G641	准停 - 轨迹控制运行		G641 ADIS=...	M...	10
G642	精磨削，带轴向精度			M...	10
G643	程序段内部精磨削			M...	10
G644	精磨削，给定轴动态			M...	10
G621	所有角处拐角延迟	仅与轨迹控制运行一起	G621 ADIS=...	M...	57
G700	尺寸单位：英寸和英寸/分钟 (长度 + 速度 + 系统变量)			M...	13
G710 [1]	公制尺寸，毫米和毫米/分钟 （长度 + 速度 + 系统变量）			M...	13
G810[1], ..., G819	给OEM用户保留的G代码组				31
G820[1], ..., G829	给OEM用户保留的G代码组				32
G931	进给规定，通过运行时间	运行时间		M...	15
G942	冻结线性进给和恒定切削速度或者主轴转速			M...	15
G952	冻结转速进给和恒定切削速度或者主轴转速			M...	15
G961	恒定切削速度“开”	进给类型同G94时	G961 S...LIMS=...F...	M...	15
G962	线性进给或者转速进给和恒定切削速度			M...	15
G971	恒定切削速度“关”			M...	15
G972	冻结线性进给或者转速进给和恒定主轴转速			M...	15
GEOAX	给几何轴1 - 3分配新的通道轴	没有参数：MD-设定起作用			
GET	占用加工轴（n)	必须使用RELEASE在其它通道中释放轴			

GETD	直接占用加工轴（n)	参见GET			
GETACTT	从相同名称的刀具组中确定一个有效的刀具				
GETSELT	提供一个预选的T号				
GETT	给刀具名确定T号				
GOTOF	跳转指令，向前（程序结束方向）				
GOTOB	跳转指令，向后（程序开始方向）				
GOTO	跳转指令首先向前，然后向后（方向首先向程序结束处，然后向程序开始处				
GOTOC	抑制报警14080 跳转目标没有找到	参见GOTO			
GWPSOF	撤销选择恒定砂轮圆周速度(SUG)		GWPSOF (T-号)	s	
GWPSON	选择恒定砂轮圆周速度(SUG)		GWPSON (T-号)	s	

H...	辅助功能，到PLC	实数/整数	可以通过MD进行设置（机床制造商）	H100 或者 H2=100		
I [4]	插补参数	实数			s	
I1	中间点坐标	实数			s	
IC	增量尺寸输入	0, ..., ±99999.999°		X=IC(10)	s	
IDS	静态同步动作标记					
IF	引入一个有条件跳转		结构：IF-ELSE-ENDIF			
INDEX	确定输入端字符串中一个符号的索引	0, ..., INT	字符串：1. 参数，符号：2. 参数			
INIT	选择模块，用于在一个通道中的加工					
INT	数据类型：带符号的整数值	- (2^{31}-1), ..., 2^{31}-1				
INTERSEC	计算两个轮廓单元之间的交点	VAR REAL [2]	故障状态：BOOL			
IP	可变插补参数 (Interpolation parameter)	实数				
IPOBRKA	运行准则，自制动斜坡开始点		制动斜坡，在100 %到0%时	IPOBRKA=.. 或者 IPOBRKA(<轴>[,REAL])	M...	

IPOENDA	在到达“IPO中止”时运行结束			IPOENDA=.. 或者 IPOENDA[n]..	m	
IPTRLOCK	将无法查找的程序段的开始冻结到下一个机床功能程序段。		冻结中断指针		m	
IPTRUNLOCK	在中断时将无法查找的程序段的结尾设置成当前的程序段。		设置中断指针		m	
ISAXIS	检查被作为参数设定的几何轴是否为1	BOOL				
ISD	插入深度	实数			m	
ISFILE	检查在NCK用户存储器中是否有一个文件。	BOOL	提供一个布尔型结果	RESULT=ISFILE("Testfile") IF (RESULT==FALSE)		
ISNUMBER	检查是否可以把输入端字符串转换成数字	BOOL	转换输入端字符串为数字			
ISVAR	检查传送参数是否包含一个NC知晓的变量	BOOL	机床数据，调整数据和变量如GUD			
J [4]	插补参数	实数			s	
J1	中间点坐标	实数			s	
JERKA	激活通过MD设定的加速度特性，用于编程的轴					
K [4]	插补参数	实数			s	
K1	中间点坐标	实数			s	
KONT	在刀具补偿时绕行轮廓				m	17
KONTC	使用曲率不变的多项式逼近或者离开				m	17
KONTT	使用切线不变的多项式逼近或者离开				m	17
L	子程序号	整数，最大7个数		L10	s	
LEAD [5]	超前角	实数			m	
LEADOF	引导值耦合“关”					
LEADON	引导值耦合“开”					
LFOF [1]	螺纹切削中断“关”				m	41
LFON	螺纹切削中断“开”				m	41
LFTXT [1]	切向退刀时的刀具方向				m	46

LFWP	非切向退刀时的刀具方向				m	46
LFPOS	向某个位置轴向抬起				m	46
LIFTFAST	在调用中断程序之前快速退刀					
LIMS	在G96/G961和G97时转速极限（主轴速度极限）	0.001, ..., 99 999. 999			m	
LN	自然对数	实数				
LOCK	使用ID锁止同步动作 (停止工艺循环)					
LOG	（十位 - ）对数	实数				
LOOP	引入一个无限循环		结构：LOOP-ENDLOOP			
M...	开关操作	0, ..., 9999 9999	机床制造商可以指定最多5个未赋值的M功能			
M0 [10]	编程停止					
M1 [10]	可选停止					
M2 [10]	主程序程序结束，复位到程序开始					
M3	主轴右旋方向，用于主主轴					
M4	主轴左旋方向，用于主主轴					
M5	主轴停止，用于主主轴					
M6	换刀					
M17 [10]	子程序结束					
M19	主轴定位					
M30 [10]	程序结束，同M2					
M40	自动换档					
M41...M45	齿轮级1,…,5					
M70	过渡到轴运行					
MASLDEF	定义主/从轴连接					
MASLDEL	分离主/从轴连接，删除连接定义					
MASLOF	关闭一个临时耦合					
MASLOFS	使用自动制动从动轴，关闭一个临时耦合					
MASLON	接通一个临时耦合					
MCALL	模态子程序调用		没有子程序名称：撤销选择			
MEAC	连续测量，不带剩余行程删除	整数，无符号			S	
MEAFRAME	从测量点中计算框架	FRAME				

