XR 机器人通用系统手册

《基本指令篇》

2024 年 12 月 12 日

目录

概述

XR 机器人通用系统基本指令篇旨在描述机器人系统的语法结构及使用方法，为用户提供便捷的指令编辑和现场部署方法。通过仔细阅读本手册，可以基本掌握 XR 机器人系统的基本变量结构、语法特点、语法格式以及简单的编程方式。本手册将分十二节分别讲述各类指令的名称、参数、使用方法、以及示例，示例仅供编程参考，不作为现场实施依据。

手册约定

为方便阅读和理解，本手册作出如下约定：

（1）指令中的可选参数以斜体字表示。

（2）可选参数后面有*表示其后是条件表达式。

（3）可选参数后面有#表示其后是变量或常量。

（4）所有指令中的数值输入均包含数值常量、数值变量、数组元素。

（5）文中涉及到的所有速度百分比(%),输入范围均限定为 0~100。

（6）文中未提及时间输入范围的，其设置范围均大于等于 0 且小于 2147483648。

（7）手册未提及的指令编辑内容，请参考《XR 系统用户操作手册》。

1 变量

XR 系统以变量的形式存储指令编程中所需的各类参数，当该变量在系统中建立后，可以在指令添加或修改时，直接选择该变量。

XR 系统的变量信息如下表 1 所示：

表 1 XR 系统变量

数据名称	字段	类型
整型（Int）	初始值/变量值	int
单精度浮点（Float）	初始值/变量值	float
布尔（Bool）	初始值/变量值	bool
速度变量（Speed）	TCP 线速度(mm/s)值	float
	关节速度百分比(%)	float
	加速度百分比(%)	float
	减速度百分比(%)	float
	外部轴速度百分比(%)	float
	外部轴加速度百分比(%)	float
	外部轴减速度百分比(%)	float
过渡变量（Zone）	CP 过渡参数百分比(%)	short
	路径过渡误差(mm)	float
	外部轴过渡百分比(%)	unsigned short
目标位置（Points）	X/J1	float
	Y/J2	float
	Z/J3	float
	RZ/J4	float
	RY/J5	float
	RX/J6	float
	Cf1	short
	Cf4	short
	Cf6	short
	手系	short
	坐标系	short
	工件	short
	工具	short
	外部轴 1(°)	float
	外部轴 2(°)	float
	外部轴 3(°)	float
	外部轴 4(°)	float
	外部轴 5(°)	float
数字输出（DO）	初始值/变量值	bool
数字输入（DI）	初始值/变量值	bool
门型运动参数（Arch）	水平运动 Z(mm)	float
	上升距离(mm)	float
	下降距离(mm)	float
焊缝参数（Seamdata）	预送气时间(s)	float
	尾送气时间(s)	float
	起弧电流(A)	float
	起弧电压(V)	float

	起弧时间(s)	float
	收弧电流(A)	float
	收弧电压(V)	float
	收弧时间(s)	float
	刮擦起弧次数	unsigned int
	后处理时间(s)	float
焊接参数（Welddata）	焊接速度(mm/s)	float
	焊接电流(A)	float
	焊接电压(V)	float
	焊接模式	unsigned short
摆弧参数（Weavedata）	摆弧类型	unsigned short
	摆弧姿态是否跟随轨迹姿态	unsigned short
	摆弧长度(mm)	float
	摆弧幅度(mm)	float
	摆弧高度(mm)	float
	左停留长度(mm)	float
	中停留长度(mm)	float
	右停留长度(mm)	float
	摆弧角度(°)	unsigned short
字符串（String）	字符串[64 字节]	char[]
过程输出（ProcessOut）	触发类型	unsigned short
	等待时间(ms)	unsigned short
	输出值（mm/%/ms）	float
	输出序号	unsigned int
	输出信号电平，0 高电平，1 低电平,2 输出取反信号	unsigned short
	输出信号选择，0：M，1：Y	unsigned short
组输出（SignalGi）	Y_Bit0~Y_Bit7	unsigned short
组输入（SignalGo）	X_Bit0~X_Bit7	unsigned short
工具（Tool）	系统内部变量，用户不可编辑；在工具标定界面标定后，可在指令中直接使用	/
薄膜按键（TPDI）	暂未开发	/
工件（User）	系统内部变量，用户不可编辑；在工具标定界面标定后，可在指令中直接使用	/

注：本手册后续提及到的所有变量均在上表中，除此之外的其他变量均属于非法变量。

1.1 系统常量

系统中集成了部分常用的变量，本手册称之为系统常量。若用户需要使用系统提供的常量之外的其他变量，可以通过自定义的方式实现。

表 1.1.1 过渡常量

类型	名称	过渡参数百分比(%)	路径过渡误差(mm)	外部轴过渡百分比(%)
Zone	Zone0	0	0	0
	Zone10	10	0	10
	Zone30	30	0	30
	Zone50	50	0	50
	Zone100	100	0	100

✧ 表 1.1.1 中各值的取值范围是大于等于 0 小于等于 100。

表 1.1.2 速度常量

类型	名称	TCPSpeed	JSpeed	Acc	Dec	exSpeed	exAcc	exDec
Speed	V5	5	1	30	30	10	30	30
	V10	10	2	30	30	30	30	30
	V30	30	3	30	30	30	30	30
	V50	50	5	30	30	30	30	30
	V100	100	10	30	30	30	30	30
	V200	200	20	30	30	30	30	30
	V500	500	50	50	50	50	50	50
	V800	800	80	50	50	50	50	50
	V1000	1000	100	100	100	100	100	100
	V1500	1500	100	100	100	100	100	100
	V2000	2000	100	100	100	100	100	100
	V3000	3000	100	100	100	100	100	100
	V4000	4000	100	100	100	100	100	100

✧ 上表中的单词含义如下：

TCPSpeed：TCP 线速度（mm/s），该值应小于基准值（默认 2000mm/s）。

JSpeed：关节速度百分比（%），范围为大于等于 0 小于等于 100。

Acc：加速度百分比（%），范围为大于等于 0 小于等于 100。

Dec：减速度百分比（%），范围为大于等于 0 小于等于 100。

exSpeed：外部轴速度百分比（%），范围为大于等于 0 小于等于 100。

exAcc：外部轴加速度百分比（%），范围为大于等于 0 小于等于 100。

exDec：外部轴减速度百分比（%），范围为大于等于 0 小于等于 100。

表 1.1.3 门型运动参数（Arch）常量

类型	名称	水平运动 Z（mm）	上升距离(mm)	下降距离(mm)
Arch	Arch0	0	0	0
	Arch1	0	10	10
	Arch2	0	20	20
	Arch3	0	30	30
	Arch4	0	40	40
	Arch5	0	50	50

表 1.1.4 Bool 系统常量

类型	名称	值
Bool	true	1
	false	0

1.2 系统变量

系统变量在系统启动时自动建立，无需用户手动建立。且系统变量的值由系统根据运行情况自动赋值，用户无需操作。系统变量的种类及详细描述见下表：

表 1.2.1 系统变量

类型	名称	备注
Tool	Tool0~Tool19	20 个工具，可标定后直接使用，无需手动建立
User	User0~User17	18 个工件，可标定后直接使用，无需手动建立
寄存器 HD	HD1000~HD2999	2000 个数据寄存器，可直接使用，无需手动建立
线圈 M	M1000~M2999	2000 个内部线圈，可直接使用，无需手动建立
Bool	PosFound	指令 Find 条件满足时，该变量自动置为 true
Int	ErrNo	发生用户错误时，该变量自动置为当前错误代码，仅支持可处理的错误代码
Points	FindPos	指令 Find 条件满足时，该变量自动记录为当前基座标系下的坐标值

	Here	系统关键字，表示当前位置
DI	X0~X17 X1_0~X1_17 ……	根据扩展 IO 模块的类型来确定名称和输入端口的数量
DO	Y0~Y17 Y1_0~Y1_17 ……	根据扩展 IO 模块的类型来确定名称和输入端口的数量

1.3 自定义变量

除 1.2 节描述的系统变量之外，用户可根据编程需要自行定义所需要的变量，新建变量可在主页->程序数据->所有->选择所需的变量类型->右上角新建变量。自定义变量命名应避免与系统变量重复，参考表 1.2.1 和附录中的附表 2 避免重复命名。

1.4 特殊变量

特殊变量的建立及编辑方式与普通变量相同，但没有为其分配指定的操作指令，一般与现有的其他指令（如 If、While）等指令搭配使用。

1.4.1 组输入变量 SignalGi

组输入变量在程序数据功能页面中新建或编辑，可分别指定 Bit0~Bit7 对应的 Y 线圈序列。其中 Bit0 代表二进制数字的低位，Bit7 代表二进制数字的高位。如下表所示，若将 X 线圈序列指定为 1-X10(在"1-X10"中，可以理解为扩展模块编号为 1，X 表示是输入端口，端口号为 10)~1-X17 的连续序列，其中 1-X12 和 1-X11 为 true，其他均为 false，则对应的十进制数为 6。该变量可在条件表达式中使用，相关使用方法可参考 If 等指令。

表 1.4.1.1 SignalGi 组输入线圈与二进制和十进制的关系

端口	1-X17	1-X16	1-X15	1-X14	1-X13	1-X12	1-X11	1-X10
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

二进制	0	0	0	0	0	1	1	0
布尔值	false	false	false	false	false	true	true	false
十进制	6							

1.4.2 组输出变量 SignalGo

组输出变量在程序数据功能页面中新建或编辑，可分别指定 Bit0~Bit7 对应的 Y 线圈序列。其中 Bit0 代表二进制数字的低位，Bit7 代表二进制数字的高位。如下表所示，若将 Y 线圈序列指定为 1-Y10(在"1-Y10"中，可以理解为扩展模块编号为 1，Y 表示是输出端口,端口号为 10)~1-Y17 的连续序列，其中 1-Y12 和 1-Y11 为 true，其他均为 false,则对应的十进制数为 6。该变量可在条件表达式中使用，相关使用方法可参考 If 等指令。

表 1.4.2.1 SignalGo 组输出线圈与二进制和十进制的关系

端口	1-Y17	1-Y16	1-Y15	1-Y14	1-Y13	1-Y12	1-Y11	1-Y10
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
二进制	0	0	0	0	0	1	1	0
布尔值	false	false	false	false	false	true	true	false
十进制	6							

2 运动指令

运动指令包括关节运动 MoveJ、直线运动 MoveL、门形运动 Jump/JumpJ/JumpL、圆弧运动 MoveC 以及与运动指令配合的点位偏移功能、过程输入输出、直到(Till)、检测(Sense)、寻位(Find)等。

2.1 可选运动参数

可选运动参数包括：直到(Till)、检测(Sense)、寻位(Find)、奇异规避(Singular)、过程输出(ProcessOut)。在指令编辑时，若勾选该功能，则对应的关键字及表达式添加至指令中，该功能生效；若未勾选，则忽略该功能。

2.1.1 直到 Till

轨迹运行时根据条件表达式是否满足（结果为 true）决定当前轨迹是否进行过渡处理和下一段轨迹运动，当 Till 条件满足时，当前轨迹执行缓停动作，运动停止后，继续执行下一条指令。

