

机器人控制系统编程手册

前言

机器人编程为使机器人完成某种任务而设置的动作顺序描述。示教是机器人编程的一种重要方式，通过预先设置好机器人要达到的位置，以指令描述出来。本手册旨在帮助读者学习和掌握汇川的机器人示教软件 InoTeachPad 的编程方法。

编程手册版本	发布时间	示教器版本	控制器版本
V8.53	2017.07.07	S01.15T01B	S01.15T01B

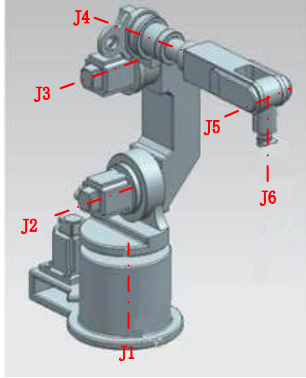
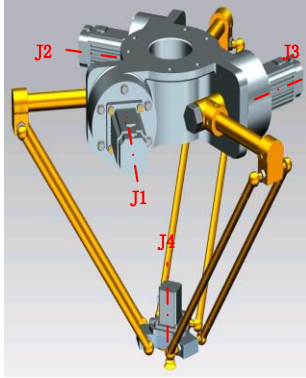

功能优化（相对 S01.15T01A）		索引
PC 版本示教器外框架优化	采用全新的外框架	~
示教盒版本键盘优化	采用全新的键盘	~
锁螺丝功能上线	功能：工程管理、工艺配置、编程指令、锁付监测	4.6 节
信号指令预处理功能	Set\Get 系列的指令，默认会预处理时就执行，表现为在光标执行到该行前就执行。在 Set\Get 前使用 WaitInPos 指令，才为到位执行。	~
指令变更		
信号处理指令变更	Set 系列负责输出信号；Get 系列负责输入信号。 不再根据电流与电压区分信号，统一使用 AD 代表模拟量输入信号，DA 代表模拟量输出信号。	2.5.1、2.5.2 节

1 基本概念

1.1 机型

根据轴数、串联/并联这些特性，可对机器人分类。如下表列出了几种常见的机器人：

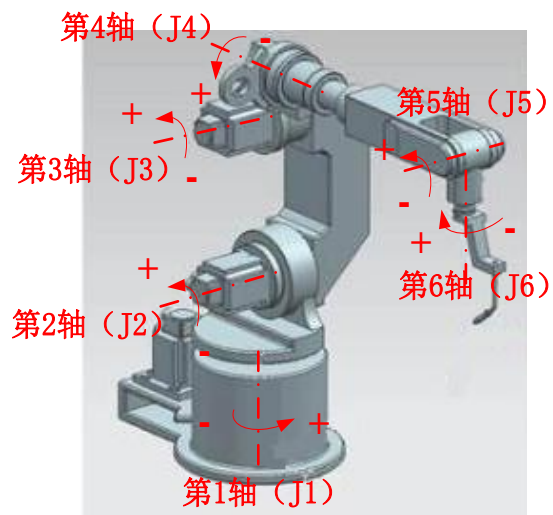
名称	串联 6 轴机器人	Delta 机器人	Scara 机器人
----	-----------	-----------	-----------

图片			
轴数	6	4	4
串/并联	串联机器人	并联机器人	串联机器人
特点	灵活性极高，几乎适合于任何轨迹或角度的工作	精度高	结构轻便、响应快
应用场合	应用范围极广，有装货、喷漆、测量、弧焊、点焊、包装、装配、锻造、铸造等	医药食品的搬运、分拣等	装配、运输

1.2 坐标系

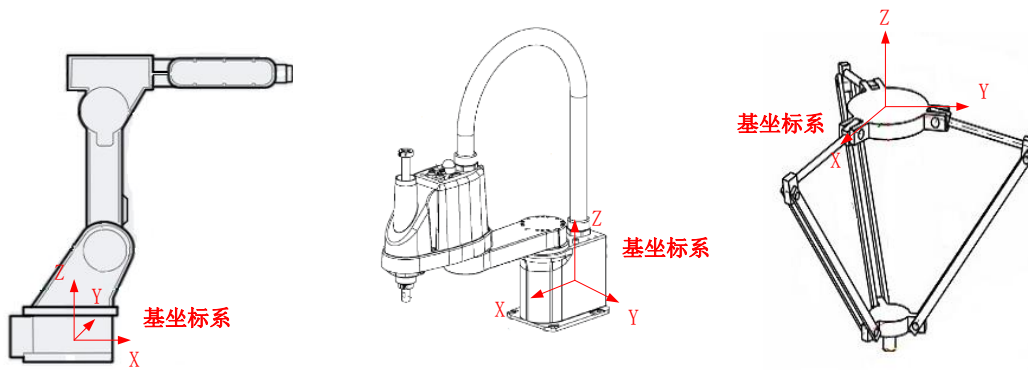
1.2.1 关节坐标系

关节坐标系存在于机器人各关节处。

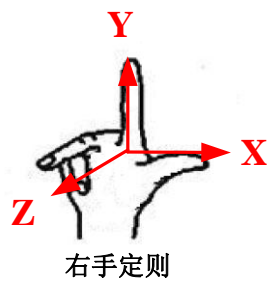


1.2.2 基坐标系

基坐标系也称机器人坐标系，一般位于机器人根部。

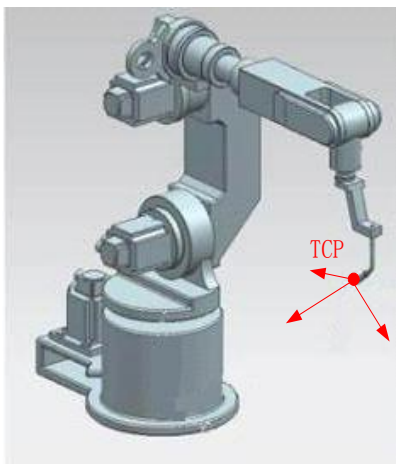


基坐标系是笛卡尔类型的坐标系（工具坐标系、用户坐标系也是）。它们都符合右手定则，即可先确定 X 和 Z 方向，再由如下手势确定 Y 方向。

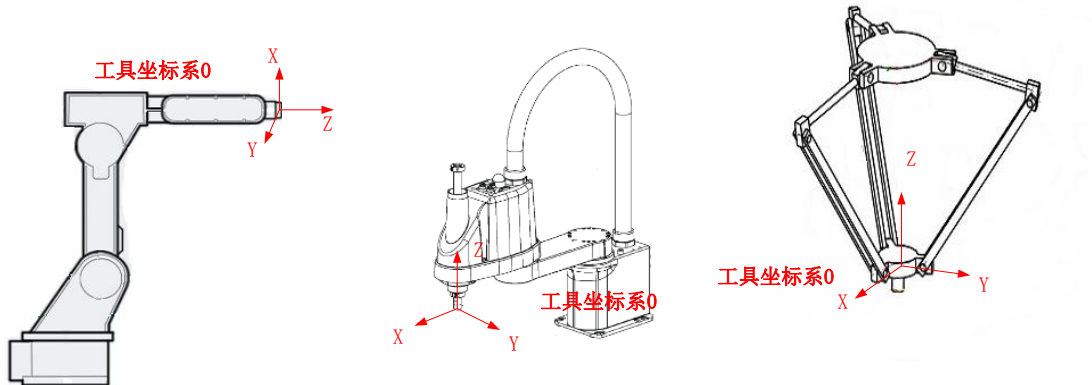


1.2.3 工具坐标系

工具坐标系附着于工具上，TCP（Tool Center Point，工具坐标原点）作为衡量机器人到达位置的参考点。一般取工具末端点作为 TCP，方向可自由定义。



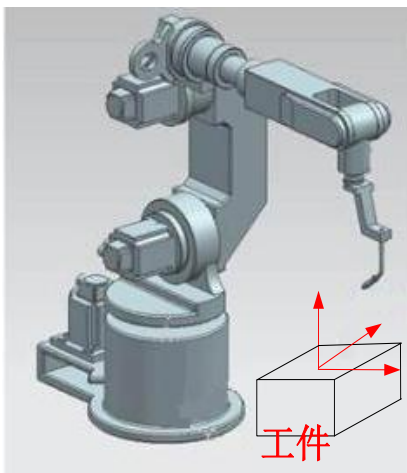
在汇川机器人控制系统中，最多可定义 16 个工具坐标系。其中工具 0 表示不使用工具，此时工具坐标系位于机器人本体末端。



工具 1~15 可由用户自定义。
工具坐标系设置方法详见 3.2.3 a 工具坐标系设置。

1.2.4 用户坐标系

用户坐标系是用户自定义的坐标系，一般定义于工件上。



在汇川机器人控制系统中，最多可定义 16 个用户坐标系。其中用户 0 表示不使用额外的用户坐标系，此时用户坐标系与基坐标系重合；用户 1~15 可由用户自定义。

用户坐标系设置详见 3.2.3 b 用户坐标系设置。

1.3 位置变量

汇川机器人控制系统中，点用“位置变量”表达，它存储了坐标值、臂参数、坐标系、工具号、用户号这些信息。

1.3.1 位置变量的存储格式

位置变量的存储格式如下：

P[0] = 26.969294, -107.023346, 0.000000, 13.795120, 12.265454, 3.645454; -1, 0, 0, 0; 1, 0, 0;

①

②

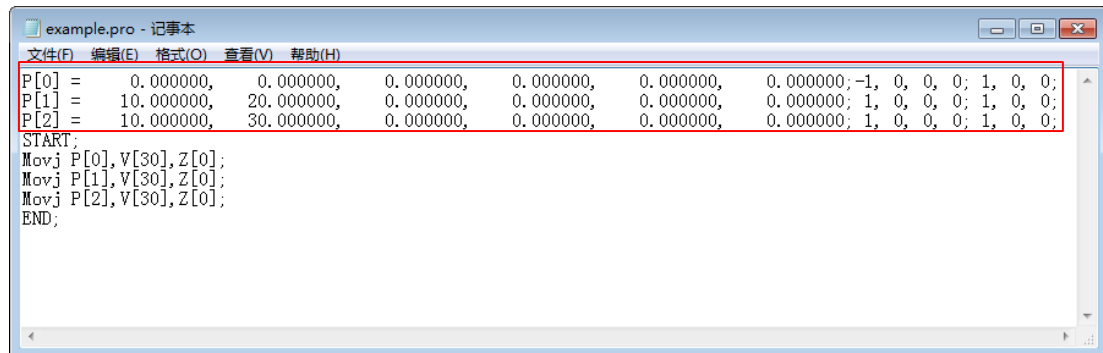
③

①记录的是位置变量的坐标数据，数据的个数恒为6个（多余的轴数值为0），以英文“,” 隔开，以“;” 结束。

②记录的是位置变量的臂参数，以英文“,” 隔开，以“;” 结束。

③分别记录的是位置变量的坐标系号+工具号+用户号，以英文“,” 隔开，以“;” 结束。每一行存储一个位置变量。

位置变量通常放在机器人程序 (*.pro) 中，并位于在文件的开头，START;指令之前。



1.3.1 坐标系与坐标值

坐标系指机器人取点时所用的坐标系。用坐标系号 1~4 表示不同的坐标系。不同坐标系，坐标值的涵义不同。

在关节坐标系下，坐标值取关节值 (J1,J2,J3,J4,J5,J6)。在基坐标系、工具坐标系、用户坐标系，坐标值用“平动(X,Y,Z)+转动(A,B,C)”的形式表达。A,B,C 分别表示绕 Z,Y,X 的旋转。

<p>关节坐标系下取点</p>	<p>坐标系号为 1, 坐标值 (J1,J2,J3,J4,J5,J6) 表示机器人各关节位置 (当前关节值相对于关节零点的位置)。</p>	
-----------------	---	--

<p>基坐标系下取点</p>	<p>坐标系号为 2，坐标值 (X,Y,Z,A,B,C) 表示机器人本体末端点相对于基坐标系的位姿。</p>	
<p>工具坐标系下取点</p>	<p>坐标系号为 3，坐标值 (X,Y,Z,A,B,C) 表示 TCP 在基坐标系下的位姿。</p>	
<p>用户坐标系下取点</p>	<p>坐标系号为 4，坐标值 (X,Y,Z,A,B,C) 表示工具在用户坐标系下的位姿。</p>	

1.3.2 工具号与用户号

工具号：指定机器人所使用的工具。

用户号：指定所选用的用户坐标系。

在汇川机器人系统中，最多可定义 16 个工具坐标系和 16 个用户坐标系。工具 0 是系统默认的，表示不采用工具，TCP 为机器人本体末端；工具 1~15 为用户定义的工具。用户号 0 是系统默认，表示不采用用户坐标系，此时用户坐标系与基坐标系重合，用户 1~15 为用户自定义的坐标系。

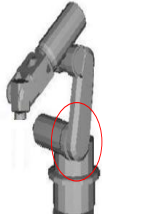
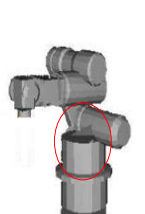
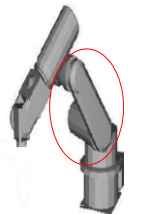

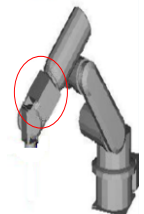
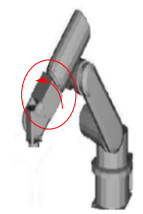
示例：

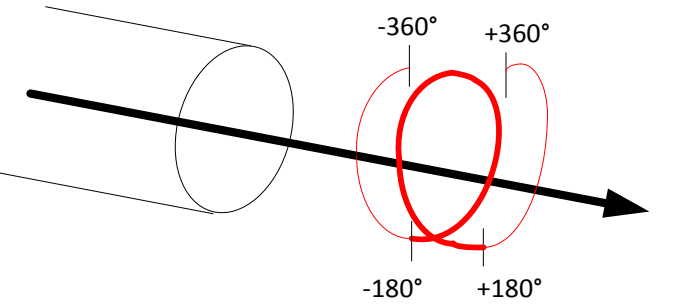
变量名	J1/X	J2/Y	J3/Z	J4/A	J5/B	J6/C	坐标系	工具号	用户号
P[000]	0.000	0.000	0.000	0.000	-90.000	0.000	1	0	0
P[001]	0.000	0.000	0.000	0.000	-90.000	0.000	1	2	3
P[002]	241.000	0.000	175.000	0.000	0.000	0.000	2	2	3
P[003]	241.000	0.000	165.000	0.000	0.000	0.000	3	2	3
P[004]	141.000	0.000	165.000	0.000	0.000	0.000	4	2	3

位置变量 P[0]表示在关节坐标系下取的点，当时所采用的工具号为 0，用户号为 0。
 位置变量 P[1]表示在关节坐标系下取的点，当时所采用的工具号为 2，用户号为 3。
 位置变量 P[2]表示在基坐标系下取的点，当时所采用的工具号为 2，用户号为 3。
 位置变量 P[3]表示在工具坐标系下取的点，当时所采用的工具号为 2，用户号为 3。
 位置变量 P[4]表示在用户坐标系下取的点，当时所采用的工具号为 2，用户号为 3。

1.3.3 臂参数

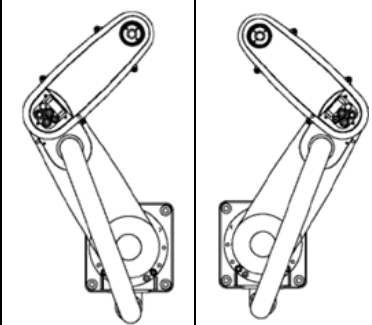
机器人控制目标点到达空间同一位姿时，机器人可能存在几种不同的手臂姿势。
 对于串联 6 轴机器人，在腰关节、肘关节、腕关节、第六轴处各有一个臂参数。

臂参数 1		臂参数 2		臂参数 3	
-1	1	-1	1	-1	1
腰部向前	腰部向后	肘部向上	肘部向下	手腕不翻转	手腕翻转
					

臂参数 4		
-1	0	1
轴 6 (-360°~-180°)	轴 6 (-180°~180°)	轴 6 (180°~360°)
		

对于 Scara 机器人，臂参数 1,3,4 有意义。

臂参数 1		臂参数 4(关节坐标 J4 的值)		
-1	1	(-540~-180)	(-180~180)	(180,540)
左手臂	右手臂	-1	0	1

	当涉及到多圈的时候，每增加 360 度臂参数加 1，每减少 360 度减 1.
---	---

对于，Delta 机器人臂参数 4 有意义。

Delta 臂参数 4(关节坐标 J4 的值)		
(-540~-180)	(-180~180)	(180,540)
-1	0	1
当涉及到多圈的时候，每增加 360 度臂参数加 1，每减少 360 度减 1.		

注意：示教后修改臂参数，将会使机器人以另外一种臂姿势到达空间同一点，运动变化很大，这很可能造成危险，需要慎重！

1.3.4 特殊工艺的位置变量

针对某些特殊的工艺，定义了更多的位置变量类型，如下：

固定相机视野内的位置点	坐标系号为 5，位置点在相机视野平面内坐标值 (X,Y,θ)，对应存储形式为 (X,Y,0,A,0,0)。工具号是所使用的工具，用户号是所使用的视觉坐标系编号	视觉功能专用
移动相机视野内的位置点	坐标系号为 6，位置点在相机视野平面内坐标值 (X,Y,θ)，对应存储形式为 (X,Y,0,A,0,0)。工具号是所使用的工具，用户号是所使用的视觉坐标系编号	视觉功能专用
跟随工艺	坐标系号为 7，坐标值 (X,Y,Z,A,B,C) 用来指定跟随传送带物体的同步运动，代表相对传送带上物体的坐标位置。	跟随工艺专用

1.4 平移变量

平移变量用于描述基于某个位置变量，以何种方式进行平移向量。

关节平移	<p>典型使用形式： <code>Movj OffsetJ(P[1],PR1),V[30],Z[0];</code> 平移变量的坐标值 (J1,J2,J3,J4,J5,J6) 表示机器人各关节的旋转度数。</p>	
------	--	--

<p>本体末端 沿基坐标 系平移</p>	<p>典型使用形式： Movj Offset (P[1],PR1) ,V[30], Z[0]; 平移变量的坐标值 (X,Y,Z,A,B,C) 表示机 器人本体末端在基坐标系下的偏移量。</p>	
<p>TCP 沿基坐标 系平移</p>	<p>典型使用形式： Movj Offset (P[1],PR1) ,V[30], Z[0],Tool[1]; 平移变量的坐标值 (X,Y,Z,A,B,C) 表示 TCP 在基坐标系下的偏移量。</p>	
<p>TCP 沿用 户坐标系 平移</p>	<p>典型使用形式： Movj Offset (P[1],PR1) ,V[30], Z[0],Tool[1],User[2]; 平移变量的坐标值(X,Y,Z,A,B,C)表示 TCP 在用户坐标系下的偏移量。</p>	
<p>TCP 沿 工具坐标 系自身的 平移</p>	<p>典型使用形式： Movj OffsetT(P[1],PR1) ,V[30], Z[0]; 平移变量的坐标值(X,Y,Z,A,B,C)表示 TCP 绕工具自身坐标系的偏移量。</p>	

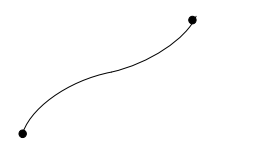
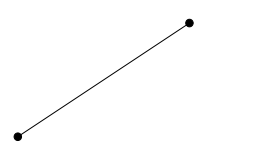
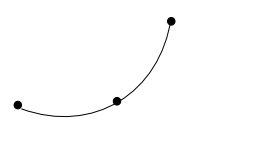
指令中的关于平移变量使用：

- 计算两点间平移变量指令——**Msft**：平移变量的计算统一采用以第一个点为基准点，即平移变量的坐标系与第一个点相同，第二个点的数据会转换成第一个点坐标系下的值，然后进行平移变量的计算。详见 2.5.1 节 **Msft** 指令。
- 平移变量直接赋值指令——**Pr***=(X,Y,Z,A,B,C)**：详见 2.5.2 节 **Pr***** 指令。

- 平移变量的加减指令——Pr Sum：详见 2.5.3 节 Pr Sum 指令。
- 基于某点的平移 Offset(P,PR)/OffsetI(P,PR)/OffsetT(P,PR)：详见 2.3.1 节 Movj 中的 Offset 使用。

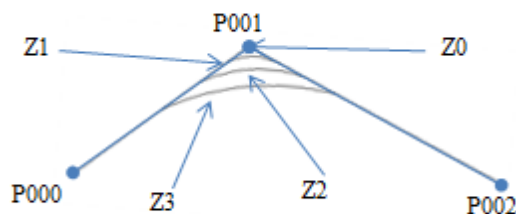
1.4 插补与插补精度

插补是机器人的基本运动形式，复杂的运动指令其实是由一系列插补运动组成的。根据轨迹的不同，分为以下三种形式。

插补类型	轨迹	特点
关节插补(Movj)		点到点的插补，各关节以最快速度运转，是速度最快的插补。运动轨迹不可预见，常用于点焊、运输等场合。
直线插补(Movl)		轨迹成直线，常用于轨迹焊接、贴装等场合
圆弧插补(Movc)		轨迹成圆弧状

注意：在执行 Movl、Movc 时，不允许机器人的臂参数变化。若需要臂参数变化，可插入 Movj 指令完成手臂姿势过渡。

在实际的连续运动过程中，很多时候运动并不是逐点精确到位，而是圆滑过渡的，因为这样不必频繁启停，可以缩短节拍时间。这些运动的中间点就会表现出轨迹逼近的形式。轨迹逼近的程度称为插补精度。插补精度分为几个等级，如下图 Z0、Z1、Z2.....



插补精度

一般的，运动指令包含过渡精度参数；但遇到触发预进*停止的指令时，过渡精度将失效，运动将会精确到位。

预进*：详见 1.7 节预进。

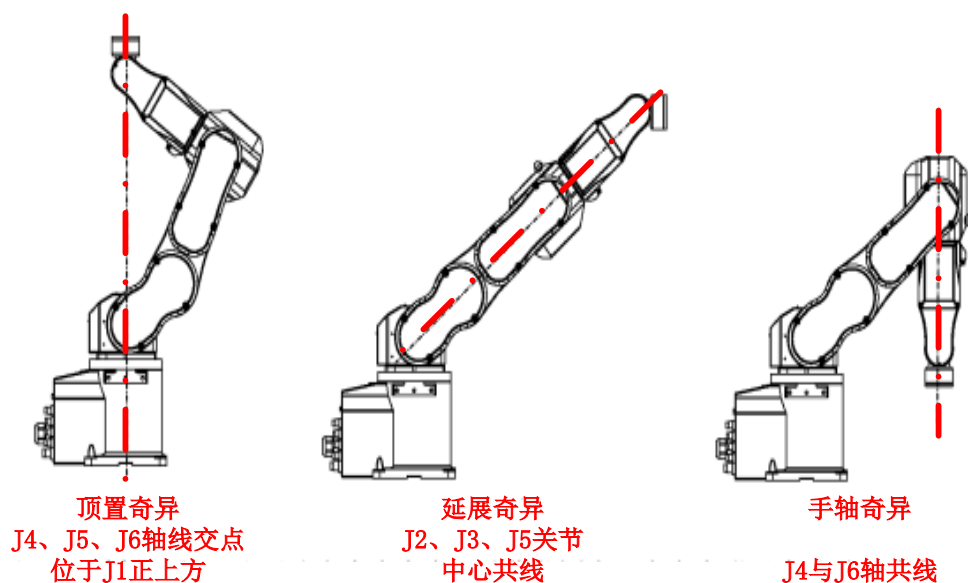
1.5 奇异位置

在非关节坐标系下运动时，机器人可能会运动到某些特殊位置，此时机器人会失去一些运动自由，这些特殊的位置称为奇异位置。

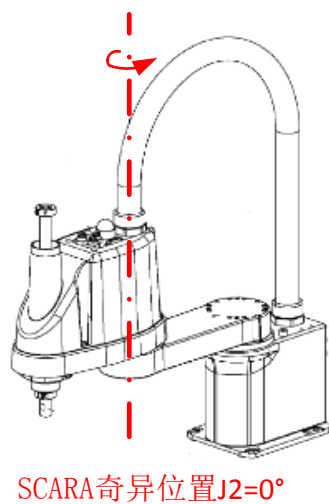
在关节插补 Movj 中，奇异位置并不会影响正常运动。而在直线插补 Movl、圆弧插补 Movc 过程中，奇异位置会使得机器人不能正常运行。

注意：遇到奇异位置报警时，可利用关节运动模式退出奇异位置。

串联 6 轴机器人存在三种奇异位置，如下图。



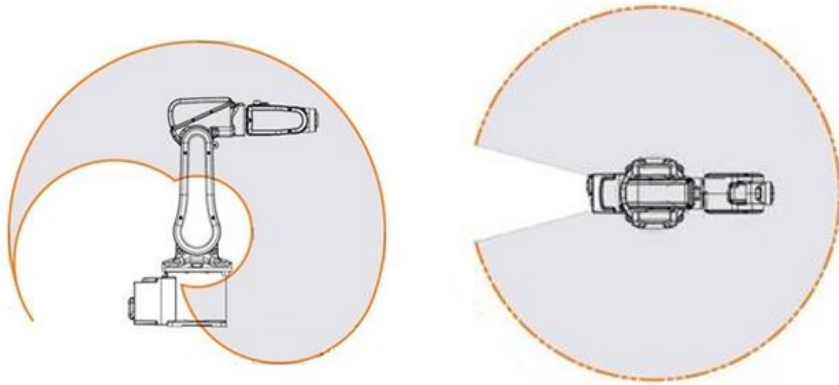
Scara 机器人只存在一个奇异位置，处于 $J2=0^\circ$ 时，此时第 1、2 臂摆成一条直线。



Delta 机器人的奇异位置不在工作范围内，无奇异位置。

1.6 工作范围与干涉区域

机器人的工作范围指机器人手臂末端所能到达的所有点的集合。工作范围与机器人臂长和关节运行范围有关。



干涉区域：在工作范围内，存在一些末端执行器禁止到达的区域，处于这些区域时，机器人会与自身部件或外部设备发生碰撞，这些区域称为干涉区域，或者用户指定希望机器人不可达到的区域，也称之为干涉区域，用户可以在【设置】-》【运动参数设置】-》【干涉区设置】中去定义。

1.7 程序的预进

在连续示教或运行过程中，会存在程序的预进。预进是指程序在执行到该行前就开始预处理数据，以便机器人控制器在运行前进行轨迹设计。

001	START;	
002	Movj P[0],V[30],Z[3];	当前显示运行行 (程序行光标)
003	Movl P[1],V[30],Z[3];	
004	Set Out[0],ON;	
005	Movl P[2],V[30],Z[3];	
006	Movj P[3],V[30],Z[3];	可能的预进到的指令 (不可见)
007	END;	

预进过程会使得监控中显示数据提前变化，显得数据好像超前了一样。

比如以连续运行模式下运行图所示程序。当程序第一次执行至第 4 行时监控中就已经显示 B0=2，等到第三次执行至第 4 行时监控中显示 B0=8，整个 B0 值显示好似超前了一样。

001	START;
002	B0 =0;
003	While B0<=2
004	Movj P[0],V[30],Z[0];
005	Movj P[1],V[30],Z[0];
006	Incr B0;
007	EndWhile;
008	B0 =8;
009	END;

注意：以单步示教运行，则不会有预进，能准确监测的变量数值。

在某些场合，如果预进一直进行，可能又会造成一些错误，比如使用 If 判断时，预进读到的判断内容可能不正确，因而影响结果。因此规定了一些特殊的指令，当预进到他们时，会触发预进停止，同时等待执行到自身这行时再执行。这些指令包括：USING MAIN、TimeOut、TimeStart、Ret、Call、If、Print、While、Wait、Movj until、Movl untill、MovC untill。

2.指令

2.1 程序文件

程序文件后缀名为“.pro”，只能由字母、数字以及下划线组成，且首位必须为字母，长度不得超过 32 个字符。仅仅大小写不同的文件会被认为是同名文件，因此会禁止使用大小写名称不同的文件。程序最多容纳 2000 行指令。所有程序都是以 START 开始，END 结束，中间根据需求编写指令行，每个指令段都以“;”结尾。大部分指令与 C 语言书写习惯相似，如运算指令、流程控制指令等，简单易懂。在程序文件中，点数据文件存储在程序指令 START; 之前,点数据存储格式详见 1.3.5 位置变量的存储格式。文件中不支持中文的标点符号，字符串变量内容除外。

2.2 变量

2.2.1 全局数值变量

全局数值变量包含 B、R、D 三种，它们一经定义，便存储于控制器中，作用域可超出当前的程序段。

B <0~255>间的任意整数。

R <-65536~65535>间任意整数。

D <-9999999.999~9999999.999>间浮点数（最多包含三位小数）。

表示方法：B^{***}/R^{***}/D^{***}（***为变量编号，从 0 最多取到 255）

范例：

收藏	变量名	数值	收藏	变量名	数值	收藏	变量名	数值
<input type="checkbox"/>	B000	8	<input type="checkbox"/>	R000	0	<input type="checkbox"/>	D000	0.000
<input type="checkbox"/>	B001	12	<input type="checkbox"/>	R001	2568	<input type="checkbox"/>	D001	2.310
<input type="checkbox"/>	B002	17	<input type="checkbox"/>	R002	6666	<input type="checkbox"/>	D002	9999.992

注意事项：在监控界面中修改 B、R、D 值会自动限定范围。在程序执行中，当变量的取值超出范围，程序报警。对于 B、R 变量，其值为整数，若运算中赋值为小数，会只保留整数部分；如在程序中执行“B1=2.8”，运算结果为 B1=2。

2.2.2 局部数值变量

对应的，存在 LB、LR、LD 三种局部数值变量，它们的作用域仅限本段程序。

LB <0~255>间的任意整数。

LR <-65536~65535>间任意整数。

LD <-9999999.999~9999999.999>间浮点数（最多包含三位小数）。

表示方法及注意事项同上。

范例：

变量名	数值	变量名	数值	变量名	数值
LB000	0	LR000	0	LD000	0.000
LB001	1	LR001	123	LD001	2.360
LB002	254	LR002	-65534	LD002	98712.340

2.2.3 位置变量

位置变量用（坐标值）+（臂参数）+（坐标系）+（工具号）+（用户号）的方式表达。在监控界面中，臂参数是隐藏的，可通过双击列表查看。

变量表示方法：P^{***}

***为变量编号，取值范围<0~9999>

范例：

变量名	J1/X	J2/Y	J3/Z	J4/A	J5/B	J6/C	坐标系	工具号	用户号
P[000]	13.227	13.730	-168.882	-2.542	0.000	0.000	1	0	0
P[001]	13.227	13.730	-168.882	-2.542	0.000	0.000	1	0	0
P[002]	375.016	130.816	-7.506	24.415	0.000	0.000	2	0	0
P[003]	373.128	164.004	-7.506	24.415	0.000	0.000	3	1	0
P[004]	323.128	114.005	-7.506	24.415	0.000	0.000	4	1	2

位置变量 P[0]表示在关节坐标系下取的点，当时所采用的工具号为 0，用户号为 0。

位置变量 P[1]表示在关节坐标系下取的点，当时所采用的工具号为 1，用户号为 2。双击 P[1]点所在的行后，即可以查看其所带的工具号以及用户号，如下图所示：

P[001]	13.227	13.730	-168.882	-2.542	0.000	0.000	1	0	0
P[002]	375.016	130.816	-7.506	24.415	0.000	0.000	2	0	0
P[003]	373.128	164.004	-7.506	24.415	0.000	0.000	3	1	0
P[1] ✕									
J1/X	<input type="text" value="13.227"/>	J2/Y	<input type="text" value="13.730"/>	坐标系	<input type="text" value="1"/>	臂参数1	<input type="text" value="1"/>	臂参数2	<input type="text" value="0"/>
J3/Z	<input type="text" value="-168.882"/>	J4/A	<input type="text" value="-2.542"/>	工具号	<input type="text" value="1"/>	臂参数3	<input type="text" value="0"/>	臂参数4	<input type="text" value="0"/>
J5/B	<input type="text" value="0.000"/>	J6/C	<input type="text" value="0.000"/>	用户号	<input type="text" value="2"/>	取当前臂参数		确定	取消

位置变量 P[2]表示在基坐标系下取的点，当时所采用的工具号为 1，用户号为 2。双击 P[2]点所在的行后，即可以查看其所带的工具号以及用户号。

位置变量 P[3]表示在工具坐标系下取的点，当时所采用的工具号为 1，用户号为 2。双击 P[3]点所在的行后，即可以查看其所带的工具号以及用户号。

位置变量 P[4]表示在用户坐标系下取的点，当时所采用的工具号为 1，用户号为 2。

备注：位置变量为局部变量，作用域为当前程序，如果需要在子程序调用主程序中的点，可通过 USING MAIN 指令，详见指令 USING MAIN。

2.2.4 全局平移变量

全局平移变量是指作为全局变量的平移变量，存储于控制器中，作用域可超出当前的程序段。

表示方法：PR***

***为变量编号，取值范围<0~255>

范例：

变量名	J1/X	J2/Y	J3/Z	J4/A	J5/B	J6/C
PR000	10.000	20.000	30.000	40.000	50.000	60.000

平移变量只是一组数值，在不同情况下使用，代表的意义不同。

详见 1.4 节。

2.2.5 局部平移变量

局部平移变量是作为局部变量的平移变量，仅在当前程序中有效。其它同全局平移变量。

表示方法：LPR***

***为变量编号，取值范围<0~255>

2.2.6 信号变量

信号变量分为模拟量和数字量两种。数字量为数值上离散的量，例如简单的 IO 信号开关，只有 ON 和 OFF 两种。模拟量为数值上连续的量，其值可包含小数，比如电流值，电压值。

2.2.7 字符串变量

字符串变量定义于程序中，只有先被定义，才能被使用。

字符串变量名由字母或数字组成，并只能以字母开头，长度不超过 10 个字符。
字符串变量的内容长度不超过 100 个的字符（示教器编程界面使用指令“字符串定义”时限制长度 50 个字符），不能包含中文、全角字符、双引号，也不能为关键字或其它已存在变量，如关节插补 Movj、速度表示 V、全局变量 B1、位置变量 P[1]（或 P）等。

范例：

```
String str1 = "abcd";    //先定义  
str1 = Left(str1,1);    //再使用
```

2.3 运算指令

2.3.1 基本运算指令

关系操作符

- == 关系等于
- > 关系大于
- <关系小于
- >= 关系大于或等于
- <= 关系小于或等于
- <>关系不等于

逻辑类：

- AND 逻辑与
- OR 逻辑或

简单运算类：

- = 赋值运算符
- + 加法运算符
- 减法运算符
- * 乘法运算符
- / 除法运算符
- % 取余运算符

函数运算类：（正三角函数的输入值单位均为度，反三角函数输出值单位也均为度）

- Sin() 正弦运算
- Cos() 余弦运算
- Tan() 正切运算
- ASin() 反正弦运算
- ACos() 反余弦运算
- ATan() 反正切运算
- Sqrt() 开平方运算

特殊符号类：

- ## 注释
- ； 分号,位于行末，代表一行语句的结束
- ： 冒号，用于提示下文，标签 L 指令、Switch-Case-Default 等中有用到
- ， 逗号，起间隔作用
- “ ” 双引号,表明该内容为字符串

2.3.2 Incr

功能：数值变量的自增

格式：Incr 数值变量;

参数	意义
数值变量	B/R/LB/LR 变量

范例：

B1=1;

Incr B1; ##B1 自增 1， 值变为 2

2.3.3 Decr

功能：数值变量的自减

格式：Decr 数值变量;

参数	意义
数值变量	B/R/LB/LR 变量

范例：

B1=1;

Decr B1; ##B1 自减 1， 值变为 0

2.3.4 数值运算

功能：数值变量的赋值

格式：数值变量= 表达式;

参数	意义
数值变量	B/R/D/LB/LR/LD 变量
表达式	由变量、数字、运算符组成的表达式

范例：

R7=17;

LD3=(Sin(30)+D1)/2.1;

2.3.5 字符串定义

功能：实现字符串变量的定义，定义的同时也可进行初始赋值。

格式：

String <变量名>;

String <变量名> = “字符串”;

参数	意义
<变量名>	由字母或数字组成，并只能以字母开头，长度不超过 10 个字符。若只输入数字当做变量名，系统会自动增加前缀“Str”，形成合法的变量名。
“字符串”	长度不超过 50 个的字符，不能包含中文、全角字符、双引号，也不能为关键字或其它已存在变量，如关节插补 Movj、速度表示 V、全

局变量 B1、位置变量 P[1]（或 P）等。

范例:

```
String ss1="hello_world";
```

2.3.6 字符串运算

功能: 进行字符串运算

格式: <变量名>=<字符表达式>;

说明: <字符表达式>有以下几种类型, 并可为两个<字符表达式>叠加。如 `str2= str1+Left(Str1,2)`。

a) <变量名>=<变量名>

功能: 将一个字符串变量赋值给另一个字符串变量

格式: <变量名>=<变量名>;

范例: `str2 = str1;`

b) <变量名>="字符串"

功能: 用于给字符串变量赋值

格式: <变量名>="字符串";

范例: `Str1 = "abcdefg";`

c) Left

功能: 取字符串变量的左边 n 个字符

格式: <变量名>=Left(字符串变量, n);

说明: n 可以为数字, 也可以为 B、R、LB、LR 变量。

范例:

```
Str1 = "abcdefg";
```

```
Str2 = Left(Str1,2);    ##取 Str1 的左边 2 的字符, 结果为 Str2 = "ab"
```

d) Right

功能: 取字符串变量的右边 n 个字符

格式: <变量名>=Right(字符串变量, n);

说明: n 可以为数字, 也可以为 B、R、LB、LR 变量。

范例:

```
Str1 = "abcdefg";
```

```
Str2 = Right(Str1,2);    ##取 Str1 的右边 2 的字符, 结果为 Str2 = "fg"
```

e) Mid

功能: 取字符串变量的中间的一段字符。

格式: <变量名>=Mid(字符串变量,i,n);

说明: 取从 i 开始 n 个字符。i 为索引号, 可为数字从 0 开始, 也可以为 B、R、LB、LR 变量。

n 为所取字符个数, 值范围 1-100, 也可取 B、R、LB、LR 变量。

范例:

```
Str1 = "abcdefg";
```

Str2 = Mid(Str1,3,3); ##取 Str1 序号从 3 开始的 3 的字符, 结果为 Str2 = "def"

f) GetAt

功能: 取字符串的某一位字符

格式: <变量名> = GetAt(字符串变量, i);

说明: 取索引号位 i 的字符。i 为索引号, 可为数字从 0 开始, 也可以为 B、R、LB、LR 变量。

范例:

Str1 = "abcdefg";

Str2 = GetAt(Str1, 3); ##取 Str1 序号为 3 的字符, 结果为 Str2 = "d"

2.3.7 字符串转换

功能: 进行字符串之间的相关转换

格式: <变量名> = <函数表达式>;

注意事项: 字符串转化并不改变等式右侧的变量参数, 只是对等式左边的变量赋值。

a) Caps

功能: 取指定字符串的大写形式

格式: <变量名> = Caps(字符串变量);

说明: Caps 对于非英文字母不作处理。

范例:

Str1 = "aB12";

Str2 = Caps (Str1); ##Str2 = "AB12"

b) LowCase

功能: 取指定字符串的小写形式

格式: <变量名> = LowCase(字符串变量);

说明: LowCase 对于非英文字母不作处理。

范例:

Str1 = "aB12";

Str3 = LowCase(Str1); ##Str 3= "ab12"

c) RToStr

功能: 将 R/LR 变量转化为字符串变量

格式: <变量名> = RToStr(R/LR***);

范例:

R1 = 45;

Str4 = RToStr(R1); ##Str4 = "45"

d) DToStr

功能: 将 D/LD 变量转化为字符串变量

格式: <变量名> = DToStr(D/LD***,m,n);

说明:

m 限定整数位数。若参数 m 小于或等于 D/LD***的实际整数位数, 则整数部分不做处理,

将全部输出；若参数 m 大于 D/LD^{***} 的整数位数，则在左侧以空格补齐位数。
 n 限定小数位数。若参数 n 小于或等于 D/LD^{***} 的实际小数位数，则小数部分四舍五入取近似值；参数 n 若大于 D/LD^{***} 的小数位数，则从右侧以 0 补齐字符串。

范例：

```
LD1 = 1.36;
```

```
Str5 = DToStr(LD1,2,1); ##结果 Str5 = " 1.4"，注意 1 前有个空格
```

e) StrToR

功能：将字符串变量转换为整型数据，并赋给 R/LR 变量

格式：R 变量 = StrToR(字符串变量);

说明：StrToR 只识别的前面的数字位（从左到右），遇到非数字位停止；若遇到第一位为非数字位，则返回 0。

范例：

```
A1 = "a12";
```

```
A2 = "12a3";
```

```
A3 = "1234"
```

```
R2 = StrToR(A1);      ##R2 = 0
```

```
R3 = StrToR(A2);      ##R3 = 12
```

```
R4 = StrToR(A3);      ##R4 = 1234
```

f) StrToD

功能：将字符串变量转换为双精度数据，赋给 D/LD 变量

格式：D/LD^{***} = StrToD(字符串变量);

说明：StrToD 只识别的前面的数字位（从左到右），遇到非数字位停止；若遇到第一位为非数字位，则返回 0。

范例：

```
C1 = "123.456"
```

```
LD2 = StrToD(C1);     ##LD2 = 123.456
```

2.3.8 Strlen

功能：求字符串的长度

格式：B/R/LB/LR^{***} = Strlen(字符串变量);

范例：

```
Str1 = "abcd";
```

```
LB1 = Strlen(Str1);   ##结果 LB1 = 4
```

2.3.9 StrFind

功能：寻找字符串变量中的某个字符串的索引值

格式：R/LR^{***} = StrFind(字符串变量,"字符串");

说明：当找不到时，返回结果为-1。字符串变量为 Port 时特指端口（接受缓冲区）数据，即查找接收缓冲区里的字符串。

范例：

```
Str1 = "abcde";
LR1 = Strlen(Str1,"cd");    ##结果 LR1 = 2
```

2.3.10 Strcmp

功能：比较两个字符串大小，结果赋值给 R/LR 变量

格式：R/LR*** = Strcmp (字符串变量 1, 字符串变量 2);

说明：两个字符串自左向右逐个字符相比（按 ASCII 值大小相比较），直到出现不同的字符或遇'\0'为止。把第一个不同的两个字符的 ASCII 码之差作为比较结果赋给 R/LR 变量，若全部相同则返回 0。

范例：

```
Str1="ABxC";
Str="ABzH";
R=Strcmp(Str1,Str2);    ##R=-2
```

2.3.11 StrGetData

功能：从指定字符串中取出数据，存放在指定的变量中，并返回取出字符串的个数

格式：返回值 = StrGetData(参数 1,参数 2,参数 3);

参数号	形式	意义
返回值	B/R/LB/LR***	分割取出的数据的个数,可以是 B/R/LB/LR 变量
参数 1	字符串变量	源字符串变量
	Port	端口接收的字符串
参数 2	分隔符	分割符,其值为一个字符串
参数 3	B/R/D***	数据存储于当前 B/R/D 变量及其后的变量中
	P[***]	按照 P 变量的格式存储数据,存储到 P[***]中
	PR***	按照 PR 变量的格式存储数,存于 PR***中

说明：将字符串变量按分隔符分割，将分割出的数据存入对象，并返回分割所得个数。

字符串变量：指待分割的字符串。当为 Port 时，用于视觉应用，指接收缓冲区的字符串。

分隔符：其值为一个字符串

存入分割数据的对象：有三种表现形式 B/R/D***、P[***]、PR***。当为 B/R/D***，会将分割所得的字符串转成相应类型，连续存储于以当前 B/R/D 变量为始的一串变量。当为 P[***]，按照 P 变量的格式存储数据，存储到 P[***]中；当为 PR***时，按照 PR 变量的格式存储数，存于 PR***中。使用 P\PR 时，只分割出的第一个字符串，按 P\PR 格式存储。当分割后的形式不符合 P\PR 类型时，将为空，同时分割个数为 0。

范例 1:

```
Str1 = "123And45Andggg"
B0= StrGetData(str1,"And",R0);
```

说明：分割成"123","45","ggg"三个字符串，结果为 B0=3, R0=123, R1=45, R2=0。

范例 2:

Str2 = "1,2,3,4,5,6;1,1,1,0;4,2,1;\$7,8,9,4,5,6;"

B0= StrGetData(str2,"\$",P[1]);

说明：分割的第一个字符串" 1,2,3,4,5,6;1,1,1,0;4,2,1;"存储到 P[1]中，剩余的部分不存储，B0=1，结果如图：

变量名	J1/X	J2/Y	J3/Z	J4/A	J5/B	J6/C	坐标系	工具号	用户号
P[001]	1.000	2.000	3.000	4.000	5.000	6.000	4	2	1

四个 ArmType 分别为 1,1,1,0

2.3.12 GetPortbuf

功能：获取接收缓冲区的字符串

格式：字符串变量 = GetPortbuf(起始位 , 字符个数);

参数	意义
起始位	可为数字或 B/LB 变量
字符个数	可为数字或 B/LB 变量
字符串变量	返回值，存放接收到的字符串

说明：常用于视觉，在接收缓冲区中从某一位 (<起始位>) 开始取连续 N 个 (<字符个数>) 字符，并传给指定的字符串变量。失败时传 0。起始位从 0 开始计数。由于字符串变量长度不能超过 100 个字符，因此字符个数<=100

范例：

Str1 = GetPortbuf(2,3); ##从索引号为 2 的位开始，取 3 个字符

2.3.13 GetCurPoint

功能：获取当前点在关节或基坐标系下的参数值

格式：GetCurPoint(参数 1,参数 2,参数 3);

参数号	形式	意义
参数 1	坐标系号	这里的坐标系号只能取 1 或 2。 坐标系号 1 表示关节坐标系，坐标值为 (J1,J2,J3,J4,J5,J6)。 坐标系号 2 表示基坐标系，坐标值为机器人本体末端在基坐标系下的坐标值(X,Y,Z,A,B,C)。
参数 2	下标号	参数 3 为 P[***]时，下标号只取 0 才有意义，表示保存转换后的位置变量的所有坐标值。 参数 3 为 D/LD***时，此时下标号取 0~5，表示从 6 个坐标值中取其中一个保存。
参数 3	D/LD***	保存转换后的位置变量中的某一项坐标值
	P[***]	保存转换后的位置变量

若最后一个参数为 P[***]时，数据存储到 P 变量中，此时下标号只取 0 才有意义，否则不会处理。

若最后一个参数为 D/LD***时，数据存储到 D/LD 变量中，此时下标号取 0~5，表示从 6 个坐标值中取其中一个保存。

范例：

GetCurPoint(2,1,D1); ##取当前点在基坐标系下的 Y 坐标值，赋值给 D1

GetCurPoint(1,0,P[1]); ##取当前点在关节坐标系下值，赋值给 P[1]

2.3.14 Cnvrt

功能：点的坐标系转换

格式：Cnvrt (参数 1,参数 2,参数 3);

Cnvrt (参数 1,参数 2,参数 3,参数 4);

参数号	形式	意义
参数 1	P[i]	待转换的点，i 可为数字或 R/LR 变量
参数 2	P[j]	转换后的点，j 可为数字或 R/LR 变量
参数 3	Joint	转换成关节坐标系点，点的工具号、用户号不变
	World	转换成基坐标系点(相当于在基坐标系下取点，坐标值为机器人本体末端在基坐标系下的位姿值)，点的工具号、用户号不变
	Tool[k]	转换成工具坐标系点(相当于在工具坐标系下取点，坐标值为工具在基坐标系下的位姿值)，点的用户号不变
	User[k]	转换成用户坐标系点(相当于在用户坐标系下取点，坐标值为工具在用户坐标系下的位姿值)，点的工具号不变
参数 4	P[k]	随动相机的拍照点，当且仅当 P[i]坐标系号为 6 时有效，其他情况均报错。

注意：

参数 1 P[i] 的坐标系号取 5，将能把固定相机坐标系下的点，转换到机器人的关节/基/工具/用户坐标系下。

参数 1 P[i] 的坐标系号取 6，将能把随动相机坐标系下的点，转换到机器人的关节/基/工具/用户坐标系下。

范例：

Cnvrt (P[1],P[2], Joint); ##将 P[1]转换到关节坐标系下，赋值给 P[2]

2.3.15 P[***]=

功能：位置变量的赋值。

格式：

P[***]=(X,Y,Z,A,B,C),(ArmType[0],ArmType[1],ArmType[2],ArmType[3]),(坐标系号,工具号,用户号);

说明：

第一个括号内的参数为坐标值，当坐标系号取 1 时为(J1,J2,J3,J4,J5,J6)，坐标系号取 2、3、4 时为(X,Y,Z,A,B,C)。(ArmType[0], ArmType[1], ArmType[2], ArmType[3])为臂参数；坐标系号、工具号、用户号的指所使用的工具和用户坐标系。

范例：P[3] = (10,50,30,40,50,60),(1,-1,1,0),(4,1,1);

2.3.16 GetPValue

功能：获取位置变量的坐标值元素

格式：GetPValue(P[***],下标,D/LD***);

说明：

P[***]：位置变量，***可以为数字或 R/LR 变量

下标：位置变量坐标值元素的下标，可为数字或 R/LR 变量

D/LD***：用于存储位置变量坐标值元素的值

范例：

P[3] = (10,50,30,40,50,60),(1,-1,1,0),(4,1,1);

GetPValue(P[3],2,D1); ##取 P[3]中索引号为 2 的元素，结果存于 D1 中。本例结果 D1=30

2.3.17 SetPValue

功能：设置位置变量的坐标值元素

格式：SetPValue(P[***],下标,D/LD***);

说明：

P[***]：位置变量，***可以为数字或 R/LR 变量

下标：位置变量元素的下标，可为数字或 R/LR 变量

D/LD***：要设置的位置变量元素的值

范例：

P[3] = (10,50,30,40,50,60),(1,-1,1,0),(4,1,1);

D1=55;

R1=2;

SetPValue(P[3],R1,D1); ##设置 P[3]中索引号为 R1 的元素值为 D1。

##本例结果 P[3] = (10,50,55,40,50,60),(1,-1,1,0),(4,1,1);

2.3.18 SetCoordParm

功能：设置位置变量的坐标系参数

格式：SetCoordParm(P[***],坐标系号,Tool[***],User[***]);

说明：

P[***]:要更改的位置变量，***可以为数字或 R/LR 变量

坐标系号（可选项）：坐标系号可为数字或 B/LB 变量。数值范围(1,7)超出范围表示该项设置无效，保持 P 点的原有设置。

Tool[工具号]（可选项）：工具号可为数字或 B/LB 变量，数值范围（0,15），超出范围表示该项设置无效，保持 P 点的原有设置。

User[用户号]（可选项）：用户号可以为 B/LB 变量，也可以为数值，数值范围（0,15），超出范围表示该项设置无效，保持 P 点的原有设置。

注意事项：坐标系号、工具号、用户号共 3 个可选项至少选择 1 项。

范例：

P[1] = (10,50,30,40,50,60),(1,-1,1,0),(4,1,1);

SetCoordParm(P[1],4,Tool[2],User[3]); ##设置 P[1]坐标系号 4，工具号 2，用户号 3。

##本例结果 P[1] = (10,50,30,40,50,60),(1,-1,1,0),(4,2,3);

2.3.19 GetCoordNo

功能：获取位置变量的坐标系号

格式：GetCoordNo(P[***],B/LB***);

说明：

P[***]:位置变量，***可以为数字或 R/LR 变量。

B/LB***: 存储坐标系号。

范例：

P[3] = (10,50,30,40,50,60),(1,-1,1,0),(4,1,5);

GetCoordNo(P[3],B1); ##获取 P[3]坐标系号，结果 B1=4

2.3.20 GetToolNo

功能：获取位置变量的工具号

格式：GetToolNo(P[***],B/LB***);

说明：

P[***]:位置变量，***可以为数字或 R/LR 变量。

B/LB***: 存储工具号。

范例：

P[3] = (10,50,30,40,50,60),(1,-1,1,0),(4,1,5);

GetToolNo(P[3],B1); ##获取 P[3]坐标系号，结果 B1=1

2.3.21 GetUserNo

功能：获取位置变量的用户号

格式：GetUserNo(P[***],B/LB***);

说明：

P[***]:位置变量，***可以为数字或 R/LR 变量。

B/LB***: 存储用户号。

范例：

P[3] = (10,50,30,40,50,60),(1,-1,1,0),(4,1,5);

GetUserNo(P[3],B1); ##获取 P[3]坐标系号，结果 B1=5

2.3.22SetArmType

功能：设置位置变量的臂参数

格式：SetArmType(P[***],P[***]);

说明：

将第一个位置变量的臂参数值设为与第二个的臂参数相同。

范例：

P[1] = (10,50,30,40,50,60),(1,1,1,0),(4,1,1);

P[2] = (10,55,30,40,50,60),(1,-1,1,0),(4,1,1);

SetArmType(P[1],P[2]); ##P[1]臂参数设为与 P[2]相同, 即也为(1,-1,1,0)

2.3.23 SetToolParm

功能: 设置工具坐标系参数.

格式 1: SetToolParm (工具号,PR***/LPR***);

说明: 利用平移变量直接设置工具坐标系参数。(事先将要设置的坐标系参数存储于平移变量中)

参数: PR***/LPR***为平移变量, 用于存储工具坐标系参数。

格式 2: SetToolParm (工具号,P[i],P[j],P[k]);

说明: 利用三点法TCP计算工具坐标系并设置。(可参看3.2.3 (a) 节中工具坐标的三点法TCP)

参数: P[i],P[j],P[k]为位置变量。

格式 3: SetToolParm (工具号,OFF);

说明: 还原原工具坐标系参数。还原为原设置的值。

注意事项:

若用到使用的 PR***/LPR***或 P[i],P[j],P[k], 则这些参数必须提前定义。

设置的坐标系值只在程序中临时使用,【坐标系设置】页面的显示值并不会改变。使用完后,利用 SetToolParm (工具号,OFF)可还原。SetToolParm 指令作用范围: 仅当前程序有效, 进入子程序或者从子程序返回主程序后, 参数还原。

2.3.24 SetUserParm

功能: 设置用户坐标系参数.

格式 1: SetUserParm (用户号,PR***/LPR***);

说明: 利用平移变量中的坐标系值直接设置用户坐标系参数。

参数: PR***/LPR***为平移变量, 用于存储用户坐标系参数。

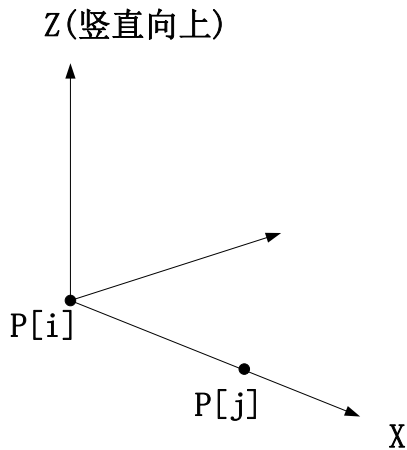
格式 2: SetUserParm (用户号,P[i],P[j],P[k]);

说明: 利用三点法计算用户坐标系值并设置。(可参看3.2.3 (b) 节中用户坐标的三点法)

参数: P[i],P[j],P[k]为位置变量。

格式 3: SetUserParm (用户号,P[i],P[j]);

说明: 利用两点法计算用户坐标系值并设置。这里的两点法, 是以P[i]作原点, 默认用户坐标系的z轴竖直朝上, 因此只需取两个点P[i],P[j]确定x轴方向。



参数：P[i],P[j]为位置变量。

格式 4: SetUserParm (工具号,OFF);

说明：还原原工具坐标系参数。还原为原设置的值。

注意事项：

若用到使用的 PR***/LPR***或 P[i],P[j],P[k]，则这些参数必须提前定义。

设置的坐标系值只在程序中临时使用，【坐标系设置】页面的显示值并不会改变。使用完后，利用 SetUserParm (工具号,OFF)可还原。SetUserParm 指令作用范围：仅当前程序有效，进入子程序或者从子程序返回主程序后，参数还原。

2.3.25 OffSetUserParm

功能:把原用户坐标系进行整体平移

格式 1: OffSetUserParm (用户号,PR***/LPR***);

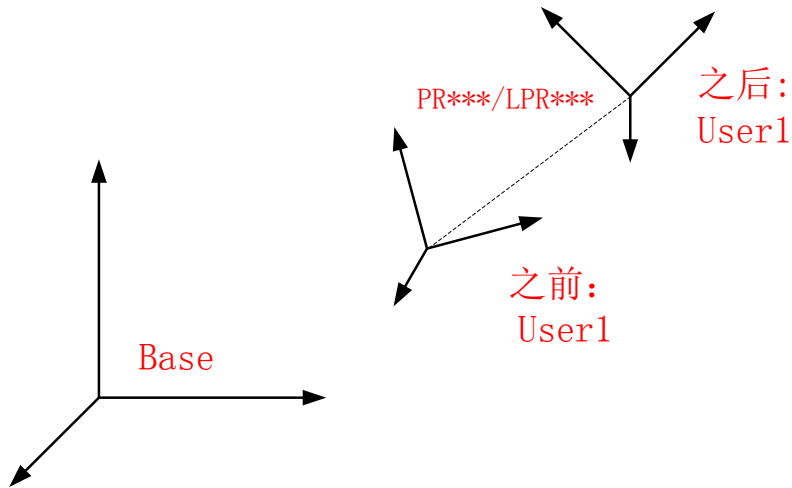
说明：对用户坐标系平移，平移量存于 PR/LPR 变量中。

参数：

用户号：要平移的用户坐标系号

PR***/LPR***：在原用户坐标系基础上进行平移的平移变量

图示：



格式 2: `OffSetUserPam (用户号,OFF);`

说明: 关闭用户坐标系的平移

参数:

用户号: 要平移的用户坐标系号

OFF: 代表关闭用户坐标系的平移, 还原用户坐标系参数

注意事项:

若用到使用的 `PR*** / LPR***`, 则这些参数必须提前定义。

设置的坐标系值只在程序中临时使用, 【坐标系设置】页面的显示值并不会改变。使用完后, 利用 `OffSetUserPam (工具号,OFF)`可还原。

范例:

```
PR1=(20,30,0,0,0,0);
```

```
OffSetUserPam(2,PR1);
```

2.3.26 Clear

功能: 清除平移变量

格式: `Clear PR*** / LPR***;`

参数: `PR*** / LPR***`: 待清除的平移变量

范例:

```
Clear PR2;
```

2.3.27 StrToAscii

功能: 字符串转换成 ASCII 码

格式: `R/LR*** = StrToAsii(字符串变量,字符个数,B/LB***);`

参数:

字符串变量: 需获取 ASCII 码的字符串

字符个数: 需从字符串中转换成 ASCII 码的字符的个数, 不得大于字符串的长度。

B/LB***: 用于存储字符串的 ASCII 码, 存储于从 B***开始的指定字符个数的连续 B 变量中
返回值: 转换成功的字符个数, 存于 R\LR 变量中。

范例:

String str1 = "qwe";

LR1 = StrToAscii(str1,3,LR1); ##LB1、LB2、LB3 分别为 97、98、99, LR1=3

2.3.28 Dist

功能: 求两个点之间的直线距离

格式: D<变量名> = Dist(P[1],P[2]);

参数:

D<变量名>: 用于存储两个点之间的距离, 可以为 D/LD 变量

P[1]/P[2]: 用于求距离的两个点

备注:

当 P[1],P[2]坐标系号不一致时, P[2]会转换成 P[1]坐标系下的点之后再求距离, 当 P[1]为关节坐标系下的点时, 默认都转换成世界坐标系下求距离, 当 P[1]为工具坐标系下的点时, 且与 P[2]有不同的工具号, 则转换成 P[2]所带的工具坐标系下的点的值, 然后再求距离。

2.4.运动指令

2.4.1 Movj

功能: 关节插补

格式: Movj <参数 1>,<参数 2>,<参数 3>,<参数 4>;

参数号	形式	意义
参数 1	P[***]	目标点。***为位置点标号, 可用数字直接表示, 也用变量 B/R/LB/LR 间接表示。如 P[2]、P[B1]。P[***]取值范围详见 2.1.3 节位置变量。
	Offset(P[***],PR***)	以偏移的方式得到的目标点。Offset 表示在点 P[***]位置偏移 PR***后得到的新位置。也可使用 LPR***代替 PR***。PR***/LPR***取值范围详见 2.1.4 节平移变量。P[***]取 PE 时代表取当前点做偏移。
	OffsetJ(P[***],PR***)	关节坐标系下平移
	OffsetT(P[***],PR***)	工具坐标系方向上平移
	Pallet(托盘号,行号,列号,层号)	根据 (托盘号,行号,列号,层号) 这些信息取托盘上的点, 详见 2.5.5、2.5.6 节 Pallet 指令。

参数 2	V[***]	指定机器人的运动速度。***是最大速度的百分比，取 1-100 的整数。如 V[50]代表 50%最大限定速度。
参数 3	Z[***]	插补精度等级设置，目前有 Z[0]~Z[5]六个等级，数字越小精度越高。
参数 4 (可选参数)	User[用户号]	选用的用户坐标系，用户号可取 0~15
	Tool[工具号]	选用的工具，工具号可取 0~15
	Acc[***]	指定机器人的运动速度。***是最大加速度的百分比，取 1-100 的整数。如 Acc[50]代表 50%的最大加速度。
	Until IN[输入端口号]==OFF/ON	运动传感指令，表示在运动过程中检测某一信号量，当条件满足，则当前运动执行完毕，运行下一行程序；若条件不满足，则运动到结束点。
	OUT(IO 号,ON/OFF ,T[n]	并行 IO 输出指令。T[n]为时间，范围 -65535.000-65535.000，单位秒。 运动期间最多 2 个 IO 触发。 n>=0 表示开始运动 n 秒后输出信号，n<0 表示到达运动点之前 n 秒时输出信号。

偏移“Offset”的使用：

以偏移的方式得到的目标点位置。根据需要在哪个坐标系上平移，选择合适的平移类型。

(1) 对于关节平移：带有参数 Offset,则表示在关节坐标系下平移，平移变量则表示关节坐标系下的平移量。

(2) 对于非关节平移：带有参数 Offset，只需在指令中指定 Tool[***],User[***]参数即可。

Tool[***]	User[***]	平移类型
×	×	机器人末端在基坐标系下的平移
√	×	工具 Tool[***]末端在基坐标系下的平移
×	√	机器人本体末端在 User[***]坐标系下的平移，当 User[***]参数中的用户号和位置变量中的用户号不一致时则分为两种情况处理： 1.位置变量为用户坐标（P 的坐标系号为 4），则表示该用户坐标值不变，对应到 User[***]用户坐标系下的位置后，然后再在 User[***]坐标系下作平移。即进行用户坐标系切换后再作平移，基准点的绝对位置发生了变化,如图 1 所示。 2.位置变量不为用户坐标（P 的坐标系号不为 4），直接作 User[***]坐标系方向上的平移，即平移的基准点不发生变化，如图 2 所示。
√	√	工具 Tool[***]末端在 User[***]坐标系下的平移，当位置变量的用户号与 User[***]不一致时，情况同上，若 Tool[***]与位置变量中的

		工具号不一致时，则进行了工具的切换之后再平移。
√	×	工具 Tool[***]末端在基坐标系下的平移

图 1:

情形: Movj 带有 User 参数, P 的坐标系号为 4。

特点: 位置变量的绝对位置发生变化, 再进行平移。如案例中, 最终运动到位置 P[3]

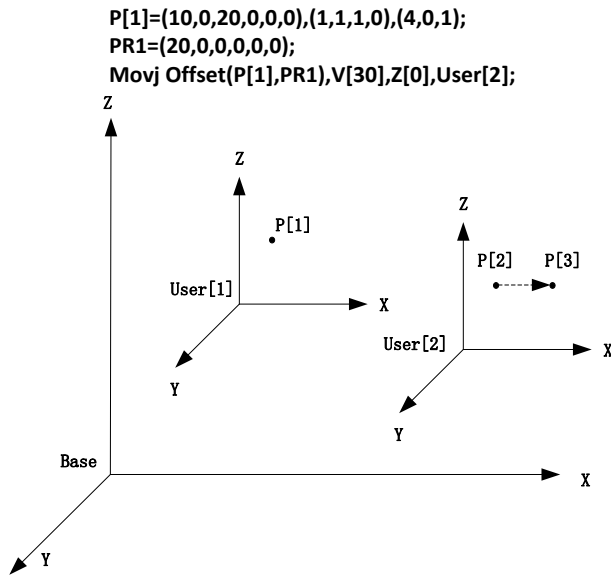


图1

图 2:

情形: Movj 带有 User 参数, P 的坐标系号为 0 或 1 或 2。

特点: 位置变量的绝对位置不发生变化, 直接进行平移。如案例中, 最终运动到位置 P[4]

