INOVANCE

汇川技术机器人

设计应用与维护手册

综合手册

A00

资料编码 19011029

前 言

首先感谢您购买使用汇川技术开发生产的机器人系统产品！

本手册描述了正确使用机器人所需的事项，在使用该机器人系统产品前，请仔细阅读本手册与其他相关手册。若对一些功能及性能方面有所疑惑，请咨询我公司的技术支持人员，以获得帮助，对正确使用本产品有利。

阅读之后，请妥善保管本手册，以便随需取阅。

说 明
◆ 本公司致力于产品的不断改善，产品功能会不断升级，所提供的资料如有变更，恕不另行通知。如需获取最新版本资料，请登录汇川技术网站 www.inovance.com 下载；
◆ 如果您使用中有问题，请与本公司各区域代理商联系，或直接与本公司客户服务中心联系。客服电话：400-777-1260

目录

安全注意事项

安全声明

1） 在安装、操作、维护产品时，请先阅读并遵守安全注意事项。

2） 为保障人身和设备安全，在安装、操作和维护产品时，请遵循产品上标识及手册中说明的所有安全注意事项。

3） 手册中的“注意”、“警告”和“危险”事项，并不代表所应遵守的所有安全事项，只作为所有安全注意事项的补充。

4） 本产品应在符合设计规格要求的环境下使用，否则可能造成故障，因未遵守相关规定引发的功能异常或部件损坏等不在产品质量保证范围之内。

5） 因违规操作产品引发的人身安全事故、财产损失等，我司将不承担任何法律责任。

安全等级定义

⚠ 危 险	“危险”表示如果不按规定操作，则导致死亡或严重身体伤害。
⚠ 警 告	“警告”表示如果不按规定操作，则可能导致死亡或严重身体伤害。
⚠ 注 意	“注意”表示如果不按规定操作，则可能导致轻微身体伤害或设备损坏。

安全注意事项

系统设计时
⚠ 危 险 ◆ 请务必对系统设置安全护栏，防止人员进入系统的动作区域内，否则可能造成严重的安全问题。 ◆ 设计和制造末端执行器时，请确保其不会因为动力变化而产生危险。
开箱验收时
⚠ 注 意 ◆ 开箱前请检查设备的外包装是否完好，有无破损、浸湿、受潮、变形等情况。 ◆ 请按照层次顺序打开包装，严禁猛烈敲打！ ◆ 开箱时请检查设备及附件表面有无残损、锈蚀、碰伤等情况。 ◆ 开箱后请仔细对照装箱清单，查验设备及附件数量、资料是否齐全。
⚠ 警 告 ◆ 开箱时发现设备及附件有损伤、锈蚀、使用过的迹象等问题，请勿安装！ ◆ 开箱时发现设备有内部进水、部件缺少或部件损坏的情况时，请勿安装！ ◆ 请仔细对照装箱清单，发现装箱清单与设备名称不符时，请勿安装！ ◆ 请按照包装箱指示的开箱方向进行开箱。
储存与运输时
⚠ 危 险 ◆ 请由具有资格的作业人员进行司索、起重机起吊作业或叉车驾驶等搬运作业，否则可能造成重伤或重大损害。

储存与运输时

警 告

- 请尽可能在原包装状态下用吊车和叉车等进行搬运。
- 使用吊车、起重机等搬运设备时，作业者需穿戴个人防护装置，搬运路线周围禁止人员站立或停留。
- 吊起设备时，请用手扶住以确保平衡，起吊不稳可能会导致设备掉落，造成重伤或重大损害。

注 意

- 请按照设备的储存与运输条件进行储存与运输，储存温度、湿度满足要求。
- 避免在水溅雨淋、阳光直射、强电场、强磁场、强烈振动等场所储存与运输。
- 避免设备储存时间超过 3 个月，储存时间过长时，请进行更严密的防护和必要的检验。
- 请将设备进行严格包装后再进行车辆运输，长途运输时必须使用封闭的箱体。
- 严禁将本设备与可能对本设备构成影响或损害的设备或物品一起混装运输。
- 在运输含锂电池的设备时，必须遵守运输规定。

安装时

危 险

- 严禁非专业人员进行设备安装、接线、保养维护、检查或部件更换！
- 请务必对系统安装安全护栏，否则可能造成严重的安全问题。
- 安装系统时，请勿与周围的建筑物、结构件或设备等产生干扰，否则可能会因工具或工件撞到外围设备造成重伤或重大损害。

警 告

- 严禁改装本设备！
- 请勿在强电场或强电磁波干扰的场所安装本设备！
- 拆卸设备安装螺钉，请扶住设备防止设备翻倒。

接线时

危 险

- 严禁非专业人员进行设备安装、接线、保养维护、检查或部件更换！
- 请务必断开电源后进行接线作业，否则可能会有触电的危险或导致系统故障。
- 接线前，请切断所有设备的电源。切断电源后设备内部电容有残余电压，请至少等待 10 分钟再进行接线等操作。
- 接线时，请务必保证紧急停止开关和安全门等安全相关输入信号正确接入，否则系统发生紧急状况时将无法发挥安全功能，可能会导致重伤或重大损害。
- 请务必保证设备的良好接地，否则会有电击的危险。
- 请遵守静电防止措施（ESD）规定的步骤，并佩戴静电手环进行接线等操作，避免损坏设备内部的电路。

警 告

- 请将线缆连接牢固。请勿在线缆上放置重物，请勿强行弯曲或拉拽线缆，否则可能造成线缆损坏、断线或接触不良，有触电的危险或导致系统故障。
- 接线时使用到的线缆必须符合相应的线径和屏蔽等要求，使用屏蔽线缆时屏蔽层需要单端可靠接地！
- 接线时请勿弄错连接关系，否则系统将无法正常工作，还可能造成安全问题。
- 接线完成后，请确保设备内部没有掉落的螺钉或裸露线缆。

操作时

危 险

- 操作前，请确认紧急停止开关与安全门开关工作正常，否则系统发生紧急状况时将无法发挥安全功能，可能会导致重伤或重大损害。
- 请在规定的环境条件下使用系统，否则可能会缩短设备的使用寿命，还可能会造成严重的安全问题。
- 请在规格范围内使用系统，否则可能会缩短设备的使用寿命，还可能会造成严重的安全问题。

操作时
⚠ 警 告 ◆ 操作前，请检查是否正确安装系统、是否正确连接线缆和气源、是否正确与周边设备进行连接。 ◆ 操作前，请确认安全护栏内侧没有人。系统动作期间，请勿进入其动作区域内，否则可能造成严重的安全问题。 ◆ 请勿随意更改出厂设置。 ◆ 如果在操作期间系统进行异常动作，请立即按下紧急停止开关，否则可能产生严重的安全问题。 ◆ 系统正常工作时，请勿通过断开电源的方法来停止系统或随意按下紧急停止开关，否则容易造成设备损坏。
保养和维修时
⚠ 危 险 ◆ 严禁非专业人员进行设备安装、接线、保养维护、检查或部件更换！ ◆ 严禁在通电状态下进行设备保养或维护，否则有触电危险！ ◆ 请勿对本手册未记载的部位进行拆卸，或按照与记载不同的方法进行维护。
⚠ 警 告 ◆ 请按照产品保修协议进行设备报修。 ◆ 请按照设备维护和保养要求对设备进行日常和定期检查与保养，并做好保养记录。 ◆ 设备出现故障或损坏时，由专业人员按照维修指导对设备进行故障排除和维修，并做好维修记录。 ◆ 请按照手册中的更换指导进行部件更换。 ◆ 维护时请勿使异物进入到设备内部与连接端子中。 ◆ 除进行保养和维护作业时以外，请不要打开设备的盖子。 ◆ 更换设备后，请务必重新对设备接线检查与参数设置。
报废时
⚠ 警 告 ◆ 请按照国家有关规定与标准进行设备的报废，以免造成财产损失或人员伤亡！ ◆ 报废的设备请按照工业废弃物处理标准进行处理回收，避免污染环境。

警告标签

机器人本体上粘贴有下述警告标签。在粘贴这些标签的位置附近存在特有的危险，操作时请充分注意。为了安全地操作、维护机器人系统，请务必遵守警告标签上记载的注意与警告内容。另外，请勿损坏、损伤或剥下这些标签。

安全标识	安全说明
	◆ 由于机器人重心前置，为防止机器前倾造成机器损坏或人身危险，请将机器人固定后再拆除底座安装螺钉。
	◆ 机器人运转期间，切勿进入到动作区域内。否则可能会撞到机器人，还可能会造成严重的安全问题，非常危险。
	◆ 如果通电期间触摸内部通电部分，则可能会导致触电。

安装篇

安装篇 - 目录

第 1 章 机器人系统组成

1.1 系统组成

机器人系统由机器人本体系统、机器人控制柜、扩展模块（含扩展卡类）示教器等组成，并常与视觉系统、上位机软件、HMI 等外围设备组成整套系统解决方案，主要应用于手机制造、锂电、TP、笔电、光伏、汽车电子等行业自动化应用领域。

其中，机器人本体、机器人控制柜为机器人系统必备部件，根据项目实际需求 , 可选配示教器、扩展模块（含扩展卡类）作为选配件。

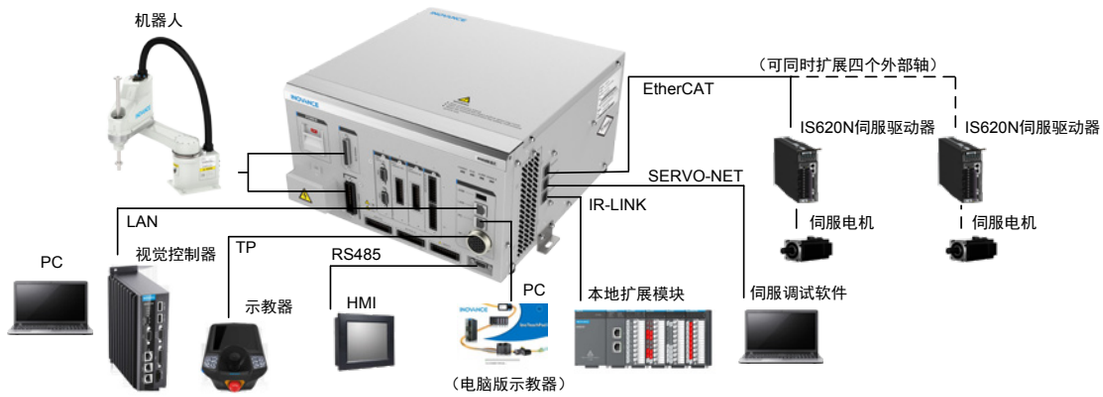


图 1-1 系统组成

类型	系列	产品简介	产品信息			型号说明
			臂长	负载	Z 轴行程	
机器人本体	IRS100 系列 SCARA 机器人	第一代紧凑型 SCARA 机器人	400mm	3kg	150mm	IRB100-3-40Z15TS3
			500mm	6kg	200mm	IRB100-6-50Z20TS3
			600mm	6kg	200mm	IRB100-6-60Z20TS3
			650mm	6kg	200mm	IRB100-6-65Z20TS3
			700mm	6kg	200mm	IRB100-6-70Z20TS3
	IRS100-20 系列 SCARA 机器人	SCARA 机器人在大负载应用方面的延伸	600mm	20kg	180mm	IRB100-20-60Z18TS3
			700mm	20kg	180mm	IRB100-20-70Z18TS3
			800mm	20kg	420mm	IRB100-20-80Z42TS3
			1000mm	20kg	420mm	IRB100-20-100Z42TS3
机器人控制柜	IRCB10 系列	4 轴 SCARA 机器人专用电气控制柜	支持本地扩展模块扩展			IRCB10-A4SH24111
	IRCB300 系列		提供用户输入 / 输出 IO，SAFETY/LAN/PC/Ether CAT/RS232/RS485 接口，支持扩展卡和本地扩展模块扩展			IRCB300-B-F IRCB300-B-G
机器人视觉控制器	IRVC100 系列	基于 Intel J1900 CPU 开发的工业视觉控制器	可实现模板匹配、缺陷检测、引导等功能；可同时挂载 2 个 GigE 工业相机，实现多维度视觉应用			IRVC100-C1-X(赛扬) IRVC100-C2-X（酷睿）
扩展模块	IO 扩展模块	控制柜配套本地扩展模块	8 通道输入 8 通道输出通用 IO 扩展模块			IMC100-0808-ETND
	编码器模块		2 通道差分输入增量编码器采集扩展模块			IMC100-2ENID
	AD 转换模块		4 通道电压和 4 通道电流模拟量转换输入采集扩展模块			IMC100-8AD
	DA 转换模块		4 通道电压或电流模拟量转换输出扩展模块			IMC100-4DA
	IRlink 通讯模块		IRlink 通讯扩展模块			IMC100-RTU-ICT

类型	系列	产品简介	产品信息			型号说明
			臂长	负载	Z 轴行程	
扩展卡	16 通道输入 IO 扩展卡	IRCB300 控制柜配套扩展卡	16 通道通用输入 IO 扩展卡			IRCB-1600END-BD
	16 通道 PNP 输出 IO 扩展卡		16 通道 PNP 型输出通用 IO 扩展卡			IRCB-0016ETPD-BD
	16 通道 NPN 输出 IO 扩展卡		16 通道 NPN 输出 IO 扩展卡			IRCB-0016ETND-BD
	2 通道增量型编码器扩展卡		2 通道差分输入增量编码器扩展卡			IRCB-2EN1D-BD
	4 通道 AD 转换扩展卡		2 通道电压和 2 通道电流采样扩展卡			IRCB-4AD-BD
	4 通道 DA 转换扩展卡		4 通道电压或电流输出扩展卡			IRCB-4DA-BD
示教器	IRTP80 系列	控制柜配套示教器	可进行触摸和物理按键操作，具有参数配置，示教编程与运行，变量与 IO 监控等多种功能			IRTP80 系列
	ITP100 系列		采用摇杆、触摸、离线编程等操作方式，支持图形化编程方式，支持用户二次开发，并支持用户自定义的函数开发			ITP100-A（带摇杆） ITP100-B（不带摇杆）

1.2 系统设计要求

1.2.1 末端夹具的设计要求

为充分发挥机器人自身具备的性能，请将机器人末端负载的大小与相对第 4 关节中心的转动惯量设为额定值以内，且勿使其从第 4 关节中心产生偏心。

NOTE

◆ 在负载或转动惯量超过额定值，或不可避免地产生偏心时，机器人运动速度将有一定程度的下降，从而影响机器人性能发挥；

◆ 负载或转动惯量超过额定值时，若机器人仍然维持较高的速度运行，可能会缩短各结构部件的使用寿命。

■ 机器人末端负载大小计算

一般情况下，末端负载是含夹具及工件的重量，在这里假设为 M。

如果在机械臂上安装相机、气动阀等情况下，需要将其重量换算为轴的等效重量 Meq，加到负载重量中。此时末端负载的重量 M0=M+Meq。

等效重量的计算公式：

安装在第 2 机械臂根部时：$Meq=M(S1)^2/(S1+S2)^2$

安装在第 2 机械臂顶端时： $Meq=M(Seq)^2/(S2)^2$

Meq：等效重量

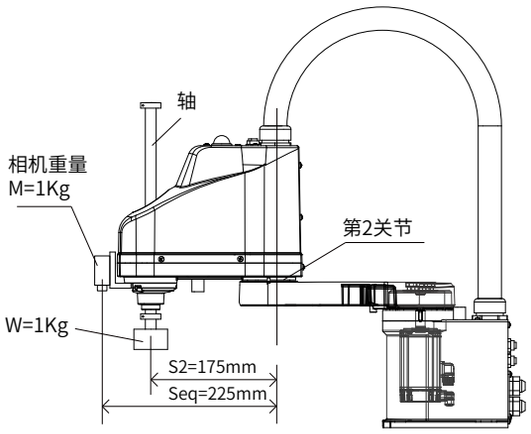
M：相机 / 气动阀等的重量

S1：第 1 机械臂长

S2：第 2 机械臂长

Seq：第 2 关节旋转中心 ~ 相机等的重心之间的距离。

举例：在负载重量 M1=1kg 的机械臂顶端（距第 2 关节旋转中心 180mm 处）安装 1kg 的相机。计算如下。



M=1

S2=175

Seq=225

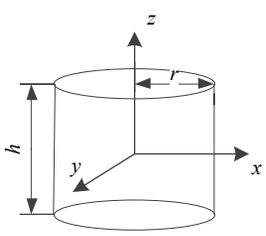
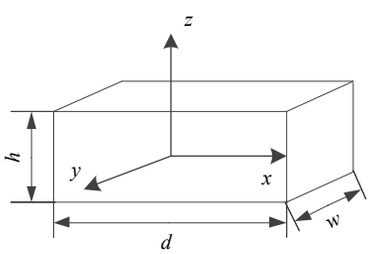
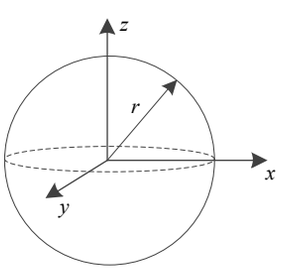
Meq=1×$225^2/175^2$=1.653→1.7（四舍五入）

M0=M+Meq=1+1.7=2.7

■ 机器人末端负载转动惯量计算

可将负载划分为若干规则形状，按各形状转动惯量之和求出全体负载的转动惯性（力矩）。

常见的工件形状转动惯量的计算公式如下：

常见形状	计算公式
圆柱 	$I_z = \frac{1}{2}mr^2$ $I_x = I_y = \frac{1}{12}m(3r^2 + h^2)$
长方体 	$I_z = \frac{1}{12}m(d^2 + w^2)$ $I_y = \frac{1}{12}m(d^2 + h^2)$ $I_x = \frac{1}{12}m(w^2 + h^2)$
实心球体 	$I = \frac{2}{5}mr^2$

1.2.2 标准动作区域

动作区域：是指标准（最大）规格时的情况。各关节电动机励磁时，在图中所示的范围内，机械手第 3 关节（轴）下端中心进行动作。

机械挡块前的区域：是指各关节电动机未励磁时，第 3 关节下端中心可移动的范围。

机械挡块：以机械方式设定不许移动到挡块以外的绝对动作区域的挡块。

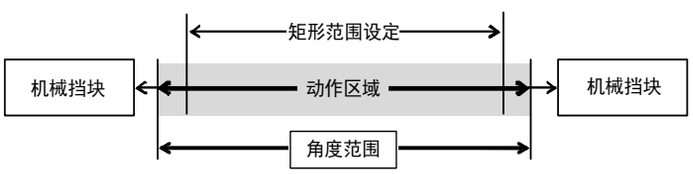
最大区域：机械臂可能产生干扰的范围。安装半径超过 60 mm（IRS100 系列）/68mm（IRS100-20）的末端夹具时，请将“机械挡块前的区域 + 末端夹具半径”设为最大区域。

有关动作区域的图形，请参阅“机器人安装空间”相关章节。

1.2.3 设定各关节运动范围

按下述三种方法设定动作区域：

- 基于角度运动范围的设定（全关节）
- 基于机械挡块的设定（第 1 关节～第 3 关节）
- 利用示教器自由设置空间中的干涉区



为了提高布局效率或出于安全考虑等而限制动作区域时，请根据 2.9.1 ～ 2.9.2 的说明进行设定。

1 基于角度运动范围的设定（全关节）

机器人的基本动作单位为度。利用各关节的角度下限值与角度上限值（角度范围）设定机器人的动作极限（动作区域）。

由伺服马达的编码器输出提供脉冲值确定运动角度， 务必将角度运动范围设在机械挡块设定值里面。

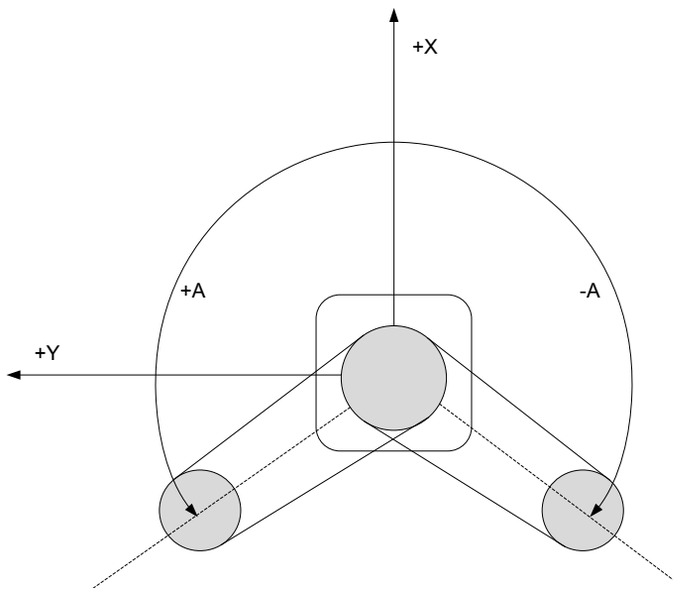
机器人接收动作命令时，会在动作之前检查发出命令的目标位置是否在角度范围内。 如果目标位置位于设定的角度范围以外，则会发生错误并不进行动作。

NOTE

◆ 角度范围在示教器“【设置】-【运动参数】-【轴参数设置】-【轴极限】”进行设置。

1） 第 1 关节最大角度范围

第 1 关节的 0 角度位置是指第 1 机械臂朝向 X 坐标轴正方向的位置。从 0 脉冲位置向逆时针方向的为正角度值，向顺时针方向的为负角度值。

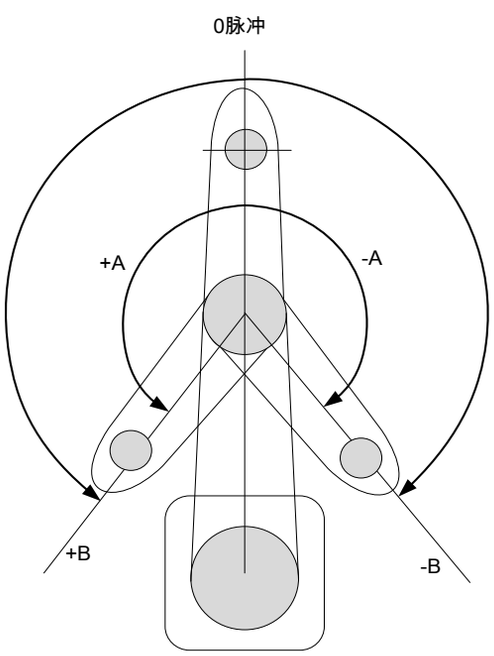


型号	A：最大动作范围
IRS100-3-40Z*	±132 °
IRS100-6-60Z*	±132 °
IRB100-20*	±132 °

图 1-2 第 1 关节最大角度范围

2） 第 2 关节最大角度范围

第 2 关节的 0 角度位置是指第 2 机械臂垂直于第 1 机械臂的位置。（第 1 机械臂朝向任何方向都是如此。）从 0 角度位置向逆时针方向的为正角度值，向顺时针方向的为负角度值。

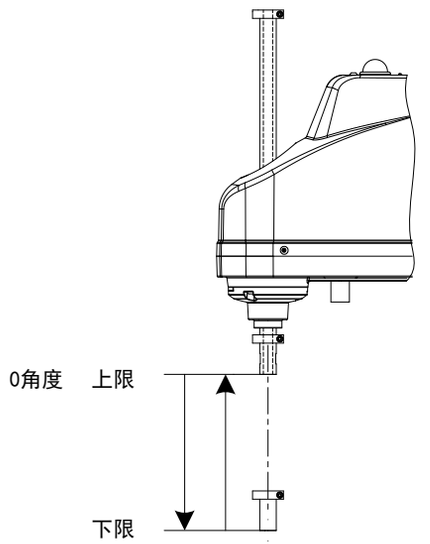


型号	A：最大动作范围
IRS100-3-40Z*	± 141 °
IRS100-6-60Z*	± 150 °
IRB100-20*	±152 °

图 1-3 第 2 关节最大角度范围

3） 第 3 关节最大行程范围

第 3 关节的 0 角度位置是指轴的上限位置。第 3 关节从 0 角度位置下降时，必定会变为负角度值。

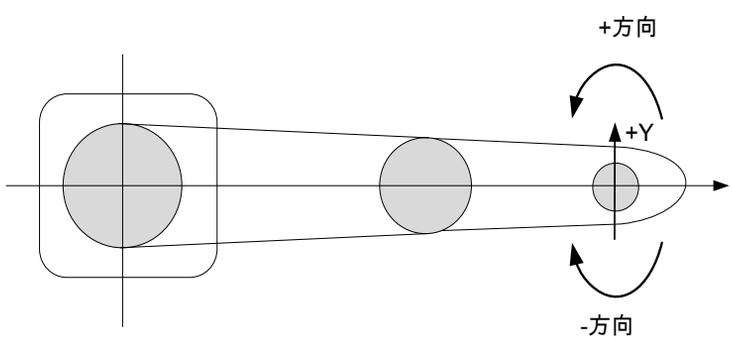


型号	第 3 关节行程
IRS100-*Z15	150 mm
IRS100-*Z20	200 mm
IRS100-*Z18	180mm
IRS100-*Z42	420mm

图 1-4 第 3 关节最大行程

4） 第 4 关节最大角度范围

第 4 关节的 0 角度位置是指轴顶端的平面朝向第 2 机械臂顶端方向的位置。（第 2 机械臂朝向任何方向都是如此。）从 0 角度位置向逆时针方向的为 + 角度值，向顺时针方向的为负角度值。



型号	A：最大角度范围
IRS100-3-40Z*	0±360°
IRS100-6-60Z*	0±360°
IRB100-20*	0±360°

图 1-5 第 4 关节最大角度范围

2 基于机械挡块的设定（第 1 关节～第 3 关节）

1） 机械限位挡块说明

如下图所示，IRS100 系列和 IRS100-20 机型的第 1、2、3 关节存在机械限位挡块，第 4 关节不存在机械限位挡块；其中第 1、2 关节的机械限位挡块为不可调节，其设定动作区域为极限运动位置区域；第 3 关节机械限位挡块为可调节的，用户可根据需要进行运动范围调节。

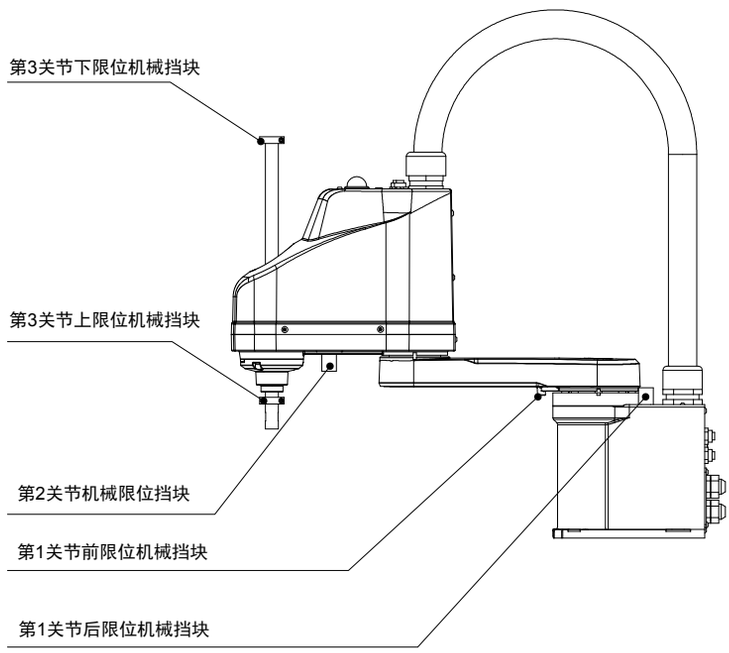


图 1-6 IRS100 系列机械挡块位置示意图

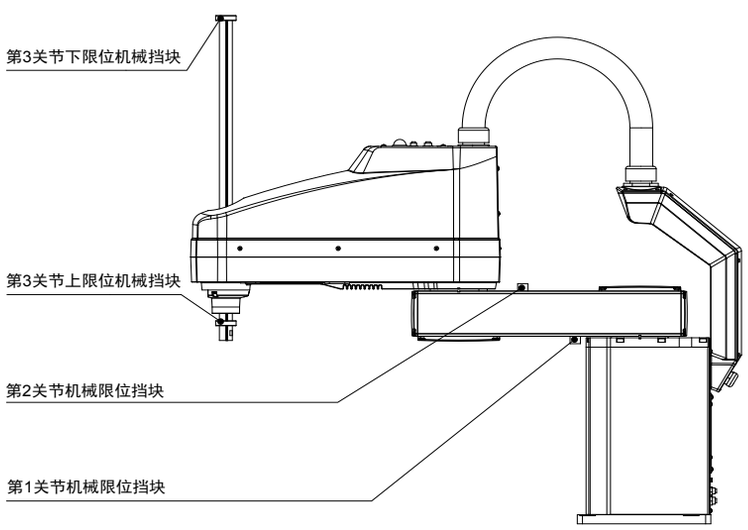


图 1-7 IRS100-20 系列机械挡块位置示意图

2） 第 3 关节基于机械挡块的设定

对于 SCARA 机型的运动范围调节，其具体步骤如下：

- 打开控制柜的电源，将电机设置为非使能状态。
- 在按住制动解除开关的同时，移动 J3 轴至新的下极限位置。

注 意

◆ 按下制动解除开关时，请用手撑住第 3 关节，防止因末端夹具的自重而产生下垂。
◆ 请注意将第 3 关节上升到上限后，机械臂上外罩将难以拆下。

■ 断开控制柜的电源。

■ 拧松下限机械挡块螺丝 (IRS100 系列：M3×10 /IRS100-20 系列：M4×16)。

■ 将下限机械挡块降低到想要限制的行程部分。比如“150mm”行程时，下限 Z 坐标值 为“-150”，要将其设为“-130”时，将下限机械挡块降低“20mm”。请在用游标卡尺等测量距离的同时降低下限机械挡块。

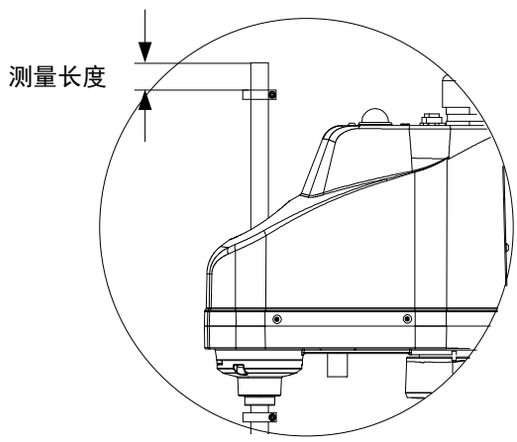


图 1-8 第 3 关节机械挡块设定

■ 紧固下限机械挡块螺丝 (IRS100 系列：M3×10 /IRS100-20 系列：M4×16)，建议紧固扭矩：245N.cm

■ 打开控制柜的电源，重新设置运动范围，软限位应在机械限位之内。

3 利用示教器自由设置空间中的干涉区

1） 示教参数设置

寸动：这里设置自定义的寸动步长，在选用  时适用。

示教速度：示教过程中的最大速度。

示教实际速度 = 示教速度 X 工具栏设定的速度百分比。

示教加速度：示教过程中的最大加速度。

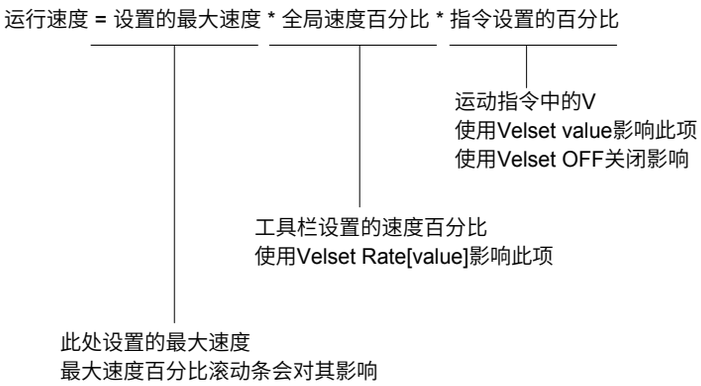
NOTE

◆ “示教参数”是指操作机器人在各坐标系下移动的参数。程序单步运行、连续运行或再现运行时使用的参数是：运行速度、加速度、停止速度！

2） 运行参数设置

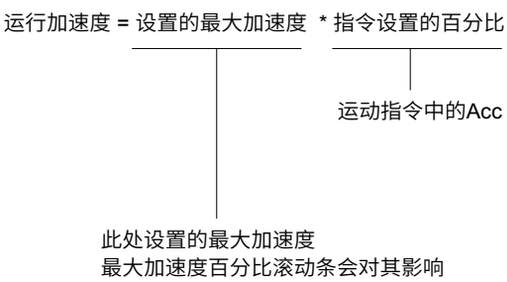
运行速度：运行过程中的最大速度。

运行实际速度由下式计算得到：



运行加速度：运行过程中的最大加速度。

运行实际加速度由下式计算得到：



停止减速度：暂停或停止时的减速度。

NOTE

◆ “运行速度”、“运行加速度”和“停止减速度”参数的设置受伺服系统性能的制约。

过渡精度：相邻两条运动指令之间的过渡单位长度。

圆弧插补：支持关节插补（姿态变化不与圆心角相关）和姿态插补（姿态变化与圆心角相关）。默认“关节插补”。

NOTE

◆ 圆弧姿态插补类型只针对六轴机器人有效。

3） 轴参数设置

轴极限：各轴的极限位置。出于安全考虑，请务必将轴极限设置在机械挡块范围以内。

跟随误差：由机器人的指令加速度和伺服的刚性等级决定。当出现跟随误差过大报警时，应适当降低加速度或增大跟随误差设定值。

到位误差：决定机器人规划完成与伺服实际到位的判定条件。

NOTE

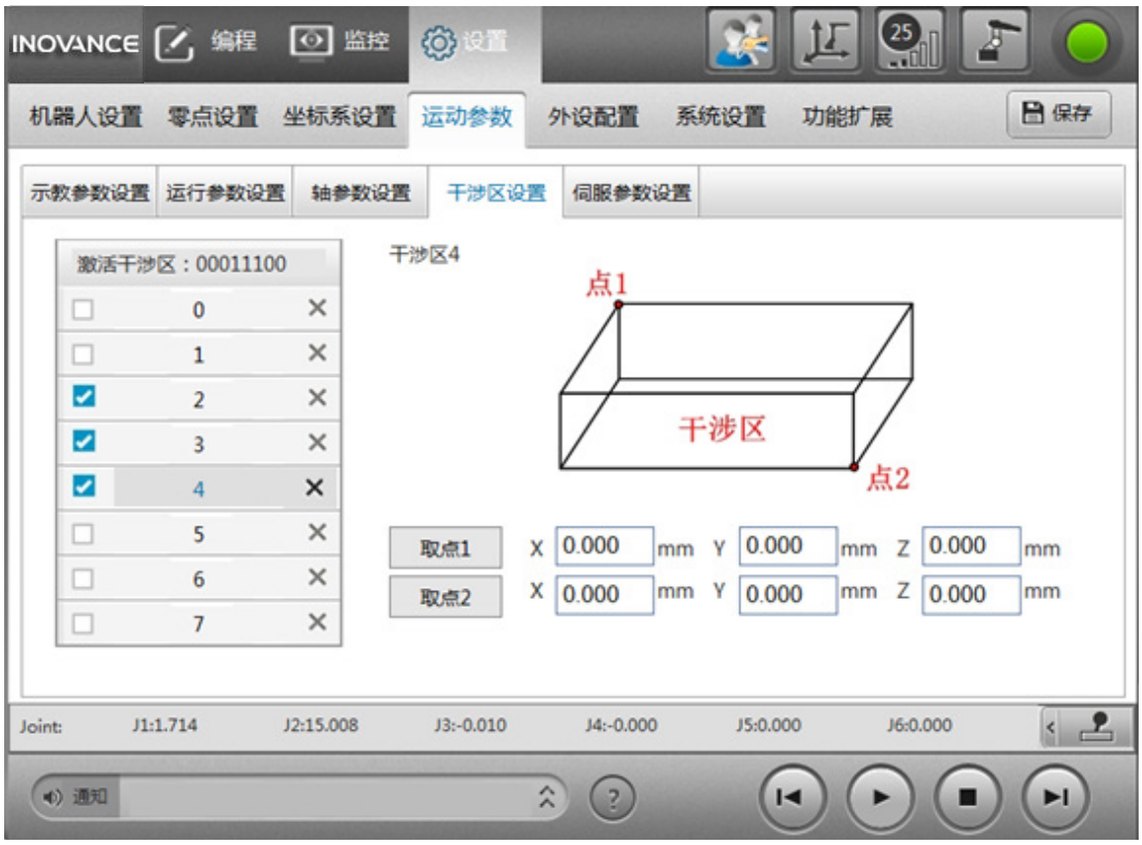
◆ “到位误差”和“跟随误差”参数需要在厂家模式下才能调整。

4） 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，机器人工具末端进入干涉区会产生报警。可同时激活多个干涉区。

编辑干涉区：选中数字，右侧显示干涉区值，编辑干涉区。

激活干涉区：勾选干涉区前面的复选框，激活干涉区。



1.2.4 系统空间要求

如果在机器人本体上安装末端夹具并进行动作，可能会因末端夹具的外径、工件的大小或机械臂的位置等导致与机器人本体接触。因此，在进行系统布局时，务必注意末端夹具的干涉区域，避免与机器人本体接触。详细空间范围请参考本手册“4 汇川机器人 SCARA 本体”章节，或者《IRS100-20 系列 SCARA 机器人用户手册 - 机械篇》和《IRS100 系列 SCARA 机器人用户手册 - 机械篇》，其中，机器人运动范围示意图如下：

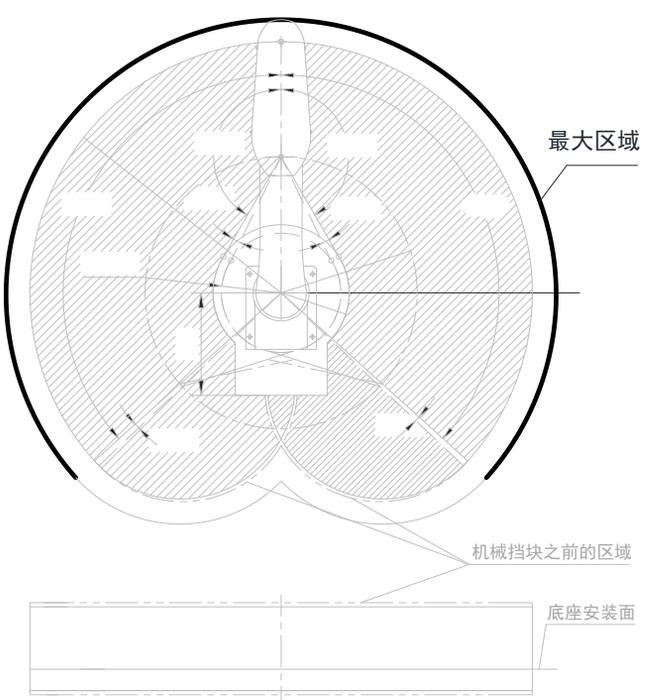


图 1-9 机器人运动范围示意图

图中所示的“最大区域”表示末端夹具半径为 60mm（IRS100 系列）/68mm（IRS100-20）以下的状况。末端夹具半径超过 60mm 时，请将该半径设为与最大区域外缘之间的距离。

除了末端夹具之外，机械臂上安装的相机或电磁阀等较大时，设定最大区域时需考虑相机或电磁阀的空间。

除了机器人、控制柜与外围装置等安装所需的面积之外，请确保下述最低所需限度的空间。

- 示教用空间；
- 维护、检查用空间；
- 线缆用空间。

在线缆的固定状态下，电源线缆与信号线缆的最小弯曲半径为 70mm。安装机器人时，请注意与障碍物之间的距离。请确保留出足够的弯曲半径空间，避免线缆极度弯曲造成线缆破损或折断。

请在最大区域与安全防护栏之间确保最低 100mm 宽的空间。

注 意

◆ 部分机型的第 3 关节有一部分运动区域在底座安装面以下（详细请参见各机型运动范围），在设计台架高度时，需参考区域大小做出相应调整，以避免夹具在运动过程中触碰到安装面导致损坏。

第 2 章 安装

2.1 安装流程

表 2-1 可以用来检查机器人的安装进度。建议把该表格复制一份，并在每个项目确认就绪时，填写“完成”一栏。

表 2-1 安装流程

步骤	操作	完成
1 机器人安装前准备		
（1）	具备专业操作资质的安装人员，请参考 2.2.1 小节。	
（2）	符合产品使用要求的安装环境，请参考 2.2.2 小节。	
（3）	按照要求，自行完成台架制作，请参考 2.2.3 小节。	
2 机器人安装空间		
准备符合机器人作业要求的安装空间，请参考 2.3 小节。		
3 开箱与搬运		
（1）	拆除机器人外包装，请参考 2.4.1 小节。	
（2）	核对装箱清单，请参考 2.4.2 小节。	
（3）	将机器人搬运到安装作业点，请参考 2.4.3 小节。	
4 安装机器人本体		
（1）	把机器人本地固定在安装板上	
5 线缆连接		
（1）	连接本体和控制柜之间的电源线缆和信号线缆	
6 用户配线 / 配管		
（1）	连接 I/O 信号线缆和气管	

2.2 机器人安装前准备

2.2.1 安装人员

安装人员须事先获知机械学知识或接受机械学培训，进而了解安装过程中各种危险情况。

2.2.2 安装环境

请将机器人系统设置在符合下述条件的环境中，以便发挥和维持本机的性能并安全的进行使用。

安装在室内通风良好的场所；

请勿在封闭环境中使用，封闭环境容易导致控制柜高温，缩短使用寿命；

避免阳光直射；

请勿在有硫化氢、氯气、氨、硫磺、氯化性气体、酸、碱、盐等腐蚀性及易燃性气体环境中使用本产品；

请勿在可燃物等附近使用本产品；

在有磨削液、油雾、铁粉、切削等的场所使用时候，请做好防护措施；

附近没有大型逆变器、大功率高频发生器、大型接触器、焊接机等电干扰源。

1 环境条件

表 2-2 环境要求 [1]

项目	条件
环境温度	0 ～ 40° C（不应有过大温度变化）
环境相对湿度	10 ～ 80%（不得结露）
电快速瞬变脉冲群抗扰度 [2]	4 kV 以下（电源线） 2 kV 以下（信号线）
静电抗扰度 [2]	6 kV 以下（接触） 8 kV 以下（空气）
环境 [1]	▯ 避免装于阳光直射的地方 ▯ 避免装于含盐分、潮湿等易生锈的地方 ▯ 无灰尘、油烟、铁屑等粉尘污染 ▯ 无易燃性、腐蚀性液体与气体，无易爆性气体 ▯ 不传递冲击与振动 ▯ 附近没有电气干扰源

[1] ：机器人不适合在易爆型喷涂作业等恶劣环境下使用。如果在不符合上述条件的场所中使用，请与销售商联系。

[2] ：根据 IEC 61800-3： 2017 标准进行电磁兼容性测试。

2 特殊环境条件

1） 机器人的表面具有一般的耐油性，若使用过程中可能会沾染特殊油时，需要事先确认，请与销售商联系。

2） 如果在温度与湿度变化较大的环境中使用，机器人内部可能会结露。

3） 直接搬运食品时，需要确认机器人有无导致食品污损的可能性。请与销售商联系。

NOTE

◆ 如果机器人在使用过程中沾染了油渍，请勿用酒精或苯等清洁液体用力擦拭机器人，否则可能会导致涂装面光泽度降低。

2.2.3 台架制作与安装要求

台架的形状和大小因机器人系统的用途而异。

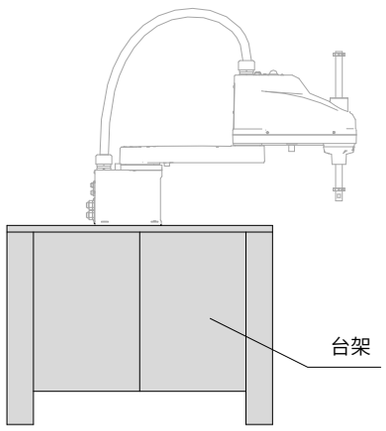


图 2-10 台架示意图

请客户自行制作用于固定机器人的台架，下面列出了台架需要的条件，请在设计台架前阅读参考。

1） 必须能承受机器人的重量；

2） 必须能承受机器人以最大加速度进行动作时的动态作用力；

如下所示为机器人动作产生的转矩与反作用力：

类 型	IRB100-3-40Z*	IRB100-6-60Z*	IRB100-20*
水平面最大转矩	250N·m	350N·m	1000N.m
水平方向最大反作用力	1000N	1500N	7500N
垂直方向最大反作用力	1000N	1500N	2000N

3） 建议连接横梁等加固材料，加强台架的强度。

4） 为了抑制振动，建议使用厚度大于 20mm、表面粗糙度小于 25μm 的钢板作为机器人的安装面板。

5） 安装台架时，请保持机器人的安装面与地面平行（使用水平仪确保台架水平）。

6） 请使用直径大于 M16 的螺钉将台架固定在地面，并且确保台架已紧固安装。

7） 如需通过台架上的开孔位，穿过机器人本体与控制柜的连接线缆，要求孔位的开孔直径不能小于 60mm。

8） 请勿从机器人主体上拆下 M/C 线缆。

9） 有关在台架中存放控制柜时的环境条件（空间条件），请参阅 IRCB300 系列 4 轴控制柜手册。

2.3 安装空间要求

2.3.1 机器人本体安装空间

详细空间范围请参考《IRS100-20 系列 SCARA 机器人用户手册 - 机械篇》和《IRS100 系列 SCARA 机器人用户手册 - 机械篇》。

2.3.2 控制柜安装空间

控制柜安装空间要求如下图所示，需保证足够的散热空间。

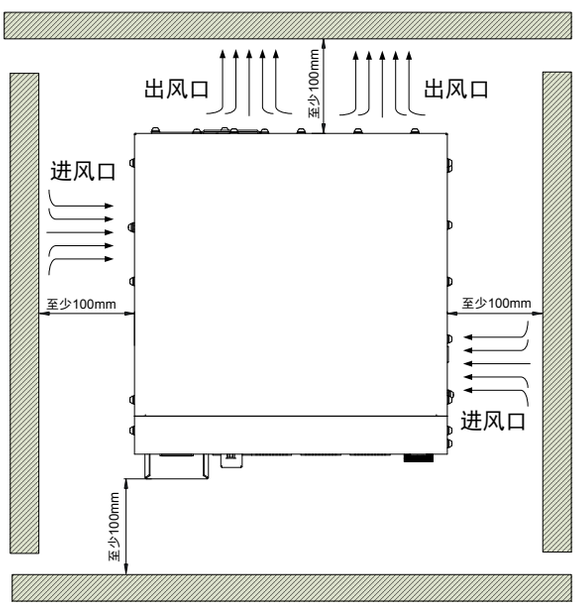


图 2-11 卧式安装空间要求

2.3.3 其他空间

除了机器人、控制柜与外围装置等安装所需的面积之外，请确保下述最低所需限度的空间。

- 示教用空间；
- 维护、检查用空间；
- 线缆用空间。

2.4 开箱

开箱准备：羊角锤一把，M8 外六角扳手、十字螺丝刀、一字螺钉刀一把、剪刀 / 美工刀一把、防护手套一双。

2.4.1 开箱步骤

1） 步骤 1. 拆外包装

请戴好防护手套，用羊角锤或一字螺丝刀将钢舌片①撬至钢扣位②，用锤子将钢带③取下，取下箱盖板和箱体的四块侧板。

工具：防护手套、一字螺丝刀、羊角锤。

NOTE

◆ 推荐使用羊角锤开箱，更加安全。
◆ 钢舌片比较锋利，戴手套操作，以防划伤。

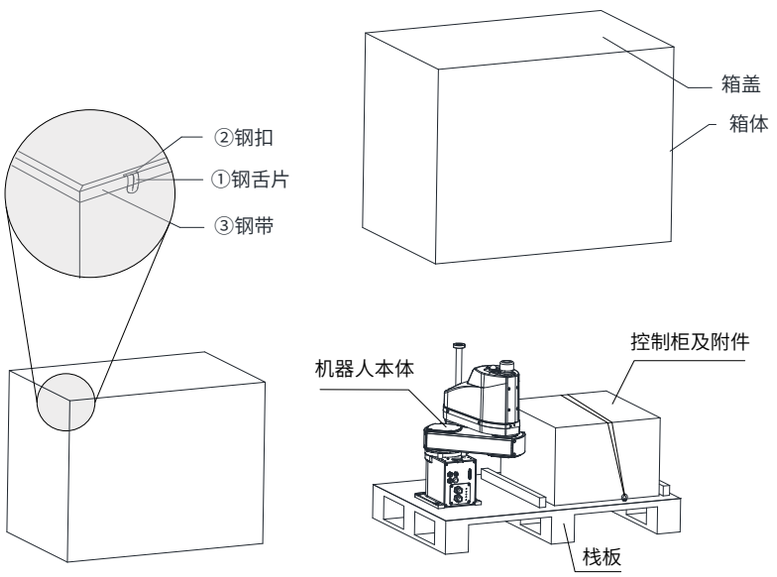


图 2-12 IRS100 系列外包装结构示意图

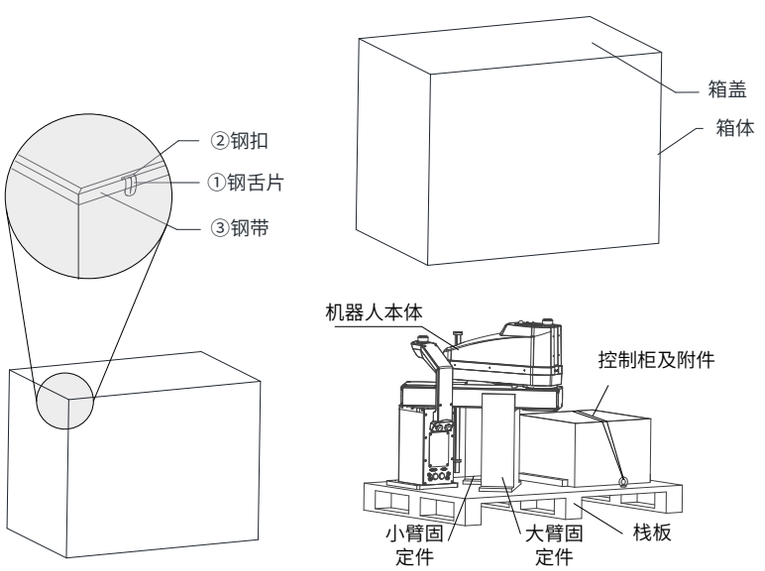


图 2-13 IRS100-20 系列外包装结构示意图

2） 步骤 2. 拆控制柜外包装（以 IRS100 系列为例）

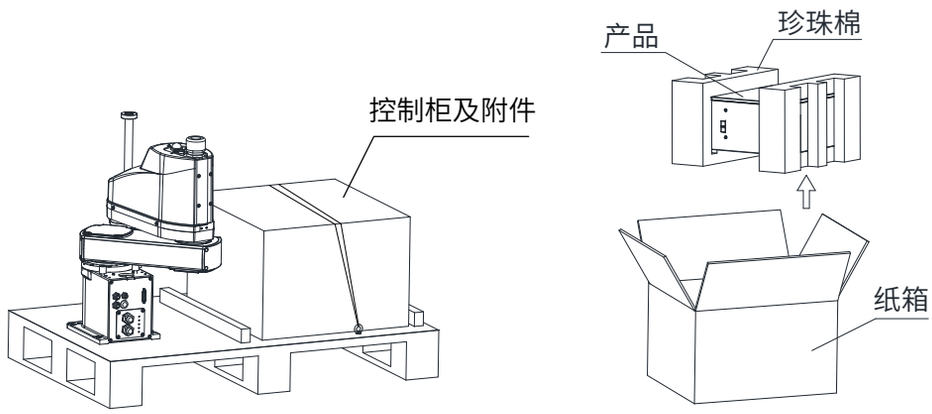
■ 用剪刀剪去固定控制柜包装纸箱的打包带；

■ 将控制柜包装箱从包装底座上搬移至空旷地面，注意包装箱的朝向指示；

■ 用剪刀划开包装箱表面的封箱胶带后，打开纸箱；

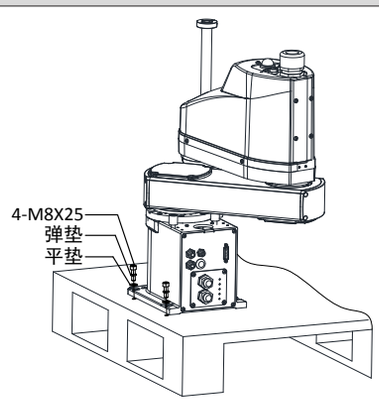
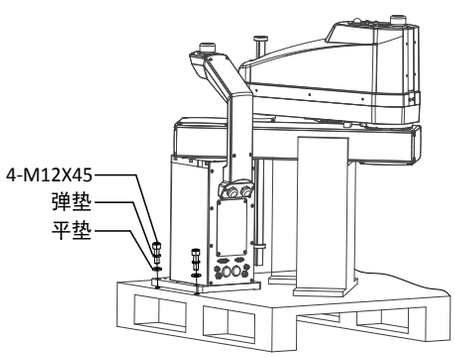
■ 抓稳控制柜两侧的手提孔或托住底部，将控制柜从包装箱内取出，搬运到目标位置。

工具：剪刀 / 美工刀。



3） 步骤 3：使用 M8 外六角扳手，取下连接机器人本体和包装基座的固定螺栓。

工具：M8 外六角扳手。

IRS100 系列的固定螺栓规格为 4-M8X25	IRS100-20 系列的固定螺栓规格为 4-M12X45
	

* 注意：拆卸本体固定螺栓时，必须先由一人固定住设备，另外一人进行拆卸，避免拆卸固定底座时，设备因重心不平衡而倾斜，造成人身伤害或设备损坏。

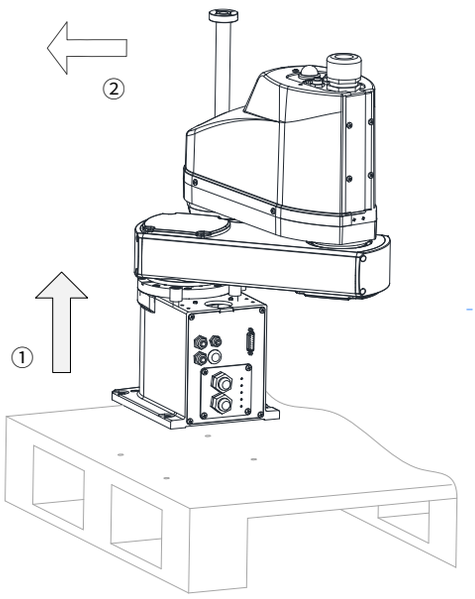
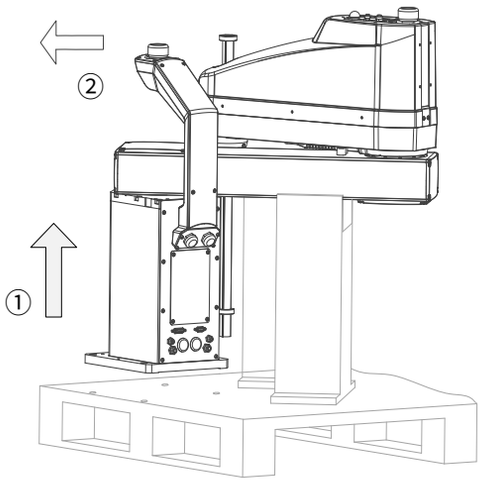
警 告

◆ 拆卸本体固定螺栓时，必须先由一人固定住设备，另外一人进行拆卸，避免拆卸固定底座时，设备因重心不平衡而倾斜，造成人身伤害或设备损坏。

4） 步骤 4. 取出机器人本体

①将机器自下而上托起，②使机身底部与固定件顶部不干涉后平行取出，再搬运到目标位置。

注意：小心取出机器，避免发生磕碰。

IRS100 系列	IRS100-20 系列
	

注 意

- 如需重新包装，请参考上述步骤反向操作即可，不再赘述。
- 请戴好防护手套以防划伤，小心作业以防碰伤机器。
- 小心取出机器，避免发生磕碰。

2.4.2 核对装箱清单

开箱后，请根据装箱清单确认产品状态及构成种类。

表 2-3 表 装箱清单

物料名称	数量（PCS）
机器人本体	1
机器人控制柜	1
《汇川技术产品保修卡》	1
《机器人本体用户手册》	1
《机器人控制柜用户手册》	1
成套电缆	1
四轴控制柜安装挡板组件 (ROHS)	1
内六角螺栓 M8*30	4
弹垫 8	4
垫圈 8	4

2.5 搬运

危 险

◆ 请由具有资格的作业人员进行司索、起重机起吊作业或叉车驾驶等搬运作业，否则可能造成重伤或重大损害。

警 告

◆ 请尽可能在原包装状态下用吊车和叉车等进行搬运。
◆ 使用吊车、起重机等搬运设备时，作业者需穿戴个人防护装置，搬运路线周围禁止人员站立或停留。
◆ 吊起设备时，请用手扶住以确保平衡，起吊不稳可能会导致设备掉落，造成重伤或重大损害。

注 意

◆ 请按照设备的储存与运输条件进行储存与运输，储存温度、湿度满足要求。
◆ 避免在水溅雨淋、阳光直射、强电场、强磁场、强烈振动等场所储存与运输。
◆ 请将设备进行严格包装后再进行车辆运输，长途运输时必须使用封闭的箱体。
◆ 严禁将本设备与可能对本设备构成影响或损害的设备或物品一起混装运输。
◆ 长期保管后的设备再次组装到系统中使用时，请进行试运转，确认没有异常之后切换为正常运转。
◆ 如果设备在运输或保管期间产生结露，请在消除结露之后再打开电源。

2.5.1 搬运准备

手套、防砸鞋、叉车等，至少需要两人

2.5.2 搬运步骤

* 注意：请穿上防砸鞋、戴上手套进行搬运。

1 搬运控制柜：

1） 搬运人员可双手徒手搬运。注意：请注意抓牢后搬运，避免控制柜滑落受损。

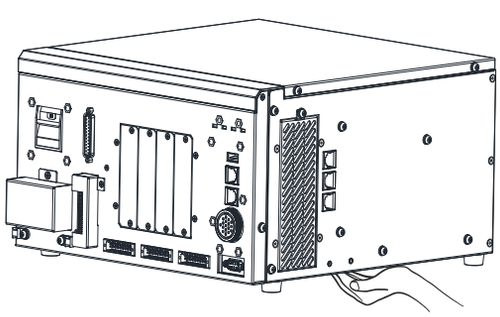


图 2-14 IRCB300 系列控制柜

2 搬运机器人本体

危 险

◆ 请由具有资格的作业人员进行司索、起重机起吊作业或叉车驾驶等搬运作业，否则可能造成重伤或重大损害。

警 告

◆ 请尽可能在原包装状态下用吊车和叉车等进行搬运。
◆ 使用吊车、起重机等搬运设备时，作业者需穿戴个人防护装置，搬运路线周围禁止人员站立或停留。
◆ 吊起设备时，请用手扶住以确保平衡，起吊不稳可能会导致设备掉落，造成重伤或重大损害。

注 意

◆ 请按照设备的储存与运输条件进行储存与运输，储存温度、湿度满足要求。
◆ 避免在水溅雨淋、阳光直射、强电场、强磁场、强烈振动等场所储存与运输。
◆ 请将设备进行严格包装后再进行车辆运输，长途运输时必须使用封闭的箱体。
◆ 严禁将本设备与可能对本设备构成影响或损害的设备或物品一起混装运输。
◆ 长期保管后的设备再次组装到系统中使用时，请进行试运转，确认没有异常之后切换为正常运转。
◆ 如果设备在运输或保管期间产生结露，请在消除结露之后再打开电源。

搬运准备：手套、防砸鞋、叉车等，至少需要两人。

1） 按出厂姿态定位机器人：

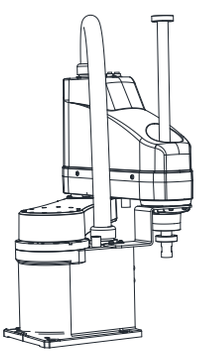


图 2-1 机器人出厂姿态（示例）

2） 断开所有装置的电源；

3） 拔下与控制柜连接的电源线缆和信号线缆；

4） 拧下底座安装固定螺钉，从安装台上拆下机器人；

5） 将机器人固定至搬运器具；

NOTE

◆ 徒手搬运机器人时，请将机器人固定在搬运器具上，或用手托住阴影部分（第 2 机械臂和底座底部）。请务必由 2 人及 2 人以上进行搬运作业。

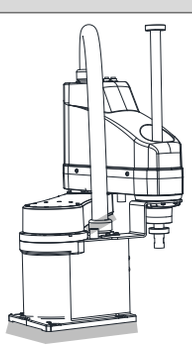
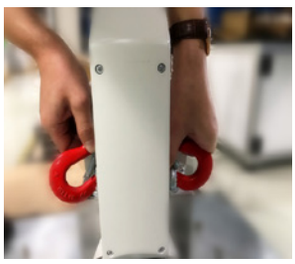


图 2-2 徒手搬运位置

6） 采用吊装方式将机器人本体抬放到安装台架上。详细步骤如下：

步骤 1: 机器人本体基座上有两个用吊装搬运的吊环，将吊装绳索①的挂钩挂在两侧吊环，确保挂钩可靠，如下图:



步骤 2：吊装绳索②穿过机器人小臂部位后，绳索两头挂扣在缆绳①的挂钩上，该挂钩带防松功能，如下图：

完成吊装绳索连接后，示意图如下：

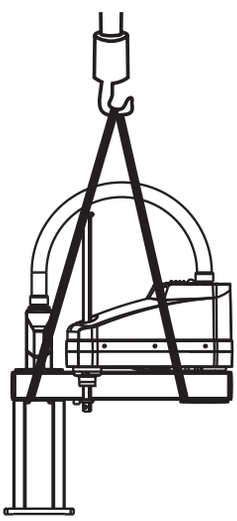


图 2-15 绳索连接机器人本体和行吊的连接装置（示意）

步骤 3：两人配合完成移动。其中一人轻扶住机器人本体以免移动过程出现晃动碰撞；另一人操作吊车，缓慢将机器人本体升起后，将机器人搬运到安装台架上。

NOTE

- ◆ 不要拖拽本体线缆，以免损坏机器。
- ◆ 建议使用扁平吊带，吊带长度为 3m，使用时请确认扁平吊带无开线或断开处，并承重负载力不小于 100kg；
- ◆ 请注意穿戴好防护用品，并确保吊装区域内足够安全，以避免碰撞危险。

2.6 安装机器人本体

危 险

- ◆ 严禁非专业人员进行设备安装、接线、保养维护、检查或部件更换！
- ◆ 请务必对系统安装安全护栏，否则可能造成严重的安全问题。
- ◆ 安装系统时，请勿与周围的建筑物、结构件或设备等产生干扰，否则可能会因工具或工件撞到外围设备造成重伤或重大损害。
- ◆ 接通电源或操作系统前需对机器人本体进行固定，否则可能导致机器人本体翻倒，造成重伤或重大损害。

警 告

- ◆ 严禁改装本设备！
- ◆ 请勿在强电场或强电磁波干扰的场所安装本设备！
- ◆ 拆卸机器人本体的安装螺钉，请扶住机器人本体防止翻倒。
- ◆ 将机器人本体安装到墙面上时，请支撑机器人本体后拧紧底座固定螺钉。底座固定螺钉未完全紧固时移开支撑，可能会导致机器人掉落，非常危险。

2.6.1 固定机器人底座

■ IRS100 系列

用 4 个 M8X25 的螺钉将底座固定到台架上。

请使用强度相当于 GB/T 3098.1 性能等级为 10.9 或 12.9 级的螺钉 。

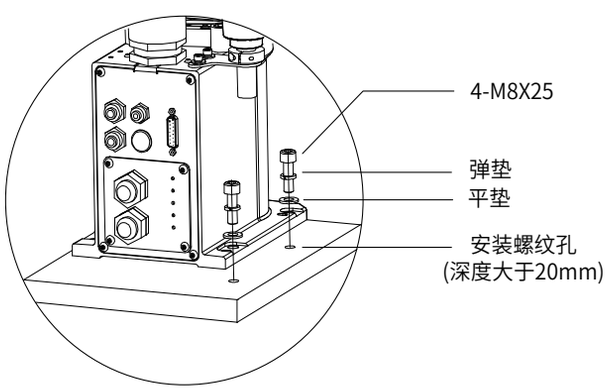


图 2-16 图 底座安装示意图

■ IRS100-20 系列

用 4 个 M12X45 的螺钉将底座固定到台架上。

请使用强度相当于 GB/T 3098.1 性能等级为 10.9 或 12.9 级的螺钉 。

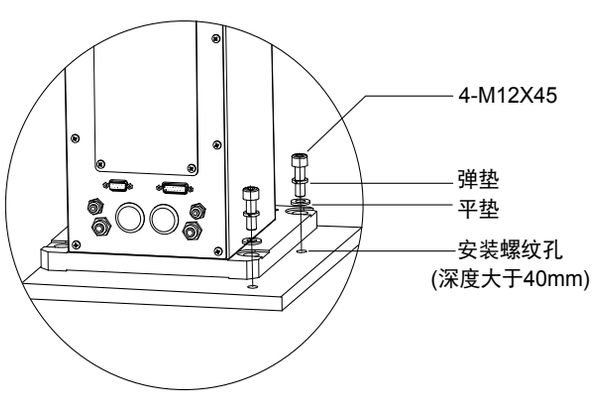


图 2-17 图 底座安装示意图

2.6.2 安装后确认

请参考如下力矩推荐值，检查固定螺栓是否紧固：

产品系列	螺栓公称直径（强度 10.9 以上）	力矩
IRS100 系列	M8	30N.m
IRS100-20 系列	M12	130N.m

2.7 安装控制柜

2.7.1 安装位置

标准安装场景，台面稳定放置即可，要求台面平整，不得倾斜、变形。

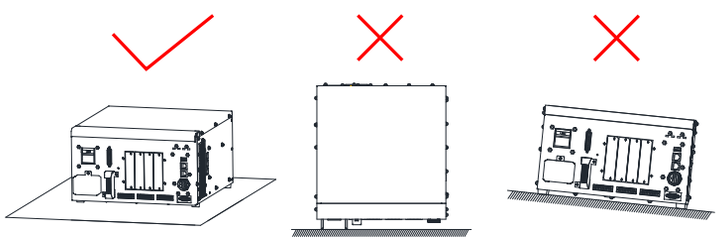


图 2-18 安装平面要求

2.7.2 安装方式与尺寸

控制柜支持卧式安装方式和立式安装方式。

1 卧式安装

用附件自带支架与螺钉将机箱固定在台面上，如下图所示：

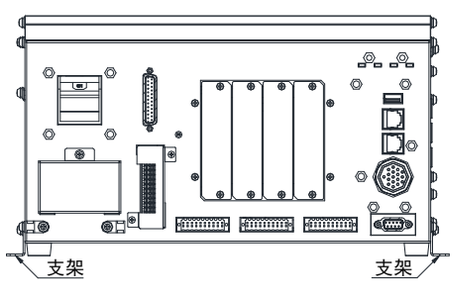


图 2-19 卧式安装

安装固定尺寸如下：

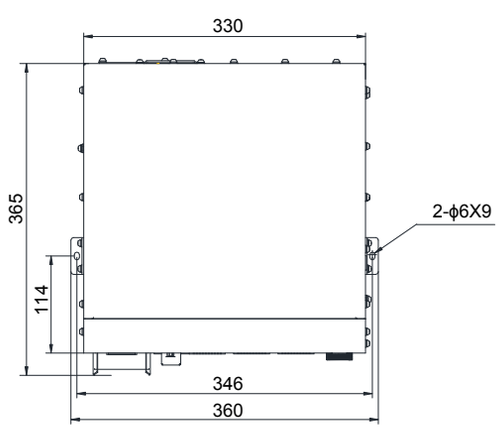


图 2-20 卧式安装尺寸（单位：mm）

两侧各增加一处安装支架，支架分别固定在两侧 Ø4 孔位处，用 M4X10 螺钉与机箱固定，用 M5X10 螺钉将支架固定在安装面上。

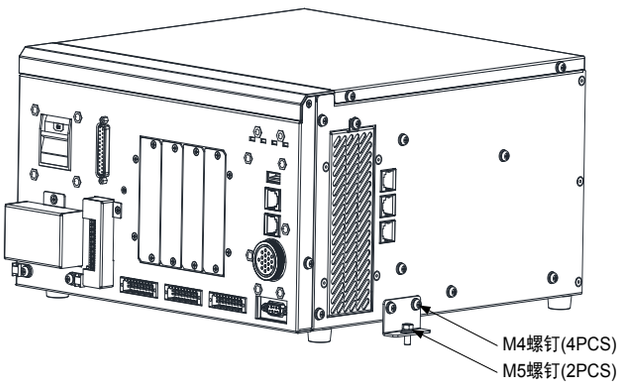


图 2-21　卧式支架安装示意图

2 立式安装

用附件自带支架与螺钉将机箱固定在台面上，如下图所示：

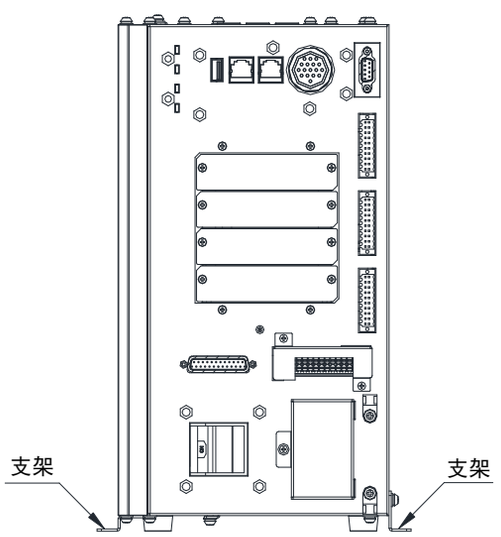


图 2-22 立式安装

安装固定尺寸如下：

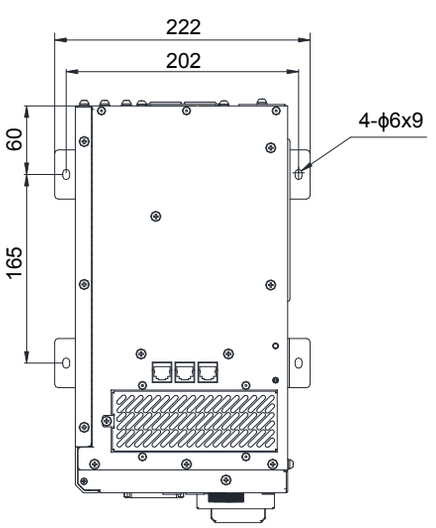


图 2-23 立式安装尺寸（单位：mm）

两侧各增加一处安装支架，支架分别固定在两侧 Ø4 孔位处，用 M4X10 螺钉与机箱固定，用 M5X10 螺钉将支架固定在安装面上。

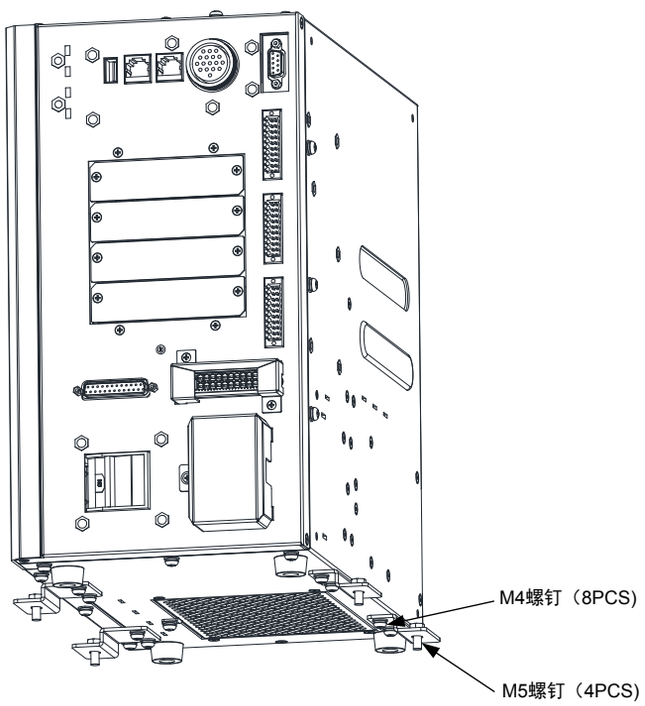


图 2-24 立式支架安装

第 3 章 系统连接

3.1 典型系统应用接线

详细请参见本章节插页图——典型系统应用连线图。

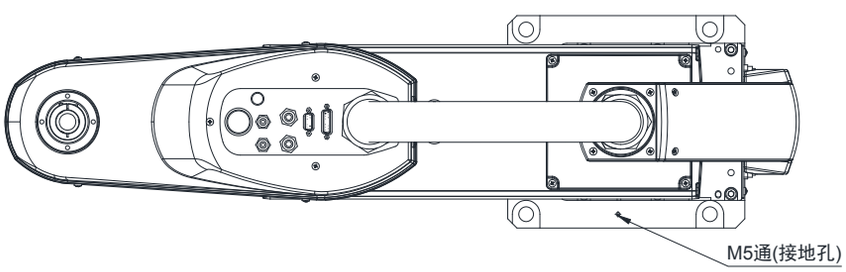
3.2 接地要求

警 告

◆ 为了防止触电，请务必将机器人及控制器进行接地。
◆ 请在切断控制器电源后进行接地作业。

1 IRB100-20 系列：

为提高系统抗干扰能力，请确保将机器人本地进行可靠接地。本地自带接地孔（如下）



请使用导体横截面积 2.0mm^2 以上，且长度在 1m 以内的地线。

3.3 电源连接

本产品采用单相 220VAC 电源，请按照如下示意图进行电源线的制作，并正确接线。

出厂时我司已配有相应线缆，如客户自选线缆，需符合相应安全规范，选定额定电流 10A 以上。

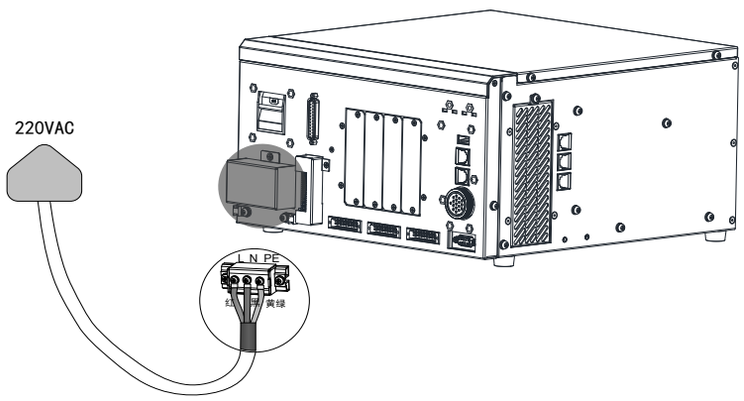


图 3-25 控制柜电源连接示意图

3.4 控制柜连接机器人本体

危 险

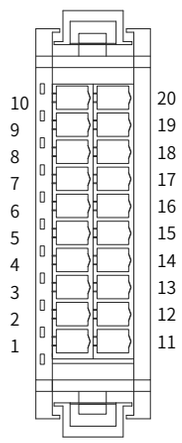
- 严禁非专业人员进行设备安装、接线、保养维护、检查或部件更换！
- 请务必断开电源后进行接线作业，否则可能会有触电的危险或导致系统故障。
- 接线前，请切断所有设备的电源。切断电源后设备内部电容有残余电压，请至少等待 10 分钟再进行接线等操作。
- 接线时，请务必保证紧急停止开关和安全门等安全相关输入信号正确接入，否则系统发生紧急状况时将无法发挥安全功能，可能会导致重伤或重大损害。
- 请务必保证设备的良好接地，否则会有电击的危险。
- 请遵守静电防止措施（ESD）规定的步骤，并佩戴静电手环进行接线等操作，避免损坏设备内部的电路。

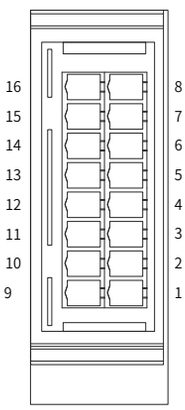
警 告

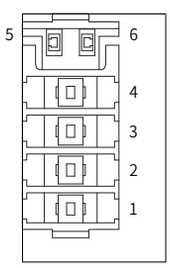
- 请将线缆连接牢固。请勿在线缆上放置重物，请勿强行弯曲或拉拽线缆，否则可能造成线缆损坏、断线或接触不良，有触电的危险或导致系统故障。
- 接线时使用到的线缆必须符合相应的线径和屏蔽等要求，使用屏蔽线缆时屏蔽层需要单端可靠接地！
- 接线时请勿弄错连接关系，否则系统将无法正常工作，还可能造成安全问题。
- 接线完成后，请确保设备内部没有掉落的螺钉或裸露线缆。

3.4.1 端口定义

1 动力线连接端口

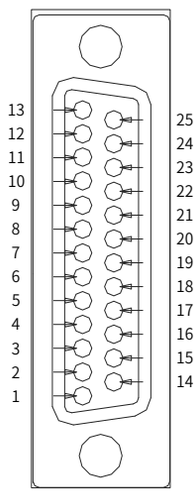
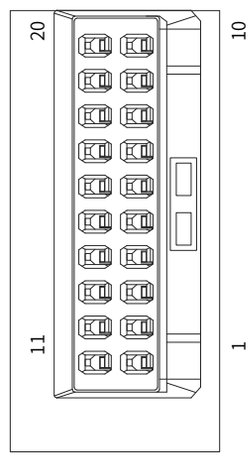
<table>
<tr><th colspan="3">控制柜侧</th><th colspan="4">机器人本体侧及电机基座侧</th></tr>
<tr><th>端子针脚分布图</th><th colspan="2">定义</th><th>端子针脚分布图</th><th></th><th>针脚号</th><th>定义</th></tr>
<tr><td rowspan="20"> 

控制柜侧动力
端子针脚分布</td><td>1</td><td>J1：U</td><td rowspan="20"> 

机器人本体
端子针脚分布
 

基座电机
端子分布</td><td rowspan="14">机器人本体侧动力端子针脚</td><td>1</td><td>J2：U</td></tr>
<tr><td>2</td><td>J1：V</td><td>2</td><td>J2：V</td></tr>
<tr><td>3</td><td>J1：W</td><td>3</td><td>J2：W</td></tr>
<tr><td>4</td><td>J1：PE</td><td>4</td><td>J2：PE</td></tr>
<tr><td>5</td><td>J2：U</td><td>5</td><td>J3：U</td></tr>
<tr><td>6</td><td>J2：V</td><td>6</td><td>J3：V</td></tr>
<tr><td>7</td><td>J2：W</td><td>7</td><td>J3：W</td></tr>
<tr><td>8</td><td>J2：PE</td><td>8</td><td>J3：BK+</td></tr>
<tr><td>9</td><td>J3：U</td><td>9</td><td>J3：BK-</td></tr>
<tr><td>10</td><td>J3：V</td><td>10</td><td>J3：PE</td></tr>
<tr><td>11</td><td>J3：W</td><td>11</td><td>J4：U</td></tr>
<tr><td>12</td><td>J3：BK+</td><td>12</td><td>J4：V</td></tr>
<tr><td>13</td><td>J3：BK-</td><td>13</td><td>J4：W</td></tr>
<tr><td>14</td><td>J3：PE</td><td>14</td><td>J4：PE</td></tr>
<tr><td>15</td><td>J4：U</td><td></td><td></td></tr>
<tr><td>16</td><td>J4：V</td><td></td><td></td></tr>
<tr><td>17</td><td>J4：W</td><td rowspan="6">基座电机动力端子针脚</td><td>1</td><td>J1：PE</td></tr>
<tr><td>18</td><td>J4：PE</td><td>2</td><td>J1：W</td></tr>
<tr><td>19</td><td>PE</td><td>3</td><td>J1：V</td></tr>
<tr><td>20</td><td>GND</td><td>4</td><td>J1：U</td></tr>
</table>

2 编码器线缆连接端口

控制柜侧			机器人本体侧		
端子针脚分布图	定义		端子针脚分布图	定义	
 控制柜侧编码器端子针脚分布	1	J1：PS+	 机器人本体侧编码器端子针脚分布	1	J1：PS+
	2	J1：PS-		2	J1：PS-
	3	J1：5V		3	J1：5V
	4	J1：GND		4	J1：GND
	5	24V			
	6	J2：PS+		5	J2：PS+
	7	J2：PS-		6	J2：PS-
	8	J2：5V		7	J2：5V
	9	J2：GND		8	J2：GND
	10	GND			
	11	J3：PS+		9	J3：PS+
	12	J3：PS-		10	J3：PS-
	13	J3：5V		11	J3：5V
	14	J3：GND		12	J3：GND
	15	DI1			
	16	J4：PS+		13	J4：PS+
	17	J4：PS-		14	J4：PS-
	18	J4：5V		15	J4：5V
	19	J4：GND		16	J4：GND
	20	DI2			
	21	LH+		17	LH+
	22	LH-		18	LH-
	23	SB+		19	SB+
	24	SB-		20	SB-
	25	DI4			

3.4.2 连接示意

> **注 意**
>
> ◆ 连接机械本体与控制柜的编码器和动力线缆时，请勿弄错连接关系。否则，机器人系统将无法正常动作，且可能导致严重安全问题。

1 动力线缆连接

第一步：将图 3-27 中控制柜前面板金属防护件拆下；

第二步：将图 3-26 中动力线缆上的 20PIN 连接器插头插入图 3-27 中控制柜前面板动力线缆连接插座。

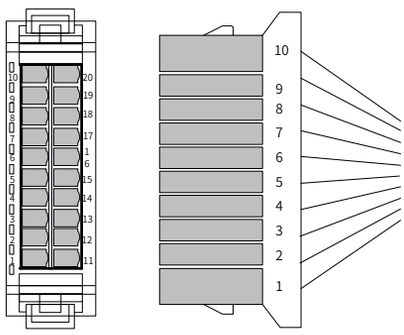


图 3-26 动力线缆上的 20PIN 连接器插头

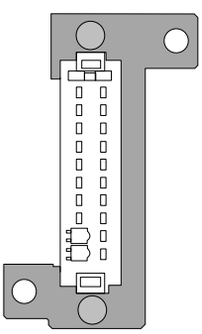


图 3-27 控制柜前面板动力线连接插座及金属防护件

第三步: 完成连接后，将第一步中拆下的金属防护件重新装回，并使用螺丝刀将其锁紧（推荐力矩为: 0.55N·m）。

> **注 意**
>
> ◆ 完成动力线缆连接后，请务必将金属防护件装回并拧紧，否则有触电危险！

2 编码器线缆连接

将图 3-27 中编码器线缆上的 DB25 连接器插头，插到图 3-28 中控制柜前面板的 DB25 信号插座，用螺丝刀锁紧 DB 插头上的固定螺钉，推荐力矩为：0.55N·m。

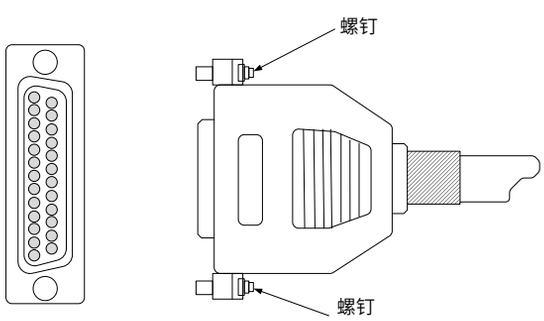


图 3-28 编码器线 DB25 连接器插头

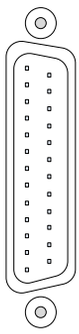


图 3-29 前面板 DB25 信号插座

3 控制柜与机器人本体连接示意

1） IRCB300 系列机器人控制柜连接 IRS100 系列机器人本体

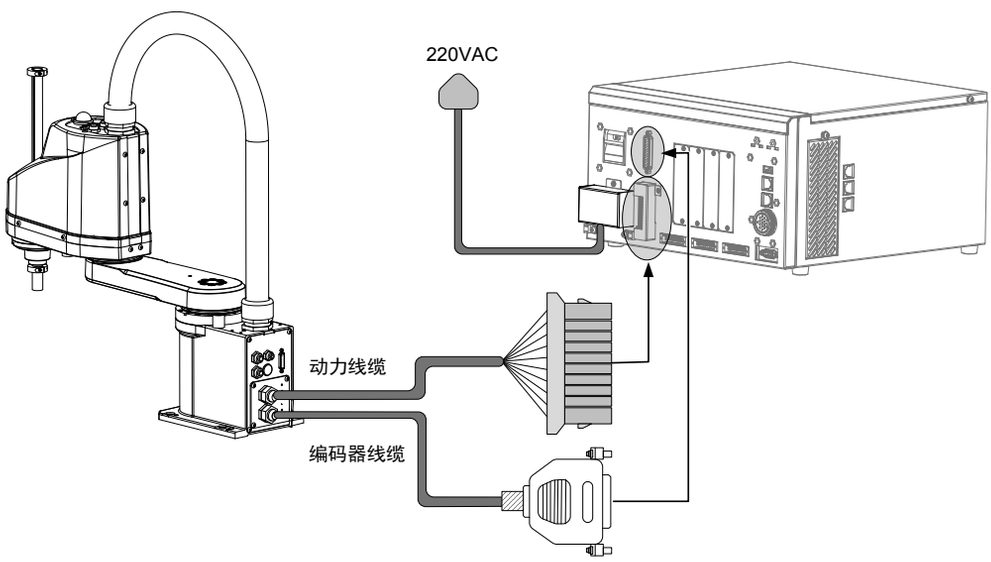


图 3-30 IRCB300 系列机器人控制柜连接 IRS100 系列机器人本体

2） IRCB300 系列机器人控制柜连接 IRS100-20 系列机器人本体

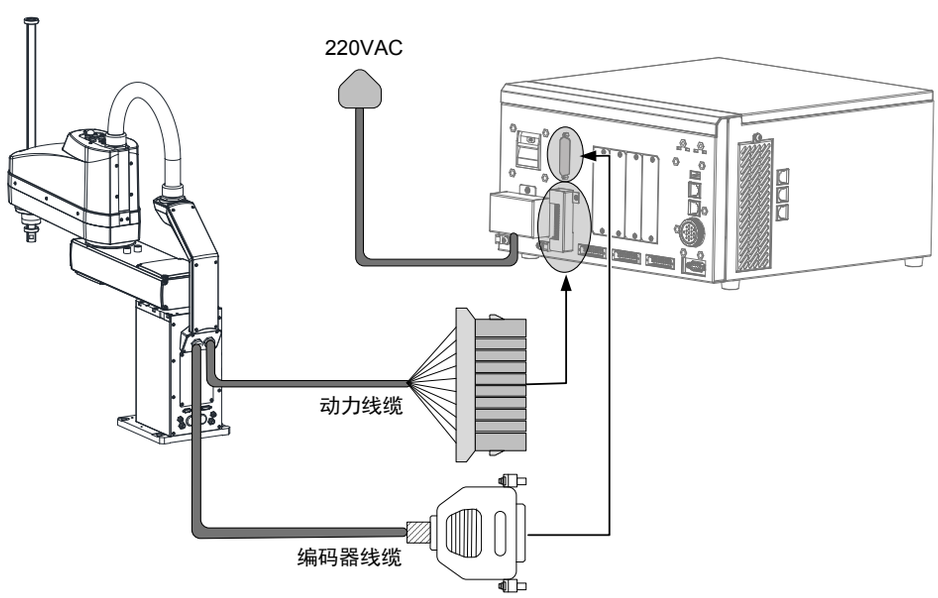
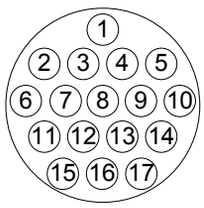


图 3-31 IRCB300 系列机器人控制柜连接 IRS100-20 系列机器人本体

3.5 控制柜连接示教器

3.5.1 示教器端口定义



序号	定义	简介	序号	定义	简介
1	24V	24V 电源	10	24V	24V 电源
2	GND	电源地	11	PE	屏蔽地
3	E-STOP1-A	急停 1-A 端	12	TP_TX+	示教器以太网数据 TX+
4	E-STOP1-B	急停 1-B 端	13	TP_TX-	示教器以太网数据 TX-
5	E-STOP2-A	急停 2-A 端	14	TP_RX+	示教器以太网数据 RX+
6	E-STOP2-B	急停 2-B 端	15	TP_RX-	示教器以太网数据 RX-
7	DMS_NO1	手压开关常开触点 1	16	PE	屏蔽地
8	24V	24V 电源	17	NET_SEL	示教网口选择输入型号
9	DMS_NO2	手压开关常开触点 2			

3.5.2 示教器接线方法

本产品可配备 IRTP80 和 ITP100 系列示教器，进行机器人示教和编程。接线图如下，示教器的详细操作请参见示教器的编程手册。

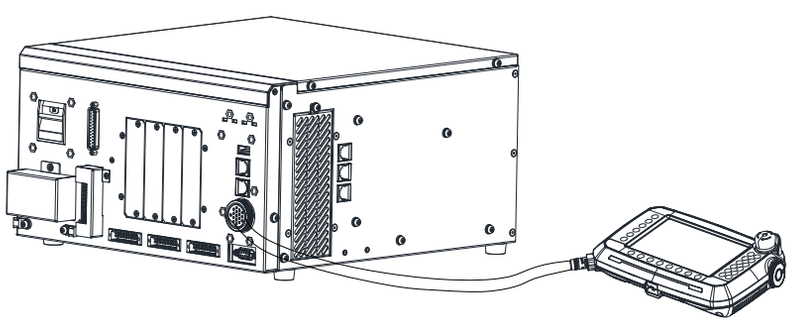


图 3-32 示教器连接示意图

3.6 用户配线 / 配管

危 险

◆ 严禁非专业人员进行设备安装、接线、保养维护、检查或部件更换！

3.6.1 配线（电线）

1 线缆规格

IRS100 系列 /IRS100-20 系列

额定电压	容许电流值	线数	导体公称截面积	备注
AC/DC30V	0.5A	15	24AWG	双绞线

警 告

◆ 请勿通过 0.5A 以上的电流。

线缆		厂家	标准
15pin	适用连接器	CONITONE	G-DB-15M-001 X

注：电缆两端连接器的相同编号针类已配好线。

2 机器人信号线连接表

■ 15Pin 连接器

针脚定义	信号	连接器	NO	连接	NO	连接器	针脚定义
供用户自行定义	用户专用信号线	15 8 9 1 IO (机械臂侧)	1		1	8 15 1 9 IO (基座侧)	供用户自行定义
			2		2		
			3		3		
			4		4		
			5		5		
			6		6		
			7		7		
			8		8		
			9		9		
			10		10		
			11		11		
			12		12		
			13		13		
			14		14		
			15		15		
			外壳		外壳		

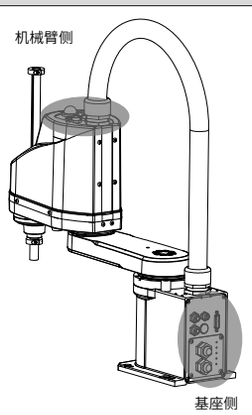
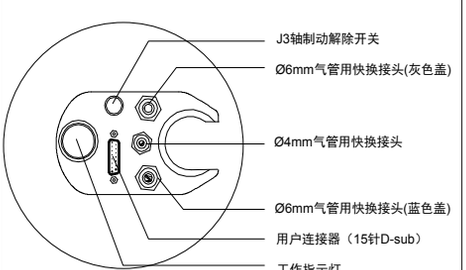
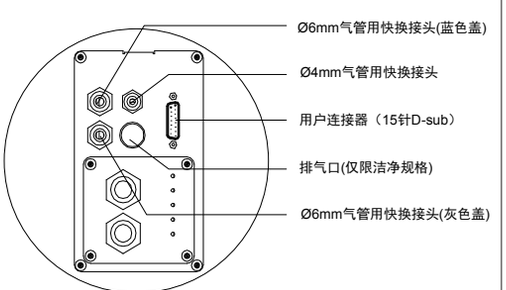
■ 9Pin 连接器

针脚定义	信号	连接器	NO	连接	NO	连接器	针脚定义
供用户自行定义	用户专用信号线	5 1 9 6 IO (机械臂侧)	1 2 3 4 5 6 7 8 9 外壳		1 2 3 4 5 6 7 8 9 外壳	5 1 9 6 IO (基座侧)	供用户自行定义

3.6.2 配管（空气管）

1 IRS100 系列

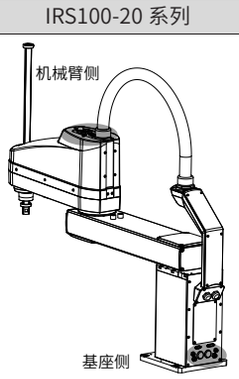
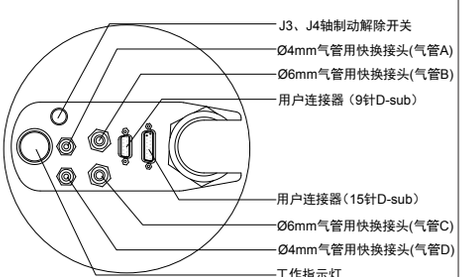
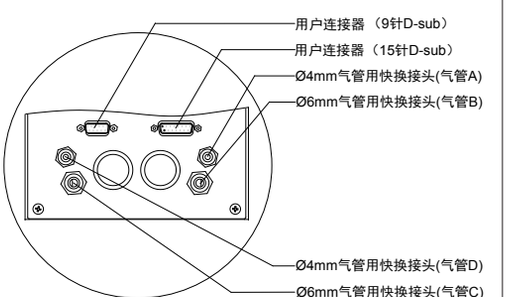
最大使用压力	数量	规格（外径 × 内径）
0.59MPa (6 kgf/cm^2)	2	ø6 mm × ø4 mm
	1	ø4 mm × ø2.5 mm

IRS100 系列	机械臂侧	基座侧
		

2 IRS100-20 系列

最大使用压力	数量	规格（外径 × 内径）
0.59MPa (6 kgf/cm^2)	2	ø6 mm × ø4 mm
	2	ø4 mm × ø2.5 mm

注：空气管的两端附带有用于管外径 ø6 mm 与 ø4 mm 的快速接头。

IRS100-20 系列	机械臂侧	基座侧
		

3.7 IO 接线

详细内容请参见本手册“5.3 输入 I/0 接线”“5.4 输出 IO 接线”。

3.8 安全接线

详细内容请参见本手册“5.5 SAFETY 接线”

3.9 末端夹具的安装注意事项

3.9.1 安装注意事项

请客户自行制作末端夹具，安装末端夹具时，请注意以下事项。

> **注 意**
>
> ◆ 请务必在断开电源和未放置工件的状态下进行卡盘的配线和空气配管，在未断开电源的情况下如果按下急停开关，此时工件会松开，可能导致机器人系统和工件损坏。
>
> ◆ 系统布局时请注意末端夹具的干涉区域。安装末端夹具后操作机器人运动，可能因末端夹具的外径、工件的大小或机械臂的位置等导致末端夹具或工件与机器人本体接触，可能造成机器人系统和工件损坏。
>
> ◆ 注意：对于末端夹具等发生故障后接触电源的电气设备，机器人主机上没有专用的接地端子，请客户正确接地。

1 J3 丝杆轴

请将末端夹具安装在丝杆轴的下端。

在 J3 丝杆轴上安装末端夹具时，请采用 M4 以上的螺纹抱紧的结构。

切勿移动 J3 丝杆轴下侧的上限机械挡块。如果进行“Jump 动作”，上限机械挡块则可能会撞击机器人主体，导致机器人无法正常进行动作。

2 制动解除开关

在安装末端夹具时，如果需要上下移动第 3 关节，或旋转转动 IRS100-20 系列的第 4 关节，请打开控制柜电源，并按下制动解除开关。该开关为瞬时型，仅在被按下期间解除制动。

制动解除开关在机器人本体上的位置如下图所示：

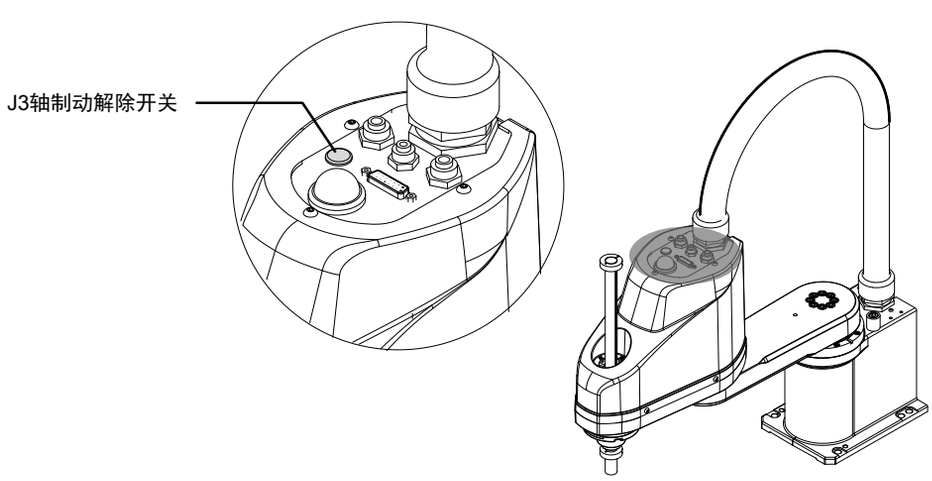


图 3-33 IRS100 系列制动解除开关位置示意图

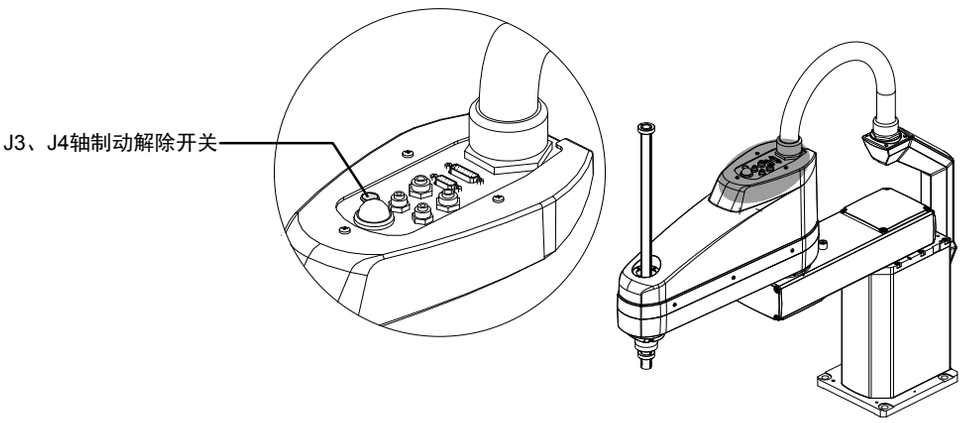


图 3-34 IRS100-20 系列制动解除开关位置示意图

注 意

- ◆ 在机器人作业期间，如果电源被切断，或通电期间电机进入关闭状态时，末端夹具会因自重而导致轴下降，存在碰撞外围装置的风险。设置制动解除开关，可防止末端夹具因自重导致下降。
- ◆ 按下制动解除开关期间，请注意因末端夹具自重而产生下垂。建议先卸下负载，如果不允许卸载负载时，请确保有人员或机械辅助，避免负载掉落伤人。

3.9.2 空间要求

如果安装末端夹具并进行动作，则可能会因末端夹具的外径、工件的大小或机械臂的位置等导致与机器人本体接触。因此，在进行系统布局时，务必注意末端夹具的干涉区域。详细空间范围请参考“机器人安装空间”章节。

在进行系统布局设计时，请参考“机器人安装空间”章节设计详细空间范围，注意末端夹具的干涉区域，避免与机器人本体接触。

3.10 安装相机 / 气动阀

IRS100 系列和 IRS100-20 系列 SCARA 机器人配置了相机和气动阀安装装置，安装位置在第 2 机械臂底部，如图所示。

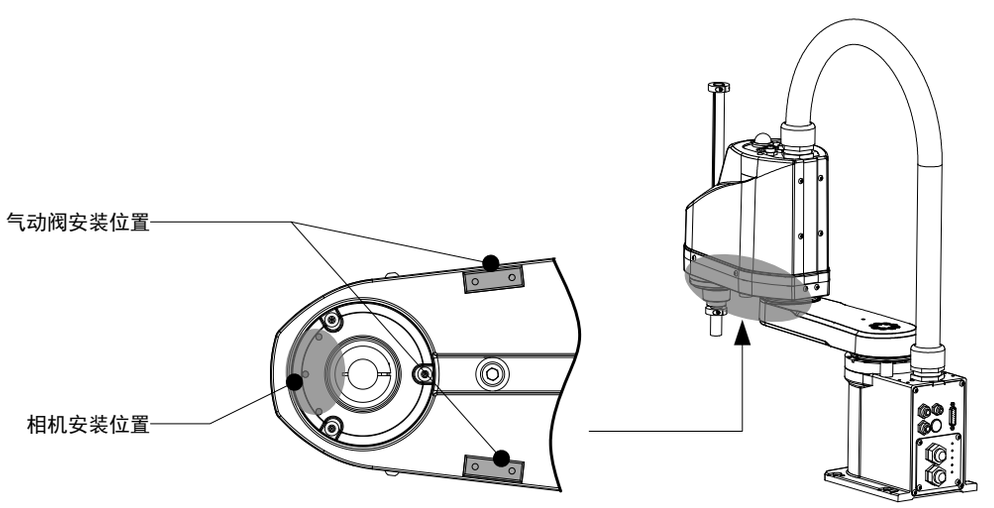


图 3-35 IRS100 系列相机 / 气动阀安装位置示意图

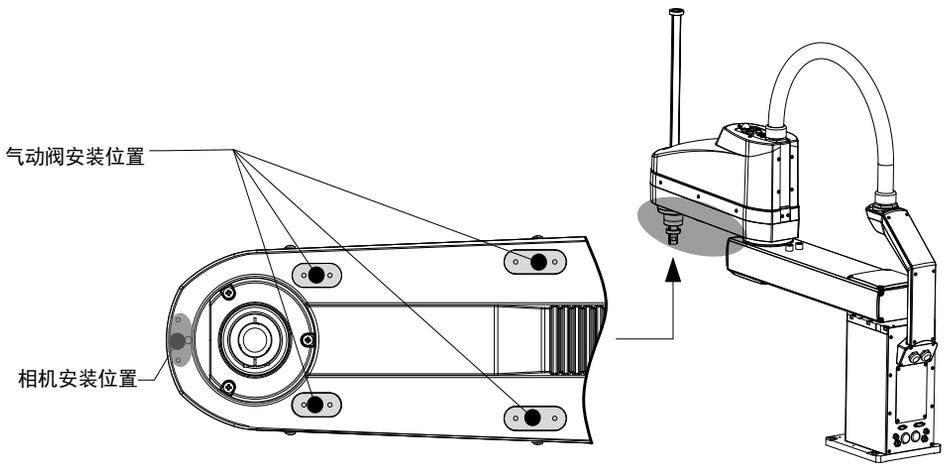


图 3-36 IRS100-20 系列相机 / 气动阀安装位置示意图

相机 / 气动阀没有专用的安装孔位，需用户自备安装固定件，安装固定件的尺寸请参考机器人相机安装孔位。如下：

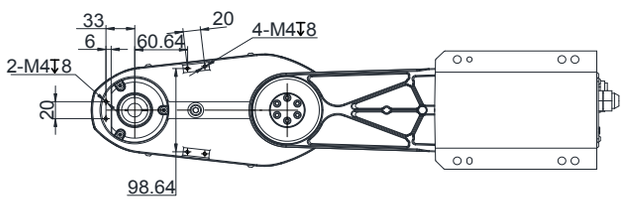


图 3-37 相机 / 气动阀安装螺纹孔尺寸（IRS100-3-40Z*）（单位：mm）

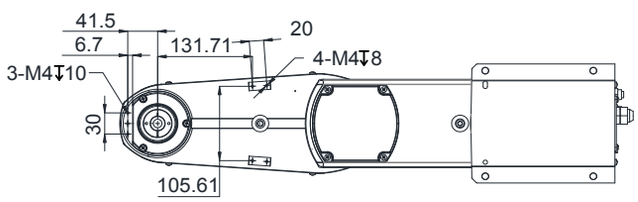


图 3-38 相机 / 气动阀安装螺纹孔尺寸（IRS100-6-60Z* RB100-6-50Z*、IRB100-6-65Z*、IRB100-6-70Z*））（单位：mm）

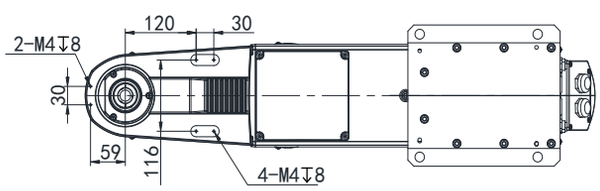


图 3-39 相机 / 气动阀安装螺纹孔尺寸（IRB100-20-60Z*，IRB100-20-70Z*）（单位：mm）

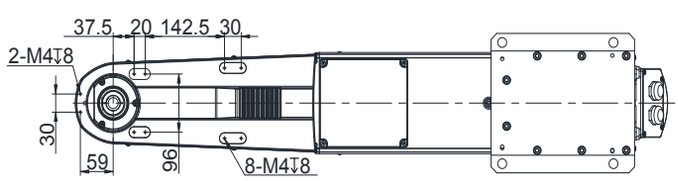


图 3-40 相机 / 气动阀安装螺纹孔尺寸（IRS100-20-80Z*，IRB100-20-100Z*）（单位：mm）

第 4 章 汇川机器人 SCARA 本体

4.1 型号说明

IRB100 -3 -40 Z15 T S 3 -00

SCARA机器人本体

负载	
3	3kg
6	6kg
20	20kg

臂长	
40	400mm
60	600mm
70	700mm
80	800mm
100	1000mm

Z轴长行程	
15	150mm
18	180mm
20	200mm
42	420mm

非标代号	
00	标准机型
其他	非标机型

线缆长度（米）

安装环境	
S	标准环境
C	洁净环境

安装方式	
T	台面
R	吊顶
W	侧壁

4.2 规格一览表

4.2.1 IRB100 系列

项目		IRB100-3-40Z15TS3	IRB100-6-50Z20TS3	IRB100-6-60Z20TS3	IRB100-6-65Z20TS3	IRB100-6-70Z20TS3
机械臂长	第 1+ 第 2 机械臂	400mm	500mm	600mm	650mm	700mm
	第 1 机械臂	225mm	225mm	325mm	375mm	425mm
	第 2 机械臂	175mm	275mm	275mm	275mm	275mm
最大动作速度 [1]	第 1+ 第 2 关节	6000mm/s	6150mm/s	6800mm/s	7130mm/s	7450mm/s
	第 3 关节	1100mm/s	1100mm/s	1100mm/s	1100mm/s	1100mm/s
	第 4 关节	2600° /s	2000° /s	2000° /s	2000° /s	2000° /s
重复精度	第 1+ 第 2 关节	±0.01mm	±0.02mm	±0.02 mm	±0.02mm	±0.02mm
	第 3 关节	±0.01mm	±0.01mm	±0.01mm	±0.01mm	±0.01mm
	第 4 关节	±0.01°	±0.01°	±0.01°	±0.01°	±0.01°
可搬运重量（负载）	额定	1kg	2kg	2kg	2kg	2kg
	最大	3kg	6kg	6kg	6kg	6kg
第 4 关节容许装载惯性 [2]	额定	0.005kg·m^2	0.01kg·m^2	0.01kg·m^2	0.01kg·m^2	0.01kg·m^2
	最大	0.05kg·m^2	0.12kg·m^2	0.12kg·m^2	0.12kg·m^2	0.12kg·m^2

项目		IRB100-3-40Z15TS3	IRB100-6-50Z20TS3	IRB100-6-60Z20TS3	IRB100-6-65Z20TS3	IRB100-6-70Z20TS3
丝杆直径	安装	ø16 mm	ϕ20mm	ø20 mm	ϕ20mm	ϕ20mm
	中空	ø9 mm	ϕ12mm	ø12 mm	ϕ12mm	ϕ12mm
底座安装孔 6xϕ9mm		120mmx 120mm/ 135mmx 200mm	150mmx 150mm	150mmx 150mm	150mmx 150mm	150mmx 150mm
		4xϕ9mm	4xϕ9mm	4xϕ9mm	4xϕ9mm	
主体重量（不含电缆重量）		13kg	18kg	19kg	20kg	21kg
驱动方式	全关节	AC 伺服马达	AC 伺服马达	AC 伺服马达	AC 伺服马达	AC 伺服马达
电机功耗	第 1 关节	400W	400W	400W	400W	400W
	第 2 关节	100W	400W	400W	400W	400W
	第 3 关节	100W	100W	100W	100W	100W
	第 4 关节	100W	100W	100W	100W	100W
选件	设置环境	标准型				
第 3 关节压入力	100N					
用户配线	15(15pin: D-sub)					
用户配管	ø6mm 空气管 2 根·耐压：0.59Mpa(6kgf/cm^2:86psi)					
	ø4mm 空气管 1 根·耐压：0.59Mpa(6kgf/cm^2:86psi)					
环境条件	环境温度	5 ～ 40° C（不应有过大温度变化）				
	环境相对湿度	10 ～ 80%（不得结露）				
噪声 级 [3]	L_{Aeq}=70dB(A)					
适用控制器	IMC100R-E-X1					
可设定值 (默认值)	Speed	1~(50)~100				
	Accel	1~(50)~100				
MTBF	3 年					
安全标准	ANSI/RIAR15.06-1999					
	NFPA79(2007Edition)					
	CSA/CANZ434-03(February2003)					
	CE 标志（机械指令、低电压指令、EMC 指令）					
最大动作范围	第 1 关节	±132°	±127°	±132 °	±127°	±127°
	第 2 关节	±141°	±145°	±150 °	±145°	±145°
	第 3 关节	150mm/120 mm	170mm / 200mm	200 mm/170mm	170mm / 200mm	170mm / 200mm
	第 4 关节	±360 °	±360°	±360°	±360°	±360°
节拍时间		0.42s	0.40s	0.42s	0.43s	0.44s

4.2.2 IRB100-20 系列

项目		IRB100-20-60-Z18TS3	IRB100-20-70-Z18TS3	IRB100-20-80-Z42TS3	IRB100-20-100-Z42TS3
机械臂长	第 1+ 第 2 机械臂	600mm	700mm	800mm	1000mm
	第 1 机械臂	300mm	400mm	350mm	550mm
	第 2 机械臂	300mm	300mm	450mm	450mm
最大动作速度 [1]	第 1+ 第 2 关节	6800mm/s	7460mm/s	9940mm/s	11250mm/s
	第 3 关节	1010mm/s	1010mm/s	1010mm/s	1010mm/s
	第 4 关节	1400° /s	1400° /s	1400° /s	1400° /s
重复精度	第 1+ 第 2 关节	±0.02mm	±0.02mm	±0.025mm	±0.025mm
	第 3 关节	±0.01mm	±0.01mm	±0.01mm	±0.01mm
	第 4 关节	±0.01°	±0.01°	±0.01°	±0.01°
可搬运重量（负载）	额定	10kg	10kg	10kg	10kg
	最大	20kg	20kg	20kg	20kg
第 4 关节容许装载惯性 [2]	额定	0.05kg·m^2	0.05 kg·m^2	0.05 kg·m^2	0.05 kg·m^2
	最大	0.45kg·m^2	0.45 kg·m^2	0.45 kg·m^2	0.45 kg·m^2
夹具末端直径	安装	ϕ25mm	ϕ25mm	ϕ25mm	ϕ25mm
	中空	ϕ18mm	ϕ18mm	ϕ18mm	ϕ18mm
	安装孔	200mmx200mm	200mmx200mm	200mmx200mm	200mmx200mm
		4xϕ16mm	4xϕ16mm	4xϕ16mm	4xϕ16mm
主体重量（不含电缆重量）		42kg	43kg	50kg	53kg
驱动方式	全关节	AC 伺服马达	AC 伺服马达	AC 伺服马达	AC 伺服马达
电机功耗	第 1 关节	750W	750W	750W	750W
	第 2 关节	750W	750W	750W	750W
	第 3 关节	400W	400W	400W	400W
	第 4 关节	200W	200W	200W	200W
选件	设置环境	标准型			
第 3 关节压入力	250 N				
用户配线	9（9pin:D-sub）和 15（15pin:D-sub）				
用户配管	ø6mm 空气管 2 根 . 耐压：0.59Mpa(6kgf/cm2:86psi)				
	ø4mm 空气管 2 根 . 耐压：0.59Mpa(6kgf/cm2:86psi)				
环境条件	环境温度	5 ～ 40° C（不应有过大温度变化）			
	环境相对湿度	10 ～ 80%（不得结露）			
噪声级 [3]	LAeq=70dB(A)				
适用控制器	IMC100R-E-X1				
可设定值 (默认值)	Speed	1~(50)~100			
	Accel	1~(50)~100			
MTBF	3 年				
安全标准	ANSI/RIAR15.06-1999				
	NFPA79(2007Edition)				
	CSA/CANZ434-03(February2003)				
	CE 标志（机械指令、低电压指令、EMC 指令）				
最大动作范围	第 1 关节	±132°	±132°	±132°	±132°
	第 2 关节	±152°	±152°	±152°	±152°
	第 3 关节	180mm	180mm	420mm	420mm
	第 4 关节	±360°	±360°	±360°	±360°
节拍时间		0.47s	0.47s	0.49s	0.5s

注：

[1] 为 PTP 命令时。CP 动作的最大动作速度为 2000mm/s（水平）。

[2] 负载重心与第 4 关节中心位置一致 , 当重心位置偏离第 4 关节中心位置时，请利用惯性（Inertia）命令设定参数。

[3] 测量条件：在机器人为额定负载、4 关节同时工作、最大速度、最大加速度、占空比 50% 的操作条件下，在机器人正面、距离动作区域 1000mm、底座安装面 50mm 以上的位置测量。

4.3 机型结构说明

4.3.1 各部件说明

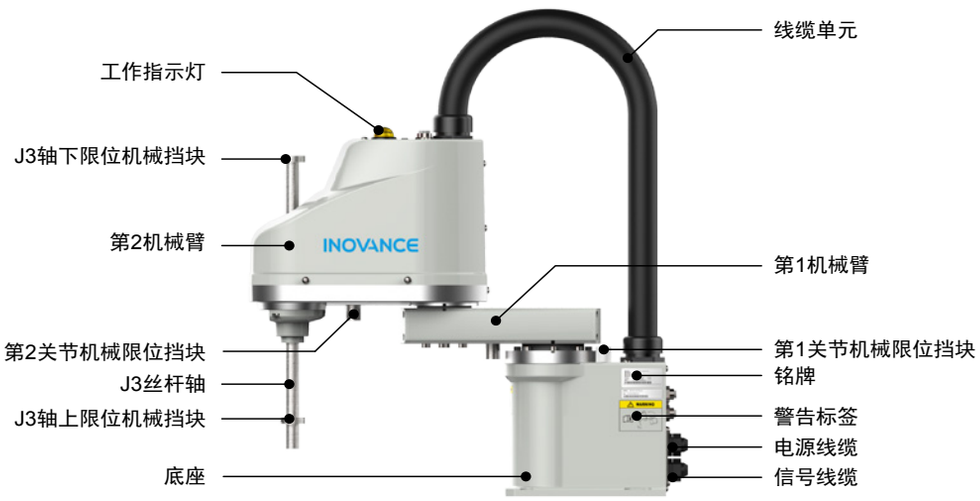


图 4-41 IRB100 系列机器人产品外观及各部件信息

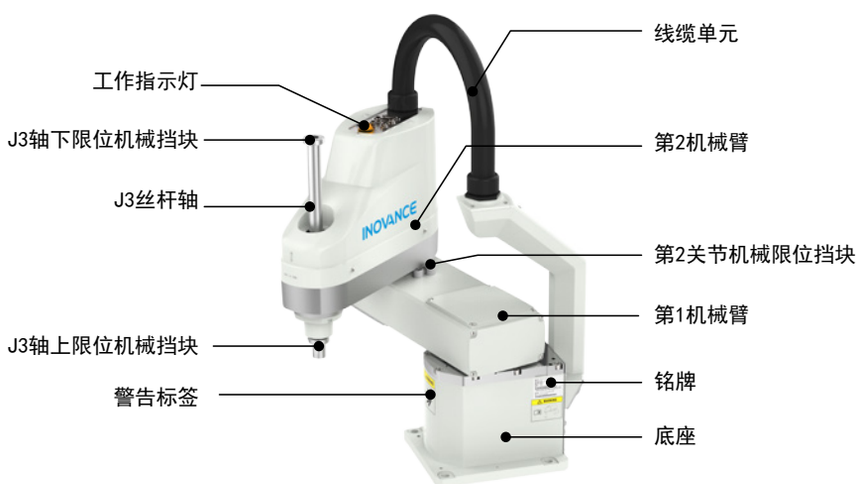


图 4-42 IRB100-20 系列机器人产品外观及各部件信息

4.3.2 安装尺寸

1 IRB100-3-40Z15TS3 安装尺寸

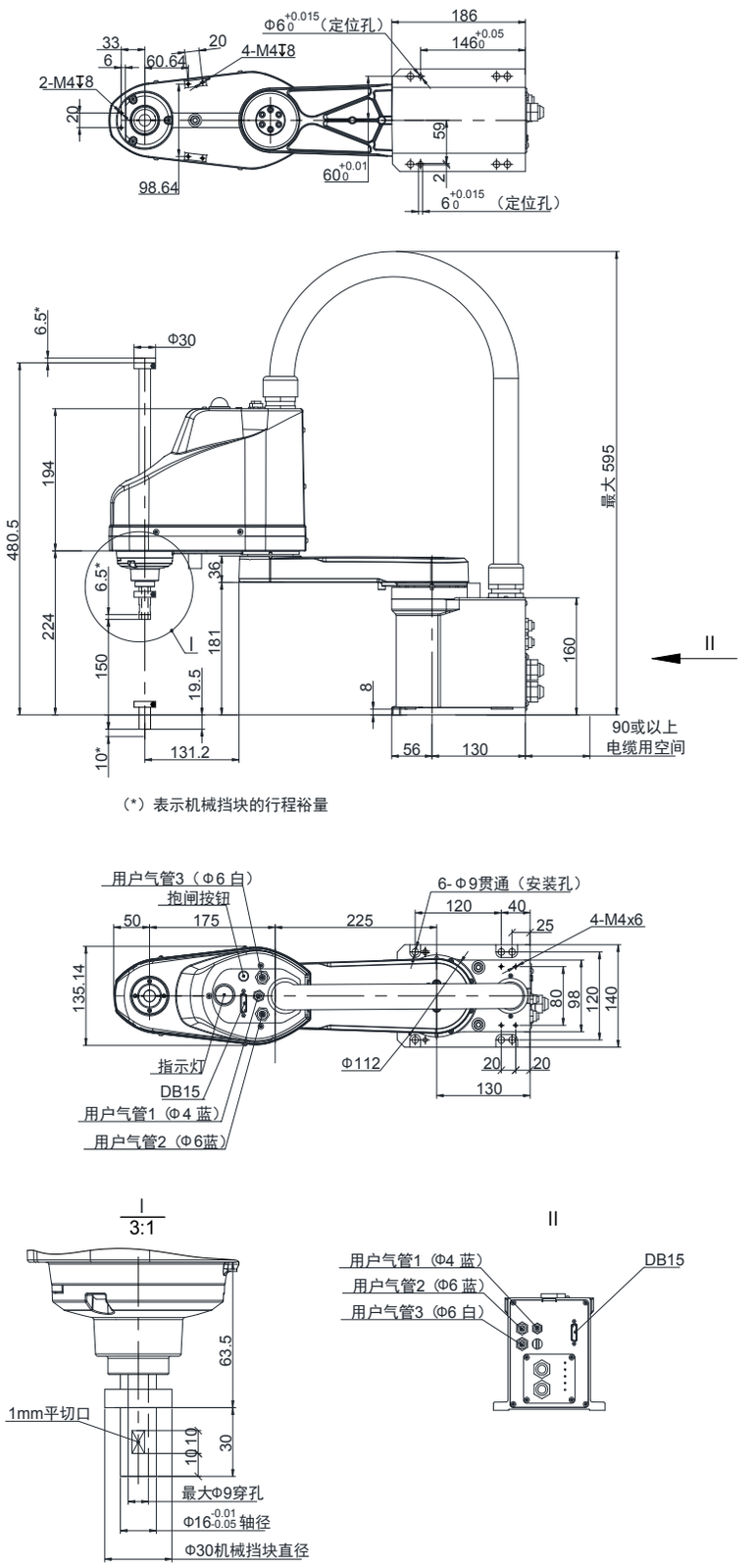
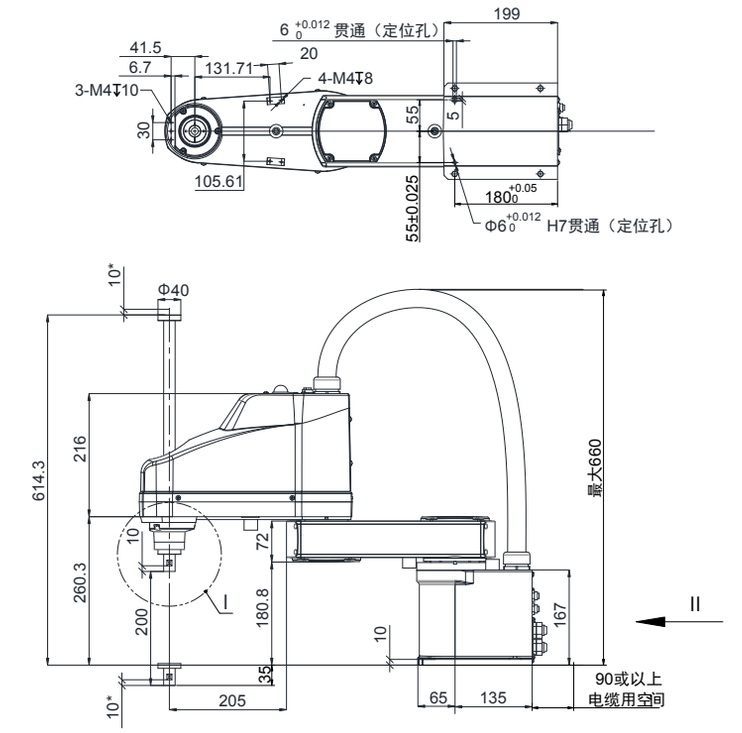
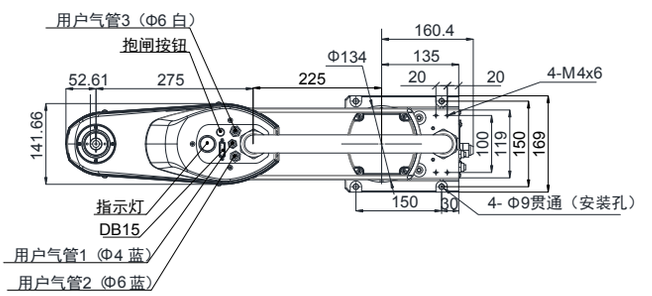


图 4-43 IRB100-3-40Z15TS3 安装尺寸

2 IRB100-6-50Z20TS3 安装尺寸



（*）表示机械挡块的行程裕量



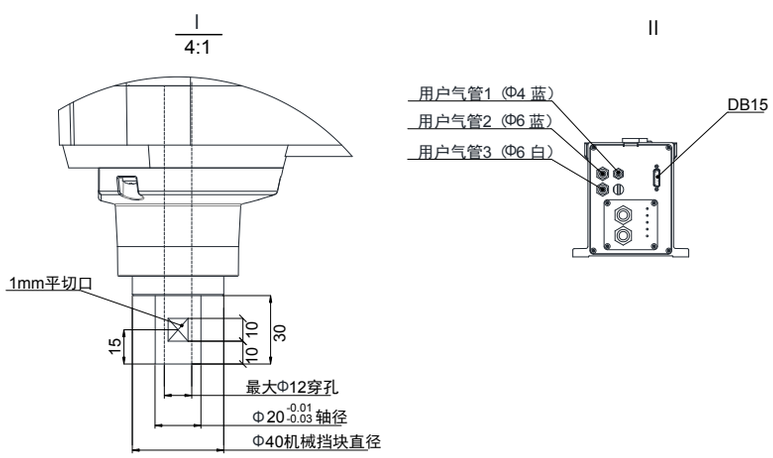


图 4-44 IRB100-6-50Z20TS3 安装尺寸

3 IRB100-6-60Z20TS3 安装尺寸

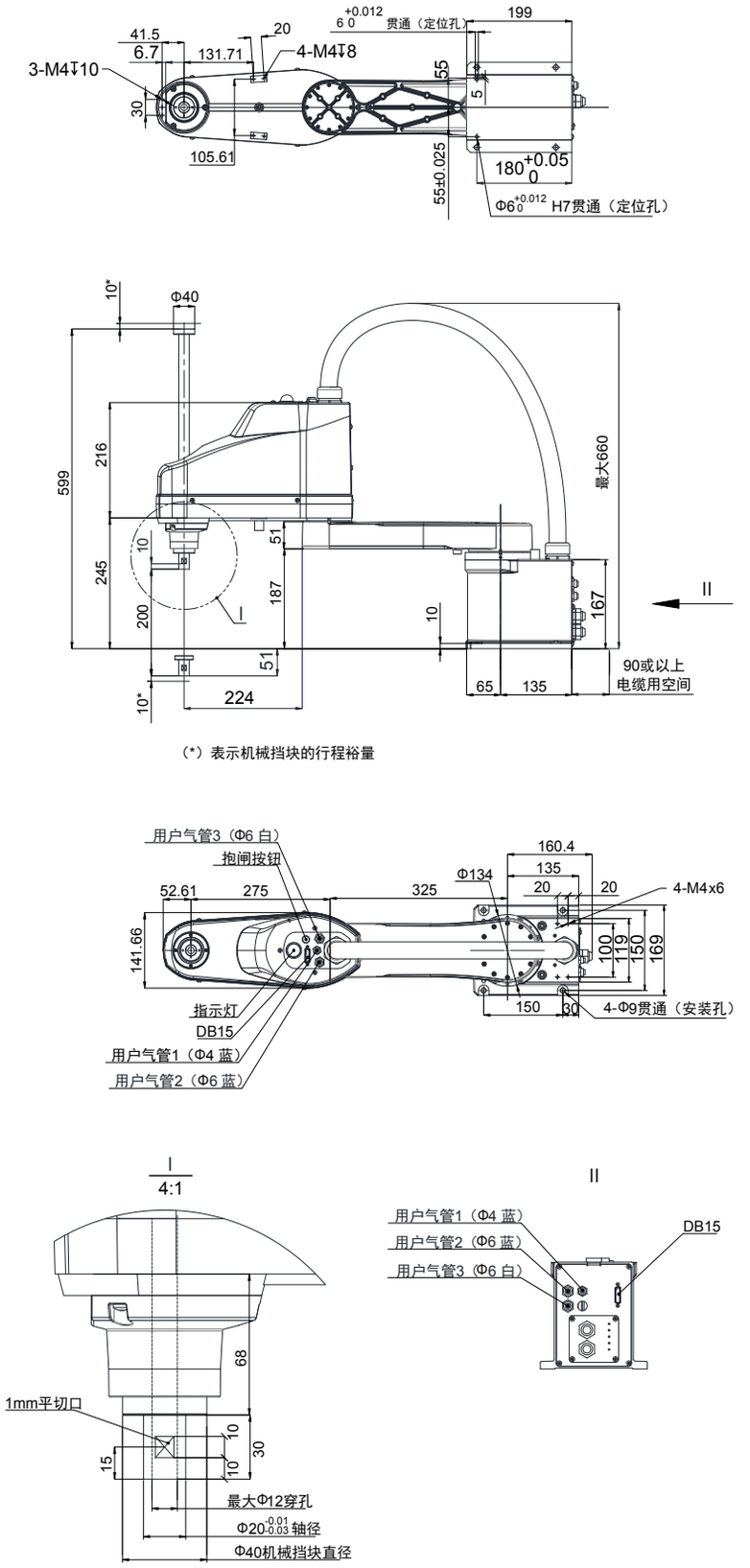


图 4-45 IRB100-6-60Z20TS3 安装尺寸

4 IRB100-6-65Z20TS3 安装尺寸

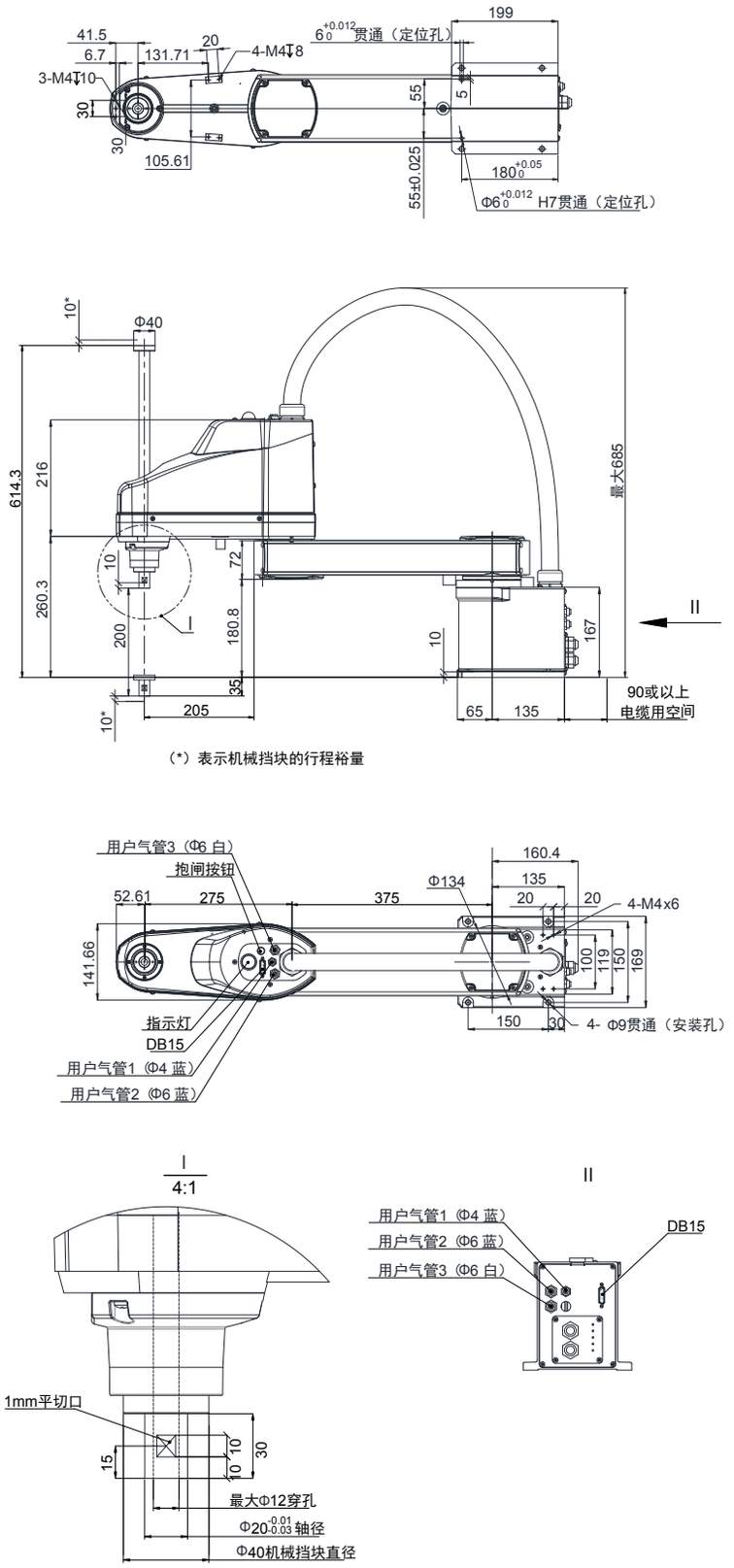


图 4-46 IRB100-6-65Z20TS3 安装尺寸

5 IRB100-6-70Z20TS3 安装尺寸

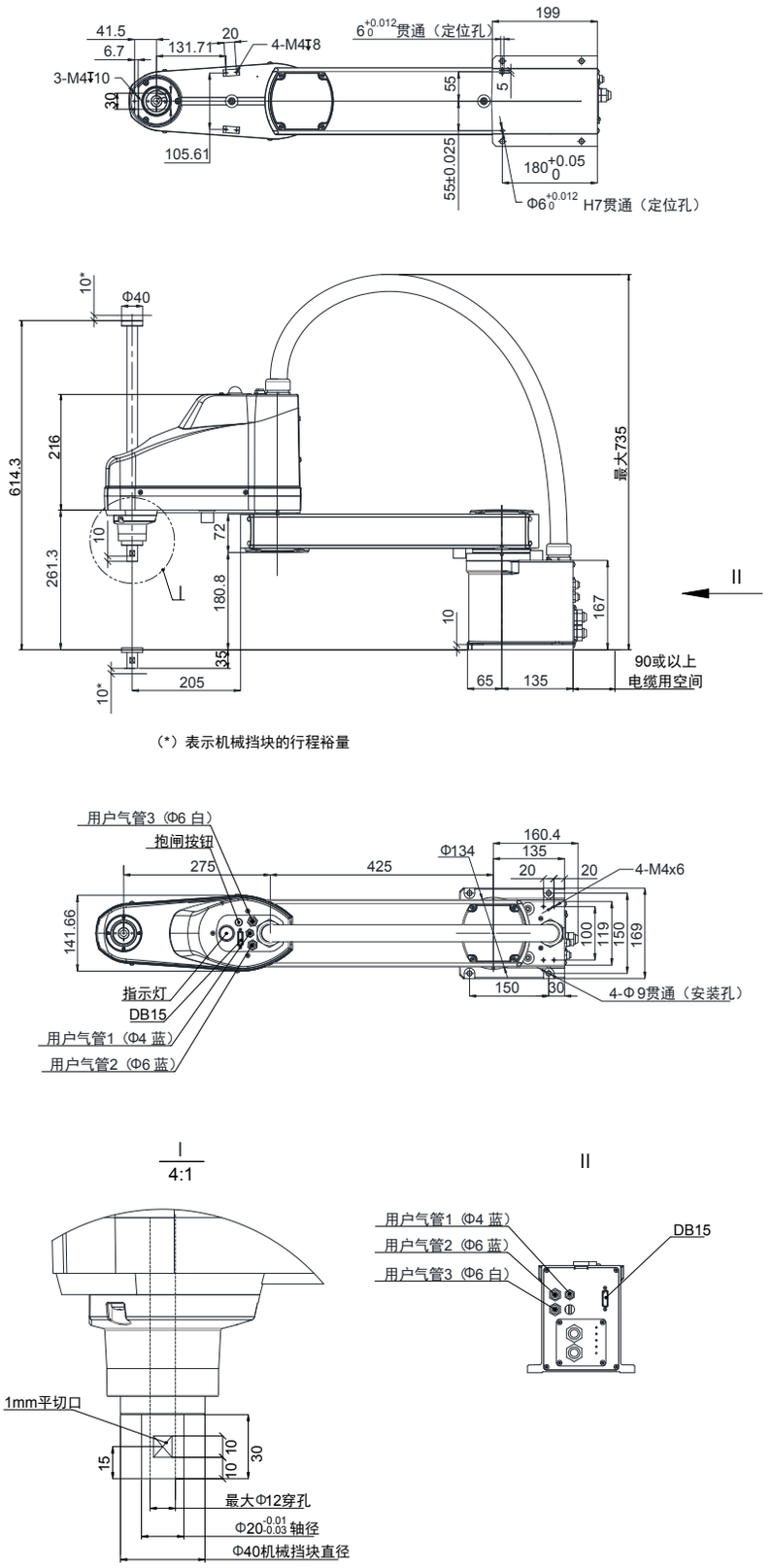
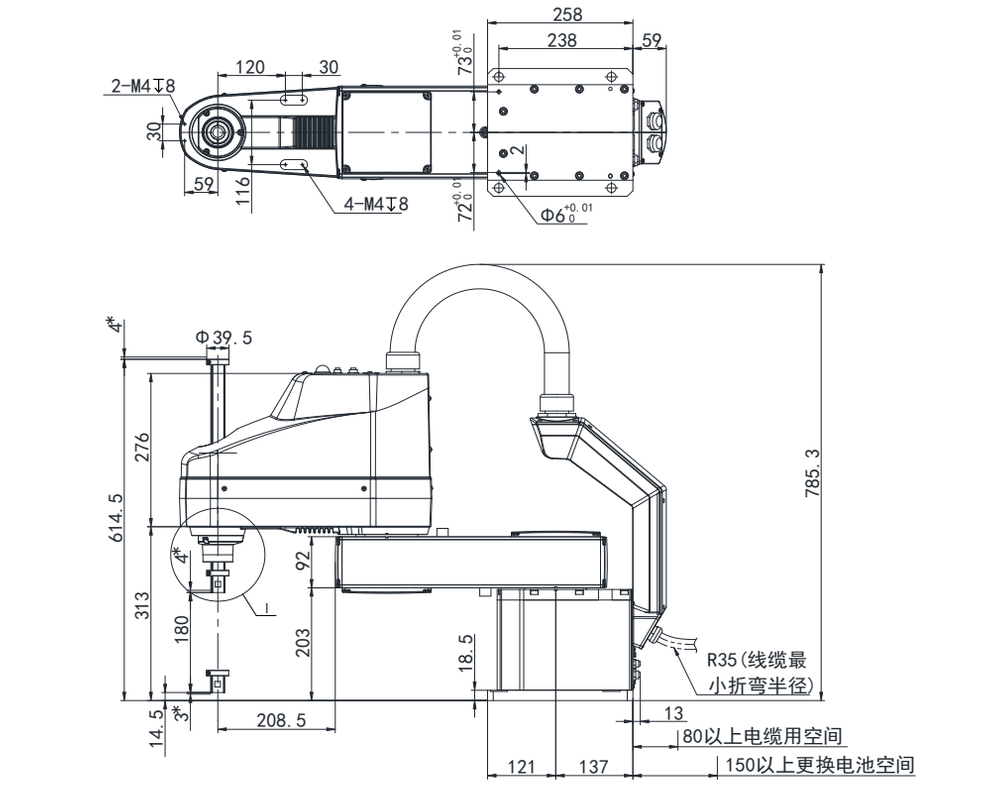