MEAS	用卡规测量	整数，无符号			S	
MEASA	测量，带剩余行程删除				s	
MEAW	用触发探针进行测量，不删除剩余行程 (measure without deleting distance to go)	整数，无符号			S	
MEAWA	测量，不带剩余行程删除				s	
MI	存取框架数据：镜像					
MINDEX	确定输入端字符串中一个符号的索引	0, ..., INT	字符串：1. Par., 字符:2. 参数			
MIRROR	可编程的镜像			MIRROR X0 Y0 Z0 ；自有程序段	s	3
MMC	从零件程序中交互式调用对话窗口MMC/HMI	字符串				
MOD	取模除法					
MOV	起动定位轴 (start Moving Positioning axis)	实数				
MSG	可编程的信息			MSG ("信息")	m	
N	程序段号 - 副程序段	0, ..., 9999 9999 仅整数，没有前置符	为了识别程序段可以使用编号； 位于程序段的开头	例如 N20		
NCK	规定数据有效区。		每个通道一次			
NEWCONF	接收修改的机床数据。根据机床数据设置有效。		也可以通过HMI软键			
NEWT	编制新的刀具		可以删除Duplo号			
NORM [1]	在刀具补偿时，在起始点和终点处正常设定				m	17
NOT	逻辑“非”					
NPROT	机床专用的保护区“开/关”					
NPROTDEF	定义某个特定机床的保护区 (NCK specific protection area definition)					
NUMBER	转换输入端字符串为数字		实数			
OEMIPO1[6,8]	OEM插补1				m	1
OEMIPO2[6,8]	OEM插补2				m	1

OF	CASE回路中的关键字					
OFFN	编程轮廓的加工余量			OFFN=5		
OMA1 [6]	OEM地址1	实数			m	
OMA2 [6]	OEM地址2	实数			m	
OMA3 [6]	OEM地址3	实数			m	
OMA4 [6]	OEM地址4	实数			m	
OMA5 [6]	OEM地址5	实数			m	
OFFN	偏移补偿 - 正常	实数			m	
OR	逻辑“或”					
ORIC [1,6]	在外角的方向改变叠加到待插入的圆弧程序段。				m	27
ORID [6]	方向改变在圆弧程序段之前执行				m	27
ORIAXPOS	虚拟的方向轴与回转轴位置的方向角				m	50
ORIEULER	欧拉角方向角				m	50
ORIAXES	线性插补加工轴或者方向轴		终点方向：矢量 说明 A3, B2, C2 附加参数：旋转矢量 A6, B6, C6 锥形的张角，单位为：度： 0<NUT<180 Grad 中间矢量：A7, B7, C7 刀具触点：XH, YH, ZH	参数如下： 方向矢量 标准化 A6=0, B6=0, C6=0 张角接在运行角之后，带有 NUT=... NUT=+... 在 ≤ 180度时 NUT= -... 在 ≥ 180度时 中间定向 标准化 A7=0, B7=0, C7=1	m	51
ORICONCW	顺时针方向圆弧表面插补				m	51
ORICONCCW	逆时针方向圆弧表面插补				m	51
ORICONIO	圆弧表面插补，说明一个中间方向				m	51
ORICONTO	在切向过渡中的某个圆侧面上的插补 (最终定向说明)				m	51
ORICURVE	定向插补，并指定刀具的两个接触点的运动				m	51
ORIPLANE	在某个平面中插补 (相当于 ORIVECT)				m	51
ORIPATH	刀具定向路径与轨迹有关		转换包处理，参见 /FB/, TE4		m	51
ORIRPY	欧拉角方向角				m	50
ORIS [5]	定向变化 （orientation smoothing factor定向平滑因数）	实数	与轨迹有关		m	
ORIVECT	大圆插补（和ORIPLANE一致）				m	51
ORIVIRT1	方向角，通过虚拟的方向轴（定义1）				m	50
ORIVIRT2	方向角，通过虚拟的方向轴（定义1）				m	50

ORIMKS [6]	机床坐标系中的刀具定向 (tool orientation in machine coordinate system)				m	25
ORIWKS [1,6]	工件坐标系中的刀具定向 (tool orientation in workpiece coordinate system)				m	25
OS	启用/关闭摆动 (Oscillating ON/OFF)	整数，无符号				
OSC [6]	恒定平滑修改刀具方向				m	34
OSCILL	给摆动进行轴赋值 - 激活摆动		轴：1 – 进给轴		m	
OSCTRL	选件摆动	整数，无符号			M	
OSB	摆动：起始点				m	
OSE	摆动：终点				m	
OSNSC	摆动：火花放电数 (oscillating:number spark out cycles)				m	
OSOF [1,6]	平滑修整刀具方向“关”				m	34
OSP1	摆动：左换向点 (oscillating:位置1)	实数			m	
OSP2	摆动：右换向点 (oscillating:位置2)	实数			m	
OSS [6]	在程序段结束处平滑刀具方向				m	34
OSSE [6]	在程序段开始处和结束处平滑刀具方向				m	34
OST1	摆动：在右换向点停止	实数			m	
OST2	摆动：在右换向点停止	实数			m	
OVR	转速补偿（倍率）	1, ..., 200%			m	
OVRA	轴向转速补偿（倍率）	1, ..., 200%			m	
P	零件程序运行次数	1, ..., 9999 整数，不带符号		例如L781 P... ;自有程序段		
PCALL	用绝对的路径参数和参数传送调用子程序		没有绝对路径 特性如同 CALL			
PDELAYOF [6]	关闭冲裁延迟 (punch with delay OFF)				m	36
PDELAYON [1,6]	开启冲裁延迟 (punch with delay ON)				m	36
PL	参数 - 间隔 - 长度	实数，无符号			S	

PM	每分钟			每分钟进给		
PO	多项式	实数，无符号			S	
POLF	位置 LIFTFAST	实数，无符号	在WKS中的几何轴，其它MKS	POLF[Y]=10 退回轴的目标位置	m	
POLFA	用 $AA_ESR_TRIGGER 启动单个轴的退回位置		用于单个轴	POLFA(AX1, 1, 20.0)	m	
POLFMASK	在轴之间**没有** 关系的情况下释放用于退回的轴		已选择的轴	POLFMASK(AX1, AX2, ...)	m	
POLFMIN	在轴之间**有** 关系的情况下释放用于退回的轴		已选择的轴	POLFMIN(AX1, AX2, ...)	m	
POLY [5]	多项式-插补				m	1
POLYPATH [5]	多项式插补可选择，用于轴组AXIS或者VECT			POLYPATH ("AXES") POLYPATH ("VECT")	m	1
PON [6]	冲压“开”				m	35
PONS [6]	冲压“开”，以IPO节拍				m	35
POS	轴定位			POS[X]=20		
POSA	轴定位，超出程序段界限			POSA[Y]=20		
POSP	在部分段中定位（摆动） (Position Axis in Parts)	实数：结束位置，部分长度； 整数：选件				
POT	平方（算术函数）	实数				
PR	每转			转速进给		
PRESETON	实际值设定，用于编程的轴		每次编程一个轴名称，并在下一个参数中编程相关值。 最多可以8个轴	PRESETON(X,10,Y,4.5)		
PRIO	在处理中断时设置优先级的关键字					
PROC	一个程序的第一个指令			程序段号 - PROC - 名称		
PTP	point to point；点对点移动				m	49
PUTFTOC	刀具精补偿，用于平行修整					
PUTFTOCF	PutFineToolCorrectionFunction-Dependent: 刀具补偿取决于一个使用 FCtDEF 为同时进行修整而设定的函数 (Continuous Dressing)					