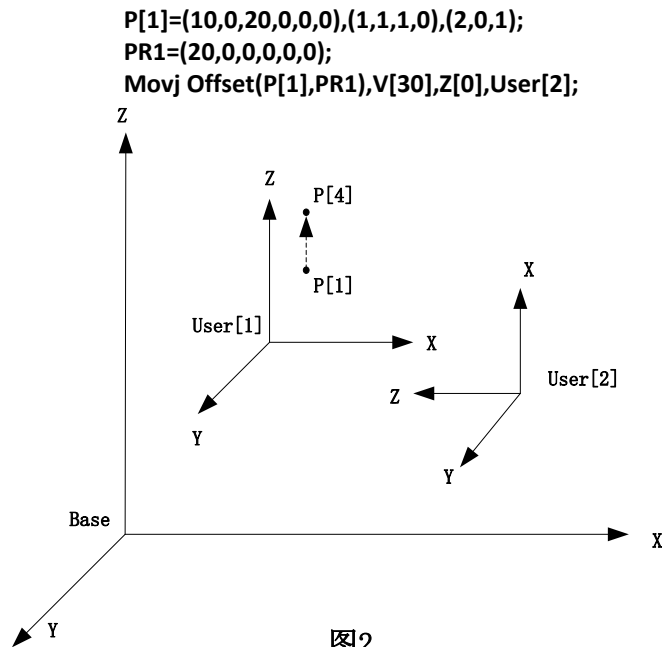


图2

(3) 工具相对工具坐标系自身的平移：带有参数 `OffsetT` 时，机器人作工具末端在工具坐标系放下下的平移。

Tool 参数：

用于切换工具。使用新的工具时，工具末端到达原工具末端的位置。

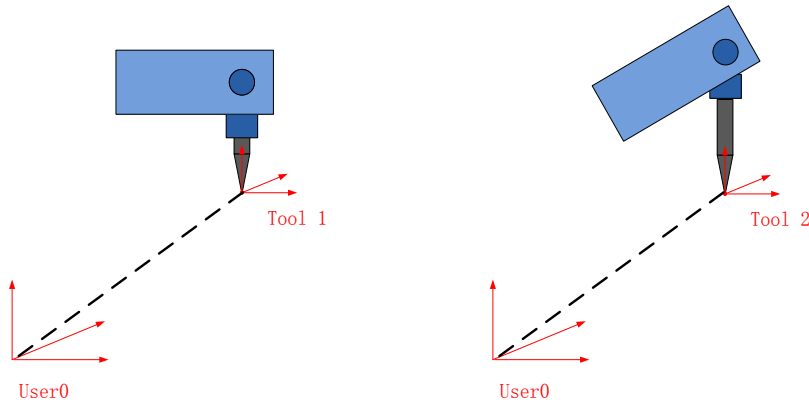
应用案例：

在一个程序中，同一机器人上加持多个工具在工件上进行同一种运动，则只需使用其中一个工具示教。使用其它工具加工时，则只用在 `Mov` 后加对应的刀具号即可。

范例：

```
START;  
Movj P[1],V[30],Z[3];  
Movj P[1],V[30],Z[3],Tool[2];  
END;
```

说明：假设定义 `P[1]`采用坐标系,4，采用工具号 1，用户号 0；则执行第二条 `Movj` 指令的效果是使用工具 2，新的 TCP 运动到空间同一位置。



注意事项：在 `Movj` 指令中使用 `Tool`，可能造成无谓的运算，使得在奇异位置处出现问题。

User 参数：

用于切换用户坐标系。使得在切换用户坐标系后，仍可到达用原来的相对位姿。

注意事项：`Mov` 中的点必须是在用户坐标系下取用的，即 `P` 的坐标系号必须为 4。当不满足该条件时，机器人将不做用户坐标系的切换，机器人仍然运行至原来取点的位置。

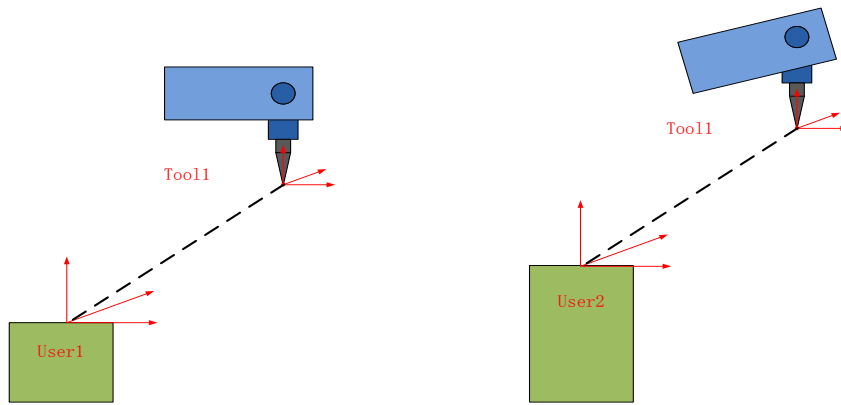
应用案例：

需要在一个程序中利用机器人对不同位置的同一批工件进行相同的加工，则只需对一处的工件（使用一个用户坐标系）示教。加工其它处的工件时，则只用在 `Mov` 后加对应的用户号即可。

范例：

```
START;  
Movj P[1],V[30],Z[3];  
Movj P[1],V[30],Z[3],User[2];  
END;
```

说明：假设定义 `P[1]`采用坐标系 4，工具号 1，用户号 0；执行第 2 条 `Movj` 指令的效果是切换到用户坐标系 2，TCP 运动到相对新的用户坐标系的相同偏移量的位置。



Mov 后同时存在 Tool、User 则是用于同时切换工具和用户坐标系。

Until 参数:

该参数的效果是向目标位置运动期间，不断检测输入端口的信号，若满足要求则立即停止，否则一直运动直到结束。

范例:

START;

Movj P[1],M[30],Z[3],ACC[50],Until IN[7]==ON;

END;

说明: 选取工具 2, 以关节插补运动向 P[1]位置运动, 加速度上限设定为 50%的最大加速度。期间, 输入端口 IN[7]收到信号为“ON”时, 运动立即停止; 若一直未收到“ON”信号, 则会运动直至结束。

OUT(IO 号,ON/OF,T[n]):

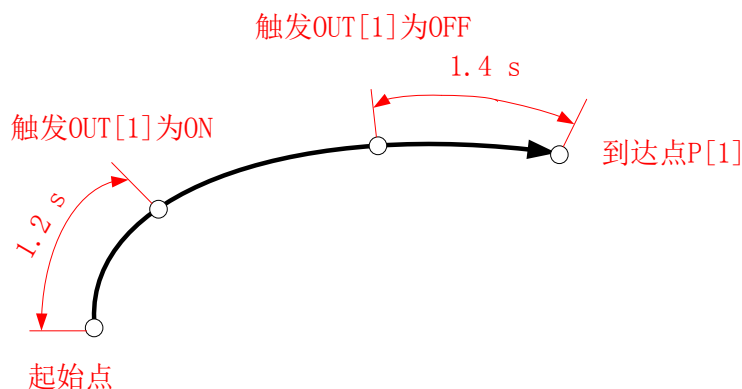
并行 IO 输出指令。一条运动指令中最多 2 个 IO 触发。

T[n]为时间, 范围-65535.000-65535.000, 单位秒。大于或者等于 0 表示开始运动 n 秒后输出信号, 小于 0 表示到达运动点之前 n 秒时输出信号。

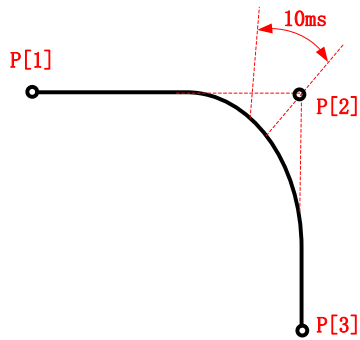
当 T 取超出运动范围外的时间时, 最多只在该段行程首末点触发, 设定的时间无效。

范例:

Movj P[1],M[30],Z[3],OUT(1,ON,T[1.2]),OUT(1,OFF,T[-1.4]);



注意: OUT(IO 号,ON/OF,T[n])在过渡阶段依然有效, 只是判定点不同。如下图所示。



2.4.2 Movl

功能：直线插补

格式：Movl <参数 1>,<参数 2>,<参数 3>,<参数 4>;

参数号	形式	意义
参数 1	P[***]	目标点。***为位置点标号，可用数字直接表示，也用变量 B/R/LB/LR 间接表示，如 P[2]、P[B1]。P[***]取值范围详见 2.1.3 节位置变量。
	Offset(P[***],PR***)	以偏移的方式得到的目标点。Offset 表示在点 P[***]位置偏移 PR***后得到的新位置。也可使用 LPR***代替 PR***。PR***/LPR***取值范围详见 2.1.4 节平移变量。P[***]取 PE 时代表取当前点做偏移。
	OffsetJ(P[***],PR***)	关节坐标系下平移
	OffsetT(P[***],PR***)	工具坐标系方向上平移
	Pallet(托盘号,行号,列号,层号)	根据（托盘号,行号,列号,层号）这些信息取托盘上的点，详见 2.5.5、2.5.6 节 Pallet 指令。
参数 2	V[***]	指定机器人的运动速度。***是最大速度的百分比，取 1-100 的整数。如 V[50]代表 50%最大限定速度。
参数 3	Z[***]	插补精度等级设置，目前有 Z[0]~Z[5]六个等级，数字越小精度越高。
参数 4 (可选参数)	User[用户号]	选用某个的用户坐标系,用户号可取 0~15
	Tool[工具号]	选用某个的工具,工具号可取 0~15
	Acc[***]	指定机器人的运动速度。***是最大加速度的百分比，取 1-100 的整数。如 Acc[50]代表 50%的最大加速度。

	<p>Until IN[输入端口号]==OFF/ON</p>	<p>运动传感指令，表示在运动过程中检测某一信号量，当条件满足，则当前运动执行完毕，运行下一行程序；若条件不满足，则运动到结束点。</p>
	<p>OUT(IO 号,ON/OFF ,T[n]/D[n]/S[n]</p>	<p>并行 IO 输出指令。可重复使用，同时多个 IO 触发。 T[n]为时间，单位：秒，范围 -65535.000~65535.000。大于 0 表示开始运动 n 秒后输出信号，小于 0 表示到达运动点之前 n 秒时输出信号。 D[n]为路径百分比，范围 0.000-100.000。表示从开始运动到结束整个路径的 n%时输出信号。 S[n]为距离，单位：mm。范围 -65535.000~65535.000。大于 0 表示从起点开始运动到 n 毫米之后时输出信号，小于 0 表示运动到距终点 n 毫米之前时输出信号。</p>

相同参数参照 Movj。

范例：

START;

Movl P[1],V[30],Z[3];

END;

说明：直线插补运动至 P[1]，运动速度设为 30%的最大速度，插补精度取 Z[3]。

OUT(IO 号,ON/OFF,T[n]/D[n]/S[n]):

并行 IO 输出指令。一条运动指令中最多设定 2 个 IO 触发。

T[n]为时间，单位：秒，范围-65535.000~65535.000。大于 0 表示开始运动 n 秒后输出信号，小于 0 表示到达运动点之前若干秒时输出信号。

D[n]为路径百分比，范围 0.000-100.000。表示从开始运动到结束整个路径的 n%时输出信号。

S[n]为距离，单位：mm。范围-65535.000~65535.000。大于 0 表示从起点开始运动到 n 毫米之后时输出信号，小于 0 表示运动到距终点 n 毫米之前时输出信号。

当 T[n]/D[n]/S[n]取超出运动范围外的设定时，最多只在该段行程首末点触发。

范例：

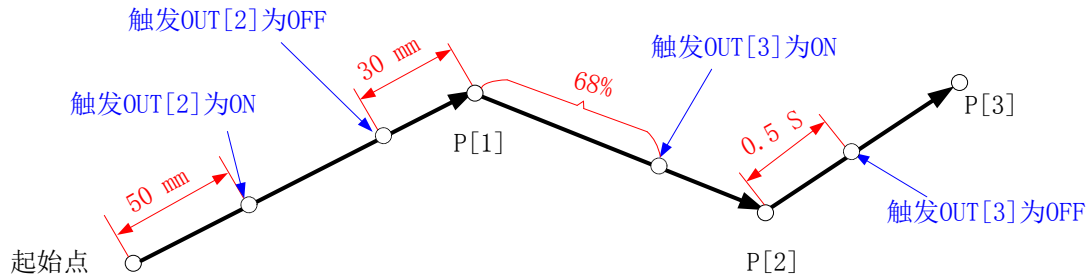
START;

Movl P[1],V[30],Z[0],OUT(2,ON,S[50]), OUT(2,OFF,S[-30]);

Movl P[2],V[30],Z[0],OUT(3,ON,D[68]);

Movl P[3],V[30],Z[0],OUT(3,OFF,T[0.5]);

END;



注意：OUT(IO 号,ON/OFF,T[n]/D[n]/S[n])在过渡阶段依然有效，只是判定点不同。

2.4.3 Movc

功能：圆弧插补

格式：Movc <参数 1>,<参数 2>,<参数 3>,<参数 4>;

参数号	形式	意义
参数 1	P[***]	目标点。***为位置点标号，可用数字直接表示，也用变量 B/R/LB/LR 间接表示。如 P[2]、P[B1]。P[***]取值范围详见 2.1.3 节位置变量。
	Offset(P[***],PR***)	以偏移的方式得到的目标点。Offset 表示在点 P[***]位置偏移 PR***后得到的新位置。也可使用 LPR***代替 PR***。PR***/LPR***取值范围详见 2.1.4 节平移变量。P[***]取 PE 时代表取当前点做偏移。
	OffsetJ(P[***],PR***)	关节坐标系下平移
	OffsetT(P[***],PR***)	工具坐标系方向上平移
	Pallet(托盘号,行号,列号,层号)	根据（托盘号,行号,列号,层号）这些信息取托盘上的点，详见 2.5.5、2.5.6 节 Pallet 指令。
参数 2	V[***]	指定机器人的运动速度。***是最大速度的百分比，取 1-100 的整数。如 V[50]代表 50%最大限定速度。
参数 3	Z[***]	插补精度等级设置，目前有 Z[0]~Z[5]六个等级，数字越小精度越高。
参数 4 (可选参数)	User[用户号]	选用某个的用户坐标系，用户号可取 0~15
	Tool[工具号]	选用某个的工具，工具号可取 0~15
	Acc[***]	指定机器人的运动速度。***是最大加速度的百分比，取 1-100 的整数。如 Acc[50]

		代表 50%的最大加速度。
	OUT(IO 号,ON/OFF ,T[n]/D[n]/S[n])	并行 IO 输出指令。可重复使用，同时多个 IO 触发。 T[n]为时间，单位：秒，范围 -65535.000~65535.000。大于 0 表示开始运动 n 秒后输出信号，小于 0 表示到达运动点之前 n 秒时输出信号。 D[n]为路径百分比，范围 0.000-100.000。表示从开始运动到结束整个路径的 n%时输出信号。 S[n]为距离，单位：mm。范围 -65535.000~65535.000。大于 0 表示从起点开始运动到 n 毫米之后时输出信号，小于 0 表示运动到距终点 n 毫米之前时输出信号。

相同参数参照 Movl。

注意事项：

1. 三点确定一段圆弧，将其分为两段 Movc 指令，因此 Movc 指令必须是成对出现。运行时，执行到第二条 Movc 时才会进行整个圆弧运动，且 Z[精度]以第一条 Movc 为准。
2. 如果两条连着的 Movc 前的上一条指令是 Movl 或 Movj 以外的指令，则单步示教时会产生报错提示“圆弧运动前缺乏 Movj 或 Movl”；连续示教或运行时则无问题。

范例：

START;

```
Movj P[1],V[80],Z[0];          ##快速运动至 P[1]位置
Movc P[2],V[80],Z[3];          ##运动经过 P[2]点到达 P[3]点，轨迹为圆弧
Movc P[3],V[80],Z[3];
END;
```

2.4.4 Jump

功能：跳跃指令

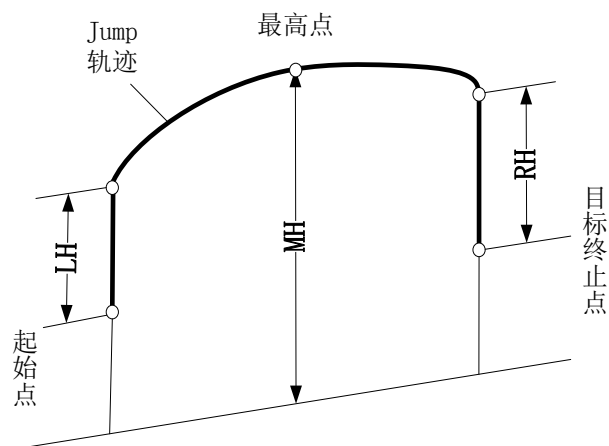
格式：Jump <参数 1>,<参数 2>,<参数 3>,<参数 4>;

参数号	形式	意义
参数 1	P[***]	目标位姿。***为位置点标号，可用数字直接表示，也用变量 B/R/LB/LR 间接表示，如 P[2]、P[B1]。P[***]取值范围详见 2.1.3 节位置变量。
	Offset(P[***],PR***)	以偏移的方式得到的目标点。Offset 表示在点 P[***]位置偏移 PR***后得到的新位置。也可使用 LPR***代替 PR***。PR***/LPR***取值范围详见 2.1.4 节平移变量。P[***]取 PE 时代表取当前点做偏移。

参数 2	OffsetJ(P[***],PR***)	关节坐标系下平移
	OffsetT(P[***],PR***)	工具坐标系方向上平移
	Pallet(托盘号,号行,号列号,层号)	根据(托盘号,行号,列号,层号)这些信息取托盘上的点, 详见 2.5.6 节 P=Pallet 指令。
	V[***]	指定机器人的运动速度。***是最大速度的百分比, 取 1-100 的整数。如 V[50]代表 50%最大限定速度。
参数 3	Z[***]	插补精度等级设置, 目前有 Z[0]~Z[5]六个等级, 数字越小精度越高。
参数 4 (LH、MH、RH 为必选参数, 其它为可选参数)	User[用户号]	选用某个的用户坐标系, 用户号可取 0~15。
	Tool[工具号]	选用某个的工具, 工具号可取 0~15。
	Acc[***]	指定机器人的运动速度。***是最大加速度的百分比, 取 1-100 的整数。如 Acc[50]代表 50%的最大加速度。
	OUT(IO 号,ON/OFF ,T[n])	并行 IO 输出指令。可重复使用, 同时多个 IO 触发。 T[n]为时间, 单位: 秒, 范围 -65535.000~65535.000。大于 0 表示开始运动 n 秒后输出信号, 小于 0 表示到达运动点之前 n 秒时输出信号。
	LH[***]	起始位置处的提升高度, 取值范围 0~2000。
	MH[***]	运行过程中最高点相对于基坐标系零点的高度, 取值范围-2000~2000。
	RH[***]	到终止位置的下降高度, 取值范围 0~2000。

相同参数参照 Movj。

Jump 指令的运动过程分为三段: 开始的一段上升 Movl, 中间的一段 Movj 和最后的一段 Movl。



注意事项:

Jump 指令仅用于 Scara 及 Delta 机器人。

若机器人的基坐标系零点设的高, MH 可能为负值。在正常情况下, 高度参数需满足 $MH >$

初始点高度+LH且 MH>终止点高度+RH。对于不满足这个条件的非正常设定,可能出现非“门”字形运动,应谨慎使用。

范例:

Jump P[1],V[30],Z[3],User[1],Tool[1],LH[60],MH[130],RH[60];

2.4.4 JumpL

功能: 直线跳跃指令, 中间段轨迹为直线

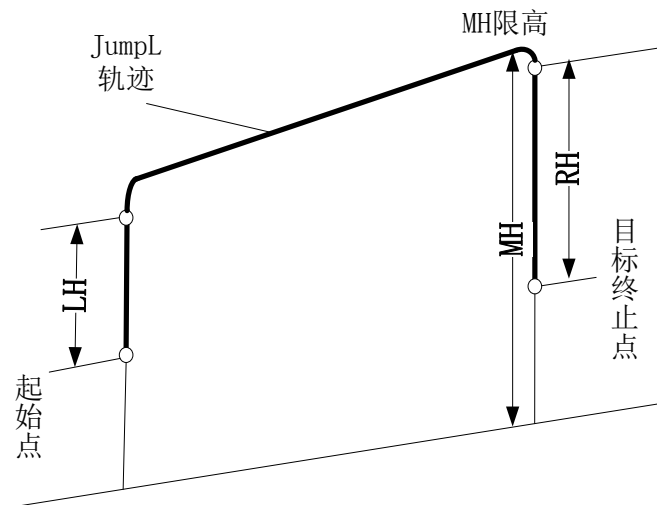
格式: JumpL <参数 1>,<参数 2>,<参数 3>,<参数 4>;

参数号	形式	意义
参数 1	P[***]	目标位姿。***为位置点标号, 可用数字直接表示, 也用变量 B/R/LB/LR 间接表示, 如 P[2]、P[B1]。P[***]取值范围详见 2.1.3 节位置变量。
	Offset(P[***],PR***)	以偏移的方式得到的目标点。Offset 表示在点 P[***]位置偏移 PR***后得到的新位置。也可使用 LPR***代替 PR***。PR***/LPR***取值范围详见 2.1.4 节平移变量。P[***]取 PE 时代表取当前点做偏移。
	OffsetJ(P[***],PR***)	关节坐标系下平移
	OffsetT(P[***],PR***)	工具坐标系方向上平移
	Pallet(托盘号,号行,号列号,层号)	根据(托盘号,行号,列号,层号)这些信息取托盘上的点, 详见 2.5.6 节 P=Pallet 指令。
参数 2	V[***]	指定机器人的运动速度。***是最大速度的百分比, 取 1-100 的整数。如 V[50]代表 50%最大限定速度。
参数 3	Z[***]	插补精度等级设置, 目前有 Z[0]~Z[5]六个等级, 数字越小精度越高。
参数 4 (LH、MH、RH 为必选参数, 其它为可选参数)	User[用户号]	选用某个的用户坐标系, 用户号可取 0~15。
	Tool[工具号]	选用某个的工具, 工具号可取 0~15。
	Acc[***]	指定机器人的运动速度。***是最大加速度的百分比, 取 1-100 的整数。如 Acc[50]代表 50%的最大加速度。
	OUT(IO 号,ON/OFF ,T[n])	并行 IO 输出指令。可重复使用, 同时多个 IO 触发。 T[n]为时间, 单位: 秒, 范围 -65535.000~65535.000。大于 0 表示开始运动 n 秒后输出信号, 小于 0 表示到达运动点之前 n 秒时输出信号。
	LH[***]	起始位置处的提升高度, 取值范围 0~2000。
MH[***]	运行过程中最高点相对于基坐标系零点的高度, 取值范围-2000~2000。	

	RH[***]	到终止位置的下降高度，取值范围 0~2000。
--	---------	-------------------------

相同参数参照 Movj。

JumpL 指令的运动过程分为三段：开始上升段 Movl，中间段 Movl 和最后下降段 Movl。



注意事项：

JumpL 指令仅用于 Scara 及 Delta 机器人。

若机器人的基坐标系零点设的高，MH 可能为负值。

在正常情况下，高度参数需满足 $MH > \text{初始点高度} + LH$ 且 $MH > \text{终止点高度} + RH$ 。对于不满足这个条件的非正常设定，应谨慎使用。

范例：

```
JumpL P[1],V[30],Z[3],User[1],Tool[1],LH[60],MH[130],RH[62];
```

2.4.5 Home

功能：回工作原点

格式：

```
Home [原点号];
```

```
Home [原点号],V[***];
```

说明：存在三个工作原点，原点号取<0~4>。V 为可选参数，代表速度的百分比，0~100。

范例：

```
Home[2],V[30];          ##以 30%最大速度回工作原点 2（即第三个工作原点）
```

2.4.6 Velset

功能：速度设置

格式：

```
Velset***;
```


VelsetRate[***];

VelsetOFF;

说明:

Velset***: 设置全局固定速度。该指令为模态指令, 一经设置, 便一直生效, 程序执行时会无视运动指令后面的V参数, 恒以最大速度的百分比, 直至遇到Velset OFF。***范围<1~100>。

VelsetRate[***]: 设置全局比率速度。该指令为模态指令, 一经设置, 便一直生效, 与其它运动指令的速度作乘法, 直至遇到Velset OFF。***范围<1~100>。

Velset OFF: 取消全局速度设置。

范例:

START;

Velset [30];

Movj P[1],V[40],Z[3]; ##实际速度 V[30]

Velset OFF;

Velset Rate [50];

Movj P[2],V[70],Z[3]; ##实际速度 V[35]

Velset OFF;

END;

注意: 当且仅当适用于位于代码上下文顺序VelSet <速度值>指令以下, VelSet OFF 以上中的运动指令, 和程序逻辑关系无关。

2.4.7 WaitInPos

功能: 等待执行完成

格式: WaitInPos;

参数: 无

注意: 预处理扫描到该行时会立即停止预处理, 等待该行之前的指令全部运行完成再继续预处理。使用这条指令会影响执行效率, 应尽量少用。

2.4.8 RefSys

功能: 切换坐标系。

格式 1: RefSysBase;

格式 2: RefSysConveyor(***,Tool[***]);

格式 3: RefSysWorkBench(***,Tool[***]);

形式	意义
RefSys Base	切换到机器人坐标系。从传送带或工作台坐标系切换回机器人坐标系后, 将停止运动。
RefSys Conveyor(***,Tool[***]);	切换到传送带坐标系。由于传送带是一个动态的坐标系, 因此执行此指令后, 工具末端会跟随传送带同步运动。 ***为传送带编号, 范围 0~3; Tool[***],为工具号, 表示以指定的工具末端同步传送带运动。

<code>RefSys WorkBench(***,Tool[***]);</code>	<p>切换到工作台坐标系。由于工作台是一个动态的坐标系，因此执行此指令后，工具末端会跟随工作台同步运动。</p> <p>***为工作台编号，范围 0~3； Tool[***],为工具号，表示以指定的工具末端同步传送带运动。</p>
---	---

说明：在使用外部运动机构，如传送带、工作台坐标系时，需要识别位置变量是处于哪个外部机构坐标系下，这时需使用 `RefSys` 切换到外部坐标系；当需要使用机器人坐标系下的位置变量时，需要使用 `Refsys` 切换回来。

注意：使用传送带跟随工艺时，对于指令 `PE`，`VelSet` 是无效的，在传送带跟随工艺过程中，所包含的运动指令中不得包含 `PE` 参数。全局指令速度设置指令 `VelSet` 也对其无效。

范例：

```

START;
CnvVision (Conveyor[1],ON,1025);    ##打开传送带 1 的视觉端口,客户端端口号 1025
P[30]=(0,0,10,0,0,0),(0,0,0,0),(7,0,0); ##定义 P[30]为在传送带物体的正上方 10mm 处
L[0]:
Movj P[0],V[30],Z[0];
GetCnvObject(1,0), Goto L[0];        ##接收 1 号传送带，0 号类型的物体的数据
RefSys Conveyor(1,Tool[2]);          ##使用工具 2 末端完成与传送带 1 的速度同步运动
Movl P[30],V[100],Z[1],Tool[2];     ##P[30]是在传送带 1 坐标系下的点
Set Out[1],ON,T[0];                 ##打开开关，吸附物体
Delay T[1];
RefSys Base;                          ##切换到机器人坐标系
Jump P[1],V[100],Z[0],LH[10],MH[-750],RH[10];    ##P[1]是在机器人坐标系下的点
Set Out[1],OFF,T[0];                 ##放置物体
Delay T[1];
Goto L[0];
CnvVision (Conveyor[1],OFF,1025);
END;

```

2.4.9 LockScrew

见 2.10.3 节

2.4.10 UnLockScrew

见 2.10.5 节

2.5 信号处理指令

2.5.1 Set

功能：设置输出信号

包含以下三类指令：

Set Out：设置单个数字输出信号（Out 信号）

Set OG：设置一组数字输出信号（一组含8个信号）

Set DA：设置模拟量输出信号

a) Set Out

功能：设置单个数字输出信号（Out 信号）

格式：Set Out[***],ON/OFF;

参数	含义
Out[***]	数字输出信号，Out[0]~Out[127]
ON/OFF	信号值

范例：

Set Out[1],ON; ##设置 Out[1]输出 ON

b) Set OG

功能：设置一组数字输出信号（一组含 8 个 Out 信号）

格式：Set OG[***],B***;

参数	含义
OG[***]	输出组信号，OG[0]~OG[15]；每组信号包含 8 个信号
B***	B 变量，取 B0~B255。代表一组信号状态

说明：

一组信号包含 8 个信号，OG[0]为 Out[0]~Out[15]，OG[1]为 Out[16]~Out[23]，以此类推……

B 变量的值，为<0~255>间的一个整数，正好为一个 8 位二进制的数，从右到左每一位代表从低位到高位的一个信号状态。

范例：

B1=7;

Set OG[0],B1; ##设置 OG[0]输出 0000 0111，即 Out[0]~Out[7]分别为 1110 0000。

c) Set DA

功能：设置模拟量输出信号

格式：Set DA[***],<值>;

参数	含义
DA[***]	模拟量输出信号，DA[0]~DA[15]
<值>	若该信号为电流，默认单位 mA 若该信号为电压，默认单位 V 根据 IRLink 模块的配置自动识别是电流还是电压。同时根据配置的电流或电压范围限制输入。

范例：

Set DA[1],1.2; ##若 DA[1]配置为电流类型，则此指令为输出 1.2 mA 的电流
 注意：在编辑指令时，当指令中的值超出了实际配置的范围时，会有限制或提示；而如果预先编好的程序中存在 IO 指令，随后修改了 IRLink 模块的配置，只会取实际配置的边界值。

2.5.2 Get

功能：读取输入/输出信号

包含以下 5 类指令：

Get In: 读取单个数字输入信号（In信号）

Get Out: 读取单个数字输出信号（Out信号）

Get IG: 读取一组数字输入信号（一组含8个In信号）

Get OG: 读取一组数字输出信号（一组含8个Out信号）

Get AD:读取单个模拟量输入信号

a) Get In

功能：读取单个数字输出信号（Out 信号）

格式：Get In [***],B***;

参数	含义
In[***]	数字输入信号，In[0]~In[127]
B***	B 变量，B0~B255。读取结果 0 为 OFF，1 为 ON

范例：

```
START;
Get In[7],B1;
Switch B1
  Case 0:
    Print "In[7] is OFF";
    Break;
  Case 1:
    Print "In[7] is ON";
    Break;
  Default:
    Print "Error";
EndSwitch;
END;
```

b) Get Out

功能：读取单个数字输出信号（Out 信号）

格式：Get Out [***],B***;

参数	含义
Out[***]	数字输出信号，Out[0]~Out[127]
B***	B 变量，B0~B255。读取结果 0 为 OFF，1 为 ON

范例：

```

START;
Get Out[0],B1;
Switch B1
  Case 0:
    Print "Out[0] is OFF";
    Break;
  Case 1:
    Print "Out[0] is ON";
    Break;
  Default:
    Print "Error";
EndSwitch;
END;

```

c) Get IG

功能：读取一组数字输入信号（一组含 8 个 In 信号）

格式：Get IG[***], B***;

参数	含义
IG[***]	输入信号组，范围 IG[0]~IG[15]，每组信号包含 8 个信号
B***	B 变量，B0~B255。代表一组信号状态

说明：

一组信号包含 8 个信号，IG[0]为 In[0]~In[15]，IG[1]为 In[16]~In[23]，以此类推……

B 变量的值，为<0~255>间的一个整数，正好为一个 8 位二进制的数，从右到左每一位代表从低位到高位的一个信号状态。

范例：

```
Get IG[1],B1;
```

```
## 读取 IG[1]的信号;
```

```
##若信号 In[16]~In[23]依次为 OFF、OFF、OFF、OFF、OFF、ON、ON、ON，则 B1=7
```

d) Get OG

功能：读取一组数字输出信号（一组含 8 个 Out 信号）

格式：Get OG[***], B***;

参数	含义
OG[***]	输入信号组，范围 OG[0]~OG[15]，每组信号包含 8 个信号
B***	B 变量，B0~B255。代表一组信号状态

说明：

一组信号包含 8 个信号，OG [0]为 Out [0]~Out [15]，OG [1]为 Out [16]~Out [23]，以此类推……

B 变量的值，为<0~255>间的一个整数，正好为一个 8 位二进制的数，从右到左每一位代表从低位到高位的一个信号状态。

范例：

```
Get OG [1],B1;
```

```
## 读取 OG [1]的信号;
```

```
##若信号 Out [16]~Out [23]依次为 OFF、OFF、OFF、OFF、OFF、ON、ON、ON，则 B1=7
```

e) Get AD

功能：读取模拟量输入信号

格式：Get AD[***],<值>;

参数	含义
DA[***]	模拟量输入信号，AD[0]~AD[15]
D***	D 变量，D0~D255。 若该信号为电流，默认单位 mA 若该信号为电压，默认单位 V 根据 IRLink 模块的配置自动识别是电流还是电压。

范例：

```
Get AD[1],D1;      ##获取 AD[1]的值。
```

2.5.3 Wait

功能：等待直至输入端口检测到指定的信号

格式：

```
Wait In[***] == ON/OFF,T[***];
```

```
Wait In[***] == ON/OFF,T[***], Goto L[***];
```

```
Wait Out[***] == ON/OFF,T[***];
```

```
Wait Out [***] == ON/OFF,T[***], Goto L[***];
```

参数	含义
In[输入端口号]	监测的输入端口,端口号范围<0~255>
Out[输出端口号]	监测的输出端口,端口号范围<0~255>
ON/OFF	收到的信号，ON 为高电平，OFF 为低电平
T[***]	等待时间，单位 s,范围<0.000~65535.000>
Goto L[***]	可选参数。 若使用该参数，超出时间而条件仍不满足时，则返回标签处； 不使用该参数，超出时间而条件仍不满足时，则会报警。

注意事项：特别地，T[0]表示等待时间为无限长，即会停止，一直等待输入信号，直到接受指定的输入信号。

范例：

```
Wait IN[6] == ON,T[10];
```

说明：等待，直到输入端口[6]的信号为 ON，才运行后面的指令；最多等待 10 秒，若 10S 后仍未满足条件，则报警。

2.5.4 Delay

功能：程序延时，时间由 T 参数控制

格式：Delay T[***];

时间单位 s，取值范围<0.000~65535.000>。由于其它条件限制，时间精度一般为 0.1s。

范例：

```
Delay T[5];      ##延时 5 秒
```

2.5.5Pulse

功能：输出对应 Out 口指定宽度的脉冲信号（输出 ON）

格式：Pulse Out[***], T[***];

参数：

参数	含义
Out[***]	输出端口,端口号范围<0~255>
T[***]	脉冲宽度, 范围<0.000~100.000s>

范例：

Pulse Out[5], T[2]; ##Out[5]输出高电平，持续 2 秒

注意：若当前 Out 已是 ON，则持续 2 秒为 ON，2 秒后为 OFF

2.5.6Invert

功能：信号取反

格式：Invert Out[***]; #信号取反输出

参数：Out 端口号范围<0,255>，也可为 B/LB 变量

范例：

Set Out[1],ON;

Invert Out[1]; ##Out[1]为 OFF

2.6 托盘指令

2.6.1 Msft

功能：计算两个位置变量间的平移量

格式：

PR*** = Msft (P[***],P[***]);

PR*** = Msft (P[***],P[***],Tcp);

说明：

无 Tcp 时，MSft 计算统一采用以第一个位置变量为基准，第二个位置变量转换到相同的坐标系下表达。然后对两点做平移计算。

有 Tcp 时，代表两个位置变量的当前工具坐标系的偏移。

对局部平移变量 LPR 依然适用。

范例：

PRO = Msft(P[1],P[2]); ##计算 P[1]运动到 P[2]间的平移变量，并赋值给 PRO

2.6.2 PR***=

功能：平移变量赋值。

格式：

PR*** = (X,Y,Z,A,B,C);

说明：该指令用于给平移变量赋值。X,Y,Z,A,B,C 可为数字或 B/R/D/LB/LR/LD 变量。对局部平移变量 LPR 依然适用。

范例 1:

PR1 = (110,120,130,10,50,60); ##直接赋值

范例 2:

LB1 = 110;

LR1 = 120;

LD1 = 130;

B1 = 10;

R1 = 50;

D1 = 60;

LPR1 = (LB1,LR1,LD1,B1,R1,D1); ##利用变量间接赋值

2.6.3 PR Sum

功能：两个平移变量加减运算

格式：PR*** = PR*** +/- PR***;

说明：两个平移变量加减运算，并赋值给另一个平移变量。对局部平移变量 LPR*** 依然适用。

范例：PR3 = PR1 - PR2;

2.6.4 P[***]=

功能：位置变量的赋值。

格式：

P[***]=(X,Y,Z,A,B,C),(ArmType[0],ArmType[1],ArmType[2],ArmType[3]),(坐标系号,工具号,用户号);

说明：

第一个括号内的参数为坐标值，当坐标系号取 1 时为(J1,J2,J3,J4,J5,J6)，坐标系号取 2、3、4 时为(X,Y,Z,A,B,C)。当坐标系号取 5 时，为固定相机坐标系下点的坐标。当坐标系号取 6 时，为随动相机坐标系下点的坐标。当坐标系号取 7 时，为跟随工艺点的坐标，此时点是动态点。

(ArmType[0], ArmType[1], ArmType[2], ArmType[3])为臂参数。

更多坐标系号、工具号、用户号的意义在参考 1.3 节。

范例：P[3] = (10,50,30,40,50,60),(1,-1,1,0)(4,1,1);

2.6.4 P=Offset

功能：基于点的偏移

格式：

P[***] = OffSetJ(P[***],PR***);

P[***] = OffSet(P[***],PR***);

P[***] = OffSetT(P[***],PR***);

说明：

OffsetJ：关节平移。表示在关节坐标系下偏移，PR 的数值为关节偏移角度 (J1,J2,J3,J4,J5,J6)。

Offset: 非关节平移。表示在空间位姿的偏移，PR 的数值为位姿变化量(X,Y,Z,A,B,C)。

OffsetT: 工具相对工具坐标系自身的偏移。工具末端在当前工具坐标系放下的平移,PR 的数值为位姿变化量(X,Y,Z,A,B,C)。

平移对局部平移变量 LPR 依然适用。

范例:

P[1] = (10,20,30,40,50,60),(1,-1,1,0)(1,0,0);

PR1=(5,0,0,0,0);

P[2] = OffsetJ(P[1],PR1); #P[2]结果为(15,20,30,40,50,60)

2.6.5 Pallet

功能: 设定简单托盘上的点, 与 P=Pallet 配合使用, 常用于码垛、搬运。

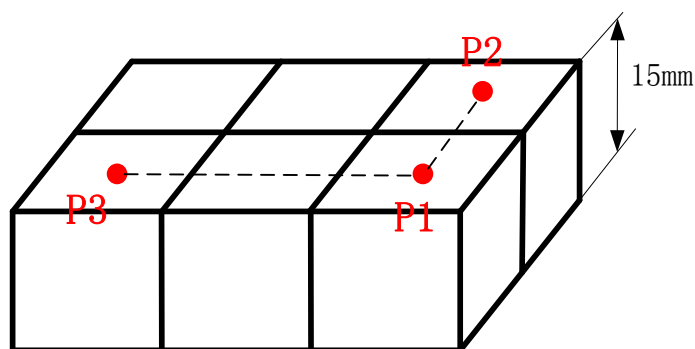
其原理是根据输入的三个点为依据创建托盘边界, 并根据行数、列数、层数、层高设定托盘模型, 以后则只需根据行、列、层的信息, 运动到托盘上的指定位置。

格式: Pallet 托盘号,P[i],P[j],P[k],行数,列数,层数,层高;

说明: 托盘号范围<0~255>。行数、列数、层数的取值范围为<0~255>, 层高单位 mm。

根据 P[i],P[j],P[k]数值构成平行四边形, 成为托盘的“边界点”。P[i]指定托盘上第一个点, P[j]与 P[i]的连线指定托盘的行方向, P[k]与 P[i]连线指定托盘的列方向。

范例: Pallet 1,P[1],P[2],P[3],2,3,1,15;



2.6.6 P=Pallet

功能: 取托盘上的点, 与 Pallet 配合使用, 常用于码垛、搬运。

格式: P[***]=Pallet(托盘号,行号,列号,层号);

说明: 根据托盘号选定托盘(托盘指令详见上一节 Pallet), 按行号, 列号, 层号运动到指定位置。注意行、列、层号是从 0 开始计数的。

范例:

P[***]=Pallet(1,1,1,0);

2.6.7 MovToPut

功能: 码垛指令, 运行至托盘上货物放置点

格式: MovToPut Pallet[***],P[***],X,Y,Z,V[***],PickV[***],Acc[***],Out(IO 号,ON/OFF ,

D[n]/T[n]/S[n]);

参数:

Pallet[***]: 托盘号, 范围 (0-255)

P[***]: 进入点, 范围 (0-9999)

X: 准备点相对于放置点的 X 方向偏移量。可用数字, 也可用 D/LD 变量。

Y: 准备点相对于放置点的 Y 方向偏移量。可用数字, 也可用 D/LD 变量。

Z: 准备点相对于放置点的 Z 方向偏移量。可用数字, 也可用 D/LD 变量。

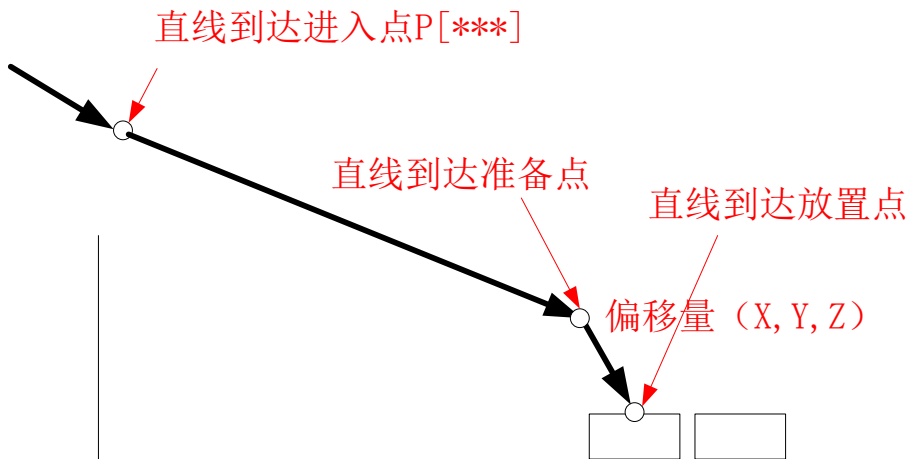
V[***]: 进入点到准备点的速度。***代表百分比, 可用数字, 也可用 B/LB 变量。

PickV [***]: 准备点到放置点的速度。***代表百分比, 可用数字, 也可用 B/LB 变量。

Acc[***]: 可选参数, 加速度百分比。

Out(IO 号,ON/OFF,D[n]/T[n]/S[n]): 可选参数, 并行 IO 输出指令。可重复使用, 同时多个 IO 触发: (运动过程指整个运动路径)

- T[n]为时间, 单位: 秒, 范围-65535.000~65535.000。大于 0 表示开始运动 n 秒后输出信号, 小于 0 表示到达运动点之前 n 秒时输出信号。
- D[n]为路径百分比, 范围 0.000-100.000。表示从开始运动到结束整个路径的 n%时输出信号。
- S[n]为距离, 单位: mm。范围-65535.000~65535.000。大于 0 表示从起点开始运动到 n 毫米之后时输出信号, 小于 0 表示运动到距终点 n 毫米之前时输出信号。



范例:

```
MovToPut Pallet[2],P[1],50,0,50,V[80],PickV[30];
```

2.6.8 MovToGet

功能: 拆垛指令, 运行至托盘上放置货物放置点

格式: MovToGet Pallet[***],P[***],X,Y,Z,V[***],PickV[***],Acc[***],Out(IO 号,ON/OFF , D[n]/T[n]/S[n]);

参数:

Pallet[***]: 托盘号, 范围 (0-255)

P[***]: 进入点, 范围 (0-9999)

X: 准备点相对于放置点的 X 方向偏移量。可用数字, 也可用 D/LD 变量。

Y: 准备点相对于放置点的 Y 方向偏移量。可用数字, 也可用 D/LD 变量。

Z: 准备点相对于放置点的 Z 方向偏移量。可用数字, 也可用 D/LD 变量。

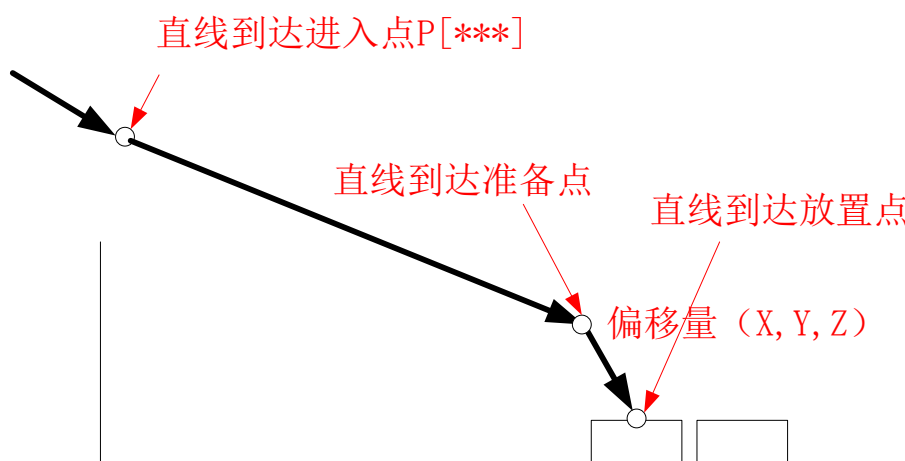
V[***]:进入点到准备点的速度。***代表百分比, 可用数字, 也可用 B/LB 变量。

PickV [***]:准备点到放置点的速度。***代表百分比, 可用数字, 也可用 B/LB 变量。

Acc[***]:可选参数, 加速度百分比。

Out(IO 号,ON/OFF,D[n]/T[n]/S[n]): 可选参数, 并行 IO 输出指令。可重复使用, 同时多个 IO 触发: (运动过程指整个运动路径)

- T[n]为时间, 单位: 秒, 范围-65535.000~65535.000。大于 0 表示开始运动 n 秒后输出信号, 小于 0 表示到达运动点之前 n 秒时输出信号。
- D[n]为路径百分比, 范围 0.000-100.000。表示从开始运动到结束整个路径的 n%时输出信号。
- S[n]为距离, 单位: mm。范围-65535.000~65535.000。大于 0 表示从起点开始运动到 n 毫米之后时输出信号, 小于 0 表示运动到距终点 n 毫米之前时输出信号。



范例: MovToGet Pallet[2],P[1],50,0,50,V[80],PickV[30];

2.6.9 MovFromPut

功能: 码垛指令, 返回至进入点

格式: MovFormPut Pallet[***],P[***],X,Y,Z,V[***],PickV[***],Acc[***],Out(IO 号,ON/OFF, D[n]/T[n]/S[n]);

参数:

Pallet[***]: 托盘号, 范围 (0-255)

P[***]: 进入点, 范围 (0-9999)

X: 准备点相对于放置点的 X 方向偏移量。可用数字, 也可用 D/LD 变量。

Y: 准备点相对于放置点的 Y 方向偏移量。可用数字, 也可用 D/LD 变量。

Z: 准备点相对于放置点的 Z 方向偏移量。可用数字, 也可用 D/LD 变量。

V[***]:准备点到进入点的速度。***代表百分比, 可用数字, 也可用 B/LB 变量。

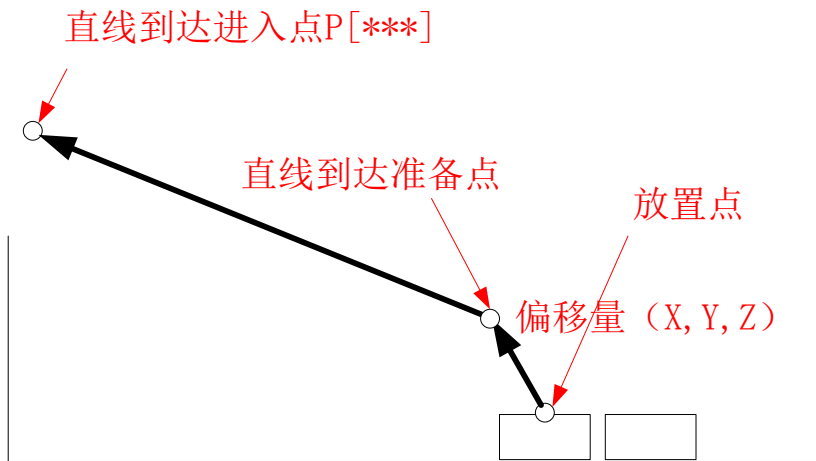
PickV [***]:放置点到准备点的速度。***代表百分比, 可用数字, 也可用 B/LB 变量。

Acc[***]:可选参数, 加速度百分比。

Out(IO 号,ON/OFF,D[n]/T[n]/S[n]): 可选参数, 并行 IO 输出指令。可重复使用, 同时多个 IO 触发: (运动过程指整个运动路径)

- T[n]为时间, 单位: 秒, 范围-65535.000~65535.000。大于 0 表示开始运动 n 秒后输出信号, 小于 0 表示到达运动点之前 n 秒时输出信号。

- D[n]为路径百分比，范围 0.000-100.000。表示从开始运动到结束整个路径的 n%时输出信号。
- S[n]为距离，单位：mm。范围-65535.000~65535.000。大于 0 表示从起点开始运动到 n 毫米之后时输出信号，小于 0 表示运动到距终点 n 毫米之前时输出信号。



范例：

```
MovFromPut Pallet[2],P[1],50,0,50,V[80],PickV[30];
```

2.6.10 MovFromGet

功能：拆垛指令，返回至进入点

格式：

```
MovFromGet Pallet[***],P[***],X,Y,Z,V[***],PickV[***],Tool[***],Acc[***],Out(IO 号,ON/OFF,
D[n]/T[n]/S[n]);
```

参数：

Pallet[***]：托盘号，范围（0-255）

P[***]：进入点，范围（0-9999）

X：准备点相对于放置点的 X 方向偏移量。可用数字，也可用 D/LD 变量。

Y：准备点相对于放置点的 Y 方向偏移量。可用数字，也可用 D/LD 变量。

Z：准备点相对于放置点的 Z 方向偏移量。可用数字，也可用 D/LD 变量。

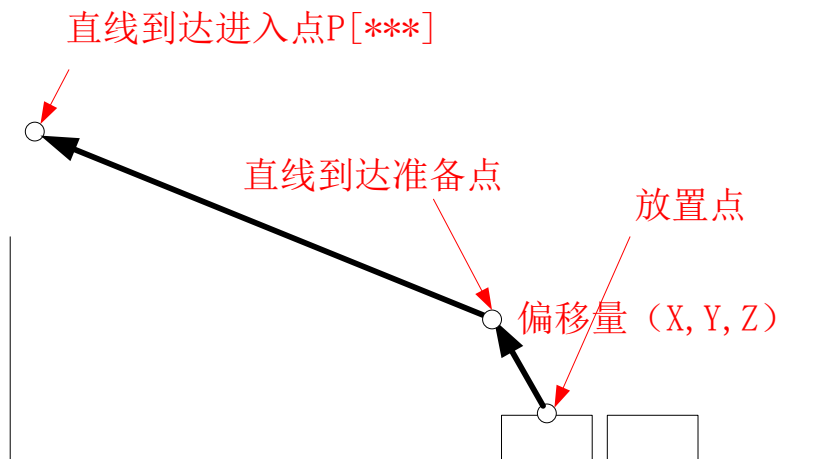
V[***]：准备点到进入点的速度。***代表百分比，可用数字，也可用 B/LB 变量。

PickV [***]：放置点到准备点的速度。***代表百分比，可用数字，也可用 B/LB 变量。

Acc[***]：可选参数，加速度百分比。

Out(IO 号,ON/OFF,D[n]/T[n]/S[n])：可选参数，并行 IO 输出指令。可重复使用，同时多个 IO 触发：（运动过程指整个运动路径）

- T[n]为时间，单位：秒，范围-65535.000~65535.000。大于 0 表示开始运动 n 秒后输出信号，小于 0 表示到达运动点之前 n 秒时输出信号。
- D[n]为路径百分比，范围 0.000-100.000。表示从开始运动到结束整个路径的 n%时输出信号。
- S[n]为距离，单位：mm。范围-65535.000~65535.000。大于 0 表示从起点开始运动到 n 毫米之后时输出信号，小于 0 表示运动到距终点 n 毫米之前时输出信号。



范例:

```
MovFromGet Pallet[2],P[1],50,0,50,V[80],PickV[30];
```

2.6.11 ResetPallet

功能: 初始化托盘

格式: ReSetPallet[***];

参数:

***: 托盘号, 范围 (0-255)

注意: 在使用其它码垛指令前, 必须先初始化托盘。

范例: ReSetPallet[1];

2.6.12 IsPalletFinished

功能: 查看托盘码垛或者拆垛是否完成

格式: IsPalletFinished(Pallet[***],B/LB***);

参数:

Pallet[***]: 托盘号, 范围 (0-255)

B/LB***:用于存储结果, 1 完成,0 未完成

范例: IsPalletFinished(Pallet[1],B1);

2.6.13 GetPalletRunNo

功能: 查看托盘码垛或者拆垛运行点号

格式: GetPalletRunNo(Pallet[***],R/LR***);

参数:

Pallet[***]: 托盘号, 范围 (0-255)

R/LR***:用于存储结果, 当前运行托盘点序号

范例: GetPalletRunNo(Pallet[1],R1);

2.6.14 SetPalletRunNo

功能：设置托盘码垛或者拆垛运行点号

格式：SetPalletRunNo(Pallet[***],数字/R/LR***);

参数：

Pallet[***]：托盘号，范围（0-255）

数字/R/LR***：要设置运行的托盘上点的序号，数字范围 0-999

注意：运动码垛或拆垛运动指令前，需先设置运行的托盘点序。

范例：

```
START;
```

```
ReSetPallet[1];
```

```
SetPalletRunNo(Pallet[1],0);    ##从序号为 0 的托盘点开始运行
```

```
For B0=0,B0<8,Step[1];
```

```
  MovToPut Pallet[1],P[1],50,0,50,V[80],PickV[30];
```

```
  MovFromPut Pallet[1],P[1],50,0,50,V[80],PickV[30];
```

```
EndFor;
```

```
End;
```

2.6.15 EOffsOn

功能：整体路径平移开启

格式：EOffsOn(X,Y,Z);

参数：X,Y,Z 为偏移的量

说明：

将运动的最终路径进行基坐标系下的 XYZ 方向偏移。与 EOffsOff 搭配使用，使之间的路径偏移。

案例：

```
Movj P[1],V[30],Z[0];
```

```
EOffsOn(10,0,0);
```

```
Movl P[2],V[30],Z[0];
```

```
Movl P[3],V[30],Z[0];
```

```
EOffsOff;
```

注意：P 变量坐标系序号为 7 时，不开启平移，（码垛机器人中不包含此指令） MovToPut、MovToGet、MovFromPut、MovFromGet 指令不开启平移。

2.6.16 EOffsOff

功能：整体路径平移关闭

格式：EOffsOff;

详见 EOffsOn。

2.7.流程控制指令

2.7.1 L-Goto

功能：L用于设置程序标签，常与跳转指令 Goto 配合使用，完成跳转动作

格式：

L[标号]: #不能重复

.....

Goto L[标签号];

范例：

START;

Movj P[0],V[30],Z[3];

L[1]: #设置标签 1

Movl P[1],V[30],Z[3];

Movl P[0],V[30],Z[3];

Goto L[1]; #跳转至标签 1

END;

说明：先运行至 P[0]位置，然后在 P[0]与 P[1]两点间往复运动。

2.7.2 If-Else-Endif

功能：条件判断

格式：

If <条件>

 语句 1;

Else

 语句 2;

Endif;

说明：如满足条件则执行语句 1，否则执行语句 2。其中<条件>内容可为变量与数字的大小判断、变量与变量的大小判断、输入信号的 ON/OFF 判断三种形式。

注意事项：

语句 1、2 即可为一行指令，也可为数行指令；Else 可缺省；Endif 作为段落的结束不可缺少。

范例 1:

START;

B0 = 1;

B1 = 2;

If B0>B1

 Movl P[1],V[50],Z[3];

Endif ;

End;

说明：程序中因不满足条件，不执行 Movl 运动

范例 2:

START;

```

If IN[1]==OFF;
    Movj P[1],V[50],Z[3];
Else
    Movj P[2],V[50],Z[3];
    Movj P[3],V[50],Z[3];
EndIf;
End;

```

说明：如满足条件则执行语句 1，否则执行语句 2。其中<条件>内容可为变量与数字的大小判断、变量与变量的大小判断、输入信号的 ON/OFF 判断三种形式。可由两个或者多个条件表达式通过 AND 或者 OR 组合。IF 指令为非预处理指令。

2.7.3 Switch-Case-Default-EndSwitch

功能：条件选择语句,根据 B/LB/R/LR 变量的值选择不同的分支执行语句。

格式：

```

Switch 变量
    Case <变量的值 1>:
        <语句>;
        Break;
    Case <变量的值 2>:
        <语句>;
        Break;
    Default:
        <语句>;
        Break;

```

EndSwitch;

说明：根据 B/LB/R/LR 变量的值选择性执行语句，若所有 Case 均不符合,则执行缺省(Default)后的语句。

注意事项：

一般情况下，每个 Case（包括 Default）段最好以 break 结尾；若某个 Case 段结尾无 Break，则不跳出，继续往后执行下个 Case 段内容，至 break 为止。Default 段落语句也可以省略。若整个条件选择段落以 Default 结尾，则 Default 内容结束后必须跟有 Break；若整个段落不使用 Default，即仅使用 Case,则最后一个 Case 内容结束后必须跟有 Break。

范例：

```

Switch B0
    Case 1:
        Movj P[1],V[50],Z[3];
        Break;
    Case 2:
        Movj P[1],V[50],Z[3];
        Movj P[2],V[50],Z[3];
        Break;
    Case 3:
        Movj P[3],V[50],Z[3];

```



```
        Break;
Default:
        Movj P[4],V[50],Z[3];
        Break;
EndSwitch;
```

2.7.4 While-EndWhile

功能:指定条件的循环

格式:

```
While <条件>
    <语句 1>;
    .....;
EndWhile;
```

说明: 条件循环语句, 若满足条件, 则执行 While 与 EndWhile 之间的语句, 完成后再转入条件判断, 往复循环, 直到不满足条件时跳出。一般的, <条件>的格式为“左操作数操作符右操作数”, 左操作数取 B/R/LB/LR 变量, 右操作数取另一个变量或数字, 如“LB[0]<3”。

范例:

```
START;
Movj P[1],V[50],Z[3];
While LB[0]<3
    Movj P[2],V[50],Z[3];
    Movj P[1],V[50],Z[3];
    Incr LB[0];
EndWhile;
```

End;

说明: 上述运行指令, 循环次数为 3, 整段程序在 P[1]至 P[2]运动来回运动三次。

2.7.5 For-EndFor

功能: 带执行次数的循环语句

格式:

```
For <赋值表达式>,<条件表达式>,Step[步长]
    <语句>;
EndFor;
```

说明: 先执行<赋值表达式>, 再判断<条件表达式>, 若满足条件则执行 For 与 EndFor 之间的内容, 执行完成一次后, 执行“Step[步长]”, 赋值表表达式中定义的变量自增, 再判断<条件表达式>, 若满足则继续刚才的步骤, 直至<条件表达式>不成立时跳出。赋值表达式指 B/R/LB/LR 变量的赋初始值, 条件表达式指对应 B/R/LB/LR 变量的条件判断表达式, 步长指对应 B/R/LB/LR 变量每一次运行的增量, 增量取值范围为-65536~65535 以内的整数。

范例:

```
For B0=0,B0<5,Step[2]
    Movj P[2],V[50],Z[3];
    Movj P[3],V[50],Z[3];
```

EndFor;

说明：两个 Movj 指令循环执行 3 次

2.7.6 Break

功能：跳出语句。用于跳出 While 或 For 循环，此外还用于在 Switch 语句中执行 Case 段后跳出。多个循环嵌套时，使用 Break 会只跳出当前循环。

格式：Break;

2.7.7 Continue

功能：碰到 Continue 语句跳出 While 或 For 循环,继续执行下一个循环。多个循环嵌套时，使用 Continue 会只跳出当前循环。

格式：Continue;

2.7.8 Call

功能：调用子程序

格式：Call “子程序”;

注意事项：常与 Ret 配合使用，详见下 Ret 指令

范例：

```
Call “abc.pro”;      ##当前目录下的 abc.pro 程序
```

```
Call “Test/ff.pro”;  ##调用 Test 文件夹下的 ff.pro 程序
```

2.7.9 Ret

功能：返回母程序，执行该次调用指令后面的语句

格式：Ret;

注意事项：常与 Call 陪使用。若子程序中无 Ret 指令，则在子程序中遇到 End 母程序也结束；若子程序中存在 Ret，则不再运行子程序 Ret 后的指令，跳回母程序并执行该次 Call 后面的语句。

范例：

存在母程序 parents.pro 和子程序 child.pro

parents. pro

```
START;  
Movj P[1],V[30],Z[0];  
Movj P[6],V[30],Z[0];  
Call "children.pro";  
Movj P[7],V[30],Z[0];  
END;
```

child. pro

```
START;  
Movj P[0],V[30],Z[0];  
Movj P[1],V[30],Z[0];  
Ret;  
Movj P[2],V[30],Z[0];  
END;
```

上述子程序 child.pro 只运行了前两条 Mov 指令，便返回至母程序。整个运动过程为：

母程序P[1] → 母程序P[6] → 子程序P[0] → 子程序P[1] → 母程序P[7]

2.7.10 Pause

功能：暂停运行

格式：Pause;

相当于示教器上的暂停按钮功能，按运行键可以继续往下执行。该指令常用于调试时查看变量。

2.7.11 LoadPointFromFile

功能：从点数据文件中加载点到系统中

格式：LoadPointFromFile(文件路径名,R/LR 变量);

参数：

文件路径名：点数据文件路径名

变量：用于返回加载点的个数

说明：常与指令 GetAPointFromFile 配合使用，用来从外部的点文件中获取位置变量。常见的“点文件”使用流程如下：

“点文件”的使用流程



案例：从点文件 err.pt 中加载点信息，并取前三个点依次赋值给 P[0]、P[1]、P[2]。

```
001 START;
002 LR0 =0;
003 LoadPointFromFile("eer.pt",LR0);
004 GetAPointFromFile("eer.pt",0,P[0]);
005 GetAPointFromFile("eer.pt",1,P[1]);
006 GetAPointFromFile("eer.pt",2,P[0]);
007 Movj P[0],V[30],Z[0];
008 Movj P[1],V[30],Z[0];
009 Movj P[2],V[30],Z[0];
010 END;
```

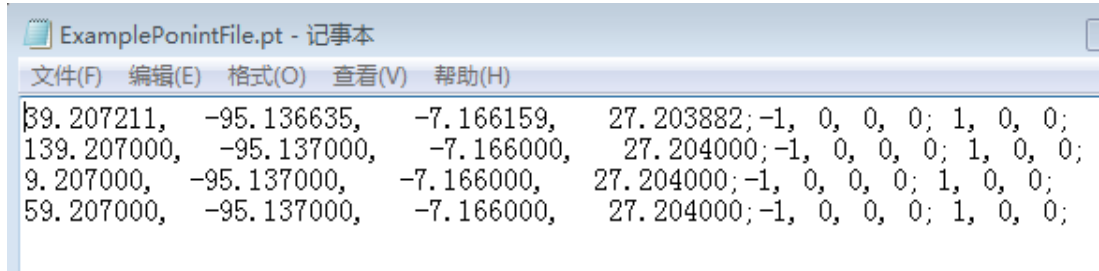
注意事项：

如果程序中没有位置变量被预先定义，（如上程序中没有 P[0]、P[1]、P[2]），使用指令“GetAPointFromFile”则会报错。

关于点文件:

点文件以“.pt”为后缀名。文件内容为位置变量的数据信息，一行代表一条位置变量信息。每行的格式参照“位置变量”的定义。每行分为3小段，前4-6个参数为第一段，机器人的坐标系值；中间四个参数为第二段，臂参数；后三个参数为第三段，分别为坐标系号、工具号、用户号。段与段之间以“;”分隔，段内数字以“,”分隔，以“;”为结尾。

一个示例如下所示:



```
ExamplePonintFile.pt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
39.207211, -95.136635, -7.166159, 27.203882;-1, 0, 0, 0; 1, 0, 0;
139.207000, -95.137000, -7.166000, 27.204000;-1, 0, 0, 0; 1, 0, 0;
9.207000, -95.137000, -7.166000, 27.204000;-1, 0, 0, 0; 1, 0, 0;
59.207000, -95.137000, -7.166000, 27.204000;-1, 0, 0, 0; 1, 0, 0;
```

2.7.12 GetAPointFromFile

功能: 从系统中加载单个点到本程序的位置变量

格式: `GetAPointFromFile(文件路径名,R/LR 变量/数字, P[**]);`

参数:

文件路径名: 点数据文件路径名

R/LR 变量/数字: 要取的点在文件中的序号

P[**]: 用户保存需要加载的点

说明: 参考 `LoadPointFromFile` 的说明。

注意事项:

如果程序中没有位置变量被预先定义, (如上程序中没有 `P[0]`、`P[1]`、`P[2]`), 使用指令“`GetAPointFromFile`”则会报错。

2.8 信息交互指令

2.8.1 Alarm

功能: 输出报警信息

格式: `Alarm [报警序号];`

有效序号范围<1~15>, 对应用户自定义的报警信息, 结果会在窗口消息栏显示。

范例: `Alarm [2];`

2.8.2 Print

功能: 输出打印信息

格式: `Print (打印信息);`

打印信息包括一串需要打印的变量以及字符串，中间以“+”分隔。打印结果会在窗口消息栏显示。平移变量只打印前 6 个坐标值数据。

范例：

START;

B0=7;

P[2]

Print B1+P[2]+LR[1];

2.8.3 GetPlcVar

功能：获取 PLC 传输的变量信息

（这里使用了四种 PLC 变量类型：Byte、Int、DInt、LReal，参照“ITC61131-3”标准）

a) GetPlcVarByte

功能：获得 PLC Byte 类型变量的值

格式：B/LB*** = GetPlcVarByte(变量号);

参数	含义
B/LB***	作返回值，PLC 变量的值赋给该参数
变量号	PLC Byte 类型变量号，范围<0~255>，可为数字或 B/R/LB/LR 变量

范例：

B1= GetPlcVarByte(3);

b) GetPlcVarInt

功能：获得 PLC Int 类型变量的值

格式：R/LR*** = GetPlcVarInt(变量号);

参数	含义
R/LR***	作返回值，PLC 变量的值赋给该参数
变量号	PLC Int 类型变量号，范围<0~255>，可为数字或 B/R/LB/LR 变量

范例：

R1= GetPlcVarInt(3);

c) GetPlcVarDInt

功能：获得 PLC DInt 类型变量的值

格式：R/LR*** = GetPlcVarDInt(变量号);

参数	含义
R/LR***	作返回值，PLC 变量的值赋给该参数
变量号	PLC DInt 类型变量号，范围<0~255>，可为数字或 B/R/LB/LR 变量

范例：

R1= GetPlcVarDInt(3);

d) GetPlcVarLReal

功能：获得 PLC LReal 类型变量的值

格式：D/LD*** = GetPlcVarLReal(变量号);

参数	含义
D/LD***	作返回值，PLC 变量的值赋给该参数
变量号	PLC LReal 类型变量号，范围<0~255>，可为数字或 B/R/LB/LR 变量

范例：

D1= GetPlcVarLReal(3.2);

2.8.4 注释

功能：输入注释

格式：## XXXXX

范例：##Program is maded at 10.

