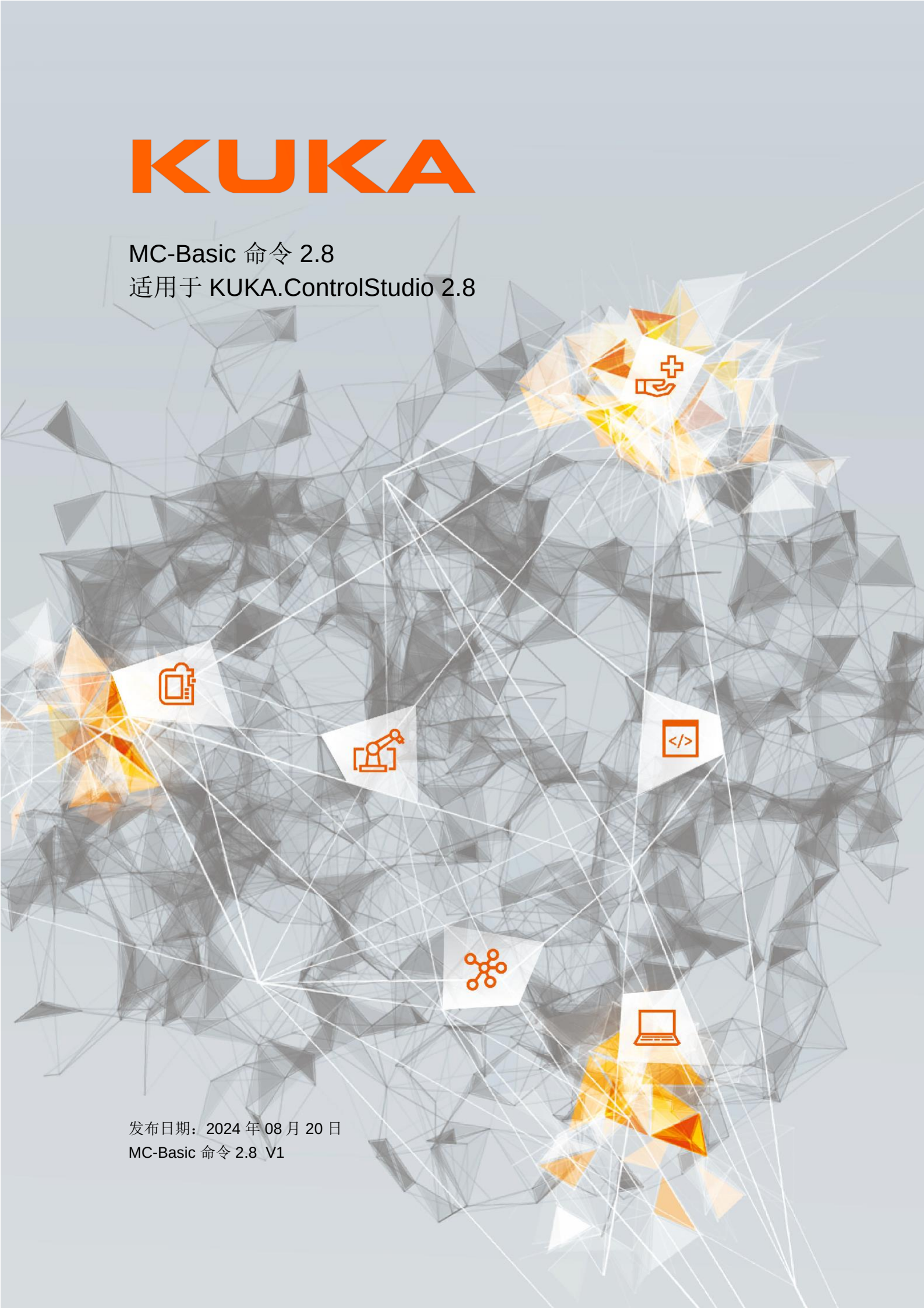
KUKA

MC-Basic 命令 2.8
适用于 KUKA.ControlStudio 2.8



发布日期：2024 年 08 月 20 日
MC-Basic 命令 2.8 V1

库卡机器人（广东）有限公司
僚莘路3号
佛山市顺德区北滘镇
广东省
中国

KIM-PS5-DOC
原版文件的译文

目录

常用命令汇总

系统命令

CLOCK	返回系统时钟周期数
ERROR	查询上次系统错误消息
ERRORNUMBER	查询上次系统错误的编号
IPADDRESSMASK	设置或查询以太网接口的 IP 地址和子网掩码
NAME	设置控制器的名称
PING	测试远程主机是否可访问。如果主机不可访问或未应答，则返回运行时错误。
RECORD	数据被记录到指定的文件中，然后可以检索该文件进行查看。
RECORDOFF	记录被停止。
RECORDON	开始记录特定数据。
TIME	查询或设置当前时间。
TMOUT	查询当前任务的 WAIT 指令是否超时

机器人控制命令

ATTACH	将任务连接到运动元素
CIRCLE	向指定的轴组/机器人发出圆周（圆弧）路径轨迹运动命令
DETACH	将任务与运动元素分离
GOHOME	将轴组移动到原点位置
HERE	返回实际的机器人笛卡尔坐标
ISMOVING	指示运动控制器是否处于活动状态
JUMP	以门形运动方式将机械臂从一个位置移动到另一个位置，包含三个点到点段
JUMP3	以门形运动方式将机械臂从一个位置移动到另一个位置，包含两个线段和一个点到点段
JUMP3CP	以门形运动方式将机械臂从一个位置移动到另一个位置，包含三个线段
MOVE	执行单轴或一组轴从当前位置到目标位置的点到点移动
MOVES	在世界坐标系空间中沿直线移动机器人

OPMODE	轴属性。此属性允许您更改驱动器运行模式。
POWER_OFF	设置机器人禁用
POWER_ON	设置机器人激活
PROCEED	继续执行被先前的 STOP 命令停止的运动
SMOVE	规划样条曲线路径运动，使机器人从给定的点移动到世界空间的目标点
STOP	停止轴组或轴运动
STOPPATH	停止机器人或轴的当前运动。当前的运动指令和等待中的运动指令将被存储
WAITFORMOTION	使程序等待，直到完成当前正在执行的运动

力矩命令

TORQUEADDCOMMAND	返回运动元素附加力矩命令值
TORQUECOMMAND	返回运动元素力矩命令值
TORQUEERROR	返回运动元素附加力矩误差值
TORQUEFEEDBACK	返回运动元素力矩反馈值
TORQUEGEARCOMMAND	返回作用在齿轮箱输出轴上的计算扭矩命令值
TORQUEGEARFEEDBACK	返回作用在齿轮箱输出轴上的计算扭矩反馈值

I/O 命令

SETIO	设置输入输出信号的值
GETIO	获取输入输出信号的当前值
PULSE	在数字 IO 信号上生成脉冲波
WAIT	停止程序，直到 io 条件满足

坐标系命令

BASE	查询系统中的当前基坐标系
TOOL	查询系统中的当前工具坐标系
WORKPIECE	查询系统中的当前工件坐标系
USERFRAME	查询系统中的当前用户坐标系
SELECTBASE	在程序中切换基坐标系
SELECTTOOL	在程序中切换工具坐标系
SELECTUSERFRAME	在程序中切换用户坐标系
SETTOOL	设置工具坐标系

SETUSERFRAME	设置用户坐标系

程序控制命令

PROGRAM_..._END_PROGRAM	界定任务中代码的主要部分
FUNCTION_..._END_FUNCTION	界定任务内的函数
SUB_..._END_SUB	界定任务内的子程序
CALL	调用子程序
IF_..._THEN_..._ELSE	根据指定条件的状态，允许一段代码或另一段代码执行
DO_..._LOOP	无限次执行一段代码
FOR_..._NEXT	多次重复循环中包含的语句
WHILE_..._END_WHILE	根据循环条件，界定 WHILE 循环
GOTO	无条件跳转到另一段代码
IMPORT	将库文件导入到任务中
SELECT_..._CASE	根据 <SelectExpression> 的值，允许多个代码段之一执行
REM	在代码中插入注释
SLEEP	将任务延迟指定的时间段（毫秒）
TRY_..._END_TRY	在一段代码中捕获同步错误

文件操作命令

CLOSE	释放文件句柄并关闭文件
DELETE	从闪存盘中删除文件
DELETE$	从闪存盘中删除文件
FILESIZE$	返回文件的大小（字节）
INPUT$	从指定的端口或文件返回字符串
LOC	返回在输入缓冲区中等待的字符数
OPEN_FILE	打开现有文件或创建新文件（在“写入”模式下）
SAVE	使用字符串表达式中给定的名称创建一个文件

套接字通信命令

ACCEPT	将 TCP 套接字绑定到指定端口并接受连接
CONNECT	从远程主机请求 TCP 连接

OPENSOCKET	创建 TCP 套接字并将套接字描述符设置到指定的设备句柄

变量命令

COMMON_OR_DIM_SHARED_..._AS_NOTE_OR_ERROR_ASCII_NUMBER	创建用户异常并定义相应的错误消息
COMMON_OR_DIM_SHARED_OR_DIM_..._AS_DOUBLE	将用户变量定义为 DOUBLE 类型
COMMON_OR_DIM_SHARED_OR_DIM_..._AS_JOINT_OF	将用户变量定义为 JOINT 类型
COMMON_OR_DIM_SHARED_OR_DIM_..._AS_LOCATION_OF	将用户变量定义为 LOCATION 类型
COMMON_OR_DIM_SHARED_OR_DIM_..._AS_LONG	将用户变量定义为 LONG 类型
COMMON_OR_DIM_SHARED_OR_DIM_..._AS_STRING	将用户变量定义为 STRING 类型
COMMON_SHARED_..._AS_SEMAPHORE	在系统中定义新的信号量变量
COMMON_SHARED_OR_DIM_SHARED_OR_DIM_..._AS_TYPE	定义新的用户数据类型，必须首先在配置文件中定义
VARLIST	返回系统中定义的变量和常量名称的列表
VARLIST$	返回系统中定义的变量和常量名称的列表
SETLOCALVAR	设置局部变量值

点操作命令

CASTJOINT	创建并返回通用关节空间坐标类型的点
CASTLOCATION	创建并返回通用笛卡尔空间坐标类型的点
CASTPOINT	创建并返回通用的点
DISTL	计算两点之间的距离（长度单位）
DISTR	计算两点之间的距离（角度单位）
GETARM	获得指定点位的手系（0/1/2）
PRINTPOINT	仅打印声明为笛卡尔空间坐标的变量
TOCART	将关节空间坐标转换为笛卡尔空间坐标
TOJOINT	将笛卡尔空间坐标转换为关节空间坐标
SETARM	设置本地坐标上的点都手

数值命令

ABS	返回参数的绝对值
ACOS	返回表达式的反余弦值（弧度）
ARRAYSIZE	返回数组指定维度的容量（1 表示第一维度，2 表示第二维度，等等）
ASIN	返回表达式的反正弦值（弧度）
ATAN2	需要知道角度和象限的绝对值时，使用 ATAN2
ATN	返回表达式的反正切值（弧度）
BIN$	返回数字的字符串表示（二进制格式）
COS	返回表达式（弧度）的余弦值
EXP	计算指数函数
HEX$	返回数字的字符串表示（十六进制格式）
INT	返回小于或等于数值表达式的最大长整型 (Long) 值
LOG	返回表达式的自然对数
ROUND	四舍五入整数
SGN	返回表达式的符号
SIN	返回角度（弧度）的正弦值
SQRT	返回数值表达式的平方根
TAN	返回给定角度（弧度）的正切值
LONGTOFLOAT	切换数据 long 类型为 float 类型
FLOATTOLONG	切换数据 float 类型到 long 类型
LONGTODOUBLE	切换数据 long 类型到 double 类型
DOUBLETOLONG	切换数据 double 类型到 long 类型

字符串命令

ASC	从 ASCII-8 字符串返回 ASCII 字符值，从 UTF-8 字符串返回 Unicode 值
CHR$	返回与给定 ASCII 值对应的单字符字符串
CLEARMSG	清除所有消息日志
INSTR	返回子字符串起始字符在字符串中的位置
LCASE$	返回所传递字符串的副本，所有大写字母转换为小写字母
LEFT$	返回字符串左侧指定数量的字符

LEN	返回输入字符串的长度，在 ASCII-8 字符串中为字符数，或在 UTF-8 字符串中为符号数
LTRIM$	返回删除开头所有空格后字符串的右侧部分
MID$	返回字符串中指定数量的字符，从位置 <start> 处的字符开始
PRINT	打印多个字符串和表达式，用逗号或分号分隔
RIGHT$	返回字符串右侧指定数量的字符
RTRIM$	返回删除结尾所有空格后字符串的左侧部分
SIZE	返回输入字符串占用的字节数
SPACE$	生成由指定数量的空格组成的字符串
STR$	返回数字的字符串表示
STRD$	返回双精度型 (Double) 数字的字符串表示
STRING$	生成具有指定字符数的新字符串
STRL$	返回长整型 (Long) 数字的字符串表示
TOASCII8$	将输入字符串的 UTF-8 编码格式转换为 ASCII-8 编码格式
TOUTF8$	将输入字符串的 ASCII-8 编码格式转换为 UTF-8 编码格式
TYPEOF	返回表示字符串类型的数字（0 表示无类型，1 表示 ASCII-8，或 2 表示 UTF-8）
UCASE$	返回所传递字符串的副本，所有小写字母转换为大写字母
UTF$	返回与给定 Unicode 值对应的 UTF-8 字符串
UTFSTRING$	生成具有指定符号数的新 UTF-8 字符串
VAL	返回输入字符串中的字符表示的实际值
SPLISTR	按指定的分割字符串符去分割字符串

运动参数命令

ACCELCMDCART	以笛卡尔坐标形式返回命令的机器人加速度
ACCELERATION	运动轨迹的加速度比率
ACCELERATIONROT	机器人的旋转加速度
ACCELERATIONSCALE	期望的加速度和减速度与此类型运动的最大可能速度相比的百分比
ACCELERATIONTRANS	机器人的平移加速度

ARMCMD	作业（命令）机器人配置
ARMFBK	实际（当前）机器人配置
BLENDMETHOD	要使用的圆滑方法（算法）
BLENDPERCENTAGE	指定圆滑对和指定圆滑方法的圆滑百分比值
BLENDENDPROTECTED	设置轨迹被保护的部分，使其不参与圆滑(从终点开始计算)
BLENDSTARTPROTECTED	设置轨迹被保护的部分，使其不参与圆滑(从起点开始计算)
BLENDDISTANCE	以路径长度为条件，设置圆滑运动的起始点
BLENDORIENTATION	以旋转角度为条件，设置圆滑运动的起始点
DECELERATION	运动轨迹的减速度比率
DECELERATIONROT	机器人的旋转减速度
DECELERATIONTRANS	机器人的平移减速度
JERK	运动元素的加加速度（加速度的变化率）
JERKROT	机器人的旋转加加速度
JERKTRANS	机器人的平移加加速度
JOFOLLOW	MOVE 笛卡尔点位时关节终点位置控制属性
POSITIONCOMMAND	运动控制器生成的位置命令
POSITIONERROR	位置跟踪误差，即位置命令与位置反馈之间的差值
POSITIONFEEDBACK	运动元素的实际位置
VELOCITYCOMMAND	速度命令
VELOCITYCOMMANDCARTESIAN	机器人 TCP 的当前笛卡尔命令速度
VELOCITYERROR	速度跟踪误差，即速度命令与速度反馈之间的差值
VELOCITYFEEDBACK	运动元素的实际空间速度
VELOCITYFEEDBACKCARTESIAN	机器人 TCP 的当前反馈速度
VELOCITYFINALROT	机器人的旋转速度
VELOCITYFINALTRANS	机器人的平移速度
VELOCITYMAX	运动元素的最大允许速度
VELOCITYMAXROT	机器人的最大允许旋转速度
VELOCITYMAXTRANS	机器人的最大允许平移速度

VELOCITYOVERRIDE	将速度乘以指定的百分比值得出的运动元素实际速度
VELOCITYOVERSPEED	轴的最大速度上限
VELOCITYRATE	运动元素速度最大比例系数
VELOCITYROT	定义机器人的旋转（方向）速度
VELOCITYROTVALUE	机器人工具尖端的旋转速度值（度/秒）
VELOCITYSCALE	指定期望的巡航速度与此类型运动的最大可能速度相比的百分比
VELOCITYTRANS	机器人的平移速度
VELOCITYTRANSFEEDBACKVALUE	机器人 TCP 反馈线性速度的大小
VELOCITYTRANSVALUE	机器人工具尖端的平移速度值（毫米/秒）
ORITYPE	SMOVE 运动指令中控制方向的属性
PASSINDEX	为 SMOVE 运动指定经过的点
XMAX	机器人笛卡尔运动在 X 轴方向的预计算最大限值
XMIN	机器人笛卡尔运动在 X 轴方向的预计算最小限值
YMAX	机器人笛卡尔运动在 Y 轴方向的预计算最大限值
YMIN	机器人笛卡尔运动在 Y 轴方向的预计算最小限值
ZMAX	机器人笛卡尔运动在 Z 轴方向的预计算最大限值
ZMIN	机器人笛卡尔运动在 Z 轴方向的预计算最小限值

系统内置应用程序命令

GRP_CLOSE_GRIPPER	关闭指定的抓手
GRP_OPEN_GRIPPER	打开指定的抓手
PAYLOADINERTIA	负载绕最后一个轴的惯量
PAYLOADLX	X 轴方向的负载质心距离
PAYLOADLY	Y 轴方向的负载质心距离
PAYLOADMASS	运动元素的负载质量
GETWARMSTATE	获取热机状态
PAYLOADMAX	运动元素的最大允许负载质量
PLT_GET_INDEX_STATUS	获取在一个托盘上物品的数量
PLT_MOVE_TO_ENTRY_POSITION	机器人移动到一个托盘的入口位置
PLT_PICK_FROM_PALLET	机器人移动到托盘中的下一个物件

PLT_PLACE_ON_PALLET	（持拿物件的）机器人移动到托盘中的下一个可用位置
PLT_RETRACT_FROM_ITEM	机器人移动到一个托盘中当前物件的远离点位置
PLT_SET_INDEX_STATUS	设置托盘上的物件数量
PLT_SET_INDEX_STATUS_EMPTY	将托盘上的物件数量设置为 0
PLT_SET_INDEX_STATUS_FULL	将托盘上的物件数量设置为满载
VCLOSETCP	手动关闭打开的套接字
VGETJOBDATA	从视觉站获取响应数据
VGETJOBERROR	从视觉站获取响应错误
VGETJOBSTATUS	从视觉站获取响应状态
VOPENTCP	打开视觉套接字
VPIXELTOPOS	将像素坐标转换为机器人坐标
VRECEIVEDATA	通过 TCP IP 协议接收用户定义的字符串
VRUNJOB	连接到视觉站并运行视觉任务
VRUNJOBFULL	获取响应数据或响应错误，而不处理响应状态
VSENDCMD	通过 TCP IP 协议发送用户定义的字符串
VSTOPJOB	停止一个正在运行的视觉站作业
DRIVE_ENABLE_I2T_PROTECTION	打开 I2T 保护功能
DRIVE_DISABLE_I2T_PROTECTION	关闭 I2T 保护功能

碰撞检测命令

PEAKVELERR	返回指定轴的峰值速度误差值
RESETPEAKVALUE	清空所有轴的峰值扭矩值和峰值速度误差值
SETVELERRTHRESHOLD	设置碰撞检测速度误差阈值
COLLISIONDETECT	设置碰撞检测开关
SETDRIVETORQUELIMIT	设置全部电机扭矩输出最大值
DRIVETORQUELIMIT	设置单个轴电机扭矩输出最大值
PEAKTORQUE	返回峰值扭矩
VELERRTHRESHOLD	设置碰撞检测的速度误差阈值
SETDEFVELERRTHRESHOLD	设置默认速度误差阈值，用于碰撞检测判断条件。

MC-Basic 命令参考

本节中将使用以下示例格式说明命令。

Command (example)

说明

此命令的说明。

注意

此命令的注意事项

（可能不存在）

缩写形式

命令语法的缩写形式。

（可能不存在）

语法

语法是显式使用命令时的格式。语法包含命令的几个关键字和参数。使用“<>”表示参数，使用“{ }”表示可选的关键字或参数。

域

域是此命令所属的运动元素。运动元素可以是系统/轴/运动元素/机器人/任务等。具有域的命令需要以“<Domain Name>.Command”的形式使用

（可能不存在）

参数

< Parameters >：参数说明、参数数据类型、参数数据限制

{< Parameters > }：可选的参数

（可能不存在）

返回值

返回值的说明

<return value>：返回值数据类型

（可能不存在）

限制

命令的使用限制

（可能不存在）

示例

有关如何使用命令的示例和说明

另请参见

带超链接的其他相关命令。您可以单击链接自动跳转到目标命令。

（可能不存在）

ABS

说明

此函数返回参数的绝对值。

语法

Abs(<expression>)

参数

<expression>: 任何有效的表达式，Double，±MaxDouble

返回值

返回参数的绝对值

限制

只读

示例

```
?Abs(-5.67)
VaSCARA = Abs(var2)
```

ABSOLUTE

说明

定义对运动元素的命令是绝对命令还是增量命令。在绝对模式下，对运动元素的位置命令是所需的绝对位置。在增量模式下，位置命令是所需的位置变化。此属性可在运动命令内使用以覆盖永久值。

缩写形式

<element>.Abs

语法

<element>.Absolute = <value>

?<element>.Absolute

域

ELEMENT

参数

<element>: 有效的运动元素

<value>: Long，绝对值

限制

要在任务中设置该值，必须将运动元素连接到该任务（使用 ATTACH 命令）。

示例

轴

A1.Absolute = False

?A1.Absolute

Move A1 100 absolute = True

机器人

SCARA.Absolute = False

Move SCARA {100, 50, 0, 0} absolute = True

另请参见

MOVE, MOVES, CURRENTABSOLUTE

ACCELCMDCART

说明

以笛卡尔坐标形式返回命令的机器人加速度。此变量是从电机命令位置或根据当前插补类型计算的每个采样周期。

语法

<point_variable> = <robot_name>.AccelCmdCart

? <robot_name>. AccelCmdCart

域

ROBOT

参数

<robot_name>: 任何有效的机器人

返回值

以笛卡尔坐标形式返回命令的机器人加速度

<point_variable>: Double

限制

独立属性，只读

示例

P1= Scara.AccelCmdCart?

Scara. AccelCmdCart

另请参见

ACCELERATIONCOMMAND, POSITIONCOMMAND

ACCELERATION

说明

此属性设置运动轨迹的加速度比率。

执行运动命令时，ACCELERATION 应小于或等于 ACCELERATIONMAX。若大于 ACCELERATIONMAX，则使用 ACCELERATIONMAX 值执行运动。

此属性可在运动命令内使用以覆盖永久值。

缩写形式

<element>.Acc

语法

<element>.Acceleration = <expression >

?<element>.Acceleration

域

ELEMENT

参数

<element>: 有效的运动元素

< expression >: Double，加速度比率值，大于 0

限制

加加速度和加速度之间的比率受以下关系限制：

*Jerk/Acc < 0.9 * pi/5T*

其中 T 是采样时间（秒）。对于 2 毫秒的周期时间，该值为 282.74。

使用 SMOOTHFACTOR 可自动计算加加速度。

要在任务中设置该值，必须将运动元素连接到该任务（使用 ATTACH 命令）。

示例

轴

A1.acceleration = 1e10

Move A1 100 acc = 2e10

机器人

SCARA.acceleration = 1e10

Move SCARA {100, 50, 0, 0} acc = 2e10

另请参见

ACCELERATIONMAX, ACCELERATIONCOMMAND, ACCELERATIONERROR, DECELERATION, JERK, JUMP, MOVE

ACCELERATIONCOMMAND

说明

此属性返回运动控制器生成的加速度命令。类型为 Double

缩写形式

<element>.AccelCmd

语法

?<element>.AccelerationCommand

域

ELEMENT

返回值

返回运动控制器生成的加速度命令

限制

只读

示例

轴

?A1.AccelCmd

机器人

?SCARA.AccelCmd

另请参见

VELOCITYCOMMAND, ACCELERATION, ACCELCMDCART, POSITIONCOMMAND

ACCELERATIONERROR

说明

此属性返回加速度跟踪误差，即加速度命令与加速度反馈之间的差值。此属性仅用于查询。与 PE 和 PEmax 不同的是，不针对最大阈值进行检查。计算使用位置误差延迟，以便从之前多个周期发出的命令中减去加速度反馈，如 POSITIONERRORDELAY 属性中所定义。

在某一轴中，加速度误差是加速度命令与加速度反馈之间的差值。

在某一轴组中，加速度误差是所有轴加速度误差平方和的平方根。

在机器人中，加速度误差使用工具尖端加速度命令与反馈之间差值的位置部分（仅 XYZ 值，方向部分不使用）来计算：

SQRT((AccelCmd{1}-AccelFbk{1})2 + (AccelCmd{2}-AccelFbk{2})2 +(AccelCmd{3}-AccelFbk{3})2)

缩写形式

<element>.AE

语法

?<element>.AccelerationError

域

ELEMENT

参数

<element>: 有效的运动元素

返回值

返回加速度跟踪误差

限制

只读

示例

轴

?A1.AE

机器人

?SCARA.AE

另请参见

ACCELERATION

ACCELERATIONFEEDBACK

说明

此属性返回运动元素的实际加速度。该值作为各个运动元素加速度的矢量返回。

缩写形式

<element>.AccelFbk

语法

?<element>.AcclerationFeedback

域

ELEMENT

参数

<element>: 有效的运动元素

返回值

返回运动元素的实际加速度

± MaxDouble

限制

只读

示例

?A1.AccelFbk

?SCARA.AccelFbk

另请参见

ACCELERATIONCOMMAND

ACCELERATIONMAX

说明

定义允许的最大运动元素加速度。

如果指定的加速度高于 ACCELERATIONMAX，系统会将该值设置为 ACCELERATIONMAX 并通知您。

缩写形式

<element>.AMax

语法

<element>.AccelerationMax = <expression>

?<element>.AccelerationMax

域

ELEMENT

参数

<element>: 有效的运动元素

< expression >: Double，最大加速度值，大于 0

限制

要在任务中设置该值，必须将运动元素连接到该任务（使用 ATTACH 命令）。

示例

轴

A1.Amax = 20e3

机器人

SCARA.Amax = 20e3

另请参见

ACCELERATION, ACCELERATIONSCALE, DECELERATIONMAX

ACCELERATIONMAXROT

说明

定义机器人的最大旋转加速度，用于 AROT 和 DROT。

该值仅限制笛卡尔运动插补（MOVES、CIRCLE）。此参数不影响轴插补的运动 (MOVE)。

缩写形式

<ROBOT>.amrot

语法

<ROBOT>.amrot=<numeric expression>

域

ROBOT

参数

< ROBOT >: 任何有效的机器人

<numeric expression>: Double，0.1 到 Maxdouble

限制

读/写，仅独立属性

示例

amrot = 6000 ‘set for default robot ---- SCARA

另请参见

ACCELERATIONROT, CIRCLE, MOVES

ACCELERATIONMAXTRANS

说明

定义机器人的最大平移加速度，用于 ATRAN 和 DTRAN。

该值仅限制笛卡尔运动插补（MOVES、CIRCLE）。此参数不影响轴插补的运动 (MOVE)。

缩写形式

<ROBOT>.amtran

语法

<ROBOT>.amtran=<numeric expression>

域

ROBOT

参数

< ROBOT >: 任何有效的机器人

<numeric expression>: Double，0.1 到 Maxdouble

限制

读/写，仅独立属性

示例

amtran = 6000 ‘set for default robot ---- SCARA

另请参见

CIRCLE, MOVES, ACCELERATIONTRANS

ACCELERATIONROT

说明

定义机器人的旋转加速度。与 ATRAN 一起，定义笛卡尔运动的加速度。

该值仅在两个运动命令中使用：MOVES 和 CIRCLE。在轴插补运动 (MOVE) 中，该值将被忽略。

该值不得大于 ACCLERATIONMAXROT。系统始终采用两者中较小的值，并向用户发送通知消息。

缩写形式

<ROBOT>.arot

语法

<ROBOT>.arot=<numeric expression>

域

ROBOT

参数

< ROBOT >: 任何有效的机器人

<numeric expression>: Double，0.1 到 Maxdouble

限制

读/写，独立/非独立属性

示例

arot = 6000 ‘set for default robot ---- SCARA

另请参见

ACCELERATIONMAXROT, CIRCLE, MOVES, ACCELERATIONTRANS

ACCELERATIONSCALE

说明

此属性指定期望的加速度和减速度与此类型运动的最大可能加速度相比的百分比。

运动生成器尝试在运动命令期间达到此加速度。运动生成器达到该值的能力受 ACCELERATION、DECELERATION、SMOOTHFACTOR 和最终位置值的限制。

此属性在运动命令内使用以覆盖永久值。

缩写形式

<element>.Ascale

语法

<element>. ACCELERATIONSCALE = <expression>

?<element>. ACCELERATIONSCALE

域

ELEMENT

参数

<element>: 有效的运动元素

< value >: Double，期望的加速度和减速度与最大值相比的百分比，0.1 到 100

限制

要在任务中设置该值，必须将运动元素连接到该任务（使用 ATTACH 命令）。

示例

轴

A1.AScale =30

Move A1 TargetPos Ascale = 20

机器人

SCARA.AScale =40

Move SCARA {100, 20, 0, 0} Ascale = 22

另请参见

ACCELERATIONMAX, ACCELERATIONRATE, VELOCITYSCALE, DECELERATION

ACCELERATIONTRANS

说明

定义机器人的平移加速度。与 AROT 一起，定义笛卡尔运动的加速度。

该值仅在两个运动命令中使用：MOVES 和 CIRCLE。在轴插补运动 (MOVE) 中，该值将被忽略。

该值不得大于 robot.ACCELERATIONMAXTRANS。系统始终采用两者中较小的值，并向用户发送通知消息。

缩写形式

<ROBOT>.atran

语法

<ROBOT>.atran=<numeric expression>

域

ROBOT

参数

< ROBOT >: 任何有效的机器人

<numeric expression>: Double，0.1 到 Maxdouble

示例

atran = 6000 ‘set for default robot ---- SCARA

另请参见

ACCELERATIONMAXTRANS, ACCELERATIONROT, CIRCLE, MOVES

ACCEPT

说明

将 TCP 套接字绑定到特定端口并接受连接，监听 TCP 套接字。<port number> 定义 TCP 端口。

语法

Accept({#<listening device number>}, #<accepting device number>, <port number>)

参数

<listening device number>: Long，1 到 255，可选参数，如果未设置该参数，监听端口和通信端口将默认使用相同的设备序号

<accepting device number>: Long，1 到 255

<port number>: Long

限制

仅任务

示例

Accept(#1,65456)

OpenSocket Options=1 as #1 'explicit socket open for listening for port

OpenSocket Options=1 as #2 'explicit socket open for listening accepting a new connection

Accept(#1, #2, 6002)

另请参见

CLOSE, CONNECT, OPENSOCKET, PING, IPADDRESSMASK

ACOS

说明

此函数返回表达式的反余弦值（弧度）。

语法

Acos(<expression>)

参数

<expression>: 任何有效的表达式，Double，-1~+1

返回值

返回表达式的反余弦值（弧度）

限制

只读

示例

Value = Acos(0.5)

另请参见

ATN, ATAN2, ASIN, COS

ARMCMD

说明

定义作业（命令）机器人配置。当目标位置给定为 Location 变量时，它即为笛卡尔点。此标志确定目标位置的关节空间坐标将采用哪种方案。

将其设置为零 (AUTO) 表示采用当前 ARMFBK 作为值。

机械臂标志根据以下内容定义 LEFTY 或 RIGHTY 配置：

对于 SCARA 机器人：

j2.pcmd > 0 -> Lefty

j2.pcmd = 0 -> Auto

j2.pcmd < 0 -> Righty

在不同的运动学模型中，此标志的计算方式不同。

注意

此标志不为 ARMFBK 时，不允许进行笛卡尔运动（MOVES 或 CIRCLE）。

缩写形式

<ROBOT>.acmd

语法

?<ROBOT>.armcmd

<ROBOT>.armcmd = <numeric expression>

域

ROBOT

参数

< ROBOT >: 任何有效的机器人

<numeric expression>: Long

限制

读/写，独立/非独立属性

示例

scara.armcmd=1

MOVES SCARA {1,2,3,4} armcmd=0

CIRCLE SCARA angle = … circlecenter = … armcmd=1

另请参见

ARMFBK

ARMFBK

说明

返回实际（当前）机器人配置。

在 SCARA 机器人中，每个笛卡尔位置均可通过第二个轴的两个不同角度获得。

这两种方案代表机器人反向运动学方程式的两个解，或者说是机器人配置。

通常，它们称为“lefty”和“righty”机器人配置。

此标志指示采用哪一种配置。此外，当此标志与命令的标志 (armcmd) 不同时，则无法进行直线运动。

例如，在 SCARA 运动学模型中，根据第二个轴的位置符号计算此标志。

j2.pfb > 0 -> Lefty

j2.pfb = 0 -> Undefined(Auto)

j2.pfb < 0 -> Righty

缩写形式

<ROBOT>.afbk

语法

?<ROBOT>.armfbk

域

ROBOT

参数

< ROBOT >: 任何有效的机器人

返回值

返回实际（当前）机器人配置

<return value>: Long

0 - 未定义

1 - Lefty

2 - Righty

限制

只读，仅独立属性

示例

?scara.armfbk

另请参见

ARMCMD

ARRAYSIZE

说明

返回数组指定维度的容量（1 表示第一维度，2 表示第二维度，等等）。

如果维数大于为数组定义的维数，则返回零。如果维数为零或负数，也返回零。

语法

<variable> = arraysize (<array_name>, <long_expression>)

参数

<array_name>: 任何有效的数组变量

<long_expression>: 给定的维度索引，Long

返回值

返回数组指定维度的容量（1 表示第一维度，2 表示第二维度，等等）

<variable>: 数组的大小，Long

示例

```
?ArraySize (Arr, 2)
Size = ArraySize (Arr, X)
```

ASC

说明

ASC 从 ASCII-8 字符串返回 ASCII 字符值，从 UTF-8 字符串返回 Unicode 值。

语法

?ASC(<string>{,<Index>})

参数

<string>: String，待选字符的字符串表达式。如果字符串为空，该函数返回值 0

<index>: 定义要选择的 ASCII-8 字符或 UTF-8 符号在 <string> 中位置的可选实际值。Long，0 到 MaxLong

返回值

从 ASCII-8 字符串返回 ASCII 字符值，从 UTF-8 字符串返回 Unicode 值

<return value>: Double，对于 ASCII-8 <string> 为 0 到 127，对于 UTF-8 <string> 为 0 到 MAXLONG

限制

只读

示例

?ASC("example",3)

97‘returns the ASCII code for the letter "a"

?ASC("example")

101‘returns the ASCII code for the letter "e"

Common Shared UTF8Str as String of UTF8

UTF8Str = UTF$(196) + UTF$(197) + UTF$(198)

?ASC(UTF8str,2)

197

另请参见

INSTR, SPACE$, STRING$, UTF$, CHR$

ASIN

说明

此函数返回表达式的反正弦值（弧度）。

语法

Asin(<expression>)

参数

<expression>: 任何有效的表达式，Double，-1~+1

返回值

返回表达式的反正弦值（弧度）

限制

只读

示例

Value = Asin(0.5)

另请参见

ACOS, ATN, ATAN2, SIN

ATAN2

说明

需要知道角度和象限的绝对值时，使用 ATAN2。第一个表达式是 y 轴上的值，第二个表达式是 x 轴上的值。结果给出 ± π 弧度范围内的实际角度。

语法

Atan2(<y_expression>, <x_expression>)

参数

< y_expression >: y 轴上的值，任何有效的表达式，Double，± MaxDouble

< x_expression >: x 轴上的值，任何有效的表达式，Double，± MaxDouble

返回值

返回表达式的反正切值（弧度）。结果给出 ± π 弧度范围内的实际角度。

限制

只读

示例

?Atan2(1, -1)

返回 0.75×π

?Atan2(-1, 1)

返回 -0.25×π

另请参见

ACOS, ASIN, ATN, TAN

ATN

说明

ATN 返回表达式的反正切值（弧度）。表达式为 y 轴上长度与 x 轴上长度之比，结果在 $\pm\pi/2$ 弧度范围内。

语法

Atn(<expression>)

参数

<expression>: 任何有效的表达式，Double，±MaxDouble

返回值

返回表达式的反正切值（弧度）

限制

只读

示例

```
Program
Dim x as double
Dim y as double
Dim Theta as double
x=1
y=2
Theta=ATN(y/x)
End Program
```

另请参见

ACOS, ASIN, ATAN2, TAN

ATTACH

说明

将任务连接到运动元素（轴组或轴）。对于从任务内对运动元素执行某些操作（例如 MOVE）来说，连接是必需的。连接可阻止其他任务访问该运动元素。

在终端中对运动元素执行运动指令时，运动元素将与终端建立隐式连接。

语法

Attach <element>

参数

< element>: 定义的运动元素

限制

已被任务连接的运动元素无法被其他任务连接。

示例

Attach SCARA

另请参见

DETACH, ATTACHEDTO, ATTACHTO, ATTACHTO$, DETACHFROM, DETACHFROM$

ATTACHEDTO

说明

返回给定运动元素已连接任务的名称。

注意

如果轴组没有连接，但属于该轴组的轴已连接，则返回活动轴及其任务列表。
如果运动元素未连接到任何程序，则返回空字符串。

语法

?<element>.AttachedTo

域

ELEMENT

参数

<element>: 有效的运动元素

返回值

返回给定运动元素已连接任务的名称

<return value>: String

限制

只读

示例

轴

?A1.AttachedTo

机器人

?SCARA.AttachedTo

另请参见

ATTACH, DETACH, ATTACHTO$, DETACHFROM, DETACHFROM$

ATTACHTO

说明

将给定运动元素连接到给定任务。与 ATTACH 功能相同，但是将会指向外部任务。

语法

AttachTo <element> File = <task>

参数

<task>: 文件规格，包括文件名和扩展名。

<element>: 运动元素规格可以是轴、轴组或机器人。

示例

attachto a16 File = ttt.prg

detachfrom a16 File = ttt.prg

另请参见

ATTACH, DETACH, ATTACHTO$, DETACHFROM, DETACHFROM$

ATTACHTO$

说明

将给定运动元素连接到给定任务。与 ATTACH 功能相同，但是将会指向外部任务。

语法

AttachTo$ <element name> File = <task name>

参数

<task name >: 包含文件规格的字符串，包括文件名和扩展名。

<element name>: 运动元素规格可以是轴、轴组或机器人。

示例

attachto$ a16 File = "ttt.prg"

detachfrom$ a16 File = "ttt.prg"

另请参见

ATTACH, DETACH, ATTACHTO, DETACHFROM, DETACHFROM$, ATTACHEDTO

AVERAGELOAD

说明

返回以 0.5 秒间隔测得的 CPU 上的平均实时负荷，以百分比表示。

缩写形式

Sys.ALoad

System.ALoad

语法

?Sys.AverageLoad

?System.AverageLoad

域

SYSTEM

返回值

返回以 0.5 秒间隔测得的 CPU 上的平均实时负荷

<return value>: Long，0 到 100

限制

只读。不能在事件条件中显示。

示例

```
?System.ALoad                    ‘returns 45
```

AXISLIST

说明

返回系统中定义的轴名称的逗号分隔列表。如果给定 Optional String 并使用通配符，查询将返回特定的轴。

语法

?AxisList {<Optional string>}

参数

<Optional string>: String。

返回值

返回系统中定义的轴名称的逗号分隔列表

示例

?AxisList

在四轴系统中返回 a1、a2、a3、a4

?axislist a*

另请参见

PLSLIST, GROUPLIST, TASKLIST, VARLIST, VARLIST$

BASE

说明

基坐标系设置了指定位置用于系统基坐标变换。

它根据 WORLD 参考定义单元格中机械臂的位置和方向。

默认的基坐标系变换是 NULL 变换，可表示为 SCARA.base = #{0,0,0,0}

语法

Base = <robot location point>

?Base

域

ROBOT

参数

<robot location point>: Location

返回值

返回世界坐标系中的基坐标系位置

<return value>: Location

示例

Scara.Base = #{90, 180, 0, 0}

?Scara.MachineTable:WorkPiece:Base:Tool

打印以下内容： {90 , 90 , 90 , 180}

另请参见

TOOL, WORKPIECE, USERFRAME, SELECTBASE

BAND

说明

BAND 是位运算符。BAND（按位与）两个二进制数的对应位进行比较，如果两个二进制数的对应位都是 1，则结果的对应位为 1，否则为 0。

语法

<Number> BAND < Number >

参数

<Number>: Long，MaxLong 到 MaxLong

返回值

<return value>: Long

示例

? BIN$(0b1010 band 0b0010)

打印以下内容：0b10

另请参见

BOR, BNOT, BXOR

BIN$

说明

BIN$ 返回数字的字符串表示（二进制格式）。

语法

BIN$(<number>)

参数

<number>: Long，MinLong 到 MaxLong

返回值

返回数字的字符串表示（二进制格式）

<return value>: String

示例

myStr = Bin$(58699)

?myStr

打印以下内容：1110010101001011

另请参见

HEX$

BLENDDISTANCE

说明

定义当前运动将与下一个运动进行圆滑的运动距离。

该参数仅适用于直角坐标运动，不适用于关节运动。它由用户指定一个期望的值，但是实际情况不一定能与期望一致，因为实际的圆滑距离也会收到下一段运动长度的影响。此外，允许用户给出不小于零的任何值，但在内部它将不大于参与圆滑运动的任意一段轨迹的半长（平移和方向）。

注意

如果设置的数值大于最大方位偏差，则使用允许的最大值。

BlendDistance 的优先级高于 BlendPercentage，如果设置了 BlendDistance，则 BlendPercentage 将被覆盖。如果同时设置了 BlendDistance 和 blendorientation，机器人将根据首先触发的条件开始圆滑运动。

仅适用于 BlendMethod = 1 (SP 混合)。在非 SP 圆滑或 PTP 运动的情况下，它的值将被忽略(不会报错)。

缩写形式

BlendDis

语法

<motion> <element> <position> BlendDistance = <value>

参数

<element>: 有效运动元素

< value >: Double, 要与下一个运动圆滑的运动路径长度，从 0 到 Maxdouble，单位为毫米

限制

非独立属性

示例

轴

Move A1 10 BlendDistance =50

机器人

Move SCARA {10,10,0,0} BlendDis =50

另请参见

BLENDMETHOD, BLENDORIENTATION, BLENDPERCENTAGE

BLENDENDPROTECTED

说明

设置当前运动受保护部分的路径长度，该部分不参与圆滑运动（从终点计算）。当前动作的受保护部分将与上一个或下一个动作完全分开执行。

BlendEndProtected 优先级高于 BlendDistance 和 blendorientention，如果 BlendEndProtected 被设置(不是 0)，当前运动的结束部分将不会与下一个运动进行圆滑，当前运动的起始部分将与上一个运动进行圆滑。

注意

如果该值大于路径长度，则使用路径长度代替，0 表示不保护任何内容。

仅适用于 BlendMethod = 1 (SP 混合)。在非 SP 圆滑或 PTP 运动的情况下，它的值将被忽略(不会报错)。

缩写形式

BLENDED

BLENDED

语法

<element>.Blendendprotected = <value>

参数

<element>: 有效运动元素

< value >: Double, 当前运动受保护部分的路径长度（从终点计算），从 0 到 Maxdouble，单位为毫米

限制

非独立属性

示例

轴

Move A1 10 BlendEndProtected=50

机器人

Move SCARA {10,10,0,0} BlendEd =50

另请参见

BLENDMETHOD, BLENDSTARTPROTECTED, BLENDPERCENTAGE

BLENDMETHOD

说明

选择要使用的圆滑方法（算法）。

0 - 无圆滑

1 - SP（叠加）

2 - AI（高级插补）

叠加圆滑允许用户在从运动的中点开始的任意位置开始圆滑运动。

高级插补将一系列的指令点作为路径点进行圆滑，最终只执行一条完整的运动。

注意

请注意，通过将 BlendMethod 的值从 2 更改为任何其他(0,1)，清除所有之前存储的 AI 路径点。

语法

<element>.BlendMethod= <value>

域

ELEMENT

参数

<element>: 有效运动元素

< value >: Long，圆滑模式

限制

如果是 AI：

如果输入其他运动命令 (CIRCLE)，将返回一个错误，并且将不会执行该命令。

如果独立属性 starttype 的值设置为 immediate 或 super immediate，将返回错误。

示例

轴

A1.BlendMethod = 1

?A1.BlendMethod

机器人

SCARA. BlendMethod = 1

?SCARA.BlendMethod

另请参见

BLENDSTARTPROTECTED, BLENDENDPROTECTED, BLENDPERCENTAGE, MOVE, CIRCLE, MOVES, BLENDDISTANCE, DOUBLEMODEPERCENTAGE, BLENDORIENTATION

