
迈科讯 | 编程手册 |

机器人系统 (RA05)



深圳市迈科讯科技有限公司

文档版本

版本号	修订日期	修订内容
V1.0	2020.5.11	初订

目录

第一章	编程指令介绍	4
1.1	编程指令进入方式.....	4
1.2	指令行结构.....	4
第二章	编程指令附加项说明	5
第三章	编程指令详解	6

第一章 编程指令介绍

1.1 编程指令进入方式

1. 拨动钥匙开关选择正确的模式：示教（TEACH）模式。
2. 切换到程序列表界面，新建或打开已有程序：<文件管理>—<打开文件>。
3. 在程序界面，可通过点击主菜单的“指令编辑”以及子菜单中的“运动指令”调用常用功能指令。
4. 输入正确指令参数后，需要按住使能开关并点击“指令正确”来完成当前指令的输入，若指令输入有误，可点击“指令退出”取消当前指令的输入。
5. 编程中需要对当前指令进行更改时，可通过点击“改变指令”来修改（按住使能开关则记住当前位置，若不用改变位置不要按使能开关，直接修改即可）。

1.2 指令行结构



指令行包括指令和若干附加指令，根据指令的不同附加指令的个数及内容也有所差异。

第二章 编程指令附加项说明

附加项格式、定义	说明
VJ=<百分比> :关节运行速度	关节运行速度=参数中轴运动最高速度× VJ × 自动运行倍率。
VL=<直线运行速度> :直线运行速度	直线运行速度单位为 MM/S，其最大值为参数设定直线运动最高速度。实际运行速度=直线运动最高速度× VL× 自动运行倍率。
UNITL <条件> :条件判断	当满足相应条件时，停止当前运行的程序行，转到下一行程序执行；当不满足相应条件时，执行到该程序行直至结束，然后再转到下一行程序执行。
GP<变量号> :全局 P 变量	<ol style="list-style-type: none"> 1. 全局 P 变量记录了当前各关节姿态，坐标等相关位置数据，变量号范围：0-999。 2. 全局 P 变量在不同程序中调用时，是同一数据（关节姿态，坐标均相同）。同一变量号在再次赋值时会覆盖上一次的数据（关节姿态，坐标出现变化）。再次使用时可点击“查看当前变量”来确认大概位置，避免发生危险。 3. 该变量可在“运行准备” - “变量” - “全局 P 变量”表中查看。
LP<变量号> :局部 P 变量	<ol style="list-style-type: none"> 1. 局部 P 变量记录了当前各关节姿态，坐标等相关位置数据，变量号范围 0-999。 2. 局部 P 变量在不同程序中调用时，对应的数据不一样，各自独立。如一个程序调用变量 LP0。第二个程序也调用了 LP0，两个 LP0 互不干扰。该变量只有在调用程序打开时，才会在变量表中出现。 3. 该变量可在“运行准备” - “变量” - “全局 P 变量”表中查看。

第三章 编程指令详解

1. 运动指令

1 关节移动 (MOVJ)	指令功能描述	以关节插补方式移动到示教位置。各关节按照各自轴设定速度×VJ×自动倍率运动。	
	附加项说明	VJ=<百分比>	VJ 速度比例，百分比： 1%-100%。
	应用举例	MOVJ VJ=50%	

2 直线移动 (MOVL)	指令功能描述	以直线插补方式移动到示教位置。使用直线插补时，机器人手腕姿态不变。各关节按照 VL×自动倍率，以直线插补运动。	
	附加项说明	VL=<直线运行速度>	直线运行速度，单位 MM/S，最大值为参 数直线运动最高速度。
	应用举例	MOVJ VJ=50%	

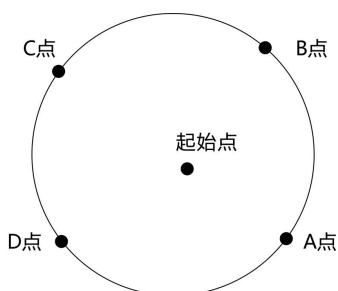
3 圆弧运动 (MOVC)	指令功能描述	以圆弧插补方式移动到示教位置。各关节按照 VL×自动倍率，以圆弧插补运动。整圆需要由两段圆弧构成，具体请参考整圆规划举例。	
	附加项说明	VL=<直线运行速度>	直线运行速度，单位 MM/S，最大值为参 数直线运动最高速度。
	变量 (POINT)	变量范围：1-3。一段圆弧轨迹通必须是由三段圆弧指令实现的，三段圆弧指令分别定义了圆弧的起始点、中间点、结束点。1P 表圆弧的起点，2P 表圆弧的中间点，3P 表圆弧的终点。	