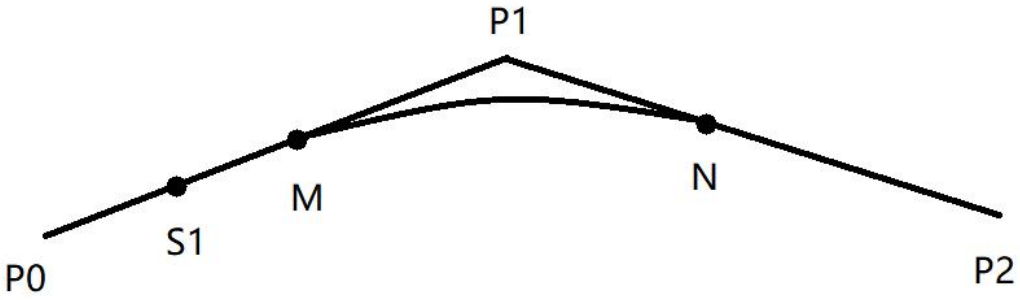
- Till 功能支持轨迹类型为全轨迹类型，Till 功能的停止点在轨迹过渡处理后的轨迹上，本质上执行了一次缓停，该轨迹停止后不可恢复。Till 触发后的停止运动中执行 RBSTOP(0.0)无效。执行 RBSTOP(-1.0)生效，急停状态优先级最高。
- 单段轨迹执行 Till 功能，停止点在当前轨迹点与轨迹终点之间的任意点，点位值由缓停时间设置值决定，多段轨迹执行 Till 功能，停止点可能会进入到下一段轨迹。

2.1.2 检测 Sense

连续轨迹运行时根据条件表达式是否满足（结果为 true）决定当前轨迹是否进行过渡处理和下一段轨迹运动，当 Sense 后的条件表达时满足时，当前轨迹

与下一段轨迹的过渡失效，运动到当前轨迹的目标点后停止运动，此时程序指令将发生阻塞，直到条件不满足时恢复；当条件表达时不满足（结果为 false）时，将继续执行后续的运动指令。

- Sense 功能支持轨迹类型为 LINE、PTP 以及 JUMP 指令，Sense 功能的停止点为用户输入加工点，该轨迹停止后不可恢复。Sense 触发后的停止运动中执行 RBSTOP(0.0)无效，执行 RBSTOP(-1.0)生效，急停状态优先级最高。当轨迹为 LINE 时，触发 Sense 功能，本质上执行了一次定长停止，严格要求停止到目标点时可能出现超加速度现象，默认为可执行不报警。
- Sense 的判断发生在轨迹过渡前，当前轨迹属于前瞻结束段或非前瞻段时，只需要完成当前段轨迹插补即可。如果当前运动轨迹为连续轨迹的最后一段，执行 Sense 无效。如果触发 Sense 时当前段轨迹剩余长度不够减速，则按照 Till 处理 Sense，轨迹执行缓停。当轨迹间存在过渡时，判断条件如下：（1）样条过渡情况，运动到‘路径过渡后剩余轨迹的一半’为生效点，取最晚触发的点作为 Sense 信号的开始解析点；（2）CP 过渡情况，生效点的选取与样条过渡情况相同。



例如，有直线 P0P1 和直线 P1P2，进行过渡处理后，轨迹由 P0M、MN、NP2 组成，S1 点位生效点；当轨迹插补至 S1M 线段之间任意时刻触发 Sense 信号，则开始响应，在 P1 点停止后不在运行后续轨迹。

2.1.3 寻位 Find

- 轨迹运行时根据条件表达式是否满足（结果为 true）决定当前轨迹是否锁存当前实时反馈位置，其中当条件表达式满足时，记录当前信号触发时刻的实时反馈位置，反之当条件表达式不满足时（结果为 false），不做任何处理。

- Find 功能支持轨迹类型为全轨迹类型，总线周期任务记录该点位的所有点位信息以及触发信号时记录的反馈位姿或关节角度（取决于该点位设置的坐标系类型，配置为关节坐标系时记录为反馈关节角度，其余坐标系类型反馈末端法兰对于基座标的位姿）、手系、cf 配置、外部轴反馈角度。

2.1.4 奇异规避 Singular

奇异规避功能可以最大限度的在指令运行中降低在奇异区附近发生错误的概率，且可以根据运行情况自动调整轨迹来规避奇异区。奇异规避的参数如下：

【规避类型】：共 3 三种规避方式，设置范围是 0 至 3。

✧ 各种规避类型如下：

- 0：奇异规避关闭。
- 1：奇异规避开启，取始末点 4/5/6 关节角度变化范围小的进行奇异规避。
- 2：奇异规避开启，不变腕部手系进行奇异规避。
- 3：奇异规避开启，变腕部手系进行奇异规避。

2.1.5 过程输出 ProcessOut

运动过程中打开信号输出功能，可在机器人运动指令执行过程中，根据运动完成程度或设定时间输出信号。过程输出参数以变量形式存储，通过程序数据页面可以进行新建和编辑，可设置的参数如下所示：

【触发类型】：过程输出判断条件类型，1：起点距离长度触发，2：起点距离百分比触发，3：终点距离长度触发，4：终点距离百分比触发，5：时间提前触发，6：时间延迟触发。

【等待时间】：延时时间，单位 ms，仅当过程输出判断条件类型为 1~4 时有效，指令延时触发信号输出；注：当延时时间超过当前轨迹段时间且当前为最后一段轨迹则不输出相应信号。

【输出值】：过程输出触发条件值，路径长度值(mm)/路径百分比(%)/时间(ms)。

【输出序号】：过程输出 M 或 Y 信号序号，例如赋值 1000，即为 M[1000]；Y 信

号赋值为十进制数值，对应在 PLC 中为八进制数，例如置位 Y[12]，即赋值 10。

【信号】：输出信号高低电平有效，0：触发输出高电平，1：触发输出低电平，2：输出取反信号。

【输出信号选择】：过程输出信号类型，0：M，1：Y。

✧ 上述各参数的具体含义如下：

- 起点距离长度触发：从起点开始，当前段轨迹运动距离大于设定长度后指令触发；如果设定长度值超过轨迹总长度，信号响应。PTP 轨迹以及 JUMP1、JUMP3 不适用此模式。
- 起点距离百分比触发：从起点开始，当前段轨迹运动距离大于设定总长度的百分比后指令触发；百分比取值范围[0,100]。JUMP1、JUMP3 不适用此模式。
- 终点距离长度触发：从终点开始往前计算，当前段轨迹运动剩余距离小于设定长度后指令触发；如果设定长度值超过轨迹总长度，进入当前轨迹信号即可响应。PTP 轨迹以及 JUMP1、JUMP3 不适用此模式。
- 终点距离百分比触发：从终点开始往前计算，当前段轨迹运动剩余距离小于设定总长度的百分比后指令触发；百分比取值范围[0,100]。JUMP1、JUMP3 不适用此模式。
- 时间提前触发：从终点开始往前计算，当前段轨迹运动剩余时间小于设定时间后指令触发；如果设定时间超过轨迹总时长，进入当前轨迹信号即可响应。
- 时间延迟触发：从终点开始往后计算，当前段轨迹切换为下一段轨迹后，下一段轨迹的运动时间大于设定时间后指令触发；如果设定时间超过轨迹总时长，信号不响应。

距离长度触发和距离百分比触发可以设定延时时间再输出响应信号；有过渡段存在时，当前段轨迹的路径长度或总运行时间计算起点为上一个过渡结束点、计算终点为下一个过渡结束点；一条运动指令最多支持 2 个 ProcessOut 变量。

2.2 关节运动 MoveJ

➢ 指令说明

以点到点的动作方式从当前位置移动到目标点位。在关节坐标系下示教点位时，该指令执行点位运动时以关节角度运动方式，并且记录的点位坐标为关节角度；而在基座标系下示教点位时，该指令执行点位运动时以笛卡尔坐标下的关节运动方式到达目标点位，并且记录的坐标为笛卡尔坐标。

➢ 语法格式

MoveJ [目标点],[速度],[过渡],[工具],[*工件*],[*Till* *],[*Sense* *],

[*Find* *],[*Singular* *],[*ProcessOut* #];

➢ 参数

【目标点】：要到达的目标点变量，可以是数组元素或单个点位变量，也可对点位进行关节偏移或笛卡尔坐标偏移。偏移功能的关键字为 OffsetJ 和 OffsetC.该功能可以在指令编辑时，修改点位的条件表达式，在功能页面开启。

【速度】：到达目标点过程的速度设定值，单位使用速度变量中的速度%。

【过渡】：设置 CP 过渡的大小。

【工具】：设置运动到目标点所用的工具号。

✧ 可选参数

【*工件*】：指定运动所需的工件坐标系，使用前需先标定工件坐标系。

【*Till* *】：添加该参数时，以 Till 关键字标识，后加条件表达式（如：Till X0 == true）；其后的条件表达式满足时，将触发停止运动，继续运动下一行指令。Till 指令在单个线程中不能超过 50 条。

【*Sense* *】：添加该参数时，以 Sense 关键字标识，后加条件表达式（如：

Sense X0 == true）；到达目标点是否过渡，其后的条件表达式满足时该信号将被触发，此时不进行过渡，并停止运动在目标点上方。

【*Find #*】：添加该参数时，以 Find 关键字标识，后加条件表达式（如：Find X0 == true）；其后的条件表达式满足时，将读取当前的坐标写入到 FindPos 变量内。Find 功能支持轨迹类型为全轨迹类型，总线周期任务记录该点位的所有点位信息以及触发信号时记录的反馈位姿或关节角度(取决于该点位设置的坐标系类型，配置为关节坐标系时记录为反馈关节角度，其余坐标系类型反馈末端法兰对于基座标的位姿）、手系、cf 配置、外部轴反馈角度。

【*Singular #*】：添加该参数时，以 Singular 关键字标识，后加整型常量或整型变量（如 Singular 1）；指令中存在 Singular 关键字时，将根据其后设置的规避类型执行运动。

【*ProcessOut #*】：添加该参数时，以 ProcessOut 关键字标识，后加整型常量或整型变量（如 ProcessOut process1,process2）;ProcessOut 关键字之后可跟两个 ProcessOut 类型的变量，以逗号分隔变量，此时可以在运动过程中同时输出两个 ProcessOut 变量中设置的信号。

【*OffsetJ*】：对当前点位进行关节偏移，偏移不受点位类型的影响，格式为如下：OffsetJ(P0,Δj1,Δj2,Δj3,Δj4,Δj5,Δj6)

其中Δj1, Δj2, Δj3, Δj4, Δj5, Δj6 分别表示对应关节的偏移量，单位为（°）；该偏移功能不会修改要偏移的点位（这里的 P0），只是在运动时计算出的临时偏移点位。

【*OffsetC*】：对当前点位进行笛卡尔偏移，偏移不受点位类型的影响，格式如下：OffsetC(P0,Δx,Δy,Δz,Δrz,Δry,Δrx)

其中Δx, Δy, Δz, Δrz, Δry, Δrx 分别表示对应笛卡尔坐标的偏移量，单位为（mm/s|°/s）；该偏移功能不会修改要偏移的点位（这里的 P0），只是在运动时计算出的临时偏移点位。