2.7.5 TimeStart

功能：启动计时器

格式：TimeStart(计时器序号);

参数：计时器序号范围 0-4。

2.8.6 TimeOut

功能：输出计时时长

格式：TimeOut(计时器序号, D/LD 变量);

参数：

计时器序号范围 0-4。

D/LD 变量为计时器时长，单位 ms。

注意：

1. 计时器开启后，数值一直累积，除非重新开启；
2. 没有开启的计时器能够被编译，但数值无效。

范例：

START;

Movj P[0],V[30],Z[0];

TimeStart(1); ##计时器 1 开启

Movj P[1],V[30],Z[0];

Timeout(1,D2); ##计时器 1 读取

Movj P[2],V[30],Z[0];

Timeout(1,D3); ##计时器 1 读取

END;

2.8.7 GetModBusCoil

功能：获取 ModBus 从站线圈的值

格式：GetModBusCoil(起始地址,线圈个数, B 变量/LB 变量);

说明：读取一个或多个 ModBus 从站线圈的值，按顺序从低位到高位排布后，存储到 B/LB

变量中。

参数	含义
起始地址	线圈的起始地址,以 bit 为单位,范围取(2048-4095)以及(6144-8191)
线圈个数	读取的线圈个数,范围 1-8
B/LB***	用于存储线圈的值。

说明:

B/LB 变量是 BYTE 类型,占用 8bit,可容纳最多 8 个线圈值。

从低位到高位存储,如起始地址的线圈值存储到变量的 bit0,“起始地址+1”的线圈值储到 bit1,依次类推……

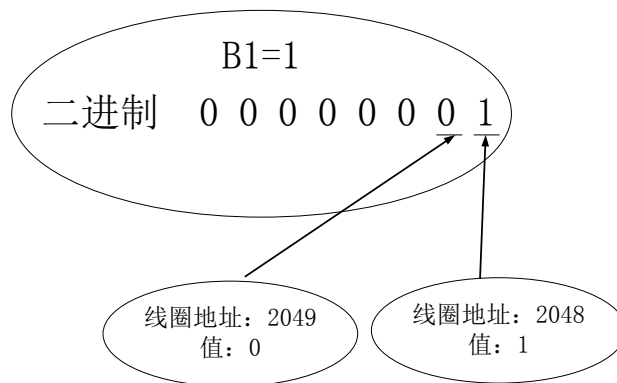
当少于 8 个线圈时,多余线圈值不会读取,对应 bit 位上值取 0。

线圈地址的有效范围(2048-4095)以及(6144-8191),出界的线圈会产生错误。

范例:

任务:读取线圈地址 2048、2049 中的值

编程: `GetModBusCoil(2048,2,B1);` #读取从地址 2048 开始的线圈,读取 2 个,存入 B1



2.8.8SetModBusCoil

功能:设置 ModBus 从站线圈的值

格式: `SetModBusCoil(起始地址,线圈个数,B 变量/LB 变量);`

说明:将 B/LB 变量的值,按位从低位到高位,设置到一个或多个线圈中。

参数	含义
起始地址	线圈的起始地址,以 bit 为单位,范围取(2048-4095)以及(6144-8191)
线圈个数	写入的线圈个数,范围 1-8
B/LB***	要写入线圈的值

说明:

B/LB 变量是 BYTE 类型,占用 8bit,能设置最多 8 个线圈。

变量从低位到高位设置,如 bit0 设置到“起始地址”的线圈 0, bit1 设置到“起始地址+1”的线圈,依次类推……

当设置的线圈个数少于 8 个时,多余 bit 位的值不会下发至线圈。

线圈地址的有效范围(2048-4095)以及(6144-8191),出界的线圈会产生错误。

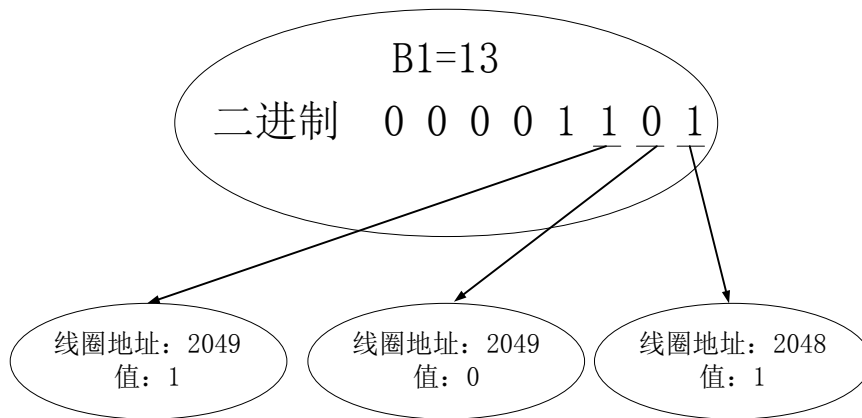
范例:

任务:设置线圈地址 2048、2049、2050 中的值依次为 1、0、1。

编程:

B1=13;

SetModBusCoil(2048,2, B1); #将 B1 值设到地址为 2048、2049、2050 的三个线圈中



##本例中 B1=5 也可以，以 B1=13 说明多余位不会下发

2.8.9 GetModBusReg

功能: 读取 ModBus 从站寄存器的值。

格式:

GetModBusReg(参数 1, 参数 2, 参数 3);

说明: 读取寄存器的值到变量中。

参数:

参数	形式 1	含义
参数 1	起始地址	寄存器的起始位置, 以 WORD 为单位, 范围(16384-32767)以及 (49152- 65535)
参数 2	整数/R/LR***	获取存储在寄存器中的值
参数 3	1 或 2	表示读 1 或 2 个寄存器

参数	形式 2	含义
参数 1	起始地址	寄存器的起始位置, 以 WORD 为单位, 范围(16384-32767)以及 (49152- 65535)
参数 2	小数/D/LD***	获取存储在寄存器中的值
参数 3	4	表示读 4 个寄存器

说明:

(1) 整数/R/LR 变量为 int32 类型, 占用 2 个寄存器的位置。当参数 2 为整数/R/LR 变量时, 参数 3 只能为 1 或者为 2。当参数 3 为 2 时, 从低位到高位读取 2 个寄存器的值存入 R/LR 变量中。当参数 3 为 1 时, 以 int16 类型读取一个寄存器的值再转成 int32 类型存入 R/LR 变量中。

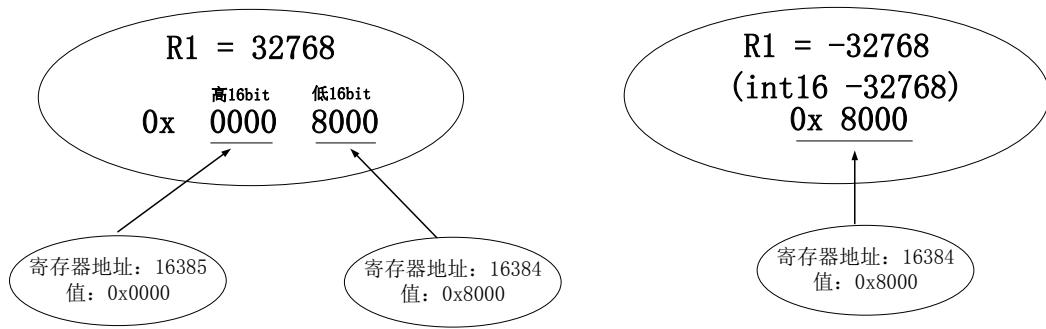
范例:

假设寄存器地 16384 中值为 0x8000、寄存器地址 16385 中的值为 0x0000。

编程:

GetModBusReg(16384, R1,2); ##读取从地址 16384 开始的 2 个寄存器的值, 值存入 R1。

GetModBusReg(16384, R1,1); ##读取从地址 16384 开始的 1 个寄存器的值，值存入 R1。
 说明：结果如下，注意读取的结果是不同的。



(2) D/LD 变量为 double 类型，占用 4 个寄存器的位置。当参数 2 为小数/D/LD 变量时，参数 3 只能为 4，表示由从低位到高位读取 4 个寄存器的值，存入 D/LD 变量中。

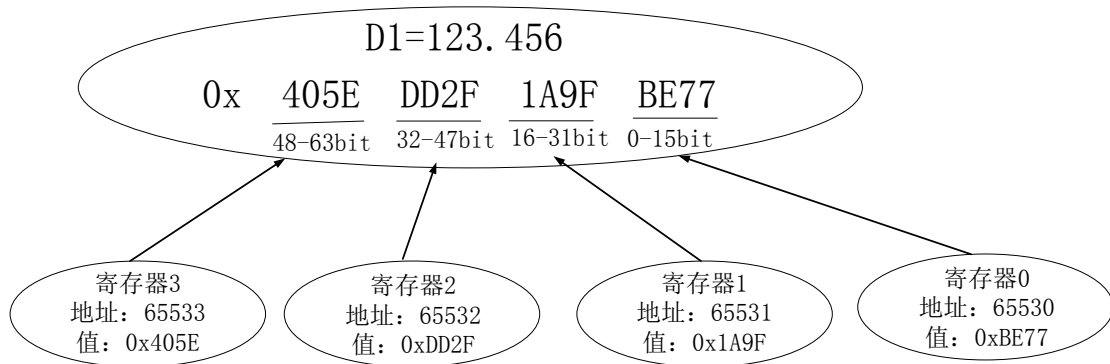
范例：

假设地址 65530~65533 寄存器地址中值分别为 0xBE77、0x1A9F、0xDD2F、0x405E。

编程：

GetModBusReg(65530, D1,4); ##读取从地址 16384 开始的 4 个寄存器，值存入 D1。

说明：结果如下。



2.8.10 SetModBusReg

功能：设置 ModBus 从站寄存器的值

格式：SetModBusReg(参数 1,参数 2,参数 3);

说明：设置 R/LR 变量的值到 1 或者 2 个寄存器中，或设置 D/LD 变量的值到 4 个寄存器中。

参数	形式 1	含义
参数 1	起始地址	寄存器的起始位置，以 WORD 为单位，范围(16384-32767)以及 (49152- 65535)
参数 2	R/LR***/整数	要设到 1~2 个寄存器中的值
参数 3	1 或 2	表示要写入的寄存器的个数

参数	形式 2	含义
参数 1	起始地址	寄存器的起始位置，以 WORD 为单位，范围(16384-32767)以及 (49152- 65535)
	D/LD***/小数	要设到 4 个寄存器中的值
参数 3	4	表示要写入的寄存器的个数

说明:

(1) 这里的整数和 R/LR 变量是 int32 类型, 占用 32bit, 其值若全部存储, 应存放到 2 个寄存器中。

当参数 3 取 2 时, 表示这个数存到两个寄存器中。数的 0-15bit 存放到寄存器 0 中, 16-31bit 存放到寄存器 1 中。

当参数 3 取 1 时, 表示这个数存到一个寄存器中。这个 int32 类型的数强制转成 int16 类型, 再存储到寄存器中。

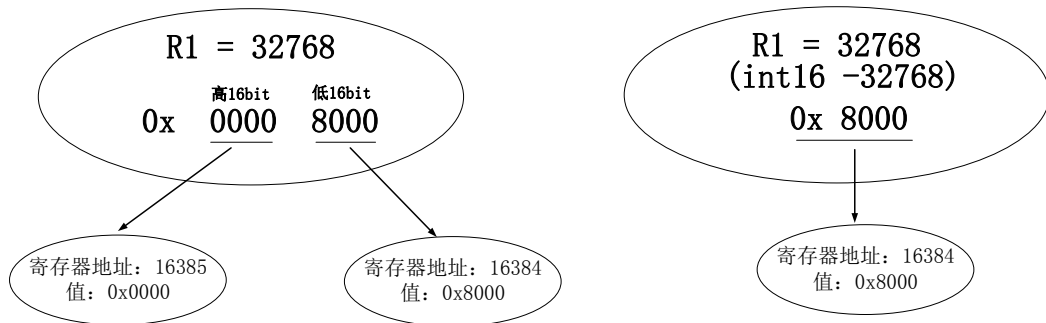
范例:

设置将 R1 值分别设置到地址为 16384、16385 两个寄存器中和 16384 一个寄存器中。

编程:

```
SetModBusReg(16384, R1,2);
```

```
SetModBusReg(16384, R1,1);
```



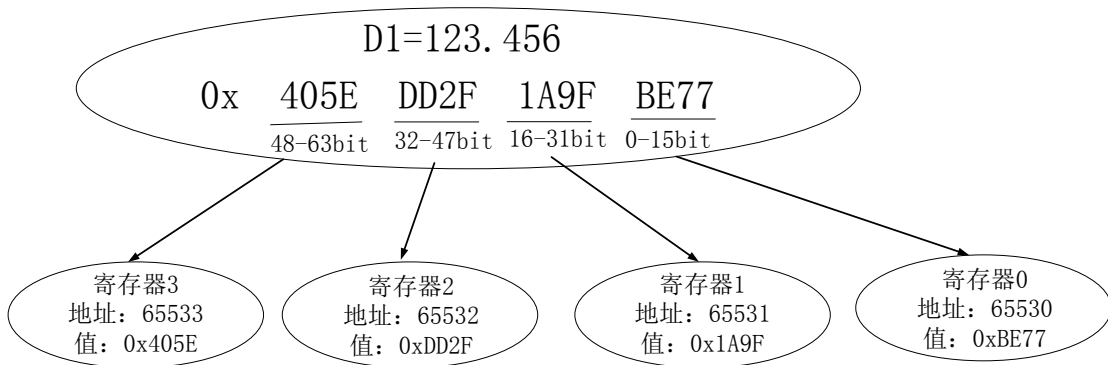
(2) 这里的小数和 D/LD 变量是 double 类型, 占用 64bit, 其值存放到 4 个寄存器中。其中, 0-15bit 存放到寄存器 0 中, 16-31bit 存放到寄存器 1 中, 32-47bit 存放到寄存器 2 中, 48-63bit 存放到寄存器 3 中。寄存器地址的有效范围(16384-32767)以及 (49152- 65535), 出界的寄存器会产生错误。

范例:

```
D1=123.456;
```

任务: 设置将 D1 值设置到地址为 65530~65533 的寄存器中。

编程: SetModBusReg(65530, D1); ##当 D1 值为 123.456 时, 则结果如下。



2.8.11 GetCommVal

功能：读取共享内存公共区的值

格式 1: GetCommVal(偏移地址, &变量, 个数);

说明：从偏移地址开始，读取一定个数变量的内存，传给变量

参数	形式	含义
参数 1	偏移地址	公共区相对于起始地址的偏移量，以 BYTE 为单位，范围(0-8191)
参数 2	&变量	B/LB/R/LR/D/LD/P/PR/LPR/字符串变量数组的起始项，用于存储读到的公共区的值
参数 3	个数	读取变量的个数，0~255

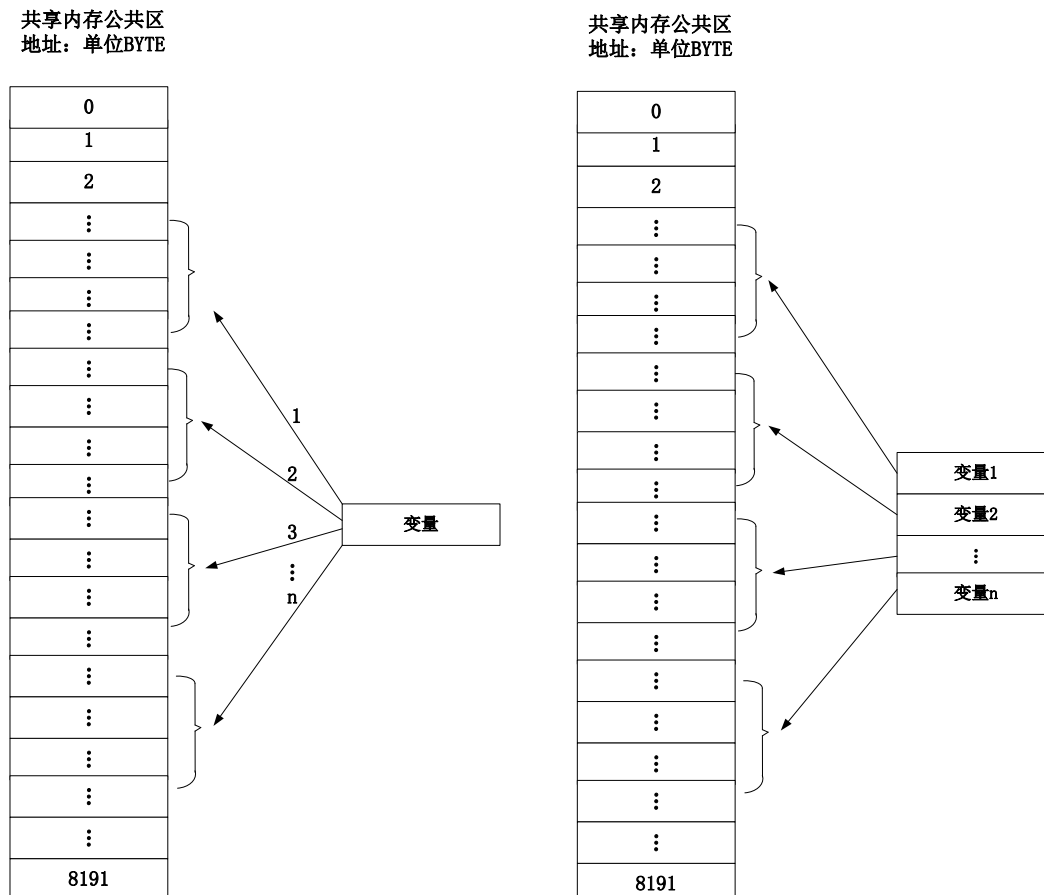
说明：关于 SetCommVal 与 GetCommVal

SetCommVal、GetCommVal 都是操作共享内存公共区，前者负责写，后者负责读。他们配合使用，通常用来操作用户自定义的参数。

(1)SetCommVal 有两种形式，一种是将值重复的设到公共区中，标志是参数 2 不带符号“&”；另一种是将变量数组依次设置到公共区中，标志是参数 2 带有符号“&”。如下图所示。而对于 GetCommVal，有参数 2 带有“&”的情况，会依次读取公共区的值。

SetCommVal (偏移地址, 数值/变量, 个数n);

SetCommVal (偏移地址, &变量, 个数n);



(2) 变量的类型不同，会占据公共区地址大小也不同。详见下图“变量占据地址大小说明表”。使用时，需要避免公共区地址的重复赋值，且不出界。

对象	类型	占用地址
B/LB 变量	BYTE	1 BYTE
R/LR 变量/数值	Int	4 BYTE
D/LD 变量	double	8 BYTE
P 变量/ PR/LPR 变量	结构体: int Coord;//坐标系号 int ToolNo;//工具号 int UserNo;//用户坐标系号 int ArmParm[4];//臂参数 double PosData[8];//坐标值	92 BYTE
字符串	参数 3 “个数” 以 BYTE 为单位	

范例：

```
R1=45;
SetCommVal (8000,R1, 2);
GetCommVal (8000, LR1, 2); ##结果 LR1=45, LR2=45
R1=46;
R2=47;
SetCommVal (8000,&R1, 2);
GetCommVal (8000, LR1, 2); ##结果 LR1=46, LR2=47
CString str1 = "abc";
SetCommVal (500,str1, 3);
GetCommVal (500, &str1, 2); ##结果 str1="ab"
```

2.8.12 SetCommVal

功能：设置公共区指定地址的值

格式：

SetCommVal (参数 1,参数 2,参数 3);

参数：

参数	形式	意义
参数 1	偏移地址	公共区相对于起始地址的偏移量，范围(0-8191)
参数 2	数值/变量	将值重复设置到公共区中,适用于数值或 B/LB/R/LR/D/LD/P/PR/LPR/字符串变量。使用数值时，数值范围-2147483648~2147483647。
	&变量	将变量数组依次设置到公共区中，适用于 B /R/D/PR 变量
参数 3	个数	变量个数，0~255

说明：

详情参见 GetCommVal 说明【关于 SetCommVal 与 GetCommVal】。

当参数 2 使用字符串变量时，把字符串变量中的内容按照字符串的存储格式存到公共区，当字符串变量中字符的个数小于字符个数时，以 ‘\0’ 写入。

2.9 视觉指令

视觉指令是控制器与外部视觉设备的通讯所用的指令，根据使用情况分为 2 种形式。

(1) 本地控制器作服务器，外部设备作客户端：

机器人控制器作为服务器，默认系统启动就打开，程序中不需要使用 Open Socket 指令打开；但需设置一个的端口号，并在视觉设备上连接按这个端口设置连接。这项设置在【设置】-【系统设置】-【通讯设置】中设定，可见见 3.2.6 (a) 节。

(2) 本地控制器作客户端，外部设备作服务器：

机器人控制器作客户端，需要在程序中利用 Open Socket 指令选择连接的 IP。

机器人作为客户端，指令中设置本地端口号1026
外部视觉作服务器，【通讯设置】设置服务器端口号1025

```
START;
PRO=(0,0,10,0,0,0);
TxBuf = "TA";
RxBuf = "";
L[2];
Movj P[0],V[30],Z[0];
L[0];
Open Socket("10.44.53.13",1025,1026,B0);
If B0 == 0
    Goto L[0];
Endif;
SetPorBuf(TxBuf);
Send Port[1026];
L[1];
Get Port[1026], T[10], Goto L[1];
RxBuf = GetPorbuf(0,100);
B1 = StrGetData(RxBuf,"#",P10);
Cnvt(P[10],P[20],World);
Movl Offset(P[20],PRO),V[30],Z[0];
Set Out[1],ON,T[0];
Delay T[1];
Jump P[1],V[100],Z[0],LH[10],MH[-750],RH[10];
Set Out[1],OFF,T[0];
Delay T[1];
Goto L[2];
Close Socket,1026;
END;
```

机器人作服务器，【通讯设置】设置服务器端口号1025
外部视觉作客户端，端口号1026

```
START;
TxBuf = "TA";
RxBuf = "";
L[2];
Movj P[0],V[30],Z[0];
SetPorBuf(TxBuf);
Send Port[1026];
L[1];
Get Port[1026], T[10], Goto L[1];
RxBuf = GetPorbuf(0,100);
B1 = StrGetData(RxBuf,"#",P10);
Cnvt(P[10],P[20],World);
Movl Offset(P[20],PRO),V[30],Z[0];
Set Out[1],ON,T[0];
Delay T[1];
Jump P[1],V[100],Z[0],LH[10],MH[-750],RH[10];
Set Out[1],OFF,T[0];
Delay T[1];
Goto L[2];
END;
```

(打开)

发送请求

接收数据

处理接收数据
并控制运动

(关闭)

对于跟随工艺，可使用 CnvVision 指令取用传送带坐标系，视觉数据自动传送到控制器并处理，指令流程为“Open ->CnvVision(ON) ->GetCnvObject ->Movl P->CnvVision(OFF)->Close”。一个简单的编程案例如下：

START;

L[0]:

##打开端口。外部设备作服务器地址 10.44.53.13，端口号 1025；

##本地控制器作客户端，端口号 1026。

Open Socket("10.44.53.13",1025,1026,B0);

If B0 == 0

Goto L[0];

CnvVision (Conveyor[1],ON);

P[30]=(0,0,10,0,0,0),(0,0,0,0),(7,0,1); ##定义 P[30]为在 1 号传送带物体的正上方 10mm 处

L[1]:

Movj P[0],V[30],Z[0];

```

GetCnvObject(1,0), Goto L[1];      ##接收 1 号传送带, 0 号类型的物体的数据
RefSys Conveyor[1];              ##取用传送带 1 坐标系
Movl P[30],V[100],Z[1];          ##运动到 P[30]
Set Out[1],ON;                   ##打开开关, 吸附物体
Delay T[1];
RefSys Base;                      ##切换到机器人坐标系
Jump P[1],V[100],Z[0],LH[10],MH[-750],RH[10];    ##将物体移动至 P[1]处
Set Out[1],OFF,T[0];             ##放置物体
Delay T[1];
Goto L[1];
CnvVision (Conveyor[1],OFF);
Close Socket,1026;
END;

```

2.9.1 Open

功能: 打开以太网端口 (仅用于本地控制器作客户端, 外部作服务器)

格式:

Open Socket(IP 地址,服务器端口号,客户端端口号);

Open Socket(IP 地址,服务器端口号,客户端端口号,B/LB***);

参数	含义
IP 地址	如 192.168.2.5
服务器端口号	范围<1024~9999>.其中 3333 不可使用
客户端端口号	范围<1024~9999>。其中 3333 不可使用
B/LB*** (可选参数)	返回值, 打开成功返回 1, 否则为 0

注意事项:

Open Socket 用于本地控制器作客户端时与外部通讯使用。

客户端与服务器不得使用相同的端口号

Open Socket 一旦启动, 端口会一直打开; 不会随程序关闭而关闭, 除非使用 close。

使用时, 一般利用循环语句, 循环执行, 对返回值进行判断, 在打开成功后跳出。

范例:

```
LB1 = 0;
```

```
While LB1<>1
```

```
Open Socket(192.168.2.5,1025,1026,LB1);
```

```
EndWhile;
```

2.9.2 Close

功能: 关闭以太网端口

格式: Close Socket,端口号;

Close Socket 即可用于关闭本地控制器作客户端时，本地的客户端端口号；也可关闭本地控制器作服务器时，外部客户端的端口号。

范围<1024~9999>，其中 3333 为系统占用，不可使用。

范例：

```
Close Socket,1025;
```

2.9.3 SetPortBuf

功能：设置发送缓冲区的值

格式：SetPortBuf(字符串变量);

说明：将字符串变量设到发送缓冲区

范例：SetPortBuf(str1);

2.9.4 GetPortbuf

功能：获取接收缓冲区的字符串

格式：字符串变量 = GetPortbuf(起始位 , 字符个数);

参数	意义
起始位	可为数字或 B/LB/R/LR 变量
字符个数	可为数字或 B/LB 变量
字符串变量	返回值，存放接收到的字符串

说明：在接收缓冲区中从某一位（<起始位>）开始取连续 N 个（<字符个数>）字符，并传给指定的字符串变量。失败时传 0。

范例：

```
Str1 = GetPortbuf(2,3); ##从第 2 位开始，取 3 个字符
```

2.9.5 Send Port

功能：将发送缓冲区的数据发送到远端端口

格式：Send Port[***];

参数：***:客户端端口号，范围 1024-9999。其中 3333 为系统占用，不可使用。

注意事项：

即可用于本地控制器作客户端，也可用于本地控制器作服务器的情况。这里的端口号恒为客户端端口号。

本地控制器作服务器，外部作唯一客户端时，当不知道外部客户端的端口号，可以使用端口号 4444。

范例：Send Port[1025];

2.9.6 Get Port

功能：接收缓冲区接收远端端口数据

格式：Get Port[***],T[***],Goto L[***];

说明：

在 T[***]时间内，接收端口数据。若接收成功则将数据存入接收缓冲区，并继续执行下一行指令（不执该句的 Goto L[***]）；若未接收到数据，则执行该句的 Goto L[***]，即跳转至 L[***]处。

参数	意义
Port[***]	客户端端口,***为端口号，范围 1024-9999。其中 3333 不可使用
T[***]	接收视觉数据的时间。***为数值，范围<0-100>，单位：秒
Goto L[***]	***标签号，指令时间内未接收成功时跳转的标签号

注意事项：

即可用于本地控制器作客户端，也可用于本地控制器作服务器的情况。这里的端口号恒为客户端端口号。

本地控制器作服务器，外部作唯一客户端时，当不知道外部客户端的端口号，可以使用端口号 4444。

范例：

```
Get Port[1025],T[1],Goto L[2];
```

2.9.7 CnvVision

功能：打开/关闭传送带的视觉端口

格式：CnvVision(Conveyor[***],ON/OFF,客户端端口号);

参数：Conveyor[***]传送带编号，***范围 0-3

说明：该语句在获取视觉数据语句 GetCnvObject 前后用到。指明使用哪条传送带以及客户端端口号。

范例：

```
CnvVision (Conveyor[0],ON,1024);
```

```
GetCnvObject(0,0), Goto L[0];
```

.....

```
CnvVision (Conveyor[0],OFF,1024);
```

2.9.8 GetCnvObject

功能：接收传送带上物体的视觉数据

格式：GetCnvObject(CnvID, ObjID),Goto L[***];

参数	意义
CnvID	传送带编号。范围 0-3
ObjID	物体类型编号，0-15
L[***]	***标签号，指令时间内未接收成功跳转的标签号，接收成功，跳转至下一行

说明：

GetCnvObject 与 CnvVision 开关视觉一起使用。打开视觉端口后，接收的视觉数据直接传出于 DSP 处理，直至关闭视觉端口。

若接收到视觉数据，则并继续执行下一行指令（不执该句的 Goto L[***]）；若未接收到数据，则执行该句的 Goto L[***]，即跳转至 L[***]处。

在循环中调用 GetCnvObject，能不断获取传送带上物体的视觉数据。指令每次使用

GetCnvObject, 会取用一条视觉数据, 获得传送带上符合这一类型的物体的位置; 下次会自动取用下一条视觉数据, 获得传送带上符合这个类型的下一个物体的位置。

范例:

```
START;
```

```
L[0]:
```

```
##打开端口。外部设备作服务器地址 10.44.53.13, 端口号 1025;
```

```
##本地控制器作客户端, 端口号 1026。
```

```
Open Socket("10.44.53.13",1025,1026,B0);
```

```
If B0 == 0
```

```
Goto L[0];
```

```
CnvVision (Conveyor[1],ON,1026);
```

```
P[30]=(0,0,10,0,0,0),(0,0,0,0),(7,0,0); ##定义 P[30]为在 1 号传送带上物体的正上方 10mm 处
```

```
L[1]:
```

```
Movj P[0],V[30],Z[0];
```

```
GetCnvObject(1,0, Goto L[1]; ##接收 1 号传送带, 0 号类型的物体的数据
```

```
RefSys Conveyor(1,Tool[2]); ##使用工具 2 末端完成与传送带 1 的速度同步运动
```

```
Movl P[30],V[100],Z[1],Tool[2]; ##P[30]是在传送带 1 坐标系下的点
```

```
Set Out[1],ON; ##打开开关, 吸附物体
```

```
Delay T[1];
```

```
RefSys Base; ##切换到机器人坐标系
```

```
Jump P[1],V[100],Z[0],LH[10],MH[-750],RH[10]; ##将物体移动至 P[1]处
```

```
Set Out[1],OFF,T[0]; ##放置物体
```

```
Delay T[1];
```

```
Goto L[1];
```

```
CnvVision (Conveyor[1],OFF,1026);
```

```
Close Socket,111;
```

```
END;
```

注意: 相机获取到传送带上物体的点, 会全部保存在一片区域中, GetCnvObject 指令只是去取用这些数据。因此当前一次运动操作未到位时, 仍能取用下一条视觉数据, 不会因为运动不及时而丢失下一个物体。除非传送带上的物体位置出界。

2.9.9 CopyCnvObject

功能: 复制传送带上物体的视觉数据

格式: CopyCnvObject(CnvID, ObjID),Goto L[***];

参数:

CnvID: 传送带编号。范围 0-3

ObjID: 物体类型编号, 0-15

L[***]: ***标签号, 指令时间内未接收成功跳转的标签号, 接收成功, 跳转至下一行。

说明: 从传送带数据队列中取一个对象, 对象在队列中仍保留。

范例: CopyCnvObject(1, 0),Goto L[1];

2.10 其它指令

2.10.1 USING MAIN

功能：声明调用主程序的位置变量

格式：USING MAIN,P[***],变量个数；

参数：

P[***]：主程序位置变量

变量个数：从 P[***]开始连续变量的个数，范围 1-9999

说明：在子程序中声明主程序中的 P[***]及其后 N 个位置变量，以便调用。

注意：

1. 子程序中被替代的点不能为空，事先要正常取点；
2. USING MAIN 指令在同一个子程序中可以被使用多次；

范例：

主程序 Main.pro

```
START;  
L[1]:  
Movj P[0],V[30],Z[0];  
Movj P[1],V[30],Z[0];  
Movj P[2],V[30],Z[0];  
Movj P[3],V[30],Z[0];  
Call"SubMain.pro";  
Movj P[4],V[30],Z[0];  
Goto L[1];  
END;
```

子程序 SubMain.pro

```
START;  
USING MAIN, P[2],2;  
L[1]:  
Movj P[0],V[30],Z[0];  
Movj P[1],V[30],Z[0];  
Movj P[2],V[30],Z[0];  
Movj P[3],V[30],Z[0];  
Delay T[2];  
Movj P[4],V[30],Z[0];  
Set Out[0],ON,T[1];  
Delay T[2];  
Set Out[0],OFF,T[1];  
Goto L[1];  
END;
```



2.10.2 LoadScrewParm

功能：加载螺丝工艺参数至伺服

格式：LoadScrewParm(工艺文件名, ***);

参数：

工艺文件名：锁付某个产品时所使用的螺丝工艺文件；

B/LB***：指令返回值，0 代表加载成功，非 0 表示加载失败

说明：

每次点击该指令后都会从相应的文件夹中遍历出所有的螺丝工艺文件，点击相应的文件名后，该指令运行后则会将工艺参数加载至伺服控制器中。因此，该指令在一个程序中只需要在程序初始化时加载一次即可。

2.10.3 LockScrew

功能：螺丝锁付

格式：LockScrew(***, P[***], V[***]);

参数描述：

参数 1：锁螺丝工艺号，即锁螺丝所使用的工艺参数号，范围 0-15；

参数 2：P[***]即锁付位置变量，***可以为数字或 R/LR 变量；

参数 3：V[***]即滑台下降的速度，用户可根据节拍自设。

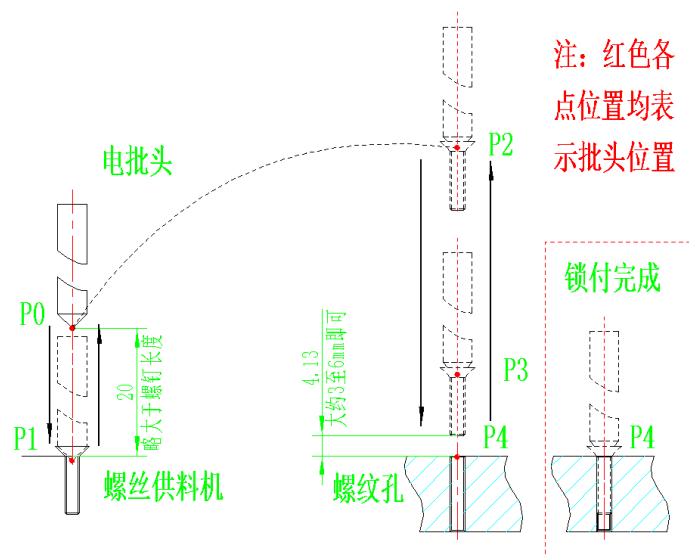
说明：

- 1、锁付过程的工艺参数均能通过工艺号选取；
- 2、LockScrew 将从搜索起始点到锁付起始点的滑台运动和电批锁付运动相结合。

范例：一个简单锁螺丝示例流程如下：

- 1) 从 P[0]处开始，运动到供料机螺丝吸附预备位 P[1]，然后打开气阀，吸附螺丝上升至 P[0]，并快速移动到准备位 P[2]，直线运动到锁付位 P[3]，执行螺丝锁紧指令 LockScrew。
- 2) LockScrew 是融合了滑台上下运动和电批旋转运动的综合指令，因此在 P[3]之后就开始执行，从 P[3]运动到 P[4]同时就同步开始进入锁螺丝的搜索阶段。CheckLock 会等待锁付完成。
- 3) 完成后松开吸附，直线回到 P[2]，快速回到初始位置 P[0]。

注意：切记示教好再再现，防止 P[2]高度过低导致碰到障碍物。



范例：

```
START;  
LoadScrewParm("aa.stp",B1);  
Movj P[0],V[30],Z[0];  
Movl P[1],V[30],Z[0];
```

```

WaitInPos;
Set Out[7],ON;
Movl P[0],V[30],Z[0];
Movj P[2],V[30],Z[0];
Movl P[3],V[30],Z[0];
LockScrew (0,P[4],V[30]);
CheckLock B0;
If B0==1
    Print "OK";
Else
    Print "NG";
EndIf;
Set Out[7],OFF;
Movl P[3],V[30],Z[0];
Movl P[2],V[30],Z[0];
END;

```

2.10.4 CheckLock

功能：螺丝拧紧结果检测

格式：CheckLock(B/LB***);

参数描述：

参数 1: B/LB***即指令返回值，1: 锁付成功 2: 滑牙 3: 浮锁

说明：

该指令为阻塞函数，一般使用 LockScrew 指令后在执行 CheckLock

2.10.5 UnLockScrew

功能：松螺丝

格式：UnLockScrew(***, P[***], V[***], ***, ***);

参数描述：

参数 1: 拆螺丝工艺号，即拆螺丝所使用的工艺参数号，范围 0-15;

参数 2: P[***]即拆螺丝起始位置变量，***可以为数字或 R/LR 变量;

参数 3: V[***]即滑台下降的速度，用户可根据节拍自设。

参数 4: 松螺丝时滑台的回退距离，单位为 mm

参数 5: 松螺丝速度，单位为 mm/s

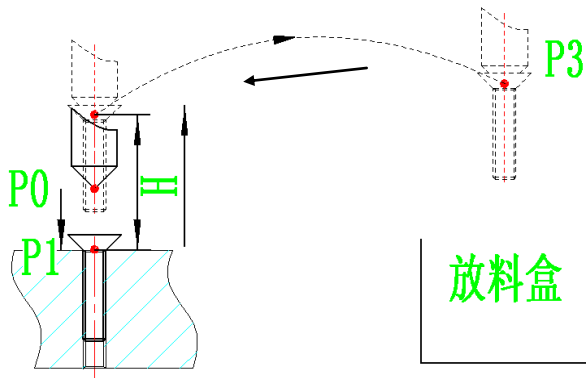
说明：

- 1、拆卸过程的工艺参数均能通过工艺号选取;
- 2、UnLockScrew 是等待点到位后开始编译，因此其前可不用加 WaitInPos。一般地，在拆卸起始位置前就开始执行这个指令，让搜索阶段提前开始。
- 3、相比锁指令，拆指令多了松螺丝时回退距离参数。

案例：一个简单拆螺丝示例流程如下：

- 1) 初始位置是 P[3]，从 P[3]处开始，运动到拆锁预备位 P[0]，吸气吸附螺丝，然后执行指令 UnLockScrew 开始进行拆的过程。

- 2) `UnLockScrew` 是等待点到位后开始编译，因此在 `P[0]`到位之后就执行。这样，从 `P[0]`运动到`P[1]`同时就同步开始进入拆螺丝的搜索阶段，待到达`P[1]`点后就边拆边回退，`CheckUnLock` 会等待拆锁完成。
- 3) 拆完后回准备位 `P[3]`，也即螺丝放置点，松气放置。



注：红色各点位置均表示批头位置

```

START;
LoadScrewPam("aa.stp",B0);
Movj P[0],V[30],Z[0];
WaitInPos;
Set Out[7],ON;
UnLockScrew(0,P[1],V[30],2,12);
CheckUnLock(B1);
If B1==1
    Print "OK";
Else
    Print "NG";
EndIf;
Movl P[3],V[30],Z[0];
WaitInPos;
Set Out[7],OFF;
Delay(0.1);
END;

```

2.10.6 CheckUnLock

功能：螺丝拧松结果检测

格式：`CheckUnLock(B/LB***)`;

参数描述：

参数 1: `B/LB***`即指令返回值，1: 拧松成功 2: 滑牙 3: 浮锁

说明：

该指令为阻塞函数，一般使用 `UnLockScrew` 指令后在执行 `CheckUnLock`。

2.10.7 ScrewStop

功能：锁螺丝或者松螺丝停止指令

格式：`ScrewStop`;

参数：无

说明：用户可根据相应的报警执行电批停止指令。

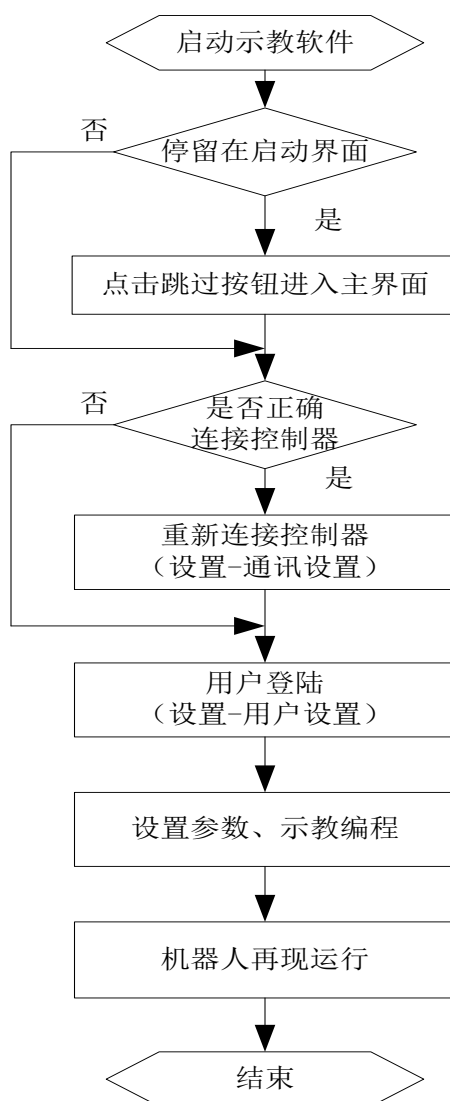
3 示教软件使用说明

3.1 软件使用入门

示教软件按功能可划分为三个模块，为编程/运行模块、监控模块和设置模块。编程模块是示教过程的主操作模块，主要用于程序的编辑等。监控模块用于观察程序参数、接口状态、日志等各项数据，此外也支持修改变量参数。设置模块用于进行机器人相关的设置，其功能十分广泛，包括系统相关设置、机器人结构/运动参数设置、扩展模块设置等等。

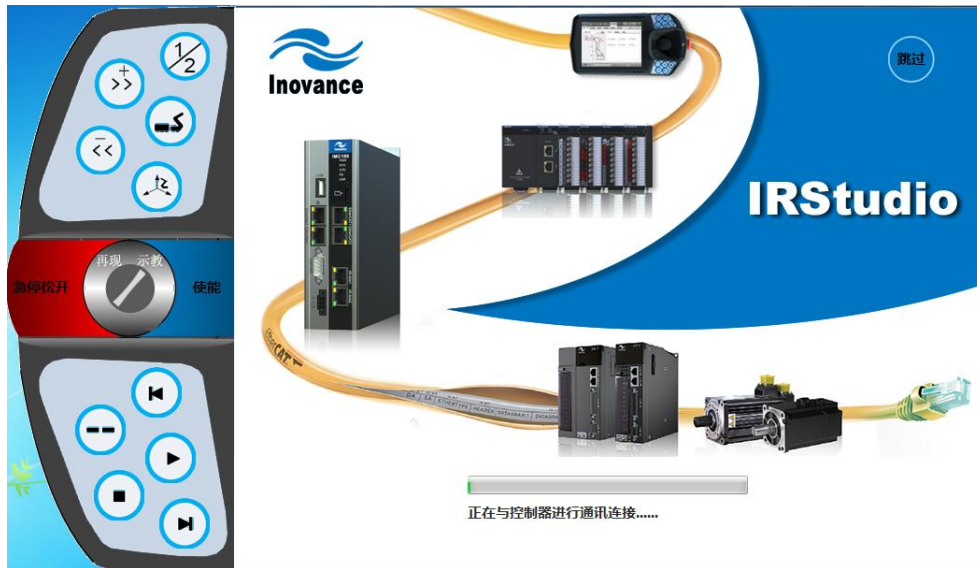
3.1.1 基本操作流程

示教编程的基本操作流程如下图所示：启动示教软件后，会根据上一次的连接地址自动连接；若连接成功则检查地址无误后进入“设置—用户设置”界面登陆；登陆后对相关参数（基本上为除系统设置外的其它所有设置）进行设置并进行示教，示教完成后运行。



3.1.2 软件开启-连接-登陆

软件开启后连接界面如图。



启动界面进度条下方显示当前状态，启动过程可分为 9 个状态：系统开始启动、正在初始化系统参数、示教盒通信连接成功、正在初始化 FPGA 参数、正在配置 DSP 参数、正在初始化 Ethcat 参数、正在初始化 IRLink、正在启动视觉线程、正在启动 DSP 通信线程、正在初始化 DSP 数据。在正常情况下，会迅速运行完上述 9 个状态变换后，并最终显示“控制器启动成功”，进入主界面；若未能成功启动，可根据停留的状态分析故障原因，详见下表。

停留状态	问题描述	处理办法
正在初始化系统参数	读取配置文件失败	1.点击跳过,重新设置系统参数; 2.恢复出厂默认值; 3.从 USB 接口重新导入系统文件
正在进行通信连接	IP 地址错误	点击跳过,在通讯设置下修改 IP 地址。
正在初始化 FPGA 参数	FPGA 启动失败	检查硬件
正在配置 DSP 参数	DSP 启动失败	检查硬件
正在初始化 Ethcat 参数	1.Ethcat 参数配置错误; 2.Ethcat 网络从站数不匹配	1.检查 Ethcat 网络连接; 2.检查系统参数配置
正在初始化 IRLink	1. IRLink 参数配置错误; 2. IRLink 网络从站数不匹配	1.检查 IRLink 网络连接; 2.检查系统参数配置
正在启动视觉线程	视觉线程启动失败	检查硬件
正在启动 DSP 通信线程	FPGA 通讯失败	检查硬件
正在初始化 DSP 数据	DSP 通讯失败	检查硬件

按之前操作流程图所述，需要检查是否正确连接控制器，这时可以点击“跳过”按钮，转到“设置-通讯设置”页面调整。

示教盒通讯

连接状态：已连接

端口：

断开

IP地址： · · ·

连接

检查连接地址

控制器Eth1设置

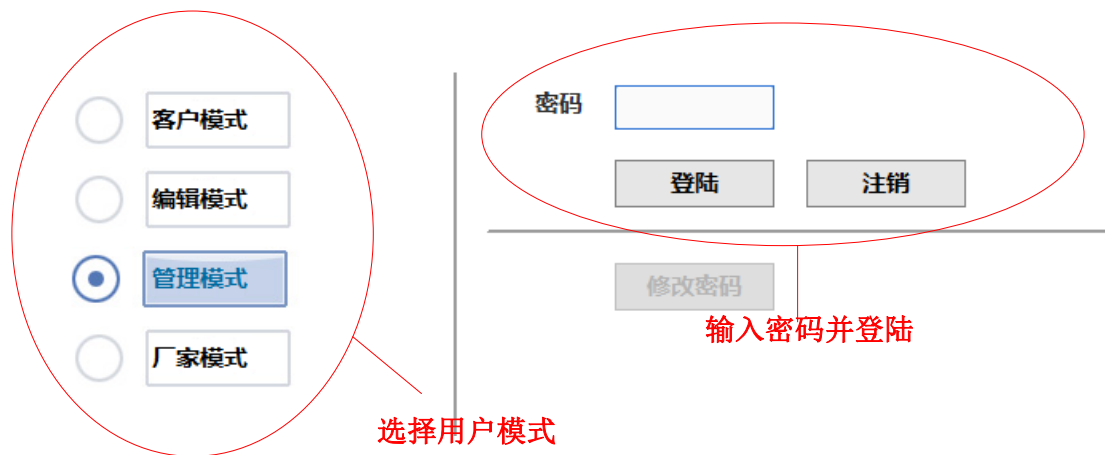
动态IP开关：

客户端

服务器

端口号

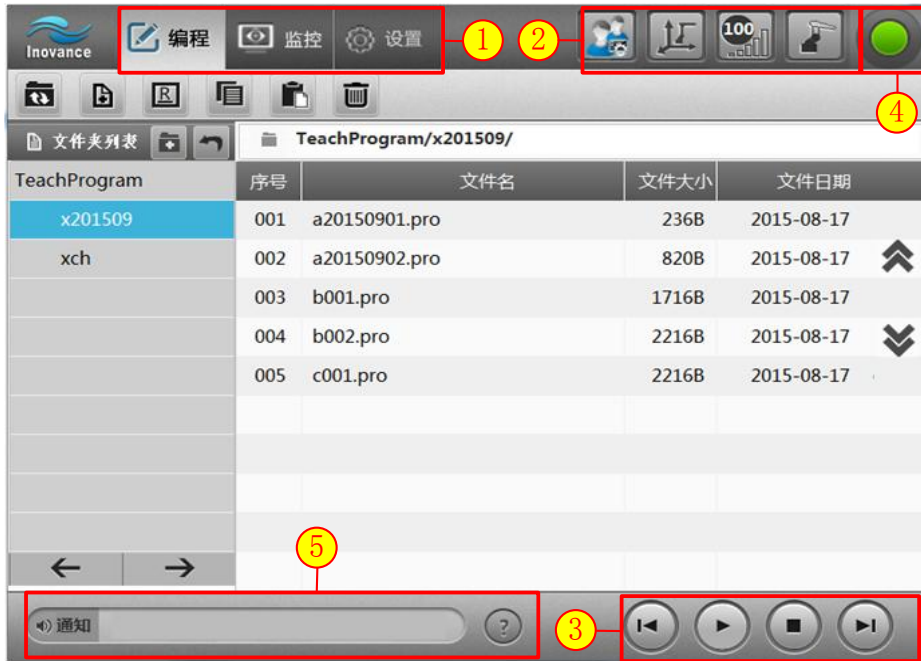
核对 IP 地址并连接，详细的方法见 3.2.6 (a) 节通讯设置。用户登陆位于“设置-用户设置”页面，选择模式并登陆，详细的用户模式说明见 3.2.6 (e) 节用户设置。



启动-连接-登陆后，方可进行正式的使用。

3.1.3 主界面功能介绍

示教软件启动并连接成功后，显示的主界面如下：



面板切换栏

通过面板切换栏显示不同的操作面板，包括编程/运行面板、监控面板和设置面板。

控制工具栏

控制工具栏有 4 种按钮，分别为用户模式按钮、坐标系切换按钮、速度倍率/寸动选择按钮、轴组切换按钮。

用户模式显示		客户模式
		编辑模式
		管理模式
		厂家模式
坐标系切换按钮		关节坐标系
		基坐标系
		工具坐标系。 上面数字代表选用的工具号

		用户坐标系。 上面数字代表选用的用户号
速度倍率/ 寸动选择按钮		设定速度的 5%运行
		设定速度的 25%运行
		设定速度的 50%运行
		设定速度的 100%运行
		返回速度调节倍率
		切换到寸动设置按钮
		设置寸动值，旋转步长 0.2°， 平移步长 0.2 mm
		设置寸动值，旋转步长 1°， 平移步长 1 mm
		设置寸动值，旋转步长 5°， 平移步长 5 mm
		用户自定义寸动值 (取设置-运动设置-精度-寸动中的值)
	轴组切换按钮	
		外部摇杆控制的轴组 J4/J5/J6(A/B/C)

寸动模式下的移动以步长衡量，运动时的速度仍与速度倍率相关。

运动控制栏

运动控制栏用于控制程序的执行，包括连续运行/暂停、停止、单步前进、单步后退四个按钮。可用于示教编程模式和再现运行模式下使用。

选项按钮	选项功能	示教编程模式	再现运行模式
	单步后退*	点动形式。一直接住会运行该行指令，直至运行完成后，后退至上一行指令；中途松开则立即停止。	无效
	连续运行	点动形式。一直接住会一直运行，直至整个程序运行结束；中途松开则立即停止。	非点动形式，点击一下，程序会一直运行。再次点击会切换暂停/运行。
	停止	由于示教编程模式下运动都是点动模式，会在运动键松开时自动停止，因此这里停止键无效。	一经按下立即终止运行，并重置程序光标行至开头。
	单步前进	点动形式。一直接住会运行该行指令，直至运行完成后，前进至下一行指令；中途松开则立即停止。	无效

***单步后退特性:**




- 单步后退是单步前进的逆向运动，只有先进行单步前进才能单步后退。若单步前进后出现其它操作（如选择其它程序行），则不能后退，提示“无后退数据”。
- 单步后退只针对运动指令，对于非运动指令不做处理。
- 后退最多支持 9 步。
- 后退的第一步总是会从当前位置走到最后一次单步前进的到位的位置，若从最后一次单步前进的到位的位置开始后退，则第一次后退将不作运动。

特别的，对于 movc 的后退：

如果后退的动作没有退到圆弧的起始点，此时再次进行同一圆弧的前进和后退操作，由于第一次操作没有记录起始点，所以第二次再进行后退操作时就会报圆弧轨迹不可控；本质原因是起始点没有记录。

状态指示灯

状态指示灯用于指示机器人当前所处的状态，包含伺服使能、待机、急停、报警和断线几种状态。注意：只有处于使能状态时机器人才能运动。

	伺服使能：此时急停被松开，伺服被使能，可进行示教和再现
	急停状态：急停按钮被按下，机器人不能运动。
	待机状态：急停按钮松开，伺服系统尚未使能。

沿工具坐标系和用户坐标系的调整模式与基坐标系类似，都是用 X、Y、Z、Rx、Ry、Rz 来表示。不同的是，工具坐标系移动是以沿当前工具坐标系移动；用户坐标系移动是工具末端沿某一用户坐标系移动。

工具附着在机器人本体末端的，机器人本体末端位姿发生变化，工具坐标系在空间中的绝对位置也发生变化。沿工具坐标系的移动是指沿着当前工具坐标系所指方向的移动。因此看到的坐标值与移动并非是统一的，如 X+移动时，坐标值可能 XY 都变化。

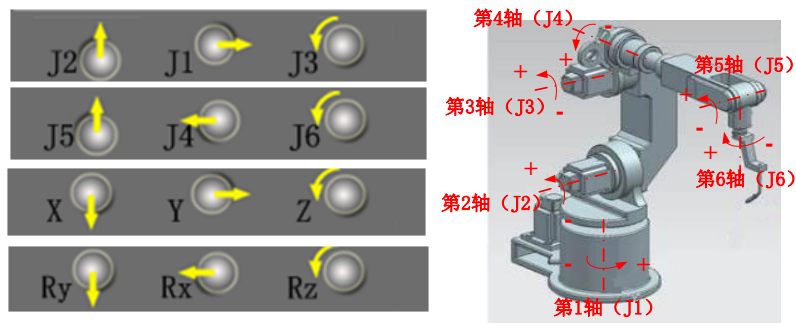
b) 摇杆操作

操作面板下方标有摇杆指示方向，表明了摇杆移动对应的机器人运动方向。

摇杆能有上下、左右和旋转三种方式的运动，对应机器人的三种运动。在关节坐标系下，摇杆能控制 1/2/3 轴或 4/5/6 轴，具体控制的轴组通过切换按钮选择。

在基坐标系下，摇杆能控制机器人末端位置 X/Y/Z 或姿态 Rx/Ry/Rz，工具坐标系移动是以沿当前工具坐标系移动；用户坐标系移动是工具末端沿某一用户坐标系移动。

他们的运动与摇杆的关系如图所示：

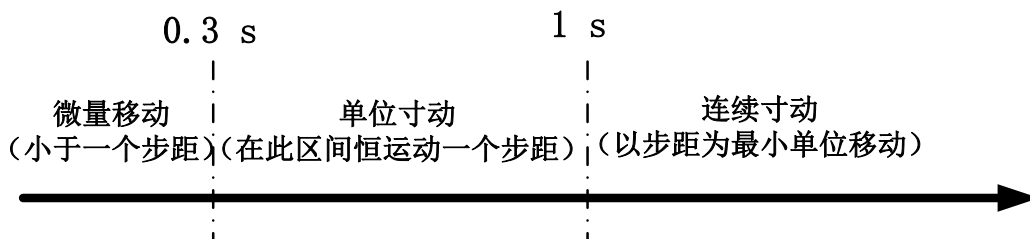


摇杆的复合运动：支持多个方向摇杆的同时操纵，如在基坐标系下同时摇动 X 和 Y，则机器人沿 X 和 Y 方向同时运动。注意：寸动模式下，禁止复合摇动摇杆。若多个方向摇动摇杆，也只在摇杆偏移最多的方向进行运动。

摇杆的力度控制：摇杆偏移的幅度越大，运动的速度越快。注意：在摇杆的幅度控制速度的同时，示教器工具栏右上角的速度调节按钮也会影响速度。此外，寸动模式下，摇杆的力度控制不生效，小幅度偏移摇杆与大幅度偏移摇杆，结果相同，寸动的速度不会受其影响。

寸动模式的摇杆控制：

在寸动模式下，摇杆控制的运动将会根据摇杆偏离中间位置的时间实行分段处理：



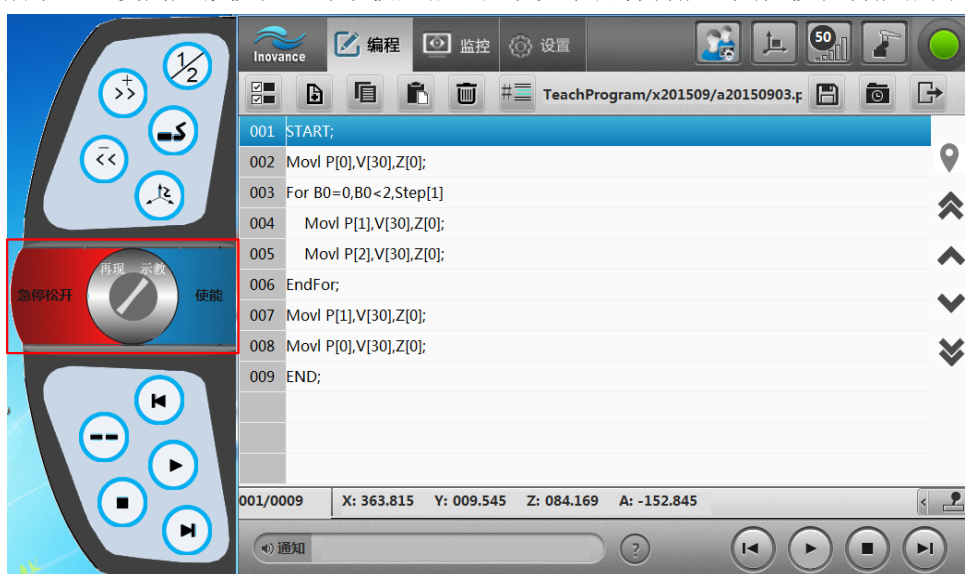
微量移动: 适用于微小距离移动（小于一个步距）场合。当摇杆的偏移时间小于 0.3 秒时，执行此种微量运动。摇杆在 0.3 秒之前一旦复位，立即停止运动。微量移动最多运行一个步距。发生此种特殊情况的举例：当步距特别小而加速度特别大时，不但在 0.3 秒内，而且是在摇杆复位前就完成了寸动，此时纵使摇杆未复位，也不进行更多运动。

单位寸动: 适用于只运动一个步距的场合。当摇杆复位的时刻处于 0.3~1 秒之间时，摇杆恒运动一个步距。

连续寸动: 适用于以步距为单位、连续多步运动的场合。当摇杆持续 1 秒未复位时，进入此阶段，以一个步距为最小单位运动。当完成一个步距时摇杆仍未复位，则开始下一个寸动；当完成一个步距时摇杆已复位，则不再运动。当摇杆复位的时刻，当前寸动未完成，则继续完成当前寸动再停止。

3.1.5PC 版本虚拟按键介绍

InoTeachPad 示教盒的物理按键见示教盒说明书，PC 版的示教软件中也有对应的按键功能。如图所示，左侧为虚拟按键，与示教盒相比只缺少了摇杆功能，其他按键功能相同。



按键	按键名称	按键功能
	急停按钮	用于控制机器人紧急停止。
	使能按钮	用于控制伺服电机的使能状态。
	示教/再现开关	用于切换机器人示教/再现模式。
	速度增	速度增加，按下按钮，速度值增 1，长时间按下按钮，速度持续上升。

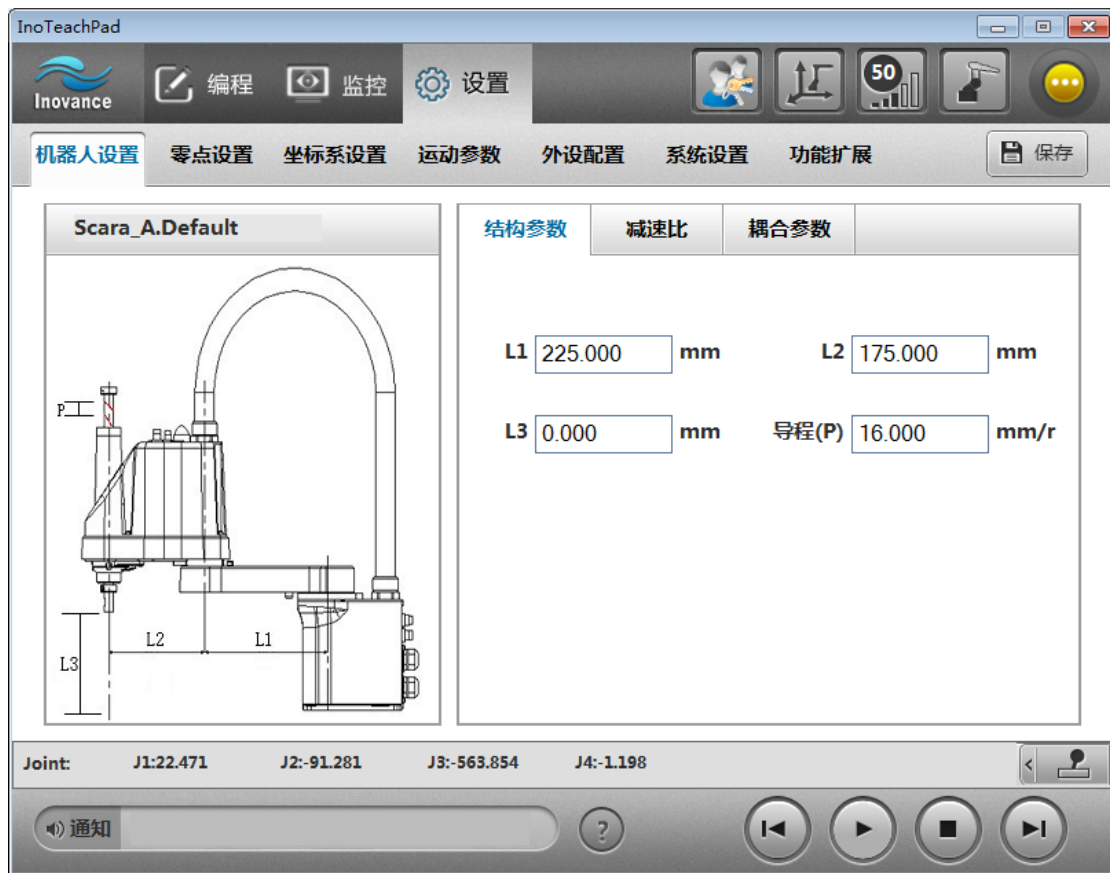
	速度减	速度减少，按下按钮，速度值减 1，长时间按下按钮，速度持续下降。
	轴切换	摇杆功能切换按钮。摇杆控制轴在 1/2/3 (X/Y/Z) 轴和 4/5/6(Rx/Ry/ Rz) 轴组之间切换。
	外部轴切换	当机器人有外部运动轴时，控制轴按钮切换至外部运动轴。
	坐标系选择	进行坐标系的切换：关节坐标系、基坐标系、工具坐标系、用户坐标系。
	示教盒版本： 示教/再现切 换 PC 版本：寸 动	示教盒版本：示教/再现切换按钮，示教模式用于取点编程及试运行；再现模式用于实际加工中执行程序。 PC 版本：寸动量调整
	运行	运行模式下，选择该按钮，机器人在再现运行所选程序； 示教模式下，按下该按钮，机器人连续运行，松开按钮，机器人暂停运行。
	停止	机器人运行时，选择该按钮，机器人停止运行。
	前进	示教模式下，程序运行所选择行，光标跳转至下一行。
	后退	示教模式下，程序运行所选择行，光标跳转至上一行。