整圆规划举例：

作出一个整圆，需要使用 A、B、C 和 C、D、A 两段圆弧，机器人末端位于起始点，下表为整圆程序举例：

MOVL VL=200MM/S PL=0	从起始点运行到 A 点，将 A 点作为圆弧第一点。
MOVC VL=200MM/S PL=0 2P	从 A 点运行到 B 点，将 B 点作为圆弧第二点。
MOVC VL=200MM/S PL=0 3P	从 B 点运行到 C 点，将 C 点作为圆弧第三点。同时将 C 点作为第二个圆弧的第一点。
MOVC VL=200MM/S PL=0 2P	从 C 点运行到 D 点，将 D 点作为圆弧的第二点。
MOVC VL=200MM/S PL=0 3P	从 D 点运行到 A 点，将 A 点作为圆弧的第三点。

整圆规划图示：



2. 逻辑指令

1 数字量输出 (DOUT)	指令功能描述	数字量输出的状态。数字量有两种形式，因此在使用该指令时有两种状态，即“ON (有效)”和“OFF (无效)”两种状态。		
	附加项说明	X<变量号>=	ON OFF	控制输出口 X<变量号>,有效 或无效状态。
	应用举例	DOUT X#(0)=ON 控制输出口 X0 为有效状态。		

2 条件等待 (WAIT)	指令功能描述	当满足设定的条件时，程序往下执行；当不满足设定的条件时，程序一直停在这里，直至设定的条件满足。其后有一个时间的设定，当不满足设定的条件时，在等待后面的设定时间之后，程序往下执行。		
	附加项说明	X<变量号>==	ON OFF	判断输入口 X<变量号>的状态是 ON 还是 OFF
	应用举例	WAIT X#(0)==ON T=0 持续等待 X0 口有效。		

3 延时指令 (TIME)	指令功能描述	在指定时间内等待或延时，时间结束时，继续执行程序。		
	附加项说明	T=<变量>	指定(延时)等待时间,变量单位 :1ms ,范围 : 1-10000ms	
	应用举例	TIME T=500 延时 500ms		

4 后台延时输出 (TDOUT)	指令功能描述	后台延时输出用于在后台延时输出状态，该指令在后台执行，不占用程序时间。		
	附加项说明	TDOUT	X<变量号> = ON/OFF	T=<变量>
	应用举例	DOUT X#(0)=ON : 输出 X#(0) 为有效 TDOUT X#(0)=OFF T=5000 : 后台延时 5000ms 输出 X#(0)为无效		

3. 焊接指令

1 起弧 (ARCSTART) 2 熄弧 (ARCEND)	指令功能描述	运行焊接指令时，程序需要调用预先设定好的焊接参数，起弧。变量号为焊接参数文件号，范围 0-99。 该两条指令成对使用（只输入单个指令会导致程序启动失败）。		
	附加项说明	#<变量号>	变量号为需要调用的焊接参数文件号。	
	应用举例	ARCSTART#(1) ARCEND#(1)	调用 1 号焊接参数文件，起弧开始焊接（运行相应的焊接运动） 起弧结束，焊接完成	

3 摆动开始 (WEAVESINE) 4 摆动结束 (WEAVEEND)	指令功能描述	运行摆动焊接指令时，程序需要调用预先设定好的摆弧参数，摆弧。变量号为摆动焊接参数文件号，范围 0-99。该两条指令成对使用（只输入单个指令会导致程序启动失败）。	
	附加项说明	#<变量号>	变量号为配对使用摆弧指令的焊接参数文件号。
	应用举例	WEAVESINE#(1) WEAVEEND#(1)	调用 1 号摆弧参数文件，摆弧 摆弧路径 摆弧结束

焊接摆弧应用实例：

ARCSTART #(1)	调用 1 号焊接参数进行焊接
WEAVESINE #(1)	调用 1 号摆动参数
MOVL VL=100MM/S	走焊接轨迹
WEAVEEND	摆动结束
ARCEND #(1)	1 号焊接工艺结束

