EPSON

Epson RC+ 8.0 Ver.8.0
SPEL+语言参考
(RC700系列 / RC90-B)

翻译版

Rev.1
SCM247S6455F

目录

1. 前言

1.1 前言

感谢您购买本公司的机器人产品。
本手册记载了正确使用Epson RC+软件所需的事项。
请阅读本手册及相关手册后正确使用系统。
阅读之后，请妥善保管，以便随时取阅。

本公司的产品均通过严格的测试和检查，以确保机器人系统的性能符合本公司的标准。但是如果在超出本手册所描述的环境中使用本产品，则可能会影响产品的基本性能。

本手册阐述了本公司可以预见的危险和问题。请务必遵守本手册中的安全注意事项，安全正确地使用机器人系统。

1.2 商标

Microsoft、Windows、Windows 标识、Visual Basic、Visual C++ 为美国 Microsoft Corporation 在美国及其它国家的注册商标或商标。

Pentium为美国英特尔公司的商标。

其它公司名称、商标名称、产品名称均为各公司的注册商标或商标。

1.3 关于标记

- Microsoft® WindowsR 10 operating system
- Microsoft® WindowsR 11 operating system

在本手册中，Windows 10和Windows 11指的是上述各操作系统。在某些情况下，Windows一般是指Windows 10和Windows 11。

1.4 注意

禁止擅自复印或转载本手册的部分或全部内容。
本手册记载的内容将来可能会随时变更，恕不事先通告。
如您发现本手册的内容有误或需要改进之处，请不吝斧正。

1.5 制造商

SEIKO EPSON CORPORATION

1.6 联系方式

联系方式的详细内容登载于以下手册中的“销售商”处。
各地区的咨询处有所不同，敬请注意。
"安全手册”- 联系方式"

从以下网站也可浏览安全手册。
URL： https://download.epson.biz/robots/

1.7 阅读本手册之前

本节介绍了您在阅读本手册之前应了解的事项。

1.7.1 安全注意事项

请由取得相关资格的人员对机器人及相关机器进行搬运和设置。另外，请务必遵守各国的相关法规与法令。

使用前请仔细阅读本手册及相关说明书，并正确使用本机器。

阅读之后，请妥善保管，以便随时取阅。

1.7.2 正文中的符号的含义

⚠ 警告

该符号表示如果无视该标识进行错误使用，则可能会导致死亡或重伤的内容。

⚠ 注意

该符号表示如果无视该标识进行错误使用，则可能会导致受伤或只发生物品损坏的内容。

1.7.3 关于Epson RC+ 8.0的安装文件夹

Epson RC+ 8.0可安装在任意指定的路径。本手册中是默认Epson RC+ 8.0被安装C:\EpsonRC80中进行说明。

2. SPEL+ 命令一览

2.1 SPEL+ 命令一览

SPEL+ 命令一览。

系统管理相关命令

命令	说明
Reset	将控制器重置为初始状态。
SysConfig	显示系统设置参数。
SysErr	返回最新的错误状态或警告状态。
Date	显示日期。
Time	显示时间。
Date$	以字符串返回日期的函数。
Time$	以字符串返回时间的函数。
Hour	显示控制器的累计通电时间。
Stat	返回控制器的状态位。
CtrlInfo	返回控制器的信息。
RobotInfo	返回机器人的信息。
RobotInfo$	返回机器人的文本信息。
TaskInfo	返回任务信息。
TaskInfo$	返回任务的文本信息。
DispDev	设置当前的显示装置。
EStopOn	返回紧急停止状态。
CtrlDev	返回当前的控制装置的编号。
Cls	清除运行窗口、操作窗口、命令窗口的文本区域以及清除TP的打印面板。
Toff	关闭LCD上执行行的显示。
Ton	打开LCD上执行行的显示。
SafetyOn	返回安全门的状态。
Eval	执行命令窗口的语句并返回错误状态。
ShutDown	关闭Epson RC+、关闭或重启Windows。
TeachOn	返回示教模式的状态。
WindowsStatus	返回Windows的启动状态。

命令	说明
History	显示命令执行的历史记录。
ClearHistory	删除所有命令执行的历史记录。

机器人控制相关命令

命令	说明
AIO_TrackingSet	用于设置距离跟踪功能。
AIO_TrackingStart	用于启用距离跟踪功能。
AIO_TrackingEnd	用于退出距离跟踪功能。
AIO_TrackingOn函数	是返回距离跟踪功能的状态的函数。
AtHome	是返回当前的机器人姿势是否是Home姿势的函数。
Calib	将当前的机械臂姿势的脉冲值转换为由Calpls设置的脉冲值。
CalPls	是设置和显示校准所使用的位置姿势脉冲值或返回设置值的函数。
Hofs	设置和显示从编码器原点到软件原点的偏移脉冲。
JointAccuracy	设置和显示关节精度补偿的校正值。
HofsJointAccuracy	设置从编码器原点到软件原点的校正脉冲，但不改变关节精度补偿。
MCal	恢复增量编码器机器人的原点(检测机械原点)。
MCalComplete	返回MCal的状态。
MCordr	是指定和显示通过MCaL命令进行原点恢复时的关节动作顺序，并返回设置值的函数。
Power	将功率模式设为High或Low，并显示当前的模式。
MHour函数	是用于返回机器人电动机累计励磁时间的函数。
Motor	打开或关闭机器人所有轴的电动机。
SFree	将指定关节切换至SFree状态。
SLock	将指定关节切换至SLock状态。
SyncRobots	开始动作预约中的机器人动作。
Jump	是以门控动作移动至目标坐标的动作(PTP动作)。
Jump3	是以三维门控动作移动至目标坐标的动作(PTP动作+CP动作)。
Jump3CP	是以三维门控动作移动至目标坐标的动作(CP动作)。
JumpTLZ	是以三维门控动作移动至目标坐标的动作(PTP动作+CP动作)。
Arch	是设置和显示Jump命令的Arch参数以及返回设置值的函数。
LimZ	是设置Jump命令时第3关节高度(Z坐标值)初始值以及返回设置值的函数。
LimZMargin	是设置以高于LimZ设置值的位置开始动作时的错误检测界限以及返回设置值的函数。
Sense	以Jump命令指定Sense时设置和显示使其在目标坐标上方停止的条件。
JS	是返回执行Jump Sense后，Sense输入是否成立的函数。

命令	说明
JT	是返回之前的Jump动作结果的函数。
Go	以PTP动作从当前位置移动到指定位置。
Pass	穿过指定点附近(不停止)的PTP动作。
Pulse	通过PTP控制使机械臂移动到各关节的脉冲值指定的点。
BGo	在选择的本地坐标系上执行偏移PTP动作。
BMove	在本地坐标系上执行偏移直线动作。
TGo	在工具坐标系上执行偏移PTP动作。
TMove	在工具坐标系上执行偏移直线动作。
Till	设置和显示通过动作命令进行Till指定时，动作中途停止并结束处理的条件。
TillOn	返回Till的状态。
!…!	在动作过程中并列进行I/O等的输入输出处理的功能。
Speed	是设置和显示PTP动作速度以及返回设置值的函数。
Accel	是设置和显示PTP动作的加减速度以及返回设置值的函数。
SpeedFactor	是设置和显示PTP动作速度以及返回设置值的函数。
Inertia	设置机器人的负载惯性和离心率。
Weight	设置和显示补偿PTP动作时的速度和加减速度的参数。
Arc	是在XY平面上执行曲线动作(CP动作)
Arc3	是在三维上执行曲线动作(CP动作)
Move	直线动作(CP动作)
Curve	制作通过自由曲线进行CP控制的数据和点。
CVMove	执行Curve命令定义的自由曲线CP动作。
SpeedS	是设置和显示CP动作速度以及返回设置值的函数。
AccelS	是设置和显示CP动作的加减速度以及返回设置值的函数。
SpeedR	是设置和显示工具姿势变化的速度以及返回设置值的函数。
AccelR	是设置和显示工具姿势变化的加减速度以及返回设置值的函数。
AccelMax	是返回Accel可以设置的加减速度的最大值的函数。
Brake	打开或关闭当前机器人指定关节的制动器。
Home	向原点位置(待机姿势)的移动(PTP动作)
HomeClr	清除原点位置的定义。
HomeDef	返回原点位置的定义。

命令	说明
HomeSet	设置和显示原点位置(待机姿势)的脉冲。
Hordr	是指定和显示执行Home时各关节的动作顺序以及返回设置值的函数。
InPos	是返回机器人的定位状态的函数。
CurPos	返回机器人的当前的动作目标位置。
TCPSpeed	返回计算的当前工具中心点速度。
Pallet	对托盘进行定义和显示定义托盘。
PalletClr	清除托盘的定义。
Fine	设置和显示目标位置的定位结束判断范围(单位：pulse)
FineDist	设置和显示目标位置的定位结束判断范围(单位：mm)
FineStatus函数	对于使用Fine或使用FineDist通过整数值返回。
QP	是设置/解除快速暂停功能，显示当前设置以及返回设置值的函数。
QPDecelR	设置有关CP动作时工具姿势变化的快速暂停减速度。
QPDecelS	设置CP动作时的快速暂停减速度。
CP	设置路径运动。
Box	设置和显示进入检测区域。
BoxClr	清除已设置的Box。
BoxDef	返回是否已设置进入检测区域。
Plane	设置和显示进入检测平面。
PlaneClr	清除(未定义)进入检测平面。
PlaneDef	返回进入检测平面的设置状态。
InsideBox	返回进入检测区域的检测状态。
InsidePlane	返回进入检测平面的检测状态。
GetRobotInsideBox	返回进入到进入检测区域内的机器人。
GetRobotInsidePlane	返回进入到进入检测平面内的机器人。
Find	设置和显示在动作命令中保存坐标的条件。
FindPos	返回在执行动作命令过程中通过Find保存的坐标。
PosFound	返回Find命令时执行的状态。
WaitPos	即使在路径运动有效的状态下，也要在执行下一语句之前，等待机器人进行减速停止。
Robot	选择机器人(以Robot函数返回机器人编号)。
RobotModel$	返回机器人的型号名称。

命令	说明
RobotName$	返回机器人名称。
RobotSerial$	返回机器人的序列号。
RobotType	返回机器人类型。
TargetOK	返回可否从当前位置向目标坐标进行PTP(point to point)动作的信息。
ROTOK函数	返回是否可以在执行操作命令时，将ROT修饰参数添加到目标坐标中。
JRange	是设置脉冲值指定关节的容许动作区域以及返回设置值的函数。
Range	设置和显示脉冲值的容许动作区域。
XYLim	是设置和显示XY平面容许动作区域以及返回设置值的函数。
XYLimClr	清除已设置的XYLim。
XYLimDef	返回是否设置XYLim的信息。
XYLimMode	是设置和显示XYLim的监控方式以及返回设置值的函数
XY	以点数据返回指定的坐标值。
Dist	返回两个机器人坐标间的距离。
DiffToolOrientation函数	返回工具坐标系每个坐标轴形成的角度。
DiffPoint函数	返回2个指定点之间的差值。
PTPBoost	是设置和显示PTP动作微移时的参数以及返回设置值的函数。
PTPBoostOK	返回从当前位置向目标坐标进行的PTP(point to point)动作是否为微移。
PTPTime	返回PTP动作命令所需时间的推测值，不执行PTP动作。
CX	是返回指定点数据X坐标值的设置和设置值的函数。
CY	是返回指定点数据Y坐标值的设置和设置值的函数。
CZ	是返回指定点数据Z坐标值的设置和设置值的函数。
CU	是返回指定点数据U坐标值的设置和设置值的函数。
CV	是返回指定点数据V坐标值的设置和设置值的函数。
CW	是返回指定点数据W坐标值的设置和设置值的函数。
CR	是返回指定点数据R坐标值的设置和设置值的函数。
CS	是返回指定点数据S坐标值的设置和设置值的函数。
CT	是返回指定点数据T坐标值的设置和设置值的函数。
Pls	是返回各关节的脉冲值的函数。
Agl	是返回指定关节的角度或位置的函数。
PAgl	是根据指定的坐标值计算和返回关节位置(旋转关节的角度、移动关节的位置)的函数。
JA	返回根据关节角度计算的机器人坐标。
AglToPls	将机器人各关节的角度转换为脉冲。

命令	说明
DegToRad	将角度转换为弧度。
RadToDeg	将弧度转换为角度。
Joint	在关节坐标系中显示当前机器人位置。
JTran	只有无需原点恢复的1关节进行PTP动作。
PTran	从当前位置进行指定脉冲量的仅1关节的PTP动作。
RealPls	返回已指定关节的脉冲值。
RealPos	返回指定机器人当前位置。
RealAccel函数	返回通过OLAccel自动调整的Accel值的函数。
PPls	是根据指定的坐标值计算并返回关节脉冲的函数。
LJM函数	返回为确保参考点相对于指定点的关节移动最小量而转换姿势标志的点数据。
AutoLJM	设置自动LJM。
AutoLJM函数	是返回自动LJM的状态的函数。
AutoOrientationFlag	用于变更N6-A1000**的姿势标志。
AutoOrientationFlag函数	是用于返回AutoOrientationFlag的状态的函数。
AvoidSingularity	设置避免奇点的功能。
AvoidSingularity函数	是返回避免奇点功能的状态的函数。
SingularityAngle	设置避免奇点功能的奇点附近角度。
SingularityAngle函数	是返回避免奇点功能的奇点附近角度的函数。
SingularitySpeed	设置避免奇点功能的奇点附近速度。
SingularitySpeed函数	是返回避免奇点功能的奇点附近速度的函数。
SingularityDist	设置奇点通过功能所需的特殊点附近距离。
SingularityDist函数	是返回奇点通过功能所需的特殊点附近距离的函数。
AbortMotion	将中断动作命令并执行动作的任务设为错误。
Align函数	是返回将机器人的姿势对准本地坐标系最近的坐标轴时转换的点数据的函数。
AlignECP函数	是返回将机器人的姿势对准ECP坐标系最近的坐标轴时转换的点数据的函数。
SoftCP	设置和显示SoftCP动作模式。
SoftCP函数	是返回SoftCP动作模式的状态的函数。
Here	将当前位置作为机器人坐标进行示教。
Where	显示机器人的当前位置数据。。
PerformMode	设置机器人的动作模式。

命令	说明
PerformMode函数	返回机器人的动作模型编号。
VSD	设置SCARA机器人的变速CP动作功能。
VSD函数	返回SCARA机器人的变速CP动作功能的设置。
CP_Offset	设置CP On时的后续动作命令的开始偏移时间。
CP_Offset函数	返回CP On时的后续动作命令的开始偏移时间。
AvgSpeedClear	清除关节的平均速度并进行初始化。
AvgSpeed	显示关节的平均速度。
AvgSpeed函数	是返回关节的平均速度的函数。
PeakSpeedClear	清除关节的峰值转矩并执行初始化。
PeakSpeed	显示关节的峰值速度。
PeakSpeed函数	是返回关节的峰值速度的函数。

转矩相关命令

命令	说明
TC	是用于显示转矩控制模式设置及当前模式的函数。
TCSpeed	是用于设置转矩控制期间的速度限制值的函数。
TCLim	是用于设置转矩控制模式下各关节转矩限制值的函数。
RealTorque	是用于返回指定关节当前转矩指令值的函数。
ATCLR	清除关节的实效转矩并进行初始化。
ATRQ	显示指定的关节的实效转矩值。
PTCLR	清除关节的峰值转矩并执行初始化。
PTRQ	显示指定关节峰值转矩值。
OLAccel	设置基于过载率的加减速度自动调整。
OLRate	显示指定关节过载率。
LimitTorque	是用于设置高功率模式时的转矩上限值、以及返回设置值的函数。
LimitTorque函数	是用于返回LimitTorque命令的设置值的函数。
LimitTorqueLP	是用于设置低功率模式时的转矩上限值并返回设置值的函数
LimitTorqueLP函数	是用于返回LimitTorque命令的设置值的函数。
LimitTorqueStop	是用于在高功率模式下达到转矩上限时设置是否停止机器人以及返回设置值的函数。
LimitTorqueStop函数	是用于返回LimitTorque命令的设置值的函数。
LimitTorqueStopLP	是用于在低功率模式下达到转矩上限时设置是否停止机器人以及返回设置值的函数。
LimitTorqueStopLP函数	是用于返回LimitTorqueStop命令的设置值的函数。

输入输出相关命令

命令	说明
On	打开输出位。
Off	关闭输出位。
Oport	读取输出位的状态。
Sw	读取I/O输入或存储器I/O的状态并通过I/O进行使用。
In	读取1字节(8位)的输入数据并通过I/O进行使用。
InW	读取1个字(16位)的输入数据并通过I/O进行使用。
InBCD	以BCD(转换为二进制代码的10进制数)格式读取输入端口的输入状态。
Out	输出1字节(8位)的输出数据并通过I/O进行使用。
OutW	输出1个字(16位)的输出数据并通过I/O进行使用。
OpBCD	以BCD格式向输出端口输出1字节的数据。
MemOn	是按照位编号打开指定的存储器I/O的命令。
MemOff	是按照位编号关闭指定的存储器I/O的命令。
MemSw	是用于返回指定存储器I/O位的状态的函数。
MemIn	是以字节为单位返回存储器I/O状态的函数。
MemOut	是以字节为单位返回存储器I/O状态的命令。
MemInW	以字为单位返回存储器I/O端口状态，字端口由16个存储器I/O位构成。
MemOutW	以16位并同时以字为单位设置存储器I/O端口的状态。
Wait	等待条件或时间。
TMOut	设置执行Wait命令时的超时时间。
TW	是返回Wait命令的条件表达式是否成立的函数。
Input	接收显示装置的输入并保存到变量中。
InReal	将2个字(32位)的输入数据作为32位浮点数据(符合IEEE754标准)来读取。通过I/O来使用。
Print	在输出画面、命令画面、操作窗口显示数据。
Line Input	读取1行输入数据并将该数据代入到字符串变量中。
Input#	从文件、通信端口、数据库或装置输入字符串数据或数值数据并保存到变量中。
Print#	将数据输出到指定的文件、通信端口、数据库和装置中。
Line Input#	从文件、通信端口、数据库、装置读取1行数据。
Lof	返回指定RS-232C端口或TCP/IP端口缓冲器的接收数据行数。
SetIn	设置虚拟I/O的输入端口(8位)。
SetInW	设置虚拟I/O的输入端口(16位)。
SetSw	设置虚拟I/O的输入。

命令	说明
IOLabel$	返回指定的输入输出的位、字节、字的I/O标签。
IONumber	返回指定的I/O标签的I/O端口编号。
IODef	返回指定的I/O标签是否被定义。
OpenCom	打开RS-232C端口。
OpenCom函数	是用于获取实施OpenCom的任务编号的函数。
CloseCom	关闭通过OpenCom打开的RS-232C端口。
SetCom	设置或显示RS-232C端口参数。
ChkCom	返回通信端口的接收缓冲器内的字符数。
OpenNet	打开TCP/IP网络端口。
OpenNet函数	是用于获取实施OpenNet的任务编号的函数。
OutReal	将实数值输出数据作为32位浮点数据(符合IEEE754标准)输出到输出端口2个字(32位)中。通过I/O来使用。
CloseNet	关闭通过OpenNet打开的TCP/IP端口。
SetNet	设置TCP/IP端口的参数。
ChkNet	返回网络端口的接收缓冲器内的字符数。
WaitNet	等待TCP/IP端口建立连接。
Read	从文件或通信端口读取指定的字符数。
ReadBin	从文件或通信端口读取二进制数据。
Write	将字符串写入到文件或通信端口中，不附加行末终止符。
WriteBin	将二进制数据写到文件或通信端口中。
InputBox	显示输入对话框并返回输入的内容。
MsgBox	显示对话框信息。
RunDialog	从SPEL+程序中启动Epson RC+画面。
LatchEnable	通过R-I/O输入，将机器人位置的闩锁功能设为有效/无效。
LatchState函数	返回通过R-I/O实现的机器人位置闩锁状态。
LatchPos函数	返回利用R-I/O输入信号进行闩锁的机器人位置。
SetLatch	设置通过R-I/O输入实现的机器人位置闩锁功能。
AIO_In函数	从模拟I/O输入通道读取模拟值。
AIO_InW函数	从模拟I/O输入通道读取1个字的输入数据。
AIO_Out	将模拟值输出到模拟I/O输出通道中。
AIO_Out函数	返回模拟I/O输出通道的输出状态。

命令	说明
AIO_OutW	将1个字的数据输出到模拟I/O输出通道中。
AIO_OutW函数	以1个字返回模拟I/O输出通道的输出状态。
AIO_Set	将速度信息输出到模拟I/O输出通道中。
AIO_Set函数	返回作为选项的模拟I/O输出通道中设置的机器人速度输出设置信息。

点控制相关命令

命令	说明
ClearPoints	删除存储器上的位置数据。
LoadPoints	将点文件读入到机器人点存储区域中。
SavePoints	将主存储器中的点数据保存到当前机器人的磁盘文件中。
ImportPoints	将点文件输入到现在正在选择的机器人的项目中。
ExportPoints	将点文件导出到PC的指定路径中。
P#	定义点数据。
PDef	返回指定的点数据是否被定义。
PDel	删除指定的点数据。
PLabel	设置指定点数据的标签。
PLabel$	返回指定点编号所定义的点标签。
PNumber	返回对应于点标签的点编号。
PList	显示当前机器人存储器中的点数据。
PLocal	设置指定点数据的本地属性。
PDescription	设置指定点数据的注释。
PDescription$	返回指定点编号中定义的点的注释。
WorkQue_Add	将工作队列数据(点数据和用户数据)追加到指定工作队列中。
WorkQue_AutoRemove	在指定的工作队列中设置自动删除功能。
WorkQue_AutoRemove函数	返回工作队列中设置的自动删除功能的状态。
WorkQue_Get函数	从指定的工作队列返回点数据。
WorkQue_Len函数	返回注册在指定的工作队列的有效工作队列数据的数值。
WorkQue_List	显示指定工作队列的工作队列数据一览(点数据和用户数据)。
WorkQue_Reject	设置和显示防止在指定工作队列中重复注册点数据的最小距离。
WorkQue_Reject函数	返回设置在指定工作队列中防止重复注册的距离。
WorkQue_Remove	从指定工作队列数据中删除工作队列数据(点数据和用户数据)。
WorkQue_Sort	设置和显示指定工作队列的Sort方法。
WorkQue_Sort函数	返回指定工作队列中设置的Sort方法。

命令	说明
WorkQue_UserData	重新设置和显示指定工作队列中注册的用户数据(实数)。
WorkQue_UserData函数	返回指定工作队列中注册的用户数据(实数)。

坐标转换相关命令

命令	说明
AreaCorrection函数	返回利用校正区域校正过的点。
AreaCorrectionClr	清除校正区域。
AreaCorrectionDef	返回校正区域的设置。
AreaCorrectionInv函数	将已完成校正的点恢复原状。
AreaCorrectionOffset函数	将已完成校正的点返回相对移动的点。
AreaCorrectionSet	设置和显示校正区域。
Arm	是选择机械臂、显示当前所选机械臂编号或返回设置值的函数。
ArmSet	设置和显示增设的机械臂。
ArmDef	返回机械臂的设置。
ArmClr	清除机械臂的设置。
ArmCalib	设置和显示机械臂长校正启用或禁用。
ArmCalibClr	清除机械臂长校正的设置。
ArmCalibDef	返回机械臂长校正的设置。
ArmCalibSet	显示机械臂长校正的设置。
Tool	是用于选择工具、或者显示已选择的工具编号或返回设置值的函数。
TLSet	设置和显示工具坐标系。
TLDef	返回工具坐标系的设置。
TLClr	清除工具坐标系的设置。
ECP	是用于选择ECP、或者显示已选择的ECP编号或返回设置值的函数。
ECPSet	定义和显示外部控制点(ECP)。
ECPDef	返回ECP的设置。
ECPClr	清除ECP的设置。
Base	定义和显示基础坐标系。
Local	定义和显示本地坐标系数据。
LocalDef	返回本地坐标系的设置。
LocalClr	清除本地坐标系。(未定义)
Elbow	是设置点的肘部姿势和返回设置的函数。
Hand	是设置点的手臂姿势和返回设置的函数。

命令	说明
Wrist	是设置点的手腕姿势和返回设置的函数。
J4Flag	是设置点的J4Flag值和返回设置值的函数。
J6Flag	是设置点的J6Flag值和返回设置值的函数。
J1Flag	是设置点的J1Flag值和返回设置值的函数。
J2Flag	是设置点的J2Flag值和返回设置值的函数。
J1Angle	是设置点的J1Angle值和返回设置值的函数。
J4Angle	是设置点的J4Angle值和返回设置值的函数。
VxCalib	创建Vision Guide以外的视觉系统的校准数据。
VxTrans	该函数用于进行从像素坐标到机器人坐标的坐标转换，以及返回已转换点数据。
VxCalInfo	该函数用于返回Vision Guide以外的视觉系统的校准结束状态和校准数据。
VxCalDelete	删除Vision Guide以外的视觉系统的校准数据。
VxCalSave	将Vision Guide以外的视觉系统的校准数据保存到文件中。
VxCalLoad	从文件中读取Vision Guide以外的视觉系统的校准数据。

程序控制相关命令

命令	说明
Function	函数定义
For...Next	反复执行For...Next之间的一系列语句。
GoSub	执行子例程。
Return	从子例程返回程序。
GoTo	将程序控制移至指定的标签中。
Call	作为子例程调用函数。
If… Then...Else...EndIf	程序的条件转移
Else	以If…Then…Else形式使用，如果不满足条件表达式，将执行Else之后的语句，Else可省略。
Select ... Send	按照比较值选择要执行的语句。
Do...Loop	在循环的起始行或结束行进行条件判断，条件一致或不一致时，在Do...Loop之间反复。
Declare	调用DLL(动态链接库)定义的外部函数。
Trap	定义中断以及发生中断时的处理。
OnErr	定义错误处理例程的位置。
Era	是返回发生错误的关节的编号的函数。
Erf$	返回发生错误的函数的名称。
Erl	是返回发生错误的行的编号的函数。

Err	是返回错误代码的函数。
Ert	是返回发生错误的任务的编号的函数。
Errb	是返回发生错误的机器人的编号的函数。
ErrMsg$	是返回与错误代码相对应的错误信息的函数。
Signal	向正在执行WaitSig命令的任务发送信号。
SyncLock	使用相互排他锁定，使多个任务同步。
SyncUnlock	解除事先由SyncLock锁定的信号编号锁定。
WaitSig	等待其它任务的Signal命令发出的同步信号。
ErrorOn	返回控制器的错误状态。
Error	发生用户定义错误。
EResume	在结束错误处理例程后，重新开始执行程序。
PauseOn	返回暂停状态(Pause状态)。
Exit	用于强制结束循环或函数。

程序执行相关命令

命令	说明
Xqt	执行函数名指定的任务。
Pause	暂停可暂停的所有任务。
Cont	重新执行暂停的所有任务。
Halt	暂停正在执行的任务。
Quit	结束任务。
Resume	继续执行暂停的任务。
MyTask	返回当前程序任务编号。
TaskDone	用作任务结束的确认函数。
TaskState	用作获取任务当前状态的函数。
TaskWait	等待指定任务的结束。
Restart	中断所有正在执行的任务，重新执行已执行的最后程序组。
Recover	将恢复动作执行到安全门打开时的位置并返回状态。
RecoverPos	返回安全门打开时的位置。
StartMain	通过后台任务执行主函数。

模拟命令

命令	说明
#define	为将识别符转换为指定字符串而进行的定义。

命令	说明
#ifdef ... #endif	带条件的编译。
#ifndef ... #endif	带条件的编译。
#include	插入指定的包含文件。
#undef	#清除#define语句定义的识别符。

文件管理相关命令

命令	说明
ChDir	更改指定驱动器的当前目录。
ChDisk	设置要进行文件操作的对象磁盘。
MkDir	创建子目录。
RmDir	删除子目录。
RenDir	更改目录名。
FileDateTime$	返回文件的日期和时间。
FileExists	检查文件是否存在。
FileLen	返回文件大小。
FolderExists	检查文件夹是否存在。
FreeFile	返回当前未使用的文件编号并预约该编号。
Del	删除文件。
Copy	复制文件。
Rename	更改文件名。
AOpen	打开文件以进行增补(附加)。
BOpen	以二进制访问模式打开文件。
ROpen	打开要读取的文件。
Uopen	打开要读取和写入的文件。
WOpen	打开要写入的文件。
Input#	从文件、通信端口、数据库、装置输入字符串或数值数据并保存到变量中。
Print#	将数据输出到指定的文件、通信端口、数据库和装置中。
Line Input#	从文件、通信端口、数据库、装置读取1行数据。
Read	从文件或通信端口读取指定的字符数。
ReadBin	从文件或通信端口读取二进制数据。
Write	将字符串写入到文件或通信端口中，不附加行末终止符。
WriteBin	将二进制数据写到文件或通信端口中。
Seek	设置当前文件指针。

命令	说明
Close	关闭文件。
Eof	返回文件的EOF(已打开的文件的指针位于尾端)。
ChDrive	更改当前驱动器。
CurDir$	以字符串返回当前目录。
CurDrive$	以字符串返回当前驱动器名。
CurDisk$	以字符串返回当前磁盘名。
Flush	将缓存写入到文件中。

现场总线相关命令

命令	说明
FbusIO_GetBusStatus	返回指定的现场总线的状态。
FbusIO_GetDeviceStatus	返回指定的现场总线装置的状态。
FbusIO_SendMsg	向现场总线I/O装置发送信息，返回答复。

数值相关命令

命令	说明
Ctr	是返回计数器的计数值的函数。
CTReset	计数器的定义与复位。
ElapsedTime	是计算节拍时间的函数。
ResetElapsedTime	复位、起动计算节拍时间用的计时器。
Tmr	计时器函数。
TmReset	复位和起动计时器。
Sin	是返回指定角度的正弦的函数。
Cos	是返回指定角度的余弦的函数。
Tan	是返回指定角度的正切的函数。
Acos	是返回指定数值的反余弦的函数。
Asin	是返回指定数值的反正弦的函数。
Atan	是返回指定数值的反正切的函数。
Atan2	是返回连接坐标原点和指定点的直角角度的函数。
Sqr	是返回平方根的函数。
Abs	是返回绝对值的函数。
Sgn	是返回数值的符号的函数。
Int	是返回舍弃数值小数部分后的值的函数。

命令	说明
BClr	清除指定的1位数值并返回该值。
BSet	设置指定的1位数值并返回结果。
BTst	返回1位数值的状态。
BClr64	清除指定的1位数值并返回该值。
BSet64	设置指定的1位数值并返回结果。
BTst64	返回1位数值的状态。
Fix	从实数值中取出整数部分。
Hex	将16进制数表示的数值转换为字符串并返回。
Randomize	随机数系列的初始化。
Redim	在运行时重新定义数组的最大元素编号。
Rnd	返回随机数。
UBound	返回指定数组中可设置的下标的最大值。

字符串相关命令

命令	说明
Asc	以10进制数返回字符串第一个字符的字符码。
Chr$	返回与字符码相关的字符。
Left$	从字符串的左侧提取指定数的字符串。
Mid$	从指定字符串的起始位置提取指定字符数的字符。
Right$	从字符串的末尾提取指定数的字符串。
Len	返回字符串的长度(字符数)。
LSet$	返回在指定字符串的最后添加空格以形成指定长度的字符串。
RSet$	返回在字符串的开头添加空格以形成指定长度的字符串。
Space$	返回指定数量的空格字符串。
Str$	将数值转换为字符串。
Val	将由数字组成的字符串转换为数值。
LCase$	以小写字符返回指定的字符串。
UCase$	以大写字符返回指定的字符串。
LTrim$	删除字符串开头的空格并返回。
RTrim$	删除字符串最后的空格并返回。
Trim$	删除字符串开头和最后的空格并返回。
ParseStr	在令牌数组中对字符串进行语法分析。

命令	说明
FmtStr	将数字或日期/时间的标记格式化。
FmtStr$	将数字或日期/时间的标记格式化。
InStr	从字符串中检索字符串并返回发现的位置。
Tab$	返回指定数量的制表符字符串。

逻辑运算相关命令

命令	说明
And	逻辑与运算
Or	逻辑或运算
LShift	数值数据的逻辑左移
LShift64	数值数据的逻辑左移
Mod	整数的取余运算
Not	非运算
RShift	数值数据的逻辑右移
RShift64	数值数据的逻辑右移
Xor	逻辑异或运算
Mask	通过Wait语句执行位宽AND运算。

变量相关命令

命令	说明
Boolean	声明Boolean型变量。
Byte	声明Byte型变量。
Double	声明Double型变量。
Global	声明全局变量。
Int32	声明4字节整数型变量。
Integer	声明2字节整数型变量。
Long	声明Long型变量。
Int64	声明8字节整数型变量。
Real	声明实数型变量。
Short	声明2字节整数型变量。
String	声明字符串型变量。
UByte	声明无符号整数型变量。
UInt32	声明4字节无符号整数型变量。
UShort	声明2字节无符号整数型变量。
UInt64	声明8字节无符号整数型变量。

安全相关命令

命令	说明
GetCurrentUser$	返回当前的Epson RC+用户的登录ID。
Login	在执行时，以其它用户身份登录到Epson RC+。

传送带跟踪相关命令

命令	说明
Cnv_AbortTrack	中断传送带CuePoint的动作命令。
Cnv_Accel函数	返回传送带跟踪前的加速度和减速度的设置值。
Cnv_Accel	设置传送带跟踪前的加速度和减速度的设置值。
Cnv_AccelLim	设置传送带跟踪后的加速度和减速度的设置值。
Cnv_AccelLim函数	返回传送带跟踪后的加速度和减速度的设置值。
Cnv_Adjust	设置是否执行获取传送带的跟踪延迟校正值的操作。
Cnv_AdjustClear	清除传送带的跟踪延迟的校正值。
Cnv_AdjustGet函数	返回传送带的跟踪延迟的校正值。
Cnv_AdjustSet	设置清除传送带的跟踪延迟的校正值。
Cnv_Downstream函数	返回传送带的下游限值的设置值。
Cnv_Downstream	设置传送带的下游限值的设置值。
Cnv_Fine函数	返回指定传送带的跟踪完成判断范围的设置。
Cnv_Fine	设置和显示相对于指定传送带的传送带跟踪完成判断范围。
Cnv_Flag函数	返回跟踪停止线的跟踪状态。
Cnv_Mode函数	返回传送带的设置模式的设置值。
Cnv_Mode	设置传送带的设置模式的设置值。
Cnv_Name$函数	返回指定传送带的名称。
Cnv_Number函数	返回指定传送带的名称的传送带编号。
Cnv_OffsetAngle	设置传送带队列数据的偏移值。
Cnv_OffsetAngle函数	返回传送带队列数据的偏移值。
Cnv_Point函数	将传感器坐标值转换为传送带坐标值并返回。
Cnv_PosErr函数	返回当前tracking位置与目标的位置偏差。
Cnv_PosErrOffset	设置一个值，用于修正当前跟踪位置和目标位置之间的偏差。
Cnv_Pulse函数	返回传送带的当前位置的脉冲。
Cnv_QueAdd	在传送带队列数据中添加点数据。
Cnv_QueGet函数	从指定传送带队列数据中返回点数据。
Cnv_QueLen函数	返回指定传送带队列数据的数量。
Cnv_QueList	显示指定传送带队列数据一览。
Cnv_QueMove	使上游传送带的队列数据移至下游传送带的队列中。

命令	说明
Cnv_QueReject	设置和显示防止传送带重复注册的最小距离。
Cnv_QueReject函数	返回防止传送带队列重复注册的距离。
Cnv_QueRemove	从传送带队列数据中删除队列数据。
Cnv_QueUserData	显示和设置与队列入口相关的用户数据。
Cnv_QueUserData函数	返回与队列入口相关的用户数据。
Cnv_RobotConveyor函数	返回跟踪中的传送带编号。
Cnv_Speed函数	返回传送带的动作速度。
Cnv_Trigger	锁定传送带的当前位置，以便执行下面的Cnv_QueAdd语句。
Cnv_Upstream函数	返回传送带的上游限值的设置值。
Cnv_Upstream	设置传送带的上游限值的设置值。

DB相关命令

命令	说明
CloseDB	关闭用OpenDB命令打开数据库，解放数据基础号码。
DeleteDB	从出于打开状态的数据库内的表中删除数据。
OpenDB	打开数据库、Excel工作薄。
SelectDB	是用于从打开的数据库内的表中检索数据的函数。
UpdateDB	更新打开的数据库内表的数据。

PG相关命令

命令	说明
PG_FastStop	突然停止连续旋转中的脉冲输出轴。
PG_LSpeed	设置脉冲输出轴开始加速时脉冲速度及减速结束时的脉冲速度。
PG_Scan	开始脉冲输出轴的连续旋转动作。
PG_SlowStop	减速停止连续旋转中的脉冲输出轴。

碰撞检测相关命令

命令	说明
CollisionDetect	设置碰撞检测功能的有效/无效。
CollisionDetect函数	是返回CollisionDetect命令的设置值的函数。

部件消耗管理相关命令

命令	说明
HealthCalcPeriod	设置部件消耗管理的运算期间。
HealthCalcPeriod函数	是返回部件消耗管理运算期间的函数。
HealthCtrlAlarmOn函数	是返回控制器的部件消耗报警状态的函数。

命令	说明
HealthCtrlInfo	显示控制器部件的推荐更换期限到期之前的剩余月数。
HealthCtrlInfo函数	是返回控制器部件的推荐更换期限到期之前的剩余月数的函数。
HealthCtrlRateOffset	设置控制器部件消耗率的偏移值。
HealthCtrlReset	对控制器的部件消耗率进行初始化。
HealthCtrlWarningEnable	设置控制器部件消耗报警通知的有效/无效。
HealthCtrlWarningEnable函数	是返回控制器部件消耗报警通知的有效/无效状态的函数。
HealthRateCtrlInfo函数	是返回控制器的部件消耗率的函数。
HealthRateRBInfo函数	是返回机器人的部件消耗率的函数。
HealthRBAlarmOn函数	是返回机器人的部件消耗报警状态的函数。
HealthRBAnalysis	显示机器人部件消耗相关的分析结果(推荐更换期限到期之前的剩余月数)。
HealthRBAnalysis函数	是返回机器人部件消耗相关的分析结果(推荐更换期限到期之前的剩余月数)的函数。
HealthRBDistance	显示指定关节的驱动量。
HealthRBDistance函数	是返回指定关节的驱动量的函数。
HealthRBInfo	显示机器人部件的推荐更换期限到期之前的剩余月数。
HealthRBInfo函数	是返回机器人部件的推荐更换期限到期之前的剩余月数的函数。
HealthRBRateOffset	设置机器人部件消耗率的偏移值。
HealthRBReset	对机器人的部件消耗率进行初始化。
HealthRBSpeed	显示指定关节的平均速度。
HealthRBSpeed函数	是返回指定关节的平均速度的函数。
HealthRBStart	开始机器人部件消耗相关分析。
HealthRBStop	结束机器人部件消耗相关分析。
HealthRBTRQ	显示指定关节的转矩值。
HealthRBTRQ函数	是返回指定关节的转矩值的函数。
HealthRBWarningEnable	设置机器人部件消耗报警通知的有效/无效。
HealthRBWarningEnable函数	是返回机器人部件消耗报警通知的有效/无效状态的函数。

模拟器相关命令

命令	说明
SimSet	进行模拟器的目标设置和操作以及机器人的动作设置。
SimGet	获取模拟器的目标设置值。

Hand功能相关命令

有关详细信息，请参阅以下手册。
《Hand功能》

命令	说明
Hand_On	夹爪 : 控制夹具执行抓取动作 电动螺丝刀 : 控制夹具执行拧紧动作
Hand_On函数	夹爪 : 当夹具处于抓取状态时，返回“True” 电动螺丝刀 : 当夹具完成拧紧动作时，返回“True”
Hand_Off	夹爪 : 控制夹具执行放开动作 电动螺丝刀 : 控制夹具执行拧松动作
Hand_Off函数	夹爪 : 当夹具处于放开状态时，返回“True” 电动螺丝刀 : 当夹具完成拧松动作时，返回“True”
Hand_TW函数	上一个Hand_On函数或Hand_Off命令超时时，返回“True”
Hand_Def函数	当夹具已被设置时，返回“True”
Hand_Type函数	返回夹具类型的编号
Hand_Label$函数	返回夹具的标签
Hand_Number函数	返回夹具的编号

安全功能相关命令

有关详细信息，请参阅以下手册。
《机器人控制器 安全功能手册》

命令	说明
SF_GetParam函数	返回安全功能参数的信息。
SF_GetParam$函数	返回安全功能参数的文本信息。
SF_GetStatus函数	返回安全功能的状态位。
SF_LimitSpeedS	启用SLS时，设置并显示关于Tool命令所设置位置的速度调整功能的速度调整值。
SF_LimitSpeedS函数	启用SLS时，返回Tool命令所设置位置的速度调整功能的速度调整值。
SF_LimitSpeedSEnable	启用SLS时，设置关于“Tool命令所设置位置的速度调整功能”的On/Off，并显示设置状态。
SF_LimitSpeedSEnable函数	启用SLS时，返回Tool命令所设置位置的速度调整功能的状态。
SF_PeakSpeedS	显示速度监控点的峰值速度值。
SF_PeakSpeedS函数	返回速度监控点的峰值速度值。
SF_PeakSpeedSClear	清除并初始化速度监控点的峰值速度值。
SF_RealSpeedS	显示速度监控点的当前速度。
SF_RealSpeedS函数	返回速度监控点的当前速度。

VRT相关命令

有关详细信息，请参阅以下手册。
《Vibration Reduction Technology》

命令	说明
VRT	选择VRT编号，或显示所选的VRT编号
VRT函数	用于返回当前设置的VRT编号的函数
VRT_Clr	清除VRT功能的设置
VRT_CPMotion	指定CP动作时启用或停用VRT功能
VRT_CPMotion函数	返回CP动作时启用或停用VRT功能的函数
VRT_Def函数	返回所选VRT编号的设置状态的函数
VRT_Description	设置所选VRT编号的注释
VRT_Description$函数	返回所选VRT编号的注释的函数
VRT_Label	设置所选VRT编号的标签
VRT_Label$函数	返回所选VRT编号的标签的函数
VRT_Number函数	返回对应VRT标签的VRT编号的函数
VRT_Set	为各VRT编号设置VRT功能的VRTParam1、VRTParam2
VRT_Set函数	返回各VRT编号设置的VRTParam1、VRTParam2设置值的函数
VRT_Trigger	输出测量触发至VR软件

3. SPEL+ 语言参考

3.1 SPEL+ 语言参考

本章在下述项目说明SPEL+命令。

格式

即显示使用该命令时的格式。根据命令，多种格式同时也表示与“说明”栏相关联的参考编号。
格式中使用符号的说明如下所示。

符号：[ ]
可以省略的参数。
使用该括号的参数有以下类型。

- 用空格分隔的参数
  如果用空格分隔，则可以更改参数的顺序或省略参数。

- 用逗号分隔的参数
  如果用逗号分隔，则顺序是固定的，不能更改。
  此外，可以省略逗号后的所有参数，但不能单独省略最后用逗号分隔的参数之前的参数。
  例：
  Inertia [负载惯性 [,离心率]]
  在本例中，如下所示。
  OK：Inertia [负载惯性]
  OK：Inertia
  NG：Inertia [, 离心率]

符号：{ }
可选参数。
在该括号中 可以选择一个用|分隔的参数。

例：
Exit { Do | For | Function }
在这种情况下，请从Do、For和Function中选择一个。

参数

对该命令参数进行说明。

结果

对该命令返回值进行说明。

返回值

对该函数返回的值进行说明。

说明

对命令功能进行详细说明。

注意

关于该命令说明重要事项。

参照

是与该命令相关联的其他命令清单。想参考相关联的命令时，可从目录中寻找相应页码。

使用示例

表示使用该命令的示例。

3.2 运算符

显示通过SPEL+语言使用的运算符。

运算符	格式示例	说明
+	A+B	加法
-	A-B	减法

运算符	格式示例	说明
*	A*B	乘法
/	A/B	除法
**	A**B	乘方
=	A=B	A等于B
>	A>B	A大于B
<	A<B	A小于B
>=	A>=B	A大于等于B
<=	A<=B	A小于等于B
<>	A<>B	A不等于B
And	A And B	逻辑与
Mod	A Mod B	整数的取模
Not	Not A	非
Or	A Or B	逻辑或
Xor	A Xor B	逻辑异或

运算符的优先顺序

在程序中，按以下顺序进行处理运算符

优先顺序	运算符	格式示例	说明
1	()	(A+B)	括号
2	**	A**B	乘方
3	*	A*B	乘法
	/	A/B	除法
4	Mod	A Mod B	整数的取模
5	+	A+B	加法
	-	A-B	减法
6	=	A=B	A等于B
	<>	A<>B	A不等于B
	<	A<B	A小于B
	>	A>B	A大于B
	<=	A<=B	A小于等于B
	>=	A>=B	A大于等于B
7	Not	Not A	非
8	And	A And B	逻辑与
9	Or	A Or B	逻辑或

优先顺序	运算符	格式示例	说明
10	Xor	A Xor B	逻辑异或

3.3 !

3.3.1 !...! 并行处理

并行处理动作过程中I/O等的输入输出。

格式
动作命令 !并行处理语句!

参数

动作命令
　　记述以下任意可并行处理的命令。
　　Arc, Arc3, Go, Jump, Jump3, Jump3CP, Move, BGo, BMove, TGo, TMove
并行处理语句
　　记述在动作中可并行处理的I/O语句。(参照下表)

说明
在开始动作的同时，开始执行夹在“!”符号之间的语句。例如，由此在机械臂还动作的时候就能执行I/O，而不等待机器人停止动作之后再执行。在机械臂动作中，备有敦促I/O执行的语句。(参照下表“Dn”)

下表汇总了可以使用的所有并行处理语句。表中的语句均可单独使用。并且，可以组合几个语句在一个动作命令中执行多个I/O语句。

Dn	用于在执行以下并行处理语句之前，指定移动量的%，并与动作命令同步。“n”表示用0到100的整数指定的%值，用于指定该动作开始执行并行处理语句的开始位置。就是说，在动作达到总移动量的n%时，将执行Dn以后的语句。 与Jump命令组合使用时，移动量的%值不包括第3关节(上下轴)的垂直移动动作。Jump动作时，仅对除第3关节(上下轴)之外的平行移动动作取得同步。要在完成第3关节的垂直移动动作时执行语句，请在语句的开头加入D0(零)。 与Jump3命令组合使用时，移动量的%值中不包括转移和接近动作。Jump3动作时，仅对跨越动作取得同步。要在完成转移动作时执行语句，请在语句的开头加入D0(零)。 “Dn”在1次并行处理语句中最多可以使用16次。
On / Off n	用于打开和关闭输出位编号“n”。
MemOn / MemOff n	用于打开和关闭存储器I/O的位编号“n”。
Out p,d OpBCD p,q OutW p,d	用于将数据“d”输出到输出端口“p”中。
MemOut p, d MemOutW p,d	用于将数据“d”输出到存储器I/O端口“p”中。
Signal s	用于产生同步信号。
WaitSig s	用于等待信号“s”后处理下一语句。
Wait t	用于等待“t”秒钟并执行下一并行处理语句。

Wait Sw(n) = j	用于等待下一并行处理语句，直至输入位“n”与由“j”定义的条件(On或Off)一致。
Wait MemSw(n) = j	用于等待下一并行处理语句，直至存储器I/O位“n”与由“j”定义的条件(On或Off)一致。
Wait其他条件	还可以有上述2模式以外的Wait语句。详情请参阅Wait。
Print	用于将数据输出到显示装置中。
Print#	用于将数据输出到指定的通信端口中。
外部函数	用于执行由Declare语句声明的外部函数。
Hand_On n Hand_Off n	执行夹具编号“n”的Hand_On/Hand_Off的操作。

注意

- 动作在I/O命令全部结束之前结束时

  如果针对特定动作命令的动作结束而所有并行处理语句并未结束执行，将在这些语句全部结束之后实行下一程序。该情况是为进行必须并行处理多个I/O命令的短距离移动时特别假设的。

- 通过停止机械臂的Till语句在中途结束动作时

  如果在移动过程中使用停止机械臂的Till语句，将视为动作100%完成并执行至D100。在结束所有并行处理语句之前，不会移至动作命令的下一语句。

- 根据AbortMotion语句或Trap在中途结束动作时

  将不执行动作结束后的D语句。

- 设置接近100% 的“n”值时的路径运动减速

  如果在路径运动中设置了较高的“n”值，机器人则可能会进行减速，以结束正在执行的动作。通常，CP On在开始减速的同时开始进行下一动作命令的加速。但是，如果在减速中指定Dn，则在结束该命令之前不会开始下一命令的加速。为避免减速，将处理语句放到动作命令之后。例如，在下例中，将On 1语句的位置从Jump P1动作中的并行处理位置移至Jump P1后。

```
CP On
Jump P1 !D96; On 1!
Go P2

CP On
Jump P1
On 1
Go P2
```

- Jump语句与并行处理

  在上升动作结束后开始执行与Jump语句一起使用的并行处理语句，并在开始下降移动前结束。

  在转移动作结束后开始执行与Jump3语句一起使用的并行处理语句，并在开始接近移动前结束。

- Here语句与并行处理

  不能将Here语句与并行处理同时放在一个动作命令内。

```
Go Here :Z(0) !D10; MemOn 1 !
```

  不能按上述使用方法执行。

```
P999 = Here
Go P999 Here :Z(0) !D10; MemOn 1 !
```

  请变更为上述程序。

参阅

Arc、Arc3、Go、Jump、Jump3、Jump3CP、Move、BGo、BMove、TGo、TMove

!...! 并行处理使用示例

下述为与动作命令一起使用并行处理功能的示例。

与Jump命令一起使用并行处理。第3关节结束上升移动并且第1、2、4关节开始移动时，输出位1会置为ON。输出位1将在Jump动作完成50%的阶段再次关闭。

```
Function test
   Jump P1 !D0; On 1; D50; Off 1!
Fend
```

与Move命令一起使用并行处理。在完成移至P1动作的10%的阶段，打开输出位5，并在0.5秒后关闭输出位5。

```
Function test2
   Move P1 !D10; On 5; Wait 0.5; Off 5!
Fend
```

3.4

3.4.1 #define

用于定义将识别符转换为指定字符串。

格式

#define 识别符 [(参数[,参数])] 转换字符串

参数

识别符

是字符串参数省略的关键字。识别符的规则如下所示。

- 以字母开始，并使用字母数字和下划线(_)进行创建。
- 识别符中不能使用空格和制表符。

参数

指定可以在转换字符串中使用的1个或多个变量，并提供像宏那样的动态定义机制。#在#define命令中最多可以使用8个参数。请用逗号分隔各参数，并将参数列表放到括弧中。

转换字符串

是编译程序时被替换为识别符的转换字符串。与转换字符串相关的规则如下所示。

- 在转换字符串中可以使用空格和制表符。
- 与语句一起使用的识别符不能作为转换字符串使用。
- 如果使用注释符号“ ’ ”，后续字符将被判断为注释，而不包含在转换字符串中。
- 转换字符串可以省略。此时，指定的识别符可以转换为空或Null字符串。由此，将从程序中删除该识别符。

说明

通过#define命令，指定的识别符将在程序内被替换。每次找到指定的识别符，都将转换为转换字符串并进行编译。但是，源代码自身不是转换字符串，而是以包括识别符的形式被保留。这是因为，在读取代码自身时，识别符比转换为字符串的代码更容易读取。

定义的识别符可以作为是否组合#ifdef和#ifndef命令进行编译的条件来使用。

如果指定参数，识别符可以作为宏使用。

注意

- #如果将#define用于变量声明和标签转换，将会出错。

  如果将#define命令用于变量声明，将会出错，敬请注意。

参阅

#ifdef, #ifndef

#define使用示例

```
'如果是调试模式，将取消对以下的行的注释
' #define DEBUG

Input #1, A$
 #ifdef DEBUG
    Print "A$ = ", A$
 #endif
Print "The End"

 #define SHOWVAL(x) Print "var = ", x
```

```
Integer a

a = 25

SHOWVAL(a)
```

3.4.2 #ifdef...#else...#endif

用于进行条件编译。

格式

#ifdef识别符...为进行条件编译的源代码
[#else]...伪条件的源代码
#endif

参数

识别符
是编译#ifdef和#else或#endif之间的源代码、进行用户定义的关键字。这样，此识别符将作为编译条件而起作用。

说明

#ifdef...#else...#endif用于条件编译已选择的源代码。是否进行了编译的条件将根据识别符来确定。首先，#ifdef将检查指定的识别符是否被当前#define定义。#else语句可以省略。

[已定义时] 如果#else语句正在使用，将编译#ifdef和#else之间的语句。#如果未使用#else语句，将编译#ifdef和#endif之间的语句。

[未定义时] 如果#else语句正在使用，将编译#else和#endif之间的语句。#如果未使用#else语句，将跳过该语句而不编译#ifdef和#endif之间的语句。

参阅

#define, #ifndef

#ifdef使用示例

下例为使用#ifdef的程序例。在下例中，可以根据有无#define DEBUG模拟命令的定义来控制是否输出变量A$的值。如果#define DEBUG模拟命令是在源码中使用，以此为例是在记载的部分之前使用，Print A$的行将被编译并且在执行程序时输出。但是，此时将输出字符串“The End”而与#define DEBUG模拟命令无关。

```
'如果是调试模式，将取消对以下的行的注释
' #define DEBUG

Input #1, A$
 #ifdef DEBUG
    Print "A$ = ", A$
 #endif
Print "The End"
```

3.4.3 #ifndef...#endif

用于进行条件编译。

格式

#ifndef识别符...为进行条件编译的源代码
[#else]...条件为真时的源代码
#endif

参数

识别符
是用户定义的关键字。未定义时，从#ifndef到#else或#endif的源代码将被编译。此识别符作为进行编译的条件而起作用。

说明

此命令也叫“如果未定义”命令，将检查与#ifdef相反的条件。#ifndef...#else...#endif用于进行已选择的源代码的条件编译。#else语句可以省略。

[已定义时] 如果#else语句正在使用，将编译#else和#endif之间的语句。#如果未使用#else语句，将不编译#ifndef和#endif之间的语句。

[未定义时] 如果#else语句正在使用，将不编译#else和#endif之间的语句。#如果未使用#else语句，将编译#ifndef和#endif之间的语句。

注意

- #ifdef和#ifndef的差异

  #ifdef用于在已定义识别符时编译指定的源代码。而#ifndef则用于在未定义识别符时编译指定的源代码。

参阅

#define, #ifdef

#ifndef使用示例

下例为使用#ifndef的程序例。在下例中，可以根据有无#define NODELAY模拟命令的定义来控制是否输出变量A$的值。如果#define NODELAY模拟命令在源码中在这里记载的示例部分之前就被使用，则在编译时，Wait 1的行将不会与该程序源码的其他部分一起被编译。如果#define NODELAY模拟命令未在记载的部分之前使用(就是说未定义NODELAY)，Wait 1的行将被编译并之后会在程序中被执行。此时将输出字符串“The End”，而与#define NODELAY模拟命令无关。

```
'如果要延迟，可对以下的行添加注释
 #define NODELAY 1

Input #1, A$
 #ifndef NODELAY
    Wait 1
 #endif
Print "The End"
```

3.4.4 #include

在使用#include语句的位置中插入指定的include文件

格式
#include "include文件名.INC "

参数

include文件名
include文件名请指定在当前项目中有效的include文件名。扩展名均为“.inc”。include文件名指定插入到当前文件中的文件名。

说明
#include用于将指定的include文件的内容插入到使用#include语句的位置中。

请在include文件中插入#define语句和全局变量的声明。

请在函数定义外使用#include语句。

可以在include文件中记述其他include文件。例如，在FILE1中include FILE2，并在FILE2中include FILE3。我们称此为文件的嵌套。

参阅
#define, #ifdef, #ifndef

#include使用示例

```
Include File (Defs.inc)

 #define DEBUG 1
 #define MAX_PART_COUNT 20

Program File (main.prg)

 #include "defs.inc"

Function main
    Integer i

    Integer Parts(MAX_PART_COUNT)

Fend
```

3.4.5 #undef

用于清除#define语句定义的识别符。

格式

#undef识别符名

参数

识别符名
　　#用于指定在define语句中使用的关键字。

参阅

#define, #ifdef, #ifndef

3.5 A

3.5.1 AbortMotion

用于将中断动作命令并执行动作的任务设为错误。

本命令用于高级方。请在充分理解命令规格之后使用。

格式
AbortMotion

参数

机器人编号
　　指定进行中断动作的机器人编号。
All
　　中断所有机器人的动作。

说明
执行AbortMotion后机器人的状态如下所示。

如果无论什么情况都要继续处理，需要用OnErr捕获错误并记述错误处理。

错误2999可以使用常数ERROR_DOINGMOTION。

错误2998可以使用常数ERROR_NOMOTION。

在继续执行(Cont)前，请制作防止执行AbortMotion超过1次的程序。

- 机器人正在执行动作命令时

  机器人将立即暂停(快速暂停)机械臂动作，并撤销剩余动作。

  在已执行的机器人动作命令的任务中，将产生错误2999(ERROR_DOINGMOTION)。

  以下的动作命令将从暂停位置直接移至目标位置。

- 机器人正在暂停(快速暂停)时

  在执行AbortMotion时未发生任何问题，但是将在内部撤销剩余动作。

  在指示继续执行(Cont)时，在已执行的机器人动作命令的任务中，将产生错误2999(ERROR_DOINGMOTION)。

  以下的动作命令将从暂停位置直接移至目标位置。

- 机器人处于WaitRecover状态(打开安全门的状态)时

  在执行AbortMotion时未发生任何问题，但是将在内部撤销剩余动作。

  可以通过Recover命令的标志选择上一个动作。

  - 如果执行“WithMove”动作，将恢复机器人的励磁并执行恢复动作。在指示继续执行(Cont)时，在已执行的机器人动作命令的任务中，将产生错误2999(ERROR_DOINGMOTION)。以下的动作命令将从完成恢复动作的位置直接移至目标位置。
  - 如果执行“WithoutMove”将恢复机器人的励磁。在指示继续执行(Cont)时，在已执行的机器人动作命令的任务中，将产生错误2999(ERROR_DOINGMOTION)。以下的动作命令将从恢复励磁的位置直接移至目标位置。(没有恢复动作。)

- 机器人正在执行动作命令以外的命令时

  在之前已执行的机器人动作命令的任务中，将产生错误2998(ERROR_NOMOTION)。如果任务因为Wait和Input命令而处于待机状态，将立即中断任务并发生错误2998。

  如果按照CP On的设置执行动作命令并且程序从动作命令中退出，即使机器人正在动作，也会发生错误2998。

- 指定的机器人未按照程序(任务)动作时

  将产生错误。

注意

- 支持的控制器型号

  不支持T/VT系列。

参阅

OnErr、Recover、Till

AbortMotion使用示例

如果打开存储器I/O的0号，将执行AbortMotion，并且机器人移至Home位置。

```
Function main
  Motor On
  Xqt sub, NoEmgAbort
  OnErr GoTo errhandle

  Go P0
  Wait Sw(1)
  Go P1

  Quit sub
  Exit Function

errstart:
  Home
  Quit sub
  Exit Function

errhandle:
  Print Err
  If Err = ERROR_DOINGMOTION Then
    Print "机器人正在动作"    '正在执行Go P0和 Go P1
    EResume errstart
  ElseIf Err = ERROR_NOMOTION Then
    Print "机器人未动作"    '正在执行Wait Sw(1)
    EResume errstart
  EndIf

  Print "Error Stop"    '发生其他错误
  Quit All
Fend

Function sub
  MemOff 0
  Wait MemSw(0)
  AbortMotion 1
  MemOff 0
Fend
```

3.5.2 Abs函数

是返回绝对值的函数。

格式
Abs (数值)

参数

数值
以表达式或直接以数值进行指定。

返回值
返回已指定数值的绝对值。

说明
绝对值是指无符号的数值。例如，Abs(-1)和Abs(1)，两者都将返回1。

参阅
Atan, Atan2, Cos, Int, Mod, Not, Sgn, Sin, Sqr, Str$, Tan, Val

Abs函数使用示例
下例通过命令窗口使用Print命令执行函数。

```
> print abs(1)
1
> print abs(-1)
1
> print abs(-3.54)
3.54
>
```

3.5.3 Accel

用于设置和显示利用Go、Jump、Pulse等的PTP动作的加减速度。

格式

Accel 加速设置值, 减速设置值 [, 转移加速设置值, 转移减速设置值, 接近加速设置值, 接近减速设置值]

参数

加速设置值
　　以大于1的整数指定相对于最大加速度的比例。(单位：%)
减速设置值
　　以大于1的整数指定相对于最大减速度的比例。(单位：%)
转移加速设置值
　　以大于1的整数指定Jump时的转移加速度。可省略。仅Jump命令时可设置。
转移减速设置值
　　以大于1的整数指定Jump时的转移减速度。可省略。仅Jump命令时可设置。
接近加速设置值
　　以大于1的整数指定Jump时的接近加速度。可省略。仅Jump命令时可设置。
接近减速设置值
　　以大于1的整数指定Jump时的接近减速度。可省略。仅Jump命令时可设置。

结果

如果省略参数，将返回当前的Accel参数。

说明

Accel用于设置所有PTP动作(利用Go、Jump、Pulse等命令发生的动作)的加减速度。

以大于1的整数值设置Accel设置的加减速度参数。此数值显示出相对于最大加速度(或减速度)的比例。通常100是最大值，但是有的机器人可能有超过100的设置。AccelMax函数用于返回可以进行Accel设置的最大值。

Accel用于重新设置加减速度、以及单纯输出当前设置值时。如要重新设置加速度和减速度后使用Accel，将需要最初的2个参数(加速设置值和减速设置值)。

转移加速设置值、转移减速设置值、接近加速设置值、接近减速设置值等4个参数仅在Jump命令时有效，可以省略。这些参数用于指定Jump动作开始时的转移动作和Jump动作结束时的接近动作的、各自的加速设置值和减速设置值。

下述某种情况时，Accel值会被初始化。

- 控制器电源ON
- 执行Motor On
- 执行SFree、SLock、Brake
- 执行Reset、Reset Error
- 利用停止按钮或执行Quit All等结束任务

注意

在低功率模式(Power Low)执行Accel命令

在低功率模式(Power Low)时执行Accel，将会保存新值，而当前值将被限制在较低水平。TEACH模式为OFF、功率设为High时，Accel将变为有效。

Accel与AccelS的差异

Accel命令不是设置直线和曲线动作的加减速度的命令。而AccelS命令则用于设置直线和曲线动作的加减速度。

超过100的Accel设置

Accel设置一般以100为最大值，但是有的机型可以设置到100以上。在正常使用中，Accel设置值为“100”时，是加减速度与定位时的振动达到平衡状态的最佳设置值。但有时会根据动作条件，以缩短循环时间为优先条件，此时可通过减小定位时的振动来提高加减速度。此时，请将Accel设置值设为100以上的值。但是，根据动作条件，即使设置了100以上的值，也可能出现循环时间不变化的情况。

参阅

AccelR、AccelS、Go、Jump、Jump3、Power、Pulse、Speed、TGo

Accel使用示例

下例为使用Accel和Speed的简单的动作程序例。Accel和Speed中使用预先定义的变量。

[例1]

```
Function acctest
  Integer slow, accslow, decslow, fast, accfast, decfast

  slow = 20      '低速的设置
  fast = 100     '高速的设置
  accslow = 20   '低加速度的设置
  decslow = 20   '低减速度的设置
  accfast = 100 '高加速度的设置
  decfast = 100 '高减速度的设置

  Accel accslow, decslow
  Speed slow
  Jump pick
  On gripper
  Accel accfast, decfast
  Speed fast
  Jump place
  .
  .
  .
Fend
```

[例2]

此例所示为Jump命令时减慢第3关节的下降减速度，以谨慎地处理部件。在此例中，需要在设置Accel时将第3关节下降减速设置值参数设得低一些。

```
>Accel 100,100,100,100,100,35

>Accel
    100     100
    100     100
    100     35
>
```

3.5.4 Accel函数

是用作返回当前的加减速度设置值的函数。

格式

Accel (设置值编号)

参数

设置值编号

以整数值指定下述各值。

- 1：加速设置值
- 2：减速设置值
- 3：Jump动作时的转移加速设置值
- 4：Jump动作时的转移减速设置值
- 5：Jump动作时的接近加速设置值
- 6：Jump动作时的接近减速设置值

返回值

返回大于1的整数(%)。

参阅

Accel

Accel函数使用示例

如下所示为程序中使用Accel函数的示例。

```
Integer currAccel, currDecel

'获取当前的加减速度
currAccel = Accel(1)
currDecel = Accel(2)
Accel 50, 50
SRVJump pick
'恢复以前的设置
Accel currAccel, currDecel
```

3.5.5 AccelMax函数

是用于返回Accel可以设置的加减速度的最大值的函数。

格式

AccelMax (最大值编号)

参数

最大值编号

以整数值指定下述各值。

- 1：加速度最大值
- 2：减速度最大值
- 3：Jump动作时的转移加速度最大值
- 4：Jump动作时的转移减速度最大值
- 5：Jump动作时的接近加速度最大值
- 6：Jump动作时的接近减速度最大值

返回值

返回大于1的整数(%)。

参阅

Accel

AccelMax函数使用示例

如下所示为程序中使用AccelMax函数的示例。

```
'显示最大加减速度
Print AccelMax(1), AccelMax(2)
```

3.5.6 AccelR

用于设置和显示有关CP动作时工具姿势变化的加减速度。

格式

(1) AccelR 加速设置值 [, 减速设置值]

(2) AccelR

参数

加速设置值

以实值(0.1～5000)指定加速设置值。(单位：deg/sec^2)

减速设置值

以实值(0.1～5000)指定减速设置值。(单位：deg/sec^2)

参数的有效设置值

	加速设置值/减速设置值
VT系列	0.1～1000
C系列、N系列、T系列、G系列、GX系列、RS系列、LS-B系列	0.1～5000

(单位：deg/sec^2)

结果

如果省略参数，则显示当前的AccelR设置值。

说明

AccelR仅在Move、Arc、Arc3、BMove、TMove、Jump3CP中使用ROT修饰参数时有效。

在下述任何一种情况下，AccelR值将被初始化。

- 控制器电源ON
- 执行Motor On
- 执行SFree、SLock、Brake
- 执行Reset、Reset Error
- 利用停止按钮或执行Quit All等结束任务

参阅

Arc、Arc3、BMove、Jump3CP、Power、SpeedR、TMove

AccelR使用示例

```
AccelR 360, 200
```

3.5.7 AccelR函数

返回指定工具姿势变化的加减速度的设置值。

格式

AccelR (设置值编号)

参数

设置值编号

以表达式或数值指定下述各值。

- 1：加速度设置值
- 2：减速度设置值

返回值

以实值(单位：deg/sec^2)返回加速或减速设置值。

参阅

AccelR

AccelR函数使用示例

```
Real currAccelR, currDecelR

'获取当前的加减速度
currAccelR = AccelR(1)
currDecelR = AccelR(2)
```

3.5.8 AccelS

用于设置机器人的直线动作和CP动作的加减速度。(请参阅Move、Arc、Arc3、Jump3、CVMove。)

格式

(1) AccelS 加速设置值 [, 减速设置值] [,转移加速设置值, 转移减速设置值, 接近加速设置值, 接近减速设置值]
(2) AccelS

参数

加速设置值

以实值指定直线动作或CP动作时的加速度(单位：mm/sec^2)。如果省略减速设置值，在加速时和减速时都将应用加速设置值。

减速设置值

以实值指定直线动作或CP动作时的减速度(单位：mm/sec^2)。可省略。

转移加速设置值

以实值指定Jump3时和Jump3CP时的转移动作的转移加速度(单位：mm/sec^2)。可省略。

转移减速设置值

以实值指定Jump3时和Jump3CP时的转移动作的转移减速度(单位：mm/sec^2)。可省略。

接近加速设置值

以实值指定Jump3时和Jump3CP时的接近动作的接近加速度(单位：mm/sec^2)。可省略。

接近减速设置值

以实值指定Jump3时和Jump3CP时的接近动作的接近减速度(单位：mm/sec^2)。可省略。

参数的有效设置值 (单位：mm/sec^2)

	加速设置值/减速设置值 转移加速设置值/转移减速设置值 接近加速设置值/接近减速设置值
N2	0.1～5000
LS20-B, T系列, VT系列	0.1～10000
C4-*901**	0.1～15000
C4-*601**, C8-*1401**, G系列, GX系列, RS系列, LS3-B, LS6-B, LS10-B, C8-*701**W, C8-*901**W, N6, C12	0.1～25000
C8-*701**, C8-*701**R, C8-*901**, C8-*901**R	0.1～35000

结果

如果省略参数，则显示加速值和减速值。

显示加速度值和减速度值时，对于加速设定值、减速设定值、退避加速设定值、退避减速设定值、接近加速设定值、接近减速设定值的，各自的显示当前夹具质量校正的加速度值和减速度值。

说明

AccelS用于实值包括直线和曲线的所有曲线动作的加速和减速。包括基于Move和Arc命令的动作。

在下述任何一种情况下，AccelS值会被初始化，加减速度会变慢。

- 控制器电源ON
- 执行Motor On
- 执行SFree、SLock、Brake
- 执行Reset、Reset Error

- 利用停止按钮或执行Quit All等结束任务

注意

- 在低功率模式(Power Low)执行AccelS命令

  在低功率模式(Power Low)时使用AccelS，将会保存新值，而当前值将被限制在较低水平。

  TEACH模式为OFF、功率设为High时，AccelS将变为有效。

- Accel与AccelS的差异

  AccelS命令并非是设置PTP动作(Go 和Jump命令的动作)的加减速度。而Accel命令则用于设置PTP动作的加减速度。

- 上限值

  SCARA机器人(包括RS系列)AccelS的上限值因Weight设置以及花键单元的位置而异。有关详细信息，请参阅以下手册。
  《机械手手册 - 设置CP动作时的ACCELS》

  垂直六轴型机器人的AccelS的上限值因Weight设置而异。有关详细信息，请参阅以下手册。
  《机械手手册 - 规格表》

参阅

Accel、Arc、Arc3、Jump3、Jump3CP、Power、Move、TMove、SpeedS

AccelS使用示例

下述为使用预先定义的变量设置Move命令的AccelS和SpeedS的简单的动作程序例。

```
Function acctest
  Integer slow, accslow, fast, accfast

  slow = 20      '设置低速
  fast = 100     '设置高速
  accslow = 200 '设置低加速度
  accfast = 5000         '设置高加速度
  AccelS accslow
  SpeedS slow
  Move P1
  On 1
  AccelS accfast
  SpeedS fast
  Jump P2
  .
  .
  .
Fend
```

3.5.9 AccelS函数

是用作返回CP动作的加减速度设置值的函数。

格式

AccelS (设置值编号)

参数

设置值编号

以整数值或表达式指定下述各值。

- 1：加速设置值
- 2：减速设置值
- 3：Jump3时和Jump3CP时的转移加速设置值
- 4：Jump3时和Jump3CP时的转移减速设置值
- 5：Jump3时和Jump3CP时的接近加速设置值
- 6：Jump3时和Jump3CP时的接近减速设置值
- 7：根据末端夹具重量补偿的加速度值
- 8：根据末端夹具重量补偿的减速度值
- 9：根据末端夹具重量补偿的Jump3和Jump3CP时的退避加速值
- 10：根据末端夹具重量补偿的Jump3和Jump3CP时的退避减速值
- 11：根据末端夹具重量补偿的Jump3和Jump3CP时的接近加速值
- 12：根据末端夹具重量补偿的Jump3和Jump3CP时的接近减速值

返回值

返回加速设置值或减速设置值(0～5000的实值，单位：mm/s^2)。

参阅

AccelS、Arc3、SpeedS、Jump3、Jump3CP

AccelS函数使用示例

```
Real savAccelS

savAccelS = AccelS(1)
```

3.5.10 Acos函数

用于返回指定数值的反余弦。

格式
Acos (数值)

参数

数值
以实值指定角度的余弦。

返回值
用于以实值(单位：弧度)返回指定数值的反余弦。

说明
Acos用于返回指定数值的反余弦。指定的数值在-1～1的范围内。返回值的范围为0～π。如果数值小于-1或大于1，将会出错。

要将弧度转换为角度时，需要使用RadToDeg函数。

参阅
Abs、Asin、Atan、Atan2、Cos、DegToRad、RadToDeg、Sgn、Sin、Tan、Val

Acos函数使用示例

```
Function acostest
  Double x

  x = Cos(DegToRad(30))
  Print "Acos of ", x, " is ", Acos(x)
Fend
```

3.5.11 Agl函数

是返回指定旋转关节的角度或移动关节的位置的函数。

格式
Agl (关节编号)

参数

关节编号
以整数值指定关节编号。范围是1～机器人的关节数。附加轴的S轴为8，T轴为9。

返回值
返回指定旋转关节的角度或移动关节的位置。

说明
Agl函数用于获取指定旋转关节的角度或移动关节的位置。

如果指定的关节是旋转关节，将以“度”为单位的实值返回从指定关节的角度0算起的旋转角度。

如果指定的关节是移动关节，将以“mm”为单位的实值返回从该关节的位置0算起的移动量。

如果通过Arm语句设置增设机械臂并选择了该增设机械臂，Agl函数将用于返回从该增设机械臂的标准机械臂的脉冲0位置算起的角度或位置。

参阅
PAgl、Pls、PPls

Agl函数使用示例
如下所示为通过命令窗口使用Print命令的示例。

```
> print agl(1), agl(2)
 17.234  85.355
```

3.5.12 AglToPls函数

用于将机器人各关节的角度转换为脉冲。

格式

AglToPls(关节位置1, 关节位置2, 关节位置3, 关节位置4 [, 关节位置5, 关节位置6] [, 关节位置7] [, 关节位置8, 关节位置9])

参数

关节位置1~6
　　以实值指定关节的角度。
关节位置7
　　以实值指定第7关节的角度。只在关节型7轴机器人上使用。
关节位置8
　　以实值指定附加轴S关节的角度。
关节位置9
　　以实值指定附加轴T关节的角度。

返回值

返回根据各关节位置计算的关节脉冲。

说明

使用AglToPls函数将关节角度变换为关节脉冲。

注意

在AglToPls函数制定的关节位置所实现的机器人形态为特异姿势时，如果将AglToPls的结果代入到点变量中，将会丢失指定关节位置的部分信息。其结果，如果使用代入的点变量进行动作，机器人将在与AglToPls指定的关节位置不同的关节位置上动作。在下例中，P1将在关节位置(0, 0, 0, 0, 0, 90)上动作。

```
P1 = AglToPls(0, 0, 0, 90, 0, 0)
Go P1
```

同样，直接将AglToPls指定为CP动作命令的自变量时，机器人将在与指定的关节位置不同的关节位置上动作。在下例中，将在关节位置(0, 0, 0, 0, 0, 90)上动作。

```
Move AglToPls(0, 0, 0, 90, 0, 0)
```

如果使用AglToPls函数作为PTP动作命令的自变量，则不会有这种奇点的问题。

参阅

Agl、JA、Pls

AglToPls函数使用示例

```
Go AglToPls(0, 0, 0, 90, 0, 0)
```

3.5.13 AIO_In函数

用于从作为选项的模拟I/O输入通道读取模拟值。

格式
AIO_In (通道编号)

参数

通道编号
　　指定模拟I/O的通道编号。

返回值
以实数返回由通道编号指定的模拟I/O通道的模拟输入值。返回值的范围取决于模拟IO电路板的输入范围设置。

说明
In函数

参阅
AIO_InW函数、AIO_Out、AIO_OutW、AIO_Out函数、AIO_OutW函数、AIO_Set、Wait

AIO_In函数使用示例

```
Function main
    Real var1
    var1 = AIO_In(2)  '用于获取模拟输入通道2的输入状态
    If var1 > 5.0 Then
        Go P1
        Go P2
        '在此处执行其它动作命令
        '.
        '.
    Else
        Print "Error in initialization!"
        Print "Sensory Inputs not ready for cycle start"
        Print "Please check analog inputs 2."
    EndIf
Fend
```

3.5.14 AIO_InW函数

用于从作为选项的模拟I/O输入通道读取模拟值。

格式
AIO_InW (通道编号)

参数

通道编号
　　指定模拟I/O的通道编号。

返回值
返回指定的模拟I/O通道的输入状态(0～65535的Long型整数)。

根据模拟I/O板卡的输入范围的设置，各输入通道的输入电压(电流)与返回值的对应关系，如下表所示。

输入数据		输入范围设置				
16进制数	10进制数	±10.24(V)	±5.12(V)	0-5.12 (V)	0-10.24 (V)	0-24 (mA)
0xFFFF	65535	10.23969	5.11984	5.12000	10.24000	24.00000
0x8001	32769	0.00031	0.00016	2.56008	5.12016	12.00037
0x8000	32768	0.00000	0.00000	2.56000	5.12000	12.00000
0x0000	0	-10.24000	-5.12000	0.00000	0.00000	0.00000

参阅
AIO_In函数、AIO_Out、AIO_OutW、AIO_Out函数、AIO_OutW函数、AIO_Set、Wait

AIO_In函数使用示例

```
Long word0

word0 = AIO_InW(1)
```

3.5.15 AIO_Out

用于从作为选项的模拟I/O输出通道输出模拟值。

格式

AIO_Out 通道编号, 输出数据 [, Forced]

参数

通道编号
　　指定模拟I/O的通道编号。
输出数据
　　以表达式或数值指定表示要输出的电压[V]或电流值[mA]的Real型实数。
Forced
　　可省略。通常会省略。

说明

用于将表示指定电压[V]或电流[mA]的Real值，输出到由通道编号指定的模拟输出端口中。利用板上的开关，设置模拟输出端口的电压输出范围、电压/电流输出选择。已指定模拟IO板输出范围设置范围以外的值时，将输出不超出范围的极限值(最大值、最小值)。

通过指定的通道输出机器人的速度信息时，AIO_Out命令会发生错误。请停止速度信息输出，然后执行AIO_Out。

注意

- Forced标志

  在要通过NoPause任务、NoEmgAbort任务(执行Xqt时指定NoPause或NoEmgAbort以开始的特别任务)，在紧急停止期间或安全门打开时将I/O输出设为ON的情况下，指定此标志。

  紧急停止期间或安全门打开时，模拟I/O输出会发生变化，因此在系统设计方面需要注意。

参阅

AIO_In函数、AIO_OutW、AIO_Out函数、AIO_OutW函数、AIO_Set

AIO_Out使用示例

从模拟I/O通道#1输出7.0[V]。

```
AIO_Out 1, 7.0
```

3.5.16 AIO_Out函数

用于以实值返回通过作为选项的模拟I/O输出通道输出的模拟值。

格式
AIO_Out (通道编号)

参数

通道编号
　　指定模拟I/O的通道编号。

返回值
以实值返回指定模拟I/O通道的电压/电流输出状态。电压输出时的单位为[V]，电流输出时的单位为[mA]。

通过指定的通道输出机器人的速度信息时，也可以执行本函数。

参阅
AIO_In函数、AIO_Out、AIO_OutW、AIO_OutW函数、AIO_Set、Wait

AIO_Out函数使用示例

```
Real rdata01

rdata01 = AIO_Out(1)
```

3.5.17 AIO_OutW

用于从作为选项的模拟I/O输出通道输出16位模拟值。

格式
AIO_OutW 通道编号, 输出数据 [, Forced]

参数

通道编号
　　指定模拟I/O的通道编号。
输出数据
　　利用表达式或数值指定输出数据(0～65535的整数)。
Forced
　　可省略。通常会省略。

说明
用于输出到由通道编号指定的模拟I/O通道中。利用表达式或数值指定输出数据(0～65535的整数)。

如下所示为根据利用板上的开关设置的输出范围设置的输出电压(电流)。

输出数据		输出范围设置					
16进制数	10进制数	±10(V)	±5(V)	0-5 (V)	0-10 (V)	4-20 (mA)	0-20 (mA)
0xFFFF	65535	9.99970	4.99985	5.00000	10.00000	20.00000	20.00000
0x8001	32769	0.00031	0.00015	2.50008	5.00015	12.00024	10.00031
0x8000	32768	0.00000	0.00000	2.50000	5.00000	12.00000	10.00000
0x0000	0	-10.00000	-5.00000	0.00000	0.00000	4.00000	0.00000

注意

- Forced标志

  在要通过NoPause任务、NoEmgAbort任务(执行Xqt时指定NoPause或NoEmgAbort以开始的特别任务)，在紧急停止期间或安全门打开时将I/O输出设为ON的情况下，指定此标志。

  紧急停止期间或安全门打开时，模拟I/O输出会发生变化，因此在系统设计方面需要注意。

参阅
AIO_In函数、AIO_Out、AIO_Out函数、AIO_OutW函数、AIO_Set、Wait

AIO_OutW使用示例

```
AIO_OutW 1, &H8000
```

3.5.18 AIO_OutW函数

用于以0～65535的Long型整数返回通过作为选项的模拟I/O输出通道输出的模拟值。

格式
AIO_OutW (通道编号)

参数

通道编号
　　指定模拟I/O的通道编号。

返回值
以0～65535的Long型整数返回指定模拟I/O通道的输出状态。

根据模拟I/O板卡的输出范围的设置，各输出通道的输出电压(电流)与返回值的对应关系，如下表所示。

输出数据		输出范围设置					
16进制数	10进制数	±10(V)	±5(V)	0-5 (V)	0-10 (V)	4-20 (mA)	0-20 (mA)
0xFFFF	65535	9.99970	4.99985	5.00000	10.00000	20.00000	20.00000
0x8001	32769	0.00031	0.00015	2.50008	5.00015	12.00024	10.00031
0x8000	32768	0.00000	0.00000	2.50000	5.00000	12.00000	10.00000
0x0000	0	-10.00000	-5.00000	0.00000	0.00000	4.00000	0.00000

通过指定的通道输出机器人的速度信息时，也可以执行本函数。

参阅
AIO_In函数、AIO_Out、AIO_OutW、AIO_Out函数、AIO_Set、Wait

AIO_OutW函数使用示例

```
Long word0

word0 = AIO_OutW(1)
```

3.5.19 AIO_Set

用于将机器人的速度信息输出到作为选项的模拟I/O输出通道中。

格式

(1) AIO_Set 通道编号, On, {RefTCPSpeed | RealTCPSpeed | RefECPSpeed | RealECPSpeed }, 最大输出时的速度 [, 最小输出时的速度] [, Cnv , 传送带编号]
(2) AIO_Set 通道编号, Off
(3) AIO_Set [通道编号]

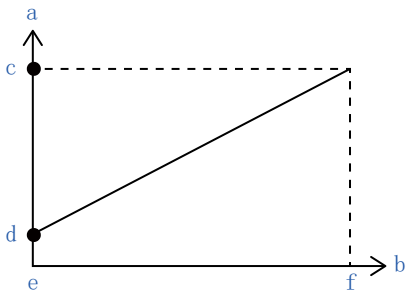
参数

通道编号
　　指定模拟I/O的通道编号。
On
　　利用表达式或数值指定输出数据(0～65535的整数)。
Off
　　结束速度信息的模拟输出并初始化为输出0。
RefTCPSpeed
　　用于输出当前选择的TCP的指令速度。
RealTCPSpeed
　　用于输出当前选择的TCP的实际速度。
RefECPSpeed
　　用于输出当前选择的ECP的指令速度。
RealECPSpeed
　　用于输出当前选择的ECP的实际速度。
最大输出时的速度
　　利用表达式或数值指定表示对输出范围最大值进行输出时速度的Real型实值(单位[mm/s])。
最小输出时的速度
　　利用表达式或数值指定表示对输出范围最小值进行输出时速度的Real型实值(单位[mm/s])。可省略。省略时为“0”[0 mm/s]。
Cnv
　　输出和传送带的相对TCP速度。和传送带编号一起指定。
传送带编号
　　指定用于计算相对TCP速度的传送带编号。

说明

用于以模拟电压/电流，将当前选择的机器人的TCP(工具中心点)或ECP(外部控制点)的速度实时输出到由通道编号指定的模拟I/O通道中。利用模拟I/O电路板上的开关与跨接线，进行模拟电压/电流的选择与输出范围的设置。

根据指定的最小输出时的速度与最大输出时的速度，通过下图所示的直线动作，确定对应于输出范围最小值与最大值的机器人速度。



符号	说明
a	顶端速度[mm/s]
b	输出[V or mA]
c	最大输出时的速度

符号	说明
d	最小输出时的速度
e	最小输出
f	最大输出

已指定(RefTCPSpeed或RefECPSpeed)指令速度时，将根据赋予机器人的指令值输出理想的速度波形。

已指定(RealTCPSpeed或RealECPSpeed)实际速度时，将输出根据机器人的实际动作(各关节的编码器值)运算得出的速度波形。

已指定(RefTCPSpeed或RealTCPSpeed)TCP时，将输出当前选择的工具(默认Tool 0)的中心点速度。

已指定(RefECPSpeed或RealECPSpeed)ECP时，将输出当前选择的外部控制点(ECP)的速度。如果未选择ECP(ECP = 0时)，将始终进行最小输出。

ECP无法和Cnv同时指定。

只能指定已在机械手上设置好的，并且完成传送带校准的传送带编号。

仅指定通道编号时，将显示指定的模拟I/O通道的输出设置信息。已省略所有自变量时，将显示所有模拟I/O通道的输出设置信息。

参阅

AIO_In函数、AIO_Out、AIO_Out函数、AIO_Out、AIO_OutW函数、AIO_Set、Wait

AIO_Set使用示例

在模拟输出通道1中设置机器人1、工具1的TCP实际速度输出。

对机器人动作中的速度进行模拟输出之后，解除速度输出设置。

```
Robot 1
Tool 1
Motor On
Power High
SpeedS 2000
AccelS 5000
AIO_Set 1, On, RealTCPSpeed, 2000.0, 0.0
Move P1
AIO_Set 1, Off
```

3.5.20 AIO_Set函数

用于返回作为选项的模拟I/O输出通道中设置的机器人速度输出设置信息。

格式

AIO_Set (通道编号、索引)

参数

通道编号
　　指定模拟I/O的通道编号。
索引
　　以整数值指定要获取的设置信息的索引。

返回值

可利用AIO_Set函数获取的信息如下所示。

索引	信息
1	On(1) / Off(0)
2	RefTCPSpeed(0)/ RealTCPSpeed(1)/ RefECPSpeed(2)/ RealECPSpeed(3)
3	最大输出时的速度 [mm/s]
4	最小输出时的速度 [mm/s]
5	传送带编号 未设置(0) / 传送带编号 (1~16)

参阅

AIO_In函数、AIO_Out、AIO_OutW、AIO_Out函数、AIO_OutW函数、AIO_Set、Wait

AIO_Set函数使用示例

```
Print "Analog Ch#1 speed output is: ", AIO_Set(1, 1)
```

3.5.21 AIO_TrackingSet

用于设置距离跟踪功能。

格式

(1) AIO_TrackingSet
通道编号, 测量值与距离的转换系数, 距离0 mm的测量值, 可跟踪范围的下限值, 可跟踪范围的上限值 [,可跟踪范围以外的动作 [,进行距离跟踪的轴]]
(2) AIO_TrackingSet
通道编号

参数

通道编号
以1～8的整数值指定连接所使用的距离传感器的模拟I/O的通道编号。
测量值与距离的转换系数
是用于将距离传感器的测量值(V、mA)转换为距离(mm)的系数。以0以外的-500～500的实数指定。(单位：mm/V，mm/mA)
距离0 mm的测量值
以下述范围的实数指定距离(位移测量仪时：位移量)为0 mm时的电压或电流值。(单位：V，mA)
用于设置模拟I/O电路板的输入范围设置的范围内的值。

输入范围设置	最小值	最大值
±10.24 V	-10.24 V	10.24 V
±5.12 V	-5.12 V	5.12 V
0-5.12 V	0 V	5.12 V
0-10.24 V	0 V	10.24 V
0-24 mA	0 mA	24 mA

可跟踪范围的下限值
可跟踪范围的下限值是指执行距离跟踪功能时容许的位移量下限值。以-300～300的实数指定下限值。(单位：mm)
请指定距离传感器可测量范围下限值以上的值。将可跟踪范围下限值指定为小于可跟踪范围上限值的值。
可跟踪范围的上限值
可跟踪范围的上限值是指执行距离跟踪功能时容许的位移量上限值。以-300～300的实数指定上限值。(单位：mm)
请指定距离传感器可测量范围上限值以下的值。将可跟踪范围上限值指定为大于可跟踪范围下限值的值。
可跟踪范围以外的动作
处于可跟踪范围(上述下限值与上限值之间)以外时，以0～1的整数指定停止还是继续机器人的动作。可省略。如果省略，将设置为“0”。常数如下所示。

常数	值	内容
AIOTRACK_ERRSTOP	0	在可跟踪范围以外时，机器人发生错误停止。
AIOTRACK_CONTINUE	1	在可跟踪范围以外时，继续进行机器人动作。

进行距离跟踪的轴
以0～5的整数指定进行距离跟踪的轴。请指定与使用的距离传感器测量方向一致的轴。
可省略。如果省略，将设置为“2”。常数如下所示。

常数	值	内容
AIOTRACK_TOOL_X	0	Tool坐标X轴
AIOTRACK_TOOL_Y	1	Tool坐标Y轴
AIOTRACK_TOOL_Z	2	Tool坐标Z轴
AIOTRACK_ECP_X	3	ECP坐标X轴

常数	值	内容
AIOTRACK_ECP_Y	4	ECP坐标Y轴
AIOTRACK_ECP_Z	5	ECP坐标Z轴

仅在外部控制点动作(ECP)选项有效时才可指定3～5。

结果

如为格式2，将在控制台中显示当前设置值。

如下所示为上述参数名称与控制台中显示的参数名称的对应表。

参数名称	控制台显示名称
测量值与距离的转换系数	ScaleFactor
距离0 mm的测量值	RefVoltage
可跟踪范围的下限值	ThresholdMin
可跟踪范围的上限值	ThresholdMax
可跟踪范围以外的动作	OutOfRangeMode
进行距离跟踪的轴	TrackingAxis

显示示例如下所示。

例1：已设置通道编号1时

```
Ch1: ScaleFactor 1.000[V/mm or mA/mm] RefVoltage 0.000 [V or mA] ThresholdMin -10.000[mm]
ThresholdMax 10.000[mm] OutOfRangeMode AIOTRACK_ERRSTOP TrackingAxis AIOTRACK_TOOL_Z
```

例2：未设置通道编号1时

```
Ch1: Undefined
```

说明

AIO_TrackingSet用于设置距离跟踪功能的参数。要设置的参数取决于使用的距离传感器或实施本功能的环境。打开控制器电源之后，务必在执行AIO_TrackingStart之前执行AIO_TrackingSet。在机器人控制器的电源置为OFF或重新启动之前，会保持已设置的参数值。

下面对参数进行详细说明。

测量值与距离的转换系数

为表示2 mm/V的位移的距离传感器时，转换系数为2。此时，+2 mm为距离延长方向的位移量。针对距离较短方向的位移，位移测量仪的电压可能被设为正电压。在这种情况下，转换系数为负值。

距离0 mm的测量值

如为距离传感器(尤其是位移测量仪)，距离为0 mm时的电压或电流值因各产品而异。另外，也有可通过用户的设置任意设置距离为0 mm时的电压或电流值的产品。请根据使用的距离传感器的设置指定数值。距离(或位移量)为0 mm时，如果距离传感器的输出电压为0V，本参数将为“0”。

可跟踪范围的上/下限值

根据应用程序容许的偏差设置上/下限值。请务必将要设置的值设为距离传感器可测量范围以内的值。距离传感器的可测量范围因各传感器或用户设置而异。实施距离跟踪功能之前，请务必进行确认。如果本参数被设为距离传感器的可测量范围以外的值，距离跟踪功能将无法正确发挥作用，机器人也可能执行意想不到的动作。

可跟踪范围以外的动作

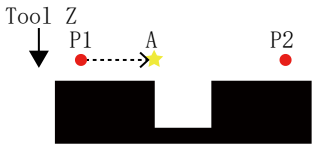
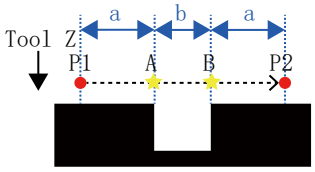
下图所示为对Tool的Z方向实施距离跟踪功能时，将“可跟踪范围以外的动作”参数设置设为“0”以及设为“1”时的机器人移动轨迹。

- P1：距离跟踪开始位置
- P2：目标位置

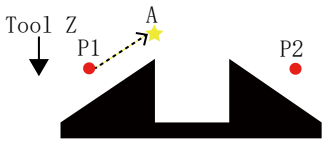
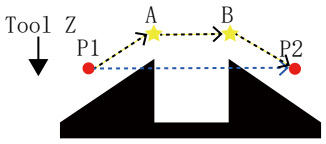
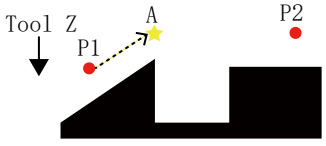
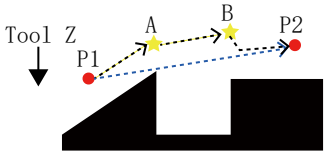
图中的A点所示为超出测量范围的对象物；B点所示为返回到测量范围内的对象物。

距离跟踪功能以功能启用位置P1的Tool Z方向的测量值(位移量)为基准值对机器人进行控制，以确保测量值始终为基准值。因此，从P1向P2移动后，机器人在P1至A点期间保持固定的测量值。

到达A点时，如果设置为“0”，机器人将在A点错误停止。如果设置为“1”，机器人将从A点继续向P2移动。但是，因为处于可跟踪范围以外，因此不会跟踪。由于从B点开始进入到可跟踪范围内，因此与从P1向A点移动期间相同，机器人在移动时保持固定的测量值。

0：机器人在范围外停止动作	1：机器人在范围外继续动作
	
	■ a：距离跟踪功能 有效 ■ b：范围外、距离跟踪功能 无效

设置为“1”时的可跟踪范围以外的机器人动作为：在进行CP动作的轨道上从开始位置(P1)向目标位置(P2)移动。在下图当中，可跟踪范围以外的A-B点之间的轨道与P1-P2之间的轨道平行。到达B点之后，即进入到可跟踪范围以内，因此以测量值为基准值的机器人控制开始启动，机器人可能突然移动。

0：机器人在范围外停止动作	1：机器人在范围外继续动作
	
	

⚠ 注意

如果未正确设置各参数，在执行AIO_TrackingStart时，机器人可能做出意想不到的动作。

请根据使用的设备或实施环境适当地进行设置。

出现异常动作时，请立即按下紧急停止按钮。

参阅

AIO_TrackingStart、AIO_TrackingEnd、AIO_TrackingOn函数

AIO_TrackingSet使用示例

以P1为动作开始位置、以P2为动作结束位置并使用距离跟踪功能移动机器人的程序示例。

⚠ 注意

使用示例中设置的参数为参考值。

因设置的参数或动作环境而导致作业未成功时，可能做出带有振动的动作。

出现异常动作时，请立即按下紧急停止按钮。

```
Function Main
  Motor On
  Power High
  SpeedS 30
  AccelS 300300

  Go P1     ' 移动到开始位置P1
  AIO_TrackingSet 1,1,0,-5,5,0,2     ' 设置距离跟踪功能
  AIO_TrackingStart 1,5,5,5     ' 启用距离跟踪功能
  Move P2     ' 在执行距离跟踪功能的同时移动到P2
  AIO_TrackingEnd     ' 退出距离跟踪功能
  Motor Off
Fend
```

3.5.22 AIO_TrackingStart

用于启用距离跟踪功能。

格式

AIO_TrackingStart 通道编号,ProportionalGain [,IntegralGain [,DifferentialGain]]

参数

通道编号
以1～8的整数值指定连接所使用的距离传感器的模拟I/O的通道编号。

ProportionalGain
用于以0以外的50以下的正实数指定距离跟踪功能的比例增益。最佳值因机器人移动速度或工件形状等而异，因此，需要根据使用环境进行设置。

IntegralGain
用于以100以下的正实数指定距离跟踪功能的积分增益。可省略。如果省略，将设置为“0”。如要提高距离跟踪精度，请提高积分增益。

DifferentialGain
用于以100以下的正实数指定距离跟踪功能的微分增益。可省略。如果省略，将设置为“0”。如要提高距离跟踪精度，请提高微分增益。

说明

距离跟踪功能使用连接到模拟I/O上的距离传感器的测量值对机器人进行控制，以确保机器人与工件之间保持一定的距离。

进行控制的机器人的轴方向，是由进行AIO_TrackingSet参数“进行距离跟踪的轴”指定的1个轴的方向。如果将保持的距离设为“基准值”，执行本命令时，由距离传感器测量的距离将成为基准值。

在执行AIO_TrackingStart时启用距离跟踪功能，执行AIO_TrackingEnd时退出。在执行AIO_TrackingEnd之前，距离跟踪功能持续有效。不使用距离跟踪功能时，请立即执行AIO_TrackingEnd，退出距离跟踪功能。

如果在执行AIO_TrackingSet之前执行AIO_TrackingStart，将发生错误。请务必在执行AIO_TrackingSet之后执行AIO_TrackingStart。

可使用距离跟踪功能的机器人类型包括水平多关节型(包括RS系列)与垂直6轴型(包括N系列)。

机器人在使用距离跟踪功能期间可执行动作，但仅限于CP动作。无法执行PTP动作。

如果在执行距离跟踪功能期间经过特殊点附近，将发生错误。

在使用距离跟踪功能期间无法执行下述命令。

切换为MOTOR OFF的命令	Motor off、SFree
PTP动作命令	BGo、Go、JTran、Jump、Jump3、Jump3CP、JumpTLZ、Pass、Ptran、Pulse、TGo
力感控制执行命令	FCKeep、带FC的动作命令、FS#.Reset、FS.Reboot
转矩控制执行命令	TC
传送带跟踪执行命令	动作命令 + Cnv_QueGet
VRT执行命令	VRT, VRT_CPMotion
设置变更命令	AIO_TrackingSet, Arm, ArmSet, Base, Calib, CalPls, ECP, ECPSet, Hofs, Inertia, MCal, Power, TLSet, Tool, Weight (AIO_TrackingSet、ArmSet、ECPSet、TLSet仅在变更正在使用的编号时发生错误。)

其它	Brake、Here、Home、VCal、WaitPos

ProportionalGain、IntegralGain、DifferentialGain的设置

ProportionalGain的设置值越大，机器人跟踪速度越快。但是，如果设置值过大，可能因机器人的动作过快而导致发生错误。

IntegralGain与DifferentialGain可省略。如要提高补偿精度，需要进行设置。

如果未设置适当的值，可能导致机器人的动作过快或产生振动。

有关各增益的设置方法，请参阅下述手册。
《Epson RC+ 用户指南 - 距离跟踪功能》

⚠ 注意

如果设置过大的ProportionalGain、IntegralGain、DifferentialGain值，机器人可能做出意想不到的动作。

请阶段性地将各参数变更为较大的值。如果一下子变更为较大的值，机器人可能做出意想不到的动作，非常危险。

出现异常动作时，请立即按下紧急停止按钮。

参阅

AIO_TrackingSet、AIO_TrackingEnd、AIO_TrackingOn函数

AIO_TrackingStart使用示例

以P1为动作开始位置、经由P2且以P3为动作结束位置并使用距离跟踪功能移动机器人的程序示例。

⚠ 注意

使用示例中设置的参数为参考值。

因设置的参数或动作环境而导致作业未成功时，可能做出带有振动的动作。

另外，出现异常动作时，请立即按下紧急停止按钮。

```
Function Main
  Motor On
  Power High
  SpeedS 30
  AccelS 300300

  Go P1     ' 移动到开始位置P1
  AIO_TrackingSet 1,1,0,-5,5,0,2     ' 设置距离跟踪功能
  AIO_TrackingStart 1,1,0,0     ' 启用距离跟踪功能
  Move P2     ' 在执行距离跟踪功能的同时移动到P2
  Move P3     ' 在执行距离跟踪功能的同时移动到P3
  AIO_TrackingEnd     ' 退出距离跟踪功能
  Motor Off
Fend
```

3.5.23 AIO_TrackingEnd

用于退出距离跟踪功能。

格式
AIO_TrackingEnd

说明
用于退出通过AIO_TrackingStart启用的距离跟踪功能。

参阅
AIO_TrackingSet、AIO_TrackingStart、AIO_TrackingOn函数

AIO_TrackingEnd使用示例
以P1为动作开始位置、经由P2且以P3为动作结束位置并使用距离跟踪功能移动机器人的程序示例。

> ⚠ 注意
>
> 使用示例中设置的参数为参考值。
>
> 因设置的参数或动作环境而导致作业未成功时，可能做出带有振动的动作。
>
> 另外，出现异常动作时，请立即按下紧急停止按钮。

```
Function Main
  Integer ChNo
  Motor On
  Power High
  SpeedS 30
  AccelS 300300
  ChNo=1

  Go P1     ' 移动到开始位置P1
  AIO_TrackingSet ChNo,10,0,-3,3,0,2     ' 设置距离跟踪功能
  AIO_TrackingStart ChNo,1,0,0     ' 启用距离跟踪功能
  Move P2     ' 在执行距离跟踪功能的同时移动到P2
  Move P3     ' 在执行距离跟踪功能的同时移动到P3
  AIO_TrackingEnd     ' 退出距离跟踪功能
  Motor Off
Fend
```

3.5.24 AIO_TrackingOn函数

用于返回指定的机器人是否在执行距离跟踪功能。

格式

AIO_TrackingOn (机器人编号)

参数

机器人编号
　　利用表达式或数值指定要获取状态的机器人的编号。

返回值

执行距离跟踪功能期间，返回True (-1)；停止期间，返回False (0)。

参阅

AIO_TrackingSet、AIO_TrackingStart、AIO_TrackingEnd

AIO_TrackingOn使用示例

```
Function Main
  Integer i
  i = AIO_TrackingOn(1)
  print i
Fend
```

命令窗口的使用示例

```
> print AIO_TrackingOn(1)
0
```

3.5.25 Align函数

是用于返回已转换的点数据的函数，以在指定点上将机器人的工具坐标系统的姿势(U、V、W)对准指定的本地坐标系的坐标轴中最近的坐标轴。

格式

Align (指定点[,本地坐标系编号[, 座標軸指定]])

参数

点指定
　　指定对象点数据。
本地坐标系编号
　　指定要设为对准姿势的基准的本地坐标系编号。如果省略，将指定基础坐标系。
指定坐标轴
　　指定要对准的坐标轴。如果省略，将对准最近的坐标轴。

常数	值	
COORD_X_PLUS	1:	+X轴
COORD_Y_PLUS	2:	+Y轴
COORD_Z_PLUS	3:	+Z轴
COORD_X_MINUS	4:	-X轴
COORD_Y_MINUS	5:	-Y轴
COORD_Z_MINUS	6:	-Z轴

说明

垂直6轴型机器人(包括N系列)可能在固定由点数据定义的工具坐标系位置(原点)的状态下，仅改变姿势以对准特定的坐标系。Align函数用于进行转换，以便将指定点数据的姿势数据(U、V、W值)对准指定本地坐标系的坐标轴中的最近坐标轴或指定坐标轴。

为垂直6轴型(包括N系列)以外的机器人时，直接返回指定点。

参阅

AlignECP函数、LJM函数

Align函数使用示例

```
Move Align(P0) ROT

P1 = Align(P0, 1)
Move P1 ROT

P2 = Align(P0, 1, 3)
Move P2 ROT
```

3.5.26 AlignECP函数

是用于返回已转换的点数据的函数，以在指定点上将机器人的工具坐标系统的姿势(U、V、W)对准指定的ECP坐标系的坐标轴中最近的坐标轴。

格式

AlignECP (指定点, ECP坐标系编号)

参数

点指定
　　指定对象点数据。
ECP坐标系编号
　　指定要设为对准姿势的基准的ECP坐标系编号。

说明

垂直6轴型机器人(包括N系列)可能在固定由点数据定义的工具坐标系位置(原点)的状态下，仅改变姿势以对准特定的坐标系。AlignECP函数用于变换指定点数据的姿势数据(U、V、W值)，以对准指定的ECP坐标系的坐标轴中最近的坐标轴。

为垂直6轴型(包括N系列)以外的机器人时，直接返回指定点。

参阅

Align函数、LJM函数

AlignECP函数使用示例

```
Move AlignECP(P0) ROT

P1 = AlignECP(P0, 1)
Move P1 ROT
```

3.5.27 And 运算符

用于进行2个值的And运算(逻辑或位)。

格式

result = 值1 And 值2

参数

值1、值2
　　在逻辑And运算中，指定返回逻辑值的值。在位And运算中，指定整数表达式。
result
　　在逻辑And运算中，返回逻辑值。在位And运算中，返回整数。

说明

逻辑And运算用于组合2个以上的值并输出Bolean型的结果。下表给出And运算的模式。

值1	值2	result
True	True	True
True	False	False
False	True	False
False	False	False

位And运算用于以位为单位比较两个数值，并按照下表将相应位输出到result中。

值1的位	值2的位	result
0	0	0
0	1	0
1	0	0
1	1	1

参阅

LShift、Mask、Not、Or、RShift、Xor

And运算符使用示例

```
Function LogicalAnd(x As Integer, y As Integer)

  If x = 1 And y = 2 Then
    Print "The values are correct"
  EndIf
Fend

Function BitWiseAnd()

  If (Stat(0) And &H800000) = &H800000 Then
    Print "The enable switch is open"
  EndIf
Fend

>print 15 and 7
7
>
```

3.5.28 AOpen

用于在增补模式(追加写入)下打开文件。

格式

UOpen 文件名As #文件编号
.
.
Close #文件编号

参数

文件名
　　指定包括路径的文件名字符串。仅指定文件名时，是指当前目录中的文件。详情请参阅ChDisk。
文件编号
　　以30～63之间的整数值或表达式进行指定。

说明

以指定的文件编号打开指定的文件。此语句用于对指定文件进行增补(追加写入)。如果不存在指定文件，将创建新文件。

指定的文件编号用于在打开文件期间识别该文件。因此，在关闭该文件之前，不能使用与其他文件相同的文件编号。按文件操作命令(Print#、Write、Seek、Flush、Close)使用文件编号。

利用Close语句关闭文件并释放文件编号。

请利用FreeFile函数获取文件编号，以免在多个任务中使用同一编号。

注意

- 可使用网络路径。
- 向文件写入时进行缓冲。
- 可利用Flush语句写入被缓冲的数据。利用Close语句关闭文件时也进行写入。

参阅

Close, Print#, BOpen, ROpen, UOpen, WOpen, FreeFile, Flush

AOpen使用示例

```
Integer fileNum, i
fileNum = FreeFile
WOpen "TEST.DAT " As #fileNum
For i = 0 To 100
   Print #fileNum, i
Next I
Close #fileNum
....
....
....
FileNum = FreeFile
AOpen "TEST.DAT" As #FileNum
For i = 101 to 200
   Print #FileNum, i
Next i
Close #FileNum
```

3.5.29 Arc、Arc3

Arc用于在基础坐标系的XY平面上以曲线动作将机械臂从当前位置移至指定位置。

Arc3用于在三维平面上以曲线动作将机械臂从当前位置移至指定位置。

这2个命令也可以用于水平多关节型(包括RS系列)机器人或垂直6轴型(包括N系列)机器人。

格式

(1) Arc 经由坐标，目标坐标 [ROT] [CP] [Till | Find] [!并行处理!] [SYNC]
(2) Arc3 经由坐标，目标坐标 [ROT] [ECP] [CP] [Till | Find] [!并行处理!] [SYNC]

参数

经由坐标
　　可在点数据或XY函数中指定。是机械臂从当前位置移至目标坐标的轨道所必须通过的点。
目标坐标
　　可在点数据或XY函数中指定。是机械臂可以通过曲线动作移动的到达地点和目标位置。
ROT
　　以工具姿势变化为优先，确定动作速度、加减速度。可省略。
ECP
　　指定外部控制点动作。可省略。(仅在使用ECP选项时有效)
CP
　　指定路径运动。可省略。
Till | Find
　　记述Till或Find表达式。可省略。

```
Till | Find
Till Sw(表达式) = {On | Off}
Find Sw(表达式) = {On | Off}
```

! 并行处理 !
　　Arc语句中可以使用并行处理语句。可省略。(详情请参阅“并行处理”。)
SYNC
　　预约动作命令。在通过SyncRobots开始动作之前，机器人不进行动作。可省略

说明

Arc和Arc3用于以曲线动作并通过经由坐标将机械臂从当前位置移至目标坐标。根据给出的3点(当前位置、经由坐标、目标坐标)自动计算曲线动作轨道，并沿着该轨道移动机械臂直至目标坐标。

若使用SCARA机器人，U坐标从当前位置移动到目标坐标点。若使用6轴机器人，U,V,W坐标以最短旋转姿态，从当前位置移动到目标坐标点。通过经由坐标指定的姿势(U、V、W)不通过。

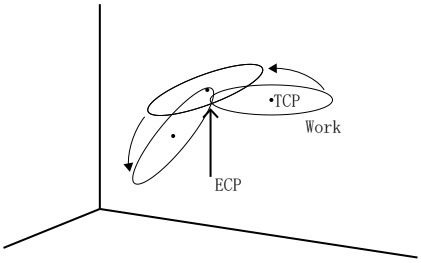
使用前，请实现确认实际动作。

Arc和Arc3的速度和加减速度分别使用SpeedS和AccelS的设置值。有关速度与加减速度之间的关系，请参阅“注意”中的“与CP同时使用Arc、Arc3”。不过，指定ROT修饰参数时的速度和加减速度分别使用SpeedR和AccelR的设置值。此时，SpeedS和AccelS的设置值变为无效状态。

通常的移动距离为0，仅姿势关节进行动作时，会发生错误。通过附加ROT修饰参数并以工具姿势变化的加速度为优先，可不出错误地进行动作。已经附加ROT修饰参数时，如果没有姿势变化，并且移动距离不是0，则会发生错误。

另外，相对于移动距离，工具姿势变化速度过大时，或指定的转速超过机械手限度时，也会发生错误。此时，请降低指定速度，或附加ROT修饰参数，并以姿势变化的加减速度优先。

使用ECP时(仅限Arc3)，在对应于指定ECP编号的外部控制点上，工件沿着圆弧轨道移动。此时，顶端关节的中心不沿着圆弧轨道移动。



Arc动作的速度和加速度

分别通过SpeedS 和AccelS 进行相对于Arc和Arc3命令的速度和加减速度的设置。通过SpeedS指定速度(单位：mm/sec)、通过AccelS指定加减速度(单位：mm/sec^2)。

注意

- Arc命令仅在水平面上有效

  按照Arc命令描画的轨迹在XY平面上将变为正圆弧。通过经由坐标指定的Z坐标或姿势(U、V、W)不通过，插补当前点和目标坐标值。
  Arc3可以在三维空间中指定圆弧轨道。通过经由坐标指定的Z坐标通过，但是姿势(U、V、W)不通过，插补当前点和目标坐标值。

- Arc命令的范围确认

  Arc和Arc3语句可以在Arc动作前进行轨道范围确认运算。因此，即使目标位置在动作区域内，如果轨道偏出区域外，可能会停止。此时可能会产生冲击，给机械臂造成障碍，所以需要预先以低速执行程序确认轨道。

- Arc动作的设置

  基于Arc命令的曲线动作是从当前位置开始，所以有时也需要在执行Arc和Arc3之前，预先使用Go和Jump及其他关联动作命令，将机器人的机械臂移至恰当的位置上。

- 与CP同时使用Arc、Arc3

  如果使用CP参数，动作命令则会在开始减速的同时将控制移交给下一语句。这在用户连续发出几个动作命令，要以一定的速度进行连续动作时非常便利。为未指定CP的Arc命令、Arc3命令时，机械臂必须减速，以停在指定的目标位置上。

常见错误

- 变更夹具末端(手腕)的属性

  所以使用Arc命令时，请注意各点的夹具末端的属性。如果在以后插补动作期间变更夹具末端的方向(例如从右手腕向左手腕、或者相反的变更等)，将会出错。机械臂的属性值(/L左腕、/R右腕)必须与实际的当前位置、经由坐标和目标坐标一致。

- 想要将机械臂移至移动范围外时

  如果指定的曲线动作要将机械臂移至移动范围外，将会出错。

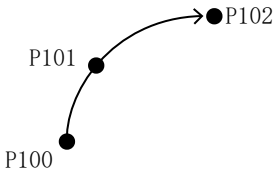
参阅

!并行处理!、AccelS、Move、SpeedS

Arc、Arc3使用示例

下述为描画如图那样的轨迹的程序例。实现从P100开始动作、经由P101到达P102的曲线动作轨迹。

```
Function ArcTest
   Go P100
   Arc P101, P102
Fend
```



提示

初次使用Arc命令时，建议使用移动范围中的机器人一侧的点，以简单的圆弧试着描画。此时，请设想一下实际描画的圆弧轨迹。请不要示教使机械臂移至正常移动范围外的点。

3.5.30 Arch

用于设置和显示Jump、Jump3、Jump3CP命令的Arch参数。

格式

(1) Arch Arch编号, 转移距离, 接近距离

(2) Arch Arch编号

(3) Arch

参数

Arch编号

以0～6的整数指定Arch编号。有效值是0～6的整数，如下页的Arch表格那样，有效值总共有7个。

转移距离

用于通过Jump命令指定水平动作前的转移距离(从出发点算起的垂直距离)。(单位：mm)通过Jump3、Jump3CP命令指定跨越动作前的转移距离。(单位：mm)

接近距离

用于通过Jump命令指定完全结束水平移动阶段的接近距离(从目的点算起的垂直距离)。(单位：mm)

mm)通过Jump3、Jump3CP命令指定完全结束跨越动作的阶段的接近距离。(单位：mm)

结果

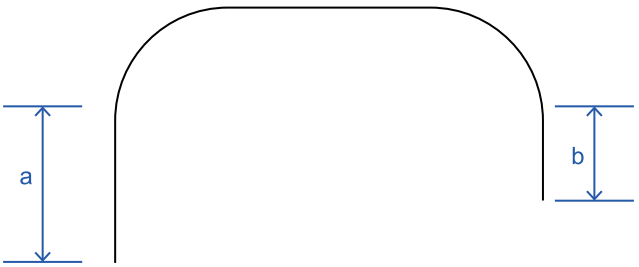
如果省略所有参数，将显示Arch表格的所有内容。

如果只指定Arch编号，将显示指定Arch编号的Arch表格的内容。

说明

通过Arch命令定义Jump动作命令所需的Arch表格的值。Arch动作将用作为Jump的修饰词的、与Arch编号对应的参数执行。(为理解Arch命令，首先请仔细阅读Jump语句。)

按照Arch的设置，如果使用Jump C [Arch编号]，可以使Z方向的角变得圆滑。(参照使用示例)在Arch表格中，设置开始水平方向移动之前的垂直方向的移动距离(转移距离)和结束水平方向移动之后的到目标坐标的垂直方向的移动距离(接近距离)。(请参照下图。)



符号	说明
a	转移距离
b	接近距离

用户定义的Arch表格的值是0～6的整数，共有7个。第8个设置值(Arch 7)是默认值，实际上是设置了门控运动(参照下图)，而不是Arch运动。如果使用默认Arch值(第8个设置值)执行Jump命令，机械臂将进行下图所示的动作。

1. 首先，只有第3关节移至LimZ命令设置的Z坐标值(最大Z值)位置。
2. 然后，机械臂水平移动到目标坐标位置，并最终到达X、Y、U位置。
3. 最后，只有第3关节动作，使机械臂降低至最终的第3关节坐标位置(Z坐标值)，并结束Jump命令动作。

门控运动 (Arch 7的jump动作)

Arch表格默认值

Arch编号	转移距离	接近距离
0	30	30
1	40	40
2	50	50
3	60	60
4	70	70
5	80	80
6	90	90

注意

- 形成门控运动的其他情况

  如果在垂直上升距离和垂直下降距离中设置了大于实际垂直移动距离，将变为门控运动而非Arch运动。

- Arch值将被保存

  Arch表格的值只要用户未变更，都将被保存。

使用Arch时的重要事项

由于Arch运动是通过轨迹控制所进行的动作合成，因此，不能保证实际的轨迹。其轨迹会因动作速度或机械臂的移动方式而异。请通过作业使用的实际速度和姿势确认实际轨迹。

- 即使在相同位置上执行带有相同C [Arch编号] 的Jump命令(或者Jump3命令)，低速时的轨迹也会低于高速动作时的轨迹。因此，即使确认没有高速碰撞到障碍物，但低速动作时也可能会发生碰撞，敬请注意。
- 与低速动作时相比，高速动作时没有合成的转移移动量会增大，而没有合成的接近移动量则会减小。没有达到期待的移动距离时，请降低速度或减速度，或将接近距离设置得长一些。
- 即使是相同距离的动作，轨迹也会因机械臂的移动方式而异。虽然因机械臂的移动方式而导致的轨迹变化多种多样，但是，如果以一般的水平过关节型机器人为例，第1机械臂的移动幅度越大，垂直上升量也越大，而垂直下降量则越小。没有达到期待的垂直下降距离时，请降低速度或减速度，或将下降距离设置得长一些。

参阅

Jump、Jump3、Jump3CP

Arch使用示例

下述为从命令窗口实施的Arch值的设置例。

```
> arch 0, 15, 15
> arch 1, 25, 50
> jump p1 c1
> arch
  arch0 =    15.000        15.000
  arch1 =    25.000        50.000
  arch2 =    50.000        50.000
  arch3 =    60.000        60.000
```

```
  arch4 =    70.000        70.000
  arch5 =    80.000        80.000
  arch6 =    90.000        90.000
>
```

3.5.31 Arch函数

用于返回Arch的设置状态。

格式

Arch (Arch编号, 参数编号)

参数

Arch编号
　　指定0～6的整数。
参数编号

- 1：转移距离
- 2：接近距离

返回值

用于返回以参数编号指定的距离。

参阅

Arch

Arch函数使用示例

```
Double archValues(6, 1)
Integer i

'保存当前Arch的设置状态
For     i = 0 to 6
    archValues(i, 0) = Arch(i, 1)
    archValues(i, 1) = Arch(i, 2)
Next i
```

3.5.32 AreaCorrection函数

返回使用校正区域校正过的点的函数

格式
AreaCorrection (点指定, 区域编号)

参数

点指定
　　指定要补偿的点数据。
区域编号
　　以表达式或数值指定区域编号(1～8的整数)。

说明
基于预先定义的校正区域，返回已校正的结果的点。坐标通过与校正前的点相同的本地坐标系定义。与动作指令(Go或Jump命令等)一起使用本函数，可将机器人移动至指定位置。通过使用本命令，可提高指定点的位置精度。在点指定中，请输入图纸上的位置。

校正仅对位置适用。校正不适用于附加轴、UVW坐标值、姿势标志。输入的点数据值是按原样输出的。

如果指定未设置的校正区域，将导致错误。

注意

- 已完成示教的点

  请勿对已完成示教的点数据适用AreaCorrection函数。已对准的示教位置被校正，位置发生偏移。

- 远离校正区域时

  如果远离通过AreaCorrectionSet设置的校正区域，校正效果将降低。设置基准点时,请确保校正区域围绕着动作点。

  如果选择了平面作为校正类型，对在垂直方向上与选为校正区域的平面存在距离的点，校正的效果将降低。请在适当的高度上设置校正区域，或者，如果可以设置基准点，则在校正类型中指定空间。

- 与已设置校正区域的姿势标志不同时

  如果通过AreaCorrectionSet设置的基准点与姿势标志不同，将发生错误。

- 与已设置校正区域的姿势(U, V, W)不同时

  使用SCARA机器人(包括RS系列)时，可进行校正。

  使用垂直6轴机器人(包括N系列)时，如果校正前的点中的工具坐标系Z轴与校正区域的基准点的工具坐标系Z轴相一致，可以进行校正。如果不一致，将无法适用校正，并发生错误。通过将DiffToolOrientation函数的轴编号指定为COORD_Z_PLUS，可获取工具坐标系Z轴的角度。

参阅
AreaCorrectionSet、AreaCorrectionClr、AreaCorrectionDef函数、AreaCorrectionInv、AreaCorrectionOffset函数、DiffToolOrientation函数

AreaCorrection使用示例

```
Function sample
    ' P(1:4) 基准点
```

```
    P1 = XY(-100, 200, -20, 0)
    P2 = XY(100, 200, -20, 0)
    P3 = XY(-100, 400, -20, 0)
    P4 = XY(100, 400, -20, 0)
   ' P(11:14) 实际上使用对P(1:4)进行示教的点
   P11 = XY(-100, 200.5, -20, 0)
    P12 = XY(100.3, 200.1, -20, 0)
    P13 = XY(-100.4, 400.8, -20, 0)
    P14 = XY(100.2, 400.4, -20, 0)
    ' 设置校正区域
   AreaCorrectionSet 1, P(1:4), P(11:14), MODE_PLANE
   P999 = AreaCorrection(P1, 1)  ' P999为已完成校正的点
   Print Dist(P11, P999)
   P999 = AreaCorrection(XY(0, 300, -20, 0), 1) ' 校正区域内的点
   Print P999
Fend
```

[输出结果]

```
0
 X:    0.100 Y:  300.450 Z:  -20.000 U:    0.000 /R /0
```

3.5.33 AreaCorrectionClr

清除校正区域。

格式

AreaCorrectionClr 区域编号

参数

区域编号
以表达式或数值指定区域编号(1～8的整数)。

结果

清除与区域编号相对应的校正区域。

在机器人运行期间无法执行本命令。请在停止状态下使用。

参阅

AreaCorrectionSet、AreaCorrectionDef函数、AreaCorrectionInv、AreaCorrectionOffset函数

AreaCorrectionClr使用示例

```
AreaCorrectionClr 1
```

3.5.34 AreaCorrectionDef函数

返回是否已设置指定的校正区域。

格式

AreaCorrectionDef(区域编号)

参数

区域编号
以表达式或数值指定区域编号(1～8的整数)。

返回值

如果已设置校正区域，将返回“True”，否则将返回“False”。

参阅

AreaCorrectionSet、AreaCorrectionClr、AreaCorrectionInv、AreaCorrectionOffset函数

AreaCorrectionDef函数使用示例

```
Function DisplayAreaCorrectionDef(areaNum As Integer)

   If AreaCorrectionDef(areaNum) = False Then
      Print "Area", areaNum, " is not defined"
   Else
      Print "Area Definition:"
      AreaCorrectionSet areaNum
   EndIf
Fend
```

3.5.35 AreaCorrectionInv函数

将已完成校正的点恢复原状的函数

格式
AreaCorrectionInv(点指定, 区域编号)

参数

点指定
　　指定要补偿的点数据。
区域编号
　　以表达式或数值指定区域编号(1～8的整数)。

说明
对已通过AreaCorrection函数校正的点，返回校正前的点数据。

如果对实际进行示教并创建的点或已完成校正的点适用AreaCorrectionInv，将获得校正前的点数据。

如果指定未设置的校正区域，将导致错误。

参阅
AreaCorrectionSet、AreaCorrectionClr、AreaCorrectionDef函数、AreaCorrectionOffset函数

AreaCorrectionInv使用示例

```
Function AreaCorrectionTest
    '  P(1:4) 基准点
    P1 = XY(-100, 200, -20, 0)
    P2 = XY(100, 200, -20, 0)
    P3 = XY(-100, 400, -20, 0)
    P4 = XY(100, 400, -20, 0)
    '  P(11:14) 实际上使用对P(1:4)进行示教的点
    P11 = XY(-100, 200.5, -20, 0)
    P12 = XY(100.3, 200.1, -20, 0)
    P13 = XY(-100.4, 400.8, -20, 0)
    P14 = XY(100.2, 400.4, -20, 0)
    '  设置校正区域
    AreaCorrectionSet 1, P(1:4), P(11:14), MODE_PLANE
    P888 = AreaCorrection(P1, 1) ' P888为已完成校正的点
    P999 = AreaCorrectionInv(P888, 1) ' P999为转换前的点
    Print Dist(P11, P888)
    Print Dist(P1, P999)
Fend
```

[输出结果]

```
0
0
```

3.5.36 AreaCorrectionOffset函数

返回相对于已校正的点移动的点的函数

格式

AreaCorrectionOffset(指定点, 指定相对移动量, 区域编号[, 选择相对关系])

参数

点指定
　　指定作为相对移动参考位置的点数据。
指定相对移动量
　　通过点数据指定相对移动量。
区域编号
　　以表达式或数值指定区域编号(1～8的整数)。
选择相对关系
　　表示以哪个坐标系为参考进行相对移动。如果省略，将参考本地坐标系。
　　以本地坐标系为参考移动时，将参考以已定义点指定的坐标系作为参考进行相对移动的坐标。以工具坐标系为参考移动时，将参考以点指定位置为参考进行相对移动的坐标。

选择相对关系	常数	值
本地坐标参考	AC_LOCAL	0
工具坐标参考	AC_TOOL	1

说明

对已通过AreaCorrection函数校正的点，返回相对移动的点。将参考通过与指定点相同的本地坐标系定义的坐标。与动作指令(Go或Jump命令等)同时使用本函数，即可将机器人移动至指定位置。

与Here函数组合使用，可执行与BGo、TGo相同的动作。如果是在校正区域内，相对移动量将较为准确。

如果指定了未设置的校正区域，将导致错误。

注意

- 已进行姿势的相对移动时

  已进行姿势的相对移动时，相对移动后的姿势无法适用校正，并可能发生错误。

参阅

AreaCorrectionSet、AreaCorrectionClr、AreaCorrectionDef函数、AreaCorrectionInv、Here

AreaCorrectionOffset使用示例

```
' 校正区域1已完成定义
' 与BGo XY(50, 0, 0, 0)相同
Go AreaCorrectionOffset(Here, XY(50, 0, 0, 0), 1)
' 与TGo XY(50, 0, 0, 0)相同
Go AreaCorrectionOffset(Here, XY(50, 0, 0, 0), 1, AC_TOOL)
```

3.5.37 AreaCorrectionSet

设置并显示校正区域。

格式

(1) AreaCorrectionSet 区域编号, 参考连续点, 示教连续点, 校正类型

(2) AreaCorrectionSet 区域编号

(3) AreaCorrectionSet

参数

区域编号

以表达式或数值指定区域编号(1～8的整数)。

参考连续点

用冒号连接起点和终点的2个点编号，像P(1:4)一样指定作为基准点的点数据的连号。
为了将校正的效果最大化，选择基准点时，请确保围绕着要校正的点。

示教连续点

用冒号连接起点和终点的2个点编号，像P(1:4)一样指定相对于参考连续点示教的点数据。请对应参考连续点设置点数据的排列顺序。

补偿的类型

表示补偿类型的整数值。
平面校正可以对由选为基准点的点组成的平面上的点进行校正。如果选择平面校正，请将参考连续点放置在平面上。至少需要3个基准点。
空间校正可以对由选为基准点的点组成的立体空间上的点进行校正。如果选择立体校正，请确保参考连续点围绕想要校正的区域。至少需要4个基准点。

校正的类型	常数	值
平面	MODE_PLANE	2
空间	MODE_SPACE	3

结果

- 如果以(1)的格式指定，校正区域将被设置在指定的区域编号。
- 如果以(2)的格式指定，将显示指定的区域编号的内容。
- 如果以(3)的格式指定，将显示所有已定义的校正区域的内容。

说明

设置用于区域校正功能的校正区域。设置校正区域并使用AreaCorrection函数、AreaCorrectionInv函数、AreaCorrectionOffset函数，即可提高校正区域内的点的位置精度。

在运行过程中无法执行本命令。请在停止状态下使用。

有关参考位置的选择方法，请参阅以下手册。
《Epson RC+ 用户指南 - 区域校正功能》

注意

- 校正区域数据

  校正区域在控制器关闭之前一直有效。控制器启动时，处于未定义校正区域的状态。

- 工具

  使用AreaCorrection函数进行校正时，请使用与对校正区域基准点进行示教时使用的工具相同的工具。使用不同的工具时，校正效果可能降低。

- 远离校正区域时

如果远离通过AreaCorrectionSet设置的校正区域，校正效果将降低。设置基准点时,请确保校正区域围绕着动作点。

如果选择了平面作为校正类型，对在垂直方向上与选为校正区域的平面存在距离的点，校正的效果将降低。请在适当的高度上设置校正区域，或者，如果可以在高度方向上设置基准点，则在校正类型中指定空间。

- 与已设置校正区域的姿势标志不同时

  如果通过AreaCorrectionSet设置的基准点与姿势标志不同，将发生错误。请将与执行动作的点相同的姿势标志设置为基准点。

- 与已设置校正区域的姿势(U, V, W)不同时

  使用SCARA机器人(包括RS系列)时，可进行校正。

  使用垂直6轴机器人(包括N系列)时，如果校正前的点中的工具坐标系Z轴与校正区域的基准点的工具坐标系Z轴相一致，可以进行校正。如果不一致，将无法适用校正，并发生错误。通过将DiffToolOrientation函数的轴编号指定为COORD_Z_PLUS，可获取工具坐标系Z轴的角度。

参阅

AreaCorrectionClr、AreaCorrectionDef函数、AreaCorrectionInv、AreaCorrectionOffset函数、DiffToolOrientation函数

AreaCorrectionSet使用示例

使用示例如下。请对P11～P14使用已示教的点。

如下所示，如果将P1～P4作为基准点在图纸上的位置，校正区域将是以P1、P2、P3、P4为顶点的宽200mm的正方形。

```
Function AreaCorrectionTest
  ' P(1:4) 基准点
  P1 = XY(-100, 200, -20, 0)
  P2 = XY(100, 200, -20, 0)
  P3 = XY(-100, 400, -20, 0)
  P4 = XY(100, 400, -20, 0)
  ' P(11:14) 实际上使用对P(1:4)进行示教的点
  P11 = XY(-100, 200.5, -20, 0)
  P12 = XY(100.3, 200.1, -20, 0)
  P13 = XY(-100.4, 400.8, -20, 0)
  P14 = XY(100.2, 400.4, -20, 0)
  ' 设置校正区域
  AreaCorrectionSet 1, P(1:4), P(11:14), MODE_PLANE
Fend
```

3.5.38 Arm

如果选择机械臂，将显示当前选择的机械臂编号。

格式

(1) Arm 机械臂编号

(2) Arm

参数

机械臂编号

以整数值或表达式进行指定。有效值范围是0～15，最多可以选择16个不同的机械臂。机械臂0是标准(默认设置)的机器人机械臂。机械臂1～15是由ArmSet定义的增设机械臂。可以省略，在省略时，将显示当前的机械臂编号。

结果

如果未设置参数而执行Arm命令，将显示当前选择的机械臂编号。

说明

用于指定用哪个机械臂执行机器人命令。可以利用Arm，在增设的机械臂上共享位置数据。如果未设置增设的机械臂，标准机械臂(机械臂编号0)将会动作。发货时已将机械臂编号设为“0”，所以没有增设机械臂时不需要变更设置。但是，在使用增设机械臂时，请通过ArmSet，设置为最初的机械臂。

即使实际的机器人构成与标准构成有差异时，也可以设置适于各机器人的恰当的增设机械臂参数。增设机械臂在下述条件下正确动作。

- 设置一个点数据，使得可以通过2个以上的机械臂。
- 使用Pallet。
- 使用CP动作。
- 进行相对位置指定。
- 使用本地坐标。

如果是使用旋转关节的水平多关节型机器人(包括RS系列)，将以按照ArmSet参数设置的值为基准计算关节角度。因此，在需要进行增设机械臂和夹具末端的设置的情况下，请使用此命令。

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

注意

- Arm 0

  无法使用ArmSet进行设置和变更。Arm 0可以用于标准的机器人构成的设置。如果将Arm值设为“0”，将可以使用机器人机械臂的标准参数。

- 使用机械臂长度校正选件时

  如果将ArmCalib设为On，则会将机械臂长度补偿值适用于Arm 0，并自动切换为Arm 0。要进行适用机械臂长度补偿值的动作时，请使用Arm 0。即使将ArmCalib设为On，但如果不是Arm 0，也不会成为适用机械臂长度补偿值的动作。

  要使用机器人机械臂的标准参数时，请将ArmCalib设为Off，并使用Arm 0。

- 未设置的机械臂编号

  如果选择未通过ArmSet定义的增设机械臂的编号，将会出错。

参阅

ArmClr、ArmSet、ECPSet、TLSet, ArmCalibSet

Arm使用示例

在下述程序例中，使用ArmSet和Arm进行增设机械臂的设置。通过ArmSet定义增设机械臂，并通过Arm设置选择哪个机械臂作为当前机械臂。(Arm 0是默认的机器人机械臂，用户无法变更。)

利用命令窗口的操作示例

```
> ArmSet 1, 300, -12, -30, 300, 0
> ArmSet
  arm0 250  0  0  300  0
  arm1 300  -12  -30  300  0

> Arm 0
> Jump P1    '通过设置标准机械臂，向P1 Jump
> Arm 1
> Jump P1    '通过设置增设机械臂1，向P1 Jump
```

3.5.39 Arm函数

用于返回当前选择的机器人的机械臂编号。

格式

Arm

返回值

用于返回当前的机械臂编号。

参阅

Arm

Arm函数使用示例

```
Print "The current arm number is: ", Arm
```

3.5.40 ArmCalib

用于将当前选择的机器人的机械臂长度补偿设为有效或无效。

格式

ArmCalib On | Off

参数

On | Off
要将机械臂长度补偿设为有效时，指定On；要设为无效时，指定Off。

说明

ArmCalib On命令用于将机械臂长度补偿设为有效状态。如果执行ArmCalib On，则会在Arm 0中设置由ArmCalibSet指定的补偿值，并且Arm会切换为“0”。

ArmCalib Off命令用于将机械臂长度补偿设为无效状态。如果执行ArmCalib Off，则会在Arm 0中设置标准参数。执行ArmCalib Off时，Arm不自动进行切换。

已购买机械臂长度补偿选项的情况下，出厂时机械臂长度补偿已设为有效。

在机械臂长度补偿有效的状态下将Arm设为“0”以外时，会按已设置的Arm编号设置进行动作。

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

注意

- 此命令只在已安装机械臂长度补偿选项时才可使用。
- 请勿将处于ArmCalib On状态的备份文件恢复到机械臂长度补偿选项无效的控制器中。

  如果将处于ArmCalib On状态的备份文件恢复到机械臂长度补偿选项无效的控制器中，则自动进入ArmCalib Off状态。要恢复到其它控制器中时，请加以注意。