➢ 示例

```
MoveJ P0,V80,Zone0,Tool0;
```

//关节运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0。

MoveJ P0,V80,Zone0,Tool0,Till X0 == true;

//关节运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0，开启 Till 功能，条件表达式满足时触发停止

MoveJ P0,V80,Zone0,Tool0,Find X0 == true;

//关节运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0，开启 Find 功能，条件表达式满足时触发寻位功能

MoveJ P0,V80,Zone0,Tool0,Sense X0 == true;

//关节运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0，开启 Sense 功能，条件表达式满足时触发 Sense 功能

MoveJ P0,V80,Zone0,Tool0,Singular 2;

//关节运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0，奇异规避类型设置为 2

MoveJ P0,V80,Zone0,Tool0,ProcessOut process1,process2;

//关节运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0，设置过程输出参数为 process1 和 process2。

MoveJ OffsetC(P0,10,10,10,0,0,0),V80,Zone0,Tool0;

//假设 P0 为基座标系下示教的点位，坐标为(300,350,290,50,60,70)，执行 OffsetC 偏移后的坐标为：(310,360,300,50,60,70)。机器人末端将运动到该位置，但原有的 P0 坐标值不变。

MoveJ OffsetJ(P0,10,10,10,0,0,0),V80,Zone0,Tool0;

//假设 P0 为关节坐标系下示教的点位，坐标为(0,-90,0,0,0,0)，执行 OffsetJ 偏移后的坐标为：(10,-80,10,0,0,0)。机器人末端将运动到该位置，但原有的 P0 坐标值不变。

MoveJ OffsetJ(P1,10,10,10,0,0,0),V80,Zone0,Tool0;

//假设 P1 为基坐标系下示教的点位，坐标为(300,350,290,50,60,70)，对应

的关节坐标为(30,40,50,10,20,25)(以实际为准，这里仅供说明),执行 OffsetJ 偏移后的基坐标为：(305,330,270,50,60,70) (以实际为准，这里仅供说明)，对应的关节坐标为（40，50，60，10，20，25）。机器人末端将运动到该位置，但原有的 P1 坐标值不变。

2.3 直线运动 MoveL

➢ 指令说明

机械手末端以直线的运动轨迹从当前位置运动到当前启用的坐标系下的目标绝对位置。记录该指令匹配的点位时，存储格式机器人所处坐标系下的点位。

➢ 语法格式

MoveL [目标点],[速度],[过渡],[工具],[*工件*],[*Till* *],[*Sense* *],

[*Find* *],[*Singular* *],[*ProcessOut* #];

➢ 参数

【目标点】：要到达的目标点变量，可以是数组元素或单个点位变量，也可对点位进行关节偏移或笛卡尔坐标偏移。偏移功能的关键字为 OffsetJ 和 OffsetC.该功能可以在指令编辑时，修改点位的条件表达式，在功能页面开启。

【速度】：到达目标点过程的速度设定值，单位使用速度变量中的 TCP 线速度（mm/s）。

【过渡】：设置 CP 过渡的大小。

【工具】：设置运动到目标点所用的工具号。

✧ 可选参数

【*工件*】：指定运动所需的工件坐标系，使用前需先标定工件坐标系。

【*Till* *】：添加该参数时，以 Till 关键字标识，后加条件表达式（如：

Till X0 == true）；其后的条件表达式满足时，将触发停止运动，继续运动下一行指令。

【*Sense **】：添加该参数时，以 Sense 关键字标识，后加条件表达式（如：Sense X0 == true）；到达目标点是否过渡，其后的条件表达式满足时该信号将被触发，此时不进行过渡，并停止运动在目标点上方。

【*Find **】：添加该参数时，以 Find 关键字标识，后加条件表达式（如：Find X0 == true）；其后的条件表达式满足时，将读取当前的坐标写入到 FindPos 变量内。Find 功能支持轨迹类型为全轨迹类型，总线周期任务记录该点位的所有点位信息以及触发信号时记录的反馈位姿或关节角度(取决于该点位设置的坐标系类型，配置为关节坐标系时记录为反馈关节角度，其余坐标系类型反馈末端法兰对于基座标的位姿）、手系、cf 配置、外部轴反馈角度。

【*Singular **】：添加该参数时，以 Singular 关键字标识，后加整型常量或整型变量（如 Singular 1）；指令中存在 Singular 关键字时，将根据其后设置的规避类型执行运动。

【*ProcessOut #*】：添加该参数时，以 ProcessOut 关键字标识，后加整型常量或整型变量（如 ProcessOut process1,process2）;ProcessOut 关键字之后可跟两个 ProcessOut 类型的变量，以逗号分隔变量，此时可以在运动过程中同时输出两个 ProcessOut 变量中设置的信号。

【*OffsetJ*】：对当前点位进行关节偏移，偏移不受点位类型的影响，格式为如下：OffsetJ(P0,Δj1,Δj2,Δj3,Δj4,Δj5,Δj6)

其中Δj1, Δj2, Δj3, Δj4, Δj5, Δj6 分别表示对应关节的偏移量，单位为（°）；该偏移功能不会修改要偏移的点位（这里的 P0），只是在运动时计算出的临时偏移点位。

【*OffsetC*】：对当前点位进行笛卡尔偏移，偏移不受点位类型的影响，格式为如下：OffsetC(P0,Δx,Δy,Δz,Δrz,Δry,Δrx)

其中Δx, Δy, Δz, Δrz, Δry, Δrx 分别表示对应笛卡尔坐标的偏移量，单位为（mm/s|°/s）；该偏移功能不会修改要偏移的点位（这里的 P0），只是在运动时计算出的临时偏移点位。

➢ 示例

MoveL P0,V80,Zone0,Tool0;

//直线运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0。

MoveL P0,V80,Zone0,Tool0,Till X0 == true;

//直线运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0，开启 Till 功能，条件表达式满足时触发停止

MoveL P0,V80,Zone0,Tool0,Find X0 == true;

//直线运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0，开启 Find 功能，条件表达式满足时触发寻位功能

MoveL P0,V80,Zone0,Tool0,Sense X0 == true;

//直线运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0，开启 Sense 功能，条件表达式满足时触发 Sense 功能

MoveL P0,V80,Zone0,Tool0,Singular 2;

//直线运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0，奇异规避类型设置为 2

MoveL P0,V80,Zone0,Tool0,ProcessOut process1,process2;

//直线运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0，设置过程输出参数为 process1 和 process2。

MoveL OffsetC(P0,10,10,10,0,0,0),V80,Zone0,Tool0;

//假设 P0 为基座标系下示教的点位，坐标为(300,350,290,50,60,70)，执行 OffsetC 偏移后的坐标为：(310,360,300,50,60,70)。机器人末端将运动到该位置，但原有的 P0 坐标值不变。

MoveL OffsetJ(P0,10,10,10,0,0,0),V80,Zone0,Tool0;

//假设 P0 为关节坐标系下示教的点位，坐标为(0,-90,0,0,0,0)，执行 OffsetJ 偏移后的坐标为：(10,-80,10,0,0,0)。机器人末端将运动到该位置，

但原有的 P0 坐标值不变。

MoveL OffsetJ(P1,10,10,10,0,0,0),V80,Zone0,Tool0;

//假设 P1 为基坐标系下示教的点位，坐标为(300,350,290,50,60,70)，对应的关节坐标为(30,40,50,10,20,25)(以实际为准，这里仅供说明)，执行 OffsetJ 偏移后的基坐标为：(305,330,270,50,60,70) (以实际为准，这里仅供说明)，对应的关节坐标为（40，50，60，10，20，25）。机器人末端将运动到该位置，但原有的 P1 坐标值不变。

2.4 圆弧运动 MoveC

➢ 指令说明

机械手末端以圆弧运动轨迹从当前位置运动到目标位置，圆弧轨迹由运动开始位置点、中间点（过渡点）及目标点三点构成。轨迹规划路径如下图所示：

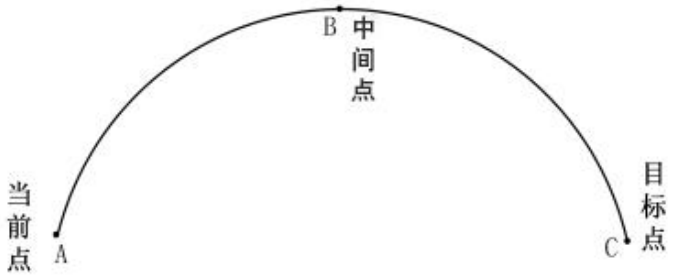


图 2.4.1 圆弧轨迹规划

➢ 语法格式

MoveC [中间点],[目标点],[速度],[过渡],[工具],[*工件*],[*Till* *],

[*Find* *],[*Singular* *],[*ProcessOut* #];

➢ 参数

【中间点】:用于确定圆弧轨迹的必经点,可以是数组元素或单个点位变量,也可对点位进行关节偏移或笛卡尔坐标偏移。偏移功能的关键字为 OffsetJ 和 OffsetC.该功能可以在指令编辑时,修改点位的条件表达式,在功能页面开启。

【目标点】:要到达的目标点变量,可以是数组元素或单个点位变量,也可对点位进行关节偏移或笛卡尔坐标偏移。

【速度】:到达目标点过程的速度设定值,单位使用速度变量中的 TCP 线速度(mm/s)。

【过渡】:设置 CP 过渡的大小。

【工具】:设置运动到目标点所用的工具号。

✧ 可选参数

【*工件*】:指定运动所需的工件坐标系,使用前需先标定工件坐标系。

【*Till **】:添加该参数时,以 Till 关键字标识,后加条件表达式(如:Till X0 == true);其后的条件表达式满足时,将触发停止运动,继续运动下一行指令。

【*Find **】:添加该参数时,以 Find 关键字标识,后加条件表达式(如:Find X0 == true);其后的条件表达式满足时,将读取当前的坐标写入到 FindPos 变量内。Find 功能支持轨迹类型为全轨迹类型,总线周期任务记录该点位的所有点位信息以及触发信号时记录的反馈位姿或关节角度(取决于该点位设置的坐标系类型,配置为关节坐标系时记录为反馈关节角度,其余坐标系类型反馈末端法兰对于基座标的位姿)、手系、cf 配置、外部轴反馈角度。

【*Singular **】:添加该参数时,以 Singular 关键字标识,后加整型常量或整型变量(如 Singular 1);指令中存在 Singular 关键字时,将根据其后设置的规避类型执行运动。

【*ProcessOut #*】:添加该参数时,以 ProcessOut 关键字标识,后加整型常量或整型变量(如 ProcessOut process1,process2);ProcessOut 关键字之后可跟两个 ProcessOut 类型的变量,以逗号分隔变量,此时可以在运动过程中同

时输出两个 ProcessOut 变量中设置的信号。

【*OffsetJ*】：对当前点位进行关节偏移，偏移不受点位类型的影响，格式为如下：OffsetJ(P0,Δj1,Δj2,Δj3,Δj4,Δj5,Δj6)

其中 Δ j1, Δ j2, Δ j3, Δ j4, Δ j5, Δ j6 分别表示对应关节的偏移量，单位为（°）；该偏移功能不会修改要偏移的点位（这里的 P0），只是在运动时计算出的临时偏移点位。

