卡诺普
CROBOTP

控制器占据半壁江山的机器人品牌
让客户用好机器人

CrobotpOS 编程指令说明书

CRP-JS-YF2-NT-00049-A8

请确保相关说明书到达本产品的最终使用者手中。

CROBOTP 相关说明书：

卡诺普机器人安全手册

CrobotpOS 简易操作手册

CrobotpOS 使用说明书

CrobotpOS 系统 PLC 说明书

CRP-G9-CD60B 电柜说明书

十分感谢您选用本公司产品！

本产品相关手册请妥善保管，以备需要时查阅！如设备需要转手，请将相关资料一并转交对方！

机器人相关手册未做说明的按键、功能、选项视为不具备，请勿使用！

修订记录		
日期	版本	内容
2022-06-13	A0	初稿
2023-04-06	A1	修改封面与部分图片
2023-10-12	A2	修订内容
2023-10-24	A3	补充成对使用指令示例
2024-01-22	A4	增加运动指令滤波等级说明
		新增门型指令
		新增中断与中断结束指令
		新增碰撞指令
		新增 If 指令条件的与或功能
		特殊指令新增创建用户坐标系指令
2024-06-20	A5	新增 Set 指令 Convert 数据类型转换功能
		新增 Set 指令 Split 字符串拆分功能
		新增 Set 指令自定义文本功能
		新增 Set 指令 ConutDis 两点距离计算功能
		新增 Socket 通讯指令
		新增视觉指令
		新增码垛指令
		CRX9 软件系统更名为 CrobotpOS
		新增计时指令
		运动指令新增偏移量、自定义文本功能，附加条件
		新增整体偏移指令

		新增激光焊接指令
		新增多层多道、寻位、点焊、电弧跟踪指令
		新增计算偏移指令
2024-11-05	A6	新增 call 指令传参功能
		新增运动综合应用示例
		修订部分内容
2025-1-20	A7	新增 Note、LatchSignalPos、GetLatchPos、Speed、运动指令增加 SP 滤波百分比、【待发布】
2025-2-10		新增 TorqueLimit
2025-3-24		新增 SET 指令针对 TOOL、USER 工具及用户坐标系参数值修改
2025-04-08		主要对指令 Switch 语句中程序举例使用了 GI 进行修改为 UI 变量
2025-04-12	A8	主要针对激光跟踪的相关指令进行更新

用户须知

指令元素中“[ ]”内的指令元素表示可选元素，“\”表示指令中的可选元素默认为不选择状态。

目　录

前 言

1. 本手册适用于 CRX9 和 CRA9 控制系统。

2. 在阅读本手册只介绍相关指令的使用，示教器操作请查阅《CrobotpOS 使用说明书（C）》或《CrobotpOS 使用说明书（E）》。

3. 在使用机器人之前，请务必仔细阅读本公司机器人相关说明书，并在理解了该项内容基础上再进行机器人操作。

4. 本公司郑重建议: 所有参与机器人操作、示教、维护、维修、点检的人员，需预先学习本公司系统的操作说明书。

5. 本公司保留未经预先通知而改变、修订或更新本手册的权利。

6. 事先未经本公司书面许可，不可以将本手册全部或其中的一部分再生或复制。

7. 请将本手册小心存放，确保本说明书到达最终使用者手中。机器人如果需要重新安装、或搬运到不同地点、或卖给其他用户时，请务必将本手册附上。一旦出现丢失或严重损坏，请您和本公司代理商或技术人员联络。

8. 所有参数指标和设计可能会随时修改，在不影响使用效果的前提下，恕不另行通告。

9. 我们试图在本说明书中描述可能多的情况。然而对于那些不必做的和不可能发生的情况，由于存在各种可能性，我们没有描述。因此，对于那些在说明书中没有特别进行描述的情况，可以视为“不可能”的情况。

10. 在本书编写的过程中难免会出现遗漏和错误，如在阅读过程中发现有错误或不能理解的地方，欢迎来电咨询并指正。

安全

简介

本节主要介绍在使用机器人时需要注意的安全原则和流程，在使用机器人之 前，请务必熟读并理解本章中所述内容，并按安全操作规程操作机器人。且使用前（安装、运转、保养、检修），请务必熟读并全部掌握本说明书和其他相关资料。

本手册给出的图表、顺序和详细解释可能并不绝对正确。所以，在使用本手册去作业时，有必要投以最大的注意力。一旦出现未说明的问题或麻烦，请与卡诺普联系。

为保证每项工作的安全，请阅读并完全理解本手册和《机器人安全手册》、相关法律法规及相关资料中各种有关安全的解释和描述，同时请为各项工作采取合适的安全措施。

除安全章节外，请注意在文档的必要部分有其他的安全提示。

安全责任说明

本手册并不对使用非本公司机器人的应用做担保。同时，我司将不会对使用这样的机器人而可能导致的事故、损害和(或)与工业产权相关的任何问题承担责任。

我司尽可能提供出可靠的安全信息，但不对因使用本手册及其中所述产品引起的意外或间接事故承担责任。

除本手册中有明确陈述之外，本手册的内容不应解释为卡诺普对个人损失、财产损坏或具体适用性做出任何担保或保证。

卡诺普对本手册可能出现的错误概不负责。

安全标志

标志	说明
 危险	表示如果无视该标识并进行错误使用，则可能会导致死亡或重伤等。
 警告	误操作时有危险，可能发生中等程度伤害或轻伤事故及设备故障。

 小心	不遵守本标志内容可能会引起人身伤害和/或机械损伤。
★ 注意	表示关于机器人规格、操作和维护的注意信息。

说明：即使是“小心”所记载的内容，也会因情况不同而产生严重后果，因此任何一条注意事项都极为重要，请务必严格遵守。

甚至在有些地方连“警告”或“危险”等内容都未记载，也是用户必须严格遵守的事项。

拟定用途

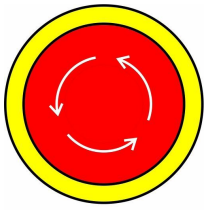
机器人控制器以及机器人只限于一般工业设备使用，不可用于与预定用途违背的应用，禁止用途包括但不限于以下情况：

- 用于易燃易爆等危险环境中；
- 用于临近水或易溅水的场所和易发生腐蚀的环境中；
- 用于移动或搬运人或其他动物的装置；
- 用于涉及人命的医疗设备等装置；
- 用于对社会性及公共性有重大影响的装置；
- 用于车载、船舶等受到振动环境；
- 用于攀爬工具使用。

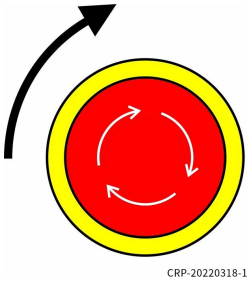
急停按钮

紧急停止属于安全停止的一种，是机器人系统中优先级最高的功能。在示教器、电柜、工位盒等均安装有急停按钮。如遇紧急情况，用户可按下急停按钮，立即切断机器人电源。

紧急停止用的急停按钮大多数使用红色的操作主体，最常见的外形是蘑菇头型。如下图所示。



若需复位，则需按照急停按钮上的箭头方向旋转（如下图所示），急停按钮将弹起复位。



使用前安全须知

1、 搬运和安装机器人时，请务必按照卡诺普公司说明书中所示的方法进行。否则可能导致机器人翻倒，引发事故；

2、 请务必在机器人安装前划分出安全区域。可在机器人工作区域周围安装栅栏及警示牌保证机器人安全工作，防止闲杂人等进入以及防止机器人伤人；

3、机器人上方不能有悬挂物，以防掉落砸坏机器人等设备；

4、严禁倚靠电控柜，或者随意触动按钮，以防机器人产生未预料的动作，引起人身伤害或者设备损坏；

5、拆分机器人时，注意机器人上可能掉落的零件而砸伤人员；

6、在进行外围设备的个别调试时，务必断开机器人电源后执行；

7、外围设备均应连接适当的地线；

8、初次使用机器人操作时，请务必先以低速运行，待运行无误后再逐渐加速。

9、请注意对电控柜与机器人、外围设备间的配线及配管采取防护措施，以免被人踩坏或被叉车辗压而坏；

10、任何工作的机器人都可能因有不可预料的动作，对工作范围内的人员造成严重的伤害或者对设备造成破坏。在准备机器人工作前，需测试各安全措施（栅栏门、抱闸、安全指示灯）的可靠性；

11、 在开启机器人前，确保机器人工作范围内没有其他人员；

12、 通过软件设定的动作范围及负载条件切勿超出产品规格表中的规定值，设置不当可能造成人员伤害或机器损坏；

13、在进入操作区域内工作前，即使机器人没有运行，也要关掉电源或者按下急停按钮；

14、当在机器人工作区内编程时，设置相应看守，保证机器人能在紧急情况下迅速停止。示教和点动机器人时不要带手套操作，点动机器人时要尽量采用低速操作，遇到异常情况时可有效控制机器人停止；

15、必须知道机器人控制器和外围控制设备上的紧急停止按钮的位置，以便在紧急情况下能准确地按下这些按钮；

16、永远不要认为机器人处于静止状态时其程序就已经完成。此时机器人很有可能是在等待让它继续运动的输入信号；

操作前注意事项

注意

★进行机器人示教作业前要检查以下事项，有异常则应及时修理或采取其他必要措施。

- 机器人动作有无异常。
- 原点是否校准正确。
- 与机器人相关联的外部辅助设备是否正常。

★操作机器人必须确认

- 操作人员是否接受过机器人操作的相关培训。
- 对机器人的运动特性有足够的认识。
- 对机器人的危险性有足够的了解。
- 未酒后上岗。
- 未服用影响神经系统、反应迟钝的药物。

紧急停止

危险

★ 操作机器人前，请按下急停键，并确认伺服主电源被切断，电机处于失电并抱闸状态。伺服电源切断后，伺服电源指示按钮为红色。

紧急情况下，若不能及时制动机器人，则可能引发人身伤害或设备损坏事故。

★ 解除急停后再接通伺服电源时，要解除造成急停的事故后再接通伺服电源。由于误操作造成的机器人动作，可能引发人身伤害事故。

机器人操作注意事项

★在机器人动作范围内示教时，请遵守以下原则：

- 保证机器人在视野范围内

- 严格遵守操作步骤
- 考虑机器人突然向自己所处方位运动时的应变方案
- 确保设置躲避场所，以防万一

由于误操作造成的机器人动作，可能引发人身伤害事故。

★进行以下作业时，请确认机器人的动作范围内操作人员和障碍物：

- 机器人控制电柜接通电源时
- 用示教编程器操作机器人时
- 试运行时
- 自动再现时

不慎进入机器人动作范围内或与机器人发生接触，都有可能引发人身伤害事故。发生异常时，请立即按下急停按钮。

★示教器用完后须放回原处，并确保放置牢固。

- 如不慎将示教编程器放在机器人、夹具或地上，当机器人运动时，示教编程器可能与机器人或夹具发生碰撞，从而引发人身伤害或设备损坏事故。
- 防止示教器意外跌落造成机器人误动作，从而引发人身伤害或设备损坏事故。
- 示教器 IP 防护等级较低

1. 运动指令

1.1. MoveJ---关节运动

1.1.1. 指令用法

MoveJ VJ=100 PL=9.0 T=1

用于机器人对末端运动路径没有直线要求的场合，机器人各轴将以设定的百分比速度运行到目标位置。所有轴均同时运动，但机器人运动轨迹是一条不确定的曲线。

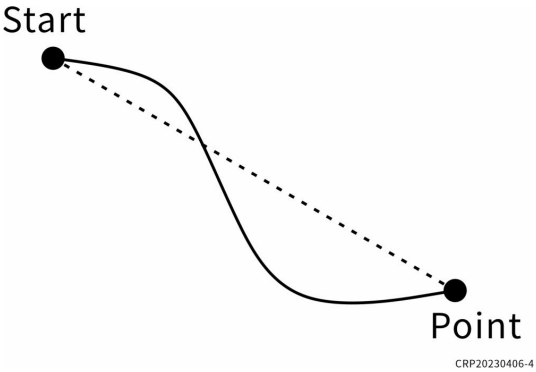


图 1.1.1

注意：

1. 指令存储位置为空间位姿（X、Y...Rz），通过指定的工具/用户坐标号及轴配置信息反解机器人关节坐标。
2. 插入运动指令需在伺服上电，且安全开关处于二档状态才能记录点位，指令插入成功。

1.1.2. 程序运行

运行指令时机器从一点迅速运行到另一点，该指令过程中各轴均以设定轴速率运动，且同时到达目标点，其运动轨迹非线性。只有当机器人各轴已经到达目标位置（目标位置与指令中 PL 参数有关系），才会执行下一条指令。一般用于没有轨迹要求的场合。

1.1.3. 可变元素

MoveJ [\Topoint] VJ=[Speed] PL=[zone] [\ACC] [\Dec] [\S] T=[Tool] [\U] [\Offset] [\signal]

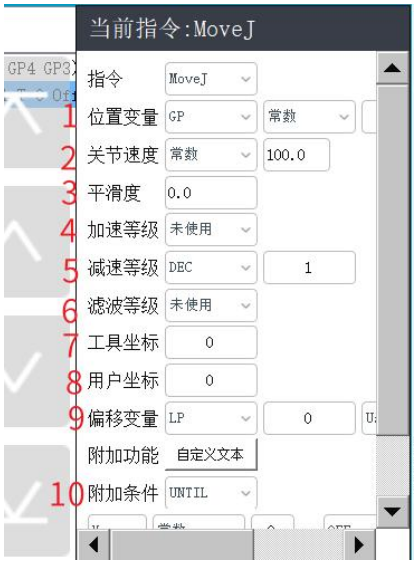


图 1.1.2

1. [\Topoint]

变量类型：GP（全局变量）、LP（局部变量）。

解释：机器人和外部轴的目标点位置。可以选择使用 GP、LP 变量。当使用该变量时，变量中的值为机器人和外部轴的目标点位置。不使用变量时机器人目标位置储存在指令中。

2. VJ=[Speed]

数据类型： float（范围：0.0-100.0）

变量类型：常数、GR（全局变量）、LR（局部变量）

解释：关节运动以最大速度为基准的百分比速度。可以选择使用常数、GR、LR 变量。当使用变量时，变量中的值作为机器人运行速度的百分比。

3. PL=[zone]

数据类型：float（范围：0.0-9.0）

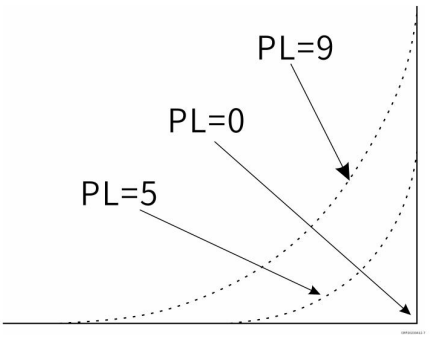


图 1.1.3

解释：机器人在运动轨迹中到达目标点的逼近等级。描述了所生成拐角的路径大小。逼近等级设置越小，机器人运行轨迹时生成拐角路径越小。

4. [\ACC]

数据类型：Int（范围：1-20）

解释：加速度，该加速度与机器人运动方式有关。加速等级设置越小机器人加速越快，加速等级设置越大加速越慢。默认不使用。

5. [\Dec]

数据类型：Int（范围：1-20）

解释：减速度，该减速度与机器人运动方式有关。减速等级设置越小机器人减速越快，减速等级设置越大减速越慢。默认不使用。

6. [\S]

数据类型：Int（范围：0-9）

解释：滤波等级共分为 0-9 级，默认为 0。滤波等级主要用于小距离、速度慢，可将等级改大，提高加减速能力与速度。滤波等级会影响节拍，滤波等级越大，时间越短。参数不合适可能造成机器人末端抖动，所以请根据实际情况调节。下图为滤波等级的一个速度示意图。

7. [\SP]

该参数需要 CrobotpOS 版本要求 1.13.0 及以上，低于此版本的软件请按要求进行升级，否则功能不能正常。

数据类型：Int（范围：10-300）

解释：滤波时间百分比，默认为 100。滤波原理同上，此处用百分比表示滤波时间倍率，允许突破参数设置的滤波时间上限。下图为滤波百分比的一个速度示意图。

注意：指令里面滤波等级只针对步长滤波有效。

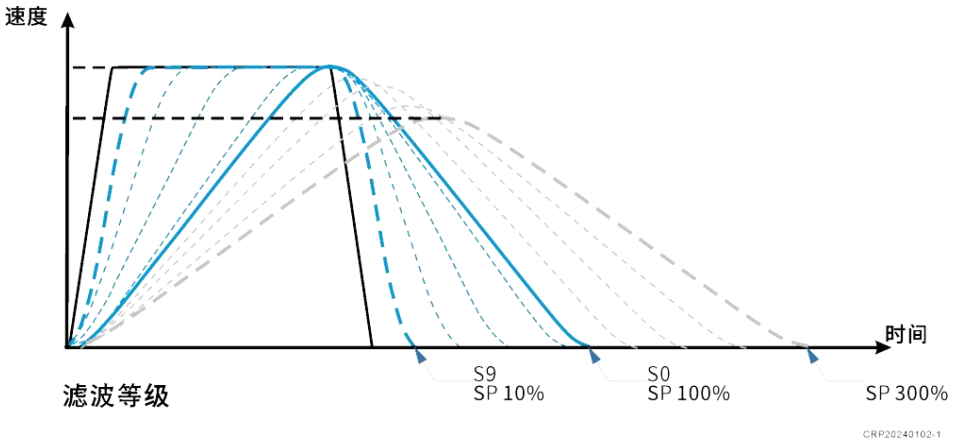


图 1.1.4

7. T=[Tool]

数据类型:Int (范围：0-49)

变量类型：常数、UI、GIN、GOT

解释：机器人移动时使用的工具坐标系。0 号工具默认工具坐标点在机器人法兰末端中心处。当使用变量时，变量中的值作为该点位对应工具坐标系进行反解点位。

8. [\U]

数据类型：Int (范围：0-49)

解释：机器人移动时使用的用户坐标系。0 号工件坐标为默认用户坐标系。可选择使用该元素。如果使用协同的外部轴时必须使用该参数。

9. [\Offset]

变量类型：GP/LP

解释：机器人在一个位置添加一个偏移量，并选择参考坐标：Tool 或者 User。在选取偏移变量的时候也要注意偏移变量不能太大，避免超出可达范围。可选择使用附加功能——自定义文本编辑偏移量，详情请查看详细举例。

10. [\signal]

变量类型:Until（X/SX/M/SM/T/C），Search（GP/LP）。

解释：该功能默认不使用，可选项如下:

- 选择 Until 时，当条件满足，该指令行停止执行，转入下一行执行。否则执行当前行直到结束后，转入下一行。
- Search 指令附加在寻位开始点，进行工件寻位和数据存储。

1.1.4. 详细举例

例 1：简单指令

MoveJ VJ=100 PL=9.0 T=1

将工具 1 的工具中心点沿非线性路径移动到指令中存储的指定位置，其速度百分比为 100%，轨迹平滑度为 9.0。

例 2：使用 GP 位置变量

MoveJ GP0 VJ=100 PL=2.0 T=2

将工具 2 的工具中心点沿非线性路径移动到位置 GP0，其速度百分比为 100%，轨迹平滑度为 2.0。

例 3：使用加速度

MoveJ VJ=GR0 PL=6.0 Acc= 1 T=1

将工具 1 的工具中心点沿非线性路径移动到指令中存储的指定位置，其速度百比为储存在变量 GR0 中的值，轨迹平滑度为 6.0，加速度为 1。

例 4：参考用户坐标移动

MoveJ GP0 VJ=100 PL=2.0 T=2 U=1

将工具 2 的工具中心点沿非线性路径移动到 GP0，在关于 U1 的用户坐标系下的指定位置，其速度百分比为 100%，轨迹平滑度为 2.0。

例 5：偏移变量

- **方法 1：**

图 1.1.5

MoveJ GP0 VJ=100 PL=2.0 Dec=5 T=2 U=1 Offset LP1 User

将工具 2 的工具中心点沿非线性路径移动到 GP0，附加指令 offset 表示对当前点位进行偏移，偏移变量为 LP1 中的参数，User 表示将参考 U1(运动指令中的用户坐标系号)进行偏移，其速度百分比为 100%，轨迹平滑度为 2.0，减速度为 5。

- **方法 2**:

使用附加功能，自定义文本设置偏移量。如下图所示。

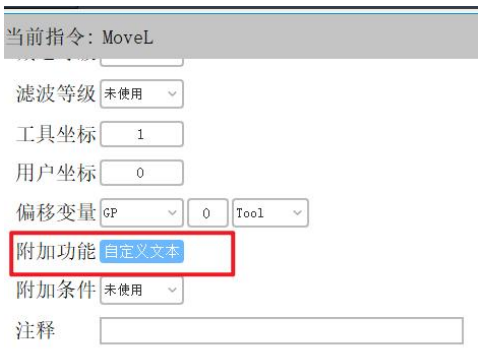


图 1.1.6

图 1.1.7

MoveJ VL=100 PL=0.0 T=1 Offset 100 0 0 \Rz=10 T00l Ex1=80

指令效果为将当前点位按照偏移量(100 0 0 \RZ=10，Ex1=80)，沿着工具坐标系 1 的 X 正方向移动 100mm，Y、Z 方向不偏移，最后绕 Z 轴正方向旋转 10°，外部轴 1 向正方向偏移 80(单位根据外部轴定义确定是°还是 mm)，获得最终点位。

总结：

1、单点偏移可以通过指令中[偏移变量]进行设置，或者通过[附加功能]进行自定义文本输入。

2、单点偏移可以和整体偏移进行叠加，使用时请注意。

例 6：附加条件—Until 条件跳转行

MoveJ GP0 VJ=100 ID=6 PL=2.0 Acc=10 Dec=5 T=2 U=1 Offset GP1 User Until X0=ON

将工具 2 的工具中心点沿非线性路径移动到 GP0，在关于 U1 的用户坐标系下偏移位置为 GP1 的指定位置，其速度百分比为 100%，轨迹平滑度为 2.0，加速度等级为 10、减速度等级为 5。运行过程中，若 X0 的值等于 ON，则直接停止运行该行指令，然后直接进入下一行运行；如果 X0 的值不等于 ON，则将该行指令运行完成后再进入下一行运行。

1.2. MoveL---直线运动

1.2.1. 指令用法

MoveL VL=100 PL=9.0 T=1

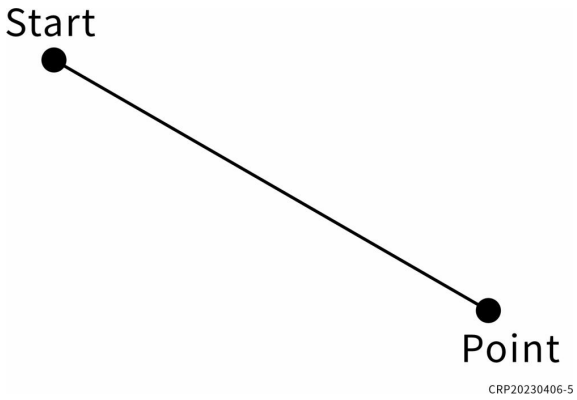


图 1.2.1

工具中心点以直线的轨迹方式运动到目标点。运动姿态参照 TCP 坐标方向。

注意：插入运动指令需在伺服上电，且安全开关处于二档状态才能记录点位，指令插入成功。

1.2.2. 程序运行

运行该指令，机器人以恒定的速度、工具末端沿直线轨迹移动。当机器人工具末端到达目标位置（目标位置与指令中 PL 参数有关系），才会执行下一条指令。一般用于有轨迹要求的场合。

1.2.3. 可变元素

MoveL [\Topoint] [\ID] VL=[Speed] PL=[zone] [\ACC] [\Dec] [\S] T=[Tool] [\U] [\Offset] [\Signal]

1. [\Topoint]

变量类型：GP（全局变量）、LP（局部变量）。

解释：机器人和外部轴的目标点位置。可以选择使用 GP、LP 变量。当选择变量时，机器人目标点位置储存在变量中。不选择变量时机器人目标位置储存在指令中。

2. [\ID]

数据类型：Int (范围：1-99)

解释：机器人 ID 号，协同同步时必须使用。在其他任何时候都不允许使用。所有协同同步任务时 ID 号必须相同，ID 号确保运行时不混淆。默认不使用

3. VL=[Speed]

数据类型： float（范围：0-3000）

变量类型：GR（全局变量）、LR（局部变量）

解释：直线运动最大速度 3000mm/s。可以选择使用 GR、LR 变量。当使用变量时，机器人运行速度为变量中的值。

4. PL=[zone]

数据类型： float（范围：0.0-9.0）

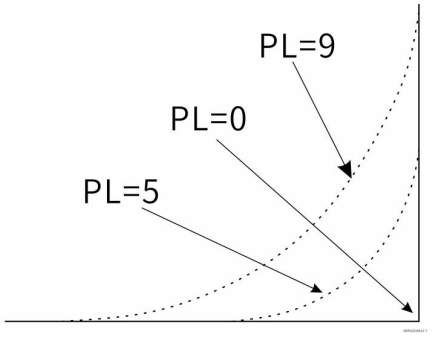


图 1.2.2

解释：机器人在运动轨迹中到达目标点的逼近等级。描述了所生成拐角的路径大小。逼近等级设置越小，机器人运行轨迹时生成拐角路径越小。

5. [\ACC]

数据类型：Int（范围：1-20）

解释：加速度，该加速度与机器人运动方式有关。加速等级设置越小机器人加速越快，加速等级设置越大加速越慢。默认不使用。

6. [\Dec]

数据类型：Int（范围：1-20）

解释：减速度，该减速度与机器人运动方式有关。减速等级设置越小机器人减速越快，减速等级设置越大减速越慢。默认不使用。

7. [\S]

数据类型：Int（范围：0-9）

解释：滤波等级共分为 0-9 级，默认为 0。滤波等级主要用于小距离、速度慢，可将等级改大，提高加减速能力与速度。滤波等级会影响节拍，滤波等级越大，时间越短。参数不合适可能造成机器人末端抖动，所以请根据实际情况调节。下图为滤波等级的一个速度示意图。

8. [\SP]

该参数需要 CrobotpOS 版本要求 1.13.0 及以上，低于此版本的软件请按要求进行升级，否则功能不能正常。

数据类型：Int（范围：10-300）

解释：滤波时间百分比，默认为 100。滤波原理同上，此处用百分比表示滤波时间倍率，允许突破参数设置的滤波时间上限。下图为滤波百分比的一个速度示意图。

注意：指令里面滤波等级只针对步长滤波有效。

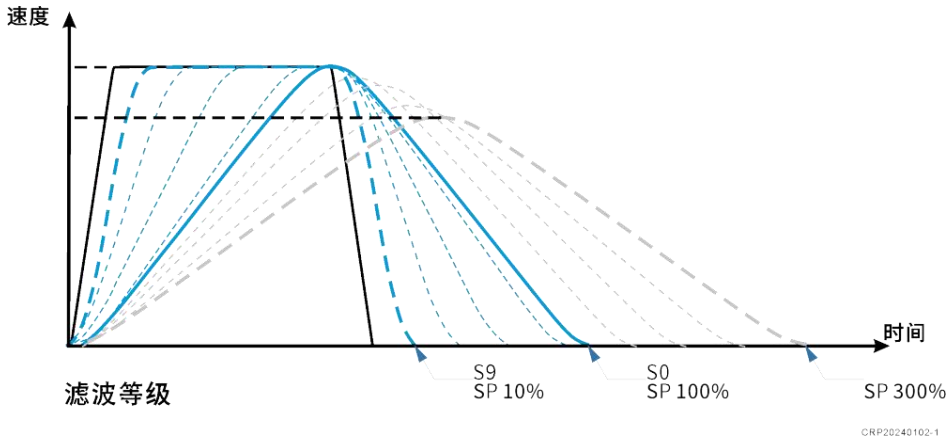


图 1.2.3

9. T=[Tool]

数据类型:Int (范围：0-49)

变量类型：常数、UI、GIN、GOT

解释：机器人移动时使用的工具坐标系。0 号工具默认工具坐标点在机器人法兰末端中心处。当使用变量时，变量中的值作为该点位对应工具坐标系进行反解点位。

10. [\U]

数据类型：Int (范围：0-49)

变量类型：常数、UI、GIN、GOT

解释：机器人移动时使用的用户坐标系。0 号用户坐标为默认用户坐标系。可选择使用该元素。如果使用协同的外部轴时必须使用该参数。当使用变量时，变量中的值作为该点位对应用户坐标系进行反解点位。

11. [\Offset]

变量类型：GP/LP

解释：机器人在一个位置添加一个偏移量，并选择参考坐标：Tool 或者 User。同时在选取偏移变量的时候也要注意偏移变量不能太大，避免超出可达范围。可选择使用附加功能——自定义文本编辑偏移量，详情请查看详细举例。

12. [\signal]

变量类型:Until（X/SX/M/SM/T/C），Search（GP/LP）。

解释：该功能默认不使用，可选项如下:

- 选择 Until 时，当条件满足，该指令行停止执行，转入下一行执行。否则执行当前行直到结束后，转入下一行。
- Search 指令附加在寻位开始点，进行工件寻位和数据存储。

1.2.4. 详细举例

例 1：简单指令

MoveL VL=100 PL=9.0 T=1

将工具 1 的工具中心点沿直线路径移动到指令中存储的指定位置，其速度为 100mm/s，轨迹平滑度为 9.0。

例 2：使用 GP 位置变量

MoveL GP0 VL=100 PL=2.0 T=2

将工具 T2 的工具中心点沿直线路径移动到位置 GP0，其速度为 100mm/s，轨迹平滑为 2.0。

例 3：使用加速度

MoveL VL=GR0 PL=6.0 Acc= 1 T=1

将工具 T1 的工具中心点沿直线路径移动到指令中存储的指定位置，其速度为 GR0 变量中的值，轨迹平滑度为 6.0，加速度为 1。

例 4：设置参考用户坐标系

MoveL GP0 VL=100 PL=2.0 T=2 U=1

将工具 T2 的工具中心点沿直线路径移动到 GP0，在关于 U1 的用户坐标系下的指定位置，其速度为 100mm/s，轨迹平滑度为 2.0。

例 5：偏移变量

- 方法 1：

图 1.2.4

MoveL VL=100 PL=0.0 T=1 Offset GP0 T00l

将工具 1 的工具中心点沿固定直线路径移动到点，附加指令 offset 表示对当前点位进行偏移，偏移变量为 GP0 中的参数，Tool 表示将参考 T1(运动指令中的工具坐标系及序号)进行偏移，其速度百分比为 100%，轨迹平滑度为 0.0。