参阅

ArmCalibClr, ArmCalibSet, ArmCalibDef

ArmCalib使用示例

下例所示为通过命令窗口执行的情况。

```
> ArmCalib On

> ArmCalib Off
```

3.5.41 ArmCalib函数

用于返回当前选择的机器人的机械臂长度补偿状态。

格式

ArmCalib

返回值

- 0 = 机械臂长度补偿无效
- 1 = 机械臂长度补偿有效

参阅

ArmCalib

ArmCalib函数使用示例

```
If ArmCalib = Off then
  Print "Arm length calibration disabled."
Else
  Print "Arm length calibration enabled."
Endif
```

3.5.42 ArmCalibClr

用于清除机械臂长度补偿设置。

格式
ArmCalibClr

说明
机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

注意

- 除非必要，否则切勿使用ArmCalibClr。

  ArmCalibClr用于清除由ArmCalibSet设置的机械臂长度补偿参数。出厂时已对ArmCalibSet进行了精密设置。(购买机械臂长度补偿选项时)如果错误地清除设置值，则需要重新在工厂进行精密测量。除非必要，否则切勿使用ArmCalibClr。

参阅
ArmCalib, ArmCalibSet

ArmCalibClr使用示例

```
ArmCalibClr
```

3.5.43 ArmCalibDef

用于返回机械臂长度补偿的设置状态。

格式
ArmCalibDef

返回值
如果已设置机械臂长度补偿，则返回“True”；如果未设置，则返回“False”。

参阅
ArmCalib, ArmCalibClr, ArmCalibSet

ArmCalibDef使用示例

```
Function DisplayArmCalibDef
  Integer i

   If ArmCalibDef = False Then
      Print "ArmCalib is not defined"
   Else
      Print "ArmCalib Definition:"
      For i = 1 to 3
        Print ArmCalibSet(i)
      Next i
   EndIf
Fend
```

3.5.44 ArmCalibSet

用于进行机械臂长度与关节偏移的设置与显示。

格式

(1) ArmCalibSet 设置值1, 设置值2, 设置值3

(2) ArmCalibSet

参数

设置值	水平多关节型
1	第1关节～第2关节之间的水平距离 (mm)
2	第2关节～姿势中心之间的水平距离 (mm)
3	第2关节的偏移角度 (°)

结果

省略所有参数并执行ArmCalibSet时，会显示机械臂长度补偿编号的参数。

说明

ArmCalibSet用于设置与机械臂长度与关节偏移有关的参数。出厂时，可对各个体的机械臂长度与关节偏移进行精密测量，并利用本命令进行设置，以提高距离精度。

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

注意

- 除非必要，否则切勿变更ArmCalibSet。

  出厂时已对ArmCalibSet进行了精密设置。(购买Arm Length Calibration选项时)如果错误地变更设置值，距离精度或轨迹精度则会降低。除非必要，否则切勿变更ArmCalibSet。

参阅

ArmCalib, ArmCalibClr, ArmCalibDef

ArmCalibSet使用示例

下例所示为通过命令窗口执行的情况。

```
> ArmCalibSet 299.989, 250.001, 0.012
```

3.5.45 ArmCalibSet函数

用于返回1个机械臂长度补偿的设置值。

格式

ArmCalibSet(设置值编号)

参数

设置值编号

以表达式或数值指定要参阅的设置值编号(0～3的整数)。(请参照下文。)

水平多关节型机器人

设置值编号	返回值
1	第1关节～第2关节之间的水平距离 (mm)
2	第2关节～姿势中心之间的水平距离 (mm)
3	第2关节的偏移角度 (°)

返回值

以实值返回指定上表某一项的参数设置值。

参阅

ArmCalibClr, ArmCalibSet

ArmCalibSet函数使用示例

```
Double L1, L2, Angle

L1 = ArmCalibSet(1)
L2 = ArmCalibSet(2)
Angle = ArmCalibSet(3)
```

3.5.46 ArmClr

用于清除机械臂的设置。

格式

ArmClr 机械臂编号

参数

机械臂编号
以整数和表达式指定15个机械臂中的要清除设置的机械臂编号。
(机械臂0为默认机械臂，不能清除。)

说明

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

参阅

Arm、ArmSet、ECPSet、Tool、Local、LocalClr、TLSet

ArmClr使用示例

```
ArmClr 1
```

3.5.47 ArmDef函数

用于返回机械臂的设置状态。

格式

ArmDef (机械臂编号)

参数

机械臂编号
以整数值指定返回状态的机械臂编号。

返回值

如果设置指定的机械臂，则返回“True”；如果未设置，则返回“False”。

参阅

Arm、ArmClr、ArmSet、ECPSet、Local、LocalClr、Tool、TLClr、TLSet

ArmDef使用示例

```
Function DisplayArmDef(armNum As Integer)

   Integer i

   If ArmDef(armNum) = False Then
      Print "Arm ", ArmNum, "is not defined"
   Else
      Print "Arm ", armNum, " Definition:"
      For i = 1 to 5
        Print ArmSet(armNum, i)
      Next i
   EndIf
Fend
```

3.5.48 ArmSet

用于设置和显示增设机械臂。

格式

(1) ArmSet 机械臂编号, 设置值1, 设置值2, 设置值3 [, 设置值4] [, 设置值5]: 水平多关节型 (包括RS系列)
(2) ArmSet 机械臂编号, 设置值1, 设置值2, 设置值3, 设置值4, 设置值5, 设置值6, 设置值7, 设置值8, 设置值9, 设置值10, 设置值11, 设置值12: 垂直多关节型 (包括N系列)
(3) ArmSet 机械臂编号
(4) ArmSet

参数

机械臂编号
以表达式或数值指定1～15的整数。增设机械臂可以设置至15个。

设置值编号	水平多关节型 (包括RS系列)	垂直多关节型 (包括N系列)
1	从第2关节到姿势中心的水平距离(mm)	从底座到第2关节的垂直距离(mm)
2	第2关节的偏移角度(度)	从第1关节到第2关节的水平距离(mm)
3	作业前端的高度方向的偏移(mm)	从第2关节到第3关节的距离(mm)
4	从第1关节到第2关节的水平距离(mm)	从第3关节到第5关节的垂直距离(mm)
5	第4关节角度的偏移(度)	从第3关节到第5关节的水平距离(mm)
6	-	从第5关节到姿势中心的距离(mm)
7	-	第1关节角度的偏移量(度)
8	-	第2关节角度的偏移量(度)
9	-	第3关节角度的偏移量(度)
10	-	第4关节角度的偏移量(度)
11	-	第5关节角度的偏移量(度)
12	-	第6关节角度的偏移量(度)

结果

如果省略所有参数执行ArmSet，将显示设置的所有增设机械臂的编号和参数。

仅指定机械臂编号时，将显示指定的机械臂编号和参数。

说明

在标准设置中增设机械臂时，要设置与该增设机械臂相关的参数。在机器人上安装了增设机械臂和增设夹具末端时，将会很方便。在使用增设机械臂时，通过Arm命令指定要作业的机械臂。

可省略设置值4的第1关节至第2关节的水平距离和设置值5的姿势关节角度的偏移。如果省略，将进行与标准机械臂相同的设置。

如要在非标准构成下使用机器人，可以设置与其相应的参数，这是增设机械臂的设置功能。例如，在第2机器人上安装第2旋转关节时，必须进行新构成的机器人机械臂的设置。实施该设置，增设机械臂可以进行适当的动作，在这种情况下，必须满足下述条件。

- 在多个机械臂上使用同一数据点。
- 使用Pallet。
- 使用CP动作。

- 进行相对位置指定。
- 使用本地坐标系。

具有旋转关节的水平多关节型(包括RS系列)机器人，将根据ArmSet的参数计算关节角度，在需要进行增设机械臂和夹具末端的设置时，此设置就变得非常重要。

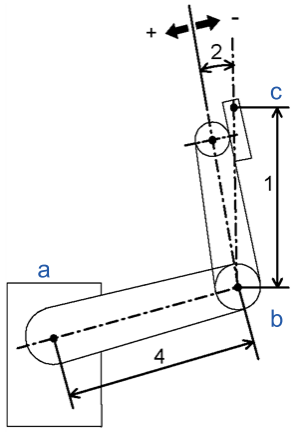
机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

注意

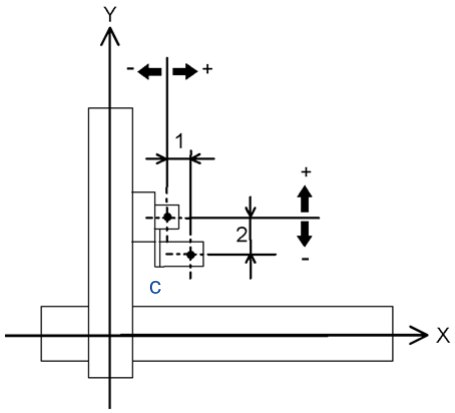
- Arm 0

  机械臂编号0用于机器人的标准构成时的设置，用户无法进行定义或变更。如果设置机械臂编号0，将使用机器人机械臂的标准参数。

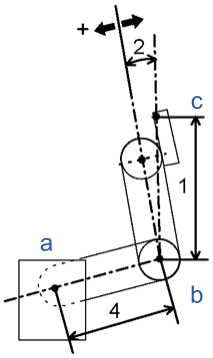
- 水平多关节型机器人



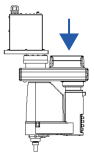
- 直角坐标型机器人



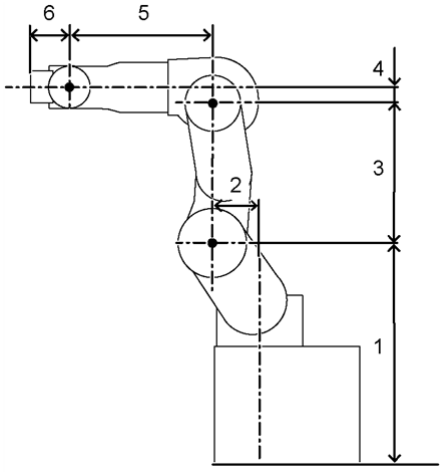
- 水平多关节型机器人(RS系列)

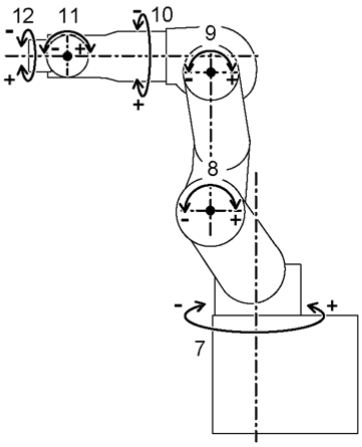


RS系列为从该方向的图解。



- 垂直多关节机器人





符号	说明
a	第1关节
b	第2关节
c	增设机械臂

参阅

Arm、ArmClr

ArmSet使用示例

下述为使用ArmSet和Arm的增设机械臂的设置示例。通过ArmSet设置增设机械臂，并通过Arm设置选择哪个机械臂。(机械臂编号0是机器人的默认设置机械臂，用户无法进行变更。)

利用命令窗口的操作示例

```
> ArmSet 1, 300, -12, -30, 300, 0
> ArmSet
  Arm 0: 125.000, 0.000, 0.000, 225.000, 0.000
  Arm 1: 300.000, -12.000, -30.000, 300.000, 0.000

> Arm 0
> Jump P1     '通过设置标准机械臂，向P1 Jump
```

```
> Arm 1
> Jump P1     '通过设置增设机械臂1，向P1 Jump
```

3.5.49 ArmSet函数

返回一个增设机械臂的设置值。

格式

ArmSet (机械臂编号, 设置值编号)

参数

机械臂编号

以表达式或数值指定要参照的机械臂编号。

设置值编号

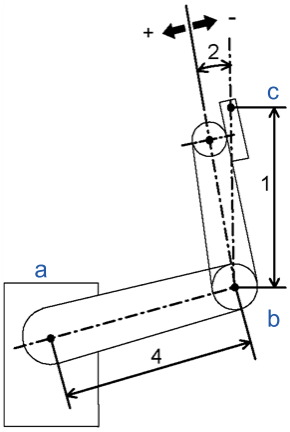
以表达式或数值指定要参阅的设置值编号(0～5的整数)。(请参阅下述内容。)

设置值编号	水平多关节型 (包括RS系列)	垂直多关节型 (包括N系列)
1	从第2关节到姿势中心的水平距离(mm)	从底座到第2关节的垂直距离(mm)
2	第2关节的偏移角度(度)	从第1关节到第2关节的水平距离(mm)
3	作业前端的高度方向的偏移(mm)	从第2关节到第3关节的距离(mm)
4	从第1关节到第2关节的水平距离(mm)	从第3关节到第5关节的垂直距离(mm)
5	第4关节角度的偏移(度)	从第3关节到第5关节的水平距离(mm)
6	-	从第5关节到姿势中心的距离(mm)
7	-	第1关节角度的偏移量(度)
8	-	第2关节角度的偏移量(度)
9	-	第3关节角度的偏移量(度)
10	-	第4关节角度的偏移量(度)
11	-	第5关节角度的偏移量(度)
12	-	第6关节角度的偏移量(度)

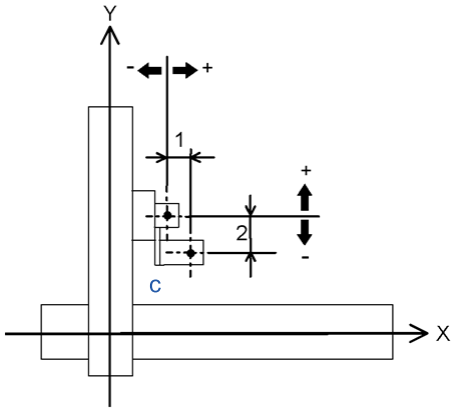
返回值

以实值返回指定了上表任一参数的设置值。

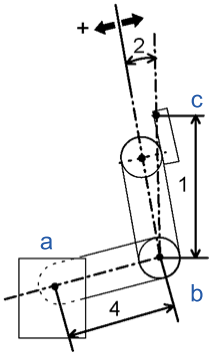
- 水平多关节型机器人



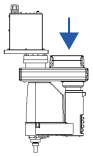
- 直角坐标型机器人



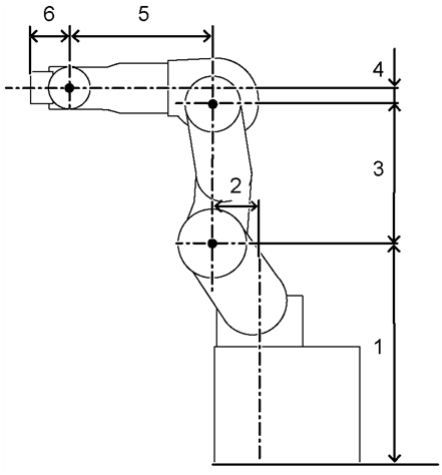
- 水平多关节型机器人(RS系列)

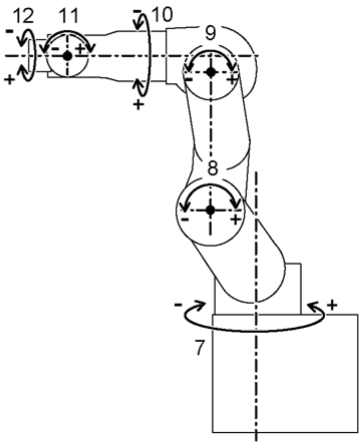


RS系列为从该方向的图解。



- 垂直多关节机器人





符号	说明
a	第1关节
b	第2关节
c	增设机械臂

参阅

ArmClr、ArmSet

ArmSet函数使用示例

```
Real x

x = ArmSet(1, 1)
```

3.5.50 Asc函数

用于返回相对于字符串开头第1字符的ASCII代码。(以十进制数返回字符码。)

格式
Asc (替换字符串)

参数

转换字符串
以字符串表达式或直接以字符串指定1个字符以上的字符串。

返回值
用于以整数值返回发送到Asc函数中的字符串的第1个字符。

说明
Asc函数用于以十进制数返回字符串开头第1个字符的字符码。发送到Asc函数中的字符串，可以是常数也可以是变量。

注意

- 仅第1个字符用于返回ASCII值

  在Asc函数中可以使用1个字符以上长度的字符串，但是实际上仅使用它的第1个字符。Asc仅用于返回字符串第1个字符的ASCII值。

参阅
Chr$, InStr, Left$, Len, Mid$, Right$, Space$, Str$, Val

Asc函数使用示例
在该例中，通过从程序中和命令窗口进行操作，来使用如下的Asc函数。

```
Function asctest
    Integer a, b, c
    a = Asc("a")
    b = Asc("b")
    c = Asc("c")
    Print "The ASCII value of a is ", a
    Print "The ASCII value of b is ", b
    Print "The ASCII value of c is ", c
Fend
```

利用命令窗口的操作示例

```
>print asc("a")
97
>print asc("b")
98
>
```

3.5.51 Asin函数

用于返回指定数值的反正弦。

格式
Asin (数值)

参数

数值
以实值指定角度的正弦。

返回值
用于以实值(单位：弧度)返回指定数值的反正弦。

说明
Asin用于返回指定数值的反正弦。指定的数值在-1～1的范围内。返回值的范围是-π/2～π/2。如果数值小于-1或大于1，将会出错。

如要将弧度值转换为“度”，需要使用RadToDeg函数。

参阅
Abs、Acos、Atan、Atan2、Cos、DegToRad、RadToDeg、Sgn、Sin、Tan、Val

Asin函数使用示例

```
Function asintest
  Double x

  x = Sin(DegToRad(45))
  Print "Asin of ", x, " is ", Asin(x)
Fend
```

3.5.52 AtHome函数

用于返回当前的机器人姿势是否在Home位置。

格式

AtHome

返回值

如果当前机器人姿势在Home位置时，返回“True”，在Home位置以外，返回“False”。

说明

本函数用于返回当前的机器人姿势是否在Home位置。注册Home位置时，使用HomeSet命令或机器人管理器。要移至Home位置，可使用Home命令。

参阅

Home、HomeClr、HomeDef、HomeSet、Hordr、MCalComplete

3.5.53 Atan函数

用于返回指定数值的反正切。

格式
Atan (数值)

参数

数值
以实值指定角度的正切。

返回值
用于以实值(单位：弧度)返回指定数值的反正切。

说明
Atan用于返回指定数值的反正切。指定的数值可以是任意数值。返回值的范围是-π～π弧度。

如要将弧度值转换为“度”，需要使用RadToDeg函数。

参阅
Abs、Acos、Asin、Atan2、Cos、DegToRad、RadToDeg、Sgn、Sin、Tan、Val

Atan函数使用示例

```
Function atantest
  Real x, y
  x = 0
  y = 1
  Print "Atan of ", x, " is ", Atan(x)
  Print "Atan of ", y, " is ", Atan(y)
Fend
```

3.5.54 Atan2函数

是返回连接坐标原点和指定点(X, Y)的直线角度的函数。单位是弧度。

格式

Atan2 (X坐标值, Y坐标值)

参数

X
　　以实值指定X坐标值。
Y
　　以实值指定Y坐标值。

返回值

用于返回已指定坐标的反正切值。范围是-π～π。

说明

Atan2(X, Y)是返回连接坐标原点(0, 0)与([x坐标值], [y坐标值])的直线的角度的函数。

此三角函数用于返回全部4个四分之一圆上的反正切值(角度)。

参阅

Abs、Acos、Asin、Atan、Cos、DegToRad、RadToDeg、Sgn、Sin、Tan、Val

Atan2函数使用示例

```
Function at2test
  Real x, y
  Print "Please enter a number for the X Coordinate:"
  Input x
  Print "Please enter a number for the Y Coordinate:"
  Input y
  Print "Atan2 of ", x, ", ", y, " is ", Atan2(x, y)
Fend
```

3.5.55 ATCLR

清除关节的实效转矩并进行初始化。

格式

ATCLR [关节指定1 [, 关节指定2 [, 关节指定3 [, 关节指定4 [, 关节指定5 [, 关节指定6 [, 关节指定7 [, 关节指定8 [, 关节指定9]]]]]]]]]

参数

关节指定1 - 关节指定9
以整数值或表达式指定关节编号。未指定参数时，所有关节的实效转矩值都会被清除。
附加轴的S轴为8，T轴为9。如果指定不存在的关节编号，将发生错误。

说明

ATCLR用于清除指定关节的实效转矩值。

执行ATRQ之前，请务必执行ATCLR。

参阅

ATRQ、PTRQ

ATCLR使用示例

[例1] 如下所示为清除所有关节的实效转矩值之后，显示指定关节编号转矩值的命令执行示例。

```
> atclr
> go p1
> atrq 1
     0.028
> atrq
     0.028    0.008
     0.029    0.009
     0.000    0.000
>
```

[例2] 如下所示为在垂直多关节机器人中清除J1、J4、J5的实效转矩值之后，显示指定关节编号转矩值的命令执行示例。

```
> atclr 4, 1, 5
> go p1
> ptrq 1
     0.227
> ptrq 4
     0.083
```

3.5.56 ATRQ

用于显示指定的关节的实效转矩。

格式
ATRQ [关节编号]

参数

关节编号
以整数值或表达式指定关节编号。可省略。附加轴的S轴为8，T轴为9。

结果
显示所有关节的当前实效转矩值。

说明
ATRQ用于显示指定的关节的实效转矩值。通过ATRQ，将会改变电动机的负载状态。以0～1的实值表示结果。最大实效转矩值是“1”。

执行ATRQ之前，请务必执行ATCLR。

ATRQ命令有时间限制。执行ATCLR后，请在60秒内执行ATRQ。如果超过60秒，将发生错误4030“扭矩执行值饱和状态”。

参阅
ATCLR、ATRQ函数、PTRQ

ATRQ使用示例

```
> atclr
> go p1
> atrq 1
     0.028
> atrq
     0.028    0.008
     0.029    0.009
     0.000    0.000
>
```

3.5.57 ATRQ函数

用于返回指定关节的实效转矩。

格式
ATRQ (关节编号)

参数

关节编号
以整数值或表达式指定关节编号。附加轴的S轴为8，T轴为9。

返回值
以0～1的实值返回。

说明
ATRQ函数用于返回指定关节的实效转矩。通过ATRQ函数，将会改变电动机的负载状态。以0～1的实值表示结果。最大实效转矩值是1。

执行ATRQ函数之前，请务必执行ATCLR。

ATRQ函数有时间限制。执行ATCLR后，请在60秒内执行ATRQ函数。如果超过60秒，将会发生错误4030。

参阅
ATRQ、PTCLR、PTRQ

ATRQ函数使用示例
如下所示为程序中使用ATRQ函数的示例。

```
Function CheckAvgTorque
  Integer i

  Go P1
  ATCLR
  Go P2
  Print "Average torques:"
  For i = 1 To 4
    Print "Joint ", i, " = ", ATRQ(i)
  Next i
Fend
```

3.5.58 AutoLJM

用于设置自动LJM。

格式
AutoLJM { On | Off }

参数

On | Off
- On：设置自动LJM为有效。
- Off：解除自动LJM。

说明
AutoLJM通过下述命令启用。

Arc、Arc3、Go、Jump3、Jump3CP、Move

如果执行AutoLJM On，与应用LJM函数时一样，将执行使关节移动量变为最小的动作，而与赋予各动作命令的位置数据中是否应用LJM函数无关。

例如，为了对 Go LJM(P1)获取相同效果，可以

```
AutoLJM On
Go P1
  AutoLJM Off
```

AutoLJM可以在程序的特定区间使LJM有效，所以就不需要变更各个动作命令。

如果执行AutoLJM Off，则仅在赋予各动作命令的位置数据中已应用LJM函数时，LJM函数的功能才有效。

下述情况时，AutoLJM表示在控制器设置中指定的设置状态(出厂时：Off)。
- 启动控制器时
- 执行Reset时
- 所有任务中断时
- 执行Motor On时
- 切换Auto / Programming作业模式时

注意

- AutoLJM与LJM函数的重复应用

  在AutoLJM On时，如果在赋予动作命令的点数据中应用LJM函数，则在执行动作时，LJM将变为重复应用。

  对于Move LJM(P1, Here)和Move LJM(P1)的动作命令，不会因为使AutoLJM.有效和使其无效而改变动作。但是，对Move LJM(P1, P0)的动作命令，在使AutoLJM有效时的Move LJM(LJM(P1, P0), Here)的动作和未使AutoLJM有效时的Move LJM(P1, P0)动作，可能会有不同的动作完成位置。

  建议设计程序使AutoLJM和LJM函数不会重复应用。

- AutoLJM使用注意事项

  根据控制器的环境设置，可以在控制器启动时启用AutoLJM功能。但是，如果通过控制器的环境设置和命令，将AutoLJM功能设为始终有效，对顾客打算大范围移动关节的动作命令，也将自动变更为关节移动量变少的姿势进行动作。

建议设计程序使用LJM函数和AutoLJM命令，仅在需要时应用LJM。

参阅

AuoLJM函数、LJM函数

AutoLJM使用示例

```
AutoLJM On
Go P1
Go P2
AutoLJM Off
```

3.5.59 AutoLJM函数

用于返回AutoLJM的状态。

格式

AutoLJM

返回值

- 0 = 关闭自动LJM
- 1 = 打开自动LJM

参阅

AutoLJM

AutoLJM函数使用示例

```
If AutoLJM = Off Then
    Print "AutoLJM is off"
EndIf
```

3.5.60 AutoOrientationFlag

用于变更N6-A1000**的姿势标志。

格式

AutoOrientationFlag { On | Off }

参数

On | Off

- On：将AutoOrientationFlag设为有效。
- Off：解除AutoOrientationFlag。(默认设置)

说明

AutoOrientationFlag通过下述命令启用。

Go、BGo、TGo、Jump3、JumpTLZ

用于变更下述姿势标志。

适用机型	参数OFF/ON	姿势标志			备注
		腕部	肘部	手腕	
N6-A1000**	OFF	-	-	-	以用户指定的姿势标志进行动作。(默认设置)
	ON	-	○	○*1	不了解姿势标志时，请选择“ON”。

○: 如果将AutoOrientationFlag设为“ON”，姿势标志将被变更。

*1：仅在变更肘姿势标志时，才会变更手腕姿势标志。变更手腕姿势标志时，将变为J4轴移动量最少的姿势标志。

与LJM函数并用

并用LJM函数时，手腕姿势Flag、J4Flag、J6Flag将变为以LJM函数选择的姿势。比如，将LJM函数的姿势标志选择设为“3”时，会选择J5轴移动量最小的手腕姿势Flag、J4Flag、J6Flag。另外，未并用LJM函数时，将选择J4轴移动量最小的手腕姿势Flag、J4Flag、J6Flag。

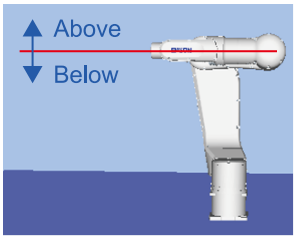
AutoOrientationFlag使用示例

```
Motor On
Power High
AutoOrientationFlag On

Go P1
Go P2
```

- 将AutoOrientationFlag设为“ON”时：

  标志根据P点与红线之间的位置关系发生如下变更。

- P点位于红线以上的位置：Above
- P点位于红线以下的位置：Below


Above
Below

3.5.61 AutoOrientationFlag函数

用于返回AutoOrientationFlag的状态。

格式

AutoOrientationFlag

返回值

- 0 = AutoOrientationFlag关闭
- 1 = AutoOrientationFlag打开

参阅

AutoOrientationFlag

AutoOrientationFlag函数使用示例

```
If AutoOrientationFlag = Off Then
    Print " AutoOrientationFlag is off"
EndIf
```

3.5.62 AvgSpeedClear

用于清除关节速度绝对值的平均值并进行初始化。

格式

AvgSpeedClear [关节指定1 [, 关节指定2 [, 关节指定3 [, 关节指定4 [, 关节指定5 [, 关节指定6 [, 关节指定7 [, 关节指定8 [, 关节指定9]]]]]]]]]

参数

关节指定1 - 关节指定9
以整数值或表达式指定关节编号。未指定参数时，所有关节速度绝对值的平均值会被清除。
附加轴的S轴为8，T轴为9。如果指定不存在的关节编号，将发生错误。

说明

AvgSpeedClear用于清除指定关节的速度绝对值的平均值。

请务必在执行AvgSpeed之前执行AvgSpeedClear。

PG附加轴不支持本命令。

参阅

AvgSpeed、PeakSpeed

AvgSpeedClear使用示例

[例1] 如下所示为清除所有关节的平均速度值之后，显示指定关节编号平均速度值的命令执行示例。

```
> AvgSpeedClear
> Go P1
> AvgSpeed 1
     0.073
> AvgSpeed
     0.073    0.044
     0.021    0.069
     0.001    0.108
     0.000    0.000
     0.000
>
```

[例2] 如下所示为在垂直多关节机器人中清除第1关节、第4关节、第5关节的平均速度值之后，显示指定关节编号平均速度值的命令执行示例。

```
> AvgSpeedClear 4, 1, 5
> Go P1
> AvgSpeed 1
     0.226
> AvgSpeed 4
     0.207
```

3.5.63 AvgSpeed

用于显示指定关节的速度绝对值的平均值。

格式
AvgSpeed [关节编号]

参数

关节编号
以整数值或表达式指定关节编号。可省略。附加轴的S轴为8，T轴为9。

结果
显示指定关节的当前速度绝对值的平均值。如未指定，则显示所有关节速度绝对值的平均值。

说明
AvgSpeed用于显示指定关节的速度绝对值的平均值。可利用AvgSpeed命令了解电动机的负载状态。以0～1的实值表示结果。最大平均速度值为“1”。

平均值为0.001以下时，将变为“0”。

请务必在执行AvgSpeed之前执行AvgSpeedClear。

AvgSpeed命令有时间限制。请务必在执行AvgSpeedClear之后60秒以内执行AvgSpeed。如果超出60秒，将发生错误4088。

为虚拟控制器方式以及空运行时，根据指令速度(而非实际速度)运算速度绝对值的平均值。

PG附加轴不支持本命令。

参阅
AvgSpeedClear、AvgSpeed函数、PeakSpeed

AvgSpeed使用示例

```
> AvgSpeedClear
> Go P1
> AvgSpeed 1
     0.226
> AvgSpeed
     0.226    0.133
     0.064    0.207
     0.003    0.314
     0.000    0.000
     0.000
>
```

3.5.64 AvgSpeed函数

用于返回指定关节的速度绝对值的平均值。

格式
AvgSpeed (关节编号)

参数

关节编号
以整数值或表达式指定关节编号。附加轴的S轴为8，T轴为9。

返回值
以0～1的实值返回。

说明
AvgSpeed函数用于返回指定关节的速度绝对值的平均值。可利用AvgSpeed函数了解电动机的负载状态。以0～1的实值表示结果。最大平均速度为1。

请务必在执行AvgSpeed函数之前执行AvgSpeedClear。

AvgSpeed函数有时间限制。请务必在执行AvgSpeed之后60秒以内执行AvgSpeed函数。如果超出60秒，将发生错误4088。

为虚拟控制器方式以及空运行时，根据指令速度(而非实际速度)运算速度绝对值的平均值。

PG附加轴不支持本函数。

参阅
AvgSpeed、AvgSpeedClear、PeakSpeed

AvgSpeed函数使用示例
如下所示为在程序中使用AvgSpeed函数的示例。

```
Function CheckAvgSpeed
  Integer i

  Go P1
  AvgSpeedClear
  Go P2
  Print "Average speeds:"
  For i = 1 To 6
    Print "Joint ", i, " = ", AvgSpeed (i)
  Next i
Fend
```

3.5.65 AvoidSingularity

用于设置奇点通过功能。

格式

AvoidSingularity { mode }

参数

mode

表示特殊点通过模式的整数表达式

常数	值	模式
SING_NONE	0	将奇点通过功能设为无效。
SING_THRU	1	将奇点通过功能设为有效。
SING_THRUROT	2	通过带ROT修饰语的CP动作使奇点通过功能有效。
SING_VSD	3	使变速CP动作功能有效。
SING_AUTO	4	自动选择奇点通过功能或变速CP动作功能。
SING_AVOID	5	将肘部奇点通过功能设为有效。

说明

AvoidSingularity通过下述命令启用。

Move、Arc、Arc3、Jump3、Jump3CP、JumpTLZ

奇点通过功能，是垂直6轴型机器人(包括N系列)与RS系列机器人在执行CP动作过程中靠近奇点时，为避免出现加速度错误，而保持原速度通过不同于原有轨迹的轨迹，并且可以在脱离奇点后返回正常的轨迹。控制器启动时，奇点回避功能会变为“1：有效”状态，通常无需变更，但在希望确保与不支持奇点通过功能的软件之间的兼容性的情况下，或诸如避免因奇点回避动作而导致轨迹偏移等不希望进行奇点回避的情况下，请将其设为无效。

变速CP动作功能在于垂直6轴型机器人(包括N系列)与RS系列机器人在执行CP动作过程中接近奇点时，为避免出现加速度错误或超速错误，在保持相同轨迹的同时自动控制速度。并且，可以在脱离奇点后恢复为正常的速度指令。为了在保持轨迹的同时通过特殊点附近，第1、第2、第4、第6关节可能会进行大的动作。

如已更改AvoidSingularity的设置值，则在下次起动控制器时有效。

起动控制器时，AvoidSingularity将变为由控制器设置指定的设置状态(工厂发货时：1)。另外，已变更AvoidSingularity的设置值时，SingularityAngle、SingularitySpeed、SingularityDist等参数会恢复为默认值。SING_AUTO是用于匹配SING_THRU与SING_VSD的模式。根据动作或速度选择SING_THRU或SING_VSD。

如果使用AvoidSingularity，仍发生加速度错误和超速错误，请将AccelS、DecelS、SpeedS调小。

注意

- 垂直6轴型机器人与N系列机器人的特异姿势近旁的条件设置

  使用第5关节的角度与第4关节的角速度，判断机器人是否接近手腕奇点。默认设置时，第5关节角度被设为±10度，第4关节的角速度被设为最大关节速度的±10%。如要变更此值，使用SingularityAngle和SingularitySpeed命令。另外，使用P点的坐标判断是否接近腕部奇点。默认设置时，P点与机器人第1关节旋转轴之间的距离被设为30 mm。如要对其进行变更，请使用SingularityDist命令。

- RS系列机器人奇点附近的条件设置

使用默认工具0坐标系原点的坐标，判断机器人是否接近腕部奇点。默认设置时，工具0坐标系的原点与机器人第1关节旋转轴之间的距离被设为30 mm。如要对其进行变更，请使用SingularityDist命令。

- N系列机器人的注意事项

  N2系列与其它机型不同，控制器启动时的奇点回避功能被设为“3：变速CP动作功能被设为有效”。

  N6系列与其它机型相同，控制器启动时的奇点回避功能被设为“1：奇点回避功能设为有效”。

  除了手腕特殊点姿势与腕部特殊点姿势之外，N系列还带有肘部奇点区域。肘部奇点区域是第3关节为0度时(第3关节与第2关节重叠时)的姿势。

  有关肘部奇点区域通过动作的详细说明，请参阅以下手册。
  《Epson RC+ 用户指南》

- SING_THRU与SING_AVOID的差异

  SING_THRU用于通过手腕奇点与肩部奇点，不通过肘部奇点。如要通过肘部奇点，请选择SING_AVOID。但是，通过肘部奇点时，相比其它奇点通过，轨道变化较大。因此使用时请注意。另外，如选择了非N系列选择SING_AVOID，将发生错误4002。

参阅

AvoidSingularity函数、SingualrityAngle、SingularitySpeed、SingularityDist

AvoidSingularity使用示例

```
AvoidSingularity SING_NONE '使奇点回避功能无效，并使其动作
Move P1
Move P2
AvoidSingularity SING_THRU
```

3.5.66 AvoidSingularity函数

返回AvoidSingularity的状态。

格式

AvoidSingularity

返回值

- 0 = 奇点通过功能无效
- 1 = 奇点通过功能有效
- 2 = 通过带ROT修饰语的CP动作命令使奇点通过功能有效。
- 3 = 用于使变速CP动作功能有效。
- 4 = 自动选择奇点通过功能与变速CP动作功能。
- 5 = 肘部特殊势通过功能有效。

参阅

AvoidSingularity

AvoidSingularity函数使用示例

```
If AvoidSingularity = SING_NONE Then
    Print "AvoidSingularity is off"
EndIf
```

3.6 B

3.6.1 Base

用于定义和显示基础坐标系。

格式

(1) Base 坐标系数据
(2) Base 原点, X轴指定, Y轴指定 [, { X | Y }]

参数

坐标系数据
　　直接以点数据指定基础坐标系的原点和方向。
原点
　　以P#(整数)或P(表达式)指定定义基础坐标系原点的机器人坐标系上的位置。
X轴指定
　　以P#(整数)或P(表达式)指定定义基础坐标系X轴上的点的机器人坐标系上的位置。
Y轴指定
　　以P#(整数)或P(表达式)指定定义基础坐标系Y轴上的点的机器人坐标系上的位置。
X
　　优先地使X轴指定与X轴一致。可省略。(默认设置)
Y
　　优先地使Y轴指定与Y轴一致。可省略。

说明

机器人上有无法变更的基准坐标系，我们称其为“机器人坐标系”。与此相对，我们将一般本地坐标系的基础、可以变更原点坐标的基本坐标系称为“基础坐标系”。

通过设置与机器人坐标系相对的本地坐标系的基点和旋转角来定义本地坐标系。

要将基础坐标系复位为默认值，请执行以下语句。一执行该语句，基础坐标系将与机器人坐标系相同。

```
Base XY(0, 0, 0, 0)
```

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

注意

- 如果变更基础坐标系，将会影响到所有本地定义。

  请在变更基础坐标系后，重新定义所有本地坐标系。

参阅

Local

Base使用示例

将基础坐标系的原点定义为X轴100 mm、Y轴100 mm的位置。

```
> Base XY(100, 100, 0, 0)
```

3.6.2 BClr函数

用于清除指定的1位数值并返回该值。

格式

BClr (数值, 位编号)

参数

数值
　　以表达式或直接以数值指定要清除位的数值。
位编号
　　以表达式或直接以数值指定要清除的位编号(0～31的整数值)。

返回值

返回已清除的位的值(整数)。

参阅

BClr64、BSet、BSet64、BTst、BTst64

BClr函数使用示例

```
flags = BClr(flags, 1)
```

3.6.3 BClr64函数

用于清除指定的1位数值并返回该值。

格式

BClr64 (数值, 位编号)

参数

数值
　　以表达式或直接以数值指定要清除位的数值。
位编号
　　以表达式或直接以数值指定要清除的位编号(0～63的整数值)。

返回值

返回已清除的位的值(整数)。

参阅

BClr、BSet、BSet64、BTst、BTst64

BClr64函数使用示例

```
flags = BClr64(flags, 1)
```

3.6.4 BGo

用于在已选择的本地坐标系上执行偏移PTP动作。

格式
Bgo目标坐标 [CP] [Till | Find] [!并行处理!] [SYNC]

参数

目标坐标
　　使用点数据，指定动作的目标位置。
CP
　　指定路径运动。可省略。
Till | Find
　　记述Till或Find表达式。可省略。

```
Till | Find
Till Sw(表达式) = {On | Off}
Find Sw(表达式) = {On | Off}
```

!并行处理!
　　动作期间可附加并行处理语句，以执行I/O等命令。可省略。
SYNC
　　预约动作命令。在通过SyncRobots开始动作之前，机器人不进行动作。

说明
用于在已选择的本地坐标系上执行偏移PTP动作。以由表示目标坐标的点数据指定的坐标系为基准，实施偏移PTP动作。

未指定本地坐标系时，以本地0(基础坐标系)为基准，实施偏移PTP动作。

无视由点数据提供的姿势标志并保持当前的姿势标志。但是，垂直6轴型机器人(包括N系列)会自动变更姿势标志，以减小关节移动量。这与在Move命令中指定LJM修饰参数时的情况相同。因此，要进行180度以上的姿势变化时，请分多次执行。

通过使用Till修饰符，可在Till条件成立时于动作中途对机器人进行减速停止，完成BGo动作。

使用Find修饰符并且动作期间Find条件变为真时，将点数据保存到FindPos中。

可使用!并行处理!，与动作并行执行其他处理。

如果附加了CP参数，则可在开始动作减速时叠加后续动作命令的加速。此时，不对目标坐标进行定位。

参阅
Accel、BMove、Find、!并行处理!、P# = 指定点、Speed、Till、TGo、TMove、Tool

BGo使用示例

```
> BGo XY(100, 0, 0, 0)    '(在本地坐标系中)向X方向移动100 mm

Function BGoTest

  Speed 50
  Accel 50, 50
  Power High

  P1 = XY(300, 300, -20, 0)
  P2 = XY(300, 300, -20, 0) /L
  Local 1, XY(0, 0, 0, 45)

  GoP1
  Print Here
```

```
  BGo XY(0, 50, 0, 0)
  Print Here

  Go P2
  Print Here
  BGo XY(0, 50, 0, 0)
  Print Here

  BGo XY(0, 50, 0, 0) /1
  Print Here

Fend
```

[输出结果]

```
 X:  300.000 Y:  300.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /R /0
 X:  300.000 Y:  350.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /R /0
 X:  300.000 Y:  300.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /L /0
 X:  300.000 Y:  350.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /L /0
 X:  264.645 Y:  385.355 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /L /0
```

3.6.5 BMove

用于在已选择的本地坐标系上执行偏移直线动作。

格式

BMove目标坐标 [ROT] [CP] [Till | Find] [!并行处理!] [SYNC]

参数

目标坐标
　　使用点数据，指定动作的目标位置。
ROT
　　以工具姿势变化优先，指定速度和加减速度。可省略。
CP
　　指定路径运动。可省略。
Till | Find
　　记述Till或Find表达式。可省略。

```
Till | Find
Till Sw(表达式) = {On | Off}
Find Sw(表达式) = {On | Off}
```

! 并行处理 !
　　动作期间可附加并行处理语句，以执行I/O等命令。可省略。
SYNC
　　预约动作命令。在通过SyncRobots开始动作之前，机器人不进行动作。

说明

用于在已选择的本地坐标系上执行偏移直线动作。以由表示目标坐标的点数据指定的坐标系为基准，实施偏移直线动作。

未指定本地坐标系时，以本地0(基础坐标系)为基准，实施偏移PTP动作。

无视由点数据提供的姿势标志并保持当前的姿势标志。但是，垂直6轴型机器人(包括N系列)会自动变更姿势标志，以减小关节移动量。这与在Move命令中指定LJM修饰参数时的情况相同。因此，要进行180度以上的姿势变化时，请分多次执行。

BMove的速度和加减速度分别使用SpeedS和AccelS的设置值。有关速度与加减速度之间的关系，请参阅“注意”中的“与BMove同时使用CP”。不过，指定ROT修饰参数时的速度和加减速度分别使用SpeedR和AccelR的设置值。此时，SpeedS和AccelS的设置值变为无效状态。

通常的移动距离为“0”，仅姿势变化时，会发生错误。通过附加ROT修饰参数并以工具姿势变化的加速度优先，可不出错误地进行动作。已经附加ROT修饰参数时，如果没有姿势变化，并且移动距离不是“0”，则会发生错误。

另外，相对于移动距离，工具姿势变化速度过大时，或指定的转速超过机械手限度时，也会发生错误。此时，请降低指定速度，或附加ROT修饰参数，并以姿势变化的加减速度优先。

通过使用Till修饰符，可在Till条件成立时于动作中途对机器人进行减速停止，完成BMove动作。

通过使用Find修饰符并且动作期间Find条件的值为真(True)时，将点数据保存到FindPos中。

可使用!并行处理!，与动作并行执行其他处理。

注意

- 与CP同时使用BMove

如果使用CP参数，动作命令则会在开始减速的同时将控制移交给下一语句。这在用户连续发出几个动作命令，要以一定的速度进行连续动作时非常便利。为未指定CP的BMove命令时，机械臂必须减速，以停在指定的目标位置上。

参阅

AccelS、BGo、Find、!并行处理!、P# = 指定点、SpeedS、TGo、Till、TMove、Tool

BMove使用示例

```
> BMove XY(100, 0, 0, 0)    '(在本地坐标系中)向X方向移动100 mm

Function BMoveTest

  Speed 50
  Accel 50, 50
  SpeedS 100
  AccelS 1000, 1000
  Power High

  P1 = XY(300, 300, -20, 0)
  P2 = XY(300, 300, -20, 0) /L
  Local 1, XY(0, 0, 0, 45)

  Go P1
  Print Here
  BMove XY(0, 50, 0, 0)
  Print Here

  Go P2
  Print Here
  BMove XY(0, 50, 0, 0)
  Print Here

  BMove XY(0, 50, 0, 0) /1
  Print Here

Fend
```

[输出结果]

```
 X:  300.000 Y:  300.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /R /0
 X:  300.000 Y:  350.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /R /0
 X:  300.000 Y:  300.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /L /0
 X:  300.000 Y:  350.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /L /0
 X:  264.645 Y:  385.355 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /L /0
```

3.6.6 Boolean

用于声明Boolean型变量。(大小：2字节)

格式

Boolean 变量名 [(数组变量的最大下标)] [,变量名 [(数组变量的最大下标)]...]

参数

变量名
　　指定声明为Boolean型的变量名。
数组变量的最大下标
　　可利用数组变量的最大下标声明到最高三维。使用下述格式。可省略。
　　(最大下标1, [最大下标2], [最大下标3])
　　由于下标从0开始，因此数组数为最大下标加上1。
　　在所有元素数不超过以下最大值的范围内指定各最大下标。

- 本地变量：2,000
- 备份变量(Global Preserve)： 4,000
- 全局变量和模块变量：100,000

说明

Boolean在将变量声明为Boolean型时使用。Boolean型变量为“True”或“False”之一。在Function开头声明本地变量。在Function之外声明全局变量和模块变量。

参阅

Byte、Double、Global、Int32、Int64、Integer、Long、Real、Short、String、UByte、UInt32、UINT64、UShort

Boolean使用示例

```
Boolean partOK
Boolean A(10)          'Boolean型的一维数组
Boolean B(10, 10)      'Boolean型的二维数组
Boolean C(5, 5, 5)     'Boolean型的三维数组

partOK = CheckPart()
If Not partOK Then
   Print "Part check failed"
EndIf
```

3.6.7 BOpen

用于以二进制访问模式打开文件。

格式

BOpen 文件名 As #文件编号
.
.
Close #文件编号

参数

文件名
　　指定包括路径的文件名字符串。仅指定文件名时，是指当前目录中的文件。详情请参阅ChDisk。
文件编号
　　以30～63之间的整数值或表达式进行指定。

说明

以指定的文件编号打开指定的文件。在以二进制访问模式访问指定文件时使用此语句。如果不存在指定文件，将创建新文件。如果存在，则从现有数据的开头读写数据。请在以二进制访问模式读取、写入数据时，分别使用ReadBin、WriteBin命令。

注意

可使用网络路径。

指定的文件编号用于在打开文件期间识别该文件。因此，在关闭该文件之前，不能使用与其他文件相同的文件编号。按文件操作命令(ReadBin、WriteBin、Seek、Eof、Flush、Close)使用文件编号。

利用Seek命令切换文件的读取和写入位置(指针)。切换读取访问和写入访问时，请利用Seek命令重新设置文件指针位置。

利用Close语句关闭文件并释放文件编号。

请利用FreeFile函数获取文件编号，以免在多个任务中使用同一编号。

参阅

Close、AOpen、FreeFile、ReadBin、ROpen、UOpen、WOpen、WriteBin

BOpen使用示例

```
Integer fileNum, i

fileNum = FreeFile
BOpen "TEST.DAT" As #fileNum
For i = 0 To 100
   WriteBin #fileNum, i
Next i

Flush #fileNum
Seek #fileNum, 10
ReadBin #fileNum, i
Print "data = ", i
Close #fileNum
```

3.6.8 Box

用于设置和显示进入检测区域。

格式

(1) Box 区域编号 [, 机器人编号], X轴下限位置, X轴上限位置, Y轴下限位置, Y轴上限位置, Z轴下限位置, Z轴上限位置 [本地编号]
(2) Box 区域编号, 机器人编号, X轴下限位置, X轴上限位置, Y轴下限位置, Y轴上限位置, Z轴下限位置, Z轴上限位置, 远程输出逻辑 [本地编号]
(3) Box 区域编号, 机器人编号
(4) Box

参数

区域编号
以1～15的整数值指定要设置的区域编号。
机器人编号
以整数值指定要设置的机器人编号。在 (1) 的格式中省略时，以当前选择的机器人为对象。在 (2)、(3) 的格式中不可省略。
X轴下限位置
以数值或表达式指定要设置的区域的下限位置X坐标值(实数)。
X轴上限位置
以数值或表达式指定要设置的区域的上限位置X坐标值(实数)。
Y轴下限位置
以数值或表达式指定要设置的区域的下限位置Y坐标值(实数)。
Y轴上限位置
以数值或表达式指定要设置的区域的上限位置Y坐标值(实数)。
Z轴下限位置
以数值或表达式指定要设置的区域的下限位置Z坐标值(实数)。
Z轴上限位置
以数值或表达式指定要设置的区域的上限位置Z坐标值(实数)。
远程输出逻辑
On | Off
设置远程输出逻辑。如要在Box进入时打开I/O输出，可设置为On；如要在Box进入时关闭I/O输出，可设置为Off。省略参数时，将设置为On。
本地编号
指定本地坐标系编号(0～15)。
请务必在编号之前附加“/LOCAL”。如果省略了参数，将设置本地坐标系编号0。

结果

- 如果以(3)的格式继续指定，则显示指定区域编号的区域设置。
- 如果以(4)的格式指定，则显示当前选择机器人所设置的所有区域设置。

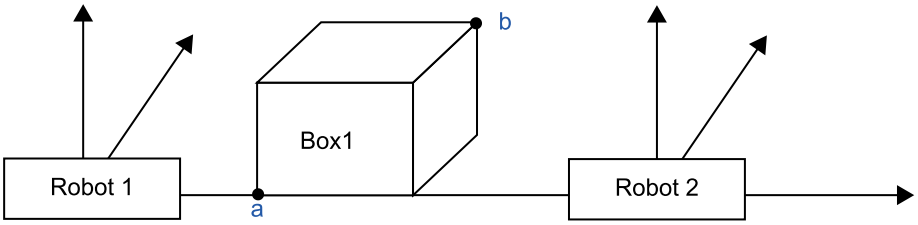
说明

Box用于设置进入检测区域。设置进入检测区域，可以检测利用当前选择的工具所计算的卡爪工具位置是否进入到设置的进入检测区域内。进入检测区域被设在机器人的基础坐标系上或由本地编号指定的本地坐标系上，在基础坐标系的X、Y、Z的各轴中指定的下限位置与上限位置之间为进入检测区域。

如果设置进入检测区域，控制器启动期间则始终执行检测处理，与机器人的电动机电源状态无关。

进入检测的结果，可以随时使用GetRobotInsideBox函数和 InsideBox函数获取。另外，作为Wait命令的等待条件表达式，可利用GetRobotInsideBox函数。也可以进行远程输出设置，以便将检测结果输出到I/O中。

多个机器人共享一个区域时，需要定义从各机器人坐标看到的区域。



符号	说明
a	X, Y, Z轴下限位置
b	X, Y, Z轴上限位置

从Robot 1看到的Box 1的设置

```
Box 1, 1, 100, 200, 0, 100, 0, 100
```

X、Y、Z轴下限位置的坐标是(100,0,0)，X、Y、Z轴上限位置的坐标是(200,100,100)

从Robot 2看到的Box 1的设置

```
Box 1, 2, -200, -100, 0, 100, 0, 100
```

X、Y、Z轴下限位置的坐标是(-200,0,0)，X、Y、Z轴上限位置的坐标是(-100,100,100)

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

注意

- 按各坐标轴关闭进入检测区域的设置

  可以按各坐标轴关闭设置。例如，要关闭Z轴的设置，请将Z轴下限位置和Z轴上限位置都设为“0”。例: Box 1, 200, 300, 0, 500, 0, 0

  在这种情况下，判别是否进入了XY的二维平面。

- 进入检测区域的默认值

  Box的默认值为“0, 0, 0, 0, 0, 0”。(进入检测区域的设置为OFF状态。)

- 工具选择

  进入检测是通过当前选择的工具进行的。因此，已变更工具选择时，虽然机器人不会进行动作，但可能会出现从区域内到区域外或相反的情况。

- 附加轴

  机器人带有附加轴(包括移动轴)S、T轴时设置的进入检测平面不依赖于附加轴的位置。而是以机器人的基础坐标系为基准进行设置。

提示

- 利用机器人管理器设置Box

  可通过Epson RC+的[工具]菜单-[机器人管理器]的[工作空间]面板设置Box值。

参阅

BoxClr、BoxDef、GetRobotInsideBox、InsideBox、Plane

Box使用示例

[例1] 如下所示为通过命令窗口设置Box值并显示当前值的简单操作示例。

```
> Box 1, -200, 300, 0, 500, -100, 0

> Box
Box 1: 1, -200.000, 300.000, 0.000, 500.000, -100.000, 0.000, ON /LOCAL0
```

[例2] 如下所示为在本地编号中指定1、2并设置Box值的简单程序。

```
Function SetBox

   Integer i

   Box 1, -200, 300, 0, 500, -100, 0 /LOCAL1

   i = 2
   Box 2, 100, 200, 0, 100, -200, 100 /LOCAL(i)

Fend
```

3.6.9 Box函数

用于返回设置的进入检测区域。

格式

Box (区域编号 [, 机器人编号], 引用数据)

参数

区域编号
　　以表达式或数值指定要确认的区域编号。
机器人编号
　　以整数值指定要设置的机器人编号。省略时，以当前选择的机器人为对象。
引用数据
　　以整数值指定作为返回值返回的数据。- 1：下限位置 - 2：上限位置

返回值

如果将引用数据指定为1，则将由Box设置值指定的X轴下限位置作为点数据X返回；将Y轴下限位置作为点数据Y返回；将Z轴下限位置作为点数据Z返回。

如果将引用数据指定为2，则将由Box设置值指定的X轴上限位置作为点数据X返回；将Y轴上限位置作为点数据Y返回；将Z轴上限位置作为点数据Z返回。

参阅

Box、BoxClr、BoxDef、GetRobotInsideBox、InsideBox

Box函数使用示例

```
P1 = Box(1,1)
P2 = Box(1,2)
```

3.6.10 BoxClr

用于清除已设置的Box。

格式

BoxClr 区域编号 [, 机器人编号]

参数

区域编号
　　以表达式或数值指定要清除设置的进入检测区域编号(1～15的整数)。
机器人编号
　　以整数值指定要设置的机器人编号。已省略时，以当前选择的机器人为对象。

说明

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

参阅

Box、BoxDef、GetRobotInsideBox、InsideBox

BoxClr函数使用示例

如下所示为使用BoxClr函数的程序。

```
Function ClearBox

   If BoxDef(1) = True Then
      BoxClr 1
   EndIf
Fend
```

3.6.11 BoxDef函数

用于返回是否已设置进入检测区域。

格式

BoxDef 区域编号[, 机器人编号])

参数

区域编号
　　指定返回状态的进入检测区域编号(1～15的整数)。
机器人编号
　　以整数值指定要设置的机器人编号。已省略时，以当前选择的机器人为对象。

返回值

由区域编号指定的进入检测区域已设置时返回“True”，未设置时返回“False”。

参阅

Box、BoxClr、GetRobotInsideBox、InsideBox

BoxDef函数使用示例

如下所示为使用BoxDef函数的程序。

```
Function ClearBox

   If BoxDef(1) = True Then
      BoxClr 1
   EndIf
Fend
```

3.6.12 Brake

用于打开和关闭当前机器人指定关节的制动器。

格式

Brake 状态, 关节编号

参数

状态

- 施加制动时：使用On。
- 解除制动时：使用Off。

关节编号

指定1～6的关节编号。

说明

Brake命令用于对垂直6轴型机器人(包括N系列)的一个关节施加或解除制动。水平多关节机器人(包括RS系列)无法使用。

此命令设计为只有维修作业人员才可以使用。

如果执行Brake命令，则会对机器人控制参数进行初始化。详情请参阅Motor On。

⚠ 警告

关闭制动时，请注意。请务必确认是否正确保持了关节请务必确认是否正确维持了关节的状态。如果关节未被正确维持其状态，可能会导致坠落、或发生机器人故障和导致作业人员身受重伤。

注意

- 请务必将紧急停止开关放置在身边后执行Brake Off命令。

  如果控制器变为紧急停止状态，电动机制动器将被锁定。执行Brake Off命令，机器人机械臂可能因为自身重量而下降。

参阅

Motor、Power、Reset、SFree、SLock

Brake使用示例

```
> brake on, 1

> brake off, 1
```

3.6.13 Brake函数

用于返回指定关节的制动状态。

格式

Brake (关节编号)

参数

关节编号
以整数值或整数表达式指定关节编号。可以指定的值是从1到机器人的关节数。

返回值

0 = 制动Off、1 = 制动On

参阅

Brake

Brake函数使用示例

```
If brake (1) = OFF Then
  Print "Joint 1 brake is off"
EndIf
```

3.6.14 BSet函数

用于设置指定数值的第1位并返回结果。

格式

Bset (数值, 位编号)

参数

数值
　　以表达式或直接以数值指定要设置位的数值。
位编号
　　以表达式或直接以数值指定要设置的位的编号(0～31的整数值)。

返回值

返回已设置的位的值(整数)。

参阅

BClr、BClr64、BSet64、BTst、BTst64

BSet函数使用示例

```
flags = BSet(flags, 1)
```

3.6.15 BSet64函数

用于设置指定数值的第1位并返回结果。

格式

BSet64 (数值, 位编号)

参数

数值
　　以表达式或直接以数值指定要设置位的数值。
位编号
　　以表达式或直接以数值指定要设置的位的编号(0～63的整数值)。

返回值

返回已设置的位的值(整数)。

参阅

BClr、BClr64、BSet、BTst、BTst64

BSet函数使用示例

```
flags = BSet64(flags, 1)
```

3.6.16 BTst函数

用于返回数值的1位状态。

格式

BTst (数值, 位编号)

参数

数值
　　以表达式或直接以数值指定要进行位测试的数值。
位编号
　　以数值指定要测试的位编号(0～31的整数值)。

返回值

以1或0返回位测试的值。

参照

BClr、BClr64、BSet、BSet64、BTst64

BTst函数使用示例

```
If BTst(flags, 1) Then
    Print "Bit 1 is set"
EndIf
```

3.6.17 BTst64函数

用于返回数值的1位状态。

格式

BTst64 (数值, 位编号)

参数

数值
　　以表达式或直接以数值指定要进行位测试的数值。
位编号
　　以数值指定要测试的位编号(0～63的整数值)。

返回值

以1或0返回位测试的值。

参照

BClr、BClr64、BSet、BSet64、BTst

BTst64函数使用示例

```
If BTst64(flags, 1) Then
    Print "Bit 1 is set"
EndIf
```

3.6.18 Byte

用于声明Byte型变量。(大小：2字节)

格式

Byte 变量名 [(数组变量的最大下标)] [, 变量名 [(数组变量的最大下标)]…]

参数

变量名
　　指定声明为Byte型的变量名。
数组变量的最大下标
　　可利用数组变量的最大下标声明到最高三维。使用下述格式。可省略。
　　(最大下标1, [最大下标2], [最大下标3])
　　由于下标从0开始，因此元素数为最大下标加上1。在所有元素数不超过以下最大值的范围内指定各最大下标。

- 本地变量：2,000
- 备份变量(Global Preserve)：4,000
- 全局变量和模块变量：100,000

说明

Byte在将变量声明为Byte型时使用。Byte变量的范围是-128～+127。在Function开头声明本地变量。在Function之外声明全局变量和模块变量。

参阅

Boolean, Double, Global, Int32, Int64, Integer, Long, Real, Short, String, UByte, UInt32, UInt64, UShort

Byte使用示例

下例为声明Byte型变量并为该变量赋值。

查看变量test_ok的最高位是1还是0。其结果将显示在显示器中。(在此例中，为变量赋值15，所以始终设置了变量test_ok值的高位。)

```
Function Test
    Byte A(10)          'Byte型的一维数组
    Byte B(10, 10)      'Byte型的二维数组
    Byte C(5, 5, 5)     'Byte型的三维数组
    Byte test_ok
    test_ok = 15
    Print "Initial Value of test_ok = ", test_ok
    test_ok = (test_ok And 8)
    If test_ok <> 8 Then
       Print "test_ok high bit is ON"
    Else
       Print "test_ok high bit is OFF"
    EndIf
Fend
```

3.7 C

3.7.1 Calib

用于将当前的机械臂姿势的脉冲值转换为由Calpls设置的脉冲值。

格式
Calib 关节编号1 [, 关节编号2] [, 关节编号3] [, 关节编号4] [, 关节编号5] [, 关节编号6] [, 关节编号7] [, 关节编号8] [, 关节编号9]

参数

关节编号
直接以数值指定进行校准的关节编号(1～9的整数)。通常每个关节进行一次校准，但在该命令中，可以同时进行9个关节的校准。附加轴的S轴为8，T轴为9。

说明
自动运算并设置偏移(Hofs)值。为对准与各关节数据原点相对应的机器人的机械原点，需要此偏移。

Calib是在改变电动机的脉冲值时使用的命令。一般性的用法是用作更换电动机后的作业。校准位置的脉冲值一般应与CalPls脉冲值相同，但是在更换电动机等的维修作业后，这两个值会出现不一致的情况。为此，就需要进行校准。

在将机械臂移至要校准的位置后执行Calib命令。执行Calib，校准位置的脉冲值将变为CalPls脉冲值(相对于校准位置的正确的脉冲值)。

为了正确地进行校准，必须设置Hofs值。在将机械臂移至进行校准的位置后，执行Calib，将自动运算Hofs值。控制器将根据校准脉冲值和CalPls脉冲值自动运算Hofs值。

注意

- 使用Calib命令时需要注意

  Calib只为维修作业而设计。除非必要，请不要使用。

  执行Calib，Hofs值将被替换。如果Hofs值在不知不觉之间被更改，机器人可能做出意想不到的动作，所以除非必要，请绝对不要使用Calib。

  支持关节精度补偿功能的机型，当执行Calib时，指定轴的关节的JointAccuracy中设置的校正值会变为“0”。

- 当使用安装有Safety板的控制器时，请在运行本命令后启动安全功能管理器

  使用安装Safety板的控制器时，控制器的Hofs值和具有安全功能的Safety板的Hofs值必须一致。

  执行此命令时，只会改变控制器的Hofs值，并且因为与Safety板的Hofs值不同，会发生警报。

  因此，执行本命令后，请启动安全功能管理器更新Safety板的设置。

  有关详细信息，请参阅以下手册。
  《机器人控制器 安全功能手册》

常见错误
如果未指定关节编号，将会出错

使用Calib命令时，如果未指定关节编号，将会出错。

参阅

CalPls, JointAccuracy, HofsJointAccuracy, Hofs

Calib使用示例

命令窗口的操作

```
> CalPls     '显示当前的CalPls值
  65523, 43320, -1550, 21351
> Pulse     '显示当前的Pulse值
PULSE: 1: 65526 pls 2: 49358 pls 3: 1542 pls 4: 21299 pls
> Calib 2     '仅第2关节执行校准
> Pulse     '显示已变更的Pulse值
PULSE: 1: 65526 pls 2: 43320 pls 3: 1542 pls 4: 21299 pls
>
```

3.7.2 Call

作为子例程调用函数。

格式
Call 函数名 [(自变量列表)]

参数

函数名称
　　指定要调用的函数名。
自变量列表
　　指定由函数声明指定的自变量列表。自变量请使用下述格式。可省略。[ByRef] 变量名 [( )]、或算式
ByRef
　　参照要调用的函数的变量时，指定ByRef。此时，可以将函数内的自变量的变更反映到调用的变量中。可以变更由参照赋予的值。可省略。

说明
通过Call命令将程序控制移至Function...Fend定义的函数。通过Call命令，程序执行从当前函数移至Call命令指定的函数。在Exit Function找到Fend之前，将按照直接调用的函数继续执行程序。而且，控制通过Call命令的下一语句返回原来的函数。

Call关键字和自变量的括弧可以省略。请参阅下例。

```
Call MyFunc(1, 2)
MyFunc 1, 2
```

可用于调用DLL(动态链接库)定义的外部函数。详情请参阅Declare语句。

要在函数内执行子例程，请使用GoSub...Return。

也可以指定变量作为自变量。可以指定ByRef参数，并将函数内的自变量的变更反映到调用的变量中。

指定ByRef参数时，函数定义(Function语句)和DLL函数定义(Declare语句)的自变量列表也一样需要指定ByRef。

将数组变量作为自变量进行赋值时，必须使用ByRef。

参阅
Function、GoSub

Call使用示例

```
[File1: MAIN.PRG]
Function main
    Call InitRobot
Fend

[File2: INIT.PRG]
Function InitRobot

    If Motor = Off Then
        Motor On
    EndIf
    Power High
    Speed 50
    Accel 75, 75
Fend
```

3.7.3 CalPls

进行校准所使用的位置姿势的脉冲值的设置和显示。

格式

(1) CalPls 第1关节脉冲值, 第2关节脉冲值, 第3关节脉冲值, 第4关节脉冲值 [,第5关节脉冲值, 第6关节脉冲值] [, 第7关节脉冲值] [, 第8关节脉冲值, 第9关节脉冲值]
(2) CalPls

参数

第1关节脉冲值
　　以表达式或数值指定第1关节的脉冲值(整数)。
第2关节脉冲值
　　以表达式或数值指定第2关节的脉冲值(整数)。
第3关节脉冲值
　　以表达式或数值指定第3关节的脉冲值(整数)。
第4关节脉冲值
　　以表达式或数值指定第4关节的脉冲值(整数)。
第5关节脉冲值
　　以表达式或数值指定第5关节的脉冲值(整数)。可省略。
第6关节脉冲值
　　以表达式或数值指定第6关节的脉冲值(整数)。可省略。
第7关节脉冲值
　　以表达式或数值指定第7关节的脉冲值(整数)。可省略。
第8关节脉冲值
　　以表达式或数值指定第8关节的脉冲值(整数)。可省略。
第9关节脉冲值
　　以表达式或数值指定第9关节的脉冲值(整数)。可省略。

返回值

如果省略脉冲值，则显示当前设置的脉冲值。

说明

用于输入和保存要进行校准的位置的正确的脉冲值。

此命令用于维护。在更换电动机时等，电动机原点偏离机械手机械臂的机械原点时使用。我们称使此原点一致的作业为校准。

在正常状态下，校准位置的脉冲值和CalPls设置的脉冲值一致,但是在进行更换电动机等的维修后，这两个值变得不一致，需要进行校准。

作为一种校准方法，有将机械臂移至特定位置上再执行Calib的方法。通过执行Calib，校准的特定位置的脉冲值将变为CalPls设置的脉冲值(相对于校准位置的正确的脉冲值)。

必须设置Hofs值，以执行校准。为自动计算Hofs值，将机械臂移至要校准的位置后执行Calib。控制器将根据校准位置的脉冲值自动运算Hofs值。

注意

- 无法通过关闭电源来更改CalPls值

  即使在关闭控制器的电源后重启，也无法初始化CalPls值。更改CalPls值的方法只有执行Calib命令。

参阅

Calib、Hofs

CalPls使用示例

监视器窗口操作

```
> CalPls     ' 显示当前的CalPls值
65523, 43320, -1550, 21351
> Pulse
PULSE: 1: 65526 pls  2: 49358 pls  3: -1542 pls  4: 21299 pls
> Calib 4
> Pulse
PULSE: 1: 65526 pls  2: 49358 pls  3: -1542 pls  4: 21351 pls
>
```

3.7.4 CalPls函数

用于返回CalPls设置的校准所使用的脉冲值。

格式
CalPls (关节编号)

参数

关节编号
以表达式或数值指定要参照的关节编号、或者返回有无CalPls设置的0。附加轴的S轴为8，T轴为9。

返回值
返回指定关节的CalPls设置值(整数值，单位：脉冲)。如果关节编号是0，返回有无CalPls设置(1或0)。

参阅
CalPls

CalPls函数使用示例
是使用CalPls函数的程序例。

```
Function DisplayCalPlsValues
   Integer i

   Print "CalPls Values:"
   For i = 1 To 4
      Print "Joint ", i, " CalPls = ", CalPls(i)
   Next i
Fend
```

3.7.5 ChDir

变更和显示当前目录。

格式

(1) ChDir 路径名
(2) ChDir

参数

路径名
直接以字符串或字符串表达式指定变更目标的路径名。详情请参阅ChDisk。

说明

- (1) 如果指定参数，将变更为已指定的目录。
- (2) 如果省略参数，将显示当前目录。用于不明确当前目录不明确之时。

当磁盘为PC时可执行。

从程序执行本命令时，请在路径名称的前后加上[″]符号。

当前目录是上次在所连接的控制器上创建的项目所在的目录。如果控制器没有创建信息，则根目录将是当前目录。

如果已通过ChDrive更改了驱动器，根目录将变为当前目录。

参数被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

参阅

ChDrive, ChDisk, CurDir$

ChDir使用示例

命令窗口操作示例

```
> ChDir \          '将当前目录变更为根目录
> ChDir..          '将当前目录变更为父目录

> Cd \TEST\H55     '将当前目录变更为\TEST\H55目录

> Cd               '显示当前目录
A:\TEST\H55\
```

程序执行示例

```
ChDir "\"          '将当前目录变更为根目录
ChDir ".."         '将当前目录变更为父目录
```

3.7.6 ChDisk

用于设置要进行文件操作的对象磁盘。

格式
ChDisk PC|USB|RAM

参数

PC
　　PC部分上的文件夹(硬盘等)
USB
　　SPEL+控制部分的USB存储器
RAM
　　SPEL+控制部分的存储器

说明
用于指定要进行文件操作的对象磁盘。默认是PC。

在机器人控制器中，作为文件操作的对象支持以下磁盘。

- PC：以PC部分的文件夹为对象。

  在PC上进行接通电源时的设置，在一般情况下，无需通过PC进行变更。

  可以访问放置在项目文件夹中的文件。

- USB：以与控制器的存储器端口连接的USB存储器为对象。在不进行与RC+的联合等不使用PC部分的情况下，想要进行文件交换时使用。

  T/VT系列机器人无法在参数中指定USB。

- RAM：以存储器上的临时文件为对象。

  如果关闭控制器的电源，将不保存文件。

  用于保管临时性的数据。

SPEL+的命令中有通过ChDisk的设置改变和不改变文件操作对象的命令。还有仅在PC上有效的ChDisk的设置命令。

- 不受ChDisk、ChDrive、ChDir的影响
  - Curve、CVMove、LoadPoints、SavePoints、ImportPoints的文件名
  - 始终以项目的文件夹为对象。
  - 仅可指定文件名，如果指定路径将会出错。
- 不受ChDisk的影响
  - OpenDB的Access、Excel文件名、ImportPoints的源路径、VLoadModel、VSaveImage、VSaveModel
  - 始终以Windows的文件夹为对象。
  - 仅指定文件名时，将会受当前驱动器和当前文件夹的影响。还可以指定全路径。
- ChDisk只有在PC上才可以执行
  - ChDir、FolderExists、MkDir、RenDir、RmDir
  - 如果按照PC以外的设置执行ChDisk，将会出错。
  - 仅指定文件名和目录名时，将会受当前驱动器和当前文件夹的影响。还可以指定全路径。

    USB和RAM中没有目录的概念。

- ChDisk在USB和RAM上也可以执行
  - Copy, Del, FileDataTime, FileExist, FileLen, AOpen, BOpen, ROpen, UOpen, WOpen, Rename
  - 在PC上执行ChDisk时：仅指定文件名和目录名时，将会受当前驱动器和当前文件夹的影响。还可以指定全路径。
  - 在USB和RAM上执行ChDisk时：仅可指定文件名，如果指定路径将会出错
- 特殊
  - Declare
  - 详情参阅Declare。直接处理指定的文件名。不受当前驱动器和当前文件夹的影响。

在PC上的ChDisk的全路径的确定方法如下所示。

- 仅文件名 `"abc.txt"`

  当前驱动器+当前目录+指定文件名

  `"C:\EpsonRC80\Projects\ProjectName\abc.txt"`

- 无驱动器的全路径 `"\abc.txt"`

  当前驱动器+指定全路径

  `"C:\abc.txt"`

- 有驱动器的全路径 `"d:\abc.txt"`

  直接采用指定全路径

  `"d:\abc.txt"`

- 驱动器是网络文件夹 `"k:\abc.txt"`

  直接采用指定全路径

  `"k:\abc.txt"`

- 网络路径 `"\\Epson\data\abc.txt"`

  直接采用指定全路径

  `"\\Epson\data\abc.txt"`

ChDisk的设置是控制器中的一个设置。

如果作为系统以多个磁盘为对象，需要进行专用控制并切换ChDisk的设置。

参阅

ChDir、ChDrive、CurDisk$

ChDisk使用示例

命令窗口操作示例

```
> ChDisk PC
```

3.7.7 ChDrive

更改当前驱动器。

格式
ChDrive 盘符

参数

盘符
　　指定有效的驱动器(1个字符)。

说明
可在PC硬盘时执行。

打开电源时，在关闭项目的状态下，C将变为当前驱动器。在打开项目的状态下，有该项目的驱动器将变为当前驱动器。

详情请参阅ChDisk。

参数被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

参阅
ChDir, ChDisk, CurDrive$

ChDrive使用示例
命令窗口操作示例

```
> ChDrive d
```

3.7.8 ChkCom函数

用于返回通信端口的接收缓冲器内的字符数。

格式

ChkCom (通信端口编号)

参数

通信端口编号

以整数值指定RS-232C的端口编号。

- SPEL+ 控制部分：1～8
- PC部分：1001~1008

返回值

返回接收字符数(整数值)。

如果不存在接收数据，以下述负值返回端口状态。

- -2：其他任务正在使用端口
- -3：未打开端口

参阅

CloseCom、OpenCom、Read、Write

ChkCom函数使用示例

```
Integer numChars

numChars = ChkCom(1)
```

3.7.9 ChkNet函数

用于返回网络端口的接收缓冲器内的字符数。

格式

ChkNet (通信端口编号)

参数

通信端口编号
　　指定TCP/IP端口编号(201～216)。

返回值

返回接收字符数(整数值)。

如果不存在接收数据，以下述负值返回端口状态。

- -1：端口已打开，但是未确立通信
- -2：其他任务正在使用端口
- -3：未打开端口

参阅

CloseNet、OpenNet、Read、Write

ChkNet函数使用示例

```
Integer numChars

numChars = ChkNet(201)
```

3.7.10 Chr$函数

用于返回相对于ASCII代码的字符。

格式
Chr$ (代码编号)

参数

代码编号
　　指定1～255的整数值。

返回值
返回与指定代码相对应的字符。

说明
Chr$用于返回与ASCII代码相对应的字符(1个字符)。如果将参数设置到1～255以外，将会出错。

参阅
Asc、InStr、Left$、Len、Mid$、Right$、Space$、Str$、Val

Chr$函数使用示例
在下述示例中，声明字符串变量并设置字符串“ABC”。Chr$函数用于将ASCII代码转换为“A”、“B”、“C”。&H意味着后续数值是16进制数。(&H41是16进制数的41)

```
Function Test
    String temp$
    temp$ = Chr$(&H41) + Chr$(&H42) + Chr$(&H43)
    Print "The value of temp = ", temp$
Fend
```

3.7.11 ClearHistory

删除全部命令执行记录。

格式
ClearHistory

说明
删除保存的全部命令执行记录。

注意

仅可利用命令窗口执行该命令。

参阅
History

ClearHistory使用示例

```
> Motor On
> Go P1
> History
Motor On
Go P1

> ClearHistory
> History
```

3.7.12 ClearPoints

用于删除机器人的存储器上的位置数据。

格式
ClearPoints

说明
ClearPoints用于将机器人的存储器上的位置数据区域中的数据初始化。在示教新位置之前，在删除存在于存储器上的位置数据时使用此命令。

参阅
Plist、LoadPoints、Savepoints

ClearPoints使用示例
下例是通过命令窗口使用ClearPoints命令的简单示例。请在执行Clearpoints命令后执行Plist命令，并确认不存在示教点。

```
>P1=100,200,-20,0/R
>P2=0,300,0,20/L
>plist
P1=100,200,-20,0/R
P2=0,300,0,20/L
>clearpoints
>Plist
>
```

3.7.13 Close

用于关闭由AOpen、BOpen、ROpen、UOpen、WOpen命令打开的文件。

格式

Close #文件编号

参数

文件编号
以30～63之间的整数值或表达式进行指定。

说明

用于关闭正在按照文件编号参照的文件，并释放文件编号。

参阅

AOpen, BOpen, Flush, FreeFile, Input #, Print #, ROpen, UOpen, WOpen

Close使用示例

下例所示为打开文件并写入数据，然后再次打开相同文件读取数据到数组中。

```
Integer fileNumber, i, j

fileNumber = FreeFile
WOpen "TEST.DAT" As #fileNum
For i = 0 To 100
    Print #fileNum, i
Next i
Close #fileNum

FileNum = FreeFile
ROpen "TEST.DAT" As #fileNum
For i = 0 to 100
    Input #fileNum, j
    Print j
Next i
Close #fileNum
```

3.7.14 CloseCom

用于关闭通过OpenCom打开的RS-232C端口。

格式

CloseCom #通信端口编号 | All

参数

通信端口编号

指定要关闭的RS-232C的端口编号。

- SPEL+ 控制部分：1～8
- PC部分：1001~1008

如果指定All，将关闭该任务打开的所有RS-232C端口。

参阅

ChkCom、OpenCom

CloseCom使用示例

```
CloseCom #1
```

3.7.15 CloseDB

用于关闭通过OpenDB命令打开的数据库并释放数据库编号。

格式

CloseDB #数据库编号

参数

数据库编号
　　指定利用OpenDB指定的数据库编号(501～508的整数值)。

说明

用于关闭数据库、Excel工作薄，并释放数据库编号。

注意

- 需要连接已安装RC+的PC。

参阅

OpenDB, SelectDB, UpdateDB, DeleteDB, Input #, Print #

CloseDB使用示例

请参阅OpenDB使用示例。

3.7.16 CloseNet

用于关闭OpenNet打开的TCP/IP端口。

格式

CloseNet #通信端口编号|All

参数

通信端口编号
　　指定要关闭的TCP/IP端口编号(201～216)。
　　如果指定All，将关闭该任务打开的所有TCP/IP端口。

参阅

ChkNet、OpenNet

CloseNet使用示例

```
CloseNet #201
```

3.7.17 Cls

用于清除Epson RC+的运行窗口、操作窗口、命令窗口中的文本区域。

运行窗口清除TP的打印面板。

格式
(1) Cls #装置ID
(2) Cls

参数

装置ID

- 21 RC+
- 24 TP(仅TP1)
- 20 TP3

省略时，显示装置将成为对象。

说明
如果从Epson RC+的命令窗口的程序执行Cls，将清除命令窗口的文本区域。

如果在程序中执行，将清除由装置ID指定的装置的画面。

如果省略装置ID，将清除显示装置的画面。

Cls使用示例
如果从运行窗口或操作员窗口执行次程序例，在执行Cls后将清除文本区域。

```
Function main
    Integer i

    Do
        For i = 1 To 10
            Print i
        Next i
        Wait 3
        Cls
    Loop
Fend
```

3.7.18 Cnv_AbortTrack

用于中断传送带队列点的跟踪动作。

格式
Cnv_AbortTrack [停止Z高度]

参数

停止Z高度
以实值指定停止时的机器人的Z位置。可省略。

说明
在执行传送带队列点(与传送带的移动同步的点)的动作命令中，Cnv_AbortTrack用于停止动作命令。

如果已指定停止Z高度参数，仅在指定值高于当前Z位置时才实施转移至指定值的动作，然后使跟踪动作减速停止。

如果省略停止Z高度参数，机械手将使跟踪动作减速停止而不实施Z方向的转移动作。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅
Cnv_RobotConveyor

Cnv_AbortTrack使用示例

```
' 监视跟踪工件流动到下游传送带上的机器人的任务

Function WatchDownstream

  Robot 1
  Do
    If g_TrackInCycle And Cnv_QueLen(1, CNV_QUELEN_DOWNSTREAM) > 0 Then
      ' 停止当前机器人的跟踪动作，并将z轴移动到0位置
      g_AbortTrackInCycle = TRUE
      Cnv_AbortTrack 0
      g_AbortTrackInCycle = FALSE
    EndIf
    Wait 0.01
  Loop
Fend
```

3.7.19 Cnv_Accel

用于设置传送带跟踪动作前的加减速度。

格式

Cnv_Accel 传送带编号, 加速度和减速度

参数

传送带编号
　　以整数值(1～16)指定传送带的编号。
加速度和减速度
　　设置跟踪动作的加速度和减速度。

说明

用于设置传送带跟踪动作的加速度和减速度。

无法分别设置加速度和减速度。

如果发生了加速度指令错误或者要缩短挑选工件时间时，请进行变更。初始值是2000[mm/sec^2]。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_Accel函数

Cnv_Accel使用示例

```
Cnv_Accel 1,2000
```

3.7.20 Cnv_Accel函数

用于返回传送带跟踪动作前的加减速度。

格式

Cnv_Accel (传送带编号)

参数

传送带编号
以表达式或数值(1～16)指定传送带的编号。

返回值

返回实值(单位：mm)。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_Accel

Cnv_Accel函数使用示例

```
Print Cnv_Accel (1)
```

3.7.21 Cnv_AccelLim

设置传送带跟踪后的加速度、减速度的设置值。

格式

Cnv_AccelLim 传送带编号, 模式编号, 加减速度

参数

传送带编号
以整数值(1～16)指定传送带的编号。
模式编号
以整数值(0～2)指定传送带跟踪的跟踪模式。
加速度和减速度
以实数值(单位：mm/sec^2)设置传送带跟踪后的加速度和减速度限制值。设置范围与AccelS命令相同。
默认值
- 挑选个数优先模式：500 [mm/sec^2]
- 挑选精度优先模式：2000 [mm/sec^2]
- 可变速传送带支持模式：6000 [mm/sec^2]

说明

如果在传送带跟踪过程中反复停止和继续运行传送带，对传送带速度变化的机器人跟踪延迟将发生。如要提高跟踪能力，则设置加速度和减速度的限制值。

无法分别设置加速度和减速度。

如果将限制值提得过高，受传送带速度不均和噪音的影响，机器人的动作将产生振动。如果将限制值降得过低，则即使停止传送带，机器人也会跟踪而不会停止,可能移动到机器人的动作区域之外。在此情况下，请设置放弃跟踪线，或通过在程序中设置，使其在下游范围停止跟踪。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_AccelLim函数

Cnv_AccelLim使用示例

```
Cnv_AccelLim 1,2,7000
```

3.7.22 Cnv_AccelLim函数

返回传送带跟踪后的加速度、减速度的限制值。

格式

Cnv_AccelLim (传送带编号, 模式编号)

参数

传送带编号
　　以整数值(1～16)指定传送带的编号。
模式编号
　　以整数值(0～2)指定传送带跟踪的跟踪模式。

返回值

返回实数值(单位：mm/sec^2)。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_AccelLim

Cnv_AccelLim函数使用示例

```
Print Cnv_AccelLim (1,2)
```

3.7.23 Cnv_Adjust

用于设置是否进行旨在获取传送带跟踪的跟踪延迟补偿值的动作。

格式

Cnv_Adjust 传送带编号, On | Off | Off

参数

传送带编号
　　以表达式或数值(1～16)指定传送带的编号。
On | Off
　　要设为获取传送带跟踪的跟踪延迟补偿值的动作时，指定“On”；设为不获取的动作时，指定“Off”。

说明

用于设置是否执行旨在获取传送带跟踪的跟踪延迟补偿值的动作。

如果在将Cnv_Adjust设为“On”的状态下执行Cnv_QueGet函数，则进行获取补偿值的动作。要拾取工件时，将Cnv_Adjust设为“Off”，然后执行Cnv_QueGet函数。

已将Cnv_Adjust设为“On”时，在获取补偿值之后，请务必设为“Off”。

Cnv_Adjust仅可用于直线传送带。圆形传送带无法使用。如果是圆形传送带，即使设为“On”，也不会获取补偿值。

如果将控制器电源设为OFF，已获取的补偿值会被清除；控制器电源ON时，设置初始值“0”。因此，获取补偿值之后，请通过print Cnv_AdjustGet返回补偿值，并在执行程序中的Cnv_QueGet之前，通过Cnv_AdjustSet设置补偿值。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_AdjustGet函数、Cnv_AdjustSet、Cnv_AdjustClear、Cnv_QueGet函数

Cnv_Adjust使用示例

```
Cnv_Adjust 1, On
Jump Cnv_QueGet(1)
 .
 .
Cnv_Adjust 1, Off
```

3.7.24 Cnv_AdjustClear

用于删除传送带跟踪的跟踪延迟补偿值。

格式

Cnv_AdjustClear 传送带编号

参数

传送带编号
以表达式或数值(1～16)指定传送带的编号。

说明

用于删除传送带跟踪的跟踪延迟补偿获取动作的结果、补偿量与补偿时间，并设为“0”。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_Adjust、Cnv_AdjustSet函数、Cnv_AdjustGet、Cnv_QueGet函数

Cnv_AdjustClear使用示例

```
Cnv_AdjustClear 1
```

3.7.25 Cnv_AdjustGet函数

用于返回传送带跟踪的跟踪延迟补偿值。

格式

Cnv_AdjustGet(传送带编号, 模式编号)

参数

传送带编号
以表达式或数值(1～16)指定传送带的编号。
模式编号

- 0：补偿获取动作的结果
- 1：补偿量
- 2：补偿时间

返回值

- 模式编号0：返回实值0～2。
  - 0：未进行补偿值的获取动作
  - 1：已获取补偿值
  - 2：获取补偿值失败
- 模式编号1：返回实数值(单位：mm)。
- 模式编号2：返回实值(单位：秒)。

说明

未使用Cnv_Adjust与Cnv_QueGet函数获取传送带跟踪的跟踪延迟补偿值时，模式编号0～2的返回值均为“0”。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_Adjust、Cnv_AdjustSet、Cnv_AdjustClear、Cnv_QueGet函数

Cnv_AdjustGet函数使用示例

```
Print Cnv_AdjustGet(1, 1)
```

3.7.26 Cnv_AdjustSet

用于设置传送带跟踪的跟踪延迟补偿值。

格式

Cnv_AdjustSet 传送带编号, 补偿量, 补偿时间

参数

传送带编号
　　以整数值(1～16)指定传送带的编号。
补偿量
　　以实值(单位：mm)指定补偿量。
补偿时间
　　以实值(单位：秒)指定补偿时间。

说明

未实施Cnv_AdjustSet的情况下，补偿量与补偿时间适用上次设置的值。

控制器电源ON之后从未执行获取补偿值的动作的情况下，补偿量与补偿时间会被设为初始值“0”。

当Cnv_Mode命令的模式编号设定为1: 挑选精度优先模式时，Cnv_AdjustSet命令的设定可用。

圆形传送带无法使用Cnv_AdjustSet。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_Adjust、Cnv_AdjustGet、Cnv_AdjustClear、Cnv_QueGet函数

Cnv_AdjustSet使用示例

```
Cnv_AdjustSet 1, 4.5, 0.1
```

3.7.27 Cnv_Downstream

设置指定传送带的下游限值。

格式

Cnv_Downstream 传送带编号, 下游限值

参数

传送带编号
　　以表达式或数值(1～16)指定传送带的编号。
下游范围
　　设置跟踪开始区域的下游侧的边界线

说明

可以变更由校准向导设置的下游限值。但是，在使用倾斜下游限值时，无法用Cnv_Downstream变更下游限值。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_Upstream

Cnv_Downstream使用示例

```
Cnv_Downstream 1,500
```

3.7.28 Cnv_Downstream函数

返回指定传送带的下游限值设置值。

格式

Cnv_Downstream (传送带编号)

参数

传送带编号
以表达式或数值(1～16)指定传送带的编号。

返回值

- 直线传送带：返回实值(单位：mm)。
- 圆形传送带：返回实值(单位：deg)。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_Upstream

Cnv_Downstream函数使用示例

```
Print "Downstream limit: ", Cnv_Downstream(1)
```

3.7.29 Cnv_Fine

用于设置和显示相对于指定传送带的传送带跟踪完成判断范围。

格式

Cnv_Fine 传送带编号 [, Fine设置值]

参数

传送带编号
以整数值(1～16)指定传送带的编号。
Fine设置值
以表示判断完成跟踪的距离的实值(单位：mm)进行指定。
如果将值设为0，则无法使用Cnv_Fine。如果省略Fine设置值，则显示当前的Cnv_Fine设置。

说明

用于判断完成传送带的跟踪，并指定到可以接收下一命令的工件的距离。如果设为“0”，将不使用Cnv_Fine，在完成动作命令后可以接收下一命令。默认值是“0”，在下一时序时，默认值将被自动设置。

- 定义传送带定义时
- 启动控制器时

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_Fine函数

Cnv_Fine使用示例

```
Cnv_Fine 1, 5
```

3.7.30 Cnv_Fine函数

用于返回指定传送带的跟踪完成判断范围的设置。

格式

Cnv_Fine (传送带编号)

参数

传送带编号
以整数值(1～16)指定传送带的编号。

返回值

以实值(单位：mm)返回Cnv_Fine设置。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_Fine

Cnv_Fine函数使用示例

```
Real f

f = Cnv_Fine(1)
```

3.7.31 Cnv_Flag函数

返回机器人的跟踪状态。

格式

Cnv_Flag (传送带编号)

参数

传送带编号
以整数值(1～16)指定传送带的编号。

返回值

- 0：已正常执行跟踪动作。(未取消或中止)
- 1：预计工件会超出跟踪终止线，因此取消执行跟踪动作。
- 2：预计工件会超出跟踪终止线，因此中止跟踪动作。Z位置未下降到指定高度。
- 3：预计工件会超出跟踪终止线，因此中止跟踪动作。Z位置下降到指定高度。
- 4：因工件在跟踪开始区域以外而取消执行跟踪动作。

仅限于设有跟踪中止线的情况下，返回“0”以外的返回值。

有关跟踪中止线的详细说明，请参阅以下手册。
《Epson RC+ 用户指南》

注意

此命令只在已安装传送带跟踪选项时才可使用。

Cnv_Flag函数使用示例

```
Print Cnv_Flag (1)
```

3.7.32 Cnv_LPulse函数

用于返回被传送带的触发锁住的脉冲。

格式

Cnv_LPulse (传送带编号)

参数

传送带编号
以整数值(1～16)指定传送带的编号。

说明

返回硬件触发的配线或被Cnv_Trigger锁住的最新传送带的脉冲。

返回值

以Long型数值返回指定的传送带的被锁住的脉冲。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_Trigger, Cnv_Pulse

Cnv_LPulse函数使用示例

```
Print "Latched conveyor position: ", Cnv_LPulse(1)
```

3.7.33 Cnv_Mode

用于设置传送带跟踪的跟踪模式。

格式
Cnv_Mode (传送带编号, 模式编号)

参数

传送带编号
以整数值(1～16)指定传送带的编号。
模式编号
- 0：挑选个数优先模式
- 1：挑选精度优先模式
- 2：可变速传送带支持模式

说明
用于设置传送带跟踪的跟踪模式。

Cnv_Mode仅在直线传送带中使用。

请在跟踪动作前设置跟踪模式。未设置时，将设置挑选个数优先模式。

- 挑选个数优先模式：虽然挑选精度比挑选精度优先模式差，但是跟踪所需时间短，适于工件间隔狭小或传送带速度快的传送带系统。
- 挑选精度优先模式：虽然跟踪所需时间比挑选个数优先模式长，但是提高了挑选精度，是适于使用小工件的传送带系统的模式。
- 可变速传送带支持模式：此模式适用于传送带与工件接触时，停止或启动的传送带系统。

圆形传送带支持的模式编号仅限于“0”。设置“1”与“2”时，机器人将进行与模式编号“0”相同的动作。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅
Cnv_Mode函数

Cnv_Mode使用示例

```
Cnv_Mode 1, 1
```

3.7.34 Cnv_Mode函数

用于返回传送带跟踪的跟踪模式。

格式

Cnv_Mode (传送带编号)

参数

传送带编号
以整数值(1～16)指定传送带的编号。

返回值

返回实值0～2。

- 0：挑选个数优先模式
- 1：挑选精度优先模式
- 2：可变速传送带支持模式

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_Mode

Cnv_Mode函数使用示例

```
Print Cnv_Mode (1)
```

3.7.35 Cnv_Name$函数

用于返回指定传送带的名称。

格式

Cnv_Name$ (传送带编号)

参数

传送带编号
以整数值(1～16)指定传送带的编号。

返回值

以字符串返回传送带名称。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_Number

Cnv_Name$函数使用示例

```
Print "Conveyor 1 Name: ", Cnv_Name$(1)
```

3.7.36 Cnv_Number函数

用于返回指定传送带的名称的传送带编号。

格式
Cnv_Number (传送带名称)

参数

传送带名称
以字符串指定传送带名称。

返回值
以整数值返回传送带编号。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅
Cnv_Name$

Cnv_Number函数使用示例

```
Integer cnvNum

cnvNum = Cnv_Number("Main Conveyor")
```

3.7.37 Cnv_OffsetAngle

用于设置传送带队列数据的偏移值。

格式

Cnv_OffsetAngle 传送带编号 [,偏移值]

参数

传送带编号
　　以整数值(1～16)指定传送带的编号。
偏移值
　　以实值(单位：degree)指定传送带队列数据的偏移值。可省略。如果省略，将显示当前偏移值。

说明

用于设置传送带队列数据的偏移值。

Cnv_OffsetAngle仅在圆形传送带中使用。

传送带跟踪根据使用的传送带速度可能会发生跟踪延迟。如果发生了跟踪延迟，机器人将按照跟踪延迟量处理偏移位置。Cnv_OffsetAngle通过赋予队列以偏移值来修正此偏移。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_OffsetAngle函数

Cnv_OffsetAngle使用示例

```
Cnv_OffsetAngle 1, 5
```

3.7.38 Cnv_OffsetAngle函数

用于返回传送带队列数据的偏移值。

格式
Cnv_OffsetAngle (传送带编号)

参数

传送带编号
以整数值(1～16)指定传送带的编号。

返回值
以实值(单位：degree)返回。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅
Cnv_OffsetAngle

Cnv_OffsetAngle函数使用示例

```
Real offsetAngle

offsetAngle = Cnv_OffsetAngle (1)
```

3.7.39 Cnv_Point函数

用于将传感器坐标值转换为传送带坐标值并返回。

格式

Cnv_Point (传送带编号, 传感器X坐标值, 传感器Y坐标值 [, 传感器U坐标值])

参数

传送带编号
　　以整数值(1～16)指定传送带的编号。
传感器X坐标值
　　以实值指定传感器X坐标值。
传感器Y坐标值
　　以实值指定传感器Y坐标值。
传感器U坐标值
　　以实值指定传感器U坐标值。可省略。

返回值

以点数据返回传送带的坐标值。

说明

Cnv_Point函数用于将作为传送带队列添加的坐标值作为点数据进行添加。在视觉传送带中，传感器X坐标值和传感器Y坐标值是摄像机的视觉坐标值。在传感器传送带中，传感器位置将变为传送带坐标系的原点。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_Speed

Cnv_Point函数使用示例

```
Boolean found
Integer i, numFound
Real x, y, u

Cnv_Trigger 1
VRun FindParts
VGet FindParts.Part.NumberFound, numFound
For i = 1 To numFound
  VGet FindParts.Part.CameraXYU(i), found, x, y, u
  Cnv_QueAdd 1, Cnv_Point(1, x, y)
Next i
```

3.7.40 Cnv_PosErr函数

用于返回当前tracking位置与目标的位置偏差。

格式

Cnv_PosErr (传送带编号)

参数

传送带编号
以整数值(1～16)指定传送带的编号。

返回值

以实值(单位：mm)返回。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_MakePoint

Cnv_PosErr函数使用示例

```
Print "Conveyor 1 position error: ", Cnv_PosErr(1)
```

3.7.41 Cnv_PosErrOffset

设置校正当前的追踪位置与目标之间的位置偏差的值。

格式

Cnv_PosErrOffset 传送带编号, 校正值

参数

传送带编号
　　以整数值(1～16)指定传送带的编号。
补偿值
　　以实数值(0～255，单位：msec)指定预测传送带速度的时间。

说明

如果在传送带跟踪过程中反复停止和运行传送带，相对于传送带速度变化的机器人跟踪延迟将加大跟踪位置与目标之间的位置偏差。

通过预测用校正值设置的时间过后的传送带速度以进行传送带跟踪，可改善位置偏差。

Cnv_PosErrOffset可在变速传送带支持模式下使用。在挑选个数优先模式和挑选精度优先模式中，无法通过设置校正值改善位置偏差。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_Mode、Cnv_PosErr

Cnv_PosErrOffset使用示例

```
Cnv_Mode 1, 2    ' 可变速传送带支持模式
Cnv_PosErrOffset 1, 10    ' 校正值10msec
```

3.7.42 Cnv_Pulse函数

用于返回传送带的当前位置的脉冲。

格式

Cnv_Pulse (传送带编号)

参数

传送带编号
以整数值(1～16)指定传送带的编号。

返回值

以Long型数值返回指定的传送带的脉冲。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_Trigger, Cnv_LPulse

Cnv_Pulse函数使用示例

```
Print "Current conveyor position: ", Cnv_Pulse(1)
```

3.7.43 Cnv_QueAdd

用于在传送带队列数据中添加点数据。

格式

Cnv_QueAdd 传送带编号, 点数据 [, 用户数据]

参数

传送带编号
　　以整数值(1～16)指定传送带的编号。
点数据
　　用于指定要在传送带队列数据中添加的点数据。
用户数据
　　以实数值指定与点数据一起注册的用户数据。可省略。

说明

点数据将添加到指定传送带队列数据的末尾。也一起收录现在闩锁的传送带和脉冲位置也一起注册现在闩锁的传送带和脉冲位置。

存在小于Cnv_QueReject指定的某一距离的队列数据时，将无法收录数据将无法注册数据，会出错。

队列数据的上限值是1000。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_RobotConveyor

Cnv_QueAdd使用示例

```
Boolean found
Integer i, numFound
Real x, y, u

Cnv_Trigger 1
VRun FindParts
VGet FindParts.Part.NumberFound, numFound
For i = 1 To numFound
  VGet FindParts.Part.CameraXYU(i), found, x, y, u
  Cnv_QueAdd 1, Cnv_Point(1, x, y)
Next i
```

3.7.44 Cnv_QueGet函数

用于从指定传送带队列数据中返回点数据。

格式

Cnv_QueGet (传送带编号 [, 索引])

参数

传送带编号
　　以整数值(1～16)指定传送带的编号。
索引
　　以整数值指定要获取的队列数据索引。(开头的索引编号是0。)可省略。

返回值

用于从指定传送带队列数据中返回点数据。

说明

Cnv_QueGet用于从传送带队列中获取点数据。如果省略索引，将返回队列数据的开头数据。如果已设置索引，将返回设置的索引的点数据。

Cnv_QueGet用于从传送带队列中删除点数据。使用Cnv_QueRemove以进行删除。

如要边开动传送带边跟踪工件，需要在动作命令中插入Cnv_QueGet。

例:

```
  Jump Cnv_QueGet(1) ' 跟踪工件
```

无法将Cnv_QueGet的返回值代入到点中，并在该点使机器人动作和跟踪工件。

```
  P1 = Cnv_QueGet(1)
  Jump P1  ' 不跟踪工件
```

如果将Cnv_QueGet的返回值代入点中，其坐标值将变为执行命令时的工件位置。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_QueLen, Cnv_QueRemove

Cnv_QueGet函数使用示例

```
' 跳到队列的开头进行跟踪
Jump Cnv_QueGet(1)
On gripper
Wait .1
Jump place
Off gripper
Wait .1
Cnv_QueRemove 1
```

3.7.45 Cnv_QueLen函数

用于返回指定传送带队列数据的数量。

格式
Cnv_QueLen (传送带编号 [, 参数编号])

参数

传送带编号
　　以整数值(1～16)指定传送带的编号。
参数编号
　　以整数值指定计算队列数据的区域。可省略。

常数	值	内容
CNV_QUELEN_ALL	0	返回队列数据的总数。
CNV_QUELEN_UPSTREAM	1	返回跟踪开始区域上游的队列数据量。
CNV_QUELEN_PICKUPAREA	2	返回跟踪开始区域内的队列数据量。
CNV_QUELEN_DOWNSTREAM	3	返回跟踪开始区域下游的队列数据量。

返回值
以整数值返回数据量。

说明
返回有效的队列数据量。特别是，在获取跟踪开始区域内的数据量时有效。

还可以作为Wait命令的自变量使用。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅
Cnv_QueGet

Cnv_QueLen函数使用示例

```
Do
  Do While Cnv_QueLen(1, CNV_QUELEN_DOWNSTREAM) > 0
    Cnv_QueRemove 1, 0
  Loop
  If Cnv_QueLen(1, CNV_QUELEN_PICKUPAREA) > 0 Then
    Jump Cnv_QueGet(1, 0) C0
    On gripper
    Wait .1
    Cnv_QueRemove 1, 0
    Jump place
    Off gripper
    Jump idlePos
  EndIf
Loop
```

3.7.46 Cnv_QueList

用于显示指定传送带队列数据一览表。

格式

Cnv_QueList 传送带编号, [显示数]

参数

传送带编号
　　以整数值(1～16)指定传送带的编号。
显示数量
　　以整数值指定显示数的数据量。可省略。如果省略，将显示所有队列数据。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_QueGet

Cnv_QueList使用示例

```
Cnv_QueList 1
```

3.7.47 Cnv_QueMove

用于使上游传送带的队列数据移至下游传送带的队列中。

格式

Cnv_QueMove 传送带编号 [, 索引] [, 用户数据]

参数

传送带编号
　　以整数值(1～16)指定传送带的编号。
索引
　　以整数值指定要移动的队列数据索引。(开头的索引编号是0。)可省略。
用户数据
　　以实值指定与数据一起注册的用户数据。可省略。

说明

将队列数据移至下游传送带的队列中。如果省略索引，将移动开头(索引为0)的队列数据。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_QueGet

Cnv_QueMove使用示例

```
Cnv_QueMove 1
```

3.7.48 Cnv_QueReject

用于设置和显示防止队列重复注册的最小距离。

格式

Cnv_QueReject 传送带编号 [,防止重复注册的距离]

参数

传送带编号
　　以整数值(1～16)指定传送带的编号。
拒绝距离
　　以实值指定防止重复注册的距离的最小值(单位：mm)。如果指定了负值，将设置为0 mm。可省略。如果省略，将显示当前防止重复注册的距离。

说明

用于设置防止队列重复注册的最小距离。通过视觉系统进行扫描，即使检测到1次以上，也只有1次可以注册。
Cnv_QueReject为防止重复注册的系统文件夹。默认为0 mm。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_QueReject函数

Cnv_QueReject使用示例

```
Cnv_QueReject 1, 20
```

3.7.49 Cnv_QueReject函数

用于返回队列的防止重复注册的距离。

格式
Cnv_QueReject (传送带编号)

参数

传送带编号
以整数值(1～16)指定传送带的编号。

返回值
以实值(单位：mm)返回。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅
Cnv_QueReject

Cnv_QueReject函数使用示例

```
Real rejectDist

RejectDist = Cnv_QueReject(1)
```

3.7.50 Cnv_QueRemove

用于从传送带队列数据中删除队列数据。

格式

Cnv_QueRemove 传送带编号 [,索引| All] | All]

参数

传送带编号
　　以整数值(1～16)指定传送带的编号。
索引
　　以整数值指定要删除的开头的索引，或者要全部删除时，以All进行指定。可省略。

说明

从传送带队列数据中删除1个以上的队列数据。在用完队列数据时，删除队列数据。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_QueAdd

Cnv_QueRemove使用示例

```
Jump Cnv_QueGet(1)
On gripper
Wait .1
Jump place
Off gripper
Wait .1

' 从传送带中删除数据
Cnv_QueRemove 1
```

3.7.51 Cnv_QueUserData

用于设置和显示与队列入口相关的用户数据。

格式

Cnv_QueUserData 传送带编号[, 索引] [, 用户数据]

参数

传送带编号
　　以整数值(1～16)指定传送带的编号。
索引
　　以整数值指定队列数据的索引。可省略。
用户数据
　　以实值指定用户数据。可省略。

说明

设置传送带队列的用户数据。参数的用户数据可以省略，但是在一般操作中需要使用。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_QueUserData函数

Cnv_QueUserData使用示例

```
Cnv_QueUserData 1, 1, angle
```

3.7.52 Cnv_QueUserData函数

用于返回与队列入口相关的用户数据。

格式

Cnv_QueUserData (传送带编号 [, 索引])

参数

传送带编号
　　以整数值(1～16)指定传送带的编号。
索引
　　以整数值指定队列数据的索引。可省略。

返回值

返回实值。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_QueUserData

Cnv_QueUserData函数使用示例

```
' 追加到队列中
Cnv_QueAdd 1, Cnv_Point(1, x, y), angle

' 从队列中删除
angle = Cnv_QueUserData(1)  ' 默认索引为“0”
Jump Cnv_QueGet(1) :U(angle)
Cnv_QueRemove 1
```

3.7.53 Cnv_RobotConveyor函数

用于返回跟踪中的传送带编号。

格式
Cnv_RobotConveyor [(机械手编号)]

参数

机械手编号
　　以整数值指定机械手编号。

返回值
以整数值返回传送带编号。如果没有要跟踪的传送带，返回0。

说明
如果正在使用多个机械手，确认机械手跟踪哪个传送带。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅
Cnv_MakePoint

Cnv_RobotConveyor函数使用示例

```
Integer cnvNum

cnvNum = Cnv_RobotConveyor(1)
```

3.7.54 Cnv_Speed函数

用于返回传送带的动作速度。

格式

Cnv_Speed (传送带编号)

参数

传送带编号
以整数值(1～16)指定传送带的编号。

返回值

以实值(单位：mm/sec)返回。在圆形传送带中，以实值(单位：degree/s)返回。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_Pulse

Cnv_Speed函数使用示例

```
Print "Conveyor speed: ", Cnv_Speed(1)
```

3.7.55 Cnv_Trigger

锁定传送带的当前位置，以便执行下面的Cnv_QueAdd语句。

格式

Cnv_Trigger 传送带编号

参数

传送带编号
以整数值(1～16)指定传送带的编号。

说明

用于在指定的传送带编码器的脉冲输出板上没有硬件触发配线的情况。Cnv_Trigger为软件触发。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_QueAdd

Cnv_Trigger使用示例

```
Boolean found
Integer i, numFound
Real x, y, u

Cnv_Trigger 1
VRun FindParts
VGet FindParts.Part.NumberFound, numFound
For i = 1 To numFound
  VGet FindParts.Part.CameraXYU(i), found, x, y, u
  Cnv_QueAdd 1, Cnv_Point(1, x, y)
Next i
```

3.7.56 Cnv_Upstream

设置指定传送带的上游限值。

格式

Cnv_Upstream 传送带编号, 上游限值

参数

传送带编号
　　以表达式或数值(1～16)指定传送带的编号。
上游范围
　　设置跟踪开始区域的上游侧的边界线。

说明

可以变更由校准向导设置的上游限值。但是，在使用倾斜上游限值时，无法用Cnv_Upstream变更上游限值。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_Downstream

Cnv_Upstream使用示例

```
Cnv_Upstream 1,200
```

3.7.57 Cnv_Upstream函数

返回指定传送带的上游限值。

格式

Cnv_Upstream (传送带编号)

参数

传送带编号
以整数值(1～16)指定传送带的编号。

返回值

- 直线传送带：返回实值(单位：mm)。
- 圆形传送带：返回实值(单位：deg)。

注意

此命令只在已安装传送带跟踪选项时才可使用。

参阅

Cnv_Downstream

Cnv_Upstream函数使用示例

```
Print "Upstream limit: ", Cnv_Upstream(1)
```

3.7.58 CollisionDetect

用于启用或退出当前机器人的碰撞检测功能(机器人动作的异常检测功能)。

格式

(1) CollisionDetect 状态

(2) CollisionDetect 状态, 关节编号

(3) CollisionDetect

参数

状态

- On：将碰撞检测(机器人动作的异常检测)设为有效。
- Off：将碰撞检测(机器人动作的异常检测)设为无效。

关节编号

SCARA机器人(包括RS系列)时，指定1～4的关节编号；垂直6轴型机器人(包括N系列)时，指定1～6的关节编号。

结果

当省略了参数时，将显示当前的CollisionDetect状态。

说明

根据机器人的预期动作速度与实际动作速度之差(速度偏差值)检测机器人动作的异常。可利用本功能检测的异常分为A与B类。

- A：机器人机械臂与末端夹具发生碰撞或接触
- B：碰撞或接触以外的机器人动作异常

此外，根据功率的状态，对B异常进行如下分类。

- 高功率状态下的异常：因Weight或Inertia设置过小而导致转矩饱和
  - 因多关节轴的复合动作或细长物体摆动而导致转矩饱和
  - 因电源电压过低而导致转矩饱和
  - 因硬件异常或软件误运作而导致异常动作
- 低功率时的异常：
  - 因硬件异常或软件误运作而导致异常动作
  - 因超出规格的夹具末端重量或保持细长物体而导致低功率时的转矩饱和

碰撞检测功能应对Epson RC+ 7.0 Ver.7.2以后版本支持的通用机器人(垂直6轴型机器人、SCARA机器人)。如果在连接X5系列等未支持的机器人的状态下使用本命令，将发生错误。

执行本命令需要处理时间。要求循环时间时，请将命令的使用控制在最低限度。

可利用命令设置所有轴的ON/OFF以及各轴的ON/OFF。默认值为所有轴打开。(固件版本是Ver7.2.1.x以后版本时为打开；Ver7.2.0.x以前版本时为关闭)

如果关闭控制器电源，将恢复为默认值，但在其他情况下，除非利用本命令明确进行设置，否则状态不会发生变化。

检测到碰撞时，将输出下述信息并停止机器人动作。

- 错误5057“在高功率状态下检测到碰撞(机器人动作的异常检测)”
- 错误5058“在低功率状态下检测到碰撞(机器人动作的异常检测)”

如要降低高功率模式时的损坏程度，可并用基于LimitTorque命令的转矩限制功能；如要降低低功率模式时的损坏程度，可并用基于LimitTorqueLP命令的转矩限制功能。

还请参阅以下的说明。
《Epson RC+ 用户指南 - 碰撞检测功能(机器人动作的异常检测功能)》

参阅

LimitTorque、LimitTorque函数、LimitTorqueLP、LimitTorqueLP函数

CollisionDetect使用示例

```
CollisionDetect On          ' 将所有轴碰撞检测设为ON
CollisionDetect Off, 5      ' 仅将第5关节设为OFF
CollisionDetect             ' 显示on, on, on, on, off, on。
```

3.7.59 CollisionDetect函数

用于返回CollisionDetect命令的设置值。

格式

CollisionDetect (关节编号)

参数

关节编号
以1～6的整数进行指定。

返回值

以整数值返回CollsionDetect命令的设置值。

- 1 = ON
- 0 = OFF

参阅

CollisionDetect

CollisionDetect函数使用示例

```
Print CollisionDetect(1)  '显示第1关节的CollisionDetect值
```

3.7.60 Cont

在重新启动变为暂停状态的控制器、继续执行所有任务时使用。

本命令用于高级人员。请在充分理解命令规格之后使用。

格式
Cont

说明
要通过程序执行本命令时，需要勾选 Epson RC+ 的［设置］-［系统配置］-［设置控制器］-［环境］的［将高级任务控制命令设为有效］复选框。但是，即使进行此设置，也无法通过由Trap SGClose启动的任务执行Cont命令。

Cont命令在重新启动因执行Pause语句和安全门输入“开”而变为暂停状态的控制器，并继续执行所有任务之时使用。拥有与[操作员窗口]的[继续执行]按钮和远程输入的“Continue”相同的功能。

在WaitRecover状态(打开安全门后的等待恢复状态)下执行Cont命令时，在所有机器人都进行了励磁和恢复动作之后，重新开始执行程序。

如果只想进行机器人的励磁和恢复动作，请使用Recover命令。

> ⚠ 注意
> **要通过程序执行Cont命令时，请理解命令的规格并确认可作为系统继续执行的条件已经备齐齐备。如果一直采取按循环执行命令等错误的使用方法，则可能会降低系统的安全性。请充分注意。**

参阅
Pause、Recover

Cont使用示例

```
Function main
    Xqt 2, monitor, NoPause
    Do
        Jump P1
        Jump P2
    Loop
Fend

Function monitor
    Do
        If Sw(pswitch) = On then
            Pause
            Wait Sw(pswitch) = Off and Sw(cswitch) = On
            Cont
        EndIf
    Loop
Fend
```

3.7.61 Copy

复制文件。

格式

Copy 复制源, 复制目标

参数

复制源
　　指定要复制的文件的路径和文件名。详情请参阅ChDisk。
复制目标
　　指定复制源文件的目标路径与要复制的文件名。详情请参阅ChDisk。

说明

将指定的源文件复制到指定的复制目标文件名中。

复制目标与复制源不能共有相同路径名或文件名。

如果已存在复制目标，将会出错。

注意

可使用网络路径。

文件名中不能使用“*”、“?”等通配符。

在命令窗口中使用时，可以省略引号和逗号。

参阅

ChDir、MkDir

Copy使用示例

利用命令窗口的操作示例

```
> copy TEST.DAT TEST2.DAT


> Copy TEST.DAT c:     'NG
!!错误：7203 访问被拒绝。
> Copy TEST.DAT c:\      'OK
>
```

3.7.62 Cos函数

是返回指定角度的余弦的函数。

格式
Cos (数值)

参数

数值
以数值(弧度)指定角度。

返回值
用于返回已指定数值的余弦值(实值)。

说明
Cos用于返回已指定角度(弧度)的余弦。请用弧度指定要指定的角度(数值)。返回值的范围为-1～1。

以度赋予角度时，必须使用DegToRad函数变换为弧度。

参阅
Abs, Atan, Atan2, Int, Mod, Not, Sgn, Sin, Sqr, Str$, Tan, Val

Cos函数使用示例
下例为使用Cos的简单程序例。

```
Function costest
  Real x
  Print "Please enter a value in radians:"
  Input x
  Print "COS of ", x, " is ", Cos(x)
Fend
```

下例为从命令窗口使用Cos的示例。

```
Display the cosine of 0.55:

>print cos(0.55)
 0.852524522059506
>

Display cosine of 30 degrees:

>print cos(DegToRad(30))
 0.866025403784439
>
```

3.7.63 CP

设置路径运动。

格式
(1) CP { On | Off }
(2) 动作命令目标坐标 CP

参数

On | Off

- On：将路径运动设为有效。
- Off：解除路径运动。

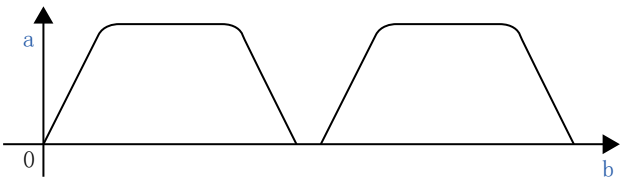
说明
通过下述命令使用路径运动。

Arc, Arc3, Go, Jump, Jump3, Jump3CP, JumpTLZ, Move

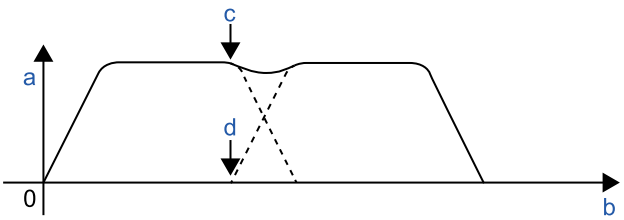
执行CP On，则动作命令将与减速开始命令同时执行以下语句，而与是否在各动作命令中指定了CP参数无关。其结果，如果在减速动作过程中开始以下动作，将合成动作轨迹。

如果执行CP Off，则仅在各动作命令中指定了CP参数时，此功能才有效。

通常动作



路径运动



符号	说明
a	速度
b	时间
c	开始减速
d	开始加速

CP动作(Arc、Arc3、Jump3、Jump3CP、JumpTLZ、Move)或者PTP动作(Go、Jump)将通过CP On合成动作轨迹。
但是，如果CP动作与PTP动作是连续轨迹，则只减速而不合成动作轨迹。
根据控制器的环境设置，在CP On时连接CP动作和PTP动作时，将合成CP动作和PTP动作的动作轨迹。根据动作加减速度的设置，可能会出现超速错误或过加速度错误。在这种情况下，请调整加减速度或将其设为无效。
当垂直6轴机器人(包括N系列)的手腕奇点作为目标值进行CP动作时，会在下一个动作和动作轨迹没有结合而减速。

如果在路径运动有效时使用Here，则在指定Here的动作之前不会合成运动轨迹，而是减速。如果采用如下的使用方法，则在P1处减速停止后开始下一个动作。

```
Go P1 CP
Go Here+X(100)
```

如果要不停止的情况下执行动作，则请修改程序以预先计算Here的位置。

```
Go P1 CP
Go P1+X(100)
```

下述情况时为CP Off。

- 启动控制器时
- 执行Motor On
- 执行SFree、SLock、Brake
- 执行Reset、Reset Error
- 利用停止按钮或执行Quit All等结束任务

参阅

CP函数、Arc、Arc3、Go、Jump、Jump3、Jump3CP、JumpTLZ、Move

CP使用示例

```
CP On
Move P1
Move P2
CP Off
```

3.7.64 CP函数

返回路径运动的状态。

格式

CP

返回值

- 0 = 关闭路径运动
- 1 = 打开路径运动

参阅

CP

CP函数使用示例

```
If CP = Off Then
    Print "CP is off"
EndIf
```

3.7.65 CP_Offset

用于在CP On时设置开始后续动作命令的偏移时间。

格式

(1) CP_Offset [On [, OffsetTime]]

(2) CP_Offset Off

参数

On | Off

- On：将CP On时的动作开始偏移功能设为有效。省略时，将显示当前的设置。
- Off：将CP On时的动作开始偏移功能设为无效。

OffsetTime

用于以10～24(单位：ms)范围内的实值设置CP On时的动作开始偏移时间。省略时，默认值将被设置为10 ms。

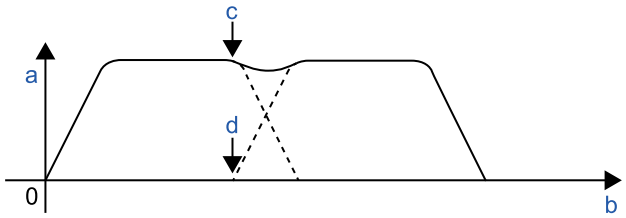
说明

CP_Offset以下述动作命令为对象。

Move、Arc、Arc3、CVMove

如果在CP On或动作命令中附加CP参数，将在开始前一动作减速的同时执行下一语句。结果如下图所示，变为减速区间与下一动作加速区间重叠的路径运动。此时，开始前一动作的减速与开始下一动作的加速之间存在语句开始处理所需的开销时间，因此并非严格一致。因此，在路径运动衔接处附近，速度将降低，不会形成等速轨道。为了改善这种现象，可使用CP_Offset功能提前后续动作语句的开始时间。

路径运动



符号	说明
a	速度
b	时间
c	开始减速
d	开始加速

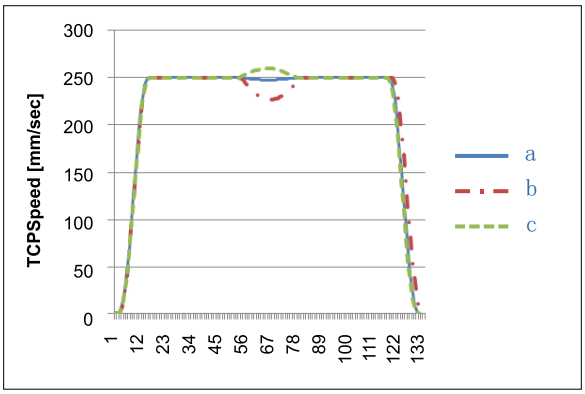
如果将CP_Offset设为ON，后续动作命令将按OffsetTime参数中设置的时间提前开始处理，实际的机器人的减速开始与下一动作的加速开始达到同步，路径运动的等速性能因此得以改善。虽然OffsetTime参数中已设置默认值，但请利用应用程序进行微调。尤其是后续的动作命令包括“!并行处理!”时，开始动作所需的开销时间将延长，因此，请将OffsetTime设为大于默认值的值(16ms左右)。

如要调整CP_Offset的设置时间(OffsetTime)，请在执行要调整的动作期间使用TCPSpeed观测工具中心点速度。如果设置适当的OffsetTime，动作衔接处的速度将接近恒定值。如果OffsetTime过大，TCPSpeed将上升；如果OffsetTime过小，TCPSpeed将降低。请在实机环境中对CP_Offset进行调整。实际的控制器与模拟器的动作开始处理时间并不相同，因此无法进行适当的调整。

TCPSpeed测量程序示例

```
Function main
  Motor On
  Power High

  SpeedS 250; AccelS 1500
  Speed 50; Accel 50, 50

  Go XY (300, 500, 500, 90, 0, 180)

  CP_Offset On
  Xqt printTcPSpeed

  Move XY (0, 500, 500, 90, 0, 180) CP
  Move XY (-300, 500, 500, 90, 0, 180)

  Quit printTcPSpeed
  CP_Offset Off
Fend

Function printTcPSpeed
  Do
    Print TCPSpeed
  Loop
Fend
```

OffsetTime调整示例



符号	说明
a	适当的OffsetTime
b	OffsetTime=0
c	OffsetTime过大

本命令不以PTP动作为对象。PTP动作时，为通常所述的路径运动。

下述情况时，CP_Offset会变为关闭状态。

- 控制器启动时
- 执行Motor On
- 执行SFree、SLock、Brake
- 执行Reset、Reset Error
- 因停止按钮操作、执行Quit All等而结束任务

参阅

CP_Offset函数、CP、Move、Arc、Arc3、CVMove

CP_Offset使用示例

```
CP_Offset On
Move P1
Move P2
CP_Offset Off
```

3.7.66 CP_Offset函数

CP On时，用于返回开始后续动作命令的偏移时间。

格式

CP_Offset

返回值

是表示动作开始偏移时间的实数

参阅

CP_Offset

CP_Offset函数使用示例

```
If CP_Offset = 0 Then
    Print "CP_Offset is off"
EndIf
```

3.7.67 Ctr函数

是返回计数器的计数值的函数。

格式

Ctr (输入位编号)

参数

输入位编号

是计数器中定义的输入位的编号。可以同时有效的计数器的最大数为16个。

返回值

计数器的当前计数值(0～65535的整数)

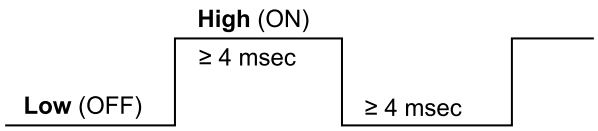
说明

Ctr通过与CTReset语句一起使用，可以设置为将I/O输入作为计数使用。

计数器设置的I/O输入每次从OFF变为ON时，其输入都会在计数器上的计数上增加1。

Ctr函数用于获取指定计数器输入的当前计数器值。无论是哪个I/O输入，都可以在计数器中进行指定。但是，可以同时有效的计数器的个数为16个。

计数器脉冲输入的时序图



参阅

CTReset

Ctr函数使用示例

下例为获取I/O输入的计数器值的程序例。

```
CTReset 3 '将输入3的计数器复位为0
On 0      '将输出开关设为ON
Wait Ctr(3) >= 5
Off 0     '在输入3的输入循环变为5时，关闭开关(关闭输出0)
```

3.7.68 CTReset

复位已指定的输入计数器的值。并且，将输入设置为计数器输入。

格式

CTReset (输入位编号)

参数

输入位编号
是作为计数器设置的输入位的编号。以整数输入有效的输入位编号。可以使用的计数器的个数最大是16。

说明

CTReset可以与Ctr函数一起使用，可以将输入作为计数器使用。CTReset用于将指定的输入位设为计数器，并启动该计数器。如果在计数器中设置了指定的输入，将进行复位并重新启动。

注意

- 关闭电源时的计数器

  如果关闭电源，所有计数器值都将被解除。

- 使用Ctr函数时

  可以使用Ctr函数了解当前I/O输入的计数器值。

参阅

Ctr

CTReset使用示例

下例为获取I/O输入的计数器值的程序例。

```
CTReset 3   '将输入3的计数器复位为0
On 0        '将输出开关设为ON
Wait Ctr(3) >= 5
Off 0       '在输入3的输入循环变为5时，关闭开关(关闭输出0)
```

3.7.69 CtrlDev函数

用于返回当前的控制装置的编号。

格式

CtrlDev

返回值

- 21: PC
- 22：远程I/O
- 26：远程以太网
- 29：远程RS232C
- 20: TP3

参阅

CtrlInfo函数

CtrlDev函数使用示例

```
Print "The current control device is: ", CtrlDev
```

3.7.70 CtrlInfo函数

用于返回控制器的信息。

格式

CtrlInfo (索引)

参数

索引

以整数值指定要检索的信息索引。

说明

可以通过CtrlInfo函数获取的信息如下表所示。

索引	位	值	说明
0	N/A		为了与原版本的兼容性而保留。 如要获取控制器的固件版本，请使用索引9。
1	控制器状态		
	0	&H1	Ready状态
	1	&H2	Start状态
	2	&H4	Pause状态
	3-7		未定义
	8	&H100	紧急停止状态
	9	&H200	安全门打开状态
	10	&H400	错误状态
	11	&H800	严重错误状态
	12	&H1000	警告状态
	13	&H2000	WaitRecover状态(打开安全门后的等待恢复状态)
	14	&H4000	Recover 状态(打开安全门后的恢复动作期间)
	15～31		未定义
2	0	&H1	TP1的Enable开关ON
	1～31		未定义
3	0	&H1	TEACH模式电路异常检测
	1	&H2	安全门电路异常检测
	2	&H4	紧急停止电路异常检测
	3～31		未定义
4	N/A		■ 0 - 实际运行模式 ■ 1 - 空运行模式

索引	位	值	说明
5	N/A		控制装置： ■ 21 - RC+ ■ 22 - 远程 ■ 26 - 远程以太网 ■ 29 - 远程RS232C ■ 20 - TP3
6	N/A		设置的机器人的台数
7	N/A		操作模式： ■ 0 - Programing模式 ■ 1 - AUTO模式
8	N/A		未定义
9	N/A		控制器的固件版本 主版本编号*1000000 + 次版本编号*10000 + 修订版编号*100 + 内部版本号 例如，1.6.2.4时：1060204
10	N/A		硬盘的SMART状态 ■ 0：SMART状态正常 ■ 1：SMART状态异常 如果SMART状态异常，可能是硬盘发生故障，请立即备份数据并更换新硬盘。 使用RAID选件时，不能使用SMART状态。始终返回正常通常返回正常。
15	N/A		DC电源电压值 可以获得已输入电压100倍的值。 例如，48.01V则是4801。 如是不支持DC电源的机型则会报错。
16	N/A		PLC厂商类型 ■ 0: None ■ 1: Allen Bradley ■ 2: CODESYS

返回值

用于返回所需的Long整数表达式的值。

参阅

RobotInfo、TaskInfo

CtrlInfo函数使用示例

```
Print "The number of robots is: ", CtrlInfo(6)
```

3.7.71 CurDir$函数

用于以字符串返回当前目录。

格式
CurDir$

返回值
返回盘符的字符串和路径名。

参阅
ChDir, CurDrive$, CurDisk$

CurDir$函数使用示例

```
Print "The current directory is: ", CurDir$
```

3.7.72 CurDisk$函数

用于以字符串返回当前磁盘名。

格式
CurDisk$

返回值
返回磁盘名的字符串。

参阅
ChDisk, CurDir$, CurDrive$

CurDisk$函数使用示例

```
Print "The current disk is: ", CurDisk$
```

3.7.73 CurDrive$函数

用于以字符串返回当前盘符。

格式
CurDrive$

返回值
返回盘符的字符串。

参阅
ChDrive, CurDir$, CurDisk$

CurDrive$函数使用示例

```
Print "The current drive is: ", CurDrive$
```

3.7.74 CurPos函数

用于返回机器人的当前的动作目标位置。

格式
CurPos

返回值
用于返回机器人的当前的动作目标位置。

参阅
InPos、FindPos、RealPos

CurPos函数使用示例

```
Function main

   Xqt showPosition
   Do
      Jump P0
      Jump P1
   Loop
Fend

Function showPosition

   Do
      P99 = CurPos
      Print CX(P99), CY(P99)
   Loop
Fend
```

3.7.75 Curve

为通过自由曲线进行CP控制而创建数据和点。定义多个点数据，正确设置路径。

格式

Curve 文件名, 打开/关闭动作曲线, 模式指定, 坐标轴数, 连续点数据指定

参数

文件名

以字符串指定保存点数据的文件名。扩展名固定为“CVT”。省略了扩展名时，将自动添加.CVT的扩展名。如果执行Curve命令，将创建文件。不能指定路径。另外，也不受ChDisk等的影响。详情请参阅ChDisk。

打开/关闭动作曲线

在曲线动作结束时，指定打开/关闭动作曲线。此参数指定以下几个值。

- C -要生成的曲线是闭曲线
- O -要生成的曲线是开曲线

如果指定开曲线，Curve命令将在连续点数据的最后的点上停止机械臂。如果指定为闭曲线，Curve命令即使通过最后的点也继续动作，并使机械臂返回连续点数据的起点后停止动作。

模式指定

指定是否进行姿势修正(机械臂是否自动向U轴的切线方向内插)。可以使用高可以使用前面4位指定ECP编号。

模式指定		姿势修正	ECP编号
16进制数	10进制数		
&H00	0	不实施	0
&H10	16		1
&H20	32		2
…	…		…
&HA0	160		10
&HB0	176		11
&HC0	192		12
&HD0	208		13
&HE0	224		14
&HF0	240		15
&H02	2	实施	0
&H12	18		1
&H22	34		2
…	…		…
&HA2	162		10
&HB2	178		11
&HC2	194		12
&HD2	210		13
&HE2	226		14
&HF2	242		15

指定姿势修正时，机械臂仅使用连续点数据起点的U轴坐标。姿势修正将维持姿势轴在XY平面始终与曲线相切的姿势。在像切刀那样需要继续向切线方向实施控制的工具时进行指定。在向U轴的切线方向指定圆弧自动内插的闭曲线时，U轴将从起点开始旋转360度。因此，在执行CVMove命令之前，为防止U轴的旋转导致的错误，请通过Range命令指定U轴的动作范围。
如果使用ECP，请在前4位上指定ECP编号。
考虑到包括点数据的附加轴的位置，在生成自由曲线时，请将第9位指定为“1”。例如，不使用姿势修正和ECP，在生成考虑了附加轴位置的自由曲线时，指定“&H100”。
如果通过附加轴生成自由曲线，S关节和T关节将各自独立与连续点数据关联，而与机器人坐标系无关。
但是，附加轴由PG轴构成时，不能通过连续点生成自由曲线，将在最后的点上生成动作数据。

坐标轴数

以2、3、4、6的整数指定在曲线动作中控制的坐标轴数。

- 2 -在不包括姿势的XY平面上生成自由曲线。(垂直6轴型(包括N系列)以外)
- 3 -在不包括姿势的XYZ平面上生成自由曲线。(垂直6轴型(包括N系列)以外)
- 4 -在包括姿势的XYZ空间上生成自由曲线。(垂直6轴型(包括N系列)以外)
- 6 -在包括姿势的XYZ空间上生成自由曲线。(仅限垂直6轴型(包括N系列))

未在控制对象中选择的轴将保持上次编码器脉冲位置，并且在Curve动作过程中不进行动作。

连续点数据指定 { 点数据|点编号(开头:结尾)} [, 输出命令] ...

