

§ 7 PMAC 编程语言描述

{坐标}{数值} [{坐标}{数值}...]

功能: 位置移动的定义。

类型: 运动程序。(PROG and ROT)

语法: {坐标}{数值} [{坐标}{数值}...]

其中:

{坐标}定义了是哪个轴 (X, Y, Z, A, B, C, U, V, W) ;

{数值}是一个常量(没有括弧),或是表达式(在括弧内)代替最后 的位置或距离, [{轴}{数值}...]多轴同步运动定义。

注释: 以上是基本的PMAC移动描述语句。它包含多个坐标轴及其相关值。对应坐标的值是标准的。(单位由坐标定义语句决定) ;

如果是绝对坐标,它代表着位置,若坐标处于增量模式它则代表距离。在两种模式中命令是一样的。

该命令表明哪一轴移动;没说明如何移动。如何运动定义应在另外的程序语句在运动之前定义

给定运动命令引起相应运动取决于运动的模式及移动起始时系统的情况。

例子 x1000
 (P1+P2)
 y(Q100+500) z35 c(P100)
 A(P1) B(P2) C(P3)
 x(Q1*sin(Q2/Q3)) U500

另参考 坐标轴定义语句 (建立一个坐标系)
运动编程语句 LINER, CIRCLEn,RAPID,SPLINE1,PVT
运动编程语句 TA,TS,TM,F,ABS,INC
I-变量 lx87,lx88,lx90.

{坐标}{数值} : {数值} [{坐标}{数值} : {数值}...]

功能: 位置及速度运动的定义。

类型: 运动编程(PROG and ROT)

语法: {坐标}{数值} : {数值} [{坐标}{数值} : {数值}...]

其中

{坐标}说明了是哪个轴{X,Y,Z,A,B,C,U,V,W}

{数值}是一个常量(没有括弧),或是表达式(在括弧内)代替最后的位置或距离。

:{数值}代表最终速度。

[[坐标]{数值}:{数值}...]类似地说明了多轴的同步运动情况。

注释: PVT(位置,速度,时间)运动模式中,每个轴的语句都说明最后的位置及速度。由坐标及两个数字构成基本语句,一个命令语句中可包含一个或多个并由中括弧分开。每个坐标的前一数值表明最终的位置或位移(取决是绝对坐标模式还是相对坐标模式。位置的单位由轴定义语句确定),第二个数值是终速度。

速度单位是由位置轴比例决定,由X轴除以时间轴确定。速度是有单位的量,而非标量。看运动程序部分PVT模式的例子。

执行PVT或TA命令,需圆整为毫秒。

在PVT模式中,若语句没给出速度,PMAC认为末速度为零。

例子 X100:50
X100:-32 z737.2:68.93
A(P1+P2):(P3) B(SIN(Q1)):0

参考 PVT模式的运动(编写的运动语句)
轴定义说明语句(建立一个坐标轴系统)
I变量 lx87,lx90
运动程序命令PVT,TA

{坐标}{数值}^ {数值}[[{坐标}{数值}^ {数值}...]

功能: 运动至触发

类型: 运动程序

语法: {坐标}{数值}^ {数值}[[{轴}{数据}^ {数据}...]

其中

{坐标}说明了是哪个轴{X,Y,Z,A,B,C,U,V,W}

第一个{数值}是一个常数(没有括弧),或表达式(括弧内),存在触点的位置或距离。

第二个{数值}(在^后)是一个常数(没有括弧),或表达式(括弧内),表示不代表到触点的距离。

[[坐标]{数值}^ {数值}...]类似说明了多轴同步的运动。

注释 在RAPID运动模式中,该移动说明语句允许move-until-trigger功能。“^”标志前轴的描述部分说明没有触点的情况下移动到哪里。在绝对坐标系里为位置,在相对坐标系为距离。两种情况下单位取决于用户坐标系单位。到达目标前没有发现触点则为标准的RAPID移动。

“^”后轴的描述部分说明有触点情况下,从触点位置到结束post-trigger运动的距离。距离用用户坐标系单位表示。

命令中所定义的轴向的电机执行一个独立的move_until_trigger运动。所有指定的电机同时运动,每一个有其独立的触发条件。如果需要共同的触发条件触发信号需连到所有电机的接口。每个电机可在不同时间完成任务;直到所有电机都执行完后,程序的下一行才继续执行。运动不可已混杂。

若lx03的17位为0电机的触发可由硬件输给触发器。或lx03的17位为1,则由电机在出错状态下的警告产生。(这种情况下lx03的16位被置为1)若使用硬件,

lx25描述的状态标志Encoder/Flag I变量2,3(例I902,I903) 确定哪个状态量的哪个沿触发。若警告跟随误差用作“力矩极限”触发，lx12确定了出错后警告的范围。

触发前或后的移动的速度，若I50为0，由lx22设定；I50为1，则由lx16设定。加速度由lx19至lx21设定。

在同一行，同时执行的一些轴可指定为普通的无处发RAPID运动。

若电机在没有触发的条件停止运动。“触发运动”状态位（第二个电机状态字的第七位返回？命令响应）在移动后设定。若发现触点，移动后该位被清零。

例子 X1000^0
X10^0.01 Y5.43^0.05
A(P1)^(P2) B10^200 C(P3)^0 X10

参考 move-until-tigger (编写运动程序)
torque-limited 触发（设定电机）
RAPID-mode 移动（写出运动程序）

{坐标}{数值}[{坐标}{数值}...]{矢量}{数值}[{矢量}{数值}...]

功能: 圆弧运定义

类型: 运动程序（PROG 及 ROT）

语法: {坐标}{数值}[{坐标}{数值}...]{矢量}{数值}[{矢量}{数值}...]

其中

{坐标}说明了是哪个轴{X,Y,Z,A,B,C,U,V,W}

{数值}是一个常量(没有括弧),或是表达式(在括弧内)代替最后的位置或距离。

[{坐标}{数值}...]说明了多轴的同步运动。

{矢量}是一个参数（I，J，或K）指出元件对于圆弧中心的方向；或是参数R说明矢量的数值。

{数值}说明矢量的值。

注释: [{矢量}{数值}...]类似说明了多轴的同步运动。
给复合的圆运动定义了终点及对于圆心的矢量。在LINER运动模式中，要说明终点，位置（相对于坐标系）或距离（相对于起点）。在圆运动命令中，圆心的位置需要说明。一般是定义圆心的矢量，它可以参考运动的起点（缺省为径向矢量的增量形式，或写成INC(R)），也可以参考运动坐标系(绝对径向矢量形式…语法形式为ABS(R))。另外，命令行中用R{data}来说明圆矢量的值。如果是这种情况PMAC会主动计算圆心位置。若{data}值为正，PMAC计算到目标的劣弧，{data}为负，则计算到目标的优弧。在一行中，不可能用R矢量定义一个整圆。

圆弧运动的平面用NORMAL来说明（缺省为……NORMAL K—1……定义在XY平面上）该命令只能定义XYZ坐标系的平面。即只有X，Y，Z坐标轴可用作圆弧补插。在同一命令行的其它轴要作线性化以同时完成。相对目标点方向用CIRCLE1（），或CIRCLE2（），来区分是顺时针，或逆时针。平面内的顺时针由平面内的NORMAL矢量的方向来确定。如果终点与起点到圆心的距离不同，则在运动过程中平滑过渡。在圆整的定义中补偿是必要的。若起点或终点到圆心的距离为零，程序会出错并停止。

如果从起点到圆心的矢量不在圆弧插补平面中，那么将用其在插补平面上的投影作矢量。若目标终点不在插补平面上，则进行螺旋线运动以到达目标点。

终点（或它这种情况下只需说明矢量在起点所在平面的投影）与起点一致，将形成

圆（用IJK矢量相同）。，因为对于任意坐标轴都不定义终点，终点自动取为起点坐标。

如果在命令行中没定义起点，径向数量，将按直线运动到终点（即便是在圆周运动模式中。

例子 X5000 Y3000 I1000 J1000
 X(P101) Z(102) I(P201) K(P202)
 X10 I5
 X10 Y20 C5 I5 J5
 Y5 Z3 R2
 J10；半径为10的圆

参考 圆周运动（编写运动程序）
 I变量 I13 Ix87,Ix88,Ix89,Ix90
 程序函数NORMAL，ABS，INC CIRCLE1，CIRCLE2，TA，TS，TM，F

A{data}

功能: A-轴移动

类型: 运动程序(PROG ,ROT)

语法: A{数值}

其中
{数值}为一浮点数常量或表达式代表一个位置或距离使用用户坐标系的单位。

注释: 该命令引起坐标轴的移动。(参考{坐标}{数值}定义)

例子 A10
 A(P23)
 A25B10Z35
 A (20*SIN(Q5))

参考 函数{坐标}{数值},B,C,U,V,W,X,Y,Z,CALL,READ

ABS

功能: 绝对移动模式

类型: 运动程序

语法: ABS[({坐标},{坐标}...)]

其中
{坐标}是{X,Y,Z,A,B,C,U,V,W}为坐标轴，或参量R为径向矢量形式。

注释: ABS函数没有参数，坐标轴中点为运动的绝对位置。即为绝对坐标模式。ABS函数使指定坐标轴运动到绝对位置，而其它的轴停留在原位置。若其中一个轴为R，在绝对模式下，用I，J，K来描述圆周运动。（例，是从圆心的矢量，而不是相对起点的矢量）。不带参数的ABS函数不会影响该矢量的表达。缺省的径向矢量为增量式。

如果在程序输给PMAC时没有打开运动缓冲器，它将被当作当前的坐标系原指令执行。

例子	ABS(X,Y) ABS ABS(V) ABS(R)
参考	圆周运动（编写运动程序） 在线命令ABS，INC 程序函数{坐标}{数值},{坐标}{数值}{矢量}{数值},INC

ADDRESS

功能:	电机/坐标系模态寻址
类型:	仅PLC程序1-31
语法:	ADDRESS[#{常数}][&{常数}] ADR[#{常数}][&{常数}]

其中
{常数}是从1至8的整数代表寻址的电机号(#)或坐标系号&。

注释: 执行了该语句，当向电机或坐标轴发出不带地址的指令时，将设定电机或坐标系由定的PLC程序寻址。这种确定了地址的坐标系控制了一套入口的Q 变量，甚至是ATAN2功能，它自动使用Q0。

该命令不影响主机的寻址，其它PLC程序的寻址，或是控制面板输入。这种寻址方式一直起作用直到有其它ADDRESS语句代替它。默认的 POWER-ON/RESET地址为#1及&1。

在运动程序里,对于COMMAND没有固定的寻址模式；每个COMMAND语句在其引用范围内必须包含特定的电机或坐标轴.一个运动程序自动地按执行程序的Q变量运行。

例子	ADDRESS &4 ADR #2 ADDRESS &2#2 ADR#! ; 模态地址电机1 CMD “J+” ; 驱动电机1运动 CMD “#2J+” ; 驱动电机2运动 CMD “J/” ; 关断电机1
----	---

参考 寻址模式(Talking to PMAC)
Q变量(程序计算特征量)
程序指令 COMMAND,Q{常量}={表达式}

ADIS{常量}

功能:	X,Y或Z坐标轴的绝对位移.
类型:	运动程序(PROG 及 ROT)
语法:	ADIS{常量}

其中
{常量}是一个整数代表用作位移矢量的三个连续的Q变量的第一个的编号。

注释: 该函数将目前选择的传递矩阵装载到坐标系中,坐标系由三个Q变量中的特定的一

个指定偏移量.函数将对坐标轴的位置产生影响,从原位置移动到下列变量值处
($X=Q\{\text{数值}\}$, $Y=Q\{\text{数值}\}+1$, $Z=Q\{\text{数值}\}+2$)。

该指令不产生任意轴的运动; 它只对目前位置重新定义。

该指令相当于 $PSET\ X(Q\{\text{数值}\})\ Y(Q\{\text{数值}\}+1)\ Z(Q\{\text{数值}\}+2)$, 和 $PSET$ 一样
 $ADIS$ 在移动中,不产生强制停止。

例子
 $Q20=7.5$
 $Q21=12.5$
 $Q22=25$
 $ADIS\ 20$

参考
轴系传递矩阵(编写运动程序)
在线指令 $DEFINE\ TBUF$
程序指令 $TSEL,AROT,IDIS,IROT,TINIT$

AND({条件})

功能: 条件语句 AND

类型: 仅用于 PLC 程序

语法: $AND\ (\{\text{条件}\})$

其中
 $\{\text{条件}\}$ 是一个简单或复合的条件。

注释: 该语句的的复合条件语句 由 $IF, WHILE$ 的 PLC 程序构成.它需跟着 $IF, WHILE, AND$ 或 OR 语句. AND 是一个逻辑运算符,将该行与程序上一行求与. AND, OR 逻辑运算符在一行的复合条件中区分优先级.(括号中优先执行),但它比 OR 语句 的优先级高。在运动程序中,在一个程序段中可以存在复合条件,但不允许行之间的操作.所以该语句不能用于运动程序中。

例子
 $IF\ (M11=1)$; 该分支将启动一个运动程序
 $AND\ (M12=1)$; $M11, M12$ 都是1,将在循环中
 $AND\ (M21=1)$; $M21$ 仍是0; 注意, $M21$ 立刻被
 $CMD\ "R"$; 置为1以便该指令在下一循环中不再执行
 $M21=1$
 $ENDIF$

参考
写 PLC 程序
条件(程序计算特征量)
程序函数 $IF, WHILE, OR$

AROT{常数}

功能: X, Y 和 Z 轴的绝对旋转或比例缩放

类型: 运动程序($PROG$ 及 ROT)

语法: $AROT\ \{\text{常数}\}$

其中
 $\{\text{常数}\}$ 为正数, 代表九个用于旋转/比例缩放的 Q 变量的第一个的编号。

注释: 该函数将被选择的传递系数矩阵的旋转/缩放系数值加载到坐标系中。其中系数值在指定的 Q 变量开始的九个 Q 变量中.该命令将 X, Y 及 Z 轴用矩阵中 XYZ 矢量重新

定义。

旋转/缩放是相对原来的由坐标轴定义语句定义的XYZ坐标系而言.数学表达为:

$$[Xrot \ Yrot \ Zrot]^T = [Rot \ Matrix][Xbase \ Ybase \ Zbase]^T$$

该指令不产生任意轴的运动; 它只对目前位置重新定义..