BLENDORIENTATION

说明

根据转向角度设置当前运动与下一个运动产生圆滑的距离。

该参数仅适用于笛卡尔运动，不适用于关节运动。 它由用户指定所需的值，但不能始终保证，因为实际的混合距离也取决于第二运动长度。 此外，允许用户给出不小于零的任何值，但在内部它将不大于参与圆滑运动的任意一段轨迹的半长（平移和方向）。

注意

如果设置的值大于最大方向变化，则使用允许的最大值。

BlendOrientation 的优先级高于 BlendPercentage，如果设置了 BlendOrientation，BlendPercentage 将被覆盖。 如果同时设置 BlendDistance 和 BlendOrientation，机器人将根据先触发的条件开始圆滑运动。

仅适用于 BlendMethod = 1（SP 混合）。 在非 SP 混合或 PTP 运动的情况下，其值将被忽略（不会报错）。

缩写形式

BlendOri

语法

<motion> <element> <position> BlendOrientation = <value>

参数

<element>: 有效运动元素

< value >: Double, 机器人将要参与圆滑运动的转向角度, 0 到 Maxdouble, 单位是度

限制

仅非独立属性

示例

轴

Move A1 10 BlendOrientation =50

机器人

Move SCARA {10,10,0,0} BlendOri =50

另请参见

BLENDMETHOD, BLENDDISTANCE, BLENDPERCENTAGE

BLENDPERCENTAGE

说明

设置将与下一个运动圆滑的动作长度百分比。

0% - 没有圆滑

100% - 从当前运动的中点开始圆滑。

适用于 BLENDMETHOD - 1(SP)。

缩写形式

BLEND

语法

<element>. BlendPercentage= <value>

域

ELEMENT

参数

<element>: 有效的运动元素

< value >: Double，运动长度值的百分比，0.1 到 100

限制

仅非独立属性

示例

MOVE SCARA P1 Blend = 100

另请参见

BLENDMETHOD, MOVE, CIRCLE, MOVES, JUMP, JUMP3, JUMP3CP, BLENDENDPROTECTED, BLENDSTARTPROTECTED, BLENDDISTANCE, BLENDORIENTATION

BLENDSTARTPROTECTED

说明

设置当前运动受保护部分的路径长度，该部分不参与圆滑运动（从起点计算）。当前动作的受保护部分将与上一个或下一个动作完全分开执行。

BlendStartProtected 优先级高于 BlendDistance 和 blendOrientention，如果 BlendStartProtected 被设置(不是 0)，当前运动的起始部分将不会与上一个运动进行圆滑，当前运动的结束部分将与下一个运动进行圆滑。

注意

如果该值大于路径长度，则使用路径长度代替，0 表示不保护任何内容。

仅适用于 BlendMethod = 1 (SP 混合)。在非 SP 圆滑或 PTP 运动的情况下，它的值将被忽略(不会报错)。

缩写形式

BlendSt

语法

<element>. Blendstartprotected = <value>

参数

<element>: 有效的运动元素

< value >: Double, 当前运动受保护部分的路径长度（从起点计算），从 0 到 Maxdouble，单位为毫米

限制

非独立属性

示例

轴

Move A1 10 BlendStartProtected=1000

机器人

Move SCARA {10,10,0,0} BlendSt = 0

另请参见

BLENDMETHOD, BLENDENDPROTECTED, BLENDPERCENTAGE

BNOT

说明

BNOT 是位运算符。BNOT（位取反）将一个数的所有位取反（0 变为 1，1 变为 0）。

语法

BNOT (Number)

参数

<Number>: Long，MaxLong 到 MaxLong

返回值

<return value>: Long

示例

?bin$(BNOT(0b101))

打印以下内容：0b11111111111111111111111111111010

另请参见

BAND, BOR, BXOR

BOR

说明

BOR 是位运算符。BOR（按位或）两个二进制数的对应位进行比较，如果两个二进制数的对应位中至少有一个为 1，则结果的对应位为 1，否则为 0

语法

<Number> BOR < Number >

参数

<Number>: Long，MaxLong 到 MaxLong

返回值

<return value>: Long

示例

?BIN$(0b1010 bor 0b0010)

打印以下内容：0b1010

另请参见

BAND, BNOT, BXOR

BXOR

说明

BXOR 是位运算符。BXOR（异或运算）将两个二进制数的对应位进行比较，如果两个二进制数的对应位不相同，则结果的对应位为 1，否则为 0

语法

<Number> BXOR < Number >

参数

<Number>: Long，MaxLong 到 MaxLong

返回值

<return value>: Long

示例

? BIN$(0b1010 BXOR 0b0101)

打印以下内容：0b1111

另请参见

BAND, BOR, BNOT

CALL

说明

CALL 命令用于调用子程序。

语法

Call <subprogram name>

参数

< subprogram name >: sub

示例

```
Program
…
Call Move_X '[Do 10 incremental moves]
…
End Program
```

另请参见

SUB ... END_SUB, PROGRAM ... END_PROGRAM

CASTJOINT

说明

使用第二个长整型 (Long) 参数给定的机器人类型创建并返回通用关节空间坐标类型的点，坐标值采用第一个参数的值。

如果第一个参数是标量，则结果点将由相同的坐标组成。使用数组作为第一个参数将生成由第一个“坐标数”（由机器人类型决定）数组元素组成的点。

语法

<joint_variable> = castjoint (<long_scalar_expression>, <long_expression>)

<joint_variable> = castjoint (<double_scalar_expression>, <long_expression>)

<joint_variable> = castjoint (<long_array>, <long_expression>)

<joint_variable> = castjoint (<double_array>, <long_expression>)

参数

<long_expression>: 指定用户要转换的类型

包含：TYPE_XY、TYPE_XYZ、TYPE_XYR、TYPE_XYZR、TYPE_XYZABC、TYPE_XYZYPR…

请参见 Point_Type_List

<long_scalar_expression >: Long

<double_scalar_expression>: Double

<long_array>: Long 数组

<double_array>: Dobule 数组

返回值

返回通用关节空间坐标类型的点

限制

只有一维数组可用作参数。

数组元素数必须等于或大于坐标数。

示例

GenJoint = CASTJOINT(1, TYPE_XYZ)

? CASTJOINT(ArrayOf4Doubles, TYPE_XYZR)

另请参见

CASTPOINT, CASTLOCATION

CASTLOCATION

说明

使用第二个长整型 (Long) 参数给定的机器人类型创建并返回通用笛卡尔空间坐标类型的点，坐标值采用第一个参数的值。

如果第一个参数是标量，则结果点将由相同的坐标组成。使用数组作为第一个参数将生成由第一个“坐标数”（由机器人类型决定）数组元素组成的点。

语法

<location_variable> = castlocation (<long_scalar_expression>, <long_expression>)

<location_variable> = castlocation (<double_scalar_expression>, <long_expression>)

<location_variable> = castlocation (<long_array>, <long_expression>)

<location_variable> = castlocation (<double_array>, <long_expression>)

参数

<long_expression>: 指定用户要转换的类型

包含：TYPE_XY、TYPE_XYZ、TYPE_XYR、TYPE_XYZR、TYPE_XYZABC、TYPE_XYZYPR…

请参见 Point_Type_List

<long_scalar_expression >: Long

<double_scalar_expression>: Double

<long_array>: Long 数组

<double_array>: Dobule 数组

返回值

返回通用笛卡尔空间坐标类型的点

限制

只有一维数组可用作参数。

数组元素数必须等于或大于坐标数。

示例

GenLocation = CASTLOCATION(0.5, TYPE_XYZ)

? CASTLOCATION(ArrayOf6Longs, TYPE_XYZR)

另请参见

CASTPOINT, CASTJOINT

CASTPOINT

说明

使用第二个长整型 (Long) 参数给定的机器人类型创建并返回通用点，点类型（Joint 或 Location）和坐标值采用第一个坐标列表参数的值。

语法

<point_variable> = castpoint (<list_of_coordinates>, <long_expression>)

参数

<list_of_coordinates >: Joint 或 Location

<long_expression>: 指定用户要转换的类型

包含：TYPE_XY、TYPE_XYZ、TYPE_XYR、TYPE_XYZR、TYPE_XYZABC、TYPE_XYZYPR…

请参见 Point_Type_List

返回值

返回一个通用点

限制

第二个参数中给定的机器人类型必须匹配坐标列表参数的坐标数。

示例

GenJoint = CASTPOINT({0.0, 10.0, 20.0}, TYPE_XYZ)

? CASTPOINT(#{0.0, 0.0, 0.0, 1.0}, TYPE_XYZR)

另请参见

CASTJOINT, CASTLOCATION

CHR$

说明

此函数返回与给定 ASCII 值对应的单字符字符串。

语法

CHR$(<number>)

参数

<number>: Long，0 到 255

返回值

返回与给定 ASCII 值对应的单字符字符串

<return value>: String

限制

只读

示例

PRINT CHR$(65)

打印以下内容：A

另请参见

SPACE$, STRING$, UTF$, ASC, STRL$

CIRCLE

说明

CIRCLE 使机器人/轴组在笛卡尔坐标下做圆弧运动。对于机器人模型，CIRCLE 命令在笛卡尔空间 (XYZ) 中发出一条圆周路径。方向角度与圆周角度成比例插补。

CIRCLE 命令有两种格式。一种指定轴组名称、角度和圆心。此格式可实现多圈圆周运动，另一种通过圆周点和圆弧的终点进行定义。

{Until <Signal> = <StopCondition>}

语法

Circle {对象} Angle CircleCenter {CirclePlane =<value> } { VTRAN =<value>} { ATRAN =<value>}{ JTRAN =<value>}{ VROT =<value>}{ AROT =<value>}{ JROT =<value>} {Blend=<value>} {BlendDistance=<value>}{BlendOrientation=<value>} {Blendendprotected=<value>} {Blendstartprotected=<value>} {AScale=<value>} {WithPls=<value>} {Until <signalName>=<value>} {Abs=<value>}

Circle {对象} CirclePoint TargetPoint { VTRAN =<value>} { ATRAN =<value>}{ JTRAN =<value>}{ VROT =<value>}{ AROT =<value>}{ JROT =<value>} {Blend=<value>} {BlendDistance=<value>}{BlendOrientation=<value>} {Blendendprotected=<value>} {Blendstartprotected=<value>} {AScale=<value>} {WithPls=<value>} {Until <signalName>=<value>} {Abs=<value>}

参数

{对象}：是机器人，此处可以不添加或是写 SCARA。

Angle：指定圆的度数，Double

CircleCenter：指定圆心，Joint 或 Location

CirclePlane：定义将在哪个平面进行圆插补，Double，0,1,2，默认值 = 0

0 (XY)

1 (XZ)

2 (YZ)

针对弧形运动

CirclePoint：定义圆弧曲线辅助点，曲线会经过辅助点，Joint 或 Location

TargetPoint：定义圆弧曲线终点，Joint 或 Location

{Blend}：轨迹圆滑百分比，范围为 0 – 100%。

每条 Moves 语句中所设定的 Blend 参数意味着，本条语句这段运动轨迹在进入和移出时，切入距离或离开距离与一半本条语句运动轨迹的比值。

例如，当该参数为 100 时，整条轨迹的中心点就成了切入点及移出点，即该轨迹本身就只剩下了一个点，其余部分都用于轨迹圆滑了。当该参数为 25 时，切入点就在轨迹开始后的 1/8 处，移出点在轨迹结束前的 1/8 处。未设置情况下默认值为 0。

{BlendDistance}：轨迹圆滑距离。

每条 Moves 语句中可以采用 BlendDistance 圆滑长度来设定 Blend 参数。

例如，当该参数为 BlendDistance=100 时，表示当前指和下条指令的平滑长度为 100mm，平滑长度不会超过两条指令中移动弧长较短的一半长度。圆滑的切入点为距离当前运动轨迹目标点弧长距离 100mm 处，切出点为距离下条运动轨迹起点 100mm 处。未设置情况下默认值为 0。

{BlendOrientation}：轨迹姿态圆滑长度。

每条 Moves 语句中可以采用 BlendOrientation 姿态圆滑长度来设定 Blend 参数。

例如，假设当前运动指令的姿态旋转角度大小是 90°，当该参数为 BlendOrientation=20 时，表示当运动到姿态旋转角度到 70°时，剩余的 20°姿态旋转角会和下一段运动指令旋转角度进行融合圆滑。设置的最大值不会超过总旋转角度的一半，即 45°。

{Blendstartprotected}：轨迹起始段圆滑保护。

每条 Moves 语句中可以采用 Blendstartprotected 来指定轨迹起始点开始到设定长度不用于圆滑，优先级高于 BlendDistance 和 blendOrientention。如果 Blendstartprotected 设置值大于 0，那么这条轨迹是不会和上一条轨迹进行圆滑。

{Blendendprotected}：轨迹结束段圆滑保护。

每条 Moves 语句中可以采用 Blendendprotected 从设定长度到轨迹目标点这段长度不用于圆滑，优先级高于 BlendDistance 和 BlendOrientention。

Blendendprotected 设置值大于 0，那么这条轨迹是不会和下一条轨迹进行圆滑。

{Armcmd}：机器人的手型。

该标记确定了使用哪种方式通过关节坐标系到达目标位置。手型值 0 为自动，1 为左手型，2 为右手型。

{WithPls}：位置触发器。

指的是在机器人执行该条运动指令的过程中，机器人到达指定位置以后将触发 IO 信号的功能。每个位置触发器都需要在项目设置中进行创建及配置。在写程序时，该参数直接挂载配置好的位置触发器名字即可。

{Until}：一种用于中断执行动作的功能属性。

当配置的信号值等于预期值时，执行动作将停止，下一动作继续。Until 命令应添加在运动命令的末尾，它也是运动命令的可选属性。基本格式为“Until <Signal> = <StopCondition>”。其中<Signal>为数字输入信号，需要事先在项目的“输入输出”页面中创建好。而<StopCondition>为 on/off，是为中断的条件。

{Abs}：全称为 Absolute。该属性用于定义输入的目标点是绝对位置还是相对位置。

在绝对位置模式下，元素的位置命令是所需的绝对位置。 在相对位置模式下，位置指令是所需的位置变化。 值为 0 代表相对位置，值为 1 代表绝对位置。在未设置情况下默认值为 1。

{VTRAN}：全称为 VELOCITYTRANS 平移运动曲线的速度，单位为 mm/sec。

执行运动指令时，该属性值应小于或等于系统最大平移速度。 如果大于系统最大平移速度，则使用系统最大平移速度值执行运动。

{ATRAN}：ACCELERATIONTRANS 平移运动曲线的加速度，单位为 mm/sec²。

执行运动指令时，该属性值应小于或等于系统最大平移加速度。 如果大于系统最大平移加速度，则使用系统最大平移加速度值执行运动。

{DTRAN}：全称为 DECELERATIONTRANS 平移运动曲线的减速度，单位为 mm/sec²。

执行运动指令时，减速度应小于或等于系统最大平移减速度。 如果减速度大于系统最大平移减速度，则以系统最大平移减速度值执行运动。

{JTRAN}：JERKTRANS 机器人的平移运动加加速度。加加速度时是加速度的变化率。

{VROT}：全称为 VELOCITYROT 旋转运动曲线的速度，单位为 degree/sec。

执行运动指令时，该属性值应小于或等于系统最大旋转速度。 如果大于系统最大旋转速度，则使用系统最大旋转速度值执行运动。

{AROT}：全称为 ACCELERATIONROT 旋转运动曲线的加速度，单位为 degree/sec²。

执行运动指令时，该属性值应小于或等于系统最大旋转加速度。 如果大于系统最大旋转加速度，则使用系统最大旋转加速度值执行运动。

{DROT}：全称为 DECELERATIONROT 旋转运动曲线的旋转减速度，单位为 degree/sec²。

执行运动指令时，减速度应小于或等于系统最大旋转减速度。 如果减速度大于系统最大旋转减速度，则以系统最大旋转减速度值执行运动。

{JROT}：JERKROT 机器人的旋转运动加加速度。加加速度时是加速度的变化率。

限制

如果运动元素被其他任务连接，可能无法移动该元素。

运动过程中经过过奇异点会报错停止。

只有对于版本高于 2.5 的 Control Studio，所有运动命令才具备 until 功能 (SMOVE 除外）。

示例

Circle SCARA Angle = 90 CircleCenter = {10,10,0,0} VTran = 500 ‘手动输入轴坐标点位为圆心，圆心角为 90°

Circle SCARA CirclePoint = {600,0,-100,0} TargetPoint = {600,200,-100,0} VTran = 500 Until signal1 = Off ‘手动输入轴坐标辅助点及目标点

注意，输入点位的格式并不会影响其他参数的设置，甚至不同的参数彼此之间都是独立互不影响的。另外，如果期望在该指令之后卡住程序指针以中断指令预读，则需要在后面添加 WaitForMotion 指令。

Circle SCARA CirclePoint = {10,20,0,0} TargetPoint = {100,200,0,0} WithPls=PT_1 ‘设置添加位置触发器

Circle SCARA CirclePoint = {10,20,0,0} TargetPoint = {100,200,0,0} Until SIGNAL1=on ‘添加 Unitl 中断功能，当信号 SIGNAL1 为 on 的时候触发

注意，圆弧运动目标点手型由起点决定，不需另外指定，如需指定建议先以指定手型运动到圆弧运动起点再开始圆弧运动。

Move PL1 Armcmd=1 ‘指定当前 Move 指令的手型

Circle SCARA Angle = 90 CircleCenter = #{700,0,0,0} VTran=200

Circle SCARA CirclePoint = {600,0,-100,0} TargetPoint = {600,200,-100,0} VTran = 500 ‘设定平移运动速度

Circle SCARA CirclePoint = {600,0,-100,0} TargetPoint = {600,200,-100,0} ATran = 500 ‘设定平移运动加速度

Circle SCARA CirclePoint = {600,0,-100,0} TargetPoint = {600,200,-100,0} JTran = 500 ‘设定平移运动加加速度

Circle SCARA CirclePoint = {600,0,-100,0} TargetPoint = {600,200,-100,0} VRot = 500 ‘设定旋转运动速度

Circle SCARA CirclePoint = {600,0,-100,0} TargetPoint = {600,200,-100,0} ARot= 500 ‘设定旋转运动加速度

Circle SCARA CirclePoint = {600,0,-100,0} TargetPoint = {600,200,-100,0} JRot = 500 ‘设定旋转运动加加速度

示例程序如下，其中点位需要自行创建：

```
Program
    Attach
    Move PL1 Armcmd=1
Circle SCARA Angle = 90 CircleCenter = #{700,0,0,0} VTran=200 ATran=1000 JTran=5000 Blend=50
Moves PL2 Blend=100 VTran=1000
Move PL1 VScale=50
Circle SCARA CirclePoint = {600,0,-100,0} TargetPoint = {600,200,-100,0} VTran = 500 Until signal1 = Off
Move PL1 VScale=50
Circle SCARA CirclePoint = {600,0,-100,0} TargetPoint = {600,300,-100,100} VRot = 500 ARot=1000 WithPls=PT_1
   Move PL3
Circle SCARA Angle = 90 CircleCenter = #{700,0,0,0} CirclePlane=1 VTran=200 ATran=1000 JTran=5000 Blend=50
    Detach
end program
```

Move SCARA #{100,0,0,0}

Circle Angle = 180 CircleCenter = #{50, 0, 0, 0}Vtran = 500 CirclePlane = PLANE_XY

另请参见

BLENDPERCENTAGE, MOVE, MOVES, ACCELERATIONMAXROT, ACCELERATIONMAXTRANS, ACCELERATIONROT, ACCELERATIONTRANS, BLENDMETHOD, DECELERATIONROT, DECELERATIONTRANS, JERKMAXROT, JERKMAXTRANS, JERKROT,

JERKTRANS, POSITIONFINAL, VELOCITYFINALROT, VELOCITYFINALTRANS, VELOCITYMAXROT, VELOCITYMAXTRANS, VELOCITYROT, VELOCITYROTVALUE, VELOCITYTRANS, VELOCITYTRANSVALUE, VSMODE

CLEARMSG

说明

清除所有消息日志。

语法

ClearMsg()

示例

Program

?33

ClearMsg()

?55

End Program

以下内容被打印出来：

55

另请参见

PRINT

CLOCK

说明

返回系统时钟周期数。这是操作系统运行的时钟。一个时钟周期对应 1 毫秒。

语法

?Sys.Clock

?System.Clock

域

SYSTEM

返回值

返回系统时钟周期数

<return value>: Long，0 到 MaxLong

限制

只读

示例

?System.Clock

CLOSE

说明

释放文件句柄并关闭文件。

< FileHandle >: 文件句柄

语法

Close #<FileHandle>

参数

< FileHandle >: Long

限制

只写

示例

Close #filehandle

文件句柄通过打开文件命令绑定。

另请参见

OPEN_FILE, INPUT$, LOC, ACCEPT, CONNECT, IPADDRESSMASK, OPENSOCKET, PING, PRINT_HASH, PRINTPOINTUSING, PRINTTOBUFF, PRINTUSING, PRINTUSING$, PRINTUSING_HASH-SIGN, PRINTUSINGTOBUFF

COMMON_OR_DIM_SHARED_OR_DIM_..._AS_LONG

说明

这些命令用于将用户变量定义为 LONG 类型。长整型 (Long) 数据是一个整数，也可以通过赋值 True 或 False 用作布尔 (Boolean) 型。

系统变量通过配置文件（在程序块之前）、用户任务（在程序块之前）和终端中的通用共享命令进行定义。这些变量具有系统范围的可见性。所有任务均可使用这些变量，且这些变量可用于协调任务之间的活动。

任务变量在用户任务中通过 Dim Shared 进行定义（在程序块之前）。这些变量具有任务范围的可见性。此任务中的所有函数和子程序块均可使用这些变量。

局部变量在用户任务的程序、子程序和函数中通过 Dim 命令进行定义。这些变量在不同的子程序、函数和程序块中可能具有相同的名称。其他块无法使用局部块中的变量。

变量必须为三种类型之一。

语法

Common Shared | Dim Shared | Dim <name> As {Const} Long = <const_value>

参数

{ Const }：将变量设置为常量变量的可选关键字

<name>: 用户指定的变量名称

<const_value>:初始值，Long

限制

不支持数组和子程序或函数定义（参数和返回值）。

用户常量必须在声明时赋值。用户常量仅在声明语句中可写。之后，这些常量变为只读。用户常量无法按“引用”传递给子程序和函数。它们必须始终按“值”进行传递。

示例

```
Common Shared CLConst as Const Long = 100

Dim Shared DL as Long = CLConst

Program

Dim Integer1 as Long = DL

Dim xFlag as Long = True

End Program
```

COMMON_OR_DIM_SHARED_OR_DIM_..._AS_DOUBLE

说明

这些命令用于将用户变量定义为 DOUBLE 类型。双精度 (Double) 数据是双精度浮点型数字。

系统变量通过配置文件（在程序块之前）、用户任务（在程序块之前）和终端中的通用共享命令进行定义。

任务变量在用户任务中通过 Dim Shared 进行定义（在程序块之前）。

局部变量在用户任务的程序、子程序和函数中通过 Dim 命令进行定义。

语法

Common Shared | Dim Shared | Dim <name> As {Const} Double = <const_value>

参数

{ Const }：将变量设置为常量变量的可选关键字

<name>: 用户指定的变量名称

<const_value>:初始值，Double

限制

不支持数组和子程序或函数定义（参数和返回值）。

用户常量必须在声明时赋值。用户常量仅在声明语句中可写。之后，这些常量变为只读。用户常量无法按“引用”传递给子程序和函数。它们必须始终按“值”进行传递。

示例

Common Shared CDConst as Const Double = 1.2345

Dim Shared DD as Double = CDConst

Program

Dim myvar as Double = DDEnd Program

COMMON_OR_DIM_SHARED_OR_DIM_..._AS_STRING

说明

这些命令用于将用户变量定义为 STRING 类型。

系统变量通过配置文件（在程序块之前）、用户任务（在程序块之前）和终端中的通用共享命令进行定义。这些变量具有系统范围的可见性。所有任务均可使用这些变量，且这些变量可用于协调任务之间的活动。

任务变量在用户任务中通过 Dim Shared 进行定义（在程序块之前）。这些变量具有任务范围的可见性。此任务中的所有函数和子程序块均可使用这些变量。

局部变量在用户任务的程序、子程序和函数中通过 Dim 命令进行定义。这些变量在不同的子程序、函数和程序块中可能具有相同的名称。其他块无法使用局部块中的变量。

变量必须为三种类型之一。

语法

Common Shared | Dim Shared | Dim <name>As {Const} String { of UTF8 } = <const_value>

参数

{ Const }：将变量设置为常量变量的可选关键字

{ of UTF8 }：将字符串设置为 UTF-8 编码的可选关键字

<name>: 用户指定的变量名称

<const_value>:初始值，String

限制

不支持数组和子程序或函数定义（参数和返回值）。

用户常量必须在声明时赋值。用户常量仅在声明语句中可写。之后，这些常量变为只读。用户常量无法按“引用”传递给子程序和函数。它们必须始终按“值”进行传递。

示例

```
Common Shared CDConst as Const String = “aaaaaa”

Dim Shared DD as String = CDConst

Program

Dim myvar as String of UTF-8

myvar = UTF$(1000) + UTF$(500)

End Program
```

另请参见

UTF$, CHR$

COMMON_OR_DIM_SHARED_OR_DIM_..._AS_JOINT_OF

说明

这些命令用于将用户变量定义为 JOINT 类型。关节空间坐标 (Joint) 数据是双精度数据矢量，用于表示关节空间坐标系中的位置。

系统变量通过配置文件（在程序块之前）、用户任务（在程序块之前）和终端中的通用共享命令进行定义。这些变量具有系统范围的可见性。所有任务均可使用这些变量，且这些变量可用于协调任务之间的活动。

任务变量在用户任务中通过 Dim Shared 进行定义（在程序块之前）。这些变量具有任务范围的可见性。此任务中的所有函数和子程序块均可使用这些变量。

局部变量在用户任务的程序、子程序和函数中通过 Dim 命令进行定义。这些变量在不同的子程序、函数和程序块中可能具有相同的名称。其他块无法使用局部块中的变量。

变量必须为三种类型之一。

以关节空间坐标形式给定的位置是 n 个关节空间坐标值的数组：

{jnt1, jnt2, jnt3, jnt4, jnt5, jnt6}

语法

Common Shared | Dim Shared | Dim <name>As {Const} Joint of < robot_type > = <const_value>

参数

{ Const }：将变量设置为常量变量的可选关键字

<name>: 用户指定的名称

<const_value>:初始值，Joint

<robot_type>: 机器人类型

根据 Point_Type_List

限制

不支持数组和子程序或函数定义（参数和返回值）。

用户常量必须在声明时赋值。用户常量仅在声明语句中可写。之后，这些常量变为只读。用户常量无法按“引用”传递给子程序和函数。它们必须始终按“值”进行传递。

示例

```
Common Shared CJConst as Const Joint of XYZR = {0,0,0,0}
Dim Shared DJ as Joint of XYZR
Program
Dim myjoint as Joint of XYZR =  CJConst
End Program
```

COMMON_OR_DIM_SHARED_OR_DIM_..._AS_LOCATION_OF

说明

这些命令用于将用户变量定义为 LOCATION 类型。笛卡尔空间坐标 (Location) 数据是双精度数据矢量，用于表示笛卡尔坐标系中的位置。

系统变量通过配置文件（在程序块之前）、用户任务（在程序块之前）和终端中的通用共享命令进行定义。这些变量具有系统范围的可见性。所有任务均可使用这些变量，且这些变量可用于协调任务之间的活动。

任务变量在用户任务中通过 Dim Shared 进行定义（在程序块之前）。这些变量具有任务范围的可见性。此任务中的所有函数和子程序块均可使用这些变量。

局部变量在用户任务的程序、子程序和函数中通过 Dim 命令进行定义。这些变量在不同的子程序、函数和程序块中可能具有相同的名称。其他块无法使用局部块中的变量。

变量必须为三种类型之一。

以笛卡尔坐标形式给定的位置是一个前面带“#”号的数组，用于区分关节空间坐标和笛卡尔空间坐标位置：

#{x_value, y_value, z_value, yaw_value, pitch_value, roll_value}

语法

Common Shared | Dim Shared | Dim <name>As {Const} Location of < robot_type > = <const_value>

参数

{ Const }：将变量设置为常量变量的可选关键字

<name>: 用户指定的名称

<const_value>:初始值，Location

<robot_type>: 机器人类型

根据 Point_Type_List

限制

不支持数组和子程序或函数定义（参数和返回值）。

用户常量必须在声明时赋值。用户常量仅在声明语句中可写。之后，这些常量变为只读。用户常量无法按“引用”传递给子程序和函数。它们必须始终按“值”进行传递。

示例

Common Shared CLConst as Const Location of XYZR = #{0,0,0,0}

Dim Shared DL as Location of XYZR

Program

Dim mylocation as Location of XYZR = CLConst

End Program

另请参见

DELETEVAR

COMMON_OR_DIM_SHARED_..._AS_NOTE_OR_ERROR_ASCII_NUMBER

说明

创建用户异常并定义相应的错误消息。

异常可在系统或任务级别进行声明。

解释器和错误处理器会处理应用程序异常，就像处理任何其他内部错误或提示一样。

应用程序异常可通过 TRY/CATCH、ONERROR 或 ONSYSTEMERROR 捕获，根据异常的严重程度，系统会以停止任务或运动的方式作为响应。

您可以指定从 22000 到 22498 的异常编号。

如果未指定异常编号，则系统会分配 22499 到 22999 范围内的任意数值。该值每次加载时可能不同。

注意

系统已定义许多错误，用户错误不应与系统错误具有相同的编号。

语法

common shared <name> as error "<message>" [number]

dim shared <name> as error "<message>" [number]

common shared <name> as note "<message>" [number]

dim shared <name> as note "<message>" [number]

参数

<name>: 用户指定的名称

<message>: 调用异常时要打印的异常消息，String

[number]：错误或提示的编号

限制

异常的严重程度级别可能仅包含 NOTE 和 ERROR 类型。

异常编号必须唯一。

不支持数组和子程序定义。

示例

Common Shared MyErr as Error "'App Error" 22001

common shared err as error "'My Error"

另请参见

LOGGER, THROW, NUM, MSG

COMMON_SHARED_..._AS_SEMAPHORE

说明

在系统中定义新的信号量变量。只有两个函数可访问此变量类型：SEMTAKE 和 SEMGIVE。此数据类型不允许进行任何其他操作（例如，打印、参数）。

语法

COMMON SHARED < variable > AS SEMAPHORE

参数

< variable >: 用户指定的变量名称

示例

common shared sem as semaphore

?semtake(sem)

semgive(sem)

另请参见

SEMAPHOREGIVE, SEMAPHORETAKE, DELETESEM

COMMON_SHARED_OR_DIM_SHARED_OR_DIM_..._AS_TYPE

说明

由于结构体是一种新数据类型，因此必须首先在配置文件中定义。只有这样，新数据类型“结构体”的名称才能用于声明结构体变量。

结构体变量可以在所有范围内定义，即在配置文件和终端中定义（全局变量），或者在任务中定义，在程序块之前定义（静态变量）或之内定义（局部变量）。结构体也可以定义为子程序和函数的参数，以及函数的返回值。结构体变量可以是标量或数组。一个数组最多可以有 10 个维度。

语法

为单个结构体元素赋值：

<structure_name>-><structure element name> =<expression>

查询单个结构体元素的语法：

? <structure_name>-><structure element name>

定义结构体：

COMMON SHARED|DIM SHARED|DIM <structure_name> AS <structure type>

定义结构体数组：

COMMON SHARED|DIM SHARED|DIM <structure_name>[ ]… AS <structure type>

参数

<Structure type>: 声明的结构体类型

<structure_name>: 定义的类型变量

<structure element name>: 结构体元素变量

限制

只写。结构体类型必须在配置文件中预定义。

示例

在配置文件中 ->

```
Type X
L1 as Long
Length as Long
End Type
```

在应用程序文件中 ->

```
Dim shared S1 as X
Program
S1->Length =2
?S1->Length
End program
```

另请参见

STRUCTURE_TYPE_DEFINITION

CONNECT

说明

从远程主机请求 TCP 连接。

语法

Connect(#<device number>, <ip address>,<port number>{,<timeout>})

参数

<device number>: Long，1 到 255

<port number>: TCP 端口，Long

<ip address>: String，IP 地址

<timeout>: Long

限制

仅任务

示例

```
OpenSocket Options=1 as #200
Connect(#200,"127.0.0.1",6002)
print Input$(LOC(200), #200)
close #200
```

另请参见

CLOSE, OPENSOCKET, ACCEPT, PING, IPADDRESSMASK

CONTINUEPATH

说明

这条命令可以用来恢复 STOPPATH 所存储的运动指令。存储的运动指令在当前运动指令完成后开始执行。

CONTINUEPATH 将直接从 STOPPATH 存储的下一条运动指令开始。

提示

CONTINUEPATH 只能在执行了 STOPPATH 之后才能使用。

如果在事件中使用了 CONTINUEPATH，请注意该命令的执行时间。恢复运动指令的路径将被重新计算。由于起始位置不同或过渡停止的原因，新的路径可以能与之前的路径完全不同。

缩写形式

Continuepath

语法

Continuepath<robot | axis>

参数

<robot | axis>: 任何有效的机器人或轴

限制

仅写

示例

Continuepath SCARA

另请参见

RECOVERPATH

COLLISIONDETECT

说明

这条命令用来设置开启/关闭碰撞检测功能。

提示

CollisionDetect 可以设置单轴设置和轴组一起设置，查询只能单轴查询，不能轴组查询。

语法

<robot | axis>.CollisionDetect = on/off

? <axis>.CollisionDetect

参数

<robot | axis>: 任何有效的机器人或轴

限制

读/写

示例

SCARA. CollisionDetect = on

J1. CollisionDetect = on

?J1. CollisionDetect

另请参见

RESETPEAKVALUE, SETVELERRTHRESHOLD, DRIVETORQUELIMIT, PEAKTORQUE, PEAKVELERR, VELERRTHRESHOLD, SETDRIVETORQUELIMIT, SETDEFVELERRTHRESHOLD

COS

说明

此函数返回表达式（弧度）的余弦值。

语法

Cos(<expression>)

参数

<expression>: 任何有效的表达式，Double

± 9.223372036853900e+18

返回值

返回表达式（弧度）的余弦值

限制

只写

示例

?Cos(3.14159/2)

VaSCARA = Cos(3.14159)

另请参见

ACOS, SIN, TAN

CUMULATIVEPOSITION

说明

此属性返回轴位置值的累积变化（绝对值）。

语法

<axis>.CumulativePosition = <value>

?<axis>.CumulativePosition

域

AXIS

参数

<axis>: 任何有效的轴

<value>: Double，0 到 MAXDouble

返回值

返回轴位置值的累积变化（绝对值）

<return value>: Double，0 到 MAXDouble

示例

A1.CumulativePosition = 0

?A1.CumulativePosition

CURRENTABSOLUTE

说明

返回当前执行运动的绝对属性

缩写形式

<element>.CAbs

语法

? <element>.CurrentAbsolute

域

ELEMENT

参数

<element>: 有效的运动元素

返回值

返回当前执行运动的绝对属性

<return value>: Long

限制

只读

示例

轴

?A1.CAb

机器人

?SCARA.CAbs

另请参见

ABSOLUTE

CURRENTTIME

说明

返回当前执行运动的总时间。值：

-1 = 无活动运动

-2 = 不适用（PrfType = 0 或 1）

> 0 = 运行运动的当前时间

注意

请注意，如果是圆滑的情况，将返回圆滑运动的第一段运动的 TotalTime/CurrentTime！圆滑停止（仅执行第二段运动，即第一段运动已完成）后，将返回第二段运动的时间值。

缩写形式

element.Ctime

语法

?<element>Ctime

域

ELEMENT

参数

<element>: 有效的运动元素

返回值

返回当前执行运动的总时间

<return value>: Long，-2⋯MaxInt

限制

只读

示例

轴

?A1.currenttime

机器人

?SCARA.currenttime

?SCARA.currenttime

DECELERATION

说明

此属性设置运动轨迹的减速度比率。

执行运动命令时，减速度应小于或等于 DECELERATIONMAX。若大于 DECELERATIONMAX，则以 DECELERATIONMAX 值执行运动。

此属性可在运动命令内使用以覆盖永久值。

缩写形式

<element>.Dec

语法

<element>.Deceleration = <expression>

?<element>.Deceleration

域

ELEMENT

参数

<element>: 有效的运动元素

< value >: Double，运动轨迹值的减速度比率，大于 0

限制

加加速度和减速度之间的比率受以下关系限制：

$\mathrm{Jerk/Dec} < 0.9 \times \pi/5 \times \mathrm{T}$

其中 T 是采样时间（秒）。

对于 2 毫秒的周期时间，该值为 282.74。使用 SMOOTHFACTOR 可自动计算加加速度。

要在任务中设置该值，必须将运动元素连接到该任务（使用 ATTACH 命令）。

示例

轴

A1.deceleration = 1e10

Move A1 100 dec = 2e10

机器人

SCARA.deceleration = 1e10

Move SCARA {100, 20,0,0} dec = 2e10

另请参见

DECELERATIONMAX, ACCELERATION, ACCELERATIONSCALE, JUMP, MOVE

DECELERATIONMAX

说明

定义允许的最大运动元素减速度。如果指定的减速度比率大于 DECELERATIONMAX，系统会将该值设置为 DECELERATIONMAX 并通知您。

缩写形式

<element>.DMax

语法

<element>.DecelerationMax = <expression>

?<element>.DecelerationMax

域

ELEMENT

参数

<element>: 有效的运动元素

< value >: Double，允许的最大运动元素减速度值，大于 0

限制

要在任务中设置该值，必须将轴连接到该任务（使用 ATTACH 命令）。

示例

轴

A1.Dmax = 20e3

机器人

SCARA.Dmax = 20e3

另请参见

ACCELERATIONMAX, DECELERATION

DECELERATIONROT

说明

定义机器人的旋转减速度。与 DTRAN 一起，定义笛卡尔运动的减速度。

该值仅在两个运动命令中使用：MOVES 和 CIRCLE。在轴插补运动 (MOVE) 中，该值将被忽略。轴组必须使用机器人模型 (model !=1) 进行定义。

该值不得大于 ACCLERATIONMAXROT。系统始终采用两者中较小的值，并向用户发送通知消息。

缩写形式

<ROBOT>.drot

语法

<ROBOT>.drot=<numeric expression>

域

ROBOT

参数

< ROBOT >: 任何有效的机器人

<numeric expression>: Double，0.1 到 Maxdouble

限制

读/写，独立/非独立属性

示例

drot = 6000

另请参见

CIRCLE, MOVES, DECELERATIONTRANS

DECELERATIONTRANS

说明

定义机器人的平移减速度。与 DROT 一起，定义笛卡尔运动的减速度。

该值仅在两个运动命令中使用：MOVES 和 CIRCLE。在轴插补运动 (MOVE) 中，该值将被忽略。轴组必须使用机器人模型 (model !=1) 进行定义。

该值不得大于 DECLERATIONMAXTRANS。系统始终采用两者中较小的值，并向用户发送通知消息。

缩写形式

<ROBOT>.dtran

语法

<ROBOT>.dtran=<numeric expression>

域

ROBOT

参数

< ROBOT >: 任何有效的机器人

<numeric expression>: Double，0.1 到 Maxdouble

限制

读/写，独立/非独立属性

示例

dtran = 6000

另请参见

CIRCLE, MOVES, DECELERATIONROT

DELETE

说明

此命令用于从闪存盘中删除文件。

语法

Delete <filename>

参数

<filename>: 文件规格，磁盘上现有文件的名称

限制

只写。只能删除未加载到 RAM 中的文件。

不能删除受密码保护的文件。

示例

Delete FILE1.PRG

另请参见

DELETE$

DELETE$

说明

此命令用于从闪存盘中删除文件。

语法

Delete <filename string>

参数

<filename>: String，磁盘上现有文件的名称

限制

只写。只能删除未加载到 RAM 中的文件。

不能删除受密码保护的文件。

示例

Delete$ “File1.PRG”

或

Common shared FName as string = ““File1.PRG”

Delete$ FName

另请参见

DELETE

DELETESEM

说明

删除系统中之前定义的信号量变量

缩写形式

dsem

语法

dsem <semaphore name>

参数

< semaphore name >: 信号量变量

示例

common shared sem as semaphore

deletesem sem

另请参见

COMMON_SHARED_..._AS_SEMAPHORE

DEPPOINT

说明

STOPPATH 指令的终点位置将被存储在 DEPPOINT。这是一个带有手系的不带任何坐标系的笛卡尔点。

DEPPOINT 总是和 RECOVERPATH 指令一起使用。

语法

<point_variable> = <elementName>.deppoint

?<element_name>.deppoint

域

ELEMENT

参数

<elementName>: 有效的运动元素

返回值

返回 deppoint 的笛卡尔坐标点

point_variable>: Location

限制

只读

只能用在使用了 STOPPATH 指令之后

示例

轴

P1=A1.Deppoint

?A1.Deppoint

机器人

P1=Scara.Deppoint

?Scara.Deppoint

DEST

说明

检索运动的当前目标点（用户单位）。如果运动元素已完成其运动，则该命令等于 PCMD。如果运动元素已通过 STOP 命令停止，将返回已停止（已取消）运动的目标。

语法

<point_variable> = <elementName>.dest

? <element_name>.dest

域

ELEMENT

参数

< elementName >: 有效的运动元素

返回值

返回运动的当前目标点（用户单位）

< point_variable >: Location

限制

仅独立属性，只读。应在 Stop 命令之后使用 Proceed 命令来获得正确的值。

示例

轴

P1= A1.Dest

?A1.Dest

机器人

P1= Scara.Dest

?Scara.Dest

另请参见

MOVES

DETACH

说明

此命令用于将任务与运动元素分离，将其释放以供其他任务使用。运动元素可以是轴或轴组。

注意

如果运动元素正在移动，DETACH 将等待，直到运动元素的运动结束。因此，用户任务将处于暂停状态，直到当前运动完成

如果忽略分离命令，当前运动将中断（停止），这表示可能无法到达程序中最后一个运动的最终位置！

语法

Detach <element>

参数

< element>: 定义的运动元素

限制

只写。定义的运动元素必须之前已连接。

示例

Detach SCARA

另请参见

ATTACH, ATTACHTO$, ATTACHTO, DETACHFROM, ATTACHEDTO, DETACHFROM$

DETACHFROM

说明

将给定运动元素与给定任务分离。与 DETACH 功能相同，但是将会指向外部任务。

语法

DetachFrom <element> File = <task>

参数

<task>: 文件规格，包括文件名和扩展名。

<element>: 运动元素规格可以是轴、轴组或机器人。

示例

attachto a16 File = ttt.prg

detachfrom a16 File = ttt.prg

另请参见

ATTACH, DETACH, ATTACHTO$, ATTACHTO, DETACHFROM$, ATTACHEDTO

DETACHFROM$

说明

将给定运动元素与给定任务分离。与 DETACH 功能相同，但是将会指向外部任务。

语法

DetachFrom$ <element name> File = "<task name>"

参数

<task name>: 包含文件规格的字符串，包括文件名和扩展名。

<elemen name >: 运动元素规格可以是轴、轴组或机器人。

示例

attachto$ a16 File = "ttt.prg"

detachfrom$ a16 File = "ttt.prg"

另请参见

ATTACH, DETACH, ATTACHTO$, ATTACHTO, ATTACHEDTO, DETACHFROM

DISABLETIMEOUT

说明

系统从禁用驱动器直到驱动器变为禁用状态所等待的最长时间。

若经过这段时间之后驱动器未被禁用，系统将返回错误 12008“Axis cannot be disabled”（无法禁用轴）。

默认值计算如下：

DISABLETIMEOUT = DISTIME + (VFB-DISSPEED)/DECSTOP

其中：

DISTIME (IDN 207) = 驱动器关闭延迟时间的最大值。

VFB (IDN 40) = 速度反馈转换为 RPM 的最大值。

DISSPEED (IDN P16) = 激活禁用阈值速度的最小值。

DECSTOP (IDN P87) = 快速减速度比率的最小值。

所有值均为驱动器单位。

缩写形式

<axis>.DTimeout

语法

?<axis>.DisableTimeout

域

AXIS

参数

<axis>: 任何有效的轴

返回值

返回系统从禁用驱动器直到驱动器变为禁用状态所等待的最长时间

<return value>: Long，0 到 MaxLong

示例

?A1.DisableTimeout

A1.DisableTimeout = 10000

DISTL

说明

计算两点之间的距离（长度单位）

语法

<variable> = distl (<location_expression>, <location_expression>)

参数

<location_expression >: Joint 或 Location

返回值

返回两点之间的距离（长度单位）

<variable>: Double

限制

两个位置参数必须具有相同的坐标数

示例

? DistL (#{1,2,3}, #{4,5,6})

D = DistL (L1, L2)

另请参见

DISTR

DISTR

说明

计算两点之间的距离（角度单位）

语法

<variable> = distr (<location_expression>, <location_expression>)