图 4-47 IRB100-6-70Z20TS3 安装尺寸

6 IRB100-20-60Z18TS3 安装尺寸



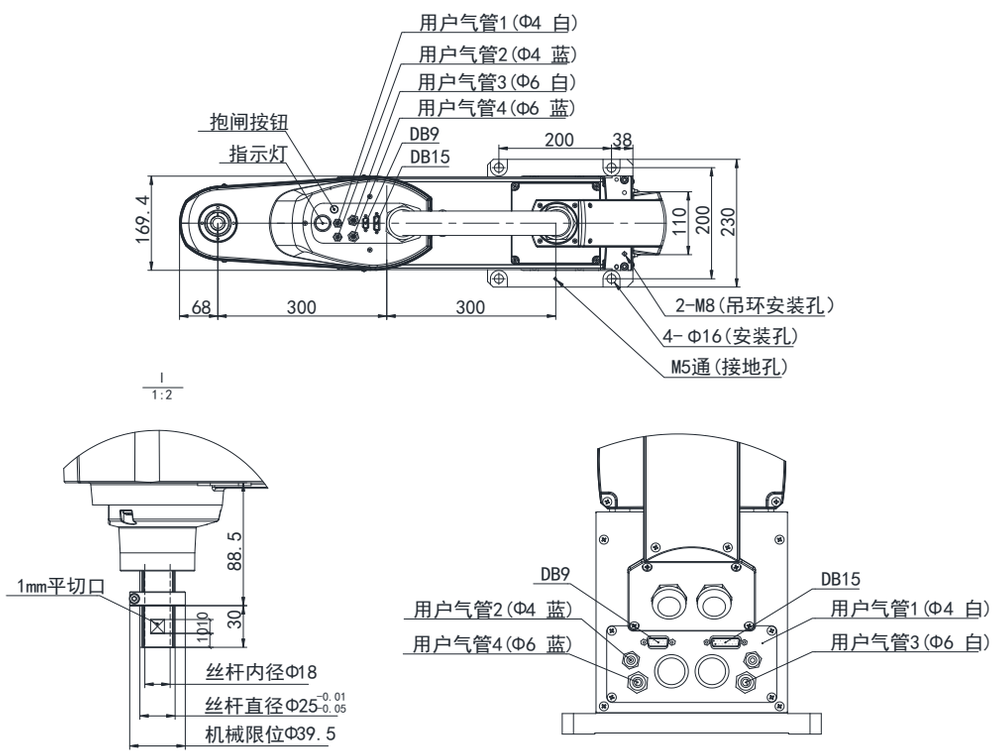


图 4-48 IRB100-20-60Z18TS3 安装尺寸

7 IRB100-20-70Z18TS3 安装尺寸

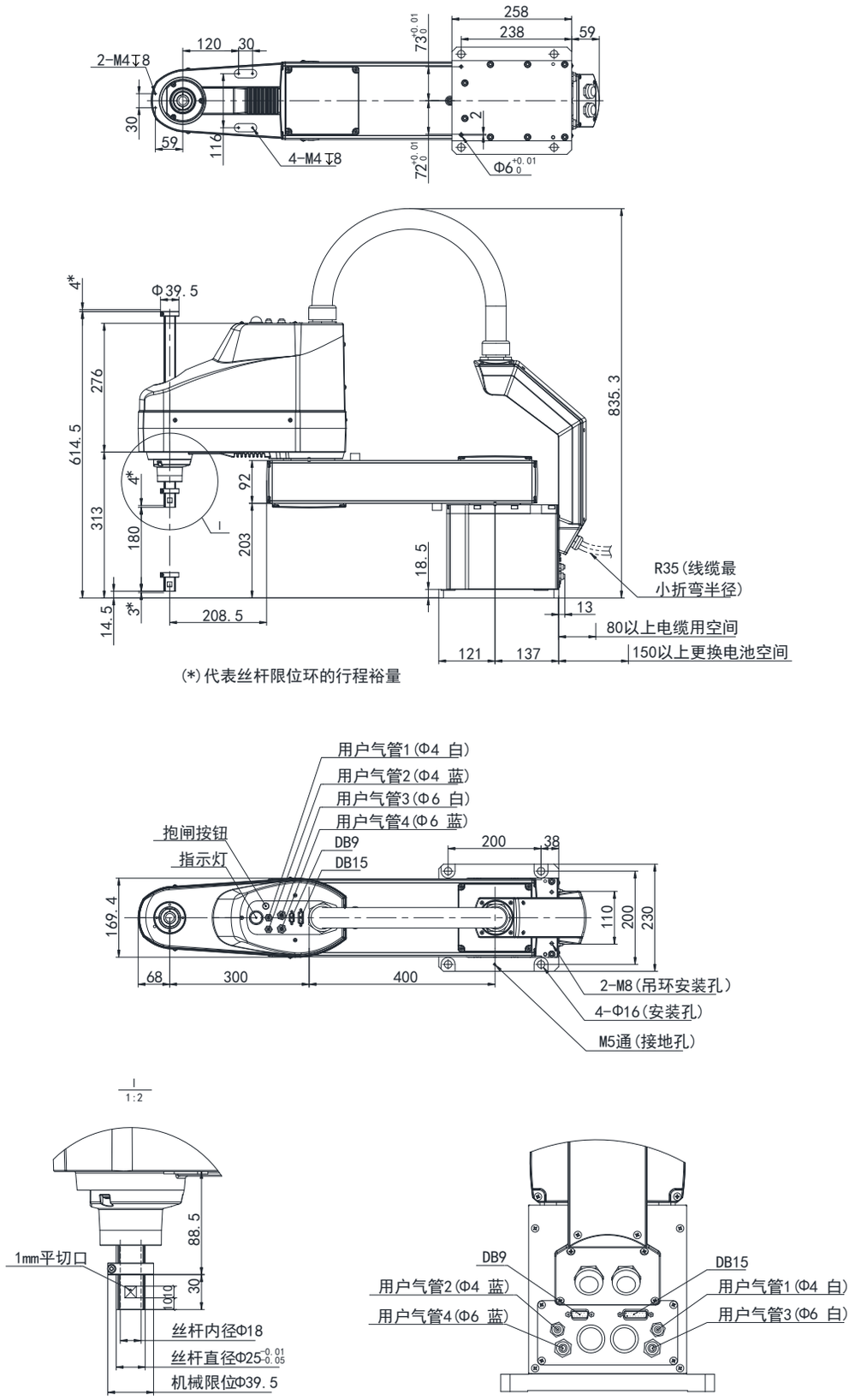


图 4-49 IRB100-20-70Z18TS3 安装尺寸

8 IRB100-20-80Z42TS3 安装尺寸

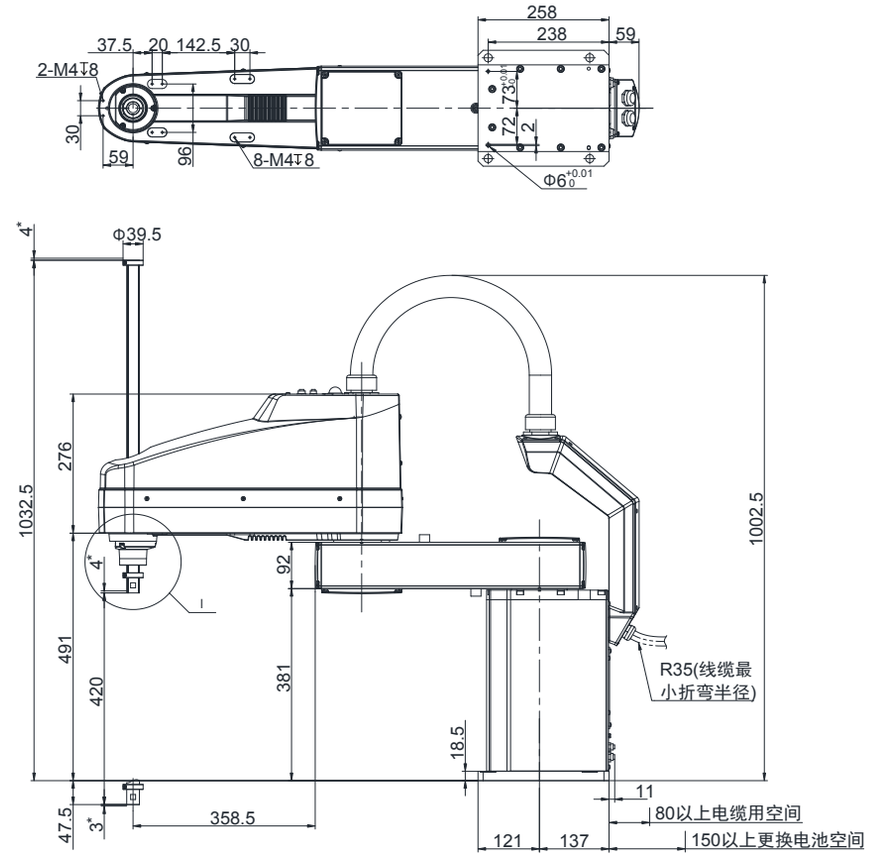


图 4-50 IRB100-20-80Z42TS3 安装尺寸

9 IRB100-20-100Z42TS3 安装尺寸

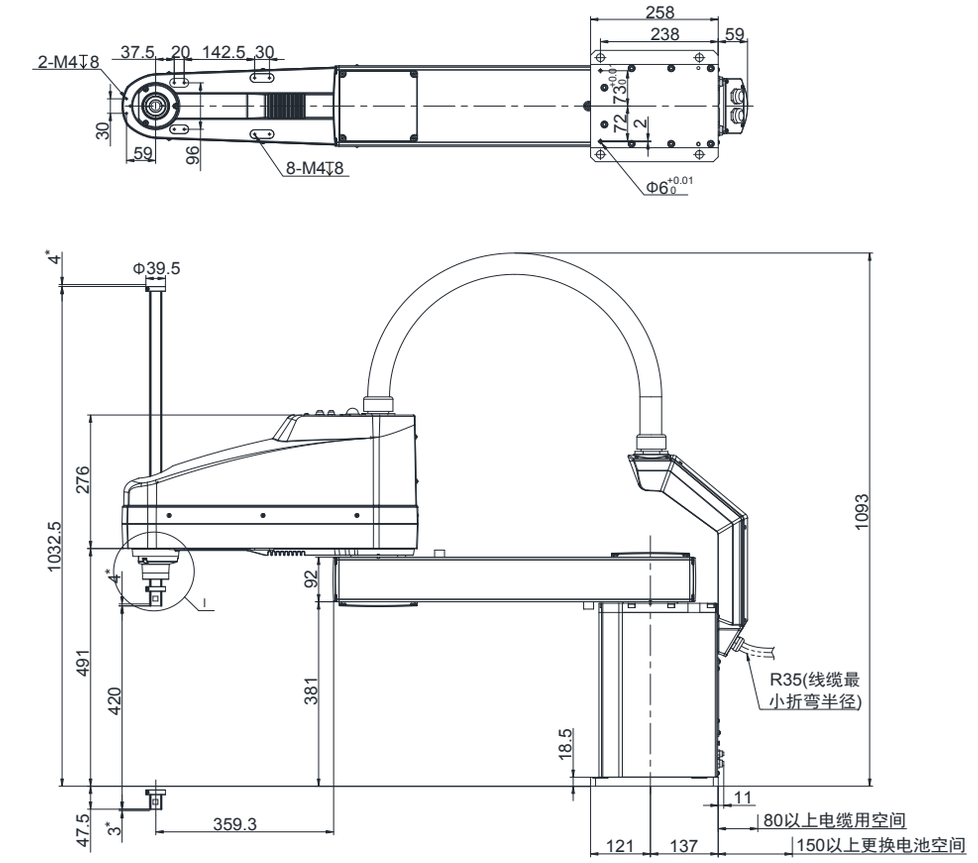


图 4-51 IRB100-20-100Z42TS3 安装尺寸

4.3.3 机器人安装空间

1 IRB100-3-40Z15TS3 标准规格机型运动范围

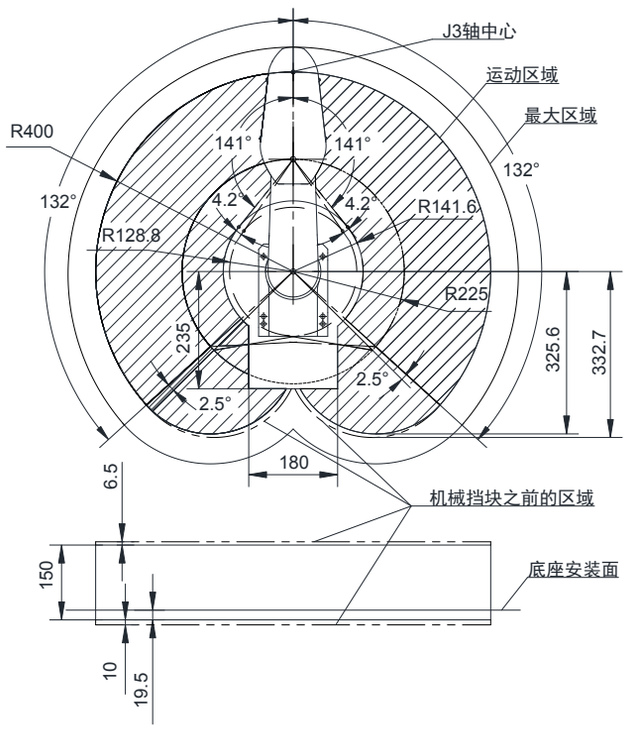


图 4-52 IRB100-3-40Z15TS3 标准规格机型运动范围（单位：mm）

2 IRB100-6-50Z20TS3 标准规格机型运动范围

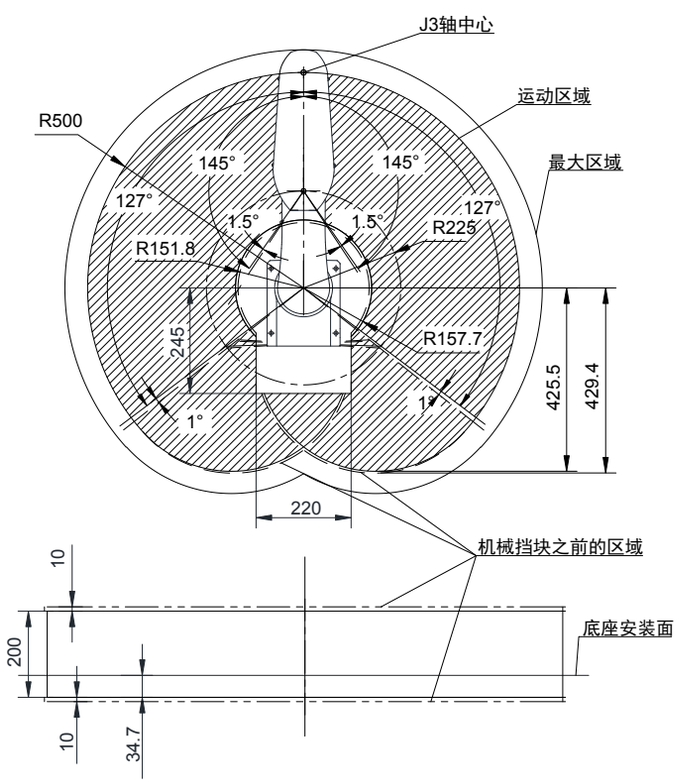


图 4-53 IRB100-6-50Z20TS3 标准规格机型运动范围（单位：mm）

3 IRB100-6-60Z20TS3 标准规格机型运动范围

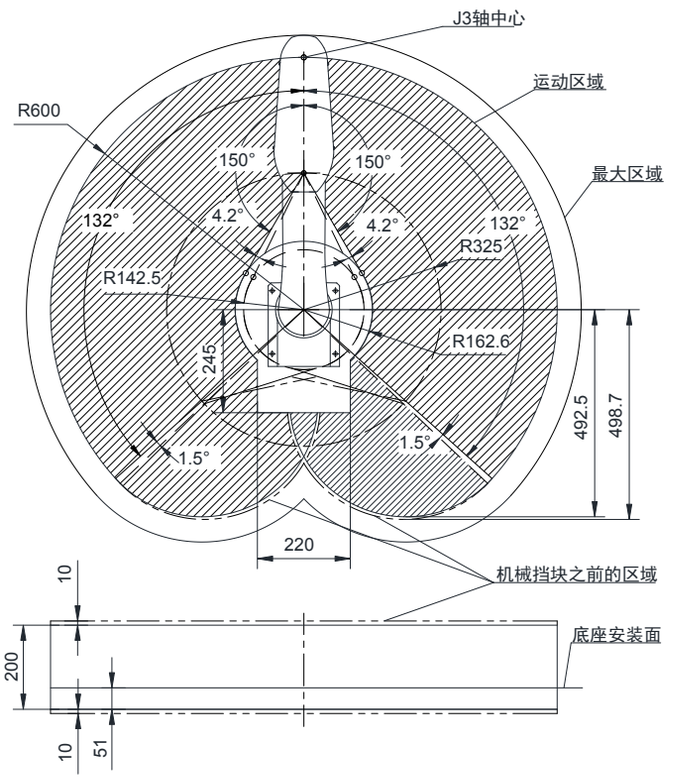


图 4-54 IRB100-6-60Z20TS3 标准规格机型运动范围（单位：mm）

4 IRB100-6-65Z20TS3 标准规格机型运动范围

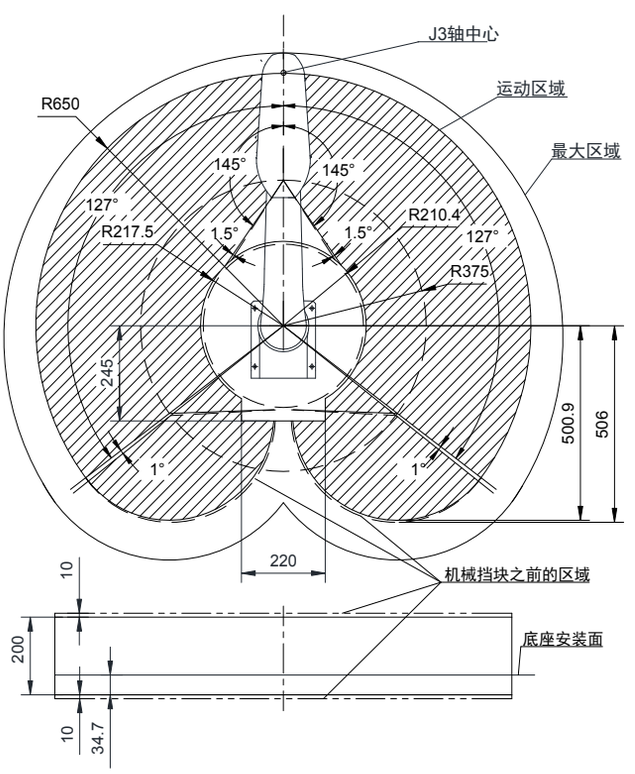


图 4-55 IRB100-6-65Z20TS3 标准规格机型运动范围（单位：mm）

5 IRB100-6-70Z20TS3 标准规格机型运动范围

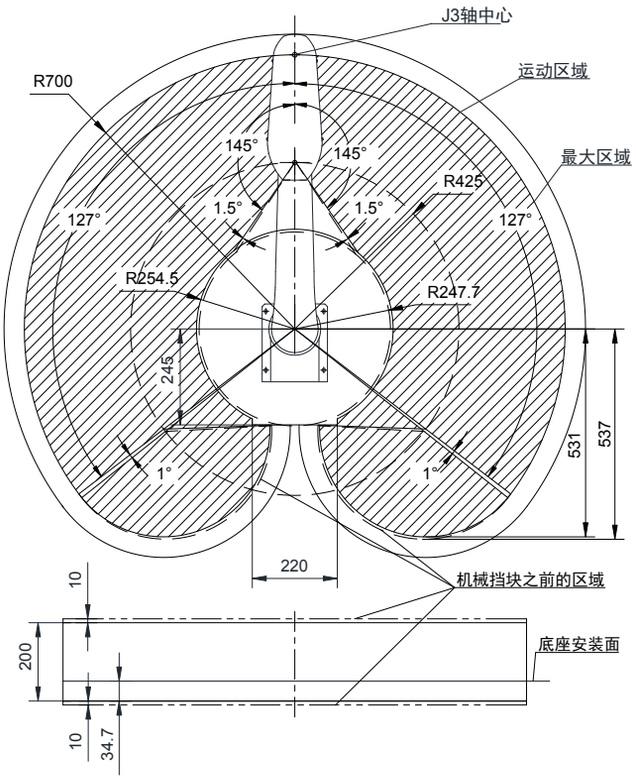


图 4-56 IRB100-6-70Z20TS3 标准规格机型运动范围（单位：mm）

6 IRB100-20-60Z18TS3 标准规格机型运动范围

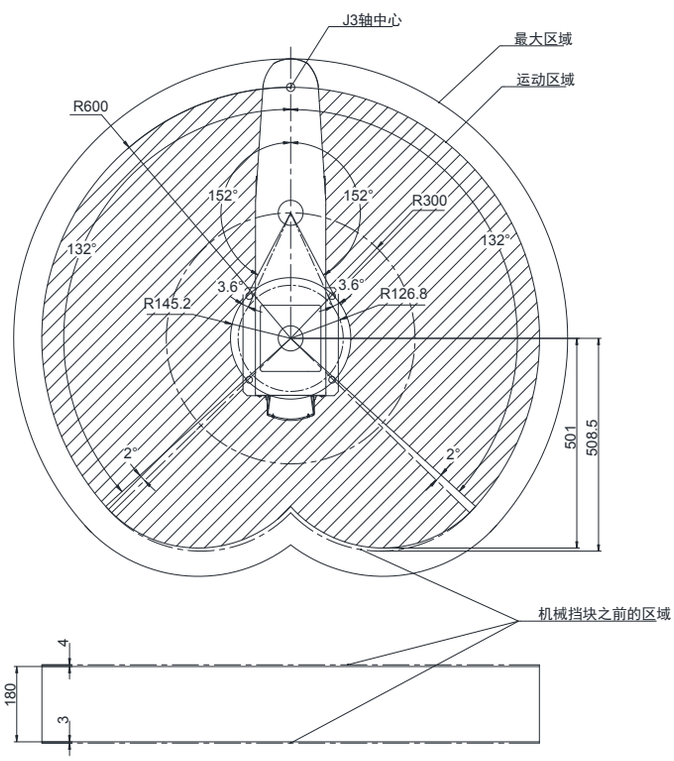


图 4-57 IRB100-20-60Z18TS3 标准规格机型运动范围（单位：mm）

7 IRB100-20-70Z18TS3 标准规格机型运动范围

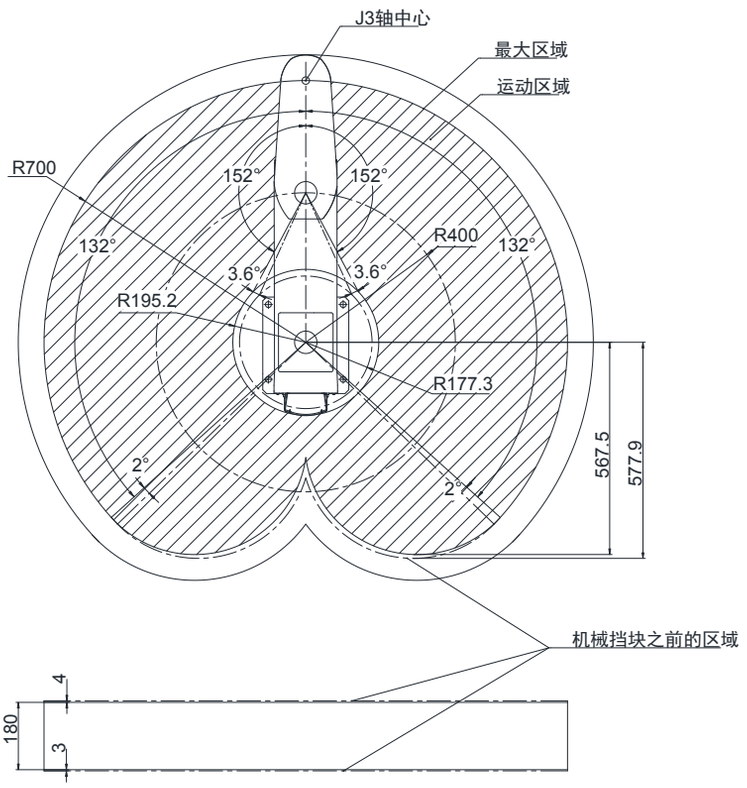


图 4-58 IRB100-20-70Z18TS3 标准规格机型运动范围（单位：mm）

8 IRB100-20-80Z42TS3 标准规格机型运动范围

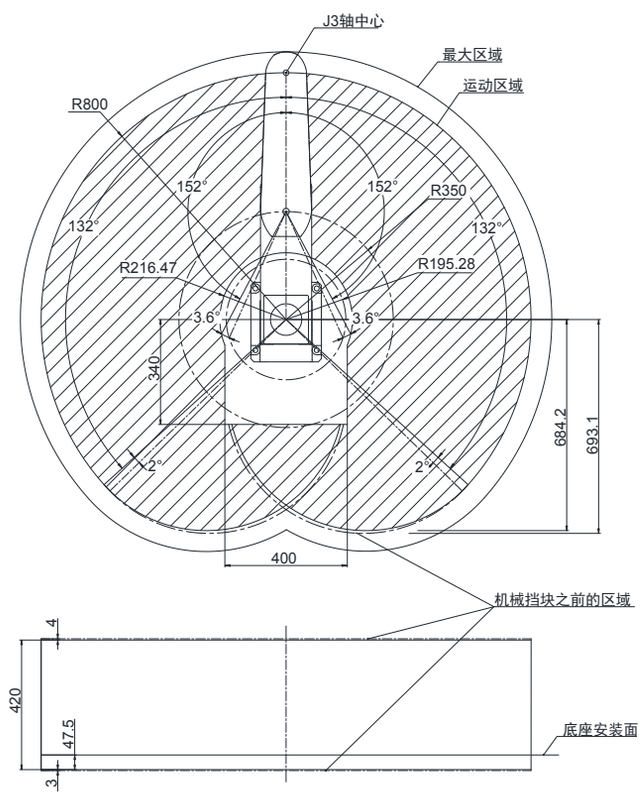


图 4-59 IRB100-20-80Z42TS3 标准规格机型运动范围（单位：mm）

9 IRB100-20-80Z42TS3 标准规格机型运动范围

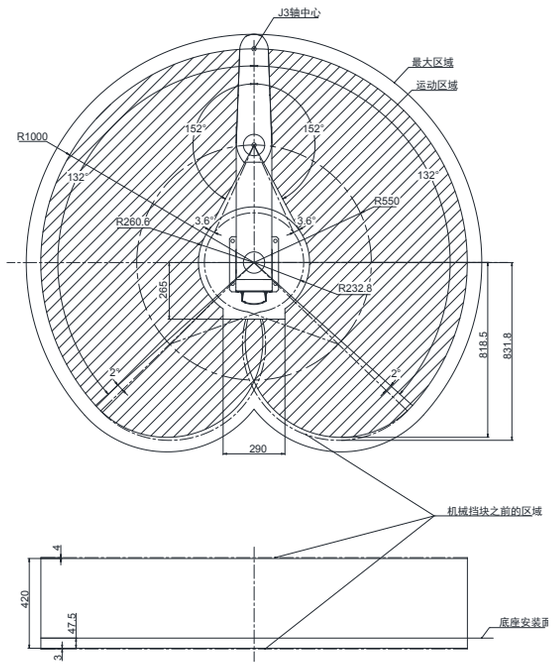


图 4-60 IRB100-6-70Z20TS3 标准规格机型运动范围（单位：mm）

图中所示的最大区域表示末端夹具半径为 60 mm（IRS100 系列）/68mm（IRS100-20）以下的状况。

第 5 章 汇川机器人控制柜（IRCB10，IRCB300）

汇川机器人控制柜现有 IRCB10 系列和 IRCB300 系列。型号说明如下：

5.1 规格说明

5.1.1 型号说明

1 IRCB10 系列

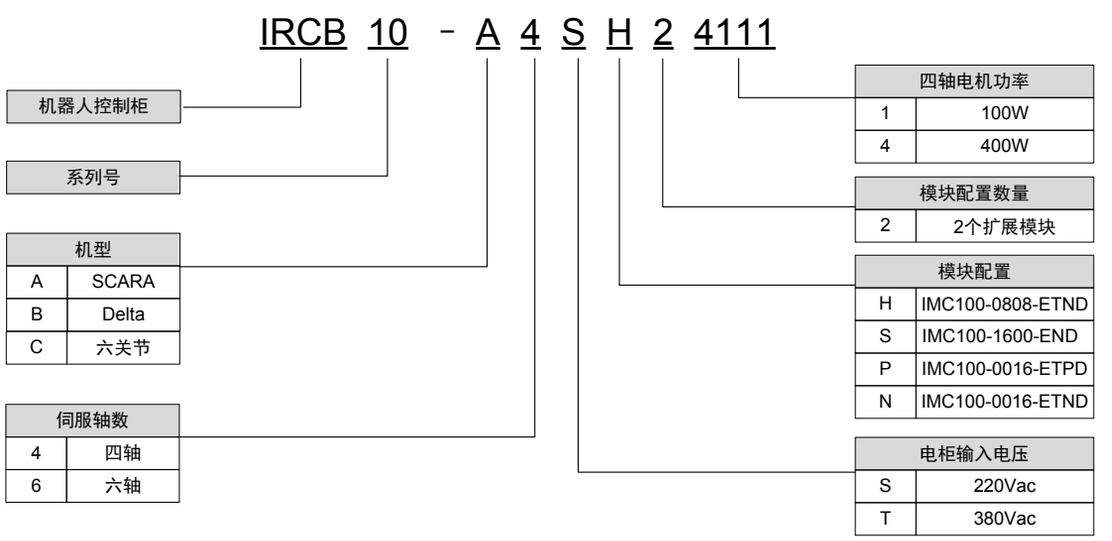


图 5-61 IRCB10 系列型号说明

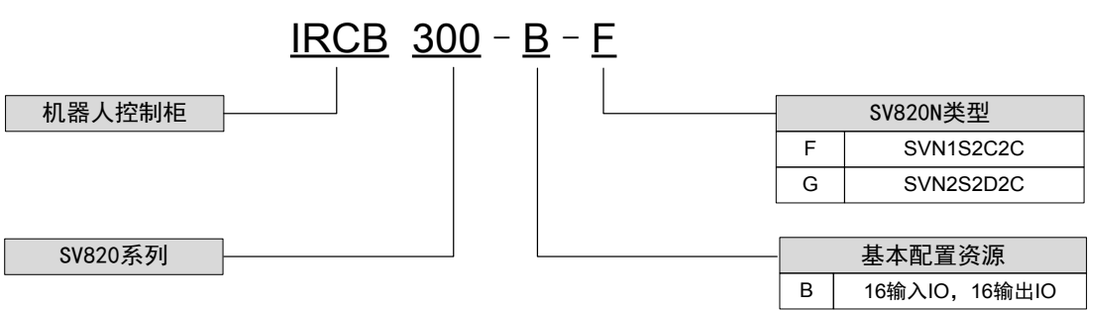
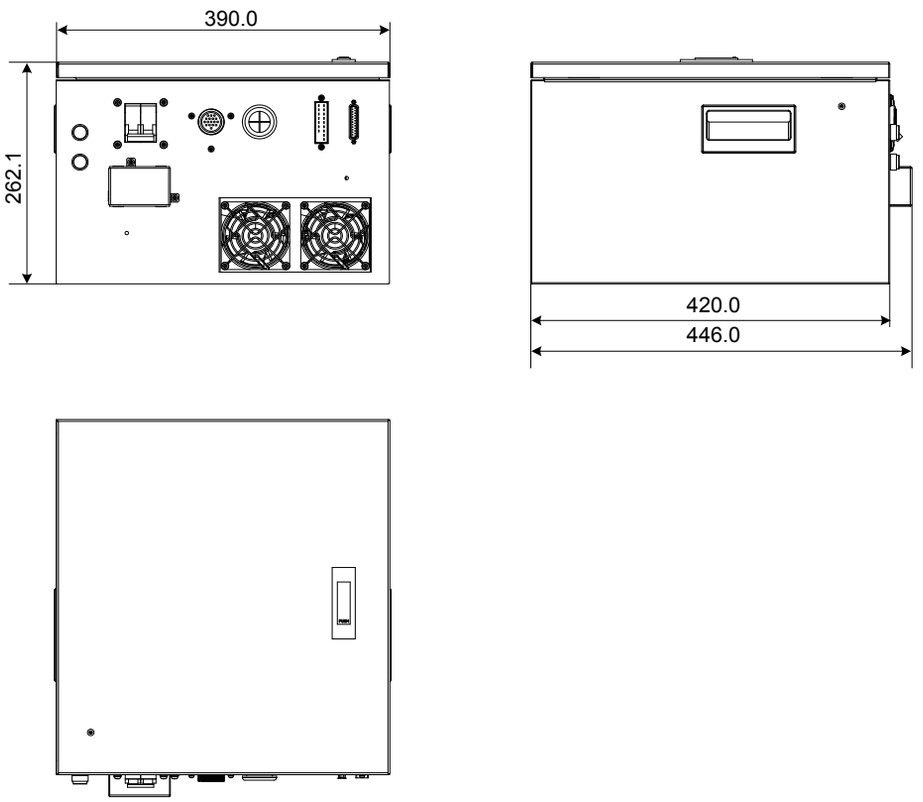


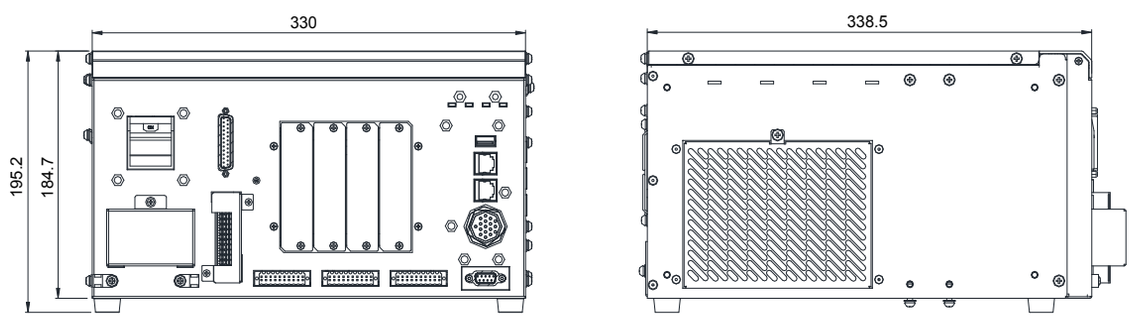
图 5-62 IRCB300 系列型号说明

5.1.2 尺寸

■ IRCB10 系列



■ IRCB300 系列



5.1.3 规格参数

1 IRCB10 系列

■ IRCB10 系列机器人控制器规格参数

项 目	规 格
输入电源	单相 AC200V-240V，+10~-10%，50/60Hz
控制轴数	4 轴
标配置 IO 点数	16 点输入，16 点输出（NPN）
以太网	一个通道扩展
IRLINK	一个通道扩展
控制器	IMC100R-E-X1
绝缘电阻	100 MΩ 以上
环境温度	5 ～ 40° C （不应有过大温度变化）
环境相对湿度	10 ～ 80% （不得结露）
电快速瞬变 脉冲群抗扰度	2 kV 以下
静电抗扰度	6 kV 以下
体积（mm）	420*446*263
重量（KG）	20

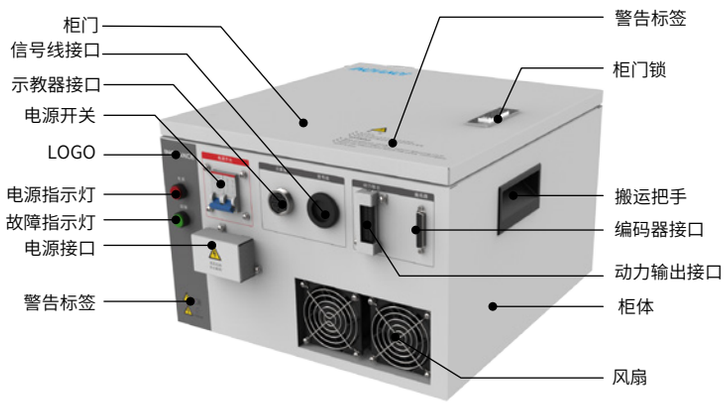
■ IRCB300 系列

项目	规格	
运动方式	PTP 方式，CP 方式（连续路径控制） 支持空间直线插补，空间圆弧插补	
控制功能	编程语言自定义	GUI 开发环境（易用化，集成化，工艺内嵌）
	关节控制	同时控制至少 4 轴，AC 伺服控制器
	速度控制	PTP 控制时，可在 1-100% 间编程控制；CP 控制时， 可以自由指定实际速度控制
	加速度控制	PTP 控制时，可在 1-100% 间编程控制；CP 控制时， 可以自由指定实际加速度控制
PLC 功能	支持 IEC61131-3 标准，PLC 编程功能，实现对所有 I/O 资源的灵活编程，可对输入口的检测、对输出口的控制和与系统内核进行数据交换	
安全功能	异常停止开关，速度偏差过量检测，位置偏差过量检测，存储异常检测，CPU 异常检测，带键模式切换，机械锁定	
动态制动功能	动态制动功能	
记忆容量	程序类：>16Mb；点数据类：最大 10000 点	
存储方式	内置 8G 容量 SD 卡，运行程序备份存储	
运行任务数	可同时运行 4 个机器人程序（一个运动程序、三个非运动程序）	
示教方法	远程示教，手持示教器	
输入电源	单相 AC200V-240V， 8A, 50/60Hz，	
控制轴数	4 轴机器人 + 4 外部扩展轴	
标准 IO	16 点输入，16 点输出（NPN 型）	
以太网	两个以太网接口	
EtherCAT	一个外部扩展轴接口	
IR-LINK	一个外部扩展模块扩展接口	
示教器接口	一个航插接口	
USB	一个 USB2.0 接口	
RS485	一个 485 总线接口	
扩展卡扩展	4 个卡位（可选配 IO、DA、AD、编码器等）	
SAFETY	带急停、安全回路接入、安全门功能	
体积 (长、宽、高)	330*340*195mm	
重量	13kg	

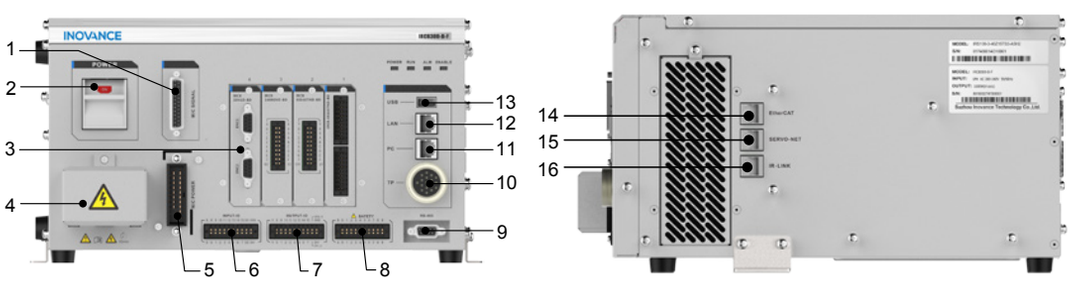
5.2 端口说明

5.2.1 各端口介绍

■ IRCB10 系列



■ IRCB300 系列



序号	名称	功能说明
1	机器人信号线接口	采集机器人位置信号，用专用线缆连接到机器人本体
2	控制柜微断开关	用于机器人系统的上下电
3	机器人扩展卡接口	用于级联扩展卡，依序扩展，可级联 IO\DA\AD\ 编码器扩展卡等
4	控制柜电源输入接口	用于接入外部单相 220VAC 电源
5	机器人动力线接口	用专用线缆连接到机器人本体
6	输入接口	用户输入信号接口
7	输出接口	用户输出信号接口
8	SAFETY 接口	用于外部急停等信号接入
9	标准 RS232、RS485 接口	如适用 modbus 等通讯协议
10	TP 接口	示教器接口
11	PC 接口	如可连 PC 电脑等
12	LAN 接口	可连机器视觉设备等
13	USB 接口	用于系统升级、程序加载等
14	控制柜指示灯	指示系统状态
15	EtherCAT 接口	外部扩展轴总线接口
16	SV820N 伺服调试接口	可接入后台软件进行参数设置、在线调试诊断等
17	IR-LINK 总线接口	用于扩展外部扩展模块

5.2.2 前面板指示灯说明

下表中， 表示灯灭， 表示灯亮

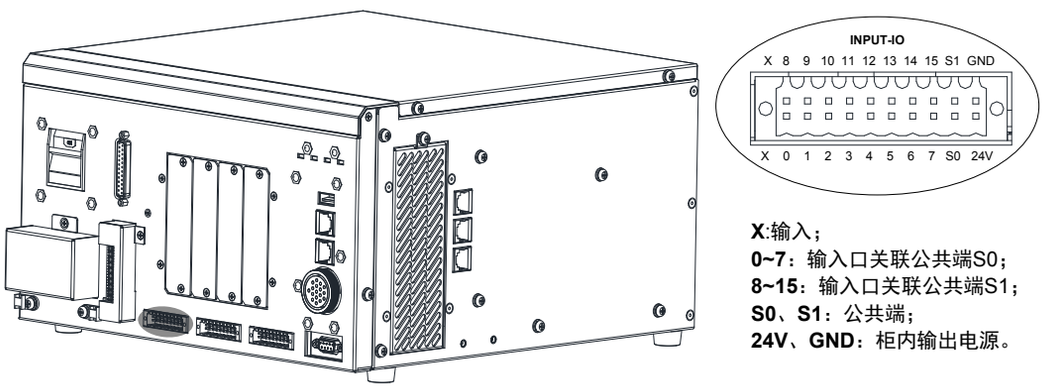
1 控制柜指示灯说明

指示灯状态		状态
POWER 电源指示灯		灯灭：控制柜未通电或控制柜故障。
		灯亮（绿色灯）：控制柜正常输入 220V 交流电，控制柜供电正常。
RUN 程序运行灯		灯灭：无加工程序运行
		灯亮（绿色灯）：加工程序正运行
ALARM 系统报警灯		灯灭：系统无报警
		灯亮（红色灯）：系统故障、报警
ENABLE 系统使能		灯灭：系统未处于使能状态
		灯亮（绿色灯）：系统使能或运行状态下

5.3 输入 IO 接线

5.3.1 输入 I0 接口介绍

1 接口分布图



X:输入；
0~7：输入口关联公共端S0；
8~15：输入口关联公共端S1；
S0、**S1**：公共端；
24V、**GND**：柜内输出电源。

1 输入 IO 技术规格

项目	规格
输入通道	16
输入连接方式	压接式接线端子
输入类型	数字量，可通过公共端选择 NPN 或者 PNP 型输入
输入电压范围 (DC)	0V 或 24V
输入电流 (典型 24V)	4mA
最大输入电压	30VDC
ON 电压	>10VDC
OFF 电压	<5VDC
输入最大信号频率	1K bps
输入阻抗	>6.6KΩ
隔离方式	光电隔离

5.3.2 输入 I/0 接线方法

1 接线规则

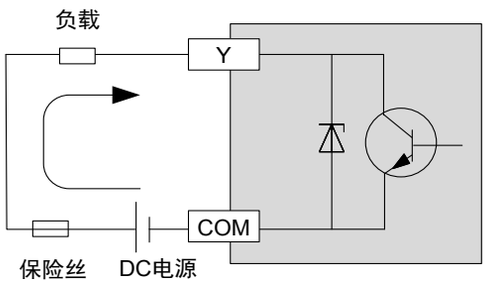
S0，X0~X7 为一组；S1，X8~X15 为一组。

S0/S1 端为公共端，当 S0/S1 接 24V 时，X* 端输入 0V 则输入信号有效（板内光耦导通）；当 S0/S1 接 0V 时，X* 端输入 24V 时则输入信号有效（板内光耦导通），由此可接 NPN 或 PNP 型输出的 IO 设备。

2 NPN 输出与 PNP 输出区别

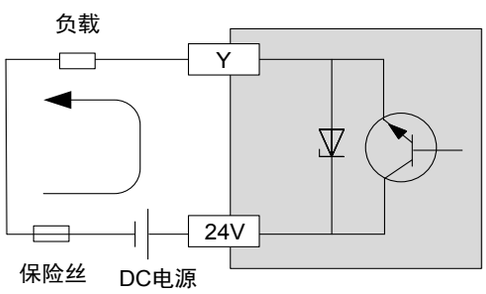
■ NPN 型（漏型）输出 (负公共端)

负载电流流到输出 (Y) 端子，这样的输出称为 NPN 型输出，为低电平输出；



■ PNP 型（源型）输出 (正公共端)

负载电流从输出 (Y) 端子流出，这样的输出称为 PNP 型输出，为高电平输出；



NOTE

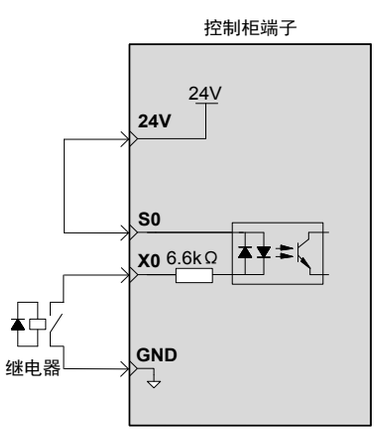
◆ 控制输出低电平的是 NPN 型，控制输出高电平的是 PNP 型。

3 接线示意

以 X0 为例（X1~X7 同 X0）

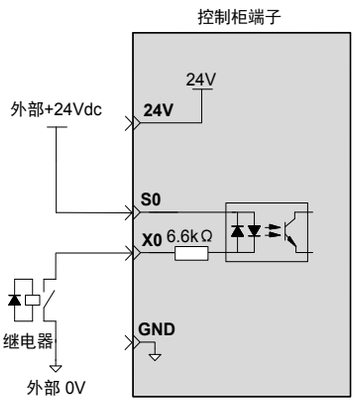
■ 当上位装置为继电器输出时：

1） 使用柜内 24V 电源

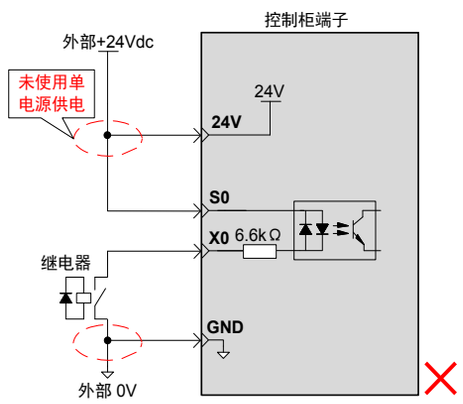


2） 使用外部 24V 电源

正确接线方法如下：

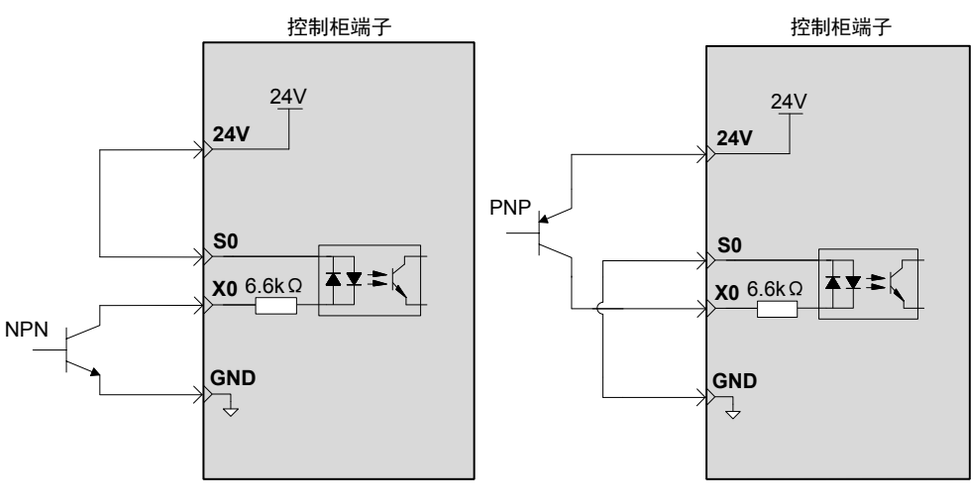


3） 错误接线方法如下：

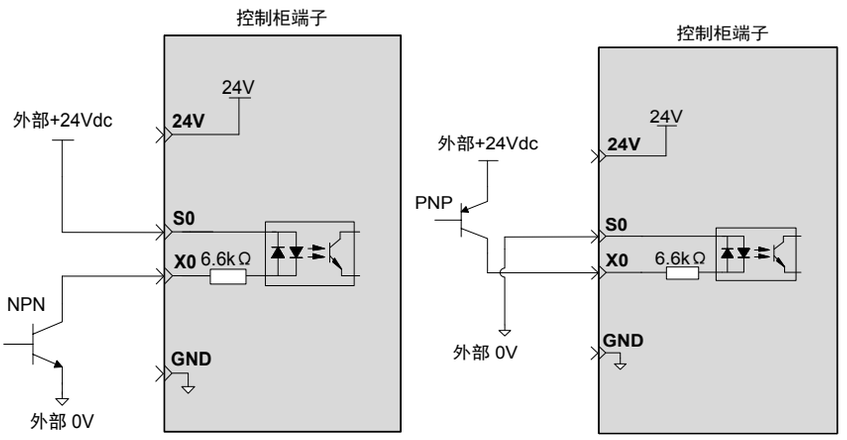


■ 当上位装置为集电极开路输出时：

1） 使用柜内 24V 电源



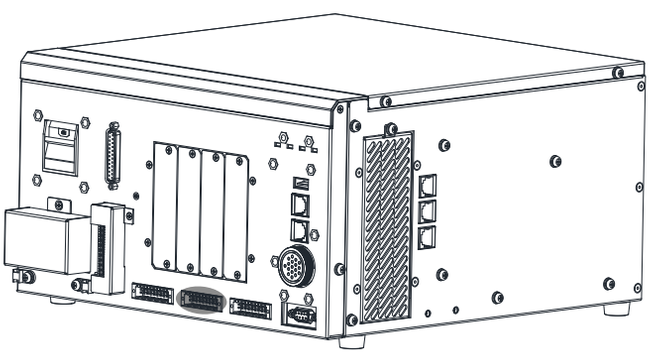
2） 使用外部 24V 电源

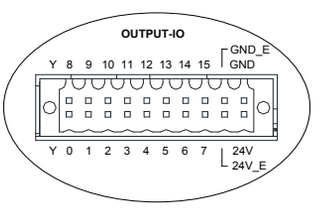


5.4 输出 IO 接线

5.4.1 输出 I0 接口介绍

1 接口分布





Y:输出；
24V_E、GND_E：外接电源输入端
24V、GND:柜内输出电源

2 输出 IO 技术规格

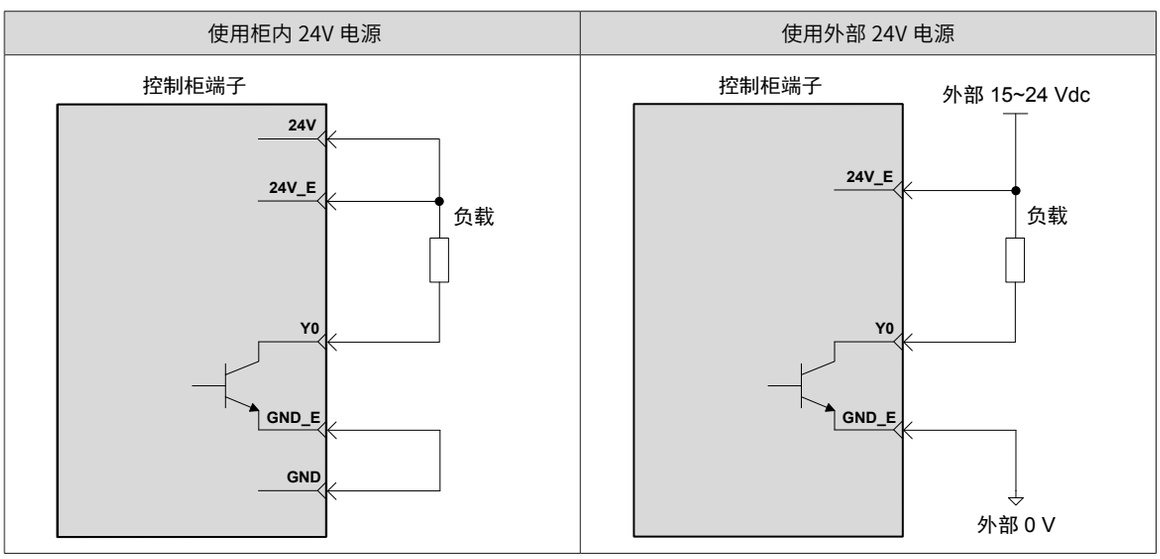
项目	规格
输出通道	16
输出连接方式	压接式接线端子
输出类型	NPN 型
输入电源电压范围 (DC)	15V—30V
最大输出阻抗	<0.1Ω
输出负载电流 (典型 24V)	0.5A
OFF 时最大漏电流	<10μA
输出最大信号频率	1K bps
隔离方式	光电隔离
保护	具有过载、短路保护
隔离方式	光电隔离

5.4.2 输出 I/0 接线方法

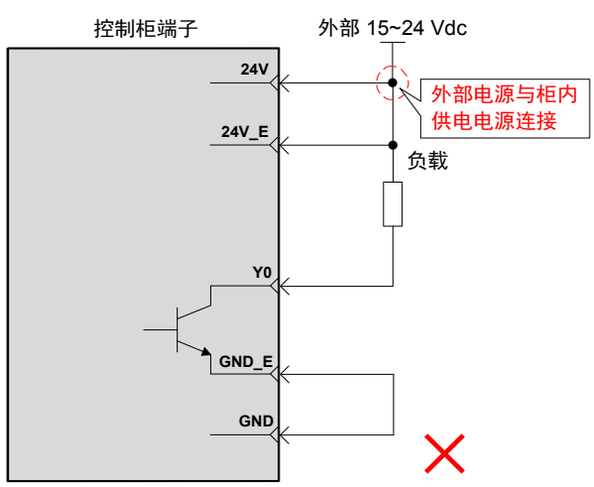
> **注 意**
>
> 为保证输出接口要正常工作，24V_E/GND_E 必须接入 24V 电源；
>
> 24V_E/GND_E 若接柜内电源（即 GND_E 和 GND、24V_E 和 24V 分别短接），需注意 16 个输出 IO 口持续输出总电流不能超过 1.5A，单个 IO 持续输出最大电流为 0.5A；
>
> 当应用需求 16 个 IO 口持续输出总电流大于 1.5A，则不应用柜内电源供电，需采用外部电源供电方式；
>
> 采用外部供电时，单个 IO 持续输出最大电流为 0.5A，持续输出总电流不大于 8A。

1 供电接线

■ 使用柜内 24V 电源



错误接线方法如下：

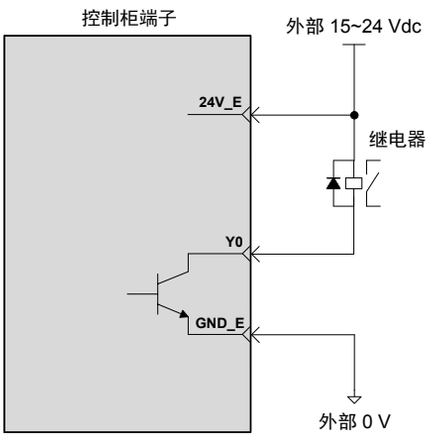


2 接线示意

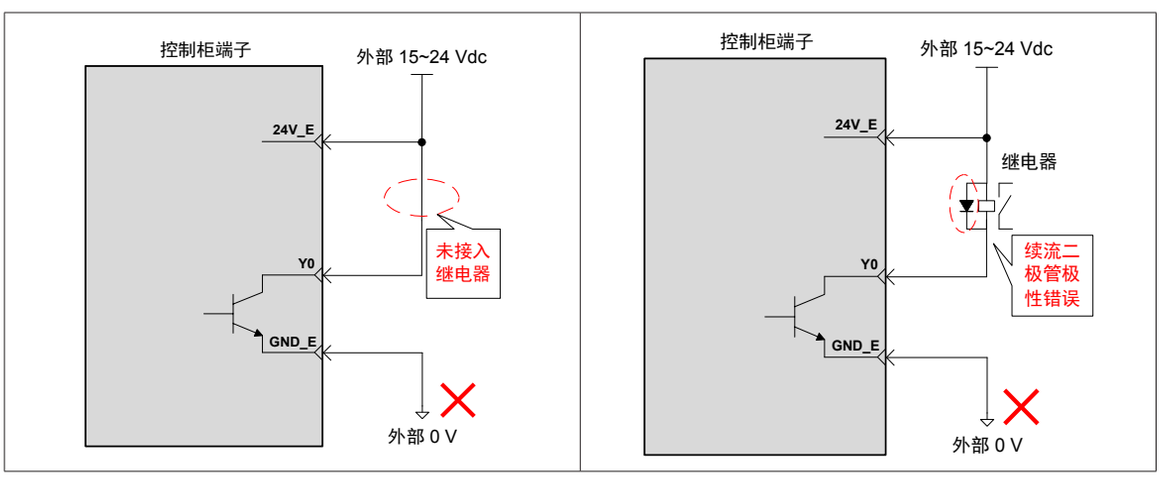
NOTE

◆ 本部分以使用外部电源为例，使用柜内电源时使用方法相同。

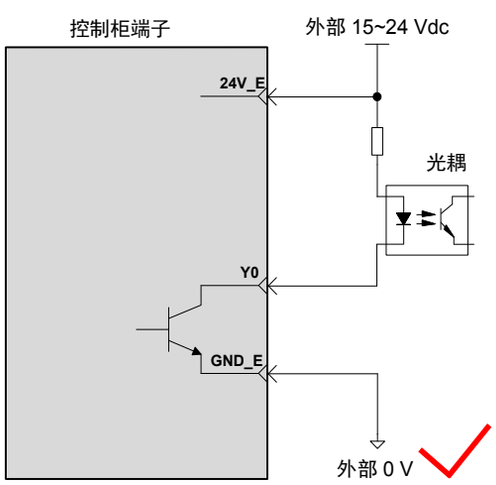
■ 当驱动继电器负载时（请务必接入续流二极管，否则有损坏风险）



错误接法如下：

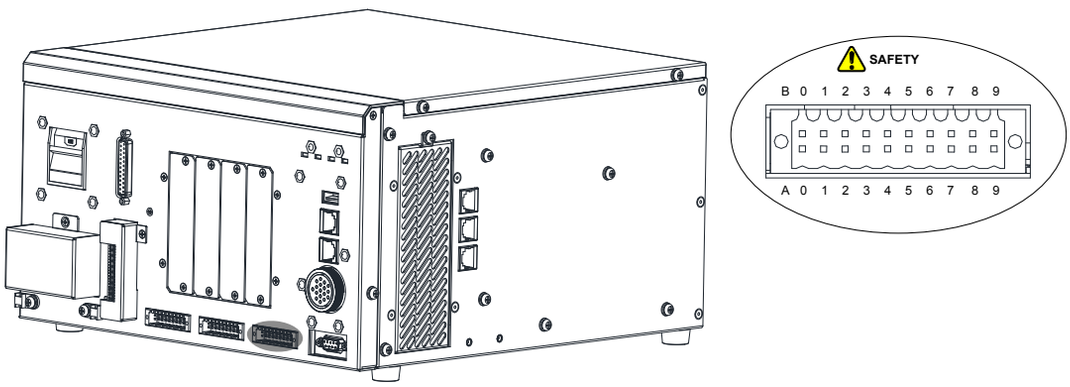


■ 当驱动光耦负载时



5.5 SAFETY 接线

5.5.1 SAFETY 端口定义



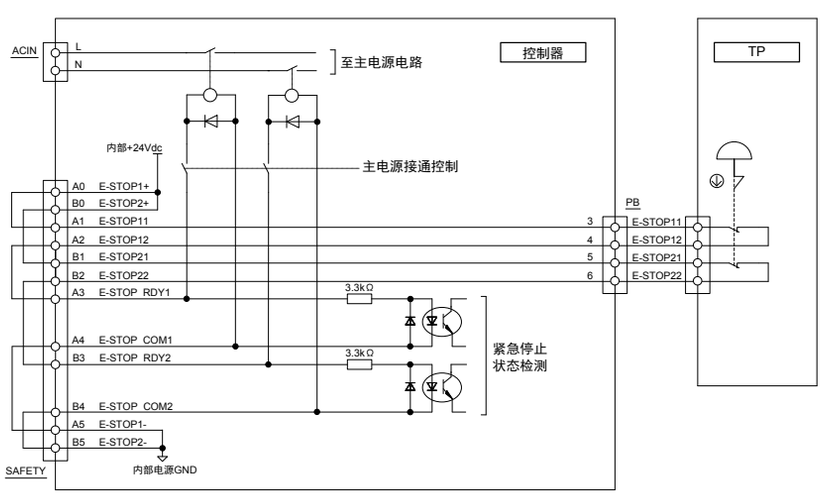
序号	定义	序号	定义	备注
A0	E-STOP_24V	B0	E-STOP_24V	控制柜提供的急停用 24V 电源
A1	E-STOP11	B1	E-STOP21	示教器急停接入点
A2	E-STOP12	B2	E-STOP22	
A3	E-STOP_RDY1	B3	E-STOP_RDY2	外部急停接入点
A4	E-STOP_COM1	B4	E-STOP_COM2	
A5	E-STOP_GND	B5	E-STOP_GND	控制柜提供的急停用 24V 电源地
A6	安全门 +	B6	预留	A8、A9、B6、B7 暂未定义
A7	安全门 -	B7	预留	
A8	预留	B8	模式 +	
A9	预留	B9	模式 -	

5.5.2 SAFETY 接线方法

急停用于紧急事件处理，用常闭开关（如旋钮、蘑菇头形式急停开关）进行串行接入。当按下急停开关，该动作信号通过光耦把信号传给控制柜进行处理，同时会断开伺服的主回路电源，达到紧急制动的目的。

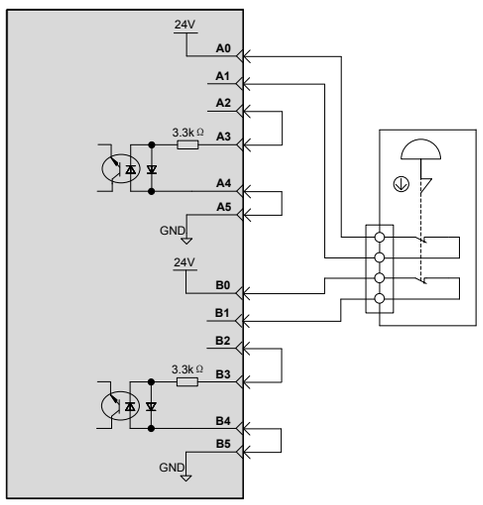
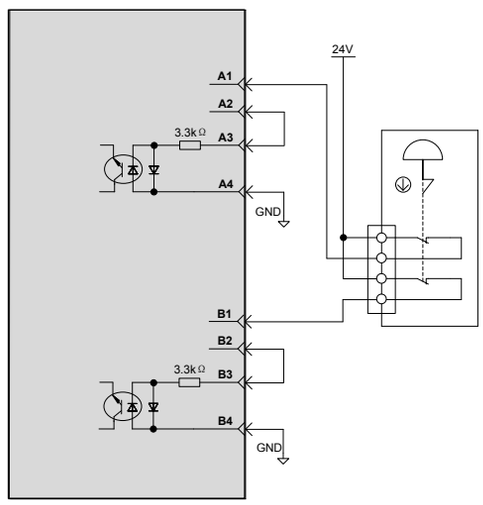
NOTE

◆ 急停是常闭触点连接，为双回路结构，任何一路急停断开都将触发急停功能。

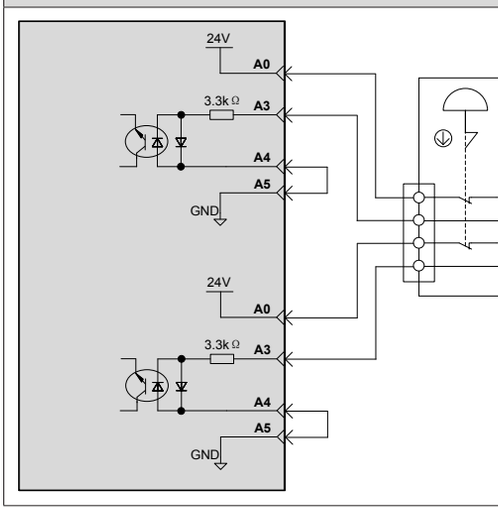
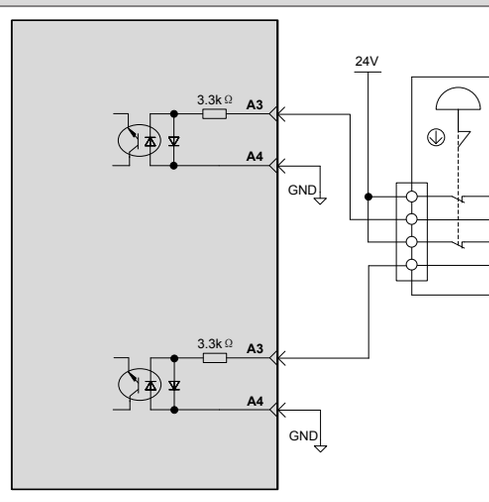


1 急停的拓扑结构图

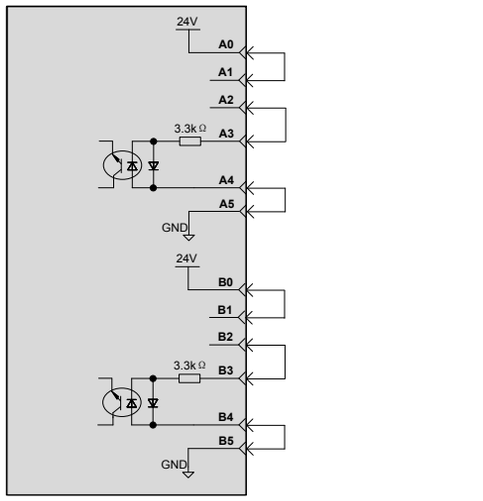
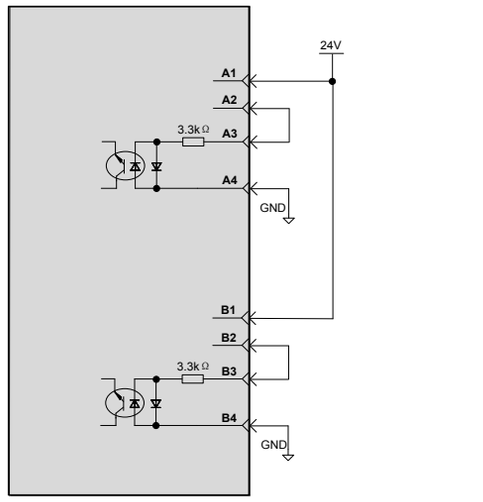
使用外部急停 + 示教器急停的连接（双路 (A/B) 互锁机制）

使用柜内 24V 电源	使用外部 24V 电源
	

■ 仅使用外部急停开关的连接（双路 (A/B) 互锁机制）

使用柜内 24V 电源	使用外部 24V 电源
	

■ 仅使用示教器急停开关的连接（双路 (A/B) 互锁机制）

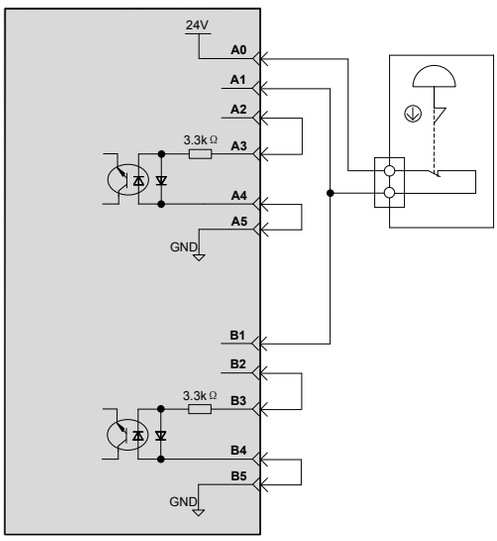
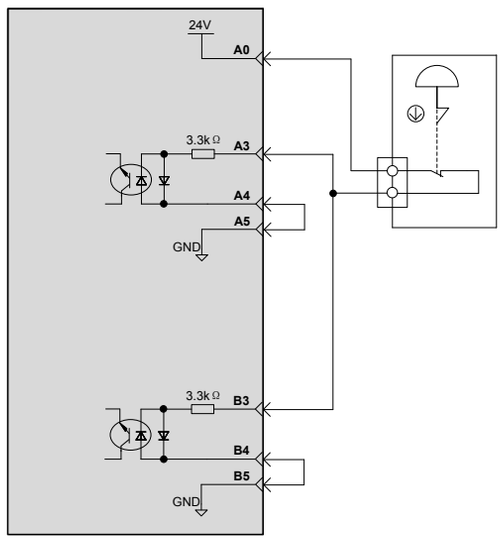
使用柜内 24V 电源	使用外部 24V 电源
	

2 单组急停连接方式

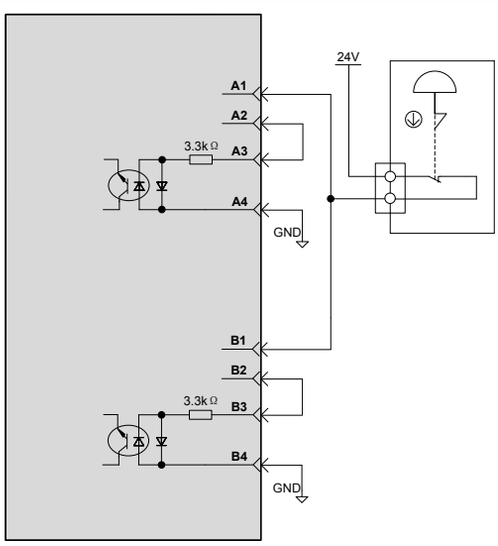
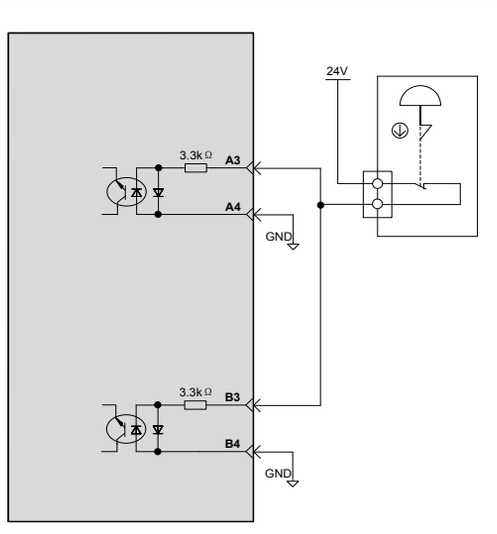
NOTE

◆ 不推荐单组急停连接方式，推荐双路 (A/B) 互锁机制。

■ 使用柜内 24V 电源

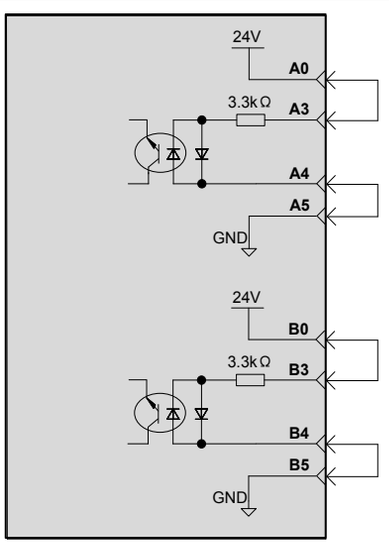
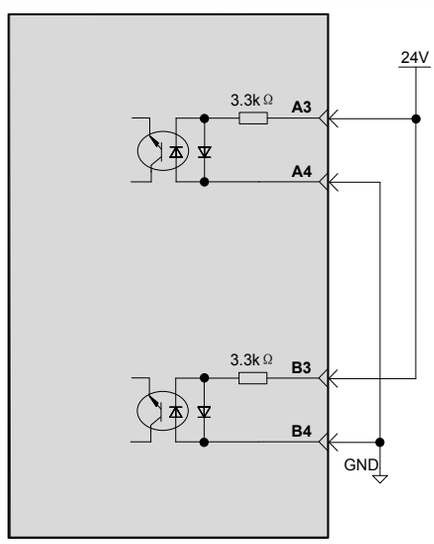
单组急停串入示教器急停	单组急停不串入示教器急停
	

■ 使用外部 24V 电源

单组急停串入示教器急停	单组急停不串入示教器急停
	

3 不使用任何物理急停的连接

> **注 意**
>
> ◆ 强烈不推荐该用法，遇到紧急情况时无法进行机器人紧急制动处理。

使用柜内 24V 电源	使用外部 24V 电源
	

5.5.3 安全门、模式接线方法

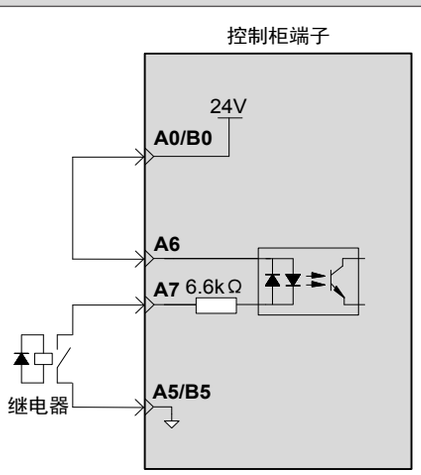
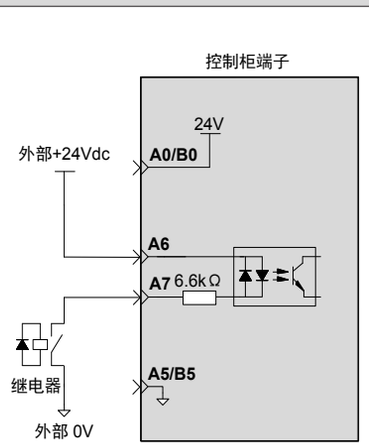
安全门功能可用于控制机器人工作时的防护栏、门。

模式用于选择机器人的运行方式（示教、再现模式）。

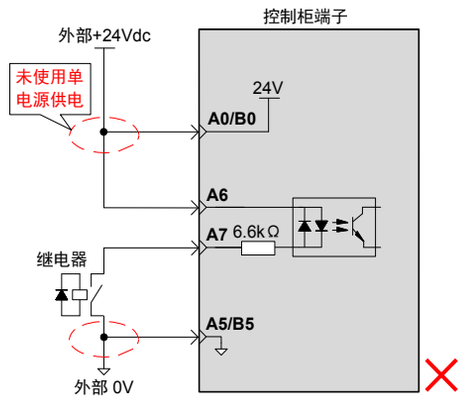
NOTE

◆ 模式与安全门的外部接线方式相同，图中以安全门接线方法为例。

4 当上位装置为继电器输出时

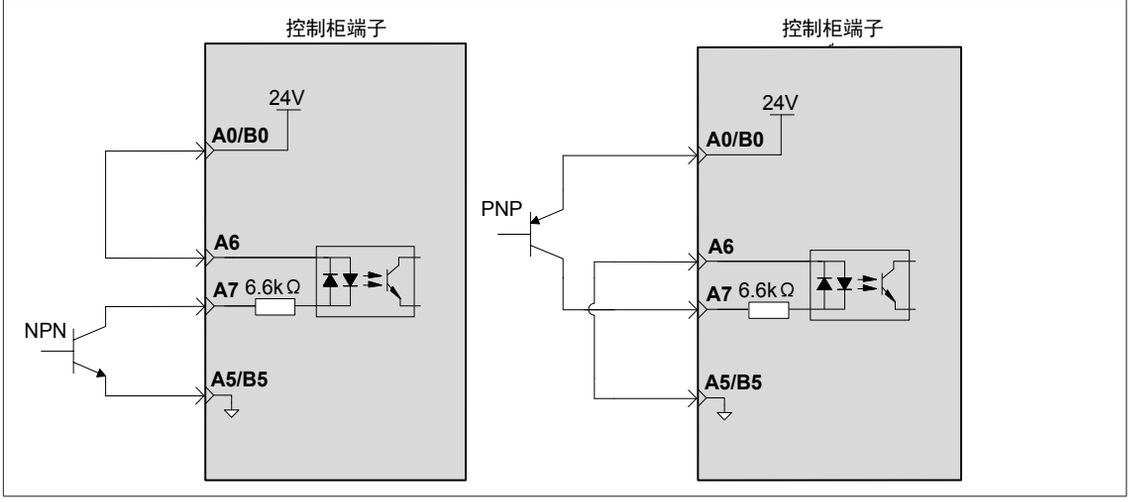
使用柜内 24V 电源	使用外部 24V 电源
	

■　错误接线方法如下：

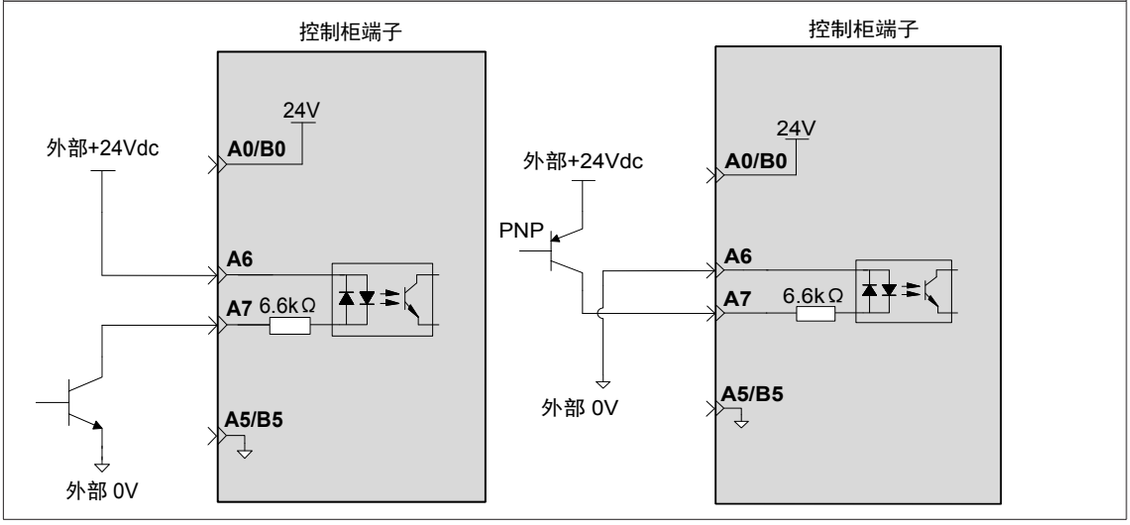


5 当上位装置为集电极开路输出时

使用柜内 24V 电源

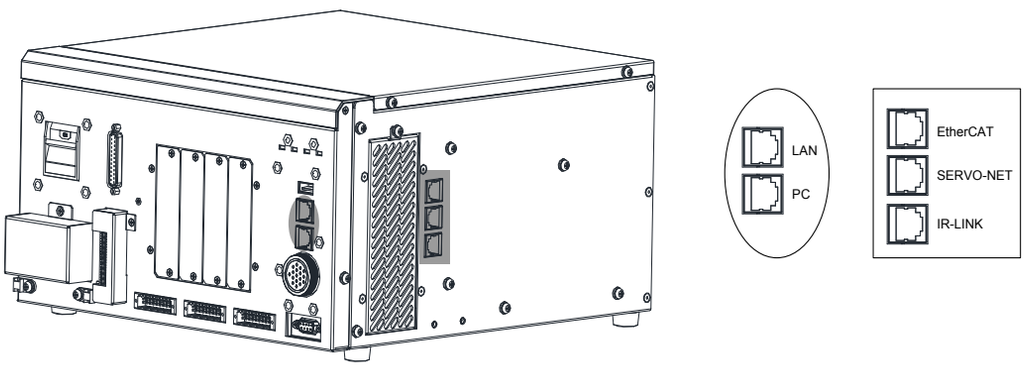


使用外部 24V 电源



5.6 通讯口接线说明（IR-LINK、EtherCAT 、以太网）

5.6.1 通讯接口介绍



1 IR-LINK 接口定义

序号	定义	描述
1	IR-LINK_R-	数据接收 -
2	IR-LINK_R+	数据接收 +
3	IR-LINK_T-	数据发送 -
6	IR-LINK_T+	数据发送 +
4、5、7、8	IR-LINK_GND	

2 EtherCAT、以太网接口定义

序号	定义	描述
1	TX+	数据发送 +
2	TX-	数据发送 -
3	RX+	数据接收 +
4	-	-
5	-	-
6	RX-	数据接收 -
7	-	-
8	-	-
外壳	PE	屏蔽

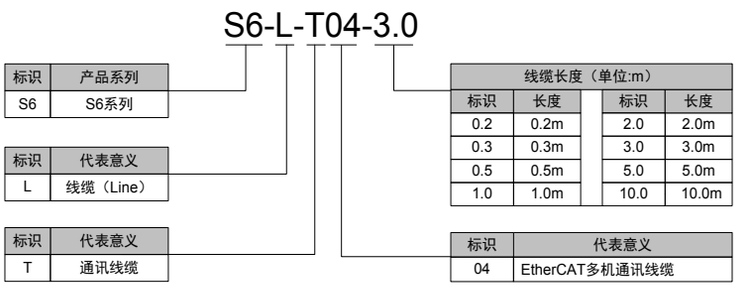
5.6.2 通讯推荐线缆

通讯线缆（适用于 IR-LINK、EtherCAT、以太网连接）推荐使用 5 类屏蔽双绞线，EMC 环境恶劣情况下，推荐使用高柔屏蔽 6 类线。

NOTE

◆ IR-LINK 只能使用直连线。

我司支持的 S6 系列线缆型号说明如下：



1 线缆订货信息

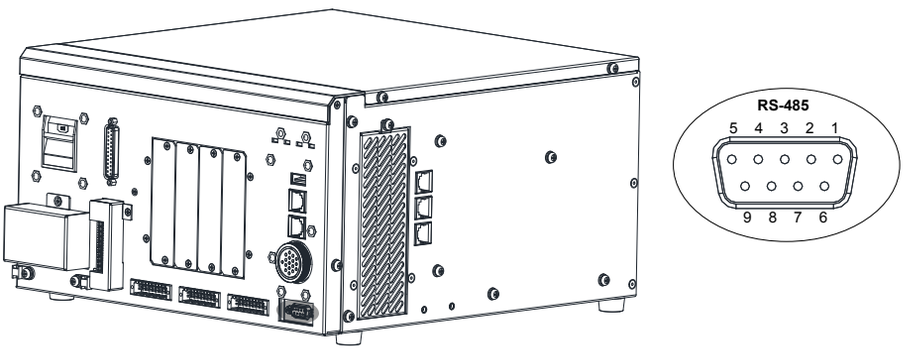
物料编码	线缆型号	规格长度（m）
15040261	S6-L-T04-0.3	0.3
15040262	S6-L-T04-3.0	3.0
15041960	S6-L-T04-0.2	0.2
15041961	S6-L-T04-0.5	0.5
15041962	S6-L-T04-1.0	1.0
15041963	S6-L-T04-2.0	2.0
15041964	S6-L-T04-5.0	5.0
15041965	S6-L-T04-10.0	10.0
15300377	高柔 6 类屏蔽网线 (恶劣 EMC 环境下推荐使用）	5.0
15300378	高柔 6 类屏蔽网线 (恶劣 EMC 环境下推荐使用）	3.0

2 5 类屏蔽双绞线规格说明

项目	详细说明
UL 认证	符合 UL 认证
超五类（CAT.5E）线缆	超五类（CAT.5E）线缆
带双层屏蔽	编织网屏蔽层（覆盖率 85%）、铝箔屏蔽层（覆盖率 100%）
环境适应性	使用环境温度：-30℃ ~ 60℃；耐工业机油、耐酸碱腐蚀。

5.7 RS485 接线

5.7.1 RS485 接口介绍



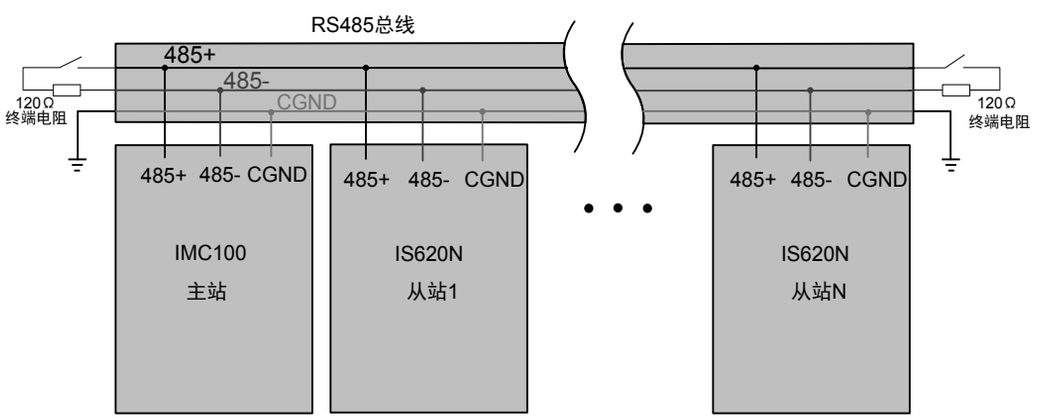
1 接口定义 -DB9 信号定义

针脚号	名称	说明
2	RS232-TX	RS232 信号
3	RS232-RX	
5	RS232-GND	
4	RS485-	RS485 总线信号
9	RS485+	
8	CGND	
1,6,7	-	-

5.7.2 RS485 接线方法

1 RS485 拓扑结构

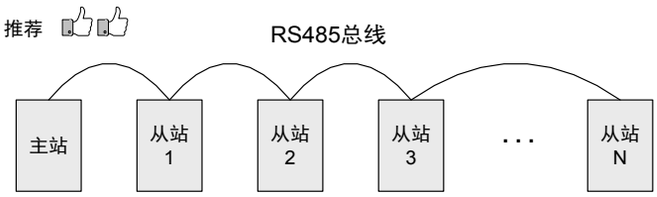
RS485 总线连接拓扑结构如 下图所示，，485 总线推荐使用带屏蔽双绞线连接。485+、485- 采用双绞线连接，只在总线两端分别连接 120Ω 终端匹配电阻防止信号反射；所有节点 485 信号的参考地连接在一起；最多连接 128 个节点，每个节点支线的距离要小于 3m。



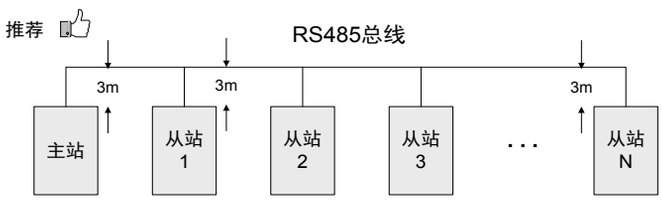
2 多节点拓扑结构

当节点数较多时，485 总线必须采用菊花链连接方式。如果需要分支线连接，总线到节点间的分支长度越短越好，建议不超过 3m，坚决杜绝星型连接。常见总线结构示意图如下：

■　菊花链连接



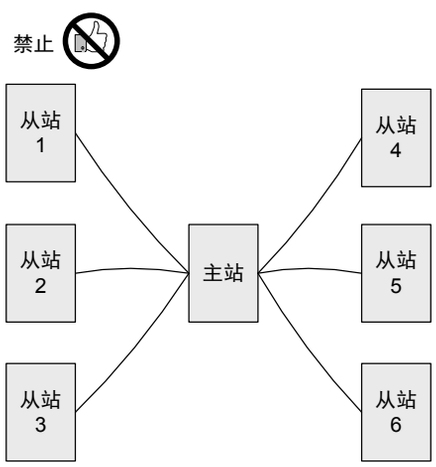
■　分支线连接



NOTE

◆ 分支线建议不要超过 3m。

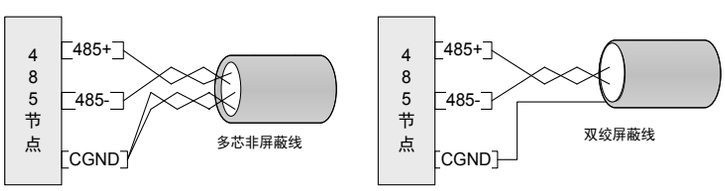
■　星形连接（禁止使用）



3 多点接线方式

■　对端口有 CGND 接线点的节点

请检查现场 485 总线是否包含三根线缆，且接线端子没有接反或者接错。如果使用的是屏蔽线缆，尤其需注意，屏蔽层也必须接 CGND 端子，在任何节点或者中途位置，除了接节点的 CGND，屏蔽层都禁止接 其它任何地方（包括现场机壳、设备接地端子等都不能接）。由于线缆的衰减作用，建议对连接长度大于 3m 的线缆均使用 AGW26 或更粗的线缆，建议 485+ 和 485- 连接线缆使用双绞线缆。



a 多芯非屏蔽线　　b 双绞屏蔽线

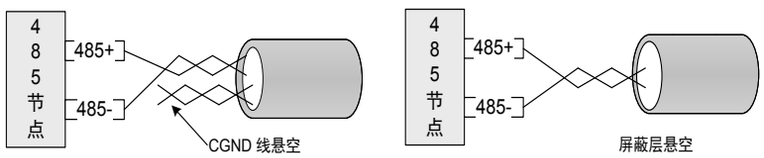
推荐接线线缆 1：带非屏蔽双绞线缆的多芯线缆，取其中一对双绞线为 485+ 和 485- 的连接线，其它多余线缆拧在一起作为 CGND 的连接线。

推荐接线线缆 2：带屏蔽层的双绞线缆，双绞线作为 485+ 和 485- 的连接线，屏蔽层作为 CGND 的连接线。

对于采用屏蔽线作为连接线缆的场合，尤其需注意，屏蔽层只能接 CGND，不能接现场大地。

■ 对端口有 CGND 接线点的节点

不能简单的将 CGND 或者屏蔽层直接接到节点的 PE 上，需按如下方法进行处理：



a 多芯非屏蔽线　　b 双绞屏蔽线

处理方法一：在这个节点其它端口寻找是否有与 485 电路共用的参考地，如果有，总线的 CGND 线缆（屏蔽层）直接接到这个 Pin 脚即可；

处理方法二：在节点单板上找到 485 电路的参考地，引线出来接 CGND 或者屏蔽层；

处理方法三：如果实在找不到 485 电路的参考地，如上图 CGND 线缆或者屏蔽层悬空，同时使用额外的接 地线将这个节点和其它节点的 PE 连起来。

5.8 IR-LINK 总线扩展

扩展卡、整机扩展模块是针对客户应用开发的可选配采购件，它基于汇川自定义总线协议：IR-LINK，具有扩展灵活、功能强大、易于使用等特点。可选的组件包括 IO、DA、AD、编码器、脉冲模块等。

5.8.1 扩展卡、整机扩展模块

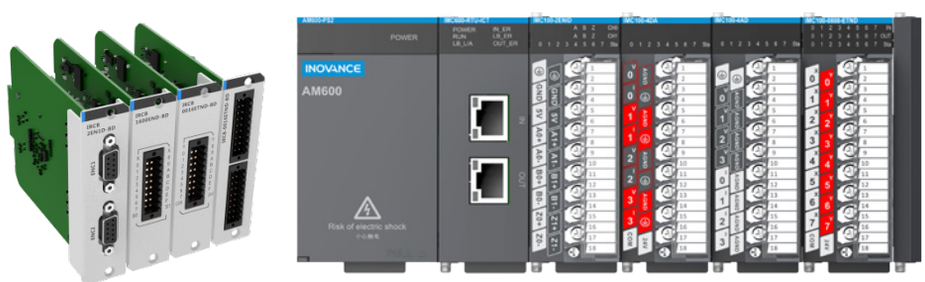


图 5-63 扩展卡、整机扩展模块形态

1 扩展卡种类

订货编码	型号	名称	软件中对应的配置选择
01650010	IRCB-2EN1D-BD	2 通道差分输入增量编码器扩展卡	2ENC
01650011	IRCB-1616ETND-BD	16 路输入与 16 路 NPN 型输出通用 IO 扩展卡	1616
01650012	IRCB-4DA-BD	4 通道电压或电流模拟量转换输出扩展卡	4DA
01650013	IRCB-4AD-BD	2 通道电压和 2 通道电流模拟量转换输入扩展卡	4AD
01650015	IRCB-1600END-BD	16 路输入通用 IO 扩展卡	1600
01650016	IRCB-0016ETND-BD	16 路 NPN 型输出通用 IO 扩展卡	0016
01650017	IRCB-0016ETPD-BD	16 路 PNP 型输出通用 IO 扩展卡	0016

2 整机扩展模块种类

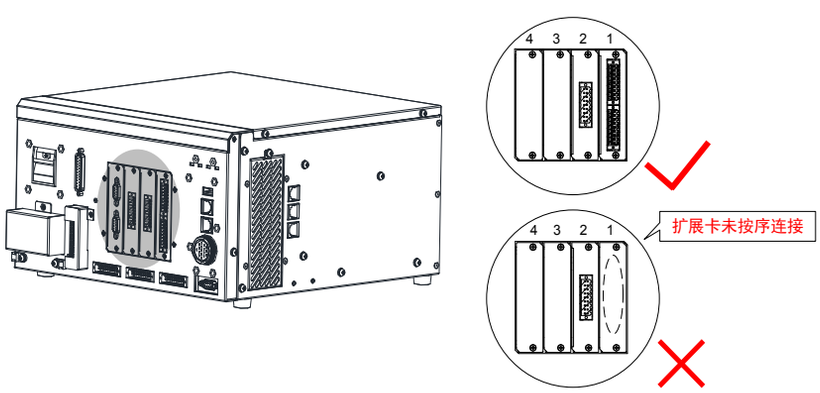
订货编码	型号	名称	软件中对应的配置选择
01440010	AM600-PS2	电源模块	-
01650001	IMC100-RTU-ICT	IMC100 系列通讯扩展模块	RTU
01650002	IMC100-8AD	4 通道电压和 4 通道电流模拟量转换输入扩展模块	8AD
01650003	IMC100-4DA	4 通道电压或电流模拟量转换输出扩展模块	4DA
01650004	IMC100-0808-ETND	8 路输入 8 路输出通用 IO 扩展模块	0808
01650005	IMC100-2ENID	2 通道差分输入增量编码器扩展模块	2ENID
01650006	IMC100-1600-END	16 输入通用 IO 扩展模块	1600
01650007	IMC100-0016-ETPD	16 路 PNP 型输出通用 IO 扩展模块	0016
01650008	IMC100-0016-ETND	16 路 NPN 型输出通用 IO 扩展模块	0016

5.8.2 扩展方式

> **注 意**
>
> ◆ 在进行扩展卡、扩展模块的拆装时，必须将系统使用的外部供应电源全部断开之后再执行操作。如果未全部断开电源，有可能导致模块故障及误动作。

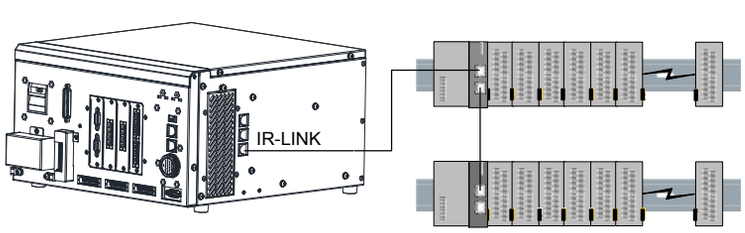
1 扩展卡

依序插入扩展卡，控制柜前面板总共 4 个卡位（有相应数字标识），1 号卡位空缺的情况下需首先插入 1 号卡位，只有在 1 号卡位已有卡的情况下，才可以插入 2 号卡，依次类推。如下图所示：



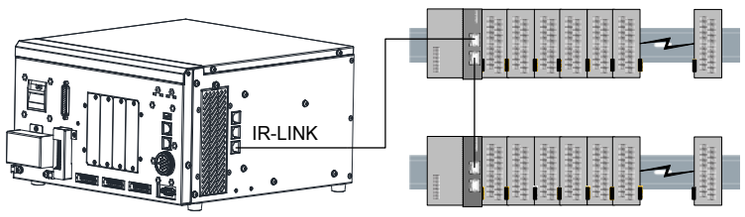
2 扩展卡 + 整机扩展模块

插入 4 张卡后 , 如还需扩展资源，则进行整机扩展模块级联，如下图拓扑结构：



3 整机扩展模块

不插入任何扩展卡，也可进行整机扩展模块扩展，拓扑结构如下：



5.8.3 扩展配置

软件根据物理实际的配置情况（和相应配置拓扑结构强关联）在界面配置中相应模块，应先进行扩展卡适配配置，再进行整机扩展模块的适配确定。

1 配置限制

整机扩展模块的硬件级联需遵循以下规则：每个 RTU 后可提供的功率为 15W。

整机扩展模块功耗

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，再在其后进行级联；除此之外还需考虑资源的限制条件，即软件支持的最大资源数，如下表：

资源	软件支持最大资源数	关联模块类型	单个关联模块占用资源数
输入 IO	64 通道	IMC100-1600-END	16 通道
		IMC100-0808-ETND	8 通道
		IRCB-1600END-BD	16 通道
		IRCB-1616ETND-BD	16 通道
输出 IO	64 通道	IMC100-0808-ETND	8 通道
		IMC100-0016-ETPD	16 通道
		IMC100-0016-ETND	16 通道
		IRCB-0016ETND-BD	16 通道
		IRCB-0016ETPD-BD	16 通道
		IRCB-1616ETND-BD	16 通道
DA	16 通道	IMC100-4DA	4 通道
		IRCB-4DA-BD	4 通道
AD	16 通道	IMC100-8AD	8 通道
		RCB-4AD-BD	4 通道
编码器	4 通道	IMC100-2ENID	2 通道
		IRCB-2EN1D-BD	2 通道

如：系统支持的扩展卡和整机扩展模块最多支持 2 个 IMC100-8AD 模块，4 个 IMC100-4DA 模块，4 个 IMC100-2ENID 模块，8 个 IMC100-0808-ETND 模块（或 4 个 IMC100-0016-ETND + 4 个 IRCB-1600END-BD 模块），以此类推。

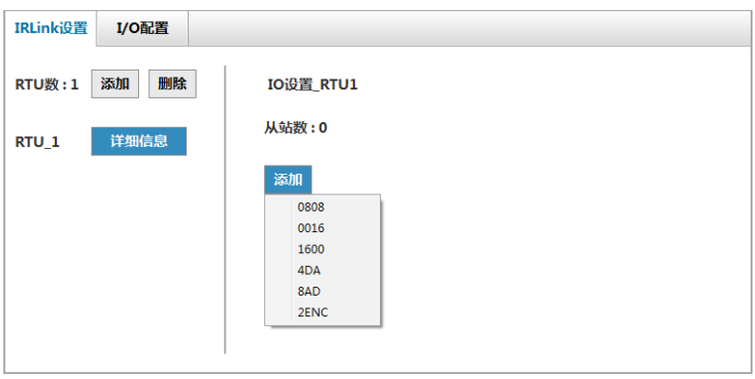
2 界面配置

在连接好硬件设备后，需要在软件中设置 IR-LINK 配置。

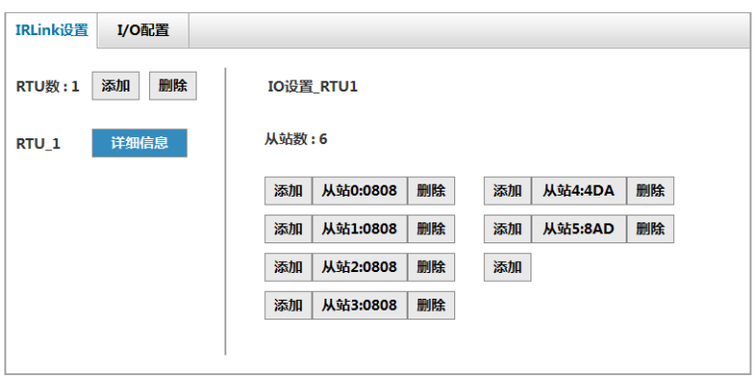
点击页面左侧的“添加”按钮，会自动产生一个 RTU，该 RTU 的详细信息会在右侧显示。最多添加五个扩展模块 RTU。

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

根据实际连接，在此处添加扩展模块。



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



5.9 压接信号线做线指导

线针结构如图示：

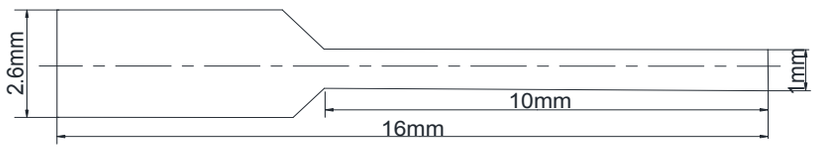


图 5-64 9025870000 线针尺寸图

推荐线针规格参数

汇川物料编码	15074039
厂家	魏德米勒
型号	9025870000
适配线径	0.50 mm^2/20-24AWG
颜色	桔黄
包装量	约 120PCS/ 包
材料	无氧铜合金、PVC 塑料
表面处理	镀锡
载流能力	最大电流 8A

推荐压线工具：环箍式手动压接工具（适配 28-10AWG）

（推荐型号：RUBICON（罗宾汉） RKY-126 ）

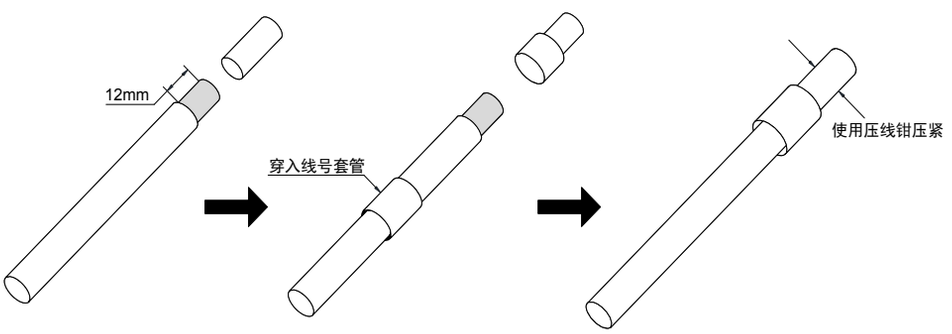
■ 线缆制作步骤：

剥除电线绝缘皮，去除外绝缘皮长度 12mm；

将线缆穿入线号套管；

将线缆导体部分穿过线针圆形孔内，用线针厂商推荐的压接钳压接；

将线针插入 IO 口接线端子排对应孔位内，回拉确认线针与端子排内部弹片锁紧。



5.10 选配扩展

5.10.1 扩展卡、整机扩展模块

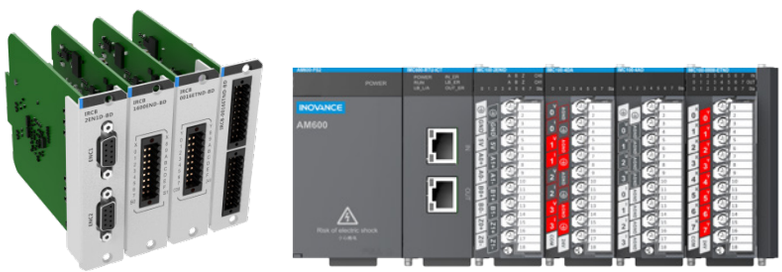


图 5-65 扩展卡、整机扩展模块形态

1 扩展卡种类

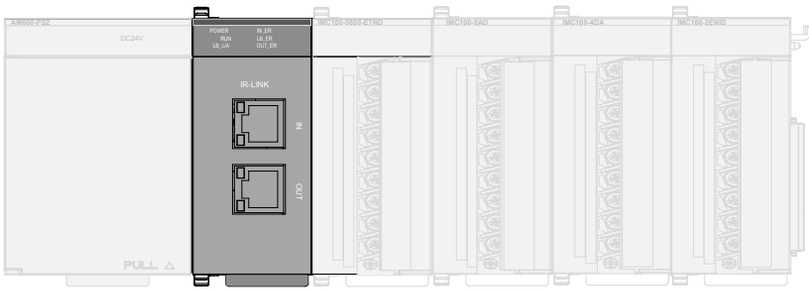
订货编码	型号	名称	软件中对应的配置选择
01650010	IRCB-2EN1D-BD	2 通道差分输入增量编码器扩展卡	2ENC
01650011	IRCB-1616ETND-BD	16 路输入与 16 路 NPN 型输出通用 IO 扩展卡	1616
01650012	IRCB-4DA-BD	4 通道电压或电流模拟量转换输出扩展卡	4DA
01650013	IRCB-4AD-BD	2 通道电压和 2 通道电流模拟量转换输入扩展卡	4AD
01650015	IRCB-1600END-BD	16 路输入通用 IO 扩展卡	1600
01650016	IRCB-0016ETND-BD	16 路 NPN 型输出通用 IO 扩展卡	0016
01650017	IRCB-0016ETPD-BD	16 路 PNP 型输出通用 IO 扩展卡	0016

2 整机扩展模块种类

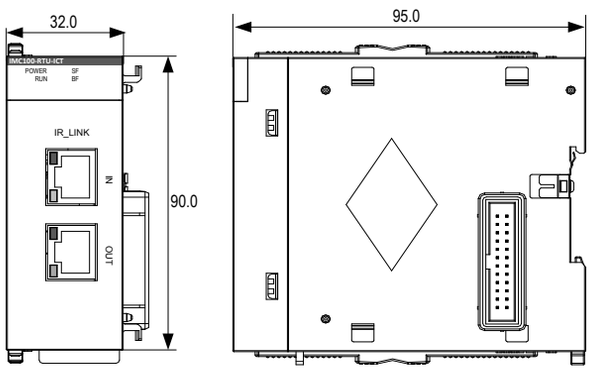
订货编码	型号	名称	软件中对应的配置选择
01440010	AM600-PS2	电源模块	-
01650001	IMC100-RTU-ICT	IMC100 系列通讯扩展模块	RTU
01650002	IMC100-8AD	4 通道电压和 4 通道电流模拟量转换输入扩展模块	8AD
01650003	IMC100-4DA	4 通道电压或电流模拟量转换输出扩展模块	4DA
01650004	IMC100-0808-ETND	8 路输入 8 路输出通用 IO 扩展模块	0808
01650005	IMC100-2ENID	2 通道差分输入增量编码器扩展模块	2ENID
01650006	IMC100-1600-END	16 输入通用 IO 扩展模块	1600
01650007	IMC100-0016-ETPD	16 路 PNP 型输出通用 IO 扩展模块	0016
01650008	IMC100-0016-ETND	16 路 NPN 型输出通用 IO 扩展模块	0016

5.10.2 整机扩展模块安装与接线

■ IR-link 通讯模块（型号：IMC100-RTU-ICT）

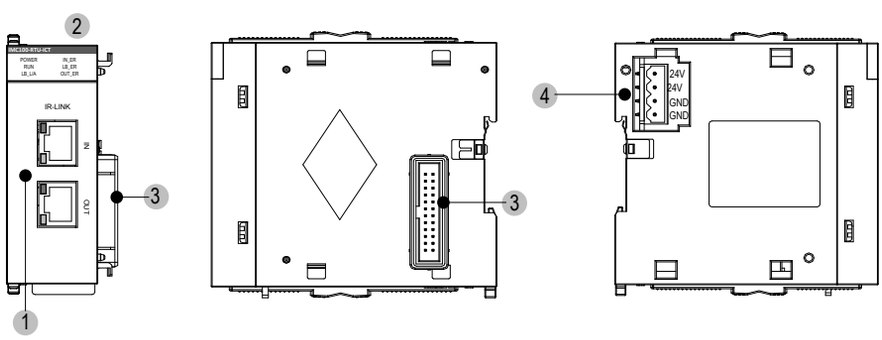


■ 安装尺寸



■ 端口定义

IR-LINK 通讯扩展模块有两个通讯端口，IN：上行端口；OUT : 下行端口；端口线序和标准 100M 以太网兼容，模块与机器人控制器的连接以及模块的扩展连接用标准直连网线即可通讯。



编号	名称	功能定义
1	IN	IR-LINK 输入口 (和机器人主机相连)
	OUT	IR-LINK 输出口（级联扩展模块）
2	信号指示灯	POWER：电源指示灯 RUN：运行指示灯 LB_L/A: IN_ER: LB_ER: OUT_ER:
3	本地扩展模块后级接口	连接后级模块
4	内部 24V 电源输入端子	连接电源模块

■　端子引脚定义

上行端口			下行端口		
管脚	定义	说明	管脚	定义	说明
1	TX-	发送负	1	RX-	接收负
2	TX+	发送正	2	RX+	接收正
3	RX-	接收负	3	TX-	发送负
4	GND	通讯地	4	GND	通讯地
5	GND	通讯地	5	GND	通讯地
6	RX+	接收正	6	TX+	发送正
7	GND	通讯地	7	GND	通讯地
8	GND	通讯地	8	GND	通讯地

■　组网

通过 IR-link 模块可以实现 IMC100 系列控制器各扩展模块的 IR-link 网络，组网方法如下图所示，理论上对站点数没有限制，可以组成无限的通讯网络，满足多点控制的系统需求。

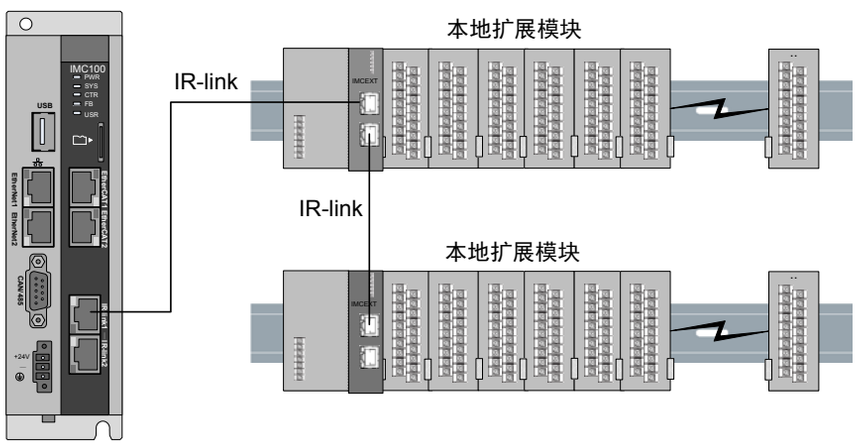
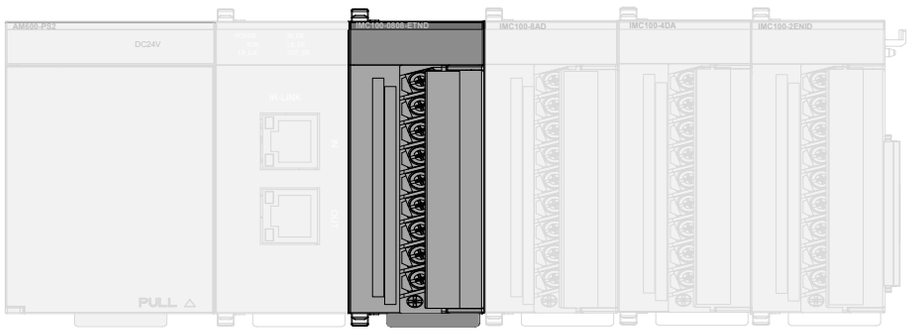
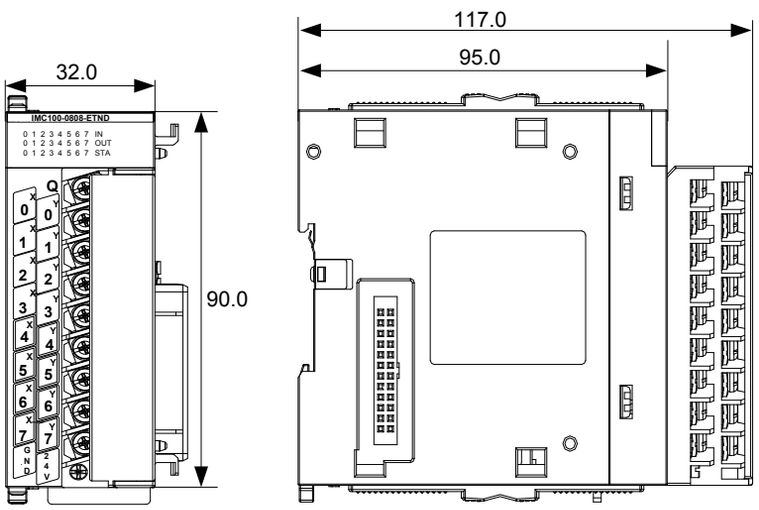


图 5-66　IR-link 通讯模块的组网示意图

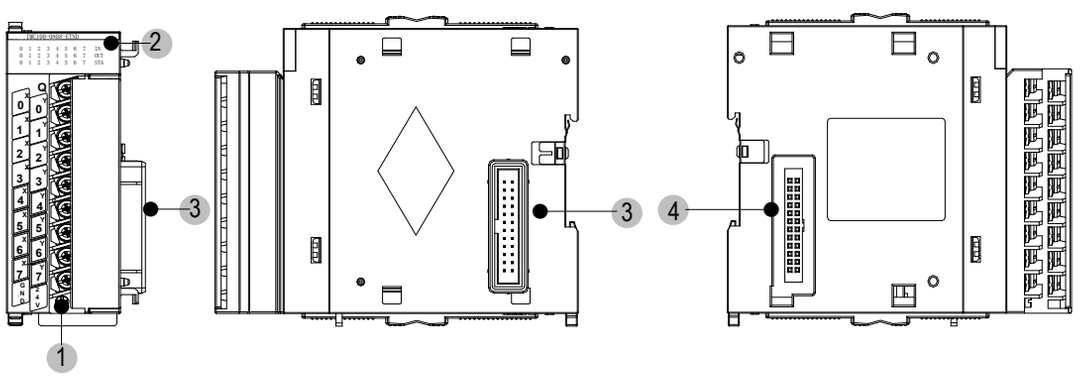
5.10.4 IO 扩展模块（型号：IMC100-0808-ETND）



■ 安装尺寸

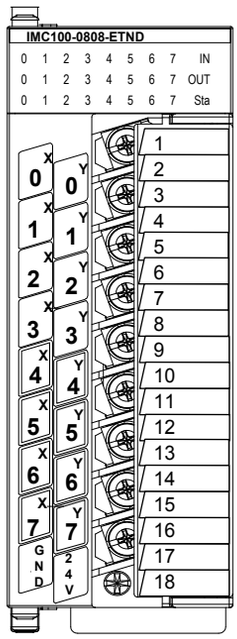


■ 端口定义



编号	名称	功能定义
1	用户端子	8 输入（X0~X7）,8 输出 (Y0~Y7) 端子
2	信号指示灯	IN： 输入端子导通时相应灯亮； OUT：输出端子导通时相应灯亮； STA:
3	本地扩展模块后级接口	连接后级模块
4	本地扩展模块前级接口	连接前级模块

■ 端子信号排列



序号	网络名	类型	功能	备注
1	X0	输入	用户输入 0	源 / 漏型输入
3	X1	输入	用户输入 1	源 / 漏型输入
5	X2	输入	用户输入 2	源 / 漏型输入
7	X3	输入	用户输入 3	源 / 漏型输入
9	X4	输入	用户输入 4	源 / 漏型输入
11	X5	输入	用户输入 5	源 / 漏型输入
13	X6	输入	用户输入 6	源 / 漏型输入
15	X7	输入	用户输入 7	源 / 漏型输入
2	Y0	输入	用户输出 0	漏型输出 (低电平输出)
4	Y1	输入	用户输出 1	漏型输出 (低电平输出)
6	Y2	输入	用户输出 2	漏型输出 (低电平输出)
8	Y3	输入	用户输出 3	漏型输出 (低电平输出)
10	Y4	输入	用户输出 4	漏型输出 (低电平输出)
12	Y5	输入	用户输出 5	漏型输出 (低电平输出)
14	Y6	输入	用户输出 6	漏型输出 (低电平输出)
16	Y7	输入	用户输出 7	漏型输出 (低电平输出)
17	COM	电源	电源地	
18	24V	电源	24V 电源	15V-30V 有效

■ 输入端口规格

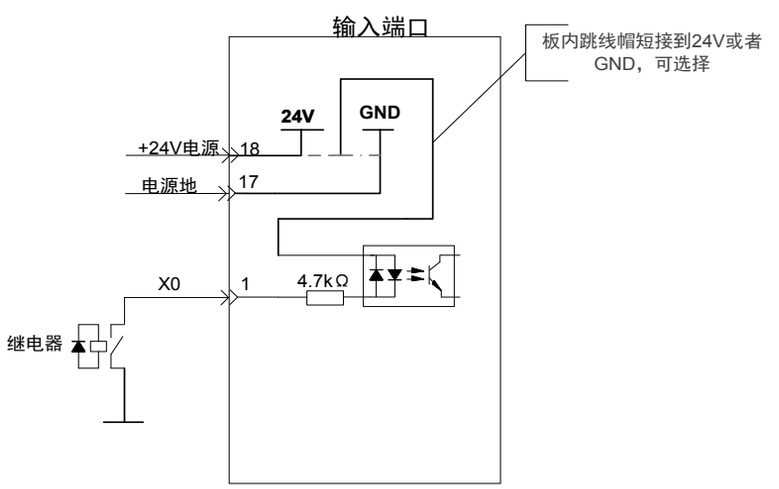
项目	规格
输入通道	8
输入连接方式	栅栏式接线端子
输入类型	数字量输入
输入方式	源 / 漏型
输入电压范围 (DC)	15V--30V
输入电流 (典型 24V)	<10mA
最大输入电流	15mA(长时间该状态下光耦寿命可能减少或损坏)
ON 电压	>15VDC
OFF 电压	<5VDC
最大输入信号频率	100K
输入阻抗	>2K
输入信号的压摆率要求	>18V/μS
输入形式	电压直流输入形式 漏型输入（SINK）：NPN 开集极输入形式； 源型输入（SOURCE）：PNP 开集极输入形式。 （靠板上跳线设置）
隔离方式	光耦隔离
输入动作显示	输入端导通时，相应输入指示灯动作

■ 输出端口规格

项目	规格
输出通道	8
输出连接方式	栅栏式接线端子
输出类型 (接入方式)	NMOS 漏型输出（OD 类型）
OFF 时最大漏电流	<10uA
最大输出信号频率	500KHZ（负载电阻 <1K）
单点最大负载电流	500mA
隔离方式	光耦隔离
输出动作显示	输出导通时，相应输出指示灯动作

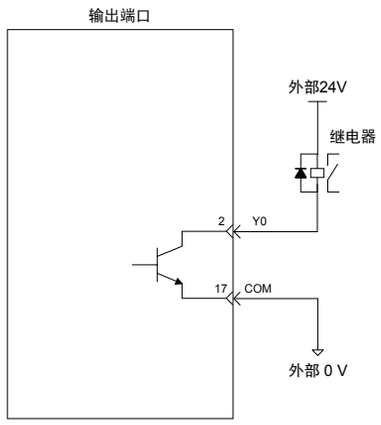
外部接线

■ 输入端口接线

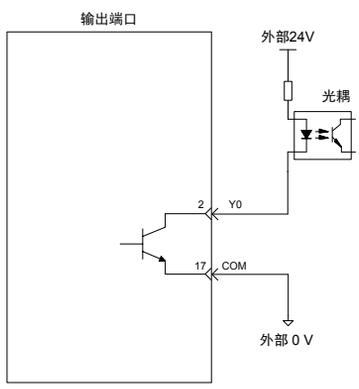


■ 输出端口接线

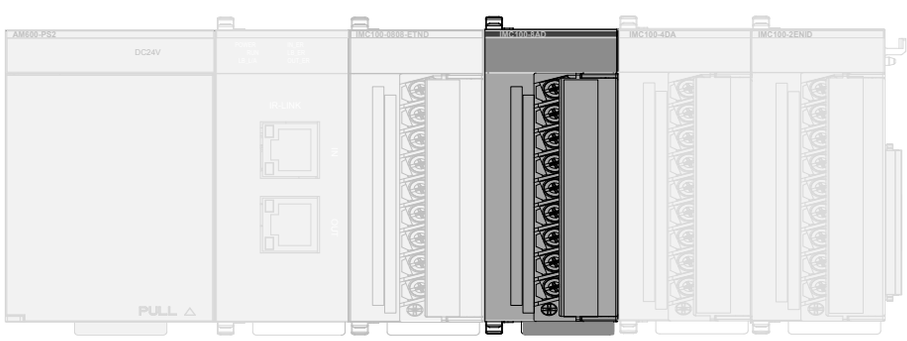
接外部继电器



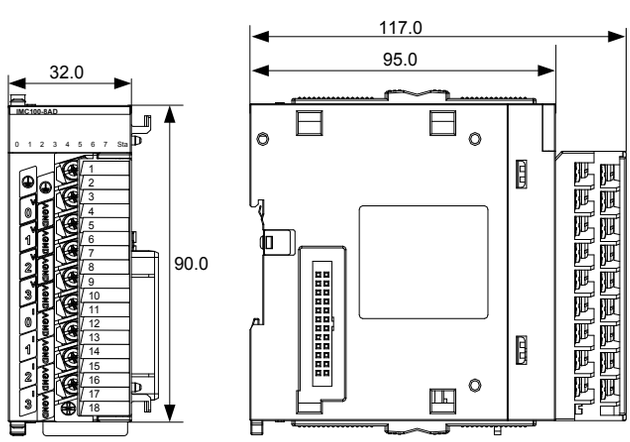
接外部光耦



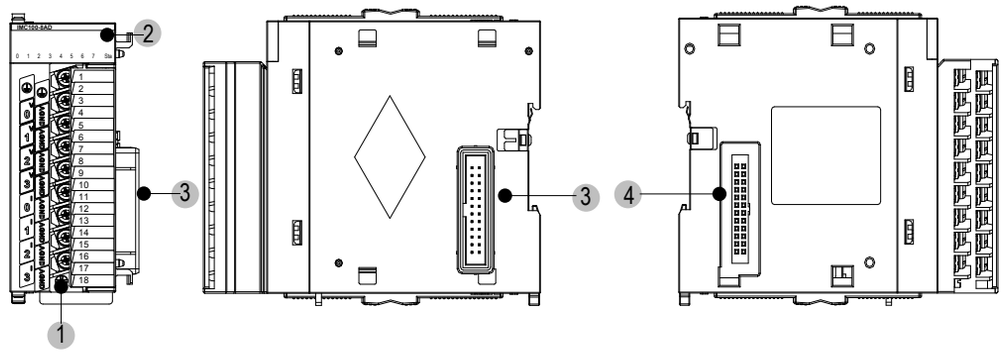
5.10.5 AD 扩展模块（型号： IMC100-8AD）



■ 安装尺寸



■ 端口定义

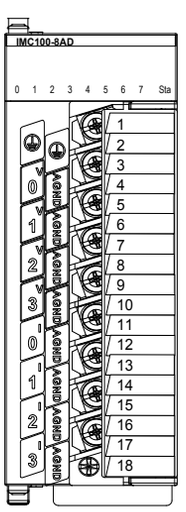


编号	名称	功能定义
1	用户端子	电压 / 电流接入端子
2	信号指示灯	STA:
3	本地扩展模块后级接口	连接后级模块
4	本地扩展模块前级接口	连接前级模块

■ 技术规格

项目	规格
输入通道	4 电流，4 通道电压
电压输入阻抗	>1MΩ
电流输入阻抗	250Ω
电压输入范围	双极性 ±5V，±10V，单极性 +5V，+10V
电流输入范围	0mA~20mA，4mA~20mA，0mA~24mA,
分辨率	电压：16 位
	电流：15 位
转换时间（最大）	10us
控制精度 (常温 25℃)	电压 ±0.1%，电流 ±0.1%
极限输入电压	±12V
极限输入电流	瞬间 30mA，平均 20mA
隔离方式	I/O 端子与电源之间：隔离； 通道之间：非隔离。

■ 端子信号排列

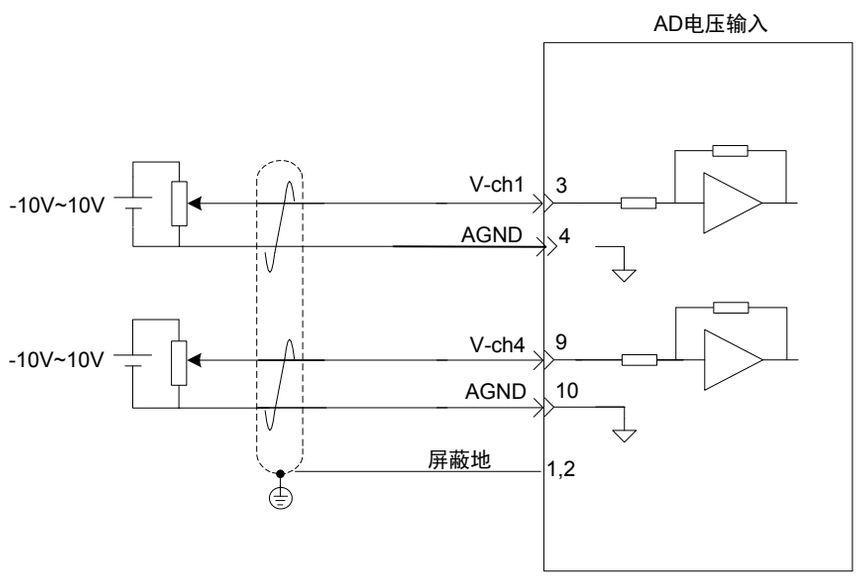


■ 端子定义

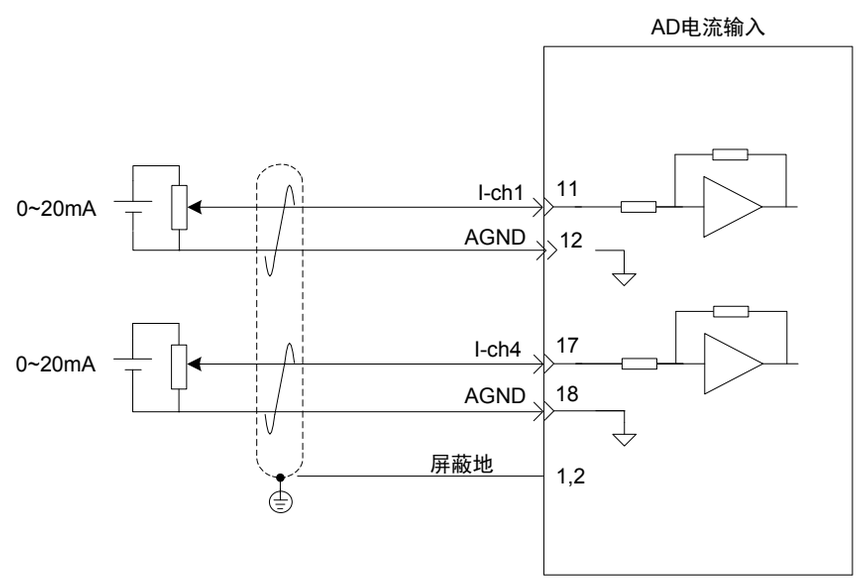
序号	网络名	类型	功能	备注
3	V1	输入	第 1 通道 V	电压输入
5	V2	输入	第 2 通道 V	电压输入
7	V3	输入	第 3 通道 V	电压输入
9	V4	输入	第 4 通道 V	电压输入
11	I1	输入	第 1 通道 I	电流输入
13	I2	输入	第 2 通道 I	电流输入
15	I3	输入	第 3 通道 I	电流输入
17	I4	输入	第 4 通道 I	电流输入
4,6,8,10, 12,14,16,18	AGND	输入	模拟输入地	
1~2	FE	机壳地	机壳地	

外部接线

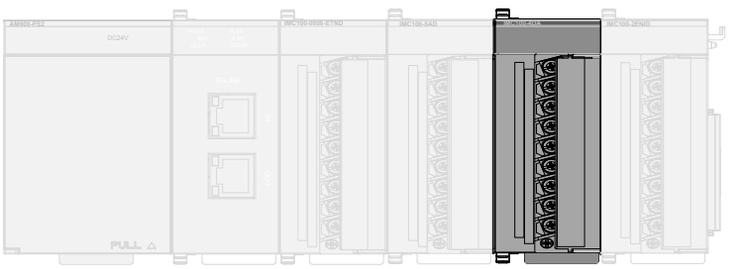
■ 电压输入：



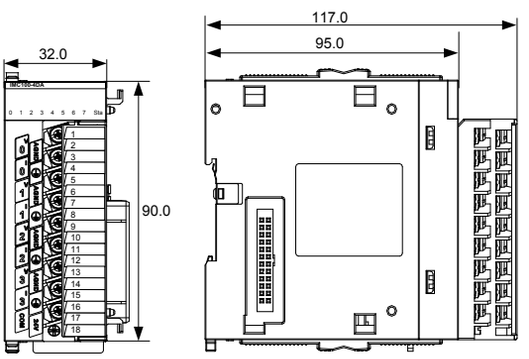
■ 电流输入：



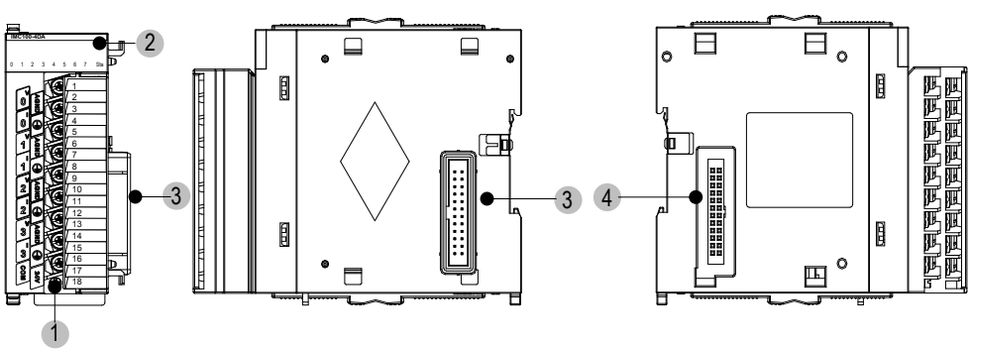
5.10.6 DA 扩展模块（型号： IMC100-4DA）



■ 安装尺寸



■ 端口定义

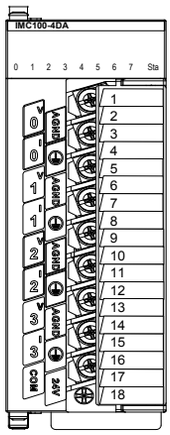


编号	名称	功能定义
1	用户端子	DA 转换输出接线端子
2	信号指示灯	STA:
3	本地扩展模块后级接口	连接后级模块
4	本地扩展模块前级接口	连接前级模块

■ 技术规格

项目	规格
输出通道	4（电压或电流方式）
输入电源电压	24 VDC（20.4 VDC~28.8 VDC）（-15%~+20%）
电压输出负载	1KΩ~1MΩ
电流出出负载	0Ω~600Ω
电压输出范围	双极性 ±5V，±10V，单极性 +5V，+10V
电流输出范围	4mA~20mA，0mA~20mA
控制精度 (常温 25℃)	电压 ±0.1%，电流 ±0.1%
分辨率	16 位
转换时间	125us/ 每通道 ,(最高 8K 的控制周期)
隔离方式	控制和转换 IC：隔离； 通道之间：非隔离。

■ 端子信号排列

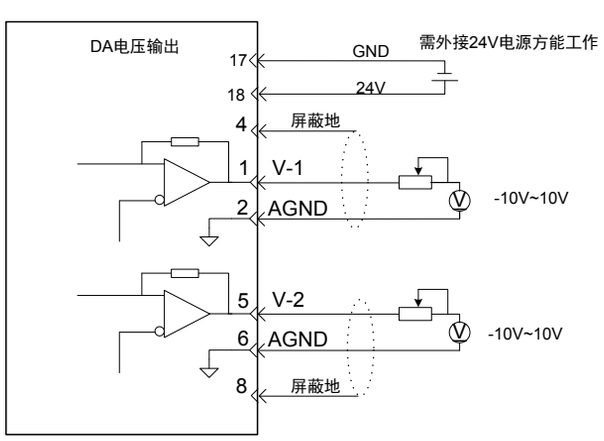
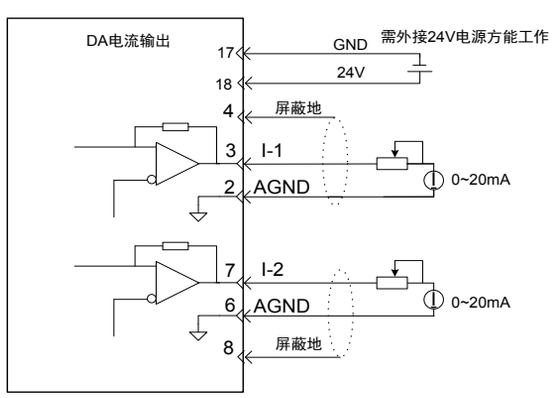


■ 端子定义

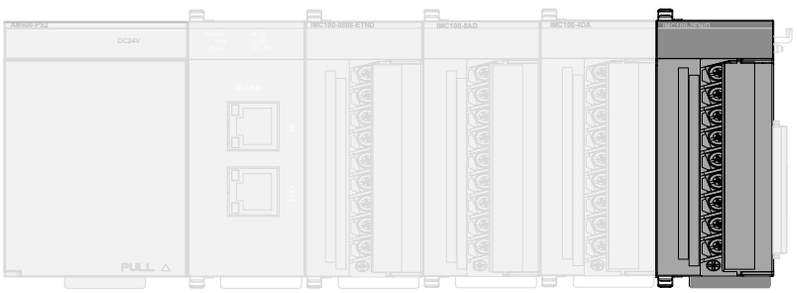
序号	网络名	类型	功能	备注
1	V1	输出	第 1 通道 V+	电压输出
3	I1	输出	第 1 通道 I+	电流输出
5	V2	输出	第 2 通道 V+	电压输出
7	I2	输出	第 2 通道 I+	电流输出
9	V3	输出	第 3 通道 V+	电压输出
11	I3	输出	第 3 通道 I+	电流输出
13	V4	输出	第 4 通道 V+	电压输出
15	I4	输出	第 4 通道 I+	电流输出
2,6,10,14	AGND			输出模拟地
4,8,12,16	FG			机壳屏蔽地
17	COM	电源地		外接电源地
18	24V	电源		外接电源