用逗号(,)分隔指定此参数各自的点数据。点数据没有遗漏并按升序或降序排列时，可用冒号连接2个点编号进行指定，比如P(1:5)。
与动作同步在中途打开和关闭I/O的输出端口时，可以用逗号(,)分隔并记述输出命令。
连续点数据一般像下面那样用逗号分隔指定。

```
Curve "MyFile", O, 0, 4, P1, P2, P3, P4
```

或者，像下面那样使用冒号指定。

```
Curve "MyFile", O, 0, 4, P(1:4)
```

在上例中，使用P1、P2、P3、P4指定曲线。输出命令可以省略，在曲线动作中控制输出操作时使用。此命令用于指定I/O或存储器I/O的ON/OFF。输出命令将在机械臂通过之前的连续点数据的特定点后执行。在1个Curve语句中可以包含的输出命令数最多为16个。在下例中，在机械臂通过P2后将执行“On2”命令，此后机械臂将通过P3～P10的所有点。

```
Curve "MyFile", C, 0, 4, P1, P2, ON 2, P(3:10)
```

说明

此命令将创建一个文件，以根据指定的点数据，使机器人机械臂进行自由曲线CP动作，并将其数据保存到控制器的文件中。根据此命令创建的数据将在根据CVMove命令执行CP动作时使用。

曲线文件被保存在控制器内的小型闪存卡中。如果执行Curve，则会向小型闪存卡执行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议只在需要时执行建议只在必要情况下执行Curve。

Curve命令使用三维花键函数独立计算指定各点的X、Y、Z、U、V、W坐标值，并据此生成轨迹。因此，如果点间的间隔过大或姿势变化大，生成的轨迹将很难估计。

动作时的速度和加减速度不需要在Curve命令前指定。在执行CVMove前，可以使用SpeedS和AccelS等命令进行变更。

如果在Curve命令参数的点群中使用在本地坐标系中设置的点，可以设置通过此位置的曲线。如果通过指定的点数据使其拥有本地属性，继Curve命令之后，可以变更Local语句的本地坐标系上的点。

注意

- 请尽量实施姿势修正

  建议尽量实施姿势修正。特别是使用CVMove使相同点群连续循环时，请实施姿势修正。如果不实施姿势修正，特别是以高速使机器人动作时，可能无法保持正确的位置。

- 开曲线上的点群的点数范围

请在开曲线上指定3～1000个点。

- 闭曲线上的点群的点数范围

  RC700、RC90系列的控制器，请对闭曲线指定3～1000点。

  T/VT，请对闭曲线指定3～300点。(如果在模拟器上指定了T/VT，则最多可以工作到1000点，但在T/VT的实际机器环境中，最多可以工作到300点。)

- 点数多时处理时间将变长

  如果以最大点数执行Curve命令，则开曲线需要几秒钟，而闭曲线则需要几十秒钟。

  尤其是闭曲线需要较长的处理时间，因此，建议在点数较多时使用开曲线。

  在Curve使用示例2中记载使用开曲线创建接近闭曲线的轨迹的使用示例。

  但是，如果用Curve命令生成自由曲线文件，并对同一文件进行多次CVMove动作，则仅在执行Curve命令的这1次需花费上述的时间。

- 文件兼容性

  使用Ver.7.5.1或更高版本固件创建的文件，不能用于较早版本的固件。但是，使用固件Ver.7.5.1 或更早版本创建的文件可以用于固件Ver.7.5.1或更高版本。

易引起的错误

- 想要使机械臂在移动范围外动作时

  Curve命令无法在设置的曲线动作范围外进行检查。这意味着在机器人机械臂的动作过程中，以后设置的曲线轨迹可能会偏出移动范围外。这种情况下，将显示“动作范围外”错误。

- 点的间隔不均匀时

  如果点间隔不均匀，则生成曲线的加速度可能会异常高。在这种情况下，可能会出现异常加速度错误。
  将以下设置设为On，则可正常动作。
  垂直6轴型机器人(包括N系列)、RS系列：AvoidSingularity SING_VSD
  水平多关节机器人(不包括RS系列)：VSD

参阅

AccelS函数、Arc、CVMove、ECP、Move、SpeedS

Curve使用示例1

在下例中，使用名为MYCURVE.CVT的自由曲线文件，跟踪通过P1～P7的曲线，在此期间通过P2打开输出端口并通过P7使机械臂减速。

设置自由曲线

```
> curve "mycurve", O, 0, 4, P1, P2, On 2, P(3:7)
```

以直线使机械臂向P1移动

```
> jump P1
```

以定义的自由曲线“mycurve”移动机械臂

```
> cvmove "mycurve"
```

Curve使用示例2

以下示例表示(1)开曲线、(2)闭曲线、(3)以开曲线执行接近闭曲线的动作。

示教点如下所示。

```
P0 = XY(0, 300, -50, 0)'起点、终点
P1 = XY(300, 200, -50, 0)
P2 = XY(300, 400, -50, 0)
P3 = XY(-300, 400, -50, 0)
P4 = XY(-300, 200, -50, 0)
P10 = XY(10, 299.7, -50, 0)    '起点的后一个点
P11 = XY(-10, 299.7, -50, 0)   '终点的前一个点
```

(1) 开曲线

设置开曲线的自由曲线

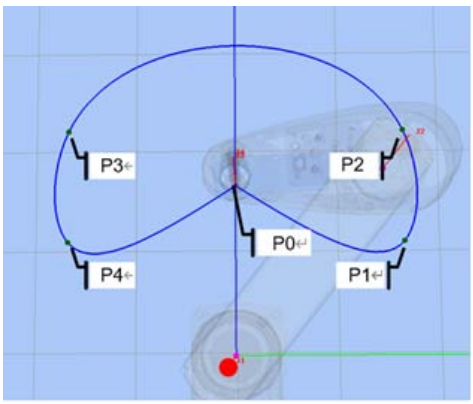
```
> Curve "mycurve_O", O, 0, 2, P(0:4), P0
```

使手臂沿直线向P0移动

```
> jump P0
```

使手臂沿定义的开曲线的自由曲线“mycurve_O”移动

```
> CVMove "mycurve_O"
```



由于是开曲线，起点和终点相同，但无法平滑顺畅地连接。

(2) 闭曲线

设置闭曲线的自由曲线

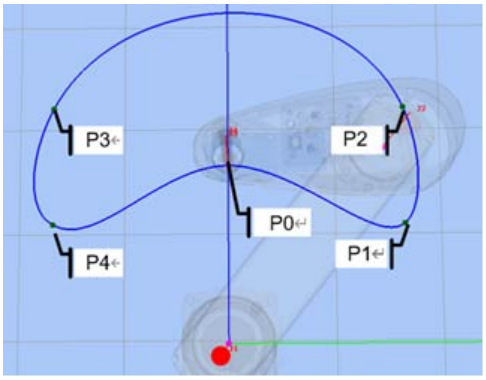
```
> Curve "mycurve_C", O, 0, 2, P(0:4)
```

使手臂沿直线向P0移动

```
> jump P0
```

使手臂沿定义的闭曲线的自由曲线“mycurve_C”移动

```
> CVMove "mycurve_C"
```



由于是闭曲线，起点与终点平滑连接。

(3) 以开曲线执行接近闭曲线的动作

设置开曲线的自由曲线。设置起点的后一个点和终点的前一个点。

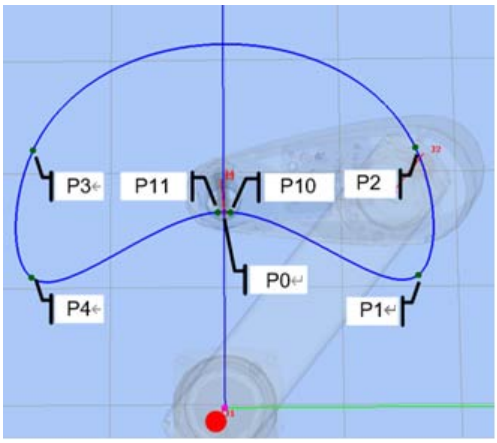
```
> Curve "mycurve_O_mod", O, 0, 2, P0, P10, P(1:4), P11, P0
```

使手臂沿直线向P0移动

```
> jump P0
```

使手臂沿定义的开曲线的自由曲线“mycurve_O_mod”移动

```
> CVMove "mycurve_O_mod"
```



虽然是开曲线，但通过经过P10、P11，使起点和终点平滑顺畅地连接。

3.7.76 CVMove

用于执行Curve命令定义的自由曲线CP动作。

格式

CVMove 文件名 [CP] [Till | Find] [SYNC]

参数

文件名
以字符串表达式或直接以字符串指定由Curve命令创建的保存在控制器中的文件名。无法指定路径。另外，不受ChDisk等的影响。详情请参阅ChDisk。

CP
指定最后的点后面的路径运动。可省略。

Till | Find
记述Till或Find表达式。可省略。

```
Till | Find
Till Sw(表达式) = {On | Off}
Find Sw(表达式) = {On | Off}
```

SYNC
预约动作命令。在通过SyncRobots开始动作之前，机器人不进行动作。

说明

CVMove用于执行设置文件数据的自由曲线的CP动作。此文件必须预先被Curve命令所创建。

如果文件名中无扩展名，将自动附加“.cvt”。

基于CVMove的CP动作的速度和加减速度，可以使用SpeedS和AccelS命令进行变更。

Curve命令可以在使用以前Local定义的点执行动作时，通过Local命令变更位置。

执行CVMove时，请充分注意与周围设备有无干扰。特别是垂直6轴型机器人(包括N系列)，如果在相邻2点之间使姿势急剧变化，因为三维花键函数的性质，其前后的点开始改变姿势并且可能会出现预想不到的动作。执行CVMove时，请注意与周围装置的干扰并进行充分的轨迹确认。

点的指定尽量以等间隔详细指定，请不要使相邻2点间的夹具末端(手腕系)姿势发生急剧变化。
如果在CVMove执行中出现异常加速度错误，将以下设置设为On，则可正常动作。
垂直6轴型机器人(包括N系列)、RS系列：AvoidSingularity SING_VSD
水平多关节机器人(不包括RS系列)：VSD

如果附加了CP参数，则可在开始动作减速时叠加后续动作命令的加速。此时，不对目标坐标进行定位。

参阅

AccelS函数、Arc、Curve、Move、SpeedS、Till、TillOn

CVMove使用示例

在下例中，使用叫做MYCURVE.CVT的自由曲线文件，跟踪通过P1～P7的曲线，通过P2打开输出端口并通过P7使机械臂减速。

设置自由曲线

```
> curve "mycurve", O, 0, 4, P1, P2, On 2, P(3:7)
```

以直线使机械臂向P1移动

```
> jump P1
```

以定义的自由曲线“mycurve”移动机械臂

```
> cvmove "mycurve"
```

3.7.77 CX、CY、CZ、CU、CV、CW、CR、CS、CT

设置点数据的坐标值。CV、CW仅可用于垂直6轴型机器人(包括N系列)。

CR仅可用于Joint型机器人。

CS、CT仅可用于设置了附加轴的机器人。

格式

CX (坐标) = 值
CY (坐标) = 值
CZ (坐标) = 值
CU (坐标) = 值
CV (坐标) = 值
CW (坐标) = 值
CR (坐标) = 值
CS (坐标) = 值
CT (坐标) = 值

参数

坐标
以P编号、P(表达式)或点标签进行指定。

值
以实值指定要设置的坐标值。

参阅

CX、CY、CZ、CU、CV、CW、CR、CS、CT函数

CX、CY、CZ、CU、CV、CW、CR、CS、CT使用示例

```
CX(pick) = 25.34
```

3.7.78 CX、CY、CZ、CU、CV、CW、CR、CS、CT函数

获取点数据的坐标值。

CV、CW函数仅可用于垂直6轴型机器人(包括N系列)。

CS、CT函数仅可用于设置了附加轴的机器人。

格式

CX (坐标)
CY (坐标)
CZ (坐标)
CU (坐标)
CV (坐标)
CW (坐标)
CR (坐标)
CS (坐标)
CT (坐标)

参数

坐标
　　指定点数据。

返回值

返回指定的坐标值。

- CX、CY和CZ函数的返回值：实值(单位：mm)。
- CU、CV和CW函数的返回值：实值(单位：deg)。
- CS、CT的各函数的返回值：是实值(单位：mm)、或实值(单位：deg)。根据附加轴的设置而不同。

说明

获取指定的点数据的坐标值。

如要获取机器人的当前位置的坐标值，可以在参数中使用Here。

参阅

CX、CY、CZ、CU、CV、CW、CR、CS、CT

CX、CY、CZ、CU、CV、CW、CR、CS、CT函数使用示例

在下例中，从“pick”提取X坐标值并设置到变量x中。

```
Function cxtest
  Real x
  x = CX(pick)
  Print "The X Axis Coordinate of point 'pick' is", x
Fend
```

3.8 D

3.8.1 Date

进行日期显示。

格式
Date

结果
显示当前的日期。

参阅
Time、Date$

Date使用示例
利用命令窗口的执行示例

```
> Date
2009/08/01
```

3.8.2 Date$函数

返回当前的日期。

格式
Date$

返回值
以字符串返回日期。

格式为yyyy/mm/dd [年/月/日]。

参阅
Date、Time、Time$

Date$使用示例

```
Print "Today's date: ", Date$
```

3.8.3 Declare

用于调用DLL(动态链接库)定义的外部函数。

格式

Declare 函数名, "DLL文件路径", "DLL内函数名" [, (自变量列表)] As函数类型

参数

函数名称
　　指定从程序中调用时的函数名。
DLL文件路径
　　以引号("")括住的字符串或由#define定义的宏指定库文件的路径和名称。
　　如果未指定路径，RC+将检索当前项目目录中的文件。如果未找到，将假定在Windows system32目录中。可以省略扩展文件名，省略时将假定为.DLL。
DLL内函数名
　　是可选参数。指定DLL中的实际函数名或函数索引。名称区分大写和小写。以被引号("")括住的字符串指定函数名。如要使用索引，在索引前附加#。如果省略，可将由“函数名”参数指定的函数名作为DLL内函数名使用。
自变量列表
　　是DLL自变量的列表。自变量请使用下述格式。可省略。

　　[{ByRef | ByVal}] 变量名[( )] As变量类型

　　ByRef
　　　　参照要调用的函数的变量时，指定ByRef。此时，可以将函数内的自变量的变更反映到调用的变量中。可以变更由参照赋予的值。可省略。
　　ByVal
　　　　是默认设置。由值(ByVal)赋予参数。由于只赋予值，在返回函数时，无法改变该变量。是调用的函数，在不变更变量的值时进行指定。可省略。
　　变量名
　　　　是必要的参数。是表示自变量的变量名，按照变量命名规则进行命名。如果作为自变量使用数组变量，请务必指定ByRef。
　　变量类型
　　　　是必要的参数。声明自变量的类型。

函数类型
　　是必要的参数。请声明函数类型。

说明

在当前程序调用DLL函数时使用。请在函数外使用Declare。

Declare语句用于确认在编译时是否存在DLL文件和函数。

以ByVal赋予数值变量

```
SPEL: Declare MyDLLFunc, "mystuff.dll", "MyDLLFunc", (a As Long) As Long
VC++ long _stdcall MyDllFunc(long a);
```

以ByVal赋予字符串变量

```
SPEL: Declare MyDLLFunc, "mystuff.dll", "MyDLLFunc", (a$ As String) As Long
VC++ long _stdcall MyDllFunc(char *a);
```

以ByRef赋予数值变量

```
SPEL: Declare MyDLLFunc, "mystuff.dll", "MyDLLFunc", (ByRef a As Long) As Long
VC++ long _stdcall MyDllFunc(long *a);
```

以ByRef赋予字符串变量

```
SPEL: Declare MyDLLFunc, "mystuff.dll", "MyDLLFunc", (ByRef a$ As String) As Long
VC++ long _stdcall MyDllFunc(char *a);
```

如果以ByRef赋予字符串，可以在DLL中变更字符串。字符串最多使用255个字符。请注意不要超过最多字符数。
SPEL+为字符串变量确保内部固定255个字符区域。

以ByRef赋予数值数组

```
SPEL: Declare MyDLLFunc, "mystuff.dll", "MyDLLFunc", (ByRef a() As Long) As Long
VC++ long _stdcall MyDllFunc(long *a);
```

DLL函数的返回值
DLL函数可以返回除了字符串(String型)以外的任何数据类型。

如需返回字符串，请参考上述“以ByRef赋予字符串变量”的内容，将字符串变量指定为参数。

如果将字符串设置为返回值，则会出现错误3614:“You cannot specify a String for Declare return data type.”。

变量类型
以下为Epson RC+ 8.0的数据类型和C/C++的变量对照表。

由于Epson RC+ 8.0没有相应的数据，C/C++的byte类型和结构无法使用。

Epson RC+ 8.0和C/C++数据类型对照表

Epson RC+ 8.0	C/C++
Boolean	short
Byte	short
Short	short
Integer	short
Long	int
Real	float
Double	double
String	char [256] * 包括Null

程序例

```
Declare ReturnLong, "mystuff.dll", "ReturnLong", As Long

Function main

    Print "ReturnLong = ", ReturnLong
Fend
```

参阅
Function...Fend

Declare使用示例

```
' 用Declare命令定义外部函数。
' 如果DLL文件路径未设置全路径，可以在当前项目文件夹。
' 和Windows System32文件夹中存放DLL文件。

Declare MyDLLTest, "mystuff.dll", "MyDLLTest" As Long
```

```
Function main
    Print MyDLLTest
Fend

' 用Declare命令定义拥有2个Integer型自变量的外部函数。

 #define MYSTUFF "mystuff.dll"

Declare MyDLLCall, MYSTUFF, "MyTestFunc", (var1 As Integer, var2 As Integer) As
Integer

' 用Declare命令以全路径指定外部函数，
' 并通过索引指定和定义函数。

Declare MyDLLTest, "c:\mydlls\mystuff.dll", "#1" As Long
```

3.8.4 DegToRad函数

用于将角度转换为弧度。

格式

DegToRad (角度)

参数

角度
　　指定要转换为弧度的角度的值(实值)。

返回值

返回Double型的弧度值。

参阅

ATan、ATan2、RadToDeg函数

DegToRad函数使用示例

```
s = Cos(DegToRad(x))
```

3.8.5 Del

删除文件。

格式
Del文件名

参数

文件名
　　指定要删除的路径和文件名。文件名要附加扩展名。详情请参阅ChDisk。

说明
用于删除指定的文件。

Del使用示例
利用命令窗口的操作示例

```
> Del TEST.PTS     '从当前目录中删除点文件

> Del c:\TEST.PTS     'NG
!! 错误: 7213 指定的文件不存在。
> Del c:\TEST.PTS     'OK
```

3.8.6 DeleteDB

用于从打开数据库内的表格中删除数据。

格式

DeleteDB #数据库编号, 表格名 [, 删除条件]

参数

数据库编号
　　指定利用OpenDB指定的数据库编号(501～508的整数值)。
表格名
　　指定要进行数据删除的表格名。
删除条件
　　指定删除条件。可使用AND、OR指定复合条件。未指定删除条件时，删除表格中的所有数据。

说明

从打开的数据库所指定的表格中删除符合删除条件的数据。

如果打开的数据库是Excel工作薄，则无法执行该命令

注意

- 需要连接已安装RC+的PC。

参阅

OpenDB、CloseDB、SelectDB、UpdateDB

3.8.7 DiffPoint函数

用于返回指定的2点之差。

格式

DiffPoint (点数据1, 点数据2)

参数

点数据1
　　指定第1个点数据。
点数据2
　　指定第2个点数据。

返回值

将从点数据1看到的点数据2的位置姿势作为点数据返回。

说明

将以点数据1为原点的坐标系中的点数据2的位置姿势作为点数据返回。返回的点数据的本地编号或Hand等标志信息为默认值。

2个点数据的任意一个点数据中有未定义的值时，则按“0”计算。

比如，将Point1指定为“XY (10,0,0,0,0,0): ST (10, 10)”、将Point2指定为“XY (10,0,0,0,0,0)”时，由于Point2中未定义S与T的值，而Point1中已进行定义，因此，返回将Point2的S与T的值按“0”进行计算后的值。

注意

- 支持的控制器型号

  不支持T/VT系列。

DiffPoint函数使用示例

```
'显示从点P1看到的P2的位置姿势。

Print DiffPoint(P1, P2)

'显示从当前位置 (Here) 看到的P1的位置姿势。

Print DiffPoint(Here, P1)
```

3.8.8 DiffToolOrientation函数

用于返回工具坐标系各坐标轴形成的角度(单位：度)，以表示通过指定的2个点所实现的各工具姿势的变化量。

格式

DiffToolOrientation (点数据1, 点数据2 , 轴编号)

参数

点数据1
　　指定第1个点数据。
点数据2
　　指定第2个点数据。
轴编号
　　指定用于求出角度变化量的工具坐标系的坐标轴。

常数	值
COORD_X_PLUS	1：+X轴
COORD_Y_PLUS	2：+Y轴
COORD_Z_PLUS	3：+Z轴
COORD_ALL	4：任意轴

返回值

角度(0～180度的正实值)

说明

以指定工具坐标系的坐标轴形成的角度(0～180度的正实值)返回分别通过指定的2个点数据所实现的工具姿势形成的姿势变化量。结果不受点数据1、2顺序的影响。另外，结果不受2点之间的原点位置关系(X、Y、Z的坐标值)的影响。

如果指定了COORD_ALL，可以返回绕任意轴的旋转量。当任意轴有两种姿态(U、V、W)时，可以绕虚拟轴(一条直线)旋转一圈的轴。它不限于每个轴，而是用于求总旋转角度。

注意

- 支持的控制器型号

  T/VT系列无法在轴编号中指定COORD_ALL。

DiffToolOrientation函数使用示例

```
'显示点P1与P2的工具坐标Z轴形成的角度。

Print DiffToolOrientation(P1, P2, COORD_Z_PLUS)
```

3.8.9 DispDev

用于设置当前的显示装置。

格式

DispDev (装置ID)

参数

装置ID

指定要指定的显示装置的装置ID。

- 21 RC+
- 24 TP(仅TP1)
- 20 TP3

还可以使用以下常数。

- DEVID_SELF 21
- DEVID_TP 24
- DEVID_TP3 20

参阅

DispDev函数

DispDev使用示例

```
DispDev DEVID_TP
```

3.8.10 DispDev函数

用于设置当前的显示装置。

格式

DispDev

返回值

用于返回设置了装置ID的显示装置的整数值。

- 21 RC+
- 24 TP(仅TP1)
- 20 TP3

参阅

DispDev

DispDev使用示例

```
Print "The current display device is ", DispDev
```

3.8.11 Dist函数

用于返回2个机器人坐标间的距离。

格式
Dist (坐标1、坐标2)

参数

坐标1、坐标2
　　指定2个机器人的坐标。

返回值
用于返回2个坐标间的距离。(单位：mm)

说明
即使使用附加轴，仅返回机器人的移动距离。比如，即使在移动轴上使用附加轴，也不考虑附加轴移动距离。

在关节型机器人中，本函数的返回值没有意义。

参阅
CU、CV、CW、CX、CY、CZ

Dist函数使用示例

```
Real distance

distance = Dist(P1, P2)
```

3.8.12 Do...Loop

在条件一致期间或者在不一致时到条件一致为止，反复DO...LOOP区段。

格式

Do [{ While | Until } 条件表达式]
[语句]
[Exit Do]
[语句]
Loop

并且或者，使用下述格式。

Do
[语句]
[Exit Do]
[语句]
Loop [{ While | Until } 条件表达式]

Do Loop 语句格式中有条件表达式和语句。

条件表达式
　　表示True或False的数字或字符串表达式。当条件表达式为空(Null)时，条件将作为False来处理。可省略。
语句
　　在条件一致期间或者在条件一致为止在条件一致期间或者到条件一致时为止，反复执行1个以上的语句

说明

作为退出Do...Loop的另一种方法，在Do...Loop中可以随时随地插入Exit Do语句。Exit Do常在用于评价If...Then等几个条件之后使用。如果在If...Then中使用Exit Do语句，则将控制移至Loop的下一语句。

如果在嵌套的Do...Loop语句中使用，则Exit Do将控制移至发生循环的上1级循环。

注意

- 请勿采取在循环语句中频繁重复XQT命令的使用方法

  请勿采取在Do…Loop等循环语句中频繁重复XQT命令的使用方法。否则可能会导致控制器进入挂机状态。如要采取这种使用方法，请追加Wait命令(Wait 0.1)。

- 空无限循环和类似无限循环的处理，请尽可能与Wait一起使用

  空Do...Loop或类似处理，可能会影响您的系统，请尽可能避免使用。当控制器检测到无限空循环，并判断该处理影响系统，则会发出2556错误 (检测到过剩的循环)。在执行需要循环的演算，或等待I/O信号时，请在循环处理中执行Wait命令 (例如，Wait 0.1)，避免占用CPU。

- 不使用Exit Do的情况下从嵌套结构退出循环时

  若重复执行使用Exit Do以外的命令(Gosub、Goto、Call 命令等)退出循环的程序时，会出现错误2020。如果要在循环中退出，请使用Exit Do命令。

参阅

For...Next、Select...Send

Do...Loop使用示例

```
Do While Not Lof(1)
    Line Input #1, tLine$
```

```
    Print tLine$
Loop
```

3.8.13 Double

用于声明Double型变量。(8字节的双精度数)

格式

Double 变量名 [(数组变量的最大下标)] [, 变量名 [(数组变量的最大下标)]...]

参数

变量名
　　指定声明为Double型的变量的名称。
数组变量的最大下标
　　可利用数组变量的最大下标声明到最高三维。使用下述格式。可省略。

　　(最大下标1, [最大下标2], [最大下标3])

　　由于下标从0开始，因此数组元素数为最大下标加上1。在所有数组数不超过以下最大值的范围内指定各最大下标。

- 本地变量：2,000
- 备份变量(Global Preserve)：4,000
- 全局变量和模块变量：100,000

说明

Double用于将指定变量声明为Double型。在Function开头声明本地变量。在Function之外声明全局变量和模块变量。

Double型的有效位数为14位。

参阅

Boolean、Byte、Global、Int32、Int64、Integer、Long、Real、Short、String、UByte、UInt32、UInt64、UShort

Double使用示例

下例为使用Double声明几个变量的示例。

```
Function doubletest
  Double var1
  Double A(10)          'Double型的一维数组
  Double B(10, 10)      'Double型的二维数组
  Double C(5, 5, 5)     'Double型的三维数组
  Double arrayvar(10)
  Integer i
  Print "Please enter a Number:"
  Input var1
  Print "The variable var1 = ", var1
  For i = 1 To 5
    Print "Please enter a Number:"
    Input arrayvar(i)
    Print "Value Entered was ", arrayvar(i)
  Next i
Fend
```

3.9 E

3.9.1 ECP

选择或显示当前的ECP(外部控制点)。

格式
(1) ECP ECP编号

(2) ECP

参数

ECP编号
利用后续的动作命令指定使用16个ECP设置(整数值0～15)中的哪一个。可省略。ECP 0用于使ECP选择无效。

结果
如果省略参数，则显示当前设置的ECP编号。

说明
ECP命令用于选择由ECP编号指定的外部控制点。

注意

- 此命令只在已安装外部控制点选件时才可使用。
- 关闭电源对ECP选择的影响

  如果关闭主电源，将清除选择的ECP的选择状态。

参阅
ECPSet

ECP使用示例

```
>ecpset 1, 100, 200, 0, 0
>ecp 1
```

3.9.2 ECP函数

用于返回当前指定的ECP(外部控制点)编号。

格式
ECP

返回值
用于以整数值返回当前指定的ECP编号。

注意

此命令只在已安装外部控制点选件时才可使用。

参阅
ECP

ECP函数使用示例

```
Integer savECP

savECP = ECP
ECP 2
Call Dispense
ECP savECP
```

3.9.3 ECPClr

清除(未设置)外部控制点的设置。

格式

ECPClr ECP编号

参数

ECP编号
以整数值指定1～15的外部控制点中的要清除(未设置)的编号。(ECP的0号是初始设置值，无法清除。)

说明

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

注意

此命令只在已安装外部控制点选件时才可使用。

参阅

Arm、ArmClr、ArmSet、ECPSet、Local、LocalClr、Tool、TLSet

ECPClr使用示例

```
ECPClr 1
```

3.9.4 ECPDef函数

用于返回外部控制点的设置状态。

格式

ECPDef (ECP编号)

参数

ECP编号
以整数值指定要调用的状态的ECP编号。

返回值

如果指定的ECP编号的外部控制点已设置，则返回“True”；如果未设置，则返回“False”。

参阅

Arm、ArmClr、ArmSet、ECPSet、Local、LocalClr、Tool、TLClr、TLSet

ECPDef使用示例

```
Function DisplayECPDef(ecpNum As Integer)

   If ECPDef(ecpNum) = False Then
      Print "ECP ", ecpNum, "is not defined"
   Else
      Print "ECP ", ecpNum, ": ",
      Print ECPSet(ecpNum)
   EndIf
Fend
```

3.9.5 ECPSet

用于定义或显示外部控制点。

格式

(1) ECPSet ECP编号, ECP点指定

(2) ECPSet ECP编号

(3) ECPSet

参数

ECP编号
　　以表达式或1～15的整数值指定定义为外部控制点的编号。
ECP点指定
　　以P编号或P(表达式)、点标签、点数据进行指定。

返回值

如果省略所有参数，则显示当前的ECP设置。

如果只指定ECP编号，则显示指定的ECP设置。

说明

设置外部控制点。

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

注意

此命令只在已安装外部控制点选件时才可使用。

ECPSet使用示例

```
ECPSet 1, P1
ECPSet 2, 100, 200, 0, 0
```

3.9.6 ECPSet函数

用于返回分配给指定ECP编号的外部控制点的点数据。

格式

ECPSet (ECP编号)

参数

ECP编号
以整数值指定要调用的点数据的ECP编号。

返回值

用于返回指定ECP编号的外部控制点的点数据。

注意

此命令只在已安装外部控制点选件时才可使用。

参阅

ECPSet

ECPSet使用示例

```
P1 = ECPSet(1)
```

3.9.7 ElapsedTime函数

用于以秒为单位返回计算节拍时间用计时器开始计时之后的经过时间。

格式
ElapsedTime

返回值
以实值(单位：秒)返回计算节拍时间用计时器的经过时间。计时器的范围为0～约1.7E+31。计时器的分辨率为0.001秒。

说明
用于返回计算节拍时间用计时器开始计时之后的经过时间。与Tmr函数不同，此函数不会将程序暂停状态的时间作为经过时间来计算。

计算节拍时间用计时器可以通过 ResetElapsedTime来重置。

```
Real overhead

ResetElapsedTime
overHead = ElapsedTime
```

参阅
ResetElapsedTime、Tmr函数

ElapsedTime使用示例

```
ResetElapsedTime        '重置计算节拍时间用的计时器
For i = 1 To 10         '执行10次
    GoSub Cycle
Next
Print ElapsedTime / 10 '计算并显示节拍时间
```

3.9.8 Elbow

用于设置点的肘姿势。

格式

(1) Elbow 指定点 [,设置值]

(2) Elbow

参数

点指定
以P编号、P(表达式)或点标签进行指定。
设置值
以整数或表达式进行指定。

- 1 = Above (/A)
- 2 = Below (/B)

返回值

如果省略2个参数，则显示机器人当前位置的肘姿势。

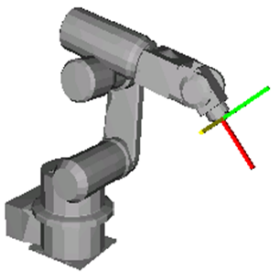
如果省略设置值参数，将显示指定点的肘姿势。

参阅

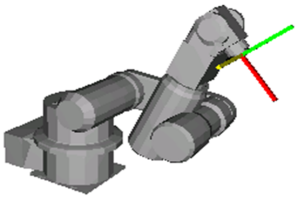
Elbow函数、Hand、J4Flag、J6Flag、Wrist

Elbow使用示例

```
Elbow P0, Below
Elbow pick, Above
Elbow P(myPoint), myElbow

P1 = 0.000,  490.000,  515.000,   90.000,  -40.000,  180.000
```



```
Elbow P1, Above
Go P1
```



```
Elbow P1, Below
Go P1
```

3.9.9 Elbow函数

用于返回点的肘姿势。

格式

Elbow [(指定点)]

参数

点指定

以表达式指定点。可省略。如果省略，将返回机器人当前位置的肘姿势。

返回值

- 1: Above (/A)
- 2: Below (/B)

参阅

Elbow、Hand、Wrist、J4Flag、J6Flag

Elbow函数使用示例

```
Print Elbow(pick)
Print Elbow(P1)
Print Elbow
Print Elbow(P1 + P2)
```

3.9.10 Eof函数

用于返回文件的EOF(已打开的文件的指针位于尾端)。

格式
Eof (文件编号)

参数

文件编号
以30～63之间的整数值或表达式进行指定。

返回值
如果文件为EOF，则返回“True”；如果不是，则返回“False”。

说明
用于在读取模式下打开文件。

如果是通过AOpen、WOpen命令打开的文件，将发生错误。

参阅
Lof

Eof函数使用示例

```
Integer fileNum
String data$

fileNum = FreeFile
UOpen "TEST.DAT" As #fileNum
Do While Not Eof(fileNum)
    Line Input #fileNum, data$
    Print "data = ", data$
Loop
Close #fileNum
```

3.9.11 Era函数

用于返回发生错误的关节的编号。

格式
Era [(任务编号)]

参数

任务编号
以整数指定0～32的任务编号。省略或“0”时，为当前任务。

返回值
以下述0～9的整数值告知发生错误的关节编号。

- 0 - 当前的错误不是关节造成的。
- 1 - 当前的错误是第1关节造成的。
- 2 - 当前的错误是第2关节造成的。
- 3 - 当前的错误是第3关节造成的。
- 4 - 当前的错误是第4关节造成的。
- 5 - 当前的错误是第5关节造成的。
- 6 - 当前的错误是第6关节造成的。
- 7 - 当前的错误是第7关节造成的。
- 8 - 当前的错误是第8关节(附加轴S)造成的。
- 9 - 当前的错误是第9关节(附加轴T)造成的。

说明
Era在发生错误时找到该错误是在哪个关节上产生并告知其关节编号。如果当前错误不是该关节造成的，将返回“0”。

如果在自动运转模式(AUTO)的一般任务与NoPause任务中出现“自动运转期间发生错误”，将中断执行并结束任务。如果在NoEmgAbort任务或后台任务中使用本函数时已结束对象任务，将发生“错误2261”。如要在任务结束之前获取信息，请使用OnErr。

参阅
Erl, Err, ErrMsg$, Ert, OnErr, Trap

Era函数使用示例

```
Function main
  OnErr Goto eHandler
  Do
    Call PickPlace
  Loop
  Exit Function
eHandler:
  Print "The Error code is ", Err
  Print "The Error Message is ", ErrMsg$(Err)
  errTask = Ert
  If errTask > 0 Then
    Print "Task number in which error occurred is ", errTask
    Print "The line where the error occurred is Line ", Erl(errTask)
    If Era(errTask) > 0 Then
      Print "Joint which caused the error is ", Era(errTask)
    EndIf
  EndIf
Fend
```

3.9.12 EResume

在结束错误处理例程后，重新开始执行程序。

格式

EResume [{标签 | Next }]

说明

- EResume

  在与错误处理例程相同的函数中发生错误时，从造成错误的语句开始重新执行程序。

  在调用的函数内发生错误时，从包括错误处理例程在内的函数内的Call语句开始重新执行程序。

- EResume Next

  在与错误处理例程相同的函数中发生错误时，从造成错误的语句的下一语句开始重新执行程序。

  在调用的函数内发生错误时，从包括错误处理例程在内的函数所调用的最后的Call语句的下一语句开始重新执行程序。

- EResume {标签}

  在与错误处理例程相同的函数内发生错误时，从包括指定标签在内的语句开始重新执行程序。

参阅

OnErr

EResume使用示例

```
Function main
  Integer retry

  OnErr GoTo eHandler
  Do
    RunCycle
  Loop
  Exit Function

eHandler:
  Select Err
    Case MyError
      retry = retry + 1
      If retry < 3 Then
        EResume '重新执行
      Else
        Print "MyError has occurred ", retry, " times
      EndIf
  Send
Fend
```

3.9.13 Erf$函数

返回发生错误的函数名。

格式
Erf$ [(任务编号)]

参数

任务编号
以0～32的整数值指定任务编号。省略或“0”时，为当前任务。

返回值
返回最后发生错误的函数名。

说明
Erf$可以与OnErr一起使用。Erf$用于返回发生错误的函数名。通过将Erf$与 Err、Ert、Erl、Era等进行组合，可以就发生的错误收集更详细的信息。

如果在自动运转模式(AUTO)的一般任务与NoPause任务中出现“自动运转期间发生错误”，将中断执行并结束任务。如果在NoEmgAbort任务或后台任务中使用本函数时已结束对象任务，将发生“错误2261”。如要在任务结束之前获取信息，请使用OnErr。

参阅
Era, Erl, Err, ErrMsg$, Ert, OnErr, Trap

Erf$函数使用示例
下述为调查如下内容的简单程序。

- 是在哪一任务中发生了错误(Ert函数)
- 是在哪一函数中发生了错误(Erf$函数)
- 是在哪里发生的(Erl函数)
- 在哪个关节上发生了错误(Era函数)

```
Function main
  OnErr Goto eHandler
  Do
    Call PickPlace
  Loop
  Exit Function
eHandler:
  Print "The Error code is ", Err
  Print "The Error Message is ", ErrMsg$(Err)
  errTask = Ert
  If errTask > 0 Then
    Print "Task number in which error occurred is ", errTask
    Print "Function at which error occurred is ", Erf$(errTask)
    Print "The line where the error occurred is Line ", Erl(errTask)
    If Era(errTask) > 0 Then
      Print "Joint which caused the error is ", Era(errTask)
    EndIf
  EndIf
Fend
```

3.9.14 Erl函数

返回发生错误的行编号。

格式
Erl [(任务编号)]

参数

任务编号
以0～32的整数值指定任务编号。省略或“0”时，为当前任务。

返回值
返回最后发生错误的行编号。

说明
Erl可以与OnErr一起使用。Erl用于返回发生错误的行编号。通过将Erl与 Err、Ert、Era等进行组合，可以就发生的错误收集更详细的信息。

如果在自动运转模式(AUTO)的一般任务与NoPause任务中出现“自动运转期间发生错误”，将中断执行并结束任务。如果在NoEmgAbort任务或后台任务中使用本函数时已结束对象任务，将发生“错误2261”。如要在任务结束之前获取信息，请使用OnErr。

参阅
Era, Erf$, Err, ErrMsg$, Ert, OnErr

Erl函数使用示例
下述为调查如下内容的简单程序。

- 是在哪一任务中发生了错误(Ert函数)
- 是在哪里发生的(Erl函数)
- 发生了什么样的错误(Err函数)
- 在哪个关节上发生了错误(Era函数)

```
Function main
  OnErr Goto eHandler
  Do
    Call PickPlace
  Loop
  Exit Function
eHandler:
  Print "The Error code is ", Err
  Print "The Error Message is ", ErrMsg$(Err)
  errTask = Ert
  If errTask > 0 Then
    Print "Task number in which error occurred is ", errTask
    Print "The line where the error occurred is Line ", Erl(errTask)
    If Era(errTask) > 0 Then
      Print "Joint which caused the error is ", Era(errTask)
    EndIf
  EndIf
Fend
```

3.9.15 Err函数

用于返回最新的错误状态。

格式
Err [(任务编号)]

参数

任务编号
以0～32的整数值指定任务编号。“0”时指定当前任务。可省略。

返回值
以整数值返回错误代码。

说明
Err函数告知用户当前的错误代码。在用于应对SPEL+的错误的同时告知发生了什么错误，使得可以进行适当的应对。Err可以与OnErr一起使用。

要获取控制器的错误，可以使用SysErr函数。

如果在自动运转模式(AUTO)的一般任务与NoPause任务中出现“自动运转期间发生错误”，将中断执行并结束任务。如果在NoEmgAbort任务或后台任务中使用本函数时已结束对象任务，将发生“错误2261”。如要在任务结束之前获取信息，请使用OnErr。

参阅
Era, Erf$, Erl, ErrMsg$, EResume, Ert, OnErr, Return, SysErr

Err使用示例
下例为检查是否有点P0～P399的简单实用程序。如果没有点，将在画面上显示将其告知用户的信息。使用CX命令测试是否定义了各点。如果有未定义的点，将转至错误处理控制并在画面上显示未定义该点。

```
Function errtest
   Integer i, errnum
   Real x

   OnErr GoTo eHandle
   For i = 0 To 399
     x = CX(P(i))
   Next i
   Exit Function
'
'
'*******************************************
'* Error Handler                           *
'*******************************************
eHandle:
   errnum = Err
   '确认是否使用未定义点
   If errnum = 78 Then
      Print "Point number P", i, " is undefined!"
   Else
      Print "ERROR: Error number ", errnum, " Occurred."
   EndIf
   EResume Next
Fend
```

3.9.16 Errb函数

用于返回发生错误的机器人的编号。

格式
Errb

返回值
用于返回发生错误的机器人的编号。

说明
Errb函数在发生错误时找到该错误是在哪个机器人上产生并告知其机器人编号。如果当前错误不是该机器人造成的，将返回“0”。

参阅
Era、Erl、Err、ErrMsg$、OnErr、Trap

Errb函数使用示例
在以下程序例中将显示下述内容。

- 是在哪一任务中发生了错误(Ert函数)
- 是在哪里发生的(Erl函数)
- 发生了什么样的错误(Err函数)
- 在哪个关节上发生了错误(Era函数)
- 在哪个机器人上发生了错误(Errb函数)

```
Function main
  OnErr Goto eHandler
  Do
    Call PickPlace
  Loop
  Exit Function
eHandler:
  Print "The Error code is ", Err
  Print "The Error Message is ", ErrMsg$(Err)
  errTask = Ert
  If errTask > 0 Then
    Print "Task number in which error occurred is ", errTask
    Print "The line where the error occurred is Line ", Erl(errTask)
    If Era(errTask) > 0 Then
      Print "Joint which caused the error is ", Era(errTask)
    EndIf
  Print "Robot number in which error occurred is ", errb
  EndIf
Fend
```

3.9.17 ErrMsg$函数

返回指定错误编号的错误信息。

格式

ErrMsg$ (错误编号, 语言编号)

参数

错误编号
以整数值指定返回信息的错误编号。
语言编号
以如下整数值指定语言。可省略。

- 0 - 英语
- 1 - 日本语
- 2 - 德语
- 3 - 法语
- 4 - 汉语(简体字)
- 5 - 汉语(繁体字)
- 6 - 西班牙语

省略时，指定英语。

返回值

用于返回错误代码表的错误信息。

参阅

Era、Erl、Err、Ert、OnErr、Trap

ErrMsg$函数使用示例

下述为调查如下内容的简单程序。

- 是在哪一任务中发生了错误(Ert函数)
- 是在哪里发生的(Erl函数)
- 是否在关节上发生了错误(Era函数)

```
Function main
  OnErr Goto eHandler
  Do
    Call PickPlace
  Loop
  Exit Function
eHandler:
  Print "The Error code is ", Err
  Print "The Error Message is ", ErrMsg$(Err)
  errTask = Ert
  If errTask > 0 Then
    Print "Task number in which error occurred is ", errTask
    Print "The line where the error occurred is Line ", Erl(errTask)
    If Era(errTask) > 0 Then
      Print "Joint which caused the error is ", Era(errTask)
    EndIf
  EndIf
Fend
```

3.9.18 Error

用于发生用户定义的错误。

格式

(1) Error 任务编号, 错误编号

(2) Error 错误编号

参数

任务编号
　　以0～32的整数值指定任务编号。“0”时指定当前任务。可省略。
错误编号
　　以整数值指定错误编号。以后错误编号为8000～8999。

说明

Error语句将产生系统或用户定义的错误。可以使用Epson RC+开发环境的用户错误编辑器，定义用户错误的标签和记述。

参阅

Era、Erl、Err、OnErr

Error使用示例

```
 #define ER_VAC 8000

If Sw(vacuum) = Off Then
    Error ER_VAC
EndIf
```

3.9.19 ErrorOn函数

用于返回控制器的错误状态。

格式
ErrorOn

返回值
如果处于错误状态，则返回“True”；如果不是，则返回“False”。

说明
本函数仅用于NoEmgAbort任务(执行Xqt时指定NoEmgAbort以开始的特别任务)与后台任务。

参阅
ErrorOn、SafetyOn、SysErr、Wait、Xqt

ErrorOn函数使用示例
下例所示为监视控制器的错误状态，并在发生错误时，根据错误编号对I/O进行ON/OFF操作的程序。

注意

- Forced标志

  本程序示例所示为在On/Off命令中指定Forced标志。

  发生错误/紧急停止期间或安全门打开时，I/O输出会发生变化，因此，在系统设计方面需要注意。

- 发生错误之后的处理

  如本例所示，发生错误并进行必要的处理之后，请立即结束任务。

```
Function main

Xqt ErrorMonitor, NoEmgAbort
:
:
Fend

Function ErrorMonitor
  Wait ErrorOn
  If 4000 < SysErr Then
    Print "Mortion Error = ", SysErr
    Off 10, Forced
    On 12, Forced
  Else
    Print "Other Error = ", SysErr
    Off 11, Forced
    On 13, Forced
  EndIf

Fend
```

3.9.20 Ert函数

用于返回发生错误的任务编号。

格式
Ert

返回值
用于返回发生错误的任务编号。

说明
Ert函数用于获取在哪一任务时发生了错误。

返回没有发生错误的任务(0)、一般任务(1～32)、后台任务(65～80)、TRAP任务(257～267)的编号。

参阅
Era, Erl, Err, ErrMsg$, OnErr, Trap

Ert函数使用示例
在以下程序例中将显示下述内容。

- 是在哪一任务中发生了错误(Ert函数)
- 是在哪里发生的(Erl函数)
- 在哪个关节上发生了错误(Era函数)

```
Function main
  OnErr Goto eHandler
  Do
    Call PickPlace
  Loop
  Exit Function
eHandler:
  Print "The Error code is ", Err
  Print "The Error Message is ", ErrMsg$(Err)
  errTask = Ert
  If errTask > 0 Then
    Print "Task number in which error occurred is ", errTask
    Print "The line where the error occurred is Line ", Erl(errTask)
    If Era(errTask) > 0 Then
      Print "Joint which caused the error is ", Era(errTask)
    EndIf
  EndIf
Fend
```

3.9.21 EStopOn函数

用于返回紧急停止状态

格式
EstopOn

返回值
如果处于紧急停止状态，则返回“True”；如果不是，则返回“False”。

说明
本函数仅用于NoEmgAbort任务(执行Xqt时指定NoEmgAbort以开始的特别任务)。

参阅
ErrorOn、SafetyOn、Wait、Xqt

EstopOn函数使用示例
下例所示为监视控制器进入紧急停止状态的情况，并在发生紧急停止时，对I/O进行ON/OFF操作的程序。

注意

- Forced标志

  本程序示例所示为在On/Off命令中指定的Forced标志。

  发生错误、紧急停止期间或安全门打开时，I/O输出会发生变化，因此，在系统设计方面需要注意。

- 发生错误之后的处理

  如本例所示，发生错误并进行必要的处理之后，请立即结束任务。

- 紧急停止时将输出端口设为OFF

  如本例所示，紧急停止之后也要执行打开或关闭I/O的任务时，建议取消勾选[设置]-[系统设置]-[控制器]-[环境设置]-[紧急停止时将输出端口设为OFF]。如果保持勾选状态，则无法保证执行通过控制器将I/O设为Off或通过任务将I/O设为On两者的前后次序。

```
Function main

  Xqt EStopMonitor, NoEmgAbort
  :
  :
Fend

Function EStopMonitor
  Wait EStopOn
  Print "EStop !!!"
  Off 10, Forced
  On 12, Forced
Fend
```

3.9.22 Eval函数

用于执行命令窗口的语句并返回错误状态。

格式
Eval (命令 [, 命令的输出结果])

参数

命令
　　以字符串指定要执行的命令。
命令的输出结果
　　指定要保存的命令输出结果的字符串变量。可省略。当命令出错时，返回“!Error: 错误代码”。如果输出结果超过255个字符，将舍去超出的结果。

返回值
返回通过执行命令返回的错误代码。

即使执行命令出错，本函数本身也不会出错。并且，也不会在系统日志中留下错误。

当命令正常结束时，返回“0”。

说明
如果使用Eval函数，可以从TCP/IP等通信端口执行任意命令。

可执行的命令是可以从命令窗口执行的命令。

执行本函数比执行一般语句要花费处理时间。

使用命令的输出结果参数以获取命令的返回值。例如，相对于“PrintSw(1)”命令，命令的输出结果将返回“1”或“0”。

参阅
错误信息一览

Eval函数使用示例
下例显示了如何执行RS-232C读取的命令。在执行命令后，向主机返回错误代码。例如，主机将发送"motor on"等的命令。

```
Integer errCode
String cmd$

OpenCom #1
Do
  Line Input #1, cmd$
  errCode = Eval(cmd$)
  Print #1, errCode
Loop
```

3.9.23 Exit

用于强制结束循环或函数。

格式

Exit { Do | For | Function }

说明

Exit 语句的格式如下所示。

Exit Do
退出Do...Loop语句。语句只能在Do...Loop 语句内使用。通过Exit Do，控制将转至Loop语句的下一语句。如果在嵌套的Do...Loop语句内使用，控制将转至有Exit Do的循环的上1级循环。

Exit For
退出For循环。只能在For...Next循环内使用。Exit For用于将控制转至Next语句的下一语句。如果在嵌套的For循环内使用，控制将转至有Exit For的循环的上1级循环。

Exit Function
用于要在任意位置上退出函数之时使用。将从调用函数的语句的下一语句开始继续执行程序。

参阅

Do...Loop、For...Next、Function...Fend

Exit使用示例

```
For i = 1 To 10
    If Sw(1) = On Then
        Exit For
    EndIf
    Jump P(i)
Next i
```

3.9.24 ExportPoints

用于将点文件导出到指定路径中。

格式

ExportPoints 文件名, 保存目标

参数

文件名
　　表示要导出的特定文件的字符串表达式扩展名为“.pts”。无法指定路径。另外，不受ChDisk等的影响。详情请参阅ChDisk。
保存目标
　　指定保存目标的路径与文件名。扩展名为“.pts”。详情请参阅ChDisk。

说明

ExportPoints用于将指定的点文件复制到PC上的文件夹中。PC上的文件夹中已存在同名文件时，将进行覆盖。

易引起的错误

- 不存在保存目标路径时

  指定的保存目标路径不存在时，将发生错误。

- 找不到指定文件时

  如果文件名中包括路径，将发生错误。

参阅

Dir、LoadPoints、SavePoints、FileExists、FolderExists

ExportPoints使用示例

```
Function main
  LoadPoints "robot1.pts"
  :
  SavePoints "robot1.pts"
  If FolderExists("c:\mypoints\") Then
    ExportPoints "robot1.pts", "c:\mypoints\model1.pts"
  EndIf
Fend
```

3.10 F

3.10.1 FbusIO_GetBusStatus函数

用于返回指定的现场总线的状态。

格式

FbusIO_GetBusStatus (总线编号)

参数

路径编号

表示现场总线系统编号的整数表达式此编号必须是16，是连接到控制器PC侧的现场总线主站端口上的总线的ID。

返回值

- 0 - OK
- 1 - 未连接
- 2 - 关闭电源

说明

FbusIO_GetBusStatus函数可以确认现场总线的状态。

注意

此命令仅在现场总线主站选件有效时可使用。

参阅

FbusIO_GetDeviceStatus, FbusIO_SendMsg

FbusIO_GetBusStatus函数使用示例

```
Long sts
sts = FbusIO_GetBusStatus(16)
```

3.10.2 FbusIO_GetDeviceStatus函数

用于返回指定的现场总线装置的状态。

格式

FbusIO_GetDeviceStatus (总线编号, 装置ID)

参数

路径编号
　　表示现场总线系统编号的整数表达式此编号必须是16，是连接到控制器PC侧的现场总线主站端口上的总线的ID。
装置ID
　　表示装置的现场总线ID的整数表达式

返回值

- 0 - OK
- 1 - 未连接
- 2 - 关闭电源
- 3 - 同步错误: 装置正在初始化，或者装置的波特率不正确。

说明

FbusIO_GetDeviceStatus函数可以确认现场总线装置的状态。

注意

此命令仅在现场总线主站选件有效时可使用。

参阅

FbusIO_GetBusStatus, FbusIO_SendMsg

FbusIO_GetDeviceStatus函数使用示例

```
Long sts
sts = FbusIO_GetDeviceStatus(16, 10)
```

3.10.3 FbusIO_SendMsg

向现场总线I/O装置发送信息、返回答复。

格式

FbusIO_SendMsg (执行编号, 装置ID, msgParam, sendData (), recvData ())

参数

路径编号
　　表示现场总线系统编号的整数表达式此编号必须是16，是连接到控制器PC侧的现场总线主站端口上的总线的ID。
装置ID
　　表示装置的现场总线ID的整数表达式
msgParam
　　表示信息参数的整数表达式在DeviceNet中无法使用。
sendData
　　以Byte型数组指定发送至装置中的数据。此数组的维度必须与发送的字节数的维度相同。不发送数据时，指定0。
recvData
　　以Byte型数组指定从装置接收的数据。此数组将自动转换为与接收字节数相对应的维度数。

说明

FBusIO_SendMsg作为相对于现场总线I/O装置的查询来使用。关于支持信息，请垂询设备制造商。

注意

此命令仅在现场总线主站选件有效时可使用。

参阅

FbusIO_GetBusStatus, FbusIO_GetDeviceStatus

FbusIO_SendMsg使用示例

```
' 向DeviceNet装置发送明确信息
Byte sendData(5)
Byte recvData(0)
Integer i

sendData(0) = &H0E  ' 命令
sendData(1) = 1     ' 等级
sendData(3) = 1     ' 实例
sendData(5) = 7     ' 属性
' msgParam is 0 for DeviceNet
FbusIO_SendMsg 16, 1, 0, sendData(), recvData()
' 显示答复
For i = 0 to UBound(recvData)
  Print recvData(i)
Next i

' 向Profibus装置发送信息
Byte recvData(0)
Integer i

' msgParam为服务编号
FbusIO_SendMsg 16, 1, 56, 0, recvData()
' 显示答复
For i = 0 to UBound(recvData)
  Print recvData(i)
Next i
```

3.10.4 FileDateTime$函数

用于返回文件的日期和时间。

格式
FileDateTime$ (文件名)

参数

文件名
以字符串指定要确认的文件名。包括盘符和路径名。仅指定文件名时，是指当前目录中的文件。详情请参阅ChDisk。

注意

可使用网络路径。

返回值
以下述格式返回文件的最终修改日期时间。

月/日/年时:分:秒

参阅
FileExists、FileLen

FileDateTime$函数使用示例

```
String myPath$
myPath$ = "c:\TEST\TEST.DAT"

If FileExists(myPath$) Then
    Print "Last access date and time: ", FileDateTime$(myPath$)
    Print "Size: ", FileLen(myPath$)
EndIf
```

3.10.5 FileExists函数

用于检查有无文件。

格式

FileExists (文件名)

参数

文件名

以字符串指定要确认的文件名。包括盘符和路径名。仅指定文件名时，是指当前目录中的文件。详情请参阅ChDisk。

注意

可使用网络路径。

返回值

- 存在文件时：True
- 不存在文件时：False

参阅

FolderExists、FileLen、FileDateTime$

FileExists函数使用示例

```
String myPath$
myPath$ = "c:\TEST\TEST.DAT"

If FileExists(myPath$) Then
    Print "Last access date and time: ", FileDateTime$(myPath$)
    Print "Size: ", FileLen(myPath$)
EndIf
```

3.10.6 FileLen函数

用于返回文件大小。

格式
FileLen (文件名)

参数

文件名
以字符串指定要确认的文件名。包括盘符和路径名。仅指定文件名时，是指当前目录中的文件。详情请参阅ChDisk。

注意

可使用网络路径。

返回值
用于返回文件的字节数。

参阅
FileDateTime$、FileExists

FileLen函数使用示例

```
String myPath$
myPath$ = "c:\TEST\TEST.DAT"

If FileExists(myPath$) Then
    Print "Last access date and time: ", FileDateTime$(myPath$)
    Print "Size: ", FileLen(myPath$)
EndIf
```

3.10.7 Find

用于设置和显示在动作命令中保存坐标的条件。

格式

Find [条件表达式]

参数

事件条件表达式

指定触发的输入状态。

[条件] 比较运算符(=、<>、>=、>、<、<=)[整数表达式]

可在条件中使用下述函数或变量。

- 函数：Sw, In, InW, Oport, Out, OutW, MemSw, MemIn, MemInW, Ctr, GetRobotInsideBox, GetRobotInsidePlane, AIO_In, AIO_InW, AIO_Out, AIO_OutW, Hand_On, Hand_Off, SF_GetStatus
- 变量：Byte, Int32, Integer, Long, Short, UByte, UInt32, UShort型备份变量、全局变量、模块变量

另外，可利用下述运算符对多个事件条件表达式附加掩码或进行复合组合。

- 运算符：And、Or、Xor

[例]

```
Sense Sw(5) = On
Sense Sw(5) = On And Sw(6) = Off
```

说明

请单独记述Find 语句或记述为动作命令语句的修饰符。

Find条件表达式必须包含1个以上的上述函数。

Find条件表达式中包含变量时，在设置Find条件时运算其值。由于可能会形成不希望有的条件，因此不建议在条件表达式中使用变量。也可以记述多个Find语句。此时，最后执行的Find 条件有效。

如果省略参数，则显示当前的Find设置。

注意

- 电源ON时的Find设置

  电源ON时Find条件的初始设置为Find Sw(0) = On。输入位编号0为ON时，设为进行坐标保存。

- 检查Find 条件成立的PosFound函数

  执行使用Find修饰符的动作命令之后，可使用PosFound函数检查Find条件是否成立。

- 在条件表达式中使用变量时
  - 可使用变量类型为整数型(Byte、Int32、Integer、Long、Short、UByte、UInt32、UShort)。
  - 不能使用数组变量。
  - 不能使用本地变量。
  - 在超过0.01秒的时间内变量值未满足条件时，系统可能不能检测到变量变化。
  - 系统内可使用的变量等待数存在限制。1个系统内可使用的变量等待数量最多为64个(也包括在Wait等条件表达式中使用的变量等待)。如果超过最大数，则会在项目创建时发生错误。
  - 如果利用Byref引用执行变量等待的变量，则会发生错误。
  - 条件表达式右边的整数表达式中包括变量时，在动作命令开始时运算其值。由于可能会形成不希望有的条件，因此不建议在整数表达式中使用变量。

参阅

FindPos, Go, Jump, PosFound, SF_GetStatus

Find使用示例

```
Find Sw(5) = On
Go P10 Find
If PosFound Then
    Go FindPos
Else
    Print "Cannot find the sensor signal."
EndIf
```

3.10.8 FindPos函数

用于在执行动作命令过程中通过Find返回保存的坐标。

格式

FindPos

返回值

用于在执行动作命令过程中通过Find返回保存的坐标。

参阅

Find、Go、Jump、PosFound、CurPos、InPos

FindPos函数使用示例

```
Find Sw(5) = On
Go P10 Find
If PosFound Then
    Go FindPos
Else
    Print "Cannot find the sensor signal."
EndIf
```

3.10.9 Fine

用于设置和显示目标位置的定位结束判断范围。

格式

(1) Fine 第1关节设置值, 第2关节设置值, 第3关节设置值, 第4关节设置值[,第5关节设置值, 第6关节设置值] [,第7关节设置值] [,第8关节设置值, 第9关节设置值]

(2) Fine

参数

第1关节设置值
　　以0～65535的整数指定第1关节的定位容许范围。
第2关节设置值
　　以0～65535的整数指定第2关节的定位容许范围。
第3关节设置值
　　以0～65535的整数指定第3关节的定位容许范围。
第4关节设置值
　　以0～65535的整数指定第4关节的定位容许范围。
第5关节设置值
　　以0～65535的整数指定第5关节的定位容许范围。可省略。只在垂直6轴型机器人(包括N系列)上使用。
第6关节设置值
　　以0～65535的整数指定第6关节的定位容许范围。可省略。只在垂直6轴型机器人(包括N系列)上使用。
第7关节设置值
　　以0～65535的整数指定第7关节的定位容许范围。可省略。只在关节型7轴机器人上使用。
第8关节设置值
　　以0～65535的整数指定第8关节的定位容许范围。可省略。只在附加轴S关节上使用。
第9关节设置值
　　以0～65535的整数指定第9关节的定位容许范围。可省略。只在附加轴T关节上使用。

*C8、C12系列时，定位容许范围为0～131070的整数。

结果

在未指定参数时，Fine显示各关节的当前设置值。

说明

Fine是相对于各关节确认动作结束时定位的动作命令，指定判断相对于指定位置的定位结束的容许范围。

此定位结束判断在从CPU向伺服系统发送目标坐标脉冲后开始实施。由于伺服延迟，在该阶段机器人未到达目标位置。在各关节设置的容许范围内，每几毫秒进行一次判断。当所有关节都到达指定的容许范围内，即结束定位。定位结束后，程序控制将转至下一语句，但是伺服系统将继续控制机器人到达目标位置。

如果通过Fine命令指定较大的范围，则定位将在比较早的阶段结束并执行下一语句。

Fine的默认设置因机器人类型而异。有关详细信息，请参阅以下手册。
《机械手手册》

注意

- 循环时间与Fine命令

  Fine值本身不影响机械臂的加速或减速，但是如果设置详细的Fine值，则伺服达到其容许范围将会花费时间(几毫秒)。系统的循环时间只延迟了该时间。如果将机械臂定位在Fine命令设置的容许范围内，CPU将执行以下命令。

- Fine的初始化(依据Motor On, SLock, SFree 等命令)

  如果使用下述任一命令，Fine的设置值将被初始化为默认值。
  SLock, SFree, Motor

执行这些命令后，请务必重新设置Fine值。

易引起的错误

如果没有在2秒内结束基于Fine的定位，将出现错误代码4024。此错误一般意味着需要进行伺服系统的平衡调整。

参阅

Accel、AccelR、AccelS、Arc、Go、Jump、Move、Speed、SpeedR、SpeedS、Pulse、FineDist、FineStatus

Fine使用示例

下述程序为从程序函数执行Fine的示例，以及从监视器窗口执行Fine的示例。

```
Function finetest
    Fine 5, 5, 5, 5     '将精度设为+/-5脉冲
    Go P1
    Go P2
Fend


> Fine 10, 10, 10, 10
>
> Fine
10, 10, 10, 10
```

3.10.10 Fine函数

用于返回指定关节的Fine设置。

格式

Fine (关节编号)

参数

关节编号
以整数值指定要获取Fine设置的关节编号。附加轴的S轴为8，T轴为9。

返回值

返回实值。

参阅

Accel、AccelS、Arc、Go、Jump、Move、Speed、SpeedS、Pulse

Fine函数使用示例

如下所示为使用Fine函数的程序例。

```
Function finetst
    Integer a
    a = Fine(1)
Fend
```

3.10.11 FineDist

用于设置和显示目标位置的定位结束判断范围。设置值的单位均为mm。

格式
(1) FineDist 设置值

(2) FineDist

参数

设置值
　　定位容许范围为0.001[mm]～10[mm]。

结果
在未指定参数时，FineDist显示当前设置值。

注意

- 支持的控制器型号

  不支持T/VT系列。

- 关于Fine和FineDist

  Fine与FineDist的不同之处为机器人动作结束时定位判断的单位。
  Fine可通过pulse设置定位判断值，分别对各轴进行定位判断。
  FineDist可设置以mm为单位的定位判断值，并通过工具编号为“0”的坐标系进行定位判断。
  不可同时使用Fine和FineDist。如果在程序中使用了Fine和FineDist(如下例)，将通过FineDist进行定位判断。
  (如果Fine与FineDist的顺序相反，将通过Fine进行定位判断。)

```
Function test
    Fine 5, 5, 5, 5
    FineDist 0.1

    Go P1
    Go P2
Fend
```

注意

- FineDist的初始化(依据Motor On、SLock、SFree等命令)

  如果使用下述任一命令，FineDist的设置值将被初始化为默认值，并通过Fine进行定位判断。
  SLock, SFree, Motor

  执行这些命令后，请务必重新设置FineDist值。

易引起的错误
如果没有在2秒内结束基于FineDist的定位，将出现错误代码4024。此错误一般意味着需要进行伺服系统的平衡调整。

参阅
Accel、AccelR、AccelS、Arc、Go、Jump、Move、Speed、SpeedR、SpeedS、Pulse、Fine、FineStatus

FineDist使用示例
下述程序为从程序函数执行FineDist的示例，以及从监视器窗口执行Fine的示例。

```
Function fineDisttest
    Fine 0.1  '将精度设为 +/-0.1 mm
    Go P1
    Go P2
Fend


> FineDist 0.1
>
> FineDist
0.1
```

3.10.12 FineStatus函数

对于正在使用Fine还是FineDist，以整数值返回。

格式

FineStatus

返回值

对于正在使用Fine还是FineDist，以整数值返回。

- 0 = 正在使用Fine
- 1 = 正在使用FineDist

参阅

Fine、FineDist

FineStatus函数使用示例

```
Print FineStatus
```

3.10.13 Fix函数

用于从实值中取出整数部分。

格式
Fix (数值)

参数

数值
　　指定实值。

返回值
将参数设置的实值转换为整数值并返回。

参阅
Int

Fix函数使用示例

```
>print Fix(1.123)
 1
>
```

3.10.14 Flush

用于将缓存写入到文件中。

格式
Flush #文件编号

参数

文件编号
以30～63之间的整数值或表达式进行指定。

说明
用于将缓存写入到现在打开的文件中。

无法用于以ROpen命令打开的文件。

Flush使用示例

```
Integer fileNum, i

fileNum = FreeFile
UOpen "TEST.DAT" As #fileNum
For i = 0 To 100
   Print #fileNum, i
Next i
Flush #fileNum
Close #fileNum
```

3.10.15 FmtStr

将数值表达式或日期/时间表达式格式化。

格式

FmtStr 格式指定表达式, 格式指定字符串, 输出字符串变量

参数

格式指定表达式
　　指定要进行格式指定的数值表达式或日期/时间表达式。
　　请以“yyyy/mm/dd hh:nn:ss”形式指定日期/时间表达式。
格式指定字符串
　　指定要进行格式指定的字符串。
输出字符串变量
　　指定输出字符串变量。

说明

按照格式数值字符串返回已设置格式的字符串。

数值格式指定符

None
　　未格式化，直接显示编号。
(0)
　　表示位位置的位表示符。表示数值或0。将数值表达式中包含的10进制数表示在格式指定字符串的“0”上。在其他情况下，在该位置上显示“0”。如果数值位数小于格式指定字符串的(在小数点的前或后)“0”的数，将显示开头或末尾的几个“0”。如果数值的小数点右侧的位数大于格式指定字符串的小数点右侧的“0”的个数，将按照“0”的数个数对数值位数进行取整。并且，如果数值的小数点左侧的位数大于格式指定字符串的小数点左侧的“0”的个数，将直接显示该多余的位。
(#)
　　表示位位置的位表示符。显示数值或什么也不显示。将数值表达式的10进制数显示在格式指定字符串的“#”位置上，但在其他情况下，该位置不显示任何内容。此字符 (#) 起0位表示符的作用，数值位数与格式指定字符串的小数点左右的“#”数相同，如果小于该数，将不显示开头或末尾的“0”。
(.)
　　表示小数点位置，并设置在小数点左右显示几位。根据本地情况，可能会用逗号表示小数点。如果格式指定字符串中，在该 (.) 的左侧没有只有数字，小于1的数字将以小数点为开头进行显示。要显示开头的0，需在小数点的左侧使用“0”。以格式指定格式并输出的值显示小数点的字符，取决于按照要所使用的Windows可识别的数值格式。
(,)
　　是用1000分隔数值的千位分隔符。根据本地情况，可使用句点。用于对小数点左面有4位以上的数值每隔3位数进行一次分隔一次小数点左面有位以上的数值。标准的使用方法是指定为格式指定字符串左右除包括伴随位表示符(0或#)外，还包含出现的千位分隔符 (,)。无论有无小数指定，如果小数点左侧靠右排列2个千位分隔符或者单独使用，则意味着表示格式指定方法为“将该数值除以1000，根据需要四舍五入”的格式指定。例如，格式指定字符串 ″##0,,″ 意味着将1亿表示为“100”。小于百万的数值将显示为“0”。如果在小数点左侧靠右之外的位置排列2个千位分隔符，则仅仅是用于分隔千位。以格式被指定格式的输出值实际用于千位分隔符的字符，按照要取决于所使用的Windows可识别的数值格式。

日期/时间格式指定符

(:)
　　时间分隔符。根据本地情况，可使用其他字符。在指定的时间值格式中，用于分隔时、分、秒的值。在指定格式输出中实际使用的字符，取决于Windows的设置。
(/)
　　日期分隔符。根据本地情况，可使用其他字符。在指定的日期值格式中，用于分隔日、月、年的值。在指定格式输出中实际使用的字符，取决于Windows的设置。

c
　　日期以ddddd形式，时间以tttt形式按此顺序显示。如果日期序列号中无分数部分，或时间信息中无整数部分，仅显示时间信息。
d
　　以开头不为0的形式显示日期。(1～31)
dd

以开头为0的形式显示日期。(01～31)

ddd
省略星期。(Sun～Sat)

dddd
显示星期。(Sunday～Saturday)

ddddd
按照Windows的短日期格式，显示日、月、年等所有信息。在Windows系统中，短日期格式的默认显示为m/d/yy。

dddddd
按照Windows的长日期格式，将日期的序列值显示为日、月、年。在Windows系统中，长日期格式的默认显示为mmmm dd, yyyy。

w
以数值形式显示星期。(1：星期日～7：星期六)

ww
以数值形式显示当前为1年中的第几个星期。(1～54)

m
以开头不为0的数值形式显示月份。(1～12)
即使在“h”或“hh”后使用，也不会作为分钟显示。请使用“n”或“nn”显示分钟。

mm
以开头为0的数值形式显示月份。(01～12)
即使在“h”或“hh”后使用，也不会作为分钟显示。请使用“n”或“nn”显示分钟。

mmm
省略月份。(Jan～Dec)

mmmm
显示月份。(January～December)

q
以数值形式显示季度。(1～4)

y
以数值形式显示当前为1年中的第几天。(1～366)

yy
以2位数显示年号。(00～99)

yyyy
以4位数显示年号。(100～9999)

h
以开头不为0的形式，显示24小时制的时间。(0～23)

hh
以开头为0的形式，显示24小时制的时间。(00～23)

n
以开头不为0的形式显示分钟。(0～59)

nn
以开头为0的形式显示分钟。(00～59)

s
以开头不为0的形式显示秒钟。(0～59)

ss
以开头为0的形式显示秒钟。(00～59)

t t t t t
按照Windows中设置的时间分隔符格式，显示时间(时、分、秒)。如果选择了“开头的0”选项，将以开头为0的形式显示10:00am/pm之前的时间。在Windows系统中，时间格式的默认显示为“h:nn:ss”。

AM/PM
以12小时制显示时间，并以“AM/PM”(大写字母)表示上午下午。

am/pm
以12小时制显示时间，并以“am/pm”(小写字母)表示上午下午。

A/P
以12小时制显示时间，并以“A/P”(大写字母)表示上午下午。

a/p
以12小时制显示时间，并以“a/p”(小写字母)表示上午下午。

AMPM
以12小时制显示时间。上午的时间为“AM”字符串，下午的时间为“PM”字符串，分别按照Windows内的格式设置显示。AM/PM可使用大写字母或小写字母，但Windows的设置与指定字符串必须一致。在Windows系统中，默认设置为AM/PM。

注意

- 数值格式指定符和日期/时间格式指定符同时存在

  如果同时输入数值格式指定符和日期/时间格式指定符，将会发生错误。

参阅

Left$、Right$、Str$

FmtStr使用示例

```
Function SaveData

    String d$, f$, t$

    ' 以月、日、时、分的形式创建文件名
    d$ = Date$
    t$ = Time$
    d$ = d$ + " " + t$
    FmtStr d$, "mmddhhnn", f$
    f$ = f$ + ".dat"
    WOpen f$ as #30
    Print #30, "data"
    Close #30
Fend
```

3.10.16 FmtStr$函数

将数值表达式格式化。

格式

FmtStr$ (格式指定表达式, 格式指定字符串)

参数

格式指定表达式
　　指定要进行格式指定的数值表达式或日期/时间表达式。
　　请以“yyyy/mm/dd hh:nn:ss”形式指定日期/时间表达式。
格式指定字符串
　　指定要进行格式指定的字符串。

返回值

返回要进行格式指定的字符串。

说明

按照格式数值字符串返回已设置格式的字符串。

数值格式指定符

None
　　未格式化，直接显示编号。
(0)
　　表示位位置的位表示符。表示数值或0。将数值表达式中包含的10进制数表示在格式指定字符串的“0”上。在其他情况下，在该位置上显示“0”。如果数值位数小于格式指定字符串的(在小数点的前或后)“0”的数，将显示开头或末尾的几个“0”。如果数值的小数点右侧的位数大于格式指定字符串的小数点右侧的“0”的个数，将按照“0”的数个数对数值位数进行取整。并且，如果数值的小数点左侧的位数大于格式指定字符串的小数点左侧的“0”的个数，将直接显示该多余的位。
(#)
　　表示位位置的位表示符。显示数值或什么也不显示。将数值表达式的10进制数显示在格式指定字符串的“#”位置上，但在其他情况下，该位置不显示任何内容。此字符 (#) 起0位表示符的作用，数值位数与格式指定字符串的小数点左右的“#”数相同，如果小于该数，将不显示开头或末尾的“0”。
(.)
　　表示小数点位置，并设置在小数点左右显示几位。根据本地情况，可能会用逗号表示小数点。如果格式指定字符串中，在该 (.) 的左侧没有只有数字，小于1的数字将以小数点为开头进行显示。要显示开头的0，需在小数点的左侧使用“0”。以格式指定格式并输出的值显示小数点的字符，取决于按照要所使用的Windows可识别的数值格式。
(,)
　　是用1000分隔数值的千位分隔符。根据本地情况，可使用句点。用于对小数点左面有4位以上的数值每隔3位数进行一次分隔一次小数点左面有位以上的数值。标准的使用方法是指定为格式指定字符串左右除包括伴随位表示符(0或#)外，还包含出现的千位分隔符 (,)。无论有无小数指定，如果小数点左侧靠右排列2个千位分隔符或者单独使用，则意味着表示格式指定方法为“将该数值除以1000，根据需要四舍五入”的格式指定。例如，格式指定字符串 "##0,," 意味着将1亿表示为“100”。小于百万的数值将显示为“0”。如果在小数点左侧靠右之外的位置排列2个千位分隔符，则仅仅是用于分隔千位。以格式被指定格式的输出值实际用于千位分隔符的字符，按照要取决于所使用的Windows可识别的数值格式。

日期/时间格式指定符

(:)
　　时间分隔符。根据本地情况，可使用其他字符。在指定的时间值格式中，用于分隔时、分、秒的值。在指定格式输出中实际使用的字符，取决于Windows的设置。
(/)
　　日期分隔符。根据本地情况，可使用其他字符。在指定的日期值格式中，用于分隔日、月、年的值。在指定格式输出中实际使用的字符，取决于Windows的设置。

c
　　日期以ddddd形式，时间以tttt形式按此顺序显示。如果日期序列号中无分数部分，或时间信息中无整数部分，仅显示时间信息。
d

以开头不为0的形式显示日期。(1～31)

dd
以开头为0的形式显示日期。(01～31)

ddd
省略星期。(Sun～Sat)

dddd
显示星期。(Sunday～Saturday)

ddddd
按照Windows的短日期格式，显示日、月、年等所有信息。在Windows系统中，短日期格式的默认显示为m/d/yy。

dddddd
按照Windows的长日期格式，将日期的序列值显示为日、月、年。在Windows系统中，长日期格式的默认显示为mmmm dd, yyyy。

w
以数值形式显示星期。(1：星期日～7：星期六)

ww
以数值形式显示当前为1年中的第几个星期。(1～54)

m
以开头不为0的数值形式显示月份。(1～12)
即使在“h”或“hh”后使用，也不会作为分钟显示。请使用“n”或“nn”显示分钟。

mm
以开头为0的数值形式显示月份。(01～12)
即使在“h”或“hh”后使用，也不会作为分钟显示。请使用“n”或“nn”显示分钟。

mmm
省略月份。(Jan～Dec)

mmmm
显示月份。(January～December)

q
以数值形式显示季度。(1～4)

y
以数值形式显示当前为1年中的第几天。(1～366)

yy
以2位数显示年号。(00～99)

yyyy
以4位数显示年号。(100～9999)

h
以开头不为0的形式，显示24小时制的时间。(0～23)

hh
以开头为0的形式，显示24小时制的时间。(00～23)

n
以开头不为0的形式显示分钟。(0～59)

nn
以开头为0的形式显示分钟。(00～59)

s
以开头不为0的形式显示秒钟。(0～59)

ss
以开头为0的形式显示秒钟。(00～59)

t t t t t
按照Windows中设置的时间分隔符格式，显示时间(时、分、秒)。如果选择了“开头的0”选项，将以开头为0的形式显示10:00am/pm之前的时间。在Windows系统中，时间格式的默认显示为“h:nn:ss”。

AM/PM
以12小时制显示时间，并以“AM/PM”(大写字母)表示上午下午。

am/pm
以12小时制显示时间，并以“am/pm”(小写字母)表示上午下午。

A/P
以12小时制显示时间，并以“A/P”(大写字母)表示上午下午。

a/p
以12小时制显示时间，并以“a/p”(小写字母)表示上午下午。

AMPM
以12小时制显示时间。上午的时间为“AM”字符串，下午的时间为“PM”字符串，分别按照Windows内的格式设置显示。AM/PM可使用大写字母或小写字母，但Windows的设置与指定字符串必须一致。在Windows系统中，默认设置为AM/PM。

注意

- 数值格式指定符和日期/时间格式指定符同时存在

  如果同时输入数值格式指定符和日期/时间格式指定符，将会发生错误。

参阅

Left$, Right$, Str$

FmtStr$使用示例

```
Function SendDateCode

    String d$, f$

    f$ = FmtStr$(10, "000.00")
    OpenCom #1
    Print #1, f$
    CloseCom #1
Fend
```

3.10.17 FolderExists函数

检查有无指定的文件夹。

格式

FolderExists (路径名)

参数

路径名
以字符串指定要检查的文件夹的路径名。包括盘符。有关路径的详细说明，请参阅ChDisk。

注意

可在PC硬盘时执行。

返回值

- 存在文件夹时：True
- 不存在文件夹时：False

参阅

FileExists、MkDir

FolderExists函数使用示例

```
If Not FolderExists("c:\TEST") Then
    MkDir "c:\TEST"
EndIf
```

3.10.18 For...Next

按照指定次数反复执行For...Next之间的一系列的语句。

格式

For 变量名 = 初始值 To 结束值 [Step增量值] 语句 Next [变量名]

参数

变量名
　　指定要反复代入数据的变量名。此变量一般是整数值，还可以定义为实值变量。
默认值
　　在指定的变量中以数值指定循环的最初代入数值。
结束值
　　指定表示循环结束的值。检查到该值，将结束For...Next循环，并继续执行Next命令的下一语句。
增量值
　　每次执行For...Next循环中的Next语句都指定一次增加变量的值。增量值也可以设为负数，此时，初始值必须大于结束值。如无增量的指定，将自动以“1”增量。可省略。
语句
　　如果是有效的SPEL+语句，均可插入到For...Next循环中。

说明

For...Next用于按照指定次数返回循环中的语句。循环的开头是For语句，结尾是Next语句。利用变量对循环内语句执行次数进行计数。

初始值为计数器的最初数值。如果正确地设置了结束值变量和增量值，则可以设置负的数值。

结束值为计数器的最终数值。一达到该值就结束循环，程序控制将转至Next命令的下一命令。

For语句的下一语句将在达到Next命令之后执行。计数器变量(变量名)仅按由增量值指定的值进行增量。如果未设置增量值，计数器将每次增减“1”。

然后，计数器变量(变量名)与最终数值进行比较。如果计数器变量小于或等于最终数值，将重新执行For命令的下一语句。如果计数器变量(变量名)大于最终数值，执行将分支到For...Next循环外，继续执行Next命令的下一命令。

注意

- 负的增量值

  如果增量值指定了负值，由于每次循环都将减少计数器变量的值，所以请设置初始值大于结束值。

- Next后的变量名可以省略

  可以省略Next后的变量名，但是在有嵌套时记入变量名，将易于了解结构。

- 从循环退出时的变量值不是结束值。

```
Function forsample
  Integer i
  For i = 0 To 3
  Next
  Print i  '显示4
Fend
```

- 不使用Exit For的情况下从嵌套结构退出循环时

  若重复执行使用Exit For以外的命令(Gosub、Goto、Call命令等)退出循环的程序时，会出现错误2020。如果要在循环中退出，请使用Exit For命令。

- 空无限循环和类似无限循环的处理，请尽可能与Wait一起使用

  空For...Next或类似处理，可能会影响您的系统，请尽可能避免使用。当控制器检测到无限空循环，并判断该处理影响系统，则会发出2556错误 (检测到过剩的循环)。

  在执行需要循环的演算，或等待I/O信号时，请在循环处理中执行Wait命令 (例如，Wait 0.1)，避免占用CPU。

参阅

Do...Loop

For...Next使用示例

```
Function fornext
    Integer counter
    For counter = 1 to 10
        Go Pctr
    Next counter

    For counter = 10 to 1 Step -1
        Go Pctr
    Next counter
Fend
```

3.10.19 FreeFile函数

返回当前未使用的文件编号并预约该编号。

格式

FreeFile

返回值

返回30～63的整数。

参阅

AOpen、BOpen、ROpen、UOpen、WOpen、Close

FreeFile函数使用示例

```
Integer fileNum, i, j

fileNum = FreeFile
WOpen "TEST.DAT" As #fileNum
For i = 0 To 100
    Print #fileNum, i
Next i
Close #fileNum

fileNum = FreeFile
ROpen "TEST.DAT" As #fileNum
For i = 0 to 100
    Input #fileNum, j
    Print "data = ", j
Next i
Close #fileNum
```

3.10.20 Function...Fend

以Function语句开始并以Fend语句结束的、可执行程序的最小单位是1个函数。

格式

Function 函数名 [(自变量列表)] [As型(函数)] 语句 Fend

参数

函数名称
　　对以Function开始并以Fend命令结束的语句组，将名称指定在64个字母以内。还可以使用下划线。
自变量列表
　　指定调用时赋予函数自变量的变量列表。如果是多个变量，用逗号进行分隔。可省略。
　　自变量的格式如下。

　　[{ByRef | ByVal}] 变量名 [( )] As 型(自变量)

　　ByRef
　　　　参照要调用的函数的变量时，指定ByRef。此时，可以将函数内的自变量的变更反映到调用的变量中。可省略。
　　ByVal
　　　　是默认设置。如不变更调用函数参照的变量的值，可指定ByVal。可省略。
　　变量名[()]
　　　　自变量的变量名，是必要的参数。请按照变量名的命名方法的规则执行。如要将数组变量用作自变量，请务必指定ByRef，并在变量后面添加空括号“()”以表示数组。
　　As类型(自变量)
　　　　是必要的参数。请声明自变量的类型。

As类型(函数)
　　是要获取返回值时附加的参数。请声明返回值的类型。

返回值

是在函数声明的最后由As指定的数据类型(As类型(函数))。

说明

Function语句表示SPEL+语句组的开始。用Fend语句表示1个函数的结束。Function与Fend语句之间的所有语句都被认为是该函数的一部分。

Function与Fend、包括它们之间的所有语句，其本身是一个函数，可以认为是容器那样的语句段。还可以在一个程序文件中使用多个函数。

要使用返回值时，请将数值代入到与函数名同名的变量中，然后结束函数。

参阅

Call、Fend、Halt、Quit、Return、Xqt

Function...Fend使用示例

[例1] 下例为1个文件中包括3个函数的示例。task2和task3被main调用并与main一起同时执行。

```
Function main
  Xqt 2, task2 '同时执行任务2
  Xqt 3, task3 '同时执行任务3
  .
  .
  .
Fend

Function task2
  Do
    On 1
    On 2
```

```
    Off 1
    Off 2
  Loop
Fend

Function task3
  Do
    On 10
    Wait 1
    Off 10
  Loop
Fend
```

[例2] 如下所示为将外围装置的压力控制序列设为自变量、将发送到外部装置时的结果设为返回值并在画面上显示时的函数示例。

```
Function main
  Integer iResult
  Real Sequence1(200)
  .
  .
  iResult = PressureControl(ByRef Sequence1())    '自变量为数组
  .
  Print "Result:", iResult
  .
Fend

Function PressureControl(ByRef Array1() As Real) As Integer
  .
  (根据Array1数组对外围装置进行压力控制)
  .
  PressureControl = 3     '返回值
  .
  .
Fend
```

3.11 G

3.11.1 GetCurrentUser$函数

用于返回当前的Epson RC+用户。

格式
GetCurrentUser$

返回值
用于返回当前的Epson RC+用户的登录ID。

注意

此命令仅在安全选件有效时可使用。

参阅
LogIn

GetCurrentUser$函数使用示例

```
String currUser$

currUser$ = GetCurrentUser$
```

3.11.2 GetRobotInsideBox函数

用于返回进入到进入检测区域内的机器人。

格式

GetRobotInsideBox (区域编号)

参数

区域编号
　　指定返回状态的进入检测区域编号(1～15的整数)。

返回值

以位为单位返回进入由区域编号指定的进入检测区域中的机器人。

位0表示机器人1，按降序以下顺延，位15表示机器人16。

如果机器人未设置进入检测区域，则相应位通常为为常0。

例如，在机器人1和3进入区域时，打开位0和位2，所以返回5。

参阅

Box、InsideBox

GetRobotInsideBox函数使用示例

如下所示为使用GetRobotInsideBox函数的程序。

等待达到进入检测区域中未进入1台机器人的状态。

```
Function WaitNoBox

  Wait GetRobotInsideBox(1) = 0
```

等待达到只有2个机器人在进入检测区域中的状态。

```
Function WaitInBoxRobot2

  Wait GetRobotInsideBox(1) = &H2
```

下述程序为在动作命令的并行处理内使用的示例。在动作执行过程中进入到特定进入检测区域时，打开I/O。有1台机器人连接到控制器上的情况。

```
Function Main
   Motor On
   Power High
   Speed 30; Accel 30, 30

   Go P1 !D0; Wait GetRobotInsideBox(1) = 1; On 1!

Fend
```

注意

请务必记述D0。

3.11.3 GetRobotInsidePlane函数

用于返回进入到进入检测平面内的机器人。

格式

GetRobotInsidePlane (平面编号)

参数

平面编号
　　指定返回状态的进入检测平面的编号(1～15的整数)。

返回值

以位为单位返回进入由平面编号指定的进入检测平面中的机器人。

位0表示机器人1，按降序依次顺延，位15表示机器人16。

如果机器人未设置进入检测平面，则相应位为常通常为0。

例如，在机器人1和3进入平面时，打开位0和位2，所以返回5。

参阅

InsidePlane、Plane

GetRobotInsidePlane函数使用示例

如下所示为使用GetRobotInsidePlane函数的程序。

等待达到进入检测平面中未进入1台机器人的状态。

```
Function WaitNoPlane

  Wait GetRobotInsidePlane(1) = 0
```

等待达到只有2个机器人进入检测平面的状态。

```
Function WaitInPlaneRobot2

  Wait GetRobotInsidePlane(1) = &H2
```

下述程序为在动作命令的并行处理内使用的示例。在动作执行过程中进入到特定进入检测平面时，打开I/O。有1台机器人连接到控制器上的情况。

```
Function Main
   Motor On
   Power High
   Speed 30; Accel 30, 30

   Go P1 !D0; Wait GetRobotInsidePlane(1) = 1; On 1!

Fend
```

注意

请务必记述D0。

3.11.4 Global

进行全局变量的说明。从哪里都可以访问全局变量。

格式

Global [Preserve] 数据类型变量名 [(数组变量的最大下标)] [, 变量名 [(数组变量的最大下标)]...]

参数

Preserve
　　如果指定Preserve，将保持变量的值。如果更改项目，将清除此值。可省略。
数据类型
　　指定Boolean、Byte、Double、Int32、Integer、Long、Real、Short、String、UByte、UInt32、UShort之一的数据类型。
变量名
　　将变量名指定在32个字符以内。
数组变量的最大下标
　　可利用数组变量的最大下标声明到最高三维。使用下述格式。可省略。
　　(最大下标1, [最大下标2], [最大下标3])
　　由于下标从0开始，因此数组元素数为最大下标加上1。
　　如果是全局变量，所有数组元素数最大是100,000。但是，String型最大是10,000。
　　如果是备份变量(Global Preserve)，所有数组元素数最大是4,000。但是，String型最大是400。
　　在所有数组元素数不超过以下最大值的范围内指定各最大下标。

说明

全局变量如果在同一项目内，就是可以由多个文件共享的变量。只要未被Preserve选项声明，每次从运行窗口和操作员窗口起动函数时，该变量都将被清除。

如果被Preserve选项声明，即使关闭控制器电源，也将保持其值。

保存的全局变量还可以在RC+ API选项中使用。

请按照下例所示，全局变量名以词头"g_" 为开始，以易于在程序中识别。

```
Global Long g_PartsCount
```

参阅

Boolean、Byte、Double、Int32、Int64、Integer、Long、Real、Short、String、UByte、UInt32、UInt64、UShort

Global使用示例

下述为2个不同的程序文件。在最初的文件中，定义并初始化了几个全局变量。在下一个文件中使用这些全局变量。

```
FILE1 (MAIN.PRG)
Global Integer g_Status
Global Real g_MaxValue

Function Main

    g_Status = 10
    g_MaxValue = 1.1
    .
    .
Fend

Function Test

    Print "status1 =" , g_Status
    Print "MaxValue =" , g_MaxValue
    .
```

```
    .
Fend
```

3.11.5 Go

用于在当前位置～指定位置之间以PTP动作移动机械臂。

格式

Go目标坐标 [CP] [LJM [选择姿势标志]] [Till | Find] [!并行处理!] [SYNC]

参数

目标坐标
　　以点数据指定目标位置。
CP
　　指定路径运动。可省略。
LJM
　　利用LJM函数转换目标坐标。可省略。
选择姿势标志
　　指定赋予LJM函数的姿势标志选择参数。可省略。
Till | Find
　　记述Till或Find表达式。可省略。

```
Till | Find
Till Sw(表达式) = {On | Off}
Find Sw(表达式) = {On | Off}
```

! 并行处理 !
　　动作期间可附加并行处理语句，以执行I/O等命令。可省略。
SYNC
　　预约动作命令。在通过SyncRobots开始动作之前，机器人不进行动作。

说明

Go用于以PTP动作将机器人机械臂的所有关节同时移动。下例为对Go命令指定目标坐标的示例。

- 以特定点编号进行指定。例: `Go P1`
- 以具体坐标值指定移动目标。例: `Go XY(50, 400, 0, 0)`
- 以点编号＋偏移值进行指定。例: `Go P1 +X(50)`
- 明示并指定点编号＋与它不同的坐标值。例: `Go P1 :X(50)`

动作的路径由于各关节都分别内插到当前点到目标坐标之间，所以无法预测轨迹。请充分注意与周围设备有无干扰。

Go命令的动作速度通过Speed命令来设置。加减速度通过Accel命令来设置。

如果附加了CP参数，则可在开始动作开始减速时叠加后续动作命令的加速。此时，不对目标坐标进行定位。

附加LJM参数时，以当前位置为参照位置，并以目标位置为利用LJM函数转换的位置。

可以将Go LJM(P1, Here,1)简化为Go P1 LJM 1进行记述。

此时，无法变更点数据P1。

LJM参数对于垂直6轴型机器人(包括N系列)与RS系列机器人有效。

以默认值使用姿势标志选择时，可省略。

```
  Go P1 LJM
```

注意

- Go与Move的差异

Move 和 Go都是使机器人机械臂动作的命令。两者之间的最大不同是Go是进行PTP动作，而Move是在直线轨道上移动机械臂。在重视到达目标点时的机械臂的姿势时，使用Go命令；而更重视控制动作中的机械臂的轨迹时，使用Move命令。

- Go与Jump的差异

  Jump 和 Go都是以PTP动作移动机器人机械臂的命令。但是，Jump拥有一个Go所没有的功能。Jump首先将机器人的夹具末端抬起到LimZ值，然后水平移动机械臂，在达到目标坐标的上空时开始下降动作。这种移动的优点是可以切实地避开障碍物，更重要的是通过吸附和配置动作可以提高作业的循环时间。

- 向Go发出适当的速度和加减速度指示

  Go命令的动作速度和加减速度的设置可以通过Speed 和Accel命令实施。Speed和Accel命令与Go命令相同，均可对PTP动作进行设置，此点是至关重要的。可以通过SpeedS和AccelS命令设置直线以及曲线动作的速度和加减速度。

- 使用Till修饰符的Go的用法

  通过使用Till修饰符，可以在到达Go命令指定的目标坐标之前，在到达此处的通过点上设置使机器人减速停止的条件。如果Till条件未成立，机器人将直接移至目标坐标。使用Till的Go的用法有以下2点。

  - (1) Go与Till修饰符

    检查当前的Till条件是否成立。如果成立，可通过在中途通过点上使机器人减速停止，来结束命令的执行，而不必等待Go完成指定动作。

  - (2) Go与Till修饰符、Sw(输入位编号)修饰符、输入条件

    在该用法中，可以在与Go命令相同的行中重新设置Till条件，而不使用预先设置的Till的现有设置条件。设置的条件就是检查1个输入位，为此需要使用Sw命令。检查输入状态是ON还是OFF，可以根据设置条件停止机械臂。此功能与如果输入条件成立则停止动作的“中断”的作用几乎相同。如果输入条件在机器人动作过程中一次也没有成立，则机械臂将到达指定的目标坐标位置。

- 使用Find修饰符的Go的用法

  通过使用Find修饰符，可以在Go命令的动作中为机器人设置记录一个位置的条件。使用Find的Go的用法有以下2点。

  - (1) Go与Find修饰符

    检查当前的Find条件是否成立。如果成立，将当前位置保存到特殊点、FindPos中。

  - (2) Go与Find修饰符、Sw(输入位编号)修饰符、输入条件

    在该用法中，可以在与Go命令相同的行中重新设置Find条件，而不使用预先设置的Find的现有设置条件。设置条件可以检查1个输入位。为此就需要使用Sw命令。检查输入状态是ON还是OFF，并将当前位置保存在特殊点和FindPos中。

- 在Go命令中，在停止前必须减速。

  在Go命令中，在将机械臂停止在动作的目标坐标之前，必须减速。

易引起的错误

- 想要使机器人在移动范围外动作时

  以直接坐标设置目标坐标时，请务必确认该坐标位置是否在机器人的移动范围内。如果指定在机器人的移动范围外，将发生错误。

参阅

!并行处理!、Accel、Find、Jump、Move、Pass、P#=指定点、Pulse、Speed、Sw、Till

Go使用示例

如下所示为在P0～P1之间执行PTP动作后以直线移动返回到P0的简单示例。在程序的后半段，机械臂向P2进行直线移动，直至输入位2变为ON状态。如果Move期间输入位2变为ON状态，机械臂则进行减速停止(即使没有到达P2)，然后执行下一程序命令。

```
Function sample
  Integer i

  Home
  Go P0
  Go P1
  For i = 1 to 10
    Go P(i)
  Next i
  Go P2 Till Sw(2) = On
  If Sw(2) = On Then
    Print "Input #2 came on during the move and"
    Print "the robot stopped prior to arriving on"
    Print "point P2."
  Else
    Print "The move to P2 completed successfully."
    Print "Input #2 never came on during the move."
  EndIf
Fend
```

如下所示为利用命令窗口的操作示例。

```
>Go Here +X(50)                  '从当前位置向X方向移动50 mm
>Go P1                           '向P1移动
>Go P1 :U(30)                    '向P1且U=30的位置移动
>Go P1 /L                        '以左腕姿势向P1移动
>Go XY(50, 450, 0, 30)           '向X=50、Y=450、Z=0、U=30的位置移动
```

[其他示例]

```
Till Sw(1) = Off And Sw(2) = On '作为Till条件指定输入1和2
Go P1 Till                      '满足预先定义的Till条件时停止
Go P2 Till Sw(2) = On           '输入位2变为ON时停止
Go P3 Till                      '满足预先定义的Till条件时停止
```

3.11.6 GoSub...Return

GoSub用于将程序控制移交给子例程。如果结束执行子例程，控制将返回至GoSub命令的下一行。

格式

GoSub {标签}

{标签：} 语句 Return

参数

标签

以标签指定移动目标。程序执行将转至有该标签的行。标签名请指定在32字符以内。但是，开头字符不能使用数字。请务必使用字母。

说明

GoSub命令用于将程序控制转至指定的标签。程序将执行移动目标语句，并通过Return命令直接执行移动目标的行。
GoSub命令在执行子例程后通过Return返回GoSub命令的下一行。子例程请务必用Return结束。

易引起的错误

- 转至不存在的语句

  如果将不存在GoSub命令的标签指定为移动目标，就会发生错误3108。

- 没有GoSub的情况下使用Return时

  Return命令用于返回执行GoSub命令的原来的程序。如果在没有GoSub的情况下使用Return，则会发生错误2383。
  没有GoSub的情况下使用Return，不知道会返回何处，所以没有意义。

参阅

GoTo、OnErr、Return

GoSub使用示例

下例为使用GoSub命令转至指定标签并执行几个I/O命令后进行返回的简单示例。

```
Function main
    Integer var1, var2

    GoSub checkio  '使用标签执行GoSub
    On 1
    On 2
    Exit Function

checkio: '子例程的起始位置
    var1 = In(0)
    var2 = In(1)
    If var1 = 1 And var2 = 1 Then
        On 1
    Else
        Off 1
    EndIf
    Return          '子例程的结束位置
Fend
```

3.11.7 GoTo

GoTo命令用于将程序控制转至指定的标签。

格式

GoTo {标签}

参数

标签

程序执行转至有标签的行。标签最多指定32个字符，开头字符不能使用数字。请务必使用字母。

说明

GoTo命令用于将程序控制转至指定的标签。程序将执行移动目标的语句，并执行该行以后的语句。

注意

- 过多使用GoTo时

  如果在一个程序中过多地使用GoTo命令，程序将难于理解，请注意。一般请尽量不使用GoTo命令。实际上，有无论如何也要使用GoTo的情况，将源代码四处转移的使用GoTo语句的方法，会导致发生错误或其他问题。

参阅

GoSub、OnErr

GoTo使用示例

下例为使用GoTo命令将控制转至行标签的简单的程序例。

```
Function main

    If Sw(1) = Off Then
        GoTo mainAbort
    EndIf
    Print "Input 1 was On, continuing cycle"
    .
    .
    Exit Function

mainAbort:
    Print "Input 1 was OFF, cycle aborted!"
Fend
```

3.12 H

3.12.1 Halt

暂停指定的正在执行的任务。

格式
Halt 任务识别符

参数

任务识别符
以整数值或表达式指定任务名或任务编号。
任务名为Xqt语句使用的函数名，或通过运行窗口或操作员窗口启动的函数名。
任务编号的指定(整数)
- 一般任务：1~32
- 后台任务：65～80
- Trap任务：257~267

说明
Halt用于暂停通过任务名或任务编号指定的正在执行的任务。

从停止位置进行恢复时，使用Resume。完全结束任务的执行时，使用Quit。要显示任务状态，单击Epson RC+工具栏上的任务管理器图标，启动任务管理器。

在指定的任务为NoPause任务、NoEmgAbort任务(执行Xqt时指定NoPause或NoEmgAbort以开始的特别任务)、Trap任务以及后台任务时，Halt也用于暂停任务的执行。但是，要暂停这些任务，需要进行充分的研究。一般情况下，建议对特殊任务不执行Halt。

参阅
Quit、Resume、Xqt

Halt使用示例
下例为通过Xqt启动名为“flicker”的函数后，通过Halt暂停，然后通过Resume恢复的示例。

```
Function main
  Xqt flicker         '执行flicker任务

  Do
    Wait 3            '执行flicke任务3秒钟
    Halt flicker

    Wait 3            '暂停flicker任务3秒钟
    Resume flicker

  Loop
Fend

Function flicker
  Do
    On 1
    Wait 0.2
    Off 1
    Wait 0.2
  Loop
Fend
```

3.12.2 Hand

用于设置点的手臂(手腕系)姿势。

格式

(1) Hand 指定点 [, Lefty | Righty]

(2) Hand

参数

点指定
　　以P编号、P(表达式)、点标签之一进行指定。
Lefty | Righty
　　指定末端夹具(手腕系)姿势。

结果

如果省略2个参数，则显示机器人当前位置的手臂(手腕系)姿势。

Lefty | Righty参数，将显示指定点的手臂(手腕系)姿势。

注意

Hand命令，不是用于控制末端夹具的命令。

- Hand (本命令)：指定机械臂的手腕系是左手腕系或右手腕系。|
- Hand_On, Hand_Off, Hand_On函数, Hand_Off函数, Hand_TW函数, Hand_Def函数, Hand_Type函数, Hand_Label$函数, Hand_Number函数：控制安装在机械臂上的末端夹具。

请注意不要混淆。

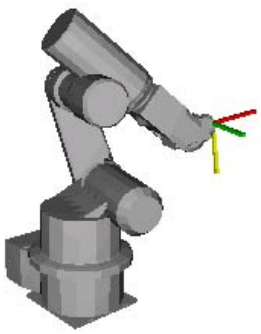
有关夹具控制命令的详细信息，请参阅以下手册。
《Hand功能》

参阅

Elbow、Hand函数、J4Flag、J6Flag、Wrist、J1Flag、J2Flag

Hand使用示例

```
Hand P0, Lefty
Hand pick, Righty
Hand P(myPoint), myHand

P1 = -364.474, 120.952, 469.384, 72.414, 1.125, -79.991
```



```
Hand P1, Righty
Go P1
```

```
Hand P1, Lefty
Go P1
```

3.12.3 Hand函数

用于返回点的手臂姿势。

格式

Hand [(指定点)]

参数

点指定
以表达式指定点。可省略。如果省略，将返回机器人当前位置的手臂姿势。

返回值

- 1: Righty (/R)
- 2: Lefty (/L)

注意

Hand命令，不是用于控制末端夹具的命令。

- Hand函数(本函数)：指定机械臂的手腕系是左手腕系或右手腕系。
- Hand_On, Hand_Off, Hand_On函数, Hand_Off函数, Hand_TW函数, Hand_Def函数, Hand_Type函数, Hand_Label$函数, Hand_Number函数：控制安装在机械臂上的末端夹具。

请注意不要混淆。

有关夹具控制命令的详细信息，请参阅以下手册。
《Hand功能》

参阅

Elbow、Wrist、J4Flag、J6Flag、J1Flag、J2Flag

Hand函数使用示例

```
Print Hand(pick)
Print Hand(P1)
Print Hand
Print Hand(P1 + P2)
```

3.12.4 HealthCalcPeriod

用于设置和显示部件消耗管理信息中的“剩余月数”的运算期间。

格式
(1) HealthCalcPeriod 期间设置值

(2) HealthCalcPeriod

参数

期间设置值
以整数值(1～7)指定要进行运算的期间。(单位：天) 默认值为“7”。

结果
如果省略参数，则显示当前的HealthCalcPeriod设置值。

说明
根据过去的动作状况，定期对部件消耗管理信息中的剩余月数进行运算。HealthCalcPeriod命令用于设置和显示该运算使用的动作期间。

如果延长期间，将进行“剩余月数”的运算，以控制因期间内的变动而导致的偏差影响。但是，如果变更了动作或速度，要正确显示“剩余月数”将需要更长的时间。

HealthCalcPeriod的设置值适用于通过已执行的控制器控制的所有机器人、关节与部件。

注意

- 支持的控制器型号

  不支持T/VT系列。T/VT 系列始终使用1(天)。

- 设置期间

  利用HealthCalcPeriod命令设置的期间为控制器的启动期间。

  与实际时间不同，敬请注意。

- 清除“剩余月数”与“消耗率”时的运算

  通过Epson RC+的“部件消耗管理信息”、HealthCtrlReset或HealthRBReset，在清除“剩余月数”与“消耗率”之后到达最初设置的期间之内，每隔1天进行一次剩余月数运算，而与HealthCalcPeriod的设置值无关。该期间的剩余月数偏差会增大。

参阅
HealthCalcPeriod函数、HealthCtrlInfo、HealthRBInfo、HealthCtrlReset、HealthRBReset

HealthCalcPeriod使用示例

```
> HealthCalcPeriod 3
> HealthCalcPeriod
 3
```

3.12.5 HealthCalcPeriod函数

用于返回当前设置的部件消耗管理信息中的“剩余月数”的运算期间。

格式
HealthCalcPeriod

返回值
以整数值返回要进行运算的期间。(单位：天)

注意

- 支持的控制器型号

  不支持T/VT系列。

参阅
HealthCalcPeriod

HealthCalcPeriod函数使用示例
如下所示为运算期间的显示示例。

```
Print "period is", HealthCalcPeriod
```

3.12.6 HealthCtrlAlarmOn函数

是用于返回控制器相关指定类别部件的部件消耗报警状态。

格式
HealthCtrlAlarmOn (部件类别)

参数

部件类别
以整数值(1)或下述常数指定返回报警状态的部件。

常数	值	模式
HEALTH_CONTROLLER_TYPE_BATTERY	1	指定电池。

返回值
如果指定的部件发生部件消耗报警，则会返回“True”；如果未发生，则返回“False”。

在通过HealthRateCtrlInfo获取的部件消耗率超出100%时，将发生部件消耗报警。

参阅
HealthCtrlInfo, HealthRateCtrlInfo

HealthCtrlAlarmOn函数使用示例
如下所示为判断是否发生控制器电池部件消耗报警的示例。

```
Function PrintAlarm
  If HealthCtrlAlarmOn(HEALTH_CONTROLLER_TYPE_BATTERY) = True Then
    Print "Controller Battery NG"
  Else
    Print "Controller Battery OK"
  EndIf
Fend
```

3.12.7 HealthCtrlInfo

用于显示控制器相关指定类别的部件的推荐更换期限之前的剩余月数。

格式

HealthCtrlInfo 部件类别

参数

部件类别

以整数值(1)或以下所示常数指定要返回的推荐更换期限之前剩余月数的部件。

常数	值	模式
HEALTH_CONTROLLER_TYPE_BATTERY	1	指定电池。

说明

用于显示控制器相关指定类别的部件的推荐更换期限之前的剩余月数。

根据基于过去使用状况的部件消耗率以及按通过HealthCalcPeriod设置的期间运转控制器获取的消耗率变化量，对推荐更换期限之前的月数进行运算。

注意

由于是根据按通过HealthCalcPeriod设置的期间运转控制器获取的消耗率变化量，对推荐更换期限之前的剩余月数进行运算，因此下述情况时，无法正常进行运算。

- 在运转时间未达到由HealthCalcPeriod设置的期间的状况下执行本命令时。
- 结束长期机器人运转停止期间之后执行本命令时。
- 更换部件并对部件消耗进行重置之后执行本命令时。
- 变更控制器的日期设置时。

在这种情况下，如果在对控制器进行HealthCalcPeriod设置期间2倍以上的运转之后执行本命令，将显示正确的值。

参阅

HealthCtrlAlarmOn, HealthRateCtrlInfo

HealthCtrlInfo使用示例

如下所示为控制器电池推荐更换期限之前的剩余月数的显示示例。

```
> HealthCtrlInfo HEALTH_CONTROLLER_TYPE_BATTERY
BATTERY            240.000
>
```

3.12.8 HealthCtrlInfo函数

是用于返回控制器相关指定类别的部件的推荐更换期限之前的剩余月数的函数。

格式

HealthCtrlInfo (部件类别)

参数

部件类别

以整数值(1)或以下所示常数指定要返回的推荐更换期限之前剩余月数的部件。

常数	值	模式
HEALTH_CONTROLLER_TYPE_BATTERY	1	指定电池。

返回值

以实数返回推荐更换期限之前的剩余月数(单位：月)。

说明

根据基于过去使用状况的部件消耗率以及按通过HealthCalcPeriod设置的期间运转控制器获取的消耗率变化量，对推荐更换期限之前的月数进行运算。

注意

由于是根据按通过HealthCalcPeriod设置的期间运转控制器获取的消耗率变化量，对推荐更换期限之前的剩余月数进行运算，因此下述情况时，无法正常进行运算。

- 在运转时间未达到由HealthCalcPeriod设置的期间的状况下执行本命令时。
- 结束长期机器人运转停止期间之后执行本命令时。
- 更换部件并对部件消耗进行重置之后执行本命令时。
- 变更控制器的日期设置时。

在这种情况下，如果在对控制器进行HealthCalcPeriod设置期间2倍以上的运转之后执行本命令，将显示正确的值。

参阅

HealthCtrlAlarmOn, HealthRateCtrlInfo

HealthCtrlInfo函数使用示例

如下所示为推荐更换期限之前的剩余月数为1个月以下时输出报警的示例。

```
Function AlarmCheck
  Real month

  month = HealthCtrlInfo(HEALTH_CONTROLLER_TYPE_BATTERY)
  If month < 1 Then
    Print "Alarm ON"
  EndIf
Fend
```

3.12.9 HealthCtrlRateOffset

对指定部件类别的消耗率设置偏移值。

格式

HealthCtrlRateOffset 部件类别, 偏移值

参数

部件类别

以整数值(1)或下述常数指定控制器相关部件。

常数	值	模式
HEALTH_CONTROLLER_TYPE_BATTERY	1	指定电池。

偏移值

指定对消耗率进行偏移的整数值。(单位：%)

说明

对指定部件类别设置消耗率的偏移值。

参阅

HealthRBAlarmOn, HealthRateRBInfo, HealthRBInfo

HealthCtrlRateOffset使用示例

如下所示为在控制器电池消耗率上加上10%的示例。

```
> HealthCtrlRateOffset HEALTH_CONTROLLER_TYPE_BATTERY,10
>
```

3.12.10 HealthCtrlReset

用于清除指定部件类别的部件的推荐更换期限之前的剩余月数与消耗率。

格式

HealthCtrlReset 部件类别

参数

部件类别

以整数值(1)或下述常数指定控制器相关部件。

常数	值	模式
HEALTH_CONTROLLER_TYPE_BATTERY	1	指定电池。

说明

针对指定部件类别，清除推荐更换期限之前的剩余月数与消耗率。也同时解除警告。

参阅

HealthCtrlAlarmOn、HealthRateCtrlInfo、HealthCtrlInfo

HealthCtrlReset使用示例

```
> HealthCtrlReset HEALTH_CONTROLLER_TYPE_BATTERY
>
```

3.12.11 HealthCtrlWarningEnable

用于将控制器相关指定类别部件的部件消耗报警通知设为有效或无效。

格式

HealthCtrlWarningEnable 部件类别[, On/Off]

参数

部件类别

以整数值或下述常数指定控制器的部件。

常数	值	模式
HEALTH_CONTROLLER_TYPE_BATTERY	1	指定电池。

On/Off

- On：将部件消耗报警通知设为有效。- Off：将部件消耗报警通知设为无效。

结果

如果省略On/Off参数，将显示当前的On/Off设置值。

说明

发生指定类别部件的部件消耗报警时，设置是否通知部件消耗报警。

注意

如果已将指定部件的部件消耗报警通知设为无效，即使到了推荐更换期限，也不会发出部件消耗报警通知。执行时，请充分注意依据本命令的设置。

参阅

HealthCtrlAlarmOn

HealthCtrlWarningEnable使用示例

如下所示为将控制器电池的部件消耗报警通知设为无效的示例。

```
> HealthCtrlWarningEnable HEALTH_CONTROLLER_TYPE_BATTERY, Off
```

如下所示为控制器电池的部件消耗报警通知设置的显示示例。

```
> HealthCtrlWarningEnable HEALTH_CONTROLLER_TYPE_BATTERY
BATTERY  Off
>
```

3.12.12 HealthCtrlWarningEnable函数

是用于返回控制器相关指定类别部件的部件消耗报警通知设置状态的函数。

格式

HealthCtrlWarningEnable (部件类别)

参数

部件类别

以整数值或下述常数指定控制器的部件。

常数	值	模式
HEALTH_CONTROLLER_TYPE_BATTERY	1	指定电池。

返回值

以整数值返回部件消耗报警的设置值。

- 1: On
- 0: Off

参阅

HealthCtrlAlarmOn

HealthCtrlWarningEnable函数使用示例

如下所示为控制器电池的部件消耗报警通知设置状态的显示示例。

```
Print HealthCtrlWarningEnable(HEALTH_CONTROLLER_TYPE_BATTERY )
```

3.12.13 HealthRateCtrlInfo函数

是用于返回控制器相关指定类别部件的部件消耗率的函数。

格式
HealthRateCtrlInfo (部件类别)

参数

部件类别
以整数值(1)或以下所示常数指定要返回的推荐更换期限之前剩余月数的部件。

常数	值	模式
HEALTH_CONTROLLER_TYPE_BATTERY	1	指定电池。

返回值
以实数返回将推荐更换期限设为100%时的当前部件消耗率(单位：%)。

说明
根据实际运转状况数据对部件消耗率进行运算。

注意

推荐更换期限是根据统计推荐更换部件的时期。
即使部件消耗率未达到100%，也可能需要进行更换。
另外，即使达到100 %，也并不意味着马上不能使用。
但是，达到100%以后的故障发生可能性将增加，因此建议尽早更换。

参阅
HealthCtrlAlarmOn, HealthCtrlInfo,

HealthRateCtrlInfo函数使用示例
如下所示为控制器电池部件消耗率达到90%以上时输出报警的示例。

```
Function AlarmCheck
  Real HealthRate

  HealthRate = HealthRateCtrlInfo (HEALTH_CONTROLLER_TYPE_BATTERY)
  If HealthRate > 90 Then
    Print "Alarm ON"
  EndIf
Fend
```

3.12.14 HealthRateRBInfo函数

是用于返回机器人相关指定类别部件的部件消耗率的函数。

格式

HealthRateRBInfo (机器人编号,部件类别,关节编号)

参数

机器人编号
以整数值(1-16)指定要返回部件消耗率的机器人编号。
部件类别
以整数值(1-6)或下述常数指定要返回部件消耗率的部件。

常数	值	模式
HEALTH_ROBOT_TYPE_BATTERY	1	指定电池。
HEALTH_ROBOT_TYPE_BELT	2	指定皮带。
HEALTH_ROBOT_TYPE_GREASE	3	指定润滑脂。
HEALTH_ROBOT_TYPE_MOTOR	4	指定电动机。
HEALTH_ROBOT_TYPE_GEAR	5	指定减速机。
HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE	6	指定滚珠丝杠花键。

关节编号
以整数值(1-9)指定要返回部件消耗率的关节。本命令无法用于附加轴。

返回值

以实数返回将推荐更换期限设为100%时的当前部件消耗率(单位：%)。

如果机器人关节未使用指定部件，则返回-1。

说明

根据实际运转状况数据对部件消耗率进行运算。

注意

推荐更换期限是根据统计推荐更换部件的时期。
即使部件消耗率未达到100%，也可能需要进行更换。
另外，即使达到100 %，也并不意味着马上不能使用。
但是，达到100%以后的故障发生可能性将增加，因此建议尽早更换。

参阅

HealthRBAlarmOn, HealthRBInfo

HealthRateRBInfo函数使用示例

如下所示为机器人1第3关节减速机的部件消耗率达到90%以上时输出报警的示例。

```
Function AlarmCheck
Real HealthRate

HealthRate = HealthRateRBInfo(1, HEALTH_ROBOT_TYPE_GEAR,3)
If HealthRate > 90 Then
Print "Alarm ON"
EndIf
Fend
```

3.12.15 HealthRBAlarmOn函数

是用于返回机器人相关指定类别部件的部件消耗报警状态的函数。

格式

HealthRBAlarmOn (机器人编号, 部件类别, 关节编号)

参数

机器人编号
以整数值(1-16)指定要返回报警状态的机器人编号。

部件类别
以整数值(1-6)或下述常数指定要返回报警状态的部件。

常数	值	模式
HEALTH_ROBOT_TYPE_BATTERY	1	指定电池。
HEALTH_ROBOT_TYPE_BELT	2	指定皮带。
HEALTH_ROBOT_TYPE_GREASE	3	指定润滑脂。
HEALTH_ROBOT_TYPE_MOTOR	4	指定电动机。
HEALTH_ROBOT_TYPE_GEAR	5	指定减速机。
HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE	6	指定滚珠丝杠花键。

关节编号
以整数值(1-9)指定要返回报警状态的关节。在部件类别中选择了电池时，因为电池在所有的关节中通用，指定任一关节均返回相同的值。本命令无法用于附加轴。

返回值

如果指定的部件发生部件消耗报警，则会返回“True”；如果未发生，则返回“False”。

在通过HealthRateRBInfo获取的部件消耗率超出100%时，将发生部件消耗报警。

如果机器人关节未使用指定部件，则返回-1。

参阅

HealthRBInfo、HealthRateRBInfo

HealthRBAlarmOn函数使用示例

如下所示为判断是否发生机器人1第3关节润滑脂部件消耗报警的示例。

```
Function PrintAlarm4
If HealthRBAlarmOn(1, HEALTH_ROBOT_TYPE_GREASE, 3) = True Then
Print "Robot1 Joint3 Grease NG"
Else
Print "Robot1 Joint3 Grease OK"
EndIf
Fend
```

3.12.16 HealthRBAnalysis函数

是用于针对某一循环的机器人动作返回指定部件类别的可使用月数的函数。

格式

HealthRBAnalysis (机器人编号, 部件类别, 关节编号)

参数

机器人编号
以整数值(1-16)指定机器人编号。
部件类别
以整数值(2-6)或下述常数指定机器人相关部件。

常数	值	模式
HEALTH_ROBOT_TYPE_BELT	2	指定皮带。
HEALTH_ROBOT_TYPE_GREASE	3	指定润滑脂。
HEALTH_ROBOT_TYPE_MOTOR	4	指定电动机。
HEALTH_ROBOT_TYPE_GEAR	5	指定减速机。
HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE	6	指定滚珠丝杠花键。

关节编号
以整数值(1-6)指定关节。本命令无法用于附加轴。

返回值

以实数返回可使用月数。

如果未在指定关节中安装指定类别的部件，则返回-1。

说明

针对某一循环的机器人动作，模拟指定部件类别的可使用月数。在部件为新品的状态下进行24小时连续运转时，运算可使用的月数。不包括过去的使用状况。

注意

- 在Auto模式下不起作用。
- 空运行(包括虚拟控制器)时不起作用。

参阅

HealthRBStart、HealthRBStop

HealthRBAnalysis函数使用示例

```
Function RobotPartAnalysis
  Real month

  Robot 1

  HealthRBStart 1
  Motor On
  Go P0
  Go P1
  Motor Off
  HealthRBStop 1

  month = HealthRBAnalysis(1, HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE, 3)
```

```
  Print "Ball Screw Spline analysis =", Str$(month)
Fend
```

3.12.17 HealthRBDistance

用于显示指定关节的电动机驱动量(旋转量)。

格式

HealthRBDistance [机器人编号] [,关节编号]

参数

机器人编号
以整数值(1-16)指定机器人编号。可省略。省略时，使用当前机器人编号。
关节编号
以整数值(1-6)指定关节。如果未指定关节编号，将返回所有关节的值。本命令无法用于附加轴。

说明

针对从HealthRBStart到HealthRBStop的机器人动作，运算并显示指定关节的电动机驱动量(旋转量)。不包括过去的使用量。

注意

- 在Auto模式下不起作用。
- 空运行(包括虚拟控制器)时不起作用。

参阅

HealthRBStart, HealthRBStop

HealthRBDistance使用示例

如下所示为SCARA机器人第1关节驱动量的显示示例。

```
> HealthRBDistance 1, 1
 1.000
>
```

3.12.18 HealthRBDistance函数

用于返回指定关节的电动机驱动量(旋转量)。

格式

HealthRBDistance ([机器人编号,] 关节编号)

参数

机器人编号
　　以整数值(1-16)指定机器人编号。可省略。省略时，使用当前机器人编号。
关节编号
　　以整数值(1-6)指定关节。本命令无法用于附加轴。

返回值

以实数返回驱动量。

说明

针对从HealthRBStart到HealthRBStop的机器人动作，返回指定关节的电动机驱动量(转速)。不包括过去的使用量。

注意

- 在Auto模式下不起作用。
- 空运行(包括虚拟控制器)时不起作用。

参阅

HealthRBStart, HealthRBStop

HealthRBDistance函数使用示例

```
Function RobotPartAnalysis
  Real healthDistance

  Robot 1

  HealthRBStart 1
  Motor On
  Go P0
  Go P1
  Motor Off
  HealthRBStop 1

  healthDistance = HealthRBDistance(1,1)
  Print "Distance =", Str$(healthDistance)
Fend
```

3.12.19 HealthRBInfo

用于显示机器人相关指定类别的部件的推荐更换期限之前的剩余月数。

格式

HealthRBInfo 机器人编号, 部件类别[, 关节编号]

参数

机器人编号

以整数值(1-16)指定要返回的推荐更换期限之前剩余月数的机器人编号。

部件类别

以整数值(0-6)或以下所示常数指定要返回的推荐更换期限之前剩余月数的部件。

常数	值	模式
HEALTH_ROBOT_TYPE_ALL	0	指定所有部件类别。
HEALTH_ROBOT_TYPE_BATTERY	1	指定电池。
HEALTH_ROBOT_TYPE_BELT	2	指定皮带。
HEALTH_ROBOT_TYPE_GREASE	3	指定润滑脂。
HEALTH_ROBOT_TYPE_MOTOR	4	指定电动机。
HEALTH_ROBOT_TYPE_GEAR	5	指定减速机。
HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE	6	指定滚珠丝杠花键。

关节编号

以整数值(1-9)指定要返回的推荐更换期限之前剩余月数的关节。在部件类别中选择了电池时，因为电池在所有的关节中通用，指定任一关节均返回相同的值。如果未指定关节编号，将返回所有关节的值。本命令无法用于附加轴。

说明

用于显示机器人相关指定类别的部件的推荐更换期限之前的剩余月数。

根据基于过去使用状况的部件消耗率以及按通过HealthCalcPeriod设置的期间运转控制器获取的消耗率变化量，对推荐更换期限之前的月数进行运算。

如果机器人关节未使用指定部件，则显示-1。

注意

由于是根据按通过HealthCalcPeriod设置的期间运转控制器获取的消耗率变化量，对推荐更换期限之前的剩余月数进行运算，因此下述情况时，无法正常进行运算。

- 在运转时间未达到由HealthCalcPeriod设置的期间的状况下执行本命令时。
- 结束长期机器人运转停止期间之后执行本命令时。
- 更换部件并对部件消耗进行重置之后执行本命令时。
- 变更控制器的日期设置时。

在这种情况下，如果在对控制器进行HealthCalcPeriod设置期间2倍以上的运转之后执行本命令，将显示正确的值。

参阅

HealthRBAlarmOn, HealthRateRBInfo

HealthRBInfo使用示例

如下所示为机器人1所有关节的所有部件的推荐更换期限之前剩余月数的显示示例。

```
> HealthRBInfo 1, HEALTH_ROBOT_TYPE_ALL
BATTERY            240.000
BELT                -1.000,   -1.000,   38.689,   95.226
GREASE              -1.000,   -1.000,   21.130,   -1.000
MOTOR              240.000,  240.000,  240.000,  240.000
GEAR               240.000,  224.357,   -1.000,   -1.000
BALL_SCREW_SPLINE   -1.000,   -1.000,  240.000,   -1.000
>
```

如下所示为机器人1所有关节减速机的推荐更换期限之前剩余月数的显示示例。

```
> HealthRBInfo 1, HEALTH_ROBOT_TYPE_GEAR
GEAR               240.000,  224.357,   -1.000,   -1.000
>
```

如下所示为机器人1第2关节电动机的推荐更换期限之前剩余月数的显示示例。

```
> HealthRBInfo 1, HEALTH_ROBOT_TYPE_MOTOR, 2
MOTOR              224.357
>
```

3.12.20 HealthRBInfo函数

是用于返回机器人相关指定类别的部件的推荐更换期限之前的剩余月数的函数。

格式

HealthRBInfo (机器人编号, 部件类别, 关节编号)

参数

机器人编号
以整数值(1-16)指定要返回的推荐更换期限之前剩余月数的机器人编号。

部件类别
以整数值(1-6)或以下所示常数指定要返回的推荐更换期限之前剩余月数的部件。

常数	值	模式
HEALTH_ROBOT_TYPE_BATTERY	1	指定电池。
HEALTH_ROBOT_TYPE_BELT	2	指定皮带。
HEALTH_ROBOT_TYPE_GREASE	3	指定润滑脂。
HEALTH_ROBOT_TYPE_MOTOR	4	指定电动机。
HEALTH_ROBOT_TYPE_GEAR	5	指定减速机。
HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE	6	指定滚珠丝杠花键。

关节编号
以整数值(1-9)指定要返回的推荐更换期限之前剩余月数的关节。在部件类别中选择了电池时，因为电池在所有的关节中通用，指定任一关节均返回相同的值。本命令无法用于附加轴。

返回值

以实数返回推荐更换期限之前的剩余月数(单位：月)。

如果机器人关节未使用指定部件，则返回-1。

说明

根据基于过去使用状况的部件消耗率以及按通过HealthCalcPeriod设置的期间运转控制器获取的消耗率变化量，对推荐更换期限之前的月数进行运算。

注意

由于是根据按通过HealthCalcPeriod设置的期间运转控制器获取的消耗率变化量，对推荐更换期限之前的剩余月数进行运算，因此下述情况时，无法正常进行运算。

- 在运转时间未达到由HealthCalcPeriod设置的期间的状况下执行本命令时。
- 结束长期机器人运转停止期间之后执行本命令时。
- 更换部件并对部件消耗进行重置之后执行本命令时。
- 变更控制器的日期设置时。

在这种情况下，如果在对控制器进行HealthCalcPeriod设置期间2倍以上的运转之后执行本命令，将显示正确的值。

参阅

HealthRBAlarmOn, HealthRateRBInfo

HealthRBInfo函数使用示例

如下所示为机器人1第3关节滚珠丝杠花键的推荐更换期限之前的剩余月数为1个月以下时输出报警的示例。

```
Function AlarmCheck
  Real month

  month = HealthRBInfo(1, HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE, 3)
  If month < 1 Then
    Print "Alarm ON"
  EndIf
Fend
```

3.12.21 HealthRBRateOffset

对指定部件类别的消耗率设置偏移值。

格式

HealthRBRateOffset 机器人编号, 部件类别, 关节编号, 偏移值

参数

机器人编号
以整数值(1-16)指定机器人编号。

部件类别
以整数值(1-6)或下述常数指定机器人相关部件。

常数	值	模式
HEALTH_ROBOT_TYPE_BATTERY	1	指定电池。
HEALTH_ROBOT_TYPE_BELT	2	指定皮带。
HEALTH_ROBOT_TYPE_GREASE	3	指定润滑脂。
HEALTH_ROBOT_TYPE_MOTOR	4	指定电动机。
HEALTH_ROBOT_TYPE_GEAR	5	指定减速机。
HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE	6	指定滚珠丝杠花键。

关节编号
以整数值(1-6)指定关节。在部件类别中选择了电池时，因为电池在所有的关节中通用，指定任一关节，均设置偏移值。本命令无法用于附加轴。

偏移值
指定对消耗率进行偏移的整数值。(单位：%)

说明

对指定部件类别、关节设置消耗率的偏移值。

参阅

HealthRBAlarmOn, HealthRateRBInfo, HealthRBInfo

HealthRBRateOffset使用示例

如下所示为在机器人1第1关节减速机的消耗率加上10%的运算示例。

```
> HealthRBRateOffset 1,HEALTH_ROBOT_TYPE_GEAR,1,10
>
```

3.12.22 HealthRBReset

用于清除指定部件类别的部件的推荐更换期限之前的剩余月数与消耗率。

格式

HealthRBReset 机器人编号, 部件类别, 关节编号

参数

机器人编号
以整数值(1-16)指定机器人编号。

部件类别
以整数值(1-6)或下述常数指定机器人相关部件。

常数	值	模式
HEALTH_ROBOT_TYPE_BATTERY	1	指定电池。
HEALTH_ROBOT_TYPE_BELT	2	指定皮带。
HEALTH_ROBOT_TYPE_GREASE	3	指定润滑脂。
HEALTH_ROBOT_TYPE_MOTOR	4	指定电动机。
HEALTH_ROBOT_TYPE_GEAR	5	指定减速机。
HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE	6	指定滚珠丝杠花键。

关节编号
以整数值(1-6)指定要返回的推荐更换期限之前剩余月数的关节。在部件类别中选择了电池时，因为电池在所有的关节中通用，指定任一关节，均进行清除。本命令无法用于附加轴。

说明

针对指定部件类别、关节，清除推荐更换期限之前的剩余月数与消耗率。

也同时解除警告。

参阅

HealthRBAlarmOn, HealthRateRBInfo, HealthRBInfo

HealthRBReset使用示例

```
> HealthRBReset 1,HEALTH_ROBOT_TYPE_GEAR,1
>
```

3.12.23 HealthRBSpeed

用于显示指定关节速度的平均值。

格式

HealthRBSpeed [机器人编号] [, 关节编号]

参数

机器人编号
以整数值(1-16)指定机器人编号。可省略。省略时，使用当前机器人编号。
关节编号
以整数值(1-6)指定关节。如果未指定关节编号，将返回所有关节的值。本命令无法用于附加轴。

说明

针对从HealthRBStart到HealthRBStop的机器人动作，显示指定关节的速度绝对值的平均值。以0～1的实值显示结果。最大平均速度值为“1”。

平均值为0.001以下时，将变为“0”。

注意

- 在Auto模式下不起作用。
- 空运行(包括虚拟控制器)时不起作用。

参阅

HealthRBStart, HealthRBStop, AveSpeed

HealthRBSpeed使用示例

如下所示为SCARA机器人第1关节速度的显示示例。

```
> HealthRBSpeed 1, 1
 0.100
>
```

3.12.24 HealthRBSpeed函数

用于返回指定关节速度绝对值的平均值。

格式

HealthRBSpeed ([机器人编号,] 关节编号)

参数

机器人编号
　　以整数值(1-16)指定机器人编号。可省略。省略时，使用当前机器人编号。
关节编号
　　以整数值(1-6)指定关节。本命令无法用于附加轴。

返回值

以0～1的实值返回。

说明

针对从HealthRBStart到HealthRBStop的机器人动作，返回指定关节的速度绝对值的平均值。以0～1的实值显示结果。最大平均速度为1。

注意

- 在Auto模式下不起作用。
- 空运行(包括虚拟控制器)时不起作用。

参阅

HealthRBStart, HealthRBStop, AveSpeed

HealthRBSpeed函数使用示例

```
Function RobotPartAnalysis
  Real healthSpeed

  Robot 1

  HealthRBStart 1
  Motor On
  Go P0
  Go P1
  Motor Off
  HealthRBStop 1

  healthSpeed = HealthRBSpeed(1,1)
  Print "AveSpeed =", Str$ (healthSpeed)
Fend
```

3.12.25 HealthRBStart

用于开始进行某一循环机器人动作的部件的可使用月数与要素的运算。

格式

HealthRBStart 机器人编号

参数

机器人编号
以整数值(1-16)指定机器人编号。

说明

用于开始相对于指定机器人编号的部件的可使用月数与要素(转矩、速度、驱动量)的运算。

如果在开始运算的状态下再次执行，上次的运算结果则会被初始化。

注意

- 在Auto模式下不起作用。
- 空运行(包括虚拟控制器)时不起作用。

参阅

HealthRBAnalysis、HealthRBStop、HealthRBTRQ、HealthRBSpeed、HealthRBDistance

HealthRBStart使用示例

```
Function RobotPartAnalysis
  Real month

  Robot 1

  HealthRBStart 1
  Motor On
  Go P0
  Go P1
  Motor Off
  HealthRBStop 1

  month = HealthRBAnalysis(1, HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE, 3)
  Print "Ball Screw Spline analysis =", Str$(month)
Fend
```

3.12.26 HealthRBAnalysis

用于针对某一循环的机器人动作模拟并显示指定部件类别的可使用月数。

格式

HealthRBAnalysis 机器人编号, 部件类别[, 关节编号]

参数

机器人编号
以整数值(1-16)指定机器人编号。

部件类别
以整数值或下述常数指定机器人相关部件。

常数	值	模式
HEALTH_ROBOT_TYPE_ALL	0	指定所有部件类别。
HEALTH_ROBOT_TYPE_BELT	2	指定皮带。
HEALTH_ROBOT_TYPE_GREASE	3	指定润滑脂。
HEALTH_ROBOT_TYPE_MOTOR	4	指定电动机。
HEALTH_ROBOT_TYPE_GEAR	5	指定减速机。
HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE	6	指定滚珠丝杠花键。

关节编号
以整数值(1-6)指定关节。如果未指定关节编号，将返回所有关节的值。本命令无法用于附加轴。

说明

针对某一循环的机器人动作，模拟指定部件类别的可使用月数。在部件为新品的状态下进行24小时连续运转时，运算并显示可使用的月数。不包括过去的使用状况。

如果未在指定关节中安装指定类别的部件，则返回-1。

注意

- 在Auto模式下不起作用。
- 空运行(包括虚拟控制器)时不起作用。

参阅

HealthRBStart, HealthRBStop

HealthRBAnalysis使用示例

如下所示为SCARA机器人所有关节的所有部件可使用月数的显示示例。

```
> HealthRBAnalysis 1, HEALTH_ROBOT_TYPE_ALL
BELT                -1.000,   -1.000,   38.689,   95.226
GREASE              -1.000,   -1.000,   21.130,   -1.000
MOTOR              240.000,  240.000,  240.000,  240.000
GEAR               240.000,  224.357,   -1.000,   -1.000
BALL_SCREW_SPLINE   -1.000,   -1.000,  240.000,   -1.000
>
```

如下所示为SCARA机器人所有关节减速机可使用月数的显示示例。

```
> HealthRBAnalysis 1, HEALTH_ROBOT_TYPE_GEAR
GEAR               240.000,  224.357,   -1.000,   -1.000
>
```

如下所示为6轴型机器人第2关节电动机可使用月数的显示示例。

```
> HealthRBAnalysis 1, HEALTH_ROBOT_TYPE_MOTOR, 2
MOTOR               224.357
>
```

3.12.27 HealthRBStop

用于结束进行某一循环机器人动作的部件的可使用月数与要素的运算。

格式

HealthRBStop 机器人编号

参数

机器人编号
以整数值(1-16)指定机器人编号。

说明

用于结束相对于指定机器人编号的部件的可使用月数与要素(转矩、速度、驱动量)的运算。

注意

- 在Auto模式下不起作用。
- 空运行(包括虚拟控制器)时不起作用。
- 运算1小时后，将自动结束。如果在自动结束之后执行该命令，将发生错误。
- 如果在未执行HealthRBStart命令的状态下执行该命令，则会发生错误。
- 如果在执行HealthRBStop之后未执行HealthRBStart，并再次执行HealthRBStop，也会发生错误。

参阅

HealthRBAnalysis、HealthRBStart、HealthRBTRQ、HealthRBSpeed、HealthRBDistance

HealthRBStop使用示例

```
Function RobotPartAnalysis
  Real month

  Robot 1

  HealthRBStart 1
  Motor On
  Go P0
  Go P1
  Motor Off
  HealthRBStop 1

  month = HealthRBAnalysis(1, HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE, 3)
  Print "Ball Screw Spline analysis =", Str$(month)
Fend
```

3.12.28 HealthRBTRQ

用于显示指定关节的使用寿命影响转矩值。

格式

HealthRBTRQ [机器人编号] [, 关节编号]

参数

机器人编号
　　以整数值(1-16)指定机器人编号。可省略。省略时，使用当前机器人编号。
关节编号
　　以整数值(1-6)指定关节。如果未指定关节编号，将返回所有关节的值。本命令无法用于附加轴。

说明

针对从HealthRBStart到HealthRBStop的机器人动作，显示指定关节的使用寿命影响转矩值。以0～1的实值显示结果。最大转矩值为“1”。

注意

- 在Auto模式下不起作用。
- 空运行(包括虚拟控制器)时不起作用。

参阅

HealthRBStart、HealthRBStop、ATRQ

HealthRBTRQ使用示例

如下所示为SCARA机器人第1关节的寿命影响转矩值的显示示例。

```
> HealthRBTRQ 1, 1
 0.020
>
```

3.12.29 HealthRBTRQ函数

用于返回指定关节的使用寿命影响转矩值。

格式

HealthRBTRQ ([机器人编号,]关节编号)