参数

<location_expression >: Joint 或 Location

返回值

返回两点之间的距离（角度单位）

<variable>: Double

限制

两个位置参数必须具有相同的坐标数

示例

? DistR (#{1,2,3,1}, #{4,5,6,3})

D = DistR (L1, L2)

另请参见

DISTL

DO_..._LOOP

说明

DO⋯LOOP 用于无限次执行一段代码。这些语句执行至少一次，因为循环终止条件在循环结束时进行判断。

DO…LOOP 内部的语句是可选的。如果循环内未包含任何内容，则 DO⋯LOOP 可以用作延迟。循环可以在条件为真时执行，也可以在条件为真之前执行。通过使用 LOOP UNTIL 或 LOOP WHILE 可进行此选择。

注意

不要使用不包含任何语句或仅包含简单操作语句的循环结构，如果您一定要如此使用，请至少额外插入一条 SLEEP 1 指令，避免出现 CPU 过载，导致系统异常。

语法

Do

<code to execute>

Loop While | Until <condition>

示例

Do … Loop Until

```
Dim Shared I as Long
Program
I=0
Do
I=I+1
Print I
Loop Until I=10
End Program
```

Do … Loop While

```
Dim Shared I as Long
Program
Do
I=I+1
Print I
Loop While I<10
End Program
```

另请参见

WHILE_..._END_WHILE, FOR_..._NEXT, PROGRAM_..._END_PROGRAM

DOUBLEMODE

说明

查询圆滑处理两个运动期间 DOUBLEMODE 插补的状态。可能的值为：

0 - 圆滑尚未开始

1 - 圆滑正在进行

缩写形式

<element>.DMode

语法

?<element>.DoubleMode

域

ELEMENT

参数

< element >: 有效的运动元素

返回值

返回圆滑处理两个运动期间 DOUBLEMODE 插补的状态

< return value >: Long，0 或 1

限制

只读

示例

轴

?A1.DoubleMode

机器人

?SCARA.DoubleMode

另请参见

DOUBLEMODEPERCENTAGE

DOUBLEMODEPERCENTAGE

说明

查询双精度模式期间圆滑的实际百分比。

0 - 圆滑开始

100 - 圆滑结束

缩写形式

<element>.DoubleModePercentage

语法

?<element>.DoubleModePercentage

域

ELEMENT

参数

<element>: 任何有效的运动元素

返回值

返回双精度模式期间圆滑的实际百分比

<return value>: Long，0 或 100

限制

只读

示例

轴

?A1.DoubleModePercentage

机器人

?SCARA.DoubleModePercentage

另请参见

BLENDMETHOD, DOUBLEMODE

DOUBLETOLONG

说明

将 double 数据类型转换成 long 型，支持多种数据的同时转换。

语法

DoubleToLong(<dArryData[*]>,<lArrayData[*]>,<lNum>,<lEndian>)

参数

<lArrayData[*]>: long 型数据存放数组

<dArrayData[*]>: double 数据存放数组

<lNum>: ：double 数据的个数

<lEndian>: 转换格式，0-小端（HGFEDCBA),1-大端（ABCDEFGH）

示例

Dim lArrayData[4] as long

Dim dArrayData[2] as double

dArrayData[1] = 105.56

DoubleToLong(dArrayData, lArrayData,1,0)

?lArrayData[1] => 42096

?lArrayData[2] =>15626

?lArrayData[3] =>55139

?lArrayData[4] =>23104

另请参见

FLOATTOLONG, LONGTODOUBLE, LONGTOFLOAT

DRIVE_ENABLE_I2T_PROTECTION

说明

打开 I2T 保护功能。

注意

此功能仅在 KUKA.ControlStudio 2.7.1 版本或以上版本中可用。

语法

Call Drive_Enable_I2t_Protection

返回

固件版本不支持

"Parameter write failed!" 参数写入失败，检查驱动是否下使能

"The current robot does not support enable I2T function" 当前机型不支持打开 I2t 功能

"The current robot does not support changing I2T function" 当前机型不支持改变 I2t 功能

"Parameter write successed, please restart the control cabinet!" 参数更改成功，请重启控制柜

另请参见

DRIVE_DISABLE_I2T_PROTECTION

DRIVE_DISABLE_I2T_PROTECTION

说明

关闭 I2T 保护功能。

注意

此功能仅在 KUKA.ControlStudio 2.7.1 版本或以上版本中可用。

语法

Call Drive_Disable_I2t_Protection

返回

"Parameter write failed!" 参数写入失败，检查驱动是否下使能

"The current robot does not support enable I2T function" 当前机型不支持打开 I2t 功能

"The current robot does not support changing I2T function" 当前机型不支持改变 I2t 功能

"Parameter write successed, please restart the control cabinet!" 参数更改成功，请重启控制柜

另请参见

DRIVE_ENABLE_I2T_PROTECTION

DRIVETORQUELIMIT

说明

设置单个轴的电机扭矩输出最大值。

语法

<element>.DriveTorqueLimit = 50

? <element>.DriveTorqueLimit

域

ELEMENT

参数

设置电机最大扭矩的百分比

<set value>: Double [1-100]

返回值

返回电机最大扭矩的百分比

<return value>: Double [1-100]

示例

J1. DriveTorqueLimit = 50

? J1. DriveTorqueLimit

Dim dArrayData[2] as double

dArrayData[1] = 105.56

DoubleToLong(dArrayData, lArrayData,1,0)

?lArrayData[1] => 42096

?lArrayData[2] =>15626

?lArrayData[3] =>55139

?lArrayData[4] =>23104

另请参见

RESETPEAKVALUE, SETVELERRTHRESHOLD, COLLISIONDETECT, PEAKTORQUE, PEAKVELERR, VELERRTHRESHOLD, SETDRIVETORQUELIMIT

ELEMENTID

说明

对于轴 - 返回轴的唯一 ID。

对于轴组 - 返回轴组 ID。

语法

? <element>.ElementID

域

ELEMENT

参数

< element >: 有效的运动元素

返回值

返回轴或者轴组的 ID

<return value>:Long

轴：1 到 64。

轴组：65 到 96。

限制

仅独立属性，只读

示例

轴

?A1.ElementID

机器人

?SCARA.ElementID

另请参见

ELEMENTSIZE, ELEMENTNAME, ELEMENTSTATUS

ELEMENTNAME

说明

以字符串形式返回运动元素名称。

语法

?<element>.ElementName

域

ELEMENT

参数

< element >: 有效的运动元素

返回值

以字符串形式返回运动元素名称

<return value>:String

限制

仅独立属性，只读

示例

轴

s = A1.ElementName

机器人

t= SCARA.ElementName

另请参见

ELEMENTID, ELEMENTSIZE

ELEMENTSIZE

说明

返回属于该运动元素的轴数

语法

?<element>.ElementSize

域

ELEMENT

参数

< element >: 有效的运动元素

返回值

返回属于该运动元素的轴数

<return value>:Long

对于轴，始终返回 0。

其它 1 到 64

限制

仅独立属性，只读

示例

轴

n = A1.ElementSize

机器人

m = SCARA.ElementSize

另请参见

ELEMENTID, ELEMENTNAME, ELEMENTSTATUS

ELEMENTSTATUS

说明

返回调用当前执行运动的程序行号。由于运动命令的缓冲机制，可以从远离给出运动的行继续执行程序。因此，在查询任务状态时，返回的行号与给出当前执行运动的行号无关。要获取命令启动当前执行运动的行号，应使用此命令。

返回的字符串格式为：

Main Program line : <line number> File : <program file name>, Library line : <line number of the library> File : <library file name>

语法

? <element>.ElementStatus

域

ELEMENT

参数

< element >: 有效的运动元素

返回值

返回调用当前执行运动的程序行号

<return value>: String

限制

仅独立属性，只读

示例

轴

?A1.ElementStatus

Main Program line : 49 File : TJ3.PRG, Library line : 79 File : STAT.LIB

机器人

?SCARA.ElementStatus

Main Program line : 49 File : TJ3.PRG, Library line : 79 File : STAT.LIB

另请参见

ELEMENTSIZE, ELEMENTID

ERROR

说明

查询上次系统错误消息。如果关键字以任务名称为前缀，则返回该任务中最后一个错误对应的错误消息

注意

此命令返回包含错误编号的错误消息。?ERRORNUMBER 查询仅返回错误的编号。

语法

?Error

?<task>.Error

参数

< task >: 任务名称

返回值

返回包含错误编号的错误消息

<return value>: String

限制

只读

示例

?Error

?MyTask.prg.error

另请参见

TASKERROR

ERRORHANDLERMODE

说明

ErrorHandlerMode 定义错误的影响，由正在执行任务所采取的操作直接生成。

Sys.ErrorHandlerMode = TRUE（默认值）时，停止所有运动，暂停已连接运动元素的所有任务，将 Sys.MotionFlag 设置为“OFF”（关）。

Sys.ErrorHandlerMode = FALSE（出于调试目的）时，仅暂停产生错误的任务，且连接到该任务的运动元素停止。

语法

Sys.ErrorHandlerMode = <value>

?System. ErrorHandlerMode

域

SYSTEM

参数

<value>: Long，0 或 1

0 = FALSE

1 = TRUE

示例

?Sys.ErrorHandlerMode

Sys.ErrorHandlerMode = 1

ERRORNUMBER

说明

查询上次系统错误的编号。

语法

?Sys.ErrorNumber

?System.ErrorNumber

域

SYSTEM

返回值

返回上次系统错误的编号

限制

只读。仅返回错误编号。

示例

?System.ErrorNumber

另请参见

SYSTEM.ERROR

EVENT

说明

此命令定义一个具有特定条件和相关子程序的事件。还可以定义优先级。当条件得到满足时，相关的子程序将被执行。

具有高优先级的事件会中断低优先级的事件并开始执行。

语法

Event<event> <condition> {Priority=<prio>} Do <sub program>

参数

< event >: 事件名称

<condition>: 触发该事件的条件

<prio>: 事件的优先级

<sub program>: 当条件满足时，执行的子程序。

限制

事件优先级必须大于调用该事件的任务的优先级。

UPG 任务的优先级为 15。在 UPG 定义的事件的优先级只能设为 1 到 14。

BKG 任务的优先级为 15。在 BKG 定义的事件的优先级只能设为 1 到 14。

示例

Event EV1 VAR1 = 1 Priority = 1 MYSUB

EVENTDELETE

说明

此命令删除指定的事件。再次执行任务并启用事件之前，事件不会响应指定的条件。

语法

EventDelete <event>

参数

< event >: 事件名称

限制

只写

示例

EventDelete EV1

另请参见

EVENTLIST, EVENTOFF, EVENTON

EVENTLIST

说明

现有事件的名称及其状态的列表。返回的信息将为以下格式：

EventName = <event>, State=Running | Stopped | Error, Priority=<priority>, Owner=<parent task>, Edge=Triggered | UnTriggered, Scan=<scan rate>, Status = On | Off

如果事件已启用，则 State 的值为 1。如果已检测到该条件（即，事件已触发），则 Edge 的值为 1。如果事件操作代码正在运行，则 State 的值设置为 Running。

如果事件已启用，则 Status 为 On。如果已检测到该条件（事件已触发），则 Edge 的值为 Triggered (1)。如果父任务或其中一个事件正在运行，则 Action 设置为 Run。

如果在 EVENTLIST 前加上 <task>，则仅列出属于指定任务的事件。

语法

?Eventlist

?<task>.EventList

参数

< task >: 任务名称

返回值

返回现有事件的名称及其状态的列表

<Return value>: String

限制

只读

示例

?EventList

打印如下：
EventName=EV1,State=Running,Priority=1,Owner=EVENTAPP.UPG,Edge=Triggered,Scan=1,Status=OFF

另请参见

EVENTOFF, EVENTON, ONEVENT, EVENTDELETE

EVENTOFF

说明

此命令禁用指定的事件。禁用的事件在满足事件条件时不会响应。

语法

Eventoff <event>

参数

< event >: 事件名称

限制

只写

示例

Eventoff EV1

另请参见

EVENTLIST, EVENTON, ONEVENT, EVENTDELETE

EVENTON

说明

此命令启用指定的事件。检查指定条件之前必须先启用事件。

语法

Eventon <event>

参数

< event >: 事件名称

限制

只写

示例

Eventon EV1

另请参见

EVENTLIST, EVENTOFF, ONEVENT, EVENTDELETE

EXP

说明

计算指数函数。（e 的数值表达式的幂）。

语法

Exp(<expression>)

参数

<expression>: Double

返回值

<Return value>: Double，± 709.7827 (Log(MaxDouble))

限制

只读

示例

?Exp(1)

*VaSCARA = Exp(Var2*Var3)*

另请参见

LOG

FILESIZE

说明

返回文件的大小（字节）。

语法

FileSize <file_name>

参数

< file_name>: 文件名

返回值

返回文件的大小（字节）

<Return value>: Long，0 到 MaxLong

限制

只读

示例

? FileSize Task1.prg

FILESIZE$

说明

返回文件的大小（字节）。

语法

FileSize$ (<file_name>)

参数

< file_name>: 文件路径的字符串

返回值

返回文件的大小（字节）

<Return value>: Long，0 到 MaxLong

限制

只读

示例

? FileSize$ (“DEMO/APP1/APP1.UPG”)

FLOATTOLONG

说明

切换 float 型数据为 long 型,支持多种数据同时转换。

语法

FloatToLong(< dArrayData[*] >, < lArrayData[*] >,< lNum >, < lEndian >)

参数

< lArrayData[*] >: long 型数据存放数组

< dArrayData[*] >: float 型数据存放数据

< lNum >: float 数据的数量

< lEndian >: 转换格式, 0- 小端 (DCBA), 1-大端 endian(ABCD)

示例

```
Dim lArrayData[4] as long
Dim dArrayData[2] as double
dArrayData[1] = 45.7
FloatToLong(dArrayData, lArrayData,1,0)
?lArrayData[1] result 52684
?lArrayData[2] result 13890
```

另请参见

LONGTOFLOAT , DOUBLETOLONG , LONGTODOUBLE

FOR_..._NEXT

说明

FOR...NEXT 循环多次重复循环中包含的语句，通过递增或递减循环计数器从起始值到结束值进行计数。

只要循环计数器没有达到结束值，循环就会继续执行。

计数器变量最初具有起始值。STEP 值可以是浮点型或整数值。默认 STEP 值为 1。

每次循环重复后，计数器的值按步长值递增或递减，并与结束值进行比较。此时，如果满足以下条件，则循环完成：

- 循环正在增加计数且计数器大于结束值，或者

- 循环正在减少计数且计数器小于结束值。

起始值、结束值和步长可以是表达式。每次在循环中引用时，都会计算结束值和步长表达式。

NEXT 关键字的计数器参数是可选的。但是，嵌套 FOR 循环时，嵌套循环的计数器必须在任何封闭循环的计数器之前显示。

注意

不要使用不包含任何语句或仅包含简单操作语句的循环结构，如果您一定要如此使用，请至少额外插入一条 SLEEP 1 指令，避免出现 CPU 过载，导致系统异常。

语法

FOR <counter> = <start value> TO <end value> {STEP <stepsize>}

{<loopstatements>}

NEXT {<counter>}

参数

<counter>: 变量作为计数器

<start value>: Long、Double 或表达式

<end value>: Long、Double 或表达式

<stepsize>: Long、Double 或表达式，如果未输入，则默认为 1

{<loopstatements>}：循环中的代码

<end value>: 计数器值超过该值时，循环结束。对于正 <stepsize>，<counter> 大于该值时会发生此情况。对于负 <stepsize>，<counter> 小于该值时会发生此情况。

示例

```
Program
dim array1[5][5] as long
dim I as long
For I = 1 to 5
    Print I 'Prints 2, 3, 4, 5
Next I
```

```
For I=4 To 2 Step -0.5
    Print I 'Prints 4.0, 3.5, 3.0, 2.5, 2.0
Next I
```

另请参见

DO ... LOOP, WHILE ... END_WHILE, PROGRAM ... END_PROGRAM

FUNCTION_..._END_FUNCTION

说明

FUNCTION 和 END FUNCTION 关键字用于分隔任务中的函数。这些函数必须在代码的主要部分（由 PROGRAM…END PROGRAM 分隔）之后显示。

如果使用关键字 Public 作为 FUNCTION_..._END_FUNCTION 的前缀，则函数将位于全局操作范围内，并可在任何其他任务中使用。

缩写形式

FUNCTION_..._END_FUNCTION

语法

{Public} Function<name>({{ByVal}<p_1> as <type_1>} {,{ByVal}<p_n> as <type_n>}) As <function type>

{local variable declaration}

{function code}

END Function

参数

{Public}：指定函数范围的可选关键字

<name>: 函数名称

{ByVal}：设置按值传递的参数。

<p_1> 和 <p_n>: 函数参数名称

<type_1> 和 <type_n>: 参数类型

<function type>: 函数返回值类型

{local variable declaration}：在函数中声明局部变量的函数主体

{function code}：函数主体

限制

只写。数组只通过引用传递。

示例

```
Program
    ?add1(5)
End Program
Function add1(byval a as long) as long
    Add1=a+1
End Function' running this program prints 6
```

另请参见

SUB_..._END_SUB, PROGRAM_..._END_PROGRAM

GETIO

说明

该指令用于读取输入输出信号当前的值

语法

? <signal_name>

参数

< singal_name >: 信号名称

返回值

返回当前查询的信号的输入输出值

示例

?signal1

另请参见

PULSE, SETIO

GETARM

说明

该指令用于获取指定点位的手系

语法

? GetArm(< Location >)

参数

< Location >: 笛卡尔点

返回值

返回当前查询的笛卡尔点的手系; 0-Auto, 1- 左手系， 2- 右手系

示例

Common shared P1 as location of xyzr

P1 = #{1,2,3,4;2}

?GetArm(P1)

另请参见

SetArm, ARMFBK

GETWARMSTATE

说明

该指令用于获取热机状态

语法

? GETWARMSTATE

返回值

Finished – 热机完成

Disabled 热机功能关闭

Warm up function is invalid in T1 mode T1 模式下热机功能无效

Remaining time:xx.xxx min 热机剩余 xx.xxx 分钟

示例

? GETWARMSTATE

GOHOME

说明

此命令用于将轴组移动到原点位置。

原点位置在项目设置中设置。

语法

GoHome(<VelocityScale>)

参数

< VelocityScale >: 设置速度比例，Double 0~100；为 -1 时，使用默认值 VelocityScale

示例

GoHme(100)

GOTO

说明

GOTO 语句用于无条件跳转到另一段代码。它引用必须在代码中显示的标签。

标签以名称后跟冒号的形式书写。标签必须在其自己的行上显示，并且可以加注释。

只能在程序、事件、函数或子程序中进行分支。GOTO 语句和标签必须位于相同程序块内。

GOTO 语句可以与条件分支语句（IF..THEN 和 SELECT CASE）进行比较。

语法

GOTO <program label>

<program label>:

参数

<program label>: 程序标签。

限制

只写。

标签必须在它自己的行上显示。

标签不能在任务中重复。

GOTO 分支必须引用相同程序块（PROGRAM…END PROGRAM、SUB…END SUB、FUNCTION…END FUNCTION 和 ONEVENT…END ONEVENT）中的标签。

示例

```
Program
<code>
GOTO Reference1
<code> ‘ This code block will be skipped, when execute GOTO
Reference1:
<code to be executed after GOTO>
End Program
```

GOTOLIMIT

说明

使给定的直线运动延伸直至达到机器人限位之一（XMIN/XMAX、....）的标志。

给定的目标点（在 MOVES 行中）仅用于方向计算。运动的真正目标位于运动起始点与给定点之间的线上，可以更近也可以更远。

语法

<ROBOT>.GoToLimit= <numeric expression>

域

ROBOT

参数

< ROBOT >: 任何有效的机器人

<numeric expression>: Long，0 或 1

限制

只写，仅非独立属性，仅用于 MOVES 命令

示例

```
Move SCARA #{400,0,0,0}
Moves SCARA #{1000,0,0,0} GoToLimit = 1
' this command will move the robot in positive X direction as far as it can go.
```

另请参见

MOVES

GROUPLIST

说明

返回系统中定义的轴组名称的列表。

每个 {group_name} 均后跟该轴组所属轴的名称列表。可以使用通配符。

如果 {group_name} 已指定，则命令返回该特定轴组的轴的名称。

语法

?GroupList {group_name}

参数

{ group_name}：任何有效的机器人

返回值

返回系统中定义的轴组名称的列表。

<return value>: String

限制

只读

示例

?GroupList

‘returns SCARA: A1, A2, A3, A4

另请参见

PLSLIST, AXISLIST, TASKLIST, VARLIST, VARLIST$

GRP_CLOSE_GRIPPER

说明

此命令关闭指定的抓手。

语法

GRP_CLOSE_GRIPPER (<EndEffectorName>, <GripperName>)

参数

< EndEffectorName >: 任何有效的 EndEffector，String

<GripperName>: 属于 EndEffector 的任何有效抓手，String

示例

GRP_CLOSE_GRIPPER ("E1","SCARA")

另请参见

GRP_OPEN_GRIPPER

GRP_OPEN_GRIPPER

说明

此命令打开指定的抓手。

语法

GRP_OPEN_GRIPPER (<EndEffectorName>, <GripperName>)

参数

< EndEffectorName >: 任何有效的 EndEffector，String

<GripperName>: 属于 EndEffector 的任何有效抓手，String

示例

GRP_OPEN_GRIPPER ("E1","SCARA")

另请参见

GRP_CLOSE_GRIPPER

HERE

说明

返回实际机器人笛卡尔坐标或轴角度。

此变量是从电机反馈位置计算的每个采样周期。

它等于 TOCART(PFB)，是 SETPOINT 的对应项。

语法

<point_variable> = <element>.here

? < element >.here

域

ELEMENT

参数

< element >: 任何有效的运动元素

返回值

返回实际机器人笛卡尔坐标或轴角度

<point_variable>: Double 或 Location

限制

仅独立属性，只读

示例

P1= Scara.Here

? Scara.Here

? A1.Here

另请参见

POSITIONFEEDBACK

HEX$

说明

HEX$ 返回数字的字符串表示（十六进制格式）。

字母以大写形式显示。

语法

HEX$(<number>)

参数

<number>: Long，MinLong 到 MaxLong

返回值

返回数字的字符串表示（十六进制格式）

<return value>: String

示例

myStr = Hex$(985421)

?myStr

打印以下内容：F094D

另请参见

BIN$, STRL$

HaltTask

说明

此命令运行将直接暂停下当前调用并运行了 HaltTask 的 UPG/BKG 程序以及停止和程序绑定的机器人运动。

语法

HaltTask()

ICMD

说明

此属性返回驱动器发出的电流命令（按照用户单位）。

驱动器单位使用属性 IFAC 转换为用户单位。

缩写形式

<axis>.ICMD

语法

?<axis>.ICMD

?<joint>.ICMD

域

AXIS

参数

<axis>: 任何有效的轴

<joint>: 任何有效的关节空间坐标

返回值

返回驱动器发出的电流命令（按照用户单位）

<return value>: Double，MinDouble 到 MaxDouble

限制

只读

示例

?A1.icmd

?J1.icmd

另请参见

IFBK

IF_..._THEN_..._ELSE

说明

根据指定条件的状态，此判定结构允许选择执行两段代码分支之一。

条件是一个表达式，计算时，如果结果不为零则为 TRUE，如果结果为零则为 FALSE。

ELSE 部分是可选的，但必须后跟至少一条语句。

IF 语句可以相互嵌套。

没有 ELSE IF 命令。如果想要在 ELSE 之后放一个 IF，则必须将 IF 放在新行上。

语法

```
IF <condition> THEN
<first statement to execute if condition is true>
<multiple statements to execute if condition is true>
{ELSE
<first statement to execute if condition is false>
<multiple statements to execute if condition is false>}
END IF
```

参数

<condition>: 任何表达式均可转换为（True 或 False）

示例

```
IF SYSTEM.din.1 = 0 Then   'check input #1
      move A1 100
Else
      move A1 200
End If
```

另请参见

SELECT_..._CASE, PROGRAM_..._END_PROGRAM

IFBK

说明

此属性返回在驱动器中测量的电流反馈（按照用户单位）。

驱动器单位使用属性 IFAC 转换为用户单位。

缩写形式

<axis>.IFBK

语法

?<axis>.IFBK

?<joint>.IFBK

域

AXIS

参数

<axis>: 任何有效的轴

<joint>: 任何有效的关节空间坐标

返回值

返回在驱动器中测量的电流反馈（按照用户单位）

<return value>: Double，MinDouble 到 MaxDouble

限制

只读

示例

?A1.ifbk

?J1.ifbk

另请参见

ICMD

IMPORT

说明

将库文件导入到任务中。

要使用库文件中定义的子程序，必须首先在程序顶部导入 .lib 文件，然后通过其定义的子程序调用。

语法

Import <lib file name>

参数

< lib file name >: 内存中的 lib 文件名

限制

库必须加载到内存中（使用 load 命令）。

示例

```
Import lib1.lib
Program
    call lib1sub(5) 'call one of lib1 subroutines named lib1sub
End Program
```

INFORMATION

说明

返回系统信息。测量运动总线的平均周期时间。结果是测试得出的平均周期时间。测试持续时间为 500 毫秒。

语法

?Sys.Information

?System.Information

域

SYSTEM

返回值

返回系统信息

<return value>: String

限制

只读

示例

?System.Information

另请参见

NAME

INPUT$

说明

从指定的端口或文件返回字符串。返回字符串长度受 <length> 参数限制。

语法

Input$ ({<length>}, #<DeviceHandle>)

参数

<length>: Long，要返回的字节数。从文件读取时，长度是可选的 - 如果未指定，则读取至行尾或 511 字节，以先到者为准。

<DeviceHandle>: Long，指定串行端口或文件的句柄

返回值

从指定的端口或文件返回字符串。返回字符串长度受 <length> 参数限制

<return value>: String

限制

只读

示例

Str_Var=Input$(50,#1)

Test=Input$(1, #1)

X=Input$(LOC(1), #1)

另请参见

CLOSE, LOC, OPEN_FILE, PRINT_HASH, PRINTPOINTUSING, PRINTUSING, PRINTUSING$, PRINTUSING_HASH-SIGN

INSTR

说明

返回子字符串起始字符在字符串中的位置。

如果其中一个输入字符串为 UTF-8 类型而另一个不是，则非 UTF-8 输入字符串的代码将隐式地转换为 UTF-8 编码方式。

语法

INSTR({<expression>},<search_string>,<sub_string>)

参数

<expression>: Long

<search_string>: String

<sub_string>: String

返回值

返回子字符串起始字符在字符串中的位置。如果 <sub_string> 未在 <search_string> 中出现，则返回值为 0。

<return value>: Long

限制

只读

示例

?INSTR("file ", ".")

打印以下内容：0

?INSTR("file.exe", "exe")

打印以下内容：6

?INSTR(7,"1-2-3-4-5-6", "-")

打印以下内容：8

另请参见

ASC

INT

说明

此函数返回小于或等于数值表达式的最大长整型 (Long) 值。

语法

Int(<expression>)

参数

<expression>: Double，-MaxDouble 到 +MaxDouble

返回值

返回小于或等于数值表达式的最大长整型 (Long) 值

<return value>: Long

限制

只读

示例

Value = Int(12.5)

?Int(12.5)

12 ‘returns 12

?Int(-12.5)

-13 ‘ returns -13

另请参见

ROUND

INTERPOLATIONTYPE

说明

返回当前或已中断运动的插补类型：

0 或 1 = MOVE 轴插补

2 = CIRCLE 圆插补

3 = JOG 运动

6 = MOVES 直线运动

7 = DELAY

8 = 高级插补

14 = 力矩插补

15 = DOPASS

16 = Kino-Dynamic 插补

17 = 正弦波插补

仅限轴 - 返回当前或已中断运动的插补类型：

1 = MOVE 插补

3 = JOG 运动

缩写形式

IType

语法

<element>.IType

域

ELEMENT

参数

<element>: 任何有效的运动元素

返回值

返回当前或已中断运动的插补类型

<return value>: Long

限制

仅对轴组有效

只读

仅独立属性

示例

轴

?A1.Itype

机器人

?SCARA.IType

IPADDRESSMASK

说明

设置或查询以太网接口的 IP 地址和子网掩码。系统会执行主要测试以确保地址和子网掩码匹配。手动更改 IP 地址或子网掩码可能会停止 ControlStudio 与系统之间的通信。IP 地址设置为“dhcp”时，系统将尝试连接 DHCP 服务器并获取 IP 地址。

注意

手动更改 IP 地址或掩码可能会停止 ControlStudio 与系统之间的通信。

通过 DHCP 服务器成功设置 IP 地址后，ControlStudio 将无法更改控制器的 IP 地址。

语法

Sys.IPAddressMask=<address:mask>

System.IPAddressMask=<address:mask>

System.IPAddressMask=<dhcp>

?Sys.IPAddressMask

?System.IPAddressMask

域

SYSTEM

参数

<address:mask>：String

返回值

返回以太网接口的 IP 地址和子网掩码

<return value>: String

限制

从 Config.prg 设置。从终端更改 IP 地址和/或子网掩码可能会导致控制器与 ControlStudio 之间的连接丢失。这仅适用于以太网接口。您必须关闭所有打开的连接，然后再更改 IP 地址和子网掩码。

示例

?Sys.IPAddressMask

sys.IPAddressMask="212.25.84.109:255.255.255.128"

Sys. IPAddressMask="dhcp"

另请参见

ACCEPT, CLOSE, CONNECT, OPENSOCKET, PING

ISMOVING

说明

此属性指示运动控制器是否处于活动状态。此标志指示运动控制器阶段。

0 - 运动元素未移动

1 - 运动元素处于第一个运动阶段（达到巡航速度）。如果巡航速度大于初始速度，则表明轴正在加速。

2 - 运动元素处于恒定速度阶段（巡航）

3 - 运动元素处于第三个运动阶段（达到最终速度）。如果最终速度小于巡航速度，则表明轴正在减速。

特殊模式：

-1 - 运动元素是仅与轴相关的从设备（齿轮或凸轮）。

-2 - 在驱动程序命令中（归位、调谐、...）

-4 - 移动坐标系跟踪模式（除非发出增量移动，否则返回该值）。

注意

isMoving 标志实际上是一个内部状态机的状态变量，采样 din 事件（或记录）时，它始终指示下一次采样的状态。因此，使用 accelecmd 和 ismoving 的时间测量之间可能会相差一次采样。

语法

?<element>.IsMoving

域

ELEMENT

参数

< element >: 有效的运动元素

返回值

返回指示运动控制器是否处于活动状态的标志

<return value>: Long，-4，-2 到 3

限制

只读

示例

轴

```
While A1.IsMoving >0
    Sleep(10)
End While
```

等待运动控制器完成

机器人

?SCARA.IsMoving

另请参见

ISSETTLED, STARTTYPE

ISSETTLED

说明

此标志指示实际的运动元素位置是否在指定的就位范围内。稳定范围通过 PESETTLE 属性定义，并通过 TIMESETTLE 属性进一步限定。

运动控制器完成后，位置误差（目标位置 - 实际位置）的绝对值与 PESETTLE 属性进行比较。

结果在 TIMESETTLE 给定的时间内小于或等于此属性时，将设置 ISSETTLED 标志。TIMESETTLEMAX 属性设置了从运动控制器完成之时起的稳定时间的限制。

0 - 未就位

1 - 就位

此属性与移动坐标系跟踪无关。这种情况下返回值为 0

语法

?<element>.IsSettled

域

ELEMENT

参数

<element>: 任何有效的运动元素

返回值

返回实际的运动元素位置是否在指定的就位范围内

<return value>: Long，0 或 1

0 - 未就位

1 - 就位

限制

只读。

示例

轴

?A1.IsSettled

机器人

?SCARA.IsSettled

另请参见

POSITIONERRORSETTLE, ISMOVING, STARTTYPE

JERK

说明

此属性用于设置运动元素的加加速度（加速度的变化率）。它表现为应用于运动轨迹的平滑量。通常，使用 SMOOTHFACTOR 来设置应用于轨迹的平滑更方便。此属性可在运动命令内使用以覆盖永久值。

大于 0。上限通过加加速度与加速度或减速度之间的比率定义，根据以下关系：

Jerk/Acc < 0.9 * π/5T

其中 T 是采样时间（秒）。对于 2 毫秒的周期时间，该值为 282.743。

语法

<element>.Jerk = <expression>

?<element>.Jerk

域

ELEMENT

参数

< element >: 有效的运动元素

<expression>: Double，加速度的变化率

限制

JERK 仅当 SMOOTHFACTOR 设置为 -1 时有效。

要在任务中设置该值，必须将轴组连接到该任务（使用 ATTACH 命令）。

示例

轴

A1.Jerk = 1.2

机器人

SCARA.Jerk = 1.2

另请参见

ACCELERATION, JERKMAX, JUMP, JUMP3, JUMP3CP, MOVE, MOVES

JERKACCELERATIONPERCENTAGE

说明

定义实际加加速度从给定值（Jerk、Jtran、Jrot）减少并应用于加速阶段的百分比比例。

缩写形式

<element>.JAPERC

语法

<element>.JAPERC=<numeric expression>

域

ELEMENT

参数

< element >: 有效的运动元素

< numeric expression >: 百分比刻度，Double，0.1-100

限制

独立/非独立属性

读/写

示例

轴

A1.japrec = 20

机器人

SCARA.japrec = 20

另请参见

JERKDECELERATIONPERCENTAGE

JERKDECELERATIONPERCENTAGE

说明

定义实际加加速度从给定值（Jerk、Jtran、Jrot）减少并应用于减速阶段的百分比比例。

缩写形式

<element>.JDPERC

语法

<element>.JDPERC=<numeric expression>

域

ELEMENT

参数

< element >: 有效的运动元素

< numeric expression >: 百分比刻度，Double，0.1-100

限制

独立/非独立属性

读/写

示例

轴

A1.jdprec = 20

机器人

SCARA.jdprec = 20

另请参见

JERKACCELERATIONPERCENTAGE

JERKMAX

说明

定义允许的最大加加速度。如果轨迹生成器发出的 JERK 命令高于 JERKMAX，则系统会将该值设置为 JERKMAX 并通知您。

缩写形式

<element>.JMax

语法

<element>.JerkMax = <expression>

?<element>.JerkMax

域

ELEMENT

参数

< element >: 有效的运动元素

< expression >: 允许的最大加加速度，Double，大于 0

限制

JERKMAX 仅当 SMOOTHFACTOR 设置为 -1 时有效。要在任务中设置该值，必须将运动元素连接到该任务（使用 ATTACH 命令）。

示例

轴

A1.JMax=1.2

机器人

SCARA.JMax=1.2

另请参见

ATTACH, JERK

JERKMAXROT

说明

定义机器人的最大旋转加加速度，用于限制 JROT。该值仅限制笛卡尔运动插补（MOVES、CIRCLE）。

此参数不影响轴插补的运动 (MOVE)。

缩写形式

<ROBOT>.jmrot

语法

<ROBOT>.jmrot=<numeric expression>

域

ROBOT

参数

< ROBOT >: 任何有效的机器人

<numeric expression>: Double，0.1 到 Maxdouble

限制

读/写，仅独立属性

示例

jmrot = 6000

另请参见

CIRCLE, MOVES

JERKMAXTRANS

说明

定义机器人的最大平移加加速度，用于限制 JTRAN。该值仅限制笛卡尔运动插补（MOVES、CIRCLE）。

此参数不影响轴插补的运动 (MOVE)。

缩写形式

<ROBOT>.jmtran

语法

<ROBOT>.jmtran=<numeric expression>

域

ROBOT

参数

< ROBOT >: 任何有效的机器人

<numeric expression>: Double，0.1 到 Maxdouble

限制

读/写，仅独立属性

示例

jmtran = 6000

另请参见

CIRCLE, MOVES

JERKROT

说明

定义机器人的旋转加加速度。与 JTRAN 一起，定义笛卡尔运动的加加速度值。

该值仅在两个运动命令中使用：MOVES 和 CIRCLE。在轴插补运动 (MOVE) 中，该值将被忽略。轴组必须使用机器人模型 (model !=1) 进行定义。

该值不得大于 JERKMAXROT。系统始终采用两者中较小的值，并向用户发送通知消息。

缩写形式

<ROBOT>.jrot

语法

<ROBOT>.jrot=<numeric expression>

域

ROBOT

参数

< ROBOT >: 任何有效的机器人

<numeric expression>: Double，0.1 到 Maxdouble

限制

独立/非独立属性

示例

jrot = 6000

另请参见

CIRCLE, MOVES

JERKTRANS

说明

定义机器人的平移加加速度。与 JROT 一起，定义笛卡尔运动的加加速度值。

该值仅在两个运动命令中使用：MOVES 和 CIRCLE。在轴插补运动 (MOVE) 中，该值将被忽略。轴组必须使用机器人模型 (model !=1) 进行定义。

该值不得大于 JERKMAXTRANS。系统始终采用两者中较小的值，并向用户发送通知消息。

缩写形式

<ROBOT>.jtran

语法

<ROBOT>.jtran=<numeric expression>

域

ROBOT

参数

< ROBOT >: 任何有效的机器人

<numeric expression>: Double，0.1 到 Maxdouble

限制

独立/非独立属性

示例

jtran = 6000

另请参见

MOVES, CIRCLE

JOFOLLOW

说明

用于给定笛卡尔点位的 MOVE 指令，控制关节终点位置。

该值用于 MOVE 指令和 JUMP、JUMP3 指令中的轴插补运动。在笛卡尔插补运动（MOVES、CIRCLE、SMOVE、JUMP3CP）中，该值将被忽略。轴组必须使用机器人模型 (model !=1) 进行定义。

该值只能为 0 和 1。

0 – 选择离起点关节角度最近的关节角度。

1 – 第四关节的旋转方向将跟随 Roll 角度的变化方向。

缩写形式

<ROBOT>.jofollow

语法

<ROBOT>. jofollow =<numeric expression>

域

ROBOT

参数

< ROBOT >: 只适用于 SCARA 机器人

<numeric expression>: int，0 或 1

限制

独立/非独立属性

示例

jofollow = 1

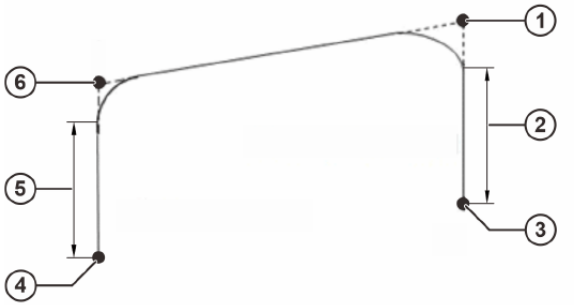
另请参见

JUMP, JUMP3, MOVE

JUMP

说明

Jump 命令以门形运动将机机械臂从一个位置移动到另一个位置。Jump 命令有三个基本阶段：向上、横向和向下。这三个阶段为关节空间坐标系运动，向上和向下运动阶段的工具方向保持不变，向上、横向和向下是以关节运行的形式来执行的。



Jump 运动如上图，6 点，1 点为向上运动终点和向下运动起点。在 Jump 指令中点 6 和点 1 是同样高度，由 LimtZ 决定。

4 点为 Jump 运动起点，3 点为目标点。

5 和 2 段为向上和向下运动的距离，既需要保护不被圆滑的长度，对应 ArchNo 的 AscendingZ 和 DescendingZ。

语法

Jump{对象} Targetpoint {LimZ=<value>} {ArchNo=<value>} {VScale=<value>} {Blend=<value>} {AScale=<value>} {Armcmd=<value>} {WithPls=<value>} {Until <signalName>=<value>} {Acc=<value>} {Dec=<value>} {Jerk=<value>}

参数

{对象}：如果运动对象是机器人，此处可以不添加或是写 SCARA。

Targetpoint：用户想要到达的结束位置，为笛卡尔坐标点位或轴坐标点位。可以是声明好的变量作为输入，也可以是直接输入点位值。

例如笛卡尔坐标 #{0, 0, 0, 0}或轴坐标{0,0,0,0}。

{Limiz}：门字型运动的向上运动的限制高度，默认为 0，设置为-100 即为门字型运动的向上运动到-100mm 开始横向运动。可根据实际需要设置。

{ ArchNo **}**：在 ProjectSetting 的 ArchSettings 里面设置，其中 AscendingZ 是需要保护的不被平滑的到起点的距离，DescendingZ 是下降段需要保护不平滑的距离，距离大小是从到达点的距离开始计算。

{VScale}：全称为 VelocityScale。速度百分比，范围为 0 – 100%。

速度参数受最大值（由用户指定）限制。未设置情况下则会用项目设置中设定的值。

注意，在程序中重新对系统 vScale 赋值以后，其作用范围仅会在该程序内部，不会影响整个系统。

{Blend}：轨迹圆滑百分比，范围为 0 – 100%。

每条 Move 语句中所设定的 Blend 参数意味着，本条语句这段运动轨迹在进入和移出时，切入距离或离开距离与一半本条语句运动轨迹的比值。

例如，当该参数为 100 时，整条轨迹的中心点就成了切入点及移出点，即该轨迹本身就只剩下了一个点，其余部分都用于轨迹圆滑了。当该参数为 25 时，切入点就在轨迹开始后的 1/8 处，移出点在轨迹结束前的 1/8 处。未设置情况下默认值为 0。

{AScale}：加速度百分比，范围为 0 – 100%。

加速度参数受最大值（由用户指定）限制。未设置情况下则会用项目设置中设定的值。

{Armcmd}：机器人的手型。

该标记确定了使用哪种方式通过关节坐标系到达目标位置。手型值 0 为自动，1 为左手型，2 为右手型。

{WithPls}：位置触发器。

指的是在机器人执行该条运动指令的过程中，机器人到达指定位置以后将触发 IO 信号的功能。每个位置触发器都需要在项目设置中进行创建及配置。在写程序时，该参数直接挂载配置好的位置触发器名字即可。

{Until}：一种用于中断执行动作的功能属性。

当配置的信号值等于预期值时，执行动作将停止，下一动作继续。Until 命令应添加在运动命令的末尾，它也是运动命令的可选属性。基本格式为“Until <Signal> = <StopCondition>”。其中<Signal>为数字输入信号，需要事先在项目的“输入输出”页面中创建好。而<StopCondition>为 on/off，是为中断的条件。

{Acc}：全称为 Acceleration。运动曲线的加速度，单位为 mm/sec²。

执行运动指令时，该属性值应小于或等于系统最大加速度。 如果大于系统最大加速度，则使用系统最大加速度值执行运动。

{Dec}：全称为 Deceleration。运动曲线的减速度，单位为 mm/sec²。

执行运动指令时，减速度应小于或等于系统最大减速度。 如果减速度大于系统最大减速度，则以系统最大减速度值执行运动。

{Jerk}：机器人的加加速度。加加速度时是加速度的变化率。

以上属性参数可根据实际需要同时填入一条运动指令当中，且顺序不限。

限制

如果运动元素被其他任务连接，可能无法移动该元素。

只有对于版本高于 2.5 的 Control Studio，所有运动命令才具备 until 功能 (SMOVE 除外）。

示例

Jump SCARA PL1 ArchNo=1 LimZ=0 Blend=50 VScale=100 AScale=20 WithPls=PT_1 ‘设置添加位置触发器

注意，如果不使用 ArchNo，那么向上运动，横向运动和下降运动之间则不会有圆滑

Jump SCARA #{500,0,-200,0} ArchNo=1 LimZ=-100 Blend=50 Until signal1 = On‘添加 Unitl 中断功能，当信号 SIGNAL1 为 on 的时候触发

注意 LimZ 高度不要小于起始点和目标点坐标的高度

```
Jump SCARA PL1 ArchNo=1 LimZ=0 Acc=7000 '设定加速度
Jump SCARA PL1 ArchNo=1 LimZ=0 Dec=7000 '设定减速度
Jump SCARA PL1 ArchNo=1 LimZ=0 Jerk=7000 '设定加加速度
```

完整程序案例如下，其中点位需自行在项目中创建。

```
program
   Attach
   Move PL1 VScale=20 Armcmd=1
Jump SCARA PL2 ArchNo=1 LimZ=0 Blend=50 VScale=100 AScale=20
WithPls=PT_1
   Moves PL3 VScale=15 Blend=10 AScale=20
Jump SCARA PL2 ArchNo=1 LimZ=-50 Blend=50 VScale=100 AScale=20
Until  signal1 =On Armcmd=2
   Move SCARA #{10,10,0,0} abs=0 VScale=20 Blend=10
   Move {10,0,-20,0} VScale=30 abs=0
   Detach
end program
```

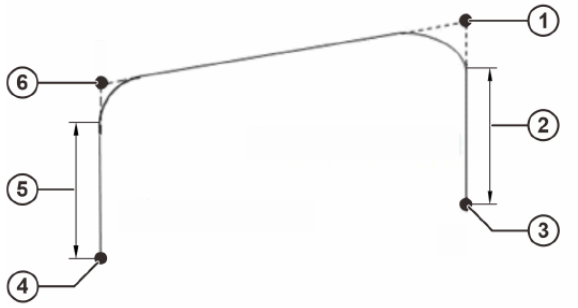
另请参见

JUMP3, JUMP3CP, VELOCITYSCALE, ACCELERATION, DECELERATION, JERK, STARTTYPE, WITHPLS, BLENDPERCENTAGE, MOVE, VSMODE

JUMP3

说明

Jump3 命令用于以门形运动将机器人机械臂从一个位置移动到另一个位置。Jump3 命令有三个基本阶段：向上、横向和向下。向上和向下阶段是笛卡尔坐标空间运动，而横向阶段是关节空间坐标系运动。