- 方法 2：

使用附加功能，自定义文本设置偏移量。如下图所示

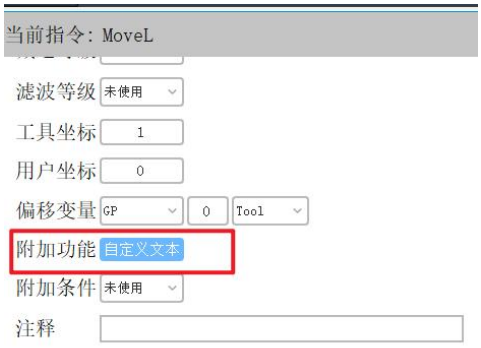


图 1.2.5

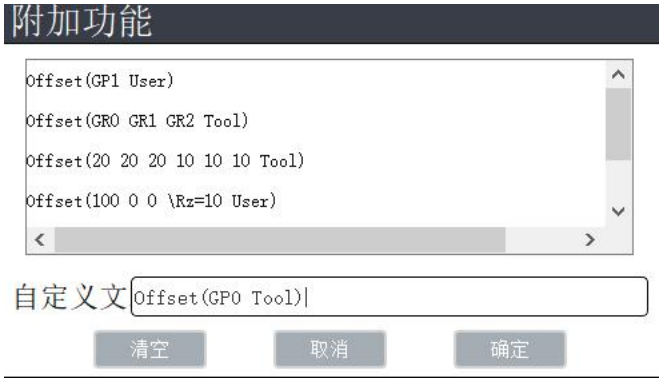


图 1.2.6

MoveL VL=100 PL=0.0 T=1 Offset 100 0 0 \Rz=10 T00l Ex1=80

指令效果为将当前点位按照偏移量(100 0 0 \RZ=10，Ex1=80)，沿着工具坐标系 1 的 X 正方向移动 100mm，Y、Z 方向不偏移，最后绕 Z 轴正方向旋转 10°，外部轴 1 向正方向偏移 80(单位根据外部轴定义确定是°还是 mm)，获得最终点位。

总结：

1、单点偏移可以通过指令中[偏移变量]进行设置，或者通过[附加功能]进行自定义文本输入。

2、单点偏移可以和整体偏移进行叠加，使用时请注意。

例 6：附加条件—Until 条件跳转行

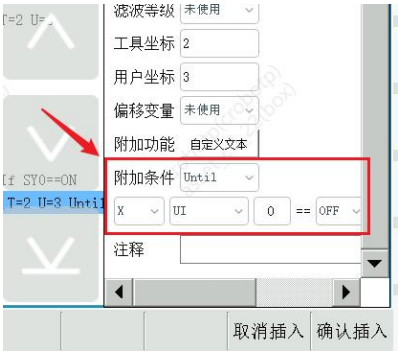


图 1.2.7

MoveJ GP0 VJ=100 ID=6 PL=2.0 Acc=10 Dec=5 T=2 U=1 Offset GP1 User Until X0=ON

将工具 2 的工具中心点沿非线性路径移动到 GP0，在关于 U1 的用户坐标系下偏移位置为 GP1 的指定位置，其速度百分比为 100%，轨迹平滑度为 2.0，加速度等级为 10、减速度等级为 5。运行过程中，若 X0 的值等于 ON，则直接停止运行该行指令，然后直接进入下一行运行；如果 X0 的值不等于 ON，则将该行指令运行完成后再进入下一行运行。

例 7：附加条件—Search 工件寻位和数据存储

图 1.2.8

```
SearchStart TN=0 //寻位开始，使用 0 号寻位工艺
MoveL VL=100 PL=0.0 T=1      //a 寻位起始点
MoveL VL=100 ACC=1 PL=0.0 T=1 Search GP1 Point //设置寻位开始点，设定位置数据存储在 GP1 变量；寻点，寻点完成后自动返回
MoveL VL=100 PL=0.0 T=1      //寻位起始点
MoveL VL=100 ACC=1 PL=0.0 T=1 Search GP2 Point X+//设置寻位开始点，设定位置数据存储在 GP2 变量；沿着寻位工艺 0 中设定的坐标系 X 正方向寻点，寻点完成后自动返回
SearchEnd TN=1  //寻位结束
```

1.3. MoveC---圆弧运动

1.3.1. 指令用法

MoveL VL=100 PL=9.0 T=1

MoveC VL=100 PL=9.0 Point=2 T=1

MoveC VL=100 PL=9.0 Point=3 T=1

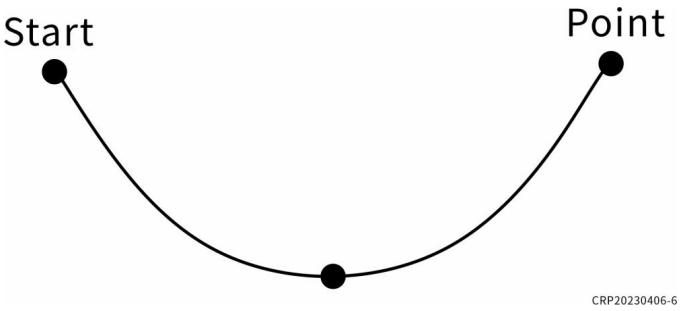


图 1.3.1

将工具中心点以圆弧的轨迹方式运行到目标点。运动姿态参照 TCP 坐标方向。机器人做圆弧运动时，分别需要一条含 Point=2 和一条含 Point=3 的圆弧指令配合使用，指令中除点位数据外所有元素参数必须一致。

注意：

1. 插入运动指令需在伺服上电状态，且安全开关处于二档状态才能记录点位，指令插入成功。
2. 在一段程序中，第一次设置 Point2 和 Point3 的程序语句之前，必须有一条 MoveL，MoveJ，MoveAbsJ 或 MoveJump 的程序语句来确定 Start 点的位置，否则会出现如下错误：“第一行指令执行失败。
3. 但如果编写一段程序，完成如下图所示的一个连续的圆弧运动，则前一次圆弧运动的 Point3 可以作为下一次圆弧的 Start 点。

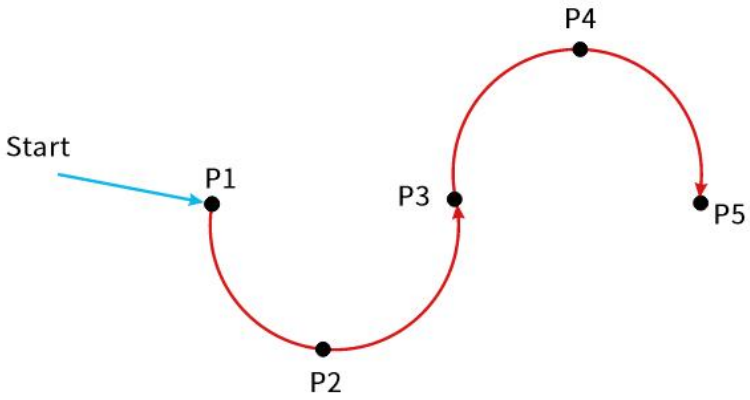


图 1.3.2

1.3.2. 程序运行

运行该指令机器人将以恒定的速度，以当前位置为起点沿圆弧轨迹移动工具。当机器人工具末端到达目标位置（目标位置与指令中 PL 参数有关系），才会执行下一条指令。一般用于有轨迹要求的场合。

1.3.3. 可变元素

MoveC [\Topoint] [\ID] VJ=[Speed] PL=[zone] Point=[PT] [\ACC] [\Dec] [\S] T=[Tool] [\U] [\Offset] [\Signal]

1. [\Topoint]

变量类型：GP（全局变量）、LP（局部变量）。

解释：机器人和外部轴的目标点位置。可以选择使用 GP、LP 变量。当选择变量时，机器人目标点位置储存在变量中。不选择变量时机器人目标位置储存在指令中。

2. [\ID]

数据类型：Int (范围：1-99)

解释：机器人 ID 号。协同同步时必须使用。在其他任何时候都不允许使用。所有协同同步任务时 ID 号必须相同，ID 号确保运行时不混淆。默认不使用。

3. VJ=[Speed]

数据类型： float（范围：0-100）

变量类型：GR（全局变量）、LR（局部变量）

解释：圆弧运动最大速度 3000mm/s。可以选择使用 GR、LR 变量。当使用变量时机器人运行速度为变量中的值。

4. PL=[zone]

数据类型：float（范围：0.0-9.0）

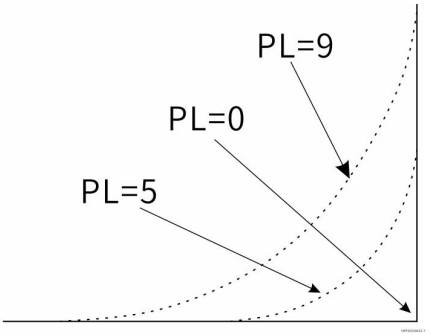


图 1.3.3

解释：机器人在运动轨迹中到达目标点的逼近等级。描述了所生成拐角的路径大小。逼近等级设置越小，机器人运行轨迹时生成拐角路径越小。

5. Point=[PT]

变量类型：Int（范围：2-3）

解释：指定指令中机器人圆弧轨迹的中间点或目标点。

6. [\ACC]

数据类型：Int（范围：1-20）

解释：加速度，该加速度与机器人运动方式有关。加速等级设置越小机器人加速越快，加速等级设置越大加速越慢。默认不使用。

7. [\Dec]

数据类型：Int（范围：1-20）

解释：减速度，该减速度与机器人运动方式有关。减速等级设置越小机器人减速越快，减速等级设置越大减速越慢。默认不使用。

8. [\S]

数据类型：Int（范围：0-9）

解释：滤波等级共分为 0-9 级，默认为 0。滤波等级主要用于小距离、速度慢，可将等级改大，提高加减速能力与速度。滤波等级会影响节拍，滤波等级越大，时间越短。参数不合适可能造成机器人末端抖动，所以请根据实际情况调节。下图为滤波等级的一个速度示意图。

9. [\SP]

该参数需要 CrobotpOS 版本要求 1.13.0 及以上，低于此版本的软件请按要求进行升级，否则功能不能正常。

数据类型：Int（范围：10-300）

解释：滤波时间百分比，默认为 100。滤波原理同上，此处用百分比表示滤波时间倍率，允许突破参数设置的滤波时间上限。下图为滤波百分比的一个速度示意图。

注意：指令里面滤波等级只针对步长滤波有效。

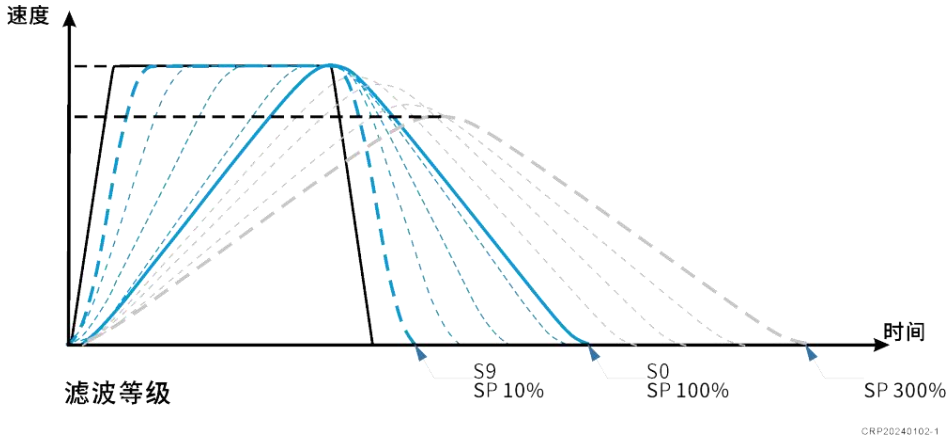


图 1.3.4

10. T=[Tool]

数据类型:Int (范围：0-49)

变量类型：常数、UI、GIN、GOT

解释：机器人移动时使用的工具坐标系。0 号工具默认工具坐标点在机器人法兰末端中心处。当使用变量时，变量中的值作为该点位对应工具坐标系进行反解点位。

11. [\U]

数据类型：Int (范围：0-49)

变量类型：常数、UI、GIN、GOT

解释：机器人移动时使用的用户坐标系。0 号用户坐标为默认用户坐标系。可选择使用该元素。如果使用协同的外部轴时必须使用该参数。当使用变量时，变量中的值作为该点位对应用户坐标系进行反解点位。

12. [\Offset]

变量类型：GP/LP

解释：机器人在一个位置添加一个偏移量，可选择参考坐标：Tool 或者 User。同时在选取偏移变量的时候也要注意偏移变量不能太大，避免超出可达范围。

13. [\signal]

变量类型:Until（X/SX/M/SM/T/C），Search（GP/LP）。

解释：该功能默认不使用，可选项如下:

- 选择 Until 时，当条件满足，该指令行停止执行，转入下一行执行。否则执行当前行直到结束后，转入下一行。
- Search 指令附加在寻位开始点，进行工件寻位和数据存储。

1.3.4. 详细举例

例 1：简单指令

```
MoveL VL=100 PL=9.0 T=1

MoveC VL=100 PL=9.0 Point=2 T=1

MoveC VL=100 PL=9.0 Point=3 T=1
```

将工具 T1 的工具中心点从当前位置以固定的姿态沿圆弧路径，以机器人当前点为起点，以 Point=2 为过渡点，运行到 Point=3 目标点，其速度为 100mm/s，轨迹平滑度为 9.0。

例 2：使用 GP 位置变量

```
MoveL VL=100 PL=9.0 T=1

MoveC GP0 VL=100 PL=2.0 Point=2 T=2

MoveC GP1 VL=100 PL=2.0 Point=3 T=2
```

将工具 T1 的工具中心点从当前位置以固定的姿态沿圆弧路径以机器人当前点为起点，GP0 为过渡点，移动到目标点 GP1，其速度为 100mm/s，轨迹平滑度为 2.0。

例 3：使用加速度

MoveL VL=100 PL=9.0 T=1

MoveC VL=GR0 PL=6.0 Point=2 Acc= 1 T=1

MoveC VL=GR0 PL=6.0 Point=3 Acc= 1 T=1

将工具 T1 的工具中心点从当前位置开始以固定的姿态圆弧线路径移动到指令中含有 Point=3 元素存储的指定位置，其速度为 GR0 储存数据，轨迹平滑度为 6.0，圆弧过渡点储存在含有 Point=2 的指令中，圆弧终点储存在含有 Point=3 的指令中。

例 4：参考用户坐标系

MoveL VL=100 PL=9.0 T=1

MoveC GP0 VL=100 PL=2.0 Point=2 Acc= 1 T=2 U=2

MoveC GP1 VL=100 PL=2.0 Point=3 Acc= 1 T=2 U=2

将工具 T2 的工具中心点从当前位置开始以固定的姿态沿圆弧路径移动到 GP1，在关于 U2 的用户坐标系下的指定位置，其中过渡点位置为 GP0，目标点位置为 GP1，速度为 100mm/s，轨迹平滑度为 2.0，加速度为 1。

例 5：使用偏移变量

- **方法 1：**

图 1.3.5

MoveC GP0 VL=100 PL=2.0 Point=2 Dec=5 T=2 U=1 Offset GP3 User

MoveC GP1 VL=100 PL=2.0 Point=3 Dec=5 T=2 U=1 Offset GP3 User

将工具 T2 的工具中心点从当前位置开始以固定的姿态沿圆弧路径移动到 GP1，在关于 U1 的用户坐标系下偏移位置为 GP3 的指定位置，其过渡点位 GP0，目标点位 GP1，速度为 100mm/s，轨迹平滑度为 2，减速度为 5。

- **方法 2：**

使用附加功能，自定义文本设置偏移量。如下图所示

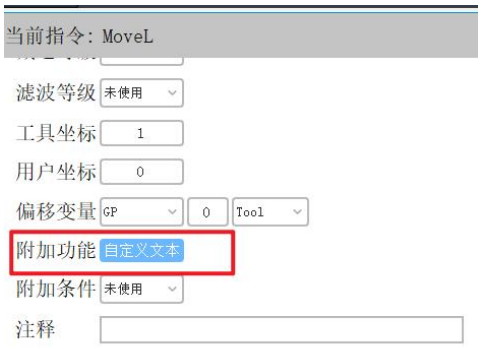


图 1.3.6

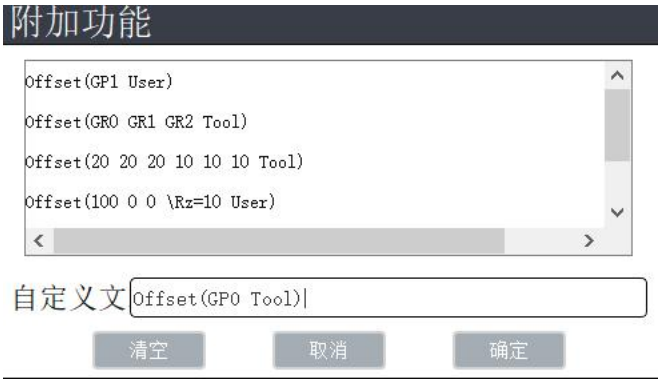


图 1.3.7

MoveC VL=100 PL=0.0 T=1 Offset 100 0 0 \Rz=10 T00l Ex1=80

指令效果为将当前点位按照偏移量(100 0 0 \RZ=10，Ex1=80)，沿着工具坐标系 1 的 X 正方向移动 100mm，Y、Z 方向不偏移，最后绕 Z 轴正方向旋转 10°，外部轴 1 向正方向偏移 80(单位根据外部轴定义确定是°还是 mm)，获得最终点位。

总结：

1、单点偏移可以通过指令中[偏移变量]进行设置，或者通过[附加功能]进行自定义文本输入。

2、单点偏移可以和整体偏移进行叠加，使用时请注意。

1.4. MoveAbsJ ---绝对关节运动

1.4.1. 指令用法

MoveAbsJ VJ=100 PL=9.0

执行该指令时机器人运动将不会受到指令中的工具坐标和用户坐标影响，机器人和外部轴将运动到一个以轴位置定义的目标位置（关节角度为参数），机器人各轴将以设定的百分比速度运行到目标位置。所有轴均同时运动，但机器人运动轨迹是一条不确定的曲线。

注意：

指令存储位置为关节位置（J1、J2...J6），机器人各轴关节将按照指令中关节位置运行到目标位置，且位置不受工具/用户坐标号影响。

插入运动指令需在伺服上电状态，且安全开关处于二档状态才能记录点位，指令插入成功。

1.4.2. 程序运行

运行该指令、该指令执行的动作不受工具坐标和用户坐标影响，机器人各轴将同步运动到以轴位置定义的目标位置（关节角度为参数）。当机器人工具末端到达目标位置（目标位置与指令中 PL 参数有关系），才会执行下一条指令。一般用于没有轨迹要求的场合。

1.4.3. 可变元素

MoveAbsJ [\Topoint] VJ=[Speed] PL=[zone] [\ACC] [\Dec] T=[Tool] [\S] [\signal]

1. [\Topoint]

变量类型：GJ（全局变量）、LJ（局部变量）。

解释：机器人和外部轴的目标点位置，可以选择使用 GJ、LJ 变量进行存储。当选择变量时，机器人目标点位置储存在变量中。不选择变量时机器人目标位置储存在指令中。

2. VJ=[Speed]

数据类型： float（范围：0-100）

变量类型：GR（全局变量）、LR（局部变量）

解释：其速度以最大速度为基准的百分比速度。可以选择使用 GR、LR 变量。当使用变量时，变量中的值作为机器人运行速度的百分比。

3. PL=[zone]

数据类型：float（范围：0.0-9.0）

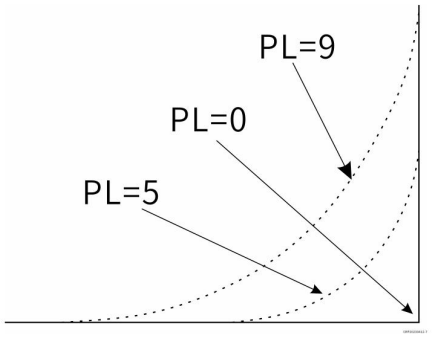


图 1.4.1

解释：机器人在运动轨迹中到达目标点的逼近等级。描述了所生成拐角的路径大小。逼近等级设置越小，机器人运行轨迹时生成拐角路径越小。

4. [\ACC]

数据类型：Int（范围：1-20）

解释：加速度，该加速度与机器人运动方式有关。加速等级设置越小机器人加速越快，加速等级设置越大加速越慢。默认不使用。

5. [\Dec]

数据类型：Int（范围：1-20）

解释：减速度，该减速度与机器人运动方式有关。减速等级设置越小机器人减速越快，减速等级设置越大减速越慢。默认不使用。

6. [\S]

数据类型：Int（范围：0-9）

解释：滤波等级共分为 0-9 级，默认为 0。滤波等级主要用于小距离、速度慢，可将等级改大，提高加减速能力与速度。滤波等级会影响节拍，滤波等级越大，时间越短。参数不合适可能造成机器人末端抖动，所以请根据实际情况调节。下图为滤波等级的一个速度示意图。

7. [\SP]

该参数需要 CrobotpOS 版本要求 1.13.0 及以上，低于此版本的软件请按要求进行升级，否则功能不能正常。

数据类型：Int（范围：10-300）

解释：滤波时间百分比，默认为 100。滤波原理同上，此处用百分比表示滤波时间倍率，允许突破参数设置的滤波时间上限。下图为滤波百分比的一个速度示意图。

注意：指令里面滤波等级只针对步长滤波有效。

8. T=[Tool]

数据类型:Int (范围：0-49)

变量类型：常数、UI、GIN、GOT

解释：机器人移动时使用的工具坐标系。0 号工具默认工具坐标。当使用变量时，变量中的值作为该点位对应工具坐标系。此处的主要目的在于获取已设定于工具坐标系中的负载信息，以便于后续的机器人运动学计算使用。

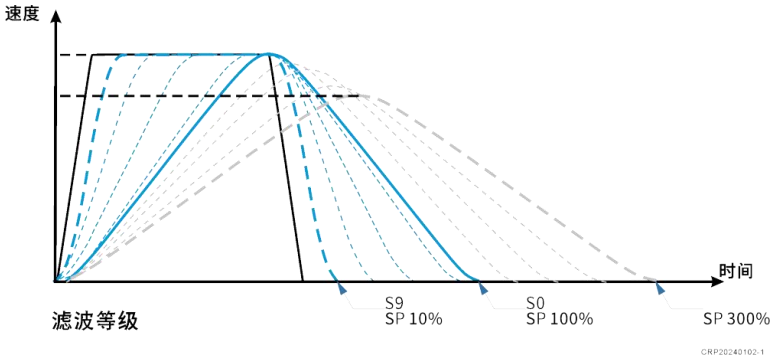


图 1.4.2

9. [\signal]

变量类型:Until（X、SX、M、SM、T、C、UI）

解释：当条件满足，该指令行停止执行，转入下一行执行。否则执行当前行直到结束后，转入下一行。可选择使用该元素。默认不使用。

1.4.4. 详细举例

例 1:

MoveAbsJ VJ=100 PL=9.0

机器人将沿非线性路径移动到指令中存储的指定绝对位置，其速度百分比为 100%，轨迹平滑度为 9.0。

例 2:

MoveAbsJ GJ0 VJ=100 PL=2.0

机器人将沿非线性路径移动到绝对位置 GJ0，其速度百分比为 100%，轨迹平滑度为 2.0。

例 3:

MoveAbsJ VJ=GR0 PL=6.0 Acc= 1

机器人将沿非线性路径移动到指令中存储的绝对位置处，其速度百比用储存在变量 GR0，轨迹平滑度为 6.0，加速度为 1。

例 4:

MoveAbsJ GJ0 VJ=100 PL=2.0

机器人将沿非线性路径移动到绝对位置 GJ0，其速度为 100mm/s，轨迹平滑度为 2.0。

例 5:

MoveAbsJ GJ0 VJ=100 PL=2.0 Dec=5

机器人将沿非线性路径移动到绝对位置 GJ0，其速度百分比为 100%，轨迹平滑度为 2.0，减速度为 5。

例 6:

MoveAbsJ GJ0 VJ=100 PL=2.0 Acc=10 Dec=5 Until X0=ON

机器人将沿非线性路径移动到绝对位置 GP0，其速度百分比为 100%，轨迹平滑度为 2.0，减速度为 5。运行过程中，若 X0 的值等于 ON，则直接停止运行该行指令，然后直接进入下一行运行；如果 X0 的值不等于 ON，则将该行指令运行完成后再进入下一行运行。

1.5. MoveJump ---门型运动

1.5.1. 指令用法

MoveJump VL=100 30 20 10 PL=9.0 T=1

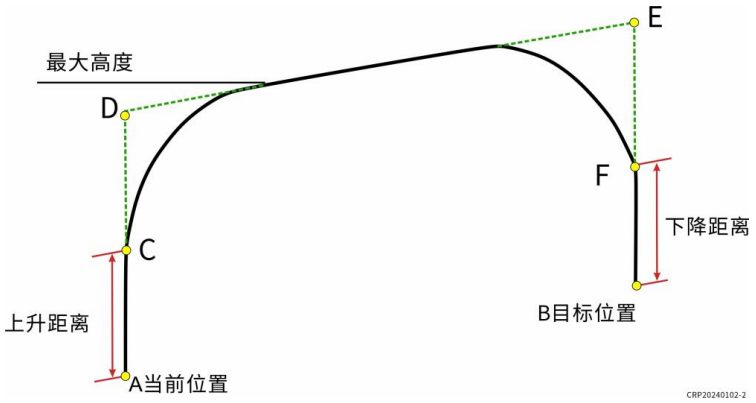


图 1.5.1

执行该指令时机器人将以门型的运动轨迹运行到目标点。

1.5.2. 程序运行

运行该两行指令时机器人以恒定的速度，固定的姿态，以当前位置为起点沿门型轨迹运动到目标点。当机器人工具末端到达目标位置（目标位置与指令中 PL 参数有关系），才会执行下一条指令。一般用于搬运的场合。

1.5.3. 可变元素

MoveJump [\Topoint] [\ID] VL=[Speed] PL=[zone] [\threshold] [\rise] [\decline] [\ACC] [\Dec] [\S] T=[Tool] [\U]

1. [\Topoint]

变量类型：GP（全局变量）、LP（局部变量）。

解释：机器人和外部轴的目标点位置。可以选择使用 GP、LP 变量。当选择变量时，机器人目标点位置储存在变量中。不选择变量时机器人目标位置储存在指令中。

2. [\ID]

数据类型：Int (范围：1-99)

解释：机器人 ID 号，协同同步时必须使用，在其他任何时候都不允许使用。所有协同同步任务时，ID 号必须相同，ID 号确保协同运行时不混淆，默认不使用。

3. VL=[Speed]

数据类型： float（范围：0-4095）

变量类型：GR（全局变量）、LR（局部变量）

解释：其速度以最大速度为基准的百分比速度。可以选择使用 GR、LR 变量。当使用变量时，变量中的值作为机器人运行速度的百分比。

4. PL=[zone]

数据类型：float（范围：0.0-9.0）

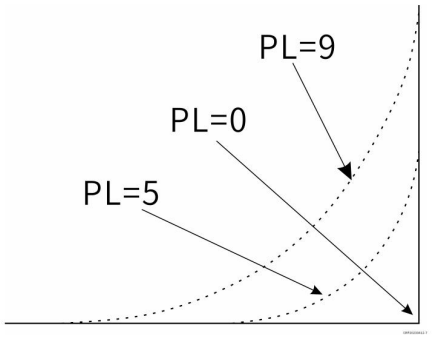


图 1.5.2

解释：机器人在运动轨迹中到达目标点的逼近等级。描述了所生成拐角的路径大小。逼近等级设置越小，机器人运行轨迹时生成拐角路径越小。

5. [\threshold]

数据类型：Int (范围：0-99999)

解释：门限高度。如图 1.5，若 A 点 Z 值比 B 点 Z 值大，门限高度则为 AD 的距离，即运动门型指令时，A 点 Z 方向向上运动的最大距离。

注意：门限高度需要大于等于 0，当门限高度为 0 时，从起始点到目标点是直线运动。对于 scara 机器人，门限高度最大不大于滚珠丝杆的向上运动的距离。

6. [\rise]

数据类型：Int (范围：0-门限高度)

解释：上升高度。如图 1.5，上升高度为 A 点向上直线运动的距离，即 AC 段距离。

注意：上升高度需要小于等于门限高度。

7. [\decline]

数据类型：Int (范围：0-门限高度)

解释：下降高度。如图 1.5，下降高度为平滑过渡结束点向下直线运动到 B 点的距离，即 FB 段距离。

注意：下降高度需要小于等于门限高度。

8. [\ACC]

数据类型：Int（范围：1-20）

解释：加速等级，该加速度与机器人运动方式有关。加速等级设置越小机器人加速越快，加速等级设置越大加速越慢。默认不使用。

9. [\Dec]

数据类型：Int（范围：1-20）

解释：减速等级，该减速度与机器人运动方式有关。减速等级设置越小机器人减速越快，减速等级设置越大减速越慢。默认不使用。

10. [\S]

数据类型：Int（范围：0-9）

解释：滤波等级分为 0-9 级，默认为 0。滤波等级主要用于小距离、速度慢，可将等级改大，提高加减速能力与速度。滤波等级会影响节拍，滤波等级越大，时间越短。参数不合适可能造成机器人末端抖动，所以请根据实际情况调节。图 1.5.3 为滤波等级的一个速度示意图。