■ 外部接线

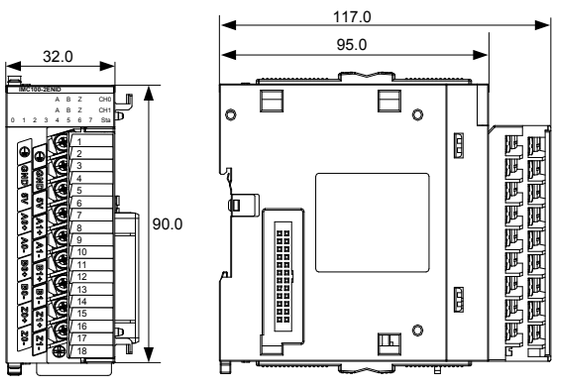
DA 的设置时为 4 个电流输出或者 4 个电压输出，不能够同时有效。

DA 电压输出 -- 外部接线	DA 电流输出 -- 外部接线
	

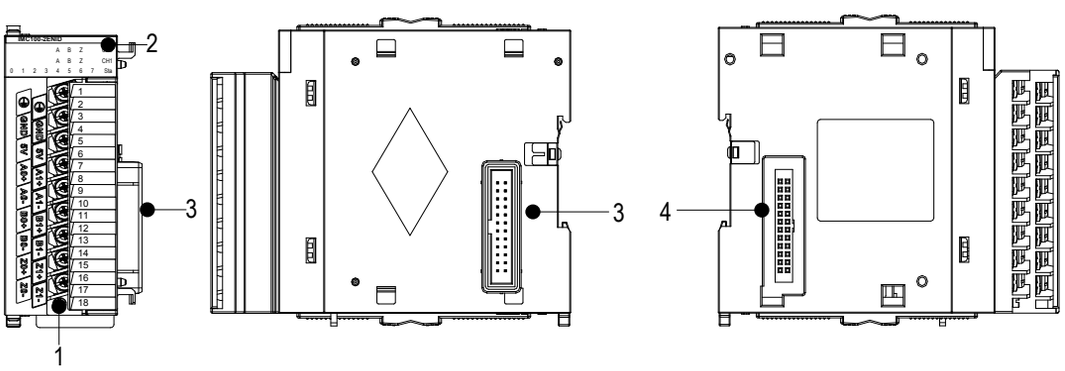
5.10.7 编码器模块（型号： IMC100-2ENID）



■ 安装尺寸



■ 外部接口

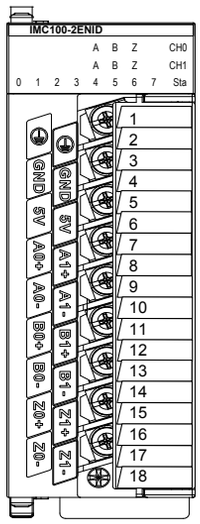


编号	接口名称	功能定义
1	用户端子	2 通道增量输入编码器端子
2	信号指示灯	CH0: A/B/Z CH1: A/B/Z STA:
3	本地扩展模块后级接口	连接后级模块
4	本地扩展模块前级接口	连接前级模块

■ 技术规格

项目	规格
输入通道	2
输入连接方式	差分 / 单端 输入，4 倍频计数方式
输入电压范围 (DC)	3.3V~5.5V
输入最大信号频率	1Mbps
输入阻抗	>750R
输入信号	A/B/Z 三相输入
隔离方式	光耦隔离
输入动作显示	光耦驱动时，相应输入指示灯动作

■ 端子信号排列



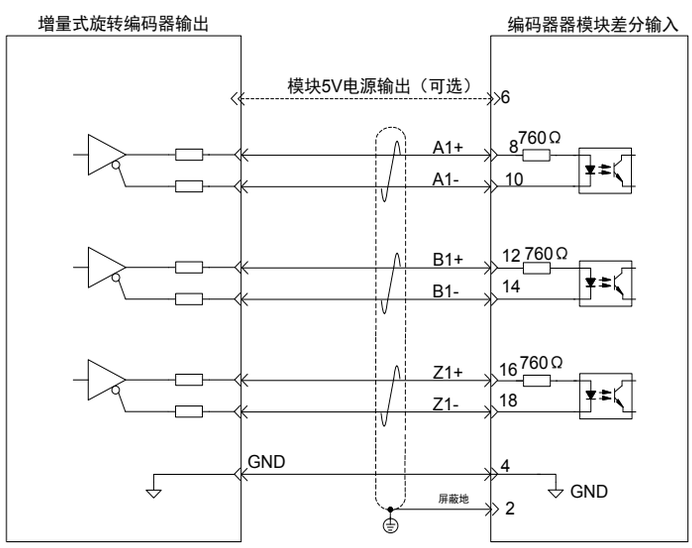
■ 端子定义

序号	网络名	类型	功能	备注
1\2	PE		PE 接地点	
3\4	GND	电源输出 (+-10%)		5V 电源输出
5\6	5V			
7	ENC0_PA+	最高输入频率：1Mbps	差分输入 A 相	A\B 相差 90 度
9	ENC0_PA-			
11	ENC0_PB+		差分输入 B 相	
13	ENC0_PB-			
15	ENC0_PZ+		差分输入 Z 相	Z 相信号输入
17	ENC0_PZ-			
8	ENC1_PA+		差分输入 A 相	A\B 相差 90 度
10	ENC1_PA-			
12	ENC1_PB+		差分输入 B 相	
14	ENC1_PB-			
16	ENC1_PZ+		差分输入 Z 相	Z 相信号输入
18	ENC1_PZ-			

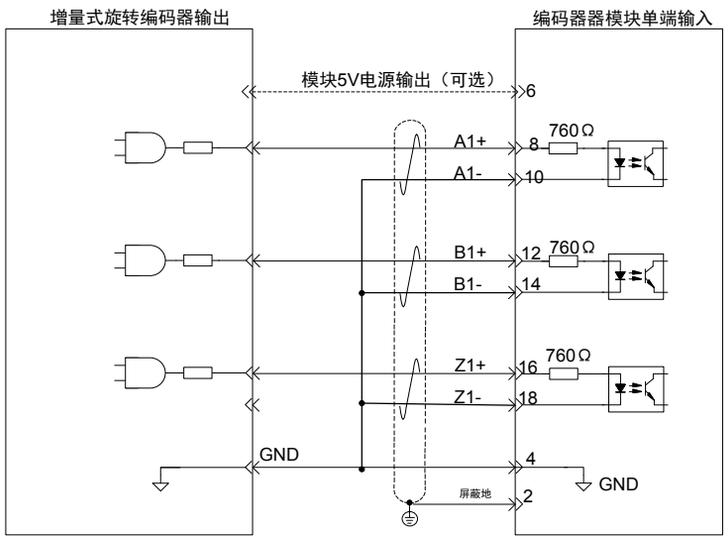
■ 外部接线

对于需要 5V 供电的编码器设备模块可提供 5V 电源，无需要则不接，但是 GND 线仍需要连接。

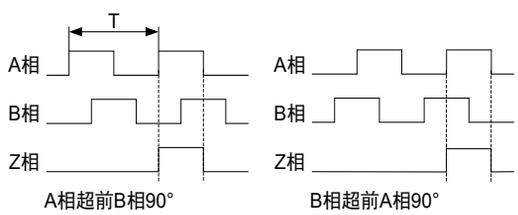
差分输入：



单端输入：

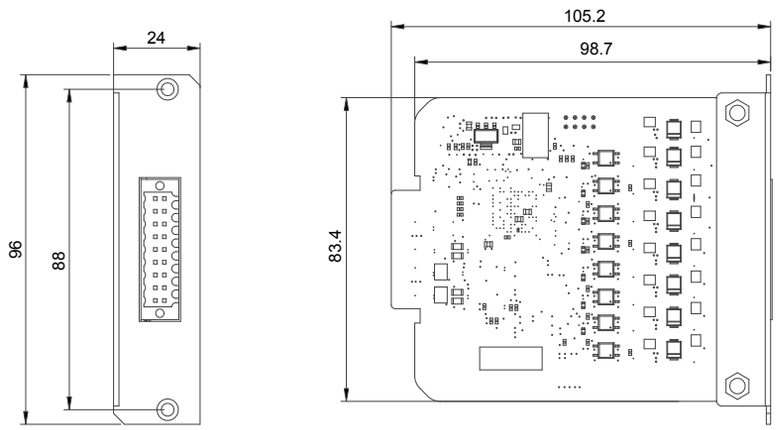

增量式旋转编码器输出
编码器器模块单端输入
模块5V电源输出（可选）
6
A1+
8
760Ω
A1-
10
B1+
12
760Ω
B1-
14
Z1+
16
760Ω
Z1-
18
GND
4
GND
屏蔽地
2

输入波形

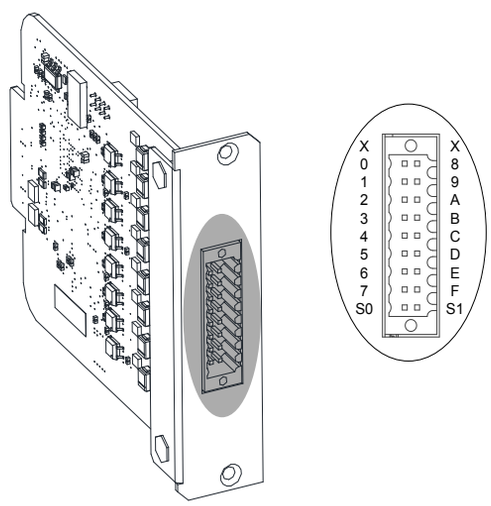

T
A相
B相
Z相
A相超前B相90°
A相
B相
Z相
B相超前A相90°

5.10.8 16 通道输入 IO 扩展卡（型号：IRCB-1600END-BD）

■ 安装尺寸



■ 端口定义



名称	功能
X0	源 / 漏型输入
X1	源 / 漏型输入
X2	源 / 漏型输入
X3	源 / 漏型输入
X4	源 / 漏型输入
X5	源 / 漏型输入
X6	源 / 漏型输入
X7	源 / 漏型输入
X8	源 / 漏型输入
X9	源 / 漏型输入
X10	源 / 漏型输入
X11	源 / 漏型输入
X12	源 / 漏型输入
X13	源 / 漏型输入
X12	源 / 漏型输入
X15	源 / 漏型输入
S0	公共端，0V 或 24V 有效
S1	公共端，0V 或 24V 有效

■ 输入端口规格

项目	规格
输入通道	16
输入连接方式	压接式接线端子
输入类型	数字量输入
输入方式	源 / 漏型
最大输入电压	30VDC
输入电流 (典型 24V)	4mA
ON 电压 (V)	>10VDC
OFF 电压 (V)	<5VDC
最大输入信号频率 (HZ)	1kHz
输入阻抗	>6.6K
隔离方式	光耦隔离

■ 外部接线

注意：

（1）扩展卡安装前撕下其金手指的保护贴胶；

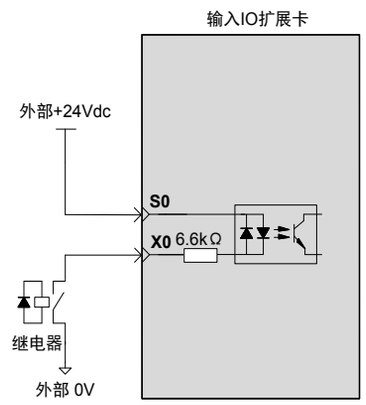
（2）S0，X0~X7 为一组；S1，X8~X15 为一组

（2）S0/S1 端为公共端，当 S0/S1 接 24V 时，X* 端输入 0V 则输入信号有效（板内光耦导通）；当 S0/S1 接 0V 时，X* 端输入 24V 时则输入信号有效（板内光耦导通），由此可接 NPN 或 PNP 型输出的 IO 设备。

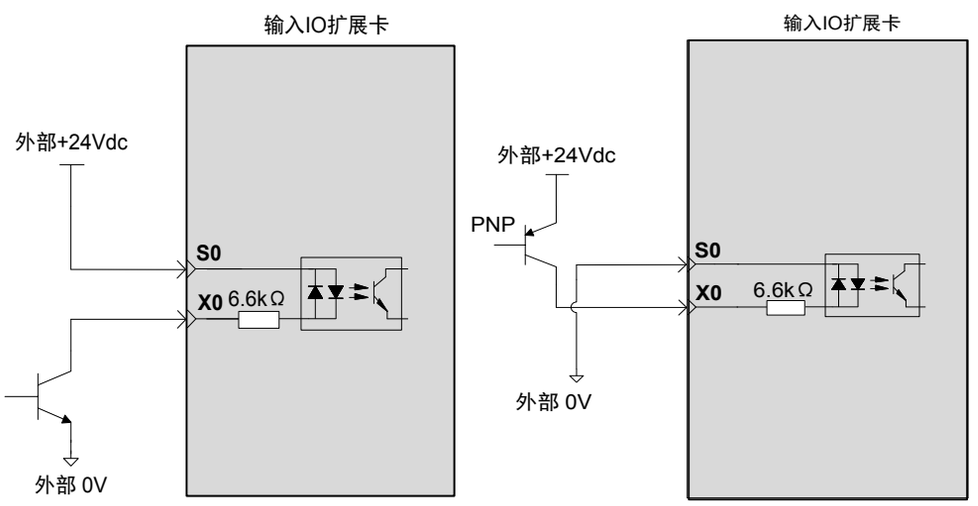
■ 接线示意

以 X0 为例（X1~X7 同 X0）

1） 当上位装置为继电器输出时：

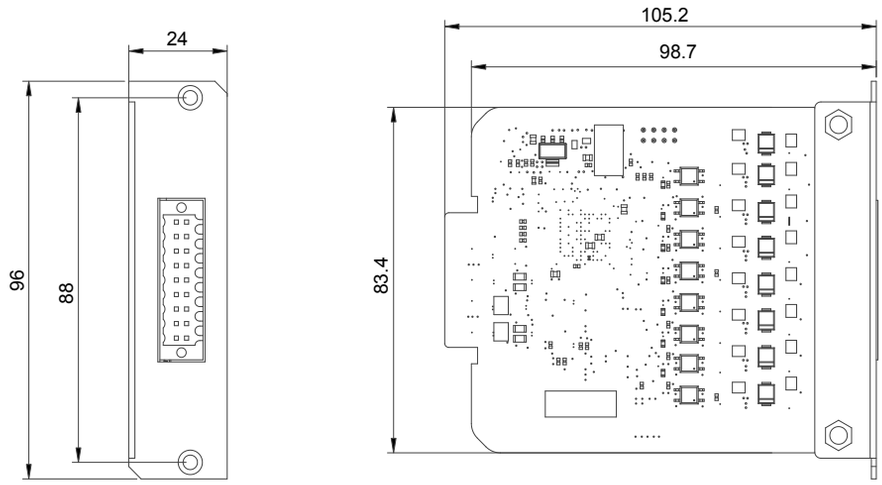


2） 当上位装置为集电极开路输出时：

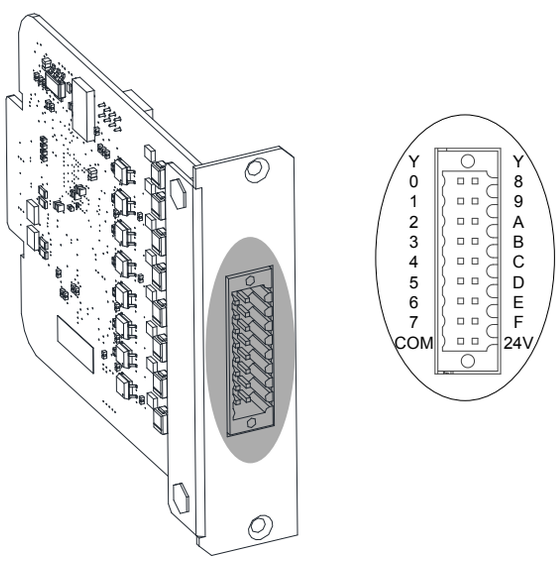


5.10.9 16 通道 PNP 输出 IO 扩展卡（型号：IRCB-0016ETPD-BD）

■ 安装尺寸



■ 端口定义

	名称	备注
	Y0	源型（PNP）输出
	Y1	源型（PNP）输出
	Y2	源型（PNP）输出
	Y3	源型（PNP）输出
	Y4	源型（PNP）输出
	Y5	源型（PNP）输出
	Y6	源型（PNP）输出
	Y7	源型（PNP）输出
	Y8	源型（PNP）输出
	Y9	源型（PNP）输出
	YA	源型（PNP）输出
	YB	源型（PNP）输出
	YC	源型（PNP）输出
	YD	源型（PNP）输出
	YE	源型（PNP）输出
	YF	源型（PNP）输出
	COM	电源地，接入外部电源
	24V	24V 电源，接入 24V±20% 电源

■ 输出端口规格

项目	规格
输出通道	16
输出连接方式	压接线端子
输出类型 (接入方式)	源型（PNP）
OFF 时最大漏电流	<10uμA
最大输出信号频率	1kHz
单点最大负载电流	500mA
隔离方式	光耦隔离
保护功能	过载、过温、短路保护
最大输入信号频率	1kHz

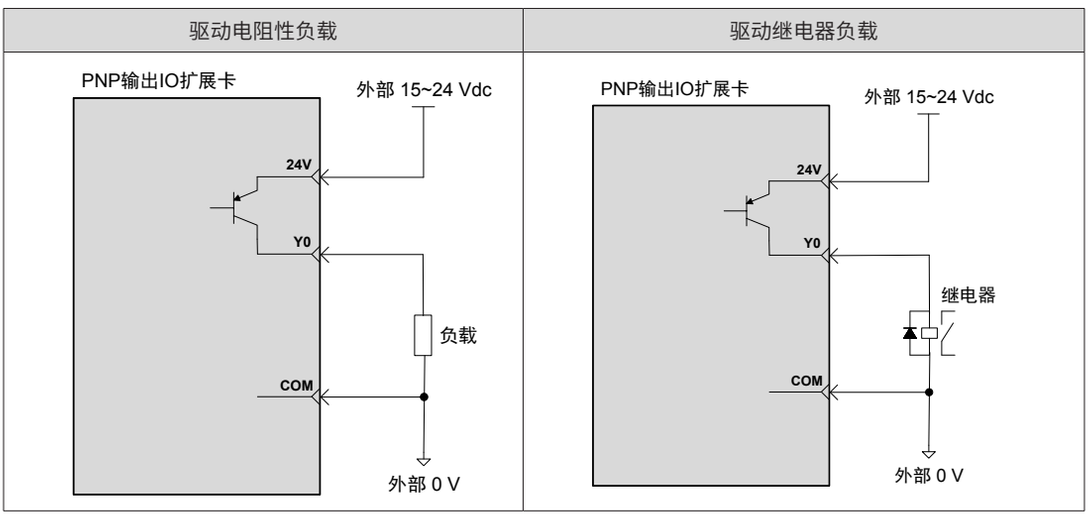
■ 外部接线

注意：

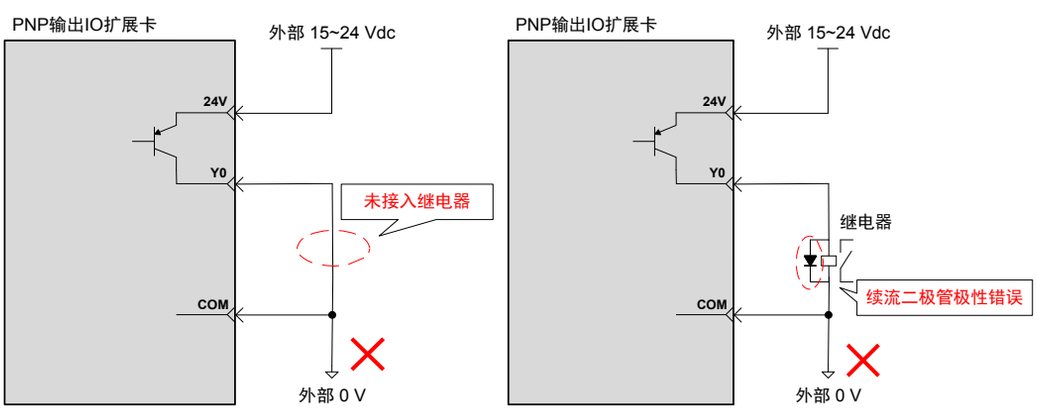
扩展卡安装前撕下其金手指的保护贴胶；

为保证输出端口正常工作，24/COM 必须接入 24V 电源；

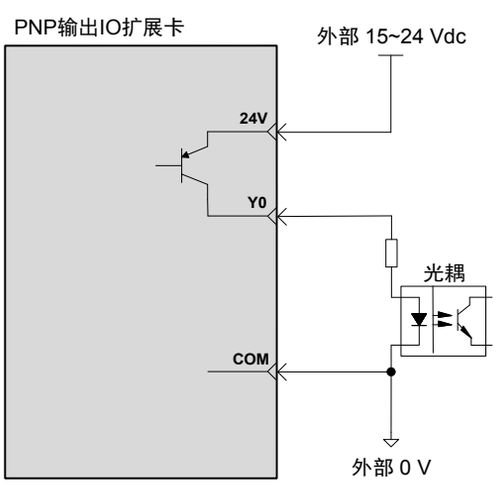
单个 IO 输出最大电流为 0.5A,16 个 IO 口持续输出总电流不大于 8A。



■ 错误接法如下图：

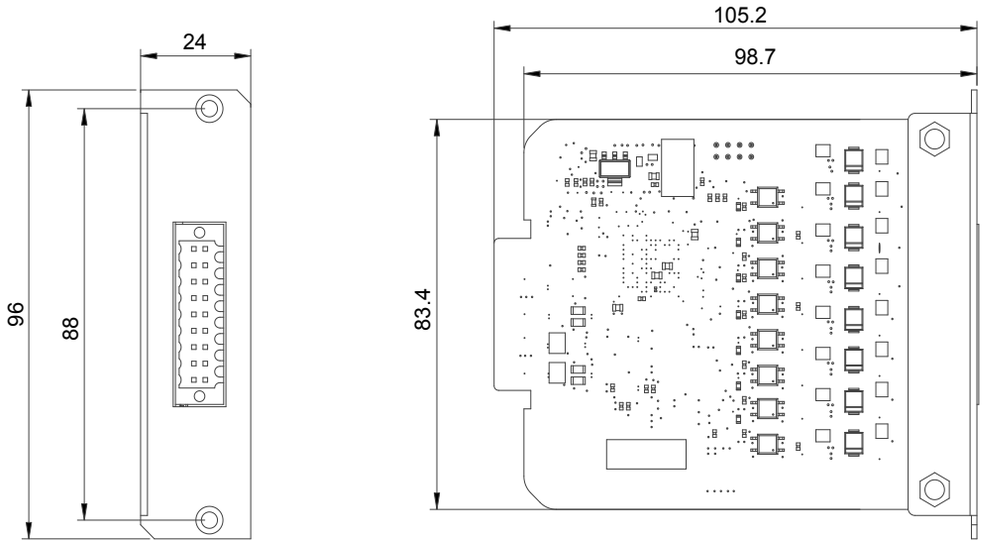


■ 驱动光耦负载：

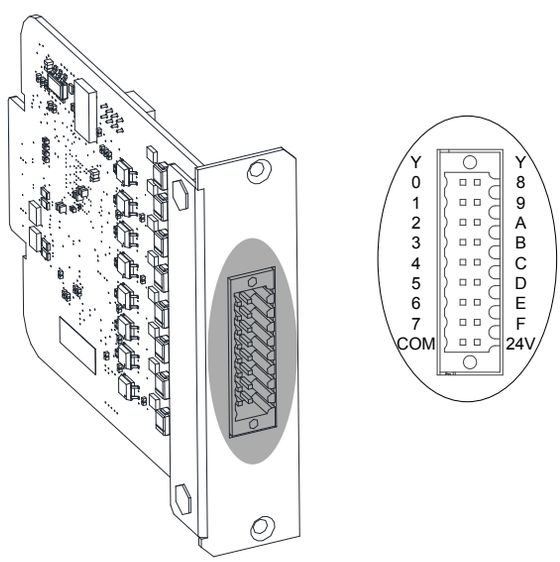


5.10.10 3 16 通道 NPN 输出 IO 扩展卡（型号：IRCB-0016ETND-BD）

■ 安装尺寸



■ 端口定义



名称	备注
Y0	漏型（NPN）输出
Y1	漏型（NPN）输出
Y2	漏型（NPN）输出
Y3	漏型（NPN）输出
Y4	漏型（NPN）输出
Y5	漏型（NPN）输出
Y6	漏型（NPN）输出
Y7	漏型（NPN）输出
Y8	漏型（NPN）输出
Y9	漏型（NPN）输出
YA	漏型（NPN）输出
YB	漏型（NPN）输出
YC	漏型（NPN）输出
YD	漏型（NPN）输出
YE	漏型（NPN）输出
YF	漏型（NPN）输出
COM	电源地，接入外部电源
24V	24V 电源，接入 24V±20% 电源

■ 输出端口规格

项目	规格
输出通道	16
输出连接方式	压接线端子
输出类型 (接入方式)	漏型（NPN）
OFF 时最大漏电流	<10uμA
最大输出信号频率	1kHz
单点最大负载电流	500mA
隔离方式	光耦隔离
保护功能	过载、过温、短路保护
最大输入信号频率	1kHz

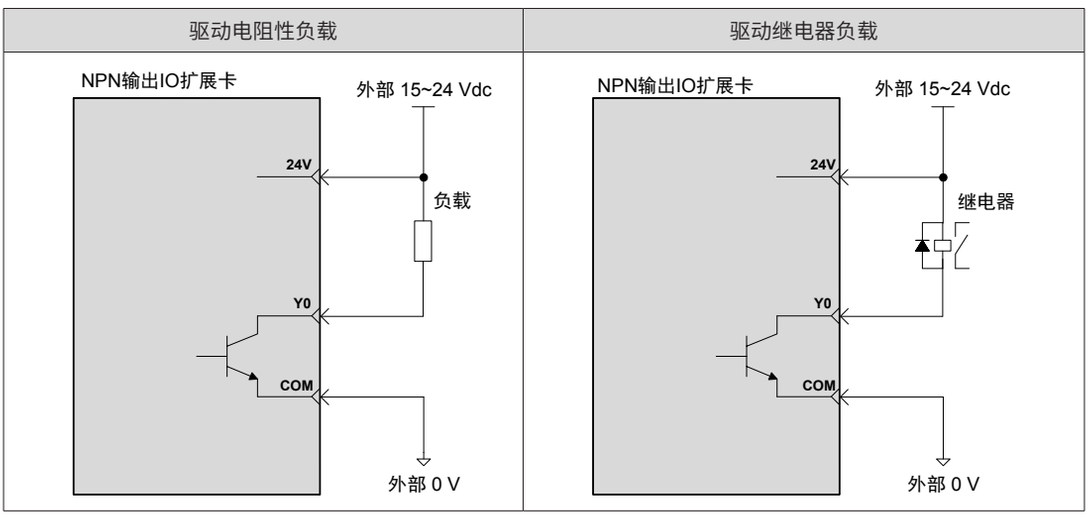
■ 外部接线

注意：

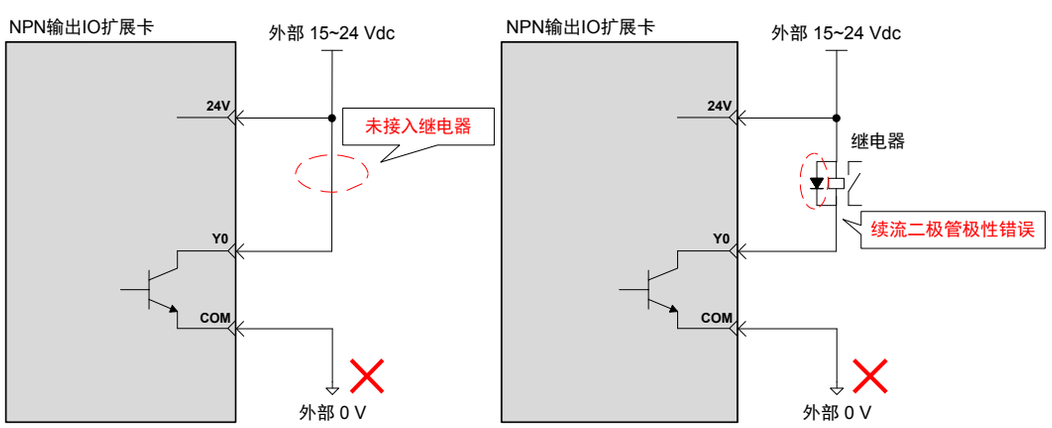
扩展卡安装前撕下其金手指的保护贴胶；

为保证输出端口要正常工作，24/COM 必须接入 24V 电源；

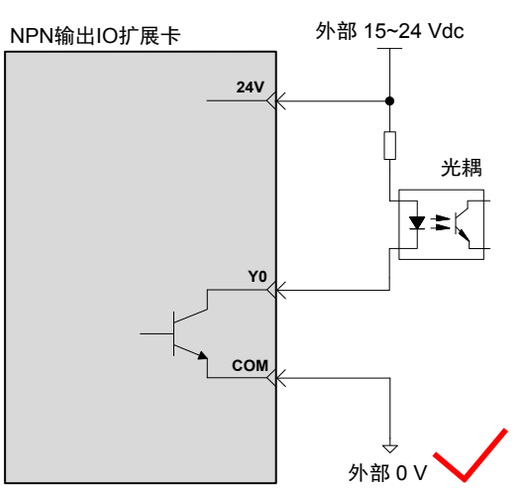
单个 IO 输出最大电流为 0.5A,16 个 IO 口持续输出总电流不大于 8A。



■ 错误接法如下图：

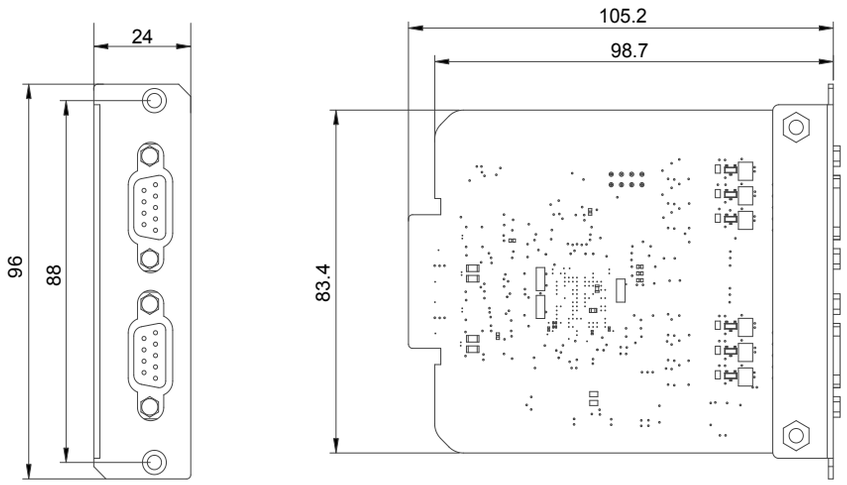


■ 驱动光耦负载：

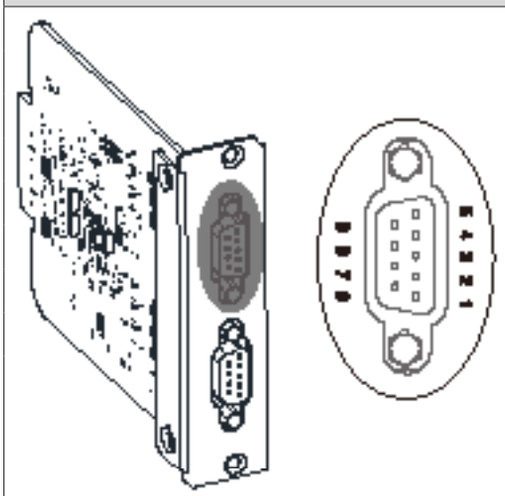


5.10.11 2 通道增量型编码器扩展卡（型号：IRCB-2EN1D-BD）

■ 安装尺寸



■ 端口定义 -DB9 信号定义

	序号	名称	说明
	1	未定义	
	2	GND	对外可提供 5V 电源接入，对外输出总电流为 1A.
	6	5V	
	5	ENC_PA+	差分输入 A 相
	9	ENC_PA-	
	4	ENC_PB+	差分输入 B 相
	8	ENC_PB-	
	3	ENC_PZ+	Index 单次捕获
	7	ENC_PZ-	

■ 端口规格

项目	规格
输入通道	2
输入连接方式	差分 / 单端 输入，4 倍频计数方式
输入电压范围 (DC)	3.3V~5.5V
输入最大信号频率	500Kbps
输入阻抗	>760R
输入信号	A/B/Z 三相输入
隔离方式	光耦隔离

■ 外部接线

注意：

扩展卡安装前撕下其金手指的保护贴胶；

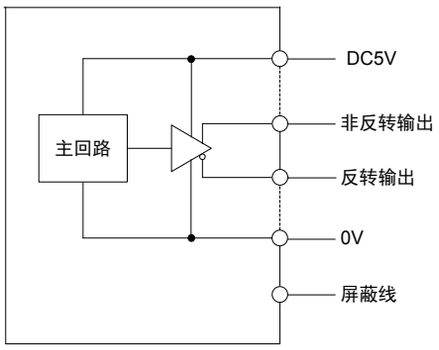
扩展模块电缆布线时，避免与动力线（高电压，大电流）等传输强干扰信号的电缆捆在一起，应该分开走线并且避免平行走线。

选用推荐线缆，建议选用屏蔽双绞线缆提高抗干扰能力。

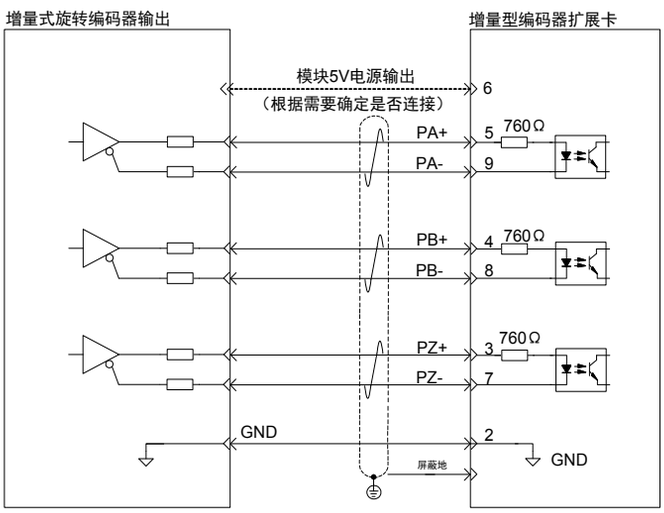
只支持 5V 电平的编码器输入信号，其余规格电平输入不支持。

1） 差分输入时（推荐使用差分输入方式）

编码器示例如下图：

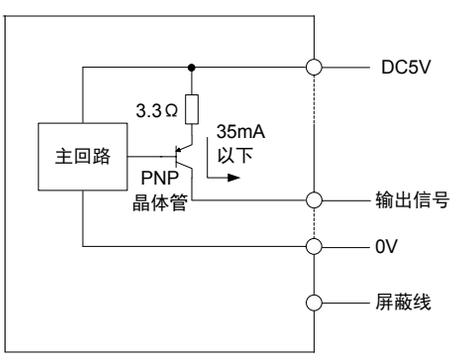


■ 接线方式：

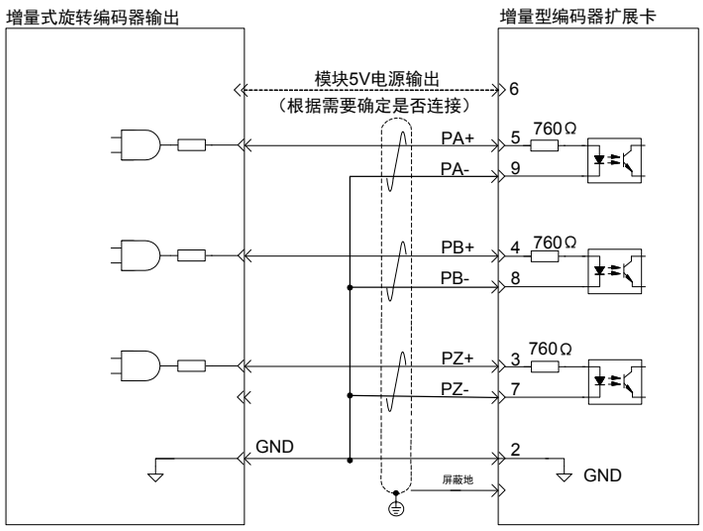


2） 单端 PNP 型 5V 输入

编码器示例如下图：

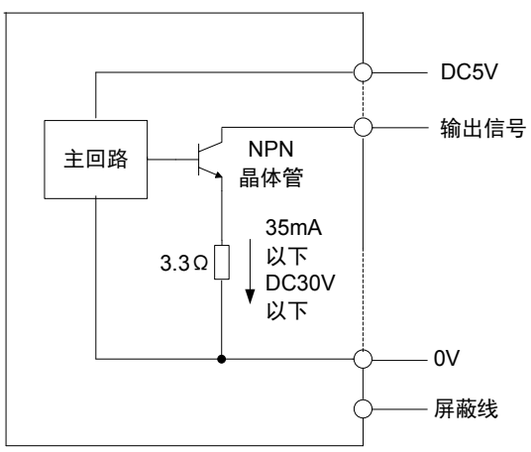


■ 接线方式：

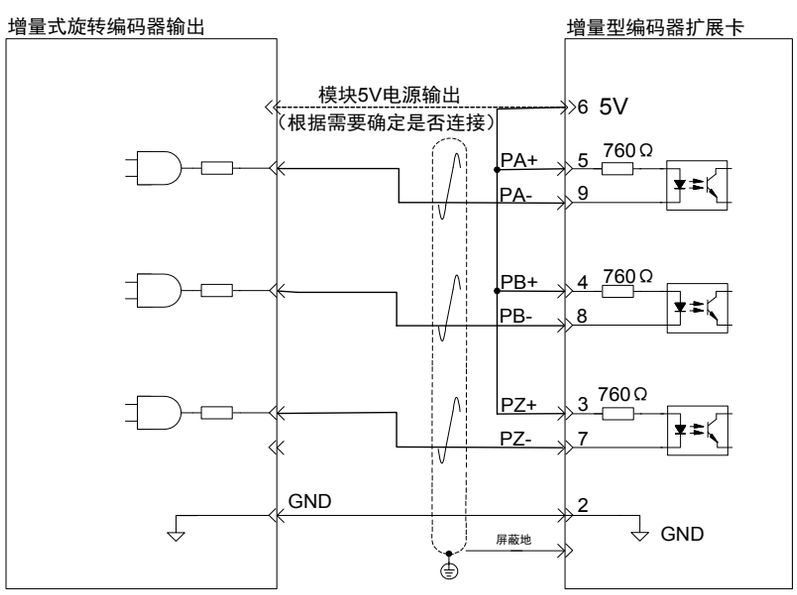


3） 单端 NPN 型 5V 输入

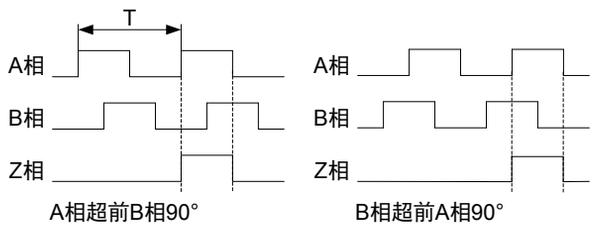
编码器示例如下图：



■ 接线方式：

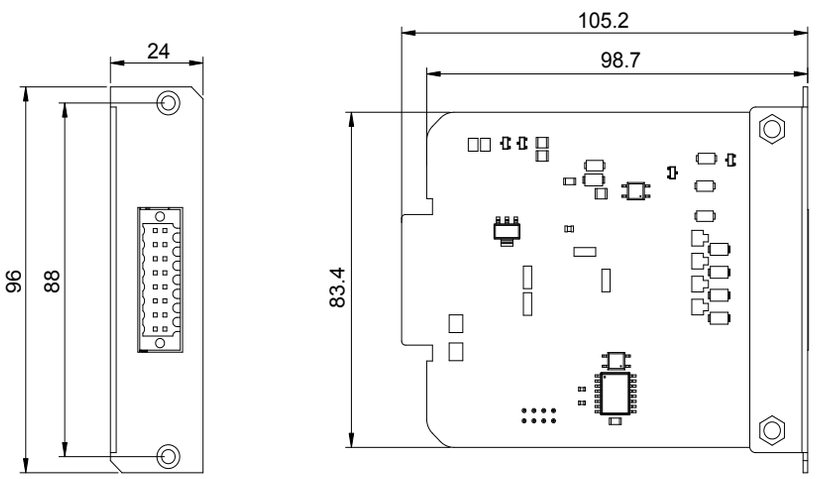


■ 单端输入波形

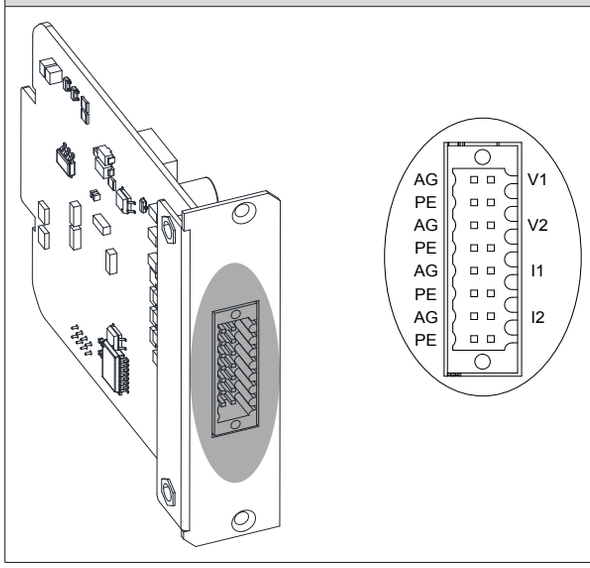


5.10.12 2 通道电压和 2 通道电流采样扩展卡（型号：IRCB-4AD-BD）

■ 安装尺寸



■ 端口定义 -DB9 信号定义



引脚	功能
AG	模拟输入地
PE	机壳地
V1	电压信号输入 1
V2	电压信号输入 2
I1	电流信号输入 1
I2	电流信号输入 2

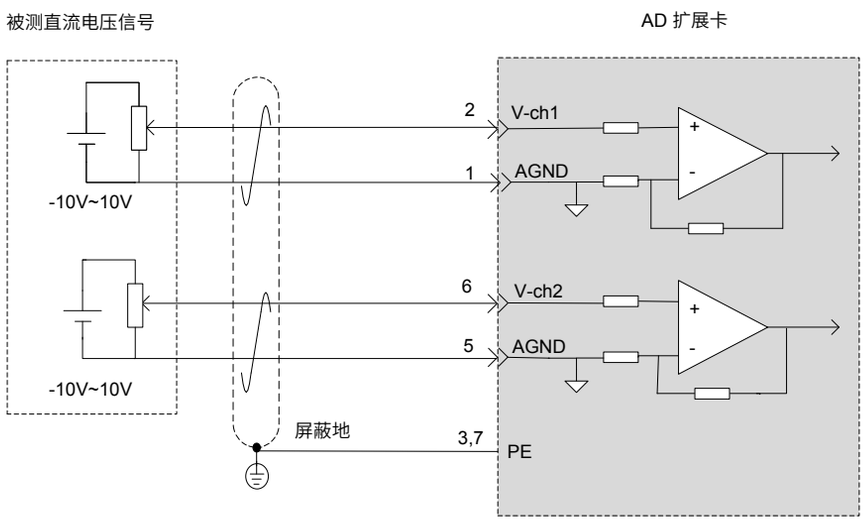
■ 端口规格

项目	规格
输入通道	2 通道电流，2 通道电压
输入连接方式	压接式接线端子
电压输入范围	-5V ～ 5V，-10V ～ 10V（直流信号）
电流输入范围	0mA ～ 20mA（直流信号）

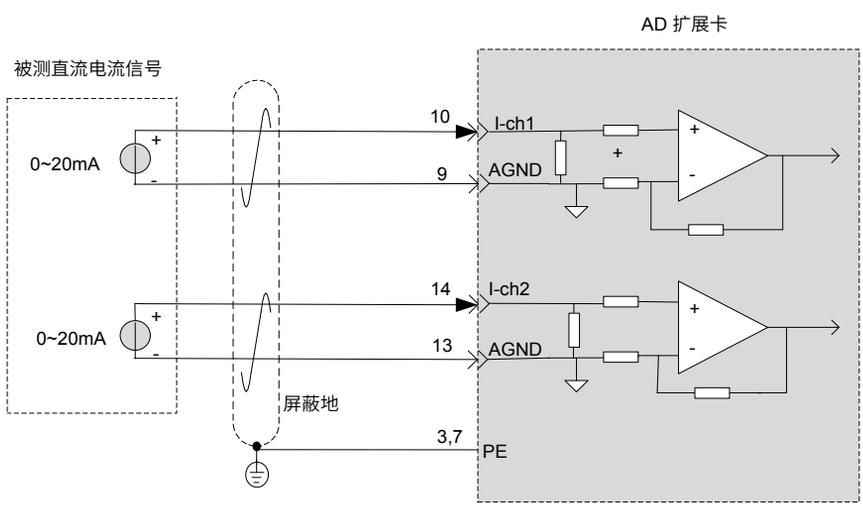
项目	规格
精度（常温 25℃）	电压 ±0.1%（全量程），电流 ±0.15%（全量程）
精度（环境 0-50℃）	电压 ±0.3%（全量程），电流 ±0.8%（全量程）
电压通道输入阻抗	1MΩ
电流通道输入阻抗	250Ω
最大输入电压	±12V
最大输入电流	40mA
最快转换时间	10μs
隔离方式	I/O 端子与电源之间：隔离； 通道之间：非隔离（共模拟地）

■ 外部接线

电压输入接线方法

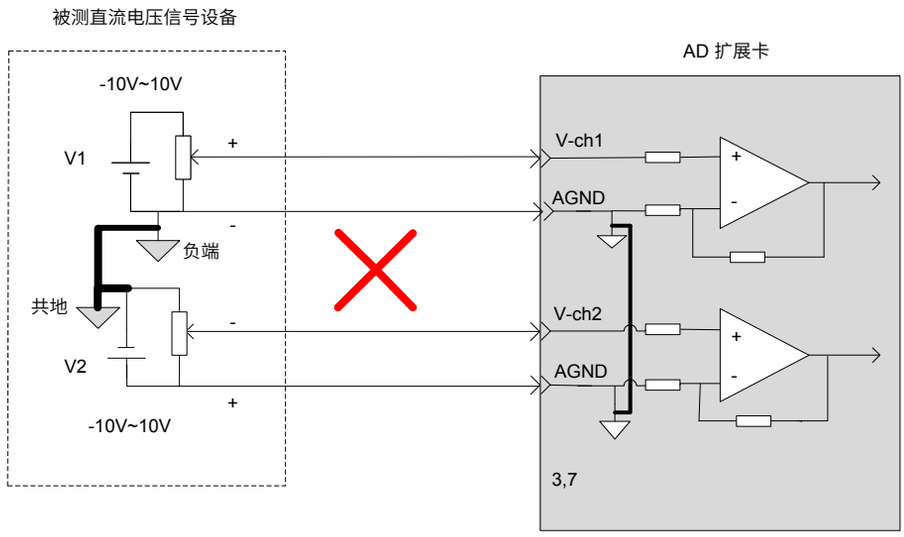


电流输入接线方法



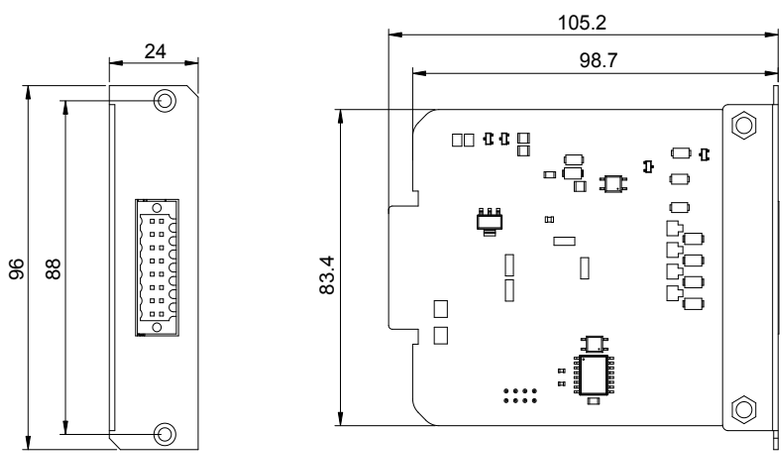
注意：由于通道间没隔离 (共模拟地)，当同时接入多个测量信号时，如果外部输入源的负端共地，应注意防止因反接引起输入源短路。

如下图所示，被测电压信号设备 V1 和 V2 的负端连载在一起，如果在测量时，V2 的负端连接 AD 扩展卡的 V-ch2，V2 的正端连接 AD 扩展卡的 AGND，则被测电压设备 V2 输出端会通过扩展卡的内部 AGND 短接在一起。

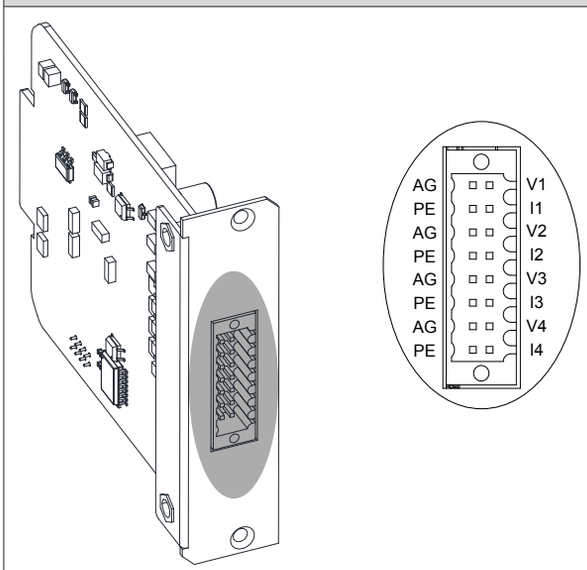


5.10.13 4 通道电压或电流输出扩展卡（型号：IRCB-4DA-BD）

■ 安装尺寸



■ 端口定义



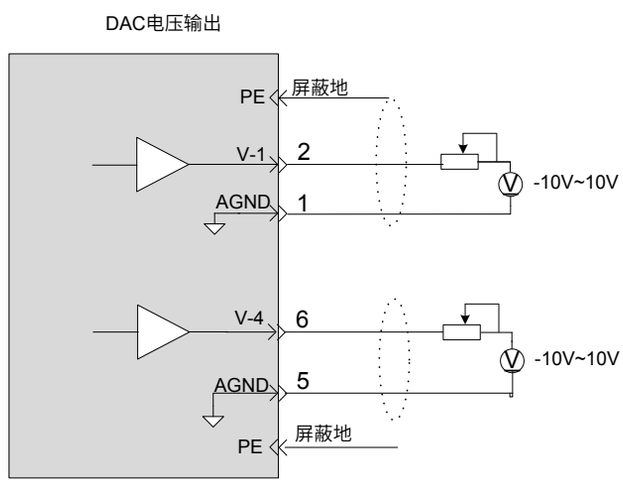
	部件名称	功能说明	备注
	V1	第 1 通道 V+	电压输出
	I1	第 1 通道 I+	电流输出
	V2	第 2 通道 V+	电压输出
	I2	第 2 通道 I+	电流输出
	V3	第 3 通道 V+	电压输出
	I3	第 3 通道 I+	电流输出
	V4	第 4 通道 V+	电压输出
	I4	第 4 通道 I+	电流输出
	AG	AGND	输出模拟地
	PE	机壳地	机壳地

■ 端口规格

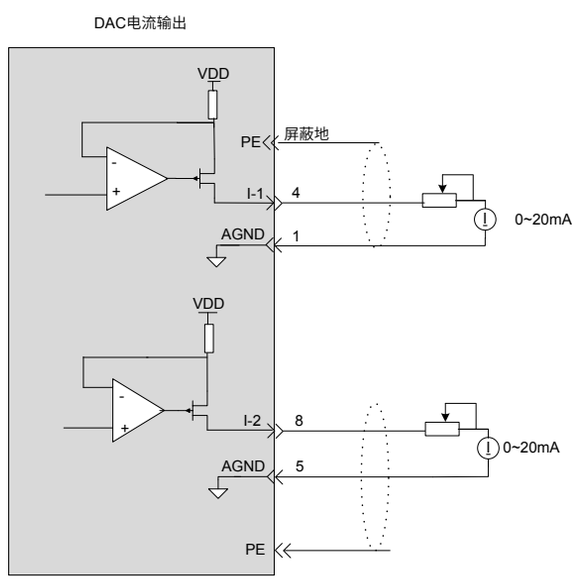
项目	规格
输出通道	4（电压或电流方式）
输出连接方式	压接式接线端子
输出信号类型	直流信号
电压输出范围	双极性 ±5V，±10V 单极性 0 ～ 5V，0 ～ 10V
电流输出范围	0mA ～ 20mA
精度（常温 25℃）	电压 ±0.1%（全量程）；电流 ±0.1%（全量程）
精度（环境温度 0-50℃）	电压 ±0.3%（全量程）；电流 ± 0.3%（全量程）
电压通道输出阻抗	0.06Ω
电流通道输出阻抗	100MΩ
电流通道最大允许阻性负载	＜ 600 Ω
转换时间	125μs/ 每通道（最高 8K 的采样周期）
最快转换时间	10μs
隔离方式	控制和转换 IC：隔离； 通道之间：非隔离（共输出模拟地）

外部接线

■ DA 电压输出接线方法



■ DA 电流输出接线方法



第 6 章 汇川机器人示教器

6.1 产品信息

6.1.1 型号说明

■ IRTP80

IRTP80-L5Z		
IRTP80 系列示教器	线缆长度	L5Z：5 米线缆

■ ITP100

ITP100-A		
ITP100 系列示教器	摇杆功能	A： 带摇杆 B：不带摇杆

6.1.2 产品外观

■ IRTP80

图 6-1 IRTP80 示教器外观示意图

■ ITP100

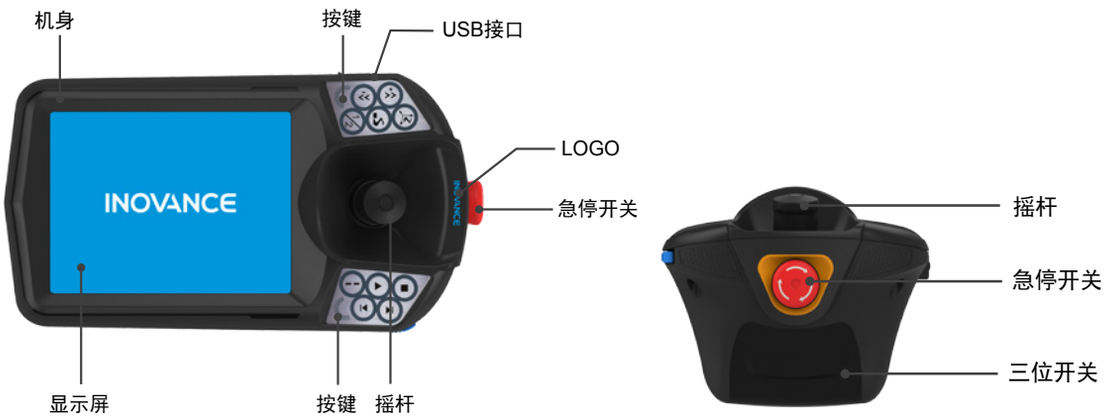


图 6-2 ITP100 示教器外观示意图

6.1.3 规格参数

型号	IRTP80	IRTP100
屏幕	7 寸 TFT-LCD，触屏操作、功能按键；IP54 密封	8 寸 TFT-LCD，触屏操作、功能按键；IP65 密封；标配摇杆
处理器	ARM A8 600M 处理器，1G bit RAM 1G bit NAND	
通讯	100M Ethernet、功能 IO、USB2.0 接口	
面板	模式选择开关，安全开关，急停按钮，移动键；	
电源	控制器额定电压：24V 输入电压范围：19V-30V	
运行温度	-10° ~45°，按照 IEC61131-2	
存储温度	-20° ~80°，按照 IEC61131-2	

6.2 各部件说明

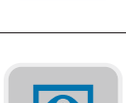
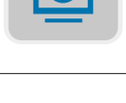
手轮 / 摇杆：机器人运动操作杆，可操纵机器人作相应的关节运动。

显示屏：机器人示教器与客户交互窗口，可进行编程、调试、监控等操作。

急停开关 : 机器人紧急停止按钮。

三位开关：包含三个位置，中间位置，机器人使能；弹起或者用力按下机器人均不使能。

■ IRTP80 按键功能表

按键	按键名称	按键功能
	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。
	模式切换	示教 / 再现切换，效果同普通示教器
	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。
	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、直角坐标系、工具坐标系、用户坐标系
	一键回零	示教模式下，伺服上使能后，所有轴回到绝对零点
	再现启动 / 示教检查连续运行	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。
	主页	主页，回到编程 / 运行页面，效果同普通示教器： 
	监控	监控，回到监控页面，效果同普通示教器： 
	设置	监控，回到监控页面，效果同普通示教器： 

■ ITP100 按键功能表

按键	按键名称	按键功能
	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(Ry/Rx/Rz) 轴或更多组之间切换。
	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、直角坐标系、工具坐标系、用户坐标系
	点动	示教运动方式，示教模式下，选择该模式，机器人使能后，按下轴方向运动按键，机器人在该方向上运动指定偏移量
	再现启动 / 示教检查连续运行	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。

■ 指示灯说明

指示灯	
电 源	代表是否上电
运 行	代表是否处于再现模式
异 常	代表出现报警
S1 S2 S3 S4	客户定制

6.3 接线

6.3.1 与控制柜的连接

IRTP80、ITP100 系列示教器通过连接线缆、接线盒与控制柜进行连接，连接示意图如下图所示：

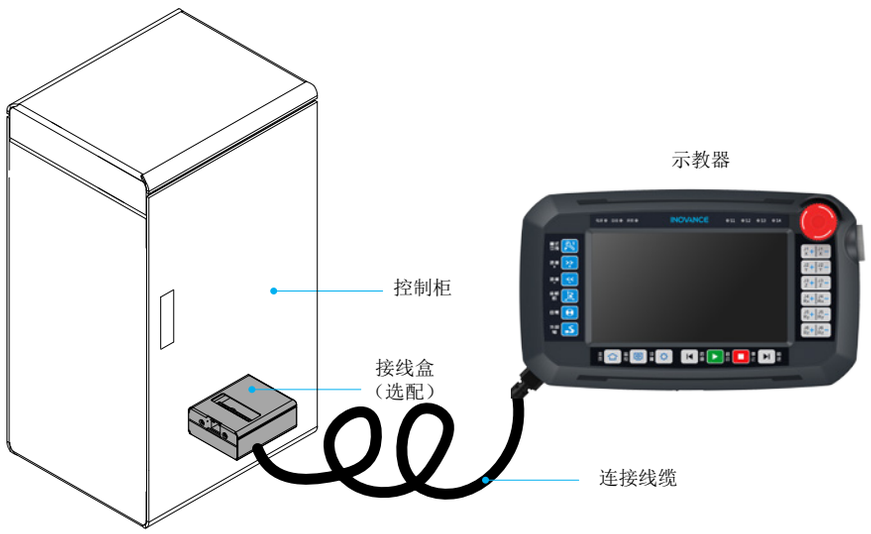


图 6-3 示教器与控制柜连接示意图

注：接线盒属于选配件，我司机器人控制柜内已配备，若使用其他控制柜需要另行购买。

6.3.2 连接线缆

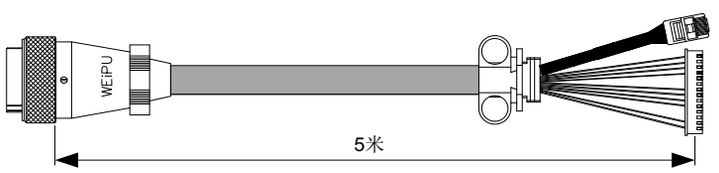
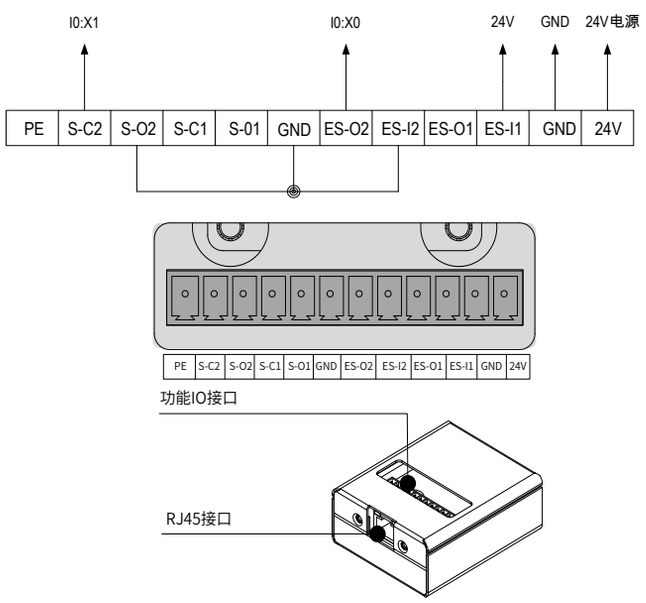


图 6-4 接线盒



6.3.3 功能 IO 接口定义

序号	信号	含义
1	PE	
2	S-C2	使能开关 2 负
3	S-O2	使能开关 2 正
4	S-C1	使能开关 1 负
5	S-O1	使能开关 1 正
6	GND	电源地
7	ES-O2	急停开关 2 负
8	ES-I2	急停开关 2 正
9	ES-O1	急停开关 1 负
10	ES-I1	急停开关 1 正
11	GND	电源地
12	24V	电源

6.3.4 J45 引脚定义

序号	信号	含义
1	TX+	发送正
2	TX-	发送负
3	RX+	接收正
4	NC	—
5	NC	—
6	RX-	接收负
7	NC	—

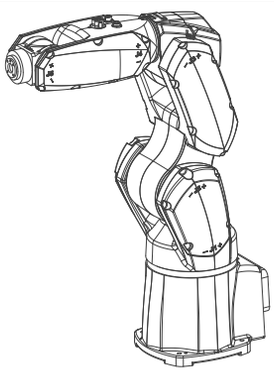
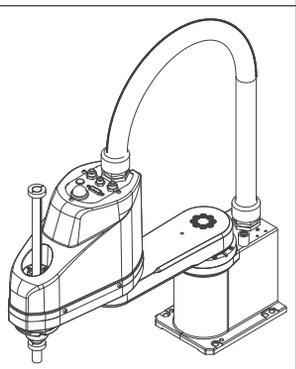
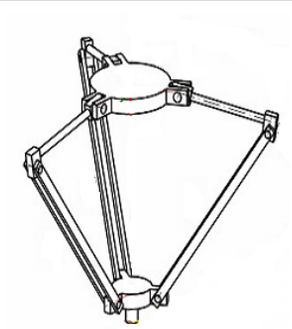
应用篇 1：编程设计与实现

应用篇 1：编程设计与实现 - 目录

第 1 章 基本术语

1.1 机型

根据轴数、串联 / 并联特性，对机器人分类如下：

名称	类型	轴数	串 / 并联	特点	应用场合
六轴机器人		6	串联	灵活性极高，几乎适合于任何轨迹或角度的工作。	装货、喷漆、测量、弧焊、点焊、包装、装配、锻造、铸造等。
SCARA 机器人		4	串联	结构轻便、响应快。	机械组装，物料配送、分发等场合，3C 行业的机器人装配，贴标，贴片，点胶等。
Delta 机器人		4	并联	精度高，速度快。	医药食品的搬运、分拣等。

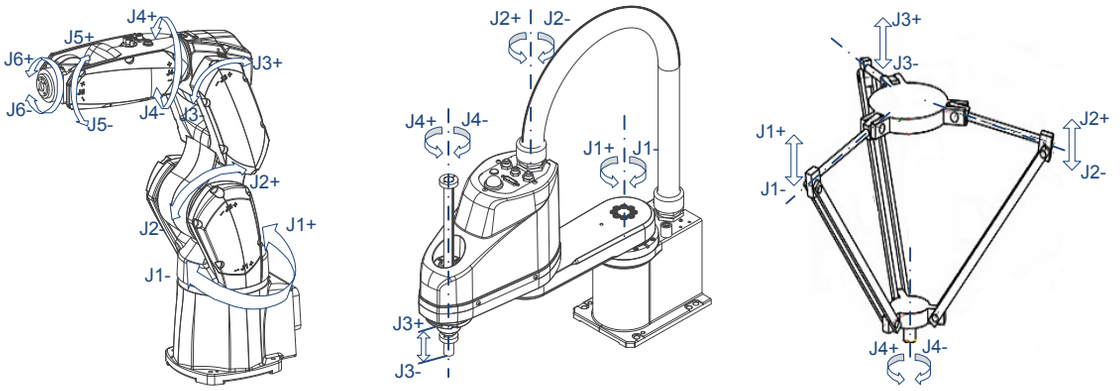
1.2 坐标系

我司机器人系统中，有 7 种坐标系，如下表所示：

坐标系号	坐标系名
1	关节坐标系
2	基坐标系
3	工具坐标系
4	用户坐标系
5	固定相机视野坐标系
6	移动相机视野坐标系
7	传送带上物体坐标系

1.2.1 关节坐标系

关节坐标系存在于机器人各关节处。

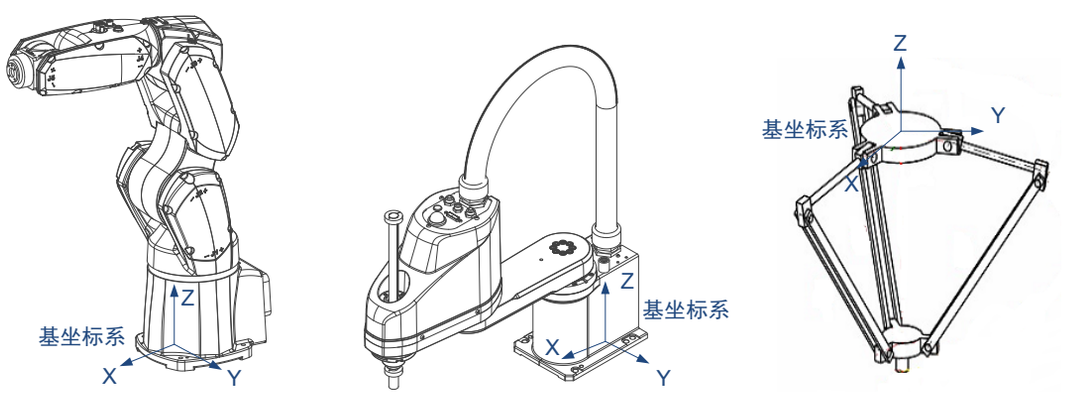


（a）六轴机器人　　（b）SCARA 机器人　　（c）Delta 机器人

图 1-5 机器人关节坐标系

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。

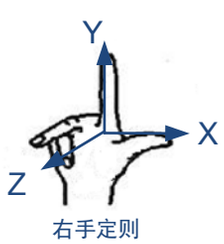


（a）六轴机器人　　（b）SCARA 机器人　　（c）Delta 机器人

图 1-6 机器人基坐标系

基坐标系是笛卡尔类型的坐标系（工具坐标系、用户坐标系也为笛卡尔类型坐标系）。

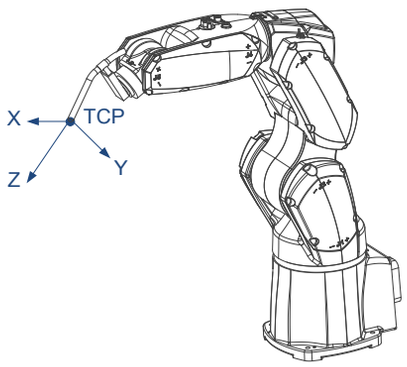
笛卡尔类型坐标系方向确定方法：先确定 X 和 Z 方向，再由“右手定则”确定 Y 方向。



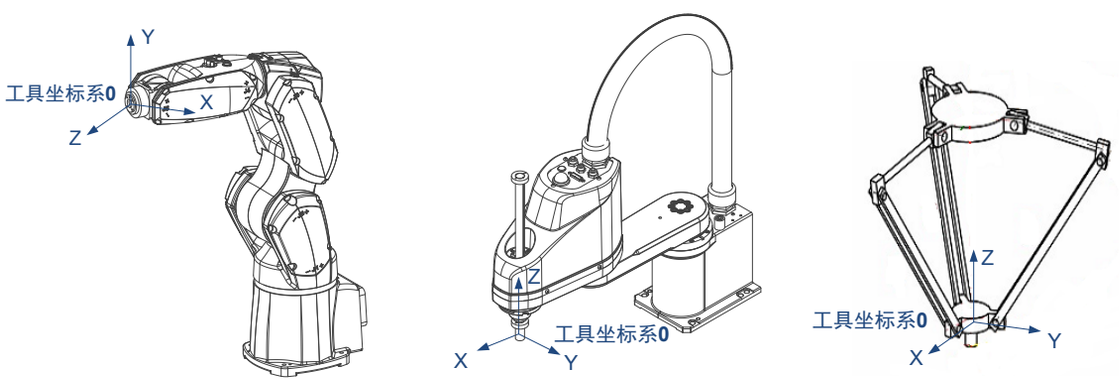
右手定则

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。



在我司机器人控制系统中，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末端。工具 1~15 可由用户自由定义。

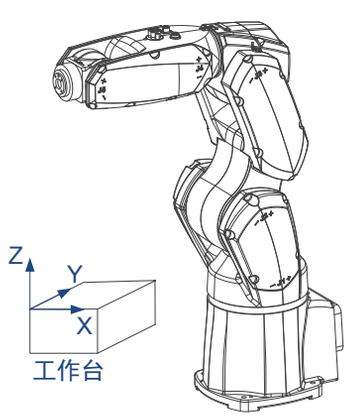


（a）六轴机器人　　（b）SCARA 机器人　　（c）Delta 机器人

图 1-7 机器人工具坐标系

1.2.4 用户坐标系

用户坐标系是用户自定义的坐标系，其原点参数描述为机器人基坐标下的坐标值。用户坐标系本质上可以任意设定，但具体应用中可以将其设置在特定物体上，如 : 工作台、传送带等。



在我司机器人控制系统中，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 可由用户自由定义。

1.2.5 其他坐标系

针对某些特殊的工艺，使用了某些专用坐标系。

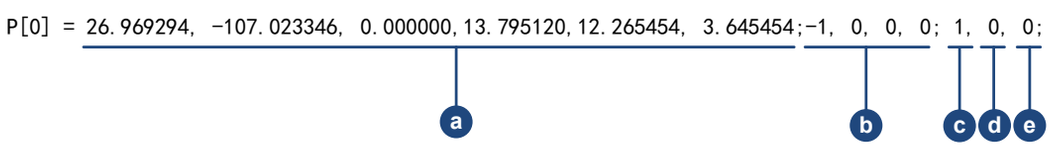
坐标系号	坐标系名	适用工艺	意义	相关
5	固定相机视野坐标系	视觉功能专用	在该坐标系下，定义固定相机视野内的位置点（X,Y,θ）	更多参见 *“1.3.2 特殊工艺的位置变量”*
6	移动相机视野坐标系	视觉功能专用	在该坐标系下，定义移动相机视野内的位置点（X,Y,θ）	
7	传送带上物体坐标系	跟随工艺专用	在该坐标系下，定义相对传送带物体坐标系的位置点（X,Y,Z,A,B,C）	

1.3 位置变量

1.3.1 位置变量参数说明

我司机器人控制系统中，空间中的点用“位置变量”描述。

“位置变量”包含以下信息：坐标值、臂参数、坐标系、工具号、用户号。



a）坐标值：

由坐标系指明坐标值含义。

当坐标系为 1 时，坐标值为机器人各关节值 (J1,J2,J3,J4,J5,J6)。关节值为臂相对于零点位置的旋转角度；

当坐标系为 2 时，坐标值为机器人法兰盘中心点 [1] 相对于基坐标系的位置姿态 (X,Y,Z,A,B,C)；

当坐标系为 3 时，坐标值为工具中心点（TCP）相对于基坐标系的位置姿态 (X,Y,Z,A,B,C)；

当坐标系为 4 时，坐标值为工具中心点（TCP）相对于用户坐标系的位置姿态 (X,Y,Z,A,B,C)。

b）臂参数 [2]：

机器人到达空间同一位置姿态有多种方式，臂参数用来区分这些方式。

c）坐标系号：

取点所用的坐标系，坐标系指明了坐标值的含义。

d）工具号：

当前所使用的工具。

e）用户号：

当前所使用的用户坐标系。

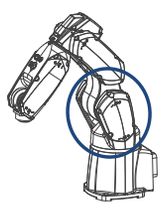
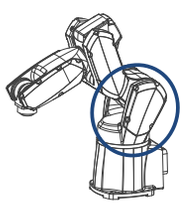
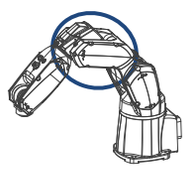
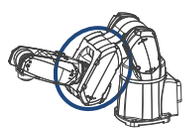
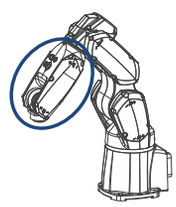
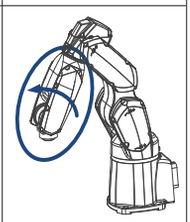
注 [1]：法兰盘中心点：机器人法兰盘末端面上的中心点，代表机器人本体的末端参考位置，后同。

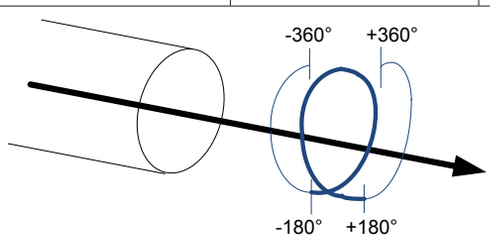
注 [2]：机器人控制目标点到达空间同一位姿时，机器人可能存在几种不同的手臂姿势。

NOTE

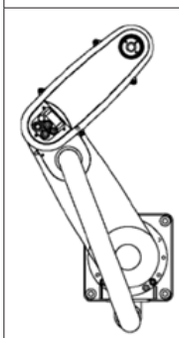
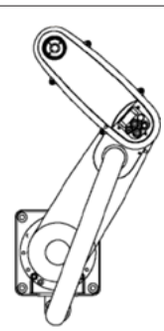
◆ 示教后修改臂参数，将会使机器人以另外一种臂姿势到达空间同一点，机器人运动状态变化很大，需要慎重修改臂参数！

■ **六轴机器人**：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°）
		

■ **SCARA 机器人**：臂参数 1、4 有意义。

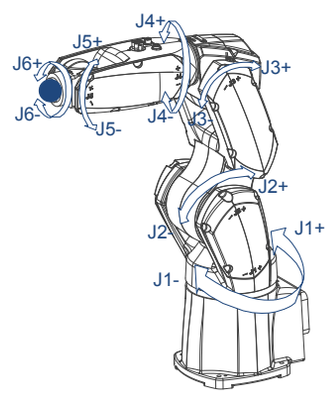
臂参数 1		臂参数 4(关节坐标 J4 的值)		
-1	1	(-540~-180)	(-180~180)	(180,540)
左手臂	右手臂	-1	0	1
		当涉及到多圈的时候，每增加 360 度臂参数加 1，每减少 360 度减 1。		

■ **Delta 机器人**：臂参数 4 有意义。

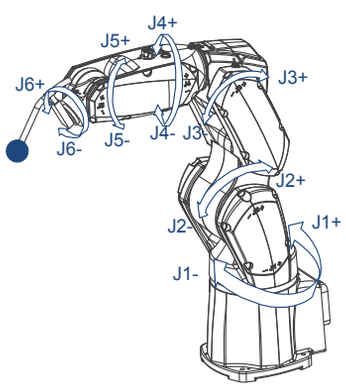
臂参数 4(关节坐标 J4 的值)		
(-540~-180)	(-180~180)	(180,540)
-1	0	1
当涉及到多圈的时候，每增加 360 度臂参数加 1，每减少 360 度减 1.		

■ **位置变量示例**

变量名	坐标值						坐标系	工具	用户号
P[1]	0	0	0	0	-90	0	1	0	0
P[2]	0	0	0	0	-90	0	1	1	0
P[3]	100	0	100	0	0	0	2	0	0
P[4]	100	0	100	0	0	0	2	1	0
P[5]	110	0	60	0	0	0	3	1	0
P[6]	50	0	60	0	0	0	4	1	1

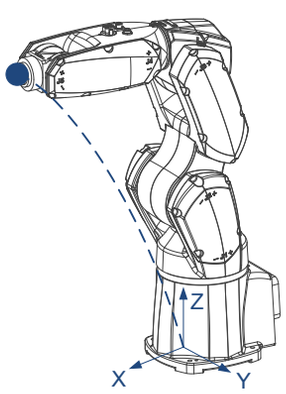


P[1]：不带工具，关节坐标系取点

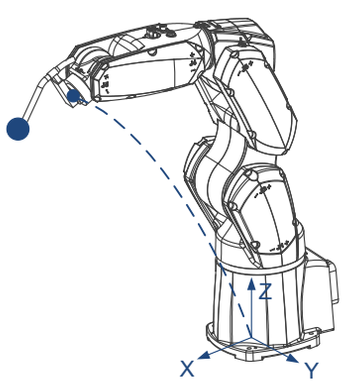


P[2]：不带工具，关节坐标系取点

注意：TCP 位置与工具关联，不由坐标系直接表示。

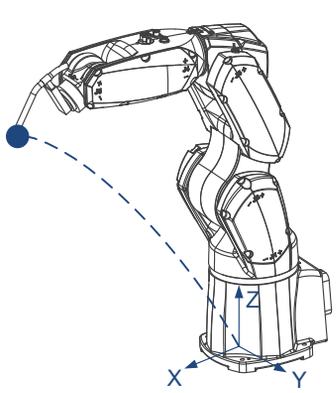


P[3]：不带工具，基坐标系取点

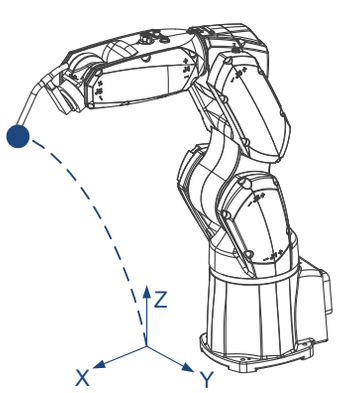


P[4]：带工具，基坐标系取点

注意：TCP 位置与工具关联，不由坐标系直接表示。



P[5]：带工具，工具坐标系取点



P[6]：带工具，用户坐标系取点

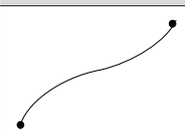
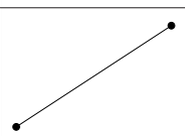
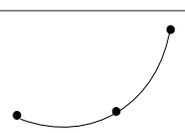
1.3.2 特殊工艺的位置变量

针对某些特殊的工艺，定义了更多的位置变量类型，如下表所示：

特殊工艺点	位置变量类型说明	适用工艺
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上物体的同步运动，代表相对传送带的物体坐标位置。	跟随工艺专用

1.4 运动与过渡

机器人的基本运动可分为以下三类：

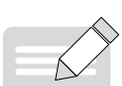
运动种类	轨迹	特点
关节运动 (Movj)		点到点的快速运动，所有轴同时到达目的位置，轨迹通常不在一条直线上。常用于点焊、运输等场合。
直线运动 (Movl)		工具中心点（TCP）线性移动到给定目标位置，轨迹为直线。常用于轨迹焊接、贴装等场合。
圆弧运动 (Movc)		工具中心点（TCP）按圆弧运动移动到给定目标位置。

在实际的连续运动过程中，很多时候运动为了加快节拍，并不需要精确到位，此时，运动的中间点就会表现出轨迹逼近的形式。这种轨迹逼近，业内用“Z”表示。关于 Z 的相关信息，请参考“2 Z”。

1.5 奇异位置

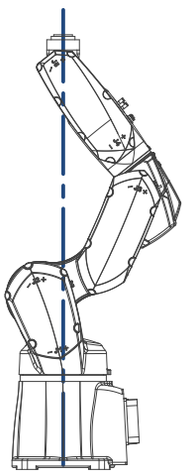
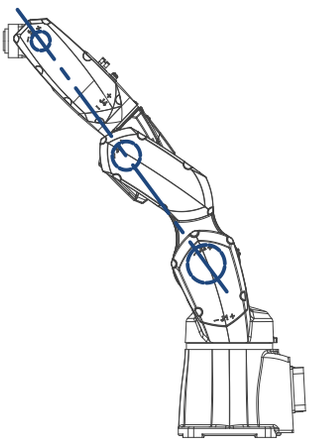
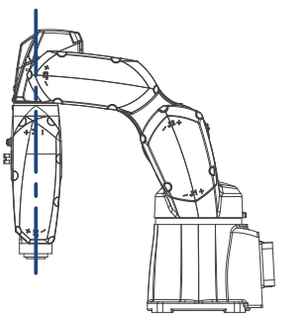
在非关节坐标系下运动时，机器人可能会运动到某些特殊位置，此时机器人会失去一些运动自由，这些特殊的位置称为奇异位置。

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，奇异位置会使机器人不能正常运行。


NOTE

◆ 遇到奇异位置报警时，可切换为关节运动模式，调节机器人姿态退出奇异位置。

■ **六轴机器人的奇异位置**

奇异类型	顶置奇异	延展奇异	手轴奇异
示意图			
说明	J4、J5、J 六轴线交点位于 J1 正上方	J2、J3、J5 关节中心共线	J4 与 J 六轴共线

■ **SCARA 机器人的奇异位置**

SCARA 机器人奇异位置处于 J2=0°时，此时第 1、2 臂摆成一条直线，如下图所示：

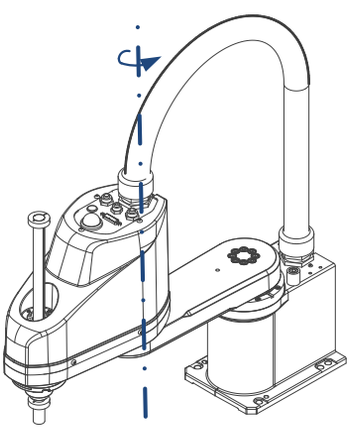


图 1-8 SCARA 机器人的奇异位置

■ **Delta 机器人的奇异位置**

Delta 机器人的奇异位置不在工作范围内。

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。具体工作范围请参考机器人用户手册机械篇相关章节。

干涉区域：设置一些末端执行器禁止到达的区域，避免发生碰撞。

第 2 章 基本操作

2.1 示教器功能简介

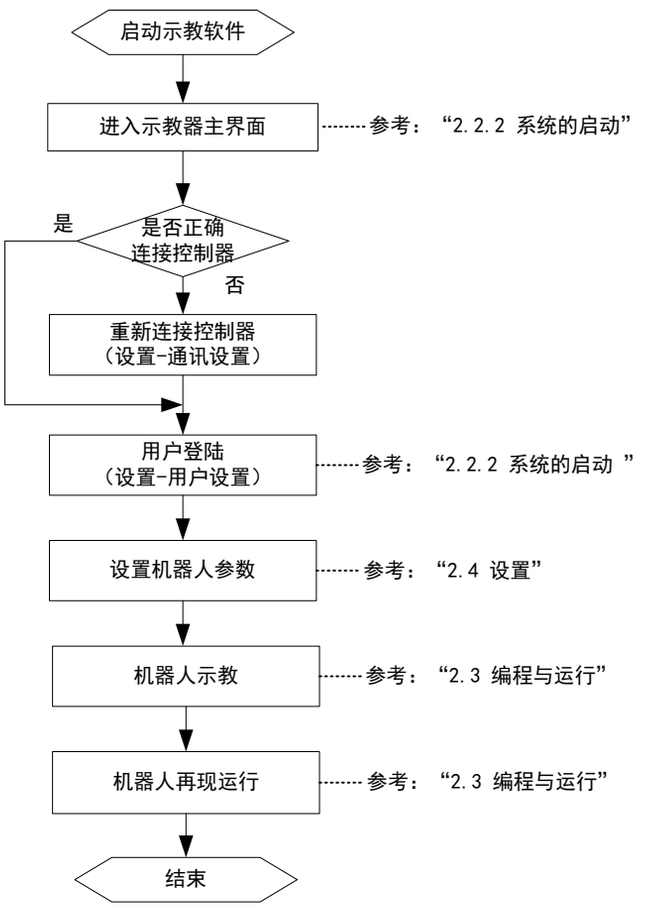
示教器是操作机器人的直接对象工具。在示教器上，有三大界面：编程 / 运行，监控，设置。通过这些界面配置和操作机器人系统。

编程/运行	编程	
	运行	
监控	全局变量	数值变量
		平移变量
		托盘变量
		字符串变量
	局部变量	数值变量
		平移变量
		托盘变量
		字符串变量
	IO监控	IN/OUT
		AD/DA
		Sys IO
	通信状态	
	伺服状态	
	日志	操作日志
		报警日志
	版本信息	
设置	机器人设置	结构参数
		减速比
		耦合参数
		内部补偿参数
	零点设置	绝对零点
		工作原点
		回零修复
	坐标系设置	工具坐标系
		用户坐标系
		工具负载
		工件负载
		臂上负载
	运动参数	示教参数设置
		运动参数设置
		轴参数设置
		干涉区设置
		伺服参数设置
	外设配置	IRLink设置
		I/O设置
	系统设置	通讯设置
		时间日期
		用户设置
		语言设置
		自定义设置
		多任务设置
		其他设置
	功能扩展（可选）	视觉标定
		码垛工艺
		跟随工艺
		螺丝工艺

2.2 软件使用入门

2.2.1 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，示教器会根据上一次的连接地址自动连接。示教器连接成功后进入“设置—系统设置 - 用户设置”界面登陆，对相关参数进行设置后进行示教，示教完成后运行。



2.2.2 系统的启动

2 上电与连接

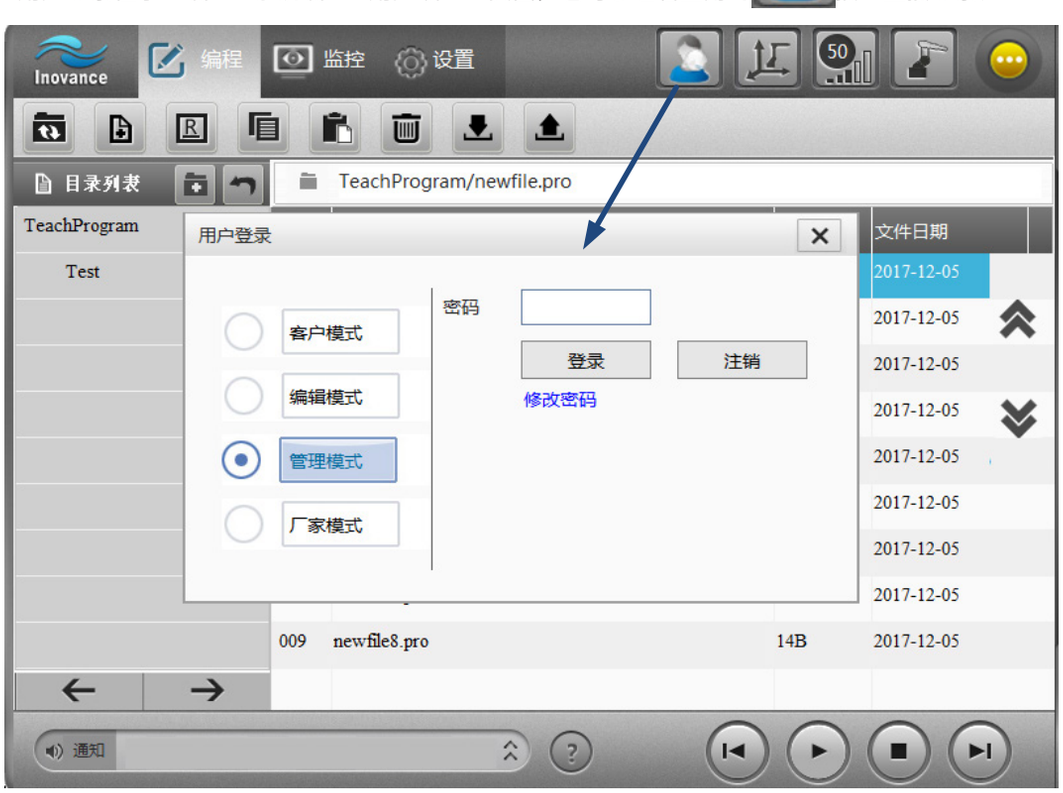
通过控制柜的总开关给系统上电。示教器启动后进入连接界面，连接成功后进入主界面。

NOTE

- 对于手持示教器，与控制柜是直接连接的，默认上电后自动连接；
- 对于 PC 版示教器，如果开始时不能连上控制器，需点击“跳过”按钮，转到“设置 - 系统设置 - 通讯设置”页面，输入 IP 地址后连接。

3 用户登录

用户登录位于“设置 - 系统设置 - 用户设置”页面，也可通过右上角的  按钮直接登录。



根据当前使用者的角色，选择模式并登录。客户模式无需密码，可直接控制机器人运动和运行程序。

不同用户模式请参阅：*“3 用户设置”。*

4 检查

连接成功后，请检查右上角的状态灯与左下方的消息提示栏。

只有当状态灯显示为待机或使能状态时，才表示正常。

其他情形表示异常，根据左下方的消息栏排查错误。

正常　　异常

2.2.3 主界面功能介绍

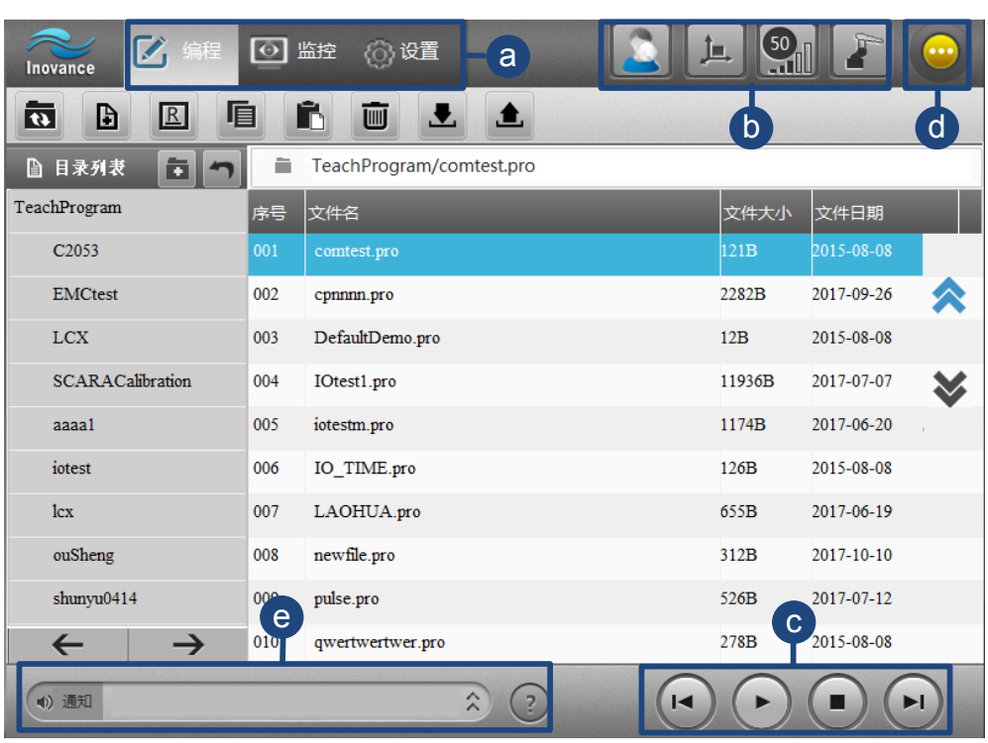
示教软件按功能可划分为三个模块：

编程模块：示教过程的主操作模块，主要用于示教编程和再现运行。

监控模块：观察程序参数、接口状态、日志等各项数据，也支持修改变量参数。

设置模块：进行机器人相关的设置，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。

示教软件启动并连接成功后，显示的主界面如下：



a）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。

b）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 右上角数字代表选用的工具号。
		用户坐标系 右上角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
		切换到速度倍率调节
		切换到寸动设置
		设置寸动值： 关节步长 0.2° 位置步长 0.2 mm 旋转步长 0.2°

速度倍率 / 寸动选择按钮		设置寸动值： 关节步长 1° 位置步长 1 mm 旋转步长 1°
		设置寸动值： 关节步长 5° 位置步长 5 mm 旋转步长 5°
		用户自定义寸动值 （取“设置 - 运动参数 - 示教参数设置 - 寸动”中的值）
轴组切换按钮		摇杆控制的轴组 J1/J2/J3（X/Y/Z）
		摇杆控制的轴组 J4/J5/J6(A/B/C)

c）运动控制栏

运动控制栏用于控制程序的执行，包括启动 / 暂停、停止、单步前进、单步后退四个按钮。可在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 [1]	按下单步后退，松开停止	无效
	启动 / 暂停	点击启动，连续运行程序；松开暂停	点击启动； 再次点击暂停
	停止	无效（由于示教下是点动模式，松开即停止）	一经按下立即终止运行，下次启动时从程序开头重新执行。
	单步前进	按下单步前进，松开停止	无效

注 [1]：单步后退特性：

1） 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”。

2） 单步后退只针对运动指令，对于非运动指令不做处理。

3） 后退最多支持 9 步。

4） 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。


NOTE

◆ 对于 movc 的后退：如果后退的动作没有退到圆弧的起始点，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。

d）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：急停按钮松开，伺服被使能。只有在使能状态下机器人才能运动。
	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，伺服尚未使能。
	报警状态：出现异常，需要立即处理。
	警告状态：出现异常，提示用户。
	断线状态：网络连接断开，不能与控制器通信。

e）消息窗口

消息窗口能显示提示信息和报警信息。例如在客户模式下进行 SD 卡格式化操作（SD 卡格式化需要管理者权限，高于客户权限），消息窗口会出现权限的提示信息。

报警信息及处理方法参考*《汇川机器人设计应用与维护手册 - 附录：机器人报警及处理方法》*。

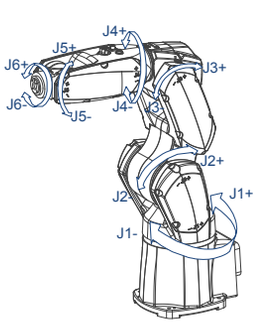
2.2.4 操纵机器人移动

1 操纵机器人移动的步骤

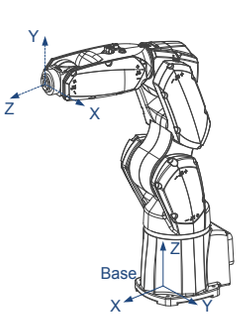
机器人移动前，先选定坐标系和速度（或寸动），按住使能开关（PC 版本示教软件，点击使能按钮保持使能即可），再点击示教面板操纵移动。

步骤 1：坐标系、速度 / 寸动的选择

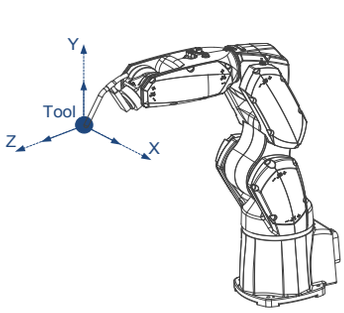
坐标系决定了运动的方向。



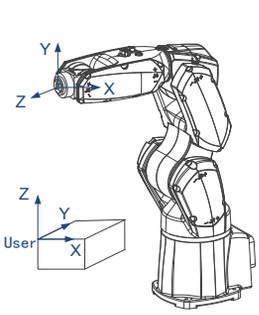
沿关节坐标系移动



沿基坐标系移动



沿工具坐标系移动



沿用户坐标系移动

工具栏上速度设置有 5%、25%、50%、100% 四档，可通过示教器上的按键  和  微调。

此外，也可以选择寸动。选择为寸动时，机器人将以步长衡量运动，每次只运动一步。寸动具有 G1、G2、G3、U 四档，详情请参阅：*“2.2.3 主界面功能介绍”。*

◆ 寸动模式的速度仍与速度倍率相关！

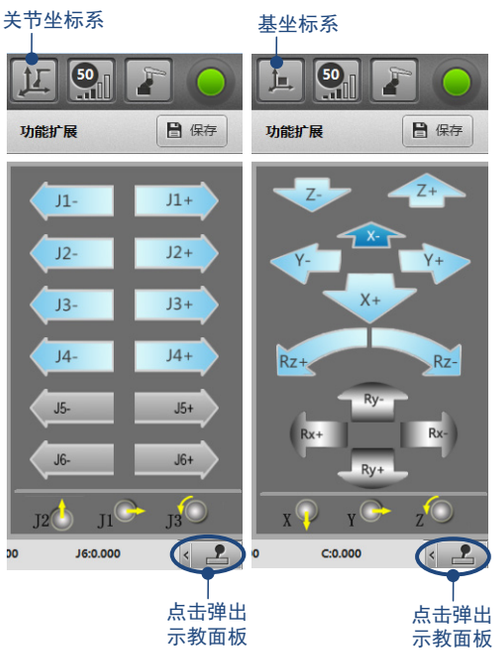
步骤 2：使能并操纵机器人运动。

操作过程中，如果是沿工具坐标系移动或沿着用户坐标系移动，则需注意当前使用的工具和用户坐标系。

NOTE

- 只有在使能状态下，机器人才能运动。
- 手持式示教器的使能方法：一直按住使能开关；
- PC 版示教软件的使能方法：点击使能开关，系统会保持使能！

当面板右下角有手柄图形按钮时，点击可弹出示教面板，用来控制机器人移动。示教面板有两种形式，关节和非关节，根据当前所选择的坐标系显示。

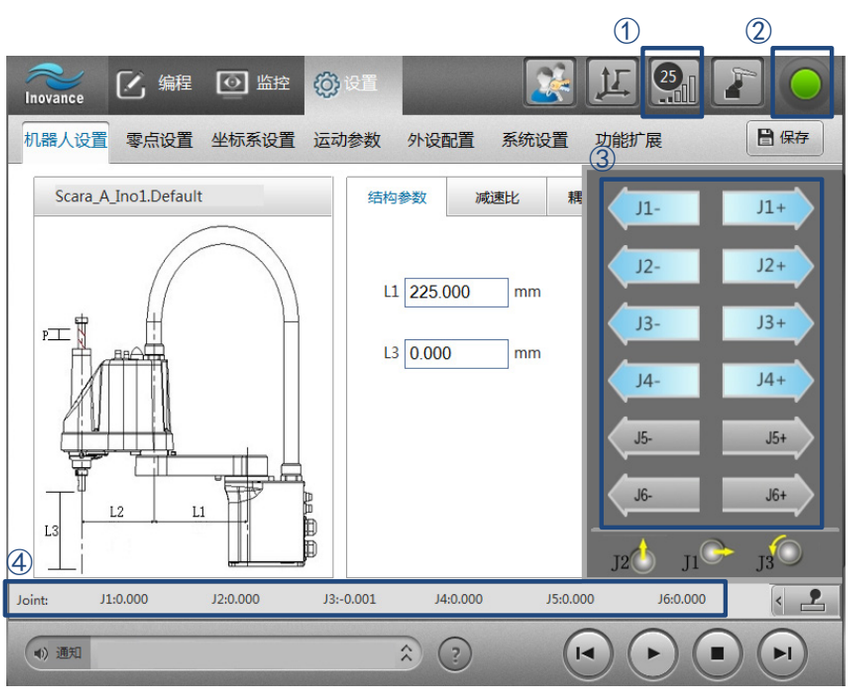


NOTE

- 对于 IRTP80 示教器，可使用示教器上的物理按键进行机器人操纵；
- 对于 ITP100，可以使用摇杆进行机器人操纵，详见*“7 摇杆操作”*。

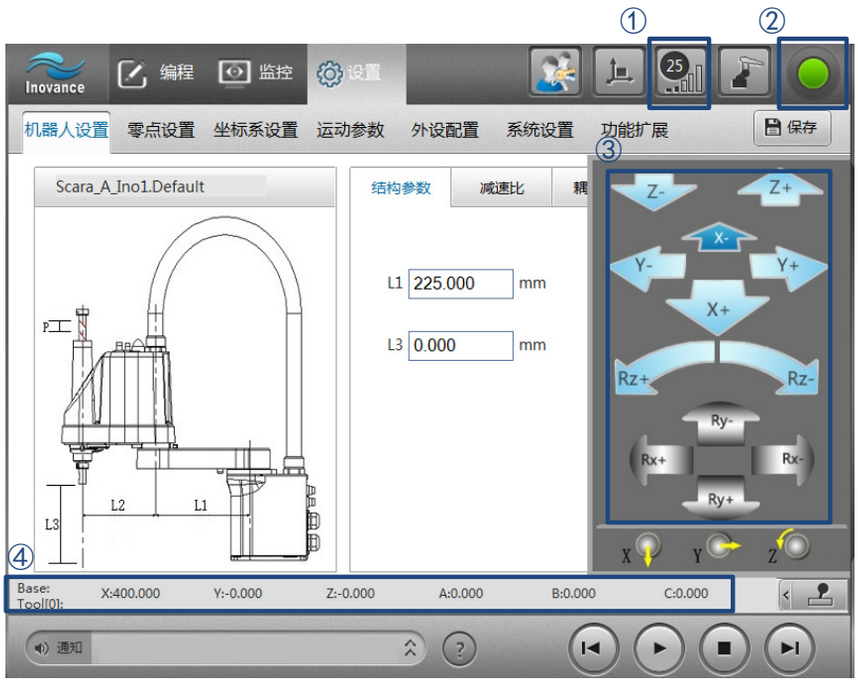
2 练习：在关节坐标系下运动

选择关节坐标系，选择速度 25%，按住使能，操作运动 (J1~J6)，观测坐标值。



3 练习：在基坐标系下运动

选择基坐标系，选择速度 25%，按住使能，操作运动 (X,Y,Z,Rx,Ry,Rz) ，观测坐标值。

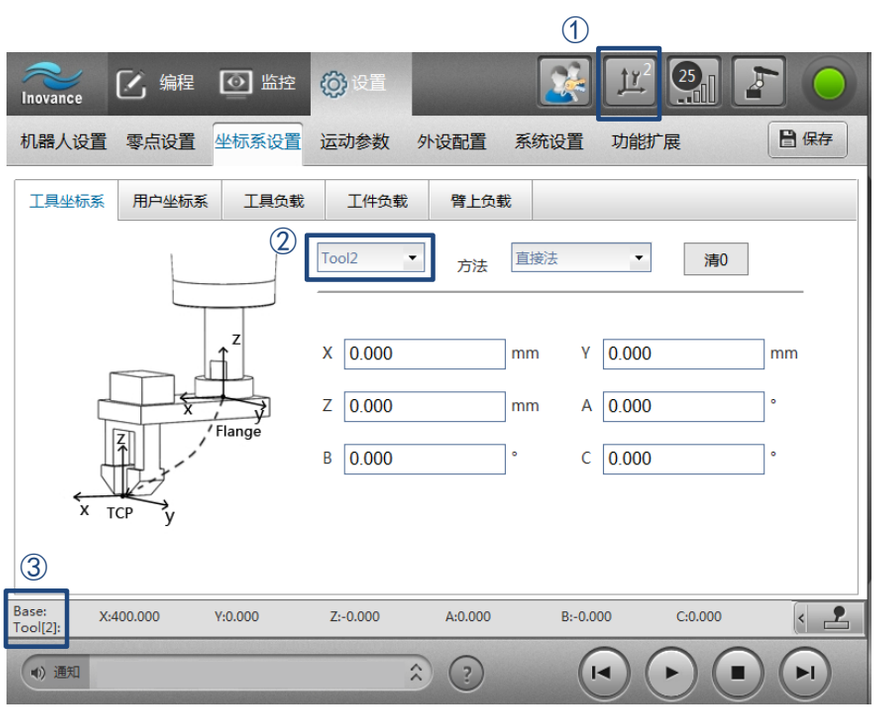


4 练习：在工具坐标系下运动

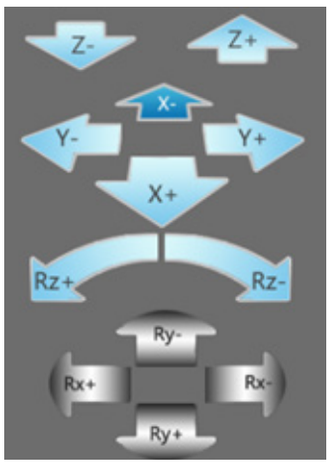
步骤 1：选择工具坐标系，选择速度 25%

步骤 2：在【设置】-【坐标系设置】-【工具坐标系】设置页面选定工具 Tool2。

此时：工具坐标系图标上显示工具号；坐标栏上会显示 Base:Tool[2]，表示当前的坐标值是 Tool[2] 相对于 Base。



步骤 3：按住使能，通过示教面板控制机器人运动 (X,Y,Z,Rx,Ry,Rz)



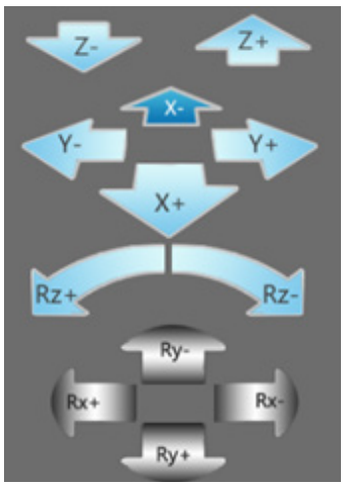
5 练习：在用户坐标系下运动

步骤 1：选择用用户坐标系，选择速度 25%

步骤 2：在【设置】-【坐标系设置】-【工具坐标系】设置页面选定工具 Tool2，在【用户坐标系】设置页面选定用户坐标系 User3，点击“保存”。

此时：用户坐标系图标上显示用户号；坐标栏上会显示 User[3]:Tool[2]、，表示当前的坐标值是 Tool[2] 相对于 User[3]。

步骤 3：按住使能，通过示教面板控制机器人运动 (X,Y,Z,Rx,Ry,Rz)

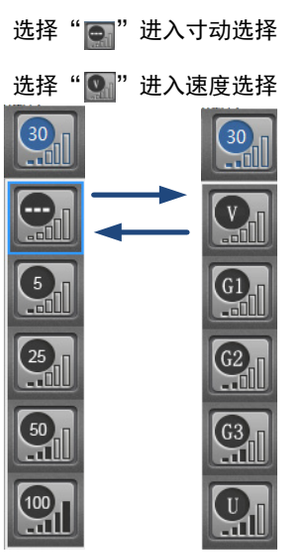


6 练习：寸动

以关节坐标系下寸动为例说明。在其他坐标系下寸动第一步选择寸动，后续步骤与速度模式下运动一致。

步骤 1：选择速度按钮，按住  进入寸动选择；练习以选择 G2 为例。

不同寸动按钮请参阅：*“2.2.3 主界面功能介绍”。*



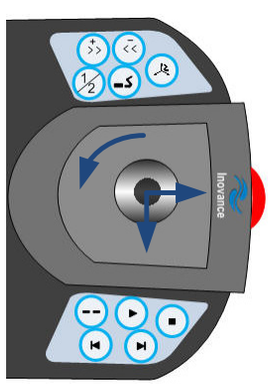
步骤 2：选择关节坐标系，按住使能，通过示教面板控制机器人运动。

7 摇杆操作

1） 摇杆操作说明

ITP100 示教器带有摇杆，可操纵摇杆运动。

示教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。如下表所示：

摇杆正方向	摇杆控制机器人运动方向	
	关节坐标系、轴组 1：	
	关节坐标系、轴组 2	
	基、工具、用户坐标系，轴组 1：	
	基、工具、用户坐标系，轴组 2：	

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。

NOTE

◆ 寸动模式下，禁止复合摇动摇杆。若多个方向摇动摇杆，机器人只在摇杆偏移最多的方向寸动。

摇杆的力度控制：摇杆偏移的幅度越大，运动的速度越快。

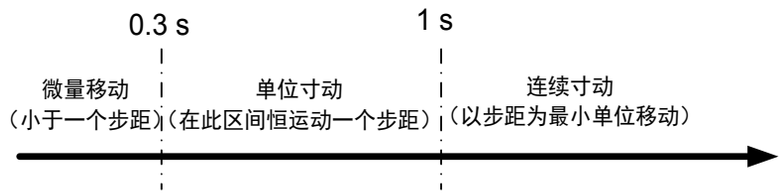
NOTE

◆ 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。

◆ 寸动模式下，摇杆的力度控制不生效，即小幅度偏移摇杆与大幅度偏移摇杆，寸动的速度不会受其影响。

2） 寸动模式的摇杆控制

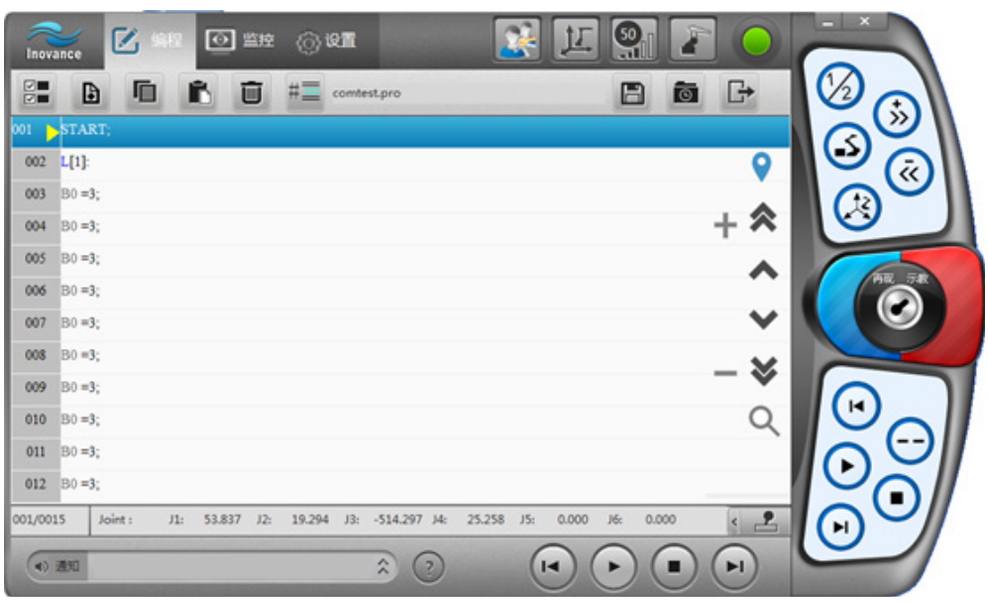
在寸动模式下，摇杆控制的运动将会根据摇杆偏离中间位置的时间实行分段处理：



寸动类型	摇杆偏移时间	适用场合
微量移动	<0.3s	适用于微小距离移动（小于一个步距）场合。
单位寸动	0.3s~1s	适用于只运动一个步距的场合。
连续寸动	>1s	适用于以步距为单位、连续多步运动的场合。

2.2.5 PC 版本虚拟按键介绍

PC 版的示教软件界面如图所示：



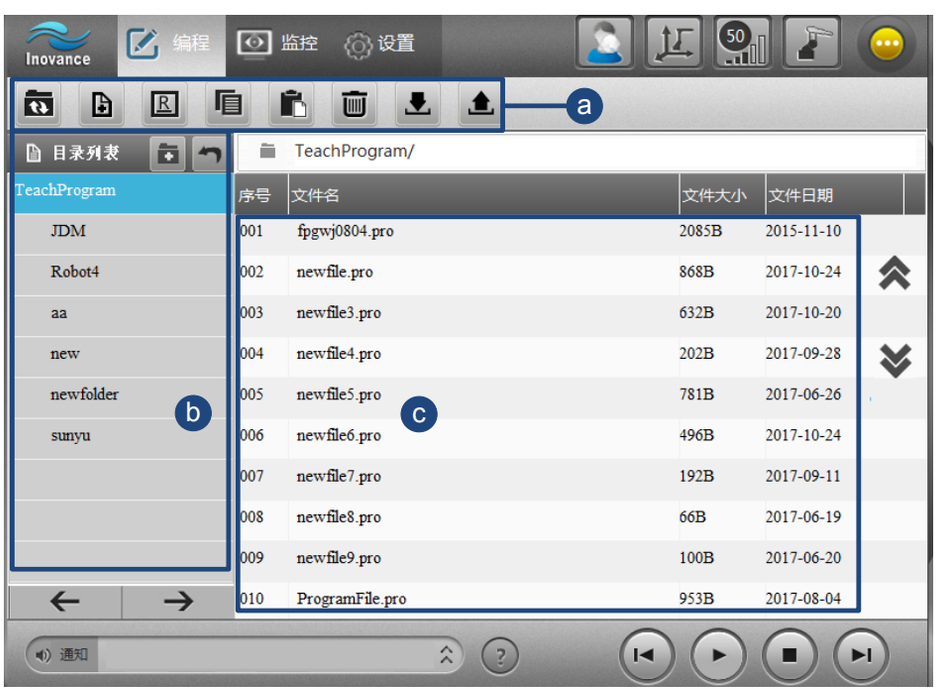
按键	按键名称	按键功能
	左侧蓝色：使能 中间：模式切换 右侧红色：急停	使能：控制伺服电机的使能状态。 模式切换：切换机器人示教 / 再现模式。 急停：控制机器人紧急停止。
	速度增	速度增加，按下本按钮，速度值增 1%； 长按本按钮，速度持续上升。
	速度减	速度减少，按下本按钮，速度值减 1%； 长按本按钮，速度持续下降。
	轴切换	摇杆控制轴切换。 1/2/3（X/Y/Z）轴和 4/5/6(Rx/Ry/ Rz) 轴组之间切换。
	外部轴切换	当机器人有外部运动轴时，按下本按钮切换至外部运动轴。
	坐标系选择	进行坐标系（关节坐标系、基坐标系、工具坐标系、用户坐标系）的切换。
	示教盒版本：示教 / 再现切换 PC 版本：寸动	示教盒版本：示教 / 再现切换按钮。 PC 版本：寸动量调整。

按键	按键名称	按键功能
	运行	运行模式下，按下本按钮可选择程序，机器人再现模式下运行所选程序； 示教模式下，按下本按钮，程序连续运行，松开本按钮，程序暂停运行。
	停止	机器人运行时，选择本按钮，机器人停止运行。
	单步前进	示教模式下，运行当前行
	单步后退	示教模式下，后退上一步运行

2.3 编程与运行

2.3.1 编程面板介绍

编程面板如下图所示：



a）文件编辑工具栏

通过工具栏可进行以下操作：

刷新：刷新当前目录。

新建：新建程序文件。（文件夹新建利用目录列表上的  ）

重命名：重命名程序文件或文件夹。

复制：复制程序文件或文件夹。

粘贴：粘贴程序文件或文件夹。

删除：删除程序文件或文件夹。

导入[1]**：**从外界导入一份程序文件到控制器。离线状态下，导入到示教器本地的离线目录下。

导出：从控制器中导出一份程序文件。离线状态下，将示教器本地的离线文件导出。

注 [1]：PC 版本示教软件支持拖动导入。

b）**文件夹列表**

新建：新建文件夹

返回：返回上一级文件夹

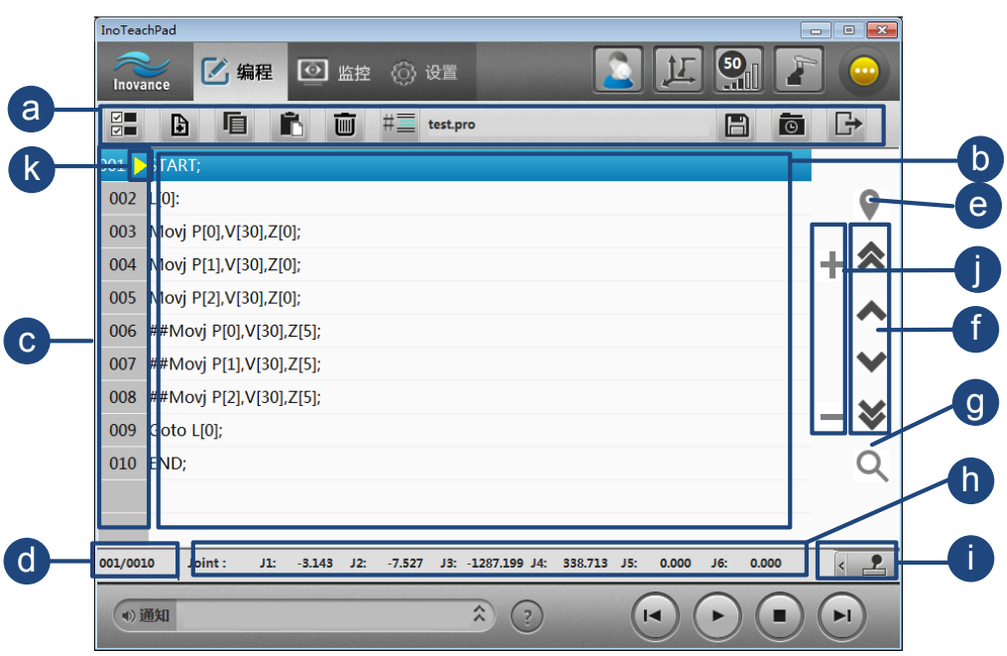
文件夹列表：单击文件夹名时，显示当前文件夹下文件；双击文件夹名时，进入文件夹。

NOTE

- 文件夹的重命名、复制、粘贴、删除可通过文件编辑工具栏按钮进行操作。
- 文件夹名称由字母、数字和下划线组成，首位必须为字母，长度不超过 16 个字符。
- 文件夹嵌套最多三层。每个文件夹下限制文件数量为 512 个。

c）**程序列表及编辑界面**

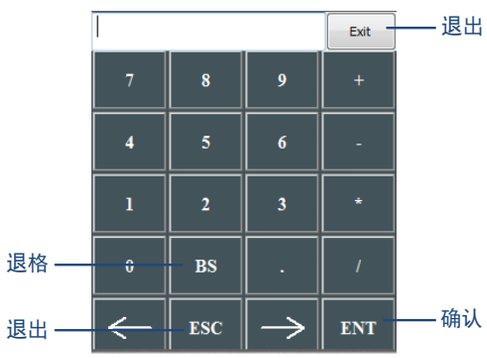
程序列表显示当前文件夹下的所有程序，点击列表头可进行文件名、大小或是日期的排序。右侧的  和  可进行上下翻页。双击程序名可进入程序编辑界面，如下图所示：



序号	功能	描述
a	程序编辑工具栏	对程序文件中的指令进行编辑操作
b	程序指令编辑区	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。双击选中行可修改该行指令。

序号	功能	描述
c	程序行号区	显示每条指令所在行的行号。
d	当前行 / 总行数	显示被选中的指令行号和总行数。
e	定位按钮 [1]	填入行号，快速跳转。
f	翻页按钮	单箭头：光标跳转上下一行 双箭头：程序翻页。
g	搜索 / 替换按钮	查找下一个：查找下一个对象。 替换：替换当前选中的对象。 全部替换：从当前选中行开始，全部替换。
h	坐标显示区	显示机器人当前坐标信息（根据坐标系显示）
i	示教面板按钮	显示 / 隐藏示教面板，用于控制机器人运动
j	缩放按钮	编辑区的缩放（仅适用 ITP100 和 PC 版本示教器）
k	启动行标识 [2]	以黄色箭头标示启动行，用来表明程序的启动位置。

注 [1]：定位按钮弹出小键盘如下图所示：



注 [2]：启动行标识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。在程序非运行过程中，人为点击列表，程序选中行变更，启动行不会变更。

2.3.2 运行面板介绍

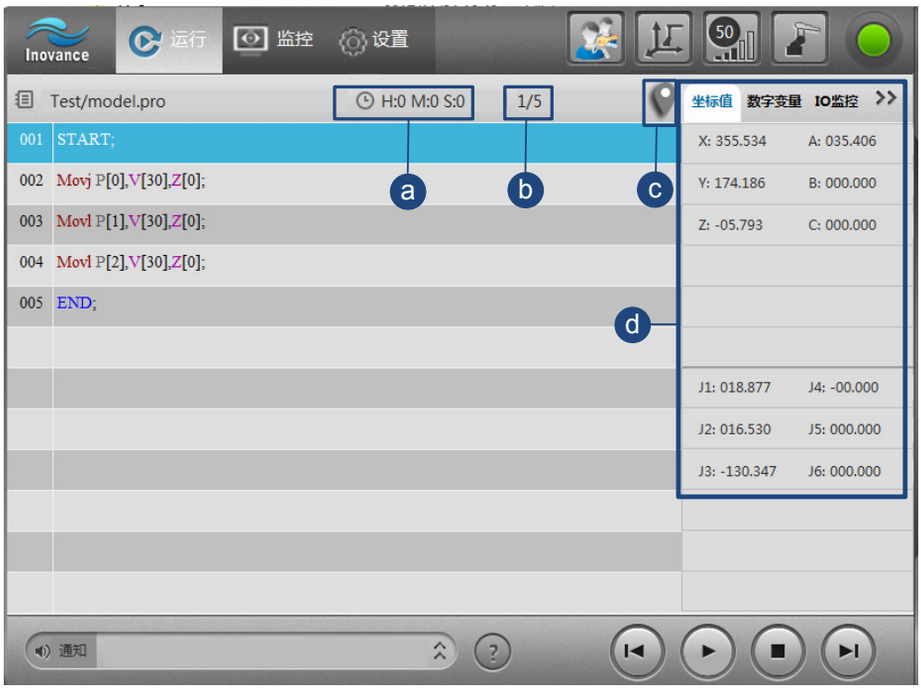
切换到运行模式，系统会自动使能，此时可运行当前程序。

NOTE

◆ 对于不同的示教器，切换按钮位于不同的位置：

1. 对于 ITP100 示教器：利用控制柜上的模式切换旋钮，切至再现运行模式。
2. 对于 IRTP80 示教器：利用示教器上的  按钮切换。
3. 对于 PC 版本示教器，利用  按钮切换

运行界面如图所示：



a）运行时间

程序启动到停止所用的时间。

NOTE

- “运行时间”不是单个程序的运行时间。
- “运行时间”在多程序连续运行（如远程 IO、主子程序调用）非常有用，提供多个程序运行的总时间。

b）行号

显示当前行号 / 总行数。

c）跳转

用于设置光标行。设置后，程序将从光标行开始执行。

d）快捷监控面板

提供了当前坐标值信息和收藏的部分数据信息。

坐标值：显示机器人基坐标系下数值（不含 Tool 和 User）和关节值。

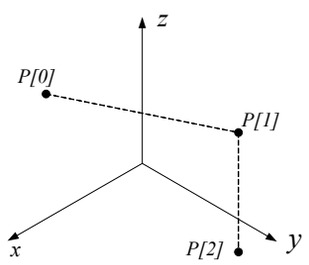
数字变量：显示收藏的 BRD 数值变量的前 10 项。

IO **监控：**显示收藏的 IO 数值变量的前 10 项。

2.3.3 练习：示教编程与运行

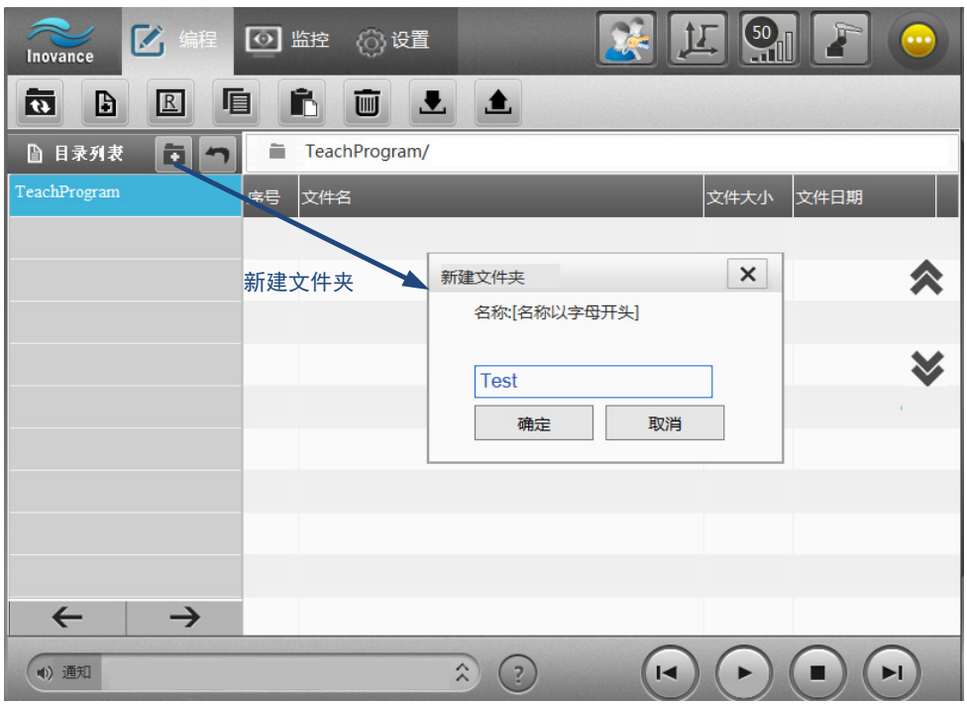
1 任务

新建程序文件夹“Test”，在文件夹下新建程序“model.pro”，编辑程序使机器人运动路径为：P[0]-P[1]-P[2]。编辑完成后，试运行程序，然后切换至再现模式，正常运行程序。

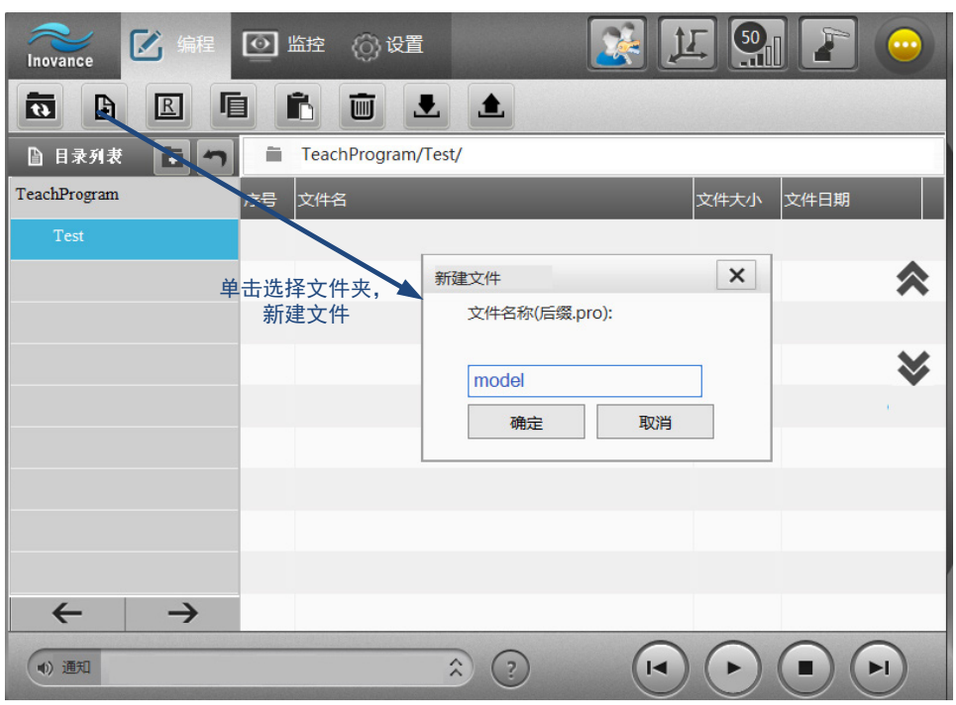


2 操作步骤

步骤 1：在默认的总目录“TeachProgram”下新建文件夹“Test”；

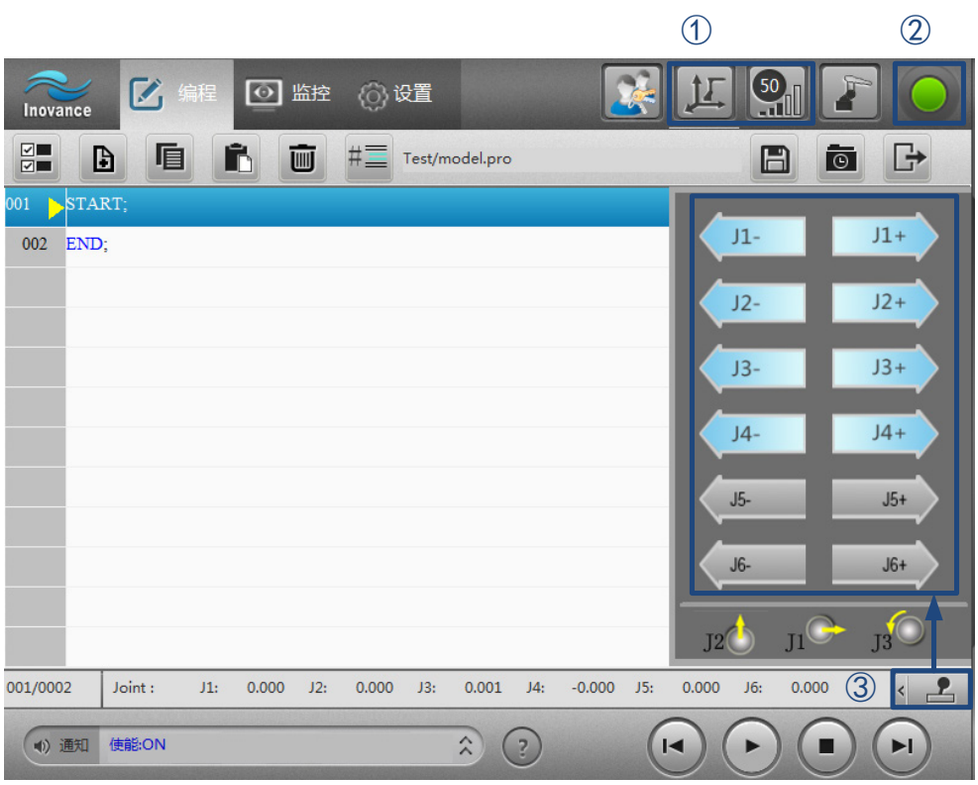


步骤 2：单击选择“Test”文件夹，新建程序“model.pro”；

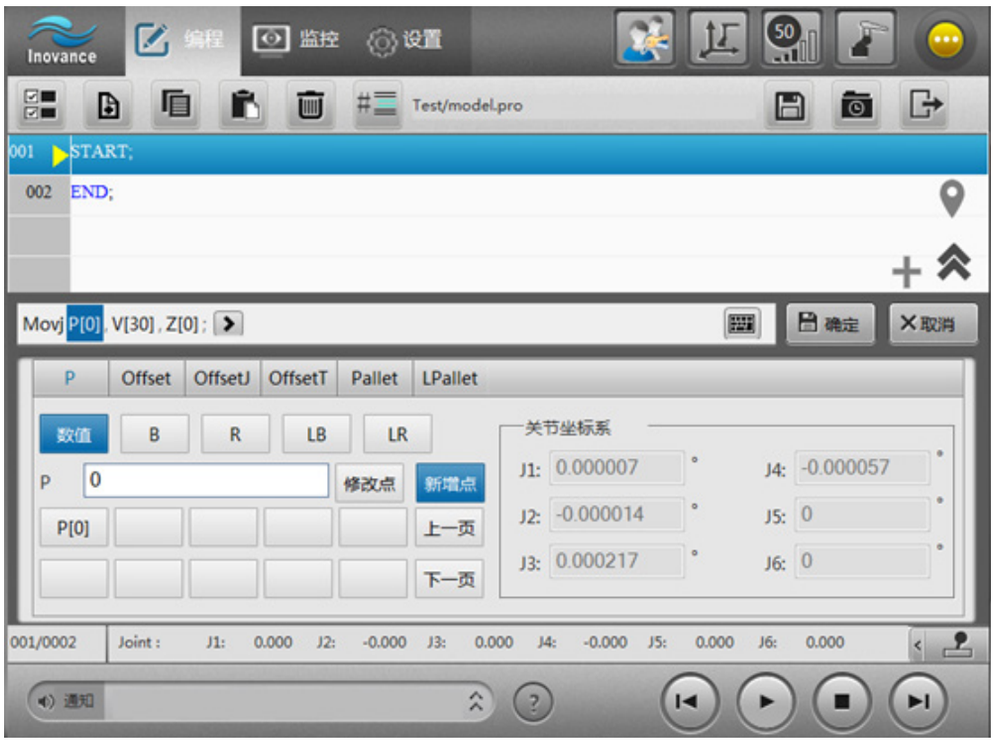


步骤 3：双击“Model.pro”，进入程序编辑界面；

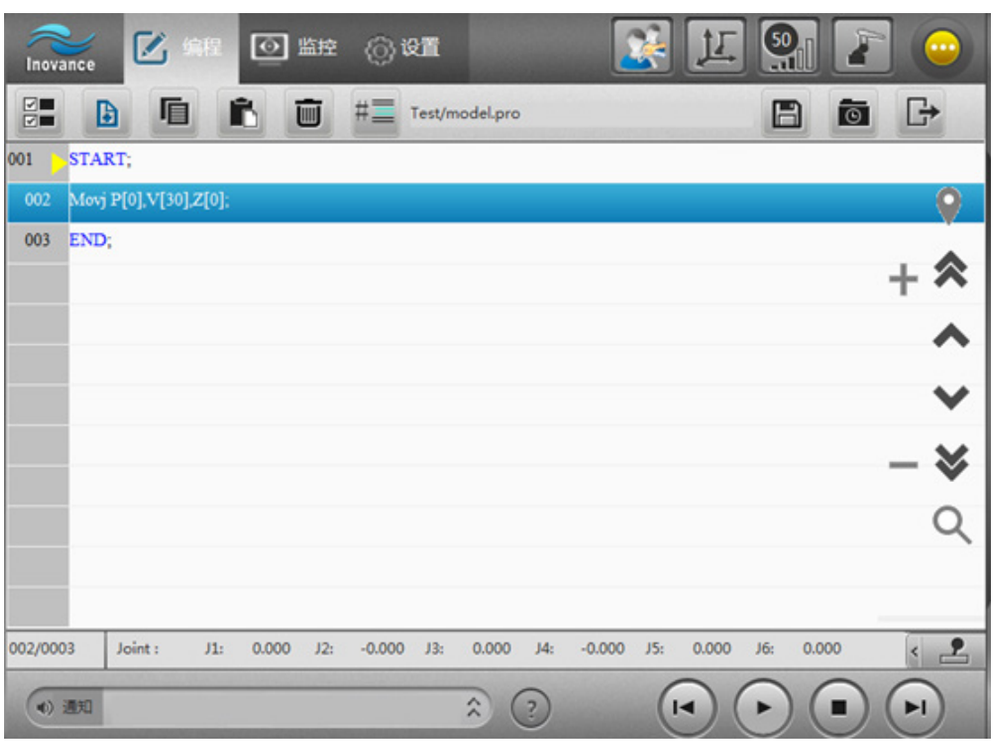
步骤 4：点击示教面板，选择合适的坐标系、速度，上使能，示教到 P[0] 点；



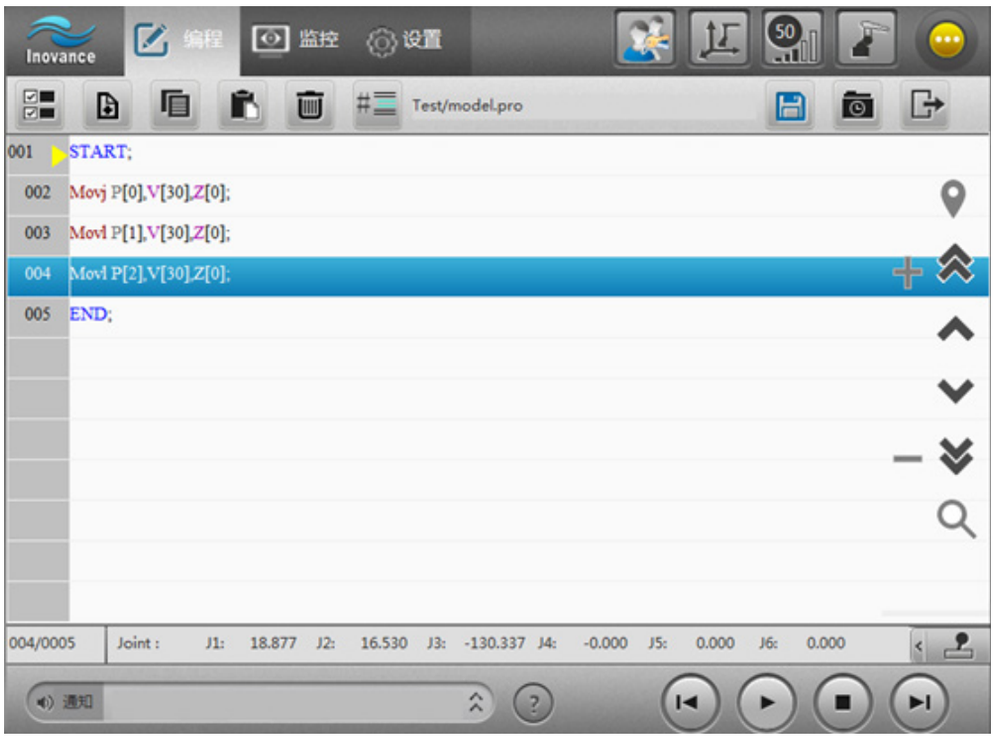
步骤 5：新建指令，在运动指令中选择 Movj，点击“新增点”；



步骤 6：点击确定后，指令显示在程序中；



步骤 7：参考步骤 4 完成 P[1] 和 P[2] 的示教，注意：通过 Movl 方式运动到 P[1] 和 P[2] 点。



步骤 8：程序编辑完成后，点击保存。然后上使能，光标行置为第一行，按住连续运行，直至程序运行结束。观察期间机器人运动；

步骤 9：切换到运行模式，点击启动程序运行。

2.3.4 离线编辑功能

我司示教器产品支持离线编辑功能，即在未连接控制器的情形下，可以离线编辑机器人程序。与普通的在线编辑模式相比有以下优点：

1） 高效：减少机器人不工作的时间；

2） 舒适安全：给编程者创造更舒适、安全的工作环境；

3） 教学：不必连接控制器，便于学习编程。

1 离线编辑功能

离线编辑功能包括：

1） 离线程序目录 [1] 下的文件或文件夹的操作（新增、复制、粘贴、导出等）；

2） 指令编辑；

3） 位置变量编辑；

4） 离线编辑完成，支持导入、导出 [2]，方便的与外部设备或控制器交互。

注 [1]：离线程序目录：当前示教软件应用程序目录下 \OfflineTeachProgram；

注 [2]：关于导入、导出：

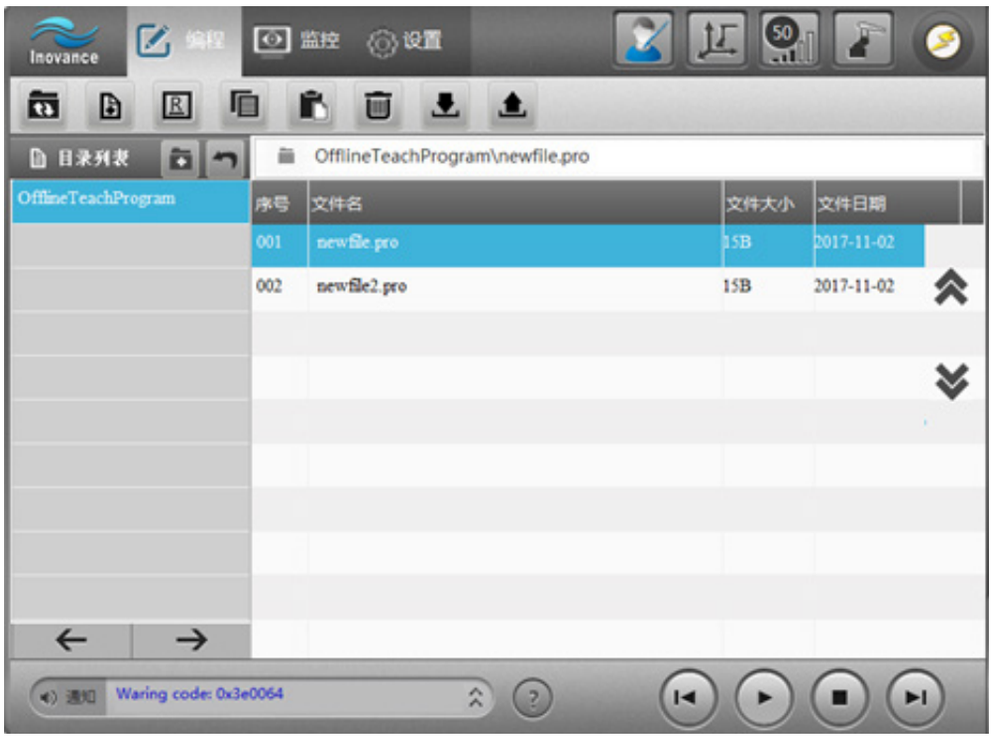
	导入	导出
在线（连接控制器）	程序文件导入控制器	将控制器中的文件导出
离线（未连接控制器）	程序文件导入示教器离线目录	将示教器离线目录下的文件导出