如上图 6 点，1 点为向上运动终点(AscendingPoint)和向下运动起点(DescendingPoint）。

4 点位 Jump3 运动起点，3 点为目标点(TargetPoint)。

5 和 2 段为垂直向上和向下运动的距离，既需要保护不被圆滑的长度，对应 ArchNo 的 AscendingZ 和 DescendingZ。

Jump3 同 Jump 不同之处在可以指定向上运动的目标点及向下运动的起始点。点 6 到点 1 的横向运动为关节运动。

语法

Jump3 **{对象}** AscendingPoint = < value > DescendingPoint = < value > TargetPoint = < value > >} {VScale=<value>} {Blend=<value>} {BlendDistance=<value>}{BlendOrientation=<value>} {Blendendprotected=<value>} {Blendstartprotected=<value>} {AScale=<value>} {Armcmd=<value>} {WithPls=<value>} {Until <signalName>=<value>} {Acc=<value>} {Dec=<value>} {Jerk=<value>}{ VTRAN =<value>} { ATRAN =<value>}{ JTRAN =<value>}{ VROT =<value>}{ AROT =<value>}{ JROT =<value>}

参数

{对象}：对象是机器人，此处可以不添加或是写 SCARA。

AscendingPoint：向上运动的目标点，为笛卡尔坐标点位或轴坐标点位。可以是声明好的变量作为输入，也可以是直接输入点位值。例如笛卡尔坐标 #{0, 0, 0, 0}或轴坐标{0,0,0,0}。

DescendingPoint：向下运动的目标点，为笛卡尔坐标点位或轴坐标点位。可以是声明好的变量作为输入，也可以是直接输入点位值。

Target point：用户想要到达的结束位置，为笛卡尔坐标点位或轴坐标点位。可以是声明好的变量作为输入，也可以是直接输入点位值。

{ ArchNo **}**：在 ProjectSetting 的 ArchSettings 里面设置，其中 AscendingZ 是需要保护的不被平滑的到起点的距离，DescendingZ 是下降段需要保护不平滑的距离，距离大小是从到达点的距离开始计算。

{VScale}：全称为 VelocityScale。速度百分比，范围为 0 – 100%。

速度参数受最大值（由用户指定）限制。未设置情况下则会用项目设置中设定的值。

{Blend}：轨迹圆滑百分比，范围为 0 – 100%。

每条 Move 语句中所设定的 Blend 参数意味着，本条语句这段运动轨迹在进入和移出时，切入距离或离开距离与一半本条语句运动轨迹的比值。

例如，当该参数为 100 时，整条轨迹的中心点就成了切入点及移出点，即该轨迹本身就只剩下了一个点，其余部分都用于轨迹圆滑了。当该参数为 25 时，切入点就在轨迹开始后的 1/8 处，移出点在轨迹结束前的 1/8 处。未设置情况下默认值为 0。

{BlendDistance}：轨迹圆滑距离。

每条 Moves 语句中可以采用 BlendDistance 圆滑长度来设定 Blend 参数。

例如，当该参数为 BlendDistance=100 时，表示当前指和下条指令的平滑长度为 100mm，平滑长度不会超过两条指令中移动较短的一半长度。圆滑的切入点为距离当前运动轨迹目标点 100mm 处，切出点为距离下条运动轨迹起点 100mm 处。未设置情况下默认值为 0。

{BlendOrientation}：轨迹姿态圆滑长度。

每条 Moves 语句中可以采用 BlendOrientation 姿态圆滑长度来设定 Blend 参数。

例如，假设当前运动指令的姿态旋转角度大小是 90°，当该参数为 BlendOrientation=20 时，表示当运动到姿态旋转角度到 70°时，剩余的 20°姿态旋转角会和下一段运动指令旋转角度进行融合圆滑。设置的最大值不会超过总旋转角度的一般既 45°。

{Blendstartprotected}：轨迹起始段圆滑保护。

每条 Moves 语句中可以采用 Blendstartprotected 来指定轨迹起始点开始到设定长度不用于圆滑，优先级高于 BlendDistance 和 blendOrientention。如果 Blendstartprotected 设置值大于 0，那么这条轨迹是不会和上一条轨迹进行圆滑。

{Blendendprotected}：轨迹结束段圆滑保护。

每条 Moves 语句中可以采用 Blendendprotected 从设定长度到轨迹目标点这段长度不用于圆滑，优先级高于 BlendDistance 和 blendOrientention。

Blendendprotected 设置值大于 0，那么这条轨迹是不会和下一条轨迹进行圆滑。

{Armcmd}：机器人的手型。

该标记确定了使用哪种方式通过关节坐标系到达目标位置。手型值 0 为自动，1 为左手型，2 为右手型。

{AScale}：加速度百分比，范围为 0 – 100%。

加速度参数受最大值（由用户指定）限制。未设置情况下则会用项目设置中设定的值。

{WithPls}：位置触发器。

指的是在机器人执行该条运动指令的过程中，机器人到达指定位置以后将触发 IO 信号的功能。每个位置触发器都需要在项目设置中进行创建及配置。在写程序时，该参数直接挂载配置好的位置触发器名字即可。

{Until}：一种用于中断执行动作的功能属性。

当配置的信号值等于预期值时，执行动作将停止，下一动作继续。Until 命令应添加在运动命令的末尾，它也是运动命令的可选属性。基本格式为“Until <Signal> = <StopCondition>”。其中<Signal>为数字输入信号，需要事先在项目的“输入输出”页面中创建好。而<StopCondition>为 on/off，是为中断的条件。

{Acc}：全称为 Acceleration。运动曲线的加速度，单位为 mm/sec²。

执行运动指令时，该属性值应小于或等于系统最大加速度。 如果大于系统最大加速度，则使用系统最大加速度值执行运动。

{Dec}：全称为 Deceleration。运动曲线的减速度，单位为 mm/sec²。

执行运动指令时，减速度应小于或等于系统最大减速度。 如果减速度大于系统最大减速度，则以系统最大减速度值执行运动。

{Jerk}：机器人的加加速度。加加速度时是加速度的变化率。

{VTRAN}：全称为 VELOCITYTRANS 平移运动曲线的速度，单位为 mm/sec。

执行运动指令时，该属性值应小于或等于系统最大平移速度。 如果大于系统最大平移速度，则使用系统最大平移速度值执行运动。

{ATRAN}：ACCELERATIONTRANS 平移运动曲线的加速度，单位为 mm/sec²。

执行运动指令时，该属性值应小于或等于系统最大平移加速度。 如果大于系统最大平移加速度，则使用系统最大平移加速度值执行运动。

{DTRAN}：全称为 DECELERATIONTRANS 平移运动曲线的减速度，单位为 mm/sec²。

执行运动指令时，减速度应小于或等于系统最大平移减速度。 如果减速度大于系统最大平移减速度，则以系统最大平移减速度值执行运动。

{JTRAN}：JERKTRANS 机器人的平移运动加加速度。加加速度时是加速度的变化率。

{VROT}：全称为 VELOCITYROT 旋转运动曲线的速度，单位为 degree/sec。

执行运动指令时，该属性值应小于或等于系统最大旋转速度。 如果大于系统最大旋转速度，则使用系统最大旋转速度值执行运动。

{AROT}：ACCELERATIONROT 旋转运动曲线的加速度，单位为 degree/sec²。

执行运动指令时，该属性值应小于或等于系统最大旋转加速度。 如果大于系统最大旋转加速度，则使用系统最大旋转加速度值执行运动。

{DROT}：全称为 DECELERATIONROT 旋转运动曲线的旋转减速度，单位为 degree/sec²。

执行运动指令时，减速度应小于或等于系统最大旋转减速度。 如果减速度大于系统最大旋转减速度，则以系统最大旋转减速度值执行运动。

{JROT}：JERKROT 机器人的旋转运动加加速度。加加速度时是加速度的变化率。

以上属性参数可根据实际需要同时填入一条运动指令当中，且顺序不限。

示例

Jump3 SCARA AscendingPoint = PL1 DescendingPoint = PL2 TargetPoint = PL3 ArchNo = 1 Vtran = 1000 AScale = 20 WithPls=PT_1 ‘设置添加位置触发器

Jump3 SCARA AscendingPoint = PL1 DescendingPoint = PL2 TargetPoint = PL3 ArchNo = 1 Vtran = 1000 Until signal1 = On ‘添加 Unitl 中断功能，当信号 SIGNAL1 为 on 的时候触发

Jump3 SCARA AscendingPoint = PL1 DescendingPoint = PL2 TargetPoint = PL3 ArchNo = 1 Vtran = 1000 Atran = 5000 ‘设置平移运动速度及加速度

Jump3 SCARA AscendingPoint = PL1 DescendingPoint = PL2 TargetPoint = PL3 ArchNo = 1 VRot= 1000 ARot = 5000 ‘设置旋转运动速度及加速度

program

Attach

Move PL4 VScale=20 Armcmd=1

Jump3 SCARA AscendingPoint = PL1 DescendingPoint = PL2 TargetPoint = PL3 ArchNo = 1 Vtran = 1000 AScale = 20 WithPls=PT_1

Move PL4 VScale=15 Blend=10 AScale=20

Jump3 SCARA AscendingPoint = PL1 DescendingPoint = PL2 TargetPoint = PL3 ArchNo = 1 Vtran = 1000 Until signal1 = On

Move PL4 VScale=15 Blend=10 AScale=20

Jump3 SCARA AscendingPoint = #{600,-200,-100} DescendingPoint = #{600,200,-80,50} TargetPoint = #{600,200,-200,60} ArchNo = 1 VRot= 1000 ARot = 5000 BlendDistance=10 Armcmd=2

Moves SCARA #{20,30,0,0} abs=0 VScale=20 Blend=10

Move {0,0,0,0} VScale=20

Detach

end program

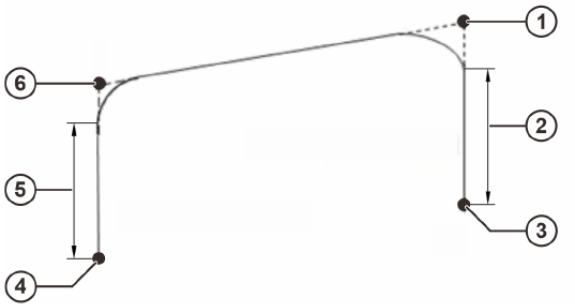
另请参见

JUMP, JUMP3CP, VELOCITYTRANS, JERK, STARTTYPE, WITHPLS, BLENDPERCENTAGE, VSMODE , BLENDPERCENTAGE, BLENDSTARTPROTECTED , BLENDORIENTATION

JUMP3CP

说明

Jump3CP 命令用于以门形运动轨迹将机械臂从一个位置移动到另一个位置。Jump3CP 命令有三个基本阶段：向上、横向和向下。这三个阶段为笛卡尔坐标空间运动。向上和向下运动阶段期间的工具方向保持不变。



如上图 6 点，1 点为向上运动终点(AscendingPoint)和向下运动起点(DescendingPoint）。

4 点为 Jump3cp 运动起始点，3 点为目标点(TargetPoint)。

5 和 2 段为垂直向上和向下运动的距离，既需要保护不被圆滑的长度，对应 ArchNo 的 AscendingZ 和 DescendingZ。

Jump3cp 同 Jump3 不同之处在点 6 到点 1 的横向运动为笛卡尔直线运动。

语法

Jump3cp **{对象}** AscendingPoint = < value > DescendingPoint = < value > TargetPoint = < value > >} {Blend=<value>} {BlendDistance=<value>}{BlendOrientation=<value>} {Blendendprotected=<value>} {Blendstartprotected=<value>} {WithPls=<value>} {Until <signalName>=<value>} { VTRAN =<value>}{ ATRAN =<value>}{ JTRAN =<value>}{ VROT =<value>}{ AROT =<value>}{ JROT =<value>}

参数

{对象}：对象是机器人，此处可以不添加或是写 SCARA。

AscendingPoint：向上运动的目标点，为笛卡尔坐标点位或轴坐标点位。可以是声明好的变量作为输入，也可以是直接输入点位值。例如笛卡尔坐标 #{0, 0, 0, 0}或轴坐标{0,0,0,0}。

DescendingPoint：向下运动的目标点，为笛卡尔坐标点位或轴坐标点位。可以是声明好的变量作为输入，也可以是直接输入点位值。

Target point：用户想要 Jump 运动到达的结束位置，为笛卡尔坐标点位或轴坐标点位。可以是声明好的变量作为输入，也可以是直接输入点位值。

{ ArchNo **}**：在 ProjectSetting 的 ArchSettings 里面设置，其中 AscendingZ 是需要保护的不被平滑的到运动起始点的距离，DescendingZ 是下降段需要保护不平滑的距离，距离大小是按到目标点的距离计算。

{Blend}：轨迹圆滑百分比，范围为 0 – 100%。

每条 Move 语句中所设定的 Blend 参数意味着，本条语句这段运动轨迹在进入和移出时，切入距离或离开距离与一半本条语句运动轨迹的比值。

例如，当该参数为 100 时，整条轨迹的中心点就成了切入点及移出点，即该轨迹本身就只剩下了一个点，其余部分都用于轨迹圆滑了。当该参数为 25 时，切入点就在轨迹开始后的 1/8 处，移出点在轨迹结束前的 1/8 处。未设置情况下默认值为 0。

{BlendDistance}：轨迹圆滑长度。

每条 Moves 语句中可以采用 BlendDistance 圆滑长度来设定 Blend 参数。

例如，当该参数为 BlendDistance=100 时，表示当前指和下条指令的平滑长度为 100mm，平滑长度不会超过两条指令中移动较短的一半长度。圆滑的切入点为距离当前运动轨迹目标点 100mm 处，切出点为距离下条运动轨迹起点 100mm 处。未设置情况下默认值为 0。

{BlendOrientation}：轨迹姿态圆滑长度。

每条 Moves 语句中可以采用 BlendOrientation 姿态圆滑长度来设定 Blend 参数。

例如，假设当前运动指令的姿态旋转角度大小是 90°，当该参数为 BlendOrientation=20 时，表示当运动到姿态旋转角度到 70°时，剩余的 20°姿态旋转角会和下一段运动指令旋转角度进行融合圆滑。设置的最大值不会超过总旋转角度的一般既 45°。

{Blendstartprotected}：轨迹起始段圆滑保护。

每条 Moves 语句中可以采用 Blendstartprotected 来指定轨迹起始点开始到设定长度不用于圆滑，优先级高于 BlendDistance 和 blendOrientention。如果 Blendstartprotected 设置值大于 0，那么这条轨迹是不会和上一条轨迹进行圆滑。

{Blendendprotected}：轨迹结束段圆滑保护。

每条 Moves 语句中可以采用 Blendendprotected 从设定长度到轨迹目标点这段长度不用于圆滑，优先级高于 BlendDistance 和 blendOrientention。

Blendendprotected 设置值大于 0，那么这条轨迹是不会和下一条轨迹进行圆滑。

{WithPls}：位置触发器。

指的是在机器人执行该条运动指令的过程中，机器人到达指定位置以后将触发 IO 信号的功能。每个位置触发器都需要在项目设置中进行创建及配置。在写程序时，该参数直接挂载配置好的位置触发器名字即可。

{Until}：一种用于中断执行动作的功能属性。

当配置的信号值等于预期值时，执行动作将停止，下一动作继续。Until 命令应添加在运动命令的末尾，它也是运动命令的可选属性。基本格式为“Until <Signal> = <StopCondition>”。其中<Signal>为数字输入信号，需要事先在项目的“输入输出”页面中创建好。而<StopCondition>为 on/off，是为中断的条件。

{VTRAN}：全称为 VELOCITYTRANS 平移运动曲线的速度，单位为 mm/sec。

执行运动指令时，该属性值应小于或等于系统最大平移速度。 如果大于系统最大平移速度，则使用系统最大平移速度值执行运动。

{ATRAN}：ACCELERATIONTRANS 平移运动曲线的加速度，单位为 mm/sec^2。

执行运动指令时，该属性值应小于或等于系统最大平移加速度。 如果大于系统最大平移加速度，则使用系统最大平移加速度值执行运动。

{DTRAN}：全称为 DECELERATIONTRANS 平移运动曲线的减速度，单位为 mm/sec²。

执行运动指令时，减速度应小于或等于系统最大平移减速度。 如果减速度大于系统最大平移减速度，则以系统最大平移减速度值执行运动。

{JTRAN}：JERKTRANS 机器人的平移运动加加速度。加加速度时是加速度的变化率。

{VROT}：全称为 VELOCITYROT 旋转运动曲线的速度，单位为 degree/sec。

执行运动指令时，该属性值应小于或等于系统最大旋转速度。 如果大于系统最大旋转速度，则使用系统最大旋转速度值执行运动。

{AROT}：ACCELERATIONROT 旋转运动曲线的加速度，单位为 degree/sec²。

执行运动指令时，该属性值应小于或等于系统最大旋转加速度。 如果大于系统最大旋转加速度，则使用系统最大旋转加速度值执行运动。

{DROT}：全称为 DECELERATIONROT 旋转运动曲线的旋转减速度，单位为 degree/sec²。

执行运动指令时，减速度应小于或等于系统最大旋转减速度。 如果减速度大于系统最大旋转减速度，则以系统最大旋转减速度值执行运动。

{JROT}：JERKROT 机器人的旋转运动加加速度。加加速度时是加速度的变化率。

以上属性参数可根据实际需要同时填入一条运动指令当中，且顺序不限。

限制

如果运动元素被其他任务连接，可能无法移动该元素。

直线运动过程中经过过奇异点会报错停止。

只有对于版本高于 2.5 的 Control Studio，所有运动命令才具备 until 功能 (SMOVE 除外）。

示例

Jump3cp SCARA AscendingPoint = PL1 DescendingPoint = PL2 TargetPoint = PL3 ArchNo = 1 Vtran = 1000 WithPls=PT_1 ‘设置添加位置触发器

Jump3cp SCARA AscendingPoint = PL1 DescendingPoint = PL2 TargetPoint = PL3 ArchNo = 1 Vtran = 1000 Until signal1 = On ‘添加 Unitl 中断功能，当信号 SIGNAL1 为 on 的时候触发

Jump3cp SCARA AscendingPoint = PL1 DescendingPoint = PL2 TargetPoint = PL3 ArchNo = 1 Vtran = 1000 Atran = 5000 ‘设置平移运动速度及加速度

Jump3cp SCARA AscendingPoint = PL1 DescendingPoint = PL2 TargetPoint = PL3 ArchNo = 1 VRot= 1000 ARot = 5000 ‘设置旋转运动速度及加速度

program

Attach

```
    Move PL4 VScale=20 Armcmd=1

Jump3cp SCARA AscendingPoint = PL1 DescendingPoint = PL2 TargetPoint
= PL3 ArchNo = 1 Vtran = 1000 WithPls=PT_1

    Move PL4 VScale=15 Blend=10 AScale=20

Jump3cp SCARA AscendingPoint = PL1 DescendingPoint = PL2 TargetPoint
=       PL3 ArchNo = 1 Vtran = 1000 Until signal1 = On

    Move PL4 VScale=15 Blend=10 AScale=20

Jump3cp SCARA AscendingPoint = #{600,-200,-100,-30} DescendingPoint =
#{600,200,-80,50} TargetPoint = #{600,200,-200,60} ArchNo = 1 VRot= 1000
ARot = 5000 BlendDistance=10

    Moves SCARA #{20,30,0,0} abs=0 VScale=20 Blend=10

    Move {0,0,0,0} VScale=20

    WaitForMotion

    Detach

end program
```

另请参见

JUMP, JUMP3, VELOCITYTRANS, JERK, STARTTYPE, WITHPLS, BLENDPERCENTAGE, VSMODE, BLENDPERCENTAGE, BLENDSTARTPROTECTED , BLENDORIENTATION

LCASE$

说明

LCASE$ 返回所传递字符串的副本，所有大写字母转换为小写字母。

语法

LCASE$(<string>)

参数

<string>: String

返回值

返回所传递字符串的副本，所有大写字母转换为小写字母

<return value>: String

限制

只读

示例

PRINT LCASE$(“ FIRST STEP”)

打印以下内容：first step

另请参见

UCASE$

LEFT$

说明

LEFT$ 返回字符串左侧指定数量的字符。

语法

LEFT$(<string>, <expression>)

参数

<string>: String

<expression>:字符串左侧的字符数，Long，0 到 <string> 的最大长度

返回值

返回字符串左侧指定数量的字符

<return value>: String

限制

只读

示例

Test=”This string is too long”

PRINT LEFT$(Test,4)

打印以下内容： This

另请参见

MID$, RIGHT$

LEN

说明

此函数返回输入字符串的长度，在 ASCII-8 字符串中为字符数，或在 UTF-8 字符串中为符号数。

NULL 字符不会从字符串中切掉，从而导致不同的行为：

?Len("A" + Chr$(0) + "B")

3

Len of Chr$(0) is 1:

?Len(Chr$(0))

1

语法

Len (<string>)

参数

<string>: String

返回值

返回输入字符串的长度，在 ASCII-8 字符串中为字符数，或在 UTF-8 字符串中为符号数

<return value>: Long

限制

只读

示例

? Len (“Hello”)

打印以下内容：5

另请参见

SIZE

LOC

说明

返回在输入缓冲区中等待的字符数。

语法

?Loc(<DeviceHandle>)

参数

<DeviceHandle>: Long，文件句柄，1 到 255

返回值

返回在输入缓冲区中等待的字符数

<return value>: Long

限制

只读

示例

?Loc(1)

另请参见

CLOSE, INPUT$, PRINT HASH, OPEN_FILE, PRINT_HASH, PRINTPOINTUSING, PRINTUSING, PRINTUSING$, PRINTUSING_HASH-SIGN

LOG

说明

返回表达式的自然对数。

语法

Log(<expression>)

参数

<expression>: Double，大于 0

返回值

返回表达式的自然对数

<Return value>: Double

限制

只读

示例

?Log(1)

VaSCARA = Log(Var2)

另请参见

EXP

LOGGER

说明

记录应用程序异常。仅执行打印和记录异常的操作。<error name> 是用户定义的异常名称。

语法

Logger <error name>

参数

< error name >: 用户定义的异常名称。

限制

必须定义 <error name>。仅接受异常名称作为参数。

示例

Logger myerror

另请参见

ONERROR

LTRIM$

说明

LTRIM$ 函数返回删除开头所有空格后字符串的右侧部分。

语法

LTRIM$(<string>)

参数

<string>: String

返回值

返回删除开头所有空格后字符串的右侧部分

<return value>: String

限制

只读

示例

PRINT “first”;LTRIM$ (“ a left-justified string “)

打印以下内容：firsta left-justified string

另请参见

RTRIM$

LONGTOFLOAT

说明

转换 long 型数据为 float 型，支持多种数据同时转换。

语法

LongToFloat(< lArrayData[*] >, < dArrayData[*] >,< lNum >, < lEndian >)

参数

< lArrayData[*] >: long 型数据存储数组

< dArrayData[*] >: float 型数据存储数组

< lNum >: double 数据的个数

< lEndian >: 转换规则, 0- 小端(DCBA) , 1-大端 endian(ABCD)

示例

Dim lArrayData[4] as long

Dim dArrayData[2] as double

lArrayData[1] = 52684

lArrayData[2] = 13890

LongToFloat(lArrayData, dArrayData,1,0)

? dArrayData[1]

Result 45.7

另请参见

FLOATTOLONG , DOUBLETOLONG , LONGTODOUBLE

LONGTODOUBLE

说明

转换 long 型数据为 double 型，支持多种数据同时转换。

语法

LongToDouble(< lArrayData[*] >, < dArrayData[*] >,< lNum >, < lEndian >)

参数

< lArrayData[*] >: long 型数据存储数组

< dArrayData[*] >: array of double

< lNum >: the number of double

< lEndian >: data format, 0- small endian(HGFEDCBA) , 1-big endian(ABCDEFGH)

示例

Dim lArrayData[4] as long

Dim dArrayData[2] as double

lArrayData[1] = 42096

lArrayData[2] = 15626

lArrayData[3] = 55139

lArrayData[4] = 23104

LongToDouble(lArrayData, dArrayData,1,0)

? dArrayData[1]

Result 105.56

另请参见

FLOATTOLONG , DOUBLETOLONG , LONGTOFLOAT

MAINFILENAME

说明

通过 Load$…As 命令加载时，任务可能会收到新的时态名称。MainFileName 属性返回任务的原始文件名。

在库的函数或子程序中查询时，MainFileName 将返回调用任务的名称。

语法

?<task>.MainFileName

? MainFileName

域

TASK

参数

<task>: 内存中的任何任务

返回值

返回任务的原始文件名

<return value>: String

限制

只读。任务必须加载到内存中。

示例

Load$ “Task1.prg” As “Task2.prg”

?Task2.prg.MainFileName

Task1.prg

或从 Task2.prg：

Program

? MainFileName ‘ should return Task1.prg

End Program

另请参见

TASKLIST

MID$

说明

MID$ 函数返回字符串中指定数量的字符，从位置 <start> 处的字符开始。如果 <start> 大于 <string> 的长度，则返回的 <string> 为空。

NULL 字符不会从字符串中切掉：

Mid$ 为第二个字符提供 NULL：

Val = Mid$("A" + Chr$(0) + "B", 2, 1)

Val 的值为“Chr$(0)”

语法

MID$(<string>, <start>, <number>)

参数

<string>: String，要进行处理的字符串

<start>: Long，字符串中的起始位置

<number>: Long，切割长度

返回值

返回字符串中指定数量的字符，从位置 <start> 处的字符开始

<return value>: String

限制

只读。

示例

MID$("Hello", 2, 3)

打印以下内容：ell

另请参见

LEFT$, RIGHT$

MOTION

说明

MOTION 值必须为 1 (ON) 才能成功执行任何运动命令。此外，MOTION 的状态通过伺服中断终止。

如果 MOTION 切换为 0 (OFF)，则中止当前正在进行的任何运动，并刷新运动缓冲区中的任何运动。驱动器仅当 MOTION 关闭时才能启用。

除了 <element>.MOTION 标志外，SYSTEM.MOTION 标志也必须为 ON，才能发出运动命令。

缩写形式

<element>.Motion

语法

<element>.Motion = <value>

?<element>.Motion

域

ELEMENT

参数

< element >: 有效的运动元素

< value >: Long，0、1

0 = OFF

1 = ON

限制

该值必须为 1 (ON) 才能成功执行任何运动命令。

示例

轴

A1.Motion = 1

?A1.Motion

机器人

SCARA.Motion = 1

?SCARA.Motion

另请参见

MOVE, SYSTEM.MOTION

MOTIONOVERLAP

说明

MOTIONOVERLAP 属性定义解释器是否将执行下一条运动命令的条件。

这表示属性设置为 0 (Off) 时，仅当上一条运动命令完成后才会执行下一条运动命令。

如果设置为 1 (ON)，解释器将照常执行下一条运动命令。

1 = ON

0 = OFF

缩写形式

<element>.MotionOverLap

语法

<element>.MotionOverLap = <value>

?<element>.MotionOverLap

域

ELEMENT

参数

< element >: 有效的运动元素

< value >: Long，0、1

0 = OFF

1 = ON

示例

轴

A1.MotionOverLap = 1

?A1.MotionOverLap

机器人

SCARA.MotionOverLap = 1

?SCARA.MotionOverLap

另请参见

WAITFORMOTION

MOVE

说明

执行机器人从当前位置到目标位置的点到点运动（PTP）。

在机器人使用此命令时，它将进行轴插补移动。如果想要进行直线运动，您需要使用 MOVES 命令。

语法

Move {对象} 目标点 {VScale=<value>} {Blend=<value>} {AScale=<value>} {Armcmd=<value>} {WithPls=<value>} {Until <signalName>=<value>} {Abs=<value>} {Acc=<value>} {Dec=<value>} {Jerk=<value>}

参数

{对象}：可以是机器人或单轴。如果运动对象是机器人，此处可以不添加或是写 SCARA。如果运动对象是单轴，此处格式则应为 J<x>，其中<x>为操作轴的编号，例如 J1 就是 1 轴。

目标点：如果 Move 对象是机器人，则此处为笛卡尔坐标点位或轴坐标点位。可以是声明好的变量作为输入，也可以是直接输入点位值。

例如笛卡尔坐标 #{0, 0, 0, 0}或轴坐标{0,0,0,0}。

如果 Move 对象是指定机器人的某个轴，则此处格式应该为角度或长度值，指的是操作轴运动的角度/距离。

MOVE 命令必须指定目标点。

{VScale}：全称为 VelocityScale。速度百分比，范围为 0 – 100%。

速度参数受最大值（由用户指定）限制。未设置情况下则会用项目设置中设定的值。

注意，在程序中重新对系统 vScale 赋值以后，其作用范围仅会在该程序内部，不会影响整个系统。

{Blend}：轨迹圆滑百分比，范围为 0 – 100%。

每条 Move 语句中所设定的 Blend 参数意味着，本条语句这段运动轨迹在进入和移出时，切入距离或离开距离与一半本条语句运动轨迹的比值。

例如，当该参数为 100 时，整条轨迹的中心点就成了切入点及移出点，即该轨迹本身就只剩下了一个点，其余部分都用于轨迹圆滑了。当该参数为 25 时，切入点就在轨迹开始后的 1/8 处，移出点在轨迹结束前的 1/8 处。未设置情况下默认值为 0。

{AScale}：加速度百分比，范围为 0 – 100%。

加速度参数受最大值（由用户指定）限制。未设置情况下则会用项目设置中设定的值。

{Armcmd}：机器人的手型。

该标记确定了使用哪种方式通过关节坐标系到达目标位置。手型值 0 为自动，1 为左手型，2 为右手型。

{WithPls}：位置触发器。

指的是在机器人执行该条运动指令的过程中，机器人到达指定位置以后将触发 IO 信号的功能。每个位置触发器都需要在项目设置中进行创建及配置。在写程序时，该参数直接挂载配置好的位置触发器名字即可。

{Until}：一种用于中断执行动作的功能属性。

当配置的信号值等于预期值时，执行动作将停止，下一动作继续。Until 命令应添加在运动命令的末尾，它也是运动命令的可选属性。基本格式为“Until <Signal> = <StopCondition>”。其中<Signal>为数字输入信号，需要事先在项目的“输入输出”页面中创建好。而<StopCondition>为 on/off，是为中断的条件。

{Abs}：全称为 Absolute。该属性用于定义输入的目标点是绝对位置还是相对位置。

在绝对位置模式下，元素的位置命令是所需的绝对位置。 在相对位置模式下，位置指令是所需的位置变化。 值为 0 代表相对位置，值为 1 代表绝对位置。在未设置情况下默认值为 1。

{Acc}：全称为 Acceleration。运动曲线的加速度，单位为 mm/sec²。

执行运动指令时，该属性值应小于或等于系统最大加速度。 如果大于系统最大加速度，则使用系统最大加速度值执行运动。

{Dec}：全称为 Deceleration。运动曲线的减速度，单位为 mm/sec²。

执行运动指令时，减速度应小于或等于系统最大减速度。 如果减速度大于系统最大减速度，则以系统最大减速度值执行运动。

{Jerk}：机器人的加加速度。加加速度时是加速度的变化率。

使用加加速度使运动曲线平滑是显而易见的。 通常使用 SmoothFactor 来设置应用于运动曲线的平滑度更为方便。 使用 SmoothFactor 可以让用户明确地设置 jerk。 加加速度是根据 SmoothFactor 进行内部计算的。

以上属性参数可根据实际需要同时填入一条运动指令当中，且顺序不限。

限制

如果运动元素被其他任务连接，可能无法移动该元素。

只有对于版本高于 2.5 的 Control Studio，所有运动命令才具备 until 功能 (SMOVE 除外）。

示例

```
Move PL1 ‘笛卡尔坐标或轴坐标，全部参数皆使用默认值
Move #{370.242, -151.397, -50, -22.24} ‘手动输入笛卡尔坐标点位
Move {20.837, 9.3, -50, 0} ‘手动输入轴坐标点位
Move J1 40 ‘仅进行单轴运动
Move PL2 Blend=35 ‘添加轨迹圆滑属性
Move PL1 VScale=50 ‘调整速度百分比
Move PL2 AScale=40 ‘调整加速度百分比
```

注意，输入点位的格式并不会影响其他参数的设置，甚至不同的参数彼此之间都是独立互不影响的。另外，在 Move 指令的运动开始执行以后，程序指针就会继续向下，指令预读功能将开始生效。如果期望在该指令之后卡住程序指针以中断指令预读，则需要在后面添加 WaitForMotion 指令。

另外，由于 Blend 功能是基于上下两条运动指令的速度进行矢量叠加实现的，如果两条运动指令速度都很大，则有可能在轨迹圆滑阶段出现超速的问题。此

外，由于轨迹圆滑功能是基于速度矢量叠加实现的效果，因此在速度进行调整或机器人停止再启动以后，机器人在轨迹圆滑阶段的运动轨迹将出现变化且不可预测。

Move PL1 Armcmd=1 ‘指定当前 Move 指令的手型

Armcmd = 2 ‘切换全局接下来所有运动指令的手型

Move PL2 ‘默认指令的手型为上一句所设定的 2（右手型）。

Move PL1 WithPls=PT_1 ‘设置添加位置触发器

Move PL2 Until SIGNAL1=on ‘添加 Unitl 中断功能，当信号 SIGNAL1 为 on 的时候触发

Move PL1 Until SIGNAL2=off ‘添加 Unitl 中断功能，当信号 SIGNAL2 为 off 的时候触发

注意，设置的手型值必须与点位的手型值保持一致，或起码有一方是 0（自动），否则会报手型不匹配的错误。

另外，在添加位置触发器之前需要在项目设置中先配置好。

最后，这里的 SIGNAL1 和 SIGNAL2 都需要实现在输入输出配置中配好，如果这两个信号本身都不存在，执行起来会报错。

Move PL1 abs=0 ‘设定输入的目标点为相对位置

Move PL2 abs=1 ‘设定输入的目标点为绝对位置

Move PL3 ‘默认也是绝对位置

Move PL1 Acc=7000 ‘设定加速度

Move PL1 Dec=7000 ‘设定减速度

Move PL1 Jerk=7000 ‘设定加加速度

完整程序案例如下，其中点位需自行在项目中创建。

```
program
    Attach
    Armcmd=0
    Move J1 -50 VScale=20
    Move PL1 VScale=20 Blend=30
    Move PL2 VScale=15 Blend=10 AScale=20
    Move PL3 VScale=30 AScale=20 Until SIGNAL1=on
    Move PL1 VScale=50 AScale=25 WithPls=PT_1
    Move PL2 Blend=10 VScale=15 AScale=20
    Move PL3 VScale=30 Armcmd=2
    Armcmd=2
    Move PL1 AScale=20 Blend=20
```

```
    Armcmd=0
    Move PL2 VScale=15 Blend=10 AScale=20 Acc=7000 Dec=7000
Jerk=7000
    Move J3 -30 VScale=20
    Move SCARA #{10,10,0,0} abs=0 VScale=20 Blend=10
    Move {10,0,-20,0} VScale=30 abs=0
    Detach
end program
```

另请参见

JUMP, ABSOLUTE, ACCELERATION, DECELERATION, JERK, STARTTYPE, WITHPLS, BLENDPERCENTAGE, BLENDMETHOD, CIRCLE, MOTION, MOVES, POSITIONFINAL, SELECTAXES, WAITFORMOTION, VSMODE

MOVES

说明

MOVES 命令沿笛卡尔坐标系中的直线移动机器人。

运动的参数为：VTRAN、ATRAN、DTRAN、JTRAN、VRTO、AROT、DROT、JROT。这些属性的语法为：

<Property Name> = <value>

在 MOVES 命令期间覆盖永久值。命令的所有标准运动规格也可用 STARTTYPE、ABS，具有完全相同功能。对于版本高于 2.5 的 Control Studio，所有运动命令都将添加一个 until 功能（SMOVE 除外）。

语法

Moves {对象} 目标点 { VTRAN =<value>} { ATRAN =<value>}{ JTRAN =<value>}{ VROT =<value>}{ AROT =<value>}{ JROT =<value>} {Blend=<value>}{BlendDistance=<value>}{BlendOrientation=<value>} {Blendendprotected=<value>} {Blendstartprotected=<value>} {WithPls=<value>} {Until <signalName>=<value>} {Abs=<value>}

参数

{对象}：是机器人，此处可以不添加或是写 SCARA。

目标点：为笛卡尔坐标点位或轴坐标点位。可以是声明好的变量作为输入，也可以是直接输入点位值。

例如笛卡尔坐标 #{0, 0, 0, 0}或轴坐标{0,0,0,0}。

{Blend}：轨迹圆滑百分比，范围为 0 – 100%。

每条 Moves 语句中所设定的 Blend 参数意味着，本条语句这段运动轨迹在进入和移出时，切入距离或离开距离与一半本条语句运动轨迹的比值。

例如，当该参数为 100 时，整条轨迹的中心点就成了切入点及移出点，即该轨迹本身就只剩下了一个点，其余部分都用于轨迹圆滑了。当该参数为 25 时，切入点就在轨迹开始后的 1/8 处，移出点在轨迹结束前的 1/8 处。未设置情况下默认值为 0。

{BlendDistance}：轨迹圆滑距离。

每条 Moves 语句中可以采用 BlendDistance 圆滑长度来设定 Blend 参数。

例如，当该参数为 BlendDistance=100 时，表示当前指和下条指令的平滑长度为 100mm，平滑长度不会超过两条指令中移动较短的一半长度。圆滑的切入点为距离当前运动轨迹目标点 100mm 处，切出点为距离下条运动轨迹起点 100mm 处。未设置情况下默认值为 0。

{BlendOrientation}：轨迹姿态圆滑长度。

每条 Moves 语句中可以采用 BlendOrientation 姿态圆滑长度来设定 Blend 参数。

例如，假设当前运动指令的姿态旋转角度大小是 90°，当该参数为 BlendOrientation=20 时，表示当运动到姿态旋转角度到 70°时，剩余的 20°姿态旋转角会和下一段运动指令旋转角度进行融合圆滑。设置的最大值不会超过总旋转角度的一半，即 45°。

{Blendstartprotected}：轨迹起始段圆滑保护。

每条 Moves 语句中可以采用 Blendstartprotected 来指定轨迹起始点开始到设定长度不用于圆滑，优先级高于 BlendDistance 和 blendOrientention。如果 Blendstartprotected 设置值大于 0，那么这条轨迹是不会和上一条轨迹进行圆滑。

{Blendendprotected}：轨迹结束段圆滑保护。

每条 Moves 语句中可以采用 Blendendprotected 从设定长度到轨迹目标点这段长度不用于圆滑，优先级高于 BlendDistance 和 blendOrientention。

Blendendprotected 设置值大于 0，那么这条轨迹是不会和下一条轨迹进行圆滑。

{Armcmd}：机器人的手型。

该标记确定了使用哪种方式通过关节坐标系到达目标位置。手型值 0 为自动，1 为左手型，2 为右手型。

{WithPls}：位置触发器。

指的是在机器人执行该条运动指令的过程中，机器人到达指定位置以后将触发 IO 信号的功能。每个位置触发器都需要在项目设置中进行创建及配置。在写程序时，该参数直接挂载配置好的位置触发器名字即可。

{Until}：一种用于中断执行动作的功能属性。

当配置的信号值等于预期值时，执行动作将停止，下一动作继续。Until 命令应添加在运动命令的末尾，它也是运动命令的可选属性。基本格式为“Until <Signal> = <StopCondition>”。其中<Signal>为数字输入信号，需要事先在项目的“输入输出”页面中创建好。而<StopCondition>为 on/off，是为中断的条件。

{Abs}：全称为 Absolute。该属性用于定义输入的目标点是绝对位置还是相对位置。

在绝对位置模式下，元素的位置命令是所需的绝对位置。 在相对位置模式下，位置指令是所需的位置变化。 值为 0 代表相对位置，值为 1 代表绝对位置。在未设置情况下默认值为 1。

{VTRAN}：全称为 VELOCITYTRANS 平移运动曲线的速度，单位为 mm/sec。

执行运动指令时，该属性值应小于或等于系统最大平移速度。 如果大于系统最大平移速度，则使用系统最大平移速度值执行运动。

{ATRAN}：ACCELERATIONTRANS 平移运动曲线的加速度，单位为 mm/sec^2。

执行运动指令时，该属性值应小于或等于系统最大平移加速度。 如果大于系统最大平移加速度，则使用系统最大平移加速度值执行运动。

{DTRAN}：全称为 DECELERATIONTRANS 平移运动曲线的减速度，单位为 mm/sec^2。

执行运动指令时，减速度应小于或等于系统最大平移减速度。 如果减速度大于系统最大平移减速度，则以系统最大平移减速度值执行运动。

{JTRAN}：JERKTRANS 机器人的平移运动加加速度。加加速度时是加速度的变化率。

{VROT}：全称为 VELOCITYROT 旋转运动曲线的速度，单位为 degree/sec。

执行运动指令时，该属性值应小于或等于系统最大旋转速度。 如果大于系统最大旋转速度，则使用系统最大旋转速度值执行运动。

{AROT}：ACCELERATIONROT 旋转运动曲线的加速度，单位为 degree/sec^2。

执行运动指令时，该属性值应小于或等于系统最大旋转加速度。 如果大于系统最大旋转加速度，则使用系统最大旋转加速度值执行运动。

{DROT}：全称为 DECELERATIONROT 旋转运动曲线的旋转减速度，单位为 degree/sec^2。

执行运动指令时，减速度应小于或等于系统最大旋转减速度。 如果减速度大于系统最大旋转减速度，则以系统最大旋转减速度值执行运动。

{JROT}：JERKROT 机器人的旋转运动加加速度。加加速度时是加速度的变化率。

以上属性参数可根据实际需要同时填入一条运动指令当中，且顺序不限。

限制

如果运动元素被其他任务连接，可能无法移动该元素。

直线运动过程中经过过奇异点会报错停止。

只有对于版本高于 2.5 的 Control Studio，所有运动命令才具备 until 功能 (SMOVE 除外）。

示例

Moves PL1 ‘笛卡尔坐标或轴坐标，全部参数皆使用默认值

Moves #{370.242, -151.397, -50, -22.24} ‘手动输入笛卡尔坐标点位

Moves {20.837, 9.3, -50, 0} ‘手动输入轴坐标点位

Moves PL1 BlendDistance=100 ‘添加轨迹圆滑属性

Move PL2 Blend=35 ‘添加轨迹圆滑属性

Move PL1 VScale=50 ‘调整速度百分比

Move PL2 AScale=40 ‘调整加速度百分比

注意，输入点位的格式并不会影响其他参数的设置，甚至不同的参数彼此之间都是独立互不影响的。另外，如果期望在该指令之后卡住程序指针以中断指令预读，则需要在后面添加 WaitForMotion 指令。

另外，由于 Blend 功能是基于上下两条运动指令的速度进行矢量叠加实现的，如果两条运动指令速度都很大，则有可能在轨迹圆滑阶段出现超速的问题。此外，由于轨迹圆滑功能是基于速度矢量叠加实现的效果，因此在速度进行调整或机器人停止再启动以后，机器人在轨迹圆滑阶段的运动轨迹将出现变化且不可预测。

Moves PL1 WithPls=PT_1 ‘设置添加位置触发器

Moves PL2 Until SIGNAL1=on ‘添加 Unitl 中断功能，当信号 SIGNAL1 为 on 的时候触发

Moves PL1 Until SIGNAL2=off ‘添加 Unitl 中断功能，当信号 SIGNAL2 为 off

的时候触发

Moves PL1 Armcmd=1 ‘指定当前 Moves 指令的手型

注意，直线运动目标点手型由起点决定，如果指定手型同起点手型不一致，会经过奇异点，一般不需另外指定，如需指定建议先以指定手型运动到直线运动起点再开始直线运动。

Move PL1 Armcmd=1 ‘指定当前 Move 指令的手型

Moves PL2

另外，在添加位置触发器之前需要在项目设置中先配置好。

最后，这里的 SIGNAL1 和 SIGNAL2 都需要实现在输入输出配置中配好，如果这两个信号本身都不存在，执行起来会报错。

Moves SCARA #{60,50,0, 100} abs=0 BlendOrientation=20 BlendDistance=30’指定平移圆滑长度和姿态圆滑长度

注意，同时指定平移圆滑长度和姿态圆滑长度会按相对各自一半长度的相对大小来决定。假设当前段轨迹的平移总长度为 60，BlendDistance=30 那么实际相对一半长度的距离为 30/30=100%，同时此段轨迹的总姿态旋转角度为 100°，BlendOrientation=20，那么实际相对一半长度的比值为 20/50=40%，小于平移的 100%，那么实际圆滑是按平移的圆滑开始点开始进行圆滑。

Moves PL1 abs=0 ‘设定输入的目标点为相对位置

Moves PL2 abs=1 ‘设定输入的目标点为绝对位置

Moves PL3 ‘默认也是绝对位置

Moves PL1 VTRAN=1000 ‘设定平移运动加速度

Moves PL1 ATRAN =5000 ‘设定平移运动减速度

Moves PL1 JTRAN =5000 ‘设定平移运动加加速度

Moves PL1 VROT=100 ‘设定旋转运动加速度

Moves PL1 AROT =1000 ‘设定旋转运动减速度

Moves PL1 JROT =7000 ‘设定旋转运动加加速度

完整程序案例如下，其中点位需自行在项目中创建。

```
program
    Attach
    Move PL1 VScale=20 Armcmd=1
    Moves PL2 VScale=15 Blend=10 AScale=20
    Moves PL3 VScale=30 AScale=20 Until SIGNAL1=on
    Moves PL1 VScale=50 AScale=25 WithPls=PT_1
    Moves PL2 BlendDistance=10 VScale=15 AScale=20
    Moves PL3 VTran=300 ATran=1000 JTran=5000
    Move PL1 AScale=20    Armcmd=2
    Move PL2 VRot=100 ARot=500 JRot=1000
```

```
Moves SCARA #{10,10,0,100} abs=0 VTran=300 ATran=1000 VRot=100
Moves SCARA #{-10,-10,0,-100} abs=0  BlendOrientation=20
Moves SCARA #{-10,-10,0, 100} abs=0  BlendOrientation=20
Moves SCARA #{60,50,0, 100} abs=0  BlendOrientation=20
BlendDistance=30
Move PL1 VScale=20
Moves SCARA #{-10,-10,0, 100} abs=0  BlendOrientation=20
Moves SCARA #{-10,-10,0, 100} abs=0  Blendstartprotected=20
Move PL1 VScale=20
Moves SCARA #{-50,-50,0, 100} abs=0  Blendendprotected =20
Moves SCARA #{-100,100,0, -100} abs=0  Blend=50
Moves SCARA #{100,-100,0, 100} abs=0  Blendendprotected =50
Move {10,0,-20,0} VScale=30 abs=0
Detach
end program
```

另请参见

VELOCITYTRANS, JERK, STARTTYPE, WITHPLS, BLENDPERCENTAGE, CIRCLE, DEST, MOVE, ABSOLUTE, ACCELERATIONMAXROT, ACCELERATIONMAXTRANS, ACCELERATIONROT, ACCELERATIONTRANS, BLENDMETHOD, DECELERATIONROT, DECELERATIONTRANS, JERKMAXROT, JERKMAXTRANS, JERKROT, JERKTRANS, VELOCITYFINALROT, VELOCITYFINALTRANS, VELOCITYMAXROT, VELOCITYMAXTRANS, VELOCITYROT, VELOCITYROTVALUE, VELOCITYTRANSVALUE, VSMODE, XMAX, XMIN, YMAX, YMIN, ZMAX, ZMIN, BLENDPERCENTAGE, BLENDSTARTPROTECTED , BLENDORIENTATION

MSG

说明

返回用户定义异常的消息。

语法

?<user exception>.Msg

域

ERROR

参数

< user exception >: 用户定义的错误名称

返回值

返回用户定义异常的消息

<return value>: String

限制

只读。

必须定义 <user exception>。

示例

?myerror.Msg'assuming myerror was defined by the user.