参数

机器人编号
　　以整数值(1-16)指定机器人编号。可省略。省略时，使用当前机器人编号。
关节编号
　　以整数值(1-6)指定关节。本命令无法用于附加轴。

返回值

以0～1的实值返回。

说明

针对从HealthRBStart到HealthRBStop的机器人动作，返回指定关节的使用寿命影响转矩值。以0～1的实值表示结果。最大实效转矩值是1。

注意

- 在Auto模式下不起作用。
- 空运行(包括虚拟控制器)时不起作用。

参阅

HealthRBStart、HealthRBStop、ATRQ

HealthRBTRQ函数使用示例

```
Function RobotPartAnalysis
  Real healthTRQ

  Robot 1

  HealthRBStart 1
  Motor On
  Go P0
  Go P1
  Motor Off
  HealthRBStop 1

  healthTRQ = HealthRBTRQ(1,1)
  Print "Torque =", Str$(healthTRQ)
Fend
```

3.12.30 HealthRBWarningEnable

用于将机器人相关指定类别部件的部件消耗报警通知设为有效或无效。

格式

HealthRBWarningEnable 机器人编号, 部件类别[, On/Off]

参数

机器人编号

以整数值(1-16)指定机器人编号。

部件类别

以整数值(1-6)或下述常数指定机器人相关部件。

常数	值	模式
HEALTH_ROBOT_TYPE_BATTERY	1	指定电池。
HEALTH_ROBOT_TYPE_BELT	2	指定皮带。
HEALTH_ROBOT_TYPE_GREASE	3	指定润滑脂。
HEALTH_ROBOT_TYPE_MOTOR	4	指定电动机。
HEALTH_ROBOT_TYPE_GEAR	5	指定减速机。
HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE	6	指定滚珠丝杠花键。

On/Off

- On：将部件消耗报警通知设为有效。
- Off：将部件消耗报警通知设为无效。

结果

如果省略On/Off参数，将显示当前的On/Off设置值。

说明

发生指定类别部件的部件消耗报警时，设置是否通知部件消耗报警。

注意

如果已将指定部件的部件消耗报警通知设为无效，即使到了推荐更换期限，也不会发出警告通知。执行时，请充分注意依据本命令的设置。

参阅

HealthRBAlarmOn

HealthRBWarningEnable使用示例

如下所示为将SCARA机器人1的润滑脂部件消耗报警通知设为无效的示例。

```
> HealthRBWarningEnable 1, HEALTH_ROBOT_TYPE_GREASE, Off
```

如下所示为SCARA机器人1的润滑脂部件消耗报警通知设置的显示示例。

```
> HealthRBWarningEnable 1, HEALTH_ROBOT_TYPE_GREASE
GREASE  Off
>
```

3.12.31 HealthRBWarningEnable函数

用于返回机器人相关指定类别部件的部件消耗报警通知的设置状态的函数。

格式

HealthRBWarningEnable (机器人编号, 部件类别)

参数

机器人编号
以整数值(1-16)指定要返回的推荐更换期限之前剩余月数的机器人编号。
部件类别
以整数值(1-6)或以下所示常数指定要返回的推荐更换期限之前剩余月数的部件。

常数	值	模式
HEALTH_ROBOT_TYPE_BATTERY	1	指定电池。
HEALTH_ROBOT_TYPE_BELT	2	指定皮带。
HEALTH_ROBOT_TYPE_GREASE	3	指定润滑脂。
HEALTH_ROBOT_TYPE_MOTOR	4	指定电动机。
HEALTH_ROBOT_TYPE_GEAR	5	指定减速机。
HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE	6	指定滚珠丝杠花键。

返回值

以整数值返回部件消耗报警的设置值。

- 1: On
- 0: Off

参阅

HealthRBAlarmOn

HealthRBWarningEnable函数使用示例

如下所示为SCARA机器人1的润滑脂部件消耗报警通知设置状态的显示示例。

```
Print HealthRBWarningEnable(1, HEALTH_ROBOT_TYPE_GREASE )
```

3.12.32 Here

用于作为机器人坐标示教当前位置。

格式

Here 坐标

参数

坐标

以P编号、P(表达式)、点标签之一进行指定。

注意

- Here语句与并行处理

  不能将Here语句与并行处理同时放在一个动作命令内。

  ```
  Go Here :Z(0) !D10; MemOn 1 !
  ```

  无法实现上述使用方法。

  ```
  P999 = Here
  Go P999 :Z(0) ! D10; MemOn 1 !
  ```

  请变更为上述程序。

- Here语句与多任务

  已经在通过Xqt等执行的多任务函数内执行了Here语句时，如果主任务内正在利用Move、Go等进行机器人动作，则会发生错误停止。

  可通过使用CurPos获取正在动作的当前位置。

  例

  ```
  Function Xqt_PrintHere
  Do
  Print CurPOS
  Wait 0.1
  Loop
  Fend
  Function main
  Xqt 10, Xqt_PrintHere
  Go P0
  Fend
  ```

- 与CP并用

  如果在路径运动有效时使用Here，则在指定Here的动作之前不会合成运动轨迹，而是减速。如果采用如下的使用方法，则在P1处减速停止后开始下一个动作。

  ```
  Go P1 CP
  Go Here+X(100)
  ```

  如果要不停止的情况下执行动作，则请修改程序以预先计算Here的位置。

```
Go P1 CP
Go P1+X(100)
```

参阅

Here函数、CurPos

Here使用示例

```
Here P1
Here pick
```

3.12.33 Here函数

用于返回机器人的当前位置。

格式
Here

返回值
用于返回机器人的当前位置。

说明
Here函数用于获取正在使用的机械手的当前位置。

参照
Here函数

Here函数使用示例

```
P1 = Here
```

3.12.34 Hex$函数

用于将16进制数表示的数值转换为字符串并返回。

格式

Hex$ (数值)

参数

数值
　　指定整数值。

返回值

用于将16进制数表示的数值转换为ASCII值的字符串并返回。

说明

Hex$函数用于将16进制数表示的数值转换为字符串并返回。字符由0～9、A～F构成。Hex$用于确认Stat函数的结果。

参阅

Str$、Stat、Val

Hex$函数使用示例

```
> print hex$(stat(0))
A00000
> print hex$(255)
FF
```

3.12.35 History

显示命令执行记录。

格式
History

结果
按照执行顺序显示命令执行记录。

说明
显示最近在命令窗口执行的命令执行记录（最大100件）。

注意

仅可利用命令窗口执行该命令。

参阅
ClearHistory

History使用示例

```
> Motor On
> Go P1
> History
Motor On
Go P1
```

3.12.36 Hofs

用于设置和显示从编码器原值原点和内部传感器原点之间的偏移脉冲值。

格式

(1) Hofs 第1关节设置值, 第2关节设置值, 第3关节设置值, 第4关节设置值 [, 第5关节设置值, 第6关节设置值] [, 第7关节设置值] [, 第8关节设置值, 第9关节设置值]

(2) Hofs

参数

第1关节设置值
　　以表达式或数值指定第1关节偏移脉冲值(整数)。
第2关节设置值
　　以表达式或数值指定第2关节偏移脉冲值(整数)。
第3关节设置值
　　以表达式或数值指定第3关节偏移脉冲值(整数)。
第4关节设置值
　　以表达式或数值指定第4关节偏移脉冲值(整数)。
第5关节设置值
　　是垂直6轴型机器人(包括N系列)的专用参数。以表达式或数值指定第5关节偏移脉冲值(整数)。
第6关节设置值
　　是垂直6轴型机器人(包括N系列)的专用参数。以表达式或数值指定第6关节偏移脉冲值(整数)。
第7关节设置值
　　是关节型7轴机器人的专用参数。以表达式或数值指定第7关节偏移脉冲值(整数)。
第8关节设置值
　　是附加轴S关节的专用参数。以表达式或数值指定第8关节(附加轴S)偏移脉冲值(整数)。
第9关节设置值
　　是附加轴T关节的专用参数。以表达式或数值指定第9关节(附加轴T)偏移脉冲值(整数)。

返回值

如果未指定参数，则显示当前的Hofs设置值。

说明

Hofs用于显示或设置原点偏移脉冲值。Hofs用于设置从编码器原点(Z相)到机械原点的偏移值。

机器人的动作控制虽然基于各关节配备的编码器的原点，但编码器原点未必与机器人的机械原点一致。因此，为了将与机械原点一致的编码器位置设为软件上的原点，需要利用Hofs设置补偿脉冲量。

注意

- Hofs值如非必要请绝对不要变更。

  在工厂发货时已精密地设置了Hofs值。如果超出必要地变更此值，可能会导致定位错误以及意想不到的动作。除非必要，否则切勿变更Hofs值。

- 重置JointAccuracy (仅限于支持关节精度补偿的机型)

  支持关节精度补偿的机型，当在Hofs中设置了原点校正脉冲值时，则变更了的关节的JointAccuracy中设置的关节精度补偿值会被重置为“0”。如果不想重置JointAccuracy中的校正值，请使用HofsJointAccuracy。

- 如要自动计算Hofs值

  为自动计算Hofs值，将机械臂移至要校准的位置后执行Calib。这样的话，控制器根据CalPls脉冲值和校准位置脉冲值自动计算Hofs值。

- Hofs的保存和还原

Hofs可以利用［系统设置］菜单 -［系统设置］对话框 -［机器人］-［校准］中的［保存］以及［读取］进行保存和还原。

- 当使用安装有Safety板的控制器时，请在运行本命令后启动安全功能管理器

  使用安装Safety板的控制器时，控制器的Hofs值和具有安全功能的Safety板的Hofs值必须一致。

  执行此命令时，只会改变控制器的Hofs值，并且因为与Safety板的Hofs值不同，会发生警报。

  因此，执行本命令后，请启动安全功能管理器更新Safety板的设置。

  有关详细信息，请参阅以下手册。
  《机器人控制器 安全功能手册》

参阅

Calib、CalPls、JointAccuracy、HofsJointAccuracy、Home、Hordr、MCal、SysConfig

Hofs使用示例

如下所示为利用监视器窗口的简单的使用示例。设置第1关节的原点偏移值为-545、第2关节的原点偏移值为514、第3关节与第4关节的原点偏移值为0，然后显示当前的原点偏移值。

```
> hofs  -545, 514, 0, 0

> hofs
-545, 514, 0, 0
>
```

3.12.37 Hofs函数

该函数用于返回各关节的编码器原值点和内部传感器(HOFS)之间的偏移脉冲值。

格式

Hofs (关节编号)

参数

关节编号
　　用于以表达式或数值指定计算Hofs值的关节编号(整数)。附加轴的S轴为8，T轴为9。

返回值

返回指定的关节的偏移脉冲值(整数值，单位：脉冲)。

参阅

Calib、CalPls、Home、Hordr、MCal、SysConfig

Hofs函数使用示例

下例为使用Hofs函数的程序例。

```
Function DisplayHofs
   Integer i

   Print "Hofs settings:"
   For i = 1 To 4
      Print "Joint ", i, " = ", Hofs(i)
   Next i
Fend
```

3.12.38 HofsJointAccuracy

用于在不变更关节精度补偿的补偿值的状态下，设置并显示从编码器原点到软件原点的补偿脉冲。

格式

(1) HofsJointAccuracy 第1关节设置值, 第2关节设置值, 第3关节设置值, 第4关节设置值 [, 第5关节设置值, 第6关节设置值] [, 第7关节设置值] [, 第8关节设置值, 第9关节设置值]

(2) HofsJointAccuracy

参数

第1关节设置值
　　以表达式或数值指定第1关节偏移脉冲值(整数)。
第2关节设置值
　　以表达式或数值指定第2关节偏移脉冲值(整数)。
第3关节设置值
　　以表达式或数值指定第3关节偏移脉冲值(整数)。
第4关节设置值
　　以表达式或数值指定第4关节偏移脉冲值(整数)。
第5关节设置值
　　是垂直6轴型机器人(包括N系列)的专用参数。以表达式或数值指定第5关节偏移脉冲值(整数)。
第6关节设置值
　　是垂直6轴型机器人(包括N系列)的专用参数。以表达式或数值指定第6关节偏移脉冲值(整数)。
第7关节设置值
　　是关节型7轴机器人的专用参数。以表达式或数值指定第7关节偏移脉冲值(整数)。
第8关节设置值
　　是附加轴S关节的专用参数。以表达式或数值指定第8关节(附加轴S)偏移脉冲值(整数)。
第9关节设置值
　　是附加轴T关节的专用参数。以表达式或数值指定第9关节(附加轴T)偏移脉冲值(整数)。

返回值

如果未指定参数，则显示当前的Hofs设置值。

说明

HofsJointAccuracy用于在不变更关节精度补偿的补偿值的状态下，显示或设置原点补偿脉冲值。有关支持关节精度补偿的机型，请参阅以下手册。
《机械手手册》

机器人的动作控制虽然基于各关节配备的编码器的原点，但编码器原点未必与机器人的机械原点一致。因此，为了将与机械原点一致的编码器位置设为软件上的原点，需要利用Hofs设置补偿脉冲量。

在各关节发生变更时，Hofs会将各JointAccuracy设置的关节精度补偿的补偿值重置为“0”，但HofsJointAccuracy不会进行重置。不想将关节精度补偿的补偿值重置为“0”时，请使用HofsJointAccuracy。

注意

- Hofs值如非必要请绝对不要变更。

  出厂时已对Hofs值进行了精密设置。如果不必要地变更该值，则可能会导致定位错误或意想不到的动作，非常危险。除非必要，否则切勿变更Hofs值。

参阅

JointAccuracy, Hofs

HofsJointAccuracy使用示例

如下所示为命令窗口的简单使用示例。将第1关节原点补偿值设为“-545”、将第2关节原点补偿值设为“514”、将第3关

节与第4关节的原点补偿值设为“0”，然后显示当前的原点补偿值。使用HofsJointAccuracy前后，关节精度补偿的补偿值保持不变。

```
> JointAccuracy 1
1000, 420, 100, 240
> HofsJointAccuracy -545, 514, 0, 0

> HofsJointAccuracy
-545, 514, 0, 0
> JointAccuracy 1
  1000, 420, 100, 240
>
```

3.12.39 Home

将机器人机械臂移动到用户定义的原点位置

格式
Home

说明
按照Hordr设置的关节顺序以低速PTP动作将机械臂移至HomeSet指定的原点(home)。

水平多关节型机器人(包括RS系列)一般在第3关节最先返回HomeSet位置后，第1关节、第2关节、第4关节同时返回各自的HomeSet坐标位置。返回原点位置的关节的顺序可以通过Hordr命令进行变更。

注意

- Home 状态的输出

  在机器人位于原点位置(home)时，控制器的Home输出将变为ON。

易引起的错误

- 未设置HomeSet值的情况下执行Home时

  如果未设置HomeSet值的情况下执行Home，将会发生错误2228。

参阅
HomeClr、HomeDef、Hordr、HomeSet

Home使用示例
Home命令可以通过下述程序来使用。

```
Function InitRobot
   Reset
   If Motor = Off Then
      Motor On
   EndIf
   Home
Fend
```

或者，通过命令窗口按如下所述发行。

```
> home
>
```

3.12.40 HomeClr

用于清除原点位置的设置。

格式
HomeClr

说明
机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

参阅
HomeDef、HomeSet

HomeClr使用示例
HomeClr函数可以通过下述程序来使用。

```
Function ClearHome

   If HomeDef = True Then
      HomeClr
   EndIf
Fend
```

3.12.41 HomeDef函数

用于返回是否设置了原点位置。

格式

HomeDef

返回值

如果已设置，则返回“True”；如果未设置，则返回“False”。

参阅

HomeClr、HomeSet

HomeClr使用示例

HomeDef函数可以通过下述程序来使用。

```
Function DisplayHomeSet

   Integer i

   If HomeDef = False Then
      Print "Home is not defined"
   Else
      Print "Home values:"
      For i = 1 To 4
         Print "J", i, " = ", HomeSet(i)
      Next i
   EndIf
Fend
```

3.12.42 HomeSet

用于设置和显示原点位置(home)的脉冲。

格式

(1) HomeSet 第1关节脉冲值, 第2关节脉冲值, 第3关节脉冲值, 第4关节脉冲值[,第5关节脉冲值, 第6关节脉冲值] [,第7关节脉冲值] [,第8关节脉冲值, 第9关节脉冲值]

(2) HomeSet

参数

第1关节脉冲值
　　以表达式或数值指定第1关节内部编码器脉冲值(整数)。
第2关节脉冲值
　　以表达式或数值指定第2关节内部编码器脉冲值(整数)。
第3关节脉冲值
　　以表达式或数值指定第3关节内部编码器脉冲值(整数)。
第4关节脉冲值
　　以表达式或数值指定第4关节内部编码器脉冲值(整数)。
第5关节脉冲值
　　是垂直6轴型机器人(包括N系列)的专用参数。以表达式或数值指定第5关节内部编码器脉冲值(整数)。
第6关节脉冲值
　　是垂直6轴型机器人(包括N系列)的专用参数。以表达式或数值指定第6关节内部编码器脉冲值(整数)。
第7关节脉冲值
　　是关节型7轴机器人的专用参数。以表达式或数值指定第7关节内部编码器脉冲值(整数)。
第8关节脉冲值
　　是附加轴S关节的专用参数。以表达式或数值指定第8关节(附加轴S)内部编码器脉冲值(整数)。
第9关节脉冲值
　　是附加轴T关节的专用参数。以表达式或数值指定第9关节(附加轴T)内部编码器脉冲值(整数)。

结果

如果省略参数，则显示当前的内部编码器脉冲值。

说明

用户可以对各关节指定脉冲值，设置原点位置(home)。

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

易引起的错误

- 未设置HomeSet值的情况下执行Home时

  如果未设置HomeSet值的情况下执行Home，将会发生错误2228。

- 未设置HomeSet值的情况下想要显示HomeSet值时

  如要在未设置HomeSet值的情况下显示原点位置的脉冲值，将会发生错误2228。

参阅

Home、HomeClr、HomeDef、Hordr、Pls

HomeSet使用示例

如下所示为利用命令窗口的操作示例。

```
> homeset 0,0,0,0   '将原点位置设为0,0,0,0
> homeset
  0   0
  0   0
```

```
> home                    '机器人向原点位置移动
```

使用Pls函数将机械臂的当前位置设为原点位置。

```
> homeset Pls(1), Pls(2), Pls(3), Pls(4)
```

3.12.43 HomeSet函数

该函数用于返回指定关节的原点位置(待机姿势)的脉冲值。

格式

HomeSet (关节编号)

参数

关节编号
　　用于以表达式或数值指定计算HomeSet值的关节编号。附加轴的S轴为8，T轴为9。

返回值

返回指定关节的原点脉冲值。如果指定关节编号为“0”，在有HomeSet值的设置时返回“1”，没有设置时返回“0”。

参阅

HomeSet

HomeSet函数使用示例

下例为使用HomeSet函数的程序例。

```
Function DisplayHomeSet

   Integer i

   If HomeSet(0) = 0 Then
      Print "HomeSet is not defined"
   Else
      Print "HomeSet values:"
      For i = 1 To 4
         Print "J", i, " = ", HomeSet(i)
      Next i
   EndIf
Fend
```

3.12.44 Hordr

进行执行Home时各关节的动作顺序的指定和显示。

格式

(1) Hordr 设置值1, 设置值2, 设置值3, 设置值4 [,设置值5] [,设置值6] [,设置值7] [,设置值8] [,设置值9]

(2) Hordr

参数

设置值1
　　以位模式指定在第1步恢复的关节。
设置值2
　　以位模式指定在第2步恢复的关节。
设置值3
　　以位模式指定在第3步恢复的关节。
设置值4
　　以位模式指定在第4步恢复的关节。
设置值5
　　以位模式指定在第5步恢复的关节。
设置值6
　　以位模式指定在第6步恢复的关节。
设置值7
　　以位模式指定在第7步恢复的关节。
设置值8
　　以位模式指定在第8步恢复的关节。
设置值9
　　以位模式指定在第9步恢复的关节。

结果

如果省略参数，则显示当前的恢复顺序设置。

说明

Hordr用于设置执行Home命令时的关节的恢复顺序。确定哪个关节移至第几个原点位置(home)。

用户可以利用Hordr命令来变更执行Home时的各关节的恢复顺序。根据机器人的类型，可分4、6或9步设置恢复顺序。用户可利用通过Hordr指定各步骤恢复的关节。也可以指定多个要在各步骤进行恢复的关节。理论上，可以同时恢复所有关节。但时，如果是水平多关节型机器人(包括RS系统的4轴机器人)，一般来说，建议第1步最先移动第3关节，然后在此后的步骤中恢复其它关节。

使用Hordr命令时，应指定各个步骤的对应位模式。各关节的位模式已经规定。如果在某步骤，位为“1”，对应的关节则移至原点位置(home)。如果将位清除为“0”，在该步骤中对应关节不会移至原点位置(待机姿势)。按如下所述分配各关节的位模式。

位模式表

关节名	位编号	2进制数标记
第1关节	bit 0	&B0001
第2关节	bit 1	&B0010
第3关节	bit 2	&B0100
第4关节	bit 3	&B1000
第5关节	bit 4	&B10000
第6关节	bit 5	&B100000

关节名	位编号	2进制数标记
第7关节	bit 6	&B1000000
第8关节	bit 7	&B10000000
第9关节	bit 8	&B100000000

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

参阅

Home、HomeSet

Hordr使用示例

下例为从命令窗口执行的水平多关节型机器人(包括RS系列的4轴机器人)的操作例。

本例以二进制数将机器人的原点位置的恢复顺序设为如下。

第1步对第3关节，第2步对第1关节，第3步对第2关节，第4步对第4关节进行原点位置恢复。

```
>hordr  &B0100, &B0001, &B0010, &B1000
```

本例以十进制数将机器人的原点位置的恢复顺序设为如下。第1步对第3关节进行原点恢复，第2步对第1关节、第2关节、第4关节同时进行原点位置恢复。

```
>hordr  4, 11, 0, 0
```

下例所示为用10进制数显示当前的恢复顺序。

```
>hordr
4, 11, 0, 0
>
```

3.12.45 Hordr函数

用于返回执行Home时指定的各关节的动作顺序。

格式

Hordr (设置值编号)

参数

设置值编号
　　指定表示Hordr命令的动作顺序的整数值。

返回值

返回指定动作顺序的Horder设置值的数值。

参阅

Home、HomeSet

Hordr函数使用示例

```
Integer a
a = Hordr(1)
```

3.12.46 Hour

用于表示控制器的累计通电时间。

格式

Hour

说明

接通控制器的电源并显示累计时间(累计通电时间)。累计时间的单位为时间。

参阅

Time

Hour使用示例

下例所示为通过命令窗口执行的情况。

```
> hour
2560
>
```

3.12.47 Hour函数

用于返回控制器的累计通电时间。

格式
Hour

返回值
以实值返回控制器的累计通电时间。

参阅
Time

Hour函数使用示例

```
Print "Number of controller operating hours: ", Hour
```

3.13 I

3.13.1 If…Then…Else…EndIf

按照指定的条件转移执行命令。

格式

(1) If条件表达式 Then
语句T1
.
.
[ElseIf 条件表达式 Then]
语句T1
.
.
[Else]
语句F1
.
.
EndIf

(2) If 条件表达式 Then 语句T1 [;语句T2...] [Else 语句F1 [;语句F2...]]

参数

条件表达式
返回真伪值(True/False的值)的有效条件表达式。真(True)时返回“0”以外的数值，伪(False)时返回“0”。(请参阅下述条件表达式示例。)
语句T1
条件表达式的值为真(True)时，也就是满足条件时执行语句。(可以将多个语句记述在If...Then...Else形式的区段中。)
语句F1
条件表达式的值为伪(False)时，也就是满足条件时执行语句。(可以将多个语句记述在If...Then...Else形式的区段中。)

说明

(1) If...Then...Else在满足条件时，执行T1以后的语句。如果不满足条件，执行F1以后的语句。
If...Then...Else命令的Else部分可省略。如果省略Else部分而又未满足条件，将执行EndIf以后的语句。请用EndIf结束If...Then...Else语句段，而与有无Else无关。

(2) If...Then...Else还可以通过段以外的形式使用。是将If...Then...Else的所有语句都记述在同一行的方法。按照此方法使用If...Then...Else时，不需要EndIf。如果满足此行记述的条件表达式(如果有相对于条件表达式的真值(True))，将执行Then与Else之间的语句。如果不满足条件表达式(如果有相对于条件表达式的伪值(False))，将执行Else后面的语句。If...Then...Else的Else部分不是必须的。在不满足条件表达式的情况下(相对于条件表达式是伪值(False))，如果没有Else关键字，控制将转至程序中的下一个语句。

注意

- 条件表达式范例
  - a = b：a等于b
  - a < b：a小于b
  - a >= b：a大于等于b
  - a <> b：a不等于b

- a > b：a大于b
- a <= b：a小于等于b

也可以使用And、Or、Xor等逻辑运算符。

- 条件表达式中使用True时

常数 True的值是-1，由于是Boolean型，在与其他型变量的比较条件中使用时需要注意。

```
Function main
  Integer i
  i = 3
  If i = True Then
    Print "i=TRUE"
  EndIf
Fend
```

执行上述程序后，将显示“i=TRUE”。这是因为包括Boolean型的条件判断是通过“0”或“非0”非来判断。

如果“i”值不是“0”，将判断为条件成立并进行显示。

参阅

Else、Select...Case、Do…Loop

If/Then/Else使用示例

[1行的If...Then...Else]

下例是为确定特定输出的ON/OFF而检查输入的简单示例。这样的任务用于经常变动的I/O任务等任务中。

```
Function main
    Do
       If Sw(0) = 1 Then On 1 Else Off 1
    Loop
Fend
```

[区段显示的If...Then...Else]

下述所示为检查几个输入情况并输出该输入状态的简单示例。

```
If Sw(0) = 1 Then Print "Input0 ON" Else Print "Input0 OFF"
'
If Sw(1) = 1 Then
    If Sw(2) = 1 Then
       Print "Input1 On and Input2 ON"
    Else
       Print "Input1 On and Input2 OFF"
    EndIf
Else
    If Sw(2) = 1 Then
       Print "Input1 Off and Input2 ON"
    Else
       Print "Input1 Off and Input2 OFF"
    EndIf
EndIf
```

[其它格式示例]

```
If x = 10 And y = 3 Then GoTo 50
If test <= 10 Then Print "Test Failed"
If Sw(0) = 1 Or Sw(1) = 1 Then Print "Everything OK"
```

3.13.2 ImportPoints

用于将点文件输入到现在正在选择的机器人的项目中。

格式

ImportPoints 源路径, 文件名, [机器人编号]

参数

源路径
　　表示要在当前项目中输入的特定轨迹和文件的字符串表达式扩展名是“.pts”或“.pnt”(Epson RC+ 3.x / 4.x 形式)。详情请参阅ChDisk。

文件名
　　表示要输入到当前机器人的当前项目中的特定文件的字符串表达式扩展名是pts。不能指定路径。另外，不受ChDisk等的影响。详情请参阅ChDisk。

机器人编号
　　指定哪个机器人与点文件相关的整数表达式可省略。机器人编号=0时，将作为通用点文件输入点文件。省略时，使用当前机器人编号。

说明

ImportPoints用于将点文件复制到现在正在选择的项目中，并将该点文件添加到现在正在选择的机器人的文件中。添加的点文件将被编译，变为LoadPoints命令可以读取的文件。如果现在正在选择的机器人已存在相同文件将被覆盖，并重新编译。

点数据被保存在控制器内的小型闪存卡中。如果执行ImportPoints，则会向小型闪存卡执行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议只在需要时执行ImportPoints。

易引起的错误

- 不存在文件时

  如果不存在指定的源路径，则会发生错误。

- 找不到指定文件时

  如果文件名中包括路径，则会发生错误。

- 指定文件不是当前机器人的文件时

  如果在文件名中指定其它机器人的点文件，则会发生错误。

参阅

LoadPoints、Robot、SavePoints

ImportPoints使用示例

```
Function main
  Robot 1
  ImportPoints "c:\mypoints\model1.pts", "robot1.pts"
  LoadPoints "robot1.pts"
Fend
```

3.13.3 In函数

以字节为单位返回指定输入端口的状态。字节端口由8个输入位构成。

格式
In (字节端口编号)

参数

字节端口编号
　　指定I/O的字节端口。

返回值
返回0～255的整数。返回值为8位，各个位分别对应于1个输入位。

说明
在In中可以同时看见8个输入位的值。In命令通过将8个I/O位的状态保存为1个变量值或与Wait命令一起使用，可以实现“在2个以上I/O位的状态符合特定条件之前，使程序保持待机状态”的使用方法。

由于1次可以检查8个输入位，因此，返回值的范围为0～255。有关各整数返回值与各输入位的对应关系，请参照下表。

输入位表(使用字节端口0时)

返回值	7	6	5	4	3	2	1	0
1	Off	Off	Off	Off	Off	Off	Off	On
5	Off	Off	Off	Off	Off	On	Off	On
15	Off	Off	Off	Off	On	On	On	On
255	On	On	On	On	On	On	On	On

输入位表(使用字节端口2时)

返回值	23	22	21	20	19	18	17	16
3	Off	Off	Off	Off	Off	Off	On	On
7	Off	Off	Off	Off	Off	On	On	On
32	Off	Off	On	Off	Off	Off	Off	Off
255	On	On	On	On	On	On	On	On

参阅
InBCD、MemIn、MemOff、MemOn、MemSw、Off、On、OpBCD、Oport、Out、Sw、Wait

In函数使用示例
下述程序例基于输入位20、21、22、23已连接传感器设备并且在各自返回表示“准备OK”的打开信号后启动应用程序这一假定进行记述。在该程序例中获取字节端口2的8个输入位的状态，并在确认输入位20、21、22、23全部打开后进入以下步骤。如有任一输入位不是ON(任一输入位返回“1”)，将显示错误信息并停止任务。

在这种情况下，In(2)的返回值必须大于240，以全部打开输入位20、21、22、23。为确认其值而将变量“var1”与数值239进行比较。(这里不识别输入位16、17、18、19，只要返回值是240～255内的值，将继续执行程序。)

```
Function main
    Integer var1
    var1 = In(2)   '获取字节端口2的8个输入位的状态
    If var1 > 239 Then
        Go P1
```

```
        Go P2
        '在此处执行其它动作命令
        '.
        '.
    Else
        Print "Error in initialization!"
        Print "Sensory Inputs not ready for cycle start"
        Print "Please check inputs 20,21,22 and 23 for"
        Print "proper state for cycle start and then"
        Print "start program again"
    EndIf
Fend
```

虽然无法从命令窗口设置输入，但是可以进行确认。下例以打开输入位1、5、15为前提。将其他输入全部关闭。

```
> print In(0)
34
> print In(1)
128
> print In(2)
0
```

3.13.4 InBCD函数

该函数以8位为一组并通过BCD返回输入端口的状态。

格式
InBCD (端口编号)

参数

端口编号
　　指定I/O的输入字节。

返回值
以BCD(0～9)返回输入端口(0～99)的状态。

说明
InBCD通过BCD同时读取8个输入列。通过InBCD命令的参数“端口编号”指定读取哪个位组的状态。例如，端口编号=0时读取0～7输入位的状态；端口编号=1时读取8～15输入位的状态。

以BCD返回8个输入位的状态。返回值是0～99的1位或2位的数值。第1位(10的位)对应按端口编号指定的8个输入位中的高4位。第2位(1的位)对应按端口编号指定的8个输入位中的低4位。

由于BCD的各个位的有效值在0～99范围，因此，不能满足I/O的所有组合。下表显示了端口编号=0时的部分可能的I/O状态组合和返回值。

输入数值(输入编号)

返回值	7	6	5	4	3	2	1	0
01	Off	Off	Off	Off	Off	Off	Off	On
02	Off	Off	Off	Off	Off	Off	On	Off
03	Off	Off	Off	Off	Off	Off	On	On
08	Off	Off	Off	Off	On	Off	Off	Off
09	Off	Off	Off	Off	On	Off	Off	On
10	Off	Off	Off	On	Off	Off	Off	Off
11	Off	Off	Off	On	Off	Off	Off	On
99	On	Off	Off	On	On	Off	Off	On

只能用10进制数指定BCD。这意味着在特定端口将所有输入位同时打开时，使用BCD的InBCD命令无法计算有效值。InBCD的最大返回值是“99”。通过上表可以看出，返回值是“99”时，不是所有输入都是ON。返回值是“99”时，输入位0、3、4、7为ON，而其他位全部OFF。

注意

- InBCD与In的不同

  在SPEL+中，InBCD与In命令有几点重要区别，如下所示为它们的不同之处。

- InBCD命令通过BCD指定8个输入位的返回值。BCD不能使用 &HA、&HB、&HC、&HD、&HE、&HF 等的值，所以无法满足8个输入位的所有组合。

- In命令与InBCD命令的功能非常相似，可以对全部8个输入位返回返回值。(In命令的返回值是0～255，而InBCD的返回值是0～99。)因此，可以满足1组8个输入位的所有组合。

参阅

In、MemOff、MemOn、MemOut、MemSw、Off、On、OpBCD、Oport、Out、Sw、Wait

InBCD使用示例

如下所示为利用命令窗口的简单操作示例。

假设输入位0、4、10、16、17、18 为On，其他输入位为Off。

```
> Print InBCD(0)
11
> Print InBCD(1)
04
> Print InBCD(2)
07
>
```

3.13.5 Inertia

用于设置机器人的负载惯性和离心率。

格式
(1) Inertia [负载惯性] [,离心率]

(2) Inertia

参数

负载惯性
以数值或表达式指定包括末端夹具和工件在内的前端关节中心周围的惯性力矩(惯性)(实数，单位：kgm2)。可省略，但不能只省略负载惯性。
离心率
以数值或表达式指定距末端夹具及工件的重心前端关节中心的距离(实数，单位：mm)。可省略。

结果
如果省略参数，则显示当前设置的惯性参数。

如果省略[离心率]，侧输入的[负载惯性]会被设置，并设置[离心率]的默认值。

所以不能只省略[负载惯性]。

当负载惯性和离心率的设定小于实际值时，则会导致加速值和减速值过生，从而可能会损坏机械手。并且可能会导致错误或发生碰撞，不仅无法正常发挥机械手的功能，还能会会由于皮带条吃而缩短零件寿命或导致错位。

说明
指定前端关节周围的惯性力矩时，使用Inertia语句。由此，可以适当地补偿前端关节的加减速以及伺服增益。此外，还可以通过离心率参数指定前端关节中心到夹具末端和工件重心的距离。

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

还可以通过“负载、惯性、离心率/偏移量测量实用程序”进行设置。

详细信息请参阅以下手册。
《Epson RC+ 用户指南 - 负载、惯性、离心率/偏移量测量实用程序》

注意

- 即使关闭电源Inertia的设置也不会更改

  Inertia的值一旦设置就会存储在控制器中。即使关闭电源也不会改变。

  如果未设置任何内容，则将使用先前设置的值。

参阅
Inertia函数

有关夹具控制命令的详细信息，请参阅以下手册。
《Hand功能》

Inertia使用示例

```
Inertia 0.02, 1
```

3.13.6 Inertia函数

返回惯性的参数值。

格式

Inertia (参数编号)

参数

参数编号
指定下述整数值。

- 0：如果机器人支持惯性参数，则返回“1”；如果不支持，则返回“0”。
- 1：返回负载惯性。(单位：kgm2)
- 2：返回离心率。(单位：mm)

返回值

以实值返回指定的设置。

参阅

Inertia

有关夹具控制命令的详细信息，请参阅以下手册。
《Hand功能》

Inertia函数使用示例

```
Real loadInertia, eccentricity

loadInertia = Inertia(1)
eccentricity = Inertia(2)
```

3.13.7 InPos函数

返回机器人的定位状态。

格式
InPos

返回值
- 完成定位时：True
- 机器人动作中：False

参阅
CurPos、FindPos、WaitPos

InPos函数使用示例

```
Function main

  P0 = XY(0, -100, 0, 0)
  P1 = XY(0, 100, 0, 0)

  Xqt MonitorPosition
  Do
    Jump P0
    Wait .5
    Jump P1
    Wait .5
  Loop

Fend

Function MonitorPosition

  Boolean oldInPos, pos

  Do
    Pos = InPos
    If pos <> oldInPos Then
      Print "InPos = ", pos
    EndIf
    oldInPos = pos
  Loop

Fend
```

3.13.8 Input

用于接收显示装置的输入并保存到变量中。

格式
Input 变量名 [, 变量名, 变量名, ...]

参数

变量名
　　指定变量名。指定多个变量时，用“,”进行分隔。我们将此时的“,”称为分隔符。

说明
用于接收显示装置的数据并代入到指定的变量中。

执行命令时，将在显示装置上显示“?”提示符。输入数据后，在键盘上按下回车键。

注意

- 数值输入规则

  进行数值输入时，如果有分隔符以外的非数值数据，将舍去该非数值数据及其以后的数据。

- 字符串输入规则

  代入到字符串中时，将把数字和字母作为字符处理。

- 与其他的Input命令有关的规则
  - 为代入对象指定多个变量时，各个要代入的数值数据必须用分隔符“,”分隔。
  - 虽然可以指定数值变量和字符串变量，但是输入数据类型必须要符合代入对象的变量类型。

易引起的错误

- 指定的变量数与输入数据的数量不一致时

  一指定多个变量，输入数据的数量就必须与指定的变量数一致。如果命令指定的变量数与从键盘接收的数值数据的数量不一致，将出现错误2505。

参阅
Input #, Line Input, Line Input #, Print, String

Input使用示例
下例为简单的Input语句。

```
Function InputNumbers
    Integer A, B, C

    Print "Please enter 1 number"
    Input A
    Print "Please enter 2 numbers separated by a comma"
    Input B, C
    Print "A = ", A
    Print "B = ", B, "C = ", C
Fend
```

如果执行上述程序，将进行下述对话。

```
Please enter 1 number
?-10000
Please enter 2 numbers separated by a comma
?25.1, -10000
A = -10000
B = 25 C = -10000
```

3.13.9 Input

用于从文件、通信端口、数据库或装置输入字符串或数值数据并保存到变量中。

格式

Input #端口编号, 变量名 [, 变量名, 变量名, ...]

参数

端口编号

是表示文件、通信端口、数据库或装置的ID编号。文件编号是由ROpen、WOpen、AOpen等语句指定的编号。通信端口编号是由OpenCom(RS-232C)或OpenNet(TCP/IP)语句指定的编号。数据库编号是由OpenDB语句指定的编号。

装置ID为以下数值。

- 21 RC+
- 24 TP(仅TP1)
- 20 TP3

变量名

指定接收数据的变量名。

说明

Input # 命令从端口编号指定的设备接收数值或字符串数据，并输入到指定该数据的变量中。

注意

- 支持的控制器型号

  若在T/VT系列中指定RS-232C端口时会发生错误。

- 数值输入规则

  进行数值输入时，如果有分隔符以外的非数值数据，将舍去该非数值数据及其以后的数据。

- 字符串输入规则

  代入到字符串中时，将把数字和字母作为字符处理。

- 最大数据长度

  本命令一次可处理的最大数据长度为256 Byte。但对象为数据库时，最大数据长度为4096 Byte。对象为通信端口(TCP/IP)时，最大数据长度为1024 Byte。

- 与其他的Input # 命令有关的规则

  - 为代入对象指定多个变量时，各个要代入的数值数据必须用分隔符“,”或空白(“ ”)分隔。
  - 指定多个字符串变量以及指定数值变量和字符串变量两者时，数值数据必须用分隔符“,”或空白(“ ”)分隔，而字符串数据必须用分隔符“,”分隔。
  - 输入数据类型必须符合代入对象的变量类型。
  - 使用通信端口交接控制器之间的字符串变量、数值变量。
  - 发送侧(任一模式即为OK。)

```
Print #PortNum, "$Status,", InData, OutData
Print #PortNum, "$Status", ",",InData, OutData
```

- 接收侧

```
Input #PortNum, Response$, InData, OutData
```

易引起的错误

- 指定的变量数与输入数据的数量不一致时

  如果命令指定的变量数与从设备接收的数值数据的数量不一致，将出现错误2505。

参阅

Input、Line Input、Line Input #、Print #、Read、ReadBin

Input #使用示例

下述为使用Input #语句的简单示例。

```
Function GetData
    Integer A
    String B$

    OpenCom #1
    Print #1, "Send"
    Input #1, A   '从端口#1获取数值
    Input #1, B$  '从端口#1获取字符串
    CloseCom #1
Fend
```

3.13.10 InputBox

显示输入对话框。等待用户输入文本或单击按钮，并返回输入内容。

格式

InputBox 提示符, 标题, 默认值, 输入字符串

参数

提示符
　　显示到对话框的信息的字符串
标题
　　显示到对话框标题栏的字符串
默认
　　文本框中默认显示的字符串未设置默认值时，设为空白。(" ")
输入字符串
　　设置用户输入的字符串的字符串型变量如果用户单击取消，此变量将被设为“@”。

说明

InputBox显示对话框，等待至用户单击[OK]按钮或[取消]按钮。在参数输入字符串中将设置用户输入的字符串。

参阅

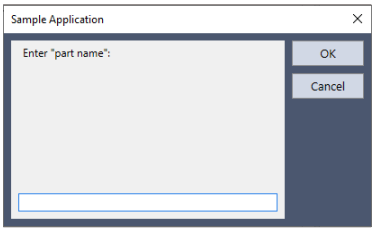
MsgBox

InputBox使用示例

此程序为InputBox语句的使用示例。

```
Function GetPartName$ As String
    String prompt$, title$, answer$

    prompt$ = "Enter " + Chr$(34) + "part name" + Chr$(34) + ":"
    title$ = "Sample Application"
    InputBox prompt$, title$, "", answer$
    If answer$ <> "@" Then
        GetPartName$ = answer$
    EndIf
Fend
```

下图为上述程序示例的结果。



限制事项

如果参数的提示、标题和默认值中包含半角逗号“,”，将无法正确显示字符串。请使用不含半角逗号的字符串。

3.13.11 InReal函数

用于将2个字(32位)的输入数据作为32位浮点数据(符合IEEE754标准)来读取。

格式

InReal (字端口编号)

参数

字端口编号
　　指定I/O输入字端口。

返回值

用于以Real型实数返回输入端口状态。

说明

从字端口编号指定的输入字端口获取2个输入字的值作为IEEE754的Real型的值。可在字端口编号参数中使用输入字标签。

本函数不能作为Wait命令以及Till、Find、Sense的条件使用。

参阅

In、InW、InBCD、Out、OutW、OpBCD、OutReal

InW函数使用示例

```
Real realVal

realVal = InReal(32)
```

3.13.12 InsideBox函数

用于返回进入检测区域的检测状态。

格式

InsideBox (区域编号 [, 机器人编号 | All])

参数

区域编号
　　指定返回状态的进入检测区域编号(1～15的整数)。
机器人编号
　　以整数值指定要检测的机器人编号。已省略机器人编号时，以当前选择的机器人为对象。指定All时，进入1台机器人也会返回True。

返回值

在指定进入区域中进入机器人的卡爪工具位置时返回True；反之返回False。

参阅

Box、BoxClr、BoxDef、GetRobotInsideBox、InsidePlane

注意

在Epson RC+5.0中可以与Wait命令组合等待InsideBox函数的结果，而在Epson RC+ 6.0, RC+ 7.0, Epson RC+ 8.0中不能与Wait命令组合。在这种情况下，请使用GetRobotInsideBox函数以替代InsideBox函数。

对应表

RC+版本	机器人控制器	Wait	Till、Find、Sense、Trap	Print等左述以外的命令/分支判定处理	GetRobotInsideBox函数的利用
RC+ 8.0	RC700系列	不可	不可	可	均可
RC+ 8.0	RC90系列	不可	不可	可	均可
RC+ 7.0	RC700系列	不可	不可	可	均可
RC+ 7.0	RC90系列	不可	不可	可	均可
RC+ 6.0	RC620	不可	不可	可	均可
RC+ 5.0	RC90系列	可	不可	可	不可

- 不可：不可利用的组合
- 可：可利用的组合
- 均可：可用于Wait、Till、Find、Sense、Trap、Print等的显示、分支判定处理

InsideBox函数使用示例

下述程序例为判断3号区域是否进入了机器人1的示例。

```
Function PrintInsideBox
If InsideBox(3,1) = True Then
Print "Inside Box3"
Else
```

```
Print "Outside Box3"
EndIf
Fend
```

3.13.13 InsidePlane函数

用于返回进入检测平面的检测状态。

格式

InsidePlane (平面编号 [, 机器人编号 | All])

参数

平面编号
　　指定返回状态的进入检测平面的编号(1～15的整数)。
机器人编号
　　以整数值指定要检测的机器人编号。已省略机器人编号时，以当前选择的机器人为对象。指定All时，进入1台机器人也会返回True。

返回值

如果机器人的卡爪工具位置进入到指定的进入检测平面，则会返回“True”；如果未进入，则会返回“False”。

参阅

InsideBox、GetRobotInsidePlane、Plane、PlaneClr、PlaneDef

注意

在Epson RC+5.0中可以与Wait命令组合等待InsidePlane函数的结果，而在Epson RC+ 6.0, RC+ 7.0, Epson RC+ 8.0中不能与Wait命令组合。在这种情况下，请使用GetRobotInsidePlane函数以替代InsidePlane函数。

对应表

RC+版本	机器人控制器	Wait	Till、Find、Sense、Trap	Print等左述以外的命令/分支判定处理	GetRobotInsideBox函数的利用
RC+ 8.0	RC700系列	不可	不可	可	均可
RC+ 8.0	RC90系列	不可	不可	可	均可
RC+ 7.0	RC700系列	不可	不可	可	均可
RC+ 7.0	RC90系列	不可	不可	可	均可
RC+ 6.0	RC620	不可	不可	可	均可
RC+ 5.0	RC90系列	可	不可	可	不可

- 不可：不可利用的组合
- 可：可利用的组合
- 均可：可用于Wait、Till、Find、Sense、Trap、Print等的显示、分支判定处理

InsidePlane函数使用示例

下述程序示例为判断3号平面是否进入了机器人1的示例。

```
Function PrintInsidePlane
If InsidePlane(3,1) = True Then
Print "Inside Plane3"
Else
```

```
Print "Outside Plane3"
EndIf
Fend
```

3.13.14 InStr函数

用于从字符串中检索字符串并返回发现的位置。

格式

InStr (字符串, 检索字符串)

参数

字符串
　　指定检索的字符串。
检索字符串
　　指定要检索的字符串。

返回值

发现要检索的字符串时返回该位置，未发现时返回-1。

参阅

Mid$

Instr函数使用示例

```
Integer pos

pos = InStr("abc", "b")
```

3.13.15 Int函数

将实值转换为整数值。返回舍弃实值小数点以下的值。

格式
Int (数值)

参数

数值
　　指定实值。

返回值
将参数设置的实值转换为整数值并返回。

说明
Int(数值)返回舍弃实值小数点以下的值。

注意

- 是小于1的数值时(负值时)

  如果参数是小于1的值，将返回向负方向舍入的值。(数值是-1.35时，返回-2。)

参阅
Abs、Atan、Atan2、Cos、Mod、Not、Sgn、Sin、Sqr、Str$、Tan、Val

Int函数使用示例
下例所示为通过命令窗口执行的情况。

```
> Print Int(5.1)
5
> Print Int(0.2)
0
> Print Int(-5.1)
-6
>
```

3.13.16 Int32

用于声明Int32型变量。(4字节整数型变量)

格式

Int32 变量名 [(数组变量的最大下标)] [, 变量名 [(数组变量的最大下标)]...]

参数

变量名
　　指定要进行变量声明的变量名。
数组变量的最大下标
　　可利用数组变量的最大下标声明到最高三维。使用下述格式。可省略。
　　(最大下标1, [最大下标2], [最大下标3])
　　由于下标从0开始，因此元素数为最大下标加上1。在所有元素数不超过以下最大值的范围内指定各最大下标。

- 本地变量：2,000
- 备份变量(Global Preserve)：4,000
- 全局变量和模块变量：100,000

说明

Int32用于声明整数型变量。整数型变量的范围为 -2147483648～2147483647。在Function开头声明本地变量。在Function之外声明全局变量和模块变量。

参阅

Boolean, Byte, Double, Global, Int64, Integer, Long, Real, Short, String, UByte, UInt32, UInt64, UShort

Int32使用示例

如下所示为使用Int32声明整数型变量的程序的示例。

```
Function int32test
    Int32 A(10)          'Int32型一维数组
    Int32 B(10, 10)      'Int32型二维数组
    Int32 C(5, 5, 5)     'Int32型三维数组
    Int32 var1, arrayvar(10)
    Integer i
    Print "Please enter an Integer Number"
    Input var1
    Print "The Integer variable var1 = ", var1
    For i = 1 To 5
        Print "Please enter an Integer Number"
        Input arrayvar(i)
        Print "Value Entered was ", arrayvar(i)
    Next i
Fend
```

3.13.17 Int64

用于声明Int64型变量。(8字节整数型变量)

格式

Int64 变量名 [(数组变量的最大下标)] [, 变量名 [(数组变量的最大下标)]...]

参数

变量名
　　指定要进行变量声明的变量名。
数组变量的最大下标
　　可利用数组变量的最大下标声明到最高三维。使用下述格式。可省略。
　　(最大下标1, [最大下标2], [最大下标3])
　　由于下标从0开始，因此元素数为最大下标加上1。在所有元素数不超过以下最大值的范围内指定各最大下标。

- 本地变量：2,000
- 备份变量(Global Preserve)：4,000
- 全局变量和模块变量：100,000

说明

Int64用于声明整数型变量。整数型变量的范围为 -9223372036854775808～9223372036854775807。在Function开头声明本地变量。在Function之外声明全局变量和模块变量。

参阅

Boolean, Byte, Double, Global, Int32, Integer, Long, Real, Short, String, UByte, UInt32, UInt64, UShort

Int64使用示例

如下所示为使用Int64声明整数型变量的程序的示例。

```
Function int64test
    Int64 A(10)          'Int64型一维数组
    Int64 B(10, 10)      'Int64型二维数组
    Int64 C(5, 5, 5)     'Int64型三维数组
    Int64 var1, arrayvar(10)
    Integer i
    Print "Please enter an Integer Number"
    Input var1
    Print "The Integer variable var1 = ", var1
    For i = 1 To 5
        Print "Please enter an Integer Number"
        Input arrayvar(i)
        Print "Value Entered was ", arrayvar(i)
    Next i
Fend
```

3.13.18 Integer

用于声明Integer型变量。(2字节整数型变量)

格式

Integer 变量名 [(数组变量的最大下标)] [, 变量名 [(数组变量的最大下标)]...]

参数

变量名
　　指定要进行变量声明的变量名。
数组变量的最大下标
　　可利用数组变量的最大下标声明到最高三维。使用下述格式。可省略。
　　(最大下标1, [最大下标2], [最大下标3])
　　由于下标从0开始，因此元素数为最大下标加上1。在所有元素数不超过以下最大值的范围内指定各最大下标。

- 本地变量：2,000
- 备份变量(Global Preserve)：4,000
- 全局变量和模块变量：100,000

说明

Integer用于声明整数型变量。整数型变量的范围为-32768～32767。在Function开头声明本地变量。在Function之外声明全局变量和模块变量。

参阅

Boolean, Byte, Double, Global, Int32, Int64, Long, Real, Short, String, UByte, UInt32, UInt64, UShort

Integer使用示例

如下所示为使用Integer声明整数型变量的程序的示例。

```
Function inttest
    Integer A(10)          'Integer型一维数组
    Integer B(10, 10)      'Integer型二维数组
    Integer C(5, 5, 5)     'Integer型三维数组
    Integer var1, arrayvar(10)
    Integer i
    Print "Please enter an Integer Number"
    Input var1
    Print "The Integer variable var1 = ", var1
    For i = 1 To 5
        Print "Please enter an Integer Number"
        Input arrayvar(i)
        Print "Value Entered was ", arrayvar(i)
    Next i
Fend
```

3.13.19 InW函数

用于以字为单位返回输入端口状态。字端口由16个输入位构成。

格式
InW (字端口编号)

参数

字端口编号
　　指定I/O的字端口。

返回值
返回输入端口状态(0～65535的Long型整数)。

注意

- 关于实时I/O的输入位等字端口编号的规则

  字端口=1、3、17、19时，以0～255整数值返回输入端口的状态。

  实时I/O的输入位不被反映。

参阅
In、Out、OutW

InW函数使用示例

```
Long word0

word0 = InW(0)
```

3.13.20 IODef函数

用于返回指定的输入输出的位、字节、字或I/O标签。

格式

IODef (IO类型, IO带宽, 端口编号)

IODef (IO标签)

参数

IO类型

表示I/O类型的整数值

- 0 - 输入
- 1 - 输出
- 2 - 存储器

IO带宽

表示端口带宽的整数值：1(位)、8(字节)、或16(字)

端口编号

表示返回标签的位、字节或字的整数值

IO标签

以字符串指定标准I/O或存储器I/O标签的字符串表达式。

返回值

定义指定输入输出的位、字节、字或I/O标签时返回“True”，除此之外返回“False”。

参阅

IOLabel$、IONumber

IODef函数使用示例

```
Integer i

     For i = 0 To 15
       If IODef( 0, 1, i) = TRUE Then
         Print "Port " , i, " is defined"
       Else
         Print "Port " , i, " is undefined"
       EndIf
     Next i
```

3.13.21 IOLabel$函数

用于返回指定的输入输出的位、字节、字的I/O标签。

格式

IOLabel$ (IO类型, IO带宽, 端口编号)

参数

IO类型

表示I/O类型的整数值

- 0 - 输入
- 1 - 输出
- 2 - 存储器

IO带宽

表示端口带宽的整数值：1(位)、8(字节)、16(字)

端口编号

表示返回标签的位、字节或字的整数值

返回值

返回包括标签在内的字符串。

参阅

PLabel$、IONumber、IODef

IOLabel$函数使用示例

```
Integer i

For i = 0 To 15
  Print "Input ", i, ": ", IOLabel$(0, 1, i)
Next i
```

3.13.22 IONumber函数

用于返回指定的I/O标签的I/O端口编号。

格式

IONumber (IO标签)

参数

IO标签
以字符串指定标准I/O或存储器I/O标签的字符串表达式。

返回值

用于返回指定的I/O标签的I/O端口编号(位、字节、字)。如果不存在指定的I/O标签，将发生错误。

参阅

IOLabel$、IODef

IONumber函数使用示例

```
Integer IObit

IObit = IONumber("myIO")

IObit = IONumber ("Station" + Str$(station) + "InCycle")
```

3.14 J

3.14.1 J1Angle

用于设置点的J1Angle属性。

格式

(1) J1Angle 指定点 [,设置值]

(2) J1Angle

参数

点指定
　　以P编号、P(表达式)、点标签之一进行指定。
设置值
　　以实值指定设置值。可省略。

结果

J1Angle属性只用于RS系列与N系列的机器人。

如果省略设置值参数，将显示相对于指定点的J1Angle值。当省略了双方的参数，将显示当前机器人位置的J1Angle值。

- RS系列：指定点的X坐标值为“0”并且Y坐标值也为“0”的奇点的第1关节角度值。为未形成奇点的点时，J1Angle属性值没有意义。
- N系列：指定第1关节/第4关节/第6关节的轴心位于直线上、第1关节与第6关节的轴心位于直线上或第1关节与第4关节的轴心位于直线上时的第1关节角度值。为未形成奇点的点时，J1Angle属性值没有意义。

参阅

Hand、J1Angle函数、J1Flag、J2Flag、J4Angle、J4Angle函数

J1Angle使用示例

```
J1Angle P0, 10.0
J1Angle P(mypoint), 0.0
```

3.14.2 J1Angle函数

用于返回点的J1Angle属性。

格式

J1Angle [(指定点)]

参数

点指定

以表达式指定点。可省略。如果省略，将返回机器人当前位置的J1Angle设置。

返回值

J1Angle属性仅用于RS系列与N系列的机器人。

- RS系列：以实值返回点的X坐标值为“0”并且Y坐标值也为“0”的奇点的第1关节角度值。
- N系列：以实值返回第1关节/第4关节/第6关节的轴心位于直线上、第1关节与第6关节的轴心位于直线上或第1关节与第4关节的轴心位于直线上时的第1关节角度值。

参阅

Hand、J1Angle、J1Flag、J2Flag、J4Angle、J4Angle函数

J1Angle函数使用示例

```
Print J1Angle(pick)
Print J1Angle(P1)
Print J1Angle
```

3.14.3 J1Flag

用于设置点的J1Flag 属性。

格式

(1) J1Flag 指定点 [,设置值]

(2) J1Flag

参数

点指定

以P编号、P(表达式)、点标签之一进行指定。

设置值

以整数或表达式指定下述任一值。可省略。

- RS系列时
  - 0 (/J1F0)：J1的范围：-90~+270(单位：度)
  - 1 (/J1F1)：J1的范围：-270~-90或+270~+450(单位：度)
- C8、C12系列时
  - 0 (/J1F0)：J1的范围：-180~+180(单位：度)
  - 1 (/J1F1)：J1的范围：-240~-180或+180~+240(单位：度)

结果

J1Flag属性指定相对于1点的第1关节的值的范围。如果省略设置值参数，将显示相对于指定点的J1Flag值。当省略了双方的参数时，将显示当前机器人位置的J1Flag值。

参阅

Hand、J1Flag函数、J2Flag

J1Flag使用示例

```
J1Flag P0, 1
J1Flag P(mypoint), 0
```

3.14.4 J1Flag函数

用于返回点的J1Flag 属性。

格式

J1Flag [(指定点)]

参数

点指定

以表达式指定点。可省略。如果省略，将返回机器人当前位置的J1Flag设置值。

返回值

- 0: /J1F0
- 1: /J1F1

参阅

Hand、J1Flag、J2Flag

J1Flag函数使用示例

```
Print J1Flag(pick)
Print J1Flag(P1)
Print J1Flag
Print J1Flag(Pallet(1, 1))
```

3.14.5 J2Flag

用于设置点的J2Flag属性。

格式

(1) J2Flag 指定点 [,设置值]

(2) J2Flag

参数

点指定
以P编号、P(表达式)、点标签之一进行指定。
设置值
以整数或表达式指定下述任一值。可省略。

- 0 (/J2F0)：J2的范围：-180~+180(单位：度)
- 1 (/J2F1)：J2的范围：-360~-180或+180~+360(单位：度)

结果

J2Flag属性指定相对于1点的第2关节的值的范围。如果省略设置值参数，将显示相对于指定点的J2Flag值。当省略了双方的参数，将显示当前机器人位置的J2Flag值。

参阅

Hand、J1Flag、J2Flag函数

J2Flag使用示例

```
J2Flag P0, 1
J2Flag P(mypoint), 0
```

3.14.6 J2Flag函数

用于返回点的J2Flag属性。

格式

J2Flag [(指定点)]

参数

点指定

以表达式指定点。可省略。如果省略，将返回机器人当前位置的J2Flag设置值。

返回值

- 0: /J2F0
- 1: /J2F1

参阅

Hand、J1Flag、J2Flag

J2Flag函数使用示例

```
Print J2Flag(pick)
Print J2Flag(P1)
Print J2Flag
Print J2Flag(P1 + P2)
```

3.14.7 J4Angle

用于设置点的J4Angle属性。

格式

(1) J4Angle 指定点 [,设置值]

(2) J4Angle

参数

点指定
　　以P编号、P(表达式)、点标签之一进行指定。
设置值
　　以实值指定设置值。可省略。

结果

J4Angle属性仅用于N系列的机器人。

指定第4关节与第6关节的轴心位于直线上时的第4关节角度值。

为未形成奇点的点时，J4Angle属性值没有意义。

如果省略设置值参数，则会显示相对于指定点的J4Angle值。如果省略两个参数，则会显示当前机器人位置的J4Angle值。

参阅

Hand、J1Angle、J1Angle函数、J4Angle函数

注意

如下所示，同时使用J4Flag与J4Angle时，以J4Angle为优先。

```
J4Angle P0,0
J4Flag P0,1
```

J4Angle使用示例

```
J4Angle P0, 10.0
J4Angle P(mypoint), 0.0
```

3.14.8 J4Angle函数

用于返回点的J4Angle属性。

格式

J4Angle [(指定点)]

参数

点指定

以表达式指定点。可省略。如果省略，将返回机器人当前位置的J4Angle设置值。

返回值

以实值返回第4关节与第6关节的轴心位于直线上时的第4关节角度值。

J4Angle属性仅用于N系列的机器人。

参阅

Hand、J1Angle、J1Angle函数、J4Angle

J4Angle函数使用示例

```
Print J4Angle(pick)
Print J4Angle(P1)
Print J4Angle
```

3.14.9 J4Flag

用于设置点的J4Flag属性。

格式

(1) J4Flag 指定点 [,设置值]

(2) J4Flag

参数

点指定
以P编号、P(表达式)、点标签之一进行指定。
设置值
以整数或表达式指定下述任一值。可省略。

- 0 (/J4F0)：J4的范围：-180~+180(单位：度)
- 1 (/J4F1)：J4的范围：-360~-180或+180~+360(单位：度)

结果

J4Flag属性指定相对于1点的第4关节的值的范围。如果省略设置值参数，将显示相对于指定点的J4Flag值。当省略了双方的参数，将显示当前机器人位置的J4Flag值。

参阅

Elbow、Hand、J4Flag函数、J6Flag、Wrist

J4Flag使用示例

```
J4Flag P0, 1
J4Flag P(mypoint), 0
```

3.14.10 J4Flag函数

用于返回点的J4Flag属性。

格式

J4Flag [(指定点)]

参数

点指定

以表达式指定点。可省略。如果省略，将返回机器人当前位置的J4Flag设置值。

返回值

- 0: /J4F0
- 1: /J4F1

参阅

Elbow、Hand、Wrist、J4Flag、J6Flag

J4Flag函数使用示例

```
Print J4Flag(pick)
Print J4Flag(P1)
Print J4Flag
Print J4Flag(Pallet(1, 1))
```

3.14.11 J6Flag

用于设置点的J6Flag属性。

格式

(1) J6Flag 指定点 [,设置值]

(2) J6Flag

参数

点指定
以P编号、P(表达式)、点标签之一进行指定。
设置值
以整数或表达式进行指定。

- 范围：0～127(/J6F0～/J6F127)
- J6相对于指定点的范围如下。
  - (-180 * (设置值+1) < J6 <= -180 *设置值)
  - (180 *设置值< J6 <= 180 * (设置值+1))

结果

J6Flag属性指定相对于1点的第6关节的值的范围。如果省略设置值参数，将显示相对于指定点的J6Flag值。当省略了双方的参数，将显示当前机器人位置的J6Flag值。

参阅

Elbow、Hand、J4Flag、J6Flag函数、Wrist

注意

J6Flag的范围因机型而异。
有关范围，请参阅各机械手册的动作区域。

J6Flag使用示例

```
J6Flag P0, 1
J6Flag P(mypoint), 0
```

3.14.12 J6Flag函数

用于返回点的J6Flag属性。

格式

J6Flag [(指定点)]

参数

点指定

以表达式指定点。可省略。如果省略，将返回机器人当前位置的J6Flag设置值。

返回值

0 ～ 127: /J6F0 ～ /J6F127

参阅

Elbow、Hand、Wrist、J4Flag、J6Flag

J6Flag函数使用示例

```
Print J6Flag(pick)
Print J6Flag(P1)
Print J6Flag
Print J6Flag(P1 + P2)
```

3.14.13 JA函数

用于返回根据关节角度计算的机器人坐标。

格式
JA (第1关节位置, 第2关节位置, 第3关节位置, 第4关节位置 [, 第5关节位置, 第6关节位置] [, 第7关节位置] [, 第8关节位置, 第9关节位置])

参数
第1关节位置～第9关节位置以实值指定关节角度(单位：deg)。为移动关节时，以(单位：mm)进行指定。第5、第6关节位置值为垂直6轴型机器人(包括N系列)和关节型6轴机器人的专用参数。第7关节位置值是关节型7轴机器人的专用参数。第8、第9关节位置值为附加轴S和T关节的专用参数。

补充
如果指定动作范围以外的角度，将发生超出动作范围的错误。

结果
返回由指定关节位置表示的机器人坐标。

说明
以关节角度指定机器人坐标时，使用JA函数。

利用JA函数返回的点为机器人的特殊方向属性时，即使针对该点执行动作命令，也未必与作为JA函数自变量赋予的关节角度一致。要按照由JA函数指定的关节角度进行动作时，需要避免机器人的特殊方向属性。

[例]

```
> go ja(0,0,0,90,0,-90)
> where
WORLD:  X:    0.000 mm  Y:  655.000 mm  Z:  675.000 mm  U:    0.000 deg V:  -90.000
deg W:  -90.000 deg
JOINT:  1:    0.000 deg 2:    0.000 deg 3:    0.000 deg 4:    0.000 deg 5:    0.000
deg 6:    0.000 deg
PULSE:  1:        0 pls 2:        0 pls 3:        0 pls 4:        0 pls 5:        0
pls 6:        0 pls
> go ja(0,0,0,90,0.001,-90)
> where
WORLD:  X:   -0.004 mm  Y:  655.000 mm  Z:  675.000 mm  U:    0.000 deg V:  -90.000
deg W:  -89.999 deg
JOINT:  1:    0.000 deg 2:    0.000 deg 3:    0.000 deg 4:   90.000 deg 5:    0.001
deg 6:  -90.000 deg
PULSE:  1:        0 pls 2:        0 pls 3:        0 pls 4:  2621440 pls 5:       29
pls 6: -1638400 pls
```

参阅
AglToPls、XY

JA函数使用示例

```
P10 = JA(60, 30, -50, 45)
GO JA(135, 90, -50, 90)
P3 = JA(0, 0, 0, 0, 0, 0)
```

3.14.14 Joint

用于在关节坐标系中显示当前机器人位置。

格式

Joint

参阅

Pulse、Where

Joint使用示例

```
>joint
JOINT:  1: -6.905 deg 2: 23.437 deg 3: -1.999 mm  4: -16.529 deg
>
```

3.14.15 JointAccuracy

用于设置并显示关节精度补偿的补偿值。

格式

(1) JointAccuracy 关节编号, 第1设置值, 第2设置值, 第3设置值, 第4设置值

(2) JointAccuracy 关节编号

参数

关节编号
　　指定关节编号。
第1设置值
　　以数值指定第1设置值(整数)。范围为0～2000。
第2设定值
　　以数值指定第2设置值(整数)。范围为0～999。
第3设定值
　　以数值指定第3设置值(整数)。范围为0～2000。
第4设定值
　　以数值指定第4设置值(整数)。范围为0～999。

各机械手的关节精度补偿有效的关节各不相同，不是有效关节时，会发生错误。有关有效关节，请参阅以下手册。
《机械手手册》

返回值

为 (2) 的情况下，会显示对应关节编号的当前关节精度补偿的补偿值。

说明

JointAccuracy用于设置指定关节的补偿值。通过适当地设置补偿值，可提高轨迹精度。

注意

- 除非必要，否则切勿变更JointAccuracy。

  出厂时已对JointAccuracy进行了精密设置。如果在不必要的情况下变更该值，则可能会导致轨迹精度降低。执行校准向导时会自动设置JointAccuracy，因此除非必要，否则切勿变更JointAccuracy。

- 执行Calib与Hofs

  如果在已设置JointAccuracy的状态下执行Calib、Hofs命令，发生变更关节的关节精度补偿的补偿值则会变为“0”。要在不变更JointAccuracy补偿值的状态下变更Hofs值时，请执行HofsJointAccuracy。

参阅

HofsJointAccuracy, Calib, Hofs

JointAccuracy使用示例

如下所示为命令窗口的简单使用示例。针对第1关节设置关节精度补偿的补偿值，第1设置值为“1000”、第2设置值为“420”、第3设置值为“100”、第4设置值为“240”。然后，显示当前第1关节的关节精度补偿的补偿值。

```
> JointAccuracy 1, 1000, 420, 100, 240

> JointAccuracy 1
1000, 420, 100, 240
>
```

3.14.16 JointAccuracy函数

显示关节补偿的补偿值。

格式

JointAccuracy (关节编号, 设定编号)

参数

关节编号

指定关节编号。

设置编号

使用以下常数或整数值(1～4)指定要显示的补偿值。

常数	值	
JAC_PARAM1	1	第1设定值
JAC_PARAM2	2	第2设定值
JAC_PARAM3	3	第3设定值
JAC_PARAM4	4	第4设定值

返回值

返回指定关节设置编号对应的关节精度补偿的补偿值(整数值)。

参阅

JointAccuracy

JointAccuracy使用示例

以下是JointAccuracy函数的使用示例。

```
Function main
    '设置第2关节的精度补偿值
    JointAccuracy 2, 32, 873, 6, 437
    '显示指定关节的精度补偿值
    DisplayJointAccuracy(2)
Fend

Function DisplayJointAccuracy(numJoint As Integer)
    Integer i
    Print "Joint =", numJoint, ", JointAccuracy settings:"
    For i = 1 To 4
        Print "  Param ", i, " = ", JointAccuracy(numJoint, i)
    Next i
Fend

[输出结果]
Joint = 2, JointAccuracy settings:
  Param  1 =  32
  Param  2 =  873
  Param  3 =  6
  Param  4 =  437
```

3.14.17 JRange

用于以脉冲值设置指定关节的容许动作区域。

格式
JRange 关节编号, 下限脉冲值, 上限脉冲值

参数

关节编号
　　以1～9的整数值设置JRange指定的关节编号。附加轴的S轴为8，T轴为9。
下限脉冲值
　　以表达式或数值指定指定关节动作范围的下限脉冲值。
上限脉冲值
　　以表达式或数值指定指定关节动作范围的上限脉冲值。

说明
以上限和下限脉冲值设置指定关节的动作范围。Range命令需要指定所有关节的动作范围，但由于JRange命令用于设置各关节的动作区域，因此，参数数量较少。要确认已设置的动作区域时，使用Range命令。

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

注意

- 请使下限脉冲值≤上限脉冲值

  请勿将JRange命令的下限脉冲值设为大于上限脉冲值。如果下限脉冲值大于上限脉冲值，则会发生错误，导致无法执行动作命令。

- Jrange设置值的变更

  一旦设置由JRange设置的值，则会一直保持，除非利用Range或JRange命令进行变更。电源OFF时，由JRange设置的限制值也不会发生变化。

- 动作区域的上下限

  由于动作区域的上限脉冲值因各机械手的类型而异，因此，有关区域上限脉冲值的设置，请参阅以下手册。
  《机械手手册》

参阅
Range、JRange函数

JRange使用示例
如下所示为利用命令窗口的操作示例。

```
> JRange 2, -6000, 7000     '定义第2关节动作范围

> JRange 1, 0, 7000     '定义第1关节动作范围
```

3.14.18 JRange函数

用作返回已指定关节容许动作区域脉冲值(范围设置值)的函数。

格式

Jrange (关节编号, 参数编号)

参数

关节编号
以表达式或数值指定要引用的关节编号(1～9的整数)。附加轴的S轴为8，T轴为9。
参数编号
以整数值指定下述2个值之一。

- 1：指定下限脉冲值
- 2：指定上限脉冲值

返回值

返回指定关节的范围设置值(整数值，单位：脉冲)。

参阅

Range、JRange

JRange函数使用示例

```
Long i, oldRanges(3, 1)

For i = 0 To 3
    oldRanges(i, 0) = JRange(i + 1, 1)
    oldRanges(i, 1) = JRange(i + 1, 2)
Next i
```

3.14.19 JS函数

是用于在执行Jump Sense、Jump3 Sense、Jump3CP Sense、JumpTLZ Sense之后返回Sense输入是否成立的函数。

格式
JS

返回值
返回“True”或“False”。

- True：Sense输入条件成立并且机械臂停在目标坐标上方时，JS将返回“True”。
- False：机械臂到达由Jump命令设置的目标位置并完成正常动作时，JS将返回“False”。

说明
JS与Jump或Sense命令配套使用。JS命令的使用目的在于，返回在进行Jump命令指示的动作期间(由Sense设置的)输入条件是否成立。如果输入条件成立，JS将返回“True”。如果输入条件不成立，机械臂到达目标位置，JS将返回“False”。

JS只是确认状态的命令，没有使其发生动作或在动作期间检查任何输入的功能。要指示动作时，使用Jump命令；根据情况，要在Jump命令下的动作期间检查特定输入时，使用Sense命令。

注意

- JS仅与此前使用的Jump命令、Jump3命令、Jump3CP、JumpTLZ命令组合使用

  JS仅对此前使用的Jump命令进行输入检查(由Sense命令指示)。如果下一Jump命令启动，JS命令则会返回有关新Jump命令的状态。不能返回最初Jump命令的状态数据。务必在其后对需要状态检查的Jump命令进行JS检查。

参阅
JT, Jump, Jump3, Jump3CP, JumpTLZ, Sense

JS函数使用示例

```
Function SearchSensor As Boolean
    Sense Sw(5) = On

    Jump P0
    Jump P1 Sense
    If JS = TRUE Then
        Print "Sensor was found"
        SearchSensor = TRUE
    EndIf
Fend
```

3.14.20 JT函数

用作返回此前Jump、Jump3、Jump3CP、JumpTLZ动作结果的函数。

格式

JT

返回值

设置或清除下述格式的Long型数值的位。

- Bit 0：上升动作开始时或上升动作量0时，为1
- Bit 1：水平动作开始时或水平移动量0时，为1
- Bit 2：下降动作开始时或下降动作量0时，为1
- Bit 16：上升动作完成时或上升动作量0时，为1
- Bit 17：水平动作完成时或水平动作量0时，为1
- Bit 18：下降动作完成时或下降动作量0时，为1

说明

此前执行的Jump命令可用作调查由Sense、Till、Abort等指定的动作停止条件是否成立。

参阅

JS, Jump, Jump3, Jump3CP, JumpTLZ, Sense, Till

JT函数使用示例

```
Function SearchTill As Boolean

    Till Sw(5) = On

    Jump P0
    Jump P1 Till
    If JT And 4 Then
        Print "Motion stopped during descent"
        SearchTill = TRUE
    EndIf
Fend
```

3.14.21 JTran

用于进行无需原点恢复的仅1关节的PTP动作。

格式

JTran 关节编号, 移动量

参数

关节编号
　　以整数值指定要动作的关节编号。附加轴的S轴为8，T轴为9。
移动量
　　指定实数。为旋转关节时，以(单位：deg)进行指定；为移动关节时，以(单位：mm)进行指定。

说明

JTran仅用于对指定的关节从当前位置进行指定量的移动。

参阅

Go、Jump、Move、PTran

JTran使用示例

```
JTran 1, 20
```

3.14.22 Jump

用于通过门控运动(首先垂直上升，然后水平移动，最后垂直下降的门型动作)使机械臂从当前位置向指定位置进行PTP动作。

格式

Jump 目标坐标 [C Arch编号] [LimZ [Z坐标值]] [CP] [{Sense|Till|Find}] [!并行处理!] [SYNC]

参数

目标坐标
　　以点数据指定目标位置。
Arch编号
　　Arch编号指定在Jump的Arch运动形状中使用哪个Arch表格的设置。请在Arch编号的开头附加字母“C”号。有效值为C0～C7。可省略。
Z坐标值
　　请考虑为利用Jump命令进行动作期间，第3关节可移动的最大值(Z限制值)、或利用Jump命令得到的Z成分的高度限制值。如果是有效的第3关节坐标值，则可为任何值。可省略。
CP
　　指定路径运动。可省略。
Sense、Till或Find
　　记述Sense、Till或Find表达式。可省略。

```
Sense | Till | Find
Sense Sw(表达式) = {On | Off}
Till Sw(表达式) = {On | Off}
Find Sw(表达式) = {On | Off}
```

!并行处理!
　　利用Jump命令进行动作期间，可添加I/O或用于执行其它命令的并行处理语句。可省略。
SYNC
　　预约动作命令。在通过SyncRobots开始动作之前，机器人不进行动作。

说明

Jump命令用于通过所谓的“Arch运动(Arch型动作)”将机械臂从当前位置移动到目标坐标。也就是说，可考虑为1次可进行3个动作的语句。比如，如果定义Arch编号，1次Jump命令则进行下述3个动作。

1. 首先，仅第3关节动作到Jump命令期间的由Arch编号计算的Z轴高度位置。
2. 其次，机械臂在到达由LimZ指定的Z限制位置之前，向Z轴方向上升，同时水平移动到目标坐标。然后，分别进行第1关节、第2关节、第4关节的动作，同时开始向Z轴方向下降。机械臂一直进行动作，直至获取最终的X、Y、U坐标位置。
3. 机械臂仅向Z轴方向移动，直至获取目标Z坐标位置，在获取目标坐标时，Jump命令结束。

由于不能在Jump命令中指定目标坐标(移动的目的位置)，因此，执行Jump命令之前，需要进行示教。利用Accel进行Jump移动的加速和减速。另外，利用Speed控制移动速度。

不能对垂直6轴型机器人(包括N系列)执行Jump。请使用Jump3。

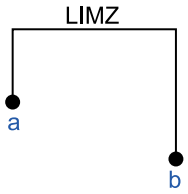
- 关于CP

  如果附加了CP参数，则可在开始动作减速时叠加后续动作命令的加速。此时，不对目标坐标进行定位。

- 关于Arch编号

  可利用Jump命令中指定的Arch编号变更Jump的Arch类型。这样的话，可在第1关节、第2关节、第4关节等各关节动作之前，确定要向Z轴方向移动多少程度。Jump命令中可有效使用的Arch编号为C0～C7之间的值。用户可利用Arch命令来定义Arch表格值相对于C0～C6之间值的设置。但C7始终定义“门控运动”。“门控运动”是指机器人在移动第1关节、第2关节、第4关节等各关节之前，首先仅将第3关节移动到由LimZ定义的坐标位置处。进行这种“门控运动”时，首先移动到由LimZ定义的Z限制值位置，然后，开始第1关节、第2关节、第4关节等各关节动作。第1关节、第2

关节、第4关节等各关节移动到各自的最终目标坐标位置之后，第3关节朝向由目标坐标定义的最终Z坐标位置进行下降动作。下图所示为“门控运动”的动作。



符号	说明
a	当前位置
b	目标坐标

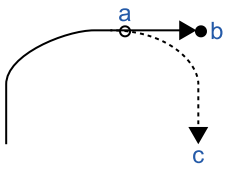
- 关于LimZ

  LimZ Z坐标值指定当前设置的本地坐标系水平移动面上的Z坐标最高值。根据指定的Arch设置，达到LimZ值之前，也许第1关节、第2关节、第4关节等各关节已开始动作，但LimZ值始终用于定义该移动的Z坐标方向的上限值。如果省略LimZ参数，则适用此前(最后)由LimZ指定的最高值。

  由LimZ指定的高度方向限制值为本地机器人坐标系上的Z坐标值。并不是Arm、Tool坐标的Z坐标值。因此，使用作业高度不同的工具或夹具末端时，请充分注意并采取必要的措施。

- 关于Sense

  是可省略的参数之一。Sense用于在第3关节进行最终下降动作之前，检查输入条件或存储器I/O条件等。如果没有问题，则将机器人机械臂停在目标坐标位置上(最终仅保留第3关节动作的位置)，并视为该命令(Jump)执行结束。不过，即使检测到由Sense指定的条件，机械臂也不会立即停止，这点敬请注意。



符号	说明
a	条件检查
b	动作结束
c	目标坐标

  与JS或Stat等命令组合，可确认Sense条件成立并且机械臂停在目标坐标位置之前，或者Sense条件不成立并且机械臂直接停在目标坐标位置上。

- 关于Till

  使用选项Till，可在执行Jump之前对机器人进行减速控制，设置停止条件。可按照包括检查1个I/O输入或1个存储器I/O这样的条件进行设置。此处使用Sw或MemSw函数。可根据事先设置的条件，检查输入为ON或OFF，对机械臂进行减速、停止控制。

  通过使用Stat函数，可确认Till条件成立并且已执行命令，或者Till条件不成立并且机器人停在目标坐标位置上。

注意

- Jump不能用于垂直6轴型机器人(包括N系列)

  请使用Jump3或Jump3CP。

- 省略Arch编号参数时

  如果省略Arch编号参数的设置，执行Jump命令时的默认Arch值则为C7。正如上文所述，Arch值为C7时，变为“门控运动”移动(请参阅上文)。

- Jump与Jump3及Jump3CP的差异

  可在垂直6轴型机器人(包括N系列)中使用Jump3和Jump3CP，但不能使用Jump。在水平多关节型机器人(包括RS系列)上向Z轴方向进行上升/下降动作时，使用Jump可缩短动作时间。也可以在Z轴以外的方向进行Jump3的接近/转移动作。

- Jump与Go的差异

  Jump与Go的最大差异在于：Go时，所有关节动作同步，各关节同时开始动作并同时停止。而Jump时，动作的开始和结束仅限于垂直方向第3关节。进行装置吸附/配置等作业时，建议使用该命令。

- Jump的减速停止

  使用Jump时，机械臂必须在减速的同时，停在目标坐标位置上。

- Jump的适当速度和加速指示

  分别利用Speed和Accel设置Jump动作时的机器人速度和加减速。仅可在要利用Jump、Go等进行点到点的动作时设置Speed和Accel命令，这点敬请注意。比如，要执行类似Move或Arc等进行直线和曲线动作的命令时，请使用SpeedS或AccelS命令。

  另外，Jump时，可分别针对第3关节的上升移动、第4关节的水平移动(包括旋转)以及第3关节的下降等设置速度和加减速。

- Jump的Pass功能

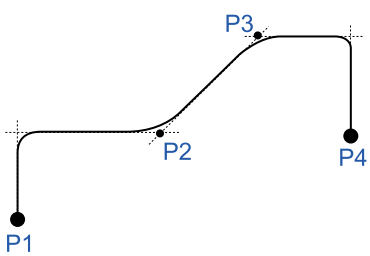
  在下降动作量为0的Jump上附加CP参数时，由于该Jump的水平动作不减速停止，因此，可平滑地连接后续的PTP动作。

  另外，在此前的PTP动作命令上附加CP参数时，由于上升动作量为0的Jump的PTP动作不减速停止，因此，可平滑地连接Jump的水平动作。

  这在希望将通常的Jump水平动作(1个PTP动作)切换为平滑连接几个PTP动作时非常便利。

  [例]

```
Go P1
Jump P2 :Z(-50) C0 LimZ -50 CP
Go P3 :Z(0) CP
Jump P4 C0 LimZ 0
```



使用Arch时的重要事项

由于Arch运动是通过轨迹控制来合成第3关节的上升或下降动作以及横向动作，因此，并不能保证实际的轨迹。其轨迹会因动作速度或机械臂的移动方式而异。请通过作业使用的实际速度和姿势确认实际轨迹。

- 即使在相同位置上执行带有相同[C Arch编号]的Jump命令，低速时的轨迹也会低于高速动作时的轨迹。因此，即使确认没有高速碰撞到障碍物，但低速动作时也可能会发生碰撞，敬请注意。
- 与低速动作时相比，会出现高速动作时垂直上升量增大、垂直下降量减小的倾向。没有达到期待的垂直下降距离时，请降低速度或减速度，或将下降距离设置得长一些。
- 即使是相同距离的动作，轨迹也会因机械臂的移动方式而异。虽然因机械臂的移动方式而导致的轨迹变化多种多样，但是，如果以一般的水平过关节型机器人为例，第1机械臂的移动幅度越大，垂直上升量也越大，而垂直下降量则越小。没有达到期待的垂直下降距离时，请降低速度和减速度，或将下降距离设置得长一些。

易引起的错误

- LimZ值设置过低时

  在第3关节机械臂位置处于比LimZ设置值还高的位置状态下，如果执行Jump，则会发生错误4005。

参阅

Accel、Arc、Arch、Go、JS、JT、LimZ、P#=指定点、Pulse、Sense、Speed、Stat、Till

Jump使用示例

下例所示为从点P0到P1进行单纯的PTP动作后，利用Jump返回到P0。在程序的后半段，机械臂执行Jump，如果输入位4未置为ON，则进行下降动作并移动到P1。输入位4为ON时，不进行下降动作。

```
Function jumptest
    Home
    Go P0
    Go P1
    Sense Sw(4) = On
    Jump P0 LimZ -10
    Jump P1 LimZ -10 Sense   '检查输入4
    If Js(0) = 1 Then
        Print "Input #4 came on during the move and"
        Print "the robot stopped prior to arriving on"
        Print "point P1."
    Else
        Print "The move to P1 completed successfully."
        Print "Input #4 never came on during the move."
    EndIf
Fend


> Jump P10+X50 C0 LimZ-20 Sense !D50;On 0;D80;On 1!
```

从命令窗口执行的操作示例。

```
> Jump P0                              '执行跳跃动作至P0
> Jump P0 C0                           '以通过门型编号C0设置的门型动作向P0执行跳跃动作
> Jump P0 LimZ -10                     '在达到Z限制值-10mm之前向P0进行跳跃动作。
> Jump P0  !D0; On 1; D50; Off 1!      '执行跳跃动作至P0。将动作的移动量变为50%之前输出的第1位设置为On，变为50%之后的第1位设置为Off。
```

输入的第1位变为“On”时，停止Jump命令并进至下一个处理。

```
Function main
    (省略)
  Till Sw(1) = On
  Jump P0 C0 CP Till
```

```
    (省略)
Fend
```

3.14.23 Jump3、Jump3CP

用于以三维门控动作移动机械臂。

Jump3用于组合2个CP动作与1个PTP动作。

Jump3CP则用于组合3个CP动作。

格式

(1) Jump3 转移坐标, 接近开始坐标, 目标坐标 [, C Arch编号] [, CP] [, LJM [,选择姿势标志]] [, Sense | Till | Find] [, !并行处理!] [, SYNC]

(2) Jump3CP 转移坐标, 接近开始坐标, 目标坐标 [, ROT] [, C Arch编号] [, CP] [LJM [选择姿势标志]] [, Sense | Till | Find] [, !并行处理!] [, SYNC]

参数

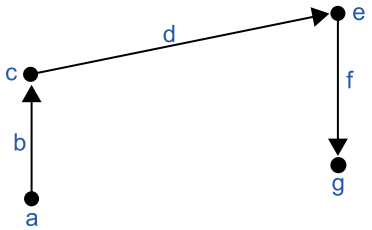
转移坐标
　　指定高于当前位置的转移点。
接近开始坐标
　　指定高于目标坐标的接近起点。
目标坐标
　　指定动作的到达目标坐标点。
ROT
　　以工具姿势变化为优先，确定动作速度、加减速度。可省略。
Arch编号
　　Arch编号用于指定确定Jump3命令Arch型动作的Arch表格。请务必在Arch编号的开头附加大写的“C”。(有效值为C0～C7。)Arch编号可省略。
CP
　　指定路径运动。可省略。
LJM
　　利用LJM函数转换转移坐标、接近坐标、目标坐标。可省略。
选择姿势标志
　　指定赋予LJM函数的姿势标志选择参数。可省略。
Sense | Till | Find
　　记述Sense、Till或Find表达式。可省略。

```
Sense | Till | Find
Sense Sw(表达式) = {On | Off}
Till Sw(表达式) = {On | Off}
Find Sw(表达式) = {On | Off}
```

!并行处理!
　　可在Jump3、Jump3CP命令中添加并行处理语句，在动作期间执行I/O或其它命令。可省略。
SYNC
　　预约动作命令。在通过SyncRobots开始动作之前，机器人不进行动作。

说明

通过三维门控动作将机械臂从当前位置移动到目标坐标位置。三维门控动作由转移动作、跨越动作与接近动作构成。从当前位置到转移坐标的转移动作被称之为CP动作。在Jump3时，从转移坐标到接近开始坐标的跨越动作为PTP动作；Jump3CP时，为CP动作。从接近开始坐标到目标坐标的接近动作为CP动作。

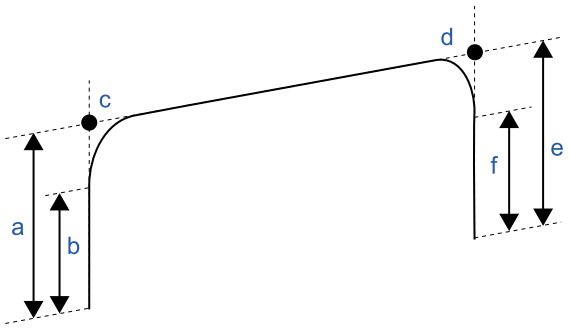


符号	说明
a	当前位置
b	转移动作 CP
c	转移坐标
d	跨越动作 PTP/CP
e	接近开始坐标
f	转移动作 CP
g	目标坐标

通过设置Arch编号进行Arch动作。

下图所示为Jump3和Jump3CP的Arch动作。

请确保转移距离大于Arch上升距离，接近距离大于Arch下降距离。



符号	说明
a	转移距离
b	Arch上升距离
c	转移坐标
d	接近开始坐标
e	接近距离
f	Arch下降距离

Jump3CP的速度和加减速度分别使用SpeedS和AccelS的设置值。有关速度与加减速度之间的关系，请参阅“注意”中的“与CP同时使用Jump3、Jump3CP”。不过，指定ROT修饰参数时的速度和加减速度分别使用SpeedR和AccelR的设置值。此时，SpeedS和AccelS的设置值变为无效状态。

通常的移动距离为0，仅姿势关节进行动作时，会发生错误。通过附加ROT修饰参数并以工具姿势变化的加速度为优先，可不出错误地进行动作。已经附加ROT修饰参数时，如果没有姿势变化，并且移动距离不是0，则会发生错误。

另外，相对于移动距离，工具姿势变化速度过大时，或指定的转速超过机械手限度时，也会发生错误。此时，请降低指定速度，或附加ROT修饰参数，并以姿势变化的加减速度为优先。

注意

- LimZ对Jump3和Jump3CP没有影响。

  由于跨越动作未必仅限于与坐标系的Z轴垂直，因此，LimZ对Jump3和Jump3CP没有影响。

- Jump3的跨越动作为PTP动作。

  由于难以预测PTP动作的轨迹，因此，请充分注意不要干扰机器人主体或外围装置。

- 与CP同时使用Jump3、Jump3CP

  如果使用CP参数，动作命令则会在开始减速的同时将控制移交给下一语句。这在用户连续发出几个动作命令，要以一定的速度进行连续动作时非常便利。为未指定CP的Jump3命令、Jump3CP命令时，机械臂必须减速，以停在指定的目标位置上。

- Jump3的Pass功能

  在接近动作量为0的Jump3上附加CP参数时，由于该Jump3的跨越动作不减速停止，因此，可平滑地连接后续的PTP动作。

  另外，在此前的PTP动作命令上附加CP参数时，由于转移动作量为0的Jump3的PTP动作不减速停止，因此，可平滑地连接Jump3的跨越动作。

  这在希望将通常的Jump3跨越动作(1个PTP动作)切换为平滑连接几个PTP动作时非常便利。

- Jump3CP的Pass功能

  在接近动作量为0的Jump3CP上附加CP参数时，由于该Jump3CP的跨越动作不减速停止，因此，可平滑地连接后续的CP动作。

  另外，在此前的CP动作命令上附加CP参数时，由于转移动作量为0的Jump3CP的CP动作不减速停止，因此，可平滑地连接Jump3CP的跨越动作。

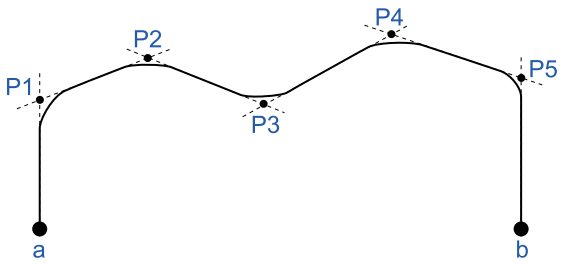
  这在希望将通常的Jump3CP跨越动作(1个CP动作)切换为平滑连接几个CP动作时非常便利。

  (例1)

```
Jump3 P1,P2,P2 CP
Go P3,P4 CP
Jump3 P4,P5,P5+tlz(50)
```

  (例2)

```
Jump3CP P1,P2,P2 CP
Move P3,P4 CP
Jump3CP P4,P5,P5+tlz(50)
```



- 与LJM同时使用Jump3、Jump3CP

  如果使用LJM参数，则可简化使用LJM函数的程序。

  比如，可将

```
P11 = LJM(P1, Here, 2)
P12 = LJM(P2, P11, 2)
P13 = LJM(P3, P12, 2)
Jump3 P11, P12, P13
```

这样的4行程序替换为下述1行程序：

```
Jump3 P1, P2, P3 LJM 2
```

可以转换为1行程序。

LJM参数对于垂直6轴型机器人(包括N系列)与RS系列机器人有效。

Jump3CP不能用于因跨越动作为直线(CP)动作而中途切换手腕姿势。因此，请勿使用可切换手腕姿势的LJM函数的选择姿势标志(LJM 1)。

使用Arch时的重要事项

由于Arch运动是通过轨迹控制所进行的动作合成，因此，不能保证实际的轨迹。其轨迹会因动作速度或机械臂的移动方式而异。请通过作业使用的实际速度和姿势确认实际轨迹。

- 即使在相同位置上执行带有相同[C Arch编号]的Jump3命令，低速时的轨迹也会低于高速动作时的轨迹。因此，即使确认没有高速碰撞到障碍物，但低速动作时也可能会发生碰撞，敬请注意。
- 与低速动作时相比，高速动作时没有合成的转移移动量会增大，而没有合成的接近移动量则会减小。没有达到期待的移动距离时，请降低速度或减速度，或将接近距离设置得长一些。
- 即使是相同距离的动作，轨迹也会因机械臂的移动方式而异。

易引起的错误

- 在转移动作(接近动作)和跨越动作中主要进行动作的关节相同时

  利用Jump3、Jump3CP命令执行Arch运动期间，可能会发生异常加速度错误。这在转移动作(接近动作)和跨越动作中主要进行动作的关节相同时尤其明显。在这种情况下，为Jump3时，请利用Accel命令，降低跨越动作的加减速度进行回避；为Jump3CP时，请通过利用AccelS命令，降低跨越动作的加减速度进行回避。另外，根据动作姿势，有时利用AccelS命令降低转移动作(接近动作)的加减速度也可能有效。

参阅

Accel、Arc、Arch、Go、JS、JT、P#=指定点、Pulse、Sense、Speed、Stat、Till

Jump3使用示例

```
'类似SCARA机器人的Jump那样进行动作的垂直6轴型机器人(包括N系列)的动作
Jump3  Here :Z(100), P3 :Z(100), P3

'使用z工具坐标的转移动作和接近动作
Jump3  Here -TLZ(100), P3 -TLZ(100), P3

'使用z基础坐标的转移动作和使用z工具坐标的接近动作
Jump3  Here +Z(100), P3 -TLZ(100), P3
```

利用Tool1进行转移动作、利用Tool2进行接近动作的示例

```
Arch 0,20,20
Tool 1
Go P1

P2 = P1 -TLZ(100)
```

```
Tool 2
Jump3 P2, P3-TLZ(100), P3 C0
```

3.14.24 JumpTLZ

用于以三维门控动作移动机械臂。
N系列专用命令。
JumpTLZ为2个CP动作与1个PTP动作的组合。

格式

JumpTLZ
目标坐标,TLZ方向移动量[, C Arch编号] [, CP] [, LJM [, 选择姿势标志]] [, Sense | Till | Find] [, !并行处理!] [, SYNC]

参数

目标坐标
　　指定动作的到达目标坐标点。
TLZ方向移动量
　　指定Tool坐标Z方向的移动量。单位为mm。使用当前设置的Tool编号的Tool坐标。
Arch编号
　　Arch编号用于指定确定JumpTLZ命令Arch型动作的Arch表格。请务必在Arch编号的开头附加大写的“C”。(有效值为C0～C7。)Arch编号可省略。
CP
　　指定路径运动。可省略。
LJM
　　利用LJM函数转换目标坐标。可省略。
选择姿势标志
　　指定赋予LJM函数的姿势标志选择参数。可省略。
Sense | Till | Find
　　记述Sense、Till或Find表达式。可省略。

```
Sense | Till | Find
Sense Sw(表达式) = {On | Off}
Till Sw(表达式) = {On | Off}
Find Sw(表达式) = {On | Off}
```

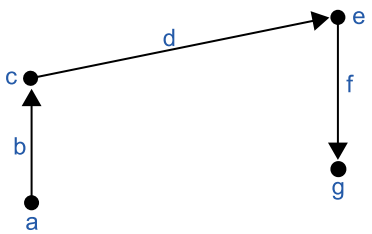
!并行处理!
　　可在Jump3、Jump3CP命令中添加并行处理语句，在动作期间执行I/O或其它命令。可省略。
SYNC
　　预约动作命令。在通过SyncRobots开始动作之前，机器人不进行动作。

说明

通过三维门控动作将机械臂从当前位置移动到目标坐标位置。三维门控动作由转移动作、跨越动作与接近动作构成。从当前位置到转移坐标的转移动作被称之为CP动作。从转移坐标到接近开始坐标的跨越动作为PTP动作。

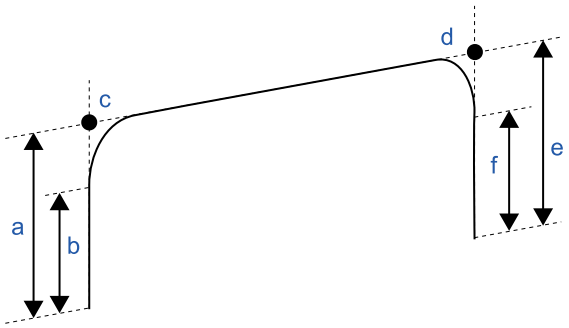
转移坐标是从当前位置向Tool坐标Z方向进行由TLZ方向移动量定义的移动后的位置。转移坐标的机器人姿势与当前位置的姿势相同。(仅在通过特殊点或特殊点附近时，机器人的姿势可能不会相同。)

接近开始坐标是从转移坐标向Tool坐标X、Y方向进行移动(移动距离为到达目标位置的移动量)后的位置。转移坐标的U、V、W坐标以及机器人姿势与目标位置相同。(仅在通过特殊点或特殊点附近时，机器人的姿势可能不相同。)



符号	说明
a	当前位置
b	转移动作 CP
c	转移坐标
d	跨越动作 PTP
e	接近开始坐标
f	转移动作 CP
g	目标坐标

通过设置Arch编号进行Arch动作。请确保转移距离大于Arch上升距离，接近距离大于Arch下降距离。



符号	说明
a	转移距离
b	Arch上升距离
c	转移坐标
D	接近开始坐标
e	接近距离
f	Arch下降距离

注意

- LimZ对JumpTLZ没有影响

  由于跨越动作未必仅限于与坐标系Z轴垂直，因此，LimZ对JumpTLZ没有影响。

- JumpTLZ的跨越动作为PTP动作

  由于难以预测PTP动作的轨迹，因此，请充分注意不要干扰机器人主体或外围装置。

- JumpTLZ与Jump3的差异

  JumpTLZ与Jump3存在下述差异。

  - JumpTLZ：
    - 不能将转移坐标设在从当前位置向Tool坐标Z方向移动的位置以外。
    - 不能将接近坐标从目标坐标移动到Tool坐标的Z方向以外。
    - 另外，无法指定接近距离。

- 不可在转移坐标、接近坐标、目标位置上选择不同的Tool坐标。(不能利用Tool1进行转移动作或利用Tool2进行接近动作)

- Jump3:
  - 可任意指定转移坐标的位置。
  - 可任意指定接近坐标的位置。
  - 可在转移坐标、接近坐标、目标位置上选择不同的Tool坐标。(可利用Tool1进行转移动作或利用Tool2进行接近动作)

- 可使用JumpTLZ的机型

  仅限于N系列可使用。

使用Arch时的重要事项

由于Arch运动是通过轨迹控制所进行的动作合成，因此，不能保证实际的轨迹。其轨迹会因动作速度或机械臂的移动方式而异。请通过作业使用的实际速度和姿势确认实际轨迹。

- 即使在相同位置上执行带有相同[C Arch编号]的JumpTLZ命令，低速时的轨迹也会低于高速动作时的轨迹。因此，即使确认没有高速碰撞障碍物，但低速动作时也可能会发生碰撞，敬请注意。
- 与低速动作时相比，高速动作时没有合成的转移移动量会增大，而没有合成的接近移动量则会减小。没有达到期待的移动距离时，请降低速度和减速度，或将接近距离设置得长一些。
- 即使是相同距离的动作，轨迹也会因机械臂的移动方式而异。

易引起的错误

- 在转移动作(接近动作)和跨越动作中主要进行动作的关节相同时

  利用JumpTLZ命令执行Arch运动期间，可能会发生异常加速度错误。这在转移动作(接近动作)和跨越动作中主要进行动作的关节相同时尤其明显。在这种情况下，请利用Accel命令，降低跨越动作的加减速度予以回避。另外，有时根据动作姿势，利用AccelS命令降低转移动作(接近动作)的加减速度也可能有效。

参阅

Accel、Arc、Arch、Go、JS、JT、P#=指定点、Pulse、Sense、Speed、Stat、Till

JumpTLZ使用示例

从当前位置向Tool坐标Z方向上升100 mm并移动到目的地(P0)时：

```
JumpTLZ  P0, -100
```

3.15 L

3.15.1 LatchEnable

用于将通过R-I/O输入实现机器人位置闩锁的功能设为有效或无效。

格式
LatchEnable { On | Off }

参数

On | Off
On：将机器人位置闩锁功能设为有效。Off：将机器人位置闩锁功能设为无效。

结果
如果省略参数，则显示当前闩锁功能的有效或无效。

说明
利用连接到R-I/O上的触发输入信号将机器人位置闩锁功能设为有效或无效。将闩锁功能设为有效之后，利用最初的触发输入锁定机器人位置。重复锁定机器人位置时，利用LatchEnable Off解除闩锁功能，然后再执行LatchEnable On。重复使用时，如果考虑各命令的处理时间，则需要约60 ms以上的间隔。可无视LatchEnable自身的执行时间。

注意

将闩锁功能设为有效之前，请利用SetLatch设置触发输入端口和触发信号逻辑。

参阅
LatchPos函数、LatchState函数、SetLatch

LatchEnable使用示例

```
Function main
    SetLatch 24, SETLATCH_TRIGGERMODE_LEADINGEDGE
    LatchEnable On    '闩锁功能有效
    Go P1
    Wait LatchState = True     '等待触发
    Print LatchPos     '显示闩锁位置
    LatchEnable Off     '闩锁功能无效
Fend
```

3.15.2 LatchState函数

是用于返回通过R-I/O实现机器人位置闩锁的状态的函数。

格式
LatchState

返回值
如果完成机器人位置的闩锁，则会返回“True”；如果未完成，则会返回“False”。

确认闩锁完成之后，利用LatchPos函数获取闩锁位置信息。

如果在SetLatch中指定了连续闩锁的此时，则当所有指定的闩锁次数都完成时返回“True”。

参阅
LatchEnable、LatchPos函数、SetLatch、Wai

LatchState函数使用示例

```
Function main
    SetLatch 24, SETLATCH_TRIGGERMODE_LEADINGEDGE
    LatchEnable On    '闩锁功能有效
    Go P1
    Wait LatchState = True     '等待触发
    Print LatchPos     '显示闩锁位置
    LatchEnable Off     '闩锁功能无效
Fend
```

3.15.3 LatchPos函数

该函数用于返回利用R-I/O输入信号进行闩锁的机器人位置。

格式

LatchPos ([WithToolArm | WithoutToolArm], 闩锁编号)

参数

WithToolArm | WithoutToolArm
调用函数时，返回基于Tool, Arm设置的位置，或Tool 0, Arm 0的位置。参数可以省略，但指定了闩锁编号时请勿省略。如果省略，则会设置WithToolArm。

- WithToolArm: 0
- WithoutToolArm: 1

WithToolArm
是值为0的常数。返回基于函数调用时的Tool、Arm设置的位置。

WithoutToolArm
是值为1的常数。返回Tool 0、Arm 0的位置，而与Tool、Arm设置无关。

闩锁编号
指定在LatchEnable On以后使用第几个R-I/O输入信号，返回闩锁的点数据。可以指定1, 2, 3, 4。在SetLatch中指定闩锁次数，可以在LatchEnable On以后，在R-I/O输入信号中最多闩锁4次点数据。参数可以省略。如果省略则返回第1个R-I/O输入信号闩锁的点数据。

返回值

以点数据返回利用R-I/O输入信号进行闩锁的机器人位置。

执行LatchPos函数约需15 msec的处理时间。

参数为WithToolArm时，返回基于函数调用时的Tool、Arm设置的位置。

参数为WithoutToolArm时，返回Tool 0、Arm 0的位置，而与Tool、Arm设置无关。

参阅

LatchEnable、LatchState函数、SetLatch

LatchPos函数使用示例

```
Function main
    SetLatch 24, SETLATCH_TRIGGERMODE_LEADINGEDGE, 4
    LatchEnable On    '闩锁功能有效
    Go P1
    Wait LatchState = True     '等待触发
    Print LatchPos(WithoutToolArm, 1)     '显示闩锁位置1
    Print LatchPos(WithoutToolArm, 2)     '显示闩锁位置2
    Print LatchPos(WithoutToolArm, 3)     '显示闩锁位置3
    Print LatchPos(WithoutToolArm, 4)     '显示闩锁位置4
    LatchEnable Off     '闩锁功能无效
Fend
```

省略参数时的使用示例

```
Function main
    SetLatch 24, SETLATCH_TRIGGERMODE_LEADINGEDGE
    LatchEnable On    '闩锁功能有效
    Go P1
    Wait LatchState = True     '等待触发
    Print LatchPos     '显示闩锁位置
    LatchEnable Off     '闩锁功能无效
Fend
```

将LatchPos的返回值代入到点数据中的示例

```
P2 = LatchPos    '代入闩锁位置1省略参数

P2 = LatchPos(WithoutToolArm, 1)    '代入闩锁位置1
P3 = LatchPos(WithoutToolArm, 2)     '代入闩锁位置2
P4 = LatchPos(WithoutToolArm, 3)     '代入闩锁位置3
P5 = LatchPos(WithoutToolArm, 4)     '代入闩锁位置4
```

3.15.4 LCase$函数

用于返回小写字符串。

格式
LCase$ (字符串)

参数

字符串
　　指定小写的字符串。

返回值
返回已转换为小写的字符串。

参阅
LTrim$、Trim$、RTrim$、UCase$

LCase$函数使用示例

```
str$ = "Data"
str$ = LCase$(str$)  ' str$ = "data"
```

3.15.5 Left$函数

用于从字符串的左侧提取指定字符串的函数。

格式
Left$ (字符串, 字符数)

参数

字符串
　　是指从左侧复制指定字符串的源字符串。
字符数
　　直接以数值形式指定从字符串左侧复制的字符数。

返回值
从指定字符串的左侧提取指定的字符数进行返回。

说明
Left$用于从指定字符串的左侧提取字符数所示数量的字符并进行返回。使用Left$可返回指定字符串内的字符数所示数量的字符。

参阅
Asc、Chr$、InStr、Len、Mid$、Right$、Space$、Str$、Val

Left$函数使用示例
如下所示为分析字符串的程序示例。

```
Function ParsePartData(DataIn$ As String, ByRef PartNum$ As String, ByRef PartName$
As String, ByRef PartCount As Integer)

    Integer pos
    String temp$

    pos = Instr(DataIn$, ",")
    PartNum$ = Left$(DataIn$, pos - 1)

    DataIn$ = Right$(datain$, Len(DataIn$) - pos)
    pos = Instr(DataIn$, ",")

    PartName$ = Left$(DataIn$, pos - 1)

    PartCount = Val(Right$(datain$, Len(DataIn$) - pos))

Fend
```

如下所示为通过命令窗口使用Left$命令的示例。

```
> Print Left$("ABCDEFG", 2)
 AB

> Print Left$("ABC", 3)
 ABC
```

3.15.6 Len函数

用于返回字符串的字符数。

格式
Len (字符串)

参数

字符串
　　指定字符串表达式。

返回值
以整数值返回作为Len命令自变量赋予的字符串字符数。

说明
Len用于以整数值(0～255)返回指定字符串的字符数。(字符串的字数限制范围为0～255。)

参阅
Asc、Chr$、InStr、Left$、Mid$、Right$、Space$、Str$、Val

Len函数使用示例
如下所示为分析字符串的程序示例。

```
Function ParsePartData(DataIn$ As String, ByRef PartNum$ As String, ByRef PartName$
As String, ByRef PartCount As Integer)

    Integer pos
    String temp$

    pos = Instr(DataIn$, ",")
    PartNum$ = Left$(DataIn$, pos - 1)

    DataIn$ = Right$(datain$, Len(DataIn$) - pos)
    pos = Instr(DataIn$, ",")

    PartName$ = Left$(DataIn$, pos - 1)

    PartCount = Val(Right$(datain$, Len(DataIn$) - pos))

Fend
```

如下所示为通过命令窗口使用Len命令的示例。

```
> ? len("ABCDEFG")
7

> ? len("ABC")
3

> ? len("")
0
>
```

3.15.7 LimitTorque

设置和显示高功率模式时的转矩上限值。

格式

(1) LimitTorque 所有关节的高功率转矩上限值

(2) LimitTorque 第1关节高功率转矩上限值, 第2关节高功率转矩上限值, 第3关节高功率转 矩上限值, 第4关节高功率转矩上限值

(3) LimitTorque 第1关节高功率转矩上限值, 第2关节高功率转矩上限值, 第3关节高功率转矩上限值, 第4关节高功率转矩上限值, 第5关节高功率转矩上限值, 第6关节高功率转矩上限值

(4) LimitTorque

参数

所有关节的高功率转矩上限值
　　以表示相对于各关节瞬时最大转矩比例的整数值，指定所有关节的高功率转矩上限值。
第n关节的高功率转矩上限值
　　以表示相对于第n关节瞬时最大转矩比例的整数值，指定第n关节的高功率转矩上限值。

结果

如果省略参数，则显示当前的LimitTorque值。

说明

限制高功率模式时的转矩上限值。一般已设置最大转矩，不需要更改本设置值。但是，例如在想要限制转矩，以减少与周围设备的干扰带来的机器人和设备的损伤、不要发生大于特定动作所需转矩的转矩时有效。上限值是通过PTRQ测量进行特定动作时的峰值转矩，并加上考虑到变动量(10%左右)的余量后的值。

本命令无法设置低于低功率时的转矩上限值的高功率转矩上限值。相对于各机型、关节的可以设置的最小值各不相同。请在设置后显示设置值，并实际确认设置的上限值。

在如下的场合，LimitTorque将返回默认值。

- 启动控制器时
- 执行Motor On
- 执行SFree、SLock、Brake
- 执行Reset、Reset Error
- 利用停止按钮或执行Quit All等结束任务

注意

- LimitTorque设置过低

  LimitTorque对特定动作以设置的转矩限制值为上限值并限制转矩，而与为了以设置的加减速度进行动作所需的转矩的大小无关。结果，特定动作需要设置上限值以上的转矩时，机器人可能会进行振动动作，并发出异响、位置偏差错误与超限等，而无法适当地进行动作。请务必在测量PTRQ之后使用转矩限制功能。如果出现上述状态，则表明转矩不足，因此，请进行调整，增大转矩上限值，以确保正常动作。

参阅

LimitTorque函数、Power、PTrq、RealTorque

LimitTorque使用示例

以下所示为将第1关节的最大转矩限制到80%进行动作的示例。

```
Function main
    Motor On
    Power high
    Speed 100;Accel 100,100
    LimitTorque 80,100,100,100    '将第1关节的最大转矩限制到80%
    Jump P1                       '执行Jump动作
Fend
```

3.15.8 LimitTorque函数

用于返回LimitTorque命令的设置值。

格式

LimitTorque (关节编号)

参数

关节编号
以1～9的整数进行指定。附加轴的S轴为8，T轴为9。

返回值

用于以整数值返回LimitTorque命令的设置值。

参阅

LimitTorque

LimitTorque函数使用示例

```
Print LimitTorque(1) '显示第1关节的LimitTorque值
```

3.15.9 LimitTorqueLP

设置和显示低功率模式时的转矩上限值。

格式

(1) LimitTorqueLP 所有关节的低功率转矩上限值

(2) LimitTorqueLP 第1关节的低功率转矩上限值, 第2关节的低功率转矩上限值, 第3关节的低功率转矩上限值, 第4关节的低功率转矩上限值

(3) LimitTorqueLP 第1关节的低功率转矩上限值, 第2关节的低功率转矩上限值, 第3关节的低功率转矩上限值, 第4关节的低功率转矩上限值, 第5关节的低功率转矩上限值, 第6关节的低功率转矩上限值

(4) LimitTorqueLP

参数

所有关节的低功率转矩上限值
　　以表示相对于各关节瞬时最大转矩比例的整数值，指定所有关节的低功率转矩上限值。
第n关节的低功率转矩上限值
　　以表示相对于第n关节瞬时最大转矩的比例的整数值，指定第n关节的低功率转矩上限值。

结果

如果省略参数，则显示当前的LimitTorqueLP值。

未通过本命令变更值时，将显示默认值。

说明

用于限制低功率模式时的转矩上限值。默认设置时，已将针对低功率动作所需的转矩设为上限值，(上限值因机型或轴而异。约为15～60%左右)，通常无需变更本设置值，但是，为了减少因与外围设备碰撞而导致的机器人或装置损坏，可进行限制，以免产生远超出无碰撞的正常动作所需的转矩。上限值是通过PTRQ测量进行正常动作时的峰值转矩，并加上考虑到变动量(推荐为40%)的余量后的值。要将同一值适用于不同的机器人时，请再增加10～20%的余量。

以低功率时的默认最大转矩为1.0来显示PTRQ的值。比如，变更前的默认值为27%、PTRQ的测量值为0.43时，27% × 0.43 × 1.4 = 16.25，数值取整的话为17。

不能在本命令中设置低于5%的值或高于默认值的值。在这种情况下，设置值不大于5时设置取5，超出默认值时取设置值设置。比如，如果设为“LimitTorqueLP 100”，由于默认值必须为100以下，因此恢复为所有轴的默认值设置。请在设置后显示设置值，并实际确认设置的上限值。

重新启动控制器之前，LimitTorqueLP的设置值保持有效。

注意

- 过低的LimitTorqueLP设置

  针对特定的动作，不论以设置的加速度进行动作所需的转矩多大，LimitTorqueLP都会以设置的转矩值为上限值对转矩进行限制。因此，特定动作需要设置上限值以上的转矩时，机器人可能发生位置偏差错误，因而无法适当地进行动作。请务必在低功率状态下测量PTRQ之后使用转矩限制功能。在这种情况下，由于转矩不足，请增大转矩上限值，进行调整以使可以正常动作。

参阅

LimitTorqueLP函数、PTRQ

LimitTorqueLP使用示例

如下所示为将第1关节的最大转矩限制为10%进行动作的示例。

```
Function main
Motor On
Power low
LimitTorqueLP 10,27,31,42     '将第1关节的最大转矩限制为10%
'为其它轴时，设置默认值
Go P1     '执行Go动作
Fend
```

3.15.10 LimitTorqueLP函数

用于返回LimitTorqueLP命令的设置值。

格式

LimitTorqueLP (关节编号)

参数

关节编号
以1～9的整数进行指定。附加轴的S轴为8，T轴为9。

返回值

以整数值返回LimitTorqueLP命令的设置值。

参阅

LimitTorqueLP

LimitTorqueLP函数使用示例

```
Print LimitTorqueLP(1) '显示第1关节的LimitTorqueLP值
```

3.15.11 LimitTorqueStop

用于在高功率模式时达到转矩上限的情况下，启用或退出机器人停止功能。

格式

(1) LimitTorqueStop 状态

(2) LimitTorqueStop 状态, 关节编号

(3) LimitTorqueStop

参数

状态

- On：将转矩上限时的停止功能设为有效。
- Off：将转矩上限时的停止功能设为无效。

关节编号

指定1～6的关节编号。(为SCARA机器人时，关节编号为1～4)

结果

如果省略参数，则显示当前的LimitTorqueStop状态。

说明

将高功率动作的转矩上限时的停止功能设为有效。达到转矩上限(默认值为100%)时立即停止机器人。通过并用基于LimitTorque的转矩限制功能，可减少因高功率模式时的碰撞或接触而导致的机器人/装置损坏。

可对所有轴的打开/关闭以及各轴的打开或关闭进行设置。默认值为所有轴关闭。

控制器启动时，将恢复为默认值，但在其他情况下，除非利用本命令明确进行设置，否则状态不会发生变化。

达到转矩上限时，将输出错误5040“高功率状态下电动机转矩输出异常”信息并停止机器人。

参阅

LimitTorque、LimitTorque函数

LimitTorqueStop使用示例

如下所示为将第1关节的最大转矩限制为30%并立即将其停止的示例。

```
Function main
Motor On
Power high
Speed 20
Accel 20,20
LimitTorque 30,100,100,100     '将第1关节的最大转矩限制到30%
LimitTorqueStop On, 1     '第1关节达到最大转矩时立即停止
Go P1     '执行Go动作
Fend
```

3.15.12 LimitTorqueStop函数

用于返回LimitTorqueStop命令的设置值。

格式

LimitTorqueStop (关节编号)

参数

关节编号
以1～6的整数进行指定。

返回值

以整数值返回LimitTorqueStop命令的设置值。

- 0 = 关闭
- 1 = 打开

参阅

LimitTorqueStop

LimitTorqueStop函数使用示例

```
Print LimitTorqueStop(1) '显示第1关节的LimitTorqueStop值
```

3.15.13 LimitTorqueStopLP

用于在低功率模式时达到转矩上限的情况下，启用或退出机器人停止功能。

格式

(1) LimitTorqueStopLP 状态

(2) LimitTorqueStopLP 状态, 关节编号

(3) LimitTorqueStopLP

参数

状态

- On：将转矩上限时的停止功能设为有效。
- Off：将转矩上限时的停止功能设为无效。

关节编号

指定1～6的关节编号。(为SCARA机器人时，关节编号为1～4)

结果

省略参数时，将显示当前的LimitTorqueStopLP状态。

说明

将低功率动作的转矩上限时的停止功能设为有效。达到转矩上限时立即停止机器人。通过并用基于LimitTorqueLP的转矩限制功能，可减少因低功率模式时的碰撞或接触而导致的机器人/装置损坏。

可对所有轴的打开/关闭以及各轴的打开或关闭进行设置。默认值为所有轴关闭。

控制器启动时，将恢复为默认值，但在其他情况下，除非利用本命令明确进行设置，否则状态不会发生变化。

达到转矩上限时，将输出错误5041“低功率状态下电动机转矩输出异常”信息并停止机器人。

参阅

LimitTorqueLP、LimitTorqueLP函数

LimitTorqueStopLP使用示例

如下所示为将第3关节的最大转矩限制为15%并立即将其停止的示例。

```
Function main
Motor On
Power low
LimitTorqueLP 20,27,15,42     '将第3关节的最大转矩限制为15% '为其它轴时，设置默认值
LimitTorqueStopLP On, 3     '第3关节达到最大转矩时立即停止
Go P1     '执行Go动作
Fend
```

3.15.14 LimitTorqueStopLP函数

用于返回LimitTorqueStopLP命令的设置值。

格式

LimitTorqueStopLP (关节编号)

参数

关节编号
以1～6的整数进行指定。

返回值

以整数值返回LimitTorqueStopLP命令的设置值。

- 0 = 关闭
- 1 = 打开

参阅

LimitTorqueStopLP

LimitTorqueStopLP函数使用示例

```
Print LimitTorqueStopLP(3) '显示第3关节的LimitTorqueStopLP值
```

3.15.15 LimZ

用于设置Jump命令时第3关节高度(Z坐标值)初始值。

格式

(1) LimZ Z坐标值

(2) LimZ

参数

Z坐标值
　　指定第3关节动作范围内的坐标值。

结果

如果省略参数，则显示当前的LimZ值。

说明

执行Jump命令时，机器人机械臂在第3关节(Z轴)方向上升，然后在X-Y平面移动，最后在第3关节(Z轴)方向下降，而LimZ则用于设置此时机械臂在第3关节(Z轴)方向上进行动作的高度上限。LimZ用于设置执行Jump命令时第3关节动作范围的最高坐标默认值。如果在执行Jump命令时未设置特定的LimZ值，则使用最后设置的LimZ值。

注意

- 将LimZ值重置为0

  重新启动控制器或执行SFree、SLock、Motor On等命令均可将LimZ值初始化为0。

- LimZ值不能用于Arm、Tool或Local坐标

  LimZ的高度限制值为机器人坐标的Z坐标值。并不是Arm、Tool或Local坐标的Z坐标值。因此，使用高度不同的夹具末端(卡爪工具)时，敬请注意。

- LimZ对Jump3和Jump3CP没有影响

  由于跨越动作未必仅限于与坐标系的Z轴垂直，因此，LimZ对Jump3和Jump3CP没有影响。

参阅

Jump

LimZ使用示例

如下所示为Jump操作时使用LimZ的示例。

```
Function main
    LimZ -10              '设置LimZ的默认值
    Jump P1               '执行Jump时水平移动-10
    Jump P2 LimZ -20      '执行Jump时水平移动-20
    Jump P3               '执行Jump时水平移动-10
Fend
```

3.15.16 LimZ函数

用于返回LimZ命令的设置值。

格式

LimZ

返回值

以实值返回LimZ命令的设置值。

参阅

LimZ

LimZ函数使用示例

```
Real savLimz

savLimz = LimZ
LimZ -25
Go pick
LimZ savLimZ
```

3.15.17 LimZMargin

是用于设置以高于LimZ设置值的位置开始动作时的错误检测界限以及返回设置值的函数。

格式

(1) LimZMargin LimZ界限

(2) LimZMargin

参数

LimZ界限
　　指定检测LimZ错误的界限值。

结果

如果省略参数，则显示当前的LimZMargin值。

说明

执行Jump命令时，第3关节(Z轴)将上升至LimZ设置的高度，但在开始Jump动作时，如果第3关节(Z轴)处于比LimZ位置高的位置，则会发生错误。LimZMargin对该错误检测设置界限值。默认设置为0.02 mm。

注意

- 将LimZMargin值重置为默认值

  重新启动控制器或执行SFree、SLock、Motor On等命令均可将LimZMargin值初始化为默认值。

参阅

LimZMargin函数、LimZ

LimZMargin使用示例

如下所示为Jump操作时使用LimZMargin的示例。

```
Function main
    LimZ -10              '设置LimZ的默认值
    LimZMargin 0.03       'LimZ的错误检测界限设为0.03 mm
    Jump P1               '执行Jump时水平移动-10
    Jump P2 LimZ -20      '执行Jump时水平移动-20
    Jump P3               '执行Jump时水平移动-10
Fend
```

3.15.18 LimZMargin函数

用于返回LimZMargin命令的设置值。

格式

LimZMargin

返回值

以实值返回LimZMargin命令的设置值。

参阅

LimZMargin、LimZ

LimZMargin函数使用示例

```
Real savLimzMargin

savLimzZMargin = LimZMargin
LimZMargin 0.03
Jump pick
LimZ savLimZMargin
```

3.15.19 Line Input

用于读入1行输入数据并将该数据代入到字符串变量中。

格式

Line Input 字符串变量名$

参数

字符串变量名$
　　指定字符串变量名。(请在字符串变量名最后附加 $。)

说明

Line Input用于从显示装置读入1行输入数据并代入到Line Input命令的字符串变量中。如果处于Line Input命令用于从用户侧接收数据的状态，显示装置中则会显示提示符“?”令。在该提示符之后输入的数据行作为字符串的值被代入。将输入数据进行输入之后，请按下[ENTER]键。

参阅

Input、Input #、Line Input#、ParseStr

Line Input使用示例

如下所示为Line Input的使用示例。

```
Function Main
    String A$
    Line Input A$  '读入1行输入数据并代入到A$中
    Print A$
Fend
```

如果执行上述程序，则进行下述对话。

```
?A, B, C
A, B, C
```

3.15.20 Line Input

用于从文件、通信端口、数据库或装置读入1行数据。

格式

Line Input #端口编号, 字符串变量名$

参数

端口编号

是表示文件、通信端口、数据库、装置的ID编号。文件编号是由ROpen、WOpen、AOpen等语句指定的编号。通信端口编号是由OpenCom(RS-232C)或OpenNet(TCP/IP)语句指定的编号。数据库编号是由OpenDB语句指定的编号。

装置ID为以下数值。

- 21 RC+
- 24 TP(仅TP1)
- 20 TP3

字符串变量名$

指定字符串变量名。(请在字符串变量名最后附加 $。)

说明

Line Input # 用于从由端口编号指定的装置读入读入1行数据，并代入到由字符串变量名$指定的变量中。

注意

- 支持的控制器型号

  若在T/VT系列中指定RS-232C端口时会发生错误。

参阅

Input、Input #、Line Input

Line Input #使用示例

下例所示为从通信端口1接收字符串数据并代入到字符串变量A$中。

```
Function lintest
    String a$
    Print #1, "Please input string to be sent to robot"
    Line Input #1, a$
    Print "Value entered = ", a$
Fend
```

3.15.21 LJM函数

用于返回为确保参照点相对于指定点的关节移动量最小而转换姿势标志的点数据。

格式

LJM (指定点 [, 指定参照点 [, 选择姿势标志]])

参数

点指定

指定对象点数据。

指定参照点

指定作为基准的点数据。省略参照点指定时，以当前位置(Here)为参照点。

选择姿势标志

垂直6轴型

- 1：通过手腕姿势(Wrist标志)、J4Flag、J6Flag与J1Flag进行转换，以使J4轴的移动量为最短。是省略“选择姿势标志”时的默认设置。
- 2：通过J4Flag与J6Flag进行转换。
- 3：通过手腕姿势(Wrist标志)、J4Flag、J6Flag与J1Flag进行转换，以使J5轴的移动量最短。
- 4：通过手腕姿势(Wrist标志)、J4Flag、J6Flag与J1Flag进行转换，以使J6轴的移动量最短。

“选择姿势标志”	腕部姿势	肘姿势	手腕姿势	J1Flag	J4Flag	J6Flag	移动量最短轴的优先顺序
1	-	-	○	○	○	○	J4
2	-	-	-	○	○	○	-
3	-	-	○	○	○	○	J5
4	-	-	○	○	○	○	J6

Note：“-”表示与“指定参照点”中指定的姿势相同的姿势。

RS系列

- 1：通过手腕姿势(Hand标志)、J1Flag与J2Flag进行转换。是省略“选择姿势标志”时的默认设置。
- 2：通过手腕姿势(Hand标志)、J1Flag与J2Flag进行转换。用于防止在转换“选择姿势标志”时，发生U轴超出动作范围的错误。

N2系列

- 1：按照J1、J5轴的优先顺序转换为关节移动量减小的姿势。作为转换对象的姿势包括腕部姿势(Hand标志)、肘姿势(Elbow标志)、手腕姿势(Wrist标志)、J4Flag与J6Flag。肘姿势(Elbow标志)必须为上肘姿势。是省略“选择姿势标志”时的默认设置。
- 2：按照J1、J4轴的优先顺序转换为关节移动量减小的姿势。作为转换对象的姿势包括腕部姿势(Hand标志)、肘姿势(Elbow标志)、手腕姿势(Wrist标志)、J4Flag与J6Flag。肘姿势(Elbow标志)必须为上肘姿势。
- 3：通过手腕姿势(Wrist标志)、J4Flag与J6Flag进行转换，以使J4轴的移动量为最短。
- 4：通过J4Flag与J6Flag进行转换。
- 5：变更为与“指定参照点”中指定的姿势不同的腕部姿势(Hand标志)，并通过手腕姿势(Wrist标志)、J4Flag与J6Flag进行转换，使J5轴的移动量为最短。作为转换对象的姿势包括腕部姿势(Hand标志)、肘姿势(Elbow标志)、手腕姿势(Wrist标志)、J4Flag与J6Flag。另外，肘姿势(Elbow标志)必须为上肘姿势。
- 6：变更为与“指定参照点”中指定的姿势不同的腕部姿势(Hand标志)，并通过手腕姿势(Wrist标志)、J4Flag与J6Flag进行转换，使J4轴的移动量为最短。作为转换对象的姿势包括腕部姿势(Hand标志)、肘姿势(Elbow标志)、手腕姿势(Wrist标志)、J4Flag与J6Flag。另外，肘姿势(Elbow标志)必须为上肘姿势。

- 7：将肘姿势(Elbow标志)变更为下肘姿势，并通过手腕姿势(Wrist标志)、J4Flag与J6Flag进行转换，以按J1、J5轴的优先顺序适用最短移动量。作为转换对象的姿势包括腕部姿势(Hand标志)、肘姿势(Elbow标志)、手腕姿势(Wrist标志)、J4Flag与J6Flag。
- 8：将肘姿势(Elbow标志)变更为下肘姿势，并通过手腕姿势(Wrist标志)、J4Flag与J6Flag进行转换，以按J1、J4轴的优先顺序适用最短移动量。作为转换对象的姿势包括腕部姿势(Hand标志)、肘姿势(Elbow标志)、手腕姿势(Wrist标志)、J4Flag与J6Flag。

“选择姿势标志”	腕部姿势	肘姿势	手腕姿势	J4Flag	J6Flag	移动量最短轴的优先顺序
1	○	*1	○	○	○	J1>J5
2	○	*1	○	○	○	J1>J4
3	-	-	○	○	○	J4
4	-	-	-	○	○	-
5	*2	*1	○	○	○	J5
6	*2	*1	○	○	○	J4
7	○	*3	○	○	○	J1>J5
8	○	*3	○	○	○	J1>J4

Note:“-”表示与“指定参照点”中指定的姿势相同的姿势。

- *1：上肘姿势
- *2：腕部姿势与“指定参照点”中指定的姿势不同。
- *下肘姿势

N6系列

- 1：通过手腕姿势(Wrist标志)、J4Flag与J6Flag进行转换，以使J4轴的移动量为最短。是省略“选择姿势标志”时的默认设置。
- 2：通过J4Flag与J6Flag进行转换。
- 3：通过手腕姿势(Wrist标志)、J4Flag与J6Flag进行转换，以使J5轴的移动量为最短。
- 4：通过手腕姿势(Wrist标志)、J4Flag与J6Flag进行转换，以使J6轴的移动量为最短。

“选择姿势标志”	腕部姿势	肘姿势	手腕姿势	J1Flag	J4Flag	J6Flag	移动量最短轴的优先顺序
1	-	-	○	○	○	○	J4
2	-	-	-	○	○	○	-
3	-	-	○	○	○	○	J5
4	-	-	○	○	○	○	J6

Note:“-”表示与“指定点”中指定的姿势相同的姿势。

说明

对于垂直6轴型机器人和N系列来说，要动作到托盘或通过相对偏移等的点运算获得的点时，手腕部分可能会转向意想不到的方向。这是因为上述点运算包含不取决于机器人机型的命令，直接进行动作而未转换必要的姿势标志的缘故。

为了防止手腕部分进行这种意想不到的旋转，LJM函数用于适当地转换点数据的姿势标志。

另外，为N系列时，通过变更腕部姿势标志或肘姿势标志，也可以缩短节拍时间，以及省略垂直6轴型机器人所需的回避点示教。

同样，就RS系列而言，要动作到托盘或通过相对偏移等的点运算获得的点时，第1机械臂可能会转向意想不到的方向。为了防止第1机械臂进行这种意想不到的旋转，LJM函数用于适当地转换点数据的姿势标志。

另外，在RS系列中，如果U轴在转换了姿势标志却要进行超出动作范围的动作，则可能会发生错误。为了防止U轴发生这种超出动作范围的错误，LJM函数用于将U轴的目标角度修正为动作范围内的目标角度。可通过在选择姿势标志中指定2的方式加以运用。

为垂直6轴型、N系列、RS系列以外的机器人时，直接返回指定点。

注意

- 参照点的省略和并行处理

  不能将参照点的省略与并行处理同时放在一个动作命令内。

  ```
  Go LJM(P10)  !D10; MemOn 1 !
  ```

  无法实现上述使用方法。

  ```
  P999 = Here
  Go LJM(P10,P999) !D10; MemOn 1 !
  ```

  请变更为上述程序。

- 关于N2系列的选择姿势标志

  - 选择姿势标志1、2：

    要缩短机器人的节拍时间时，请选择姿势标志1或2。

    由于采取第1关节移动量最小的姿势，因此，几乎可按最短的节拍时间进行所有动作。要在减小第5关节移动量的状态下进行动作时，请选择姿势标志1；要在减小第4关节移动量的状态下进行动作时，请选择姿势标志2。

  - 选择姿势标志3、4：

    要以与垂直6轴型相同的方式使用时，请进行选择。

    姿势标志3与垂直6轴型的姿势标志1相同。

    姿势标志4与垂直6轴型的姿势标志2相同。

  - 选择姿势标志5、6：

    在机器人动作期间，夹具末端接触到机器人周围的墙壁等情况下，请选择姿势标志5或6。由于夹具末端会通过机器人的原点附近位置，因此，机器人进行动作时不易接触到周围的障碍物。要在减小第5关节移动量的状态下进行动作时，请选择姿势标志5；要在减小第4关节移动量的状态下进行动作时，请选择姿势标志6。

  - 选择姿势标志7、8：

    如要适用下肘姿势，请选择姿势标志7或8。执行部分动作时，机器人可能会像姿势选择标志5或6那样进行通过原点附近的动作，因此，即使其周边有障碍物，也可进行动作而不易接触到障碍物。要在减小第5关节移动量的状态下进行动作时，请选择姿势标志7；要在减小第4关节移动量的状态下进行动作时，请选择姿势标志8。

- 本地编号

  利用LJM函数返回的点的本地编号是与“指定点”相同的本地编号。

参阅

Pallet

LJM函数使用示例

```
Function main
  Integer i, j

  P0 = XY(300, 300, 300, 90, 0, 180)
  P1 = XY(200, 280, 150, 90, 0, 180)
  P2 = XY(200, 330, 150, 90, 0, 180)
  P3 = XY(-200, 280, 150, 90, 0, 180)

  Pallet 1, P1, P2, P3, 10, 10

  Motor On
  Power High
  Speed 50; Accel 50, 50
  SpeedS 1000; AccelS 5000

  Go P0
  P11 = P0 -TLZ(50)

  For i = 1 To 10
      For j = 1 To 10
        '点的指定
        P10 = P11    '转移点
      P12 = Pallet(1, i, j)    '目标点
      P11 = P12 -TLZ(50)    '接近起点
      '各点的LJM转换
      P10 = LJM(P10)
      P11 = LJM(P11, P10)
      P12 = LJM(P12, P11)
      '执行动作
        Jump3 P10, P11, P12 C0
    Next
  Next
Fend

Function main2
  P0 = XY(300, 300, 300, 90, 0, 180)
  P1 = XY(400, 0, 150, 90, 0, 180)
  P2 = XY(400, 500, 150, 90, 0, 180)
  P3 = XY(-400, 0, 150, 90, 0, 180)
  Pallet 1, P1, P2, P3, 10, 10

  Motor On
  Power High
  Speed 50; Accel 50, 50
  SpeedS 1000; AccelS 5000

  Go P0

  Do
    '点的指定
    P10 = Here -TLZ(50)    '转移点
    P12 = Pallet(1, Int(Rnd(9)) + 1, Int(Rnd(9)) + 1)    '目标点
    P11 = P12 -TLZ(50)    '接近起点

    If TargetOK(P11) And TargetOK(P12) Then    '点的检查
    '各点的LJM转换
    P10 = LJM(P10)
    P11 = LJM(P11, P10)
    P12 = LJM(P12, P11)
    '执行动作
      Jump3 P10, P11, P12 C0
    EndIf
   Loop
Fend
```

3.15.22 LoadPoints

用于将点文件读入到机器人点存储区域中。

格式

LoadPoints 文件名 [, Merge]

参数

文件名
　　指定要读入到机器人点存储区域中的文件的字符串扩展名固定为“.pts”点。省略扩展名时，添加“.pts”名。指定的文件仅限于项目内的文件。不能指定路径。另外，也不受ChDisk等的影响。详情请参阅ChDisk。

Merge
　　用于读入新的点之前，不想清除当前的点时进行设置。如果进行设置，则将新的点添加到设置的点中。文件中已存在要添加的点时，执行覆写。可省略。

说明

LoadPoints用于将点文件读入到控制器的主存储器中。

合并点文件时，设置Merge。比如，假设1个主点文件在0～100的范围内包括通用点。要在不清除这些点的状态下，读入针对目前正在动作的各部件的新点文件时，设置Merge。在这种情况下，范围变为101～999。

常见错误

- 不能指定路径

  文件名包括路径时，会发生错误。

- 找不到指定文件时(没有文件)

  未找到文件时，会发生错误。

- 其它机器人的点文件

  在文件名中指定其它机器人的点文件时，会发生错误。在这种情况下，利用项目编辑器追加点文件，或执行SavePoints、ImportPoints。

参阅

ImportPoints、Robot、SavePoints

LoadPoints使用示例

```
Function main

  '将通用的点读入到当前的机器人中
  LoadPoints "R1Common.pts"

  '合并部件模型1的点
  LoadPoints "R1Model1.pts", Merge

  机器人2
  ' 读入机器人2的点文件
  LoadPoints "R2Model1.pts"

Fend
```

3.15.23 Local

用于定义和显示本地坐标系。

格式

(1) Local 本地坐标系编号, (点编号1 : 点编号2), (点编号3 : 点编号4) [, { L | R }] [, BaseU]

(2) Local 本地坐标系编号, 坐标系数据

(3) Local 本地坐标系编号, 原点, [X轴指定], [{ X | Y }]

(4) Local 本地坐标系编号

参数

本地坐标系编号
　　指定本地坐标系编号。可利用1～15的整数定义最多15个本地坐标系。
点编号1, 点编号3
　　以点变量指定本地坐标系的点数据。
点编号2, 点编号4
　　以点变量指定基础坐标系的点数据。
L | R
　　将本地原点对准左(1号)或右(2号)。可省略。
BaseU
　　如果指定，U轴坐标则使用基础坐标系。可省略，如果省略，U轴坐标则使用本地坐标系。
坐标系数据
　　直接以点数据指定本地坐标系的原点和方向。水平多关节机器人(包括RS)，请将V坐标和W坐标设置为“0”。
原点
　　以P#(整数)或P(表达式)指定定义本地坐标系原点的机器人坐标系上的位置。
X轴指定
　　以P#(整数)或P(表达式)指定定义本地坐标系X轴上的点的机器人坐标系上的位置。可省略。
Y轴指定
　　以P#(整数)或P(表达式)指定定义本地坐标系Y轴上的点的机器人坐标系上的位置。可省略。
X
　　将连接原点与X轴指定的直线定义为本地坐标系的X轴。由于根据X轴和由3个点确定的平面来计算本地坐标系，因此，Y轴指定未必是Y轴上的点。可省略。(默认设置)
Y
　　将连接原点与Y轴指定的直线定义为本地坐标系的Y轴。由于根据Y轴和由3个点确定的平面来计算本地坐标系，因此，X轴指定未必是X轴上的点。可省略。

说明

(1) Local用于指定基础坐标系的2点位置数据、与点编号2及点编号4一致的本地坐标系的2点、点编号1和点编号3，定义本地坐标系。

例:

```
Local 1, (P1:P11), (P2:P12)
```

P1和P2为本地坐标系的点。P11和P12为基础坐标系的点。

如果本地坐标系上指定的2点间距与基础坐标系的2点间距不同，则在本地坐标系的2点的中点与基础坐标系的2点的中点一致的位置上定义本地坐标系。

同样，使用2个坐标系的中点来定义本地坐标系的Z轴。

(2) 以原点和相对于基础坐标系的角度定义本地坐标系。

例：

```
Local 1, XY(x, y, z, u)
Local 1, XY(x, y, z, u, v, w)
Local 1, P1
```

(3) 指定原点、X轴上的点、Y轴上的点，定义三维本地坐标系。仅使用各点中的X、Y、Z坐标，无视U、V、W坐标。如果指定参数X，X轴指定则位于本地坐标系的X轴上，并且仅使用Y轴指定的Z坐标。如果指定参数Y，Y轴指定则位于本地坐标系的Y轴上，并且仅使用X轴指定的Z坐标。

例：

```
Local 1, P1, P2, P3
Local 1, P1, P2, P3, X
Local 1, P1, P2, P3, Y
```

(4) 显示指定的本地设置。

- L/R参数的使用：如前所述，主要是利用中点定义本地坐标系，而且，可使用选项L或R指定坐标系的左右。
  - 左手本地：左手本地用于在本地坐标系的点、点编号1与基础坐标系的点、点编号2一致的位置上定义本地坐标。(也包括Z轴方向。)
  - 右手本地：右手本地用于在本地坐标系的点、点编号3与基础坐标系的点、点编号4一致的位置上定义本地坐标。(也包括Z轴方向。)
- BaseU参数的使用：如果省略BaseU参数，则自动对本地坐标系的U轴进行补偿，以适合设置的4点的XY坐标值。这样的话，基础坐标系的2点最初也可能包括U轴坐标值。

  比起自动补偿，我们还是建议通过示教来补偿旋转轴等，根据基础坐标系2点的U轴值，对本地坐标系的U轴进行补偿。此时，需要设置BaseU参数。

  机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

注意

- 使用水平多关节机器人时，请不要设置V和W。

  使用水平多关节机器人时，请不在基础坐标系中设置V坐标和W坐标的值，或设置为“0”。否则可能会由于第4关节超出范围而报错。

参阅

ArmSet、Base、ECPSet、LocalClr、TLSet、Where

Local使用示例

如下所示为利用命令窗口的操作示例。

利用左手本地定义本地坐标系原点的示例：

```
> p1 = 0, 0, 0, 0/1
> p2 = 100, 0, 0, 0/1
> p11 = 150, 150, 0, 0
> p12 = 300, 150, 0, 0
> local 1, (P1:P11), (P2:P12), L

> p21 = 50, 0, 0, 0/1
> go p21
```

将原点定义为本地坐标系原点的示例：

```
> local 1, 100, 200, -20, 0
```

将X轴旋转45度的原点定义为本地坐标系原点的示例：

```
> local 2, 50, 200, 0, 0, 45, 0
```

将P2对准本地坐标系X轴时的位置定义为三维本地坐标系原点的示例：

```
> local 3, p1, p2, p3, x
```

将P3对准本地坐标系Y轴时的位置定义为三维本地坐标系原点的示例：

```
> local 4, p1, p2, p3, y
```

3.15.24 Local函数

用作返回已设置本地坐标系数据的函数。

格式

Local (本地坐标系编号)

参数

本地坐标系编号
以表达式或数值指定本地坐标系编号(1～15的整数)。

返回值

将已设置的本地坐标系数据作为点数据进行返回。

参阅

Local

Local函数使用示例

```
P1 = Local(1)
```

3.15.25 LocalClr

用于清除(未定义)本地坐标系。

格式

LocalClr 本地坐标系编号

参数

本地坐标系编号
以表达式或数值指定要清除(未定义)设置的本地坐标系编号(1～15的整数)。

说明

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

参阅

Arm、ArmSet、ECPSet、Local、Tool、TLClr、TLSet

LocalClr使用示例

```
LocalClr 1
```

3.15.26 LocalDef函数

用于返回本地坐标系的设置状态。

格式

LocalDef (本地坐标系编号)

参数

本地坐标系编号
　　指定返回状态的本地坐标系编号(1～15的整数)。

返回值

如果已设置指定的本地坐标系，则返回“True”；如果未设置，则返回“False”。

参阅

Arm、ArmClr、ArmSet、ECPSet、Local、LocalClr、Tool、TLClr、TLSet

LocalDef函数使用示例

```
Function DisplayLocalDef(localNum As Integer)

   If LocalDef(localNum) = False Then
      Print "Local ", localNum, "is not defined"
   Else
      Print "Local 1: ",
      Print Local(localNum)
   EndIf
Fend
```

3.15.27 Lof函数

用于返回指定RS-232C端口或 TCP/IP端口缓冲器的接收数据行数。

格式

Lof (通信端口编号)

参数

通信端口编号
　　指定由OpenCom(RS-232C)或OpenNet(TCP/IP)语句指定的编号。

返回值

返回接收缓冲器的数据行数。没有接收数据时，返回“0”。

说明

Lof函数用于确认有无指定的通信端口接收数据。利用Input#命令提取接收数据。可利用Wait命令使Lof函数的返回值处于待机状态。

注意

- 支持的控制器型号

  若在T/VT系列中指定RS-232C端口时会发生错误。

  使用PC COM端口(1001～1008)时，不能在Wait命令中使用Lof函数。

参阅

ChkCom、ChkNet、Input#、Wait

Lof函数使用示例

下例所示为通过命令窗口执行的情况。显示通信端口1的接收数据行数。

```
>print lof(1)
 5
>
```

3.15.28 LogIn

用于以其它用户身份登录到Epson RC+。

格式

LogIn 日志ID, 密码

参数

日志ID
　　用户登录ID的字符串表达式
密码
　　用户密码的字符串表达式

说明

可通过应用程序操作控制Epson RC+的安全等级。比如，可显示用于其它用户登录到系统的菜单。用户拥有独自的安全权限。有关安全的详细说明，请参阅以下手册。
《Epson RC+ 用户指南》

在开发环境下执行程序时，程序停止之后，程序返回到开始前的用户侧。

要在Auto模式下执行操作窗口时，除非Auto LogIn有效，否则应用程序将以Guest用户的身份登录。在这种情况下，如果在Epson RC+系统中已进行了设置，应用程序则以当前窗口用户的身份进行登录。

注意

仅可在安全选项有效时使用该命令。

参阅

GetCurrentUser$函数

LogIn使用示例

```
Integer errCode
errCode = LogIn("operator", "oprpass")
```

3.15.29 Long

用于声明Long型变量。(4字节整数型变量)

格式

Long 变量名 [(数组变量的最大下标)] [, 变量名 [(数组变量的最大下标)]...]