11. [\SP]

该参数需要 CrobotpOS 版本要求 1.13.0 及以上，低于此版本的软件请按要求进行升级，否则功能不能正常。

数据类型：Int（范围：10-300）

解释：滤波时间百分比，默认为 100。滤波原理同上，此处用百分比表示滤波时间倍率，允许突破参数设置的滤波时间上限。图 1.5.3 为滤波等级的一个速度示意图。

注意：指令里面滤波等级只针对步长滤波有效。

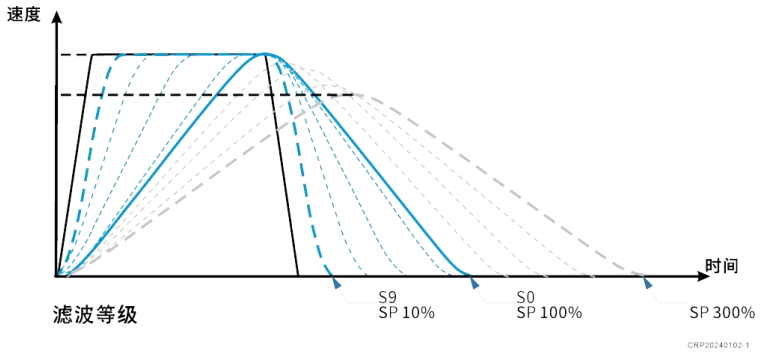


图 1.5.3

12. [\U]

数据类型：Int (范围：0-49)

变量类型：常数、UI、GIN、GOT

解释：机器人移动时使用的用户坐标系。0 号用户坐标为默认用户坐标系。可选择使用该元素。如果使用协同的外部轴时必须使用该参数。当使用变量时，变量中的值作为该点位对应用户坐标系进行反解点位。

13. [\U]

数据类型：Int (范围：0-49)

变量类型：常数、UI、GIN、GOT

解释：机器人移动时使用的用户坐标系。0 号用户坐标为默认用户坐标系。可选择使用该元素。如果使用协同的外部轴时必须使用该参数。当使用变量时，变量中的值作为该点位对应用户坐标系进行反解点位。

1.5.4. 详细举例

例 1:

MoveJump VL=100 30 20 10 PL=9.0 T=1 //起点为 A，目标点为 B

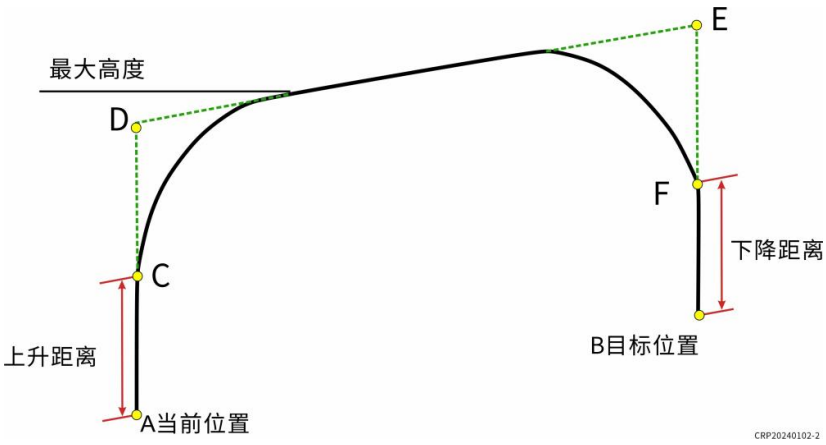


图 1.5.4

执行程序器人将以 100mm/s 的速度，从 A 点以直线运动向上运动 20mm 到 C 点，开始平滑过渡运动到最大高度(若 A 点的 Z 方向值大于等于 B 点的 Z 方向值，那么最大高度就是 A 点的 Z 方向值加上门限高度 30mm，若 A 点的 Z 方向值小于 B 点的 Z 方向值，那么最大高度就是 B 点的 Z 方向值加上门限高度 30mm)。继续直线运动，然后平滑过渡到 F 点，从 F 点以直线运动向下运动 10mm 到达 B 目标点。

例 2:

MoveJump GP1 VL=2000 100 50 50 PL=9.0 S=9 T=1

机器人以 2000mm/s 的速度从起始点以门型轨迹运动到目标点 GP1，门限高度 100mm，上升高度 50mm，下降高度 50mm，轨迹平滑度为 9.0，滤波等级为 9，使用工具 1。

1.6. 综合应用示例

运动指令插入说明:

- **方法一：**在示教模式下，伺服上电状态，设置好机械臂的坐标系模式后使用右侧各轴相对应的“+/-”按键调整机械臂位姿到所需的目标点位；点击插入所需运动指令并编辑相关**指令参数后，按下安全开关使其处于二档状态后点击“确认插入”。**
- **方法二：**在示教模式下，伺服上电状态，点击插入所需运动指令，编辑相关指令参数时使用位置变量对机械臂的目标点进行设定，按下安全开关使其处于二档状态后点击“确认插入”。

注：运动指令中的位置变量需提前设置好点位数据，也可通过查看点位进行修改。

编程示例:

机械臂从原点开始进行 5 次运动，具体如下:

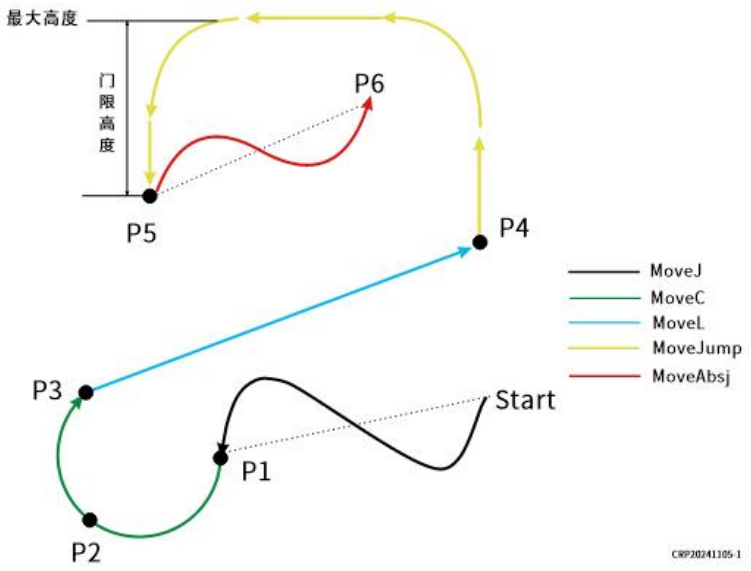


图 1.6.1

1、以关节运动，速度为 5 mm/s 的方式运动到点 1;

2、以圆弧运动，速度为 180 mm/s 的方式运动到点 2;

3、以直线运动，速度为 180 mm/s 的方式运动到点 3;

4、以门型运动（上升高度 50mm，下降高度 100mm，门限高度 200mm），速度 180 mm/s 的方式运动到点 4;

5、以绝对关节运动，速度为 5 的方式运动到点 5;

代码：

```
MoveJ VJ=5 PL=0.0 T=1 U=1

MoveC VL=180 PL=0.0 Point=2 T=1 U=1

MoveC VL=180 PL=0.0 Point=3 T=1 U=1

MoveL VL=180 PL=0.0 T=1 U=1

MoveJump VL=180 200.0 50.0 100.0 PL=0.0 T=1 U=1

MoveAbsJ VJ=5 PL=0.0 T=1 U=1
```

注意：

1. 编程时使用的工具坐标号必须是标定的工具号或者为 T=0（默认为法兰中心处），否则无法运行。同时选择工具号时，需要按照自己的需求，根据实际安装运行的工具来选择相应的工具号，否则也会产生错误。
2. 在选取偏移变量的时候也要注意偏移变量不能太大，避免超出可达范围，否则机器人会报错无法运行。

2. I/O 指令

2.1. DOut ---输出

2.1.1. 指令用法

```
DOut Y1=OFF
```

用于输出信号状态，将信号状态置位为 ON 或 OFF。

2.1.2. 程序运行

程序执行到当前行时将输出端口的信号状态量置位成指定状态。在信号获得新的状态时存在短暂的延时。

2.1.3. 可变元素

```
DOut [signal]
```

1. [signal]

变量类型：Y、M。

解释：输出信号的类型和地址。

2.1.4. 详细举例

例 1:

DOut Y1=OFF

将 Y1 置位为 OFF。

例 2:

DOut Y(UI0)=ON

将 Y（UI0）置位为 ON。Y 的信号口数据储存在变量 UI0 中。

例 3:

DOut M(UI0)=ON

执行到该行程序时将 M（UI0）置位为 ON。M 的地址号为变量 UI0 中的值。

例 4:

DOut M1=OFF

执行到该行程序时将 M1 置位为 OFF。

2.2. TDOut ---延时输出

2.2.1. 指令用法

TDOut Y1=OFF T100

通过延迟指定时间后改变输出信号状态，不影响程序向下执行。

2.2.2. 程序运行

执行到当前程序行时，后台自动开始计时，程序继续向下执行，当后台计时等于指令中设定时间时，将输出指令中设置的信号状态。

2.2.3. 可变元素

TDOut [signal] [time]

1. [signal]

变量类型：Y、M。

解释：输出信号的类型和地址。

2. [Time]

数据类型：Int(0-9999)变量类型：UI

解释：延时输出的延时时间。

2.2.4. 详细举例

例 1:

TDOut Y1=OFF T100

延时 100ms 后，将 Y1 置位为 OFF。

例 2:

TDOut Y(UI0)=ON T=200

延时 200ms 后，将 Y（UI0）置位为 ON。Y 的地址号为变量 UI0 的值。

例 3:

TDOut M1=ON T=100

执行到该行程序时,系统在后台延时 100ms 后将 M1 置位为 ON。执行过程中程序不停止继续向下运行。

例 4:

TDOut M(UI0)=OFF T=200

执行到该行程序时,系统在后台延时 200ms 后将 M(UI0)置位为 OFF。执行过程正程序不停止继续向下运行。M 的地址号存储在变量 UI0 中。

例 5:

TDOut M1=OFF T=(UI0)

执行到该行程序时,系统在后台延时 UI0 后将 M1 置位为 OFF。执行过程正程序不停止继续向下运行。延时时间为变量 UI0 的值。

2.3. Pulse ---脉冲输出

2.3.1. 指令用法

Pulse Y1=OFF T100

用于脉冲信号状态输出。

2.3.2. 程序运行

程序执行到当前行时，后台执行输出一个指令中设定时长的脉冲信号，在脉冲启动后程序不停止继续向下执行下一条指令。

2.3.3. 可变元素

Pulse [signal] [time]

1. [signal]

变量类型：Y、M。
解释：输出信号的类型及地址。

2. [Time]

数据类型：Int(0-9999)变量类型：UI
解释：脉冲输出持续时间，单位毫秒。

2.3.4. 详细举例

例 1:

Pulse Y1=OFF T100

输出持续 100ms 的将 Y1 置位为 OFF 的脉冲信号。

例 2:

Pulse M(UI1)=ON T100

输出持续 100ms 的将 M（UI0）置位为 ON 的脉冲信号。M 信号的地址为变量 UI1 的值。

例 3:

Pulse M(UI1)=ON T100

执行到该行程序时,系统在后台执行一个持续时间为 100ms 的将 M（UI1）置位为 ON 的脉冲信号。脉冲启动后程序不停止、继续向下运行。M 的地址号为变量 UI1 的值。

例 4:

Pulse M(UI1)=ON T=（UI0）

执行到该行程序时,系统在后台执行一个持续时间为 UI0 的将 M（UI1）置位为 ON 的脉冲信号。脉冲启动后程序不停止继续向下运行。M 的地址号为变量 UI1 的值，脉冲持续时间为比变量 UI0 的值。

2.4. Wait ---等待

2.4.1.指令用法

Wait X0==ON DT=0 CT=0

用于程序等待，直到预定信号出现，程序继续向下执行。

2.4.2. 程序运行

程序执行到当前行时，等待信号满足指令中预定状态后，信号持续时间满足设定时间后，程序继续向下执行。

2.4.3. 可变元素

Wait [Signal] DT=[time] CT=[time]

1. [Signal]

变量类型：X、SX、Y、SY、M、SM、T、C

解释：输入的信号类型、地址以及等待的时间。

2. DT=[Time]

数据类型：Int(0-9999)

变量类型：UI

解释：允许等待信号的时间，超时则向后执行。

3. CT=[Time]

数据类型：Int(0-9999)

变量类型：UI

解释：检测信号的持续时间，信号持续时间到了，程序向下执行。

2.4.4. 详细举例

例 1:

Wait X0==ON DT=0 CT=0

持续等待 X0 信号输入为 ON，不检测信号持续时间，检测到信号后向下执行。

例 2:

Wait X0==ON DT=0 CT=100

持续等待 X0 信号输入为 ON,并信号持续 100ms 后程序向下执行。

例 3:

Wait X0==ON DT=100 CT=0

执行到该行程序时,等待 X0 置位为 ON 时，X0 置位为 ON 后，程序立即向下执行，允许的最长等待时间为 100ms。如果超过允许等待最长时间后， X0 仍不满足设定信号，程序仍继续向下执行。

例 4:

wait X0==ON DT=100 CT=UI1

执行到该行程序时,等待 X0 置位为 ON 时，X0 置位为 ON 的持续时间为 UI1 后，程序继续向下执行，允许的最长等待时间为 100ms。如果超过允许等待最长时间后， X0 仍不满足设定信号，程序仍继续向下执行。持续时间为变量 UI1 的值。

例 5:

Wait M（UI0）==ON DT=0 CT=100

执行到该行程序时,持续等待 M（UI0）置位为 ON 并持续 100ms 后，程序继续向下执行。M 信号地址为变量 UI0 的值。

2.5. Aout ---模拟量输出

2.5.1. 指令用法

AOut AV1=5.000

用于模拟量信号数据输出。

2.5.2. 程序运行

程序执行到当前行时将在指定的模拟量输出端口输出一个指定的模拟量信号。在信号口获得新的状态时存在短暂的延时。

2.5.3. 可变元素

AOut [Signal]

1. [Signal]

变量类型：AV

解释：模拟量输出端口。

2.5.4. 应用举例

例 1:

AOut AV1=5.000

执行到该行程序时将输出模拟量 AV1 为 5.000。

例 2:

AOut AV1=GR2

执行到该行程序时将输出模拟量 AV1，AV1 的值为变量 GR2 的值。

例 3:

AOut AV(UI1)=5.000

执行到该行程序时,模拟量端口 AV（UI1）输出模拟量为 5.000。其中 AV 的地址号为变量 UI1 的值。

例 3:

AOut AV(UI2)=GR3

执行到该行程序时,模拟量端口 AV（UI2）输出模拟量为 GR3。其中 AV 的地址号为变量 UI2 的值。模拟量的值为变量 GR3 的值。

3. 控制指令

3.1. If ---判断

3.1.1. 指令用法

判断条件是否满足，执行不同指令。

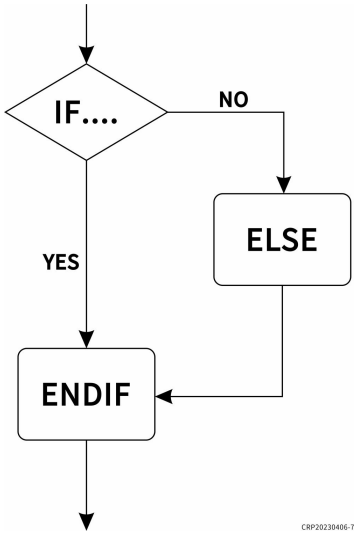


图 3.1.1

3.1.2. 程序运行

程序执行时将依次判断是否满足条件，直到满足其中一个条件。如果条件满足将继续执行程序。如果不满足任何条件，将通过符合 Else 的指令继续执行程序。如果满足多个条件，仅执行与第一个此类条件相关的指令。

3.1.3. 可变元素

IF [Conditional] [statement list]

[ElseIf]

[Else]

EndIf

1. [Conditional]

变量类型：X、SX、Y、SY、SM、M、T、C、GI、LI、GR、LR、UI 解释：用于判断的条件。

2. [statement list]

解释：执行的语句或指令。

3. [ElseIf]

解释：在 IF 判断语句中，当上一个条件不满足时，执行的下一段判断的指令。

4. [Else]

解释：在 IF 判断语句中，当判断条件不满足时，执行的下一段语句或指令。

3.1.4. 详细举例

例 1：

```
If GI0 > 100
    Dout Y0=ON
ElseIf GI0 < 0
    Dout Y1=OFF
EndIf
```

当满足 GI0 信号的值大于 100 时，输出 Y0 状态为 ON，If 判断语句结束。如果 GI0 小于或等于 100 则判断 GI0 是否小于 0，如果 GI0 小于 0 则输出 Y1 为 OFF，ElseIf 判断语句结束。如果 GI0 大于等于 0，则语句结束。

例 2：

```
If GI0 > 100
```

```
    Dout Y0=ON
ElseIf GI0 < 0
    Dout Y1=OFF
Else
    Dout Y2=OFF
EndIf
```

当满足 GI0 信号的值大于 100 时，输出 Y0 状态为 ON，If 判断语句结束。如果 GI0 小于或等于 100 则判断 GI0 是否小于 0，如果 GI0 小于 0 则输出 Y1 为 OFF，If 判断语句结束。如果 GI0 大于等于 0，则输出 Y2 为 OFF。

例 3:

```
If M0==ON AND (M1==ON OR M2==ON)
    Dout Y0=ON
Else
    Dout Y1=OFF
EndIf
```

当满足 M0 信号的值为"ON"并且 M1 或者 M2 其中一个信号及以上为"ON"时，输出 Y0 状态为 ON，If 判断语句结束。否则输出 Y1 状态为 OFF，If 判断语句结束。

3.2. While---条件循环

3.2.1. 指令用法

While [conditional expression]

[statement list]

EndWhile

在 While 指令块中当条件表达式满足时就执行指令块中的指令表或语句表达式，如果指令块中条件表达式不满足，则直接执行指令块后的程序。

注意：在【statement list】中需包含对【conditional expression】中判断循环是否继续的变量改变的语句，否则循环无限执行。

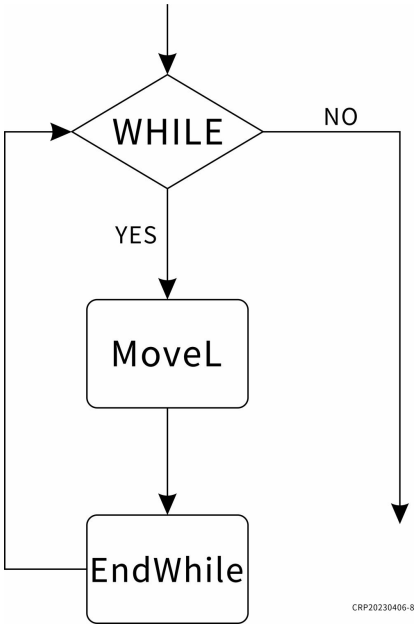


图 3.2.1

3.2.2. 程序运行

程序执行时将循环判断条件表达式的值是否满足，满足后就执行 While 循环内的指令或表达式，如果不满足则不执行。

3.2.3. 可变元素

While [Conditional] [statement list]

EndWhile

1. [Conditional]

变量类型：X、SX、Y、SY、SM、M、T、C、GI、LI、GR、LR、UI 解释：用于判断的条件。

2. [statement list]

解释：执行的语句或指令。

3.2.4. 详细举例

例 1：

While X0==ON

GI0 = GI0 + 2

EndWhile

当满足 X0 信号的值为 ON 时，重复执行 While 内 GI0 加 2 后将值赋值给 GI0 的语句。当 X0 信号的值不为 ON 时循环结束。

例 2：

While GI0 < 10

GI0 = GI0 + 2

EndWhile

当满足 GI0 小于 10 时，重复执行 While 内 GI0 加 2 后将值赋值给 GI0 的语句。当 GI0 大于等于 10 时循环结束。

3.3. Switch---条件选择

3.3.1. 指令用法

Switch 为条件选择指令，由 SWITCH、CASE、BREAK、DEFAULT 和 ENDSWITCH 构成一个完整的指令结构。

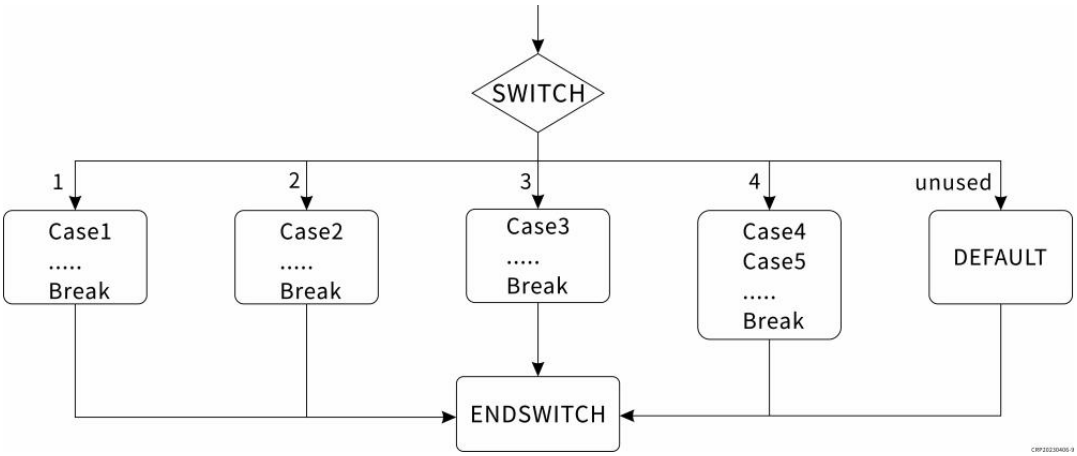


图 3.3.1

SWITCH 根据表达式或数据的值，判断该值和哪个 CASE 的值相等，找到相等的 CASE 后，程序从该位置开始执行，执行到第一个 BREAK 结束，并跳转到 ENDSIWITCH 行。即 Swicth 指令可以根据所指定表达式或数据的值，选择执行不同指令表或语句表达式。

3.3.2. 程序运行

执行指令块时将对比 Switch 中的数据是否与 Case 条件中的值一致，如果一致则执行相关 Case 后的指令表或语句表达式，执行完成后通过 Break 指令，跳转到程序块中 EndSwitch 位置，结束指令块执行，继续执行下一行程序，如果 Case 条件后不带 Break 指令，指令块将会继续执行下一个 Case 分支。

如果 Switch 中的数据是否与 Case 条件中的不一致，如果指令块中含有 Default 语句，则执行 Default 后的指令表或语句表达式，执行完成后通过 EndSwitch 指令，继续执行程序。若没有 Default 语句，则直接执行 EndSwitch 指令，继续执行程序。

3.3.3. 可变元素

```
Switch [Var]

CASE [Value1]

[\Break]

[\Default]

EndSwitch
```

1. [Var]

变量类型：UI、GIN、GOT

解释：用于调用程序块中分支的变量。

2. CASE [Value1]

解释：指令块中被调用的程序分支，可以任意添加删减。Switch 指令块中至少包含一个 Case 分支。

3. [\Break]

解释：直接到达 EndSwitch 指令。Break 不是必须存在指令块中。如果指令中无 Break，一旦 case 相应的值成功，那么将会无条件地继续向下执行其他 case 中的语句，直到遇到 break 语句或者到达 Endswitch 指令，才会结束条件选择语句。

4. [\Default]

解释：Default 不是必须存在指令块中若所有 case 条件都不满足条件，则执行 default，如果后面有 case 语句并执行 default 语句之后的 case 语句，直到 break 或 Endswitch 指令。

3.3.4. 详细举例

例 1:

```
Switch UI0
 Case 1
  Dout Y0=ON
  Break
 Case 2
  Dout Y1=ON
  Break
EndSwitch
```

当 UI0 的值为 1 时，执行 Case1 指令，将 Y0 设置为 ON。完成该指令后，通过 Break 语句跳出当前 Switch 结构，从而结束指令块的执行。若在后续执行过程中 UI0 的值变为 2，则执行 Case2 指令，将 Y1 设置为 ON，并直接跳转至 EndSwitch，以终止 Switch 结构内的进一步处理，结束指令块的执行。

例 2:

```
Switch UI0
 CASE 1
  Dout Y0=ON
  Break
 CASE 2
  Dout Y1=ON
  Break
 Default
  Dout Y2=ON
  Break
EndSwitch
```

当 UI0 的值为 1 或 2 时，将分别触发 Case1 或 Case2 中定义的逻辑流程。若在执行过程中，UI0 的值发生变化且不再等于 1 或 2，则系统将跳转至 Default 分支，并在此分支内执行相关指令，最终将 Y2 设置为 ON 状态。

ⓘ提示

1. 建议每个 case 分支（包括 default 分支）以 break 语句结束。如果某个 case 分支末尾缺少 break 语句，则程序将不会跳出当前 case，而是继续执行后续的 case 分支代码，直至遇到 break 语句或到达整个 switch 结构的末尾。
2. default 分支是可选的。当 switch 语句中包含 default 分支且该分支位于所有条件选择之后时，应在 default 分支的内容后添加 break 语句；若整个 switch 语句不包含 default 分支，即仅由一系列 case 构成，则最后一个 case 分支也必须以 break 语句结尾，以确保程序正确地退出 switch 结构。

3.4. Time ---延时

3.4.1. 指令用法

Time 100

程序延时等待 100ms。

3.4.2. 程序运行

程序运行时，当程序执行到该行程序后，程序开始延时等待，直到到达指令设定的时间后，程序继续向下执行。

3.4.3. 可变元素

Time [Time]

1. [time]

变量类型：C、UI

解释：延时等待设定的时间

3.4.4. 详细举例

例 1:

Time 100

执行到该行程序时程序延时等待 100ms 后继续向下执行。

例 2:

Time UI0

执行到该行程序时程序延时等待 UI0 后继续向下执行。延时等待时间储存在变量 UI0 中。

3.5. Pause---暂停

3.5.1. 指令用法

用于暂停正在运行的程序，分条件暂停和直接暂停。

3.5.2. 程序运行

程序运行时，当程序执行到该行程序后，直接暂停程序运行。如果指令后跟随判断条件，若条件满足则程序暂停运行，若条件不满足则继续运行程序，不执行暂停指令。

3.5.3. 可变元素

Pause [\verdict]

1. [\verdict]

可选类型：If

解释：用于判断是否执行暂停指令

3.5.4. 详细举例

例 1:

Pause If X0==ON

执行到该行程序时如果 X0 的值为 ON 则程序暂停运行、否则程序继续向下执行。

例 2:

Pause If GI0 > 5

执行到该行程序时如果 GI0 的值大于 5 则程序暂停运行、否则程序继续向下执行。

3.6. Jump---跳转

3.6.1. 指令用法

用于程序中的跳转，分条件跳转和直接跳转。使用跳转指令时必须和“*”跳转标识符指令配套使用。

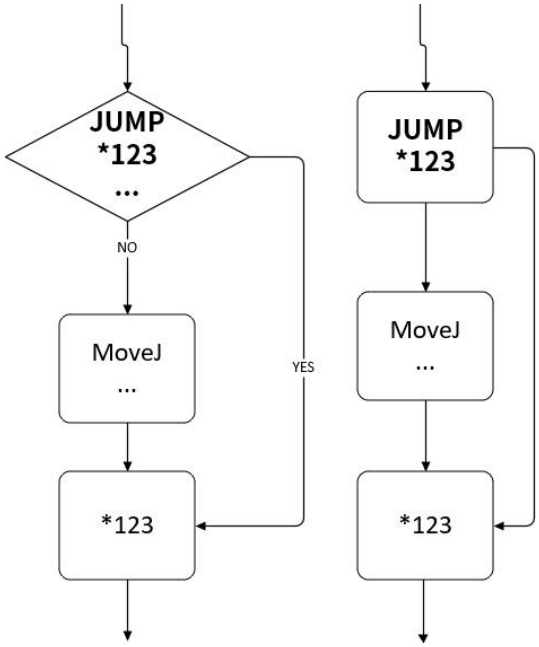


图 3.6.1

.....

Jump * ABC

.....

* ABC

程序执行到该指令行时，直接跳转到带有*ABC 的跳转目标标识符处继续向下执行程序。

3.6.2. 程序运行

程序运行时，当程序行执行到该行程序后，直接跳转到跳转目标标识符处继续执行程序。如果指令后跟随判断条件，若条件满足则程序跳转，若条件不满足则继续运行程序，不执行跳转指令。

3.6.3. 可变元素

Jump [target] [\verdict]

1. [target]

变量类型：float（数字，字母，汉字，最大 15 个汉字）解释：跳转目标标识符。必须和“*”的类型一致

2. [\verdict]

可选类型：If

解释：用于判断是否执行暂停指令。

3.6.4. 详细举例

例 1:

.....

Jump * ABC

.....

* ABC

程序执行到该指令行时，直接跳转到带有*ABC 的跳转目标标识符处继续向下执行程序。

例 2:

.....

Jump *ABC If X0==ON

.....

*ABC

执行到该行程序时如果 X0 的值为 ON 则程序跳转到带有*ABC 跳转目标标识符处继续向下执行程序。

例 3:

.....

Jump *ABC If GI0 >5

.....