另请参见

ERROR.NUM

NAME

说明

此属性设置控制器的名称。该查询返回控制器的名称。如果未设置任何名称（或为空字符串），则回复为“no name”（无名称）。

语法

Sys.Name = “<name>”

System.Name = “<name>”

?Sys.Name

?System.Name

域

SYSTEM

参数

<name>: 要设置的新名称，String

返回值

返回控制器的名称

<return value>: 系统名称

限制

system.name 的长度不得超过 11 个字符。

示例

System.Name = “XY Table”

?System.name

另请参见

INFORMATION

NOOFCOORDINATES

说明

返回点的坐标数。

语法

<long_variable> = noofcoordinates (<point_expression>)

参数

< point_expression >: Joint 或 Location

返回值

返回点的坐标数

<long_variable>: Long

示例

LongVar = NOOFCOORDINATES(#{0.0, 10.0, 20.0})

? NOOFCOORDINATES(JointVar)

LongVar = NOOFCOORDINATES(Robot.PCMD)

NUM

说明

返回用户定义异常的编号。

语法

?<user exception>.Num

域

ERROR

参数

< user exception >: 用户定义的错误名称

返回值

返回用户定义异常的编号

<return value>: String

限制

只读。必须定义 <user exception>。

示例

?myerror.num'assuming myerror was defined by the user.

另请参见

ERROR.MSG

NUMBEROFLOOPS

说明

返回长整型 (Long) 值，表示任务循环剩余次数

如同从本任务查询一样，可以从另一个任务的代码行中查询。

如果从同一个任务查询，则无需任务名称。

语法

?<task>.NumberOfLoops

?NumberOfLoops

域

TASK

参数

<task>: 内存中的任何任务

返回值

返回长整型 (Long) 值，表示任务循环剩余次数

<return value>: Long，1 到 MaxLong。

限制

只读。任务必须加载到内存中

示例

StartTask Task1.prg Priority=3 NumberOfLoops=3 ‘Run 3 times

?Task1.prg.NumberOfLoops

打印以下内容：3

或从 Task1.prg：

program

? NumberOfLoops ‘ should return 1

end program

ONERROR

说明

ONERROR 语句用于捕获和处理任务中的同步和异步错误。

同步错误是用户任务导致的错误，可通过解释器检测，而异步错误与程序代码的特定行无关。

示例包括位置跟踪误差。任务中未被 TRY/FINALLY 机制捕获的错误将被 ONERROR 捕获。

捕获到错误时，将运行指定的错误处理代码并停止任务。任务处于状态 4。

通过在错误处理代码中显式输入 CONTINUETASK 命令可恢复任务执行。

注意

SYNCHRONOUS 错误停止通过“onerror”方式捕获错误的任务，因此需要“continuetask”

ASYNCHRONOUS 错误即使从“onerror”捕获错误也不会停止任务，因此任务会自动继续。

语法

```
Program          'Beginning of program
 <Code>
 OnError         'Start of OnError block
  {catch <error number X>
   <code to execute when error X occurs>}
  {catch <error number Y>
   <code to execute when error Y occurs>}
  {catch else
   <code to execute for all other errors>}
 End OnError       'End of OnError block
 <Code>
End Program        'End of program
```

限制

ONERROR 仅捕获与定义它们的任务相关联的错误。

不允许嵌套 ONERROR。

示例

```
OnError
catch 3017
print "Position Following Error"
System.motion = 0'Stop all motion
killtask task2.prg
```

End OnError

另请参见

LOGGER, TRY ... END_TRY

ONEVENT

说明

ONEVENT 命令定义事件启用（使用 EVENTON）后定期检查事件条件是否发生的同步任务。

当条件满足时，将执行事件操作块。事件条件可以是任何条件表达式。

事件操作代码作为其中定义该代码的任务的中断运行。

同一任务中的多个事件可彼此中断，但其自身并非多任务。不同任务中定义的事件是多任务的。

Priority 是所执行操作代码的优先级。可为其分配 1 到 16 之间的任意值。事件优先级必须高于其中定义该事件的任务的优先级。

ScanTime 被给定为 SERCOS 时间周期数。值为 1 会在每个周期检查该条件。

如果定义的事件没有任何条件，则每次扫描时执行事件操作，从而允许定期执行应用程序代码。

语法

OnEvent <event> {<condition>} {Priority=<priority>} {ScanTime=}

<command block that defines the action>

End Onevent

参数

<event>: 字母数字字符串，事件名称

<condition>: 表达式

<priority>: Long

<scan time>: Long

限制

必须启用事件（使用 EVENTON）才能进行处理。事件无法终止其父任务（其中定义该事件的任务）。事件条件不包含无法在同一周期中传递的变量或属性。SYSTEM.ALOAD 不能在事件条件中。不要在子程序或用户函数中声明事件。OnEvent 无法通过 IF、WHILE 或其他循环定义。

示例

```
OnEvent EV1 SysDin.1=1 'Trigger event when input 1 is 1
    Move X-axis 50000
End OnEvent
```

另请参见

EVENTLIST, EVENTOFF, EVENTON

ONSYSTEMERROR

说明

ONSYSTEMERROR 语句用于捕获和处理系统中的错误。

它是通过组合 TRY、ONERROR 和 ONSYSTEMERROR 形成的分层错误处理结构的上层。

ONSYSTEMERROR 在任务主体中写入，但任何时候系统中只能存在一个实例。

它用于捕获所有任务中的同步和异步错误，以及系统上下文中发生的错误。

同步错误由用户任务导致，可通过解释器检测，而异步错误与程序代码的特定行无关。示例包括位置跟踪误差。

系统错误与特定任务无关。系统错误的一个例子是位置跟踪误差，该误差是由于某些外力施加到未连接到任务的轴上而发生的。

捕获到错误时，将执行指定的错误处理代码并停止任务。任务处于状态 4。通过在错误处理代码中显式输入 CONTINUETASK 命令可恢复任务执行。

ONSYSTEMERROR 用于捕获并非专门通过 TRY 或 ONERROR 捕获的错误。然后，它用于执行系统的有序关闭，或有序的恢复程序。

注意

SYNCHRONOUS 错误停止通过“onerror”方式捕获错误的任务，因此需要“continuetask”。

ASYNCHRONOUS 错误即使从“onsystemerror”捕获错误也不会停止任务，因此任务会自动继续。

语法

```
Program                 'Beginning of program
  <Code>
  OnSystemError         'Start of OnSystemError block
     {catch <error number X>
        <code to execute when error X occurs>}
     {catch <error number Y>
        <code to execute when error Y occurs>}
     {catch else
        <code to execute for all other errors>}
  End OnSystemError      'End of OnSystemError block<br>
  <Code>
End Program             'End of program
```

限制

系统中只能显示一个 ONSYSTEMERROR 实例

示例

```
OnSystemError
```

```
catch 12055        'catch Missing SERCOS telegram error
 Print "Missing telegram(s): communication interrupted"
End OnSystemError
```

另请参见

ONERROR

OPEN_FILE

说明

打开现有文件或使用字符串表达式中给定的名称创建新文件（在“写入”模式下）。

文件名不应超过 8 个字符，且为下列扩展名之一：

PRG、DAT、TSK、CMP 用于文件，在闪存盘中打开

或 REC、TXT，在 RAM 磁盘中打开。

根据模式标志打开文本文件以读取、写入或附加到现有文件。

“r” - 打开文本文件进行读取

“w” - 截断为零长度或创建用于写入的文本文件

“a” - 附加；打开或创建文本文件以在文件末尾写入

使用 APPEND 在文件原始内容的末尾添加新行。

使用 WRITE 覆盖以前的文件或创建新文件。

语法

Open<file name>MODE= <mode flag>As #<DeviceHandle>

参数

< file name>: String，不超过 12 个字符（8 个字符的名称 + 点 + 3 个字符的扩展名）

<mode flag>: String

“r” - 读取模式

“w” - 写入模式

“a” - 附加模式

<DeviceHandle>: Long，文件句柄，1 到 255

限制

对于读取和附加模式，定义的文件应存在于 DiskOnChip 中

示例

Open "File1.PRG" Mode="w" as #1

或

Common shared FileNameStr as string = "File1.PRG"

Common shared ModeStr as string = "w"

Open FileNameStr Mode= ModeStr as #1

另请参见

CLOSE, INPUT$, LOC

OPENSOCKET

说明

创建 TCP 套接字并将套接字描述符设置到指定的设备句柄。

语法

OpenSocket Options=<num> as # <device number>

参数

Options：设置套接字类型，Long

0: 数据将被缓存知道端口缓冲区数据达到储存上限或者超时，该模式可以被用于提升网络质量

1: NO_DELAY. 立刻发送数据.

<device number>: 文件句柄，Long

限制

无效选项。

示例

OpenSocket Options=1 as #1

另请参见

CLOSE, CONNECT, ACCEPT, PING, IPADDRESSMASK

OPMODE

说明

轴属性。分配此属性允许您更改驱动器运行模式。

2 - TORQUEMODE

不会检查位置误差或加速度最大值，但会照常检查 vfb < vospd、te < terrmax 和 tcmd < tmax。

1 - VELOCITYMODE

不会检查位置误差，但会照常检查：vfb < vospd、te < terrmax、acmd < amax 和 tcmd < tmax。

0 - POSITIONMODE

检查所有标准偏差阈值：pe < pemax、vfb < vospd、te < terrmax、acmd < amax 和 tcmd < tmax。

DRIVEOPMODE 不会更改轴运行模式，因此轴不“知道”其处于例如 **TORQUE** 模式，并会继续检查位置和速度。此外，在这种情况下，**TORQUE** 命令无法正常运行。

注意

注意 DRIVEOPMODE 与 OPMODE 之间的区别。

命令 DRIVEOPMODE 也可以更改驱动器运行模式，但运动模块不支持此命令。

当运动总线不处于第 4 阶段，查询 Opmode 将返回 -1。

语法

<axis>.OpMode = <value>

<lvalue> = <axis>.OpMode

域

AXIS

参数

<axis>: 任何有效的轴

<value>: 给定的允许的运行模式，Long，0、1、2

限制

仅在 CP4 中写入，仅在禁用状态下允许 Opmode 更改。

不允许属于已连接轴组的轴进行 OPMODE 更改

仅独立属性

示例

A1.OpMode = 2

i = A1.OpMode

另请参见

TORQUECOMMAND, TORQUEFEEDBACK

ORITYPE

说明

这个指令是用于在 SMOVE 指令移动期间控制其方向。

0 -- 标准模式，表示 SMOVE 可精准地达到每个点的方向。

1-- 恒定模式，意味着 SMOVE 将保持与起始点相同的方向。

2 -- 最优模式, 意味着 SMOVE 将计算出最优的方向轨迹。给定点的方向不被保证。

如果没有指定 ORITYPE 值，初始值被设置为零。

缩写形式

<ROBOT>.oritype

语法

<ROBOT>.oritype=<numeric expression>

域

ROBOT

参数

<expression>: INT, range from 0 to 2.

限制

读/写, 模态/节点

示例

SCARA.ORITYPE = 0

另请参见

SMOVE

PASSINDEX

说明

The PASSINDEX is to control how many points in the given point array are to be passed for SMOVE command.

这个属性用于控制给定的点的数组里有多少个点被传递给 SMOVE 命令。

语法

PASSINDEX = INDEX1, INDEX2

域

只适用于 SMOVE 命令

参数

INDEX1: INT, start index of given point array

INDEX2: INT, end index of given point array

限制

只写

只用于节点

示例

SMOVE Array PASSINDEX=1,50

另请参见

SMOVE

PAYLOADINERTIA

说明

负载绕最后一个轴的惯量。

通过动态模型计算轴力矩的过程中将考虑该值。

语法

<ROBOT>.PayloadInertia = <expression>

域

ROBOT

参数

< ROBOT >: 有效的机器人

< expression >: Double

限制

读/写

仅独立属性

示例

SCARA.PayloadInertia = 0.05

? SCARA.PayloadInertia

另请参见

PAYLOADLX, PAYLOADLY, PAYLOADMASS

PAYLOADLX

说明

机器人在工具坐标系 x 轴方向的负载质心 (C.M) 距离。

通过动态模型计算轴力矩的过程中将考虑该值。

语法

< ROBOT >.PayloadLx = <expression>

域

ROBOT

参数

< ROBOT >: 有效的机器人

< expression >: Double

限制

读/写

仅独立属性

示例

SCARA.PAYLOADLX = 0.0

? SCARA.PAYLOADLX

另请参见

PAYLOADLY, PAYLOADINERTIA, PAYLOADMASS

PAYLOADLY

说明

机器人在工具坐标系 y 轴方向的负载质心 (C.M) 距离。

通过动态模型计算轴力矩的过程中将考虑该值。

语法

< ROBOT >.PayloadLy = <expression>

域

ROBOT

参数

< ROBOT >: 有效的机器人

< expression >: Double

限制

读/写

仅独立属性

示例

SCARA.PAYLOADLY = 0.0

? SCARA.PAYLOADLY

另请参见

PAYLOADLX, PAYLOADINERTIA, PAYLOADMASS

PAYLOADMASS

说明

运动元素的负载质量。

通过动态模型计算轴力矩的过程中将考虑该值。

语法

< ROBOT >.PayloadMass = <expression>

域

ROBOT

参数

< ROBOT >: 有效的机器人

< expression >: Double

限制

读/写

仅独立属性

示例

SCARA.PAYLOADMASS = 3

? SCARA.PAYLOADMASS

另请参见

PAYLOADLX, PAYLOADLY, PAYLOADINERTIA

PAYLOADMAX

说明

运动元素负载的最大允许质量。

如果 paylodMass 属性设置为更大的值，则会引发错误。

语法

< ROBOT >.PayloadMax = <expression>

域

ROBOT

参数

< ROBOT >: 有效的机器人

< expression >: Double

限制

读/写

仅独立属性

示例

SCARA. PAYLOADMAX = 10

? SCARA. PAYLOADMAX

另请参见

PAYLOADLX, PAYLOADLY, PAYLOADMASS

PEAKTORQUE

说明

返回指定轴的峰值扭矩。单位为最大扭矩的百分比。

语法

?< ELEMENT >.PeakTorque

域

ELEMENT

参数

< ELEMENT>: 有效的运动元素

限制

读

仅独立属性

示例

?J1.PeakTorque

另请参见

RESETPEAKVALUE, SETVELERRTHRESHOLD, COLLISIONDETECT, DRIVETORQUELIMIT, PEAKVELERR, VELERRTHRESHOLD,

PEAKVELERR

说明

返回指定轴的速度误差值。单位为最大误差值的百分比。

语法

?< ELEMENT >.PeakVelErr

域

ELEMENT

参数

< ELEMENT>: 有效的运动元素

限制

读

仅独立属性

示例

?J1. PeakVelErr

另请参见

RESETPEAKVALUE, SETVELERRTHRESHOLD, COLLISIONDETECT, DRIVETORQUELIMIT, PEAKTORQUE, VELERRTHRESHOLD, SETDRIVETORQUELIMIT, SETDEFVELERRTHRESHOLD

PING

说明

PING 命令用于测试远程主机是否可访问。如果主机不可访问或未应答，则返回运行时错误。

语法

?Ping (<ip address>)

参数

<ip address> : String, IP 地址

返回值

<returned value>:

-1 = 远程主机不可访问

0 = 正常

示例

? Ping (”212.25.84.109”)

另请参见

ACCEPT, CLOSE, CONNECT, OPENSOCKET, IPADDRESSMASK

PLSLIST

说明

返回系统中定义的 PLS 名称列表，该信息显示为以下格式：

PlsName=<pls>, AxisName=<axis>, Output=<pls output>

如果使用通配符，则查询返回相应的现有 PLS 数据。

语法

?PlsList {<pls>}

参数

< pls >: 指定的 pls

返回值

<return value>: String

返回系统中定义的 PLS 名称列表，该信息显示为以下格式：

PlsName=<pls>, AxisName=<axis>, Output=<pls output>

限制

只读

示例

?Plslist

?plslist PT*

另请参见

AXISLIST, GROUPLIST, TASKLIST, VARLIST, VARLIST$

PLSOUTPUT

说明

此属性设置或查询与 PLS 关联的输出。

缩写形式

<pls>.Pout

语法

<pls>.PlsOutput = <digital output signal >

?<pls>.PlsOutput

域

PLS

参数

< pls >: 任何有效的 pls 变量

<system output>: 任何类型为数字输出的信号变量

限制

仅当禁用 PLS 时才能进行设置。

信号变量必须存在，且其类型必须为数字输出。

示例

PLS1.Pout = SIGNAL1

?PLS1.PlsOutput

‘returns “SIGNAL1”

另请参见

PLSSOURCE

PLSPOLARITY

说明

设置或查询与 PLS 关联的系统输出的初始极性。

PLS 的初始极性指定为 1 或 0。默认值为 0。输出状态在启用 PLS (PLSEnable) 时设置。

根据以下方案，输出值在 PLSenable 时刻通过轴位置确定：

如果 Pcmd < PLSPosition[1] → 输出 = 负 PLSPolarity

缩写形式

<pls>.Ppol

语法

<pls>.PlsPolarity = <polarity>

?<pls>.PlsPolarity

域

PLS

参数

< pls >: 任何有效的 pls 变量

< polarity >: Long，0 或 1

限制

仅当禁用 PLS 时才能进行设置。

示例

Pls1.PlsPolarity = 0

另请参见

PLSSOURCE

PLSPOSITION

说明

设置或查询各个 PLS 位置值。

初次定义 PLS 时，它没有任何关联位置。CreatePLSData 命令用于定义 PLS 位置的值。

可以定义无限数量的 PLS 位置，但数组必须单调递增。

PLS 位置值始终为绝对值（而不是增量）。PLS 位置以用户单位给定，因此受轴 Displacement 和 PositionFactor 属性影响。

PLS 位置单位取决于 PLSSource 定义的位置类型。例如，如果 PLSSource 类型为 PlsPercentage，则单位为百分比。如果 PLSSource 类型为 PlsTime，则单位为毫秒。

缩写形式

<pls>.Ppos

语法

<pls>.PlsPosition[<index>] = <PLS position>

域

PLS

参数

< pls >: 任何有效的 pls 变量

<index>: 1 到 PLS 数据结构的大小

<PLS position>: Double，± MaxDouble

限制

仅当禁用 PLS 时才能进行设置。

示例

PLS1.PlsPosition[1] = 2076.56

另请参见

PLSSOURCE

PLSRELATEDTO

说明

这是 PathLength、PathPercentage 和 PathTime 使用的标志，用于定义 PLS 位置的计算是相对于运动起始位置还是目标位置。

语法

<pls>.PlsRelatedTo = <value>

?<pls>.PlsRelatedTo

域

PLS

参数

< pls >: 任何有效的 pls 变量

<value >: Long，0 或 1

示例

PLS1.RelatedTo = 1

另请参见

PLSSOURCE

PLSSOURCE

说明

定义切换 PLS 输出的位置类型。

PLSSource 可以基于绝对位置（XYZ 或轴），也可以基于当前运动的相对位置。

覆盖运动元素的 PLSSource 值。

源	说明	绝对/相对
PathLength	路径上的距离（按照用户单位）	相对于起始/目标位置
PathPercentage	路径的百分比	相对于起始/目标位置
PathTime	持续时间（毫秒）	相对于起始/目标位置

语法

<pls>.PLSSOURCE= <source type>

域

PLS

参数

< pls >: 任何有效的 pls 变量

< source type >: 源类型

示例

PLS1.PLSSOURCE = scara.PATHLENGTH

另请参见

PLSOUTPUT, PLSPOLARITY, PLSPOSITION, PLSRELATEDTO

PLT_GET_INDEX_STATUS

说明

该指令用于获取指定托盘上当前存在的货物数量.

语法

PLT_GET_INDEX_STATUS (<PalletName>)

参数

<PalletName>: String, 任何有效的托盘

返回值

返回指定托盘上当前存在的货物数量.

<return value>: Long

示例

?PLT_GET_INDEX_STATUS ("PLT1")

另请参见

PLT_SET_INDEX_STATUS

PLT_MOVE_TO_ENTRY_POSITION

说明

此命令指示机器人移动到一个托盘的入口位置。如果机器人位于物件旁（处于抓取或放置位置），它将在去往入口位置的途中经过远离点位置（如果已定义并启用）。

语法

PLT_MOVE_TO_ENTRY_POSITION (<ROBOT>, <PalletName>)

参数

<ROBOT>: 任何有效的机器人

<PalletName>: String, 任何有效的托盘

示例

PLT_MOVE_TO_ENTRY_POSITION (SCARA,"PLT1")

另请参见

PLT_PICK_FROM_PALLET, PLT_PLACE_ON_PALLET, PLT_RETRACT_FROM_ITEM

PLT_PICK_FROM_PALLET

说明

此命令指示机器人移动到托盘中的下一个物件。机器人移动通过入口和靠近点（如果这两者已定义并激活）。

语法

PLT_PICK_FROM_PALLET(<ROBOT>, <PalletName>)

参数

<ROBOT>: 任何有效的机器人

<PalletName>: String, 任何有效的托盘

示例

PLT_PICK_FROM_PALLET(SCARA,"PLT1")

另请参见

PLT_MOVE_TO_ENTRY_POSITION, PLT_PLACE_ON_PALLET, PLT_RETRACT_FROM_ITEM

PLT_PLACE_ON_PALLET

说明

此命令指示（持拿物件的）机器人移动到托盘中的下一个可用位置。机器人移动通过入口和靠近点位置（如果这两者已定义并激活）。

语法

PLT_PLACE_ON_PALLET(<ROBOT>, <PalletName>)

参数

<ROBOT>: 任何有效的机器人

<PalletName>: String, 任何有效的托盘

示例

PLT_PICK_FROM_PALLET(SCARA,"PLT1")

另请参见

PLT_MOVE_TO_ENTRY_POSITION, PLT_PICK_FROM_PALLET, PLT_RETRACT_FROM_ITEM

PLT_RETRACT_FROM_ITEM

说明

此命令指示机器人移动到一个托盘中当前物件的远离点位置。

语法

PLT_RETRACT_FROM_ITEM (<ROBOT>, <PalletName>)

参数

<ROBOT>: 任何有效的机器人

<PalletName>: String, 任何有效的托盘

示例

PLT_RETRACT_FROM_ITEM (SCARA,"PLT1")

另请参见

PLT_MOVE_TO_ENTRY_POSITION, PLT_PICK_FROM_PALLET, PLT_PLACE_ON_PALLET

PLT_SET_INDEX_STATUS

说明

此命令定义托盘上的物件数量。此命令在启动应用时非常有用。

语法

PLT_SET_INDEX_STATUS (<PalletName>, <index>)

参数

<PalletName>: String, 任何有效的托盘

<index>: Long, 要设置的当前物件索引

示例

PLT_SET_INDEX_STATUS ("PLT1", 2)

另请参见

PLT_SET_INDEX_STATUS_EMPTY, PLT_SET_INDEX_STATUS_FULL, PLT_GET_INDEX_STATUS

PLT_SET_INDEX_STATUS_EMPTY

说明

此命令将托盘上的物件数量定义为 0。此命令在启动应用时非常有用。

语法

PLT_SET_INDEX_STATUS_EMPTY (<PalletName>)

参数

<PalletName>: String, 任何有效的托盘

示例

PLT_SET_INDEX_STATUS_EMPTY ("PLT1")

另请参见

PLT_SET_INDEX_STATUS, PLT_SET_INDEX_STATUS_FULL

PLT_SET_INDEX_STATUS_FULL

说明

此命令将托盘上的物件数量定义为满载。此命令在启动应用时非常有用。

语法

PLT_SET_INDEX_STATUS_FULL (<PalletName>)

参数

<PalletName>: String, 任何有效的托盘

示例

PLT_SET_INDEX_STATUS_FULL ("PLT1")

另请参见

PLT_SET_INDEX_STATUS, PLT_SET_INDEX_STATUS_EMPTY

POSITIONCOMMAND

说明

此属性返回运动控制器生成的位置命令。轴组中的轴处于跟随模式时，位置命令设置为位置反馈的值。驱动器被禁用时，或者使用 <axis>.FOLLOWINGMODE 属性显式地将轴设为跟随模式时，轴处于跟随模式。

注意

坐标为关节空间坐标，而不是电机坐标。这些值在乘以耦合矩阵后返回。

缩写形式

<element>.PCmd

语法

?<element>.PositionCommand

域

ELEMENT

参数

< element >: 有效的运动元素

返回值

返回运动控制器生成的位置命令

<return value>: Double 矢量

限制

只读

示例

轴

?A1.PCmd

机器人

?SCARA.PCmd

另请参见

ACCELERATIONCOMMAND, VELOCITYCOMMAND, POSITIONFEEDBACK, POSITIONFINAL, POSITIONTOGO, ACCELCMDCART, POSITIONCOMMANDHISTORY, START_JOINT

POSITIONCOMMANDHISTORY

说明

此属性返回运动控制器生成的位置命令的历史记录。它可返回位置历史记录缓冲区作为位置误差延迟值的函数。

<element>.PositionErrorDelay= <NUM>

<element>.PositionCommandHistory[1]→ PCMD(t-1)

<element>.PositionCommandHistory[<NUM>] → PCMD(t-<NUM>)

注意

查询 t=0 时的位置历史记录应返回与查询 PCMD 相同的值。仅当从事件或记录查询时此情况为真。从终端/用户任务查询时无法保证。

缩写形式

<element>.PCmdH

语法

?<element>.PositionCommandHistory[t]

域

ELEMENT

参数

< element >: 有效的运动元素

[t]: PositionCommandHistory 的索引

返回值

返回运动控制器生成的位置命令的历史记录

<return value>: Double

限制

只读

示例

轴

?A1.PCmdH

机器人

?SCARA.PCmdh

另请参见

POSITIONCOMMAND

POSITIONERROR

说明

此属性返回位置跟踪误差，即位置命令与位置反馈之间的差值。该计算会考虑 SERCOS 延迟，以从之前的两个周期发出的命令中减去位置反馈。它将在每个采样时间进行计算，并与最大位置误差 (PEMAX) 进行比较。

如果超出 PEMAX 的值，运动元素将立即停止。如果检测到错误时运动元素处于静止状态，则驱动器将被禁用。

驱动器正确调整时，跟踪误差最小。在 CDHD 驱动器中启用微插补时，将增加额外的伺服延迟。在这种情况下，通过系统计算的位置误差稍大，但运动更平滑。

该计算会考虑位置误差延迟，以从之前多个周期发出的命令中减去位置反馈，如 POSITIONERRORDELAY 属性中所定义。

对于机器人 - 返回 SetPoint 各位置部分与 Here 之间的实际距离：

$SQRT((SetPoint\{1\}-Here\{1\})^2 + (SetPoint\{2\}-Here\{2\})^2 + (SetPoint\{3\}-Here\{3\})^2)$

缩写形式

<element>.PE

语法

?<element>.PositionError

域

ELEMENT

参数

< element >: 有效的运动元素

返回值

返回 SetPoint 各位置部分与 Here 之间的实际距离。

<return value>: Double

限制

只读

示例

轴

?A1.PE

机器人

?SCARA.PE

另请参见

POSITIONERRORSETTLE

POSITIONERRORSETTLE

说明

此属性定义运动元素被定义为就位的范围。运动控制器完成后，位置误差（目标位置 - 实际位置）的绝对值与 PESETTLE 值进行比较。结果在 TIMESETTLE 给定的时间内小于或等于该值时，将设置 ISSETTLED 标志。

缩写形式

<element>.PESettle

语法

<element>.PositionErrorSettle = < expression >

?<element>.PositionErrorSettle

域

ELEMENT

参数

< element >: 有效的运动元素

< expression >: 运动元素被定义为就位的范围，Double，0 到 MaxDouble

限制

要在任务中设置该值，必须将运动元素连接到该任务（使用 ATTACH 命令）。

示例

轴

A1.PESettle =0.0144

机器人

SCARA.PESettle =0.0144

另请参见

ATTACH, ISSETTLED, POSITIONERROR, STARTTYPE

POSITIONFEEDBACK

说明

此属性返回运动元素的实际位置。该值作为各个轴位置的矢量返回。

注意

坐标为关节空间坐标，而不是电机坐标。这些值在乘以耦合矩阵后返回。

缩写形式

<element>.PFb

语法

?<element>.PositionFeedback

域

ELEMENT

参数

< element >: 有效的运动元素

返回值

返回运动元素的实际位置

<return value>: Double 矢量

限制

只读

示例

轴

?A1.pfb

机器人

?SCARA.pfb

另请参见

POSITIONCOMMAND

POSITIONFINAL

说明

此属性返回运动元素目标位置。

对于轴 - 目标位置在 MOVE 或 CIRCLE 运动命令内设置。

对于轴组 - 这是一个维度等于轴组中轴数的矢量。

缩写形式

<element>.PFinal

语法

?<element>.PositionFinal

域

ELEMENT

参数

< element >: 有效的运动元素

返回值

返回运动元素目标位置

<return value>: Double 矢量

限制

目标位置在每条运动命令中指定。

作为一个单独的属性，它是只读的。

示例

轴

?A1.Pfinal

机器人

?SCARA.Pfinal

另请参见

POSITIONCOMMAND, POSITIONTOGO, CIRCLE, MOVE

POSITIONTOGO

说明

此属性返回命令要求的运动元素位置与最终位置之间的距离。

缩写形式

<axis>.PToGo

语法

?<axis >.PositionToGo

域

AXIS

参数

< axis >: 有效运动轴

返回值

返回命令要求的运动元素位置与最终位置之间的距离

<return value>: Double

限制

只读

示例

?A1.PositionToGo

另请参见

POSITIONCOMMAND, POSITIONFINAL

POWER_OFF

说明

此命令将轴组设置为禁用。Power_off 命令应在运动命令之后使用。

语法

Power_off()

示例

在任务中：

```
Program
    Power_on()
    Attach scara
        Move scara {0,0,0,0}
        Waitformotion scara
    Detach scara
    Power_off()
End program
```

另请参见

POWER_ON

POWER_ON

说明

此命令将轴组设置为激活。Power_on 命令应在运动命令之前使用。

语法

Power_on()

示例

在任务中：

```
Program
    Power_on()
    Attach scara
        Move scara {0,0,0,0}
        Waitformotion scara
    Detach scara
    Power_off()
End program
```

另请参见

POWER_OFF

PRINT

说明

打印多个字符串和表达式，用逗号或分号分隔。

分号将两个表达式放在一起，而逗号则在两者之间放一个制表符。表达式末尾的分号将取消末尾默认打印的 CR-LF 符号，从而联接多条 print 语句。

该消息打印到 ControlStudio 的输出窗口，最后一条 print 语句打印时末尾不带分号。只需查询变量即可“监控”变量。

ControlStudio Watch 窗口管理对指定变量的查询。使用 Print "" 可打印一个空行。

您可以通过用分号终止 PRINT 在行尾按回车。

DetectOverflow 参数可处理不适合内部消息队列（64 条消息）的打印输出。

值为零时，这些打印输出将被静默删除，且任务不会延迟。不过，如果 DetectOverflow 为“1”且打印队列中没有可用空间，则任务将因运行时错误而停止。不使用时，此可选标志默认为零。

缩写形式

?

语法

PRINT <Expression> {, <Expression>}

PRINT <Expression>{ <Expression>}

PRINT <Expression> {; <Expression>}

参数

< Expression>: Long、Double、String、Joint、Location

限制

仅当任务暂停时，才能从任务外部（例如，从终端）查询任务内定义的变量。在任务内，只有在声明局部变量的代码段中的任务处于暂停状态时，局部变量才可用。

示例

Print "Hello,";

Print "world."

打印以下内容：

Hello,world.

或

Print "Hello," "world."

打印以下内容：

Hello,world.

或

Print "Hello," , “world.”

打印以下内容：

Hello, world.

或

Print "Hello,"

Print "world."

打印以下内容：

Hello,

world.

或

Print "Hello," ; "world."

打印以下内容：

Hello,world.

另请参见

PRINTUSING, PRINTTOBUFF, PRINTUSINGTOBUFF, PRINTPOINT, PRINT_HASH, PRINTPOINTUSING

PRINT_HASH

说明

将多个字符串和表达式打印到串行端口，用逗号或分号分隔。

分号将两个表达式放在一起，而逗号则在两者之间放一个制表符。

语法

PRINT #<DeviceHandle> <Expression>{, <Expression>}

PRINT #<DeviceHandle> <Expression>{ <Expression>}

PRINT #<DeviceHandle> <Expression>{; <Expression>}

参数

<DeviceHandle>: 文件句柄，Long，1 到 255

< Expression >: Long、Double、String、Joint 或 Location

限制

只读

示例

Print #1, Nm$, Dept$, Level, Age

另请参见

PRINT, PRINTUSING, CLOSE, INPUT$, LOC

PRINTPOINT

说明

PrintPoint 仅打印声明为笛卡尔空间坐标的变量。

此命令的其余属性与 PRINT 命令类似。

语法

PRINTPOINT < Expression >

参数

< Expression>: Location

限制

仅当任务暂停时，才能从任务外部（例如，从终端）查询任务内定义的变量。

在任务内，只有在声明局部变量的代码段中的任务处于暂停状态时，局部变量才可用。

示例

common shared PNT as Location of XYZR

PNT = #{0,0,0,0}

PrintPoint PNT

打印以下内容： #{0 , 0 , 0 , 0;}

common shared PNT as Joint of XYZR

PNT = {0,0,0,0}

PrintPoint PNT

打印以下内容：{0 , 0 , 0 , 0}

另请参见

PRINTPOINTUSING, PRINT, PRINTUSING

PRINTPOINTUSING

说明

使用指定的格式字符串打印格式化的点类型数值变量。格式字符串包含要打印的文本，以及用于设置数值表达式格式的字符。

以下字符可用于设置数值表达式格式：

数字位置

. 小数点位置

+ 打印表达式的符号

^^^^ 以指数格式打印

缩写形式

PrintPointU

语法

PrintPointUsing <format_string>;[<expression>];{[<expression>]}*

参数

<format_string>: String

<expression>: 点（Joint 或 Location）

限制

只允许单一格式设置文本（包含文本和格式设置字符），后跟要打印的表达式。

示例

```
common shared PNT as Location of XYZR
PNT = #{0,0,0,0}
PrintPointUsing "#.###"; PNT
```

打印以下内容：#{0.000 , 0.000 , 0.000 , 0.000;}

另请参见

PRINT, PRINTPOINT, CLOSE, INPUT$, LOC

PRINTTOBUFF

说明

多个表达式将打印到串行端口中的缓冲区，用逗号或分号分隔。

分号将两个表达式放在一起，而逗号则在两者之间放一个制表符。

分配给可选 Send 参数的值确定命令结束时是将缓冲区内容发送到串行端口 (Send = True) 还是存储在缓冲区中 (Send = False)。

如果 Send 赋值已取消，则 Send 默认为 False。

语法

PRINTTOBUFF #<DeviceHandle>, [Expression] {, ;[Expression]} {SEND = TRUE or FALSE}

参数

<DeviceHandle>: 文件句柄，Long，1 到 255

< Expression >: Long、Double、String、Joint 或 Location

限制

只读

示例

PrintToBuff #1, Nm$, Dept$, Level, Age

PrintToBuff #1, Nm$, Dept$, Level, Age Send = True

另请参见

PRINT, CLOSE

PRINTUSING

说明

使用指定的格式字符串打印格式化的数值变量。格式字符串包含要打印的文本，以及用于设置数值表达式格式的字符。

以下字符可用于设置数值表达式格式：

数字位置

. 小数点位置

+ 打印表达式的符号

^^^^ 以指数格式打印

只允许单一格式设置字符串（包含文本和格式设置字符），后跟要打印的表达式。

缩写形式

PrintU

语法

PrintUsing <format_string>;[<expression>];{[<expression>]}*

参数

<format_string>: 输出格式，String

<expression>: Long、Double 或 String

限制

只能显示小数点后的 15 位。语句，

printu "#.################";My_Var

仅打印 15 位小数，而不是格式字符串指定的 16 位。仅显示小数点前的 16 位。语句：

printu "#.#";1e15

打印为

100000000000000.0

而

printu "#.#";1e16

打印为

1.0e16

示例

通过使用“#”符号和小数点，指定要打印的最少字符数。

B = 33.344

PrintU "B is ##.##";B

‘prints B IS 33.34

PrintU "B is ##.####";B

‘prints B IS 33.3440

PrintU "B is ##.##^^^^";B

‘prints B IS 3.33e+01

(you must use exactly four ^s for this format to work)

另一个例子：

PrintU "The number is # ";j1,j2

打印“The number is 1 The number is 2 ”

指定要打印的位数，以表格形式打印数据，因为打印的位数不随数字值而变化：

PrintU "Keep numbers to a fixed length with PrintUsing: ######"; 100

打印“Keep numbers to a fixed length with PrintUsing: 100”

另请参见

CLOSE, INPUT$, LOC, PRINT, PRINT_HASH, PRINTPOINT, PRINTUSING$, PRINTUSING_HASH-SIGN, PRINTUSINGTOBUFF

PRINTUSING$

说明

使用指定的格式 <string> 将格式化的数值变量打印到字符串型 (String) 变量。

格式字符串包含要打印的文本，以及用于设置数值表达式格式的字符。

缩写形式

PrintU$

语法

PrintUsing$<string_name>{[ ]}, <format_string>;
[<expression>];{[<expression>]}*

PrintUsing$<structure_name>-><string_name>{[ ]}, <format_string>;
*[<expression>];{[<expression>]}**

参数

< string_name >: String。字符串也可以是结构体字段。

<format_string>: String

<expression>: Long 或 Double

限制

只能显示小数点后的 15 位。语句：

printu “#.##################”;My_Var

示例

通过使用“#”符号和小数点，指定要打印的最少字符数。

Common shared StSCARA as string

Common Shared B as Double

B = 33.344

PrintU$ StSCARA , “B is ##.###”;B

?StSCARA

B IS 33.34

另请参见

CLOSE, INPUT$, LOC, PRINTUSING

PRINTUSING_HASH-SIGN

说明

PRINTUSING #

使用指定的格式 <string> 将格式化的数值变量打印到串行端口。格式字符串包含要打印的文本，以及用于设置数值表达式格式的字符。只允许单一格式设置字符串（包含文本和格式设置字符），后跟要打印的表达式。

缩写形式

PrintU #

语法

PrintUsing #<DeviceHandle>, <format_string>;
[<expression>];{[<expression>]}*

参数

<DeviceHandle>: 文件句柄，Long，1 到 255

<format_string>: String

<expression>: Long 或 Double

限制

只能显示小数点后的 15 位。语句，

printu “#.################”;My_Var,

示例

通过使用“#”符号和小数点，指定要打印的最少字符数。 B = 33.344 PrintU #1 “B is ##.##”;B ‘prints B is 33.34

PrintU #1 “B is ##.####”;B ‘prints B IS 33.3440

PrintU #1 “B is ##.##^^^^”;B ‘prints B IS 3.33e+01

另请参见

CLOSE, LOC, INPUT$, PRINTUSING

PRINTUSINGTOBUFF

说明

PRINTUSINGTOBUFF #

使用指定的模板字符串 <format_string> 将格式化的数值变量打印到串行端口的缓冲区。

格式字符串包含要打印的文本，和用于设置数值表达式格式的字符。

只允许单一格式设置字符串（包含文本和格式设置字符），后跟要打印的表达式。

分配给 Send 参数的值确定命令结束时是将缓冲区内容发送到串行端口 (Send = True) 还是存储在缓冲区中 (Send = False)。

如果 Send 赋值已取消，则 Send 默认为 False。

缩写形式

PrintUToBuff #

语法

PrintUsing #< DeviceHandle>, <format_string>;
[<expression>];{[<expression>]}* {Send = True | False}

参数

<DeviceHandle>: 文件句柄，Long，1 到 255

< Expression >: Long、Double、String、Joint 或 Location

<format_string>: String

限制

只能显示小数点后的 15 位。语句，printu “#.################”;My_Var,

示例

PrintUToBuff #1 “B is ##.##”;B

另请参见

CLOSE, PRINTUSING, PRINT

PROCEED

说明

此命令用于继续执行被先前的 STOP 命令停止的运动。

因此，当 STOP 命令在与移动命令相同的任务中出现时，不需要 PROCEED 命令；但当 STOP 命令由除移动轴或轴组的任务以外的任务发出时，则需要 PROCEED 命令。<proceed type> 可以是：

1 (Continue)

2 (NextMotion)

3 (ClearMotion)

语法

Proceed <group | axis> {ProceedType=<proceed type>}

参数

<robot | axis>：任何有效的机器人或轴

<proceed type>: Long，1，2，3

限制

如果未指定 <proceed type>，则使用 <proceed type> 的永久值，由 PROCEEDTYPE 属性指定。

示例

Proceed SCARA ProceedType = NextMotion

另请参见

PROCEEDTYPE, STOP

PROCEEDTYPE

说明

此属性定义 PROCEED 命令要进行的操作。它可以设置为永久值（独立属性），也可以在 PROCEED 命令内使用（非独立属性），以暂时覆盖永久值。

1 (CONTINUE) - 继续已停止的运动。如果缓冲区中有两个运动，它会执行两者（一个已执行，另一个在发出 STOP 命令时挂起）。

2 (NEXTMOTION) - 中止当前运动并执行运动缓冲区中的下一条运动命令。（在 STOP 期间执行的运动被遗忘，而执行另一个挂起的运动。）

3 (CLEARMOTION) - 中止运动缓冲区中的所有命令。实际上从完整的历史记录中清除运动缓冲区

4 (INTERRUPTED) - 仅执行中断的运动，不执行挂起的运动（注意：程序将不会继续，要释放等待的程序，您需要发出 PENDING 选项）

5 (PENDING) - 执行挂起的运动并释放等待的任务。（注意：仅当执行 INTERRUPTED 选项后才能发出。）

语法

<element>.ProceedType = <expression>

? <element>.ProceedType

域

ELEMENT

参数

< element >: 有效的运动元素

< expression >: 继续模式，Long，1 到 5

示例

轴

A1.ProceedType = Continue

?A1.ProceedType

Proceed A1 ProceedType=Next

机器人

SCARA.ProceedType = Continue

?SCARA.ProceedType

Proceed SCARA ProceedType=Next

另请参见

PROCEED

PROGRAM_..._END_PROGRAM

说明

PROGRAM…END PROGRAM 关键字用于分隔任务中代码的主要部分。

使用 DIM SHARED 定义的任务变量必须出现在 Program 关键字之前。局部变量（单独使用 DIM 定义）出现在 Program 关键字之后，但在代码之前。子程序必须在 End Program 关键字之后出现。