【*OffsetC*】：对当前点位进行笛卡尔偏移，偏移不受点位类型的影响，格式为如下：OffsetC(P0,Δx,Δy,Δz,Δrz,Δry,Δrx)

其中 Δ x, Δ y, Δ z, Δ rz, Δ ry, Δ rx 分别表示对应笛卡尔坐标的偏移量，单位为（mm/s|°/s）；该偏移功能不会修改要偏移的点位（这里的 P0），只是在运动时计算出的临时偏移点位。

➢ 注意

✧ 圆弧由当前点、过渡点、目标点拟合而成，若其中两点或三点是同一位置，圆弧将退化为直线，此时机器人在运动过程中会按直线运动，且不会报警。现场应用中应避免圆弧中的点位相同，否则可能会出现指令和轨迹执行错误的情况。

➢ 示例

```
MoveC P0,P1,V80,Zone0,Tool0;
```

//圆弧运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0。

```
MoveC P0,P1,V80,Zone0,Tool0,Till X0 == true;
```

//圆弧运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0，开启 Till 功能，条件表达式满足时触发停止

```
MoveC P0,P1,V80,Zone0,Tool0,Find X0 == true;
```

//圆弧运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0，开

启 Find 功能，条件表达式满足时触发寻位功能

MoveC P0,P1,V80,Zone0,Tool0,Singular 2;

//圆弧运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0，奇异规避类型设置为 2

MoveC P0,P1,V80,Zone0,Tool0,ProcessOut process1,process2;

//圆弧运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0，设置过程输出参数为 process1 和 process2。

MoveC OffsetC(P0,10,10,10,0,0,0),P1,V80,Zone0,Tool0;

//假设 P0 为基座标系下示教的点位，坐标为(300,350,290,50,60,70)，执行 OffsetC 偏移后的坐标为：(310,360,300,50,60,70)。机器人末端将运动到该位置，但原有的 P0 坐标值不变。

MoveC OffsetJ(P0,10,10,10,0,0,0),P1,V80,Zone0,Tool0;

//假设 P0 为关节坐标系下示教的点位，坐标为(0,-90,0,0,0,0)，执行 OffsetJ 偏移后的坐标为：(10,-80,10,0,0,0)。机器人末端将运动到该位置，但原有的 P0 坐标值不变。

MoveC OffsetJ(P1,10,10,10,0,0,0),P2,V80,Zone0,Tool0;

//假设 P1 为基坐标系下示教的点位，坐标为(300,350,290,50,60,70)，对应的关节坐标为(30,40,50,10,20,25)(以实际为准，这里仅供说明)，执行 OffsetJ 偏移后的基坐标为：(305,330,270,50,60,70)（以实际为准，这里仅供说明），对应的关节坐标为（40，50，60，10，20，25）。机器人末端将运动到该位置，但原有的 P1 坐标值不变。

2.5 门形运动 Jump JumpJ JumpL

➢ 指令说明

门形运动指令，指机械手以门型运动轨迹从起始位置运动到目标位置。运动过程分为三段：首先垂直上升至设定的水平运动高度 LimitZ，然后水平移动到

目标点上方，最后垂直下降到达目标点。门形运动由三段轨迹组成，分别为三段关节、三段直线、关节+直线形式，其对应的运动指令分别为 JumpJ、JumpL、Jump。

➢ 语法格式

Jump/JumpJ/JumpL [目标点],[速度],[过渡],[工具],[*工件*],[*Till* *],

[*Sense* *],[*Find* *],[*Singular* *],[*ProcessOut* #];

➢ 参数

【目标点】：要到达的目标点变量，可以是数组元素或单个点位变量，也可对点位进行关节偏移或笛卡尔坐标偏移。偏移功能的关键字为 OffsetJ 和 OffsetC.该功能可以在指令编辑时，修改点位的条件表达式，在功能页面开启。

【速度】：到达目标点过程的速度设定值，JumpL 轨迹的速度单位为 mm/s，Jump 和 JumpJ 的速度单位是%。

【过渡】：设置 CP 过渡的大小。

【工具】：设置运动到目标点所用的工具号。

✧ 可选参数

【*工件*】：指定运动所需的工件坐标系，使用前需先标定工件坐标系。

【*Till **】：添加该参数时，以 Till 关键字标识，后加条件表达式（如：Till X0 == true）；其后的条件表达式满足时，将触发停止运动，继续运动下一行指令。

【*Sense **】：添加该参数时，以 Sense 关键字标识，后加条件表达式（如：Sense X0 == true）；到达目标点是否过渡，其后的条件表达式满足时该信号将被触发，此时不进行过渡，并停止运动在目标点上方。

【*Find **】：添加该参数时，以 Find 关键字标识，后加条件表达式（如：Find X0 == true）；其后的条件表达式满足时，将读取当前的坐标写入到 FindPos 变量内。Find 功能支持轨迹类型为全轨迹类型，总线周期任务记录该点

位的所有点位信息以及触发信号时记录的反馈位姿或关节角度(取决于该点位设置的坐标系类型，配置为关节坐标系时记录为反馈关节角度，其余坐标系类型反馈末端法兰对于基座标的位姿）、手系、cf 配置、外部轴反馈角度。

【*Singular *】：添加该参数时，以 Singular 关键字标识，后加整型常量或整型变量（如 Singular 1）；指令中存在 Singular 关键字时，将根据其后设置的规避类型执行运动。

【*ProcessOut #*】：添加该参数时，以 ProcessOut 关键字标识，后加整型常量或整型变量（如 ProcessOut process1,process2）;ProcessOut 关键字之后可跟两个 ProcessOut 类型的变量，以逗号分隔变量，此时可以在运动过程中同时输出两个 ProcessOut 变量中设置的信号。

【*OffsetJ*】：对当前点位进行关节偏移，偏移不受点位类型的影响，格式为如下：OffsetJ(P0,Δj1,Δj2,Δj3,Δj4,Δj5,Δj6)

其中Δj1, Δj2, Δj3, Δj4, Δj5, Δj6 分别表示对应关节的偏移量，单位为（°）；该偏移功能不会修改要偏移的点位（这里的 P0），只是在运动时计算出的临时偏移点位。

【*OffsetC*】：对当前点位进行笛卡尔偏移，偏移不受点位类型的影响，格式为如下：OffsetC(P0,Δx,Δy,Δz,Δrz,Δry,Δrx)

其中Δx, Δy, Δz, Δrz, Δry, Δrx 分别表示对应笛卡尔坐标的偏移量，单位为（mm/s|°/s）；该偏移功能不会修改要偏移的点位（这里的 P0），只是在运动时计算出的临时偏移点位。

➢ 示例

✧ 以下示例中的 Jump 可替换为 JumpJ 或 JumpL，本例仅以 Jump 进行说明。

```
Jump P0,V80,Zone0,Tool0;
```

//门型运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0。

```
Jump P0,V80,Zone0,Tool0,Till X0 == true;
```

//门型运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0，

开启 Till 功能，条件表达式满足时触发停止

Jump P0,V80,Zone0,Tool0,Find X0 == true;

//门型运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0，开启 Find 功能，条件表达式满足时触发寻位功能

Jump P0,V80,Zone0,Tool0,Sense X0 == true;

//门型运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0，开启 Sense 功能，条件表达式满足时触发 Sense 功能

Jump P0,V80,Zone0,Tool0,Singular 2;

//门型运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0，奇异规避类型设置为 2

Jump P0,V80,Zone0,Tool0,ProcessOut process1,process2;

//门型运动方式、速度 80%到达目标点 P0，CP 过渡设置为 0，工具号设为 0，设置过程输出参数为 process1 和 process2。

Jump OffsetC(P0,10,10,10,0,0,0),V80,Zone0,Tool0;

//假设 P0 为基座标系下示教的点位，坐标为(300,350,290,50,60,70)，执行 OffsetC 偏移后的坐标为：(310,360,300,50,60,70)。机器人末端将运动到该位置，但原有的 P0 坐标值不变。

Jump OffsetJ(P0,10,10,10,0,0,0),V80,Zone0,Tool0;

//假设 P0 为关节坐标系下示教的点位，坐标为(0,-90,0,0,0,0)，执行 OffsetJ 偏移后的坐标为：(10,-80,10,0,0,0)。机器人末端将运动到该位置，但原有的 P0 坐标值不变。

Jump OffsetJ(P1,10,10,10,0,0,0),V80,Zone0,Tool0;

//假设 P1 为基坐标系下示教的点位，坐标为(300,350,290,50,60,70)，对应的关节坐标为(30,40,50,10,20,25)(以实际为准，这里仅供说明)，执行 OffsetJ 偏移后的基坐标为：(305,330,270,50,60,70) (以实际为准，这里仅供说明)，对应的关节坐标为（40，50，60，10，20，25）。机器人末端将运动到该位置，但原有的 P1 坐标值不变。

3 IO 指令

IO 指令主要用于操作机器人系统的数字输出端口，用于控制机械动作或发出信号。可以单独输出一个端口，也可以成组的输出多个端口。

3.1 单独输出 SetDO

➢ 指令说明

操作 IO 输出的状态，可操作输出 Y 线圈，也可以设置输出信号的持续时间，时间到达后将对信号取反输出；有阻塞效果，SetDo 会等待运动结束才会执行后续的指令。

➢ 语法格式

SetDO [输出线圈],[布尔值],[*持续时间#*];

➢ 参数

【输出线圈】：要输出的线圈地址，如 Y10。

【布尔值】：输出值，true 或 false，不支持布尔变量。

✧ 可选参数

【*持续时间#*】：输出值的有效时间（单位 ms），计时器在后台执行，不影响前台指令运行；时间到达后，输出值将被取反后再次输出。

➢ 注意

✧ SetDo 指令的持续时间是后台计时，若该指令后有 WaitTime 指令，不会等到 WaitTime 指令延时结束再开始计时，即两者独立计时互不影响，但是计时过程可能会重叠。

✧ 后台的持续时间计数器共有 20 个，当设置的持续时间过长且处于空闲状态的计数器为 0 时，将输出错误代码。使用时应合理设置指令逻辑，避免计数器超限。

➢ 示例

```
SetDO Y2,true,ConstTime 1000;
//输出 Y2 值为 true，1000ms 后输出为 false
SetDO Y2,false,ConstTime 1000;
//输出 Y2 值为 false，1000ms 后输出为 true
```

3.2 组输出 SetGOut

➢ 指令说明

将指定的整型变量值换算成 8 位二进制数，并输出到指定的用户 IO 的 Y 线圈序列中。组输出最大 IO 数量为 8 位。

➢ 语法格式

```
SetGOut [组输出变量],[输出值];
```

➢ 参数

【组输出变量】：在程序数据区建立的输出的 Y 线圈序列，范围 2~8 个。

【输出值】：8 位二进制数对应的十进制整型数值，取值 0~255，实际设置的值不能大于设置的组输出 Y 序列对应的十进制值，否则产生错误。可同时输出 2-8 个 Y 线圈序列的值，若输出不足 8 位，则高位默认补 0，序列可不连续。本体 Y 线圈不可组输出，且不提供操作接口。