注意: 该函数用于坐标轴的缩放时,不能用于半径,圆心描述的圆命令格式。半径不能被缩放.要用I,J和K矢量来描述。

例子: 建立一个3*3的矩阵使XY平面从原位置转过30度。
Q40=COS(30) Q41=SIN(30) Q42=0
Q43=-SIN(30) Q44=COS(30) Q45=0
Q46=0 Q47=0 Q48=1
AROT 50 ; 产生变化

建立一个3*3的矩阵使XYZ比例变成3倍
Q 50=3 Q51=0 Q52=0
Q53=0 Q54=3 Q55=0
Q56=0 Q57=0 Q58=3
AROT50 ; 产生变化

参考 轴的传递矩阵(编写运动程序)
在线命令 **DEFINE TBUF**
程序命令 **TSEL,ADIS,IDIS,IROT,TINT**

B{数值}

功能: B轴的移动

类型: 运动程序(PORG 和 ROT)

语法: B{数值}

其中
{数值}为一浮点型常数或一表达式代表U轴的位置或距离(用户定义单位)

注释: 该函数引起B轴的移动(参考以上的{坐标}{数值}的表达)

参考

BLOCKSTART

功能:

类型: 运动程序(PROG 和 ROT)

语法: BLOCKSTART
BSTART

注释: 该语句允许在一个' stet' 命令行执行一个复杂的移动.程序中' stet' 命令会执行到下一个BLOCKSTOP语句.(在没有BLOCKSTART指令,"Step"指令只执行一个伺服命令).并且如果 I x92=1(禁止运动的复合),所有BLOCKSTART和BLOCKSTOP之间的运动将被复合到一起.如果Ix92=0,这不会影响到从' run' 语句开始的程序的执行。

这种结构对于执行一系列简单的PVT模式的运动是很方便的.因为个别的语句在零

速度时并不停止,这对实现一些简单的操作是非常困难的。

例子

程序段
BLOCKSTART
INC
X10:100
X20:100
X20:100
X10:0
BLOCKSTOP

一个简单的"S"指令将执行上述四段运动。

参考

I变量 Ix92
在线命令 <CONTROL-S>,R,S.
程序命令 BLOCKSTART,BLOCKSTOP

BLOCKSTOP

- 功能:** 标志Stepping Block结束
- 类型:** 运动程序(PROG and ROT)
- 语法:** BLOCKSTOP BSTOP
- 注释:** 该语句标志由BLOCKSTART开始的程序块的结束。该程序块可由"Step"执行,或者如Ix9²=1民间则使其中的多运动叠加在一起。如Ix9²=1它不会影响由RUN开始的程序的执行。
- 例子** 参阅上面BLOCKSTART中的例子
- 参阅** I变量 Ix9²
在线命令 <CONTROL-S>,R,S.
程序命令 BLOCKSART,STOP

C{data}

- 功能:** C-轴运动
- 类型:** 运动程序(PROG and ROT)
- 语法:** C{数据}
- 其中
{数据}为浮点常数或表达式,代表以用户单位为单位的U轴的位置或距离。
- 注释:** 该命令能使C-轴运动(查阅{轴}{数据}定义)

CALL

- 功能:** 用Return跳转入于程序
- 类型:** 运动程序(PROG and ROT)
- 语法:** CALL{数值} [{字母}{数值}...]

其中
第一个{数值}是一个浮点型常数或一个表达式,从1.00000到32767.99999,其中它的

整数部分代表被调用的运动程序段号，小数部分代表被调用程序段的行标号(N或O)，(行标号等于小数部分乘以100,000；每个运动程序在开始部分有一个固定的N0)；

{字母}是除了N和O以外的任一字母，代表一个变量(Q101到Q126 分别对应A到Z)；

下面的{数值}是一个浮点型的常数或表达式代表变量的值。

注释:

该命令使程序执行其子程序并返回到程序的下一行.输入到程序的子程序与程序段是一样的,用PROGn作标号(这样一个程序可以象调用子程序一样调用其它程序).PROG前的值n为CALL后的值:CALL7将执行PROG7并返回。执行一个不存在的子程序会引起程序的错误中断。接在CALL后的值可以包含小数.如果没有小数部分则被调用的程序段从头开始。如果存在小数部分则被调用的程序段把小数部分作为程序段开始的标号(如果该标号不存在则PMAC会产生一个错误并停止执行).PMAC使用五位小数作为行标号；如果位数不足,PMAC将自动补零.例如CALL 35.1被翻译为CALL 35.10000,这将使程序35跳到标号N10000处执行.CALL47.123使程序段47跳到标号N12300处执行。

如果跟在CALL{数值}后的是字母和数值(例X1000)，这些可作为传递给子程序的参数。如果有参数传递，则子程序执行的第一行应该是READ语句。该语句会取出与特定字母关联的值赋予合适的Q变量。例如，对于程序执行的坐标系A后的数值将放在Q101中，接下来B的值放在Q102中；诸如此类Z将被放在Q126中，子程序将使用这些变量。如果子程序带参数的调用另一子程序，使用同样的Q变量。详细内容参考READ指令。

如果在子程序中没用READ语句,或不是在CALL行的所有字母的值都是” read” (如果CALL行中的字母不在READ的字母列表中,READ语句会停止运行),剩余的字母程序行将按它们的普通功能从子程序中返回执行.例如:G01 X10 Y10 等价于CALL 1000.01 X10 Y10。要实现G01的普通功能(线性移动模式),在PROG 1000应该有下列子程序:

N1000 LINEAR RETURN

一旦返回,X10 Y10将根据移动模式强制执行一次移动,该例为线性。

如果指定程序段或行标号不存在，该CALL指令被忽略，程序继续执行,就好象这一指令不存在一样。

例子

CALL500 ; 跳转到PROG500的首位置(N0)
CALL500.1 ; 跳转到PROG500标号为N10000
CALL500.12 ; 跳转到PROG500标号为N12000
CALL500.123 ; 跳转到PROG500标号为N12300
CALL500.1234 ; 跳转到PROG500标号为N12340
CALL500.12345 ; 跳转到PROG500标号为N12345

CALL700 D10 E20 ; 跳转到PROG700经过D和E

参考

在线指令B{常数}

程序指令GOTO,GOSUB,READ,RETURN,G{数值},M{数值},T{数值},D{数值},N{数值},O{数值},PRELUDE

CC0

功能: 关断圆半径补偿

类型: 运动程序(PROG ROT)

语法: CC0

注释: 关断圆半径补偿模式,在下一次的移动中逐渐减小.这等同于数控机床的RS-274语言的G40语句。

例子

CCR0.5	; 1/2单位的圆半径剪切
CC1	; 向左的切割补偿
X10 Y10	; 这次移动中引入补偿
X10 Y20	
X20 Y20	
X20 Y10	
X10 Y10	
CC0	; 关断剪切补偿
X0 Y0	; 移动中删除补偿
OPEN PROG 1000	; G代码子程序
...	
N40000 CC0 RETURN	; 在PMAC中直接完成G40指令

参考 剪切半径补偿
程序指令CC1,CC2,CCR{数值}

CC1

功能: 开剪切半径向左补偿

类型: 运动程序(PROG ROT)

语法: CC1

注释: 这是开剪切半径补偿模式,在下一次移动中逐渐增加补偿。剪切使编程的刀具偏向左,沿着剪切运动方向观察。补偿平面由NORMAL指令决定。这等价于数控机床RS-274语言中的G41语句。

例子

CCR0.25	; 1/4单位的剪切半径补偿
CC1	; 向左的剪切补偿
X10 Y10	; 此次移动中引入补偿
X10 Y20	
X20 Y20	
X20 Y10	
X10 Y10	
CC0	; 关断补偿
X0 Y0	; 移动中删除补偿
OPEN PROG 1000	; G代码子程序
...	
N41000 CC1 RETURN	; 在PMAC中直接完成G41指令

参考: 剪切半径补偿
程序指令CC2,CC0,CCR{数值},NORMAL

CC2

功能: 开向右剪切半径补偿

类型: 运动程序(PROG ROT)

语法: CC2

注释: 这是开剪切半径补偿模式,在下一次移动中逐渐增加补偿。剪切使编程的刀具偏向右,

沿着剪切运动方向观察。补偿平面由**NORMAL**指令决定。这等价于数控机床**RS-274**语言中的**G42**语句。

例子

CR1.5	； 1.5个单位的剪切补偿
CC2	； 向右的剪切补偿
X10 Y10	； 在移动中逐渐引入补偿
X10 Y20	
X20 Y20	
X20 Y10	
X10 Y10	
CC0	； 关断补偿
X0 Y0	； 在移动中删除补偿
OPEN PROG 1000	； G代码子程序
...	
N42000 CC2 RETURN	； 在PMAC中直接完成G42指令

参考

剪切半径补偿
程序指令CC1,CC0,CCR,NORMAL

CCR{数值}

功能: 设定补偿半径

类型: 运动程序(PROG ROT)

语法: CCR{数值}

注释: 该语句用用户单位设定要补偿的半径.该功能是**RS-274**代码中使用的**D**工具数据的一部分。

参考

CIRCLE1

功能 设定复合的顺时针的圆运动.

类型 运动程序(PROG ROT)

语法 CIRCLE1
CIR1

注释

该函数将程序设定为顺时针运动模式。由最近的**NORMAL**语句设定圆运动插补平面,也定义了平面的顺时针还是逆时针。程序用一种圆运动模式代替另一种运动模式: 其余的圆运动模式,LINEAR,PVT, RAPID等等。任一圆运动指令都有关于**R**或**IJK**矢量的描述: 或者在圆运动模式中也可以执行直线运动。

注意

PMAC必须处于分段移动模式(I13>0)以便执行圆插补。如果I13=0(没有分段移动), 移动为线性插补。

例子

LINEAR	； 线性插补模式
X10Y10 F2	； 线性移动
CIRCLE1	； 顺时针插补模式
X20 Y20 I10	； 半径为10的弧
X25 Y15 J-5	； 半径为5的弧
LINEAR	； 返回到线性模式
X25 Y5	； 线性移动

参考: 圆运动(编写运动程序)
I变量I13
程序命令NORMAL,CIRCLE2,LINEAR,RVT,RAPID,SPLINE1,{坐标轴}{数值}{矢量}{数值}

CIRCLE2

功能	设定复合的逆时针圆运动模式
类型	运动程序(PROG ROT)
语法	CIRCLE2 CIR2
注释	<p>CIRCLE2命令使程序按逆时针圆运动模式。由最近的NORMAL指令确定圆插补指令确定它也定义了如何确定平面的顺时针和逆时针运动。</p> <p>程序用一种圆运动模式代替另一种运动模式：其他的圆运动模式，LINEAR，PVT，RAPID等等。任一圆运动指令都有关于R或IJK矢量的描述:或者在圆运动模式中也可以执行直线运动。</p>
注意:	PMAC必须处于分段移动模式(I13>0)以便执行圆插补。如果I13=0(没有分段移动),移动为线性插补。
例子	<pre>LINEAR ; 线性插补模式 X10Y0 F2 ; 线性移动 CIRCLE2 ; 逆时针圆弧插补模式 X20 Y10 J10 ; 半径为10的圆弧 X15 Y15 I-5 ; 半径为5的圆弧 CIRCLE1 ; 顺时针模式 X5 Y25 J10 ; 半径为10的圆弧移动</pre>
参考	圆运动(编写运动程序) I变量I13 程序指令NORMAL,CIRCLE1,LINEAR,PVT,RAPID,SPLINE1,{坐标轴}{数值}{矢量}{数值}

COMMAND “{命令}”

功能	发送程序命令
类型	运动程序(PROG ROT); PLC程序
语法	COMMAND “{命令}” CMD “{命令}”
注释	<p>该函数向PMAC传递一条指令,象从主机发出一样(除寻址方式以外)。如果在引用的字符串中有电机或指定的坐标系，命令会直接传递给电机或坐标轴。如果没有指定的电机或坐标轴则直接传给最先的电机或坐标系。任何一条命令指令内的描述是无模式的；它不影响任何程序的许多地址描述或无模式描述，以及它自己本身。</p> <p>如果I62=0，PMAC在任何一次对命令的数值响应之后自动发出回车符号。如果I62=1，PMAC则不发出回车符号<CR>；在这种情况下，一定要用SEND^M发出回车符号<CR>。</p> <p>对于电机和坐标系，每个PLC程序都有自己的选址模式，相互独立且与主机选址模式无关。它受这PLC程序的选址命令控制。这种形式的选址对从即没电机又没坐标系描述的PLC程序中发出的命令有影响。在启动/重启情况下,所有PLC程序会对电机1和坐</p>

标系1进行选址。

在运动程序中没有模式寻址方式。从没有选址运动程序中发出的任一指定电机或坐标系的命令将各自分别自动地对电机1或坐标系<C.S>1，选址程序发出的命令被放置在命令的队列中，在适当的时候通过PMAC的命令注释解释和执行，可以在后台执行，在两个后台任务之间执行。如果指令来自PLC程序在下一次移动或运动程序中的指令被计算之前不会被翻译。如果指令是PLC，在读入当前PLC程序之前不会翻译。这种延迟使运动看起来不是按顺序进行的。由于指令的等候理论及指令的翻译的优先级比其读入的优先级低，队列将可能溢出。如果对于一个新指令没有位置，程序执行将会暂时中断直到新的指令可以进入队列中。

同样，如果没有主机或主机没有准备好读响应的话，指令向主机产生的响应可能会排满响应队列。在响应队列被清空之前程序可能会暂停执行。在独立的应用中，将I1设为1比较好。禁止连续的信号输入输出，这样响应会在任何时候都传递给串联通道，即使是在没有主机接收的情况。

在PLC程序中，较好的办法是至少设一个状态位，使其置为false来立即产生指令发送的引起的错误。这样可以防止在顺序扫描PLC程序时，同一指令不会被重复地读入。并且避免了指令或响应队列的溢出。在典型的运动程序中，在运动之间的时间间隔避免了这种溢出情况，除非是有许多指令并且需要在很短的时间内进行许多的移动。