参数

变量名

指定声明为Long型的变量名。数组变量的最大下标 可利用数组变量的最大下标声明到最高三维。使用下述格式。可省略。
(最大下标1, [最大下标2], [最大下标3])
由于下标从0开始，因此元素数为最大下标加上1。在所有元素数不超过以下最大值的范围内指定各最大下标。

- 本地变量：2,000
- 备份变量(Global Preserve)：4,000
- 全局变量和模块变量：100,000

说明

Long用于声明整数型变量。Long型变量的范围为-2147483648～2147483647。在Function开头声明本地变量。在Function之外声明全局变量和模块变量。

参阅

Boolean, Byte, Double, Global, Int32, Int64, Integer, Real, Short, String, UByte, UInt32, UInt64, UShort

Long使用示例

如下所示为使用Long声明Long型变量的程序。

```
Function longtest

    Long A(10)          'Long型一维数组
    Long B(10, 10)      'Long型二维数组
    Long C(5, 5, 5)     'Long型三维数组
    Long var1, arrayVar(10)
    Long i
    Print "Please enter a Long Number"
    Input var1
    Print "The Integer variable var1 = ", var1
    For i = 1 To 5
        Print "Please enter a Long Number"
        Input arrayVar(i)
        Print "Value Entered was ", arrayVar(i)
    Next I
Fend
```

3.15.30 LSet$函数

用于返回在指定字符串的最后添加空格以形成指定长度的字符串。

格式

LSet$ (字符串, 字符串的长度)

参数

字符串
　　指定字符串表达式。
字符串的长度
　　以整数或表达式指定返回字符串的长度。

返回值

用于返回在指定字符串的最后添加空格的字符串。

参阅

RSet$、Space$

LSet$函数使用示例

```
temp$ = "123"
temp$ = LSet$(temp$, 10)   ' temp$ = "123       "
```

3.15.31 LShift函数

用于将数值数据左移指定的位数。

格式
LShift (数值, 移位数)

参数

数值
　　指定要移位的整数值。
移位数
　　指定进行左移位的位数(0～31的整数值)。

返回值
返回将指定数值左移指定位数的结果。

说明
将指定数值向左(高位侧)移动指定位数。通常，移位部分的低位被设为0。

Lshift与数值 * 2移位数(将数值乘以2移位数的次数)相同。

注意

- 数值数据类型

  包括有多种数值类型。Lshift可以使用Byte型、Double型、Int32型、Integer型、Long型、Real型、Short型、UByte型、UInt32型、UShort型的数值。

参阅
And、LShift64、Not、Or、RShift、RShift64、Xor

LShift函数使用示例

```
Function lshiftst
    Integer i
    Integer num, snum
    num = 1
    For i = 1 to 10
         Print "i =", i
         snum = LShift(num, i)
         Print "The shifted num is ", snum
    Next i
Fend
```

如下所示为利用命令窗口返回Lshift函数结果的其它示例。

```
> Print LShift(2,2)
8
> Print LShift(5,1)
10
> Print LShift(3,2)
12
>
```

3.15.32 LShift64函数

用于将数值数据左移指定的位数。

格式

LShift64 (数值, 移位数)

参数

数值
　　指定要移位的整数值。
移位数
　　指定进行左移位的位数(0～63的整数值)。

返回值

返回将指定数值左移指定位数的结果。

说明

将指定数值向左(高位侧)移动指定位数。通常，移位部分的低位被设为0。

Lshift64与数值 * 2移位数(将数值乘以2移位数的次数)相同。

注意

- 数值数据类型

  包括有多种数值类型。Lshift64可以使用Int64型、UInt64型的数值。

参阅

And, LShift, Not, Or, RShift, RShift64, Xor

LShift64函数使用示例

```
Function lshiftst
    Int64 i
    Int64 num, snum
    num = 1
    For i = 1 to 10
         Print "i =", i
         snum = LShift64(num, i)
         Print "The shifted num is ", snum
    Next i
Fend
```

如下所示为利用命令窗口返回Lshift64函数结果的其它示例。

```
> Print LShift64(2,2)
8
> Print LShift64(5,1)
10
> Print LShift64(3,2)
12
>
```

3.15.33 LTrim$函数

用于删除字符串左侧空格并进行返回。

格式

LTrim$ (字符串)

参数

字符串
　　指定字符串表达式。

返回值

用于返回删除了左侧空格的字符串。

参阅

RTrim$、Trim$

LTrim$函数使用示例

```
str$ = "  data  "
str$ = LTrim$(str$)  ' str$ = "data  "
```

3.16 M

3.16.1 Mask运算符

用于以位为单位屏蔽表示Wait命令条件的值。

格式

Wait 值1 Mask 值2

参数

值1
　　指定表示Wait输入条件的值。
值2
　　指定以result返回的数值。

说明

Mask运算符用于对表示Wait输入条件的值进行位And运算。

参阅

Wait

Mask运算符使用示例

```
'在输入端口0的低3位变为1之前进行待机
Wait In(0) Mask 7 = 1
```

3.16.2 MCal

用于恢复增量编码器机器人的原点(检测机械原点)。

格式
MCal

说明
务必对增量型编码器的机器人执行原点恢复(检测机械原点)。请在打开电源之后执行这一原点恢复。如果在进行原点恢复之前执行动作命令或需要其它当前位置数据的命令，则会发生错误。

按照由MCordr命令指定的关节顺序执行原点恢复。由于出厂时的MCordr初始值因机械手机型而异，因此，详情请参阅以下手册。
《机械手手册》

易引起的错误

- 在执行MCal之前执行动作命令时

  如果在进行原点恢复之前执行动作命令或需要其它当前位置数据的命令(比如，Plist*命令等)，则会发生错误。

- 安装绝对编码器的机器人

  安装绝对编码器的机器人不需要MCal。

机器人设置安装注意事项

- 用于第3关节原点恢复的空间

  第3关节进行原点恢复时，首先，进行上升移动，然后下降，停在原点位置上。故此，确保第3关节可进行原点恢复的空间是至关重要的。作为推荐值，建议在Z上限值的上方确保6 mm以上的空间。(请勿在机器人上部6 mm之内的空间安装工具或夹具。)

参阅
Hofs、Home、Hordr、Mcorg、MCordr

MCal使用示例

如下所示为利用监视窗口的操作示例。

```
> Motor On
> Mcal
>
```

3.16.3 MCalComplete函数

用于返回MCal的状态。

格式

MCalComplete

返回值

MCal正常运作时返回 True，除此之外时返回False。

参阅

MCal

MCalComplete函数使用示例

```
If Not MCalComplete Then
    MCal
EndIf
```

3.16.4 MCordr

用于指定和显示通过MCal进行原点恢复时的关节动作顺序。仅限于装有增量编码器的机器人需要。

格式

(1) MCordr 设置值1, 设置值2, 设置值3, 设置值4 [, 设置值5] [, 设置值6] [, 设置值7] [, 设置值8] [, 设置值9]

(2) MCordr

参数

设置值1
以位模式(2进制数值)指定在MCal进程的第1步进行原点恢复的关节(0～9)。(请参阅下述位模式表。)
设置值2
以位模式(2进制数值)指定在MCal进程的第2步进行原点恢复的关节(0～9)。(请参阅下述位模式表。)
设置值3
以位模式(2进制数值)指定在MCal进程的第3步进行原点恢复的关节(0～9)。(请参阅下述位模式表。)
设置值4
以位模式(2进制数值)指定在MCal进程的第4步进行原点恢复的关节(0～9)。(请参阅下述位模式表。)
设置值5
以位模式(2进制数值)指定在MCal进程的第5步进行原点恢复的关节(0～9)。(请参阅下述位模式表。)
设置值6
以位模式(2进制数值)指定在MCal进程的第6步进行原点恢复的关节(0～9)。(请参阅下述位模式表。)
设置值7
以位模式(2进制数值)指定在MCal进程的第7步进行原点恢复的关节(0～9)。(请参阅下述位模式表。)
设置值8
以位模式(2进制数值)指定在MCal进程的第8步进行原点恢复的关节(0～9)。(请参阅下述位模式表。)
设置值9
以位模式(2进制数值)指定在MCal进程的第9步进行原点恢复的关节(0～9)。(请参阅下述位模式表。)

返回值

如果省略参数，则显示当前的机械原点恢复顺序。

说明

电源ON时，请务必在进行机械臂动作之前执行MCal命令。如果执行MCal命令，各关节则会移动到各自的原点返回位置。

指定执行MCal命令时的关节动作顺序。按照由设置值1指定的关节进行动作，结束原点恢复之后，由设置值2指定的关节进行动作这样的顺序，依次对设置值3对应的关节、设置值4对应的关节进行原点恢复。

MCordr命令的意义在于用户可变更原点恢复时各关节的恢复顺序。分9步设置恢复顺序。用户可利用通过MCordr指定各步骤恢复的关节。也可以指定多个要在各步骤进行恢复的关节。但一般来说，建议第1步最先移动第3关节，然后在此后的步骤中恢复其它关节。(请参阅注意。)

使用MCordr命令时，应指定9个步骤的对应位模式。各关节的位模式已经规定。如果在某步骤，位为“1”，对应的关节则进行原点恢复。如果位为“0”，对应的关节则不在该步骤进行原点恢复。按如下所述分配各关节的位模式。

位模式表

关节名	位编号	2进制数标记
第1关节	bit 0	&B000001
第2关节	bit 1	&B000010
第3关节	bit 2	&B000100
第4关节	bit 3	&B001000
第5关节	bit 4	&B010000

关节名	位编号	2进制数标记
第6关节	bit 5	&B100000
第7关节	bit 6	&B1000000
第8关节	bit 7	&B10000000
第9关节	bit 8	&B100000000

注意

- MCordr与Hordr的差异

  Hordr与MCordr命令之间存在明显的差异。MCordr与MCal同时使用，指定机器人原点恢复时的关节恢复顺序，而Hordr与Home同时使用的话，指定面朝原点位置的关节恢复顺序。

- 朝原点恢复位置的恢复顺序默认设置。

  如下所示为出厂设置。

- 第1步，进行第3关节(Z)的恢复。
- 第2步，第1关节(X)和第2关节(Y)恢复到原点恢复位置，并且第4关节(U)也同时恢复到原点恢复位置。
- 不使用第3步和第4步。如下所示为默认值。

```
MCordr &B0100, &B1011, 0, 0
```

- 通常，首先使第3关节(Z)进行原点恢复

  最先单独恢复第3关节(Z)的理由是在进行水平移动之前从工件表面移开工具。这在原点恢复时可防止工具干扰动作区域内的工件。

- 保持MCordr值

  由用户变更MCordr值，或在重新设置机器人之前保持该值。

参阅

MCal

MCordr使用示例

如下所示为利用监视窗口的4轴机器人操作示例。

本例所示为使用2进制数，按如下所述设置原点恢复顺序。

第1步对第3关节，第2步对第1关节，第3步对第2关节，第4步对第4关节进行原点恢复。

```
> MCordr  &B0100, &B0001, &B0010, &B1000
```

本例所示为使用10进制数，按如下所述设置原点恢复顺序。

第1步对第3关节进行原点恢复，第2步对第1关节、第2关节、第4关节同时进行原点恢复。

```
> MCordr  4, 11, 0, 0
```

下例所示为用10进制数显示当前的原点恢复顺序。

```
>mcordr
4, 11, 0, 0
>
```

3.16.5 MCordr函数

用于返回MCordr参数设置。

格式
Mcordr (设置值编号)

参数

设置值编号
以表达式或数值指定要引用的设置值编号(1～9的整数)。

返回值
以2进制数值(整数)返回要进行原点恢复的指定关节设置值。

说明
返回利用MCal进行原点恢复的关节动作顺序。

参阅
MCal

MCordr函数使用示例
如下所示为使用MCordr函数的程序示例。

```
Integer a
a = MCordr(1)
```

3.16.6 MemIn函数

用于返回指定存储器I/O端口的状态。各端口有8个存储器位。

格式
MemIn (端口编号)

参数

端口编号
　　指定存储器I/O的字节。

返回值
返回0～255的整数。返回值为8位，各个位分别对应于1个存储器I/O位。

说明
利用MemIn每次可查看8个存储器I/O位的值。MemIn命令可用于将8个存储器I/O位的状态保存为1个变量，或与Wait同时使用，在符合与2个以上I/O位有关的特定条件之前，使程序保持待机状态。

由于1次可获取8位的值，因此，返回值的范围为0～255。有关各返回值与各存储器I/O位状态的对应关系，请参照下表。

存储器I/O位表(使用端口0时)

返回值	7	6	5	4	3	2	1	0
1	Off	Off	Off	Off	Off	Off	Off	On
5	Off	Off	Off	Off	Off	On	Off	On
15	Off	Off	Off	Off	On	On	On	On
255	On	On	On	On	On	On	On	On

存储器I/O位表(使用端口31时)

返回值	255	254	253	252	251	250	249	248
3	Off	Off	Off	Off	Off	Off	On	On
7	Off	Off	Off	Off	Off	On	On	On
32	Off	Off	On	Off	Off	Off	Off	Off
255	On	On	On	On	On	On	On	On

注意

- MemIn与MemSw的差异

  可利用MemSw命令读取存储器I/O位1的值。MemSw的返回值为0或1时，表示存储器I/O位为On或Off。MemSw用于单独检查存储器I/O位。从检查存储器I/O位的状态方面来看，MemIn命令类似于MemSw。不同之处在于，MemSw命令逐位检查存储器I/O的状态，而MemIn命令则同时检查8位。MemIn的返回值返回的是0～255的值。可根据该值了解8位中的何者为On，何者为Off。

参阅
In, InBCD, Off, MemOff, On, MemOn, OpBCD, Oport, Out, MemOut, Sw, MemSw, Wait

MemIn使用示例

如下所示为求出最初8个存储器I/O的当前值之后，确认全部8个的值为“0”并继续执行的程序示例。返回值不是“0”时，显示错误信息，中止任务。

```
Function main
    Integer var1

    var1 = MemIn(0)  '获取最初8个存储器I/O位的值
    If var1 = 0 Then
        Go P1
        Go P2
    Else
        Print "Error in initialization!"
        Print "First 8 memory I/O bits were not all set to 0"
    EndIf
Fend
```

如下所示为利用命令窗口的简单操作示例。

```
> memout 0, 1
> print MemIn(0)
1
> memon 1
> print MemIn(0)
3
> memout 31,3
> print MemIn(31)
3
> memoff 249
> print MemIn(31)
1
>
```

3.16.7 MemInW函数

用于按字单位返回存储器I/O端口状态。字端口由16个存储器I/O位构成。

格式

MemInW (字端口编号)

参数

字端口编号
　　指定I/O的字端口。

返回值

返回存储器I/O端口状态(0～65535的Long型整数)。

参阅

MemIn、MemOut、MemOutW

MemInW函数使用示例

```
Long word0

word0 = MemInW(0)
```

3.16.8 MemOff

用于将存储器I/O的指定位设为OFF。

格式

MemOff {位编号 | 存储器IO标签 }

参数

位编号
以整数指定存储器I/O的位。
存储器IO标签
指定存储器I/O的标签。

说明

MemOff用于将指定的存储器I/O位设为OFF(0)。对由MemSw命令指定的存储器位执行状态检查。Wait命令也用于存储器位，并且在变为指定的存储器I/O状态之前使系统保持待机状态。

注意

- 存储器I/O的OFF

  在控制器重新启动时，存储器I/O被设为OFF。紧急停止、安全门打开、程序结束、Reset、Epson RC+ 重新启动时不设为OFF。

参阅

In、MemIn、InBCD、Off、On、MemOn、OpBCD、Oport、Out、MemOut、Sw、MemSw、Wait

MemOff使用示例

如下所示为使用2个分别启动动作命令的任务的示例。在这2个任务中，在各自的对方结束机器人动作命令时会启动联锁功能，以便按顺序获取控制。因此，2个任务可执行按顺序分别指示的动作语句。MemSw与Wait命令同时使用。为了确保安全，再次开始动作之前，等待存储器I/O位1变为适当的值。MemOn和MemOff用于对存储器I/O进行ON/OFF切换，以正确地实现同步。

```
Function main
    Integer I
    MemOff 1
    Xqt 2, task2
    For i = 1 to 100
        Wait MemSw(1) = Off
        Go P(i)
        MemOn 1
    Next I
Fend


Function task2
    Integer I
    For i = 101 to 200
        Wait MemSw(1) = On
        Go P(i)
        MemOff 1
    Next I
Fend
```

如下所示为利用命令窗口的简单操作示例。

```
> MemOn 1          '将存储器I/O位1设为ON
> Print MemSw(1)
```

```
1
> MemOff 1        '将存储器I/O位1设为OFF
> Print MemSw(1)
0
```

3.16.9 MemOn

用于将存储器I/O的指定位设为ON。

格式

MemOn {位编号 | 存储器IO标签}

参数

位编号
　　以整数指定存储器I/O的位。
存储器IO标签
　　指定存储器I/O的标签。

说明

MemOn用于将指定的位设为ON(1)。对由MemSw命令指定的存储器位执行状态检查。Wait命令也用于存储器位，并且在变为指定的S/W状态之前使系统保持待机状态。

注意

- 存储器I/O的OFF

  在控制器重新启动时，存储器I/O被设为OFF。紧急停止、安全门打开、程序结束、Reset、Epson RC+ 重新启动时不设为OFF。

参阅

In、MemIn、InBCD、Off、MemOff、On、OpBCD、Oport、Out、MemOut、Sw、MemSw、Wait

MemOn使用示例

如下所示为使用2个分别启动动作命令的任务的示例。在这2个任务中，在各自的对方结束机器人动作命令时会启动联锁功能，以便按顺序获取控制。因此，2个任务可执行按顺序分别指示的动作语句。MemSw与Wait命令同时使用。为了确保安全，再次开始动作之前，等待存储器I/O位1变为适当的值。MemOn和MemOff用于对存储器I/O进行ON/OFF切换，以正确地实现同步。

也可以使用Signal和Waitsig命令，实现任务同步。

```
Function main
    Integer I
    MemOff 1
    Xqt 2, task2
    For i = 1 to 100
        Wait MemSw(1) = Off
        Go P(i)
        MemOn 1
    Next I
Fend

Function task2
    Integer I
    For i = 101 to 200
        Wait MemSw(1) = On
        Go P(i)
        MemOff 1
    Next I
Fend
```

如下所示为利用命令窗口的简单操作示例。

```
> memon 1
> print memsw(1)
1
> memoff 1
> print memsw(1)
0
```

3.16.10 MemOut

用于同时设置8个存储器I/O位。

格式

MemOut 端口编号, 输出数据

参数

端口编号

指定存储器I/O的字端口编号。按如下所述，端口编号对应相应的位。

端口编号	输出数据
0	0-7
1	8-15
...	...

输出数据

以0～255的整数值指定由端口编号指定的输出组的输出模式。以16进制数显示时，范围为&H0～&HFF。低数位表示低位(或最初的4个输出位)，高数位表示高位(或后续的4个输出位)。

说明

MemOut用于通过组合指示设置输出位的端口编号与输出数据，同时设置8个存储器I/O位。利用端口编号参数指定使用哪组(哪8个输出位)。比如，端口编号 = 0 时，指定输出位0～7。端口编号 = 1 时，指定输出位8～15。

首先，利用端口编号指定8个输出位之后，利用输出数据参数来定义特定的输出模式。输出数据参数可获取的值为0～255，以16进制数或10进制数的整数进行指定。(&H0～&HFF或0～255)

下表所示为部分I/O组合示例，以及端口编号为“0”和“1”时分别对应的输出数据值。

端口编号=0时的输出设置(输出位编号)

输出数据值	7	6	5	4	3	2	1	0
01	Off	Off	Off	Off	Off	Off	Off	On
02	Off	Off	Off	Off	Off	Off	On	Off
03	Off	Off	Off	Off	Off	Off	On	On
08	Off	Off	Off	Off	On	Off	Off	Off
09	Off	Off	Off	Off	On	Off	Off	On
10	Off	Off	Off	Off	On	Off	On	Off
11	Off	Off	Off	Off	On	Off	On	On
99	Off	On	On	Off	Off	Off	On	On
255	On	On	On	On	On	On	On	On

端口编号=1时的输出设置(输出位编号)

输出数据值	15	14	13	12	11	10	9	8
01	Off	Off	Off	Off	Off	Off	Off	On
02	Off	Off	Off	Off	Off	Off	On	Off
03	Off	Off	Off	Off	Off	Off	On	On

输出数据值	15	14	13	12	11	10	9	8
08	Off	Off	Off	Off	On	Off	Off	Off
09	Off	Off	Off	Off	On	Off	Off	On
10	Off	Off	Off	Off	On	Off	On	Off
11	Off	Off	Off	Off	On	Off	On	On
99	Off	On	On	Off	Off	Off	On	On
255	On	On	On	On	On	On	On	On

参阅

In、MemIn、InBCD、Off、MemOff、On、MemOn、OpBCD、Oport、Out、Sw、MemSw、Wait

MemOut使用示例

如下所示为启动名为“iotask”的主任务的程序。“iotask”是将S/W存储器I/O位的0～3设为ON/OFF的简单任务。如果使用MemOut命令，则可利用1个命令进行上述操作，而不必单独将S/W存储器I/O位设为ON/OFF。

```
Function main
  Xqt 2, iotask
  Go P1
  .
  .
Fend

Function iotask

  Do
    MemOut 0, &H

    Wait 1
    MemOut 0, &H0
    Wait 1
  Loop
Fend
```

如下所示为利用命令窗口的简单操作示例。

```
> MemOut 1,6     '将存储器I/O位9和10设为ON
> MemOut 2,1     '将存储器I/O位8设为ON
> MemOut 3,91     '将存储器I/O位24、25、27、28、30设为ON
```

3.16.11 MemOutW

用于以16位并同时按字单位设置存储器I/O端口的状态。

格式

MemOutW 字端口编号, 输出数据

参数

字端口编号
　　指定存储器I/O的字。
输出数据
　　以表达式或数值指定存储器I/O数据(0～65535的整数)。

说明

将由参数字端口编号指定的存储器I/O端口组的状态变更为指定的输出数据。

参阅

MemIn、MemInW、MemOut

MemOutW使用示例

```
MemOutW 0, 25
```

3.16.12 MemSw函数

用于返回指定存储器I/O位的状态。

格式
MemSw (位编号)

参数

位编号
以表示存储器I/O位编号的数值进行指定。

返回值
指定的位为ON时返回“1”，为OFF时返回“0”。

说明
MemSw用于返回1个存储器I/O位的状态。MemSw通常与MemOn及MemOff命令配套使用。MemOn用于将指定的位设为ON，MemOff则用于将指定的位设为OFF。

参阅
In、MemIn、InBCD、Off、MemOff、On、MemOn、OpBCD、Oport、Out、MemOut、Sw、Wait

MemSw使用示例
在下例当中，2个任务分别具有启动动作命令的功能，并进行联锁，一方未进行机器人控制时，另一方则会控制机器人动作。这样，各任务就可以完全按照预定的顺序执行所赋予的动作语句。

通过与Wait命令同时使用MemSw，可在存储器I/O位1变为可安全进行后续动作的适当值之后，再开始动作。

```
Function main
    Integer I
    MemOff 1
    Xqt 2, task2
    For i = 1 to 100
        Wait MemSw(1) = Off
        Go P(i)
        MemOn 1
    Next I
Fend

Function task2
    Integer I
    For i = 101 to 200
        Wait MemSw(1) = On
        Go P(i)
        MemOff 1
    Next I
Fend
```

如下所示为利用命令窗口的简单操作示例。

```
> memon 1
> print memsw(1)
1
> memoff 1
> print memsw(1)
0
```

3.16.13 MHour函数

用于返回机器人电动机的累计励磁时间。

格式

MHour ([机器人编号])

参数

机器人编号
以整数值指定要确认励磁时间的机器人的编号。已省略时，以当前选择的机器人为对象。

返回值

以实值返回电动机的累计励磁时间。

参阅

Time、Hour

MHour函数使用示例

```
Robot 2
Print MHour
Print MHour(1)
```

3.16.14 Mid$函数

用于从利用字符串指定的起始位置提取指定字符数的字符。

格式

Mid$ (字符串, 起始位置 [,字符数])

参数

字符串
　　指定源字符串。
起始位置
　　指定提取字符串的起始位置。
字符数
　　指定从字符串中提取的字符数。可省略。如果省略，则返回起始位置～字符串最后的字符。

返回值

返回从源字符串中提取的字符串。

说明

Mid$用于从利用源字符串指定的起始位置提取字符数部分的字符串。

参阅

Asc、Chr$、InStr、Left$、Len、Right$、Space$、Str$、Val

Mid$函数使用示例

下例所示为从字符串“ABCDEFGHIJ”出的正中间提取2个字符以及起始位置5的剩余字符串。

```
Function midtest
    String basestr$, m1$, m2$
    basestr$ = "ABCDEFGHIJ"
    m1$ = Mid$(basestr$, (Len(basestr$) / 2), 2)
    Print "The middle 2 characters are: ", m1$
    m2$ = Mid$(basestr$, 5)
    Print "The string starting at 5 is: ", m2$
Fend
```

3.16.15 MkDir

用于生成子目录。

格式
MkDir 目录名

参数

目录名
　　指定要生成子目录的路径和名称的字符串。有关路径的详细说明，请参阅ChDisk。

说明
在指定的路径上生成新目录。已省略路径时，在当前目录内生成子目录。

注意

可在PC硬盘时执行。

参阅
ChDir、ChDrive、RenDir、RmDir

MkDir使用示例
利用命令窗口的操作示例

```
> MkDir \Data

> MkDir \Data\PTS

> MkDir \TEST1 \TEST2
```

3.16.16 Mod运算符

用于返回数值表达式除以其它数值表达式得到的商值。

格式
被除数 Mod 除数

参数

被除数
　　指定被除的数。
除数
　　指定除数。

结果
返回被除数除以除数的商值。

说明
Mod用于获取2个数值相除的商值(整数)。使用Mod命令对于调查偶数或奇数等是非常便利的。

如下所示为Mod命令的步骤

：用被除数除以除数。将除法运算的商作为Mod命令的返回值进行返回。

参阅
Abs, Atan, Atan2, Cos, Int, Not, Sgn, Sin, Sqr, Str$, Tan, Val

Mod运算符使用示例
下例所示为调查某数值(var1)为偶数还是奇数。如果数值为偶数，Mod命令则返回结果“0”。如果为奇数，则返回结果“1”。

```
Function modtest
Integer var1, result

Print "Enter an integer number:"
Input var1
result = var1 Mod 2
Print "Result = ", result
If result = 0 Then
Print "The number is EVEN"
Else
Print "The number is ODD"
EndIf
Fend
```

作为其它使用示例，下面列举了利用命令窗口进行的操作示例。

```
> Print 36 Mod 6
> 0

> Print 25 Mod 10
> 5
>
```

3.16.17 Motor

用于将机器人的所有轴的电动机设为ON或OFF。

格式

Motor {On | Off}

参数

On | Off
要将电动机电源设为ON状态时，指定On；要设为OFF状态时，指定Off。

说明

Motor On命令用于将电动机电源设为ON，并解除所有轴的制动。Motor Off命令用于将电动机电源设为OFF，并设置所有轴的制动。

要开动机器人时，电动机电源必须处于ON状态。

发生紧急停止或需要Reset命令这样的错误之后，在执行Reset命令之后设为Motor On。

如果执行Motor On命令，则将机器人控制参数设为下述设置值。

机器人控制参数

- Speed和SpeedR、SpeedS的设置值：(被初始化为初始值。)
- Accel和AccelR、AccelS的设置值：(被初始化为初始值。)
- QPDecelR、QPDecelS的设置值：(被初始化为初始值。)
- LimZ参数的设置值：(初始化为0。)
- CP参数的设置值：(初始化为Off。)
- SoftCP参数的设置值：(初始化为Off。)
- Fine的设置：(初始化为初始值。)
- Power Low设置：(变为低功率模式。)
- PTPBoost的设置值：(初始化为初始值。)
- TCLim、TCSpeed的设置值：(初始化为初始值。)
- PgLSpeed的设置值：(初始化为初始值。)
- PerformMode的设置值：(初始化为标准模式。)

参阅

Brake、Power、Reset、SFree、SLock

Motor使用示例

下例所示为通过命令窗口执行的情况。

```
> Motor On

> Motor Off
```

3.16.18 Motor函数

用于返回已指定机器人电动机电源的状态。

格式

Motor [(机器人编号)]

参数

机器人编号
以整数值指定要确认状态的机器人编号。已省略时，以当前选择的机器人为对象。

返回值

- 0 = 电动机OFF
- 1 = 电动机ON

参阅

Motor

Motor函数使用示例

```
If Motor = Off Then
    Motor On
EndIf
```

3.16.19 Move

用于在当前位置～指定位置之间以直线动作移动机械臂。

格式

Move 目标坐标 [ROT] [ECP] [CP] [Till | Find] [!并行处理!] [SYNC]

参数

目标坐标
　　以点数据指定目标位置。
ROT
　　以工具姿势变化为优先，确定动作速度、加减速度。可省略。
ECP
　　指定外部控制点动作。可省略。(仅在使用ECP选项时有效)
CP
　　指定路径运动。可省略。
Till | Find
　　记述Till或Find表达式。可省略。

```
Till | Find
Till Sw(表达式) = {On | Off}
Find Sw(表达式) = {On | Off}
```

! 并行处理 !
　　动作期间可附加并行处理语句，以执行I/O等命令。可省略。
SYNC
　　预约动作命令。在通过SyncRobots开始动作之前，机器人不进行动作。

说明

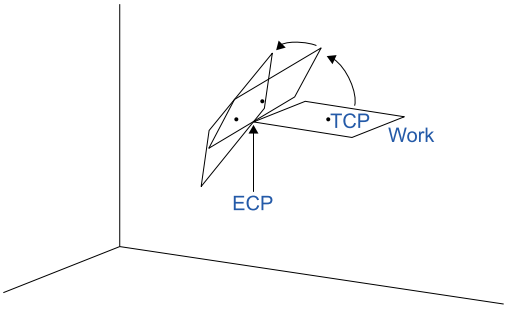
Move用于在当前位置～指定位置之间直线移动机械臂。使用Move时，所有关节同时开始/停止动作。执行Move命令之前，请事先对目标坐标进行示教。由AccelS命令控制Move的加减速。由SpeedS命令控制Move的速度。即使有1个由SpeedS设置的值超出各关节的容许速度，也会切断电动机的励磁，机器人停止动作。

Move的速度和加减速度分别使用SpeedS和AccelS的设置值。有关速度与加减速度之间的关系，请参阅“注意”中的“与CP同时使用Tmove”。不过，指定ROT修饰参数时的速度和加减速度分别使用SpeedR和AccelR的设置值。此时，SpeedS和AccelS的设置值变为无效状态。

通常，移动距离为“0”，但如果仅进行姿势关节的动作，会发生错误。通过附加ROT修饰参数并以工具姿势变化的加速度为优先，可不出错误地进行动作。已经附加ROT修饰参数时，如果没有姿势变化，并且移动距离不是“0”，则会发生错误。

另外，相对于移动距离，工具姿势变化速度过大时，或指定的转速超过机械手限度时，也会发生错误。此时，请降低指定速度，或附加ROT修饰参数，并以姿势变化的加减速度为优先。

使用ECP时，在对应于指定ECP编号的外部控制点上，工件沿着直线移动。此时，顶端关节的中心不沿着直线移动。



要在Move动作完成之前对机器人执行减速停止时，用户可使用Till修饰符指定该条件。此处指定的条件就是检查其中1个输入位，因此需要使用Sw命令。用户检查输入的状态是ON还是OFF，并根据指定的条件停止机械臂动作。该功能类似于输入条件成立时停止Move的中断。如果Move动作期间输入条件从未成立，机械臂则到达由目标坐标指定的位置。

Till修饰符可省略。有关Till修饰符的详细说明，请参阅Till命令。

注意

- 不能利用Move进行的操作

  进行动作之前，不能确认动作范围。这样的话，即使目标坐标位置在容许动作范围之内，而如果到达此处的轨迹通过容许动作范围以外位置，则可能会导致机械臂突然停止，并造成伺服冲击，导致发生故障，这很危险。为了防止发生这种情况，高速执行Move之前，请先以低速确认动作范围。也就是说，即使目标坐标在机械臂动作范围之内，从物理角度来讲，如果通过Move动作到达此处的轨迹超出机械臂容许动作范围，机械臂则动不了。

- 与CP同时使用Move

  如果使用CP参数，动作命令则会在开始减速的同时将控制移交给下一语句。这在用户连续发出几个动作命令，要以一定的速度进行连续动作时非常便利。为未指定CP的Move命令时，机械臂必须减速，以停在指定的目标位置上。

- 向Move发出适当的速度/加减速度指示

  SpeedS和AccelS命令用于指定Move动作期间的机械手速度和加减速度，而SpeedS和AccelS则为针对直线和曲线动作的命令，敬请注意它们之间的区别。Speed和Accel命令适用于PTP动作。

易引起的错误

- 执行直线移动距离为0的动作时

  如果要利用Move进行仅使4自由度机器人(水平多关节型(包括RS系列)等)的U坐标值或6自由度机器人(垂直6轴型(包括N系列))的U、V、W坐标值发生变化的动作，则会发生错误。此时请使用ROT参数。

- 超出关节限制速度的错误

  进行指示的动作期间，即使1个关节超出容许速度，也会发生超速错误。发生电动机超速错误时，机械臂停止动作，电动机励磁被切断。

- RS系列执行通过原点附近的动作时

  如果RS系列利用Move执行通过原点附近的动作，则可能会发生超速错误。请针对通过原点附近的动作采取下述防范措施。
  - 请降低SpeedS的设置速度。
  - 请变更为不通过原点附近的通路。
  - 请使用Go等PTP动作以替代Move。

参阅

AccelS、Arc、CP、Go、Jump、Jump3、Jump3CP、SpeedS、Sw、Till

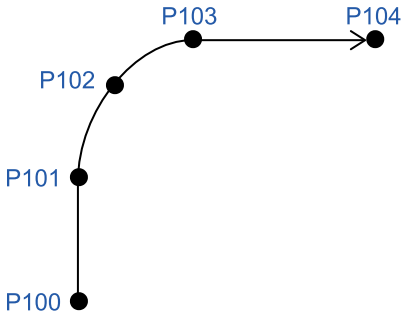
Move使用示例

如下所示为在P0～P1之间执行机械臂PTP动作之后，直线返回到P0的简单示例。在程序的后半段，机械臂面向P2进行直线移动，直至输入位2变为ON状态。如果动作期间输入位2变为ON状态，机械臂则进行减速停止(即使没有到达P2)，然后执行下一程序命令。

```
Function movetest
  Home
  Go P0
  Go P1
```

```
  Move P0
  Move P2 Till Sw(2) = On
  If Sw(2) = On Then
    Print "Input #2 came on during the move and"
    Print "the robot stopped prior to arriving on"
    Print "point P2."
  Else
    Print "The move to P2 completed successfully."
    Print "Input #2 never came on during the move."
  EndIf
Fend
```

如下所示为与CP同时使用Move的示例。下图所示为从P100开始的圆弧轨迹。在P100～P101之间进行直线移动，圆弧从P101开始，通过P102向P103绘制圆弧。P103～P104之间为直线动作，在此期间进行减速停止。在P104停止之前，不会在中途的点上进行减速停止，这点敬请注意。利用下述函数对这种动作进行编程。




```
Function CornerArc
   Go P100
   Move P101 CP        '不在P101停止
   Arc P102, P103 CP   '不在P103停止
   Move P104           '减速停止到P104
Fend
```

3.16.20 MsgBox

用于显示对话框信息并等待用户选择按钮。

格式

MsgBox 信息$ [, 按钮的类型] [, 标题$] [, 接收结果的整数]

参数

信息$

显示的信息

按钮的类型

指定相关数值或表达式。相关数值是指指定显示按钮数量和类型、图标样式、按钮标题等值的合计值。Epson RC+里包括事先确定用于该参数的常数，并使用下表所示的值。可省略。

常数	值	含义
MB_OK	0	仅显示[OK]按钮
MB_OKCANCEL	1	显示[OK]和[取消按钮]
MB_ABORTRETRYIGNORE	2	显示[中止]、[重试]、[无视]按钮
MB_YESNOCANCEL	3	显示[是]、[否]、[取消按钮]
MB_YESNO	4	显示[是]、[否]按钮
MB_RETRYCANCEL	5	显示[重试]和[取消按钮]
MB_ICONSTOP	16	Stop符号
MB_ICONQUESTION	32	“?”标记
MB_ICONEXCLAMATION	64	“!”标记
MB_DEFBUTTON1	0	第1个按钮为默认
MB_DEFBUTTON2	256	第2个按钮为默认

标题$

指定在对话框标题栏上显示的字符。可省略。

接收结果的整数

指定接收表示用户选择值(整数)的变量。Epson RC+里包括事先确定用于该参数的常数。下表所示为由该参数返回的值。可省略。

常数	值	含义
IDOK	1	选择[OK]按钮。
IDCANCEL	2	选择[取消]按钮。
IDABORT	3	选择[中止]按钮。
IDRETRY	4	选择[重试]按钮。
IDYES	6	选择[是]按钮。
IDNO	7	选择[否]按钮。

说明

MsgBox用于自动对信息执行格式化。要设为空白状态时，对信息使用CRLF。请参阅下例。

参阅

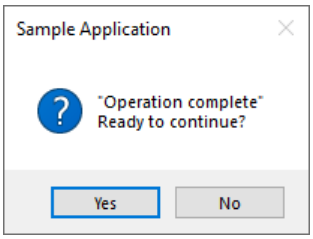
InputBox

MsgBox使用示例

下例所示为向用户显示是否继续作业的确认信息框。信息框中显示[是]和[否]2个按钮。也显示“?”显标记图标。如果用户选择按钮并返回MsgBox，则可以再考量答复。如果选择“否”，则利用Quit命令结束所有任务。

```
Function msgtest
  String msg$, title$
  Integer mFlags, answer

  msg$ = Chr$ (34) + "Operation complete" + Chr$ (34)  + CRLF
  msg$ = msg$ + "Ready to continue?"
  title$ = "Sampl Application"
  mFlags = MB_YESNO + MB_ICONQUESTION
  MsgBox msg$, mFlags, title$, answer
  If answer = IDNO then
    Quit All
  EndIf
Fend
```

下述画面所示为由上述代码生成的信息框。



限制事项

如果参数的msg$和title$中包含半角逗号“,”，将无法正确显示字符串。请使用不含半角逗号的字符串。

3.16.21 MyTask函数

用于返回当前程序任务编号。

格式

MyTask

返回值

用于返回当前任务的任务编号。任务编号的范围为下述整数值。

- 一般任务：1~32
- 后台任务：65～80
- Trap任务：257~267

说明

MyTask用于以数值返回当前程序的任务编号。在程序中记述MyTask函数并执行时，返回执行程序的任务编号。

参阅

Xqt

MyTask函数使用示例

下例所示为将1～8的I/O端口设为ON/OFF。

```
Function main
  Xqt 2, task      '执行任务2
  Xqt 3, task      '执行任务3
  Xqt 4, task      '执行任务4
  Xqt 5, task      '执行任务5
  Xqt 6, task      '执行任务6
  Xqt 7, task      '执行任务7
  Xqt 8, task      '执行任务8
  Call task
Fend

Function task
  Do
    On MyTask      '将编号与当前任务编号相同的I/O端口设为ON
    Off MyTask     '将编号与当前任务编号相同的I/O端口设为OFF
  Loop
Fend
```

3.17 N

3.17.1 Next

For和Next命令组合使用以形成循环。重复用户指定次数的For与Next之间的一系列命令。

格式

For 变量名1 = 初始值To结束值 [Step增量值] 语句 Next变量名1

参数

变量名1
　在For/Next循环中指定代入重复数据的计数器变量名。通常定义为整数变量，不过也可以指定实数。
默认值
　在指定的变量中以表达式或直接以数值指定循环的最初代入数值。
结束值
　以表达式或直接以数值指定表示循环结束的数值。如果该值成立，For/Next循环则会结束并转向执行Next命令的下一语句。
增量值
　以表达式或直接以数值指定每次For/Next循环时增加变量的值。该值可指定为正值或负值。但指定负值时，请设为初始值 > 结束值。如果省略该参数，则自动将“1”果作为增量值进行增减。可省略。
语句
　用于记述插入到For/Next循环中的SPEL+语句。

说明

For/Next用于按指定次数重复循环内的一系列语句。循环以For语句开始，以Next语句结束。利用变量对循环内的语句重复执行的次数进行计数。

初始值为计数器的最初数值。如果正确设置结束值与增量值，则正值或负值均可。

结束值为计数器的最终数值。如果达到该值，For/Next循环则会结束，程序控制将移交给Next命令的后续命令。

到达Next命令之前，程序执行For语句的下一语句。仅按由增量值设置的值增减计数器变量。如果未设置增量值，则按“1”的步幅进行增减。

接下来比较计数器变量值与结束值。如果计数器变量值小于或等于结束值，则返回到For语句之后的命令的执行。如果计数器变量值大于结束值，则转向For/Next循环的下一个执行，在Next命令之后的命令下继续执行。

注意

- 负增量值

  增量值为负值时，计数器变量值随着循环的进行逐渐减少。此时请务必确保初始值大于结束值。

参阅

For

For/Next使用示例

```
Function fornext
    Integer ctr
    For ctr = 1 to 10
        Go Pctr
    Next ctr
    '
    For ctr = 10 to 1 Step -1
        Go Pctr
```

```
    Next ctr
Fend
```

3.17.2 Not运算符

用于以位为单位计算数值的补数。

格式
Not运算符

参数

操作数
　　指定整数值。

结果
返回操作数的1的补数。

说明
Not函数用于以位为单位计算操作数的补数。结果位反转所对应操作数的位。0位转换为1，1位转换为0。

参阅
Abs、And、Atan、Atan2、Cos、Int、LShift、Mod、Or、RShift、Sgn、Sin、Sqr、Str$、Tan、Val、Xor

Not使用示例
如下所示为通过命令窗口中使用Not的示例。

```
>print not(1)
 -2
>
```

3.18 O

3.18.1 Off

用于将由位编号指定的输出位设为OFF，或在指定时间OFF之后设为ON。

格式

Off { 输出位编号|输出标签 } [, 时间] [, 非同步指定] [, Forced]

参数

输出位编号
以整数值指定要设为OFF的I/O输出位。

输出标签
指定输出标签。

时间
以秒指定输出位设为OFF的时间。经过指定的时间之后，输出位再变为ON。请以0.01秒～10秒的范围指定OFF时间。可省略。

非同步指定
如果设置计时器，则按非同步指定执行后续命令的时序。可省略。

- 0 - 在将输出位设为OFF的同时执行后续命令。
- 1 - 默认设置。指定时间OFF之后变为ON并执行后续命令。

Forced
可省略。通常会省略。

说明

Off用于将指定的输出位返回到OFF(或0)。

如果指定时间参数，则在指定时间OFF之后，按输出位编号指定的输出位再次设为ON。如果执行OFF之前输出位已变为OFF，则在经过指定时间后设为ON。

在指定时间为下述设置时，非同步指定设置有效。

- 1：将输出位设为OFF，经过指定时间后再设为ON，然后执行后续命令。(非同步指定设置以此为默认设置。如果省略该参数，则为“1”的设置。)
- 0：将输出位设为OFF，同时执行后续命令。

注意

- 分配用于远程控制的输出位

  如果指定设置用于系统输出的输出位，则会发生错误。远程输出位会根据系统的状态自动设为ON/OFF。

- 发生紧急停止时

  Epson RC+发生紧急停止时，所有输出位都会变为OFF状态。如果要在紧急停止时保持设置，则可通过[设置]菜单-[系统设置]-[控制器]-[环境设置]面板的[紧急停止时将输出端口设为OFF]复选框重新进行设置。

- Forced标志

  在要通过NoPause任务、NoEmgAbort任务(执行Xqt时指定NoPause或NoEmgAbort以开始的特别任务)，在紧急停止期间或安全门打开时将I/O输出设为OFF的情况下，指定此标志。

  紧急停止期间或安全门打开时，I/O输出会发生变化，因此，在系统设计方面需要注意。

参阅

In、InBCD、MemOn、MemOff、MemOut、MemSw、OpBCD、Oport、Out、Wait

Off使用示例

下例所示为启动名为“iotask”的主任务的情况。“iotask”是分别将输出位1和2设为ON/OFF，并在10秒钟之后再次执行的简单任务。

```
Function main
  Xqt 2, iotask
  Go P1
  .
  .
  .
Fend


Function iotask
  Do
    On 1
    On 2
    Off 1
    Off 2
    Wait 10
  Loop
Fend
```

如下所示为利用命令窗口的简单操作示例。

```
> on 1
> off 1, 10     '将输出1设为OFF，10秒钟之后再次设为ON
> on 2
> off 2
```

3.18.2 OLAccel

用于设置基于过载率的加减速度自动调整。

格式

OLAccel {On | Off}

参数

On | Off

- On：将基于过载率的加减速度自动调整设为有效。
- Off：解除基于过载率的加减速度自动调整。

说明

OLAccel用于设置是否将基于机器人过载率(OLRate)的加减速度自动调整功能设为有效。如果将OLAccel设为On，执行PTP动作命令时，则根据当时的机器人过载率自动调整加减速度。也就是说，如果在执行PTP动作时机器人过载率超出一定的值，则自动降低动作时的加减速度，这样就起到预防过载错误发生的作用。原来，对于发生过载错误的高占空比动作，客户需要通过编程来停止机器人或调节速度或加速度来加以应对，通过使用OLAccel，这种必要性就会减少。但是，由于不可能所有循环都不发生过载错误，对于按非常高的占空比运行的大负载循环，未必能消除过载错误的发生。在这种情况下，需要客户停止机器人运转或调节速度和加减速度。另外，根据使用环境，可能在并不发生过载错误的状态下，因机器人连续进行动作而导致电动机温度上升，从而发生过热错误。

如果在适当的负载状态下，则不使用该命令。

测试循环时，可使用OLRate检查机器人是否处于易引起过载错误的状态。

下述某种情况时，OLAccel值会被初始化。

- 启动控制器时
- 执行Motor On
- 执行SFree、SLock、Brake
- 执行Reset、Reset Error
- 利用停止按钮或执行Quit All等结束任务

注意

如果对不支持加减速度自动调整功能的机器人执行OLAccel On，则会发生错误。

参阅

OLAccel函数、OLRate

OLAccel使用示例

```
>olaccel on
>olaccel
OLACCEL is ON

Function main
Motor On
Power High
Speed 100
Accel 100, 100
OLAccel On
Xqt 2, MonitorOLRate
Do
Jump P0
Jump P1
```

```
Loop
Fend

Function MonitorOLRate
Do
'显示OLRate
OLRate
Wait 1
Loop
Fend
```

3.18.3 OLAccel函数

用于返回基于过载率的加减速度自动调整的状态。

格式

OLAccel

返回值

- Off = 解除基于过载率的加减速度自动调整
- On = 基于过载率的加减速度自动调整有效

参阅

OLAccel、OLRate

OLAccel函数使用示例

```
If OLAccel = Off Then
  Print "OLAccel is off"
EndIf
```

3.18.4 OLRate

用于显示指定关节过载率。

格式
OLRate [关节编号]

参数

关节编号
以1～9的整数进行指定。附加轴的S轴为8，T轴为9。

说明
OLRate是用于显示指定关节过载率的函数。OLRate可用于检查循环是否对关节施加了过大的负担。如果在负载较大的循环中使用，温度或电流原因都可能会导致伺服错误。利用OLRate检查机器人是否处于易发生伺服错误的状态。

循环运转期间，执行监视OLRate以外的其它任务。如果存在OLRate超出1.0的关节，则会发生伺服错误。

负载过大时，最易发生伺服错误。测试循环时，可使用OLRate确认机器人的速度或加速度，以便在实际使用时采取预防措施，以免发生伺服错误。

请在机器人动作期间执行OLRate，以便获取有效值。

如果在适当的负载状态下，则不使用该命令。

参阅
OLRate函数

OLRate使用示例

```
>olrate
0.10000  0.20000
0.30000  0.40000
0.50000  0.60000

Function main
  Power High
  Speed 50
  Accel 50, 50
  Xqt 2, MonitorOLRate
  Do
    Jump P0
    Jump P1
  Loop
Fend

Function MonitorOLRate
  Do
    OLRate    '显示OLRate
    Wait 1
  Loop
Fend
```

3.18.5 OLRate函数

用于返回指定关节过载率。

格式
OLRate (关节编号)

参数

关节编号
以1～9的整数进行指定。附加轴的S轴为8，T轴为9。

返回值
返回指定关节过载率。返回值的范围为0.0～2.0。

说明
OLRate可用于检查循环是否对关节施加了过大的负担。如果在负载较大的循环中使用，温度或电流原因都可能会导致伺服错误。利用OLRate检查机器人是否处于易发生伺服错误的状态。

循环运转期间，执行监视OLRate以外的其它任务。如果存在OLRate超出1.0的关节，则会发生伺服错误。

负载过大时，最易发生伺服错误。测试循环时，可使用OLRate确认机器人的速度或加速度，以便在实际使用时采取预防措施，以免发生伺服错误。

请在机器人动作期间执行OLRate，以便获取有效值。

如果在适当的负载状态下，则不使用该命令。

参阅
OLRate

OLRate函数使用示例

```
Function main
  Power High
  Speed 50
  Accel 50, 50
  Xqt 2, MonitorOLRate
  Do
    Jump P0
    Jump P1
  Loop
Fend

Function MonitorOLRate
  Integer i
  Real olRates(4)
  Do
    For i = 1 to 4
      olRates(i) = OLRate(i)
      If olRate(i) > .5 Then
        Print "Warning: OLRate(", i, ") is over .5"
      EndIf
    Next i
  Loop
Fend
```

3.18.6 On

用于将指定的输出位设为ON，经过指定时间后再设为OFF。

格式

On { 输出位编号|输出标签 } [, 时间] [, 非同步指定] [, Forced]

参数

输出位编号
以整数值指定要设为ON的I/O输出位。

输出标签
指定输出标签。

时间
以秒指定指定输出位设为ON的时间。经过该时间后，指定输出位设为OFF。(请指定0.01秒钟以上的经过时间。)
可省略。

非同步指定
已进行时间设置时，可按非同步指定来指定后续命令的执行时序。可省略。

- 0 - 在将输出位设为ON的同时执行后续命令。此时的最大时间设置为10秒。
- 1 - 默认设置。指定时间ON之后设为OFF并执行后续命令。

Forced
可省略。通常会省略。

说明

On用于将指定的输出位设为ON(设为1)。如果设置时间参数，则将指定的输出位设为ON，经过指定时间后再设为OFF。

已设置时间时，按如下所述使用非同步指定参数的设置。

- 1：将输出位设为ON，经过指定时间后再设为OFF，然后执行后续命令。(非同步指定设置以此为默认设置。如果省略该参数，则为“1”的设置。)
- 0：将输出位设为ON，同时执行后续命令。

注意

- 远程设置的输出位

  如果指定远程设置的输出位，则会发生错误。远程输出位会根据系统的状态自动设为ON或OFF。有关远程的详细说明，请参阅以下手册。
  《Epson RC+ 用户指南》

  要将远程连接器的各个位设为输出或I/O时，通过 Epson RC+的[设置]菜单 - [系统配置] - [控制器] - [远程控制]面板进行设置。

- 发生紧急停止时

  机器人控制器发生紧急停止时，所有输出位都会变为OFF状态。如果要在紧急停止时保持设置，则可通过[设置]菜单-[系统设置]-[控制器]-[环境设置]面板的[紧急停止时将输出端口设为OFF]复选框重新进行设置。

- Forced标志

  在要通过NoPause任务、NoEmgAbort任务(执行Xqt时指定NoPause或NoEmgAbort以开始的特别任务)以及后台任务，在紧急停止期间或安全门打开时将I/O输出设为ON的情况下，指定此标志。

  紧急停止期间或安全门打开时，I/O输出会发生变化，因此，在系统设计方面需要注意。

参阅

In、InBCD、MemOff、MemOn、Off、OpBCD、Oport、Out、Wait

On使用示例

下例所示为启动名为“iotask”的主任务的情况。“iotask”是分别将输出位1和2设为ON/OFF，并在10秒钟之后再次执行的简单任务。

```
Function main
    Xqt iotask
    Go P1
    .
    .
    .
Fend


Function iotask
    Do
        On 1
        On 2
        Off 1
        Off 2
        Wait 10
    Loop
Fend
```

如下所示为利用命令窗口的简单操作示例。

```
> on 1
> off 1, 10     '将输出1设为OFF，10秒钟之后再次设为ON
> on 2
> off 2
```

3.18.7 OnErr

用于设置发生错误时将控制分支给错误处理子例程的中断。这样用户就可处理错误。

格式

OnErr GoTo { 标签 | 0}

参数

标签
　　指定发生错误时移交目标的语句标签。
0
　　指定清除OnErr设置的参数。

说明

用户可利用OnErr进行错误处理。如果未使用OnErr，发生错误时任务则会被中止，并显示错误。而如果使用OnErr，则可将控制移交给错误处理子例程，以便自动从错误恢复过来。错误恢复之后，控制移交给由EResume命令指定的返回目标。这样，即使发生错误，也可以自动进行错误处理，不必中断任务的执行。另外，由于始终按相同的方式自动处理容易复杂化的问题，因此，可更顺利地发挥工作单元的作用。

利用OnErr命令指定并设置参数“0”时，当前的OnErr设置则被清除。(执行OnErr 0之后，如果发生错误，则停止执行程序。)

参阅

Err、EResume

OnErr使用示例

在如下所示的简单实用程序示例中，检查点数据P0-P399是否存在。如果不存在点数据，画面中则会显示“该点不存在”的信息。该程序使用CX命令来测试是否定义了点。如果未定义，则将控制移交给错误处理子例程，并在画面中显示未定义的点。

```
Function errDemo
    Integer i, errNum

    OnErr GoTo errHandler

    For i = 0 To 399
        temp = CX(P(i))
    Next i
    Exit Function
    '
    '
    '*******************************************
    '* Error Handler                           *
    '*******************************************
errHandler:
    errNum = Err
    '确认是否使用未定义点
    If errNum = 7007 Then
       Print "Point number P", i, " is undefined!"
    Else
       Print "ERROR: Error number ", errNum, " occurred while"
       Print "       trying to process point P", i, " !"
    EndIf
    EResume Next
Fend
```

3.18.8 OpBCD

用于以2进制编码的10进制数(BCD)同时设置8位输出。

格式

OpBCD 端口编号, 输出数据 [, Forced]

参数

端口编号

指定I/O的输出字节。按如下所述，指定数值对应各自的输出位。

端口编号	输出位
0	0-7
1	8-15
2	16-23
3	24-31
...	...

输出数据

以0～99的整数值指定由端口编号指定的输出组的输出模式。第2位(个位)表示所选组中的低4位输出位，第1位(10位)表示高4位输出位。

Forced

可省略。通常会省略。

说明

OpBCD用于以BCD同时设置8个输出位。按每8个为一组，对标准和扩展输出位进行分组。利用OpBCD命令的端口编号参数指定使用哪组(哪8个输出位)。比如，端口编号 = 0 时指定输出位0～7，端口编号 = 1 时指定输出位8～15。

选择端口编号(8个输出位)之后，利用输出数据参数来定义特定输出模式。输出数据参数为1或2位的值，有效范围为0～99。第1位(10位)表示由端口编号选择的8个输出位中的高4位输出位，第2位(个位)表示由端口编号选择的8个输出位中的低4位输出位。

由于BCD的各个位的有效值在0～9范围，因此，不能满足I/O的所有组合。下表所示为端口编号=0时的I/O组合示例以及所对应的输出编号值。

输出设置(输出编号)

输出编号	7	6	5	4	3	2	1	0
01	Off	Off	Off	Off	Off	Off	Off	On
02	Off	Off	Off	Off	Off	Off	On	Off
03	Off	Off	Off	Off	Off	Off	On	On
08	Off	Off	Off	Off	On	Off	Off	Off
09	Off	Off	Off	Off	On	Off	Off	On
10	Off	Off	Off	On	Off	Off	Off	Off
11	Off	Off	Off	On	Off	Off	Off	On
99	On	Off	Off	On	On	Off	Off	On

只能用10进制数值指定BCD。因此，如果是使用BCD的OpBCD命令，则不能将所有输出位都设为ON。由于各个位的输出编号最大值均为“9”，因此，OpBCD可使用的最大值为“99”。从上表可以看出，为“99”时不能打开所有输出。输出编号值为“99”时的输出状态：0、3、4、7为ON状态，其它均为OFF状态。

注意

- OpBCD与Out的差异

  SPEL+的Out与OpBCD之间的最大差异表现在下述方面。

  - OpBCD命令用于为指定设为ON/OFF的8个输出位而使用2进制编码的10进制数(BCD)格式。由于2进制编码的10进制数格式不能使用&HA、&HB、&HC、&HD、&HE、&HF等的值，因此，不能满足8个输出位的所有组合。
  - Out命令用于对将输出设为ON/OFF的8位值使用0～255的值。(OpBCD时为0～99。)这样的话，可满足8位输出组的所有可能的组合，并根据用户的规格进行指定。

- 远程设置的输出位

  如果针对远程设置的输出位，指定OpBCD为ON，则会发生错误。远程输出位会根据系统的状态自动设为ON/OFF。有关远程的详细说明，请参阅以下手册。
  《Epson RC+ 用户指南》

  通过[设置]菜单 - [系统配置] - [控制器] - [远程控制]面板，可设置远程连接器的各个位(设为远程或I/O等)。

- 紧急停止时的输出

  机器人控制器具有发生紧急停止时将所有输出都设为OFF的功能。要将该功能设为有效或OFF，可通过[设置]菜单-[系统设置]-[控制器]-[环境设置]面板的[紧急停止时将输出端口设为OFF]复选框进行设置。

- Forced标志

  在要通过NoPause任务、NoEmgAbort任务(执行Xqt时指定NoPause或NoEmgAbort以开始的特别任务)以及后台任务，在紧急停止期间或安全门打开时将I/O输出设为ON的情况下，指定此标志。

  紧急停止期间或安全门打开时，I/O输出会发生变化，因此，在系统设计方面需要注意。

参阅

In、InBCD、MemOff、MemOn、MemSw、Off、On、Oport、Out、Sw、Wait

OpBCD使用示例

如下所示为启动名为“iotask”的主任务的程序。“iotask”是将输出位1和2设为ON之后，将输出位0和3设为ON的简单任务。如果将输出位1和2设为ON，输出位0和3则会被设为OFF。另外，前者为OFF时，后者则变为ON状态。

```
Function main
    Xqt 2, iotask
    Go P1
    .
    .
    .
Fend

Function iotask
    Do
        OpBCD 0, 6
        OpBCD 0, 9
        Wait 10
    Loop
Fend
```

如下所示为利用命令窗口的简单操作示例。

```
> OpBCD 1,6     '将输出1和2设为ON
> OpBCD 2,1     '将输出8设为ON
> OpBCD 3, 91   '将输出24、28、31设为ON
```

3.18.9 OpenDB

用于打开数据库和Excel工作簿。

格式

OpenDB #数据库编号, 数据库类型 [, SQL服务器名],数据库名

参数

数据库编号
　　指定501～508的整数。
数据库类型
　　从[SQL]、[Access]、[Excel]中选择要打开的数据库类型。
SQL服务器名
　　按数据库类型指定为[SQL]时，指定SQL服务器名。省略时，指定(LOCAL)服务器。不能指定网络上的SQL服务器。
　　按数据库类型指定[Access]或[Excel]时，不指定SQL服务器名。
数据库名

- 按数据库类型指定[SQL]时，指定SQL服务器的数据库名。
- 已指定[Access]时，指定Access文件名。已省略Access文件名路径时，检索当前文件夹。详情请参阅ChDisk。
- 已指定[Excel]时，指定Excel文件名。可指定为Excel文件的格式为Excel 2007工作薄、Excel 97-2003工作簿文件。已省略Excel文件名路径时，检索当前文件夹。详情请参阅ChDisk。

说明

以指定的文件编号打开指定的数据库。

指定的数据库必须存在已安装RC+的PC的硬盘中。如果不存在，则会发生错误。指定的文件编号用于在打开数据库期间识别该数据库，因此，在利用CloseDB命令关闭数据库之前，不能用于引用其它数据库。按数据库操作命令(SelectDB、Print#、Input#、CloseDB)使用文件编号。

不能使用Microsoft office 2010 64 bit版的Access和Excel文件。

注意

- 需要连接已安装RC+的PC。

参阅

SelectDB,CloseDB,UpdateDB,DeleteDB,Input #, Print #

OpenDB使用示例

SQL数据库的使用示例

如下所示为从使用SQL服务器2000样本数据库Northwind的表格读入数据的简单示例。

```
Integer count, i, eid
String Lastname$, Firstname$, Title$

OpenDB #501, SQL, "(LOCAL)", "Northwind"
count = SelectDB(#501, "Employees")
For i = 0 To count - 1
  Input #501, eid, Lastname$, Firstname$, Title$
  Print eid, ",", Lastname$, ",", Firstname$, ",", Title$
Next
CloseDB #501
```

Access数据库的使用示例 如下所示为从使用Microsoft Access 2007样本数据库学生名册的表格读入数据的简单示例。

```
Integer count, i, eid
String Lastname$, Firstname$, dummy$

OpenDB #502, Access, "c:\MyDataBase\学生名册.accdb"
count = SelectDB(#502, "学生")
For i = 0 To count - 1
  Input #502, eid, dummy$, dummy$, Lastname$, dummy$, Firstname$
  Print eid, ",", Lastname$, ",", Firstname$
Next
CloseDB #502
```

Excel工作簿的使用示例 如下所示为从使用Microsoft Excel工作簿学生名册的表单读入数据的简单示例。

```
Integer count, i, eid
String Lastname$, Firstname$

OpenDB #503, Excel, "c:\MyDataBase\学生名册.xls"
count = SelectDB(#503, "[学生$]")
For i = 0 To count - 1
  Input #503, eid, Lastname$, Firstname$
  Print eid, ",", Lastname$, ",", Firstname$
Next
CloseDB #503
```

3.18.10 OpenCom

用于打开RS-232C端口。

格式

OpenCom #端口编号

参数

端口编号
以整数值指定要打开的RS-232C的端口编号。
SPEL+ 控制部分
1～8
PC部分
1001～1008

说明

将RS-232C端口设为可用于任务。

要使用SPEL+控制部分的端口时，需要在控制器中安装I/O选件卡。

要使用PC部分的端口时，需要设置RC+。详情请参阅以下手册中有关RC-232C的记述。
《Epson RC+ 用户指南 - [设置]菜单》

参阅

ChkCom、CloseCom、SetCom

OpenCom使用示例

```
Integer PortNo

PortNo = 1001
OpenCom #PortNo
Print #PortNo, "Data from COM1"
CloseCom #PortNo
```

3.18.11 OpenCom函数

用于获取实施OpenCom的任务编号。

格式

OpenCom (端口编号)

参数

端口编号
　　以整数值指定RS-232C的端口编号。
SPEL+ 控制部分
　　1～8
PC部分
　　1001～1008

说明

用于获取实施OpenCom的任务编号。

参阅

ChkCom、CloseCom、OpenCom、SetCom

OpenCom函数使用示例

```
Print OpenCom(PortNo)
```

3.18.12 OpenNet

用于打开TCP/IP网络端口。

格式

OpenNet #端口编号 As { Client | Server }

参数

端口编号

指定要打开的TCP/IP端口编号的整数值。端口编号的范围为201～216。

说明

OpenNet用于打开TCP/IP端口，以便与网络上的其它电脑进行通信。

1个系统作为服务器打开，其它系统作为客户端打开。先启动哪个都可以。

参阅

ChkNet、CloseNet、SetNet

OpenNet使用示例

下例所示为2个控制器的TCP/IP设置。

Controller #1:
Port: #201
Host Name: 192.168.0.2
TCP/IP Port: 1000

```
Function tcpip
  OpenNet #201 As Server
  WaitNet #201
  Print #201, "Data from host 1"
Fend
```

Controller #2:
Port: #201
Host Name: 192.168.0.1
TCP/IP Port: 1000

```
Function tcpip
  String data$
  OpenNet #201 As Client
  WaitNet #201
  Input #201, data$
  Print "received '", data$, "' from host 1"
Fend
```

3.18.13 OpenNet函数

用于获取实施OpenNet的任务编号。

格式

OpenNet (端口编号)

参数

端口编号
　　指定TCP/IP端口编号的整数值。端口编号的范围为201～216。

说明

用于获取实施OpenNet的任务编号。

参阅

ChkNet、CloseNet、OpenNet、SetNet

OpenNet函数使用示例

```
Print OpenNet(PortNo)
```

3.18.14 Oport函数

用作返回指定输出位状态的函数。

格式
Oport (输出位编号)

参数

输出位编号
　　指定I/O的输出位。

返回值
以0或1的整数返回I/O的指定输出位状态。

- 0：关闭
- 1：打开

说明
Oport用于检查输出位的状态。与针对输入位的Sw命令的功能非常相似。进料器、传送带和夹爪经常用到Oport。另外，也用于检查通过I/O发挥作用的、与其它装置主机等连接的输出位的状态。以1或0的值返回由Oport函数检查的状态。这些值表示指定的输出位处于ON还是OFF状态。

注意

- Oport与Sw的差异

  Oport与Sw命令存在明显差异。两者都用于检查I/O状态，但检查的I/O类型不同。Sw命令用于检查输入位。Oport命令用于检查标准及扩展硬件的输出位。这些硬件端口用于与控制器外部装置进行相互通信的独立输出位。

参阅
In、InBCD、MemIn、MemOff、MemOn、MemOut、MemSw、Off、On、OpBCD、Out、Sw、Wait

Oport函数使用示例
如下所示为将输出位5设为ON，并在检查其是否变为ON状态之后继续执行程序的示例。

```
Function main
    TMOut 10
    OnErr errchk
    Integer errnum
    On 5    '将输出5设为ON
    Wait Oport(5)
    Call mkpart1
    Exit Function

errchk:
    errnum = Err(0)
    If errnum = 94 Then
       Print "TIME Out Error Occurred during period"
       Print "waiting for Oport to come on. Check"
       Print "Output #5 for proper operation. Then"
       Print "restart this program."
    Else
       Print "ERROR number ", errnum, "Occurred"
       Print "Program stopped due to errors!"
    EndIf
    Exit Function
Fend
```

如下所示为利用命令窗口的简单操作示例。

```
> On 1
> Print Oport(1)
1
> Off 1
> Print Oport(1)
0
>
```

3.18.15 Or运算符

以位为单位或以逻辑对2个值进行Or运算。

格式
值1 Or 值2

参数

值1
　　指定整数值或Boolean值。
值2
　　指定整数值或Boolean值。

结果
以整数值指定时，返回以位为单位对值进行Or运算的结果。以Boolean值指定时，返回逻辑Or运算的结果。

说明
在整数值情况下，Or运算符以位为单位对操作数进行Or运算。运算结果是对2个操作数的各个位进行Or处理的值。

为Boolean值时，如果值的某一方为真(True)，运算结果也为真(True)。

参阅
And、LShift、Mod、Not、RShift、Xor

Or运算符使用示例
下例所示为位单位Or运算符的状况。

```
>print 1 Or 2
 3
>
```

下例所示为逻辑Or运算符的状况。

```
If a = 1 Or b = 2 Then
c = 3
EndIf
```

3.18.16 Out

用于同时设置(输出)8个输出位。

格式

Out 端口编号, 输出数据 [, Forced]

参数

端口编号

指定I/O的输出字节。按如下所述，指定数值对应各自的输出位。

端口编号	输出位
0	0-7
1	8-15
...	...

输出数据

以0～255的整数值指定由端口编号指定的输出组的输出模式。以16进制数显示时，范围为&H0～&HFF。低数位表示低位(或最初的4个输出位)，高数位表示高位(或后续的4个输出位)。

Forced

可省略。通常会省略。

说明

Out用于通过组合端口编号与输出数据，同时设置8个输出位。利用端口编号参数指定使用哪组(哪8个输出位)。比如，端口编号 = 0 时指定输出位0～7。端口编号 = 1 时指定输出位8～15。

首先，利用端口编号指定8个输出位之后，利用输出数据参数来指定特定的输出模式。输出数据参数可获取的值为0～255，以16进制数或10进制数的整数进行指定。(&H0～&HFF或0～255)

下表所示为部分I/O组合示例，以及端口编号为“0”和“1”时分别对应的输出数据值。

端口编号=0时的输出设置(输出位编号)

输出数据值	7	6	5	4	3	2	1	0
01	Off	Off	Off	Off	Off	Off	Off	On
02	Off	Off	Off	Off	Off	Off	On	Off
03	Off	Off	Off	Off	Off	Off	On	On
08	Off	Off	Off	Off	On	Off	Off	Off
09	Off	Off	Off	Off	On	Off	Off	On
10	Off	Off	Off	Off	On	Off	On	Off
11	Off	Off	Off	Off	On	Off	On	On
99	Off	On	On	Off	Off	Off	On	On
255	On	On	On	On	On	On	On	On

端口编号=1时的输出设置(输出位编号)

输出数据值	15	14	13	12	11	10	9	8
01	Off	Off	Off	Off	Off	Off	Off	On
02	Off	Off	Off	Off	Off	Off	On	Off

输出数据值	15	14	13	12	11	10	9	8
03	Off	Off	Off	Off	Off	Off	On	On
08	Off	Off	Off	Off	On	Off	Off	Off
09	Off	Off	Off	Off	On	Off	Off	On
10	Off	Off	Off	Off	On	Off	On	Off
11	Off	Off	Off	Off	On	Off	On	On
99	Off	On	On	Off	Off	Off	On	On
255	On	On	On	On	On	On	On	On

注意

- OpBCD与Out的差异

  SPEL+的Out与OpBCD之间的最大差异表现在下述方面。

  - OpBCD命令用于为指定设为ON/OFF的8个输出位而使用2进制编码的10进制数(BCD)格式。由于2进制编码的10进制数格式不能使用&HA、&HB、&HC、&HD、&HE、&HF等的值，因此，不能满足8个输出位的所有组合。
  - Out命令用于对将输出设为ON/OFF的8位值使用0～255的值(OpBCD时，为0～99)。这样的话，可满足8位输出组的所有可能的组合，并根据用户的规格进行指定。

- Forced标志

  在要通过NoPause任务、NoEmgAbort任务(执行Xqt时指定NoPause或NoEmgAbort以开始的特别任务)以及后台任务，在紧急停止期间或安全门打开时将I/O输出设为ON的情况下，指定此标志。

  紧急停止期间或安全门打开时，I/O输出会发生变化，因此，在系统设计方面需要注意。

参阅

In、InBCD、MemOff、MemOn、MemOut、MemSw、Off、On、Oport、Sw、Wait

Out使用示例

如下所示为启动名为“iotask”的主任务的程序。“iotask”是将输出位的0～3设为ON/OFF的简单任务。如果使用Out命令，则可利用1个命令进行上述操作，而不必单独将输出位设为ON/OFF。

```
Function main

  Xqt iotask
  Do
    Go P1
    Go P2
  Loop
Fend

Function iotask

  Do
    Out 0, &H0F
    Out 0, &H00
    Wait 10
  Loop
Fend
```

如下所示为利用命令窗口的简单操作示例。

```
> Out 1,6     '将输出9和10设为ON
> Out 2,1     '将输出8设为ON
> Out 3,91    '将输出24、25、27、28、30设为ON
```

3.18.17 Out函数

用于以字节为单位返回输出端口状态。

格式

Out (端口编号)

参数

端口编号

指定I/O的输出字节。按如下所述，指定数值对应各自的输出位。

端口编号	输出位
0	0-7
1	8-15
...	...

返回值

以字节为单位返回指定输出端口的状态。

参阅

Out

Out函数使用示例

```
Print Out(0)
```

3.18.18 OutReal

用于将实值输出数据作为32位浮点数据(符合IEEE754标准)设置输出端口2个字(32位)的状态。

格式

OutReal 字端口编号, 输出数据 [, Forced]

参数

字端口编号
　　指定I/O的输出字。
输出数据
　　以表达式或数值指定输出数据(Real型实数)。
Forced
　　可省略。通常会省略。

说明

将指定IEEE754的Real值输出到由字端口编号指定的输出字端口和后续输出字端口中。可在字端口编号参数中使用输出字标签。

注意

- Forced标志

  在要通过NoPause任务、NoEmgAbort任务(执行Xqt时指定NoPause或NoEmgAbort以开始的特别任务)，在紧急停止期间或安全门打开时将I/O输出设为ON的情况下，指定此标志。

  紧急停止期间或安全门打开时，I/O输出会发生变化，因此，在系统设计方面需要注意。

参阅

In、InW、InBCD、InReal、Out、OutW、OpBCD、OutReal函数

OutReal使用示例

```
OutReal 32, 2.543
```

3.18.19 OutReal函数

用于以32位浮点数据(符合IEEE754标准)获取输出端口的状态。

格式

OutReal (字端口编号)

参数

字端口编号
　　指定I/O的输出字。

返回值

以32位浮点数据(符合IEEE754标准)返回指定输出端口的状态。

参阅

In、InW、InBCD、InReal、Out、OutW、OpBCD、OutReal

OutReal函数使用示例

```
Real rdata01

rdata01 = OutReal(0)
```

3.18.20 OutW

用于以16位并按字单位设置输出端口的状态。

格式

OutW 字端口编号, 输出数据 [, Forced]

参数

字端口编号
　　指定I/O的输出字。
输出数据
　　利用表达式或数值指定输出数据(0～65535的整数)。
Forced
　　可省略。通常会省略。

说明

将由参数字端口编号指定的用户I/O输出端口组的状态变更为指定的输出数据。

注意

- Forced标志

  在要通过NoPause任务、NoEmgAbort任务(执行Xqt时指定NoPause或NoEmgAbort以开始的特别任务)以及后台任务，在紧急停止期间或安全门打开时将I/O输出设为ON的情况下，指定此标志。

  紧急停止期间或安全门打开时，I/O输出会发生变化，因此，在系统设计方面需要注意。

参阅

In、InW、Out

OutW使用示例

```
OutW 0, 25
```

3.18.21 OutW函数

用于以字为单位(双字节)返回输出端口状态。

格式

OutW (字端口编号)

参数

字端口编号
　　指定I/O的输出字。

返回值

以16位返回指定输出端口的状态。

参阅

OutW

OutW函数使用示例

```
OutW 0, &H1010
```

3.19 P

3.19.1 P#(1. 点定义)

用于定义点数据。

格式

P点编号 = 指定点

点标签 = 指定点

参数

点编号

以数值或带括号的表达式指定。

- P编号
- P(表达式)

点标签

指定点标签。

点指定

以点数据指定下述点数据。

P点编号、点标签、Here、Pallet、点数据函数(Here函数、XY函数、JA函数、Pulse函数等)

详情请参阅“P#(2. 指定点)”。

说明

通过代入其它点或点表达式来定义点数据。

参阅

Local、Pallet、PDef、PDel、Plist

点的代入示例

如下所示为利用命令窗口的操作示例。

在P1位置设置坐标数据。

```
> P1 = XY(300,200,-50,100)
```

指定左机械臂姿势：

```
> P2 = XY(-400,200,-80,100)/L
```

在P2的X坐标数据中加上20，并将其值定义为P3。

```
> P3 = P2 +X(20)
> plist 3
P3 = XY(-380,200,-80,100)/L
```

从P2的Y坐标数据中减去50，将Z坐标数据设为 -30，然后作为右机械臂的姿势，将其值定义为P4。

```
>P4=P2 -Y(50) :Z(-30) /R
> plist 4
P4 = XY(-450,200,-30,100)/R
```

在Pallet(3, 5)的U coord(U坐标值)中加上90，定义为P6。

```
> P5 = Here
> P6 = pallet(3,5) +U(90)
```

3.19.2 P#(2. 指定点)

指定用于指定点和动作命令的点数据。

格式

point [{ +| - } 指定点] [本地编号] [末端夹具(手腕系)姿势] [肘姿势] [手腕姿势] [j4flag值] [j6flag值] [j1flag值] [j2flag值] [相对偏移值] [绝对坐标]

参数

点指定
以下述某项指定点数据。
- P编号
- P(表达式)
- 点标签
- Pallet(托盘编号、分区编号)
- 指定当前位置
- Here
- 点数据函数
- XY函数、JA函数、Pulse函数等

本地编号
指定本地编号(1～15)。请务必在编号前附加斜杠“/”或“@”符号。“/”表示坐标为本地坐标。“@”表示坐标已被转换为本地坐标。可省略。

末端夹具(手腕系)姿势
是指水平多关节型(包括RS系列)和垂直6轴型(包括N系列)机器人的末端夹具(手腕系)姿势。指定/L(左腕姿势)或/R(右腕姿势)。可省略。

肘姿势
是使用垂直6轴型机器人(包括N系列)时需要的参数。指定/A(ABOVE)或/B(BELOW)。

手腕姿势
是使用垂直6轴型机器人(包括N系列)时需要的参数。指定/F(FLIP)或/NF(NOFLIP)。

j4flag值
是使用垂直6轴型机器人(包括N系列)时需要的参数。指定/J4F0或/J4F1。

j6flag值
是使用垂直6轴型机器人(包括N系列)时需要的参数。指定/J6F0～/J6F127的值。

j1flag值
是使用RS系列与垂直6轴型机器人(不包括N系列)时需要的参数。指定/J1F0或/J1F1。

j2flag值
是使用RS系列时需要的参数。指定/J2F0或/J2F1。

j1angle值
是使用RS系列与N系列时需要的参数。指定/J1A(实值)。

j4angle值
是使用N系列需要的参数。指定J4A(实值)。

相对偏移值
调整1个或1个以上的相对坐标。可省略。
{+ | -} {X | Y | Z | U | V | W | RZ | RY | RX | R | S | T | ST } (表达式)
TL偏移为当前工具坐标系的相对偏移。
{+ | -} {TLX | TLY | TLZ | TLU | TLV | TLW} (表达式)

绝对坐标
指定1个或1个以上的绝对坐标。请务必在坐标前附加冒号“:”。可省略该绝对坐标。
:{X | Y | Z | U | V | W|R | S | T | ST } (表达式)

说明

点代入或动作命令用到点表达式。(请参阅下例)

```
Go P1 + P2
P1 = P2 + XY(100, 100, 0, 0)
```

相对偏移值的使用
可进行相对基础点的1个或1个以上的坐标偏移。下例所示为从当前位置将机器人向X轴正方向移动20 mm。

```
Go Here +X(20)
```

如果再次执行，则变为相对移动，机器人再向X轴方向移动20 mm。

为垂直6轴型机器人(包括N系列)时，如果要进行坐标轴周围的相对旋转，则按如下所示进行指定。下例所示为以当前姿势为基准，将机器人的工具姿势向X轴正方向旋转20° 的情况。

```
Go Here +RX (20)
```

可进行相对工具偏移。

```
Go Here +TLX(20) -TLY(5.5)
```

为垂直6轴型机器人(包括N系列)时，要动作到通过相对偏移获得的点时，手腕部分可能会转向意想不到的方向。这是因为相对偏移运算包含不取决于机器人机型的命令，直接进行动作而未转换必要的姿势标志的缘故。LJM函数可用于防止手腕部分进行这种意想不到的旋转。

```
Go LJM(Here +X(20))
```

绝对坐标的使用
可使用绝对坐标变更基础点的1个或1个以上的坐标。下例所示为将机器人移动到X轴的20 mm位置。

```
Go Here :X(20)
```

即使再次执行，机械臂也会在此前移动中指定的绝对坐标位置，因此，不会移动。

在暂时修正点数据时，使用相对偏移值和绝对坐标是非常便利的。下例所示为通过指定Z的相对偏移10 mm，移动到拾取位置上方10 mm的地方，然后缓慢移动到拾取位置。

```
Speed fast
Jump pick +Z(10)
Speed slow
Go pick
```

下例所示为通过指定第3关节的绝对值“0”，使其从当前位置垂直向上移动的情况。

```
LimZ 0
Jump Here :Z(0)
```

本地坐标系的使用
可使用“/”或“@”符号指定本地编号。“/”与“@”符号的功能各不相同。

在本地坐标系中指定坐标时，使用“/”。在下述语句示例中，通过附加“/1”，将“P1的位置”指定为本地1的0,0,0,0”。

```
P1 = XY(0, 0, 0, 0) /1
```

要将坐标转换为本地坐标时，使用“@”。在下述语句示例中，将表示当前位置的本地坐标值注册到P1中。

```
P1 = Here @1
```

参阅
Go、LJM、Local、Pallet、Pdel、Plist、Hand、Elbow、Wrist、J4Flag、J6Flag、J1Flag、J2Flag

指定点使用示例
如下所示为使用代入语句或动作命令的指定点示例。

```
P1 = XY(300,200,-50,100)
P2 = P1 /R
P3 = pick /1
P4 = P5 + P6
P(i) = XY(100, 200, CZ(P100), 0)
Go P1 -X(20) :Z(-20) /R
Go Pallet(1, 1) -Y(25.5)
Move pick /R
Jump Here :Z(0)
Go Here :Z(-25.5)
Go JA(25, 0, -20, 180)
pick = XY(100, 100, -50, 0)

P1 = XY(300,200, -50,100, -90, 0)
P2 = P1 /F /B
P2 = P1 +TLV(25)
```

3.19.3 PAgl函数

用于根据指定的坐标值计算并返回关节位置。

格式

PAgl (坐标值, 关节编号)

参数

坐标值
　　指定计算关节位置的坐标值。
关节编号
　　以表达式或数值指定关节编号(1～9的整数值)。附加轴的S轴为8，T轴为9。

返回值

以实值返回计算的关节位置。旋转关节(单位：deg)、移动关节(单位：mm)

参阅

Agl、CX、CY、CZ、CU、CV、CW、CR、CS、CT、PPls

PAgl函数使用示例

```
Real joint1

joint1 = PAgl(P10, 1)
```

3.19.4 Pallet

用于定义托盘和显示定义托盘。

格式

(1) Pallet [Outside,] [托盘编号, 点编号1, 点编号2, 点编号3 [, 点编号4], 分区数1, 分区数2]

(2) Pallet [Outside,] 托盘编号, 坐标系数据1, 坐标系数据2, 坐标系数据3 [, 坐标系数据4], 分区数1, 分区数2

(3) Pallet

参数

Outside
　　在指定的行和列的范围以外生成可接近的托盘。可省略。
托盘编号
　　以表达式或数值指定托盘编号(0～15的整数)。
点编号1～3
　　指定用于托盘定义(标准的3点定义)的点变量。
点编号4
　　进行4点定义时，与点编号1～3同时使用。可省略。
分区数1
　　以整数指定托盘的点编号1(坐标系数据1)和点编号2(坐标系数据2)的分区数。范围为1～32767。(分区数1×分区数2 < 32767)
分区数2
　　以整数指定托盘的点编号1(坐标系数据1)和点编号3(坐标系数据3)的分区数。范围为1～32767。(分区数1×分区数2 < 32767)
坐标系数据1~3
　　直接以点数据指定用于托盘定义(标准的3点定义)的坐标系。
坐标系数据4
　　进行4点定义时，与坐标系数据1～3同时使用。可省略。

结果

(3)如果省略参数，则显示所有定义的托盘。

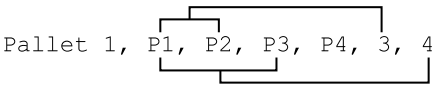
说明

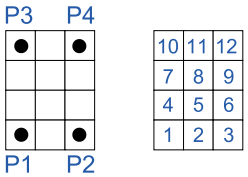
在机器人上至少对点编号1(坐标系数据1)、点编号2(坐标系数据2)、点编号3(坐标系数据3)进行示教，并指定点编号1(坐标系数据1)和点编号2(坐标系数据2)的分区数以及点编号1(坐标系数据1)和点编号3(坐标系数据3)的分区数，定义托盘。

如果是精度较高的方形托盘，只需指定边角4点中的3点位置就足够了，在大多数情况下，建议指定全部边角4点的位置来定义托盘。

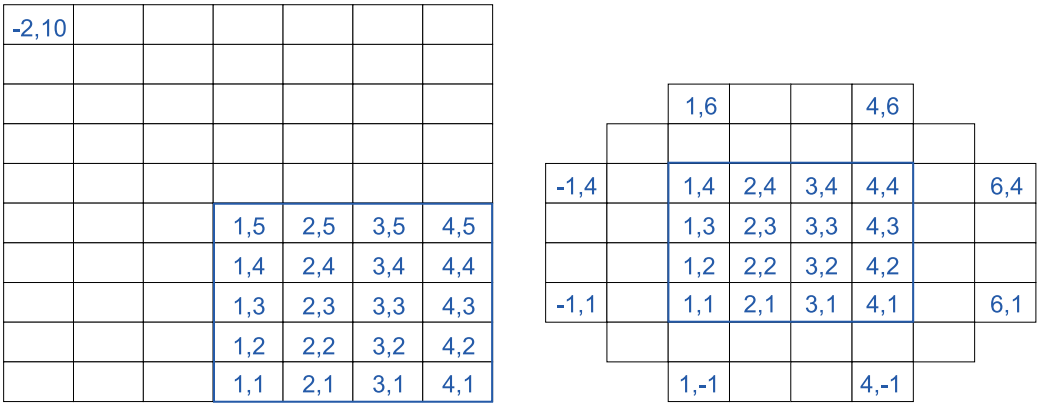
定义托盘时，首先对边角的3点或4点进行示教，具体如下所示。

4点定义时：下面所示为P1、P2、P3和P4。P1～P2之间有3点，P1～P3之间有4点，使用共计12点按下述格式进行定义。





10	11	12
7	8	9
4	5	6
1	2	3



表示托盘分区的各个点自动分配分区编号。为上图所示情况时，从P1开始。Pallet函数也要用到该分区编号。

可利用Outside指定在行和列的范围以外生成可接近的托盘。

例：

```
Pallet Outside 1, P1, P2, P3, 4, 5
Jump Pallet(1, -2, 10)
```

注意

- 托盘尺寸的上限

  托盘定义使用的点数必须小于32767。(分区数1×分区数2 < 32767)

- 错误的托盘形状定义

  如果弄错点的顺序或点间的分区数，则会错误地定义托盘形状。

- 托盘面的定义

  由边角3点的Z坐标值定义托盘平面的高度。而且也可以定义竖放托盘。

- 单列托盘的托盘定义

  也可以利用3点指定的Pallet命令来定义单列托盘。单列时，对两端的2点进行示教，并按如下所示进行输入和执行。同一编号方向的分区数为1。

  ```
  > Pallet 2, P20, P21, P20, 5, 1     '定义5×1托盘
  ```

- UVW坐标值

  Pallet命令中指定的3点(4点)UVW坐标值不同时，使用点编号1与坐标系数据1的UVW坐标值。

点编号2～4与坐标系数据2～4的UVW坐标值会被无视。

- 附加轴坐标值

  Pallet命令指定的3点(4点)坐标值保持附加轴坐标值(ST轴值)时，即使是附加轴坐标值也进行均等分配。也就是说，由于将附加轴用作移动轴时，托盘定义也要考虑移动轴动作来进行计算，因此，可定义大于顾及移动轴位置的机器人动作范围的较大托盘。相反，即使定义对托盘定义没有影响的附加轴，托盘定义时也需要注意附加轴位置。

参阅

Pallet函数

Pallet使用示例

如下所示为通过命令窗口设置由P1、P2、P3定义的托盘的示例。托盘面上均等配置15点，托盘点编号1～3排列在P1～P2之间。

```
> pallet 1, P1, P2, P3, 3, 5
> jump pallet(1, 2)          '跳跃到托盘的指定位置
```

如下所示为由该设置的结果生成的托盘。

P3

13	14	15
10	11	12
7	8	9
4	5	6
1	2	3

P1 P2

3.19.5 Pallet函数

用作引用托盘上位置的点数据函数。

格式

(1) Pallet (托盘编号, 托盘位置编号)

(2) Pallet (托盘编号, 分区横坐标, 分区竖坐标)

参数

托盘编号
　　以数值指定托盘编号(0～15的整数)。
托盘位置编号
　　以表达式或数值(1～32767)指定分区点的指定编号(整数)。
分区横坐标
　　以数值(-32768～32767)指定由托盘定义指定的横坐标。
分区纵坐标
　　以数值(-32768～32767)指定由托盘定义指定的竖坐标。

说明

Pallet用于返回事先由Pallet语句定义的托盘上1点位置。通过同时使用该函数与动作命令(Go或Jump命名等)，可将机械臂移动到托盘上的指定位置。

可利用表达式或整数值定义托盘位置编号。

注意

- 垂直6轴型机器人(包括N系列)的托盘动作

  对于垂直6轴型机器人(包括N系列)来说，要动作到托盘或通过相对偏移等的点运算获得的点时，手腕部分可能会转向意想不到的方向。这是因为上述点运算包含不取决于机器人机型的命令，直接进行动作而未转换必要的姿势标志的缘故。

  LJM函数可用于防止手腕部分进行这种意想不到的旋转。

- RS系列的托盘动作

  同样，就RS系列而言，要动作到托盘或通过相对偏移等的点运算获得的点时，第1机械臂可能会转向意想不到的方向。为了防止第1机械臂进行这种意想不到的旋转，LJM函数用于适当地转换点数据的姿势标志。

  另外，在RS系列中，如果U轴在转换了姿势标志却要进行超出动作范围的动作，则可能会发生错误。为了防止U轴发生这种超出动作范围的错误，LJM函数用于将U轴的目标角度修正为动作范围内的目标角度。可通过在选择姿势标志中指定2的方式加以运用。

- UVW坐标值

  Pallet命令中指定的3点(4点)UVW坐标值不同时，使用点编号1与坐标系数据1的UVW坐标值。

  点编号2～4与坐标系数据2～4的UVW坐标值会被无视。

- 附加轴坐标值

  Pallet命令指定的3点(4点)坐标值保持附加轴坐标值(ST轴值)时，即使是附加轴坐标值也进行均等分配。也就是说，由于将附加轴用作移动轴时，托盘定义也要考虑移动轴动作来进行计算，因此，可定义大于顾及移动轴位置的机器人动作范围的较大托盘。相反，即使定义对托盘定义没有影响的附加轴，托盘定义时也需要注意附加轴位置。

参阅

LJM、Pallet

Pallet函数使用示例

如下所示为将工件从托盘1移动到托盘2的程序示例。

```
Function main
  Integer index
  Pallet 1, P1, P2, P3, 3, 5      '定义托盘1
  Pallet 2, P12, P13, P11, 5, 3   '定义托盘2
  For index = 1 To 15
    Jump Pallet(1, index)         '移动到托盘1的点索引处
    On 1                          '拾取工件
    Wait 0.5
    Jump Pallet(2, index)         '移动到托盘2的点索引处
    Off 1                         '释放工件
    Wait 0.5
  Next I
Fend

Function main
  Integer i, j

  P0 = XY(300, 300, 300, 90, 0, 180)
  P1 = XY(200, 280, 150, 90, 0, 180)
  P2 = XY(200, 330, 150, 90, 0, 180)
  P3 = XY(-200, 280, 150, 90, 0, 180)

  Pallet 1, P1, P2, P3, 10, 10

  Motor On
  Power High
  Speed 50; Accel 50, 50
  SpeedS 1000; AccelS 5000

  Go P0
  P11 = P0 -TLZ(50)

  For i = 1 To 10
      For j = 1 To 10
        '点的指定
        P10 = P11    '转移点
      P12 = Pallet(1, i, j)     '目标点
      P11 = P12 -TLZ(50)     '接近起点
      '各点的LJM转换
      P10 = LJM(P10)
      P11 = LJM(P11, P10)
      P12 = LJM(P12, P11)
      '执行动作
        Jump3 P10, P11, P12 C0
    Next
  Next
Fend

Function main2
  P0 = XY(300, 300, 300, 90, 0, 180)
  P1 = XY(400, 0, 150, 90, 0, 180)
  P2 = XY(400, 500, 150, 90, 0, 180)
  P3 = XY(-400, 0, 150, 90, 0, 180)
  Pallet 1, P1, P2, P3, 10, 10

  Motor On
  Power High
  Speed 50; Accel 50, 50
```

```
  SpeedS 1000; AccelS 5000

  Go P0

  Do
    '点的指定
    P10 = Here -TLZ(50)     '转移点
    P12 = Pallet(1, Int(Rnd(9)) + 1, Int(Rnd(9)) + 1)     '目标点
    P11 = P12 -TLZ(50)     '接近起点

    If TargetOK(P11) And TargetOK(P12) Then     '点的检查
    '各点的LJM转换
    P10 = LJM(P10)
    P11 = LJM(P11, P10)
    P12 = LJM(P12, P11)
    '执行动作
      Jump3 P10, P11, P12 C0
    EndIf
   Loop
Fend
```

3.19.6 PalletClr

用于清除已设置的Pallet。

格式

PalletClr 托盘编号

参数

托盘编号

以数值指定要清除设置的托盘编号(0～15的整数)。

参阅

Pallet

PalletClr使用示例

```
PalletClr 1
```

3.19.7 ParseStr/ParseStr函数

用于分析字符串并返回令牌数组。

格式

ParseStr 字符串$, 令牌$(), 分隔字符$

令牌数 = ParseStr (输入字符串$, 令牌$(), 分隔字符$)

参数

字符串$
　　指定要分析的字符串。
令牌$()
　　指定保存令牌的数组。不能指定已进行ByRef声明的数组。
分隔字符$
　　指定1个字符以上的分隔字符。

返回值

用作函数时，返回已分析的令牌数。

参阅

Redim、String

ParseStr/ParseStr函数使用示例

```
String toks$(0)
Integer i

ParseStr "1 2 3 4", toks$(), " "

For i = 0 To UBound(toks$)
   Print "token ", i, " = ", toks$(i)
Next i
```

3.19.8 Pass

用于进行穿过指定点附近(不停止)的PTP动作。

格式

Pass 指定点 [, {On | Off | MemOn | MemOff} 位编号 [, 指定点...]] [LJM [选择姿势标志]]

参数

点指定
　　以P编号、P(表达式)、点标签进行指定。
　　点数据没有遗漏并按升序或降序排列时，可用冒号连接2个点编号进行指定，比如P(1:5)。
位编号
　　指定要设为ON/OFF的I/O输出位或存储器I/O位。以整数或输出标签进行指定。
LJM
　　利用LJM函数转换转移坐标、接近坐标、目标坐标。可省略。
选择姿势标志
　　指定赋予LJM函数的姿势标志选择参数。可省略。

说明

Pass用于移动机械臂并使其穿过指定点数据(位置)附近。不穿过指定点数据(位置)自身。

指定点数据时，使用(P0,P1, ...)。请利用逗号对各点之间进行分隔。

要在动作执行期间将输出位设为ON/OFF时，请利用逗号分隔各点，然后插入On/Off命令。机械臂到达即将插入On/Off的点之前时，执行On/Off。

如果Pass后接上另一个Pass语句，则将控制移交给后续的Pass，而不在最初Pass的最终指定点附近停止。

如果Pass后接上Pass以外的动作命令，机器人则停在Pass语句的最终指定点上，不进行Fine定位。

如果Pass后接上动作命令以外的命令、语句、函数等，则在机械臂到达由Pass指定的最终点之前，执行这些命令、语句或函数等。

要求通过Fine精确地定位到目标位置时，请按下例所示，指定目标位置并在Pass之后置入Go。

```
Pass P5; Go P5; On 1; Move P10
```

加速值或减速值越大，机械臂穿过时就越靠近指定点。可通过使用Pass命令来避开障碍物。

如果使用LJM参数，则可简化使用LJM函数的程序。

比如，可将

```
P11 = LJM(P1, Here, 1)
P12 = LJM(P2, P11, 1)
P13 = LJM(P3, P12, 1)
Pass P11, P12, P13
```

这样的4行程序替换为下述1行程序：

```
  Pass P1, P2, P3 LJM 1
```

可以转换为1行程序。

在垂直6轴型机器人(包括N系列)和RS系列机器人中，LJM参数是有效的。

以默认值使用姿势标志选择时，可省略。

```
  Pass P1, P2, P3 LJM
```

参阅

Accel、Go、Jump、Speed

Pass使用示例

如下所示为使用Pass命令操作机械臂的示例。

```
Function main
  Jump P1
  Pass P2    '使机械臂靠近P2并在到达P2之前执行后续命令
  On 2
  Pass P3
  Pass P4
  Off 0
  Pass P5
Fend
```

3.19.9 Pause

用于暂停可暂停的所有任务。

格式
Pause

说明
如果执行Pause，则暂停可暂停的所有任务(未指定利用Xqt命令的NoPause或NoEmgAbort的任务)。正在执行动作命令的任务全部暂停。

后台任务不会因Pause而暂停。

注意

- QP对Pause的影响

  QP命令用于设置在执行Pause时，立即停止机器人动作或在动作结束之后暂停程序。详情请参阅帮助的QP命令。

Pause使用示例
下例所示为利用Pause暂停任务。在Pause行程序暂停。按下操作员窗口中的[继续]按钮，重新开始执行任务。

```
Function main

  Xqt monitor
  Go P1
  On 1
  Jump P2
  Off 1
  Pause             '暂停程序
  Go P40
  Jump P50
Fend
```

3.19.10 PauseOn函数

用于返回暂停状态(Pause状态)。

格式
PauseOn

返回值
如果处于暂停状态，则返回“True”；如果不是，则返回“False”。

说明
本函数仅用于NoPause任务、NoEmgAbort任务(执行Xqt时指定NoPause或NoEmgAbort以开始的特别任务)与后台任务。

参阅
ErrorOn、EstopOn、SafetyOn、Wait、Xqt

PauseOn函数使用示例
下例所示为监视控制器暂停状态，并在发生暂停时对I/O进行ON/OFF操作的程序。因打开安全门而进入暂停状态时，不对I/O进行On/Off操作。

```
Function main

  Xqt PauseMonitor, NoPause
  :
  :
Fend

Function PauseMonitor
  Boolean IsPause
  IsPause = False
  Do
    Wait 0.1
    If SafetyOn = True  Then
      If IsPause = False Then
        Print "Safety On"
        IsPause = True
      EndIf
    ElseIf PauseOn = True  Then
      If IsPause = False Then
        Print "InPause"
        If SafetyOn = False Then
          Off 10
          On 12
        EndIf
      IsPause = True
      EndIf
    Else
      If IsPause = True Then
        Print "OutPause"
        On 10
        Off 12
        IsPause = False
      EndIf
    EndIf
  Loop
Fend
```

3.19.11 PDef函数

用于返回指定的点数据是否定义。

格式

PDef (点数据)

参数

点数据

以整数值、P编号、P(表达式)、点标签进行指定。有关兼容性的注意事项不能将点数据自变量指定为变量。要使用变量时，请记述为Pdef(P(varName)。

返回值

如果已定义点文件，则返回“True”；如果不是，则返回“False”。

参阅

Here、Pdel

PDef函数使用示例

```
If Not PDef(1) Then
    Here P1
Endif
Integer i
For     i = 0 to 10
    If PDef (P(i)) Then
        Print "P(";i;") is defined"
    EndIf
Next
```

3.19.12 PDel

用于删除指定的点数据。

格式

PDel 起点编号 [, 终点编号]

参数

起点编号
　　以整数值或表达式指定要删除的点数据范围的开始编号。
终点编号
　　以整数值或表达式指定要删除的点数据范围的结束编号。

说明

用于从机器人的存储器上删除指定的点数据。删除参数的起点编号～终点编号之间的点数据。请将起点编号设为小于终点编号的值。

PDel使用示例

```
> p1=10,300,-10,0/L
> p2=0,300,-40,0
> p10=-50,350,0,0
> pdel 1,2        '删除点1和2
> plist
P10  =   -50.000,  350.000,    0.000,    0.000 /R /0
> pdel 50         '删除点50
> pdel 100,200    '删除点100～200
>
```

3.19.13 PDescription

用于设置指定点数据的注释。

格式

PDescription 点数据,新注释

参数

点数据
以整数值、P编号、P(表达式)、点标签进行指定。无法在点数据自变量中指定变量。如要使用变量，请记述PDescription (P(varName))和“新注释”。

新注释
对已指定的点数据的注释进行指定的字符串。

说明

PDescription用于将注释保存到控制器存储器中的指定点数据中。

如果创建项目或执行程序，将从存储器中删除保存到控制器存储器中的注释。需要保存时，请执行“SavePoints”，保存为点文件。

参阅

PDef函数、PDescription$函数、PLabel、PLabel$函数

PDescription使用示例

```
PDescription 1, "Comment"
```

3.19.14 PDescription$函数

用于返回指定点编号中定义的点的注释。

格式

PDescription$ (点数据)

参数

点数据

以整数值、P编号、P(表达式)、点标签进行指定。无法在点数据自变量中指定变量。要使用变量时，请记述为PDescription$(P (varName))。

返回值

以字符串返回指定点编号的注释。

参阅

PDef函数、PDescription、PLabel、PLabel$函数

PDescription$函数使用示例

```
Print PDescription$(1)
Print PDescription$(P(i))
```

3.19.15 PeakSpeedClear

用于清除关节的峰值速度并进行初始化。

格式

PeakSpeedClear [关节指定1 [, 关节指定2 [, 关节指定3 [, 关节指定4 [, 关节指定5 [, 关节指定6 [, 关节指定7 [, 关节指定8 [, 关节指定9]]]]]]]]]

参数

关节指定1 - 关节指定9
以整数值或表达式指定关节编号。未指定参数时，所有关节的峰值速度将被清除。附加轴的S轴为8，T轴为9。如果指定不存在的关节编号，将发生错误。

说明

PeakSpeedClear用于清除指定关节的峰值速度值。

请务必在执行PeakSpeed之前执行PeakSpeedClear。

PG附加轴不支持本命令。

参阅

AvgSpeed、PeakSpeed

PeakSpeedClear使用示例

[例1] 如下所示为清除所有关节的峰值速度值之后，显示指定关节编号速度值的命令执行示例。

```
> PeakSpeedClear
> Go P1
> PeakSpeed 1
    -0.273
> PeakSpeed
    -0.273    -0.164
    -0.080     0.258
    -0.005     0.401
     0.000     0.000
     0.000
>
```

[例2] 如下所示为在垂直多关节机器人中清除第1关节、第4关节、第5关节的峰值速度值之后，显示指定关节编号峰值速度值的命令执行示例。

```
> PeakSpeedClear 4, 1, 5
> Go P1
> PeakSpeed 1
     -0.273
> PeakSpeed 4
     0.258
```

3.19.16 PeakSpeed

用于显示指定关节的峰值速度值。

格式

PeakSpeed [关节编号]

参数

关节编号
以整数值或表达式指定关节编号。可省略。附加轴的S轴为8，T轴为9。

结果

显示所有关节的当前峰值速度值。

说明

PeakSpeed用于显示在关节速度绝对值达到最大时在速度上附加符号的值。最大速度为1。以-1～1的实值表示峰值速度。

请在执行PeakSpeedClear之后执行PeakSpeed，显示关节的峰值速度值。

为虚拟控制器方式以及空运行时，根据指令速度(而非实际速度)运算峰值速度。

PG附加轴不支持本命令。

参阅

AvgSpeed、PeakSpeedClear、PeakSpeed函数

PeakSpeed使用示例

```
> PeakSpeedClear
> Go P1
> PeakSpeed 1
     -0.273
> PeakSpeed
    -0.273     0.163
    -0.080     0.258
    -0.005    -0.401
     0.000     0.000
     0.000
>
```

3.19.17 PeakSpeed函数

用于返回指定关节的峰值速度。

格式

PeakSpeed (关节编号)

参数

关节编号
以整数值或表达式指定关节编号。附加轴的S轴为8，T轴为9。

返回值

以-1～1的实值返回。

说明

PeakSpeed函数用于返回在关节速度绝对值达到最大时在速度上附加符号的值。最大速度为1。以-1～1的实值表示峰值速度。

请在执行PeakSpeedClear之后执行PeakSpeed，显示关节的峰值速度值。

为虚拟控制器方式以及空运行时，根据指令速度(而非实际速度)运算峰值速度。

PG附加轴不支持本函数。

参阅

AvgSpeed、PeakSpeedClear、PeakSpeed

PeakSpeed函数使用示例

如下所示为程序中使用PeakSpeed函数的示例。

```
Function DisplayPeakSpeed
   Integer i

   PeakSpeedClear
   Go P1
   Print "Peak Speeds:"
   For i = 1 To 6
      Print "Joint ", i, " = ", PeakSpeed (i)
   Next i
Fend
```

3.19.18 PerformMode

可以选择机器人的模式。

格式

(1) PerformMode [模式编号] [, 机器人编号]

(2) PerformMode

参数

模式编号

用一个整数值(0～2)或以下所示常数，指定要设置的模式编号。仅在从命令窗口执行时可省略。

常数	值	内容
MODE_STANDARD	0	设置标准模式。
MODE_BOOST	1	设置高速模式。
MODE_LOW_VIBRATRION	2	设置低振动模式。

机器人编号

以整数值指定要设置的机器人编号。已省略时，以当前选择的机器人为对象。

结果

- 如果指定(1)的格式，则用指定的模式编号设置动作模式。
- 如果指定(2)的格式，将显示当前机器人设置的模式编号。

说明

PerformMode是根据应用，更改根据用途优先的机器人的性能(模式)的功能。各型号机械手的可选模式，请参阅以下手册。

《机械手手册》

- 标准模式：该模式优先考虑节拍时间、动作占空比和动作停止时的振动平衡。适用于所有应用。
- 高速模式：该模式优先缩短单个动作时间。与标准模式相比，动作占空比和动作停止时的振动会变大，但是缩短了1个动作时间。若使用在高负载应用中，该模式可以缩短循环时间。

  推荐应用：搬运等
- 低振动模式：该模式优先抑制动作停止时产生的振动。与标准模式相比，动作时间变长，但是停止时的振动会变小。

  推荐应用：精密部件的搬运和组装等

各模式的性能对比

模式	比较项目		
	单个动作时间 (*1)	停止时的振动	动作Duty (*2)
标准	○	○	○
高功率	◎	△	△
低振动	△	◎	○

表中的符号表示性能的程度。

○： 标准 ◎：稍高 ：稍低

- (*1) 单个动作时间是指，机器人从当前位置移动至目标坐标所花费的时间。

- (*2) 动作占空比是指，没有过载错误的情况下，可以在最大加减速度下运行的运行时间百分比。

注意

- 如果再不支持该功能的机型中，将模式更改为高功率模式或低振动模式时，会发生错误。
- 对应动作指令：PTP动作(Go, BGo, TGo, Jump, JTran, PTran, Pulse)
- CP动作以下的性能不变化。
  - 轨迹精度
  - AccelS、AccelR、SpeedS、SpeedR的上限值
  - 加速度指令错误、速度指令错误的发生概率。
- 模式自动变更为初始模式(标准模式)的条件

  下表给出模式自动变更的条件。

	模式的变化
控制器电源ON	变为初始模式
重启控制器	变为初始模式
电动机ON	变为初始模式
创建/重建	不变更
执行Reset时	不变更
执行SFree时	变为初始模式

参阅

Bo、Go、Jump、JTran、PerformMode函数、TGo

PerformMode使用示例

```
PerformMode MODE_STANDARD
Go P1
PerformMode 2
Go P2
```

3.19.19 PerformMode函数

用于返回机器人的动作模式的状态。

格式

PerformMode ([机器人编号])

参数

机器人编号
以整数值指定要确认状态的机器人编号。已省略时，以当前选择的机器人为对象。

返回值

以整数值返回当前设置的动作模式的值。

- 0 = 标准模式
- 1 = 高功率模式
- 2 = 低振动模式

参阅

PerformMode

PerformMode函数使用示例

```
Print PerformMode(1)
```

3.19.20 PG_FastStop

用于紧急停止正在连续旋转的脉冲输出轴。

格式

PG_FastStop

说明

如果执行PG_FastStop，则紧急停止当前选择的PG机器人的连续旋转动作。通常停止时，请使用减速停止(PG_SlowStop)。

参阅

PG_Scan、PG_SlowStop

PG_FastStop使用示例

下例所示为对脉冲输出轴进行10秒钟的连续旋转动作然后紧急停止。

```
Function main
  Motor On
  PG_Scan 0
  Wait 10
  PG_FastStop                    '紧急停止连续旋转动作
Fend
```

3.19.21 PG_LSpeed

用于设置脉冲输出轴开始加速时的脉冲速度以及结束减速时的脉冲速度。

格式

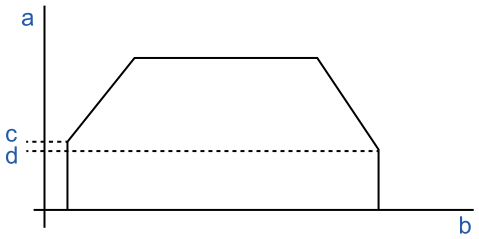
PG_LSpeed 速度设置值 [, 减速末速度],

参数

速度设置值
　　以表达式或数值指定脉冲速度(1～32767的整数，单位：pulse/sec)。
减速结束速度
　　以表达式或数值指定结束减速时的脉冲速度(1～32767的整数，单位：pulse/sec)。

说明

PG_LSpeed用于指定脉冲输出轴开始加速时的脉冲速度以及结束减速时的脉冲速度。用于在最大自启动频率范围内设置较高的步进电动机初始/结束速度，发挥高性能电动机的性能，以及防止设置较低的步进电动机速度造成的失调。默认值为300pulse/sec，通常直接使用该值。



符号	说明
a	速度
b	时间
c	加速初速度
d	减速末速度

如果省略减速末速度，速度设置值则为减速末速度的设置值。

下述某种情况时，PG_LSpeed值会被初始化。

- 启动控制器时
- 执行Motor On
- 执行SFree、SLock、Brake
- 执行Reset、Reset Error
- 利用停止按钮或执行Quit All等结束任务

参阅

PG_LSpeed函数

PG_LSpeed使用示例

可在命令窗口和程序中使用PG_LSpeed。如下所示为两种情况下的使用示例。

```
Function pglspeedtst
        Motor On
        Power High
Speed 30;Accel 30.30
PG_LSpeed 1000
```

```
Go P0
Fend
```

如下所示为通过命令窗口设置PG_LSpeed值的示例。

```
> PG_LSpeed 1000,1100
>
```

3.19.22 PG_LSpeed函数

用于返回当前设置的脉冲输出轴开始加速时的脉冲速度以及结束减速时的脉冲速度。

格式

PG_LSpeed [(参数编号)]

参数

参数编号

从下述编号中指定1个设置值编号。省略时，视为指定1。

- 1：开始加速时的脉冲速度
- 2：结束减速时是脉冲速度

返回值

返回1～32767的整数值，单位：pulse/sec。

参阅

PG_LSpeed

PG_LSpeed函数使用示例

```
Integer savPGLSpeed

savPGLSpeed = PG_LSpeed(1)
```

3.19.23 PG_Scan

用于开始脉冲输出轴的连续旋转动作。

格式

PG_Scan 旋转方向

参数

旋转方向
指定连续旋转的旋转方向。

- 0：+ (CW) 方向
- 1：- (CCW) 方向

说明

如果执行PG_Scan，则开始当前选择的PG机器人的连续旋转动作。

要执行连续旋转动作时，需要在机器人设置中将PG参数的连续旋转设为有效。

程序执行任务结束时，停止连续旋转。

参阅

PG_FastStop

PG_Scan使用示例

下例所示为对脉冲输出轴进行10秒钟的连续旋转动作然后紧急停止。

```
Function main
  Motor On
  Power High
  Speed 10; Accel 10,10
  PG_Scan 0
  Wait 10
  PG_SlowStop
Fend
```

3.19.24 PG_SlowStop

用于减速停止正在连续旋转的脉冲输出轴。

格式

PG_SlowStop

说明

如果执行PG_SlowStop，则减速停止当前选择的PG机器人的连续旋转动作。

参阅

PG_Scan、PG_FastStop

PG_SlowStop使用示例

下例所示为对脉冲输出轴进行10秒钟的连续旋转动作然后紧急停止。

```
Function main
  Motor On
  PG_Scan 0
  Wait 10
  PG_SlowStop                    '紧急停止连续旋转动作
Fend
```

3.19.25 PLabel

用于设置指定点数据的标签。

格式

PLabel点编号, 新标签

参数

点编号
　　以表达式或数值指定点编号。
新标签
　　指定用于指定点数据的标签的字符串。

参阅

PDef函数、PDescription、PDescription$函数、PLabel$函数、PNumber函数

PLabel使用示例

```
PLabel 1, "pick"
```

3.19.26 PLabel$函数

用于返回定义为指定点编号的点标签。

格式

PLabel$ (点数据)

参数

点数据

以整数值、P编号、P(表达式)、点标签进行指定。有关兼容性的注意事项不能将点数据自变量指定为变量。要使用变量时，请记述为PLabel$(P(varName))。

参阅

PDef函数、PDescription、PDescription$函数、PLabel、PNumber函数

PLabel$函数使用示例

```
Print PLabel$(1)
Print PLabel$(P(i))
```

3.19.27 Plane

用于设置/显示进入检测平面。

格式

(1) Plane 平面编号 [, 机器人编号], 坐标系数据 Plane 平面编号 [, 机器人编号], 原点, X轴指定, Y轴指定

(3) Plane 平面编号 [, 机器人编号]

(4) Plane

参数

平面编号
　　指定进入检测平面的编号。可利用1～15的整数定义最多15个进入检测平面。
机器人编号
　　以整数值指定要设置的机器人编号。已省略机器人编号时，以当前选择的机器人为对象。
坐标系数据
　　直接以点数据指定进入检测平面的坐标系。
原点
　　以P#(整数)或P(表达式)指定定义进入检测平面原点的机器人坐标系上的位置。
X轴指定
　　以P#(整数)或P(表达式)指定定义进入检测平面X轴上的点的机器人坐标系上的位置。
Y轴指定
　　以P#(整数)或P(表达式)指定定义进入检测平面Y轴上的点的机器人坐标系上的位置。

结果

- 如果以(3)的格式继续指定，则显示指定平面编号的进入检测平面设置。
- 如果以(4)的格式指定，则显示当前选择机器人所设置的所有进入检测平面设置。

说明

Plane用于设置进入检测平面。进入检测平面用于检测利用当前选择的工具所计算的卡爪工具位置是否进入到按设置的进入检测平面划分的空间一侧。由位于机器人基础坐标系上的任意坐标系的XY平面来设置进入检测平面。并且，在按该平面划分的空间中，如果卡爪工具位置进入到包括该坐标系Z轴+方向的空间之内，则视为进入到进入检测平面中。

如果设置进入检测平面，控制器启动期间则始终执行检测处理，与机器人的电动机电源状态无关。

下面说明各种格式。

(1) 利用以基础坐标系为基准并且表示平移量和旋转量的点数据来指定进入检测平面的源坐标系，然后设置进入检测平面。

例：

```
Plane 1, XY(x, y, z, u, v, w)
Plane 1, P1
```

(2) 指定原点、X轴上的点、Y轴上的点共3点，定义进入检测平面(XY平面)。仅使用各点中的X、Y、Z坐标，无视U、V、W坐标。同时计算Z轴以形成右手坐标系，并设置进入检测方向。

例：

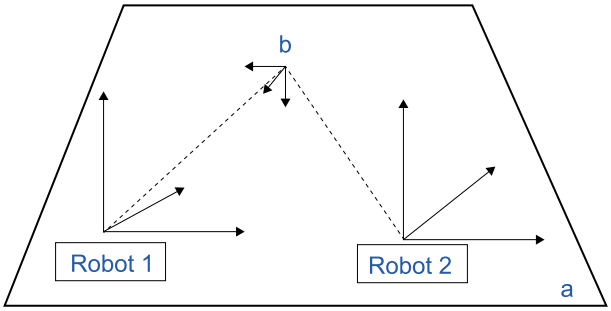
```
Plane 1, P1, P2, P3
```

(3) 显示指定进入检测平面的设置。

(4) 显示设置的所有进入检测平面的设置。

可利用GetRobotInsidePlane函数、InsidePlane函数随时获取对于进入检测平面的进入检测结果。另外，作为Wait命令的等待条件表达式，可利用GetRobotInsidePlane函数。也可以进行远程输出设置，以便将检测结果输出到I/O中。

多个机器人共享一个平面时，需要定义从各机器人坐标看到的平面。



符号	说明
a	进入检测平面
b	进入检测平面坐标系

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

注意

- 工具选择

  利用当前选择的工具进行进入检测。已变更工具选择时，机器人不会进行动作，可能会出现从平面内到平面外或相反的情况。

- 附加轴

  机器人带有附加轴(包括移动轴)S、T轴时设置的进入检测平面不依赖于附加轴的位置。而是以机器人的基础坐标系为基准进行设置。

参阅

Box, GetRobotInsidePlane, InsidePlane, PlaneClr, PlaneDef

提示

- 利用机器人管理器设置Plane

  在Epson RC+中，可通过[项目]菜单 - [机器人管理器]的[进入检测平面]面板设置Plane值。

Plane使用示例

从命令窗口执行的操作示例。

如下所示为以机器人坐标系为基准，在Z轴方向-20 mm水平面上将向下方向定义为检测方向的示例：

```
> plane 1, xy(100, 200, -20, 90, 0, 180)
```

如下所示为以机器人坐标系为基准，将向X方向移动50 mm、向Y方向移动200 mm并沿着Y轴转动45度获得的坐标系形成的XY平面定义为进入检测平面的示例：

```
> plane 2, xy(50, 200, 0, 0, 45, 0)
```

直接使用机器人的工具坐标系，设置进入检测平面。(垂直6轴型机器人)

```
> plane 3, here
```

3.19.28 Plane函数

用作返回已设置进入检测平面的函数。

格式

Plane (平面编号 [, 机器人编号])

参数

平面编号
　　以表达式或数值指定要确认的进入检测平面编号(1～15的整数)。
机器人编号
　　以整数值指定要设置的机器人编号。已省略机器人编号时，以当前选择的机器人为对象。

返回值

将已设置的进入检测平面源坐标系数据作为点数据进行返回。

参阅

GetRobotInsidePlane、InsidePlane、Plane、PlaneClr、PlaneDef

Plane函数使用示例

```
P1 = Plane(1)
```

3.19.29 PlaneClr

用于清除(未定义)进入检测平面。

格式

PlaneClr 平面编号 [, 机器人编号]

参数

平面编号
　　以表达式或数值指定要清除(未定义)设置的进入检测平面编号(1～15的整数)。
机器人编号
　　以整数值指定要设置的机器人编号。已省略机器人编号时，以当前选择的机器人为对象。

说明

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

参阅

GetRobotInsidePlane、InsidePlane、Plane、PlaneDef

PlaneClr使用示例

```
PlaneClr 1
```

3.19.30 PlaneDef函数

用于返回进入检测平面的设置状态。

格式

PlaneDef (平面编号 [, 机器人编号])

参数

平面编号
　　指定返回状态的进入检测平面的编号(1～15的整数)。
机器人编号
　　以整数值指定要设置的机器人编号。已省略机器人编号时，以当前选择的机器人为对象。

返回值

如果已设置由平面编号指定的进入检测平面，则返回“True”；如果未设置，则返回“False”。

参阅

GetRobotInsidePlane、Box、InsidePlane、Plane、PlaneClr

PlaneDef函数使用示例

```
Function DisplayPlaneDef(planeNum As Integer)

   If PlaneDef(planeNum) = False Then
      Print "Plane ", planeNum, "is not defined"
   Else
      Print "Plane 1: ",
      Print Plane(PlaneNum)
   EndIf
Fend
```

3.19.31 PList

用于显示当前机器人存储器中的点数据。

格式
(1) Plist

(2) Plist 点编号

(3) Plist 起点编号,

(4) Plist 起点编号, 终点编号

参数

点编号
　　编号范围为0～999。
起点编号
　　以0～999指定要显示的开头的点编号。
终点编号
　　以0～999指定要显示的最后的点编号。

结果
返回点数据。

说明
Plist用于显示当前机器人存储器中的点数据。

指定范围的点数据不存在时，不显示数据。

如果将起点编号指定为大于终点编号的值，则会发生错误。

- (1) PList：显示所有位置数据。
- (2) PList点编号：显示指定的1个点编号的位置数据。
- (3) PList 起点编号,：显示起点编号～位置数据的结尾。
- (4) PList起点编号,：终点编号：显示指定的起点编号~终点编号之间的位置数据。

PList使用示例
显示的格式因机器人类型或附加轴有无而异。

下例所示为SCARA型机器人不带附加轴的情况。

下例所示为显示指定的点数据。

```
> plist 1
P1   = XY(   200.000,    0.000,  -20.000,    0.000 )  /R /0
>
```

下例所示为显示10～20的点数据。在本例当中，指定的范围内存在3个点数据。

```
> plist 10, 20
P10  =   XY( 290.000,   0.000,  -20.000,         0.000 ) /R /0
P12  =   XY( 300.000,   0.000,  0.000,  0.000 ) /R /0
P20  =   XY( 285.000,   10.000, -30.000,         45.000 )         /R /0
>
```

下例所示为显示10～结尾的点数据。

```
> plist 10,
P10  =   XY( 290.000,   0.000,  -20.000,        0.000 ) /R /0
P12  =   XY( 300.000,   0.000,  0.000,  0.000 ) /R /0
P20  =   XY( 285.000,   10.000, -30.000,        45.000 )        /R /0
P30  =   XY( 310.000,   20.000, -50.000,        90.000 )        /R /0
```

3.19.32 PLocal

用于设置指定点数据的本地属性。

格式

Plocal (点数据) = 本地编号

参数

点数据
以整数值、P编号、P(表达式)、点标签进行指定。有关兼容性的注意事项不能将点数据自变量指定为变量。要使用变量时，请记述为PLocal(P(varName))。

本地编号
以0～15的整数值或表达式指定要设置的本地属性编号。

参阅

PLocal函数

PLocal使用示例

```
PLocal(pick) = 1
```

3.19.33 PLocal函数

用于返回设为指定点编号的本地编号。

格式

Plocal (点数据)

参数

点数据

以整数值、P编号、P(表达式)、点标签进行指定。有关兼容性的注意事项不能将点数据自变量指定为变量。要使用变量时，请记述为PLocal(P(varName))。

返回值

返回指定点编号的本地编号。

参阅

PLocal

PLocal函数使用示例

```
Integer localNum

localNum = PLocal(pick)
```

3.19.34 Pls函数

用于返回当前位置各关节的脉冲值。

格式
Pls (关节编号)

参数

关节编号
　　指定求出当前脉冲值的关节。附加轴的S轴为8，T轴为9。

返回值
以数值返回由关节编号指定的关节当前位置的脉冲值。

说明
Pls用于读入各关节的当前脉冲值。可保存求出的值，并在此后与Pulse命令同时使用。

参阅
CX、CY、CZ、CU、CV、CW、Pulse

Pls函数使用示例
如下所示为求出各关节脉冲值并输出该值的简单示例。

```
Function plstest
  Real t1, t2, z, u
  t1 = pls(1)
  t2 = pls(2)
  z = pls(3)
  u = pls(4)
  Print "T1 joint current Pulse Value: ", t1
  Print "T2 joint current Pulse Value: ", t2
  Print "Z joint current Pulse Value: ", z
  Print "U joint current Pulse Value: ", u
Fend
```

3.19.35 PNumber函数

用于返回对应于点标签的点编号。

格式

Pnumber (点标签)

参数

点标签
　　指定当前点文件中的点标签或表示点标签的字符串。

参阅

PDef函数、PLabel$函数

PNumber函数使用示例

```
Integer pNum
String pointName$

pNum = PNumber(pick)

pNum = PNumber("pick")

pointName$ = "place"
pNum = PNumber(pointName$)
```

3.19.36 PosFound函数

用于返回Find命令时执行的状态。

格式

PosFound

返回值

如果在机械臂移动期间找到坐标位置，则返回“True”；如果不是，则返回“False”。

参阅

Find

PosFound函数使用示例

```
Find Sw(5) = ON
Go P10 Find
If PosFound Then
    Go FindPos
Else
    Print "Error: Cannot find the sensor signal."
EndIf
```