*ABC

执行到该行程序时如果 GI0 的值大于 5 则程序跳转到带有*ABC 跳转目标标识符处继续向下执行程序。

3.7. *---跳转目标

用于程序跳转时的跳转目标，与 Jump 指令配合使用，详细信息参考 Jump 指令使用方法。

3.8. Call---子程序调用

3.8.1. 指令用法

程序执行过程中，将程序执行转移到另一个程序中执行并进行传参，分条件调用和无条件调用（传参功能可选用）。

```
Call robot/program/ABC.pro
```

执行到该行程序时直接调用程序名称为 ABC 程序继续执行。

3.8.2. 程序运行

程序运行时，当程序行执行到该行程序后，直接调用该程序行指定的程序并继续执行。如果指令后跟随判断条件，若条件满足则执行程序调用，若条件不满足则继续运行程序，不执行程序调用。

3.8.3. 可变元素

```
Call [Subroutine] [\Parameter] [\verdict]
```

1. [Subroutine]

解释：被调用的程序。

2. [\Parameter]

解释：参数设置，允许最大传入 9 个参数。ARG 变量作为间接变量，而 UI/GI/LI/GR/LR/S 作为直接变量，ARG 变量只能作为源值向直接变量赋值。Call 指令后面引用传参参数第一个对应 ARG0、第二个对应 ARG1 后面依此类推；以“Call robot/program/aaa.pro (123,3.14,"abc")”举例，123 参数对应 ARG0、3.14 对应 ARG1、abc 对应 ARG2;

3. [\verdict]

可选类型：If

解释：用于判断是否执行暂停指令

3.8.4. 详细举例

例 1:

Call robot/program/ABC.pro

执行到该行程序时直接调用程序名称为 ABC 程序继续执行。

例 2:

Call robot/program/ABC.pro If X0==ON

执行到该行程序时，如果 X0 的值为 ON 则调用程序名称为 ABC 程序。

例 3:

Call robot/program/ABC.pro If GI0 > 5

执行到该行程序时，如果 GI0 的值大于 5 则调用程序名称为 ABC 程序。

例 4:

- **主程序:**

.....

Call robot/program/ABC.pro (123,3.14,"avc")

.....

执行到该行程序时直接调用程序名称为 ABC 程序继续执行，并将参数 ARG0=123，ARG1=3.14，ARG2=“avc”传入子程序 ABC 中继续执行。

- **ABC 子程序:**

Set UI0=ARG0

Set GR0=ARG1

Set S0=ARG2

If UI>5

.....

EndIf

子程序执行后，UI0=123，GR0=3.14，S0="avc"。

3.9. Return---程序返回

3.9.1. 指令用法

结束子程序运行，返回主程序。分为条件程序返回和直接返回。

Return

执行到该行程序时，直接退出当前程序回到主程序继续运行。

3.9.2. 程序运行

当程序执行到该行程序后，直接退出子程序并回到主程序继续执行。如果指令后跟随判断条件，若条件满足则执行子程序返回，若条件不满足则继续运行程序，不执行子程序返回。

3.9.3. 可变元素

Return [\verdict]

1. [\verdict]

可选类型：If

解释：用于判断是否执行程序返回。

3.9.4. 详细举例

例 1:

.....

Return

执行到该行程序时，直接退出当前程序回到主程序继续运行。

例 2:

.....

Return If X0==ON

执行到该行程序时，如果 X0 的值为 ON 则结束当前程序并返回到主程序继续运行。

例 3:

.....

Return If GI0 > 5

执行到该行程序时，如果 GI0 的值大于 5 则结束当前程序并返回到主程序继续运行。

3.10. Trap---中断指令和 TrapEnd-结束指令

3.10.1. 指令用法

开始检测中断信号，中断信号触发时响应中断程序。切换不同模式时指令不会失效，只有执行 TrapEnd 指令时才会结束中断触发。

Trap X0==ON Call 测试； //中断指令，满足条件进入中断程序

..... //中间程序

TrapEnd ； //中断结束指令

3.10.2. 指令逻辑

检测到触发信号(上升沿\下降沿信号)后，立即中断当前运行的程序并减速停止 (记录下当前被中断的程序行)，然后开始调用中断程序。中断程序运行结束后，机器人回到中断位置并从中断程序行继续往下执行程序。若主程序未运行到中断结束指令，则向下执行主程序过程中，随时等待中断信号响应。

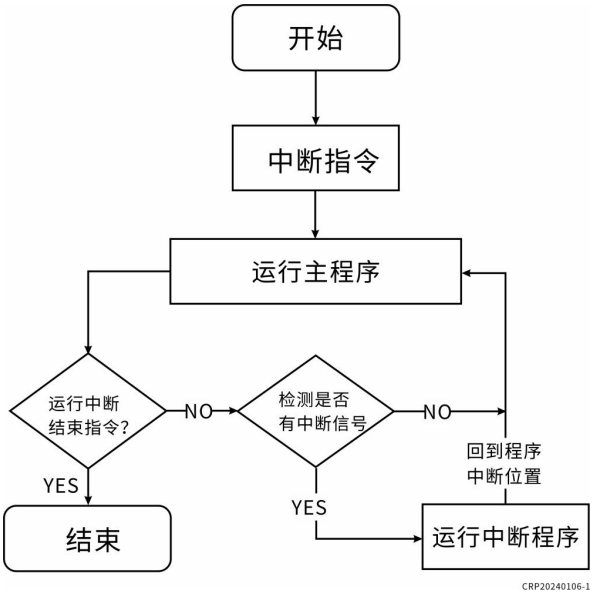


图 3.10.2

3.10.3. 程序运行

执行程序中，若检测到中断信号时，立即停止当前主程序行后，进入中断程序运行，直到中断程序运行完回到停止的主程序行继续向下运行程序行。若主程序未运行到中断结束指令，则在向下执行主程序时仍会等待中断信号响应。

```
Trap [A][B]==[C] Call [D]

...

TrapEnd
```

1. **[A]**

变量类型：状态变量。

解释：[A]可以选择 X/Y/M,即通用输入口/通用输出口/内部辅助继电器。

2. [B]

变量类型：整型变量

解释：[B]为[A]的变量号，如[A]选择 X(通用输入口)时，[A][B]即为 X<变量号>，根据实际情况选择数据，如选择 X0。

3. [C]

变量类型：状态变量。

解释：可以选择 ON 或 OFF。ON=1，代表上升沿信号；OFF=0，代表下降沿信号。

4. [D]

变量类型：字符变量。

解释：[D]为调用的中断程序名。

3.10.4. 详细应用举例

例 1:

```
Trap X0==ON Call 测试;        //检测到通用输入接口 X0 有上升沿信号后，调用名为“测试”的中断程序
MoveL VL=500mm/s PL=0.0 ;
MoveL VL=50mm/s PL=0.0 ;
Time 1000 ;
MoveL VL=50mm/s PL=0.0 ;
TrapEnd ;
```

开始检测通用输入接口 X0 是否有上升沿信号后，若有信号则调用名为“测 试”的中断程序，没有则继续向下执行运行指令，等待 X0 信号触发中断，执行到 TrapEnd 指令结束对 X0 的中断触发。

3.11. Note --- 注释

CrobotpOS 版本要求 1.13.0 及以上，低于此版本的软件请按要求进行升级，否则功能不能正常。

3.11.1. 指令用法

```
// 程序注释
```

注释对编写程序更易于理解。

3.11.2. 程序运行

执行该指令，不会产生其他影响。

3.11.3. 可变元素

输入参数 1

```
// [str]
```

1. **[str]**

解释：程序段内容注释，可任意内容。

3.11.4. 应用举例

```
//进入到 a 程序流程

Call ./a.pro

//进入到 b 程序流程

Call ./b.pro
```

3.12. Speed --- 全局倍率

CrobotpOS 版本要求 1.13.0 及以上，低于此版本的软件请按要求进行升级，否则功能不能正常。

3.12.1. 指令用法

```
Speed 50
```

用于设定全局速度倍率的指令。

3.12.2. 程序运行

执行该指令，会以当前指令的速度设置全局倍率。

3.12.3. 可变元素

Speed [Var]

输入参数 1

[Var]

数据类型：UInt（范围 1-100）

变量类型：常数、UI、GIN

解释：设置自动模式全局倍率为指令指定数值；

3.12.4. 应用举例

Speed 50

MoveL VL=100 PL=9.0 T=1

运行 MoveL 指令速度为 100mm/s×50%的倍率，最终运行速度为 50mm/s

4. 运算指令

4.1. ADD---加运算

4.1.1. 指令用法

Add GI1=GI2+5

用于数值、变量或者永久数据对象的加法运算。

4.1.2. 程序运行

执行此程序指令，对应的变量进行加运算，运算完成后并赋值到指定变量中储存。当数据类型不一致时，不能进行运算。

4.1.3. 可变元素

Add [Variable}=[AddValue]+[AddDvalue]

1. [Variable]

数据类型：GI/LI/GJ/LJ/GP/LP/GR/LR/UI 解释：数值变量或者永久数据

2. [AddValue]

数据类型：GI/LI/GJ/LJ/GP/LP/GR/LR/UI 解释：数值变量或者永久数据

3. [AddDvalue]

数据类型：GI/LI/GJ/LJ/GP/LP/GR/LR/UI 解释：数值变量或者永久数据

4.1.4. 详细举例

例 1:

Sub GR1=GR2+GR3

将 GR2 的值加上 GR3 计算完成后并赋值给 GR1

例 2:

Add GI1=GI2+5

将 5 增加到 GI2 中计算完成并赋值给 GI1

例 3:

Sub GP0.X=GP1.X+5

将 GP1 的 X 方向坐标值加上 5，计算完成后并赋值给 GP0 的 X 方向坐标。

例 4:

Add GP(UI1).X=LP3.X+LR2

将局部位置变量 LP3 的 X 方向坐标值增加变量 LR2 中的值。运算完成后将值赋值给全局位置变量 GP（UI1）的 X 方向坐标。GP 号为变量 UI1 的值。

例 5:

Add LP(UI1).Z=LP(UI2).Z+GI(UI3)

将局部变量 LP（UI2）的 Z 方向坐标值增加 GI（UI3）运算完成后赋值给变量 LP（UI1）的 Z 方向坐标。局部变量 LP 的变量号分别为变量 UI1、UI2 的值，全局变量 GI 的变量号为变量 UI3 的值。

例 6:

Add GJ5.J6=GJ5.J6+UI3

将机器人在 GJ5 位置 J6 轴的关节坐标值增加 UI3，运算完成后将值赋值给 GJ5 的 J6 轴的关节坐标。增加的数据为变量 UI3 的值。

4.2. Sub---减运算

4.2.1. 指令用法

Sub GI1=GI2-5 //减运算指令

用于数值变量或者永久数据对象减一个数值。

4.2.2. 程序运行

执行此程序行，对应的变量进行减运算，运算完成后并赋值到指定变量中储存。当数据类型不一致时，不能进行运算。

4.2.3. 可变元素

Sub [Variable}=[SubValue]-[SubDvalue]

1. [Variable]

数据类型：GI/LI/GJ/LJ/GP/LP/GR/LR/UI 数值变量或者永久数据

2. [SubValue]

数据类型：GI/LI/GJ/LJ/GP/LP/GR/LR/UI 解释：数值变量或者永久数据

3. [SubDvalue]

数据类型：GI/LI/GJ/LJ/GP/LP/GR/LR/UI 解释：数值变量或者永久数据

4.2.4. 详细举例

例 1：

Sub GR1=GR2-GR3

将 GR2 的值减去 GR3 计算完成后并赋值给 GR1

例 2:

Sub GI1=GI2-5

将 GI2 的值减去 5，计算完成后并赋值给 GI1

例 3:

Sub GP0.X=GP1.X-5

将 GP1 的 X 方向坐标值减去 5，计算完成后并赋值给 GP0 的 X 方向坐标。

例 4:

Sub GP1.X=GP2.X-5

将 GP2 的 X 方向坐标值减去 5，计算完成后并赋值给 GP1 的 X 方向坐标。

例 5:

Sub GP(UI1).J6=LP3.J6-UI2

将局部变量 LP3 的 J6 轴的坐标值减变量 UI2 的值。运算完成后将值赋值给全局位置变量 GP（UI1）的 J6 轴的坐标。GP 号为变量 UI1 的值。

例 6:

Sub LP(UI1).Z=LP(UI2).Z-GI(UI3)

将局部变量 LP（UI2）的 Z 方向的坐标值减少 GI（UI3）运算完成后赋值给 LP（UI1）的 Z 方向坐标。局部变量 LP 的变量号分别为变量 UI1、UI2 的值，全局变量 GI 的变量号为变量 UI3 的值。

例 7:

Sub GJ5.J6=GJ5.J6-UI3

将机器人在 GJ5 位置 J6 轴的关节坐标值减少 UI3，运算完成后将值赋值给 GJ5 位置 J6 轴的关节坐标。减少的数据存储在 UI3 中。

4.3. Mul---乘运算

4.3.1. 指令用法

Mul GI1=GI2*5

用于数值变量或者永久数据对象乘以一个数值。

4.3.2. 程序运行

执行此程序行，对应的变量进行乘法运算，运算完成后并赋值到指定变量中储存。当数据类型不一致时，不能进行运算。

4.3.3. 可变元素

Mul [Variable}=[MulValue]*[MulDvalue]

1. [Variable]

数据类型：GI/LI/GR/LR/UI 解释：数值变量或者永久数据

2. [MulValue]

数据类型：GI/LI/GR/LR/UI 解释：数值变量或者永久数据

3. [MulDvalue]

数据类型：GI/LI/GR/LR/UI 解释：数值变量或者永久数据

4.3.4. 详细举例

例 1:

Mul GI1=GI2*5

将 GI2 的值乘以 5 计算完成后并赋值给 GI1

例 2：

Mul LI(UI1)=GI3*GI(UI0)

将 GI3 的值乘以 GI（UI0）的值，计算完成后并赋值给 LI（UI1）。局部变量 LI 的变量号为变量 UI1 的值，全局变量 GI 的变量号为变量 UI0 的值。

例 3：

Mul GR(UI2)=LR(UI5)*GI(UI4)

将变量 LR（UI5）的值乘以 GI（UI4）的值，计算完成后将值赋值给 GR（UI2）。变量 GR 的变量号为变量 UI2 的值，变量 LR 的变量号为变量 UI5 的值，变量 GI 的变量号为 UI4 的值。

4.4. Div---除运算

4.4.1. 指令用法

Div GI1=GI2/5

用于数值变量或者永久数据对象除以一个数值。

4.4.2. 程序运行

执行此程序行，对应的变量进行除法运算，运算完成后并赋值到指定变量中储存。当数据类型不一致时，不能进行运算。

4.4.3. 可变元素

Div [Variable}=[DivValue]/[DivDvalue]

1. [Variable]

数据类型：GI/LI/GR/LR/UI 解释：数值变量或者永久数据

2. [DivValue]

数据类型：GI/LI/GR/LR/UI 解释：数值变量或者永久数据 [DivDvalue]

数据类型：GI/LI/GR/LR/UI 解释：数值变量或者永久数据

4.4.4. 详细举例

例 1:

Div GI1=GI2/5

将 GI2 的值除以 5 计算完成后并赋值给 GI1。

例 2:

Div LI(UI1)=GI3/GI(UI0)

将 GI3 的值除以 GI（UI0）的值，计算完成后并赋值给 LI（UI1）。局部变量 LI 的变量号为变量 UI1 的值，全局变量 GI 的变量号为变量 UI0 的值。

例 3:

Div GR(UI2)=LR(UI5)/GI(UI4)

将变量 LR（UI5）的值除以 GI（UI4）的值，计算完成后将值赋值给 GR（UI2）。变量 GR 的变量号为变量 UI2 的值，变量 LR 的变量号为变量 UI5 的值，变量 GI 的变量号为变量 UI4 的值。

4.5. Inc---加一运算

4.5.1. 指令用法

Inc GI1

用于数值变量对象加一运算。

4.5.2. 程序运行

执行此程序行，对指定变量进行加一运算，运算完成后并赋值到该变量中储存。

4.5.3. 可变元素

Inc [Variable]

[Variable]

数据类型：GI/LI/UI 解释：数值变量

4.5.4. 详细举例

例 1:

Inc GI1

将 GI1 的值进行加一计算完成后并赋值给 GI1，与 Add GI1=GI1+1 用法一致。

例 2:

Inc LI(UI0)

将 LI(UI0）的值进行加一计算完成后并赋值给 LI(UI0），与 Add LI(UI0）=LI(UI0）+1 用法一致。变量 LI 的变量号为变量 UI0 的值。

4.6. Dec---减一运算

4.6.1. 指令用法

Dec GI1

用于数值变量对象减一运算。

4.6.2. 程序运行

执行此程序行，对指定变量进行减一运算，运算完成后并赋值到该变量中储存。

4.6.3. 可变元素

Dec [Variable]

1. [Variable]

数据类型：GI/LI/UI 解释：数值变量

4.6.4. 详细举例

例 1:

Dec GI1

将 GI1 的值进行减一计算完成后并赋值给 GI1，与 Sub GI1=GI1-1 用法一致。

例 2:

Dec LI(UI0)

将 LI(UI0）的值进行减一计算完成后并赋值给 LI(UI0），与 Sub LI(UI0）=LI(UI0）+1 用法一致。变量 LI 的变量号为变量 UI0 的值。

4.7. Set---赋值指令

4.7.1. = 赋值功能

指令用法

用于数值变量或者永久数据对象赋值运算。

程序运行

执行此程序行，将“=”符号右侧的数据保存到“=”左侧并将原来的数据覆盖。

可变元素

Set [Variable]=[SetValue]

图 4.7.1

1. [Variable]

变量类型：GI/LI/GJ/LJ/GP/LP/GR/LR/UI/B/S/TOOL/USER

解释：数值变量，可选变量 GI/LI/GJ/LJ/GP/LP/GR/LR/UI/B/S/TOOL/USER

2. [SetValue]

变量类型：GI/LI/GJ/LJ/GP/LP/GR/LR/UI/B/S/TOOL/USER

数值变量或者永久数据，可选变量 GI/LI/GJ/LJ/GP/LP/GR/LR/UI/B/S/TOOL/USER，根据目标类型可选变量类型会有所差异

详细举例

例 1：

Set GI1=65536

将 65536 数值赋值给 GI1 变量。

例 2：

Set LP2.X=100

将 100 保存到局部位置变量 LP 的 X 方向数值中，并将原来 LP 中 X 方向的值覆盖。

例 3：

Set GP(UI2).Y=GP(UI3).Y

将全局位置变量 GP（UI3）中的 Y 值，保存到全局位置变量 GP（UI2）的 Y 值中并将原来的值覆盖。

4.7.2. Convert 数据类型转换功能

函数用法

Set GI1=Convert （Int GI0）

用于数值变量类型转换。

程序运行

执行此程序行，将指定变量数据类型转换为目标数据类型。

可变元素

Set [Variable]=Convert （[Var Tyte] [Variable] [Data Tyte] [\Length] ）

图 4.7.2

1. [Variable]

变量类型：GI/LI/GR/LR/UI/B/S

目标变量，将转换类型后的数值放入变量中。

2. [Var Typte]

变量类型：int,Float,Ushort,String

设置需要将变量数值转换的类型。

3. [Variable]

变量类型：GI/LI/GR/LR/UI/B/S

需要转换类型的变量。

4. [\Data Typte]

变量类型：Hex、Str

5. [\Length]

变量类型：UI，常数

当需要转换类型的变量（源值）为 B 变量时，需设置数据类型。可选择转换后的数据类型为字符串或十六进制数。

详细举例

例 1：

当选择转换后的数据类型为字符串时需要设置字符串长度。

Set GI1=Convert （Int GI0）

将 GI0 变量的值转换成整型（Int）后存放在 GI1 中。

例 2：

Set GI1=Convert （Int B8 Hex）

将 B8 变量中的值转换成整型（Int），以十六进制形式存放在 GI1 中。

例 3:

Set GR1=Convert （Float B4 Str 2）

将 B4变量中的值转换成浮点型（Float），以字符串的形式存放在 GR1 中，字符长度为 2。

例 4:

Set B(UI2)=Convert （Ushort B4 Str UI4）

将 B4 变量中的值转换成 16 位无符号整数，以字符串的形式存放在 B(UI2)中（取 UI2 变量的值作为 B 变量的变量号），字符长度为 UI4 的值。

例 5:

Set B3=Convert （String GR4）

将 GR4 变量中的值转换成字符串类型，存放在 B3 中。

4.7.3. Split 字符串拆分功能

函数用法

Set S3=Split （S1 ' , '）

将指定的字符串作为分隔符，用于将原字符串拆分成若干个子字符串，并返回一个包含分割结果的列表。（仅 S 变量使用）

可变元素

Set [Variable1]=Split （ [Variable2] [String] ）

图 4.7.3

1. [Variable1]

变量类型：S

拆分结果存放起始变量。

2. [Variable2]

变量类型：S

需要拆分的原字符串变量。

3. [String]

变量类型：string

用于拆分的字符

详细举例

例 1：

Set S2=Split（S1 ' , '）

将字符变量 S1 用“，”拆分字符串，以 S2 作为拆分结果的存放起始变量。

例：S1="1,2,3,4,5"，使用“,”进行拆分时，返回结果就会从赋给的变量作为起始变量，分别返回结果到 S2=1、S3=“2”、S4=“3”、S5="4"、S6="5"。

例 2：

Set S(UI3)=Split（S(UI1) ' - '）

将字符变量 S(UI3)用“-”拆分字符串，以 S(UI1)作为拆分结果的存放起始变量。

例：若 UI1=5，UI3=6，则 S5="0-9-6-4-7"，使用“-”进行拆分时，返回结果就会从赋给的变量作为起始变量，分别返回结果到 S6="0"、S7="9"、S8="6"、S9="4"、S10="7"。

4.7.4. CountDis 两点距离计算功能

函数用法

Set GR1=CountDis （GP2 GP3 ）

Set 指令的两点距离计算功能，当赋给变量选择 GR/LR 时可选“CountDis”函数，通过工具末端两点求取两点的距离。

可变元素

Set [Variable1]=CountDis （[Point1] [Point2] ）

图 4.7.4

1. [Variable1]

变量类型：GR/LR

计算结果存放变量。

2. [Point1]

变量类型：GP/LP

点 1 位置变量。

3. [Point2]

变量类型：GP/LP

点 2 位置变量。

详细举例

例 1:

Set GR1=CountDis （GP1 GP2 ）

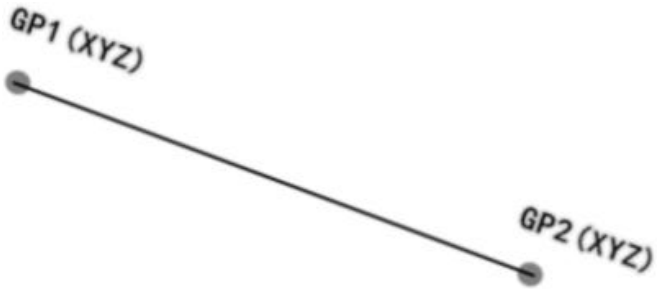


图 4.7.5

计算 GP1 与 GP2 的绝对距离，然后存放到 GR1 中。（点数据取“XYZ”进行计算，“Rx/Ry/Rz”及外部轴数据不参与计算。）

例 2:

Set GR(UI2)=CountDis （GP2 GP(UI4) ）

计算 GP2 到 GP(UI4)的绝对距离，然后存放到 GR(UI2)中。（点数据取“XYZ”进行计算，“Rx/Ry/Rz”及外部轴数据不参与计算。）

例：若 UI4=5，UI2=6，则计算 GP2 与 GP5 的绝对距离，并将数据存放在 GR6 中。

4.7.5. CRoboT 读取空间位姿函数

函数用法

CRobT(Current Robot Target) 用于读取机械臂和外部轴的当前位置。 该函数返回空间位姿（x、y、z、Rx、Ry、Rz）、机械臂轴配置（CF1、CF4、CF6、CFX）和外部轴位置（E1、E2、E3、E4、E5、E6）。

可变元素

Set [Variable1]=CRobT（ [Unit] [Tool] [User] ）

图 4.7.6

1. [Variable1]

变量类型：GP/LP

计算结果存放变量。

2. [Unit]

变量类型：R1/B1/S1

选择机械单元组。

3. [Tool]

变量类型：Number

工具坐标号选择。

4. [User]

变量类型：Number

用户坐标号选择。

详细举例

Set GP0=CRobT(R1 1 0)

将机械臂和外轴的当前位置储存在 GP0 中。工具 1 和用户 0 用于计算位置。

注意：如果需要机器人位置精度较高需要使机器人停稳后执行该指令，可通过该条指令的上一条运动指令使 PL=0 实现上述操作。

程序运行

将返回当前的机械单元组下机器人指定工具基于用户下的坐标值到 GP 里。

4.7.6. CJointT 读取当前关节角函数

函数用法

用于读取机械臂轴和外部轴的当前角度。该函数返回关节位置（J1、J2、J3、J4、J5、J6、J7）和外部轴位置（E1、E2、E3、E4、E5、E6）。

可变元素

Set [Variable1]=CJointT（[Unit]）

图 4.7.7

1. [Variable1]

变量类型：GJ/LJ

计算结果存放变量。

2. [Unit]

变量类型：R1/B1/S1

选择机械单元组。

详细举例

SetGJO=CJointT(R1)

将机械臂和外轴的关节角度储存在 GJ0 中。

注意：如果需要机器人位置精度较高需要使机器人停稳后执行该指令，可通过该条指令的上一条运动指令使 PL=0 实现上述操作。

程序运行

将返回当前的机械单元组下机器人关节值到 GJ 里。

4.7.7. CalcRobT 关节位置转空间位姿函数

函数用法

CalcRobT(Calculate Robot Target) 用于将机器人关节位置转换为空间位姿。 该函数返回关节位置（J1、J2、J3、J4、J5、J6、J7）和外部轴位置（E1、E2、E3、E4、E5、E6）。

可变元素

Set [Variable1]=CalcRobT （[Variable2] [Tool] [User])

图 4.7.8

1. [Variable1]

变量类型：GP/LP

计算结果存放变量。

2. [Variable2]

变量类型：GJ/LJ

指定用于计算的关节位置变量

3. [Tool]

变量类型：Number

工具坐标号选择。

4. [User]

变量类型：Number

用户坐标号选择。

详细举例

Set GP0=CalcRobT(GJ0 1 0)

将指定的关节变量 GJ0 关节位置基于工具 1 用户 0 计算出空间位姿返回到 GP0 中

程序运行

将给定的关节位置进行计算返回空间位姿。

4.7.8. CalcJointT 空间位姿转关节位置函数

函数用法

CalcRobT(Calculate Joint Target) 用于将机器人空间位姿转换为关节位置。 该函数返回空间位姿（x、y、z、Rx、Ry、Rz）、机械臂轴配置（CF1、CF4、CF6、CFX）和外部轴位置（E1、E2、E3、E4、E5、E6）。

可变元素

Set [Variable1]=CalcJointT （[Variable2] [Tool] [User]）

图 4.7.9

1. [Variable1]

变量类型：GJ/LJ

计算结果存放变量。

2. [Variable2]

变量类型：GP/LP

指定用于计算的位置变量

3. [Tool]

变量类型：Number

工具坐标号选择。

4. [User]

变量类型：Number

用户坐标号选择。

详细举例

Set GJ0=CalcJointT(GP0 1 0)

将指定的位置变量 GP0 空间位姿基于工具 1 用户 0 计算出关节位置返回到 GJ0 中

程序运行

将给定的空间位姿进行计算返回关节位置。

4.7.9. CalcCfg CFG 轴配置计算函数

函数用法

CalcCfg(Calculate Cfg)用于使用参考点计算更新目标点的 CFG 位置信息（目标点是通过加减运算得的点需要计算该姿态下的 CFG 信息）。

可变元素

Set [Variable1]=CalcCfg（[Variable2] [Coord. type] [Tool] [User]）

图 4.7.10

1. [Variable1]

变量类型：GP/LP

计算结果存放变量。

2. [Variable2]

变量类型：GP/LP/GJ/LJ

设定用于反解到关节的参考点

3. [Coord. type]

坐标类型选择，可选世界坐标、用户坐标、直角坐标

4. [Tool]

变量类型：Number

工具坐标号选择。

5. [User]

变量类型：Number

用户坐标号选择。

详细举例

Set GP1=CalcCfg(GP0 user 1 0)

将参考点 GP0 基于工具 1 和用户 0 的反解到关节，计算更新 GP1 的 CFG 轴配置信息

程序运行

会基于参考点的关节坐标计算更新目标点的 CFG 轴配置信息

4.7.10. Sin 计算正弦值

函数用法

Sin 用于计算一个角值的正弦值。输入的数值单位是角度。

可变元素

Set [Variable1]=Sin [Variable2]

4.7.11

1. [Variable1]

变量类型：GR/LR

计算结果存放变量。

2. [Variable2]

变量类型：GR/LR/AIN/AOT

用于计算正弦值的目标变量

返回值

范围【-1，1】

详细举例

Set GR1=Sin(GRO)

GR1 将获得 GR0 计算出的正弦值。例如，如果 GR0 是 30°，那么 GR1 将变成 0.5。

4.7.11. Cos 计算余弦值

函数用法

Cos 用于计算一个角值的余弦值。 输入的数值单位是角度。

可变元素

Set [Variable1]=Cos [Variable2]

图 4.7.12

1. [Variable1]

变量类型：GR/LR

计算结果存放变量。

2. [Variable2]

变量类型：GR/LR/AIN/AOT

用于计算余弦值的目标变量

返回值

余弦值，范围＝【-1，1】。

详细举例

Set GR1=Cos(GRO)

GR1 将获得 GR0 计算出的余弦值。例如，如果 GR0 是 30°，那么 GR1 将变成 0.866。

4.7.12. Tan 计算正切值

函数用法

Tan 用于计算一个角值的正切值。输入的数值单位是角度。

可变元素

Set [Variable1]=Tan [Variable2]

图 4.7.13

1. [Variable1]

变量类型：GR/LR

计算结果存放变量。

2. [Variable2]

变量类型：GR/LR/AIN/AOT

用于计算正切值的目标变量

返回值

余弦值，范围＝【-1，1】。

详细举例

Set GR1=Tan(GRO)

GR1 将获得 GR0 计算出的正切值。例如，如果 GR0 是 45°，那么 GR1 将变成 1。

4.7.13. ASin 计算反正弦值

函数用法

ASin 函数接收一个介于 -1 到 1 之间的数值作为输入，返回该数值的反正弦角度值。反余弦函数的输出范围-90 度到 90 度。如果输入的数值超出了这个范围，执行该指令将会报错。

可变元素

Set [Variable1]=ASin [Variable2]

图 4.7.14

1. [Variable1]

变量类型：GR/LR

计算结果存放变量。

2. [Variable2]

变量类型：GR/LR/AIN/AOT

用于计算反正弦值的目标变量

详细举例

Set GR1=ASin(GRO)

GR1 将获得 GR0 计算出的反正玄值。例如，如果 GR0 是 0.5，那么 GR1 将变成 30°。

4.7.14. ACos 计算反余弦值

函数用法

ACos 函数接收一个介于-1 到 1 之间的数值作为输入，返回该数值的反余弦角度值。反余弦函数的输出范围 0 度到 180 度。如果输入的数值超出了这个范围[-1, 1]，执行该指令将会报错。

可变元素

Set [Variable1]=ACos [Variable2]

4.7.15

1. [Variable1]

变量类型：GR/LR

计算结果存放变量。

2. [Variable2]

变量类型：GR/LR/AIN/AOT

用于计算反余弦值的目标变量

详细举例

Set GR1=ACos(GRO)

GR1 将获得 GR0 计算出的反余弦值。例如，如果 GR0 是 0.5，那么 GR1 将变成 60°。

4.7.15. ATan 计算反正切角度值

用法

变量类型：GR/LR/AIN/AOT

用于计算反正切值的目标变量

详细举例

Set GR1=ATan(GRO)

GR1 将获得 GR0 计算出的反正切角度值。例如，如果 GR0 是 1，那么 GR1 将变成 45°。

4.7.16. CountNode 三条线计算交点计算函数

函数用法

CountNode 函数接受六个参数，每个参数代表一条线段的一个端点坐标。通过传递这六个点的坐标，函数可以计算出由这三条线段组成的平面图形的交点（如果存在）。

可变元素

Set [Variable1]=CountNode （[Point 1] [Point 2] [Point 3] [Point 4] [Point 5] [Point 5]）

图 4.7.17

1. [Variable1]

变量类型：GR/LR

计算结果存放变量。

2. [Point 1]

变量类型：GP/LP

线 1 的第一个点位置。

3. [Point 2]

变量类型：GP/LP

线 1 的第二个点位置。

4. [Point 3]

变量类型：GP/LP

线 2 的第一个点位置。

5. [Point 4]

变量类型：GP/LP

线 2 的第二个点位置。

6. [Point 5]

变量类型：GP/LP

线 3 的第一个点位置。

7. [Point 6]

变量类型：GP/LP

线 3 的第二个点位置。

详细举例

Set GP3=CountNode(GP1 GP2 GP3 GP4 GP5 GP6)

在这个例子中，GP3 将会接收到如果三条线段有交点的话，该交点的坐标值。

4.7.17. Offset 偏移函数

函数用法

将 GP 点位按指定的坐标系进行偏移坐标值

可变元素

Set [Variable1]=Offest （[Point 1] [offsetValue] [Offset Coord.]）

图 4.7.18

1. [Variable1]

变量类型：GR/LR

计算结果存放变量。

2. [Point 1]

变量类型：GP/LP

计算点位数据

3. [offsetValue]

类型：GI/LI/GR/LR/Num

设置偏移值

4. [Offset Coord.]