PMAC不会向程序中的合法指令发出认收字符<ACK>或<LF>。而在程序发出无效字符时发出字符<BELL>，除非是I6被设置为2。在早期开发时I6最好不要设置为2，当PMAC拒收这样的指令的时候时，你会知道。在实际应用中，将I6设置为2会防止由于响应队列溢出而产生中止，或是防止扰乱正常地主机传输协议。

许多运动程序发出的有效指令都会被拒收。例如，在执行程序时你不可以干涉坐标系的任一电机，因为这些电机都按指定程序运动，即使是当时程序没有向电机发出运动指令。

程序发出的指令，保证在命令行中包含所有的必要的语法(电机和/或坐标系的说明)或使用ADDRESS语句。例如，用CMD “4HN”和CMD “&1A” 语句代替CMD “HM” 及CMD “A”。否则，电机，坐标轴指令会被传递给最近的可寻址的电机及坐标轴，而它们又很可能不是你所要操作的。

例子

```
COMMAND "#1J"
CMD "#4HM"
CMD "&1B5R"

CMD "P1"
47.5
ADDRESS#3
COMMAND "J-"
IF (M40=1 AND M41=1)
CMD "&4R"
M41=0
ENDIF
```

参考:

寻址模式,在线命令(Talking to Pmac)
I变量I1,I3,I6
程序指令ADDRESS,COMMAND^{字母}
编写PLC程序段

COMMAND^{字母}

功能 发送程序控制字指令

类型 运动程序(PROG ROT), PLC 程序

语法	<p>COMMAND^{字母} CMD^{字母}</p> <p>其中{字母}是从A到Z的字符代表相应的控制字。</p>
注释	<p>该语句使运动程序产生一个类似主机发出的控制字符指令,所有的控制字指令都是全局的,所以没有寻址语句。</p>
注:	<p>不要使用引号中带^的字母指令(不使用COMMAND" ^A")否则PMAC试图发出例子中的带两个非控制字符^, A的指令代替控制指令。</p> <p>程序发出的命令被放置在命令的队列中,在适当的时候通过PMAC的命令注释解释和执行,或可以在后台执行,在两个后台任务之间执行。如果指令来自PLC程序在下次移动或运动程序中的指令被计算之前不会被翻译。如果指令是PLC,在读入当前PLC程序之前不会翻译。这种延迟使运动看起来不是按顺序进行的。由于指令的等候理论及指令的翻译的优先级比其读入的优先级低,队列将可能溢出。如果对于一个新指令没有存放空间,程序执行将会暂时中断直到该新的指令可以进入队列。</p> <p>同样,如果没有主机或主机没有准备好读响应的话,指令向主机产生的响应可能会排满响应队列。在响应队列被清空之前程序可能会暂停执行。在独立的应用中,将I1设为1比较好。禁止连续的信号输入输出,这样响应会在任何时候都传递给串联通道,即使是在没有主机接收的情况下。</p> <p>在PLC程序中,较好的办法是至少设一个状态位,使其看为flace来立即产生指令发送。这样可以防止在顺序扫描PLC程序时,同一指令不会被重复地读入.并且避免了指令或响应队列的溢出.在典型的运动程序中,在运动之间的时间间隔避免了这种溢出情况,除非是有许多指令并且在很短的时间内进行许多的移动。</p> <p>PMAC不会向程序中的有效指令发出认收字符<ACK>或<LF>。而是会向程序中的无效字符发出字符<BELL>的,除非是I6被设置为2。在早期开发时I6最好不要设置为2,当PMAC拒收这样的指令的时候时,你会知道。在实际应用中,将I6设置为2会防止由于响应队列溢出而产生中止,或是防止扰乱正常地主机传输协议。</p>
例子	<p>CMD^D禁止PLC程序(相当于主机发出的<CONTROL-D>指令)</p> <p>CMD^K禁止PMAC的电机工作</p> <p>CMD^A中断所以程序并使PMAC继续工作,并使处于开环的循环闭环。</p>
参考	<p>I变量I1, I6</p> <p>在线命令<CONTRAL-A>到<CONTRAL-Z></p> <p>程序命令COMMAND "{命令}"</p>

D{数值}

功能	工具数据(D-代码)
类型	运动程序
语法	<p>D{数值}</p> <p>其中</p> <p>{数值}是一个浮点型常数或一个表达式,范围从0.000到999.999,指定将跳向的程序数和行标号。</p>
注释	<p>PMAC把这指令解释为CALL 10n3。 ({数值}*1000)命令,其中,n是{数值}的百位,{数值}是{数值}没有百位的值(数学术语为模100).也就是说该语句产生一个跳转到运动程序10n3处(没有返回),和特定的行号。(程</p>

序10n3常常用作完成工具数据的准备工作,象设计者所想的一样。){数值}的值是从0.0到99.999,对应行标号N0到N99999。

这种结构允许在运动程序10n3中用D代码写出子程序。常数按D代码格式一组或多组{字母}{数值},传递给子程序,象CALL及READ语句一样。

大多数用户程序在0—99行有D代码,仅允许使用PROG1003,并允{数值'}等价于{数值}来描述行标号。

例子	D01 jumps to N1000 of PROG 1003 D12 jumps to N1200 of PROG 1003 D115 jumps to N1500 of PROG 1013
参考	程序指令CALL{数值},G{数值},M{数值},T{数值},RETURN

DELAY{数值}

功能 延迟指定的时间

类型 运动程序

语法 DELAY{数值}
DLY{数值}

其中
{数值}是一个浮点型常数或表达式,以毫秒为单位指定延迟时间。

注释 该指令让PMAC保持坐标系内所有轴的指令位置在指定的时间内静止。DELAY和DWELL之间存在不同。第一点,如果DELAY指令在一个复合运动之后,在DELAY时间之内产生TA减速时间而不是之前产生。第二,实际的延迟时间不同于时间基准改变(当前的%值,几乎所有情况),DWELL使用同样的时间基准(%100)。第三,PMAC在DELAY时间中预先计算下一次的运动,但在DWELL时间内并不做。

DELAY指令相当于在指定的时间内运动零位移。至于移动,如果指定的DELAY时间比加速时间短(TA或 $2 \times TS$),延迟将用加速时间,而非指定的DELAY时间。

例子 DELAY750

DELAY(Q1+100)

参考: 时间基准控制(使PMAC与外部事件同步)
I变量I10,Ix87,Ix88
在线命令%{常数}
程序命令DWELL,TA,TS

DISABLE PLC{常数}[,{常数}...]

功能 禁止PLC程序

类型 运动程序(PROG ROT),PLC程序

语法 DISABLE PLC {常数}[,{常数}...]
DISABLE PLC {常数}[...{常数}]
DIS PLC {常数}[,{常数}...]
DIS PLC {常数}[..{常数}]

注释 该指令禁止PLC程序执行。程序段由数值指定,可个别使用,在表中由逗号隔开,或是一个连续的范围。禁止PLC,不能在PLC程序的扫描中中断PLC; 这样可防止第二次

扫描PLC程序。

例子
DISABLE PLC 1
DISABLE PLC 4,5
DISABLE PLC 7..20
DIS PLC 3,8,11
DIS PLD 0..31

参考: I变量I5

在线命令ENABLE PLC, DISABLE PLC, ENABLE PLCC, DISABLEPLCC,
<CONTROL-D>

DISABLE PLCC{常数}[,{常数}...]

功能 禁止编译PLC程序

类型 运动程序(PROG ROT),PLC程序(不编译或编译)除PLC0及PLCC0

语法
DISABLE PLCC {常数}[,{常数}...]
DISABLE PLCC {常数}[..{常数}]
DIS PLCC {常数}[,{常数}...]
DIS PLCC {常数}[..{常数}]

其中
{常数}是从0到31的整数编译的PLC程序号。

注释 该程序禁止指定的编译后的PLC程序段工作。这些程序段由数值指定，可一行一行的使用。在表中由逗号分开，或是用一个连续的范围。

I变量I5是一个独立的PLC程序工作的控制器。I5的两位可看作门的开关，PLC及PLCC的使能/禁止位，象屋内的灯的继电器。PLC工作是开关和继电器都要处于开的状态，开关和继电器可独立工作而不影响其它设备。

例子
DISABLE PLCC 1
DISABLE PLCC 4,5
DISABLE PLCC 7..20
DIS PLCC 3,8,11
DIS PLC 0..31

参考 I变量I5

在线命令ENABLE PLC,DISABLE PLC,ENABLE PLCC,DISABLE
PLCC,<CONTROL-D>
程序命令ENABLE PLC, DISABLE PLC, ENABLE PLCC

DISPLAY[{常数}] “{信息}”

功能 在显示端口显示字符

类型 运动程序，PLC程序

语法
DISPLAY[{常数}]” {信息}”
DISP [{常数}]” {信息}”

其中
第一个{常数}是从0到79的整数代表要显示的位置；

第二个{常数}是从2到16的整数代表要显示的字符的个数(正数,小数,分数)

第三个{常数}是从0到9的正数且要比第二个数小2代表要显示的分数的数。

{变量}是要显示的变量的名字。

注释 该指令使PMAC产生一个特定的值,包含指定的输出到显示端口的变量值。I, P, Q或M中的任意一个的值会和该指令一起显示。第一个常数值指定了在显示屏上的字符串的起始点。值从0到79,其中0是左上方,39是右上方,40是左下方,79是右下方。第二个常数为显示的字符的个数;值从2到16。第三个常数为到右面小数点位置的个数。值从0到9,并且至少字符数少2。在语句中指定最后的变量名I,P,Q及M。

DWELL

功能 延迟指定的时间

类型 运动程序(PROG ROT)

语法 DWELL{数值}
DWE{数值}

其中
{数值}是一个非负数的浮点型常数或表达式用毫秒代替延迟的时间。

注释 该指令使坐标系中的所有的轴的位置保持{数值}指定的时间不变。

DWELL和DELAY之间有三点不同。第一,如果前面伺服系统命令为复合移动,停止前要有TA的时间减速,之后开始延迟DWELL时间。第二,DWELL是随时间基准变化,它总是按“real time”工作(在I10中定义)。

第三,在DWELL中不对之后的移动做预计算;PMAC会在完成之后做进一步的运算,执行的时间由I11及I12指定。

使用DWELL指令,甚至是DWELL0指令,外部时间基会与主信号不同步。

例子 DWELL250
DWELL(P1+P2)
DWE0

参考 Dwell 和 Delay(写出运动程序)
I变量I10, I11, I12
程序指令DELAY

ELSE

功能 开始错误条件分支

类型 运动程序(PROG),PLC程序

语法 ELSE (运动程序或PLC程序)
ELSE(条件) (运动程序)

注释 该语句与IF语句配对使用(ELSE语句前边需要有一个IF语句,但IF语句后可没有ELSE语句)。它接在IF条件为真的语句之后。其后接IF语句为假时的执行语句。ELSE可以采用两种模型(在PLC中只可以用第一种);如果在该行没有语句,所有之后到ENDIF的语句都会在IF条件为假时执行。

ELSE

```

    {语句}
    [{语句}...]
ENDIF

```

接在该行的语句，在IF条件为假只执行第一句，该情况下不需使用ENDIF语句。

```
ELSE{语句} [{语句}...]
```

例子 第一个例子有多个是和非的分支。在运动程序及PLC程序中都可以使用。

```

IF (M11=0)
P1=17
P2=13
ELSE
P1=13
P2=17
ENDIF

```

第二个例子有多行是分支，只有一行非的语句。该结构只用于运动程序中。

```

IF (M11=0)
X(P1)
DWEELL 1000
ELSE DWEELL 500

```

以下例子有一单行是分支，并有多行非分支。该结构仅用于运动程序中。

```

IF(P1!<5)X10
ELSE X-10

```

参考 程序命令IF，ENDIF

ENABLE PLC

功能 PLC缓冲区使能

类别 运动程序(PROG),PLC程序

语法 ENABLE PLC {常数}[,{常数}...]
 ENABLE PLC {常数}[..{常数}...]
 ENA PLC {常数}[,{常数}...]
 ENA PLC {常数}[..{常数}...]

注释 I5设置正确的前提下，该命令允许指定PLC缓冲区操作。

例子 ENABLE PLC 0
 ENABLE PLC 1,2,5
 ENABLE PLC 1..16
 ENA PLC 7

参考 I变量I5
 在线命令ENABLE PLC,DISABLE PLC,<CONTROL-D>
 程序命令DISABLE PLC

ENABLE PLCC

功能 已编译PLC程序使能

类别 运动程序（PROG和ROT），PLC程序（未编译和已编译）

语法 ENABLE PLCC {常数}[,{常数}...]