2 操作步骤

步骤 1：进入编辑模式

在离线状态下，用户模式选择“编辑模式”，输入密码：000000，即可编辑程序。

离线编辑的程序，默认目录为当前应用程序所在目录 \OfflineTeachProgram 文件夹下。



步骤 2：在离线编辑模式下，默认取点均为基坐标系下零点位置（0,0,0,0,0,0;0,0,0,0,0;2,0,0），在【监控】-【局部变量】-【位置变量】中可对位置变量进行编辑。

步骤 3：编辑完成后，可导出到 U 盘或本地 PC 机，然后连接控制器，通过文件导入功能，将编辑的程序导入至控制器。

a）导入　　b) 导出

2.3.5 *PC 示教器快捷键

对于 PC 版本示教软件，编程界面支持的快捷键如下表：

快捷键命令	功能说明
CTRL +A	全选，同时调出多选栏。
CTRL +Q	调出多选栏
CTRL +C	复制
CTRL +V	粘贴
CTRL +D	注释行
CTRL +X	新建指令，打开指令列表
CTRL +F	打开查找功能
CTRL +G	打开跳转行号功能
CTRL +S	保存文件
CTRL + ↑	放大
CTRL + ↓	缩小
CTRL + R	快速新建一条运动指令
Tab	增加行缩进
SHIFT + Tab	减少行缩进
指令搜索	打开指令列表页面，直接输入字符串即可搜索指令；按键退格、Del 回删搜索字符串；ESC 取消搜索，回到常用指令列表。

2.4 设置

在出厂时，已经完成了初始设置。如果您是用户，在投入运行前，一般只需根据需求，进行【零点设置】、【坐标系设置】两项内容的设置，可直接参看这两部分的内容。如果您是厂家调试人员，则需要关注更多。

NOTE

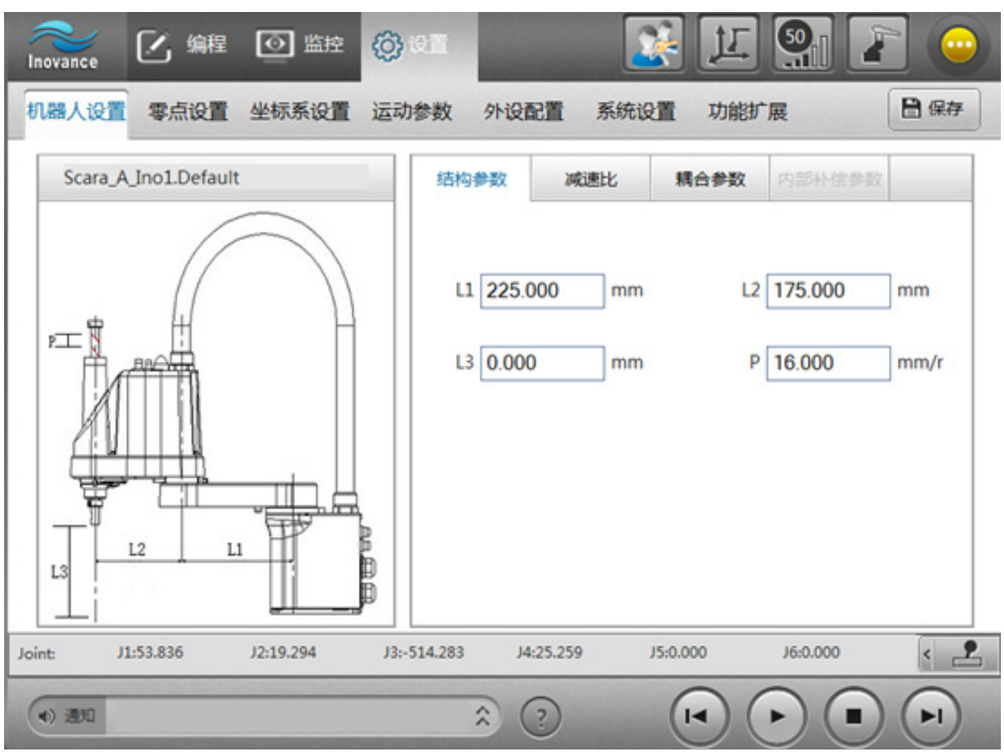
◆ 修改当前页面参数后，记得点“保存”，保存当前页面信息！

2.4.1 机器人设置

机器人设置包含结构参数、减速比、耦合参数等，按照实际情况填写。

NOTE

◆ 机器人设置需要厂家模式下进行。



2.4.2 零点设置

零点设置页面包括：绝对零点、工作原点。

SCARA 机器人零点设置页面包括：绝对零点、工作原点，回零修复。

1 绝对零点

绝对零点设置通过以下步骤完成：

步骤 1：通过示教器调整机器人关节到零点位置，点击“当前值”按钮，编码器数值显示于面板的编辑框中。

步骤 2：按下急停键，点击“保存”按钮，完成后重启。

NOTE

◆ J1~J6 编码器数值范围为 -2^{21} ～ 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。

2 工作原点

工作原点的位置由用户自行定义，共可设置 5 个工作原点。

回工作原点方法：在程序中通过“Home”指令回到工作原点。

工作原点可手动输入，也可移动机器人到理想位置，点击“当前值”自动获取。完成后，点击“保存”。如图所示。

3 回零修复

该功能仅适用 SCARA 机器人。由于撞击、参数设置不合理等原因，可能导致零点丢失，此时可利用回零修复功能找回原来的零点，避免了重新设置绝对零点后，原有的程序不再适用的问题。

■ **“回零修复”步骤**

步骤 1：选择机型，勾选要修复的轴；

步骤 2：按右侧界面上的操作事项调整机器人到指定位置；

步骤 3：点击“启动修复”，等待自动修复完成。

■ **“回零修复”页面按钮说明**

停止修复：当修复过程需要紧急停止时使用。

高级设置：厂家调整回零配置，需要在厂家模式下设置。

2.4.3 坐标系设置

在坐标系设置中定义工具坐标系和用户坐标系，对于需要考虑负载的场合，还需配置工具负载、手持工件负载、臂上负载。

1 工具坐标系

工具坐标系建立在工具上，其参数为 TCP 在机器人末端坐标系下的位姿偏移。

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，再通过另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

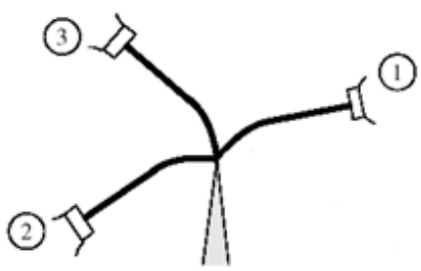
方法	特点	适用场景
五点法 TCP+ZX	通过五点对位获取工具的位置，再通过另外三点获取工具的姿态	需要人工标定工具坐标系的位置和姿态值。取点更多，比三点法 TCP+ZX 更加准确。
锁螺丝四点法	通过选取水平面上的四个点，依次操作机器人使电批末端对准	适用于锁螺丝机型

1）　直接法

直接输入参数即可。

2）　三点法 TCP

在机器人末端安装上工具后，调整工具的位姿，使 TCP 以三种不同的方向对准空间中一点，通过示教软件的“取点”按钮记录，点击“生成”，得到工具坐标系的参数。



“三点法 TCP”界面操作步骤：

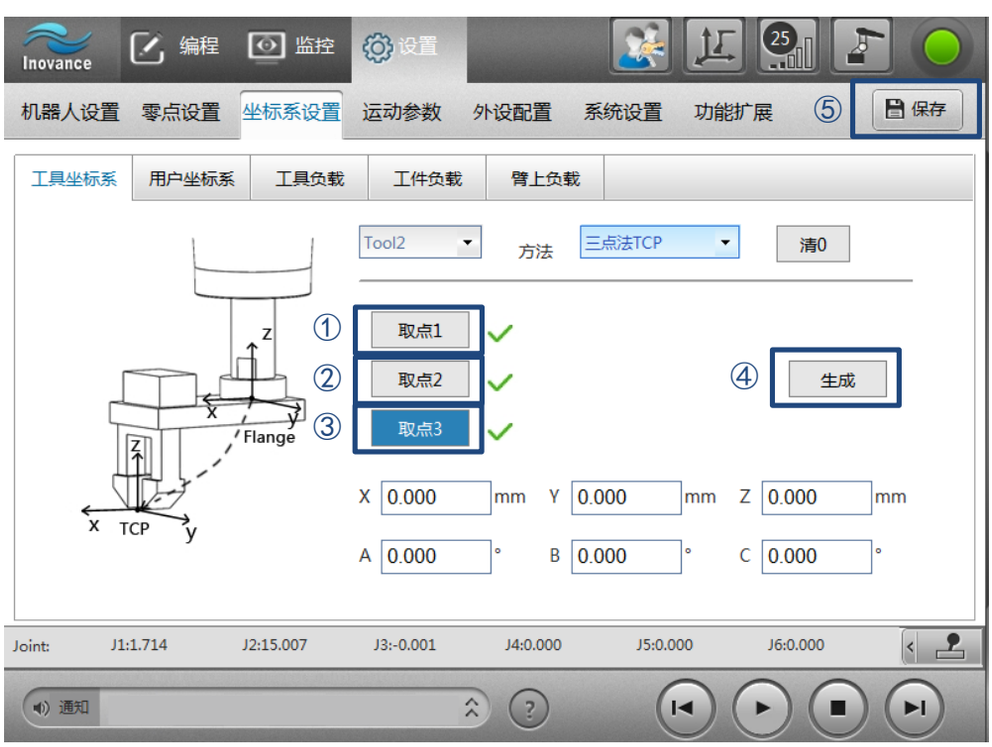
步骤 1：控制机器人运动，使 TCP 从第一种方向对准参照点，并点击“取点 1”；

步骤 2：控制机器人运动，使 TCP 从第二种方向对准参照点，并点击“取点 2”；

步骤 3：控制机器人运动，使 TCP 从第三种方向对准参照点，并点击“取点 3”；

步骤 4：点击“生成”，工具坐标系参数自动生成；

步骤 5：点击“保存”。

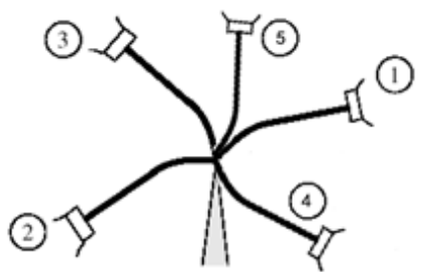


NOTE

- 三点法所选取的三种不同姿态应尽量间隔较大，推荐相隔 20°以上。五点法同。
- 点击生成后，会显示误差参数，当“最大误差小于 < 机器人本体绝对精度 +0.1mm”为比较准确。当超过太多时，建议重新取点生成。
- SCARA 和 Delta 机器人的工具末端与末端轴同轴线（即只有 Z 向有值）时，无法通过三点法 TCP 求得 Z 向位置参数。五点法同。

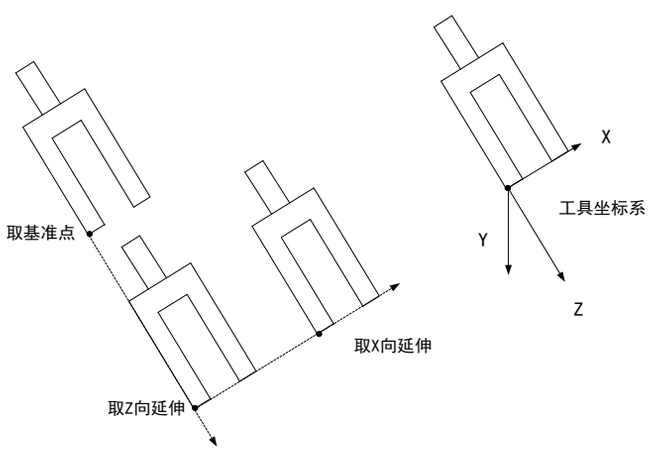
3） 五点法 TCP

与“三点法 TCP”类似，区别是：TCP 以五种不同的方向对准空间中一点。



4） 三点法 TCP+ZX

在三点法 TCP 的基础上，另外取三个点，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：



“三点法 TCP+ZX”界面操作步骤：

步骤 1：控制机器人运动，使 TCP 从第一种方向对准参照点，并点击“取点 1”；

步骤 2：控制机器人运动，使 TCP 从第二种方向对准参照点，并点击“取点 2”；

步骤 3：控制机器人运动，使 TCP 从第三种方向对准参照点，并点击“取点 3”；

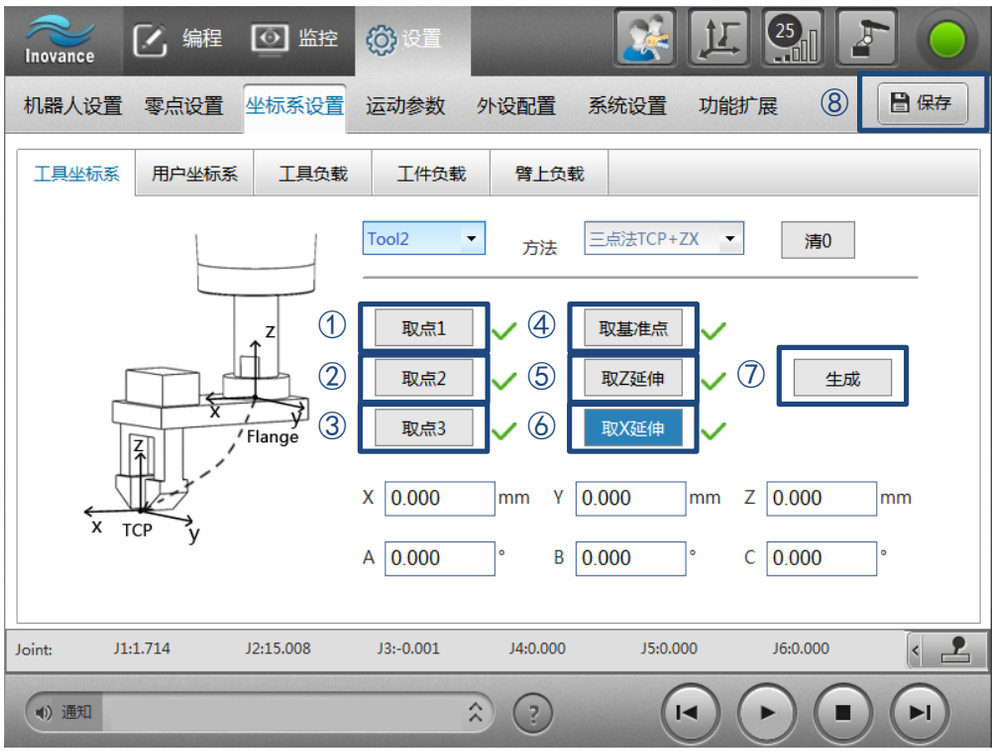
步骤 4：控制机器人运动，选取任一时刻 TCP 作为基准点，点击“取基准点”；

步骤 5：控制机器人运动，使工具 Z 向延伸一段距离，点击“取 Z 延伸”；

步骤 6：控制机器人运动，使工具 X 向延伸一段距离，点击“取 X 延伸”；

步骤 7：点击“生成”，工具坐标系参数自动生成；

步骤 8：点击“保存”。



5） 三点法 TCP+ZX

与“三点法 TCP+ZX”类似，区别是：TCP 以五种不同的方向对准空间中一点。

6） 锁螺丝四点法

仅适用于锁螺丝机型！

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，显示误差。

此外，页面会提示“是否运用标定结果修正机器人结构参数”，用户根据需求选择是否修正机器人结构参数。

注意事项：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。

NOTE

◆ 四个点必须位于水平面上，四个点应尽量间隔较大，推荐使用矩形的四个顶点。

2 用户坐标系

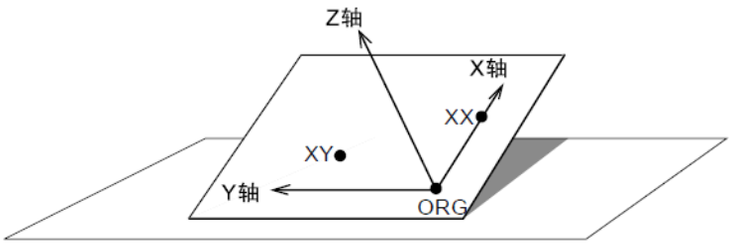
用户坐标系的参数设置是用户坐标相对于基坐标系的表达。

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	带转台，且用户坐标系位于转台中心

1） 直接法

直接输入参数即可。

2） 三点法 TCP



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

步骤 1：第一点取用户坐标系的原点，点击“取点一”第一点取用户坐标系的原点，点击“取点一”；

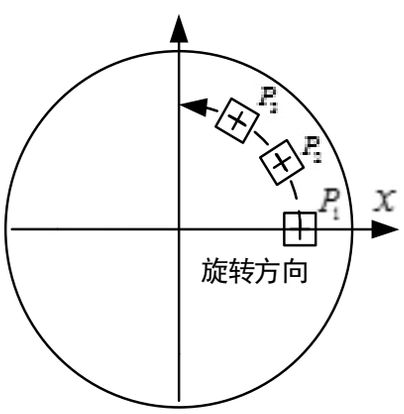
步骤 2：第二点取用户坐标系 X 轴正方向上一点，点击“取点二”；

步骤 3：第三点取用户坐标系 XY 平面上 Y+ 方向的一点，点击“取点三”；

步骤 4：点击“生成”，用户坐标系参数自动生成；

步骤 5：点击“保存”。

3） 旋转法



步骤 1：在转盘上标记一个固定点，使用 TCP 对准该标记点后，点击“取点 1”；

步骤 2：旋转一个角度，再次使用 TCP 对准该标记点后，点击“取点 2”；

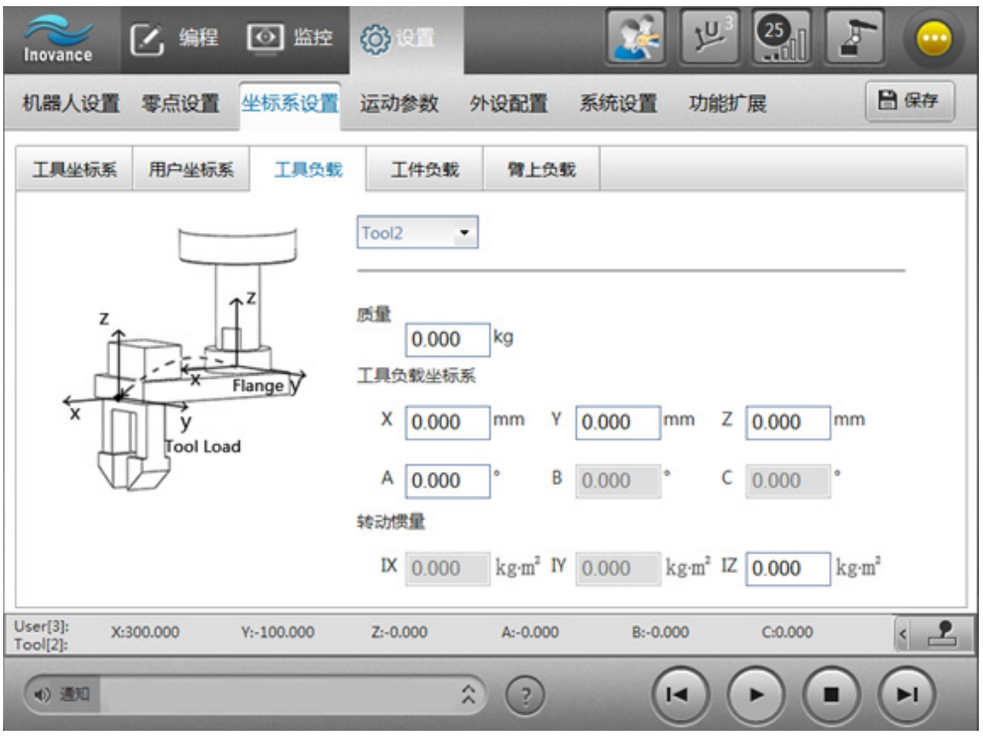
步骤 3：再次旋转一个任意角度，使用工具对准该标记点后，点击“取点 3”；

步骤 4：点击“生成”，用户坐标系参数自动生成；

步骤 5：点击“保存”。

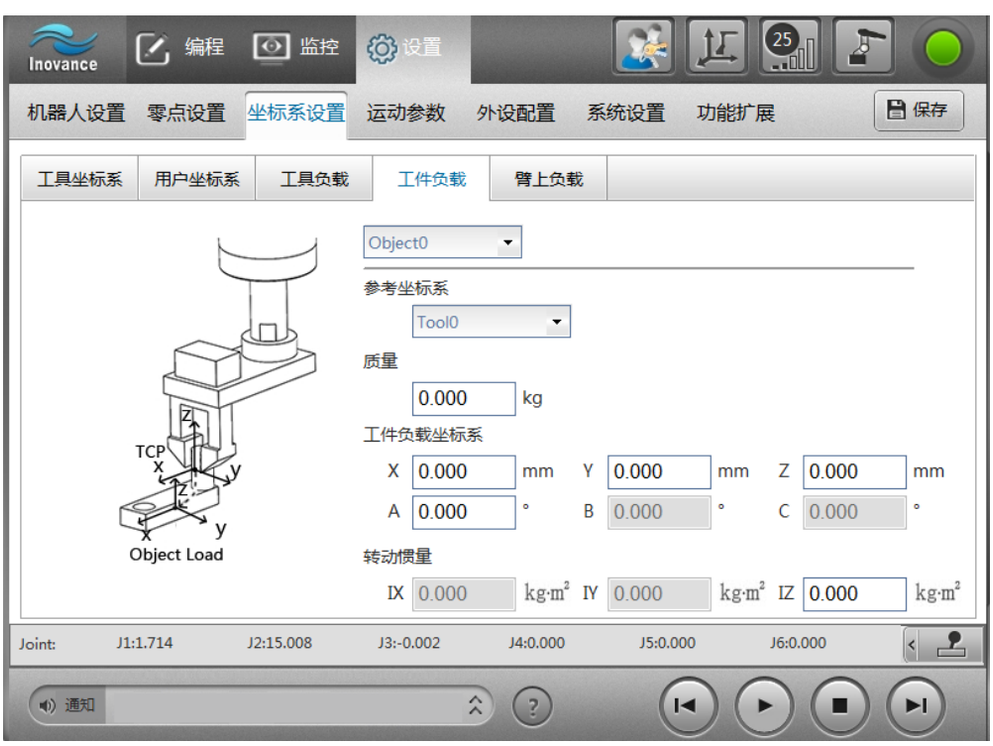
3 工具负载

配置工具负载，包含设置工具的质量、负载坐标系、转动惯量。



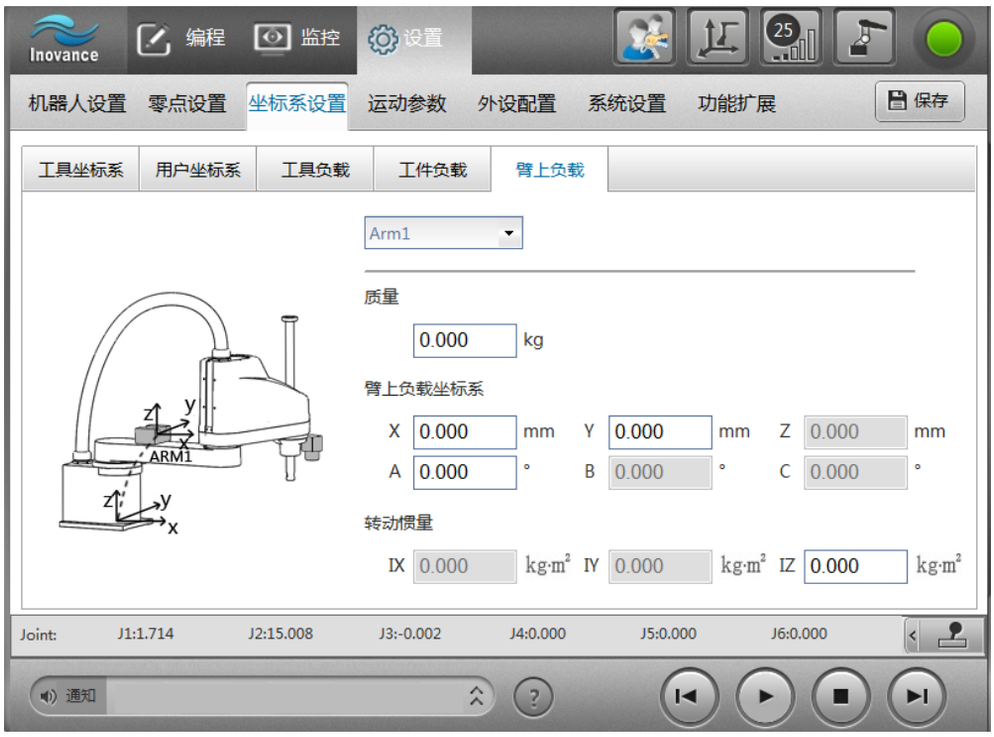
4 工件负载

配置手持工件负载，包含设置工件的质量、负载坐标系、转动惯量。



5 壁上负载

配置臂上负载，包含臂上负载的质量、负载坐标系、转动惯量。



2.4.4 运动设置

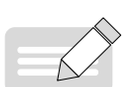
1 示教参数设置

寸动：这里设置自定义的寸动步长，在选用  时适用。

示教速度：示教过程中的最大速度。

示教实际速度 = 示教速度 X 工具栏设定的速度百分比。

示教加速度：示教过程中的最大加速度。



◆ “示教参数”是指操作机器人在各坐标系下移动的参数。程序单步运行、连续运行或再现运行时使用的参数是：运行速度、加速度、停止速度！

2 运行参数设置

运行速度：运行过程中的最大速度。

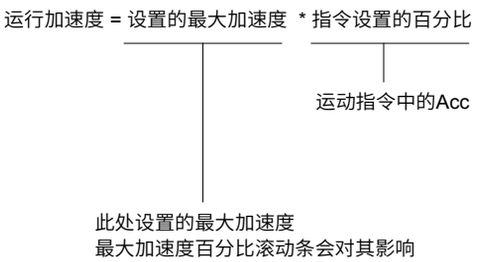
运行实际速度由下式计算得到：

运行速度 = 设置的最大速度 * 全局速度百分比 * 指令设置的百分比

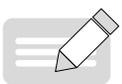
- 指令设置的百分比：运动指令中的V；使用Velset value影响此项；使用Velset OFF关闭影响
- 全局速度百分比：工具栏设置的速度百分比；使用Velset Rate[value]影响此项
- 设置的最大速度：此处设置的最大速度；最大速度百分比滚动条会对其影响

运行加速度：运行过程中的最大加速度。

运行实际加速度由下式计算得到：



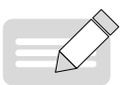
停止减速度：暂停或停止时的减速度。


NOTE

◆ “运行速度”、“运行加速度”和“停止减速度”参数的设置受伺服系统性能的制约。

过渡精度：相邻两条运动指令之间的过渡单位长度。

圆弧插补：支持关节插补（姿态变化不与圆心角相关）和姿态插补（姿态变化与圆心角相关）。默认“关节插补”。


NOTE

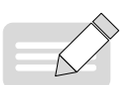
◆ 圆弧姿态插补类型只针对六轴机器人有效。

3 轴参数设置

轴极限：各轴的极限位置。出于安全考虑，请务必将轴极限设置在机械挡块范围以内。

跟随误差：由机器人的指令加速度和伺服的刚性等级决定。当出现跟随误差过大报警时，应适当降低加速度或增大跟随误差设定值。

到位误差：决定机器人规划完成与伺服实际到位的判定条件。


NOTE

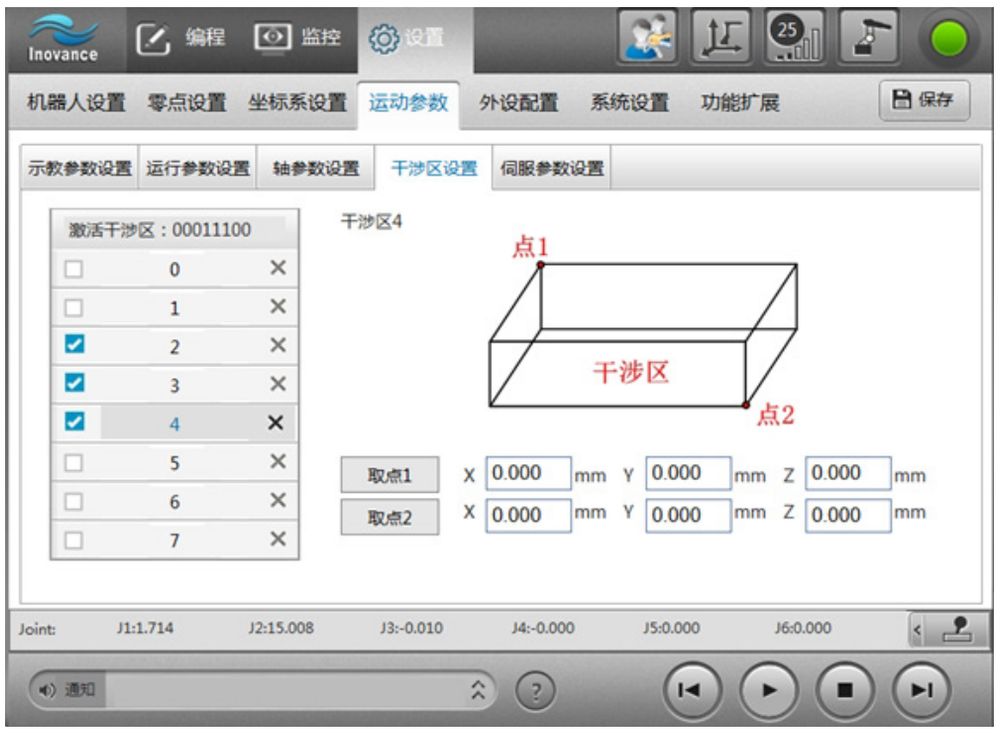
◆ “到位误差”和“跟随误差”参数需要在厂家模式下才能调整。

4 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，机器人工具末端进入干涉区会产生报警。可同时激活多个干涉区。

编辑干涉区：选中数字，右侧显示干涉区值，编辑干涉区。

激活干涉区：勾选干涉区前面的复选框，激活干涉区。



2.4.5 外设配置

1 IRLink 设置

IRLink 是为 IMC100 系列的扩展产品，用于控制管理 IO。IRLink 配置需要示教器具有 IRlink 配置权。

NOTE

- IRLink 配置权说明：默认 IRLink 配置权在示教器上。
- 当使用 InoRobShop 配置过 IRLink，则 IRLink 配置权转移到 InoRobShop 上，此时使用示教器配置 IRLink 将出现报警。如需再次使用示教器配置 IRLink，需通过 InoRobShop 下载一个空的 IRLink 配置到控制器中。

■ IRlink 配置规格

对于 IRCB10 系列控制柜，只能通过整机扩展模块的形式增加 IO，存在功耗规格限制和资源规格限制。

对于 IRCB300 系列控制柜，可通过 “扩展卡 + 整机扩展模块” 的形式增加 IO。对于扩展卡，支持最多 4 张扩展卡，不存在功耗规格，但存在资源规格限制；对于整机扩展模块，存在功耗规格限制和资源规格限制。

1） 功耗规格

整机扩展模块的硬件级联需遵循以下规则：每个 RTU 后可提供的功率为 15W，下表为整机扩展模块功耗：

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

在配置时需要保证单个 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，再在其后串联模块。

2） 资源规格

软件支持的资源总数是有限制的，资源限制如下表：

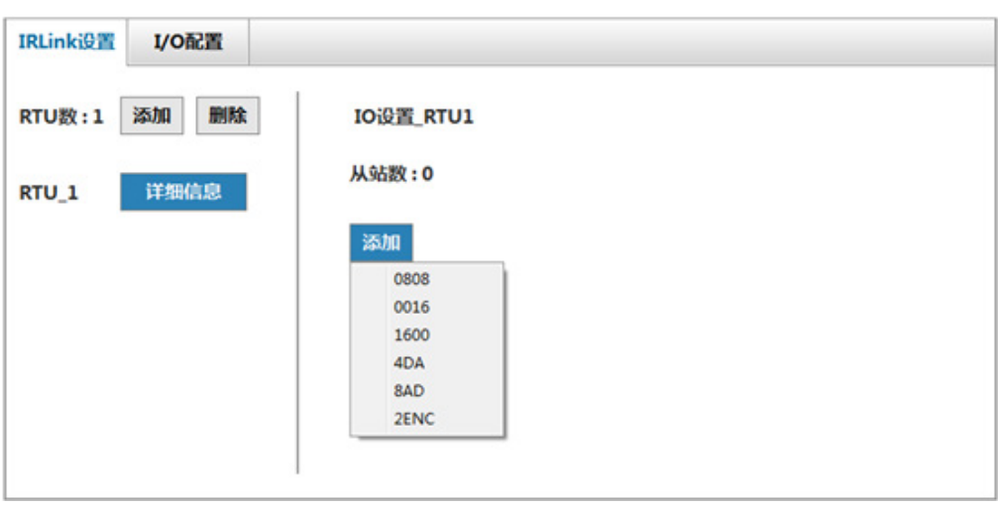
资源	软件支持最大资源数	关联模块类型	单个关联模块占用资源数
输入 IO	64 通道	IMC100-1600-END	16 通道
		IMC100-0808-ETND	8 通道
		IRCB-1600END-BD	16 通道
输出 IO	64 通道	IMC100-0808-ETND	8 通道
		IMC100-0016-ETPD	16 通道
		IMC100-0016-ETND	16 通道
		IRCB-0016ETND-BD	16 通道
		IRCB-0016ETPD-BD	16 通道
DA	16 通道	IMC100-4DA	4 通道
		IRCB-4DA-BD	4 通道
AD	16 通道	IMC100-8AD	8 通道
		IRCB-4AD-BD	4 通道
编码器	4 通道	IMC100-2ENID	2 通道
		IRCB-2EN1D-BD	2 通道

如：系统支持的扩展卡和整机扩展模块最多支持 2 个 IMC100-8AD 模块，4 个 IMC100-4DA 模块，4 个 IMC100-2ENID 模块，8 个 IMC100-0808-ETND 模块（或 4 个 IMC100-0016-ETND + 4 个 IRCB-1600END-BD 模块），以此类推。

■ 配置 IRLink 方法（软件层面）

连接好硬件设备后，按下述步骤配置 IRLink：

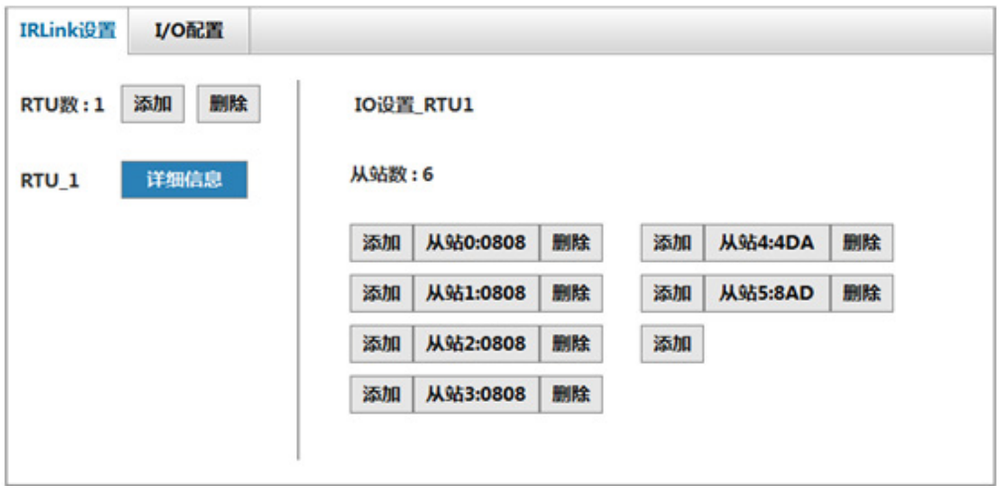
步骤 1：点击页面左侧的“添加”按钮，会自动产生一个 RTU。该 RTU 的详细信息会在右侧显示。最多添加五个扩展模块 RTU；



步骤 2：点击页面右侧的“添加”按钮，会弹出选项框，可选：0808、0016、1600、4DA、8AD、4AD、2ENC，对应的模块类型如下表所示：

选型	对应扩展模块类型	对应扩展卡类型
0808	IMC100-0808-ETND	\
1600	IMC100-1600-END	IRCB-1600END-BD
0016	IMC100-0016-ETPD 或 IMC100-0016-ETND	IRCB-0016ETND-BD 或 IRCB-0016ETPD-BD
4DA	IMC100-4DA	IRCB-4DA-BD
8AD	IMC100-8AD	\
4AD	\	IRCB-4AD-BD
2ENC	IMC100-2ENID	IRCB-2EN1D-BD

例如，依次添加 4 个 0808，1 个 4DA，1 个 8AD，结果如下图所示：



步骤 3：配置模块参数。

对于 8AD、4DA、2ENC 三种模块，单击后进入配置页面。

1） 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）

需配置“输入范围”和“过采样率”（所有通道配置相同）。

输入范围：输入范围有 2 个档位。

输入电压 -5V~5V 时，输入电流范围 0~20mA；

输入电压 -10V~10V 时，输入电流范围 0~40mA。

过采样率：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。

2） 4DA（4 通道电压或电流模拟量转换输出扩展模块）

设置四个通道输出模拟量的范围。

3） 2ENC（2 通道差分输入增量编码器采集扩展模块）

可设置通道是否翻转、信号滤波时间。滤波时间越长，额定信号输入频率降低。

滤波时间 = 采样深度 X 采样时间。

2.4.6 系统设置

1 通讯设置

a）示教器通讯

示教器有两种通讯方式，对应的控制器上的网口 EtherNet1 与 EtherNet2。EtherNet1 为默认为动态 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。

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”栏后连接。

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 即可连接。

NOTE

◆ 关于 EtherNet1 与 EtherNet2：对于 IRCB300 系列控制柜，LAN 对应 EtherNet1，PC 和示教器插槽对应 EtherNet2。

b）控制器 EtherNet1 设置

动态 IP 开关：EtherNet1 的动态 IP 开关默认是开启的，表示 EtherNet1 是动态端口。

EtherNet1 设为静态端口方法：通过 EtherNet2 直连（比如通过手持示教器），取消 EtherNet1 动态 IP 开关选项，并定义 IP 地址。

NOTE

◆ EtherNet1 设为静态端口的应用：当机器人控制器同时连接示教器和外部设备时，示教器与 EtherNet2 口连接，外部设备与 EtherNet1 口连接，此时关闭“动态 IP 开关”，将 EtherNet1 口设置为静态 IP，外部设备就能与控制器正常通讯。

NOTE

- 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 开关调整后需要重启控制器生效！

客户端与服务器：与外部设备通讯时，机器人可选择作为客户端或服务器。这一功能多用于视觉，指定视觉功能中机器人的角色是作为客户端还是服务器。

客户端：机器人控制器作为客户端，外部设备作为服务器。需要在视觉编程应用中，需要在程序中利用 Open Socket 指令选择连接的 IP。

服务器：机器人控制器作为服务器，最多支持与 5 个客户端连接。

设置时，需指明一个的端口号，端口号范围 1024~9999，其中 3333 不可用。设置完成后重启机器人系统，服务器自动打开；此时作为客户端的外部设备必须在机器人程序运行之前打开，否则会有报警提示。

2 时间日期

显示控制器中的时间，通过“设置时间”按钮调整。

3 用户设置

根据当前使用者的角色，选择模式并登录：

模式	面向对象	初始默认密码
用户权限	面向生产线上负责运行、监控机器人动作的操作者。可以直接控制机器人移动，运行机器人程序。	无需密码
编辑模式	面向示教编程者。比用户模式多了编辑程序权限，有控制设备、机械锁定权限。	初始密码 000000
管理模式	面向系统维护人员。除了上述操作权限外，还包含大多数系统的操作。	初始密码 000000
厂家模式	面向我司的技术人员。拥有最高权限，可以做升级等操作。	厂家保留

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
控制移动	√	√	√	√
运行程序	√	√	√	√
修改、编辑程序	×	√	√	√

操作内容	使用权限			
	用户模式	编辑模式	管理模式	厂家模式
机器人设置	×	×	×	√
工作原点	×	√	√	√
绝对零点	×	×	√	√
回零修复	×	×	√	√
坐标系设置	仅支持选用，不可修改	√	√	√
寸动	×	×	√	√
示教或运行的最大速度 / 加速度 / 减速度	×	×	×	√
过渡特性	×	×	√	√
轴极限	×	×	√	√
干涉区	仅支持选用，不可修改	√	√	√
外设配置	×	×	√	√
通讯设置	仅示教器通讯	√	√	√
时间日期	×	×	√	√
语言选择	√	√	√	√
自定义报警	×	√	√	√
多任务配置	×	√	√	√
屏幕校准	×	×	√	√
屏幕翻转	×	×	√	√
亮度及屏保	×	×	√	√
摇杆校准	×	×	√	√
配置文件备份	×	×	√	√
配置文件加载	×	×	√	√
程序备份	×	√	√	√
程序加载	×	√	√	√
点文件加载	×	√	√	√
系统升级	×	×	×	√
恢复出厂设置	×	×	×	√
SD 卡格式化	×	×	√	√
清除历史报警	×	×	√	√
网络调试	√	√	√	√
控制器调试	×	×	√	√
错误信息保存	×	×	√	√
错误信息导出	×	×	√	√
控制设备	×	√	√	√
机械锁定	×	√	√	√
功能扩展	×	×	√	√

4 语言

可中英文切换，需要重启生效！

5 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。

在编程时，可利用 Alarm 指令调出自定义报警。自定义报警会在报警日志中同步记录。

6 多任务配置

PLC[0]、PLC[1] 为设置型任务，详见 *《汇川机器人编程手册》* 多任务相关章节。

激活：勾选后启用此任务；

静态启动：勾选后机器人上电后即启动程序，不勾选表示机器人上电时不会立即启动，与主任务（任务 0）一起，受“启动”、“停止”、“暂停”按钮控制，

默认程序：在这里设置的默认程序，机器人控制器开机后，在再现模式默认打开该程序。

7 其他设置

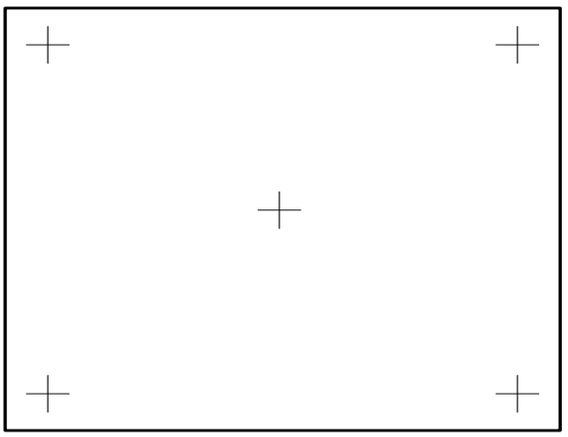
其他设置包括“示教器设置”、“备份与加载”、“系统更新”、“系统还原”、“其它”。

其中“示教器设置”仅针对手持示教器生效，PC 版无效。

a）示教器设置

■ 屏幕校准

屏幕触摸不准确时，可使用此功能校准屏幕。使用笔尖点击依次出现的“+”中心点（“+”依次出现在屏幕中央和四角位置），最后再次点击任一空白处，即完成设置。



■ **屏幕翻转**

默认的示教器显示都是适用于左手持屏。对于 ITP100 示教器，可以通过屏幕翻转功能，适用于右手持屏。

■ **亮度及屏保**

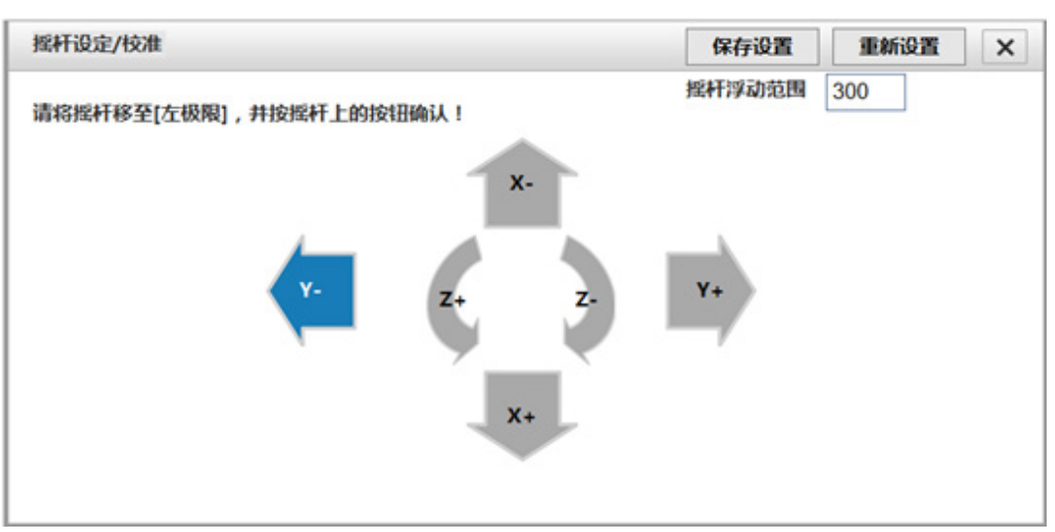
亮度：共有 1~6 六个亮度等级，可自由选择亮度。

屏保：可选择性激活屏保功能，并设置屏保时间。

■ **摇杆校准**

用于厂家维护人员校准摇杆方向，需要在厂家模式下设置。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，按提示的方向和顺序依次操纵摇杆即可。

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



b）备份与加载

■　配置文件备份

配置文件：机器人相关的设置参数（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），都被保存在 robotcfg.cfg.bk 文件中。

配置文件备份方法：在控制器 / 控制柜上插入 U 盘，点击“配置文件备份”按钮 ，将机器人配置文件备份到 U 盘中根目录下。

◆ 操作前，在控制器 / 控制柜插入 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。在接下来操作过程中保持 U 盘通讯良好。

■　配置文件加载

配置文件加载方法：提前在 U 盘根目录下准备好配置文件 robotcfg.cfg.bk ，插入到控制器 / 控制柜上，点击“配置文件加载”按钮，将机器人配置文件加载到控制器中。完成后，重新给控制器上电生效。

◆ 操作前，在控制器 / 控制柜插入 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。在接下来操作过程中保持 U 盘通讯良好。

◆ 禁止不同机型的配置文件加载。

■　程序备份

程序备份加载内容：

	14 版本前（含）	15 版本后（含）
程序备份	1.TeachProgram 2.PalletInfo	1.TeachProgram 2.PalletInfo 3.TecParameter
程序加载	1.TeachProgram 2.PalletInfo	1. TeachProgram 2.PalletInfo 3.TecParameter

TeachProgram：程序文件的文件夹，内含所有“.pro”程序文件。

PalletInfo：托盘文件的文件夹，内含码垛、托盘信息。当使用托盘变量时，需要使用它。

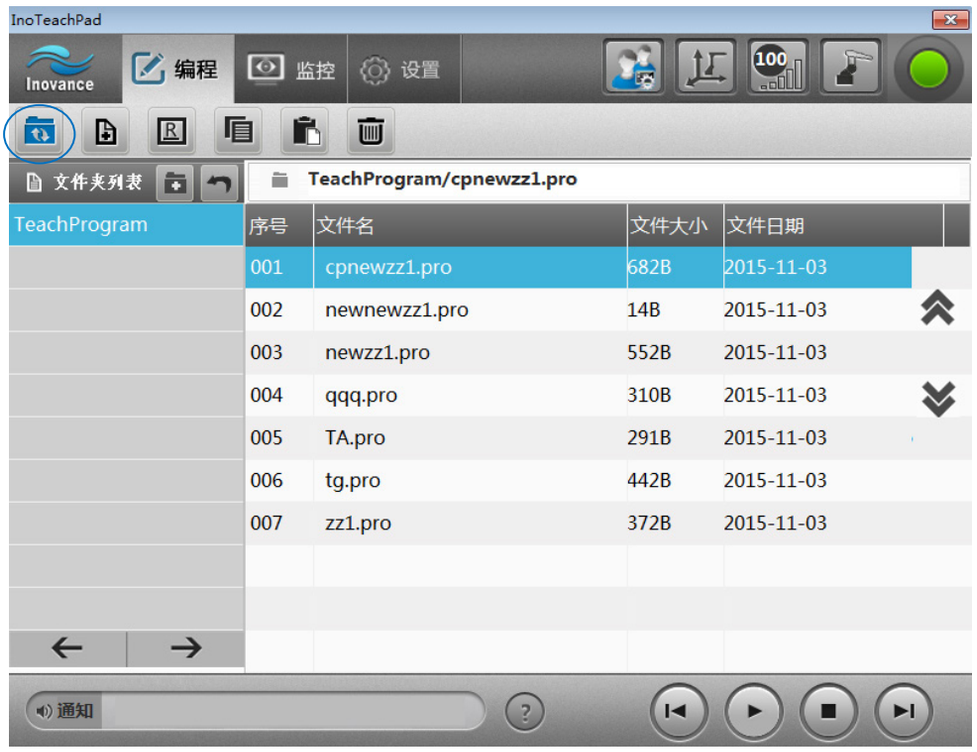
TecParameter：工艺文件夹，内含锁螺丝、点胶工艺的信息。

程序备份方法：在控制器 / 控制柜上插入 U 盘，点击“程序备份”按钮，将 SD 卡中的程序备份到 U 盘根目录下。

◆ 操作前，在控制器 / 控制柜上插入 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。在接下来操作过程中保持 SD 卡、U 盘通讯良好。

■　程序加载

程序加载方法：提前在 U 盘根目录下准备好相应文件，再插入到控制器 / 控制柜上，点击“程序加载”按钮，将 U 盘中的程序加载到 SD 卡中。加载完成后，对于程序而言，进入编程界面，需点击左上角的“刷新”按钮，刷新显示。



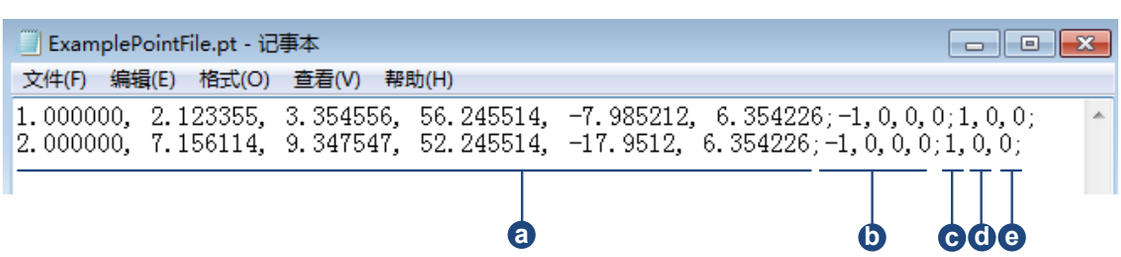
NOTE

◆ 操作前，在控制器 / 控制柜上插入 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。在接下来操作过程中保持 SD 卡、U 盘通讯良好。

■ 加载点文件

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。

示例如下图：a) 为机器人坐标系值；b）为臂参数；c）为坐标系号；d）为工具号；e）为用户号。



点文件加载方法：提前在 U 盘上准备好点文件，在示教器上插入 U 盘，浏览目标点文件，将其加载到机器人控制系统中。

c）系统

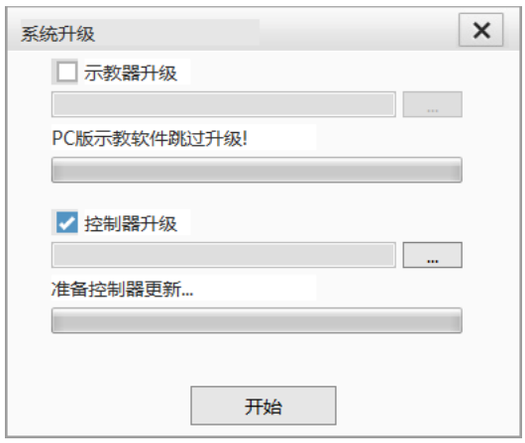
■ 系统升级

“系统升级”步骤：

步骤 1：勾选项目，选择进行哪些升级（可同时勾选，支持同时升级示教器与控制器）；

步骤 2：通过浏览按钮选择新版本的软件包。；

步骤 3：选择完成后，点击“开始”即可进行升级。



NOTE

- PC 版本示教不可进行升级，示教器升级只针对手持示教器！
- 控制器升级过程中请勿断电，否则可能造成异常，只有通过对控制器进行刷机才可恢复！

■ 恢复出厂设置

将机器人所有的设置参数全部初始化为出厂状态。注意：请在厂家指导下操作。

■ SD 卡格式化

将控制器上的 SD 卡格式化成适合机器人使用的程序存储卡。该操作会清空 SD 卡内的程序，建议操作前先备份程序。

NOTE

- 操作前，在控制器 / 控制柜插入 SD 卡，检查连接状态。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。

■ 清除历史报警

清除监控中的日志和报警记录。

d）调试

■ 网络调试

调试示教器、控制器与其他设备的通讯，相当于 Ping 功能。

填入 IP 地址，点击“Ping”按钮即可。

NOTE

- 示教器连接控制器时，为控制器 Ping 功能，用于检测控制器与其他设备网络连接情况。
- 示教器未连接控制器时，为示教器 Ping 功能，用于检测示教器与其他设备的网络连接情况。

■ 控制器调试

控制器在运行过程中，能输出并记录过程信息，监控控制器的状态与流程，以便相关人员在线查看和事后分析。

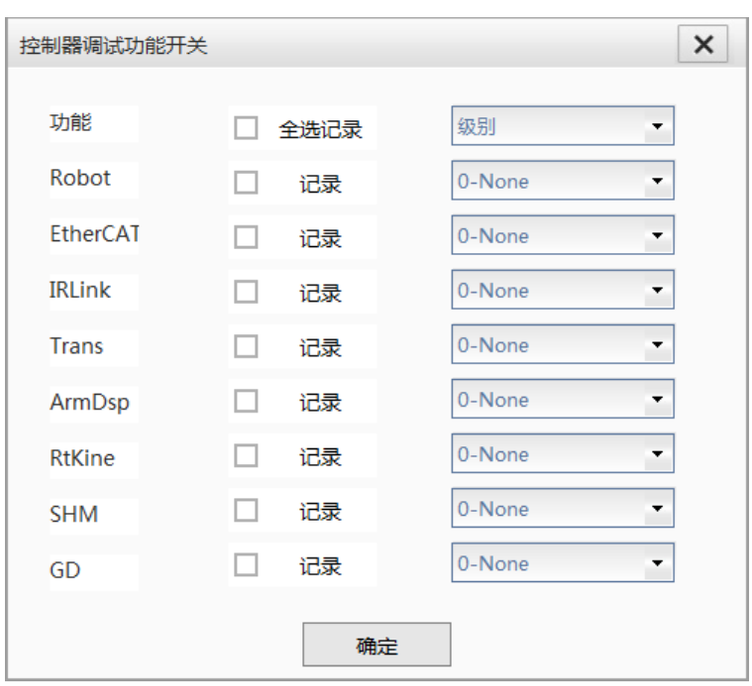
调试对象：机器人系统包含多个模块，每个模块包含的调试信息记录的程度不同。为了管理的方便，调试信息可以选择不同模块输出，目前控制器有以下模块可进行调试输出：

名称	说明
Robot	核心调度模块
EtherCAT	EtherCAT 通信模块
IRLink	IRLink 通信模块
Trans	语言解释器模块
ArmDsp	ARM 与 DSP 交互模块
RtKine	运动学模块
shM	共享内存模块
GD	配置信息模块

调试级别：对于每个模块，都可选择记录的级别，随着级别序号的升高，记录的内容项越多，内容也越详细。

名称	说明
0-None	不记录
1-Alert	只记录最严重的错误
2-Critical	记录严重的错误
3-Error	记录错误级事件
4-Warning	记录警告级事件
5-Notice	记录注意型事件
6-Information	记录一般事件
7-Debug	记录所有详细信息

“控制器调试”步骤：点击“控制器调试”按钮，进入调试级别选择页面。在使用时，根据需要勾选对象，并选择对应调试级别。确认后立即生效。



1） 在线调试：通过控制器的串口或 TCP/IP 端口打印信息。

串口：通过串口连接控制器，需要注意控制器和控制柜上串口是 485，需准备 232 到 485 的转接头。

TCP/IP：通过其他设备（如 PC）以 TCP/IP 协议连接控制器，端口号 5555。

2） 调试信息的保存、导出：

系统将自动记录近期的调试信息，通过【错误信息保存】-【错误信息导出】导出到 USB 上。

■ **错误信息保存**

将控制器的错误信息保存到 SD 卡中。厂家模式下使用。

■ **错误信息导出**

将 SD 卡中的错误信息导出到控制器的 USB 。厂家模式下使用。

e）其他

■ **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，此时不能通过示教器来控制机器人（包括修改参数、运行程序等内容），此时示教器只能起到监控的作用。

当控制权在“远程 IO 单元”或“远程 Modbus 单元”时，需要在启动速度栏设置程序第一次运行的启动速度百分比。通过外部 IO 修改速度（如远程 IO 模式下的速度加、速度减）后，该值失效，机器人以设置的速度运行。通常启动速度百分比低于 100，确保第一次慢速运行。

■ **机械锁定**

在机械锁定模式下，机器人将不再运动。

2.5 监控

利用监控页面可对各类变量以及机器人各种状态属性进行监视。

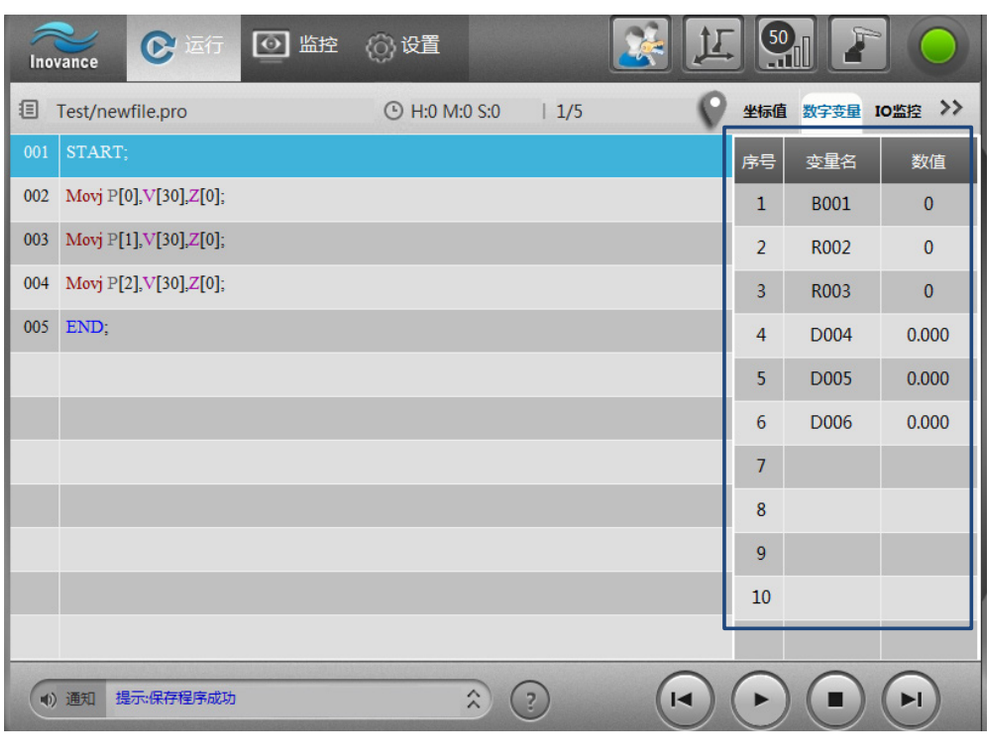
2.5.1 监控页面基本操作介绍

翻页：列表头为蓝色时，表明该列表被选中，可通过右侧按钮 和 对当前列表翻页。

修改：双击对象直接修改。

收藏：对于全局数值变量和 IO，利用变量名前的复选框，可以收藏 / 取消收藏。右侧的按钮  用于切换全部显示或仅显示收藏项目。

收藏的前十项会显示在运行界面的快捷监控面板上。



2.5.2 全局变量监控

1 全局数值变量

双击变量弹后出如下窗口，修改变量的值。

2 全局平移变量

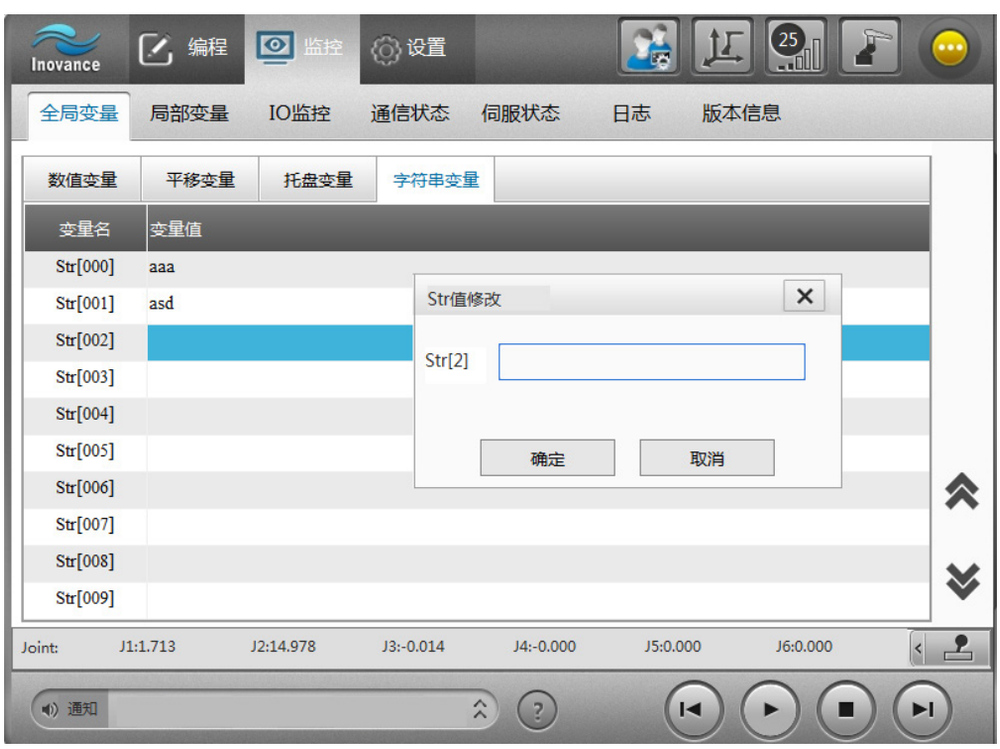
双击修改平移变量。可直接填入值，或通过取两点计算所得。

3 全局托盘变量

全局托盘变量是基于码垛工艺的专用托盘变量，参考《汇川机器设计应用与维护手册 - 应用篇 2: 高级功能应用》中“码垛工艺”相关内容。

4 全局字符串变量

双击变量弹后出如下窗口，可修改变量的值。



2.5.3 局部变量监控

1 局部数值变量

双击变量弹后出如下窗口，可修改变量的值。

2 位置变量

位置变量监控界面默认情况下不显示臂参数，如需观察臂参数，双击某行弹出详细信息界面即可。

位置变量可双击进行修改，其他操作介绍如下：

a）替换：选中位置变量，点击“替换”按钮，位置变量的值被机器人当前位置替换。

b）添加：点击“添加”按钮，添加一个新的位置变量，变量值为机器人当前位置。

c）重命名：重命名位置变量。

d）删除：删除位置变量。

e）保存：位置变量编辑完成后，需要点击“保存”才会保存。

f）快速定位：输入变量号，快速定位到相应位置。

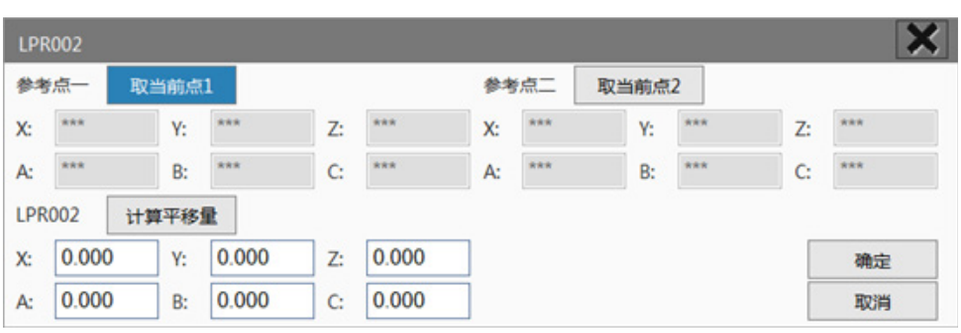
g）运行到位置变量：选中该行变量，点击下方的“运行”按钮即可运行到位置变量处。

NOTE

◆ 机器人暂停或停止时，监控窗口显示的位置变量才会刷新。

3 局部平移变量

双击修改平移变量。既可直接填入值，也可通过取两点计算所得。



4 局部托盘变量

局部托盘变量只能通过 LPallet 指令编辑，此处只起到监控显示的作用。局部托盘变量只在当前程序中有效，进入新的程序，上一个程序文件里的局部托盘变量值会重置为空。

数值变量 | 位置变量 | 平移变量 | 托盘变量

托盘序号	行数	列数	层数	层高
LPallet[000]	--	--	--	--
LPallet[001]	--	--	--	--
LPallet[002]	--	--	--	--
LPallet[003]	--	--	--	--
LPallet[004]	--	--	--	--
LPallet[005]	--	--	--	--
LPallet[006]	--	--	--	--
LPallet[007]	--	--	--	--
LPallet[008]	--	--	--	--
LPallet[009]	--	--	--	--

2.5.4 IO 监控

NOTE

◆ 在使用 IO 监控前，请保证 IRLink 配置正确！

对于 IRCB10 系列控制柜，IO 监控界面包含 IN/OUT、AD/DA 两个部分；对于 IRCB300 系列控制柜，IO 界面包含 IN/OUT、AD/DA、SysIO 三个部分。

1 IN/OUT

IN **强制开关：**默认情况下，输入信号的状态由外部源决定。当强制开关为“强制状态”时，IN 信号可被人为强制为 ON 或 OFF。

状态：直接点击对信号的状态，将信号置为相反状态。

NOTE

◆ 橙色标识说明：代表该 IO 被其它占用，不能被控制或使用。

IN 橙色——仅 IRCB10 系列控制柜会出现，IN[000]-IN[002] 为系统 IO，被占用。

OUT 橙色——输出端口被占用，控制权不为 RC_ACTIVE。

NOTE

◆ 控制权说明：

控制权名称	说明
RC_STATIC	代表 RC 系统占用。通过【外设配置】-【IO 配置】中将 Out 与系统功能绑定，此时输出端口信号只与功能相关，不能人为更改信号状态。
RC_ACTIVE	代表 RC 正常控制状态。此时信号能通过【IO 监控】中变更或通过 Set 指令变更控制。
PLC_ACTIVE	代表 PLC 控制状态，此时信号只受 InoRobShop 等 PLC 软件控制。

NOTE

◆ 当在【外设设置】-【IO 配置】中配置了 Out[***}，或在 InoRobshop 软件中将 Out[***} 控制权变为 PLC，此时 Out[***} 会变为橙色状态（占用状态）。

2 AD/DA

类型：模拟量信号是电流还是电压信号。

范围：模拟量信号的范围，不同 IRIInk 产品的模拟量端口有不同的选定范围。

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。单击列表中的状态项修改参数值。

开关：决定 DA 状态值是否有效。可手动切换。

3 Sys IO

■ IRCB10 **系列控制柜**

系统 IO 为 In[0]~In[2]，分别关联急停、使能、模式切换，显示在 IN/OUT 界面。

■ IRCB300 **系列控制柜**

系统 IO 有 16 输入和 16 输出，有单独的界面显示，输入 / 输出对应功能说明如下：

输入	功能	输出	功能
SysIn[0]	急停	SysOut[0]	系统运行指示灯
SysIn[1]	使能	SysOut [1]	系统报警指示灯
SysIn[2]	模式切换（示教 / 再现）	SysOut [2]	系统使能指示灯

输入	功能	输出	功能
SysIn[3]	-	SysOut [3]	EtherNet1 连接指示灯 亮：连接； 灭：未连接
SysIn[4]	-	SysOut [4]	EtherNet1 帧传递指示灯 闪烁：有数据传输； 常亮：连接但没有数据传输
SysIn[5]	安全门	SysOut [5]	EtherNet2 连接指示灯 亮：连接； 灭：未连接；
SysIn[6]	ITP100 使能	SysOut [6]	EtherNet2 帧传递指示灯 闪烁：有数据传输； 常亮：连接但没有数据传输
SysIn[7]	-	SysOut [7]	-
SysIn[8]	-	SysOut [8]	机器人本体使能灯
SysIn[9]	-	SysOut [9]	-
SysIn[10]	-	SysOut [10]	-
SysIn[11]	-	SysOut [11]	-
SysIn[12]	-	SysOut [12]	-
SysIn[13]	IO 板卡故障指示	SysOut [13]	-
SysIn[14]	IO 板卡故障指示	SysOut [14]	-
SysIn[15]	IO 板卡故障指示	SysOut [15]	-

2.5.5 通信状态

在通信状态界面中可查看控制器上各端口通信状态。

2.5.6 伺服状态

在伺服状态面板中能实时监控每个伺服参数。

2.5.7 日志

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*《汇川机器人设计应用与维护手册 - 附录：机器人报警及处理方法》*。

NOTE

◆ 橙如需修改 Out 的橙色状态（占用状态）时，可修改【外设配置】-【IO 配置】的配置或在 InoRobShop 软件修改控制权为非 PLC 控制状态。

2.5.8 版本信息

在版本信息界面可查看版本号。

NOTE

- 示教器与控制器版本不一致时，会产生报警。
- 在厂家模式下，能看到更多信息。

2.6 TCP/IP 通信

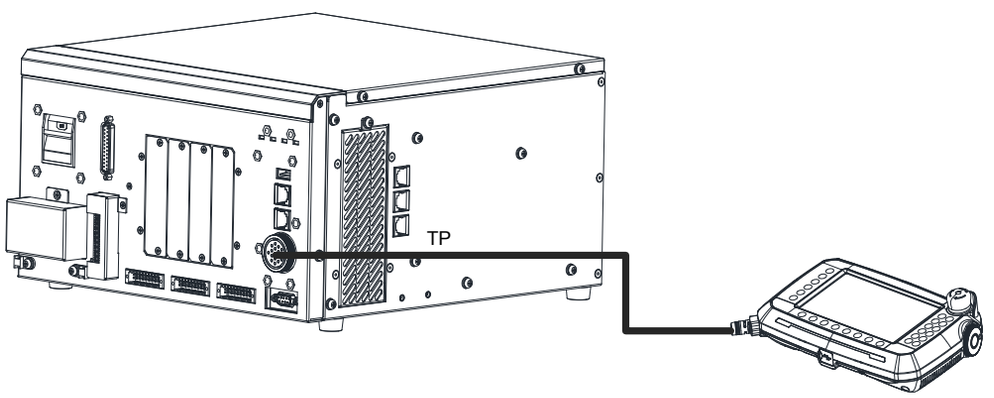
本节介绍手持示教器 /PC 版示教器与机器人控制器连接及通信相关功能。

NOTE

- 控制器在同一时刻只能与一个示教器连接。

2.6.1 手持示教器连接

手持示教器与控制柜的 TP 接口连接，默认能直接连接。如不能连接，请查看通讯设置中的示教器通讯 IP 地址，IP 设为 192.168.23.25。



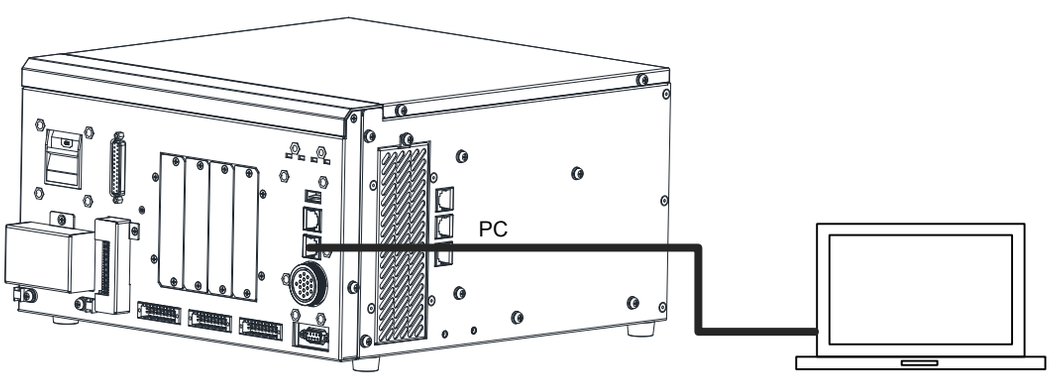
NOTE

- 手持示教器插到控制柜时，是连接到控制柜内部的机器人控制器的 EtherNet2 口。控制器 EtherNet2 IP 地址为 192.168.23.25，手持示教器 IP 地址为 192.168.23.100，因而两者能默认连上。

2.6.2 PC 版示教器直接连接

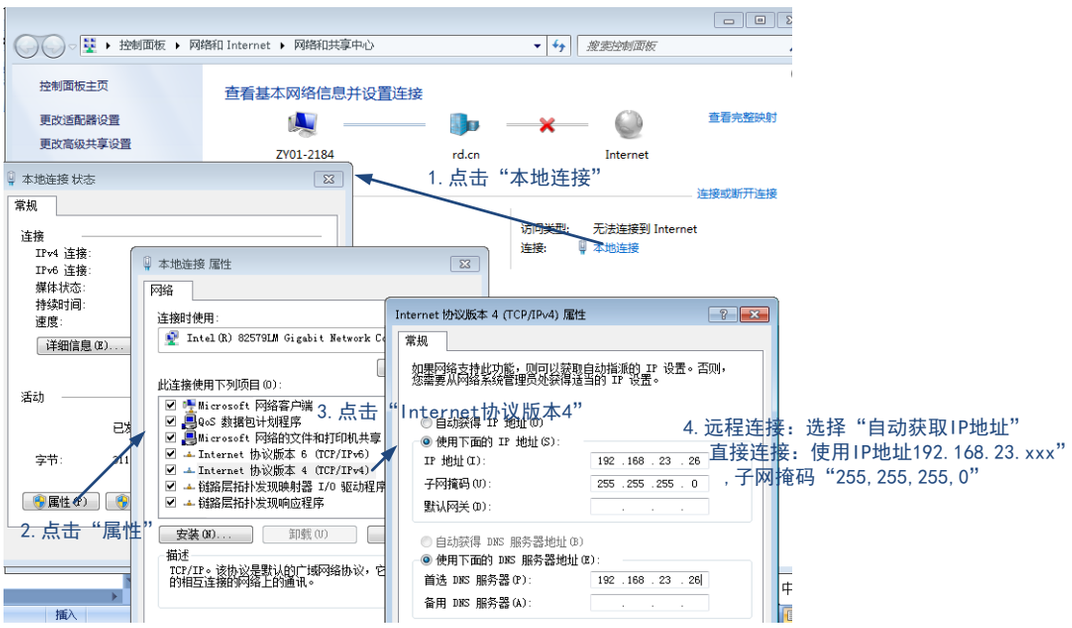
PC 版示教器位于个人计算机上，个人计算机与机器人控制器的 EtherNet2 直接连接。

对于 IRCB300，PC 版示教器与控制柜 PC 接口连接，如下图所示：



控制器 EtherNet2 IP 地址为 192.168.23.25，需设置个人计算机的 IP 与之匹配，设置步骤如下：

步骤 1：在网络和共享中心中，选取本地连接，配置计算机本地 IP 为 192.168.23.XXX;

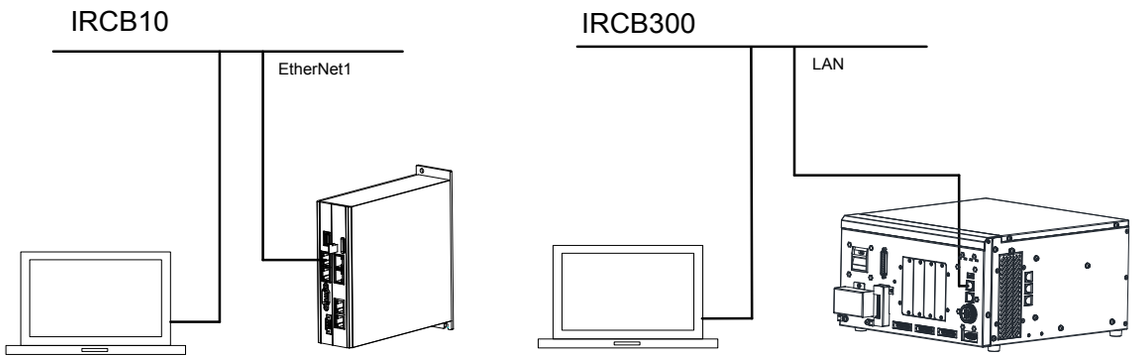


步骤 2：进入 PC 示教器应用程序，点击“跳过”，设置要连接的 IP 地址为 192.168.23.25。

2.6.3 PC 版示教器远程连接

应用场合 1：PC 示教器与控制器处于动态 IP 网络中。

PC 和控制器的 EtherNet1 接入动态 IP 网络。对于 IRCB300，则是控制柜 LAN 端口接入动态 IP 网络。



此时个人计算机的本地连接设为动态获取，打开 PC 版示教器应用程序，点击“跳过”，设置要连接的控制器 EtherNet1 的 IP。

NOTE

◆ 若不知道控制器 EtherNet1 的 IP，可通过直接连接方式将示教器连接控制器，在监控页面可以查询 EtherNet1 的 IP。

应用场合 2：PC 示教器与控制器处于静态 IP 网络中。

PC 和控制器的 EtherNet1 接入静态 IP 网络。对于 IRCB300，则是控制柜 LAN 端口接入静态 IP 网络。

此时，只要配置控制器的 EtherNet1 和个人计算机处于同一网关下（IP 地址前三位相同即可，比如都为 192.168.0.XXX）即可。

例：设置控制器 EtherNet1 的 IP 为 192.168.0.26；个人计算机的本地连接设为 192.168.0.27。

操作：

步骤 1：利用示教器直连连接控制器，在【通讯设置】EtherNet1 中取消掉勾选的动态 IP 开关，输入 192.168.0.26 并保存。

步骤 2：设置个人计算机的 IP 为 192.168.0.27

步骤 3：打开打开 PC 版示教器应用程序，点击“跳过”，设置要连接的控制器 EtherNet1 的 IP——192.168.23.27。

2.6.4 Ftp 文件传送功能

控制器加入 FTP（File TransferProtocol）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。

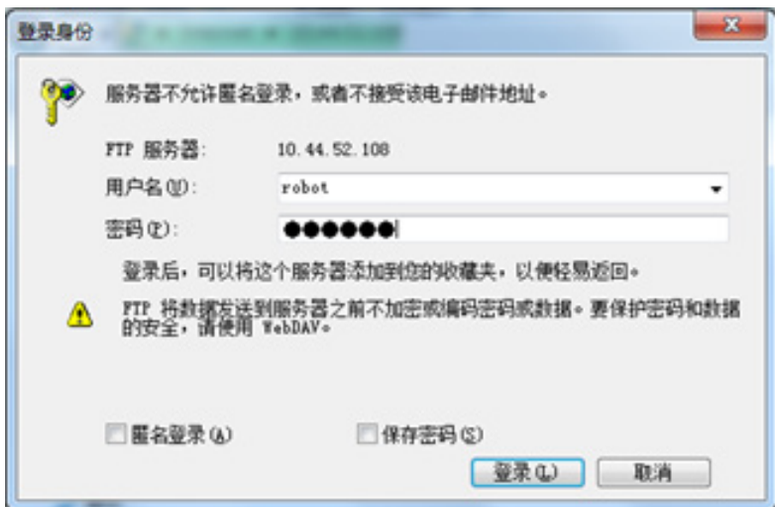
NOTE

◆ 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。

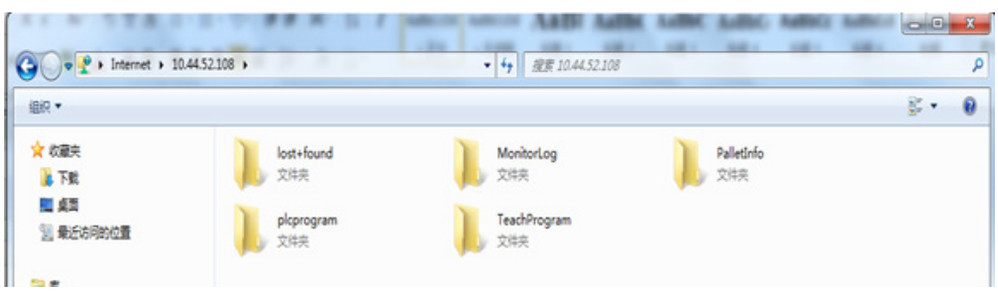
示例：控制器中取出机器人程序“A.pro”

步骤 1：假设某计算机与控制器通过网络连接，已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；

步骤 2：按回车键后，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）；



步骤 3：默认的目录为控制器上的 SD 卡目录：



步骤 4：从 TeachProgram 文件夹中，将 A.pro 文件复制出即可。

NOTE

◆ FTP 登入方式还可采用文件管理器方式，DOS 命令行方式等。

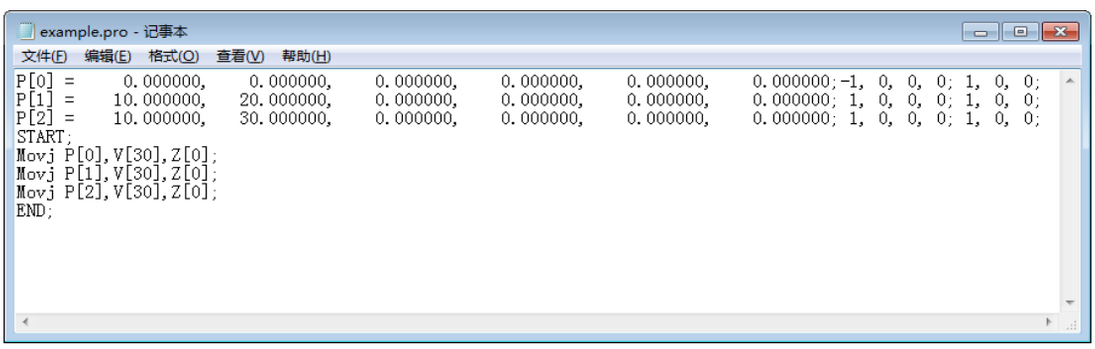
通过此篇，您可以了解我司机器人的编程的概念，学习对常见指令的编程方法。如果你需要了解更多指令，请参见*《汇川机器人设计应用与维护手册 - 附录：机器人指令集》*。

第 3 章 编程

3.1 机器人程序文件

机器人程序文件为后缀名为“.pro”的文件，存储在控制器的 SD 中 TeachProgram 目录下。文件中包含位置变量和指令。其中位置变量存储在前面，指令存储在后面。

一个典型的程序文件如下所示：



NOTE

◆ 在示教器上，程序只显示指令部分，位置变量显示在监控页面中。

NOTE

◆ **文件夹命名规范：**文件夹名称由字母、数字和下划线组成，首位必须为字母，长度不超过 16 个字符。

◆ **程序文件的容量：**程序中指令部分最多可容纳 2000 行指令。

NOTE

◆ 程序文件取用途径：

1. 单个程序文件的取用：【编程】- 文件编辑工具栏【上传】【下载】
2. 所有程序文件的取用：【设置】-【系统设置】-【其它设置】-【程序备份】【程序加载】
3. 通过 Ftp 文件传输功能访问控制器。

3.1.1 程序中的位置变量

位置变量记录着空间中点的位置、姿态以及机器人到达此点的相关信息。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。各参数意义参见*“1.3 位置变量”。*

3.1.2 程序中的指令

程序文件中的指令以 START 开始，END 结束。指令以行的形式存储在程序文件中，每行都以“;”结尾。

3.2 变量

3.2.1 全局变量与局部变量

全局变量是所有程序都可以使用的变量，局部变量是只能在当前程序使用的变量。

■ **全局变量**

名称	变量	说明
全局 B 变量	B[0]~B[255]	Byte 型变量，取值范围 0-255
全局 R 变量	R[0]~R[255]	int 型变量，取值范围 -65535-65535
全局 D 变量	D[0]~D[255]	double 型变量，取值范围 -9999999.999~9999999.999
全局平移变量	PR[0]~PR[255]	6 个参数描述空间的平移

名称	变量	说明
全局托盘变量	Pallet[0]~ Pallet[255]	码垛工艺专用的变量
全局字符串变量	Str[0]~ Str[255]	字符串类型

■ **局部变量**

名称	变量	说明
位置变量	P[0]~P[9999]	机器人的位置，详见点位描述
局部 LB 变量	LB[0]~LB[255]	Byte 型变量，取值范围 0-255
局部 LR 变量	LR[0]~LR[255]	int 型变量，取值范围 -65535-65535
局部 LD 变量	LD[0]~LD[255]	double 型变量，取值范围 -9999999.999~9999999.999
局部平移变量	LPR[0]~LPR[255]	6 个参数描述空间的平移
局部托盘变量	LPallet[0]~LPallet[255]	托盘指令生成的托盘变量
局部字符串变量	指令定义名称	字符串类型

1 数值变量

用于存储数据，包括全局和局部的 B\R\D 变量，分别对应 Byte、int、double 类型的数据。

■ **编程示例**

B1 = 42;

R1 = 1000;

D1 = 123456.789;

R2 = R1+2;

LD1=(Sin(30)+D1)/2.1;

2 平移变量

平移变量用于描述空间的运动，可以描述在关节坐标系下 J1-J6 的运动，也可以描述三维空间内位置姿态 XYZABC 的变动。根据配合使用的 Offset 指令辨别。

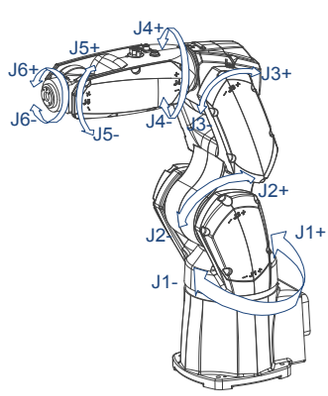
a）OffsetJ

■ **功能**

关节平移。

■ **描述**

PR 的六个坐标值为在关节坐标系的平移。



■ **编程示例**

P[2]=OffsetJ(P[1],PR1);

Movj OffsetJ(P[1],PR1) ,V[30], Z[0];

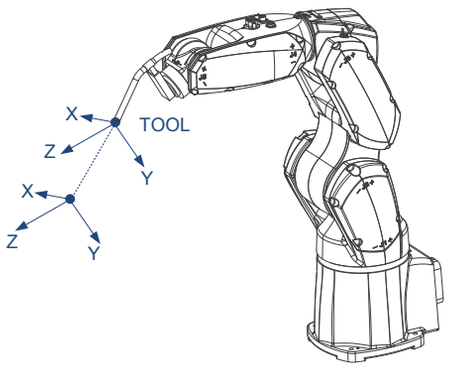
b）OffsetT

■ **功能**

沿当前工具位姿的平移。

■ **描述**

在当前的工具坐标系基础上进行平移，PR 为移动和旋转 (X,Y,Z,A,B,C)，平移顺序：X>Y>Z>A>B>C。



■ **编程示例**

P[2]=OffsetT(P[1],PR1);

Movj OffsetT(P[1],PR1) ,V[30], Z[0],Tool[1];

NOTE

◆ 若与运动指令一起使用，如 Mov OffsetT(P,PR)，则运动指令中必带有 Tool，不带 User。

c）Offset

■ **功能**

表示目标点在某个笛卡尔参考系下的平移。目标点可以是 TCP 也可以是法兰盘中心点，参考系可以是用户坐标系也可是基坐标系。

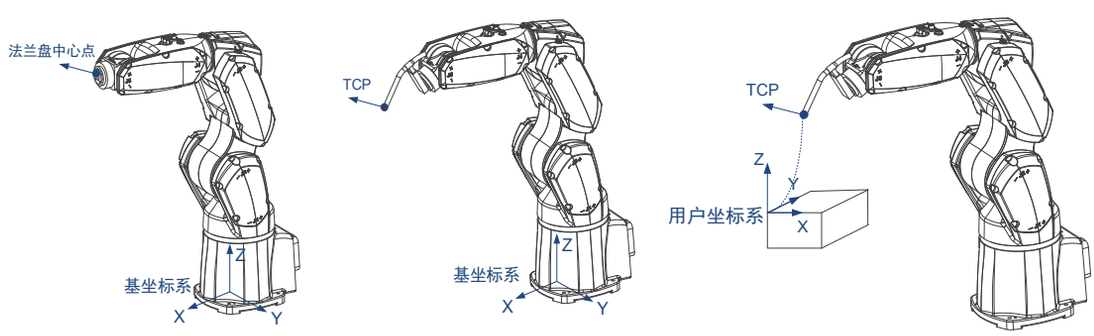
■ **描述**

表示目标点在笛卡尔参考系下的平移，PR 为移动和旋转 (X,Y,Z,A,B,C)，平移顺序：X>Y>Z>A>B>C。

对于 P=Offset(P,PR)，是在位置变量的坐标值上作平移。

由于坐标值的含义由坐标系指定，因此平移的目标点及参考坐标系由坐标系号指定。

对于 Mov Offset(P,PR)……，目标点由运动指令中的 Tool 指定，参考系由运动指令中的 User 指定。且运动指令带有与 P 不同的 Tool 或 User，则先更换工具或用户坐标系再平移。



■ **编程示例**

P[2]=OffsetT(P[1],PR1);

Movj OffsetT(P[1],PR1) ,V[30], Z[0],Tool[1];

3 托盘变量

全局托盘变量是为码垛工艺专用的变量。只在涉及码垛工艺时使用！内含构建的码垛信息，以及生成的一系列的的目标点。详见*《汇川机器人设计应用与维护手册 - 应用篇 2：高级功能应用》*中码垛工艺相关章节。

局部托盘变量只能通过指令 LPallet 或 LPallet4 生成，只在当前程序中有效，进入新的程序，上一个程序文件里的局部托盘变量会置空。局部托盘变量内含构建的托盘信息和生成的一系列的的目标点。

■ 编程示例

```
## 提前利用码垛工艺构建好 Pallet[1]
## 取托盘上第一个点（索引为 0 层、0 列、0 层），将其赋值为 P[0]
P[0]=Pallet(1,0,0,0);
## 由 P[1]、P[2]、P[3] 三点构建 2 行 3 列 1 层的托盘变量 LPallet[1]
LPallet 1,P[1],P[2],P[3],2,3,1,15;
## 取 LPallet[1] 上索引为 1 行、1 列、0 层的点，将其赋值为 P[10]
P[10]=LPallet(1,1,1,0);
```

4 字符串变量

全局字符串变量的名称固定为 Str[***]。对于局部字符串变量，名称是在程序中定义的，只有先被定义，才能被使用。

字符串变量名规格：由字母、数字、下划线组成，长度不超过 10 个的字符。

NOTE

◆ 字符串变量命名需注意：

1. 只能以字母开头；
2. 不能包含中文、全角字符；
3. 不能为关键字或其它已存在变量，如“Movj”、“V”、“B1”、“P”等。

字符串内容规格：由字母、数字、下划线组成，长度不超过 100 个的字符。

NOTE

◆ 字符串内容需注意：

1. 只能以字母开头；
2. 不能包含中文、全角字符；
3. 利用示教器编程时，由于界面限制，字符串长度不超过 50 个字符。

■ 编程示例

```
## 全局字符串直接使用
Str[1] =“a1b2c3”;
## 局部字符串定义后使用
String str1 =“abcd”;
str1 = Left(str1,1);
```

3.2.2 信号变量

信号变量是与 IO 有关的变量，有 In、Out、AD、DA 四种。

名称	变量	说明
数字量输入信号	In[0]-In[63]	有 ON/OFF 两种状态
数字量输出信号	Out[0]-Out[63]	有 ON/OFF 两种状态
模拟量输入信号	AD[0]-AD[15]	取值为一个数，由 IRLink 模块配置决定范围及单位
模拟量输出信号	DA[0]-DA[15]	取值为一个数，由 IRLink 模块配置决定范围及单位

NOTE

◆ 信号变量均为全局变量；
◆ 在实际使用过程时，可使用的信号变量受限于配置的 IRLink；
◆ 信号变量在缺少权限时不能使用。

■ **编程示例**

Set Out[1], ON;

Set DA[1], 1.2;

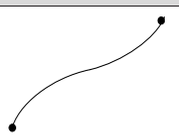
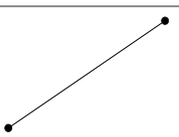
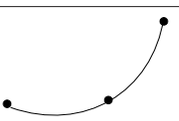
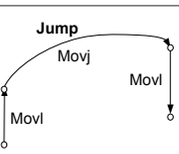
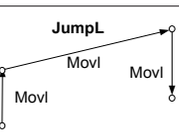
Get In[7], B1;

Get AD[1], D1;

3.3 运动编程

3.3.1 运动指令简介

机器人的运动相关指令主要有以下 5 种：

运动种类	轨迹	特点
关节运动 (Movj)		点到点的快速运动，所有轴同时到达目的位置，轨迹通常不在一条直线上。常用于点焊、运输等场合。
直线运动 (Movl)		工具中心点（TCP）线性移动到给定目标位置，轨迹为直线。常用于轨迹焊接、贴装等场合。
圆弧运动 (Movc)		工具中心点（TCP）按圆弧运动移动到给定目标位置。
跳跃运动 (Jump)		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。常用于 SCARA 机器人的拾取、放置物体。
直线跳跃 (JumpL)		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。常用于 SCARA 机器人的拾取、放置物体。

■ **编程示例**

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

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

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

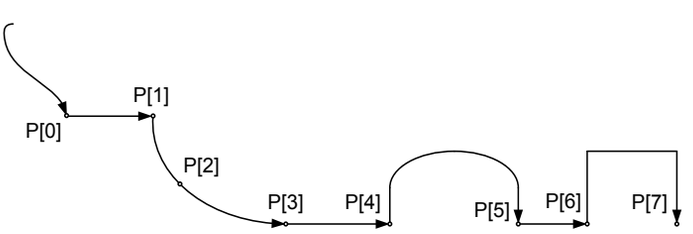
Movc P[3],V[30],Z[0];

Movl P[4],V[30],Z[0];

Jump P[5],V[30],Z[0],LH[20],MH[-300],RH[20];

Movl P[6],V[30],Z[0];

Jump P[7],V[30],Z[0],LH[30],MH[-300],RH[30];



3.3.2 运动指令的形式和参数

Move 系列运动主要有五种，形式如下：

Movj P, V, Z, [User], [Tool], [Acc], [NWait], [Until In == value], [Out(No,value,Type)…];

Movl P, V, Z, [User], [Tool], [Acc], [NWait], [Until In == value], [Out(No,value,Type)…];

Movc P, V, Z, [User], [Tool], [Acc], [NWait], [Until In == value], [Out(No,value,Type)…];

Jump P, V, Z, [User], [Tool], [Acc], [NWait], [Out(No,value,Type)…], LH, MH, RH;

JumpL P, V, Z, [User], [Tool], [Acc], [NWait], [Out(No,value,Type)…], LH, MH, RH;

参数项	说明
P	要达到的目标点。也可为偏移形式和托盘取点形式
V	指定速度
Z	到位与过渡参数
User（可选参数）	选用的用户坐标系
Tool（可选参数）	选用的工具坐标系
Acc（可选参数）	指定加速度
NWait（可选参数）	不等待标识
Until In == value （可选参数）	运行中检测信号，满足条件则立即停止
Out(No,value,Type)（可选参数）	运行中并行输出信号。

以上参数中，只有 P、V、Z 是必须使用的，其它参数是可选参数，适用于某些特殊情形。

下面将针对一些关键参数介绍。

1 V 与 Acc

V 指定速度，Acc 指定加速，以百分比的形式影响运动。

运行速度 = 设置的最大速度 * 全局速度百分比 * 指令设置的百分比

运动指令中的V

运行加速度 = 设置的最大加速度 * 指令设置的百分比

运动指令中的Acc

■ 编程示例

```
Movj  P[0] ,V[70], Z[3];                  ## 速度为 70%，加速度默认为 100%
Movj  P[1] ,V[30], Z[3],Acc[50];          ## 速度为 30%，加速度 50%
Movj  P[2] ,V[30], Z[3],Acc[80];          ## 速度为 30%，加速度 80%
```

2 Z

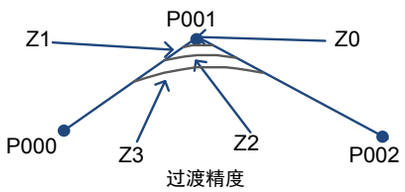
Z 是到位与过渡参数。有 Fine、Z[0]、Z[1]、Z[2]、Z[3]、Z[4]、Z[5]、Z[CP] 八种选择。

Fine: 代表精确到位。机器人进入并停留在设定的到位误差范围内时，表示精确到位，再执行后面的运动。

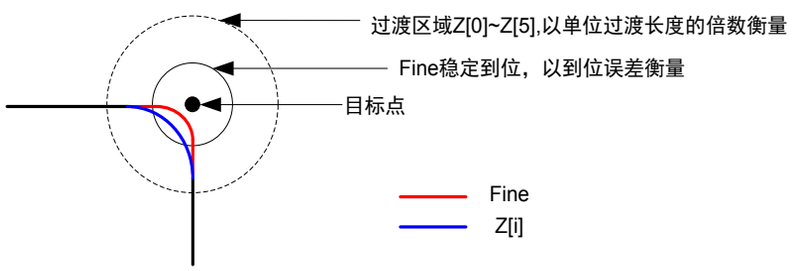
Z[0]~Z[5]: 表示机器人不在目标点停留，而是平滑的通过设定的过渡区域。

类型	方式
Fine	代表精确到位。机器人进入并停留在设定的到位误差范围内时，被算作精确到位，然后再执行后面的运动。
Z[0]	无过渡，但不能保证机器人精确到达目标点

类型	方式
Z[1]	有过渡，以 1 倍过渡单位长度过渡
Z[2]	有过渡，以 2 倍过渡单位长度过渡
Z[3]	有过渡，以 3 倍过渡单位长度过渡
Z[4]	有过渡，以 4 倍过渡单位长度过渡
Z[5]	有过渡，以 5 倍过渡单位长度过渡
Z[CP]	有过渡，以最大过渡长度过渡

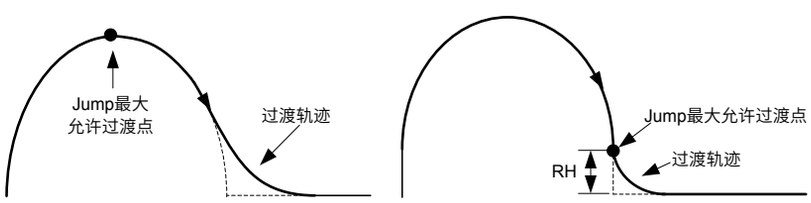


使用过渡能加快运动节拍。过渡范围是以目标点为圆心，以过渡长度为半径的区域。过渡长度等于过渡等级与过渡单位长度的乘积。



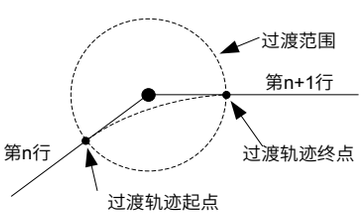
单位过渡长度分为关节过渡单位和线性过渡单位，MoveJ 和 Jump 指令使用关节单位（SCARA 机型 3 轴仍使用线性单位），其他运动指令使用线性单位。

对于具体的运动，允许的过渡长度是有限制的，当设定过渡长度超出允许的最大过渡长度时，取最大过渡长度作为实际的过渡长度。MovL、MoveJ、MovC 的最大允许过渡长度是总长度的一半。对于 Jump 和 JumpL 指令，当 RH(或 LH) 等于零时，最大允许过渡长度为总长度的一半；当 RH(或 LH) 不等于零时，最大允许过渡长度为 RH(或 LH)。



RH=0的Jump过渡　　RH不等于0的Jump过渡

过渡对运动 IO 的影响：当存在下图中虚线所示的过渡区域时，设定在过渡区域内的运动 IO 生效精度是不能保证的，实际会在过渡轨迹起点和过渡轨迹终点之间不确定的位置生效。



■ 编程示例

```
Movj  P[0] ,V[70], Fine;          ## 精确到位
Movj  P[1] ,V[30], Z[0];          ## 无过渡
Movj  P[2] ,V[30], Z[5];          ##Z[5] 过渡
```

3 User

位置变量包含 Tool 信息，因此，运动指令可以不使用 Tool 参数。

注：

位置变量为：*P[1]=(100,0,0,0,0,0;1,0,0,0;1,2,3);*

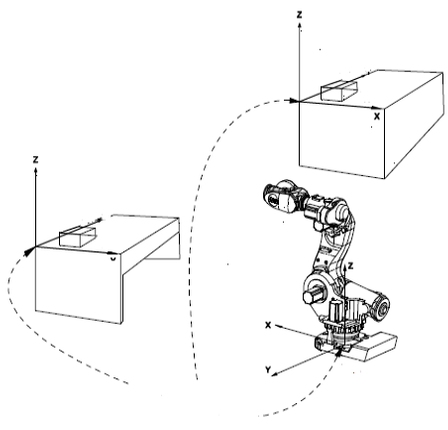
运动指令为：*Movj P[1],V[30],Z[0];* 或 *Movj P[1],V[30],Z[0],Tool[2],User[3];*

此时，程序运行的结果一样。

利用运动指令中 User 参数，实现切换用户坐标系。

应用场合：原来已示教编好程序。后来工件位置发生变化，不必逐点重新示教，直接在运动指令中指明新用户坐标系即可。

注意：重要的前提条件：需要原程序取点都在用户坐标系下。



■ **案例**

原来在工作台下的一批工件，在工作台上建立 User1，使用 User1 作为当前的用户坐标系。然后在用户坐标系下示教取点编程：

Movj P[1],V[30],Z[0];		*Movj P[1],V[30],Z[0],User[1];*
MovI P[2],V[30],Z[3];	或	*MovI P[2],V[30],Z[3],User[1];*
MovI P[3],V[30],Z[3];		*MovI P[3],V[30],Z[3],User[1];*

当工作台的摆放位置改变时，只需在新工作台上建立 User2。那么可以直接更改：

MovjP[1],V[30],Z[0],User2;
MovlP[2],V[30],Z[3],User2;
MovlP[3],V[30],Z[3],User2;

NOTE

◆ 若希望机器人通过奇异位置，则只能使用 Movj 指令，并且 P 的坐标系号为 1，不能切换 Tool。

4 Tool

位置变量包含 Tool 信息，因此，运动指令可以不使用 Tool 参数。

注：

位置变量为：*P[1]=(100,0,0,0,0,0;1,0,0,0;1,2,3);*

运动指令为：*Movj P[1],V[30],Z[0];* 或 *Movj P[1],V[30],Z[0],Tool[2],User[3];*

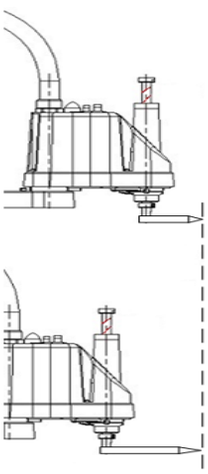
此时，程序运行的结果一样。

运动指令中的 Tool 参数，可用于切换工具，新 TCP 到达原 TCP 的位置姿态。

应用场合：首先使用旧工具示教编写了一段加工程序；由于工具变换，后期使用新工具进行同样的加工，此时在运动指令中指明新工具即可，可不必逐点重新示教。

NOTE

◆ 换工具功能，仅要求位置变量的工具号选择正确。与位置变量中坐标值的参数无关。



■ **案例**

首先使用 Tool1 示教取点，注意：位置变量的工具号为 1。编程如下：

```
Movj P[1],V[30],Z[0];
MovI P[2],V[30],Z[3];
MovI P[3],V[30],Z[3];
```

或

```
Movj P[1],V[30],Z[0],Tool[1];
MovI P[2],V[30],Z[3],Tool[1];
MovI P[3],V[30],Z[3],Tool[1];
```

后替换工具，使用 Tool2 进行加工，此时不必重新示教，直接更改指令中的 Tool：

```
MovjP[1],V[30],Z[0],Tool2;
MovlP[2],V[30],Z[3],Tool2;
MovlP[3],V[30],Z[3],Tool2;
```

NOTE

◆ 若希望机器人通过奇异位置，只能使用 Movj 指令实现，要求 P 的坐标系号为 1，不能切换 Tool、User，不能带有平移。

5 NWait

NWait 为不等待标识，表示还未运行到目标点时，提前执行下一条运动指令前的非运动指令（如运算、信号、逻辑判断等）。

区别如下：

```
## 不等待运动到位，提前置信号
Movj P[1],V[30],Z[0],NWait;
Set Out[3],ON;
```

```
## 运动完成后才置信号
Movj P[1],V[30],Z[0];
Set Out[3],ON;
```

■ **编程示例**

```
## 不等待运动到位，一开始运动立即置信号
Movj P[1],V[30],Z[0];
MovI P[2],V[30],Z[3],NWait;
MovI P[3],V[30],Z[3],NWait;
Set Out[3],ON                          ## MovI P[2] 开始时就提前置 Out[3]
Movj MovI P[3],V[30],Z[0];
Set Out[4],ON;                         ## 等待 Movj P[3] 运动结束再置 Out[4]
```

6 Until

在运动过程中检测输入信号量，当条件满足时，当前行运动立即中止，执行后面行的程序；若条件不满足时，运动到结束点。

注意：使用 Until 参数时，Z 会被视为 Z[0]。

Until In == Value：

In：输入信号

Value：输入信号值，ON 或 OFF

■ 编程示例

```
## 运动中检测 In[5]，当为 ON 时立即中止运动；否则一直运动直至结束
Movj P[1],V[30],Z[0], Until In[5] == ON;
```

7 运动中的 IO

“Out(No,value,Type)…”参数用于在运行过程中并行输出信号。

NOTE

◆ 一条运动指令中最多并行输出 2 个信号。

Out(No,value,Type)…

No: 输出信号序号。

Value: 输出信号值。

Type: 运动中输出信号的类型。不同类型运动能使用不同的 Type。

Type	意义	适用的运动类型
T[n]	在运动中按时间输出信号 T[n] 代表时间，单位：秒，范围 -65535.000~65535.000。 n>=0 表示开始运动 n 秒后输出信号；n<0 表示到达目标点之前 n 秒时输出信号。 当超出运动范围外的时间被设定时，最多只在该段行程首末点触发，设定的时间无效	Movj/ Movl/ Movc/Jump/JumpL
D[n]	在运动中按路径百分比输出信号 D[n] 表示路径百分比。表示从开始运动到结束的 n% 路径时输出信号。	Movj/ Movl/ Movc/Jump/JumpL
S[n]	在运动中按距离输出信号 S[n] 表示距离，单位：mm。范围 -65535~65535。 n>=0 表示从起点开始运动到 n 毫米之后时再输出信号；n<0 表示运动到距终点 n 毫米之前时输出信号。	Movl/ Movc/JumpL

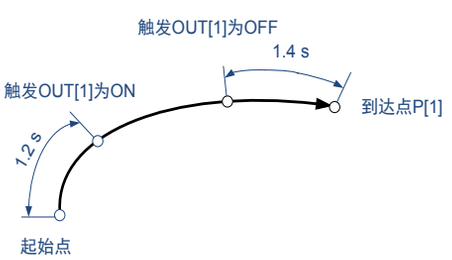
NOTE

◆ 运动 IO 若被设定在过渡区域内，精度不能得到保证。

◆ 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。

■ 编程示例

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



8 Offset

关于偏移“Offset”的使用：

a）OffsetJ

■ **功能**

关节平移。

■ **典型使用形式**

P[2]=OffsetJ(P[1],PR1);

Movj OffsetJ(P[1],PR1) ,V[30], Z[0];

■ **特征**

使用 OffsetJ 时，此时 PR 的六个坐标值为关节旋转角度。内部会自动根据位置变量 P 计算关节值，然后进行平移。

若与运动指令一起使用，形如 MovOffsetJ(P,PR)，则运动指令中不能带有 Tool、User。

■ **编程示例**

在 P[1] 基础上，机器人第一关节额外旋转 10°，第二关节额外旋转 20°（方法一）

PR1=(10,20,0,0,0,0);

Movj OffsetJ(P[1],PR1) ,V[30], Z[0];

在 P[1] 基础上，机器人第一关节额外旋转 10°，第二关节额外旋转 20°（方法二）

PR1=(10,20,0,0,0,0);

P[2]=OffsetJ(P[1],PR1);

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

b）OffsetJ

■ **功能**

沿当前工具位姿的平移。

■ **典型使用形式**

P[2]=OffsetT(P[1],PR1);

Movj OffsetT(P[1],PR1) ,V[30], Z[0],Tool[1];

■ **特征**

使用 OffsetT 时，在当前的工具坐标系基础上进行平移。PR 为移动和旋转 (X,Y,Z,A,B,C)，平移顺序：X>Y>Z>A>B>C

若与运动指令一起使用，形如 MovOffsetT(P,PR)，则运动指令中必带有 Tool，不带 User。

■ **编程示例**

要到达的目标点的位置为，在 P[1] 基础上，沿着当前的工具坐标系运动，沿 X 方向移动 10mm，绕 Z 旋转 10°，再绕 Y 旋转 20°的位置，最后绕 X 旋转 30°的位置（方法一）

PR1=(10,0,0,10,20,30);

MovjOffsetT(P[1],PR1) ,V[30], Z[0],Tool[1];

要到达的目标点的位置为，在 P[1] 基础上，沿着当前的工具坐标系运动，沿 X 方向移动 10mm，绕 Z 旋转 10°，再绕 Y 旋转 20°的位置，最后绕 X 旋转 30°的位置（方法二）

PR1=(10,0,0,10,20,30);

P[2]=OffsetT(P[1],PR1);

Movj P[2],V[30],Z[0],Tool1;

c）Offset

■ **功能**

表示目标点在某个笛卡尔参考系下的平移。目标点可以是 TCP 也可以是法兰盘中心点，参考系可以是用户坐标系也可以是基坐标系。

■ **典型使用形式**

P[2]=Offset (P[1],PR1);

Movj Offset(P[1],PR1) ,V[30], Z[0],Tool[1],User[1];

■ **特征**

使用 Offset 时，目标点在笛卡尔参考系下的平移。此时 PR 为移动和旋转 (X,Y,Z,A,B,C)，平移顺序：X>Y>Z>A>B>C。

对于 P=Offset(P,PR)，是直接在位置变量的坐标值上作平移。由于坐标值的含义由坐标系指定，因此平移的目标点及参考坐标系由坐标系号指定：

1） 当位置变量中的坐标系号为 1 时，表示法兰盘中心点在基坐标下平移。

2） 当位置变量中的坐标系号为 2 时，表示法兰盘中心点在基坐标下平移。

3） 当位置变量中的坐标系号为 3 时，表示 TCP 在基坐标下平移。工具由位置变量的工具号指定。

4） 当位置变量中的坐标系号为 4 时，表示 TCP 在用户坐标下平移，工具由位置变量的工具号指定，用户坐标系由位置变量的用户号指定。

NOTE

◆ 可以利用 Cnvrt 指令更改位置变量的在指定的坐标系下表示，再配合 P=Offset(P,PR)，就可以明确平移的目标点和参考系。我们推荐这样使用会更清晰！

■ **编程示例**

编程示例 1：

基坐标系下取点，法兰盘在基坐标系下移动。

P[1]=(10,20,30,40,50,60;1,0,0,0;2,1,0);## 坐标系号为 1

PR1=(10,0,0,0,0,0);

P[2]=Offset(P[1],PR1);

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

由于 P[1] 在基坐标系下取点（坐标系号 2），因此是法兰盘中心相对基坐标的平移。注意此时的目标点已经不是 TCP。此时若需要 TCP 相对基坐标系的平移，可编程如下：

P[1]=(10,20,30,40,50,60;1,0,0,0;2,1,0);

Cnvrt(P[1],P[1],Tool[1]);## 将 P[1] 转换到工具坐标系下表达

P[2]=Offset(P[1],PR1);

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

如果 P[1] 是在工具坐标系下取点（坐标系号 3），可以不用 Cnvrt 转换。再比如，由于某种特殊需求，需要 TCP 相对额外的一个用户坐标系 User1 平移，可编程如下：

P[1]=(10,20,30,40,50,60;1,0,0,0;2,1,0);

Cnvrt(P[1],P[1],User[1]);## 将 P[1] 转换到 User1 下表达

P[2]=Offset(P[1],PR1);

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

编程示例 2：

工具在基坐标系下平移

P[1]=(10,20,30,40,50,60;1,0,0,0;2,1,0);

Movj Offset(P[1],PR1),V[30],Z[0], Tool[1];

此时，只要运动指令 Movj 带有 Tool 不带 User，就是工具在基坐标系下平移。位置变量 P 可以在任何坐标系下取。

编程示例 3：

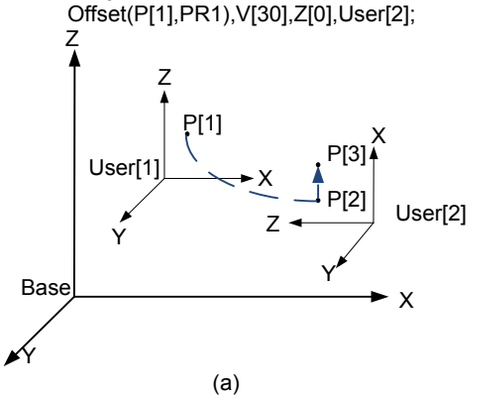
先换用户坐标系再平移时，需要注意切换用户坐标系需满足条件！

当 P 不为用户坐标系下取的点时，由于不符合更换用户坐标系条件，平移起始点不发生变化，但是平移是沿着

指令中 User 坐标系下的。

当 P 为用户坐标系下取的点时，先更换用户坐标系（平移起始点发生变化），再沿用户坐标系平移。

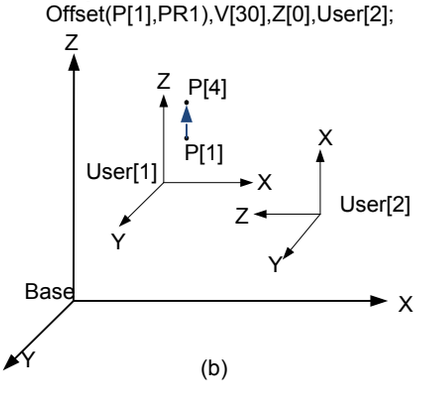
```
P[1]=(10,0,20,0,0,0),(1,1,1,0),(4,0,1);
PR1=(20,0,0,0,0,0);
Movj
Offset(P[1],PR1),V[30],Z[0],User[2];
```



(a)

图(a)：
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。如图(a)中，最终运动到位置P[3]

```
P[1]=(50,0,60,0,0,0),(1,1,1,0),(2,0,1);
PR1=(20,0,0,0,0,0);
Movj
Offset(P[1],PR1),V[30],Z[0],User[2];
```



(b)

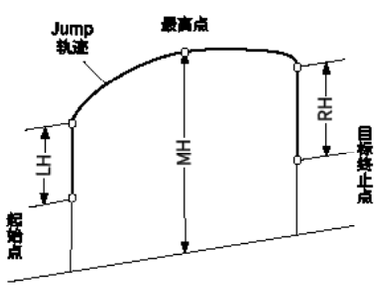
图(b)：
情形：Movj带有User参数，P的坐标系号为0或1或2。
特点：位置变量的绝对位置不发生变化，直接进行平移。如图(b)中，最终运动到位置P[4]

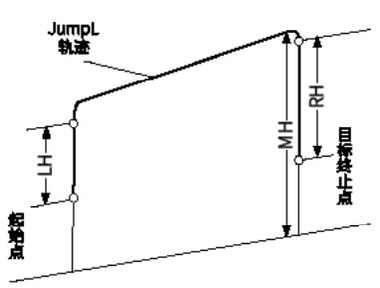
9 Jump 与 JumpL

Jump 指令仅适用于 SCARA 和 Delta 机器人。它们都是跳跃指令，按照抬起 - 移动 - 下降的方式运动，其本质是由 Movj 与 Movl 组成。两者的区别在于 Jump 中间是一段 Movj，JumpL 中间是一段 Movl。

Jump 格式：Jump P, V, Z, [User], [Tool], [Acc], [NWait], [Out(No,value,Type)…], LH, MH, RH;

JumpL 格式：JumpL P, V, Z, [User], [Tool], [Acc], [NWait], [Out(No,value,Type)…], LH, MH, RH;





Jump 指令中大部分参数与 Move 指令参数相同，对 Jump 指令特有的参数进行说明

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, 用于限制高度；

RH: 到终止位置的下降高度，取值范围 0~2000。

NOTE

- 由于 SCARA 机器人的基坐标系零点位于上方，MH 的衡量是一个绝对值，因此 MH 多数情形下可能为负值。
- 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，可用于特殊场合。

■ 编程示例

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

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

3.4 IO 编程

3.4.1 编程前的准备

在 IO 编程前，需要保证 IRLink 配置正确。且对于输出信号的控制，需要具有 IO 控制权。在示教器上，我们能很容易的确认这些前提条件，在示教器上的【监控】-【IO 监控】界面，观测是否具有此 IO，且 IO 颜色不为橙色，即表示前提条件已满足。

比如：当出现下面这种情形，表示无法使用 Out[2] 编程。

3.4.2 IO 读写

以下指令可以完成 IO 的单个读写：

指令	说明	备注
Set Out,value;	设置 Out 信号变量	
Set DA,Value;	设置 DA 信号变量	
Get In,B;	获取 In 信号变量	信号的值放入 B 变量。0 为 OFF，1 为 ON
Get Out,B;	获取 Out 信号变量	信号的值放入 B 变量。0 为 OFF，1 为 ON
Get AD,D;	获取 AD 信号变量	信号的值放入 D 变量

NOTE

◆ 对于模拟量 IO，根据 IRLink 模块的配置决定 value 值的单位。若该信号被配置为电流，则单位为 mA；若该信号被配置为电压，则单位为 V。

■ **编程示例**

```
Set Out[1],ON;                          ## 设置 Out[1] 输出 ON
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
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;
Get Out[0],B2;                          ## 获取 Out[0] 值
Get AD[1],D1;                           ## 获取 AD[1] 的值
```

3.4.3 组 IO 读写

用 OG 代表一组 Out 信号变量，用 IG 代表一组 In 信号变量。默认情形下，一组信号包含 8 个信号。如 OG[0] 为 Out[0]~Out[7]，OG[1] 为 Out[8]~Out[15]，以此类推。

一组信号的值可存于 B 变量中。B 变量的值，为 Byte 型，对应 8 位二进制数，这样从右到左每一位代表从低位到高位的一个信号状态。

指令	说明	备注
Set OG,B;	设置一组输出信号	B 变量包含一组信号的值
Get IG,B;	获取一组输入信号	B 变量包含一组信号的值
Get OG,B;	获取一组输入信号	B 变量包含一组信号的值

■ **编程示例**

```
B1=7;
Set OG[0],B1;                    ## 设置 OG[0] 输出 0000 0111，即 Out[0]~Out[7] 分别为 1110 0000
Get  IG[1],B1;                   ## 读取 IG[1] 的信号
## 若信号 In[15]~In[8] 依次为 OFF、OFF、OFF、OFF、OFF、ON、ON、ON，则 B1=7
Get  OG [1],B2;                  ## 读取 OG [1] 的信号
## 若信号 Out [15]~Out [8] 全为 OFF，则 B2=0
```

使用 Group 指令，可重新定义组信号。组中信号的数目可小于 8。但要注意，仅在当前程序有效。

■ **格式**

Group OG, n0, n1, n2…, n7;

Group IG, n0, n1, n2…, n7;

■ **参数**

OG：待重定义的输出组，OG[0]~OG[8]。

IG：待重定义的输入组，IG[0]~IG[8]。

n0, n1, n2…, n7：信号的序号。可多选，最多 8 个，最少 2 个。

■ **编程示例**

Group OG[0] 1,2,7; ## 重定义 OG[0] 为 Out[1]、Out[2]、Out[7]。

3.4.4 其它 IO 指令

Invert：反转信号

Wait：等待信号

Plus：脉冲输出

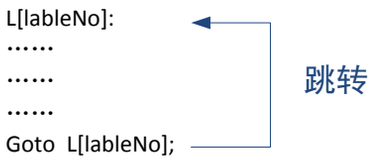
运动中 IO

详见*《汇川机器人设计应用与维护手册 - 附录：机器人指令集》*。

3.5 逻辑控制编程

3.5.1 跳转

L-Goto 逻辑结构实现跳转功能。其中，L 用于设置程序标签，与跳转指令 Goto 配合使用，完成跳转动作



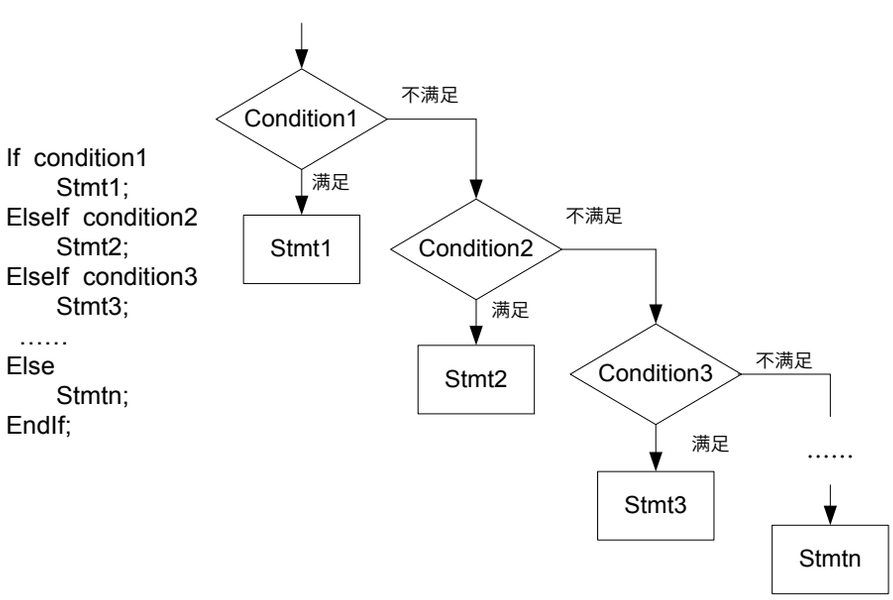
■ **编程示例**

```
## 先运行至 P[0] 位置，然后在 P[1] 与 P[0] 两点间往复运动。
START;
Movj P[0],V[30],Z[3];
L[1]:                                    # 设置标签 1
MovI P[1],V[30],Z[3];
MovI P[0],V[30],Z[3];
Goto L[1];                               # 跳转至标签 1
END;
```

3.5.2 选择类

1 If--EndIf

逐个进行条件判断，若满足则执行对应语句并结束，不满足条件则继续判断，所有条件均不满足时，执行 Else 后语句。



此结构中，只有 If—EndIf 语句是必须的，ElseIf、Else 可省略。这样仅执行一次判断，若满足才执行对应语句。

NOTE

- 判断条件 Condition 可为多个条件复合。
- 语句 Stmt 可为数行指令。

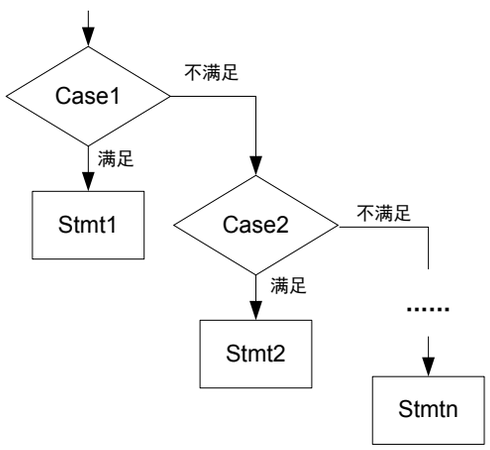
■ **编程示例**

```
START;
If IN[1]==ON And IN[2]==ON ;
        Movj P[1],V[50],Z[3];
Else
        Movj P[2],V[50],Z[3];
Movj P[3],V[50],Z[3];
EndIf;
End;
```

2 Switch-Case-Default-EndSwitch

用 Case 值逐一去匹配 Switch 对象，若满足，则执行对应语句则结束，不满足条件则继续匹配，若所有 Case 均不符合，则执行缺省 Default 后的语句

```
Switch Var
  Case
  value1:
      Stmt1;
      Break;
  Case
  value2:
      Stmt2;
      Break;
  ……
  Default:
      Stmtn;
      Break;
EndSwitch;
```



NOTE

◆ **扩展：**Case A To B : 可使用 Case A To B 代替 Case Value。Case A To B 代表从 A 到 B 两个整数范围内 (包含 A、B) 的的选择。当变量值在这一范围内时，代表此 Case 项匹配。

NOTE

◆ 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 段落语句可省略。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。

■ **编程示例**

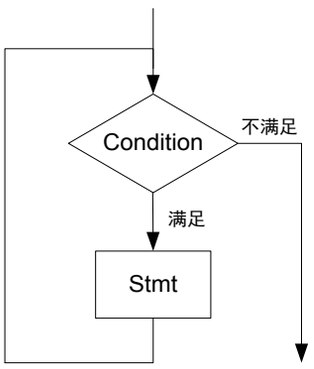
```
START;
   Switch B0
      Case 1:
         Movj P[1],V[50],Z[3];
         Break;
      Case 2 To 4:
         Movj P[1],V[50],Z[3];
         Movj P[2],V[50],Z[3];
         Break;
      Case 5:
          Movj P[3],V[50],Z[3];
          Break;
      Default:
         Movj P[4],V[50],Z[3];
         Break;
   EndSwitch;
End;
```

3.5.3 循环类

1 While-EndWhile

判断条件，若满足条件，则执行循环体内的语句，完成后再次进行判断，以此循环。一旦不满足条件，则跳出。

```
While
Condition
    Stmt;
EndWhile;
```



NOTE

- 判断条件 Condition 可为多个条件复合。
- 语句 Stmt 可为数行指令。

■ 编程示例

```
## 在 P[1] 至 P[2] 之间来回运动三次
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;
```

2 For-EndFor

带执行次数的循环语句。先执行一个初始化赋值语句，再判断条件，若满足条件则执行循环体内的内容，并执行一个步长，使得初始化赋值语句中的变量自增。再此判断条件 , 若满足则继续刚才的步骤，直至条件不成立时跳出。

■ 编程示例

```
# 循环执行 3 次两个 Movj 指令
START;
For B0=0,B0<5,Step[2]
Movj P[2],V[50],Z[3];
Movj P[3],V[50],Z[3];
EndFor;
End;
```

3.5.4 调用子程序

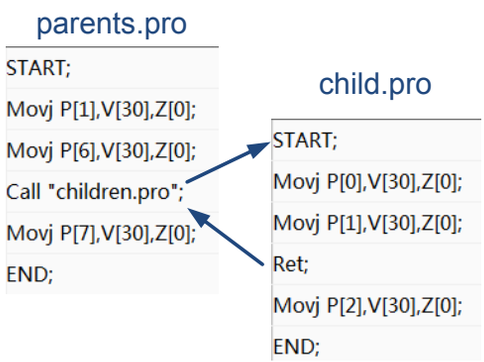
1 Call 与 Ret

可以在一个程序中调用另一个子程序。从而进入子程序程序，在子程序中遇到 Ret 指令，则返回主程序继续执行；若未遇到 Ret 指令，则执行到子程序的 End 就结束，不再返回主程序。

支持嵌套调用子程序，但嵌套使用不能超过 3 层。

■ **编程示例**

主程序 parents.pro 中调用子程序 child.pro



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：

主程序**P[1]** → 主程序**P[6]** → 子程序**P[0]** → 子程序**P[1]** → 主程序**P[7]**

2 USING MAIN

在主子程序的交互过程中，一般使用全局变量来交互。位置变量无全局变量，我们可以通过 USING MAIN 指令实现在子程序中调用主程序中的点。

■ **格式**

USING MAIN,P,VarNum；

■ **参数**

P：主程序中位置变量。

VarNum：从 P 开始要使用的连续变量的个数（1~9999）。

■ **编程示例**

 # 用主程序的 P[2]、P[3] 替代子程序的 P[2]、P[3]

 # 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"
Goto L[1];
END;

 # 子程序 Main.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;

3.6 TCP/IP 通信编程

3.6.1 外设与控制器 TCP/IP 通信简介

控制器与外部设备在可以通过 TCP/IP 通信。

操作简介：

步骤 1：本地控制器作服务器，外部设备作客户端：

需要设定控制器的 EtherNet1 作为服务器，并指定端口号。

默认系统启动时就开启，作客户端的外部设备可与控制器通讯，直接编写指令负责收发、处理数据即可。

步骤 2：本地控制器作客户端，外部设备作服务器：

需要设定控制器的 EtherNet1 作为客户端。

客户端的打开需要通过指令“Open Socket”，然后才可编写指令负责收发、处理数据，最后通过指令“Close Socket”关闭。


NOTE

◆ 设定控制器的 EtherNet1 通过示教器 -【设置】-【系统设置】-【通讯设置】中设定，更改后重启生效！

3.6.2 TCP/IP 通信指令

通信过程中常用到的指令如下：

1 Open Socket

■ **格式 1**

Open Socket(IP, SvrPort, ClientPort)

■ **功能简介 1**

通信打开，连接到服务器。输入服务器的 IP 地址和端口号指定连接对象。输入客户端端口号以指定通信的客户端端口。


NOTE

◆ 本格式仅适用于机器人控制器作客户端，外部以太网设备作服务器的情况！

◆ 为了支持在程序中打开多个客户端端口，连接不同服务器，因此需要在指令中指定客户端口号。

■ **格式 2**

Open Socket(IP, SvrPort, ClientPort,B)

■ **功能简介 2**

功能简介：相比格式 1 相比，多一个参数 B，表示打开成功与否。在使用时，一般利用循环语句，循环执行，对参数 B 进行判断，在打开成功后跳出。

2 Close Socket

■ **格式**

Close Socket,ClientPort

■ **功能简介**

关闭客户端的端口号。仅用于机器人控制器作客户端，外部以太网设备作服务器。

3 Send Port

■ **格式**

格式 1：Send Port[Port],String

格式 2：Send Port[Port],String,Type;

其中：Type：发送数据类型，可以为 String/Binary/Hex。没有此参数时，默认为以字符串形式发送，相当于使用 String。

■ **功能简介**

将字符串发送到网络的远端端口。此 Port 为客户端端口号。

◆ 特别的，本地控制器作服务器，外部仅有唯一客户端时，当不知道外部客户端的端口号，可以使用端口号 4444。

4 Get Port

■ **格式**

Get Port[Port],T[Time],Goto L[Index]

■ **功能简介**

在一定时间内，接收远端端口的数据，放入缓冲区。当超过指定的时间而未接收到数据，则跳转到指定的标签处。此 Port 为客户端端口号。

◆ 特别的，本地控制器作服务器，外部仅有唯一客户端时，当不知道外部客户端的端口号，可以使用端口号 4444。

5 GetPortbuf

■ **格式**

StrVar = GetPortbuf(StartBit , Num, Port)

■ **功能简介**

从接收缓冲区指定位置开始读取指定数量的字符并存储到指定的字符串变量中。

6 其它

接收是以字符串形式，往往需要字符串处理相关指令。如 StrGetData 字符串分割等指令。更多见*《汇川机器人设计应用与维护手册 - 附录：机器人指令集》*。

3.6.3 通信编程示例

```
# 机器人作客户端，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;
PR0=(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);                    # 打开客户端端口
```

```
If B0 == 0
Goto  L[0];
EndIf;
Send Port[1026],TxBuf;                                  # 发送请求
L[1]:                                                   # 接收数据
Get Port[1026], T[10], Goto L[1];
RxBuf = GetPorbuf(0,100,1026);
B1 = StrGetData(RxBuf,”#”,P10);                         # 处理接收数据并控制运动
Cnvrt(P[10],P[20],World]);
Movl Offset(P[20],PR0),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
# 外部视觉作服务器，【通讯设置】设置服务器端口号 1025
START;
TxBuf = "TA";
RxBuf = "";
L[2]:
Movj P[0],V[30],Z[0];
Send Port[1026],TxBuf;                                  # 发送请求
L[1]:                                                   # 接收数据
Get Port[1026], T[10], Goto L[1];
RxBuf = GetPorbuf(0,100,1026);
B1 = StrGetData(RxBuf,”#”,P10);                         # 处理接收数据并控制运动
Cnvrt(P[10],P[20],World]);
Movl Offset(P[20],PR0),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];                                    # 关闭端口
Delay T[1];
Goto L[2];
END;
```