3.19.37 Power

用于将功率模式设为High或Low。

格式

(1) Power { High | Low } [, Forced]

参数

High | Low
　　设置High或Low。默认设置为Low。
Forced
　　可省略。通常会省略。

结果

如果省略参数，则显示当前的功率模式。

说明

用于将功率模式设为High或Low。另外，显示当前的功率模式。

- Low：如果将功率模式设为Low，低功率模式则会变为ON状态。这表示机器人缓慢地(250 mm/sec以下的速度)进行动作。另外，将电动机功率输出限制在较低水平。
- High：如果将功率模式设为High，低功率模式则会变为OFF状态。这表示机器人以由Speed、Accel、SpeedS、AccelS指定的速度、加减速度进行动作。

如下所示为切换到低功率模式的操作。此时，速度和加减速度被限制为各机器人的默认值。默认值记载于各机器人的规格表中。请参阅以下手册。
《Epson RC+ 用户指南 - 关于安全》

变为低功率模式的条件

- 控制器电源ON
- 执行Motor On
- 执行SFree、SLock、Brake
- 执行Reset、Reset Error
- 利用停止按钮或执行Quit All等结束任务

限制为默认值的设置

- Speed
- Accel
- SpeedS
- AccelS

注意

- 低功率模式(Power Low)与最大速度的关系

  在低功率模式下，电动机输出受到限制，实际的动作设速度在初始值的范围之内。设置低功率模式时，即使通过命令窗口或利用程序发出设为高速的指示，也以初始值速度进行动作。如果需要以更高的速度进行动作，必须设为Power High。

  机器人在高功率模式中运动时，当切换至低功率模式，可能会导致超速错误和低功率扭矩错误。

- 高功率模式(Power High)与最大速度的关系

在高功率模式下，可使用高于初始值的高速速度。

- Forced标志

  可在机器人动作期间(也包括暂停期间)变更功率模式。

  如果在机器人在以低功率进行动作期间切换为高功率模式，可能从下一动作开始以设置的速度进行高速动作。

  如果在机器人在以高功率进行动作期间切换为低功率模式，可能导致超速错误或低功率转矩错误。

  请暂停机器人动作，然后指定Forced标志并切换为低功率模式。

参阅

Accel, AccelS, Speed, SpeedS

Power使用示例

如下所示为通过命令窗口使用Power的操作示例。

```
> Speed 50            '在Low Power模式下设置高速度

> Accel 100, 100      '设置高加速度

> Jump P1             '以速度、低加速度进行移动

> Speed               '显示当前的速度
Low Power Mode
   50
   50     50

> Accel               '显示当前的加速度
Low Power Mode
   100    100
   100    100
   100    100

> Power High          '设为High Power模式

> Jump P2             '以高速度进行机器人动作
```

3.19.38 Power函数

用于返回功率模式。

格式

Power [(机器人编号)]

参数

机器人编号
以整数值指定要确认状态的机器人编号。已省略时，以当前选择的机器人为对象。

返回值

- 0 = 低功率模式
- 1 = 高功率模式

参阅

Power

Power函数使用示例

```
If Power = 0 Then
    Print "Low Power Mode"
EndIf
```

3.19.39 PPls函数

用于根据指定的坐标值计算并返回关节脉冲。

格式

PPls (坐标值, 关节编号)

参数

坐标值
　　指定点数据。
关节编号
　　以表达式或数值指定关节编号(1～9的整数)。附加轴的S轴为8，T轴为9。

返回值

以Long型数值返回计算的关节脉冲。

参阅

Agl、CX、CY、CZ、CU、CV、CW、PAgl

PPls函数使用示例

```
Long pulses1

pulses1 = PPls(P10, 1)
```

3.19.40 Print

用于在运行窗口、操作窗口、命令窗口、宏窗口等显示器界面上显示数据。

格式

Print 显示数据 [, 显示数据...] [,]

Print

参数

显示数据
　　以数值或字符串表达式进行指定。可省略。
,(逗号)
　　如果语句的结尾有逗号，则不进行改行。可省略。

结果

返回变量数据或指定的字符串。

说明

在显示装置中显示变量数据或字符串。

除非行末有逗号，否则自动进行改行。

注意

本命令一次可处理的最大数据长度为256 Byte。

- 请勿在循环语句中仅使用Print命令

  如果在循环语句中仅使用Print命令，可能会导致控制器变为挂起状态。

  请与Wait命令或动作命令一起使用。

  不良示例

```
Do
    Print "1234"
Loop
```

  良好示例

```
Do
    Print "1234"
    Wait 0.1
Loop
```

参阅

Print #

Print使用示例

下例所示为从P100的坐标值中减去U轴坐标值，然后将该差值代入到变量uvar中得到的值显示在当前的显示器中。

```
Function test
  Real uvar
  uvar = CU(P100)
```

```
  Print "The U Axis Coordinate of " + Chr$(34) + "P100" + Chr$(34) + " is ", uvar
Fend
```

3.19.41 Print

用于将数据输出到指定的文件、通信端口、数据库或装置中。

格式

Print #端口编号, 输出数据 [, 输出数据...] [,]

参数

端口编号
　　是表示文件、通信端口、数据库或装置的ID编号。文件编号是由ROpen、WOpen、AOpen等语句指定的编号。通信端口编号是由OpenCom(RS-232C)或OpenNet(TCP/IP)语句指定的编号。数据库编号是由OpenDB语句指定的编号。
　　装置ID为以下数值。
- 21 RC+
- 24 TP(仅TP1)
- 20 TP3

输出数据...
　　指定数值或字符串。
,(逗号)
　　如果语句的结尾有逗号，则不进行改行。可省略。

说明

Print #用于将变量数据、数值或字符串输出到由端口编号指定的通信端口或装置中。

注意

- 支持的控制器型号

  若在T/VT系列中指定RS-232C端口时会发生错误。

- 最大数据长度
  - 最大数据长度

  本命令一次可处理的最大数据长度为256 Byte。但对象为数据库时，最大数据长度为4096 Byte。对象为通信端口(TCP/IP)时，最大数据长度为1024 Byte。

- 与其它控制器进行变量交换时

  指定多个字符串变量，以及指定数值变量和字符串变量两者时，需要在字符串数值数据中明确添加分隔符(“,”)。

  使用通信端口交接控制器之间的字符串变量、数值变量。

  发送侧(任一模式即为OK。)

```
Print #PortNum, "$Status,", InData, OutData
Print #PortNum, "$Status", ",",InData, OutData
```

  接收侧

```
Input #PortNum, Response$, InData, OutData
```

- 向文件写入时进行缓冲。

  可利用Flush语句写入被缓冲的数据。利用Close语句关闭文件时也进行写入。

- 请勿同时使用Print #命令、Wait命令和动作命令
  请勿在循环语句中仅使用Print #命令

  如果在循环语句中仅使用Print #命令，可能会导致控制器变为挂起状态。

  根据控制器的负载情况，即使使用Wait命令或动作命令时，信息也可能无法正常显示。如果是TP1输出，请将Wait时间设置在1(秒)以上。其他输出的情况下，请设置在0.1(秒)以上。

  不良示例

```
Do
    Print #24, "1234"
Loop
```

  良好示例

```
Do
    Print #24,"1234"
    Wait 1
Loop
```

参阅

Input#、Print、Write、WriteBin

Print #使用示例

如下所示为使用Print #的简单示例。

```
Function printex
  String temp$
  Print #1, "5"     ' 将"5"输出到端口1中 temp$ = "hello"
  Print #1, temp$
  Print #2, temp$
  Print #1 " Next message for " + Chr$(34) + "port 1" + Chr$(34)
  Print #2 " Next message for " + Chr$(34) + "port 2" + Chr$(34)
Fend
```

3.19.42 PTCLR

用于清除关节的峰值转矩并执行初始化。

格式

PTCLR [关节指定1 [, 关节指定2 [, 关节指定3 [, 关节指定4 [, 关节指定5 [, 关节指定6 [, 关节指定7 [, 关节指定8 [, 关节指定9]]]]]]]]]

参数

关节指定1 - 关节指定9
以整数值或表达式指定关节编号。未指定参数时，所有关节的峰值转矩值都会被清除。附加轴的S轴为8，T轴为9。如果指定不存在的关节编号，将发生错误。

说明

PTCLR用于清除指定关节的峰值转矩值。

执行PTRQ之前，请务必执行PTCLR。

参阅

ATRQ、PTRQ

PTCLR使用示例

[例1] 如下所示为清除所有关节的峰值转矩值之后，显示指定关节编号转矩值的命令执行示例。

```
> ptclr
> go p1
> ptrq 1
    0.227
> ptrq
    0.227    0.118
    0.249    0.083
    0.000    0.000
>
```

[例2] 如下所示为在垂直多关节机器人中清除J1、J4、J5的峰值转矩值之后，显示指定关节编号转矩值的命令执行示例。

```
> ptclr 4, 1, 5
> go p1
> ptrq 1
    0.227
> ptrq 4
    0.083
```

3.19.43 PTPBoost

用于设置/显示PTP(point to point)动作微移时的加减速算法调整参数。

格式

(1) PTPBoost 调整设置值 [, Jump转移调整] [, Jump接近调整]

(2) PTPBoost

参数

调整设置值
　　以表达式或数值指定调整设置值(0～100的整数)。
Jump转移调整
　　以表达式或数值指定Jump动作期间Z坐标的转移调整设置值(0～100的整数)。可省略。
Jump接近调整
　　以表达式或数值指定Jump动作期间Z坐标的接近调整设置值(0～100的整数)。可省略。

结果

如果省略参数时，则显示当前的PTPBoost设置值。

说明

用于设置PTP动作微距移动时的加减速算法。仅在移动距离为微小时本设置值才有效。可利用PTPBoostOK函数确认移动到目标坐标的距离是否为微小距离。

通常使用PTPBoost时无需变更设置值。请用于微移动作时也要稍微缩短循环时间之时，或相反地，即使延长循环时间也要减轻残留振动之时。

如果增大设置值，循环时间则会缩短，但停止时易于发生振动。另外，如果减小设置值，循环时间则会延长，但停止时不易发生振动。如果在停止时振动较大的状态下使用机器人，则会发生错误或冲击，这不仅会无法充分发挥机器人的性能，也可能会导致机械手的使用寿命缩短，敬请注意。

下述某种情况时，PTPBoost值会被初始化。

- 启动控制器时
- 执行Motor On
- 执行SFree、SLock、Brake
- 执行Reset、Reset Error
- 利用停止按钮或执行Quit All等结束任务

参阅

PTPBoost函数、PTPBoostOK

PTPBoost使用示例

```
PTPBoost 50, 30, 30
```

3.19.44 PTPBoost函数

用于返回指定的PTPBoost值。

格式

PTPBoost (参数编号)

参数

设置编号

以整数设置下述设置值。

- 1：调整设置值
- 2：Jump转移调整设置值
- 3：Jump接近调整设置值

返回值

返回调整设置值(0～100的整数值)。

参阅

PTPBoost、PTPBoostOK

PTPBoost函数使用示例

```
Print PTPBoost(1)
```

3.19.45 PTPBoostOK函数

用于返回从当前位置向目标坐标进行的PTP(point to point)动作是否为微移。

格式

PTPBoostOK (目标坐标)

参数

目标坐标
以点数据指定目标坐标。

返回值

从当前位置向目标坐标进行的PTP动作如果为微小移动，则返回“True”；如果不是，则返回“False”。

说明

从当前位置向目标坐标进行的PTP动作用于确认通过PTPBoost命令实现的加减速算法调整是否为有效微移。

参阅

PTPBoost

PTPBoostOK函数使用示例

```
If PTPBoostOK(P1) Then
  PTPBoost 50
EndIf
Go P1
```

3.19.46 PTPTime函数

用于返回PTP动作命令的推测所需时间。不进行PTP动作。

格式

(1) PTPTime (目标坐标, 目标机械臂, 目标工具)

(2) PTPTime (起始坐标, 起始机械臂, 起始工具, 目标坐标, 目标机械臂, 目标工具)

参数

开始坐标
　　以点表达式指定起始位置。
目标坐标
　　以点表达式指定目标坐标。
目标机械臂
　　以整数值或表达式指定目标位置的机械臂编号。
目标工具
　　以整数值或表达式指定目标位置的工具编号。
起始机械臂
　　以整数值或表达式指定起始位置的机械臂编号。
起始工具
　　以整数值或表达式指定起始位置的工具编号。

返回值

以秒为单位返回实值。

说明

可使用PTPTime函数估算PTP动作命令(Go)所需执行时间。要估算从当前位置到目标位置的所需时间时，请使用格式(1)；要估算从起始位置到目标位置的所需时间时，请使用格式 (2)。

即使执行该函数，也不进行实际的PTP动作。当前的位置设置、机械臂设置及工具设置未被变更。

如果设置无法到达的位置，或错误设置机械臂或工具，返回值则会变为0。

机器人带有附加轴并且附加轴由伺服轴构成时，也要加上附加轴的动作时间。附加轴为脉冲输出轴时，返回机器人自身的动作时间。

参阅

ATRQ、Go、PTRQ

PTPTime函数使用示例

```
Real secs

secs = PTPTime(P1, 0, 0, P2, 0, 1)
Print "Time to go from P1 to P2 is:", secs

Go P1
secs = PTPTime(P2, 0, 1)
Print "Time to go from P1 to P2 is:", secs
```

3.19.47 PTran

用于从当前位置进行指定脉冲量的仅1关节的PTP动作。

格式

PTran 关节编号, 脉冲

参数

关节编号
　　指定要移动的关节编号的整数值。附加轴的S轴为8，T轴为9。
脉冲
　　指定要移动的脉冲量。

说明

Ptran用于从当前位置进行指定脉冲量的仅1关节的移动。

参阅

Go、JTran、Jump、Move

Ptran使用示例

```
PTran 1, 2000
```

3.19.48 PTRQ

用于显示指定关节峰值转矩值。

格式
PTRQ [关节编号]

参数

关节编号
以整数值或表达式指定关节编号。可省略。附加轴的S轴为8，T轴为9。

结果
显示所有关节的当前峰值转矩值。

说明
请在执行PTCLR之后执行PTRQ，显示关节的峰值转矩值。

以0～1的实值表示峰值转矩。

参阅
ATRQ、PTCLR、PTRQ函数

PTRQ使用示例

```
> ptclr
> go p1
> ptrq 1
    0.227
> ptrq
    0.227    0.118
    0.249    0.083
    0.000    0.000
>
```

3.19.49 PTRQ函数

用于返回指定关节的峰值转矩。

格式

PTRQ (关节编号)

参数

关节编号
以整数值或表达式指定关节编号。附加轴的S轴为8，T轴为9。

返回值

以0～1的实值返回。

参阅

ATRQ、PTCLR、PTRQ

PTRQ函数使用示例

如下所示为程序中使用PTRQ函数的示例。

```
Function DisplayPeakTorque
   Integer i

   Print "Peak torques:"
   For i = 1 To 4
      Print "Joint ", i, " = ", PTRQ(i)
   Next i
Fend
```

3.19.50 Pulse

用于以PTP动作将机械臂移动到由各关节脉冲值指定的点位置。

格式

(1) Pulse 第1关节脉冲值, 第2关节脉冲值, 第3关节脉冲值, 第4关节脉冲值[, 第5关节脉冲值, 第6关节脉冲值] [, 第7关节脉冲值] [, 第8关节脉冲值, 第9关节脉冲值]

(2) Pulse

参数

第1～第4关节的脉冲值
　　指定最初4个关节的脉冲值。以整数值或Long表达式指定由Range命令指定的范围内的值。
第5关节脉冲值、第6关节脉冲值
　　用于垂直6轴型机器人(包括N系列)和关节型6轴机器人。可省略。
第7关节脉冲值
　　用于关节型7轴机器人。可省略。
第8关节脉冲值、第9关节脉冲值
　　用于附加轴。可省略。

结果

如果省略参数，则显示表示当前机器人位置的脉冲值。

说明

Pulse用于使用基于0脉冲位置的的各关节脉冲值(而非直角坐标系的坐标)来表示机械臂位置。使用Pulse命令以PTP动作方式移动机械臂。

由Range命令设置可用于Pulse命令的上限值和下限值。

注意

- 使用Pulse之前，请确认轨迹上没有障碍物

  与Jump不同，Pulse同时进行所有关节的动作，包括第3关节的上升和下降到目标位置。因此，使用Pulse时，请充分注意，以免夹具末端碰到障碍物。

常见错误

- 超出限制值的脉冲值

  如果由Pulse命令指示的脉冲值超出由Range设置的限制值范围，则会发生错误。

参阅

Go、Accel、Range、Speed、Pls、Pulse函数

Pulse使用示例

如下所示为利用命令窗口的操作示例。

将机械臂移动到由各关节脉冲值指定的位置上。

```
> pulse 16000, 10000, -100, 10
```

显示当前机械臂位置的第1～第4关节的脉冲值。

```
> pulse
PULSE:  1:  27306 pls 2:  11378 pls 3:  -3072 pls 4:   1297 pls
>
```

3.19.51 Pulse函数

用于将由脉冲指定的机器人坐标值返回到各关节。

格式

Pulse (第1关节脉冲值, 第2关节脉冲值, 第3关节脉冲值, 第4关节脉冲值 [, 第5关节脉冲值, 第6关节脉冲值] [, 第7关节脉冲值] [, 第8关节脉冲值, 第9关节脉冲值])

参数

第1～第4关节的脉冲值
　　指定第1关节～第4关节等各关节的脉冲值。以整数值或Long表达式指定由Range命令指定的范围内的值。
第5关节脉冲值、第6关节脉冲值
　　用于垂直6轴型机器人(包括N系列)和关节型6轴机器人。可省略。
第7关节脉冲值
　　用于关节型7轴机器人。可省略。
第8关节脉冲值、第9关节脉冲值
　　用于附加轴。可省略。

返回值

以指定脉冲返回所使用的机器人坐标。

参阅

Go、JA、Jump、Move、Pulse、XY

Pulse函数使用示例

```
Jump Pulse(1000, 2000, 0, 0)
```

3.20 Q

3.20.1 QP

用于设置/解除快速暂停功能，显示当前设置。

格式
(1) QP { On | Off }

(2) QP

参数

On | Off
　　指定快速暂停的On(设置)或Off(解除)。

结果
如果省略参数，则显示当前的QP设置。

说明
利用快速暂停功能设置在执行动作命令时，是否按下Pause开关；控制器内有暂停输入时，是否立即停止机器人；是否等待动作命令执行结束后停止机器人。

将立即减速停止称之为“快速暂停”。

如果指定参数“On”，QP则设为快速暂停。如果指定参数“Off”，QP则解除快速暂停。

QP用于显示当前的设置状态，比如，机器人是否对暂停输入作出反应，立即停止动作，或在完成动作之后是否停止。QP命令用于显示是否设置/解除快速暂停功能的状态。

注意

- 电源ON时，快速暂停功能被设为默认设置。

  即使在执行Reset命令之后，也会保存由QP命令设置的快速暂停功能设置。但如果将控制器或驱动单元的电源设为OFF，然后再设为ON，快速暂停功能将变为默认的“ON(设置)”。

- QP和安全门输入

  即使解除快速暂停功能设置，如果安全门输入为“开”，机器人则会立即停止动作。

参阅
Pause

QP使用示例
在下述利用命令窗口的操作示例中，显示使用暂停输入时，机器人是否立即停止(是否设置快速暂停功能)。

```
> qp
QP ON

> qp on   '设为快速暂停模式
>
```

3.20.2 QPDecelR

用于设置有关CP动作时工具姿势变化的快速暂停减速度。

格式
(1) QPDecelR QP减速度设置值

(2) QPDecelR

参数

QP减速度设置值
以实值指定CP动作的快速暂停时的减速度。(单位：deg/sec^2)

结果
如果省略参数，则显示当前的QPDecelR设置值。

说明
QPDecelR仅在Move、Arc、Arc3、BMove、TMove、Jump3CP中使用ROT修饰参数时有效。

如果在上述动作期间执行快速暂停，则可能会发生关节过加速度错误。这是因为，在通常的快速暂停动作中，自动设置的快速暂停限制速度超出了关节容许减速度的缘故。尤其在CP动作的AecelS设置值较大时，或在通过机器人特殊方向属性附近的情况下，易于发生这种错误。发生这种错误时，请利用QPDecelR将快速暂停减速度设得低一些。如果QPDecelR过小，快速暂停所需的移动量则会增加，因此，请尽可能设置较大的值。通常无需设置QPDecelR。

QPDecelR设置的减速度不能小于由AccelR设置的CP动作姿势变化减速度。否则会发生超出参数范围错误。

另外，设置QPDecelR之后，如果利用AccelR设置大于已设置QP减速度设置值的减速度，QPDecelR则自动设置与利用AccelR设置的减速度相同的QP减速度。

下述任一情况时，QPDecelR值将被初始化为默认的最大减速度。

- 启动控制器时
- 执行Motor On
- 执行SFree、SLock、Brake
- 执行Reset、Reset Error
- 利用停止按钮或执行Quit All等结束任务

参阅
QPDecelR函数、QPDecelS、AccelR

QPDecelR使用示例
如下所示为设置Move命令的QPDecelS的简单动作程序示例。

```
Function QPDecelTest
  AccelR 3000
QPDecelR 4000
  SpeedR 100
  Move P1 ROT
  .
  .
  .
Fend
```

3.20.3 QPDecelR函数

用作返回有关CP动作的工具姿势变化的快速暂停减速度设置值的函数。

格式

QPDecelR

返回值

返回有关CP动作的工具变化的快速暂停减速度设置值(实值，单位：deg/s^2)。

参阅

QPDecelR, QPDecelS函数

QPDecelR函数使用示例

```
Real savQPDecelR

savQPDecelR = QPDecelR
```

3.20.4 QPDecelS

用于设置CP动作时的快速暂停减速度。

格式

(1) QPDecelS QP减速度设置值 [, Jump3转移QP减速度设置值, Jump3接近QP减速度设置值]

(2) QPDecelS

参数

QP减速度设置值

以实值指定CP动作的快速暂停时的减速度。(单位：mm/sec^2)

Jump3转移QP减速度设置值

以实值指定Jump3转移动作的快速暂停时的减速度。(单位：mm/sec^2)

Jump3接近QP减速度设置值

以实值指定Jump3接近动作的快速暂停时的减速度。(单位：mm/sec^2)

结果

如果省略参数，则显示当前的QPDecelS设置值。

说明

如果在CP动作期间执行快速暂停，则可能会发生关节过加速度错误。这是因为，在通常的快速暂停动作中，自动设置的快速暂停减速度超出了关节容许减速度的缘故。尤其在CP动作的AecelR设置值较大时，或在通过机器人特殊方向属性附近的情况下，易于发生这种错误。发生这种错误时，请利用QPDecelS将快速暂停减速度设得低一些。如果QPDecelS过小，快速暂停所需的移动量则会增加，因此，请尽可能设置较大的值。通常无需设置QPDecelS。

QPDecelS设置的减速度不能小于由AccelS设置的CP动作减速度。否则会发生超出参数范围错误。

另外，设置QPDecelS之后，如果利用AccelS设置大于已设置QP减速度设置值的减速度，QPDecelS则自动设置与利用AccelS设置的减速度相同的QP减速度。

下述某种情况时，QPDecelR值会被初始化为默认的最大减速度。

- 启动控制器时
- 执行Motor On
- 执行SFree、SLock、Brake
- 执行Reset、Reset Error
- 利用停止按钮或执行Quit All等结束任务

参阅

QPDecelS函数、QPDecelR、AccelS

QPDecelS使用示例

如下所示为设置Move命令的QPDecelR的简单动作程序示例。

```
Function QPDecelTest
  AccelS 3000
QPDecelS 4000
  SpeedS 100
  Move P1
  .
  .
  .
Fend
```

3.20.5 QPDecelS函数

用作返回CP动作减速度设置值的函数。

格式

QPDecelS (设置值编号)

参数

设置值编号

以整数值或表达式指定下述各值。

- 1：CP动作时的快速暂停减速度设置值
- 2：Jump3/Jump3CP时的转移动作快速暂停减速度设置值
- 3：Jump3/Jump3CP时的接近动作快速暂停减速度设置值

返回值

返回快速暂停减速度设置值(实值，单位：mm/s^2)。

参阅

QPDecelS、QPDecelR函数

QPDecelS函数使用示例

```
Real savQPDecelS

savQPDecelS = QPDecelS(1)
```

3.20.6 Quit

用于结束所指定的任务或所有任务的执行。

格式

Quit { 任务识别符 | All }

参数

任务识别符

以整数值或表达式指定任务名或任务编号。任务名为Xqt语句使用的函数名，或通过运行窗口或操作员窗口启动的函数名。

任务编号的指定(整数)

- 一般任务：1~32
- 后台任务：65～80
- Trap任务：257~267
- All：要结束后台任务以外的所有任务时进行指定。

说明

Quit用于结束当前执行的任务或利用Halt暂停的任务。

在指定的任务为NoPause任务、NoEmgAbort任务(执行Xqt时指定NoPause或NoEmgAbort以开始的特别任务)与后台任务时，Quit也用于结束任务的执行。另外，Quit All用于结束包括这些任务在内的后台任务以外的所有任务的执行。

如果执行Quit All，则将机器人控制参数设为下述设置值。

机器人控制参数

- Speed和SpeedR、SpeedS的设置值：(被初始化为初始值。)
- Accel和AccelR、AccelS的设置值：(被初始化为初始值。)
- QPDecelR、QPDecelS的设置值：(被初始化为初始值。)
- LimZ参数的设置值：(初始化为0。)
- CP参数的设置值：(初始化为Off。)
- SoftCP参数的设置值：(初始化为Off。)
- Fine的设置：(初始化为初始值。)
- Power Low设置：(变为低功率模式。)
- PTPBoost的设置值：(初始化为初始值。)
- TCLim、TCSpeed的设置值：(初始化为初始值。)
- PgLSpeed的设置值：(初始化为初始值。)

参阅

Exit、Halt、Resume、Xqt

Quit使用示例

如下所示为10秒钟之后结束2个任务的示例。

```
Function main
  Xqt winc1   '开始任务winc1
  Xqt winc2   '开始任务winc2
  Wait 10
  Quit winc1  '结束任务winc1
  Quit winc2  '结束任务winc2
Fend

Function winc1
```

```
  Do
     On 1; Wait 0.2
     Off 1; Wait 0.2
  Loop
Fend

Function winc2
  Do
     On 2; Wait 0.5
     Off 2; Wait 0.5
  Loop
Fend
```

3.21 R

3.21.1 RadToDeg函数

用于将弧度转换为角度。

格式
RadToDeg (弧度)

参数

弧度
以实值指定要转换为角度的弧度。

返回值
返回表示角度的Double型数值。

参阅
ATan、ATan2、DegToRad函数

RadToDeg函数使用示例

```
s = Cos(RadToDeg(x))
```

3.21.2 Randomize

用于进行随机数系列的初始化。

格式

(1) Randomize Seed值

(2) Randomize

参数

Seed值
以0以上的实值指定用于求出随机数的基础值。

参阅

Rnd函数

Randomize使用示例

```
Function main
    Real r
    Randomize
    Integer randNum

    randNum = Int(Rnd(10)) + 1
    Print "Random number is:", randNum
Fend
```

3.21.3 Range

用于设置/显示各伺服关节的容许动作区域。

格式

(1) Range 设置值1，设置值2，设置值3，设置值4，设置值5，设置值6，设置值7，设置值8 [, 设置值9, 设置值10, 设置值11, 设置值12] [, 设置值13, 设置值14] [, 设置值15, 设置值16, 设置值17, 设置值18]

(2) Range

参数

设置值1
　　第1关节的下限脉冲值(单位：脉冲)
设置值2
　　第1关节的上限脉冲值(单位：脉冲)
设置值3
　　第2关节的下限脉冲值(单位：脉冲)
设置值4
　　第2关节的上限脉冲值(单位：脉冲)
设置值5
　　第3关节的下限脉冲值(单位：脉冲)
设置值6
　　第3关节的上限脉冲值(单位：脉冲)
设置值7
　　第4关节的下限脉冲值(单位：脉冲)
设置值8
　　第4关节的上限脉冲值(单位：脉冲)
设置值9
　　第5关节的下限脉冲值(单位：脉冲)是垂直6轴型机器人(包括N系列)和关节型6轴机器人的专用参数。
设置值10
　　第5关节的上限脉冲值(单位：脉冲)是垂直6轴型机器人(包括N系列)和关节型6轴机器人的专用参数。
设置值11
　　第6关节的下限脉冲值(单位：脉冲)是垂直6轴型机器人(包括N系列)和关节型6轴机器人的专用参数。
设置值12
　　第6关节的上限脉冲值(单位：脉冲)是垂直6轴型机器人(包括N系列)和关节型6轴机器人的专用参数。
设置值13
　　第7关节的下限脉冲值(单位：脉冲)是关节型7轴型机器人的专用参数。
设置值14
　　第7关节的上限脉冲值(单位：脉冲)是关节型7轴型机器人的专用参数。
设置值15
　　第8关节的下限脉冲值(单位：脉冲)是附加轴S关节的专用参数。
设置值16
　　第8关节的上限脉冲值(单位：脉冲)是附加轴S关节的专用参数。
设置值17
　　第9关节的下限脉冲值(单位：脉冲)是附加轴T关节的专用参数。
设置值18
　　第9关节的上限脉冲值(单位：脉冲)是附加轴T关节的专用参数。

结果

如果省略参数，则显示当前的Range值。

说明

Range用于设置各电动机关节的下限/上限脉冲值。单位为脉冲。这样，用户可定义各关节的最大/最小容许动作范围。同样，使用XYLim命令来定义XY坐标方向的限制值。

Range的初始值因机器人的机型而已。在电源OFF之后，由该命令设置的值也会被保存。

省略参数时显示当前的Range值。

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

易引起的错误

- 移动到容许范围以外时

  即使1个关节移动到机械臂的容许动作范围以外，也会发生错误。

- 关节不动作

  下限脉冲值大于等于上限脉冲值时，关节不进行动作。

注意

第6关节的上限脉冲值与下限脉冲值的设置范围因机型而异。
有关设置范围，请参阅各机械手册的动作区域。

参阅

JRange、SysConfig、XYLim

Range使用示例

如下所示为通过命令窗口显示和变更当前Range设置值的简单示例。

```
> range
-18205, 182045, -82489, 82489, -36864, 0, -46695, 46695
>
> range 0, 32000, 0, 32224, -10000, 0, -40000, 40000
>
```

3.21.4 Read

用于从文件或通信端口读入指定的字符数。

格式

Read #端口编号, 字符串变量$, 字符数

参数

端口编号
　　是表示文件或通信端口的ID编号。文件编号是由ROpen、WOpen、AOpen等语句指定的编号。通信端口编号是由OpenCom(RS-232C)或OpenNet(TCP/IP)语句指定的编号。
字符串变量$,
　　指定接收字符串的字符串变量名。
字符数
　　指定要读入的字节数。

注意

- 支持的控制器型号

  若在T/VT系列中指定RS-232C端口时会发生错误。

参阅

ChkCom, ChkNet, OpenCom, OpenNet, Write, ReadBin

Read使用示例

```
Integer numOfChars
String data$

numOfChars = ChkCom(1)

If numOfChars > 0 Then
    Read #1, data$, numOfChars
EndIf
```

3.21.5 ReadBin

用于从文件或通信端口读取二进制数据。

格式

ReadBin #端口编号, 变量名

ReadBin #端口编号, 数组变量名(), 字节数

参数

端口编号
　　是表示文件或通信端口的ID编号。文件编号是由BOpen等语句指定的编号。通信端口编号是由OpenCom(RS-232C)或OpenNet(TCP/IP)语句指定的编号。
变量名
　　指定接收数据字节以及Byte型变量、整数变量或Long型变量的名称。
数组变量名()
　　指定接收数据字节以及Byte型变量、整数变量或Long型变量的名称。可指定一维数组变量。
字节数
　　指定要读入的字节数。需为最大数组下标以下且小于256Byte。以通信端口(TCP/IP)为对象时，需为最大数组下标以下且小于1024Byte。

注意

- 支持的控制器型号

  若在T/VT系列中指定RS-232C端口时会发生错误。

参阅

Write, WriteBin, Read

ReadBin使用示例

```
Integer data
Integer dataArray(10)

numOfChars = ChkCom(1)

If numOfChars > 0 Then
    ReadBin #1, data
EndIf

numOfChars = ChkCom(1)

If numOfChars > 10 Then
    ReadBin #1, dataArray(), 10
EndIf
```

3.21.6 Real

用于声明Real型变量。(4字节的实值)

格式

Real 变量名 [(数组变量的最大下标)] [, 变量名 [(数组变量的最大下标)]...]

参数

变量名
　　指定声明为Real型的变量名。
数组变量的最大下标
　　可利用数组变量的最大下标声明到最高三维。使用下述格式。可省略。
　　(最大下标1, [最大下标2], [最大下标3])
　　由于下标从0开始，因此元素数为最大下标加上1。在所有元素数不超过以下最大值的范围内指定各最大下标。-
　　本地变量：2,000 - 备份变量(Global Preserve)： 4,000 - 全局变量和模块变量：100,000

说明

Real用于声明Real型变量。在Function开头声明本地变量。在Function之外声明全局变量和模块变量。

Real型的有效位数为6位。

参阅

Boolean、Byte、Double、Global、Int32、Int64、Integer、Long、Short、String、UByte、UInt32、UInt64、UShort

Real使用示例

下例所示为使用Real声明Real型变量的程序。

```
Function realtest
  Real var1
  Real A(10)          'Real型的一维数组
  Real B(10, 10)      'Real型的二维数组
  Real C(5, 5, 5)     'Real型的三维数组
  Real arrayVar(10)
  Integer i
  Print "Please enter a Real Number:"
  Input var1
  Print "The Real variable var1 = ", var1
  For i = 1 To 5
    Print "Please enter a Real Number:"
    Input arrayVar(i)
    Print "Value Entered was ", arrayVar(i)
  Next i
Fend
```

3.21.7 RealAccel函数

是用作OLAccel调整后返回加减速度设置值的函数。

格式

RealAccel (设置值编号)

参数

设置值编号

以整数值指定下述各值。

- 1：加速设置值
- 2：减速设置值
- 3：Jump动作时的转移加速设置值
- 4：Jump动作时的转移减速设置值
- 5：Jump动作时的接近加速设置值
- 6：Jump动作时的接近减速设置值

返回值

返回大于1的整数(%)。

用途

通过使用RealAccel，可得到机器人能连续动作的最大加减速度。

步骤如下。

1. 在OLAccel命令设为On的状态下运行机器人。
2. 执行OLRate命令，确认过载率是否上升。
3. 过载率上升并达到0.5时，将开始自动调整加减速度。
4. 过了一定的时间后，请确认执行OLRate命令后，过载率是否停止上升。
5. 确认过载率不再上升后，执行RealAccel函数。
6. RealAccel函数返回的值为(1)的动作中机器人能连续动作的最大加减速度。
   - 如果在过载率上升时使用了RealAccel函数，无法获取机器人能连续动作的最大加减速度。
   - 如果发生过热错误，执行上述步骤也无法获取机器人能连续动作的最大加减速度。

参阅

Accel、OLAccel、OLRate

RealAccel函数使用示例

如下所示为程序中使用RealAccel函数的示例。

```
Integer RealAccel1, RealDecel1

Accel 100, 100
OLAccel on

'获取当前的加减速度
RealAccel1 = RealAccel (1)
RealDecel1 = RealAccel (2)

显示当前的加速度
Print RealAccel1
显示当前的减速度
Print RealDecel1
```

3.21.8 RealPls函数

用于返回已指定关节的脉冲值。

格式

RealPls (关节编号)

参数

关节编号
　　指定返回当前脉冲值的关节。附加轴的S轴为8，T轴为9。

返回值

以整数值返回由关节编号指定的关节的当前编码器脉冲值。

说明

以RealPls函数返回各关节的当前编码器位置或脉冲值。可保存该值并由Pulse命令使用。

参阅

CX、CY、CZ、CU、CV、CW、Pulse

RealPls函数使用示例

```
Function DisplayPulses

  Long joint1Pulses

  joint1Pulses = RealPls(1)
  Print "Joint 1 Current Pulse Value: ", joint1Pulses
Fend
```

3.21.9 RealPos函数

用于返回指定机器人当前位置。

格式
RealPos

返回值
返回指定机器人当前位置的点。

说明
RealPos函数用于返回机器人当前位置。

参阅
CurPos、CX、CY、CZ、CU、CV、CW、RealPls

RealPos函数使用示例

```
Function ShowRealPos

  Print RealPos
Fend

P1 = RealPos
```

3.21.10 RealTorque函数

用于返回指定关节当前转矩指令值。

格式

RealTorque (关节编号)

参数

关节编号
以表达式或数值指定要获取转矩指令值的关节的关节编号。附加轴的S轴为8，T轴为9。

返回值

以-1～1的实值返回当前功率模式下相对于最大转矩的比例。

正值表示关节角度的正方向；负值表示关节角度的负方向。

参阅

TC、TCSpeed、TCLim

RealTorque函数使用示例

```
Print "当前Z轴转矩指令值(SCARA机器人)：", RealTorque(3)
```

3.21.11 Recover

用于将恢复动作执行到安全门打开时的位置并返回状态。

本命令用于高级人员。请在充分理解命令规格之后使用。

格式

(1) Recover 机器人编号 | All

(2) Recover 机器人编号 | All , WithMove | WithoutMove

参数

机器人编号
　　指定进行恢复动作的机器人编号。
All
　　所有机器人均执行恢复动作。省略时为All。
WithMove
　　是值为0的常数。
　　用于恢复励磁，移动到打开安全门时的位置。省略时为WithMove。
WithoutMove
　　是值为1的常数。
　　仅用于恢复励磁。通常不使用。与AbortMotion组合，用于实现特殊的恢复。

说明

要通过程序执行本命令时，需要勾选Epson RC+的［设置］-［系统配置］-［设置控制器］-［环境］的［将高级任务控制命令设为有效］复选框。

Recover用于在关闭安全门之后再次将电动机设为ON，并以低功率的PTP动作将机器人返回到打开安全门时的位置。恢复动作完成之后，可使用Cont继续进行循环。

在控制器中设为多个机器人并指定All时，按机器人编号从小到大的顺序执行恢复动作。

参阅

AbortMotion、Cont、Recover函数、RecoverPos

Recover使用示例

> ⚠ 注意
>
> **要通过程序执行Recover命令时，请理解命令的规格并确认可作为系统进行恢复动作的条件已经备齐。如果一直采取按循环执行命令等错误的使用方法，则可能会降低系统的安全性。请充分注意。**

```
Function main
    Xqt 2, monitor, NoPause
    Do
        Jump P1
        Jump P2
    Loop
Fend

Function monitor
    Do
        If Sw(SGOpenSwitch) = On then
            Wait Sw(SGOpenSwitch) = Off and Sw(RecoverSwitch) = On
            Recover All
        EndIf
```

```
    Loop
Fend
```

3.21.12 Recover函数

用于将恢复动作执行到安全门打开时的位置并返回状态。

本命令用于高级人员。请在充分理解命令规格之后使用。

格式

(1) Recover

(2) Recover (机器人编号 | All)

(3) Recover (机器人编号 | All , WithMove | WithoutMove)

参数

机器人编号
　　指定进行恢复动作的机器人编号。省略机器人编号时，所有机器人均执行恢复动作。
All
　　所有机器人均执行恢复动作。省略时为All。
WithMove
　　是值为0的常数。
　　用于恢复励磁，移动到打开安全门时的位置。省略时为WithMove。
WithoutMove
　　是值为1的常数。
　　仅用于恢复励磁。通常不使用。与AbortMotion组合，用于实现特殊的恢复。

返回值

返回Boolean型的值。如果完成恢复动作，则会返回“True”。如果未完成，则返回“False”。

说明

要通过程序执行本命令时，需要勾选Epson RC+的［设置］-［系统配置］-［设置控制器］-［环境］的［将高级任务控制命令设为有效］复选框。

Recover用于在关闭安全门之后再次将电动机设为ON，并以低功率的PTP动作将机器人返回到打开安全门时的位置。恢复动作完成之后，可使用Cont继续进行循环。恢复动作顺利完成时返回“True”，如果在恢复动作期间发生暂停、中断或安全门打开，Recover则返回“False”。

在控制器中设为多个机器人并指定All时，按机器人编号从小到大的顺序执行恢复动作。

> ⚠ 注意
>
> **要通过程序执行Recover命令时，请理解命令的规格并确认可作为系统进行恢复动作的条件已经备齐。如果一直采取按循环执行命令等错误的使用方法，则可能会降低系统的安全性。请充分注意。**

参阅

AbortMotion、Cont、Recover、RecoverPos

Recover函数使用示例

```
Boolean sts
Integer answer

sts = Recover
If sts = True Then
MsgBox "Ready to continue", MB_ICONQUESTION + MB_YESNO, "MyProject", answer
If answer = IDYES Then
  Cont
```

```
EndIf
EndIf
```

3.21.13 RecoverPos函数

用于返回安全门打开时的位置。

本命令用于高级人员。请在充分理解命令规格之后使用。

格式
RecoverPos ([机器人编号])

参数

机器人编号
以整数值指定机器人编号。已省略机器人编号时，以当前选择的机器人为对象。

结果
用于返回安全门打开时的位置。

安全门未打开或机器人结束恢复动作时，返回X～W的值中添加0的点数据。

说明
用于返回利用Cont或Recover进行恢复动作时的机器人恢复位置。

参阅
AbortMotion、Cont、Recover、Recover函数、RealPos

RecoverPos函数使用示例
恢复动作的直线距离小于10 mm时进行恢复动作，大于10 mm时结束程序。

```
If Dist(RecoverPos, RealPos) < 10 Then
Recover All
Else
Quit All
EndIf
```

3.21.14 Redim

用于更改运行时间时数组的最大下标。

格式

Redim [Preserve] 数组名(数组变量的最大下标)

参数

Preserve
　　保存数组前的值。可省略，如果省略，数组则被清除。
数组名
　　指定数组变量的名称。按通常的变量名惯例进行指定。必须事先声明数组。
数组变量的最大下标
　　指定数组变量的新的最大下标。请赋予与变量声明时相同数量的最大下标。使用下述格式。
　　(最大下标1, [最大下标2], [最大下标3])
　　由于下标从0开始，因此元素数为最大下标加上1。
　　在所有元素数不超过以下最大值的范围内指定各最大下标。

	String型以外	String型
本地变量	2,000	200
备份变量(Global Preserve)	4,000	400
全局变量和模块变量	100,000	10,000

说明

Redim用于变更运行时数组变量的最大下标。要保存前面的值时，指定Preserve。

不能对Byref指定的数组变量执行Redim。

频繁地执行Redim会降低程序的执行速度。尤其是针对备份变量执行Redim，建议将其控制在所需最低限度。

参阅

UBound

Redim使用示例

```
Integer i, numParts, a(0)

Print "Enter number of parts "
Input numParts

Redim a(numParts)

For i = 0 to UBound(a)
    a(i) = i
Next

'在最大下标上加上20
Redim Preserve a(numParts + 20)

'保持最初的下标
For i = 0 to UBound(a)
    Print a(i)
Next
```

3.21.15 Rename

用于变更文件名。

格式

Rename 变更源文件名, 变更目标文件名

参数

变更源文件名
　　指定要变更名称的文件名和路径字符串表达式。详情请参阅ChDisk。
变更目标文件名
　　将由变更源文件名指定的文件名称指定为新命名的名称。详情请参阅ChDisk。

说明

将由变更源文件名指定的文件名称变更为由变更目标文件名指定的名称。如果省略路径，则该语句从当前目录中查找变更源文件名。

仅在同一驱动器内指定变更源文件名和变更目标文件名时才可执行。

不能变更为与相同路径下存在的其它文件相同的名称。

用作通配符的字符不能用于变更源目录名、变更目标目录名的参数。

参阅

Copy

Rename使用示例

利用命令窗口的执行示例

```
> Rename A.PRG  B.PRG
```

3.21.16 RenDir

用于变更目录名。

格式

Rendir 变更源目录名, 变更目标目录名

参数

变更源目录名
　　以字符串表达式指定要变更的目录路径和目录名。
变更目标目录名
　　以字符串表达式指定变更后的目录路径和目录名。有关路径的详细说明，请参阅ChDisk。

说明

变更目标目录的路径必须存在于变更源目录的路径中。

省略变更源目录名和变更目标目录名的路径，并且仅指定目录名时，是指位于当前目录中的目录。

用作通配符的字符不能用于变更源目录名、变更目标目录名的参数。

注意

可在PC硬盘时执行。

参阅

MkDir

RenDir使用示例

```
RenDir "c:\mydata", "c:\mydata1"
```

3.21.17 Reset

用于将控制器重置为初始状态。

格式

(1) Reset

(2) Reset Error

说明

- Reset用于重置下述项目。
- Reset Error用于结束所有的一般任务，并且仅对错误状态和机器人控制参数进行重置。
- 要通过程序执行Reset Error命令时，需要勾选 Epson RC+的[设置] - [系统配置] - [设置控制器] - [环境]的[将高级任务控制命令设为有效]复选框。
  - 紧急停止状态(仅Reset时)
  - 错误状态
  - 输出位(仅Reset时)：分配给远程输出的I/O和夹具以外的所有输出位，均变为OFF状态。可通过Epson RC+解除该功能。
  - 机器人控制参数
    - Speed和SpeedR、SpeedS的设置值：(被初始化为初始值。)
    - Accel和AccelR、AccelS的设置值：(被初始化为初始值。)
    - QPDecelR、QPDecelS的设置值：(被初始化为初始值。)
    - LimZ参数的设置值：(初始化为0。)
    - CP参数的设置值：(初始化为Off。)
    - SoftCP参数的设置值：(初始化为Off。)
    - Fine的设置：(初始化为初始值。)
    - Power Low设置：(变为低功率模式。)
    - PTPBoost的设置值：(初始化为初始值。)
    - TCLim、TCSpeed的设置值：(初始化为初始值。)
    - PgLSpeed的设置值：(初始化为初始值。)

在伺服相关错误、紧急停止状态或需要其它重置的状态下，不受理Reset以外的命令。此时，请首先执行Reset，然后进行其它必要处理。

比如，紧急停止之后，首先要确认周围环境和操作安全，然后执行Reset。完成之后，请执行Motor On。

不能利用Reset解除严重错误状态。

发生严重错误时，请将控制器电源设为OFF并排除错误原因。

不能通过后台任务或由Trap Emergency、Trap Error启动的任务等执行Reset命令。不能利用程序来解除紧急停止状态。

注意

- [通过Reset将输出端口设为Off]复选框

勾选Epson RC+的[设置] - [系统配置] - [环境]的[通过Reset将输出端口设为OFF]复选框时，如果发出Reset命令，除了设置为夹具的位之外，所有的输出全部变为OFF状态。因该设置而变为输出OFF时，必须要考虑接线状况，以免发生工件掉落或类似的状况。

有关夹具的详细信息，请参阅以下手册。
《Hand功能》

参阅

Accel, AccelS, Fine, LimZ, Motor, Off, On, PTPBoost, SFree, SLock, Speed, SpeedS

Reset使用示例

如下所示为通过命令窗口执行Reset命令的示例。

```
>reset
>
```

3.21.18 ResetElapsedTime

用于重置由ElapsedTime函数使用的计算节拍时间用的计时器。

格式

ResetElapsedTime

说明

重置、重启计算节拍时间用的计时器。

参阅

ElapsedTime函数

ResetElapsedTime使用示例

```
ResetElapsedTime        '重置计算节拍时间用的计时器
For i = 1 To 10         '执行10次
    GoSub Cycle
Next
Print ElapsedTime / 10 '计算并显示节拍时间
```

3.21.19 Restart

用于重新执行当前的主程序。

本命令用于高级人员。请在充分理解命令规格之后使用。

格式
Restart

说明
Restart用于中断正在执行的所有任务，并重新执行最后执行的主程序。

后台任务继续执行，不会中断。

由于所有的Trap设置被解除，因此即使利用本命令中断任务，也不执行Trap Abort。

可通过执行本命令解除Pause状态。

如果在错误状态下执行本命令，则会发生错误。请首先利用Reset Error命令等解除错误。

如果在紧急停止状态下执行本命令，则会发生错误。不能利用程序来解除紧急停止状态。

> ⚠ 注意
> **要通过程序执行Restart命令时，请理解命令的规格并确认可作为系统重新执行的条件已经备齐。如果一直采取按循环执行命令等错误的使用方法，则可能会降低系统的安全性。请充分注意。**

注意

使用远程I/O控制时，请勿同时执行SPEL+程序中Restart命令和远程输入的Start信号。否则会导致程序重复运行并可能发生2503错误。

参阅
Quit、Reset、Trap、Xqt

Restart使用示例

```
Function main
    Trap Error Xqt eTrap
    Motor On
    Call PickPlac
Fend

Function eTrap

Wait Sw(ERresetSwitch)
Reset Error
Wait Sw(RestartSwitch)
Restart
Fend
```

3.21.20 Resume

用于继续执行因Halt命令而暂停的任务。

格式

Resume { 任务识别符 | All }

参数

任务识别符

以整数值或表达式指定任务名或任务编号。任务名为Xqt语句使用的函数名，或通过运行窗口或操作员窗口启动的函数名。

任务编号的指定(整数)

- 一般任务：1～32
- 后台任务：65～80
- Trap任务：257~267

All

继续执行所有任务时指定。

说明

Resume用于继续执行因Halt命令而暂停的任务。

参阅

Halt、Quit、Xqt

Resume使用示例

如下所示为在Halt命令之后使用Resume命令的示例。

```
Function main
   Xqt 2, flicker    '在任务2中执行flicker

   Do
      Wait 3         '执行flicker三秒钟
      Halt flicker   '停止flicker任务
      Wait 3
      Resume flicker '暂停flicker任务
   Loop
Fend

Function flicker
   Do
      On 1
      Wait 0.2
      Off 1
      Wait 0.2
   Loop
Fend
```

3.21.21 Return

组合使用Return语句与GoSub语句。GoSub用于将程序控制移交给子例程。子例程结束时，利用Return在开始子例程的GoSub命令的下一行继续执行程序。

格式

Return

说明

组合使用Return语句与GoSub语句。Return语句的主要作用是将程序控制返回到将控制移交给子例程的GoSub命令的后续命令。

GoSub命令用于将程序控制分支到用户指定的语句行或标签。程序执行移动目标行及其后续行，直至遇到Return命令。Return命令用于将程序控制返回到指示移交给子例程的GoSub的下一行。(也就是说，利用GoSub开始执行子例程，然后利用Return返回到GoSub的下一语句。)

常见错误

- 没有GoSub却使用Return时

  Return命令用于从子例程返回到发出GoSub的源程序。如果在没有GoSub的情况下使用Return，则会发生错误2383。由于不知道系统应返回到何处，因此单独的Return命令没有意义。

参阅

OnErr, GoSub, GoTo

Return使用示例

如下所示为利用GoSub命令分支到checkio标签并检查最初16个用户输入的简单示例。然后从该子例程返回到主程序。

```
Function main
     Integer var1, var2
     GoSub checkio
     On 1
     On 2
     Exit Function

checkio: '子例程的起始位置
     var1 = In(0)
     var2 = In(1)
     If var1 <> 0 Or var2 <> 0 Then
          Print "Message to Operator here"
     EndIf
finished:
     Return  '子例程的结束位置返回到第40行
Fend
```

3.21.22 Right$函数

用于从字符串的最后提取指定数量字符串的函数。

格式
Right$ (字符串, 字符数)

参数

字符串
　　从最后提取指定数量的字符串。以最多255个字符的字符串表达式或直接以字符串进行指定。
字符数
　　以表达式或直接以数值指定要从字符串最后复制的字符数(正整数)。

返回值
从指定字符串的最后提取并返回指定数量的字符串。

说明
Right$用于从指定字符串的最后返回指定数量的字符串。使用Right$仅可直接返回字符串中仅有的字符数。

参阅
Asc、Chr$、InStr、Left$、Len、Mid$、Space$、Str$、Val

Right$使用示例
如下所示为输入工件的数据字符串之后，返回工件的部件编号、名称、部件计数等信息的程序示例。

```
Function SplitPartData(DataIn$ As String, ByRef PartNum$ As String, ByRef PartName$
As String, ByRef PartCount As Integer)

    PartNum$ = Left$(DataIn$, 10)

    DataIn$ = Right$(datain$, Len(DataIn$) - pos)
    pos = Instr(DataIn$, ",")

    PartName$ = Mid$(DataIn$, 11, 10)

    PartCount = Val(Right$(dataIn$, 5))

Fend
```

另外，如下所示为利用命令窗口的操作示例。

```
> Print Right$("ABCDEFG", 2)
 FG

> Print Right$("ABC", 3)
 ABC
```

3.21.23 RmDir

用于删除子目录。

格式

Rmdir 目录名

参数

目录名

以字符串表达式指定要删除的目录路径和目录名。省略路径并且仅指定目录名时，是指当前目录中的子目录。有关路径的详细说明，请参阅ChDisk。

说明

用于删除指定的子目录。执行该语句之前，必须事先删除子目录内的文件。

不能删除当前目录或父目录。

如果通过命令窗口执行，则可省略引号。

注意

可在PC硬盘时执行。

Rmdir使用示例

利用命令窗口的执行示例

```
> RmDir \mydata
```

3.21.24 Rnd函数

用于返回随机数。

格式
Rnd (最大值)

参数

最大值
以实值设置最大返回值。

返回值
返回0～由参数指定的最大值之间的实值随机数。

说明
用于生成随机数。

参阅
Int, Randomize

Rnd函数使用示例
如下所示为生成1～10之间随机数的示例。

```
Function main
    Real r
    Integer randNum

    Randomize
    randNum = Int(Rnd(9)) + 1
    Print "Random number is:", randNum
Fend
```

3.21.25 Robot

用于选择机器人。

格式

Robot 机器人编号

参数

机器人编号
　　指定机器人编号。范围为1～设置的机器人数。

说明

Robot语句用于选择接下来执行动作命令的机器人。

1台机器人时，无需使用Robot语句。

参阅

Accel、AccelS、Arm、ArmSet、Go、Hofs、Home、HOrdr、Local、Move、Pulse、Robot函数、Speed、SpeedS

Robot使用示例

```
Function main
    Integer I
    For I = 1 to 100
        Robot 1
        Go P(i)
        Robot 2
        Go P(i)
    Next I
Fend
```

3.21.26 Robot函数

用于返回当前的机器人编号。

格式
Robot

返回值
返回当前机器人编号的整数值。

参阅
Robot

Robot函数使用示例

```
Print "The current robot is: ", Robot
```

3.21.27 RobotInfo函数

用于返回机器人的状态信息。

格式

RobotInfo (索引)

参数

索引

以整数值指定要检索的信息索引。

返回值

返回已指定信息的整数值。

说明

下表所示为返回值的位信息。

索引	位	值	说明
0	0	&H1	未定义
	1	&H2	发生可重置的错误
	2	&H4	发生不可重置的错误
	3	&H8	电动机ON
	4	&H10	功率High
	5	&H20	未定义
	6	&H40	未定义
	7	&H80	未定义
	8	&H100	机器人处于Halt状态
	9	&H200	机器人未处于Halt状态(动作期间或快速暂停期间)
	10	&H400	因暂停或安全门而停止机器人
	11		未定义
	12		未定义
	13		未定义
	14	&H4000	满足动作命令之后的TILL条件
	15	&H8000	满足动作命令之后的SENSE条件
	16～31		未定义
1	0	&H1	跟踪动作期间(传送带跟踪期间)
	1	&H2	等待恢复动作(WaitRecover状态)
	2	&H4	正在执行恢复动作
	3～31		未定义
2	0	&H1	机器人处于原点位置
	1～31		未定义

索引	位	值	说明
3	0	&H1	正进行第1关节伺服励磁
	1	&H2	正进行第2关节伺服励磁
	2	&H4	正进行第3关节伺服励磁
	3	&H8	正进行第4关节伺服励磁
	4	&H10	正进行第5关节伺服励磁
	5	&H20	正进行第6关节伺服励磁
	6	&H40	正进行第7关节伺服励磁
	7	&H80	正进行S关节伺服励磁
	8	&H100	正进行T关节伺服励磁
	9～31		未定义
4	N/A	0～32 -1	执行机器人命令的任务编号 ■ 0 = 从命令窗口或宏执行命令 ■ -1 = 任务未使用机械手
5	0	&H1	第1关节制动器ON
	1	&H2	第2关节制动器ON
	2	&H4	第3关节制动器ON
	3	&H8	第4关节制动器ON
	4	&H10	第5关节制动器ON
	5	&H20	第6关节制动器ON
	6	&H40	第7关节制动器ON
	7	&H80	S关节制动器ON
	8	&H100	T关节制动器ON
	9～31		未定义

参阅

CtrlInfo、RobotInfo$、TaskInfo

RobotInfo函数使用示例

```
If (RobotInfo(3) And &H1) = &H1 Then
  Print "Joint 1 is locked"
Else
  Print "Joint 1 is free"
EndIf
```

3.21.28 RobotInfo$函数

用于返回机器人的文本信息。

格式

RobotInfo$ (索引)

参数

索引
以整数值指定要检索的信息索引。

返回值

返回已指定信息的字符串。

说明

- 0：机器人名
- 1：型号名称
- 2：默认点文件名
- 3：未定义
- 4：机器人的序列号

参阅

CtrlInfo、RobotInfo、TaskInfo

RobotInfo$函数使用示例

```
Print "Robot Name: ", RobotInfo$(0)
```

3.21.29 RobotModel$函数

用于返回机器人的型号名称。

格式

RobotModel$

返回值

返回型号名称的字符串。这是记载在机器人后侧面板上的机器人名称。

参阅

RobotType

RobotModel$函数使用示例

```
Print "The robot model is ", RobotModel$
```

3.21.30 RobotName$函数

用于返回机器人名称。

格式
RobotName$

返回值
以字符串返回机器人名称。

参阅
RobotInfo、RobotModel$

RobotName$函数使用示例

```
Print "The robot name is ", RobotName$
```

3.21.31 RobotSerial$函数

用于返回机器人的序列号。

格式
RobotSerial$

返回值
以字符串返回机器人的序列号。

参阅
RobotInfo、RobotName$、RobotModel$

RobotSerial$函数使用示例

```
Print "The robot serial number is ", RobotSerial$
```

3.21.32 RobotType函数

用于返回机器人类型。

格式

RobotType

返回值

- 1：关节型
- 2：直角坐标型
- 3：水平多关节型
- 5：垂直6轴型
- 6：RS系列
- 7：N系列

参阅

RobotModel$

RobotType函数使用示例

```
If RobotType = 3 Then
  Print "Robot type is SCARA"
EndIf
```

3.21.33 ROpen

用于在读取专用模式下打开文件。

格式

ROpen 文件名As #文件编号 . . Close #文件编号

参数

文件名
　　指定包括路径的文件名字符串。仅指定文件名时，是指当前目录中的文件。详情请参阅ChDisk。
文件编号
　　以30～63之间的整数值或表达式进行指定。

说明

以指定的文件编号打开指定的文件。该语句用于从指定的文件中读出数据。

注意

- 仅PC硬盘
- 可使用网络路径。

指定的文件编号用于在打开文件期间识别该文件。因此，在关闭该文件之前，不能使用与其它文件相同的文件编号。按文件操作命令(Input#、Read、Seek、Eof、Close)使用文件编号。

利用Close语句关闭文件并释放文件编号。

请利用FreeFile函数获取文件编号，以免在多个任务中使用同一编号。

参阅

Close, Input #, AOpen, BOpen, UOpen, WOpen, FreeFile

ROpen使用示例

```
Integer fileNum, i, j

fileNum = FreeFile
WOpen "TEST.DAT" As #fileNum
For i = 0 To 100
    Print #fileNum, i
Next i
Close #fileNum

fileNum = FreeFile
ROpen "TEST.DAT" As #fileNum
For i = 0 to 100
    Input #fileNum, j
    Print "data = ", j
Next i
Close #fileNum
```

3.21.34 ROTOK函数

向目标坐标发出动作命令时，用于返回可否附加ROT修饰参数。

格式

ROTOK (目标坐标)

ROTOK (基准坐标, 目标坐标)

参数

目标坐标
　　利用点数据指定用于调查可否附加ROT修饰参数的目标坐标。
基准坐标
　　利用点数据指定用于调查可否附加ROT修饰参数的基准坐标。

返回值

如果可附加ROT修饰参数，则返回“True”；如果不是，则返回“False”。

注意

- 支持的控制器型号

  不支持T/VT系列。

说明

实际进行机器人动作之前确认可否附加ROT修饰参数。

ROT修饰参数是指可附加到Move等直线插补动作命令中并指定以工具姿势变化的加减速为优先的参数。

要利用Move等直线插补动作命令从基准坐标移动到目标坐标时，可以在动作之前通过ROTOK函数判定附加ROT之后是否会发生错误。

省略基准坐标时，以当前位置 (Here) 为基准返回判定结果。

姿势变化角度为“0”或发生微小变化时，判定附加ROT修饰参数之后会发生的错误。但动作期间机器人的关节速度或关节加速度是否超出机械手的限度——如4242错误等——则无法判定。在这种情况下，请进行诸如降低SpeedS、SpeedR、AccelS、AccelR值等的调整。

参阅

Move

ROTOK函数使用示例

```
If ROTOK(P1) = True Then
  Move P1 ROT
Else
  Move P1
EndIf
```

3.21.35 RSet$函数

用于返回在字符串的开头添加空格以形成指定长度的字符串。

格式

RSet$ (字符串, 字符串的长度)

参数

字符串
　　指定字符串表达式。
字符串的长度
　　以整数或表达式指定返回字符串的长度。

返回值

返回在开头添加空格的指定字符串。

参阅

LSet$、Space$

RSet$函数使用示例

```
temp$ = "123"
temp$ = RSet$(temp$, 10)   ' temp$ = "       123"
```

3.21.36 RShift函数

用于数值数据的逻辑右移

格式
RShift (数值数据, 移位数)

参数

数值数据
　　以表达式或直接以数值指定要进行逻辑移位的数值。
移位数
　　指定进行右逻辑移位的位数值(0～31的整数值)。

返回值
返回将指定数值数据进行右逻辑移位的值。

说明
RShift用于将指定数值数据向右(低位方向)进行指定位数的移位。通常，移位部分的高位被设为0 。

作为最简单的说明，Rshift用于返回数值数据除以2的移位乘积的数。

注意

- 数值数据类型

  数值数据如为有效的数值数据类型，则可以返回任何数据类型。

  RShift对应于下述数据类型。
  Byte型、Double型、Int32型、Integer型、Long型、Real型、Short型、UByte型、UInt32型、UShort型

参阅
And、LShift、LShift64、Not、Or、RShift64、Xor

RShift使用示例
如下所示为对于从“0”下开始的Integer型数值数据，表示所有Rshift值的程序示例。

```
Function rshiftst
  Integer num, snum, i
  num = 32767
  For i = 1 to 16
     Print "i =", i
     snum = RShift(num, i)
     Print "RShift(32767, ", i, ") = ", snum
  Next i
Fend
```

如下所示为利用命令窗口操作Rshift命令的示例。

```
> Print RShift(10,1)
5
> Print RShift(8,3)
1
> Print RShift(16,2)
4
```

3.21.37 RShift64函数

数值数据的逻辑右移

格式
RShift64 (数值数据, 移位数)

参数

数值数据
　　以表达式或直接以数值指定要进行逻辑移位的数值。
移位数
　　指定进行右逻辑移位的位数值(0～63的整数值)。

返回值
返回将指定数值数据进行右逻辑移位的值。

说明
RShift64用于将指定数值数据向右(低位方向)进行指定位数的移位。通常，移位部分的高位被设为0。

作为最简单的说明，Rshift64用于返回数值数据除以2的移位乘积的数。

注意

- 数值数据类型

  包括有多种数值类型。Rshift64可以使用Int64型、UInt64型的数值。

参阅
And, LShift, LShift64, Not, Or, RShift, Xor

RShift64使用示例
如下所示为对于从“0”下开始的UInt64型数值数据，表示所有Rshift64值的程序示例。

```
Function rshif64tst
  UInt64 num, snum, i
  num = 18446744073709551615
  For i = 1 to 63
    Print "i =", i
    snum = RShift64(num, i)
    Print "RShift64(18446744073709551615, ", i, ") = ", snum
  Next i
Fend
```

如下所示为利用命令窗口操作Rshift64命令的示例。

```
> Print RShift64(10,1)
5
> Print RShift64(8,3)
1
> Print RShift64(16,2)
4
```

3.21.38 RTrim$函数

用于删除字符串右侧空格并进行返回。

格式

RTrim$ (字符串)

参数

字符串
以字符串表达式或直接以字符串进行指定。

返回值

删除右侧空格的字符串

参阅

LTrim$、Trim$

RTrim$函数使用示例

```
str$ = "  data  "
str$ = RTrim$(str$)  ' str$ = "  data"
```

3.21.39 RunDialog

用于启动通过运行SPEL+程序的Epson RC+画面。

格式

(1) RunDialog 对话框ID

(2) RunDialog DLG_ROBOTMNG [, 机器人选择位]

参数

对话框ID

指定包括有效对话框ID在内的整数值。事先由下述常数定义这些值。

- DLG_ROBOTMNG: 100：启动机器人管理器对话框
- DLG_IOMON: 102：启动I/O监控器
- DLG_VGUIDE: 110：启动Vision Guide对话框

机器人选择位

仅在对话框ID中指定DLG_ROBOTMNG时有效。以位值指定可由机器人管理器选择的机器人。

设置示例	设置值	bit15	bit14	...	bit2	bit1	bit0
机器人1	&H0001	Off	Off		Off	Off	On
机器人2	&H0002	Off	Off		Off	On	Off
机器人1和2	&H0003	Off	Off		Off	On	On
:							
机器人16	&H1000	On	Off		Off	Off	Off

说明

如要通过SPEL+的任务启动/显示Epson RC+画面，请使用RunDialog。关闭画面之前，任务保持暂停状态。

要通过Epson RC+画面执行机器人命令时，请确认在显示该画面期间没有其它机器人控制任务。如果其它机器人控制任务启动，则会发生错误。

参阅

InputBox、MsgBox

RunDialog使用示例

```
If Motor = Off Then
    RunDialog DLG_ROBOTMNG
    If Motor = Off Then
        Print "Motors are off, aborting program"
        Quit All
    EndIf
EndIf
```

3.22 S

3.22.1 SafetyOn函数

用于返回安全门的状态。

格式
SafetyOn

返回值
如果安全门处于打开状态，则返回“True”；如果不是，则返回“False”。

说明
本函数仅用于NoPause任务、NoEmgAbort任务(执行Xqt时指定NoPause或NoEmgAbort以开始的特别任务)与后台任务。

参阅
ErrorOn、EstopOn、PauseOn、Wait、Xqt

SafetyOn函数使用示例
下例所示为监视控制器安全门打开状态，并在安全门打开时对I/O进行ON/OFF操作的程序。

注意

- Forced标志

  本程序示例所示为在On/Off命令中指定的Forced标志。

  发生错误、紧急停止期间或安全门打开时，I/O输出会发生变化，因此，在系统设计方面需要注意。

```
Function main

  Xqt SafetyOnOffMonitor, NoPause
  :
  :
Fend

Function SafetyOnOffMonitor
  Do
    Wait SafetyOn = True
    Print "Saftey Open"
    Off 10, Forced
    On 12, Forced

    Wait SafetyOn = False
    Print "Saftey Close"
    On 10, Forced
    Off 12, Forced
  Loop
Fend
```

3.22.2 SavePoints

用于将主存储器中的点数据保存到当前机器人的Disk文件中。

格式

SavePoints 文件名

参数

文件名

以字符串表达式指定保存点数据的目标文件名。扩展名固定为“.pts”。不能指定路径。另外，不受ChDisk等的影响。详情请参阅ChDisk。

说明

SavePoints用于将点保存到指定的文件中。扩展名固定为“.pts”。省略扩展名时，添加“.pts”名。

另外，点文件不存在时，SavePoints命令则将其添加到当前机器人的项目中。

点数据被保存在控制器内的小型闪存卡中。如果执行SavePoints，则会向小型闪存卡执行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议仅在需要保存点数据时执行SavePoints。

常见错误

- 超出硬盘容量时

  硬盘没有容量时，会发生错误。

- 指定文件不是当前机器人的文件时

  如果在文件名中指定其它机器人的点文件，则会发生错误。

- 找不到指定文件时

  如果文件名中包括路径，则会发生错误。仅可指定当前项目的文件名。

- 文件名错误

  文件名内包括空格或无效字符时，会发生错误。

参阅

ImportPoints、LoadPoints

SavePoints使用示例

```
ClearPoints
For i = 1 To 10
   P(i) = XY( i, 100, 0, 0 )
Next i
SavePoints "TEST.PTS"
```

3.22.3 Seek

用于变更指定文件的文件指针位置。

格式

Seek #文件编号, 指针

参数

文件编号
　　以30～63之间的整数值或表达式进行指定。
指针
　　以整数值或表达式指定0～文件大小之间的文件指针。

参阅

BOpen、Read、ROpen、UOpen、Write、WOpen

Seek使用示例

```
Integer fileNum
String data$

fileNumber = FreeFile
UOpen "TEST.DAT" As #fileNum
Seek #fileNum, 20
Read #fileNum, data$, 2
Close #fileNum
```

3.22.4 Select...Send

用于根据表达式的值将控制移交给几个语句中的某个语句。

格式

Select式 Case项目 语句 [Case 项目 语句] [Default 语句] Send

参数

式
　指定数值或字符串表达式。
项目
　指定类型与表达式一致的数值或字符串表达式。
语句
　指定1个或多个有效的SPEL+语句或多语句。

说明

如果Case语句项目中存在与Select语句表达式结果一致的内容，则执行最初一致的Case语句后的语句群。执行之后，程序控制将移交给Send语句的下一语句。

如果Case语句项目中不存在与Select语句表达式结果一致的内容，则执行Default语句，并将程序控制移交给Send语句的下一语句。

如果Case语句项目中没有与Select语句表达式结果一致的内容，并且省略Default，则不进行任何执行，将程序控制移交给Send语句的下一语句。

可在Select语句表达式中指定常数、变量以及And、Or、Xor等的运算符。也可在Case语句项目中指定常数、变量以及And、Or、Xor等的运算符。在这种情况下，将Case语句项目的运算结果与Select语句表达式进行比较。另外，因为动作将变得复杂，请勿在Case语句项目中指定变量。

参阅

If…Then…Else

Select...Send使用示例

如下所示为简单的Select...Send示例。

```
Function Main
  Integer I
  For i = 0 To 10
    Select I
      Case 0
           Off 1;On 2;Jump P1
      Case 3
           On 1;Off 2
           Jump P2;Move P3;On 3
      Case 7
           On 4
      Default
           On 7
    Send
  Next
Fend
```

3.22.5 SelectDB

用作从打开数据库内的表格中检索数据的函数。

格式

SelectDB (#数据库编号, 表格名, Select条件, Sort方法)

参数

数据库编号
　　指定利用OpenDB指定的数据库编号(501～508的整数值)。
表格名
　　指定要进行数据检索的表格名。由文件编号指定的数据库类型为Excel工作簿时，指定Excel表单或加名的表格。指定Excel表单时，在表单名之后附加 $ 并用 [] 括起。指定由Excel表单内的名称指定的区域时，用[]括起名称。
Select条件
　　指定检索条件。可使用AND、OR指定复合条件。未指定检索条件时，检索表格中的所有数据。
Sort方法
　　指定提取已检索数据的顺序。指定Sort键和Sort顺序(升序 [ASC] /降序 [DESC])。省略Sort顺序时，说明已指定Sort键的升序。省略Sort方法时，根据已打开的数据库确定顺序。

返回值

返回检索列的总数。

说明

根据Sort条件，从已打开数据库的指定表格中对符合Select条件的数据进行分类(Sort)。

务必在利用Input#、Print#读入、写入数据之前执行。

已打开的数据库为Excel工作簿时，请在由表单和名称定义的区域的第1行记述用于检索的列名。

另外，为Excel 2007工作簿时，请指定表单名。不能访问由名称定义的区域。

注意

- 需要连接已安装RC+的PC。

参阅

OpenDB、CloseDB、UpdateDB、DeleteDB、Input #、Print #

SelectDB函数使用示例

SQL数据库的使用示例

如下所示为利用SQL服务器2000样本数据库Northwind的表格Employees或TitleOfCourtesy，按EmployeeID的降序读入Ms.数据的简单示例。

```
Integer count, i, eid
String Lastname$, Firstname$, Title$

OpenDB #501, SQL, "(LOCAL)", "Northwind"
count = SelectDB(#501, "Employees", "TitleOfCourtesy = 'Ms.'", "EmployeeID DESC")
For i = 0 To count - 1
  Input #501, eid, Lastname$, Firstname$, Title$
  Print eid, ",", Lastname$, ",", Firstname$, ",", Title$
Next
CloseDB #501
```

Access数据库的使用示例

如下所示为从Microsoft Access 2007样本数据库学生名册的学生表格中，按ID的升序读入职务为组长的数据的简单示例。

```
Integer count, i, eid
String Lastname$, Firstname$, dummy$

OpenDB #502, Access, "c:\MyDataBase\学生名册.accdb"
count = SelectDB(#502, "学生", "职务 = '组长'", "ID")
For i = 0 To count - 1
  Input #502, eid, dummy$, dummy$, Lastname$, dummy$, Firstname$
  Print eid, ",", Lastname$, ",", Firstname$
Next
CloseDB #502
```

Excel工作簿的使用示例

如下所示为从Microsoft Excel工作簿学生名册的学生表格中，按ID的升序读入年龄不满25岁的数据的简单示例。

```
Integer count, i, eid
String Lastname$, Firstname$

OpenDB #503, Excel, "c:\MyDataBase\学生名册.xls"
count = SelectDB(#503, "[学生$]", "Age < 25", "ID ASC")
For i = 0 To count - 1
  Input #503, eid, Lastname$, Firstname$
  Print eid, ",", Lastname$, ",", Firstname$
Next
CloseDB #503
```

3.22.6 Sense

用于设置/显示利用Jump、Jump3、Jump3CP指定Sense时，停在目标坐标上方的条件。

格式

Sense [条件表达式]

参数

事件条件表达式

指定触发的输入状态。
[条件] 比较运算符(=、<>、>=、>、<、<=)[整数表达式]
可在条件中使用下述函数或变量。

- 函数：Sw, In, InW, Oport, Out, OutW, MemSw, MemIn, MemInW, Ctr, GetRobotInsideBox, GetRobotInsidePlane, AIO_In, AIO_InW, AIO_Out, AIO_OutW, Hand_On, Hand_Off, SF_GetStatus
- 变量：Byte, Int32, Integer, Long, Short, UByte, UInt32, UShort型备份变量、全局变量、模块变量

另外，可利用下述运算符对多个事件条件表达式附加掩码或进行复合组合。

- 运算符：And、Or、Xor

[例]

```
Sense Sw(5) = On
Sense Sw(5) = On And Sw(6) = Off
```

说明

Sense用于在执行Jump命令期间按第3关节下降开始前的时序检查输入条件。

另外，也用于在执行Jump3、Jump3CP命令期间按接近动作即将开始的时序检查输入条件。

Sense条件表达式必须包含1个以上的上述函数。

Sense条件表达式中包括变量时，在设置Sense条件时运算其值。由于可能会形成不希望有的条件，因此不建议在条件表达式中使用变量。可使用多个Sense语句。切换到下一Sense语句之前，最后执行的输入条件有效。

- Jump和Sense修饰符

  检查当前的Sense条件是否成立。如果成立，则在机器人停在目标坐标上方的状态下完成Jump命令。也就是说，Sense条件为“True”时，机器人在目标坐标的上方并且第3关节即将开始下降的状态下停止。Sense条件为“False”时，机器人在目标坐标上完成Jump命令执行的动作。

- Jump3、Jump3CP和Sense修饰符

  检查当前的Sense条件是否成立。如果成立，则在机器人停在接近起始位置的状态下完成Jump3、Jump3CP命令。

  如果省略参数，则显示当前的Sense设置。

注意

- 电源ON时的Sense设置

  电源ON时Sense条件的初始设置为Sense Sw(0)= On。输入位编号0为ON时，设为机器人不进行下降动作。

- 用于检查Sense条件成立的JS函数和State函数

  执行使用Sense修饰符的动作命令之后，可使用JS或State函数检查Sense条件是否成立。

- 在条件表达式中使用变量时

- 可使用变量类型为整数型(Byte、Int32、Integer、Long、Short、UByte、UInt32、UShort)。
- 不能使用数组变量。
- 不能使用本地变量。
- 在超过0.01秒的时间内变量值未满足条件时，系统可能不能检测到变量变化。
- 系统内可使用的变量等待数存在限制。1个系统内可使用的变量等待数量最多为64个(也包括在Wait等条件表达式中使用的变量等待)。如果超过最大数，则会在项目创建时发生错误。
- 如果利用Byref引用执行变量等待的变量，则会发生错误。
- 条件表达式右边的整数表达式中包括变量时，在动作命令开始时运算其值。由于可能会形成不希望有的条件，因此不建议在整数表达式中使用变量。

参阅

In、JS、Jump、Jump3、Jump3CP、MemIn、MemSw、Stat、Sw、SF_GetStatus

Sense使用示例

如下所示为Sense命令的简单使用示例。

```
Function test
  .
  .
TrySense:
    Sense Sw(1) = Off    '设为在输入为1为Off时停在目标坐标上方
    Jump P1 C2 Sense
    If JS = True Then
         GoSub ERRPRC    '机械臂停在目标坐标上方时
         GoTo TrySense   '执行ERRPRC，移动到TrySense
    EndIf
    On 1; Wait 0.2; Off 1
  .
  .
Fend
```

[其它格式示例]

```
> Sense Sw(1)=1 And MemSw(1)=1

> Sense Sw(0) Or (Sw(1) And MemSw(1))
```

3.22.7 SetCom

用于设置/显示RS-232C端口参数。

格式

SetCom #通信端口编号 [, 通信速度] [, 数据位长度] [, 停止位长度] [, 奇偶性] [, 收发行末] [, H/W流控制] [, S/W流控制] [, 超时时间]

参数

通信端口编号
以整数值指定RS-232C的端口编号。
- SPEL+ 控制部分：1～8
- PC部分：1001~1008

通信速度
指定波特率。如下所示为有效值。可省略。(默认值：9600)
- 110
- 300
- 600
- 1200
- 2400
- 4800
- 9600
- 14400
- 19200
- 38400
- 57600
- 115200

使用PC部分的端口时，如果通信速度大于等于19200，数据接收则可能会遗漏。

数据位长度
以7或8的数值指定每个字符的数据位长度。可省略。

停止位长度
以1或2的数值指定每个字符的停止位长度。可省略。

奇偶性
指定奇偶性。奇数时指定O，偶数时指定E，没有时指定N。可省略。

收发行末
指定CR、LF、CRLF中某个收发行末。可省略。

H/W流控制
硬件控制有效时指定RTS，无效时指定NONE。可省略。

S/W流控制
软件控制有效时指定XON，无效时指定NONE。可省略。

超时时间
以表达式或数值指定超时时间(正实值，单位：秒)。如果指定0，超时则变为无限。可省略。

说明

如果省略所有参数，则显示通信端口的设置。

如果将多个端口设为19200以上的通信速度并同时通信，则可能会发生错误2929或2922。此时，请选择较慢的传输速度或不同时进行通信。

使用PC部分的端口时，如果通信速度大于19200，数据接收则可能会遗漏。

发生数据接收遗漏时，请选择较慢的传输速度，或使用SPEL+控制部分的端口。

参数被保存在控制器内的小型闪存卡中。如果执行SetCom，则会向小型闪存卡执行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议仅在需要变更参数时执行SetCom。

参阅

OpenCom、CloseCom、SetNet

SetCom使用示例

```
SetCom #1, 9600, 8, 1, N, CRLF, NONE, NONE, 0

SetCom #2, 4800
```

3.22.8 SetLatch

用于设置通过R-I/O输入实现机器人位置闩锁的功能。

格式

SetLatch {端口编号, 输入逻辑 , 连续闩锁次数}

参数

端口编号

指定连接触发输入信号的R-I/O输入端口的端口编号。
如下所示为可指定的端口编号。指定连接对象机器人的单元的端口编号。

		点	端口编号
控制单元	输入	2点	24.25
	输出	-	-
驱动单元1	输入	2点	56.57
	输出	-	-
驱动单元2	输入	2点	280,281
	输出	-	-

作为端口编号，定义以下常数。

常数	端口编号
SETLATCH_PORT_CU_0	24
SETLATCH_PORT_CU_1	25
SETLATCH_PORT_DU1_0	56
SETLATCH_PORT_DU1_1	57
SETLATCH_PORT_DU2_0	280
SETLATCH_PORT_DU2_1	281

输入逻辑

指定连接到R-I/O上的触发输入信号的逻辑。可利用以下常数指定逻辑。

常数	值	含义
SETLATCH_TRIGGERMODE_TRAILINGEDGE	0	负逻辑
SETLATCH_TRIGGERMODE_LEADINGEDGE	1	正逻辑

负逻辑时，按输入信号从High到Low的切换边沿闩锁机器人位置。正逻辑时，按输入信号从Low到High的切换边沿闩锁机器人位置。

连续闩锁次数

通过R-I/O输入信号指定机器人位置的连续闩锁次数。可以指定1,2,3,4。LatchEnable On后，可以将指定连续闩锁次数的点数据进行闩锁。最多可闩锁4次。参数可以省略。如省略则默认闩锁次数为1。

说明

用于设置通过R-I/O输入信号实现机器人位置闩锁的条件。1台机器人不能同时等待多个端口的触发信号。执行SetLatch约需40 msec的处理时间。

注意

如果指定与所选机器人无关的其它单元的端口编号，则会发生超出参数范围错误。

参阅

LatchEnable、LatchState函数、LatchPos函数

SetLatch使用示例

```
Function main
  SetLatch 24, SETLATCH_TRIGGERMODE_LEADINGEDGE, 4
'正逻辑  设定连续闩锁4次
  LatchEnable On    '闩锁功能有效
  Go P1
  Wait LatchState = True     '等待触发
  Print LatchPos(WithoutToolArm, 1)     '显示闩锁位置1
  Print LatchPos(WithoutToolArm, 2)     '显示闩锁位置2
  Print LatchPos(WithoutToolArm, 3)     '显示闩锁位置3
  Print LatchPos(WithoutToolArm, 4)     '显示闩锁位置4
  LatchEnable Off     '闩锁功能无效
Fend
```

省略参数时的使用示例：

```
Function main
  SetLatch 24, SETLATCH_TRIGGERMODE_LEADINGEDGE    '正逻辑
  LatchEnable On    '闩锁功能有效
  Go P1
  Wait LatchState = True     '等待触发
  Print LatchPos     '显示闩锁位置
  LatchEnable Off     '闩锁功能无效
Fend
```

3.22.9 SetIn

用于设置虚拟I/O的输入端口(8位)。

格式

SetIn 端口编号, 设置值

参数

端口编号
　　指定I/O的输入字节。
设置值
　　以0～255范围的整数指定端口编号。

说明

虚拟I/O有效时，同时设置8个输入位。

虚拟I/O无效时，本命令会发生错误。

参阅

SetInReal, SetSW, SetInW

SetIn使用示例

```
> setin 0, 1  '将端口0的最初位设为ON
```

3.22.10 SetInReal

用于设置虚拟I/O的输入端口(32位)。

格式

SetInReal 端口编号, 设置值

参数

端口编号
　　指定I/O的输入字。
设置值
　　指定Real类型的值。

说明

虚拟I/O有效时，同时设置32个输入位。
虚拟I/O无效时，本命令会发生错误。

参阅

SetSw, SetIn, SOetInW

SetIn使用示例

```
> setinReal 32, 2.543 '设置为输入端口32
```

3.22.11 SetInW

用于设置虚拟I/O的输入端口(16位)。

格式
SetInW 端口编号, 设置值

参数

端口编号
　　指定I/O的输入字。
设置值
　　以0～65535范围的整数指定字。

注意

- 关于实时I/O的输入位等字端口编号的规则

  实时I/O的输入位不被反映。

  实时I/O的输入位等字端口=1、3、17、19时，请将设置值指定在0～255的整数范围之内。指定的值大于255时，将出现错误。

说明
虚拟I/O有效时，同时设置16个输入位。

虚拟I/O无效时，本命令会发生错误。

参阅
SetInReal, SetSw, SetIn

SetInW使用示例

```
> setinw 0, 1   '将字0的最初位设为ON
```

3.22.12 SetNet

用于设置TCP/IP端口参数。

格式

(1) SetNet #通信端口编号, 主机地址 [, TCP/IP端口编号 [, 终止符 [, 流控制 [, 超时时间 [, 通信协议 [, CloseNet超时时间]]]]]]

(2) SetNet

参数

通信端口编号
　　指定要设置参数的TCP/IP的端口编号。范围为201～216。
主机地址
　　指定主机的IP地址。
TCP/IP端口编号
　　指定TCP/IP端口编号。
终止符
　　指定CR、LF、CRLF中某个行末字符。
流控制
　　是指软件流控制。指定NONE。
超时时间
　　以秒指定收发的最长时间。指定0时，超时则变为无限。
协议
　　指定通信的协议(TCP/UDP/UDP_SEND/UDP_RECV)类型。
- TCP：TCP通信
- UDP：UDP通信
- UDP_SEND：UDP发送
- UDP_RECV：UDP接收

CloseNet超时时间
　　指定使用CloseNet关闭套接字之前的时间，以秒为单位。(整数: 0 - 5)
　　指定0以关闭套接字而不等待对关闭请求的相应。

说明

参数被保存在控制器内的小型闪存卡中。如果执行SetNet，则会向小型闪存卡执行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议仅在需要变更参数时执行SetNet。

参阅

OpenNet、WaitNet、CloseNet、SetCom

SetNet使用示例

```
SetNet #201, "192.168.0.1", 2001, CRLF, NONE, 0, TCP, 5
```

3.22.13 SetSw

用于设置虚拟I/O的输入。

格式
SetSw 位编号, 设置值

参数

位编号
　　指定I/O的输入位。
设置值
　　指定0(OFF)或1(ON)的整数。

说明
虚拟I/O有效时，设置输入位。

虚拟I/O无效时，本命令会发生错误。

参阅
SetInReal, SetIn, SetInW

SetSw使用示例

```
> setsw 2, on  '将第2个输入位设为ON
```

3.22.14 SF_GetParam函数

返回安全功能参数的信息。

格式
SF_GetParam (索引)

参数

索引
以整数值或常数指定要检索的信息索引。常数末尾指定了“_EN”索引时，启用时返回1，停用时返回0。

返回值
返回指定的信息的整数值。

说明
返回指定的安全功能参数值。

索引	常数	说明
1	DRYRUNOFF	空运行 禁用状态
2	SLS_1_HAND_EN	SLS_1的夹具速度监控状态
3	SLS_1_SPEED	SLS_1的监控速度设置值
4	SLS_1_ELBOW_EN	SLS_1的肘部(水平多关节型: J2, 垂直6轴型: J3) 速度监控状态 *1
5	SLS_1_JOINT_EN	SLS_1的关节速度监控状态
6	SLS_1_JOINTSPEED	SLS_1的监控关节速度设置值
7	SLS_1_WRIST_EN	SLS_1的手腕(垂直6轴型: J5)速度监控状态 *1
8	SLS_1_SHOULDER_EN	SLS_1的肩部(垂直6轴型: J2)速度监控状态 *1
9	SLS_2_HAND_EN	SLS_2的夹具速度监控状态
10	SLS_2_SPEED	SLS_2的监控速度设置值
11	SLS_2_ELBOW_EN	SLS_2的肘部(水平多关节型: J2, 垂直6轴型: J3) 速度监控状态 *1
12	SLS_2_JOINT_EN	SLS_2的关节速度监控状态
13	SLS_2_JOINTSPEED	SLS_2的监控关节速度设置值
14	SLS_2_WRIST_EN	SLS_2的手腕(垂直6轴型: J5)速度监控状态 *1
15	SLS_2_SHOULDER_EN	SLS_2的肩部(垂直6轴型: J2)速度监控状态 *1
16	SLS_3_HAND_EN	SLS_3的夹具速度监控状态
17	SLS_3_SPEED	SLS_3的监控速度设置值
18	SLS_3_ELBOW_EN	SLS_3的肘部(水平多关节型: J2, 垂直6轴型: J3) 速度监控状态 *1
19	SLS_3_JOINT_EN	SLS_3的关节速度监控状态
20	SLS_3_JOINTSPEED	SLS_3的监控关节速度设置值
21	SLS_3_WRIST_EN	SLS_3的手腕(垂直6轴型: J5)速度监控状态 *1

索引	常数	说明
22	SLS_3_SHOULDER_EN	SLS_3的肩部(垂直6轴型: J2)速度监控状态 *1
23	SLS_T2_HAND_EN	SLS_T2的夹具速度监控状态
24	SLS_T2_SPEED	SLS_T2的监控速度设置值
25	SLS_T2_ELBOW_EN	SLS_T2的肘部(水平多关节型: J2, 垂直6轴型: J3) 速度监控状态 *1
26	SLS_T2_JOINT_EN	SLS_T2的关节速度监控状态
27	SLS_T2_JOINTSPEED	SLS_T2的监控关节速度设置值
28	SLS_T2_WRIST_EN	SLS_T2的手腕(垂直6轴型: J5)速度监控状态 *1
29	SLS_T2_SHOULDER_EN	SLS_T2的肩部(垂直6轴型: J2)速度监控状态 *1
30	SLS_T_SPEED	SLS_T的监控速度设置值
31	SLS_T_JOINT_EN	SLS_T的关节速度监控状态
32	SLS_T_JOINTSPEED	SLS_T的监控关节速度设置值
33	SLS_HAND_OFS_X	SLS的X轴方向 TCP偏移位置
34	SLS_HAND_OFS_Y	SLS的Y轴方向 TCP偏移位置
35	SLS_HAND_OFS_Z	SLS的Z轴方向 TCP偏移位置
36	SLS_1_DELAY	SLS_1的延迟时间设置值
37	SLS_2_DELAY	SLS_2的延迟时间设置值
38	SLS_3_DELAY	SLS_3的延迟时间设置值
39	SLS_JOINT_POS_EN	关节角度极限的监控状态
40	SLS_JOINT_POS_ANGLE	最大关节角度 设置值
41	SLP_A_XU_EN	SLP_A的XU(壁面：X2，限制区域：X1)位置监控状态 *2
42	SLP_A_XU_POS	SLP_A的XU(壁面：X2，限制区域：X1)监控位置 设置值 *2
43	SLP_A_XL_EN	SLP_A的XL(壁面：X1，限制区域：X2)位置监控状态 *2
44	SLP_A_XL_POS	SLP_A的XL(壁面：X1，限制区域：X2)监控位置 设置值 *2
45	SLP_A_YU_EN	SLP_A的YU(壁面：Y2，限制区域：Y1)位置监控状态 *2
46	SLP_A_YU_POS	SLP_A的YU(壁面：Y2，限制区域：Y1)监控位置 设置值 *2
47	SLP_A_YL_EN	SLP_A的YL(壁面：Y1，限制区域：Y2)位置监控状态 *2
48	SLP_A_YL_POS	SLP_A的YL(壁面：Y1，限制区域：Y2)监控位置 设置值 *2
49	SLP_A_ZU_EN	SLP_A的Z2 位置监控状态 *2
50	SLP_A_ZU_POS	SLP_A的Z2 监控位置 设置值 *2
51	SLP_A_ZL_EN	SLP_A的Z1 位置监控状态 *2
52	SLP_A_ZL_POS	SLP_A的Z1 监控位置 设置值 *2
53	SLP_B_XU_EN	SLP_B的XU(壁面：X2，限制区域：X1)位置监控状态 *2
54	SLP_B_XU_POS	SLP_B的XU(壁面：X2，限制区域：X1)监控位置 设置值 *2

索引	常数	说明
55	SLP_B_XL_EN	SLP_B的XL(壁面：X1，限制区域：X2)位置监控状态 *2
56	SLP_B_XL_POS	SLP_B的XL(壁面：X1，限制区域：X2)监控位置 设置值 *2
57	SLP_B_YU_EN	SLP_B的YU(壁面：Y2，限制区域：Y1)位置监控状态 *2
58	SLP_B_YU_POS	SLP_B的YU(壁面：Y2，限制区域：Y1)监控位置 设置值 *2
59	SLP_B_YL_EN	SLP_B的YL(壁面：Y1，限制区域：Y2)位置监控状态 *2
60	SLP_B_YL_POS	SLP_B的YL(壁面：Y1，限制区域：Y2)监控位置 设置值 *2
61	SLP_B_ZU_EN	SLP_B的Z2 位置监控状态 *2
62	SLP_B_ZU_POS	SLP_B的Z2 监控位置 设置值 *2
63	SLP_B_ZL_EN	SLP_B的Z1 位置监控状态 *2
64	SLP_B_ZL_POS	SLP_B的Z1 监控位置 设置值 *2
65	SLP_C_XU_EN	SLP_C的XU(壁面：X2，限制区域：X1)位置监控状态 *2
66	SLP_C_XU_POS	SLP_C的XU(壁面：X2，限制区域：X1)监控位置 设置值 *2
67	SLP_C_XL_EN	SLP_C的XL(壁面：X1，限制区域：X2)位置监控状态 *2
68	SLP_C_XL_POS	SLP_C的XL(壁面：X1，限制区域：X2)监控位置 设置值 *2
69	SLP_C_YU_EN	SLP_C的YU(壁面：Y2，限制区域：Y1)位置监控状态 *2
70	SLP_C_YU_POS	SLP_C的YU(壁面：Y2，限制区域：Y1)监控位置 设置值 *2
71	SLP_C_YL_EN	SLP_C的YL(壁面：Y1，限制区域：Y2)位置监控状态 *2
72	SLP_C_YL_POS	SLP_C的YL(壁面：Y1，限制区域：Y2)监控位置 设置值 *2
73	SLP_C_ZU_EN	SLP_C的Z2 位置监控状态 *2
74	SLP_C_ZU_POS	SLP_C的Z2 监控位置 设置值 *2
75	SLP_C_ZL_EN	SLP_C的Z1 位置监控状态 *2
76	SLP_C_ZL_POS	SLP_C的Z1 监控位置 设置值 *2
77	SLP_J2_MON_RAD	SLP的J2轴 监控范围半径 设置值
78	SLP_J3_MON_RAD	SLP的J3轴 监控范围半径 设置值
79	SLP_J5_MON_RAD	SLP的J5轴 监控范围半径 设置值
80	SLP_J6_MON_RAD	SLP的J6轴 监控范围半径 设置值
81	SLP_J1_RANGE_MAX	轴软限位的J1轴 限位范围 最大 设置值
82	SLP_J1_RANGE_MIN	轴软限位的J1轴 限位范围 最小 设置值
83	SLP_J2_RANGE_MAX	轴软限位的J2轴 限位范围 最大 设置值
84	SLP_J2_RANGE_MIN	轴软限位的J2轴 限位范围 最小 设置值
85	SLP_J3_RANGE_MAX	轴软限位的J3轴 限位范围 最大 设置值
86	SLP_J3_RANGE_MIN	轴软限位的J3轴 限位范围 最小 设置值
87	SLP_J4_RANGE_MAX	轴软限位的J4轴 限位范围 最大 设置值
88	SLP_J4_RANGE_MIN	轴软限位的J4轴 限位范围 最小 设置值

索引	常数	说明
89	SLP_J5_RANGE_MAX	轴软限位的J5轴 限位范围 最大 设置值
90	SLP_J5_RANGE_MIN	轴软限位的J5轴 限位范围 最小 设置值
91	SLP_J6_RANGE_MAX	轴软限位的J6轴 限位范围 最大 设置值
92	SLP_J6_RANGE_MIN	轴软限位的J6轴 限位范围 最小 设置值
93	SIN_1_SLS_1_EN	对于SAFETY_IN1的SLS_1功能分配状态
94	SIN_1_SLS_2_EN	对于SAFETY_IN1的SLS_2功能分配状态
95	SIN_1_SLS_3_EN	对于SAFETY_IN1的SLS_3功能分配状态
96	SIN_1_SLP_A_EN	对于SAFETY_IN1的SLP_A功能分配状态
97	SIN_1_SLP_B_EN	对于SAFETY_IN1的SLP_B功能分配状态
98	SIN_1_SLP_C_EN	对于SAFETY_IN1的SLP_C功能分配状态
99	SIN_1_SG_EN	对于SAFETY_IN1的保护停止功能分配状态
100	SIN_1_ESTOP_EN	对于SAFETY_IN1的紧急停止功能分配状态
101	SIN_2_SLS_1_EN	对于SAFETY_IN2的SLS_1功能分配状态
102	SIN_2_SLS_2_EN	对于SAFETY_IN2的SLS_2功能分配状态
103	SIN_2_SLS_3_EN	对于SAFETY_IN2的SLS_3功能分配状态
104	SIN_2_SLP_A_EN	对于SAFETY_IN2的SLP_A功能分配状态
105	SIN_2_SLP_B_EN	对于SAFETY_IN2的SLP_B功能分配状态
106	SIN_2_SLP_C_EN	对于SAFETY_IN2的SLP_C功能分配状态
107	SIN_2_SG_EN	对于SAFETY_IN2的保护停止功能分配状态
108	SIN_2_ESTOP_EN	对于SAFETY_IN2的紧急停止功能分配状态
109	SIN_3_SLS_1_EN	对于SAFETY_IN3的SLS_1功能分配状态
110	SIN_3_SLS_2_EN	对于SAFETY_IN3的SLS_2功能分配状态
111	SIN_3_SLS_3_EN	对于SAFETY_IN3的SLS_3功能分配状态
112	SIN_3_SLP_A_EN	对于SAFETY_IN3的SLP_A功能分配状态
113	SIN_3_SLP_B_EN	对于SAFETY_IN3的SLP_B功能分配状态
114	SIN_3_SLP_C_EN	对于SAFETY_IN3的SLP_C功能分配状态
115	SIN_3_SG_EN	对于SAFETY_IN3的保护停止功能分配状态
116	SIN_3_ESTOP_EN	对于SAFETY_IN3的紧急停止功能分配状态
117	SIN_4_SLS_1_EN	对于SAFETY_IN4的SLS_1功能分配状态
118	SIN_4_SLS_2_EN	对于SAFETY_IN4的SLS_2功能分配状态
119	SIN_4_SLS_3_EN	对于SAFETY_IN4的SLS_3功能分配状态
120	SIN_4_SLP_A_EN	对于SAFETY_IN4的SLP_A功能分配状态
121	SIN_4_SLP_B_EN	对于SAFETY_IN4的SLP_B功能分配状态
122	SIN_4_SLP_C_EN	对于SAFETY_IN4的SLP_C功能分配状态

索引	常数	说明
123	SIN_4_SG_EN	对于SAFETY_IN4的保护停止功能分配状态
124	SIN_4_ESTOP_EN	对于SAFETY_IN4的紧急停止功能分配状态
125	SIN_5_SLS_1_EN	对于SAFETY_IN5的SLS_1功能分配状态
126	SIN_5_SLS_2_EN	对于SAFETY_IN5的SLS_2功能分配状态
127	SIN_5_SLS_3_EN	对于SAFETY_IN5的SLS_3功能分配状态
128	SIN_5_SLP_A_EN	对于SAFETY_IN5的SLP_A功能分配状态
129	SIN_5_SLP_B_EN	对于SAFETY_IN5的SLP_B功能分配状态
130	SIN_5_SLP_C_EN	对于SAFETY_IN5的SLP_C功能分配状态
131	SIN_5_SG_EN	对于SAFETY_IN5的保护停止功能分配状态
132	SIN_5_ESTOP_EN	对于SAFETY_IN5的紧急停止功能分配状态
133	SOUT_1_STO	对于SAFETY_OUT1的STO功能分配状态
134	SOUT_1_SLS_1	对于SAFETY_OUT1的SLS_1功能分配状态
135	SOUT_1_SLS_2	对于SAFETY_OUT1的SLS_2功能分配状态
136	SOUT_1_SLS_3	对于SAFETY_OUT1的SLS_3功能分配状态
137	SOUT_1_SLS_T2	对于SAFETY_OUT1的SLS_T2功能分配状态
138	SOUT_1_SLS_T	对于SAFETY_OUT1的SLS_T功能分配状态
139	SOUT_1_SLP_A	对于SAFETY_OUT1的SLP_A功能分配状态
140	SOUT_1_SLP_B	对于SAFETY_OUT1的SLP_B功能分配状态
141	SOUT_1_SLP_C	对于SAFETY_OUT1的SLP_C功能分配状态
142	SOUT_1_EP_RC	对于SAFETY_OUT1的紧急停止(控制器)功能分配状态
143	SOUT_1_EP_TP	对于SAFETY_OUT1的紧急停止(示教器)功能分配状态
144	SOUT_1_EN_SW	对于SAFETY_OUT1的启用开关功能分配状态
145	SOUT_2_STO	对于SAFETY_OUT2的STO功能分配状态
146	SOUT_2_SLS_1	对于SAFETY_OUT2的SLS_1功能分配状态
147	SOUT_2_SLS_2	对于SAFETY_OUT2的SLS_2功能分配状态
148	SOUT_2_SLS_3	对于SAFETY_OUT2的SLS_3功能分配状态
149	SOUT_2_SLS_T2	对于SAFETY_OUT2的SLS_T2功能分配状态
150	SOUT_2_SLS_T	对于SAFETY_OUT2的SLS_T功能分配状态
151	SOUT_2_SLP_A	对于SAFETY_OUT2的SLP_A功能分配状态
152	SOUT_2_SLP_B	对于SAFETY_OUT2的SLP_B功能分配状态
153	SOUT_2_SLP_C	对于SAFETY_OUT2的SLP_C功能分配状态
154	SOUT_2_EP_RC	对于SAFETY_OUT2的紧急停止(控制器)功能分配状态
155	SOUT_2_EP_TP	对于SAFETY_OUT2的紧急停止(示教器)功能分配状态
156	SOUT_2_EN_SW	对于SAFETY_OUT2的启用开关功能分配状态

索引	常数	说明
157	SOUT_3_STO	对于SAFETY_OUT3的STO功能分配状态
158	SOUT_3_SLS_1	对于SAFETY_OUT3的SLS_1功能分配状态
159	SOUT_3_SLS_2	对于SAFETY_OUT3的SLS_2功能分配状态
160	SOUT_3_SLS_3	对于SAFETY_OUT3的SLS_3功能分配状态
161	SOUT_3_SLS_T2	对于SAFETY_OUT3的SLS_T2功能分配状态
162	SOUT_3_SLS_T	对于SAFETY_OUT3的SLS_T功能分配状态
163	SOUT_3_SLP_A	对于SAFETY_OUT3的SLP_A功能分配状态
164	SOUT_3_SLP_B	对于SAFETY_OUT3的SLP_B功能分配状态
165	SOUT_3_SLP_C	对于SAFETY_OUT3的SLP_C功能分配状态
166	SOUT_3_EP_RC	对于SAFETY_OUT3的紧急停止(控制器)功能分配状态
167	SOUT_3_EP_TP	对于SAFETY_OUT3的紧急停止(示教器)功能分配状态
168	SOUT_3_EN_SW	对于SAFETY_OUT3的启用开关功能分配状态
169	POS_ROT_U	设置平面旋转 U_ROT 设置值
170	POS_ROT_V	设置平面旋转 V_ROT 设置值
171	POS_ROT_W	设置平面旋转 W_ROT 设置值
172	POS_OFS_X	安装位置偏移 X_OFS 设置值
173	POS_OFS_Y	安装位置偏移 Y_OFS 设置值
174	POS_OFS_Z	安装位置偏移 Z_OFS 设置值

*1 安全功能管理器的安全极限速度的监控关节J2、J3、J5与本手册中提及的超速部位(夹具、手腕、肘部和肩部)之间的对应关系如下所示。

- 水平多关节型
  - J2：肘
  - J3：无
- 垂直6轴型
  - J2：肩
  - J3：肘
  - J5：手腕
  - Hand：末端夹具

*2 安全功能管理器的安全极限位置的监控位置X1、X2、Y1、Y2、Z1、Z2与本手册提及的监控位置XL、XU、YL、YU、ZL、ZU的对应关系如下所示。

- 在监控位置选择“壁面”时
  - X1 = XL, X2 = XU
  - Y1 = YL, Y2 = YU
  - Z1 = ZL, Z2 = ZU (仅垂直6轴型)
- 在监控位置选择“限制区域”时
  - X1 = XU, X2 = XL
  - Y1 = YU, Y2 = YL
  - Z1 = ZL, Z2 = ZU (仅垂直6轴型)

本命令可用于安装有Safety板的控制器。

SF_GetParam函数使用示例

```
If SF_GetParam (SLS_1_HAND_EN) = 1 Then
Print "SLS_1 hand speed monitoring is enabled."
EndIf
```

3.22.15 SF_GetParam$函数

返回安全功能参数的文本信息。

格式
SF_GetParam$ (索引)

参数

索引
以整数值或常数指定要检索的信息索引。

返回值
返回指定的信息的字符串值。

说明
返回指定的安全功能参数值。

索引	常数	说明
1	SF_TOOLVERSION	设置工具版本
2	SF_CHECKSUM	安全功能参数校验和
3	SF_LAST_MODIFIED	安全功能参数更新日期
4	SF_ROBOT_MODEL_NAME	机器人型号名称
5	SF_ROBOT_CHECKSUM	机器人参数校验和
6	SF_HOFS	相对于物理的机器人原点的编码器偏移值(Hofs)
7	SF_HOFS_LAST_MODIFIED	Hofs最新日期

本命令可用于安装有Safety板的控制器。

SF_GetParam$函数使用示例

```
String Checksum$
Checksum$ = SF_GetParam$(SF_CHECKSUM)
Print "Safety function parameter Checksum is ", Checksum$

> Print SF_GetParam$(SF_LAST_MODIFIED)
2022/01/01 00:00:00
```

3.22.16 SF_GetStatus函数

返回安全功能的状态位。

格式
SF_GetStatus (索引)

参数

索引
以整数值指定要检索的信息索引。

返回值
以整数值返回指定的索引的信息。

说明
返回值的位信息如下表所示。

索引	位	值	说明
0	0-6	-	已预约
	7	&H80	Safety板的故障检测
1	0	&H1	启用SLS_1功能
	1	&H2	启用SLS_2功能
	2	&H4	启用SLS_3功能
	3-7	-	已预约
2	0	&H1	启用SLP_A功能
	1	&H2	启用SLP_B功能
	2	&H4	启用SLP_C功能
	3-6	-	已预约
	7	&H80	启用轴软限位功能(通常为启用)
3	0	&H1	SAFETY_IN1信号High(功能关闭) *1
	1	&H2	SAFETY_IN2信号High(功能关闭) *1
	2	&H4	SAFETY_IN3信号High(功能关闭) *1
	3	&H8	SAFETY_IN4信号High(功能关闭) *1
	4	&H10	SAFETY_IN5信号High(功能关闭) *1
	5-7	-	已预约
4	0	&H1	SAFETY_OUT1信号High(功能关闭) *2
	1	&H2	SAFETY_OUT2信号High(功能关闭) *2
	2	&H4	SAFETY_OUT3信号High(功能关闭) *2
	3-7	-	已预约
5～11	0-7	-	已预约
12	0-7	-	STF_ID

索引	位	值	说明
13	0-7	-	STF_DET_L
14	0-7	-	STF_DET_U
15	0-7	-	已预约

*1. 安全输入信号为负逻辑(Active Low)。
如果在安全输入中输入信号水平High，本函数将返回1，分配给安全输入的功能不执行动作。
如果在安全输入中输入信号电平Low，本函数将返回0，分配给安全输入的功能将执行动作。如果不将设备连接到安全输入，将变为此状态。
*2 安全输出信号为负逻辑(Active Low)。
如果分配给安全输出的任一功能处于动作状态，安全输出将被启用，安全输出的信号水平变为Low，本函数返回0。
如果分配给安全输出的任一功能均不处于动作状态，安全输出将被禁用，安全输出的信号水平变为High，本函数返回1。
如果未对安全输出分配任何功能，安全输出的信号水平将变为Low，本函数返回0。

使用STF_ID和STF_DET_L、STF_DET_H，即可确认错误发生原因。

使用SPEL+进行确认的方法将在后面说明。

STF_ID和STF_DET_L、STF_DET_H的信息如下所示。

STF_ID	错误名称	STF_DET_L	STF_DET_U
100	停止通知 安全输入	安全输入端口 SAFETY_IN1 0×01 SAFETY_IN2 0×02 SAFETY_IN3 0×04 SAFETY_IN4 0×08 SAFETY_IN5 0×10	不使用
101	停止通知 SLS_1 超速 关节	关节编号 J1 0×01 / J2 0×02 J3 0×04 / J4 0×08 J5 0×10 / J6 0×20	不使用
102	停止通知 SLS_1 超速 部位	部位 夹具 0×01 / 肘部 0×02*1 手腕 0×04 / 肩部 0×08 *1	不使用
103	停止通知 SLS_2 超速 关节	关节编号 J1 0×01 / J2 0×02 J3 0×04 / J4 0×08 J5 0×10 / J6 0×20	不使用
104	停止通知 SLS_2 超速 部位	部位 夹具 0×01 / 肘部 0×02*1 手腕 0×04 / 肩部 0×08 *1	不使用
105	停止通知 SLS_3 超速 关节	关节编号 J1 0×01 / J2 0×02 J3 0×04 / J4 0×08 J5 0×10 / J6 0×20	不使用

STF_ID	错误名称	STF_DET_L	STF_DET_U
106	停止通知 SLS_3 超速 部位	部位 夹具 0×01 / 肘部 0×02*1 手腕 0×04 / 肩部 0×08 *1	不使用
107	停止通知 SLS_T 超速 关节	关节编号 J1 0×01 / J2 0×02 J3 0×04 / J4 0×08 J5 0×10 / J6 0×20	不使用
108	停止通知 SLS_T 超速 部位	部位 夹具 0×01 / 肘部 0×02*1 手腕 0×04 / 肩部 0×08 *1	不使用
109	停止通知 SLS_T2 超速 关节	关节编号 J1 0×01 / J2 0×02 J3 0×04 / J4 0×08 J5 0×10 / J6 0×20	不使用
110	停止通知 SLS_T2 超速 部位	部位 夹具 0×01 / 肘部 0×02*1 手腕 0×04 / 肩部 0×08 *1	不使用
115	停止通知 SLP_A位置违规 监控位置	监控位置 YL 0×01 / YU 0×02 XL 0×04 / XU 0×08 ZL 0×10 / ZU 0×20 *2	关节编号 J6 0×08 J5 0×04 J3 0×02 J2 0×01
116	停止通知 SLP_B位置违规 监控位置	监控位置 YL 0×01 / YU 0×02 XL 0×04 / XU 0×08 ZL 0×10 / ZU 0×20 *2	关节编号 J6 0×08 J5 0×04 J3 0×02 J2 0×01
117	停止通知 SLP_C位置违规 监控位置	监控位置 YL 0×01 / YU 0×02 XL 0×04 / XU 0×08 ZL 0×10 / ZU 0×20 *2	关节编号 J6 0×08 J5 0×04 J3 0×02 J2 0×01
118	停止通知 轴软限位	关节编号 J1 0×01 / J2 0×02 J3 0×04 / J4 0×08 J5 0×10 / J6 0×20	不使用
121	停止通知 开关输入	开关编号 使能开关 0×01 紧急停止开关(示教器) 0×02 紧急停止开关(控制器连接) 0×04	不使用

STF_ID	错误名称	STF_DET_L	STF_DET_U
122	停止通知 模式控制	模式 参数通信许可 0×08 安全功能(Safety板) 禁用 0×04 操作模式切换 0×02 参数设置 已认证 0×01	不使用
123	停止通知 减速监控	检测异常 减速异常0×08, 0×04 减速完成 0×02 经过了监视时间 0×01	不使用
124	停止通知 关节角度极限	关节编号 J1 0×01 / J2 0×02 J3 0×04 / J4 0×08 J5 0×10 / J6 0×20	不使用
131	故障停止通知 编码器通信异常	关节编号 J1 0×01 / J2 0×02 J3 0×04 / J4 0×08 J5 0×10 / J6 0×20	不使用
132	故障停止通知 位置异常	关节编号 J1 0×01 / J2 0×02 J3 0×04 / J4 0×08 J5 0×10 / J6 0×20	不使用
133	故障停止通知 重复输入异常	检测到异常的部位 安全输入端口 SAFETY_IN 1 0×01 SAFETY_IN 2 0×02 SAFETY_IN 3 0×04 SAFETY_IN 4 0×08 SAFETY_IN 5 0×10 使能开关 0×20 紧急停止开关(示教器) 0×40 紧急停止开关(控制器连接) 0×80	不使用
134	故障停止通知 重复输出异常	检测到异常的部位 安全输出端口 SAFETY_OUT 1 0×01 SAFETY_OUT 2 0×02 SAFETY_OUT 3 0×04 STO 0×80	不使用
135	故障停止通知 Safety板异常	检测到异常的部位 通信总线 0×20 电源(3.3V) 0×08 电源(5V) 0×04 看门狗定时器检测 0×02 继电器融接 0×01	不使用

STF_ID	错误名称	STF_DET_L	STF_DET_U
136	故障停止通知 Safety板 MCU异常	检测到异常的部位 序列监控 0×10 CPU 0×08 RAM 0×04 程序ROM 0×02 数据ROM 0×01	DET_L = 0×01时 (数据ROM) 0×00 - 0×FE 数据故障位置 0×FF 参数的故障
137	故障停止通知 Safety板 重复内部异常	检测到异常的部位 TCP位置不一致 0×02 状态不一致 0×01	不使用
138	故障停止通知 编码器 内部异常	关节编号 J1 0×01 / J2 0×02 J3 0×04 / J4 0×08 J5 0×10 / J6 0×20	不使用
139	故障停止通知 控制器 内部异常	检测到异常的部位 操作模式接收错误 0×01	不使用

*1 安全功能管理器的安全极限速度的监控关节J2、J3、J5与本手册中提及的超速部位(夹具、手腕、肘部和肩部)之间的对应关系如下所示。

- 水平多关节型
  - J2：肘
  - J3：无
  - J5：无
  - Hand：末端夹具
- 垂直6轴型
  - J2：肩
  - J3：肘
  - J5：手腕
  - Hand：末端夹具

*2 安全功能管理器的安全极限位置的监控位置X1、X2、Y1、Y2、Z1、Z2与本手册提及的监控位置XL、XU、YL、YU、ZL、ZU的对应关系如下所示。

- 在监控位置选择“壁面”时
  - X1 = XL, X2 = XU
  - Y1 = YL, Y2 = YU
  - Z1 = ZL, Z2 = ZU (仅垂直6轴型)
- 在监控位置选择“限制区域”时
  - X1 = XU, X2 = XL
  - Y1 = YU, Y2 = YL
  - Z1 = ZL, Z2 = ZU (仅垂直6轴型)

使用STF_ID和STF_DET_L、STF_DET_H确认错误发生原因的方法

在命令窗口等按以下顺序输入并确认命令。

```
> SF_GetStatus (12)
115     '表示由于"停止通知 SLP_A 位置违规"而产生的错误。
```

```
> SF_GetStatus (13)
1     '表示超过"YL"方向的监控位置。(确认STF_ID: 115的内容。)
> SF_GetStatus (14)
1     '表示"J2轴"已超过。(确认STF_ID: 115的内容。)
```

综上所述，可知错误发生的原因在于“J2轴超过了SLP_A的YL的监控位置”。

错误信息也被记录在Epson RC+的系统历史中。错误发生原因则被记录在“附加信息”中。请参阅《状态与错误信息》中关于错误信息的内容。

本命令可用于安装有Safety板的控制器。

SF_GetStatus函数使用示例

```
If (SF_GetStatus(3) And &H1) = &H1 Then
Print "SAFETY_IN1 is High"
Else
Print "SAFETY_IN1 is Low"
EndIf
```

3.22.17 SF_LimitSpeedS

启用SLS时，设置并显示关于Tool命令所设置位置的速度调整功能的速度调整值。

格式

SF_LimitSpeedS [SLS编号 [, 速度调整值]]

参数

SLS编号

以整数值(1～3)或以下所示常数指定SLS编号。可省略。

常数	值	内容
SLS_1	1	启用SLS_1时的速度调整功能
SLS_2	2	启用SLS_2时的速度调整功能
SLS_3	3	启用SLS_3时的速度调整功能
_SLS_T	9	启用SLS_T时的速度调整功能 *1
_SLS_T2	10	启用SLS_T2时的速度调整功能 *1

*1 SLS_T和SLS_T2无法使用本命令设置速度调整值。而是使用安全功能管理器中设定的监控速度进行速度调整。有关详细信息，请参阅以下手册。
《机器人控制器 安全功能手册》

速度调整值

以整数值(0～10000、单位：mm/sec)指定速度。可省略。如果指定为0，将自动设置速度调整值。默认值为0。

说明

启用SLS时，设置并显示关于Tool命令所设置位置的速度调整功能。请注意，使用本功能调整速度的部位为Tool命令当前选择的工具位置，并非使用安全功能参数设置的TCP位置。

启用SLS时，设置速度调整功能指定的SLS编号的速度调整值[mm/sec]。省略第二个参数时，将显示指定的SLS编号的速度调整值。

省略所有参数时，将显示所有SLS编号的速度调整值。

本命令可用于安装有Safety板的控制器。

仅在启用安全功能选项时可用。

注意

- 使用SF_LimitSpeedS调整速度的位置为Tool命令选择的工具位置

  使用SF_LimitSpeedS设置的速度调整值应用至Tool命令设置的工具位置的速度。安全功能管理器的TCP位置不自动设置至Tool命令。请利用Tool命令，设置正确的TCP位置。

  此外，速度自动调整不能正常动作时，请使用SF_PeakSpeedS、SF_RealSpeedS测量机器人手臂运动速度，并使用Speed、SpeedFactor、SpeedS等进行控制，使机器人手臂运动速度不超过SLS最大速度(安全功能参数)。

SF_LimitSpeedS使用示例

将SLS_1的速度调整值设置为1500mm/sec的方法

```
SF_LimitSpeedS SLS_1, 1500
```

显示SLS_2的速度调整值的方法(使用命令窗口)

```
> SF_LimitSpeedS SLS_2
SLS_2: 400
```

显示所有SLS编号的速度调整值的方法(使用命令窗口)

```
> SF_LimitSpeedS
SLS_1: 1500
SLS_2: 400
SLS_3: 750
SLS_T: 250
SLS_T2: 3000
```

3.22.18 SF_LimitSpeedS函数

启用SLS时，返回Tool命令所设置位置的速度调整功能的速度调整值。

格式

SF_LimitSpeedS (SLS编号)

参数

SLS编号

以整数值或以下所示常数指定SLS编号。

常数	值	内容
SLS_1	1	启用SLS_1时的速度调整功能
SLS_2	2	启用SLS_2时的速度调整功能
SLS_3	3	启用SLS_3时的速度调整功能
_SLS_T	9	启用SLS_T时的速度调整功能
_SLS_T2	10	启用SLS_T2时的速度调整功能

返回值

返回指定SLS编号的速度调整值[mm/sec]。

说明

启用SLS时，返回速度调整功能指定的SLS编号的速度调整值[mm/sec]。

本命令可用于安装有Safety板的控制器。

SF_LimitSpeedS函数使用示例

```
Integer i
i = SF_LimitSpeedS(SLS_1)
Print "SLS_1 limit speed is ", i
```

3.22.19 SF_LimitSpeedSEnable

启用SLS时，设置关于“Tool命令所设置位置的速度调整功能”的On/Off，并显示设置状态。

格式

SF_LimitSpeedSEnable [SLS编号 [, {On | Off}]]

参数

SLS编号

以整数值或以下所示常数指定SLS编号。可省略。

常数	值	内容
SLS_1	1	启用SLS_1时的速度调整功能
SLS_2	2	启用SLS_2时的速度调整功能
SLS_3	3	启用SLS_3时的速度调整功能
_SLS_T	9	启用SLS_T时的速度调整功能 *1
_SLS_T2	10	启用SLS_T2时的速度调整功能 *1

*1 SLS_T和SLS_T2无法使用本命令设置速度调整功能。SLS_T在操作模式为TEACH和TEST T1时，速度调整功能为ON。SLS_T2在操作模式为TEST T2时，速度调整功能为ON。有关详细信息，请参阅以下手册。
《机器人控制器 安全功能手册》

On | Off

指定速度调整功能的On/Off。可省略。默认设置为On。

返回值

无

说明

设置并显示指定的SLS编号为监控状态时Tool命令所设置位置的速度调整功能的On/Off。如果设为On，启用SLS时调整设置工具位置的速度。如果设为Off，即使启用了SLS，也不调整速度。无论On/Off设置如何，停用SLS时不调整工具位置的速度。

省略第二个参数时，将显示指定的SLS编号的速度调整状态(On/Off)。

省略所有参数时，将显示所有SLS编号的速度调整状态(On/Off)。

本命令可用于安装有Safety板的控制器。

仅在启用安全功能选项时可用。

SF_LimitSpeedSEnable使用示例

将SLS_1的速度调整功能设置为启用的方法

```
SF_LimitSpeedSEnable SLS_1, On
```

显示SLS_2的速度调整功能状态的方法(使用命令窗口)

```
> SF_LimitSpeedSEnable SLS_2
SLS_2: Off
```

显示所有SLS编号的速度调整功能状态的方法(使用命令窗口)

```
> SF_LimitSpeedSEnable
SLS_1: On
SLS_2: Off
SLS_3: Off
```

```
SLS_T: On
SLS_T2: On
```

注意

- SF_LimitSpeedSEnable无法与特异点回避、传送带跟踪和力控制同时使用。

  SF_LimitSpeedSEnable的默认设置为On。如果在力控制、传送带跟踪、特异点回避运行过程中通过本功能进行速度调整，将发生动作错误(分别为4093、4403、5830)。

  在这些动作中使用SLS时，请在动作开始前将SF_LimitSpeedSEnable设置为Off。

  此外，请使用SF_PeakSpeedS、SF_RealSpeedS测量机器人手臂速度，并运动使用Speed、SpeedFactor、SpeedS等命令控制机器人手臂运动速度，使其不超过SLS监控速度(安全功能参数)。

- 使用SF_LimitSpeedS调整速度的位置为Tool命令设置的工具位置

  SF_LimitSpeedSEnable为On时应用的速度调整值(使用SF_LimitSpeedS设定)会应用至Tool命令设置的工具位置的速度。安全功能管理器的TCP位置不自动设置至Tool命令。请利用Tool命令，设置正确的TCP位置。

  此外，速度自动调整不能正常动作时，请使用SF_PeakSpeedS、SF_RealSpeedS测量机器人手臂运动速度，并使用Speed、SpeedFactor、SpeedS等进行控制，使机器人手臂运动速度不超过SLS最大速度(安全功能参数)。

3.22.20 SF_LimitSpeedSEnable函数

启用SLS时，返回Tool命令所设置位置的速度调整功能的状态。

格式

SF_LimitSpeedSEnable (SLS编号)

参数

SLS编号

以整数值(1～3)或以下所示常数指定SLS编号。

常数	值	内容
SLS_1	1	启用SLS_1时的速度调整功能
SLS_2	2	启用SLS_2时的速度调整功能
SLS_3	3	启用SLS_3时的速度调整功能
_SLS_T	9	启用SLS_T时的速度调整功能
_SLS_T2	10	启用SLS_T2时的速度调整功能

返回值

- 如果指定的SLS编号的速度调整为On，将返回1。
- 如果指定的SLS编号的速度调整为Off，将返回0。

说明

指定的SLS编号处于监控状态时，返回调整TCP运动速度的功能的状态。

本命令可用于安装有Safety板的控制器。

SF_LimitSpeedSEnable函数使用示例

```
If SF_LimitSpeedSEnable(SLS_1) = 0 Then
Print "SLS_1 linked speed adjustment function is disabled."
Endif
```

3.22.21 SF_PeakSpeedS

显示速度监控点的峰值速度值。

格式

SF_PeakSpeedS [速度监控点编号]

参数

速度监控点编号

以整数值(1～4)指定速度监控点编号。可省略。

- 1：末端夹具
- 2：肘部
- 3:手腕
- 4：肩

返回值

无

说明

显示指定的速度监控点的峰值速度值。

省略参数时，将显示所有速度监控点的峰值速度值。

夹具的速度是在安全功能管理器中设置的TCP偏移位置上的速度。

本命令可用于安装有Safety板的控制器。

SF_PeakSpeedS使用示例

显示夹具的峰值速度值的方法(使用命令窗口)

```
> SF_PeakSpeedS 1
250
```

显示所有速度监控点的峰值速度值的方法(使用命令窗口)

```
> SF_PeakSpeedS
250       150
240       100
```

显示顺序如下所示。

末端夹具　　肘

手腕　　肩

3.22.22 SF_PeakSpeedS函数

返回速度监控点的峰值速度。

格式

SF_PeakSpeedS (速度监控点编号)

参数

速度监控点编号

以整数值(1～4)指定速度监控点编号。可省略。

- 1：末端夹具
- 2：肘部
- 3：手腕
- 4：肩

返回值

返回峰值速度[mm/sec]。

说明

返回指定的速度监控点的峰值速度。

夹具的速度是在安全功能管理器中设置的TCP偏移位置上的速度。

本命令可用于安装有Safety板的控制器。

SF_PeakSpeedS函数使用示例

```
Print "Hand peak speed is ", SF_PeakSpeedS (1)
```

3.22.23 SF_PeakSpeedSClear

清除并初始化速度监控点的峰值速度值。

格式

SF_PeakSpeedSClear [速度监控点编号1 [, 速度监控点编号2]]

参数

速度监控点编号1
以整数值(1～4)指定第1个速度监控点编号。可省略。
速度监控点编号2
以整数值(1～4)指定第2个速度监控点编号。可省略。

- 1：末端夹具
- 2：肘部
- 3：手腕
- 4：肩

返回值

无

说明

清除(初始化)指定的速度监控点的峰值速度值。

如果不指定参数，所有速度监控点的峰值速度值将被清除。

本命令可用于安装有Safety板的控制器。

SF_PeakSpeedSClear使用示例

清除肘部的峰值速度值的方法

```
SF_PeakSpeedSClear 1
```

清除所有速度监控点的峰值速度值的方法

```
SF_PeakSpeedSClear
```

3.22.24 SF_RealSpeedS

显示速度监控点的当前速度。

格式

SF_RealSpeedS [速度监控点编号]

参数

速度监控点编号

以整数值(1～4)指定速度监控点编号。可省略。

- 1：末端夹具
- 2：肘部
- 3：手腕
- 4：肩

返回值

无

说明

显示指定的速度监控点的当前速度[mm/sec]。

夹具的速度是在安全功能管理器中设置的TCP偏移位置上的速度。

省略参数时，将显示所有速度监控点的当前速度。

本命令可用于安装有Safety板的控制器。

SF_RealSpeedS使用示例

显示夹具当前速度的方法(使用命令窗口)

```
> SF_RealSpeedS 1
250
```

显示所有速度监控点的当前速度的方法(使用命令窗口)

```
>SF_RealSpeedS
250       150
240       100
```

显示顺序如下所示。

末端夹具　　肘

手腕　　肩

3.22.25 SF_RealSpeedS函数

返回速度监控点的当前速度。

格式

SF_RealSpeedS (速度监控点编号)

参数

速度监控点编号

以整数值(1～4)指定速度监控点编号。可省略。

- 1：末端夹具
- 2：肘部
- 3：手腕
- 4：肩

返回值

返回当前速度[mm/sec]。

说明

返回指定的速度监控点的当前速度[mm/sec]。

夹具的速度是在安全功能管理器中设置的TCP偏移位置上的速度。

本命令可用于安装有Safety板的控制器。

SF_RealSpeedS函数使用示例

```
Print "Hand real speed is ", SF_RealSpeedS (1)
```

3.22.26 SFree

将指定关节更改为SFree状态。

格式
SFree 关节编号[, 关节编号,...]

参数

关节编号
以表达式或数值指定关节编号(1～9的整数)。附加轴的S轴为8，T轴为9。

说明
SFree用于切断指定关节的电动机电源。此时的状态称为SFree。该命令用于进行直接示教，或仅切换位SFree状态的关节进行嵌入等。要释放关节的SFree状态时，请执行SLock命令或Motor On/ Motor Off。

如果执行SFree命令，则会对机器人控制参数进行初始化。

详情请参阅Motor On。

注意

- 执行SFree时，部分系统设置会被初始化

  SFree用于对有关机器人动作速度或加减速度的参数(Speed、SpeedS、Accel、AccelS等)和LimZ参数进行初始化，以确保安全。详情请参阅Motor On。

  在固件版本7.5.1.0之前，SFree命令只能在Motor On时执行。

  SFree的操作，会因固件版本发生如下变化。

固件	能否SFree
7.5.1.0之前	仅电机开启时
7.5.1.0或以后	电机开启或关闭时

重要事项

- 将SFree用于水平多关节型机器人(包括RS系列)的第3/第4关节时

  由于水平多关节型机器人(包括RS系列)的第3关节施加有电磁制动，因此，即使设置SFree，第3关节也不会立即进行动作。要手动移动第3关节时，需要按住位于机械臂上部的制动解除开关。

  另外，有些机型的第4关节带有制动器。为第4关节带有制动器的机型时，由于第4关节施加有制动，因此，即使设置SFree，第4关节也不会立即进行动作。要手动移动第4关节时，在按住位于机械臂上部的制动解除开关的同时进行移动。

- 不能将SFree用于垂直6轴型机器人(包括N系列)

  如果在垂直6轴型机器人(包括N系列)中使用SFree，将发生错误。

  手动操作机械臂时，需要在关闭电机后在Beake Off中解除电磁制动器。

- 关节处于SFree状态时执行动作命令

如果在关节处于SFree状态时执行动作命令，控制器则会在默认状态下发生错误。即使在1个关节处于SFree的状态下也要执行动作命令时，勾选［设置］菜单中的［设置控制器］-［环境设置］面板的［允许一个或多个关节在刹车释放状态下动作］复选框。

- 传送带跟踪期间请勿使用SFree

  如果在传送带跟踪期间使用SFree，将发生错误5057、5058等。请利用Cnv_AbortTrack命令结束传送带跟踪，然后再使用SFree。

参阅

Brake、LimZ、Motor、SFree函数、SLock

SFree使用示例

如下所示为SFree命令的简单使用示例。在本例当中，要进行动作时，必须勾选［设置］菜单中的[设置控制器］-［环境设置］面板的［允许一个或多个关节在刹车释放状态下动作］复选框。

```
Function GoPick
   Speed pickSpeed
   SFree 1, 2    '将J1和J2设为SFree状态，
然后移动Z和U关节以安装部件
   Go pick
   SLock 1, 2    '对J1和J2解除SFree
Fend
```

3.22.27 SFree函数

用于返回指定关节的SFree状态。

格式

Sfree (关节编号)

参数

关节编号
以数值或表达式指定要检查SFree状态的关节的编号(整数)。附加轴的S轴为8，T轴为9。

返回值

- 1：SFree状态
- 0：非SFree状态

参阅

SFree

SetFree使用示例

```
If SFree(1) Then
    Print "Joint 1 is free"
EndIf
```

3.22.28 Sgn函数

用于返回数值的符号。

格式
Sgn (操作数)

参数

操作数
　　指定数值。

返回值

- 1：操作数为正值
- 0：操作数是0
- -1：操作数为负值

说明
Sgn函数用于返回数值的符号。

参阅
Abs、And、Atan、Atan2、Cos、Int、Mod、Or、Not、Sin、Sqr、Str$、Tan、Val、Xor

Sgn函数使用示例
如下所示为通过命令窗口使用Sgn的示例。

```
>print sgn(123)
 1
>print sgn(-123)
 -1
>
```

3.22.29 Short

用于声明Short型变量。(2字节整数型变量)

格式

Short 变量名 [(数组变量的最大下标)] [, 变量名 [(数组变量的最大下标)]...]