PW	点 - 加权	实数，无符号			S	
QECLRNOF	关闭象限误差补偿 (Quadrant Error Compensation Learning OFF)					
QECLRNON	启用象限误差补偿 (Quadrant Error Compensation Learning ON)					
QU	快速辅助功能输出					
R...	计算参数 也作为可调式地址标记且带有数字扩展	±0.0000001, ..., 9999 9999	R参数个数可以通过MD设定	R10=3 ;R-参数赋值 X=R10 ;轴值 R[R10]=6 ;间接编程		
RDISABLE	读入禁止					
READ50	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。		信息为字符串			
READAL	读报警		报警按上升的号查找。			
REAL	数据类型：带有前置符的浮点变量 (reale Zahlen)	相当于处理器的 64-Bit 浮点格式				
REDEF	机床数据设定，NC语言单元和系统变量，在它们的用户组中显示					
RELEASE	加工轴使能		可以编程多个轴			
REP	关键字，用同一个值初始化一个数组的所有单元					
REPEAT	重复一个程序循环		直至 (UNTIL) 满足某个条件时为止			
REPEATB	重复一个程序行		nnn-次			
REPOSA	线性重新定位所有轴： 用所有的轴，线性逼近轮廓				s	2
REPOSH	半圆重新定位: 在半圆中重新进行轮廓逼近				s	2

REPOSHA	所有轴半圆重新定位: 所有轴重新进行轮廓逼近；半圆中的几何轴				s	2
REPOSL	线性重新定位： 轮廓的线性重新逼近				s	2
REPOSQ	Repositionierung quarter circle: 在象限中重新进行轮廓逼近				s	2
REPOSQA	所有轴以象限圆重新定位: 用所有轴重新进行线性轮廓逼近；象限中的所有几何轴				s	2
RESET	复位工艺循环		可以编程一个或者多个IDs			
RET	子程序结束		替代M17使用 - 没有函数发送给PLC	RET		
RINDEX	确定输入端字符串中一个符号的索引	0, ..., INT	String: 第1个参数，字符：2. 参数			
RMB	在程序段起始点重新逼近 (Repos mode begin of block)				m	26
RME	在程序段结束重新逼近 (Repos mode end of block)				m	26
RMI [1]	在中断点进行重新逼近 (Repos mode interrupt)				m	26
RMN	重新逼近最近的轨迹点 (Repos mode end of nearest orbital block)				m	26
RND	轮廓角倒圆	实数，无符号		RND=...	s	
RNDM	模态倒圆	实数，无符号		RNDM=... RNDM=0:关闭M. V.	m	
ROT	可编程旋转	旋转，围绕 第一几何轴： -180° ..180° 第二几何轴： -89.999°, ..., 90° 第三几何轴： -180° .. 180°		ROT X...Y...Z... ROT RPL= ;自有程序段	s	3

ROTS	可编程的框架旋转，带立体角			ROT X...Y... ROT Z...X... ROT Y...Z ROT RPL= ;自有程序段	s	3
ROUND	园整逗号后的数据	实数				
RP	极半径	实数			m,s [3]	
RPL	在平面中旋转 (rotation plane)	实数，无符号			S	
RT	用于存取框架数据的参数：旋转					
S	主轴转速或 （G4，G96/G961）其他意义	0.1, ..., 99999999.9	主轴转速的单位是转/分钟 G4：主轴旋转中的停留时间 G96/G961：切削速度，单位为：米/分钟	S...:转速，用于 主主轴 S1...:主轴1的速度	m,s	
SAVE	在子程序调用时保护信息		保护：所有模态G功能和当前框架			
SBLOF	抑制单程序段 (Single block OFF)		后面的程序段以单段执行，如同一个程序段。			
SBLON	取消单程序段抑制 (Single block ON)					
SC	用于存取框架数据的参数：刻度					
SCALE	可编程的比例（刻度）			SCALE X...Y...Z... ;自有程序段	s	3
SD	样条度	整数，无符号			S	
SEFORM	在步距编辑器中的结构指令，用于由此生成步距图，用于高级HMI		在步距编辑器中处理	SEFORM(<程序段名>, <平面>, <icon>)		
设定	关键字，用列表值初始化一个数组的所有单元					
SETAL	设置报警					
SETDNO	重新设置刀具（T）的D号以及其刀沿					
SETINT	确定在出现一个NCK输入端时应该激活哪一个中断程序		分析 Flanke 0 → 1			
SETM	设置一个/多个标号，用于通道协调		自身通道中的加工不受此影响。			

SETMS	转换回到机床数据中确定的主主轴					
SETMS (n)	主轴n应该作为主主轴					
SETPIECE	考虑用于所有刀具的数量，它们分配到主轴		没有主轴号：适用于主主轴			
SF	起始点偏置，用于螺纹切削	0.0000,..., 359.999°			m	
SIN	正弦（三角函数）	实数				
SOFT	限制冲击的轨迹加速度				m	21
SOFTA	接通限制冲击的轴加速度，用于编程的轴					
SON [6]	步冲“开”				m	35
SONS [6]	步冲“开”，以IPO节拍				m	35
SPATH [1]	FGROUP轴的轨迹基准为弧长。				m	45
SPCOF	主主轴或者主轴从转速控制转换到位置控制			SPCON SPCON (n)		
SPCON	主主轴或者主轴(n)从位置控制转换到转速控制			SPCON SPCON (n)		
SPIF1 [1,6]	快速NCK-输入/输出，用于冲切/步冲 字节1 (stroke/punch interface 1)		参见 /FB/, N4:冲裁和步冲		m	38
SPIF2 [6]	快速NCK-输入/输出，用于冲切/步冲 字节2 (stroke/punch interface 2)		参见 /FB/, N4:冲裁和步冲		m	38
SPLINE-PATH	确定样条连接		最多8个轴			
SPOF [1,6]	关闭冲程，冲裁，关闭剪切 (stroke/punch OFF)				m	35
SPN [6]	每个程序段的路径段数目 (stroke/punch number)	整数			s	
SPP [6]	路径段的长度 (stroke/punch path)	整数			m	
SPOS	主轴位置			SPOS=10 或者 SPOS[n]=10	m	
SPOSA	主轴位置超过程序段界限			SPOSA=5 或者 SPOSA[n]=5	m	
SQRT	平方根 (Square root; 算术函数)	实数				
SR	摆动退回行程 (sparking out retract path)，用于同步动作	实数，无符号			S	

SRA	外部输入的摆动退回行程，轴向 (sparking out retract)，用于同步动作			SRA[Y]=0.2	m	
ST	摆动修光时间 (sparking out time)，用于同步动作	实数，无符号			S	
STA	摆动修光时间轴向（轴向火花放电时间）用于同步动作				m	
START	从运行的程序中，在几个通道中同时启动所选择的程序		对自身的通道无效			
STAT	铰接位置	整数			s	
STARTFIFO[1]	执行加工；与此并行补足进刀缓存器				m	4
STOPFIFO	停止加工；补足进刀运行缓存器，直至STARTFIFO识别进刀缓存器已满或者程序结束				m	4
STOPRE	进刀停止，直至所有预备的程序段由主运行加工完毕					
STOPREOF	取消进刀停止					
字符串	数据类型：字符串	最多200个字符				
STRLEN	确定一个字符串的长度	INT				
SUBSTR	确定输入端字符串中一个符号的索引	实数	字符串：1.参数，符号：2.参数			
SUPA	抑制当前零点偏移		包括编程位移，手轮位移(DRF)，外部零点位移和PRESET-位移		s	9
SYNFCT	计算一个多项式，取决于运动同步动作中的一个条件	VAR REAL				
SYNR	同步读取变量，即在执行时(Synchronous Read)					
SYNRW	同步读写变量，即在执行时(Synchronous Read-Write)					
SYNW	同步写入变量，即在执行时(Synchronous Write)					