ENABLE PLCC {常数}[.{常数}...]
ENA PLCC {常数}[,{常数}...]
ENA PLCC {常数}[..{常数}...]
以上{常数}为一0到31的整数，代表已编译PLC号。

注释 I5设置正确的前提下，该命令允许指定PLC(PLCC)缓冲区操作。程序是由编号指定的，编号(程序)可以单独使用，也可在一行中有逗号隔开，或在一个连范围内。I变量I5为一独立的PLC程序操作主控单元。假设I5中的两位数为一厂房的主电路断路器，那么PLC和PLCC的使能/禁止位就可以看作厂房内独立的电灯的开关。断路器和开关可以独立的工作而不影响其它位的设置。

例子 ENABLE PLCC 0
 ENABLE PLCC 1,2,5
 ENABLE PLCC 1..16
 ENA PLCC 7

参考 I变量I5
 在线命令ENABLE PLC,DISABLE PLC,ENABLE PLCC,DISABLE PLCC
 ,<CONTROL-D>
 程序命令DISABLE PLC,DISABLE PLC,DISABLE PLCC

ENDIF

功能 条件块结束标志

类别 运动程序（仅PROG），PLC程序

语法 ENDIF
 ENDI

注释 该语句标志由IF起始的条件块的结束。在没有“非”分支的前提下，它可以结束IF后面的“是”分支，或可以结束ELSE后面的“非”分支。在嵌套条件的情况下，ENDIF和IF或ELSE的正确配合很重要。在PLC程序中，每一个IF或IF/ELSE对必须对应一个ENDIF，因此ENDIF总是与距它最近的IF相配合。在运动程序中，IF或ELSE语句只作用于同一行时，不需要ENDIF，此时ENDIF就和前一个IF语句相配合。

例子 IF (P1>0)
 X1000
 ENDIF
 IF (P5=7)
 X1000
 ELSE
 X2000
 ENDIF

参考 逻辑结构(编写运动程序)
 条件语句(编写PLC程序)
 程序指令IF，ELSE

ENDWHILE

功能 表示条件循环的结束

类型 运动程序(PROG)，PLC程序

语法 ENDWHILE
 ENDW

注释 该语句表示一个由WHILE语句引导的条件循环语句的结束。WHILE语句可以是嵌

套的，所以ENDWHILE语句是和最近的WHILE语句配合，而不是和前边的WHILE语句。在运动程序中由WHILE引导的单行运动，不需与之配合的ENDWHILE。执行一个运动程序时，当遇到ENDWHILE语句时，PLC程序的扫描就结束了(通讯或其它的PLC程序)。该PLC从WHILL语句开始下一次扫描。在执行运动程序中，在寻找移动指令时，如果程序产生两个向后的跳转，PMAC会使运动尽可能复合起来。它会继续执行其它指令，重新执行后面读到

的运动程序。可以引起向后跳转的两条指令是：ENDWHILE及GOTO。(RETURN未计算在内)

相关的结果是在执行移动指令中PMAC遇到两个ENDWHILE指令将不产生运动合成。

例子 WHILE (Q10<10)
 Q10=Q10+1
 ENDWHILE

参考 程序指令WHLIE，ENDIF

F{数值}

功能 设定进给速度

类型 运动程序(PROG ROT)


语法 F{数值}
 其中
 {数值}是一个正浮点型常数或表达式代表用户定义的时间坐标和长度单位下的速度矢量。

注释 该语句设定了下一次LINER，CIRCLE模式的复合运动的指令速度。在其它类型的

运动将被忽略(SPLINE，PVT及RAPID)。它代替了前边的TM及F语句的值，并由后面的TM及F语句所代替。

F指令指定的速度单位是每个时间单位下(由“Feed Time Unit”坐标系的I变量：Ix90)的位置值(由轴定义语句定义)。

这里指定速度是所有进给轴中的矢量速度。及，运动时间按进给轴的矢量距离(各独立轴的平方的二次方根)被此处定义的进给速度的值除。最小的运动进给

速度要使移动时间小于毫秒，最大的进给速度要使移动时间大于1毫秒。在同一行没有进给轴移动指令的移动指令，会采用在同一时间内完成的必要速度。

如果一指定进给速度的移动的矢量距离的计算移动的时间比加速时间(TA,2*TS)小，移动会采用全加速来代替，轴的移动比F指令指定的速度低。用FRZX指令指定进给轴。如果没有使用FRAX指令，默认的进给轴是X，Y和Z轴。在圆弧插补中涉及到的轴会自动成为进给轴，而不管上一条FRAX指令指定了哪一轴。在多轴的系统中，在解析几何中移动的进给速度明细表是很有用的。在平面或空间这些移动的任一个都会给出一定量的速度，而不管运动的方向。

例子 F100
 F31.25
 F(Q10)
 F(SIN(P8*P9))

参考 I变量Ix87,Ix88,Ix90
 在线命令#{常量}→{轴定义},FRAX
 程序指令FRAX，LINEAR，CIRCLE，TM。TA，TS

FRAX

功能 指定进给轴

类型 运动程序(PROG,ROT)

语法 FRAX[({坐标轴},{坐标轴}...)]

其中

{坐标轴}是(X,Y,Z,A,B,X,U,V,W)中任一指定进给速度计算中的进给轴。

注释 该指令在预先指定进给的运动中指定用于矢量进给运算的坐标轴。PMAC按矢量距离(各轴距离的平方和的平方根)除以各轴的进给速度来计算时间。同一行没有指定进给速度的轴的移动在同一时间完成,并选择合适的移动速度。

矢量进给速度只在笛卡坐标系中有明显的几何意义。正因此导致了不管方向的恒定的速度,但在非笛卡坐标系或轴以上的坐标系中这是可能的。

不带常数的FRAX指令使坐标系中所有的轴在随后的移动指令中成为进给轴。带常数的FRAX指令在随后的移动指令中使特定的轴成为进给轴,而其它轴为非进给轴。

如果没有打开运动程序缓冲器,当该指令传递到PMAC时,它将作为一个在线的坐标系。

例子 以毫米标尺的三轴笛卡坐标系:

```
FRAX(X,Y)
INC
X30 Y40 Z10 F100
```

矢量距离 $\text{SORT}(30^2+40^2)=50\text{mm}$.移动速度为 100mm/sec ,移动时间为 0.5sec .X轴速度 $30/0.5=60\text{mm/sec}$; Y轴速度为 $40/0.5=80\text{mm/sec}$; z轴速度为

```
10/0.5=20mm/sec
Z20
```

矢量距离为 $\text{SORT}(0^2+0^2)=0\text{mm}$.移动时间(非复合)为 0.0sec ,所以Z轴速度受加速度常数。

```
FRAX(X,Y,Z)
INC
X-30 Y-40 Z120 F65
```

矢量距离为 $\text{SORT}(-30^2+-40+120^2)$.移动时间为 $130/65=2.0\text{sec}$.X轴速度为 $30/2.0=15\text{mm/sec}$; Y轴为 $40/2.0=20\text{mm/sec}$; Z轴速度为 $120/2.0=60\text{mm/sec}$.

参考 I变量lx87,lx88,lx89,lx90
在线指令FRAX
程序指令F,LINEAR,CIRCLE,{坐标轴}{数值}.

G{数值}

功能 准备代码(G代码)

类型 运动程序

语法 G{数值}

其中

{数值}是一个浮点型常数或一表达式,其范围在0.000到999.999,指定要跳转到的程序段号及行号。

注释 PMAC将该指令解释为CALL10n0.({data'}*1000)指令, n是{data}的百位, {data'}是{data}去掉百位后的值(即数学上的模100)。那么, 该语句使运动程序跳转(及返回)到10n0, 及指定行。(程序10n0用于实现满足程序设计者的要求的预备代码)。对应于行号N0到N9999, {data}的值从0.0到99.999。

该结构允许通过编写运动程序10n0的子程序, 实现用于机床工具应用的G代码用户化。与CALL和READ语句一样, 参数可由其后的含一组或多组{letter}{data}传递给这些子程序。

多数用户只使用0-99范围内的G代码, 只允许使用PROG 1000, 允许{data'}等于{data}来直接指定行号。

例子 G01 跳转到PROG 1000的N1000
G12 跳转到PROG 1000 的N12000
G115 跳转到PROG 1010的N15000

参考 程序命令CALL {data},D{data},M{data},T{data},RETURN

GOSUB

功能 带返回语句的无条件跳转

类别 运动程序(仅PROG)

语法 GOSUB(data)

其中

{data}为一代表跳转到的行号的常数或表达式;

{letter}(可选)为除N或O外的任一字母。

注释 该命令使运动程序跳转到同一运动程序段中{data}指定的行号(N或O), 当程序遇到GOSUB后的RETURN命令后马上返回。如果{data}是一个常数, 到子程序的路径在程序运行之前已经连接完成, 所以跳转非常快。如果{data}是一个变量表达式, 在运行时必须求出其值和相应的行号, 然后再去查找。该查找一直向下直到程序结尾, 然后继续(如果需要的话)从下一程序的头部开始。变化的GOSUB命令允许与许多高级语言中的CASE语句相同的结构。如果指定的行号未找到, GOSUB命令可以忽略, 并且就象没有执行该指令一样, 继续执行程序。CALL与此相同, 只是它可以跳转到另一个运动程序。

例子 GOSUB300跳转到程序段N300, 且在return语句处返回。
GOSUB8743跳转到程序段N8743, 且在return语句处返回。
GOSUB(P17)跳转到程序段的行标号,

GOTO

功能 不带返回语句的条件跳转

类型 运动程序(PROG)

语法 GOTO{数值}

其中

{数值}是一个正数或表达式, 值从0到99.999。

注释 该指令使程序跳到{数值}指定的行处继续执行, 并不返回。如果{数值}是一个常数,

会在程序运行之前编译好，所以跳转是很快的。如果{数值}是一个变量表达式，只能在执行中计算，然后在寻找指定的行。从开始一直向下搜索的程序的最后，然后继续(如果必要)在从头开始向下查找。一个带变量的GOTO指令等价于高级语言中的CASE语句。(看以下例子)如果没有找到指定的行标号，程序会停止运行，坐标系的运行时间错误位被置位。

例子 GOTO750
 GOTO35000
 GOTO1
 GOTO(50+P1)
 N51 P10=50*SIN(P11)
 GOTO60
 N52 P10=50*COS(P11)
 GOTO60
 N53 P10=50*TAN(P11)
 N60 X(P10)

参考 写出运动程序;
 程序指令CALL, GOSUB, N, O。

HOME

功能 程序复位

类型 运动程序

语法 HOME{常数}[, [常数]...]
 HOME{常数}..{常数}[, {常数}..{常数}...]
 HM{常数}[, [常数]...]
 HM{常数}..{常数}[, {常数}..{常数}...]

其中

{常数}是从1到8的正数，代表电机号。

注释 该指令使指定电机完成复位循环。注意电机必须由数字直接指定，而不是与坐标系名与之匹配。你必须指定那一电机要复位。所有在HOME语句中指定的电机都将处于复位状态。同时，如果你希望电机顺序复位，在串行的指令中指定(例如 HOME1 HOME2)，可以在同一行。

复位移动开始之前，所有的移动都要停止。在所有指定的电机都复位之前，所有其它的程序段都停止执行。复位的方向，速度，加速度都由电机I变量指定。如果指定的电机并不在当前执行程序的坐标系中,该指令或其一部分将被忽略，但并不产生错误。

复位的速度由Ix23决定。如果Ix23=0指定轴的复位指令将被忽略。

不象在线复位指令，复位电机的号由HOME后的数字指定，而不是之前。而且一个在线复位指令仅仅启动一个复位查询--并不给出任何复位结束的自动识别，在程序段中连续执行。一个PLC程序仅仅发送一个在线复位指令。

例子 HOME1 ; 这是运动程序指令
 HM1,2,3
 HOME1..2,5..7
 HM1..8
 #1HOME ; 这是在线指令
 #1HM,#2HM,#3HM

参考: 复位寻零移动(基本电机移动)

HOMEZ

功能 编程零移动复位

类型 运动程序

语法 HOMEZ{常数}[, {常数}...]
HOMEZ{常数}..{常数}[, {常数}..{常数}...]
HMZ{常数}[, {常数}...]
HMZ{常数}..{常数}[, {常数}..{常数}...]

其中
{常数}是从1到8的正数代表电机号。

注释 该指令使电机产生一个伪复位循环指令。在执行中，目前的电机位置被设为电机的零位置。在执行HOMEZ指令中有错误或在坐标轴定义有偏差，指令之后的位置等于随后错误及轴定义偏差的负值。注意电机必须由数字直接指定，而不能用电机名来匹配。必须指定那些电机要复位。所有在HOMEZ指令中指定的电机都将同时复位。

不象在线复位指令，复位电机的号由HOME后的数字指定，而不是之前。

例子 HOMEZ1 ; 这是运动程序指令
HMZ1,2,3
HOMEZ1..3,5..7
HMZ1..8
#1HOMEZ
#!HMZ,#2HMZ,#3HMZ

参考: 复位寻零运动(基本电机移动)
在线电机指令HOME,HOMEZ
程序指令HOME

I{数值}

功能 I变量指定循环运动或NORMAL矢量

类型 运动程序(PROG ROT)

语法 I{数值}

其中
{数值}是一个浮点型常数或一表达式代表用用户坐标系单位I的值为单位。

注释 在循环运动中，指定了平行于X轴的矢量的分量。起始点或是开始运动的点或(用于增量坐标模式)是XYZ零点(用于绝对坐标模式)。在NORMAL指令中，这指定圆插补指令的平面中的矢量的分量及平行于X轴的刀具半径补偿。

例子 X10 Y20 I5 J5
X(2*P1) I(P1)
I33.333 指定一个整周期循环圆心为从起点到终点X方向33.333处。
NORMAL I—1 指定矢量平面为YZ平面

参考: 圆弧插补，刀具半径补偿(写出运动程序)

在线指令 I{常数}
程序指令
{坐标轴}{数值}{矢量}{数值},ABS,INC,NORMAL,J,K,I{常数}={表达式}

I{常数}={表达式}

功能 设定I变量的值

类型 运动程序(PROG ROT)，PLC程序

语法 I{常数}={表达式}

其中
{常数}是从0到1023的值代表I变量的号。
{表达式}为指定的I变量的值。

注释 该指令设定指定的I变量的值为右边的等式的值。在运动程序正式执行时赋值就已经完成了(因为在程序前需要计算)。如果你想在程序下一次运动前给I变量赋值，你要将M变量赋值给I变量的寄存器，使用同步的M变量赋值语句(M{常数}=={表达式})。

例子 I130=30000
I902=1
I131=p131+1000

参考: PMAC如何执行运动程序(编写运动程序)
在线命令 I{常数}={表达式}
程序指令 M{常数}={表达式}
P{常数}={表达式}
Q{常数}={表达式}
M{常数}={表达式}

IDIS{常数}

功能 X, Y及Z轴的增量位移

类型 运动程序(PROG ROT)

语法 IDIS{常数}

其中
{常数}为一个正数，代表增量矢量中使用的三个连续的Q变量中的第一个。

注释 该指令加到目前选择(用TSEL)的坐标系的传递矩阵的偏移值及包含在指定开始的三个Q变量。这用变量($X_{new}=X_{old}+Q\{常数\}$, $Y_{new}=Y_{old}+Q\{常数\}+1$)的值来改变目前X,Y,Z的位置(从上一轮的指令移动)。该指令不产生任何轴的移动；它仅仅改变目前位置的名字。这条指令与PSET指令相似，而且IDIS也是增量式的在移动中并不强制停止，象PSET指令一样。

例子 X0 Y0 Z0
Q20=7.5
Q21=12.5
Q22=20
IDIS 20 ; 它使当前位置为X7.5 Y12.5 Z20
IDIS 20 ; 它使当前位置为X15 Y25 Z40

参考: 坐标轴传递矩阵(编写出运动程序)
在线命令 DEFINE TBUF
程序命令 TSEL,ADIS,AROT,IROT,TINIT

IF({条件})

功能 条件分支

类型 运动及PLC程序

语法 IF({条件}) (仅在固定的运动程序反PL程序中有效)
IF({条件}) {动作}[{动作}...] 仅用于循环或固定的运动程序
其中
{条件}包含一个或多个{表达式}
{比较器}{表达式}, 用逻辑运算操作AND或OR连接。
{动作}是程序指令

注释 该指令允许程序中的条件分支。该行运动语句或这行以下的语句, 在条件为真是执行这些语句; 在条件为假时, 执行ELSE语句之后的指令。如果下一行不是由ELSE指令开始, 在这一行之后需要ENDIF。当IF语句的同一行有ELSE语句, 该ELSE语句自动与该IF语句匹配, 而不是IF语句嵌套中的前述的IF语句。

如果条件为真, 在该行后没有语句, PMAC会执行到下一个ENDIF前的所有语句或是ELSE语句(这种语法在运动及PLC程序中是非法的)。如果条件为假则跳到ENDIF后是ELSE语句并继续执行。

在循环程序中, 允许使用只有IF而没有ENDIF及ELSE的语句。在PLC程序中, 复合条件语句可由AND或OR连接多个程序行。在固定的程序及PLC程序中IF条件及WHILE循环嵌套是有限制的。在循环语句中不允许嵌套。

例子 IF (P1>10) M1=1

IF (M11=0 AND M12!=0) M2=1 M3=1

IF (M1=0) P1=P1-1
ELIF (M11=0)
P1=1000*SIN(P5)
X(P1)
ENDIF
IF (P1<0 OR P2!<0)
AND (P50=1)
X(P1)
DWEELL 1000
ELSE
X(P1*2)
DWEELL 2000
ENDIF

参考: 条件语句(程序计算特征)
程序命令ELSE, ENDIF, WHILE, AND, OR

INC

功能 增量移动模式

类型 运动程序

语法 INC[({坐标轴})[({坐标轴})...]]