参数

变量名
　　指定要进行变量声明的变量名。
数组变量的最大下标
　　可利用数组变量的最大下标声明到最高三维。使用下述格式。可省略。
　　(最大下标1, [最大下标2], [最大下标3])
　　由于下标从0开始，因此数元素为最大下标加上1。在所有元素数不超过以下最大值的范围内指定各最大下标。

- 本地变量：2,000
- 备份变量(Global Preserve)： 4,000
- 全局变量和模块变量：100,000

说明

Short用于声明整数型变量。整数型变量的范围为-32768～32767。在Function开头声明本地变量。在Function之外声明全局变量和模块变量。

参阅

Boolean、Byte、Double、Global、Int32、Int64、Integer、Long、Real、String、UByte、UInt32、UInt64、UShort

Short使用示例

如下所示为使用Short声明整数型变量的程序。

```
Function shorttest
  Short A(10)          'Short型的一维数组
  Short B(10, 10)      'Short型的二维数组
  Short C(5, 5, 5)     'Short型的三维数组
  Short var1, arrayvar(10)
  Integer i
  Print "Please enter an Integer Number"
  Input var1
  Print "The Integer variable var1 = ", var1
  For i = 1 To 5
    Print "Please enter an Integer Number"
    Input arrayvar(i)
    Print "Value Entered was ", arrayvar(i)
  Next i
Fend
```

3.22.30 ShutDown

用于关闭Epson RC+、关闭或重启 Windows。

格式

ShutDown [模式] [, Forced]

参数

模式

以整数值设置下述模式。

常数	值	内容
省略模式	-1	显示对话框，用户可选择关闭方法。
SHUTDOWN_ALL	0	关闭Epson RC+和Windows。
SHUTDOWN_RESTART	1	用于关闭Epson RC+、重启Windows。
SHUTDOWN_EPSONRC	2	关闭Epson RC+。

Forced

进行强制关闭时设置。可省略。

说明

可使用Shutdown结束RC+，或利用程序关闭或重启Windows。如果使用Forced参数，则强制关闭。

注意

如果在执行任务期间强制关闭，则可能会导致数据受损。请在关闭之前保存数据。

- 支持的控制器型号

  Shutdown命令不支持T/VT系列。

- 当在虚拟控制器中将控制器设置为“合作模式”时

  当在虚拟控制器中将控制器设置为“合作模式”时，执行ShutDown不会保存备份变量。如需保存备份变量，请不要使用ShutDown。

参阅

Restart

ShutDown使用示例

```
ShutDown 0     ' 关闭Epson RC+与Windows
```

3.22.31 ShutDown函数

用于关闭Epson RC+、关闭或重启 Windows。

格式

ShutDown (模式[, Forced])

参数

模式

以整数值设置下述模式。

常数	值	内容
省略模式	-1	显示对话框，用户可选择关闭方法。
SHUTDOWN_ALL	0	关闭Epson RC+和Windows。
SHUTDOWN_RESTART	1	用于关闭Epson RC+、重启Windows。
SHUTDOWN_EPSONRC	2	关闭Epson RC+。

Forced

进行强制关闭时设置。可省略。

说明

可使用Shutdown结束RC+，或通过程序关闭或重启Windows。如果使用Forced参数，则强制关闭。

注意

如果在执行任务期间强制关闭，则可能会导致数据受损。请在关闭之前保存数据。

- 支持的控制器型号

  Shutdown函数不支持T/VT系列。

- 当在虚拟控制器中将控制器设置为“合作模式”时

  当在虚拟控制器中将控制器设置为“合作模式”时，执行ShutDown不会保存备份变量。如需保存备份变量，请不要使用ShutDown函数。

返回值

返回以下整数值。

- -1：显示对话框时，如果用户选择取消，则返回该值。
- 0：关闭失败时返回该值。
- 1：关闭成功时返回该值。

ShutDown函数使用示例

```
If Shutdown(SHUTDOWN_EPSONRC) = 1 Then
    Print "Shutdown: OK"
Else
    Print "Shutdown: NG"
EndIf
```

3.22.32 Signal

用于向正在执行WaitSig命令的任务发送信号。

格式

Signal 信号编号

参数

信号编号
　　指定0～63的要发送的信号编号。

说明

Signal用于实现与多任务的同步。执行WaitSig前的Signal被无视。

参阅

WaitSig

Signal使用示例

```
Function Main
   Xqt 2, SubTask
   Call InitSys
   Signal 1

Fend

Function SubTask
   WaitSig 1

Fend
```

3.22.33 SimGet

用于获取模拟器的各种目标的属性设置值。

格式

SimGet Object.Property, Var

SimGet Robot.Hand.Propoerty, Var

参数

Object
　　表示要获取属性值的目标名称的字符串变量
Robot
　　表示已安装由“Hand”指定的末端夹具的机器人名称的字符串变量
Hand
　　表示要获取属性值的末端夹具名称的字符串变量
Property
　　是要获取值的属性名称。有关属性，将在后文叙述。
Var
　　表示要返回的值的变量

说明

该命令用于获取模拟器的各种目标的属性设置值。

可通过指定下述属性，获取目标的设置值。

属性	说明	单位	数据类型	返回值
PositionX	获取X坐标位置	毫米 (mm)	Double	
PositionY	获取Y坐标位置	毫米 (mm)	Double	
PositionZ	获取Z坐标位置	毫米 (mm)	Double	
RotationX	获取X轴旋转角度	度 (degree)	Double	
RotationY	获取Y轴旋转角度	度 (degree)	Double	
RotationZ	获取Z轴旋转角度	度 (degree)	Double	
CollisionCheck	获取碰撞检测的有效/无效	-	Boolean	True或False
CollisionCheckSelf	获取机器人自身碰撞检测的有效/无效	-	Boolean	True或False
Visible	获取显示/隐藏的状态	-	Boolean	True或False
Type	获取目标类型	-	Integer	Layout: 0 Part: 1 Mounted Device: 3
HalfSizeX	获取Box对象在X方向的长度	毫米(mm)	Double	
HalfSizeY	获取Box对象在Y方向的长度	毫米(mm)	Double	
HalfSizeZ	获取Box对象在Z方向的长度	毫米(mm)	Double	
HalfSizeHeight	获取Plane对象的长度	毫米(mm)	Double	

属性	说明	单位	数据类型	返回值
HalfSizeWidth	获取Plane对象的宽	毫米(mm)	Double	
PlaneType	获取Plane对象的类型	-	Integer	Horizontal: 0 Vertical: 1
Radius	获取Sphere对象或Cylinder对象的半径	毫米(mm)	Double	
Height	获取Cylinder对象的高度	毫米(mm)	Double	
Name	获取对象名称		String	
Color	获取对象的显示颜色		String	颜色名称或十六进制颜色代码(ARGB)

可通过下表所示的组合获取属性。

属性	目标							
	Robot	Hand	Box	Sphere	Cylinder	Plane	CAD	Camera
PositionX	○	○	○	○	○	○	○	○
PositionY	○	○	○	○	○	○	○	○
PositionZ	○	○	○	○	○	○	○	○
RotationX	○	○	○	○	○	○	○	○
RotationY	○	○	○	○	○	○	○	○
RotationZ	○	○	○	○	○	○	○	○
CollisionCheck	○	○	○	○	○	○	○	○
CollisionCheckSelf	○	-	-	-	-	-	-	-
Visible	-	○	○	○	○	○	○	○
Type	-	-	○	○	○	○	○	-
HalfSizeX	-	-	○	-	-	-	-	-
HalfSizeY	-	-	○	-	-	-	-	-
HalfSizeZ	-	-	○	-	-	-	-	-
HalfSizeHeight	-	-	-	-	-	○	-	-
HalfSizeWidth	-	-	-	-	-	○	-	-
PlaneType	-	-	-	-	-	○	-	-
Radius	-	-	-	○	○	-	-	-
Height	-	-	-	-	○	-	-	-
Name	○	○	○	○	○	○	○	○
Color	-	-	○	○	○	○	-	-

参阅

SimSet

SimGet使用示例

```
'用于获取SBox_1目标的X坐标值
Double boxPosX
SimGet SBox_1.PositionX, boxPosX

'用于获取SBox_1目标的显示/隐藏状态
Boolean boxVisible
SimGet SBox_1.Visible, boxVisible

'用于获取SBox_1目标的类型
Integer boxType
SimGet SBox_1.Type, boxType
```

3.22.34 SimSet

用于设置模拟器的各种目标的属性。另外，也用于进行机器人动作、目标操作、模拟器设置等相关操作。

格式

(1) 目标的属性设置

```
SimSet Object.Property, Value
SimSet Robot.Hand.Property, Value
```

(2) 机器人的动作设置(Pick与Place)

```
SimSet Robot.Pick, Object [,Tool]
SimSet Robot.Place, Object
```

(3) 目标的操作设置(父目标的指定)

```
SimSet Object.SetParent [, ParentObject]
```

(4) 模拟器的设置(碰撞检测的重置)

```
SimSet ResetCollision
```

参数

(1) 目标的属性设置

Object
　　表示要设置属性值的项目名称的字符串变量
Robot
　　表示已安装由“Hand”指定的末端夹具的机器人名称的字符串变量
Hand
　　表示要设置属性值的末端夹具名称的字符串变量
Property
　　是要重新设置值的属性名称。有关属性，将在后文叙述。
Value
　　新值的表达式。数据类型取决于属性。

(2) 机器人的动作设置(Pick与Place)

Robot
　　表示要进行Pick或Place的机器人名称的字符串变量
Object
　　表示已进行Pick或Place的目标名称的字符串变量
Tool
　　表示Pick时要利用的Tool编号的表达式

(3) 目标的操作设置(父目标的指定)

Object
　　表示要设置父目标的目标名称的字符串变量
ParentObject
　　表示父目标名称的字符串变量

说明

该命令用于变更模拟器的各种目标的属性设置、操作、机器人动作、模拟器设置。

(1) 目标的属性设置

可通过指定如下所示的属性，获取目标的设置。

属性	说明	单位	数据类型	设置值
PositionX	用于设置X坐标位置	毫米 (mm)	Double	最大值：100000 最小值：-100000
PositionY	用于设置Y坐标位置	毫米 (mm)	Double	最大值：100000 最小值：-100000
PositionZ	用于设置Z坐标位置	毫米 (mm)	Double	最大值：100000 最小值：-100000
RotationX	用于设置X轴旋转角度	度 (degree)	Double	最大值：360 最小值：-360
RotationY	用于设置Y轴旋转角度	度 (degree)	Double	最大值：360 最小值：-360
RotationZ	用于设置Z轴旋转角度	度 (degree)	Double	最大值：360 最小值：-360
CollisionCheck	用于设置碰撞检测的有效/无效	-	Boolean	True或False
CollisionCheckSelf	用于设置机器人自身碰撞检测的有效/无效	-	Boolean	True或False
Visible	用于设置显示/隐藏	-	Boolean	True或False
HalfSizeX	获取Box对象在X方向的长度	毫米(mm)	Double	最大值：100000 最小值：0.001
HalfSizeY	获取Box对象在Y方向的长度	毫米(mm)	Double	最大值：100000 最小值：0.001
HalfSizeZ	获取Box对象在Z方向的长度	毫米(mm)	Double	最大值：100000 最小值：0.001
HalfSizeHeight	获取Plane对象的长度	毫米(mm)	Double	最大值：100000 最小值：0.001
HalfSizeWidth	获取Plane对象的宽	毫米(mm)	Double	最大值：100000 最小值：0.001
PlaneType	获取Plane对象的类型	-	Integer	Horizontal: 0 Vertical: 1
Radius	获取Sphere对象或Cylinder对象的半径	毫米(mm)	Double	最大值：100000 最小值：0.001
Height	获取Cylinder对象的高度	毫米(mm)	Double	最大值：100000 最小值：0.001
Name	获取对象名称		String	
Color	获取对象的显示颜色		String	颜色名称或十六进制颜色代码(ARGB)

可通过下表所示的组合设置属性。

属性	目标							
	Robot	Hand	Box	Sphere	Cylinder	Plane	CAD	Camera
PositionX	○	○	○	○	○	○	○	○
PositionY	○	○	○	○	○	○	○	○
PositionZ	○	○	○	○	○	○	○	○
RotationX	○	○	○	○	○	○	○	○
RotationY	○	○	○	○	○	○	○	○
RotationZ	○	○	○	○	○	○	○	○
CollisionCheck	○	○	○	○	○	○	○	○
CollisionCheckSelf	○	-	-	-	-	-	-	-
Visible	-	○	○	○	○	○	○	○
HalfSizeX	-	-	○	-	-	-	-	-
HalfSizeY	-	-	○	-	-	-	-	-
HalfSizeZ	-	-	○	-	-	-	-	-
HalfSizeHeight	-	-	-	-	-	○	-	-
HalfSizeWidth	-	-	-	-	-	○	-	-
PlaneType	-	-	-	-	-	○	-	-
Radius	-	-	-	○	○	-	-	-
Height	-	-	-	-	○	-	-	-
Name	○	○	○	○	○	○	○	○
Color	-	-	○	○	○	○	-	-

(2) 机器人的动作设置(Pick与Place)

可设置如下所示的机器人动作。

- Pick

  由“Robot”指定的机器人对由“Object”指定的目标进行夹持动作。

  被夹持的目标将被注册为该机器人的部件。另外，可通过在“Tool”中指定任意工具编号，利用指定的工具进行夹持动作。省略“Tool”指定时，利用Tool0进行夹持动作。

  不能夹持已注册为部件的目标或已设为机械臂安装设备的目标。另外，也不能夹持摄像机。

- Place

  由“Robot”指定的机器人对由“Object”指定的目标进行配置的动作。被配置的目标将被解除作为该机器人部件注册的状态。

  不能配置已解除作为部件注册的目标。

  可通过下表所示的组合对目标进行夹持或配置。

动作	目标							
	Robot	Hand	Box	Sphere	Cylinder	Plane	CAD	Camera
Pick	-	-	○	○	○	○	○	-
Place	-	-	○	○	○	○	○	-

(3) 目标的操作设置(父目标的指定)

可设置如下所示的目标操作。

- SetParent

  相对于由“Object”指定的目标，将由“ParentObject”指定的目标设为父目标。可省略“ParentObject”。在这种情况下，由“Object”指定的目标变为父目标。比如，由“Object”指定的目标为某些目标的子目标时，将被解除作为子目标的设置。

  另外，由“Object”指定的目标被设为部件或机械臂安装设备时，不能指定父目标。

  下表所示为可指定SetParent的目标。针对摄像机目标，仅限于被设为固定摄像机的目标方可利用SetParent。

操作	目标							
	Robot	Hand	Box	Sphere	Cylinder	Plane	CAD	Camera
SetParent	-	-	○	○	○	○	○	○

  另外，可按下表所示的组合利用SetParent。

		ParentObject(父目标)							
		Robot	Hand	Box	Sphere	Cylinder	Plane	CAD	Camera
Object (子目标)	Robot	-	-	-	-	-	-	-	-
	Hand	-	-	-	-	-	-	-	-
	Box	-	-	○	○	○	○	○	○
	Sphere	-	-	○	○	○	○	○	○
	Cylinder	-	-	○	○	○	○	○	○
	Plane	-	-	○	○	○	○	○	○
	CAD	-	-	○	○	○	○	○	○
	Camera	-	-	○	○	○	○	○	○

(4)模拟器的设置(碰撞检测的重置)可变更如下所示的模拟器设置。

- ResetCollision

  用于对碰撞检测进行重置。执行ResetCollision之后，如果机器人与目标之间未发生碰撞，碰撞状态将被解除，并更新模拟器的3D显示。如果机器人与目标之间发生碰撞，不解除碰撞状态，也不更新模拟器的3D显示。

参阅

SimGet

SimSet使用示例

```
'将SBox_1目标的X坐标值设为100.0 mm
SimSet SBox_1.PositionX, 100.0

'在Robot1上利用Tool1夹持SBox_1
SimSet Robot1.Pick, SBox_1, 1

'配置由Robot1夹持的SBox_1
SimSet Robot1.Place, SBox_1

'将CAD_1设为SBox_1的父目标
SimSet SBox_1.SetParent, CAD_1

'将SBox_1设为父目标
SimSet SBox_1.SetParent

'用于对碰撞检测进行重置
SimSet ResetCollision
```

3.22.35 Sin函数

用于返回已指定数值的正弦值。

格式
Sin (角度)

参数

角度
以实值指定角度。

返回值
以数值返回表示指定角度的正弦值。

说明
返回已指定角度(弧度)的正弦值。参数单位必须为弧度。

返回值的范围为-1～1。

要将弧度转换为角度时，需要使用RadToDeg函数。

参阅
Abs、Atan、Atan2、Cos、Int、Mod、Not、Sgn、Sqr、Str$、Tan、Val

Sin函数使用示例
如下所示为使用Sin函数的示例。

```
Function sintest
    Real x
    Print "Please enter a value in radians:"
    Input x
    Print "Sin of ", x, " is ", Sin(x)
Fend
```

3.22.36 SingularityAngle

用于设置超过特殊方向属性功能所需的特殊点附近的角度。

格式

SingularityAngle {角度设置值}

参数

角度设置值

以表达式或数值指定用于判断垂直6轴型机器人(包括N系列)手腕特异姿势近旁所需的第5关节角度(0.1以上的实数，单位是deg)。

结果

如果省略参数，则显示当前的SingularityAngle设置值。

说明

此命令只在使用特殊方向属性功能时有效。

默认值被设为10 deg。用于要调整在接近特殊方向时起动的回避动作的起始位置之时。如果设置比默认值小的值，在靠近特殊方向属性后起动回避动作。一般不需要更改参数，在超过特殊方向属性而发生错误时，可以抑制错误。

如已更改SingularityAngle的设置值，则在下次起动控制器时有效。

参阅

AvoidSingularity、SingualrityAngle函数、SingularitySpeed

SingularityAngle使用示例

```
SingularityAngle 7.0 '将特殊方向附近角度设置为7度
```

3.22.37 SingularityAngle函数

返回SingularityAngle的设置值。

格式

SingularityAngle

返回值

返回已设置的特殊方向属性附近角度(单位：deg)。

参阅

AvoidSingularity、SingularityAngle、SingularitySpeed、SingularitySpeed函数

SingularityAngle函数使用示例

```
Real currSingularityAngle
currSingularityAngle = SingularityAngle
```

3.22.38 SingularityDist

用于设置奇点通过功能所需的特殊点附近距离。

格式

SingularityDist {距离设置值}

参数

距离设置值

以表达式或数值指定用于判断垂直6轴型机器人(包括N系列)与RS系列肩部特异姿势近旁所需的P点与第1关节旋转轴距离(0以上的实数，单位为mm)。

结果

如果省略参数，将显示当前的SingularityDist设置值。

说明

该命令仅在使用奇点通过功能时有效。

默认设置为30 mm。用于调整接近肩部奇点时进行的回避动作的开始位置。如果设置比默认值小的值，在接近奇点后开始回避动作。通常无需变更本参数，但在奇点通过时发生错误的情况下，可能起到抑制错误的作用。

已变更SingularityDist的设置值时，在下次启动控制器之前将保持有效。

参阅

AvoidSingularity、SingularityAngle、SingualrityAngle函数、SingularityDist函数、SingularitySpeed、SingularitySpeed函数

SingularityDist使用示例

```
SingularityDist 10.0 '将奇点附近距离设为10 mm
```

3.22.39 SingularityDist函数

用于返回SingularityDist的设置值。

格式

SingularityDist

返回值

返回已设置的奇点附近距离(单位：mm)。

参阅

SingularityDist、AvoidSingularity、SingularityAngle、SingularityAngle函数、SingularitySpeed、SingularitySpeed函数

SingularityDist函数使用示例

```
Real currSingularityDist
currSingularityDist = SingularityDist
```

3.22.40 SingularitySpeed

用于设置超过特殊方向属性功能所需的特殊点附近的角速度。

格式

SingularitySpeed {角速度设置值}

参数

角速度设置值

以表达式或数值指定用于判断垂直6轴型机器人(包括N系列)手腕特异姿势近旁所需的第4关节角度(1以上100以下的实数，单位是%)。

结果

如果省略参数，则显示当前的SingularitySpeed设置值。

说明

此命令只在使用特殊方向属性功能时有效。

默认值设置为10%。用于要调整在接近特殊方向时起动的回避动作的起始位置之时。如果设置比默认值大的值，在靠近特殊方向属性后起动回避动作。一般不需要调整、变更参数，在超过特殊方向属性而发生错误时，可以设置较小的值以抑制错误。

如已更改SingularitySpeed的设置值，则在下次起动控制器时有效。

参阅

AvoidSingularity函数、SingualrityAngle、SingularitySpeed

SingularitySpeed使用示例

```
SingularitySpeed 30 '将特异姿势近旁角速度的比例设为30%
```

3.22.41 SingularitySpeed函数

返回SingularitySpeed的设置值。

格式

SingularitySpeed

返回值

返回相对于设置的特殊方向属性附近角速度的最大角速度的比例(单位：%)。

参阅

SingularitySpeed、SingularityAngle、AvoidSingularity

SingularitySpeed函数使用示例

```
Real currSingularitySpeed
currSingularitySpeed = SingularitySpeed
```

3.22.42 SLock

用于解除指定关节的SFree。

格式
SLock 关节编号 [, 关节编号,...]

参数

关节编号
以表达式或数值指定关节编号(1～9的整数)。附加轴的S轴为8，T轴为9。

说明
当为了直接示教或安装工件等，使用SFree使关节进入SFree状态后，可以使用SLock来解除SFree状态。

如果省略关节编号，则解除所有关节的SFree状态。

如果对第3关节执行SLock，电磁制动器则会被解除。

如果执行SLock命令，则会对机器人控制参数进行初始化。详情请参阅Motor On。

垂直6轴型机器人(包括N系列)无法使用SFree命令进入SFree状态。如果执行SLock则会发生错误。

参阅
Brake、LimZ、Reset、SFree

SLock使用示例
如下所示为使用SLock命令的示例。在本例当中，要进行动作时，必须将［设置］菜单中的［系统配置］-［常规］面板的［允许一个或多个关节在刹车释放状态下动作］选件按钮设为有效。

```
Function test
   .
   .
   .
   SFree 1, 2      '将J1和J2设为SFree状态，然后移动Z和U关节以安装部件
   Go P1
   SLock 1, 2      '对J1和J2解除SFree状态
   .
   .
   .
Fend
```

3.22.43 SoftCP

用于设置SoftCP动作模式。

格式

SoftCP { On | Off }

参数

On | Off

- On：将SoftCP动作模式设为有效。
- Off：解除SoftCP动作模式。

说明

SoftCP动作模式用于抑制以高加减速度执行CP动作时的振动动作。

可进行重视轨迹跟踪性能或等速动作性能的通常的CP动作设置。以高加减速度进行动作时，可能会产生振动。为了抑制这种振动，将SpeedS或AccelS等加减速度设置限制在较低程度是一种有效的做法。但有时会要求利用某些应用程序以高加减速度执行振动更少的CP动作，而不仅仅是高轨迹跟踪性或等速性。在SoftCP动作模式下，通过利用通常的SoftCP动作来控制CP动作时的轨迹跟踪性或等速性，可减轻设为高加减速度的CP动作所带来的振动。

在SoftCP动作模式下，以下CP动作命令有效。
Move、BMove、TMove、Arc、Arc3、CVMove、Jump3CP

在通常的CP动作所引起的振动不成问题的情况下，或在重视轨迹跟踪性或等速性的应用程序中，请勿使用SoftCP动作模式。

重要事项

- CP On时连接CP动作与PTP动作的情况下

  如下所示，要连接CP动作与PTP动作时，请将SoftCP设为有效。如果未设为有效，机器人则可能因动作而产生异响。连接完成之后，请将SoftCP设为无效。

```
SoftCP On
Go P1 CP
Move P2
SoftCP Off
```

参阅

SoftCP函数

SoftCP使用示例

```
SoftCP On
Move P1
Move P2
SoftCP Off
```

3.22.44 SoftCP函数

用于返回SoftCP动作模式的状态。

格式

SoftCP

返回值

- 0 = SoftCP动作模式OFF
- 1 = SoftCP动作模式ON

参阅

SoftCP

SoftCP函数使用示例

```
If SoftCP = Off Then
    Print "SoftCP is off"
EndIf
```

3.22.45 Space$函数

用于返回指定数量的空格字符串。

格式
Space$ (指定数)

参数

指定数量
　　指定作为返回值返回的空格数。

返回值
已指定数量的空格字符串

说明
Space$用于返回指定数量的空格字符串。可利用Space$返回最多255的字符空格(字符串变量的容许范围)。

Space$用于在其它字符串之前、之后、中间添加空格。

参阅
Asc、Chr$、InStr、Left$、Len、LSet$、Mid$、Right$、RSet$、Str$、Val

Space$使用示例

```
> Print "XYZ" + Space$(1) + "ABC"
XYZ ABC

> Print Space$(3) + "ABC"
   ABC
>
```

3.22.46 Speed

用于设置/显示利用Go、Jump、Pulse命令等的PTP动作速度。

格式

(1) Speed 速度设置值 [, 转移速度, 接近速度]

(2) Speed

参数

速度设置值
　　以表达式或数值指定相对于最大动作速度(PTP动作)的比例(1～100的整数，单位：%)。
转移速度
　　以表达式或数值指定Jump命令时的转移动作速度(1～100的整数，单位：%)。可省略。仅Jump命令时可设置。
接近速度
　　以表达式或数值指定Jump命令时的接近动作速度(1～100的整数，单位：%)。可省略。仅Jump命令时可设置。

结果

如果省略参数，则显示当前的Speed设置值。

说明

Speed用于指定所有PTP动作命令的速度。其中包括有关Go、Jump、Pulse等动作命令的速度设置。速度设置是指以1～100的整数指定相对于最大速度的比例(%)。如果指定“100”，则以最大速度进行动作。

转移速度和接近速度仅适用于Jump命令。如果省略，速度设置值则为转移速度和接近速度的设置值。

下述某种情况时，Speed值会被初始化。

- 启动控制器时
- 执行Motor On
- 执行SFree、SLock、Brake
- 执行Reset、Reset Error
- 利用停止按钮或执行Quit All等结束任务

低功率模式时，Speed设置值低于默认值。即使通过命令窗口或在程序中输入高于默认值的值，也会被设为默认值。高功率模式时，动作速度为由Speed设置的值。

需要更快的动作速度时，请利用Power High设置高功率模式并关闭安全门。如果打开安全门，Speed值则被变更为默认值。

如果在低功率模式时执行Speed，则会显示信息。在下例中，由于机器人处于低功率模式，即使Speed设置值为“80”，机器人也以默认值“5”进行动作。

```
> speed 80
> speed
Low Power Mode
    80
    80     80
>
```

参阅

Accel、Go、Jump、Power、Pass、Pulse、SpeedS

Speed使用示例

可在命令窗口和程序中使用Speed。如下所示为两种情况下的使用示例。

```
Function speedtst
    Integer slow, fast, i
    slow = 10
    fast = 100
    For i = 1 To 10
        Speed slow
        Go P0
        Go P1
        Speed fast
        Go P0
        Go P1
    Next i
Fend
```

如下所示为通过命令窗口设置Speed值的示例。

```
> Speed 100,100,50     '将Z关节的下降速度设为50%
> Speed 50
> Speed
    Low Power State: Speed is limited to 5
       50
       50         50
>
```

3.22.47 Speed函数

用于返回当前设置的PTP动作速度。

格式

Speed [(参数编号)]

参数

参数编号

从下述编号中指定1个设置值编号。省略时，视为指定1。

- 1：PTP动作速度
- 2：JUMP转移速度
- 3：JUMP接近速度

返回值

返回1～100的整数值。

参阅

Speed

Speed函数使用示例

```
Function SpeedEx
    Integer savSpeed

    savSpeed = Speed(1)
    Speed 50
    Go pick
    Speed savSpeed
Fend
```

3.22.48 SpeedFactor

设置机器人动作的速度系数以及返回设置值。

格式

(1) SpeedFactor 速度比

(2) SpeedFactor

参数

速度比

以表达式或数值指定机器人动作的速度比(1～100的整数，单位：%)。

结果

如果省略参数，则显示当前的SpeedFactor值。

说明

SpeedFactor是控制器中设置的所有机器人的所有动作的速度系数。一般SpeedFactor设置为100%，各机器人、各动作命令的速度通过Speed，SpeedS等函数进行设置。SpeedFactor在以固定速度比变更变更所有机器人的所有动作的速度时有效。例如，在将速度比设为50%时，Speed 80％的动作的动作速度将变为40％。

SpeedFactor对用于考虑机器人动作的加减速度平衡的加速度，也以相同的比例进行变更。

SpeedFactor与设置操作员窗口的速度比等效，与其联动变化。

SpeedFactor在起动控制器时默认值被初始化为100%。

参阅

SpeedFactor函数

SpeedFactor使用示例

```
Function main
    Motor On
    Power High
    SpeedFactor 80

    Speed 100; Accel 100,100
    Go P1            '以Speed 80; Accel 80,80进行动作

    Speed 50; Accel 50,50
    Go P2            '以Speed 40; Accel 40,40进行动作
Fend
```

3.22.49 SpeedFactor函数

返回SpeedFactor命令的设置值。

格式

SpeedFactor

返回值

以整数值返回SpeedFactor命令的设置值。

参阅

SpeedFactor

SpeedFactor函数使用示例

```
Real savSpeedFactor

savSpeedFactor = SpeedFactor
SpeedFactor 80
Go P1
Go P2
SpeedFactor savSpeedFactor
```

3.22.50 SpeedR

用于设置/显示使用ROT时的CP动作的工具姿势变化速度。

格式

(1) SpeedR 速度设置值

(2) SpeedR

参数

速度设置值
以表达式或数值指定CP动作时的工具姿势变化速度(0.1以上的实数，单位为deg/sec)。
参数的有效设置值： 0.1～1000

结果

如果省略参数，则显示当前的SpeedR设置值。

说明

该命令仅在利用Move、Arc、Arc3、BMove、TMove、Jump3CP等动作命令指定ROT修饰参数时有效。

下述某种情况时，SpeedR值会被初始化为默认值(低速)。

- 启动控制器时
- 执行Motor On
- 执行SFree、SLock、Brake
- 执行Reset、Reset Error
- 利用停止按钮或执行Quit All等结束任务

参阅

AccelR、Arc、Arc3、BMove、Jump3CP、Power、SpeedR函数、TMove

SpeedR使用示例

```
SpeedR 200
```

3.22.51 SpeedR函数

用于返回工具姿势变化速度。

格式
SpeedR

返回值
以实值(单位：deg/sec)返回设置的速度。

参阅
AccelR、SpeedR

SpeedR函数使用示例

```
Real currSpeedR

currSpeedR = SpeedR
```

3.22.52 SpeedS

用于设置/显示Move 、Arc、Arc3、Jump3、Jump3CP等CP动作时的机械臂速度。

格式

(1) SpeedS 速度设置值 [, 转移速度, 接近速度]

(2) SpeedS

参数

速度设置值
　　以表达式或数值指定速度(整数，单位：mm/sec)。
转移速度
　　以实数或表达式指定Jump3转移速度(单位：mm/sec)。可省略。
接近速度
　　以实数或表达式指定Jump3接近速度(单位：mm/sec)。可省略。

参数的有效设置值： 0.1～2000

- 非N系列：0.1~2000
- N系列：0.1~1120

结果

如果省略参数，则显示当前的SpeedS值。

说明

SpeedS用于指定执行CP动作(Move和Arc)命令时的速度。

SpeedS值为机器人速度的指定值。指定单位为mm/sec。默认值因机器人机型而已。有关SpeedS的默认值，请参阅以下手册。
《机械手手册》

控制器电源ON时，始终自动设置该值。

下述某种情况时，SpeedS值会被初始化。

- 启动控制器时
- 执行Motor On
- 执行SFree、SLock、Brake
- 执行Reset、Reset Error
- 利用停止按钮或执行Quit All等结束任务

低功率模式时，SpeedS值的默认值和设置值中的较低一方的速度有效。如果通过命令窗口或在程序中指定较高的速度设置，速度则会被设为默认值。高功率模式时，动作速度为SpeedS设置的值。

需要更快的动作速度时，请利用Power High设置高功率模式并关闭安全门。如果打开安全门，SpeedS值则被变更为默认值。

参阅

AccelS, Arc, Jump3, Move, Speed

SpeedS使用示例

可在命令窗口和程序中使用SpeedS。如下所示为两种情况下的使用示例。

```
Function speedtst
  Integer slow, fast, i
  slow = 50
```

```
  fast = 500
  For i = 1 To 10
    SpeedS slow
    Move P0
    Move P1
    SpeedS fast
    Move P0
    Move P1
  Next i
Fend
```

也可以利用命令窗口设置SpeedS。

```
> speeds 1000
> speeds 500
> speed 30      '设置PTP动作速度
> go p0         '以PTP动作进行移动
> speeds 100    '以mm/sec单位设置直线速度
> move P1       '直线移动
```

3.22.53 SpeedS函数

用于返回SpeedS设置。

格式

SpeedS [(参数编号)]

参数

参数编号

以整数或表达式从下述各项中指定要返回的SpeedS值。可省略。

- 1：CP速度
- 2：Jump3转移速度
- 3：Jump3接近速度

返回值

返回实值。单位为mm/sec。

参阅

SpeedS

SpeedS函数使用示例

```
Real savSpeeds

savSpeeds = SpeedS

Print "Jump3 depart speed = ", SpeedS(2)
```

3.22.54 Sqr函数

用于返回平方根。

格式
Sqr (数值)

参数

数值
以实值或表达式进行指定。

返回值
用于返回平方根。

说明
用于返回平方根。

常见错误

- 负数运算符

  数值为负数时，会发生错误。

参阅
Abs, And, Atan, Atan2, Cos, Int, Mod, Not, Or, Sgn, Sin, Str$, Tan, Val, Xor

Sqr函数使用示例
如下所示为通过命令窗口使用Sqr函数的示例。

```
>print sqr(2)
 1.414214
>
```

如下所示为使用Sqr的简单程序示例。

```
Function sqrtest
    Real x
    Print "Please enter a numeric value:"
    Input x
    Print "The Square Root of ", x, " is ", Sqr(x)
Fend
```

3.22.55 ST函数

将已设置的附加轴坐标值作为点数据进行返回。

格式

ST (s坐标要素, t坐标要素)

参数

s坐标元素
　　以实值指定s轴的坐标。
t坐标元素
　　以实值指定t轴的坐标。

返回值

将已设置的附加轴坐标值作为点数据进行返回。

说明

仅在使用附加轴(S轴和T轴)时使用本函数。

采取类似Go ST(10,20)的使用方法时，附加轴移动到指定的坐标值位置，但机器人不动作。如果要同时使机器人进行动作，请进行Go XY(60,30,-50,45) : ST(10,20)这样的指定。

有关详细信息，请参阅以下手册。
《Epson RC+ 用户指南 - 附加轴》

参阅

XY函数

ST函数使用示例

```
P10 = ST(10, 20)
```

3.22.56 StartMain

用于通过后台任务执行主函数。

本命令用于高级人员。请在充分理解命令规格之后使用。

格式

StartMain 主函数名

参数

主函数名
　　指定要执行的主函数名(main～main63)。

说明

要通过程序执行本命令时，需要勾选Epson RC+的［设置］-［系统配置］-［设置控制器］-［环境］的［将高级任务控制命令设为有效］复选框。

在后台任务中利用Xqt命令执行任务时，已执行的任务也变为后台任务。如果使用本命令，则可通过后台任务将主函数作为一般任务予以执行。

已执行主函数时或通过一般任务执行本命令时，会发生错误。

> ⚠ 注意
>
> **要通过程序执行StartMain命令时，请理解命令的规格并确认可作为系统执行主函数的条件已经备齐。如果一直采取按循环执行命令等错误的使用方法，则可能会降低系统的安全性。请充分注意。**

参阅

Xqt

StartMain使用示例

```
Function bgmain

    :
  If Sw(StartMainSwitch) = On And Sw(ErrSwitch) = Off Then
    StartMain main
  EndIf
    :

Fend
```

3.22.57 Stat函数

用于返回控制器的状态。

格式

Stat (地址)

参数

地址

指定表示控制器状态的地址(0～2的整数)。

返回值

返回表示控制器状态的4字节值。(参照下表。)

说明

Stat命令用于返回下表所示的数据。请参阅各个位的内容。

地址	Bit		位为ON时表示的控制器状态
0	0-15	&H1-&H8000	任务1～16正在执行(Xqt)或处于Halt状态
	16	&H10000	正在执行任务
	17	&H20000	暂停状态
	18	&H40000	错误状态
	19	&H80000	TEACH模式
	20	&H100000	紧急停止状态
	21	&H200000	低功率模式(Power Low)
	22	&H400000	安全门输入处于ON状态
	23	&H800000	Enable开关处于ON状态
	24	&H1000000	未定义
	25	&H2000000	未定义
	26	&H4000000	测试模式
	27	&H8000000	T2模式状态
	28～31		未定义

地址	Bit		位为ON时表示的控制器状态
1	0	&H1	Jump...Sense语句条件成立时目标坐标上方停止的记录。(如果执行后续的Jump语句，该记录则会被删除。)
	1	&H2	Go/Jump/Move...Till语句条件成立时动作中途停止的记录。(如果执行后续的Go/Jump/Move...Till语句，该记录则会被删除。)
	2	&H4	未定义
	3	&H8	Trap语句条件成立时动作中途停止的记录
	4	&H10	Motor On 状态
	5	&H20	目前处于原点位置
	6	&H40	低功率状态
	7	&H80	未定义
	8	&H100	正进行第4关节励磁
	9	&H200	正进行第3关节励磁
	10	&H400	正进行第2关节励磁
	11	&H800	正进行第1关节励磁
	12	&H1000	正进行第6关节励磁
	13	&H2000	正进行第5关节励磁
	14	&H4000	正进行第T关节励磁
	15	&H8000	正进行第S关节励磁
	16	&H10000	正进行第7关节励磁
	17～31		未定义
2	0-15	&H1-&H8000	任务17～32正在执行(Xqt)或处于Halt状态

参阅

EStopOn函数、TillOn函数、PauseOn函数、SafetyOn函数

Stat函数使用示例

```
Function StatDemo

    Integer rbt1_sts
    rbt1_sts = RShift((Stat(0) And &H070000), 16)
    Select TRUE
        Case (rbt1_sts And &H01) = 1
            Print "Tasks are running"
        Case (rbt1_sts And &H02) = 2
            Print "Pause Output is ON"
        Case (rbt1_sts And &H04) = 4
            Print "Error Output is ON"
    Send
Fend
```

3.22.58 Str$函数

用作返回将数值转换为字符串的函数。

格式
Str$ (数值)

参数

数值
以表达式或直接以数值指定转换为字符串的数值。

返回值
将数值转换为字符串并进行返回。

说明
Str$用于将数值(正负均可)转换为字符串。

参阅
Abs、Asc、Chr$、InStr、Int、Left$、Len、Mid$、Mod、Right$、Sgn、Space$、Val

Str$函数使用示例
下例所示为将几个不同的数值转换为字符串并在屏幕上显示。

```
Function strtest
    Integer intvar
    Real realvar
    '
    intvar = -32767
    Print "intvar = ", Str$(intvar)
    '
    realvar = 567.9987
    Print "realvar = ", Str$(realvar)
    '
Fend
```

如下所示为通过命令窗口使用Str$命令的结果。

```
> Print Str$(99999999999999)
 1.000000E+014

> Print Str$(25.999)
 25.999
```

3.22.59 String

用于将变量声明为字符串变量。

格式

String 变量名$[(数组变量的最大下标)] [, 变量名$[(数组变量的最大下标)]...]

参数

变量名$

指定声明为字符串型的变量的名称。可利用数组变量的最大下标声明到最高三维。使用下述格式。可省略。
(最大下标1, [最大下标2], [最大下标3])
由于下标从0开始，因此数元素为最大下标加上1。在所有元素数不超过以下最大值的范围内指定各最大下标。

- 本地变量：200
- 备份变量(Global Preserve)： 400
- 全局变量和模块变量：10,000

说明

String用于进行变量的字符串型声明。字符串型变量最大为255个字符。在Function开头声明本地变量。在Function之外声明全局变量和模块变量。

- String运算符

  字符串变量也可以使用下述运算符。

- +: 合并多个字符串变量。可用于字符串变量的分配语句或Print命令。

```
Example: name$ = fname$ + " " + lname$
```

- =: 比较多个字符串变量。包括Case的2个字符串完全一致时返回True。

```
Example: If temp1$ = "A" Then GoSub test
```

- < >: 比较多个字符串变量。2个字符串中1个以上的字符不同时返回True。

```
Example: If temp1$ <> "A" Then GoSub test
```

注意

- 请在变量名最后附加“$”

  请在String型变量名最后附加“$”。

参阅

Boolean、Byte、Double、Global、Int32、Int64、Integer、Long、Real、UByte、UInt32、UInt64、UShort

String使用示例

```
String password$
String A$(10)          'String型一维数组
String B$(10, 10)      'String型二维数组
String C$(5, 5, 5)     'String型三维数组

Print "Enter password:"
Input password$
If UCase$(password$) = "Epson" Then
    Call RunMaintenance
```

```
Else
    Print "Password invalid!"
EndIf
```

3.22.60 Sw函数

用作返回指定输入位状态的函数。

格式
Sw (位编号)

参数

位编号
以整数或表达式指定I/O的输入位。

返回值
如果指定的输入为ON，则返回“1”；如果为OFF，则返回“0”。

说明
Sw用于检查I/O输入的状态。Sw最常用于对通过I/O进行动作的装载机、传送带、夹爪以及其它外围装置上连接的传感器进行检查。利用Sw命令进行检查的输入状态包括“1”或“0”。分别表示装置处于ON(1)或OFF(0)状态。

参阅
In、InBCD、MemOn、MemOff、MemSw、Off、On OpBCD、Oport、Out、Wait

Sw函数使用示例
如下所示为检查输入位5并根据其结果分支程序的示例。为了进一步明确，使用“On”以替代“1”。

```
Function main
    Integer i, feed5Ready
    feed5Ready = Sw(5)
    '确认Feeder的准备是否完成
    If feed5Ready = On Then
        Call mkpart1
    Else
        Print "Feeder #5 is not ready. Please reset and"
        Print "then restart program"
    EndIf
Fend
```

如下所示为利用命令窗口的简单操作示例。

```
> print sw(5)
1
>
```

3.22.61 SyncLock

用于使用相互排他锁定，使多个任务同步。

格式

SyncLock 信号编号 [, 超时]

参数

信号编号
　　以表达式或数值指定要接收的信号编号(0～63的整数)。
超时
　　以表达式或数值(实数)指定锁定之前等待的最长待机时间。可省略。

说明

SyncLock用于进行同步锁定，以便1次只有1个任务使用通用资源。某任务用完通用资源之后，调用SyncUnlock解除锁定，以便其它任务使用该资源。

仅可利用事先自己锁定的syncID解除1个任务的锁定。

要解除任务锁定时，必须执行SyncUnlock。任务结束之后，该任务的锁定则会被解除。

如果对同一信号编号连续2次执行SyncLock，则会发生错误。

指定超时参数时，请使用Tw函数检查锁定是否成功。

注意

Epson RC+5.0在任务结束后，也不会自动解除锁定，但Epson RC+6.0, RC+7.0, Epson RC+ 8.0会自动解除。

参阅

Signal、SyncLock、Tw、Wait、WaitPos

SyncLock使用示例

下例所示为使用SyncLock和SyncUnlock设为1次只有1个任务将信息写入到记录文件中。

```
Function Main

    Xqt Func1
    Xqt Func2
Fend

Function Func1
  Long count
  Do
    Wait .5
    count = count + 1
    LogMsg "Msg from Func1, " + Str$(count)
  Loop
Fend

Function Func2
  Long count
  Do
    Wait .5
    count = count + 1
    LogMsg "Msg from Func2, " + Str$(count)
  Loop
Fend
```

```
Function LogMsg(msg$ As String)
  SyncLock 1
  OpenCom #1
  Print #1, msg$
  CloseCom #1
  SyncUnlock 1
Fend
```

如下所示为使用超时(选项)的SyncLock的示例。使用Tw检查锁定是否成功。通过使用超时，可在等待锁定资源的时间内执行其它程序。

```
Function MySyncLock(syncID As Integer)
  Do
    SyncLock syncID, .5
    If Tw = 0 Then
      Exit Function
    EndIf
    If Sw(1) = On Then
      Off 1
    EndIf
  Loop
Fend
```

3.22.62 SyncUnlock

用于解除事先由SyncLock锁定的信号编号锁定。

格式

SyncUnlock 信号编号

参数

信号编号
以表达式或数值指定要接收的信号编号(1～63的整数)。

说明

SyncUnlock用于解除事先由SyncLock锁定的信号编号的锁定。

如果不是事先锁定的信号编号，则不能解除锁定。

参阅

Signal、SyncLock、Wait、WaitPos

SyncUnlock使用示例

```
Function Main

    Xqt task
    Xqt task
    Xqt task
    Xqt task
Fend

Function task
  Do
    SyncLock 1
    Print "resource 1 is locked by task", MyTask
    Wait .5
    SyncUnlock 1
  Loop
Fend
```

3.22.63 SyncRobots

用于开始动作预约中的机器人动作。

格式

SyncRobots 机器人编号 [, 机器人编号] [,...]

SyncRobots All

参数

机器人编号
　　以表达式或数值(整数)指定开始动作机器人的编号。
All
　　指定动作预约中的所有机器人。

说明

SyncRobots用于使用支持各动作命令的SYNC参数开始已进行动作预约的机器人动作。按相同的时序，由SyncRobots指定的机器人开始动作。由于没有任务切换的影响，与使用通常多任务程序I/O信号的事件等待的同步相比，可更准确地实现机器人动作开始的同步。

如果指定未进行动作预约的机器人编号，则会发生错误。

参阅

SyncRobots函数

SyncRobots使用示例

下例所示为使用动作命令的SYNC参数和SyncRobots使2台机器人同时开始动作。

```
Function Main
    Xqt Func1
    Xqt Func2
    Do
       Wait 0.1
       If (SyncRobots And &H03) = &H03 Then
        Exit Do
      EndIf
    Loop
SyncRobots 1,2
Fend

Function Func1
  Robot 1
  Motor On
  Go P1 SYNC
Fend

Function Func2
  Robot 2
  Motor On
  Go P1 SYNC
Fend
```

3.22.64 SyncRobots函数

用作返回动作预约中的机器人状态的函数。

格式

SyncRobots

返回值

机器人处于动作预约状态时，返回对应于其编号的位“1”，未处于预约状态时，返回设为“0”的整数值。

- 位0：机器人编号1
- 位1：机器人编号2
- :
- 位15：机器人编号16

说明

SyncRobots函数用于检查由机器人动作命令的SYNC参数设置的机器人动作预约状态。由对应于机器人编号的位的状态来表示利用SyncRobots进行检查的状态。各个位分别表示机器人处于动作预约状态(1)或未处于预约状态(0)。可利用SyncRobots命令开始预约中的机器人动作。

参阅

SyncRobots

SyncRobots函数使用示例

下例所示为使用动作命令的SYNC参数和SyncRobots使2台机器人同时开始动作。

```
Function Main
    Xqt Func1
    Xqt Func2
    Do
       Wait 0.1
       If (SyncRobots And &H03) = &H03 Then
        Exit Do
      EndIf
    Loop
SyncRobots 1,2
Fend

Function Func1
  Robot 1
  Motor On
  Go P1 SYNC
Fend

Function Func2
  Robot 2
  Motor On
  Go P1 SYNC
Fend
```

3.22.65 SysConfig

显示系统设置参数。

格式
SysConfig

结果
显示系统设置参数。

说明
作为系统控制数据，显示当前设置的值。机器人交货时或变更数据时，请保存该数据。通过Epson RC+的［工具］-［维护］菜单 -［备份控制器］进行保存。

显示下述数据。(下述数值为示例，实际数值因控制器机型而异。)

```
' Version:
'   Firmware 1, 0, 0, 0

' Options:
'   External Control Point
'   RC+ API

' HOUR: 414.634

' Controller:
'   Serial #: 0001

' ROBOT 1:
' Name: Mnp01
' Model: PS3-AS10
' Serial #: 0001
' Motor On Time: 32.738
'   Motor 1: Enabled, Power = 400
'   Motor 2: Enabled, Power = 400
'   Motor 3: Enabled, Power = 200
'   Motor 4: Enabled, Power = 50
'   Motor 5: Enabled, Power = 50
'   Motor 6: Enabled, Power = 50

  ARCH 0, 30, 30
  ARCH 1, 40, 40
  ARCH 2, 50, 50
  ARCH 3, 60, 60
  ARCH 4, 70, 70
  ARCH 5, 80, 80
  ARCH 6, 90, 90
  ARMSET 0, 0, 0, 0, 0, 0
  HOFS 0, 0, 0, 0, 0, 0
  HORDR 63, 0, 0, 0, 0, 0
  RANGE -7427414, 7427414, -8738134, 2621440, -3145728, 8301227, -5534152, 5534152,
-3640889, 3640889, -6553600, 6553600
  BASE 0, 0, 0, 0, 0, 0
  WEIGHT 2, 0
  INERTIA  0.1, 0
  XYLIM 0, 0, 0, 0, 0, 0


' Extended I/O Boards:
'   1: Installed
'   2: Installed
'   3: None installed
```

```
'   4: None installed

' Fieldbus I/O Slave Board:
'   Installed
'   Type: PROFIBUS

' Fieldbus I/O Master Board:
'   None installed

' RS232C Boards:
'   1: Installed
'   2: None installed

' PG Boards:
'   1: None installed
'   2: None installed
'   3: None installed
'   4: None installed
```

SysConfig使用示例

```
> SysConfig
```

3.22.66 SysErr函数

用于返回最新的错误状态或警告状态。

格式
SysErr [(信息编号)]

参数

信息编号
以整数值指定获取错误代码还是警告代码。(可省略)0：错误代码 (省略时)1：警告代码

返回值
以整数值返回控制器的错误代码或警告代码。

说明
本函数仅用于NoEmgAbort任务(执行Xqt时指定 NoEmgAbort以开始的特别任务)与后台任务。

控制器的错误代码和警告代码是指LCD上显示的错误代码和警告代码。

未发生错误或警告时，返回“0”。

参阅
ErrMsg$、ErrorOn、Trap、Xqt

SysErr函数使用示例
下例所示为监视控制器的错误状态，并在发生错误时，根据错误编号对I/O进行ON/OFF操作的程序。

注意

- Forced标志

  本程序示例所示为在On/Off命令中指定的Forced标志。

  发生错误及紧急停止期间或安全门打开时，I/O输出会发生变化，因此，在系统设计方面需要注意。

- 发生错误之后的处理

  如本例所示，发生错误并进行必要的处理之后，请立即结束任务。

```
Function main

Xqt ErrorMonitor, NoEmgAbort
:
:
Fend

Function ErrorMonitor
  Wait ErrorOn
  If 4000 < SysErr Then
    Print "Mortion Error = ", SysErr
    Off 10, Forced
    On 12, Forced
  Else
    Print "Other Error = ", SysErr
    Off 11, Forced
    On 13, Forced
  EndIf
```

```
Fend
```

3.23 T

3.23.1 Tab$函数

用于返回指定数量的制表符字符串。

格式
Tab$ (指定数量)

参数

指定数量
以整数值指定制表符的数量。

返回值
返回包括制表符字符在内的字符串。

说明
用于返回指定数量的制表符字符串。

参阅
Left$、Mid$、Right$、Space$

Tab$函数使用示例

```
Print "X", Tab$(1), "Y"
Print
For i = 1 To 10
    Print x(i), Tab$(1), y(i)
Next i
```

3.23.2 Tan函数

用于返回已指定数值的正切值。

格式
Tan (角度)

参数

角度
　　指定表示角度的实值。

返回值
以数值返回表示指定角度的数值的正切值。

说明
Tan用于返回正切值。参数单位必须为弧度。

要将弧度转换为角度时，需要使用RadToDeg函数。

参阅
Abs、Atan、Atan2、Cos、Int、Mod、Not、Sgn、Sin、Sqr、Str$、Val

Tan函数使用示例

```
Function tantest
    Real num
    Print "Enter number in radians to calculate tangent for:"
    Input num
    Print "The tangent of ", num, "is ", Tan(num)
Fend
```

如下所示为通过命令窗口使用Tan函数的示例。

```
> print tan(0)
0.00
> print tan(45)
1.6197751905439
>
```

3.23.3 TargetOK函数

用于返回可否从当前位置向目标坐标进行PTP (point to point) 动作。

格式

TargetOK (目标坐标)

参数

目标坐标
以点数据指定调查可否进行动作的目标坐标。

返回值

可从当前位置向目标坐标进行PTP动作时返回“True”，除此之外时返回“False”。

说明

实际移动(机器人)之前确认可否到达目标坐标或姿势。但未顾及到达目标坐标的轨迹。

参阅

CurPos、FindPos、InPos、WaitPos

TargetOK函数使用示例

```
If TargetOK(P1) Then
  Go P1
EndIf

If TargetOK(P10 /L /F) Then
  Go P10 /L /F
EndIf
```

3.23.4 TaskDone函数

用作任务结束的确认函数。

格式

TaskDone (任务识别符)

参数

任务识别符

以整数值或表达式指定任务名或任务编号。任务名为Xqt语句使用的函数名，或通过运行窗口或操作员窗口启动的函数名。

任务编号的指定(整数)

- 一般任务：1～32
- 后台任务：65～80
- Trap任务：257~267

返回值

如果任务已结束，则返回“True”；如果不是，则返回“False”。

说明

TaskDone用于确认任务是否结束。

参阅

TaskState、TaskWait

TaskDone函数使用示例

```
Xqt 2, conveyor
Do
    .
    .
Loop Until TaskDone(conveyor)
```

3.23.5 TaskInfo函数

用于返回任务的状态信息。

格式

TaskInfo$ (任务识别符, 索引)

参数

任务识别符

以整数值指定任务名或任务编号。任务名为Xqt语句使用的函数名，或通过运行窗口或操作员窗口启动的函数名。
任务编号的指定(整数)

- 一般任务：1～32
- 后台任务：65～80
- Trap任务：257~267

索引

以整数值指定要检索的信息索引。

返回值

以整数值返回指定的信息。

说明

索引	说明
0	任务编号
1	0 - 后台任务 1 - 一般任务、NoPause任务或NoEmgAbort任务
2	任务类型 0 - 一般任务未利用Xqt指定任何内容或指定Normal开始的任务 1 - NoPause任务利用Xqt指定NoPause开始的任务 2 - NoEmgAbort任务利用Xqt指定NoEmgAbort开始的任务 3 - Trap任务 4 - 后台任务
3	-1 - 未执行指定的任务 1 - 正在执行指定的任务 2 - 指定的任务正在等待事件 3 - 指定的任务暂停或处于Halt状态 4 - 指定的任务处于快速暂停状态 5 - 指定的任务处于错误状态
4	事件待机期间发生超时(与TW相同)
5	事件等待时间(msec)
6	任务选择的机器人编号
7	任务正在使用的机器人编号

参阅

CtrlInfo、RobotInfo、TaskInfo$函数

TaskInfo函数使用示例

```
If (TaskInfo(1, 3) <> 0 Then
  Print "Task 1 is runnning"
Else
  Print "Task 1 is not running"
EndIf
```

3.23.6 TaskInfo$函数

用于返回任务的文本信息。

格式

TaskInfo$ (任务识别符, 索引)

参数

任务识别符

以整数值指定任务名或任务编号。任务名为Xqt语句使用的函数名，或通过运行窗口或操作员窗口启动的函数名。
任务编号的指定(整数)

- 一般任务：1～32
- 后台任务：65～80
- Trap任务：257~267

索引

以整数值指定要检索的信息索引。

返回值

以字符串返回指定的信息。

说明

如下表所示为可利用TaskInfo$检索的信息。

索引	说明
0	任务名
1	开始日期时间
2	正在执行的函数名
3	包括函数在内的程序的行编号

参阅

CtrlInfo、RobotInfo、TaskInfo

TaskInfo$函数使用示例

```
Print "Task 1 started: "TaskInfo$(1, 1)
```

3.23.7 TaskState函数

用作获取任务当前状态的函数。

格式

TaskState (任务识别符)

参数

任务识别符

以整数值指定任务名或任务编号。任务名为Xqt语句使用的函数名，或通过运行窗口或操作员窗口启动的函数名。
任务编号的指定(整数)

- 一般任务：1～32
- 后台任务：65～80
- Trap任务：257~267

返回值

- 0：未执行指定任务
- 1：正在执行指定任务
- 2：指定任务正处在事件待机状态
- 3：指定任务正处在暂停状态
- 4：指定任务正处于快速暂停状态
- 5：指定任务处于错误状态

说明

TaskState用于获取由任务编号或任务名指定的任务的当前状态。

参阅

TaskDone、TaskWait

TaskState函数使用示例

```
If TaskState(conveyor) = 0 Then
    Xqt 2, conveyor
EndIf
```

3.23.8 TaskWait

用于等待指定任务的结束。

格式

TaskWait (任务识别符)

参数

任务识别符

以整数值或表达式指定任务名或任务编号。任务名为Xqt语句使用的函数名，或通过运行窗口或操作员窗口启动的函数名。

任务编号的指定(整数)

- 一般任务：1～32
- 后台任务：65～80
- Trap任务：257~267

参阅

TaskDone、TaskState

TaskWait使用示例

```
Xqt 2, conveyor
TaskWait conveyor
```

3.23.9 TC

用于设置转矩控制模式，以及显示当前模式。

格式
(1) TC { On | Off }

(2) TC

参数

On | Off
- On：将转矩控制模式设为有效。
- Off：将转矩控制模式设为无效。

返回值
省略参数时，显示当前的转矩控制模式。

说明
利用TC On/Off将转矩控制模式设为有效、无效。

转矩控制模式是指通过设置电动机输出限制值以产生固定力的模式，用于以固定的力将夹具末端按压在对象物上，或在保持夹具末端与对象物相接触的同时，进行与对象物动作不匹配的动作等。

将转矩控制设为有效之前，请利用TCLim设置转矩限制值。

即使在转矩控制模式期间执行动作命令，机器人也会进行动作，以定位到目标位置。机器人接触对象物并且电动机输出达到转矩限制值时，机器人停止动作并保持固定转矩。

下述某种情况时，将转矩控制模式设为无效。
- 启动控制器时
- 执行Motor On
- 执行SFree、SLock、Brake
- 执行Reset、Reset Error
- 利用停止按钮或执行Quit All等结束任务

参阅
TCLim、TCSpeed

TC使用示例1

```
Speed 5
Go ApproachPoint

'将z轴转矩限制值设为20。
TCLim -1, -1, 20, -1

TC On
Go TargetPoint
Wait 3
Go ApproachPoint
TC Off
```

注意

- 在装有Safety板的控制器中检测到位置异常时

请修改程序，确保“机器人的当前位置”和“机器人当前的动作目标位置”之间相距不远。

如果相距过远，Safety板将判断为故障，并显示“错误No.9801 Safety板检测到位置错误”的错误。(安装有Safety板的控制器)

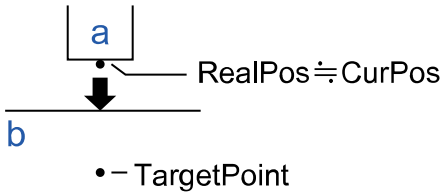
机器人的当前位置可以通过RealPos函数获取。

机器人的当前运动目标位置可以通过CurPos函数获取。

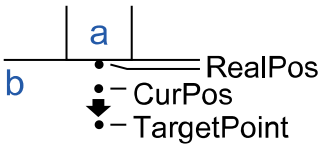
使用SCARA机器人时，建议使RealPos的差和CurPos的差不超过J1:10度、J2:10度、J3:40mm、J4:90度。

发生错误的值因机型而异。在不发生错误的范围内，可分别对各机型进行尽可能大的设置。

使用TCSpeed限制速度时，会导致在接触到目标物体之前，RealPos和CurPos之间产生差异。请不要使用TCSpeed，或者在使用TCSpeed时将其设置为与Speed相同的值。



1. 不接触目标部分时



2. 接触目标部分时

符号	说明
a	末端夹具
b	目标物体

1. 机器人的当前位置(RealPos)靠近机器人的当前动作目标位置(CurPos)。

2. 机器人的当前位置(RealPos)保持与对象物接触的位置。
   机器人当前的动作目标位置(CurPos)向TagetPoint前进。
   该距离拉开一定程度，将发生错误No.9801。

使用时不引发Safety板的位置异常的样本程序如下所示。

使用Till命令，使“机器人的当前位置”和“机器人的当前动作目标位置”的差达到一定程度并进至下一个处理，即可避免错误。

以尽可能小的值设置该差异，从而尽快进至下一个处理，有助于提高经过时间。

TC使用示例2

```
Speed 5
Go ApproachPoint

'将Z轴扭矩限制值设为20%。
TCLim -1, -1, 20, -1

Xqt posDiffChk(10.0) '启动位置差检测任务。达到J3轴10.0mm或更大差时添加标志

TC On
Go TargetPoint Till MemSw(0)
Wait 3
Go ApproachPoint
TC Off
```

```
Function posDiffChk(Zth As Double)
    Do
        If (Abs(CZ(RealPos) - CZ(CurPos)) > Zth) Then
'"机器人的当前位置"和"机器人的当前目标位置"的差是否已超过阈值?
            MemOn (0) ' 位置偏差大标志 ON
        Else
            MemOff (0) ' 位置偏差大标志 清除
        EndIf
        Wait 0.01
    Loop
Fend
```

3.23.10 TCLim

用于设置转矩控制模式下各关节转矩限制值。

格式

TCLim [第1关节转矩限制值, 第2关节转矩限制值, 第3关节转矩限制值, 第4关节转矩限制值 [, 第5关节转矩限制值] [, 第6关节转矩限制值], [, 第7关节转矩限制值] [, 第8关节转矩限制值] [, 第9关节转矩限制值]]

参数

第1关节转矩限制值
以表达式或数值指定相对于瞬时最大转矩的比例(1～100的整数，单位：%)。为-1时，将转矩限制值设为无效，变为通常的位置控制模式。
第2关节转矩限制值
以表达式或数值指定相对于瞬时最大转矩的比例(1～100的整数，单位：%)。为-1时，将转矩限制值设为无效，变为通常的位置控制模式。
第3关节转矩限制值
以表达式或数值指定相对于瞬时最大转矩的比例(1～100的整数，单位：%)。为-1时，将转矩限制值设为无效，变为通常的位置控制模式。
第4关节转矩限制值
以表达式或数值指定相对于瞬时最大转矩的比例(1～100的整数，单位：%)。为-1时，将转矩限制值设为无效，变为通常的位置控制模式。
第5关节转矩限制值
可省略。以表达式或数值指定相对于瞬时最大转矩的比例(1～100的整数，单位：%)。为-1时，将转矩限制值设为无效，变为通常的位置控制模式。
第6关节转矩限制值
可省略。以表达式或数值指定相对于瞬时最大转矩的比例(1～100的整数，单位：%)。为-1时，将转矩限制值设为无效，变为通常的位置控制模式。
第7关节转矩限制值
可省略。以表达式或数值指定相对于瞬时最大转矩的比例(1～100的整数，单位：%)。为-1时，将转矩限制值设为无效，变为通常的位置控制模式。
第8关节转矩限制值
可省略。以表达式或数值指定相对于S轴的瞬时最大转矩的比例(1～100的整数，单位：%)。为-1时，将转矩限制值设为无效，变为通常的位置控制模式。
第9关节转矩限制值
可省略。以表达式或数值指定相对于T轴的瞬时最大转矩的比例(1～100的整数，单位：%)。为-1时，将转矩限制值设为无效，变为通常的位置控制模式。

返回值

省略参数时，显示当前的转矩限制值。

说明

请在TC Off时设置转矩限制值。如果设为TC On，则设置的值有效。

如果设置值过低，机器人则不会进行动作，动作命令在到达目标位置之前结束。

下述某种情况时，TCLim设置值会被初始化。

- 启动控制器时
- 执行Motor On
- 执行SFree、SLock、Brake
- 执行Reset、Reset Error
- 利用停止按钮或执行Quit All等结束任务

参阅

TC、TCLim函数、TCSpeed

TCLim使用示例

```
Speed 5
Go ApproachPoint

' 将z轴扭矩限制值设为20%。
TCLim -1, -1, 20, -1

TC On
Go TargetPoint
Wait 3
Go ApproachPoint
TC Off
```

注意

- 在装有Safety板的控制器中检测到位置异常时

  请修改程序，确保“机器人的当前位置”和“机器人当前的动作目标位置”之间相距不远。

  如果相距过远，Safety板将判断为故障，并显示“错误No.9801 Safety板检测到位置错误”的错误。(安装有Safety板的控制器)

  详情请参阅TC命令的说明。

3.23.11 TCLim函数

用于返回指定关节转矩限制值。

格式

TCLim (关节编号)

参数

关节编号
以表达式或数值指定要获取转矩限制值的关节的关节编号。附加轴的S轴为8，T轴为9。

返回值

返回表示当前转矩限制值的1～100的整数。转矩限制值无效时返回-1。

参阅

TC、TCLim、TCSpeed

TCLim函数使用示例

```
Print "当前的Z轴转矩限制值：", TCLim(3)
```

3.23.12 TCPSpeed函数

用于返回计算出来的当前工具中心点(TCP)速度。

格式
TCPSpeed

返回值
以实值返回计算得出的当前工具中心点速度。(单位：mm/秒)

说明
以mm/秒 单位返回执行CP动作命令时计算的工具中心点速度。

CP动作命令是指Move、TMove、Arc、Arc3、CVMove和Jump3CP。该速度与实际速度不同。该速度是调用函数时系统按计划进行动作的工具中心点速度。

未顾及电动机的跟踪延迟。如果机器人执行PTP动作命令，将返回“0”。

即使使用附加轴，仅返回机器人的移动速度。比如，即使在移动轴上使用附加轴，也不考虑附加轴移动速度。

参阅
AccelS、CurPos、InPos、SpeedS

TCPSpeed函数使用示例

```
Function MoveTest
AccelS 4000, 4000
SpeedS 200
Xqt ShowTCPSpeed
Do
  Move P1
  Move P2
Loop
Fend

Function ShowTCPSpeed
  Do
    Print "Current TCP speed is: ", TCPSpeed
    Wait .1
  Loop
Fend
```

3.23.13 TCSpeed

用于设置转矩控制期间的速度限制值。

格式
TCSpeed [速度]

参数

速度
以表达式或数值指定相对于最大速度的比例(1～100的整数，单位：%)。

说明
转矩控制期间，与Speed命令等的速度设置无关，限制为由TCSpeed设置的速度。

如果在转矩控制期间检测到大于限制值的速度，则会发生错误。

下述某种情况时，TCSpeed设置值会被初始化为100%。

- 启动控制器时
- 执行Motor On
- 执行SFree、SLock、Brake
- 执行Reset、Reset Error
- 利用停止按钮或执行Quit All等结束任务

参阅
TC、TCLim、TCSpeed函数

TCSpeed使用示例

```
Speed 5
Go ApproachPoint

' 将Z轴转矩限制值设为20%。
TCLim -1, -1, 20, -1
' 将进行转矩控制期间的速度设为5% (与Speed设置相同)。
TcSpeed 5

TC On
Go TargetPoint
Wait 3
Go ApproachPoint
TC Off
```

注意

- 在装有Safety板的控制器中检测到位置异常时

  请修改程序，确保“机器人的当前位置”和“机器人当前的动作目标位置”之间相距不远。

  使用TCSpeed限制速度时，“机器人的当前位置”和“机器人当前的动作目标位置”之间会产生差异。请不要使用TCSpeed，或者在使用TCSpeed时将其设置为与Speed相同的值。

  如果相距过远，Safety板将判断为故障，并显示“错误No.9801 Safety板检测到位置错误”的错误。(安装有Safety板的控制器)

  详情请参阅TC命令的说明。

3.23.14 TCSpeed函数

用于返回转矩控制期间的速度限制值。

格式

TCSpeed

返回值

返回表示当前速度限制值的1～100的整数。

参阅

TC、TCSpeed、TCLim

TCSpeed使用示例

```
Integer var
var = TCSpeed
```

3.23.15 TeachOn函数

用于返回示教模式的状态。

格式
TeachOn

返回值
如果处于示教模式，则返回“True”；如果不是，则返回“False”。

说明
本函数仅用于后台任务。

参阅
ErrorOn, EstopOn, SafetyOn, Xqt

TeachOn函数使用示例
下例所示为监视控制器示教模式，并在切换为示教模式时对I/O进行ON/OFF操作的程序。

```
Function BGMain
  Do
    Wait 0.1
    If TeachOn = True Then
      On teachBit
    Else
      Off teachBit
    EndIf
    If SafetyOn = True Then
      On safetyBit
    Else
      Off safetyBit
    EndIf
    If PauseOn = True Then
      On PauseBit
    Else
      Off PauseBit
    EndIf
  Loop

Fend
```

3.23.16 TGo

用于在当前工具坐标系上执行偏移PTP动作。

格式

TGo 目标坐标 [CP] [Till | Find] [!并行处理!] [SYNC]

参数

目标坐标
　　使用点数据，指定动作的目标位置。
CP
　　指定路径运动。可省略。
Till | Find
　　记述Till或Find表达式。可省略。

```
Till | Find
Till Sw(表达式) = {On | Off}
Find Sw(表达式) = {On | Off}
```

! 并行处理!
　　动作期间可附加并行处理语句，以执行I/O等命令。可省略。
SYNC
　　预约动作命令。在通过SyncRobots开始动作之前，机器人不进行动作。

说明

用于在当前工具坐标系上执行偏移PTP动作。

无视由点数据提供的姿势标志并保持当前的姿势标志。但是，垂直6轴型机器人(包括N系列)会自动变更姿势标志，以减小关节移动量。

通过使用Till修饰符，可在Till条件成立时于动作中途对机器人进行减速停止，完成TGo动作。

使用Find修饰符并且动作期间Find条件成真时，将点数据保存到FindPos中。

可使用!并行处理!，与动作并行执行其它处理。

如果附加了CP参数，则可在开始动作减速时叠加后续动作命令的加速。此时，不对目标坐标进行定位。

参阅

Accel、CP、Find、!并行处理!、P#=指定点、Speed、Till、TMove、Tool

TGo使用示例

```
> TGo XY(100, 0, 0, 0)  '(在工具坐标系中)向X方向移动100 mm
Function TGoTest

  Speed 50
  Accel 50, 50
  Power High

  Tool 0
  P1 = XY(300, 300, -20, 0)
  P2 = XY(300, 300, -20, 0) /L

  Go P1
  Print Here
  TGo XY(0, 0, -30, 0)
  Print Here

  Go P2
```

```
  Print Here
  TGo XY(0, 0, -30, 0)
  Print Here

Fend
```

[输出结果]

```
 X:  300.000 Y:  300.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /R /0
 X:  300.000 Y:  300.000 Z:  -50.000 U:    0.000 V:    0.000 W:    0.000 /R /0
 X:  300.000 Y:  300.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /L /0
 X:  300.000 Y:  300.000 Z:  -50.000 U:    0.000 V:    0.000 W:    0.000 /L /0
```

3.23.17 Till

用于设置/显示利用Jump、Go、Move或其它动作命令指定Till时，在动作中途停止并结束处理的条件。

格式

Till [条件表达式]

参数

事件条件表达式

指定触发的输入状态。
[条件] 比较运算符(=、<>、>=、>、<、<=)[整数表达式]
可在条件中使用下述函数或变量。

- 函数：Sw, In, InW, Oport, Out, OutW, MemSw, MemIn, MemInW, Ctr, GetRobotInsideBox, GetRobotInsidePlane, Force, AIO_In, AIO_InW, AIO_Out, AIO_OutW, Hand_On, Hand_Off, SF_GetStatus
- 变量：Byte、Int32、Integer、Long、Short、UByte、UInt32、UShort型备份变量、全局变量、模块变量

另外，可利用下述运算符对多个事件条件表达式附加掩码或进行复合组合。

- 运算符：And、Or、Xor

[例]

```
Till Sw(5) = On
Till Sw(5) = On And Till(6) = Off
```

说明

请单独记述Till 语句或记述为动作命令语句的修饰符。

Till条件表达式必须包含1个以上的上述函数。

Till条件表达式式中包括变量时，在设置Till条件时运算其值。由于可能会形成不希望有的条件，因此不建议在条件表达式中使用变量。也可以记述多个Till语句。此时，最后执行的Till条件有效。

如果省略参数，则显示当前的Till设置。

注意

- 电源ON时的Till设置

  电源ON时Till条件的初始设置为Till Sw(0) = On。输入位编号0为ON时，设为进行减速停止。

- 用于检查Till条件成立的Stat函数和TillOn函数

  执行使用Till修饰符的动作命令之后，可使用Stat函数或TillOn函数检查Till条件是否成立。

- 在条件表达式中使用变量时
  - 可使用变量类型为整数型(Byte、Int32、Integer、Long、Short、UByte、UInt32、UShort)。
  - 不能使用数组变量。
  - 不能使用本地变量。
  - 在超过0.01秒的时间内变量值未满足条件时，系统可能不能检测到变量变化。
  - 系统内可使用的变量等待数存在限制。1个系统内可使用的变量等待数量最多为64个(也包括在Wait等条件表达式中使用的变量等待)。如果超过最大数，则会在项目创建时发生错误。
  - 如果利用Byref引用执行变量等待的变量，则会发生错误。
  - 条件表达式右边的整数表达式中包括变量时，在动作命令开始时运算其值。由于可能会形成不希望有的条件，因此不建议在整数表达式中使用变量。

参阅

Find, Go, In, InW, Jump, MemIn, MemSw, Move, Stat, Sw, TillOn, SF_GetStatus

Till使用示例

如下所示为在程序中使用Till命令的示例。

```
Till Sw(1) = Off                    '设置Till条件(输入位1为OFF)
Go P1 Till                          '满足前一行的条件时停止
Till Sw(1) = On And Sw($1) = On     '设置新的Till条件
Move P2 Till                        '满足前一行的条件时停止
Move P5 Till Sw(10) = On            '满足该行的条件时停止
```

3.23.18 TillOn函数

用于返回Till的状态。

格式
TillOn

返回值
如果在即将使用Till之前的动作命令中Till条件成立，则返回True。

说明
Till条件成立时返回True。

TillOn与下述内容相同。

```
((Stat (1) And 2) <> 0)
```

参阅
EStopOn、SafetyOn、Sense、Stat、Till

TillOn函数使用示例

```
Go P0 Till Sw(1) = On
If TillOn Then
    Print "Till condition occurred during move to P0"
EndIf
```

3.23.19 Time

用于显示时间。

格式
Time

说明
以24小时标记方式显示当前的时间。

参阅
Date, Time$

Time使用示例
利用命令窗口的执行示例

```
> Time
10:15:32
```

3.23.20 Time函数

用于返回控制器的累计通电时间。

格式

Time (指定单位)

参数

指定单位

指定0～2的整数值。以0～2表示控制器的累计通电时间。

- 0：时
- 1：分
- 2：秒

说明

以整数值返回控制器的累计通电时间。

参阅

Hour

Time函数使用示例

下例所示为通过命令窗口执行的情况。

```
Function main
    Integer h, m, s

    h = Time(0)    '以小时返回通电时间
    m = Time(1)    '以分钟返回通电时间
    s = Time(2)    '以秒钟返回通电时间
    Print "This controller has been used:"
    Print h, "hours, ",
    Print m, "minutes, ",
    Print s, "seconds"
Fend
```

3.23.21 Time$函数

用于返回当前的系统时间。

格式
Time$

返回值
以24小时标记的字符串返回时间。

格式为hh:mm:ss [时:分:秒]。

参阅
Date, Date$, Time

Time$函数使用示例

```
Print "The current time is: ", Time$
```

3.23.22 TLClr

用于清除工具坐标系设置。

格式

TLClr 工具编号

参数

工具编号

以整数或表达式指定要清除的工具。(工具0为默认工具，不能清除。)

说明

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

参阅

Arm、ArmClr、ArmSet、ECPSet、Local、LocalClr、Tool、TLSet

TLClr使用示例

```
TLClr 1
```

3.23.23 TLDef函数

用于返回工具设置状态。

格式

TLDef (工具编号)

参数

工具编号
以整数或表达式指定要返回状态的工具。

返回值

如果已设置指定的工具，则返回“True”；如果未设置，则返回“False”。

参阅

Arm、ArmClr、ArmSet、ECPSet、Local、LocalClr、Tool、TLClr、TLSet

TLDef函数使用示例

```
Function DisplayToolDef(toolNum As Integer)

   If TlDef(toolNum) = False Then
      Print "Tool ", toolNum, "is not defined"
   Else
      Print "Tool ", toolNum, ": ",
      Print TlSet(toolNum)
   EndIf
Fend
```

3.23.24 TLSet

用于设置/显示工具坐标系。

格式

(1) TLSet 工具坐标系编号, 工具设置数据

(2) TLSet 工具坐标系编号

(3) TLSet

参数

工具坐标系编号
　　以1～15的整数值指定要设置的工具。(Tool 0为默认工具，不能变更。)
刀具设置数据
　　以P编号、P(表达式)、点标签或表达式指定要设置的工具坐标系的原点和方向。

结果

如果省略所有参数，则显示所有的TLSet设置。

如果只指定工具编号，则显示指定的TLSet设置。

说明

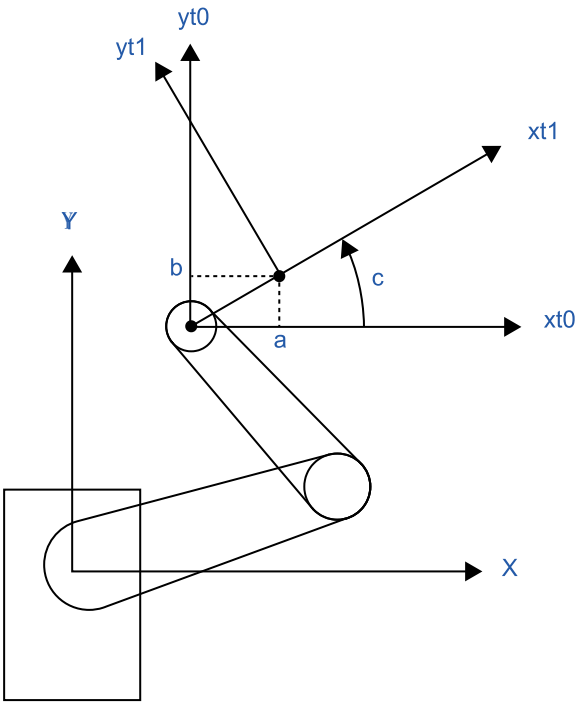
指定针对Tool 0坐标系(夹具末端坐标系)的相对原点位置和相对旋转角度，定义工具坐标系Tool 1、Tool 2 、Tool 3。

```
TLSet 1, XY(50,100,-20,30)
TLSet 2, P10 +X(20)
```

上述情况时，引用坐标值P10并在X值上加上20。无视机械臂属性和本地坐标系编号。

```
TLSET 1, XY(100,60,-20,30)
      a      b   c   d   e
```

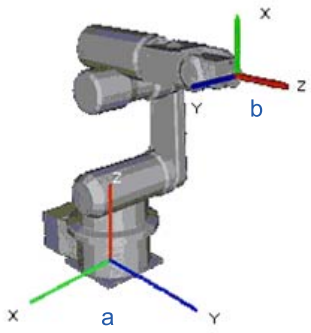
符号	说明
a	工具坐标系编号
b	工具坐标系原点的X轴方向位置(下图a)
c	工具坐标系原点的Y轴方向位置(下图b)
d	工具坐标系原点的Z轴方向位置
e	工具坐标系的旋转角度(下图c)



符号	说明
X, Y	机器人坐标系
xt0、yt0	工具0坐标系
xt1、yt1	工具1坐标系

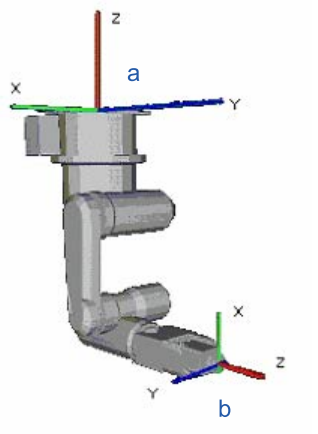
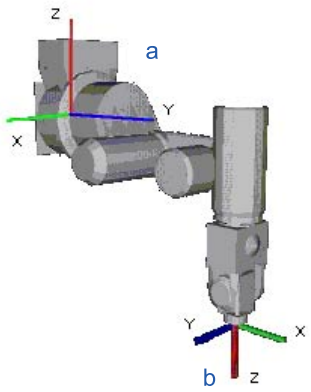
在垂直6轴型机器人中使用TLSet

在垂直6轴型机器人中，如果将所有关节设为0度位置，则以将第6关节的法兰面中心作为原点，垂直上方向为X轴，机器人坐标系X轴方向为Y轴，与第6关节法兰面垂直的方向为Z轴形成的坐标系视为工具0坐标系。(请参照下图。)



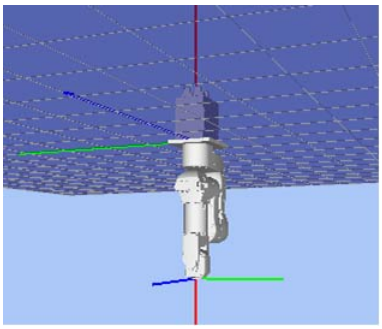
符号	说明
a	机器人坐标系
b	工具0坐标系

工具0坐标系的定义因垂直6轴型机器人的安装方法而异。

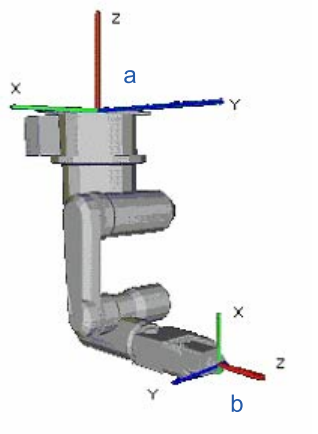
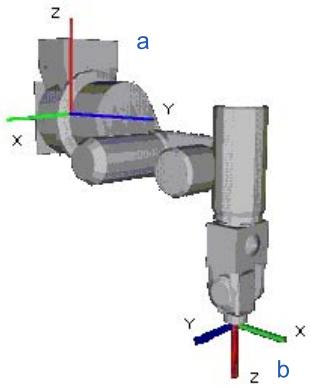
吊顶安装	侧壁安装
	
a：机器人坐标系 b：工具0坐标系	

在N系列中使用TLSet

在N系列中将所有关节设为0度位置时，机器人坐标系-X轴方向为X轴，机器人坐标系Y轴方向为Y轴，机器人坐标系-Z轴方向为Z轴的坐标系为工具0坐标系。(请参照下图。)



工具0坐标系的定义因N系列的设置方法而异。

吊顶安装	侧壁安装
	
a：机器人坐标系 b：工具0坐标系	

说明

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

注意

- TLSet值被保持。

  TLSet值被保持。要清除工具定义时，使用TLClr。

参阅

Tool、Arm、ArmSet、TLSet函数、TLClr

TLSet使用示例

如下所示为进行工具定义时的动作，以及为便于理解不进行工具定义时的动作差异而通过命令窗口进行操作的示例。

```
> TLSet 1, XY(100, 0, 0, 0)   '在工具坐标系中定义Tool 1(从夹具末端的坐标系向
                              'X方向100 mm)
> Tool 1                      '选择由TLSet定义的Tool 1
> TGo P1                      '将Tool 1的目标坐标设为P1
> Tool 0                      '设为在此后的动作中不使用工具
> Go P1                       '将U关节的中心设为P1
```

3.23.25 TLSet函数

用于返回已设置的工具坐标系数据。

格式

TLSet (工具坐标系编号)

参数

工具坐标系编号
以整数值指定工具编号。

返回值

返回工具坐标系数据。

参阅

TLSet

TLSet函数使用示例

```
P1 = TLSet(1)
```

3.23.26 TMOut

用于设置执行Wait命令时发生超时错误(错误2280)之前的时间。

格式

TMOut 秒

参数

秒

以整数值指定超时时间。范围为0～2147483。(单位：秒)

说明

TMOut用于设置执行Wait命令时发生超时错误(错误2280)之前的时间。将超时时间设为0秒时，超时不生效，Wait命令用于在指定的条件成立之前进行无限期待机。

TMOut的默认值为“0”。

参阅

In、MemSw、OnErr、Sw、TW、Wait

TMOut使用示例

```
TMOut 5
Wait MemSw(0) = On
```

3.23.27 TMove

用于在当前工具坐标系上执行偏移直线动作。

格式

TMove 目标坐标 [ROT] [CP] [Till | Find] [!并行处理!] [SYNC]

参数

目标坐标
　　使用点数据，指定动作的目标位置。
ROT
　　以工具姿势变化为优先，确定动作速度、加减速度。可省略。
CP
　　指定路径运动。可省略。
Till | Find
　　记述Till或Find表达式。可省略。

```
Till | Find
Till Sw(表达式) = {On | Off}
Find Sw(表达式) = {On | Off}
```

! 并行处理 !
　　动作期间可附加并行处理语句，以执行I/O等命令。可省略。
SYNC
　　预约动作命令。在通过SyncRobots开始动作之前，机器人不进行动作。

说明

用于在当前工具坐标系上执行偏移直线动作。

无视由点数据提供的姿势标志并保持当前的姿势标志。但是，垂直6轴型机器人(包括N系列)会自动变更姿势标志，以减小关节移动量。这与在Move命令中指定LJM修饰参数时的情况相同。因此，要进行180度以上的姿势变化时，请分多次执行。

TMove的速度/加减速度分别使用SpeedS和AccelS的设置值。有关速度与加减速度之间的关系，请参阅“注意”中的“与CP同时使用Tmove”。不过，使用ROT修饰参数时的速度和加减速度分别使用SpeedR和AccelR的设置值。此时，SpeedS和AccelS的设置值变为无效状态。

通常，移动距离为“0”，但如果仅进行姿势关节的动作，会发生错误。通过附加ROT修饰参数并以工具姿势变化的加速度为优先，可不出错误地进行动作。已经附加ROT修饰参数时，如果没有姿势变化，并且移动距离不是“0”，则会发生错误。

另外，相对于移动距离，工具姿势变化速度过大时，或指定的转速超过机械手限度时，也会发生错误。此时，请降低指定速度，或附加ROT修饰参数，并以姿势变化的加减速度为优先。

通过使用Till修饰符，可在Till条件成立时于动作中途对机器人进行减速停止，完成TMove动作。

通过使用Find修饰符并且动作期间Find条件的值成真(True)时，将点数据保存到FindPos中。

可使用!并行处理!，与动作并行执行其它处理。

注意

- 与CP同时使用TMove

  如果使用CP参数，动作命令则会在开始减速的同时将控制移交给下一语句。这在用户连续发出几个动作命令，要以一定的速度进行连续动作时非常便利。为未指定CP的TMove命令时，机械臂必须减速，以停在指定的目标位置上。

参阅

AccelS、CP、Find、!并行处理!、P#=指定点、SpeedS、TGo、Till、Tool

TMove使用示例

```
> TMove XY(100, 0, 0, 0) '(在工具坐标系中)向X方向移动100 mm
Function TMoveTest

  Speed 50
  Accel 50, 50
  SpeedS 100
  AccelS 1000, 1000
  Power High

  Tool 0
  P1 = XY(300, 300, -20, 0)
  P2 = XY(300, 300, -20, 0) /L

  Go P1
  Print Here
  TMove XY(0, 0, -30, 0)
  Print Here

  Go P2
  Print Here
  TMove XY(0, 0, -30, 0)
  Print Here

Fend
```

[输出结果]

```
 X:  300.000 Y:  300.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /R /0
 X:  300.000 Y:  300.000 Z:  -50.000 U:    0.000 V:    0.000 W:    0.000 /R /0
 X:  300.000 Y:  300.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /L /0
 X:  300.000 Y:  300.000 Z:  -50.000 U:    0.000 V:    0.000 W:    0.000 /L /0
```

3.23.28 Tmr函数

Tmr函数用于以秒为单位返回计时器开始计时之后的经过时间。

格式

Tmr (计时器编号)

参数

计时器编号
以表达式或数值指定整数(0～63)，以确定对64个计时器中的哪个计时器进行检查。

返回值

以实值(单位：秒)返回指定计时器的经过时间。计时器的范围为0～约1.7E+31。计时器的分辨率为0.001秒。

说明

用于返回指定计时器开始计时之后的经过时间。与ElapsedTime函数不同，此函数还要将程序暂停状态的时间作为经过时间来计算。

可利用TmReset重置计时器。

```
Real overhead

TmReset 0
overHead = Tmr(0)
```

参阅

ElapsedTime函数、TmReset

Tmr函数使用示例

```
TmReset 0           '重置计时器0
For i = 1 To 10     '执行10次
    GoSub Cycle
Next
Print Tmr(0) / 10   '计算并显示循环时间
```

3.23.29 TmReset

用于重置由Tmr函数使用的计时器。

格式

TmReset 计时器编号

参数

计时器编号
以整数(0～63)指定64个计时器中要重置计时器的编号。

说明

用于重置由计时器编号指定的计时器并开始计时。

Tmr函数用于获取指定计时器的经过时间。

参阅

Tmr

TmReset使用示例

```
TmReset 0           '重置计时器0
For i = 1 To 10     '执行10次
    GoSub CYL
Next
Print Tmr(0)/10     '计算并显示循环时间
```

3.23.30 Toff

用于将在LCD上执行的行的显示设为OFF。

格式
Toff

说明
如果执行Toff，则不在LCD上显示任务的执行行。

注意

- 支持的控制器型号

  不支持RC90/T/VT系列。

参阅
Ton

Toff使用示例
如下所示为通过命令窗口进行测试的示例。可通过本例理解工具设置时和未设置时的动作差异。

```
Function main
  Ton MyTask
  ...
  Toff
Fend
```

3.23.31 Ton

用于指定在LCD上显示执行行的任务。

格式

Ton 任务识别符

Ton

参数

任务识别符

以整数值或表达式指定任务名或任务编号。任务名为Xqt语句使用的函数名，或通过运行窗口或操作员窗口启动的函数名。

任务编号的指定(整数)

- 一般任务：1～32

注意

- 支持的控制器型号

  不支持RC90/T/VT系列。

说明

在初始状态下，显示任务编号1的执行行。

如果使用Ton，则可在LCD上显示指定任务的执行行。

如果省略任务识别符，则在LCD上显示执行Ton的任务的执行行。

参阅

Toff

Ton使用示例

```
Function main
  Ton MyTask
  ...
  Toff
Fend
```

3.23.32 Tool

用于选择工具或显示所选择的工具编号。

格式

(1) Tool 工具编号

(2) Tool

参数

工具编号
　　利用后续的动作命令指定使用16个工具(整数值0～15)中的哪一个。可省略。

结果

如果省略参数，则显示当前设置的工具编号。

说明

Tool用于选择由工具编号指定的工具。工具编号为“0”时，没有选择工具，所有的动作均相对于顶端关节(旋转关节)中心进行。但选择工具编号“1”、“2”、“3”等情况下，相对于进行过工具设置的工具中心点进行动作。

注意

- 电源OFF时对工具选择的影响

  即使电源OFF，所选择的工具坐标系也不会被变更。

- 小型闪存卡的使用寿命

  机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

参阅

TGo、TLSet、TMove

Tool使用示例

如下所示为通过命令窗口进行测试的示例。可通过本例理解工具设置时和未设置时的动作差异。

```
>tlset 1, 100, 0, 0, 0    '在工具坐标系中定义Tool 1(从夹具末端的坐标系向
                          'X方向100 mm)
>tool 1                   '选择由TLSet定义的Tool 1
>tgo p1                   '将Tool 1的目标坐标设为P1
>tool 0                   '设为在此后的动作中不使用工具
>go p1                    '将U关节的中央设为P1
```

3.23.33 Tool函数

用于返回当前设置的工具编号。

格式

Tool

返回值

以整数值返回工具编号。

参阅

Tool

Tool函数使用示例

```
Integer savTool

savTool = Tool
Tool 2
Go P1
Tool savTool
```

3.23.34 Trap(用户定义触发)

用于定义中断以及发生中断时的处理。

如果使用Trap命令，则可通过事件发生跳跃到标签或调用函数。Trap命令包括2种类型。

- 将用户定义的输入状态设为触发的4个Trap
- 将系统状态设为触发的7个Trap。

本项目说明用户定义触发的Trap。

格式

Trap Trap编号, 条件表达式 GoTo标签

Trap Trap编号, 条件表达式 Call函数名

Trap Trap编号, 条件表达式 Xqt函数名

Trap Trap编号

参数

Trap编号
　以表达式或直接以数值指定Trap编号(1～4的整数)。(SPEL+支持最多4个同时有效的Trap。)
事件条件表达式
　指定触发的输入状态。
　[条件] 比较运算符(=、<>、>=、>、<、<=)[整数表达式]
　可在条件中使用下述函数或变量。
　- 函数：Sw, In, InW, Oport, Out, OutW, MemSw, MemIn, MemInW, Ctr, GetRobotInsideBox, GetRobotInsidePlane, AIO_In, AIO_InW, AIO_Out, AIO_OutW, Hand_On, Hand_Off, SF_GetStatus
　- 变量：Byte, Int32, Integer, Long, Short, UByte, UInt32, UShort型备份变量、全局变量、模块变量

　另外，可利用下述运算符对多个事件条件表达式附加掩码或进行复合组合。
　- 运算符：And、Or、Xor

　[例]

```
Trap 1, Sw(5) = On Call, TrapFunc
Trap 1, Sw(5) = On And Till(6) = Off, Call TrapFunc
```

标签
　是Trap条件成立时移交程序执行的目标的标签。
函数名称
　是Trap条件成立时进行Call或Xqt的函数。不能指定带有自变量的函数。

说明

Trap用于在条件成立时执行由GoTo、Call、Xqt等指定的中断处理。Trap条件式必须包含1个以上的上述函数。

Trap条件表达式中包括变量时，在设置Trap条件时运算其值。由于可能会形成不希望有的条件，因此不建议在条件表达式中使用变量。

一旦执行中断处理，该Trap设置则会被清除。要进行相同的中断处理时，必须再次执行Trap命令。

要取消Trap设置时，仅指定Trap编号参数并执行Trap命令。

[例]“Trap 3”表示取消Trap #3。

另外，如果退出声明的函数，则自动取消Trap Goto。

如果声明的任务结束，则取消Trap Call。

所有的任务结束之前，不能取消Trap Xqt。

指定GoTo时

在已设置Trap的任务中，对正在执行的命令进行下述处理，并将控制移交给指定目标的标签。

- 立即暂停(快速暂停)机械臂动作。
- 利用Wait或Input命令中断待机状态。
- 在移交控制之前完成所有其它命令。

指定Call时

执行与GoTo相同的处理之后，将控制移交给指定目标的函数。

函数结束时，程序返回到发生中断时的下一语句。

不能将Call用于Trap处理的函数。

另外，在Trap处理的函数中发生错误时，通过OnErr进行的错误移交变为无效状态，因此，肯定发生错误。

指定Xqt时

程序控制作为中断处理专用任务，生成指定的函数。在这种情况下，继续执行Trap命令的任务。

不能通过中断处理任务再次利用Xqt执行任务。

注意

- 针对Epson RC+4.x用户

  Epson RC+ 4.x为止的Trap Call功能在Epson RC+ 8.0中被替换为Trap Xqt。

  Epson RC+ 4.x为止的Trap GoSub功能在Epson RC+ 8.0中被删除。请使用Trap Call进行替代。

- 在条件表达式中使用变量时
  - 可使用变量类型为整数型(Byte、Int32、Integer、Long、Short、UByte、UInt32、UShort)。
  - 不能使用数组变量。
  - 不能使用本地变量。
  - 在超过0.01秒的时间内变量值未满足条件时，系统可能不能检测到变量变化。
  - 系统内可使用的变量等待数存在限制。1个系统内可使用的变量等待数量最多为64个(也包括在Wait等条件表达式中使用的变量等待)。如果超过最大数，则会在项目创建时发生错误。
  - 如果利用Byref引用执行变量等待的变量，则会发生错误。
  - 条件式右边的整数表达式包括变量时，在设置Trap条件时运算其值。由于可能会形成不希望有的条件，因此不建议在整数表达式中使用变量。

参阅

Call, GoTo, Xqt, SF_GetStatus

Trap使用示例

[例1] 用户定义的错误处理

Sw(0) Input为用户定义的错误输入。

```
Function Main
    Trap 1, Sw(0)= On GoTo EHandle  '定义Trap
    .
```

```
    .
    .
EHandle:
    On 31                               '信号塔点亮
    OpenCom #1
    Print #1, "Error is issued"
    CloseCom #1
Fend
```

[例2] 多任务使用

```
Function Main
    Trap 2, MemSw(0) = On Or MemSw(1) = On Call Feeder
    .
    .
    .
Fend
.


Function Feeder
  Select TRUE
    Case MemSw(0) = On
      MemOff 0
      On 2
    Case MemSw(1) = On
      MemOff 1
      On 3
  Send

  '再次设置下一循环的Trap
  Trap 2, MemSw(0) = On Or MemSw(1) = On Call Feeder
Fend
```

[例3] 在条件中使用全局变量

```
Global Integer gi

Function main
  Trap 1, gi = 5 GoTo THandle
  Xqt sub
  Wait 100
  Exit Function

THandle:
  Print "IN Trap ", gi

Fend

Function sub
  For gi = 0 To 10
    Print gi
    Wait 0.5
  Next
Fend
```

3.23.35 Trap(系统状态触发)

用于定义中断以及发生中断时的处理。

如果使用Trap命令，则可通过事件发生跳跃到标签或调用函数。Trap命令包括2种类型。

- 将用户定义的输入状态设为触发的4个Trap
- 将系统状态设为触发的7个Trap

本项目说明系统状态触发的Trap。

格式

```
Trap {Emergency | Error | Pause | SGOpen | SGClose | Abort | Finish } Xqt 函数名

Trap {Emergency | Error | Pause | SGOpen | SGClose | Abort | Finish }
```

参数

Emergency
　　发生紧急停止时，执行指定的函数。
Error
　　发生错误时，执行指定的函数。
Pause
　　进入暂停状态时，执行指定的函数。
SGOpen
　　安全门电路处于开路状态时，执行指定的函数。
SGClose
　　安全门电路处于闭合状态时，执行指定的函数。
Abort
　　因用户或系统停止所有任务(后台任务除外)时(执行相当于Abort All的命令或按下中断按钮时)，执行指定的函数。
Finish
　　所有任务(后台任务除外)结束时，执行指定的函数。未在执行了Trap Abort的条件下执行。
函数名称
　　是系统状态成立时进行Xqt的中断处理任务的函数。不能指定带有自变量的函数。但是，如果参数中已指定“Error”，则可以指定3个带有自变量的函数。

注意

Epson RC+ 4.x为止的Trap *** Call功能在Epson RC+ 8.0中被替换为Trap *** Xqt。

说明

执行系统状态成立时指定的中断处理任务。

即使执行中断处理任务，其Trap设置也不会被清除。

要清除Trap设置时，省略函数名并执行Trap命令。

[例]“Trap Emergencyc”表示清除“Trap Emergencyc”。

如果一般任务全部结束并且控制器进入Ready状态，所有的Trap设置则会被清除。

不能通过中断处理任务再次利用Xqt执行任务。

⚠ 注意

- Forced标志

  通过在On、Off等I/O输出命令中指定Forced标志，即使在紧急停止期间、安全门打开时、示教模式期间或发生错误时，也可以进行I/O输出的ON/OFF操作。

  请绝对不要将致动器等伴随有机械动作的外部设备连接到指定Forced标志的I/O输出上。否则外部设备可能会在紧急停止期间、安全门打开时、示教模式期间或发生错误时进行动作，非常危险。

  假设将指定Forced标志的I/O输出连接到状态显示LED等不会产生机械动作的外部设备上。

- 指定Emergency时

  发生紧急停止时，按NoEmgAbort任务属性执行指定的函数。

可通过中断处理任务执行的命令为可执行NoEmgAbort任务的命令。

紧急停止的中断处理完成之后，请立即结束任务。如果任务未结束，控制器则无法进入Ready状态。不能通过中断处理任务执行Reset命令，自动解除紧急停止。

要通过中断处理任务执行打开或关闭I/O的任务时，请取消勾选[控制器设置]-[环境设置]-[]复选框。如果保持勾选状态，则不能保证执行通过控制器将I/O设为Off或通过任务将I/O设为On两者何者为先。

- 指定Error时

  发生错误时，按NoEmgAbort任务属性执行指定的函数。

可通过中断处理任务执行的命令为可执行NoEmgAbort任务的命令。

错误的中断处理完成之后，请立即结束任务。如果任务未结束，控制器则无法进入Ready状态。

可以在用户函数中指定三个可省略的参数(错误编号、机器人编号、关节编号)。如要使用这些参数，请在Trap函数中加入三个byval整数参数。

如果发生运动控制错误，将设置错误编号、机器人编号、关节编号。

如果发生运动控制以外的错误，将在机器人编号、关节编号中设置“0”。

- 指定Pause时

  进入暂停状态时，按NoPause任务属性执行指定的函数。

- 指定SGOpen时

  安全门电路处于开路状态时，按NoPause任务属性执行指定的函数。

- 指定SGClose时

  安全门电路处于闭合状态时，按NoPause任务属性执行指定的函数。

通过中断处理任务执行Cont命令时，会发生错误。

- 指定Abort时

  因用户或系统停止所有任务(后台任务除外)时(执行相当于Abort All的命令或按下中断按钮时)，按照NoPause任务属性执行指定的函数。

中断处理完成之后，请立即结束任务。如果任务未结束，控制器则无法进入Ready状态。

即使在由Trap Abort执行的任务中发生错误，也不执行Trap Error的处理任务。

任务因Shutdown命令或Restart命令而被中断时，也不执行Trap Abort或Trap Finish的处理任务。

- 指定Finish时

  所有任务(后台任务除外)结束时，按NoPause任务属性执行指定的函数。未在执行了Trap Abort处理任务的条件下执行。

结束的中断处理完成之后，请立即结束任务。如果任务未结束，控制器则无法进入Ready状态。

参阅

Era, Erl, Err, Ert, ErrMsg$, OnErr, Reset, Restart, SysErr, Xqt

Trap使用示例

```
Function main
    :
  Trap Error Xqt suberr
    :
Fend

Function suberr
  Print "Error =", Err
  On ErrorSwitch
Fend

Function main

  Trap Error Xqt trapError

FEnd


Function trapError(errNum As Integer, robotNum As Integer, jointNum As Integer)
Print "error number = ", errNum
Print "robot number = ", robotNum
Print "joint number = ", jointNum
If Ert = 0 Then
  Print "system error"
Else
  Print "task error"
  Print "function = ", Erf$(Ert)
  Print "line number = ", Erl(Ert)
EndIf
FEnd
```

3.23.36 Trim$函数

用于返回前后不含空格且与指定字符串相同的字符串。

格式

Trim$ (字符串)

参数

字符串
　　指定字符串表达式。

返回值

指定字符串前后含有空格时，删除空格。

参阅

LTrim$、RTrim$

Trim$函数使用示例

```
str$ = "  data  "
str$ = Trim$(str$)  ' str$ = "data"
```

3.23.37 TW函数

用于返回Wait命令、WaitNet命令、WaitSig命令的状态。

格式
TW

返回值
在指定时间内Wait状态成立时返回“False”。

发生超时时返回“True”。

说明
如果此前的Wait命令条件成立，则返回“False”；如果发生超时，则返回“True”。

参阅
TMOut、Wait、Hand_TW

TW函数使用示例

```
Wait Sw(0) = On, 5     '输入位0变为ON状态之前待机5秒钟
If TW = True Then
    Print "Time Up"    '5秒以上时显示“Time Up”
EndIf
```

3.24 U

3.24.1 UBound函数

用于返回指定数组中可设置下标的最大值。

格式

UBound (数组变量名 [, 维度])

参数

数组变量名

根据通常变量名的命名方法进行命名。

维度

以下述整数值设置要返回下标最大值的维度。可省略，如果省略，则假设为“1”。

- 1：第1个维度
- 2：第2个维度
- 3：第3个维度

参阅

Redim

UBound函数使用示例

```
Integer i, a(10)

For i = 0 to UBound(a)
    a(i) = i
Next
```

3.24.2 UByte

用于声明UByte型变量。(无符号整数型、大小：2字节)

格式

UByte 变量名 [(数组变量的最大下标)] [, 变量名 [(数组变量的最大下标)]...]

参数

变量名
　　指定声明为UByte型的变量名。
数组变量的最大下标
　　可利用数组变量的最大下标声明到最高三维。使用下述格式。可省略。
　　(最大下标1, [最大下标2], [最大下标3])
　　由于下标从0开始，因此数元素为最大下标加上1。在所有元素数不超过以下最大值的范围内指定各最大下标。

- 本地变量：2,000
- 备份变量(Global Preserve)： 4,000
- 全局变量和模块变量：100,000

说明

UByte在将变量声明为UByte型时使用。UByte变量的范围是0～255。在Function开头声明本地变量。在Function之外声明全局变量和模块变量。

参阅

Boolean、Byte、Double、Global、Int32、Int64、Integer、Long、Real、String、UInt32、UInt64、UShort

UByte使用示例

下例声明UByte型变量并为该变量赋值。

查看变量test_ok的最高位是1还是0。其结果将显示在显示器中。(在此例中，为变量赋值15，所以始终设置了变量test_ok值的高位。)

```
Function Test
    UByte A(10)          'UByte型的一维数组
    UByte B(10, 10)      'UByte型的二维数组
    UByte C(5, 5, 5)     'UByte型的三维数组
    UByte test_ok
    test_ok = 15
    Print "Initial Value of test_ok = ", test_ok
    test_ok = (test_ok And 8)
    If test_ok <> 8 Then
       Print "test_ok high bit is ON"
    Else
       Print "test_ok high bit is OFF"
    EndIf
Fend
```

3.24.3 UCase$函数

用于以大写字符返回小写字符串。

格式

UCase$ (文字列)

参数

字符串
　　指定要大写的字符串。

返回值

返回大写字符串。

参阅

LCase$、LTrim$、Trim$、RTrim$

UCase$函数使用示例

```
str$ = "Data"
str$ = UCase$(str$)  ' str$ = "DATA"
```

3.24.4 UInt32

用于声明UInt32型变量。(无符号4字节整数型变量)

格式

UInt32 变量名 [(数组变量的最大下标)] [, 变量名 [(数组变量的最大下标)]...]

参数

变量名
　　指定要进行变量声明的变量名。
数组变量的最大下标
　　可利用数组变量的最大下标声明到最高三维。使用下述格式。可省略。
　　(最大下标1, [最大下标2], [最大下标3])
　　由于下标从0开始，因此数元素为最大下标加上1。在所有元素数不超过以下最大值的范围内指定各最大下标。

- 本地变量：2,000
- 备份变量(Global Preserve)： 4,000
- 全局变量和模块变量：100,000

说明

UInt32用于声明整数型变量。整数型变量的范围为0～4294967295。在Function开头声明本地变量。在Function之外声明全局变量和模块变量。

参阅

Boolean、Byte、Double、Global、Int32、Int64、Integer、Long、Real、Short、String、UByte、UInt64、UShort

UInt32使用示例

如下所示为使用UInt32声明整数型变量的程序。

```
Function uint32test
    UInt32 A(10)          'UInt32型一维数组
    UInt32 B(10, 10)      'UInt32型二维数组
    UInt32 C(5, 5, 5)     'UInt32型三维数组
    UInt32 var1, arrayvar(10)
    Integer i
    Print "Please enter an Integer Number"
    Input var1
    Print "The Integer variable var1 = ", var1
    For i = 1 To 5
        Print "Please enter an Integer Number"
        Input arrayvar(i)
        Print "Value Entered was ", arrayvar(i)
    Next i
Fend
```

3.24.5 Uint64

用于声明UInt64型变量。(无符号8字节整数型变量)

格式

Uint64 变量名 [(数组变量的最大下标)] [, 变量名 [(数组变量的最大下标)]...]

参数

变量名
　　指定要进行变量声明的变量名。
数组变量的最大下标
　　可利用数组变量的最大下标声明到最高三维。使用下述格式。可省略。
　　(最大下标1, [最大下标2], [最大下标3])
　　由于下标从0开始，因此数元素为最大下标加上1。在所有元素数不超过以下最大值的范围内指定各最大下标。

- 本地变量：2,000
- 备份变量(Global Preserve)：4,000
- 全局变量和模块变量：100,000

说明

UInt64用于声明整数型变量。整数型变量的范围为0～18446744073709551615。在Function开头声明本地变量。在Function之外声明全局变量和模块变量。

参阅

Boolean, Byte, Double, Global, Int32, Int64, Integer, Long, Real, Short, String, UByte, UInt32, UShort

UInt64使用示例

如下所示为使用UInt64声明整数型变量的程序的示例。

```
Function uint64test
    UInt64 A(10)          'UInt64型一维数组
    UInt64 B(10, 10)      'UInt64型二维数组
    UInt64 C(5, 5, 5)     'UInt64型三维数组
    UInt64 var1, arrayvar(10)
    Integer i
    Print "Please enter an Integer Number"
    Input var1
    Print "The Integer variable var1 = ", var1
    For i = 1 To 5
        Print "Please enter an Integer Number"
        Input arrayvar(i)
        Print "Value Entered was ", arrayvar(i)
    Next i
Fend
```

3.24.6 UOpen

用于在读出和写入两种模式下打开文件。

格式

UOpen 文件名As #文件编号 . . Close #文件编号

参数

文件名
　　指定包括路径的文件名字符串。仅指定文件名时，是指当前目录中的文件。详情请参阅ChDisk。
文件编号
　　以30～63之间的整数值或表达式进行指定。

说明

以指定的文件编号打开指定的文件。该语句用于将数据写入到指定的文件中或读出数据。

注意

可使用网络路径。

如果指定不存在的文件，则会生成该文件并写入数据。如果指定存在的文件，则从现有数据的开头读写数据。

利用Seek命令切换文件的读入/写入位置(指针)。切换读入访问和写入访问时，请利用Seek命令重新设置文件指针。

指定的文件编号用于在打开文件期间识别该文件。因此，在关闭该文件之前，不能使用与其它文件相同的文件编号。按文件操作命令(Print#、Input#、Read、Write、Seek、Eof、Flush、Close)使用文件编号。

利用Close语句关闭文件并释放文件编号。

请利用FreeFile函数获取文件编号，以免在多个任务中使用同一编号。

参阅

Close, Print #, Input#, AOpen, BOpen, ROpen, WOpen, FreeFile, Seek

UOpen使用示例

```
Integer fileNum, i, j

fileNum = FreeFile
UOpen "TEST.DAT" As #fileNum
For i = 0 To 100
   Print #fileNum, i
Next i
Close #fileNum

fileNum = FreeFile
UOpen "TEST.DAT" As #fileNum
Seek #fileNum, 10
Input #fileNum, j
Print "data = ", j
Close #fileNum
```

3.24.7 UpdateDB

更新已打开数据库内检索的表格中的数据。

格式

UpdateDB #数据库编号, 项目, 值

参数

数据库编号
　　指定利用OpenDB指定的数据库编号(501～508的整数值)。
项目
　　指定要更新的表格的项目名。
值
　　指定要更新的值。

说明

以指定项目值更新已打开数据库的Select的表格中的数据。

在更新数据之前，必须执行SelectDB并选择要更新的记录。

注意

- 需要连接已安装RC+的PC。

参阅

OpenDB, CloseDB, SelectDB, DeleteDB

UpdateDB使用示例

SQL数据库的使用示例 如下所示为在SQL 服务器2000的样本数据库Northwind的表格Employees中登记数据，并更新已登记数据的项目的简单示例。

```
  Integer count, i, eid
  String Lastname$, Firstname$, Title$

  OpenDB #501, SQL, "(LOCAL)", "Northwind"
  count = SelectDB(#501, "Employees", "TitleOfCourtesy = 'Mr.'")
  Print #501, "Epson", "Taro", "Engineer", "Mr."
  count = SelectDB(#501, "Employees", "LastName = 'Epson' and    FirstName =
'Taro'")
  Input #501, eid, Lastname$, Firstname$, Title$
  Print eid, ",", Lastname$, ",", Firstname$, ",", Title$
  UpdateDB #501, "Title", "Chief Engineer"
  count = SelectDB(#501, "Employees", "LastName = 'Epson' and    FirstName =
'Taro'")
  Input #501, eid, Lastname$, Firstname$, Title$
  Print eid, ",", Lastname$, ",", Firstname$, ",", Title$
  CloseDB #501
```

3.24.8 UShort

用于声明UShort型变量。(无符号2字节整数型变量)

格式

UShort 变量名 [(数组变量的最大下标)] [, 变量名 [(数组变量的最大下标)]...]

参数

变量名
　　指定要进行变量声明的变量名。
数组变量的最大下标
　　可利用数组变量的最大下标声明到最高三维。使用下述格式。可省略。
　　(最大下标1, [最大下标2], [最大下标3])
　　由于下标从0开始，因此数元素为最大下标加上1。在所有元素数不超过以下最大值的范围内指定各最大下标。

- 本地变量：2,000
- 备份变量(Global Preserve)：4,000
- 全局变量和模块变量：100,000

说明

UShort用于声明整数型变量。整数型变量的范围为0～65535。在Function开头声明本地变量。在Function之外声明全局变量和模块变量。

参阅

Boolean、Byte、Double、Global、Int32、Int64、Integer、Long、Real、Short、String、UByte、UInt32、UInt64

UShort使用示例

如下所示为使用UShort声明整数型变量的程序。

```
Function ushorttest
    UShort A(10)          'UShort型的一维数组
    UShort B(10, 10)      'UShort型的二维数组
    UShort C(5, 5, 5)     'UShort型的三维数组
    UShort var1, arrayvar(10)
    Integer i
    Print "Please enter an Integer Number"
    Input var1
    Print "The Integer variable var1 = ", var1
    For i = 1 To 5
        Print "Please enter an Integer Number"
        Input arrayvar(i)
        Print "Value Entered was ", arrayvar(i)
    Next i
Fend
```

3.25 V

3.25.1 Val函数

用于将由数字组成的指定字符串转换为数值并返回该值。

格式
Val (字符串)

参数

字符串
　　指定仅由数字组成的字符串表达式。字符串也包括前缀
　　&H(16进制)、&O(8进制)、&B(2进制)

返回值
返回由整数或输入的字符串表达式组成的浮点数。带有小数点的输入字符串表达式被转换为浮点数。除此之外时，返回值返回的是整数值。

说明
Val用于将由数字组成的字符串表达式转换为数值。结果为整数或浮点数。如果向Val命令赋予带有小数点的输入字符串表达式，则返回浮点数；如果不是，则返回整数值。

参阅
Abs、Asc、Chr$、Int、Left$、Len、Mid$、Mod、Right$、Sgn、Space$、Str$

Val函数使用示例
如下所示为将几个不同的字符串表达式转换为数值，并在画面中显示其结果的程序示例。

```
Function ValDemo
    String realstr$, intstr$
    Real realsqr, realvar
    Integer intsqr, intvar

    realstr$ = "2.5"
    realvar = Val(realstr$)
    realsqr = realvar * realvar
    Print "The value of ", realstr$, " squared is: ", realsqr

    intstr$ = "25"
    intvar = Val(intstr$)
    intsqr = intvar * intvar
    Print "The value of ", intstr$, " squared is: ", intsqr
Fend
```

如下所示为利用命令窗口的操作示例。

```
> Print Val("25.999")
25.999
>
```

3.25.2 VSD

用于设置SCARA机器人的变速CP动作功能。

格式
VSD { ON | Off }

参数

On | Off

- On：将SCARA机器人的变速CP动作功能设为有效。
- Off：将SCARA机器人的变速CP动作功能设为无效。

说明
VSD通过下述命令启用。
Move、Arc、Arc3

本命令仅对SCARA机器人有效。对于SCARA机器人以外的机型，请使用AvoidSinularity SING_VSD。

变速CP动作功能在SCARA机器人执行CP动作过程中防止加速度错误和超速错误的发生。在保持动作轨迹的状态下自动限制关节速度以执行动作的功能。关节速度受限时，不保持通过SpeedS设置的工具中心点速度，但关节速度低于限制范围时，将恢复为原来的工具中心点速度。如要优先等速，请将AccelS、DecelS、SpeedS调小，避免错误发生。

如果使用VSD，仍发生加速度错误和超速错误，请将AccelS、DecelS、SpeedS调小。

如果变更了VSD的设置值，将保持有效至下一次控制器启动时。

控制器启动时，VSD将切换为OFF状态。

参阅
VSD函数

VSD使用示例

```
VSD On '将变速CP动作设为有效以使其执行动作
Move P1
Move P2
VSD Off
```

3.25.3 VSD函数

返回SCARA机器人的变速CP动作功能的设置值。

格式

VSD

返回值

- On = 变速CP动作功能有效
- Off = 变速CP动作功能无效

参阅

VSD

VSD函数使用示例

```
If VSD = Off Then
    Print "Variable Speed Drive is off"
EndIf
```

3.25.4 VxCalib

请在Vision Guide以外的客户准备的图像系统中使用该命令。

用于生成客户准备的图像系统的校准数据。

格式

(1) VxCalib CalNo

(2) VxCalib CalNo, CamOrient, P(pixel_st : pixel_ed), P(robot_st : robot_ed) [, TwoRefPoints]

(3) VxCalib CalNo, CamOrient, P(pixel_st : pixel_ed), P(robot_st : robot_ed),P(ref0) [, P(ref180)]

参数

CalNo
以整数值指定校准数据的编号。可利用0～15的整数定义最多16个编号。

CamOrient
以下述整数值指定摄像机的安装方向。仅可利用格式(2)指定1～3。仅可利用格式(3)指定4～7。

- 1：固定摄像机
- 2：向下固定摄像机
- 3：向上固定摄像机
- 4：移动摄像机第2轴安装
- 5：移动摄像机第4轴安装
- 6：移动摄像机第5轴安装
- 7：移动摄像机第6轴安装

P(pixel_st : pixel_ed)
以连续点数据指定像素坐标(仅X、Y)。

P(robot_st : robot_ed)
以连续点数据指定机器人坐标。根据在CamOrient中指定的摄像机安装方向的不同，指定的点数有所差异。CamOrient = 1～3时，请以当前的TOOL和ARM设置机器人坐标。CamOrient = 4～7时，请以TOOL：0、ARM：0的方式设置机器人坐标。

TwoRefPoints
格式(1)时可指定。如果使用2个测量点，则指定“True”；如果使用1个测量点，则指定“False”。通过指定2个测量点，可实施更准确的校准。默认设置为“False”。可省略。

P(ref0)
格式(3)时可指定。以点数据指定基准点的机器人坐标。

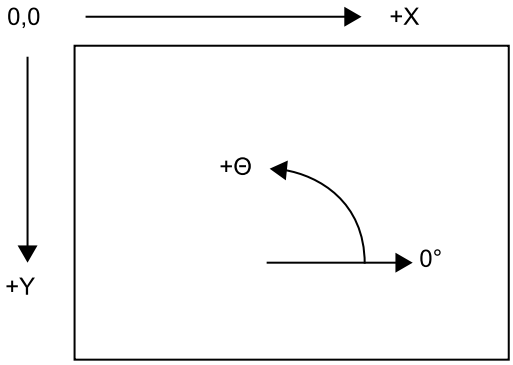
P(ref180)
格式(3)时可指定。以点数据指定第2个基准点的机器人坐标。通过指定2个基准点，可实施更准确的校准。可省略。

说明

根据使用指定摄像机姿势的校准编号、由自变量赋予的像素坐标、机器人坐标、基准点(仅限于移动摄像机)运算校准数据。

仅指定CalNo时，显示定义时的点数据等(仅命令窗口)。

下表所示为像素坐标的坐标系。单位为像素。



关于像素坐标和机器人坐标，请将画面上的左上位置设为点1，然后按下表顺序将右下位置设为点9。根据CamOrient、TwoRefPoints自变量划分为4种类型。

1) CamOrient = 1～3(固定、向下固定、向上固定)、TwoRefPoints = False时

数据顺序	画面位置	像素坐标	机器人坐标
1	左上	检测坐标1	测量点坐标1
2	中上	检测坐标2	测量点坐标2
3	右上	检测坐标3	测量点坐标3
4	右中	检测坐标4	测量点坐标4
5	中中	检测坐标5	测量点坐标5
6	左中	检测坐标6	测量点坐标6
7	左下	检测坐标7	测量点坐标7
8	中下	检测坐标8	测量点坐标8
9	右下	检测坐标9	测量点坐标9

2) CamOrient = 2(向下固定)、TwoRefPoints = True时

已正确定义工具时，不需要TwoRefPoints。请设为“False”。

如果将TwoRefPoints设为“True”，则可使用2个测量点，因此可实施更准确的校准。

仅机器人坐标需要18个点。

设置1～9的测量点坐标之后，请将U轴转动180度，然后设置将夹具末端(杆等)对准校准目标位置的测量点坐标10～18。

数据顺序	画面位置	像素坐标	机器人坐标
1	左上	检测坐标1	测量点坐标1
2	中上	检测坐标2	测量点坐标2
3	右上	检测坐标3	测量点坐标3
4	右中	检测坐标4	测量点坐标4
5	中中	检测坐标5	测量点坐标5
6	左中	检测坐标6	测量点坐标6
7	左下	检测坐标7	测量点坐标7
8	中下	检测坐标8	测量点坐标8

数据顺序	画面位置	像素坐标	机器人坐标
9	右下	检测坐标9	测量点坐标9
10	左上	ーーー	测量点坐标10
11	中上	ーーー	测量点坐标11
12	右上	ーーー	测量点坐标12
13	右中	ーーー	测量点坐标13
14	中中	ーーー	测量点坐标14
15	左中	ーーー	测量点坐标15
16	左下	ーーー	测量点坐标16
17	中下	ーーー	测量点坐标17
18	右下	ーーー	测量点坐标18

3) CamOrient = 3(向上固定)、TwoRefPoints = True时

已正确定义工具时，不需要TwoRefPoints。请设为“False”。

如果将TwoRefPoints设为“True”，则可使用2个检测点，因此可实施更准确的校准。

仅像素坐标需要18个点。

在各测量点坐标设置1～9检测坐标之后，请设置将U轴转动180度的位置的检测坐标10～18。

数据顺序	画面位置	像素坐标	机器人坐标
1	左上	检测坐标1	测量点坐标1
2	中上	检测坐标2	测量点坐标2
3	右上	检测坐标3	测量点坐标3
4	右中	检测坐标4	测量点坐标4
5	中中	检测坐标5	测量点坐标5
6	左中	检测坐标6	测量点坐标6
7	左下	检测坐标7	测量点坐标7
8	中下	检测坐标8	测量点坐标8
9	右下	检测坐标9	测量点坐标9
10	左上	检测坐标10	ーーー
11	中上	检测坐标11	ーーー
12	右上	检测坐标12	ーーー
13	右中	检测坐标13	ーーー
14	中中	检测坐标14	ーーー
15	左中	检测坐标15	ーーー
16	左下	检测坐标16	ーーー
17	中下	检测坐标17	ーーー

数据顺序	画面位置	像素坐标	机器人坐标
18	右下	检测坐标18	ーーー

4) CamOrient = 4～7时

数据顺序	画面位置	像素坐标	机器人坐标
1	左上	检测坐标1	测量点坐标1
2	中上	检测坐标2	测量点坐标2
3	右上	检测坐标3	测量点坐标3
4	右中	检测坐标4	测量点坐标4
5	中中	检测坐标5	测量点坐标5
6	左中	检测坐标6	测量点坐标6
7	左下	检测坐标7	测量点坐标7
8	中下	检测坐标8	测量点坐标8
9	右下	检测坐标9	测量点坐标9

注意

- 除上表之外，请指定基准点的机器人坐标。

  如果使用2个基准点，则可实施更准确的校准。此时，需要2个U轴相差180度的点。

  设置第1个基准点坐标之后，请将U轴转动180度，然后设置将夹具末端(杆等)对准校准目标位置的第2个基准点坐标。已正确定义工具时，不需要使用2个基准点。

参阅

VxTrans函数、VxCalInfo函数、VxCalDelete、VxCalSave、VxCalLoad

VxCalib使用示例

```
Function MobileJ2

  Integer i
  Double d(8)

  Robot 1
  LoadPoints "MobileJ2.pts"

  VxCalib 0, 4, P(21:29), P(1:9), P(0)

  If (VxCalInfo(0, 1) = True) Then
    For i = 0 To 7
      d(i) = VxCalInfo(0, i + 2)
    Next i
    Print "Calibration result:"
    Print d(0), d(1), d(2), d(3), d(4), d(5), d(6), d(7)

    P52 = VxTrans(0, P51, P50)
    Print "Coordinates conversion result:"
    Print P52
    SavePoints "MobileJ2.pts"
    VxCalSave "MobileJ2.caa"
  Else
```

```
    Print "Calibration failed"
  EndIf

Fend
```

3.25.5 VxCalDelete

请在Vision Guide以外的客户准备的图像系统中使用该命令。

用于删除客户准备的图像系统的校准数据。

格式

VxCalDelete CalNo

参数

CalNo
以整数值指定校准数据的编号。可利用0～15的整数定义最多16个编号。

说明

用于删除由指定校准编号定义的校准数据。

参阅

VxCalib、VxTrans函数、VxCalInfo函数、VxCalSave、VxCalLoad

VxCalDelete使用示例

```
VxCalDelete "MobileJ2.caa"
```

3.25.6 VxCalLoad

请在Vision Guide以外的客户准备的图像系统中使用该命令。

用于从文件读入客户准备的图像系统的校准数据。

格式

VxCalLoad FileName

参数

FileName
以字符串表达式指定读入校准数据的文件名。扩展名固定为“.caa”。省略时，添加“.caa”。不是“.caa”时，转换为“.caa”。不能指定路径。

说明

用于从当前项目内的指定文件读入校准数据。

参阅

VxCalib、VxTrans函数、VxCalInfo函数、VxCalDelete、VxCalSave

VxCalLoad使用示例

```
VxCalLoad "MobileJ2.caa"
```

3.25.7 VxCalInfo函数

请在Vision Guide以外的客户准备的图像系统中使用该命令。

用于返回客户准备的图像系统的校准结束状态和校准数据。

格式

VxCalInfo (CalNo, CalData)

参数

CalNo

以整数值指定校准数据的编号。可利用0～15的整数定义最多16个编号。

CalData

以下表所示的整数值指定要获取的校准数据类型。

CalData	校准数据类型
1	校准的结束状态
2	X方向的平均偏差 [mm]
3	X方向的最大偏差 [mm]
4	X方向单位像素的长度 [mm]
5	X方向的倾斜度
6	Y方向的平均偏差 [mm]
7	Y方向的最大偏差 [mm]
8	Y方向单位像素的长度 [mm]
9	Y方向的倾斜度

返回值

CalData = 1时，以Bool型数值返回指定的校准数据；CalData = 2～9时，以Double型数值返回指定的校准数据。

说明

可确认校准数据是由哪个校准编号定义的。

也可以获取校准数据的值。

参阅

VxCalib、VxTrans函数、VxCalDelete、VxCalSave、VxCalLoad

VxCalInfo函数使用示例

```
Print VxCalInfo(0, 1)
```

3.25.8 VxCalSave

请在Vision Guide以外的客户准备的图像系统中使用该命令。

用于将客户准备的图像系统的校准数据保存到文件中。

格式

VxCalSave FileName

参数

FileName
以字符串表达式指定保存校准数据的文件名。扩展名固定为“.caa”。省略时，自动添加“.caa”。不是“.caa”时，自动转换为“.caa”。不能指定路径。

说明

用于以指定的文件名保存校准数据。文件被保存到当前项目内。存在同名文件时，覆盖校准数据。

参阅

VxCalib、VxTrans函数、VxCalInfo函数、VxCalDelete、VxCalLoad

VxCalSave使用示例

```
VxCalSave "MobileJ2.caa"
```

3.25.9 VxTrans函数

请在Vision Guide以外的客户准备的图像系统中使用该命令。

用于进行从像素坐标到机器人坐标的坐标转换，以及返回已转换点数据。

格式

VxTrans (CalNo, P(pixel) [, P (camRobot)]) As Pose

参数

CalNo
以整数值指定校准数据的编号。可利用0～15的整数定义最多16个编号。
P(pixel)
以点数据指定图像像素坐标(仅X、Y、U)。
P(camRobot)
可省略。为移动摄像机时，指定拍摄时的机器人位置。省略时，使用机器人的当前位置。请以TOOL：0、ARM：0的方式设置点数据。

返回值

以点数据返回计算出来的机器人坐标。

说明

使用指定校准编号的校准数据，进行从像素坐标到机器人坐标的坐标转换。

在移动摄像机中指定当前值以外数值时，指定拍摄时的机器人坐标P(camRobot)。请以TOOL：0、ARM：0的方式设置P(camRobot)。已设置机器人坐标的第4/第6关节角度用于计算。

参阅

VxCalib、VxCalInfo函数、VxCalDelete、VxCalSave、VxCalLoad

VxTrans函数使用示例

```
P52 = VxTrans(0, P51, P50)
```

3.26 W

3.26.1 Wait

使用MemSw或Sw，在指定的条件成立之前或指定时间内，使程序处于等待状态。(也可以替代Sw，使用Oport检查I/O输出。)

也可以等待全局变量的值发生变化。

格式

(1) Wait 时间

(2) Wait 条件表达式

(3) Wait 条件表达式, 时间

参数

时间

以0～2147483的实数(单位：秒)指定待机时间。最小有效位为0.01秒。

事件条件表达式

以下述格式指定条件。

[条件] 比较运算符(=、<>、>=、>、<、<=)[整数表达式]

可在条件中使用下述函数或变量。

- 函数：Sw, In, InW, Oport, Out, OutW, MemSw, MemIn, MemInW, Ctr,GetRobotInsideBox, GetRobotInsidePlane, MCalComplete,Motor, LOF, ErrorOn, SafetyOn, EstopOn, TeachOn , Cnv_QueLen, WindowsStatus, AtHome, LatchState, WorkQue_Len, PauseOn, AIO_In, AIO_InW, AIO_Out, AIO_OutW, Hand_On, Hand_Off, SF_GetStatus
- 变量：Byte, Int32, Integer, Long, Short, UByte, UInt32, UShort型备份变量、全局变量、模块变量

另外，可利用下述运算符对多个条件表达式附加掩码或进行复合组合。

- 运算符：And, Or, Xor, Mask

说明

(1) 仅指定时间时

作为计时器使用Wait命令时，仅使程序暂停指定的时间长度，然后继续执行程序。

(2) 仅指定条件表达式时

作为条件联锁功能使用Wait时，在指定的条件成立之前，使程序处于待机状态。如果通过TMout命令指定了超时，即使经过指定时间，如果条件式未成立，也会发生错误。可使用And、Mask、Or或Xor命令等对1个Wait命令进行多条件检查。请参阅使用示例。

(3) 指定时间和条件表达式时

指定条件表达式和时间时，如果条件成立或经过指定时间，则执行后续命令。可使用Tw函数确认条件表达式是否成立，或是否经过指定时间。

注意

- 在Wait中并用超时时

  Wait命令中未指定待机时间时，如果指定超时，则可设置等待指定状态的限制时间。可使用TMOut命令指定超时。详情请参阅该TMOut命令的项目。(TMOut命令的默认值为“0”，表示没有时间限制。)

- 利用Wait等待变量时
  - 可用于变量等待的变量类型为整数型(Byte、Int32、Integer、Long、Short、UByte、UInt32、UShort)。
  - 不能使用数组变量。
  - 不能使用本地变量。
  - 在超过0.01秒的时间内变量值未满足条件时，系统可能不能检测到变量变化。
  - 系统内可使用的变量等待数存在限制。1个系统内可使用的变量等待数量最多为64个(也包括在Till等条件表达式中使用的变量等待)。如果超过最大数，则会在项目创建时发生错误。
  - 如果利用Byref引用执行变量等待的变量，则会发生错误。
  - 条件式右边的整数表达式包括变量或函数时，在设置Wait条件时运算其值。由于可能会形成不希望有的条件，因此不建议在整数表达式中使用变量或函数。
- 使用PC COM端口(1001～1008)时

  能在Wait命令中使用Lof函数。
- 在执行Wait期间程序暂停时

  在执行Wait命令期间，程序暂停时(Pause状态)Wait命令也并不会停止。当满足条件表达式或经过了指定的时间后，Wait命令会结束。

  如果在Wait命令中设置了时间，在指定的时间过去之前，通过连续执行来恢复程序时，则会重置之前经过的时间，而程序会等待指定的时间。

参阅

AtHome, Cnv_QueLen, Ctr, ErrorOn, EstopOn, GetRobotInsideBox, GetRobotInsidePlane, In, InW, LatchState, LOF, Mask, MCalComplete, MemIn, , MemInW, MemSw Motor, Oport, Out, OutW, PauseOn, SafetyOn, Sw, TeachOn, TMOut, WindowsStatus, Tw, WorkQue_Len, SF_GetStatus

Wait使用示例

在下例当中，对于可分别启动动作命令的2个任务，除非一方进行机器人控制，否则可进行机器人控制的联锁功能则会启动。这样可按顺序进行各任务指定的动作。并用MemSw的Wait命令用于在存储器I/O位1变为适当值之前，使程序处于待机状态，在达到安全状态之后重新开始动作。

```
Function main
    Integer I
    MemOff 1
    Xqt !2, task2
    For i = 1 to 100
       Wait MemSw(1) = Off
       Go P(i)
       MemOn 1
    Next I
Fend

Function task2
    Integer i
    For i = 101 to 200
       Wait MemSw(1) = On
       Go P(i)
       MemOff 1
    Next i
Fend

'等待输入0变为ON状态
Wait Sw(0) = On

'在等待60.5秒钟之后继续执行
```

```
Wait 60.5

'等待输入0变为OFF、输入1变为ON状态
Wait Sw(0) = Off And Sw(1) = On

'等待存储位0变为ON或存储位1变为ON状态
Wait MemSw(0) = On Or MemSw(1) = On

'等待1秒钟，然后将输出1设为ON
Wait 1; On 1

'在输入端口0的低3位变为1之前进行待机
Wait In(0) Mask 7 = 1

'等待全局Integer型变量giCounter的值超过10
Wait giCounter > 10

'在全局Long型变量glCheck的值到达30000之前待机10秒钟
Wait glCheck = 30000, 10
```

3.26.2 WaitNet

用于等待TCP/IP端口建立连接。

格式

WaitNet #端口编号 [, 超时时间]

参数

端口编号
　　以201～216的整数值指定等待连接的TCP/IP的端口编号。
超时时间
　　指定最长等待连接时间。可省略。

参阅

OpenNet、CloseNet

WaitNet使用示例

如下所示为2个控制器的TCP/IP设置示例。

```
Controller #1:
Port: #201
Host Name: 192.168.0.2
TCP/IP Port: 1000

Function tcpip
  OpenNet #201 As Server
  WaitNet #201
  Print #201, "Data from host 1"
Fend

Controller #2:
Port: #201
Host Name: 192.168.0.1
TCP/IP Port: 1000

Function tcpip
  String data$
  OpenNet #201 As Client
  WaitNet #201
  Input #201, data$
  Print "received '", data$, "' from host 1"
Fend
```

3.26.3 WaitPos

用于执行即使在路径运动有效的状态下，也要在执行下一语句之前，等待机器人进行减速停止。

格式

WaitPos

说明

通常，路径运动处于有效状态(指定CP On或CP参数)时，动作命令会在开始减速的同时将控制移交给下一语句。

但如果在其后插入WaitPos命令，则会在完成减速动作之后切换到后续动作。

参阅

Wait、WaitSig、CP

WaitPos使用示例

```
Off 1
CP On
Move P1
Move P2
WaitPos  '等待机器人减速
On 1
CP Off
```

3.26.4 WaitSig

用于等待其它任务的Signal命令发出的同步信号。

格式

WaitSig 信号编号 [, 超时时间]

参数

信号编号
　　以整数值(0～63)指定要接收的信号编号。
超时时间
　　以实值指定最长等待时间。可省略。

说明

用于等待来自其它任务的信号。执行WaitSig之后进入信号等待状态，并无视此前的信号。

参阅

Wait、WaitPos、Signal

WaitSig使用示例

```
Function Main
  Xqt SubTask
  Wait 1
  Signal 1
  .
  .
Fend

Function SubTask
  WaitSig 1
  Print "signal received"
  .
Fend
```

3.26.5 Weight

用于设置和显示补偿PTP动作时的速度和加减速度的参数。

格式

Weight [末端夹具重量 [, {机械臂长度 | S | T }]]

Weight

参数

末端夹具重量
　　指定施加到机械臂上的末端夹具重量。(单位：Kg 小数点以下2位)可以省略，但是不能只省略[末端夹具重量]。
机械臂长
　　仅水平多关节机器人(包括RS系列)有效。指定第2关节中心～第3关节中心之间的距离。(单位：mm)可省略。
S
　　指定对附加轴S关节施加的负载重量。(单位：kg 小数点以下2位)
T
　　指定对附加轴T关节施加的负载重量。(单位：kg 小数点以下2位)