类型：用户/工具/世界坐标

偏移坐标系选择

详细举例

例 1:

Set GP1=Offset(GP0.X+200 User)

GR1 将获得 GR0 的 X 方向基于用户坐标系的 X 坐标方向加 200。

例 2:

Set GP1=Offset(GP0.X+200 Too1)

GR1 将获得 GR0 的 X 方向基于工具坐标系的 X 坐标方向加 200。

例 3:

Set GP1=Offset(GP0.X+200 Word User=0)

GR1 将获得 GR0 的 X 方向基于世界坐标系的 X 坐标方向加 200 并换算到用户 0 下面。

4.7.18. 字符串组合

函数用法

字符串组合通常涉及到将两个或多个字符串连接成一个新的字符串。通过使用加号 (+) 运算符进行拼接。【编辑方式-详见：4.7.19 附加功能-自定义文本】

可变元素

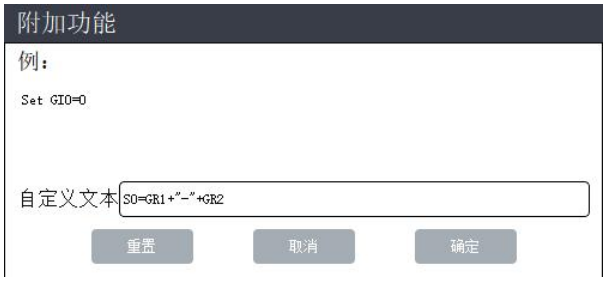


图 4.7.19

详细举例

例 1:

Set S0=S1+S2

在这一行中，S0 将被设定为 S1 和 S2 字符串的组合。例如，如果 S1 是"Hello "，而 S2 是"World!"，那么 S0 将变成"Hello World!"。

例 2:

Set S0=GR0+","+GR1+"#"

在这个例子中，SO 将被设定为一个由 GRO、逗号(,)、GR1 以及另一个井号(#)组成的字符串。例如：如果 GR0=3.14，而 GR1 是 5.56，那么 S0 将变成“3.14,5.56#”

4.7.19. 附加功能-自定义文本

图 4.7.20

点击 Set 指令详细编辑界面下方的自定义文本，可打开编辑内容。

图 4.7.21

自定义文本内容默认为上一次插入的 Set 指令，可直接在自定义文本中编辑或修改指令。

点击“确定”后，Set 指令根据自定义文本内容进行变动，再点击“确认插入”即可插入指令。

若点击“重置”，自定义文本内容恢复到初始默认内容。

4.7.20. 工具坐标系修改

描述

动态设置工具坐标系值，适用于参数化编程，离线仿真等应用场景

可变元素

Set TOOL [Num] [ToolParam]=[Variable]

输出信息 1

[Num]

常数：1-49

变量：UI/GIN/GOT

解释：输出到指定的工具号

输出信息 2

[ToolParam]

变量要素：ALL/X/Y/Z/Rx/Ry/Rz/W/Xg/Yg/Zg/Lx/Ly/Lz

输入参数 1

[Variable]

变量：TOOL/GR/LR/AIN/AOT/常数

解释：输入需要赋值的参数值

详细举例

例 1:

Set TOOL1.ALL=TOOL0.ALL

将 Tool0 坐标系变量的所有要素值复制并赋值给 Tool1 的相应变量中。

例 2:

Set TOOL1.X=450

将 Tool1 坐标系 X 坐标赋值修改为 450。

例 3:

Set TOOL1.W=20

将 Tool1 坐标系 W 工具质量赋值修改为 20。

注意

1. 在使用该功能之前，请确保您已经充分理解了机器人点位记录过程中，位置和姿态是基于用户坐标系对工具坐标系的参考描述。

2. 请勿将工具坐标系值相较之前修改差异过大，以防止在执行先前记录的轨迹点位时，由于点位坐标反解至机器人各关节过程中出现 CFG 轴配置不一致的问题。

4.7.21. 用户坐标系修改

描述

动态设置用户坐标系值，适用于参数化编程，离线仿真等应用场景

可变元素

Set USER [Num] [UserParam]=[Variable]

输出信息 1

[Num]

常数：1-49
变量：UI/GIN/GOT
解释：输出到指定的用户号

输出信息 2

[ToolParam]

变量要素：ALL/X/Y/Z/Rx/Ry/Rz

输入参数 1

[Variable]

变量：USER/GR/LR/AIN/AOT/常数

解释：输入需要赋值的参数值

详细举例

例 1:

Set USER1.ALL=USER0.ALL

将 USER0 坐标系变量的所有要素值复制并赋值给 USER1 的相应变量中。

例 2:

Set USER1.X=450

将 USER1 坐标系 X 坐标赋值修改为 450。

注意

1. 在使用该功能之前，请确保您已经充分理解了机器人点位记录过程中，位置和姿态是基于用户坐标系对工具坐标系的参考描述。

2. 请勿将用户坐标系值相较之前修改差异过大，以防止在执行先前记录的轨迹点位时，由于点位坐标反解至机器人各关节过程中出现 CFG 轴配置不一致的问题。

4.8. CreateUser---创建用户坐标系指令

4.8.1. 指令用法

建立用户坐标系。

4.8.2. 指令逻辑

用户需给出三个空间点位，第一个点位用于确定坐标系原点，第二个点位用于确定 X 轴方向，第三个点位用于确定 Y 轴方向，Z 轴方向通过右手定则确定，通过三个空间点建立一个用户坐标系。

4.8.3. 可变元素

CreateUser [A] GP=[B] GP=[C] GP=[D]

1. [A]

变量类型：整型变量。

解释：[A]为用户坐标系号。

2. [B]

变量类型：整型变量。

解释：[B]为 GP 变量号，该 GP 变量用于确定用户坐标系原点。

3. [C]

变量类型：整型变量。

解释：[C]为 GP 变量号，该 GP 变量用于确定用户坐标系 X 轴方向。

4. [D]

变量类型：整型变量。

解释：[D]为 GP 变量号，该 GP 变量用于确定用户坐标系 Y 轴方向。

4.8.4. 应用举例

CreateUser 1 GP=1 GP=2 GP=3； //以 GP1、GP2、GP3 中的点位信息建立用户坐标系 1

MoveL VL=500mm/s PL=0.0 User=1；

MoveL VL=50mm/s PL=0.0 User=1；

5. 焊接指令

5.1. ArcOn---起弧

5.1.1. 指令用法

用于弧焊起弧。需要与 ArcOff 指令配合使用。

5.1.2. 应用举例

例 1:

ArcOn （0）

调用弧焊工艺 0 号参数文件。

例 2:

ArcOn （0）VL=10

调用弧焊工艺 0 号参数文件，弧焊速度为 10mm/s。

5.1.3. 程序运行

执行此程序行，机器人系统将会调用既定的焊接工艺号，并使用指令中设置的相关参数。

5.1.4. 可变元素

ArcOn [Process] [\Speed] [\Current 、Voltage] [\Pattern] [\Cataed] [\La][\MArc] [\FileN]

1. [process]

数据类型：num

解释：数值变量或者永久数据。

2. [\Speed]

数据类型：num 解释: 焊接速度。

3. [\Current、Voltage]

数据类型：num（I：0-160 V：0-20）解释: 焊接电流电压。

4. [\Pattern]

变量类型：Int（0-99）

解释: 焊机模式包含麦格米特等焊机模式 Mode 和调用的焊机工作文件 JOB。

5. [\Cataed]

数据类型：Int（0-9999）

解释: 鱼鳞焊，包含时间 T，间距 L1 和空行 L2 数据。

6. [\La]

数据类型：Int（-1，0，1）

解释: 搭接，-1 为关闭、0 为保持、1 为开启。

7. [\MArc]

数据类型：Int（-1，0，1）

解释: 多次起弧，-1 为关闭、0 为保持、1 为开启。

8. [\FileN]

数据类型：Int（0，15）

解释: 焊机文件号，可选择使用变量 UI，卡诺普焊机专用。

5.1.5. 详细举例

例 1:

```
.....   ;焊接安全点
.....   ; 机器人移动到焊接起始点
ArcOn(1)    VL=10I=160 V=17  ; 起弧
.....           ; 焊接到结束点
ArcOff      ; 灭弧
.....   ; 机器人移动到安全点
```

机器人从安全点移动到焊接起始点后，机器人弧焊起弧并调用焊接工艺号 1，机器人按设定的焊接速度为 10mm/s、电流为 160A、电压为 17V 开始焊接，一直焊接到结束点后灭弧。

例 2:

机器人运行到指定点后，机器人弧焊起弧并调用 UI1，变量 UI1 的值为焊接工艺号，机器人按设定的焊接速度为 10mm/s、电流为 160A、电压为 17V、焊机控制模式为 2、调用焊机工作号为 3、搭接模式为开启，开始焊接，一直焊接到结束点后灭弧。

例 3:

```
.....   ;焊接安全点
.....   ; 机器人移动到焊接起始点
ArcOn(UI1) VL=10 I=160 V=17 T=600 L2=8 MArc=1 FileN=1 ;起弧
.....           ; 焊接到结束点
ArcOff      ; 灭弧
.....   ; 机器人移动到安全点
```

机器人运行到指定点后，机器人弧焊起弧并调用 UI1，变量 UI1 的值为焊接工 艺号，机器人按设定的焊接速度为 10mm/s、电流为 160A、电压为 17V、鱼鳞焊开启、焊接时间为 600ms、空行距离为 8mm、多次起弧开启、搭接焊接开启，使用焊机工作文件 1，然后开始焊接，一直焊接到结束点后灭弧。

5.2. ArcOff---灭弧

5.2.1. 指令用法

用于弧焊灭弧。需要与 ArcOn 指令配合使用。

5.2.2. 应用举例

例 1:

ArcOff

机器人灭弧。

例 2:

ArcOff I=160 V=17

机器人执行灭弧，灭弧电流电压渐变为 160A， 17V。

5.2.3. 程序运行

执行此程序行，机器人系统将会灭弧，焊接参数将会从起弧工艺号中的参数渐变为灭弧指令中的设定参数。

5.2.4. 可变元素

ArcOff [\Current 、Voltage] [\ArcSA] [\ArcST] [\ArcSD] [\WSD]

1. [\Current 、Voltage]

数据类型：num（I：0-160 V：0-20）
解释:灭弧时，从起弧焊接工艺中的电流电压渐变至灭弧时的电流电压。

2. [\ArcSA]

数据类型：float（0-10）

解释:收弧衰减，ArcSA1 为正常衰减、ArcSA2 为提前衰减。

3. [\ArcST]

数据类型：Int（1-5）解释:收弧次数。

4. [\ArcSD]

数据类型：Int（-1，0，1）

解释:灭弧检测，-1 为关闭、0 为保持、1 为开启。

5. [\WSD]

数据类型：Int（-1，0，1）

解释:防粘丝，-1 为关闭、0 为保持、1 为开启。

5.2.5. 详细举例

例 1:

```
.....   ;焊接安全点
.....          ；机器人移动到焊接起始点
ArcOn(1) VL=10 I=160 V=20；起弧
.....   ；移动到焊接结束点
ArcOff I=160 V=20；灭弧
MoveL VL=100.0 PL=9.0 T=1；机器人移动到安全点
```

机器人运行到指定点后，机器人执行灭弧，电压电流从起弧焊接工艺号中的电流电压渐变为指令中设置的 160A 、20V。

例 2:

```
.....   ;焊接安全点
.....           ；机器人移动到焊接起始点
ArcOn(1)    VL=10I=160 V=20  ；起弧
.....   ；移动到焊接结束点
ArcOff I=160  V=20  ArcSA1=0.5  ArcST=2
.....   .....   ；机器人移动到安全点
```

机器人运行到指定点后，机器人执行灭弧，灭弧时，电压电流从起弧焊接工艺号中的电流电压渐变为灭弧指令中设置的 160A、20V，正常收弧衰减参数为 0.5、灭弧次数为 2 次。

例 3:

```
.....   ;焊接安全点
.....           ；机器人移动到焊接起始点
ArcOn(1)    VL=10I=160 V=20  ；起弧
.....   ；移动到焊接结束点
ArcOff(UI0)  I=160 V=20 ArcST=2 ArcSD=1 WSD=1
.....   ；机器人移动到安全点
```

机器人运行到指定点后，机器人灭弧并关闭调用 UI0，变量 UI0 的值为焊接工艺号，灭弧时，电压电流从起弧焊接工艺号中的电流电压渐变为灭弧指令中设置的 160A、20V，灭弧检测开启，防粘丝功能开启。

5.3. ArcOn 起弧与 ArcOff 应用示例

```
MoveJ      VJ=10.0%   PL=9.0      T=1   ;焊接安全点
MoveL      VL=100.0    PL=9.0     T=1 ; 机器人移动到焊接起始点
ArcOn(1)    VL=10      I=160 V=17 ; 起弧
MoveL      VL=100     PL=9.0      T=1   ; 焊接到结束点
ArcOff I=120 V=13
MoveL      VL=100.0    PL=9.0     T=1   ; 机器人移动到安全点
```

机器人从安全点移动到焊接起始点后，机器人弧焊起弧并调用焊接工艺号 1，机器人按设定的焊接速度为 10mm/s、电流为 160A、电压为 17V 开始焊接。机器人一直移动到焊接结束点后执行灭弧，灭弧时，电压电流从起弧焊接工艺号中的电流电压渐变为灭弧指令中设置的 120A 、13V。

5.4. LaserWeldOn---激光焊接开始

5.4.1. 指令用法

用于激光焊接，需要与 LaserWeldOff 指令配合使用。

5.4.2. 程序运行

使用激光焊接工艺时，开启激光，进行工件焊接。

5.4.3. 可变元素

图 5.4.1

LaserWeldOn [Process] [Speed] [Power 、Wire Feed Rate][Pattern1] [Pattern2]

1. [Process]

数据类型：Int（0-99）

解释：工艺序号，为数值变量或者常数。

2. [Speed]

数据类型：float（0.001-999.999）

单位：mm/s

解释: 焊接速度。设置此参数时将不调用工艺中的速度，而生效当前参数。

3. [Current、Voltage]

数据类型：Int（W/%：0-999999、m/min：0-999）

单位：W、m/min

解释: 焊接电流电压。设置此参数时将不调用工艺中的功率和送丝速度，而生效当前参数。

4. [Pattern1]

变量类型：Int（0-63）

解释: 设置需要调用的焊机工作文件。设置的 0-63 的数字将转换为 6 位二进制信号，通过功能配置中焊机 JOB0-5 配置端口号输出信号(JOB0-5 代表从信号的低位到高位)，焊机通过接收 6 路信号后调用对应焊机工作文件号。若不启用焊接 JOB，则不会调用焊机中任何工作文件。

图 5.4.2

5. [Pattern2]

变量类型：Int（0-63）

解释: 设置需要调用的焊枪工作文件。设置的 0-63 的数字将转换为 6 位二进制信号，通过功能配置中焊枪 JOB0-5 配置端口号输出信号(JOB0-5 代表从信号的低位到高位)，焊枪工件通过接收 6 路信号后调用对应焊枪工作文件号。若不启用焊枪 JOB，则不会调用焊枪中任何工作文件。

5.4.4. 详细举例

例 1:

```
MoveL VL=100 PL=0.0 T=1
LaserWeldOn（0） WelderJob=0 WeldGunJob=0
MoveL VL=100 PL=0.0 T=1
LaserWeldOff
```

#调用激光 0 号工艺参数文件。并且调用焊机及焊枪 0 号工艺(在启用焊机及焊枪 JOB 信号及信号线连接的情况下生效)。

例 2:

```
MoveL VL=100 PL=0.0 T=1
LaserWeldOn（0）VL=5 I=2000 V=6 WelderJob=0 WeldGunJob=0
MoveL VL=100 PL=0.0 T=1
LaserWeldOff
```

#调用激光工艺 0 号参数文件，且修改焊接速度为 5mm/s，焊接功率为 2000W，送丝速度为 6m/min。

5.5. LaserWeldOff---激光焊接结束

5.5.1. 指令用法

用于激光焊接，需要与 LaserWeldOff 指令配合使用。

5.5.2. 程序运行

停止激光焊接。

5.5.3. 可变元素

图 5.5.1

LaserWeldOff [change] [ArcST] [ArcSD] [WSD] [Advance] [Attenuate] [BackWire]

1. [change]

数据类型：状态开关--使用/不使用

解释：选择焊接过程中，是否开启功率和送丝速度的渐变。若选择使用，则需要设置渐变功率和参数，且焊接过程中，功率和送丝速度从焊接开始时的正常焊接参数值逐渐变化到焊接结束指令中设置的渐变参数值。

2. [ArcST]

数据类型：Int-(1-5)

解释：焊接结束后收弧的次数。

3. [ArcSD]

数据类型：状态开关--保持/打开/关闭

解释：

1、保持/打开/关闭状态数值 0/1/-1，且选择保持状态时，指令中不显示相关参数。

2、选择灭弧检测功能的保持/打开/关闭(灭弧检测功能在机器人灭弧动作完成后检测是否成功灭弧。

4. [WSD]

数据类型：状态开关-保持/打开/关闭

解释：

1、保持/打开/关闭状态数值 0/1/-1，且选择保持状态时，指令中不显示相关参数。

2、选择防粘丝功能的模式。防粘丝功能是在焊接结束时自动进行焊丝粘合工件检测，且在短时间内施加大电压来熔断粘合部分的焊丝，解除焊枪与工件粘合的功能。

5. [Advance]

数据类型：状态切换--(距离/时间)

解释：

1、距离/时间状态数值分别为 0/1。

2、选择提前指定距离或者时间进行焊接状态参数的衰减。

6. [Attenuate]

数据类型：状态切换-(不使用/给定)

单位：mm/ms(设置为给定)

解释：

1、选择不使用时，将不开启提前模式。选择给定时，需设置提前模式的提前衰减参数。

2、设置的提前衰减参数值范围为 0~10000。

3、提前衰减参数效果为激光焊接的功率和送丝速度由正常值渐变到渐变参数中设置的功率和送丝速度。变化的距离/时间由提前模式及提前衰减参数决定。

7. [BackWire]

数据类型：状态切换-(不使用/给定)

单位：mm/ms(设置为给定)

解释：

1、选择不使用时，将不开启提前模式。选择给定时，需设置提前模式的提前回抽丝参数。

2、设置的提前衰减参数值范围为 0~10000。

5.5.4. 详细举例

例 1：

```
MoveL VL=100 PL=0.0 T=1
LaserWeldOn（0） WelderJob=0 WeldGunJob=0
MoveL VL=100 PL=0.0 T=1
LaserWeldOff
```

#调用激光 0 号工艺参数文件。并且调用焊机及焊枪 0 号工艺(在启用焊机及焊枪 JOB 信号及信号线连接的情况下生效)。

例 2:

```
MoveL VL=100 PL=0.0 T=1
LaserWeldOn（0）VL=5 I=2000 V=6 WelderJob=0 WeldGunJob=0
MoveL VL=100 PL=0.0 T=1
LaserWeldOff
```

#调用激光工艺 0 号参数文件，且修改焊接速度为 5mm/s，焊接功率为 2000W，送丝速度为 6m/min。

5.6. WeaveOn--摆弧开启

5.6.1. 指令用法

用于弧焊时开启并调用摆弧工艺。需要与 WeaveOff 指令配合使用。

5.6.2. 应用举例

例 1:

WeaveOn（0）

开启并调用弧焊摆弧工艺 0。

例 2:

WeaveOn（0）F=2 R=3.5

开启并调用摆弧工艺 0，摆动频率为 2Hz，摆动幅度为 3.5mm。

5.6.3. 程序运行

执行此程序行，机器人系统将会开启并调用指定的摆弧工艺，并按照指令中设置的参数进行摆弧。

5.6.4. 可变元素

图 5.6.1

WeaveOn [Process] [\F] [\R] [Mode] [refer]

1. [process]

数据类型：num

解释：数值变量或者永久数据。

2. [\F]

数据类型：float（1-20）

单位：hz

解释:摆动频率。

3. [\R]

数据类型：float（0-100）

单位：mm

解释:摆动幅度。

4. [Mode]

数据类型：Int（0，1）

解释:变化模式（0：渐变，1：突变）。

5. [refer]

变量类型：Tra、Tool、Use

解释:参考坐标的坐标方向。

5.6.5. 详细举例

例 1:

```
.....   ;焊接安全点
.....          ; 机器人移动到焊接起始点
ArcOn （0）        ; 起弧
WeaveOn（0）F=2 R=3.5       ; 摆弧开始
.....                ; 移动到焊接结束点
WeaveOff          ; 摆弧结束
ArcOff       ; 灭弧
.....   ; 焊接结束点
```

机器开启弧焊后，运行到指令行系统开启并调用摆弧工艺 0，摆动频率为 2，摆动幅度为 3.5。

例 2:

```
.....      ; 机器人移动到安全点
.....      ; 机器人移动到焊接起始点
ArcOn （0）    ; 起弧
WeaveOn（UI0）F=LR0  R=3.5  Tool=X  ; 摆弧开始
.....              ; 移动到焊接结束点
WeaveOff     ; 摆弧结束
ArcOff       ; 灭弧
.....         ; 机器人移动到安全点
```

机器开启弧焊后，运行到指令行系统开启并调 UI0，变量 UI0 的值为焊接摆弧工艺号，摆动频率为储存在 LR0 中的数据，摆动幅度为 3.5，参考坐标为工具坐标的 X 方向。

5.7. WeaveOff--摆弧关闭

5.7.1. 指令用法

用于弧焊时关闭摆弧工艺。需要与 WeaveOn 指令配合使用。

5.7.2. 应用举例

例 1:

WeaveOff

关闭弧焊摆弧工艺。

例 2:

WeaveOff F=2 R=3.5

关闭弧焊摆弧工艺，摆动频率从相应工艺的摆动频率变为 2，摆动幅度从相应工艺中的摆动幅度渐变为 3.5。

5.7.3. 程序运行

执行此程序行，机器人系统将会关闭摆弧工艺，并将工艺号中的参数渐变成指令中设置的参数。

5.7.4. 可变元素

图 5.7.1

WeaveOff [Mode] [\F] [\R]

1. [Mode]

数据选项：默认/渐变

解释：选择默认时，程序运行到摆弧关闭指令时，摆弧动作停止。选择渐变时，摆弧运行由工作参数渐变到当前设置参数，并停止摆弧运行。

2. [\F]

数据选项：GR/LR/常数

变量类型：float（0.00-10.00）

解释: 摆动频率，单位 Hz。

3. [\R]

数据选项：GR/LR/常数

变量类型：float（0.000-100.000），单位 mm。

解释: 设置摆动幅度。

5.7.5. 详细举例

例 1:

```
.....  ;焊接安全点
.....          ；机器人移动到焊接起始点
ArcOn （0）        ；起弧
WeaveOn（0）F=2 R=3.5      ；摆弧开始
.....          ；焊接到结束点
WeaveOff F=LR2 R=3.5；摆弧结束
ArcOff       ；灭弧
.....  ；焊接结束点
```

关闭弧焊摆弧工艺，摆动频率从相应工艺中的摆动参数渐变为变量 LR2 的值，摆动幅度从相应工艺中的参数渐变为 3.5。

例 2:

关闭弧焊摆弧工艺，摆动频率从工艺相应中的参数渐变为储存在变量 LR2 中的值，摆动幅度从相应工艺中的参数渐变为储存在变量 GR1 中的值。

5.8. WeaveOn 与 WeaveOff 的应用示例

```
MoveJ      VJ=10.0%   PL=9.0      T=1   ;焊接安全点
MoveL      VL=100.0    PL=9.0     T=1 ; 机器人移动到焊接起始点
ArcOn(1)   VL=10I=160 V=17 ; 起弧
WeaveOn（0）F=2 R=3.5            ; 摆弧开始
MoveL      VL=100      PL=9.0     T=1   ; 焊接到结束点
WeaveOff F=LR2         R=3.5      ; 摆弧结束
ArcOff  I=120  V=13     ; 灭弧
MoveL      VL=100.0    PL=9.0     T=1   ; 机器人移动到安全点
```

机器人从安全点移动到焊接起始点后，机器人弧焊起弧并调用焊接工艺号 1，机器人按设定的焊接速度为 10mm/s、电流为 160A、电压为 17V 开始焊接。然后机器开启弧焊后，运行到指令行系统开启并调用摆弧工艺 0，摆动频率为 2，摆动幅度为 3.5。

移动到焊接结束点后，关闭弧焊摆弧工艺，摆动频率从工艺中的相应的摆动参数，渐变为变量 LR2 的值，摆动幅度从相应工艺中的参数渐变为 3.5。然后执行灭 弧，灭弧电压电流从工艺号中的电流电压渐变为指令中设置的 120A 、13V。最后机器人移动到安全点。

5.9. MpStart---开始多层多道

5.9.1. 指令用法

开始运行多层多道工艺，需要与 MpEnd 指令配合使用。

5.9.2. 程序运行

焊接过程中，执行多层多道工艺。

5.9.3. 可变元素

图 5.9.1

MpStart [Process] [\Start] [\End]

1. [Process]

数据选项：常数/UI

数据类型：int(0~999)

解释：生效的多层多道工艺号。

2. [\Start]

数据选项：常数/UI

数据类型：int(1~999)

解释：多层多道工艺的起始生效层数。

3. [\End]

数据选项：常数/UI

数据类型：int(1~999)

解释：多层多道工艺的结束生效层数。

注意：起始层层数需要小于结束层层数

5.9.4. 应用举例

```
MoveJ VJ=30.0 PL=9.0 T=1 //安全待机点

MpStart 0 1-3 //调用 0 号多层多道工艺，开始层 1，结束层：3.

MoveL VL=100 PL=9.0 T=1 //焊接准备点

OffsetOn GP0 User //偏移开启

MoveL VL=100 PL=9.0 T=1 //焊接起始点

ArcOn (0) //起弧

WeaveOn (0) Tra=Y Mode=0 //开始摆弧

MoveL VL=10 PL=9.0 T=1 //焊接终点

WeaveOff Mode=0 //摆弧结束

ArcOff //灭弧

OffsetEnd //偏移结束

MoveL VL=100 PL=9.0 T=1 //焊接完成过渡点

MpEnd //结束多层多道工艺

MoveL VL=200 PL=0.0 T=1 //安全待机点
```

5.10. MpEnd---结束多层多道

5.10.1. 指令用法

结束多层多道，需要与 MpStart 指令配合使用。

5.10.2. 程序运行

停止执行多层多道工艺。

5.10.3. 可变元素

图 5.10.1

暂无。

5.11. SPOTJ---点焊关节

5.11.1. 指令用法

以关节运动形式，进行点焊操作。

5.11.2. 程序运行

以关节运动形式到达规划点位后，使用点焊工艺进行焊接作业。当机器人各轴已经到达目标位置（目标位置与指令中 PL 参数有关系），才会执行下一条指令。

5.11.3. 可变元素

SPOTJ [\Topoint] VJ=[Speed] PL=[zone] [\ACC] [\Dec] [\S] [Tool] [User] [\Mode] [\GUN]