表 3.2 组输出示意

端口	1-Y17	1-Y16	1-Y15	1-Y14	1-Y13	1-Y12	1-Y11	1-Y10
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
布尔值	false	false	false	true	false	true	true	true
二进制	0	0	0	1	0	1	1	1
十进制	23							

- 示例

SetGOut signalGo1,23;

//signalGo1 变量设置的 Y 线圈序列如表 3.2 所示，输出结果为表 3.2 所示的布尔值对应的行。

4 控制指令

流程控制指令包括程序执行的逻辑判断、状态跳转、函数调用、循环等，可以构建程序执行的全局逻辑。

4.1 条件指令 If

➢ 指令说明

用于判断条件成立时，执行 If 和 Endif 之间的程序块，If 条件语句必须使用 Endif 结尾，否则程序运行时将产生错误。

➢ 语法格式

```
If [条件表达式] Then
    < 语句块 >
Elseif [条件表达式] Then
    < 语句块 >
Else
    < 语句块 >
Endif
```

➢ 参数

【条件表达式】：用于执行逻辑语句的必要表达式，如 num1 == 1。支持添加一个逻辑运算符（and 或 or）。

【语句块】：条件满足或不满足时需要执行的逻辑或动作语句。

➢ 注意

✧ If 条件语句可以嵌套使用，嵌套层数最多为 3 层，内嵌的语句块没有语法

限制。

- ✧ 如果直接从 If 语句块内执行，则后续的执行逻辑是不确定的，应保证从 If 语句开始执行。
- ✧ 单个 If 或者 Elseif 中最多支持两个条件判断。

➢ 示例

```
If num1 > 6 Then
    SetDO Y0,true;
Elseif num2 < 9 Then
    SetDO Y1,true;
Else
    SetDO Y1,true;
Endif
```

4.2 有限循环 For

➢ 指令说明

有限次的重复执行循环体内的语句块，可指定变量的初始值、结束值、步长。循环体内可以嵌套 For 循环或 While 循环，以及其他语句块。

➢ 语法格式

```
For [循环变量] From [初始值] To [结束值] Step [步长] Do;
    < 语句块 >
Endfor
```

➢ 参数

【循环变量】：用于存储循环次数的整型变量。

【初始值】：循环执行的起始值，类型为整型。

【结束值】：循环执行的结束值，类型为整型。

【步长】：每次循环后的递增量，类型为整型。

➢ 注意

✧ 若初始值小于结束值，且步长大于 0，则循环的结束条件为循环变量大于结束值；若初始值大于结束值，且步长小于 0，则循环的结束条件为循环变量小于结束值。若不满足循环的结束条件，可能会造成死循环。

✧ 若直接从循环体内的语句开始执行，则不会对循环变量初始化，循环的结束条件由循环变量的当前值确定。

➢ 示例

```
For num1 From 1 To 10 Step 1 Do
    num2 = num2 + 1;
Endfor
```

//假设 num1 循环变量的初始值为 0，num2 变量的初始值也为 0，执行该循环后，num2 的结果为 10。

4.3 条件循环 While

➢ 指令说明

重复执行循环体内的语句块，直到条件不满足时结束循环，结束条件由条件表达式的运算结果决定。循环体内的语句块可以是单条指令或其他循环，也可以是具有一定逻辑功能的语句块。

➢ 语法格式

```
While [条件表达式] Do
    < 语句块 >
Endwhile
```

➢ 参数

【条件表达式】：运算结果为 false 时，结束循环。

➢ 注意

✧ While 循环先判断表达式的结果，再执行循环体，若首次执行时条件表达式的运算结果为 false，则不执行循环体内的语句，直接退出循环。

➢ 示例

```
While X1 == true Do
    num2 = num2 + 2;
Endwhile
```

//X1 线圈的值为 true 时，执行循环体内的算数表达式 num2 = num2 + 2;

当 X1 为 false 时，结束循环。

4.4 标签 Label

➢ 指令说明

在插入标签的位置做跳转标记，与 Goto 指令配合使用，顺序执行程序时，若 Goto 语句在该标签之后，越过该标签不做任何处理。

➢ 语法格式

```
Label [标签];
```

➢ 参数

【标签】：标记跳转位置的字符串标识。

➢ 注意

✧ 若指令中没有对应的 Goto 该标签的语句，则该标签执行时将被忽略。

✧ 标签的位置不应单独设置在 If、While、For 内，若从其他位置直接跳转至上输入控制指令内，可能会造成意想不到的的结果。

✧ 标签指令的作用域为所在函数。

➢ 示例

```
num1 = num2 * 3;
Label tab1;
If num1 > 6 Then
  SetDO Y0,true;
Elseif num2 < 9 Then
    SetDO Y1,true;
Else
    SetDO Y1,true;
Endif
Goto tab1;
```

//首次执行到 Label tab1;语句时，该语句将被忽略，执行到 Goto tab1 时，程序将跳转到 tab1 标记的位置往下执行。

4.5 跳转 Goto

➢ 指令说明

跳转到该指令指定的标签位置后，执行标签之后的程序。

➢ 语法格式

```
Goto [标签];
```

➢ 参数

【标签】：要跳转到的标签名称，与 Label 指令的标签名称对应。

➢ 注意

- ✧ Goto 指令只能跳转到已设置标签位置的指令行，且 Goto 指令的标签名称必须与 Label 指令的标签名称严格一致，否则将无法跳转或发生错误。Goto 指令是无条件跳转，若要依赖条件进行跳转，可与 If 指令配合使用。
- ✧ 该指令是无条件跳转指令，应格外注意跳转的目标位置是否在 If、While、For 等控制指令中，使用不当可能会造成死循环或意想不到的结果。
- ✧ 跳转指令可能会扰乱程序的执行逻辑，若无特殊需要，不建议在程序中大量使用跳转指令。
- ✧ 该指令的作用域为所在函数，只能跳转至所在函数内对应的标签。

➢ 示例

```
num1 = num2 * 3;
Label tab1;
If num1 > 6 Then
    SetDO Y0,true;
Elseif num2 < 9 Then
    SetDO Y1,true;
Else
    Goto tab1;
Endif
```

//首次执行到 Label tab1;语句时，该语句将被忽略，执行到 Goto tab1 时，程序将跳转到 tab1 标记的位置往下执行。

4.6 函数调用 Call

➢ 指令说明

函数是一个可重用的代码块，用于实现特定的功能。编程时将可重用的代码块写入到一个函数内，函数的名称唯一且无形式参数和返回值，通过 Call 指令可调用该函数执行特定功能。

➢ 语法格式

Call [函数名称];

➢ 参数

【函数名称】：用户自定义的可被重复调用的函数，函数名称是函数的唯一标识，且不可用“main”开头的单词命名。

➢ 注意

✧ 例如：“main1”、“main2”、“main3”、“main4”、“main5”分别对应五个线程的主函数，主函数为整个线程的程序入口。函数可以多级调用，如 A 调用 B，B 调用 C,C 调用 D。最多可以调用三层。

✧ 若开始执行程序时，程序指针指向被调函数内的指令，则执行到被调函数的 Endfunc 时，程序指针将跳转到被调函数的函数头处继续执行，不会跳回到主函数内。

✧ 若程序直接从被调函数内部开始执行，则被调函数的执行模式与当前设置的运行模式保持一致，如当前运行模式为周期，则执行到 Endfunc 时结束。

➢ 示例

```
Func main1()
```

```
    ……
    Call math();
Endfunc
Func math()
    num3 = num1 + num2;
Endfunc
//在 main1()主函数内调用子函数 math()
```

5 运算指令

运算指令用于数学运算，赋值，取值等操作。参与运算的数据类型不局限于整型和浮点型，如 Points、Bool 等类型。

5.1 赋值 =

➢ 指令说明

将“=”号右侧的运算结果赋值给左侧的变量。

➢ 语法格式

[赋值目标] = [赋值源];

➢ 参数

【赋值目标】：被赋值的变量，包括的类型为：Arch、Bool、Float、Int、Points、Speed、Zone、Seamdata、Weavedata、Welddata。

【赋值源】：可以是常量、变量或算术表达式，运算结果类型必须与赋值目标类型一致。

➢ 注意

✧ 赋值运算不支持隐式转换，若类型不一致将产生错误。

➢ 示例

```
num3 = num1 * 3 + num2;
//数学运算
```

```
Bool1 = true;
//常量赋值
P1 = P2;
//点位变量赋值
```

5.2 赋值到寄存器 SetValue

➢ 指令说明

将指定类型的数据赋值到指定地址和长度的 HD 数据寄存器或 M 线圈内。

➢ 语法格式

```
SetValue [寄存器/线圈类型],[起始地址],[数据类型],[变量名称];
```

➢ 参数

【寄存器/线圈类型】：被赋值的是 HD 寄存器或 M 线圈。

【起始地址】：寄存器或线圈的起始地址。

【数据类型】：赋值变量的数据类型，Int、Float 类型可赋值给 HD 寄存器，Bool 类型可赋值给 M 线圈，其他方式不可赋值。

【变量名称】：赋值变量的名称。

➢ 注意

✧ 赋值的长度取决于变量的类型，若变量是数组，则数组的大小决定赋值到 HD 寄存器或 M 线圈的个数；若非数组，则 Int、Float 类型可赋值两个 HD 寄存器，Bool 类型可赋值一个 M 线圈。

✧ HD 寄存器地址只能为偶数，不能为奇数，否则将输出错误。

➢ 示例

SetValue HD,1000,Int,num1;

//若 num1 为长度 10 的 Int 型数组，则 HD1000~HD1019 组成的 10 个双字将被赋值为 num1 数组的所有元素值。若 num1 为 Int 型变量，则 HD1000 和 HD1001 组成的双字将被赋值为 num1 的值。

SetValue M,1000,Bool,bool1;

//若 bool1 为长度 10 的 Bool 型数组，则 M1000~M1009 的 10 个 M 线圈将被赋值为 bool1 数组的所有元素值。若 bool1 为 Bool 型变量，则 M1000 将被赋值为 bool1 的值。

5.2 取寄存器值 GetValue

➢ 指令说明

读取指定地址和长度的 HD 数据寄存器或 M 线圈内的值，赋值到指定的变量中。

➢ 语法格式

GetValue [寄存器/线圈类型],[起始地址],[数据类型],[变量名称];

➢ 参数

【寄存器/线圈类型】：要读取的是 HD 寄存器或 M 线圈。

【起始地址】：寄存器或线圈的起始地址。

【数据类型】：要被赋值变量的数据类型，Int、Float 类型可赋值给 HD 寄存器，Bool 类型可赋值给 M 线圈，其他方式不可赋值。

【变量名称】：要被赋值变量的名称。

➢ 注意

✧ 赋值的长度取决于变量的类型，若变量是数组，则数组的大小决定读取 HD 寄存器或 M 线圈的个数；若非数组，则 Int、Float 类型可读取两个 HD 寄存器，Bool 类型可读取一个 M 线圈。

✧ HD 寄存器地址只能为偶数，不能为奇数，否则将输出错误。

➢ 示例

GetValue HD,1000,Int,num1;

//若 num1 为长度 10 的 Int 型数组，则读取 HD1000~HD1019 组成的 10 个双字的值赋值给 num1 数组的所有元素。若 num1 为 Int 型变量，则 HD1000 和 HD1001 组成的双字将被赋值给 num1。