T	调用刀具 （机床数据中指定时才会改变；或者在需要 M6指令时）	1 ... 32 000	通过T编号调用： 或者通过刀具标志符：	例如 T3 或者 T=3 例如 T="BOHRER"		
TAN	正切（三角函数）	实数				
TANG	由两个所给定的引导轴确定切线，用于跟随运行					
TANGOF	关闭切向跟踪 (tangential follow up mode OFF)					
TANGON	启用切向跟踪 (tangential follow up mode ON)					
TCARR	需要刀柄（编号“m”）	整数	m=0:撤销选择有效的WZ刀架	TCARR=1		
TCOABS [1]	从当前刀具定向中确定刀具长度分量		装调之后需要，例如通过		m	42
TCOFR	从当前框架的方向确定刀具长度分量		手动设置		m	42
TCOFRX	在选择刀具时确定一个有效框架的刀具方向，刀具指向X方向		刀具垂直于斜面		m	42
TCOFRY	在选择刀具时确定一个有效框架的刀具方向，刀具指向Y方向		刀具垂直于斜面		m	42
TCOFRZ	在选择刀具时确定一个有效框架的刀具方向，刀具指向Z方向		刀具垂直于斜面		m	42
TILT [5]	侧向角	实数			m	
TMOF	撤销选择刀具监控		仅当具有该编号的刀具没有激活时才需要T编号。	TMOF (T-号)		
TMON	选择刀具监控		T号 = 0：取消所有刀具的监控	TMON (T-号)		
TO	表示FOR计数循环中的终点值					
TOFFOF	复位在线刀具长度补偿					
TOFFON	激活联机刀具长度补偿 (**T**ool **Off**set **ON**)		说明三维补偿方向	TOFFON (Z, 25) 补偿方向为 Z 偏移值为 25		

TOFRAME	设置当前的可编程的框架到刀具坐标系统	在刀具方向中旋转框架		m	53
TOFRAMEX	X轴平行于刀具方向，辅助轴Y,Z			m	53
TOFRAMEY	Y轴平行于刀具方向，辅助轴Z,X			m	53
TOFRAMEZ	Z轴平行于刀具方向，辅助轴X,Y			m	53
TOFROF	关闭刀具方向中的框架旋转			m	53
TOFROT	Z轴平行于刀具方向	启用框架旋转 已编程框架的旋转比例		m	53
TOFROTX	X轴平行于刀具方向			m	53
TOFROTY	Y轴平行于刀具方向			m	53
TOFROTZ	Z轴平行于刀具方向			m	53
TOLOWER	一个字符串的字母转换成小写字母				
TOWSTD	基本设定值，用于刀具长度补偿	考虑刀具磨损		m	56
TOWBCS	基本坐标系BKS中的磨损值			m	56
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	56
TOWMCS	机床坐标系MKS中的磨损值			m	56
TOWTCS	刀具坐标系中的磨损值（刀盘基准点T位于刀具夹持装置中）			m	56
TOWWCS	机床坐标系中的磨损值			m	56
TOUPPER	一个字符串的字母转换成大写字母				
TR	用于存取框架数据的参数：平移				
TRAANG	倾斜轴转换	每个通道可设置多个转换			
TRACEOF	圆弧格式测试：数值传送“关”				
TRACEON	圆弧格式测试：数值传送“开”				
TRACON	级联转换 (Transformation Concatenated)				
TRACYL	圆柱：表面转换	参见 TRAANG			

TRAFOOF	关闭转换			TRAFOOF( )		
TRAILOF	关闭轴同步联动 (Trailing OFF)					
TRAILON	启用轴同步联动 (Trailing ON)					
TRANS	可编程的偏移			TRANS X. Y. Z.;自有程序段	s	3
TRANSMIT	极坐标转换		参见 TRAANG			
TRAORI	4-, 5-轴转换，类转换 (Transformation oriented)		激活已约定的定向转换	转换类 TRAORI(1,X,Y,Z)		
TRUE	逻辑常量：真	BOOL	可通过 Integer型常量1替换			
TRUNC	去除小数点后位数	实数				
TU	轴交角	整数		TU=2	s	
TURN	螺旋线圈数	0, ..., 999			s	
UNLOCK	使能带ID的同步动作（继续工艺循环）					
UNTIL	结束一个REPEAT循环的条件					
UPATH	FGROUP轴的轨迹基准为曲线参数。				m	45
VAR	关键字：参数转让方式		用VAR：由基准调用			
WAITC	等候，直到满足轴/主轴的偶合程序段转换条件时为止 (wait for couple condition)		可以编程2个以下的轴/主轴	WAITC(1,1,2)		
WAITM	等待设定通道中的标记; vorhergeh.使用精确停止结束程序段			WAITM(1,1,2)		
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			WAITMC(1,1,2)		
WAITP	等待至运行结束			WAITP(X) ；独立的程序段		
WAITS	等待到达主轴位置			WAITS (主主轴) WAITS (n,n,n)		
WALIMOF	工作区域限制“关”			；独立的程序段	m	28
WALIMON[1]	工作区域限制“开”			；独立的程序段	m	28
WHILE	WHILE程序循环开始		结束：ENDWHILE			
WRITE	程序段写入到文件系统。在所说明文件的结束处加上一个程序段		这些程序段在M30之后插入			

X	轴	实数			m,s [3]	
XOR	逻辑“异 - 或”					
Y	轴	实数			m,s [3]	
Z	轴	实数			m,s [3]	

图例说明：

[1] 程序初始的默认设置（若没有另行编程，即为控制器的出厂设置）。

[2] 组编号相当于“G功能/位移条件列表”段落中的表格。

[3] 绝对终点：模态方式；增量终点：逐段方式；否则依据G功能语法规定为模态/逐段方式。

[4] IPO参数作为圆中心起增量作用。使用AC它们可以绝对编程。如果是其它含义（例如螺距）就忽略地址修改。

[5] 关键字对SINUMERIK 810D无效。

[6] 关键字对SINUMERIK 810D/NCU571无效。

[7] 关键字仅适用于SINUMERIK FM-NC

[8] OEM用户可以增加两个附加的插补类型。名称可以由OEM用户改变。

[9] 不允许将扩展式地址写入方式用于这些功能。

缩略符列表

A

A.1 缩略符

A	输出端
AS	可编程控制系统
ASCII	美国信息交换标准代码信息交换美国代码标准
ASIC	专用集成电路应用器件用户开关回路
ASUP	异步子程序
AV	工作准备部分
AWL	指令表
BA	工作方式
BAG	工作方式组
BB	运行准备
BuB, B&B	操作与编程
BCD	二 - 十进制二进制编码十进制数
BHG	操作设备
BIN	二进制文件**Bin**ary Files)
BIOS	基本输入输出系统
BKS	基本坐标系统
BOF	操作界面
BOT	引导文件SIMODRIVE 611D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	计算机数字控制计算机数字控制系统
COM	通讯
CP	通讯处理器
CPU	中央处理单元中央处理单元
CR	回车
CRT	阴极射线管阴极射线管
CSB	中央服务板PLC 组件
CTS	清除发送通过串行数字接口通知准备就绪