5 鱼鳞焊开始 (STITCHSTART) 6 鱼鳞焊结束 (STITCHEND)	指令功能描述	STITCHSTART 鱼鳞纹焊接开始，与起弧指令 ARCSTART 配合使用。 STITCHEND 鱼鳞纹焊接结束，与收弧指令 ARCEND 配合使用。	
	附加项说明	T<变量>ms	点焊时间，单位 ms
		L1<变量>mm	焊接距离，单位 mm
L2<变量>mm		空走距离，单位 mm	
应用举例	ARCSTART#(0)；打开焊接工艺(0) STITCHSTART L1=200mm L2=20mm；开始鱼鳞焊接，点焊 200mm,空走 20mm，再点焊 200mm MOVL VL=50MM/S PL=0；焊接轨迹，鱼鳞焊功能接入，空走的速度为 50MM/S,焊接轨迹是直线 STITCHEND；鱼鳞焊接结束 ARCEND#(0)；焊接工艺(0)结束		

4.辅助指令

1 条件判断 (IF → IF)	指令功能描述	<p>IF 条件判断指令由 IF、ELSEIF (可省略或重复使用)、ELSE、ENDIF 构成一个完整结构。</p> <p>首先判断 IF 条件是否成立, 成立则执行 IF 之后的语句。如果 IF 条件不成立, 则判断 ELSEIF 之后条件(ELSEIF 可根据实际选择使用或不使用), ELSEIF 条件成立, 则执行 ELSEIF 之后程序。如果 ELSEIF 条件不成立, 则执行 ELSE 之后程序。程序结尾使用 ENDIF。</p>	
	附加项说明	<p>IF 条件判断, 条件成立, 则执行该语句之后程序。</p> <p>如果 IF 条件不成立, 执行 ELSEIF 条件判断, 如果 ELSEIF 之后条件成立, 则执行 ELSEIF 之后程序。</p> <p>如果 IF 和 ELSEIF 之后条件都不成立, 则执行 ELSE 之后程序。</p>	
	应用举例	<pre>IFX#(0)==ON 0 ELSEIF 0</pre>	<p>如果 X0=有效。执行.....</p> <p>如果 X0=无效, 执行.....</p>

2 条件判断 (IF → ELSEIF)	指令功能描述	<p>IF 条件判断指令由 IF、ELSEIF (可省略或重复使用)、ELSE、ENDIF 构成一个完整结构。</p> <p>首先判断 IF 条件是否成立, 成立则执行 IF 之后的语句。如果 IF 条件不成立, 则判断 ELSEIF 之后条件(ELSEIF 可根据实际选择使用或不使用), ELSEIF 条件成立, 则执行 ELSEIF 之后程序。如果 ELSEIF 条件不成立, 则执行 ELSE 之后程序。程序结尾使用 ENDIF。</p>	
	应用举例	<pre>IF X#(0)==ON 0 ELSEIF X#(1)==ON 0 ELSEIF X#(2)==ON 0 ELSE 0 ENDIF 0</pre>	<p>如果 X0=有效。执行.....</p> <p>如果 X1=有效, 执行.....</p> <p>如果 X2=有效, 执行.....</p> <p>否则, 执行.....</p> <p>0 号 IF 指令结构结束</p>
	附加项说明	<p>IF 条件判断, 条件成立, 执行该语句之后程序。</p>	

		<p>如果 IF 条件不成立，执行 ELSEIF 条件判断，如果 ELSEIF 之后条件成立，则执行 ELSEIF 之后程序。如果 IF 和 ELSEIF 之后条件都不成立，则执行 ELSE 之后程序。ENDIF 结束 IF 判断。</p>
--	--	--