3.7 编程案例

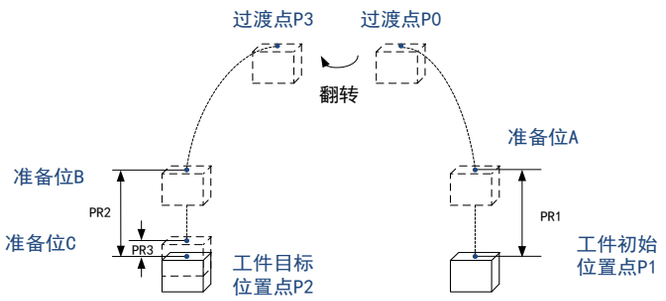
■ **案例描述**

采用六自由度机器人吸附工件，从一处搬运至另一处。

■ **步骤说明**

步骤 1：为吸附工件，在机器人上设置好吸附装置，并连接好 IO。

步骤 2：在示教器上新建一个程序，并示教编程。示教运动流程如下：



步骤 3：运行程序，结束。

■ **数据配置**

1） 输入输出信号

Out[1]	机器人吸附开关　　ON：开 /OFF：关

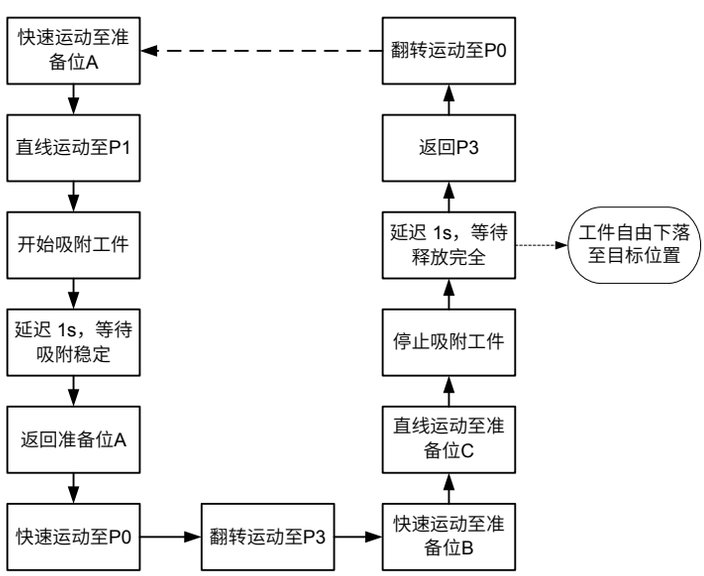
NOTE

◆ 本案例中，机器人吸附开关在开启和关闭时，等待 1s。

2） 位置变量

P1	工件初始位置，在该点执行吸附操作，将工件拾取。
P2	工件目标位置，工件想要达到的点
Offset(P[1],PR1)	准备位 A，拾取工件的准备位置，快速运动和直线插补的临界点
Offset(P[2],PR2)	准备位 B，释放工件的准备位置，快速运动和直线插补的临界点
Offset(P[2],PR3)	准备位 C，释放工件的位置
P0	过渡点 P0，在 P0 与 P3 间翻转工件，在安全高度完成姿态变化
P3	过渡点 P3

■ **程序流程图**



■ **编程示例**

```
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;
```

应用篇 2：高级功能应用

应用篇 2：高级功能应用 - 目 录

第 1 章 性能调试

1.1 机器人作业节拍优化调试指导

节拍优化的最高原则是：以机器人最小的运动代价达到期望的节拍。当机器人的运动代价越小，核心部件的损耗也就越小，机器人的寿命能有效地得到保障。

机器人作业节拍的调试与现场情况密切相关，因此，机器人作业节拍的调试应以实践经验积累为主，本节主要提供节拍调试的思路和建议。

节拍优化流程：

1） 系统流程优化：判断系统是否处于必要的等待状态；

2） 机器人运动流程优化：优化轨迹、点位和过渡特性；

3） 加速度参数优化：判断最大电流与平均电流是否在规定范围内；

4） 伺服参数优化：调节伺服参数，折中考虑抖动和精度。

1.1.1 系统流程优化

系统流程主要优化以下两个方面：

■ 优化作业流程

作业流程是否可以优化的判断标志是：机器人是否处于不必要的等待状态中。例如在机器人和转盘配合的应用中，可根据实际情况让机器人和转盘同时运动，而不是等到转盘到位之后，机器人再开始运动，以节省作业时间。作业流程的优化一般情况下对提升节拍时间是最直接有效的，作业流程的优化从客户方案设计阶段就可以介入。

■ 机器人周边设备的性能调试

对于机器人周边设备的性能调试，核心在于掌握这些设备的作业性能对机器人作业节拍的影响。例如电池阀压力检测对机器人的抓取等待时间可能产生的影响，不同的电缸速度参数对作业节拍可能产生的不同影响等。

调试机器人周边设备的性能时，请特别关注执行部件（如夹持具）的性能。

1.1.2 机器人运动流程优化

机器人运动流程的优化和现场作业流程密切相关，本小节重点是梳理优化思路。机器人运动流程优化，是指在完成既定作业的情况下，机器人运动轨迹或点位是最优的。机器人运动流程主要优化以下五个方面：

■ 机器人的目标作业范围应处于机器人相对舒服的操作空间内

不同机型的机器人有对应的操作空间，如果机器人目标作业范围过于接近机器人操作空间的最内侧或最外侧，则机器人的运动能力会大大降低。因此在介入客户目标作业范围的设计工作时，使机器人目标作业范围都落在机器人（以 SCARA 为例）最内侧作业圆与最外侧作业圆靠近中间的那一部分空间。

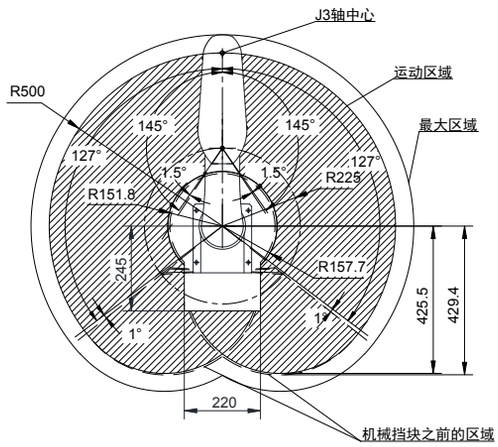


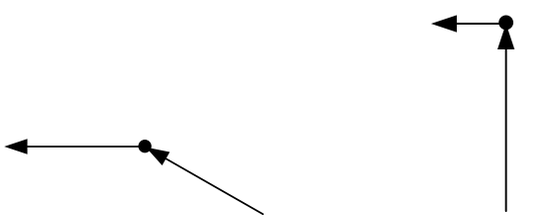
图 1-9 SCARA 机器人操作空间（以 IRB100-6-50Z20TS3 为例）

■ 在不发生碰撞的情况下，调整运动姿态，尽量减少过渡点

不同机型的机器人的各个关节运动能力不同，当机器人末端安装了末端执行器后，运动过程中夹具可能会和外部环境发生干涉。此时首先考虑的不应该是增加过渡点以避免干涉，以 SCARA 机器人为例，J4 轴运动速度远远大于 J1 和 J2 轴，此时应考虑能否充分利用第四轴的运动能力，使机器人在运动过程中，姿态发生较大的变化时不会与环境发生干涉。

■ 无法减少过渡点时，应该选取合适的过渡点

过渡点位请尽量选过渡前后两段运动轨迹的方向变动尽量小、且两段过渡轨迹的长度大致相同的地方。



（a）推荐的过渡点位 （b）不推荐的过渡点位

图 1-10 过渡点位类型

根据实际情况选择合适的过渡点后，在不干涉的情况下，尽可能提高过渡长度。

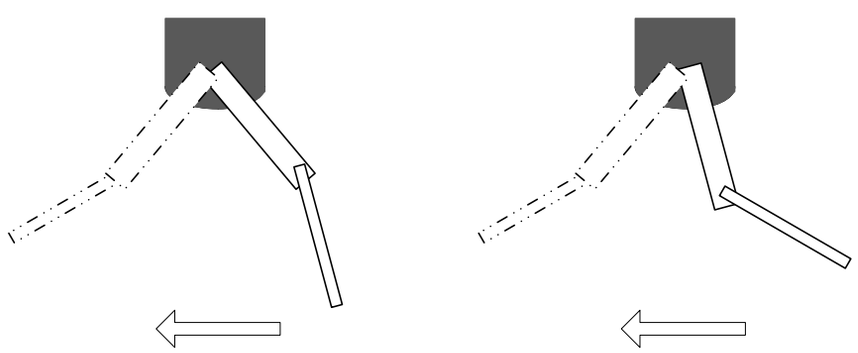
NOTE

◆ “过渡长度”在示教器上设置：【设置】-【运动参数】-【过渡精度】。

◆ 对于运动距离较短且方向变化较大的过渡段，建议采用直线过渡，其它场合建议采用关节过渡。具体使用哪种过渡方式是以现场调试效果为准。

■ 确认好所有的目标点位之后，考虑以何种姿态到达目标点位更优

以 SCARA 机器人为例，以何种姿态（SCARA 机器人为左右手姿态）到达目标点位更优，和实际的负载、是否干涉、运动点位都相关，请以现场实际调试效果为准。



注：同一目标点不同姿态的到达方式运动时间可能会不同。

■ 在无法过渡的点位，如果没有到位精度的要求，则不必检测到位精度。

经过上述五个步骤之后，如果依然无法获得满意的节拍，则可以进入到下一个环节，即加速度参数优化。

1.2 SCARA 机器人本体精度调试

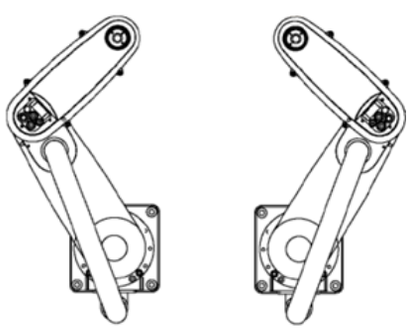
工业机器人静态性能的两个主要评价指标为 : 重复定位精度和绝对定位精度。一般情况下，工业机器人的重复定位精度都比较高，SCARA 机器人的重复定位精度为 ±0.01~±0.03mm，六轴机器人的重复定位精度为 ±0.02~±0.05mm，重复定位精度较容易满足规格要求，但对于绝对定位精度，不同厂家机器人之间差异较大，主要受测量仪器、机器人标定算法以及机器人运动学算法的影响。

由于某些操作或者其他原因导致 J2 关节发生碰撞，J2 关节的机械零点位置将发生偏移，会影响机器人的绝对位置精度。下面介绍如何验证机器人的本体精度以及 J2 关节发生碰撞后机器人本体零点标定。

1.2.1 SCARA 机器人本体精度验证

1 基于视觉左右手验证精度

第 1 步： 非使能状态下，将 SCARA 机器人摆到如图 1-4 的左手（或右手）姿势，基于视觉检测安装于机器人第四轴的特征，记录像素坐标，然后切换到直角坐标系中，示教该点为 P[0]，复制该点坐标并示教为 P[1]，如图 1-5 所示。



（a）左手姿势　　（b）右手姿势

图 1-11 SCARA 机器人左右手姿势

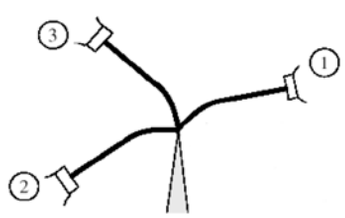
变量名	J1/X	J2/Y	J3/Z	J4/A	J5/B	J6/C	坐标系	工具号	用户号
P[000]	392.883	-53.522	-0.000	0.835	0.000	0.000	2	0	0
P[001]	392.883	-53.522	-0.000	0.835	0.000	0.000	2	0	0

图 1-12 示教位置变量

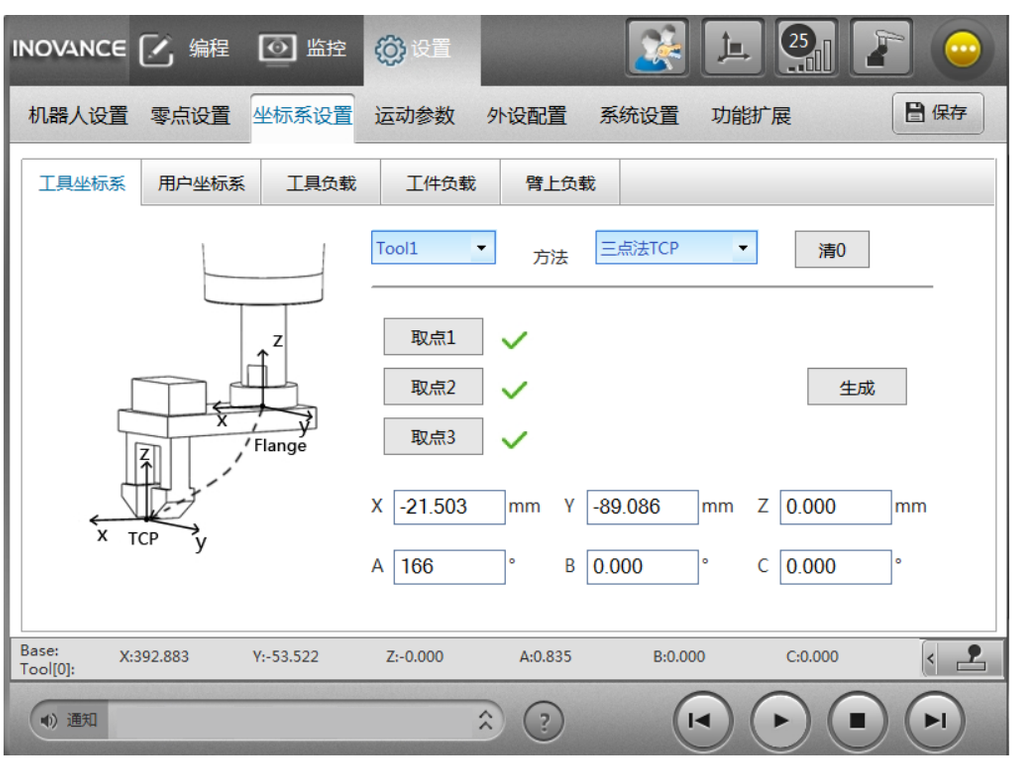
第 2 步： 非使能状态下，将 SCARA 机器人摆到第 1 步中的相反姿势，然后双击示教点 P[1]，取当前臂参数，如图 3 所示，保存 P[1]，确保 P[0] 和 P[1] 位置参数相同。运动到示教点 P[1]，记录特征点像素坐标，两像素坐标之间的距离像素乘以像素当量即可定量机器人的绝对精度。

2 基于工具拟合误差验证精度

*第 1 步：*以工具坐标系中的三点法为例，下顶针固定，分别以较大姿态差（60°以上）示教三个位置，三个位置的上顶针与下顶针对齐。



*第 2 步：*建立工具坐标系，查看工具拟合误差。



1.3 SCARA 机器人本体零点标定

SCARA 机器人本体精度主要与 J2 关节零点、大臂杆长 L1、小臂杆长 L2、减速比等参数相关，若机器人发生碰撞，其绝对位置精度会下降，因此需要在现场对机器人进行校准。

1.3.1 基于视觉—零点标定

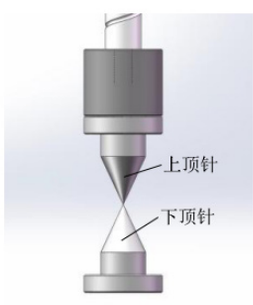
第 1 步： 使能状态下，将机器人移动到某一示教点位 P0 的左手姿势，基于视觉检测安装于机器人第四轴的特征，记录像素坐标、姿态 A，并且记录此时机器人左手姿势下的脉冲 pulse_left，如图所示。

第 2 步： 使能状态下，将机器人首先移动到示教点位 P0 的右手姿势，直角坐标系下微调机器人位置、姿态，使得视觉检测特征点像素坐标、姿态与第 1 步中一样，记录右手姿势下的脉冲 pulse_right，最终零点为（pulse_left+pulse_right）/2，输入控制器中。

第 3 步： 零点生效后，通过机器人的多点位置验证机器人本体精度。

1.3.2 基于工具—零点标定

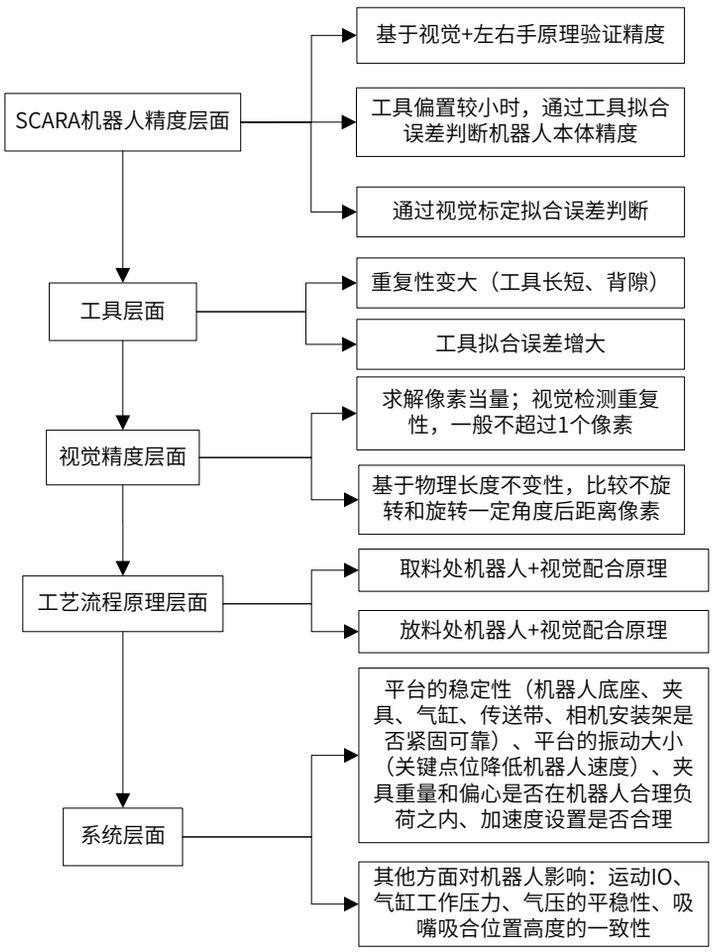
第 1 步： 选择如图所示的顶针，上顶针安装于机器人的第四轴上（无偏心状态），下顶针固定，非使能状态下，将机器人移动到左右手姿势，并使标定治具的上下顶针对齐。



第 2 步： 记录左手姿势下的脉冲 pulse_left、右手姿势下的脉冲 pulse_right，最终零点为（pulse_left+-pulse_right）/2，输入控制器中。

第 3 步： 零点生效后，通过机器人的多点位置验证机器人本体精度。

对机器人绝对精度有要求的应用现场，机器人一般与视觉联合使用，其总体精度环节相互耦合，因此对机器人和视觉应用现场精度解耦进行了梳理，如下所示：



第 2 章 视觉标定

2.1 功能简介

视觉标定是机器人系统视觉功能使用的前提，标定是为了获得相机与机器人的相对位置关系，以便后续其它操作可以将像素坐标转到机器人坐标。我司提供 SCARA 机器人与六关节机器人，两者在视觉标定功能的操作步骤上存在如下区别：

标定流程	SCARA 机器人视觉标定 - 步骤	六关节机器人视觉标定 - 步骤
标定末端治具	-	标定末端治具
标定用户坐标系	-	标定用户坐标系（使用已标定的末端治具）
标定界面参数设置	进入视觉标定界面，选择标定方式、相机安装方式等参数	进入视觉标定界面，选择标定方式、相机安装方式等参数
选择基准点	选择两点基准点	选择三点基准点
九点示教标定	进行九点示教标定	进行九点示教标定
完成	完成	完成

视觉标定主流程如下图所示：

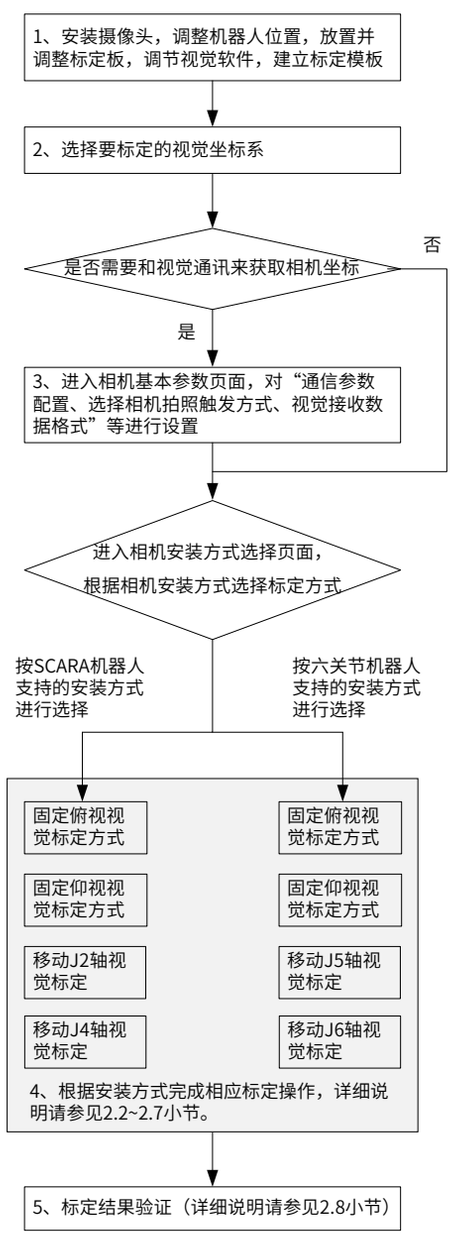


图 2-1 视觉标定主流程

2.1.1 SCARA 机器人视觉标定

SCARA 机器人视觉标定，主要包括固定俯视视觉标定、固定仰视视觉标定、移动 J2 轴视觉标定及移动 J4 轴视觉标定。所有控制器版本均支持 SCARA 机器人视觉标定。

由于其结构限制，SCARA 机器人一般作业于与基平面平行的用户面，其标定过程无需关联用户坐标系，且基准点示教中仅需选取 2 个基准点。SCARA 机器人视觉标定流程如图 2-2：

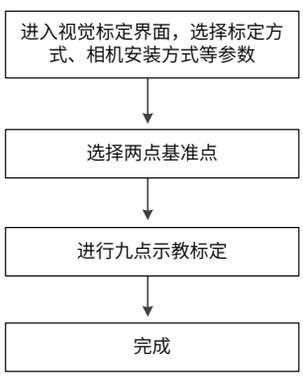


图 2-2 SCARA 机器人视觉标定流程

2.1.2 六关节机器人视觉标定

六关节机器人视觉标定，主要包括固定俯视视觉标定、固定仰视视觉标定、移动 J5 轴视觉标定及移动 J6 轴视觉标定。17 版本后的控制器可支持六关节机器人视觉标定。

六关节机器人，不仅可以像 SCARA 机器人一样，作业于与基平面平行的用户面，还可以作业于与基平面不平行的用户平面（但 Z 方向必须朝上），其标定过程中需要选择关联的用户坐标系，在选择关联的用户坐标系时，需要确认工具和用户坐标系都已先后做过标定。在基准点示教中需要选取 3 个基准点以确定标定工具（注意：每 2 个基准点之间至少需改变 2 个方向姿态，且姿态需变化 20°以上）。六关节机器人视觉标定的流程如图 2-3:

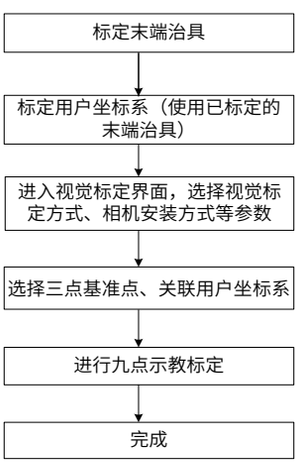


图 2-3 六关节机器人视觉标定流程

NOTE

◆ 在标定过程中，需要注意视觉标定误差的引入。影响相机标定精度的因素主要有人为调点精度、工装治具精度、视觉检测精度、本体绝对精度、标定算法精度。

2.1.3 操作步骤

根据相机的安装方式，采取不同的标定方法，详细操作请参见 2.2~2.7 小节。

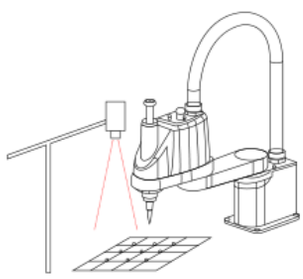
固定俯视标定方式	第 234 页上的“2.2 固定俯视视觉标定”
固定仰视标定方式	第 240 页上的“2.3 固定仰视视觉标定”
移动 J2 轴视觉标定方式	第 241 页上的“2.4 移动 J2 轴视觉标定”
移动 J4 轴视觉标定方式	第 243 页上的“2.5 移动 J4 轴视觉标定”
移动 J5 轴视觉标定方式	第 244 页上的“2.6 移动 J5 轴视觉标定”
移动 J6 轴视觉标定方式	第 245 页上的“2.7 移动 J6 轴视觉标定”

2.1.4 结果验证

各种标定方式得到的标定结果，可以通过指令编程来验证结果，查看标定效果是否符合需求。详细请参见第 246 页上的“2.8 标定结果验证”。

2.2 固定俯视视觉标定

固定俯视：指的是相机俯视安装在标定板上方，如下图所示：



1 安装条件

图中摄像头正下方有一块九宫格的标定板，保持摄像头、机器人、标定板三者在同一水平面上，标定板包含 9 个需要机器人标定的标志点。确定标定板上的 9 个标志点在视觉镜头能够完全成像。在机器人的末端安装治具，可以为尖端治具，也可以为视觉可识别的治具。

2 标定步骤

对于六关节机器人而言，在开始下列步骤前，需要先①标定末端治具，②使用已标定的末端治具标定用户坐标系。详细使用请参见《汇川机器人设计应用与维护手册 - 应用篇 1: 机器人编程设计与实现》第“2.4.3 节 坐标系设置”。然后再按下列步骤进行标定。

对于 SCARA 机器人，直接按下列步骤进行标定。

第 1 步： 安装摄像头，调整机器人位置，放置并调整标定板，调节视觉软件，建立标定模板等。

第 2 步： 选择要标定的视觉坐标系，如下图所示：

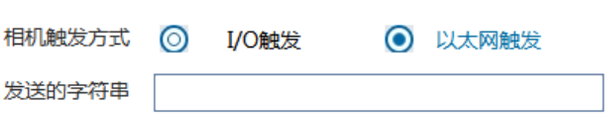
*第 3 步：*选择【视觉标定】按钮，进入相机基本参数页面：

① 通讯参数配置：设置视觉的 IP 地址信息，包含相机 IP、端口号、相机名（可不设置）。完成设置后点“连接”即可建立通讯。

② 相机拍照触发方式：可选 I/O 触发、以太网触发。

■ 当选择 I/O 触发时，需要配置如下参数：

■ 当选择以太网触发方式时，需要设置触发相机拍照的字符串，如下：



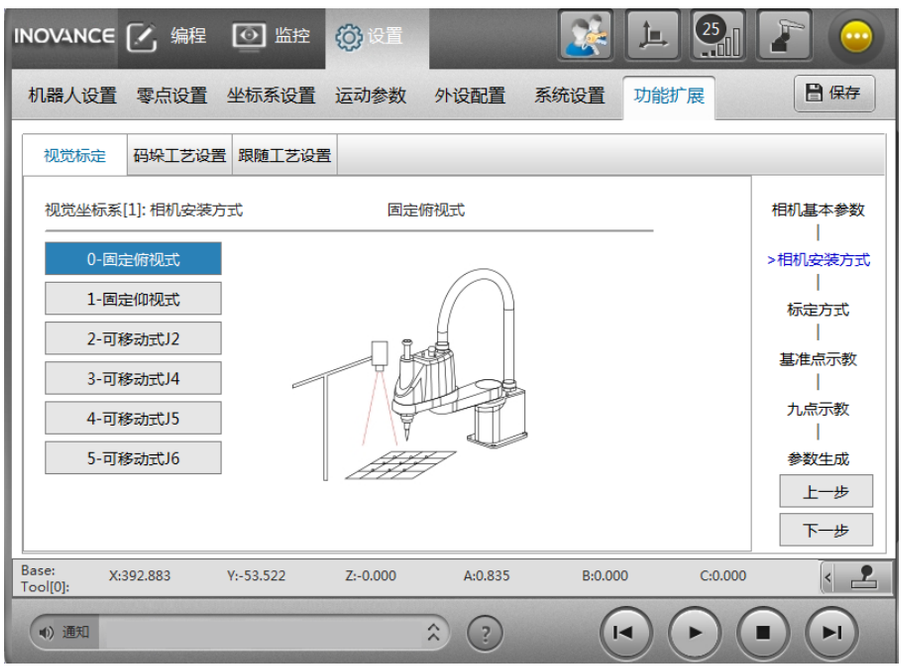
③ 视觉接收数据格式：包括帧头、分隔符、结束符。

如果不需要和视觉通讯来获取相机坐标，选择“下一步”即可。

3 相机通讯测试

用于测试与视觉软件之间的通讯，由机器人控制器向视觉端触发信号（IO 信号或者发送字符串），视觉端发送的字符串将会显示在数据接收区，查看发送过来的数据格式，是否与指定的格式一致。

*第 4 步：*进入相机安装方式选择页面，选择“固定俯视式”，选择“下一步”；



*第 5 步：*选择标定方式，选择完毕后点击“下一步”；

标定方式分为三种：

1） 手动标定：需要手动示教机器人在标定板中标定 9 个标志点，并获取对应的相机坐标。

2） 半自动标定：通过设定托盘 3 个标志点，以标定托盘的方式生成 9 个标志点，机器人自动运行到 9 个标志点，并获取对应的相机坐标。

3） 全自动标定：通过示教视野中心点以及设置 9 个点之间的距离，自动生成 9 个标志点，机器人自动运行到 9 个标志点，并获取对应的相机坐标。

第 6 步: 进入基准点的获取页面，图 a 表示 SCARA 基准点示教，图 b 表示六关节基准点示教，获取完毕后点击“下一步。

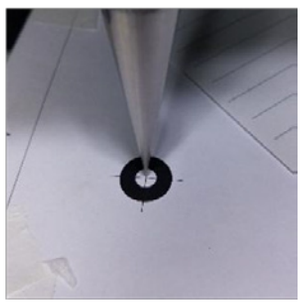
（图 a）

（图 b）

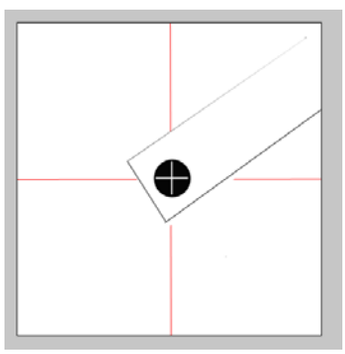
注意：在 SCARA 上需示教两个基准点，在六关节上需示教三个基准点。请根据选择的标定方式设置基准点。

1） 手动标定：若机器人的末端标定治具为尖端治具，则示教机器人用治具尖端对准标定板中视野中心的标志点，SCARA 机器人进行基准点示教时需要示教 2 个基准点，首先获取机器人位置作为基准点 1，调整机器人姿态，旋转一定角度后重新对准标志点，获取机器人位置作为基准点 2。

区别于 SCARA，六关节需要示教 3 个基准点（注意：每 2 个基准点之间至少需改变 2 个方向姿态，且姿态需变化 20°以上），步骤同于 SCARA。(注：示教的三个基准点尽量改变不少于两个方向姿态的进行示教，姿态间角度间隔尽量大)。此外六关节视觉标定中需要选取关联的用户坐标系，此用户坐标系为机器人带末端标定治具标定获得。



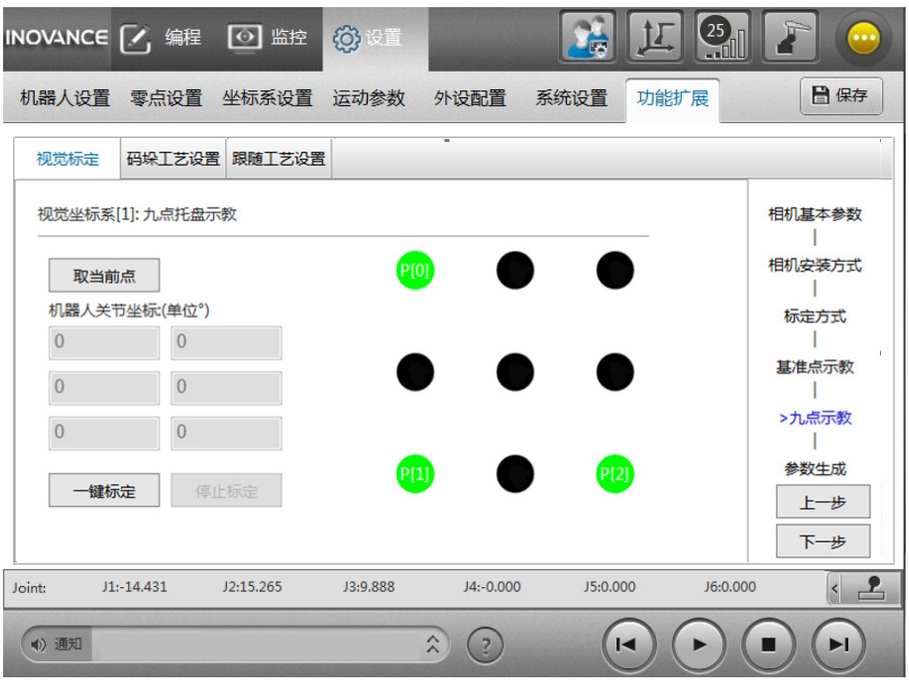
2） 半自动标定、全自动标定：若机器人的末端标定治具为视觉可识别的模板工具，可调整机器人姿态，旋转标定工具一定角度后获取其余基准点。



第 7 步： 进入九点示教界面，请根据选择的标定方式进行获取，获取完毕后点击“下一步”；

1） 手动标定：可以通过通讯方式获取单个像素坐标，也可以通过手动输入的方式获取单个像素坐标。区别是：勾选了“获取相机坐标（单位：像素）”前面的选项时，会在获取机器人坐标时通过 TCP 通讯自动获取相机坐标；不勾选“获取相机坐标（单位：像素）”前面的选项时，需要手动输入像素坐标。调节机器人位置，通过手动标定取基准点的方式【参考第 6 步 1）】，获取机器人在标定板上的 9 个位置以及输入其对应的 9 个像素坐标。

2） 半自动标定：通过自动标定获取基准点的方式【参考第 6 步 2）】，在视野中图中对应位置获取 P[1]、P[2]、P[3] 三个机器人位置，选择“一键标定”，则机器人会自动运行到视野中对应的九个位置，保存机器人坐标以及对应的像素坐标，直到运动完成。



3） 全自动标定：通过自动标定获取基准点的方式【参考第 6 步 2）】，在视野中图中对应位置获取 PO 机器人位置，计算在视野中视野范围最小值对应机器人坐标系下的距离，计算并输入点与点之间的间距（单位：mm），选择“一键标定”，则机器人会自动运行到视野中对应的九个位置，保存机器人坐标以及对应的像素坐标，直到运动完成。

第 8 步： 进入示教点列表页面，检查每个机器人的位置坐标以及像素是否异常，也可以双击列表中点进行点数据修改，点击“下一步”；

第 9 步： 系统计算并生成视觉坐标系标定矩阵以及标定结果参数，如下图所示：

其中包括误差分析：X 方向平均误差、Y 方向平均误差、X 方向最大误差、Y 方向最大误差以及 X 方向单位像素尺寸、Y 方向单位像素尺寸、标定工具 X 方向偏移、标定工具 Y 方向偏移等参数。

其中标定工具 X 方向偏移、标定工具 Y 方向偏移可作为当前标定工具的工具参数。

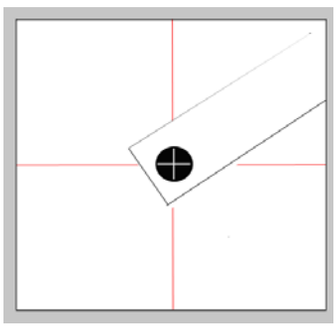
第 10 步： 选择“完成”，完成视觉标定操作。

2.3 固定仰视视觉标定

固定仰视：指的是相机仰视安装在机器人下方，如下图所示：

1 安装条件

图中摄像头安装在机器人正下方，方向朝上，保持摄像头、机器人在同一水平面上，在机器人的末端安装治具，为视觉可识别的治具。调节摄像头焦距以及机器人的高度，确保治具上的标志点能够在视野中清晰识别。



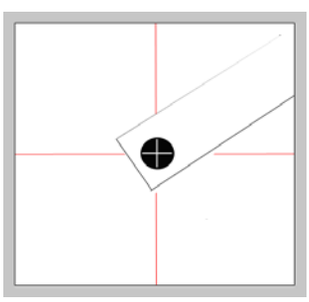
2 标定步骤

对于六关节机器人而言，在开始下列步骤前，需要先①标定末端治具，②使用已标定的末端治具标定用户坐标系。详细使用请参见《汇川机器人设计应用与维护手册 - 应用篇 1: 机器人编程设计与实现》第“2.4.3 节 坐标系设置”。然后再按下列步骤进行标定。

对于 SCARA 机器人，直接按下列步骤进行标定。

第 1~5 步： 详细操作与*“2.1 固定俯视视觉标定”*中第 1~5 步一致，分别完成【相机基本参数】、【相机安装方式】、【标定方式】参数的设置。

第 6 步： 对于 SCARA，在视野中把治具上的标志点移到视野中心，如上图所示位置。获取基准点 1，调整机器人姿态，旋转治具方向，然后把标志点移动到视野中心，获取基准点 2，六关节需获取 3 个基准点（注意：每 2 个基准点之间至少需改变 2 个方向姿态，且姿态需变化 20°以上），选择标定台关联的用户坐标系，参照*第 234 页上的“2.2 固定俯视视觉标定”*。



第 7 步： 详细操作与*“2.1 固定俯视视觉标定”*中第 7 步一致，完成九点标定的操作。其中手动标定取点方式，同步骤 6：操作机器人、移动治具标志点、在视野中均匀获取 9 个点，尽量分布在视野的九宫格范围内，并同时获取对应的视觉坐标，可以通过手动输入或者自动获取的方式获得。

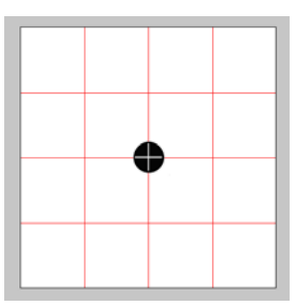
第 8~10 步： 详细操作与*“2.1 固定俯视视觉标定”*中第 8~10 步一致。

2.4 移动 J2 轴视觉标定

移动 J2 轴标定，指的是相机俯视安装在机器人 J2 轴手臂上（仅 SCARA 机器人支持，六关节机器人不支持），如下图所示：

1 安装条件

相机安装向下在 J2 轴手臂上，保持相机平面与水平面平行，无标定治具，如下图所示，图中黑色标志点位标定板上的标志点，红色线条代表摄像头视野中的网格线，网格线的交叉点位所需的标志点。



2 标定步骤

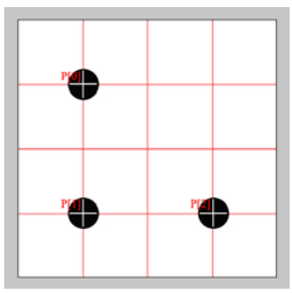
第 1~5 步： 详细操作与 *“2.1 固定俯视视觉标定”* 中第 1~5 步一致，分别完成【相机基本参数】、【相机安装方式】、【标定方式】参数的设置。

第 6 步： 移动机器人调整相机姿态，对准中心标志点，获取两个不同的机器人点作为基准点。

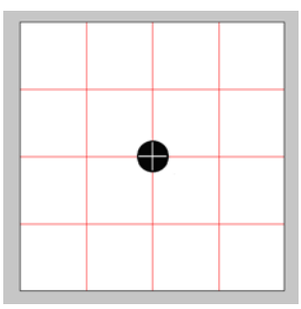
第 7 步： 完成九点标定的操作。

手动标定：移动机器人，分别用视野中九个标志点对准标定板上的标志点，分别获取机器人坐标以及像素坐标，其中相机坐标可以自动获取，也可以手动输入。

半自动标定：移动机器人，分别用视野中的 P[0]、P[1]、P[2] 对准标定板上的标志点，完成托盘三点示教后，点击“一键标定”，完成九点标定操作。



全自动标定：移动机器人，把视野中心对准标定板上的标志点，输入视野网格长度对应机器人坐标系的长度（大致即可，确保自动生成的 9 个点不超出视野范围）。点击“一键标定”，完成九点标定操作。



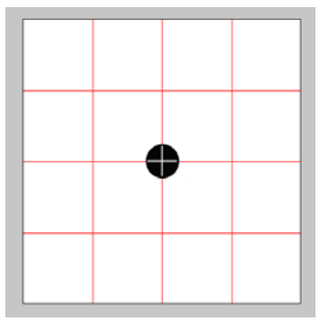
第 8~10 步： 详细操作与*“2.1 固定俯视视觉标定”*中第 8~10 步一致。

2.5 移动 J4 轴视觉标定

移动 J4 轴标定：指的是相机俯视安装在机器人 J4 轴手臂上（仅 SCARA 机器人支持，六关节机器人不支持），如下图所示：

■ 安装条件

相机安装向下在 J2 轴手臂上，保持相机平面与水平面平行，无标定治具，如下图所示，图中黑色标志点位标定板上的标志点，红色线条代表摄像头视野中的网格线，网格线的交叉点位所需的标志点。

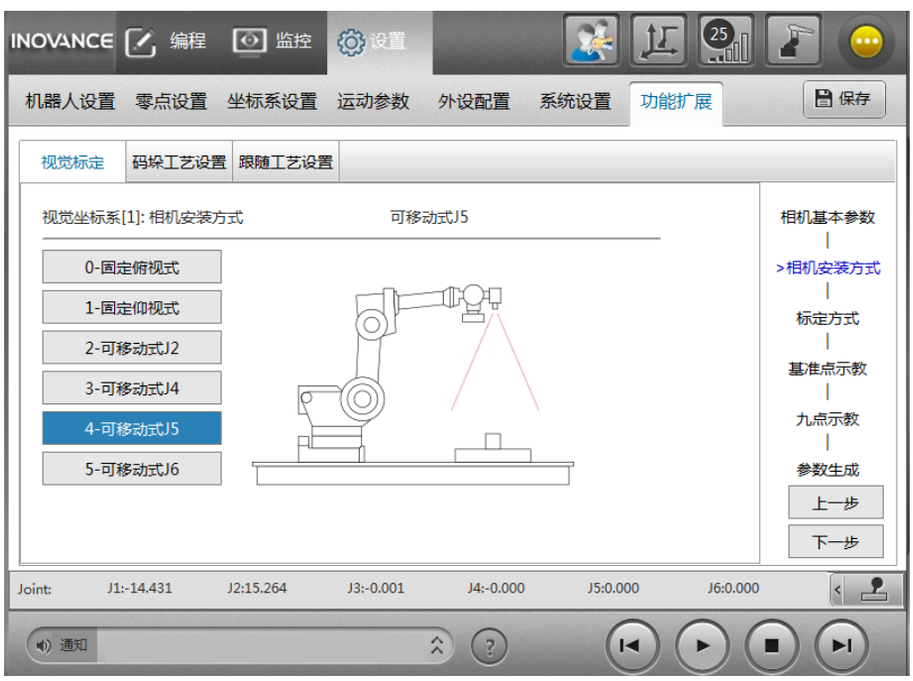


■ 标定步骤

详细操作与*“2.4 移动 J2 轴视觉标定”*的标定步骤一致。

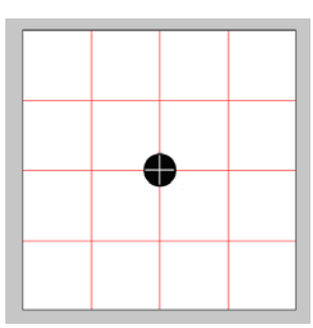
2.6 移动 J5 轴视觉标定

移动 J5 轴标定：指的是相机俯视安装在机器人 J5 轴手臂上（仅六关节机器人支持，SCARA 机器人不支持），如下图所示：



■ 安装条件

相机安装向下在 J2 轴手臂上，保持相机平面与水平面平行，无标定治具，如下图所示，图中黑色标志点位标定板上的标志点，红色线条代表摄像头视野中的网格线，网格线的交叉点位所需的标志点。



■ 标定步骤

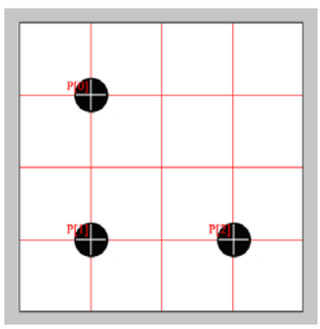
*步骤 1-5：*详细操作与*第 234 页上的“2.2 固定俯视视觉标定”*步骤 1-5 一致，分别完成【相机基本参数】、【相机安装方式】、【标定方式】参数的设置。

*第 6 步：*移动机器人调整姿态，使末端工具对准中心标志点，获取 3 个不同的机器人点作为基准点，并选择关联的用户坐标系，基准点和用户坐标系的选择原则请参见*第 234 页上的“2.2 固定俯视视觉标定”*。

*第 7 步：*选择【下一步】，进入九点标定页面。

7-1 手动标定：移动机器人，分别用视野中九个标志点对准标定板上的标志点，分别获取机器人坐标以及像素坐标，其中相机坐标可以自动获取，也可以手动输入。

7-2 半自动标定：移动机器人，分别用视野中的 P[0]、P[1]、P[2] 对准标定板上的标志点，完成托盘三点示教，如下图所示。



选择【一键标定】按钮，完成九点标定操作。

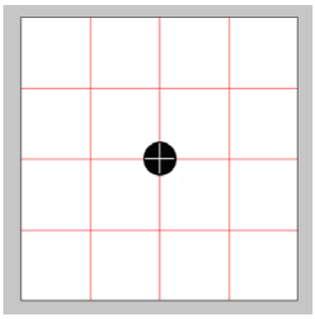
7-3 全自动标定：移动机器人，把视野中心对准标定板上的标志点，输入视野网格长度对应机器人坐标系的长度（大致即可，确保自动生成的 9 个点不超出视野范围）。

2.7 移动 J6 轴视觉标定

移动 J6 轴标定：指的是相机俯视安装在机器人 J6 轴手臂上（仅六关节机器人支持，SCARA 机器人不支持），如下图所示：

■ 安装条件

相机安装向下在 J2 轴手臂上，保持相机平面与水平面平行，无标定治具，如下图所示，图中黑色标志点位标定板上的标志点，红色线条代表摄像头视野中的网格线，网格线的交叉点位所需的标志点。



■ 标定步骤

详细操作与*第 244 页上的“2.6 移动 J5 轴视觉标定”*的标定步骤一致。

2.8 标定结果验证

当视觉标定带有标定治具时，如果需要用标定过程中的治具作为验证工具，请进入【设置】-【坐标系设置】-【工具坐标系】-【直接输入法】进行设置。

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。

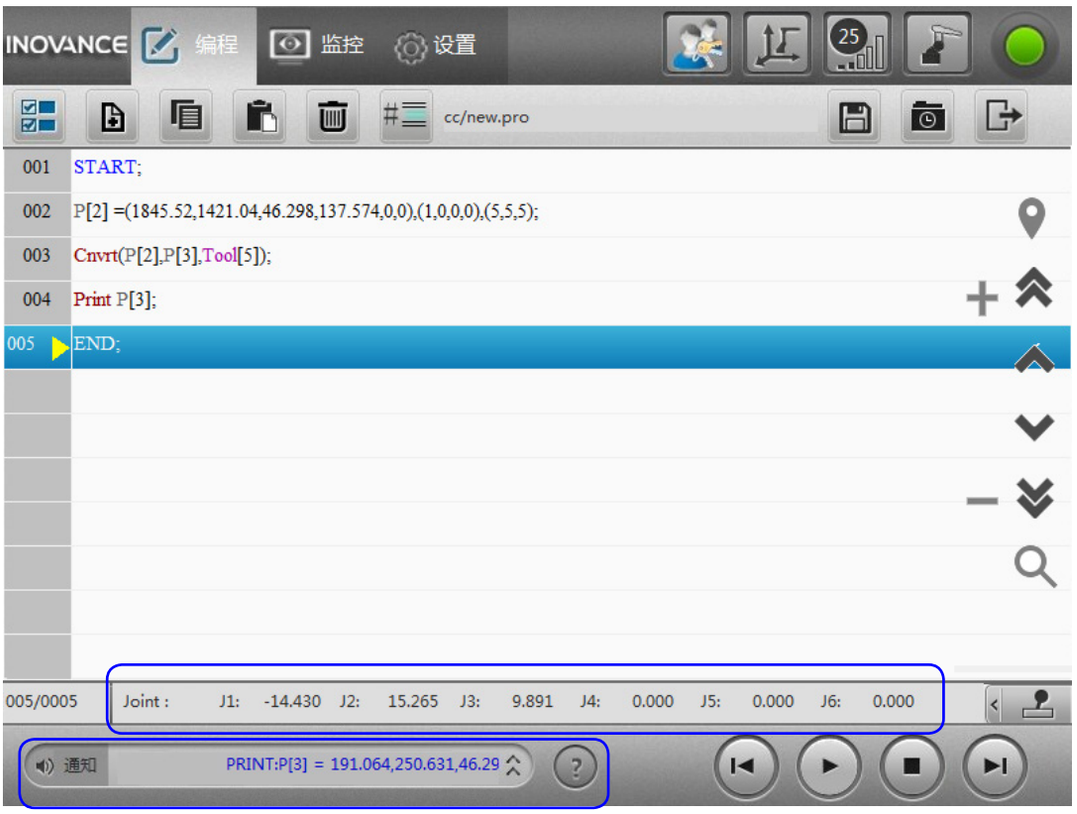
以下给出一个完整的视觉标定编程的范例。

```
001 START;
002 P[2] =(1845.52,1421.04,46.298,137.574,0,0),(1,0,0,0),(5,5,5);
003 Cnvrt(P[2],P[3],Tool[5]);
004 Print P[3];
005 END;
```

P[2] 点输入的坐标是视觉坐标系中的像素坐标，其中坐标系参数中的（5,5,5）分别表示：第一个表示固定相机坐标，第二个表示在视觉标定的治具作为工具 5 的参数输入到工具坐标系 5 中。第三个表示视觉坐标系号为 5. 其中臂参数 (1,0,0,0) 为拍照点机器人的臂参数

Cnvrt 指令说明：将 P[2] 点转换到工具坐标系下的点存储于 P[3] 中。

“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。其中转换后的工具坐标的 A 值 -140.896 与机器人拍照点工具坐标 A 值 137.574 不一致，是因为视觉坐标系的方向、机器人坐标系的方向、以及工具方向不一致导致的。

其中 A 表示视觉坐标系下的物体的旋转角，转换城工具坐标系下的角度为 A1，则

A1 = A +A’ ，A’ 表示转换角度差

A2 = 机器人坐标系方向和相机坐标系方向的角度差

A3 = 机器人坐标系和工具夹爪方向或者货物抓取方向角度差。

A4 = 机器人拍照点工具坐标系下的角度

机器人 A 4= A + A’ + A2 + A3 或者 A4 = A1 + A2 +A3.

上表中，已知 A4 = 137.574，A = 137.574, A1 = -140.896，则可知晓：

A2 + A3 = A4-A1 = 137.574 – (-140.896) = 278.47 或者

A’ + A2 + A3 = A4-A = 137.574 – 137.574 = 0

所以当给定一个像素坐标 A, 即可求出拍照点的机器人的坐标 A4 = A + A’ + A2 + A3 = A + 0;

或者由像素坐标 A 通过转换 CNVRT 智力得到其对应的工具坐标系下的值 A1，则

A4= A1 + A2 +A3 = 278.47 + A1.

当视觉坐标系为随动相机坐标系时，选择【视觉基准点】选项框，并输入拍照点 P[3].

第 3 章 跟随应用

3.1 概述

跟随工艺是跟踪移动的物体，以移动的物体为参考规划机器人的运动，传送带跟随是跟随工艺中的一种典型应用。传送带跟随是通过视觉或光电传感器检测到传送带上的物体，机器人在物体移动过程中对其进行准确抓取。

3.1.1 工作流程

跟随工艺工作过程主要分两部分：检测和跟随运动。检测是通过视觉或光电传感器获得物体在传送带上的位置，将获得的物体位置信息放入传送带对象队列中，并通过编码器反馈跟踪其位置的变化；跟随运动是以队列中移动物体为参考坐标系的机器人运动。检测与跟随运动并行处理，具体流程如下图所示。

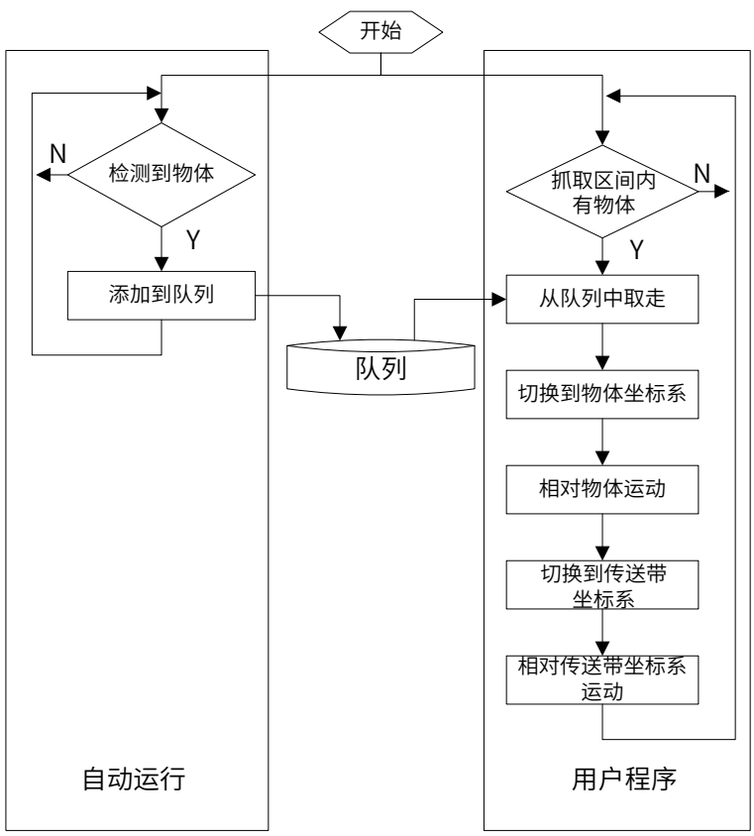


图 3-1 传送带跟随工艺处理流程

3.1.2 跟随坐标系描述

跟随运动以物体坐标系为参考系，跟随工艺中各坐标系介绍如下：

1） 传送带坐标系（Ocny）：用于描述传送带在机器人基坐标系下的位置和方向，传送带坐标系是静态的。

2） 物体坐标系（Oobj）：用于描述物体在传送带上的位置和方向，是物体基准点在传送带坐标系下的坐标。它固连在物体上，随物体移动，是动态的。

3） 视觉坐标系（O_{vis}）：描述了相机与传送带之间的位置转换关系。通过相机获得了物体在传送带上的瞬时位置，通过编码器实时获得物体在传送带上的运动量，从而实现机器人对物体位置的实时追踪。

跟随工艺中各坐标系关系如下：

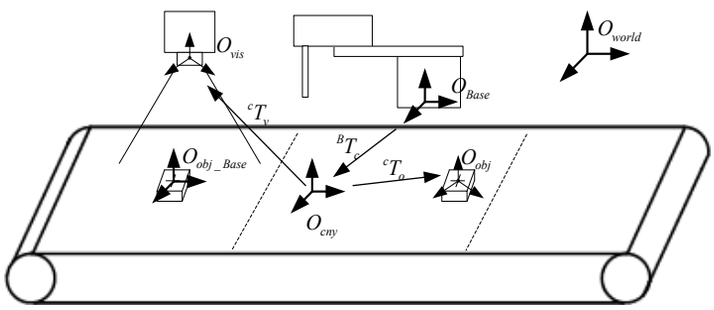


图 3-2 跟随工艺各坐标系关系

3.1.3 功能规格

1、传送带类型：直线、圆盘。

2、传送带数量：可配置 4 条，最多同时使用 2 条。

3、检测方式：视觉或光电传感器，最多使用一个视觉。

4、跟随运动指令：仅支持 Movl，Movc，JumpL（即在跟随开启前后（Refsys 之间），只能使用这些运动指令）。

5、工件种类数：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、视觉通讯格式：固定格式（参考后文）。

8、一次拍照中物体个数：0-10 个。

9、物体存储队列长度：500 个。

3.1.4 配置流程

使用跟随工艺时需要以下几部分工作，各部分详细内容参考后文相关章节：

1） 硬件物理连接；

2） 传送带坐标系标定；

3） 传送带参数设置；

4） 用户程序编写。

3.2 硬件配置

跟随工艺硬件配置如下图，其中光电传感器和视觉系统根据具体采用的检测方式可选配。如果使用视觉检测，相机触发必须采用硬件触发的方式。

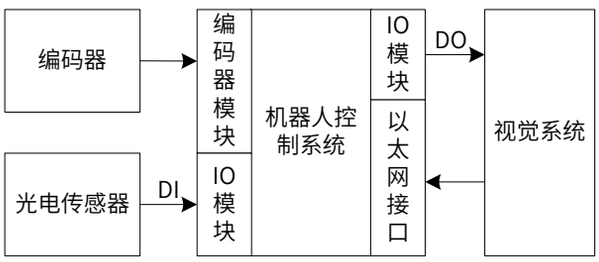


图 3-3 跟随工艺系统框架图

3.2.1 编码器

需要采用脉冲型编码器，将编码器信号线和电源线接到机器人控制柜中的编码器接口上，具体接线逻辑参考电气接线手册。物理连接完成后转动编码器，通过下图界面取编码器的值，观察编码器值是否变化判断物理连接是否正确。

3.2.2 光电传感器

光电传感器在被物体触发时输出不同的电平信号，机器人控制器通过信号边沿检测物体，将传感器信号线接到控制柜 IO 模块的普通 DI 接口上，接线要求参考电气接线手册。接线完成后可通过监控界面观察对应的 DI 是否变化判断传感器工作是否正常。

3.2.3 视觉

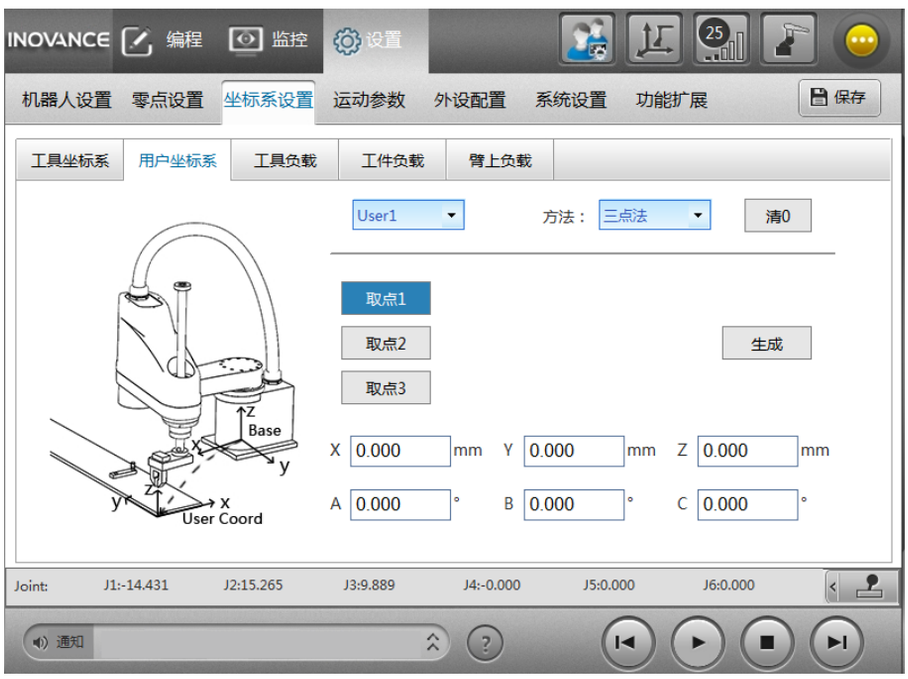
视觉与机器人之间的交互包括 DO 触发和以太网通讯两部分。其中 DO 触发需要从机器人控制柜 DO 接口处引出信号线，与相机的 IO 触发接口相连。物理连接完成后应手动输出 DO，观察相机是否可以正常触发。

3.3 坐标系设置

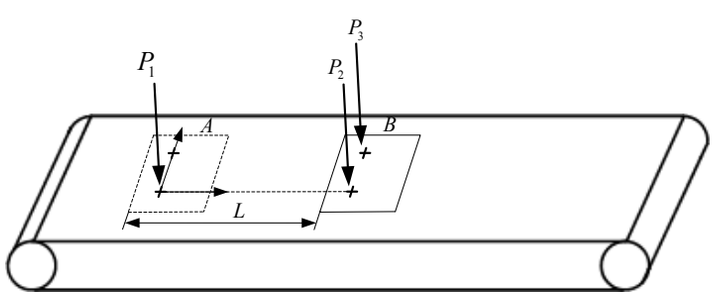
传送带坐标系是作特殊功用的用户坐标系，需要在【设置】-【坐标系设置】-【用户坐标系】进行坐标系参数设置或标定。传送带坐标系描述可传送带的位置和方向，其中 x 轴的正方向与传送带移动方向重合。

■ 直线型传送带坐标系设置

如果是直线传送带，使用“三点法”，设置界面如下：



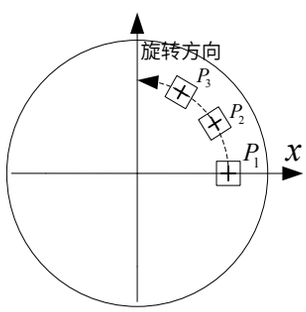
三个点的选取如下图所示，将标定板或其他标定基准放置在传送带上，使用机器人末端对齐基准点，取点 1，点 1 即传送带坐标系的原点；保持标定板在运动过程中与传送带位置不变，使 P1 移动到 P2 位置，P1-P2 为 x 正方向；Z 轴垂直于传送带向上，根据右手定则取 xy 第一象限内的点 P3。



■ 圆盘型传送带坐标系设置

如果是圆盘形传送带，需要使用旋转法，标定界面如下图所示：

旋转法同样是取三个点，如下图所示，将标定基准点在圆盘上固定，将基准点旋转到合适位置后取第一点 P1，旋转中心到 P1 的方向为坐标系 x 轴正方向。逆时针转动圆盘，依次取 P2，P3 点，三点确定一个圆弧，圆弧中心即坐标系原点。



NOTE

◆ 为了提高精度，建议三个点之间的距离尽量远。末端有工具时需要先标定工具坐标系，且能与工具坐标系编号对应。

3.4 参数设置

传送带参数包括基本参数、编码器校准、工件高度校准、边界参数设置、检测参数设置几部分，其中根据检测方式的不同检测参数分为视觉参数或传感器参数。传送带参数在【设置】-【功能扩展】-【跟随工艺设置】页面下设置，主界面如下：

■ 界面参数说明

编辑：勾选某个传送带编号之后，可以查看或编辑这个传送带的参数；

使能：勾选后，在用户程序中可以使用该传送带；

基本参数：显示编辑选中传送带的基本信息，此处只能显示不能修改；

位置补偿参数：可以对抓取位置进行整体微调；

参数设置按钮：点击后，进入参数编辑页面。

3.4.1 基本参数设置

点击主界面“参数设置”后，进入参数设置界面，如下图所示。参数设置采用向导式界面，点击右下角的“上一步 / 下一步”可切换界面，最后一个界面“完成”按钮代替“下一步”。切换界面时参数被临时生效，但未固化存储到控制器，断电后会丢失。如果想永久保存参数，需要点击“保存”或“完成”按钮。

■ 界面参数说明

传送带类型：选择直线型或圆盘形。

编码器通道：选择传送带编码器在机器人控制器上的信号输入通道编号。

关联用户坐标系：选择标定得到的用户坐标系编号，将用户坐标系与传送带关联起来。

检测方式：选择传感器或视觉。

3.4.2 编码器校准

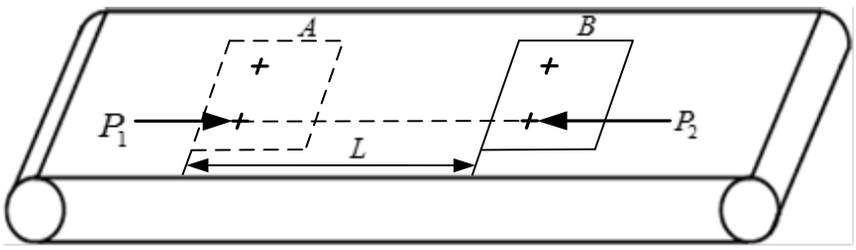
编码器校准界面完成编码器分辨率和方向的设置。编码器分辨率是传送带移动单位长度（mm 或 rad）时编码器的脉冲增量。编码器方向指传送带正向移动时编码器脉冲值的增或者减。

具体标定过程如下：

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】，完成分辨率和方向的自动计算。



3.4.3 工件高度设置

第三个界面是“工件高度校准”，如下图所示。视觉或传感器都不能检测到工件的高度，需要通过参数设定指定每种工件类型的高度。工件高度值是工件基准点（物体坐标系原点）在传送带坐标系下的 Z 坐标值。

具体标定方法如下：

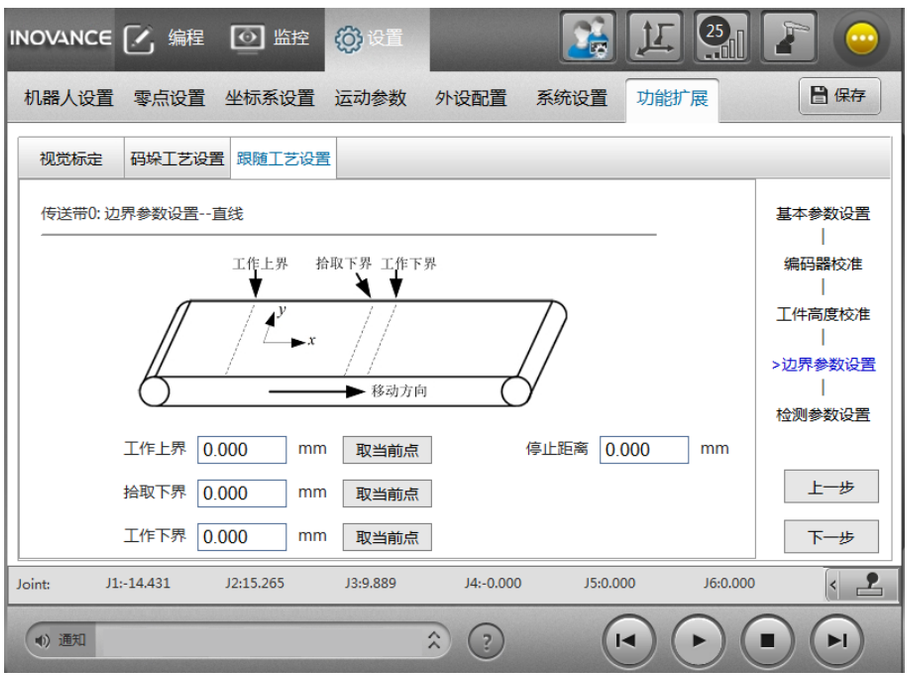
1） 点击工件列表选择需要设置的工件编号。

2） 将前面用户坐标系标定时采用的工具末端移动到工件基准点，点击【取当前高度】完成高度自动读取。如果机器人末端装有工具，需要使能对应的工具坐标系编号。

3.4.4 边界参数设置

边界参数设置界面分为直线传送带和圆盘传送带的设置界面分别如图 3-4 中 a、b 所示。

工作上界、工作下界、拾取下界是垂直于传送带坐标系 x 轴方向的三条线，用 x 坐标表示。这三个参数可以直接输入，也可以将机器人移动到合适的位置，点击“取当前点”自动获取。停止距离一般设置为 5 °或 5mm。



（a）直线传送带边界参数设置界面

（b）圆盘传送带边界参数设置界面

图 3-4 边界参数设置界面

■ 界面参数说明

工作上界：机器人能够到达的最上界，同时表示物体查找区域的上边界，当物体超过该边界后就能被传送带查询指令查询到。工作上界用传送带坐标系下的 x 坐标值表示，要保证工作上界在机器人直角工作空间内，远离奇异位置。

工作下界：允许机器人到达的最下边界，当机器人距离该边界小于停止距离时报警“机器人超出工作下界”。

拾取下界：传送带物体查找区域的下边界，当物体超出该边界且还没有被拾取时将不再抓取，物体从队列中被丢弃。动态抓取需要一定时间，已经接近工作下界的物体在抓取过程中尚未完成抓取就已经超出工作下界导致报警，适当设置拾取下界可以防止这种情况。设置拾取上界时，保证物体从拾取边界移动到工作下界的时间应该大于抓取时间。

停止距离：指物体接近工作下界时的平滑停止距离，当物体距离工作下界的距离小于该参数时开始平滑停止。

3.4.5 检测参数设置 - 传感器参数

根据所选择检测方式的不同，有传感器和视觉两种形式，使用传感器检测时，参数和设置方法如下：

传感器检测参数可分为传感器设置和传感器位置校准两部分。传感器设置是设置传感器的基本参数，包括传感器 DI 口、信号类型、工件类型、重复剔除距离几个参数。

- 传感器 DI 口 : 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。
- 重复剔除距离：在检测到一个有效信号后，在后续一段距离内如果再检测到信号变化，认为是无效信号自动剔除。

设置完上述参数后先点击“下发”，再进行传感器位置校准。

传感器位置校准是为了获得传感器被触发的瞬间，物体基准点在传送带坐标系下的位姿，即上图左侧的检测位置。为了得到检测位置，可以让物体经过传感器后停止在机器人工作空间内的某个位置，移动机器人工具末端对准物体基准点，读取示教位置即物体当前的位置，同时机器人自动记录下触发传感器后传送带的移动量，根据当前示教位置和偏移量自动计算出触发传感器时刻物体的位置，具体操作步骤如下：

将传送带速度调节到正常工作的速度，物体放置在传感器上游，物体随传送带移动，经过传感器时传感器被触发，物体移动到机器人工作空间后停止传送带 , 保证该过程中传感器只被触发过一次。

移动机器人工具末端对准物体基准点，点击“示教位置读取”，此时机器人在传送带坐标系下的位置和姿态就是物体坐标系的位姿态。

点击“检测位置计算”，控制器自动计算出物体触发传感器时刻物体的位姿。

NOTE

◆ 保证标定时工件经过传感器的速度与正常工作时相同。
◆ 标定过程中勿使人手或其他物体触发传感器。
◆ 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。

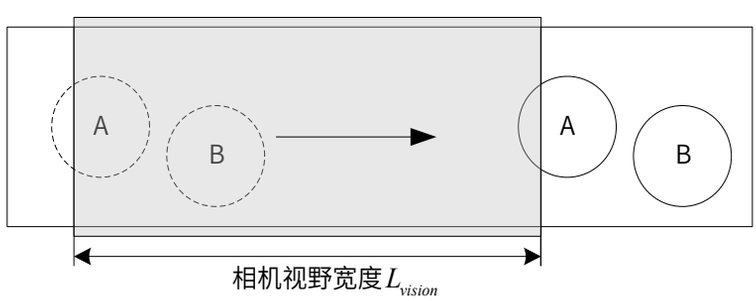
如果使用视觉检测，则需要设置视觉相关参数，视觉设置分为基本参数和视觉标定两部分，基本参数设置界面如下：

相机触发 DO: 相机硬件触发拍照的 IO 信号在 IRLINK 模块上输入端口。

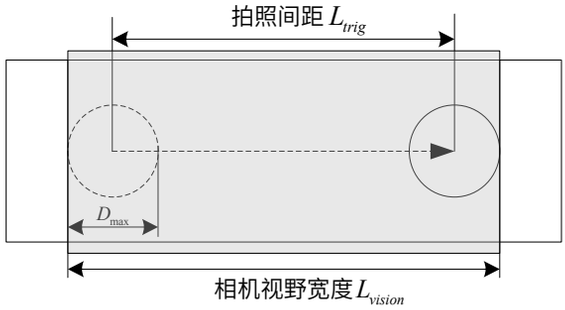
相机触发信号类型：相机被触发时，IO 模块输出信号的变化类型，上升沿或下降沿，应与相机的实际情况一致。

拍照间距：传送带视觉拍照由传送带的移动距离控制，传送带每移动一个“拍照间距”控制器发出一个 DO 信号给相机。

拍照间距的设置原则是保证能拍全传送带上的每个物体标识，保证不漏拍。如果设置的拍照间距等于相机视野宽度，相机前后两次拍到的画面无缝衔接，此时能拍全传送带上的每一部分，如果某个物体刚好位于前后两次拍照的衔接处，则前后两次各拍到此物体的一半，此物体不能被识别。如下图所示，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，前后两个画面应略有重叠，重叠宽度大于物体标识的最大宽度，实现至少有一次拍照是完整的，如下图所示。

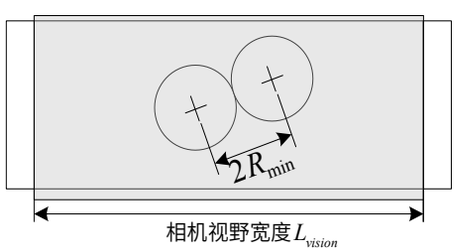


视觉必须在一个拍照周期内完成图像识别，拍照周期 = 拍照间距 / 传送带速度，拍照间距设置过小会导致拍照周期变短，要求视觉的处理速度更快，因此在满足不漏拍的前提下拍照间距尽量大。综合各方面需求，拍照间距不能太大也不能太小，可按下列公式给定，其中 L_{trig} 为拍照间距，L_{vision} 为视野宽度，D_{max} 为物体识别标识最大宽度，δ_{vision} 为设置余量。

$$\begin{cases} L_{trig} = L_{vision} - D_{\max} - \delta_{vision} \\ \delta_{vision} < 10mm \end{cases}$$

重复检测判定距离：按上述拍照间距设置原则，前后两次拍照画面有重叠，这有可能导致同一个物体在前后两次拍照中同时被识别，造成重复，此时需要剔除其中一个。控制器内部会对前后两次拍照物体的位置进行比较，如果两个物体的距离小于重复检测判定距离，认为是重复，剔除后一次拍照得到的物体。重复检测判定距离的设置原则是正确识别重叠的同一物体和正常靠近的两个物体。如下图所示，当两个物体距离大于 $2R_{\min}$ 时可认为是两个物体，否则认为是同一个物体，因此重复剔除距离按如下原则设置：

$$L_{remove} < 2R_{\min} = D_{\min}$$



相机数据类型：像素或机器人坐标，视觉发送给机器人的数据是像素时选择“像素”，否则选择“机器人坐标”。

视觉坐标系编号：如果视觉系统发送给控制器的数据为像素坐标，需要在控制器中进行视觉标定和坐标变换，视觉标定结果存储在视觉坐标系列表中，该参数指定视觉坐标系编号。

如果视觉系统发送给机器人的位置信息是像素坐标，需要在机器人控制器中进行视觉标定。完成上述基本参数设置后点击【下一步】，进入“视觉标定—视觉点获取”界面：

将标定板放置在相机下方的传送带上，在视觉端拍照获取标定板 9 个点的像素坐标，按顺序填入“视觉点获取”界面中对应的点号位置。点击”取当前位置”读取拍照时刻传送带的位置信息。

完成像素坐标输入和拍照位置读取后，点击【下一步】，进入“视觉标定—示教点获取”界面：

保持标定板与传送带的相对位置不变移动传送带，标定板随传送带进入机器人工作空间后停止传送带。按照视觉像素点的顺序依次移动机器人末端对准这 9 个点，点击“取当前点”获得 9 个点的坐标。点击“取当前位置”读取此时传送带的位置。读取传送带当前位置和 9 个点的坐标后点击“偏移”计算出 9 个点在拍照位置处的坐标。点击【标定 / 下发】计算标定结果并保存到视觉坐标系中。

完成后点击【完成】结束所有设置并保存结果。

3.5 跟随指令

3.5.1 CnvVison

■ 功能

打开 / 关闭传送带的视觉端口，开启传送带视觉后，控制器将视觉检测到的物体自动存储到队列中，无需用户对视觉坐标编程处理。

■ 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：开启传送带视觉后，视觉数据需要按固定格式发送给机器人控制器，格式如下：

有物体：TA,X1,Y1, A1,T1, TA,X2,Y2, A2,T2……;（最多 10 个）

无物体：NG;

注意其中角度单位是度，不要漏掉分号，一次拍照对应一个数据包，不允许分多次发送。

3.5.2 GetCnvObject

■ 功能

从传送带物体队列中查询拾取区域（工作上界与拾取下界之间）内是否有指定类型的物体，有则从队列中取出，没有则跳转。

■ 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：

视觉或光电传感器检测到的物体自动放入机器人控制器存储队列中，控制器实时追踪这些物体的位置，当物体进入拾取区域后，就能够被 GetCnvObject 查询到。如果拾取区域内有多个物体，则取最下游的一个。

若查找到物体，则将该物体从存储队列中取出作为目标物体，执行下一行指令（不执该句的 Goto L[***]）；若未查询到物体，则执行该句的 Goto L[***]，即跳转至 L[***] 处。

■ 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];
EndIf;
CnvVision (Conveyor[1],ON);
```

```
## 定义 P[30] 为在 1 号传送带物体的正上方 10mm 处。注意坐标系号为 7，为一个跟随传动带运动的点。
P[30]=(0,0,10,0,0,0),(0,0,0,0),(7,0,1);
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;
```

3.5.3 CopyCnvObject

■ 功能

从传送带物体队列中查询拾取区域（工作上界与拾取下界之间）内是否有指定类型的物体，有则从队列中复制出来，没有则跳转。

■ 格式

参考 GetCnvObject。

■ 说明

该指令与 GetCnvObject 格式、语法、功能基本相同，不同的是 CopyCnvObject 将队列中的物体复制出来使用，下次查找时仍然能够找到该物体；GetCnvObject 是从队列中取走物体，该物体在队列中被删除，下次不能再被查找到。

3.5.4 RefSys

■ 功能：切换运动参考坐标系。

■ 格式 1：RefSysBase;

■ 格式 2：RefSysConveyor(***,Tool[***]);

■ 格式 3：RefSysWorkBench(***,Tool[***]);

参数	意义
RefSys Base	运动参考系切换到静态
RefSys Conveyor(***,Tool[***]);	运动参考系切换到传送带上的物体坐标系。由于物体坐标系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物体同步运动。 *** 为传送带编号，范围 0~3； Tool[***], 为工具号，表示切换过程中指定的工具末端与传送带物体实现同步。
RefSys WorkBench(***,Tool[***]);	运动参考系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。 *** 为工作台编号，范围 0~3； Tool[***], 为工具号，表示切换过程中指定的工具末端与工作台同步

■ 说明

运动参考系切换到传送带、工作台等物体坐标系后，轨迹和点位的描述都是相对物体坐标系而言的。以传送带为例，执行 RefSys Conveyor 后，机器人的运动都是相对物体坐标系而言的，当没有运动指令时机器人与物体保持相对静止（与物体同步运动），有运动指令时轨迹是物体坐标系下的轨迹，目标点是物体坐标系下的坐标（类型 7）。

◆ 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。跟随过程中不能使用 WaitInPos,Print 等需要机器人静止才能执行的指令

```
START;
P[30]=(0,0,10,0,0,0),(0,0,0,0),(7,0,0);        ## 定义 P[30] 为物体坐标系下的点，坐标为（0,0,10,0,0,0）
Movj P[0],V[30],Z[0];                           ## 移动到等待位置
L[2]:
GetCnvObject(1,0), Goto L[2];                   ## 从队列中查询 1 号传送带，0 号类型的物体
RefSys Conveyor(1,Tool[2]);                     ## 将运动参考坐标系切换到刚刚查找到的传送带物体上
Movl P[30],V[100],Z[1],Tool[2];                 ## 移动到物体坐标系下的 P30, 在物体上走过的轨迹是直线
RefSys Base;                                    ## 运动参考系切换到机器人下
END;
```

3.5.5 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） 跟随模式下的运动为笛卡尔轨迹，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：（运行时会停在该句指令，这些指令需要机器人处于静止状态）

Wait IN,Print TimeStart,TimeOut

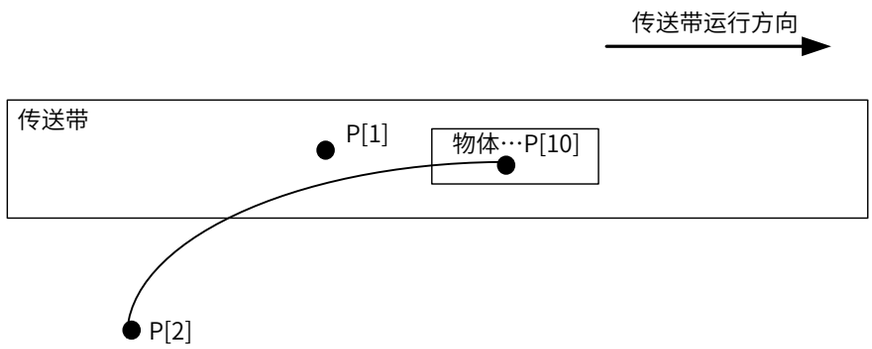
禁止使用：（若使用则导致运行错误）

Home、Until、Movj、Jump(但 JumpL 支持)、GetCurPoint、PE。

3.6 应用案例

■　点位示意图

图中，P[10] 是物体上的点，使用动态物体坐标系（类型 7），P[1] 是等待点，P[2] 是放料点位。机器人从 P[1] 点等待，有物体流过工作上界后被 GetCnvObjet 查询到，运动参考系切换到物体坐标系，机器人运动到物体坐标系下的 P[10] 点完成抓取，抓取后并抬升 10mm 离开传送带表面，再切换到传送带坐标系，运动到放料点 P[2] 完成放料。



■　程序

```
START;
B0=0;                                    ##B0 变量初始化
P[10]=(0,0,0,0,0,0),(0,0,0,0),(7,0,2);   ## 定义物体坐标系下的点 : 物体原点
P[11]=(0,0,10,0,0,0),(0,0,0,0),(7,0,2);  ## 定义物体坐标系下的点：物体上方 10mm 处
While B0<>1
Open Socket(“192.168.24.55”,1025,1026,B0);  ## 建立视觉坐标系的连接
EndWhile;
ConVision(Converyor[0],ON,1026);         ## 打开传送带视觉
L[0];                                    ## 标识 L[0]
Jump P[1],V[100],Z[0],LH[10],MH[-50],RH[10];  ## 机器人等待点，一般此点设置在工作上界附近
 L[1];                                   ## 程序跳转标识 L[1]
GetCnvObjet (0,1),Goto L[1];             ## 查询物体，有物体往下执行，没有则跳转到 L[1] 处
ResSys Converyor[0];                     ## 运动参考系切换到传送带下的物体坐标系
JumpL P[10],V[100],Z[0],LH[0],MH[10,RH[0];  ## 运动到物体坐标系下的 P[10] 处
Set Out[8],ON;                           ## 打开开关，吸附物体
Movl  P[11] ,V[30], Z[0];                ## 抬升 10mm，离开传送带表面
RefSys Base                              ## 与传送带解除同步，运动参考系切换到机器人下
Jump P[2],V[100],Z[0],LH[0],MH[10],RH[0];  ## 运行到放料点 P[2]
Set Out[8],OFF;                          ## 关闭开关，放下物体
Goto L[1];                               ## 程序循环
END
```

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

第 4 章 码垛工艺

4.1 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：

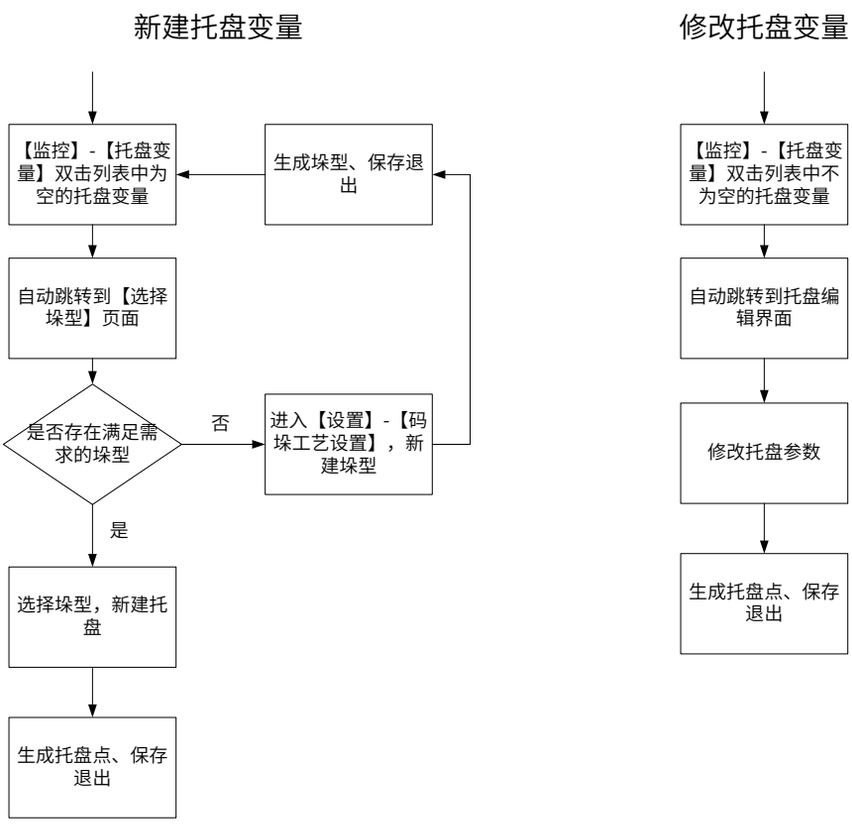
	第一列	第二列	第三列
列数量	4	2	4
列方向	水平	竖直	水平

4.2 编辑托盘变量

对于编辑托盘变量，在【监控】-【全局变量】-【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。



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）。

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：

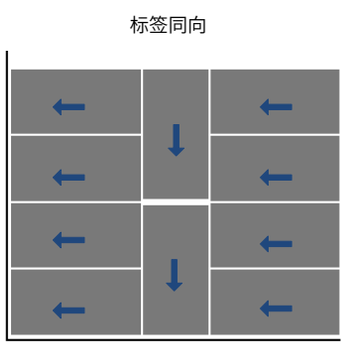
一般的货物的标签贴在货物的 4 个侧面上，利用本功能可以人为设置标签的朝向。

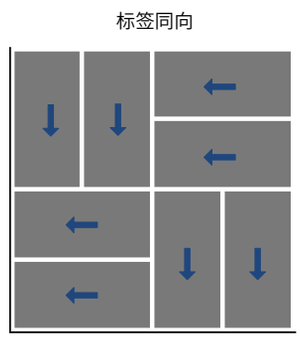
设后续所有货物的标签朝向与第一个朝向所成角度为 X。

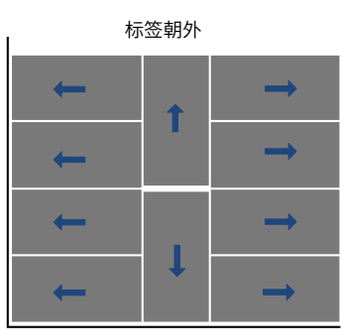
无标签：不考虑标签朝向，按最相差最少的角度排布，特点：-90< x <=90 。

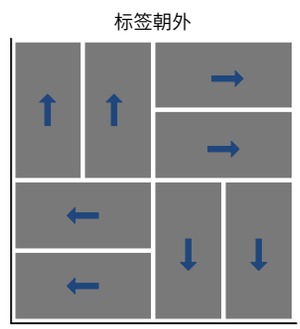
标签朝外：保证第一个标签（基准点）朝外，则其它所有标签朝外，特点：-180< x <=180 。

标签同向：横竖朝向都相同，特点：0<= x <180 。









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 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。

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。

第 5 章 锁螺丝应用

5.1 螺丝拧紧工艺

锁螺丝功能由 15 非标版本支持。由于 15 版本特性，需要使用 WatiInPos 功能阻断预处理问题。

5.1.1 拧紧工艺配置

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。

5.1.2 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：低速搜索 - 高速拧紧 - 低速反拧 - 低速拧紧 - 拧紧 OK，工艺参数就是负责控制这些阶段执行的节奏。

螺丝锁付时拧紧动作分解如下图所示：

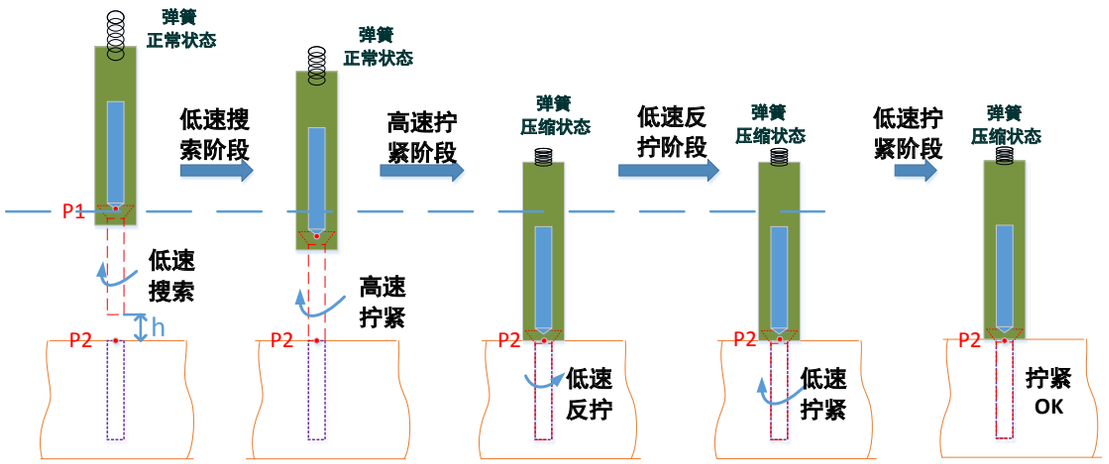


图 5-1 拧紧工艺原理图

1 低速搜索

电批头从 P3 点开始向下运行时使能电批，则批头以“搜索速度”低速空转，方便批头和螺帽对准咬合。与此同时，批头转动所设置的“搜索圈数”左右后，使螺丝末端刚好差不多到达螺孔表面 P4，然后进入高速拧紧。

2 高速拧紧

高速拧紧过程中，批头带着螺丝以“高速转速”旋转，用于提高锁付效率，随着拧紧深度不断增加，电批扭矩会进一步增大，当达到“高速转矩”阀值时，随后便切入低速反拧阶段，表明高速拧紧已完成。

3 低速反拧

低速反紧阶段批头通过反转一小角度，可减少批头和螺帽之间的相互作用力，避免高速阶段的高速转矩过大导致太大的过冲，因为过冲太大容易导致拧紧扭矩一下子达到目标扭矩，从而导致提前拧紧，低速阶段无法前进。

4 低速拧紧

低速拧紧阶段批头以均匀的低“目标转速”将螺丝拧紧，低速阶段可提高拧紧质量。同样，低速运行中，当扭矩到达“目标扭矩”时，表明低速拧紧已完成，并保持一段时间“目标转矩保持时间”，防止扭力反转。

参数解释：

1） 备注名：用户可自定义，目前不支持中文

2） 软启动时间：(15 版本及后续版本可忽略)

3） 搜索速度：低速搜索阶段批头的旋转速度，批头带着螺丝低速旋转寻找锁付孔位，用于批头良好对孔。

4） 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。搜索圈数 N 的理论值计算如下：

- 粗略计算：根据指令 LockScrew(0，P[4]，V[10])，N = h/(V[10]*Vact)*Vn/60，其中 h = P3 – P4 - $L_{螺丝长度}$，P3 为螺丝锁付口上方一点，P4 为指令中的螺丝锁付口，V[10] 为轴 3 速度等级，Vact 为轴 3 实际物理最大数值，Vn 表示搜索转速（rpm）。一般粗略估算即可算出搜索圈数 N 值，若想计算更精准，可采用精准计算。
- 精准计算：LockScrew 指令机器人运动的理论时间的计算情况比较复杂。一般来说，对于运动长度不是特

别长的 LockScrew 指令，机器人运动时间 t = 8*sqrt(h/amax)，如果计算出来的时间小于 200ms，取 t = 200ms。在加速度设置为 8000mm/s^2，3 轴关节速度为 3000r/min，速度比例为 100 的情况下，Lock-Screw 指令机器人运动距离若小于 50mm，则运动时间为 200ms。计算出机器人运动时间后再依据电批搜索速度来计算需要设置的搜索圈数，建议设置的搜索圈数能大致保证机器人到位和电批切高速同步。忽略电批加速的影响，圈数 N = t*（Vn/60），建议默认值设置为 N=0.2*(Vn 默认 /60)。

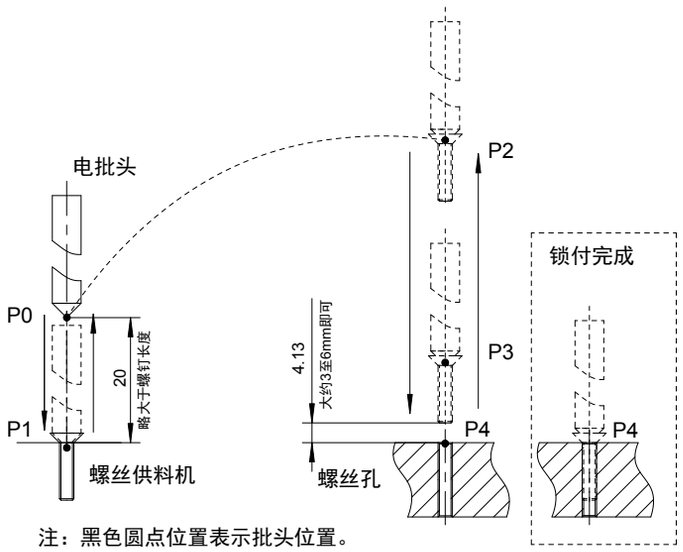


图 5-2 圈数 N 的计算模拟图

注意：特殊情况分析，如上图，从 P3 点到 P4 点，理想情况是螺钉下部接触到 P4 点就正好进入高速阶段，设该段距离为 l1，$l_{1=}$ P3P4 – $l_{螺钉长度}$，因此在滑台移动该段距离的同时刚好完成搜索圈数。而实际中主要有两种情况：

■ 螺钉下部未接触到 P4 点已经完成搜索后立即切入高速，则有一部分高速阶段是没有拧螺丝的；

■ 螺钉下部接触到 P4 点后一小段时间才完成搜索，这样搜索速度会带动螺丝稍微转入螺纹孔一点点。

5） 高速转速：高速拧紧阶段的电机转速。

6） 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。

■ 如果高速扭矩设置过高会可能会在高低速切换时位置过冲太大提前拧紧，从而无法再低速前进；

■ 如果高速扭矩设置过低也会导致低速段运行时间过长，降低拧紧效率。

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，锁付完成。

1） 目标转矩设置过大：可能会导致螺丝拧的太紧，不易拆卸。

2） 目标转矩设置过小：导致未完全拧紧

锁付最小圈数：小于最小圈数就到达目标扭矩，代表浮锁。

锁付最大圈数：大于最大圈数仍没到达目标扭矩，代表滑牙。

搜索超时时间：如果在设定的搜索超时时间内还未达到搜索圈数，那么伺服会报 ER.963 故障，提示搜索超时滑牙，主要原因是从吸取螺丝开启使能到伺服判断接触螺孔的过程重，达到搜索圈数的时间超过了搜索超时设定时间。防止搜索时间过长，导致效力过低。

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。

5.1.3 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);                    # # 加载锁螺丝工艺参数文件
Movj P[0],V[30],Z[0];
Movl P[1],V[30],Z[0];
WaitInPos;
## 运动到 P1 点后打开气阀吸附螺丝
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];
## 开始锁付
LockScrew (0,P[4],V[30]);
CheckLock B0;                                  # # 监测锁付结果
If B0==1
Print “OK”;
Else
Print “NG”;
EndIf;
Set Out[7],OFF;                                # # 关闭气阀
Movl P[2],V[30],Z[0];
END;
```

5.1.4 拧紧状态监控

监控状态在【监控】-【锁螺丝状态】-【锁付统计】中。

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

注意：图中计数器清零按钮只有在非再现模式下才可以清零。

5.2 螺丝拧松工艺

5.2.1 拧松工艺配置

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。

5.2.2 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。其过程分为以下三个阶段：低速搜索 - 高速拧松 - 低速等待，各个阶段具体描述如下：

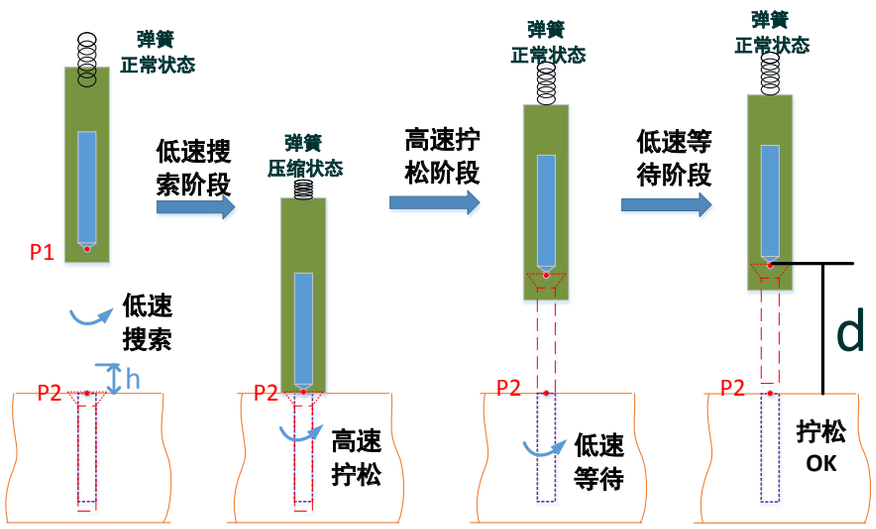


图 5-3 拧紧工艺原理图

a、搜索阶段：批头未接触螺帽时，批头随着轴 3 往下移动，同时批头以“搜索速度”缓慢旋转；

b、拧松阶段：当批头接触螺帽达到“搜索圈数”就开始计算拧松的圈数，并以拧松转速开始执行拆的工艺，直到电机转动相应的“拆螺丝圈数”；

c、等待阶段：达到目标圈数后批头会以搜索速度旋转设定的“等待时间”，以便上抬过程将螺丝吸取出来。

关键参数解释：

备注名：用户可自定义，目前不支持中文

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。

目标转速：拧松阶段的电批转速 rpm。

拆螺丝牙距：螺丝的牙距，单位：0.01mm，如设置值为 50 时，即牙距为 50x0.01mm=0.5mm。

拆螺丝圈数：拧松圈数。

等待时间：达到目标圈数后批头会以搜索速度旋转设定的“等待时间”，以便上抬过程将螺丝吸取出来。

5.2.3 拧松工艺编程

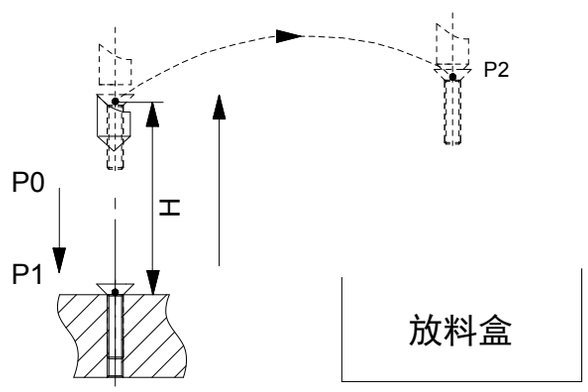
拆螺丝指令 UnLockScrew 集成了螺丝拧松的整个过程，因此使用者在编程时不必关心这些过程的具体实现，只需直接使用指令。下面是一个简单的拆螺丝应用示例：

加载工程：程序使用 LoadScrewParm 指令，将锁螺丝工程文件中的拧松参数加载。

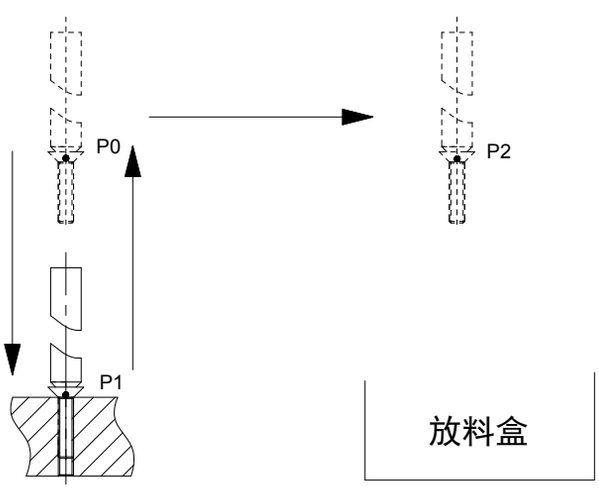
准备拧松：机器人通过 Movj 等运动指令运动到起始位置 P[3]，然后快速移动批头至螺丝拧松预备位 P[0]，等待 P[0] 到位后通过 Set 指令打开气阀

拧松：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。

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：黑色圆点位置表示批头位置。



程序：

```
START;
LoadScrewParm(“aa.stp”,B0);                    ## 加载锁螺丝工艺参数文件
Movj P[0],V[30],Z[0];
WaitInPos;
## 运动到 P0 点后打开气阀，并开始执行螺丝拧松指令
Set Out[7],ON;
UnLockScrew(0,P[1],V[30],2,12);
CheckUnLock(B1);# # 监测拧松结果
If B1==1
Print “OK”;
Else
Print “NG”;
EndIf;
Movl P[3],V[30],Z[0];
WaitInPos;
Set Out[7],OFF; ## 关闭气阀
Delay(0.1);
END;
```

5.3 基于视觉的螺丝锁付

在现场，通过相机获取待锁付螺纹孔图像，将待锁付螺纹孔图像发送至控制器；控制器对机器人和锁付装置进行综合控制，最终将螺丝锁付。

案例流程

1） 锁螺丝机本体精度大致验证

以批筒为示教末端，通过多次左右手进行锁螺丝机本体精度大致验证，建议后期设计一种锁螺丝机手动标定工具，保证精度验证便捷性和准确度。

2） 视觉标定

■ 基准点 1/2 选取

实验过程中相机安装方式是“手动 - 随动二轴”，在示教器中进行基准点 1/2 选取时，由于锁螺丝机没有工具，因此直接将批筒末端作为示教末端，进行左右手操作来选取基准点，特别值得注意的是在标定过程选取基准点 1/2 时，视觉软件中的模板训练所选取的图形需要与该基准点一样。

■ 九点标定（采用手动标定）

在相机视野范围内，通过移动批筒末端，示教九个点位，分别将这些位置值输入到示教器中，按照示教器中标定流程往下走即可得到标定精度结果。

标定结果显示（如下表）

X 方向平均误差（mm）	0.021	Y 方向平均误差（mm）	0.017
X 方向最大误差（mm）	0.049	Y 方向最大误差（mm）	0.045
X 方向单位像素尺寸（mm）	0.029	Y 方向单位像素尺寸（mm）	0.036
标定工具 X 方向偏移	-0.024	标定工具 Y 方向偏移	-0.002

3） 程序编辑

程序流程（视觉程序和机器人程序）

■ 视觉程序总流程：接收数据—> 数据比较—> 相机图像—> 模板匹配—> 发送数据

■ 机器人程序总流程：建立通讯—> 运动到拍照点位—> 软触发相机拍照—> 接收像素位置数据—> 转换至机器人坐标下位置—> 运动至锁付孔位置开始锁付。

同臂拍照锁付（如左臂拍照，左臂锁付）。

4） 机器人程序

```
START;
VelSet 100;
L[12]:
B0 =0;
While B0<>1
Open Socket(“192.168.24.98”,2000,3005,B0) ;          # # 打开 socket
EndWhile;
String shou;
L[1]:
Jump P[5],V[30],Z[0],LH[20],MH[-10],RH[10];
WaitInPos;
## 运动到 P5 点后给视觉发送请求
String str1="TA";
SetPortBuf(str1) ;
Send Port[3005];
WaitInPos;
Delay T[3];
L[0]:
Get Port[3005],T[0.5],Goto L[0];                     # # 接收数据
shou = GetPortbuf(0,100);
B0 = StrGetData(shou,",",D0);                        # # 解析
Print D0;
P[0] =(D0,D1,0,0,0,0),(-1,0,0,0),(6,0,2);            # # 随动相机坐标系下的 X/Y 值
P[1] =(0,0,0,0,0,0),(-1,0,0,0),(2,0,0);              # # 直角坐标系下的值设置
Print P[0];
Cnvrt(P[0],P[1],Tool[0],P[5]);                       # # 将随动相机坐标系的值转换成直角坐标系下的值
Print P[1];
WaitInPos;
PR0 = (0,0,-25,0,0,0);
PR1 = (0,0,-42,0,0,0);
LoadScrewParm("newproject.stp",B202);                ## 加载工艺参数文件
## 运动到 P102 后打开气阀
Jump P[102],V[30],Z[0],LH[10],MH[0],RH[30];
WaitInPos;
Set Out[15],ON;
Jump Offset(P[1],PR0),V[30],Z[0],LH[50],MH[0],RH[10];  ## 运动相应的偏移位置
P[200] =OffSet(P[1],PR1);
LockScrew 0,P[200],V[30];                            ## 开始锁付
CheckLock B201;
Set Out[15],OFF;
WaitInPos;
Delay T[1.5];
Print P[1];
Goto L[1];
Close Socket,3005;                                   ## 关闭 socket
Goto L[12];
END;
```

第 6 章 远程 IO 应用

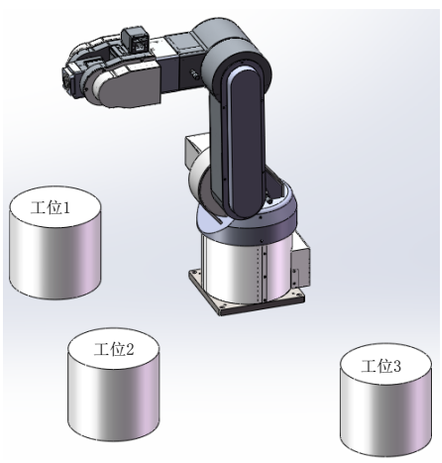
6.1 概述

利用远程 IO 可以控制机器人运动，多用于“工位预约”的场合。

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。

所谓“工位”是指机器人能在多个加工区域进行加工。外部有多个用户输入，当用户指定要工位 1 运作时，则机器人转到工位 1 区域，执行完成工位 1 的机器人程序；当用户指定要工位 2 运作时，则机器人转到工位 2 区域，执行完成工位 2 的机器人程序。以此类推。

所谓“预约”是指用户可以提前指定各个工位的运动，在当前机器人运动中就指定后面几个工位的运作。例如工位 1 先发出请求，则会机器人先运行工位 1 的程序；在运动时，预先指定后面依次运作工位 2、3，则机器人会在工位 1 程序运行结束后，再先后运行工位 2 程序、工位 3 程序。



指定工位动作的用户输入可以依靠 IO 来控制，即通过输入信号通知机器人进行工位运动。此外，为了更好的对机器人控制，还可用输入信号来通知机器人运行、暂停、停止、速度调节等。为了检测机器人状态，可将输出信号关联到机器人的状态信息，这样通过输出信号，用户就能判断机器人状态。

NOTE

- 只有控制设备选用为“远程 IO 单元”，用户的 IO 信号才能控制机器人；否则，不能控制机器人。
- 使用工位预约时，示教软件可以依然可以起到监控作用，但不能进行控制操作。
- 一个工位正在运行或已预约了，再次进行预约无效。
- 在使用前，需要进行远程 IO 配置，包含硬件和软件的配置。参见*第 280 页上的“6.1 远程 IO 的配置”*。
- 使用时，切换控制设备为远程 IO 单元，通过控制 IO 来控制机器人。参见*第 283 页上的“6.2 远程 IO 的使用”*。

6.2 远程 IO 的配置

远程 IO 配置是指 IO 信号与机器人功能的建立连接，如急停、程序选择、启动、停止等。下面从硬件和软件两方面说明：

1 硬件：

连接外部设备与机器人系统的 IO 线路。

（当机器人自带的 IO 不满足需求时，需要给机器人增设 IRLink 模块或板卡）


NOTE

◆ 接线时，保证所有用户自己设定的控制按钮为接触式。即按钮都是常开状态状态，按下响应一次，然后自动弹起。以此使得命令不会被反复发送。

2 软件：

提前准备好被控制的机器人程序。

确保软件中 IRLink 配置与实际硬件符合

配置 IO 关联的机器人功能。

软件操作可参见《汇川机器人编程设计与实现 - 基础操作篇》“2.4.5 节 IRLink 配置、IO 配置”。

基于远程 IO 控制的工位预约：

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。它是多个远程 IO 应用的集合。一个工位上存在一套远程 IO 应用。当某个工位能发出加工请求时，就能利用远程 IO 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）

注意：一个工位正在被运行或处于待运行状态时，该工位再次请求会无效！

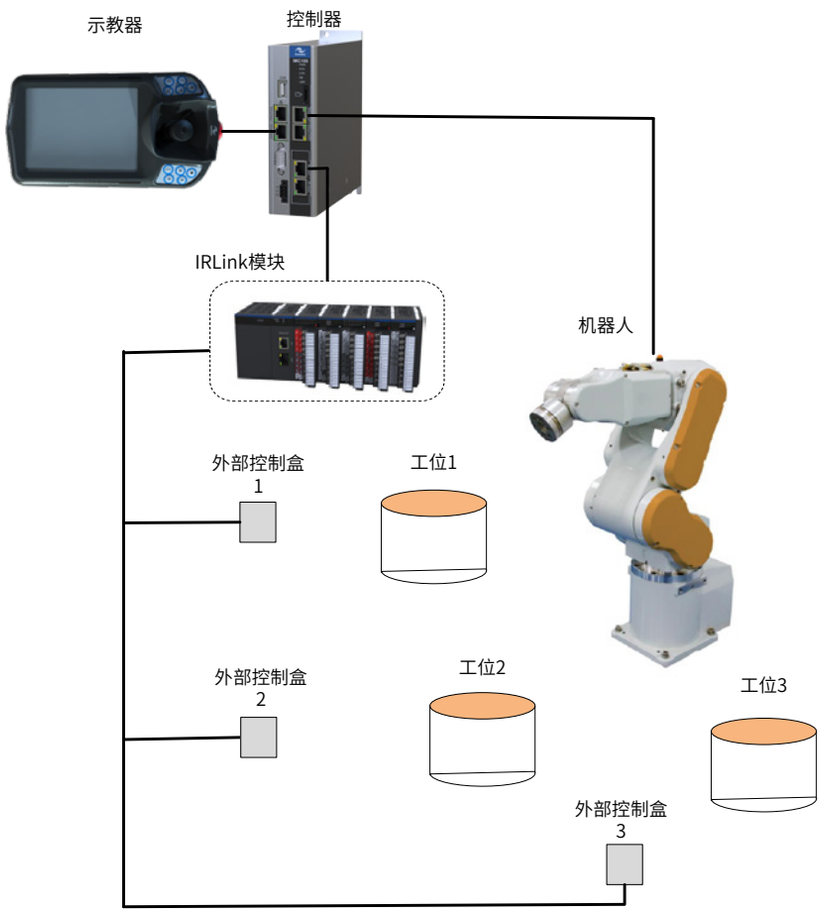
3 下面以一个 3 工位的案例说明：

1） 案例需求：3 个工位，每个工位上有一个控制盒，每个控制器盒上按钮连接着 IO，具有如下功能：

序号	控制	序号	报警
1	急停	1	报警
2	清除报警		
3	启动		
4	停止		
5	暂停		
6	工位程序的选择		
7	速度加		
8	速度减		

上述的“6 工位程序”分别为 3 个工位对应的机器人程序。

在使用时，计划只要按住对应按钮即可实现对应控制。



2） 实施：

按上述需求，至少需要 10 个输入（7 个控制功能 +3 个工位程序）、1 个输出。

机器人出厂默认有 16 输入、16 输出，已满足需求，因此无需增加 IO。直接接线，并配置 IO 即可。对于输出报警，默认关联有 Out[1]，不必进行配置。

这里我们假设使用 IRCB10 系列电柜，且电柜当前只有一个 0808 模块的情形。由于 IRCB 系列电柜 IN-[0]~IN[2] 为系统 IO，因此需要增加一个 0808 模块，并分配这 10 项输入功能匹配 IN[3]~IN[12]，唯一的输出“报警”按默认匹配 Out[1]。

3） 步骤：

（1）利用示教器编辑好 3 个工位的机器人程序。

（2）在电柜上增加一个 IRLink 0808 模块，接好物理线路。并在软件上配置对应 IRlink，完成后重启生效。

（3）依照下表接线，并在示教器上进行 IO 配置。

IO 功能	IO 选项
急停	In[3]
工位程序 1	In[4]
工位程序 2	In[5]
工位程序 3	In[6]
启动	In[7]
暂停	In[8]
停止	In[9]
速度加	In[10]
速度减	In[11]
清除报警	In[12]
报警显示	Out[1]

6.3 远程 IO 的使用

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”，才能开始使用。

选择远程 IO 单元，需要指定启动速度。

启动速度是指: 切换到远程 IO 单元后，机器人的启动速度。该值是一个百分比，如设为 30 代表以 30% 速度运行。

切换后，系统将自动转换到再现运行状态。此时，即可操作 IO 控制机器人。

基于远程 IO 的工位预约控制时：先按下外部控制的“启动”按钮，然后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：

1） 在远程 IO 控制下，示教软件只能进行监控和切换回示教器控制，不能进行其它操作。

2） 运行过程中不能进行控制设备的变更。

3） 所选程序应当避免死循环。因为一旦死循环出现，则一直在此程序中运行；尤其是在工位预约中，即使再触发的其它程序运行，也不会有效果。

4） 远程 IO 控制中，利用外部 IO 控制机器人急停后，退出远程 IO 控制，此时需要拍下急停、再拉起急停才能恢复到不急停状态。

6.4 相关：Modbus 控制

利用 modbus 也能控制机器人运动；且 modbus 能达到更多的控制，如设定具体的速度值，读取伺服报警等等。Modbus 应用参照第 8 章 Modbus 通信及控制应用。

第 7 章 利用 API 编程

7.1 API 调用说明

利用 API 函数接口，用户可以通过 VB、VC、C# 等程序语言开发出专有的机器人系统应用软件。下面以 VS 平台中 C/C++ 应用开发为例，介绍基本功能的实现及调用范例。

7.1.1 连接 / 断开机器人

调用 IMC100_Init_ETH() 函数可以通过网络连接机器人，调用 IMC100_Exit_ETH() 可以断开机器人。

在调用其他任何 API 之前，开发者应调用一次 IMC100_Init_ETH()，以确保每次打开应用时机器人已经连接。如果该函数返回值非零，请检查机器人控制系统是否正常启动，并参见《汇川机器人设计应用与维护手册 - 附录篇》“API 故障连接表”。

调用 IMC100_Exit_ETH() 应该在调用其他 API 之后，该 API 返回 0 后机器人断开连接。

调用 IMC100_Init_ETH() 代码示例如下：

```
int ret = 0;
        DWORD dwIP1 = 0xc0a81719; // 对应 IP：192.168.23.25
        int ipPort = 2222;
        int timeOut = 5;  // 通讯超时时间 5s
        int robotNo = 0;
        ret = IMC100_Init_ETH(dwIP1, ipPort, timeOut, robotNo);
        if(ret < 0)
        {
                // 此处加入异常处理代码
                return;
        }
```

调用 IMC100_Exit_ETH() 代码示例如下：

```
ret = IMC100_Exit_ETH(0)
        if(ret < 0)
        {
                // 此处加入异常处理代码
                return;
        }
```

7.1.2 监测机器人状态

IMC100 机器人提供 API 函数接口用于监测机器人状态。该类函数不受控制权、用户级别的约束。调用该类函数前，请确认机器人系统已经正常启动，并且连接机器人成功。该类函数包括 IMC100_Get_PosHere()、IMC100_Get_DINum()、IMC100_Get_StrPara()、IMC100_Get_P() 等。

调用 IMC100_Get_PosHere() 代码示例如下：

```
        // 查询机器人当前坐标系下的位置值
        int ret = 0;
        int robotNo = 0;
        int dinum = 0;
        int dists = 0;
ROBOT_POS posTemp;
```

```
 memset(&posTemp, 0, sizeof(posTemp));
ret = IMC100_Get_PosHere(&posTemp, robotNo);
if(ret < 0)
 {
        // 此处加入异常处理代码
}
```

调用 IMC100_Get_DI() 代码示例如下：

```
// 查询 DI0 状态，dists 为 1 表示 on
ret = IMC100_Get_DI(dinum, &dists, robotNo);
if(ret < 0)
{
        // 此处加入异常处理代码
}
```

7.1.3 获取控制许可

调用 IMC100_AcqPermit() 可以获取到机器人控制权许可，调用 IMC100_CurPermit() 可以查询当前获得许可的客户端。

由于一个机器人系统可以连接多个以太网客户端，所以当其中一个客户端需要控制机器人时，必须获得控制许可。调用前确认机器人系统已经正常启动，并且连接机器人系统成功。并且，示教器界面“系统设置 - 其他设置 - 其他 - 控制设备”选项下请选择“远程以太网客户端”。

调用 IMC100_CurPermit() 及 IMC100_AcqPermit() 代码示例如下：

```
int ret = 0;
  int ower = 0;
  DWORD IpAddr = 0;
  int ipPort = 0;
      int robotNo = 0;
  ret = IMC100_CurPermit(&ower, &IpAddr, & ipPort, robotNo);
      if(ret < 0)
      {
            // 此处加入异常处理代码
            return;
      }
      if(ower != 1)  // 当前客户端设备未获得许可
      {
            ret = IMC100_AcqPermit(1, robotNo);  // 强制获取许可，ower 为 0 时可普通获取
            if(ret < 0)
            {
                  // 此处加入异常处理代码
            }
      }
```

7.1.4 用户登录

调用 IMC100_CurUserType() 可以查询当前设备用户等级，调用 IMC100_UserLogin() 进行用户登录，调用 IMC100_UserLogout() 可以退出当前用户登录状态。

不同的用户等级允许用户对机器人进行不同程度和范围的控制和操作。

调用前确认机器人系统已经正常启动，并且连接机器人系统成功。示教器界面“系统设置 - 其他设置 - 其他 -

控制设备”选项下已选择“远程以太网客户端”。

调用 IMC100_CurUserType()、IMC100_UserLogin() 及 IMC100_UserLogout() 代码示例如下：

```
int ret = 0;
        int type = 0;
        char password[8];
        int robotNo = 0;
   ret = IMC100_CurUserType(&type, robotNo);
        if(ret < 0)
        {
                // 此处加入异常处理代码
        }
        type = 2;
        memcpy(password, "000000", sizeof(password));
ret = IMC100_UserLogin(type, password, robotNo); // 登录管理模式，密码同示教器密码
        if(ret < 0)
        {
                // 此处加入异常处理代码
        }
        ret = IMC100_UserLogout(robotNo);
        if(ret < 0)
        {
                // 此处加入异常处理代码
        }
```

7.1.5 机器人回原点

调用 IMC100_DsMode() 打开数据流模式，调用 IMC100_Home() 控制机器人回原点。

调用前确认机器人系统已经正常启动，机器人系统成功连接，并已获得客户端控制许可。

调用 IMC100_DsMode() 和 IMC100_Home() 代码示例如下：

```
int ret = 0;
        int sts = 0;
        int robotNo = 0;
   ret = IMC100_Get_DsMode(&sts, robotNo);
        if(ret < 0)
        {
                // 此处加入异常处理代码
        }
        if(sts == 0)  // 数据流模式关闭
        {
                int cmd = 1;
                ret = IMC100_DsMode(cmd, robotNo);  // 开启数据流模式
                if(ret < 0)  // 此处加入异常处理代码
        }
        int num = 0;
        ret = IMC100_Home(num, robotNo);  // 机器人回 0 号原点
        if(ret < 0)
        {
                // 此处加入异常处理代码
        }
```

7.1.6 开发语言选择

■ 基于 VB 开发

1） 以 Visual Basic 6.0 环境为例，新建工程。

2） 将产品提供的 IMC100API.bas、IMC100API.dll 文件拷贝至工程相应目录下。

3） 菜单中选择“工程 / 添加模块 / 现存”，找到对应工程目录下的 IMC100API.bas，添加到工程中。

4） 调用具体的函数接口，编写应用程序。

■ 基于 VC 开发

1） 以 Visual Studio 2010 环境为例，新建 C++ 工程。

2） 将产品提供的 IMC100API.lib、IMC100API.dll、IMC100API.h 文件拷贝至工程相应目录下。

3） 在程序文件中，增加 #include “IMC100API.h”语句。

4） 在程序文件中增加 #pragma comment(lib,”IMC100API.lib”) 语句，或者在“工程属性 / 链接器 / 输入 / 附加依赖库”中添加 IMC100API.lib。

5） 调用具体的函数接口，编写应用程序。

■ 基于 VB.Net 开发

1） 以 Visual Studio 2010 环境为例，新建 Visual Basic .NET 工程。

2） 将产品提供的 IMC100API.vb 文件拷贝至工程目录下，IMC100API.dll 文件拷贝至工程目标输出路径下。

3） 菜单中选择“项目 / 添加现有项”，找到对应工程目录下的 IMC100API.vb，添加到工程中。

4） 调用具体的函数接口，编写应用程序。

■ 基于 C# 开发

1） 以 Visual Studio 2010 环境为例，新建 Visual C# .NET 工程。

2） 将产品提供的 IMC100API.cs 文件拷贝至工程目录下，IMC100API.dll 文件拷贝至工程目标输出路径下。

3） 菜单中选择“项目 / 添加现有项”，找到对应工程目录下的 IMC100API.cs，添加到工程中。

4） 调用具体的函数接口，编写应用程序。

7.1.7 基本功能描述

函数接口主要分为三类，初始化应用类、监控类以及控制类。

初始化应用类：该类函数是其它函数调用的基础，包括函数 IMC100_Init_ETH()、IMC100_Exit_ETH()。

监控类：该类函数在 IMC100_Init_ETH() 成功调用后均可正常调用，没有控制权、用户级别约束。包括 IMC100_Get_PosHere()、IMC100_Get_DINum()、IMC100_Get_StrPara()、IMC100_Get_P() 等。

控制类：该类函数不仅需要 IMC100_Init_ETH() 成功调用，还需要控制设备为“远程以太网客户端”，且需要调用 IMC100_AcqPermit() 获取控制许可，部分函数依据用户等级还需要调用函数 IMC100_UserLogin() 登陆。包括 IMC100_EmergStop()、IMC100_MovJ()、IMC100_Set_DO()、IMC100_Set_P()、IMC100_Set_Home-Pos() 等。

详细功能描述参考汇川机器人设计应用与维护手册 - 附录篇》“API 说明”。

7.2 典型应用案例

利用已提供的函数接口，用户可以根据实际需求，使用 VB、VC、C# 等程序语言开发出专有的 IMC100 机器人系统应用软件。

7.2.1 上位机打开机器人程序并再现运行

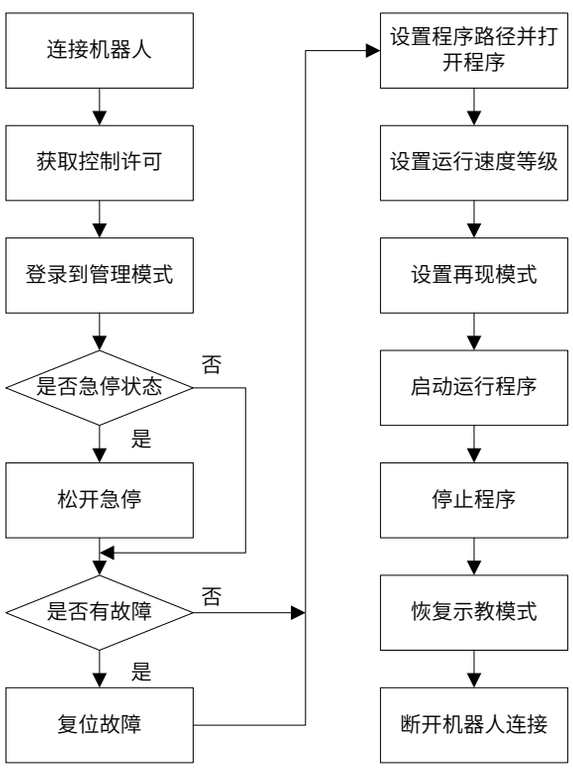
■ 案例描述

实现远程以太网控制机器人运行。

基于 VC 通过调用 API 函数实现上位机控制机器人运行。

实现急停控制、故障复位、机器人程序选择、速度设置、运行程序、停止程序等功能。

■ 程序流程图



■ 程序实例

```
int ret = 0;                                                    
DWORD dwIP1 = ntohl(inet_addr(“192.168.23.25”));                //IP 地址转换
int ipPort = 2222;                                              // 机器人服务端口号
int timeOut = 5;                                                // 通讯超时时间 5s
int robotNo = 0;                                                // 机器人序号

ret = IMC100_Init_ETH(dwIP1, ipPort, timeOut, robotNo);         // 连接机器人操作部分
if(ret < 0)
{                                                               // 此处可加入异常处理代码
return;
}

int ower = 0;
DWORD IpAddr = 0;
int ipPort = 0;
```

```
ret = IMC100_CurPermit(&ower, &IpAddr, & ipPort, robotNo);      / 获取控制许可
if(ret < 0)
{                                                               // 此处可加入异常处理代码
}
if(ower != 1)                                                   // 当前客户端设备未获得许可
{
ret = IMC100_AcqPermit(1, robotNo);                             // 强制获取许可
if(ret < 0)
{                                                               // 此处可加入异常处理代码
}
}

int type = 2;                                                   // 管理模式
char password[8];
memcpy(password, "000000", sizeof(password));                   // 密码同示教器密码

ret = IMC100_UserLogin(type, password, robotNo);                // 登录管理模式
if(ret < 0)
{                                                               // 此处可加入异常处理代码
}

int sts = 0;
ret = IMC100_Get_EStopSts(&sts, robotNo);                       // 查询急停状态
if(ret < 0)
{                                                               // 此处可加入异常处理代码
}
if(sts == 1)                                                    // 当前急停按下
{
int cmd = 0;
ret = IMC100_EmergStop(cmd, robotNo);                           // 急停松开
if(ret < 0)                                                     // 此处可加入异常处理代码
{
}
}
```

```
int err = 0;
ret = IMC100_Get_SysErr(&err, robotNo);                    // 故障查询
if(ret < 0)
{                                                          // 此处可加入异常处理代码
}
if(err != 0)                                               // 系统有故障
{
ret = IMC100_ResetErr(robotNo);                            // 复位故障
if(ret < 0)
    {                                                      // 此处可加入异常处理代码
}
}

char path[128];
memcpy(path, "TeachProgram/Test.pro", sizeof(path));

ret = IMC100_Set_CurPrgPath(path, robotNo);                // 设置程序路径
if(ret < 0)
{                                                          // 此处可加入异常处理代码
}

int vel = 50;
ret = IMC100_Set_Vel(vel, robotNo);                        // 设置运行速度
if(ret < 0)
{                                                          // 此处可加入异常处理代码
}

int mode = 2;                                              // 再现模式
ret = IMC100_Set_Mode(mode, robotNo);                      // 设置再现模式
if(ret < 0)
{                                                          // 此处可加入异常处理代码
}
```

```
cmd = 0;
ret = IMC100_PrgCtrl(cmd, robotNo);                        // 停止程序
if(ret < 0)
{                                                          // 此处可加入异常处理代码
}

int mode = 1;
ret = IMC100_Set_Mode(mode, robotNo);                      // 设置示教模式
if(ret < 0)
{                                                          // 此处可加入异常处理代码
}

ret = IMC100_Exit_ETH(robotNo)                             // 断开机器人连接
if(ret < 0)
{                                                          // 此处可加入异常处理代码
}
```

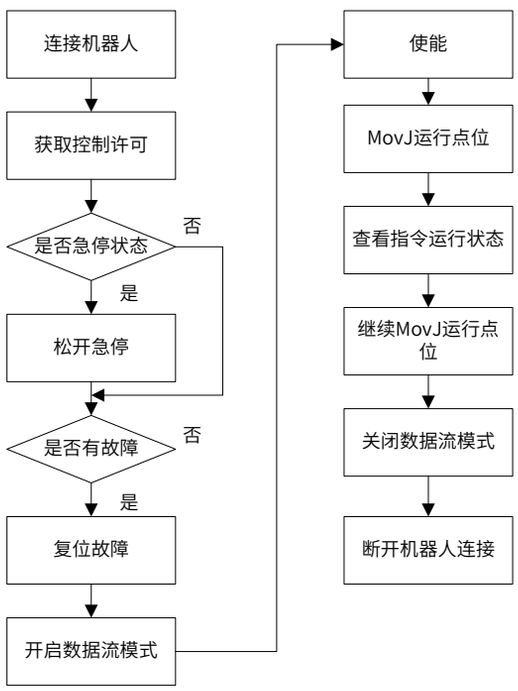
7.2.2 上位机规划点位控制机器人运动

■ 案例描述

上位机规划机器人运动的期望位置。

基于 VC 通过调用 API 函数实现上位机控制机器人点位运动。

■ 程序流程图



■ 程序实例

```
int ret = 0;                                                    //IP 地址转换
DWORD dwIP1 = ntohl(inet_addr("192.168.23.25"));
int ipPort = 2222;                                              // 机器人服务端口号
int timeOut = 5;                                                // 通讯超时时间 5s
int robotNo = 0;                                                // 机器人序号

ret = IMC100_Init_ETH(dwIP1, ipPort, timeOut, robotNo);         // 连接机器人操作部分
if(ret < 0)
{                                                               // 此处可加入异常处理代码
return;
}

int ower = 0;
DWORD IpAddr = 0;
int ipPort = 0;
```

```
 ret = IMC100_CurPermit(&ower, &IpAddr, & ipPort, robotNo);          // 获取控制许可
if(ret < 0)
{
                                                                     // 此处可加入异常处理代码
}
if(ower != 1)                                                        // 当前客户端设备未获得许可
{
ret = IMC100_AcqPermit(1, robotNo);                                  // 强制获取许可
if(ret < 0)
{
                                                                     // 此处可加入异常处理代码
}
}


int sts = 0;
ret = IMC100_Get_EStopSts(&sts, robotNo);                            // 查询急停状态
if(ret < 0)
{
                                                                     // 此处可加入异常处理代码
}
if(sts == 1)                                                         // 当前急停按下
{
int cmd = 0;
ret = IMC100_EmergStop(cmd, robotNo);                                // 急停松开
if(ret < 0)
{
                                                                     // 此处可加入异常处理代码
}
}


int err = 0;
ret = IMC100_Get_SysErr(&err, robotNo);                              // 故障查询
if(ret < 0)
{
                                                                     // 此处可加入异常处理代码
}
if(err != 0)                                                         // 系统有故障
{
ret = IMC100_ResetErr(robotNo);                                      // 复位故障
if(ret < 0)
{
                                                                     // 此处可加入异常处理代码
}
}
```

```
cmd = 1;
ret = IMC100_DsMode(cmd, robotNo);                                   // 开启数据流模式
if(ret < 0)
{
                                                                     // 此处可加入异常处理代码
}


cmd = 1;
ret = IMC100_MotorEnable(cmd, robotNo);                              // 使能
if(ret < 0)
{
                                                                     // 此处可加入异常处理代码
}
```

```
 ROBOT_POS pos;                                          // 定义位置变量，上位机规划
memset(&pos, 0, sizeof(pos));
pos. pos[0] = 10;                                        //J1=10
pos. pos[1] = 10;                                        //J2=10
pos.coord = 1;                                           // 关节坐标系

ret1 = IMC100_MovJ2(pos, 100, 0, robotNo)                // MovJ 运动
if(ret < 0)
{                                                        // 此处加入异常处理代码
}

ret = IMC100_Get_CurCmdSts(&sts, robotNo);               // 运动完成状态检查
if(ret < 0)                                              // 或使用 IMC100_Get_MotionSts
{                                                        // 此处可加入异常处理代码
}
if(sts == 0)                                             // 运动未完成
{                                                        // 自定义操作 , 可循环查询
}

memset(&pos, 0, sizeof(pos));
pos. pos[0] = 100;                                       //J1=100
pos.coord = 1;                                           // 关节坐标系

ret1 = IMC100_MovJ2(pos, 100, 0, robotNo)                // MovJ 运动下一点
if(ret < 0)
{                                                        // 此处可加入异常处理代码
}

cmd = 0;
ret = IMC100_DsMode(cmd, robotNo);                       // 关闭数据流模式
if(ret < 0)
{                                                        // 此处可加入异常处理代码
}

ret = IMC100_Exit_ETH(robotNo)                           // 断开机器人连接
if(ret < 0)
{                                                        // 此处可加入异常处理代码
}
```

Modbus 通讯功能需搭配 InoRobShop 软件产品使用，通过 InoRobShop 软件平台配置 Modbus 协议建立与控制器的通讯。

关于 Modbus 配置：摘自 InoRobshop 帮助文档手册。

第 8 章 Modbus 通信及控制应用

8.1 Modbus 应用概述

8.1.1 Modbus 通讯协议

Modbus 通讯协议是一种标准的工业现场总线协议。IMC100 机器人系统支持基于 RS485 总线连接的 Modbus RTU 协议，也支持基于 Ethernet 连接的 Modbus TCP 协议。

8.1.2Modbus 控制应用场景

在现场 Modbus 应用中，IMC100 机器人一般作为被控对象（该文档不对机器人作为 Modbus 主站的应用予以说明）。

支持标准 Modbus 主站功能的设备，比如触摸屏（HMI）、PLC、工业 PC 等，不论基于 RS485 还是基于 Ethernet 连接，均可以对 IMC100 机器人进行操作。

在机器人配置为 Modbus 从站的前提下，主站设备根据机器人对应的 Modbus 地址表，可以对机器人进行监控、运行等操作。机器人未配置 Modbus 从站时，对应 Modbus 功能无效。

8.1.3Modbus 应用的流程

Modbus 应用的流程分为 2 步：Modbus 通讯配置、控制应用构建。

1） Modbus 通讯配置：借助我司二次开发平台 InoRobShop 软件，进行 Modbus 通讯配置（详细请参见*第 295 页上的“8.2Modbus 通讯配置”*）。

2） 控制应用构建：根据所需的通讯需求，利用从站地址表，完成应用程序的编写（详细请参见*第 297 页上的“8.3 Modbus 控制应用”*）。

3） 上述步骤完成后，即可使用。

4） 对于监控功能，可以直接对机器人相关状态和参数等进行查询；对于控制功能，需要通过示教器将 IMC100 机器人的【控制设备】设置为【远程 Modbus 单元】，方可进行。如下图所示。

8.2 Modbus 通讯配置

用户可以使用InoRobShop 软件对机器人进行 Modbus 相关配置。另外，S01.17 以上版本的示教器软件也直接支持机器人 Modbus 相关配置。配置完成后，建立通讯连接。

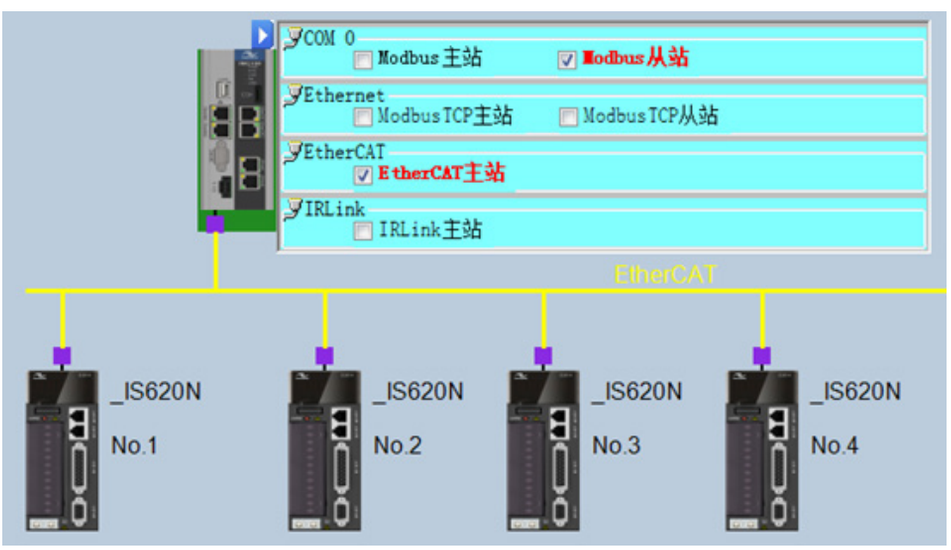
下面详细介绍利用 InoRobShop 配置 Modbus。对基于 RS485 连接的控制而言，机器人配置为 Modbus 从站；对基于 Ethernet 连接的控制而言，机器人配置为 Modbus TCP 从站（也是服务器）。

ModbusTCP 特性：ModbusTCP 是采用标准的 TCP/IP，把 Modbus 信息包打包压缩。这样 ModbusTCP 设备就可以通过以太网和光纤网络进行连接和通讯。与 RS-485 接口相比，ModbusTCP 还允许使用更多的地址、可以采用多主站架构、传送速率可达到 GB/s 的水平。ModbusTCP 网络的从站数量仅受限于网络物理层的能力。通常从站的数量一般在 1024 个左右。