1. **[\Topoint]**

变量类型：GP（全局变量）、LP（局部变量）。

解释：机器人和外部轴的目标点位置。可以选择使用 GP、LP 变量。当选择变量时，变量中的值为机器人和外部轴的目标点位置。不选择变量时机器人目标位置储存在指令中。

2. VJ=[Speed]

数据类型： float（范围：0.0-100.0）

变量类型：GR（全局变量）、LR（局部变量）

解释：关节运动以最大速度为基准的百分比。可以选择使用 GR、LR 变量。当选择变量时，使用变量中的值作为机器人运行速度的百分比。

3. PL=[zone]

数据类型：float（范围：0.0-9.0）

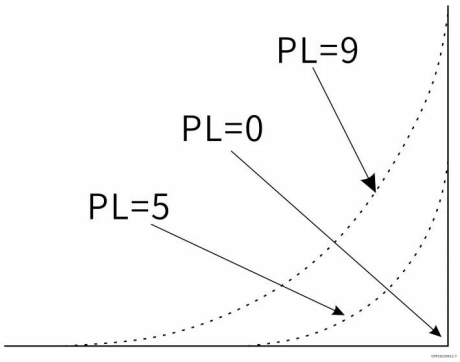


图 5.11.1

解释：机器人在运动轨迹中到达目标点的逼近等级。描述了所生成拐角的路径大小。逼近等级设置越小，机器人运行轨迹时生成拐角路径越小。

4. [\ACC]

数据类型：Int（范围：1-20）

解释：加速度，该加速度与机器人运动方式有关。加速等级设置越小机器人加速越快，加速等级设置越大加速越慢。默认不使用。

5. [\Dec]

数据类型：Int（范围：1-20）

解释：减速度，该减速度与机器人运动方式有关。减速等级设置越小机器人减速越快，减速等级设置越大减速越慢。默认不使用。

6. [\S]

数据类型：Int（范围：0-9）

解释：滤波等级分为 0-9，默认为 0。滤波等级主要用于小距离、速度慢，可将等级改大，提高加减速能力与速度。滤波等级会影响节拍，滤波等级越大，时间越短。参数不合适可能造成机器人末端抖动，所以请根据实际情况调节。图 1.1.2 为滤波等级的一个速度示意图。

注意：指令里面滤波等级只针对步长滤波有效。

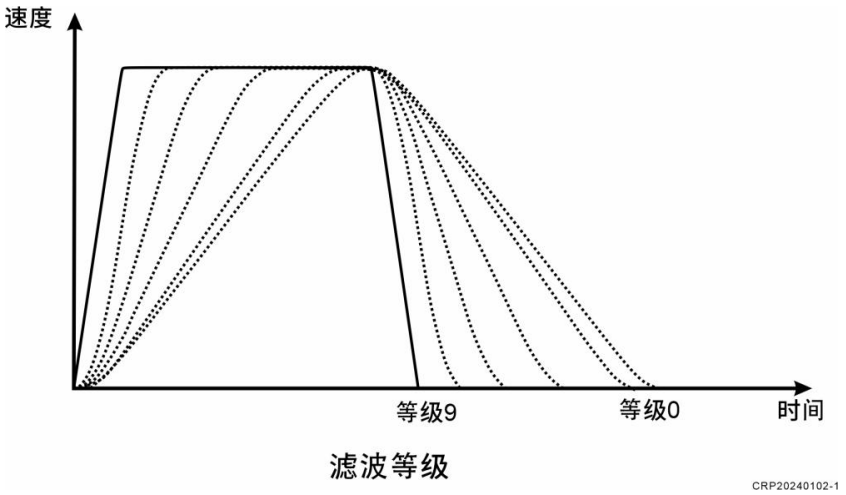


图 5.11.2

7. [Tool]

数据类型:Int (范围：0-49)

解释：机器人移动时使用的工具坐标系。0 号工具默认工具坐标点在机器人法兰末端中心处。

8. [User]

数据类型：Int (范围：0-49)

解释：机器人移动时使用的用户坐标系。0 号工件坐标为默认用户坐标系。可选择使用该元素。如果使用协同的外部轴时必须使用该参数。

9. [\Mode]

变量类型：SPOT/TD/EC/DC

解释：切换模式，焊接加压/磨损加压/空打补偿/基准板补偿工艺。

10. [\GUN]

变量类型:常数(范围：1-4)

解释：设定当前的焊枪编号，根据焊枪调用对应的焊枪参数，默认为 1 号焊枪。

5.11.4. 详细举例

示例 1:

SPotJ VJ=100 PL=0.0 T=1 U=1 SPOT=1 GUN=1

解释：以关节方式运动到目标点，且使用焊枪 1，开始焊接加压工艺。

5.12. SPOTL---点焊直线

5.12.1. 指令用法

以直线运动形式，进行点焊操作。

5.12.2. 程序运行

以直线运动形式到达规划点位后，使用点焊工艺进行焊接作业。当机器人各轴已经到达目标位置（目标位置与指令中 PL 参数有关系），才会执行下一条指令。

5.12.3. 可变元素

SPOTL [\Topoint] VJ=[Speed] PL=[zone] [\ACC] [\Dec] [\S] [Tool] [User] [\Mode] GUN=[\numberl]

1. **[\Topoint]**

变量类型：GP（全局变量）、LP（局部变量）。

解释：机器人和外部轴的目标点位置。可以选择使用 GP、LP 变量。当选择变量时，变量中的值为机器人和外部轴的目标点位置。不选择变量时机器人目标位置储存在指令中。

2. VJ=[Speed]

数据类型： float（范围：0.0-100.0）

变量类型：GR（全局变量）、LR（局部变量）

解释：关节运动以最大速度为基准的百分比。可以选择使用 GR、LR 变量。当选择变量时，使用变量中的值作为机器人运行速度的百分比。

3. PL=[zone]

数据类型：float（范围：0.0-9.0）

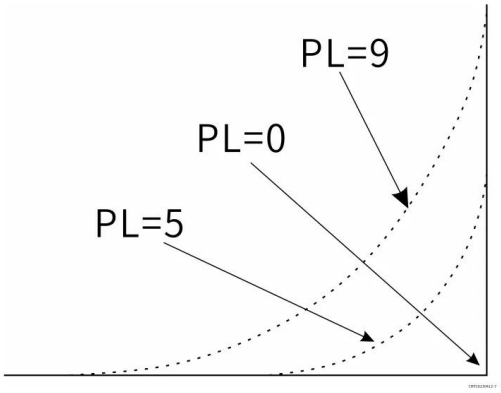


图 5.12.1

解释：机器人在运动轨迹中到达目标点的逼近等级。描述了所生成拐角的路径大小。逼近等级设置越小，机器人运行轨迹时生成拐角路径越小。

4. [\ACC]

数据类型：Int（范围：1-20）

解释：加速度，该加速度与机器人运动方式有关。加速等级设置越小机器人加速越快，加速等级设置越大加速越慢。默认不使用。

5. [\Dec]

数据类型：Int（范围：1-20）

解释：减速度，该减速度与机器人运动方式有关。减速等级设置越小机器人减速越快，减速等级设置越大减速越慢。默认不使用。

6. [\S]

数据类型：Int（范围：0-9）

解释：滤波等级分为 0-9 级，默认为 0。滤波等级主要用于小距离、速度慢，可将等级改大，提高加减速能力与速度。滤波等级会影响节拍，滤波等级越大，时间越短。参数不合适可能造成机器人末端抖动，所以请根据实际情况调节。图 1.1.2 为滤波等级的一个速度示意图。

注意：指令里面滤波等级只针对步长滤波有效。

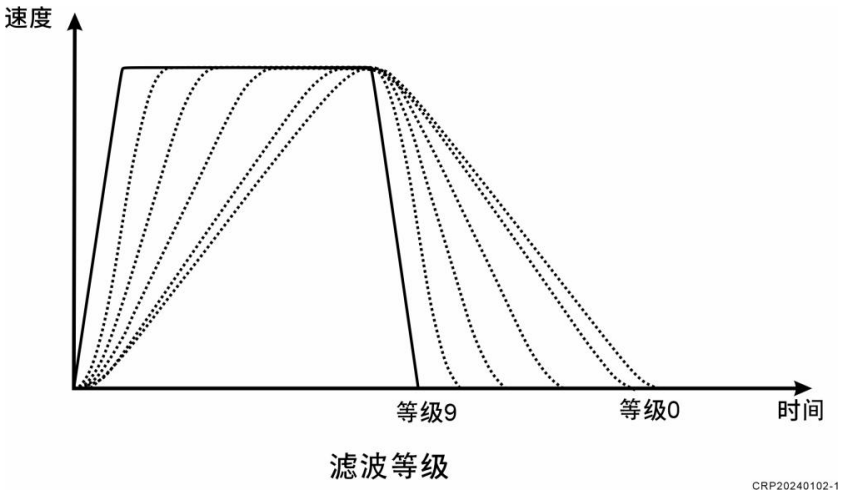


图 5.12.2

7. [Tool]

数据类型:Int (范围：0-49)

解释：机器人移动时使用的工具坐标系。0 号工具默认工具坐标点在机器人法兰末端中心处。

8. [User]

数据类型：Int (范围：0-49)

解释：机器人移动时使用的用户坐标系。0 号工件坐标为默认用户坐标系。可选择使用该元素。如果使用协同的外部轴时必须使用该参数。

9. [\Mode]

变量类型：SPOT/TD/EC/DC

解释：切换焊接加压/磨损加压/空打补偿/基准板补偿工艺。

10. [\GUN]

变量类型:常数(范围：1-4)

解释：设定当前的焊枪编号，根据焊枪调用对应的焊枪参数，默认为 1 号焊枪。

5.12.4. 详细举例

示例 1:

SPotL VL=100 PL=0.0 T=1 U=1 SPOT=1 GUN=1

解释：以直线方式运动到目标点，且使用焊枪 1，开始焊接加压工艺。

5.13. ARCTRACKSTART---电弧跟踪开始

5.13.1. 指令用法

开启电弧跟踪功能，需要与摆弧指令(需实现左右跟踪时)配合使用。

5.13.2. 程序运行

执行此指令后，机器人将通过焊丝触碰工件，获取电信号进行计算，开启焊缝上下跟踪及左右跟踪(需加上摆弧动作)实现焊接轨迹的一致性。

5.13.3. 可变元素

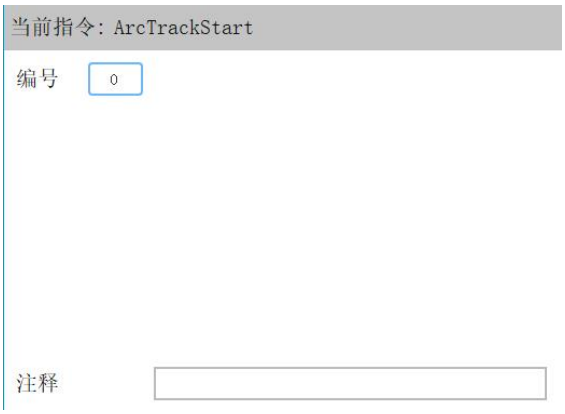


图 5.13.1

ArcTrackStart [Process]

1. [process]

数据选项：常数

变量类型：int（0-999）

解释: 开启电弧跟踪功能调用的电弧跟踪工艺号。

5.13.4. 详细举例

```
MoveL VL=200 PL=0.0 T=1 U=1  //焊接准备点
ArcOn (0) //起弧
WeaveOn (0) Tra=Y Mode=0 //摆弧开始
ArcTrackStart Index=0 //跟踪开始
MoveL VL=100 PL=0.0 T=1 U=1  //焊接结束点
ArcTrackEnd  //停止电弧跟踪
WeaveOff (0)  //停止摆弧
ArcOff   //灭弧
MoveL VL=200 PL=0.0 T=1 U=1  //过渡点
```

5.14. ARCTRACKEND---电弧跟踪结束

5.14.1. 指令用法

关闭电弧跟踪功能，需要与 ArcTrackStart 指令配合使用。

5.14.2. 程序运行

执行此指令后，机器人将不再获取电信号进行计算，关闭电弧跟踪功能。

5.14.3. 可变元素

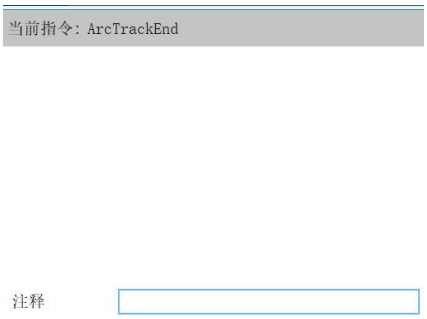


图 5.14.1

暂无可变元素。

5.14.4. 详细举例

```
MoveL VL=200 PL=0.0 T=1 U=1  //焊接准备点
ArcOn (0) //起弧
WeaveOn (0) Tra=Y Mode=0 //摆弧开始
ArcTrackStart Index=0 //跟踪开始
MoveL VL=100 PL=0.0 T=1 U=1  //焊接结束点
ArcTrackEnd  //停止电弧跟踪
WeaveOff (0)  //停止摆弧
ArcOff   //灭弧
MoveL VL=200 PL=0.0 T=1 U=1  //过渡点
```

5.15. SearchStart---寻位开始

5.15.1. 指令用法

用于寻找工件特征点，需要与 SearchEnd 及 MoveL 指令配合使用。

5.15.2. 程序运行

执行此指令后，配合设置了附加条件的运动指令沿指定方向运行，在触碰/扫描到工件后停止并记录信息。

5.15.3. 可变元素

图 5.15.1

SearchStart TN=[Process] [\SS] [\WS] [\Para] [\SPoint] [\SNum][\STime]

1. [process]

数据类型：int (0-999)

解释：寻位功能调用的寻位工艺号。

2. [\SS]

数据类型：int (0-1)

解释：寻位传感器类型，可选择“接触”/“线激光”。其中“接触”表示使用焊丝触碰工件寻位，“线激光”表示使用激光传感器。（0：“触碰”，选择该类型时隐藏该参数及后续参数 3~7；1：“线激光”，选择该类型时显示该参数及后续参数 3~7。）

3. [\WS]

数据类型：int (0-999)

变量类型：UI

解释：进行线激光寻位时，参考的焊缝类型。当传感器类型选择“线激光”，即 SS=1 时显示。

4. [\Para]

数据类型：int (0-99)

变量类型：UI

解释：焊缝类型对应的激光传感器特征参数组号，控制器将下发参数到激光器，进行寻位时激光器将使用该组号对应的特征参数。可选择“不使用”/“常数”/“UI”，其中选择“不使用”时，控制器不下发参数到激光器，并且该参数隐藏。

5. [\SPoint]

数据类型：字符型("laser"/"TCP")

解释：寻位成功时点位记录类型，可选择记录激光检测到的焊缝点或者机器人末端 TCP 位置。可选择“激光检测”/“机器人 TCP”(laser：激光检测。TCP：机器人 TCP)。

6. [\SNum]

数据类型：int (1-99)

解释：寻位时，针对存在噪声数据，必须寻到设置的检测个数以上点位数据，并且两个采集数据之间的采集时间小于寻位滤波时间才认为是寻位成功。可选择“不使用”/“检测个数”，其中选择“不使用”时，该参数隐藏，若选择“检测个数”出厂默认使用 5。

7. [\STime]

数据类型：int (0-99999)

解释：寻位时采集的两个点位数据之间的最大间隔时间，单位 ms，出厂默认使用 200。当参数 6 选择“不使用时”该参数隐藏。

5.15.4. 详细举例

示例 1:

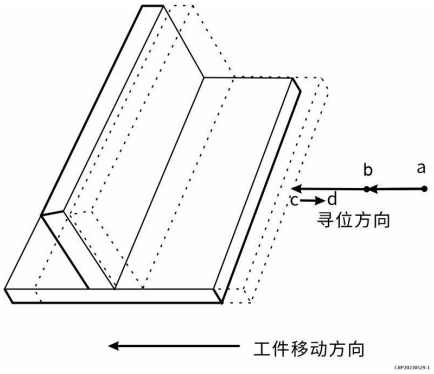


图 5.15.2

```
SearchStart TN=0 //寻位开始，使用 0 号寻位工艺//
MoveL VL=100 PL=0.0 T=1       //a 点，寻位起始点
MoveL VL=100 ACC=1 PL=0.0 T=1 Search GP1 Point //设置寻位置开始点 b，设定位置数据存储在 GP1 变量；寻
c 点，自动返回 d 点//
SearchEnd TN=1  //寻位结束
```

示例 2:

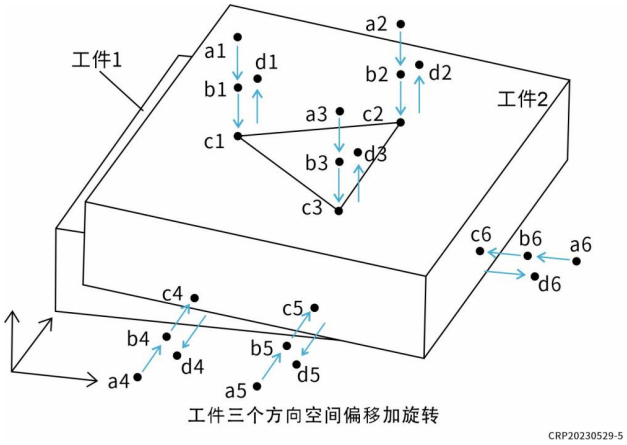


图 5.15.3

```
SearchStart TN=1              //寻位开始，调用寻位工艺号 1
MoveL VL=5 PL=0.0 T=1
MoveL VL=100 PL=0.0 T=1          //寻位开始点 a1
MoveL VL=5 PL=0.0 T=1 Search LP0 point //b1 点寻 c1 点，自动退 d1 点，位置数据存 LP0
MoveL VL=5 PL=0.0 //寻位开始点 a2
MoveL VL=5 PL=0.0 Search LP1 point //b2 点寻 c2 点，自动退 d2 点，位置数据存 LP1
MoveL VL=5 PL=0.0 //寻位开始点 a3
MoveL VL=5 PL=0.0 Search LP2 point //b3 点寻 c3 点，自动退 d3 点，位置数据存 LP2
MoveL VL=5 PL=0.0 //寻位开始点 a4
MoveL VL=5 PL=0.0 Search LP3 point //b4 点寻 c4 点，自动退 d4 点，位置数据存 LP3
MoveL VL=5 PL=0.0 //寻位开始点 a5
MoveL VL=5 PL=0.0 Search LP4 point //b5 点寻 c5 点，自动退 d5 点，位置数据存 LP4
MoveL VL=5 PL=0.0 //寻位开始点 a6
MoveL VL=5 PL=0.0 Search LP6 point //b6 点寻 c6 点，自动退 d6 点，位置数据存 LP5
SearchEnd //寻位结束
CountOffset CT=4 ST=GP0 FP1=LP3 FP2=LP4 FP3=LP5 FP4=LP6 FP5=LP0 FP6=LP1 FP7=LP2 //数据计算。结果
放入 GP1 中。
OffsetStrat GP0 User
MoveL VL=100 PL=0.0
MoveL VL=100 PL=0.0
OffsetEnd
```

示例 3：

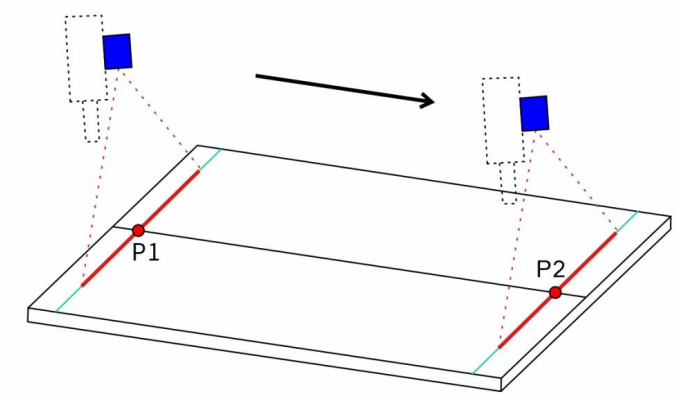


图 5.15.4

```
MoveL VL=100 PL=0.0 T=1          //过渡点
OpenLaser 0 OffsetX=0 OffsetY=0 OffsetZ=0 //打开激光
SearchStart TN=1 SS=1 WS=0 Para=5 SPoint=laser SNum=5 STime=200 //寻位开始，调用寻位工艺号 1，使用线激光寻位，焊缝类型为 5，焊缝参数使用 5 号参数，寻位返回激光检测点数据，寻位检测个数是 5，寻位滤波时间是 200.
MoveL VL=100 PL=0.0 T=1          //寻位开始点
MoveL VL=5 PL=0.0 T=1 Search LP0 point None//进行激光寻位，寻位成功后返回激光检测的点位数据 P1 到 LP0
SearchEnd //寻位结束
CloseLaser //关闭激光结束
MoveL LP0 VL=100 PL=0.0 //机器人运动到 P1 点
MoveL VL=100 PL=0.0 //过渡点
//若需要详细了解激光功能，可查看激光传感器说明书。
```

5.16. SearchEnd---寻位结束

5.16.1. 指令用法

用于停止寻位功能，与 SearchStart 指令配合使用。

5.16.2. 程序运行

执行此指令后，结束寻位相关的功能和动作。

5.16.3. 可变元素

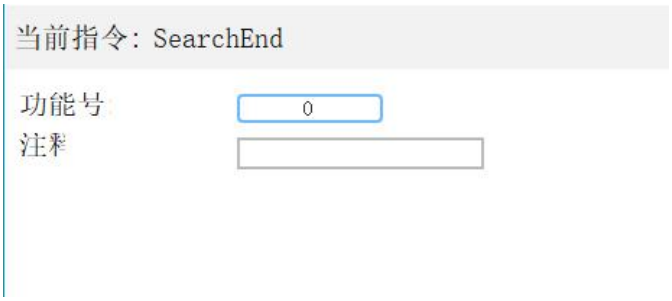


图 5.16.1

SearchEnd TN=[Process]

1. [process]

数据类型：int (0-999)

解释：寻位时使用的寻位工艺号。

5.16.4. 详细举例

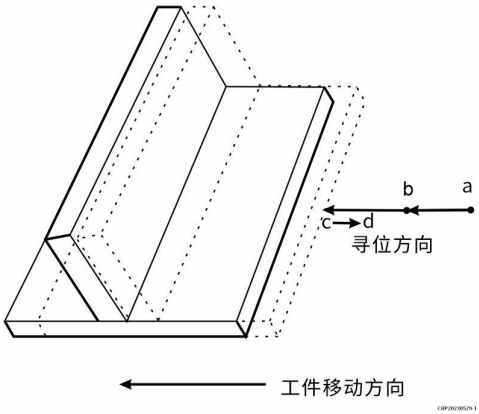


图 5.16.2

```
SearchStart TN=0 //寻位开始，使用 0 号寻位工艺
MoveL VL=100 PL=0.0 T=1      //a 点，寻位起始点
MoveL VL=100 ACC=1 PL=0.0 T=1 Search GP1 Point //设置寻位置开始点 b，设定位置数据存储在 GP1 变量；寻
c 点，自动返回 d 点//
SearchEnd TN=0  //寻位结束
```

5.17. CountOffset---计算偏移指令

5.17.1. 指令用法

用于计算特征点的偏差量位置数据，需要配合寻位指令使用。

5.17.2. 程序运行

执行此指令后，通过输入的特征点数据，计算出对应的偏移量。结合偏移指令可实现整体偏移效果。

5.17.3. 可变元素

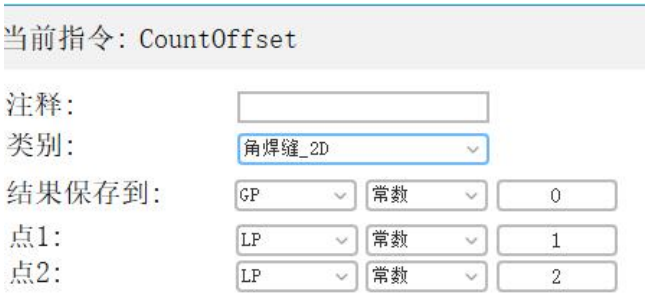


图 5.17.1

CountOffset [CT] [ST] [FP]

1. [CT]

数据选项：常数

变量类型：int（0-5）

解释: 寻位的类型，目前分为角焊缝 1D、2D、3D、2D+、3D+以及内外径方式，分别对应CT0~5。

2. [ST]

数据选项：GP/LP

变量类型：位置变量

解释: 计算出偏移量数据后的变量保存位置。

3. [FP]

数据选项：GP/LP

变量类型：位置变量

解释: 选择寻位时寻到的位置数据变量。

注意：选择变量时，特征点类型要与运动指令寻位时的特征点类型一致，且选择对应的位置变量。如使用 2D+方式时，寻位后找到直线 1 第 1 点位置 GP0、直线 1 第 2 点位置 GP1、直线 2 第 1 点位置 GP2、直线 2 第 2 点位置 GP3。使用 CountOffset 计算偏移量时，需要输入对应的位置变量。如直线 1 上的第一个点，对应的位置变量需选择 GP0(即计算特征点直线 1 第一点与寻位的特征点直线 1 第 1 点对应)；直线 1 上的第二个点，对应的位置变量选择 GP1。以此类推。

图 5.17.2

5.18. SearchLaser---激光搜寻指令

5.18.1. 指令用法

实现点位搜寻、储存搜寻点位、识别焊缝特征及焊缝宽度。

5.18.2. 程序运行

执行此指令后，机器人通过与激光传感器通讯设置的选项，

1-进行点位搜寻：获取当前激光传感器识别到的工件绝对位置，并使用 MoveL 直线运动到该位置；

2-进行点位存储：将激光传感器识别到的工件绝对位置或当前机器人 TCP 位置存入位置变量；

3-焊缝识别：获取当前激光传感器识别到的焊缝类型；

4-焊缝宽度识别：获取当前当前激光传感器识别到的焊缝宽度。

5.18.3. 可变元素

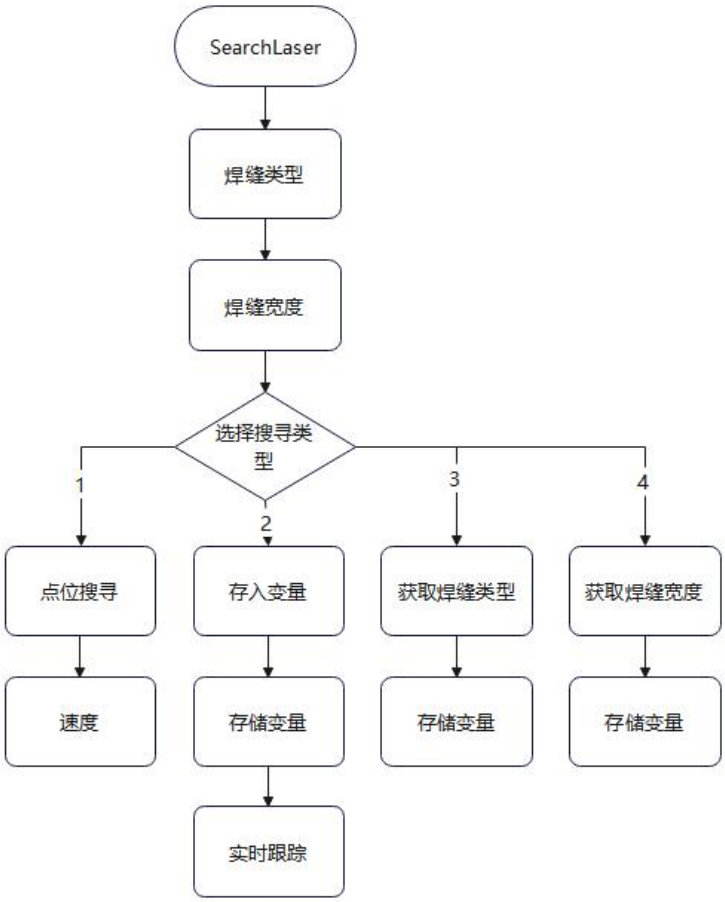


图 5.18.1

SearchStart Wtype=[number] [\para] SType=[Process] （SPeed=[number] / [Point] [Track] / [data1] / [data2]）

1. [number]

数据类型：int (0-9)

变量类型：UI

解释：进行线激光寻位时，参考的焊缝类型。

2. [\para]

类型：int (0~99)

变量类型：UI

解释：焊缝类型对应的激光传感器特征参数，控制器将下发参数到激光器。可选择“不使用”/“常数”/“UI”，其中选择“不使用”时，控制器不下发参数到激光器，并且该参数隐藏。

3. [Process]

类型：char（“Move”/“Save”/“WeldType”/“WeldWidth”）

解释：进行激光搜寻的类型，可选择“点位搜寻”/“存入变量”/“获取焊缝类型”/“获取焊缝宽度”。（Move：点位搜寻”；Save：“存入变量”；WeldType：获取焊缝类型”；WeldWidth：“获取焊缝宽度”）

4. [number]

类型：int (1~100)

解释：进行点位搜寻时，搜寻到点位后机器人直线运动到点的速度。(只有搜寻类型选择“点位搜寻”时才可进行设置)

5. [Point]

变量类型 1：GP（全局位置变量）、LP（局部位置变量）

变量类型 2：UI

示例：GP5、GP(UI2)

解释：进行存入变量搜寻时，将搜寻的点位存入位置变量。(只有搜寻类型选择“存入变量”时才可进行设置)

6. [Track]

数据类型：char(“ON”，“OFF”)

解释：进行存入变量搜寻时，是否继续保持实时跟踪的状态。(只有搜寻类型选择“存入变量”时才可进行设置)

7. [data1]

变量类型 1：GI（全局整型变量）、LI（局部整型变量）

变量类型 2：UI

示例：GI5、GI(UI2)

解释：进行获取焊缝类型搜寻时，将搜寻的焊缝类型存入整型变量。(只有搜寻类型选择“获取焊缝类型”时才可进行设置)