3.2 设置

在示教前，需要进行一系列设置，做好准备工作。设置包括机器人设置、零点设置、坐标系设置、运动设置、工艺设置、系统设置六大项。在初次使用时，需要根据机器人的实际属性，进行机器人设置、零点设置、运动范围、运动特性设置。一般情况下，若不更换机器人匹配，则今后无需重设这些参数。工作原点、坐标系的设置则根据用户需求自由安排。设置完某项参数后，请注意使用右上角的保存按钮及时保存。

3.2.1 机器人设置

机器人设置包含结构参数、减速比、耦合参数，按照实际配置情况填写。



在厂家模式下，能编辑内部补偿参数。

3.2.2 零点设置

零点设置包括绝对零点、增量式回零两种方法，绝对零点适用于使用绝对编码器的机器人，增量式回零适用于使用增量式编码器的机器人。

c) 绝对零点

绝对零点设置界面如图所示，通过以下两步完成：

第一步：通过示教器调整机器人关节到零点位置，点击“当前值”按钮，编码器数值自动显示在面板的文本框中。

第二步：急停机器人，点击“保存”按钮，完成后重启控制器。



示教软件支持的 J1~J6 编码器数值范围为 $-2^{21} \sim 2^{21}$ ，如果在机器人零点时编码器读数过大，建议清除编码器圈数，重新按上述步骤操作。清除编码器圈数的操作查询相应的伺服、编码器手册。

d) 工作原点

工作原点与零点不同，是用户自定义的位置变量，可在程序中使用。系统可以设置 5 个工作原点，保存在变量 Home[0]-Home[4]中。工作原点坐标可手动输入，也可通过示教使机器人运动到原点位置，点击“当前值”自动获取。获取零点坐标后点击“保存”，原点被保存到下来，如图所示。



3.2.3 坐标系设置

在坐标系设置中可以定义工具坐标系和用户坐标系。

注意：区分编辑和使能。

编辑：指正在被编辑的坐标系，右侧会显示对应坐标系的编辑界面。

使能：正在被使用的坐标系。

工具号 0 为默认的工具坐标系，代表不带工具，是不可编辑的。用户号 0 为默认的用户坐标系，代表用户坐标系取基坐标系，也是不可编辑的。



a) 工具坐标系设置

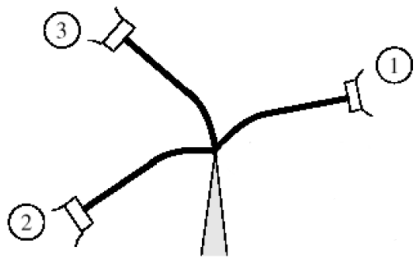
工具坐标系建立在工具上，其参数为工具坐标系在机器人本体末端坐标系下的位姿偏移。参数设置的方法如下。

方法	特点	适用场景
直接法	直接输入坐标系参数	已知坐标系参数
三点法 TCP	通过三点对位获取工具的位置	需要人工标定工具坐标系的位置值
五点法 TCP	通过五点对位获取工具的位置	需要人工标定工具坐标系的位置值。取点更多，比三点法 TCP 更加准确。
三点法 TCP+ZX	通过三点对位获取工具的位置，再通过额外三点获取工具的姿态	需要人工标定工具坐标系的位置和姿态值
五点法 TCP+ZX	通过五点对位获取工具的位置，再通过额外五点获取工具的姿态	需要人工标定工具坐标系的位置和姿态值。取点更多，比三点法 TCP+ZX 更加准确。
锁螺丝四点法	通过选取水平面上的四个点，依次操作机器人使电批末端对准	适用于锁螺丝机型的工具坐标系

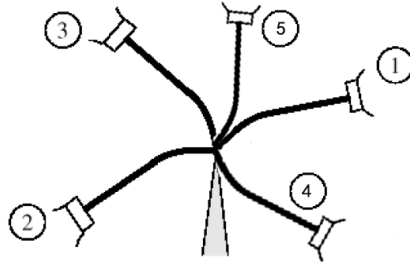
操作方法：

直接法：直接输入参数即可。

三点法 TCP：在机器人末端安装上工具后，使用示教功能调整工具的位姿，使工具末端以三种不同姿态对准空间中一点，通过示教软件的“取点”按钮记录，点击“生成”便可自动计算出工具坐标系的位置参数。



三点法



五点法示教工具坐标系

三点法 TCP 界面操作如下所示：

第一步：通过示教功能使工具到达上图中的 1 位置。

第二步：点击“取点 1”，其后会产生一个绿色的钩。

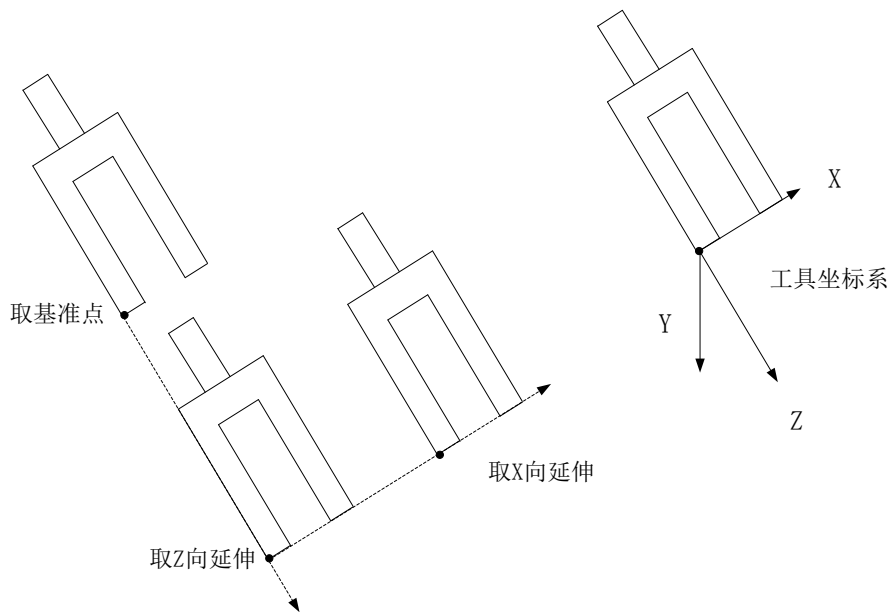
第三步：重复第一步、第二步，完成点 2、点 3 的定义。

第四步：点击“生成”，工具坐标系参数自动生成。若计算成功，会弹框显示本次标定的工具位置误差。坐标系结果显示于下面编辑框。

注意：三点法所选取的三种不同姿态应尽量间隔较大，一般的，推荐相隔 20° 以上。后面的五点法等同样需注意此项。



三点法 TCP+ZX: 在三点法 TCP 的基础上，额外取三个点，标定工具坐标系的姿态。Z 向延伸点与基准点构成工具坐标系的 Z 向，X 向延伸点与 Z 向延伸点构成工具坐标系的 X 向。Y 向由 Z 向与 X 向根据右手定则得到。如下图所示。



界面操作如下所示：

第一步：按“三点法 TCP”步骤取点 1、2、3。

第二步：依次示教“取基准点”、“取 Z 向延伸”、“取 X 向延伸”

第三步：点击“生成”，工具坐标系参数自动生成。结果显示于下面编辑框。



五点法TCP: 与三点法原理相同、方法类似，只不过取用五个不同姿态对同一点。

五点法TCP+ZX: 参照三点法TCP+ZX，只不过取前3个位置点变成了取5个点。

锁螺丝四点法：通过选取水平面上的四个点，依次操作机器人使电批末端对准，每对准一个点，点击界面对应的取点按钮。完成后点击【生成】。生成后，会出先现误差提示对话框；再出现“是否运用标定结果修正机器人结构参数”的提示。

注意事项：

- 1.只适用于锁螺丝机型。
- 2.四个点必须位于水平面上，相隔分开较好，推荐使用矩形的四个顶点。
- 3.这一标定过程不仅能得到工具坐标系数据，还可修正机器人结构参数，用户可选择是否修正。



注意：

Scara 和 Delta 机器人的工具与末端末轴同轴线（即只有 Z 向有值）时，无法通过三点法 TCP 或五点法 TCP 求得这个 Z 向位置参数。

Scara 和 Delta 机器人使用的工具的姿态只有 A 向有值，即工具相对于机器人末端坐标系只存在 Z 向旋转。

b) 用户坐标系设置

用户坐标系的设置参数是用户坐标系在基坐标系中的坐标，

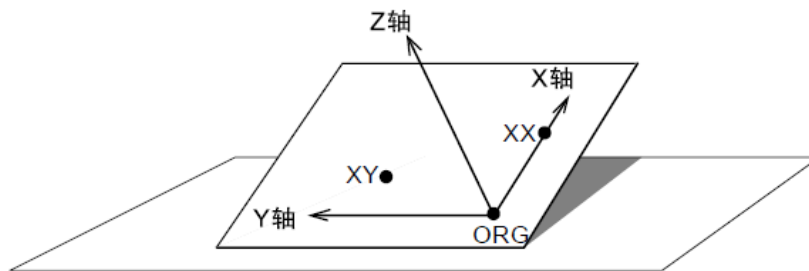
方法	特点	适用场景
直接法	直接输入坐标系参数	已知坐标系参数
三点法	通过三点标定用户坐标系	需要人工标定用户坐标系的值
旋转法	转盘上做标记点，旋转转盘，示教取点	用户坐标系位于外设转台中心（如使用旋转传送带）



操作方法:

直接法: 直接输入用户坐标系参数。

三点法: 第一点取用户坐标系的原点, 第二点取 X 轴正方向上一点, 第三点取 XY 平面上 Y 取正的一点, 其他操作与工具坐标系相同



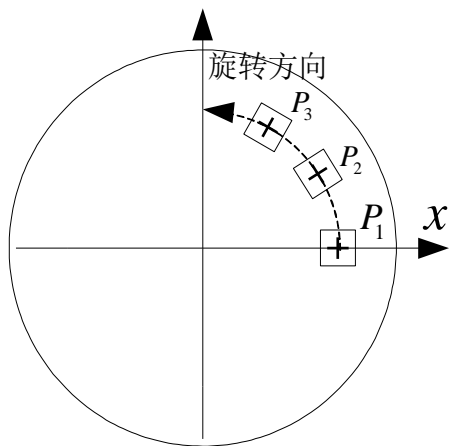
用户坐标定义点

ORG: 原点位置

XX: X 轴上的点

XY: XY 平面上的点

旋转法: 在转盘上标记一个固定点, 示教取点 1; 再分别经过旋转 2 次, 示教取点 2、3, 得到以旋转中心为原点, 以 P1 为 x 正方向的用户坐标系。



3.2.4 运动设置

a) 示教参数设置

寸动：指用户自定义的寸动步长，包含关节步长和线性步长。关节步长既指关节坐标系下每个关节的旋转寸动步长，也指基/工具/用户坐标系下的A/B/C方向的旋转寸动步长。线性步长指基/工具/用户坐标系下的X/Y/Z方向的平动寸动步长。

示教参数设置	运行参数设置	轴极限设置	干涉区设置																
寸动																			
<table border="0"> <tr> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 40%;"></td> </tr> <tr> <td style="background-color: #0056b3; color: white; text-align: center;">寸动</td> <td></td> <td></td> <td></td> </tr> <tr> <td style="background-color: #cccccc; text-align: center;">示教速度</td> <td></td> <td></td> <td></td> </tr> <tr> <td style="background-color: #cccccc; text-align: center;">示教加速度</td> <td></td> <td></td> <td></td> </tr> </table>								寸动				示教速度				示教加速度			
寸动																			
示教速度																			
示教加速度																			
<table border="0" style="width: 100%;"> <tr> <td colspan="4" style="text-align: center;">寸动</td> </tr> <tr> <td colspan="4" style="border-top: 1px solid black; border-bottom: 1px solid black;"></td> </tr> <tr> <td style="width: 30%;">关节步长</td> <td style="width: 15%; text-align: center;">0.100</td> <td style="width: 10%; text-align: center;">°</td> <td style="width: 45%;"></td> </tr> <tr> <td>线性步长</td> <td style="text-align: center;">0.100</td> <td style="text-align: center;">mm</td> <td></td> </tr> </table>				寸动								关节步长	0.100	°		线性步长	0.100	mm	
寸动																			
关节步长	0.100	°																	
线性步长	0.100	mm																	

示教速度：示教过程中的速度。包含最大允许位置速度、最大允许姿态速度、各关节最大允许速度。

全局最大速度百分比用于调整全局最大速度。它与运行参数设置中的全局最大速度百分比为同一参数，影响示教和再现速度。该值能与运动指令中的速度和界面右上角的速度倍率设置共同作用。

示教加速度：示教过程中的加速度。包含最大允许位置加速度、最大允许姿态加速度、各关节最大允许加速度。

全局最大加速度百分比用于调整全局最大加速度。它与运行参数设置中的全局最大加速度百分比为同一参数，影响示教和再现加速度。该值能与运动指令中的加速度共同作用。

示教参数设置	运行参数设置	轴极限设置	干涉区设置
寸动 示教速度 示教加速度	最大允许位置速度	<input type="text" value="30.000"/>	mm/s
	最大允许姿态速度	<input type="text" value="8.000"/>	°/s
	最大允许关节速度		
	J1	<input type="text" value="15.000"/>	°/s
	J2	<input type="text" value="15.000"/>	°/s
	J3	<input type="text" value="300.000"/>	°/s
	J4	<input type="text" value="15.000"/>	°/s
全局最大速度百分比 : 82% (30%-100%)		<input type="range" value="82"/>	

b) 运行参数设置

运行速度：包含最大允许位置速度、最大允许姿态速度、各关节最大允许速度。

全局最大速度百分比用于调整全局最大速度。它与示教参数设置中的全局最大速度百分比为同一参数，影响示教和再现速度。该值能与运动指令中的速度和界面右上角的速度倍率设置共同作用。

运行加速度：再现运行时的加速度。在再现运行中，指令中的加速度 Acc 将以此处加速度为基准。

全局最大加速度百分比用于调整全局最大加速度。它与示教参数设置中的全局最大加速度百分比为同一参数，影响示教和再现加速度。该值能与运动指令中的加速度共同作用。

停止减速度：停止减速度指操纵示教器暂停或停止时，运动停止的减速度。

注意：这些参数受实际使用的伺服系统性能的制约。

示教参数设置	运行参数设置	轴极限设置	干涉区设置
运行速度 运行加速度 停止减速度 过渡特性设置	最大允许位置速度	<input type="text" value="1000.000"/>	mm/s
	最大允许姿态速度	<input type="text" value="900.000"/>	°/s
	最大允许关节速度		
	J1	<input type="text" value="360.000"/>	°/s
	J2	<input type="text" value="360.000"/>	°/s
	J3	<input type="text" value="9000.000"/>	°/s
	J4	<input type="text" value="900.000"/>	°/s
全局最大速度百分比 : 82% (30%-100%)		<input type="range" value="82"/>	

过渡特性设置：包含过渡精度、圆弧姿态插补两项。

过渡精度：过渡精度单位代表运动指令中的最小过渡取值。关节过渡单位指关节插补中每个关节的过渡单位旋转量。线性过渡单位指XYZ方向合成的过渡单位值。其中，指令中设置为Z[0]时，表示无过渡，精确到位。Z[1]等级过渡长度等于此处设置的过渡单位长度，而Z[2] = Z[1]*2，Z[3]=Z[1]*3，依次类推。

圆弧插补：存在两种方式，姿态变化不与圆心角相关，姿态变化与圆心角相关。一般的，采用默认的第一种方式。注：该模式只针对6自由度串联机器人选择有效。

示教参数设置	运行参数设置	轴极限设置	干涉区设置
	<div style="border: 1px solid black; padding: 5px;"> <p>运行速度</p> <p>运行加速度</p> <p>停止减速度</p> <p style="background-color: #0070C0; color: white; padding: 2px;">过渡特性设置</p> </div>	<p>过渡精度</p> <hr/> <p>关节过渡单位 <input type="text" value="3.000"/> °</p> <p>线性过渡单位 <input type="text" value="10.000"/> mm</p> <p>圆弧姿态插补类型</p> <hr/> <p> <input checked="" type="radio"/> 关节插补 <input type="radio"/> 姿态插补 </p>	

c) 轴参数设置

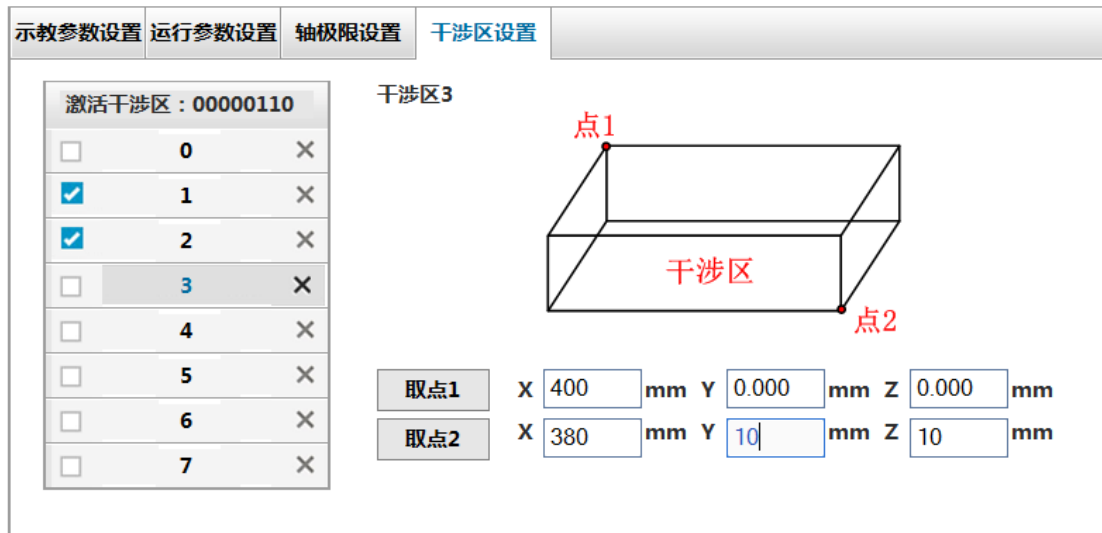
包含机器人轴极限、跟随误差、到位误差三项。

注意：出于安全考虑，请务必将轴极限设置在机械挡块范围以内。

示教参数设置	运行参数设置	轴极限设置	干涉区设置															
		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>序号</th> <th>负向</th> <th>正向</th> </tr> </thead> <tbody> <tr> <td>J1</td> <td><input type="text" value="-130.000"/> °</td> <td><input type="text" value="130.000"/> °</td> </tr> <tr> <td>J2</td> <td><input type="text" value="-148.000"/> °</td> <td><input type="text" value="148.000"/> °</td> </tr> <tr> <td>J3</td> <td><input type="text" value="-3700.000"/> °</td> <td><input type="text" value="10.000"/> °</td> </tr> <tr> <td>J4</td> <td><input type="text" value="-360.000"/> °</td> <td><input type="text" value="360.000"/> °</td> </tr> </tbody> </table>	序号	负向	正向	J1	<input type="text" value="-130.000"/> °	<input type="text" value="130.000"/> °	J2	<input type="text" value="-148.000"/> °	<input type="text" value="148.000"/> °	J3	<input type="text" value="-3700.000"/> °	<input type="text" value="10.000"/> °	J4	<input type="text" value="-360.000"/> °	<input type="text" value="360.000"/> °	
序号	负向	正向																
J1	<input type="text" value="-130.000"/> °	<input type="text" value="130.000"/> °																
J2	<input type="text" value="-148.000"/> °	<input type="text" value="148.000"/> °																
J3	<input type="text" value="-3700.000"/> °	<input type="text" value="10.000"/> °																
J4	<input type="text" value="-360.000"/> °	<input type="text" value="360.000"/> °																

d) 干涉区设置

干涉区是机器人工具末端禁止到达的区域。可设置 8 组干涉区，每组干涉区是由对角线两点 XYZ 确定的一个长方体区域。干涉区只考虑位置，不考虑姿态。在干涉区激活时，进入干涉区会产生报警。可同时激活多个干涉区。



3.2.5 外设配置

a) IRLink 设置

IRLink 是为 IMC100 系列的扩展产品,用于控制管理 IO,可在 IRLinkRTU 页面设置 IRLink 模块相关配置。IRLink 配置需要示教器具有 IRLink 配置权。

*IRLink 配置权: 默认 IRLink 配置权在示教器上,一旦使用过 InoRobShop 配置过 IRLink, 则 IRLink 配置权转移到 InoRobShop 上。此时示教器变更 IRLink 配置将出现报警,除非再通过 InoRobShop 下载一个空的 IRLink 配置到控制器中。详见 5.3.2 IRLink 组态配置权

IRLink 配置规格:

(1) 功耗规格

每个 RTU 后可提供的功率为 15W, 各模块功耗如下图:

类型	功耗
IMC100-0808-ETND	1.44W
IMC100-1600-END	1.25W
IMC100-0016-ETPD	1.25W
IMC100-0016-ETND	1.25w
IMC100-4DA	1.44W
IMC100-8AD	2.88W
IMC100-2ENID	2.88W

在配置时需要保证每个 RTU 后的模块消耗功率之和不超过 RTU 提供的功率。若需要更多模块,应增加 RTU 模块,再在其后串联。

(2) 资源规格

每个 IMC100 最多支持 2 个 IMC100-8AD 模块, 4 个 IMC100-4DA 模块, 8 个 IMC100-2ENID 模块, 16 个 IMC100-0808-ETND 模块。1 个 IMC100-1600-END 占用 2 个 IMC100-0808-ETND 模块的输入, 1 个 IMC100-0016-ETPD (IMC100-0016-ETND) 占用 2 个 IMC100-0808-ETND 模块的输出。

关于 IRLink 及扩展模块详细使用说明见《IMC100R 系列扩展模块用户手册》扩展模块手册。

配置 IRLink:

在连接好硬件设备后，需要在软件中设置 IRLink 配置。

点击页面左侧的“添加”按钮，会自动产生一个 RTU，该 RTU 的详细信息会在右侧显示。最多添加五个扩展模块 RTU。

点击右侧的“添加”按钮，会弹出选项框，有 0808、0016、1600、4DA、8AD、2ENC 六种选择。

选项	含义（模块类型）
0808	IMC100-0808-ETND
1600	IMC100-1600-END
0016	IMC100-0016-ETPD 或 IMC100-0016-ETND
4DA	IMC100-4DA
8AD	IMC100-8AD
2ENC	IMC100-2ENID

根据实际连接，在此处添加扩展模块。

The screenshot shows the 'I/O配置' (I/O Configuration) tab. On the left, 'RTU数: 1' (Number of RTUs: 1) is displayed with '添加' (Add) and '删除' (Delete) buttons. Below it, 'RTU_1' is listed with a '详细信息' (Details) button. On the right, 'IO设置_RTU1' (IO Settings_RTU1) is shown with '从站数: 0' (Slave Count: 0). A '添加' (Add) button is highlighted, and a dropdown menu is open, listing the module options: 0808, 0016, 1600, 4DA, 8AD, and 2ENC.

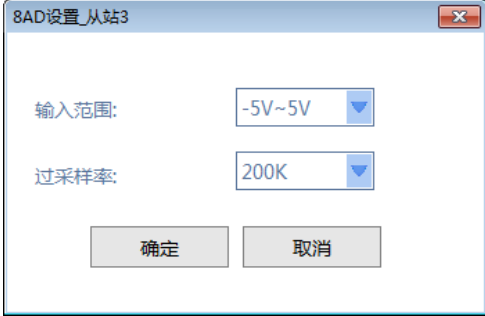

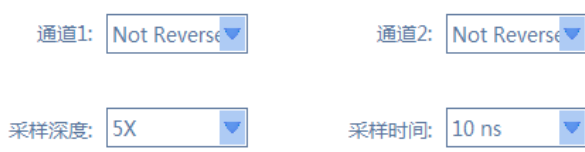
例如，次添加 4 个 0808，1 个 4DA，1 个 8AD，结果如下所示。

The screenshot shows the 'I/O配置' (I/O Configuration) tab after adding modules. 'RTU数: 1' remains. On the right, '从站数: 6' (Slave Count: 6) is displayed. Below this, there are several control buttons for each module:

- Four buttons for 0808 modules: '添加 从站0:0808 删除', '添加 从站1:0808 删除', '添加 从站2:0808 删除', and '添加 从站3:0808 删除'.
- One button for 4DA: '添加 从站4:4DA 删除'.
- One button for 8AD: '添加 从站5:8AD 删除'.
- One '添加' (Add) button for the remaining slot.

对于 8AD、4DA、2ENC 三种模块，还可点击该项，进入配置页面。

模块	配置页面	配置参数
----	------	------

<p>8AD (4 通道电压和 4 通道电流模拟量转换输入采集扩展模块)</p>		<p>可配置输入模拟量的范围和采样率。(所有通道的配置相同)其中,输入范围有两档;输入电压 -5V~5V 时默认输入电流范围 0~20mA;输入电压 -10V~10V 时默认输入电流范围 0~40mA。过采样率影响输入值采样精度,过采样率设置越小值越精确。</p>
<p>4DA (4 通道电压或电流模拟量转换输出扩展模块)</p>		<p>设置四个通道输出模拟量的范围</p>
<p>2ENC (2 通道差分输入增量编码器采集扩展模块)</p>		<p>可设置通道是否翻转、信号滤波时间。滤波时间为采样深度与采样时间的乘积。滤波时间越长,额定信号输入频率降低。</p>

b) I/O 设置

I/O 设置是与工位预约 (详见 4.1 工位预约) 配合使用的, 设置的内容仅在工位预约模式下有效。I/O 设置是将外部控制信号与机器人的某些功能关联起来, 以达到工位预约过程中控制机器人的目的。每个功能的 I/O 通过点击右侧的下拉框选择; 对于 I/O 子程序及中断子程序, 通过左键点击其按钮可设置要关联的子程序。

可配置 I/O 功能的介绍如下:

IO 功能		说明
输入 (用于外部控制)	启动	外部信号控制启动
	停止	外部信号控制停止
	暂停	外部信号控制暂停

	急停	外部信号控制急停
	清除报警	外部信号控制发出清除报警命令
	I/O 子程序	可设置三个有优先级顺序（顺序如下图所示）的 I/O 加工子程序。同一个程序只能预约一次，且当前程序正在运行过程中，再次预约无效。
	中断子程序	启动特定的中断子程序，子程序完成后再回来执行原程序。可设置两个中断子程序，不能嵌套使用。
	速度加	增加一个等级的速度倍率。每 5% 为一个等级，依次为 1%，5%，10%，15%……100%
	速度减	减少一个等级的速度倍率。每 5% 为一个等级，依次为 1%，5%，10%，15%……100%
输出 (用于显示状态)	报警	输出程序是否报警
	运行	输出程序是否运行
	停止	输出程序是否停止
	启动完成	输出控制器是否启动完成
	使能	输出当前是否使能



注意事项:

IO 设置的输入控制需要机器人处于工位预约模式下，非工位预约下无效。

关联了 IO 功能的输出端口将被占用，无法通过 IO 监控界面调整。

输出的运行和停止不是指机器人的运动或停止，而是程序的运行与停止。当程序运行结束后，不点击界面的停止按钮，仍将算作是程序运行状态。

要执行工位预约功能，需要机器人处于工位预约模式下，可通过在【设置】-【其他设置】-【控制设备】中设定为“远程 IO 单元”或“远程 Modbus 单元”。

3.2.6 系统设置

系统设置包含通讯设置、ECAT 扩展、IRLinkRTU、时间日期、用户设置、I/O 设置、其它设置。

a) 通讯设置

示教器有两种通讯方式，对应的控制器上有两个网口 EtherNet1 与 EtherNet2。EtherNet1 为默认动态 IP 网口，对应远程连接方式；EtherNet2 为静态 IP 网口，对应直接连接方式。

远程连接：机器人控制器通过 EtherNet1 接入互联网，示教软件只需在 IP 地址栏填入动态 IP 即可通过互联网访问控制器。

直接连接：示教器通过网线直接连接到控制器的 EtherNet2，控制器 IP 地址固定为 192.168.23.25，将此数据填入“IP 地址”栏。

IP 地址栏下方会显示当前链接控制所使用的端口。

示教盒通讯

连接状态：已连接

端口：

3333

断开

IP地址：

10

· 44

· 52

· 38

连接

控制器Eth1设置

动态IP开关：



客户端

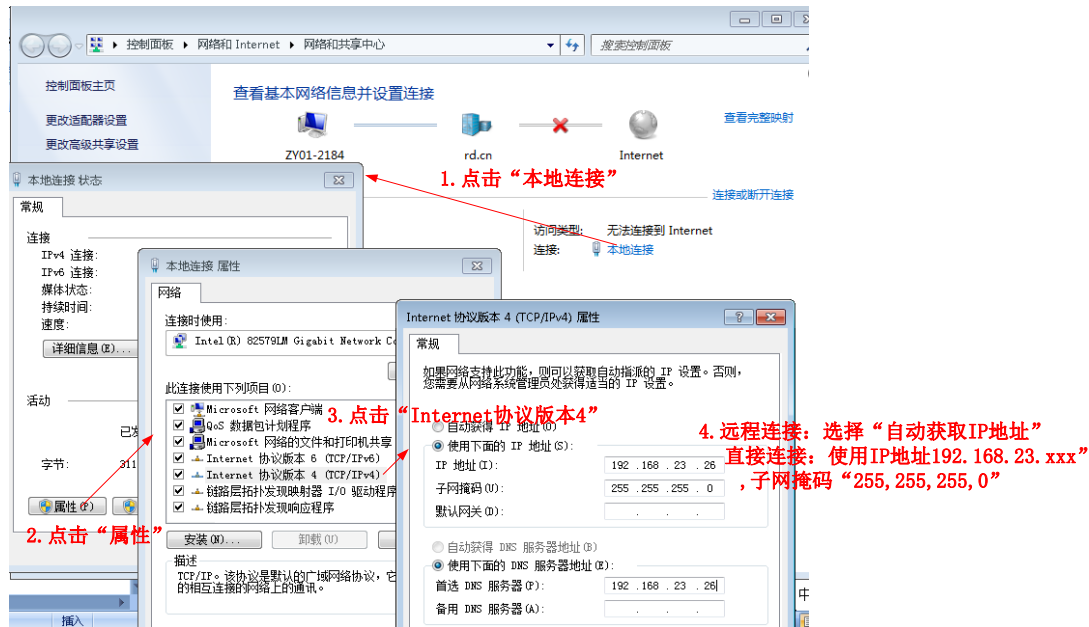


服务器

端口号

111

对于示教盒版，一般采用直接连接的方式，可直接使用，但对于 PC 版的示教软件，一般使用远程连接的方式。需在 PC 上设置 IP 地址，使之与控制器处于同一子网中，修改步骤如下图所示。



动态 IP 开关和客户端/服务器操作需要重启生效，当发生相关操作时，会提示重启。

动态 IP 开关:

在默认情况下，动态 IP 开关是开启的，表示 EtherNet1 是动态端口。但可以改变这一端口，将其设为静态端口：

先通过 EtherNet2 静态口连接，取消 EtherNet1 动态 IP 开关选项并定义 IP 地址，再断开即可。应用：当有使用示教器的同时，需要使用外部设备连接控制器时，常利用示教器连接控制器的 EtherNet2，外部设备连接控制器的 EtherNet1。此时可关闭“动态 IP 开关”，设置成静态 IP。这样外部设备就能与控制器直接连接。

注意：动态 IP 改成静态 IP 时，静态 IP 不能和默认的静态 IP 处于同一子网中，即不能设为（192.168.23.xxx）。

动态IP开关： 10 . 44 . 52 . 38

客户端与服务器:

这一功能用于视觉，指定视觉功能中机器人的角色，作为客户端还是服务器。

使用时需注意：

1. 机器人控制器作为服务器，默认系统启动就打开，程序中不需要打开；但需设置一个的端口号，并在视觉设备上连接按这个端口设置连接。此时作客户端的视觉系统必须在机器人程序运行之前打开，否则报警提示。控制器作服务器时，最多支持 5 个客户端连接。



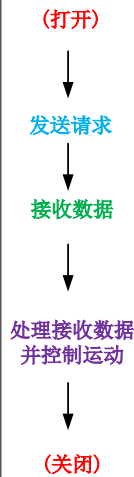
2. 机器人作客户端，需要在程序中利用 Open Socket 指令选择连接的 IP。

3. 设置的本地服务器的端口号范围 1024~9999，其中 3333 不可用。

示例用法如下：

机器人作为客户端，指令中设置本地端口号1026
外部视觉作服务器，【通讯设置】设置服务器端口号1025

```
START;
PRO=(0,0,10,0,0,0);
TxBuf = "TA";
RxBuf = "";
L[2]:
Movj P[0],V[30],Z[0];
L[0]:
Open Socket("10.44.53.13",1025,1026,B0);
If B0 == 0
  Goto L[0];
EndIf;
SetPorBuf(TxBuf);
Send Port[1026];
L[1]:
Get Port[1026], T[10], Goto L[1];
RxBuf = GetPorbuf(0,100);
B1 = StrGetData(RxBuf,"#",P10);
Cnvrt(P[10],P[20],World);
Movl Offset(P[20],PRO),V[30],Z[0];
Set Out[1],ON,T[0];
Delay T[1];
Jump P[1],V[100],Z[0],LH[10],MH[-750],RH[10];
Set Out[1],OFF,T[0];
Delay T[1];
Goto L[2];
Close Socket,1026;
END;
```



机器人作服务器，【通讯设置】设置服务器端口号1026
外部视觉作客户端，端口号1026

```
START;
TxBuf = "TA";
RxBuf = "";
L[2]:
Movj P[0],V[30],Z[0];
SetPorBuf(TxBuf);
Send Port[1026];
L[1]:
Get Port[1026], T[10], Goto L[1];
RxBuf = GetPorbuf(0,100);
B1 = StrGetData(RxBuf,"#",P10);
Cnvrt(P[10],P[20],World);
Movl Offset(P[20],PRO),V[30],Z[0];
Set Out[1],ON,T[0];
Delay T[1];
Jump P[1],V[100],Z[0],LH[10],MH[-750],RH[10];
Set Out[1],OFF,T[0];
Delay T[1];
Goto L[2];
END;
```

b) 时间日期

显示控制器中的当前时间，能通过“设置时间”按钮更改。

通讯设置	时间日期	用户设置	自定义报警	其他设置
日期: <input type="text" value="2017"/> - <input type="text" value="04"/> - <input type="text" value="27"/> 时间: <input type="text" value="09"/> : <input type="text" value="57"/> : <input type="text" value="21"/> <input type="button" value="设置时间"/>				

c) 用户设置

用户设置是选择不同的用户模式，在不同用户模式下拥有不同的操作权限，如下表所示。

操作内容	使用权限			
	用户模式	编辑模式	管理模式	厂家模式
手动示教	支持	支持	支持	支持
修改、编辑程序	不支持	支持	支持	支持
运行程序	支持	支持	支持	支持
机器人设置	不支持	不支持	不支持	支持
零点设置	不支持	不支持	支持	支持

坐标系设置	不支持	不支持	支持	支持
运动参数	不支持	不支持	支持(示教/运行的速度、加速度除外)	支持
外设配置	不支持	不支持	支持	支持
通讯设置	支持(仅示教器通讯)	支持(仅示教器通讯)	支持	支持
时间日期	不支持	不支持	支持	支持
自定义报警	不支持	不支持	支持	支持
屏幕校准	不支持	不支持	支持	支持
屏幕翻转	不支持	不支持	支持	支持
亮度及屏保	不支持	不支持	支持	支持
摇杆校准	不支持	不支持	不支持	支持
配置文件备份	不支持	不支持	支持	支持
配置文件加载	不支持	不支持	支持	支持
程序备份	不支持	不支持	支持	支持
程序加载	不支持	不支持	支持	支持
示教器更新	不支持	不支持	不支持	支持
控制器更新	不支持	不支持	不支持	支持
恢复出厂设置	不支持	不支持	不支持	支持
SD卡格式化	不支持	支持	支持	支持
清除历史报警	不支持	支持	支持	支持
模式切换	不支持	支持	支持	支持
功能扩展	不支持	不支持	支持	支持

更改用户模式界面如图所示，左边用户列表中选择用户模式，在右边输入登录密码，点击登录。登陆成功后，下方弹出密码修改界面，同时右上角控制工具栏更新用户模式。通讯连接后，系统默认为客户模式登陆。客户模式不需要登录密码。

d) 语言

可中英文切换，需要重启生效！

e) 自定义报警

可以设置 16 个自定义报警，每条报警最多 40 个字符。

通讯设置	时间日期	用户设置	自定义报警	其他设置	
报警号		报警描述			
0	<input type="text" value="NULL"/>				
1	<input type="text" value="NULL"/>				
2	<input type="text" value="NULL"/>				
3	<input type="text" value="NULL"/>				
		上一页	1/4	下一页	

在编程时，可利用 Alarm 指令调出报警。报警时，报警日志中也会同步记录。通过右上角的



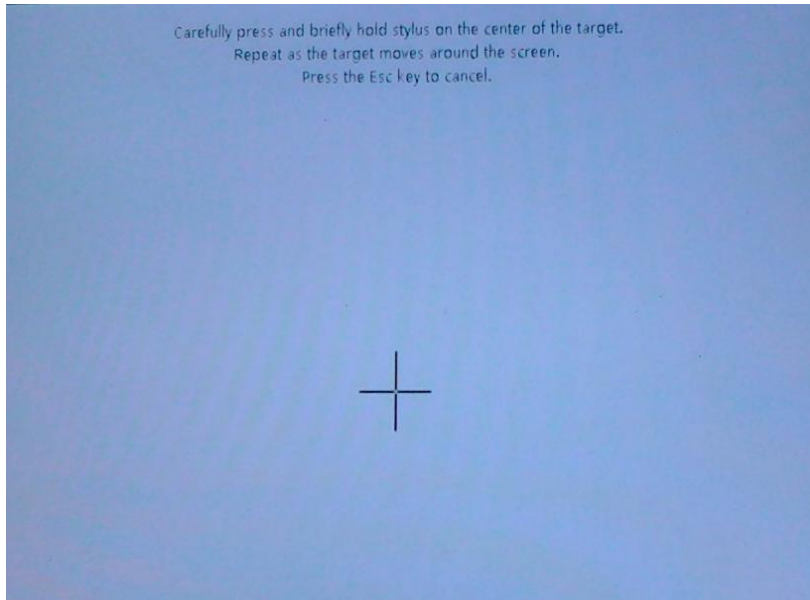
按钮可清除该报警。

f) 其他设置

其他设置包括示教器设置、备份与加载、系统更新、系统还原、其它五类。其中第一类“示教器设置”仅针对 InoTechPad 示教盒版生效，PC 版无效。

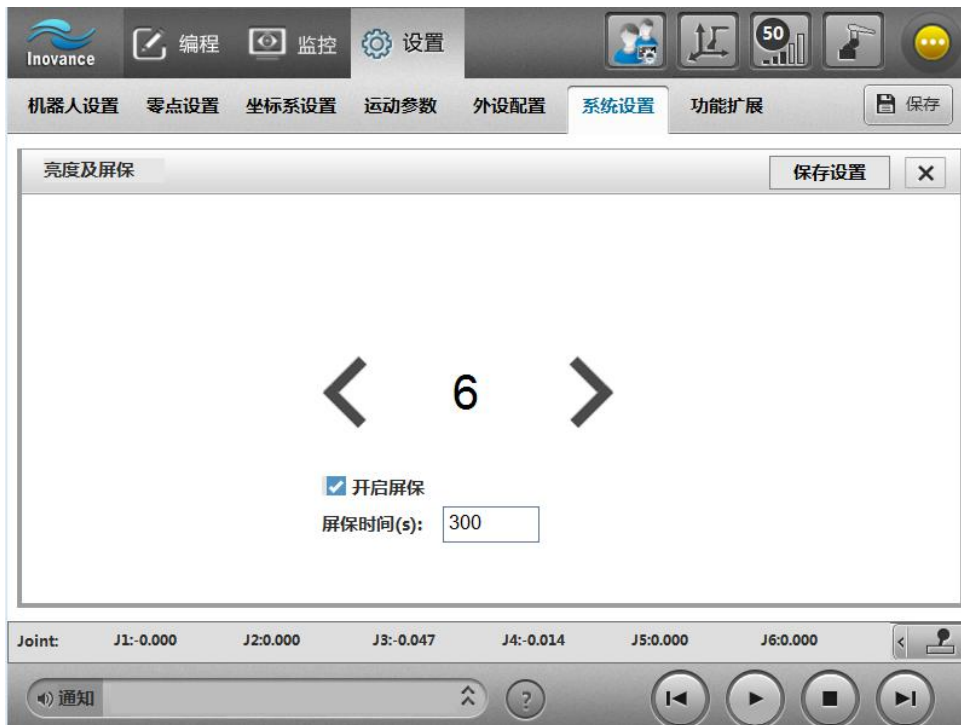
通讯设置	时间日期	用户设置	自定义报警	其他设置		
示教器设置		备份与加载		系统更新	系统还原	其它
<input type="button" value="屏幕校准"/>		<input type="button" value="配置文件备份"/>		<input type="button" value="示教器更新"/>	<input type="button" value="恢复出厂设置"/>	<input type="button" value="控制设备"/>
<input type="button" value="屏幕翻转"/>		<input type="button" value="配置文件加载"/>		<input type="button" value="控制器更新"/>	<input type="button" value="SD卡格式化"/>	<input type="button" value="机械锁定"/>
<input type="button" value="亮度及屏保"/>		<input type="button" value="程序备份"/>		<input type="button" value="一键升级"/>	<input type="button" value="清除历史报警"/>	
<input type="button" value="摇杆校准"/>		<input type="button" value="程序加载"/>				
		<input type="button" value="加载点文件"/>				

屏幕校准：屏幕校准界面如下，使用笔尖准确的点击“+”中心点。“+”依次出现在屏幕中央和四角位置，依次点击完成后，再次点击空白处即可回到原页面。



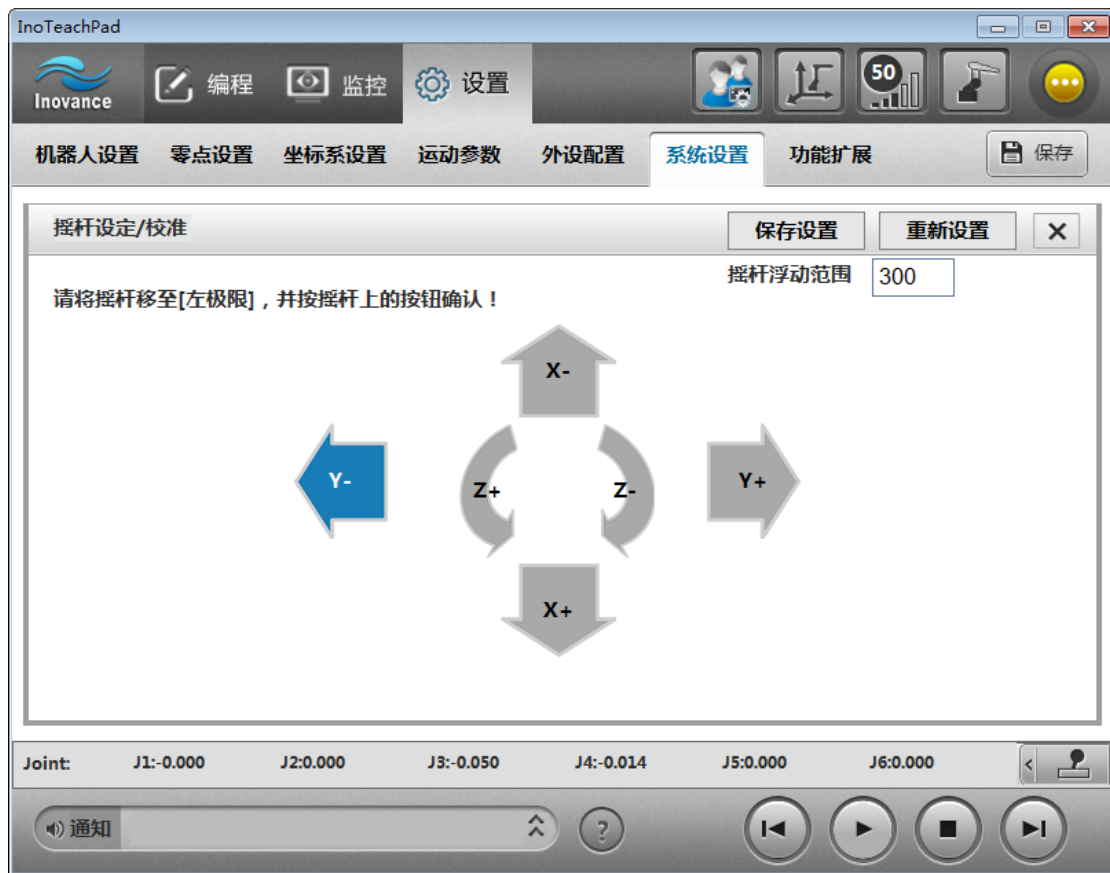
屏幕翻转：屏幕可以在左手屏幕和右手屏幕间切换，点击“屏幕翻转”按钮弹出确认窗口，点击“是”则屏幕翻转。注意：确认后系统会自动进入触摸屏校准界面，需用户再次校准屏幕。

亮度及屏保：亮度：共有 1~6 六个亮度等级。屏保：可选择性激活屏保功能，设置屏保时间。



摇杆校准：该功能用于厂家维护人员校准摇杆方向，需要厂家模式权限。摇杆运动方向包含左右运动 (Y-/Y+)、上下运动 (X-/X+)、顺/逆时针旋转 (Z-/Z+) 三个方向，依次按提示的方向和顺序操纵摇动即可。摇杆与坐标系方向的对应关系见 3.1.4 节操纵机器人移动。

摇杆浮动范围：用来在软件层面控制摇杆中间位置的敏感度，默认取 300。值越大，则防晃动能力越强；值越小，则轻微触动更敏感。建议不要轻易更改该值！

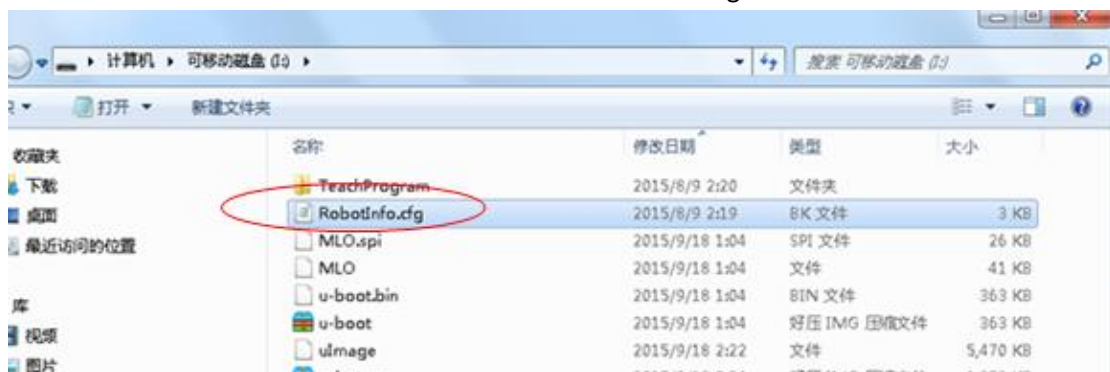


配置文件备份：将机器人配置文件（包含机器人设置、零点设置、坐标系、运动范围、运动特性的各项参数文件）备份到 U 盘中。

（1）在控制器上插入 U 盘，检查连接状态。若示教软件中监控的通讯状态显示“USB 控制器已插上设备并成功挂载”说明通讯良好。否则，请检查连接。



- (2) 点击“配置文件备份”按钮并确认操作。系统自动备份，完成后退出。
- (3) 完成后 U 盘上根目录下会新出现一个名为 RobotInfo.cfg 的文件，即为备份的配置文件。



配置文件加载：将 U 盘中配置文件加载到控制器中。

- (1) 同上，在控制器上插入 U 盘，检查监控页面 USB 连接状态。
- (2) 点击“配置文件加载”按钮并确认操作。系统自动加载，完成后退出并重新给控制器上电。

程序备份：将 SD 卡中的控制程序备份到 U 盘中。

- (1) 在控制器上插入 U 盘，检查连接 USB 和 SD 卡连接状态。若示教软件中监控的通讯状态显示“USB 控制器已插上设备并成功挂载”、“SD 卡插上并成功挂载”说明通讯良好。否则，请检查连接。

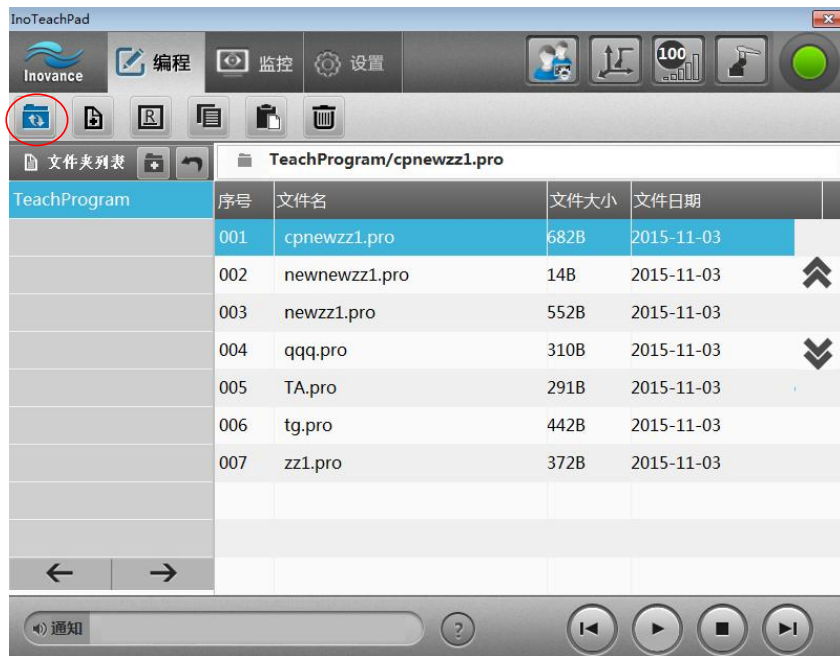


- (2) 点击“备份程序”按钮并确认操作。系统自动备份，完成后退出。
- (3) 完成后 U 盘上根目录下会新出现一个名为 TeachProgram 的文件夹，里面包含有备份的控制程序。



程序加载：将 U 盘中的程序加载到 SD 卡中。

- (1) 在控制器上插入 U 盘，检查连接 USB 和 SD 卡连接状态。同上，若示教软件中监控的通讯状态显示“USB 控制器已插上设备并成功挂载”、“SD 卡插上并成功挂载”说明通讯良好。否则，请检查连接。
- (2) 点击“加载程序”按钮并确认操作。系统自动加载，完成后退出。
- (3) 进入编程界面，点击左上角的“刷新”按钮，加载的程序即可显示于列表中。

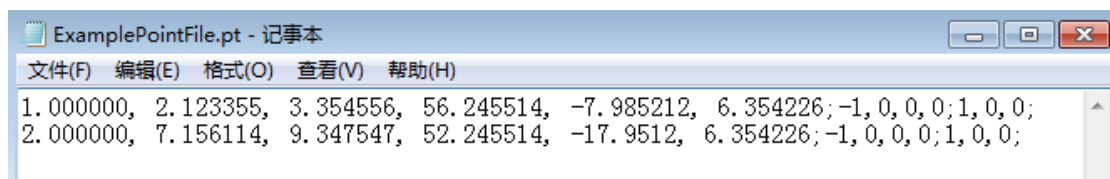


点文件加载：将 U 盘中的点文件加载到机器人控制系统中。点击按钮，即可浏览 U 盘中的目录选择点文件加载。

关于点文件：

点文件以“.pt”为后缀名。文件内容为位置变量的数据信息，一行代表一条位置变量信息。每行的格式参照“位置变量”的定义。每行分为 3 小段，前 6 个参数为第一段，为机器人的坐标系值；中间四个参数为第二段，为臂参数；后三个参数为第三段，分别为坐标系号、工具号、用户号。段与段之间以“；”分隔，段内数字以“，”分隔。

一个示例如下所示：



示教盒更新：

这是一种旧有的升级方式，利用示教器上的 USB 接口升级。S01.15 及以上版本推荐使用一键升级功能。

确认示教软件放在 U 盘\InoTeachPad_ce\CE 目录下，然后即可在示教盒上插入 U 盘，点击示教盒更新，完成后重启示教盒即可。（示教盒 USB 接口位于示教盒上方；PC 版不能使用此法）



控制器更新:

这是一种旧有的升级方式，利用控制器上的 USB 接口升级。S01.15 及以上版本推荐使用一键升级功能。

控制器更新程序直接放到 U 盘根目录，在控制器上插入 U 盘再点击更新，完成后重新连接即可。

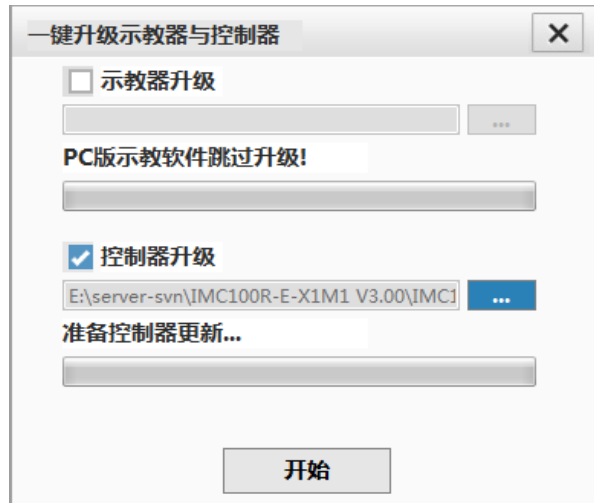


一键升级:

推荐 S01.15 及以上版本使用此方法进行升级。

操作方法:

- 对于示教器版本的示教软件，将带有升级包的 U 盘插入示教器 USB 接口，即可进行升级。对于 PC 版本示教软件，只能进行控制器升级（控制器升级包位于本地电脑或 U 盘均可）。
- 通过复选框勾选，可选择性升级示教器或控制器，也可同时升级示教器与控制器。
- 通过浏览按钮选择文件路径。示教盒版本示教器默认 U 盘目录下，PC 版本示教器默认电脑我的文档。
- 选择完成后，点击“开始”即可进行升级。
- 升级过程信息会显示在进度条上方。



注意：控制器升级过程中请勿断电，否则可能造成异常，只能对控制器进行刷机才可恢复！

恢复出厂设置：将所有设置全部初始化为出厂状态。

SD 卡格式化：将 SD 卡格式化成适合机器人控制器上使用的程序存储卡。因为若 SD 卡中原有程序，该操作会清空 SD 卡内的程序，所以建议操作前先备份程序。

(1) 检查 SD 卡通讯状态。若示教软件中监控的通讯状态显示“SD 卡插上并成功挂载”说明通讯良好。否则，请检查连接。



(2) 点击“备份程序”按钮并确认操作。系统自动备份，完成后退出。

清除历史报警:

清除监控中的日志和报警记录。

控制设备:

点击【控制设备】按钮，弹出界面，可选择哪个设备拥有对控制器的控制权。



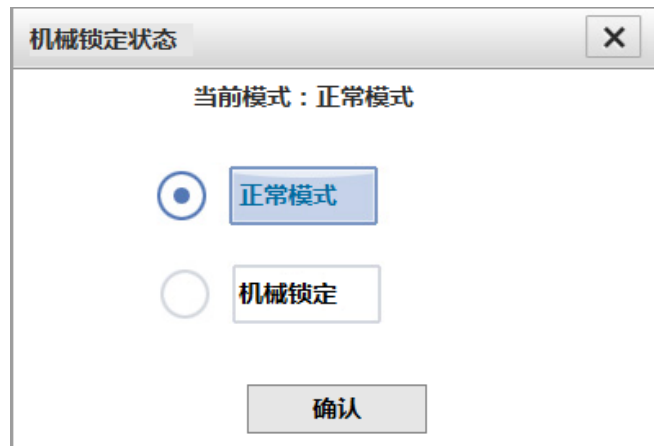
当控制权不在示教器上时，示教软件工具栏出现一个“锁”的标识。将不能操作示教器控制控制器，比如修改坐标系参数、运行程序等，此时示教器只能起到观测作用。



当控制权在“远程 IO 单元”或“远程 Modbus 单元”时，需要在启动速度栏指示程序第一次运行的启动速度百分比。通过外部 IO 修改速度后，（工位预约模式下的速度加、速度减），该值失效，机器人以设置的速度运行。通常该启动速度低于 100，在诸如工位预约模式下，确保第一次慢速运行。

机械锁定

在机械锁定模式下，示教或运行时，机器人不运动，但其他功能都正常。



错误信息保存

将控制器的错误信息保存到 SD 卡中。

错误信息导出

将 SD 卡中的错误信息导出到控制器的 USB。

3.2.7 功能扩展

a) 视觉标定

用于确定视觉坐标系与机器人相关坐标系的关系。详见 4.5 节视觉标定

b) 码垛工艺设置

在码垛工艺设置界面，可以查看并新增垛型。垛型用于在【监控】-【托盘变量】中进一步发展为托盘变量。详见 4.3 节码垛工艺

c) 跟随工艺设置

用于配置直线型传送带和圆盘形转台的跟随工艺，配合与外部相机的交互，实现跟随动态抓取。详见 4.4 节跟随工艺。

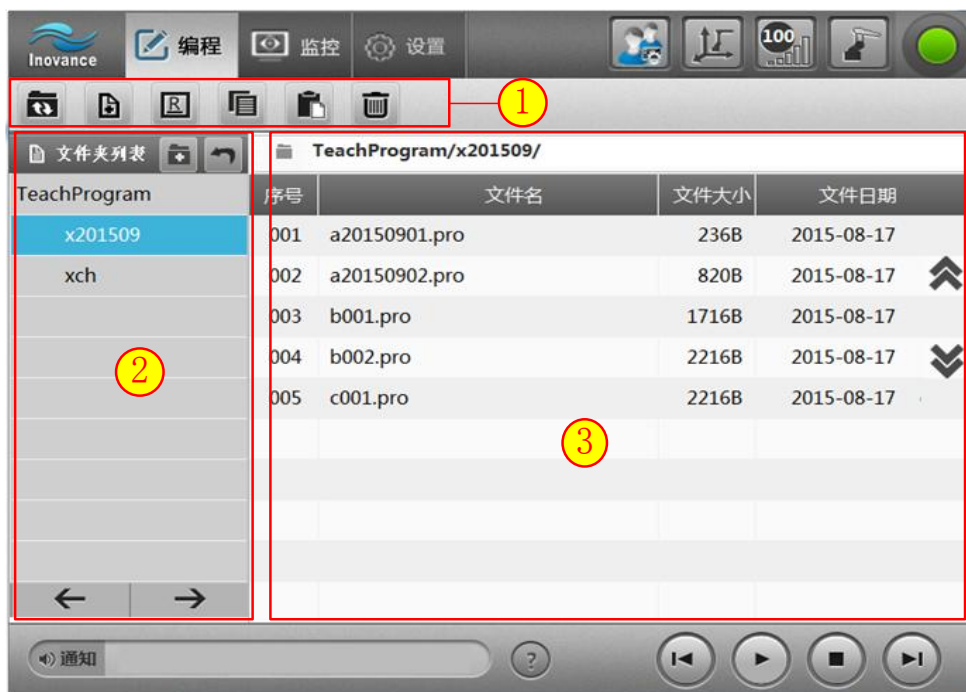
d) 锁螺丝工艺

设置螺丝锁紧过程的工艺参数，配合锁螺丝相关指令编程，实现锁螺丝动作。详见 4.6 节锁螺丝工艺。

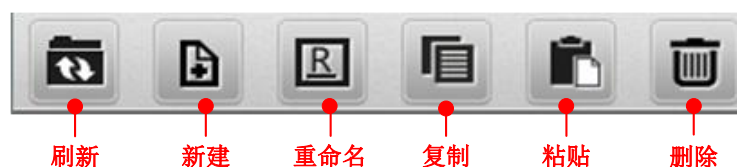
3.3 编程与运行

3.3.1 编程面板介绍

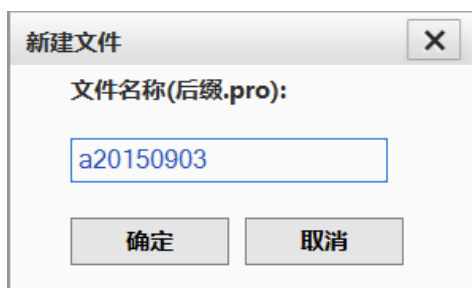
编程/运行面板如下图所示，面板左侧标号②部分为文件夹列表，右侧标号③部分为程序文件列表，上方标号①部分为文件编辑工具条。



文件编辑工具栏：通过文件操作工具栏中的工具可新建、删除程序文件，也可对已有的文件进行复制粘贴，双击程序列表中的文件名可以将其打开。



点击新建按钮创建一个新的程序文件，弹出如下窗口：



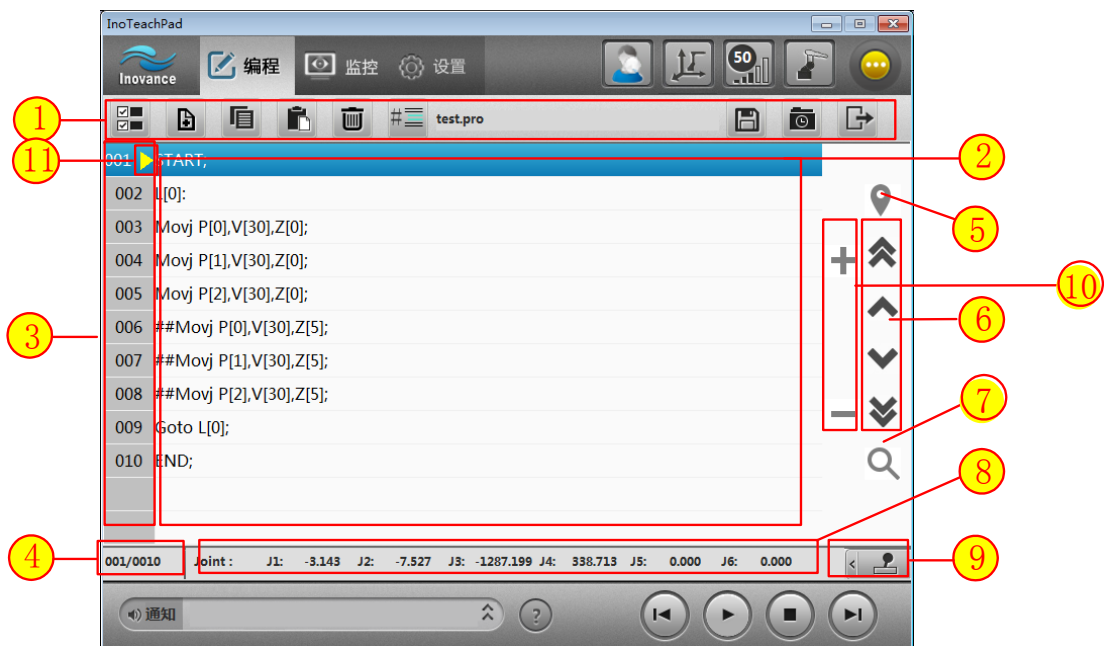
在文件名文本框中输入程序文件的名称，文件名只能由字母、数字以及下划线组成，且首位必须为字母，长度不得超过 32 个字符。仅仅大小写不同的文件会被认为是同名文件，因此会禁止使用大小写名称不同的文件。

输入文件名后点击确定完成文件创建，双击文件名称进入程序编辑面板。

文件夹列表：显示全部文件夹，使用文件夹便于对程序进行分类管理，每个文件夹中可包含多个程序文件。点击“新建”可新建一个文件夹，单击某个文件夹可显示该文件夹下的程序列表，双击可进入该文件夹，点击“返回”按钮退出当前文件夹。文件夹名称要求与程序名类似，只能由字母、数字以及下划线组成，且长度限定为 16 个字符。文件夹嵌套最多三层