CUTOM	Cutter radius compensation:刀具半径补偿
DAU	数字模拟转换器
DB	PLC中数据块
DBB	PLC中数据块字节
DBW	PLC中数据块字
DBX	PLC中数据块位
DC	直接控制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	数据输入/输出数据传送显示
DIR	路径路径
DLL	动态连接程序库
DOE	数据传送设备
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	DRF功能直接测量功能（手轮）
DRY	空运行空运行
DSB	单段译码单段译码
DW	数据字
E	输入端
E/A	输入/输出
E/R	SIMODRIVE 611(D) 的馈电/再生反馈模块（电源）
EIA 代码	专门穿孔带编码，每个字符的穿孔数始终为奇数
ENC	编码器实际值编码器
EPROM	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	功能调用：PLC中功能块
FDB	生产数据库
FDD	软盘驱动器
FEPROM	闪存EPROM读写存储器
FIFO	先进先出 不使用设定地址工作并且其数据按照保存的顺序读取的存储器。
FIPO	精插补
FM	功能模块
FM-NC	功能模块数字控制
FPU	浮点单位浮点单位

FRA	框架模块
FRAME	数据块
FRK	铣削半径补偿
FST	进给停止进给停止
FUP	功能图（PLC编程方法）
GP	主程序
GUD50	全局用户数据全局用户数据
HD	硬盘硬盘
HEX	十六进制数代号
HiFu	辅助功能
HMI	人机对话接口： SINUMERIK用于操作、编程和模拟的操作功能。 HMI 的含义与MMC相同
HMS	高分辨率测量系统
HSA	主轴驱动
HW	硬件
IBN	开机调试
IF	驱动模块脉冲使能
IK (GD)	隐含通讯（全局数据）
IKA	可插补补偿可插补补偿
IM	接口模块接口模块
IMR	接收方接口模块接收方接口模块
IMS	发送方接口模块发送方接口模块
INC	增量方式增量方式
INI	初始化数据初始化数据
IPO	插补器
ISA	国际标准体系
ISO	国际标准组织
ISO 代码	专门穿孔带编码，每个字符的穿孔数始终为偶数
JOG	手动工作方式手动工作方式
K1 ..K4	通道1到通道4
K-Bus	通讯总线
KD	坐标旋转
KOP	功能图（PLC编程方法）
K_v	回路放大系数
$K_Ü$	传动比
LCD	液晶显示液晶显示
LED	Light-Emitting Diode:发光二极管
LF	线路馈电
LMS	位置测量系统
LR	位置调节器
LUD	局部用户数据
MB	兆字节

MD	机床数据
MDA	手动数据输入：手动输入，自动运行
MK	测量回路
MKS	机床坐标系
MLFB	机器可识别产品符
MMC	人机通讯操作、编程和模拟运行界面
MPF	主程序文件NC零件程序（主程序）
MPI	多端口接口多端口接口
MS-	微软（软件制造商）
MSTT	机床控制面板
NC	数字控制数字控制
NCK	数字控制核心数字控制核心
NCU	数字控制单元NCK硬件单元
NRK	NCK操作系统名称
NST	接口信号
NURBS	非均匀有理B样条
NV	零点偏移
OB	PLC中组织块
OEM	原设备制造商
OP	操作面板操作面板
OPI	操作面板接口操作面板接口
OPT	选件选件
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	个人计算机存储卡国际协会存储器插卡标准
PCU	PC Unit: PC-Box (计算机单元)
PG	编程器
PLC	可编程逻辑控制器可编程逻辑控制器
POS	定位
RAM	随机存取存储器程序存储器，可读写
REF	回参考点运行
REPOS	再定位功能
RISC	简化的计算机指令系统处理器类型，具有较小的指令组、快速的指令处理能力
ROV	快速倍率快速倍率
RPA	R参数有效存储器范围 NCK中用于R参数号

RPY	旋转定位移动一种坐标系旋转方式
RTS	请求发送开启发送方，控制信号自串行数据接口
SBL	单段单段
SD	设定数据
SDB	系统数据块
SEA	设定数据有效设定数据符号（文件类型）
SFB	系统功能块
SFC	系统功能调用
SK	软键
SKP	程序段跳跃程序段跳跃
SM	步进电机
SPF	子程序文件子程序
SPS	存储器可编程控制
SRAM	静态存储器（缓存）
SRK	刀尖补偿
SSFK	丝杠螺距误差补偿
SSI	串行同步接口串行同步接口
SW	软件
SYF	系统文件系统文件
TEA	测试数据有效机床数据标志
TO	刀具补偿刀具补偿
TOA	刀具补偿有效刀具补偿符号（文件类型）
TRANSMIT	铣削转换为车削铣削加工中车床坐标系换算
UFR	用户框架零点偏移
UP	子程序
VSA	进给驱动
V.24	串行接口（DEE和DUE之间数据交换定义）
WKS	工件坐标系
WKZ	刀具
WLK	刀具长度补偿
WOP	现场编程
WDP	工件目录工件目录
WRK	刀具半径补偿
WZK	刀具补偿
WZW	换刀
ZOA	零点偏移有效零点偏移数据符号（文件类型）
µC	微米级控制器

词汇表

A样条

Akima样条始终以编程的支点进行切线移动（三项式）。

B 样条

在B样条中，编程的位置不是支点，而仅仅是“控制点”。产生的曲线不是直接经过控制点，而仅仅是在它们的附近（可选择一级、二级或三级多项式）。

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

CNC 标准语言提供： -> 用户自定义变量， -> 系统变量， -> 宏技术。

CNC编程语言

CNC编程语言的基础是DIN66025,带高级语言扩展。此外，CNC高级语言和编程语言允许使用宏指令定义（单个指令的汇编）。

COM

NC控制系统部件，用于执行和和协调通讯。

CPU

Central Processor Unit（中央处理单元），参见 -> 可编程控制系统

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。使用三级多项式。

DRF

DRF功能NC功能，在自动方式下利用电子手轮产生增量式零点偏移。

HIGHSTEP

AS300/AS400系统中PLC所有编程方法的汇编。

Jog方式

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，再定位以及预设定（设定实际值）。

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

数字控制NC 控制装置包括所有机床控制装置的组件： -> NCK, -> PLC, MMC/HMI, -> COM.