8. [data2]

变量类型 1：GR（全局浮点型变量）、LR（局部浮点型变量）

变量类型 2：UI

示例：GR5、GR(UI2)

解释：进行获取焊缝宽度搜寻时，将搜寻的焊缝宽度存入整型变量。(只有搜寻类型选择“获取焊缝宽度”时才可进行设置)

5.18.4. 详细举例

示例 1：

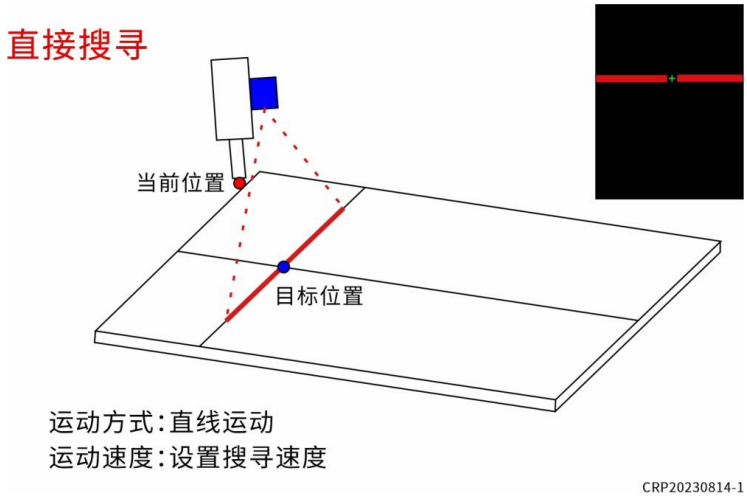


图 5.18.2

```
MoveL VL=100 PL=0.0 T=1          //直线运动到当前位置
OpenLaser 0 OffsetX=0 OffsetY=0 OffsetZ=0 //打开激光，且使用 0 号跟踪工艺
SearchLaser Wtype=0 Para=2 Stype=Move Speed=20 //开始激光搜寻，搜寻类型为点位搜寻，焊缝参数设置为
2，搜寻速度为 20mm/s
Time=1000
CloseLaser //关闭激光
MoveL VL=100 PL=0.0 T=1          //直线运动回到当前位置
```

示例 2:

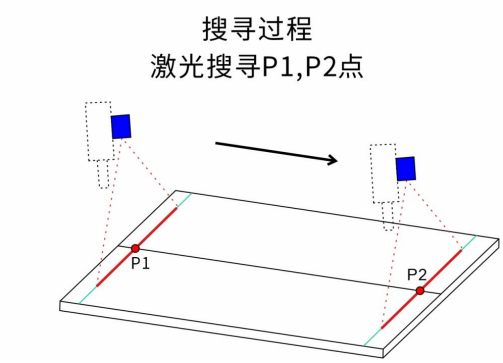


图 5.18.3

```
MoveL VL=100 PL=0.0 T=1          //直线运动到 P1 点上方
OpenLaser 0 OffsetX=0 OffsetY=0 OffsetZ=0 //打开激光，且使用 0 号跟踪工艺
SearchLaser Wtype=0 Para=2 Stype=Save GP1 Track=OFF //开始激光搜寻，搜寻类型为存入变量，将搜寻到的点位存入 GP0，实时跟踪关闭
MoveL VL=100 PL=0.0 T=1          //直线运动到 P2 点上方
SearchLaser Wtype=0 Para=2 Stype=Save GP1 Track=OFF //开始激光搜寻，搜寻类型为存入变量，将搜寻到的点位存入 GP1，实时跟踪关闭
CloseLaser //关闭激光
MoveL VL=100 PL=0.0 T=1          //直线运动回到当前位
```

示例 3:

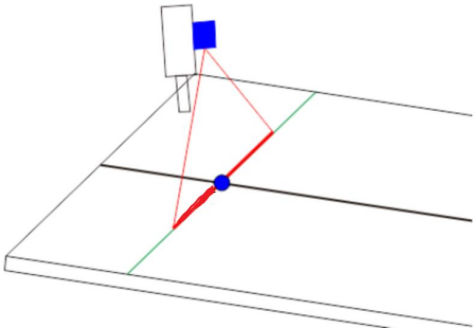


图 5.18.4

```
MoveL VL=100 PL=0.0 T=1          //直线运动到目标点
OpenLaser 0 OffsetX=0 OffsetY=0 OffsetZ=0 //打开激光，且使用 0 号跟踪工艺
SearchLaser Wtype=0 Para=2 Stype=WeldType GI(UI1) //开始激光搜寻，搜寻类型为获取焊缝类型，将搜寻到的类型存入变量 GI(UI1)中。
SearchLaser Wtype=0 Para=2 Stype=WeldWidth GR3 //开始激光搜寻，搜寻类型为获取焊缝宽度，将搜寻到的点位存入 GR3 中。
CloseLaser //关闭激光
MoveL VL=100 PL=0.0 T=1          //直线运动回到初始点
```

5.19. OpenLaserTrack---开启激光跟踪指令

5.19.1. 指令用法

用于连续获取激光传感器的采集数据，需要配合关闭激光跟踪指令使用。

5.19.2. 程序运行

执行此指令后，机器人可以不断从激光传感器获取数据直到跟踪状态结束。

5.19.3. 可变元素

OpenLaserTrack WType=[number] Track=[Process] [Para] [\state]

1. [number]

数据类型：int (0~99)

变量类型：UI

解释：进行激光跟踪时，参考的焊缝类型。

2. [Process]

数据类型：char (“Real” / “Record” / “Pcloud”)

解释：设置激光跟踪的跟踪模式，可选择“实时”/“记录”/“点云”（Real：实时；Record：“记录”；Pcloud：“点云"）。实时模式时将激光传感器反馈的焊缝轨迹数据采集，且机器人随采集数据移动；；记录模式时将激光传感器反馈的焊缝轨迹数据采集，但不改变机器人运动轨迹；点云模式是进行扫描焊缝轨迹，生成焊缝轨迹数据，用于管板等智能化焊接应用中。

3. [Para]

数据类型：int (0~99)

变量类型：UI

解释：激光跟踪的跟踪模式，选择“点云”时设置的焊缝参数(焊缝类型对应的激光传感器特征参数，控制器将下发参数到激光器)。

4. [\state]

数据类型：int (0~99)

变量类型：UI

解释：：激光跟踪的跟踪模式，选择“点云”时检测时返回的点云状态值。

图 5.19.1：开始激光跟踪界面 1

图 5.19.2：开始激光跟踪界面 2

5.19.4. 详细举例

示例 1：

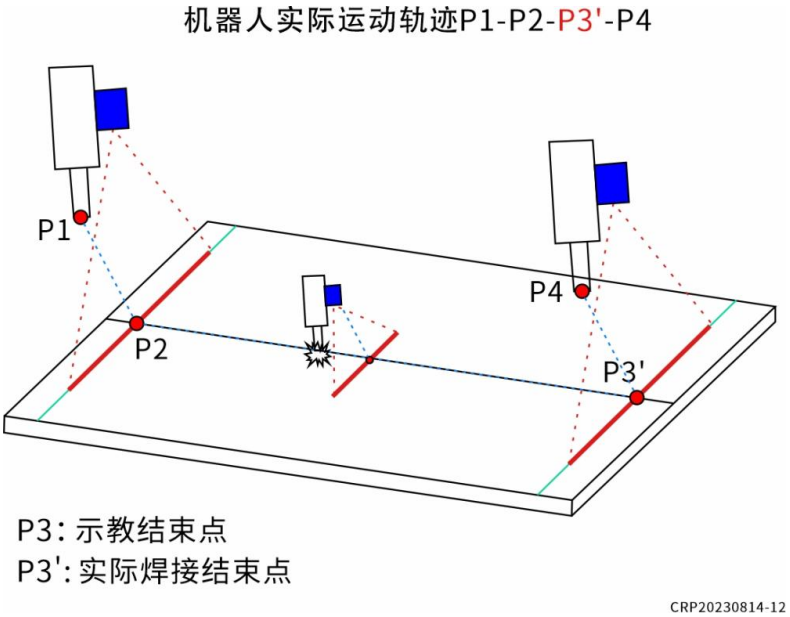


图 5.19.3

```
MoveL VL=100 PL=0.0 T=1          //到达搜寻点 P1
OpenLaser 0 OffsetX=0 OffsetY=0 OffsetZ=0 //打开激光，使用 0 号跟踪工艺
SearchLaser Wtype=0 Para=2 Stype=Move Speed=20 //开始激光搜寻，搜寻类型为点位搜寻，焊缝参数设置为
2，搜寻速度为 20mm/s
OpenLaserTrack Wtype=0 Track=Real  //开始激光跟踪状态，跟踪模式为实时
ArcOn(1)    VL=5   //起弧，焊接速度为 5mm/s
MoveL VL=100 PL=0.0 T=1 //运动到结束点 P3
CloseLaserTrack //关闭激光跟踪状态
CloseLaser //关闭激光
ArcOff  //灭弧
MoveL VL=100 PL=0.0 T=1   //运动到过渡点 P4
```

5.20. CloseLaserTrack---关闭激光跟踪指令

5.20.1. 指令用法

用于关闭激光跟踪的状态，需要配合打开激光跟踪指令使用。

5.20.2. 程序运行

执行此指令后，将关闭连续获取激光传感器的采集数据的状态。

5.20.3. 可变元素

无

5.20.4. 详细举例

无

5.21. OpenLaser---打开激光指令

5.21.1. 指令用法

建立与激光跟踪器的连接并控制打开激光光源。

5.21.2. 程序运行

执行此指令后，控制器将建立与激光跟踪器的连接，并控制激光跟踪器打开激光光源。

5.21.3. 可变元素

OpenLaser [Process] [offset]

1. [Process]

数据类型：int (0-9)

变量类型：UI

解释：激光跟踪工艺号。

2. [offset]

类型：float (-1000-1000)

解释：对获取到的激光点位数据都叠加上偏移效果，偏移参考坐标系为当前使用的工具坐标系。

5.21.4. 详细举例

示例 1：

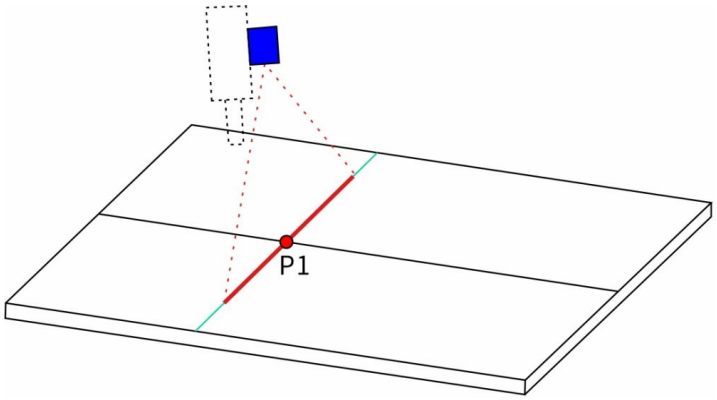


图 5.21.1

```
MoveL VL=100 PL=0.0 T=1          //过渡点
OpenLaser 0 OffsetX=0 OffsetY=0 OffsetZ=0 //打开激光
SearchStart TN=1 SS=1 WS=0 Para=5 SPoint=laser SNum=5 STime=200 //寻位开始，调用寻位工艺号 1，使用线激光寻位，焊缝类型为 5，焊缝参数使用 5 号参数，寻位返回激光检测点数据，寻位检测个数是 5，寻位滤波时间是 200.
MoveL VL=100 PL=0.0 T=1          //寻位开始点
MoveL VL=5 PL=0.0 T=1 Search LP0 point //进行激光寻位，寻位成功后返回激光检测的点位数据 P1 到 LP0
SearchEnd //寻位结束
CloseLaser //关闭激光结束
MoveL LP0 VL=100 PL=0.0 //机器人运动到 P1 点
MoveL VL=100 PL=0.0 //过渡点
```

5.22. CloseLaser---关闭激光指令

5.22.1. 指令用法

控制关闭激光光源并断开与激光跟踪器的连接。

5.22.2. 程序运行

执行此指令后，控制器控制激光跟踪器关闭激光光源并断开与激光跟踪器的连接。

5.22.3. 可变元素

暂无

6. 特殊指令

6.1. AxisAct---轴禁止开启

6.1.1. 指令用法

用于禁止机器人或机械单元组的指定轴。轴禁止后需要轴禁止结束指令解除轴禁止。

6.1.2. 应用举例

例 1:

```
AxisAct B1 S1
```

运行该行程序时，机器人系统将禁止机械单元 B1 的 S1 轴移动。、

6.1.3. 程序运行

执行此程序行，机器人系统将会禁止指令中设置的轴，需要使用轴禁止解除指令解除相应被禁止的轴。

6.1.4. 可变元素

AxisAct [B] [E]

1. [B]

变量类型：B(机械单元)

解释：机器人系统在添加机械单元时配置的机械单元。

2. [E]

变量类型：E(轴)

解释：机器人系统在添加机械单元时，机械单元内的轴。

6.2. AxisDeact---轴禁止解除

6.2.1. 指令用法

用于解除机器人或机械单元组被轴禁止指令禁止的轴组。

6.2.2. 应用举例

例 1：

AxisDeact B1 E1

远行完成指令后将解除机械单元 B1 中的 E1 轴轴禁止。

6.2.3. 程序运行

执行此程序行，机器人系统将会解除指令中设置相应的轴组。

6.2.4. 可变元素

AxisDeact [B] [E]

1. [B]

变量类型：B(机械单元)

解释：机器人系统在添加机械单元时配置的机械单元。

2. [E]

变量类型：E(轴)

解释：机器人系统在添加机械单元时，机械单元内的轴。指令中可以解除被禁止的多个轴。

6.3. AxisAct 与 AxisDeact 应用示例

```
.....
AxisAct B1 S1      ;禁止 B1 单元的 S1 轴移动
.....
AxisDeact B1 S1   ; 解除禁止 B1 单元的 S1 轴移动
.....
```

6.4. ClkStart---计时开始

6.4.1. 指令用法

ClkStart 用于启动计时指令，当 ClkStart 执行后，软件内部开始计时.

6.4.2. 可变元素

ClkStart [ID] [Var]

图 6.4.1

1. [ID]

变量类型：Num，范围 0~4

计时编号，与结束计时编号相对应，用于区分多个计时程序

2. [Var]

变量类型：GR/GI/UI

计时时间存放变量

6.4.3. 应用举例

例 1:

ClkStart(0) GR0

…

ClkEnd(0)

从当前行开始计时，当执行到 ClkEnd 时结束计时并将计时时间单位 s 累加到 GR0;

例 2:

ClkStart(0) GI0

…

ClkEnd(0)

从当前行开始计时，当执行到 ClkEnd 时结束计时并将计时时间单位 ms 累加到 GI0;

其他说明：GR 统计时间单位是 s，GI\UI 统计单位是 ms；ClkStart 到 ClkEnd 是统计区间累加计时时间并放置到指定的变量，如无特殊需求请先插入指令：SET 指定变量=0，再插入 ClkStart (0) 指定变量

6.5. ClkEnd---计时结束

6.5.1. 指令用法

ClkStart 用于结束计时指令。

6.5.2. 可变元素

ClkEnd [ID]

图 6.5.1

1. **[ID]**

变量类型：Num，范围 0~4

计时编号，与计时开始编号相对应，用于区分多个计时程序

6.5.3. 程序运行

当 ClkEnd 执行后，软件内部停止计时，并将计算结果累加至指定变量

6.5.4. 应用举例

例 1:

ClkStart(0) GR0

...

ClkEnd(0)

从 ClkStart 执行到 ClkEnd 计时结束，并将计时结果累加到 GR0 变量中;

6.6. OffsetOn---整体偏移开启

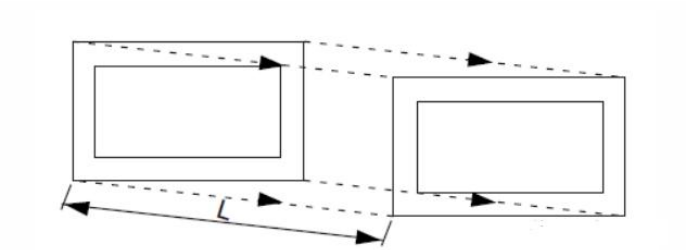


图 6.6.1

6.6.1. 指令用法

使得整体偏移开启指令与整体偏移结束指令之间的运动指令叠加偏移效果，调用偏移量以指定坐标系进行偏移。

6.6.2. 指令逻辑

1、叠加偏移效果的运动指令为 MoveL、MoveC 以及 MoveJ 指令，不包括绝对关节指令 MoveAbsJ。

2、若指定坐标系为 User/Tool，则运动指令按照插入时参考的用户/工具坐标系进行偏移，如运动指令的参考坐标系为 User3/Tool2，则实际点位调用偏移量以坐标系 User3/Tool2 进行偏移。

3、若偏移有嵌套，即整体偏移中又包含单点的偏移，则先进行整体偏移，后进行单点偏移。

6.6.3. 可变元素

图 6.6.2

格式一：OffsetOn [variable] [number] [coordinate system]

1. [variable]

变量类型：GP/LP。

解释：位置型变量，可选择 GP 及 LP 类型。

2. [number]

变量类型：整型变量。

解释：为 GP/LP 变量的序号。

3. [coordinate system]

变量类型：User/Tool。

解释：偏移参考坐标系，可选择 User 和 Tool。

格式二(附加功能-自定义文本)：OffsetOn [X] [Y] [Z] [Rx] [Ry] [Rz] [coordinate system] [Ex1] [Ex2] [Ex3] [Ex4] [Ex5] [Ex6]

1. [X]

变量类型：常数/GR/LR/GI/UI。

解释：[X]为沿参考坐标系 X 轴方向上移动的变化量。

2. [Y]

变量类型：常数/GR/LR/GI/UI。

解释：[Y]为沿参考坐标系 Y 轴方向上移动的变化量。

3. [Z]

变量类型：常数/GR/LR/GI/UI。

解释：[Z]为沿参考坐标系 Z 轴方向上移动的变化量。

4. [Rx]

变量类型：常数/GR/LR/GI/UI。

解释：[Rx]为绕参考坐标系 X 轴旋转的变化量。

5. [Ry]

变量类型：常数/GR/LR/GI/UI。

解释：[Ry]为绕参考坐标系 Y 轴旋转的变化量。

6. [Rz]

变量类型：常数/GR/LR/GI/UI。

解释：[Rz]为绕参考坐标系 Z 轴旋转的变化量。

7. [coordinate system]

变量类型：User/Tool。

解释：偏移参考坐标系，可选择 User 用户和 Tool 工具坐标系。

8. [Ex1]

变量类型：常数/GR/LR/GI/UI。

解释：[Ex1]为外部轴 1 的变化量。

9. [Ex2]

变量类型：常数/GR/LR/GI/UI。

解释：[Ex2]为外部轴 2 的变化量。

10. [Ex3]

变量类型：常数/GR/LR/GI/UI。

解释：[Ex3]为外部轴 3 的变化量。

11. [Ex4]

变量类型：常数/GR/LR/GI/UI。

解释：[Ex4]为外部轴 4 的变化量。

12. [Ex5]

变量类型：常数/GR/LR/GI/UI。

解释：[Ex5]为外部轴 5 的变化量。

13. [Ex6]

变量类型：常数/GR/LR/GI/UI。

解释：[Ex6]为外部轴 6 的变化量。

格式三(附加功能-自定义文本)：OffsetOn [X] [Y] [Z] [Rx/Ry/Rz] [coordinate system] [Ex1] [Ex2] [Ex3] [Ex4] [Ex5] [Ex6]

1. [X]

变量类型：常数/GR/LR/GI/UI。

解释：[X]为沿参考坐标系 X 轴方向上移动的变化量。

2. [Y]

变量类型：常数/GR/LR/GI/UI。

解释：[Y]为沿参考坐标系 Y 轴方向上移动的变化量。

3. [Z]

变量类型：常数/GR/LR/GI/UI。

解释：[Z]为沿参考坐标系 Z 轴方向上移动的变化量。

4. [Rx/Ry/Rz]

变量类型：常数/GR/LR/GI/UI。

解释：[Rx]为绕参考坐标系 X 轴旋转的变化量。[Ry]为绕参考坐标系 Y 轴旋转的变化量。[Rz]为绕参考坐标系 Z 轴旋转的变化量。

5. [coordinate system]

变量类型：User/Tool。

解释：偏移参考坐标系，可选择 User 用户和 Tool 工具坐标系。

6. [Ex1]

变量类型：常数/GR/LR/GI/UI。

解释：[Ex1]为外部轴 1 的变化量。

7. [Ex2]

变量类型：常数/GR/LR/GI/UI。

解释：[Ex2]为外部轴 2 的变化量。

8. [Ex3]

变量类型：常数/GR/LR/GI/UI。

解释：[Ex3]为外部轴 3 的变化量。

9. [Ex4]

变量类型：常数/GR/LR/GI/UI。

解释：[Ex4]为外部轴 4 的变化量。

10. [Ex5]

变量类型：常数/GR/LR/GI/UI。

解释：[Ex5]为外部轴 5 的变化量。

11. [Ex6]

变量类型：常数/GR/LR/GI/UI。

解释：[Ex6]为外部轴 6 的变化量。

6.6.4. 应用举例

示例 1：

```
OffsetOn 100 0 0 20 20 20 User 0 0 0 0 0 0; //以设置值(坐标 X 轴正方向移动 100mm，分别绕 X、Y、Z 轴旋转 20°）作为偏移变量，偏移指定参考坐标系为用户，开启偏移。
MoveL VL=500mm/s PL=0.0 T=1 U=1;  //叠加偏移。
MoveL VL=50mm/s PL=0.0 T=1 U=1;  //叠加偏移。
OffsetOff; //整体偏移结束
```

解释：开启整体偏移，偏移量为：100 0 0 20 20 20 ，分别代表 X、Y、Z、A、B、C 上的偏移数据。参考坐标系为用户。按照运动指令中的坐标系进行偏移，如指令中参考坐标系都是用户 1，则指令点位偏移参考用户坐标系 1，按照偏移量(100 0 0 20 20 20)进行偏移，当执行整体偏移结束指令时停止偏移效果。

示例 2：

```
OffsetOn  GP1 User;  //以  GP1  值作为偏移变量，偏移指定坐标系为用户，开启偏移。
MoveL VL=500mm/s PL=0.0 T=1 U=1;  //叠加偏移效果
MoveL VL=50mm/s PL=0.0 T=1 U=1;  //叠加偏移效果
OffsetOff;  //整体偏移结束
```

解释：开启整体偏移，偏移量为位置变量 GP1 值，参考坐标系为用户。按照运动指令中的坐标系进行偏移，如指令中参考坐标系都是用户 1，则指令点位参考用户坐标系 1 按照偏移量 GP1 进行偏移，当执行整体偏移结束指令时，停止偏移效果。

示例 3：

```
OffsetOn 100 0 0 \Rz=20 Tool;  //以设定值作为偏移变量，偏移指定坐标系为工具，开启偏移。
MoveL VL=500mm/s PL=0.0 T=1 U=1 OffsetOn GP1 User;  //叠加单点和整体偏移效果
MoveL VL=50mm/s PL=0.0 T=1 U=1;  //叠加偏移效果
OffsetOn  GP1 User;  //以  GP1  值作为偏移变量，偏移指定坐标系为用户，开启偏移。
MoveL VL=500mm/s PL=0.0 T=1 U=1;  //叠加偏移效果
MoveL VL=50mm/s PL=0.0 T=1 U=1;  //叠加偏移效果
OffsetOff;  //整体偏移结束
```

解释：

开启整体偏移，偏移量为 X=100 Y、Z=0、C=20，参考坐标系为工具。第一条 MoveL 运动指令中附加了单点偏移。则先进行整体偏移，参考工具坐标系 1 和偏移量 X=100 Y、Z=0、C=20 进行偏移。后进行单点偏移，参考用户坐标系 1 和偏移量 GP1 进行再次偏移。第二条 MoveL 运动指令的点位参考工具坐标系 1 和偏移量 X=100 Y、Z=0、C=20 进行偏移。

运行第二条整体偏移指令，偏移量为 GP1，参考坐标系为用户。则之后的 MoveL 运动指令的点位参考用户坐标系 1 和偏移量 GP1 进行偏移。

当执行整体偏移结束指令时停止偏移效果。、

6.7. OffsetOff---整体偏移关闭

6.7.1. 指令用法

与整体偏移指令成对，整体偏移关闭。

6.7.2. 指令逻辑

取消整体偏移效果

6.7.3. 可变元素

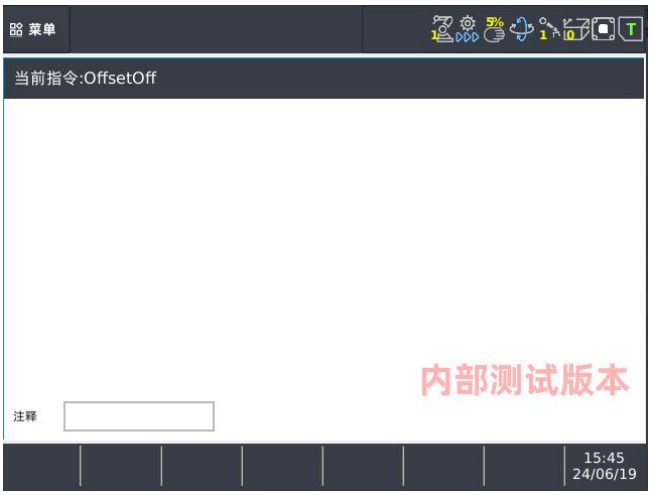


图 6.7.1

无

6.7.4. 应用举例

示例 1：

```
OffsetOn GP1 User;  //以 GP1 值作为偏移变量，偏移指定坐标系为用户，开启偏移。
MoveL VL=500mm/s PL=0.0 T=1 U=1;  //叠加偏移效果
MoveL VL=50mm/s PL=0.0 T=1 U=1;  //叠加偏移效果
OffsetOff;  //整体偏移结束
```

6.8. LatchSignalPos --- 设置锁存位置

CrobotpOS 版本要求 1.13.0 及以上，低于此版本的软件请按要求进行升级，否则功能不能正常。

6.8.1. 指令用法：

监控指定信号变化后，将机器人的位置锁存下来，并通过 GetLatchPos 获取坐标位置。

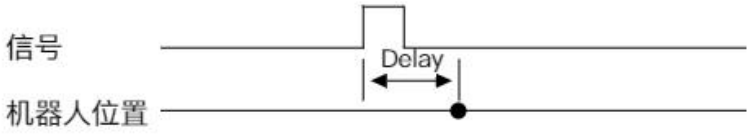


图 6.8.1

如上图所示：当检测到信号从 OFF 变为 ON 状态后，开始计时经过指定的延迟时间记录机器人的当前位置。

6.8.2. 程序运行：

当执行 LatchSignalPos 当判断条件成立后会锁存机器人当前位置。

6.8.3. 可变元素：

LatchSignalPos [Id] [Condition] ==[Stmt] [Delay]

输入参数 1

[Id]

常数：范围 0-7

变量：UI/GIN

解释：锁存编号，与获取锁存编号对应，用于区分多个信号的位置获取。

输入参数 2

[num] [Condition]

变量：X/Y/M/SX/SY/SM

变量号：常数/UI/GIN

条件：ON/OFF

解释：对采集信号触发判断条件。

输入参数 3

[Delay]

常数：范围 0-65535（单位 ms）。

变量：UI/GIN

解释：延时指定时间记录机器人位置信息。

6.8.4. 应用举例：

举例 1

LatchSignalPos Id=0 Y8==ON Delay=5

监控采集 Y8 信号从 OFF 变成 ON，延迟 5ms 进行锁存机器人的当前位置信息。

举例 2

LatchSignalPos Id=UI0 Y(UI0)==ON Delay=UI1

监控采集获取 UI0 当前值作为 Y 信号从 OFF 变成 ON，延迟时间获取 UI1 变量值作为延迟时间。

6.9. GetLatchPos --- 获取锁存位置

CrobotpOS 版本要求 1.13.0 及以上，低于此版本的软件请按要求进行升级，否则功能不能正常。

6.9.1. 指令用法：

获取锁存的机器人位置信息。

6.9.2. 程序运行：

当执行 GetLatchPos 后获取 LatchSignalPos 锁存的位置返回到指定的位置变量中。

6.9.3. 可变要素：

GetLatchPos [Id] [Point] [ToolId] [UserId] [TimeOut] [State]

输入参数 1

[Id]

常数：锁存编号 0-7。

变量：UI/GIN

解释：与锁存位置指令编号对应，用于区分多个信号的位置获取。

输出信息 1

[point]

变量：LP/GP

输入参数 2

[ToolId]

常数：范围 0-49

变量：UI/GIN

解释：返回位置点需要指定的工具号下。

输入参数 3

[UserId]

常数：范围 0-49

变量：UI/GIN

解释：返回位置点需要指定的用户号下。

输入参数 4

[TimeOut]

常数：范围 0-65535

变量：UI/GIN

解释：指定指令执行的超时时间。

输出信息 2

[state]

变量：LI/GI

解释：-1 获取失败，1 获取成功。

6.9.4. 应用举例：

举例 1

GetLatchPos Id=0 Point=LP0 T=0 U=0 TimeOut=2000 State=LI0 获取编号为 0 的锁存位置，将锁存位置转换到指定用户坐标系下，并以指定的工具表示，最终放置到 LP0 中。设置 2000 毫秒的超时时间，如获取锁存位置成功，则指令向后执行，并返回 1 到状态变量 LI0 中代表获取成功；如果超过 2000 毫秒未获取到锁存位置，则返回-1 到状态变量中代表获取失败。举例 2

GetLatchPos Id=UI0 Point=LP0 T=UI1 U=UI2 TimeOut=UI3 State=LI0 获取 UI0 的值作为获取锁存位置指令的对应编号，通过 UI1 的值作为工具号，UI2 的值作为用户号，将锁存位置转换到指定用户坐标系下，并以指定的工具表示，最终放置到 LP0 中。从 UI3 获取值作为超时时间。