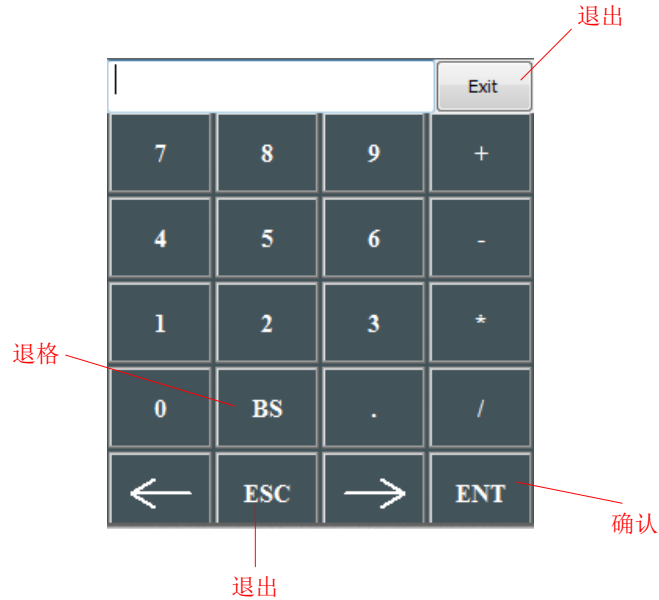
程序列表: 显示当前文件夹下的所有程序，程序按字母 a-z，数字 0-9 的顺序排列，双击某个程序可以将其打开。



编号	功能	描述
1	程序编辑工具条	对程序文件进行编辑操作，详细说明见 3.3.4 程序编辑
2	程序指令编辑区	该区域显示程序指令的具体内容，单击可选中某行，被选中的行变为蓝色。双击某行可修改该行指令。
3	程序行号区	显示每条指令所在行的行号。
4	当前行/总行数	显示被选中的命令行的行号和该程序当前包含的总行数。
5	定位按钮	点击该按钮，弹出下图所示窗口，输入行号，光标可跳转到指定行。
6	页面滚动条	点击单箭头光标跳转一行，点击双箭头，程序翻页。
7	搜索/替换按钮	搜索或替换程序中的内容

8	坐标显示区	显示机器人当前坐标信息。
9	示教面板按钮	点击该按钮，调出示教控制面板。如图所示，示教面板的使用说明参见 3.3.2.
10	缩放按钮	编辑区的缩放
11	启动行标识	以黄色箭头标示启动行，用来表明程序的启动位置。

定位按钮弹出小键盘如下图所示：



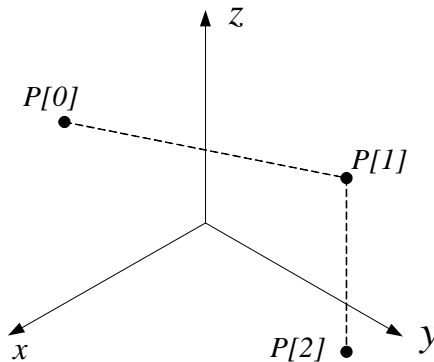
关于黄色箭头——启动行：

启动行，用来表明程序的启动位置。在程序运行过程中，启动行会随着程序选中行（蓝色选中状态）同步向后执行。但非运动时，人为点击列表，程序选中行变更，启动行不会变更。


3.3.2 指令编辑

指令编辑器提供了机器人编程所需的全部指令以及语法格式，免去使用键盘输入的麻烦，降低用户的开发难度。文档以以下任务为例，讲解示教编程过程，相关指令的详细描述参考第 2 章。

任务描述：如图所示，P[0]、P[1]、P[2]是空间中的三个点，要使机器人末端完成以下轨迹：P[0]- P[1]- P[2]- P[1]- P[2]- P[1]- P[0]。



首先通过 3.3.2 中的示教方法将机器人末端移动到 P[0]点，在程序编辑工具条中点击“添加”

按钮弹出指令编辑器界面，如图所示。图中右边部分给出了常用指令，左边部分对指令进行了分类，点击指令类别按钮，右边将显示该类型的全部指令。



当点击某个指令时，将显示该指令的语法结构和参数列表，此处选择“运动指令”->“Movl”，显示如下界面。



通过 Movl 指令将机器人移动到期望位置点，完成该指令的编辑需要以下四步：

第一步：示教期望位置

通过示教将机器人移动到期望的点 P[0]（上文已完成），图中框①显示了机器人当前位置的坐标。

第二步：添加位置点变量

图中框②部分可添加/修改位置点变量，点击“新增点”当前位置点 P[0]被保存为新的位置变量；选中某个已有的位置变量后点击“修改点”，位置变量更新为当前位置数据。

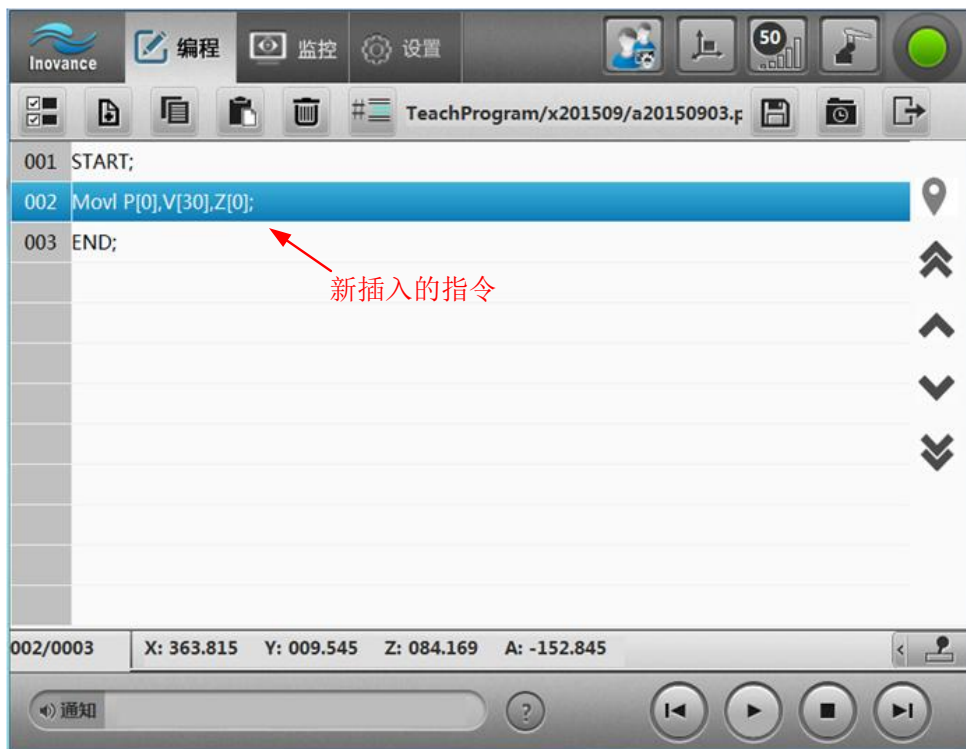
注意：在非关节坐标系下取奇异点会出现报错！

第三步：指令参数编辑

如图中框③所示，指令列表已经给出，用户只需修改参数值，“[]”内为用户可修改的参数值，参数值可以是数值或变量。每个参数的含义与使用参考第 2 章。

第四步：插入指令

点击框④中的保存，编辑的指令被插入到当前行的下一行，如图所示。



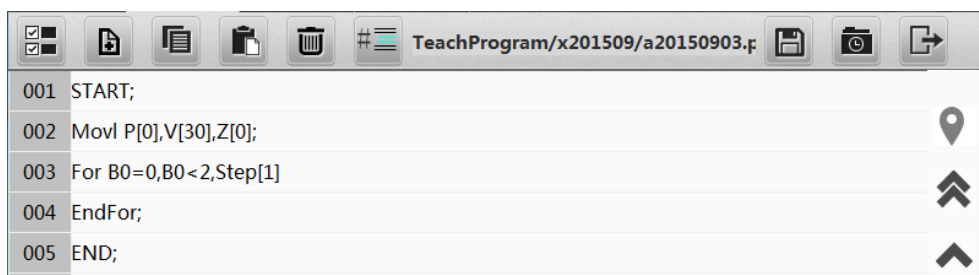
执行图中指令机器人可实现从当前位置运动到 P[0]点的任务。任务中需要在 P[1]、P[2]间循环两次，此处选择使用 For 指令。点击：流程控制指令->For，弹出 For 指令编辑窗口，如图所示：



For 指令有三个参数，分别点击各参数对其进行编辑，如图所示：



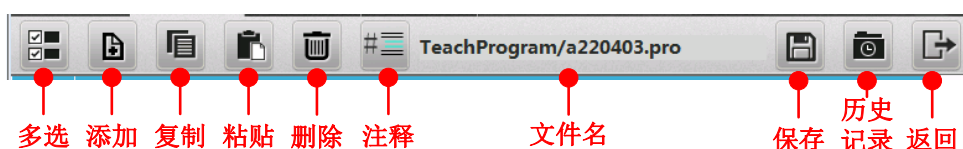
最后点击“保存”，For 循环指令插入到程序中，如图所示：



完成任务的剩下操作将在下节补充。

3.3.3 文件编辑

编写程序时不但需要插入新的指令，还需要对已有的指令进行编辑操作，通过编辑工具可实现编辑操作，编辑工具条如图所示：



多选：点击该按钮后，程序行号前出现多选框，可以选中多行程序。

添加：添加一条新的运动指令，点击该按钮后出现指令编辑器窗口。在指令编辑器中选择、编辑要插入的指令。

复制：复制选中的一条或多条指令。

粘贴：将复制的指令插入到当前行的下一行。

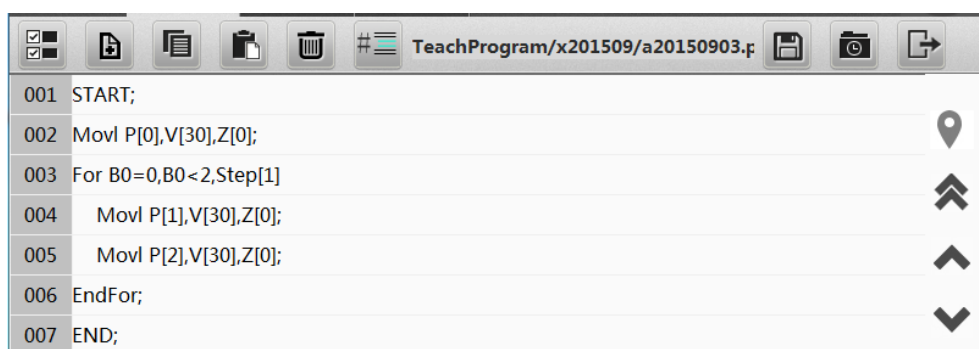
注释：注释掉一条语句，使得该语句无效，对已注释的语句施放会取消注释

保存：将程序指令保存到机器人控制器中。

历史记录:显示最近打开的文档。

返回：返回到程序列表面板。

接着按照 3.3.3 中的操作添加到达 B 点和 C 点的运动指令，如图所示：



任务中机器人需要从 C 点返回 B 点，指令与上图中第 4 行指令相同，最后回到 A 点，与第 2 行指令相同，可以选择使用复制/粘贴工具。

第一步：单击选中要复制的指令，点击复“复制”按钮

第二步：选中粘贴位置的上一行，点击“粘贴”按钮，如图所示。

```

001 START;
002 Movl P[0],V[30],Z[0];
003 For B0=0,B0<2,Step[1]
004   Movl P[1],V[30],Z[0];
005   Movl P[2],V[30],Z[0];
006 EndFor;
007 Movl P[1],V[30],Z[0];
008 Movl P[0],V[30],Z[0];
009 END;

```

程序中使用了循环语句，如果不使用循环，可将三行指令复制，此时需要用到多选，点击“多选”按钮，行号前出现多选框，此时可选中多行指令，如图所示。

```

 001 START;
 002 Movl P[0],V[30],Z[0];
 003 For B0=0,B0<2,Step[1]
 004   Movl P[1],V[30],Z[0];
 005   Movl P[2],V[30],Z[0];
 006 EndFor;
 007 Movl P[1],V[30],Z[0];
 008 Movl P[0],V[30],Z[0];
 009 END;

```

选中后进行复制/粘贴操作，并将 For 指令去掉，也可实现任务中的运动轨迹，如图所示

```

001 START;
002 Movl P[0],V[30],Z[0];
003 ##For B0=0,B0<2,Step[1]
004   Movl P[1],V[30],Z[0];
005   Movl P[2],V[30],Z[0];
006 ##EndFor;
007 Movl P[1],V[30],Z[0];
008 Movl P[2],V[30],Z[0];
009 Movl P[1],V[30],Z[0];
010 Movl P[0],V[30],Z[0];
011 END;

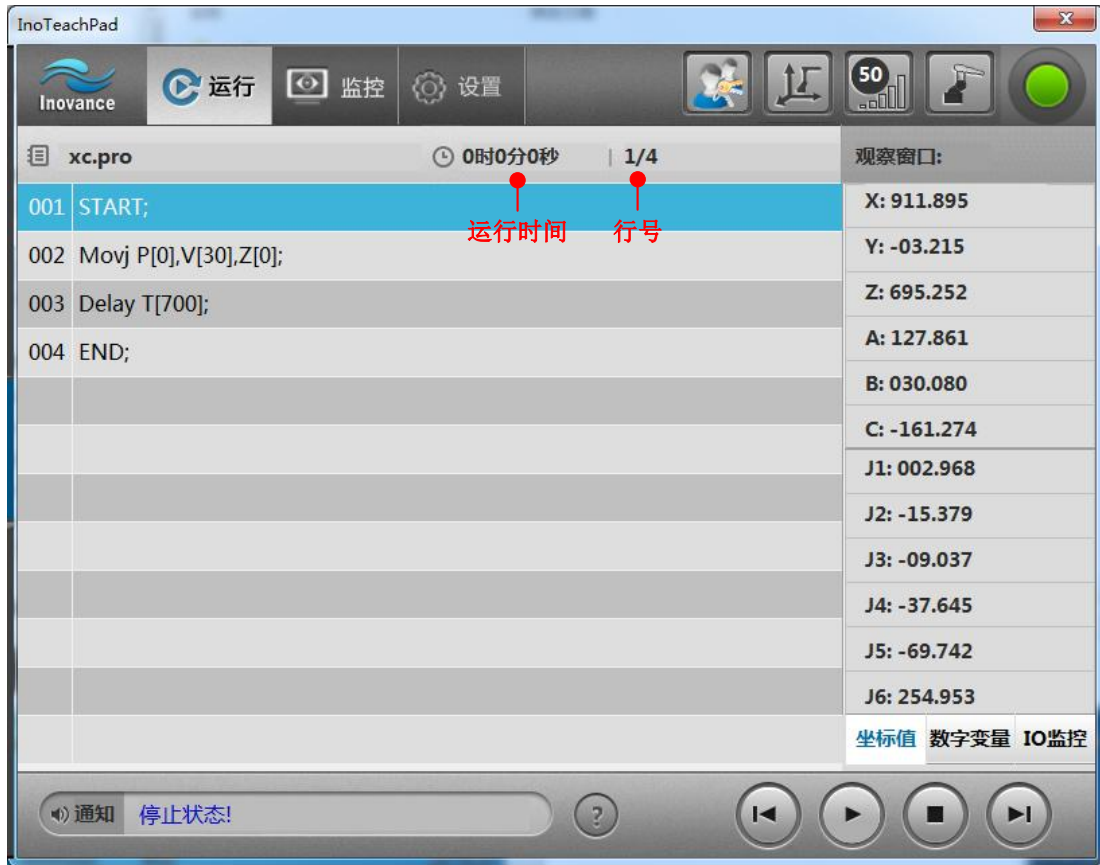
```

编辑完成后保存程序。

3.3.4 程序运行

在运行前，建议在编辑状态下调试程序。注意：示教编程模式下，程序的连续运行、单步前进、单步后退均为点动模式。调试完成后，切换到运行模式，即可运行当前程序。界面如图所示，左边显示程序，右边显示运动过程中机器人关节角度和基坐标点（机器人本体末端在

基坐标系下位姿)的变化,上方显示当前的运行时间和所处的行号,可通过右下方的运动控制按钮控制程序的启停。



运行时间:是指启动到停止所用的时间。而非仅仅单个程序运行时间,这在多程序连续运行(如工位预约、主子程序调用)非常有用,提供运行的总时间。

3.4 监控

3.4.1 变量监控

通过变量监控面板可以监控程序运行过程中各变量的值,变量包括全局数值变量、局部数值变量、位置变量,全局平移变量和局部平移变量,变量在程序中的使用说明参考第2章。列表头为蓝色时,表明该列表被选中,可通过右侧翻页控制按钮选择性翻页。



a) 全局数值变量和局部数值变量

双击某个变量弹出如下窗口，此时可修改变量的值。



b) 位置变量

可替换、添加、删除、重命名位置变量，如图所示。

选中某个位置变量，点击“替换”位置变量的值被机器人当前位置替换。

点击“添加”按钮，可添加一个新的位置变量，变量值为机器人当前位置。



运行到位置变量：

选中该行变量，点击下方的“运行”按钮即可运行到位置变量处。

双击某个变量，可对其进行修改，如图所示。



位置变量监控特性：

1. 暂停或停止时，监控窗口显示的位置变量才刷新。
2. 由于程序的预进，观测到的位置变量值可能存在超前。

c) 全局平移变量和局部平移变量

双击某个变量弹出如图所示窗口，可对变量进行修改。平移变量的值可在框中直接输入，也可在通过“取点 1”、“取点 2”，然后点击“计算平移量”得到。单击“确定”，参数被保存。

PR001

参考点一 取当前点1 参考点二 取当前点2

J1: 14.967 J2: -27.890 J3: -2450.510 J1: 26.424 J2: -18.039 J3: -2450.510

J4: -349.780 J5: *** J6: *** J4: -349.780 J5: *** J6: ***

Crd: 1 Tol 0 Usr 0 Crd: 1 Tol 0 Usr 0

PR000 计算平移量

J1: 11.456 J2: 9.851 J3: 0.000 Crd: 1 Tol 0 确定

J4: 0.000 J5: 0.000 J6: 0.000 Usr 0 取消

d) 托盘变量

托盘变量包含着托盘可以复制、粘贴、删除、修改对应的托盘变量。（托盘变量与平移变量类似，一共存在 256 个托盘变量可供使用）。

InoTeachPad

Inovance 编程 监控 设置

变量 IO监控 通信状态 伺服状态 日志 版本信息

全局数值变量	局部数值变量	位置变量	全局平移变量	局部平移变量	托盘变量	
托盘序号	托盘名	层数	每层个数	奇偶层反向	标签朝外	垛型名
000	0	2	18	0	0	PM_A_18_189
001	--	--	--	--	--	--
002	--	--	--	--	--	--
003	--	--	--	--	--	--
004	--	--	--	--	--	--
005	--	--	--	--	--	--
006	--	--	--	--	--	--
007	--	--	--	--	--	--
008	--	--	--	--	--	--
009	--	--	--	--	--	--

Joint: J1:66.763 J2:-44.959 J3:-791.839 J4:-257.366

通知 ? [Back] [Play] [Stop] [Next]

双击一个空的托盘变量，会跳转到“垛型选择”页面；选择完成，点击“新建托盘”后，则进入托盘变量的编辑界面。

关于更多托盘变量的信息详见 4.3 节码垛工艺。

3.4.2 IO 监控

IO 监控面板可以监控/改变系统 IO 的状态，分为数字信号监控和模拟信号监控两部分。系统会根据开机连接时读取的 IRLink 模块信息来显示 IO。

a) IN/OUT

界面左侧列表为输入数字（IN）信号，右侧列表为输出数字（OUT）信号。每个输入数字信号（In[***]）的状态都在当前页面中显示。



翻页：

列表头为蓝色时，表明该列表被选中，可通过右侧翻页控制按钮选择性翻页。

输入数字信号的强制控制：

开启强制模式，模拟输入信号变更为人为设置状态。此时，信号的状态与外部接线的实际输入无关。当恢复回“不强制”状态，输入信号的状态又返回到外部接线输入的实际值。

在使用时，先点击列表中对 In 信号的强制开关，使之变更为“强制”；然后即可点击状态，人为切换 ON/OFF。

注意：IN[000]-IN[003]为系统 IO，分别代表急停、使能、切换，用户更改无效。

输出信号的状态控制：

一般情况下，直接点击对 Out 信号的状态项，切换 ON/OFF。

切换需要 IO 控制权状态为 RC_ACTIVE，否则状态显示为灰色，尝试在示教器人为更改状态时会失败，并提示缺少控制权。这些操作都会变更 Out 控制权：利用 InoRobShop 配置 IRLink 端口控制权为 PLC；在示教器外设配置-I/O 配置中为某项功能关联了输出信号。详情见 5.3.3 IO 权限管理。

b) AD/DA

界面左侧列表为输入模拟量（AD）信号，右侧列表为输出模拟量（DA）信号。每个模拟量信号的状态都在当前页面中显示。

输出类型：配置模拟量信号的类型是电流还是电压。

范围：模拟量信号的范围，根据 IRLink 产品型号，每个模拟量端口都有几种选定范围。

状态：模拟量的参数值。当为电压信号时，以 V 为单位；当为电流信号时，以 mA 为单位。

点击状态项，即可指定输出值。

在示教器【设置】-【外配置】-【IRLink 配置】中，或 InoRobShop 等其它软件中，可以更改模拟量端口的属性（包括这里监控中显示的输出类型、范围），但需要重启生效。

输出模拟量信号的开关：用于控制输出信号是否被设置为状态里指定的值。当开关为 OFF 时，即使状态值被变更，实际也不会生效。点击即可切换开关。

与 OUT 类似，DA 信号的状态变更，也需要该 DA 的 IO 控制权为 RC_ACTIVE。关于 IO 控制权，详见 5.3.3 IO 权限管理。



3.4.3 通信状态



该界面可查看系统硬件的状态，如图所示，包含以下几种情况：

名称	状态分类	名称	状态分类
EtherCAT1	网线未连接	EtherCAT2	网线未连接
	动态 IP: XX.XX.XX.XX		静态 IP: 192.168.23.25
	被禁用		被禁用
	未定义状态		未定义状态
	信息获取失败		信息获取失败
控制器 USB	设备未连接	SD 卡	设备未连接
	已连接并成功挂载		已连接并成功挂载
	已连接但挂载失败		已连接但挂载失败
	未定义状态		未定义状态
	信息获取失败		信息获取失败
EtherCAT1	通信正常	IR-link1	通信正常
	从站掉线		从站掉线
	网线未连接		网线未连接
	连接了非 EtherCAT 设备		连接了非 IR-link 设备
	被禁用		被禁用
	未定义状态		未定义状态
	信息获取失败		信息获取失败

EtherNet2 直接连接时，如果 EtherNet1 接了外网，则能看到 EtherNet1 的动态 IP。

注意：在控制器上电时，不允许直接拔插 SD 卡。

3.4.4 伺服状态

在伺服状态面板中能实时监控每个伺服参数。



序号	当前值	参数名	备注
00	1020	伺服软件版本号	H0100
01	462850	伺服软件非标版本号	H0002
02	23110	编码器软件版本号	H0004
03	14101	电机型号	H0000
04	2	伺服型号	H0102
05	1	绝对位置模式	H0201
06	-1	实际电机转速	H0B00
07	0	转矩指令	H0B01
08	0	平均负载率	H0B12
09	-50740613	绝对位置反馈	H0B17

Joint: J1:66.763 J2:-44.959 J3:-791.839 J4:-257.366

3.4.5 日志

通过监控中的日志面板可以查看操作记录以及报警记录，如图所示。日志记录了每一次操作的内容与操作时间；报警记录了操作过程中出现的异常，日志和报警记录可以为故障诊断提供信息支持，机器人报警的处理方法参考附录三。



注意：清除报警是通过双击右上角的报警灯来清除。当出现限位报警时，可以通过反向运动清除。

3.4.6 版本信息

该界面可查看版本号，如图所示。



厂家模式下，能看到更多信息。

3.4.7 锁螺丝状态

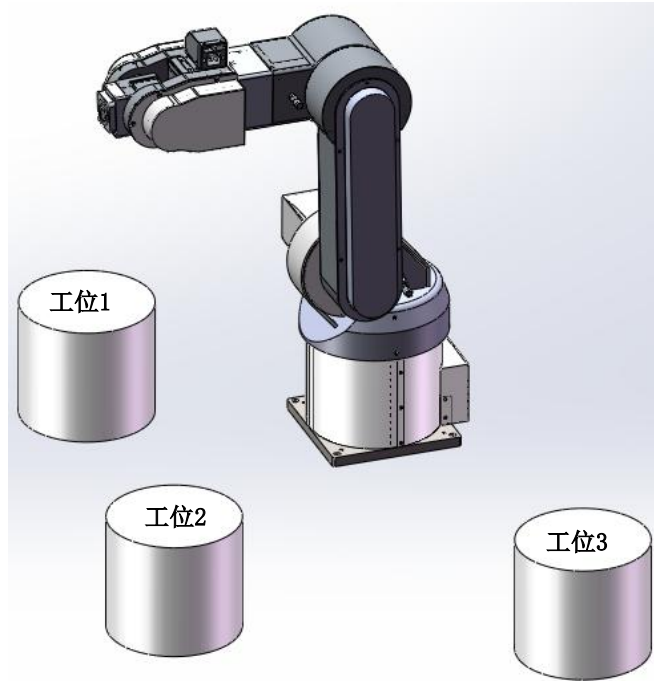
用于显示当前单个锁螺丝的过程记录和所有锁付操作的统计。详见 4.6 节锁螺丝工艺。



4 应用性功能

4.1 工位预约

工位预约是多个工位共用一套机器人控制器和示教器的解决方案。该项功能支持最多三个工位，对应有三个加工程序；当某个工位能发出加工请求时，就能利用工位预约控制机器人在该工位上加工。其它工位发加工请求，机器人则会在当前程序执行完后前往相应工位加工。当前程序执行完之前，多个工位发出请求，则机器人会按优先级处理。例如工位 1 先发出请求，随后另外两个工位 3、2 也先后发出请求，则系统会先运行第一个发出请求的程序（即工位 1），随后按优先级运行剩下的程序。（若工位 2 优先级高于 3，则先执行 2 再执行 3）



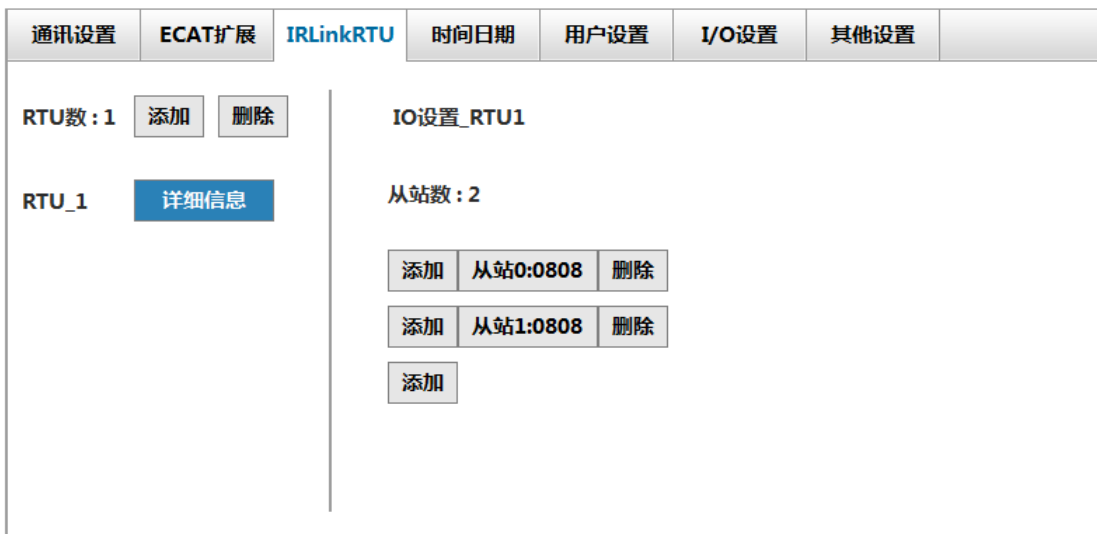
在实施前，提前设计外部控制盒上的按钮与对应的 IO 功能，以及对应的 IRLink 接线端口。然后连接硬件设备并在示教器上进行 IO 设置（在【设置】-【系统设置】-【IO 设置】窗口，详见 3.2.6 (f) I/O 设置）。最后切换到工位预约模式即可。

配置示例：

需求：机器人有 3 个工位。3 个工位协同运作，控制机器人运动。

实施：

1. 配置一个 IRLinkRTU，后接至少两个 0808 模块，在示教器的 IRLinkRTU 页面增加对应配置。

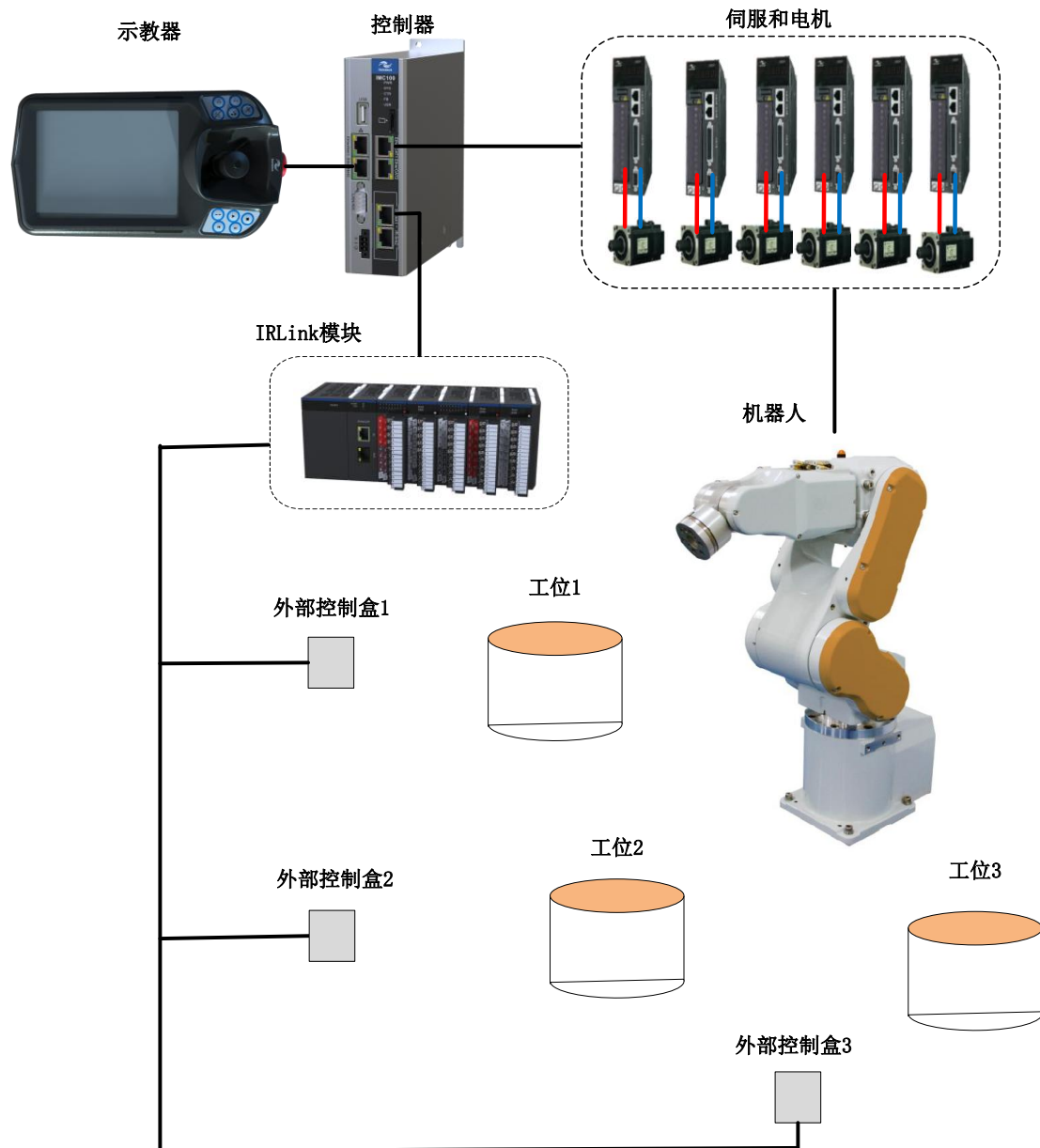


2. 依照下表接线，并进行 IO 设置。

IO 功能	IO 选项	IO 接线
急停	IN[3]	第一个 0808 模块的第 4 个输入端口
清除报警	IN[4]	第一个 0808 模块的第 5 个输入端口

启动	IN[5]	第一个 0808 模块的第 6 个输入端口
停止	IN[6]	第一个 0808 模块的第 7 个输入端口
暂停	IN[7]	第一个 0808 模块的第 8 个输入端口
工位程序 1	IN[8]	第二个 0808 模块的第 1 个输入端口
工位程序 2	IN[9]	第二个 0808 模块的第 2 个输入端口
工位程序 3	IN[10]	第二个 0808 模块的第 3 个输入端口
速度加	IN[11]	第二个 0808 模块的第 4 个输入端口
速度减	IN[12]	第二个 0808 模块的第 5 个输入端口

总体连接示意图如下：



3. 连接示教器，在正常模式下，在每个工位上进行示教，保存程序。在 IO 设置中选择 IO 子程序。

I/O 功能	<input type="button" value="上一页"/>	2页 [输入配置]	<input type="button" value="下一页"/>	I/O 选项
I/O子程序:/robot/TeachProgram/A.pro				IN[8] ▼
I/O子程序:/robot/TeachProgram/B.pro				IN[9] ▼
I/O子程序:/robot/TeachProgram/C.pro				IN[10] ▼
中断子程序:NULL				NULL ▼
中断子程序:NULL				NULL ▼

4. 在示教器【系统设置】-【其他设置】-【控制设备】中切换为“远程IO单元”，系统将自动切换到运行模式。按下外部控制的“启动”按钮，在某个工位需要加工时，触发其工位程序按钮即可。

注意事项：

- 一个工位正在被运行或处于待运行状态时，再次响应该工位会无效。
- 运行过程中不能进行控制设备的变更。
- 在工位预约模式下，示教软件只能进行监控和解除工位预约模式，不能进行其它操作。
- 工位预约启动速度是指开启工位预约后，默认的机器人运行速度，是一个百分比。通过外部IO修改速度后，（工位预约模式下的速度加、速度减），该值失效，机器人以设置的速度运行。工位预约的启动速度根据需求自由配置，默认值为100代表以最大速度的100%运行。
- 工位预约中的工位程序应当尽量避免死循环，但是一旦出现，则亦可以使用，比如在工位1的程序出现无限循环现象，则一直在此处循环，不进入工位2。

modbus 工位预约

通常的工位预约是IO接线直接控制的，只能控制“IO设置”中的内容。而利用modbus工位预约能达到更多的控制，如具体的速度值设定，读取伺服报警等等。

配置：

- 1、通过二次开发配置modbus从站。（见modbus具体配置说明）
 - 2、【设置-外设配置-I/O配置】中选择子程序路径，其后IO选项要选择非NULL，可以是未配置过的任意IN端口。
- 注意：使用modbus外设控制时，应避免再使用外部实际接线的IO控制，以免控制冲突。
- 3、【系统设置-其他设置-控制设备】中选择远程Modbus设备。

操作：

参照《modbus从站地址表》使用控制命令控制。modbus从站地址表相比示教器上的IO设置能提供更多的控制功能。Modbus从站地址表：（详见附录三）

4.2 API 功能

API 是外部设备作为客户端，与机器人控制器作服务器通讯接口，外部设备可以利用这些 API 监控或设置机器人的状态。

API 使用的前提条件

应用端的利用 API 控制机器人或变更参数需要获取控制系统的控制权。否则只能利用 API 读取、监控机器人状态。关于系统控制权详见 5.3.1 权限管理。

API 使用

VS 平台 C++语言应用 API 简介：

- 1、启动 VS，新建一个 C++工程。
- 2、将提供的动态链接库（IMC100API.dll）、头文件（IMC100API.h）和 lib 文件（IMC100API.lib）复制到工程文件夹中。
- 3、在应用程序文件中加入头文件的声明，`#include "IMC100API.h"`。
- 4、用户就可以在工程中调用 API 库中的任何函数，开始编写应用程序。

API 指令详见附录三。

监控类应用

- 1、需要与目标机器人建立连接，调用 `Init_ETH()` 函数。
- 2、调用对应监控类函数即可。

控制类应用

- 1、通过示教器，选择控制设备为远程以太网设备。
- 2、与目标机器人建立连接，调用 `Init_ETH()` 函数。
- 3、调用 `Acqpermit()` 函数取得当前控制权，如果部分控制指令需要较高级别用户模式，则需要调用 `UserLogin()` 函数登陆到对应模式。
- 4、调用相关控制函数。

4.3 码垛工艺

码垛工艺是围绕托盘变量进行的设置、编程。托盘变量包含着一个码垛托盘相关的一系列信息，既包含垛型名、托盘层数、每层个数、奇偶特性等信息，也包含生成的托盘的放置点位置信息。托盘变量是依据垛型产生的，垛型指码垛托盘的一层排布规则，用户可根据自身需求，扩充垛型。

完整的码垛操作可分为三个步骤：

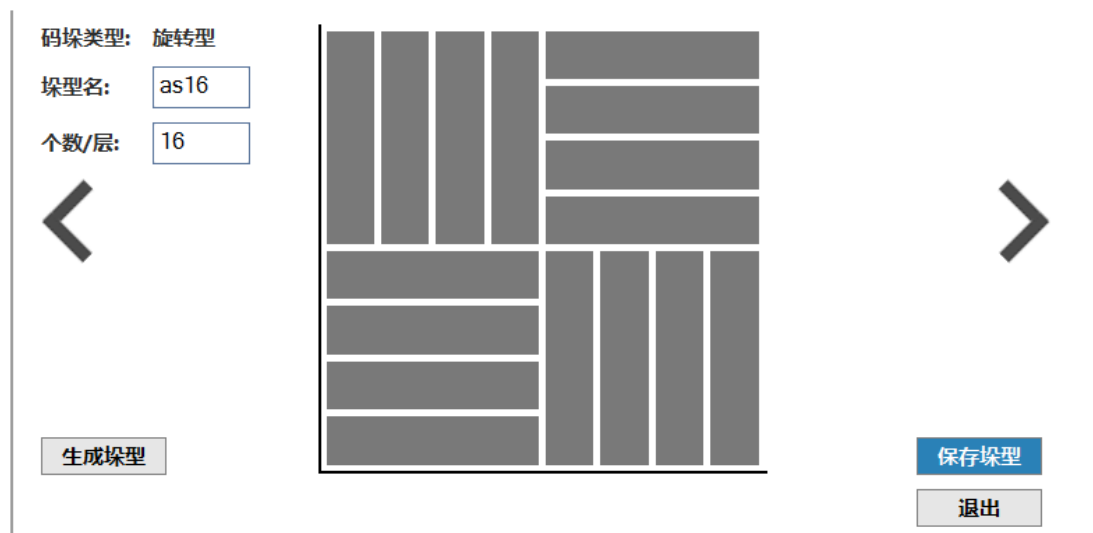
1. 新建垛型（若已存在满足需求的垛型，可跳过）
2. 编辑托盘变量
3. 码垛编程

4.3.1 新建垛型

新建垛型是在【设置】-【工艺设置】-【码垛工艺设置】页面操作的。在码垛工艺设置页面，能看到所创建的所有垛型。点击“新建垛型”按钮，即可开始新建。

垛型是基于两种基本垛型的，一种是旋转垛型，一种是阵列型。新建时按各自的方式生成垛型，保存后退出即可。

对于旋转型，几个货物平行排布成一堆，四堆一致的货物排成旋转样式。新建该种垛型时，只需填入每层个数及垛型名即可生成垛型。最终生成的垛型名会自动增加前缀“PM_R_”。如下图，新建一个每层 16 个货物的旋转型垛型。



对于阵列型，几个货物以竖直或水平方向平行的排成一列，多个具有各自特点的列组合成整个垛型。填写“新列数量”和“新列方向”，然后点击“新增一列”，则可完成一列的创建。按照这种方式，新增个多列，构成总体垛型。最终生成的垛型名会自动增加前缀“PM_A_”。如下生成一个三列的阵列垛型：

	第一列	第二列	第三列
列数量	4	2	4
列方向	水平	竖直	水平

码垛类型: 阵列型

垛型名: 42

新列数量: 0

新列方向: 水平

新增一列

删除一列

保存垛型

退出

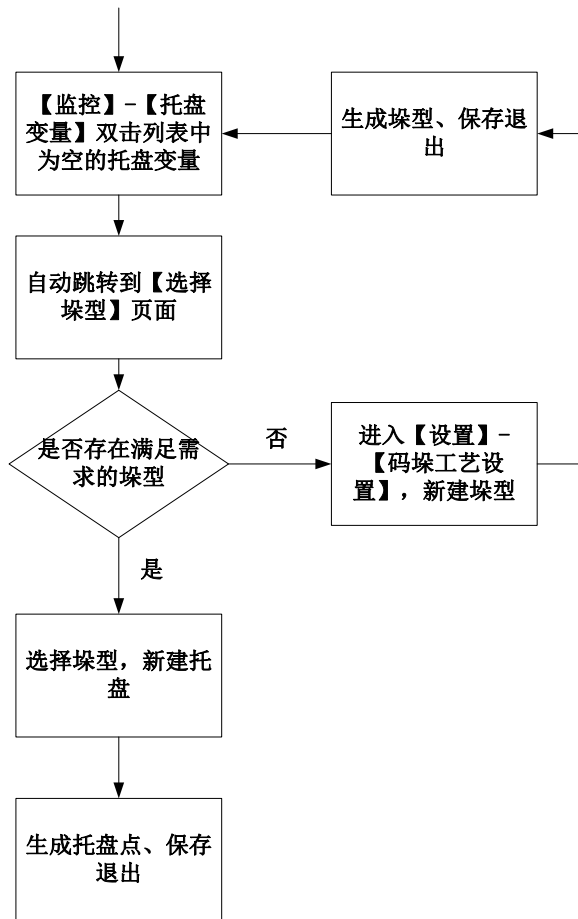
4.3.2 编辑托盘变量

对于编辑托盘变量，在【监控】-【托盘变量】页面完成。

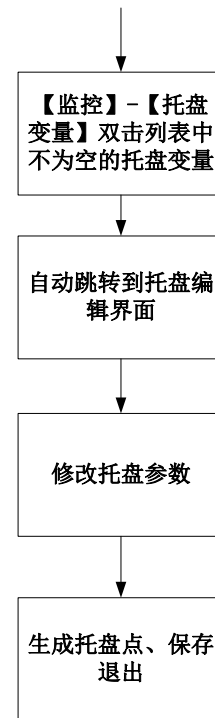
双击托盘变量，若该托盘变量为空，则自动新建托盘变量，此时需要选择垛型，然后依据该垛型配置托盘，最终生成托盘放置点。若现有垛型不满足需求，则应进入【设置】-【码垛工艺设置】页面，新增垛型。

若监控页面选择的托盘变量不为空，则自动进入托盘编辑页面。在该页面中修改托盘参数，重新生成托盘点。

新建托盘变量



修改托盘变量



注意：

修改托盘变量仅限于保持同一垛型的托盘修改，若需要更改垛型，则应在【监控】-【托盘变量】页面，删除该托盘变量，再新建托盘变量。

其中，生成托盘点的标准流程：

托盘校准-托盘设置（奇偶层、标签设置）-生成点-托盘调节及验证（修改点、层复制、排序、运行点）。

全局数值变量	局部数值变量	位置变量	全局平移变量	局部平移变量	托盘变量			
托盘号	0	序号	X	Y	Z	A	B	C
货物长(mm)	640.00	PO	1277.992	60.028	1518.231	-89.990	-0.005	-0.007
货物宽(mm)	110.00	001	0.000	0.000	0.000	0.000	0.000	0.000
层高(mm)	320.00	002	0.000	110.000	0.000	0.000	0.000	0.000
层数	3	003	0.000	220.000	0.000	0.000	0.000	0.000
间距(mm)	0.00	004	0.000	330.000	0.000	0.000	0.000	0.000
托盘尺寸mm	1000x1000x1000	005	375.000	265.000	0.000	90.000	0.000	0.000
托盘校准	层复制	006	485.000	265.000	0.000	90.000	0.000	0.000
奇偶同向	无标签	007	595.000	265.000	0.000	90.000	0.000	0.000
生成点	运行点	008	705.000	265.000	0.000	90.000	0.000	0.000
		009	-265.000	705.000	0.000	-90.000	0.000	0.000

托盘校准：采用类似用户坐标系设置的三点法构建坐标系，作为托盘的第一点。

奇偶同向/奇偶反向：点击按钮切换奇偶特性。奇偶同向指奇数层与偶数层货物方向相同；奇偶反向指奇数层与偶数层货物方向相反，货物以交替排布的方式堆叠。

无标签/标签朝外/标签同向：

一般的货物的标签贴在货物的 4 个侧面上，利用本功能可以人为设置标签的朝向。

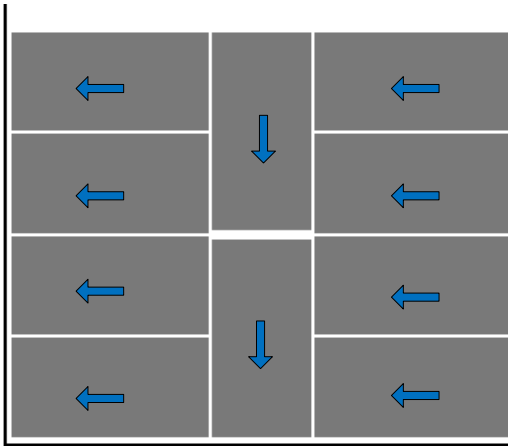
设后续所有货物的标签朝向与第一个朝向所成角度为 x 。

无标签：不考虑标签朝向，按最相差最少的角度排布，特点： $-90 < x \leq 90$ 。

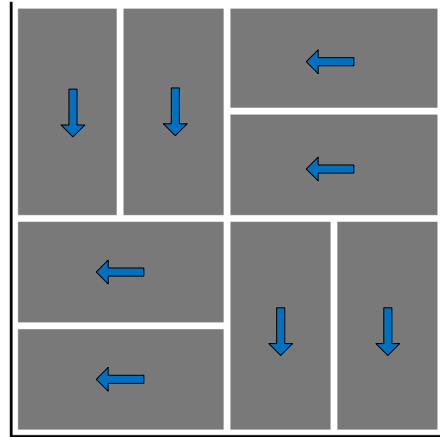
标签朝外：保证第一个标签（基准点）朝外，则其它所有标签朝外，特点： $-180 < x \leq 180$ 。

标签同向：横竖朝向都相同，特点： $0 \leq x < 180$ 。

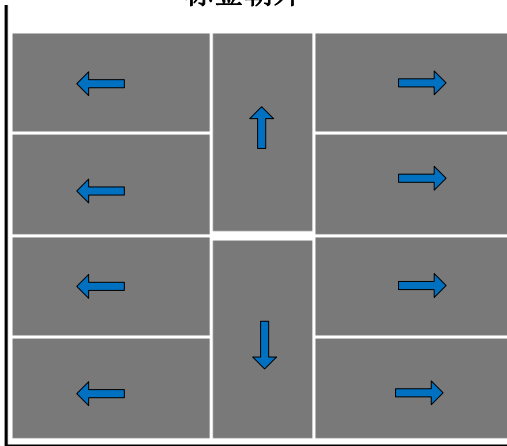
标签同向



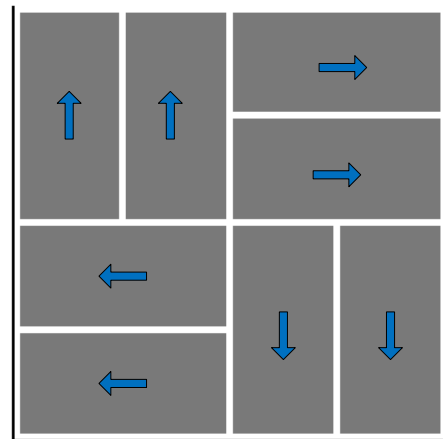
标签同向



标签朝外



标签朝外



生成点：根据垛型、货物长宽高、层数、间距、基准点，生成最终的托盘放置点数据，包含着该托盘上放置货物的每个位置点。

修改点：双击托盘变量数据列表行，修改值。

层复制：将一层数据复制到另一层。

运行点：运行到右侧列表中的选中行的位置，常用于检测。



排序：生成点之后，需要调整托盘中货物拜访的先后次序。在托盘变量页面右上角有个按钮，点击跳转到新页面，如下：



该页面中，托盘上货物的默认顺序会以数字在图形上显示。点击“排序”按钮，托盘上货物变得可点击。依次点击各个货物，便按点击顺序的先后确定了货物的顺序。

注意：

若需修改，再次点击“排序”按钮即可。

在一次排序完之前禁止再次点击“排序”按钮进行排序操作。

排序完成后注意保存。

4.3.3 码垛/拆垛编程

一个完整的码垛程序编制包含以下两个步骤：

1. ReSetPallet 指令初始化托盘
2. 制作循环，执行码垛点的放置和收回

注意：循环每执行完一次，下次再运行放垛指令，会自动移动到下一个托盘点。若需强制运行到其它托盘点，则应使用 SetPalletRunNo 指令设置。运行过程中因故障停止后，可利用 SetPalletRunNo 指令重新设置开始点，接着上次运行。

范例：

START;

ReSetPallet[1];

For B0=0,B0<8,Step[1]; ##托盘上有 8 个货物，循环执行 8 次

 MovToPut Pallet[1],P[1],50,0,50,V[80],PickV[30];

 MovFromPut Pallet[1],P[1],50,0,50,V[80],PickV[30];

EndFor;

```
IsPalletFinished(Pallet[1],LB1);
```

```
If LB1==1
```

```
Print "OK";
```

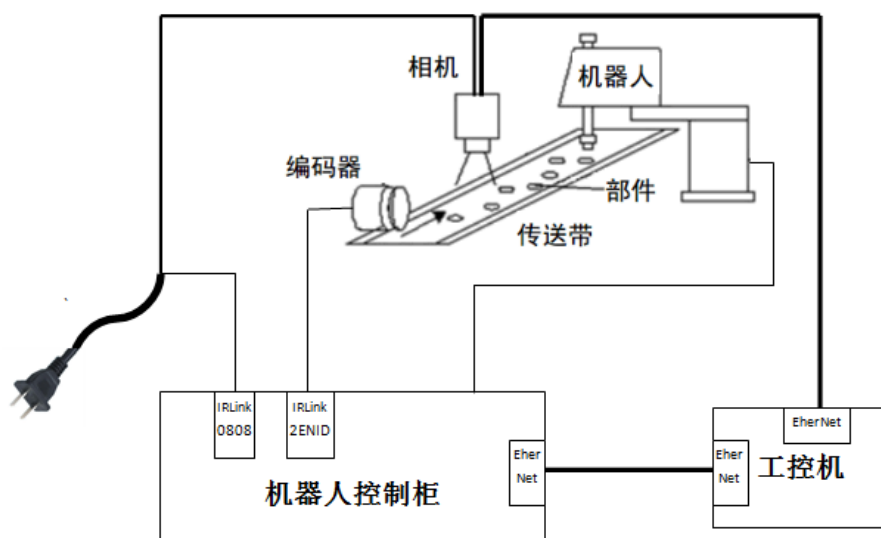
```
EndIf;
```

```
End;
```

拆垛与码垛类似。

4.4 跟随工艺

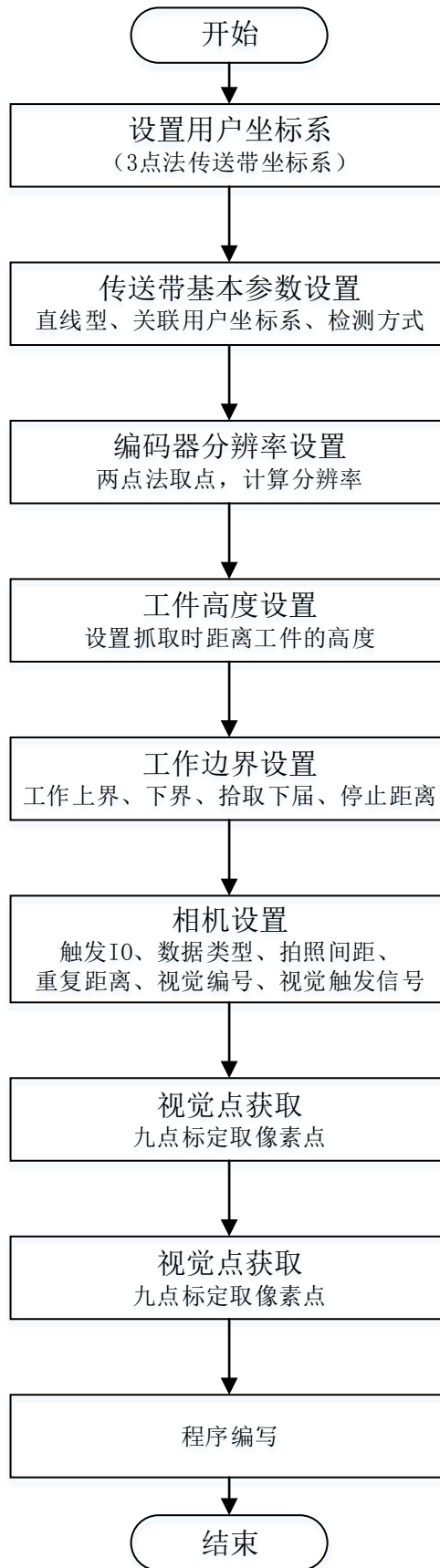
跟随工艺是针对在传送带上运动的工件，机器人根据其运动的情况进行跟踪并进行拾取等操作的一种工艺，整个过程中工件在传送带上保持运动状态不停止，机器人与其建立动态同步关系，即机器人“跟踪”工件的运动，如示意图所示。



完整的跟随工艺操作可分为三个步骤：

1. 坐标系设置。在传送带上建立用户坐标系，按照跟随运动的直线和圆盘传送带的两种形式，用户坐标系设置分别选用“三点法”和“旋转法”。
2. 传送带参数设置。其中包含基本参数、编码器、工件高度、检测参数等多个步骤，需要按照先后次序进行设置。
3. 跟随指令编程。

实际流程大概为



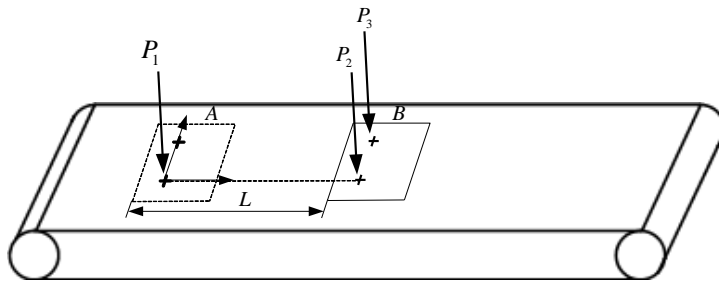
功能规格:

- (1) 支持直线和圆盘两种传送带类型;

- (2) 传送带数量：可配置4条，最多只能使能2条；
- (3) 检测方式：视觉或光电传感器；
- (4) 跟随运动指令：MoveL, MoveC, JumpL, 对于SCARA不支持MoveJ, Jump;
- (5) 工件种类：同一传送带可设置16种不同类型的工件；
- (6) 视觉数据类型：机器人坐标或像素坐标；
- (7) 视觉通信格式：TA, X1,Y1, theta1,T1, TA, X2,Y2, theta2,T2·····;
- (8) 一帧图像中对象的数量：最多10个；
- (9) 对象存储队列长度：500个。
- (10) 传送带速度限制：直线型 60m/min, 圆盘形 30r/min.

4.4.1 坐标系设置

坐标系设置是在【设置】-【坐标系设置】-【用户坐标系】页面操作的。
直线运动平台：



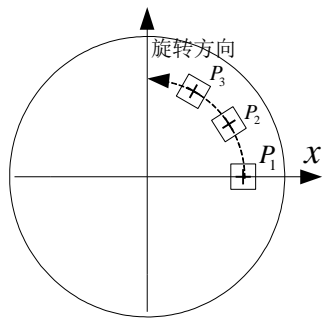
P1 为原点，保持标定板在运动过程中与传送带位置不变，使 P1 移动到 P2 位置，P1-P2 为 x 正方向。P3 为 xy 第一象限内的点。

取点具体方法：

三点法：（原理：三点确定一个平面）

工具坐标系		用户坐标系	
当前用户号1		用户1	直接输入法 三点法 旋转法
<input type="checkbox"/>	0	状态	
<input checked="" type="checkbox"/>	1	取点1	未定义
<input type="checkbox"/>	2	取点2	未定义
<input type="checkbox"/>	3	取点3	未定义
<input type="checkbox"/>	4	生成	
<input type="checkbox"/>	5	X	0.000 mm Y 0.000 mm Z 0.000 mm
<input type="checkbox"/>	6	A	0.000 ° B 0.000 ° C 0.000 °
<input type="checkbox"/>	7		
<input type="button" value="上一页"/> <input type="button" value="下一页"/>			

请注意取点时为了提高精度，建议三个点的位置尽量远，同时末端有工具时请勾选对应工具。
旋转运动平台：



旋转使传送带上同一位置分别移动到 P1、P2、P3，示教这三个点，得到以旋转中心为原点，以 P1 为 x 正方向的坐标系。

旋转法：（原理：三点确定一段固定的圆弧）

工具坐标系	用户坐标系																																				
<table border="1"> <tr><th colspan="3">当前用户号1</th></tr> <tr><td><input type="checkbox"/></td><td>0</td><td>X</td></tr> <tr><td><input checked="" type="checkbox"/></td><td>1</td><td>X</td></tr> <tr><td><input type="checkbox"/></td><td>2</td><td>X</td></tr> <tr><td><input type="checkbox"/></td><td>3</td><td>X</td></tr> <tr><td><input type="checkbox"/></td><td>4</td><td>X</td></tr> <tr><td><input type="checkbox"/></td><td>5</td><td>X</td></tr> <tr><td><input type="checkbox"/></td><td>6</td><td>X</td></tr> <tr><td><input type="checkbox"/></td><td>7</td><td>X</td></tr> </table>		当前用户号1			<input type="checkbox"/>	0	X	<input checked="" type="checkbox"/>	1	X	<input type="checkbox"/>	2	X	<input type="checkbox"/>	3	X	<input type="checkbox"/>	4	X	<input type="checkbox"/>	5	X	<input type="checkbox"/>	6	X	<input type="checkbox"/>	7	X									
当前用户号1																																					
<input type="checkbox"/>	0	X																																			
<input checked="" type="checkbox"/>	1	X																																			
<input type="checkbox"/>	2	X																																			
<input type="checkbox"/>	3	X																																			
<input type="checkbox"/>	4	X																																			
<input type="checkbox"/>	5	X																																			
<input type="checkbox"/>	6	X																																			
<input type="checkbox"/>	7	X																																			
<table border="1"> <tr> <td>用户1</td> <td>直接输入法</td> <td>三点法</td> <td>旋转法</td> </tr> <tr> <td colspan="4">状态</td> </tr> <tr> <td>取点1</td> <td colspan="2">未定义</td> <td></td> </tr> <tr> <td>取点2</td> <td>未定义</td> <td colspan="2">生成</td> </tr> <tr> <td>取点3</td> <td colspan="2">未定义</td> <td></td> </tr> <tr> <td>X</td> <td>0.000 mm</td> <td>Y</td> <td>0.000 mm</td> </tr> <tr> <td>Z</td> <td>0.000 mm</td> <td></td> <td></td> </tr> <tr> <td>A</td> <td>0.000 °</td> <td>B</td> <td>0.000 °</td> </tr> <tr> <td>C</td> <td>0.000 °</td> <td></td> <td></td> </tr> </table>		用户1	直接输入法	三点法	旋转法	状态				取点1	未定义			取点2	未定义	生成		取点3	未定义			X	0.000 mm	Y	0.000 mm	Z	0.000 mm			A	0.000 °	B	0.000 °	C	0.000 °		
用户1	直接输入法	三点法	旋转法																																		
状态																																					
取点1	未定义																																				
取点2	未定义	生成																																			
取点3	未定义																																				
X	0.000 mm	Y	0.000 mm																																		
Z	0.000 mm																																				
A	0.000 °	B	0.000 °																																		
C	0.000 °																																				
<p>上一页 下一页</p>																																					

请注意取点时为了提高精度，建议三个点的位置尽量远，同时末端有工具时请勾选对应工具。

4.4.2 传送带参数设置

该设置部分在【设置】-【功能扩展】-【跟随工艺设置】页面完成。

该界面为跟随工艺主界面，

视觉标定	码垛工艺设置	跟随工艺设置																											
<table border="1"> <tr> <th>传送带当前编号</th> <th>编辑</th> <th>使能</th> </tr> <tr><td>0</td><td><input checked="" type="checkbox"/></td><td><input type="checkbox"/></td></tr> <tr><td>1</td><td><input type="checkbox"/></td><td><input type="checkbox"/></td></tr> <tr><td>2</td><td><input type="checkbox"/></td><td><input type="checkbox"/></td></tr> <tr><td>3</td><td><input type="checkbox"/></td><td><input type="checkbox"/></td></tr> <tr><td>4</td><td><input type="checkbox"/></td><td><input type="checkbox"/></td></tr> <tr><td>5</td><td><input type="checkbox"/></td><td><input type="checkbox"/></td></tr> <tr><td>6</td><td><input type="checkbox"/></td><td><input type="checkbox"/></td></tr> <tr><td>7</td><td><input type="checkbox"/></td><td><input type="checkbox"/></td></tr> </table>			传送带当前编号	编辑	使能	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	1	<input type="checkbox"/>	<input type="checkbox"/>	2	<input type="checkbox"/>	<input type="checkbox"/>	3	<input type="checkbox"/>	<input type="checkbox"/>	4	<input type="checkbox"/>	<input type="checkbox"/>	5	<input type="checkbox"/>	<input type="checkbox"/>	6	<input type="checkbox"/>	<input type="checkbox"/>	7	<input type="checkbox"/>	<input type="checkbox"/>
传送带当前编号	编辑	使能																											
0	<input checked="" type="checkbox"/>	<input type="checkbox"/>																											
1	<input type="checkbox"/>	<input type="checkbox"/>																											
2	<input type="checkbox"/>	<input type="checkbox"/>																											
3	<input type="checkbox"/>	<input type="checkbox"/>																											
4	<input type="checkbox"/>	<input type="checkbox"/>																											
5	<input type="checkbox"/>	<input type="checkbox"/>																											
6	<input type="checkbox"/>	<input type="checkbox"/>																											
7	<input type="checkbox"/>	<input type="checkbox"/>																											
<table border="1"> <tr> <td colspan="3">传送带0: 基本参数</td> </tr> <tr> <td>传送带类型:</td> <td>直线型</td> <td>关联坐标系: 0</td> </tr> <tr> <td>编码器通道:</td> <td>0</td> <td>检测方式: 视觉</td> </tr> <tr> <td colspan="3">抓取位置补偿</td> </tr> <tr> <td>dX</td> <td>0.000 mm</td> <td>dY 0.000 mm</td> </tr> <tr> <td>dZ</td> <td>0.000 mm</td> <td>参数设置</td> </tr> <tr> <td>dA</td> <td>0.000 °</td> <td>dB 0.000 °</td> </tr> <tr> <td>dB</td> <td>0.000 °</td> <td>dC 0.000 °</td> </tr> </table>			传送带0: 基本参数			传送带类型:	直线型	关联坐标系: 0	编码器通道:	0	检测方式: 视觉	抓取位置补偿			dX	0.000 mm	dY 0.000 mm	dZ	0.000 mm	参数设置	dA	0.000 °	dB 0.000 °	dB	0.000 °	dC 0.000 °			
传送带0: 基本参数																													
传送带类型:	直线型	关联坐标系: 0																											
编码器通道:	0	检测方式: 视觉																											
抓取位置补偿																													
dX	0.000 mm	dY 0.000 mm																											
dZ	0.000 mm	参数设置																											
dA	0.000 °	dB 0.000 °																											
dB	0.000 °	dC 0.000 °																											
<p>上一页 下一页</p>																													

主界面包含了编辑选项、使能选项、基本参数、补偿参数、参数设置按钮等内容。

编辑：选定了当前编号之后，对该编号的传送带进行编辑。

使能：选中表示传送带处于运行状态。

在设置过程中需要说明的是，界面由顶部页面标题、右侧导航栏和中间的参数选型三部分组成。页面标题显示了传送带编号和当前界面内容。点击上/下一步时参数临时保存，断电消失；最后一个界面点【完成】参数被固化，中间界面点保存也可固化。

点击【参数设置】按钮就可以进入具体设置步骤了。

进入第一个界面“基本参数设置”：



如图所示，其中包含了：

1、传送带类型；2、编码器通道；3、用户坐标系；4、检测方式；

注意：由于用户坐标系 0 是系统固定的，且与基坐标系重合不能修改。关联的坐标系一般选用 1~15。

按照需求进行设置即可，完成后点击【下一步】。

进入第二个界面“编码器校准”：



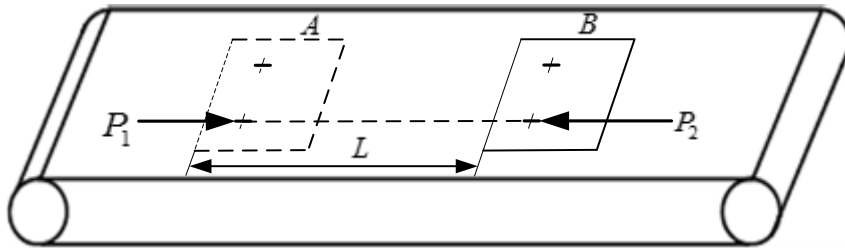
如图所示，其中包含了：

1、取两个点；2、编码器方向；3、编码器分辨；

其中分辨率设置按照如下步骤:

分辨率: 传送带每移动 1mm 或 1 弧度, 编码器转过的脉冲数。

设置方式: 标定或手动输入, 编码器方向也可以手动选择。

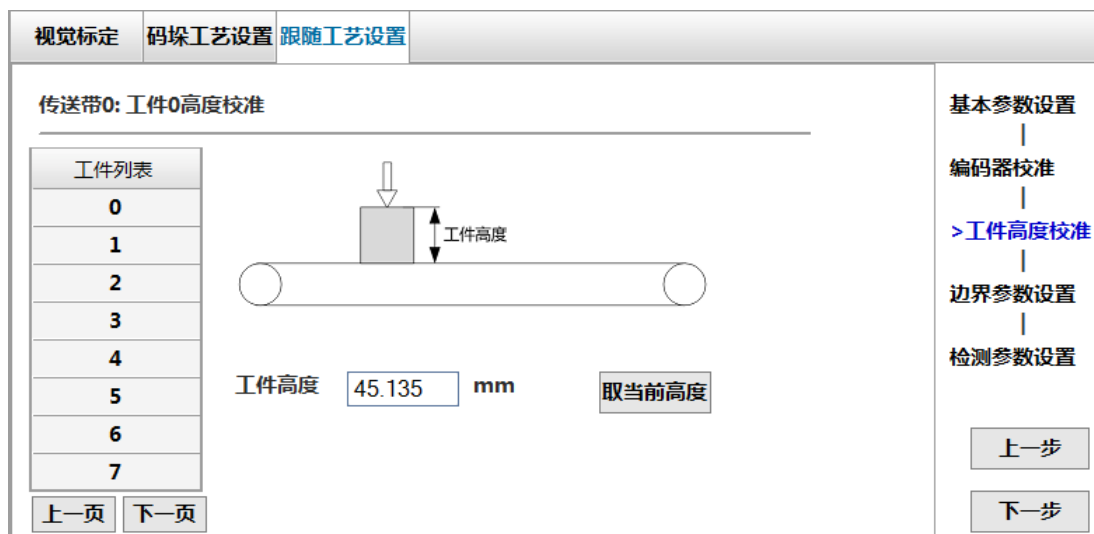


如图, 标定方法:

- 1) 在传送带上放置一个 mark 点 P1, 示教机器人对准 P1, 点击【取第一点】。
- 2) 保持 mark 点相对传送带位置不变, 移动传送带使其到达 P2 位置, 使机器人对准 P2 取第二点。
- 3) 点击【计算】。

按照需求进行设置即可, 完成后点击【下一步】。

进入第三个界面“工件高度校准”:

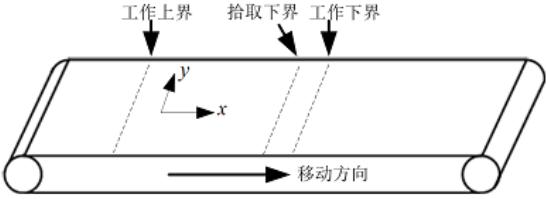
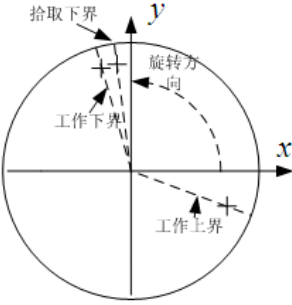


首先解释设置工件高度的意义: 其概念为工件抓取点到传送带坐标系零点的相对高度。即抓取位置在传送带坐标系下的 z 坐标值。设置器高度的功能是, 视觉或光电仅能检测到工件的 xy 位置, 通过该参数设定抓取工件时的高度值。按照以下方法:

标定方法:

- 1) 点击工件列表选择需要设置的工件。
 - 2) 将机器人工具末端移动到工件抓取位置, 点击【取当前高度】。
- 完成后点击【下一步】。

进入第四个界面“边界参数设置”:

视觉标定	码垛工艺设置	跟随工艺设置	
<p>传送带0: 边界参数设置--直线</p>  <p>工作上界 <input type="text" value="0.000"/> mm <input type="button" value="取当前点"/> 停止距离 <input type="text" value="0.087"/> mm</p> <p>工作下界 <input type="text" value="1.047"/> mm <input type="button" value="取当前点"/></p> <p>拾取下界 <input type="text" value="0.785"/> mm <input type="button" value="取当前点"/></p>			<p>基本参数设置</p> <p>编码器校准</p> <p>工件高度校准</p> <p>>边界参数设置</p> <p>检测参数设置</p> <p><input type="button" value="上一步"/></p> <p><input type="button" value="下一步"/></p>
<p>传送带0: 边界参数设置--圆盘</p>  <p>工作上界 <input type="text" value="0"/> ° <input type="button" value="取当前点"/></p> <p>工作下界 <input type="text" value="60"/> ° <input type="button" value="取当前点"/></p> <p>拾取下界 <input type="text" value="45"/> ° <input type="button" value="取当前点"/></p> <p>停止距离 <input type="text" value="5"/> °</p>			<p>基本参数设置</p> <p>编码器校准</p> <p>工件高度校准</p> <p>>边界参数设置</p> <p>检测参数设置</p> <p><input type="button" value="上一步"/></p> <p><input type="button" value="下一步"/></p>

由于在“基本参数设置”中“传送带类型”选择不同，直线型和圆盘型两种形式，所以在该部分设置中呈现的界面有所区别。以下解释边界设置的意义和设置方法：

工作边界：用上下界两条直线表示，保证两直线间的部分都在机器人工作范围内。直线型传送带边界线垂直于移动方向，用传送带坐标系 x 值表示；圆盘形用相对传送带坐标系 x 轴的偏角表示。

拾取下界：若工件靠近但未到达工作下界时开始执行抓取动作，机器人将在运动过程中超界。为了防止这种情况，需要设置拾取边界，当工件超出拾取下界时就不再抓取。拾取下界的设置原则为：保证从拾取下界到工作下界的时间内能完成抓取动作。

标定方法：使机器人移动到靠近机器人工作边界的位置，点击【取当前点】。

停止距离：机器人在工作边界时会出现急停，动作剧烈。通过设定停止距离实现平滑停止，距边界还剩该距离时开始平滑，一般设置为 5° 或 5mm 。

完成后点击【下一步】。

进入第五个界面“检测参数设置”，由于由于在“基本参数设置”中“检测方式”选择不同，有传感器和视觉两种形式，这两种形式检测方法相去甚远，所以分开介绍，首先介绍传感器设置：

视觉标定	码垛工艺设置	跟随工艺设置				
传送带[0]: 检测参数设置—传感器设置						
传感器 DI 口	15	信号类型	上升沿	工件类型编号	1	下发
传送带[0]: 检测参数设置—传感器位置校准						
示教位置读取	X	-9.435	mm			
Y	309.044	mm	A	59.790	°	
检测位置计算	X	-18.245	mm			
Y	-205.936	mm	A	-179.139	°	
			使工件经过传感器并停止在工作空间内			
			基本参数设置 编码器校准 工件高度校准 边界参数设置 >检测参数设置			
			上一步			
			完成			

如图所示，设置分为两个部分：1、传感器设置；2、传感器位置校准；按照如下方法：

1、传感器设置：

DI:光电传感器在 IRLINK 模块上的输入端口号；

信号类型：传感器被工件触发时输出的信号边沿。

工件类型编号：一条传送带最多支持 16 种工件，使用传感器时不能识别工件类型，只能指定一种。

检测位置：触发传感器的瞬间，工件在传送带坐标系下的位置。

2、传感器检测位置标定方法：

1) 将工件放到传感器上游，开启传送带使其以工作速度运动，工件以自然状态经过传感器，运动到工作空间时使传送带停止。控制器自动记录下传送带的偏移量。

2) 移动机器人到达工件上方，调整机器人位置和姿态，点击【示教位置读取】记录当前位姿。

3) 点击【检测位置计算】计算出触发传感器时工件的位姿，该位姿即工件坐标系参数。

注意：

1) 保证标定时工件经过传感器的速度与正常工作时相同。

2) 标定过程中勿使人手或其他物体触发传感器。

3) 如果机器人末端装有工具，读取示教位置时请选择对应的工具号。

设置完成后，点击【完成】结束所有设置。

然后介绍视觉检测方式：

视觉标定	码垛工艺设置	跟随工艺设置	
传送带0: 检测参数设置--相机基本参数			
相机触发DO	<input type="text" value="31"/>	相机数据类型	<input type="text" value="机器人坐标"/>
拍照间距	<input type="text" value="130"/> mm	重复检测判定距离	<input type="text" value="5"/> mm
相机触发信号	<input type="text" value="上升沿"/>	视觉坐标系编号	<input type="text" value="0"/>

基本参数设置

|

编码器校准

|

工件高度校准

|

边界参数设置

|

[>检测参数设置](#)

“相机基本参数”界面：

相机触发 DO:相机硬件触发拍照 IO 在 IRLINK 模块上输入端口号。

相机触发信号类型：相机被触发时，IO 模块输出信号的变化类型，上升沿或下降沿。

拍照间距：传送带每移动该长度或角度，触发一次拍照。拍照间距应略小于相机视野。一般设置为（相机视野<减去>工件大小），详细设置方法参考 4.4.3。

重复检测判定距离：前后两次拍照可能会识别到同一物体，需要内部判断是否重复，当两个物体距离小于该距离时认为是同一个物体，剔除一个，详细设置方法参考 4.4.3。

相机数据类型：像素或机器人坐标。

视觉坐标系编号：当视觉系统发送给控制器的数据为像素坐标时，需要在控制器中完成坐标变换，该参数指定视觉坐标系参数的编号。传送带视觉标定结果保存在该视觉坐标系下。

设置后点击【下一步】。进入“视觉标定—视觉点获取”界面：

视觉标定	码垛工艺设置	跟随工艺设置			
传送带[0]: 检测参数设置—视觉标定--视觉点获取					
<input type="button" value="相机触发"/>	传送带拍照位置	<input type="text" value="0"/> <input type="button" value="取当前位置"/>			
点号	X	Y	点号	X	Y
1	<input type="text" value="0.000"/>	<input type="text" value="0.000"/>	6	<input type="text" value="0.000"/>	<input type="text" value="0.000"/>
2	<input type="text" value="0.000"/>	<input type="text" value="0.000"/>	7	<input type="text" value="0.000"/>	<input type="text" value="0.000"/>
3	<input type="text" value="0.000"/>	<input type="text" value="0.000"/>	8	<input type="text" value="0.000"/>	<input type="text" value="0.000"/>
4	<input type="text" value="0.000"/>	<input type="text" value="0.000"/>	9	<input type="text" value="0.000"/>	<input type="text" value="0.000"/>
5	<input type="text" value="0.000"/>	<input type="text" value="0.000"/>			

基本参数设置

|

编码器校准

|

工件高度校准

|

边界参数设置

|

[>检测参数设置](#)

若视觉数据为像素坐标，需要在控制器中完成标定，采用 9 点法。视觉标定分两步：该界面输入 9 个点的像素坐标，读取拍照时传送带的位置。

完成后点击【下一步】，进入“视觉标定—示教点获取”界面：

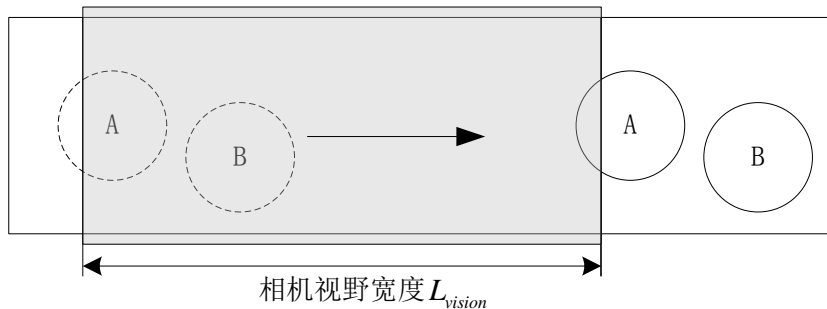
视觉标定	码垛工艺设置	跟随工艺设置				
传送带[0]: 检测参数设置—视觉标定--示教点获取						
传送带示教位置 <input type="text" value="0"/> <input type="button" value="取当前位置"/>		基本参数设置 编码器校准 工件高度校准 边界参数设置 >检测参数设置 <input type="button" value="上一步"/> <input type="button" value="完成"/>				
机器人示教点 <input type="text" value="1"/> <input type="button" value="取当前点"/> <input type="button" value="偏移"/> 未偏移						
点号	X		Y	点号	X	Y
1	<input type="text" value="0.000"/>		<input type="text" value="0.000"/>	6	<input type="text" value="0.000"/>	<input type="text" value="0.000"/>
2	<input type="text" value="0.000"/>		<input type="text" value="0.000"/>	7	<input type="text" value="0.000"/>	<input type="text" value="0.000"/>
3	<input type="text" value="0.000"/>		<input type="text" value="0.000"/>	8	<input type="text" value="0.000"/>	<input type="text" value="0.000"/>
4	<input type="text" value="0.000"/>		<input type="text" value="0.000"/>	9	<input type="text" value="0.000"/>	<input type="text" value="0.000"/>
5	<input type="text" value="0.000"/>		<input type="text" value="0.000"/>	<input type="button" value="标定/下发"/>		

将标定板移动到机器人工作空间，依次示教 9 个点，读取传送带示教位置，点击“偏移”计算出 9 个点在拍照位置时的坐标。点击【标定/下发】将标定结果保存到视觉坐标系中。完成后点击【完成】结束所有设置。

4.4.3 关键参数说明

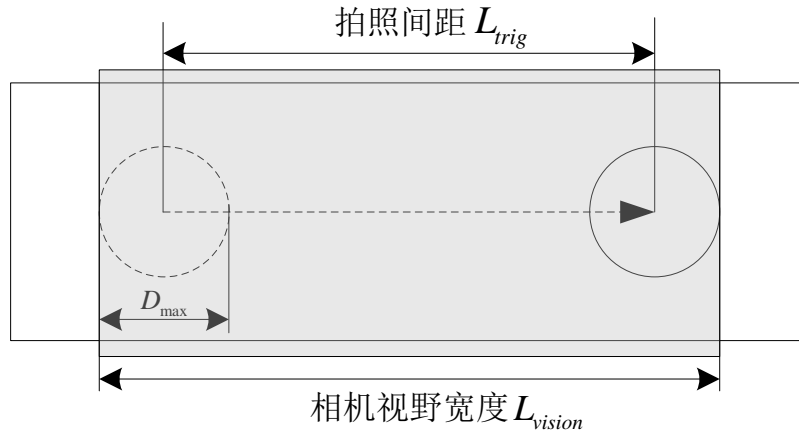
1、拍照间距

如图所示，阴影部分为相机视野，物体经过相机下方时相机被触发拍照，用户需要设定拍照间距，即传送带每移动多大距离触发一次拍照。直观的理解是拍照间距等于视野宽度，此时相机能拍到传送带上每一部分。这样设置时存在问题，如图 2 所示，第一次拍照时工件位于虚线位置，第二次拍照时工件运动到了右侧的实线位置，前后两次拍照都不能识别到工件 A。



为了避免这种情况，需要设置合适的拍照间距，原则是保证每个工件至少有一次被完整拍到。考虑极限情况如图 2 所示，第一次拍照时工件位于图中左侧虚线位置，即将全部进入相机视野，第二次拍照时工件位于右侧实线位置，即将离开相机视野，这样前后两次都不能识别到该工件，而只要拍照间距略小于这个距离就能保证至少有一次完整拍到。因此合适的拍照间距为：

$$L_{trig} < L_{vision} - D_{max}$$



上式给出了拍照间距的上界，需要说明的是拍照间距并不是越小越好。拍照间距减小，则拍照时间间隔减小，要求相机的处理时间更短，当拍照触发时间间隔小于视觉处理时间时将不能正常工作，它们之间的关系为：

$$T_{trig} = \frac{L_{trig}}{v_{conveyor}} > T_{vision_process}$$

综上所述，拍照间距不能太大也不能太小，可按下列公式给定，其中 δ_{vision} 为视觉综合误差。

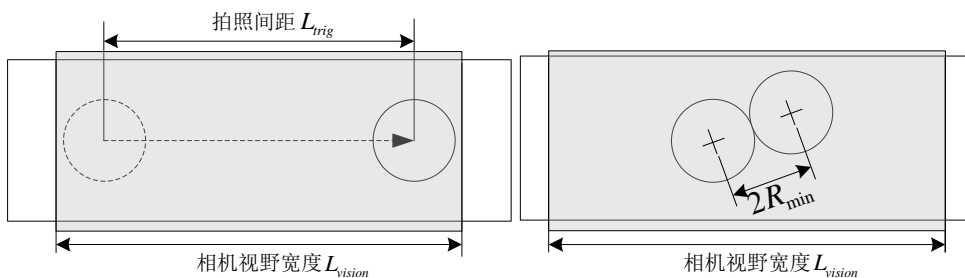
$$\begin{cases} L_{trig} = L_{vision} - D_{max} - \delta_{vision} \\ \delta_{vision} < 10mm \end{cases}$$

按上述公式得到拍照间距后，如果发现运行中报“视觉处理时间过长”，说明不符合条件*。此时不应修改拍照间距，可以降低传送带速度或者优化视觉缩短处理时间。

2、重复检测判定距离

拍照间距的设置原则是保证每个物体至少被完整拍到一次，这样有可能同一个物体被拍到两次，如下图所示，左侧虚线为第一次拍照时工件的位置，右侧实线为第二次拍照时工件的位置，前后两次都能识别到该工件，两次的结果都传输给机器人去抓取将出现一次空抓。为了避免这种情况需要设定重复剔除距离，当两个工件的距离小于重复检测判定距离时，将剔除其中一个。重复剔除距离的设置原则是不能误剔除两个正常靠近的物体。因此可按下列公式给定：

$$L_{remove} < 2R_{min} = D_{min}$$



4.4.4 跟随指令编程

以下给出一个完整的跟随工艺的范例，本例采用视觉检测的方式进行跟随工艺的操作。

程序代码

```
START;
L[0]:
##打开端口。外部设备作服务器地址 10.44.53.13，端口号 1025;
##本地控制器作客户端，端口号 1026。
Open Socket("10.44.53.13",1025,1026,B0);
If B0 == 0
Goto L[0];
CnvVision (Conveyor[1],ON,1026);
P[30]=(0,0,10,0,0,0),(0,0,0,0),(7,0,0); ##定义 P[30]为在 1 号传送带上物体的正上方 10mm 处
L[1]:
Movj P[0],V[30],Z[0];
GetCnvObject(1,0), Goto L[1]; ##接收 1 号传送带，0 号类型的物体的数据
RefSys Conveyor(1,Tool[2]); ##使用工具 2 末端完成与传送带 1 的速度同步运动
Movl P[30],V[100],Z[1],Tool[2]; ##P[30]是在传送带 1 坐标系下的点
Set Out[1],ON; ##打开开关，吸附物体
Delay T[1];
RefSys Base; ##切换到机器人坐标系
Jump P[1],V[100],Z[0],LH[10],MH[-750],RH[10]; ##将物体移动至 P[1]处
Set Out[1],OFF,T[0]; ##放置物体
Delay T[1];
Goto L[1];
CnvVision (Conveyor[1],OFF,1026);
Close Socket,111;
END;
```

以上程序中的指令在“2.控制指令”中都有介绍，下面针对其中有关视觉检测的几个指令进行详细的使用介绍：

1.指令 CnvVison ON 是传送带视觉开启/关闭指令：

用来开启传送带视觉，开启后控制器将自动接收视觉对象，将接受到的数据保存在视觉队列中并最终其在传送带上的位置。该指令应放在程序初始化部分。

视觉对象数据包括 $x,y,\theta,type$ 四个参数， θ 的单位是度；一次拍照最多可处理 10 个对象，按如下格式与控制器通信：

TA, X1,Y1, θ 1,T1, TA, X2,Y2, θ 2,T2……;

每个有效对象都以 TA 开头，遇到非 TA 标识符，后面的数据不再解析。

相机未拍摄到物体，则返回 NG。

2.指令 GetCnvObject/CopyCnvObject 是传送带对象查询指令：

参数：传送带编号、类型编号、标签号

功能：查询对象是否进入工作空间，若有对象在工作空间内执行下一条，若没有则跳转到 L 处。若使用 Copy 查询后该对象在队列中仍保留，下次查询仍能查到，可用于码垛等多次跟

随同一个对象的情况，若使用 `get` 查询后该对象在队列中删除。

```
GetCnvObject( 0 , 1 ), Goto L[2];
```

传送带编号:	<input type="text" value="0"/>	标签号:	<input type="text" value="2"/>
类型编号:	<input type="text" value="1"/>		

3.指令 `RefSys` 是切换参考系:

参数: `Base`(基坐标系)、`Conveyor`(传送带)、`WorkBench`(待扩展)

功能: 机器人运动分静态和动态, 默认为静态运动, 当使用动态运动时需要指定参考坐标系。通过该指令切换参考系。

执行完 `RefSys` 后, 机器人进入与动平台同步模式。

```
RefSys Conveyor[0];
```

	<input type="button" value="Base"/>	<input checked="" type="button" value="Conveyor"/>	<input type="button" value="WorkBench"/>
序号	<input type="text" value="0"/>		

注意事项:

1、执行完 `RefSys Conveyor` 后机器人将一直与传送带保持同步运动, 直到执行 `RefSys Base`, 若不能及时执行 `RefSys Base` 机器人将会跟随传送带运动至出界。因此在逻辑上一定要保证 `RefSys Conveyor` 与 `RefSys Base` 成对存在, 如果使用 `Goto`, 子程序等, 一定要及时返回, 保证 `RefSys Base` 逻辑上能执行到, 且两条指令之间程序段的时间不能大于传送带移动到边界的时间。

2、切换坐标系指令应成对使用, 一次动态跟随结束时应先切换到基坐标系, 不允许在不同动态参考系之间直接切换。

3、同步模式下的运动为直角插补, 禁止使用关节运动。进入跟随传送带同步模式后, 机器人一直处于运动状态, 需要机器人静止的指令将无效, 具体如下:

无效: (使用无效果, 但不会造成其他影响)

`SetToolParm`、`SetUserParm`、`OffSetUserParm`、`Cnvrt`、运动指令中的 `User[***]`。

无法执行: (运行时会在该句指令处, 这些指令需要机器人处于静止状态)

`Wait IN`、`Print TimeStart`、`TimeOut`

禁止使用: (若使用则导致运行错误)

`Home`、`Until`、`Movj`、`Jump`(但 `JumpL` 支持)、`GetCurPoint`、`PE`。

4.5 视觉标定

视觉标定是机器人系统视觉功能使用的前提，标定就是为了获得相机与机器人的相对位置关系和标定的误差，同时将像素坐标转到机器人坐标。

影响相机标定精度的因素主要有调点精度、工装治具精度、视觉检测精度、本体绝对精度、标定算法精度。以下分别介绍“手动-固定仰视”和“手动-随动二轴”。

以下主要介绍一个完整的“手动-固定仰视”过程，然后根据设置的不同补充介绍“手动-随动二轴”的标定方式。

视觉标定操作后，可分为 2 个步骤：

1. 视觉标定参数设置
2. 指令编程及标定结果验证

4.5.1 视觉标定参数设置

该设置部分在【设置】-【功能扩展】-【视觉标定】页面完成。

该界面为视觉标定主界面，

视觉标定		码垛工艺设置	跟随工艺设置
视觉坐标系编号	编辑	视觉坐标系 [5]: 未定义	
0	<input type="checkbox"/>	相机名: 未定义	
1	<input type="checkbox"/>	相机安装方式: 未定义	
2	<input type="checkbox"/>	X方向平均误差(mm): <input type="text" value="0"/>	Y方向平均误差(mm): <input type="text" value="0"/>
3	<input type="checkbox"/>	X方向最大误差(mm): <input type="text" value="0"/>	Y方向最大误差(mm): <input type="text" value="0"/>
4	<input type="checkbox"/>	X方向单位像素尺寸(mm): <input type="text" value="0"/>	Y方向单位像素尺寸(mm): <input type="text" value="0"/>
5	<input checked="" type="checkbox"/>	标定工具X方向偏移(mm): <input type="text" value="0"/>	标定工具Y方向偏移(mm): <input type="text" value="0"/>
6	<input type="checkbox"/>		
7	<input type="checkbox"/>		
<input type="button" value="上一页"/> <input type="button" value="下一页"/>		<input type="button" value="相机基本参数"/>	<input type="button" value="视觉标定"/>

主界面包含了视觉坐标系编号编辑选项和部分标定结果与误差。

在进行其它操作前，需要先选择要编辑的视觉坐标系。（相机视觉坐标系的编号范围 0-15）再点击【视觉标定】进入标定步骤：

首先进入第一个页面“相机基本参数”：

视觉标定	码垛工艺设置	跟随工艺设置
视觉坐标系[0]: 相机基本参数 → 页面标题		
相机 IP	192 . 168 . 0 . 1	
相机名:	<input type="text"/>	端口号: 0 → 参数编辑
相机触发方式	<input type="radio"/> I/O触发 <input checked="" type="radio"/> 以太网触发	
发送的字符串	<input type="text"/>	
接收数据格式	帧头 <input type="text"/>	分隔符 <input type="text"/> 结束符 <input type="text"/>
相机通讯测试		
数据接收区: <input type="text"/>		
← 导航栏		

> 相机基本参数

| 相机安装方式

| 标定方式

| 基准点示教

| 九点示教

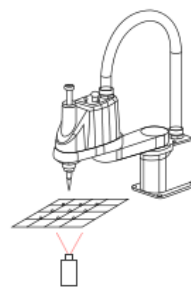
| 参数生成

[上一步]

[下一步]

相机名命名规范：字母开头，由字母、数字以及下划线组成，最多 10 个字符。

当前为全手动无通讯不需要设置相机基本参数。
按照需求进行设置即可，完成后点击【下一步】。
进入第二个界面“相机安装方式”：

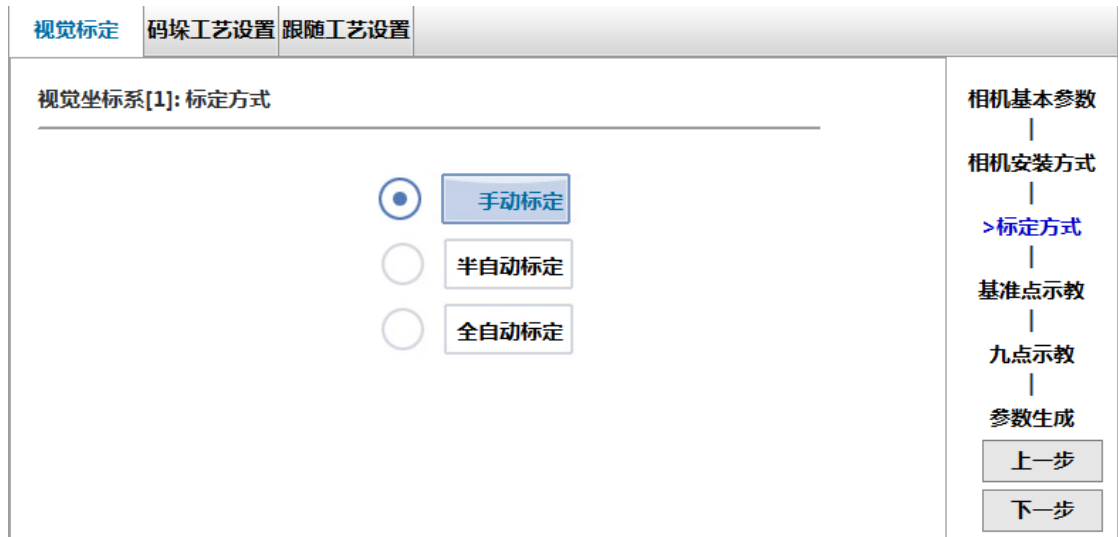
视觉标定	码垛工艺设置	跟随工艺设置
视觉坐标系[5]: 相机安装方式 固定仰视式		
<div style="display: flex; justify-content: space-between;"> <div style="width: 30%;"> <p>0-固定俯视式</p> <p>1-固定仰视式</p> <p>2-可移动式J2</p> <p>3-可移动式J4</p> <p>4-可移动式J5</p> <p>5-可移动式J6</p> </div> <div style="width: 40%; text-align: center;">  </div> </div>		<p>相机基本参数</p> <p>> 相机安装方式</p> <p> 标定方式</p> <p> 基准点示教</p> <p> 九点示教</p> <p> 参数生成</p> <p>[上一步]</p> <p>[下一步]</p>

相机安装方式	说明
可移动式J2	相机安装在SCARA机器人或直角坐标型机器人的接口2上。
可移动式J4	相机安装在SCARA机器人或直角坐标型机器人的接口4上。
可移动式J5	相机安装在6-轴机器人的接口5上。
可移动式J6	相机安装在6-轴机器人的接口6上。
固定俯视式	相机和目标对象不移动，相机俯视机器人的工作范围。相机获取机器人坐标系中的位置信息 相机安装后必须与指定坐标系的XY平面垂直。（角度差可能导致准确度不够） 指定的坐标系为机器人坐标系和本地坐标系。
固定仰视式	相机不移动并仰视机器人工作范围（的一部分）。例如，该安装方式

用于检查机器人所运载的对象的位置。

按照需求进行设置即可，完成后点击【下一步】。

进入第三个界面“标定方式”：



选择“手动标定”

完成后点击【下一步】。

进入第四个界面“基准点示教”：



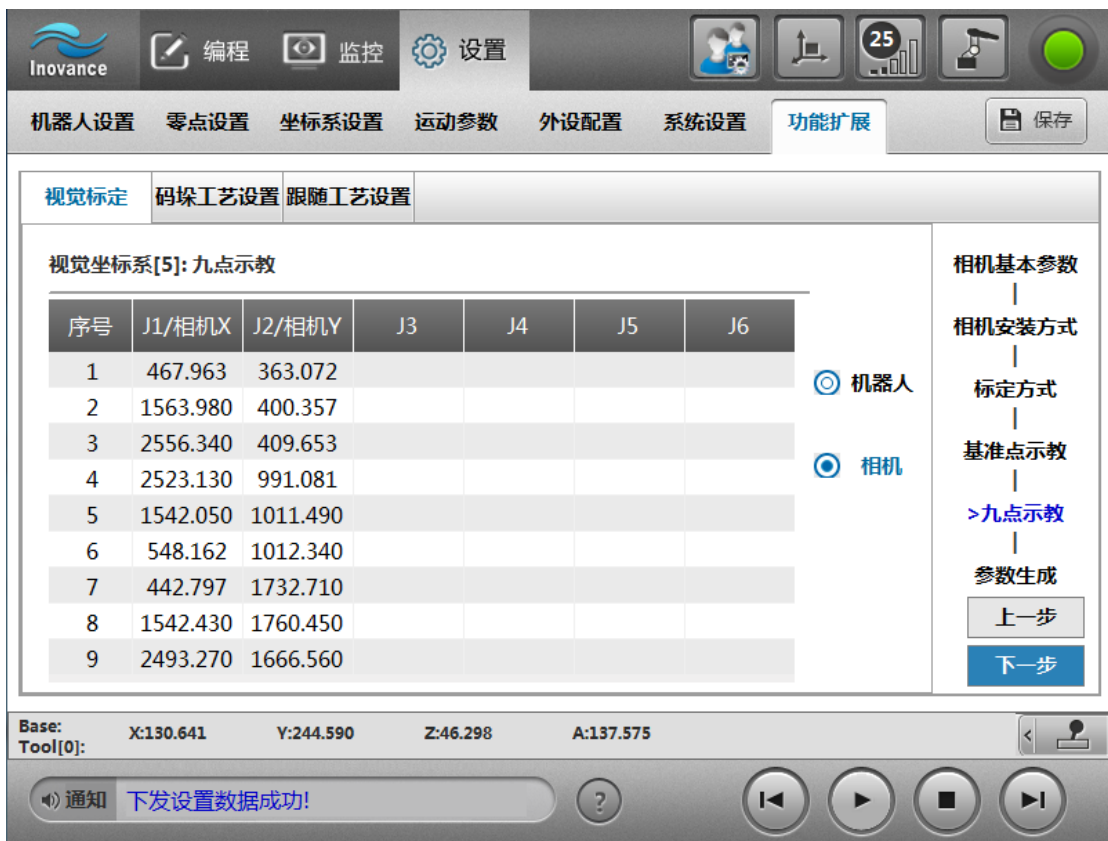
调整机器人 XY 平面运动对准视觉视野中心，选取基准点一，然后绕 Z 轴旋转一段距离后，再次调整 XY 平面运动对准视觉视野中心选取基准点二。

完成后点击【下一步】。

进入第五个界面“九点示教”：



按照图中的顺序在视觉视野中均匀取点，每移动一个点就通过视觉软件读出相机坐标,手动输入到相机坐标 XY 中，若机器人与视觉系统建立通讯连接，则勾选“获取相机坐标”就可以获取需要的坐标值，然后点击“取当前值”下发到示教器中。完成后点击【下一步】。



“机器人”、“相机”页面分别显示了刚才读入的九个示教点。双击某个点可以修改该点数值。完成后点击【下一步】。



记录“标定工具 X 方向偏移”，“标定工具 Y 方向偏移”这两个值；
 点击【完成】；



进入【设置】-【坐标系设置】-【工具坐标系】-【直接输入法】

将最后记录的标定工具 XY 方向上的两个偏差分别输入到工具坐标系的 X 和 Y，建立工具坐标系。

4.5.2 指令编程及标定结果验证

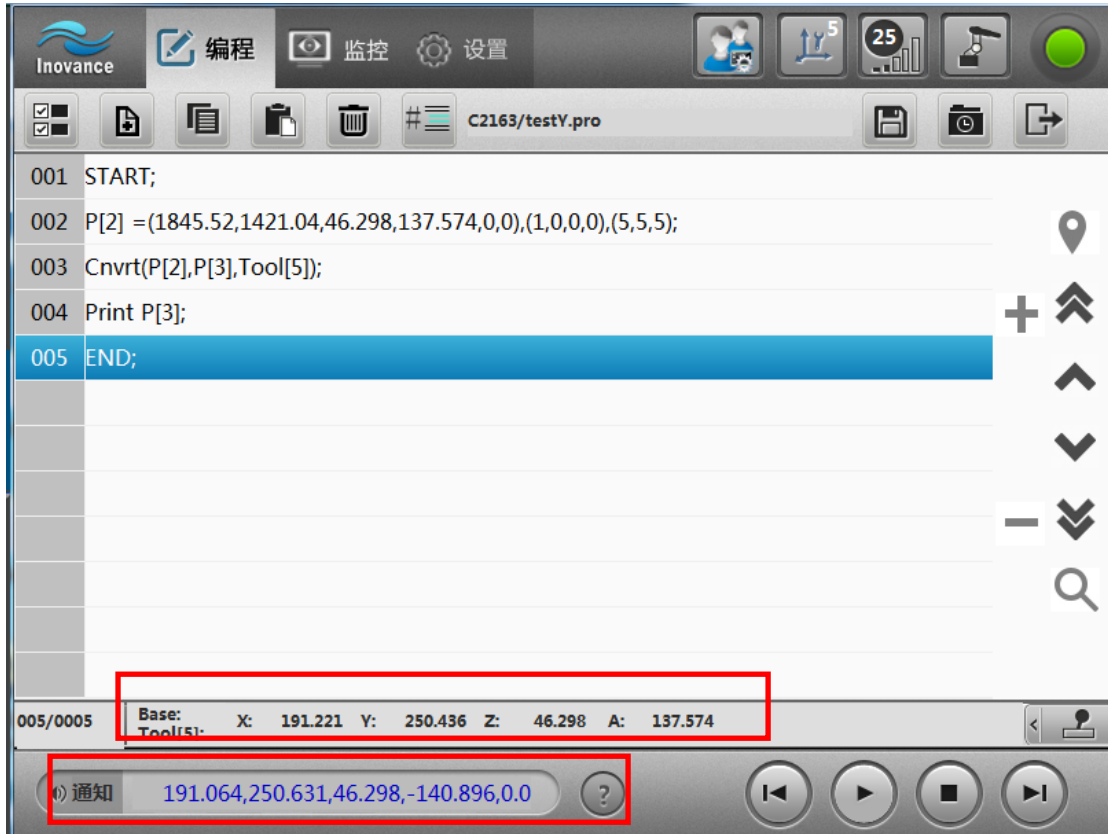
以下给出一个完整的视觉标定编程的范例。



取 P[2] 点注意设置最后一项坐标系号，工具号，用户号。

Cnvr 指令说明：将 P[2] 点转换到工具坐标系下的 P[3] 点。

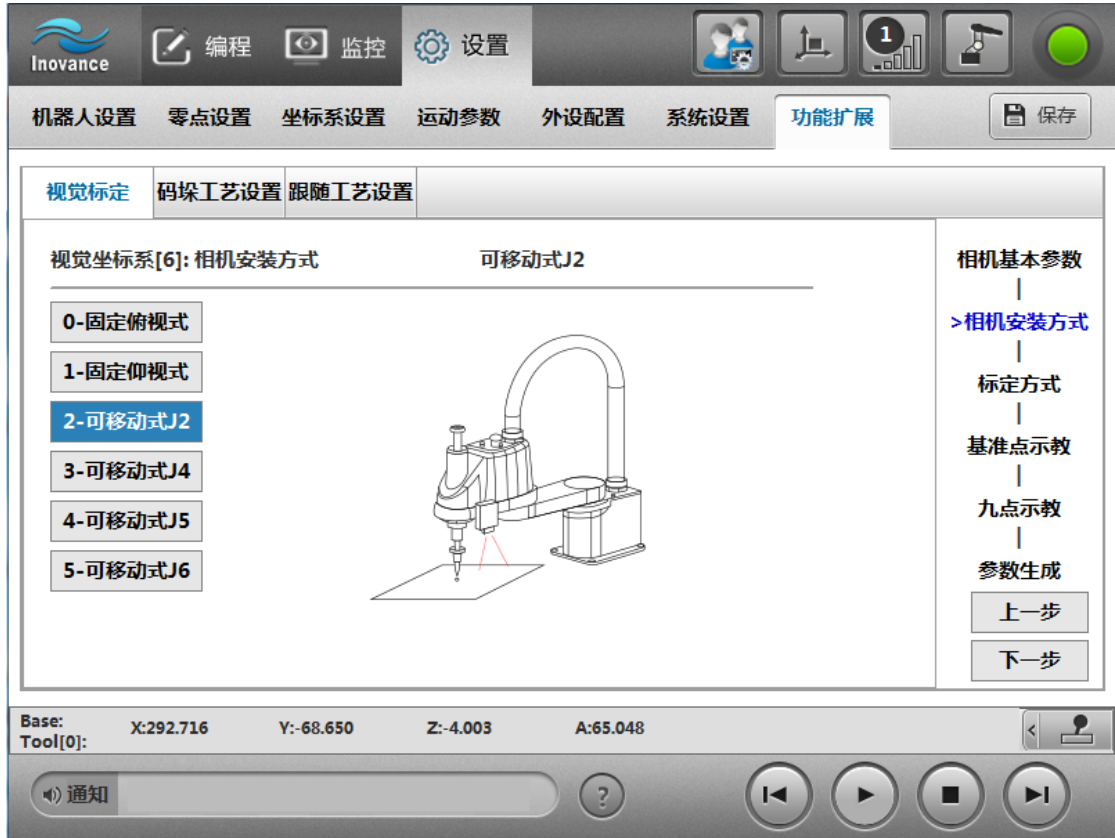
“固定标定”的方式不需要勾选“视觉基准点”。



最后的结果显示在通知栏中，与标定点进行比较确定标定的精确度。

4.5.3 “手动-随动二轴” 补充介绍

为了明确数值的来龙去脉且以上步骤与“手动-固定仰视”操作过程一样，所以仅保留图片以说明数值的正确性。



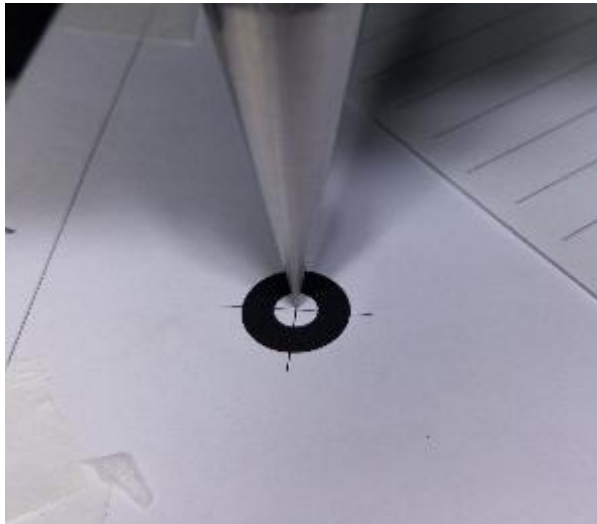
完成后点击【下一步】。



完成后点击【下一步】。



使用尖端通过不同的姿态对同一个点，用于建立工具坐标系，并计算在工具坐标系下的点作为基准点。



完成后点击【下一步】。

视觉标定 码垛工艺设置 跟随工艺设置

视觉坐标系[6]: 九点示教

取当前点

获取相机坐标:(单位:像素)

相机X 258.875

相机Y 238.114

机器人关节坐标:(单位:°)

-70.260 87.074 -0.025

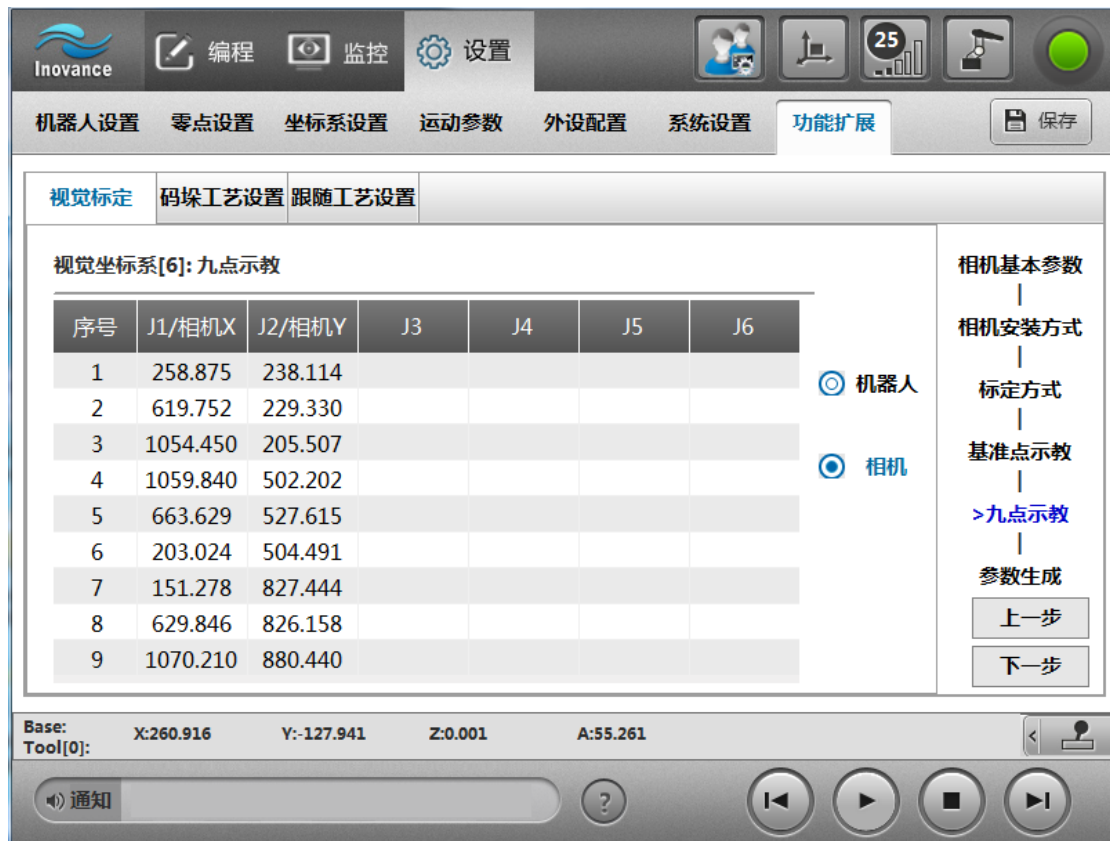
38.450 0.000 0.000

相机基本参数
|
相机安装方式
|
标定方式
|
基准点示教
|
>九点示教
|
参数生成
|
上一步
|
下一步

Base: X:260.916 Y:-127.941 Z:0.001 A:55.261
Tool[0]:

通知 ? |< |> |■|>

完成后点击【下一步】。



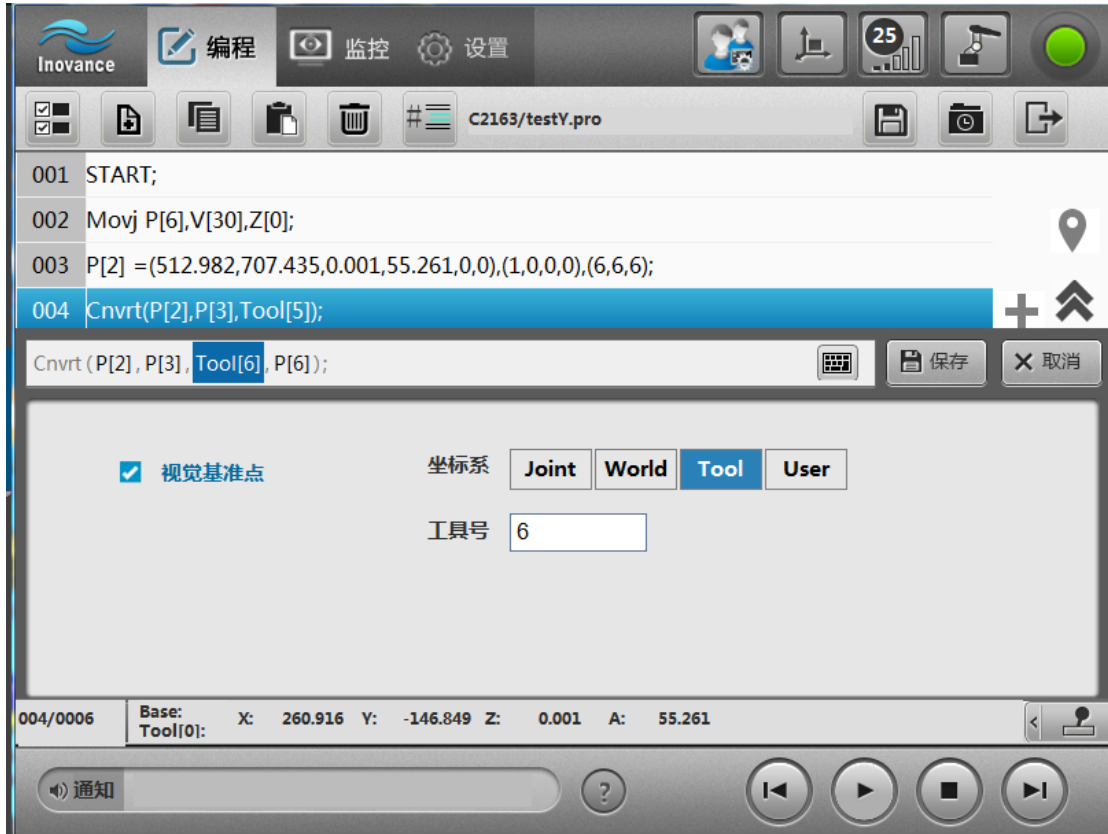
完成后点击【下一步】。



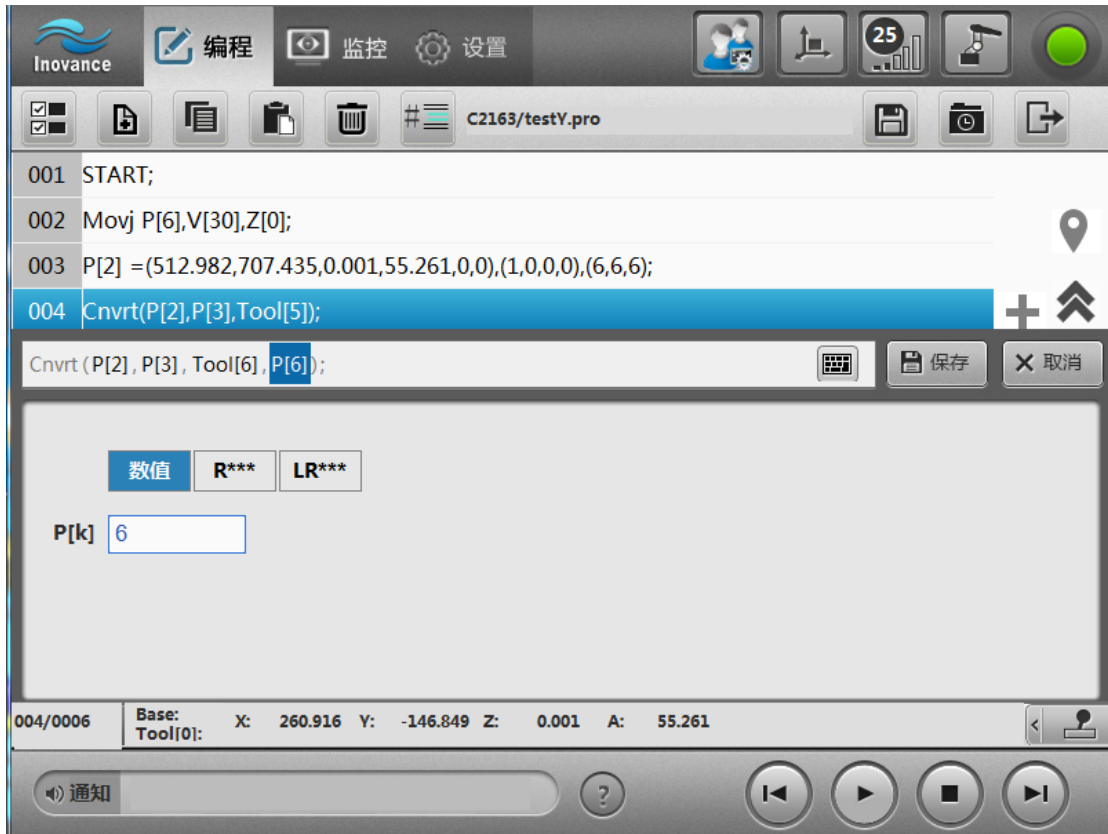
点击【完成】。



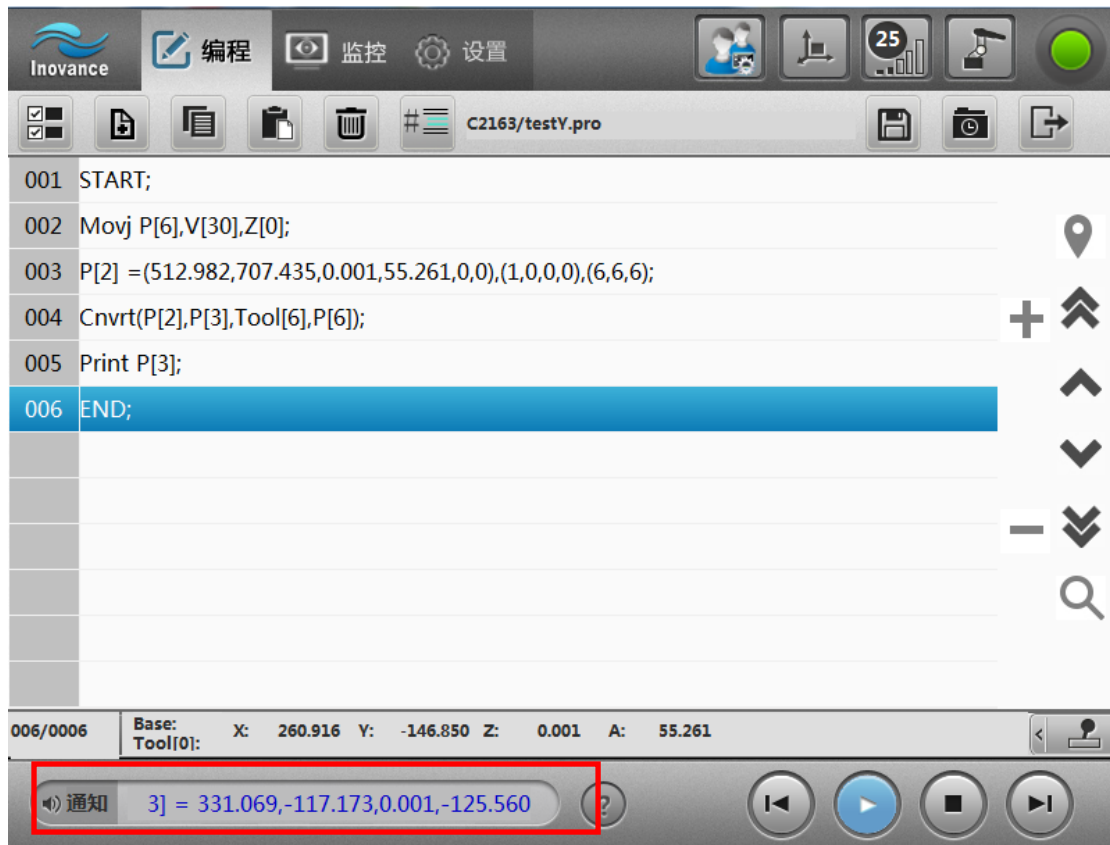
指令编程部分:



此处需要勾选“视觉基准点”



Cnvrt 指令说明：将 P[2] 点转换为工具坐标系下 P[3] 点，用户号不变，其中拍照点为 P[6]。



P[3]为打印点，完成变量配置和指令配置之后，运行：



P[7]是直角坐标系转换到工具坐标系下的点，将通知栏中的显示的坐标和 P[7]点的坐标做

误差分析。

4.6 锁螺丝工艺

锁螺丝机是基于 Scara 机器人的一款机器人，用锁螺丝电批轴取代原有的 J4 轴。在使用时，提前预设好锁螺丝/拆螺丝的工艺，锁螺丝机将自动完成螺丝的搜索、拧紧或拧松工作。完整的流程可分为如下三步：

- 1、配置锁或拆工艺：利用示教器配置锁螺丝工程文件*。
- 2、拧紧/拧松编程：对锁而言，围绕使用一条指令 **LockScrew**，即可自动按照配置的工艺执行整个锁付过程；拆同理，也只围绕使用一条指令 **UnLockScrew**。
- 3、运行。

锁螺丝工程文件*：锁螺丝工程文件以“.stp”为后缀名，每个工程包含最多 16 组锁工艺和 16 组拆工艺。每组锁或拆工艺各自具有自己的一套工艺参数，以满足各自的锁或拆的动作需求。

4.6.1 螺丝锁付

(1) 配置拧紧工艺

进入示教器【设置】-【功能扩展】-【锁螺丝工艺设置】页面配置工艺。页面最左侧为锁螺丝工程文件列表，显示当前存在的工程文件。配有相应新增、重命名、删除、复制、粘贴、列表翻页功能。进行如下操作：



(a) 选择工程。若初次使用，列表为空，点击“新建”按钮，新建一个的锁螺丝工程，如新建的 phone.stp。

(b) 选择拧紧或拧松，对于螺丝锁付，这里选择“拧紧”。

(c) 配置工程中的工艺。右上角工具栏配有新增、上传、删除、复制、粘贴、下载功能，以对每套工艺进行操作。

- 新增：在当前工程中新增一套工艺，所有工艺参数均为 0。

- 删除：删除当前工程中的工艺。
- 复制、粘贴：将一套工艺中的所有参数复制到另一套工艺中。
- 保存：通用的“保存”按钮，保存当前工程（包括所有工艺）。
- 上传：上传当前伺服中的锁或拆工艺。
- 下载：将当前工程中的这套工艺下载到伺服中。

*工艺参数解析

在编程时，一个锁螺丝指令可以完成螺丝拧紧的全部操作。其过程分为以下几个阶段：搜索-高速拧紧-低速拧紧-保持-结束，工艺参数就是负责控制这些阶段执行的节奏。



锁付时拧紧动作分解：

- 1) 搜索阶段：轴 3 带动电批吸嘴向下运动，同时电批以“搜索速度”开始空转，这样可以方便批头和螺帽对准。当电批刚好转动“搜索圈数”左右时，螺丝接触到螺孔表面，电批则自动切换到高速拧紧阶段；
- 2) 高速拧紧：此阶段会以较高的速度进行快速拧紧，当扭矩到达设定的“高速转矩”表明高速拧紧已完成，然后切换到低速拧紧阶段
- 3) 低速拧紧：同样，电批以低速转动，扭矩不断增大，当到达“目标扭矩”时，表明低速拧紧已完成
- 4) 随后保持拧紧一段时间，至此结束，即完成一个螺丝的锁付。

参数解释：

搜索速度：低速搜索阶段寻找孔位的速度，用于批头对孔。

搜索圈数：当扭矩到达“搜索圈数”时，表明已找到孔位，然后进入高速拧紧阶段。

高速转速：高速拧紧阶段的电机转速。

高速转矩：当扭矩到达“高速转矩”时，表明高速拧紧已完成，然后进入低速拧紧阶段。

目标转速：低速拧紧阶段的电机转速。

目标扭矩：当扭矩到达“目标扭矩”时，表明低速拧紧已完成，随后保持一段时间(见下“目标转矩保持时间”)，锁付完成。

软启动时间：(15 及以上版本可忽略)

目标转矩保持时间：到达目标转矩后将保持一段时间，用以稳固拧紧、防止扭力反弹。

锁付最小圈数：达目标扭矩时，若转动的圈数小于该参数，代表浮锁。

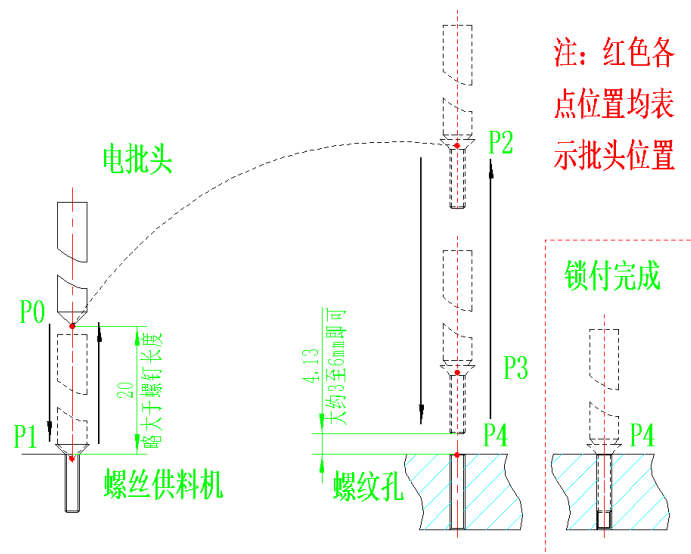
锁付最大圈数：达目标扭矩时，若转动的圈数大于该参数，代表滑牙。

搜索超时时间：超过搜索时间，仍未达到搜索扭矩，表明未找到孔位，此时也定义为滑牙。

(2) 拧紧工艺编程

锁螺丝指令 LockScrew 集成了单个螺丝拧紧的全部过程，使用者可以很方便地进行编程。注意：锁螺丝指令 LockScrew 使用前需要 LoadScrewParm 先加载工程文件，使用后需要 CheckLock 等待锁付完成并检测结果。下面是一个典型的锁螺丝应用示例：

- 1) 加载工程：**程序需要使用 LoadScrewParm 指令，将锁螺丝工程文件中的工艺加载
 - 2) 拾取螺丝：**机器人通过 Movj 运动到起始位置 P[0]，随后移动批头至供料机螺丝吸附预备位 P[1]，等待到达 P[1]点后，通过 Set 指令打开气阀（连接气阀的 IO 口用户自定义），吸附螺丝上升至 P[0]，并快速移动到位置 P[2]，直线运动到锁付准备位 P[3]，
 - 3) 锁付：**执行螺丝锁紧指令 LockScrew。LockScrew 指令控制轴 3 从 P[3]运动到 P[4]，同时使能电批，使得电批轴同步进入拧紧过程，并完成锁付。CheckLock 会不断检测此次拧紧效果，并返回锁付结果。
 - 4) 返回：**锁付完成后通过 Set 指令关闭气阀，直线回到 P[2]，快速回到初始位置 P[0]。
- 注意：切记示教好再再现，防止 P[2]高度过低导致碰到障碍物。



范例：

```
START;
LoadScrewParm("aa.stp",B1);
Movj P[0],V[30],Z[0];
Movl P[1],V[30],Z[0];
WaitInPos;
Set Out[7],ON;
Movl P[0],V[30],Z[0];
Movj P[2],V[30],Z[0];
Movl P[3],V[30],Z[0];
LockScrew (0,P[4],V[30]);
CheckLock B0;
If B0==1
    Print "OK";
Else
    Print "NG";
EndIf;
Set Out[7],OFF;
Movl P[2],V[30],Z[0];
END;
```


(3) 拧紧状态监控

在【监控】-【锁螺丝状态】-【锁付统计】中，监控状态。列表前四项为当前螺丝的锁付状态。后六项为整个锁付过程（含多个螺丝的拧紧）的统计。

其中，第一项参数“单个锁付结果”开始为 NULL，锁付完一个，即会显示结果：

结果	含义
OK	锁付完成，锁付正常
滑牙	未找到孔位或大于最大圈数仍没到达目标扭矩
浮锁	小于最小圈数就到达目标扭矩

The screenshot shows the 'Locking Statistics' (锁付统计) window in the Inovance software. The window has two tabs: 'Locking Statistics' (锁付统计) and 'Locking Waveform' (锁付波形). The 'Locking Statistics' tab is active, displaying a table with the following data:

参数项	值
锁付结果	NULL
终点扭矩(mN·m)	***
锁付周期(ms)	***
锁付圈数	***
锁付总数	0
合格个数	0
滑牙个数	0
浮锁个数	0
NG个数	0
合格率	0%

There is a 'Clear Count' (计数清零) button to the right of the table. The status bar at the bottom shows joint information: Joint: J1:0.000 J2:-0.000 J3:0.000 J4:0.000 J5:0.000 J6:0.000.