程序加载到内存中（使用 LOAD 命令）后，直到已执行 STARTTASK 命令后，该程序才开始执行。

通过将 Continue 关键字添加到 Program 关键字，程序会在其加载后自动执行。

任务到达 End Program 后，它将停止运行，但仍加载到内存中。End Program 可替换为 Terminate Program。

在这种情况下，任务完成后将从内存自动卸载。

语法

```
Program
<code to execute>
End Program
```

示例

```
Dim Shared vaSCARA as Long
Program
    Dim I As Long
    for I = 1 to 10
        vaSCARA = vaSCARA + 1
    Next I
End Program
```

另请参见

SUB_..._END_SUB, FUNCTION_..._END_FUNCTION, WHILE_..._END_WHILE, FOR_..._NEXT, TRY_..._END_TRY, DO_..._LOOP, WITH, WITHGLOBAL, REM, IF_..._THEN_..._ELSE, SELECT_..._CASE, CALL

PULSE

说明

PULSE 指令用于输出一个脉冲波信号，指令中只可以使用数字输出类型的信号变量，你可以设置从 100 到 30000 毫秒脉冲宽度。

注意

该指令是非阻塞式的，最多可以同时有 8 个脉冲信号操作同时执行，因此，您需要注意避免在已经存在 8 个脉冲波时，继续使用 PULSE 指令。

语法

PULSE <digital io signal> PVALUE = <pulse value> PWIDTH = <pulse width>

参数

<digital io signal>: 信号，用户已经创建的数字输出信号

<pulse value>: Long, On/ Off (1 / 0), 设置脉冲波的类型

<pulse width>: Long, 100 到 30000 毫秒，设置脉冲波的宽度

示例

PULSE signal PVALUE = on PWIDTH = 500

另请参见

GETIO , SETIO

RECORD

说明

此命令用于启用（而不是触发）数据记录。

数据被记录到指定的文件中，然后可以检索该文件进行查看。记录文件的扩展名必须始终为 REC。记录的数据文件存储在 RAM 驱动器中，因此断电后文件会丢失。

注意

数据是实时记录的，因此复杂的表达式会增加系统负载。由于内存分配过程，记录包含字符串、点或结构体的表达式可能会导致重叠错误。RECORD 仅定义要进行记录的数据。

实际的记录过程通过 RECORDON 触发。

将点数设置为负值会导致激活连续记录，与将 RingBuffer 设置为 ON 相同。

语法

RECORD <record file name> <number of points> {Gap = <record gap>} {RingBuffer = <ON|OFF>}RecData = <expression>{, <expression>}

参数

<record file name>: 文件规格，包括文件名和扩展名。任何文件名扩展名都必须为 REC

<number of points>: Long，采样点数

<ring buffer>: 1/0

1 = ON

0 = OFF

<record gap>: Long，采样周期，单位为毫秒，1 到 64，指定两次采样之间的 EtherCAT 周期数

<expression>: Long，Double

限制

记录文件存储在 RAM 驱动器上，因此有大小限制。如果 RAM 驱动器上没有可用的空间，则停止记录。RECORDCLOSE 命令必须在发出第二条 RECORD 命令之前发出。

局部变量无法进行记录。

示例

Record Motion.rec 1000 Gap = 1RecData = a1.pcmd, a1.pfb, a2.pcmd, MyVar

RecordOn

RecordOff

RecordClose

另请参见

RECORD$, RECORDCLOSE, RECORDING, RECORDOFF, RECORDON

RECORD$

说明

此命令用于启用（而不是触发）数据记录。

数据被记录到指定的文件中，然后可以检索该文件进行查看。记录文件的扩展名必须始终为 REC。记录的数据文件存储在 RAM 驱动器中，因此断电后文件会丢失。

注意

数据是实时记录的，因此复杂的表达式会增加系统负载。由于内存分配过程，记录包含字符串、点或结构体的表达式可能会导致重叠错误。RECORD 仅定义要进行记录的数据。

实际的记录过程通过 RECORDON 触发。

语法

RECORD$ <record file name> <number of points> {Gap = <record gap>} {RingBuffer = <ON|OFF>}RecData = <expression>{, <expression>}

参数

<record file name>: String，包括文件名和扩展名。

<number of points>: Long，采样点数

<ring buffer>: 1/0

1 = ON

0 = OFF

<record gap>: Long，采样周期，单位为毫秒，1 到 64，指定两次采样之间的 EtherCAT 周期数

<expression>: Long，Double

限制

记录文件存储在 RAM 驱动器上，因此有大小限制。如果 RAM 驱动器上没有可用的空间，则停止记录。RECORDCLOSE 命令必须在发出第二条 RECORD 命令之前发出。

局部变量无法被记录。

示例

Record$ “Motion.rec” 1000 Gap = 1RecData = a1.pcmd, a1.pfb, a2.pcmd, MyVar

Dim shared RecFileString as string = “Motion.rec”

Record$ RecFileString 1000 Gap = 1 RecData =a1.pcmd, a1.pfb, a2.pcmd, MyVar

RecordOn

RecordClose

Record$ RecFileString 1000 RingBuffer=On RecData = a1.pcmd, a1.pfb, a2.pcmd, MyVar

RecordOn

RecordClose

另请参见

RECORD, RECORDCLOSE, RECORDING, RECORDOFF, RECORDON

RECORDCLOSE

说明

此命令关闭记录数据文件并使其能够被检索。

您必须在发出下一条 RECORD 命令之前发出此命令。

语法

RecordClose

限制

只写

示例

RecordClose

另请参见

RECORD, RECORDING, RECORDOFF, RECORDON, RECORD$

RECORDING

说明

返回数据记录器的状态。将返回以下值：

0 - 记录器未定义且未激活。

1 - 记录器已定义，但未激活。（需要 RECORDON 命令来激活记录器。）

2 - 正在进行记录（给出 RECORDON 后的数据采集过程。）

3 - 记录已暂停。（RECORDOFF 命令已发出。）

4 - 数据采集自动完成。记录器仍在定义中。（需要 RECORDCLOSE 命令以刷新磁盘数据。）

语法

?Recording

返回值

返回数据记录器的状态

<return value>: Long，0 到 4

限制

只读

示例

?Recording

另请参见

RECORD, RECORDCLOSE, RECORDOFF, RECORDON, RECORD$

RECORDOFF

说明

使用此命令或如果已记录指定数量的点，则记录操作停止。

语法

RecordOff

限制

只写

示例

RecordOff

另请参见

RECORD, RECORDCLOSE, RECORDING, RECORDON, RECORD$

RECORDON

说明

此命令开始记录过程。

语法

RecordOn

限制

只写。要记录的数据必须使用 RECORD 命令来定义。

示例

RecordOn

另请参见

RECORD, RECORDCLOSE, RECORDING, RECORDOFF, RECORD$

RECOVERPATH

说明

这条命令用来恢复 STOPPATH 所存储的运动指令。存储的运动指令将在当前运动指令完成后开始执行。

RECOVERPATH 将从 STOPPATH 存储的当前运动指令开始运行。

这条命令可以与 STOPPATH 命令一起使用

提示

RECOVERPATH 只能在 STOPPATH 执行完毕后使用。

RECOVERPATH 应在达到 DEPPOINT 后使用，并具有相同的手系配置。

缩写形式

RecoverPath

REM

说明

注释可以作为单独的行或在代码行的末尾自由插入到代码中。

注释以 REM 关键字或撇号 (’) 开头，并以行尾结束。

缩写形式

'

语法

REM <comments>

参数

< comments >: String，注释。

限制

一行的最大长度为 16,384 个字符。

示例

```
REM This is a comment line
' This is also a comment line
Move a1 156.7rem Move axis a1 to position 156.7
Move a2 0 ' Comment with an apostrophe
```

另请参见

PROGRAM ... END_PROGRAM

RESETPEAKVALUE

说明

清空所有轴的峰值扭矩值和峰值速度误差值。

语法

RESETPEAKVALUE()

示例

RESETPEAKVALUE()

另请参见

RESETPEAKVALUE, COLLISIONDETECT, DRIVETORQUELIMIT, PEAKTORQUE, PEAKVELERR, VELERRTHRESHOLD, SETDRIVETORQUELIMIT, SETDEFVELERRTHRESHOLD

RIGHT$

说明

RIGHT$ 返回字符串右侧指定数量的字符。

语法

RIGHT$(<string>,<expression>)

参数

<string>: String

<expression>: Long，0 到 <string> 的最大长度

返回值

返回字符串右侧指定数量的字符

<return value>: String

限制

只读

示例

Test=”This string is too short”

PRINT RIGHT$(Test,4)

打印以下内容：hort

另请参见

LEFT$, MID$

ROUND

说明

将参数四舍五入为最接近的偶数整数，因此随机样本中的累积误差永远不会超过 0.5。例如：

2.5 -> 3.0

1.5 -> 2.0

1.49 -> 1.0

语法

Round(<expression>)

参数

<expression>: Double，± MaxDouble

返回值

<return value>: Double

限制

只读

示例

?Round(4.5)

打印以下内容：5

VaSCARA = Round(Var2)

另请参见

INT

RTRIM$

说明

RTRIM$ 返回删除结尾所有空格后字符串的左侧部分。

语法

RTRIM$(<string>)

参数

<string>: String

返回值

返回删除结尾所有空格后字符串的左侧部分

<return value>: String

限制

只读

示例

PRINT RTRIM$(“a left-justified string ”);”next”

打印以下内容：a left-justified stringnext

另请参见

LTRIM$

SELECT_..._CASE

说明

此决策结构激活要执行的多个代码段之一，具体视 <SelectExpression> 的值而定。

在 CASE 命令块的第一行，指定要测试的变量或表达式。

指定变量或表达式后，列出变量可以采用的一个或多个值或值范围。

有四种指定 CASE 的方式：精确值、逻辑条件、范围、其他。

如果 <SelectExpression> 匹配与其中一个 CASE 子句相关联的条件，则执行该 CASE 子句后跟的语句，直到下一个 CASE 子句，或对于最后一个子句，执行到 END SELECT。控制随即传递给 END SELECT 后跟的语句。

如果有基于值的多条执行路径，则流程控制结构更倾向于 IF…THEN。

<SelectExpression> 使用 CASE <expression> 语法，根据需要与尽可能多的精确值进行比较。

<SelectExpression> 使用 CASE IS <relational-operator> <expression> 语法与其他表达式进行逻辑比较。可以使用六个关系运算符中的任何一个（>、>=、<、<=、=、<>）。

<SelectExpression> 可使用 CASE <expression> TO <expression> 语法与值范围进行比较。

如果使用 TO 关键字指示值范围，则必须首先输入较小的值。如果第一个表达式大于第二个表达式，则忽略 CASE 语句，且不会标记任何错误。

如果使用 CASE ELSE，其关联语句仅在 <SelectExpression> 不匹配任何其他 CASE 选择时执行。

语法

SELECT CASE <SelectExpression>

{CASE <expression>

{statement_list} }

{CASE IS <relational-operator> <expression>

{statement_list} }

{CASE <expression> TO <expression>

{statement_list} }

{CASE <expression> comma <expression>

{statement_list} }

{CASE ELSE

{statement_list} }

END SELECT

参数

< SelectExpression >: 变量或常量，Double、Long、String

<expression>: 变量或常量，Double、Long、String

<relational-operator>：>、>=、<、<=、=、<>。

示例

```
Dim I as Long
Select Case I
Case 0
Print "I = 0"
Case is >= 10
Print "I >= 10"
Case is < 0 'No requirement for statements after Case<0
Case 5 To 10
Print "I is between 5 and 10"
Case 2, 3, 5 'Added in Version 4.7.1
Print "I is 2, 3 or 5"
Case Else
Print "Any other I value"
End Select
```

另请参见

IF_..._THEN_..._ELSE, PROGRAM_..._END_PROGRAM

SELECTAXES

说明

选择将采用哪些目标坐标，以及不采用哪些坐标（对应位值设置为零）。该值是一个位域，用于定义运动中涉及的轴。（1 - 轴已采用）。需要取消整个系统中某些轴的运动时使用。

例如：

仅移动第一个轴 - 二进制 0001 (十进制 1)

仅移动第二个轴 - 二进制 0010 (十进制 2)

仅移动第三个轴 - 二进制 0100 (十进制 4)

仅移动第四个轴 - 二进制 1000 (十进制 8)

移动所有四个轴 - 二进制 1111 (十进制 15)

缩写形式

<ROBOT>.SAxes

语法

<ROBOT>.SelectAxes = <value>

域

ROBOT

参数

<ROBOT>: 任何有效的机器人

<value>: Long，1 到 15

限制

仅对轴组有效。

示例

SCARA.selectaxes = 1+2

Move SCARA {10,20,30,40} saxes = 1+2 ‘only first two axes move

Move scara {10,20,30,40} saxes = 1+2+4 ‘ only first three axes move

另请参见

MOVE

SELECTBASE

说明

此命令可用于在程序中切换基坐标系。所选基坐标系应已预定义。它将覆盖此程序中的全局基坐标系设置。

注意

这个指令可以被用在 attach 到 detach 之间，并且当坐标系被改变时，机器人的 TCP 坐标可能发生突变。因此，用户需要确认他设置的目标运动位置是正确的！

语法

SelectBase (<BaseFrame>)

参数

< BaseFrame >: String，任何有效的基坐标系

示例

SelectBase (“BASE1”)

另请参见

BASE, SELECTTOOL, SELECTUSERFRAME, SETWORKSPACE

SELECTTOOL

说明

此命令可用于在程序中切换工具坐标系。所选工具坐标系应已预定义。它将覆盖此程序中的全局工具坐标系设置。

注意

这个指令可以被用在 attach 到 detach 之间，并且当坐标系被改变时，机器人的 TCP 坐标可能发生突变。因此，用户需要确认他设置的目标运动位置是正确的！

语法

SelectTool (<ToolFrame>)

参数

< ToolFrame >: String，任何有效的工具坐标系

示例

SelectTool (“TOOL1”)

另请参见

TOOL, SELECTBASE, SELECTUSERFRAME, SETWORKSPACE

SELECTUSERFRAME

说明

此命令可用于在程序中切换用户坐标系。所选用户坐标系应已预定义。它将覆盖此程序中的全局工具坐标系设置。

注意

这个指令可以被用在 attach 到 detach 之间，并且当坐标系被改变时，机器人的 TCP 坐标可能发生突变。因此，用户需要确认他设置的目标运动位置是正确的！

语法

SelectUserFrame (<UserFrame>)

参数

< UserFrame >: String, 任何有效的用户坐标系

示例

SelectUserFrame (“US1”)

另请参见

USERFRAME, SELECTBASE, SELECTTOOL, SETWORKSPACE

SETDEFVELERRTHRESHOLD

描述

这条指令用于设置默认速度误差值，用于碰撞检测的触发条件。

注意

单位为最大速度误差的百分比。使用默认误差值，碰撞检测效果较差，建议使用测定值。

语法

SetDefVelErrThreshold ()

示例

SetDefVelErrThreshold ()

另请参见

RESETPEAKVALUE, SETVELERRTHRESHOLD, COLLISIONDETECT, DRIVETORQUELIMIT, PEAKTORQUE, PEAKVELERR, VELERRTHRESHOLD, SETDRIVETORQUELIMIT

SETDRIVETORQUELIMIT

描述

这条指令用于设置电机扭矩最大值。

注意

单位为电机最大转矩的百分比。

语法

SetDriveTorqueLimit (<LimitJ1>, < LimitJ2>,< LimitJ3>,< LimitJ4>)

参数

< LimitJ1>: double 类型, 范围[1-100]

< LimitJ2>: double 类型, 范围[1-100]

< LimitJ3>: double 类型, 范围[1-100]

< LimitJ4>: double 类型, 范围[1-100]

示例

SetDriveTorqueLimit (50,50,50,100)

另请参见

RESETPEAKVALUE, SETVELERRTHRESHOLD, COLLISIONDETECT, DRIVETORQUELIMIT, PEAKTORQUE, PEAKVELERR, VELERRTHRESHOLD, SETDRIVETORQUELIMIT

SETTOOL

描述

这条指令用于在程序中设置用户坐标系。选中的用户坐标系可以被重定义。可以重新覆盖全局用户坐标系的值。

注意

当用户坐标系改变了，TCP 可能突然被改变。因此你需确认你的运动目标点是正确的。

语法

SetTool (<ToolFrame>, <location>)

参数

< ToolFrame >: 字符串, 任何的工具坐标

< location >: 坐标点, 类型 xyzr

示例

SetTool (“ToolName”,Point)

另请参见

USERFRAME, SELECTBASE, SELECTTOOL, SETWORKSPACE, SETUSERFRAME

SETVELERRTHRESHOLD

描述

该指令用于设置 4 个轴的速度误差阈值，用于碰撞检测的判定条件。

注意

范围[1-100]

语法

SetVelErrThreshold (<ThresholdJ1>, < ThresholdJ2>,< ThresholdJ3>,< ThresholdJ4>)

参数

< ThresholdJ1>: double, 范围[1-100]

< ThresholdJ2>: double, 范围[1-100]

< ThresholdJ3>: double, 范围[1-100]

< ThresholdJ4>: double, 范围[1-100]

示例

SetVelErrThreshold (50,50,50,50)

另请参见

RESETPEAKVALUE, COLLISIONDETECT, DRIVETORQUELIMIT, PEAKTORQUE, PEAKVELERR, VELERRTHRESHOLD, SETDRIVETORQUELIMIT, SETDEFVELERRTHRESHOLD

SETUSERFRAME

说明

这条命令用于在程序中设置用户坐标系。被选中的用户坐标系可以被重定义。可覆盖全局用户坐标系的值。

注意

当用户坐标系改变了，TCP 将突然改变。因此，请再次确认运动目标点是正确的。

语法

SetUserFrame (<UserFrame>, <location>)

参数

< UserFrame >: 字符串，任何有效的坐标

< location >: 坐标点, 类型为 xyzr

示例

SetUserFrame (“UserName”,Point)

另请参见

USERFRAME, SELECTBASE, SELECTTOOL, SETWORKSPACE, SETTOOL

SEMAPHOREGIVE

说明

SEMGIVE 释放一个信号量。

如果之前已获取该信号量，则将其标记为空闲信号量，任何任务均可通过使用 SEMTAKE 命令获取该信号量。

如果信号量已为空闲，则其状态不会更改。

如果任务使用 SEMTAKE 获取了信号量，则可通过相同任务或通过系统中的任何任务（甚至从命令行）释放该信号量。

缩写形式

semgive

语法

semgive(<semaphore name>)

参数

< semaphore name >: 信号量变量

示例

Common shared globalVariable as long

在 task1 中：

semTake(SEM1)

globalVariable = globalVariable + 1

sleep(1000)

semGive(SEM1) ‘release the semaphore

在 task2 中：

Sleep(100)

While semTake(SEM1,1000) = 0 ‘wait for the release of the semaphore

Sleep(10)

End While

globalVariable = globalVariable + 1

semGive(SEM1) ‘release the semaphore

task1 执行 semGive(SEM1) 时，task2 开始运行 globalVariable

另请参见

COMMON_SHARED_..._AS_SEMAPHORE, SEMAPHORESTATE

SEMAPHORESTATE

说明

SEMSTATE 返回信号量状态：1（如果之前已获取信号量（繁忙））或 0（如果信号量已释放）。

适用于全局用户信号量、结构体元素信号量、按引用传递的信号量（函数\子块内）。

缩写形式

semstate

语法

?semstate(<semaphore name>)

参数

< semaphore name >: 信号量变量

返回值

返回信号量状态：1（如果之前已获取信号量（繁忙））或 0（如果信号量已释放）

<return value>: Long，0 或 1

限制

按引用传递的信号量（函数\子块内）。

示例

common shared sem1 as semaphore

?SemState(sem1)

打印以下内容：0

?SemTake(sem1)

?semstate(sem1)

打印以下内容：1

SemGive(sem1)

?SemState(sem1)

打印以下内容：0

另请参见

SEMAPHORETAKE, SEMAPHOREGIVE

SEMAPHORETAKE

说明

SEMTAKE 获取可用于多任务同步的信号量。

如果信号量为空闲，SEMTAKE 将返回 1，且该信号量被标记为“已获取”。

如果该信号量已被获取，则 SEMTAKE 返回 0 且信号量的状态不会更改。

SEMTAKE 可用于阻止同时访问系统中的共享资源或关键用户软件部分。

如果提供了超时参数，系统将在指定的时长内尝试获取信号量。

Semtake 可能在指定的超时之前返回，但未获取信号量，因此检查返回值至关重要。

缩写形式

semTake

语法

<lval> = SemTake(<semaphore name> { , <timeout>})

参数

< semaphore name >: 信号量变量

<timeout>: 0 到 5000

返回值

返回 1，且该信号量被标记为“已获取”

< lval >: Long

0 = 失败（信号量未获取）

1 = 成功（信号量已获取）

示例

```
While semTake(SEM1,1000) = 0 ‘wait for the release of the semaphore
’wait for resource
Sleep(10)
End While
```

或

```
semTake(SEM1) ‘take the semaphore
semTake(SEM1 , 100 )
```

另请参见

COMMON_SHARED_..._AS_SEMAPHORE, SEMAPHORESTATE

SETIO

说明

该命令用于为信号赋值。

语法

<signal_name > = <value >

域

IO

参数

<signal_name>: 信号名称

<value >: 浮点数

限制

设置的值需要是一个非负数

示例

Signal1 = 1 (signal1 is digital)

Signal2 = 2.3 (signal2 is analog)

另请参见

GETIO , PULSE

SETPOINT

说明

返回命令的机器人笛卡尔坐标。

此变量是从电机命令位置或根据当前插补类型计算的每个采样周期，等于 TOCART(PCMD)，是 HERE 的对应项。

语法

<point_variable> = <robot_name>.setpoint

? <robot_name>.setpoint

? <axis_name>.setpoint

域

ROBOT

参数

<robot_name>: 任何有效的机器人

<axis_name>: 任何有效的轴

返回值

返回命令的机器人笛卡尔坐标

<point_variable>: Double 或 Location

限制

独立属性，只读

示例

P1= Scara.SetPoint

? Scara.SetPoint

另请参见

START

SETARM

说明

设置 location 点的手系.

注意

当点的手系改变了,TCP 可能会突然改变。因此，需再次确认运动终点是否正确。

语法

SetArm(< Location >,< Arm >)

参数

<Location >: xyzr 系的坐标点

<Arm>: long 类型; 0/1/2

示例

SetArm(point,1)

另请参见

SETPOINT

SETWORKSPACE

描述

该命令用于设置是否启用指定的工作空间.

语法

SetWorkSpace (<WorkSpaceFrame>, <BoolValue>)

参数

< WorkSpaceFrame >: String, 任何有效的工作空间坐标系

<BoolValue>: Long, ON or OFF

示例

SetWorkSpace (“WS1”, ON)

另请参见

SELECTTOOL, SELECTUSERFRAME, SELECTBASE

SGN

说明

此函数返回表达式的符号。

如果 <expression> 为正，则 SGN 返回 +1

如果 <expression> 为零，则 SGN 返回 0。

如果 <expression> 为负，则 SGN 返回 -1

语法

SGN(<expression>)

参数

<expression>: Double，± Max Double

返回值

返回表达式的符号

<return value>: Long，-1，0，1

限制

只读

示例

?SGN(VaSCARA)

I = SGN(Var2)

SHL

说明

SHL 是位运算符。SHL（左移）将 <Number> 中的每一位向左移动 N 位，最左侧的 N 位将丢失。

语法

<Number> SHL <N>

参数

<Number>: Long，MaxLong 到 MaxLong

<N>: Long，0 到 31

返回值

<return value>: Long

示例

?16 SHL 3

打印以下内容：2

另请参见

SHR

SHR

说明

SHR 是位运算符。SHR（右移）将 <Number> 中的每一位向右移动 N 位，最右侧的 N 位将丢失。

语法

<Number> SHR <N>

参数

<Number>: Long，MaxLong 到 MaxLong

<N>: Long，0 到 31

返回值

<return value>: Long

示例

?16 SHR 3

打印以下内容：2

另请参见

SHL

SIMULATED

说明

此属性用于将轴运行模式定义为仿真或真实。在仿真模式下，运动命令不会发送到驱动器，因为轴未与物理驱动器关联。应用运动命令之前需要激活模拟轴（使用 <axis>.ENABLE=ON）。

0（真实）

1（仿真）

语法

<axis>.Simulated = <expression>

?<axis>.Simulated

域

AXIS

参数

<expression>: Long，0 或 1

限制

只能在运动总线通信阶段 0 期间进行设置。

示例

A1.Simulated=1

SIN

说明

此函数返回角度（弧度）的正弦值。

语法

SIN(<expression>)

参数

<expression>: 任何有效的表达式，Double

± 9.223372036853900e+18

返回值

返回角度（弧度）的正弦值

限制

只读

示例

?Sin (3.14159/2)

VaSCARA = Sin(3.14159)

另请参见

ASIN, COS, TAN

SIZE

说明

此函数返回输入字符串占用的字节数。

在 ASCII-8 字符串中，SIZE() 的值等于 LEN() 的值；而在 UTF-8 字符串中，SIZE() 的值可能明显大于 LEN() 的值。

语法

Size(<string>)

参数

<string>: String

返回值

返回输入字符串占用的字节数

<return value>: Long

限制

只读

示例

?SIZE (CHR$(0xC4)) ‘ The letter Ä

打印以下内容：1

?SIZE (UTF$(0xC4))

打印以下内容：2

另请参见

LEN

SLEEP

说明

此命令将任务延迟指定的时间段（毫秒）。

该延迟自上一命令结束时开始。

可通过 ContinueTask 命令终止休眠（任务被唤醒）。

注意

在两个混合运动之间使用 SLEEP 时，会影响混合的结果。如果 SLEEP 的时间比上一个动作的时间长，则混合将不起作用。

语法

Sleep <time>

参数

<time>: Long，1 到 MaxLong，单位为毫秒

限制

只写

示例

Sleep 100‘Delay task for 100 msec

另请参见

WAITFORMOTION

SMOVE

说明

SMOVE 命令执行一个样条路径运动，使机器人从给定的点移动到世界空间的目标点。

点阵可以是 LOCATION 或 JOINT 点阵。

运动的参数是： vtran, atran, dtran, vrot, arot, drot.

属性的语法是： <属性名>=<值>。

永久值在 SMOVE 的持续时间内被覆盖。给定的点数组的长度应该是 3 到 256。SMOVE 命令不支持与其他运动之间 blend 过渡，同时也不支持 until 功能。 WithPLS 命令可以用于 SMOVE。此外，SMOVE 支持通过索引触发的 pls。

请在距离和方向上均匀地分配点。应避免以下的点分布行为：

位置上的大跳跃。

两点之间方向变化很大。

两个点之间的转角很小或两个点是相反方向等。

如果不满足上述原则，SMOVE 运动可能会产生不可预测的路径，并且在运动过程中可能会发生速度跳跃。

请在低速下检查生成的路径，以确保它不会与环境发生碰撞。如果给定的点被修改，请再次验证生成的路径是否安全。

请注意，离最后一个点太近的点会被自动排除。

如果设置了不适当的速度，就会发生一些相关的错误，例如过大的扭矩。请降低 SMOVE 运动的给定速度和加速度。

语法

SMove<ROBOT> <Array> {PASSINDEX=INDEX1, INDEX2}{ORITYPE} {Optional Property}*

参数

<ROBOT>: 任何有效的机器人

<Array>: 在关节或位置上的点数组

{PASSINDEX=INDEX1, INDEX2}: 可选属性，用于设置阵列中的哪些点将被执行 SMOVE 运动。INDEX1 是数组的起始索引，INDEX2 是数组的结束索引。在 INDEX1 和 INDEX2 之间的点将被传递给 SMOVE 运动。如果没有指定 PASSINDEX，那么整个数组将被传递给 SMOVE 运动。

{ORITYPE}: 可选的属性，用于设置在 SMOVE 运动中方向的变化。如果没有指定该值，默认值被设置为标准。

{Optional Property}*: 可选的属性，用于设置移动速度和加速度怎样被各自的最大值（由用户设定）限制。

限制

只适用于机器人

存储点的数组长度可设为 3 到 256。

点应该是均匀分布的。

只能写。在一个任务中，机器人必须被绑定到那个任务。

示例

Attach SCARA

Smove array1 passindex = 1, 50 oritype = 0 vtran = 100

Smove array2 vtran = 100

另请参见

VELOCITYTRANS, JERK, STARTTYPE, WITHPLS, BLENDPERCENTAGE, CIRCLE, DEST, MOVE, ABSOLUTE, ACCELERATIONMAXROT, ACCELERATIONMAXTRANS, ACCELERATIONROT, ACCELERATIONTRANS, BLENDMETHOD, DECELERATIONROT, DECELERATIONTRANS, JERKMAXROT, JERKMAXTRANS, JERKROT, JERKTRANS, VELOCITYFINALROT, VELOCITYFINALTRANS, VELOCITYMAXROT, VELOCITYMAXTRANS, VELOCITYROT, VELOCITYROTVALUE, VELOCITYTRANSVALUE, XMAX, XMIN, YMAX, YMIN, ZMAX, ZMIN, VSMODE, PASSINDEX, ORITYPE

SPACE$

说明

SPACE$ 生成由指定数量的空格组成的字符串。

语法

SPACE$(<number>)

参数

<number>: Long

返回值

<return value>: String

限制

只读

示例

Test = “First” + Space$(10) + ”Next”

?Test

打印以下内容： First Next

另请参见

CHR$, STRING$, ASC

SQRT

说明

此函数返回数值表达式的平方根。

语法

SQRT(<expression>)

参数

<expression>: Double，大于或等于零

返回值

返回数值表达式的平方根

限制

只读

示例

?Sqrt (VaSCARA)

VaSCARA = Sqrt(Var2)

START

说明

检索运动的当前初始点（笛卡尔坐标）。

如果轴组已完成其运动，则该命令等于 SETPOINT。

如果轴组已通过 STOP 命令停止，将返回已停止（已取消）运动的初始点。

语法

<point_variable> = <element>.start

? < element >.start

域

ELEMENT

参数

< element >: 任何有效的运动元素

返回值

以笛卡尔坐标形式返回运动的当前初始点

<point_variable>: Double 或 Location

限制

仅独立属性，只读

示例

P1= Scara.Start

? Scara.Start

另请参见

SETPOINT

START_JOINT

说明

检索运动的当前初始点（关节空间坐标）。

如果轴组已完成其运动，则该命令等于 PCMD。

如果轴组已通过 STOP 命令停止，将返回已停止（已取消）运动的初始点。

语法

<point_variable> = < element >.start_joint

? < element >.start_joint

域

ELEMENT

参数

< element >: 任何有效的运动元素

返回值

以关节空间坐标形式返回运动的当前初始点

<point_variable>: Double 或 Joint

限制

仅独立属性，只读

示例

P1= Scara.Start_Joint

? Scara.Start_Joint

另请参见

POSITIONCOMMAND

STARTTYPE

说明

此属性确定下一条运动命令开始的时间点。若发出一条运动命令的同时上一条运动命令仍在执行，则与此属性相关。INPOSITION 选项是运动完成最严格的条件。

此属性在运动命令内使用以覆盖永久值。

1 - IMMEDIATE：从系统当前位置立即执行运动命令，而不是等待当前运动命令完成。当前运动将取消，且所有缓冲的运动都将延迟，直到 IMMEDIATE 命令执行。

2 - INPOSITION：设置 ISSETTLED 标志时执行运动命令。这表示系统已完成其之前的运动并到达所需位置。

3 - GENERATORCOMPLETED：运动控制器生成最后一个运动命令参考后，该运动命令即执行。

4 - SYNC：运动开始通过 SYNCSTART 命令同步。

5 - SUPERIMMEDIATE：与 IMMEDIATE 类似，但该命令的计算实时进行，而不是在后台进行。

INPOSITION 选项是运动完成最严格的条件。此属性在运动命令内使用以覆盖永久值

注意

对单轴运动使用 IMMEDIATE 和 SUPERIMMEDIATE，在轴组/机器人中使用可能导致不想要的效果（跳跃）。

语法

<element>.StartType = <expression>

?<element>.StartType

域

ELEMENT

参数

< element >: 有效的运动元素

< expression >: 启动模式，Long，1 到 5

限制

此属性无法在运动命令内使用。要在任务中设置该值，必须将运动元素连接到该任务（使用 ATTACH 命令）。

示例

轴

A1.Strattype - InPosition

Move A1 100 StratType = Immed

机器人

SCARA.StartType = InPosition

Move SCARA {100, 20,0,0} StartType = 1

另请参见

ATTACH, ISMOVING, ISSETTLED, POSITIONERRORSETTLE, JUMP, JUMP3, JUMP3CP, MOVE, MOVES, WAITFORMOTION

STATE

说明

返回一个长整型 (Long) 值，指示事件任务的状态。

1：正在运行 (TASK_RUNNING)

2：由于 IDLETASK 或在 STEPIN 指令后停止 (TASK_STOPPED)

4：由于运行时错误而停止 (TASK_ERROR)

语法

?<event>.State

域

EVENT

参数

< event >: 事件名称

返回值

返回一个长整型 (Long) 值，指示事件任务的状态

<Return value>: Long，1，2，4

限制

事件必须加载到内存中（在执行 OnEvent 语句之后，调用任务正在运行时）。

示例

```
?Event1.State
If Event1.State = TASK_ERROR Then
‘Check if event is stopped due to run-time error
```

另请参见

STATUS

STATUS

说明

返回内存中所加载事件的状态。返回的信息将为以下格式：

State <description> Error <last error number> Source <line of source code> Owner: <calling task>

其中，<state> 为以下项之一：

1：正在运行

2：已停止（由于 IDLETASK）

4：由于运行时错误而停止

语法

?<event>.Status

域

EVENT

参数

< event >: 事件名称

返回值

返回内存中所加载事件的状态

<Return value>: String

限制

事件必须加载到内存中（在执行 OnEvent 语句之后，调用任务正在运行时）。

示例

? Event1.Status

另请参见

STATE

STOPPATH

说明

指令用于停止被绑定的机器人和轴的当前运动。当前的运动指令和待定的运动指令被存储。在 STOPPATH 被执行后，用户任务不会被阻塞。以下运动指令在用户程序中会自动开始执行。

这条命令可以与 CONTINUEPATH 命令一起使用。

NOTE

STOPPATH 在使用之前需要绑定特定的机器人和轴.

STOPPATH 在以下用例中再次使用：

CONTINUEPATH 已经被执行。

解绑已经被绑定的组/轴。

绑定的任务线程已经被杀掉。

这条指令的停止类型默认为 ONPATH。

运动指令可能会在正在执行的应用程序中被暂停按钮阻塞。但 STOPPATH 指令可以打断这个阻塞，然后这个阻塞的运动指令直接被执行。这一般发生在事件里。

缩写形式

Stoppath

语法

Stoppath < robot | axis>

参数

< robot | axis>: any valid robot or axis

限制

只写

示例

Stoppath SCARA

STOP

说明

此命令用于停止轴组或轴的运动。如果未指定 StopType，则使用 StopType 的永久值。此命令停止从轴并清除其作为从设备（SLAVE 属性设置为 0）。要同时停止所有系统运动元素，请将 SYSTEM.MOTION 设置为 0。运动将被抑制，直到把 SYSTEM.MOTION 设置回 1。

根据 <group/axis>.STOPTYPE 的值，将采取不同的操作：

StopType=1 *(Immediate) 根据给定的最大减速度（<axis>.DecMax 或轴组时每个所属轴的 <axis>.DecMax），当前运动的速度降低为零；如果是轴组运动，停止路径不必位于运动的轨迹上。*

如果存在另一个挂起的运动（在运动缓冲区中等待），将存储而不执行该运动。如果从另一个实例（另一个任务或终端窗口）发出 STOP 命令，则仅在发出 PROCEED 命令后才能继续新运动。

StopType=2 *(OnPath) 根据给定的最大减速度 (DecMax)，当前运动的速度降低为零；如果是轴组运动，停止路径将位于运动的轨迹上。*

如果存在另一个挂起的运动（在运动缓冲区中等待），将存储而不执行该运动。如果从另一个实例（另一个任务或终端窗口）发出 STOP 命令，则仅在发出 PROCEED 命令后才能继续新运动。

StopType=3 *(EndMotion) 当前运动将按发出的命令完成，没有任何干扰。*

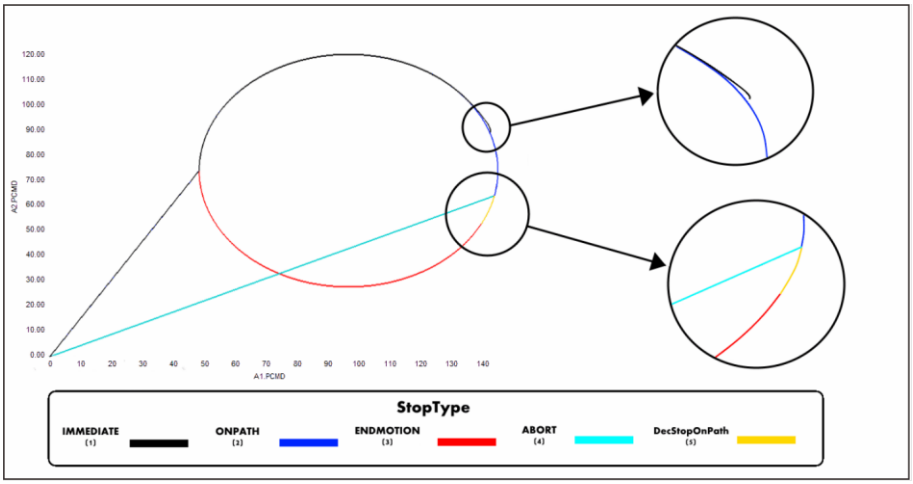
如果存在另一个挂起的运动（在运动缓冲区中等待），将存储而不执行该运动。如果从另一个实例（另一个任务或终端窗口）发出 STOP 命令，则仅在发出 PROCEED 命令后才能继续新运动。

StopType=4 *(Abort) 根据给定的最大减速度，当前运动的速度降低为零；如果是轴组运动，停止路径不必位于运动的轨迹上。*

如果存在另一个挂起的运动（在运动缓冲区中等待），将存储而不执行该运动。要继续其他运动，不需要 PROCEED 命令。

StopType=5 *(DecStopOnPath) 根据当前运动的减速度值 (DecStop/DecTran/DecRot)，当前运动的速度降低为零；如果是轴组运动，停止路径将位于运动的轨迹上。*

如果存在另一个挂起的运动（在运动缓冲区中等待），将存储而不执行该运动。如果从另一个实例（另一个任务或终端窗口）发出 STOP 命令，则仅在发出 PROCEED 命令后才能继续新运动。



注意

STOP 不会直接影响任务执行。该命令停止运动，但不会停止任务执行。

缩写形式

Stop

语法

Stop < robot | axis> {StopType=<stop type>}

参数

<robot | axis>：任何有效的机器人或轴

<stop type>: Long，1 到 5

限制

只写

示例

Stop SCARA

Stop A1 StopType=EndMotion

另请参见

PROCEED

STR$

说明

STR$ 返回数字的字符串表示。

语法

STR$(<number>)

参数

<number>: Double，0 到 MaxDouble

返回值

返回数字的字符串表示

<return value>: String

限制

只读

示例

I1=12345

PRINT STR$(i1)+"A"

打印以下内容：12345A

另请参见

STRD$, STRL$

STRD$

说明

STRD$ 返回双精度型 (Double) 数字的字符串表示。

注意

如果使用无效的格式字符串，则行为未定义

语法

STRD$(<number>, <format_string>)

参数

<number>: Double，Long，MinDouble 到 MaxDouble

<format_string>: String，有效的 sprintf() 浮点型格式

%f - 非指数显示

%e, %E - 指数显示

%g, %G - 显示类型取决于输入值

返回值

返回双精度型 (Double) 数字的字符串表示

<return value>: String

限制

只读

示例

PRINT STRD$(45.5, "%f")

打印以下内容：45.500000

PRINT STRD$(45.5,"%e")

打印以下内容：4.550000e+01

PRINT STRD$(45.5,"%g")

打印以下内容：45.5

另请参见

VAL, STR$, STRL$

STRING$

说明

STRING$ 生成具有指定字符数的新字符串。

每个字符均为指定字符串参数的第一个字符或指定的 ASCII 码。

注意

ASCII 码集合包含 256 个代码（0 到 255）。如果指定的 <ascii code> 大于 255，则 <ascii code> 将对 256 取模，余数用作 ASCII 码。

语法

STRING$(<number>,<string>)

STRING$(<number>,<ascii code>)

参数

<number>: Long

<string>: String

<ascii code>: Long

返回值

返回具有指定字符数的新字符串。

<return value>: String

限制

只读。输入字符串不能是 UTF-8 字符串。

示例

? STRING$(23, “*”) ‘generates a string of 23 asterisks

? STRING$(23, “*5”) ‘generates a string of 23 asterisks

? STRING$(9, 65) ‘generates a string of 9 A

另请参见

CHR$, SPACE$, UTFSTRING$, ASC

STRL$

说明

STRL$ 返回长整型 (Long) 数字的字符串表示。

注意

双精度型 (Double) 输入将隐式转换为长整型 (Long)

如果使用无效的格式字符串，则行为未定义

语法

STRL$(<number>, <format_string>)

参数

<number>: Long，Double

<format_string>: String，有效的 sprintf() 整数格式

%d, %i - 有符号十进制

%u - 无符号十进制

%o - 八进制

%x, %X - 十六进制

%c - 无符号字符

返回值

返回长整型 (Long) 数字的字符串表示

<return value>: String

限制

只读

示例

PRINT STRL$(60, "%d")

打印以下内容：60

PRINT STRL$(60,"%o")

打印以下内容：74

PRINT STRL$(60,"%x")

打印以下内容：3c

PRINT STRL$(60,"%c")

打印以下内容：<

另请参见

VAL, STR$, STRD$, HEX$, CHR$

SPLITSTR$

说明

使用特定的分隔符分割字符串

语法

SplitStr(<string>, <split>,sOutput[*])

参数

< string >: string 类型, 需要被分割的字符串

< split >: String 类型, 分割符

< sOutput[*] >: 结果，分割后的字符串数组

返回值

返回分割后的字符串数量.