注释 不带常数的INC指令产生随后的指令位置, 该位置为被看作从上一轮的指令点开始的增量为位置, 适合于当前运动程序中的坐标系下的所有轴。这就是所谓增量模式, 与绝对模式相对。带常数的INC语句使指定轴处于增量模式, 其它的不变。

如果R被指定为一个轴，圆移动的I，J和K及半径矢量描述将被指定为增量模式。如果没有打开运动程序缓冲器，当指令传递到PMAC时，它将按在线坐标系指令执行。

例子 INC(A,B,C)
 INC
 INC(U)
 INC(R)

参考: 圆弧移动(编写运动程序)
 在线指令ABS，INC
 程序指令{坐标轴}{数值},{坐标轴}{数值}{矢量}{数值},ABS.

IROT{常数}

功能 X,Y和Z轴的增量旋转/X,Y,Z轴的比例系数

类型 运动程序(PROG ROT)

语法 IROT{常数}

其中
{常数}为一正数代表旋转矩阵中的九个Q变量的第一个。

注释 该指令使目前选择的坐标系的传递矩阵和包含在指定起始位的九个Q变量中的变换矩阵相乘。这将影响到目前X，Y和Z轴的指令位置，与旋转/比例矩阵相乘，改变了旋转的角度及比例系数。

相对最近一次的XYZ坐标系的旋转与缩放，由最近的AROT或IROT指令定义。

数学执行上为：

[新的旋转矩阵]=[原旋转矩阵][增量旋转矩阵]

$[Xrot \ Yrot \ Zrot]^T = [新旋转矩阵][Xbase \ Ybase \ Zbase]^T$

该指令不会引起轴的运动；它只是给当前位置重命名。

例子 建一个在XY平面内绕原点转30度的3×3的矩阵
 Q40=COS(30) Q41=SIN(30) Q42=0
 Q43=-SIN(30) Q44=COS(30) Q45=0
 Q46=0 Q47=0 Q48=1
 IROT 40 :完成变换，从目前位置旋转30度
 IROT 40 :再旋转30度

 建一个在XYX空间内放大3倍的3×3矩阵
 Q50=3 Q51=0 Q52=0
 Q53=0 Q54=3 Q55=0
 Q56=0 Q57=0 Q58=3
 IROT 50 :完成放大3倍
 IROT 50 :继续放大3倍

参考: 系数矩阵变换(编写运动程序)
 在线指令DEFINE TBUF
 程序指令TSEL，ADIS，AROT，TINIT

J{数值}

功能 描述圆运动J矢量

类型	运动程序(PROG ROT)
语法	J{数值}
	其中 {数值}是一浮点型常数，或一表达式代表用户定义的坐标系矢量的J分量。
注释	在圆弧运动中，用以描述圆弧的矢量平行于Y轴的分量。矢量的起始点为移动的起始点(INC(R)模式——默认形式)或是XYZ坐标轴的原点(ABS(R)模式)。在NORMAL指令中，这指定了平面内圆插补及刀具补偿的平面矢量的平行于Y轴的分量。
例子	X10 Y20 I5 J5 Y(2*P1) J(P1) J33.333 指定一个圆心为33.333，起点终点都在Y轴的整圆。 NORMAL J-1 指定ZX平面的矢量。
参考	圆弧插补，刀具半径补偿(编写运动程序)运动程序指令{坐标}{数值}{矢量}{数值}，ABS，INC，NORMAL，I，K。

K{数值}

功能	为圆运动指定K矢量
类型	运动程序(PROG ROT)
语法	K{数值}
	其中 {数值}是一个浮点型常数或一表达式，代表用户坐标系下的矢量K分量的值。
注释	在圆弧运动中，它指定了对于圆弧中心的矢量的平行于Z轴的分量的值。矢量的起点是运动的起点(在INC(R)模式中——缺省形式)或是XYZ坐标原点(对于ABS(R)模式)。在NORMAL指令中，它指定了圆插补及刀具半径补偿的矢量平行于Y轴的分量。
例子	X10 Z20 I5 K5 Z(2*P1) K(P1) K33.333设定一个半径为33.333整圆，起点终点都在Z轴正向 NORMAL K-1 设定XY平面的矢量
参考	圆弧插补，刀具半径补偿(写出运动程序)运动程序指令{坐标}{数值}{矢量}{数值}，ABS，INC，NORMAL，I，J。

LINEAR

功能	复合线性插补运动模式
类型	运动程序(PROG ROT)
语法	LINEAR LIN
注释	LINEAR使程序执行复合线性运动模式(是上电/复位的默认工作模式)。以后程序将按照这种运动模式执行。在每个坐标轴运动的速度由最近的进给(F)和运动时间(TM)指令决定。LINEAR指令是程序脱离其它运动模式(CIRCLE，PVT，SPLINE)，任何其它类型的运动模式不能脱离LINEAR模式执行。
例子	LINEAR ABS

```

CIRCLE1 X10 Y20 I5
  LINEAR X10 Y0
  OPEN PROG 1000 CLEAR
  N1000 LINEAR RETURN

```

参考 线性复合运动(编写运动程序)
 I变量 Ix87,Ix88,Ix89,Ix90;
 程序指令CIRCLE, PVT, RAPID, SPLINE, TA, TS, TM, F, {坐标}{数值}

M{constant}={expression}

功能 设置M变量值

类型 运动程序（PROG和ROT）

语法 M{constant}={expression}

{constant}是从0到1023的整型变量，表示M-变量的号码。

{expression}是个数学表达式，用以给M-变量赋值。

备注 本命令将等号右侧表达式的值赋给M-变量

在运动程序中，任务是逐行顺序完成的，将任何运动程序的最终执行顺序排序是不必要的。如果它是在运动的中间，那么赋值将会在程序的一两步前执行完毕。（原因是需要在程序中计算须提前）

M{constant}=={expression}

功能 同步M变量值分配

类型 运动程序

语法 M{constant}=={expression}

{constant}是从0到1023的整型变量，表示M-变量的号码。

{expression}是个数学表达式，用以给M-变量赋值。

备注 此命令将允许M-变量在下一次运动或停止的开始同步更新。这在将M-变量用于输出时非常有用，因此在运动的开始或结束时输出会同步改变。非同步的计算（用“=”）将在前一次运动中提前执行。

此命令不能用于任何手轮复用形式的M-变量（TWB, TWD, TWR, TWS）。

对于这种形式，右侧的表达式是非同步计算的，但是其结果直到下一个运动的开始才赋值给M-变量。

请记住如果在下一次运动前又使用了M-变量，则你将不会得到此命令赋给M-变量的值。

例子 X10
 M1==1 ; 在下一个运动开始将输出置1
 X20
 M60==P1+P2

参考 PMAC如何执行运动程序，同步变量赋值（编写运动程序）
程序命令I{constant}=, P{constant}=, Q{constant}=, M{constant}=.

M{constant}&={expression}

功能 M-变量“与”赋值

类型 运动程序（PROG和ROT）

语法 M{constant}&={expression}

constant}是从0到1023的整型变量，表示M-变量的号码。

{expression}是将与M-变量做“与运算”的数学表达。

备注 此命令与M{constant}=M{constant}&{expression}等价，除了M-变量的位与运算和赋值都只在运动命令的开始和结束时才执行。数学表达式在程序行解释时已被作为异步表达式计算出来。

此命令不能用于任何手轮复用形式的M-变量（TWB，TWD，TWR，TWS），或任何双字形式的变量（L，D，F）。

请记住如果在下一次运动前又使用了M-变量，则你将不会得到此命令赋给M-变量的值。

例子 M20&=\$FE ; M20变量底字节赋值

M346&=2 ; 除第一位其他均清零

参考 PMAC如何执行运动程序，同步变量赋值（写运动程）
程序命令M{constant}=, M{constant}==, M{constant}|=, M{constant}^=。

M{constant}|={expression}

功能 M-变量“或”赋值

类型 运动程序（PROG和ROT）

语法 M{constant}1={expression} M{constant}1={expression}

{constant}是从0到1023的整型变量，表示M-变量的号码。

{expression}是将与M-变量做“或运算”的数学表达。

备注 此命令与M{constant}=M{constant}|{expression}等价，除了M-变量的位或运算和赋值都只在运动命令的开始和结束时才执行。数学表达式在程序行解释时已被作为异步表达式计算出来。

此命令不能用于任何手轮复用形式的M-变量（TWB，TWD，TWR，TWS），或任何双字形式的变量（L，D，F）。

请记住如果在下一次运动前又使用了M-变量，则你将不会得到此命令赋给M-变量的值。

例子 M20|=\$01 ; M20低位置1, 其它位不变
 M875|=\$FF00 ; 设置高位, 低位不变

参考 PMAC如何执行运动程序, 同步变量赋值 (写运动程序)
 程序命令M{constant}=, M{constant}==, M{constant}&=, M{constant}^=。

M{constant}^={expression}

功能 M-变量 “异或” 赋值

类型 运动程序 (PROG和ROT)

语法 M{constant}^={expression}
 {constant}是从0到1023的整型变量, 表示M-变量的号码。
 {expression}是将与M-变量做 “异或运算” 的数学表达。

备注 此命令与M{constant}=M{constant}^({expression})等价, 除了M-变量的位异或运算
 和赋值都只在运动命令的开始和结束时才执行。数学表达式在程序行解释时已被作
 为异步表达式计算出来。

此命令不能用于任何手轮复用形式的M-
变量 (TWB, TWD, TWR, TWS), 或任何双字形式的变量 (L, D, F)
。

请记住如果在下一次运动前又使用了M-
变量, 则你将不会得到此命令赋给M-变量的值。

例子 M20^=\$FF ; M20所有位赋1
 M99^=\$80 ; M99的7位, 其余不变

参考 PMAC如何执行运动程序, 同步变量赋值 (写运动程序)
 程序命令M{constant}=, M{constant}==, M{constant}&=, M{constant}|=。

M{data}

功能 机器码 (M-code)

类型 运动程序

语法 M{data}

{data}是个在0.000~999.999间的浮点常数或表达式, 说明程序号和跳转的行数。

备注 PMAC将此行翻译为命令CALL 10n1.({data'}*1000), n是{data}的百位数字,
 {data'}是没有百位数的{data}值。此命令产生一个带返回的跳转到运动程序10n1
 或指定行。(程序10n1常添入系统设计者认为合适的机器码。{data'}值的范围为
 0.0~99.999, 对应行号N0~N99999)

此结构允许用10n1运动子程序完成自己的M-
code机床例程程序。参数可用来传递接着的M-code的一个或多个{letter}
{data},就象CALL和READ指令一样。

多数用户只用到0~99的M-code, 这就会只用到PROG
1001, 从而使得{data'}和{data}指向相同的行数。

例子 M01跳转到PROG 1001的N1000
 M12跳转到PROG 1001的N12000

M115跳转到PROG 1011的N15000

参考 程序命令CALL{data}, D{data}, M{data}, T{data}, RETURN

MACROAUXREAD

功能 读MACRO附件参数值

类型 后台PLC（非运动程序，PLC1，或已编译的PLC）

语法 MACROAUXRED{NodeNum}{ParamNum}{Variable}
MXR{NodeNum}{ParamNum}{Variable}
{NodeNum}是0~15范围的整型常数，用于指明结点的附属号
{ParamNum}是0~65535范围的整型常量，用于指明结点的辅助参数号
{Variable}是将进行值拷贝的PMAC的变量名（I，P，Q或M）

备注 此命令允许PMAC将辅助寄存器的值通过MACRO串写入辅助结点。此命令必须指明辅助结点的结点号、此结点的辅助参数号以及进行值拷贝的PMAC变量名。每一个命令行只能进行一个单一结点的一个辅助读写。

为了能够访问MACRO结点n的辅助寄存器，I1000的第n位必须置1。

当辅助结点返回错误消息或在32个伺服周期内没有反应时，PMAC会记录此错误状况。全局状态寄存器X:\$0003的第5位是用来报告MACRO附件的通讯错误的。寄存器X:\$0798保留有错误值。\$010000为超时错误，当辅助结点报错时为\$xxxxFE，其中xxxx是辅助结点返回的16位错误代码。