结果

如果省略参数，则显示当前的Weight设置值。

如果省略[机械臂长度]，侧输入的[末端夹具重量]会被设置，并设置[机械臂长度]的默认值。

所以不能只省略[末端夹具重量]。

说明

指定用于计算PTP动作的最大加减速度的参数。Weight命令用于设置夹具末端和工件的重量。仅限于水平多关节型机器人(包括RS系列)需要指定机械臂长度。机械臂长度为第2关节中心～第3关节中心之间的距离。水平多关节型机器(包括RS系列)以外机型时无效。机器人中设有附加轴时，需要使用S、T参数单独设置附加轴自带的负载重量。

如果根据设置的值计算的等效重量超出最大可搬运重量，则会发生错误。

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

还可以通过“负载、惯性、离心率/偏移量测量实用程序”进行设置。

详细信息请参阅以下手册。
《Epson RC+ 用户指南 - 负载、惯性、离心率/偏移量测量实用程序》

常见错误

- 超出最大容许负载重量时

  如果根据设置的值计算的等效重量超出最大容许负载重量，则会发生错误。

- 对机械臂的损伤

  如果将Weight的夹具末端重量设为明显低于实际重量的小的值，则会设置过大的加速值和减速值，可能会导致机械手损伤，敬请注意。

注意

- 即使关闭电源Weight的设置也不会更改

  Weight的值一旦设置就会存储在控制器中。即使关闭电源也不会改变。

如果未设置任何内容，则将使用先前设置的值。

参阅

Accel、Inertia

有关夹具的详细信息，请参阅以下手册。
《Hand功能》

Weight使用示例

如下所示为通过命令窗口利用Weight命令显示当前设置值的示例。

```
> weight
2.000, 200.000
>
```

如下所示为利用Weight命令设置夹具末端重量(3 kg)的示例。

```
Weight 3.0
```

如下所示为利用Weight命令设置附加轴S负载重量(30 kg)的示例。

```
Weight 30.0, S
```

3.26.6 Weight函数

用于返回由Weight命令设置的夹具末端重量和机械臂长度。

格式
Weight (参数编号)

参数

参数编号
以下述整数值设置参数编号。
- 1：末端夹具重量
- 2：机械臂长
- 3：附加轴S关节负载重量
- 4：附加轴T关节负载重量

返回值
以实值返回参数。

参阅
Inertia、Weight

有关夹具的详细信息，请参阅以下手册。
《Hand功能》

Weight函数使用示例

```
Print "The current Weight parameters are: ", Weight(1)
```

3.26.7 Where

用于显示机器人的当前位置数据。

格式
Where [本地编号]

参数

本地编号
　　指定本地坐标系编号。默认值为“Local 0”。可省略。

参阅
Joint、PList、Pulse

Where使用示例
显示的格式因机器人类型或附加轴有无而异。

下例所示为SCARA型机器人不带附加轴的情况。

```
>where
WORLD: X: 350.000 mm Y: 0.000 mm  Z: 0.000 mm  U: 0.000 deg V: 0.000 deg W: 0.000
deg
JOINT: 1:  0.000 deg 2: 0.000 deg 3: 0.000 mm  4: 0.000 deg
PULSE: 1:     0 pls  2:     0 pls 3:    0 pls  4:     0 pls

> local 1, 100,100,0,0

> where 1
WORLD: X: 250.000 mm Y:-100.000 mm  Z: 0.000 mm U: 0.000 deg V: 0.000 deg W: 0.000
deg
JOINT: 1:  0.000 deg 2:   0.000 deg 3: 0.000 mm 4: 0.000 deg
PULSE: 1:     0 pls  2:       0 pls 3:    0 pls 4:     0 pls
```

3.26.8 WindowsStatus函数

用于返回Windows的启动状态。

格式

WindowsStatus

返回值

以整数值返回当前的Windows启动状态。以位图像返回Windows启动状态，并表示下述状态。

功能名	系统预约	可发挥RC+功能	可发挥PC功能
位编号	15～2	1	0
可利用功能的详细说明		Vision Guide(取帧器型) RC+ API 现场总线主站	PC文件 PC RS-232C 数据库访问 DLL调用

注意

- 支持的控制器型号

  不支持T/VT系列。

说明

本函数用于在将控制器设置设为“独立模式”时确认控制器的启动状态。控制器设置被设为“联合模式”时，在可使用RC+功能和PC功能双方之前，不能开始执行程序。

WindowsStatus函数使用示例

```
Print "The current PC Booting up Status is: ", WindowsStatus
```

3.26.9 WOpen

用于在写入模式下打开文件。

格式

WOpen 文件名 As #文件编号
.
.
Close #文件编号

参数

文件名
　　指定包括路径的文件名字符串。仅指定文件名时，是指当前目录中的文件。详情请参阅ChDisk。
文件编号
　　以30～63之间的整数值或表达式进行指定。

说明

以指定的文件编号打开指定的文件。该语句用于打开指定文件并写入数据。(要在现有的文件中添写数据时，请参阅有关AOpen的说明。)

指定的文件不存在时，新建文件并写入到其中。如果存在指定的文件，则删除全部现有数据并重新写入数据。

指定的文件编号用于在打开文件期间识别该文件。因此，在关闭该文件之前，不能使用与其它文件相同的文件编号。按文件操作命令(Print#、Write、Seek、Flush、Close)使用文件编号。

利用Close语句关闭文件并释放文件编号。

请利用FreeFile函数获取文件编号，以免在多个任务中使用同一编号。

注意

- 可使用网络路径。
- 向文件写入时进行缓冲。

  可利用Flush语句写入被缓冲的数据。利用Close语句关闭文件时也进行写入。

参阅

AOpen, BOpen, Close, Print#, ROpen, UOpen, FreeFile

WOpen使用示例

```
Integer fileNum, i, j

fileNum = FreeFile
WOpen "TEST.DAT" As #fileNum
For i = 0 To 100
    Print #fileNum, i
Next i
Close #fileNum

fileNum = FreeFile
ROpen "TEST.DAT" As #fileNum
For i = 0 to 100
    Input #fileNum, j
    Print "data = ", j
Next i
Close #fileNum
```

3.26.10 WorkQue_Add

将工作队列数据(点数据和用户数据)追加到指定工作队列中。

格式

WorkQue_Add 工作队列编号, 点数据 [, 用户数据]

参数

工作队列编号
　　以整数值(1～16)指定工作队列编号。
点数据
　　指定要在工作队列中添加的点数据。
用户数据
　　以实数值指定与点数据一起注册的用户数据。可省略。省略后，0(实数)将注册为用户数据。

说明

点数据和用户数据将添加到工作队列的末尾。

但是，如果已通过WorkQue_Sort设置了Sort方法，则按照设置的Sort方法注册。

如果已通过WorkQue_Reject设置了防止重复注册的距离，则将计算与已注册的点数据间的距离，当点数据小于该距离，将不会在工作队列中添加点数据和用户数据。这种情况下，不会出现错误。

工作队列数据的上限值是“1000”。在用完工作队列数据时，通过WorkQue_Remove删除工作队列数据。

参阅

WorkQue_AutoRemove、WorkQue_Len、WorkQue_Reject、WorkQue_Remove、WorkQue_Sort

WorkQueAdd使用示例

```
Integer x, y
Real u

P0 = XY(300, 300, 300, 90, 0, 180)
P1 = XY(200, 280, 150, 90, 0, 180)
P2 = XY(200, 330, 150, 90, 0, 180)
P3 = XY(-200, 280, 150, 90, 0, 180)

Pallet 1, P1, P2, P3, 10, 10
x = 1
y = 1
u = 5.3
WorkQue_Add 1, Pallet(1, x, y), u
```

3.26.11 WorkQue_AutoRemove

对指定的工作队列设置自动删除功能。

格式

WorkQue_AutoRemove 工作队列编号 ,{True | False}

参数

工作队列编号
　　以整数值(1～16)指定工作队列编号。
True | False
　　True：将自动删除功能设为有效。False：将自动删除功能设为无效。

说明

对工作队列设置自动删除功能。如果将自动删除功能设为有效，通过WorkQue_Get从工作队列中获取点数据后，将从工作队列中自动删除点数据和用户数据。

如果将自动删除功能设为无效，将不删除点数据和用户数据。使用WorkQue_Remove进行删除。

如已通过WorkQue_UserData获取用户数据，将不进行自动删除。

可对每个工作队列分别设置自动删除功能的有效或无效。

参阅

WorkQue_AutoRemove函数、WorkQue_Get

WorkQue_AutoRemove使用示例

```
WorkQue_AutoRemove 1, True
```

3.26.12 WorkQue_AutoRemove函数

返回工作队列中设置的自动删除功能的状态。

格式

WorkQue_AutoRemove (工作队列编号)

参数

工作队列编号
以整数值(1～16)指定工作队列编号。

返回值

将指定工作队列的自动删除功能设为有效时返回“True”，设为无效时返回“False”。

参阅

WorkQue_AutoRemove、WorkQue_Get

WorkQue_AutoRemove函数使用示例

```
Boolean autoremove

autoremove = WorkQue_AutoRemove (1)
```

3.26.13 WorkQue_Get函数

从指定的工作队列中返回点数据。

格式

WorkQue_Get (工作队列编号 [, 索引])

参数

工作队列编号
　　以整数值(1～16)指定工作队列编号。
索引
　　以整数值指定要获取的队列数据索引。(开头的索引编号是0。)可省略。

返回值

从指定的工作队列中返回点数据。

说明

WorkQue_Get用于从工作队列中获取点数据。如果省略索引，将返回队列数据的开头数据。如果已设置索引，将返回设置的索引的点数据。

如果通过WorkQue_AutoRemove，将队列数据的自动删除功能设为有效，将通过WorkQue_Get删除点数据和用户数据。

如果设为无效，将不删除点数据和用户数据。使用WorkQue_Remove进行删除。

参阅

WorkQue_AutoRemove、WorkQue_Len、WorkQue_Reject、WorkQue_Remove、WorkQue_Sort

WorkQue_Get函数使用示例

```
' 用于跳到队列开头的工件并进行跟踪
Jump WorkQue_Get(1)
On gripper
Wait .1
Jump place
Off gripper
Wait .1
WorkQueRemove 1
```

3.26.14 WorkQue_Len函数

返回注册在指定的工作队列的有效工作队列数据的数值。

格式
WorkQue_Len (工作队列编号)

参数

工作队列编号
以整数值(1～16)指定工作队列编号。

返回值
以整数值返回有效的工作队列数据的注册量。

说明
返回有效的工作队列数据的注册量。

还可以作为Wait命令的自变量使用。

参阅
WorkQue_Add、WorkQue_Get、WorkQue_Remove

WorkQue_Len函数使用示例

```
Do
  Do While WorkQue_Len(1) > 0
    WorkQue_Remove 1, 0
  Loop
  If WorkQue_Len(1) > 0 Then
    Jump WorkQue_Get(1, 0) C0
    On gripper
    Wait .1
    WorkQue_Remove 1, 0
    Jump place
    Off gripper
    Jump idlePos
  EndIf
Loop
```

3.26.15 WorkQue_List

显示指定工作队列的工作队列数据一览(点数据和用户数据)。

格式

WorkQue_List 工作队列编号 [, 显示数]

参数

工作队列编号
　　以整数值(1～16)指定工作队列编号。
显示数量
　　以整数值指定显示数的数据量。可省略。如果省略，将显示所有队列数据。

注意

仅可利用命令窗口执行该命令。

参阅

WorkQue_Add、WorkQue_Get、WorkQue_Remove

WorkQue_使用示例

利用命令窗口的操作示例

```
> WorkQue_List 1
Queue 0    = XY(     1.000,    1.000,    0.000,    0.000 )  /R /0 (   0.000)
Queue 1    = XY(     3.000,    1.000,    0.000,    0.000 )  /R /0 (   2.000)
Queue 2    = XY(     4.000,    1.000,    0.000,    0.000 )  /R /0 (   3.000)
Queue 3    = XY(     5.000,    1.000,    0.000,    0.000 )  /R /0 (   4.000)
Queue 4    = XY(     6.000,    1.000,    0.000,    0.000 )  /R /0 (   5.000)
```

3.26.16 WorkQue_Reject

设置和显示在指定的工作队列中防止点数据重复注册的最小距离。

格式

WorkQue_Reject 工作队列编号 [, 防止重复注册距离]

参数

工作队列编号
　　以整数值(1～16)指定工作队列编号。
防止重复注册距离
　　以实值指定防止重复注册的距离的最小值(单位：mm)。如果指定了负值，将设置为0 mm。仅限从命令窗口执行时可省略。如果省略，将显示当前防止重复注册距离。

说明

设置防止点数据重复注册的最小距离。即使通过WorkQue_Add注册了小于最小距离的点数据，也不会注册在工作队列中。WorkQue_Reject是用于防止重复注册的系统过滤器。默认为0 mm。

通过WorkQue_Add添加工作队列数据(点数据和用户数据)之前，需执行WorkQue_Reject。

可对每个工作队列分别设置防止重复注册距离。

参阅

WorkQue_Add、WorkQue_Reject函数

WorkQue_Reject使用示例

```
WorkQue_Reject 1, 2.5
```

3.26.17 WorkQue_Reject函数

返回指定工作队列中设置的防止重复注册距离。

格式

WorkQue_Reject (工作队列编号)

参数

工作队列编号
以整数值(1～16)指定工作队列编号。

返回值

返回实数值(单位：mm)。

参阅

WorkQue_Add、WorkQue_Reject

WorkQue_Reject函数使用示例

```
Real rejectDist

RejectDist = WorkQue_Reject(1)
```

3.26.18 WorkQue_Remove

指定工作队列数据中删除工作队列数据(点数据和用户数据)。

格式

WorkQue_Remove 工作队列编号 [, 索引 | All]

参数

工作队列编号
　　以整数值(1～16)指定传送带的编号。
索引
　　以整数值指定要删除的工作队列数据的索引。(开头的索引编号是0。)可省略。如要从工作队列中删除所有工作队列数据，请以ALL指定。

说明

从工作队列数据中删除1个以上的工作队列数据(点数据和用户数据)。在用完工作队列数据时，删除队列数据。

参阅

WorkQue_Add

WorkQue_Remove使用示例

```
Jump WorkQue_Get(1)
On gripper
Wait .1
Jump place
Off gripper
Wait .1

' 用于从工作队列中删除数据
WorkQue_Remove 1
```

3.26.19 WorkQue_Sort

设置和显示指定工作队列的Sort方法。

格式

WorkQue_Sort 工作队列编号 [, Sort方法]

参数

工作队列编号
　　以整数值(1～16)指定工作队列编号。
Sort方法
　　以整数值(0～6)或下述常数指定Sort方法。仅限从命令窗口执行时可省略。如果省略，将显示当前Sort方法。

常数	值	内容
QUE_SORT_NONE	0	无排序(注册到工作队列的顺序)
QUE_SORT_POS_X	1	X坐标升序
QUE_SORT_INV_X	2	X坐标降序
QUE_SORT_POS_Y	3	Y坐标升序
QUE_SORT_INV_Y	4	Y坐标降序
QUE_SORT_POS_USER	5	用户数据(实数)升序
QUE_SORT_INV_USER	6	用户数据(实数)降序

说明

在工作队列中设置Sort方法。通过WorkQue_Add添加点数据和用户数据后，将按照设置的Sort方法注册在工作队列中。

通过WorkQue_UserData重新设置用户数据后，将按照设置的Sort方法更改工作队列的排列顺序。

通过WorkQue_Add添加工作队列数据(点数据和用户数据)之前，需执行WorkQue_Sort。

通过WorkQue_UserData重新设置用户数据之前，需执行WorkQue_Sort。

可对每个工作队列分别设置Sort方法。

参阅

WorkQue_Add, WorkQue_UserData

WorkQue_Sort使用示例

```
WorkQue_Sort 1, QUE_SORT_POS_X
```

3.26.20 WorkQue_Sort函数

返回指定工作队列中设置的Sort方法。

格式

WorkQue_Sort (工作队列编号)

参数

工作队列编号
以整数值(1～16)指定工作队列编号。

返回值

以整数值返回工作队列中设置的Sort方法。

- 0 = 排序(注册到工作队列的顺序)
- 1 = X坐标升序
- 2 = X坐标降序
- 3 = Y坐标升序
- 4 = Y坐标降序
- 5 = 用户数据(实数)升序
- 6 = 用户数据(实数)降序

参阅

WorkQue_Add、WorkQue_Sort、WorkQue_UserData

WorkQue_Sort函数使用示例

```
Integer quesort

quesort = WorkQue_Sort(1)
```

3.26.21 WorkQue_UserData

重新设置和显示指定工作队列中注册的用户数据(实数)。

格式

WorkQue_UserData 工作队列编号 [, 索引] [, 用户数据]

参数

工作队列编号
　　以整数值(1～16)指定工作队列编号。
索引
　　以整数值指定工作队列数据的索引。(开头的索引编号是0。)仅限从命令窗口执行时可省略。
用户数据
　　以实数值指定要重新设置的用户数据。仅限从命令窗口执行时可省略。如果省略，将显示当前用户数据(实数)。

说明

重新设置和显示工作队列中注册的用户数据。

如果已通过WorkQue_Sort设置了下述Sort方法，将按照设置的Sort方法更改工作队列数据的排列顺序。

- QUE_SORT_POS_USER：用户数据(实数)升序
- QUE_SORT_INV_USER：用户数据(实数)降序

参阅

WorkQue_UserData函数

WorkQue_UserData使用示例

```
WorkQue_UserData 1, 1, angle
```

3.26.22 WorkQue_UserData函数

返回指定工作队列中注册的用户数据(实数)。

格式

WorkQue_UserData (工作队列编号 [, 索引])

参数

工作队列编号
　　以整数值(1～16)指定工作队列编号。
索引
　　以整数值指定工作队列数据的索引。(开头的索引编号是0。)可省略。

返回值

返回实值。

参阅

WorkQue_UserData

WorkQue_UserData函数使用示例

```
' 从队列中删除
angle = WorkQue_UserData(1)  '默认索引为"0"
Jump WorkQue_Get(1) :U(angle)
WorkQue_Remove 1
```

3.26.23 Wrist

用于设置点的手腕姿势。

格式

(1) Wrist 指定点 [, Flip | NoFlip]

(2) Wrist

参数

点指定
　　以P编号、P(表达式)、点标签之一进行指定。
Flip | NoFlip
　　指定手腕姿势。

结果

如果省略2个参数，则显示机器人当前位置的手腕姿势。

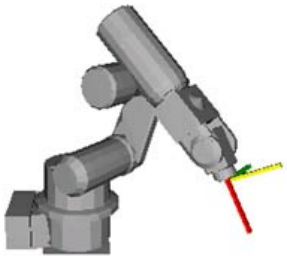
如果省略Flip | NoFlip ，则显示指定点的手腕姿势。

参阅

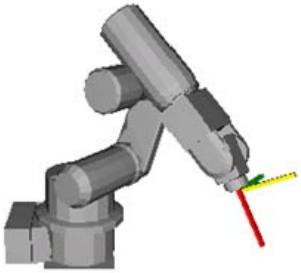
Elbow、Hand、J4Flag、J6Flag、Wrist函数

Wrist使用示例

```
Wrist P0, Flip
Wrist P(mypoint), NoFlip

P1 = 320.000, 400.000, 350.000, 140.000, 0.000, 150.000
```



```
Wrist P1, NoFlip
Go P1
```



```
Wrist P1, Flip
Go P1
```

3.26.24 Wrist函数

用于返回点的手腕姿势。

格式

Wrist [(指定点)]

参数

点指定

以P编号、P(表达式)、点标签、点表达式之一进行指定。可省略，如果省略指定点，则返回当前机器人位置的手腕姿势。

返回值

- 1: NoFlip (/NF)
- 2: Flip (/F)

参阅

Elbow、Hand、J4Flag、J6Flag、Wrist

Wrist函数使用示例

```
Print Wrist(pick)
Print Wrist(P1)
Print Wrist
Print Wrist(P1 + P2)
```

3.26.25 Write

用于将字符串写入到文件或通信端口中。不附加行末终止符。

格式

Write #端口编号, 字符串

参数

端口编号
　　是表示文件或通信端口的ID编号。文件编号是由ROpen、WOpen、AOpen等语句指定的编号。通信端口编号是由OpenCom(RS-232C)或OpenNet(TCP/IP)语句指定的编号。

字符串
　　指定要写入的字符串。

说明

Write命令不同于Print命令，不附加行末终止符。

注意

- 支持的控制器型号

  若在T/VT系列中指定RS-232C端口时会发生错误。

- 向文件写入时进行缓冲。

  可利用Flush语句写入被缓冲的数据。利用Close语句关闭文件时也进行写入。

参阅

Print, Read, WriteBin

Write使用示例

```
OpenCom #1
For i = 1 to 10
    Write #1, data$(i)
Next i
CloseCom #1
```

3.26.26 WriteBin

用于将二进制数据写到文件或通信端口中。

格式

WriteBin #端口编号, 写入数据

WriteBin #端口编号, 数组变量名(), 字节数

参数

端口编号
　　是表示文件或通信端口的ID编号。文件编号是由BOpen等语句指定的编号。通信端口编号是由OpenCom(RS-232C)或OpenNet(TCP/IP)语句指定的编号。
写入数据
　　以整数或表达式指定要写入的数据。
数组变量名()
　　指定保存写出数据字节的Byte型变量、整数变量或Long型变量的名称。可指定一维数组变量。
字节数
　　指定要写出的字节数。需为最大数组下标以下且小于256Byte。以通信端口(TCP/IP)为对象时，需为最大数组下标以下且小于1024Byte。

注意

- 支持的控制器型号

  若在T/VT系列中指定RS-232C端口时会发生错误。

参阅

ReadBin、Write

WriteBin使用示例

```
Integer i, data(100)

OpenCom #1
For i = 0 To 100
  WriteBin #1, i
Next i
WriteBin #1, data(), 100
CloseCom #1
```

3.27 X

3.27.1 Xor运算符

以位为单位对2个值进行Xor运算(异或运算)。

格式

result = 值1 Xor 值2

参数

值1、值2
　　指定数值或变量名。
result
　　返回整数。

结果

以位为单位返回Xor运算result。

说明

Xor运算符用于以位为单位进行Xor运算。result位是以位为单位对2个值进行Xor的结果。

值1的位	值2的位	result
0	0	0
0	1	1
1	0	1
1	1	0

参阅

And、LShift、Not、Or、Rshift

Xor运算符使用示例

```
>print 2 Xor 6
 4
>
```

3.27.2 Xqt

用于执行由函数名指定的程序并生成任务。

格式

Xqt [任务编号,] 函数名 [(自变量列表)] [, Normal | NoPause | NoEmgAbort]

参数

任务编号
　以1～32的整数指定要执行任务的任务编号。可省略。为后台任务时，指定65～80的整数。
函数名称
　指定要执行的函数名。
自变量列表
　指定调用时赋予函数的自变量列表。存在多个自变量时，请利用逗号进行分隔。可省略。
任务类型
　可省略。通常会省略。为后台任务时，任务类型指定没有意义。
Normal
　生成通常的任务。
NoPause
　发生Pause语句或Pause输入信号时，以及要在安全门打开的状态下生成不暂停的任务时进行指定。
NoEmgAbort
　紧急停止时以及要在发生错误时生成继续处理的任务时指定。

说明

Xqt用于开始指定的函数并立即进行返回。

通常无需任务编号参数。如果省略任务编号，SPEL+则自动在函数上附加任务编号，因此，用户不必管理任务编号。

注意

- 任务类型

  通过按任务类型指定NoPause或NoEmgAbort，可生成监视控制器全体的任务。

  但强烈建议在充分理解SPEL+任务的动作和特殊任务的限制事项之后使用这些任务。

  有关特殊任务的详细说明，请参阅以下手册。
  《Epson RC+ 用户指南 - 特殊任务》

- 后台任务

  通过后台任务执行Xqt命令时，生成的任务也变为后台任务。

  通过后台任务执行主函数时，请使用StartMain命令。

  有关后台任务的详细说明，请参阅以下手册。
  《Epson RC+ 用户指南 - 特殊任务》

不能在NoEmgAbort任务和后台任务中执行的命令

不能在NoEmgAbort任务和后台任务中执行以下命令。

A	Accel
	AccelR
	AccelS
	AIO_TrackingStart

	AIO_TrackingEnd
	Arc
	Arc3
	Arch
	Arm
	ArmCalib
	ArmCalibCLR
	ArmCalibSET
	ArmClr
	ArmSet
	AutoLJM
	AutoOrientationFlag
	AvoidSingularity
B	Base
	BGo
	BMove
	Box
	BoxClr
	Brake
C	Calib
	Cnv_AbortTrack
	Cnv_Accel
	Cnv_AccelLim
	Cnv_Adjust
	Cnv_AdjustClear
	Cnv_AdjustGet
	Cnv_AdjustSet
	Cnv_DownStream
	Cnv_Fine
	Cnv_Mode
	Cnv_OffsetAngle
	Cnv_PosErrOffset
	Cnv_QueAdd
	Cnv_QueMove
	Cnv_QueReject

	Cnv_QueRemove
	Cnv_QueUserData
	Cnv_Trigger
	Cnv_UpStream
	CollisionDetect
	CP
	CP_Offset
	Curve
	CVMove
E	ECP
	ECPClr
	ECPSet
F	Find
	Fine
	FineDist
G	Go
H	Hand_On
	Hand_Off
	Home
	HomeClr
	HomeSet
	Hordr
I	Inertia
J	JTran
	Jump
	Jump3
	Jump3CP
	JRange
L	LatchEnable
	LimitTorque
	LimZ
	LimZMargin
	Local
	LocalClr
M	MCal

	MCordr
	Motor
	Move
O	OLAccel
P	Pass
	PerformMode
	Pg_LSpeed
	Pg_Scan
	Plane
	PlaneClr
	Power
	PTPBoost
	Pulse
Q	QP
	QPDecelR
	QPDecelS
R	Range
	Reset *1
	Restart *2
S	Sense
	SetLatch
	SFree
	SF_LimitSpeedS
	SF_LimitSpeedSEnable
	SF_RealSpeedS
	SingularityAngle
	SingularityDist
	SingularitySpeed
	SLock
	SoftCP
	Speed
	SpeedFactor
	SpeedR
	SpeedS
	SyncRobots

T	TC
	TGo
	Till
	TLSet
	TLClr
	TMove
	Tool
	Trap
V	VCal
	VcalPoints
	VCls
	VCreateCalibration
	VCreateObject
	VCreateSequence
	VDefArm
	VDefGetMotionRange
	VDefLocal
	VDefSetMotionRange
	VDefTool
	VDeleteCalibration
	VDeleteObject
	VDeleteSeuence
	VEditWindow
	VGet
	VGoCenter
	VLoad
	VLoadModel
	VRun
	VSave
	VSaveImage
	VSaveModel
	VSet
	VShowModel
	VStasShow
	VStatsReset

	VStatsResetAll
	VStatsSave
	VSD
	VStatsShow
	VTeach
	VTrain
W	WaitPos
	Weight
	WorkQue_Add
	WorkQue_Reject
	WorkQue_Remove
	WorkQue_Sort
	WorkQue_UserData
X	Xqt *3
	XYLim

- *1 Reset Error可执行
- *2 可通过Trap Error的处理任务执行
- *3 可通过后台任务执行

请勿采取在循环语句中频繁重复XQT命令的使用方法。请勿采取在Do…Loop等循环语句中频繁重复XQT命令的使用方法。否则可能会导致控制器进入挂机状态。如要采取这种使用方法，请追加Wait命令(Wait 0.1)。

参阅

Function/Fend、Halt、Resume、Quit、Startmain、Trap

Xqt使用示例

```
Function main
    Xqt flash           '开始任务flash
    Xqt Cycle(5)        '开始任务Cycle

    Do
        Wait 3          '执行任务flash 3秒钟
        Halt flash      '暂停任务

        Wait 3
        Resume flash    '重新开始任务
    Loop
Fend

Function Cycle(count As Integer)
    Integer i

    For i = 1 To count
        Jump pick
        On vac
        Wait .2
        Jump place
        Off vac
        Wait .2
```

```
    Next i
Fend

Function flash
    Do
        On 1
        Wait 0.2
        Off 1
        Wait 0.2
    Loop
Fend
```

3.27.3 XY函数

以点数据返回指定的坐标值。

格式
XY (x坐标要素, y坐标要素, z坐标要素, u坐标要素 [, v坐标要素, w坐标要素])

参数

x坐标元素
　　以实值指定x坐标。
y坐标元素
　　以实值指定y坐标。
z坐标元素
　　以实值指定z坐标。
u坐标元素
　　以实值指定u坐标。
v坐标元素
　　以实值指定v坐标。是用于垂直6轴型机器人(包括N系列)的参数，可省略。
w坐标元素
　　以实值指定w坐标。是用于垂直6轴型机器人(包括N系列)的参数，可省略。

返回值
以点数据返回指定的坐标值。

说明
不使用附加轴(S轴、T轴)时无需注意。

通过采取Go XY (60，30，-50，45) 这样的的使用方法，可将机器人移动到指定的坐标值位置。

使用附加轴(S轴和T轴)时需要注意。

XY函数仅用于返回附加轴以外的机器人的点数据。

采取Go XY (60，30，-50，45) 这样的使用方法时，机器人移动到指定的坐标值位置，但附加轴不动作。在同时使附加轴动作的情况下，请进行Go XY (60，30，-50，45) : ST (10，20) 这样的指定。

有关详细信息，请参阅以下手册。《Epson RC+ 用户指南 - 附加轴》

参阅
JA, P# = 指定点、ST函数

XY函数使用示例

```
P10 = XY(60, 30, -50, 45) + P20
```

3.27.4 XYLim

用于设置和显示容许动作区域。

格式

XYLim X轴下限位置, X轴上限位置, Y轴下限位置, Y轴上限位置 [, Z轴下限位置] [, Z轴上限位置]

XYLim

参数

X轴下限位置
　　以数值或表达式指定机械手可进行动作的下限位置X坐标值(实数)。
X轴上限位置
　　以数值或表达式指定机械手可进行动作的上限位置X坐标值(实数)。
Y轴下限位置
　　以数值或表达式指定机械手可进行动作的下限位置Y坐标值(实数)。
Y轴上限位置
　　以数值或表达式指定机械手可进行动作的上限位置Y坐标值(实数)。
Z轴下限位置
　　以数值或表达式指定机械手可进行动作的下限位置Z坐标值(实数)。可省略。
Z轴上限位置
　　以数值或表达式指定机械手可进行动作的上限位置Z坐标值(实数)。可省略。

结果

如果省略参数，则显示当前的XYLim值。

说明

XYLim用于设置容许动作区域。在许多机器人系统中，用户都可以设置关节的容许动作范围，但使用SPEL+语言时，不仅可设置关节的动作范围，还可以设置容许动作范围。这样就可以根据机器人的应用来限制运转范围。

利用XYLim值设置的动作范围，会应用在由XYLimMode命令设置的监控方式中。有关监控方式的详细信息，请参阅XYLimMode语句。

机器人参数数据被保存到控制器内的小型闪存卡中。因此，如果执行本命令，将向小型闪存卡进行写入操作。过于频繁地向小型闪存卡执行写入会影响到卡的使用寿命。建议将本命令的执行控制在所需最低限度。

注意

- 将容许动作区域的设置设为OFF

  有很多用途不必设置容许动作区域。因此可简单地将该设置设为OFF。要将该设置设为OFF时，请将各参数值(X轴下限位置、X轴上限位置、Y轴下限位置、Y轴上限位置)设为“0”。例: XYLim 0, 0, 0, 0.

- 容许动作限制值的默认值

  XYLim的默认值为“0，0，0，0”。(容许动作区域的设置为OFF状态。)

提示

- 指向并单击XYLim进行设置

  在Epson RC+中，可通过［工具］菜单 -［机器人管理器］的［XY限定］面板设置XYLim值。

参阅

Range, XYLimMode

XYLim使用示例

如下所示为通过命令窗口设置XYLim值并显示当前值的简单操作示例。

```
> XYlim -200, 300, 0, 500

> XYLim
-200.000, 300.000, 0.000, 500.000
```

3.27.5 XYLim函数

用于返回设置的XY平面容许动作区域。

格式

XYLim (引用数据)

参数

引用数据

以整数值指定作为返回值返回的容许区域。

- 1：下限值
- 2：上限值

返回值

如果将引用数据指定为1，则将由XYLim设置值指定的X轴下限位置作为点数据X返回；将Y轴下限位置作为点数据Y返回；将Z轴下限位置作为点数据Z返回。

如果将引用数据指定为2，则将由XYLim设置值指定的X轴上限位置作为点数据X返回；将Y轴上限位置作为点数据Y返回；将Z轴上限位置作为点数据Z返回。

参阅

XYLim

XYLim函数使用示例

```
P1 = XYLim(1)
P2 = XYLim(2)
```

3.27.6 XYLimClr

用于清除已设置的XYLim。

格式

XYLimClr

参阅

XYLim、XYLimDef

XYLimClr函数使用示例

如下所示为使用XYLimClr函数的程序。

```
Function ClearXYLim

   If XYLimDef = True Then
      XYLimClr
   EndIf
Fend
```

3.27.7 XYLimDef函数

用于返回是否设置XYLim。

格式
XYLimDef

返回值
如果已设置XYLim，则返回“True”；如果未设置，则返回“False”。

参阅
XYLim、XYLimClr

XYLimDef函数使用示例
如下所示为使用XYLimDef函数的程序。

```
Function ClearXYLim

   If XYLimDef = True Then
      XYLimClr
   EndIf
Fend
```

3.27.8 XYLimMode

设定和显示XYLim的监控方式。

格式

(1) XYLimMode 监控方式

(2) XYLimMode

参数

监控方式

代表所使用的XYLim监控方式的证书表达式

常数	值	内容
XYLIM_STANDARD	0	将XYLim应用于动作命令的目标坐标。(不影响Pulse。)
XYLIM_STRICT	1	除了XYLIM_STANDARD的监控方式外，XYLim还应用于运动轨迹和脉冲运动。

结果

当省略参数时，则显示当前设置的XYLim监控方式。

说明

XYLimMode设置指定机器人的XYLim监控方式。

如果指定XYLIM_STANDARD，则XYLim设置的操作范围只对动作命令的目标坐标有效。不适用于从动作起点到目标坐标的运动轨迹。因此，在操作过程中，机械手可能会超出XYLim设定的范围。在此模式下，XYLim不适用于脉冲运动。

指定XYLIM_STRICT时，XYLim中设定的运动范围适用于，动作命令的目标坐标和动作起点到目标坐标的运动轨迹。因此，当机械手超过XYLim设置的范围，则会出现错误。在此模式下，XYLim也适用于脉冲运动。但如果期限在XYLim范围外，目标坐标在XYLim范围内时，从范围外移动外范围内则不会出线XYLim范围外的错误。

建议使用XYLIM_STRICT，来防止与机器人周边设备的干涉。

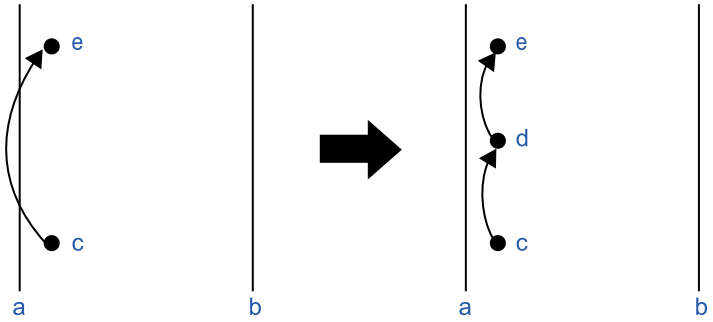
用户可通过Epson RC+中控制器的设置，来修改控制器启动时的监控方式。XYLimMode命令中设置的值，仅在控制器重启之前有效。当控制器重启XYLim的监控方式会恢复为控制器设置中，指定的监控方式。

> ⚠ 注意
>
> **使用XYLIM_STANDARD时，机械手可以在XYLim设定的范围外运动。此时，设定XYLim时，需考虑与周围设备的空间，在机器人低速运动的模式下，检查XYLim边界范围时的动作，避免机器人与周边设备干涉。**

常见错误

- 在XYLim边界附近指定PTP操作

  在执行Go命令等PTP动作时，起点和目标左边如下图所示轨迹移动。XYLIM_STRICT中，目标坐标在设置的范围内但运动轨迹超出了设定范围。此时，可以如下图中右边所示，在起点和目标左边之间，添加一个经由坐标来避免运动轨迹超出XYLim设定的范围。



符号	说明
a	XYLim下限位置
b	XYLim上限位置
c	起点
d	经由坐标
e	目标坐标

- 使用旧程序时

  当控制器的固件版本低于Ver 7.5.2.0或7.5.52.0时，将XYLim的监控方式设置为XYLIM_STRICT，可能会因为运动轨迹超出设定范围而出线错误。此时，请修改程序，可通过添加经由坐标等方式，确保运动轨迹在设定的范围内。

参阅

XYLim

XYLimMode 命令使用例

以下为使用XYLimMode的示例。使用XYLIM_STANDARD从当前位置移动到P1，使用XYLIM_STRICT从P1移动到P2。

```
Function XYLimMode_sample
Motor On
XYLimMode XYLIM_STANDARD
Go P1   'XYLim仅适用于目标坐标
XYLimMode XYLIM_STRICT
Go P2   'XYLim适用于目标坐标和运动轨迹
Fend
```

以下是在命令窗口中使用XYLimMode的示例。显示当前的XYLim监控方式。

```
> XYLimMode
1
```

3.27.9 XYLimMode函数

获取XYLim中设定的监控方式。

格式

XYLimMode

返回值

返回XYLim中设定的监控方式。

- 0 = XYLim适用于运动命令的目标坐标。(不适用于脉冲动作)
- 1 = XYLim适用于运动命令的目标左边和动作轨迹。(适用于脉冲动作)

参照

XYLimMode

XYLimMode函数使用例

以下为使用XYLimMode函数的示例。该程序可以在变量中获取并显示XYLim的监控方式。

```
Function XYLimMode_sample
Integer iVar

iVar = XYLimMode
Print iVar

Fend
```

4. Appendix A：SPEL+ 命令使用条件

4.1 Appendix A:SPEL+ 命令使用条件

- 命令窗口:在命令窗口中使用。
- 程序：可在SPEL+程序中作为语句使用。
- 函数：可用作函数使用。

命令		命令窗口		程序	函数
		RC+	TP3/TP4		
A	AbortMotion	○	○	○	-
	Abs	-	-	○	○
	Accel	○	○	○	○
	AccelMax	-	-	○	○
	AccelR	○	○	○	○
	AccelS	○	○	○	○
	Acos	-	-	○	○
	Agl	-	-	○	○
	AglToPls	-	-	○	○
	AIO_In	○	○	-	○
	AIO_InW	○	○	-	○
	AIO_Out	○	○	○	○
	AIO_OutW	○	○	○	○
	AIO_Set	○	○	○	○
	AIO_TrackingSet	○	○	○	-
	AIO_TrackingStart	-	-	○	-
	AIO_TrackingEnd	-	-	○	-
	AIO_TrackingOn	○	○	○	○
	Align	-	-	○	○
	AlignECP	-	-	○	○
	And	-	-	○	-
	AOpen	○	○	○	-
	Arc	○	○	○	-
	Arc3	○	○	○	-
	Arch	○	○	○	○
	AreaCorrection	○	○	○	○
	AreaCorrectionClr	○	○	○	-
	AreaCorrectionDef	○	○	○	○

命令		命令窗口		程序	函数
		RC+	TP3/TP4		
	AreaCorrectionInv	○	○	○	○
	AreaCorrectionOffset	○	○	○	○
	AreaCorrectionSet	○	○	○	-
	Arm	○	○	○	○
	ArmClr	○	○	○	-
	ArmDef	-	-	○	○
	ArmSet	○	○	○	○
	Asc	-	-	○	○
	ArmCalib	○	○	○	○
	ArmCalibClr	○	○	○	-
	ArmCalibDef	-	-	○	○
	ArmCalibSet	○	○	○	○
	Asin	-	-	○	○
	Atan	-	-	○	○
	Atan2	-	-	○	○
	ATCLR	○	○	○	-
	AtHome	-	-	○	○
	ATRQ	○	○	○	○
	AutoLJM	○	○	○	○
	AvoidSingularity	○	○	○	○
B	Base	○	○	○	○
	BClr	-	-	○	○
	BClr64	-	-	○	○
	BGo	○	○	○	-
	BMove	○	○	○	-
	Boolean	-	-	○	-
	BOpen	○	○	○	-
	Box	○	○	○	○
	BoxClr	○	○	○	-
	BoxDef	-	-	○	○
	Brake	○	○	○(仅函数)	○
	BSet	-	-	○	○
	BSet64	-	-	○	○

命令		命令窗口		程序	函数
		RC+	TP3/TP4		
	BTst	-	-	○	○
	BTst64	-	-	○	○
	Byte	-	-	○	-
C	Calib	○	○	○	-
	Call	-	-	○	-
	CalPls	○	○	○	○
	ChDir	○	○	○	-
	ChDisk	○	○	○	-
	ChDrive	○	○	○	-
	ChkCom	-	-	○	○
	ChkNet	-	-	○	○
	Chr$	-	-	○	○
	ClearHistory	○	-	-	-
	ClearPoints	○	○	○	-
	Close	○	○	○	-
	CloseCom	○	○	○	-
	CloseDB	○	○	○	-
	CloseNet	○	○	○	-
	Cls	○	○	○	-
	Cnv_AbortTrack	○	○	○	-
	Cnv_Accel	○	○	○	○
	Cnv_AccelLim	○	○	○	○
	Cnv_Adjust	○	○	○	-
	Cnv_AdjustClear	○	○	○	-
	Cnv_AdjustGet	○	○	○	○
	Cnv_AdjustSet	○	○	○	-
	Cnv_Downstream	○	○	○	○
	Cnv_Fine	○	○	○	○
	Cnv_LPulse	-	-	○	○
	Cnv_Mode	○	○	○	○
	Cnv_Name$	-	-	○	○
	Cnv_Number	-	-	○	○
	Cnv_OffsetAngle	○	○	○	○

命令		命令窗口		程序	函数
		RC+	TP3/TP4		
	Cnv_Point	-	-	○	○
	Cnv_PosErr	-	-	○	○
	Cnv_PosErrOffset	-	-	○	○
	Cnv_Pulse	-	-	○	○
	Cnv_QueAdd	○	○	○	-
	Cnv_QueGet	-	-	○	○
	Cnv_QueLen	-	-	○	○
	Cnv_QueList	○	○	-	-
	Cnv_QueMove	○	○	○	-
	Cnv_QueReject	○	○	○	○
	Cnv_QueRemove	○	○	○	-
	Cnv_QueUserData	○	○	○	○
	Cnv_RobotConveyor	-	-	○	○
	Cnv_Speed	-	-	○	○
	Cnv_Trigger	○	○	○	-
	Cnv_Upstream	○	○	○	○
	CollisionDetect	○	○	○	○
	Cont	○	-	○	-
	Copy	○	○	○	-
	Cos	-	-	○	○
	CP	○	○	○	○
	Ctr	-	-	○	○
	CTReset	○	○	○	-
	CtrlDev	-	-	○	○
	CtrlInfo	-	-	○	○
	CurDir$	-	-	○	○
	CurDisk$	-	-	○	○
	CurDrive$	-	-	○	○
	CurPos	-	-	○	○
	Curve	○	○	○	-
	CVMove	○	○	○	-
	CP_Offset	○	○	○	○
	CR	○	○	○	○

命令		命令窗口		程序	函数
		RC+	TP3/TP4		
	CS	○	○	○	○
	CT	○	○	○	○
	CU	○	○	○	○
	CV	○	○	○	○
	CW	○	○	○	○
	CX	○	○	○	○
	CY	○	○	○	○
	CZ	○	○	○	○
D	Date	○	○	○	-
	Date$	-	-	○	○
	Declare	-	-	○	-
	DegToRad	-	-	○	○
	Del	○	○	○	-
	DeleteDB	○	○	○	-
	DiffPoint	○	○	○	○
	DispDev	○	○	○	○
	Dist	-	-	○	○
	Do...Loop	-	-	○	-
	Double	-	-	○	-
E	ECP	○	○	○	○
	ECPClr	○	○	○	-
	ECPDef	-	-	○	○
	ECPSet	○	○	○	○
	ElapsedTime	-	-	○	○
	Elbow	○	○	○	○
	Eof	-	-	○	○
	Era	-	-	○	○
	EResume	○	○	○	-
	Erf$	-	-	○	○
	Erl	-	-	○	○
	Err	-	-	○	○
	Errb	○	○	○	○
	ErrMsg$	-	-	○	○

命令		命令窗口		程序	函数
		RC+	TP3/TP4		
	Error	○	○	○	-
	ErrorOn	-	-	○	○
	Ert	-	-	○	○
	EStopOn	-	-	○	○
	Eval	-	-	○	○
	Exit	-	-	○	-
	ExportPoints	○	○	○	-
F	FbusIO_GetBusStatus	-	-	○	○
	FbusIO_GetDeviceStatus	-	-	○	○
	FbusIO_SendMsg	○	○	○	-
	FileDataTime$	-	-	○	○
	FileExists	-	-	○	○
	FileLen	-	-	○	○
	Find	○	○	○	-
	FindPos	-	-	○	○
	Fine	○	○	○	○
	FineDist	○	○	○	○
	FineStatus	○	○	○	○
	Fix	-	-	○	○
	Flush	○	○	○	-
	FmtStr	○	○	○	-
	FmtStr$	-	-	○	○
	FolderExists	-	-	○	○
	For...Next	-	-	○	-
	FreeFile	-	-	○	○
	Function...Fend	-	-	○	-
G	GClose	○	○	○	-
	GetCurrentUser$	-	-	○	○
	GetRobotInsideBox	-	-	○	○
	GetRobotInsidePlane	-	-	○	○
	GGet	○	○	○	-
	Global	-	-	○	-
	Go	○	○	○	-

命令		命令窗口		程序	函数
		RC+	TP3/TP4		
	Gosub...Return	-	-	○	-
	Goto	-	-	○	-
	GSet	○	○	○	-
	GShow	-	-	○	-
	GShowDialog	-	-	○	○
H	Halt	-	-	○	-
	Hand	○	○	○	○
	HealthCalcPeriod	○	○	○	○
	HealthCtrlAlarmOn	○	○	○	○
	HealthCtrlInfo	○	○	○	○
	HealthCtrlRateOffset	○	○	○	-
	HealthCtrlReset	○	○	○	-
	HealthCtrlWarningEnable	○	○	○	○
	HealthRateCtrlInfo	○	○	○	○
	HealthRateRBInfo	○	○	○	○
	HealthRBAlarmOn	○	○	○	○
	HealthRBAnalysis	○	○	○	○
	HealthRBDistance	○	○	○	○
	HealthRBInfo	○	○	○	○
	HealthRBRateOffset	○	○	○	-
	HealthRBReset	○	○	○	-
	HealthRBSpeed	○	○	○	○
	HealthRBStart	○	○	○	-
	HealthRBStop	○	○	○	-
	HealthRBTRQ	○	○	○	○
	HealthRBWarningEnable	○	○	○	○
	Here	○	○	○	○
	Hex$	-	-	○	○
	History	○	-	-	-
	Hofs	○	○	○	○
	HofsJointAccuracy	○	○	○	-
	Home	○	○	○	-
	HomeClr	○	○	○	-

命令		命令窗口		程序	函数
		RC+	TP3/TP4		
	HomeDef	-	-	○	○
	HomeSet	○	○	○	○
	Hordr	○	○	○	○
	Hour	○	○	○	○
I	If...Then..Else…EndIf	-	-	○	-
	ImportPoints	○	○	○	-
	In	-	-	○	○
	InBCD	-	-	○	○
	Inertia	○	○	○	○
	InPos	-	-	○	○
	Input	○	○	○	-
	Input #	○	○	○	-
	InputBox	○	○	○	-
	InReal	-	-	○	○
	InsideBox	-	-	○	○
	InsidePlane	-	-	○	○
	InStr	-	-	○	○
	Int	-	-	○	○
	Int32	-	-	○	-
	Integer	-	-	○	-
	InW	-	-	○	○
	IODef	-	-	○	○
	IOLabel$	-	-	○	○
	IONumber	-	-	○	○
J	J1Angle	○	○	○	○
	J4Angle	○	○	○	○
	J1Flag	○	○	○	○
	J2Flag	○	○	○	○
	J4Flag	○	○	○	○
	J6Flag	○	○	○	○
	JA	-	-	○	○
	Joint	○	○	○	-
	JointAccuracy	○	○	○	○

命令		命令窗口		程序	函数
		RC+	TP3/TP4		
	JRange	○	○	○	○
	JS	-	-	○	○
	JT	-	-	○	○
	JTran	○	○	○	-
	Jump	○	○	○	-
	Jump3	○	○	○	-
	Jump3CP	○	○	○	-
	JumpTLZ	○	○	○	-
L	LatchEnable	○	○	○	-
	LatchPos	-	-	○	○
	LatchState	-	-	○	○
	LCase$	-	-	○	○
	Left$	-	-	○	○
	Len	-	-	○	○
	LimitTorque	○	○	○	○
	LimitTorqueLP	○	○	○	○
	LimitTorqueStop	○	○	○	○
	LimitTorqueStopLP	○	○	○	○
	LimZ	○	○	○	○
	LimZMargin	○	○	○	○
	Line Input	○	○	○	-
	Line Input #	○	○	○	-
	LJM	-	-	○	○
	LoadPoints	○	○	○	-
	Local	○	○	○	○
	LocalClr	○	○	○	-
	LocalDef	-	-	○	○
	Lof	-	-	○	○
	LogIn	-	-	○	○
	Long	-	-	○	-
	LSet$	-	-	○	○
	LShift	-	-	○	○
	LShift64	-	-	○	○

命令		命令窗口		程序	函数
		RC+	TP3/TP4		
	LTrim$	-	-	○	○
M	Mask	-	-	○	-
	MCal	○	○	○	-
	MCalComplete	-	-	○	○
	MCordr	○	○	○	○
	MemIn	-	-	○	○
	MemInW	-	-	○	○
	MemOff	○	○	○	-
	MemOn	○	○	○	-
	MemOut	○	○	○	-
	MemOutW	○	○	○	-
	MemSw	-	-	○	○
	MHour	-	-	○	○
	Mid$	-	-	○	○
	MkDir	○	○	○	-
	Mod	-	-	○	-
	Motor	○	○	○	○
	Move	○	○	○	-
	MsgBox	○	○	○	○
	MyTask	-	-	○	○
N	Next	-	-	○	-
	Not	-	-	○	-
O	Off	○	○	○	-
	OLAccel	○	○	○	○
	OLRate	○	○	○	○
	On	○	○	○	-
	OnErr	-	-	○	-
	OpBCD	○	○	○	-
	OpenCom	○	○	○	○
	OpenDB	○	○	○	-
	OpenNet	○	○	○	○
	Oport	-	-	○	○
	Or	-	-	○	-

命令		命令窗口		程序	函数
		RC+	TP3/TP4		
	Out	○	○	○	○
	OutReal	○	○	○	○
	OutW	○	○	○	○
P	P#	○	○	○	-
	PAgl	-	-	○	○
	Pallet	○	○	○	○
	PalletClr	○	○	○	-
	ParseStr	○	○	○	○
	Pass	○	○	○	-
	Pause	-	-	○	-
	PauseOn	-	-	○	○
	PDescription	○	○	○	-
	PDescription$	○	○	-	○
	PDef	-	-	○	○
	PDel	○	○	○	-
	PerformMode	○	○	○	○
	PG_FastStop	○	○	○	-
	PG_LSpeed	○	○	○	○
	PG_Scan	○	○	○	-
	PG_SlowStop	○	○	○	-
	PLabel	○	○	○	-
	PLabel$	-	-	○	○
	Plane	○	○	○	○
	PlaneClr	○	○	○	-
	PlaneDef	-	-	○	○
	PList	○	○	○	-
	PLocal	○	○	○	○
	Pls	-	-	○	○
	PNumber	-	-	○	○
	PosFound	-	-	○	○
	Power	○	○	○	○
	PPls	-	-	○	○
	Preserve	-	-	○	-

命令		命令窗口		程序	函数
		RC+	TP3/TP4		
	Print	○	○	○	-
	Print #	○	○	○	-
	PTCLR	○	○	○	-
	PTPBoost	○	○	○	○
	PTPBoostOK	-	-	○	○
	PTPTime	-	-	○	○
	PTran	○	○	○	-
	PTRQ	○	○	○	○
	Pulse	○	○	○	○
Q	QP	○	○	○	-
	QPDecelR	○	○	○	○
	QPDecelS	○	○	○	○
	Quit	-	-	○	-
R	RadToDeg	-	-	○	○
	Randmize	○	○	○	-
	Range	○	○	○	-
	Read	○	○	○	-
	ReadBin	○	○	○	-
	Real	-	-	○	-
	RealAccel	-	-	○	○
	RealPls	-	-	○	○
	RealPos	-	-	○	○
	RealTorque	-	-	○	○
	Recover	○	-	○	○
	RecoverPos	-	-	○	○
	Redim	○	○	○	-
	Rename	○	○	○	-
	RenDir	○	○	○	-
	Reset	○	○	○	-
	ResetElapsedTime	○	○	○	-
	Restart	○	-	○	-
	Resume	-	-	○	-
	Return	-	-	○	-

命令		命令窗口		程序	函数
		RC+	TP3/TP4		
	Right$	-	-	○	○
	RmDir	○	○	○	-
	Rnd	-	-	○	○
	Robot	○	○	○	○
	RobotInfo	-	-	○	○
	RobotInfo$	-	-	○	○
	RobotModel$	-	-	○	○
	RobotName$	-	-	○	○
	RobotSerial$	-	-	○	○
	RobotType	-	-	○	○
	ROpen	○	○	○	-
	ROTOK	○	○	○	○
	RSet$	-	-	○	○
	RShift64	-	-	○	○
	RShift	-	-	○	○
	RTrim$	-	-	○	○
	RunDialog	-	-	○	-
S	SafetyOn	-	-	○	○
	SavePoints	○	○	○	-
	Seek	○	○	○	-
	Select...Send	-	-	○	-
	SelectDB	○	○	○	○
	Sense	○	○	○	-
	SetCom	○	○	○	-
	SetIn	○	○	○	-
	SetInReal	○	○	○	-
	SetInW	○	○	○	-
	SetLatch	○	○	○	-
	SetNet	○	○	○	-
	SetSw	○	○	○	-
	SF_GetParam	○	○	○	○
	SF_GetParam$	○	○	○	○
	SF_GetStatus	○	○	○	○

命令		命令窗口		程序	函数
		RC+	TP3/TP4		
	SF_LimitSpeedS	○	○	○	○
	SF_LimitSpeedSEnable	○	○	○	○
	SF_RealSpeedS	○	○	○	○
	SF_PeakSpeedS	○	○	○	○
	SF_PeakSpeedSClear	○	○	○	-
	SFree	○	○	○	○
	Sgn	-	-	○	○
	Short	-	-	○	-
	Shutdown	○	○	○	○
	Signal	○	○	○	-
	SimGet	-	-	○	-
	SimSet	○	○	○	-
	Sin	-	-	○	○
	SingularityAngle	○	○	○	○
	SingularityDist	○	○	○	○
	SingularitySpeed	○	○	○	○
	SLock	○	○	○	-
	SoftCP	○	○	○	○
	Space$	-	-	○	○
	Speed	○	○	○	○
	SpeedFactor	○	○	○	○
	SpeedR	○	○	○	○
	SpeedS	○	○	○	○
	SPELCom_Event	○	○	○	-
	Sqr	-	-	○	○
	ST	-	-	○	○
	StartMain	-	-	○	-
	Stat	-	-	○	○
	Str$	-	-	○	○
	String	-	-	○	-
	Sw	-	-	○	○
	SyncLock	-	-	○	-
	SyncUnlock	-	-	○	-

命令		命令窗口		程序	函数
		RC+	TP3/TP4		
	SyncRobots	○	○	○	○
	SysConfig	○	○	-	-
	SysErr	-	-	○	○
T	Tab$	-	-	○	○
	Tan	-	-	○	○
	TargetOK	-	-	○	○
	TaskDone	-	-	○	○
	TaskInfo	-	-	○	○
	TaskInfo$	-	-	○	○
	TaskState	○	○	○	○
	TaskWait	○	○	○	-
	TC	○	○	○	-
	TCLim	○	○	○	○
	TCPSpeed	-	-	○	○
	TCSpeed	○	○	○	○
	TeachOn	-	-	○	○
	TGo	○	○	○	-
	Till	○	○	○	-
	TillOn	-	-	○	○
	Time	○	○	○	○
	Time$	-	-	○	○
	TLClr	○	○	○	-
	TLDef	-	-	○	○
	TLSet	○	○	○	○
	TMOut	○	○	○	-
	TMove	○	○	○	-
	Tmr	-	-	○	○
	TmReset	○	○	○	-
	Toff	○	○	○	-
	Ton	○	○	○	-
	Tool	○	○	○	○
	Trap	-	-	○	-
	Trim$	-	-	○	○

命令		命令窗口		程序	函数
		RC+	TP3/TP4		
	TW	-	-	○	○
U	UBound	-	-	○	○
	UByte	-	-	○	-
	UCase$	-	-	○	○
	UInt32	-	-	○	-
	UOpen	○	○	○	-
	UpdateDB	○	○	○	-
	UShort	-	-	○	-
V	Val	-	-	○	○
	VCal	○	○	○	-
	VCalPoints	○	○	○	-
	VCls	-	-	○	-
	VCreateCalibration	-	-	○	-
	VCreateObject	-	-	○	-
	VCreateSequence	-	-	○	-
	VDefArm	-	-	○	-
	VDefGetMotionRange	○	-	○	-
	VDefLocal	-	-	○	-
	VDefSetMotionRange	○	-	○	-
	VDefTool	-	-	○	-
	VDeleteCalibration	-	-	○	-
	VDeleteObject	-	-	○	-
	VDeleteSequence	-	-	○	-
	VGet	-	-	○	-
	VGoCenter	-	-	○	-
	VisCalib	-	-	-	-
	VisCalInfo	-	-	-	○
	VisCalLoad	-	-	-	-
	VisCalSave	-	-	-	-
	VisTrans	-	-	-	○
	VLoad	-	-	○	-
	VLoadModel	-	-	○	-
	VRun	-	-	○	-

命令		命令窗口		程序	函数
		RC+	TP3/TP4		
	VSave	-	-	○	-
	VSaveImage	-	-	○	-
	VSaveModel	-	-	○	-
	VSD	○	○	○	○
	VSet	-	-	○	-
	VShowModel	○	-	○	-
	VStatsReset	-	-	○	-
	VStatsResetAll	-	-	○	-
	VStatsSave	-	-	○	-
	VStatsShow	○	○	○	-
	VTeach	-	-	○	-
	VTrain	○	-	○	-
	VxCalib	○	○	○	-
	VxCalDelete	○	○	○	-
	VxCalLoad	○	○	○	-
	VxCalInfo	-	-	○	○
	VxCalSave	○	○	○	-
	VxTrans	-	-	○	○
W	Wait	○	○	○	-
	WaitNet	○	○	○	-
	WaitPos	○	○	○	-
	WaitSig	○	○	○	-
	Weight	○	○	○	○
	Where	○	○	-	-
	WindowStatus	-	-	○	○
	WorkQue_Add	○	○	○	-
	WorkQue_AutoRemove	○	○	○	○
	WorkQue_Get	○	○	○	○
	WorkQue_Len	○	○	○	○
	WorkQue_List	○	○	-	-
	WorkQue_Reject	○	○	○	○
	WorkQue_Remove	○	○	○	-
	WorkQue_Sort	○	○	○	○

命令		命令窗口		程序	函数
		RC+	TP3/TP4		
	WorkQue_UserData	○	○	○	○
	WOpen	○	○	○	-
	Wrist	○	○	○	○
	Write	○	○	○	-
	WriteBin	○	○	○	-
X	Xor	-	-	○	-
	Xqt	-	-	○	-
	XY	-	-	○	○
	XYLim	○	○	○	○
	XYLimClr	○	○	○	-
	XYLimDef	-	-	○	○
	XYLimMode	○	○	○	○

5. Appendix B：兼容性相关注意事项

5.1 B-1：Epson RC+ 6.0兼容性相关注意事项

5.1.1 概要

Epson RC+ 8.0与Epson RC+ 7.0兼容。

如果您是使用过RC620控制器和Epson RC+ 6.0软件的用户，在使用RC700系列控制器和Epson RC+ 7.0, Epson RC+ 8.0软件前，请关注此内容。

由于硬件、可兼容的机械手型号和关节数量的差异，Epson RC+ 7.0, Epson RC+ 8.0的部分功能与Epson RC+ 6.0不同。请事先理解这些内容，安全地使用机器人。

Epson RC+ 7.0, Epson RC+ 8.0使用了最新的设计技术，实现软件升级的同时最大程度的维持了与当前产品的兼容性。但是为了进一步提高机器人控制器的特殊性和易用性，有些部分与原有的Epson RC+ 6.0并不兼容，有些部分已删除。

有关是Epson RC+ 6.0与Epson RC+ 7.0, Epson RC+ 8.0兼容性的对比。

5.1.2 总体差异

以下为Epson RC+ 6.0与Epson RC+ 7.0, Epson RC+ 8.0的总体差异。

项目	Epson RC+ 7.0, Epson RC+ 8.0	Epson RC+ 6.0
任务数	最多32个任务（后台任务最多为16个任务）	最多32个任务（后台任务最多为16个任务）
任务类型	可指定NoPouse任务 可指定NoEmgAbort任务 可指定后台任务	可指定NoPouse任务 可指定NoEmgAbort任务 可指定后台任务
TRAP ERROR等的特殊TRAP	支持	支持
利用TRAP编号启动的任务	专用任务编号	专用任务编号
复合型机械手	支持	支持
机器人编号	1~16	1~16
Real型的有效位数	6位	6位
Double型的有效位数	14位	14位
数组下标	字符串变量以外 本地变量：2,000 全局变量：100,000 模块变量：100,000 备份变量：4,000	字符串变量以外 本地变量：2,000 全局变量：100,000 模块变量：100,000 备份变量：4,000
	字符串变量 本地变量：200 全局变量：10,000 模块变量：10,000 备份变量：400	字符串变量 本地变量：200 全局变量：10,000 模块变量：10,000 备份变量：400

项目	Epson RC+ 7.0, Epson RC+ 8.0	Epson RC+ 6.0
装置编号	21:电脑 22:远程 24:TP 20:TP3	21:电脑 22:远程 24:TP 28:LCD
控制装置	远程I/O 电脑 远程COM 远程Ethernet TP3	远程I/O 电脑
计时器编号的范围	0~63	0~63
程序容量	8MB	8MB
SyncLock、SyncUnlock的信号编号范围	0~63	0~63
WaitSig、Signal的信号编号范围	0~63	0~63
存储器I/O点数	1024	1024
I/O端口编号	与Epson RC+ 6.0通用	
以太网端口编号	201~216	201~216
远程输入输出分配	有标准分配	无默认设置
RS-232C通讯端口编号 SPEL+控制部分	1~8, 1001~1008	1~8, 1001, 1002
执行RS-232C通讯端口的OpenCom	必须	必须
文件的输入输出	支持	支持
文件访问的文件编号	30~63	30~63
数据库访问编号	501~508	501~508
VisionGuide	智能相机 图像采集卡	智能相机 图像采集卡
传送带跟踪	支持	支持
PG机器人	支持	支持
OCR	支持	支持
安全	支持	支持
VB Guide 6.0 (RC+ API)	支持	支持
现场总线I/O的使用	使用普通的I/O命令	使用普通的I/O命令
现场总线主站	不保证应答性	不保证应答性
现场总线从站	保证应答性	保证应答性
GUI Builder	支持	支持
错误编号	与Epson RC+6.0通用	

5.1.3 命令兼容性一览

- +: 存在新增功能或功能变更，但能向上兼容的命令
- -: 无变更的命令
- !: 功能更改或语法更改的命令，需注意
- !!: 有重大变更的命令，需注意
- ×: 删除的命令

命令		兼容性	备注
A	Abs函数	-	
	Accel	-	
	Accel函数	-	
	AccelMax函数	-	
	AccelR	-	
	AccelR函数	-	
	AccelS	-	
	AccelS函数	-	
	Acos函数	-	
	AglToPls函数	-	
	Agl函数	-	
	AlignECP函数	-	
	Align函数	-	
	And	-	
	Arc	-	
	Arc3	-	
	Arch	-	
	Arch函数	-	
	Arm	-	
	ArmClr	-	
	ArmDef函数	-	
	ArmSet	-	
	ArmSet函数	-	
	Arm函数	-	
	Asc函数	-	
	Asin函数	-	
	Atan2函数	-	
	Atan函数	-	
	ATCLR	-	

命令		兼容性	备注
	ATRQ	-	
	ATRQ函数	-	
B	Base	-	
	Base函数	-	
	BClr函数	-	
	BGo	+	添加动作模式指定
	BMove	-	
	Boolean	-	
	Box	+	添加远程输出逻辑设定指定
	Box函数	-	
	BoxClr函数	-	
	BoxDef函数	-	
	Brake	-	
	Brake函数	-	
	BSet函数	-	
	BTst函数	-	
	Byte	-	
C	Call	-	
	ChkCom函数	-	
	ChkNet函数	-	
	Chr$函数	-	
	ClearPoints	-	
	CloseCom	-	
	CloseNet	-	
	Cls	-	
	Cos函数	-	
	CP	-	
	CP函数	-	
	CTReset	-	
	CtrlDev函数	-	
	CtrlInfo函数	-	
	Ctr函数	-	
	CurPos函数	-	
	Curve	-	

命令		兼容性	备注
	CVMove	-	
	CX~CW	-	
	CX~CW函数	-	
D	Date	-	
	Date$函数	-	
	DegToRad函数	-	
	DispDev	-	
	DispDev函数	-	
	Dist函数	-	
	Do...Loop	-	
	Double	-	
E	ECP	-	
	ECPClr	-	
	EcpDef函数	-	
	ECPSet	-	
	ECPSet函数	-	
	ECP函数	-	
	Elbow	-	
	Elbow函数	-	
	Era函数	-	
	EResume	-	
	Erf$函数	-	
	Erl函数	-	
	ErrMsg$函数	-	
	Error	-	
	ErrorOn函数	-	
	Err函数	-	
	Ert函数	-	
	EStopOn函数	-	
	Exit	-	
	Find	-	
	FindPos函数	-	
	Fine	-	
	Fine函数	-	

命令		兼容性	备注
	Fix函数	-	
	FmtStr$	-	
	For...Next	-	
	Function...Fend	-	
G	Global	-	
	Go	+	添加动作模式指定
	Gosub...Return	-	
	Goto	-	
H	Halt	-	
	Hand	-	
	Hand函数	-	
	Here	-	
	Here函数	-	
	Hex$函数	-	
	Home	-	
	HomeClr	-	
	HomeDef函数	-	
	HomeSet	-	
	HomeSet函数	-	
	HOrdr	-	
	HOrdr函数	-	
	Hour	-	
	Hour函数	-	
I	If...EndIf	-	
	In函数	-	
	InBCD函数	-	
	Inertia	-	
	Inertia函数	-	
	InPos函数	-	
	Input	-	
	Input#	-	
	InsideBox函数	-	
	InsidePlane函数	-	
	InStr函数	-	

命令		兼容性	备注
	Integer	-	
	Int函数	-	
	InW函数	-	
	IOLabel$函数	-	
	IONumber函数	-	
J	J1Flag	-	
	J1Flag函数	-	
	J2Flag	-	
	J2Flag函数	-	
	J4Flag	-	
	J4Flag函数	-	
	J6Flag	-	
	J6Flag函数	-	
	JA函数	-	
	Joint	-	
	JRange	-	
	JRange函数	-	
	JS函数	-	
	JTran	-	
	JT函数	-	
	Jump	+	添加动作模式指定
	Jump3	-	
	Jump3CP	-	
L	LCase$函数	-	
	Left$函数	-	
	Len函数	-	
	LimZ	-	
	LimZ函数	-	
	Line Input	-	
	Line Input#	-	
	LJM函数	-	
	LoadPoints	-	
	Local	-	
	LocalClr	-	

命令		兼容性	备注
	LocalDef函数	-	
	Local函数	-	
	Lof函数	-	
	Long	-	
	LSet$函数	-	
	LShift函数	-	
	LTrim$函数	-	
M	Mask	-	
	MemInW函数	-	
	MemIn函数	-	
	MemOff	-	
	MemOn	-	
	MemOut	-	
	MemOutW	-	
	MemSw函数	-	
	Mid$函数	-	
	Mod	-	
	Motor	-	
	Motor函数	-	
	Move	-	
	MyTask函数	-	
N	Not	-	
O	Off	-	
	OLAccel	-	
	OLAccel函数	-	
	OLRate	-	
	OLRate函数	-	
	On	-	
	OnErr	-	
	OpBCD	-	
	OpenCom	-	
	OpenNet	-	
	Oport函数	-	
	Or	-	

命令		兼容性	备注
	Out	-	
	OutW	-	
	OutW函数	-	
	Out函数	-	
P	P#	-	
	PAgl函数	-	
	Pallet	+	添加坐标值指定
	Pallet函数	-	
	ParseStr	-	
	ParseStr函数	-	
	Pass	-	
	Pause	-	
	PauseOn函数	-	
	PDef函数	-	
	PDel	-	
	PLabel	-	
	PLabel$函数	-	
	Plane	-	
	Plane函数	-	
	PlaneClr	-	
	PlaneDef函数	-	
	PList	-	
	PLocal	-	
	PLocal函数	-	
	Pls函数	-	
	PNumber函数	-	
	PosFound函数	-	
	Power	-	
	Power函数	-	
	PPls函数	-	
	Print	-	
	Print#	-	
	PTCLR	-	
	PTPBoost	-	

命令		兼容性	备注
	PTPBoostOK函数	-	
	PTPBoost函数	-	
	PTPTime函数	-	
	PTran	-	
	PTRQ	-	
	PTRQ函数	-	
	Pulse	-	
	Pulse函数	-	
Q	QP	-	
	Quit	-	
R	RadToDeg函数	-	
	Randmize	-	
	Range	-	
	Read	-	
	ReadBin	-	
	Real	-	
	RealPls函数	-	
	RealPos函数	-	
	RealTorque	-	
	Redim	-	
	Reset	-	
	Resume	-	
	Return	-	
	RobotInfo函数	-	
	RobotInfo$函数	-	
	RobotModel$函数	-	
	RobotName$函数	-	
	RobotSerial$函数	-	
	RobotType函数	-	
	RSet$函数	-	
	RShift函数	-	
	RTrim$函数	-	
S	SafetyOn函数	-	
	SavePoints	-	

命令		兼容性	备注
	Select...Send	-	
	Sense	-	
	SetCom	-	
	SetInW	-	
	SetIn	-	
	SetNet	-	
	SetSw	-	
	SFree	-	
	SFree函数	-	
	Sgn函数	-	
	Signal	-	
	Sin函数	-	
	SLock	-	
	SoftCP	-	
	SoftCP函数	-	
	Space$函数	-	
	Speed	-	
	SpeedR	-	
	SpeedR函数	-	
	SpeedS	-	
	SpeedS函数	-	
	Speed函数	-	
	SPELCom_Event	-	
	Sqr函数	-	
	Stat函数	-	
	Str$函数	-	
	String	-	
	Sw函数	-	
	SyncLock	-	
	SyncUnlock	-	
	SysConfig	-	
	SysErr函数	-	
T	Tab$函数	-	
	Tan函数	-	

命令		兼容性	备注
	TargetOK函数	-	
	TaskDone函数	-	
	TaskInfo函数	-	
	TaskInfo$函数	-	
	TaskState	-	
	TaskState函数	-	
	TaskWait	-	
	TC	-	
	TCLim	-	
	TCLim函数	-	
	TCSpeed	-	
	TCSpeed函数	-	
	TGo	+	添加动作模式指定
	TillOn函数	-	
	Time	-	
	Time函数	-	
	Time$函数	-	
	TLClr	-	
	TlDef函数	-	
	TLSet	-	
	TLSet函数	-	
	TMOut	-	
	TMove	-	
	TmReset	-	
	Tmr函数	-	
	Toff	-	
	Ton	-	
	Tool	-	
	Tool函数	-	
	Trap	-	
	Trim$函数	-	
	Tw函数	-	
U	UBound函数	-	
	UCase$函数	-	

命令		兼容性	备注
V	Val函数	-	
W	Wait	-	
	WaitNet	-	
	WaitPos	-	
	WaitSig	-	
	Weight	-	
	Weight函数	-	
	Where	-	
	Wrist	-	
	Wrist函数	-	
	Write	-	
	WriteBin	-	
X	Xor	-	
	Xqt	-	
	XY函数	-	
	XYLim	-	
	XYLim函数	-	
	XYLimClr	-	
	XYLimDef	-	
	XYLimDef函数	-	

5.2 B-2：Epson RC+ 5.0兼容性相关注意事项

5.2.1 概要

Epson RC+ 8.0与Epson RC+ 7.0兼容。

如果您是使用过RC180控制器和Epson RC+ 5.0软件的用户，在使用RC700系列、RC90系列控制器和Epson RC+ 7.0, Epson RC+ 8.0软件前，请关注此内容。

由于硬件、可兼容的机械手型号和关节数量的差异，Epson RC+ 7.0, Epson RC+ 8.0的部分功能与Epson RC+ 5.0不同。请事先理解这些内容，安全地使用机器人。

Epson RC+ 7.0, Epson RC+ 8.0使用了最新的设计技术，实现软件升级的同时最大程度的维持了与当前产品的兼容性。但是为了进一步提高机器人控制器的特殊性和易用性，有些部分与原有的Epson RC+ 5.0并不兼容，有些部分已删除。

有关是Epson RC+ 5.0与Epson RC+ 7.0, Epson RC+ 8.0兼容性的对比。

5.2.2 总体差异

以下为Epson RC+ 5.0与Epson RC+ 7.0, Epson RC+ 8.0的总体差异。

项目	Epson RC+ 7.0, Epson RC+ 8.0	Epson RC+ 5.0
任务数	最多32个任务 (后台任务最多为16个任务)	最多16个任务
任务类型	可指定NoPouse任务 可指定NoEmgAbort任务 可指定后台任务	可指定NoPouse任务 可指定NoEmgAbort任务
TRAP ERROR等的特殊TRAP	支持	不支持
利用TRAP编号启动的任务	专用任务编号	专用任务编号
复合型机械手	支持	不支持
机器人编号	1~16	1
Real型的有效位数	6位	6位
Double型的有效位数	14位	14位
数组下标	字符串变量以外 本地变量：2,000 全局变量：100,000 模块变量：100,000 备份变量：4,000	字符串变量以外 本地变量：1,000 全局变量：10,000 模块变量：10,000 备份变量：1,000
	字符串变量 本地变量：200 全局变量：10,000 模块变量：10,000 备份变量：400	字符串变量 本地变量：100 全局变量：1,000 模块变量：1,000 备份变量：100
装置编号	21:电脑 22:远程 24:TP 20:TP3	21:电脑 22:远程 23:OP 24:TP
控制装置	远程I/O 电脑 远程COM 远程Ethernet TP3	远程I/O 电脑 OP1 远程Ethernet
计时器号的范围	0~63	0~15
程序容量	8MB	4MB
SyncLock、SyncUnlock的信号编号范围	0~63	0~15
WaitSig、Signal的信号编号范围	0~63	0~5
存储器I/O点数	1024	256
I/O端口编号	与Epson RC+ 5.0通用	
以太网端口编号	201~216	201~208
远程输入输出分配	有标准分配	有标准分配

项目	Epson RC+ 7.0, Epson RC+ 8.0	Epson RC+ 5.0
RS-232C通讯端口编号 SPEL+控制部分	1~8, 1001~1008	1~8
执行RS-232C通讯端口的OpenCom	必须	必须
文件的输入输出	支持	不支持
文件访问的文件编号	30~63	不支持
数据库访问编号	501~508	不支持
VisionGuide	智能相机 图像采集卡	智能相机
传送带跟踪	支持	不支持
PG机器人	支持	不支持
OCR	支持	不支持
安全	支持	不支持
VBGuide 5.0 (RC+ API)	支持	支持Lite版
现场总线I/O的使用	使用普通的I/O命令	使用普通的I/O命令
现场总线主站	不保证应答性	不保证应答性
现场总线从站	保证应答性	保证应答性
GUI Builder	支持	不支持
错误编号	与Epson RC+ 5.0通用	

5.2.3 命令兼容性一览

- +: 存在新增功能或功能变更，但能向上兼容的命令
- -: 无变更的命令
- !: 功能更改或语法更改的命令，需注意
- !!: 有重大变更的命令，需注意
- ×: 删除的命令

命令		兼容性	备注
A	Abs函数	-	
	Accel	-	
	Accel函数	-	
	AccelMax函数	-	
	AccelR	-	
	AccelR函数	-	
	AccelS	-	
	AccelS函数	-	
	Acos函数	-	
	AglToPls函数	-	

命令		兼容性	备注
	Agl函数	-	
	AlignECP函数	-	
	Align函数	-	
	And	-	
	Arc	-	
	Arc3	-	
	Arch	-	
	Arch函数	-	
	Arm	-	
	ArmClr	-	
	ArmDef函数	-	
	ArmSet	-	
	ArmSet函数	-	
	Arm函数	-	
	Asc函数	-	
	Asin函数	-	
	Atan2函数	-	
	Atan函数	-	
	ATCLR	-	
	ATRQ	-	
	ATRQ函数	-	
B	Base	-	
	Base函数	-	
	BClr函数	-	
	BGo	+	添加动作模式指定
	BMove	-	
	Boolean	-	
	Box	+	添加机器人编号的指定
	Box函数	+	添加机器人编号的指定
	BoxClr函数	+	添加机器人编号的指定
	BoxDef函数	+	添加机器人编号的指定
	Brake	-	
	Brake函数	-	
	BSet函数	-	

命令		兼容性	备注
	BTst函数	-	
	Byte	-	
C	Call	+	外部函数调用
	ChkCom函数	-	
	ChkNet函数	-	
	Chr$函数	-	
	ClearPoints	-	
	CloseCom	-	
	CloseNet	-	
	Cls	-	
	Cos函数	-	
	CP	-	
	CP函数	-	
	CTReset	-	
	CtrlDev函数	!	变更控制装置
	CtrlInfo函数	-	
	Ctr函数	-	
	CurPos函数	-	
	Curve	-	
	CVMove	-	
	CX~CW	+	添加CR、CS、CT
	CX~CW函数	+	添加CR、CS、CT
D	Date	!	仅显示
	Date$函数	-	
	DegToRad函数	-	
	DispDev	-	
	DispDev函数	-	
	Dist函数	-	
	Do...Loop	-	
	Double	-	
E	ECP	-	
	ECPClr	-	
	EcpDef函数	-	
	ECPSet	-	

命令		兼容性	备注
	ECPSet函数	-	
	ECP函数	-	
	ElapsedTime函数	-	
	Elbow	-	
	Elbow函数	-	
	Era函数	-	
	EResume	-	
	Erf$函数	-	
	Erl函数	-	
	ErrMsg$函数	-	
	Error	-	
	ErrorOn函数	-	
	Err函数	-	
	Ert函数	-	
	EStopOn函数	-	
	Exit	-	
	Find	-	
	FindPos函数	-	
	Fine	-	
	Fine函数	-	
	Fix函数	-	
	FmtStr$	-	
	For...Next	-	
	Function...Fend	-	
G	Global	-	
	Go	+	添加动作模式指定
	Gosub...Return	-	
	Goto	-	
H	Halt	-	
	Hand	-	
	Hand函数	-	
	Here	-	
	Here函数	-	
	Hex$函数	-	

命令		兼容性	备注
	Home	-	
	HomeClr	-	
	HomeDef函数	-	
	HomeSet	-	
	HomeSet函数	-	
	HOrdr	-	
	HOrdr函数	-	
	Hour	-	
	Hour函数	-	
I	If...EndIf	-	
	In函数	-	
	InBCD函数	-	
	Inertia	-	
	Inertia函数	-	
	InPos函数	-	
	Input	-	
	Input#	+	添加装置编号
	InsideBox函数	!	添加机器人编号和All的指定。无法Wait等待
	InsidePlane函数	!	添加机器人编号和All的指定。无法Wait等待
	InStr函数	-	
	Integer	-	
	Int函数	-	
	InW函数	-	
	IOLabel$函数	-	
	IONumber函数	-	
J	J1Flag	-	
	J1Flag函数	-	
	J2Flag	-	
	J2Flag函数	-	
	J4Flag	-	
	J4Flag函数	-	
	J6Flag	-	
	J6Flag函数	-	
	JA函数	-	

命令		兼容性	备注
	Joint	-	
	JRange	-	
	JRange函数	-	
	JS函数	-	
	JTran	-	
	JT函数	-	
	Jump	+	添加动作模式指定
	Jump3	+	
	Jump3CP	+	
L	LCase$函数	-	
	Left$函数	-	
	Len函数	-	
	LimZ	-	
	LimZ函数	-	
	Line Input	-	
	Line Input#	+	添加装置编号
	LJM函数	-	
	LoadPoints	-	
	Local	-	
	LocalClr	-	
	LocalDef函数	-	
	Local函数	-	
	Lof函数	-	
	Long	-	
	LSet$函数	-	
	LShift函数	-	
	LTrim$函数	-	
M	Mask	-	
	MemInW函数	-	
	MemIn函数	-	
	MemOff	-	
	MemOn	-	
	MemOut	-	
	MemOutW	-	

命令		兼容性	备注
	MemSw函数	-	
	Mid$函数	-	
	Mod	-	
	Motor	-	
	Motor函数	-	
	Move	-	
	MyTask函数	-	
N	Not	-	
O	Off	-	
	OLAccel	-	
	OLAccel函数	-	
	OLRate	-	
	OLRate函数	-	
	On	-	
	OnErr	-	
	OpBCD	-	
	OpenCom	-	
	OpenNet	-	
	Oport函数	-	
	Or	-	
	Out	-	
	OutW	-	
	OutW函数	-	
	Out函数	-	
P	P#	-	
	PAgl函数	-	
	Pallet	-	添加坐标值指定
	Pallet函数	-	
	ParseStr	-	
	ParseStr函数	-	
	Pass	+	
	Pause	-	
	PauseOn函数	-	
	PDef函数	-	

命令		兼容性	备注
	PDel	-	
	PLabel	-	
	PLabel$函数	-	
	Plane	+	添加机器人编号的指定
	Plane函数	+	添加机器人编号的指定
	PlaneClr	+	添加机器人编号的指定
	PlaneDef函数	+	添加机器人编号的指定
	PList	!	表示形式变更
	PLocal	-	
	PLocal函数	-	
	Pls函数	-	
	PNumber函数	-	
	PosFound函数	-	
	Power	-	
	Power函数	-	
	PPls函数	-	
	Print	-	
	Print#	+	变更装置编号
	PTCLR	-	
	PTPBoost	-	
	PTPBoostOK函数	-	
	PTPBoost函数	-	
	PTPTime函数	-	
	PTran	-	
	PTRQ	-	
	PTRQ函数	-	
	Pulse	-	
	Pulse函数	-	
Q	QP	-	
	Quit	-	
R	RadToDeg函数	-	
	Randmize	-	
	Range	-	
	Read	-	

命令		兼容性	备注
	ReadBin	-	
	Real	-	
	RealPls函数	-	
	RealPos函数	-	
	RealTorque	-	
	Redim	-	
	Reset	-	
	ResetElapsedTime	-	
	Resume	-	
	Return	-	
	RobotInfo函数	+	添加信息
	RobotInfo$函数	+	添加默认点文件名的显示
	RobotModel$函数	-	
	RobotName$函数	-	
	RobotSerial$函数	-	
	RobotType函数	-	
	RSet$函数	-	
	RShift函数	-	
	RTrim$函数	-	
S	SafetyOn函数	-	
	SavePoints	-	
	Select...Send	-	
	Sense	-	
	SetCom	-	
	SetInW	-	
	SetIn	-	
	SetNet	-	
	SetSw	-	
	SFree	-	
	SFree函数	-	
	Sgn函数	-	
	Signal	-	
	Sin函数	-	
	SLock	-	

命令		兼容性	备注
	SoftCP	-	
	SoftCP函数	-	
	Space$函数	-	
	Speed	-	
	SpeedR	-	
	SpeedR函数	-	
	SpeedS	-	
	SpeedS函数	-	
	Speed函数	-	
	SPELCom_Event	-	
	Sqr函数	-	
	Stat函数	+	添加信息
	Str$函数	-	
	String	-	
	Sw函数	-	
	SyncLock	-	
	SyncUnlock	-	
	SysConfig	+	添加信息
	SysErr函数	+	添加警告获取功能
T	Tab$函数	-	
	Tan函数	-	
	TargetOK函数	-	
	TaskDone函数	-	
	TaskInfo函数	-	
	TaskInfo$函数	-	
	TaskState	+	添加后台任务的显示
	TaskState函数	-	
	TaskWait	-	
	TC	-	
	TCLim	-	
	TCLim函数	-	
	TCSpeed	-	
	TCSpeed函数	-	
	TGo	+	添加动作模式指定

命令		兼容性	备注
	TillOn函数	-	
	Time	!	仅显示
	Time函数	-	
	Time$函数	-	
	TLClr	-	
	TlDef函数	-	
	TLSet	-	
	TLSet函数	-	
	TMOut	-	
	TMove	-	
	TmReset	-	
	Tmr函数	-	
	Toff	-	
	Ton	-	
	Tool	-	
	Tool函数	-	
	Trap	!	添加将控制器状态设为触发的Trap
	Trim$函数	-	
	Tw函数	-	
U	UBound函数	-	
	UCase$函数	-	
V	Val函数	-	
W	Wait	!	在等待条件中添加全局变量等
	WaitNet	-	
	WaitPos	-	
	WaitSig	-	
	Weight	+	添加S、T的指定
	Weight函数	+	添加S、T的指定
	Where	-	
	Wrist	-	
	Wrist函数	-	
	Write	-	
	WriteBin	-	
X	Xor	-	

命令		兼容性	备注
	Xqt	-	
	XY函数	-	
	XYLim	-	
	XYLim函数	-	
	XYLimClr	-	
	XYLimDef	-	
	XYLimDef函数	-	

5.2.4 沿用Epson RC+ Ver.4.*的命令 (Epson RC+5.0不支持)

AOpen	BOpen	Calib
CalPls	ChDir	ChDrive
Close	Cnv_AbortTrack	Cnv_Downstream
Cnv_Fine	Cnv_Fine函数	Cnv_Name$函数
Cnv_Number函数	Cnv_Point函数	Cnv_PosErr函数
Cnv_Pulse函数	Cnv_QueAdd	Cnv_QueGet函数
Cnv_QueLen函数	Cnv_QueList	Cnv_QueMove
Cnv_QueReject	Cnv_QueReject函数	Cnv_QueRemove
Cnv_QueUserData	Cnv_QueUserData函数	Cnv_RobotConveyor函数
Cnv_Speed函数	Cnv_Trigger	Cnv_Upstream函数
Cont	Copy	CurDir$函数
CurDrive$函数	Declare	Del
Eof函数	Eval函数	FbusIO_GetBusStatus函数
FbusIO_GetDeviceStatus函数	FbusIO_SendMsg	FileDateTime$函数
FileExists函数	FileLen函数	FolderExists函数
FreeFile函数	GetCurrentUser$	Hofs
Hofs函数	ImportPoints	InputBox
LogIn函数	MCalComplete函数	MCal
MCordr	MCordr函数	MKDir
MsgBox	Recover函数	Rename
RenDir	Restart	RmDir
Robot	Robot函数	ROpen
RunDialog	Seek	Shutdown
UOpen	WOpen	

6. Appendix C：Epson RC+8.0的命令

6.1 C-1：EPSON RC+ 4.0或更高版本中添加的命令

A		
AbortMotion	AccelMax函数	AglToPls函数
AIO_Out	AIO_Out函数	AIO_OutW
AIO_OutW函数	AIO_Set	AIO_Set函数
AIO_TrackingSet	AIO_TrackingStart	AIO_TrackingEnd
AIO_TrackingOn函数	AIO_In函数	AIO_InW函数
Align函数	AlignECP函数	AreaCorrection函数
AreaCorrectionClr	AreaCorrectionDef函数	AreaCorrectionInv函数
AreaCorrectionOffset函数	AreaCorrectionSet	ArmCalib
ArmCalib函数	ArmCalibSet	ArmCalibSet函数
ArmCalibClr	ArmCalibDef函数	ArmDef函数
ATCLR	AtHome函数	ATRQ
ATRQ函数	AutoLJM	AutoLJM函数
AvoidSingularity	AvoidSingularity函数	

B		
BClr函数	BClr64函数	Box
Box函数	BoxClr函数	BoxDef函数
Brake函数	BSet函数	BSet64函数
BTst函数	BTst64函数	

C		
ChDisk	ChkCom函数	ChkNet函数
CloseCom	CloseDB	CloseNet
Cls	CP	CP函数
CP_Offset	CP_Offset函数	CR
CR函数	CS	CS函数
CT	CT函数	CtrlDev函数
Curve	CVMove	Cnv_Accel
Cnv_Accel函数	Cnv_AccelLim	Cnv_AccelLim函数
Cnv_Adjust	Cnv_AdjustClear	Cnv_AdjustGet
Cnv_AdjustSet	Cnv_DownStream	Cnv_Mode
Cnv_Mode函数	Cnv_OffsetAngle	Cnv_OffsetAngle函数
Cnv_PosErrOffset	Cnv_Upstream	CollisionDetect

CollisionDetect函数

D		
DegToRad函数	DeleteDB	DiffPoint函数
DispDev	DispDev函数	Dist函数

E		
EcpDef函数	ElapsedTime函数	EResume
Errb函数	ErrorOn函数	Error
EStopOn函数	Exit	ExportPoints

F		
FindPos函数	Find	FineDist
FineDist函数	FineStatus函数Fix函数	Flush
Fmtstr		

G	
GetRobotInsideBox函数	GetRobotInsidePlane函数

H		
Hand_On	Hand_On函数	Hand_Off
Hand_Off函数	Hand_TW函数	Hand_Def函数
Hand_Type函数	Hand_Label$函数	Hand_Number函数
HealthCalcPeriod	HealthCalcPeriod函数	HealthCtrlAlarmOn函数
HealthCtrlInfo	HealthCtrlInfo函数	HealthCtrlRateOffset
HealthCtrlReset	HealthCtrlWarningEnable	HealthCtrlWarningEnable函数
HealthRateCtrlInfo函数	HealthRateRBInfo函数	HealthRBAlarmOn函数
HealthRBAnalysis	HealthRBAnalysis函数	HealthRBDistance
HealthRBDistance函数	HealthRBInfo	HealthRBInfo函数
HealthRBRateOffset	HealthRBReset	HealthRBSpeed
HealthRBSpeed函数	HealthRBStart	HealthRBStop
HealthRBTRQ	HealthRBTRQ函数	HealthRBWarningEnable
HealthRBWarningEnable函数	Here	Here函数
Hex$函数	HofsJointAccuracy	HomeClr
HomeDef函数		

I		
InReal函数	InsideBox函数	InsidePlane函数

I		
InStr函数	IODef函数	IOLabel$函数
IONumber函数		

J		
J1Angle	J1Angle函数	J4Angle
JA函数	Joint	JointAccuracy
JointAccuracy函数	JumpTLZ	JTran

L		
LatchEnable	LatchState函数	LatchPos函数
LimZMargin	LimZMargin函数	LimitTorque
LimitTorque函数	LimitTorqueLP	LimitTorqueLP函数
LimitTorqueStop	LimitTorqueStop函数	LimitTorqueStopLP
LimitTorqueStopLP函数	LJM函数	LocalDef函数
LShift64函数		

M		
MemInW函数	MemOutW	MHour函数

O		
OLAccel	OLAccel函数	OpenCom
OpenCom函数	OpenDB	OpenNet
OpenNet函数	OutReal	OutReal函数

P		
P#	PalletClr	PauseOn函数
PDef函数	PDel	PDescription
PDescription函数	PerformMode	PerformMode函数
PG_FastStop	PG_LSpeed	PG_LSpeed函数
PG_Scan	PG_SlowStop	PLabel
PLabel$函数	PlaneClr	PlaneDef
Plane	Plane函数	PList
PLocal	PLocal函数	PNumber函数
PosFound函数	PTCLR	PTPBoostOK函数
PTPTime函数	PTran	PTRQ
PTRQ函数		

Q		
QPDECELR	QPDECELR函数	QPDECELS
QPDECELS函数		

R		
RadToDeg函数	Randomize	ReadBin
Read	RealAccel函数	RealPls函数
RealPos函数	RealTorque函数	RecoverPos函数
Recover	Redim	ResetElapsedTime
Rnd函数	RobotInfo函数	RobotInfo$函数
RobotModel$函数	RobotName$函数	RobotSerial$函数
RobotType函数	ROTOK函数	RShift64函数

S		
SafetyOn函数	SelectDB	SetCom
SetInW	SetIn	SetNet
SetSw	SF_GetParam函数	SF_GetParam$函数
SF_GetStatus函数	SF_LimitSpeedS	SF_LimitSpeedS函数
SF_LimitSpeedEnable	SF_LimitSpeedEnable函数	SF_PeakSpeedS
SF_PeakSpeedS函数	SF_PeakSpeedSClear	SF_RealSpeedS
SF_RealSpeedS函数	Shutdown函数	SimGet
SimSet	SingularityAngle	SingularityAngle函数
SingularityDist	SingularityDist函数	SingularitySpeed
SingularitySpeed函数	SoftCP	SoftCP函数
SpeedFactor	SpeedFactor函数	StartMain
SyncRobots	SyncRobots函数	SysErr函数

T		
Tab$函数	TargetOK函数	TaskDone函数
TaskInfo函数	TaskInfo$函数	TaskState
TaskState函数	TaskWait	TC
TCLim	TCLim函数	TCSpeed
TCSpeed函数	TeachOn函数	TillOn函数
TlDef函数	Toff	Ton

U	
UBound函数	UpdateDB

V		
VDefArm	VDefLocal	VDefSetMotionRange
VDefGetMotionRange	VDefTool	VGoCenter
VSD	VSD函数	VxCalib
VxCalDelete	VxCalLoad	VxCalInfo函数
VxCalSave		

W		
WaitNet	WaitPos	Where
WindosStatus函数	WorkQue_Add	WorkQue_AutoRemove
WorkQue_AutoRemove函数	WorkQue_Get函数	WorkQue_Len函数
WorkQue_List	WorkQue_Reject	WorkQue_Reject函数
WorkQue_Remove	WorkQue_Sort	WorkQue_Sort函数
WorkQue_UserData	WorkQue_UserData函数	WriteBin
Write		

X		
XYLimClr	XYLimDef	XY函数

6.2 C-2：Epson RC+ 8.0各版本中添加的命令

Epson RC+ 8.0版本号	新增命令	
Ver.8.0.0	ClearHistory	History

6.3 C-3：EPSON RC+ 7.0各版本中添加的命令

EPSON RC+ 6.0, 5.0, 4.0 通用

EPSON RC+ 7.0版本号	新增命令
Ver.7.5.4	Cnv_AccelLim, Cnv_AccelLim函数 SF_GetParam函数, SF_GetParam$函数 SF_GetStatus函数, SF_LimitSpeedS SF_LimitSpeedS函数, SF_LimitSpeedEnable SF_LimitSpeedEnable函数, SF_PeakSpeedS SF_PeakSpeedS函数, SF_PeakSpeedSClear SF_RealSpeedS, SF_RealSpeedS函数
Ver.7.5.3	AreaCorrection函数, AreaCorrectionClr AreaCorrectionDef函数, AreaCorrectionInv函数 AreaCorrectionOffset函数, AreaCorrectionSet Cnv_PosErrOffset
Ver.7.5.2	XYLimMode, XYLimMode函数
Ver.7.5.1	ArmCalib, ArmCalib函数 ArmCalibSet, ArmCalibSet函数 ArmCalibClr, ArmCalibDef函数 JointAccuracy, JointAccuracy函数 HofsJointAccuracy, Hand_On Hand_On函数, Hand_Off Hand_Off函数, Hand_TW函数 Hand_Def函数, Hand_Type函数 Hand_Label$函数, Hand_Number函数 Cnv_Adjust, Cnv_AdjustClear Cnv_AdjustGet, Cnv_AdjustSet DiffPoint函数, ROTOK函数
Ver.7.4.3	AIO_TrackingSet, AIO_TrackingStart AIO_TrackingEnd, AIO_TrackingOn函数
Ver.7.4.1	AutoOrientationFlag, AutoOrientationFlag函数
Ver.7.3.4	SimGet, SimSet
Ver.7.3.3	HealthCtrlWarningEnable, HealthCtrlWarningEnable函数 HealthRBWarningEnable, HealthRBWarningEnable函数
Ver.7.3.2	PDescription, PDescription函数
Ver.7.3.1	AIO_Out, AIO_Out函数 AIO_OutW, AIO_OutW函数 AIO_Set, AIO_Set函数AIO_In函数 AIO_InW函数, HealthCalcPeriod HealthCalcPeriod函数
Ver.7.3.0	VDefTool, VDefArm VDefLocal, VGoCenter VDefSetMotionRange, VDefGetMotionRange

EPSON RC+ 7.0版本号	新增命令
Ver.7.2.0	CP_Offset, CP_Offset函数 HealthCtrlAlarmOn函数, HealthCtrlInfo HealthCtrlInfo函数, HealthCtrlRateOffset HealthCtrlReset, HealthRateCtrlInfo函数 HealthRateRBInfo函数, HealthRBAlarmOn函数 HealthRBAnalysis, HealthRBAnalysis函数 HealthRBDistance, HealthRBDistance函数 HealthRBInfo, HealthRBInfo函数 HealthRBRateOffset, HealthRBReset HealthRBSpeed HealthRBSpeed函数, HealthRBStart HealthRBStop, HealthRBTRQ HealthRBTRQ函数, J4Angle JumpTLZ, LimitTorqueLP LimitTorqueLP函数, LimitTorqueStop LimitTorqueStop函数, LimitTorqueStopLP LimitTorqueStopLP函数, VSD VSD函数
Ver.7.1.4	CollisionDetect函数
Ver.7.1.3	CollisionDetect, MHour函数
Ver.7.1.2	SingularityDist, SingularityDist函数 ExportPoints
Ver.7.1.0	BClr64函数, BSet64函数 BTst64函数, FineDist FineDist函数, FineStatus函数 Fmtstr, IODef函数 LShift64函数, RealAccel函数 RShift64函数, WorkQue_Add WorkQue_AutoRemove, WorkQue_AutoRemove函数 WorkQue_Get函数, WorkQue_Len函数 WorkQue_List, WorkQue_Reject WorkQue_Reject函数, WorkQue_Remove WorkQue_Sort, WorkQue_Sort函数 WorkQue_UserData, WorkQue_UserData函数
Ver.7.0.3	PerformMode, PerformMode函数

要点

Epson RC+7.0 Ver.7.0.0,

添加的命令与Epson RC+ 6.0, 5.0, 4.0部分不同。

Epson RC+ 6.0, 5.0添加的命令

Epson RC+ 7.0版本号	Epson RC+ 6.0	Epson RC+ 5.0
Ver.7.0.0	AutoLJM AutoLJM函数 AvoidSingularity AvoidSingularity函数	AbortMotion AutoLJM AutoLJM函数 AvoidSingularity AvoidSingularity函数
	Cnv_Accel Cnv_Accel函数 Cnv_DownStream Cnv_Mode Cnv_Mode函数 Cnv_Upstream	ChDisk CloseDB Cnv_Accel Cnv_Accel函数 Cnv_DownStream Cnv_Mode Cnv_Mode函数 Cnv_Upstream
		CR CR函数 CS CS函数 CT CT函数
	DeleteDB	DeleteDB
	ElapsedTime函数 Errb函数	Errb函数 Flush GetRobotInsideBox函数 GetRobotInsidePlane函数 J1Angle J1Angle函数
	LimZMargin LimZMargin函数 LimitTorque LimitTorque函数	LimZMargin LimZMargin函数 LimitTorque LimitTorque函数 OpenDB
	PalletClr	PalletClr PG_FastStop PG_LSpeed PG_LSpeed函数 PG_Scan PG_SlowStop QPDECELR QPDECELR函数 QPDECELS QPDECELS函数 RecoverPos函数 Recover
	ResetElapsedTime	SelectDB Shutdown函数

Epson RC+ 7.0版本号	Epson RC+ 6.0	Epson RC+ 5.0
	SingularityAngle SingularityAngle函数 SingularitySpeed SingularitySpeed函数 SpeedFactor SpeedFactor函数	SingularityAngle SingularityAngle函数 SingularitySpeed SingularitySpeed函数 SpeedFactor SpeedFactor函数
		StartMain SyncRobots SyncRobots函数 TeachOn函数
	UpdateDB	UpdateDB WindosStatus函数

Epson RC+ 4.0添加的命令

Epson RC+ 7.0版本号	Epson RC+ 4.0
Ver.7.0.0	AbortMotion, AccelMax函数 AglToPls函数, Align函数 AlignECP函数, ArmDef函数 ATCLR, AtHome函数 ATRQ, ATRQ函数 AutoLJM, AutoLJM函数 AvoidSingularity, AvoidSingularity函数
	BClr函数, Box Box函数, BoxClr函数 BoxDef函数, Brake函数 BSet函数, BTst 函数
	ChDisk, ChkCom函数 ChkNet函数, CloseCom CloseDB, CloseNet Cls, CP CP函数, CR CR函数, CS CS函数, CT CT函数, CtrlDev函数 Curve, CVMove Cnv_Accel, Cnv_Accel函数Cnv_DownStream Cnv_Mode Cnv_Mode函数, Cnv_OffsetAngle Cnv_OffsetAngle函数, Cnv_Upstream
	DegToRad函数, DeleteDB DispDev, DispDev函数 Dist函数
	EcpDef函数, EResume Errb函数, ErrorOn函数 Error, EStopOn函数 Exit
	FindPos函数, Find FineStatus函数, Fix函数 Flush
	GetRobotInsideBox函数, GetRobotInsidePlane函数
	Here, Here函数 Hex$函数, HomeClr HomeDef函数
	InReal函数, InsideBox函数 InsidePlane函数, InStr函数 IOLabel$函数, IONumber函数
	J1Angle, J1Angle函数 JA函数, Joint JTran

Epson RC+ 7.0版本号	Epson RC+ 4.0
	LatchEnable, LatchState函数 LatchPos函数, LimZMargin LimZMargin函数, LimitTorque LimitTorque函数, LJM函数 LocalDef函数
	MemInW函数, MemOutW
	OLAccel, OLAccel函数 OpenCom, OpenCom函数 OpenDB, OpenNet OpenNet函数, OutReal OutReal函数
	P#, PalletClr PauseOn函数, PDef函数 PDel, PG_FastStop PG_LSpeed, PG_LSpeed函数 PG_Scan, PG_SlowStop PLabel, PLabel$函数 PlaneClr, PlaneDef Plane, Plane函数 PList, PLocal PLocal函数, PNumber函数 PosFound函数, PTCLR PTPBoostOK函数, PTPTime函数 PTran, PTRQ PTRQ函数
	QPDECELR, QPDECELR函数 QPDECELS, QPDECELS函数
	RadToDeg函数, Randomize ReadBin, Read RealPls函数, RealPos函数 RealTorque函数, RecoverPos函数 Recover, Redim RobotInfo函数, RobotInfo$ 函数RobotModel$函数, RobotName$函数 RobotSerial$函数, RobotType函数
	SafetyOn函数, SelectDB SetCom, SetInW SetIn, SetNet SetSw, Shutdown函数 SingularityAngle, SingularityAngle函数 SingularitySpeed, SingularitySpeed函数 SoftCP, SoftCP函数 SpeedFactor, SpeedFactor函数 StartMain, SyncRobots SyncRobots函数, SysErr函数

Epson RC+ 7.0版本号	Epson RC+ 4.0
	Tab$函数, TargetOK函数 TaskDone函数, TaskInfo函数 TaskInfo$函数, TaskState TaskState函数, TaskWait TC, TCLim TCLim函数, TCSpeed TCSpeed函数, TeachOn函数 TillOn函数, TlDef函数 Toff, Ton
	UBound函数, UpdateDB
	VxCalib, VxCalDelete VxCalLoad, VxCalInfo函数 VxCalSave, VxTrans函数
	WaitNet, WaitPos Where, WindosStatus函数 WriteBin, Write
	XYLimClr, XYLimDef XY函数

6.4 C-4：EPSON RC+ 7.0各版本中删除的命令

删除了EPSON RC+ 6.0, 5.0, 4.0中以下命令

EPSON RC+ 7.0版本号	EPSON RC+ 6.0	EPSON RC+ 5.0	EPSON RC+ 4.0
Ver.7.1.2	SetLCD	SetLCD	SetLCD
Ver.7.0.0	Dir Type	-	Dir Type

7. Appendix D: 常数

7.1 Appendix D: 常数

SPEL+程序中有可以使用的常数。构建项目时使用常数值。

常数名称	值	使用方法
TRUE	-1	逻辑表达式
FALSE	0	逻辑表达式
High	1	
Low	0	
Off	0	
On	1	
Above	1	
Below	2	
NoFlip	1	
Flip	2	
Righty	1	
Lefty	2	
J1	1	
J2	2	
J3	4	
J4	8	
J5	16	
J6	32	
J7	64	
CR	CHR$(13)	
CRLF	CHR$(13)+CHR$(10)	
LF	CHR$(10)	
MB_OK	0	MsgBox标记
MB_OKCANCEL	1	MsgBox标记
MB_ABORTRETRYIGNORE	2	MsgBox标记
MB_YESNOCANCEL	3	MsgBox标记
MB_YESNO	4	MsgBox标记
MB_RETRYCANCEL	5	MsgBox标记
MB_ICONSTOP	16	MsgBox标记
MB_ICONQUESTION	32	MsgBox标记
MB_ICONEXCLAMATION	48	MsgBox标记

常数名称	值	使用方法
MB_ICONINFORMATION	64	MsgBox标记
MB_DEFBUTTON1	0	MsgBox标记
MB_DEFBUTTON2	256	MsgBox标记
IDOK	1	MsgBox返回值
IDCANCEL	2	MsgBox返回值
IDABORT	3	MsgBox返回值
IDRETRY	4	MsgBox返回值
IDIGNORE	5	MsgBox返回值
IDYES	6	MsgBox返回值
IDNO	7	MsgBox返回值
BACKCOLORMODE_VISUALSTYLE	0	用于GUI Builder
BACKCOLORMODE_USER	1	用于GUI Builder
BORDERSTYLE_NONE	0	用于GUI Builder
BORDERSTYLE_FIXEDSINGLE	1	用于GUI Builder
BORDERSTYLE_FIXED3D	2	用于GUI Builder
CNV_QUELEN_ALL	0	Cnv_QueLen
CNV_QUELEN_UPSTREAM	1	Cnv_QueLen
CNV_QUELEN_PICKUPAREA	2	Cnv_QueLen
CNV_QUELEN_DOWNSTREAM	3	Cnv_QueLen
DEVID_SELF	21	CLS
DEVID_TP	24	CLS
DEVID_TP3	30	CLS
DIALOGRESULT_NOE	0	用于GUI Builder
DIALOGRESULT_OK	1	用于GUI Builder
DIALOGRESULT_CANCEL	2	用于GUI Builder
DLG_IOMON	102	RunDialog
DLG_ROBOTMNG	100	RunDialog
DLG_VGUIDE	110	RunDialog
DOCK_NONE	0	用于GUI Builder
DOCK_TOP	1	用于GUI Builder
DOCK_BOTTOM	2	用于GUI Builder
DOCK_LEFT	3	用于GUI Builder
DOCK_RIGHT	4	用于GUI Builder
DOCK_FILL	5	用于GUI Builder

常数名称	值	使用方法
DROPDOWNSTYLE_SIMPLE	0	用于GUI Builder
DROPDOWNSTYLE_DROPDOWN	1	用于GUI Builder
DROPDOWNSTYLE_DROPDOWNLIST	2	用于GUI Builder
ERROR_DOINGMOTION	2999	用于GUI Builder
ERROR_NOMOTION	2998	用于GUI Builder
EVENTTASKTYPE_NORMAL	0	用于GUI Builder
EVENTTASKTYPE_NOPAUSE	1	用于GUI Builder
EVENTTASKTYPE_NOEMGABORT	2	用于GUI Builder
FORMBORDERSTYLE_NONE	0	用于GUI Builder
FORMBORDERSTYLE_FIXEDSINGLE	1	用于GUI Builder
FORMBORDERSTYLE_FIXED3D	2	用于GUI Builder
FORMBORDERSTYLE_FIXEDDIALOG	3	用于GUI Builder
FORMBORDERSTYLE_SIZABLE	4	用于GUI Builder
IMAGEALIGN_TOPLEFT	1	用于GUI Builder
IMAGEALIGN_TOPCENTER	2	用于GUI Builder
IMAGEALIGN_TOPRIGHT	3	用于GUI Builder
IMAGEALIGN_MIDDLELEFT	4	用于GUI Builder
IMAGEALIGN_MIDDLECENTER	5	用于GUI Builder
IMAGEALIGN_MIDDLERIGHT	6	用于GUI Builder
IMAGEALIGN_BOTTOMLEFT	7	用于GUI Builder
IMAGEALIGN_BOTTOMCENTER	8	用于GUI Builder
IMAGEALIGN_BOTTOMRIGHT	9	用于GUI Builder
IOTYPE_INPUT	0	IOLabel函数
IOTYPE_OUTPUT	1	IOLabel函数
IOTYPE_MEMORY	2	IOLabel函数
IOSIZE_BIT	1	IOLabel函数
IOSIZE_BYTE	8	IOLabel函数
IOSIZE_WORD	16	IOLabel函数
LANGID_ENGLISH	0	ErrMsg$
LANGID_JAPANESE	1	ErrMsg$
LANGID_GERMAN	2	ErrMsg$
LANGID_FRENCH	3	ErrMsg$
LANGID_SIMPLIFIED_CHINESE	4	ErrMsg$
LANGID_TRADITIONAL_CHINESE	5	ErrMsg$

常数名称	值	使用方法
MODE_STANDARD	1	PerformMode
MODE_HIGH_SPEED	2	PerformMode
MODE_LOW_OSCILLATION	3	PerformMode
ORIENT_HORIZONTAL	0	用于GUI Builder
ORIENT_VERTICAL	1	用于GUI Builder
PROGRESSBAR_STYLE_BLOCKS	0	用于GUI Builder
PROGRESSBAR_STYLE_CONT	1	用于GUI Builder
PROGRESSBAR_STYLE_MARQUEE	2	用于GUI Builder
SCROLLBARS_NONE	0	用于GUI Builder
SCROLLBARS_HORIZ	1	用于GUI Builder
SCROLLBARS_VERT	2	用于GUI Builder
SCROLLBARS_BOTH	3	用于GUI Builder
SETLATCH_PORT_CU_0	24	SetLatch
SETLATCH_PORT_CU_1	25	SetLatch
SETLATCH_PORT_DU1_0	56	SetLatch
SETLATCH_PORT_DU1_1	57	SetLatch
SETLATCH_PORT_DU2_0	280	SetLatch
SETLATCH_PORT_DU2_1	281	SetLatch
SETLATCH_TRIGGERMODE_LEADINGEDGE	1	SetLatch
SETLATCH_TRIGGERMODE_TRAILINGEDGE	0	SetLatch
SHUTDOWN_ALL	0	Shutdown
SHUTDOWN_RESTART	1	Shutdown
SHUTDOWN_EPSONRC	2	Shutdown
SING_NONE	0	AvoidSingularity
SING_THRU	1	AvoidSingularity
SING_THRUROT	2	AvoidSingularity
SING_VSD	3	AvoidSingularity
SING_AUTO	4	AvoidSingularity
SIZEMODE_NORMAL	0	用于GUI Builder
SIZEMODE_STRETCHIMAGE	1	用于GUI Builder
SIZEMODE_AUTOSIZE	2	用于GUI Builder
SIZEMODE_CENTERIMAGE	3	用于GUI Builder
SIZEMODE_ZOOM	4	用于GUI Builder
STARTPOSITION_MANUAL	0	用于GUI Builder

常数名称	值	使用方法
STARTPOSITION_CENTERSCREEN	1	用于GUI Builder
STARTPOSITION_CENTERPARENT	2	用于GUI Builder
TEXTALIGN_LEFT	1	用于GUI Builder
TEXTALIGN_CENTER	2	用于GUI Builder
TEXTALIGN_RIGHT	3	用于GUI Builder
TEXTALIGN_TOPLEFT	1	用于GUI Builder
TEXTALIGN_TOPCENTER	2	用于GUI Builder
TEXTALIGN_TOPRIGHT	3	用于GUI Builder
TEXTALIGN_MIDDLELEFT	4	用于GUI Builder
TEXTALIGN_MIDDLECENTER	5	用于GUI Builder
TEXTALIGN_MIDDLERIGHT	6	用于GUI Builder
TEXTALIGN_BOTTOMLEFT	7	用于GUI Builder
TEXTALIGN_BOTTOMCENTER	8	用于GUI Builder
TEXTALIGN_BOTTOMRIGHT	9	用于GUI Builder
TICKSTYLE_NONE	0	用于GUI Builder
TICKSTYLE_TOPLEFT	1	用于GUI Builder
TICKSTYLE_BOTTOMRIGHT	2	用于GUI Builder
TICKSTYLE_BOTH	3	用于GUI Builder
VISION_SORT_NONE	0	用于Vision Guide
VISION_SORT_PIXELX	1	用于Vision Guide
VISION_SORT_PIXELY	2	用于Vision Guide
VISION_SORT_PIXELXY	3	用于Vision Guide
VISION_SORT_CAMERAX	4	用于Vision Guide
VISION_SORT_CAMERAY	5	用于Vision Guide
VISION_SORT_CAMERAXY	6	用于Vision Guide
VISION_SORT_ROBOTX	7	用于Vision Guide
VISION_SORT_ROBOTY	8	用于Vision Guide
VISION_SORT_ROBOTXY	9	用于Vision Guide
VISION_SIZETOFIND_ANY	0	用于Vision Guide
VISION_SIZETOFIND_LARGEST	1	用于Vision Guide
VISION_SIZETOFIND_SMALLEST	2	用于Vision Guide
VISION_BACKCOLOR_NONE	0	用于Vision Guide
VISION_BACKCOLOR_BLACK	1	用于Vision Guide
VISION_BACKCOLOR_WHITE	2	用于Vision Guide

常数名称	值	使用方法
VISION_CAMORIENT_STANDALONE	1	用于Vision Guide
VISION_CAMORIENT_FIXEDDOWN	2	用于Vision Guide
VISION_CAMORIENT_FIXEDUP	3	用于Vision Guide
VISION_CAMORIENT_MOBILEJ2	4	用于Vision Guide
VISION_CAMORIENT_MOBILEJ4	5	用于Vision Guide
VISION_CAMORIENT_MOBILEJ5	6	用于Vision Guide
VISION_CAMORIENT_MOBILEJ6	7	用于Vision Guide
VISION_FOUNDCOLOR_LIGHTGREEN	1	用于Vision Guide
VISION_FOUNDCOLOR_DARKGREEN	2	用于Vision Guide
VISION_GRAPHICS_ALL	1	用于Vision Guide
VISION_GRAPHICS_POSONLY	2	用于Vision Guide
VISION_GRAPHICS_NONE	3	用于Vision Guide
VISION_OPERATION_OPEN	1	用于Vision Guide
VISION_OPERATION_CLOSE	2	用于Vision Guide
VISION_OPERATION_ERODE	3	用于Vision Guide
VISION_OPERATION_DILATE	4	用于Vision Guide
VISION_OPERATION_SMOOTH	5	用于Vision Guide
VISION_OPERATION_SHARPEN1	6	用于Vision Guide
VISION_OPERATION_SHARPEN2	7	用于Vision Guide
VISION_OPERATION_HORIZEDGE	8	用于Vision Guide
VISION_OPERATION_VERTEDGE	9	用于Vision Guide
VISION_OPERATION_EDGEDETECT1	10	用于Vision Guide
VISION_OPERATION_EDGEDETECT2	11	用于Vision Guide
VISION_OPERATION_LAPLACE1	12	用于Vision Guide
VISION_OPERATION_LAPLACE2	13	用于Vision Guide
VISION_OPERATION_THIN	14	用于Vision Guide
VISION_OPERATION_THICKEN	15	用于Vision Guide
VISION_OPERATION_BINARIZE	16	用于Vision Guide
VISION_OPERATION_ROTATE	17	用于Vision Guide
VISION_OPERATION_FLIPHORIZ	18	用于Vision Guide
VISION_OPERATION_FLIPVERT	19	用于Vision Guide
VISION_OPERATION_FLIPBOTH	20	用于Vision Guide
VISION_OPERATION_COLORFILTER	21	用于Vision Guide
VISION_OPERATION_SUBTRACTABS	22	用于Vision Guide

常数名称	值	使用方法
VISION_OPERATION_ZOOM	23	用于Vision Guide
VISION_ACQUIRE_NONE	0	用于Vision Guide
VISION_ACQUIRE_STATIONARY	1	用于Vision Guide
VISION_ACQUIRE_STROBED	2	用于Vision Guide
VISION_TRIGGERMODE_LEADINGEDGE	1	用于Vision Guide
VISION_TRIGGERMODE_TRAILINGEDGE	2	用于Vision Guide
VISION_THRESHCOLOR_BLACK	1	用于Vision Guide
VISION_THRESHCOLOR_WHITE	2	用于Vision Guide
VISION_OBJTYPE_CORRELATIO	1	用于Vision Guide
VISION_OBJTYPE_BLOB	2	用于Vision Guide
VISION_OBJTYPE_EDGE	3	用于Vision Guide
VISION_OBJTYPE_POLAR	4	用于Vision Guide
VISION_OBJTYPE_LINE	5	用于Vision Guide
VISION_OBJTYPE_POINT	6	用于Vision Guide
VISION_OBJTYPE_FRAME	7	用于Vision Guide
VISION_OBJTYPE_IMAGEOP	8	用于Vision Guide
VISION_OBJTYPE_OCR	9	用于Vision Guide
VISION_OBJTYPE_CODEREADER	10	用于Vision Guide
VISION_OBJTYPE_GEOMETRIC	11	用于Vision Guide
VISION_DETAILLEVEL_MEDIUM	1	用于Vision Guide
VISION_DETAILLEVEL_HIGH	2	用于Vision Guide
VISION_DETAILLEVEL_VERYHIGH	3	用于Vision Guide
VISION_IMAGESOURCE_CAMERA	1	用于Vision Guide
VISION_IMAGESOURCE_FILE	2	用于Vision Guide
VISION_CODETYPE_AUTO	0	用于Vision Guide
VISION_CODETYPE_EAN13	2	用于Vision Guide
VISION_CODETYPE_CODE39	3	用于Vision Guide
VISION_CODETYPE_INTERLEAVED25	4	用于Vision Guide
VISION_CODETYPE_CODE128	5	用于Vision Guide
VISION_CODETYPE_CODABAR	6	用于Vision Guide
VISION_CODETYPE_PDF417	8	用于Vision Guide
VISION_CODETYPE_QR	10	用于Vision Guide
VISION_CODETYPE_EAN8	13	用于Vision Guide
VISION_CODETYPE_UPCA	18	用于Vision Guide

常数名称	值	使用方法
VISION_CODETYPE_UPCE	19	用于Vision Guide
VISION_CODETYPE_UPC	20	用于Vision Guide
VISION_EDGETYPE_SINGLE	1	用于Vision Guide
VISION_EDGETYPE_PAIR	2	用于Vision Guide
VISION_IMAGECOLOR_ALL	1	用于Vision Guide
VISION_IMAGECOLOR_RED	2	用于Vision Guide
VISION_IMAGECOLOR_GREEN	3	用于Vision Guide
VISION_IMAGECOLOR_BLUE	4	用于Vision Guide
VISION_IMAGECOLOR_GRAYSCALE	5	用于Vision Guide
VISION_POINTTYPE_POINT	0	用于Vision Guide
VISION_POINTTYPE_ENDPOINT	1	用于Vision Guide
VISION_POINTTYPE_MIDPOINT	2	用于Vision Guide
VISION_POINTTYPE_PERPTOLINE	3	用于Vision Guide
VISION_POINTTYPE_STARTPOINT	4	用于Vision Guide
VISION_POINTTYPE_PERPTOSTARTPOINT	5	用于Vision Guide
VISION_POINTTYPE_PERPTOMIDPOINT	6	用于Vision Guide
VISION_POINTTYPE_PERPTOENDPOINT	7	用于Vision Guide
VISION_REFTYPE_TAUGHTPOINTS	1	用于Vision Guide
VISION_REFTYPE_UPWARDCAMERA	2	用于Vision Guide
VISION_IMAGESIZE_320X240	1	用于Vision Guide
VISION_IMAGESIZE_640X480	2	用于Vision Guide
VISION_IMAGESIZE_800X600	3	用于Vision Guide
VISION_IMAGESIZE_1024X768	4	用于Vision Guide
VISION_IMAGESIZE_1280X1024	5	用于Vision Guide
VISION_IMAGESIZE_1600X1200	6	用于Vision Guide
VISION_IMAGESIZE_2048X1536	7	用于Vision Guide
VISION_IMAGESIZE_2560X1920	8	用于Vision Guide
VISION_WINTYPE_RECTANGLE	1	用于Vision Guide
VISION_WINTYPE_ROTATEDRECT	2	用于Vision Guide
VISION_WINTYPE_CIRCLE	3	用于Vision Guide
VISION_ORIENT_BOTH	1	用于Vision Guide
VISION_ORIENT_HORIZ	2	用于Vision Guide
VISION_ORIENT_VERT	3	用于Vision Guide
VISION_DIRECTION_INSIDEOUT	1	用于Vision Guide

常数名称	值	使用方法
VISION_DIRECTION_OUTSIDEIN	2	用于Vision Guide
VISION_POLARITY_DARK	1	用于Vision Guide
VISION_POLARITY_LIGHT	2	用于Vision Guide
VISION_PASSTYPE_SOMEFOUND	1	用于Vision Guide
VISION_PASSTYPE_ALLFOUND	2	用于Vision Guide
VISION_PASSTYPE_SOMENOTFOUND	3	用于Vision Guide
VISION_PASSTYPE_ALLNOTFOUND	4	用于Vision Guide
WIN_IOMON	-1	用于GUI Builder
WIN_TASKMGR	-2	用于GUI Builder
WIN_FORCEMON	-3	用于GUI Builder
WIN_SIMULATOR	-4	用于GUI Builder
WINDOWSTATE_NORMAL	0	WindowsStatus
WINDOWSTATE_MINIMIZED	1	WindowsStatus
WINDOWSTATE_MAXIMIZED	2	WindowsStatus
WithMove	0	Recover
WithoutMove	1	Recover
DRYRUNOFF	1	SF_GetParam函数
SLS_1_HAND_EN	2	SF_GetParam函数
SLS_1_SPEED	3	SF_GetParam函数
SLS_1_ELBOW_EN	4	SF_GetParam函数
SLS_1_JOINT_EN	5	SF_GetParam函数
SLS_1_JOINTSPEED	6	SF_GetParam函数
SLS_1_WRIST_EN	7	SF_GetParam函数
SLS_1_SHOULDER_EN	8	SF_GetParam函数
SLS_2_HAND_EN	9	SF_GetParam函数
SLS_2_SPEED	10	SF_GetParam函数
SLS_2_ELBOW_EN	11	SF_GetParam函数
SLS_2_JOINT_EN	12	SF_GetParam函数
SLS_2_JOINTSPEED	13	SF_GetParam函数
SLS_2_WRIST_EN	14	SF_GetParam函数
SLS_2_SHOULDER_EN	15	SF_GetParam函数
SLS_3_HAND_EN	16	SF_GetParam函数
SLS_3_SPEED	17	SF_GetParam函数
SLS_3_ELBOW_EN	18	SF_GetParam函数

常数名称	值	使用方法
SLS_3_JOINT_EN	19	SF_GetParam函数
SLS_3_JOINTSPEED	20	SF_GetParam函数
SLS_3_WRIST_EN	21	SF_GetParam函数
SLS_3_SHOULDER_EN	22	SF_GetParam函数
SLS_T2_HAND_EN	23	SF_GetParam函数
SLS_T2_SPEED	24	SF_GetParam函数
SLS_T2_ELBOW_EN	25	SF_GetParam函数
SLS_T2_JOINT_EN	26	SF_GetParam函数
SLS_T2_JOINTSPEED	27	SF_GetParam函数
SLS_T2_WRIST_EN	28	SF_GetParam函数
SLS_T2_SHOULDER_EN	29	SF_GetParam函数
SLS_T_SPEED	30	SF_GetParam函数
SLS_T_JOINT_EN	31	SF_GetParam函数
SLS_T_JOINTSPEED	32	SF_GetParam函数
SLS_HAND_OFS_X	33	SF_GetParam函数
SLS_HAND_OFS_Y	34	SF_GetParam函数
SLS_HAND_OFS_Z	35	SF_GetParam函数
SLS_1_DELAY	36	SF_GetParam函数
SLS_2_DELAY	37	SF_GetParam函数
SLS_3_DELAY	38	SF_GetParam函数
SLS_JOINT_POS_EN	39	SF_GetParam函数
SLS_JOINT_POS_ANGLE	40	SF_GetParam函数
SLP_A_XU_EN	41	SF_GetParam函数
SLP_A_XU_POS	42	SF_GetParam函数
SLP_A_XL_EN	43	SF_GetParam函数
SLP_A_XL_POS	44	SF_GetParam函数
SLP_A_YU_EN	45	SF_GetParam函数
SLP_A_YU_POS	46	SF_GetParam函数
SLP_A_YL_EN	47	SF_GetParam函数
SLP_A_YL_POS	48	SF_GetParam函数
SLP_A_ZU_EN	49	SF_GetParam函数
SLP_A_ZU_POS	50	SF_GetParam函数
SLP_A_ZL_EN	51	SF_GetParam函数
SLP_A_ZL_POS	52	SF_GetParam函数

常数名称	值	使用方法
SLP_B_XU_EN	53	SF_GetParam函数
SLP_B_XU_POS	54	SF_GetParam函数
SLP_B_XL_EN	55	SF_GetParam函数
SLP_B_XL_POS	56	SF_GetParam函数
SLP_B_YU_EN	57	SF_GetParam函数
SLP_B_YU_POS	58	SF_GetParam函数
SLP_B_YL_EN	59	SF_GetParam函数
SLP_B_YL_POS	60	SF_GetParam函数
SLP_B_ZU_EN	61	SF_GetParam函数
SLP_B_ZU_POS	62	SF_GetParam函数
SLP_B_ZL_EN	63	SF_GetParam函数
SLP_B_ZL_POS	64	SF_GetParam函数
SLP_C_XU_EN	65	SF_GetParam函数
SLP_C_XU_POS	66	SF_GetParam函数
SLP_C_XL_EN	67	SF_GetParam函数
SLP_C_XL_POS	68	SF_GetParam函数
SLP_C_YU_EN	69	SF_GetParam函数
SLP_C_YU_POS	70	SF_GetParam函数
SLP_C_YL_EN	71	SF_GetParam函数
SLP_C_YL_POS	72	SF_GetParam函数
SLP_C_ZU_EN	73	SF_GetParam函数
SLP_C_ZU_POS	74	SF_GetParam函数
SLP_C_ZL_EN	75	SF_GetParam函数
SLP_C_ZL_POS	76	SF_GetParam函数
SLP_J2_MON_RAD	77	SF_GetParam函数
SLP_J3_MON_RAD	78	SF_GetParam函数
SLP_J5_MON_RAD	79	SF_GetParam函数
SLP_J6_MON_RAD	80	SF_GetParam函数
SLP_J1_RANGE_MAX	81	SF_GetParam函数
SLP_J1_RANGE_MIN	82	SF_GetParam函数
SLP_J2_RANGE_MAX	83	SF_GetParam函数
SLP_J2_RANGE_MIN	84	SF_GetParam函数
SLP_J3_RANGE_MAX	85	SF_GetParam函数
SLP_J3_RANGE_MIN	86	SF_GetParam函数

常数名称	值	使用方法
SLP_J4_RANGE_MAX	87	SF_GetParam函数
SLP_J4_RANGE_MIN	88	SF_GetParam函数
SLP_J5_RANGE_MAX	89	SF_GetParam函数
SLP_J5_RANGE_MIN	90	SF_GetParam函数
SLP_J6_RANGE_MAX	91	SF_GetParam函数
SLP_J6_RANGE_MIN	92	SF_GetParam函数
SIN_1_SLS_1_EN	93	SF_GetParam函数
SIN_1_SLS_2_EN	94	SF_GetParam函数
SIN_1_SLS_3_EN	95	SF_GetParam函数
SIN_1_SLP_A_EN	96	SF_GetParam函数
SIN_1_SLP_B_EN	97	SF_GetParam函数
SIN_1_SLP_C_EN	98	SF_GetParam函数
SIN_1_SG_EN	99	SF_GetParam函数
SIN_1_ESTOP_EN	100	SF_GetParam函数
SIN_2_SLS_1_EN	101	SF_GetParam函数
SIN_2_SLS_2_EN	102	SF_GetParam函数
SIN_2_SLS_3_EN	103	SF_GetParam函数
SIN_2_SLP_A_EN	104	SF_GetParam函数
SIN_2_SLP_B_EN	105	SF_GetParam函数
SIN_2_SLP_C_EN	106	SF_GetParam函数
SIN_2_SG_EN	107	SF_GetParam函数
SIN_2_ESTOP_EN	108	SF_GetParam函数
SIN_3_SLS_1_EN	109	SF_GetParam函数
SIN_3_SLS_2_EN	110	SF_GetParam函数
SIN_3_SLS_3_EN	111	SF_GetParam函数
SIN_3_SLP_A_EN	112	SF_GetParam函数
SIN_3_SLP_B_EN	113	SF_GetParam函数
SIN_3_SLP_C_EN	114	SF_GetParam函数
SIN_3_SG_EN	115	SF_GetParam函数
SIN_3_ESTOP_EN	116	SF_GetParam函数
SIN_4_SLS_1_EN	117	SF_GetParam函数
SIN_4_SLS_2_EN	118	SF_GetParam函数
SIN_4_SLS_3_EN	119	SF_GetParam函数
SIN_4_SLP_A_EN	120	SF_GetParam函数

常数名称	值	使用方法
SIN_4_SLP_B_EN	121	SF_GetParam函数
SIN_4_SLP_C_EN	122	SF_GetParam函数
SIN_4_SG_EN	123	SF_GetParam函数
SIN_4_ESTOP_EN	124	SF_GetParam函数
SIN_5_SLS_1_EN	125	SF_GetParam函数
SIN_5_SLS_2_EN	126	SF_GetParam函数
SIN_5_SLS_3_EN	127	SF_GetParam函数
SIN_5_SLP_A_EN	128	SF_GetParam函数
SIN_5_SLP_B_EN	129	SF_GetParam函数
SIN_5_SLP_C_EN	130	SF_GetParam函数
SIN_5_SG_EN	131	SF_GetParam函数
SIN_5_ESTOP_EN	132	SF_GetParam函数
SOUT_1_STO	133	SF_GetParam函数
SOUT_1_SLS_1	134	SF_GetParam函数
SOUT_1_SLS_2	135	SF_GetParam函数
SOUT_1_SLS_3	136	SF_GetParam函数
SOUT_1_SLS_T2	137	SF_GetParam函数
SOUT_1_SLS_T	138	SF_GetParam函数
SOUT_1_SLP_A	139	SF_GetParam函数
SOUT_1_SLP_B	140	SF_GetParam函数
SOUT_1_SLP_C	141	SF_GetParam函数
SOUT_1_EP_RC	142	SF_GetParam函数
SOUT_1_EP_TP	143	SF_GetParam函数
SOUT_1_EN_SW	144	SF_GetParam函数
SOUT_2_STO	145	SF_GetParam函数
SOUT_2_SLS_1	146	SF_GetParam函数
SOUT_2_SLS_2	147	SF_GetParam函数
SOUT_2_SLS_3	148	SF_GetParam函数
SOUT_2_SLS_T2	149	SF_GetParam函数
SOUT_2_SLS_T	150	SF_GetParam函数
SOUT_2_SLP_A	151	SF_GetParam函数
SOUT_2_SLP_B	152	SF_GetParam函数
SOUT_2_SLP_C	153	SF_GetParam函数
SOUT_2_EP_RC	154	SF_GetParam函数

常数名称	值	使用方法
SOUT_2_EP_TP	155	SF_GetParam函数
SOUT_2_EN_SW	156	SF_GetParam函数
SOUT_3_STO	157	SF_GetParam函数
SOUT_3_SLS_1	158	SF_GetParam函数
SOUT_3_SLS_2	159	SF_GetParam函数
SOUT_3_SLS_3	160	SF_GetParam函数
SOUT_3_SLS_T2	161	SF_GetParam函数
SOUT_3_SLS_T	162	SF_GetParam函数
SOUT_3_SLP_A	163	SF_GetParam函数
SOUT_3_SLP_B	164	SF_GetParam函数
SOUT_3_SLP_C	165	SF_GetParam函数
SOUT_3_EP_RC	166	SF_GetParam函数
SOUT_3_EP_TP	167	SF_GetParam函数
SOUT_3_EN_SW	168	SF_GetParam函数
POS_ROT_U	169	SF_GetParam函数
POS_ROT_V	170	SF_GetParam函数
POS_ROT_W	171	SF_GetParam函数
POS_OFS_X	172	SF_GetParam函数
POS_OFS_Y	173	SF_GetParam函数
POS_OFS_Z	174	SF_GetParam函数
SF_TOOLVERSION	1	SF_GetParam$函数
SF_CHECKSUM	2	SF_GetParam$函数
SF_LAST_MODIFIED	3	SF_GetParam$函数
SF_ROBOT_MODEL_NAME	4	SF_GetParam$函数
SF_ROBOT_CHECKSUM	5	SF_GetParam$函数
SF_HOFS	6	SF_GetParam$函数
SF_HOFS_LAST_MODIFIED	7	SF_GetParam$函数
SLS_1	1	SF_LimitSpeedS, SF_LimitSpeedSEnable
SLS_2	2	SF_LimitSpeedS, SF_LimitSpeedSEnable
SLS_3	3	SF_LimitSpeedS, SF_LimitSpeedSEnable
_SLS_T	9	SF_LimitSpeedS, SF_LimitSpeedSEnable
_SLS_T2	10	SF_LimitSpeedS, SF_LimitSpeedSEnable