8.2.1 InoRonshop 配置 Modbs 从站

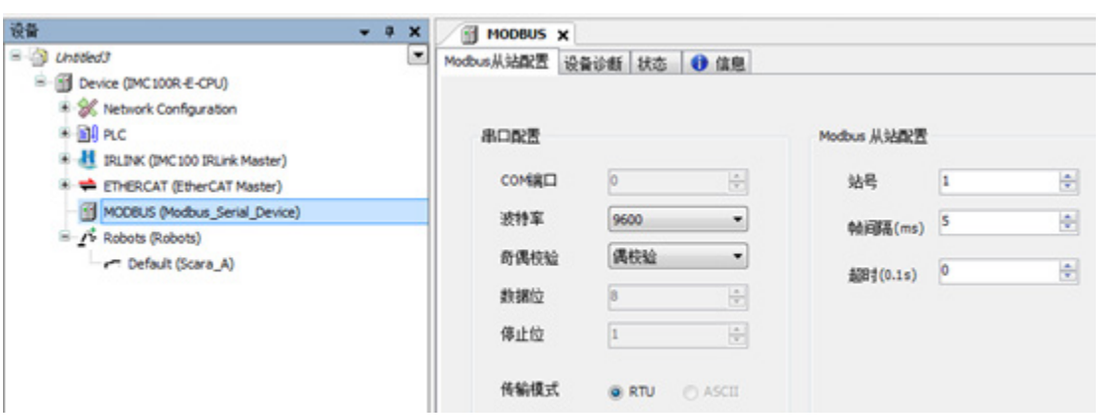
1 组态配置

通过设备树窗口的【Network Configuration】项，在总线配置框中勾选 Modbus 从站。



2 从站配置

双击【Modbus(Modbus_Serial_Device)】进入配置页面。

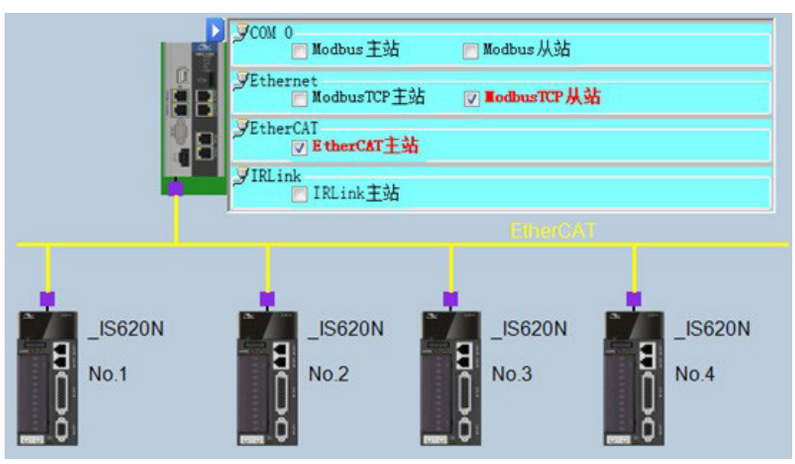


这里包含串口配置和从站配置。波特率、奇偶校验与主站保持一致，站号根据主站设置确定，帧间隔、超时按需求自由设定。一般使用 HMI 时，只控制 1 个机器人控制器，站号、帧间隔、超时保持默认即可，只需保证波特率和奇偶校验对应上。

8.2.2 InoRobShop 配置 Modbus-TCP 从站

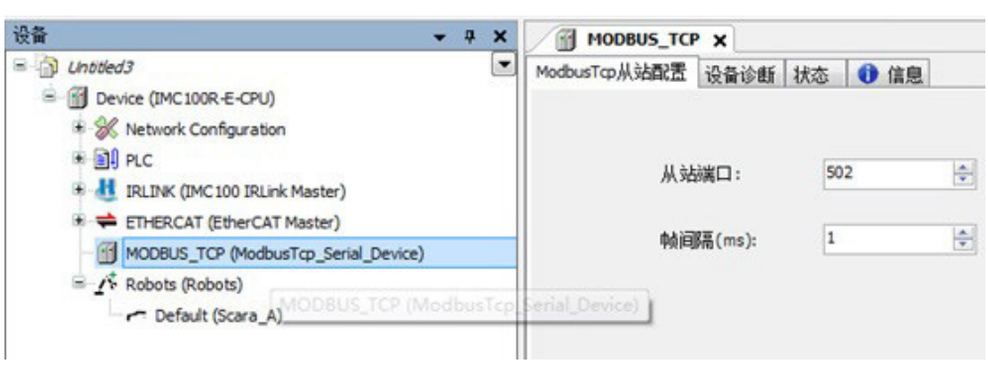
1 组态配置

通过设备树窗口的【Network Configuration】项，在总线配置框中勾选 ModbusTCP 从站。

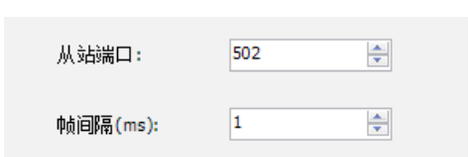


2 从站配置

双击【Modbus_TCP(ModbusTcp_Serial_Device)】进入配置页面。



设置从站端口和帧间隔。



8.2.3 示教器配置 Modbus 从站或 ModbusTCP 从站

在该页面可以配置 Modbus 从站。如下左侧为 Modbus 串口通讯；右侧为 MobusTCP 通讯。

勾选激活 Modbus 从站或 ModbusTCP 从站，然后设置参数即可。

8.3 Modbus 控制应用

8.3.1Modbus 地址说明

IMC100 机器人 Modbus 从站地址表的分配参考了标准的 Modbus 协议，对象分为离散量输入、线圈、输入寄存器、保持寄存器，具体见表 1。

由于对象属性的限制，实现 IMC100 机器人的一个功能控制可能涉及多个地址对象的操作。

表 8-4 Modbus 地址说明

对象	地址范围	对象类型	访问类型	功能码	内容
离散量输入	0 - 4095	单个比特	只读	02	系统提供数据（状态、标志等）
线圈	4096 - 8191	单个比特	读写	01/05/15	用户可改变的数据（控制命令等）
输入寄存器	0 - 32767	16 比特字	只读	04	系统提供数据（当前参数、变量值等）
保持寄存器	32768 - 65535	16 比特字	读写	03/06/16	用户可改变的数据（系统参数、变量值等）

当需要利用 Modbus 通讯控制机器人程序运行时，需在示教器【外设配置】-【IO 配置】-【IO 程序】中设置 3 个工位程序。然后参照 Modbus 从站地址表，对应调用相应的程序。

0x1026	QW65282,bit6	位	工位程序 1
0x1027	QW65282,bit7	位	工位程序 2
0x1028	QW65282,bit8	位	工位程序 3

8.3.2Modbus 控制功能说明

IMC100 机器人的 Modbus 功能主要分为两大类: 一类是监控功能，包括系统状态、当前参数值、变量值等的监控; 另一类是控制功能，包括控制系统启停、修改系统参数、写入系统变量等。

对于监控功能，主站设备可以在机器人正常启动后，在任何条件下对机器人相关状态和参数等进行查询。对于控制功能，需要通过示教器将 IMC100 机器人的【控制设备】设置为【远程 Modbus 单元】，设置成功后，主站设备需要根据具体功能流程对机器人进行操作控制。

在控制功能中，控制命令（见机器人 Modbus 地址表中线圈对象）的类型分为电平型和复归型。对于电平型命令，为使命令生效，用户需向对应线圈写 1，为使命令取消，用户需向对应线圈写 0。对于复归型命令（用户输入命令的上升沿有效），为使命令生效，用户需在对应线圈为 0 时写 1，保持一段时间（建议 200ms 以上）后清 0，可参考 HMI 的复归型按钮的电平时序关系。

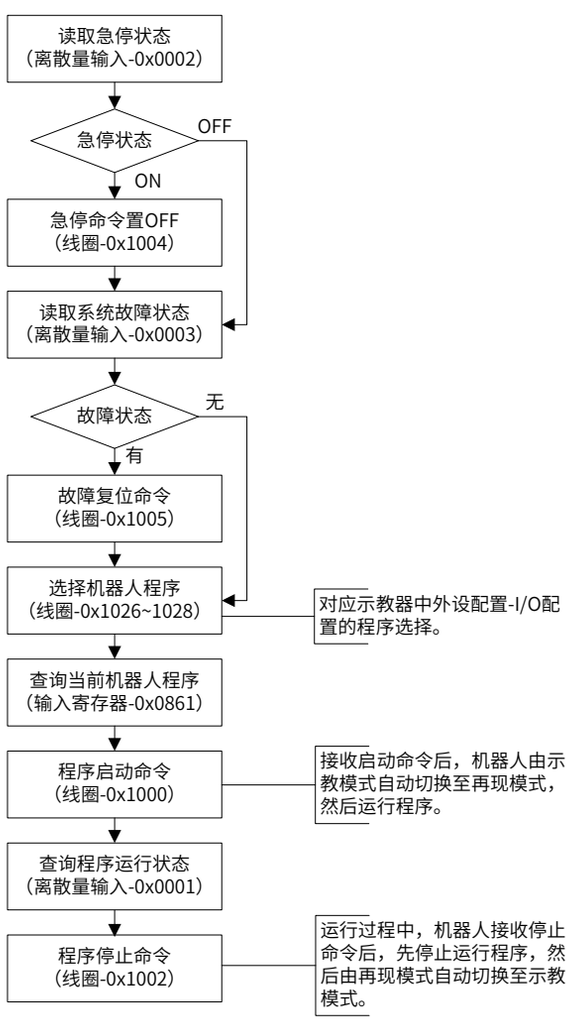
8.3.3 控制案例

1 主站控制机器人程序的启停

■ 案例描述

实现远程 Modbus 控制机器人运行。实现急停控制、故障复位、机器人程序选择、运行程序、停止程序等功能。

■ 操作流程图

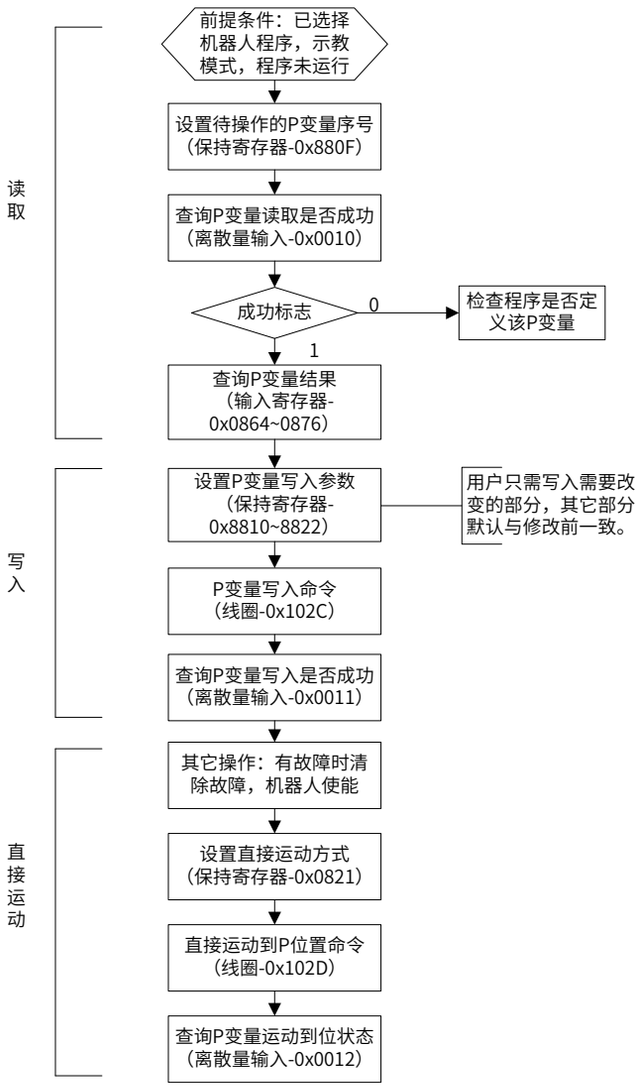


2 主站控制机器人点位示教

■ 案例描述

实现 P 变量的显示和修改。 实现控制机器人运动到指定点（P 变量对应点）。

■ 操作流程图

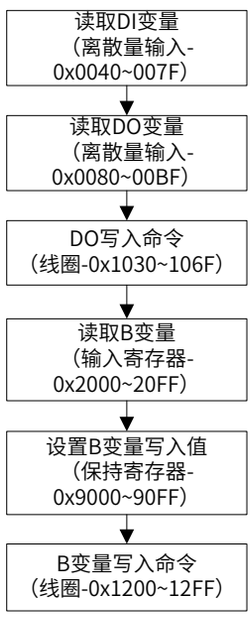


3 主站读写机器人 IO 变量和 B 变量

■ 案例描述

实现机器人的 IO 变量读写。 实现机器人的 B 变量读写。

■ 操作流程图



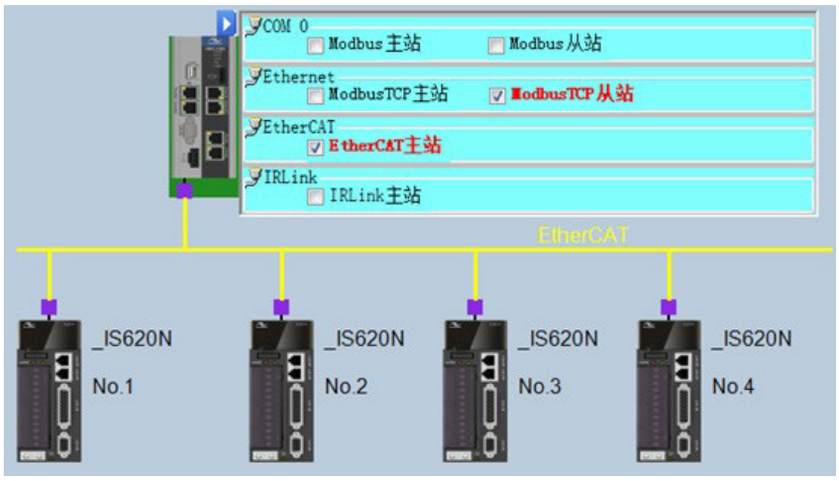
8.4 综合案例——HMI 通过 Modbus 控制机器人运动

本案例上位机采用我司 HMI，使用 Modbus 通讯方式，控制机器人运动。

1 控制器网口配置

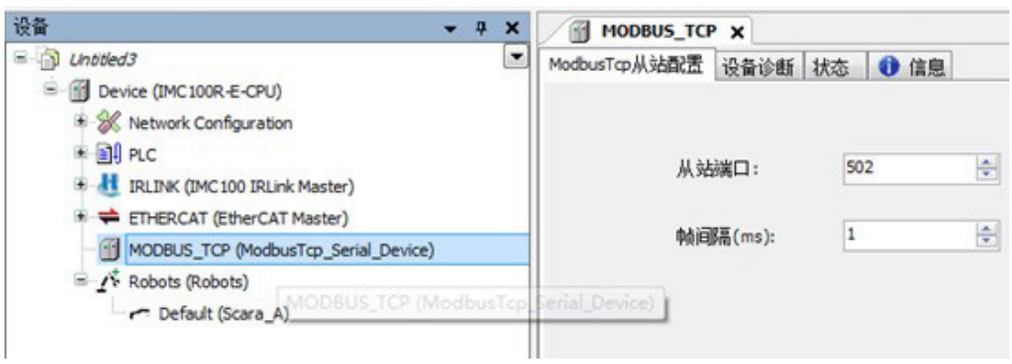
首先，通过机器人示教器操作界面，将机器人控制器网口 EtherNet1 IP 设置为 192.168.22.100。

其次，打开 InoRobShop 软件平台，通过设备树窗口的【Network Configuration】项，在总线配置框中勾选 ModbusTCP 从站。

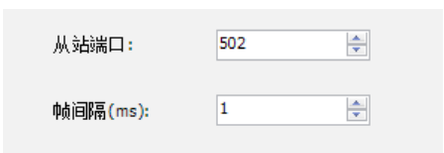


2 从站配置

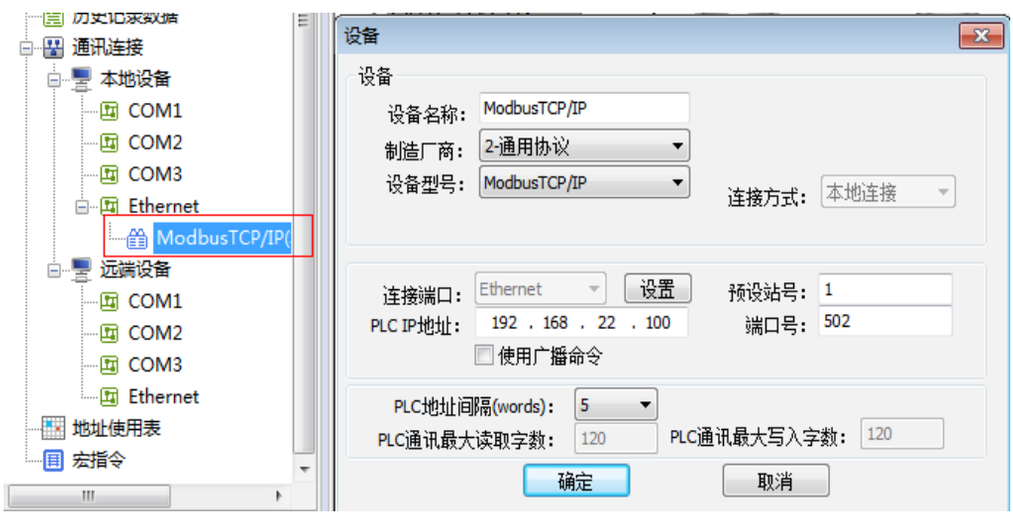
双击【Modbus_TCP(ModbusTcp_Serial_Device)】进入配置页面。



设置从站端口和帧间隔。



打开 InoTouch Editor（HMI）软件平台，通过项目管理的【通讯连接】项，本地设备中的 EtherNet 添加 ModbusTCP/IP（具体配置如下图所示）；



3 在 HMI 上搭建功能项

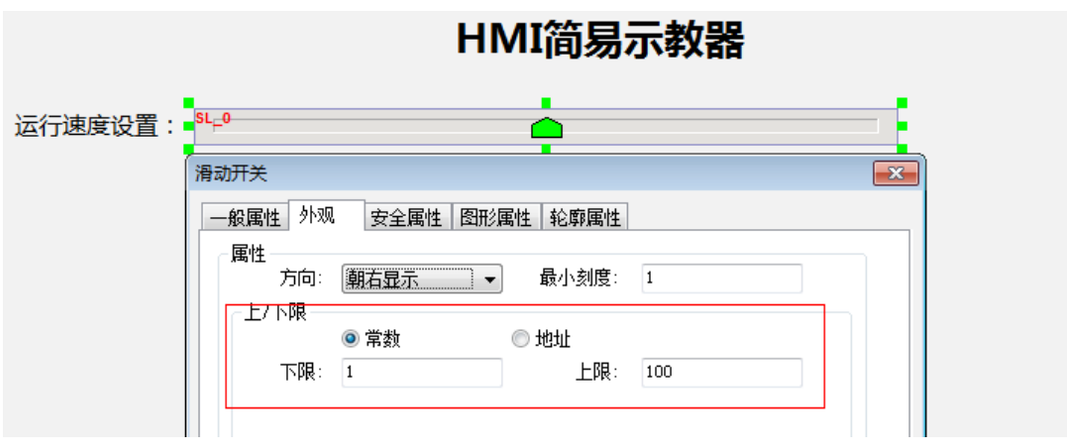
请参见《汇川机器人编程设计与实现 - 附录篇》“Modbus 从站地址表”，如下：

地址 (10 进制)	地址 (16 进制)	变量名称	数据类型	内容	备注
4112	0x1010	QW65281,bit0	位	示教 J1/X +	（复归型）
4113	0x1011	QW65281,bit1	位	示教 J2/Y +	（复归型）
4114	0x1012	QW65281,bit2	位	示教 J3/Z +	（复归型）
4115	0x1013	QW65281,bit3	位	示教 J4/A +	（复归型）
4120	0x1018	QW65281,bit8	位	示教 J1/X -	（复归型）
41210	0x1019	QW65281,bit9	位	示教 J2/Y -	（复归型）
4122	0x101a	QW65281,bit10	位	示教 J3/Z -	（复归型）
4123	0x101b	QW65281,bit11	位	示教 J4/A -	（复归型）
34816	0x8800	MW34816	字	坐标系选择 (1- 关节 ,2- 直角 ,3- 工具 ,4- 用户)	
4099	0x1003	QW65280,bit3	位	使能开关 (0-OFF,1-ON)	
34817	0x8801	MW34817	字	速度设置（1-100）	
2049	0x0801	MW2049	字	当前速度	
2048	0x0800	MW2048	字	当前坐标系	
2052	0x0804	MW2052	单精度浮点	J1/X 坐标低位	
2053	0x0805	MW2053	单精度浮点	J1/X 坐标高位	
2054	0x0806	MW2054	单精度浮点	J2/Y 坐标低位	
2055	0x0807	MW2055	单精度浮点	J2/Y 坐标高位	
2056	0x0808	MW2056	单精度浮点	J3/Z 坐标低位	
2057	0x0809	MW2057	单精度浮点	J3/Z 坐标高位	
2058	0x080a	MW2058	单精度浮点	J4/A 坐标低位	
2059	0x080b	MW2059	单精度浮点	J4/A 坐标高位	

根据上述地址表，设置控件功能，操作如下：

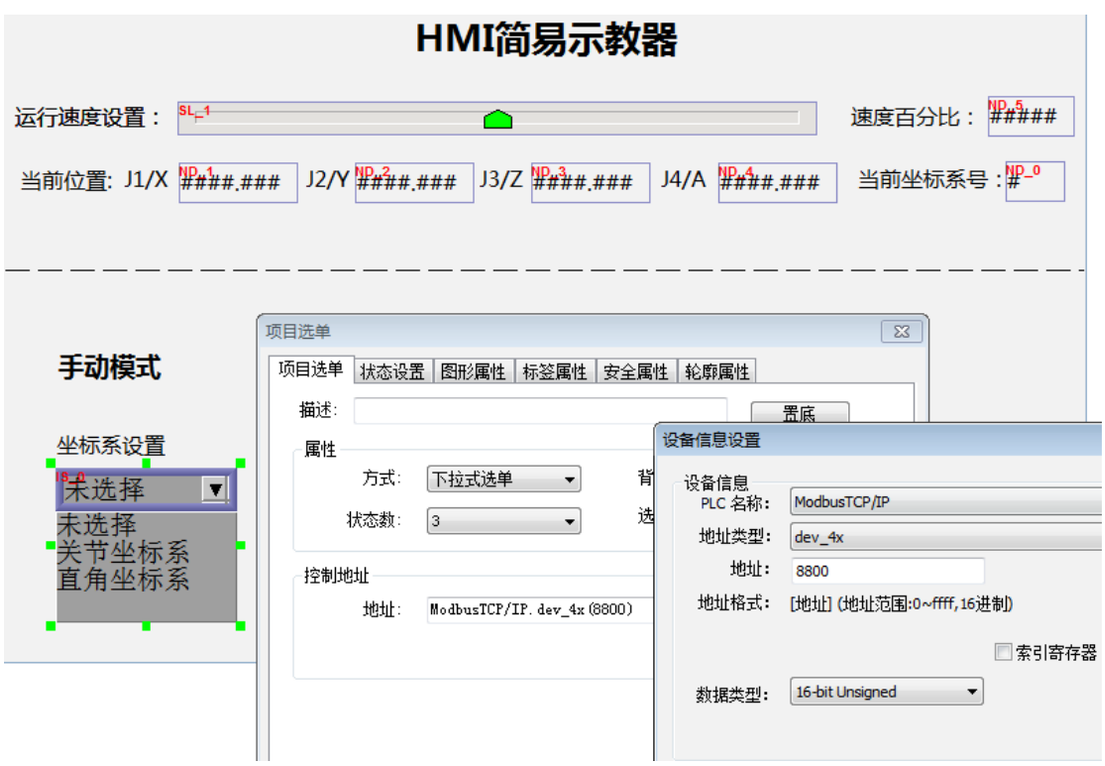
1） 运行速度设置功能操作键设置：添加滑动开关控件，写入地址（地址格式为 16 进制）0x8801；

控件上下限设置为 1-100；



2） 当前位置（各轴姿态）监控窗口：添加数值显示控件，写入地址（地址格式为 16 进制），例如 J1 轴地址为 0x0804，数据类型为 32 位浮点数；其余轴同样设置；速度百分比和当前坐标系号同样选择数值显示控件即可，数据类型以及数值上下限根据地址表规格设置即可；

3） 坐标系设置功能键：添加项目选单控件，输入写入地址（地址格式为 16 进制）0x8800，状态数根据使用需求设置，案例中只使用关节坐标和直角坐标，状态数为 3；



4） 轴操作功能键：添加位状态切换开关控件，输入写入地址（地址格式为 16 进制），例如 J2/Y+ 操作键地址 0x1011，控件开关类型需设置为复归型，其余轴同样设置；

5） 使能操作功能键：添加位状态切换开关控件，输入写入地址（地址格式为 16 进制）0x1003，控件开关类型需设置为切换开关；

经过上述案例操作，HMI 简易示教器操作界面制作已完成，如下图所示：

机器人控制器与 HMI 之间网络连接正常后，需将机器人控制权限设置为远程 Modbus 单元，如下图所示，在示教器操作界面中，点击【控制设备】按钮，弹出界面，选择控制权为远程 Modbus 单元。

应用篇 3：典型应用案例

应用篇 3：典型应用案例 - 目 录

第 1 章 编程技巧与分享

1.1 编程框架思路与解析

1.1.1 概述

在 OEM 的研发或者批量设备中，需要用到机器人的项目大多都存在 PLC，因此 PLC 和机器人间存在信号交互，以实现对机器人状态的监控和指令的下发。目前我司的机器人控制器支持 IO 交互、TCP/IP 交互、Modbus RTU、Modbus TCP 及 MC 通讯协议等多种交互方式，可满足目前主流品牌 PLC 之间的交互。

机器人和 PLC 之间的交互包含以下两种方式：

- 第一种是机器人周边的夹具，如吸嘴、气缸等 IO 控制元件由机器人控制，机器人自成一套逻辑；
- 第二种是机器人周边的夹具由 PLC 控制，机器人只是作为执行机构，机器人到位后给定到位信号，逻辑控制由 PLC 完成。

第一种方案，机器人自成一套逻辑，需要专业的机器人工程师完成程序的开发，机器人异常情况下的交互需要和 PLC 做好相应的处理，由于交互信号较多，对后期功能改变、优化和维护均增加一定的工作量；而第二种方案，机器人只作为执行机构，到位后给定到位信号即可，逻辑组合由 PLC 完成，机器人侧只是各动作的并列，一般工程师即可完成相应的维护，程序思路简单明了，对后期的功能的改变、优化和维护提供了很大的便捷。

针对以上第一种方式，一般工程师都拥有该思路，此处不做多余赘述，本文主要阐述第二种工作方式，以此搭建编程框架，做到程序逻辑简单明了、维护便捷的目的。

2 编程框架介绍

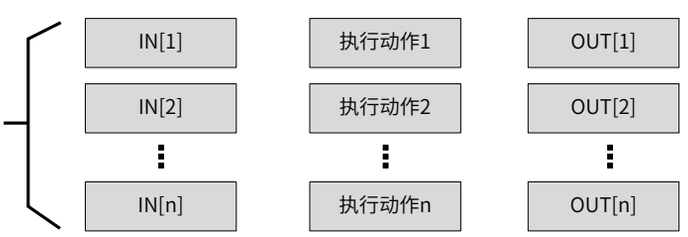
机器人只作为执行机构，即运动部件。通过预先编程的动作程序，由 PLC 侧给定不同的信号，机器人收到信号后运动到相应的位置，并作为反馈信号，然后由 PLC 控制器机器人外围夹具，再次给定机器人运动信号，机器人再次运动，以此往复循环，要做到该工作方式，编程思路及编程框架上需满足以下三点。

执行机构	一问一答	严格握手
机器人做执行机构 尽量不进行或减少外围信号的控制	动作并行化，逻辑由 PLC 给定 机器人执行完动作后要回复相应的信号	做好交互信号握手 所有动作输入输出尽量一致，保证程序的一致性

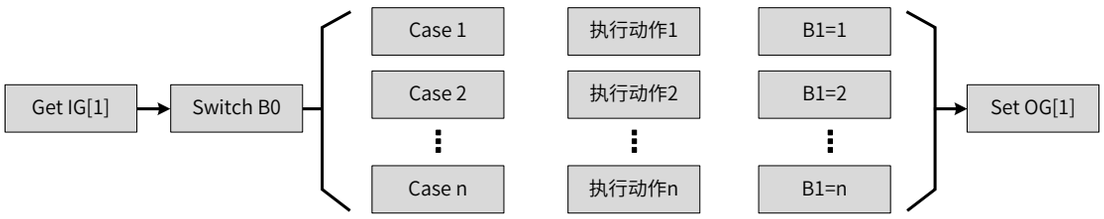
3 交互方式简介

针对现有的系统，我司的机器人控制器支持以下几种交互方式：

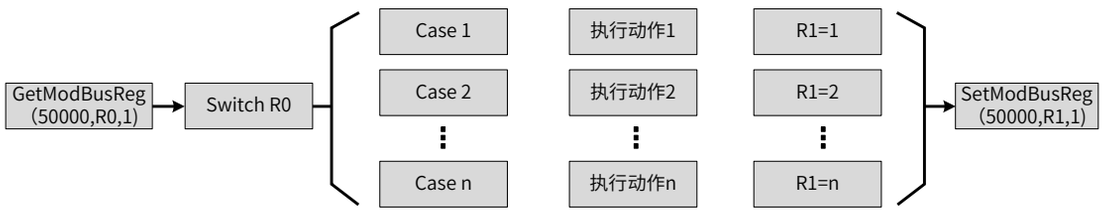
- 单个 IO 方式，PLC 侧给定一个输入信号，机器人执行一个动作，执行完毕后，机器人输出一个信号，以反馈给 PLC；



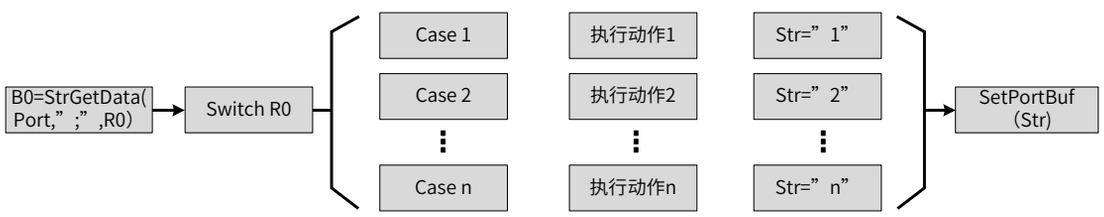
- 组合 IO 方式，PLC 侧给定组合信号输入，机器人执行一个动作，动作执行完毕后，机器人输出一个组合信号，以反馈给 PLC；



■ Modbus RTU/TCP 方式，PLC 侧给约定的地址空间写值，机器人执行相应的动作，执行完毕后，机器人在约定的地址空间写入一个值，以反馈给 PLC；



■ 以太网 TCP/IP 方式，PLC 侧发送约定的字符串，机器人执行相应的动作，执行完毕后，机器人发送约定的字符串，以反馈给 PLC；



各种交互方式优劣势对比如下。

表 1-5 各种交互方式优劣势对比

交互方式	优点	缺点	备注
单个 IO 方式	稳定可靠速度快	占用 IO 较多，接线多	
组合 IO 方式	组合方式，程序整体性好，易理解	存在单个或多个 IO 损坏后，误动作的风险	
Modbus RTU/TCP	寄存器直接读写，减少接线	编程要求高，对通讯有一定理解	
以太网 TCP/IP 方式	字符串方式读写，减少接线	编程要求高，对通讯有一定理解	

■ 注意事项

在与 PLC 进行交互确定编程框架时，请注意：

1） 尽量将机器人周边的 IO 信号转移到 PLC 侧；

2） 做好点位规划，输出点位信息表格；

3） 选择合适的交互方式；

4） 搭建程序框架，要注意握手信号；

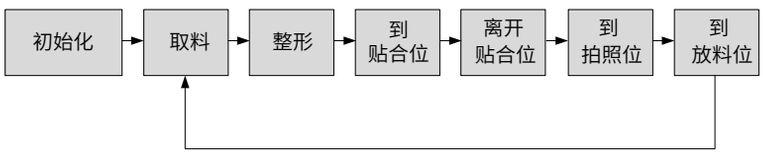
5） 务必关注初始化、各种变量及信号的复位，并考虑各种情况下的复位动作的路径规划。

1.1.2 编程实例

1 单个 IO 交互方式

该实例中共有 7 个点位，每个动作之间相对并列，由 PLC 指定执行动作顺序，每个动作到位后给定相应的输出信号，以反馈机器人的位置给 PLC。其相关信息的表格如下：

该项目的执行工艺如下：



在程序中，以上的动作逻辑顺序由 PLC 给定，在机器人侧，以上动作均为并列，且由 PLC 给定逻辑执行，用户只需理解其中一个动作的程序即可，后续的功能优化和动作添加可模仿该动作的编程方式完成。

表 1-6　IO 点位信息

输出 输入信号		点位意义	输出 - 输入信号	
PLC 侧	机器人侧		机器人侧	PLC 侧
Y1.0	IN[11]	原点位，也是安全位	OUT[11]	X1.0
Y 1.1	IN[12]	取料	OUT[12]	X1.1
Y 1.2	IN[13]	整形	OUT[13]	X1.2
Y 1.3	IN[14]	到封腔贴合位	OUT[14]	X1.3
Y 1.4	IN[15]	离开封腔贴合位	OUT[15]	X1.4
Y 1.5	IN[16]	到拍照位	OUT[16]	X1.5
Y 1.6	IN[17]	到放料位	OUT[17]	X1.6
注：该种应用方式，为了避免在调试过程中，动作有增加，建议做多 20%IO 裕量规划				

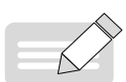
■　程序示例

-311-

```
START;
L[0]:
If IN[11]==ON                                                                    ## IN[11] 原点
Set Out[12],OFF;1·
Set Out[13],OFF;
Set Out[14],OFF;                                                                 ## 考虑不同的位置时的复位操作
Set Out[15],OFF;
Set Out[16],OFF;
Set Out[17],OFF;                                                                 ## 到位后置相应的输出信号
GetCurPoint(2,1,D0);                                                             ##IN[12 取料
If D0>400
Jump P[4],V[30],Z[0],Out(7,ON,T[0.01]),Out(7,OFF,T[-0.01]),LH[10],MH[-60],RH[10];
EndIf;
Jump P[0],V[30],Z[0],Out(7,ON,T[0.01]),Out(7,OFF,T[-0.01]),LH[50],MH[-50],RH[50];
WaitInPos;
Set Out[11],ON;                                                                  ## 到位后置相应的输出信号
EndIf;                                                                           ##IN[13] 整形
If IN[12]==ON
Set Out[11],OFF;
Set Out[13],OFF;
Set Out[14],OFF;
Set Out[15],OFF;
Set Out[16],OFF;                                                                 ## 到位后置相应的输出信号
Set Out[17],OFF;                                                                 ##IN[14] 到封腔贴合位
Jump P[1],V[B100],Z[0],Out(7,ON,T[0.01]),Out(7,OFF,T[-0.01]),LH[20],MH[-50],RH[20];
WaitInPos;
Set Out[12],ON;
EndIf;                                                                           ## 到位后置相应的输出信号
If IN[13]==ON                                                                    ##IN[15] 离开封腔贴合位
Set Out[11],OFF;
Set Out[12],OFF;
Set Out[14],OFF;
Set Out[15],OFF;
Set Out[16],OFF;
Set Out[17],OFF;                                                                 ## 到位后置相应的输出信号
Jump P[2],V[B100],Z[0],Out(7,ON,T[0.01]),Out(7,OFF,T[-0.01]),LH[20],MH[-60],RH[20];  ##IN[16]  到拍照位
WaitInPos;
Set Out[13],ON;
EndIf;
If IN[14]==ON                                                                    ## 到位后置相应的输出信号
Set Out[11],OFF;                                                                 ##IN[17]  到放料位
Set Out[12],OFF;
Set Out[13],OFF;
Set Out[15],OFF;
Set Out[16],OFF;                                                                 ## 到位后置相应的输出信号
Set Out[17],OFF;
JumpL P[3],V[B100],Z[0],Out(7,ON,T[0.01]),Out(7,OFF,T[-0.01]),LH[20],MH[-
60],RH[20];
WaitInPos;
Set Out[14],ON;
EndIf;
If IN[15]==ON
Set Out[11],OFF;
Set Out[12],OFF;
Set Out[13],OFF;
Set Out[14],OFF;
Set Out[16],OFF;
Set Out[17],OFF;
JumpL P[4],V[B100],Z[0],Out(7,ON,T[0.01]),Out(7,OFF,T[-0.01]),LH[20],MH[-
60],RH[20];
WaitInPos;
Set Out[15],ON;
```


-312-

```
EndIf;
If IN[16]==ON
Set Out[11],OFF;
Set Out[12],OFF;
Set Out[13],OFF;
Set Out[14],OFF;
Set Out[15],OFF;
Set Out[17],OFF;
Jump P[5],V[B100],Z[0],Out(7,ON,T[0.01]),Out(7,OFF,T[-0.01]),LH[20],MH[-60],RH[20];
WaitInPos;
Set Out[16],ON;
EndIf;
If IN[17]==ON
Set Out[11],OFF;
Set Out[12],OFF;
Set Out[13],OFF;
Set Out[14],OFF;
Set Out[15],OFF;
Set Out[16],OFF;
GetCurPoint(2,1,D0);
Jump P[6],V[B100],Z[0],Out(7,ON,T[0.01]),Out(7,OFF,T[-0.01]),LH[20],MH[-50],RH[20];
WaitInPos;
Set Out[17],ON;
EndIf;
Goto L[0];
END;
```



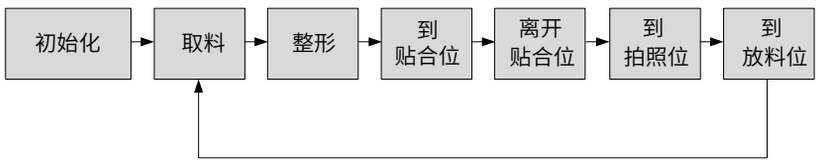
NOTE

◆ 程序中使用到的 Waitinpos 指令，在 16 及以后版本可以不使用。

2 组合 IO 交互方式

该实例中共有 7 个点位，每个动作之间相对并列，由 PLC 指定执行动作顺序，每个动作到位后将给定相应的输出信号，以反馈机器人的位置给 PLC。其相关信息的表格如下：

该项目的执行工艺如下：



在程序中，以上的动作逻辑顺序由 PLC 给定，在机器人侧，以上动作均为并列，并由 PLC 给定逻辑执行，用户只需理解其中一个动作的程序即可，后续的功能优化和动作添加可以模仿该动作的编程方式完成。

表 1-7　组合 IO 规划表

	输出——》输入			输入——》输出			备注
	PLC	机器人	功能	PLC	机器人	功能	
1# 模块	Y1.0	IN0	急停（系统 , 常闭）	X1.0	OUT0		
	Y1.1	IN1	使能（系统 ）	X1.1	OUT1		
	Y1.2	IN2		X1.2	OUT2		
	Y1.3	IN3	启动	X1.3	OUT3	报警状态输出	
	Y1.4	IN4	暂停	X1.4	OUT4	机器人启动完成	
	Y1.5	IN5	停止	X1.5	OUT5		
	Y1.6	IN6	清除报警	X1.6	OUT6		
	Y1.7	IN7	速度切换	X1.7	OUT7	机器人接收到组合信号输出	
2# 模块	Y2.0	IN8	组合交互启动确认	X2.0	OUT8	SetOG 组合输出 完成对应的动作信号	4 路编码组合，可走点位 15 个，点位信息自由定义
	Y2.1	IN9	程序调用（短接）	X2.1	OUT9		
	Y2.2	IN10		X2.2	OUT10		
	Y2.3	IN11		X2.3	OUT11		
	Y2.4	IN12	SetIG 组合输入 动作执行信号	X2.4	OUT12		
	Y2.5	IN13		X2.5	OUT13		
	Y2.6	IN14		X2.6	OUT14		
	Y2.7	IN15		X2.7	OUT15		

■ 注意：

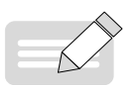
1） 目前指令默认仅支持 8 位一组，程序中通过移位的思维变为 4 位一组。

2） IN[8] 是组合确认，因为 PLC 给定的组合输出有时间差，为了确保 IO 的稳态，做了一个 IN[8] 信号，以确保动作正确，OUT[7] 是收到 PLC 信号机器人并运动后的反馈。

■ 程序示例

```
START;
B1 =0;
Set OG[1],B1;
Set Out[7],OFF;
L[0]:                                                    ##IN[8] 是 PLC 给定机器人 IO 完毕后的确
If IN[8]==ON                                             认信号，确保 IO 信号给定，防止误动作
Set IG[1],B0;
B0 =B0/16;                                               ## 初始原位，Out[7] 是机器人控制器收到
If B0==1                                                 PLC 信号的反馈，运行到位清除为 OFF
Jump P[0],V[30],Z[0],Out(7,ON,T[0.01]),LH[80],MH[-1],RH[80];
WaitInPos;                                               ## 初始化动作，B0=1 时执行初始化，初
B1 =1;                                                   始化完成置 R0 为 1 标志
R0 =1;
Set OG[1],B1;
Set Out[7],OFF;
EndIf;
If R0==1
Switch B0
Case 2 :                                                 ## 根据 B0 的值切换到相应的正常点
B0 =0;                                                   ## 料盘取料位
Jump P[1],V[B100],Z[5],Out(7,ON,T[0.01]),LH[15],MH[-5],RH[15];
B1 =2;
WaitInPos;
Set OG[1],B1;
Set Out[7],OFF;
Break;
 Case 3 :                                                ## 转盘上料位上方等待点
B0 =0;
PR2 = (0,0,40,0,0,0);
Jump Offset(P[3],PR2),V[B100],Z[5],Out(7,ON,T[0.01]),LH[15],MH[-10],RH[0];
B1 =3;
WaitInPos;
Set OG[1],B1;
Set Out[7],OFF;
Break;
Case4 :                                                  ## 转盘上料位
B0 =0;
If B1==3
Movl P[3],V[B100],Z[5],Out(7,ON,T[0.01]);
Else
Jump P[3],V[B100],Z[5],Out(7,ON,T[0.01]),LH[15],MH[-10],RH[15];
EndIf;
B1 =4;
WaitInPos;
Set OG[1],B1;
Set Out[7],OFF;
Break;                                                   ## 转盘下料位取料
Case 5 :
B0 =0;
Jump P[4],V[B100],Z[5],Out(7,ON,T[0.01]),LH[15],MH[-10],RH[15];
B1 =5;
WaitInPos;
Set OG[1],B1;
Set Out[7],OFF;                                          ## 转盘下料位上方等待点
Break;
Case 6 :
B0 =0;
PR3 = (0,0,30,0,0,0);
Jump Offset(P[4],PR3),V[B100],Z[5],Out(7,ON,T[0.01]),LH[15],MH[-10],RH[15];
B1 =6;
WaitInPos;
Set OG[1],B1;
Set Out[7],OFF;                                          ## 转盘下料位取料
Break;
```

```
Case 7 :
B0 =0;
If B1==6
Movl P[4],V[B100],Z[5],Out(7,ON,T[0.01]);
Else
Jump P[4],V[B100],Z[5],Out(7,ON,T[0.01]),LH[15],MH[-10],RH[15];
EndIf;
B1 =7;
WaitInPos;
Set OG[1],B1;
Set Out[7],OFF;
Break;
EndSwitch;
EndIf;
EndIf;
Goto L[0];
END;
```



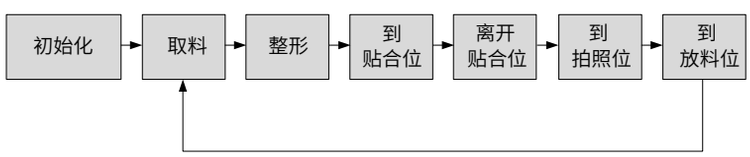
NOTE

◆ 程序中使用到的 switch-case 或等同功能语句，在进入所在分支后，需要把变量赋成其他值，并避免再次进入。

3 Modbus RTU/TCP 交互方式

该实例中共有 7 个点位，每个动作之间相对并列，由 PLC 指定执行动作顺序，每个动作到位后给定相应的输出信号，以反馈机器人的位置给 PLC。其相关信息的表格如下：

该项目的执行工艺如下：



在程序中，以上的动作逻辑顺序由 PLC 给定，在机器人侧，以上动作均为并列，且由 PLC 给定逻辑执行，用户只需理解其中一个动作的程序即可，后续的功能优化和动作添加可模仿该动作的编程方式完成。

表 1-8　Modbus 交互表

输出——》输入			输出——》输入		
PLC	机器人	功能	机器人	PLC	功能
50000 R0		读取 PLC 给定的输入命令	R1 50001		输出给 PLC 运动到位信号

■　程序示例

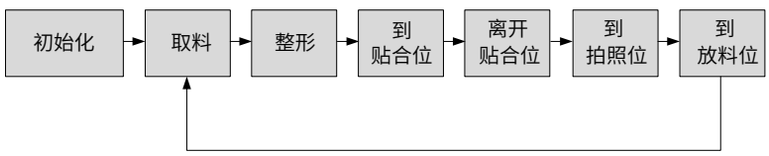
```
START;
L[0]:
GetModBusReg (50000,R0,1);                  ##R0 为 PLC 给定的动作顺序
Switch R0
Case 1 :                                    ## 初始化，回安全位
Jump P[1],V[30],Z[0],LH[40],MH[-10],RH[40];
WaitInPos;
R1 =1;
SetModBusReg (50001,R1,1);                  ## 置 50001 为 1 ，表示动作完成，同时清除 50000 的值为 0
SetModBusReg (50000,0,1);
Break;                                      ## 料盘取料
 Case 2 :
Jump P[2],V[30],Z[0],LH[40],MH[-10],RH[40];
WaitInPos;
R1 =2;                                      ## 置 50001 为 2 ，表示动作完成，同时清除 50000 的值为 0
SetModBusReg (50001,R1,1);                  ## 转盘上料等地点
SetModBusReg (50000,0,1);
 Case 3 :
Jump P[3],V[30],Z[0],LH[40],MH[-10],RH[40];
WaitInPos;                                  ## 置 50001 为 2 ，表示动作完成，同时清除 50000 的值为 0
R1 =3;
SetModBusReg (50001,R1,1);                  ## 转盘上料
SetModBusReg (50000,0,1);
Break;

Case 4 :
Jump P[4],V[30],Z[0],LH[40],MH[-10],RH[40]; ## 置 50001 为 3 ，表示动作完成，同时清除 50000 的值为 0
WaitInPos;
R1 =4;                                      ## 转盘下料等待
SetModBusReg (50001,R1,1);
SetModBusReg (50000,0,1);
Break;
 Case 5 :                                   ## 置 50001 为 5 ，表示动作完成，同时清除 50000 的值为 0
Jump P[5],V[30],Z[0],LH[40],MH[-10],RH[40];
WaitInPos;                                  ## 转盘下料
R1 =5;
SetModBusReg (50001,R1,1);
SetModBusReg (50000,0,1);
Break;                                      ## 置 50001 为 6 ，表示动作完成，同时清除 50000 的值为 0
 Case 6 :
Jump P[6],V[30],Z[0],LH[40],MH[-10],RH[40]; ## 料盘下料
WaitInPos;
R1 =6;
SetModBusReg (50001,R1,1);
SetModBusReg (50000,0,1);                   ## 置 50001 为 6 ，表示动作完成，同时清除 50000 的值为 0
Break;
 Case 7 :
Jump P[7],V[30],Z[0],LH[40],MH[-10],RH[40];
WaitInPos;
R1 =7;
SetModBusReg (50001,R1,1);
SetModBusReg (50000,0,1);
Break;
EndSwitch;
Goto L[0];
END;
```

4 以太网 TCP/IP 交互方式

该实例中共有 7 个点位，每个动作之间相对并列，由 PLC 指定执行动作顺序，每个动作到位后给定相应的输出信号，以反馈机器人的位置给 PLC。其相关信息的表格如下：

该项目的执行工艺如下：



在程序中，以上的动作逻辑顺序由 PLC 给定，在机器人侧，以上动作均为并列，且由 PLC 给定逻辑执行，用户只需理解其中一个动作的程序即可，后续的功能优化和动作添加可模仿该动作的编程方式完成。

TCP/IP 交互方式可选机器人作为客户端或作为服务器端，具体介绍请参考机器人使用说明书，此处以作客户端为例进行阐述。

表 1-9 信息交互表

输出——》输入			输出——》输入		
PLC	机器人	功能	机器人	PLC	功能
B1 = StrGetData(Port,”；",R0);		读取 PLC 给定的输入命令到 R0	SetPortBuf(Str) ; Send Port[3000];		通过赋值 Str，然后输出给 PLC

■ 程序示例

```
START;
String Str;                                    ## 定义一个字符串
While  B0==0
Open Socket("192.168.1.55",3000,3000,B0) ;     ## 机器人作为客户端需要，否则不需要
EndWhile;
L[0]:
B1 = StrGetData(Port,”；”,R0);                 ## 约定数据格式，以分号结尾，将输入信号放到 R0
If B1<>0                                       ##R0 为 PLC 给定的动作顺序
Switch R0                                      ## 初始化，回安全位
Case 1 :                                       ## 添加初始化动作，注意不同停顿点初始化
Jump P[1],V[30],Z[0],LH[40],MH[-10],RH[40];    ## 置 50001 为 1 ，表示动作完成，同时清除 50000 的值为 0
WaitInPos;
Str = "1";
SetPortBuf(Str) ;                              ## 料盘取料
Send Port[3000];
Break;
Case 2 :
Jump P[2],V[30],Z[0],LH[40],MH[-10],RH[40];
WaitInPos;
Str = "2";                                     ## 料盘取料
SetPortBuf(Str) ;
Send Port[3000];
 Case 3 :
Jump P[3],V[30],Z[0],LH[40],MH[-10],RH[40];
WaitInPos;
Str = "3";
SetPortBuf(Str) ;                              ## 转盘上料
Send Port[3000];
Break;
Case 4 :
Jump P[4],V[30],Z[0],LH[40],MH[-10],RH[40];
WaitInPos;
Str = "4";
SetPortBuf(Str) ;                              ## 转盘下料等待
Send Port[3000];
Break;
 Case 5 :
Jump P[5],V[30],Z[0],LH[40],MH[-10],RH[40];
WaitInPos;
Str = "5";
SetPortBuf(Str) ;                              ## 转盘下料
Send Port[3000];
Break;
 Case 6 :
Jump P[6],V[30],Z[0],LH[40],MH[-10],RH[40];
WaitInPos;
Str = "6";
SetPortBuf(Str) ;                              ## 料盘下料
Send Port[3000];
Break;
 Case 7 :
Jump P[7],V[30],Z[0],LH[40],MH[-10],RH[40];
WaitInPos;
Str = "7";
SetPortBuf(Str) ;
Send Port[3000];
Break;
EndSwitch;
EndIf;
Goto L[0];
END;
```

1.2 多任务应用

1.2.1 概述

在机器人实际使用工况中，若只是简单应用（与外围设备——IO、上位机或 PLC 进行简单交互），无复杂逻辑时只需机器人的主任务即可。但若遇到复杂应用（与外围设备——IO、上位机或 PLC 进行复杂或有时序交互时），机器人需要适用多个任务同时运行。 其中主任务负责执行机器人的运动，另外几个“PLC 任务”负责外界逻辑处理，且由于 PLC 的特性，PLC 任务是不断循环运行的。

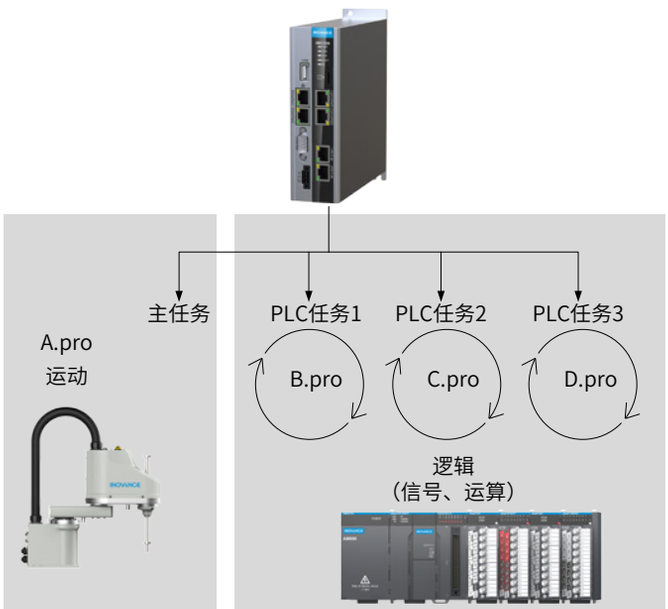


图 1-4 机器人多任务示意图

■ 任务分类：

机器人支持多个任务同时运行，按任务的类型分为以下 3 类：

1） 任务 0：机器人的主任务。通常直接运行的程序默认为主任务，该任务始终激活，多 用于负责进行机器人运动。受启动、停止、暂停命令影响。（注意区分命令与指令，命令： 界面操作，指令：程序中的指令）

2） 任务 1、2：设置型 PLC 任务。 静态启动：上电立即启动，并一直运行，只能通过页面关闭“激活”停止。非静态启动：上电不立即启动，与主任务相同，受启动、停止、暂停命令影响。

图 1-5 设置型 PLC 任务设置界面

3） 任务 3：指令型 PLC 任务。受指令控制启停，且受停止命令控制。

```
001 START;
002 Movj P[5],V[30],Z[0];
003 Xqt 3,"tr.pro";
004 Wait In[3]==ON,T[0];
005 Delay T[2];
006 Halt 3;
007 Wait In[3]==ON,T[0];
008 Delay T[2];
009 Quit 3;
010 Wait In[3]==ON,T[0];
011 Delay T[2];
012 Pause;
```

图 1-6 指令型 PLC 任务编程启动、停止、退出程序

涉及相关指令： Xqt、Halt、Quit，详细请参见《汇川机器人设计应用与维护手册 -- 附录：机器人指令表》。

由上所述，PLC 任务与任务 0 执行存在关联性：

表 1-10 PLC 任务与任务 0 关联性

任务 0	受启动、暂停、停止命令控制
静态启动的 PLC 任务	开机一直运行，只有界面“激活”按钮影响，不受任务 0 影响
非静态启动的 PLC 任务	随任务 0 同步启动、暂停、停止
指令型 PLC 任务	依靠指令控制启停，此外，也会随着任务 0 停止而停止

1.2.2 多任务的使用

1 常见用法

机器人程序中一个负责运动的任务作为主任务，一个或多个 PLC 任务作为逻辑任务。主任务随着 PLC 任务处理得到的数据时时变更运动。

■ 案例一：根据多任务的通讯数据选择性运动

表 1-11 案例一应用说明

场景	控制器同时进行两个任务：一边不断与外部通讯，根据接受外的部数据值是否为 201，置对 应标志；另一边不断根据标志值，选择性进行运动 Movj P[1] 与 Movj P[0]。
设计	主任务，一直运行，在循环体内不断判断某一全局数值变量值（如下 B4），根据数值选择性 执行运动。
PLC 任务	作为 PLC 任务，不断与外界通讯，根据读取的结果，置 B4 值。

表 1-12 案例一多任务程序

主任务：Motion.pro	PLC 任务：Communication.pro

START;
Movj P[2],V[30],Z[0]; L[0]:
If B4 == 2
Movj P[0],V[30],Z[0];
EndIf;
If B4 == 4
Movj P[1],V[30],Z[0];
EndIf; Goto L[0]; END;
	START;
	String strs=“request”; String recv=“000000”;
	L[1]: ##PLC 任务特性，自身不断循环，此行也可去掉
	SetPortBuf(strs) ; Send Port[4444]; L[0]:
	Get Port[4444],T[0],Goto L[0];
	recv = GetPortbuf(0,100); If StrToR(recv) == 201
	B4 =4;
	Else
	B4 =2;
	EndIf;
	Goto L[1]; ##PLC 任务特性，自身不断循环，此行也可去掉
	END;

[1] 对于上述 PLC 任务，若将其作为任务 1（或 2），并置为静态任务：上电立即运行，但主任务（任务 0）不会运行。使用时需要启动命令（比如按示教器上的启动按钮）启动主任务。需要结束时，停止命令停 止主任务，激活按钮关闭 PLC 任务； [2] 若置为非静态任务：通过启动命令，两个任务同步进行。需要结束时，通过停止命令结束。 两个任务的控制完全同步； [3] 若将与外界通讯的任务作为任务 3，则任务 0 程序应在开头增加：Xqt 3,“Communicatin.pro”。	

2 特殊应用

远程模式时，运行 PLC 任务，在 PLC 任务中利用 xqt 指令启动主任务。此时该功能可以代替工位预约功能。

■ 案例二： 利用多任务模仿实现工位预约功能场景：

表 1-13 案例二应用说明

场景	利用多任务模仿实现工位预约功能，控制器同时进行两个任务：一个用来不断检测 IO，控制程序的暂停与退出；另一边根据 IO 选择性执行子程序。
设计	在 PLC 任务中启用主任务，主任务检测外部输入信号，调用对应程序当做工位预约子程序。
PLC 任务	PLC 任务用来不断检测输入信号，控制暂停、停止。设置该 PLC 任务为静态启动，上电立即运行。

表 1-14 案例二多任务程序

主任务：tr.pro	PLC 任务：duorenwu.pro

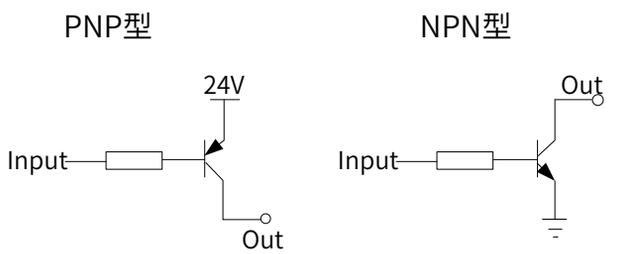
START;
L[0]:
If IN 4 == ON
Call a.pro;
EndIf;
If IN 5 == ON
Call b.pro;
EndIf;
Goto L[0];
END;
	START;
	L[2]: ##PLC 任务特性，自身不断循环，此行也可去掉
	If IN 6 == ON
	Xqt 0,”tr.pro”;
	EndIf;
	If IN 7 == ON
	Halt 0;
	EndIf;
	If IN 8 == ON
	Quit 0;
	EndIf;
	Goto L[2]; ##PLC 任务特性，自身不断循环，此行也可去掉
	END;

[1] 对于 PLC 任务：可通过 IN 6、IN7、IN8 控制主任务的启动、暂停、停止； [2] 对于主任务：可通过 IN4、IN5 调用对应子程序 a.pro、b.pro。	

1.3 NPN 输入输出应用

1.3.1 概述

机器人作为单体系统设备，在应用过程中与 PLC 等进行交互工作。市面上常见 IO 交互除了采用低电平 NPN 型外，对于 PLC 则普遍采用 PNP 高电平信号进行交互。PNP 与 NPN 区别如下。



■ PNP 与 NPN 内部电路结构

结构上，NPN 三极管的中间是 P 区，两端是 N 区，而 PNP 三极管正相反。使用上，NPN 三极管工作时是集电极接高电压，发射极接低电压，基极输入电压升高时趋向导通，基极输入电压降低时趋向截止；而 PNP 三极管工作时则是集电极接低电压，发射极接高电压，基极输入电压升高时趋向截止，基极输入电压降低时趋向导通。即 NPN 和 PNP 主要就是电流方向和电压正负不同。
NPN 是用 B → E 的电流（IB）控制 C → E 的电流（IC），E 极电位最低，且正常放大时通常 C 极电位最高，即 VC > VB > VE
PNP 是用 E → B 的电流（IB）控制 E → C 的电流（IC），E 极电位最高，且正常放大时通常 C 极电位最低，即 VC < VB < VE

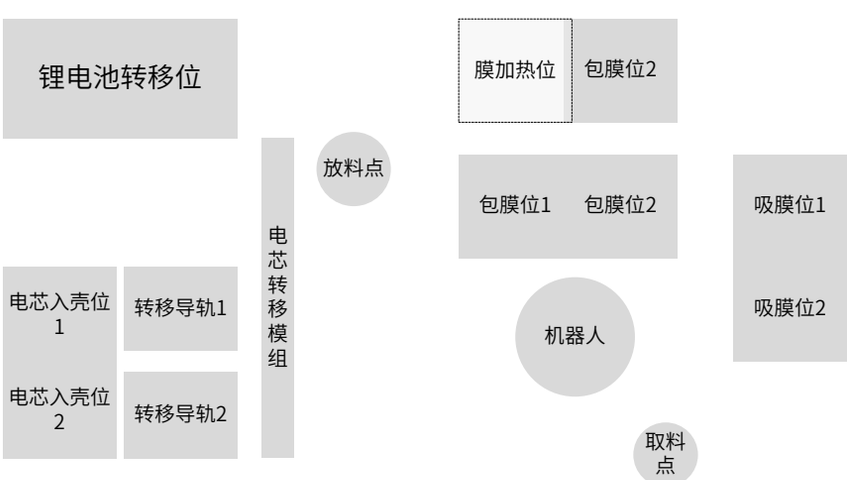
NPN 型主要为漏型输出，即集电极开路输出方式，一般控制地 GND；

PNP 型主要为源型输出，主要控制电源。

两者相对来说，PNP 型在内部电路上对电源抗干扰和滤波较好。而 NPN 型没有抗干扰及滤波部分，只能通过外部增加滤波器进行处理，稳定性较弱一点。我们目前电控柜主要以 NPN 漏型输出作为主要应用，本文主要针对 PNP 源型输出处理进行说明。

1.3.2 数据配置

如下图所示，SCARA600 机器人与施耐德 PLC 进行 IO 交互（PNP 方式），输入需要采用跳帽方式实现，输出可采用两种方式：①采用 modbus TCP/RTU 进行寄存器地址交互；②采用继电器（主要 PNP 型继电器）中转方式。本文主要将以非继电器中转方式、采用通讯方式处理全部输入输出状态量为例进行阐述。



位置变量		点位含义	
P[0]		机器人上升位 / 下降位，根据 D2 高度给定决定	
P[1]		复位至安全位	
P[2]		电芯取料位	
P[3]		吸膜位	
P[4]		包膜 1 放	
P[5]		包膜 2 放	
P[4]		包膜 1 取	
P[5]		包膜 2 取	
P[6]		电芯下料位	
D 变量	变量含义	R 变量	变量含义
D0	当前点位 X 值	R0	modbus 49152 接收
D1	当前点位 Y 值	R1	modbus 49154 发送
D2	当前点位 Z 值	R2	电芯下料 / 上料位判断
D3	上升 / 下降高度赋值	R3	下料区域判断 modbus 49156 发送

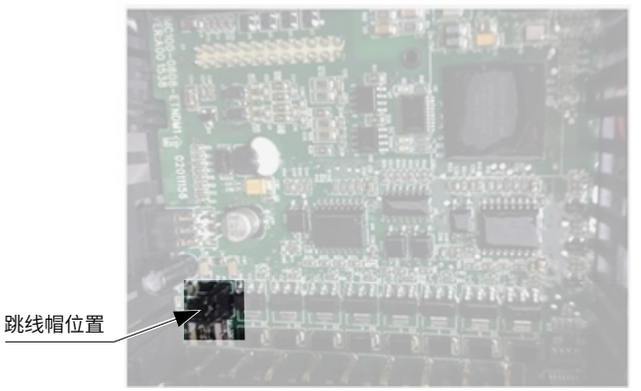
■ 输入状态：

输入地址	R0	执行动作
MW49152	1	电芯取料
	2	吸膜
	3	包膜 1 放
	4	包膜 2 放
	5	包膜 1 取
	6	包膜 2 取
	7	电芯下料
	8	当前位抬升
	9	复位至安全位

■ 输出状态：

输出地址	R1		执行动作
MW49154	1		电芯取料完成
	2		吸膜完成
	3		包膜 1 放完成
	4		包膜 2 放完成
	5		包膜 1 取完成
	6		包膜 2 取完成
	7		电芯下料完成
	8		当前位抬升完成
	9		复位至安全位完成
机器人状态输出地址	输出状态	bit 位	
MW49158	使能	1	1_ 表示状态已触发 0_ 表示状态未触发
	报警	1	
	运行	1	
	停止	1	

■ 跳线帽位置：



■ 接线方式：

不带示教器应用时：第一个 0808 模块 X0 短接 24V，第二个模块全部采用外部供电方式；

带示教器应用时：将示教器急停接线反接，即接线端子上急停 + 接 0V，急停 - 接 24V。

1.3.3 程序实例

```
START;
## 变量复位
R0 =0;
R1 =0;
R2 =0;
R3 =0;
L[1]:
## 读取当前位置信息
GetCurPoint(2,0,P[0]);
## 读取当前位置 X 值
GetCurPoint(2,0,D0);
## 读取当前位置 Y 值
GetCurPoint(2,1,D1);
## 读取当前位置 Z 值
GetCurPoint(2,2,D2);
## 判断是否处于下料区域，是 R3=1，否 R3=0
If D1>185
R3 =1;
SetModBusReg (49156,R3,2);
Else
R3 =0;
SetModBusReg (49156,R3,2);
EndIf;
## 获取 PLC 输入信息，执行对应动作
L[0]:
GetModBusReg (49152,R0,2);
If R0==0
Goto L[0];
EndIf;
Print R0;
##switch 判断接收结果，1- 电芯取料，2- 吸膜，3- 包膜 1 放，4- 包膜 2 放，5- 包膜 1 取
##6- 包膜 2 取，7- 电芯下料，8- 抬升，9- 安全位
 Switch R0
 Case 1 :
## 电芯取料
If D2<=0
```

```
D3 =0;
SetPValue(P[0],2,D3);
## 高度抬升
Movj P[0],V[30],Z[0];
EndIf;
Jump P[2],V[30],Z[0],LH[0],MH[0],RH[20];
WaitInPos;
R1 =1;
R2 =1;
SetModBusReg (49154,R1,2);
L[2]:
GetModBusReg (49152,R0,2);
If R0<>0
Goto L[2];
EndIf;
Break;
 Case 2 :
## 吸膜
If D2<=0
D3 =0;
SetPValue(P[0],2,D3);
## 高度抬升
Movj P[0],V[30],Z[0];
EndIf;
Jump P[3],V[30],Z[0],LH[0],MH[0],RH[50];
R1 =2;
WaitInPos;
SetModBusReg (49154,R1,2);
L[3]:
GetModBusReg (49152,R0,2);
If R0<>0
Goto L[3];
EndIf;
Break;
 Case 3 :
## 包膜 1 放料
If D2<=0
D3 =0;
SetPValue(P[0],2,D3);
## 高度抬升
Movj P[0],V[30],Z[0];
EndIf;
Jump P[4],V[30],Z[0],LH[0],MH[0],RH[50];
WaitInPos;
R1 =3;
SetModBusReg (49154,R1,2);
L[4]:
GetModBusReg (49152,R0,2);
If R0<>0
Goto L[4];
EndIf;
Break;
 Case 4 :
## 包膜 2 放料
If D2<=0
D3 =0;
SetPValue(P[0],2,D3);
## 高度抬升
Movj P[0],V[30],Z[0];
EndIf;
Jump P[5],V[30],Z[0],LH[0],MH[0],RH[50];
```

```
WaitInPos;
R1 =4;
SetModBusReg (49154,R1,2);
L[5]:
GetModBusReg (49152,R0,2);
If R0<>0
Goto L[5];
EndIf;
Break;
 Case 5 :
## 包膜 1 取料
If D2<=0
D3 =0;
SetPValue(P[0],2,D3);
## 高度抬升
Movj P[0],V[30],Z[0];
EndIf;
Jump P[4],V[30],Z[0],LH[0],MH[0],RH[50];
WaitInPos;
R1 =5;
SetModBusReg (49154,R1,2);
L[6]:
GetModBusReg (49152,R0,2);
If R0<>0
Goto L[6];
EndIf;
Break;
 Case 6 :
## 包膜 2 取料
If D2<=0
D3 =0;
SetPValue(P[0],2,D3);
## 高度抬升
Movj P[0],V[30],Z[0];
EndIf;
Jump P[5],V[30],Z[0],LH[0],MH[0],RH[50];
WaitInPos;
R1 =6;
SetModBusReg (49154,R1,2);
L[7]:
GetModBusReg (49152,R0,2);
If R0<>0
Goto L[7];
EndIf;
Break;
 Case 7 :
## 电芯下料
If D2<=0
D3 =0;
SetPValue(P[0],2,D3);
## 高度抬升
Movj P[0],V[30],Z[0];
EndIf;
Jump P[6],V[30],Z[0],LH[0],MH[0],RH[50];
WaitInPos;
R2 =1;
R1 =7;
SetModBusReg (49154,R1,2);
L[8]:
GetModBusReg (49152,R0,2);
If R0<>0
```

```
Goto L[8];
EndIf;
Break;
 Case 8 :
## 高度抬升
If D2<=0
D3 =0;
SetPValue(P[0],2,D3);
## 高度抬升
Movj P[0],V[30],Z[0];
WaitInPos;
EndIf;
If R2==1
Jump P[1],V[30],Z[0],LH[0],MH[0],RH[50];
WaitInPos;
R2 =0;
EndIf;
R1 =8;
SetModBusReg (49154,R1,2);
L[9]:
GetModBusReg (49152,R0,2);
If R0<>0
Goto L[9];
EndIf;
Break;
 Case 9 :
## 复位至安全区
If D2<=0
D3 =0;
SetPValue(P[0],2,D3);
## 高度抬升
Movj P[0],V[30],Z[0];
EndIf;
Movj P[1],V[30],Z[0];
WaitInPos;
R1 =9;
SetModBusReg (49154,R1,2);
L[10]:
GetModBusReg (49152,R0,2);
If R0<>0
Goto L[10];
EndIf;
Break;
EndSwitch;
Goto L[1];
END;
```

第 2 章 通信连接与数据传输

2.1 TCP/IP 通讯应用

本案例为包装行业内一种具备贴合工艺的机型应用。方案主体架构为机器人 + 视觉系统，两者数据交互基于标准 TCP/IP 通信协议，标定功能由视觉完成，视觉将计算出的纠偏值发送给机器人，机器人内部计算后运动到最终的贴合位置。在此类应用中，机器人与视觉的 TCP/IP 通讯将涵盖三种不同的应用方式：①机器人作为服务器（客户端口号不固定）、②机器人作为服务器（客户端口号固定）、③机器人作为客户端。以下结合三个典型案列进行讲解。

2.1.1 案例一：纸盒生产过程中的贴合应用 1

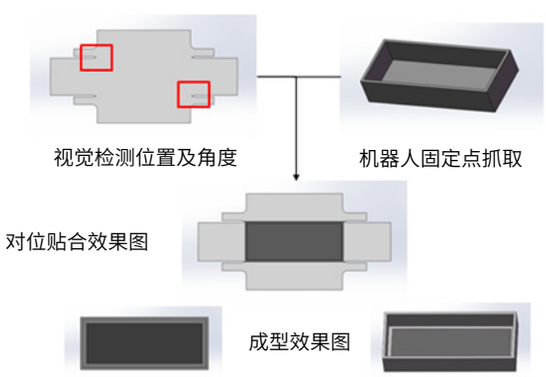
1 案例描述

1） 面纸通过传送带到达拍照位，视觉识别面纸对角进行检测定位，发送纠偏值给机器人。

2） 机器人从固定位置吸取纸盒，根据视觉发送的纠偏值，将纸盒准确的贴至面纸上。

3） 完成贴合后，经过成型机将面纸折叠，最终生产出一个纸盒成品。

4） 贴合示意图：



2 步骤说明

1） 面纸通过传送带到达拍照位，视觉进行拍照和创建模板，保证面纸不动，机器人从固定位置吸取纸盒，然后示教纸盒位置到面纸上，初步通过人工创建一个贴合基准位置。

2） 手动测试：将面纸进行位置和角度偏移，视觉检测后发送基于模板的纠偏量给机器人，机器人在贴合基准位置基础上，加上纠偏量，然后移动到贴合位置，完成贴合。

3） 自动测试：传送带连续运行，重复多次贴合动作。根据成型效果，在视觉软件上进行偏移补偿，满足精度要求后，开始稳定生产。

3 数据配置

1） 输入输出信号（按照对应端子进行接线）

■ 输入

In(3)	急停
In(4)	暂停

In(5)	启动
In(6)	停止
In(7)	清除报警
In(11)	有盒信号
In(12)	真空检测
In(13)	程序选择

■ 输出

Out(2)	夹盒气缸
Out(3)	吸气
Out(4)	定位完成
Out(5)	顶盒气缸
Out(6)	吹气
Out(7)	程序运行中
Out(8)	停止
Out(9)	未吸到盒子报警

2） 点位信息

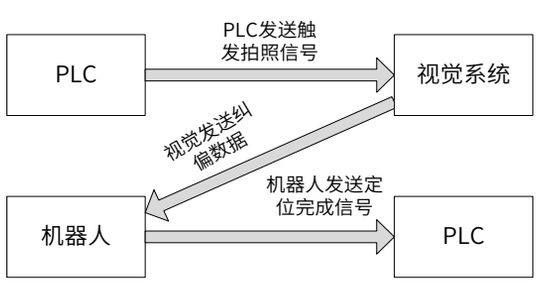
P0	存放取料高度和角度	需要示教（只示教取料高度和角度即可）
P1	取纸盒前上方等待位	
P2	纸盒取料位	
P3	放料基准位	需要示教
P4	纸盒抓取中心计算基准点	需要示教（配合尖端）
P5	存放纸盒规格尺寸	需要手动输入（长宽高依次对应 XYZ）
P6	贴合后上方等待位	
P7	贴合位	
P8	取完纸盒上方等待位	
P9	贴合前上方等待位	
P10	贴合前等待位高度	需要示教
P11	接收视觉数据等待点	
P12	接收视觉数据等待点 XY 坐标	需要示教

3） 数据交互流程说明

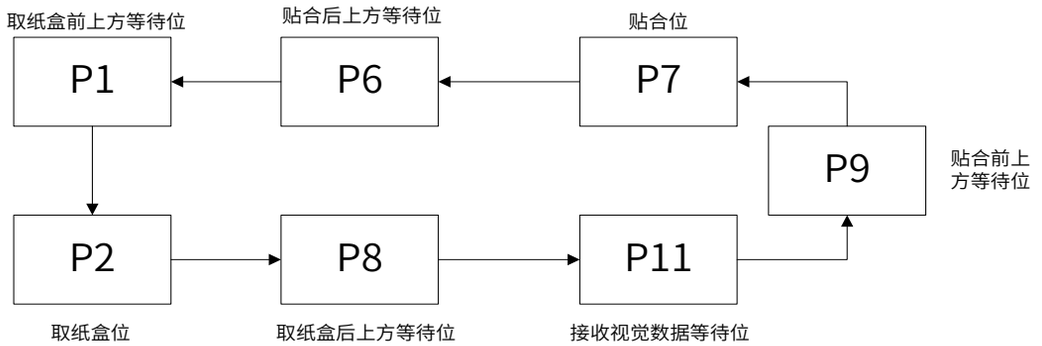
■ 机械手与视觉的通信为 TCP 网络通信。需要机械手提供对应的 IP 地址和端口号，且将机械手设为服务端、视觉为客户端（客户端端口号不固定）；

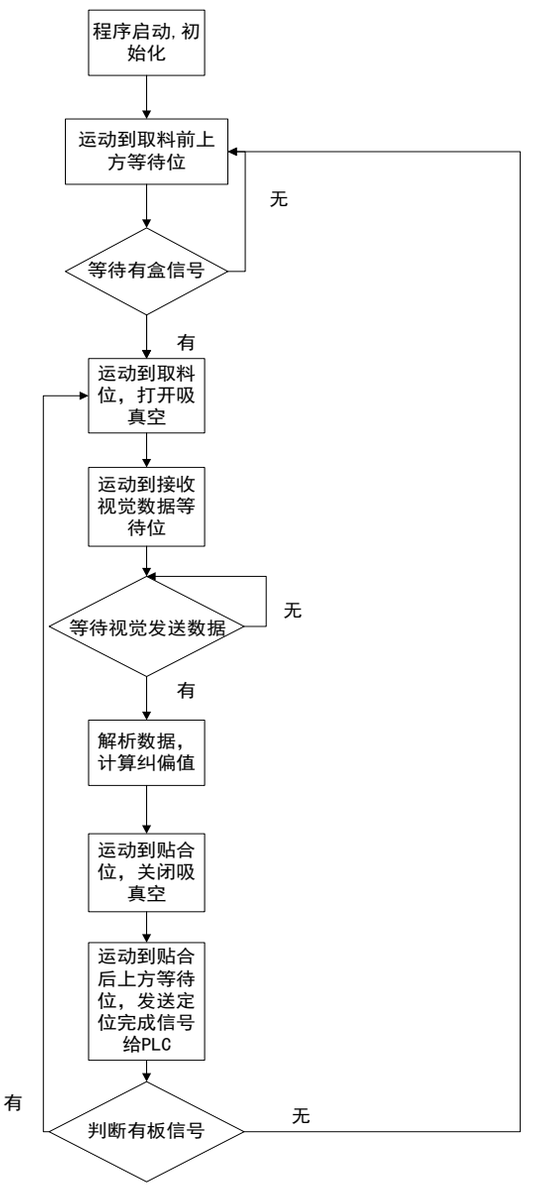
■ 打开视觉程序，通信连接成功；

■ 偏移量的格式由 7 位字符串组成，例如 :X:12.15，Y:-1.23，Q:2.35 字符串格式 :0012150，-001230，0002350 正号补零，负号保持不变，其他的方式不变，分隔符为“，”。



4 程序流程图


取纸盒前上方等待位
P1
贴合后上方等待位
P6
贴合位
P7
P9
贴合前上方等待位
P2
P8
P11
取纸盒位
取纸盒后上方等待位
接收视觉数据等待位


程序启动，初始化
运动到取料前上方等待位
无
等待有盒信号
有
运动到取料位，打开吸真空
运动到接收视觉数据等待位
等待视觉发送数据
无
有
解析数据，计算纠偏值
运动到贴合位，关闭吸真空
运动到贴合后上方等待位，发送定位完成信号给PLC
有
判断有板信号
无

5 程序实例

```
START;
GetPValue(P[3],0,D0);                          ## 获取贴合模板的坐标值
GetPValue(P[3],1,D1);
GetPValue(P[3],2,D2);
GetPValue(P[3],3,D3);

GetPValue(P[4],0,D4);                          ## 获取纸盒中心计算的基准值
GetPValue(P[4],1,D5);

GetPValue(P[5],0,D6);                          ## 获取纸盒长宽高
GetPValue(P[5],1,D7);
GetPValue(P[5],2,D8);

GetPValue(P[0],2,D9);                          ## 获取取料位高度
GetPValue(P[0],3,D91);

GetPValue(P[10],2,D92);                        ## 获取取完纸盒后上方等待点高度

GetPValue(P[12],0,D80);                        ## 获取接收视觉数据等待点 XY 坐标
GetPValue(P[12],1,D81);

D20 =D4+D6/2;                                  ## 纸盒取料点 XY 坐标计算
D21 =D5+D7/2;

D22 =D9+D8+20;                                 ## 纸盒取料上方等待点高度计算

D49 =D2+D8+20;                                 ## 贴合位上方等待点高度计算

P[1] =(D20,D21,D22,D91,0,0),(-1,0,0,0),(2,0,0);
P[2] =(D20,D21,D9,D91,0,0),(-1,0,0,0),(2,0,0);
P[8] =(D20,D21,D92,D91,0,0),(-1,0,0,0),(2,0,0);
P[11] =(D80,D81,D92,D91,0,0),(-1,0,0,0),(2,0,0);
Velset 100;

L[0]:                                          ## 初始化
Set Out[2],OFF;
Set Out[3],OFF;
Set Out[4],OFF;
Set Out[5],OFF;
Set Out[6],OFF;
Set Out[9],OFF;
Set Out[6],ON;
Delay T[0.5];
Set Out[6],OFF;
Jump P[1],V[30],Z[5],LH[50],MH[-35],RH[50];
L[1]:
Movj P[1],V[30],Z[5];                          ## 运动到取料上方等待位
Wait IN[11]==ON,T[0],Goto L[1];                ## 等待有盒信号
```

```
Delay T[0.1];
Set Out[2],ON;                                  ## 打开夹盒气缸
Set Out[5],ON;                                  ## 打开顶盒气缸
Movj P[2],V[30],Z[0];                           ## 运动到取料位
Delay T[0];
Set Out[3],ON;                                  ## 打开吸真空
Wait IN[12]==ON,T[0.7],Goto L[2];               ## 等待真空检测信号
Delay T[0.05];
L[2]:                                           ## 在取料位吸气时间过长的异常处理
If IN[12]==OFF
Delay T[0.05];
If IN[12]==ON
Goto L[20];
EndIf;
Set Out[3],OFF;
Set Out[6],ON;
Delay T[0.2];
Set Out[6],OFF;
Movj P[1],V[30],Z[5];
Delay T[0];
Set Out[9],ON;
Wait IN[7]==ON,T[0];
Set Out[9],OFF;
Set Out[2],OFF;
Set Out[5],OFF;
Pause;
Delay T[0.1];
Goto L[1];
EndIf;
L[20]:
Set Out[2],OFF;
Delay T[0.15];
Movj P[8],V[30],Z[5];
Movj P[11],V[30],Z[5];
If IN[12]==OFF
Delay T[0.2];
Movj P[1],V[30],Z[5];
WaitInPos;
Set Out[9],ON;
Wait IN[7]==ON,T[0];
Set Out[9],OFF;
Set Out[3],OFF;
Set Out[6],ON;
Set Out[2],OFF;
Set Out[5],OFF;
Delay T[0.2];
Set Out[6],OFF;
Pause;
Delay T[0.1];
Goto L[1];
```

```
EndIf;

Call "shujujiexi.pro";                                  ## 调用视觉处理子程序
D40 =D0+D11;                                            ## 计算贴合位 XYθ 坐标
D41 =D1+D10;
D42 =D3+D12;
P[7] =(D40,D41,D2,D42,0,0),(-1,0,0,0),(2,0,0);
P[9] =(D40,D41,D92,D42,0,0),(-1,0,0,0),(2,0,0);
Movj P[9],V[30],Z[5],Out(5,OFF,T[0.2]);
Movj P[7],V[30],Z[0];                                   ## 运动到贴合位
Delay T[0];
Set Out[3],OFF;
Set Out[6],ON;
Delay T[0.1];                                           ## 发送定位完成信号给 PLC
Set Out[6],OFF;
P[6] =(D40,D41,D49,D42,0,0),(-1,0,0,0),(2,0,0);
Movj P[6],V[30],Z[5];
Set Out[4],ON;
If IN[11]==ON
Movj P[1],V[30],Z[5],Out(2,ON,T[0.1]);
Set Out[4],OFF;
Movj P[2],V[30],Z[0],Out(3,ON,T[0.01]);
Set Out[5],ON;
Goto L[2];
EndIf;
Goto L[1];
END;
```

子程序代码（视觉数据解析）：

```
START;
String str2;
String str_0="+";
String str13;
String str14;
String str15;
String str16;
String str17;
String str18;
String str19;
String str20;
String str21;
L[5]:

Get Port[4444],T[0],Goto L[5];                          ## 获取视觉发送的字符串数据，用 str2 表示，存储前 100 位数据，当客户端端口号不固定时，Get Port[] 内参数要设置成 4444

str2 = GetPortbuf(0,100);

                                                        ## 以下为视觉数据 X 坐标解析，存储在 D10 中
str19 = Mid(str2,0,1);
str13 = Mid(str2,1,3);
str14 = Mid(str2,4,3);
D13 = StrToD(str13);
```

```
D14 = StrToD(str14);
R10 = Strcmp(str19,str_0);
If R10==0
D10 =D13+0.001*D14;
Else
D10 =-1*(D13+0.001*D14);
EndIf;
D10 =0-D10;
                                                    ## 以下为视觉数据 Y 坐标解析，存储在 D11 中
str20 = Mid(str2,7,1);
str15 = Mid(str2,8,3);
str16 = Mid(str2,11,3);
D15 = StrToD(str15);
D16 = StrToD(str16);
R11 = Strcmp(str20,str_0);
If R11==0
D11 =D15+0.001*D16;
Else
D11 =-1*(D15+0.001*D16);
EndIf;

str21 = Mid(str2,14,1);                             ## 以下为视觉数据 θ 坐标解析，存储在 D12 中
str17 = Mid(str2,15,3);
str18 = Mid(str2,18,3);
D17 = StrToD(str17);
D18 = StrToD(str18);
R12 = Strcmp(str21,str_0);
If R12==0
D12 =D17+0.001*D18;
Else
D12 =-1*(D17+0.001*D18);
EndIf;
Ret;
END;
```

2.1.2 案例二：纸盒生产过程中的贴合应用 2

该现场案列设备与案例一均属于同一行业机型，因此在方案及动作流程上基本一致，仅在部分数据配置及上位机视觉通讯方式上有一定的差异。

1 案例描述

与案例一相同。

2 步骤说明

与案例一相同。

3 数据配置

1） 输入输出信号（请按照对应端子进行接线）

■ 输入

In(3)	急停
In(4)	暂停
In(5)	清楚报警
In(8)	有板信号
In(9)	视觉手自动（程序选择）
In(10)	启动
In(11)	停止

■ 输出

Out(2)	程序运行
Out(3)	程序停止
Out(8)	定位完成
Out(9)	吸真空

2） 点位信息

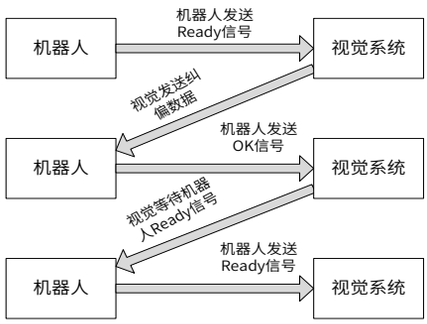
P0	取纸盒上方等待位	
P1	纸盒取料位	需要示教
P2	放料基准位	需要示教
P4	贴合前上方等待位	
P5	贴合位	
P6	纠偏位上方等待位	
P7	纠偏位	需要示教
P10	接收视觉数据等待位	需要示教

3） 数据交互流程说明

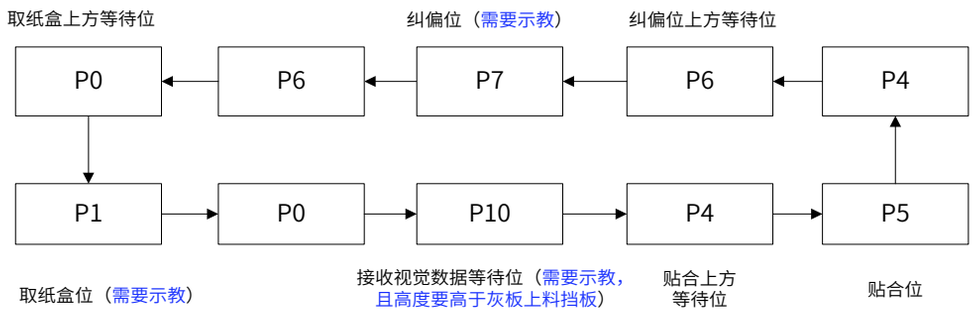
■ 机械手与视觉的通信是 TCP 网络通信。需要机械手提供对应的 IP 地址和端口号。将机械手设为服务器、视觉为客户端（客户端端口号固定为 4000），需要先打开机械手程序，再打开视觉程序。

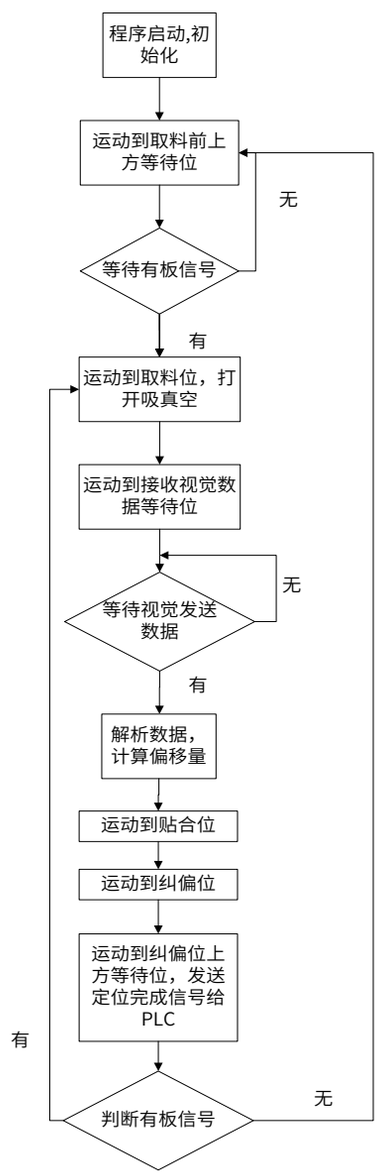
■ 打开视觉程序，若通信连接成功则收到机械手发送的字符串信息，否则未连接上。连接成功后，若在每次放完盒之后会发送一个 Ready 信号，表示机械手已准备好，视觉程序才会再次发送数据给机械手。

■ 偏移量的格式由八位字符串组成，例如 :X:12.15，Y:-1.23，Q:2.35 字符串格式 :00012150 -0001230 00002350 正号补零，负号保持不变，其他的方式不变，收到数据后反馈 OK 信号给视觉程序。


机器人
机器人发送
Ready信号
视觉系统
视觉发送纠
偏数据
机器人
机器人发送
OK信号
视觉系统
视觉等待机器
人Ready信号
机器人
机器人发送
Ready信号
视觉系统

4 程序流程图


取纸盒上方等待位
纠偏位（需要示教）
纠偏位上方等待位
P0
P6
P7
P6
P4
P1
P0
P10
P4
P5
取纸盒位（需要示教）
接收视觉数据等待位（需要示教，
且高度要高于灰板上料挡板）
贴合上方
等待位
贴合位


程序启动,初
始化
运动到取料前上
方等待位
无
等待有板信号
有
运动到取料位，打
开吸真空
运动到接收视觉数
据等待位
无
等待视觉发送
数据
有
解析数据，
计算偏移量
运动到贴合位
运动到纠偏位
运动到纠偏位上
方等待位，发送
定位完成信号给
PLC
有
无
判断有板信号

5 程序实例

```
START;
String str_0=”0”;
String str1=”Ready”;
String str2=”OK”;
String str3;
String str4=”Chao Chu Ji Xie Shou Yun Dong Fan Wei”;
String str13;
String str14;
String str15;
String str16;
String str17;
String str18;
String str19;
String str20;
String str21;
GetPValue(P[1],0,D4);
GetPValue(P[1],1,D5);                                   ## 获取取料位 XYZθ 坐标值
GetPValue(P[1],2,D7);
GetPValue(P[1],3,D8);

GetPValue(P[2],0,D0);
GetPValue(P[2],1,D1);                                   ## 获取放料基准位 XYZθ 坐标值
GetPValue(P[2],2,D2);
GetPValue(P[2],3,D3);

GetPValue(P[10],2,D50);
                                                        ## 获取接收视觉数据等待位 Z 坐标值
GetPValue(P[7],0,D90);
GetPValue(P[7],1,D91);                                  ## 获取纠偏位 XYθ 坐标值
GetPValue(P[7],3,D93);
P[0] =(D4,D5,D50,D8,0,0),(1,0,0,0),(2,0,0);
P[3] =(D4,D5,D50,D8,0,0),(1,0,0,0),(2,0,0);
P[6] =(D90,D91,D50,D93,0,0),(1,0,0,0),(2,0,0);

L[0]:
Set Out[8],OFF;
Set Out[9],OFF;
Set Out[10],OFF;
Set Out[11],OFF;
Delay T[0.5];
Jump P[0],V[30],Z[0],LH[70],MH[-20],RH[70];
Velset 100;                                             ## 初始化

L[1]:
Movj P[0],V[30],Z[5],Out(8,OFF,T[0.01]);
                                                        ## 运动到取料前上方等待位
SetPortBuf(str1) ;
Send Port[4000];                                        ## 发送 ready 信号给视觉

Wait IN[8]==ON,T[0],Goto L[1];
                                                        ## 等待有板信号
Movj P[1],V[30],Z[1];
Delay T[0];                                             ## 运动到取料位

Set Out[9],ON;
L[2]:                                                   ## 打开吸真空
Delay T[0.1];

Movj P[0],V[30],Z[5];
Movj P[10],V[30],Z[5];
                                                        ## 运动到接收视觉数据等待位
L[3]:
Get Port[4000],T[0],Goto L[3];
str3 = GetPortbuf(0,100);                               ## 获取视觉数据，此时客户端端口号固定为 4000，Get Port[] 内参数需要设置为 4000

SetPortBuf(str2) ;
Send Port[4000];

str19 = Mid(str3,0,1);                                  ## 发送 OK 信号给视觉
str13 = Mid(str3,2,3);
str14 = Mid(str3,5,3);                                  ## 解析视觉发送的 X 坐标
D13 = StrToD(str13);
D14 = StrToD(str14);
R10 = Strcmp(str19,str_0);
```

```
If R10==0
D10 =D13+0.001*D14;                                    ## 判断正负
Else
D10 =-1*(D13+0.001*D14);
EndIf;

str20 = Mid(str3,9,1);
str15 = Mid(str3,11,3);                                ## 解析视觉发送的 Y 坐标
str16 = Mid(str3,14,3);
D15 = StrToD(str15);
D16 = StrToD(str16);
R11 = Strcmp(str20,str_0);
If R11==0
D11 =D15+0.001*D16;                                    ## 判断正负
Else
D11 =-1*(D15+0.001*D16);
EndIf;

str21 = Mid(str3,18,1);
str17 = Mid(str3,20,3);                                ## 解析视觉发送的 θ 坐标
str18 = Mid(str3,23,3);
D17 = StrToD(str17);
D18 = StrToD(str18);
R12 = Strcmp(str21,str_0);

If R12==0
D12 =D17+0.001*D18;                                    判断正负
Else
D12 =-1*(D17+0.001*D18);
EndIf;                                                 ## 计算机器人贴合位置

D40 =D0-D11;
D41 =D1+D10;                                           ## 判断贴合位置是否超出运动范围
D42 =D3+D12;
D45 =Sqrt(D40*D40+D41*D41);
If D45>=600
Print str4;
Goto L[3];
EndIf;
P[4] =(D40,D41,D50,D42,0,0),(1,0,0,0),(2,0,0);         ## 运动到贴合位
P[5] =(D40,D41,D2,D42,0,0),(1,0,0,0),(2,0,0);

Movj P[4],V[30],Z[5];
Movj P[5],V[30],Z[0];
Delay T[0];
Delay T[0.3];                                          ## 运动到纠偏位

Movj P[4],V[30],Z[5];
Movj P[6],V[30],Z[5];
Movj P[7],V[30],Z[5];
Delay T[0];
Set Out[9],OFF;                                        ## 发送定位完成信号给 PLC
Delay T[0.1];
Movj P[6],V[30],Z[5];

Set Out[8],ON;
If IN[8]==ON
Movj P[0],V[30],Z[5];
Set Out[8],OFF;
SetPortBuf(str1) ;
Send Port[4000];
Movj P[1],V[30],Z[1];
Delay T[0];
Set Out[9],ON;
Goto L[2];
EndIf;
Goto L[1];
END;
```

2.1.3 案例三：纸盒生产过程中的贴合应用 3

该现场案列设备与案例一均属于同一行业机型，因此在方案及动作流程上基本一致，仅在部分数据配置以及上位机视觉通讯方式上有一定的差异。

1 案例描述

与案例一相同。

2 步骤说明

与案例一相同。

3 数据配置

1） 输入输出信号（按照对应端子进行接线）

■ 输入

In(3)	急停
In(4)	暂停
In(5)	清楚报警
In(8)	有板信号
In(9)	视觉手自动（程序选择）
In(10)	启动
In(11)	停止

■ 输出

Out(2)	程序运行
Out(3)	程序停止
Out(8)	定位完成
Out(9)	吸真空

2） 点位信息

P0	取纸盒上方等待位	
P1	纸盒取料位	需要示教
P2	放料基准位	需要示教
P4	贴合前上方等待位	
P5	贴合位	
P10	接收视觉数据等待位	需要示教

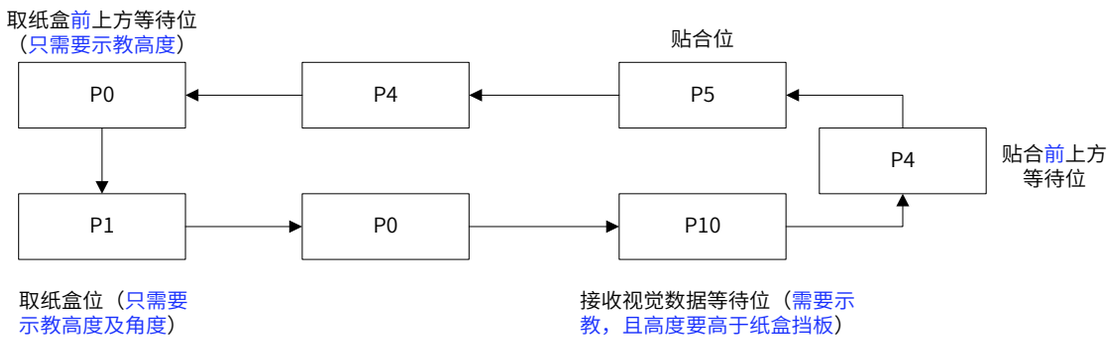
3） 数据交互流程说明

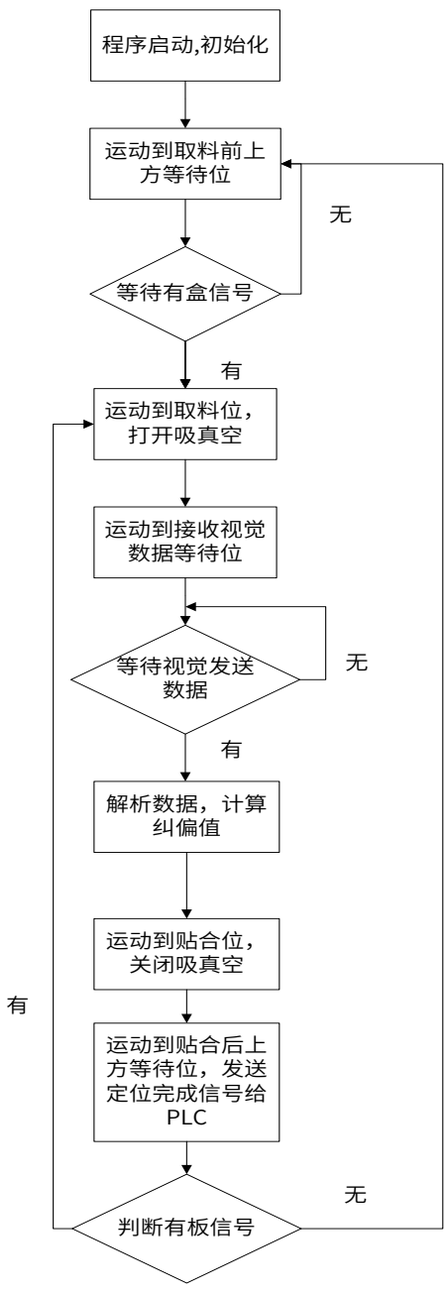
■ 机械手与视觉的通信是 TCP 网络通信。需要机械手提供对应的 IP 地址和端口号。机械手为客户端，视觉为服务器；服务器端口号为 2000，客户端端口号为 3000，需要先打开机械手程序，再打开视觉程序

■ 打开视觉程序，若通信连接成功会收到机械手发送的字符串信息。否则未连接上。连接成功后，若在每次放完盒之后会发送一个 Ready 信号，表示机械手准备好，视觉程序才会再次发送数据给机械手。

■ 偏移量的格式由八位字符串组成，例如 :X:12.15，Y:-1.23，Q:2.35 字符串格式 :12,150 -1,230 2,350 收到数据后反馈 OK 信号给视觉程序。

4 程序流程图


取纸盒前上方等待位
（只需要示教高度）
P0
P4
贴合位
P5
P4
贴合前上方
等待位
P1
P0
P10
取纸盒位（只需要
示教高度及角度）
接收视觉数据等待位（需要示
教，且高度要高于纸盒挡板）


程序启动,初始化
运动到取料前上
方等待位
无
等待有盒信号
有
运动到取料位，
打开吸真空
运动到接收视觉
数据等待位
等待视觉发送
数据
无
有
解析数据，计算
纠偏值
运动到贴合位，
关闭吸真空
有
运动到贴合后上
方等待位，发送
定位完成信号给
PLC
无
判断有板信号

5 程序实例

```
START;
String str1=”Ready”;
String str2=”OK”;
String str3;
String str4=”Chao Chu Ji Xie Shou Yun Dong Fan Wei”;

GetPValue(P[1],0,D4);
GetPValue(P[1],1,D5);                          ## 获取取料位 XYθ 坐标值
GetPValue(P[1],3,D8);

GetPValue(P[2],0,D0);
GetPValue(P[2],1,D1);                          ## 获取放料基准位 XYZθ 坐标值
GetPValue(P[2],2,D2);
GetPValue(P[2],3,D3);

GetPValue(P[10],2,D50);
P[0] =(D4,D5,D50,D8,0,0),(1,0,0,0),(2,0,0);    ## 获取接收视觉数据等待位 Z 坐标值

While  B0==0                                   ## 初始化
Open Socket(“192.168.24.22”,2000,3000,B0) ;    ## 与视觉建立通讯连接，服务器端口号为 2000，客户端端口号为 3000
EndWhile;
Set Out[8],OFF;
Set Out[9],OFF;
Set Out[10],OFF;
Set Out[11],OFF;
Delay T[0.5];
Jump P[0],V[30],Z[0],LH[70],MH[-20],RH[70];
Velset 100;
L[1]:
Movj P[0],V[30],Z[5],Out(8,OFF,T[0.01]);       ## 运动到取料前上方等待位
SetPortBuf(str1) ;                             ## 发送 ready 信号给视觉，客户端端口号为 3000
Send Port[3000];
Wait IN[8]==ON,T[0],Goto L[1];                 ## 等待有板信号
Movj P[1],V[30],Z[1];                          ## 运动到取料位
Delay T[0];
Set Out[9],ON;                                 ## 打开吸真空
L[2]:
Delay T[0.1];
Movj P[0],V[30],Z[5];
Movj P[10],V[30],Z[5];                         ## 运动到接收视觉数据等待位
L[3]:
B1 = 0;                                        ## 接收视觉数据，将视觉发送的数据存放在变量 str3 中，然后将 str3 中的数
Get Port[3000],T[0],Goto L[3];                 据以“，”为分割符进行分割，且将分割的数据依次存放到 D10/D11/D12 中
str3 = GetPortbuf(0,100);
B1 = StrGetData(Str3,”,”,D10);

If B1<>0                                       ## 收到数据后发送 OK 信号给视觉
SetPortBuf(str2);
Send Port[3000];
Else
Goto L[3];
EndIf;
                                               ## 机器人贴合位置坐标计算
D40 =D0-D11;
D41 =D1+D10;
D42 =D3+D12;
                                               ## 超限判断
D45 =Sqrt(D40*D40+D41*D41);
If D45>=600
Print str4;
Goto L[3];
EndIf;
```

```
P[4] =(D40,D41,D50,D42,0,0),(1,0,0,0),(2,0,0);
P[5] =(D40,D41,D2,D42,0,0),(1,0,0,0),(2,0,0);          ## 运动到贴合位
Movj P[4],V[30],Z[5];
Movj P[5],V[30],Z[0];
Delay T[0];
Delay T[0.3];
Set Out[9],OFF;
Delay T[0.1];                                          ## 发送定位完成信号给 PLC
Movj P[4],V[30],Z[5];
Set Out[8],ON;
If IN[8]==ON
Movj P[0],V[30],Z[5];
Set Out[8],OFF;
SetPortBuf(str1) ;
Send Port[3000];
Movj P[1],V[30],Z[1];
Delay T[0];
Set Out[9],ON;
Goto L[2];
EndIf;
Goto L[1];
END;
```

2.2 机器人与三菱 Q 系列 PLC 通讯

在一些现场，我司机器人需要与三菱的 Q 系列 PLC 采用 MELSEC 通讯协议（即 MC 协议）进行通讯。MC 通讯协议是通过 Q 系列 C24 或者 E71 来对 PLC 的软元件数据和程序的进行读出 / 写入操作的通讯协议。

2.2.1 案例描述

该自动化产线是用于手机 CG 与 Frame 组装的自动化生产线，我司机器人在整个产线中，负责手机支架 Frame 的撕膜工序。载有 Frame 的治具通过流水线到达撕膜工位后，PLC 发送到位信号给机器人，机器人执行撕膜动作，撕膜完成后发送信号给 PLC，使得治具流到后续工位。

图 2-4 为贴有薄膜的手机支架，机器人末端装有夹爪，首先运动到低于薄膜高度的位置，打开夹嘴，再提升机器人接触薄膜位置，打开下压气缸压紧薄膜，走后续点位完成撕膜动作。图 2-5 为撕膜工位示意图。

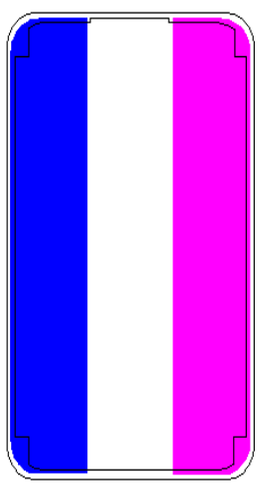
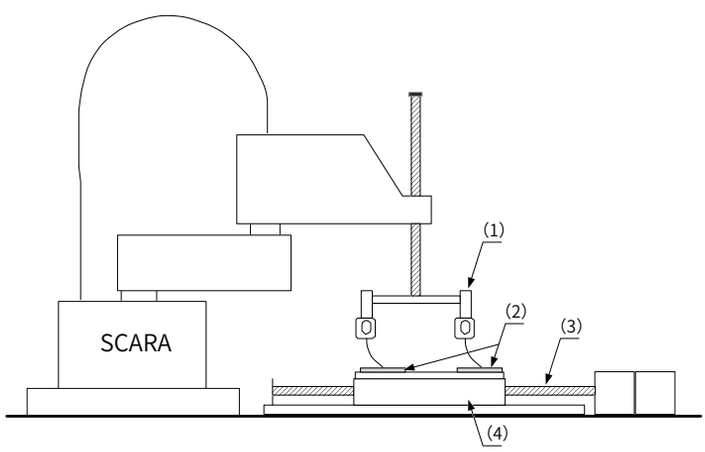


图 2-7 手机支架 Frame 示意图



（1）夹具、（2）带膜屏幕、（3）丝杆滑台、（4）载具台

图 2-8 机器人撕膜工位示意图

2.2.2 步骤说明

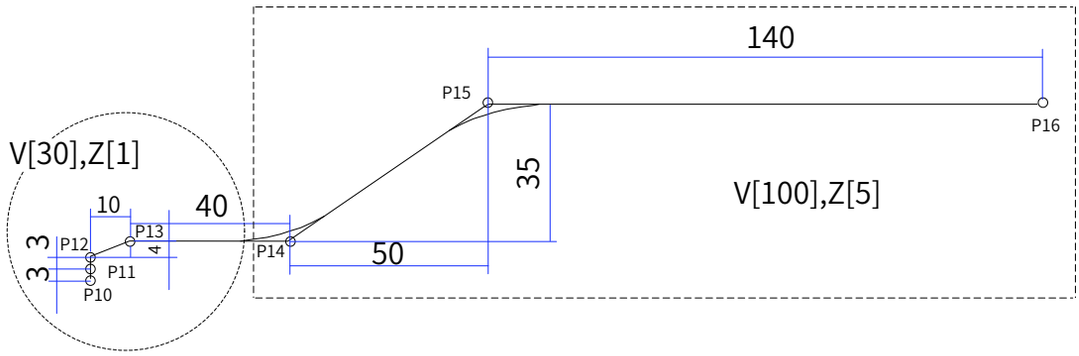


图 2-9 撕膜动作涂示意图

撕膜动作流程如下：

1） P10 为撕膜起始点，到达 P10 后，夹紧气缸打开，等待夹紧气缸 1，夹紧气缸 2 到位信号；

2） 运动到 P11 后，下压气缸 1，下压气缸 2 打开，夹紧薄膜；

3） P12~P16 完成撕膜动作；

4） P17 点位丢膜，打开夹紧气缸和下压气缸，并发送撕膜完成信号。

2.2.3 数据配置

1 输入输出信号

■ 输入

In(3)	机器人系统启动　　1：启动
In(4)	机器人系统停止　　1：暂停
In(5)	机器人系统复位　　1：停止
In(6)	机器人系统暂停　　1：复位
In(8)	夹紧气缸 1 到位检测信号
In(10)	夹紧气缸 2 到位检测信号
In(12)	顶升气缸 1 到位
In(14)	顶升气缸 2 到位

■ 输出

Out(5)	机器人系统报警　　1：报警
Out(6)	夹紧气缸输出　　0：关闭 1：开启
Out(9)	下压气缸 1 输出
Out(10)	下压气缸 2 输出

2 点位信息

以下坐标点预先输入了坐标点数据。

P0	等待点（原点）
P10	撕膜动作起点
P11	撕膜动作过程点
P12	撕膜动作过程点
P13	撕膜动作过程点
P14	撕膜动作过程点
P15	撕膜动作过程点
P16	撕膜动作过终点
P17	丢膜点

2.2.4 通讯协议说明

三菱 Q 系列 PLC 通信协议有很多种，此案例选择了 MC 协议。MC 协议发送数据有 ASCII 方式和二进制方式。MC 协议的协议方式有 A-1E 模式、Qna-3E 模式等。 A-1E 为较早的通信版本，对地址的操作范围有限（数据寄存器区的 D0~D6143、D9000~D9255），Qna-3E 可访问 D0~D12287 数据。

在此案例中，我司机器人与三菱 Q 系列 PLC，通过 MC 协议的 Qna-3E 模式，采用二进制，对寄存器的值进行读写，实现撕膜到位、撕膜完成信号的交互。MC 协议通讯的使用，即简化了接线工作与减少 IO 点的使用，也具有较好的实时性和通讯稳定性。

MC 协议进行的数据通讯采用半双工通讯方式进行，对于刚发送的命令文件需得到响应文件后，再发送下一个命令文件。

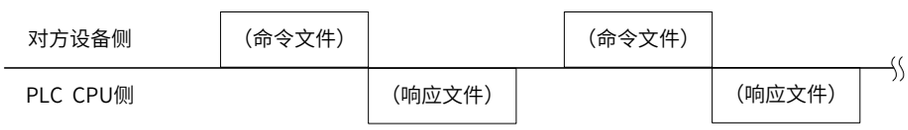


图 2-10 半双工通讯方式逻辑图

以下举例说明数据帧含义：

■ 读取

数据格式：5000 00 FF FF 03 00 0C 00 10 00 01 04 00 00 C8 00 00 A8 01 00

返回：D0 00 00 FF FF 03 00 04 00 00 00 00 00（13 个）

含义：

0C 是代表其后边有多少二进制数组，16 进制的 0C 代表 10 进制的 13；

01 04 是读；

C8 00 00 是寄存器地址 200（16 进制转换为 10 进制）；

01 00 是数据长度；

00 00 寄存器数值（16 进制得转换成 10 进制），高低位反向，如 01 00 代表 1， 00 01 代表 256。

■ 写入

数据格式：5000 00 FF FF 03 00 0E 00 10 00 01 14 00 00 C8 00 00 A8 01 00 01 02

返回：D0 00 00 FF FF 03 00 01 00 00 00（11 个）

0E 是代表其后边有多少二进制数组，16 进制的 0E 代表 10 进制的 14；

01 14 是写；

C8 00 00 是寄存器地址 200（16 进制转换为 10 进制）；

01 00 是数据长度；

01 02 是写入至寄存器的值（16 进制得转换成 10 进制）， 高低位反向，如 01 02 代表 513， 01 00 代表 1。

2.2.5 程序流程

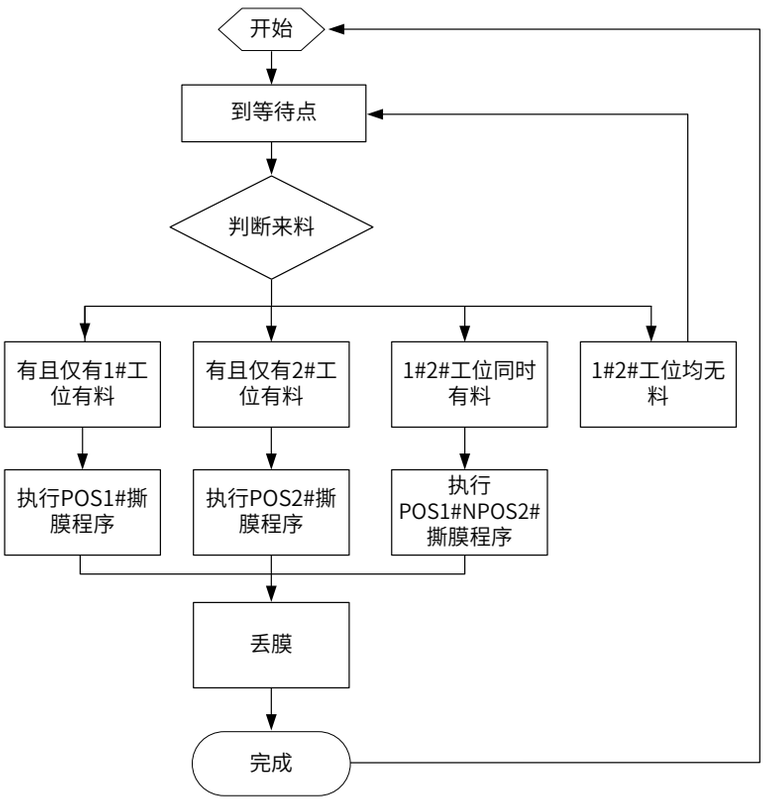


图 2-11 程序流程图

2.2.6 程序实例

```
P[0] =  297.945258,  -61.187132,  -59.885395,  101.951279; 1, 0,-1,-1; 2, 0, 0;
START;
B0 =0;                                                                    ## 初始化
B100 =0;
Set Out[9],OFF;
Set Out[10],OFF;
Delay T[0.1];
Set Out[6],OFF;
Set Out[12],OFF;
Set Out[13],OFF;
String read9512=”500000FFFF03000C0001000104000028250 0A80100”; ## 读 D9512 命令
String rev9512;
String read9572=”500000FFFF03000C0001000104000064250 0A80100”; ## 读 D9572 命令
String rev9572;
Jump P[0],V[30],Z[0],LH[20],MH[-40],RH[0];                                ## 到等待点 P0
L[0]:
While  B0<>1
Open Socket(“192.168.3.240”,4999,4999,B0) ;
EndWhile;
L[10]:
SetPortBuf(read9512) ;
Send Port[4999];
L[11]:
Get Port[4999],T[0],Goto L[11];
rev9512 = GetPortbuf(23,23);                                              ## 接收读 D9512 的返回值
D0 = StrToD(rev9512);
```

```
Delay T[0.1];
L[20]:
SetPortBuf(read9572) ;
Send Port[4999];
L[21]:
Get Port[4999],T[0],Goto L[21];
rev9572 = GetPortbuf(23,23);                                    ## 接收读 D9572 的返回值
D1= StrToD(rev9572);
If D0<>1 And D1<>1
B100 =0;
Goto L[10];
EndIf;
If D0==1 And D1<>1
B100 =1;
EndIf;
If D0<>1 And D1==1
B100 =02;
EndIf;
If D0==1 And D1==1
B100 =03;
EndIf;
Switch B100
 Case 1 :
Call “zx/POS1.pro”;
Break;
 Case 2 :
Call “zx/POS2.pro”;
Break;
 Case 3 :
Call “zx/POS1N2.pro”;
Break;
EndSwitch;
Goto L[0];
END;
```

以撕膜工位 1 和工位 2 同时有料的子程序为例，对程序简要注明：

```
P[0] =  297.945292,  -61.187052,  -59.885428,  101.951046; 1, 0,-1,-1; 2, 0, 0;
P[10] =  297.944848,  -61.186838, -131.791222,  101.948116; 1, 0,-1,-1; 2, 0, 0;
P[11] =  297.945000,  -61.187000, -130.791000,  101.948000; 1, 0,-1,-1; 2, 0, 0;
P[12] =  297.945000,  -61.187000, -125.791000,  101.948000; 1, 0,-1,-1; 2, 0, 0;
P[13] =  297.945000, -101.187000, -100.791000,  101.948000; 1, 0,-1,-1; 2, 0, 0;
P[14] =  297.945000, -121.187000, -100.791000,  101.948000; 1, 0,-1,-1; 2, 0, 0;
P[15] =  297.945000, -141.187000,  -75.791000,  101.948000; 1, 0,-1,-1; 2, 0, 0;
P[16] =  297.945000, -251.187000,  -75.791000,  101.948000; 1, 0,-1,-1; 2, 0, 0;
P[17] =   9.938718, -323.963938,  -45.392057,  101.416183; 1, 0,-1,-1; 2, 0, 0;
START;
String write95141=”500000FFFF03000C000100010400002A2500A801000100”;
String write95140=”500000FFFF03000C000100010400002A2500A801000000”;
String write95741=”500000FFFF03000C0001000104000066 2500A801000100”;
String write95740=”500000FFFF03000C0001000104000066 2500A801000000”;
Wait IN[12]==ON,T[0];                                           ## 顶升气缸 1 到位
Wait IN[14]==ON,T[0];                                           ## 顶升气缸 2 到位
Movj P[10],V[30],Z[0];                                          ## 撕膜动作起点
WaitInPos;
Set Out[6],ON;                                                  ## 夹紧气缸打开
Delay T[0.1];
Wait IN[9]==ON,T[0];                                            ## 下压气缸 1 打开
Wait IN[10]==ON,T[0];                                           ## 下压气缸 2 打开
```

```
PR0 = (0,0,2,0,0,0);
Movj P[11],V[30],Z[0];
WaitInPos;
Set Out[10],ON;
Delay T[0.1];
PR1 = (0,0,5,0,0,0);
Movj P[12],V[30],Z[0];
WaitInPos;
Movc P[13],V[30],Z[0];
Movc P[14],V[30],Z[0];
Movl P[15],V[30],Z[0];
Movl P[16],V[30],Z[0];
WaitInPos;
SetPortBuf(write95141) ;                    ## 撕膜 1 工位动作完成信号
Send Port[4999];
SetPortBuf(write95741) ;                    ## 撕膜 2 工位动作完成信号
Send Port[4999];
JumpL P[17],V[30],Z[0],LH[0],MH[-40],RH[10];    ## 丢膜点
WaitInPos;
SetPortBuf(write95140) ;                    ## 撕膜 1 工位动作完成信号复位
Send Port[4999];
SetPortBuf(write95740) ;                    ## 撕膜 2 工位动作完成信号复位
Send Port[4999];
PR1 = (0,0,5,0,0,0);
Movj Offset(PE,PR1),V[30],Z[0];
WaitInPos;
Set Out[9],OFF;                             ## 下压气缸 1 关闭
Set Out[10],OFF;                            ## 下压气缸 2 关闭
Set Out[6],OFF;                             ## 夹紧气缸关闭
Delay T[0.1];
Jump P[0],V[30],Z[0],LH[20],MH[-40],RH[0];      ## 运动到等待点
Ret;
END;
```

2.3 机器人与上位机进行 ModBus 通讯应用

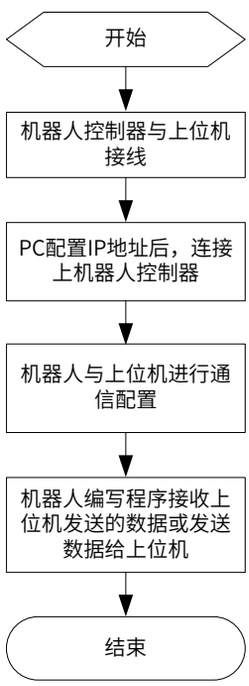
ModBus 协议是控制器之间通讯的协议，遵循主从原则，有标准、开放、免费等特点，支持多种电气接口（RS-232、RS-485、以太网等）。

有以下三种传输模式：

- ASCII 模式：一个信息中的每 8 个比特作为 2 个 ASCII 字符传输，可以自定义数据格式；
- RTU 模式：数据帧传送之间没有间隔，相同波特率下传输数据的密度要比 ASCII 高，传输速度更快；
- TCP 模式：采用以太网口的 Modbus 协议，而 ASCII 模式和 RTU 模式一般采用串口 RS232 或 RS485/422 硬件接口。

2.3.1 案例描述

上位机通过 modbus TCP 模式与机器人进行通讯，发送指令控制机器人运动，运动完成后，机器人返回运动数据，达到上位机控制机器人运动的目的。



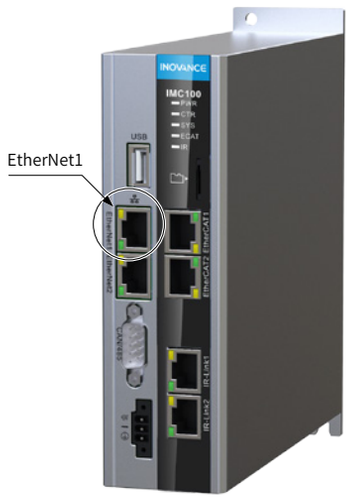
2.3.2 步骤说明

我司机器人支持的 Modbus 通讯有两种方式，分别为 ModbusTCP 通讯和 ModbusRTU 通讯。二者的区别主要在于通讯实现所需的物理层链路不同。

1 ModbusTCP 通讯方法步骤

1） 系统连接方式

将机器人控制器的 EtherNet1 口与上位机的网口通过网线连接，并将机器人控制器的 EtherNet1 口的 IP 地址与上位机配置在同一个网段（如，将机器人控制器的 EtherNet1 口的 IP 地址配置为 192.168.1.99，上位机的 IP 地址配置为 192.168.1.98）。EtherNet1 口在控制器上的位置如下图。注意 S1.16 版本之前的控制器更改 IP 地址后需要重启。



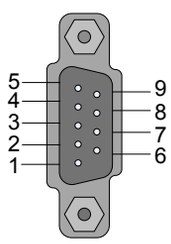
2） 参照“数据配置”部分完成数据配置。ModbusTCP 只需在二次开发软件中选中 modbusTCP 从站即可，无需配置波特率及数据位等信息。

2 ModbusRTU 通讯方法步骤

1） 系统连接方式

机器人控制器 DB9 端口定义如下图。

图 2-12　DB9 端口引脚定义

引脚	定义	说明	外观
1	CGND	通讯地	
2	CANL	CAN 负	
3	CGND	通讯地	
4	NC	NC	
5	485+	485 正	
6	NC	NC	
7	CANH	CAN 正	
8	NC	NC	
9	485-	485 负	

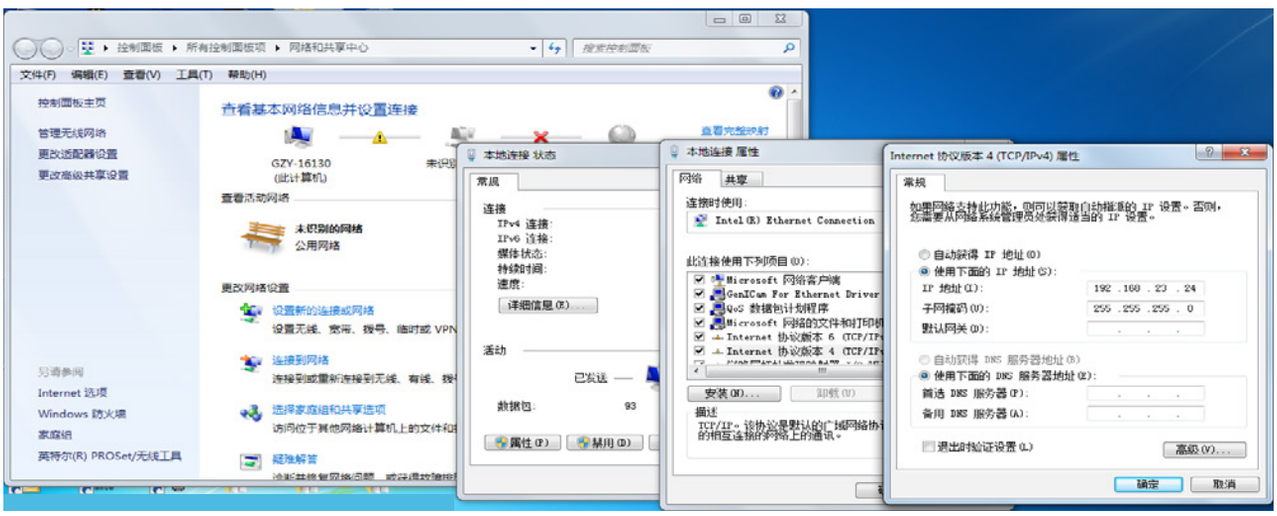
DB9 接口在控制器上的位置如下图。

因此，只要将机器人控制器上的 485+ 和 485- 接口通过焊线连到上位机的 485+ 和 485- 接口，串口配置与机器人配置相同即可进行通讯。

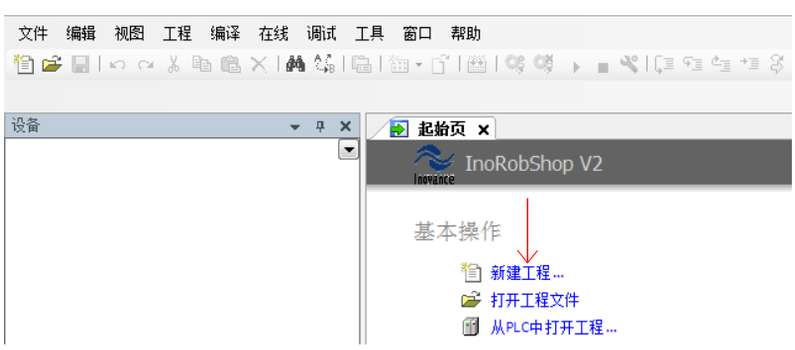
2） 完成数据配置

以配置较复杂的 modbusRTU 为例。先用 PC 上的 InoRobShop 进行配置，完成后再下载到机器人控制器中（备注：案例以 IRCB10 标准电柜为例）。

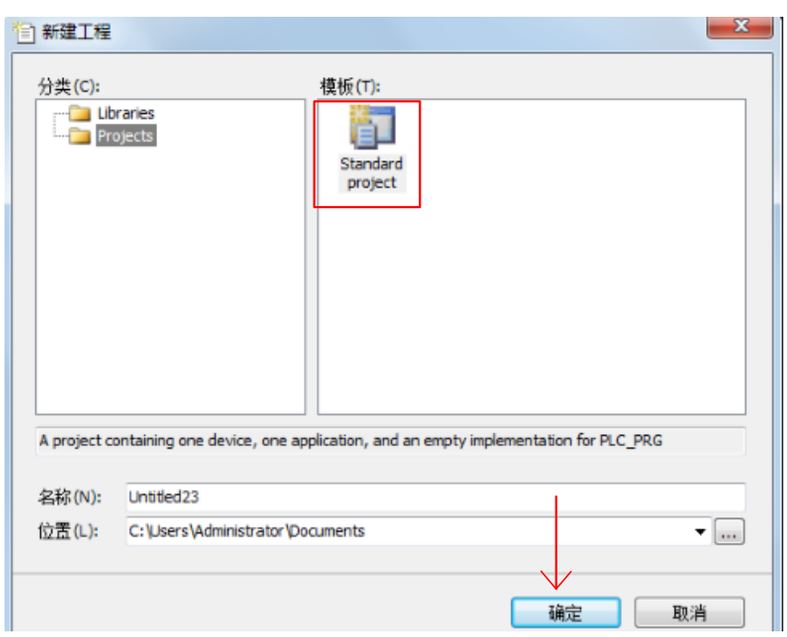
■ 将网线插到控制器 2 口，不需要修改控制器的 IP 地址。另一端连上电脑，将电脑的本地连接的 IP 地址改为与机器人控制器同一个网段，即 192.168.23.xx，但不能与控制器 IP 地址完全相同。



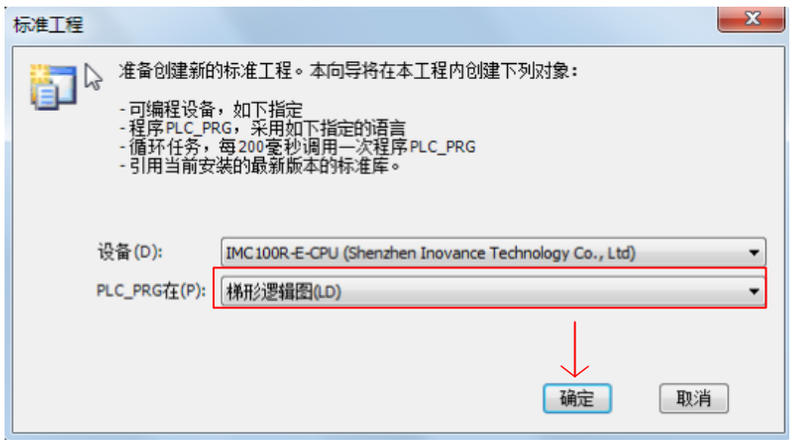
■ 双击打开 InoRobShop，点击新建工程。



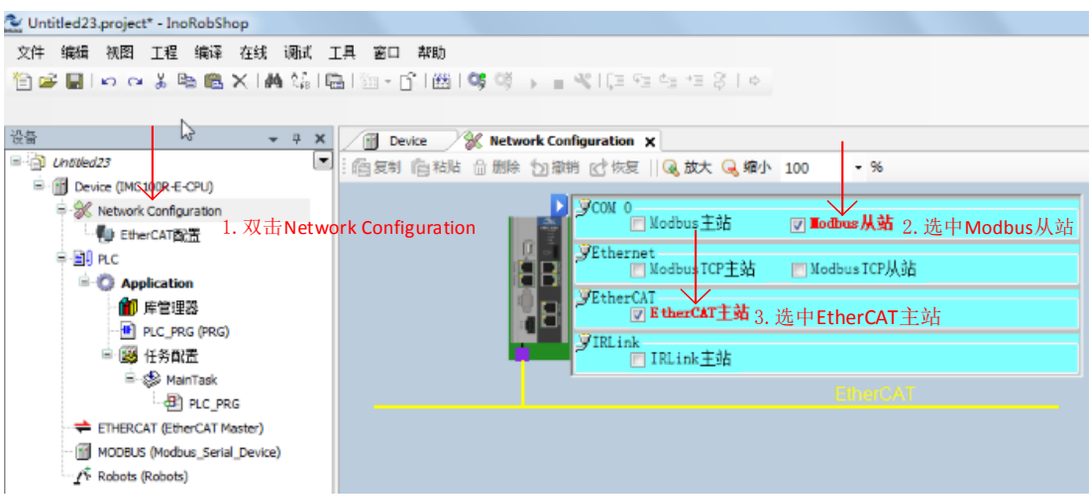
■ 选中“Standard project”，点击确定



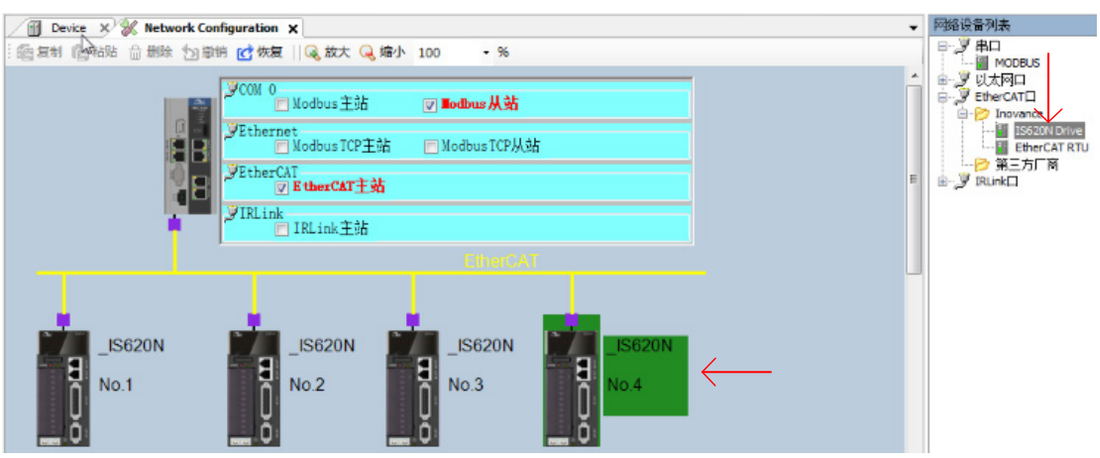
■ PLC 语言选择任意一种均可。比如梯形逻辑图，然后点击确定。



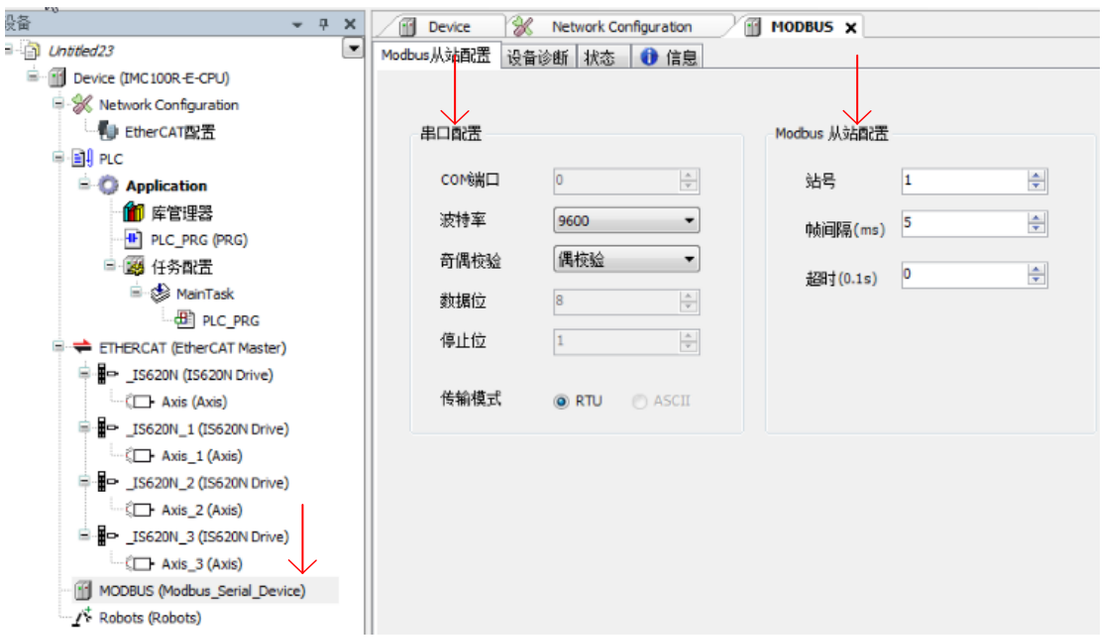
■ 点击左侧的 Network Configuration，再点击中间的控制器图案，会弹出通讯可选项，选中 Modbus 从站和 EtherCAT 主站。



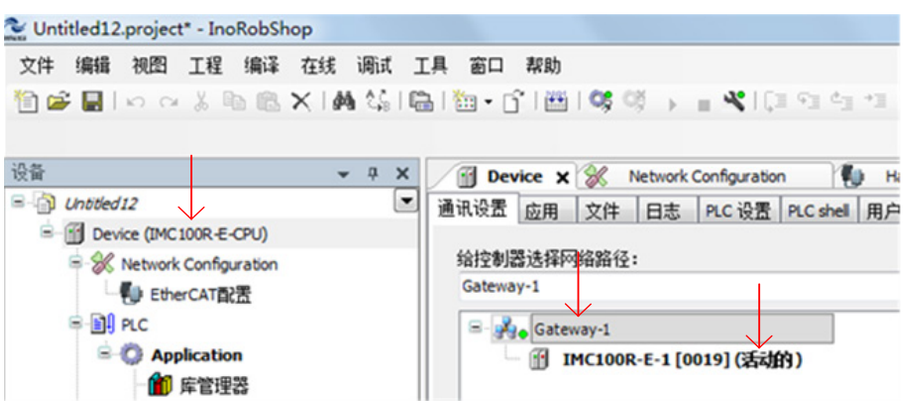
■ 点开右侧的 EtherCAT 口，双击四次 IS620N，即选上四个 IS620N。



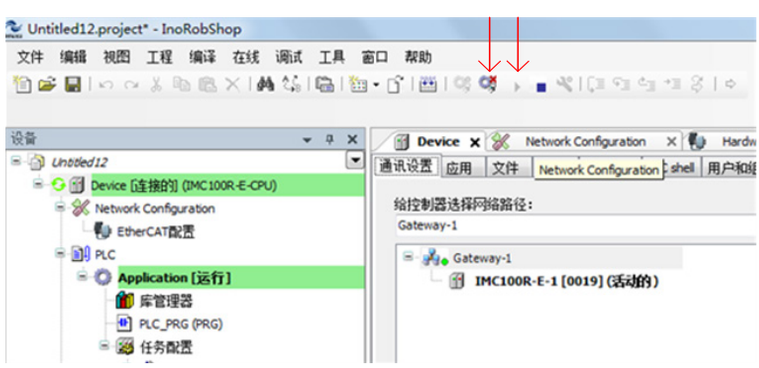
■ 点击左侧的 Modbus 选项，按照实际情况配置通讯格式，包括波特率和奇偶校验。PLC 与控制器通讯时，需要确认通讯格式是否符合。