4.6.2 螺丝拆解

(1) 配置拧松工艺

与螺丝锁付配置工艺方法类似，先选择工程，再选择“拧松”，然后配置工艺。



*工艺参数解析

一个拆螺丝指令可以完成螺丝拧松的全部操作。其过程分为以下两个阶段：低速搜索-高速拧松，各个阶段具体描述如下：

- 1) 低速搜索阶段：电批以低速“搜索速度”进行空转，这样可以方便批头和螺帽对准。当电批转动到“搜索圈数”左右时，批头接触到螺帽表面，电批则自动切换到高速拧松阶段；
- 2) 高速拧松阶段：该阶段中，机器人轴 3 根据拆指令中的回退高度、回退速度进行运动，电批轴则根据“回退速度”和“拆螺丝牙距”等参数进行拧松。

关键参数解释：

搜索速度：低速搜索阶段寻找孔位的速度，以便批头和螺帽对准。

搜索圈数：当扭矩到达“搜索圈数”时，表明已批头已接触到螺帽，然后进入高速拧松阶段。

拆螺丝牙距：螺丝牙距。

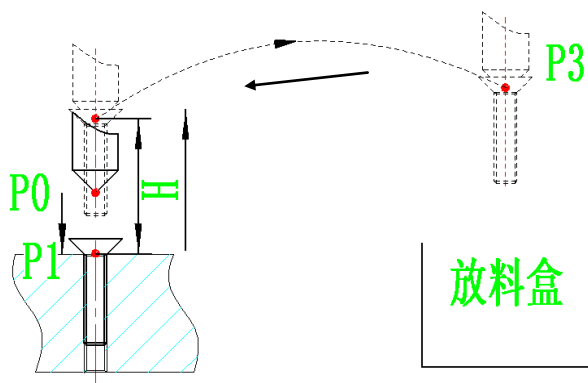
拆螺丝圈数：即拆螺丝所需要拧松圈数。

等待时间：15B 版本已无，可忽略

(2) 拧松工艺编程

拆螺丝指令 UnLockScrew 集成了螺丝拧松的整个过程，因此使用者在编程时不必关心这些过程的具体实现，只需直接使用指令。下面是一个简单的拆螺丝应用示例：

- 1) **加载工程：**程序使用 LoadScrewPam 指令，将锁螺丝工程文件中的拧松参数加载。
- 2) **准备拆卸：**机器人通过 Movj 等运动指令运动到起始位置 P[3]，然后快速移动批头至螺丝拧松预备位 P[0]，等待 P[0]到位后通过 Set 指令打开气阀
- 3) **拆卸：**执行 UnLockScrew 指令使得机器人轴 3 带动电批从 P[0]运动到 P[1]，与此同时，电批同步开始进入拆螺丝的搜索阶段，待批头到达 P[1]点后，批头开始拧松螺丝，而且轴 3 边回退（向上移动），CheckUnLock 会等待拆锁完成，并返回拧松结果。
- 4) **返回：**拆完后运动到起始位 P[3]，也即螺丝放置点，通过 Set 指令关闭气阀，放置螺丝至放料盒。



注：红色各点位置均表示批头位置

程序：

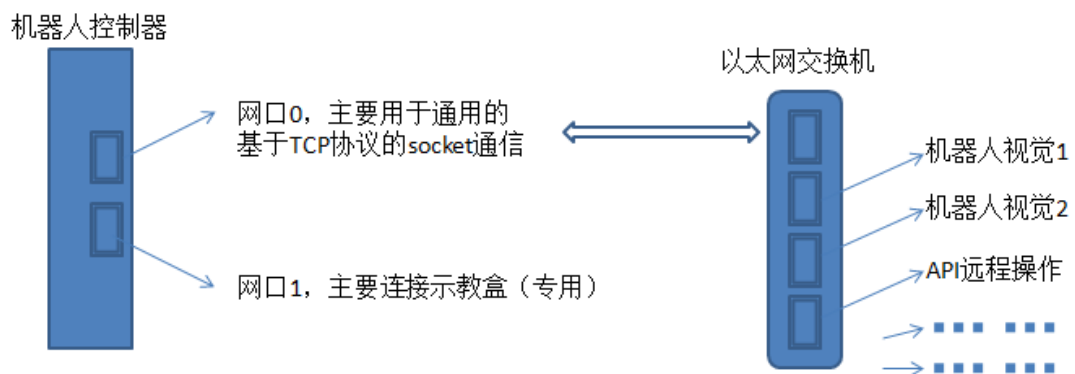
```

START;
LoadScrewPam("aa.stp",B0);
Movj P[0],V[30],Z[0];
WaitInPos;
Set Out[7],ON;
UnLockScrew(0,P[1],V[30],2,12);
CheckUnLock(B1);
If B1==1
    Print "OK";
Else
    Print "NG";
EndIf;
Movl P[3],V[30],Z[0];
WaitInPos;
Set Out[7],OFF;
Delay(0.1);
END;

```

5 其它

5.1 TCP 多端口连接功能



多个视觉或 API 操作通过以太网交换机与机器人控制器相连，并进行通信。
如两台 PC 都以远程连接方式连接控制器，发出 API 指令，控制器能处理两边发出的指令。

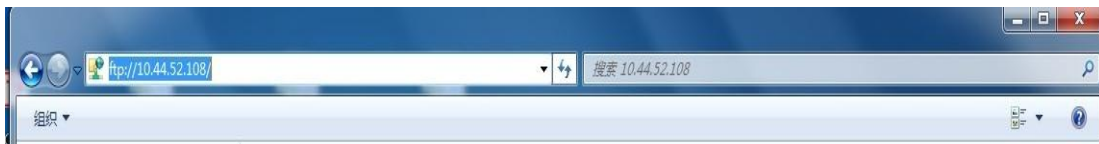
5.2 控制器 FTP 服务器功能

控制器加入 FTP 服务器功能，可通过 FTP 协议实现对控制器文件的远程访问及修改，控制器 FTP 服务器为用户提供的账户名为 robot，登入密码为 123456，登入后的目录默认为控制器 SD 卡目录。

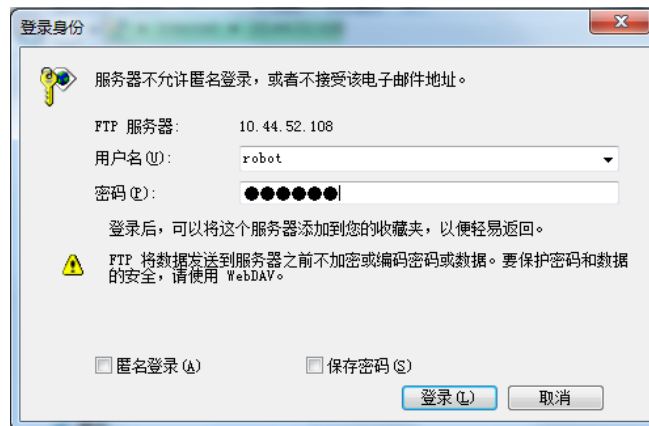
注意：使用 FTP 功能需要保证控制器中已插入 SD 卡，否则可能会导致连接失败情况。

示例——从控制器中取出机器人程序“A.pro”：

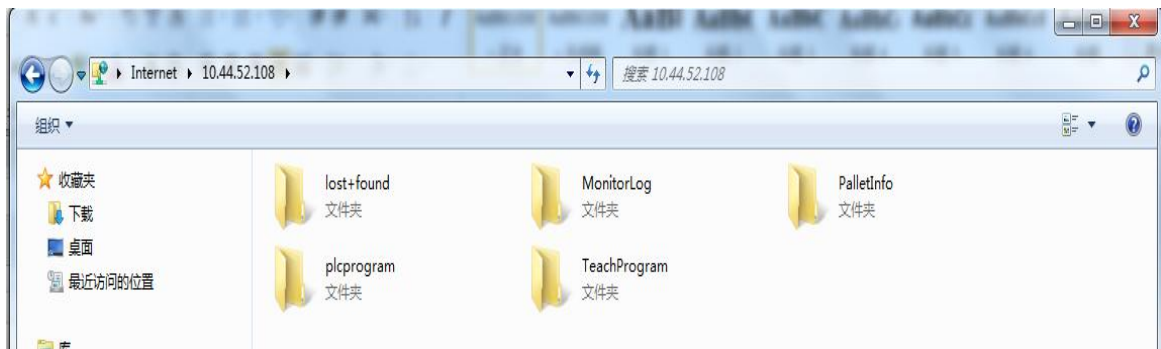
假设某计算机与控制器通过网络连接，得知控制器网络地址为 10.44.52.108，在计算机的文件浏览器的地址栏输入 ftp://10.44.52.108，如下图所示：



敲入回车键，即可弹出登入界面，在界面中输入用户名及密码。（账户名为 robot，登入密码为 123456）



登入后，默认的目录为控制器上的 SD 卡目录：



从 TeachProgram 文件夹中，将 A.pro 文件复制出即可。

需要说明的是，FTP 登入方式有许多种，可采用文件管理器方式，DOS 命令行方式，一些常用的 FTP 软件或者可通程序编写规定的 FTP 用户协议，对控制器文件及文件夹进行操作。

5.3 权限管理

5.3.1 控制器系统控制权

控制器系统控制权是设备对机器人进行控制操作的先决条件，只有获取控制权后，设备才能实现对机器人的当前所有功能，否则只能读取和监控机器人状态。系统控制权同一时刻只能属于一个设备。

在示教器的【系统设置】-【其他设置】-【控制设备】中，能对权限进行管理。修改控制权限需要编辑及以上权限，且必须在机器人处于非运动状态下。

系统出厂默认的控制权为示教器所有。当需要其他设备（InoRobShop、远程以太网设备、IO设备、Modbus 设备）控制机器人时，必须通过示教器切换控制权至对应设备。有需要时，也可将控制权切换回示教器。

注意：

控制权切换不需重启机器人控制器。

无论控制权为哪类设备所属，硬件上的急停开关始终有效。

a) 示教器取得控制权

当系统控制权为示教器所有时，示教器能操作控制器。其他设备只能读取或观测参数，不能修改参数或操纵控制器。

b) InoRobShop 取得控制权

当系统控制权为 InoRobShop 所有时，InoRobShop 能操作控制器。其他设备只能读取或观测参数，不能修改参数或操纵控制器。

c) 远程以太网设备取得控制权

当系统控制权为远程以太网设备所有时，远程以太网设备能操作控制器。其他设备只能读取或观测参数，不能修改参数或操纵控制器。

最多同时可以有 4 个连接远程以太网设备与机器人连接。当系统控制权由其他设备切换至以太网设备时，则第一次 4 个以太网设备均无实际控制权，需要通过指令申请获得控制权。

一个以太网设备申请到了系统控制权，就可实现对控制器的控制。其它三个设备没有控制权，只能进行参数读取和状态监控。但他们均可通过指令申请强制获取控制权，获取成功后原有控制权的设备将失去控制权。控制权在以太网设备上后，以后每次开机默认控制权在第一个以太网设备上。

d) 远程 IO 设备获取控制权

当系统控制权为远程 IO 设备所有时，远程 IO 设备能操作控制器。其他设备只能读取或观测参数，不能修改参数或操纵控制器。工位预约就可以是此种情形的应用。

e) 远程 Modbus 设备获取控制权

当系统控制权为远程 Modbus 设备所有时，远程 Modbus 设备能操作控制器。其他设备只能读取或观测参数，不能修改参数或操纵控制器。工位预约就可以是此种情形的应用。

5.3.2 IRLink 组态配置权限

可以通过 InoRobShop 或示教器两种途径配置 IRLink 模块。

只有当 InoRobShop 未配置 IRLink 主站时，示教器才可以配置 IRLink 模块。一旦通过 InoRobShop 配置 IRLink 模块后，示教器上不可以再配置 IRLink 模块。除非再次利用 InoRobShop 配置一个空的 IRLink 模块。无论哪种方式，更改配置后需要重启系统才能生效。

目前 SCARA 系统出厂默认由示教器配置 IRLink，且配置为 2 个 0808 模块。

5.3.3 IO 控制权

这里的 IO 控制主要指输出端口（DO、DA 等）的控制，不包含输入端口。

控 制 权	
RC	系统占用：【外设配置】-【IO 配置】，将输出端口与特定系统功能绑定。此时，输出端口信号只与功能相关。不能人为更改信号状态。
	非系统占用：一般状态下的信号端口。能人为更改信号状态，在【IO 监控】中变更或通过 Set 指令变更。
PLC	PLC 拥有控制权，端口输出只听从 InoRobShop 等 PLC 控制

当使用 InoRobShop 配置 IRLink 组态时，默认配置模块的前 16 个 DO 端口（两个 0808 或一个 0016）的控制权均属于 RC。其后配置的 DO 端口的控制权默认为 PLC。默认配置模块的 AO 端口控制权均属于 PLC。利用 InoRobshop，可进行控制权 RC\PLC 切换，在线生效无需重启控制器。

当使用示教器新增 IRLink 模块时，IO 控制权均默认为 RC，且示教器无法将控制权修改为 PLC。但在【外设配置】-【I/O 配置】中，可将某项功能关联给某个 DO，从而变为系统占用。

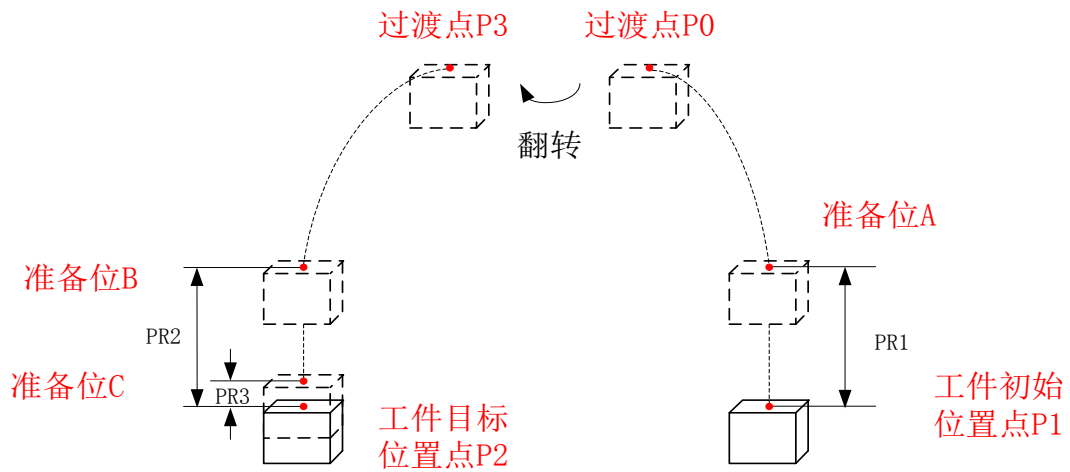
6 应用案例

案例一：六自由度机器人搬运

任务描述：采用六自由度机器人吸附工件，从一处搬运至另一处。工件的起始方位与终点方位存在姿态上不同。

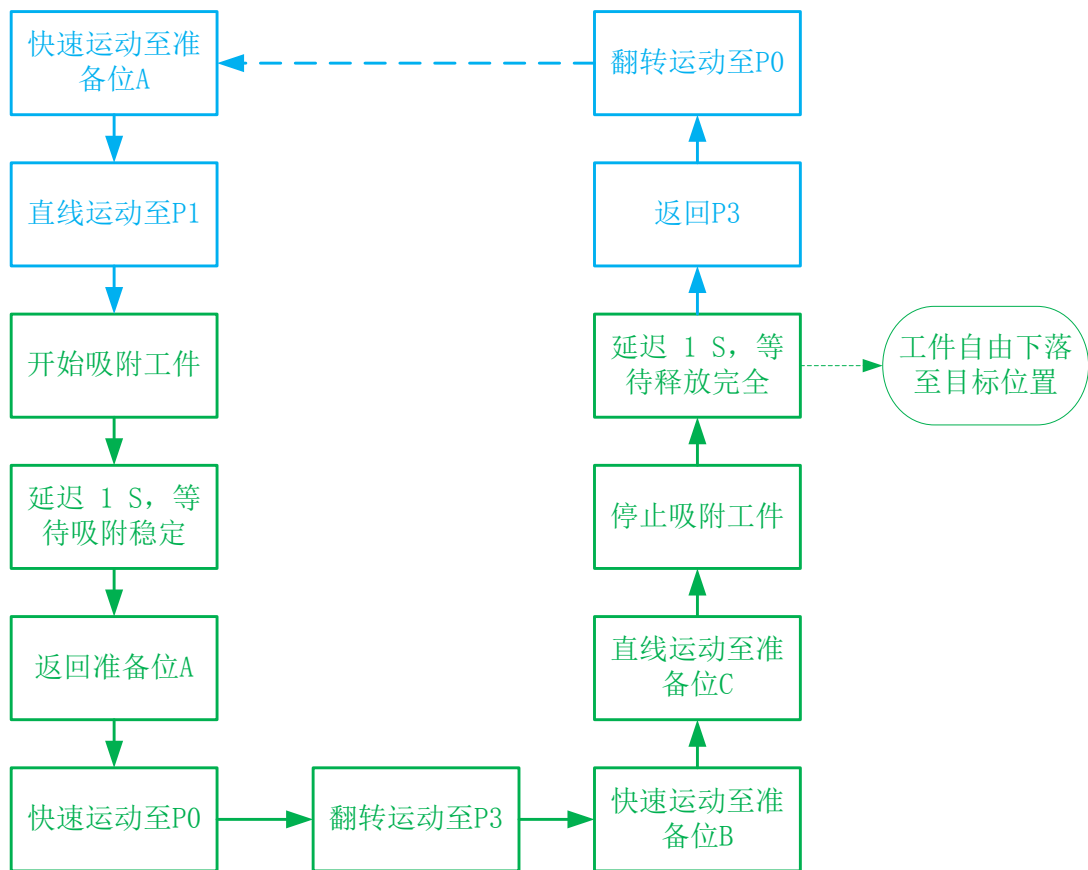


任务分析：



位置	表达形式	任务描述
工件初始位置	P1	在该点执行吸附操作，将工件拾取。
工件目标位置	P2	工件想要达到的点
准备位 A	Offset(P[1],PR1)	拾取工件的准备位置，快速运动和直线插补的临界点
准备位 B	Offset(P[2],PR2)	释放工件的准备位置，快速运动和直线插补的临界点
准备位 C	Offset(P[2],PR3)	释放工件的位置
过渡点 P0	P0	在 P0 与 P3 间翻转工件，在安全高度完成姿态变化
过渡点 P3	P3	同上

流程图：



编程:

START;

Movj Offset(P[1],PR1),V[100],Z[2]; ##移动至 P[1]上方准备位

Movl P[1],V[50],Z[0]; ##移动到 P[1]处

Set Out[1],ON; ##开启吸附装置

Delay T[1]; ##延时 1S, 等待吸附稳定

Movl Offset(P[1],PR1),V[100],Z[2]; ##移回准备位 A

Movj P[0],V[100],Z[2]; ##运动至过渡点 P0

Movj P[3],V[100],Z[2]; ##翻转运动至过渡点 P[3], 负责工件姿态的变换

Movj Offset(P[2],PR2),V[100],Z[2]; ##移动至准备位 B

Movl Offset(P[2],PR3),V[50],Z[2]; ##准备位 C 为距离 P[2]上方微小间隙的位置, 留给空间供吸力释放

Set Out[1],OFF; ##关闭吸附装置

Delay T[1]; ##延迟 1S, 工件下落

Movj P[3],V[100],Z[2]; ##退回到 P3

Movj P[0],V[100],Z[2]; ##退回到 P0

.....

.....

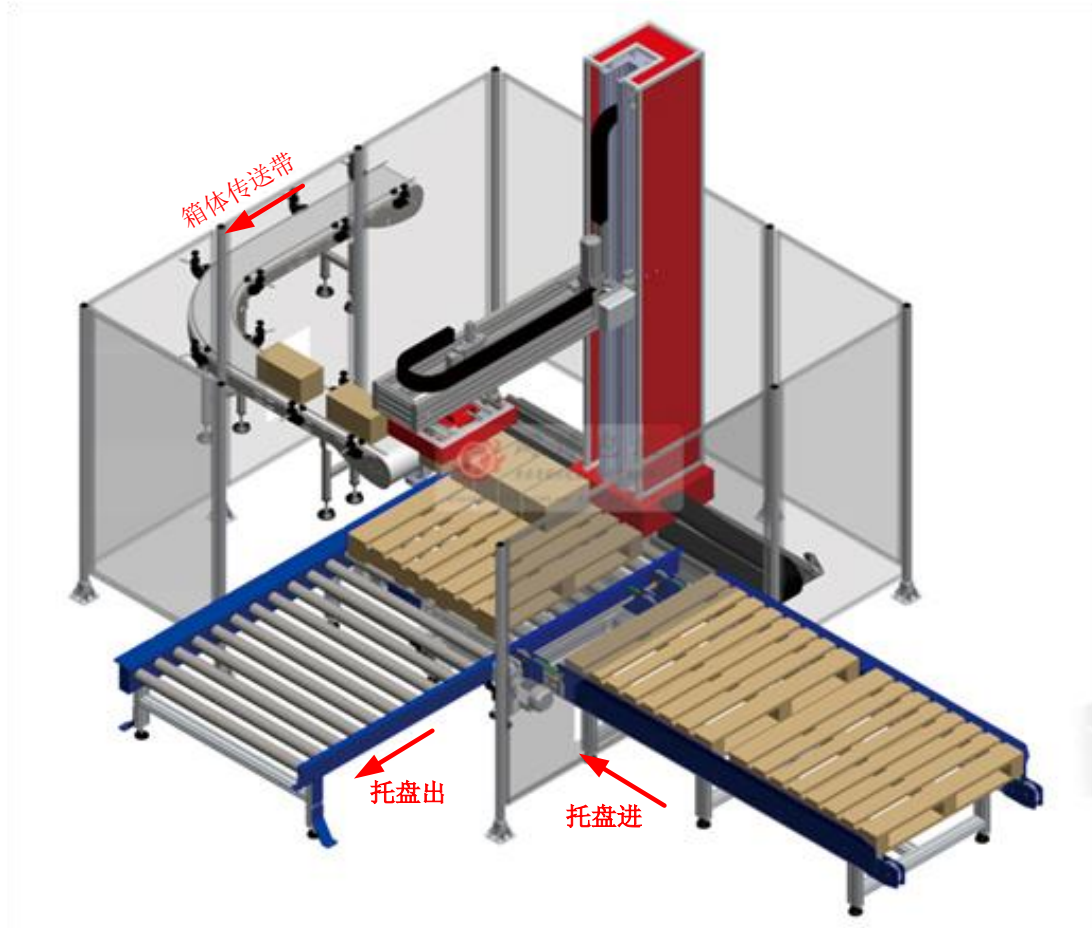
.....

END;

案例二：直角坐标机器人码垛应用

工作场景

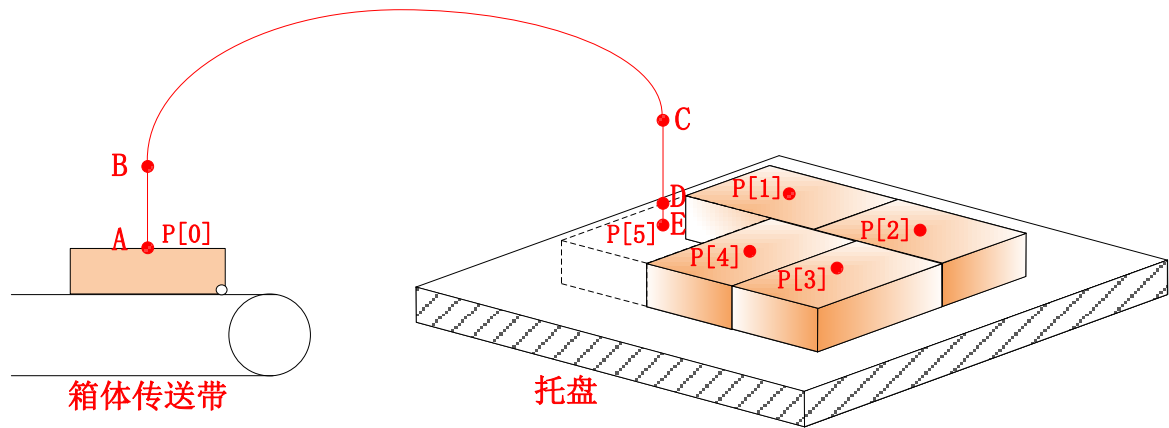
如图所示，是流水线生产中的一个环节，上个环节已完成产品装箱，本环节需要完成码垛，将箱体按顺序摆排放放在托盘上，每层放 5 个，共 4 层，相邻两层间排列顺序不同。



任务分析

完整的工作流程包括箱体输送、机器人码垛以及托盘输送，箱体和托盘的输送由传送带、光电开关和 PLC 实现，码垛由直角坐标机器人完成，本文只介绍机器人码垛部分。

如图所示，箱体传送带将箱体传送到 A 位置，机械手需要从 A 处抓取箱体，按顺序摆放到托盘上，箱体的拾取通过气压吸盘实现。机器人末端运动轨迹为 A-B-C-D-C-B-A，循环操作，其中 C、D、E 点随箱体摆放位置的不同而变化。

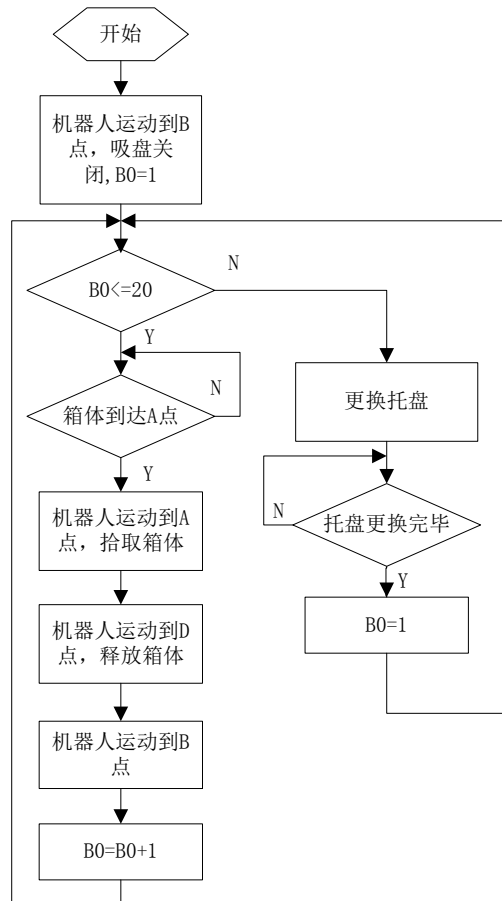


位置	表达形式	任务描述
A	P[0]	箱体拾取位置，在该点打开吸盘，拾取箱体，通过示教得到。
B	Offset(P[0],PR0)	拾取过度位置，在 A 点正上方，偏移量 PR0。
E	P[n]	箱体摆放点， $n=1\sim 20$ ，不同箱体位置、姿态不同，通过示教得到。
C	Offset(P[n],PR1)	摆放过度点，在该点前完成姿态调整，在 E 点正上方，偏移 PR1。
D	Offset(P[n],PR2)	摆放释放点，在该点松开吸盘，箱体靠重力落下，在 E 点正上方，偏移 PR2。

工作流程

机器人工作流程如图所示，箱体传送带上装有传感器检测箱体是否到达指定位置，传感器信号接到机器人控制器的 IN[2]，同时该信号用来触发箱体传送带的启停。实际使用中将传送带调节到适当的速度，避免机器人等待。机器人循环码放 20 个箱体后需要更换托盘，通过 OUT[2]触发更换，通过 IN[3]检测是否更换完毕。

机器人工作流程图如下图所示：



程序代码

```

START;
Home[0];                                ##移动到工作原点 0
Movj  Offset(P[0],PR0),V[100],Z[2];     ##快速移动至 B 点
Set Out[1],OFF;                          ##端口 Out[1]为 OFF, 关闭吸盘。
B0=1;                                     ##变量初始化
L[0]:                                     ##与后文 GOTO 配合使用, 循环操作
For (B0=1,B0<21,Step[1])                ##每个托盘放 20 个箱体, 循环操作 20 次
    Wait In[2] == OFF,T[10]              ##等待箱体被传送到 A 位置
    Movl P[0],V[50],Z[0];                ##直线插补移动至 A 点
    Set Out[1],ON,T[0];                  ##启动吸盘, 吸附箱体
    Delay T[1];                          ##延时 1S, 等待吸附稳定
    Movl Offset(P[0],PR0),V[100],Z[2];   ##直线插补移动至 B 点
    Movj  Offset(P[B0],PR1),V[100],Z[2]; ##快速移动至 C 点, 准备摆放第 B0 个箱体
    Movl  Offset(P[B0],PR2),V[50],Z[2];  ##直线插补移动至 D 点
    Set Out[1],OFF;                      ##关闭吸盘, 放下箱体
    Delay T[1];                          ##延迟 1S, 等待吸附结束, 箱体落下
    Movj  Offset(P[0],PR0),V[100],Z[2];   ##快速移动至 B 点, 准备拾取下一个箱体
EndFor;
Set Out[2],ON;                          ##输出端口 2 置 ON, 通知更换托盘
Wait In[3] == OFF,T[10]                  ##等待托盘更换完毕
  
```

```
Set Out[2],OFF;
B0=1;
Goto L[0];
END;
```

```
##输出端口 1 复位
```

```
##变量置 1，开始下一个托盘
```

附录一：机器人报警及处理方法列表

故障码	中文注释	故障原因	处理方法
0x0001	初始化失败	1. 创建或打开 ParaFile.PF 文件失败; 2. 创建或打开 ComErrorFile.PF 文件失败; 3. 创建或打开 ServoWarnFile.PF 文件失败;	检查系统硬件; 断电重新启动;
0x0002	启动示教盒通信线程失败	没有正常启动示教盒通讯线程或硬件损坏	断电重新启动或更换硬件
0x0003	启动视觉通信线程失败	没有正常启动视觉通讯线程或硬件损坏	断电重新启动或更换硬件
0x0004	启动 DSP 通信线程失败	没有正常启动 dsp 通讯线程或硬件损坏	断电重新启动或更换硬件
0x0005	启动调度线程失败	没有正常启动 ARM 调度线程或硬件损坏	断电重新启动或更换硬件
0x0006	启动插补测试线程失败	没有正常启动插补测试线程或没有开放内部测试功能	检查软件版本
0x0007	EtherCAT 通信打开失败	1. 配置文件错误; 2. EtherCat 从站与系统配置不符;	1. 恢复出厂设置, 并重新上电; 2. 检查从站配置
0x0008	打开参数配置文件失败	参数配置文件开发失败或文件损坏	恢复出厂设置, 并重新上电
0x0009	译码错误	程序语法错误	检查程序编写规范
0x000A	译码行号错误	示教盒发送行号指令超出范围	检查示教程序是否有误
0x000B	IO 等待时间超时	I0 等待的时间超出设置时间	1. 检查 I0 端口; 2. 重新设置等待时间
0x000C	读取指令错误	1. 示教文件损坏; 2. 示教文件编写不符合规范	1. 重新示教文件; 2. 检查示教程序编写规范;
0x000D	子程序不容许嵌套调用	子程序有嵌套调用	更改示教程序
0x000E	运动指令译码错误	运动指令译码错误	检查示教程序
0x000F	无法找到初始化文件	初始化文件损坏或丢失	1. 恢复出厂默认值, 重新上电; 2. 更换硬件
0x0010	再现数据计算错误	1. 示教点取点错误; 2. 示教点在奇异范围	重新选取示教点

0x0011	创建轴插补线程失败	1. 测试插补线程创建失败; 2. 内部测试功能没有开放;	1. 更换硬件; 2. 更换软件版本;
0x0012	jump 指令失败	1. jump 指令中点数据计算错误	重新选取示教点
0x0013	IRLink 初始化失败	1. IRLink 从站个数配置错误; 2. IRLink 从站顺序配置错误;	重新确认 IRLink 配置
0x0014	示教程序保存失败	SD 卡松动或无法识别	检查 SD 卡
0x0015	DSP 通信错误	DSP 软件运行错误	控制器重新上电
0x0016	DSP 使能错误	DSP 软件运行错误	控制器重新上电
0x0017	DSP 程序运行错误	DSP 软件运行错误	控制器重新上电
0x0018	回零失败	相对编码器回零失败	重新进行回零
0x0019	使能缺失	运行状态丢失使能	检查系统是否使能状态
0x0020	停止到启动过快	停止到启动过快	慢速将停止切换至启动
0x0021	传入参数错误	译码传入参数错误	检查指令参数
0x0022	找不到指令行	输入行号超出范围	选择运行的指令行
0x0023	找不到点数据	点未定义	检查点是否定义
0x0024	数据逆解计算错误	该点在机器人的奇异点	修改该点的坐标
0x0025	点数据坐标系参数错误	坐标系值超出范围	重新取点
0x0026	直线或圆弧指令臂参数不允许突变	MOVL MOVC 指令不允许臂参数突变	重新取点或者增加关节过度点
0x0027	V 参数超出范围	V 参数超出范围 (1-100)	修改 V 参数
0x0028	Z 参数超出范围	Z 参数超出范围 (0-5)	修改 Z 参数
0x0029	TOOL 参数超出范围	TOOL 参数超出范围(0-15)	修改 Tool 参数
0x002A	USER 参数超出范围	USER 参数超出范围(0-15)	修改 User 参数
0x002B	ACC 参数超出范围	ACC 参数超出范围(1-100)	修改 Acc 参数
0x002C	until 参数超出范围	IO 号超出范围(0-255)	修改 Until In 参数
0x002D	Pallet 参数错误	Pallet (PNo, i, j, k), PNo, i, j, k 大于等于 0	修改 Pallet 参数
0x002E	托盘未定义	托盘序号未定义	托盘定义后使用
0x002F	Repeat 参数错误	Repeat 参数超出范围	修改 Repeat 参数
0x0030	运行译码错误	运行过程中指令解析出错	检查运行的指令, 根据提示修改
0x0031	基坐标系寸动时数据逆解运算错误	基坐标系下寸动目标位置超出运行空间或处于奇异区	1、 检查寸动步长大小 2、 检查寸动方向
0x0032	工具坐标系寸动时数据逆解运算错误	工具坐标系下寸动目标位置超出运行空间或处于奇异区	1、 检查寸动步长大小 2、 检查寸动方向
0x0033	用户坐标系寸动时数据逆解运算错误	用户坐标系下寸动目标位置超出运行空间或处于奇异区	1、 检查寸动步长大小 2、 检查寸动方向
0x004D	以太网通信出错	以太网通信数据不完整	检查通信线路环境, 重新传输数据;
0x004E	pipe 出错	以太网被多个终端连接	检查是否有多个终端连接同一个控制器
0x004F	视觉指令错误	视觉指令打开错误	检查视觉指令
0x0050	SD 卡拔出	SD 卡连接后移除	检查 SD 卡的连接情况

0x0051	EtherCAT 断开	EtherCAT 连接后断开	检查EtherCAT通信情况
0x0052	IRLink 断开	IRLink 连接后断开	检查 IRLink 通信情况
0x0053	启动和暂停相隔时间太短	启动和暂停间隔时间短	重新启动运行
0x0054	获取视觉特征值失败	执行指令时视觉特征值获取失败	检查视觉处理后重新获取视觉特征值
0x0055	译码未完成	程序可能存在指令符号或语法错误	检查编辑程序的指令语法
0x0056	机器人类型错误	机器人没有相关的类型固件	重启或检查 DSP 固件
0x0057	FPGA 初始化错误	FPGA 固件错误或启动异常	重启或检查 FPGA 固件
0x0058	DSP 初始化错误	FPGA 固件错误或启动异常	重启或检查 DSP 固件
0x0059	运行状态模式转换	运行时错误进行模式转换	检查模式设置
0x005A	设置 I0 参数错误	I0 相关参数设置错误	检查 I0 参数设置情况
0x005B	IP 地址错误	IP 地址获取或者设置错误	检查网线连接情况
0x005C	示教器无 IRLink 配置权	二次开发平台已经配置 IRLink	使用当前配置，或者取消二次开发配置
0x005D	共享内存映射错误	RC 与 PLC 的共享内存映射错误	联系技术支持
0x005E	(用户自定义报警 1)	系统触发了用户定义的报警	检查用户报警
0x005F	(用户自定义报警 2)	同上	同上
0x0060	(用户自定义报警 3)	同上	同上
0x0061	(用户自定义报警 4)	同上	同上
0x0062	(用户自定义报警 5)	同上	同上
0x0063	(用户自定义报警 6)	同上	同上
0x0064	(用户自定义报警 7)	同上	同上
0x0065	(用户自定义报警 8)	同上	同上
0x0066	(用户自定义报警 9)	同上	同上
0x0067	(用户自定义报警 10)	同上	同上
0x0068	(用户自定义报警 11)	同上	同上
0x0069	(用户自定义报警 12)	同上	同上
0x006A	(用户自定义报警 13)	同上	同上
0x006B	(用户自定义报警 14)	同上	同上
0x006C	(用户自定义报警 15)	同上	同上
0x006D	(用户自定义报警 16)	同上	同上
0x006E	干涉区 1 报警	机器人处于干涉区域内	检查机器人位置和干涉区域设置值
0x006F	干涉区 2 报警	同上	同上
0x0070	干涉区 3 报警	同上	同上
0x0071	干涉区 4 报警	同上	同上
0x0072	干涉区 5 报警	同上	同上
0x0073	干涉区 6 报警	同上	同上
0x0074	干涉区 7 报警	同上	同上
0x0075	干涉区 8 报警	同上	同上
0x0076	干涉区 9 报警	同上	同上

0x0077	干涉区 10 报警	同上	同上
0x0078	干涉区 11 报警	同上	同上
0x0079	干涉区 12 报警	同上	同上
0x007A	干涉区 13 报警	同上	同上
0x007B	干涉区 14 报警	同上	同上
0x007C	干涉区 15 报警	同上	同上
0x007D	干涉区 16 报警	同上	同上
0x007F	数据流模式未关闭	系统处于数据流模式	解除数据流模式
0x0080	运行模式按下急停	急停键被按下	解除急停，清除报警
0x0081	无后退数据	后退数据已经执行完毕	终止后退操作
0x0082	关闭端口号错误	端口号不在设定范围或端口号根本没有被打开过	检查端口号
0x0083	TCP 端口溢出	外围 TCP 应用连接过多	关闭无用的 TCP 连接
0x0084	API 处理错误	API 通道被其它应用长时间暂用或之前的 API 应用处理出现故障阻塞	关闭或减少之前 API 应用工艺
0x0085	圆弧轨迹不可控	圆弧起始点是不确定的点	圆弧指令前后要增加其它运动指令
0x0089	IP 冲突	IP 设置冲突	重新设置 IP
0x008A	SD 卡未识别到正确的文件系统	SD 卡上的文件系统不正确	更换或重新格式化 SD 卡
0x008B	伺服参数读取失败	控制器读取伺服参数失败	优化伺服与控制器连接环境
0x008C	锁螺丝工艺组不在范围内	工艺组只有 16 组，设置并非在范围内	重新设置工艺组，确保在范围内 0~15
0x008D	不合法的 I/O 配置	配置的 I/O 被 PLC 控制或不存在	重新配置 I/O
0x008E	不合法的 I/O 设置操作	设置的 I/O 缺少 I/O 控制权	检查 I/O 控制权
0x0090	拧紧启动失败	锁螺丝启动参数发送至伺服端失败	1、检查电批伺服固件是否和控制器匹配 2、清零重启
0x0091	电批停止失败	锁螺丝停止参数发送至伺服端失败	1、检查电批伺服固件是否和控制器匹配 2、清零重启
0x0092	螺丝状态检测失败	从伺服端读取锁螺丝锁状态失败	1、检查电批伺服固件是否和控制器匹配 2、清零重启
0x0093	读拧紧设置参数失败	从伺服端读取锁螺丝设置参数失败	1、检查电批伺服固件是否和控制器匹配 2、清零重启
0x0094	写拧紧设置参数失败	往伺服端写锁螺丝设置参数失败	1、检查电批伺服固件是否和控制器匹配 2、清零重启
0x0095	锁螺丝数据显示失败	从伺服端获取锁螺丝显示数据失败	1、检查电批伺服固件是否和控制器匹配

			2、 清零重启
0x0096	锁螺丝计数清零失败	往伺服端写清锁螺丝计数器标志失败	1、 检查电批伺服固件是否和控制器匹配 2、 清零重启
0x0097	伺服错误获取线程失败	伺服错误获取线程死掉	重新启动机器人
0x0098	拧松启动失败	拆螺丝启动参数发送至伺服端失败	1、 检查电批伺服固件是否和控制器匹配 2、 清零重启
0x0099	写拧松设置参数失败	往伺服端写拆螺丝设置参数失败	检查电批伺服固件是否和控制器匹配
0x009A	读拧松设置参数失败	从伺服端获取拆螺丝设置数据失败	检查电批伺服固件是否和控制器匹配
0x009B	拧松回退点设置超限	回退点设置超限	将拧松回退点设置在限位范围内
0x00A0	TCP 端口错误	TCP 端口设置错误	重新设置端口
0x00A1	动态视觉没有关闭	动态视觉没有关闭情况下使用普通视觉	关闭动态视觉
0x00A2	传送带错误或相机像素错误	传送带或相机的传输数据类型错误	重新设置传送带号或相机传输数据类型
0x00A3	动态视觉转换错误	动态视觉像素和相机坐标系转换出错	检查转换矩阵
0x00A4	无后退数据	没有可以后退的数据	清除报错
0x00A5	圆弧运动前缺乏 Mov j 或 Mov l	圆弧运动前一条指令不是 Mov j 或 Mov l（仅在单步示教时出现）	保证圆弧指令前一条指令是为 Mov j 或 Mov l
0x00A6	找不到指定文件	文件不存在或文件路径错误	检查文件是否存在或文件路径是否正确
0x00A7	保存状态错误	锁螺丝机保存状态错误	重新保存，或检查参数范围
0x00A8	导出状态错误	锁螺丝机导出状态错误	重新导出
0x1001	目录存在重复创建	将要创建目录此路径下已经存在	更换要创建的目录名
0x1002	内存操作错误，不存在前级目录	当前创建的目录不存在父目录	更换路径重新创建目录
0x1003	重命名原目录不存在	为不存在的目录重命名	刷新目录，检查目录是否存在
0x1004	删除目录不存在	将要删除的目录不存在	刷新目录，检查目录是否存在
0x1005	发送目录错误(不是目录或者不存在)	被要求发送给上位机的目录不合法	刷新目录，检查目录是否存在
0x1006	创建文件，内存出错，路径错误	创建路径有错	刷新文件，检查创建路径
0x1007	重命名原文件不存在	为不存在的文件重命名	刷新文件，检查文件是否存在
0x1008	删除文件不存在或者是路径不存在	删除文件不存在	刷新文件，检查文件是否存在

0x1009	文件的给定路径不存在	给定创建的文件路径不合法	检查创建路径的合法性
0x100A	发送的是非文件	发送的文件不是文件	手持盒端检查要求发送的是否是文件
0x100B	发送的是非目录	发送的目录不是目录	手持盒端检查要求发送的是否是目录
0x100C	帧顺序错误	大文件发送过程中帧顺序错误	重新发送文件
0x100D	异常导致网线断开	1. 不按正常操作关闭手持盒; 2. 异常的断网	1. 非正常操作的错误,请在关闭时候主动断开; 2. 异常要结合目前已有的错误码检查错误原因
0x100E	设置时间格式错误	设置时间格式错误	按照用户手册给出正确的时间格式
0x100F	执行系统校时出错	系统计算时间回路有错	检查当前网络连接环境
0x1010	执行 RTC 校时出错	RTC 外部电池不存在或者电量不足	重新更换电池或者检查当前硬件
0x1011	执行拷贝出错	拷贝文件过程操作不当	参考用户手册重新执行拷贝操作
0x1012	执行剪切出错	剪切文件过程操作不当	参考用户手册重新执行剪切操作
0x1013	视觉设备通信异常	1. 无法连接视觉设备 2. 通讯网络异常断开	1. 检查视觉设备是否正常; 2. 检查当前网络连接环境
0x1101	ARM 从 DSP 得到数据长度无效	向 DSP 请求数据时, DSP 未按要求返回有效数据	检查 dsp 软件版本或更换硬件
0x1102	ARM 从 DSP 获得数据的校验和出错	向 DSP 请求数据时, DSP 未按要求返回有效数据或返回数据有误码	检查 dsp 软件版本或更换硬件
0x1103	ARM 往 FPGA 写入块数据出错	GPMC 通道异常	检查 dsp 软件版本或更换硬件
0x1104	ARM 从 FPGA 读入块数据出错	GPMC 通道异常	检查 dsp 软件版本或更换硬件
0x1105	FPGA 上的块数据缓冲区满	FPGA 缓冲区已满, 不能接受新数据	延时一段时间后再次尝试写入数据
0x1106	打开 ARM 与 DSP 间的通道出错	GPMC 通道异常或已经被占用	重启
0x1107	打开 ARM 与 DSP 间的通道出错	GPMC 通道异常或已经被关闭	重启
0x1108	DSP 处于 BUSY 状态, 对 ARM 指令无应答	DSP 对 ARM 指令不做回应	检查 DSP 状态, 是否被暂停或终止运行
0x1109	当前线程试图获取保护 ARM 与 DSP 通道的信号量出错	CPMC 通道被频繁占用, 当前不能申请使用 CPMC 资源	延时一段时间后再次尝试使用

0x110A	等待 DSP 应答超时	DSP 对 ARM 指令应答时间超过设置最大等待时间	检查 DSP 状态, 是否被暂停或终止运行
0x110B	DSP 未能成功执行 ARM 下发的指令	DSP 对 ARM 指令执行不成功	检查 dsp 软件版本或更换硬件
0x110C	ARM 设置给 DSP 的参数非法	ARM 设置给 DSP 的参数不正确, 如超出参数范围	检查函数调用接口的参数, 确保参数无误
0x110D	ARM 设置给 DSP 的命令非法	ARM 向 DSP 请求的命令无效	检查名字字, 确保 DSP 中有对该命令的处理
0x110E	系统配置的轴数和在线扫描的轴数不一致	机型不匹配或者伺服有掉线	检查机型及伺服连线, 确保当前使用机器人和配置的机器人一致
0x110F	发送给 DSP 的轴数据和再次从 DSP 读取的数据不一致	数据校验出错	检查软件版本或更换硬件
0x1110	发送给 DSP 的 IO 数据和再次从 DSP 读取的数据不一致	数据校验出错	检查软件版本或更换硬件
0x1111	EtherCAT 指令控制伺服进入 Homing 模式时出错	伺服无法进入 Homing 模式	检查软件版本或更换硬件
0x1112	EtherCAT 指令控制伺服退出 Homing 模式时出错	伺服无法退出 Homing 模式	检查软件版本或更换硬件
0x1113	EtherCAT 指令在设置伺服 Homing 参数时出错	所设置 Homing 参数伺服不接受	检查软件版本或更换硬件
0x1114	ARM 设置给 DSP 相关参数时数据校验错	数据校验出错	检查软件版本或更换硬件
0x1115	打开 GPMC 通道时参数错	GPMC 通道号不正确	默认通道为 0, 确保通道号正确
0x1116	GPMC 通道内存映射出错	GPMC 数据通道无法映射至内存	检查软件版本或更换硬件
0x1117	以内存映射方式打开 GPMC 通道出错	GPMC 通道异常	检查软件版本或更换硬件
0x1118	以 IOCTL 控制方式打开 GPMC 通道出错	GPMC 通道异常	检查软件版本或更换硬件
0x1119	GPMC 设备处于错误状态	GPMC 通道异常	检查软件版本或更换硬件
0x111A	以 IOCTL 方式打开 DMA 通道出错	GPMC 通道异常	检查软件版本或更换硬件
0x111B	成功读取的数据长度和期望的不一致	GPMC 通道异常	检查软件版本或更换硬件
0x111C	成功写入的数据长度和期望的不一致	GPMC 通道异常	检查软件版本或更换硬件
0x111D	ARM 上程序申请内存出错	系统无法分配所要求内存	查看系统内存是否接近消耗极限或申请内存是

			否过大
0x111E	IR-LINK 总线上配置 IO 数据偏置信息出错	数据校验出错	检查软件版本或更换硬件
0x111F	IR-LINK 总线上配置 AD 数据偏置信息出错	数据校验出错	检查软件版本或更换硬件
0x1120	IR-LINK 总线上配置 DA 数据偏置信息出错	数据校验出错	检查软件版本或更换硬件
0x1121	IR-LINK 总线上配置 Encoder 数据偏置信息出错	数据校验出错	检查软件版本或更换硬件
0x1122	IR-LINK 总线上配置 AD 参数（量程）出错	数据校验出错	检查软件版本或更换硬件
0x1123	IR-LINK 总线上配置 DA 参数（量程）出错	数据校验出错	检查软件版本或更换硬件
0x1124	IR-LINK 总线上设置模块数码出错	数据校验出错	检查软件版本或更换硬件
0x1125	将控制器规划位置与编码器反馈位置进行同步时出错	当前轴不存在或者 GPMC 通道异常或 DSP 固件异常	检查软件版本或更换硬件
0x2001	段数据重合	前次输入和本次输入的目标位置一样	重新示教点
0x2002	圆弧输入参数计算错误	无法计算出圆弧插补信息	重新示教其他点计算圆弧
0x2003	直线输入参数计算错误	无法计算出直线插补信息	重新示教其他点计算直线
0x2004	逆解运算错误	出现速度过大或出现编码器位置突变	下伺服，然后切换至关节模式，清除报警后再上伺服，并将机器人移动至合适位置
0x2005	奇异位置错误报警	机器人运动到奇异位置点，如 2,3 关节拉直，5 关节处于 0 度附近	切换至关节模式，然后移动机器人离开奇异位置点
0x2006	再现运动中出现掉伺服	可能某个关节的驱动器出现故障	检查驱动器是否出现异常
0x2007	保留	姿态变化太大	重新示教其他点计算本段规划
0x2008	IO 的 Index 访问范围超出	IO 访问物理端不存在	检查是否有对应的物理 IO 模块
0x2009	jump 参数设置错误	最大高度大于限位，或起始位置大于最大高度，或终止位置大于最大高度	修改对应的参数，修改限高，或重新选取起始位置或终止位置

0x200A	臂型参数错误	直线或者圆弧运动时，对于6关节机器人，前一个运动和后一个运动中，3和5关节的臂型，有穿越了0度位置；对于scara，第二关节穿越了0点位置	将后条运动指令改成movJ或者重新取点，保证臂型一致
0x200B	设置运动特性参数不合理	运动参数输入范围不合理	重新修改运动参数
0x200C	DA操作错误	通道配置成电流输出，但使用了电压指令，或配置成电压，使用了电流指令操作	使用与配置一致的指令操作DA端口
0x200D	发出伺服使能命令，但实际反馈未使能	伺服主电未上	检查控制柜的强电按钮是否按下
0x200E	关节运动输入参数错误	MoveJ运动中目标位置不正确	关节参数正解计算的空间位置超出了Delta的工作空间，检查参数，调整对应的关节位置
0x200F	机器人未回零	使用增量编码器时机器人未进行回零操作。	对于相对编码器，先进行回零操作
0x2010	机器人半径方向越界	机器人末端X、Y合成半径大于设定的半径	在直角坐标系下，jog使机器人末端X、Y合成半径减小的方向运动
0x2011	机器人Z正方向越界	机器人末端Z大于设定值	在直角坐标系下，jog使机器人末端朝Z负方向运动
0x2012	机器人Z负方向越界	机器人末端Z小于设定值	在直角坐标系下，jog使机器人末端朝Z正方向运动
0x2013	机器人越界	在线运行时，示教点越界	更改示教点，使示教点处于机器人工作空间内
0x2016	码垛机器人的2、3轴夹角太小	码垛机器人的2、3轴夹角太小	示教模式下，正向转动第3轴，或负向转动第2轴
0x2017	码垛机器人的2、3轴夹角太大	码垛机器人的2、3轴夹角太大	示教模式下，负向转动第3轴，或正向转动第2轴
0x2018	机器人速度异常	机器人关节速度超出了允许的最大速度	降低直线速度
0x2019	运动参数错误	运动规划参数异常	检查运动参数的设置是否合理
0x201A	机器人位置速度或姿态速度超过设定值	机器人末端运动超出了设定的位置速度或姿态速度	1. 如果使用了MoveJ指令，请将MoveJ速度系数设小 2. 检查设置的姿态速度与J4关节速度是否一致

0x2021	机器人超出工作范围上界	在跟随过程中超出了设定的工作范围	根据传输带参数设置说明，调整上界位置
0x2022	机器人超出工作范围下界	在跟随过程中超出了设定的工作范围	根据传输带参数设置说明，调整下界位置
0x2023	传送带速度过大	传送带速度超出合理范围	传输带速度超出最大速度限制（线性传输带最大 1m/s，旋转传输盘最大 180 度/s）
0x2024	传送带速度波动过大	传送带速度波动	检查传输带电机是否存在速度波动过大或者传输带有异常
0x2025	视觉数据等待超时	发送视觉触发信号后长期未收到返回数据，视觉处理周期大于拍照时间间隔	检查视觉处理一次的时间是否大于两次视觉出发的时间间隔
0x2026	机器人坐标类型错误	跟随指令中使用了静态坐标或普通运动中使用了动态坐标。	检查RefSys 间的运动指令点类型，点坐标系应为 7
0x2027	动态点坐标错误	给定的动态目标位置错误、奇异或出界	检查传输带的参数设置是否合适
0x2028	传送带跟随命令语法错误	连续使用了 RefConveyor 或 RefBase	检查是否 RefConveyor 和 RefBase 配对使用
0x2029	建立抓取工件坐标系失败	没执行 GetCnvObject 就执行了 RefConveyor。	先调用 GetCnvObject 指令
0x202A	传送带视觉端口错误	使用了多个视觉传送带	检查是否同时使用了 2 个以上视觉输入
0x202B	不允许单步示教	传送带给随相关的指令不允许单步示教	不允许对 RefConveyor 和 RefBase 之间的指令进行单步操作
0x202C	指令中使用了未使能的传送带	指令中使用了未使能的传送带	检查使用的传输带编号，是否未使能
0x202D	跟随过程中不允许 PTP 运动	跟随工艺中使用了 MoveJ 等关节运动	检查 RefConveyor 和 RefBase 之间是否使用了 ptp 或者 jump 指令（注：delta 中可使用 jump）
0x202E	传送带速度方向错误	检测到传送带速度为负值	1. 检查传输带界面中的编码器分辨率值的正负号是否正确 2. 检查传输带是否有打滑现场

0x202F	直角示教启动的位置在奇异 无法逆解的位置	直角示教启动的位置在奇异位置，无 无法逆解	关节移除奇异位置
0x2041	规划轨迹中有触发限位	规划的轨迹有限位	检查对应段的规划轨 迹，中间过程点是否有 触发限位
0x2042	规划轨迹中有进入奇异位置	规划轨迹有奇异位置	检查对应段的规划轨 迹，中间过程点是否有 触发奇异位置
0x0243	规划轨迹中有无法逆解的 位置	规划轨迹中间点无法逆解计算	修改出错段的位置取 点
0x2044	动态跟随预处理位置超出 工作下界	动态跟随预处理时检测到目标点超 出工作下界	1、检查接收到的视觉数 据点是否在合理范围内 2、检查给定的运动指令 动态坐标是否超出工作 下界
0x2101	轴 1 正限位报警	到达关节极限位置	往关节的反方向运动， 若非关节模式，先切换 至关节模式
0x2102	轴 1 负限位报警	到达关节极限位置	往关节的反方向运动， 若非关节模式，先切换 至关节模式
0x2103	轴 1 驱动报警	驱动器出现报警	根据驱动器功能码，做 相应的故障排除
0x2104	轴 1 规划溢出报警	规划值超出了最大计算范围 (-1073741823~1073741824)	检查绝对原点位置是否 选择在靠近计数极限边 沿位置，若是，则在原 点位置时，将驱动器位 置清圈数
0x2105	轴 1 跟随误差过大报警	规划位置 and 实际位置差过大	调整伺服参数，将相应 滞后减小
0x2106	轴 1 速度过大报警	运行速度大于设定的最大速度	降低笛卡尔空间的最大 速度
0x2107	轴 1 的驱动强电未上	强电未上	检查驱动器电路，是否 强电未上
0x2111	轴 2 正限位报警	同轴 1	同轴 1
0x2112	轴 2 负限位报警		
0x2113	轴 2 驱动报警		
0x2114	轴 2 规划溢出报警		
0x2115	轴 2 跟随误差过大报警		
0x2116	轴 2 速度过大报警		
0x2117	轴 2 的驱动强电未上		
0x2201	轴 3 正限位报警	同轴 1	同轴 1
0x2202	轴 3 负限位报警		

0x2203	轴 3 驱动报警		
0x2204	轴 3 规划溢出报警		
0x2205	轴 3 跟随误差过大报警		
0x2206	轴 3 速度过大报警		
0x2207	轴 3 的驱动强电未上		
0x2211	轴 4 正限位报警	同轴 1	同轴 1
0x2212	轴 4 负限位报警		
0x2213	轴 4 驱动报警		
0x2214	轴 4 规划溢出报警		
0x2215	轴 4 跟随误差过大报警		
0x2216	轴 4 速度过大报警		
0x2217	轴 4 的驱动强电未上		
0x2301	轴 5 正限位报警	同轴 1	同轴 1
0x2302	轴 5 负限位报警		
0x2303	轴 5 驱动报警		
0x2304	轴 5 规划溢出报警		
0x2305	轴 5 跟随误差过大报警		
0x2306	轴 5 速度过大报警		
0x2307	轴 5 的驱动强电未上		
0x2311	轴 6 正限位报警	同轴 1	同轴 1
0x2312	轴 6 负限位报警		
0x2313	轴 6 驱动报警		
0x2314	轴 6 规划溢出报警		
0x2315	轴 6 跟随误差过大报警		
0x2316	轴 6 速度过大报警		
0x2317	轴 6 的驱动强电未上		
0x8001	无网络设备错误	ECAT 通信初始化错误	检查 FPGA 及 DSP 固件加载是否成功
0x8002	无主站错误		
0x8003	无效域错误		
0x8004	无此从站错误		
0x8005	无效过程数据错误		
0x8006	无效服务数据错误		
0x8007	无效入口对象错误		
0x8008	域内存地址分配错误		
0x8009	激活主站失败错误		
0x800A	服务数据公共错误		
0x800B	注册周期回调错误		
0x800C	过程通信配置错误		
0x800D	初始化模块错误		
0x800E	解析配置错误		
0x800F	配置 DSP 通道参数错误		
0x8010	域注册错误		
0x8011	创建定时器错误		

0x8012	启动定时器错误		
0x8013	配置 ECAT 通信周期错误	ECAT 从站配置错误	检查 ECAT 配置信息 确保配置正确保存后重 启
0x8014	配置 ECAT 版本选择错误		
0x8015	配置 ECAT 伺服从站数错误		
0x8016	配置 ECAT IO 从站数错误		
0x8017	配置 ECAT IO 模块数错误		
0x8018	配置 ECAT IO 类型错误		
0x8019	配置 ECAT IO 不支持错误		
0x801A	配置 ECAT 内存申请错误		
0x801B	配置 ECAT 报警共享内存错 误		
0x801C	配置 ECAT 伺服操作模式错 误		
0x801D	配置 ECAT 寄存器错误		
0x801E	配置 ECAT IO 数量与在线 IO 数量不匹配错误		
0x801F	配置 ECAT 伺服数量与在线 伺服数量不匹配错误		
0x8020	配置 ECAT 伺服供应商代码 不支持错误		
0x8028	写缓冲区错误	ECAT 通信运行错误	检查 ECAT 网络连接情况 检查 ECAT 从站是否有掉 电等
0x8029	写启动命令错误		
0x802A	读状态寄存器错误		
0x802B	读数据链路状态错误		
0x802C	读服务数据通道错误		
0x802D	读服务数据长度错误		
0x802E	服务数据长度错误		
0x802F	服务数据接收错误		
0x8030	服务数据通道忙错误		
0x8031	服务数据报文错误		
0x8032	读过程数据通信错误		
0x8033	读过程数据长度错误		
0x8034	过程数据长度错误		
0x8035	过程数据接收错误		
0x8036	网络设备打开错误		
0x8037	GPMC IOCTRL 错误		
0x8038	GPMC ECAT 读错误		
0x8039	GPMC ECAT 写错误		
0x803A	读发送时间戳错误		
0x803B	读接收时间戳错误		
0x803C	读过程数据剩余错误		
0x803D	读应用时间戳错误		

0x803E	现场总线 LED 打开错误		
0x803F	现场总线 LED IOCTRL 错误		
0x8040	ARM 看门狗错误		
0x8041	DSP 看门狗错误		
0x8042	ECAT 从站掉线错误		确保ECAT从站网线已正确连接后 掉电重启
0x8N42	ECAT 从站 N 掉线错误		
0x8043	接入了非 ECAT 从站设备错误		将 ECAT 网线插入 ECAT 网口
0x8044	ECAT 网口 1 未连接错误		
0x8045	ECAT 网口 2 未连接错误		
0x8046	设置 ECAT 启动时间错误	ECAT 周期通信启动错误	检查从站状态是否异常
0x805C	翻转位未改变错误		
0x805D	SDO 协议超时错误		
0x805E	命令无效或未知错误		
0x805F	对象不可被访问错误		
0x8060	试图读一个只写对象错误		
0x8061	试图写一个只读对象错误		
0x8062	对象不存在对象字典中错误		
0x8063	对象不能被映射为过程数据错误		
0x8064	被映射的对象长度超出过程数据长度错误		
0x8065	基本参数不兼容错误		
0x8066	设备内部不兼容错误		
0x8067	硬件导致访问失败错误		
0x8068	服务参数长度不匹配错误	ECAT 从站服务数据通信错误	检查服务数据请求 如何服功能码操作数据格式等
0x8069	服务参数长度太长错误		
0x806A	服务参数长度太短错误		
0x806B	子索引不存在错误		
0x806C	参数值越界错误		
0x806D	所写参数值太大错误		
0x806E	所写参数值太小错误		
0x806F	最大值小于最小值错误		
0x8070	数据无法传输或存储错误		
0x8071	本地控制导致数据无法存储错误		
0x8072	设备状态导致数据无法存储错误		
0x8073	对象字典动态总错误或对象字典不存在错误		

0x807A	配置 IR-LINK 通信周期错误	IR-LINK 从站配置错误	检查 IR-LINK 配置信息 确保配置正确保存后重 启
0x807B	配置 IR-LINK 版本选择错误		
0x807C	配置 IR-LINK 伺服从站数错误		
0x807D	配置 IR-LINK 从站数错误		
0x807E	配置 IR-LINK 模块数错误		
0x807F	配置 IR-LINK 类型错误		
0x8080	配置 IR-LINK 不支持错误		
0x8081	配置 IR-LINK 内存申请错误		
0x8082	配置 IR-LINK 报警共享内存错 误		
0x8083	配置 IR-LINK 伺服操作模式错 误		
0x8084	配置 IR-LINK 寄存器错误		
0x8085	配置 IR-LINK IO 数量与在线 IO 数量不匹配错误		
0x8086	配置 IR-LINK 从站数量与在线 从站数量不匹配错误		
0x8087	配置 IR-LINK 伺服供应商代码 不支持错误	IR-LINK 运行错误	检查 IR-LINK 网络连接 情况 检查 IR-LINK 从站是否 掉电等
0x8090	写缓冲区错误		
0x8091	读服务数据通道错误		
0x8092	读服务数据长度错误		
0x8093	服务数据长度错误		
0x8094	服务数据接收错误		
0x8095	服务数据通道忙错误		
0x8096	服务数据报文错误		
0x8097	读过程数据通信错误		
0x8098	读过程数据长度错误		
0x8099	过程数据长度错误		
0x809A	过程数据接收错误		
0x809B	网络设备打开错误		
0x809C	GPMC IR-LINK 读错误		
0x809D	GPMC IR-LINK 写错误		
0x809E	IR-LINK 从站掉线错误	确保 IR-LINK 从站网线 已正确连接后 掉电重启	
0x8N9E	IR-LINK 从站 N 掉线错误		
0x809F	接入了非 IR-LINK 从站设备错 误	IR-LINK 网口未连接	将 IR-LINK 网线插入 IR-LINK 网口
0x80A0	IR-LINK 网口 0 未连接错误		
0x80A1	IR-LINK 网口 1 未连接错误		

0x80A2	设置 IR—Link 启动时间错误	IR—Link 周期信启动错误	检查从站状态是否异常
0xE001	FPGA 运行正常	正常运行告警	无需处理
0xE002	ARM 从 SD 卡加载 FPGA 固件文件失败	从 sd 卡的第一个分区读取 FPGA_FW.bin 固件异常, 可能原因是没有这个文件或者该文件被破坏	检查 SD 卡是否存在 FPGA_FW.bin 固件
0xE003	ARM 从 SPI flash 加载 FPGA 固件文件失败	从 SPI flash robotfw 分区读取 FPGA_FW.bin 固件异常, 可能原因是没有这个文件或者该文件被破坏	检查 SPI flash robotfw 是否存在 FPGA_FW.bin 固件
0xE004	ARM 复位 FPGA 失败	ARM 给 FPGA 芯片复位信号没有收到应答	需要检查 FPGA 与 ARM 通信管脚配置
0xE005	传输 FPGA 数据过程失败	ARM 给 FPGA 芯片发送数据异常	需要检查 FPGA 与 ARM 通信管脚配置
0xE006	加载固件后与 FPGA 握手失败	ARM 给 FPGA 发送数据完毕后, 没有收到 FPGA 正常运行报告	需要检查 FPGA 自身芯片或者固件是否正确
0xE007	控制通道 1 运行正常	正常运行告警	无需处理
0xE008	控制通道 1 分配内存失败	系统软件出错	检查是软件版本, 断电重启
0xE009	ARM 从 SD 卡加载控制通道 1 固件文件失败	SD 卡上找不到控制通道 1 上所配置的固件	检查 SD 卡是否存放于系统配置控制通道 1 对应的固件
0xE00A	ARM 从 SPI flash 加载控制通道 1 固件文件失败	SPI flash robotfw 分区上找不到控制通道 1 上所配置的固件	检查 SPI flash robotfw 分区上是否存放于系统配置控制通道 1 对应的固件
0xE00B	控制通道 1 固件长度超出范围	目前假设固件最大为 1Mbyte, 该固件已经超出	检查固件是否已经超出长度
0xE00C	控制通道 1 通过 ARM 联络 FPGA 失败	FPGA 可能没有正常工作	检查 FPGA 运行灯是否正常闪烁
0xE00D	复位控制通道 1 失败	控制通道 1 可能没有将其配置从 SPI 启动固件程序	检查控制通道 1 硬件启动方式
0xE00E	控制通道 1 发 START WORD 失败	ARM 与控制通道 1 之间 spi 通信失败	检查 ARM 侧与控制通道 1 侧 spi 硬件线路
0xE00F	控制通道 1 发 PING OP WORD 失败	ARM 与控制通道 1 之间 spi 通信失败	检查 ARM 侧与控制通道 1 侧 spi 硬件线路
0xE010	控制通道 1 SPI 变速失败	ARM 与控制通道 1 之间 spi 通信失败	检查控制通道 1 和 ARM 芯片是否异常
0xE011	控制通道 1 加载数据超时	可能制作固件时配置出错或者固件被破坏	检查制作固件是否正确, 或者固件是否完整
0xE012	控制通道 1 运行应答异常	控制通道 1 初始化时间过久导致应答异常	需要更新固件
0xE013	控制通道 0 运行正常	正常运行告警	无需处理