注意

对于SINUMERIK 840D控制系统而言，称作CNC控制系统更为贴切：计算机数控系统。

NCK

数字控制核心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

系统内部的运动控制和轨迹插补根据NURBS(Non Uniform Rational B-Splines)进行。
因此，在 SINUMERIK 840D 控制系统内部有一个统一的方法可供所有插补使用。

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。

PG

编程器

PLC

可编程逻辑控制器存储器可编程控制 NC的组件：用于执行机床控制逻辑的转接控制。

PLC-编程

使用软件 **STEP 7** 对PLC 进行编程。 编程软件 STEP 7 基于**WINDOWS**
标准操作系统，并且含有 STEP 5 使用创新技术进行编程的功能。

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。
PLC用户存储器可以通过存储器扩展至96K字节。

REPOS

1. 通过操作再次逼近轮廓

   使用函数 REPOS 可以借助方向键重新逼近到中断点。

2. 通过操作再次逼近轮廓

   通过程序指令可选用多种逼近方法：
   逼近中断点、逼近程序段起始点、逼近程序段结束点、逼近程序段开始点和中断点之间的某个轨迹点。

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。

S7-300 总线

S7-300总线是一个串连数据总线，通过该数据总线模块可以相互进行通讯，同时总线自身提供电源。模块之间的联系通过总线连接器建立。

S7配置

S7配置是一个工具，用此工具可以给模块设定参数。
使用S7配置在PG上划分CPU和外设模块的各种参数块。这些参数传送到CPU中。

SPS

参见 -> 可编程控制系统

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）的加工过程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RS232

为了输入输出数据，

- 在MMC模块MMC100上有一个串行接口RS232，
- 在MMC模块MMC101和MMC102中有两个RS232接口。

可用。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工作流程所需要的所有数据。

主轴

主轴功能分为两种功率级别：

1. 主轴： 转速或者位置控制式主轴驱动装置，数字式 (SINUMERIK 840D)
2. 辅助主轴： 转速控制式主轴驱动装置功能包“辅助主轴”，例如用于被驱动的刀具。

仿真器模块

仿真器模块是一种模块，

- 通过操作部件可以模拟数字输入量，
- 显示数字式输出值。

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”。

全局主程序/子程序

在一个目录下每个全局主程序/子程序只可以出现一次，不可以在不同目录下有不同内容的程序具有相同的程序名。

公制测量系统

单位均为公制： 例如长度mm（毫米），m（米）。

关键字

有确定写法的字，它们在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。
为了减少接近时间，给快速运动和进给定义 -> 准停界限。

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

描述工件坐标系中的某个工件。

几何轴

几何轴用来描述工件坐标系中的一个二维或者三维区间。

刀具

机床中进行加工的部件，诸如车刀、铣刀、钻头、激光...

刀具半径补偿

为了能够直接编程某个所需的工件轮廓，控制系统必须在考虑所使用刀具半径的情况下，以等同于所编程轮廓的轨迹执行运动(G41/G42)。

刀具补偿

在程序段中编程一个 **T功能** (5个十进制整数)即可实现刀具选择。每个T号可以最多有9个刀沿（D地址）。控制系统中所管理的刀具数量可以通过设计进行修改。

刀尖补偿

在编程一个轮廓时，往往从刀具的尖端计算。
因为实际上无法实现这一点，所以要将所使用刀具的曲率半径告知控制系统并且由其进行考虑。在此计算的加工点就位于其中心点，距离为半径的长度。

初始化文件

对应于每个工件可以编制一个初始化文件。在初始化文件中可以编制不同变量的赋值指令，它们仅适用于一个工件。

初始化模块

初始化模块是专用的程序模块。它包含在程序处理之前须执行的赋值。初始化模块主要用于初始化预定义的数据或者全局用户数据。

剩磁

剩磁是指数据块中的数据区以及定时器、计数器和标志位在新启动时，或者在掉电时不会丢失。

加工

系统操作区。

加工空间

用加工空间定义一个三维空间，在此空间内刀尖可以移动。 参见 -> 保护空间。

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，带冲击限制

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突变式加速度和平缓式加速度之间进行转换。

参数

- S7-300: 我们将参数分成两类：
  - STEP 7 指令参数

    某个 STEP 7 指令的参数是需要进行处理的运算对象的地址或者是一个常量。
  - 一个参数块的参数

    一个参数块的参数用来确定某个模块的属性。

- 840D:
  - 系统操作区。
  - 计算参数，零件程序的编程员可以为各种目的在程序中任意设置或者查询这些参数。

参考点

加工轴的测量系统作为参考用的机床上的点。

反比时间进给

在SINUMERIK840D中，轴运动不是编程进给速度，而是编程一个程序段轨迹位移所需要的时间（G93）。

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

借助插补式补偿可以视加工条件而定，对丝杠导程误差和测量系统误差进行 补偿（SSFK, MSFK）。

可编程的工作范围

将刀具的运动空间限制到一个由编程好的界限所定义的空间中。

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

   在加工工件的过程中，可以从CNC程序出发，将工艺函数（辅助函数）发送给PLC。例如，通过这些辅助函数来控制机床的辅助装置，如顶尖套筒、抓手、卡盘等等。

2. 快速输出辅助函数

   对于时间比较紧迫的开关函数而言，可以将辅助函数的应答时间降低到最低程度，并且可避免在加工过程中出现不必要的停止点。

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。

回参考点

如果所使用的行程测量系统没有绝对值编码器，就需要进行找零运行来确保测量系统所发送的实际值与机床坐标系参数相符。

回机床固定点

返回到预定义的机床固定点。

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。

回转轴无限旋转

根据具体的应用场合，回转轴可以旋转小于360度，或者在两个方向无限旋转。无限旋转的回转轴，比如可以用于非圆加工、磨削加工和绕线加工。

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

型材导轨

型材导轨用于固定S7-300的模块。

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。

基本坐标系统

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，程序员使用基本坐标系的轴名称。 当没有 -> 转换激活时，就等于 -> 机床坐标系。不同点在于轴名称。

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。

增量方式

通过相对尺寸说明加工行程。 增量数可以作为调整数据保存，或者通过相应的标识有10, 100, 1000, 10 000的按键来选择。

备份

存储器内容存储到外部存储设备中。

备份存储器

备份存储器保证存储器存储区的缓冲状态下工作，CPU没有缓存电池。定时器、计数器、标志和数据字节数可以设定参数并缓存。

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
- 仿真器模块

外部零点偏移

由PLC给定的零点偏移。

多端口接口

多点接口 (MPI) 是一种9针 D-Sub间接口。通过多端口接口可以连接一系列设备，相互可以进行通讯：

- 编程器
- 操作与监控系统
- 其它可编程控制器

CPU的参数块"Multipoint Interface MPI"含有 -> 用来规定多点接口属性的参数。

多项式-插补

使用多项式插补可以生成迥然各异的曲线，如**直线函数、抛物线函数、势函数** (SINUMERIK 840D).