GetValue M,1000,Bool,bool1;

//若 bool1 为长度 10 的 Bool 型数组，则 M1000~M1009 的 10 个 M 线圈将赋值给 bool1 数组的所有元素。若 bool1 为 Bool 型变量，则 M1000 将赋值给 bool1。

5.3 获取当前基座标 GetCurCPos

➢ 指令说明

读取当前基座标系下带工具的坐标值，并赋值给指定的点位变量。

➢ 语法格式

GetCurCPos [点位变量];

➢ 参数

【点位变量】：将读取到的当前坐标值写入到指定的点位变量内。

➢ 注意

✧ 读取当前位置时，会一并将当前的工具号、手系、坐标系、工件号以及 Cf 参数读取，并全部写入到指定的点位变量内。

➢ 示例

GetCurCPos P0;

//获取当前坐标系下对应工具号的坐标值写入到 P0 点位中。

GetCurCPos P1[1,1];

//获取当前坐标系下对应工具号的坐标值写入到 P1[1,1]数组点位中。

5.4 获取当前关节坐标 GetCurJPos

➢ 指令说明

读取当前关节坐标系下的关节坐标值，并赋值给指定的点位变量。

➢ 语法格式

GetCurCPos [点位变量];

➢ 参数

【点位变量】：将读取到的当前坐标值写入到指定的点位变量内。

➢ 注意

✧ 读取当前位置时，会一并将当前的工具号、手系、坐标系、工件号以及 Cf 参数读取，并全部写入到指定的点位变量内。

➢ 示例

GetCurJPos P0;

//获取当前坐标系下对应工具号的坐标值写入到 P0 点位中。

GetCurJPos P1[1,1];

//获取当前坐标系下对应工具号的坐标值写入到 P1[1,1]数组点位中。

6 等待指令

等待指令主要用于程序的时序控制，可阻塞程序的执行，强制等待指定的时间；也可根据条件表达式的运算结果，决定等待结束的时机。

6.1 延时等待 WaitTime

➢ 指令说明

阻塞程序的执行，在插入该指令的位置延时等待指定的时间。

➢ 语法格式

WaitTime [时间];

➢ 参数

【时间】：要等待的时间，单位为 ms，输入范围是 0~65535。

➢ 注意

✧ 若要精确延时，建议延时时间不低于 20ms；若要等待运动结束再执行某项非运动指令，可在运动指令后添加 WaitTime 0;语句，使其运动结束后延时等待 0ms。

✧ 若输入的等待时间为负数或大于 65535，则产生错误。

➢ 示例

```
MoveL P0,V80,Zone0,Tool0;
WaitTime 5000;
```

SetDO Y2,true;

//直线运动到 P0 后，等待 5000 毫秒后输出 Y2 为 true

6.2 条件等待 WaitCondition

➢ 指令说明

阻塞程序执行，直到条件表达式的运算结果为 true 时才执行后续指令。若添加最大等待时间，在等待时间到达时无论条件表达式的运算结果是否为 true，均解除阻塞，继续执行后续指令。

➢ 语法格式

WaitCondition [条件表达式],[*TimeMax #*];

➢ 参数

【条件表达式】：运算结果决定是否解除阻塞执行后续指令。

✧ 可选参数

【*TimeMax #*】：最大等待时间，单位为 ms，范围是 0~2147483648。达到设定的最大等待时间后，无论条件表达式的运算结果是否为 true，均解除阻塞执行后续指令。若在最大时间内条件表达式的运算结果为 true，则立即解除阻塞并执行后续指令。

➢ 注意

✧ 若输入的等待时间为负数，或大于 2147483648，则产生错误。

✧ 在运动线程中，最大等待时间从运动指令执行结束后开始计时。

✧ 该指令前一条为运动指令时，在前一条运动指令开始运动时，就会判断该指令中的条件表达式是否满足，若满足，可立刻预读后一条运动指令的点位，

可实现 CP 过渡。若要等待前一条运动指令运动结束时再判断该指令的条件，可在该指令前添加 WaitTime 0;指令。

➢ 示例

```
MoveL P0,V80,Zone0,Tool0;
WaitCondition X0 == true,TimeMax 5000;
SetDO Y2,true;
```

//直线运动到 P0 后，5000 毫秒内到达后条件表达式运算结果为 true，继续执行 SetDo 指令,否则等待 5000 毫秒后再执行 SetDo 指令。

7 系统指令

系统指令主要用于常用的打印信息、自定义报警、询问窗口、添加注释、全局参数设置等。

7.1 注释 //

➢ 指令说明

为程序指令添加说明信息。注释独占一行，最大长度不超过 400 个英文字符或 100 个汉字，添加位置不受限制。

➢ 语法格式

//［注释内容］;

➢ 参数

【注释内容】：可以是中英文字符串，最大可写 400 英文字符（包括空格及其他可见字符）或 100 个汉字（包括空格）。

➢ 示例

```
MoveL P0,V80,Zone0,Tool0;
//直线运动至 P0 点
MoveL P1,V80,Zone0,Tool0;
//直线运动至 P1 点
```

7.2 询问窗口 AskQuestion

➢ 指令说明

执行该指令时，将弹出询问窗口，用户点击对应的按钮后赋值给指定变量。可设置最大等待时间，时间到达时未按下按钮则产生错误。根据按下的键，返回数值 1~5 的变量。如果按下功能键 1，则返回 1，以此类推。

➢ 语法格式

AskQuestion [变量],[窗口信息],[按钮 1], [按钮 2], [按钮 3], [按钮 4], [按钮 5],[*TimeMax #*];

➢ 参数

【变量】：用于存储按钮的索引。

【窗口信息】：窗口弹出时显示的字符串信息。

【按钮】：最大可设置五个按钮，按钮索引为 1～5，按钮的名称为字符串，最大长度为 64 英文字符或 21 个汉字，当 5 个按钮全显示时，建议使用时汉字控制在 4 个以内，汉字过长将显示不全。

✧ 可选参数

【*TimeMax #*】：窗口最大显示时间，最大时间内用户如果未触发按钮，则产生报警并停止程序运行。

➢ 注意

✧ 若按钮的字符串值为空，则该按钮不显示。

✧ 变量的赋值由按钮的排列顺序决定，与指令中空字符按钮所在位置无关，按钮从左至右对应的索引为 1 ～ 5 。

✧ 该指令执行时会阻塞所有线程中的 Print 指令，以防止窗口信息被覆盖。当窗口关闭时，阻塞将被解除，print 可继续执行。

➢ 示例

```
If X0 == true Then
    AskQuestion num1,msg,btn1,btn2,btn3,btn4,btn5,TimeMax 5000;
Endif
```

//当 If 条件满足时，执行 AskQusetion 指令后显示询问窗口，窗口的消息是 msg 字符串变量的值;若用户在 5 秒内按下了显示 btn2 字符串的按钮，则 num1 变量将被赋值为 2；若用户 5 秒内没有任何操作，将输出超时报警。

7.3 打印 Print

➢ 指令说明

将指定的字符串在消息窗口打印输出，单条打印字符串长度不超过 64 字符或 21 个汉字。

➢ 语法格式

```
Print [变量];
```

➢ 参数

【变量】：存放打印消息的变量，包括字符串变量、字符串常量、整型变量(Int)、浮点变量(Float)、点位变量(Points)。

➢ 注意

✧ 两条 Print 指令的执行应间隔 100ms 以上，若低于 100ms，系统将根据队列情况自动舍去示教器未读取到的消息，显示当前最新消息。

- ✧ 若自定义消息长度超过 64 字节，系统将舍去超过部分，仅显示有效长度内的字符消息。
- ✧ 若打印点位变量，示教器消息窗口仅显示基础坐标，不包含 Cf、工具、手系等其他信息。
- ✧ 打印消息时，系统自动补全输出当前消息的线程号。
- ✧ 该指令的优先级低于 AskQuestion 指令和 Error 指令，与前述指令同时执行时，Print 指令将被阻塞，等待 Print 队列缓存消息输出完成后，显示前述 AskQuestion 和 Error 指令的消息。

➢ 示例

```
Print “这是一条输出信息”
```

//在示教器的消息窗口将显示一条最新消息：“这是一条输出信息”。

```
Print StrMsg;
```

//若 StrMsg 是字符串变量，内容是“这是一条输出信息”，将在示教器窗口输出：“这是一条输出信息”。

```
Print P0;
```

//若 P0 是点位变量，将在示教器消息窗口输出 P0 的坐标值。

7.4 自定义错误代码 Error

➢ 指令说明

以弹窗形式输出用户自定义的错误信息，并终止所有线程的执行。

➢ 语法格式

```
Error [字符串];
```

➢ 参数

【字符串】：要显示的自定义错误消息内容，可以是字符串变量或常量。

➢ 注意

- ✧ Error 指令应在逻辑指令中添加，且执行一次，请勿在非运动线程中循环执行，否则可能导致错误无法清除。
- ✧ 自定义错误的报警代码为 10700。
- ✧ 字符串最长支持 64 个字节，即不超过 64 个字符或 21 个汉字。
- ✧ 该指令执行时会阻塞所有线程中的 Print 指令，以防止窗口信息被覆盖。当窗口关闭时，阻塞将被解除，print 可继续执行。

➢ 示例

```
Error errStr;
//输出错误代码 10700，弹窗显示自定义错误消息。
```

7.5 全局速度 VelSet

➢ 指令说明

设定全局速度系数，该指令后续的所有运动指令的速度都会乘以该系数，直到执行下一条 VelSet 指令时才会更改系数。

➢ 语法格式

```
VelSet [系数变量];
```

➢ 参数

【系数变量】：用于设置全局速度的系数，默认值是 100，单位是%，范围是 1~100。

➢ 注意

✧ 速度系数影响机器人末端运行速度，其算数关系为：

实际末端运行速度=基准速度×状态栏速度×指令速度×全局速度系数。

✧ 程序停止、编译、急停等操作不会影响系数，只有重新执行该指令后才会修改全局速度系数。

➢ 示例

```
VelSet 50;
MoveL P0,V80,Zone0,Tool0;
//运动速度降低为原来的 50%
```

8 外部轴指令

外部轴指令包含外部轴激活和关闭、外部轴运动等功能。

8.1 外部轴开关 SetExtSys

➢ 指令说明

打开或关闭外部轴系统。打开后可执行外部轴运动，关闭后，外部轴运动失效。

➢ 语法格式

SetExtSys [外部轴系统号],[Bool 变量];

➢ 参数

【外部轴系统号】：要操作的外部轴系统编号，范围 1~2。

【Bool 变量】：对指定的外部轴系统开启或关闭，true 表示开启，false 表示关闭。

➢ 示例

```
SetExtSys 1,true;
//开启外部轴系统 1
```

8.2 外部轴运动

➢ 指令说明

外部轴坐标信息存储在 Points 类型的点位中，若要执行外部轴运动，可使用 MoveJ 或 MoveL 指令触发运动。

➢ 语法格式

MoveJ/MoveL [外部轴目标点],[速度],[过渡],[工具];