<p>3 条件判断 (IF→ELSE)</p>	<p>指令功能描述</p>	<p>IF 条件判断指令由 IF、ELSEIF (可省略或重复使用)、ELSE、ENDIF 构成一个完整结构。</p> <p>首先判断 IF 条件是否成立，成立则执行 IF 之后的语句。如果 IF 条件不成立，则判断 ELSEIF 之后条件(ELSEIF 可根据实际选择使用或不使用)，ELSEIF 条件成立，则执行 ELSEIF 之后程序。如果 ELSEIF 条件不成立，则执行 ELSE 之后程序。程序结尾使用 ENDIF。</p>	
	<p>附加项说明</p>	<p><编号></p>	<p>指令标号，一个完整的指令结构内该标号必须一致，否则程序将报错或跳出。</p> <p>一个完整指令结构所使用的编号和另一个完整指令结构编号可以相同。</p> <p>标号范围：0 - 8。</p>
	<p>应用举例</p>	<pre>IF X#(1)==ON 0 ELSE 0 END IF 0 IF X#(1)==OFF 1 ELSE 0 END IF 1</pre>	<p>如果 X1=有效。执行.....</p> <p>否则执行</p> <p>0 号 IF 指令结构结束</p> <p>如果 X1=无效，</p> <p>执行</p> <p>否则执行</p> <p>1 号 IF 指令结构结束</p>

<p>4 条件判断 (IF→ENDIF)</p>	<p>指令功能描述</p>	<p>IF 条件判断指令由 IF、ELSEIF (可省略或重复使用)、ELSE、ENDIF 构成一个完整结构。</p> <p>首先判断 IF 条件是否成立，成立则执行 IF 之后的语句。如果 IF 条件不成立，则判断 ELSEIF 之后条件(ELSEIF 可根据实际选择使用或不使用)，ELSEIF 条件成立，则执行 ELSEIF 之后程序。如果 ELSEIF 条件不成立，则执行 ELSE 之后程序。程序结尾使用 ENDIF。</p>	
---------------------------------	---------------	--	--

	<p>附加项说明</p>	<p>一个单独的指令结构内指令标号必须一致,否则程序将报错或跳出。一个单独的指令结构所使用的编号和另一个单独的指令结构编号可以相同。 标号范围:0-8。</p>	
	<p>应用举例</p>	<p>IF <条件> <编号> ELSEIF <条件> <编号> ELSE<编号> END IF <编号></p>	<p>IF 条件 IF 条件成立后执行程序 ESLEIF 条件 ELSEIF 条件成立执行程序 上面条件都不成立执行下面程序 ENDIF 结束 IF 结构</p>

<p>5 跳转指令 标号(*)</p>	<p>指令功能描述</p>	<p>标记 JUMP 跳转位置,需和 JUMP 指令配合使用(单独使用可能导致程序错误),当程序条件满足时,跳转到指定位置。</p>	
	<p>附加项说明</p>	<p>*<变量></p>	<p>变量可以为任意字符或数字。</p>
	<p>应用举例</p>	<p>* aab1 : 标记跳转位置: *aab1</p>	

<p>6 跳转指令 JUMP</p>	<p>指令功能描述</p>	<p>1、使用跳转指令时,要配合使用标号指令(*)。标号位置为程序跳转到的位置。 2、跳转指令后无条件,程序执行到此行,则直接跳到标号所处的位置。 3、跳转指令后有条件,程序执行到该行指令时,需要等到后面的条件满足时,程序才跳转到标号所处的位置。条件不满足时,程序顺序执行。</p>	
	<p>附加项说明</p>	<p>*<变量></p>	<p>跳转标号、标记,表示跳转到具有*<变量>的位置;变量可以为任意字符、数字。</p>
	<p>应用举例</p>	<p>第一行 *1M : 标号*1M 位置 第五行 JUMP *ABC IF M#(1)==ON : 如果 M1=ON, 跳转到*ABC 位置。 第十行 JUMP *1M : 无条件跳转*1M 位置。 第十一行 *ABC : 标号*ABC 位置 : 后续程序</p>	

7 子程序调用 (CALL)	指令功能描述	1、建立子程序需要在子程序的末尾加上 RET 指令。 2、%表示所需调用的程序，程序执行到此行时，直接调用该子程序。 3、当后面有条件时，程序执行到此行，需要满足后面的条件时，程序才调用该子程序。	
	附加项说明	%<程序名>	程序名为调用的子程序名称。
	应用举例	CALL %HANJIE :调用程序名为 HANJIE 的子程序 CALL %HANJIE IF LD#(1) > LD#(2) :如果 LD1>LD2 , 则调用 HANJIE 子程序。	

8 子程序调用返回 (RET)	指令功能描述	子程序调用返回用于子程序的返回。返回主程序的界面，在 CALL 程序行后继续运行主程序	
	附加项说明	无	
	应用举例	RET：子程序返回	