子程序

一个子程序的连续指令，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。没有授权的读取和显示会被子程序禁止。循环是子程序的一种。

存储器可编程控制

存储器可编程的控制系统（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。存储器可编程的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。

存取权限

CNC程序块和数据通过一个7级存取权限进行保护。

- 三个口令字，分别用于系统生产厂家、机床制造商和用户，以及
- 4个钥匙开关位置，可以由PLC进行利用

存档

从数据和/或者目录中读出给一个 **外部**存储设备。

安全功能

系统中所具有的安全监控功能，通过安全监控功能数控系统中以及PLC和机床中的故障均可以尽早地予以识别，从而排除一切对工件、刀具或者机床可能造成的危害。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（比如刀库，托盘运输）。（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

比如主轴在一给定角度位置停止，从而可以在某一固定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，它与编程的尺寸系统（G70/G71）无关。

工件

机床待加工的零件。

工件坐标系

工件坐标系的初始点即为工件的零点。
在工件坐标系中进行编程时，尺寸和方向均以该坐标系为参照。

工件轮廓

需要建立/加工的工件的额定轮廓。

工件零点

工件的零点构成工件坐标系的初始点。它由与机床零点的距离定义。

工作区域限制

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，在程序加工期间处理器可以对用户程序进行存取。

工作方式

SINUMERIK 控制系统的运行过程控制方式。它们是下面几种工作方式：Jog（手动运行方式），MDA（手动输入，自动运行方式），自动方式。

工作方式组

在某一时刻将所有轴/主轴准确分配给某个通道。
每个通道均分配有一个工作方式组。同一个工作方式组中的通道均有相同的工作方式。

工具

一个工具是指用于输入和修改参数组参数的软件工具。 主要工具有：

- S7配置
- S7-TOP
- S7-信息

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）的子程序。

引导

上电后装载系统程序。

循环

受到保护的子程序，用来执行一个在工件上重复出现的加工过程。

循环辅助

在“程序”操作区菜单“循环”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。

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）。
通过数字式CNC输出，可以通过程序控制方式迅速触发开关功能 (SINUMERIK 840D)。

快速离开工件轮廓

当出现某个中断时，可以通过CNC加工程序启动某种运动，使得可以从刚刚加工好的工件轮廓上迅速抬起刀具。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。

总线连接器

总线连接器是一种 S7-300-配件，可以与 -> 外设模块一起供货。通过总线连接器，S7-300总线可以从CPU或者一个外设模块扩展到其相邻的外设模块。

成品轮廓

成品工件的轮廓。 参见 -> 坯件。

报警

所有的信息和报警均在操作面板上显示其文本，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。

1. 零件程序中的报警和信息

   机床的报警和信息可以直接从PLC程序中以纯文本的形式显示出来。

2. PLC的报警和信息

   机床的报警和信息可以直接从PLC程序中以纯文本的形式显示出来。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

系统操作区。

操作界面

操作界面（BOF）是CNC控制系统的显示形式，带屏幕。
操作界面设计有水平和垂直排列的功能键。

攻丝，不带补偿衬套

用此功能可以不带补偿衬套攻丝螺纹。
通过作为旋转轴的主轴和钻削轴的插补运动将螺纹精确切削到最终孔深，例如盲孔螺纹（前提条件：主轴作为进给轴运行）。

数字量输入/输出模块

数字量模块用于二进制过程信号的处理。

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，比如零件程序、刀具补偿等等。PCIN程序可以在标准工业计算机中MSDOS下运行。

数据块

1. 数据单元，PLC可以对HIGHSTEP程序进行存取。
2. 数据块->NC：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

在数据块中两个字节大小的数据单位。

文本编辑器

参阅 -> 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。

机床固定点

通过机床唯一定义的点，例如基准点。

机床坐标系

以机床轴为基准的坐标系。

机床控制面板

机床中具有各个操作按键、旋钮开关以及各个显示单元如 LEDs 的控制面板，它们通过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。

极限速度

最大/最小（主轴）速度：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
- 适用于车削

在“程序”操作区菜单“循环”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。

框架

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。 框架含有组件 -> 零点偏移, -> 旋转, -> 缩放, -> 镜像。

模块

模块是指编程和程序执行时所需要的所有文件。

模拟量输入/输出模块

模拟输入输出模块是模拟处理信号的信号生成器。

模拟输入模块将模拟测量值转换成可在CPU中进行处理的数字值。

模拟量输出模块用来把数字量数值转换为模拟量的调节参数。

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

用来加工一个工件的零件。

波特率

数据传送时的速度（位/秒）。

测量回路

SINUMERIK 840D: 测量编码器分析功能在 SIMODRIVE 611D-驱动模块中。
最大配置可以达到8个进给轴和主轴，其中最多允许5个主轴。

清零

清零时会删除CPU的下列存储器：

- 工作存储器
- 装载存储器的读写区
- 系统存储器
- 备份存储器

用户号码

如果几个用户通过网络进行通讯，则用户号码表示一个CPU或者编程器的“动作地址”，或者一个其它的智能外设模块的“动作地址”。 使用S7-工具"S7-配置" 将设备编号分配给CPU或者PG。

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。

基本模块类型有：
代码模块： 代码模块：
数据模块：该模块含有用于STEP7程序的常量和变量。

电子手轮

利用电子手轮可以在手动运行状态运行所选择的轴。
通过外部零点偏移增量加权来确定手轮的分度条纹加权。

示教

使用**示教**功能可以建立或者修正零件程序。各个程序段可以通过键盘输入，并可立即运行。通过方向键或者手轮运行的位置也可以存储。附加数据，如G功能、进给率或者M概念可以输入到同一个程序段中。

程序

1. 系统操作区。
2. 对控制系统发出指令的顺序

程序块

程序块含有零件程序的主程序和子程序。

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。

程序段搜索

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

无需编程员的工作，已经存在的变量。 系统变量是通过某种数据类型和以字符**$** 开头的变量名称来定义的。 参见 -> 用户自定义变量。

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。

线性轴

与回转轴相反，线性轴指按直线运行的轴。

结构技术

SINUMERIK 840D 可作为紧凑型模块按照在SIMODRIVE 611D变频系统中。尺寸与一个50毫米宽的SIMODRIVE 611D模块一致。SINUMERIK 840D 模块由NCU模块和NCU盒组成。

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

零件程序的编程语言中具有规定含义的字符和字符顺序（参见编程说明）。

编辑器

利用编辑器可以进行程序/文本/程序段的编辑、修改、合并和插入。

网络

网络指通过连接电缆连接几个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。

英制尺寸系统

长度以“英寸”及其导出单位为尺寸的测量系统。

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：

1. 平面中的回转运动
2. 垂直于该平面的线性运动

补偿值

测量传感器所测得的轴位置与所要求的、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

如果是PLC的CPU 314，则装载存储器就等于工作存储器。

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。

轨迹控制运行

轨迹控制运行的目的在于：避免在零件程序的程序段结束处轨迹轴产生较大的制动，影响系统、机床以及运行和用户参数值，从而尽可能地以相同的轨迹速度更换到下一个程序段。

轨迹轴

轨迹轴是 -> 通道的所有加工轴，可由-> 插补器引导，使其同时启动、加速、停止和到达终点。

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

在某个直角坐标系中进行编程，在某个非直角坐标系中（例如加工轴为旋转轴）进行处理。

轮廓

工件的轮廓

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。

轮廓破坏预先识别

控制系统识别和通报以下的轮廓冲突情形：

1. 轨迹行程短于刀具半径。
2. 内角的宽度小于刀具直径。

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。每个轴可以给定两组数值，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。

轴名称

进给轴根据DIN 66217标准中右向旋转直角坐标系命名：X,Y,Z

围绕X、Y、Z旋转的回转轴命名为A、B、C。其它平行的进给轴可以用其它地址字母标识。

参见 -> 轴标识符

轴名称

进给轴根据DIN 66217标准中右向旋转直角坐标系命名：X,Y,Z

围绕X、Y、Z旋转的回转轴命名为A、B、C。其它平行的进给轴可以用其它地址字母标识。

参见 -> 轴标识符

轴地址

参见 -> 轴标识符

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

通过“N”引导的程序段，包含一个加工步骤的信息，比如一个位置说明。

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单位和位置控制单位，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。
  辅助轴不参与实际的加工，例如进刀滑台、刀库。