例子 MACROAUXWRITE1, 24, P1; 将P1的值写入结点1的参数24
MXW5, 128, M100 ; 将M100的值写入结点5的参数128

参考 再线命令MACROAUX, MACROAUXREAD, MACROAUXWRITE
程序命令MACROAUXREAD

N{constant}

功能 程序行标号

类型 运动程序（PROG和ROT）

语法 N{constant}
{constant}是个0~262.143的整型变量($2^{18}-1$)

备注

这是一行的标号，可以让由GOTO，GOSUB，ALL，G.M.T或D声明或B命令跳转执行到此标号处。

一行可以只有个标号，如果用户希望跳至此行。标号可以不按顺序，但必须写在该行的开始。请记住每定位一个标号将占用部分PMAC的存储空间。

经常在每个运动程序的开始默认为N0。在读程序时明确N0是十分有用的。在别处设置N0不仅会使程序阅读者困惑而且是毫无用处的。

例子 N1
N65537 X1000

参考 编写运动程序之子例程和子程序
在线命令B{constant}

NORMAL

功能	定义一个平面圆弧插补和切削半径补偿的标准向量
类型	运动程序（PROG 和ROT）
语法	<p>NORMAL{vector}{data} [{vector}{data}...] NRM {vector}{data} [{vector}{data}...]</p> <p>{vector}是I, J, K之中的一个字母，表征向量中分别表示与X, Y, Z平行的部分。{data}是个常量或表达式表示重要的向量组成部分。</p>
备注	<p>此命令定义了XYZ-空间圆插补和切削半径补偿的平面方向，设置了垂直于此平面的标准向量。</p> <p>向量由I（X轴），J（Y轴）和K（Z轴）组成。向量组成部分的比率决定了标准向量的方向因素是不重要的，因为它定义了是按时钟方向的圆弧运动和切削补偿抵偿方向。PMAC使用右手法则，即右手坐标系。</p>
例子	<p>在主要的平面上定义一个圆的标准方法：</p> <p>NORMAL K-1 ; XY平面—等价于G17 NORMAL J-1 ; ZX平面—等价于G18 NORMAL I-1 ; ZX平面—等价于G19</p> <p>用多于一个向量元素后，可以得到从主平面如下定义的斜曲平面：</p> <p>NORMAL I0.866 J0.500 NORMAL J25 K-25 NORMAL J(-SIN(Q1)) K(-COS(Q1)) NORMAL I(P101)J(P201)K(301)</p>
参考	<p>混合曲线运动，切削半径补偿（写运动程序） 笛卡耳坐标系（设置坐标系统） 程序命令 CIRCLE1, CIRCLE2, CC0, CC1, CC2</p>

O{constant}

功能	改变行标号
类型	运动程序（PROG 和ROT）
语法	<p>O{constant} {constant}是0~262.143（2¹⁸-1）</p>
备注	<p>这是运动程序标号的替代形式。它允许程序流程用GOTO, GOSUB, CALL, G.M, T, 或D, B命令执行跳转。PMAC将用N{constant}存储并报告，但O命令允许合法的传递程序缓冲。（N10和O10对PMAC是等同的。）</p> <p>一行可以只有个标号，如果用户希望跳至此行。标号可以不按顺序，但必须写在该行的开始。请记住每定位一个标号将占用部分PMAC的存储空间。</p>
例子	<p>O1 O65537 X1000</p>
参考	<p>编写运动程序之子例程和子程序 在线命令 B{constant} 程序命令O{constant}, GOTO, GOSUB, CALL, G.M, T.D.</p>

OR ({condition})

功能 条件或

类型 PLC程序

语法 OR({condition})

备注

此命令形式组成在PLC程序中可执行的扩展混合条件。它必须紧跟在IF，WHILE，AND或OR命令之后。这里OR是个布尔逻辑操作符，将该行的混合条件与程序中的上一行做运算。

当一行没有插入的混合条件时,该操作符的优先级是较低的，在一行开始其优先级也比AND操作符要低。（在AND的组合中有OR操作符。）

在运动程序中，混合条件可能不在一个程序行内，同时也不是复合交叉的程序行，因此这个命令在运动程序中是不允许的。

这个逻辑或，是对应条件而言的，请不要和按位或“|”操作符相混淆。

例子

```
IF (M11=1)      ; 这个分支每一周期当M11和M12不同时增加P1的
AND (M12=0)     ; 值，相同时减少P1的值。
OR (M11=0)
AND (M12=1)
P1=P1+1
ELSE
P1=P1-1
ENDIF
IF (M11=1 AND M12=0) ; 这与上面的程序相当
OR (M11=0 AND M12=1)
P1=P1+1
ELSE
P1=P1-1
ENDIF
```

参考 条件（程序计算特点）
写PLC程序
程序命令IF，WHILE，AND

P (constant)={expression}

功能 设置P变量值

类型 运动程序（PROG和ROT）

语法 P{constant}={expression}
{constant}是从0到1023的整型变量，表示P-变量的号码。
{expression}是将赋与P-变量值的数学表达。

备注 此命令将等号右侧的表达式值赋给P-变量。在该行执行时赋值，运动程序经常提前一两步执行后面的程序（计算在运动开始前完成的需要）计算。

例子

```
P1=0
P764=P20+P40
P893=SIN (Q100) -0.5
```

参考 PMAC如何执行运动程序，同步变量赋值（写运动程序）
 在线程序命令P{constant}={expression}
 程序命令I{constant}={expression}
 Q{constant}={expression}, M{constant}={expression}。

PRELUDE

功能 指定自动子程序访问程序

类型 运动程序

语法 PRELUDE1{command}
 PRELUDE0
 {command}是由CALL{constant},G{constant},M{constant},
 T{constant},D{constant}设定的子程序。

备注

PRELUDE1命令允许在每个顺序的运动（例如：X10Y10）或其它字母命令（例如：L10）之前而不是在运动程序的每行之前自动的插入一个子程序。这个过程相当于在顺序的运动之前插入一端运动程序文本或字母命令。

子程序调用可在PRELUDE1命令的CALL，G.M.T或D命令。在CALL或字母命令之后的值必须是个常量而不是个变量或表达式，并且不一定非要整型。如果例程用READ声明调用子程序，那么运动或字母命令本身可以作为子程序的参数。在PRELUDE1命令中的子程序或子程序调用中的运动命令将被作为运动命令直接执行，而不必要在它前面加上自动运行的子程序调用。

当运动或字母命令在程序行的开始或紧接在程序行标号后，则PMAC将只运行

PRELUDE1功能。如果在程序行中有别的命令在它的前面，则在运动或字母命令之前不会有PRELUDE1程序执行。如果命令是在子程序或子例程已有的由CALL或GOSUB调用，那么PRELUDE1功能将不被执行。

每个PRELUDE1命令会取代先前的PRELUDE1命令。同时屏蔽PRELUDE1调用也是不可能的，但是一个自动PRELUDE1调用可以被外在的子程序或子例程所覆盖。PRELUDE0显示所有自动子例程调用。

例子 PRELUDE1 CALL10 ; 在后面的运动之前插入CALL10
 X10 Y20 ; 在此运动之前完成CALL10
 X20 Y30 ; 在此运动之前完成CALL10
 OPEN PROG 10 CLEAR ; 子程序
 Z-1 ; 向下移动
 DWELL500 ; 保持位置
 Z1 ; 向上移动
 RETURN
 G71 X7 Y15 P5 ; G71调用PROG 1000 N71000
 X8 Y16 P5 ; 用PRELUDE指明G71的模式
 X9 Y15 P8 ; 用PRELUDE指明G71的模式
 G70 ; 停止模态封装周期
 OPEN PROG 1000
 N70000 ; G70子例程
 PRELUDE0 ; 停止PRELUDE调用
 RETURN
 N71000 ; G71子例程
 PRELUDE1 G71.1 ; 用PRELUDE将G71封装
 RETURN
 N71100 ; G71形式上执行的是G71.1例程
 READ (X, Y, P) ; 读入与X, Y, P相关的值
 {与运动有关的参数}

RETURN

参考 子例程和子程序（写运动程序）
运动命令CALL, GOSUB, READ, G, M, T, D

PSET

功能 重新定义当前轴位置（Position SET）

类型 运动程序

语法 PSET{axis}{data} [{axis}{data}...]
{axis}是指明轴X, Y, Z, A, B, C, U, V, W的字母
{data}是定义轴新的位置的常量或表达式

备注 本命令允许用户在程序运行当中更改轴位置的值。它与RS-274的G代码G92相当。

在命令结束后，将不会有任何运动—在当前轴位置的命令很少设置为指定值。

就本身来说，此命令改变了此轴所属的每个电机的位置偏差寄存器的值。此寄存器保存了轴零点和电机零点间的差值。

此命令通常自动使轴移动有一个间歇；不会在Pset中混合其他运动。对于运行中空间中的轴只对X,Y,Z轴更有效和灵活的偏置，请参照矩阵运算命令ADIS和IDIS。

例子 X10Y20
PSET X0 Y0 ; 设置位置（0, 0）
N92000READ（X, Y, Z） ; 在PROG 1000中实现G92
PSETX（Q124）Y（Q125）Z（Q126）；与G92的X, Y, Z相等。

参考 轴（建立坐标系统）
在线命令{axis}={expression}
程序命令ADIS, AROT, IDIS, IROT
M-变量的建议定义Mx64
内存映射寄存器D: \$08D3, 等等。

PVT

功能 设置位置-速度-时间模式

类型 运动程序（PROG和ROT）

语法 PVT{data}
{data}是个表明毫秒级时间段的常量或表达式（实际应用时PMAC会转化为最接近的整型值）。

备注 此命令定义设置位置-速度-时间运动模式，声明运动片段的时间。在这个模式下，程序中的每个运动片段必须指明轴的结束位置和速度。在获得起始位置、速度（从上个运动片段），结束位置、速度后，PMAC会计算单一的三次曲线（速度曲线）来驱动。

片段时间在持续的运动中可以随时更改，即可用另一个PVT命令，也可用TA命令，TS, TM和F设置与此模式有关联。

PVT命令是与其他运动模式（LINEAR, CIRCLE, SPLINE, RAPID）相对的。
在任何其它运动模式的命令将会使程序脱离PVT运动模式。
请参照手册中写运动程序的章节以获得此模式的细节。

例子	INC	； 增长模式，用距离指定运动
	PVT200	； 进入此模式，运动时间200毫秒
	X100: 1500	； 以1500单位/秒终点的速度平移100个单位长
	X500: 3000	； 以3000单位/秒终点的速度平移500个单位长
	X500: 3000	； 以1500单位/秒终点的速度平移500个单位长
	X100: 0	； 以0单位/秒终点的速度平移100个单位长
	PVT (P37)	

参考 位置-速度-时间运动模式（编写运动程序）
程序命令{axis}{data}:{data}..., TA, LINEAR, CIRCLEn, RAPID, SPLINE1.

Q{constant}={expression}

功能	设置Q变量值
类型	运动程序（PROG和ROT）
语法	Q{constant}={expression} {constant}是从0到1023的整型变量，表示Q-变量的号码。 {expression}是个数学表达式，用以给Q-变量赋值。
备注	本命令将等号右侧表达式的值赋给Q-变量。在运动程序中，任务是逐行顺序完成的，将任何运动程序的最终执行顺序排序是不必要的。如果它是在运动的中间，那么赋值将会在程序的一两步前执行完毕。（在程序中需要计算提前）

由于每个坐标系统中均拥有自己的Q-变量，所以在命令运行前指明哪一个Q-变量是非常重要的。当在运动程序中执行此命令时，所改变的是当前运动程序所在的坐标系统的Q-变量。

当在PLC程序中执行时，本命令对PLC程序最近一次执行ADDRESS命令所指明的坐标系统。如果从上电/重置到现在没有运行过ADDRESS命令，那么本命令运行所改变的是坐标系统1的Q-变量。

例子 Q1=3
Q99=2.71828
Q124=P100+ATAN(Q120)

参考 Q-变量（程序计算特征）
在线命令Q{constant}={expression}
程序命令ADDRESS, I{constant}={expression}, M{constant}={expression}, P{constant}={expression}

R{data}

功能	设圆半径
类型	运动程序（PROG和ROT）
语法	R{data} {data}是个表示以用户定长的圆弧运动的半径。一般是常量或表达式。
备注	

这部分命令定义了由该命令所指定的圆弧运动的半径。它对其他运动命令并无影响。（如果在命令行中既没有‘R’半径定义，也没有‘IJK’向量定义，那么运动将非常的线性化，即使在CIRCLE模式之中的运动。

如果{data}定义的半径比0大，那么圆弧运动的终点将使按指定半径的圆弧的角度小于或等于180度。如果{data}定义的半径比0小，那么圆弧运动的终点将使按{data}值的绝对值为半径的圆弧的角度大于或等于180度。

如果用AROT或IROT命令来缩放坐标系统，请不要用半径中心声明进行圆命令。半径是没有缩放的。请用I, J, K向量声明代替。

如果从起始点到结束点的距离大于两倍的{data}所定的数值，将可能不会有圆弧运动。如果距离大于两倍的{data}所定的数值同时小于Ix96（用户长度定义），PMAC将以螺旋线接近结束点。如果距离大于Ix96，PMAC将会停止程序并且报告运行时间错误。

例子 APID X0 Y0 ; 回到原位
 CIRCLE1 ; 顺时针圆弧模式
 X10 Y10 R10 ; 到（10，10）的四分之一圆
 X0 Y0 R-10 ; 回到（0，0）的四分之三圆
 X（P101） R（P102/2） ; 到（P101，0）的半圆

参考 混合圆运动（写运动程序）
 I-变量I13, Ix96
 程序命令CIRCLE1, CIRCLE2, {axis}{data}{vector}{data}...