■ 点击左侧上方的 Device，再点击图中的 Gateway 会开始扫描控制器。如果有能连接上的控制器会出现在下方，如果没有则需要确定第一步的 IP 地址是否配置正确。然后双击下方的 IMC100R，出现（活动的）即可。



■ 点击 ALT+F8 进行下载，下载完成后点击启动（F5）即可。



2.3.3 机器人操作指令说明

机器人程序中可用的读写寄存器地址范围为：%MW49152-%MW65535。可用功能码为：0x03,0x10。使用到的两个指令为 GetModBusReg 与 SetModBusReg。

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
读写 (32768) 保存寄存器 , 功能码：0x03,0x10	49152	0xc000	MW49152	字	-	SetModbusReg- 可用 GetModbusReg- 可用	允许写地址 允许读地址
	49153	0xc001	MW49153	字	-		
	49154	0xc002	MW49154	字	-		
	…		…….	字	-		
	65535	0xffff	MW65535	字	-		

如果需要与机器人底层交换数据，例如获取当前坐标、坐标系、速度等参数，无需在机器人程序中使用 GetModBusReg 或 SetModBusReg 指令，直接对相应寄存器进行操作即可。例如读取机器人当前速度，十进制寄存器地址为 MW2048，功能码为 0x04。（注意：不同厂家的产品的功能码定义可能与我司有所不同）

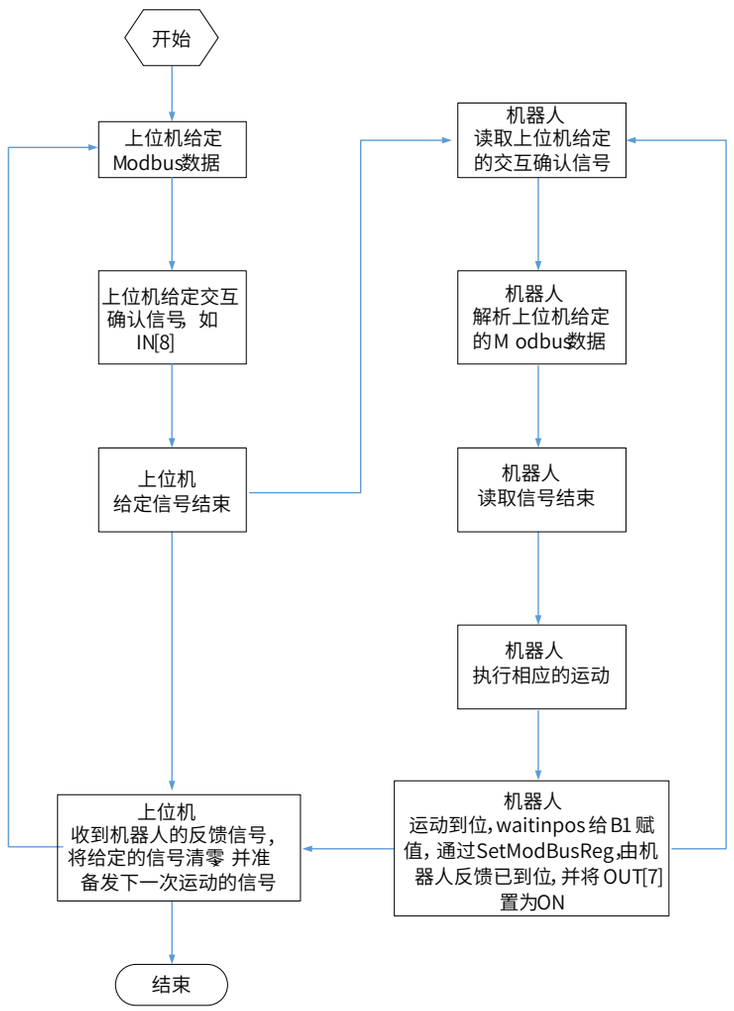
1） 使用机器人从 PLC 接收数据时，使用 GetModbusReg 指令，定义好起始地址、接受数据的变量及寄存器的个数即可。可用 Print 指令显示接收到的数据值。例如编写以下程序，运行后即可接收到相应地址的数据。

```
START;
GetModBusReg(49152,R0,1);
Print R0;
END;
```

2） 使用机器人给 PLC 寄存器发数据时，使用 SetModbusReg 指令，同样的定义好起始地址、发送数据的变量及寄存器的个数即可。例如编写以下程序，运行后即可将 R0 中的数据存入相应地址并发送。

```
START;
R0 =2;
SetModBusReg(49152,R0,1);
END;
```

2.3.4 程序流程图


开始
上位机给定Modbus数据
上位机给定交互确认信号 如IN[8]
上位机给定信号结束
上位机收到机器人的反馈信号，将给定的信号清零 并准备发下一次运动的信号
结束
机器人读取上位机给定的交互确认信号
机器人解析上位机给定的M odbus数据
机器人读取信号结束
机器人执行相应的运动
机器人运动到位，waitinpos 给 B1 赋值，通过SetModBusReg,由机器人反馈已到位，并将 OUT[7]置为ON

2.3.5 程序实例

```
START;                                          ## 开始
R0 =0;
R1 =0;
SetModBusReg (49252,R1,1);                      ## 初次上电时数据复位
Set Out[7],OFF;
L[0]:
If IN[8]==ON                                    ## 动作执行信号确认，IN[8] 为交互启动确认
GetModBusReg (49152,R0,1);                      ## 获取上位机发送的 modbus 数据
Switch R0                                       ## 判断上位机发送的数据，进行不同的运动
Case 1 :                                        ## 如果数据为 1
If R1<>1                                        ## 判断上次返回的数据不是 1，防止重复运动
Set Out[7],OFF;
Jump P[13],V[B10],Z[5],LH[50],MH[-10],RH[0];    ## 机器人进行相应的运动
Jump P[14],V[B10],Z[5],LH[0],MH[-10],RH[50];
WaitInPos;
R1 =1;
SetModBusReg (49252,R1,1);                      ## 运动完成后将 modbus 数据反馈给上位机
Set Out[7],ON;                                  ## 打开交互结束确认信号 Out[7]
EndIf;
Break;
Case 2 :                                        ## 如果数据为 2
If R1<>2                                        ## 判断上次返回的数据不是 2，防止重复运动
Set Out[7],OFF;
Jump P[15],V[B10],Z[5],LH[50],MH[-10],RH[0];    ## 机器人进行相应的运动
WaitInPos;
R1 =2;
SetModBusReg (49252,R1,1);                      ## 运动完成后将 modbus 数据反馈给上位机
Set Out[7],ON;                                  ## 打开交互结束确认信号 Out[7]
EndIf;
Break;
EndSwitch;
EndIf;
Goto L[0];                                      ## 循环执行
END;                                            ## 结束
```

2.4 232 自由协议应用

在 PLC 与机器人建立通讯后，PLC 通过 232 接口传输特定格式字符串给机器人，机器人进行解析后按照 PLC 要求完成不同动作，其中包括设置运行速度、修改点位、选择前后盖子程序等。

2.4.1 案例描述

1） 机器人控制器串口为 RS485，需接转接头才能进行 232 通信。

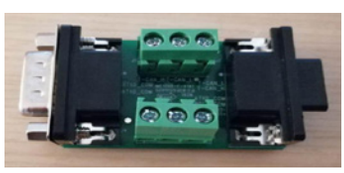
2） 对字符串解析，完成 PLC 不同功能要求。

3） 本项目要求点位数据、参数能够掉电不清零。目前我司通过指令（SetPValue、B1=7 等）写入变量的形式不能实现掉电不清零，目前有两种实现不清零方式：a. 将数据写入底层；b. 将数据保存至“点文件”，上电重启后重新读取。

2.4.2 硬件及软件版本配置

1 硬件接口

485 转 232 转接头如下，输入为 DB9 公头，输出为 DB9 母头，将装接头接入控制器 485 口即可。



2 232 通信软件配置

16 版本中串口 232 模式分为两种模式，一是调试模式，二是通信模式；调试模式用于监控控制器的信息状态，并可通过此串口对控制器进行后台操作；通信模式则用于普通的 232 串口通信。

刷机后，默认的模式为调试模式，目前可通过两种方式进行切换。

1） 控制器开机切换（不推荐使用）

■ 将控制器串口连接电脑（波特率 115200，8 位数据，1 位停止位，无奇偶校验，无流控）

■ 控制器启动后，立马常按（Ctrl+P）三秒左右，实现切换

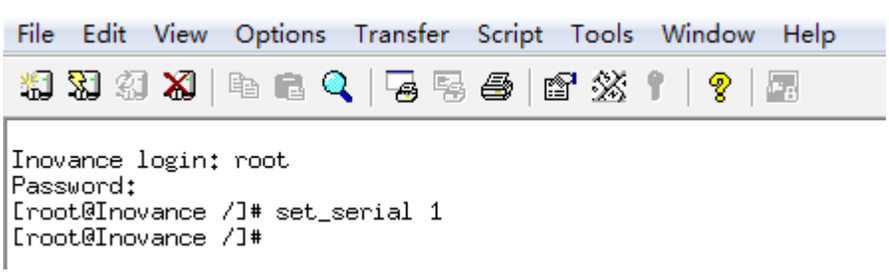
此方法无法指定模式，只能进行模式切换，且时间要求较高，开机后需立马长按（Ctrl+P），现场实现难度较大。

2） 命令行切换

■ 通过串口或是网口连接机器人后，由“SecureCRT”软件打开后台（账户 root，密码 r）；

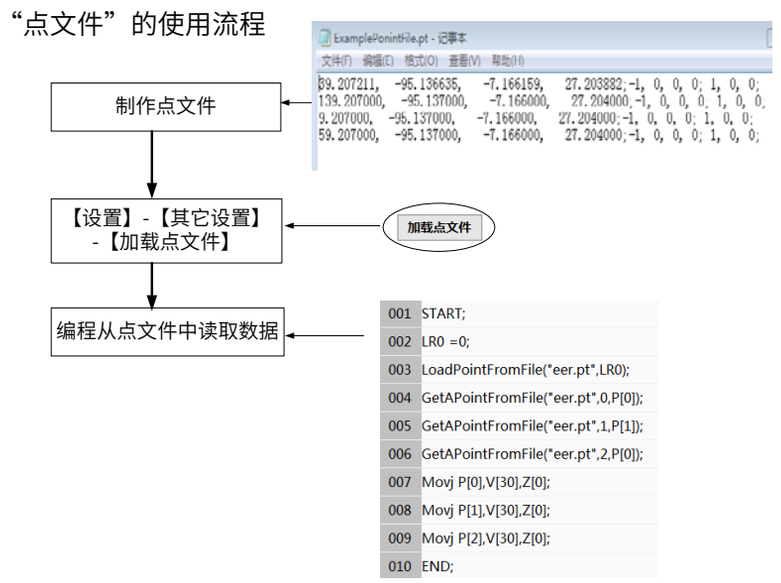
■ 选择模式：输入命令 set_serial 1 打开通信模式（允许使用 232 通信），输入命令 set_serial 0 打开调试模式；

■ 重启生效。



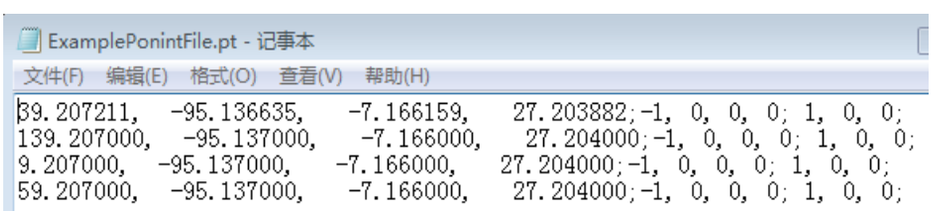
3）“掉电保持”软件配置：

因用到“变量掉电保持”，需要加载点文件（如只需要 232 通信可不配置）。



3 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。示例如下：



2.4.3 数据配置表

■ 输入输出信号

Out(4)	触发信号（预留）

■ P 变量定义

P0~P12	运动点位
P13~P14	B/R/D 变量寄存

■ D 变量定义

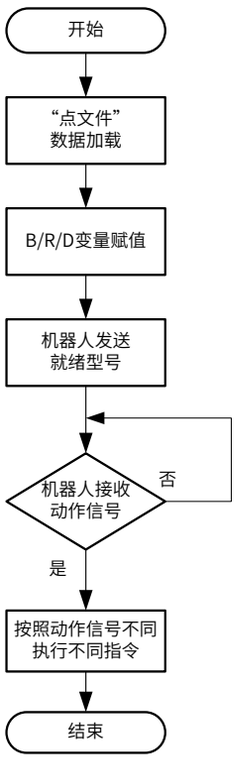
D10~D13	所有点位 Z 轴上升高度、料盘下降高度、转盘下降高度、抽检下降高度
D22	位置数据寄存

■ R 变量定义

R10~R12	正反印标志、速度标志、前后盖标志
R20	动作指令选择标志
R21	读取、修改 P 变量序号

2.4.4 程序流程图

整体框架流程图：



2.4.5 程序实例

```
START;
LoadPointFromFile ("sanxing.pt",LR1);                    ## 加载“点文件”
For LR0=0,LR0<15,Step[1]
GetAPointFromFile ("sanxing.pt",LR0,P[LR0]);             ## 循环读取“点文件”，赋值点位
EndFor;

GetPValue(P[13],0,D15);                                  ## 将点位数据解析，赋值 R 变量
R10 =D15;
GetPValue(P[13],1,D16);
R11 =D16;
GetPValue(P[13],2,D17);
R12 =D17;

 Switch R11                                              ## 读取运行速度并赋值
 Case 0 :
VelSet Rate[100];
Break;
 Case 1 :
Velset Rate[50];
Break;
EndSwitch;

GetPValue(P[14],1,D11);                                  ## 将点位数据解析，赋值 D 变量
```

```
GetPValue(P[14],2,D12);
GetPValue(P[14],3,D13);

String m0="";                                   ## 字符串变量初始化
String m1="";
String m2="";
String m3="";
String m4="";
String TX="com_ok";
String RX="";
B100 =0;
B101 =0;

L[0]:                                           ## 打开串口，波特率为 115200
Open Com(2,115200,B1) ;
If  B1 <> 1
Goto L[0];
EndIf;
SetPortBuf(TX) ;                                ## 发送就绪信号
Send Port[2];

L[2]:                                           ## 等待 PLC 发送动作信号
Get Port[2],T[0],Goto L[3];
RX =GetPortbuf(0,100);                          ## 字符串数据解析
B0 =StrGetData(RX,",",D20);
B0 =StrGetData(RX,",",R20);
 Switch R20
 Case 301 :                                     ## 不同动作指令
Set Out[4],ON;                                  ##OUT[4] 触发信号（预留）
Break;
 Case 302 :
Set Out[4],OFF;
Break;
 Case 303 :                                     ## 正印
R10 =0;
D15 =R10;
SetPValue(P[13],0,D15);                         ## 保存 R10 变量至 P 变量中，“曲线”掉电保持
WriteAPointToFile("sanxing.pt",13,P[13]);
SavePointFile("sanxing.pt");
Break;
 Case 304 :                                     ## 反印
R10 =180;
D15 =R10;
SetPValue(P[13],0,D15);
WriteAPointToFile("sanxing.pt",13,P[13]);
SavePointFile("sanxing.pt");
Break;
 Case 305 :                                     ## 设置机台为高速运动
R11 =0;
VelSet Rate[100];
D16 =R11;
SetPValue(P[13],1,D16);
WriteAPointToFile("sanxing.pt",13,P[13]);
```

```
SavePointFile("sanxing.pt");
m0 ="-3";
m1 =RToStr(R10);
m2 =RToStr(R11);
m3 =RToStr(R12);
m4 ="0";
TX =m0 + "," + m1 + "," + m2 + "," + m3 + "," + m4;
Print TX;
SetPortBuf(TX) ;
Send Port[2];                                   ## 反馈设置数据，确保正确写入
Break;
 Case 306 :                                     ## 设置机台为低速运动
R11 =1;
Velset Rate[50];
D16 =R11;
SetPValue(P[13],1,D16);
WriteAPointToFile("sanxing.pt",13,P[13]);
SavePointFile("sanxing.pt");
m0 ="-3";
m1 =RToStr(R10);
m2 =RToStr(R11);
m3 =RToStr(R12);
m4 ="0";
TX =m0 + "," + m1 + "," + m2 + "," + m3 + "," + m4;
SetPortBuf(TX) ;
Send Port[2];
Break;                                          ## 反馈设置数据，确保正确写入
 Case 307 :
Break;
 Case 308 :
Break;
 Case 400 :
GetCurPoint(2,0,D0);
GetCurPoint(2,1,D1);                            ## 读取当前坐标
GetCurPoint(2,2,D2);
GetCurPoint(2,3,D3);
m0 ="-1";
m1 =DToStr(D0,3,3);
m2 =DToStr(D1,3,3);
m3 =DToStr(D3,3,3);
m4 =DToStr(D2,3,3);
TX =m0 + "," + m1 + "," + m2 + "," + m3 + "," + m4;
SetPortBuf(TX) ;
Send Port[2];
Break;
 Case 401 :                                     ## 反馈设置数据，确保正确写入
m0 ="-2";
m1 =DToStr(D10,3,3);
m2 =DToStr(D11,3,3);                            ## 读取 Z 轴设定坐标
m3 =DToStr(D12,3,3);
m4 =DToStr(D13,3,3);
TX =m0 + "," + m1 + "," + m2 + "," + m3 + "," + m4;
SetPortBuf(TX) ;
```

```
Send Port[2];
Break;
 Case 500 :
GetPValue(P[R21],0,D0);                          ## 反馈设置数据，确保正确写入
GetPValue(P[R21],1,D1);
GetPValue(P[R21],2,D2);
GetPValue(P[R21],3,D3);                          ## 读取 P 点坐标，P 点序号 R21 决定
m0 =RToStr(R21);
m1 =DToStr(D0,3,3);
m2 =DToStr(D1,3,3);
m3 =DToStr(D3,3,3);
m4 =DToStr(D2,3,3);
TX =m0 + "," + m1 + "," + m2 + "," + m3 + "," + m4;
SetPortBuf(TX) ;
Send Port[2];
Break;
 Case 600 :
SetPValue(P[R21],0,D22);
WriteAPointToFile ("sanxing.pt",R21,P[R21]);     ## 反馈设置数据，确保正确写入
SavePointFile("sanxing.pt");
Break;
 Case 601 :                                      ## 写入 P 点 X 轴坐标，P 点序号 R21 决定
SetPValue(P[R21],1,D22);
WriteAPointToFile ("sanxing.pt",R21,P[R21]);
SavePointFile("sanxing.pt");
Break;
 Case 602 :                                      ## 写入 P 点 Y 轴坐标，P 点序号 R21 决定
SetPValue(P[R21],3,D22);
WriteAPointToFile ("sanxing.pt",R21,P[R21]);
SavePointFile("sanxing.pt");
Break;
 Case 603 :                                      ## 写入 P 点 A 轴坐标，P 点序号 R21 决定
D10 =D22;
For LR0=0,LR0<13,Step[1]
SetPValue(P[LR0],2,D10);
WriteAPointToFile ("sanxing.pt",LR0,P[LR0]);
SavePointFile("sanxing.pt");                     ## 写入所有点位 Z 轴上升高度
EndFor;
Break;
 Case 706 :
D11 =D22;
SetPValue(P[14],1,D11);
WriteAPointToFile("sanxing.pt",14,P[14]);
SavePointFile("sanxing.pt");
Break;
 Case 707 :
D12 =D22;
SetPValue(P[14],2,D12);
WriteAPointToFile("sanxing.pt",14,P[14]);
SavePointFile("sanxing.pt");
Break;                                           ## 写入料盘 Z 轴下降高度
 Case 708 :
D13 =D22;
```

```
SetPValue(P[14],3,D13);
WriteAPointToFile("sanxing.pt",14,P[14]);
SavePointFile("sanxing.pt");
Break;                                          ## 写入转盘 Z 轴下降高度
 Case 309 :
R12 =1;
D17 =R12;
SetPValue(P[13],2,D17);
WriteAPointToFile("sanxing.pt",13,P[13]);
SavePointFile("sanxing.pt");                    ## 后盖程序参数设置
m0 ="-3";
m1 =RToStr(R10);
m2 =RToStr(R11);
m3 =RToStr(R12);
m4 ="0";
TX =m0 + "," + m1 + "," + m2 + "," + m3 + "," + m4;
SetPortBuf(TX) ;
Send Port[2];
Break;
 Case 310 :
R12 =0;
D17 =R12;
SetPValue(P[13],2,D17);                         ## 反馈设置数据，确保正确写入
WriteAPointToFile("sanxing.pt",13,P[13]);
SavePointFile("sanxing.pt");
m0 ="-3";                                       ## 前盖程序参数设置
m1 =RToStr(R10);
m2 =RToStr(R11);
m3 =RToStr(R12);
m4 ="0";
TX =m0 + "," + m1 + "," + m2 + "," + m3 + "," + m4;
SetPortBuf(TX) ;
Send Port[2];
Break;
EndSwitch;

Switch R12
 Case 0 :
Call "com232/qiangai.pro";                      ## 反馈设置数据，确保正确写入
Break;
 Case 1 :
Call "com232/hougai.pro";
Break;
EndSwitch;                                      ## 前后盖程序调用
Goto L[2];
END;
```

第 3 章 视觉组合标定与应用

3.1 概述

1 标定简介

视觉是机器人的眼睛。在对位、检测、装配、插件、点胶等应用上，机器人和视觉是强关联的关系，通常将“机器人 + 视觉”的系统称为“手眼系统”。“手眼系统”可分为“眼在手上”（eye-in-hand）和“眼在手外”（eye-to-hand）两种方式。现在要建立机器人和视觉的关联，必需先进行视觉标定。通过标定，建立视觉坐标系和机器人基坐标系的关系，从而达到“手眼一体”的目的。

2 标定前准备

1） 带有 mark 点（视觉能识别的）的工具（如末端贴有圆形图标的长条刀片）

2） 视觉系统

3） Scara 机器人

3 scara 几种常见相机安装方式：

1） 固定仰视安装：主要用于校正机器人末端产品的偏差。

2） 固定俯视安装：用于拍摄区域固定的场合，常与机器人实现并行处理，影响节拍效率较低。

3） 随动 J2 轴俯视安装：使用于拍摄小但多区域的场合，拍摄区域灵活，但是与机器人执行任务串行，影响整体效率。

4） 随动 J4 轴俯视安装：使用于拍摄小但多区域的场合，拍摄区域相较于 J2 轴更为灵活，Z 轴方向可上下调整，且视野中心可以作为工具中心，但是与机器人执行任务串行，影响整体效率，另外增加末端工具的转动。

本文以上下对位场景作为案例，两种标定实现方式（标定①在机器人系统实现和②在视觉系统实现）进行阐述说明。

3.2 上下对位贴合 / 机器人侧标定案例

3.2.1 系统架构

机器人 + 固定仰视相机 + 随动 J2 俯视相机。

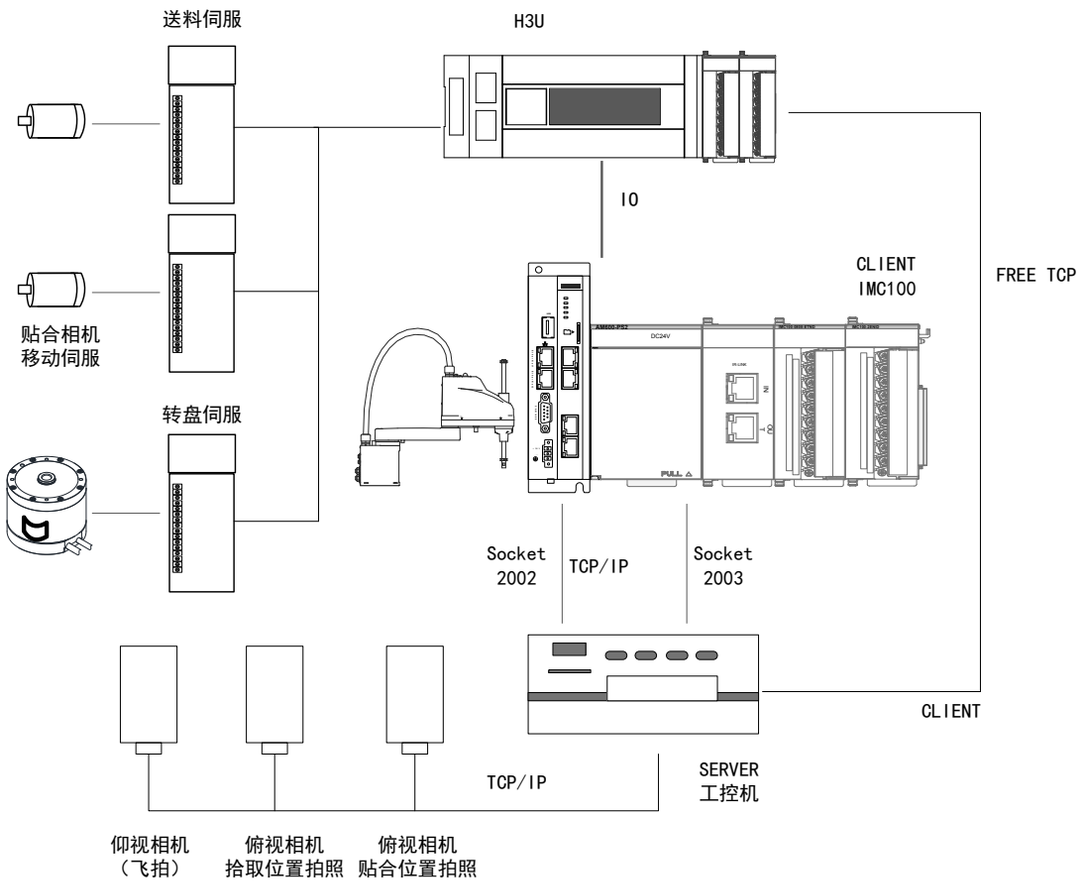


图 3-1 系统架构示意图

3.2.2 相机的标定方式

1 固定仰视相机的标定

- 步骤 1：利用吸嘴吸取一个 mark 点，运动到仰视相机视野中，通过调整 J3 轴高度，以及调节相机焦距完成聚焦，使 mark 点在相机视野中清晰成像。
- 步骤 2：通过视觉找圆工具，识别 mark 点特征，并在视野中显示 mark 点坐标。
- 步骤 3：打开示教软件，进入设置 -- 功能扩展 -- 视觉标定界面 -- 选择视觉坐标系编号 1-- 点击视觉标定。选择标定方式为手动标定，相机安装方式为固定仰视安装。进入视觉手动标定界面。
- 步骤 4：基准点确定。将 mark 点移动到大致视野中心的位置，记录当前 mark 点在相机下的像素坐标（X1，Y1），并点击“基准点一，取当前点”完成第一个基准点的标定；将机器人 J4 轴旋转一个角度（建议大于 45 度），再移动机器人使得 mark 点在相机下的坐标仍然为（X1，Y1），点击“基准点二，取当前点”。完成第二个基准点的标定。

注意：

1） 让 mark 点出现在视野中心是为了减少相机边缘畸变对识别精度带来的误差。

2） 用仰视相机取标定基准点可以最大程度减小肉眼对点带来的误差。

- 步骤 5：保持 J4 轴角度不变，移动机器人到 9 个位置进行九点标定。具体标定方法请参见《汇川机器人设计应用与维护手册 - 应用篇 2：高级功能应用》第 2 章 视觉标定。

2 随动 J2 俯视相机的标定

- 步骤 1：工作区域附近贴一个 mark 点，移动机器人使得 J2 随动相机视野中心大致对准 mark 点，调整焦距使得相机清晰成像。

- 步骤 2：利用在 J4 轴安装尖端或者大头针，对用尖端对准 mark 点，取两次基准点（步骤类似固定仰视标定步骤 4。

注意：利用尖端对准 mark 点时可用放大镜、手机摄像头进行精确校准。

- 步骤 3：移动机器人（保持 J4 角度不变）进行九点标定。具体标定方法请参见《汇川机器人设计应用与维护手册 - 应用篇 2：高级功能应用》第 2 章 视觉标定。
- 步骤 4：利用 cnvrt 指令初步验证标定结果。

原理简述：完成标定后，在机器人 J4 上安装尖端，利用相机拍照后将拍照点的像素坐标发送给机器人设置为点 P1，将像素坐标 P1 利用 cnvrt 指令转换到机器人基坐标系下的点 P2，机器人携带非偏心尖端运动到 P2 点，观察对点精度。

3.2.3 程序及注释解析（位置变量请自行添加）

```
START;
##------ 变量定义 ------
B0 =0;
B1 =0;
##------ 通讯字符串变量定义 ------
String str_send=”TA”;
String str_reciev;
##------ 打开通讯端口，机器人做服务器时可注释掉 ------
While  B0 <> 1
Open Socket(“192.168.1.1”,2000,2002,B0) ;
EndWhile;
##------ 机器人运动到拍照点 P0------
Movj P[0],V[30],Z[0];
WaitInPos;
##------ 延时 0.1s 保证机器人末端或相机无残余震动影响精度，低速时可注释掉 ------
Delay T[0.1];
##------ 将字符串 TA 发送到通讯发送缓存区 ------
L[1]:
SetPortBuf(str_send) ;
##------ 发送字符串 TA 触发相机拍照并接收拍照数据 ------
##------ 若等到 2s 后仍未接受到数据，则再次发送 TA 触发拍照 ------
Send Port[2002];
Get Port[2002],T[2],Goto L[1];
##------ 从缓存区读取数据，读取长度为 100------
str_reciev = GetPortbuf(0,100);
##------ 数据解析 ------
B1 = StrGetData(str_reciev,”，”,D0);
##------ 注意：P[1] 的臂参数需要和实际示教点的臂参数一致 ------
P[1] =(D0,D1,0,D2,0,0),(1,0,0,0),(6,0,1);
##------ 将像素坐标系下的点 P[1] 转换到基坐标系下 P[2] 点 ------
Cnvrt(P[1],P[2],World,P[0]);
##------ 打印 P[2] 值 ------
Print P[2];
##------ 设置 P[2] 高度为 -50------
SetPValue(P[2],2,-50);
##------ 运动到 P[2] 点 ------
Jump P[2],V[30],Z[0],LH[10],MH[-10],RH[10];
##------ 暂停，观察尖端位置 ------
Pause;
##------ 流程结束 ------
END;
```

3.2.4 贴合流程及程序解析

1 设备布局

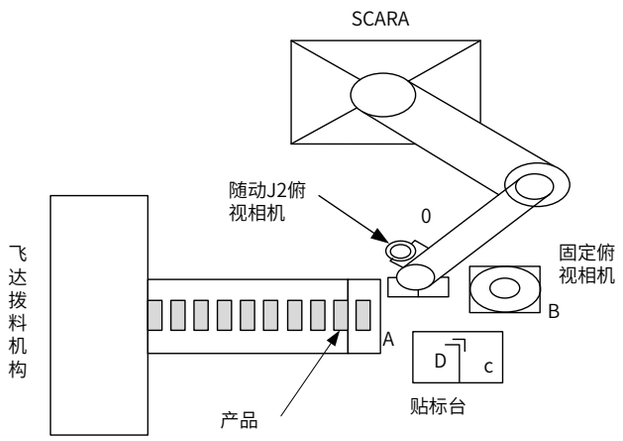


图 3-2 设备布局示意图

2 动作流程

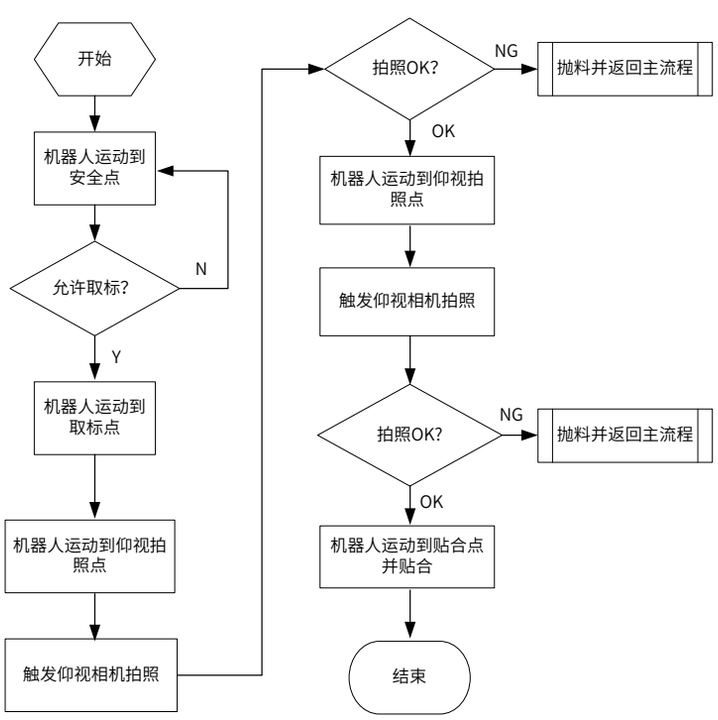
■ 动作流程

0-A-B-C-D-0

■ 技术关键点：动态工具坐标系的使用

动态工具坐标系，指的是将仰视相机对位所传输过来的点位视为机器人的一个末端工具，目的是在俯视相机对位时，将对位点转换到此工具坐标系下，完成贴合点的对应。一般用 SetToolParm 指令进行建立（具体参数可参见《汇川机器人控制系统编程手册》）。

3 流程图



4 程序及解析（位置变量请自行添加）

```
START;
##------ 变量定义 ------
B0 =0;
B1 =0;
B2 =0;
R0 =0;
R1 =0;
##------ 通讯字符串变量定义 ------
String str_send1=”TA”;
String str_send2=”TB”;
String str_NGA=”NGA”;
String str_NGB=”NGB”;
String str_recv1;
String str_recv2;
##======IO 复位 ======
##------ 吸嘴真空电磁阀信号 ------
Set Out[3],OFF;
##------ 取标完成信号 ------
Set Out[4],OFF;
##------ 仰视拍照 OK 信号 ------
Set Out[5],OFF;
##------ 仰视拍照 NG 信号 ------
Set Out[6],OFF;
##------ 俯视拍照 OK 信号 ------
Set Out[7],OFF;
##------ 俯视拍照 NG 信号 ------
Set Out[8],OFF;
##------ 俯视拍照 NG 信号 ------
Set Out[9],OFF;
##------ 贴标完成信号 ------
Set Out[10],OFF;
##------ 破真空信号 ------
Set Out[15],OFF;
##------ 打开通讯端口，机器人做服务器时可注释掉 ------
While  B0 <> 1
Open Socket(“192.168.24.15”,2000,2000,B0) ;
EndWhile;
##------ 机器人运动到安全点 ------
L[0]:
Jump P[0],V[30],Z[0],LH[10],MH[-40],RH[10];
L[1]:
##------IO 复位 ------
Set Out[3],OFF;
Set Out[4],OFF;
Set Out[5],OFF;
Set Out[6],OFF;
Set Out[7],OFF;
Set Out[8],OFF;
Set Out[9],OFF;
Set Out[10],OFF;
##------ 等待允许取标信号，若无信号则跳转到标签 L0------
Wait In[3]==ON,T[0.1],Goto L[0];
```

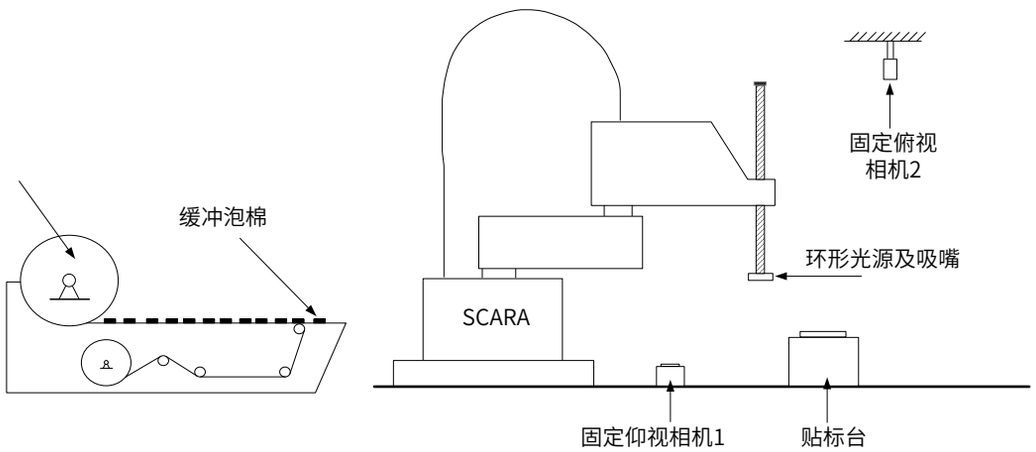
```
##------ 机器人运动到取标点，在到达点前 0.2s 打开吸嘴电磁阀 ------
Jump P[1],V[30],Z[0],Out(3,ON,T[-0.2]),LH[10],MH[-40],RH[10];
WaitInPos;
##------ 延时 0.1s------
Delay T[0.1];
##------ 运动到仰视拍照点，同时触发取标完成信号 ------
##------ 此处特别注意，P2 点角度应为 0 度，保证动态工具坐标系方向与基坐标系一致 ------
Jump P[2],V[30],Z[0],Out(4,ON,T[-0.2]),LH[10],MH[-40],RH[10];
WaitInPos;
##------ 延时 0.1s，保证机器人处于完全停止状态 ------
Delay T[0.1];
##------ 发送字符串 TA，触发仰视相机拍照 ------
SetPortBuf(str_send1) ;
Send Port[2000];
##------ 接收数据 ------
L[2]:
Get Port[2000],T[0],Goto L[2];
str_recv1 =GetPortbuf(0,100);
##------ 打印接收到的数据 ------
Print str_recv1;
##------ 如果接收到的数据为 NGA，视为仰视拍照 NG，抛料，返回标签 L1------
R0 = Strcmp(str_recv1,str_NGA);
If  R0 == 0
Set Out[6],ON;
Goto L[1];
EndIf;
##------ 视觉发送过来的数据格式为 X1，Y1，A1，进行解析 ------
B1 =StrGetData(str_recv1,”,”,D0);
##------P[10] 点为仰视相机像素坐标系下的点 ------
P[10] =(D0,D1,-61.157,D2,0,0),(-1,0,0,0),(5,0,1);
##------ 将像素坐标系下的点转换到基坐标系下点 P11 中 ------
Cnvrt(P[10],P[11],World);
##------ 打印点 P11------
Print P[11];
##------ 将 P11 的 A 值存入寄存器 D5 中 ------
GetPValue(P[11],3,D5);
##------ 设定 P11 点角度值为 0，目的是使动态工具坐标系方向与基坐标系一致 ------
SetPValue(P[11],3,0);
##------ 建立动态工具坐标系 ------
PR1 =Msft(P[2],P[11]);
SetToolParm(1,PR1);
##------ 发送仰视拍照完成信号给 PLC------
Set Out[5],ON;
##------ 运行到随动俯视相机拍照点 ------
Movj P[3],V[30],Z[0];
WaitInPos;
##------ 触发俯视相机拍照 ------
SetPortBuf(str_send2) ;
Send Port[2000];
##------ 接受端口数据 ------
L[3]:
Get Port[2000],T[0],Goto L[3];
```

```
##------ 将缓存区中数据取出赋值给字符串变量 ------
str_recv2 =GetPortbuf(0,100);
##------ 打印所接受到的数据 ------
Print str_recv2;
##------ 若接收到 NGB，则俯视拍照 NG，抛料，返回流程 L1------
R1 = Strcmp(str_recv2,str_NGB);
If  R1 == 0
Set Out[8],ON;
Goto L[1];
EndIf;
##------ 收到的数据格式为 X2，Y2，A2，解析后赋值给 D 变量 ------
B2 = StrGetData(Str2,”,”,D10);
##------P20 为相机像素坐标系下的点 ------
P[20] =(D10,D11,-116.3,D12,0,0),(-1,0,0,0),(6,0,2);
##------ 将像素坐标系下的点转换到工具 1 坐标系下的点 P21，随动拍照基准点为 P3------
Cnvrt(P[20],P[21],Tool[1],P[3]);
##------ 将 P21 点的 A 值赋值给 D15------
GetPValue(P[21],3,D15);
##------ 将仰视拍照的角度值与俯视拍照的角度值做差 ------
D16 =D15-D5;
##------ 取反 ------
D25 =0-D16;
##------D25 为真实贴合角度 ------
SetPValue(P[21],3,D25);
##------ 设置贴合高度 ------
SetPValue(P[21],2,-116.5);
##------ 打印贴合点位 ------
Print P[21];
##------ 补偿量预留 ------
PR21 = (0,0,0,0,0,0);
##------ 机器人运动到贴合点位 ------
Jump Offset(P[21],PR21),V[30],Z[0],LH[0],MH[-40],RH[10];
WaitInPos;
##------ 破真空 ------
Set Out[3],OFF;
Set Out[15],ON;
Delay T[0.1];
PR5 = (0,0,10,0,0,0);
##------ 上抬 10mm------
Movj Offset(PE,PR5),V[30],Z[0];
##------ 发送贴合完成信号 ------
Set Out[10],ON;
Goto L[1];
END;
```

3.3 上下对位贴标 / 自动标定案例

3.3.1 系统架构

固定仰视相机 + 固定俯视相机 +scara 机器人。



自动标定即将机器人与视觉的标定关系做在上位机 / 视觉软件侧，相比于将标定关系做在机器人侧，标定关系做在上位机侧的优势在于：

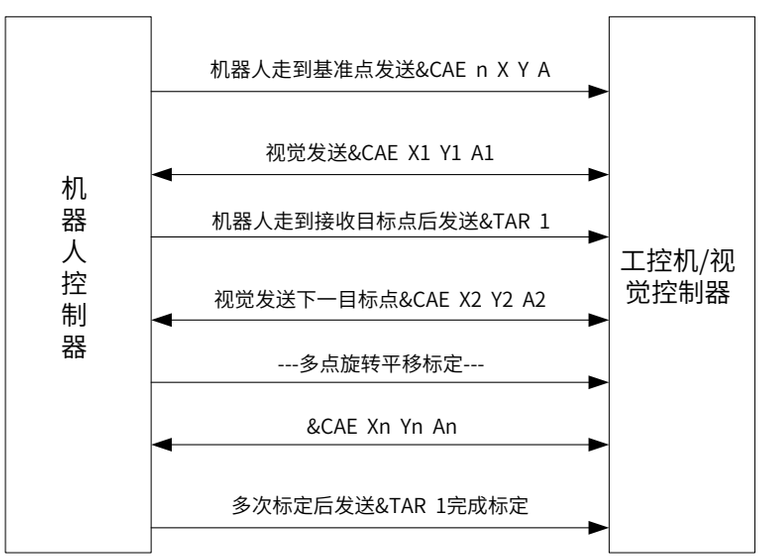
1） 标定效率的提高：一般进行和机器人系统交互的自动标定相比手动标定可以节省标定流程的时间；

2） 标定精度的提高：自动标定用相机对标定点进行识别，可消除由肉眼对点引起的标定误差；

3） 调试效率的提高与系统维护的便利：将标定工作做在视觉软件测，对标定结果的验证、补偿量的调整均可在视觉软件侧（PC 软件）进行，方便调试的进行。

3.3.2 自动标定的交互过程

1 自动标定的交互流程



■ 步骤 1：机器人携带标定工具（刀片、标定块等），通过视觉图像匹配工具，识别标定工具特征点，示教到相机视野中心且清晰成像，取此点位标定基准点 P2；

- 步骤 2：编写自动标定程序（详见 3. 程序及解析）；
- 步骤 3：建立通讯连接，运行自动标定程序，流程如上图所示；
- 步骤 4：标定完成后，验证标定结果。

2 标定结果的验证方法

- 方法 1：

完成自动标定后，通过视觉图像匹配工具，识别标定工具特征点，将此特征点坐标显示在视觉软件界面上，注意：此坐标为标定后视觉和机器人基坐标系所对应的绝对坐标（即 XYZA 坐标系）。通过机器人在基坐标系下单动 X,Y 轴移动一段已知距离，观察相机下显示坐标的 XY 值的变化，验证标定结果。

如：

相机坐标系下当前坐标为（113.86,75.13），且坐标数据波动在 +-0.02mm 以内（视相机像素当量而定，一般在 1-2 个像素以内波动为宜）；

现将机器人向 X 轴正方向移动 10mm，此时相机下理论坐标应为（123.86,75.13），若下相机实际识别坐标为（123.86+-0.05,75.13+-0.05），则此标定结果通过验证，否则验证不通过；

- 方法 2：

视觉软件在进行旋转平移标定后，可以将 X,Y 方向的像素当量、机器人平台坐标（机器人末端旋转中心、丝杆中心）显示在标定界面上，用户可进行两次以上的重复标定，观察不同标定次数下的标定结果是否一致来判断标定结果是否通过。

如：

在一个视野为 10X10mm 的相机图像下（正方形视野），标定的两次 X,Y 方向的像素当量分别为（0.013421,0.013471）、（0.013462,0.013453），若两次结果的像素当量较为接近，则判断标定结果通过。

再如：

两次旋转平移标定后机器人平台坐标（丝杆中心坐标）分别为（142.984,67.771）以及（142.723,67.677），两次标定的平台坐标相差较多（超过 0.1mm），则标定结果不通过，需进行更多次标定，直到两次相邻标定结果相差较小。

若标定结果一直无法通过，需通过左右手方法验证机器人绝对定位精度、以及从视觉打光、视觉识别精度来解耦此问题。此处不做赘述。

3.3.3 自动标定程序及解析（位置变量请自行添加）

```
START;
##========== 一键标定 ==========
##--------- 变量定义 ------------
B0 =0;
B100 =0;
##------------- 运行到标定基准点 P2（示教）-------------
Movj P[2],V[30],Z[0];
##------- 客户端 Client 建立通讯连接 / 做服务器时可注释掉 -------
While B0<>1
Open Socket(“192.168.10.100”,8502,8502,B0);
EndWhile;
##-------------- 通讯字符定义 -----------------
String str_send1;
String str_send2=”&TAR 1”;
String str_rsp1;
String str_cn1=”&CAE”;
String str_cn2=”1”;
```

```
String str_SPACE=”  “;
String str_cnX;
String str_cnY;
String str_cnA;
##---------- 获取基准点的 X,Y,A 坐标值并赋值给字符串变量 ----------
GetPValue(P[2],0,LD0);
str_cnX = DToStr(LD0,3,3);
GetPValue(P[2],1,LD1);
str_cnY = DToStr(LD1,3,3);
GetPValue(P[2],3,LD2);
str_cnA = DToStr(LD2,1,3);
##------ 字符串拼接：CAE 1 X Y A------
str_send1 =str_cn1 + str_SPACE + str_cn2 + str_SPACE + str_cnX + str_SPACE + str_cnY + str_SPACE + str_cnA;
##------ 数据交互：机器人发送当前位置给视觉，视觉回复下一目标点位给机器人，循环九次 ------
SetPortBuf(str_send1) ;
Send Port[8502];
For R0=1,R0<=9,Step[1]
L[1]:
Get Port[8502],T[0],Goto L[1];
##---------- 数据解析 ----------
str_rsp1 = GetPortbuf(5,100);
Print str_rsp1;
B100 = StrGetData(str_rsp1,”  “,D0);
##---------- 运行目标点 P0，且定义运行高度 ----------
P[0] =(D0,D1,-76.124,D2,0,0),(1,0,0,0),(2,0,0);
Movj P[0],V[30],Z[0];
WaitInPos;
##--------- 延时 0.5s，可根据具体工况修改 ---------
Delay T[0.5];
WaitInPos;
##---------- 到位后回复到位指令 &TAR 1----------
SetPortBuf(str_send2) ;
Send Port[8502];
WaitInPos;
EndFor;
##---------- 标定完成收到视觉发送 &CAE 0--------
L[2]:
Get Port[8502],T[0],Goto L[2];
str_rsp2 = GetPortbuf(0,100);
Print str_rsp2;
END;
```

3.3.4 贴合流程及程序解析（位置变量请自行添加）

1 贴合流程

- 步骤 1：进行固定仰视、固定俯视相机和机器人的自动标定；
- 步骤 2：验证自动标定的标定精度；
- 步骤 3：按照所绘制流程图进行程序编写；
- 步骤 4：低速试运行程序，进行程序修改与优化；
- 步骤 5：正常运行程序并保持生产状态的监控与数据记录（良率、产能等）。

2 程序解析

```
START;
##------ 变量定义 ------
B0 =0;
B1 =0;
B2 =0;
R0 =0;
R1 =0;
##------ 通讯字符串变量定义 ------
String str_send1=”TA”;
String str_send2=”TB”;
String str_NGA=”NGA”;
String str_NGB=”NGB”;
String str_recv1;
String str_recv2;
##======IO 复位 ======
##------ 吸嘴真空电磁阀信号 ------
Set Out[3],OFF;
##------ 取标完成信号 ------
Set Out[4],OFF;
##------ 仰视拍照 OK 信号 ------
Set Out[5],OFF;
##------ 仰视拍照 NG 信号 ------
Set Out[6],OFF;
##------ 俯视拍照 OK 信号 ------
Set Out[7],OFF;
##------ 俯视拍照 NG 信号 ------
Set Out[8],OFF;
##------ 俯视拍照 NG 信号 ------
Set Out[9],OFF;
##------ 贴标完成信号 ------
Set Out[10],OFF;
##------ 破真空信号 ------
Set Out[15],OFF;
##------ 打开通讯端口，机器人做服务器时可注释掉 ------
While  B0 <> 1
Open Socket(“192.168.24.15”,2000,2000,B0) ;
EndWhile;
##------ 机器人运动到安全点 ------
L[0]:
Jump P[0],V[30],Z[0],LH[10],MH[-40],RH[10];
L[1]:
##------IO 复位 ------
Set Out[3],OFF;
Set Out[4],OFF;
Set Out[5],OFF;
Set Out[6],OFF;
Set Out[7],OFF;
Set Out[8],OFF;
Set Out[9],OFF;
Set Out[10],OFF;
##------ 等待允许取标信号，若无信号则跳转到标签 L0------
```

```
Wait In[3]==ON,T[0.1],Goto L[0];
##------ 机器人运动到取标点，在到达点前 0.2s 打开吸嘴电磁阀 ------
Jump P[1],V[30],Z[0],Out(3,ON,T[-0.2]),LH[10],MH[-40],RH[10];
WaitInPos;
##------ 延时 0.1s------
Delay T[0.1];
##------ 运动到仰视拍照点，同时触发取标完成信号 ------
Jump P[2],V[30],Z[0],Out(4,ON,T[-0.2]),LH[10],MH[-40],RH[10];
WaitInPos;
##------ 延时 0.1s，保证机器人处于完全停止状态 ------
Delay T[0.1];
##------ 发送字符串 TA，触发仰视相机拍照 ------
SetPortBuf(str_send1) ;
Send Port[2000];
##------ 接收数据 ------
L[2]:
Get Port[2000],T[0],Goto L[2];
str_recv1 =GetPortbuf(0,100);
##------ 打印接收到的数据 ------
Print str_recv1;
##------ 如果接收到的数据为 NGA，视为仰视拍照 NG，抛料，返回标签 L1------
R0 = Strcmp(str_recv1,str_NGA);
If  R0 == 0
Set Out[6],ON;
Goto L[1];
EndIf;
##------ 发送仰视拍照完成信号给 PLC------
Set Out[5],ON;
##------ 运行到随动俯视相机拍照点 ------
Movj P[3],V[30],Z[0];
WaitInPos;
##------ 触发俯视相机拍照 ------
SetPortBuf(str_send2) ;
Send Port[2000];
##------ 接受端口数据 ------
L[3]:
Get Port[2000],T[0],Goto L[3];
##------ 将缓存区中数据取出赋值给字符串变量 ------
str_recv2 =GetPortbuf(0,100);
##------ 打印所接受到的数据 ------
Print str_recv2;
##------ 若接收到 NGB，则俯视拍照 NG，抛料，返回流程 L1------
R1 = Strcmp(str_recv2,str_NGB);
If  R1 == 0
Set Out[8],ON;
Goto L[1];
EndIf;
##------ 收到的数据格式为 X2，Y2，A2，解析后赋值给 D 变量 ------
B2 = StrGetData(Str2,”,”,D10);
##------ 将相机发送的基坐标系下 X,Y,A 值赋给机器人位置变量 ------
P[20] =(D10,D11,-116.3,D12,0,0),(-1,0,0,0),(6,0,2);
##------ 设置贴合高度 ------
SetPValue(P[20],2,-116);
##------ 打印贴合点位 ------
Print P[20];
##------ 补偿量预留 ------
PR20 = (0,0,0,0,0,0);
##------ 机器人运动到贴合点位 ------
Jump Offset(P[20],PR20),V[30],Z[0],LH[0],MH[-40],RH[10];
WaitInPos;
##------ 破真空 ------
Set Out[3],OFF;
Set Out[15],ON;
Delay T[0.1];
PR5 = (0,0,10,0,0,0);
##------ 上抬 10mm------
```

```
Movj Offset(PE,PR5),V[30],Z[0];
##------ 发送贴合完成信号给 PLC------
Set Out[10],ON;
WaitInPos;
Goto L[1];
END;
```

3.4 用户坐标系类——两点拍照，多点贴合

1 多目标贴合的应用场合

在需要对多个目标进行组装、贴合的场合，往往采用建立动态用户坐标系的方法计算这些多目标贴合点位，以大大提高工作效率。

2 随动贴合的数学模型

随动贴合项目的应用关键点在于建立动态用户坐标系，通过识别工作平面上的两个 mark 点建立动态用户坐标系（通过 SetUserParm 指令），所有的工作点位都基于工作平面上的这两个 mark 点来进行计算。

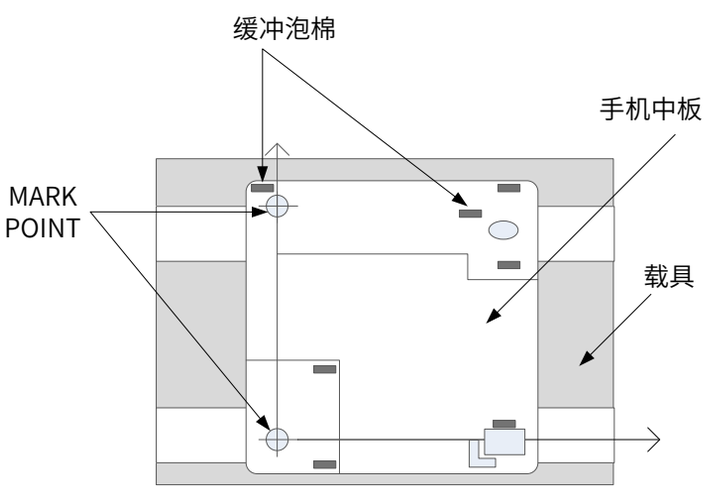


图 3-3 原理图

3 程序解析

```
START;
##------ 变量定义 ------
B0 =0;
B1 =0;
B2 =0;
##------B3 为贴合号 ------
B3 =10;
R0 =1;
R1 =1;
##------ 通讯字符串变量定义 ------
String str_send1=”TA1”;
String str_send2=”TA2”;
String str_send3=”TB”;
String str_NGA=”NGA”;
String str_NGB=”NGB”;
String str_NGup=”NGup”;
String str_OK=”OK”;
String str_recv1;
String str_recv2;
String str_recv3;
```

```
##======IO 复位 ======
##------ 吸嘴真空电磁阀信号 ------
Set Out[3],OFF;
##------ 取标完成信号 ------
Set Out[4],OFF;
##------ 仰视拍照 OK 信号 ------
Set Out[5],OFF;
##------ 仰视拍照 NG 信号 ------
Set Out[6],OFF;
##------ 俯视拍照 OK 信号 ------
Set Out[7],OFF;
##------ 俯视拍照 NG 信号 ------
Set Out[8],OFF;
##------ 俯视拍照 NG 信号 ------
Set Out[9],OFF;
##------ 单次贴标完成信号 ------
Set Out[10],OFF;
##------ 整体工件贴标完成信号 /6 次贴合完成信号 ------
Set Out[11],OFF;
##------ 破真空信号 ------
Set Out[15],OFF;
##------ 打开通讯端口，机器人做服务器时可注释掉 ------
While  B0 <> 1
Open Socket(“192.168.24.15”,2000,2000,B0) ;
EndWhile;
##------ 机器人运动到安全点 ------
L[0]:
Jump P[0],V[30],Z[0],LH[10],MH[-40],RH[10];
##------IO 复位 ------
Set Out[3],OFF;
Set Out[4],OFF;
Set Out[5],OFF;
Set Out[6],OFF;
Set Out[7],OFF;
Set Out[8],OFF;
Set Out[9],OFF;
Set Out[10],OFF;
##------ 等待允许拍照信号，若无信号则跳转到标签 L0------
Wait In[3]==ON,T[0.1],Goto L[0];
##------ 机器人运动到拍照点 1------
Movj P[1],V[30],Z[0];
WaitInPos;
##------ 延时 0.1s，等待机台稳定 ------
Delay T[0.1];
##------ 发送视觉拍照指令 ------
SetPortBuf(str_send1) ;
Send Port[2000];
##------ 接收 OK/NG 字符 ------
L[1]:
Get Port[2000],T[0],Goto L[1];
str_recv1 = GetPortbuf(0,100);
##------ 将接受到的字符串和 NGA 比较 ------
R0 =Strcmp(str_recv1,str_NGA);
##------ 若拍照 NG，则发送 NG 信号给 PLC，并返回主流程 ------
If  R0 == 0
Set Out[8],ON;
Goto L[0];
EndIf;
##------ 若拍照 OK，则解析数据 ------
B1 =StrGetData(str_recv1,”，”,D0);
##------ 将拍照点位赋值给坐标点（位置变量）------
```

```
P[1] =(D0,D1,-80,D2,0,0),(1,0,0,0),(2,0,0);
##------ 机器人运动到拍照点 2------
Movj P[2],V[30],Z[0];
WaitInPos;
##------ 延时 0.1s，等待机台稳定 ------
Delay T[0.1];
##------ 发送视觉拍照指令 ------
SetPortBuf(str_send2) ;
Send Port[2000];
##------ 接收 OK/NG 字符 ------
L[2]:
Get Port[2000],T[0],Goto L[2];
str_recv2 =GetPortbuf(0,100);
##------ 将接受到的字符串和 NGB 比较 ------
R1 =Strcmp(str_recv2,str_NGB);
##------ 若拍照 NG，则发送 NG 信号给 PLC，并返回主流程 ------
If  R1 == 0
Set Out[8],ON;
Goto L[0];
EndIf;
##------ 若拍照 OK，则解析数据 ------
B1 =StrGetData(str_recv2,”，”,D10);
##------ 将拍照点位赋值给坐标点（位置变量）------
P[2] =(D10,D11,-80,D12,0,0),(1,0,0,0),(2,0,0);
##------ 建立动态用户坐标系 ------
PR1 =Msft(P[1],P[2]);
SetUserParm(1,PR1);
##------ 根据图纸尺寸贴合要求，求出各个贴合点在用户坐标系 1 下的坐标 ------
##------ 共 6 个贴合点，变量号为 P10-P15------
P[10] =(37.5,42,-80,0,0,0),(1,0,0,0),(4,0,1);
P[11] =(57.5,47,-80,0,0,0),(1,0,0,0),(4,0,1);
P[12] =(27.5,57,-80,0,0,0),(1,0,0,0),(4,0,1);
P[13] =(92,31,-80,0,0,0),(1,0,0,0),(4,0,1);
P[14] =(75.5,63,-80,0,0,0),(1,0,0,0),(4,0,1);
P[15] =(81,86,-80,0,0,0),(1,0,0,0),(4,0,1);
##------ 机器人取标流程 L10------
L[10]:
##------ 机器人等待允许取标信号 ------
Wait In[3]==ON,T[0],Goto L[10];
##------ 运动到取标点 ------
Jump P[3],V[30],Z[0],LH[0],MH[-65],RH[5];
WaitInPos;
##------ 取标真空开 ------
Set Out[3],ON;
Delay T[0.1];
##------ 取标完成，运行到仰视相机拍照点 ------
Jump P[4],V[30],Z[0],LH[0],MH[-65],RH[0];
WaitInPos;
Delay T[0.1];
##------ 触发仰视相机拍照 ------
SetPortBuf(str_send3) ;
Send Port[2000];
##------ 接收端口数据 ------
L[5]:
Get Port[2000],T[0],Goto L[5];
##------ 接收缓存区数据 ------
str_recv3 =GetPortbuf(0,100);
##------NG 判断 ------
R2 =Strcmp(str_recv3,str_NGup);
##------ 若接受到 NG 信号，则到 P100 点抛料，返回取标流程 ------
If  R2 == 0
```

```
Set Out[6],ON;
Jump P[100],V[30],Z[0],LH[0],MH[-65],RH[0];
Set Out[3],OFF;
##------ 打开破真空 ------
Set Out[15],ON;
Delay T[0.1];
Set Out[15],OFF;
Goto L[10];
EndIf;
##------ 若拍照 OK 则解析数据 ------
B2 = StrGetData(str_recv3,”，”,D20);
##------ 建立动态工具坐标系 ------
P[20] =(D20,D21,-80,D22,0,0),(1,0,0,0),(2,0,0);
PR2 =Msft(P[4],P[20],Tcp);
SetToolParm(1,PR2);
##------ 机器人运动到贴合点位 ------
##------ 注意：PB3 为 P10-P15 的点（USER1 下的点），用动态工具坐标系 TOOL1 运行 ------
Jump P[B3],V[30],Z[0],Tool[1],LH[0],MH[-65],RH[10];
WaitInPos;
##------ 破真空 ------
Set Out[3],OFF;
Set Out[15],ON;
Delay T[0.1];
PR5 = (0,0,10,0,0,0);
##------ 上抬 10mm------
Movj Offset(PE,PR5),V[30],Z[0];
##------ 发送单次贴合完成信号 ------
Set Out[10],ON;
WaitInPos;
##------ 贴合循环计算 ------
##------ 若未贴满 6 次则跳转到取标循环，若贴满 6 次则跳转到主循环 ------
Incr B3;
If  B3 > 15
B3 =10;
##------ 整体贴合完成信号 ------
Set Out[11],ON;
Goto L[0];
Else
Goto L[10];
EndIf;
##------ 返回主流程 ------
END;
```

第 4 章 典型工艺应用

4.1 动态抓取

4.1.1 案例概述

动态跟随，亦称动态追踪，是指在传送带（直线或圆盘式传送装置）不停止的状态下，机器人通过与其建立动态同步关系，然后完成对传送带上的工件进行抓取、装配、分拣等操作。广泛应用于工业制造、食品、药品等需要大批量上下料且效率高的场合。

■ 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 MovJ、Jump 运动；

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。

4.1.2 系统组成

下图为机器人动态抓取系统的硬件架构图，包含传送带、机器人视觉系统等主要部件。

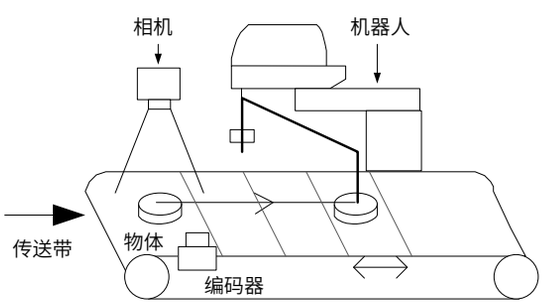


图 4-1 系统组成示意图

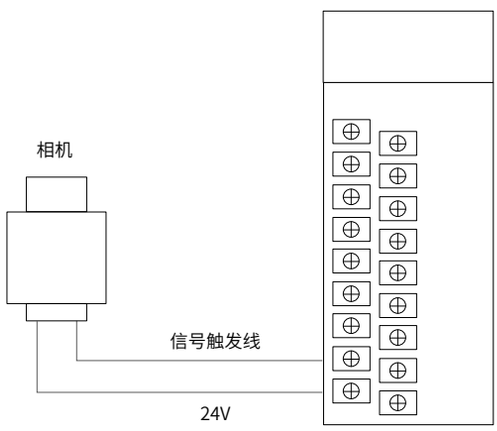
编号	名称	规格
1	SCARA 机器人整套	SCARA400 机器人本体、电控柜、示教器
2	IRLink 编码器采集模块	支持 5V 外置编码器信号的采集
3	编码器	单向、差分均可（5V 电平）
4	视觉控制器	低电平触发 / 输入信号
5	相机	130 万像素（根据检测精度确定）
6	光源和光源控制器	一对条形光源和 2 路光源控制器
7	传送带和相机安装支架	
8	吸嘴夹具	
9	标定纸板	大小规格根据相机视野大小确认
10	标定尖端	尖端处于夹具中心

4.1.3 电气接线

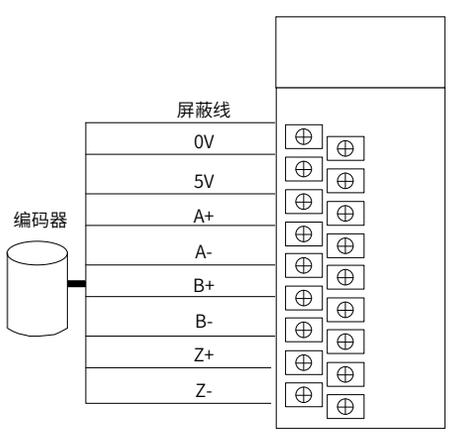
① 相机触发 IO 接线：

以汇川相机为例，红色线缆接机器人 IO 模块的 24V 端子，黑色线缆接机器人 IO 模块的输出端子。

相机信号触发信号线接机器人 IO 输出模块的 Out[6] 端子 , 相机电源线接机器人 IO 输出模块的 24V 端子，示意图如下。



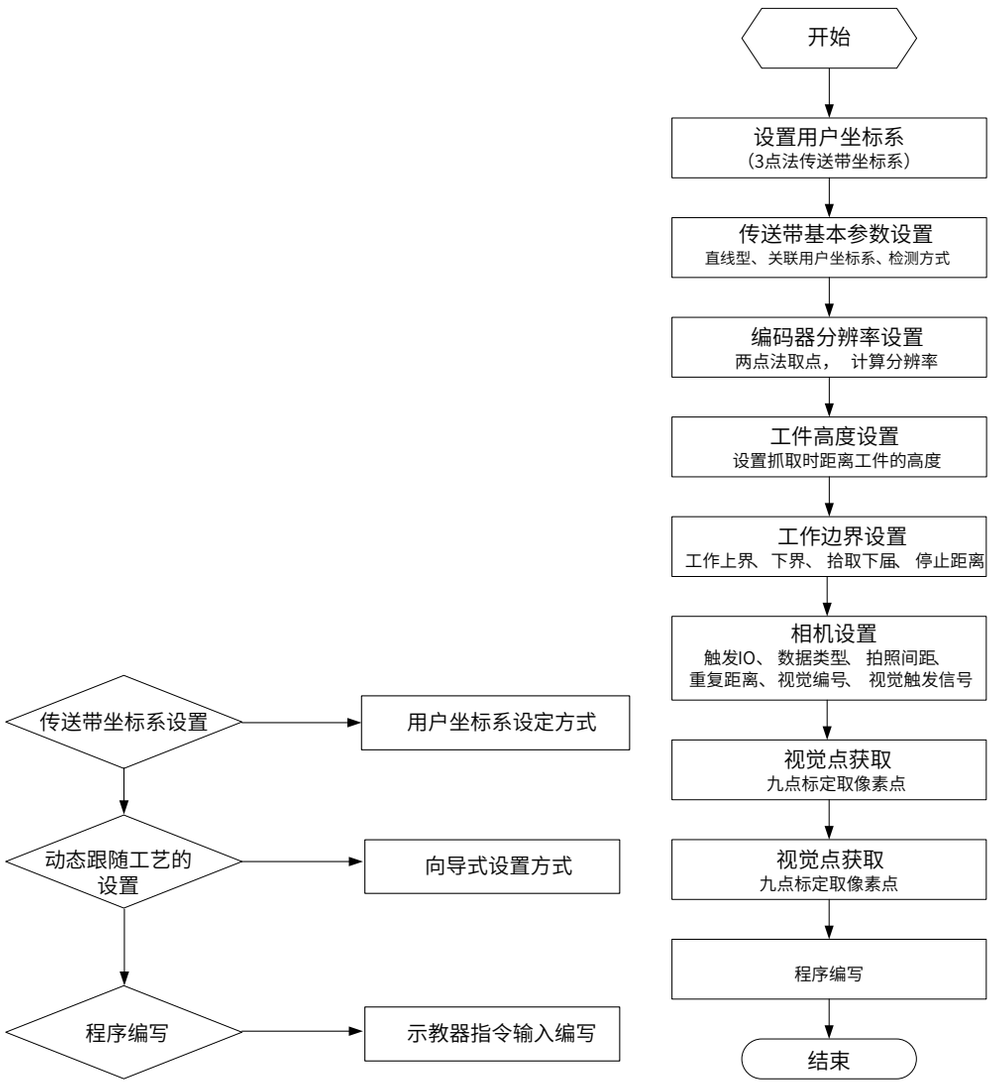
编码器接线示意图：



序号	网络名	类型	功能	备注
1\2	PE		PE 接地点	
3\4	GND	电源输出 (+-10%)	5V 电源输出	5V 电源输出
5\6	5V			
7	ENC0_PA+	最高输入频率：1Mbps	差分输入 A 相	A\B 相差 90 度
9	ENC0_PA-			
11	ENC0_PB+		差分输入 B 相	
13	ENC0_PB-			
15	ENC0_PZ+		差分输入 Z 相	Z 相信号输入
17	ENC0_PZ-			
8	ENC1_PA+		差分输入 A 相	A\B 相差 90 度
10	ENC1_PA-			
12	ENC1_PB+		差分输入 B 相	
14	ENC1_PB-			
16	ENC1_PZ+		差分输入 Z 相	Z 相信号输入
18	ENC1_PZ-			

4.1.4 操作流程

该工艺操作流程主要包含传送带坐标系设置、动态跟随工艺设置、程序编写 3 个环节。



具体参数设置可以参照《高级功能应用》相关指导。

■ 注意事项：

1） 先要固定好机器人底座、传送带，保证平台没有晃动。

2） 用户坐标系的 X 轴方向要与传送带运动方向一致。

3） 编码器的屏蔽线一定要接，否则编码器信号会受到干扰，造成编码器信号不准确，进而导致报传送带异常、抓取不准等问题。

4） 严格按照向导式操作执行每一个步骤，不能省略相关参数设置。

4.1.5 程序案例

本实例以榨菜包的抓取作为程序实例

在视觉配合下，在直线传送带不停止的前提下，机器人从 P[0] 动态抓取料包 , 然后放置于指定位置 P[2] 处，以此循环，完成料包的动态抓取过程。

图 4-2 料包实物图

图 4-3 视觉处理下的料包

图 4-4 实验装置实物图

1 工作过程示意图

图 4-5 中，P[0] 是工件上的坐标点，坐标系是工件坐标系，编号 7，P[1] 是机器人等待点位，P[2] 是放料点位。

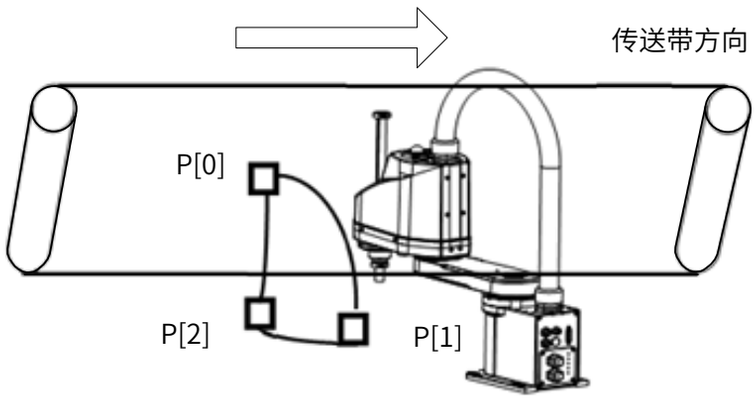


图 4-5 工作过程示意图

2 程序实例

```
START                                          ## 程序开始标志
B0=0;                                          ##B0 变量初始化
P[0]=(0,0,0,0,0,0),(0,0,0,0),(7,0,2);          ## 抓取对象上的坐标点，7 表示工件坐标系
While B0<>1
        Open Socket(“192.168.24.55”,2000,B0);  ## 建立视觉坐标系的连接
EndWhile;
L[0];                                          ## 程序跳转标识 L[0]
Jump P[1],V[100],Z[0],LH[10],MH[-50],RH[10];   ## 机器人等待点，一般此点设置在工作上界附近
ConVision(Converyor[0],ON);                    ## 打开视觉坐标系
L[1];                                          ## 程序跳转标识 L[1]
Get CnvObjet(0,1),Goto L[1];                   ## 查询对象，当查询到时，程序往下执行，查询不到，跳转到 L[1]
ResSys Converyor[0];                           ## 切换到传送带坐标系，建立同步关系
Jump P[0],V[100],Z[0],LH[0],MH[10,RH[0]; .     ## 建立同步后，执行到工件坐标点
RefSys Base                                    ## 切换到机器人基坐标系，解除同步关系
Jump P[2],V[100],Z[0],LH[0],MH[10],RH[0];      ## 解除同步后，运行到放料点 P[2]
Goto L[1];                                     ## 程序循环
```

3 异常问题解决

■ 视觉数据等待超时

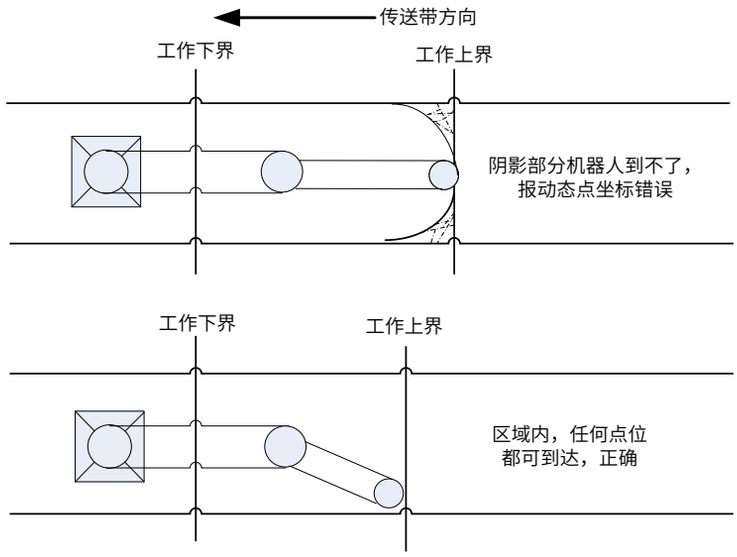
故障原因：视觉第一次发送过来的像素坐标还未被机器人收到时，相机又进行了第二次拍照。

解决办法：将拍照间距设置值调大一点，传送带速度稍微慢一点，确保机器人收到第一次拍照的像素坐标后，相机再进行第二次拍照；另外将视觉处理数据的时间减小（减小视觉的曝光时间等参数）。

■ 动态点坐标错误

故障原因：由于机器人在传送带上设置的工作上界和工作下界有关联关系，当机器人到达不了工作上界和工作下界内的某一个点位时，将会报此故障。

解决办法：将工作上界下界重新设置，让机器人能到达范围内的任何一点。



■ 传送带上没有料，机器人也去抓取

解决办法：重复检测距离太小，可将重复检测距离稍微改大一点。

■ 视觉处理时间过长

解决办法：拍照间距设置太短，导致相机拍照时间间隔太短，机器人未收到像素坐标，应将传送带速度调小或者减少视觉处理时间。

4.2 托盘 / 阵列应用

4.2.1 案例概述

托盘是在生产加工过程中对产品起转运或包装的物件，通过人力或运动执行机构将产品或工件依次放置在相应的位置中，一般情况下，该托盘的相邻位置属于等间距设置，如图 4-6 所示，由于摆放位置间等间距形成一定的阵列，因此该种应用也称为阵列应用。该种托盘主要应用在工件的装箱、食品药品分拣后的包装或加工环节的转运作用，对于大量的阵列分布，一一教点方式必然浪费大量的人力物力，汇川技术针对该类应用开发了专用的指令，由此大大简化了该类应用的调试时间。

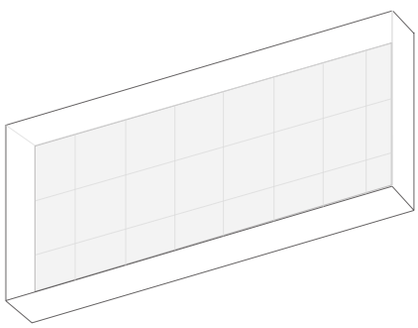
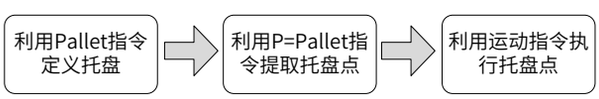


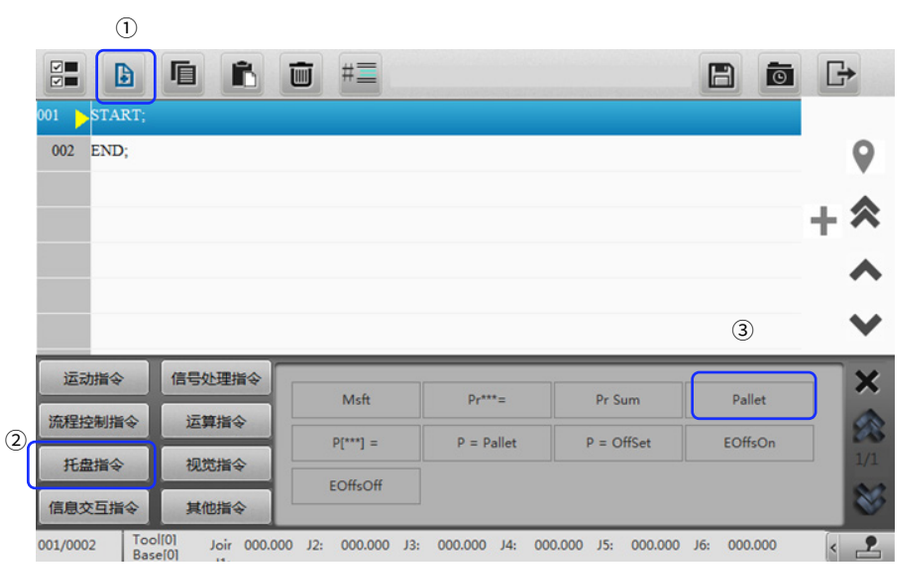
图 4-6 托盘示意图

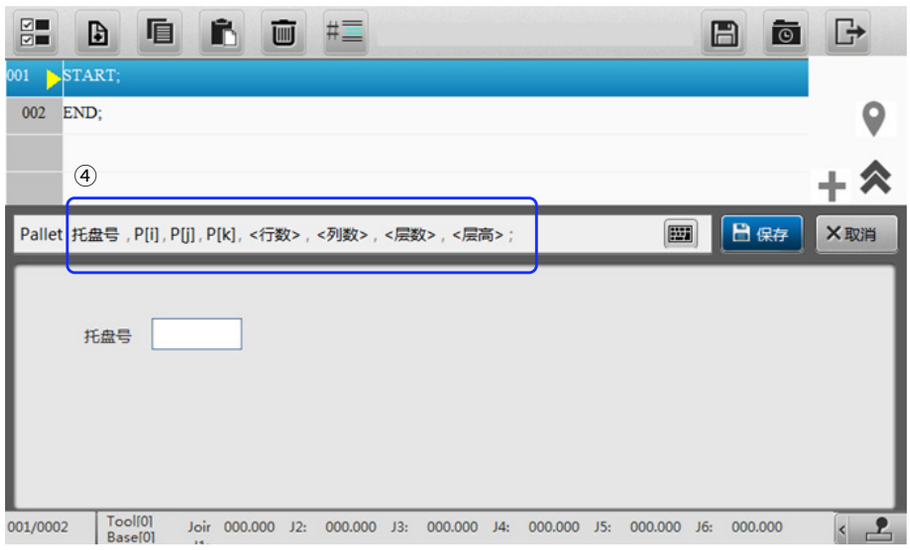
涉及相关指令： Pallet、P=Pallet，详细请参见《汇川机器人设计应用与维护手册 - 附录：机器人指令表》。

4.2.2 托盘建立步骤



1） 定义推盘：指定托盘序号，确定托盘的行列层及层高，输入相应的参数：





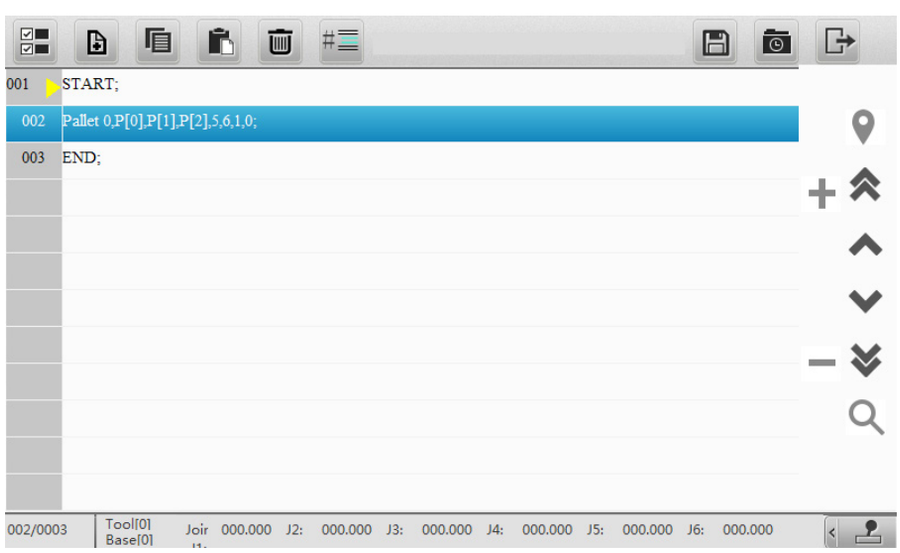
① 点击添加指令按钮；

② 点击托盘指令；

③ 选择 Pallet 指令；

④ 依次点击托盘号，坐标点，行列层及层高并设置参数，点击保存按钮即可。

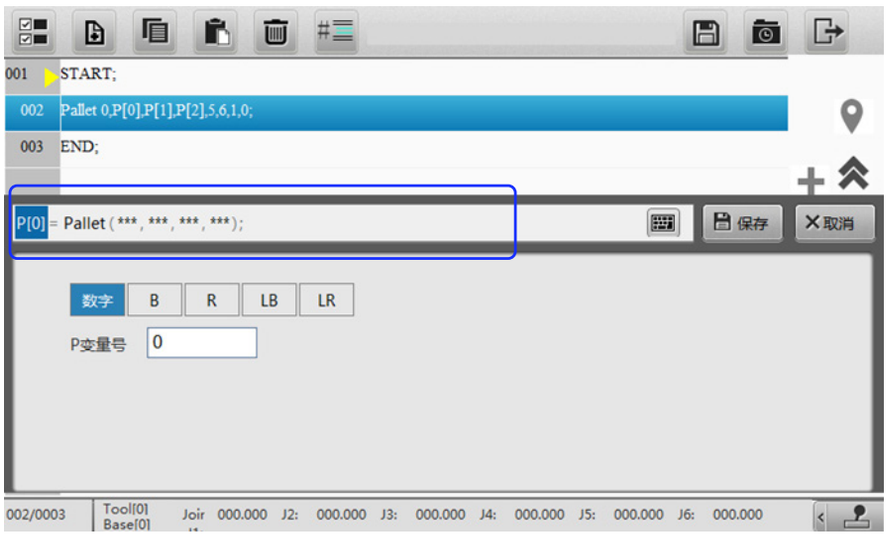
⑤ 至此，托盘号为 0，以 P0 为原点、P1 为行，以 P2 为列的 5 行 6 列 1 层的托盘即定义完毕，点击保存按钮完成程序保存。如下：

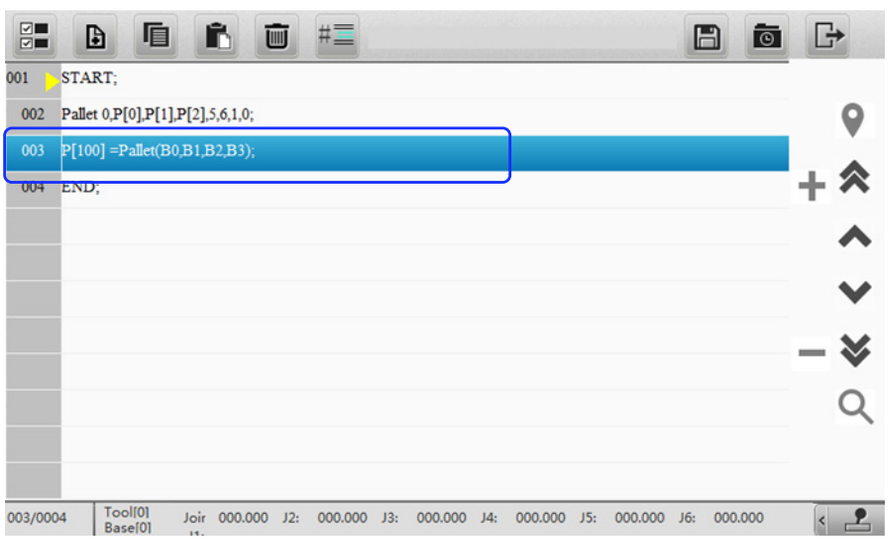


2） 提取托盘点位

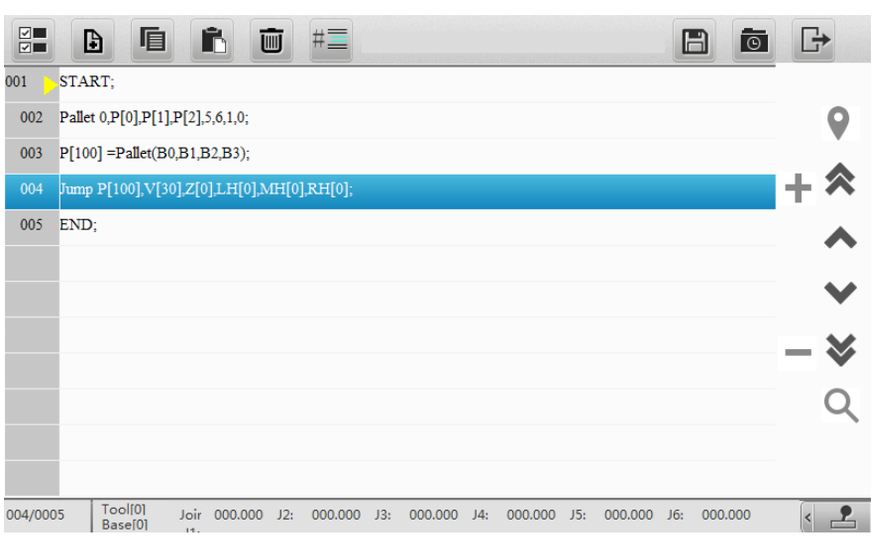
提取托盘点赋值给 P 点，需指定托盘号、行号、列号、层号。

由于托盘的号、行号、列号、层号均支持变量给定，通过提取托盘点赋值给 P 点，即可完成相应的赋值动作，点击保存，然后执行运动指令即可。



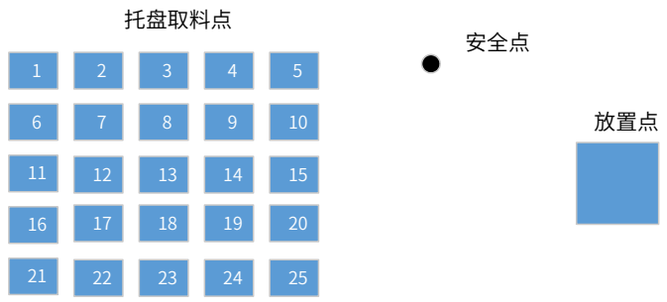


3） 执行运动指令，运动到相应的 P 点，如



4.2.3 数据配置

如图所示，建立一个托盘号位 1，5 行 1 列 1 层层高为 0 的托盘，从托盘提取工件后，到图示位置放置，然后返回，依次循环执行到托盘结束运行至等待点。



名称	点位	描述
托盘取料点	共 25 个点	取料位置
安全点	P[0]	取料之前或放料完成等待点
放置点	P[1]	取料后，工件放置点
托盘原点行列点	P[2] P[3] P[4]	P[2] 取于 1 号位
托盘提取点位	P[100]	提取托盘后执行点位

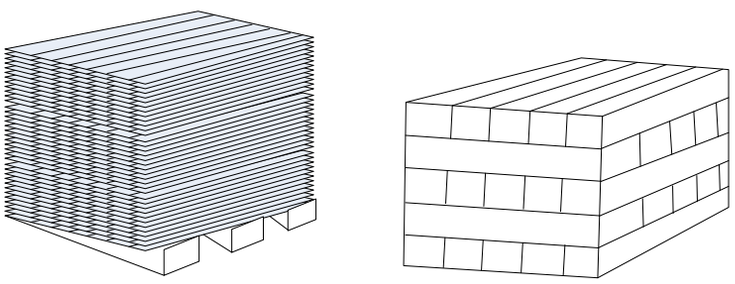
4.2.4 程序实例

```
START; ## 程序开始标志
Pallet 1,P[2],P[3],P[4],5,5,1,0;                    ## 定义托盘
Jump P[0],V[30],Z[0],LH[20],MH[-10],RH[20];         ## 运行到等待位
For B1=0,B1<5,Step[1]                               ## 行循环遍历
For B2=0,B2<5,Step[1]                               ## 列循环遍历
P[100] =Pallet(0,B1,B2,0);                          ## 提取托盘点
Jump P[100],V[30],Z[0],LH[20],MH[-10],RH[20];       ## 运行到指定的托盘点位
WaitInPos;                                          ## 等待到位指令
Delay T[0.2];                                       ## 延时 0.2 秒，模拟到位吸放动作
Jump P[1],V[30],Z[0],LH[20],MH[-10],RH[20];         ## 运行到工件放料位
WaitInPos;                                          ## 等待到位指令
Delay T[0.2];                                       ## 延时 0.2 秒，模拟工件放置动作
Jump P[0],V[30],Z[0],LH[20],MH[-10],RH[20];         ## 运行到取料等待位，可以不要
EndFor;                                             ##for 循环语法要求
EndFor;                                             ##for 循环语法要求
END;
```

4.3 码垛应用

码垛应用指将物料按照一定模式一件件堆成码垛，以便对单元化的码垛实现物料的搬运、存储、装卸运输等物流活动。

码垛分为两种：堆垛和分垛，在包装物流行业应用最为广泛。汇川机器人针对码垛应用，专门开发了码垛机专用指令。通过示教用户坐标系（可以理解为托盘坐标系）的方式，自动计算码垛中各跺位点位数据，节省示教时间，大大提高易用性。可针对不同跺型进行自定义选择、设置和模块化参数应用，大大节省调试时间。

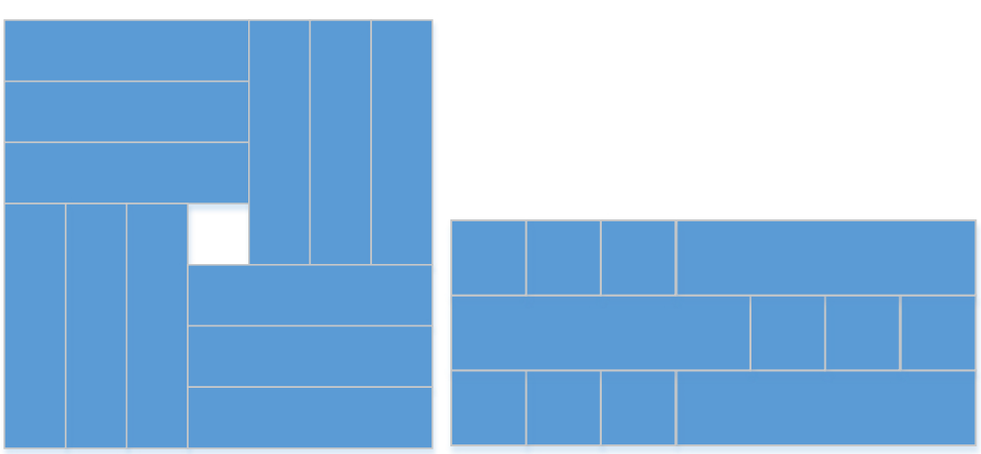


重叠式货垛　　纵横交错式货垛

图 4-7 常见的单元化码垛方式（示意图）

4.3.1 案例描述

- 夹具一次夹取两块料；
- 跺型采用旋转式，没区放三块料；
- 跺型示意图：



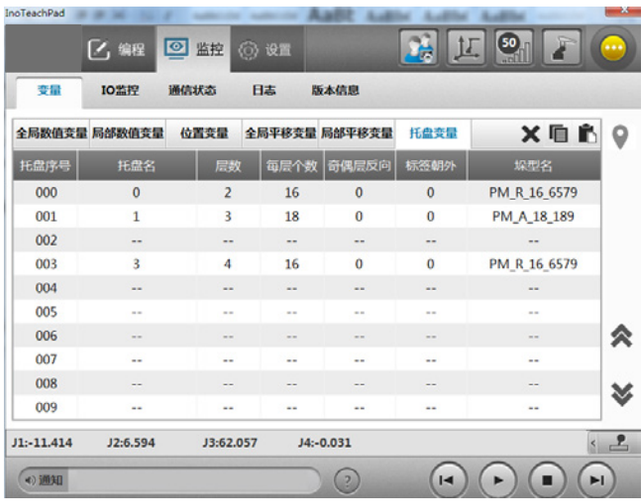
俯视图　　正视图

4.3.2 跺型建立步骤

1 建立跺型

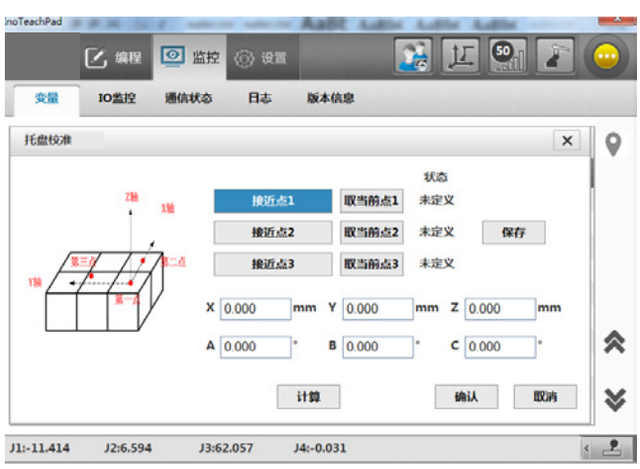
2 设置跺型参数

3 建立托盘



4 设置托盘尺寸参数

5 建立跺型基准点



- 第一个点选择：右垛：选择离机器人码垛最远的点；并且此时夹具方向和 X+ 方向相同，此时要和托盘的第一个放置点位置一致，点击“接近点一”，点击“取当前点”，即可。
- 第二个点选择：朝着 X+ 方向运动一段距离，不要超过托盘长度；点击“接近点二”，点击“取当前点”，即可。
- 第三个点选择：在第二个点基础上，朝着 Y+ 方向运行一段距离，点击“接近点三”，点击“取当前点”，即可。

点击“计算”、“确认”、“保存”。

左垛：选择离机器人码垛最远的点；并且此时夹具方向和 X+ 方向相同，此时要和托盘的第一个放置点位置一致，点击“接近点一”，点击“取当前点”，即可。

- 第二个点选择：朝着 X+ 方向运动一段距离，不要超过托盘长度；点击“接近点二”，点击“取当前点”，即可。
- 第三个点选择：在第二个点基础上，朝着 Y- 方向运行一段距离，点击“接近点三”，点击“取当前点”，即可。

点击“计算”、“确认”、“保存”。

4.3.3 数据配置

1 输入输出信号

In(3)	启动
In(4)	停止
In(5)	急停
In(7)	暂停
In(9)	子程序调用 1
In(10)	子程序调用 2
In11)	子程序调用 3
In(12)	上料 ready
In(13)	左托盘 ready
In(14)	右托盘 ready

Out(0)	启动输出
Out(1)	电磁阀 1
Out(2)	电磁阀 2
Out(3)	报警
Out(4)	完成
Out(5)	停止
Out(6)	单包堆放完成
Out(7)	堆放跺型完成

1） 机器人机械手开关时，等待时间为 0.1s；

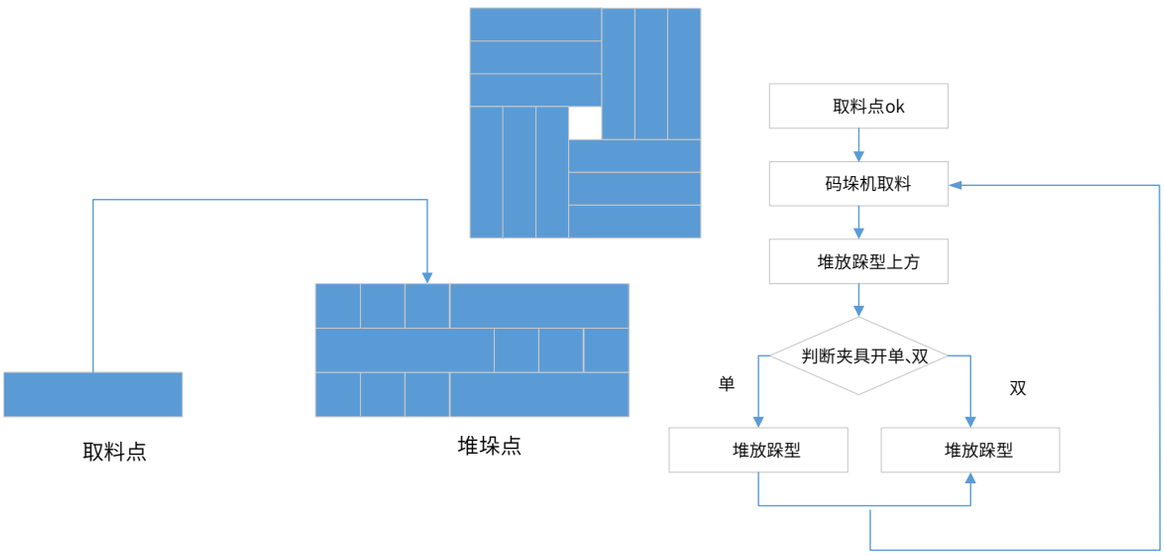
2） 以下坐标点预先输入了坐标点数据：

P0	待取料点
P4	托盘过渡点
P55	安全点

3） 移动时采用最高速度，移动至部件前时采用较慢速度动作；

4） 动作至托盘上方、部件供给装置上方 Z=50mm 的位置为止。

2 程序流程图



4.3.4 程序实例

```
START;
L[0]:
## 人工干预信号检测
##Wait IN[10]==ON,T[0];
## 检测右托盘信号
##Wait IN[8]==ON,T[0];
ReSetPallet[0];
IsPalletFinished(Pallet[0],B3);                    ## 清除托盘数据，复位各输出信号
##SetPalletRunNo(Pallet[0],41);
Set Out[4],ON;
Set Out[5],ON;
Set Out[6],ON;
Set Out[7],ON;
## 安全点
Movj P[55],V[100],Z[1];                            ## 机器人回安全点
B3 =1;
GetPalletRunNo(Pallet[0],R1);                      ## 获取托盘点位信号
## R10=0（不抓料，程序内部自加一）；R10=1(抓料，程序内部自加一)；R10=2(抓料，程序加二)
R10 =2;                                            ## 定义抓料信号赋值
 Switch R1
 Case 1 :
 Case 5 :
 Case 7 :
 Case 15 :
 Case 17 :
 Case 25 :
 Case 27 :
 Case 35 :
 Case 37 :
 Case 45 :
 Case 47 :
```

```
 Case 55 :
 Case 57 :
R10 =1;
Break;
 Case 6 :
 Case 16 :
 Case 26 :
 Case 36 :
 Case 46 :
 Case 56 :
R10 =0;
Break;
EndSwitch;
R5 =0;
While B3<37
If R10==2 Or R10==1
## 待取料点                                                    ## 判断抓手信号
If R5==0
Movl P[0],V[100],Z[2],Acc[70],
Out(6,ON,T[0.5]),Out(7,ON,T[0.5]);                             ## 待取料位复位 IO 抓手信号
Else
Movl P[0],V[100],Z[2],Acc[70],
Out(4,ON,T[0.5]),Out(5,ON,T[0.5]);
EndIf;
## 检测来料信号
##Wait IN[6]==ON,T[0];
## 取料点
Movl P[1],V[80],Z[0];
## 夹具关
Set Out[4],OFF;                                                ## 取料点关闭抓手信号
Set Out[5],OFF;
Set Out[6],OFF;
Set Out[7],OFF;
Delay T[1];
## 回待取料点
Movl P[0],V[100],Z[5];
EndIf;
If R10==2
Incr R1;
SetPalletRunNo(Pallet[0],R1);
EndIf;
## 进入托盘                                                    ## 设置跺型点位
 Switch R1
 Case 6 :
 Case 16 :
 Case 26 :
 Case 36 :
 Case 46 :
 Case 56 :
## 单包第二包进入托盘
```

```
MovToPut Pallet[0],P[53],30,50,350,
V[100],PickV[40];
Break;                                                    ## 放置物料至跺型各位置
Default:
## 进入托盘
MovToPut Pallet[0],P[4],30,50,150,
V[100],PickV[40];
Break;
EndSwitch;
R20 =0;
 Switch R1
 Case 15 :
 Case 25 :
 Case 36 :
 Case 55 :
R20 =1;
Break;
 Case 16 :
 Case 26 :
 Case 35 :
 Case 56 :
R20 =2;
Break;
 Case 27 :
 Case 29 :
 Case 31 :
 Case 33 :
R20 =3;
Break;
 Case 6 :
 Case 26 :
 Case 35 :
 Case 46 :
R20 =4;
Break;
 Case 5 :
 Case 25 :
 Case 36 :
 Case 45 :
R20 =5;
Break;
EndSwitch;
## 夹具开
##R20=0，开 4、5；R20=1，开 4；R20=2，开 5；
##R20=3，开 6、7；R20=4，开 6；R20=5，开 7；
If R20==0
Set Out[4],ON;
Set Out[5],ON;
Delay T[0.5];
EndIf;
```

```
If R20==1
Set Out[4],ON;
Delay T[0.5];
EndIf;
If R20==2
Set Out[5],ON;
Delay T[0.5];
EndIf;
If R20==3
Set Out[6],ON;
Set Out[7],ON;
R5 =1;
Delay T[0.5];
EndIf;
If R20==4
Set Out[6],ON;
Delay T[0.5];
EndIf;
If R20==5
Set Out[7],ON;
Delay T[0.5];
R5 =1;
EndIf;
## 退出托盘
 Switch R1
 Case 5 :
 Case 15 :
 Case 25 :
 Case 35 :
 Case 45 :
 Case 55 :
GetCurPoint(2,2,D2);
D2 =D2+400;
SetPValue(P[53],2,D2);
## 单包第一包退出托盘
MovFromPut Pallet[0],P[53],0,0,350,
V[100],PickV[60];
Break;
Default:
## 退出托盘                                                    ## 退出跺型点位位置
MovFromPut Pallet[0],P[4],0,0,150,
V[100],PickV[60],Out(4,ON,T[1]),Out(5,ON,T[1]);
## 回过渡点
Movl P[4],V[100],Z[5];
R5 =0;
Break;
EndSwitch;
GetPalletRunNo(Pallet[0],R1);
## R10=0（不抓料，程序内部自加一）；R10=1(抓料，程序内部自加一)；R10=2(抓料，程序加二)
R10 =2;
```

```
 Switch R1
 Case 1 :
 Case 5 :
 Case 7 :
 Case 15 :
 Case 17 :
 Case 25 :
 Case 27 :
 Case 35 :
 Case 37 :
 Case 45 :
 Case 47 :
 Case 55 :
 Case 57 :
R10 =1;
Break;
 Case 6 :
 Case 16 :
 Case 26 :
 Case 36 :
 Case 46 :
 Case 56 :
R10 =0;
Break;
EndSwitch;
Incr B3;
EndWhile;
## 回安全点
Movj P[55],V[100],Z[1];
Goto L[0];
END;
```

4.4 轨迹控制应用

机器人用于轨迹控制的现场主要有：打磨、点胶、检测等，这些应用现场有一个共同点：都是由工具（打磨头、点胶阀、相机）与待加工工件或待检测工件构成。就工具与工件的相对运动状态，可将轨迹控制应用分类如下：（1）工具运动，工件固定；（2）工具固定，工件运动；（3）工具运动，工件运动。

我司 SCARA 机器人轨迹控制的应用现场主要有前两种，分别为：（1）工具运动，工件固定：某款全面屏手机听筒处 U 型槽侧棱异形点胶设备，采用我司 400 臂长 SCARA 机器人带动 VERMES 点胶阀进行轨迹运动并实现点胶动作；（2）工具固定，工件运动：某全面屏手机 CG 盖板 Frame 中框瑕疵检测设备，采用我司 400 臂长 SCARA 机器人吸取手机 CG 盖板进行轨迹运动并执行 Frame 中框瑕疵检测。接下来将该应用现场的工艺要求、实现方式、调试难点及解决方式进行阐述。

4.4.1 案例一：工具运动、工件固定的应用

1 概述

某款全面屏手机 U 型槽侧棱异形点胶设备，是我司 SCARA 机器人轨迹控制应用现场中工具运动，工件固定的典型现场。该设备为某款全面屏手机 CG 盖板上部安装 Camera、听筒、距离传感器处的 U 型槽侧棱进行喷胶以防止屏幕漏光的设备，具体实现方式：

- 采用双俯视相机拍摄 CG 屏幕的两个对角，通过机器人端的视觉标定计算出实际 CG 屏幕的中心位置及其相对于基坐标系的偏转角度，并基于此值建立动态用户坐标系；
- VERMES 胶阀安装于机器人丝杆末端，基于步骤 1 建立的动态用户坐标系，先通过测高传感器对全面屏手机 U 型槽内的 CG 盖板高度进行测量，PLC 将测高数值发送至机器人端，机器人对数据进行取平均值处理；
- 机器人运动到擦胶点，对喷嘴上的预留胶进行手动擦胶；
- 根据步骤 2 得到的高度平均值，对喷涂轨迹位置进行整体补偿偏移，同时机器人丝杆带动 VERMES 胶阀对屏幕 U 型槽侧棱进行非接触式喷胶；
- 为保证在喷涂热熔胶的过程中 VERMES 胶阀保持匀速运动状态并避免出现堆胶现象，采用分段式喷涂：实际喷涂位置的点位向前或向后偏移并结合运动 IO 进行控制——将 U 型槽分为 5 段处理，其中有 3 段直线、2 段圆弧。

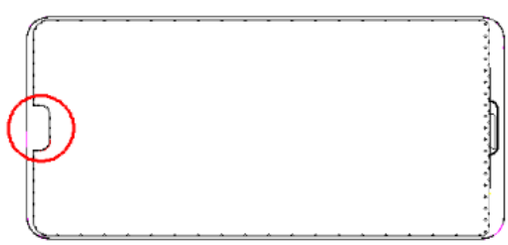


图 4-8 全面屏 U 型槽示意图

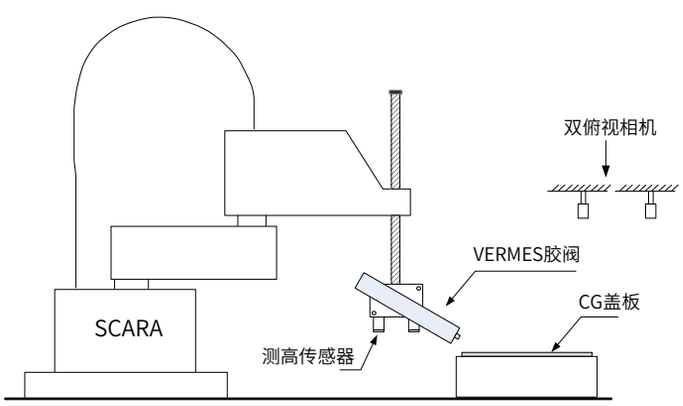


图 4-9 机器人点胶示意图

2 步骤说明

本方案中机器人的轨迹控制先通过双俯视相机拍摄 LCM 显示模组的两个对角点，然后基于此两角点的坐标数据计算得出全面屏中心点位置坐标及偏转角度，最终建立动态用户坐标系，并在屏幕相对“固定”的 U 型槽处进行分段式喷胶。

机器人分段式喷涂方式操作步骤：

根据全面屏的 CAD 图形，选择工件容易被相机识别的 2 个对角得到对角线并依此线得到矩形中心作为动态用户坐标系的原点，并平行于工件的 2 个相邻的边分别作为该动态用户坐标系的 X、Y 轴。

1） 建立动态用户坐标系：

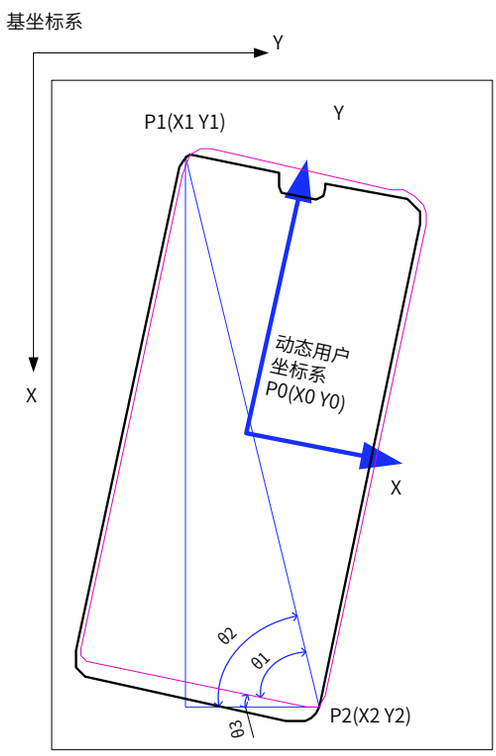


图 4-10 动态用户坐标系建立原理图

由于将 LCM 显示模组放置在工作台上的位置及角度不固定，所以只能通过建立动态用户坐标系，并基于此坐标系，对工件上各特征点进行确定。其中，P1 及 P2 点为视觉角点检测所检测到的点，并将其发送至机器人，通过计算可算出工件中心的 P0 点的坐标，并由 P1、P2 点在基坐标系下的横、纵坐标可计算出工件相对于基坐标系的偏转角度。为取点标准化，再将偏转后的坐标系旋转 90°，其中垂直于工件短边的坐标轴为动态用户坐标系的 X 轴，垂直于工件长边的坐标轴为动态用户坐标系的 Y 轴。

2） 动态用户坐标系建立：

P0 点的坐标：X0=(X1+X2)/2、Y0=(Y1+Y2)/2；

在基坐标下工件上两对角 P1、P2 点连线与 Y 轴的夹角：

θ2=ATan[(X2-X1) /(Y2-Y1)]；

且工件对角线与其短边的夹角为：θ1；

所以工件相对于基坐标的偏转角度：θ3=θ2-θ1；

动态用户坐标系相对于基坐标系的偏转角度：θ4=θ3+90°；

并可由此建立点位 P120=（X0，Y0，0，θ4），另定义点位 P102（用于补偿用户坐标系的 X、Y、Z 值），通过指令：

PR10 =Msft(P[102],P[120])——计算两点间的偏移量，并赋值给 PR10；

SetUserParm(1,PR10)——利用平移量 PR10 建立动态用户坐标系。

3） U 型槽特征点选取：

通过 LCM 显示模组的 CAD 图纸，确定 12 个特征点位：a. 第一条短直线——P4、P5，b. 第一段圆弧——P30、P6、P31，c. 第二条长直线——P7、P8，d. 第二段圆弧——P32、P9、P33，e. 第三条短直线——P10、P11。

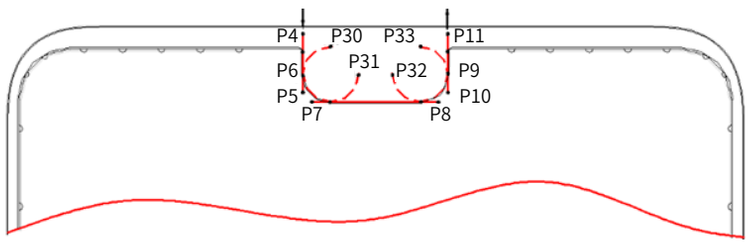


图 4-11 分段式喷涂示意图

3 数据配置

■ 输入输出信号

In(3)	机器人系统启动	1：启动
In(4)	机器人系统暂停	1：暂停
In(5)	机器人系统停止	1：停止
In(6)	机器人系统复位	1：复位

Out(5)	机器人系统报警	1：报警
Out(6)	VERMES 胶阀喷胶	0：关闭 1：开启

4 点位数据

以下坐标点预先输入了坐标点数据。

P0	等待点（原点）
P4	第 1 段直线起点
P5	第 1 段直线终点
P30	第 1 段圆弧起点
P6	第 1 段圆弧中间点
P31	第 1 段圆弧终点
P7	第 2 段直线起点
P8	第 2 段直线终点
P32	第 2 段圆弧起点
P9	第 2 段圆弧中间点
P33	第 2 段圆弧终点
P10	第 2 段直线起点
P11	第 2 段直线终点
P200	安全过渡点
P888	预喷胶位置
P889	手动擦胶位置

5 工艺参数调试

为保证全面屏 U 型槽侧棱所喷涂热熔胶的胶厚及胶宽满足要求，机器人的运动速度需要配合 VERMES 胶阀进行修改。热熔胶的喷涂要求及胶厚、胶宽数据参数主要为：

1） 胶厚最大不大 0.2mm；

2） 胶宽没有明确数据指标，大约为 0.5—0.7mm；

3） 热熔胶不能飞溅到 CG 盖板或背光上；

4） 对屏幕进行点亮测试，保证 U 型槽侧棱内部不漏光。

6 程序实例

```
P[0] =  -125.000000,  70.000000,-2541.000000,  -30.000000,   0.000000,   0.000000; 1, 0, 0, 0; 1, 1, 0;
P[4] =   -6.650000,  74.570000,   0.350000, -180.000000,   0.000000,   0.000000; 1, 0, 0, 0; 4, 0, 1;
P[5] =   -6.650000,  65.120000,   0.350000, -180.000000,   0.000000,   0.000000; 1, 0, 0, 0; 4, 0, 1;
P[6] =   -6.650000,  67.120000,   0.500000, -180.000000,   0.000000,   0.000000; 1, 0, 0, 0; 4, 0, 1;
P[7] =   -6.100000,  64.570000,   0.200000,  -90.000000,   0.000000,   0.000000; 1, 0, 0, 0; 4, 0, 1;
P[8] =    6.100000,  64.570000,   0.200000,  -90.000000,   0.000000,   0.000000; 1, 0, 0, 0; 4, 0, 1;
P[9] =    4.100000,  64.570000,   0.345000,   0.000000,   0.000000,   0.000000; 1, 0, 0, 0; 4, 0, 1;
P[10] =    6.650000,  63.120000,   0.330000,   0.000000,   0.000000,   0.000000; 1, 0, 0, 0; 4, 0, 1;
P[11] =    6.650000,  71.570000,   0.350000,   0.000000,   0.000000,   0.000000; 1, 0, 0, 0; 4, 0, 1;
P[30] =   -4.100000,  69.670000,   0.600000, -270.000000,   0.000000,   0.000000; 1, 0, 0,-1; 4, 0, 1;
P[31] =   -1.550000,  67.120000,   0.400000,   0.000000,   0.000000,   0.000000; 1, 0, 0, 0; 4, 0, 1;
P[32] =    1.550000,  67.120000,   0.305000, -180.000000,   0.000000,   0.000000; 1, 0, 0, 0; 4, 0, 1;
P[33] =    4.100000,  69.670000,   0.418000,  90.000000,   0.000000,   0.000000; 1, 0, 0, 0; 4, 0, 1;
P[100] =  1416.281000, 1100.784000,   0.000000,   0.000000,   0.000000,   0.000000; 1, 0, 0, 0; 5, 0, 1;
P[101] =  1512.065000,  154.001000,   0.000000,   0.000000,   0.000000,   0.000000; 1, 0, 0, 0; 5, 0, 0;
P[102] =    0.250000,   0.530000,  140.880000,   0.000000,   0.000000,   0.000000; 1, 0, 0, 0; 2, 0, 0;
P[110] =   146.337694, -213.742244,   0.000000,  89.492533,   0.000000,   0.000000; 1, 0, 0, 0; 3, 0, 1;
P[111] =   285.634610, -151.404449,   0.000000,  -89.932238,   0.000000,   0.000000; 1, 0, 0, 0; 3, 0, 0;
P[120] =   215.986152, -182.573347,   0.000000,  89.109347,   0.000000,   0.000000; 1, 0, 0, 0; 2, 0, 0;
P[200] =   -28.690118, -327.284849,  -97.000000,  -85.000000,   0.000000,   0.000000; 1, 0, 0, 0; 2, 0, 0;
P[888] =   -6.650000,  95.000000,   0.000000, -180.000000,   0.000000,   0.000000; 1, 0, 0, 0; 4, 0, 1;
P[889] =   -6.650000,  95.000000,  60.000000, -180.000000,   0.000000,   0.000000; 1, 0, 0, 0; 4, 0, 1;
P[890] =   31.027000,  90.513000,   8.780000, -180.000000,   0.000000,   0.000000; 1, 0, 0, 0; 4, 0, 1;
P[891] =   41.027000,  90.513000,   8.780000, -180.000000,   0.000000,   0.000000; 1, 0, 0, 0; 4, 0, 1;
START;
R0 =0;                                                  ## 初始化
R1 =0;                                                  ##R0 为 PLC 接收测高传感器的触发信号
R10 =0;                                                 ##R1 为 ROBOT 接收到 PLC 测高传感器数据的反馈信号
R11 =0;                                                 ##R10 为 PLC 接收到测高传感器的标志信号
R12 =0;                                                 ##R11 为 ROBOT 接收到 PLC 测高传感器的高度数据
D30 =0;                                                 ##R12 为 ROBOT 程序执行完毕的信号
SetModBusReg (49200,R0,1);
SetModBusReg (49201,R1,1);
SetModBusReg (49210,R10,1);
SetModBusReg (49211,R11,1);
SetModBusReg (49202,R12,1);
TimeStart(0);                                           ## 定义字符串
String STR1=”TB”;
String STR2=”TA”;
String cam1;
String cam2;
String cam3=”NG;”;
B0 =0;
B0 =1;
P[200] =(0,0,0,0,0,0),(0,0,0,0),(2,0,0);                ## 回安全位置
GetCurPoint(2,0,P[200]);
SetPValue(P[200],2,-113);
Movl P[200],V[10],Z[0];
SetPValue(P[200],3,-85);
Movl P[200],V[10],Z[0];
Jump P[0],V[60],Z[1],LH[0],MH[-113],RH[0];
WaitInPos;
L[0]:                                                   ##ROBOT 与视觉进行通讯
Open Socket(“192.168.1.99”,2000,2014,B0) ;
If B0==0
Goto L[0];
EndIf;
L[1]:                                                   ##ROBOT 接收工件左上角的点位数据
SetPortBuf(STR1) ;
Send Port[2014];
Get Port[2014],T[5],Goto L[1];
```

```
cam1 = GetPortbuf(1,100);
R0 = Strcmp(cam1,cam3);
If R0==0
B0 =B0+1;
If B0>=6
Alarm[0];
EndIf;
Goto L[1];
EndIf;
B0 = StrGetData(cam1,”,”,D0);                    ##ROBOT 转换工件左上角的点位坐标
P[100] =(D1,D2,0,0,0,0),(1,0,0,0),(5,0,1);
Cnvrt(P[100],P[110],Tool[0]);
Cnvrt(P[100],P[110],World);
WaitInPos;
L[2]:                                            ##ROBOT 接收工件右下角的点位数据
SetPortBuf(STR2) ;
Send Port[2014];
Get Port[2014],T[5],Goto L[2];
cam2 = GetPortbuf(1,100);
R2 = Strcmp(cam2,cam3);
If R2==0
B1 =B1+1;
If B1>=6
Alarm[1];
EndIf;
Goto L[2];
EndIf;
B1 = StrGetData(cam2,”,”,D4);                    ##ROBOT 转换工件右下角的点位坐标
P[101] =(D5,D6,0,0,0,0),(1,0,0,0),(5,0,0);
Cnvrt(P[101],P[111],Tool[0]);
Cnvrt(P[101],P[111],World);
WaitInPos;
GetPValue(P[110],0,D10);                         ## 通过转换后的点位坐标进行计算工件中心点坐标
GetPValue(P[110],1,D11);
GetPValue(P[111],0,D14);
GetPValue(P[111],1,D15);
D20 =(D10+D14)/2;
D21 =(D11+D15)/2;
P[120] =(0,0,0,0,0,0),(1,0,0,0),(2,0,0);
D30 =D10-D14;
D31 =D11-D15;
D32 =ATan(D30/D31);
D22 =65-D32+90;
SetPValue(P[120],0,D20);                         ## 建立动态用户坐标系
SetPValue(P[120],1,D21);
SetPValue(P[120],3,D22);
PR10 =Msft(P[102],P[120]);                       ##PI02 为建立动态用户坐标系的基准点，可在监控
SetUserParm(1,PR10);                               中修改 X、Y、Z 的值对用户坐标系进行偏移
Movl P[890],V[60],Z[1],Tool[1],Acc[35];          ##ROBOT 到达工件上第一测高点
WaitInPos;
R0 =1;
Delay T[0.5];
SetModBusReg (49200,R0,1);                       ##ROBOT 发送给 PLC 第一测高点数据触发信号 R0
Delay T[0.2];
L[20]:
GetModBusReg (49210,R10,1);                      ##PLC 已接收到第一测高点数据反馈信号 R10
If R10==1
Goto L[10];
Else
Goto L[20];
EndIf;
L[10]:
Movl P[891],V[60],Z[1],Tool[1],Acc[35];            ##ROBOT 到达工件上第二测高点
WaitInPos;
```

```
R0 =2;
Delay T[0.2];
SetModBusReg (49200,R0,1);                          ##ROBOT 发送第二侧高点数据触发信号 R0
Delay T[0.5];
L[21]:
GetModBusReg (49210,R10,1);                         ##PLC 已接收到第二测高点数据反馈信号 R10
If R10==2
Goto L[11];
Else
Goto L[21];
EndIf;
L[11]:
GetModBusReg (49211,R11,2);                         ##ROBOT 接收到 PLC 测高传感器的高度数据 R11
If R11<>0
Goto L[12];
Else
Goto L[11];
EndIf;
L[12]:
D30 =R11/1000;                                      ## 将测高传感器的高度数据 R11 转换为实际高度 D30
D31 =-0.079-D30;                                    ## 标准高度 -0.079，此工件 Z 轴的补偿量为 D31
Print D31;
R1 =1;                                              ##ROBOT 接收到 PLC 测高传感器的高度数据反馈 R1
SetModBusReg (49201,R1,1);
Movl P[888],V[60],Z[1],Tool[1],Acc[35];             ## 到达预喷点，并喷射 0.6 秒
WaitInPos;
Set Out[6],ON;
Delay T[0.6];
Set Out[6],OFF;
Movl P[889],V[30],Z[1],Tool[1],Acc[49];             ## 到达擦胶点，手动擦胶
Pause;
PR0 = (0,0,D31,0,0,0);                              ## 喷涂轨迹在 Z 向补偿 D31
                                                    ## 喷射第一段短直线
Movl Offset(P[4],PR0),V[60],Z[1],Tool[1],Acc[25];
WaitInPos;
Delay T[0.5];                                       ## 喷射第一段圆弧
Movl Offset(P[5],PR0),V[10],Z[1],Tool[1],Acc[25],Out(6,ON,S[5.6]),Out(6,OFF,S[-1.6]);
Movl Offset(P[30],PR0),V[10],Z[1],Tool[1],Acc[15];
Delay T[0.1];
Movc Offset(P[6],PR0),V[12],Z[1],Tool[1],Acc[80];
Movc Offset(P[31],PR0),V[12],Z[1],Tool[1],Acc[60],Out(6,ON,D[33.3]),Out(6,OFF,D[66.6]);
                                                    ## 喷射第二段长直线
Movl Offset(P[7],PR0),V[10],Z[1],Tool[1],Acc[25];
Delay T[0.1];
Movl Offset(P[8],PR0),V[10],Z[1],Tool[1],Acc[25],Out(6,ON,S[1.8]),Out(6,OFF,S[-1.6]);
                                                    ## 喷射第二段圆弧
Movl Offset(P[32],PR0),V[10],Z[1],Tool[1],Acc[15];
Delay T[0.1];
Movc Offset(P[9],PR0),V[12],Z[1],Tool[1],Acc[50];
Movc Offset(P[33],PR0),V[12],Z[1],Tool[1],Acc[49],Out(6,ON,D[33.3]),Out(6,OFF,D[66.6]);
                                                    ## 喷射第三段短直线
Movl Offset(P[10],PR0),V[10],Z[1],Tool[1],Acc[15];
Delay T[0.1];
Movl Offset(P[11],PR0),V[10],Z[1],Tool[1],Acc[25],Out(6,ON,S[3.8]),Out(6,OFF,S[-2.2]);
Delay T[0.1];
                                                    ## 运动至初始位置
Jump P[0],V[60],Z[1],Tool[1],LH[15],MH[-110],RH[0];
TimeOut(0,D3);
Print D3;
R12 =1;
SetModBusReg (49202,R12,1);                         ## 发送 ROBOT 程序执行完毕信号
Pause;
END;
```

4.4.2 案例二：工具固定、工件运动的应用

1 概述

某全面屏手机 CG 盖板 Frame 中框瑕疵检测设备，是我司 SCARA 机器人轨迹控制应用现场中工具固定，工件运动的典型现场。该设备中工具为俯视、水平、仰视相机，工件为某全面屏手机的 CG 盖板 FRAME 边框。具体实现方式：

1） 我司 SCARA 机器人丝杆末端通过吸嘴在固定位置以相同的位姿吸取手机 CG 盖板；

2） 吸取 CG 盖板后机器人按照既定轨迹通过输出 IO 信号依次到位触发俯视、仰视相机以及水平相机拍照从而执行瑕疵检测；

3） 拍摄完毕后，将 CG 盖板放回初始取料位置；

4） 为保证 CG 盖板 Frame 边框拍摄完整，整个边框轮廓的拍照点位：长边——5 个，短边——3 个，圆弧——1 个，共 20 个点位；

5） 需要注意的是为防止水平相机拍照时受到俯视、仰视相机闪光灯的影响，所以需要先同时触发俯视、仰视相机拍照，延时 3ms 后再触发水平相机拍照。

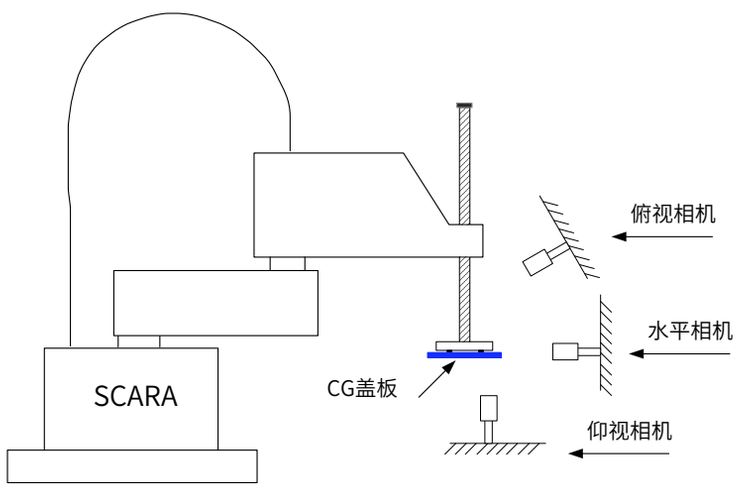


图 4-12 机器人工作示意图

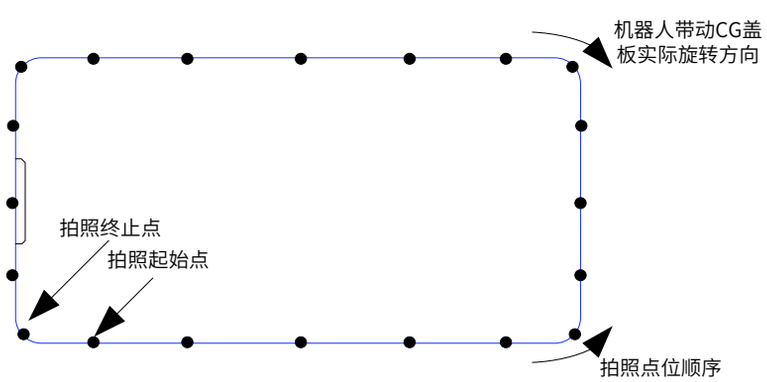


图 4-13 拍照顺序示意图

2 步骤说明

1） 本设备中的俯视、仰视及水平相机不可同时触发拍照，所以需要机器人到达“固定”拍照位置后依次触发 IO 进行拍照；

2） 整个 CG 盖板的拍照点位分布如图所示：

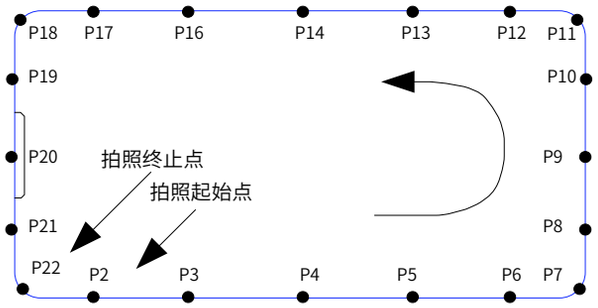


图 4-14 拍照点位分布图

3 数据配置

■ 输入输出信号

In(3)	机器人系统启动	1：启动
In(4)	机器人系统暂停	1：暂停
In(5)	机器人系统停止	1：停止
In(6)	机器人系统复位	1：复位
In(7)	真空信号	1：已成功吸取 CG 盖板

Out(4)	机器人系统报警	1：报警
Out(5)	机器人停止状态	1：机器人停止
Out(6)	机器人运行状态	1：机器人运行
Out(7)	吸真空	1：吸取 CG 盖板
Out(7)	俯视相机拍照	1：触发相机拍照
Out(7)	水平相机拍照	1：触发相机拍照
Out(7)	仰视相机拍照	1：触发相机拍照

■ 点位数据

以下坐标点预先输入了坐标点数据。

P0	等待点（原点）
P1	吸 / 放 CG 盖板点
P2—P6	CG 盖板第 1 段长直线 5 个拍照点
P7	CG 盖板第 1 段圆弧拍照点
P8—P10	CG 盖板第 1 段短直线 3 个拍照点
P11	CG 盖板第 2 段圆弧拍照点
P12—P16	CG 盖板第 2 段长直线 5 个拍照点
P17	CG 盖板第 3 段圆弧拍照点
P18—P20	CG 盖板第 2 段短直线 3 个拍照点
P21	CG 盖板第 4 段圆弧拍照点
P100	安全过渡点

4 程序实例

```
P[0] =  239.886000,  231.896000,  -90.631000, -175.330000,   0.000000,   0.000000;-1, 0, 1, 1; 2, 0, 0;
P[1] =  239.883000,  231.895000,  -96.455000, -175.327000,   0.000000,   0.000000;-1, 0, 1, 1; 2, 0, 0;
P[2] =  276.877699,   52.773929, -123.042115, -176.364622,   0.000000,   0.000000;-1, 0, 1, 1; 2, 0, 0;
P[3] =  276.877800,   25.642398, -123.042170, -176.364495,   0.000000,   0.000000;-1, 0, 1, 1; 2, 0, 0;
```

```
P[4] =  276.877663,  -0.041891, -123.042381, -176.365175,  0.000000,  0.000000;-1, 0, 1, 1; 2, 0, 0;
P[5] =  276.877687,  -28.571858, -123.042590, -176.364893,  0.000000,  0.000000;-1, 0, 1, 1; 2, 0, 0;
P[6] =  276.877582,  -55.317877, -123.042679, -176.364690,  0.000000,  0.000000;-1, 0, 1, 1; 2, 0, 0;
P[7] =  237.280666,  -22.006382, -123.043542,  134.934523,  0.000000,  0.000000;-1, 0, 0, 1; 2, 0, 0;
P[8] =  237.995441,  16.975918, -123.044273,  94.146790,  0.000000,  0.000000;-1, 0, 0, 0; 2, 0, 0;
P[9] =  237.995362,  -9.969706, -123.044557,  94.146242,  0.000000,  0.000000;-1, 0, 0, 0; 2, 0, 0;
P[10] =  235.149651,  23.509393, -123.044928,  53.308314,  0.000000,  0.000000;-1, 0, 0, 0; 2, 0, 0;
P[11] =  274.589279,  57.102940, -123.045290,  4.120154,  0.000000,  0.000000;-1, 0, 0, 0; 2, 0, 0;
P[12] =  274.589109,  31.296653, -123.045471,  4.120254,  0.000000,  0.000000;-1, 0, 0, 0; 2, 0, 0;
P[13] =  275.846577,  2.772763, -123.045829,  4.120196,  0.000000,  0.000000;-1, 0, 0, 0; 2, 0, 0;
P[14] =  275.846705,  -25.614312, -123.046540,  4.121321,  0.000000,  0.000000;-1, 0, 0, 0; 2, 0, 0;
P[15] =  277.647229,  -50.181132, -123.047068,  4.121344,  0.000000,  0.000000;-1, 0, 0, 0; 2, 0, 0;
P[16] =  242.082088,  -24.111813, -123.048039,  -39.138010,  0.000000,  0.000000;-1, 0, 0, 0; 2, 0, 0;
P[17] =  243.289953,  18.320891, -123.048095,  -86.599471,  0.000000,  0.000000;-1, 0, 0, 0; 2, 0, 0;
P[18] =  243.289719,  -9.562559, -123.048554,  -86.599396,  0.000000,  0.000000;-1, 0, 0, 0; 2, 0, 0;
P[19] =  239.905829,  25.527216, -123.049330, -131.127216,  0.000000,  0.000000;-1, 0, 0, 0; 2, 0, 0;
P[20] =  239.906000,  231.903000, -103.000000, -175.389000,  0.000000,  0.000000;-1, 0, 0,-1; 2, 0, 0;
P[21] =  239.906000,  231.903000, -100.000000, -175.389000,  0.000000,  0.000000;-1, 0, 0,-1; 2, 0, 0;
P[100] =  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000; 0, 0, 0, 0; 0, 0, 0;
START;
TimeStart(0);
Set Out[10],OFF;                                        ## 输出 OUT 复位
Set Out[11],OFF;
Set Out[12],OFF;
P[100] =(0,0,0,0,0,0),(0,0,0,0),(2,0,0);                ## 回安全位置
GetCurPoint(2,0,P[100]);
GetPValue(P[100],1,D1);
If D1>=118                                              ## 当前位置判断
JumpL P[0],V[100],Z[0],Acc[100],LH[100],MH[-81],RH[100];
Else
SetPValue(P[100],0,237.6);
Movl P[100],V[30],Z[0],Acc[50];
JumpL P[0],V[100],Z[0],Acc[100],LH[100],MH[-81],RH[100];
EndIf;
TimeStart(1);
Movl P[1],V[100],Z[0],Acc[100];                         ## 抓取工件
WaitInPos;
Set Out[8],ON;
Velset 100;                                             ## 开始检测（P2-P19）
PR0 = (0,0,0,0,0,0);                                    ## 预留点位偏移量
                                                        ## 第 1 段长直线（P2-P6）
Jump Offset(P[2],PR0),V[30],Z[0],Acc[100],LH[50],MH[-95],RH[50];
WaitInPos;
Set Out[10],ON;
Set Out[12],ON;
Delay T[0.005];
Set Out[10],OFF;
Set Out[12],OFF;
Set Out[11],ON;
Movj Offset(P[3],PR0),V[30],Z[0],Acc[100],Out(11,OFF,T[0.005]);
WaitInPos;
Set Out[10],ON;
Set Out[12],ON;
Delay T[0.005];
Set Out[10],OFF;
Set Out[12],OFF;
```