示例

```
Dim sOutput[4] as string
?SplitStr(“100,200,300”,”,”,sOutput)
Return 3
?sOutput[1] = “100”
?sOutput[2] = “200”
?sOutput[3] = “300”
```

```
Dim sOutput[2] as string
?SplitStr(“100,200,300”,”,”,sOutput)
Return 2
?sOutput[1] = “100”
?sOutput[2] = “200,300”
```

```
Dim sOutput[3] as string
?SplitStr(“100,200,300”,”:”,sOutput)
Return 1
?sOutput[1] = “100,200,300”
```

另请参见

MID$, RIGHT$, LEFT$

STRUCTURE_TYPE_DEFINITION

说明

由于结构体是一种新数据类型，因此必须首先进行定义。

只有这样，新数据类型“结构体”的名称才能用于声明结构体变量。

结构体数据类型定义可以在 config.prg 文件的声明部分（PROGRAM 块之前）或库文件（第一个函数块之前）中完成。该代码块定义结构体元素的名称、类型和数组大小。

结构体支持的不同类型如下：长整型、双精度、字符串、关节空间坐标或笛卡尔空间坐标、通用轴、通用轴组、信号量、用户错误和用户提示。

用户错误和提示类型的结构体元素是通用的，即应赋值以获取错误编号和消息。

语法

```
TYPE <structure_type_name>
<field_name>{[<index>]…} as <type> {of <type>}
…
END TYPE
```

参数

<structure_type_name>: 类型名称

<field_name>: 类型成员名称

<index>: 数组大小

<type>: 类型成员的类型

限制

数组类型的结构体元素的维数不能超过 10。

信号量类型的结构体元素只能是标量。

用户错误和提示类型的结构体元素只能是标量。

库重新加载期间，不允许更改类型定义。

示例

```
Type ST
 LngElem      as long           'Long scalar element
 LngArr[10]   as long           'Long array element
 DblElem      as double         'Double scalar element
 DblArr[2][3] as double         'Double array element
 StrElem      as string         'String scalar element
 StrArr[4][2][5]  as string     'String array element
 JntElem      as joint of XYZ   'Joint scalar element
 JntArr[7][10][3] as joint of XYZ   'Joint array element
 LocElem      as location of XYZR   'Location scalar element
```

```
  LocArr [6]    as location of XYZR      'Location array element
  A1        as generic axis        'Generic axis scalar element
  AxarSCARA[4]     as generic axis        'Generic axis array element
  GSCARA        as generic group       'Generic group scalar element
  GrarSCARA[3][5]   as generic group       'Generic group array element
  Sem1       as semaphore          'Semaphore element
  ErSCARA       as error           'User error element
  Nt1        as note            'User note element
End Type
'usage
dim st1 as ST
st1->DblElem = 0.1
```

SUB_..._END_SUB

说明

定义可从局部任务调用的库子程序。如果使用关键字 Public 作为 SUB..END SUB 的前缀，则子程序将位于全局操作范围内，并可从任何其他任务调用。

语法

{Public} SUB <name> ({ {ByVal} <p_1> as <type_1>}…{, {ByVal} <p_n> as <type_n>})

{ local variable declaration }

{ subroutine code }

END SUB

参数

{Public}：指定子程序范围的可选关键字

<name>: 子程序名称，某些特定符号除外

{ByVal}：将参数转换为值

<p_1>: 函数参数

<type_1>: 参数类型

<p_n>: 函数参数

<type_n>: 参数类型

{local variable declaration}：子程序主体

{ subroutine code}：子程序主体

限制

数组只通过引用传递。

示例

```
Public Sub Sub1(lastLoop as Long) 'Pass parameters by reference
    Dim Index as long
    For Index = 1 to lastloop
        move a1 1 abs = 0
    Next Index
End Sub
```

另请参见

CALL, FUNCTION_..._END_FUNCTION, PROGRAM_..._END_PROGRAM

SYSTEM.ERROR

说明

查询上次系统错误消息。此命令返回包含错误编号的错误消息。

SYSTEM.ERRORNUMBER 查询仅返回错误的编号。

语法

?Sys.Error

?System.Error

域

SYSTEM

返回值

返回包含错误编号的错误消息

限制

只读

system.error 包含 Try-catch 块未处理的最后一个错误。

示例

?System.Error

SYSTEM.MOTION

说明

SYSTEM.MOTION 必须为 1 (ON) 才能成功执行任何运动命令。运动的状态由运动控制器终止。

如果 MOTION 切换为 0 (OFF)，则停止当前正在进行的任何运动，并以最大减速度值将速度减速为零，然后刷新运动缓冲区中的任何运动。

每个轴的运动标志也必须为 ON 才能执行运动命令。

语法

Sys.Motion = <value>

System.Motion = <value>

?Sys.Motion

?System.Motion

域

SYSTEM

参数

<value>: Long，0 或 1

0 = OFF

1 = ON

限制

默认系统错误处理器将 SYSTEM.MOTION 标志设置为零。发生这种情况时，SYSTEM.MOTION 标志只能从终端重置为 1。

示例

System.Motion = On

另请参见

MOVE, MOTION

TAN

说明

此函数返回给定角度（弧度）的正切值。

语法

TAN(<expression>)

参数

<expression>: 任何有效的表达式，Double，± MaxDouble

返回值

返回给定角度（弧度）的正切值

限制

只读

示例

?Tan (3.14159/2)

VaSCARA = Tan(3.14159)

另请参见

ATAN2, ATN, COS, SIN

TASKERROR

说明

打印任务中发生的最后一个错误的文本说明

语法

?TaskError (<task name>)

参数

<task name>: String

返回值

返回任务中发生的最后一个错误的文本说明

<return value>: String

限制

只读

示例

?TaskError("task1.prg")

或

Common Shared StSCARA asString

StSCARA = “Task1.Prg”

?TaskError(StSCARA)

另请参见

ERROR

TASKERRORNUMBER

说明

打印任务中发生的最后一个错误的编号

语法

?TaskErrorNumber (<task name>)

参数

<task name>: String

返回值

返回任务中发生的最后一个错误的编号

<return value>: Long

限制

只读

示例

?TaskErrorNumber("task1.prg")

或

Common Shared StSCARA asString

StSCARA = “Task1.Prg”

?TaskErrorNumber(StSCARA)

另请参见

TASKERROR

TASKID

说明

返回一个长整型 (Long) 值，指示唯一的任务标识号。

如同从本任务查询一样，可以从另一个任务的代码行中查询。

如果从同一个任务查询，则无需任务名称。

语法

?<task>.TASKID

?TASKID

域

TASK

参数

<task>: 内存中任务的名称

返回值

返回一个长整型 (Long) 值，指示唯一的任务标识号

<return value>: Long

限制

只读。任务必须加载到内存中

示例

?Task1.prg.TaskID hex

0x7B6CEE0

或从 Task1.prg：

Program

? TaskID hex

‘ should return 0x7B6CEE0

end program

TASKLIST

说明

列出已加载任务的名称和状态。每个任务均为以下格式：

TaskName=<task>,State=<state>, Priority=<priority> {,FileName=<file>}

任务状态：

1：正在运行

2：已停止

3：单步模式

4：错误

5：已终止

6：继续模式

7：就绪 (SUSPEND STATE)

8：任务处于“继续”状态（此状态无效）

9：不正确的状态（可能终止任务失败）

10：已终止 (SUSPEND STATE)

除上述状态外，任务还可为“+已锁定”。任务优先级等级从 1 到 16。最高优先级为 1。

如果未加载任何任务，则响应为“No tasks found”（未找到任何任务）。

FileName 字段仅当使用 Loas$…As 命令加载与其原始文件名具有不同名称的任务时显示。

语法

?TaskList

返回值

返回已加载任务的名称和状态。

<return value>: String，格式：

TaskName=<task>,State=<state>, Priority=<priority> {,FileName=<file>}

限制

只读

示例

?TaskList

TaskName=TP_ALIVE.PRG,State=Running,Priority=1

TaskName=RBT_CFG.PRG,State=Killed,Priority=16

TaskName=MCU.PRG,State=Running,Priority=16

GlobalLibraryName=UTILS.LIB

GlobalLibraryName=I18N.LIB

另请参见

PLSLIST, AXISLIST, GROUPLIST, VARLIST, MAINFILENAME, VARLIST$

TASKSTATE

说明

返回一个长整型 (Long) 值，指示任务的状态。有两种情况下会修改常规任务状态：

1) 任务已锁定

2) 任务已中断

如果任务正在等待系统事件完成执行，则处于状态 2、4 和 10 的任务可能已锁定。

在这种情况下，会给常规任务状态添加值 256 (0x100)。任务有三个连续运动时可能会发生这种情况。直到第一个运动完成，解释器均在等待执行第二个运动。

如果从终端输入 STOP 命令，任务仍处于“正在运行”模式。

从终端输入 IDLETASK 命令，任务将切换为“已停止”和“已锁定”状态。其锁定原因是等待 Proceed 命令以继续运动。

任务通过 OnEvent 或 OnError 中断。在这种情况下，会给常规任务状态添加值 512 (0x200)。

1：正在运行 (TASK_RUNNING)

2：由于 IDLETASK 或在 STEPIN 指令后停止 (TASK_STOPPED)

4：由于运行时错误而停止 (TASK_ERROR)

5：已终止。

可在以下情况短暂看到该状态：1.终止程序 2.卸载

7：LOAD 后就绪 (TASK_READY)

9：终止任务启动。这是成功后任务终止程序的第一种状态

终止状态内部更改为 TASK_KILLED(10)。在某些情况下，由于 IO 操作未完成或

延迟，任务可能会保持在此状态，并需要其他终止命令。

10：在 KILLTASK 或 END PROGRAM 后终止 (TASK_KILLED)

如果任务被事件中断，则向任务状态添加值 512。如果任务已锁定，则向任务状态添加值 256。

语法

?TaskState (<task name>)

参数

<task name>: String

返回值

返回一个长整型 (Long) 值，指示任务的状态

<return value>: Long，1，2，4，5，7，9，10

限制

任务必须加载到内存中

示例

?TaskState(“Task1.prg”)

或

Common Shared StSCARA as String

StSCARA = “Task1.prg”

?TaskState(StSCARA)

另请参见

TASKSTATUS, ELEMENTSTATUS

TASKSTATUS

说明

此查询返回加载到内存中的任务的状态。 返回的信息将采用以下格式：

State <state>: <description> Error <last error number> Source <line of source code> Main program <line of source code>

其中 <state> 是以下之一：

1: Running

2: Stopped (due to IDLETASK)

4: Stopped due to run-time error

5: Terminated.

可以在以下时刻看到该状态：

1. Terminate program

2. Unload

7: Ready (after LOAD)

9:Task Kill Start .

10: Killed (after KILLTASK or END PROGRAM)

语法

?TaskState (<task name>)

参数

<task name>: String

返回值

返回一个字符串，指示任务的状态

<return value>: String, 格式如下：

State <state>: <description> Error <last error number> Source <line of source code> Main program <line of source code>

限制

任务必须加载到内存中

示例

Dim Shared MyTask as string = "<TaskName.PRG>"

?TaskStatus(MyTask)

另请参见

TASKSTATE

THROW

说明

断言应用程序异常。错误名称是用户定义的异常名称。如果未指定错误名称，将断言任务中发生的最后一个错误。

语法

Throw <error name>

参数

< error name >: 错误名称

限制

仅接受异常名称作为参数。

示例

Throw myerror

TIME

说明

此命令查询或设置当前时间。

小时、分钟和秒钟分别用两位数指定。系统使用 24 小时制表示法。

语法

Sys.Time = “<hours>:<minutes>:<seconds>”

System.Time = “<hours>:<minutes>:<seconds>”

域

SYSTEM

参数

String

<hours>: 00 到 23

<minutes>: 0 到 59

<seconds'>：0 到 59

示例

System.Time = “15:15:25” ‘set time to 15:15:25

TMOUT

说明

此命令查询当前任务中 WAIT 指令是否超时。

语法

? Sys.Tmout

域

SYSTEM

参数

无

返回值

返回当前任务中 WAIT 指令是否超时的状态，TRUE-超时，FALSE-未超时

示例

```
WAIT Signal1 = on tmout = 1000
? Sys.Tmout
```

TOASCII8$

说明

将输入字符串的 UTF-8 编码格式转换为 ASCII-8 编码格式。位于 ASCII-8 范围外的符号（高于 0xFF 的 Unicode 值）将替换为问号。

语法

? TOASCII8$(<string>)

参数

<string>: String，UTF-8 编码格式

返回值

返回 ASCII-8 编码格式字符串

<Return value>: String

示例

Common shared UTF8Str as string of UTF8

? TOASCII8$(UTF8Str)

另请参见

TYPEOF

TOCART

说明

将关节空间坐标转换为笛卡尔空间坐标。执行正向运动学功能。输入参数：

<Robot> = 计算正向运动学的轴组名称，执行 WITH 命令时，不需要此参数（逗

号也可省略）。

<joint> = 要转换为笛卡尔坐标的点的关节空间坐标。关节空间坐标变量类型必须

与机器人类型兼容。

语法

<location_variable>=ToCart (<Robot>, <joint>)

参数

< Robot >: 机器人名称

<joint>: Joint，要转换为笛卡尔坐标的点的关节空间坐标

返回值

<Return value>: 笛卡尔坐标形式的位置点

限制

笛卡尔空间坐标变量和关节空间坐标参数必须具有与轴组参数完全相同的轴组类型，或具有相同的坐标数。

示例

? ToCart (SCARA, {1,2,3})

? ToCart (SCARA, SCARA.Pfb)

L1 = ToCart (SCARA, J1)

另请参见

TOJOINT

TOJOINT

说明

通过执行反向运动学功能将笛卡尔空间坐标转换为关节空间坐标。

输入参数：

<Robot>

计算正向运动学的轴组名称，执行 WITH 命令时，不需要此参数（逗号也应省略）。

<Location>

要转换为关节空间坐标的点的笛卡尔空间坐标。

<configuration>

描述返回的关节空间坐标位置配置的整数（对于 SCARA，它是 ARMCMD 或 ARMFBK）。

配置标志的数量和类型取决于每种机器人类型。

请注意，机器人属性 ARMCMD、ECMD、WCMD 均有三个值（0、1、2），其中 0 已保留，用于 AUTO 选项。

但配置位只有两个值（0、1）。

对于 SCARA 机器人

机械臂（位 0）

值 = 0 - Lefty (armcd = 1)

值 = 1 - Righty (armcmd = 2)

注意

常量 lefty、righty、above、below、flip 和 noflip 定义为 MC-Basic 语言的内置常量。

语法

<joint_variable>=ToJoint (<Robot>, <location>, <configuration>)

参数

< Robot >: 机器人名称

<location>: Location，要转换为关节空间坐标的点的笛卡尔坐标

< configuration >: Long，0 或 1，描述返回的关节空间坐标位置配置的整数

返回值

<Return value>: 关节空间坐标形式的位置点

限制

关节空间坐标变量和笛卡尔空间坐标参数必须为与轴组参数完全相同的轴组类型，或具有相同的坐标数。

配置参数（长整型参数）的值只能为：0、1 或 2，与 ARMCMD 含义相同。

示例

? ToJoint (SCARA, #{1,2,3}, 1)

? ToJoint (SCARA, SCARA.Here, X)

?ToJoint(#{768.198 , 0 , 233.198 , 0 , 180 , 0},0b010)

J1 = ToJoint (SCARA, L1, 0)

另请参见

TOCART

TOOL

说明

工具坐标系设置了指定位置用于系统工具坐标变换。

它定义工具尖端相对于工具法兰中心的位置和方向。

默认的工具坐标系变换是 NULL 变换，可表示为 TOOL = #{0,0,0,0}

语法

Tool = <robot location point>

域

ROBOT

参数

<robot location point>: Location

返回值

返回世界坐标系中的工具坐标系位置

<return value>: Location

示例

Scara.Tool = #{90, 180, 0, 0}

?Scara.MachineTable:WorkPiece:Base:Tool

{90 , 90 , 90 , 180}

另请参见

BASE, USERFRAME, WORKPIECE, SELECTTOOL

TORQUEADDCOMMAND

说明

轴附加扭矩的命令值。

缩写形式

<axis>.TAddCmd

语法

?<axis>. TORQUEADDCOMMAND

域

AXIS

参数

< axis >: 任何有效轴

返回值

返回轴附加扭矩的命令值。

<return value>:

轴 – Double, MINDouble 到 MAXDouble

限制

只读

示例

轴

?A1.TAddCmd

另请参见

TORQUEFEEDBACK, TORQUEERROR, OPMODE

TORQUECOMMAND

说明

返回缩写形式

<element>.TCMD

语法

?<element>.TCMD

域

ELEMENT

参数

<element>: 任何有效的运动元素

返回值

返回运动元素力矩命令值

<return value>:

轴组 - 点

轴 - Double，MINDouble 到 MAXDouble

限制

只读

示例

轴

?A1.TCMD

机器人

?SCARA. TCMD

另请参见

TORQUEFEEDBACK, OPMODE

TORQUEERROR

说明

轴转矩误差值，变量 TorqueAddCommand 和 TorqueFeedback 的实时在线差值。

如果这个值超过了 TorqueErrorMax，根据 TorqueErrorStopType 和 TorqueErrorDisableType 中用户定义的设置，扭矩错误应对程序会被激活(Stop 和 Disable)，并抛出运动错误。

缩写形式

<axis>. TE

语法

?<axis>. TORQUEERROR

域

AXIS

参数

< axis >: 任何有效轴

返回值

返回轴转矩误差值

<return value>:

轴 – Double, MINDouble 到 MAXDouble

限制

只读

示例

轴

?A1. TE

另请参见

TORQUEFEEDBACK, TORQUEADDCOMMAND, OPMODE

TORQUEFEEDBACK

说明

返回运动元素力矩反馈值

缩写形式

<element>.Tfb

语法

?<element>.Tfb

域

ELEMENT

参数

<element>: 任何有效的运动元素

返回值

返回运动元素力矩反馈值

<return value>:

轴组 - 点

轴 - Double，MINDouble 到 MAXDouble

限制

只读

示例

轴

?A1.Tfb

机器人

?SCARA. Tfb

另请参见

TORQUECOMMAND, TORQUEERROR, TORQUEADDCOMMAND, OPMODE

TORQUEGEARCOMMAND

说明

此属性返回作用在变速箱输出轴上的计算扭矩，该扭矩减去了反映到变速箱输出轴上的摩擦扭矩和转子惯性扭矩。该扭矩的计算方法是基于用于计算 TORQUEADDCOMMAND 的动力学模型。

缩写形式

<element>.TGCmd

语法

?<element>.TORQUEGEARCOMMAND

域

ELEMENT

参数

< element >: 任何有效的运动元素

返回值

返回作用在变速箱输出轴上的计算扭矩

<return value>:

轴组 - 点

轴 – Double, MINDouble 到 MAXDouble

限制

只读

示例

轴

?A1.TGCmd

机器人

?SCARA.TGCmd

另请参见

TORQUEGEARFEEDBACK, OPMODE

TORQUEGEARFEEDBACK

说明

此属性返回作用在变速箱输出轴上的计算扭矩，该扭矩减去了反映到变速箱输出轴上的摩擦扭矩和转子惯性扭矩。 该扭矩的计算方法基于用于计算对应属性 TorqueGearCommand 的动态模型，但使用了反馈值而不是指令值。

缩写形式

<element>.TgFbk

语法

?<element>.TorqueGearFeedback

域

ELEMENT

参数

< element >: 任何有效的运动元素

返回值

返回作用在变速箱输出轴上的计算扭矩

<return value>:

轴组 – 点

轴 – Double, MINDouble 到 MAXDouble

限制

只读

示例

轴

?A1. TgFbk

机器人

?SCARA. TgFbk

另请参见

TORQUEGEARCOMMAND, OPMODE

TOTALTIME

说明

返回当前所执行运动的总时间。

值：

-2 - 不适用（PrfType = 0 或 1）。

-1 - 无活动运动。

0 - 不可能发生，因为没有持续时间为零的运动。

> 0 - 运行运动的总时间。

注意

请注意，如果是圆滑的情况，将返回圆滑运动的第一段运动的 TotalTime/CurrentTime！圆滑停止（仅执行第二段运动，即第一段运动已完成）后，将返回第二段运动的时间值。

缩写形式

<element>.Ttime

语法

<element>?Ttime

域

ELEMENT

参数

< element >: 有效的运动元素

返回值

返回当前所执行运动的总时间

<return value>: Long，-2 到 MaxInt

限制

只读

示例

轴

?A1.totaltime

机器人

TOUTF8$

说明

将输入字符串的 ASCII-8 编码格式转换为 UTF-8 编码格式。

语法

? TOUTF8$(<string>)

? TOUTF8$(<string>)

参数

<string>: String，ASCII-8 编码格式

返回值

返回 UTF-8 编码格式字符串

<Return value>: String

示例

Common shared ASCII8Str as string

? TOUTF8$(ASCII8Str)

Common shared ASCII8Str as string

? TOUTF8$(ASCII8Str)

另请参见

TYPEOF

TRY_..._END_TRY

说明

TRY…END TRY 用于捕获同步错误。

同步错误是由用户任务引起并由解释器检测到的错误。

此类错误与用户自定义任务中的特定程序代码行相关联。

示例包括移动命令中的除零参数和超出范围的参数。

TRY 块用于对程序代码的特定区域执行特定操作。它可在 OnError 或 OnSystemError 块中使用。可以使用 User Error 的 .num 属性捕获用户错误。

注意

如果捕获子句再次抛出自身，则不应被“Catch Else”或“Finally”指令捕获。

语法

```
Try          ‘ Start of Try block
<code being Terminaled>
{Catch <Error Number X>
<code to execute when error X occurs> }
{Catch <MyError.num>
<code to execute when MyError occurs > }
{Catch Else
{code to catch allother errors} }
{Finally
<code to execute only if error occurred and was trapped> }
End Try          ‘ End Try block
```

限制

如果嵌套的 TRY 只出现在 Catch 块中，则允许嵌套 TRY 块。GOTO 指令允许在 Catch 部分内，但引用的标签必须在该部分内。

示例

```
common shared appErroSCARA as error "Application error" 20561
program
 try
' come code
 catch 8001
' division by zero
 catch appErroSCARA.num
 Print "Application error"
 end try
 end program
```

另请参见

ONERROR, PROGRAM_..._END_PROGRAM

TYPEOF-ROBOT

说明

返回轴组模型：

语法

<point_variable> = <robot_name>.typeof

? <robot_name>.typeof

? <robot_name>.typeof

域

ROBOT

参数

<robot_name>: 任何有效的机器人

返回值

返回轴组模型

<point_variable>: Long

示例

? Scara.Typeof

TYPEOF

说明

此函数返回表示字符串类型的数字（0 表示无类型，1 表示 ASCII-8，或 2 表示 UTF-8）。

语法

TYPEOF(<string>)

TYPEOF(<string>)

参数

<string>: String

返回值

返回表示字符串类型的数字（0 表示无类型，1 表示 ASCII-8，或 2 表示 UTF-8）

<return value>: Long，0，1，2

限制

只读

示例

?TYPEOF(“…”)

?TYPEOF(“…”)

‘ returns 0

?TYPEOF(TOASCII8$(“…”))

‘ returns 1

?TYPEOF(TOUTF8$(“…”))

‘ returns 2

另请参见

TOASCII8$, TOUTF8$

UCASE$

说明

UCASE$ 函数返回所传递字符串的副本，所有小写字母转换为大写字母。

语法

UCASE$(<string>)

参数

<string>: String

返回值

返回所传递字符串的副本，所有小写字母转换为大写字母

<return value>: String

限制

只读

示例

PRINT UCASE$(“first step”)

打印以下内容： FIRST STEP

另请参见

LCASE$

USERFRAME

说明

用户坐标系是一个机器人属性，用户可以通过 location 常量设置系统的用户坐标系，用于坐标变换计算。

它以相对于 WORLD 参考系的位置和方向定义了机器工作空间。

默认的用户坐标系变换是 NULL 变换，可表示为 SCARA.UserFrame = #{0,0,0,0}。

语法

UserFrame = <robot location point>

域

ROBOT

参数

<robot location point> : Location

返回值

返回世界坐标系中的用户坐标系位置

<return value>: Location

示例

Scara.UserFrame = #{90, 180, 0, 0}

?Scara.UserFrame:WorkPiece:Base:Tool

{90 , 90 , 90 , 180}

另请参见

BASE, TOOL, WORKPIECE, SELECTUSERFRAME

UTF$

说明

此函数返回与给定 Unicode 值对应的 UTF-8 字符串。

语法

UTF$(<number>)

参数

<number>: Long，0 到 MAXLONG

返回值

返回与给定 Unicode 值对应的 UTF-8 字符串

<return value>: UTF-8 字符串，

限制

只读

示例

?UTF$(256)

打印以下内容： Ā

另请参见

CHR$, UTFSTRING$, ASC

UTFSTRING$

说明

UTFSTRING$ 生成具有指定符号数的新 UTF-8 字符串。每个符号均为指定字符串参数的第一个字符或指定的 Unicode 码。

语法

UTFSTRING$(<number>,<string>)

UTFSTRING$(<number>,<Unicode_value>)

参数

<number>: Long，0 到 MaxLong

<string>: UTF-8 字符串

<ascii code>: Long，0 到 MaxLong

返回值

<return value>: UTF-8 字符串

限制

只读。输入字符串必须是 UTF-8 字符串。

示例

```
Common shared UTF8Str as String of UTF8 = UTF$(196) + UTF$(197)
? UTFSTRING$(10, UTF8Str) ‘ generates a string of 10Ä
? UTFSTRING$(10, 196) ‘ generates a string of 10Ä
```

另请参见

STRING$, UTF$

VAL

说明

VAL 返回输入字符串中的字符表示的实际值。

输入字符串可以是十进制模式、科学模式、科学记数法或十六进制模式（但不能是二进制模式）的数字。

语法

VAL(<string>)

参数

<string>: String

返回值

返回输入字符串中的字符表示的实际值

<return value>: Double 精度，± MaxDouble

限制

只读

示例

value = VAL("123 Example")

值 = 123（仅转换第一个合法数字）

value = VAL("123 E124xa4m3p24le")

值 = 123（系统提取数字）

value = VAL("1.2E-2")

值 = 0.012

value = VAL("0xFF")

值 = 255

?VAL("")

打印以下内容：0

?VAL("One hundred")

打印以下内容：0

?VAL("12e7a3")

打印以下内容：12e7

?VAL("65+32")

打印以下内容：65

?VAL("cos(32)")

打印以下内容：0

?VAL("321")

打印以下内容：321

另请参见

STRD$, STRL$

VARLIST

说明

返回系统中定义的变量和常量名称的列表。如果使用通配符，查询将返回正确的现有变量或常量数据。查询带名称空间的变量的 varlist 时，可以使用问号 (?) 代替通配符。

变量类型及其值显示为以下格式：

{const} <type> <var_name> = <value>

常量名称显示有 const 字样。

语法

?VarList {<var_name>}

参数

{<var_name>}：变量名称

返回值

返回系统中定义的变量和常量名称的列表

<return value>: String，格式：

{const} <type> <var_name> = <value>

限制

只读

示例

?varlist

?varlist i*'Returns the variables that start with the letter i

LONG iVariable = 221

CONST LONG iConst = 1

另请参见

PLSLIST, AXISLIST, GROUPLIST, TASKLIST

VARLIST$

说明

返回系统中定义的变量和常量名称的列表。如果使用通配符，查询将返回正确的现有变量或常量数据。

变量类型及其值显示为以下格式：

{const} <type> <var_name> = <value>

常量名称显示有 const 字样。

语法

?VarList$ (<string>)

参数

<string>: String，字符串形式的变量名称

返回值

返回系统中定义的变量和常量名称的列表

<return value>: String，格式：

{const} <type> <var_name> = <value>

{const} <type> <var_name> = <value>

限制

只读

示例

```
?varlist$("")
?varlist$("i*")'Returns the variables that start with the letter i
LONG iVariable = 221
CONST LONG iConst = 1
```

另请参见

PLSLIST, AXISLIST, GROUPLIST, TASKLIST

SETLOCALVAR

说明

设置局部变量值。

语法

SETLOCALVAR < task_name > <var_name> = < expression >

参数

< task_name >: String，变量所属的任务名称

<var_name>: String，变量名称

< expression >：long 或者 double 类型表达式

限制

变量类型必须为整型或者浮点型，此外不支持数组类型

示例

SETLOCALVAR taskname var = 1

SETLOCALVAR taskname var = 1.0

SETLOCALVAR taskname var = 1.0 + 1

VCLOSETCP

说明

此命令可用于手动关闭通过 VOPENTCP 命令打开的套接字。

语法

VCloseTcp(<StationName>)

参数

<StationName>: 任何有效的视觉站, String

示例

```
VOpenTcp(“VS1”)
If lState then
    VReceiveData("VS1",SDATA,lState, 1000)
End if
VCloseTcp(“VS1”)
```

另请参见

VOPENTCP, VRUNJOBFULL

VELERRTHRESHOLD

说明

此命令用于设置指定轴的速度误差阈值，用于碰撞检测的触发条件,必须 attach scara 后才能设置。

语法

<ELEMENT>.VelErrThreshold

参数

< ELEMENT >: 合法的运动元素

示例

J1. VelErrThreshold = 100

? J1. VelErrThreshold

另请参见

RESETPEAKVALUE, SETVELERRTHRESHOLD, COLLISIONDETECT, DRIVETORQUELIMIT, PEAKTORQUE, PEAKVELERR, SETDRIVETORQUELIMIT, SETDEFVELERRTHRESHOLD

VELOCITYCOMMAND

说明

此属性返回速度命令。该值通过运动控制器生成。

缩写形式

<element>.VCmd

语法

?<element>.VelocityCommand

域

ELEMENT

参数

< element >: 有效的运动元素

返回值

返回速度命令

对于轴 - Double

对于轴组 - 机器人的关节空间坐标系数据类型

± Max Double

限制

只读

示例

轴

?A1.VCmd

机器人

?SCARA.VCmd

?SCARA.VCmd

另请参见

VELOCITYCOMMANDCARTESIAN, VELOCITYFEEDBACK, ACCELERATIONCOMMAND, POSITIONCOMMAND

VELOCITYCOMMANDCARTESIAN

说明

返回机器人 TCP 的当前笛卡尔命令速度。

前三个组成部分为线性速度，而最后一个组成部分为角速度。

缩写形式

<ROBOT>.VCMDCART

语法

<ROBOT>.VelocityCommandCartesian

域

ROBOT

参数

< ROBOT >: 任何有效的机器人

返回值

返回机器人 TCP 的当前笛卡尔命令速度

<return value>: LOCATION，0 到 MaxDouble

限制

只读

示例

?Scara.VCMDCART

另请参见

VELOCITYCOMMAND

VELOCITYERROR

说明

此属性返回速度跟踪误差，即速度命令与速度反馈之间的差值。

此属性仅用于查询。与 PE 和 PEmax 不同的是，没有针对最大阈值的检查。

该计算使用位置误差延迟，以从之前多个周期发出的命令中减去速度反馈，如 POSITIONERRORDELAY 属性中所定义。

在轴中，速度误差是速度命令与速度反馈之间的差值。

在轴组中，速度误差是所有轴速度误差平方和的平方根。

在机器人中，速度误差使用速度命令与工具尖端反馈之间差值的位置部分（仅 XYZ 值，方向部分不使用）来计算：

$SQRT((VCmd\{1\}-VFb\{1\})^2 + (VCmd\{2\}-VFb\{2\})^2 +(VCmd\{3\}-VFb\{3\})^2)$

缩写形式

<element>.VE

语法

?<element>.VelocityError

域

ELEMENT

参数

< element >: 有效的运动元素

返回值

返回速度跟踪误差，即速度命令与速度反馈之间的差值

<return value>: Double

限制

只读

示例

轴

?A1.VE

机器人

?SCARA.VE

机器人

?SCARA.VE?SCARA.VE

VELOCITYFEEDBACK

说明

此属性返回运动元素的实际空间速度。

缩写形式

<element>.VFb

语法

?<element>.VelocityFeedback

域

ELEMENT

参数

< element >: 有效的运动元素

返回值

返回运动元素的实际空间速度

<return value>: Double

限制

只读

示例

轴

?A1.Vfb

机器人

?SCARA.VFb

另请参见

VELOCITYCOMMAND

VELOCITYFEEDBACKCARTESIAN

说明

返回机器人 TCP 当前反馈速度的矢量。

前三个组成部分为线性速度，而最后一个组成部分为角速度。

缩写形式

<ROBOT>.VFBCART

语法

<ROBOT>.VelocityFeedbackCartesian

域

ROBOT

参数

<ROBOT>: 任何有效的机器人

返回值

返回机器人 TCP 当前反馈速度的矢量

<return value>: LOCATION，0 到 MaxDouble

限制

只读

示例

?Scara.VFBCART

?Scara.VFBCART

VELOCITYFINALROT

说明

定义机器人的最终旋转速度。与 VFTRAN 一起，定义笛卡尔运动的最终速度。

如果速度不为零，则另一个最终速度 (VFTRAN) 必须也不为零。

该值仅在两个运动命令中使用：MOVES 和 CIRCLE。在轴插补运动 (MOVE) 中，该值将被忽略。

轴组必须使用机器人模型 (model !=1) 进行定义。

该值不得大于 VELOCITYMAXROT。

系统始终采用两者中较小的值，并向用户发送通知消息。

缩写形式

<ROBOT>.vfrot

语法

<ROBOT>.vfrot=<numeric expression>

域

ROBOT

参数

< ROBOT >: 任何有效的机器人

<numeric expression>: Double，0.1 到 Maxdouble

限制

写入，仅非独立属性

示例

MoveS Scara #{10,20,30,0} vfrot = 6000

另请参见

CIRCLE, MOVES

VELOCITYFINALTRANS

说明

定义机器人的最终平移速度。与 VFROT 一起，定义笛卡尔运动的最终速度。

如果速度不为零，则另一个最终速度 (VFROT) 必须也不为零。

该值仅在两个运动命令中使用：MOVES 和 CIRCLE。在轴插补运动 (MOVE) 中，该值将被忽略。

轴组必须使用机器人模型 (model !=1) 进行定义。

该值不得大于 VELOCITYMAXTRANS。

系统始终采用两者中较小的值，并向用户发送通知消息。

缩写形式

<ROBOT>.vftran

语法

<ROBOT>.vftran=<numeric expression>

域

ROBOT

参数

< ROBOT >: 任何有效的机器人

<numeric expression>: Double，0.1 到 Maxdouble

限制

写入，仅非独立属性

示例

MoveS Scara #{10,20,30,0} vftran=600

另请参见

CIRCLE, MOVES

VELOCITYMAX

说明

定义运动元素的最大允许速度。系统将运动元素速度命令限制为该值。

在实际使用中，它受物理参数（特别是最大电机速度）的限制。

缩写形式

<element>.Vmax

语法

<element>.VelocityMax = <value>

域

ELEMENT

参数

< element >: 有效的运动元素

<value>: Double，大于零

限制

只写。要在任务中设置该值，必须将运动元素连接到该任务（使用 ATTACH 命令）。

示例

轴

A1.Vmax = 300

机器人

SCARA.VMax=300

另请参见

VELOCITYSCALE, ATTACH, VELOCITYOVERSPEED

VELOCITYMAXROT

说明

定义机器人的最大旋转速度，用于 VFROT 和 VROT。

该值仅限制笛卡尔运动插补（MOVES、CIRCLE）。此参数不影响轴插补的运动 (MOVE)。

缩写形式

<ROBOT>.vmrot

语法

<ROBOT>.vmrot=<numeric expression>

域

ROBOT

参数

< ROBOT >: 任何有效的机器人

<numeric expression>: Double，0.1 到 Maxdouble

限制

读/写，仅独立属性

示例

vmrot = 6000

vmrot = 6000

另请参见

VELOCITYROT, CIRCLE, MOVES

VELOCITYMAXTRANS

说明

定义机器人的最大平移速度，用于 VFTRAN 和 VTRAN。

该值仅限制笛卡尔运动插补（MOVES、CIRCLE）。

此参数不影响轴插补的运动 (MOVE)。

缩写形式

<ROBOT>.vmtran

语法

<ROBOT>.vmtran=<numeric expression>

域

ROBOT

参数

< ROBOT >: 任何有效的机器人

<numeric expression>: Double，0.1 到 Maxdouble

限制

读/写，仅独立属性

示例

vmtran = 6000

vmtran = 6000

另请参见

VELOCITYTRANS, CIRCLE, MOVES

VELOCITYOVERRIDE

说明

VELOCITYOVERRIDE 通过将运动元素实际速度乘以指定的百分比值来修改该速度。

与其他运动属性不同的是，它立即生效。

实际速度变化率受轴减速度值限制。SYSTEM.VELOCITYOVERRIDE 也会影响运动元素实际速度，并用作附加乘数。

更改 VELOCITYOVERRIDE 需要 4 个周期（如果轴正在运动），并将任务执行延迟此时间量。

轴作为轴组的一部分进行运动时，轴速度百分比没有影响。

注意

VELOCITYOVERRIDE 不会影响从轴，因为该轴不通过轨迹生成器运动。

缩写形式

<element>.Vord

语法

<element>.VelocityOverride = <value>

域

ELEMENT

参数

<element>: 任何有效的运动元素

<value>: Double，0.1 到 100

限制

只写

实际速度变化率受轴减速度值限制。

示例

```
SCARA.VelocityOverride=20      'Set VORD to 20%
```

另请参见

VELOCITYOVERRIDE

VELOCITYOVERSPEED

说明

设置或查询轴的 VELOCITYOVERSPEED 值。VELOCITYOVERSPEED 定义绝对电机速度限制。超出此限制后，将生成一个错误，并立即停止该轴。停止时，不会检查超速条件。通过程序错误处理器可采取进一步措施。

注意

VelocitySafetyLimit 和 VOSPD 应大于：1000/ser.cycletime/VFac。如果 vfac 设置过低，常规速度值可能会触发这些速度限制。

缩写形式

<axis>.VOSPD

语法

<axis>.VelocityOverspeed = <value>

?<axis>.VelocityOverspeed

域

AXIS

参数

<value>: Double，大于零。

限制

要在任务中设置该值，必须将轴连接到该任务（使用 ATTACH 命令）。

示例

A1.VelocityOverspeed = a1.vmax

另请参见

ATTACH, VELOCITYMAX

VELOCITYROT

说明

定义机器人的旋转（方向）速度。与 VTRAN 一起，定义笛卡尔运动的巡航速度。

该值仅在两个运动命令中使用：MOVES 和 CIRCLE。在轴插补运动 (MOVE) 中，该值将被忽略。

轴组必须使用机器人模型 (model !=1) 进行定义。

该值不得大于 VELOCITYMAXROT。

系统始终采用两者中较小的值，并向用户发送通知消息。

缩写形式

<ROBOT>.vrot

语法

<ROBOT>.vrot=<numeric expression>

域

ROBOT

参数

< ROBOT >: 任何有效的机器人

<numeric expression>: Double，0.1 到 Maxdouble

限制

读/写，独立/非独立属性

示例

vrot = 6000

vrot = 6000

另请参见

VELOCITYMAXROT, CIRCLE, MOVES

VELOCITYROTVALUE

说明

返回机器人工具尖端的旋转速度值（度/秒），这是一个只读的值。

缩写形式

<ROBOT>.VELOCITYROTVALUE

语法

?<ROBOT>.VELOCITYROTVALUE

域

ROBOT

参数

<ROBOT>: 任何有效的机器人

返回值

返回机器人工具尖端的旋转速度值（度/秒）

<return value>: DOUBLE，0 到 MaxDouble

限制

只读

示例

?scara.velocityrotvalue

?scara.velocityrotvalue

另请参见

CIRCLE, MOVES

VELOCITYSCALE

说明

此属性指定期望的巡航速度与此类型运动的最大可能速度相比的百分比。运动生成器尝试在运动命令期间达到此速度。

运动生成器达到该值的能力受 ACCELERATION、DECELERATION、SMOOTHFACTOR 和最终位置值的限制。

此属性在运动命令内使用以覆盖永久值。

仅当 VelocitySettings 设置为 1 时激活。

缩写形式

<element>.Vscale

语法

<element>.VelocityScale = <expression>

?<element>.VelocityScale

域

ELEMENT

参数

< element >: 有效的运动元素

< expression >: Double，0.1 到 100

限制

要在任务中设置该值，必须将运动元素连接到该任务（使用 ATTACH 命令）。

示例

轴

A1.VelocityScale =30

Move A1 TargetPos Vscale = 20

机器人

VelocityScale =40

Move SCARA {100, 200} Vscale = 22

另请参见

VELOCITYMAX, ATTACH, ACCELERATIONSCALE, JUMP

VELOCITYTRANS

说明

定义机器人的平移速度。与 VROT 一起，定义笛卡尔运动的巡航速度。

该值仅在两个运动命令中使用：MOVES 和 CIRCLE。在轴插补运动 (MOVE) 中，该值将被忽略。

轴组必须使用机器人模型 (model !=1) 进行定义。

该值不得大于 VELOCITYMAXTRANS。

系统始终采用两者中较小的值，并向用户发送通知消息。

缩写形式

<robot>.vtran

语法

<robot>.vtran=<numeric expression>

域

ROBOT

参数

< ROBOT >: 任何有效的机器人

<numeric expression>: Double，0.1 到 Maxdouble

限制

读/写，独立/非独立属性

示例

vtran = 6000

另请参见

VELOCITYMAXTRANS, CIRCLE, MOVES, JUMP3, JUMP3CP, VELOCITYTRANSFEEDBACKVALUE, VELOCITYTRANSVALUE

VELOCITYTRANSFEEDBACKVALUE

说明

返回机器人 TCP 反馈线性速度的大小。

此属性等同于采用 VELOCITYFEEDBACKCARTESIAN 前三个组成部分的标准。

缩写形式

<ROBOT>.vtfbk

语法

<ROBOT>.VelocityTransFeedbackValue

域

ROBOT

参数

<ROBOT>: 任何有效的机器人

返回值

返回机器人 TCP 反馈线性速度的大小

<return value>: Double，0 到 MaxDouble

限制

只读

示例

?Scara.vtfbk

?Scara.vtfbk

另请参见

VELOCITYTRANS

VELOCITYTRANSVALUE

说明

返回机器人工具尖端的平移速度值（毫米/秒）。这是一个只读的值。

缩写形式

< ROBOT >.velocitytransvalue

语法

?< ROBOT >.velocitytransvalue

域

ROBOT

参数

<ROBOT>: 任何有效的机器人

返回值

返回机器人工具尖端的平移速度值（毫米/秒）

<return value>: Double，0 到 MaxDouble

限制

只读。

示例

?SCARA.velocitytransvalue

?SCARA.velocitytransvalue

另请参见

VELOCITYTRANS, CIRCLE, MOVES

VGETJOBDATA

说明

如果 VGetJobStatus 命令返回 GET_DATA，您可以使用 VGetJobData 命令从视觉站获取响应数据。

VGetJobData 命令会返回响应数据的数组。例如，如果视觉站发送 JSON 消息 {"Data": [{"X":1.0, "Y":2.0, "Z":3.0}, {"X":4.0, "Y":5.0, "Z":6.0}]}，VGetJobData 会返回如下数组：

"1.0" "2.0" "3.0"

"4.0" "5.0" "6.0"

语法

VGetJobData (<StationName>, <JobName>, <Dimension> , <DataNum> , <Data>)

参数

<StationName>: 任何有效的视觉站，String

<JobName>: 属于视觉站的任何有效作业，String

<Dimension>: 存储数据反馈的维度，Long

<DataNum>: 存储数据反馈的轴组编号，Long

<Data>: 存储数据反馈，二维字符串数组

返回值

返回视觉任务数据。

<Dimension>: 存储数据反馈的维度，Long

<DataNum>: 存储数据反馈的轴组编号，Long

<Data>: 存储数据反馈，二维字符串数组

示例

? VGetJobData ("S1", "TEMPLATEJOB", Dim, Num, SDATA)

另请参见

VGETJOBERROR, VGETJOBSTATUS, VRUNJOB

VGETJOBERROR

说明

如果 VGetJobStatus 命令返回 GET_ERROR，您可以使用 VGetJobError 命令从视觉站获取响应错误。

语法

VGetJobError (<StationName>, <JobName> , <ERROR>)

参数

<StationName>: 任何有效的视觉站，String

<JobName>: 属于视觉站的任何有效作业，String

<ERROR>: 存储错误字符串，String

返回值

返回函数执行状态。

<ERROR>: 存储错误字符串。

示例

```
RET = VGetJobStatus("S1", "PositionAdjustment2D", MYSTATUS,10000)
select RET
case GET_STATUS
    print MYSTATUS ‘ deal with Respond Status
case GET_DATA
    LOOPCTL = false
    ?VGetJobData(“S1”, “PositionAdjustment2D”, MYDIM,MYNUM, ASDATA)
case GET_ERROR
    LOOPCTL = false
    ?VGetJobError(“S1”, “PositionAdjustment2D”, MYERR)
end select
```

另请参见

VGETJOBDATA, VGETJOBSTATUS, VRUNJOB

VGETJOBSTATUS

说明

运行作业后，您可以使用 VGetJobStatus 命令获取响应状态。

在视觉站发送响应状态、响应数据或响应错误消息前，程序都将阻止。

语法

VGetJobStatus (<StationName>, <JobName>, <ERROR >, <Timeout>)

参数

<StationName>: 任何有效的视觉站，String

<JobName>: 属于视觉站的任何有效作业，String

<Timeout>: TCP 连接的超时时间，Long，单位为毫秒

<ERROR>: 存储错误字符串，String

返回值

返回视觉任务状态。

<ERROR>: 存储错误字符串。

示例

```
?VRunJob(“S1”, “PositionAdjustment2D”, 5000)
LOOPCTL = true
while LOOPCTL
    RET = VGetJobStatus(“S1”, “PositionAdjustment2D”,
    MYSTATUS, 10000)
    select RET
    case GET_STATUS
        print MYSTATUS ‘ deal with Respond Status
    case GET_DATA
        LOOPCTL = false
    case GET_ERROR
        LOOPCTL = false
    end select
end while
```

另请参见

VGETJOBDATA, *VGETJOBERROR*, *VRUNJOB*

VOPENTCP

说明

如果您想通过 TCP IP 协议发送用户定义的字符串，则可以通过调用“VOpenTcp”手动打开套接字。

语法

VOpenTcp (<StationName>)

参数

<StationName>: 任何有效的视觉站，String

示例

VOpenTcp ("S1")

另请参见

VRUNJOB, VCLOSETCP

VPIXELTOPOS

说明

设置视觉校准后，您可以使用“VPixelToPos”将像素坐标转换为机器人坐标。在校准所选视觉站之前，无法使用此命令。

语法

VPixelToPos (<StationName>, <Pix_X> , <Pix_Y> , <Pos_X> , <Pos_Y>)

参数

<StationName>: 任何有效的视觉站，String

< Pix_X >: 像素坐标 X 轴上的给定位置，Double

< Pix_Y >: 像素坐标 Y 轴上的给定位置，Double

< Pos_X >: 返回笛卡尔坐标 X 轴上的位置，Double

< Pos_Y >: 返回笛卡尔坐标 Y 轴上的位置，Double

返回值

< Pos_X >: 返回笛卡尔坐标 X 轴上的位置，Double

< Pos_Y >: 返回笛卡尔坐标 Y 轴上的位置，Double

示例

VPixelToPos("S1",PIX,PIY,PX,PY)

另请参见

VRUNJOB, VRUNJOBFULL, VRECEIVEDATA

VRECEIVEDATA

说明

此命令可用于通过 TCP IP 协议接收用户定义的字符串。执行命令 VOpenTcp 之前，无法使用此命令。

语法

VReceiveData (<StationName>, <ReceiveData> , <RecState>, <Timeout>)

参数

<StationName>: 任何有效的视觉站，String

< ReceiveData >: 返回从视觉站接收的数据，String

< RecState >: 返回命令发送状态，Long

< Timeout> : 设置当未收到任何数据的最大等待时间，单位为毫秒，Long

返回值

< ReceiveData >: 返回从视觉站接收的数据，String

< RecState >: 返回命令发送状态，Long

示例

```
VOpenTcp(“VS1”)
VSendCmd("VS1","TrigerStart",lState)
If lState then
    VReceiveData("VS1",SDATA,lState, 1000)
End if
```

另请参见

VPIXELTOPOS, VRUNJOB

VRUNJOB

说明

此命令可用于连接到视觉站和运行视觉任务。控制器将解析视觉站和作业信息，并与视觉站建立连接。

语法

VRunJob (<StationName>, <JobName>, <Timeout>)

参数

<StationName>: 任何有效的视觉站，String

<JobName>: 属于视觉站的任何有效作业，String

<Timeout>: TCP 连接的超时时间，Long，单位为毫秒

返回值

返回函数执行状态。

<return value>: Long

示例

?VRunJob("S1", "TEMPLATEJOB", 5000)

另请参见

VRUNJOBFULL, VRECEIVEDATA, VGETJOBDATA, VGETJOBSTATUS, VGETJOBERROR, VOPENTCP, VPIXELTOPOS, VSENDCMD, VSTOPJOB

VRUNJOBFULL

说明

此命令使您无需处理响应状态即可获取响应数据或响应错误。该命令不断调用“VGetJobStatus”，直到其返回响应数据或响应错误。

此命令集成所有相关命令，然后按“VRunJob”、多个“VGetJobStatus”、“VGetJobData”或”VGet-JobError”的顺序执行。

“VRunJobFull”命令是没有详细信息的简化命令。如果要定义与作业状态相关的操作，则不要使用该命令。

语法

VRunJobFull (<StationName>, <JobName>, <Timeout>, <Dimension> , <DataNum> , <Data>, <ERROR>)

参数

<StationName>: 任何有效的视觉站，String

<JobName>: 属于视觉站的任何有效作业，String

<Timeout>: TCP 连接的超时时间，Long，单位为毫秒

<Dimension>: 存储数据反馈的维度，Long

<DataNum>: 存储数据反馈的轴组编号，Long

<Data>: 存储数据反馈，二维字符串数组

<ERROR>: 存储错误字符串，String

返回值

返回函数执行状态。

<Dimension>: 存储数据反馈的维度，Long

<DataNum>: 存储数据反馈的轴组编号，Long

<Data>: 存储数据反馈，二维字符串数组

<ERROR>: 存储错误字符串，String

示例

?VRunJobFull ("CAMERA", "GET_POINTS", 5000, LDIM, LNUM, SDATA, SERROR)

另请参见

VRUNJOB, VCLOSETCP, VPIXELTOPOS, VSTOPJOB

VSENDCMD

说明

此命令用于通过 TCP IP 协议发送用户定义的字符串。

执行命令 VOpenTcp 之前，无法使用此命令。

语法

VSendCmd (<StationName>, <Command> , <CmdState>)

参数

<StationName>: 任何有效的视觉站，String

< Command >: 命令消息将发送到视觉站，String

< CmdState >: 返回命令发送状态，Long

返回值

< CmdState >: 返回命令发送状态

示例

VOpenTcp(“VS1”)

VSendCmd("VS1","TrigerStart",lState)

另请参见

VRUNJOB

VSMODE

说明

VSmode（振动抑制模式）是受支持的 MotionMode 相关命令之一。它允许通过将值设置为“开”或“关”来相应地启用和禁用振动抑制功能。通过适当设置相关参数，此功能可减少残余振动幅度和时间。

注意

您必须先在界面中完成对于负载的创建和设置并启用负载，才能够通过该指令完成振动抑制模式的启用。另外，该指令仅针对下一条运动生效，因此，您需要在每一条需要使用振动抑制的运动指令之前都插入 VSmode 指令。

如果您需要更详细的使用方法，请参考用户手册。

语法

VSMode = < value >

参数

<value>: Long, On – 开启 vsmode

Off – 关闭 vsmode

域

ROBOT

示例

SCARA.VSmode = On

MOVE SCARA {0,0,0,0}

另请参见

MOVE , JUMP, CIRCLE, MOVES, JUMP3, JUMP3CP

VSTOPJOB

说明

此命令用于停止一个正在运行的视觉站作业。

语法

VStopJob (<StationName>, <JobName>)

VStopJob (<StationName>, <JobName>)

参数

<StationName>: 任何有效的视觉站，String

<JobName>: 属于视觉站的任何有效作业，String

示例

VStopJob ("S1", "TEMPLATEJOB")

VStopJob ("S1", "TEMPLATEJOB")

另请参见

VRUNJOBFULL, VRUNJOB

WAIT

说明

该命令会停止程序直到一些条件被满足，其中条件<condition>包含许多类型。条件<Condition> 由 <expression> <operator> <value> 组成，并且只支持 IO 命令.