RAPID

	功能
设置快速运动模式	
类型	运动程序（PROG和ROT）
语法	RAPID RPD
备注	<p>此命令将程序进入到所定义的到终点的指令轴运动的所谓手动模式中。此模式尽量将点间运动的耗时最少。持续的运动在这个状态中是不被混合的，不同的电机到达终点的时间也不必相同。</p> <p>本模式加速和减速是被电机的手动加速I-变量Ix19, Ix20和Ix21决定。如果全局I-变量I50设为0，则本模式的速度将被电机的手动I-变量Ix22决定。如果I50设为1，则它们被电机最大速度I-变量Ix16决定。只有做最大距离-速度比电机才用这样的速度；其他的电机将以较低的速度同时完成运动，从而使运动近似线形。</p> <p>RAPID命令是与其他运动模式（LINEAR, CIRCLE, SPLINE, PVT）相对的。在任何其它运动模式的命令将会使程序脱离RAPID运动模式。</p>
例子	RAPID X10 Y20 ; 尽快进入切削位置 M1=1 ; 打开切削开关 LINEAR X12 Y25 F2 ; 开始切削曲线 ... M1=0 ; 关闭切削开关 RAPID X0 Y0 ; 快速归原位
参考	快速运动模式（写运动程序）

I-变量I50, Ix16, Ix19, Ix22
 程序命令 LINEAR, CIRCLE, PVT, SPLINE

READ

功能 从子过程中读取参数

类型 运动程序（仅限于PROG）

语法 READ ({letter},{letter}...)
 {letter}可以是除N和O外任意英文字母，表示在程序行中的字母所对的变量值正是读入的值。

请记住：READ和左括号间没有空格。

备注

本命令允许子程序和子过程通过专用过程获得参数。这有点象一行余下的部分执行了这个例程（CALL, G,M,T,D），从指定的字母中取值后赋给坐标系统特别的Q-变量。对于字母表的第N个字母，其值将赋给Q（100+N）。

它将检索直至第一个不是READ列表中的字母或调用行的最后。每一个字母的值如果正确地传递给Q-变量，则Q100为1，请注意它已被读（字母表第N个字母对应第N-1位。

如果在最近的READ命令中有任何字母没有正确读入，Q100的对应位为0。

Q-变量和与之联系的Q100的对应位：

字母	对应变数	Q100对应位	十进制位值	十六进制位值
A	Q101	0	1	\$01
B	Q102	1	2	\$02
C	Q103	2	4	\$04
D	Q104	3	8	\$08
E	Q105	4	16	\$10
F	Q106	5	32	\$20
G	Q107	6	64	\$40
H	Q108	7	128	\$80
I	Q109	8	256	\$100
J	Q110	9	512	\$200
K	Q111	10	1,024	\$400
L	Q112	11	2,048	\$800
M	Q113	12	4,096	\$1000
N*	Q114*	13*	8,192*	\$2000*
O*	Q115*	14*	16,384*	\$4000*
P	Q116	15	32,768	\$8000
Q	Q117	16	65,536	\$10000
R	Q118	17	131,072	\$20000
S	Q119	18	262,144	\$40000
T	Q120	19	524,288	\$80000
U	Q121	20	1,048,57	\$100000
V	Q122	21	2,097,15	\$200000
W	Q123	22	4,194,304	\$400000

X	Q124	23	8,388,608	\$800000
Y	Q125	24	16,777,216	\$1000000
Z	Q126	25	33,554,432	\$2000000

* 不可用

任何除了N或O（它们被保留为行标（并且只能用在任何行的开始））外的任何字母均可被READ命令接受。如果一个字母的值从调用行读入数值，则以此字母为标志的函数将被覆盖，所以字母很少用来传递参数给子例程。如果有剩下的字母没有被调用行用完，那么在从子例程执行完并返回后，这些字母仍与原来的函数关联。

例子

在标准的机床代码中，例如一个两秒的DWEll将以G04X2000对应。在PMAC中，一个G04被解释为CALL到程序1000的行号N04000去，因此为了使这个功能执行正确，PROG1000可以带有下列的代码：

```
N04000 READ (X)
DWEll (Q124)
RETURN
```

同理，在标准的机床代码中，将当前分配给轴的位置可以被转变为G92代码，跟下来的字母及新的值将改变任何轴（例如G92 X20 Y30）。这对于用G92命令给轴赋新值是非常重要的。因此PMAC子例程实现G92时用了PSET命令以查看是否有轴被指定。

```
N2000 READ (X)
IF (Q100 & $800000 > 0) PSET X(Q124)
IF (Q100 & $1000000 > 0) PSET Y(Q125)
IF (Q100 & $2000000 > 0) PSET Z(Q126)
RETURN
```

参考

子例程和子程序（编写运动程序）
程序命令CALL，GOSUB，G,M,T,D

RETURN

功能 从子例程返回到主程序跳转/结束点

类型 运动程序（仅限于PROG）

语法 RETURN
RET

备注 RETURN
命令告诉运动程序返回到调用它的跳转点。如果此程序是由单行命令（RUN）开始的，程序会停止执行，程序指针将重置到这个运动程序的顶端——控制将反还给PMAC的“操作系统”。

如果本例程是由运动程序的GOSUB，CALL，G,M,T或D命令开始的，程序将返回到紧接跳转点后面的程序。

如果CLOSE命令在运动程序的缓冲的最后，PMAC回自动的在程序的结尾后面加上ETURN命令。如果OPEN命令发给已有的运动程序缓冲，那么最后的RETURN命令将会自动取消。

例子 OPEN PROG 1CLEAR
X20 F10


```

X0
CLOSE          ; PMAC在这里放置一个RETURN
OPEN PROG      1000CLEAR
N0 RAPID RETURN ; 在单行例程完成后执行跳转返回
N1000  LINEAR RETURN ; 同上
N2000  CIRCLE1RETURN ; 同上
CLOSE          ; PMAC在此放置个RETURN

```

参考 子例程和子程序（编写运动程序）
 在线命令OPEN, CLOSE
 程序命令CALL, GOSUB, G, M, T, D

S{data}

功能 轴数据命令

类型 运动程序（PROG和ROT）

语法 S{data}
 {data}是个表达式或常数，将值传递给存储变量，以留做将来之用。

备注 此命令将{data}的值存在运动程序执行所在的坐标系的Q127中。除此之外别无他途。

它打算将轴速度数据传递给机床程序。控制轴的运算法则便是在各自的例程中使用Q127，例如：设置手动速度，电压输出。

与S{data}命令将信息传递给子例程不同，READ（S）将值放在坐标系统的Q119中。

例子 S1800 ; 将1800存进Q127
 S(P1) ; 将P1的值存进Q127
 G96 S50 ; 这里S-命令是G-代码调用的参数
 (PROG 1000) ; 这是子程序执行G96命令
 N96000 READ（S） ; 将50存入Q119

参考 Q-变量（程序计算特征）
 完成一个机床设计程序（编写运动程序）
 运动程序命令READ
 程序样例SPINDLE.PMC

SEND

功能 让PMAC发消息

类型 运动程序（PROG和ROT）； PLC程序

语法 SEND "{message}"
 SENDS "{message}"
 SNEDP "{message}"

备注 此命令将使PMAC从通讯端口发出一个特定的消息。这在调试程序是十分有用。它还可以用来做提示符或在特定条件下通知主机。

如果I62=0，PMAC将自动在消息的结尾加上传输回转字符（<CR>）。如果I62=0，PMAC则不会加字符；在这种情况下，SEND^M将用来发<CR>。

如果在端口上没有接受消息的主机，或主机没有准备好接收消息，则消息会进入等待队列。如果不少消息在队列中，那么程序会认定在消息被读之前，程序会因发送消息而停止运行。这在PLC程序中将SEND命令用在边缘触发条件时的通常错误。请看第三章

在串口上，可以使用将端口握手标志的I1=1，从而可以将消息发给一个不存在的主机。

SEND会在激活的通讯反应端口上传输，无论是串口、并口（PC总线或STD总线），VME总线还是ASCII DPRAM缓冲。

SEND经常不管串口是否是当前激活的反应端口而传输消息。

SENDP经常在并口（PC总线或STD总线）上传递消息而不管并口是否是当前激活反应端口。

实际上只有SENDV命令专门用来在VME总线上传递消息，在VME端口上SEND命令必须用激活的反应端口。

当PMAC上电或重置活的反应端口是串口。当有任何命令从总线端口传输进来，激活的反应端口变为总线口。PMAC必须在接到一个〈CONTROL-Z〉命令后才将激活的反应端口变为串口。

如果一个程序，特别是PLC程序，在上电/重置时发送消息会造成主计算机程序的混乱（例如PMAC可执行程序），使它在寻找PMAC的同时等待特殊的回应。

如果可能，特别是在PLC程序中，要使消息的发送比端口的物理发送速度要快。这样在PLC程序每次扫描SEND命令时发生。正因如此，这就有机会看到当SEND命令执行出错时，会立即在以后的扫描中不再执行这个SEND命令。

如果想让PMAC发送变量的值，用COMMAND命令代替，并且声明所引用的变量名。

例子

```
SEND "Motion Program Started"
SEND "DONE"
SENDP "Spindle Command Given"
IF(M188=1) ; C.S.1 警告下面的位设置？
  IF(P188=0) ; 但没有设成最后的扫描？（P188在M188后）
    SEND "Excessive Following Error" ; 注意操作符
    P188=1 ; 不要重复消息
  ENDIF
ELSE ; F.E.A位没有设
  P188=0 ; 为下次做准备
ENDIF
SEND "THE VALUE OF P7 IS:" ; PMAC发送消息串
CMD "P7" ; PMAC返回P7的值
```

参考

I-变量I1, I62
程序命令COMMAND, DISPLAY, SEND^{letter}
编写PLC程序

SEND^{letter}

功能 让PMAC发出控制字符

类型 运动程序（PROG和ROT）； PLC程序

语法 SEND^{letter}
SENDS^{letter}
SENDP^{letter}
{letter}下列字符的一个： @ABC...xyz[\]^_

备注 本命令让PMAC在一个通讯端口上发送特殊控制字符。这在打印和终端控制的代码编写上特别有用，尤其在与其它计算机的通讯上。

控制字符有对应的ASCII码值（0~31（\$1F）。这个{letter}所声明的值决定哪一个控制字符在命令执行时发送出去。发送出去的控制字符的值比{letter}的值要小64（\$40）。

可以用的字符及其对应值如下：

{letter}	Letter的值	控制字符	值
@	64	NULL	0
A	65	<CTRL-A>	1
B	66	<CTRL-B>	2
C	67	<CTRL-C>	3
...			
X	88	<CTRL-X>	24
Y	89	<CTRL-Y>	25
Z	90	<CTRL-Z>	26
[91	ESC	27
\	92		28
]	93		29
^	94		30
_	95		31

注意：不要将要发送的字符用引号包括起来（例如SEND “^A”），PMAC会试图将两个非控制字符和A发送出去，而不是本来希望的控制字符。

^

SEND命令会在任何激活的通讯端口上，无论是串口、并口（PC总线或STD总线）还是VME总线。

SEND经常不管串口是否是当前激活的反应端口而传输消息。

SENDP经常在并口（PC总线或STD总线）上传递消息而不管并口是否是当前激活反应端口。

实际上只有SENDV命令专门用来在VME总线上传递消息，在VME端口上SEND命令必须用激活的反应端口。

当PMAC上电或重置，激活的反应端口是串口。当有任何命令从总线端口传输进

来，激活的反应端口变为总线口。PMAC必须在接到一个〈CONTROL-Z〉命令后才将激活的反应端口变为串口。

如果可能，特别是在PLC程序中，要使消息的发送比端口的物理发送速度要快。这样在PLC程序每次扫描SEND命令时发生。正因如此，这就有机会看到当SEND命令执行出错时，会立即在以后的扫描中不再执行这个SEND命令。

SPLINE1

功能	将程序置于单一三次插值运动模式
类型	运动程序（PROG和ROT）
语法	SPLINE1
备注	<p>这个模态命令将程序设为三次插值模式中。在SPLINE1模式中，每个编程运动将进行TA时间长度（默认Ix87）--</p> <p>没有允许回馈速度声明。每个轴运动均计算为三次曲线轨迹，中间位置是松弛的，因此同时的混合运动的速度和加速度是非连续的。</p> <p>在连续运动的第一个运动之前，从停止到运动有个TA时间，在连续运动的最后，从运动到停止也会有TA的缓和时间。如果TA时间设定在运动的中间被修改，那么运动会停止，同时产生一个附加的TA1运动和TA2运动。</p> <p>本命令是与其他运动模式（LINEAR，CIRCLE，RAPID，PVT）相对的。在任何其它运动模式的命令将会使程序脱离SPLINE运动模式。</p>
例子	<pre>RAPID X10 Y10 SPLINE1 TA100 X20 Y15 X32 Y21 X43 Y26 X50 Y30 DWEELL100 RAPID X0 Y0</pre>
参考	<p>三次插值模式（编写运动程序）</p> <p>I-变量Ix87</p> <p>程序命令LINEAR，CIRCLE，PAPID，PVT，SPLINE2，TA</p>

SPLINE2

功能	将程序置于非单一三次插值运动模式
类型	运动程序 （PROG和ROT）
语法	SPLINE2
备注	<p>本命令将程序置于非单一三次插值运动模式。这种模式事实上除了TA时间可变以外与SPLINE1命令的单一三次插值模式没有区别。这使得SPLINE2在使用上比SPLINE1灵活，但计算的时间较多。</p>
例子	<pre>RAPID X10 Y10 SPLINE2</pre>

X20 Y15 TA100
X32 Y21 TA120
X43 Y26 TA87
X50 Y30 TA62
DWE1100
RAPID X0 Y0

参考 三次插值模式（编写运动程序）
I-变量I_{x87}
程序命令LINEAR，CIRCLE，PAPID，PVT，SPLINE2，TA

STOP

功能 停止运动执行

类型 运动程序（PROG）

语法 STOP

备注 本命令挂起程序运行，当由RUN或SETP命令开始时，程序寄存器将从所记录的下一行开始继续运行。因此在STOP命令后，可以用RUN或STEP继续下面的程序。

例子 A10 B10
A20 B0
STOP
A0 B0

参考 在线命令<CONTROL-Q>，Q，R，S
程序命令BLOCKSTART，BLOCKSTOP

T{data}

功能 机床选择代码

类型 运动程序

语法 T{data}
{data}是从0.000~999.999的浮点常数或表达式。表示程序的编号和要跳转的行数。

备注 PMAC将本命令解释为CALL
10n2.({data’}*1000)命令，n是{data}的百位数字，{data’}是{data}去除百位后的数字（数学上去100的模）。因此，本命令发生一个到程序10n2，指定行数的带返回的跳转。（程序10n2通常是用来完成系统设计的代码而用）。{data’}是0.0~99.999，对应行标N0~N99999的数值。