➢ 参数

【外部轴目标点】：外部轴要运动到的目标位置。

【速度】：外部轴的运动速度，单位%，范围是 1~100。

【过渡】：外部轴过渡参数，单位%，范围是 1~100。

【工具】：对外部轴运动无效，默认选择 Tool0 即可。

➢ 注意

✧ 外部轴运动模式目前仅支持“机器人不运动且外部轴单独运动”模式，暂不支持协同运动模式。

➢ 示例

```
SetExtSys 1,true;
Movej extP0,V100,Zone0,Tool0;
WatiTime 0;
SetExtSys 1,false;
```

//开启外部轴系统 1，然后运动至目标点 extP0，等待运动结束后关闭外部轴系统 1。

9 计时指令

计时指令用于管理计时时钟的开闭，以及测量某一段程序执行所耗费的时间。一般可用于节拍测量、程序执行周期测量等。系统提供 10 个计时器供用户自由使用。

9.1 开启计时器 ClockStart

➢ 指令说明

开启对应编号的计时器时钟，时钟开启后立刻计时，计时时间不受程序启停的影响。

➢ 语法格式

ClockStart [计时器编号];

➢ 参数

【计时器编号】：要操作的计时器的编号，范围是 1~10。

➢ 注意

✧ ClockStart 指令不会影响运动指令的过渡效果，由于点位预读可能会导致 ClockStart 指令在运动未结束时执行，若要等待运动结束时开始计

时，可在运动指令后加一条 WaitTime 0；指令。

➢ 示例

```
Label tab1;
ClockStart 1;
MoveJ P0,V80,Zone0,Tool0;
MoveJ P1,V80,Zone0,Tool0;
WaitTime 0;
ClockStop 1,num1;
Goto tab1;
```

//开启计时器 1，等待 P1 运动结束后，停止计时器并输出计时时间至 num1。

9.2 关闭计时器 ClockStop

➢ 指令说明

关闭指定编号的计时器。

➢ 语法格式

ClockStop [计时器编号],[输出时间];

➢ 参数

【计时器编号】：要操作的计时器的编号，范围是 1~10。

【输出时间】：存储输出的计时时间，Int 型变量，单位是 ms。

➢ 注意

✧ ClockStop 指令不会影响运动指令的过渡效果，由于点位预读可能会导致 ClockStop 指令在运动未结束时执行，若要等待运动结束时停止计时，

可在运动指令后加一条 WaitTime 0；指令。

➢ 示例

```
Label tab1;
ClockStart 1;
MoveJ P0,V80,Zone0,Tool0;
MoveJ P1,V80,Zone0,Tool0;
WaitTime 0;
ClockStop 1,num1;
Goto tab1;
```

//开启计时器 1，等待 P1 运动结束后，停止计时器并输出计时时间至 num1 内。

9.3 重置计时器 ClockReset

➢ 指令说明

重置指定编号的计时器，已计时时间将被清零。

➢ 语法格式

ClockReset [计时器编号];

➢ 参数

【计时器编号】：要操作的计时器的编号，范围是 1~10。

➢ 注意

✧ 计时器开启后就会在后台持续计时，与程序是否运行无关，直到执行了 ClockStop 指令才会停止计时；再次执行 ClockStart 指令时会从上次执

行 ClockStop 时的时间继续计时，直到执行了 ClockReset 指令才会将计时时间清零。

✧ ClockReset 指令不会影响运动指令的过渡效果，由于点位预读可能会导致 ClockReset 指令在运动未结束时执行，若要等待运动结束时停止计时，可在运动指令后加一条 WaitTime 0；指令。

➢ 示例

```
ClockStart 1;
Label tab1;
ClockReset 1;
MoveJ P0,V80,Zone0,Tool0;
MoveJ P1,V80,Zone0,Tool0;
WaitTime 0;
ClockStop 1,num1;
Goto tab1;
```

//开启计时器 1，将计时器 1 的计时值清零，等待 P1 运动结束后，停止计时器并输出计时时间至 num1 内，然后再跳转至 tab1 处重复执行上述动作。

10 错误处理指令

错误处理指令用于程序执行过程中出现的简单用户错误的处理，当发生指定范围内的错误代码时，系统将自动进入用户编写的错误处理程序，若错误处理成功，程序将继续往下执行，若处理失败，则最大尝试 4 次重新处理；4 次处理均失败后，将输出发生该错误的代码。用户可处理的错误代码如下表所示：

表 10 可处理的错误代码

错误代码	错误原因
10604	条件表达式中的除数为 0
10605	数组下标的乘积大于 200
10608	WaitTime 的延时时间大于 65535 毫秒
10609	WaitCondition 指令的最大时间不在 0 至 2147483648 内
10618	料盘编号大于 20
10619	料盘分区数小于等于 0
10621	料盘点位序号小于 1 或大于设定的点位总数
10622	4 点示教料盘时，分区数量等于 1
10624	数组下标越界
10632	焊接速度不在合理范围内
10634	在设定时间内起弧失败
10635	在设定时间内收弧失败
10640	给定焊接参数错误
10653	组输出指令的操作值大于已选择的 IO 对应的十进制值
10665	寄存器地址为奇数
10669	指令中的线速度大于设定的基准线速度
10674	Singular 参数取值不在 0~3 范围内

10676	输入的计时器编号不在 1~10 范围内
10678	速度系数不在 1~100 范围内

10.1 重试 Retry

➢ 指令说明

发生错误时，程序将跳转至错误处理程序内执行，若在错误处理程序中执行了 Retry 指令，程序将跳转至发生错误的指令重新尝试执行，重试执行 4 次若未解除错误，则输出错误代码。该指令必须添加至错误处理程序中才可生效。

➢ 语法格式

```
Retry;
```

➢ 参数

【无参数】

➢ 示例

```
Func main1()
    num2=1;
    num1 = num2 - 1;
    num5 = num4 / num1;
    num2 = num2 + 1;
Errorfunc
    If ErrNo == 10604 Then
        num1 = 1;
        Retry;
    Endif
```

```
Endfunc
```

//当 num1=0 时，将发生 10604 错误，随后进入错误处理程序，将 num1 赋值为 1 后，重新执行 num5 = num4 / num1;后无错误，程序将继续正常运行。

10.2 尝试下一行 TryNext

➢ 指令说明

发生错误时，程序将跳转至错误处理程序内执行，若在错误处理程序中执行了 TryNext 指令，程序将跳转至发生错误的指令的下一行执行，重试执行 4 次若未解除错误，则输出错误代码。该指令必须添加至错误处理程序中才可生效。

➢ 语法格式

```
TryNext;
```

➢ 参数

【无参数】

➢ 示例

```
Func main1()
    num2=1;
    num1 = num2 - 1;
    num5 = num4 / num1;
    num2 = num2 + 1;
Errorfunc
    If ErrNo == 10604 Then
        num1 = 1;
        TryNext;
    Endif
```

Endfunc

//当 num1=0 时，将发生 10604 错误，随后进入错误处理程序，将从发生错误的指令行的下一行继续执行;若无错误，程序将继续正常运行。

11 料盘指令

料盘指令包括设置料盘和调用料盘点位，料盘仅支持四边形形状。在使用前应先设置料盘（包括示教点位、料盘行列数等），然后在运动指令中调用对应料盘坐标的点位。

11.1 设置料盘 SetPallet

➢ 指令说明

设置指定编号的料盘，可设置料盘的定位点位，行列数（分区数）。系统会根据示教的点位和行列数自动计算所有的点位坐标。

➢ 语法格式

SetPallet [料盘编号],[点位 1],[点位 2],[点位 3],[*点位 4*],[分区 1],[分区 2];

➢ 参数

【料盘编号】：料盘的 Id 标识，范围是 1~20。

【点位 1】：即 P1 点，指定料盘的原点。

【点位 2】：即 P2 点，指定料盘的横向方向和边界。

【点位 3】：即 P3 点，指定料盘的纵向方向和边界。

【*点位 4*】：即 P4 点，指定料盘的第 4 点方向，用于确定不规则四边形的边界。该点省略时，将根据点位 1~点位 3 自动生成的平行四边形区域计算点位。

【分区 1】：P1->P2 方向的分割数量，范围是 1~65535。

【分区 2】：P1->P3 方向的分割数量，范围是 1~65535。

➢ 注意

- ✧ 根据分区 1 与分区 2 的乘积为料盘点位数量，不能大于 100000。
- ✧ 示教的料盘的所有点位姿态与点位 1 姿态一致。
- ✧ 如图 11.1.1 所示，若仅示教 P1~P3 点，且分区 1 为 3，分区 2 为 4，则系统会自动计算出 12 个点位形成平行四边形的排列方式，点位的排列方向为先 P1->P2 方向，再 P1->P3 方向。执行 SetPallet 指令后，根据点位 1~点位 3 及分区 1、分区 2 确定生成的点位数量及具体的点位坐标，点位的排序自动生成且用户不可修改。
- ✧ 如图 11.1.2 所示，示教 P1 点作为原点（坐标 1，1），示教 P2 和 P3 确定横向和纵向的长度，且保证 P1~P4 点位的旋转角度一致，若不一致，则以 P1 的旋转角度为参考角度。根据分区 1 和分区 2 的值，确定横向列数和纵向行数。P4 用于不规则四边形的区域确定。

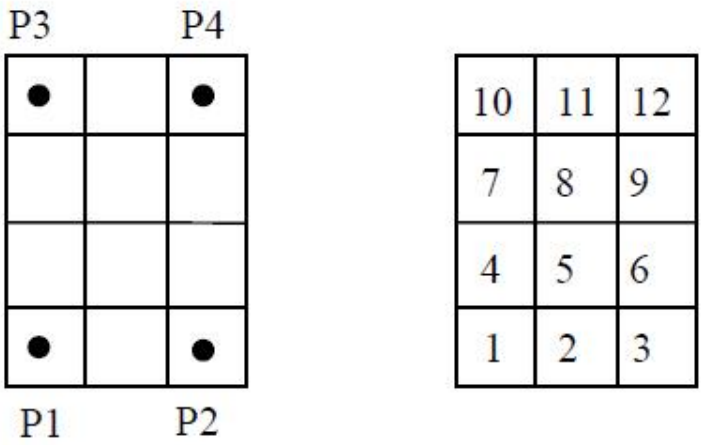


图 11.1.1 点位排序方式

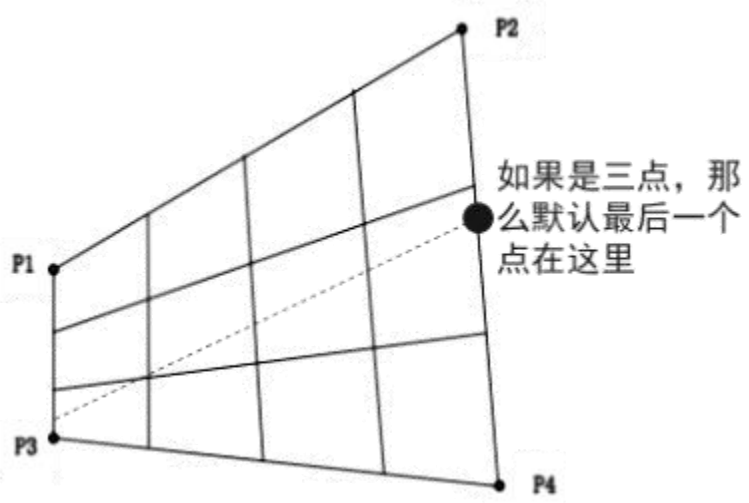


图 11.1.2 P4 点确定不规则四边形的边界

➢ 示例

```
SetPallet 1,P1,P2,P3,P4,3,3;
Jump Pallet(1,num1),V100,Zone0,Tool0,Arch0;
```