连接电缆

连接电缆是预装配式或者用户自行装配的、带有两个连接插头的双线导线。连接电缆通过多端口接口（MPI）把CPU与编程器或者其它CPU相连。

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。

通道结构

利用通道结构可以同时/分开加工各个通道的程序。

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见功能（->Look Ahead）。

钥匙开关

1. S7-300:钥匙开关是CPU的工作方式开关。钥匙开关的操作通过一个可以插拔的钥匙进行。
2. 840D:
   机床控制面板上的钥匙开关共有四个被控制系统的操作系统赋予功能的位置。钥匙开关有3个不同颜色的开关，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

补偿机械加工间隙，例如滚珠丝杠主轴的反向间隙。对于每个轴，可以分别输入间隙补偿。

零件程序

发送给NC控制系统的指令顺序，在这些指令的共同作用下生产出某个工件。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. 可设置

   SINUMERIK 840D:
   有一些可设计的可设置零点偏移可供每个CNC轴使用。通过G功能可选择的偏移可以选择性地使用。

2. 外部

   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

   使用指令 TRANS 可以给所有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。

预见功能

使用**前瞻**功能可通过“预先查看”一些可编程的运动程序段，达到优化加工速度的目的。

预设定

使用预设定功能可以在机床坐标系中重新定义系统的零点。在预设定中轴没有运动，它仅仅给当前轴的位置输入一个新的位置值。

驱动

数控系统SINUMERIK 840D 通过一个快速数字并行总线与变频系统SIMODRIVE 611D相连。

索引

B

C

D

E

F

G

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

三

上

中

主

交

从

优

传

伺

位

侧

保

修

倍

值

倾

停

偶

关

内

再

冲

刀

分

切

列

初

删

剩

单

压

发

取

变

可

同

向

周

咬

圆

在

复

外

多

夹

子

字

存

学

宏

定

实

框

模

欧

步

比

求

测

激

生

用

电

确

程

站

端

等

边

运

返

进

退

选

逐

通

逻

部

采

重

铣

链

间

阻

除

零

非

预

额

驱

收件人 SIEMENS AG A&D MC BMS Postfach 3180 D-91050 Erlangen, Germany 电话： +49 (0) 180 5050 – 222 [热线] 传真：+49 (0) 9131 98 – 2176 [文档] 电子邮件：motioncontrol.docu@erlf.siemens.de	建议 更正
	出版物/手册： SINUMERIK 840D/840Di/810D 工作准备部分 用户文档
此信来自 姓名 公司/部门 地址： 邮政编码： 城市： 电话： / 传真： /	编程说明 订货号： 6FC5 298-7AB10-0RP1 版本 10.04 您在阅读本出版物时，如果遇到任何印刷错误，请使用该表格通知我们。同时欢迎您提出改进建议。

建议和/或更正

Overview of SINUMERIK 840D/840Di/810D Documentation (10.2004)

General Documentation

- *SINUMERIK* 840D/810D — Brochure
- *SINUMERIK* 840D/840Di/810D/ — Catalog Ordering Info. NC 60 *)
- *Safety Integrated* Application Manual — **Safety Integrated** Application Manual

User Documentation

- *SINUMERIK* 840D/810D/FM-NC — **AutoTurn**
  - Short Guide
  - Programming/ Setup
- *SINUMERIK* 840D/840Di/810D — Operator's Guide **– HT 6**
- *SINUMERIK* 840D/840Di/810D — **Diagnostics Guide *)**
- *SINUMERIK* 840D/840Di/810D — **Operator's Guide *)**
  - Short Guide
  - HMI Embedded
  - HMI Advanced

User Documentation

- *SINUMERIK* 840D/840Di/810D — **Programming Guide**
  - Short Guide
  - Fundamentals *)
  - Advanced *)
  - Cycles
  - Measuring Cycles
  - ISO Turning/Milling
  - Lists **System Variables**
- *SINUMERIK* 840D/810D — Operator's Guide
  - **ManualTurn**
  - Short Guide ManualTurn
  - **ShopMill**
  - Short Guide ShopMill
  - **ShopTurn**
  - Short Guide ShopTurn
- *SINUMERIK* 840Di — **System Overview**

Manufacturer/Service Documentation

- *SINUMERIK* — **Configuring** (HW) *)
  - 810D
  - 840D
- *SINUMERIK* 840D/840Di/810D — **Operator Components** (HW) *)
- *SINUMERIK* 840D/840Di/810D — Description of Functions
  - **ManualTurn**
  - **ShopMill**
  - **ShopTurn**

Manufacturer/Service Documentation

- *SINUMERIK* 611D 840D/810D — Description of Functions **Synchronized Actions**
- *SINUMERIK SIMODRIVE* 840D/840Di/810D — Description of Functions **Drive Functions *)**
- *SINUMERIK* 840D/840Di/810D — **Description of Functions**
  - Basic Machine *)
  - Extended Functions
  - Special Functions
- *SINUMERIK* 840D/810D — Description of Functions **Tool Management**
- *SINUMERIK* 840D/810D — **Configuring Kit** HMI Embedded
- *SINUMERIK* 840D/810D — Description of Functions Operator Interface **OP 030**
- *SINUMERIK* 840D/810D — **MCIS**
  - Computer Link
  - Tool Data Inform. Syst.
  - NC Data Management
  - NC Data Transfer
  - Tool Data Communication

Manufacturer/Service Documentation

- *SINUMERIK SIMODRIVE* — Description of Functions SINUMERIK **Safety Integrated**
- *SINUMERIK* 840D — Description of Functions **Digitizing**
- *SINUMERIK SIMODRIVE* — **Installation & Start-Up Guide *)**
  - 810D
  - 840D/611D
  - HMI
- *SINUMERIK SIMODRIVE* 840D/840Di 810D 611D — **– Lists *)**
- *SINUMERIK SIMODRIVE* 840D 611D — Description of Functions **Linear Motor**
- *SINUMERIK SIMODRIVE* 840D 611D — Description of Functions **– Hydraulics Module** **– Analog Module**
- *MOTION CONTROL SYSTEMS* — **EMC** Guidelines

Electronic Documentation

- *SINUMERIK SIMODRIVE* 840D/840Di/810D 611, Motors — **DOC ON CD *)** The SINUMERIK System

Manufacturer/Service Documentation

- *SINUMERIK* 840D/840Di/810D — Description of Functions **ISO Dialects** for SINUMERIK
- *SINUMERIK* 840Di — **Manual** (HW + Installation and Start-Up)
- *SINUMERIK* 840D/810D — Description of Functions **Remote Diagnosis**
- *SINUMERIK* 840D/840Di/810D — **Manual @ Event**

*) These documents are a minimum requirement

Siemens AG
Automation & Drives
Motion Control Systems
Postfach 3180
91050 ERLANGEN
德国

www.siemens.com/motioncontrol

订货号: 6FC5 298-7AB10-3RP1

在德国印刷