0xE014	分配控制通道0内存失败	系统软件出错	检查是软件版本，断电重启
0xE015	ARM从SD卡加载控制通道0固件文件失败	SD卡上找不到控制通道0上所配置的固件	检查SD卡是否存放于系统配置控制通道0对应的固件
0xE016	ARM从SPI flash加载控制通道0固件文件失败	SPI flash robotfw分区上找不到控制通道0上所配置的固件	检查SPI flash robotfw分区上是否存放于系统配置控制通道0对应的固件
0xE017	控制通道0固件长度超出范围	目前假设固件最大为1Mbyte，该固件已经超出	检查固件是否已经超出长度
0xE018	控制通道0通过ARM联络FPGA失败	FPGA可能没有正常工作	检查FPGA运行灯是否正常闪烁
0xE019	复位控制通道0失败	控制通道0可能没有将其配置从SPI启动固件程序	检查控制通道0硬件启动方式
0xE01A	控制通道0发START WORD失败	ARM与控制通道0之间spi通信失败	检查ARM侧与控制通道0侧spi硬件线路
0xE01B	控制通道0发PING OP WORD失败	ARM与控制通道0之间spi通信失败	检查ARM侧与控制通道0侧spi硬件线路
0xE01C	控制通道0SPI变速失败	ARM与控制通道0之间spi通信失败	检查控制通道0和ARM芯片是否异常
0xE01D	控制通道0加载数据超时	可能制作固件时配置出错或者固件被破坏	检查制作固件是否正确，或者固件是否完整
0xE01E	控制通道0运行应答异常	控制通道0初始化时间过久导致应答异常	需要更新固件

附录二：Modbus从站地址表

主站通讯属性		地址	地址	变量名称	数据类型	内容	示教器编程	二次开发编程
		(10进制)	(16进制)					
比特访问	只读(4096个)物理离散量输入, 功能码: 0x02	0	0x0000	QW65024,bit 0	位	使能	SetModbus Coil-不可用 GetModbus Coil-可用	禁止写地址位允许
		1	0x0001	QW65024,bit 1	位	启动		
		2	0x0002	QW65024,bit 2	位	急停		
		3	0x0003	QW65024,bit 3	位	故障		

读地址位

4	0x0004	QW65024,bit 4	位	伺服故障
5	0x0005	QW65024,bit 5	位	-
6	0x0006	QW65024,bit 6	位	-
7	0x0007	QW65024,bit 7	位	-
8	0x0008	QW65024,bit 8	位	-
9	0x0009	QW65024,bit 9	位	-
10	0x000a	QW65024,bit 10	位	-
11	0x000b	QW65024,bit 11	位	-
12	0x000c	QW65024,bit 12	位	-
13	0x000d	QW65024,bit 13	位	-
14	0x000e	QW65024,bit 14	位	-
15	0x000f	QW65024,bit 15	位	-
16	0x0010	QW65025,bit 0	位	-
...		位	-
63	0x003F	QW65027,bit 15	位	-
64	0x0040	QW65028,bit 0	位	IN[000]
65	0x0041	QW65028,bit 1	位	IN[001]
66	0x0042	QW65028,bit 2	位	IN[002]
67	0x0043	QW65028,bit 3	位	IN[003]
68	0x0044	QW65028,bit 4	位	IN[004]
69	0x0045	QW65028,bit 5	位	IN[005]
70	0x0046	QW65028,bit 6	位	IN[006]

71	0x0047	QW65028,bit 7	位	IN[007]
72	0x0048	QW65028,bit 8	位	IN[008]
73	0x0049	QW65028,bit 9	位	IN[009]
74	0x004A	QW65028,bit 10	位	IN[010]
75	0x004B	QW65028,bit 11	位	IN[011]
76	0x004C	QW65028,bit 12	位	IN[012]
77	0x004D	QW65028,bit 13	位	IN[013]
78	0x004E	QW65028,bit 14	位	IN[014]
79	0x004F	QW65028,bit 15	位	IN[015]
80	0x0050	QW65029,bit 0	位	IN[016]
81	0x0051	QW65029,bit 1	位	IN[017]
82	0x0052	QW65029,bit 2	位	IN[018]
83	0x0053	QW65029,bit 3	位	IN[019]
84	0x0054	QW65029,bit 4	位	IN[020]
85	0x0055	QW65029,bit 5	位	IN[021]
86	0x0056	QW65029,bit 6	位	IN[022]
87	0x0057	QW65029,bit 7	位	IN[023]
88	0x0058	QW65029,bit 8	位	IN[024]
89	0x0059	QW65029,bit 9	位	IN[025]
90	0x005A	QW65029,bit 10	位	IN[026]
91	0x005B	QW65029,bit 11	位	IN[027]
92	0x005C	QW65029,bit 12	位	IN[028]

93	0x005D	QW65029,bit 13	位	IN[029]
94	0x005E	QW65029,bit 14	位	IN[030]
95	0x005F	QW65029,bit 15	位	IN[031]
96	0x0060	QW65030,bit 0	位	IN[032]
97	0x0061	QW65030,bit 1	位	IN[033]
98	0x0062	QW65030,bit 2	位	IN[034]
99	0x0063	QW65030,bit 3	位	IN[035]
100	0x0064	QW65030,bit 4	位	IN[036]
101	0x0065	QW65030,bit 5	位	IN[037]
102	0x0066	QW65030,bit 6	位	IN[038]
103	0x0067	QW65030,bit 7	位	IN[039]
104	0x0068	QW65030,bit 8	位	IN[040]
105	0x0069	QW65030,bit 9	位	IN[041]
106	0x006A	QW65030,bit 10	位	IN[042]
107	0x006B	QW65030,bit 11	位	IN[043]
108	0x006C	QW65030,bit 12	位	IN[044]
109	0x006D	QW65030,bit 13	位	IN[045]
110	0x006E	QW65030,bit 14	位	IN[046]
111	0x006F	QW65030,bit 15	位	IN[047]
112	0x0070	QW65031,bit 0	位	IN[048]
113	0x0071	QW65031,bit 1	位	IN[049]
114	0x0072	QW65031,bit 2	位	IN[050]

115	0x0073	QW65031,bit 3	位	IN[051]
116	0x0074	QW65031,bit 4	位	IN[052]
117	0x0075	QW65031,bit 5	位	IN[053]
118	0x0076	QW65031,bit 6	位	IN[054]
119	0x0077	QW65031,bit 7	位	IN[055]
120	0x0078	QW65031,bit 8	位	IN[056]
121	0x0079	QW65031,bit 9	位	IN[057]
122	0x007A	QW65031,bit 10	位	IN[058]
123	0x007B	QW65031,bit 11	位	IN[059]
124	0x007C	QW65031,bit 12	位	IN[060]
125	0x007D	QW65031,bit 13	位	IN[061]
126	0x007E	QW65031,bit 14	位	IN[062]
127	0x007F	QW65031,bit 15	位	IN[063]
128	0x0080	QW65032,bit 0	位	OUT[000]
129	0x0081	QW65032,bit 1	位	OUT[001]
130	0x0082	QW65032,bit 2	位	OUT[002]
131	0x0083	QW65032,bit 3	位	OUT[003]
132	0x0084	QW65032,bit 4	位	OUT[004]
133	0x0085	QW65032,bit 5	位	OUT[005]
134	0x0086	QW65032,bit 6	位	OUT[006]
135	0x0087	QW65032,bit 7	位	OUT[007]
136	0x0088	QW65032,bit 8	位	OUT[008]

137	0x0089	QW65032,bit 9	位	OUT[009]
138	0x008A	QW65032,bit 10	位	OUT[010]
139	0x008B	QW65032,bit 11	位	OUT[011]
140	0x008C	QW65032,bit 12	位	OUT[012]
141	0x008D	QW65032,bit 13	位	OUT[013]
142	0x008E	QW65032,bit 14	位	OUT[014]
143	0x008F	QW65032,bit 15	位	OUT[015]
144	0x0090	QW65033,bit 0	位	OUT[016]
145	0x0091	QW65033,bit 1	位	OUT[017]
146	0x0092	QW65033,bit 2	位	OUT[018]
147	0x0093	QW65033,bit 3	位	OUT[019]
148	0x0094	QW65033,bit 4	位	OUT[020]
149	0x0095	QW65033,bit 5	位	OUT[021]
150	0x0096	QW65033,bit 6	位	OUT[022]
151	0x0097	QW65033,bit 7	位	OUT[023]
152	0x0098	QW65033,bit 8	位	OUT[024]
153	0x0099	QW65033,bit 9	位	OUT[025]
154	0x009A	QW65033,bit 10	位	OUT[026]
155	0x009B	QW65033,bit 11	位	OUT[027]
156	0x009C	QW65033,bit 12	位	OUT[028]
157	0x009D	QW65033,bit 13	位	OUT[029]
158	0x009E	QW65033,bit 14	位	OUT[030]

159	0x009F	QW65033,bit 15	位	OUT[031]
160	0x00A0	QW65034,bit 0	位	OUT[032]
161	0x00A1	QW65034,bit 1	位	OUT[033]
162	0x00A2	QW65034,bit 2	位	OUT[034]
163	0x00A3	QW65034,bit 3	位	OUT[035]
164	0x00A4	QW65034,bit 4	位	OUT[036]
165	0x00A5	QW65034,bit 5	位	OUT[037]
166	0x00A6	QW65034,bit 6	位	OUT[038]
167	0x00A7	QW65034,bit 7	位	OUT[039]
168	0x00A8	QW65034,bit 8	位	OUT[040]
169	0x00A9	QW65034,bit 9	位	OUT[041]
170	0x00AA	QW65034,bit 10	位	OUT[042]
171	0x00AB	QW65034,bit 11	位	OUT[043]
172	0x00AC	QW65034,bit 12	位	OUT[044]
173	0x00AD	QW65034,bit 13	位	OUT[045]
174	0x00AE	QW65034,bit 14	位	OUT[046]
175	0x00AF	QW65034,bit 15	位	OUT[047]
176	0x00B0	QW65035,bit 0	位	OUT[048]
177	0x00B1	QW65035,bit 1	位	OUT[049]
178	0x00B2	QW65035,bit 2	位	OUT[050]
179	0x00B3	QW65035,bit 3	位	OUT[051]
180	0x00B4	QW65035,bit 4	位	OUT[052]

181	0x00B5	QW65035,bit 5	位	OUT[053]
182	0x00B6	QW65035,bit 6	位	OUT[054]
183	0x00B7	QW65035,bit 7	位	OUT[055]
184	0x00B8	QW65035,bit 8	位	OUT[056]
185	0x00B9	QW65035,bit 9	位	OUT[057]
186	0x00BA	QW65035,bit 10	位	OUT[058]
187	0x00BB	QW65035,bit 11	位	OUT[059]
188	0x00BC	QW65035,bit 12	位	OUT[060]
189	0x00BD	QW65035,bit 13	位	OUT[061]
190	0x00BE	QW65035,bit 14	位	OUT[062]
191	0x00BF	QW65035,bit 15	位	OUT[063]
192	0x00C0	QW65036,bit 0	位	J1 伺服告 警
193	0x00C1	QW65036,bit 1	位	J2 伺服告 警
194	0x00C2	QW65036,bit 2	位	J3 伺服告 警
195	0x00C3	QW65036,bit 3	位	J4 伺服告 警
196	0x00C4	QW65036,bit 4	位	J5 伺服告 警
197	0x00C5	QW65036,bit 5	位	J6 伺服告 警
198	0x00C6	QW65036,bit 6	位	J7 伺服告 警
199	0x00C7	QW65036,bit 7	位	J8 伺服告 警
200	0x00C8	QW65036,bit 8	位	J9 伺服告 警（预留）
201	0x00C9	QW65036,bit 9	位	J10 伺服 告警（预 留）

	202	0x00CA	QW65036,bit 10	位	J11 伺服 告警（预 留）		
	203	0x00CB	QW65036,bit 11	位	J12 伺服 告警（预 留）		
	204	0x00CC	QW65036,bit 12	位	J13 伺服 告警（预 留）		
	205	0x00CD	QW65036,bit 13	位	J14 伺服 告警（预 留）		
	206	0x00CE	QW65036,bit 14	位	J15 伺服 告警（预 留）		
	207	0x00CF	QW65036,bit 15	位	J16 伺服 告警（预 留）		
	位	-		
	2047	0x07FF	QW65151,bit 15	位	-		
	2048	0x0800	QW65152,bit 0	位	（示例： 取料）	SetModbusCoil- 可用 GetModbusCoil- 可用	允许 写 地 址 位 允 许 读 地 址 位
	2049	0x0801	QW65152,bit 1	位	（示例： 放料）		
	2050	0x0802	QW65152,bit 2	位	-		
	2051	0x0803	QW65152,bit 3	位	-		
	2052	0x0804	QW65152,bit 4	位	-		
	2053	0x0805	QW65152,bit 5	位	-		
	位	-		
	4095	0x0fff	QW65279,bit 15	位	-		
读写(4096个)线 圈功能码： 0x01, 0x05, 0x0f	4096	0x1000	QW65280,bit 0	位	运行	SetModbusCoil- 可用	允许 写 地
	4097	0x1001	QW65280,bit 1	位	暂停	GetModbusCoil- 可用	

4098	0x1002	QW65280,bit 2	位	停止
4099	0x1003	QW65280,bit 3	位	使能
4100	0x1004	QW65280,bit 4	位	急停
4101	0x1005	QW65280,bit 5	位	故障复位
4102	0x1006	QW65280,bit 6	位	示教 J1/X + (预留)
4103	0x1007	QW65280,bit 7	位	示教 J1/Y + (预留)
4104	0x1008	QW65280,bit 8	位	示教 J3/Z + (预留)
4105	0x1009	QW65280,bit 9	位	示教 J4/A + (预留)
4106	0x100a	QW65280,bit 10	位	示教 J5 + (预留)
4107	0x100b	QW65280,bit 11	位	示教 J6 + (预留)
4108	0x100c	QW65280,bit 12	位	示教 J7 + (预留)
4109	0x100d	QW65280,bit 13	位	示教 J8 + (预留)
4110	0x100e	QW65280,bit 14	位	示教 J9 + (预留)
4111	0x100f	QW65280,bit 15	位	示教 J10 + (预留)
4112	0x1010	QW65281,bit 0	位	示教 J11 + (预留)
4113	0x1011	QW65281,bit 1	位	示教 J12 + (预留)
4114	0x1012	QW65281,bit 2	位	示教 J13 + (预留)
4115	0x1013	QW65281,bit 3	位	示教 J14 + (预留)
4116	0x1014	QW65281,bit 4	位	示教 J15 + (预留)
4117	0x1015	QW65281,bit 5	位	示教 J16 + (预留)
4118	0x1016	QW65281,bit 6	位	示教 J1/X - (预留)
4119	0x1017	QW65281,bit 7	位	示教 J2/Y - (预留)

址
位
允
许
读
地
址
位

4120	0x1018	QW65281,bit 8	位	示教 J3/Z - (预留)		
4121	0x1019	QW65281,bit 9	位	示教 J4/A - (预留)		
4122	0x101a	QW65281,bit 10	位	示教 J5 - (预留)		
4123	0x101b	QW65281,bit 11	位	示教 J6 - (预留)		
4124	0x101c	QW65281,bit 12	位	示教 J7 - (预留)		
4125	0x101d	QW65281,bit 13	位	示教 J8 - (预留)		
4126	0x101e	QW65281,bit 14	位	示教 J9 - (预留)		
4127	0x101f	QW65281,bit 15	位	示教 J10 - (预留)		
4128	0x1020	QW65282,bit 0	位	示教 J11 - (预留)		
4129	0x1021	QW65282,bit 1	位	示教 J12 - (预留)		
4130	0x1022	QW65282,bit 2	位	示教 J13 - (预留)		
4131	0x1023	QW65282,bit 3	位	示教 J14- (预留)		
4132	0x1024	QW65282,bit 4	位	示教 J15 - (预留)		
4133	0x1025	QW65282,bit 5	位	示教 J16 - (预留)		
4134	0x1026	QW65282,bit 6	位	工位程序 1		
4135	0x1027	QW65282,bit 7	位	工位程序 2		
4136	0x1028	QW65282,bit 8	位	工位程序 3		
...		位	-		
6143	0x17ff	QW65407,bit 15	位	-		
6144	0x1800	QW65408,bit 0	位	-	SetModbusCoil- 可用 GetModbusCoil- 可用	允 许 写 地 址 位
6145	0x1801	QW65408,bit 1	位	-		
6146	0x1802	QW65408,bit 2	位	-		

		6147	0x1803	QW65408,bit 3	位	-			允许 读 地 址 位
		6148	0x1804	QW65408,bit 4	位	-			
		位	-			
		8191	0x1fff	QW65535,bit 15	位	-			
16 比 特 访 问	只读(32768)输入 寄存器功能码: 0x04	0	0x0	MW0	字	预留系统 其他用途 使用 (2048 字)	SetModbusReg- 不可用 GetModbusReg- 可用	禁止 写 地 址 允 许 读 地 址	
		字				
		2047	0x07ff	MW2047	字				
		2048	0x0800	MW2048	字	当前坐标 系			
		2049	0x0801	MW2049	字	当前速度			
		2050	0x0802	MW2050	字	故障记录			
		2051	0x0803	MW2051	字				
		2052	0x0804	MW2052	单 精 度 浮 点	J1/X 坐标 低位			
		2053	0x0805	MW2053		J2/X 坐标 高位			
		2054	0x0806	MW2054	单 精 度 浮 点	J2/Y 坐标 低位			
		2055	0x0807	MW2055		J2/Y 坐标 高位			
		2056	0x0808	MW2056	单 精 度 浮 点	J3/Z 坐标 低位			
		2057	0x0809	MW2057		J3/Z 坐标 高位			
		2058	0x080a	MW2058	单 精 度 浮 点	J4/A 坐 标低位			
		2059	0x080b	MW2059		J4/A 坐 标高位			
		2060	0x080c	MW2060	单 精 度 浮 点	J5/B 坐标 低位			
2061	0x080d	MW2061		J5/B 坐标 高位					
2062	0x080e	MW2062	单 精	J6/C 坐标 低位					

2063	0x080f	MW2063	度 浮 点	J6/C 坐标 高位
2064	0x0810	MW2064	单 精 度	J7 坐标 低位（预 留）
2065	0x0811	MW2065	浮 点	J7 坐标 高位（预 留）
2066	0x0812	MW2066	单 精 度	J8 坐标 低位（预 留）
2067	0x0813	MW2067	浮 点	J8 坐标 高位（预 留）
2068	0x0814	MW2068	单 精 度	J9 坐标 低位（预 留）
2069	0x0815	MW2069	浮 点	J9 坐标 高位（预 留）
2070	0x0816	MW2070	单 精 度	J10 坐标 低位（预 留）
2071	0x0817	MW2071	浮 点	J10 坐标 高位（预 留）
2072	0x0818	MW2072	单 精 度	J11 坐标 低位（预 留）
2073	0x0819	MW2073	浮 点	J11 坐标 高位（预 留）
2074	0x081a	MW2074	单 精 度	J12 坐标 低位（预 留）
2075	0x081b	MW2075	浮 点	J12 坐标 高位（预 留）
2076	0x081c	MW2076	单 精 度	J13 坐标 低位（预 留）
2077	0x081d	MW2077	浮 点	J13 坐标 高位（预

				留)
2078	0x081e	MW2078	单精度	J14 坐标低位 (预留)
2079	0x081f	MW2079	浮点	J14 坐标高位 (预留)
2080	0x0820	MW2080	单精度	J15 坐标低位 (预留)
2081	0x0821	MW2081	浮点	J15 坐标高位 (预留)
2082	0x0822	MW2082	单精度	J16 坐标低位 (预留)
2083	0x0823	MW2083	浮点	J16 坐标高位 (预留)
...		字	-
2116	0x0844	MW2116	字 (无符号)	J1 伺服告警码
2117	0x0845	MW2117	字 (无符号)	J2 伺服告警码
2118	0x0846	MW2118	字 (无符号)	J3 伺服告警码
2119	0x0847	MW2119	字 (无符号)	J4 伺服告警码
2120	0x0848	MW2120	字 (无符号)	J5 伺服告警码
2121	0x0849	MW2121	字 (无符号)	J6 伺服告警码

2122	0x084A	MW2122	字 (无 符 号)	J7 伺服告 警码		
2123	0x084B	MW2123	字 (无 符 号)	J8 伺服告 警码		
2124	0x084C	MW2124	字 (无 符 号)	J9 伺服告 警码 (预 留)		
2125	0x084D	MW2125	字 (无 符 号)	J10 伺服 告 警 码 (预留)		
2126	0x084E	MW2126	字 (无 符 号)	J11 伺服 告 警 码 (预留)		
2127	0x084F	MW2127	字 (无 符 号)	J12 伺服 告 警 码 (预留)		
2128	0x0850	MW2128	字 (无 符 号)	J13 伺服 告 警 码 (预留)		
2129	0x0851	MW2129	字 (无 符 号)	J14 伺服 告 警 码 (预留)		
2130	0x0852	MW2130	字 (无 符 号)	J15 伺服 告 警 码 (预留)		
2131	0x0853	MW2131	字 (无 符 号)	J16 伺服 告 警 码 (预留)		
...		字	-		
16383	0x3fff	MW16383	字	-		
16384	0x4000	MW16384	整	(示例:	SetModbusReg-	允

				型	产 能 低 位)	可用 GetModbusReg-	许 写 地 址 允 许 读 地 址	
	16385	0x4001	MW16385		(示 例 : 产 能 高 位)	可用		
	16386	0x4002	MW16386	整 型	(示 例 : 不 良 品 低 位)			
	16387	0x4003	MW16387		(示 例 : 不 良 品 高 位)			
	16388	0x4004	MW16388	单 精 度 浮 点	(示 例 : 不 良 率 低 位)			
	16389	0x4005	MW16389		(示 例 : 不 良 率 高 位)			
	16390	0x4006	MW16390	字	-			
	16391	0x4007	MW16391	字	-			
	16392	0x4008	MW16392	字	-			
	字	-			
	32767	0x7fff	MW32767	字	-			
读写(32768)保存 寄存器,功能码: 0x03, 0x10	32768	0x8000	MW32768	字	预留系统 其他用途 使用 (2048 字)	SetModbusReg- 不可用 GetModbusReg- 可用		允 许 写 地 址 允 许 读 地 址
	32769	0x8001	MW32769	字				
	字				
	34815	0x87ff	MW34815	字	坐标系选 择			
	34816	0x8800	MW34816	字				
	34817	0x8801	MW34817	字	速度设置			
	34818	0x8802	MW34818	字	-			
	34819	0x8803	MW34819	字	-			
	字	-			
	49151	0xbfff	MW49151	字	-	SetModbusReg- 可用 GetModbusReg- 可用		
	49152	0xc000	MW49152	字	-			
	49153	0xc001	MW49153	字	-			
	49154	0xc002	MW49154	字	-			
...		字	-				
65535	0xffff	MW65535	字	-				

附录三：API 指令

序号	函数名称	说明	参数	返回值	指令类型	备注
1	int Init_ETH (DWORD IpAddr, int IpPort, int timeOut=5, int comId=0)	建立机器人网络连接	IpAddr: 机器人控制器网络 IP 地址 IpPort: 机器人控制器网络端口号(固定为 2222) timeOut: 通讯超时时间设置, 默认 5s comId: 连接编号, 标记相同目标 IP 和端口号下不同的连接, 默认值 0, 最大值 4 (下同)	返回 0 表示连接成功, 小于 0 表示失败	立即指令 (无连接时阻塞 2s)	
2	int Exit_ETH (int comId=0)	关闭机器人网络连接	comId: 连接编号, 标记对应连接 (下面该参数不再赘述)	返回 0 表示关闭成功, 小于 0 表示失败		
3						
4	int EmergStop (int cmd, int comId=0)	急停开关控制	cmd: 急停命令, 1-按下急停, 0-松开急停	返回 0 表示急停命令完成, 小于 0 表示失败		
5	int MotorEnable (int cmd, int comId=0)	电机使能控制	cmd: 电机使能命令, 1-使能, 0-去使能	返回 0 表示电机使能命令完成, 小于 0 表示失败		
6	int ResetErr (int comId=0)	故障复位		返回 0 表示故障复位命令完成, 小于 0 表示失败	延时指令 (约 50ms)	
7	int Set_Mode (int mode, int comId=0)	设置系统运行模式	mode: 模式, 1-示教, 2-再现, 3-单步运行, 5-连续运行	返回 0 表示模式设置成功, 小于 0 表示失败		
8	int Set_CurPrgPath (char prgPath[128], int comId=0)	设置当前示教程序路径	prgPath[]: 程序路径, 例如 TeachProgram/a. pro	返回 0 表示路径设置成功, 小于 0 表示失败		
9	int PrgCtrl (int cmd, int comId=0)	示教程序运行控制	cmd: 控制命令, 0-停止, 1-启动/继续, 2-前进, 3-后退, 4-暂停	返回 0 表示示教程序控制成功, 小于 0 表示失败		

10	int Set_Vel (int val, int comId=0)	设置当前运行速度等级	val: 当前速度等级, 范围1-100	返回0表示设置速度成功, 小于0表示失败		
11	int DsMode (int cmd, int comId=0)	数据流模式控制	cmd: 数据流命令, 0-关闭, 1-开启	返回0表示数据流模式控制完成, 小于0表示失败		
12	int Set_DO (int num, int status, int comId=0)	按位设置DO的输出状态 (对象为RC拥有控制权的DO)	num: DO位序号 status: DO状态, 0-off, 1-On	返回0表示设置成功, 小于0表示失败		
13	int Set_DOGroup (int num, int status, int comId=0)	按组设置DO的输出状态	num: DO组序号, 范围1-7 status: 每组中DO状态, 范围0-255, 其中bit0-bit7分别对应每组序号最小至最大的DO状态	返回0表示设置成功, 小于0表示失败		
14	int Set_DA (int num, float val, int comId=0)	按序号设置DA的输出值	num: DA序号, 最大范围0-15 val: DA值, 电流型最大范围0-20mA, 电压型最大范围-10V到10V, 具体依据DA通道类型	返回0表示设置成功, 小于0表示失败		
15	保留					
16	保留					
17	int InchMode (int cmd, int comId=0)	寸动示教模式控制	cmd: 寸动示教模式命令, 0-关闭, 1-开启	返回0表示寸动示教模式控制完成, 小于0表示失败		
18	int Set_InchStep (int val, int comId=0)	设置寸动示教运行的步长等级	val: 步长等级值, 范围1-4, 其中1表示步长为0.5, 2表示步长为2, 3表示步长为5, 4表示步长为寸动参数设置值, 关节坐标寸动时单位为度, 基坐标寸动时单位为mm	返回0表示设置成功, 小于0表示失败		
19	保留					

20	int Jog (int mode, int axis, int cmd, int comId=0)	连续示教运动命令	mode: 示教模式, 0-关节示教, 1-基坐标示教 axis: 轴序号, 范围 1-6, 关节示教时分别对应 J1-J6 轴, 基坐标示教时分别对应 X/Y/Z/RZ/RX/RX cmd: 示教命令, 0-停止, 1-正向示教, -1-反向示教	返回 0 表示命令发送成功, 小于 0 表示失败		
21	int Inch (int mode, int axis, int cmd, int comId=0)	寸动示教运动命令	mode: 寸动模式, 0-关节寸动, 1-基坐标寸动 axis: 轴序号, 范围 1-6, 关节寸动时分别对应 J1-J6 轴, 基坐标寸动时分别对应 X/Y/Z/RZ/RX/RX cmd: 寸动命令, 1-正向寸动, -1-反向寸动	返回 0 表示命令发送成功, 小于 0 表示失败		
22	int Home (int num, int comId=0)	回原点运动命令	num: 原点序号, 范围 0-4	返回 0 表示命令发送成功, 小于 0 表示失败		
23	int MovJ (int posNum, int vel=100, int zone=0, int comId=0)	关节插补方式运动到指定序号的位置点	posNum: 目标位置点序号 vel: 运动速度, 范围 1-100, 默认为 100 zone: 插补精度, 范围 0-5, 默认为 0	返回 0 表示命令发送成功, 小于 0 表示失败		
24	int MovL (int posNum, int vel=100, int zone=0, int comId=0)	直线插补方式运动到指定序号的位置点	posNum: 目标位置点序号 vel: 运动速度, 范围 1-100, 默认为 100 zone: 插补精度, 范围 0-5, 默认为 0	返回 0 表示命令发送成功, 小于 0 表示失败		
25	int MovJ2 (ROBOT_POS pos, int vel=100, int zone=0, int comId=0)	关节插补方式运动到指定值的位置点	pos: 位置参数结构体 vel: 运动速度, 范围 1-100, 默认为 100 zone: 插补精度, 范围 0-5, 默认为 0	返回 0 表示命令发送成功, 小于 0 表示失败		
26	int MovL2 (ROBOT_POS pos, int vel=100, int zone=0, int comId=0)	直线插补方式运动到指定值的位置点	pos: 位置参数结构体 vel: 运动速度, 范围 1-100, 默认为 100 zone: 插补精度, 范围 0-5, 默认为 0	返回 0 表示命令发送成功, 小于 0 表示失败		
27						
28	int Get_PosHere (ROBOT_POS &pos, int comId=0)	查询当前位置点的位置参数(与当	pos: 位置参数结构体, 代表查询的结果	返回 0 表示查询成功, 小于 0 表示失败		

		前坐标系相关)				
29	int Get_PosHereJ (ROBOT_POS &pos, int comId=0)	查询当前位置点的关节坐标下位置参数	pos: 位置参数结构体, 代表查询的结果(仅坐标值有效, 臂参数、坐标参数无意义)	返回0表示查询成功, 小于0表示失败		
30	int Get_PosHereC (ROBOT_POS &pos, int comId=0)	查询当前位置点的基坐标下位置参数	pos: 位置参数结构体, 代表查询的结果(仅坐标值有效, 臂参数、坐标参数无意义)	返回0表示查询成功, 小于0表示失败		
31	int Get_SysErr (int &error, int comId=0)	查询系统当前故障码	error: 系统故障码, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
32	int Get_CurPrgPath (char prgPath[128], int comId=0)	查询当前示教程序的路径	prgPath: 当前程序路径, 代表查询的结果, 例如 TeachProgram/a.pro	返回0表示查询成功, 小于0表示失败		
33	int Get_PrgSts (int &sts, int comId=0)	查询当前示教程序运行状态	sts: 示教程序运行状态, 代表查询的结果, 0-停止, 1-启动/继续, 2-前进, 3-后退, 4-暂停	返回0表示查询成功, 小于0表示失败		
34	int Get_StartLine (int &line, int comId=0)	查询当前示教程序启动的行号	line: 当前程序启动的行号, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
35	int Get_CurPrgLine (int &line, int comId=0)	查询当前示教程序执行的行号	line: 当前程序执行的行号, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
36	int Get_InitSts (int &sts, int comId=0)	查询系统初始化状态	sts: 系统的初始化状态, 代表查询的结果, 范围为-1至11	返回0表示查询成功, 小于0表示失败		
37	int Get_Coord (int &type, int comId=0)	查询当前坐标系类型	type: 当前坐标系类型, 代表查询的结果, 范围1至4, 1-关节坐标系, 2-基坐标系, 3-工具坐标系, 4-用户坐标系	返回0表示查询成功, 小于0表示失败		
38	int Get_Vel (int &val, int comId=0)	查询当前速度等级值	val: 当前速度等级值, 代表查询的结果, 范围1-100	返回0表示查询成功, 小于0表示失败		
39	int Get_Mode (int &mode, int comId=0)	查询系统当前运行模式	mode: 系统运行模式, 代表查询的结果, 1-示教, 2-再现, 3-单步运行, 5-连续运行	返回0表示查询成功, 小于0表示失败		
40	int Get_DsMode (int &val, int comId=0)	查询数据流模式是否开	val: 数据流模式开启情况, 代表查询的结果, 0-关闭,	返回0表示查询成功, 小于0		

		启	1-开启	表示失败		
41	int Get_InchMode (int &val, int comId=0)	查询寸动模式是否开启	val: 寸动模式开启情况, 代表查询的结果, 0-关闭, 1-开启	返回0表示查询成功, 小于0表示失败		
42	int Get_EStopSts (int &sts, int comId=0)	查询当前急停开关状态	sts: 急停开关状态, 代表查询的结果, 0-急停松开, 1-急停按下	返回0表示查询成功, 小于0表示失败		
43	int Get_MotorSts (int &sts, int comId=0)	查询当前电机使能状态	sts: 电机使能状态, 代表查询的结果, 0-未使能, 1-使能	返回0表示查询成功, 小于0表示失败		
44	int Get_MotionSts (int &sts, int comId=0)	查询当前系统运动状态	sts: 系统运动状态, 代表查询的结果, 0-停止/运动完成, 1-运动中, 2-运动中	返回0表示查询成功, 小于0表示失败		
45	int Get_SysMode (int &mode, int comId=0)	查询系统当前模式	mode: 系统模式, 代表查询的结果, 0-正常模式, 1-测试模式	返回0表示查询成功, 小于0表示失败		
46	int Get_PrgRunTime (unsigned int &second, int comId=0)	查询系统示教程序运行时间	second: 时间计数值(秒), 代表查询的结果	返回0表示查询成功, 小于0表示失败		
47	int Get_CurCmdNum (unsigned int &num, int comId=0)	查询当前发送成功的运动指令 (Home、MovJ、MovL) 的编号	num: 指令编号, 代表查询的结果	返回0表示查询成功, 小于0表示失败		仅在数据流模式下有效
48	int Get_CurCmdSts (int &sts, int comId=0)	查询当前发送成功的运动指令实际完成状态 (是否到位)	sts: 完成状态, 代表查询的结果, 0-运动未完成, 1-运动完成	返回0表示查询成功, 小于0表示失败		仅在数据流模式下有效
49	int Get_CmdSts (int num, int &sts, int comId=0)	查询指定编号的运动指令实际完成	num: 指令编号 sts: 完成状态, 代表查询的结果, 0-运动未完成, 1-	返回0表示查询成功, 小于0表示失败		仅在数

		状态	运动完成			据流模式下有效
50						
51	int Get_DI Num(int &num, int comId=0)	查询系统DI总数	num: DI 总数, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
52	int Get_DONum (int &num, int comId=0)	查询系统DO总数	num: DO 总数, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
53	int Get_AD Num(int &num, int comId=0)	查询系统AD总数	num: AD 总数, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
54	int Get_DANum (int &num, int comId=0)	查询系统DA总数	num: DA 总数, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
55	int Get_DI (int num, int &sts, int comId=0)	按位查询DI的输入状态	num: DI 序号 (不超过 DI 总数) sts: DI 状态, 代表查询的结果, 0-off, 1-On	返回0表示查询成功, 小于0表示失败		
56	int Get_DIGroup (int num, int &sts, int comId=0)	按组查询DI的输入状态	num: DI 组序号 sts: 每组的DI 状态, 代表查询的结果, 范围0-255, 其中bit0-bit7分别对应每组序号最小至最大的DI 状态	返回0表示查询成功, 小于0表示失败		
57	int Get_AD (int num, float &val, int comId=0)	按序号查询AD的输入值	num: AD 序号 (不超过 AD 总数) val: AD 值, 代表查询的结果, 电流型单位为mA, 电压型单位为V	返回0表示查询成功, 小于0表示失败		
58	int Get_DOCfg (int num, int &val, int comId=0)	查询DO的配置权	num: DO 序号 (不超过 DO 总数) val: 配置权, 代表查询的结果, 1 表由RC控制, 0 表由PLC控制	返回0表示查询成功, 小于0表示失败		
59	int Get_DOGroupCfg (int num, int &val, int comId=0)	查询每组DO的配置权	num: DO 组号 val: 配置权, 代表查询的结果, bit0-bit7 分别代表该组每个DO的配置权, 1 表	返回0表示查询成功, 小于0表示失败		

			由 RC 控制, 0 表由 PLC 控制			
60	int Get_D0 (int num, int &sts, int comId=0)	按位查询 D0 的输出状态	num: D0 序号 (不超过 D0 总数) sts: D0 状态, 代表查询的结果, 0-off, 1-On	返回 0 表示查询成功, 小于 0 表示失败		
61	int Get_D0Group (int num, int &sts, int comId=0)	按组查询 D0 的输出状态	num: D0 组序号 sts: 每组的 D0 状态, 代表查询的结果, 范围 0-255, 其中 bit0-bit7 分别对应每组序号最小至最大的 D0 状态	返回 0 表示查询成功, 小于 0 表示失败		
62	int Get_DACfg (int num, int &val, int comId=0)	查询 DA 的配置权	num: DA 序号 val: 配置权, 代表查询的结果, 1 表由 RC 控制, 0 表不可由 RC 控制 (由 PLC 控制或未连接)	返回 0 表示查询成功, 小于 0 表示失败		
63	int Get_DA (int num, float &val, int comId=0)	按序号查询 DA 的输出值	num: DA 序号 (不超过 DA 总数) val: DA 值, 代表查询的结果, 电流型单位为 mA, 电压型单位为 V	返回 0 表示查询成功, 小于 0 表示失败		
64	int Get_DevSts (int sts[6], int comId=0)	查询系统设备的连接情况	sts[]: 系统设备连接情况, 代表查询的结果。其中, sts[0]: 网卡 1 状态, 0 表未连接, 1 表连接, 2 表被禁用; sts[1]: 网卡 2 状态, 同上; sts[2]: USB 设备状态, 0 表未连接, 1 表连接挂载成功, 2 表挂载失败; sts[3]: SD 卡状态, 0 表未连接, 1 表连接挂载成功, 2 表挂载失败, 3 表文件格式错误; sts[4]: EtherCAT0 通信状态, 0 表通信正常, 1 表从站掉线, 2 表未连接网线, 3 表连接非 ECAT 设备, 4 表已禁用; sts[5]: IRLink0 通信状态, 同上	返回 0 表示查询成功, 小于 0 表示失败		
65	int Get_FwVersion (char ver[32], int comId=0)	查询系统控制器软件版	*ver: 当前系统软件版本, 代表查询的结果, 例如	返回 0 表示查询成功, 小于 0		

		本	S01.12T01ES	表示失败		
66	int Get_SysTime (char time[16], int comId=0)	查询当前系统时间	*time: 时间字符串 (年月日时分秒), 代表查询的结果	返回0表示查询成功, 小于0表示失败		
67						
68	int Get_ServoSts (int sts[8], int comId=0)	查询系统中所有伺服的故障状态 (包括机器人轴和外部轴两部分)	sts[8]: 伺服的故障状态, 代表查询的结果。目前最大支持8个伺服轴, sts[0]对应第0号伺服, 依次类推, 0-无故障, 1-有故障	返回0表示查询成功, 小于0表示失败		
69	int Get_ServoErr (int num, int &error, int comId=0)	查询系统中单个伺服的故障码	num: 伺服轴序号, 从0开始 error: 伺服故障码, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
70						
71	int Get_StrPara (float para[6], int comId=0)	查询机器人结构参数	para[]: 结构参数, 代表查询的结果 (对于 SCARA 机器人, para[0]-para[3]有效, 对于六轴机器人, para[0]-para[5]有效, 下同)	返回0表示查询成功, 小于0表示失败		
72	int Set_StrPara (float para[6], int comId=0)	设置机器人结构参数	para[]: 结构参数	返回0表示设置成功, 小于0表示失败		
73	int Get_StrParaComp (float para[6], int comId=0)	查询机器人结构补偿参数	para[]: 结构补偿参数, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
74	int Set_StrParaComp (float para[6], int comId=0)	设置机器人结构补偿参数	para[]: 结构补偿参数	返回0表示设置成功, 小于0表示失败		
75	int Get_RdctRatio (float para[6], int comId=0)	查询关节减速比	para[]: 各关节减速比, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
76	int Set_RdctRatio (float para[6], int comId=0)	设置关节减速比	para[]: 各关节减速比	返回0表示设置成功, 小于0表示失败		
77	int Get_CpParam (float para[6], int comId=0)	查询关节主耦合参数	para[]: 各关节主耦合参数, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
78	int Set_CpParam (float para[6], int comId=0)	设置关节主耦合参数	para[]: 各关节主耦合参数	返回0表示设置成功, 小于0表示失败		

79	int Get_CpParaS (float para[6], int comId=0)	查询关节从耦合参数	para[]: 各关节从耦合参数, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
80	int Set_CpParaS (float para[6], int comId=0)	设置关节从耦合参数	para[]: 各关节从耦合参数	返回0表示设置成功, 小于0表示失败		
81	int Get_HomePos (int num, double pos[6], int comId=0)	查询工作原点	num: 工作原点序号, 范围0-4 pos[]: 工作原点对应的关节坐标值, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
82	int Set_HomePos (int num, double pos[6], int comId=0)	设置工作原点	num: 工作原点序号, 范围0-4 pos[]: 工作原点对应的关节坐标值	返回0表示设置成功, 小于0表示失败		
83	int Get_ZeroPos (int pluse[6], int comId=0)	查询绝对零点	pluse[]: 绝对零点对应的脉冲值, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
84	int Set_ZeroPos (int pluse[6], int comId=0)	设置绝对零点	pluse[]: 绝对零点对应的脉冲值	返回0表示设置成功, 小于0表示失败		
85	int Get_InchStep (int &val, int comId=0)	查询寸动步长等级	val: 寸动运行的步长等级, 代表查询结果	返回0表示设置成功, 小于0表示失败		
86	int Get_StepMotionJ (float ¶, int comId=0)	查询寸动示教的关节步长	para: 关节步长值, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
87	int Set_StepMotionJ (float para, int comId=0)	设置寸动示教的关节步长	para: 关节步长值	返回0表示设置成功, 小于0表示失败		
88	int Get_StepMotionL (float ¶, int comId=0)	查询寸动示教的线性步长	para: 线性步长值, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
89	int Set_StepMotionL (float para, int comId=0)	设置寸动示教的线性步长	para: 线性步长值	返回0表示设置成功, 小于0表示失败		
90	int Get_TeachVelLimJ (float para[6], int comId=0)	查询示教时关节速度上限	para[]: 最大允许关节速度, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
91	int Set_TeachVelLimJ (float para[6], int comId=0)	设置示教时关节速度上限	para[]: 最大允许关节速度	返回0表示设置成功, 小于0表示失败		
92	int Get_TeachVelLimL (float para[2], int comId=0)	查询示教时位置、姿态速度上限	para[2]: 最大允许位置和姿态速度, 代表查询的结果	返回0表示查询成功, 小于0表示失败		

93	int Set_TeachVelLimL (float para[2], int comId=0)	设置示教时位置和姿态速度上限	para[2]: 最大允许位置和姿态速度	返回0表示设置成功, 小于0表示失败		
94	int Get_TeachAccLimJ (float para[6], int comId=0)	查询示教时关节加速度上限	para[]: 最大允许关节加速度, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
95	int Set_TeachAccLimJ (float para[6], int comId=0)	设置示教时关节加速度上限	para[]: 最大允许关节加速度	返回0表示设置成功, 小于0表示失败		
96	int Get_TeachAccLimL (float para[2], int comId=0)	查询示教时位置、姿态加速度上限	para[]: 最大允许位置和姿态加速度, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
97	int Set_TeachAccLimL (float para[2], int comId=0)	设置示教时位置和姿态加速度上限	para[]: 最大允许位置和姿态加速度	返回0表示设置成功, 小于0表示失败		
98	int Get_RunVelLimJ (float para[6], int comId=0)	查询运行时关节速度上限	para[]: 最大允许关节速度, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
99	int Set_RunVelLimJ (float para[6], int comId=0)	设置运行时关节速度上限	para[]: 最大允许关节速度	返回0表示设置成功, 小于0表示失败		
100	int Get_RunVelLimL (float para[2], int comId=0)	查询运行时位置、姿态速度上限	para[]: 最大允许位置和姿态速度, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
101	int Set_RunVelLimL (float para[2], int comId=0)	设置运行时位置、姿态速度上限	para[]: 最大允许位置和姿态速度	返回0表示设置成功, 小于0表示失败		
102	int Get_RunAccLimJ (float para[6], int comId=0)	查询运行时关节加速度上限	para[]: 最大允许关节加速度, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
103	int Set_RunAccLimJ (float para[6], int comId=0)	设置运行时关节加速度上限	para[]: 最大允许关节加速度	返回0表示设置成功, 小于0表示失败		
104	int Get_RunAccLimL (float para[2], int comId=0)	查询运行时位置、姿态加速度上限	para[]: 最大允许位置和姿态加速度, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
105	int Set_RunAccLimL (float para[2], int comId=0)	设置运行时位置、姿态加速度上限	para[2]: 最大允许位置和姿态加速度	返回0表示设置成功, 小于0表示失败		
106	int Get_StopDecLimJ (float para[6], int comId=0)	查询运行时关节减速度上限	para[]: 最大允许关节减速度, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
107	int Set_StopDecLimJ (float para[6], int comId=0)	设置运行时关节减速度	para[]: 最大允许关节减速度	返回0表示设置成功, 小于0		

		上限		表示失败		
108	int Get_StopDecLimL (float para[2], int comId=0)	查询运行时位置、姿态减速度上限	para[]: 最大允许位置和姿态减速度, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
109	int Set_StopDecLimL (float para[2], int comId=0)	设置运行时位置、姿态减速度上限	para[]: 最大允许位置和姿态减速度	返回0表示设置成功, 小于0表示失败		
110	int Get_ZonePara (float para[2], int comId=0)	查询过渡精度参数	para[]: 线性 and 关节过渡精度, 代表查询的结果	返回0表示设置成功, 小于0表示失败		
111	int Set_ZonePara (float para[2], int comId=0)	设置过渡精度参数, 参数分别为线性和关节过渡精度	para[]: 线性 and 关节过渡精度	返回0表示设置成功, 小于0表示失败		
112	int Get_CInterpMode (int &type, int comId=0)	查询圆弧姿态插补类型	type: 插补类型, 代表查询结果, 0-关节插补, 1-姿态插补	返回0表示设置成功, 小于0表示失败		
113	int Set_CInterpMode (int type, int comId=0)	设置圆弧姿态插补类型	type: 插补类型	返回0表示设置成功, 小于0表示失败		
114	int Get_AxisNLim (int axis, float ¶, int comId=0)	查询机器人轴的负向轴极限	axis: 轴序号, 与轴数有关, 最大范围1-6, 分别对应J1-J6轴 para: 负向轴极限值, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
115	int Set_AxisNLim (int axis, float para, int comId=0)	设置机器人轴的负向轴极限	axis: 轴序号, 最大范围1-6, 分别对应J1-J6轴 para: 负向轴极限值	返回0表示设置成功, 小于0表示失败		
116	int Get_AxisPLim (int axis, float ¶, int comId=0)	查询机器人轴的正向轴极限	axis: 轴序号, 最大范围1-6, 分别对应J1-J6轴 para: 正向轴极限值, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
117	int Set_AxisPLim (int axis, float para, int comId=0)	设置机器人轴的正向轴极限	axis: 轴序号, 最大范围1-6, 分别对应J1-J6轴 para: 正向轴极限值	返回0表示设置成功, 小于0表示失败		
118	int Get_ToolC (int num, double pos[6], int comId=0)	查询工具坐标系参数	num: 工具号, 范围1-15 pos[]: 工具坐标系参数, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
119	int Set_ToolC (int num, double pos[6], int comId=0)	设置工具坐标系参数	num: 工具号, 范围1-15 pos[]: 工具坐标系参数	返回0表示设置成功, 小于0表示失败		

120	int Get_UserC (int num, double pos[6], int comId=0)	查询用户坐标系参数	num: 用户号, 范围 1-15 pos[]: 用户坐标系参数, 代表查询的结果	返回 0 表示查询成功, 小于 0 表示失败		
121	int Set_UserC (int num, double pos[6], int comId=0)	设置用户坐标系参数	num: 用户号, 范围 1-15 pos[]: 用户坐标系参数	返回 0 表示设置成功, 小于 0 表示失败		
122	int Get_ToolCNum (int &num, int comId=0)	查询当前工具坐标系号	num: 当前选择的工具号, 代表查询的结果	返回 0 表示查询成功, 小于 0 表示失败		
123	int Set_ToolCNum (int num, int comId=0)	设置当前工具坐标系号	num: 工具号	返回 0 表示设置成功, 小于 0 表示失败		
124	int Get_UserCNum (int &num, int comId=0)	查询当前用户坐标系号	num: 当前选择的用户号, 代表查询的结果	返回 0 表示查询成功, 小于 0 表示失败		
125	int Set_UserCNum (int num, int comId=0)	设置当前用户坐标系号	num: 用户号	返回 0 表示设置成功, 小于 0 表示失败		
126	int Set_Coord (int type, int comId=0)	设置当前坐标系类型	type: 当前坐标系类型, 范围 1 至 4, 1-关节坐标系, 2-基坐标系, 3-工具坐标系, 4-用户坐标系	返回 0 表示设置成功, 小于 0 表示失败		
127	int Get_Interf (int num, double pos[6], int comId=0)	查询干涉区边界点坐标参数	num: 干涉区序号, 范围 0 至 7 pos[]: 干涉区边界点坐标, 代表查询的结果, pos[0]至 pos[2] 分别对应点 1 的 XYZ 坐标, pos[3] 至 pos[5] 分别对应点 2 的 XYZ 坐标	返回 0 表示查询成功, 小于 0 表示失败		
128	int Set_Interf (int num, double pos[6], int comId=0)	设置干涉区边界点坐标参数	num: 干涉区序号 pos[]: 干涉区边界点坐标	返回 0 表示设置成功, 小于 0 表示失败		
129	int Get_CurInterf (int &num, int comId=0)	查询当前激活的干涉区编号	num: 干涉区编号, 代表查询的结果, 范围 0 至 255, 其中 bit0 至 bit7 分别对应干涉区 0 至干涉区 7, 0-未激活, 1-激活	返回 0 表示查询成功, 小于 0 表示失败		
130	int Set_CurInterf (int num, int comId=0)	设置需激活的干涉区编号	num: 干涉区编号, 同上	返回 0 表示设置成功, 小于 0 表示失败		
131	int SavePara (int comId=0)	保存系统参数, 掉电可存储		返回 0 表示操作成功, 小于 0 表示失败		

132	int RecoverPara (int comId=0)	恢复系统参数（恢复至上一次保存操作后的参数）		返回0表示操作成功，小于0表示失败	延时指令（约50ms）	
133						
134	int Get_P (int pNum, ROBOT_POS &pos, int comId=0)	查询位置变量对应的位置参数	pNum: 位置变量序号, 有效范围依据当前示教程序 pos: 位置参数结构体, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
135	int Set_P (int pNum, ROBOT_POS pos, int comId=0)	设置位置变量对应的位置参数	pNum: 位置变量序号, 有效范围依据当前示教程序 pos: 位置参数结构体	返回0表示设置成功, 小于0表示失败		
136	int Set_Phere (int pNum, int comId=0)	用当前点位的参数设置位置变量	pNum: 位置变量序号	返回0表示设置成功, 小于0表示失败		
137	int Get_PR (int prNum, ROBOT_POS &pos, int comId=0)	查询全局平移变量对应的参数	prNum: 全局平移变量序号, 范围0至255 pos: 位置参数结构体, 代表查询的结果, 其中臂参数无效	返回0表示查询成功, 小于0表示失败		
138	int Set_PR (int prNum, ROBOT_POS pos, int comId=0)	设置全局平移变量对应的参数	prNum: 全局平移变量序号, 范围0至255 pos: 位置参数结构体, 其中臂参数无效	返回0表示设置成功, 小于0表示失败		
139	int WriteFile_PR (int comId=0)	保存所有PR变量, 掉电可存储		返回0表示操作成功, 小于0表示失败		
140	int Get_LPR (int prNum, ROBOT_POS &pos, int comId=0)	查询局部平移变量对应的参数	prNum: 局部平移变量序号, 范围0至255 pos: 位置参数结构体, 代表查询的结果, 其中臂参数无效	返回0表示查询成功, 小于0表示失败		
141	int Set_LPR (int prNum, ROBOT_POS pos, int comId=0)	设置局部平移变量对应的参数	prNum: 局部平移变量序号, 范围0至255 pos: 位置参数结构体, 其中臂参数无效	返回0表示设置成功, 小于0表示失败		
142	int Get_B (int num, int &val, int comId=0)	查询全局B变量的值	num: B变量序号 val: B变量值, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
143	int Set_B (int num, int val, int comId=0)	设置全局B变量的值	num: B变量序号 val: B变量值, 范围0-255	返回0表示设置成功, 小于0表示失败		

144	int Get_R (int num, int &val, int comId=0)	查询全局R变量的值	num: R 变量序号 val: R 变量值, 代表查询的结果	返回0 表示查询成功, 小于0 表示失败		
145	int Set_R (int num, int val, int comId=0)	设置全局R变量的值	num: R 变量序号 val: R 变量值, 范围-65536 至 65535	返回0 表示设置成功, 小于0 表示失败		
146	int Get_D (int num, double &val, int comId=0)	查询全局D变量的值	num: D 变量序号 val: D 变量值, 代表查询的结果	返回0 表示查询成功, 小于0 表示失败		
147	int Set_D (int num, double val, int comId=0)	设置全局D变量的值	num: D 变量序号 val: D 变量值, 范围-9999999.999 至 9999999.999	返回0 表示设置成功, 小于0 表示失败		
148	int Get_LB (int num, int &val, int comId=0)	查询局部LB变量的值	num: LB 变量序号 val: LB 变量值, 代表查询的结果	返回0 表示查询成功, 小于0 表示失败		
149	int Set_LB (int num, int val, int comId=0)	设置全局LB变量的值	num: LB 变量序号 val: LB 变量值, 范围0-255	返回0 表示设置成功, 小于0 表示失败		
150	int Get_LR (int num, int &val, int comId=0)	查询全局LR变量的值	num: LR 变量序号 val: LR 变量值, 代表查询的结果	返回0 表示查询成功, 小于0 表示失败		
151	int Set_LR (int num, int val, int comId=0)	设置全局LR变量的值	num: LR 变量序号 val: LR 变量值, 范围-65536 至 65535	返回0 表示设置成功, 小于0 表示失败		
152	int Get_LD (int num, double &val, int comId=0)	查询全局LD变量的值	num: LD 变量序号 val: LD 变量值, 代表查询的结果	返回0 表示查询成功, 小于0 表示失败		
153	int Set_LD (int num, double val, int comId=0)	设置全局LD变量的值	num: LD 变量序号 val: LD 变量值, 范围-9999999.999 至 9999999.999	返回0 表示设置成功, 小于0 表示失败		
154	int Get_CommonVarUchar (int address, unsigned char &val, int comId=0)	查询公共变量区的 unsigned char 型数据	adress: 公共区偏移地址, 范围0-8191 (下同) val: 查询的结果	返回0 表示设置成功, 小于0 表示失败		
155	int Set_CommonVarUchar (int address, unsigned char val, int comId=0)	设置公共变量区的 unsigned char 型数据	adress: 公共区偏移地址 val: 需要设置的数据	返回0 表示设置成功, 小于0 表示失败		
156	int Get_CommonVarChar (int address, char &val, int comId=0)	查询公共变量区的 char 型数据	adress: 公共区偏移地址 val: 查询的结果	返回0 表示设置成功, 小于0 表示失败		

157	int Set_CommonVarChar (int address, char val, int comId=0)	设置公共变量区的 char 型数据	address: 公共区偏移地址 val: 需要设置的数据	返回 0 表示设置成功, 小于 0 表示失败		
158	int Get_CommonVarUshort (int address, unsigned short &val, int comId=0)	查询公共变量区的 unsigned short 型数据 (2 bytes)	address: 公共区偏移地址 val: 查询的结果	返回 0 表示设置成功, 小于 0 表示失败		
159	int Set_CommonVarUshort (int address, unsigned short val, int comId=0)	设置公共变量区的 unsigned short 型数据	address: 公共区偏移地址 val: 需要设置的数据	返回 0 表示设置成功, 小于 0 表示失败		
160	int Get_CommonVarShort (int address, short &val, int comId=0)	查询公共变量区的 short 型数据 (2 bytes)	address: 公共区偏移地址 val: 查询的结果	返回 0 表示设置成功, 小于 0 表示失败		
161	int Set_CommonVarShort (int address, short val, int comId=0)	设置公共变量区的 short 型数据	address: 公共区偏移地址 val: 需要设置的数据	返回 0 表示设置成功, 小于 0 表示失败		
162	int Get_CommonVarUint (int address, unsigned int &val, int comId=0)	查询公共变量区的 unsigned int 型数据 (4 bytes)	address: 公共区偏移地址 val: 查询的结果	返回 0 表示设置成功, 小于 0 表示失败		
163	int Set_CommonVarUint (int address, unsigned int val, int comId=0)	设置公共变量区的 unsigned int 型数据	address: 公共区偏移地址 val: 需要设置的数据	返回 0 表示设置成功, 小于 0 表示失败		
164	int Get_CommonVarInt (int address, int &val, int comId=0)	查询公共变量区的 int 型数据 (4 bytes)	address: 公共区偏移地址 val: 查询的结果	返回 0 表示设置成功, 小于 0 表示失败		
165	int Set_CommonVarInt (int address, int val, int comId=0)	设置公共变量区的 int 型数据	address: 公共区偏移地址 val: 需要设置的数据	返回 0 表示设置成功, 小于 0 表示失败		
166	int Get_CommonVarFloat (int address, float &val, int comId=0)	查询公共变量区的 float 型数据 (4 bytes)	address: 公共区偏移地址 val: 查询的结果	返回 0 表示设置成功, 小于 0 表示失败		
167	int Set_CommonVarFloat (int address, float val, int comId=0)	设置公共变量区的 float 型数据	address: 公共区偏移地址 val: 需要设置的数据	返回 0 表示设置成功, 小于 0 表示失败		

		据				
168	int Get_CommonVarDouble (int address, double &val, int comId=0)	查询公共变量区的 double 型数据 (8 bytes)	address: 公共区偏移地址 val: 查询的结果	返回 0 表示设置成功, 小于 0 表示失败		
169	int Set_CommonVarDouble (int address, double val, int comId=0)	设置公共变量区的 double 型数据	address: 公共区偏移地址 val: 需要设置的数据	返回 0 表示设置成功, 小于 0 表示失败		
170	int Get_CommonVarP (int address, ROBOT_POS &pos, int comId=0)	查询公共变量区的机器人位置数据	address: 公共区偏移地址 pos: 查询的结果	返回 0 表示设置成功, 小于 0 表示失败		
171	int Set_CommonVarP (int address, ROBOT_POS pos, int comId=0)	设置公共变量区的机器人位置数据	address: 公共区偏移地址 pos: 需要设置的数据	返回 0 表示设置成功, 小于 0 表示失败		
172	int Get_ModbusCoil (int address, int sum, int &val, int comId=0)	查询 Modbus 变量区的线圈值	address: Modbus 区线圈地址, 范围 0-8191 sum: 读取的线圈总个数, 范围 1-8 val: 线圈值, 代表查询的结果	返回 0 表示查询成功, 小于 0 表示失败		
173	int Set_ModbusCoil (int address, int sum, int val, int comId=0)	设置 Modbus 变量区的线圈值	address: Modbus 区线圈地址, 范围 2048-4095, 6144-8191 sum: 读取的线圈总个数, 范围 1-8 val: 线圈值	返回 0 表示设置成功, 小于 0 表示失败		
174	int Get_ModbusRegUshort (int address, int sum, unsigned short val[], int comId=0)	查询 Modbus 变量区的寄存器值, 数据类型为 unsigned short	address: modbus 区寄存器地址, 范围 0-65535 sum: 读取的寄存器总个数 val: 代表查询的结果	返回 0 表示查询成功, 小于 0 表示失败		
175	int Set_ModbusRegUshort (int address, int sum, unsigned short val[], int comId=0)	设置 Modbus 变量区的寄存器值, 数据类型为 unsigned short	address: modbus 区寄存器地址, 范围 16384-32767, 49152-65535 sum: 读取的寄存器总个数 val: 代表查询的结果	返回 0 表示设置成功, 小于 0 表示失败		
176	int Get_ModbusRegFloat (int address, int sum, float val[], int comId=0)	查询 Modbus 变量区的寄存器值, 数据类型为	address: modbus 区寄存器地址, 范围 0-65535 sum: 读取的寄存器总个数 val: 代表查询的结果(一个	返回 0 表示查询成功, 小于 0 表示失败		

		float	float 数据占用2个寄存器)			
177	int Set_ModbusRegFloat (int address, int sum, float val[], int comId=0)	设置Modbus变量区的寄存器值,数据类型为float	address: modbus 区寄存器地址, 范围16384-32767, 49152-65535 sum: 读取的寄存器总个数 val: 代表查询的结果	返回0表示设置成功, 小于0表示失败		
178	int Get_PlcVarByte (int num, unsigned char &val, int comId=0)	查询PLC Byte 型变量的值	num: Byte 变量序号, 范围0-255 val: 变量值, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
179	int Get_PlcVarInt (int num, short &val, int comId=0)	查询PLC Int 型变量的值	num: Int 变量序号, 范围0-255 val: 变量值, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
180	int Get_PlcVarDInt (int num, int &val, int comId=0)	查询PLC DInt 型变量的值	num: DInt 变量序号, 范围0-255 val: 变量值, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
181	int Get_PlcVarLReal (int num, double &val, int comId=0)	查询PLC LReal 型变量的值	num: LReal 变量序号, 范围0-255 val: 变量值, 代表查询的结果	返回0表示查询成功, 小于0表示失败		
182	int Get_UserAlarm (int num, char alarm[40], int comId=0)	查询自定义报警的内容	num: 自定义报警序号, 范围0-15 alarm: 报警内容描述, 代表查询的结果, 字节长不超过40 bytes	返回0表示查询成功, 小于0表示失败		
183	int Set_UserAlarm (int num, char alarm[40], int comId=0)	设置自定义报警的内容	num: 自定义报警序号, 范围0-15 alarm: 报警内容描述, 字节长不超过40byte	返回0表示设置成功, 小于0表示失败		
184	int Get_Print (char val[120], int comId=0)	查询控制器打印信息, 包括程序 print 指令的打印内容和系统错误提示内容	val: 打印内容, 代表查询的结果, 字节长不超过120 bytes	返回0表示设置成功, 小于0表示失败		
185						

186	<code>int Get_InCfg(int func, int &diNum, int comId=0)</code>	查询输入功能所对应的 DI 序号	func: 输入功能序号, 0-启动, 1-停止, 2-暂停, 3-急停, 4-清除报警, 5-程序 1, 6-程序 2, 7-程序 3, 8 和 9 为预留, 10-速度加, 11-速度减 diNum: DI 序号, 代表查询的结果, -1 表示未设置对应 DI, 其它范围为 3-15	返回 0 表示查询成功, 小于 0 表示失败		
187	<code>int Set_InCfg(int func, int diNum, int comId=0)</code>	设置输入功能所对应的 DI 序号	func: 输入功能序号, 0-启动, 1-停止, 2-暂停, 3-急停, 4-清除报警, 5-程序 1, 6-程序 2, 7-程序 3, 8 和 9 为预留, 10-速度加, 11-速度减 diNum: DI 序号, 范围为 3-15, -1 表示不设置	返回 0 表示设置成功, 小于 0 表示失败		
188	<code>int Get_OutCfg(int func, int &doNum, int comId=0)</code>	查询输出功能所对应的 DO 序号	func: 输出功能序号, 0-报警, 1-运行, 2-停止, 3-启动完成, 4-使能 doNum: DO 序号, 代表查询的结果, -1 表示未设置对应 DO, 其它范围为 0-15	返回 0 表示查询成功, 小于 0 表示失败		
189	<code>int Set_OutCfg(int func, int doNum, int comId=0)</code>	设置输出功能所对应的 DO 序号	func: 输出功能序号, 0-报警, 1-运行, 2-停止, 3-启动完成, 4-使能 doNum: DO 序号, 范围为 0-15, -1 表示不设置	返回 0 表示设置成功, 小于 0 表示失败		
190						
191	<code>int CurCtrlDev(int &dev, int comId=0)</code>	查询当前控制权所属设备	dev: 当前控制权设备编号, 0-示教器, 1-InoRobShop 平台, 2-远程以太网设备, 3-远端 I0, 4-远端 modbus	返回 0 表示查询成功, 小于 0 表示失败		
192	<code>int CurPermit(int &owner, DWORD &IpAddr, int &IpPort, int comId=0)</code>	查询当前拥有控制权许可的以太网设备信息	owner: 获得许可的以太网设备身份, 代表查询的结果, 0-无以太网设备获得许可, 1-当前设备获得许可, 2-其它以太网设备获得许可 IpAddr: 设备网络 IP 地址, 代表查询的结果, 当第一个返回值为 0 时, 该值无意义 IpPort: 设备网络端口号, 代表查询的结果	返回 0 表示查询成功, 小于 0 表示失败		

193	int AcqPermit (int cmd=0, int comId=0)	当前API网络客户端设备请求获取控制权许可	cmd: 请求命令, 0表示一般请求, 1表示强制获取, 默认为0	返回0表示获取成功, 小于0表示失败		
194	int RemovePermit (int comId=0)	当前API网络客户端设备释放控制权		返回0表示释放成功, 小于0表示失败		
195	int CurUserType (int &type, int comId=0)	查询当前用户的模式	type: 用户模式, 代表查询的结果, 0-客户模式, 1-编辑模式, 2-管理模式, 3-厂家模式	返回0表示查询成功, 小于0表示失败		
196	int UserLogin (int type, char password[8], int comId=0)	当前API网络客户端设备登陆对应的用户模式	type: 用户模式, 0-客户模式, 1-编辑模式, 2-管理模式, 3-厂家模式 password: 登陆的密码	返回0表示登陆成功, 小于0表示失败		
197	int UserLogout (int comId=0)	当前API网络客户端设备退出当前登陆模式, 恢复为默认的客户模式		返回0表示退出成功, 小于0表示失败		
198						
199	int Set_SysTime (char time[16], int comId=0)	设置当前系统时钟	time: 时间字符串(年月日时分秒), 长度为14	返回0表示设置成功, 小于0表示失败		