//设置料盘 1，并示教 4 个边界点位，分区数均为 3；执行 jump 运动到料盘 1 的 num1 变量指定序号的点位。

➢ 用法说明

料盘设置完成后，可通过运动指令以不同轨迹去往指定的料盘中的一个点位。

➢ 语法格式

```
Pallet([料盘编号],[点位序号]);
```

➢ 参数

【料盘编号】：料盘的 Id 标识，范围是 1~20。

【点位序号】：料盘中生成的点位的序号，范围是 1~100000。

➢ 注意

✧ 语法格式中所述内容为料盘点位的索引方式，与 Points 应用方式相同，可在所有运动指令中应用。但应注意在 MoveC 指令中应用时，需考虑起点、中间点及目标点的位置关系，以免发生碰撞。

➢ 示例

```
Func main1()
    SetPallet 1,P1,P2,P3,P4,3,4;
    For num1 From 1 To 12 Step 1 Do
        Jump P0,V100,Zone0,Tool0,Arch0;
        Jump Pallet(1,num1),V100,Zone0,Tool0,Arch0;
    Endfor
Endfunc
//在 P0（取料点）和料盘点位往复取料和放料。
```

12 焊接指令

焊接指令主要包括起弧指、轨迹焊接、收弧。

12.1 轨迹起弧 ArcLStart / ArcCStart

➢ 指令说明

以直线或圆弧运动轨迹到达焊接起点，然后根据设定的起弧参数起弧。

➢ 语法格式

ArcLStart/ArcCStart [焊接起点],[速度],[过渡],[工具],[起弧收弧参数],

[主焊接参数];

➢ 参数

【ArcLStart】：以直线运动轨迹到达焊接起点。

【ArcCStart】：以圆弧运动轨迹到达焊接起点。

【焊接起点】：当前要焊接的焊缝的起始点。

【速度】：去往焊接起点的轨迹运动速度，单位 mm/s。

【过渡】：此处过渡无效。

【工具】：当前使用的焊枪工具号。

【起弧收弧参数】：设置焊缝的起弧和收弧相关的数据，类型为 Seamdata。

【主焊接参数】：设置焊缝焊接过程相关的数据，包括焊接电压、电流以及焊接速度等，类型为 Welddata。

➢ 注意

✧ 圆弧由当前点、过渡点、目标点拟合而成，若其中两点或三点是同一位置，圆弧将退化为直线，此时机器人在运动过程中会按直线运动，且不会报警。现场应用中应避免圆弧中的点位相同，否则可能会出现指令和轨迹执行错误的情况。

➢ 示例

ArcLStart P10,V100,Zone0,Tool0,seam1,weld1;

//以直线轨迹运动到焊接起点，并执行起弧操作。

12.2 焊接 ArcL / ArcC

➢ 指令说明

起弧成功后，以直线或圆弧轨迹进行焊接作业。

➢ 语法格式

ArcL/ArcC [焊缝目标点],[速度],[过渡],[工具],[起弧收弧参数],

[主焊接参数],[摆弧参数 #];

➢ 参数

【ArcL】：以直线运动轨迹进行焊接作业。

【ArcC】：以圆弧运动轨迹进行焊接作业。

【焊缝目标点】：当前焊接的焊缝的结束位置。

【速度】：焊接过程中焊枪的移动速度（即焊接速度），单位 mm/s。

【过渡】：设置焊缝轨迹之间的过渡。

【工具】：当前使用的焊枪工具号。

【起弧收弧参数】：设置焊缝的起弧和收弧相关的数据，类型为 Seamdata。

【主焊接参数】：设置焊缝焊接过程相关的数据，包括焊接电压、电流以及焊接速度等，类型为 Welddata。

【摆弧参数 #】：设置摆弧运动有关的数据，类型为 Weavedata。

➢ 注意

✧ 圆弧由当前点、过渡点、目标点拟合而成，若其中两点或三点是同一位置，圆弧将退化为直线，此时机器人在运动过程中会按直线运动，且不会报警。现场应用中应避免圆弧中的点位相同，否则可能会出现指令和轨迹执行错误的情况。

➢ 示例

ArcLStart P10,V100,Zone0,Tool0,seam1,weld1;
ArcL P11,V100,Zone0,Tool0,seam1,weld1;
//以直线轨迹运动到 P10 点然后起弧，起弧成功后焊接 P10->P11 之间的直线焊缝。

12.3 轨迹收弧 ArcLEnd / ArcCEnd

➢ 指令说明

以直线或圆弧轨迹完成当前的焊接轨迹后进行收弧。

➢ 语法格式

ArcLEnd/ArcCEnd [焊接目标点],[速度],[过渡],[工具],[起弧收弧参数],
[主焊接参数],[摆弧参数 #];

➢ 参数

【ArcLEnd】：以直线运动轨迹完成当前焊接。
【ArcCEnd】：以圆弧运动轨迹完成当前焊接。
【焊接目标】：当前要焊接的焊缝的结束点。
【速度】：去往焊接目标点的焊枪运动速度（即焊接速度），单位 mm/s。

【过渡】：此处过渡无效。

【工具】：当前使用的焊枪工具号。

【起弧收弧参数】：设置焊缝的起弧和收弧相关的数据，类型为 Seamdata。

【主焊接参数】：设置焊缝焊接过程相关的数据，包括焊接电压、电流以及焊接速度等，类型为 Welddata。

【摆弧参数 #】：设置摆弧运动有关的数据，类型为 Weavedata。

➢ 注意

✧ 圆弧由当前点、过渡点、目标点拟合而成，若其中两点或三点是同一位置，圆弧将退化为直线，此时机器人在运动过程中会按直线运动，且不会报警。现场应用中应避免圆弧中的点位相同，否则可能会出现指令和轨迹执行错误的情况。

➢ 示例

```
ArcLStart P10,V100,Zone0,Tool0,seam1,weld1;
ArcL P11,V100,Zone0,Tool0,seam1,weld1;
ArcLEnd P12,V100,Zone0,Tool0,seam1,weld1;
//焊接完成后，在 P12 点处收弧。
```

附录一 系统关键字和系统变量

变量命名仅支持半角英文字母和数字组成，禁止使用除此之外的其他字符，同时避免与系统关键字重复，否则将发生错误。

1. 变量命名规则

（1）必须以字母或下划线开头。

（2）只能由字母、数字、下划线组成。

（3）不可使用中文或其他外文字符。

（4）不可与下述系统关键字或系统变量重复。

（5）不可与系统中的指令字符重复。

附表 1 系统关键字

类型	名称
Points	Here

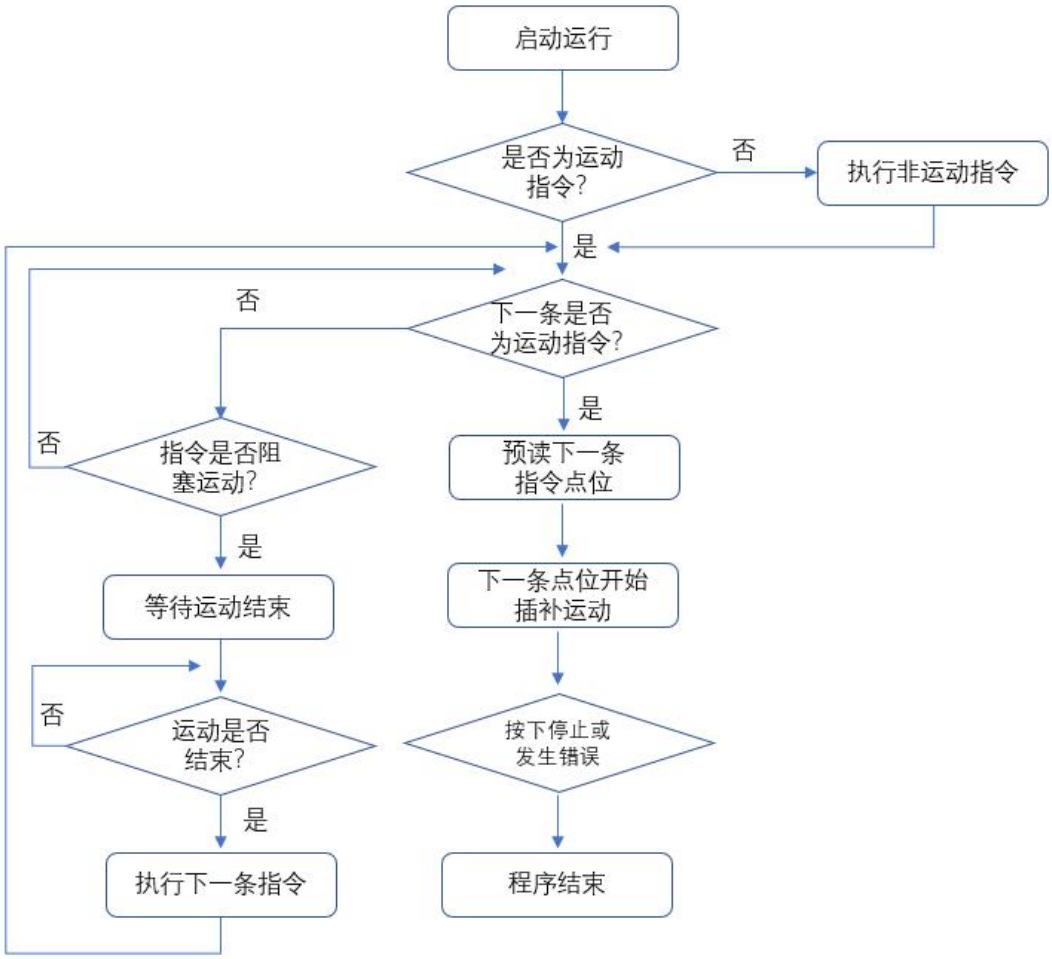
附表 2 系统变量

变量类型	变量名称
Arch	Arch0、Arch1、Arch2、Arch3、Arch4、Arch5
Bool	true、false、PosFound
DI	X0~X15(8 进制)
DO	Y0~Y17(8 进制)
Points	FindPos
Speed	V5、V10、V30、V50、V100、V200、V500、V800、V1000、V1500、V2000、V3000、V4000
Zone	Zone0、Zone10、Zone30、Zone50、Zone100
Int	ErrNo

附录二 程序执行逻辑

指令程序的执行采用运动指令预读取点位和非运动指令直接执行的方式，即当第一条运动指令开始执行时，立刻预读取下一行的指令点位，此时指针（箭头）会指向下一行的运动指令，红点将指向当前正在运动的指令。若下一行指令是非运动指令，且不会阻塞程序执行，则在运动指令执行过程中，会按程序逻辑顺序执行，直到发生阻塞的指令或下一条运动指令时，指针（箭头）停止向下执行。

程序执行逻辑如下：



附录三 应用案例

1.SCARA 物料搬运