该命令可以设置超时时间， 如果超时时间未设置，程序会一直等待，直到设置的条件被满足。

数字 IO 条件:

<expression> 包含信号变量

<operator> 包含： '=', '<>'

<value> 是整型数据

模拟 IO 条件:

<expression> 包含信号变量

<operator> 包含： '=', '>', '<', '>=', '<=' and '<>'

<value> 包含整型和浮点型数据

边沿触发 IO 条件:

<expression> 包含信号变量

<operator> 包含： '='

<value> 包含： 'edge', 'on+' and 'off-'

语法

wait <condition> { and/or <condition> } { tmout = <timeout> }

参数

<condition>: 与 I/O 相关的条件表达式, 包括数字信号和模拟信号。 条件表达式可以用'and'或者'or'相互组合

<timeout>: Long, 从 1 到 86400000

示例

wait singal2 = 1 and signal2 = 0

wait signal3> 1 or signal4 < 2 tmout = 10000

wait signal5= On+

另请参见

GETIO , SETIO

WAITFORMOTION

说明

此命令让程序在当前执行的运动完成前等待，且仅在当前执行的运动完成后才执行下一行。

语法

WaitForMotion <element>

参数

<element>: 任何有效的运动元素

限制

需要连接。

示例

```
Attach SCARA
Move SCARA {10,0,0,0}
WaitForMotion SCARA
Move SCARA {20,0,0,0}
Detach SCARA
```

另请参见

MOVE, STARTTYPE, SLEEP

WHILE_..._END_WHILE

说明

While 和 End While 关键字可分隔 WHILE 循环。

While 循环用于在指定条件保持为真时执行一段代码。

该条件将在 WHILE 结构内部的语句开始执行之前进行一次判断，因此，假如 WHILE 循环第一次条件判断为假，那么内部的语句将一次都不会被执行。WHILE 内部可以包含任意数量的语句，当然，也可以不添加任何待执行语句。如果未包含任何语句，则 WHILE…End WHILE 可以被用来充当一个延时操作。

注意

不要使用不包含任何语句或仅包含简单操作语句的循环结构，如果您一定要如此使用，请至少额外插入一条 SLEEP 1 指令，避免出现 CPU 过载，导致系统异常。

语法

While <condition>

<code to execute as long as condition is true>

End While

参数

< condition >:Long，while 条件表达式

示例

```
While A1.IsMoving = 1 ‘wait for profiler to finish
    sleep 20
End While
While A2.VelocityFeedback < 1000
    Print "Axis 2 Velocity Feedback still under 1000"
End While
```

另请参见

FOR_..._NEXT, DO_..._LOOP, PROGRAM_..._END_PROGRAM

WITH

说明

此命令设置默认运动元素（轴或轴组）。此后，对运动元素进行操作而不必显式指示运动元素名称。

With 语句的使用方式有 4 种：

With - Config.prg - 在 Config.prg 文件中设置为默认运动元素 (DME) 的元素，用作整个系统的 DME。因此，更改/停止 DME 的唯一途径是“终止” Config.prg 任务，然后手动更改默认值。

如果使用 Local With 语句 - DME 将成为在局部语句中声明的运动元素，并在任务完成后改回“Config.prg With”。

Local With - 设置用于局部任务的 DME。与“Config With”不同 - 使用 local With 语句需要使用 With <element>.....End With 方法。

这种使用方式将当前 DME（在 Config.prg 或在终端中声明）替换为新 DME，直到任务完成。

Terminal With - 这种使用方式指定用于运行时更改。在终端声明 DME 后，只要用户没有通过终端发送“End With”命令，所选运动元素就将用作 DME。在终端范围内使用的 With 命令不会影响任何程序，反之亦然。

WithGlobal - 使用“WithGlobal”命令创建的 DME 将替换 With - Config.prg 而不关闭系统，或在没有 DME 时创建新 DME。有关更多信息 - WITHGLOBAL

语法

With <element name>

参数

<element name>: 任何有效的运动元素

限制

无法嵌套 With 命令。

从 WITH 块内调用的子程序不会继承默认运动元素，因此该元素不会在子程序内定义，并且将标记错误 (Local With)。

WITH 块内不允许 GOTO 命令；将标记错误 (Local With)。

示例

A1.VMax=5000

A1.Vord=5000

A1.VCruise=3000

A1.PEMax=10

A1.PESettle=0.01

Move A1 100

可以简化为使用：

With A1

VMax=5000

Vord=5000

```
VCruise=3000
PEMax=10
PESettle=0.01
Move 100
End With
```

另请参见

WITHGLOBAL, PROGRAM_ ... _END_PROGRAM

WITHGLOBAL

说明

此命令在全局范围内设置了默认运动元素（轴或轴组），

就像“WITH”语句一样，允许直接对运动元素进行操作而不必显式指示运动元素名称。

默认运动元素仅在 **WithGlobal** 语句后存在，可通过三种方式终止：

使用新 WithGlobal 语句，在每个范围中声明。

在终端内使用“reset all”命令。

使用仅在局部任务运行时有效的 Local With。

使用新 WithGlobal 语句，在每个范围中声明。

在终端内使用“reset all”命令。

使用仅在局部任务运行时有效的 Local With。

注意

此命令通常在 CONFIG.PRG 文件中设置。

与使用 local/Terminal WITH 语句（需要 With.....→ End With 方法）不同的是，使用“WithGlobal”语句时，无需声明“End WithGlobal”。

语法

WithGlobal <element name>

参数

<element name>: 任何有效的运动元素

限制

无法嵌套 WithGlobal 命令，因为没有 End WithGlobal 语句。

示例

```
A1.VMax=5000
A1.Vord=5000
A1.VCruise=3000
A1.PEMax=10
A1.PESettle=0.01
Move A1 100
```

可以简化为使用：

```
WithGlobal A1
   VMax=5000
      Vord=5000
      VCruise=3000
      PEMax=10
      PESettle=0.01
```

```
Move 100
```

另请参见

PROGRAM ... END_PROGRAM, WITH

WITHPLS

说明

作为不可单独使用的属性，指定路径 PLS。

WithPLS 参数指定将由运动命令激活的 PLS 开关。

指定的 PLS 将在给定运动的执行期间启用，并在运动结束时禁用。

语法

{MOVE|MOVES|CIRCLE|...} {<element name>} {nodal assignments} {WithPLS= <pls name>} ...

参数

<element name>: 任何有效的运动元素

<pls name>: 位置触发器的名称

限制

PLS 位置数据必须是单调的。

作为不可单独使用的属性，一行最多可以给定 16 个 PLS

示例

Move SCARA {0, 0, 0, 0} withpls = pls1

另请参见

JUMP, JUMP3, JUMP3CP, MOVE, MOVES

WORKPIECE

说明

工件坐标系设置了指定位置用于系统工件坐标变换。

它定义工件坐标系相对于 MACHINETABLE FRAME 参考的位置和方向。

默认的基坐标系变换是 NULL 变换，可表示为 SCARA.WorkPiece = #{0,0,0,0}

语法

WorkPiece = <robot location point>

域

ROBOT

参数

<robot location point>: Location

返回值

返回世界坐标系中的工件坐标系位置

<return value>: Location

示例

Scara .WorkPiece = #{90, 180, 0, 0}

?Scara.MachineTable:WorkPiece:Base:Tool

{90 , 90 , 90 , 180}

另请参见

BASE, TOOL, USERFRAME

XMAX

说明

该值在机器人笛卡尔运动（MOVES、CIRCLE）的预计算阶段使用。Xmin、Xmax、Ymin、Ymax、Zmin、Zmax 的值允许以笛卡尔坐标形式定义安全工作区。这些限制将被检查。

语法

<ROBOT>.XMax = <numeric expression>

域

ROBOT

参数

<ROBOT>: 任何有效的机器人

<numeric expression>: Double，0.1-maxdouble

限制

读/写，仅独立属性

示例

Xmax = 60

另请参见

XMIN, YMAX, ZMAX, MOVES

XMIN

说明

该值在机器人笛卡尔运动（MOVES、CIRCLE）的预计算阶段使用。Xmin、Xmax、Ymin、Ymax、Zmin、Zmax 的值允许以笛卡尔坐标形式定义安全工作区。这些限制将被检查。

语法

<ROBOT>.XMin = <numeric expression>

域

ROBOT

参数

<ROBOT>: 任何有效的机器人

<numeric expression>: Double，0.1-maxdouble

限制

读/写，仅独立属性

示例

Xmin = 60

另请参见

YMIN, XMAX, ZMIN, MOVES

YMAX

说明

该值在机器人笛卡尔运动（MOVES、CIRCLE）的预计算阶段使用。Xmin、Xmax、Ymin、Ymax、Zmin、Zmax 的值允许以笛卡尔坐标形式定义安全工作区。这些限制将被检查。

语法

<ROBOT>.Ymax = <numeric expression>

域

ROBOT

参数

<ROBOT>: 任何有效的机器人

<numeric expression>: Double，0.1-maxdouble

限制

读/写，仅独立属性

示例

Ymax = 60

另请参见

XMAX, YMIN, ZMAX, MOVES

YMIN

说明

该值在机器人笛卡尔运动（MOVES、CIRCLE）的预计算阶段使用。Xmin、Xmax、Ymin、Ymax、Zmin、Zmax 的值允许以笛卡尔坐标形式定义安全工作区。这些限制将被检查。

语法

<ROBOT>.YMin = <numeric expression>

域

ROBOT

参数

<ROBOT>: 任何有效的机器人

<numeric expression>: Double，0.1-maxdouble

限制

读/写，仅独立属性

示例

Ymin = 60

另请参见

XMIN, YMAX, ZMIN, MOVES

ZMAX

说明

该值在机器人笛卡尔运动（MOVES、CIRCLE）的预计算阶段使用。Xmin、Xmax、Ymin、Ymax、Zmin、Zmax 的值允许以笛卡尔坐标形式定义安全工作区。这些限制将被检查。

语法

<ROBOT>.ZMax = <numeric expression>

域

ROBOT

参数

<ROBOT>: 任何有效的机器人

<numeric expression>: Double，0.1-maxdouble

限制

读/写，仅独立属性

示例

Zmax = 60

另请参见

XMAX, YMAX, ZMIN, MOVES

ZMIN

说明

该值在机器人笛卡尔运动（MOVES、CIRCLE）的预计算阶段使用。它用于防止机器人与它自己的主体（基柱）发生碰撞。

定义“最低”机器人位置的 Z 坐标。不允许任何点低于此平面。

连接的设备和参数未纳入机器人设置可能会显著减少机器人工作区！

注意

配置新机器人之前，始终检查 ZMIN 值。

语法

<ROBOT>.ZMin = <numeric expression>

域

ROBOT

参数

<ROBOT>: 任何有效的机器人

<numeric expression>: Double，0.1-maxdouble

限制

读/写，仅独立属性

示例

Zmin = 60

另请参见

XMIN, YMIN, ZMAX, MOVES

范例

通过以下几个经典范例, 使用户快速学习相关常用指令。

运动指令范例

```
'*************************************************************************************************************
'Define global variables
common shared Pick as location of xyzr   = #{325,175,-100,0}
common shared Pick2 as location of xyzr = #{325,75,-100,0}
common shared Place as location of xyzr = #{325,-175,-100,0}
common shared HomePos as joint of xyzr = {0,0,-10,0}
common shared lProductID as long = 1

'Local variables
dim shared dLimZ as double = 0
dim shared xAllowPlace as long

program
   'Internal variable
   dim lSpeed as long = 50
   dim Pos1 as location of xyzr
   dim PickUp as location of xyzr
   dim PrevClock as long

   call Init
   'Go home point
   if j3.pfb > dLimZ then
      move j3 dLimZ VScale = 10 'Z move to dLimZ
   end if

   Pos1 = toCart(SCARA, HomePos)   'joint point convert to location point
   if getDist(Pos1) > 10 then
      'do something
   end if

   'get current arm type; 0 - singular point(J2.pcmd = 0); 1- left type; 2 - right type
   if ARMFBK = 1 then
      'do something
   else '
      'do something
   end if
```

```
jump HomePos limz = dLimZ VScale = 10 'jump to home position, no blend
waitformotion SCARA                   'wait move finish

while 1
   sleep 10
   PrevClock = sys.clock   'start the time
   select case lProductID
      case 1
         if getArm(Pick) <> 1 then
            setArm(Pick,1) 'set pick arm to left type
         end if
         PickUp = #{Pick{1}, Pick{2}, -10, Pick{4};1}
         move PickUp VScale= 10 blend = 20
         move Pick VScale= 10
         waitformotion SCARA
         sleep 100
         PickUp = #{Pick{1}, Pick{2}, -20, Pick{4};1}
         move PickUp VScale= 10 blend = 20
         lProductID = 2
      case 2
         if getArm(Pick2) <> 2 then
            setArm(Pick2,2) 'set pick arm to right type
         end if
         jump Pick2 archno = 1 blend = 20 Vscale= 20
         waitformotion SCARA
         sleep 100
         PickUp = #{Pick2{1}, Pick2{2}, -20, Pick2{4};2}
         move PickUp VScale= 10 blend = 20
         lProductID = 1
      case else
         'do something
   end select

   if xAllowPlace then
      setArm(Place,2)
      jump Place archno = 1 blend = 10 Vscale= 20
      waitformotion SCARA
```

```
        moves #{Place{1},Place{2},dLimZ,Place{4};2} blend = 20 Vscale= 20
      end if
      waitformotion SCARA
      print "Run time is ",sys.clock - PrevClock
   end while

   detach

end program
'*************************************************************************************************************
sub Init
   power_on()
   Attach SCARA
   vord = 100
   dLimZ = 0
   xAllowPlace = true
end sub
'*************************************************************************************************************
'Get the distance between two points (setpoint and pos)
function getDist (byval Pos as location of xyzr) as double
   getDist = distl(setpoint,Pos)
end function
'*************************************************************************************************************
```

字符串处理范例

```
'*************************************************************************************************************
dim shared sVisionData as string
program
    dim locPos as location of xyzr
    dim sRetVal as string
    dim sArrStr[10] as string
    dim lGetNumber      as long
    dim jPos as joint of xyzr
    dim lLocation as long
    dim i as long

    'Get the robot's current position
    call getCurrentPosition

    'Get vision data
    call getVisionData

    lGetNumber = splitstr ( sVisionData,",",sArrStr ) 'split string by comma, the result is placed in array
sArrPoint
    for i = 1 to lGetNumber
        print "sArrStr[" + STR$(i) + "]=",sArrStr[i]
    next
    'convert string type to double type
    locPos{1} = val(sArrStr[1])
    locPos{2} = val(sArrStr[2])
    locPos{3} = val(sArrStr[3])
    locPos{4} = val(sArrStr[4])

    'check the point whether reachable
    try
        jPos = toJoint(SCARA, locPos) 'check whether reachable
        print "the point is reachable"
    catch else
        print "the point is unreachable"
    end try
```

```
    sVisionData = RIGHT$(sArrStr[5],1) 'get value of arm
    ' lLocation = INSTR(sArrStr[5],"arm=")
    ' sVisionData = mid$(sArrStr[5],lLocation + LEN("arm="), 1)

    'set arm type
    locPos.arm = val(sVisionData)

    attach
    power_on()
    move locPos vscale = 10
    waitformotion SCARA

    print `"robot is arrive the location"`; ' result is "robot is arrive the location"
    print `"robot is arrive the location"`  ' result is "robot is arrive the location"\r\n

end program
'*****************************************************************************************************
sub getVisionData
    "We can get data from the host computer or PLC through TCP/IP communication.
    'More detailed use of TCP,can see the template program <DEMO_TCP_SERVER> or
     <DEMO_TCP_CLIENT>.
    'Here we simulate and get the data from TCP/IP.
    sVisionData = "325.2,-175.1,-10,90,arm=1,vscale=100"

end sub
'*****************************************************************************************************
'Get the robot's current position
sub getCurrentPosition
    dim sCurrentCmdLocPos as string
    dim sX as string
    dim sY as string
    dim sZ as string
    dim sA as string

    print "Current command location posion is ",setpoint
    print "Current X command value is ",setpoint{1}

    print "Current feedback location posion is ",here
```

```
  print "Current X feedback value is ",here{1}

  print "Current command joint posion is ",pcmd
  print "Current J2 command posion is ",pcmd{2}

  print "Current feedback joint posion is ",pfb
  print "Current J2 feedback posion is ",pfb{2}

  'convert to string, keep two decimals
  sX = STRD$(setpoint{1},"%.2f")
  sY = STRD$(setpoint{2},"%.2f")
  sZ = STRD$(setpoint{3},"%.2f")
  sA = STRD$(setpoint{4},"%.2f")

  sCurrentCmdLocPos = "X=" + sX + ",Y=" + sY + ",Z=" + sZ + ",A="  + sA
  print sCurrentCmdLocPos

end sub
'*********************************************************************************************************
```

数据转换范例

```
'***************************************************************************************************************
program

   dim lData[16] as long
   dim dData[4] as double
   dim locPos   as location of xyzr

   dData[1] = 225.2
   dData[2] = -175.1
   dData[3] = -10
   dData[4] = 90

   '-----------------------------------------------------------------
   'convert float(32) type to long type
   FloatToLong(dData, lData, 4, 1)'1- big Endian; 0- little endian

   'lData[*] can be mapped to a analog output signal address, transmitted to the PLC
   ?lData[1] '17249
   ?lData[2] '13107

   ?lData[3] '49967
   ?lData[4] '6554

   ?lData[5] '49440
   ?lData[6] '0

   ?lData[7] '17076
   ?lData[8] '0

   '-----------------------------------------------------------------
   'convert long type to float(32) type
   'we can get lData[*] value from a analog input signal, this signal map to BUS address.
   LongToFloat(lData, dData, 4, 1)'1- big Endian; 0- little endian

   'get float data, move to this point
   locPos{1} = dData[1]' result is 225.2
```

```
locPos{2} = dData[2]' result is -175.1
locPos{3} = dData[3]' result is -10
locPos{4} = dData[4]' result is 90

'-------------------------------------------------------------------
'convert double(64) type to long type
DoubleToLong(dData, lData, 4, 1)'1- big Endian; 0- little endian

'lData[*] can be mapped to a signal, transmitted to the PLC
?lData[1] ' result is 16492
?lData[2] ' result is 9830
?lData[3] ' result is 24576
?lData[4] ' result is 0

?lData[5] ' result is 49253
?lData[6] ' result is 58163
?lData[7] ' result is 16384
?lData[8] ' result is 0

?lData[9] ' result is 49188
?lData[10] ' result is 0
?lData[11] ' result is 0
?lData[12] ' result is 0

?lData[13] ' result is 16470
?lData[14] ' result is 32768
?lData[15] ' result is 0
?lData[16] ' result is 0

'-------------------------------------------------------------------
'convert long type to double(64) type
'we can get lData[*] value from a analog input signal, this signal map to BUS address.
LongToDouble(lData, dData, 4, 1) '1- big Endian; 0- little endian

?dData[1]' 225.2
?dData[2]' -175.1
?dData[3]' -10
?dData[4]' 90
```

```
end program
'*********************************************************************************************************
```

碰撞检测范例

功能说明：

通过规划速度和实际速度之差(速度偏差值)检测机器人与外部是否发生碰撞。此功能可以在碰撞发生时时减少机器人的损害，但不能完全避免损害。此外，这些功能无法用于保障人身安全目的。

为减少手臂或夹具末端由于与其外围环境碰撞导致的损害，提供两项功能：
- 碰撞检测功能
- 扭矩限制功能

碰撞检测功能：

开启后，通过实时监测机器人动作的速度偏差值(规划速度和实际速度之差)在因碰撞而超过设定的速度偏差阈值后，判定为碰撞发生，并立即停止机器人。

扭矩限制功能

设置伺服扭矩的输出最大值。

开启该功能后，检测的错误分类为两类。
1：3358：机器人检测到碰撞，驱动器被禁用。
2：除碰撞以外的其他错误
- 轴位置跟随误差相关错误
- 轴速度跟随误差相关错误
- 由于限制扭矩造成的相关错误
- 驱动层面相关错误

默认是扭矩没有限制，CollsionDetect 命令设置为关状态，重启控制器可恢复至默认值。

在发生碰撞时机器人受到的作用力大致可以分为两类：碰撞瞬间速度的冲击力和碰撞后由于电机扭矩输出造成的压紧力。碰撞检测功能和扭矩限制功能可以限制电机扭矩输出，从而有效减小碰撞后由于压紧力导致的损害。对于速度的冲击力无效果。

注意事项：

1. 为减小碰撞及误检测风险，碰撞检测打开后，继续程序时，缓存区里的运动(1-3 条)会低速运行。
2. 在手臂轨迹有变化时，或一些场景，会有概率导致意外触发检测，大致有以下几种场景：
   - 与外部环境有强力接触，如取放料时，与工装接触。
   - 扭矩值设置不合理。
   - 碰撞检测速度误差阈值设置过小时。
   - Auto模式下，Step（单步）运行，运动退化为不带blend的情形。
   - 急停或碰撞发生后，继续运行程序时，极小概率发生。
   - 轨迹运行速度与学习时的速度不一致。
   - 修改轨迹。

对于外部环境有强力接触的情形，可通过以下措施来降低误检测。
- 与外部环境有强接触时，降低加减速度。
- 降低速度。
- 设置正确的负载值。

其他情形可通过以下几个操作来降低误检测的发生。
- 适当调整扭矩限制值。
- 调整碰撞检测误差阈值。
- 重新学习。

步骤：

执行碰撞检测功能，需要执行以下步骤。

- 重置并初始化峰值扭矩和峰值速度误差值，执行ResetPeakValue。
- 确保已经关闭碰撞检测。
- 执行需要保护的运动路径，学习几遍轨迹动作，通过PeakTorque得到峰值扭矩，通过PeakVelErr得到最大速度误差值。
- 根据获得该路径的峰值扭矩，加上阀值，对SetDriveTorqueLimit进行赋值。
- 根据获得的最大速度误差，加上阀值，对SetvelErrThreshold进行赋值。
- 在机器人程序内打开碰撞检测。

如果机器人学习中暂停并继续，可能得到大于正常程序运行的扭矩值和速度误差值。在这种情况下，测量的峰值扭矩和速度误差值，将包含在测量中。

程序样例：

以下是一个学习及配置碰撞检测功能和扭矩限制功能的样例程序。

程序开始测量前，恢复扭矩最大值及关闭碰撞检测，在最开始的3次运行中，测量最大扭矩及最大速度误差值后，在测量值加上系数1.2/1.3 倍，第4次开始设置上限扭矩值、碰撞检测阈值并打开碰撞检测。

```
'*************************************************************************************************************
program
    dim lCount    as long = 3
    Attach SCARA
    power_on()
    ResetPeakValue() 'Clear peak torque and peak velocity error of all axis.
    SetDriveTorqueLimit(100, 100, 100, 100) 'Set torque limit 100
    SetvelErrThreshold(100,100,100,100)
    SCARA.CollisionDetect = off
    While true
        If lCount = 0 then
        waitformotion SCARA
        SetDriveTorqueLimit(J1.PeakTorque*1.2, J2.PeakTorque*1.2, J3.PeakTorque*1.3,
        J4.PeakTorque*1.3)
        SetvelErrThreshold(J1.PeakVelErr*1.2, J2.PeakVelErr*1.2, J3.PeakVelErr*1.3,
        J4.PeakVelErr*1.3)
        scara.CollisionDetect = on
        end if
        move {90,90,-100,0}
        move {-90,-90,-100,0}
        'move P1
        'move P2
        'move Pn
        move {0,0,0,0}
```

```
            waitformotion SCARA
            if lCount >= 0 then
                lCount = lCount - 1
            end if
        end while
end program
'***************************************************************************************************************
```

范围

下表指示每条命令的使用范围。
任务：可在用户程序或用户库中使用
终端：可以在终端中使用
函数：可以在函数或子程序中使用

项目	任务	终端	函数
ABS	√	√	√
ABSOLUTE	√	√	√
ACCELCMDCART	√	√	√
ACCELERATION	√	√	√
ACCELERATIONCOMMAND	√	√	√
ACCELERATIONERROR	√	√	√
ACCELERATIONFEEDBACK	√	√	√
ACCELERATIONMAX	√	√	√
ACCELERATIONMAXROT	√	√	√
ACCELERATIONMAXTRANS	√	√	√
ACCELERATIONROT	√	√	√
ACCELERATIONSCALE	√	√	√
ACCELERATIONTRANS	√	√	√
ACCEPT	√	-	√
ACOS	√	√	√
ARMCMD	√	√	√
ARMFBK	√	√	√
ARRAYSIZE	√	√	√
ASC	√	√	√
ASIN	√	√	√
ATAN2	√	√	√
ATN	√	√	√
ATTACH	√	-	√
ATTACHEDTO	√	√	√
ATTACHTO	√	√	√
ATTACHTO$	√	√	√
AVERAGELOAD	√	√	√
AXISLIST	-	√	-
BASE	√	√	√
BIN$	√	√	√
BLENDDISTANCE	√	√	√
BLENDENDPROTECTED	√	√	√
BLENDMETHOD	√	√	√
BLENDORIENTATION	√	√	√
BLENDPERCENTAGE	√	√	√
BLENDSTARTPROTECTED	√	√	√
CALL	√	-	√
CASTJOINT	√	√	√
CASTLOCATION	√	√	√
CASTPOINT	√	√	√
CHR$	√	√	√
CIRCLE	√	-	√
CLEARMSG	√	√	√
CLOCK	√	√	√
CLOSE	√	√	√
COMMON_OR_DIM_SHARED_OR_DIM_..._AS_LONG	√	√	√
COMMON_OR_DIM_SHARED_OR_DIM_..._AS_DOUBLE	√	√	√
COMMON_OR_DIM_SHARED_OR_DIM_..._AS_STRING	√	√	√

项目	任务	终端	函数
COMMON_OR_DIM_SHARED_OR_DIM_..._AS_JOINT_OF	√	√	√
COMMON_OR_DIM_SHARED_OR_DIM_..._AS_LOCATION_OF	√	√	√
COMMON_OR_DIM_SHARED_VAR_NAME_AS_NOTE_OR_ERROR_ASCII_NUMBER	√	√	√
COMMON_SHARED_..._AS_SEMAPHORE	√	√	√
COMMON_SHARED_OR_DIM_SHARED_OR_DIM_..._AS_TYPE	√	√	√
CONNECT	√	-	√
COS	√	√	√
CUMULATIVEPOSITION	√	√	√
CURRENTABSOLUTE	√	√	√
CURRENTTIME	√	-	√
DECELERATION	√	√	√
DECELERATIONMAX	√	√	√
DECELERATIONROT	√	√	√
DECELERATIONTRANS	√	√	√
DELETE	-	√	-
DELETE$	-	√	-
DELETESEM	√	√	√
DEST	√	√	√
DETACH	√	-	√
DETACHFROM	√	√	√
DETACHFROM$	√	√	√
DISABLETIMEOUT	√	√	√
DISTL	√	√	√
DISTR	√	√	√
DO_..._LOOP	√	-	√
DOUBLEMODE	√	√	√
DOUBLEMODEPERCENTAGE	√	√	√
ELEMENTID	√	√	√
ELEMENTNAME	√	√	√
ELEMENTSIZE	√	√	√
ELEMENTSTATUS	√	√	√
ERROR	√	√	√
ERRORHANDLERMODE	√	√	√
ERRORNUMBER	√	√	√
EVENTDELETE	√	-	√
EVENTLIST	-	√	-
EVENTOFF	√	√	√
EVENTON	√	√	√
EXP	√	√	√
FILESIZE	√	√	√
FILESIZE$	√	√	√
FOR_..._NEXT	√	-	√
FUNCTION_..._END_FUNCTION	√	-	√
GETIO	√	√	√
GOHOME	√	-	√
GOTO	√	-	√
GOTOLIMIT	√	√	√
GROUPLIST	-	√	-
GRP_CLOSE_GRIPPER	√	√	√
GRP_OPEN_GRIPPER	√	√	√
HERE	√	√	√
HEX$	√	√	√
ICMD	√	√	√

项目	任务	终端	函数
IF_..._THEN_..._ELSE	√	-	√
IFBK	√	√	√
IMPORT	√	-	√
INFORMATION	√	√	√
INPUT$	√	√	√
INSTR	√	√	√
INT	√	√	√
INTERPOLATIONTYPE	√	√	√
IPADDRESSMASK	√	√	√
ISMOVING	√	√	√
ISSETTLED	√	√	√
JERK	√	√	√
JERKACCELERATIONPERCENTAGE	√	√	√
JERKDECELERATIONPERCENTAGE	√	√	√
JERKMAX	√	√	√
JERKMAXROT	√	√	√
JERKMAXTRANS	√	√	√
JERKROT	√	√	√
JERKTRANS	√	√	√
JUMP	√	-	√
JUMP3	√	-	√
JUMP3CP	√	-	√
LCASE$	√	√	√
LEFT$	√	√	√
LEN	√	√	√
LOC	√	√	√
LOG	√	√	√
LOGGER	√	√	√
LTRIM$	√	√	√
MAINFILENAME	√	√	√
MID$	√	√	√
MOTION	√	√	√
MOTIONOVERLAP	√	√	√
MOVE	√	-	√
MOVES	√	-	√
MSG	√	√	√
NAME	-	√	-
NOOFCOORDINATES	√	√	√
NUM	√	√	√
NUMBEROFLOOPS	√	√	√
ONERROR	√	-	√
ONEVENT	√	-	√
ONSYSTEMERROR	√	-	√
OPEN_FILE	√	√	√
OPENSOCKET	√	√	√
OPMODE	√	√	√
PAYLOADINERTIA	√	√	√
PAYLOADLX	√	√	√
PAYLOADLY	√	√	√
PAYLOADMASS	√	√	√
PAYLOADMAX	√	√	√
PING	√	√	√
PLSLIST	-	√	-
PLSOUTPUT	√	√	√
PLSPOLARITY	√	√	√
PLSPOSITION	√	√	√
PLSRELATEDTO	√	√	√

项目	任务	终端	函数
PLSSOURCE	√	√	√
PLT_GET_INDEX_STATUS	√	√	√
PLT_MOVE_TO_ENTRY_POSITION	√	-	√
PLT_PICK_FROM_PALLET	√	-	√
PLT_PLACE_ON_PALLET	√	-	√
PLT_RETRACT_FROM_ITEM	√	-	√
PLT_SET_INDEX_STATUS	√	√	√
PLT_SET_INDEX_STATUS_EMPTY	√	√	√
PLT_SET_INDEX_STATUS_FULL	√	√	√
POSITIONCOMMAND	√	√	√
POSITIONCOMMANDHISTORY	√	√	√
POSITIONERROR	√	√	√
POSITIONERRORSETTLE	√	√	√
POSITIONFEEDBACK	√	√	√
POSITIONFINAL	√	√	√
POSITIONTOGO	√	√	√
POWER_OFF	√	√	√
POWER_ON	√	√	√
PRINT	√	√	√
PRINT_HASH	√	√	√
PRINTPOINT	√	√	√
PRINTPOINTUSING	√	√	√
PRINTTOBUFF	√	√	√
PRINTUSING	√	√	√
PRINTUSING$	√	√	√
PRINTUSING_HASH-SIGN	√	√	√
PRINTUSINGTOBUFF	√	√	√
PROCEED	√	√	√
PROCEEDTYPE	√	√	√
PROGRAM_..._END_PROGRAM	√	-	√
PULSE	√	-	√
RECORD	√	√	√
RECORD$	√	√	√
RECORDCLOSE	√	√	√
RECORDING	√	√	√
RECORDOFF	√	√	√
RECORDON	√	√	√
REM	√	-	√
RIGHT$	√	√	√
ROUND	√	√	√
RTRIM$	√	√	√
SELECT_..._CASE	√	-	√
SELECTAXES	√	√	√
SELECTBASE	√	√	√
SELECTTOOL	√	√	√
SELECTUSERFRAME	√	√	√
SEMAPHOREGIVE	√	√	√
SEMAPHORESTATE	√	√	√
SEMAPHORETAKE	√	√	√
SETIO	√	√	√
SETPOINT	√	√	√
SETWORKSPACE	√	√	√
SGN	√	√	√
SHL	√	√	√
SHR	√	√	√
SIMULATED	√	√	√
SIN	√	√	√

项目	任务	终端	函数
SIZE	√	√	√
SLEEP	√	-	√
SPACE$	√	√	√
SQRT	√	√	√
START	√	√	√
START_JOINT	√	√	√
STARTTYPE	√	√	√
STATE	√	√	√
STATUS	√	√	√
STOP	√	√	√
STR$	√	√	√
STRD$	√	√	√
STRING$	√	√	√
STRL$	√	√	√
STRUCTURE_TYPE_DEFINITION	-	-	-
SUB_..._END_SUB	√	-	√
SYSTEM.ERROR	√	√	√
SYSTEM.MOTION	√	√	√
TAN	√	√	√
TASKERROR	√	√	√
TASKERRORNUMBER	√	√	√
TASKID	√	√	√
TASKLIST	-	√	-
TASKSTATE	√	√	√
TASKSTATUS	√	√	√
THROW	√	√	√
TIME	-	√	-
TOASCII8$	√	√	√
TOCART	√	√	√
TOJOINT	√	√	√
TOOL	√	√	√
TORQUEADDCOMMAND	√	√	√
TORQUEGEARCOMMAND	√	√	√
TORQUEGEARFEEDBACK	√	√	√
TORQUEERROR	√	√	√
TORQUECOMMAND	√	√	√
TORQUEFEEDBACK	√	√	√
TOTALTIME	√	√	√
TOUTF8$	√	√	√
TRY_..._END_TRY	√	-	√
TYPEOF-ROBOT	√	√	√
TYPEOF	√	√	√
UCASE$	√	√	√
USERFRAME	√	√	√
UTF$	√	√	√
UTFSTRING$	√	√	√
VAL	√	√	√
VARLIST	√	√	√
VARLIST$	√	√	√
VCLOSETCP	√	√	√
VELOCITYCOMMAND	√	√	√
VELOCITYCOMMANDCARTESIAN	√	√	√
VELOCITYERROR	√	√	√
VELOCITYFEEDBACK	√	√	√
VELOCITYFEEDBACKCARTESIAN	√	√	√
VELOCITYFINALROT	√	√	√
VELOCITYFINALTRANS	√	√	√

项目	任务	终端	函数
VELOCITYMAX	√	√	√
VELOCITYMAXROT	√	√	√
VELOCITYMAXTRANS	√	√	√
VELOCITYOVERRIDE	√	√	√
VELOCITYOVERSPEED	√	√	√
VELOCITYROT	√	√	√
VELOCITYROTVALUE	√	√	√
VELOCITYSCALE	√	√	√
VELOCITYTRANS	√	√	√
VELOCITYTRANSFEEDBACKVALUE	√	√	√
VELOCITYTRANSVALUE	√	√	√
VGETJOBDATA	√	-	√
VGETJOBERROR	√	-	√
VGETJOBSTATUS	√	-	√
VOPENTCP	√	-	√
VPIXELTOPOS	√	-	√
VRECEIVEDATA	√	-	√
VRUNJOB	√	-	√
VRUNJOBFULL	√	-	√
VSENDCMD	√	-	√
VSMODE	√	√	√
VSTOPJOB	√	-	√
WAIT	√	-	√
WAITFORMOTION	√	-	√
WHILE_..._END_WHILE	√	-	√
WITH	√	√	√
WITHGLOBAL	√	√	√
WITHPLS	-	-	-
WORKPIECE	√	√	√
XMAX	√	√	√
XMIN	√	√	√
YMAX	√	√	√
YMIN	√	√	√
ZMAX	√	√	√
ZMIN	√	√	√

常见语法错误列表

序号	描述	原因	解决方法
1	变量或常量已声明	重复声明同一个变量。	修改变量名字。
2	变量或常量不存在令牌：	1. 没有声明该变量或常量 2. 声明变量的作用域不在调用变量的作用域内	在作用域内声明该变量。
3	在无效上下文中给出的指令	当前语法和上下文的指令不配对。	查看上下文的指令，参考指令格式，输入正确的配对关键字。
4	翻译语法错误	1. 使用不存在的指令 2. 错误的指令格式	参考 MC-Basic 手册，编写正确的指令格式。
5	程序结构错误	1. Program… End Program 没有成对出现，缺少其中一个关键字 2. Program…End Program 模块内出现语法错误。	通过屏蔽部分代码，定位出错的部分，进行修改。
6	语法错误	指令格式出错	点击错误，将显示所在错误行。可参考 MC-Basic 手册的指令介绍。
7	字符串太长（限制 = 83 个字符）	字符串变量一次被赋值超过 83 个字符。	用"+"号方式将字符串分多次赋值。
8	尺寸数与声明不符	声明的数组维数大于或者小于使用时的维数。 e.g: dim shared aa[2][10] as long ?aa[1]	声明合理的数组维数或使用时按照声明的维数访问。
9	索引类型错误。索引必须是长型	数组的索引号必须是长型，可能使用了字符串类型或者 double 类型作为索引号。 e.g: dim shared aa[2] as long dim shared a as string ?aa[a]	修改索引号变量的类型，或使用正确的变量或常数作为索引号。
10	输入类型错误	被赋值变量和赋值变量为不同类型。 e.g: dim shared aa[2] as long dim shared a as string aa[1] =a	修改被赋值变量或者赋值变量的变量类型。
11	名称是保留的命令名称或包含无效的符号令牌：	1. 使用了关键字作为变量/函数名字。 2. 使用了系统保留的名称作为变量/函数名字。 e.g: dim shared **dim** as long =2	修改声明的变量名字。

序号	描述	原因	解决方法
12	指令中的 Pls 未定义	使用的 PLS 没有在"Project Editor-Project Settings-Position Trigger"页面 中添加。 e.g: Move SCARA PL2 withpls=PT_bb PT_BB 没在配置页面中显示	在"Project Editor-Project Settings-Position Trigger"页面 中添加并配置。
13	在 subroutine/function 声明中，通过引用传递的变量有另一种类型	声明的函数参数变量和使用时传递的函数参数变量的变量类型不一致。 e.g: dim cc as string ?aa(cc) function aa(bb as long) as long end function	1.修改实际传递的函数变量类型 2. 使用正确的变量传递值 3.修改声明的参数变量类型。
14	不能通过引用 subroutine/function 来传递常量、返回值和复杂表达式	传递的实参为常数，但函数参数的声明为变量 e.g: ?aa("11") function aa(bb as string) as long end function	1. 在形参前面加上 "byval "关键字 2. 使用变量作为实参传递
15	该函数没有返回值	function 函数内没有编写返回值。	可添加函数返回值，但大多不会影响运行。(参考 function 指令格式)
16	在 user 上下文，不能声明 protected 函数或变量	声明的函数名或变量名和系统函数名相同。	修改函数名或变量名。

17	如果/循环/对于块不匹配	模块关键字没有配对
		e.g:
		if true then
		end if
		end if
			找到没配对的块模块，增加或删除块关键字。

18	这是一个 function 而不是一个 subroutine	调用 Function 类型的函数时，没有把返回值赋予给变量或使用"?"，把返回值打印出来。 e.g: call aa function aa as long end function	将函数返回值赋值给一个变量或在函数名前添加"?"指令。

序号	描述	原因	解决方法
19	这是一个 subroutine 而不是一个 function	调用没有参数的 subroutine 类型的函数时，没有使用 call <函数名> 的形式。 调用带参数的 subroutine 类型的函数时，可不使用 call 指令，但函数没有返回值可返回。 e.g: ?aa sub aa end sub	修改调用函数的格式。（参考 sub 指令格式）
20	无法删除文件/文件夹	1. 文件/ 文件夹不存在 2. 文件/ 文件夹正在被使用	确定文件/文件夹的使用情况，先关闭再删除。