6.10. TorqueLimit --- 单轴力矩限制

6.10.1. 指令用法：

通过下发指令限制百分比，当伺服超出指定限制值时进行报警。

6.10.2. 程序运行：

当执行 TorqueLimit 后将按指定的力矩百分比，限制运动过程力矩，当超出力矩会立即报警。

6.10.3. 可变要素：

TorqueLimit [Axisnum] =[value]

输入参数 1：

[Axisnum]

常数：1-8

变量：UI/GIN

解释：输入需要限制的对应轴号，仅限 220V 驱控一体

输入参数 2：

[value]

单位：百分比

常数：0-300

变量：UI/GIN

解释：限制力矩对应 1-300%力矩，设置 0%为关闭当前力矩检测；

6.10.4. 应用举例：

举例 1:

TorqueLimit J1=80

控制器对 J1 轴设置下发 80%的限制力矩值，当伺服输出力矩超出 80%伺服进行报警

举例 2:

TorqueLimit J(UI0)=UI1

控制器对从 UI0 获取值对应轴设置限制力矩值，当伺服输出力矩超出 UI1 中的值伺服进行报警

7. 碰撞指令

7.1. ShckSet --- 碰撞等级设置

7.1.1. 指令用法

```
ShckSet (1) //设置碰撞检测等级指令
.....
ShckRst ;  //碰撞等级解除指令
```

使用碰撞预设号 1 号预设值进行碰撞检测

7.1.2. 程序用法

在程序里使用碰撞检测的指令时调用对应碰撞检测文件号，如果不使用该指令，则按照系统默认最大碰撞等级检测。

7.1.3. 可变元素

ShckSet [Id]

输入参数 1

[Id]

常数：预设号 1-7。

解释：Id 为碰撞检测预设文件号。

7.1.4. 应用举例

```
ShckSet (1) //设置碰撞等级 1 预设值

MoveL VL=50MM/S PL=0.0 T=1 //直线运动机器人

Dout Y0=ON //放工件

MoveL VL=50MM/s PL=0.0 T=2 //直线运动机器人

ShckRst //碰撞等级 1 解除
```

机器人系统切换 1 号碰撞等级预设文件号提高灵敏度检测等级，进行放置工件动作。

7.2. ShckRst --- 碰撞等级解除

7.2.1. 指令用法

```
ShckSet (1) //设置碰撞检测等级指令

.....

ShckRst ;  //碰撞等级解除指令
```

碰撞等级解除，解除前面碰撞等级设置使用系统预设默认最大等级检测。

7.2.2. 程序用法

执行此程序行，将使用的碰撞预设文件号更改为默认文件号，且默认文件号预设值出厂预设，无法修改。

7.2.3. 可变元素

ShckRst

无

7.2.4. 应用举例

```
ShckSet (1) //设置碰撞等级 1 预设值
MoveL VL=50MM/S PL=0.0 T=1 //直线运动机器人
Dout Y0=ON ； %放工件
MoveL VL=50MM/s PL=0.0 T=2 //直线运动机器人
ShckRst //碰撞等级 1 解除
```

机器人系统切换 1 号碰撞等级预设文件号提高灵敏度检测等级，进行放置工件动作。当 ShckRst 指令执行后解除前面碰撞等级设置使用系统预设默认最大等级检测。

7.3. ShckAct --- 碰撞检测激活

7.3.1. 指令用法

临时激活碰撞检测功能。切换示教模式，关机重启时，执行碰撞检测激活指令失效。临时作用，此指令不会改变碰撞检测预设文件中设置的“是否启用碰撞检测”的状态。

7.3.2. 程序用法

执行此程序行，机器人系统会临时激活碰撞检测功能。切换示教模式，关机重启时执行碰撞检测禁用指令失效。

7.3.3. 可变元素

ShckAct

无

7.3.4. 应用举例

例 1:

```
MoveL VL=200MM/S PL=0.0 T=1     //必须使用正确的工具，T1 是抓取的负载工具
ShckDeact   //工具重量过大，碰撞检测禁用指令
MoveL VL=1000MM/S PL=9.0 T=1   // 中间过程路径点位
MoveL VL=1000MM/S PL=9.0 T=1
DOUT Y#(0)=ON   //放工件
ShckAct   //碰撞检测激活，放完工具后负载小
MoveL VL=50MM/S PL=0.0 T=1   //运动指令
```

7.4. ShckDeact --- 碰撞检测禁用

7.4.1. 指令用法

执行此程序行，机器人系统会临时禁用碰撞检测功能。切换示教模式，关机重启时执行碰撞检测禁用指令失效。临时作用，此指令不会改变碰撞检测预设文件中设置的“是否启用碰撞检测”的状态。

7.4.2. 程序用法

执行此程序行，机器人系统会临时禁用碰撞检测功能。切换示教模式，关机重启时，执行碰撞检测禁用指令失效。主要用于某些场景需要的力较大，会报警碰撞，临时关闭，执行操作后开启。

7.4.3. 可变元素

ShckDeact

无

7.4.4. 应用举例

```
MoveL VL=200MM/S PL=0.0 T=1     //必须使用正确的工具，T1 是抓取的负载工具
ShckDeact   //工具重量过大，碰撞检测禁用指令
MoveL VL=1000MM/S PL=9.0 T=1   // 中间过程路径点位
MoveL VL=1000MM/S PL=9.0 T=1
DOUT Y#(0)=ON   //放工件
ShckAct   //碰撞检测激活，放完工具后负载小
MoveL VL=50MM/S PL=0.0 T=1   //运动指令
```

8. 视觉指令

8.1. RunVision 指令与 CloseVision 指令

8.1.1. 指令用法

RunVision (1) //运行视觉工艺 1

.....

CloseVision(1)； //关闭视觉工艺 1

开启视觉工艺与关闭视觉工艺，设置好的视觉功能号与视觉指令绑定。

8.1.2. 程序用法

RunVision (1)： 执行此程序行，开启视觉工艺，并使用工艺号 0 的设置。

CloseVision(1)；执行此程序行，关闭视觉工艺 1。

8.1.3. 可变元素

RunVision ([A])

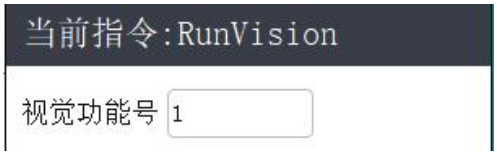


图 8.1.1

1. [A]

变量类型：整型变量。

解释：[A]为视觉工艺文件号，范围为 0 到 9。

CloseVision 关闭视觉指令：

与“RunVision 运行视觉”指令相同。

8.2. GetVisionData 从视觉缓冲区读取数据 指令

8.2.1. 指令用法

GetVisionData (1) blockTime=0 result GI=0 // 从视觉缓冲区读取数据

.....

如果选择了跟踪，视觉转换的数据就放在跟踪缓冲区，而不是视觉缓冲区，就需要从跟踪缓冲区中得到数据，得到跟踪缓冲区的数据，放在 GP50、GP51 中。若使用 1 号跟踪号，必须与视觉工艺里使用的跟踪工艺号相同。

8.2.2. 可变元素

GetVisionData [Process] [Point] [PrePoint] [Height] [blockTime] [result]

图 8.2.1

1. [Process]

视觉工艺号，范围 0-9。

2. [Point]

抓取点位置存放地址设置。可选变量 GP、LP，范围 0-1023。

3. [PrePoint]

准备抓取点位存放地址设置。可选变量 GP、LP，范围 0-1023。

4. [Height]

准备抓取点高度偏移（Z 偏移）设置，输入数据可以为正负，数据类型为浮点数，默认为 0。

5. [blockTime]

阻塞时间，单位 ms。设置方法如下：

- 0：一直等待获取视觉数据，直到成功获取；
- 其他：等待的时长，超时则不等待。

6. [result]

为执行结果，保存指令获取数据的执行结果，-1 表示未获取到，1 表示成功获取。

8.3. GetVisionUser 获取视觉用户坐标系指令

8.3.1. 指令用法

```
GetVisionUser (0) User=1 // 使用视觉工艺 0，获取用户坐标系存为用户坐标 1
.....
```

指定工艺号，使用对应视觉工艺号设置的通信协议去获取和计算用户坐标系。

8.3.2. 可变元素

GetVisionUser [Process] [\User]

图 8.3.1

1. [Process]

视觉工艺号，范围 0-9。

指定工艺号，使用对应视觉工艺号设置的通信协议去获取和计算用户坐标系。通讯协议设置“XYZφ”，则需要获取 3 个数据。按照 ORG、XX、YY 的顺序进行

解析并计算到指定的用户坐标系号中。通讯协议设置“XYZABC”，则只需获取一个

数组，数据作为用户坐标系结果直接放到对应的用户坐标系。

2. [User]

指定放到的对应的用户坐标系，写入过程为暂时的。开始写入的时间节点为执行“GetVisionUser”指令执行，结束写入的时间节点为“CloseVision”或钥匙开关切换状态时。

8.4. ClearVisionData 清除视觉数据指令

8.4.1. 指令用法

ClearVisionUser (0) // 清除视觉工艺 0 的视觉数据

.....

工件位置改变、工件个数发生改变时，可通过该指令清除视觉缓存区数据。

8.4.2. 可变元素

ClearVisionUser [Process]

图 8.4.1

1. [Process]

视觉工艺号，范围 0-9。

8.5. TriggerVision 触发视觉系统指令

8.5.1. 指令用法

TriggerVision (0) // 清除视觉工艺 0 的视觉数据

.....

8.5.2. 可变元素

TriggerVision [Process]

图 8.5.1

1. [Process]

视觉工艺号，范围 0-9。

8.6. 视觉指令使用案例

8.6.1. 不带跟踪功能示例程序

```
RunVision(0)        //运行视觉 0 号工艺
*1      //跳转标志
TriggerVision(0)    //触发视觉系统
Time T=200 //延时 200ms
Jump *1 If GI110==0                 //假如视觉缓冲区没有缓冲数据，跳转再次触发
GetTrackData(0)  //得到视觉缓冲区的数据，放在 GP52、GP53 中
Add GP53.Z=GP53.Z+5   //GP53 的 Z 方向向上提高 5mm，防止撞到工件
MoveL GP53 VL=200 PL=0.0 T=0        //运动到 GP53
MoveL GP52 VL=200 PL=0.0 T=0        //运动到 GP52
Dout Y(0)=ON        //抓取
Wait X(0)==ON DT=0 CT=0        //等待抓取到位
MoveL GP53 VL=200 PL=0.0 T=0                //运动到 GP53
……
```

注意

1. 视觉缓存个数的 GI 地址可以在视觉工艺的“变量关联”中去自定义了。
2. 视觉数据缓存的地址不只固定的 GP52、53 地址；可以在指令中设置想要缓存的地址。

8.6.2. 带跟踪功能示例程序

```
RunVision(0)        //运行视觉 0 号工艺
*1      //跳转标志
TriggerVision(0)    //触发视觉系统
Time T=200 //延时 200ms
Jump *1 If GI52<=0        //假如视觉缓冲区没有缓冲数据，跳转再次触发
GetVisionData(0)  //如果选择了跟踪，视觉转换的数据就放在跟踪缓冲区，而不是视觉缓冲区，就需要从跟踪缓冲区中得到数据，得到跟踪缓冲区的数据，放在 GP50、GP51 中。使用 0 号跟踪号，必须与视觉工艺里使用的跟踪工艺号相同。
TrackStart(0)       //跟踪开始
Add GP51.Z=GP51.Z+5   //GP51 的 Z 方向向上提高 5mm，防止撞到工件
MoveL GP51 VL=200 PL=0.0 T=0        //运动到 GP51
MoveL GP50 VL=200 PL=0.0 T=0        //运动到 GP50
Dout Y(0)=ON        //抓取
Wait X(0)==ON DT=0 CT=0       //等待抓取到位
MoveL GP51 VL=200 PL=0.0 T=0                //运动到 GP51
TrackEnd(0)         //跟踪结束
.....
```

注意

1. 跟踪数据缓存个数判断的 GI 地址变为视觉工艺中视觉缓存个数的 GI 地址。
2. 跟踪缓存数据 GP 地址可在获取跟踪指令中设置，详情参考跟踪工艺说明书。

9. 码垛指令

9.1. Pallet 码垛指令

9.1.1. 指令用法

Pallet 指令下拉框有 UI、常数、无三种方式可以选择。

图 9.1.1

1. 使用已有码垛工艺

当已创建码垛工艺，则下拉框选择 UI、无。

```
Pallet （1） //运行码垛工艺 1
```

调用设置好的码垛工艺。

2. 直接创建码垛工艺

当选择无时，代表不使用码垛工艺号，设置相关参数直接生成码垛工艺。

```
Pallet （N） //直接生成码垛工艺
```

9.1.2. 可变元素

- **使用码垛工艺号：**

```
Pallet （Num）
```

1. **[Num]**

变量类型：整型变量。

解释：[Num]为码垛工艺文件号，范围为 0 到 199，可使用 UI 做中间变量。

- **不使用码垛工艺号：**

图 9.1.2

Pallet [Process] [Column] [Row] [Height] [Point1] [Point2] [Ang- Point]

1. [Process]

无码垛工艺号。

2. [Column]

设置码盘行数，范围 1-1000。

3. [Row]

设置码盘列数，范围 1-1000。

4. [Height]

码垛点 Z 方向偏移高度。

5. [Point1]

码垛的起始点变量。

6. [Point2]

码垛的最后一个点变量。

7. [Ang-Point]

除起始点与终点外的另一个角点。

注意
这里的起点、终点、角点需要在外面位置型变量中单独记录!

9.2. GetPalletPoint 获取码垛点指令

9.2.1. 指令用法

GetPalletPoint 获取码垛点指令必须和 PALLET 码垛指令配套使用。码垛工艺号后面下拉框有 UI、常数、无可以选择，这里的选择必须同 Pallet 码垛后面选择相同。

图 9.2.1

9.2.2. 可变元素

Pallet [Process] [Variable] [Tag] [Tran-Point] [PalletPoint] [LeavePoint] [Pre-Point] [EndTag]

1. [Process]

码垛工艺号，可选 UI/常数/无。

2. [Variable]

码垛数量寄存器，程序逻辑使用，可以任意定义未使用的变量。

3. [Tag]

增减标识，拆垛或码垛设置，正值为码垛，负值为拆垛；。

4. [Tran-Point]

过渡点变量编号指定，程序中使用，可以任意定义未使用过的变量。

5. [PalletPoint]

码垛点变量编号指定，程序中使用，可以任意定义未使用过的变量。

6. [LeavePoint]

离开点变量编号指定，程序中使用，可以任意定义未使用过的变量。

7. [Pre-Point]

准备点变量编号指定，程序中使用，可以任意定义未使用过的变量。

8. [EndTag]

完成标识定义，当程序执行到最后一垛时，根据指令的设置在码垛完成后，置 ON 或者 OFF，可以用在 PLC 或程序逻辑处理，可以任意定义未使用过的变量。

9.3. 码垛编程示例

9.3.1. 程序示例 1

不带工具，不带用户单线单垛程序示例如下：

```
MoveJ VJ=50 PL=0.0 T=0 //运行到待机点
Set GI1=1    //将垛号设置成 1，意思从第一垛开始码垛
*1     //跳转标志 1
Wait X0==ON DT=0 CT=100      //等待流水线上有料待抓取
MoveJ VJ=50 PL=0.0 T=0       //关节运行到流水线上方点
MoveL VL=500 PL=0.0 T=0      //直线运行到抓取点
DOut Y0=ON       //夹抓闭合夹紧
Time T=500 //延时 0.5 秒
MoveL VL=500 PL=0.0 T=0      //直线运行到抓取点上方点
Pallet (1)     //调用码垛工艺 1
GetPalletPoint ID(1) PalletViable(GI1) Type=1 //获取码垛 1 点位信息
MoveJ GP1=50 PL=0.0 T=0      //关节运行到过渡点
MoveLGP2=500 PL=0.0 T=0      //直线运行到准备点
MoveL GP3=500 PL=0.0 T=0     //直线运行到放料垛点
DOut Y0=OFF      //打开夹抓
Time T=500 //延时 0.5 秒
MoveL GP4=500 PL=0.0 T=0     //直线运行到离开点
Jump *1 If GI1<=20       //如果没有码满 20 垛就跳转到标志 1 继续抓料码垛
MoveJ VJ=50 PL=0.0 T=0       //运行到待机点
```

带工具、带用户码垛程序，只需要把程序 012 行到 017 行带上当前码垛工艺设置时记录过渡点界面显示的工具用户就可以了。

程序已经编好成上面例程序，比如需要带入 2 号工具坐标，3 号用户坐标，只需要把光标移到 012 行点修改指令。

同理把 13 到 17 行全改了就可以了，改后如下：

```
MoveJ GP1=50 PL=0.0 T=2 U=3        //关节运行到过渡点
MoveLGP2=500 PL=0.0 T=2 U=3        //直线运行到准备点
MoveL GP3=500 PL=0.0 T=2 U=3       //直线运行到放料垛点
DOut Y0=OFF        //打开夹抓
Time T=500 //延时  0.5  秒
MoveL GP4=500 PL=0.0 T=2 U=3       //直线运行到离开点
```

9.3.2. 程序示例 2

带工具 2，带用户 3，单线双垛程序，当 X1 为 ON,调用工艺 1，当 X2 为 ON 调用工艺 2。

```
MoveJ VJ=50 PL=0.0 T=2 U=3  //运行到待机点
Set GI1=1    //将码垛 1 垛号设置成 1，意思从第一垛开始码垛
Set GI2=1    //将码垛 2 垛号设置成 1，意思从第一垛开始码垛
*1     //跳转标志 1
MoveJ VJ=50 PL=0.0 T=2 U=3  //关节运行到流水线上方点
Wait X0==ON DT=0 CT=100      //等待流水线上有料待抓取
MoveL VL=500 PL=0.0 T=2 U=3//直线运行到抓取点
DOut Y0=ON         //夹抓闭合夹紧
Time T=500 //延时  0.5  秒
MoveL VL=500 PL=0.0 T=2 U=3//直线运行到抓取点上方点
If X1==ON    //如果 X1 为 ON
Pallet (1)     //调用码垛工艺 1
GetPalletPoint ID(1) PalletViable(GI1) Type=1 //获取码垛 1 点位信息
MoveJ GP1=50 PL=0.0 T=2 U=3        //关节运行到过渡点
MoveLGP2=500 PL=0.0 T=2 U=3        //直线运行到准备点
MoveL GP3=500 PL=0.0 T=2 U=3       //直线运行到放料垛点
DOut Y0=OFF        //打开夹抓
Time T=500 //延时  0.5  秒
```

```
MoveL GP4=500 PL=0.0 T=2 U=3        //直线运行到离开点
EndIf
Jump *1 If GI1<=20          //如果没有码满 20 垛就跳转到标志 1 继续抓料码垛
Jump *2
If X2==ON    //如果 X2 为 ON
Pallet (2)     //调用码垛工艺 2
GetPalletPoint ID(2) PalletViable(GI2) Type=1 //获取码垛 2 点位信息
MoveJ GP1=50 PL=0.0 T=2 U=3          //关节运行到过渡点
MoveLGP2=500 PL=0.0 T=2 U=3          //直线运行到准备点
MoveL GP3=500 PL=0.0 T=2 U=3         //直线运行到放料垛点
DOut Y0=OFF         //打开夹抓
Time T=500 //延时 0.5 秒
MoveL GP4=500 PL=0.0 T=2 U=3         //直线运行到离开点
EndIf
Jump *1 If GI2<=20          //如果没有码满 20 垛就跳转到标志 1 继续抓料码垛
```

10. 通讯指令

10.1. ScoketCreat 指令

10.1.1. 指令用法

ScoketCreat (0) TCP //声明套接字号为 0，使用 TCP 通讯方式

用于针对基于 TCP 或 UDP 通信的连接，创建新的套接字。

10.1.2. 可变元素

ScoketCreat [A] [TCP/UDP]

图 10.1.1

1. [A]

指定需要声明的套接字号，范围：0-49。

2. [TCP/UDP]

需要创建的套接字使用 TCP/UDP 通讯的方式。

10.1.3. 应用举例：

例 1：

SocketCreate(0) TCP

使用 0 号套接字，创建使用 TCP 连接类型的套接字设备。

例 2：

SocketCreate(0) UDP

使用 0 号套接字，创建使用 UDP 连接类型的套接字设备。

10.2. ScoketClose 指令

10.2.1. 指令用法

ScoketClose (0) //关闭通讯

当不再使用套接字连接时，使用 SocketClose 关闭套接字连接。

10.2.2. 可变元素

ScoketClose [A]

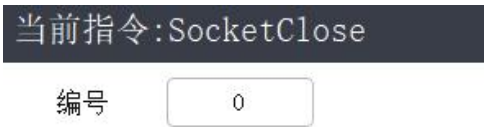


图 10.2.1

1. [A]

指定需要关闭的套接字号，范围：0-49。

10.2.3. 应用举例：

例 1:

SocketClose(0)

关闭 0 号套接字连接，且不能再进行使用。

10.3. SocketConnect 套接字连接指令

10.3.1. 指令用法

SocketConnect 192.16.0.100 1 100 GI=0 //套字连接

用于将套接字与客户端应用中的远程计算机相连。尝试连接指定 IP 地址和端口号的远程服务器，程序执行将进行等待连接，连接成功或失败将返回状态到指定寄存器上。

10.3.2. 可变元素

SocketConnect [A] [IP] [Port] [Time] [Status]

图 10.3.1

1. [A]

指定套接字号，范围：0-49。

2. [IP]

指定需要连接服务器的 IP 地址。

3. [Port]

指定需要连接服务器的端口号，范围：1—65535。

4. [Time]

设定连接的超时时间，范围：0—20000ms。

5. [Status]

状态存放 GI，-1 表示执行失败，0 表示未执行、1 表示执行成功。

10.3.3. 应用举例：

例 1：

SocketConnect(0) 192.168.0.100 8080 5000 LI0

使用 0 号套接字尝试与 192.168.0.100 端口号 8080 的远程服务器连接通讯，连接超时时间 5000ms，连接成功/失败返回状态到 LI0 中。

10.4. SocketSend 套接字发送指令

10.4.1. 指令用法

SocketSend（0） B=0 1 GI=0 //发送字节数据

发送指定的字节数据发给服务器。

10.4.2. 可变元素

SocketSend [A] [Variable] [\length] [Status]

图 10.4.1

1. [A]

指定套接字号，范围：0-49。

2. [Variable]

变量类型选择，“B”为字节变量，“S”为字符串变量，字符串变量不需要设置字节长度。

3. [\length]

设置发送多少字节，范围：1—1024。

4. [Status]

发送状态存放 GI/LI，-1 表示执行失败，0 表示未执行、1 表示执行成功。

10.4.3. 应用举例：

例 1：

SocketSend(0) S0 LI0

使用 0 号套接字，将 S0 字符串变量内容发送给远程计算机，并返回状态到 LI0

例 2：

SocketSend(0) B0 10 LI0

使用 0 号套接字，将 B0 字节变量开始 10 个字节（B0-B9）内容发送给远程计算机，并返回状态到 LI0

10.4.4. 程序运行：

将指定的数据发送到远程计算机，程序执行将进行发送，发送成功或失败将返回状态到指定寄存器上。

10.5. SocketRecv 套接字接收指令

10.5.1. 指令用法

SocketRecv (0) B=0 1 0 GI=0 //接收字节数据

将接收到服务器的数据存放至字节中。

10.5.2. 可变元素

SocketRecv [A] [Variable] [\length] [Status]

图 10.5.1

1. [A]

指定需要关闭的套接字号，范围：0-49。

2. [Variable]

变量类型选择，“B”为字节变量，“S”为字符串变量。

3. [\length]

设置发送多少字节，范围：1—1024。

4. [Time]

超时时间，指等待接收的时间，范围：0-20000ms。

5. [Status]

发送状态存放 GI/LI，-1 表示执行失败，0 表示未执行、1 表示执行成功。

注意
当接收时按照指令设置长度接收，当满足接收长度时指令执行结束；当未满足接收长度按等待超时时间满足设置时指令执行结束。

10.5.3. 应用举例：

例 1：

SocketRecv(0) S0 50 500 LI0

从 0 号套接字连接的远程计算机接收 50 个字节数据，超时时间 500ms，并将实际接收数据大小存放至字符串变量 S0，指令执行结束对应返回接收失败或成功至 LI0。

例 2：

SocketRecv(0) B0 10 500 LI0

从 0 号套接字连接远程计算机接收 10 个字节数据，超时时间 500ms，并将实际接收数据大小存放至字符串变量 B0-B9 中，指令执行结束对应返回接收失败或成功至 LI0。注意：这里 B 代表字节，接收的数据长度是以当前指定的 B0 作为起始地址依次往后存放

10.5.4. 程序运行：

将等待执行该指令，直至数据满足接收字节长度或满足超时时间时向下执行。

10.6. Socket 套接字编程示例

Socket 套接字编程流程：

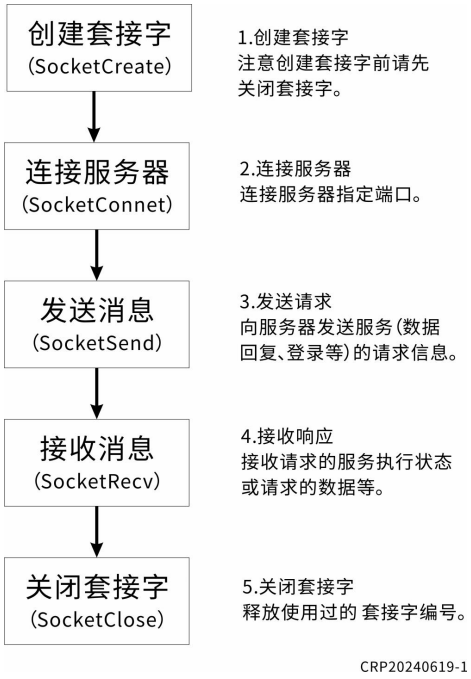


图 10.6.1

编程示例：

```
SocketClose (0) //关闭 Socket 通讯连接
SocketCreate (0)  TCP //创建套接字 0 号连接，TCP 通讯
SocketConnect (0) 192.168.10.150 2756 500      LI=0 //Socket TCP 连接
SocketSend (0)    S=0 LI=1 //发送字符串 S0 变量，“A”触发相机拍照
SocketRecv (0) S=1 2 10000    LI=2 //等待相机返回“B”，拍照成功标识符
If S1==OK //相机返回拍照成功标识符
 SocketRecv (0) S=1      200 10000 LI=3 //等待相机返回坐标
 Set S2=Split(S1 ',') //将接收的字符串按“,”拆分开
 Set S7=Split(S7 ';') //将结束符“;”拆分开
 Set GR0=Convert(Float S2) //将字符串转换成浮点 X
 Set GR1=Convert(Float S3) //将字符串转换成浮点 Y
 Set GR2=Convert(Float S4) //将字符串转换成浮点 Z
 Set GR3=Convert(Float S5) //将字符串转换成浮点 Rx
 Set GR4=Convert(Float S6) //将字符串转换成浮点 Ry
```

```
 Set GR5=Convert(Float S7) //将字符串转换成浮点 Rz
 Set GP0.X=GR0 //将 X 数据赋值给 GP0.X
 Set GP0.Y=GR1 //将 Y 数据赋值给 GP0.Y
 Set GP0.Z=GR2 //将 Z 数据赋值给 GP0.Z
 Set GP0.Rx=GR3 //将 Rx 数据赋值给 GP0.Rx
 Set GP0.Ry=GR4 //将 Ry 数据赋值给 GP0.Ry
 Set GP0.Rz=GR5 //将 Rz 数据赋值给 GP0.Rz
EndIf
Set GP1=GRobT (R1 0 0) //使用 CRobT 函数按工具 0 用户 0 读取空间位置
S8=GR100+" "+GR101+" " +GR102+" "+GR103+" "+GR104+" "+GR105 //使用 Set 组合指令将当前空间位置 X、
Y、Z...转换到字符串 S8
SocketSend (0) S=8       LI=8 //使用 Socket 发送将当前位置发送给服务器
```

编程步骤：

1. 每次执行程序先关闭 socket 连接；

2. 再创建一个 0 号 TCP 类型的 socket；

3. 再使用刚刚创建的 0 号 socket，使用“192.168.10.150”连接对方的服务器 IP，端口号为 2756，连接等待时间 500ms，当连接超过 500ms 则连接失败，执行下一条程序并返回失败结果-1 到状态 LI0；（这里可以根据返回状态进行判断是否重新进行连接或者报警）

4. Socket 发送字符串 S0 变量，同样返回发送成功/失败的结果到状态 LI1；

5. Socket 接收字符串 S1 变量，指定接收字节长度 2，返回接收成功/失败的结果到 LI2 变量中；

6. 对 S1 字符串内容判断是否拍照成功，拍照成功则进行后续点位拆分（例如相机返回 OK/NG 的标识符）；

7. socket 接收数据到 S1，字节长度 200，接收时间 10000ms，返回状态到 LI3(可根据返回状态进行成功失败的判断)；

8. 第 8-9 行主要针对接收的字符串数据进行拆分处理，如上述“Set 赋值指令（Split 函数）”说明；

9. 第 10-15 行主要将使用 Split 分割的子字符串分别转换到浮点数里；

10. 第 16-21 行主要讲浮点的数据向 GP 点 X Y Z Rx Ry Rz 值赋值；

11. 第 23 行使用 CRobT 函数按工具 0 用户 0 读取空间位置；

12. 第 24 行使用 Set 组合指令将当前空间位置 X、Y、Z...转换到字符串 S8；

13. 第 25 行使用 Socket 发送将当前位置发送给服务器。

、

成都卡诺普机器人技术股份有限公司

CHENGDU CRP ROBOT TECHNOLOGY CO.,LTD

400-668-8633

crobotp@crprobot.com

www.crprobot.com

四川成都市成华区华月路 188 号