这个结构允许用自己的运动程序10n2来完成T-代码机床类型的应用。与CALL和READ命令相似，T-代码后面参数的传递可以用一组或几组{letter}{data}来完成。

多数用户将T-代码的值限制在0~99间，这就允许只用PROG 1002的程序代码，在程序标号直接声明时允许{data’}与{data}相等。

例子 T01跳到PROG1002的N1000

T12跳到PROG1002的N12000
T115跳到PROG1012的N15000

参考 程序命令CALL {data}, D{data}, M{data}, T{data}, RETURN

TA{data}

功能 设置加速时间

类型 运动程序 (PROG和ROT)

语法 TA{data}
{data}是表示毫秒级加速时间的常数或表达式

备注

本命令指定了混合运动间的加速时间，以及开始和结束运动的时间。在PVT和SP LINE1

模式运动中，通常是连续的加速和减速，它指定了运动片段时间。时间单位是毫秒。PMAC会在具体执行本命令时转换成最接近的整数级毫秒数。（在存贮该值时没有在缓冲里圆整）

请确认指定的加速时间（TA或2*TS）大于零，即使你只想用最大加速比参数lx17。在指定的加速时间为零时，会产生个被零除的错误。指定的最小时间为TA1 TS0

如果指定的S-曲线时间（从TS，或lx88获得）比TA的1/2要大，那么混合运动的加速时间将比指定S-曲线时间要大。

加速时间同时也是混合运动的最小时间；如果指定回馈比运动的时间非常的快以致于计算的运动时间比加速时间还短，或时间指定运动的时间比加速时间少，那么运动将会在加速阶段完成。这会使运动减慢。如果让TA控制运动时间，它必须比I13时间和I8周期要大。

在LINEAR模式（I13=0无运动分段）中，坐标系统的任何电机被要求超过它的最大加速比（lx17），加速时间会自动扩展

在TA命令之前执行的运动，将会用坐标系统I-变量lx87来作为默认的加速时间。

在执行TA命令时，PMAC会圆整指定的值到最接近整数的毫秒数。（在存贮该值时没有在缓冲里圆整。）

例子 TA100
TA (P20)
TA (45.3+SQRT(Q10))

参考 线形，圆插补运动，三次插值运动，PVT运动（编写运动程序）
I -变量lx87, lx88, lx17
程序命令TS, TM, PVT

TINIT

功能 初始化所选择的传递矩阵

类型	运动程序（PROG和ROT）
语法	TINIT
备注	<p>本命令会将有TSEL命令指定的传输矩阵做初始化，以让坐标系统设置为其特征矩阵。</p> <p>设置转角为0，缩放比为1，移动为0，这样坐标系统的XYZ点由坐标定义创建。P MAC仍将进行矩阵运算，即使它们没有意义。TSEL0将命令停止做矩阵计算。矩阵可以被ADIS，IDIS，AROT和IROT命令改变。</p>
例子	<pre>TSEL 4 ; 选择传输矩阵4 TINIT ; 初始化为特征矩阵 IROT 71 ; 由Q71-Q79决定做增量旋转/缩放</pre>
参考	<p>轴传输矩阵（编写运动程序）</p> <p>在线命令DEFINE TBUF</p> <p>程序命令TSEL, ADIS, AROT, IROT</p>

TM{data}

功能	设置运动时间
类型	运动程序
语法	<p>TM{data}</p> <p>{data}是浮点常量或表达式。用以表示毫秒级运动时间。TM最大值为2²³毫秒,最小为1毫秒。</p>
备注	<p>本命令建立了以后LINEAR或CIRCLE（混合）模式运动的运动时间。它超过以前的TM或F命令，并可被以后的TM或F命令所超过。它与RAPID，SPLINE，PVT模式运动无关，但到由这些运动返回混合运动模式之前，它仍然保持激活状态。</p> <p>加速时间是混合运动的最小时间；如果特殊运动的时间比加速时间要少，那么运动会在加速过程完成。这会使运动减慢。如果让TM控制运动时间，它必须比I13时间和I8周期要大。</p> <div style="border: 1px solid black; padding: 10px; margin-top: 10px;"> <p>对于I13=0的LINEAR模式运动，如果任何电机所要求的速度（距离/TM）大于它自身的加速时间限制（Ix16），那么坐标系中所有电机都会降低速度以使没有任何电机超过限制。</p> </div>
例子	<pre>TM30 TM47.635 TM(P1/3)</pre>
参考	<p>线形和圆插补运动（写运动程序）</p> <p>变量Ix16</p> <p>程序命令F, TA, TS, LINEAR, CIRCLE</p>

TS{data}

功能	设S-曲线加速时间
类型	运动程序（PROG和ROT）
语法	TS{data} {data}是表示毫秒级S-曲线加速时间的常数或表达式
备注	<p>本命令定义了开始和结束运动的加速时间，LINEAR和CIRCLE混合模式依赖于S-曲线加速。如果TS是0，那么TA的加速是常数，速度曲线是梯形。如果TS大于0，那么加速会从零开始以TS时间线形递增，然后在TA-TS前保持定值。（从TC开始），从TA时间线形递减至0（TA=2TS+TC）。如果TS与TA/2相等，那么加速全过程将是S-曲线形式。（TS大于TA/2将抛弃TA的值；总加速时间为2TS。</p>

对于PMAC没有间隔模式的（I13=0）的LINEAR运动模式，如果加速比超过了坐标系中任何电机的加速比Ix17，电机的加速时间会降低以使没有任何电机超过其最大加速比。

TS不影响RAPID，PVT或SPLINE模式运动，但返回混合运动后其值仍然有效。

请确认特殊的加速时间（TA或2*TS）比零大，即使你只想用最大加速比参数Ix17。在指定的加速时间为零时，会产生个被零除的错误。指定的最小时间为TA1TS0

在执行TS命令时，PMAC会圆整指定的值到最接近整数的毫秒数。（在存贮该值时没有在缓冲里圆整。）

在TS命令之前执行的运动，将会用坐标系统I-变量Ix88来作为默认的S-曲线加速时间。

例子 TS20
TS (Q17)
TS (39.32+P43)

参考 直线，圆插补运动(写运动程序)
I-变量Ix87, Ix88, Ix17, Ix13, Ix21
程序命令TA, TM, F, LINEAR, CIRCLE

TSELECT{constant}

功能	为X, Y和Z轴选择激活的传递矩阵
类型	运动程序（PROG和ROT）
语法	TSELECT{constant} {constant}是表示选择矩阵号的整数常量或表达式
备注	<p>本命令选择特定的矩阵以用来进行运动程序坐标系统X, Y, Z轴的变换。这个矩阵可以被TINIT, ADIS, AROT, IDIS, IROT命令进行平移、旋转、缩放变换直到下一个矩阵被选择。</p> <p>这个矩阵必须是由在线命令DEFINE TBUF创建的。该命令指定创建的矩阵的数量，并且必须不能少于TSEL所定的数（</p>

不能选择一个不存在的矩阵)。

TSEL0将不选择任何传输矩阵，这样可以节省时间。

例子 DEFINE TBUF 5 ; 创建5个传输矩阵
OPEN PROG 10 CLEAR
... TSEL 3 ; 选择传输阵
TINIT ; 将矩阵3初始化

参考 轴传输矩阵（编写运动程序）
在线命令DEFINE TBUF
程序命令TINIT, ADIS, AROT, IROT, IDIS

U{data}

功能 U-轴运动

类型 运动程序

语法 U{data}
{data}为浮点常数或表达式，表示用户标度下U-轴的位置或距离

备注 本命令会使U-轴运动（参照上面的{axis}{data}的描述）

例子 U10
U (P17+2.345)
X20 U20
U(COS(Q10)) V(SIN(Q10))

参考 程序命令{axis}{data}, A, B, C, V, W, X, Y, Z, CALL, READ

V{data}

功能 V-轴运动

类型 运动程序

语法 V{data}
{data}为浮点常数或表达式，表示用户标度下V-轴的位置或距离

备注 本命令会使V-轴运动（参照上面的{axis}{data}的描述）

例子 V20
U56.5V(P320)
Y10 V10
V(SQRT(Q20*Q20+Q21*Q21))

参考 程序命令{axis}{data}, A, B, C, U, W, X, Y, Z, CALL, READ

W{data}

功能 W-轴运动

类型 运动程序

语法	W{data} {data}为浮点常数或表达式，表示用户标度下W-轴的位置或距离
备注	本命令会使W-轴运动（参照上面的{axis}{data}的描述）
例子	W5 W(P10+33.5) Z10 W10 W(ABS(Q22*Q22))
参考	程序命令{axis}{data}，A，B，C，U，V，X，Y，Z，CALL，READ

WAIT

功能	暂停程序执行
类型	运动程序（PROG和ROT）
语法	WAIT
备注	<p>本命令可以作为WHILE的条件行上，以暂停程序执行以等待条件变为假。当条件变为假后，程序将从下一行程序开始继续执行。用WAIT命令可以允许不确定的暂停而无须用伺服命令（DWELL或DELAY）重复使用来延迟时间。当然，暂停的时间是无法预测的。</p> <p>当WHILE的条件为假时，WAIT允许快速恢复程序。同时，程序计时器在WAIT期间是停止运行的。这样就允许在位变为真（它可以用来进行事件触发或下一个运动）</p> <p>当PMAC执行WHILE{condition}WAIT命令时，停止每个实时中断直到条件为假。这与PLC0是十分相似的。这会占用过多的处理时间并且在一些比较挑剔的条件下PMAC的Watchdog计时器会同时运行几个运动程序来用WAIT命令或大的PLC0程序。例如：果条件只需每20毫秒检查一遍而不用实时中断，可以考虑用DWELL命令去在WHILE循环里反复执行。</p> <pre> WHILE ({condition}) DWELL20 ENDW </pre>
例子	<pre> WHILE (M11=0) WAIT ; 在机器输入设为1前等待 WHILE (M187=0) WAIT ; 在轴到位前等待 M1=1 ; 输出为1以打开钻机 </pre>
参考	I-变量Ix28 程序命令 DWELL ， DELAY ， STOP

WHILE（{condition}）

功能	条件循环
类型	运动程序（PROG和ROT）；PLC程序
语法	<pre> WHILE（条件） WHILE（条件）{执行语句} </pre> <p>{条件}由一个或多个条件{表达式}{比较符}{表达式}组成,由逻辑运算符</p>

AND或OR连接

{执行语句}是程序命令

备注

本命令允许当条件为真时重复执行命令。这是PMAC唯一的循环结构。它分为两种形式：（只在运动程序中有效）在条件为真的期间，与命令同一行的命令将会反复执行。不用ENDWHILE来停止循环。

WHILE（条件）{执行语句}

（在运动程序和PLC程序中有效）在命令的同一行中没有其它命令，它会反复顺序执行WHILE与ENDWHILE区间的命令行。

WHILE（条件）

{语句}

[语句...]

ENDWHILE

在运动程序中如果WHILE循环内没有运动、DWEELL、DELAY在内的命令，PMAC将会尽量执行两倍实时中断周期的循环（条件真）（只被两次跳转阻止更多的循环），这非常象PLC0。这会使后台任务得不到足够时间，甚至会触发看门狗定时。PMAC不会在发现循环条件为真两次或更多时试图在这样的“空”WHILE循环运动里混合运动。

在PLC程序里，WHILE的扩展条件是可以被由AND或OR命令连接的，是紧接在WHILE命令后面的。（这种结构在运动程序里是没有的。）每个程序行里的条件可以是简单的，也可以是复合的。在一行里的AND和OR操作符比不同行的AND和OR优先级要高。

例子

WHILE（P20=0）

...
ENDWHILE

WHILE（Q10<5 AND Q11>1）

ENDWHILE

WHILE（M11=0） WAIT ; 等条件真时再继续

INC

WHILE（M11=0 OR M12=0） X100 ; 等两个输入为真时再增加做与FOR/NEXT等
同的循环：

P1=0 ; 循环记数初始化

WHILE（P1<10） ; 记数超过再停止

X1000 ; 循环的动作

P1=P1+1 ; 循环记数增长

ENDWHILE ; 回到循环开始处

在PLC中做延时程序，以伺服周期计数器为时钟

P90=16777216 ; 记数翻转值（ 2^{24} ）

P91=M0 ; 设置计数器开始值M0（X: \$0）

P92=0 ; 时间记时

WHILE（P92<P93） ; 在到时前保持循环

P92=（M0-P91）%P90 ; 计算过去的时间

; 取模运算符（%）处理翻转

ENDWHILE ; 回到循环开始处

在PLC程序中的扩展复合条件

WHILE（M11=1 AND M12=1）

OR（M13=1 AND M14=1）

AND（P1>1）

...
ENDWHILE

参考 程序逻辑（编写运动程序、编写PLC程序）
PMAC如何执行运动程序（写运动程序）
程序命令AND, OR, IF, ELSE, ENDIF, ENDWHILE

X{data}

功能	X-轴运动
类型	运动程序
语法	X{data} {data}为浮点常数或表达式，表示用户标度下X-轴的位置或距离
备注	本命令会使X-轴运动（参照上面的{axis}{data}的描述）
例子	X10 X15 Y20 X (P1) Y30 X (Q10*COS (Q1)) Y (Q10*SIN (Q1)) X3.76 Z2.92 I0.075 K3.42
参考	程序命令{axis}{data}, A, B, C, U, V, W, Y, Z, CALL, READ

Y{data}

功能	Y-轴运动
类型	运动程序
语法	Y{data} {data}为浮点常数或表达式，表示用户标度下Y-轴的位置或距离
备注	本命令会使Y-轴运动（参照上面的{axis}{data}的描述）
例子	Y50 Y (P100) X35 Y75 X-0.221 Z3.475 Y(ABS(P3+P4)) A(INT(P3-P4))
参考	程序命令{axis}{data}, A, B, C, U, V, W, X, Z, CALL, READ

Z {data}

功能	Z-轴运动
类型	运动程序
语法	Z{data} {data}为浮点常数或表达式，表示用户标度下Z-轴的位置或距离
备注	本命令会使Z-轴运动（参照上面的{axis}{data}的描述）
例子	Z20

Z (Q25)

X10 Y20 Z30

Z23.4 R10.5

Z(P301+2*P302/P303)

参考

程序命令{axis}{data}, A, B, C, U, V, W, X, Y, CALL, READ