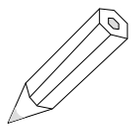
```
Set Out[11],ON;
Movj Offset(P[4],PR0),V[30],Z[0],Acc[100],Out(11,OFF,T[0.005]);
WaitInPos;
Set Out[10],ON;
Set Out[12],ON;
Delay T[0.005];
Set Out[10],OFF;
Set Out[12],OFF;
Set Out[11],ON;
Movj Offset(P[5],PR0),V[30],Z[0],Acc[100],Out(11,OFF,T[0.005]);
WaitInPos;
Set Out[10],ON;
Set Out[12],ON;
Delay T[0.005];
Set Out[10],OFF;
Set Out[12],OFF;
Set Out[11],ON;
Movj Offset(P[6],PR0),V[30],Z[0],Acc[100],Out(11,OFF,T[0.005]);
WaitInPos;
Set Out[10],ON;
Set Out[12],ON;
Delay T[0.005];
Set Out[10],OFF;
Set Out[12],OFF;
Set Out[11],ON;
                                                    ## 第 1 段圆弧（P7）
Movj Offset(P[7],PR0),V[30],Z[0],Acc[100],Out(11,OFF,T[0.005]);
WaitInPos;
Set Out[10],ON;
Set Out[12],ON;
Delay T[0.005];
Set Out[10],OFF;
Set Out[12],OFF;
Set Out[11],ON;
                                                    ## 第 1 段短边（P8-P10）
Movj Offset(P[8],PR0),V[30],Z[0],Acc[100],Out(11,OFF,T[0.005]);
WaitInPos;
Set Out[10],ON;
Set Out[12],ON;
Delay T[0.005];
Set Out[10],OFF;
Set Out[12],OFF;
Set Out[11],ON;
Movj Offset(P[9],PR0),V[30],Z[0],Acc[100],Out(11,OFF,T[0.005]);
WaitInPos;
Set Out[10],ON;
Set Out[12],ON;
Delay T[0.005];
Set Out[10],OFF;
Set Out[12],OFF;
Set Out[11],ON;
Movj Offset(P[10],PR0),V[30],Z[0],Acc[100],Out(11,OFF,T[0.005]);
WaitInPos;
Set Out[10],ON;
Set Out[12],ON;
Delay T[0.005];
```

```
Set Out[10],OFF;
Set Out[12],OFF;
Set Out[11],ON;
                                                    ## 第 2 段圆弧（P11）
Movj Offset(P[11],PR0),V[30],Z[0],Acc[100],Out(11,OFF,T[0.005]);
WaitInPos;
Set Out[10],ON;
Set Out[12],ON;
Delay T[0.005];
Set Out[10],OFF;
Set Out[12],OFF;
Set Out[11],ON;
                                                    ## 第 2 段长直线（P12-P16）
Movj Offset(P[12],PR0),V[30],Z[0],Acc[100],Out(11,OFF,T[0.005]);
WaitInPos;
Set Out[10],ON;
Set Out[12],ON;
Delay T[0.005];
Set Out[10],OFF;
Set Out[12],OFF;
Set Out[11],ON;
Movj Offset(P[13],PR0),V[30],Z[0],Acc[100],Out(11,OFF,T[0.005]);
WaitInPos;
Set Out[10],ON;
Set Out[12],ON;
Delay T[0.005];
Set Out[10],OFF;
Set Out[12],OFF;
Set Out[11],ON;
Movj Offset(P[14],PR0),V[30],Z[0],Acc[100],Out(11,OFF,T[0.005]);
WaitInPos;
Set Out[10],ON;
Set Out[12],ON;
Delay T[0.005];
Set Out[10],OFF;
Set Out[12],OFF;
Set Out[11],ON;
Movj Offset(P[15],PR0),V[30],Z[0],Acc[100],Out(11,OFF,T[0.005]);
WaitInPos;
Set Out[10],ON;
Set Out[12],ON;
Delay T[0.005];
Set Out[10],OFF;
Set Out[12],OFF;
Set Out[11],ON;
Movj Offset(P[16],PR0),V[30],Z[0],Acc[100],Out(11,OFF,T[0.005]);
WaitInPos;
Set Out[10],ON;
Set Out[12],ON;
Delay T[0.005];
Set Out[10],OFF;
Set Out[12],OFF;
Set Out[11],ON;
                                                    ## 第 3 段圆弧（P17）
Movj Offset(P[17],PR0),V[30],Z[0],Acc[100],Out(11,OFF,T[0.005]);
WaitInPos;
```

```
Set Out[10],ON;
Set Out[12],ON;
Delay T[0.005];
Set Out[10],OFF;
Set Out[12],OFF;
Set Out[11],ON;
                                                    ## 第 2 段段直线（P18-P20）
Movj Offset(P[18],PR0),V[30],Z[0],Acc[100],Out(11,OFF,T[0.005]);
WaitInPos;
Set Out[10],ON;
Set Out[12],ON;
Delay T[0.005];
Set Out[10],OFF;
Set Out[12],OFF;
Set Out[11],ON;
Movj Offset(P[19],PR0),V[30],Z[0],Acc[100],Out(11,OFF,T[0.005]);
WaitInPos;
Set Out[10],ON;
Set Out[12],ON;
Delay T[0.005];
Set Out[10],OFF;
Set Out[12],OFF;
Set Out[11],ON;
Movj Offset(P[20],PR0),V[30],Z[0],Acc[100],Out(11,OFF,T[0.005]);
WaitInPos;
Set Out[10],ON;
Set Out[12],ON;
Delay T[0.005];
Set Out[10],OFF;
Set Out[12],OFF;
Set Out[11],ON;
                                                    ## 第 4 段圆弧（P21）
Movj Offset(P[21],PR0),V[30],Z[0],Acc[100],Out(11,OFF,T[0.005]);
WaitInPos;
Set Out[10],ON;
Set Out[12],ON;
Delay T[0.005];
Set Out[10],OFF;
Set Out[12],OFF;
Set Out[11],ON;
                                                    ## 拍摄完毕，放回工件
Jump Offset(P[1],PR0) ,V[30],Z[0],Acc[50],Out(11,OFF,T[0.005]),LH[50],MH[-100],RH[50];
VelSet OFF;
WaitInPos;
Set Out[8],OFF;
TimeOut(1,D101);
Movl P[0],V[100],Z[0],Acc[100];                     ## 回到等待位
Print D101;
END;
```

Memo NO. _____

Date / /



指令篇

指令篇 - 目 录

第 1 章 机器人指令索引

符号类

No.	指令	功能	参考
1	==	关系等于	“2.1.1 关系运算符”
2	>	关系大于	
3	<	关系小于	
4	>=	关系大于或等于	
5	<=	关系小于或等于	
6	<>	关系不等于	
7	=	赋值运算符	“2.1.3 简单运算类”
8	+	加法运算符	
9	-	减法运算符	
10	*	乘法运算符	
11	/	除法运算符	
12	%	取余运算符	
13	##	注释	“2.1.4 特殊符号类”
14	;	分号 , 位于行末，代表一行语句的结束	
15	:	冒号，用于提示下文，标签 L 指令、Switch-Case-Default 等中有用到	
16	,	逗号，起间隔作用	
17	“ ”	双引号 , 表明该内容为字符串	
18	< 变量名 > = < 变量名 >	进行字符串运算	“2.5 字符串指令”

A

No.	指令	功能	参考
1	Abs(x)	取绝对值	
2	Alarm	在窗口消息栏显示用户自定义的报警信息	
3	AND	逻辑与	
4	AngleToRad(x)	角度转弧度	
5	ASin(x)	反正弦运算	
6	ATan(x)	反正切运算	
7	ArmChange	切换 SCARA 机器人左右手手臂	

B

No.	指令	功能	参考
1	Break	结束循环体	
2	BToAscii	将 B/LB 变量按 Ascii 码转换成对应字符	

C

No.	指令	功能	参考
1	Call	子程序调用	
2	Caps	字符串大写	
3	CheckLock	螺丝锁付结果检测	
4	CheckUnLock	松螺丝结果检测	
5	Clear	清除对象的值	
6	ClearCnvObject	传送带物体删除	
7	ClearLatchPos	锁存位置清除	
8	Close Com	关闭打开的串口，切断串口连接	
9	Close Socket	关闭打开的以太网端口，切断以太网连接	
10	Cnvrt	点的坐标系转换	
11	CnvVision	打开 / 关闭传送带的视觉端口	
12	Continue	结束本次循环	
13	CopyCnvObject	复制传送带上物体的视觉数据	
14	Cos(x)	余弦运算	

D

No.	指令	功能	参考
1	Decr	数值变量的自减	
2	Delay	程序延时	
3	Dist	求两个点之间的直线距离	
4	DToStr	将 D/LD 变量转化为字符串变量	

E

No.	指令	功能	参考
1	EOffsOff	关闭路径平移	
2	EOffsOn	开启整体路径平移	

F

No.	指令	功能	参考
1	For-EndFor	带执行次数的循环语句	

G

No.	指令	功能	参考
1	Get AD	读取单个模拟量输入信号	
2	Get IG	读取一组数字输入信号（默认一组含 8 个 In 信号）	
3	Get In	读取单个数字输入信号（In 信号）	
4	Get OG	读取一组数字输出信号（默认一组含 8 个 Out 信号）	
5	Get Out	读取单个数字输出信号（Out 信号）	
6	Get Port	从远端端口接收数据	
7	GetAlarmNo	获取报警号	
8	GetAPointFromFile	从系统中加载单个点到本程序的位置变量	
9	GetArmType	获取位置变量的臂参数	

No.	指令	功能	参考
10	GetAt	取字符串中的某位字符	
11	GetCnvObject	接收传送带上物体的视觉数据	
12	GetCommVal	读取共享内存公共区的值	
13	GetCoordNo	获取位置变量的坐标系号	
14	GetCurPoint	获取当前关节或基坐标系下的坐标值	
15	GetLatchPos	锁存位置读取	
16	GetModBusCoil	获取 ModBus 从站线圈值	
17	GetModBusReg	读取 ModBus 从站寄存器值	
18	GetPalletRunNo	查看码垛或者拆垛运行点号	
19	GetPlcVar	获取 PLC 传输的变量值	
20	GetPortbuf	获取接收缓冲区的字符串	
21	GetPrValue	获取平移变量的一项值	
22	GetPValue	获取位置变量的一项坐标值	
23	GetRunState	获取系统运行状态	
24	GetStrAscii	将字符串转换成 Ascii 码	
25	GetToolNo	获取位置变量的工具号	
26	Group	配置输入、输出组信号	

H

No.	指令	功能	
1	Halt	暂停选定的任务	
2	HexToStr	将 16 进制数转换成字符串	
3	Home	回工作原点	

I

No.	指令	功能	
1	If-Else-EndIf	条件判断	
2	Incr	数值变量的自增	
3	Invert	反转输出信号	
4	IsPalletFinished	查看码垛或者拆垛是否完成	

J

No.	指令	功能	
1	Jump	跳跃指令	
2	JumpL	直线跳跃指令	

L

No.	指令	功能	
1	L-Goto	逻辑跳转	
2	LatchEnable	位置锁存功能开启 / 关闭	
3	Left	取字符串左边特定字符	
4	Load Obj	加载工件动力学参数	
5	LoadPointFromFile	从点文件中加载点到系统中	
6	LoadScrewParm	加载螺丝工艺参数至伺服	
7	LockScrew	螺丝锁付	
8	LowCase	字符串小写	

No.	指令	功能	
9	LPallet	设定局部托盘变量	
10	LPallet4	4 点设定局部托盘变量	

M

No.	指令	功能	
1	Max(x,y)	求 x 和 y 中的最较大值	
2	Movc	圆弧插补	
3	MovFromGet	拆垛返回指令，拆垛放置完成后，从托盘上货物的放置点返回	
4	MovFromPut	码垛返回指令，码垛放置完成后，从托盘上货物的放置点返回	
5	Movj	关节插补	
6	Movl	直线插补	
7	MovToGet	拆垛指令，运行至托盘上货物的放置点	
8	MovToPut	码垛指令，运行至托盘上货物的放置点	
9	Mid	取字符串的中间特定字符	
10	Min(x,y)	求 x 和 y 中的最较小值	
11	Msft	计算两个位置变量间的平移变量	

O

No.	指令	功能	
1	OffSetUserParm	平移用户坐标系	
2	Open Socket	指定服务器 IP 地址及端口号，连接到服务器	
3	Open Com	指定串口号和波特率，连接到指定串口	
4	OR	逻辑或	

P

No.	指令	功能	
1	P[***]	位置变量的直接赋值	
2	PR***=	平移变量的直接赋值	
3	P=Offset	点偏移	
4	P=Pallet	取托盘上的点	
5	Pause	暂停运行	
6	Pow(x,y)	计算 x 的 y 次幂	
7	PR Sum	两个平移变量加减运算	
8	Print	在窗口消息栏打印变量或者字符串	
9	PToStr	点数据转换成字符串	
10	Pulse	输出 Out 为脉冲信号	

S

No.	指令	功能	
1	SavePointFile	保存点文件	
2	Send Port	发送数据到远端端口	
3	Set DA	设置模拟量输出信号	
4	Set OG	设置一组数字输出信号	
5	Set Out	设置单个数字输出信号	

No.	指令	功能	
6	SetAccRamp	设置运动段的加加速度	
7	SetArmType	设置位置变量的臂参数	
8	SetCoordParm	设置位置变量的坐标系类参数	
9	SetCommVal	设置公共区指定地址的值	
10	StrGetData	从字符串中取出数据	
11	SetModBusCoil	设置 ModBus 从站线圈值	
12	SetModBusReg	设置 ModBus 从站寄存器的值	
13	SetPalletRunNo	设置码垛或者拆垛运行点号	
14	SetObjCog	设置工件的质心	
15	SetObjInertia	设置工件的转动惯量	
16	SetObjMass	设置工件的质量	
17	SetObjOrient	设置工件的姿态	
17	SetPortBuf	设置发送缓冲区的值	
18	SetPrValue	设置平移变量的一项值	
19	SetPValue	设置位置变量的一项坐标值	
20	SetToolCog	设置工具的质心	
21	SetToolInertia	设置工具的转动惯量	
22	SetToolMass	设置工具的质量	
23	SetToolOrient	设置工具的姿态	
24	SetToolParm	动态设置工具坐标系参数	
25	SetUserParm	动态设置用户坐标系参数	
26	Sin(x)	正弦运算	
27	SlewMode	设置旋转优化方式	
28	SPrintf	以指定形式格式化字符串	
29	Sqrt(x)	开平方运算	
30	Strcmp	比较两个字符串大小	
31	StrFind	查找的指定字符串出现的起始索引	
32	StrFindEnd	在字符串中找到指定字符串最后出现的位置	
33	Strlen	取字符串的长度	
34	StrToD	将字符串转换为双精度数据，赋给 D/LD 变量	
35	StrToHex	将字符串转成 16 进制	
36	StrToR	将字符串转换为整型数据，并赋给 R/LR 变量	
37	StrRePlace	替换字符串	
38	StrReverse	反转字符串	
39	Switch-Case-Default-EndSwitch	条件选择语句	

Q

No.	指令	功能	
1	Quit	停止并退出选定的任务	

R

No.	指令	功能	
1	RadToAngle(x)	弧度转角度	
2	RefSys	切换坐标系	
3	ResetPallet	在使用其它码垛指令前，必须先初始化托盘。	
4	Resume	继续启动（恢复）选定的任务	
5	Ret	返回主程序	
6	Right	取字符串的右边特定字符	
7	RToStr	将 R/LR 变量转化为字符串变量	

T

No.	指令	功能	
1	Tan(x)	正切运算	
2	TrimLeft	去掉字符串中左边的空格	
3	TimeOut	输出计时器时长到指定变量中	
4	TrimRight	去掉字符串中右边的空格	
5	TimeStart	启动定时器并开始计时	

U

No.	指令	功能	
1	UnLoad Obj	卸载工件动力学参数	
2	UnLockScrew	松螺丝	
3	USING MAIN	声明调用主程序的位置变量	

V

No.	指令	功能	
1	Velset	速度设置	

W

No.	指令	功能	
1	Wait	等待直至检测到输入输出信号满足条件	
2	While-EndWhile	条件循环	
3	WriteAPointToFile	向点文件中写入点	
4	WZBoxDef	定义矩形干涉区	
5	WZCylDef	定义圆柱干涉区	
6	WZDisable	禁用干涉区	
7	WZDOSet	定义干涉后的动作并激活干涉区	
8	WZEnable	激活干涉区	
9	WZLimJDef	定义关节干涉区	
10	WZSphDef	定义球形干涉区	

X

No.	指令	功能	
1	Xqt	启动主任务或指令型 PLC 任务，实现多任务程序共同运行	

第 2 章 运算指令

2.1 运算符

2.1.1 关系运算符

指令	功能
==	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

2.1.2 逻辑类

指令	功能
AND	逻辑与
OR	逻辑或

2.1.3 简单运算类

指令	功能
=	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

2.1.4 特殊符号类

指令	功能
##	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 等中用到
,	逗号，起间隔作用
“ ”	双引号 , 表明该内容为字符串

2.2 Incr

■ **功能**

数值变量的自增。

■ **格式**

Incr var;

■ **参数**

var:B/R/LB/LR 变量

■ **范例**

```
B1=1;
Incr B1;                                    ##B1 自增 1，值变为 2
```

2.3 Decr

■ **功能**

数值变量的自减。

■ **格式**

Decr var;

■ **参数**

var:B/R/LB/LR 变量

■ **范例**

```
B1=1;
Derc B1;                                    ##B1 自减 1，值变为 2
```

2.4 数值运算

■ **功能**

数值变量的赋值。

■ **格式**

<Var> = <Exp>;

■ **参数**

<Exp>：由变量、数字、运算符组成的表达式。如 LD3= (Sin(30)+D1)/2.1。

■ **返回值**

<Var>：B/R/D/LB/LR/LD 变量。

■ **范例**

```
R7=17;
LD3=( Sin(30)+D1)/2.1;
```

2.4.1 Sin

■ **功能**

正弦运算。

■ **描述**

求正弦运算，以角度为单位。

■ **格式**

<Var>= Sin(<Exp>);

■ **参数**

<Exp>: 运算表达式，以角度为单位

■ **返回值**

<Var>：B/R/D/LB/LR/LD 变量

■ **范例**

D1=Sin(60+D2);

2.4.2 Cos

■ **功能**

余弦运算。

■ **描述**

求余弦运算，以角度为单位。

■ **格式**

<Var>= Cos(<Exp>);

■ **参数**

<Exp>: 运算表达式

■ **返回值**

<Var>：B/R/D/LB/LR/LD 变量

■ **范例**

D1=Cos(60+D2);

2.4.3 Tan

■ **功能**

正切运算。

■ **描述**

求正切运算，以角度为单位。

■ **格式**

<Var>= Tan(<Exp>);

■ **参数**

<Exp>: 运算表达式

■ **返回值**

<Var>：B/R/D/LB/LR/LD 变量

■ **范例**

D1= Tan (60+D2);

2.4.4 Asin

■ **功能**

反正弦运算。

■ **描述**

求反正弦运算。

■ **格式**

<Var>= Asin(<Exp>);

■ **参数**

<Exp>: 运算表达式

■ **返回值**

<Var>：B/R/D/LB/LR/LD 变量，以角度为单位。

■ **范例**

D1= Asin (60+D2);

2.4.5 Acos

■ **功能**

反余弦运算。

■ **描述**

求反余弦运算。

■ **格式**

<Var>= Acos(<Exp>);

■ **参数**

<Exp>: 运算表达式

■ **返回值**

<Var>：B/R/D/LB/LR/LD 变量，以角度为单位。

■ **范例**

D1= Acos(60+D2);

2.4.6 Atan

■ **功能**

反正切运算。

■ **描述**

求反正切运算。

■ **格式**

<Var>= Atan (<Exp>);

■ **参数**

<Exp>: 运算表达式

■ **返回值**

<Var>：B/R/D/LB/LR/LD 变量，以角度为单位。

■ **范例**

D1= Atan (60+D2);

2.4.7 Sqrt

■ **功能**

开平方运算。

■ **描述**

求开平方运算。

■ **格式**

<Var>= Sqrt (<Exp>);

■ **参数**

<Exp>: 运算表达式

■ **返回值**

<Var>：B/R/D/LB/LR/LD 变量

■ **范例**

D1= Sqrt (9); *##D1=3*

2.4.8 Pow

■ **功能**

x 的 y 次方。

■ **描述**

计算 x 的 y 次方。

■ **格式**

<Var>= Pow(<Exp_x>, <Exp_y>) ;

■ **参数**

<Exp_x>: 做底数的运算表达式

<Exp_y>: 做幂的运算表达式

■ **返回值**

<Var>：B/R/D/LB/LR/LD 变量

■ **范例**

D1= Pow(2,3); *##D1=8*

2.4.9 Abs

■ **功能**

绝对值。

■ **描述**

取绝对值。

■ **格式**

<Var>= Abs (<Exp>);

■ **参数**

<Exp>: 运算表达式

■ **返回值**

<Var>：B/R/D/LB/LR/LD 变量

■ **范例**

```
D1= Abs (-2);                    ##D1=2
```

2.4.10 Max

■ **功能**

最大值比较。

■ **描述**

取两个数值的最大值。

■ **格式**

<Var>= Max (<Exp_x>, <Exp_y>);

■ **参数**

<Exp_x>、<Exp_y>：作比较的运算表达式

■ **返回值**

<Var>：B/R/D/LB/LR/LD 变量

■ **范例**

```
B1=1;
R1=2;
D1= Max(B1,R1);                  ##D1=2
```

2.4.11 Min

■ **功能**

最小值比较。

■ **描述**

取两个数值的最小值。

■ **格式**

<Var>= Min (<Exp_x>, <Exp_y>);

■ **参数**

<Exp_x>、<Exp_y>：作比较的运算表达式

■ **返回值**

<Var>：B/R/D/LB/LR/LD 变量

■ **范例**

```
B1=1;
R1=2;
```

```
D1= Max(B1,R1);                    ##D1=1
```

2.4.12 AngleToRad

■ **功能**

角度转弧度。

■ **描述**

角度转弧度。

■ **格式**

<Var>= AngleToRad (<Exp_x>, <Exp_y>);

■ **参数**

<Exp>: 运算表达式

■ **返回值**

<Var>：B/R/D/LB/LR/LD 变量

■ **范例**

```
D1= AngleToRad (90);
```

2.4.14 RadToAngle

■ **功能**

弧度转角度。

■ **描述**

弧度转角度。

■ **格式**

<Var>= RadToAngle (<Exp_x>, <Exp_y>);

■ **参数**

<Exp>: 运算表达式

■ **返回值**

<Var>：B/R/D/LB/LR/LD 变量

■ **范例**

```
D1= AngleToRad (3.14);
```

2.5 字符串指令

进行字符串运算。

■ **格式**

<StrVar> = <StrExp>;

■ **参数**

< StrVar >：字符串变量，全局字符串或局部字符串

■ **返回值**

< StrExp >：由一个或多个字符表达式组成。如 Str2=Str1+ Left(Str1,2)。

2.5.1 Left

■ **功能**

字符串取左。

■ **描述**

取字符串左边几个字符，如下格式所示，截取右边 n 个字符。

■ **格式**

StrDest= Left(StrSource, n);

■ **参数**

StrSource: 源字符串对象。

n: 为取左字符串的数量。

■ **返回值**

StrDest: 截取后的字符串

■ **范例**

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

2.5.2 Right

■ **功能**

字符串取右。

■ **描述**

取字符串的右边几个字符，如下格式所示，截取右边 n 个字符。

■ **格式**

StrDest= Right(StrSource, n);

■ **参数**

StrSource: 源字符串对象。

n: 为取右字符串的数量。

■ **返回值**

StrDest: 截取后的字符串

说明：n 为取右字符串的数量。

■ **范例**

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

2.5.3 Mid

■ **功能**

字符串取中间。

■ **描述**

取字符串的中间一段字符，如下格式所示，截取从 i 开始 n 个字符。

■ **格式**

StrDest = Mid(StrSource,i,n);

■ **参数**

StrSource: 源字符串对象。

i: 起始索引。

n: 截取数量。

■ **返回值**

StrDest: 截取后的字符串

■ **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”
```

2.5.4 GetAt

■ **功能**

字符串字符获取。

■ **描述**

取字符串中的某一位字符，如下格式所示，截取索引为 i 的字符。

■ **格式**

StrDest = GetAt(StrSource, i);

■ **参数**

StrSource: 源字符串对象。

i: 要截取字符的索引。

■ **返回值**

StrDest: 截取后的字符串

■ **范例**

```
Str1 = “abcdefg”;
Str2 = GetAt(Str1, 3);                  ## 取 Str1 序号为 3 的字符，结果为 Str2 = “d”
```

2.5.5 StrFindEnd

■ **功能**

反向查找。

■ **描述**

在字符串中找到指定字符串最后出现的位置。寻找成功返回其索引号，失败返回 -1。

■ **格式**

Index = StrFindEnd (StrSource ,StrItem)；

■ **参数**

StrSource：源字符串对象。

StrItem：需要在源字符串中查找的字符串对象

■ **返回值**

Index: 查询到的索引。失败返回 -1。

■ **范例**

```
LB1=StrFindEnd(“abcaba”,”a”);                ## 结果为 LB1=5
```

2.5.6 StrRePlace

■ **功能**

字符串替换。

■ **描述**

在字符串中以新替换字符串 Str1 为 Str2。

■ **格式**

StrDest = StrRePlace (StrSource,OldSubStr, NewSubStr);

■ **参数**

StrSource：源字符串

OldSubStr：需要替换的原子字符串

NewSubStr：用来替换的新子字符串

■ **返回值**

StrDest: 替换后的字符串

■ **范例**

```
String Str1=”aoe”;
Str1= StrRePlace (Str1,”a”,”b”);             ## 结果为 Str1 = “boe”
```

2.5.7 StrReverse

■ **功能**

字符串反转。

■ **描述**

反转字符串，颠倒字符串中字符的顺序。

■ **格式**

StrDest = StrReverse(StrSource)；

■ **参数**

StrSource：源字符串

■ **返回值**

StrDest: 反转后的字符串

■ **范例**

```
String Str1=”aoe”;
```

```
Str1= StrReverse (Str1);                    ## 结果为 Str1 = “eoa”
```

2.5.8 Strlen

■ **功能**

字符串取长。

■ **描述**

取字符串的长度。

■ **格式**

length = Strlen(StrSource);

■ **参数**

StrSource：源字符串

■ **返回值**

length：字符串长度

■ **范例**

```
Str1 = “abcd”;
LB1 = Strlen(Str1);                         ## 结果 LB1 = 4
```

2.5.9 StrFind

■ **功能**

字符串查找。

■ **描述**

在字符串中查找的指定字符串出现的起始索引。

■ **格式**

Index = StrlFind(StrSource,SubStr);

■ **参数**

StrSource：源字符串。

SubStr：要查询的子字符串

■ **返回值**

Index：查询结果，指定字符串的起始索引。当找不到时，返回结果为 -1。

■ **范例**

```
Str1 = “abcd”;
LR1 = Strlen(Str1,”cd”);                    ## 结果 LR1 = 2
```

2.5.10 TrimLeft

■ **功能**

字符串左修剪。

■ **描述**

去掉字符串中左边的空格。

■ **格式**

StrDest = TrimLeft(StrSource);

■ **参数**

StrSource：源字符串。

■ **返回值：**

StrDest：修剪后的字符串

■ **范例**

```
Str1 = “ abc”;
Str1 = TrimLeft(Str1);                    ## 结果为 Str1 = “abc”
```

2.5.11 TrimRight

■ **功能**

字符串右修剪。

■ **描述**

去掉字符串中右边的空格。

■ **格式**

StrDest = TrimRight(StrSource);

■ **参数**

StrSource：源字符串。

■ **返回值**

StrDest：修剪后的字符串。

■ **范例**

```
Str1 = “abc ”;
Str1 = TrimLeft(Str1);                    ## 结果为 Str1 = “abc”
```

2.5.12 Strcmp

■ **功能**

字符串比较。

■ **描述**

比较两个字符串大小。将两个字符串自左向右进行逐个字符相比（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作为返回值，若全部相同则返回 0。

■ **格式**

value = Strcmp (Str1 , Str2);

■ **参数**

Str1：被比较的字符串。

Str2：比较的字符串

■ **返回值**

value：比较的结果。值为第一个不同的两个字符的 ASCII 码之差，若全部相同则值为 0。

■ **范例**

```
Str1=”ABxC”;
Str=”ABzH”;
R1=Strcmp(Str1,Str2);                    ##R1=-2
```

2.5.13 StrGetData

■ **功能**

字符串数据获取。

■ **描述**

从字符串中取出数据，存放在指定的变量中，并返回取出数据的个数。

■ **格式**

num = StrGetData(StrSource, Separator, Data);

■ **参数**

StrSource: 源字符串。当前只支持局部字符串和 Port，Port 代表接收缓冲区的字符串。

Separator：分割符，为一个字符串

Data: 分割后存储的数据对象，有以下三种形式。

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。

■ **返回值**

num: 取出的数据的个数，可选用 B/R/LB/LR 变量存储

注意事项：

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。

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**

```
## 范例 1：
Str1 = “123And45Andggg”;
B0= StrGetData(str1,”And”,R0);
```

说明：范例 1 将“123And45Andggg”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。

2.5.14 GetPortbuf

■ **功能**

获取接收缓冲区的字符串。

■ **描述**

从接收缓冲区指定位置开始读取指定数量的字符并存储到指定的字符串变量中。

■ **格式**

StrVar = GetPortbuf(StartBit , Num, Port);

■ **参数**

StartBit：数字 /B/LB/R/LR 变量（0~1023），接收缓冲区的起始读取位置。

Num：数字 /B/LB 变量（0~100），读取的字符个数。

Port：当使用客户端 - 服务器通信时，此 Port 为客户端端口号，应满足 1024-9999，其中 3333 不可使用，因为它为系统占用。当使用串口通信时，此 Port 为串口号，应满足 0-15。然而由于该指令可配对上述两种通信，因此利用示教器输入时，只限制输入范围 0-9999。

■ **返回值**

StrVar：保存从接收缓冲区读取的字符串。

说明：在接收缓冲区中从某一位 StartBit 开始取连续 Num 个字符，并传给指定的字符串变量。失败时传 0。

■ **范例**

```
## 从第 2 位开始，取 3 个字符
L[2]:
Get Port[1025],T[1],Goto L[2];
Str1 = GetPortbuf(2,3,1025);
```

2.6 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。

2.6.1 Caps

■ **功能**

字符串大写。

■ **描述**

取字符串的大写形式，并返回。

■ **格式**

StrDest = Caps(StrSource);

■ **参数**

StrSource：源字符串

■ **返回值**

StrDest: 转换后的字符串

注意事项：Caps 对于非英文字母不作处理。

■ **范例**

```
Str1 = “aB12”;
Str2 = Caps (Str1);                    ##Str2 = “AB12”
```

2.6.2 LowCase

■ **功能**

字符串小写。

■ **描述**

取字符串的小写形式，并返回。

■ **格式**

StrDest =LowCase(StrSource);

■ **参数**

StrSource：源字符串

■ **返回值**

StrDest: 转换后的字符串

注意事项：LowCase 对于非英文字母不作处理。

■ **范例**

```
Str1 = “aB12”;
Str2 = LowCase (Str1);                 ##Str 3= “ab12”
```

2.6.3 RToStr

■ **功能**

R 转 Str。

■ **描述**

将 R/LR 变量转化为字符串变量。

■ **格式**

StrValue = RToStr(R);

■ **参数**

R：待转换的 R/LR 变量

■ **范例**

```
R1 = 45;
Str4 = RToStr(R1);                     ##Str4 = “45”
```

2.6.4 DToStr

■ **功能**

D 转 Str。

■ **描述**

将 D/LD 变量转化为字符串变量。

■ **格式**

StrValue = DToStr(D,m,n);

■ **参数**

D: 待转换的 D/LD 变量。

m: 转换的整数位数。

n: 转换的小数位数。

■ **返回值**

StrValue：转换后的字符串变量

注意事项：

m 限定转换的整数位数。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 限定转换的小数位数。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
```

2.6.5 StrToR

■ **功能**

Str 转 R。

■ **描述**

将字符串转换为整型数据，并赋给 R/LR 变量。

■ **格式**

R = StrToR(StrSource);

■ **参数**

StrSource：要转换的源字符串

■ **返回值**

R：转换成的 R\LR 变量。

注意事项：StrToR 只识别前面的数字位（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
```

2.6.6 StrToD

■ **功能**

Str 转 D。

■ **描述**

将字符串转换为双精度数据，赋给 D/LD 变量

■ **格式**

D = StrToD(StrSource);

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
```

2.6.7 PToStr

■ **功能**

P 转 Str。

■ **描述**

点数据转换成字符串。

■ **格式**

StrValue = PToStr(P);

■ **参数**

P: 需要转换的点位

■ **返回值**

StrValue: 转换后的字符串

注意：前面 6 个坐标值转换后保留小数点后 6 位。

■ **范例**

```
P[0]=(10,0,0,0,0,0),(1,0,0,0),(1,0,0);
Str[1]=PToStr(P[0]);
##Str[1] 结果为 10.000000,0.000000,0.000000,0.000000,0.000000,0.000000;1,0,0,0; 1,0,0;
```

2.6.8 BToAscii

■ **功能**

B 转 Ascii 。

■ **描述**

将 B/LB 变量按 Ascii 码转换成对应字符

■ **格式**

StrValue = BToAscii(B);

■ **参数**

B:B/LB 变量，保存着待转换的整型的值

■ **返回值**

StrValue：转换后的字符串。

■ **范例**

```
String NewStr;
LB1=65;
NewStr = BToAscii(LB1);                    ## 结果为 NewStr=“A”
```

2.6.9 HexToStr

■ **功能**

十六进制转字符串。

■ **描述**

将存储在字符串中的数据转成 16 进制表示的字符串。

■ **格式**

StrToHex(StrSource,StrDest)；

■ **参数**

StrSource: 传入参数，需要转换的字符串

StrDest: 传出参数，用于转换后的字符串存储的变量

■ **返回值**

length：转换后字符串的长度

■ **范例**

```
GetPort 接收数据存储字符串 StrSource 中
      ! : q j 4 w
调用 StrToHex 转换存储 StrDest 中
21 3A 71 6A 34 77
```

2.6.10 StrToHex

■ **功能**

字符串转 16 进制。

■ **描述**

将 16 进制表示的字符串 StrSource 转换成该 16 进制数对应的字符组成的字符串存储在 StrDest 中。

■ **格式**

length = HexToStr(StrSource,StrDest)；

■ **参数**

StrSource: 传入参数 , 需要转换的字符串

StrDest: 传出参数，转换后的字符串存储的变量

■ **返回值**

length: 转换后字符串的长度

■ **范例**

定义字符串变量 StrSource = “213A716A3477”
调用 HexToStr 指令转换成字符串
“!:qj4w”存储于字符串 StrDest 中

2.6.11 GetStrAscii

■ **功能**

字符串转换 Ascii 码。

■ **描述**

字符串转换成 Ascii 码，存储于一段 B 变量中。

■ **格式**

RetNum = GetStrAscii(StrSource,num,B);

■ **参数**

StrSource：源字符串

num：需转换成 ASCII 码的字符个数。范围 0-255，且不大于字符串的长度。

B：B/LB 变量，数据存储在从该 B 变量开始的一系列连续 B 变量中。

■ **返回值**

RetNum: 转换成功的字符个数

注意：

使用时应保证不超过 B/LB 变量的个数范围。比如，下面会产生报错：

String str1 = “abc”；

LR1 = StrToAscii(str1,3,LB254); ##LB 个数范围 LB0-LB255，这里使用 LB254、LB255、LB256

■ **范例**

String str1 = “abc”;
LR1 = GetStrAscii (str1,3,LB1); *##LB1、LB2、LB3 分别为 97、98、99，LR1=3*

2.7 SPrintf

■ **功能**

格式化字符串。

■ **描述**

以指定形式格式化字符串，同 C 语言 sprintf(char *buffer, const char *format, [argument] ...)

■ **格式**

SPrintf(StrValue, format, argument);

■ **参数**

StrValue: 待格式化的字符串

Format：格式字符串。可为以下形式的组合：%[flag][width][.precision]type

参数	说明
Flag	-：有 - 表示左对齐输出，如省略表示右对齐输出。 0：有 0 表示指定空位填 0, 如省略表示指定空位不填。
width	格式化后总长度。如果数据的长度小于 width，则左端补以空格，若大于 width，则按实际位数输出。
precision	指定小数精度的位数。未指定时，默认小数位数为 6。
Type	d：有符号十进制整数 f：浮点数 (包括 float 和 doulbe) x(X)：十六进制整数 s：字符串

argument:（可选参数）变量列表，可以是任何类型的数据。根据 Format 参数，该函数会期望一系列附加参数，每个参数包含一个值，用于替换格式字符串中的格式说明符。

■ **范例**

```
String Str1;
R1=20;
D1=12.123;
Sprinf(Str1,”R1=%d And D1=%.2f!”,R1,D1);        ## 结果为 Str1 = “R1=20 And D1=12.12!”
```

2.8 GetCurPoint

■ **功能**

取当前点。

■ **描述**

获取当前关节或基坐标系下的坐标值

■ **格式**

格式 1：GetCurPoint(CoordType,Index,D);

格式 2：GetCurPoint(CoordType, Index,P);

■ **参数**

CoordType：只能取 1 或 2。

1 表示关节坐标系，当前取用关节值（J1,J2,J3,J4,J5,J6）。

2 表示基坐标系，当前取用基坐标值 (X,Y,Z,A,B,C)。（不含工具）

Index：只在格式 1 时有效，为坐标值下标索引。表示取 6 个坐标值中的一个。

D：D/LD 变量，保存位置变量中的某一项坐标值

P：保存位置变量

注意：

若最后一个参数为 D 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。

若最后一个参数为 P 时，数据存储到 P 变量中，此时下标号 0-5 均可以，无意义。

■ **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]
```

2.9 Cnvrt

■ **功能**

点转换。

■ **描述**

点的坐标系转换。

■ **格式**

格式 1：Cnvrt (FromP, ToP,type);

格式 2：Cnvrt (FromP, ToP,type, TrigP);

■ **参数**

FromP：待转换的点

ToP：转换后的点

Type：坐标类型，决定将 FromP 转换成何种坐标系点。有以下形式：

参数	说明
Joint	关节标系类类型。表示转换成关节坐标系点，点的工具号、用户号不变
World	坐标系类型。表示转换成基坐标系点 (相当于在基坐标系下取点，坐标值为本体法兰盘末端在基坐标系下的位姿值)，点的工具号、用户号不变
Tool[i]	工具坐标系类型。表示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 TCP 在基坐标系下的位姿值)，点的用户号不变
User[i]	用户坐标系类型。表示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 TCP 在用户坐标系下的位姿值)，点的工具号不变

TrigP：(可选参数) 随动相机的拍照点，当且仅当 FromP 坐标系号为 6 时有效，其他情况均报错。

注意：

使用该指令，要求 P[i] 的坐标系号取 1-6，不能为 7。

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]
```

2.10 P[***]=

■ **功能**

点赋值。

■ **描述**

位置变量的直接赋值。

■ **格式**

P=(X,Y,Z,A,B,C),(ArmType[0],ArmType[1],ArmType[2],ArmType[3]),(CoordNo,ToolNo,UserNo);

■ **参数**

(X,Y,Z,A,B,C)：当坐标系号取 1 时为关节值 (J1,J2,J3,J4,J5,J6)，坐标系号取 2、3、4 时为位姿值 (X,Y,Z,A,B,C)

(ArmType[0],ArmType[1],ArmType[2],ArmType[3])：机器人臂参数

CoordNo：坐标系号

ToolNo：工具号

UserNo：用户号

■ **范例**

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

2.11 GetPValue

■ **功能**

点值获取。

■ **描述**

获取位置变量的一项坐标值。

■ **格式**

GetPValue(P,Index,D);

■ **参数**

P：位置变量 P[***]，*** 可以为数字或 R/LR 变量

Index：位置变量的下标索引，可为数字 0-5 或用 R/LR 变量表达。

D：D/LD 变量，用于存储位置变量的某项坐标值

■ **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
```

2.12 SetPValue

■ **功能**

点值设置。

■ **描述**

设置位置变量的一项坐标值。

■ **格式**

SetPValue(P,Index,D);

■ **参数**

P：位置变量 P[***]，*** 可以为数字或 R/LR 变量

Index：位置变量的下标索引，可为数字 0-5 或用 R/LR 变量表达。

D：D/LD 变量，要设置的位置变量某项坐标值

■ **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);
```

2.13 GetPrValue

■ **功能**

Pr 值获取。

■ **描述**

获取平移变量的一项值。

■ **格式**

GetPrValue(PR,Index,D);

■ **参数**

PR：PR/LPR 平移变量。

Index：平移变量的下标索引，可为数字 0-5 或用 R/LR 变量表达。

D：D/LD 变量，用于存储平移变量某项的值

■ **范例**

```
PR0 = (10,20,30,0,0,0);
GetPrValue(PR0,2,D1);                    ## 取 PR0 中索引号为 2 的元素，结果存于 D1 中。本例结果 D1=30
```

2.14 SetPrValue

■ **功能**

设置平移变量的值。

■ **描述**

设置平移变量的一项值。

■ **格式**

SetPrValue(PR,Index,Value);

■ **参数**

PR：PR/LPR 平移变量。

Index：平移变量的下标索引，可为数字 0-5 或用 R/LR 变量表达。

Value：数值 /D/LD 变量，要设置的平移变量某项的值

■ **范例**

```
PR0 = (10,20,30,0,0,0);
SetPrValue(PR0,2,35);                    ## 设置 PR0 中索引号为 2 的元素值为 35
## 本例结果 PR0= (10,20,35,0,0,0);
```

2.15 SetCoordParm

■ **功能**

点坐标系参数修改。

■ **描述**

设置位置变量的坐标系类参数。

■ **格式**

SetCoordParm(P,CoordType|Tool|User);

■ **参数**

P：待更改的位置变量 P[***]

CoordType：（可选项）坐标系号可 1~7

Tool：（可选项）可为 Tool[0]~Tool[15]

User：（可选项）可为 User[0]~ User [15]

注意：

该指令会强行更改位置变量的坐标系参数，但坐标值、臂参数维持不变。

三个可选参数 CoordType、Tool、User 应至少存在一项，也可多项同时存在，表示同时修改。

当使用可选参数时，当指定的数不符合要求，如 CoordType 取为 8，Tool 取 Tool[16] 时。设置无效，保持 P 点的对应参数项的原有值。

■ **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);
```

2.16 GetCoordNo

■ **功能**

点的坐标系号获取。

■ **描述**

获取位置变量的坐标系号。

■ **格式**

GetCoordNo(P, B);

■ **参数**

P: 位置变量 P[***]。

B：B/LB 变量，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
```

2.17 GetToolNo

■ **功能**

点的工具号获取。

■ **描述**

获取位置变量的工具号。

■ **格式**

GetToolNo(P,B);

■ **参数**

P: 位置变量。

B：B/LB 变量，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] 的工具号，结果 B1=1
```

2.18 GetUserNo

■ **功能**

点的用户号获取。

■ **描述**

获取位置变量的用户坐标系号

■ **格式**

GetToolNo(P,B);

■ **参数**

P: 位置变量 P[***]。

B：B/LB 变量，存储用户坐标系号。

■ **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] 的用户号，结果 B1=5
```

2.19 SetArmType

■ **功能**

臂参数设置。

■ **描述**

设置位置变量的臂参数

■ **格式 1**

SetArmType(SourceP, ReferP);

■ **说明 1**

将位置变量的臂参数值设为与参考位置变量的臂参数相同。

■ **参数 1**

SourceP：源位置变量

ReferP：参考的位置变量

■ **范例 1**

```
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)
```

■ **格式 2**

SetArmType(SourceP, Index, value);

■ **说明 2**

直接设置臂参数

■ **参数 2**

SourceP：源位置变量

Index：臂参数索引，值为 0-3

Value：臂参数值

■ **范例 2**

```
P[1] = (10,50,30,40,50,60),(1,1,1,0),(4,1,1);
SetArmType(P[1],1,-1);                    ##P[1] 臂参数为 (1,-1,1,0)
```

2.20 GetArmType

■ **功能**

臂参数获取。

■ **描述**

获取位置变量的臂参数。

■ **格式**

GetArmType(P , Index, R);

■ **参数**

P： 要获取的位置变量

Index：臂参数索引，值为 0-3

R：R/LR 变量，存储臂参数值

■ **范例**

```
P[1] = (10,50,30,40,50,60),(1,1,1,0),(4,1,1);
GetArmType(P[1],3,R1);                    ## 结果 R1=0
```

2.21 SetToolParm

■ **功能**

工具坐标系设置。

■ **描述**

动态设置工具坐标系参数，可采用直接法或三点法 TCP，如下格式 1 和格式 2。一经设置后，在本程序中工具坐标系值会立即变更，直到使用 SetToolParm (ToolNo,OFF) 关闭变更，恢复原值。

■ **格式 1**

SetToolParm (ToolNo,PR);

■ **说明**

直接设置工具坐标系的值

■ **参数 1**

ToolNo：工具号，值范围 1-15

PR：用来存储要设置的工具坐标系参数值。

■ **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**

ToolNo：工具号，值范围 1-15

P[i],P[j],P[k]：三点法 TCP 计算工具坐标系所用的三个点。

■ **格式 3**

SetToolParm (ToolNo,OFF);

■ **说明**

还原原工具坐标系参数。还原为原设置的值。

■ **参数**

ToolNo：工具号，值范围 1-15

注意事项：

若使用格式 2，P[i],P[j],P[k] 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。

2.22 SetUserParm

■ **功能**

用户坐标系设置

■ **描述**

动态设置用户坐标系参数，可采用直接法或三点法、两点法，如下格式 1、2、3。一经设置后，在本程序中用户坐标系值会立即变更，直到使用 SetUserParm (UserNo,OFF) 关闭变更，恢复原值。

■ **格式 1**

SetUserParm (UserNo,PR);

■ **说明 1**

直接设置用户坐标系的值。

■ **参数 1**

UserNo：用户号，值范围 1-15

PR：用来存储要设置的工具坐标系参数值。

■ **格式 2**

SetUserParm (UserNo,P[i],P[j],P[k]);

说明：利用三点法计算用户坐标系并设置。

■ **参数 2**

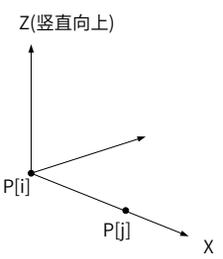
UserNo：用户号，值范围 1-15

P[i],P[j],P[k]：三点法计算用户坐标系所用的三个点。

■ **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 3**

用户号：可为数值 1-15 或 B/LB 变量。

P[i],P[j] 为位置变量。

■ **格式 4**

SetUserParm (UserNo,OFF);

■ **说明**

还原原用户坐标系参数。还原为原设置的值。

■ **参数**

UserNo：用户号，值范围 1-15

注意事项：

若用到 P[i],P[j],P[k]，则这些参数必须提前定义。

使用完后，利用 SetUserParm (ToolNo,OFF) 可还原。

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。

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。

2.23 OffSetUserParm

■ **功能**

用户坐标系平移。

■ **描述**

将用户坐标系进行整体平移。一经设置后，在本程序中用户坐标系值会立即变更，直到使用 OffSetUserParm (UserNo,OFF) 关闭变更，恢复原值。

■ **格式 1**

OffSetUserParm (UserNo,PR);

■ **说明 1**

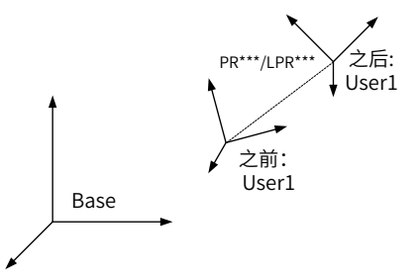
对用户坐标系平移，平移量存于 PR/LPR 变量中。

■ **参数 1**

UserNo：要平移的用户坐标系号

PR：在原用户坐标系基础上进行平移的平移变量

图示：



■ **格式 2**

OffSetUserParm (UserNo,OFF);

■ **说明 2**

关闭用户坐标系的平移 .

■ **参数**

UserNo：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：

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 OffSetUserParm (工具号 ,OFF) 可还原。

OffSetUserParm 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。

■ **范例**

```
PR1=(20,30,0,0,0,0);
OffSetUserParm(2,PR1);
```

2.24 Clear

■ **功能**

清除对象。

■ **描述**

清除对象的值。这个对象可以是数值变量、平移变量、工具坐标系、用户坐标系、手持工件负载、Out 信号。清除时，可以指定个数，实现批量清除，清除以选定对象开始的连续多个变量。特别的，Clear All 可以清除“所有”变量。清除的含义见什么是“清除”。

■ **格式 1**

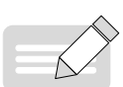
Clear object,Num;

■ **参数**

Object：可以是以下多种类型的对象：

参数	说明
Tool	表示清除工具坐标系参数，Tool[0]~Tool[15]
User	表示清除用户坐标系参数，User[0]~ User [15]
Obj	表示清除手持工件负载参数，Obj[0]~ Obj [15]
Out	表示清除 Out 变量，Out[0]~ Out [63]
Fin	表示清除对应 In 变量的状态为不强制状态，Fin[0]~Fin[63]
B	表示清除 B/LB 变量
R	表示清除 R/LR 变量
D	表示清除 D/LD 变量
PR	表示清除 PR/ LPR 变量

Num: 清除的数量。从 Object 起连续清除多个变量。


NOTE

◆ 什么是“清除”？

1. Tool/User/Obj 清除指的是还原系统参数中设置的变量值，并不是表示把对象的值清 0，在程序中 SetToolParm、SetUserParm、SetToolMass、SetToolCog、SetToolOrient、SetToolInettia、SetObjMass、SetObjCog、SetObjOrient、SetObjInertia 指令更改工具、用户、工件参数，可以通过 Clear 指令来还原其原来的值。
2. Out 清除是指将 Out 信号置为 OFF。
3. Fin 清除是将 In 信号置为不强制状态。
4. B/R/D 清除是将值置 0。
5. PR 清除是将 PR 变量里的每个参数置 0，即置为 (0,0,0,0,0,0)。

■ **范例**

Clear Tool[1],2;
Clear Out[4],3;
Clear PR2,4;

■ **格式 2**

Clear All;

■ **说明**

清除所有变量。包括上述所有的 Tool、User、Obj、B、R、D、PR 形式的变量。但不包括清除 Fin、Out。

2.25 Dist

■ **功能**

两点取距离。

■ **描述**

求两个点之间的直线距离。

■ **格式**

D = Dist(FromP, ToP);

■ **参数**

FromP: 起始点

ToP：终止点

■ **返回值**

D：D/LD 变量，用于存储结果，即两个点之间的距离。

注意：以在第一个位置变量的坐标系为参考，求两个点的直线距离

2.25 SetAccRamp

■ **功能**

设置运动段的加加速度。

■ **描述**

设置运动段的加加速度与减减速度。

■ **格式**

SetAccRamp(StartValue,EndValue);

■ **参数**

StartValue: 起始段的加加速度百分比。整型数据，范围（1-100）；

EndValue: 结束段的加加速度。整型数据，范围（1-100）。

注意：

该指令对 Movj、Movl、Movc、Jump、JumpL、ArmChange 的运动有效。

此指令一经设置，便一直生效，直至被再次设置为止。

程序文件重新打开时，加加速度和减减速度还原为默认值 100。

第 3 章 运动指令

3.1 Movj

■ **功能**

关节插补。

■ **描述**

用于将机器人从一个点快速的移动到另一个点，轨迹通常不在一条直线上。所有轴同时到达目的位置。（若希望机器人通过奇异位置，则只能使用 Movj 指令）

■ **格式**

Movj P, V, Z, [User], [Tool], [Acc], [NWait], [Until In == value], [Out(No,value,Type)…];

■ **参数**

P：要达到的目标点

常见形式：P[1],P[B1] 等。也可为偏移形式和托盘取点形式。

偏移形式——OffsetJ(P , PR)、OffsetT(P , PR)、Offset(P , PR)。

特别的，Offset 中可使用 PE，表示从当前位置偏移。

需要注意带偏移形式对其后的 [User], [Tool] 参数有一定要求。

更多见关于*“Offset”的使用*。

托盘取点形式——Pallet(PalletNo, Row, Colunm, Lay)、LPallet(PalletNo, Row, Colunm, Lay)。根据托盘号、行号、列号、层号这些信息取托盘上的点，详见 *Pallet 指令。*

V：指定速度。如 V[50] 代表 50% 最大速度。

Z：到位与过渡参数。有 Fine、Z[0]、Z[1]、Z[2]、Z[3]、Z[4]、Z[5] 七种选择。

参数	说明
Fine	代表精确到位。机器人进入并停留在设定的到位误差范围内时，被算作精确到位，然后再执行后面的运动。
Z[0]~Z[5]	机器人不在目标点停留，而是平滑的通过设定的过渡区域，需要说明的是即使 Z[0] 也不能保证机器人精确到达目标点。
Z[CP]	机器人不在目标点停留，以最大过渡长度过渡，平滑的通过设定的过渡区域。

更多见*过渡特性*。

User：（可选参数）选用的用户坐标系，取 User[0]~User[15]。

更多见关于 *User 参数扩展使用。*

Tool：（可选参数）选用的工具坐标系，取 Tool [0]~ Tool [15]。

更多见关于 *Tool 参数扩展使用*。

Acc：（可选参数）指定加速度。如 Acc[50] 代表 50% 最大加速度。

NWait：（可选参数）不等待标识。

使用此项，代表不等到运行到目标点，立即执行后面的非运动指令。

这会使得其后的非运动指令，如运算、信号、逻辑判断等等提前执行，直到遇到下一条运动指令。

■ **使用范例**

```
## 不等待运动到位，一开始运动立即置信号
```

Movj P[1],V[30],Z[0],NWait;
Set Out[3],ON;

Until In == value：（可选参数）运行中检测信号，满足条件则立即停止。

在运动过程中检测输入信号量，当条件满足，则当前行运动立即中止，继续执行后面行程序；若条件不满足，则运动直到结束点。

子参数：

In：输入信号

Value：输入信号值，ON 或 OFF

■ **使用范例：**

运动中检测 In[5]，当为 ON 时立即中止运动；否则一直运动直至结束。
Movj P[1],V[30],Z[0], Until In[5] == ON;

Out(No,value,Type)…：（可选参数）运行中并行输出信号。一条运动指令中最多并行输出 2 个信号。

No: 输出信号序号。

Value: 输出信号值。

Type: 运动中输出信号的类型。不同类型运动能使用不同的 Type。

Type	说明	适用的运动类型
T[n]	在运动中按时间输出信号 T[n] 代表时间，单位：秒，范围 -100.000~100.000。 n>=0 表示开始运动 n 秒后输出信号；n<0 表示到达目标点之前 n 秒时输出信号。 当超出运动范围外的时间被设定时，最多只在该段行程首末点触发，设定的时间无效。	Movj/ Movl/ Movc/Jump/ JumpL
D[n]	在运动中按路径百分比输出信号 D[n] 表示路径百分比。表示从开始运动到结束的 n% 路径时输出信号。	Movj/ Movl/ Movc/Jump/ JumpL
S[n]	在运动中按距离输出信号 S[n] 表示距离，单位：mm。范围 -65535~65535。 n>=0 表示从起点开始运动到 n 毫米之后时再输出信号；n<0 表示运动到距终点 n 毫米之前时输出信号。	Movl/ Movc/JumpL

注意：

1. 设定在过渡区域内的运动 IO，精度是不能保证的

2. 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。

■ **使用范例**

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

3.2 Offset

3.2.1 OffsetJ

■ **功能**

关节平移。

■ **典型使用形式**

P[2]=OffsetJ(P[1],PR1);

2） 对于 P=Offset(P,PR)，是直接在位置变量的坐标值上作平移。

3） 由于坐标值的涵义由坐标系指定，因此平移的目标点及参考坐标系由坐标系号指定：

当位置变量中的坐标系号为 1 时，表示法兰盘中心点在基坐标下平移。

当位置变量中的坐标系号为 2 时，表示法兰盘中心点在基坐标下平移。

当位置变量中的坐标系号为 3 时，表示 TCP 在基坐标下平移。工具由位置变量的工具号指定。

当位置变量中的坐标系号为 4 时，表示 TCP 在用户坐标下平移，工具由位置变量的工具号指定，用户坐标系由位置变量的用户号指定。

- 可以利用 Cnvrt 指令更改位置变量的在指定的坐标系下表示，再配合 P=Offset(P,PR)，就可以明确平移的目标点和参考系。我们推荐这样使用会更清晰！使用可参考示例 1。
- 对于 MovOffset(P,PR)……，目标点由运动指令中的 Tool 指定，参考系由运动指令中的 User 指定。且运动指令带有与 P 不同的 Tool 或 User，则先更换工具或用户坐标系再平移。使用可参考示例 2。
- 当 Mov 不带 Tool 参数时，目标点为机器人法兰盘末端；其它情况为 TCP；当 Mov 不带 User 参数时，参考系为基坐标系；其它情况为用户坐标系。

注意：

1） 使用 MovOffset(P,PR)……时，目标点和参考系由运动指令的附带参数 Tool、User 决定，与位置变量 P 无关！

2） 更换用户坐标系仍需满足条件“P 的坐标系号为 4”，此时，参考点由原用户坐标系下的点切换至指令 User 坐标系下的点，然后在 User 坐标系方向上平移。否则表示为直接在指令中 User 坐标系下平移。参考示例 3。

■ 示例

示例 1：

基坐标系下取点，法兰盘在基坐标系下移动。

```
P[1]=(10,20,30,40,50,60;1,0,0,0;2,1,0);## 坐标系号为 1 也可以
PR1=(10,0,0,0,0,0);
P[2]=Offset(P[1],PR1);
Movj P[2],V[30],Z[0];
```

由于 P[1] 在基坐标系下取点（坐标系号 2），因此是法兰盘中心相对基坐标的平移。注意此时的目标点就不是 TCP 了。

此时若需要 TCP 相对基坐标系的平移，则可以如下使用：

```
P[1]=(10,20,30,40,50,60;1,0,0,0;2,1,0);
Cnvrt(P[1],P[1],Tool[1]);## 将 P[1] 转换到工具坐标系下表达
P[2]=Offset(P[1],PR1);
Movj P[2],V[30],Z[0];
```

当然若 P[1] 是在工具坐标系下取点（坐标系号 3），可以不用 Cnvrt 转换。

再比如，由于某种特殊需求，需要 TCP 相对额外的一个用户坐标系 User1 平移，则可如下：

```
P[1]=(10,20,30,40,50,60;1,0,0,0;2,1,0);
Cnvrt(P[1],P[1],User[1]);## 将 P[1] 转换到 User1 下表达
P[2]=Offset(P[1],PR1);
Movj P[2],V[30],Z[0];
```

示例 2：

工具在基坐标系下平移示例。

```
P[1]=(10,20,30,40,50,60;1,0,0,0;2,1,0);
Movj Offset(P[1],PR1),V[30],Z[0], Tool[1];
```

此时，只要运动指令 Movj 带有 Tool 不带 User，就是工具在基坐标系下平移。位置变量 P 可以在任何坐标系下取。

示例 3：

MovjOffsetJ(P[1],PR1) ,V[30], Z[0];

■ **特征**

1） 此时 PR 的六个坐标值为关节旋转角度。内部会自动根据位置变量 P 计算关节值，然后进行平移。

2） 若与运动指令一起使用，形如 MovOffsetJ(P,PR)，则运动指令中不能带有 Tool、User。

■ **示例**

```
## 在 P[1] 基础上，机器人第一关节额外旋转 10°，第二关节额外旋转 20°。
## 方法一
PR1=(10,20,0,0,0,0);
MovjOffsetJ(P[1],PR1) ,V[30], Z[0];
## 方法二
PR1=(10,20,0,0,0,0);
MovjP[2],V[30],Z[0];
```

3.2.2 OffsetT

■ **功能**

沿当前工具位姿的平移。

■ **典型使用形式**

P[2]=OffsetT(P[1],PR1);

MovjOffsetT(P[1],PR1) ,V[30], Z[0],Tool[1];

■ **特征**

1） 在当前的工具坐标系基础上进行平移。PR 为移动和旋转 (X,Y,Z,A,B,C)，平移顺序：X>Y>Z>A>B>C

2） 若与运动指令一起使用，形如 MovOffsetT(P,PR)……，则运动指令中必带有 Tool，不带 User

■ **示例**

```
## 要到达的目标点的位置为，在 P[1] 基础上，沿着当前的工具坐标系运动，沿 X 方向移动 10mm，绕 Z 旋转 10°，再绕 Y 旋转 20°的位置，最后绕 X 旋转 30°的位置。
## 方法一
PR1=(10,0,0,10,20,30);
MovjOffsetT(P[1],PR1) ,V[30], Z[0],Tool[1];
## 方法二
PR1=(10,0,0,10,20,30);
P[2]=OffsetT(P[1],PR1);
MovjP[2],V[30],Z[0],Tool1;
```

3.2.3 Offset

■ **功能**

Offest 表示目标点在某个笛卡尔参考系下的平移。目标点可以是 TCP 也可以是法兰盘中心点，参考系可以是用户坐标系也可是基坐标系。

■ **典型使用形式**

P[2]=Offset (P[1],PR1);

MovjOffset(P[1],PR1) ,V[30], Z[0],Tool[1],User[1];

■ **特征**

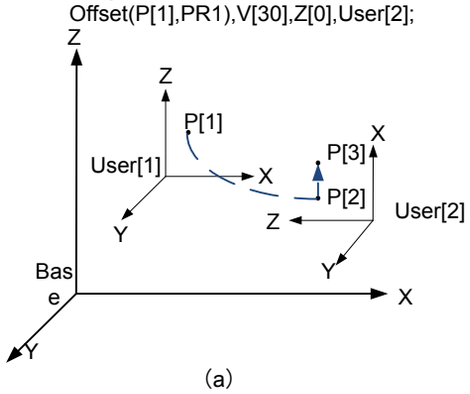
1） 目标点在笛卡尔参考系下的平移。此时 PR 为移动和旋转 (X,Y,Z,A,B,C)，平移顺序：X>Y>Z>A>B>C

先换用户坐标系再平移时，需要注意切换用户坐标系需满足条件！

当 P 不为用户坐标系下取的点时，由于不符合更换用户坐标系条件，平移起始点不发生变化，但是平移是沿着指令中 User 坐标系下的。

当 P 为用户坐标系下取的点时，先更换用户坐标系（平移起始点发生变化），再沿用户坐标系平移。

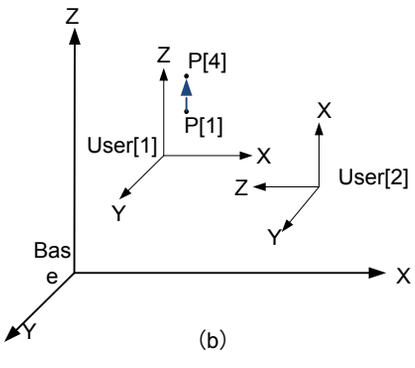
```
P[1]=(10,0,20,0,0,0),(1,1,1,0),(4,0,1);
PR1=(20,0,0,0,0,0);
Movj
Offset(P[1],PR1),V[30],Z[0],User[2];
```



（a）

图（a）：
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。如图1中，最终运动到位置P[3]

```
P[1]=(50,0,60,0,0,0),(1,1,1,0),(2,0,1);
PR1=(20,0,0,0,0,0);
Movj
Offset(P[1],PR1),V[30],Z[0],User[2];
```



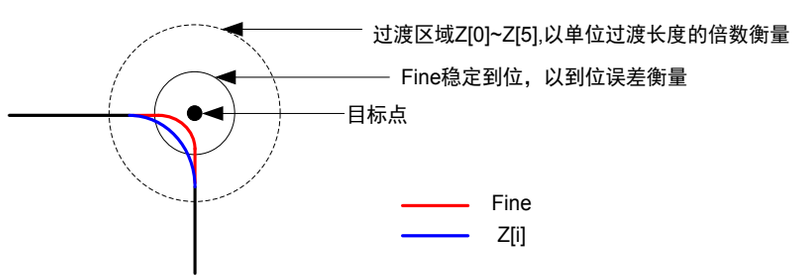
（b）

图（b）：
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。如图2中，最终运动到位置P[4]

■ 关于过渡特性

与 Fine 不同，Z[0]~Z[5] 表示机器人不在目标点停留，而是平滑的通过设定的过渡区域，需要说明的是即使 Z[0] 也不能保证机器人精确到达目标点。使用过渡能加快运动节拍。过渡范围是以目标点为圆心，以过渡长度为半径的区域。过渡长度等于过渡等级与过渡单位长度的乘积。

类型	方式
Fine	代表精确到位。机器人进入并停留在设定的到位误差范围内时，被算作精确到位，然后再执行后面的运动。
Z[0]	无过渡，但不能保证机器人精确到达目标点
Z[1]	有过渡，以 1 倍过渡单位长度
Z[2]	有过渡，以 2 倍过渡单位长度
Z[3]	有过渡，以 3 倍过渡单位长度
Z[4]	有过渡，以 4 倍过渡单位长度
Z[5]	有过渡，以 5 倍过渡单位长度
Z[CP]	有过渡，以最大过渡长度过渡（参见最大过渡长度）

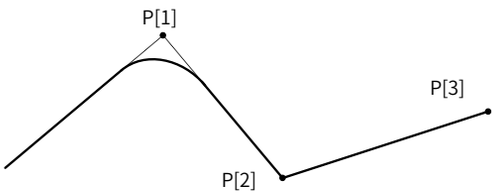


单位过渡长度分为关节过渡单位和线性过渡单位，MoveJ 和 Jump 指令使用关节单位（SCARA 机型 3 轴仍使

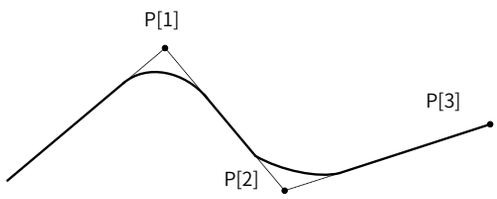
用线性单位），其他运动指令使用线性单位。

过渡的限制：在不使用 NWait 的情形下，只有相邻的两条运行指令间，才会产生过渡。也就是说如果两条运动指令间存在其它指令，比如 Set Out、B=1、Call 等等，则所指定的过渡 Z[0]~Z[5] 不生效，而是会精确到位。若仍然希望过渡，可使用 Nwait。

```
Movl P[1],V[30],Z[3];
Movl P[2],V[30],Z[3];
B1=1;
Movl P[3],V[30],Fine;
```

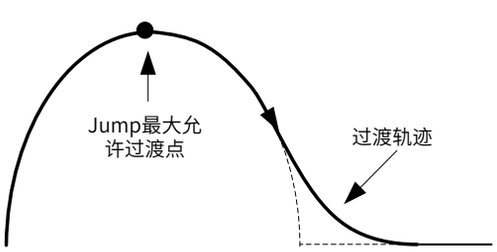



```
Movl P[1],V[30],Z[3];
P[2],V[30],Z[3],NWait;
B1=1;
Movl P[3],V[30],Fine;
```

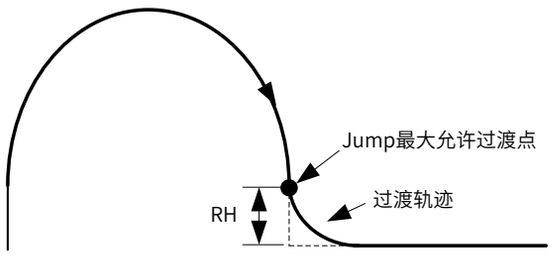


利用 L-Goto 跳转衔接的运动 ,Z[1]~Z[5] 不会生效，都将认为是 Z[0]。

最大过渡长度：对于具体的运动，允许的过渡长度是有限制的，当设定过渡长度超出允许的最大过渡长度时，取最大过渡长度作为实际的过渡长度。MovL、MoveJ、MovC 的最大允许过渡长度是总长度的一半。对于 Jump 和 JumpL 指令，当 RH(或 LH) 等于零时，最大允许过渡长度为总长度的一半，如图 1 所示；当 RH(或 LH) 不等于零时，最大允许过渡长度为 RH(或 LH)，如图所示。

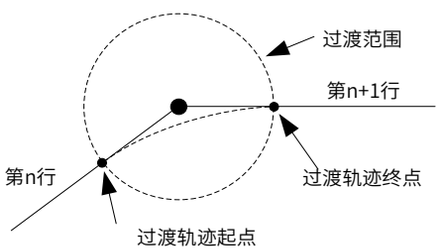


RH=0的Jump过渡



RH不等于0的Jump过渡

过渡对运动 IO 的影响：当存在下图中虚线所示的过渡区域时，设定在过渡区域内的运动 IO 生效精度是不能保证的，实际会在过渡轨迹起点和过渡轨迹终点之间不确定的位置生效。

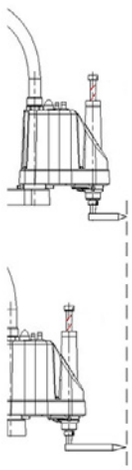


■ 关于 Tool 参数扩展使用

利用运动指令中 Tool 参数，可实现切换工具。使得在使用新工具时，新 TCP 到达原 TCP 的位置姿态。

应用场合：原来使用旧工具示教编写了一段加工程序；后来由于工具变换，使用新工具进行同样的加工，那么可不必逐点重新示教，直接在运动指令中指明新工具即可。

注意：换工具功能，只要求位置变量的工具号选择正确，位置变量可以在任意坐标系下取，即使是关节、基坐标系。



■ **案例**

首先使用 Tool1 示教取点，注意：位置变量的工具号为 1。编程如下：

Movj P[1],V[30],Z[0];		*Movj P[1],V[30],Z[0],Tool[1];*
Movl P[2],V[30],Z[3];	或	*Movl P[2],V[30],Z[3],Tool[1];*
Movl P[3],V[30],Z[3];		*Movl P[3],V[30],Z[3],Tool[1];*

后替换工具，使用 Tool2 进行加工，此时不必重新示教，直接更改指令中的 Tool：

MovjP[1],V[30],Z[0],Tool2;

MovlP[2],V[30],Z[3],Tool2;

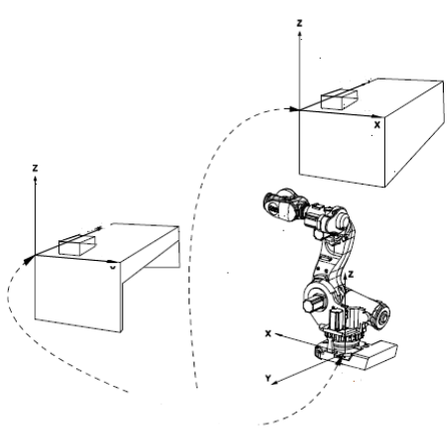
MovlP[3],V[30],Z[3],Tool2;

■ **关于 User 参数扩展使用：**

利用运动指令中 User 参数，实现切换用户坐标系。

应用场合：原来已示教编好程序。后来工件位置发生变化，不必逐点重新示教，直接在运动指令中指明新用户坐标系即可。

注意：重要的前提条件：需要原程序取点都在用户坐标系下。



■ **案例**

原来在工作台下的一批工件，在工作台上建立 User1，使用 User1 作为当前的用户坐标系。然后在用户坐标系下示教取点编程：

```
Movj P[1],V[30],Z[0];                Movj P[1],V[30],Z[0],User[1];
Movl P[2],V[30],Z[3];        或      Movl P[2],V[30],Z[3],User[1];
Movl P[3],V[30],Z[3];                Movl P[3],V[30],Z[3],User[1];
```

后来，工作台的摆放位置变了，在新工作台上建立 User2。那么可以直接更改：

```
MovjP[1],V[30],Z[0],User[2];
MovlP[2],V[30],Z[3],User[2];
MovlP[3],V[30],Z[3],User[2];
```

注意：P 的坐标系号为 1，不能切换 User。

3.3 Movl

■ **功能**

直线插补。

■ **描述**

用于线性的移动到给定目标位置。

■ **格式**

Movl P, V, Z, [User], [Tool], [Acc], [NWait], [Until In == value], [Out(No,value,Type)…];

■ **参数**

P：要达到的目标点

常见形式：P[1],P[B1] 等。也可为偏移形式和托盘取点形式。

偏移形式——OffsetJ(P , PR)、OffsetT(P , PR)、Offset(P , PR)。

特别的，Offset 中可使用 PE，表示从当前位置偏移。

需要注意带偏移形式对其后的 [User], [Tool] 参数有一定要求。

更多见关于*偏移“Offset”*的使用。

托盘取点形式——Pallet(PalletNo, Row, Colunm, Lay)、LPallet(PalletNo, Row, Colunm, Lay)。根据托盘号、行号、列号、层号这些信息取托盘上的点，*详见 2.6.7 节 Pallet 指令*。

V：指定速度。如 V[50] 代表 50% 最大速度。

Z：到位与过渡参数。有 Fine、Z[0]、Z[1]、Z[2]、Z[3]、Z[4]、Z[5] 七种选择。

参数	说明
Fine	代表精确到位。机器人进入并停留在设定的到位误差范围内时，被算作精确到位，然后再执行后面的运动。
Z[0]~Z[5]	机器人不在目标点停留，而是平滑的通过设定的过渡区域，需要说明的是即使 Z[0] 也不能保证机器人精确到达目标点。
Z[CP]	机器人不在目标点停留，以最大过渡长度过渡，平滑的通过设定的过渡区域。

更多见*过渡特性*。

User：（可选参数）选用的用户坐标系，取 User[0]~User[15]。

更多见关于 *User 参数扩展使用。*

Tool：（可选参数）选用的工具坐标系，取 Tool [0]~ Tool [15]。

更多见关于 *Tool 参数扩展使用*。

Acc：（可选参数）指定加速度。如 Acc[50] 代表 50% 最大加速度。

NWait：（可选参数）不等待标识。

使用此项，代表不等到运行到目标点，立即执行后面的非运动指令。

这会使得其后的非运动指令，如运算、信号、逻辑判断等等提前执行，直到遇到下一条运动指令。

■ **使用范例**

```
## 不等待运动到位，一开始运动立即置信号
Movj P[1],V[30],Z[0],NWait;
Set Out[3],ON;
```

Until In == value：（可选参数）运行中检测信号，满足条件则立即停止。

在运动过程中检测输入信号量，当条件满足，则当前行运动立即中止，继续执行后面行程序；若条件不满足，则运动直到结束点。

子参数：

In：输入信号

Value：输入信号值，ON 或 OFF

■ **使用范例：**

```
## 运动中检测 In[5]，当为 ON 时立即中止运动；否则一直运动直至结束。
Movj P[1],V[30],Z[0], Until In[5] == ON;
```

Out(No,value,Type)…：（可选参数）运行中并行输出信号。一条运动指令中最多并行输出 2 个信号。

No: 输出信号序号。

Value: 输出信号值。

Type: 运动中输出信号的类型。不同类型运动能使用不同的 Type。

Type	说明	适用的运动类型
T[n]	在运动中按时间输出信号 T[n] 代表时间，单位：秒，范围 -100.000~100.000。 n>=0 表示开始运动 n 秒后输出信号；n<0 表示到达目标点之前 n 秒时输出信号。 当超出运动范围外的时间被设定时，最多只在该段行程首末点触发，设定的时间无效。	Movj/ Movl/ Movc/Jump/ JumpL
D[n]	在运动中按路径百分比输出信号 D[n] 表示路径百分比。表示从开始运动到结束的 n% 路径时输出信号。	Movj/ Movl/ Movc/Jump/ JumpL
S[n]	在运动中按距离输出信号 S[n] 表示距离，单位：mm。范围 -65535~65535。 n>=0 表示从起点开始运动到 n 毫米之后时再输出信号；n<0 表示运动到距终点 n 毫米之前时输出信号。	Movl/ Movc/JumpL

注意：

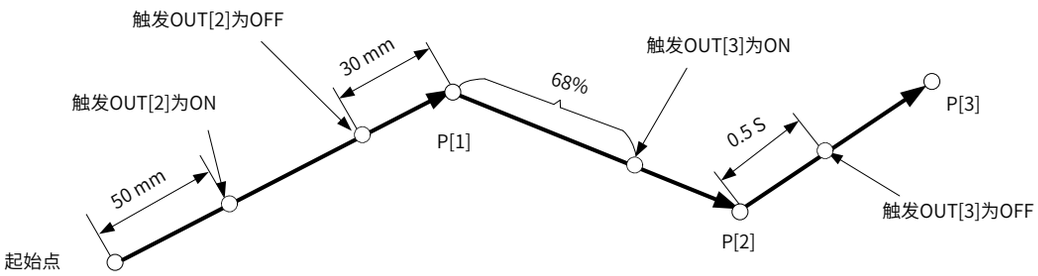
1. 设定在过渡区域内的运动 IO，精度是不能保证的

2. 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。

■ **使用范例：**

```
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;



3.4 Movc

■ **功能**

圆弧插补。

■ **描述**

按圆弧运动移动到给定目标位置。

■ **格式**

Movc P, V, Z, [User], [Tool], [Acc], [NWait], [Until In == value], [Out(No,value,Type)…

■ **参数**

P：要达到的目标点

常见形式：P[1],P[B1] 等。也可为偏移形式和托盘取点形式。

偏移形式——OffsetJ(P , PR)、OffsetT(P , PR)、Offset(P , PR)。

特别的，Offset 中可使用 PE，表示从当前位置偏移。

需要注意带偏移形式对其后的 [User], [Tool] 参数有一定要求。

更多见关于*偏移“Offset”*的使用。

托盘取点形式——Pallet(PalletNo, Row, Colunm, Lay)、LPallet(PalletNo, Row, Colunm, Lay)。根据托盘号、行号、列号、层号这些信息取托盘上的点，*详见 2.6.7 节 Pallet 指令*。

V：指定速度。如 V[50] 代表 50% 最大速度。

Z：到位与过渡参数。有 Fine、Z[0]、Z[1]、Z[2]、Z[3]、Z[4]、Z[5] 七种选择。

参数	说明
Fine	代表精确到位。机器人进入并停留在设定的到位误差范围内时，被算作精确到位，然后再执行后面的运动。
Z[0]~Z[5]	机器人不在目标点停留，而是平滑的通过设定的过渡区域，需要说明的是即使 Z[0] 也不能保证机器人精确到达目标点。
Z[CP]	机器人不在目标点停留，以最大过渡长度过渡，平滑的通过设定的过渡区域。

更多见*过渡特性*。

User：（可选参数）选用的用户坐标系，取 User[0]~User[15]。

更多见*关于 User 参数扩展使用*。

Tool：（可选参数）选用的工具坐标系，取 Tool [0]~ Tool [15]。

更多见*关于 Tool 参数扩展使用*。

Acc：（可选参数）指定加速度。如 Acc[50] 代表 50% 最大加速度。

NWait：（可选参数）不等待标识。

使用此项，代表不等到运行到目标点，立即执行后面的非运动指令。

这会使得其后的非运动指令，如运算、信号、逻辑判断等等提前执行，直到遇到下一条运动指令。

Out(No,value,Type)…：（可选参数）运行中并行输出信号。

注意：一条运动指令中最多并行输出 2 个信号。

子参数：

No: 输出信号序号。

Value: 输出信号值。

Type: 运动中输出信号的类型。不同类型运动能使用不同的 Type。

Type	说明	适用的运动类型
T[n]	在运动中按时间输出信号 T[n] 代表时间，单位：秒，范围 -100.000~100.000。 n>=0 表示开始运动 n 秒后输出信号；n<0 表示到达目标点之前 n 秒时输出信号。 当超出运动范围外的时间被设定时，最多只在该段行程首末点触发，设定的时间无效。	Movj/ Movl/ Movc/Jump/ JumpL
D[n]	在运动中按路径百分比输出信号 D[n] 表示路径百分比。表示从开始运动到结束的 n% 路径时输出信号。	Movj/ Movl/ Movc/Jump/ JumpL
S[n]	在运动中按距离输出信号 S[n] 表示距离，单位：mm。范围 -65535~65535。 n>=0 表示从起点开始运动到 n 毫米之后时再输出信号；n<0 表示运动到距终点 n 毫米之前时输出信号。	Movl/ Movc/JumpL

注意：

1. 设定在过渡区域内的运动 IO，精度是不能保证的

2. 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。

注意事项：

1. 三点确定一段圆弧，因此 Movc 指令必须是成对出现。中间不能间隔任何指令。如果两条连着的 Movc 前的上一条指令是 Movl 或 Movj 以外的指令，则会产生报错提示“圆弧运动前缺乏 Movj 或 Movl”。

2. 一段圆弧运动中两条 Movc 指令，除了 P 以外的所有参数，都以第一条为准。

■ 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;
```

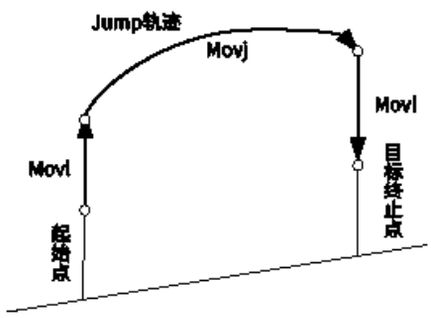
3.5 Jump

■ **功能**

跳跃指令。

■ **描述**

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。



■ **格式**

Jump P, V, Z, [User], [Tool], [Acc], [NWait], [Out(No,value,Type)…], LH, MH, RH;

■ **参数**

P：要达到的目标点

常见形式：P[1],P[B1] 等。也可为偏移形式和托盘取点形式。

偏移形式——OffsetJ(P , PR)、OffsetT(P , PR)、Offset(P , PR)。

特别的，Offset 中可使用 PE，表示从当前位置偏移。

需要注意带偏移形式对其后的 [User], [Tool] 参数有一定要求。

更多见关于*偏移“Offset”*的使用。

托盘取点形式——Pallet(PalletNo, Row, Colunm, Lay)、LPallet(PalletNo, Row, Colunm, Lay)。根据托盘号、行号、列号、层号这些信息取托盘上的点，详见 *2.6.7 节 Pallet 指令*。

V：指定速度。如 V[50] 代表 50% 最大速度。

Z：到位与过渡参数。有 Fine、Z[0]、Z[1]、Z[2]、Z[3]、Z[4]、Z[5] 七种选择。

参数	说明
Fine	代表精确到位。机器人进入并停留在设定的到位误差范围内时，被算作精确到位，然后再执行后面的运动。
Z[0]~Z[5]	机器人不在目标点停留，而是平滑的通过设定的过渡区域，需要说明的是即使 Z[0] 也不能保证机器人精确到达目标点。
Z[CP]	机器人不在目标点停留，以最大过渡长度过渡，平滑的通过设定的过渡区域。

更多见*过渡特性*。

User：（可选参数）选用的用户坐标系，取 User[0]~User[15]。

更多见*关于 User 参数扩展使用*。

Tool：（可选参数）选用的工具坐标系，取 Tool [0]~ Tool [15]。

更多见*关于 Tool 参数扩展使用*。

Acc：（可选参数）指定加速度。如 Acc[50] 代表 50% 最大加速度。

NWait：（可选参数）不等待标识。

使用此项，代表不等到运行到目标点，立即执行后面的非运动指令。

这会使得其后的非运动指令，如运算、信号、逻辑判断等等提前执行，直到遇到下一条运动指令。

子参数：

In：输入信号

Value：输入信号值，ON 或 OFF

一条运动指令中最多并行输出 2 个信号。

子参数：

No: 输出信号序号。

Value: 输出信号值。

Type: 运动中输出信号的类型。不同类型运动能使用不同的 Type。

Type	说明	适用的运动类型
T[n]	在运动中按时间输出信号 T[n] 代表时间，单位：秒，范围 -100.000~100.000。 n>=0 表示开始运动 n 秒后输出信号；n<0 表示到达目标点之前 n 秒时输出信号。 当超出运动范围外的时间被设定时，最多只在该段行程首末点触发，设定的时间无效。	Movj/ Movl/ Movc/Jump/ JumpL
D[n]	在运动中按路径百分比输出信号 D[n] 表示路径百分比。表示从开始运动到结束的 n% 路径时输出信号。	Movj/ Movl/ Movc/Jump/ JumpL
S[n]	在运动中按距离输出信号 S[n] 表示距离，单位：mm。范围 -65535~65535。 n>=0 表示从起点开始运动到 n 毫米之后时再输出信号；n<0 表示运动到距终点 n 毫米之前时输出信号。	Movl/ Movc/JumpL

注意：

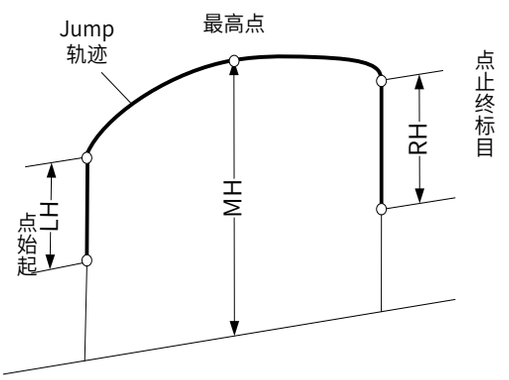
1. 设定在过渡区域内的运动 IO，精度是不能保证的

2. 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。

LH: 起始位置处的提升高度，取值范围 0.000~2000.000

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 -2000.000~2000.000。

RH: 到终止位置的下降高度，取值范围 0.000~2000.000。



注意：

1.Jump 指令仅用于 SCARA 及 Delta 机器人。

2. 由于 SCARA 机器人的基坐标系零点位于上方，因此 MH 多数情形下可能为负值。

3. 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，可用于特殊场合。

■ **范例**

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

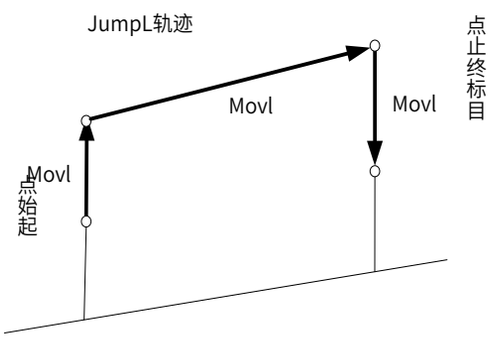
3.6 JumpL

■ **功能**

直线跳跃指令，中间段轨迹为直线。

■ **描述**

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。



■ **格式**

JumpL P, V, Z, [User], [Tool], [Acc], [NWait], [Out(No,value,Type)…], LH, MH, RH;

■ **参数**

P：要达到的目标点

常见形式：P[1],P[B1] 等。也可为偏移形式和托盘取点形式。

偏移形式——OffsetJ(P , PR)、OffsetT(P , PR)、Offset(P , PR)。

特别的，Offset 中可使用 PE，表示从当前位置偏移。

需要注意带偏移形式对其后的 [User], [Tool] 参数有一定要求。

更多见关于*偏移“Offset”*的使用。

托盘取点形式——Pallet(PalletNo, Row, Colunm, Lay)、LPallet(PalletNo, Row, Colunm, Lay)。根据托盘号、行号、列号、层号这些信息取托盘上的点，详见 *2.6.7 节 Pallet 指令*。

V：指定速度。如 V[50] 代表 50% 最大速度。

Z：到位与过渡参数。有 Fine、Z[0]、Z[1]、Z[2]、Z[3]、Z[4]、Z[5] 七种选择。

参数	说明
Fine	代表精确到位。机器人进入并停留在设定的到位误差范围内时，被算作精确到位，然后再执行后面的运动。
Z[0]~Z[5]	机器人不在目标点停留，而是平滑的通过设定的过渡区域，需要说明的是即使 Z[0] 也不能保证机器人精确到达目标点。
Z[CP]	机器人不在目标点停留，以最大过渡长度过渡，平滑的通过设定的过渡区域。

更多见*过渡特性*。

User：（可选参数）选用的用户坐标系，取 User[0]~User[15]。

更多见*关于 User 参数扩展使用*。

Tool：（可选参数）选用的工具坐标系，取 Tool [0]~ Tool [15]。

更多见*关于 Tool 参数扩展使用*。

Acc：（可选参数）指定加速度。如 Acc[50] 代表 50% 最大加速度。

NWait：（可选参数）不等待标识。

使用此项，代表不等到运行到目标点，立即执行后面的非运动指令。

这会使得其后的非运动指令，如运算、信号、逻辑判断等等提前执行，直到遇到下一条运动指令。

子参数：

In：输入信号

Value：输入信号值，ON 或 OFF

一条运动指令中最多并行输出 2 个信号。

子参数：

No: 输出信号序号。

Value: 输出信号值。

Type: 运动中输出信号的类型。不同类型运动能使用不同的 Type。

Type	说明	适用的运动类型
T[n]	在运动中按时间输出信号 T[n] 代表时间，单位：秒，范围 -100.000~100.000。 n>=0 表示开始运动 n 秒后输出信号；n<0 表示到达目标点之前 n 秒时输出信号。 当超出运动范围外的时间被设定时，最多只在该段行程首末点触发，设定的时间无效。	Movj/ Movl/ Movc/Jump/ JumpL
D[n]	在运动中按路径百分比输出信号 D[n] 表示路径百分比。表示从开始运动到结束的 n% 路径时输出信号。	Movj/ Movl/ Movc/Jump/ JumpL
S[n]	在运动中按距离输出信号 S[n] 表示距离，单位：mm。范围 -65535~65535。 n>=0 表示从起点开始运动到 n 毫米之后时再输出信号；n<0 表示运动到距终点 n 毫米之前时输出信号。	Movl/ Movc/JumpL

注意：

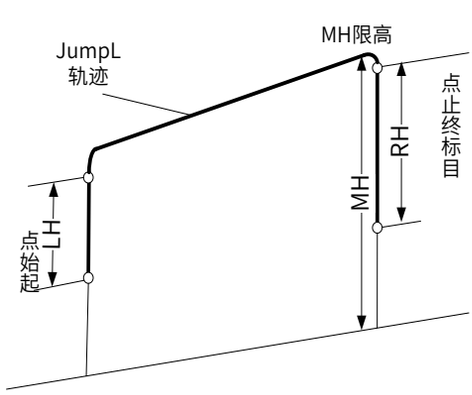
1. 设定在过渡区域内的运动 IO，精度是不能保证的

2. 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。

LH: 起始位置处的提升高度，取值范围 0.000~2000.000

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 -2000.000~2000.000。

RH: 到终止位置的下降高度，取值范围 0.000~2000.000。



注意：

1.Jump 指令仅用于 SCARA 及 Delta 机器人。

2. 由于 SCARA 机器人的基坐标系零点位于上方，因此 MH 多数情形下可能为负值。

3. 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，可用于特殊场合。

■ **范例**

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

3.6 Home

■ **功能**

回原点。

■ **描述**

回工作原点，可以指定期间的运动速度。

■ **格式**

Home[Index], [V];

■ **参数**

Index：工作原点号，0-4

V：（可选参数）回原点运动的速度。

■ **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）

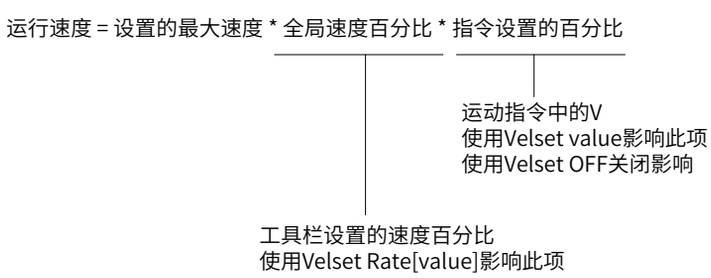
3.7 Velset

■ **功能**

速度设置。

■ **描述**

设置全局速度有两种形式“Velset Rate[value]”与“Velset value”，一个用来设置全局速度百分比，另一个用来强制指令速度。对于第二种形式“Velset value”，此指令一经设置，便一直生效，直至遇到 Velset OFF 停止或被再次设置替换。



■ **格式**

格式 1：Velset Rate[value];

格式 2：Velset value;

格式 3：Velset OFF;

■ **参数**

Value：速度值。代表指定以百分之多少的速度。

说明：

1） 格式 1 说明：设置全局速度百分比。

运行此指令，工具栏上的全局速度百分比会随之变化。

2） 格式 2 说明：强制指令速度。

无视 Move、Jump 系列指令后面的 V 参数，恒以 value 值作为指令设置的百分比。

注意 Velset value 不会参与程序逻辑，在该指令行之后，都会受其影响。除非遇到 Velset OF 去除影响或被再次设置替换其影响。相当于宏替换指令，即程序译码完成后对程序中 V[***] 进行替换。

作用域：当前程序文件中 Velset 数值到 Velset OFF 之间。

3） 格式 3 说明：

取消格式 1“Velset value”的速度设置。让运动指令中的速度设置继续生效。

■ **范例**

```
## 假设当前设置工具栏上速度为 100%
START;
Movj  P[0] ,V[70], Z[3];                    ## 实际速度为 100%*70%=70%
Velset Rate [50];
Movj  P[1] ,V[70], Z[3];                   ## 实际速度为 50%*70%=35%
Velset [30];
Movj  P[2] ,V[70], Z[3];                    ## 实际速度为 50%*30%=15%
 Velset OFF;
END;
```

3.8 WaitInPos

■ **功能**

等待运动完成。

■ **格式**

WaitInPos;

■ **案例**

```
Movj  P[1] ,V[40], Z[2];
WaitInPos;
Movj  P[2] ,V[40], Z[0];
## 等到 Movj P[1] 运动完成，再进行 Movj P[2] 运动，此时的 Z[2] 无效。
```

3.9 RefSys

■ **功能**

切换坐标系。

■ **描述**

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，需要再使用 Refsys 切换回来。

■ **格式 1**

RefSys Base;

■ **说明 1**

切换到基坐标系。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。

■ **格式 2**

RefSysConveyor(ConveyIndex, Tool);

■ **说明 2**

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，TCP 会跟随传送带同步运动。

■ **参数 2**

ConveyIndex：传送带编号，范围 0~3

Tool: 工具号，表示以指定的 TCP 同步传送带运动。

■ **格式 3**

RefSysWorkBench(ConveyIndex, Tool);

说明：

切换到旋转工作台坐标系。由于旋转工作台是一个动态的坐标系，因此执行此指令后，TCP 会跟随工作台同步运动。

■ **参数**

ConveyIndex：传送带编号，范围 0~3

Tool: 工具号，表示以指定的 TCP 同步传送带运动。

注意：使用传送带跟随工艺时，对于指令 PE，VelSet 是无效的。即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;
```

3.10 LockScrew

■ **功能**

螺丝锁付。

■ **描述**

完成螺丝的锁付操作。

■ **格式**

LockScrew(TechID，P，V[Vel]);

■ **参数**

TechID：锁螺丝工艺号，即锁螺丝所使用的工艺参数号（0~15）。

P：锁付的位置变量编号。

Vel：滑台下降速度，用户根据节拍自设。

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。

3.11 UnLockScrew

■ **功能**

松螺丝。

■ **描述**

从指定的位置拆除一个螺丝。

■ **格式**

UnLockScrew(TechID，P，V[Vel]，RetreatDist，UnlockVel);

■ **参数**

TechID：松螺丝工艺号，即拆螺丝所使用的工艺参数号（0~15）。

P：松螺丝的起始位置编号。

Vel：滑台下降的速度，用户可根据节拍自设。

RetreatDist：松螺丝时滑台的回退距离，单位 mm。

UnlockVel：松螺丝速度，单位 mm/s。

说明：

拆卸过程的工艺参数均能通过工艺号选取；

相比锁指令，拆指令多了松螺丝时回退距离参数。

3.12 ArmChange

■ **功能**

SCARA 手臂切换。

■ **描述**

用于 SCARA 机器人切换左右手。

■ **格式**

ArmChange value,V;

■ **参数**

Value：手臂姿势，可为 -1、1、0。

-1 代表左边手臂。

1 代表右手臂。

0 代表自动切换，变为与当前当前手臂类型相反。

V：切换运动的速度，如 V[50] 代表 50% 最大速度。

■ **范例**

ArmChange 1,V[20];

3.13 SlewMode

■ **功能**

旋转优化方式的设置。

■ **描述**

设置 SCARA 的 J4 或标准 6 关节的 J6 的旋转优化方式。它使得其后的运动中，都采取一定的方式优化运动，使得最终走到目标点的位置，但可能最终不会采用目标点所指定的臂参数。也就是说保证了目标点的空间位置姿态 (X,Y,Z,A,B,C)，但可能会改变臂参数。多用于相机坐标转换成机器人坐标后，采用该指令优化运动，使得机器人的最后一个轴不会过多的旋转。该指令一经设置，对其后运动一直生效，除非再次设置。

■ **格式**

SlewMode nMode;

■ **参数**

nMode：旋转优化方式，有 0、1、2、3 四种模式。（对于标准 6 关节机器人，用 J6 替代下面的 J4）

0：不采用优化，以位置变量中的臂参数为准

1：采用优化模式 1，始终保证运动中 J4 关节在 (-180~180] 范围内。

2：采用优化模式 2，始终保运动中 J4 以最近的方式运动。即衡量要运动到的目标点是否需要经过 J4 旋转 180。若角度差 <=180°，则完全运动到目标点。若 >180°，则 J4 以相反的方向运动，最终运动到距目标点的 J4 相差一圈（360°）的位置；期间，J4 反向运动若超出机器人极限范围，则停止并报警。

3: 采用优化模式 3，基本方式与 2 类似，以最近的方式运动。差别在于 J4 反向运动若超出机器人极限范围，不会报警，而是不进行反向运动，而是完全采取原方式正常运动。

■ **范例**

```
SlewMode 3;                                    ## 采取模式 3 优化
Movj P[1],V[30],Z[0];
Movj P[2],V[30],Z[0];
SlewMode 0;                                    ## 关闭优化
```

第 4 章 信号处理指令

4.1 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

4.1.1 Set Out

■ **功能**

Out 输出。

■ **描述**

设置单个数字输出信号。

■ **格式**

Set Out,value;

■ **参数**

Out：数字输出信号，Out[0]~Out[63]

Value：数字信号的值只能为 ON/OFF

■ **范例**

Set Out[1],ON; *## 设置 Out[1] 输出 ON*

4.1.2 Set OG

■ **功能**

OG 输出。

■ **描述**

设置一组数字输出信号。

■ **格式**

Set OG, B;

■ **参数**

OG：输出组信号，OG[0]~OG[15]；默认每组信号包含 8 个信号

B：B 变量，代表一组信号状态

说明：

默认一组信号包含 8 个信号，OG[0] 为 Out[0]~Out[7]，OG[1] 为 Out[8]~Out[15]，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
```

4.1.3 Set DA

■ **功能**

DA 设置。

■ **描述**

设置模拟量输出信号。

■ **格式**

Set DA,Value;

■ **参数**

DA：模拟量输出信号，DA[0]~ DA[15]

Value: 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。

4.2 Get

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（默认一组含 8 个 In 信号）

Get OG：读取一组数字输出信号（默认一组含 8 个 Out 信号）

Get AD: 读取单个模拟量输入信号

4.2.1 Get In

■ **功能**

In 读取 。

■ **描述**

读取单个数字输出信号。

■ **格式**

Get In,B;

■ **参数**

In：数字输入信号，In[0]~In[63]

B：B 变量，保存读取的结果。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;
```

4.2.2 Get Out

■ **功能**

Out 读取。

■ **描述**

读取单个数字输出信号。

■ **格式**

Get Out,B;

■ **参数**

Out: 数字输出信号，Out[0]~Out[63]

B：B 变量，保存读取的结果。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;
```

4.2.3 Get IG

■ **功能**

IG 读取。

■ **描述**

读取一组数字输入信号。

■ **格式**

Get IG, B;

■ **参数**

IG：输入信号组，范围 IG[0]~IG[15]，默认每组信号包含 8 个信号

B：B 变量，保存读取结果。代表一组信号状态

■ **说明**

默认一组信号包含 8 个信号，IG[0] 为 In[0]~In[7]，IG[1] 为 In[8]~In[15]，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[15]~In[8] 依次为 OFF、OFF、OFF、OFF、OFF、ON、ON、ON，则 B1=7
```

4.2.4 Get OG

■ **功能**

OG 读取。

■ **描述**

读取一组数字输出信号。

■ **格式**

Get OG[***], B***;

OG：输出信号组，范围 OG[0]~OG[15]，默认每组信号包含 8 个信号

B：B 变量，保存读取结果。代表一组信号状态

说明：

默认一组信号包含 8 个信号，OG [0] 为 Out[0]~Out [7]，OG [1] 为 Out [8]~Out [15]，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 [15]~Out [8] 依次为 OFF、OFF、OFF、OFF、OFF、ON、ON、ON，则 B1=7
```

4.2.5 Get AD

■ **功能**

AD 读取。

■ **描述**

读取模拟量输入信号。

■ **格式**

Get AD,D;

■ **参数**

AD：模拟量输入信号，AD[0]~AD[15]

D：D 变量，保存结果。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。*

4.3 Wait

■ **功能**

信号等待。

■ **描述**

等待直至检测到输入输出信号满足条件。

■ **格式**

Wait In == value, T;

Wait In == value, T, Goto Lable;

Wait Out == value, T;

Wait Out == value, T, Goto Lable;

■ **参数**

In：监测的输入端口，In[0]~In[63]

Out: 监测的输出端口，Out[0]~Out[63]

value: 信号值，只为 ON/ OFF

T: 等待时间，单位 s, 范围 <0.000~65535.000>。特别地，T[0] 表示等待时间为无限长，即一直等待输入信号，直到接受指定的输入信号。

Goto：（可选参数）不使用该参数，超出时间而条件仍不满足时，则会报警。若使用该参数，超出时间而条件仍不满足时，则返回标签处。

■ **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。

4.4 Delay

■ **功能**

延时。

■ **描述**

程序延时，时间由 T 参数控制。

■ **格式**

Delay T;

■ **参数**

T：时间，单位 s，取值范围 <0.000~65535.000>。时间精度一般为 0.1 s。

■ **范例**

```
Delay T[5];                                    ## 延时 5 秒
```

4.5 Invert

■ **功能**

信号取反。

■ **描述**

反转输出信号。

■ **格式**

Invert Out;

■ **参数**

Out：输出信号，Out[0]~Out[63]

■ **范例**

```
Set Out[1],ON;
Invert Out[1];                                 ##Out[1] 为 OFF
```

4.6 Group

■ **功能**

组信号配置。

■ **描述**

配置输入、输出组信号。一个组信号，默认包含 8 个信号，如 OG[0] 中包含 Out0-7，如 OG[1] 中包含 Out 8-15…… 使用 Group 指令后，可重新定义输出。新组中信号的数目可小于 8。但要注意，仅在当前程序有效。

■ **格式**

Group OG, n0, n1, n2…, n7;

Group IG, n0, n1, n2…, n7;

■ **参数**

OG：待重定义的输出组，OG[0]~OG[8]。

IG：待重定义的输入组，IG[0]~IG[8]。

n0, n1, n2…, n7：信号的序号。可多选，最多 8 个，最少 2 个。

■ **范例**

```
Group OG[0] 1,2,7;                             ## 重定义 OG[0] 为 Out[1]、Out[2]、Out[7]。
```

第 5 章 托盘指令

5.1 Msft

■ **功能**

平移计算。

■ **描述**

计算两个位置变量间的平移变量。

■ **格式 1**

PR = Msft (FromP,ToP);

说明：计算从 FromP 到 ToP 的平移变量。平移变量是以 P[1] 所用的坐标系衡量。

■ **格式 2**

PR = Msft (FromP, ToP,Tcp);

说明：计算从 FromP 到 ToP 的平移变量。平移变量是以 P[1] 当前位姿衡量。

■ **参数**

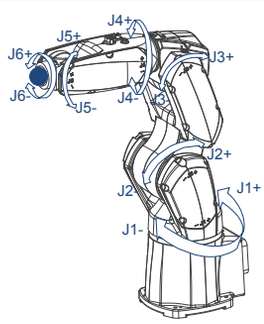
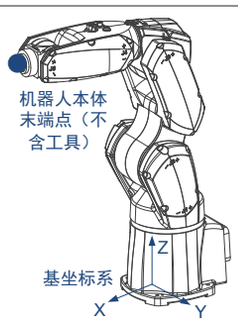
FromP: 起始点

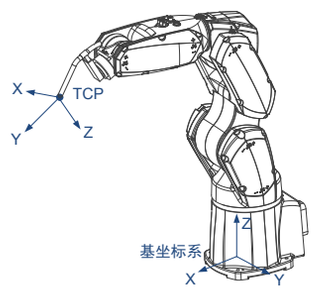
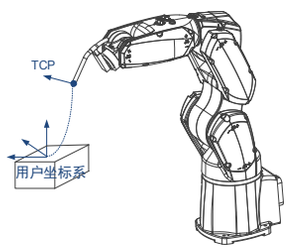
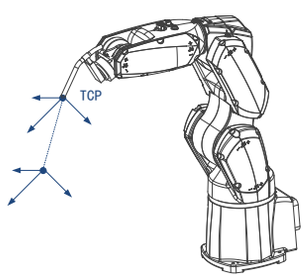
ToP：终止点

■ **返回值**

PR： PR/LPR 变量，作为计算结果

不同的平移计算见下表：

平移类型	用法	示意图
关节平移计算	使用条件：P[1] 坐标系号为 1 典型使用形式： P[1] = (0,0,0,0,90,0),(1,-1,1,0)(1,0,0); P[2] = (10,10,10,10,100,10),(1,-1,1,0)(1,0,0); PR1= Msft (P[1],P[2]); 备注：当 P[2] 为非关节点时，会先转化为关节下的点。	
本体末端沿基坐标系平移	使用条件：P[1] 坐标系号为 2 典型使用形式： P[1] = (100,0,200,0,0,0),(1,-1,1,0)(2,0,0); P[2] = (110, 0,190,0,0,0),(1,-1,1,0)(2,0,0); PR1= Msft (P[1],P[2]); 备注：当 P[2] 为非基坐标点，会先转化为机器人末端在基坐标系下点。再参与平移计算。	

TCP 沿基坐标系平移	典型使用形式：P[1] 坐标系号为 3 P[1] = (110,0,150,0,0,0),(1,-1,1,0)(3,1,0); P[2] = (120, 0,140,0,0,0),(1,-1,1,0)(3,1,0); PR1= Msft (P[1],P[2]); 备注：当 P[2] 为非工具点，会先转化为工具末端在基坐标系下点。再参与平移计算。	
TCP 沿用户坐标系平移	典型使用形式：P[1] 坐标系号为 4 P[1] = (20,0,20,0,0,0),(1,-1,1,0)(4,1,2); P[2] = (30, 0,10,0,0,0),(1,-1,1,0)(4,1,2); PR1= Msft (P[1],P[2]); 备注：当 P[2] 为非用户点，会先转化为工具末端在用户坐标系下点。再参与平移计算。	
TCP 沿 工具坐标系自身的平移	典型使用形式：Msft 指令带有 Tcp 参数 P[1] = (10,0,20,0,0,0),(1,-1,1,0)(4,1,2); P[2] = (120, ,140,0,0,0),(1,-1,1,0)(4,1,2); PR1= Msft (P[1],P[2],Tcp); 备注：沿 P[1] 工具当前位置姿态移动到 P[2]。	

5.2 PR***=

■ 功能

PR 赋值。

■ 描述

平移变量的直接赋值。

■ 格式

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

■ 参数

(X,Y,Z,A,B,C)：作为平移量的参数值。

■ 返回值

PR：PR/LPR 变量，赋值的对象

■ 范例

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
```

5.3 PR Sum

■ **功能**

PR 求和 / 差。

■ **描述**

两个平移变量加减运算

■ **格式**

格式 1：PR3 = PR1 + PR2;

格式 2：PR3 = PR1 – PR2;

说明: 两个平移变量直接进行数值上的加减运算，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;
```

5.4 P[***]=

■ **功能**

点赋值。

■ **描述**

直接对位置变量赋值。

■ **格式**

P=(X,Y,Z,A,B,C),(ArmType[0],ArmType[1],ArmType[2],ArmType[3]),(CoordNo, ToolNo, UserNo);

■ **参数**

(X,Y,Z,A,B,C)：坐标值。

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

(CoordNo, ToolNo, UserNo): 坐标系号，工具号，用户号

■ **返回值**

P：位置变量

■ **范例**

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

5.5 P=Offset

■ **功能**

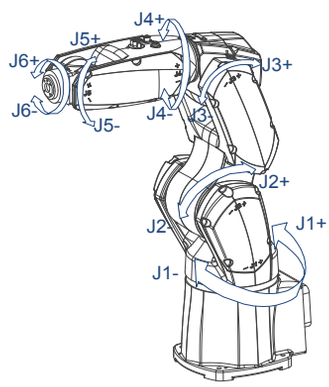
点偏移。

■ **描述**

基于点的平移，有以下三种形式：

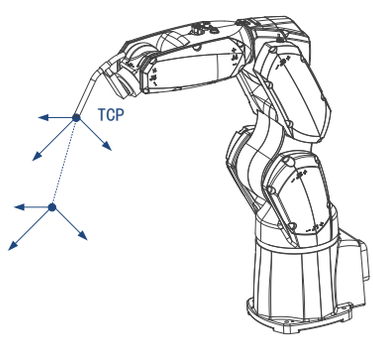
格式 1：DestP = OffSetJ(SourceP,PR);

说明：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）。



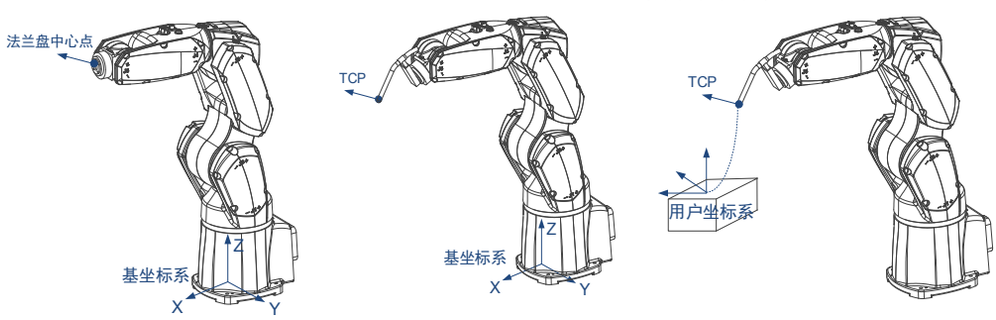
格式 2：DestP = OffSetT(SourceP,PR);

说明：在当前的工具坐标系基础上进行平移。PR 为移动和旋转 (X,Y,Z,A,B,C)，平移顺序：X>Y>Z>A>B>C。



格式 3：DestP = OffSet(SourceP,PR);

说明：Offest 表示目标点在某个笛卡尔参考系下的平移。目标点可以是 TCP 也可以是法兰盘中心点，参考系可以是用户坐标系也可是基坐标系。此时 PR 为移动和旋转 (X,Y,Z,A,B,C)，平移顺序：X>Y>Z>A>B>C。



由于坐标值的涵义由坐标系指定，因此平移的目标点及参考坐标系由坐标系号指定。

1） 当位置变量中的坐标系号为 1 时，表示法兰盘中心点在基坐标下平移。

2） 当位置变量中的坐标系号为 2 时，表示法兰盘中心点在基坐标下平移。

3） 当位置变量中的坐标系号为 3 时，表示 TCP 在基坐标下平移。工具由位置变量的工具号指定。

4） 当位置变量中的坐标系号为 4 时，表示 TCP 在用户坐标下平移，工具由位置变量的工具号指定，用户坐标系由位置变量的用户号指定。

推荐：可以利用 Cnvrt 指令更改位置变量的在指定的坐标系下表示，再配合 P=Offset(P,PR)，就可以明确平移的目标点和参考系。

■ **参数**

SourceP：源点。

PR：PR/LPR 变量，作为平移量。

■ **返回值**

DestP：偏移后的点

■ **范例**

范例 1

```
PR1=(10,20,0,0,0,0);
P[2]=OffsetJ(P[1],PR1);
MovjP[2],V[30],Z[0];
## 在 P[1] 基础上，机器人第一关节额外旋转 10°，第二关节额外旋转 20°。
```

范例 2

```
PR1=(10,0,0,10,20,30);
P[2]=OffsetT(P[1],PR1);
MovjP[2],V[30],Z[0],Tool1;
## 要到达的目标点的位置为，在 P[1] 基础上，沿着当前的工具坐标系运动，沿 X 方向移动 10mm，绕 Z 旋转 10°，再绕 Y 旋转 20°的位置，最后绕 X 旋转 30°的位置。
```

范例 3

```
## 基坐标系下取点，法兰盘在基坐标系下移动。
P[1]=(10,20,30,40,50,60;1,0,0,0;2,1,0);                    ## 坐标系号为 1 也可以
PR1=(10,0,0,0,0,0);
P[2]=Offset(P[1],PR1);
## 由于 P[1] 在基坐标系下取点（坐标系号 2），因此是法兰盘中心相对基坐标的平移。注意此时的目标点就不是 TCP 了。
## 此时若需要 TCP 相对基坐标系的平移，则可以如下使用：
P[1]=(10,20,30,40,50,60;1,0,0,0;2,1,0);
Cnvrt(P[1],P[1],Tool[1]);                                  ## 将 P[1] 转换到工具坐标系下表达
P[2]=Offset(P[1],PR1);
Movj P[2],V[30],Z[0];
## 当然若 P[1] 是在工具坐标系下取点（坐标系号 3），可以不用 Cnvrt 转换。
## 再比如，由于某种特殊需求，需要 TCP 相对额外的一个用户坐标系 User1 平移，则可如下：
P[1]=(10,20,30,40,50,60;1,0,0,0;2,1,0);
Cnvrt(P[1],P[1],User[1]);                                  ## 将 P[1] 转换到 User1 下表达
P[2]=Offset(P[1],PR1);
Movj P[2],V[30],Z[0];
```

5.6 LPallet

■ **功能**

LPallet 定义。

■ **描述**

设定局部托盘变量。常与 P=Pallet 配合使用，用于使用简单的托盘。

根据三个点 P[i],P[j],P[k] 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。该指令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**

LPallet LPalletNo,P[i],P[j],P[k],nRow, nColumn,nLay,LayHeight;

■ **参数**

LPalletNo：局部托盘变量的序号

P[i],P[j],P[k]：定义托盘的三个点

nRow：行数，取值范围为 <1~1000>

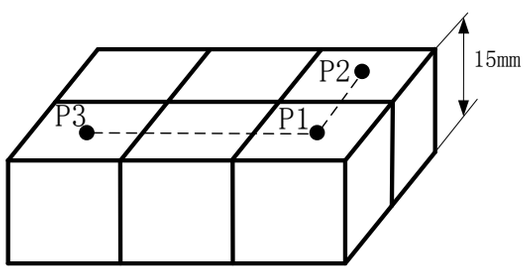
nColumn：列数，取值范围为 <1~1000>

nLay：层数，取值范围为 <1~1000>

LayHeight：层高，单位 mm

■ **范例**

LPallet 1,P[1],P[2],P[3],2,3,1,15;



5.7 LPallet4

■ **功能**

LPallet 的 4 点定义。

■ **描述**

4 点设定局部托盘变量。常与 P=Pallet 配合使用，用于使用简单的托盘。

与 LPallet 不同，这种托盘是以 4 个点确定 4 条边界，以行数、列数进行分割，所得各点规则的。

■ **格式**

LPallet4 LPalletNo,P[i],P[j],P[k], P[m],nRow, nColumn,nLay,LayHeight;

■ **参数**

LPalletNo：局部托盘变量的序号

P[i],P[j],P[k], P[m]：定义托盘的四个点

nRow：行数，取值范围为 <1~1000>

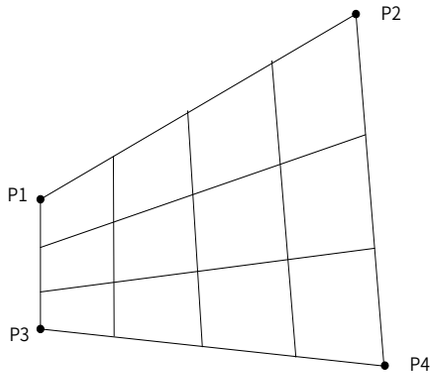
nColumn：列数，取值范围为 <1~1000>

nLay：层数，取值范围为 <1~1000>

LayHeight：层高，单位 mm

■ **范例**

LPallet4 2,P[1],P[2],P[3], P[4],4,3,1,15;



5.8 P=Pallet

■ **功能**

取托盘点。

■ **描述**

取托盘上的点。与 LPllet 指令配合使用，可使用局部托盘变量中的点；与码垛工艺设置配合使用，可设置全局托盘变量中的点。

■ **格式**

格式 1：P=Pallet(PalletNo, Row, Column, Lay);

格式 2：P=LPallet(PalletNo, Row, Column, Lay);

■ **参数**

PalletNo：托盘号，使用格式 1，则为全局托盘变量号；使用格式 1，则为局部托盘变量号

Row：行号

Column：列号

Lay：层号

■ **返回值**

P：位置变量，保存托盘上的选定点

注意：

行、列、层号是从 0 开始计数的。

■ **范例**

P[2]=Pallet(1,1,1,0);

5.9 MovToPut

■ **功能**

码垛指令。

■ **描述**

运行至托盘上货物的放置点。

■ **格式**

MovToPut PalletVar, P, X, Y, Z, V, PickV, [Acc], [Tool], [Out(No,Value,D[n]|T[n]|S[n])];

■ **参数**

PalletVar：全局托盘变量 Pallet[***]

P：进入点 P[***]

X：准备点相对于放置点的 X 方向偏移量。

Y：准备点相对于放置点的 Y 方向偏移量。

Z：准备点相对于放置点的 Z 方向偏移量。

V：进入点到准备点的速度 V[***]。*** 代表百分比

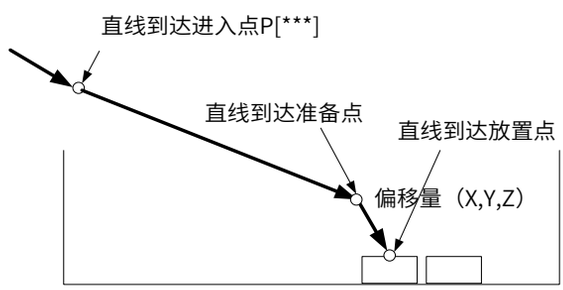
PickV：准备点到放置点的速度 PickV [***]。*** 代表百分比

Acc：（可选参数）加速度 Acc[***]，*** 代表百分比

Tool：（可选参数）所使用的工具，Tool[0]~Tool[15]

Out(No,Value,D[n]|T[n]|S[n])：（可选参数）并行 IO 输出指令，整个运动路径中同时最多 2 个 IO 触发：（运动过程指整个运动路径）。三种触发方式可选其一。

参数	说明
T[n]	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]	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]	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];

5.10 MovToGet

■ **功能**

拆垛指令。

■ **描述**

运行至托盘上货物的放置点。

■ **格式**

MovToGet PalletVar, P, X, Y, Z, V, PickV, [Acc], [Tool], [Out(No,Value,D[n]|T[n]|S[n])];

■ **参数**

PalletVar：全局托盘变量 Pallet[***]

P：进入点 P[***]

X：准备点相对于放置点的 X 方向偏移量。

Y：准备点相对于放置点的 Y 方向偏移量。

Z：准备点相对于放置点的 Z 方向偏移量。

V：进入点到准备点的速度 V[***]。*** 代表百分比

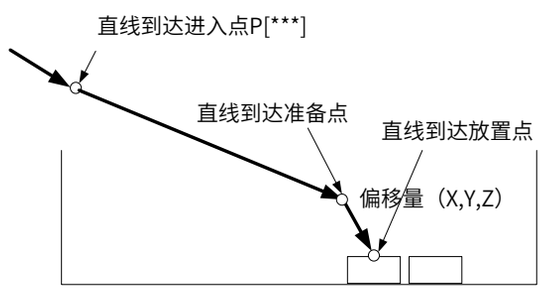
PickV：准备点到放置点的速度 PickV [***]。*** 代表百分比

Acc：（可选参数）加速度 Acc[***]，*** 代表百分比

Tool：（可选参数）所使用的工具，Tool[0]~Tool[15]

Out(No,Value,D[n]|T[n]|S[n])：（可选参数）并行 IO 输出指令，整个运动路径中同时最多 2 个 IO 触发：（运动过程指整个运动路径）。三种触发方式可选其一。

参数	说明
T[n]	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]	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]	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];

5.11 MovFromPut

■ **功能**

码垛返回指令。

■ **描述**

码垛放置完成后，从托盘上货物的放置点返回。

■ **格式**

MovFormPut PalletVar, P, X, Y, Z, V, PickV, [Acc], [Tool], [Out(No,Value,D[n]|T[n]|S[n])];

■ **参数**

PalletVar：全局托盘变量 Pallet[***]

P：进入点 P[***]

X：准备点相对于放置点的 X 方向偏移量。

Y：准备点相对于放置点的 Y 方向偏移量。

Z：准备点相对于放置点的 Z 方向偏移量。

V：准备点到进入点的速度 V[***]。*** 代表百分比

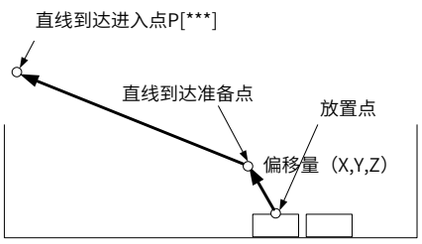
PickV：放置点到准备点的速度 PickV [***]。*** 代表百分比

Acc：（可选参数）加速度 Acc[***]，*** 代表百分比

Tool：（可选参数）所使用的工具，Tool[0]~Tool[15]

Out(No,Value,D[n]|T[n]|S[n])：（可选参数）并行 IO 输出指令，整个运动路径中同时最多 2 个 IO 触发：（运动过程指）。三种触发方式可选其一。

参数	说明
T[n]	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]	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]	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];

5.12 MovFromGet

■ **功能**

拆垛返回指令。

■ **描述**

拆垛放置完成后，从托盘上货物的放置点返回。

■ **格式**

MovFromGet PalletVar, P, X, Y, Z, V, PickV, [Acc], [Tool], [Out(No,Value,D[n]|T[n]|S[n])];

■ **参数**

PalletVar：全局托盘变量 Pallet[***]

P：进入点 P[***]

X：准备点相对于放置点的 X 方向偏移量。

Y：准备点相对于放置点的 Y 方向偏移量。

Z：准备点相对于放置点的 Z 方向偏移量。

V：准备点到进入点的速度 V[***]。*** 代表百分比

PickV：放置点到准备点的速度 PickV [***]。*** 代表百分比

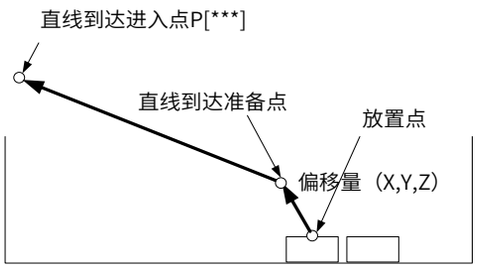
Acc：（可选参数）加速度 Acc[***]，*** 代表百分比

Tool：（可选参数）所使用的工具，Tool[0]~Tool[15]

Out(No,Value,D[n]|T[n]|S[n])：（可选参数）并行 IO 输出指令，整个运动路径中同时最多 2 个 IO 触发：（运动过程指）。三种触发方式可选其一。

参数	说明
T[n]	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。

参数	说明
D[n]	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]	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];

5.13 ResetPallet

■ **功能**

托盘初始化。

■ **描述**

在使用其它码垛指令前，必须先初始化托盘。

■ **格式**

ReSetPallet[PalletNo];

■ **参数**

PalletNo：托盘号

■ **范例**

ReSetPallet[1];

5.14 IsPalletFinished

■ **功能**

码垛完成查询。

■ **描述**

查看码垛或者拆垛是否完成。

■ **格式**

IsPalletFinished(PalletVar, B);

■ **参数**

PalletVar：全局托盘变量

B: B/LB 变量，用于存储结果，1 完成 ,0 未完成

■ **范例**

IsPalletFinished(Pallet[1],B1);

5.15 GetPalletRunNo

■ **功能**

码垛运行点查询。

■ **描述**

查看码垛或者拆垛运行点号。

■ **格式**

GetPalletRunNo(PalletVar, R);

■ **参数**

PalletVar：全局托盘变量

R: R/LR 变量，用于存储结果当前运行托盘点的序号

■ **范例**

GetPalletRunNo(Pallet[1],R1);

5.16 SetPalletRunNo

■ **功能**

托盘运行起始点设置。

■ **描述**

设置码垛或者拆垛运行点号，表示从该点开始继续往后运动。

■ **格式**

SetPalletRunNo(PalletVar, PNo);

■ **参数**

PalletVar：全局托盘变量

PNo：要设置的托盘上运行点的序号

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;
```

5.17 EOffsOn

■ **功能**

路径平移开启。

■ **描述**

开启整体路径平移。运动的最终路径将在基坐标系下的进行额外 XYZ 方向偏移。一经开启，一直生效，直到遇到 EOffsOff 指令关闭路径偏移。

■ **格式**

EOffsOn(X,Y,Z);

■ **参数**

X：基坐标系下的 X 方向偏移量

Y：基坐标系下的 Y 方向偏移量

Z：基坐标系下的 Z 方向偏移量

■ **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：

1. 路径的偏移只作用于 EoffsOn 和 EoffsOff 之间的指令。路径偏移与程序逻辑无关，只与上下文有关，如下：

```
Movj P[1],V[30],Z[0];
B1 = 2;
If B1<1
EOffsOn(10,0,0);                    ## 不管该条指令有没有被执行，偏移自此处以后就开启
EndIf;
Movl P[2],V[30],Z[0];               ## 有偏移
If B1>1
Goto L[1];
EndIf;
L[2]:
Movl P[3],V[30],Z[0];               ## 此处不管从其他何处跳转而来，该条指令处于 EoffsOn 和
                                    EoffSetoff 之间，因而均会平移
EOffsOff;
L[1]:
Movl P[4],V[30],Z[0];               ## 该条指令在 EoffsOn 和 EoffSetoff 之外， 无偏移
Goto L[2];
```

2.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。

5.18 EOffsOff

■ **功能**

路径平移关闭。

■ **描述**

关闭路径平移。使用 EOffsOn 后，路径平移一直启用，只有遇 EOffsOff 关闭。

■ **格式**

EOffsOff;

详见 EOffsOn。

第 6 章 流程控制指令

6.1 L-Goto

■ **功能**

逻辑跳转。

■ **描述**

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[lableNo]: # 不能重复

……

Goto L[lableNo];

■ **参数**

lableNo：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;
## 本段执行效果：先运行至 P[0] 位置，然后在 P[1] 与 P[0] 两点间往复运动。
```

6.2 If-Else-EndIf

■ **功能**

条件判断。

■ **描述**

逐个进行条件判断，若满足则执行对应语句。先判断 If 条件 condition1，满足执行语句 Stmt1，完成后结束；若不满足且存在 ElseIf，则判断 ElseIf 条件 condition2，满足执行语句 Stmt2，完成后结束；以此类推，判断其它 ElseIf；以上条件都不满足且存在 Else，执行 Else 后语句 Stmtn，完成后结束。

■ **格式**

```
If condition1
    Stmt1;
ElseIf condition2
    Stmt2;
ElseIf condition3
    Stmt3;
……
Else
```

Stmtn;

EndIf;

■ **参数**

Condition：代表条件

Stmt：满足该条件的执行语句

说明：

判断条件 Condition 可为多个条件复合。

语句 Stmt 可为数行指令。

If 作为开头，Else 可缺省，EndIf 作为段落的结束不可缺少。

■ **范例**

```
START;
If IN[1]==ON And IN[2]==ON ;
   Movj P[1],V[50],Z[3];
Else
   Movj P[2],V[50],Z[3];
   Movj P[3],V[50],Z[3];
EndIf;
End;
```

6.3 Switch-Case-Default-EndSwitch

■ **功能**

条件选择语句。

■ **描述**

根据 Switch 后变量，选择不同的 Case，执行后面的 Stmt1。若所有 Case 均不符合 , 则执行缺省（Default 后）的语句。

■ **格式**

Switch Var

Case value1:

Stmt1;

Break;

Case value2:

Stmt2;

Break;

……

Default:

Stmtn;

Break;

EndSwitch;

■ **参数**

Var：变量，一般为 B/LB/R/LR/ 字符串变量。

Value：变量的值。可为整数，对应 B/LB/R/LR 变量；也可以字符串，对应字符串变量。

Stmt：满足该条件的执行语句

扩展：

Case A To B : 可使用 Case A To B 代替 Case Value。Case A To B 代表从 A 到 B 两个整数范围内 (包含 A、B) 的的选择。当变量值在这一范围内时，代表此 Case 项匹配。

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 段落语句可省略。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 To 4:
    Movj P[1],V[50],Z[3];
    Movj P[2],V[50],Z[3];
  Break;
  Case 5:
    Movj P[3],V[50],Z[3];
  Break;
  Default:
    Movj P[4],V[50],Z[3];
  Break;
EndSwitch;
```

6.4 While-EndWhile

■ **功能**

条件循环。

■ **描述**

判断条件，若满足条件，则执行循环体内的语句。一旦不满足条件，则跳出。

■ **格式**

While Condition

    Stmt;

EndWhile;

■ **参数**

Condition：代表条件

Stmt：满足该条件的执行语句

说明：

判断条件 Condition 可为多个条件复合。

语句 Stmt 可为数行指令。

■ **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。

6.5 For-EndFor

■ **功能**

带执行次数的循环语句。

■ **描述**

先执行一个初始化赋值语句，再判断条件，若满足条件则执行循环体内的内容，执行完成一次后，执行一个步长，使得初始化赋值语句中的变量自增，再判断条件 , 若满足则继续刚才的步骤，直至条件不成立时跳出。

■ **格式**

For InitExp, Condition, Step[n]

　　Stmt;

EndFor;

■ **参数**

InitExp：初始化赋值语句，这里限定为 B/R/LB/LR 变量的赋值语句。

Condition：条件

n：步长，执行一次循环体内容后，对应变量每一次的增量。取值范围为 -65536~65535 以内的整数。

Stmt：满足该条件的执行语句

■ **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

6.6 Break

■ **功能**

结束循环体。

■ **描述**

完全结束一个循环，跳出循环体，执行循环后面的语句。如跳出 While 或 For 循环、Switch 语句中执行 Case 段后跳出。多层循环嵌套时，只作用于当前层级的循环。

■ **格式**

Break;

■ **范例**

```
START;
LB1 = 0;
For B1=0,B1<5,Step[1]
If LB1>1
Break;
EndIf
Movj P[0],V[30],Z[0];
Movj P[1],V[30],Z[0];
Incr LB1;
EndFor;
END;
## 执行 2 次“Movj P[0]、Movj P[1]”。
```

6.7 Continue

■ **功能**

结束本次循环。

■ **描述**

跳过当次循环中剩下的语句，执行下一次循环。多层循环嵌套时，只作用于当前层级的循环。

■ **格式**

Continue;

■ **范例**

```
START;
LB1 = 0;
For B1=0,B1<5,Step[1]
If LB1==1
Continue;
EndIf
Movj P[0],V[30],Z[0];
Movj P[1],V[30],Z[0];
Incr LB1;
EndFor;
END;
## 执行 4 次“Movj P[0]、Movj P[1]”。
```

6.8 Call

■ **功能**

子程序调用。

■ **描述**

调用一个子程序。调用执行后，进入子程序程序；在子程序中遇到 Ret 指令，则返回主程序继续执行。

■ **格式**

Call “subProgram”;

■ **参数**

SubProgram：需要调用的子程序名

注意事项：常与 Ret 配合使用，详见下 Ret 指令。嵌套使用子程序不能超过 3 层。若子程序中无 Ret 指令，则在子程序中遇到 End 主程序也结束。

■ **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
```

6.9 Ret

■ **功能**

返回主程序。

■ **描述**

返回主程序，在子程序中执行到该指令时，不再执行子程序后续内容，返回主程序，执行主程序后面的语句。

■ **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 主程序也结束。范例：

在主程序 parents.pro 中调用子程序 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，便返回至主程序。整个运动过程为：

主程序P[1] ⟶ 主程序P[6] ⟶ 子程序P[0] ⟶ 子程序P[1] ⟶ 主程序P[7]

6.10 Pause

■ **功能**

暂停。

■ **描述**

暂停运行。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。

■ **格式**

Pause;

6.11 点文件系列

6.11.1 LoadPointFromFile

■ **功能**

点文件加载。

■ **描述**

从点文件中加载点到系统中。

■ **格式**

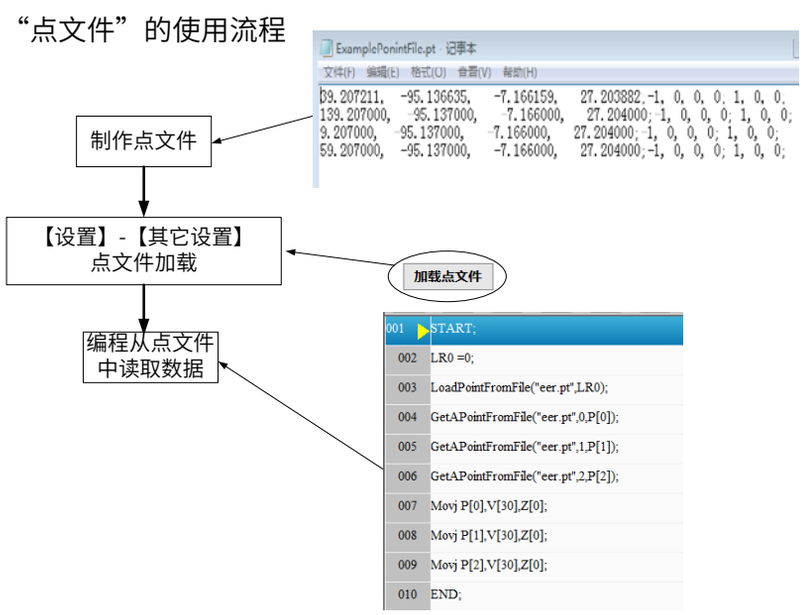
LoadPointFromFile(PtFile, R);

■ **参数**

PtFile：点文件名

R：R/LR 变量，用于保存读加载点的个数

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：



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]。

注意事项：

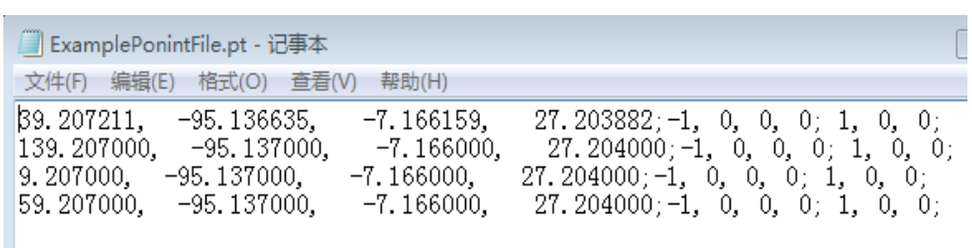
如果程序中没有位置变量被预先定义，（如上程序中没有 P[0]、P[1]、P[2]），使用指令“GetAPointFromFile”则会报错。

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 1-6 个参数为第一段，为坐标系值；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。

坐标值参数可指定 1-6 个，不足 6 个的部分按 0 算。

一个示例如下所示：



6.11.2 GetAPointFromFile

■ **功能**

从点文件中获取点。

■ **描述**

从系统中加载单个点到本程序的位置变量。

■ **格式**

GetAPointFromFile(PtFile, Index , P);

■ **参数**

PtFile：点文件路径名

Index：点在文件中的索引号（行号）

P：提取的点放在此位置变量中

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，使用指令“GetAPointFromFile”则会报错。

6.11.3 WriteAPointToFile

■ **功能**

向点文件中写入点。

■ **描述**

指定点文件中的某行，向其写入点。

■ **格式**

WriteAPointToFile(PtFile, Index, P);

■ **参数**

PtFile：点文件路径名

Index：要存的点点在文件中的索引号（行号）

P：要放入点文件的位置变量

说明：写入点文件后必须调用指令 SavePointFile 才能保存

■ **范例**

WriteAPointToFile(“eer.pt”, 0, P[0]);
WriteAPointToFile(“eer.pt”, 1, P[1]);
SavePointFile (“eer.pt”);

6.11.4 SavePointFile

■ **功能**

保存点文件。

■ **格式**

SavePointFile (PtFile);

■ **参数**

PtFile：点文件路径名

6.12 多任务指令

机器人最多可以支持多个任务同时运行，按任务类型分为以下 3 类：

（1）任务 0：机器人的主任务。通常直接运行的程序默认为主任务，该任务始终激活，多用于负责进行机器人运动。受启动、停止、暂停命令影响。

除任务 0 外，其它任务均为 PLC 型任务，负责同步处理运算、逻辑和信号处理。

（2）任务 1、2：设置型 PLC 任务。激活是通过设置页面的“激活”按钮控制，激活后可设属性：

静态启动：上电立即启动，并一直运行，只能通过页面关闭“激活”停止。

非静态启动：上电不立即启动，与主任务一道，受启动、停止、暂停命令影响。

（3）任务 3：指令型 PLC 任务。受指令控制启停，还受停止命令控制。

多任务指令就是围绕这几种任务的启停应运而生。

```
001 START;
002 Movj P[5],V[30],Z[0];
003 Xqt 3,"tr.pro";
004 Wait In[3]==ON,T[0];
005 Delay T[2];
006 Halt 3;
007 Wait In[3]==ON,T[0];
008 Delay T[2];
009 Quit 3;
010 Wait In[3]==ON,T[0];
011 Delay T[2];
012 Pause;
```

6.12.1 Xqt

■ **功能**

启动任务。

■ **描述**

利用该指令启动主任务或指令型 PLC 任务，实现多任务程序共同运行。

■ **格式**

Xqt Index, filepath ;

■ **参数**

Index：任务号，可为 0、3。需要注意任务 0 适用于特殊应用。（详见 5.5.2 节 多任务的使用 - 特殊应用）

filepath：指令型任务关联的程序路径名，如”ABC.pro”

■ **范例**

Xqt 3, “ABC.pro”;

6.12.2 Halt

■ **功能**

暂停任务。

■ **描述**

暂停选定的任务。

■ **格式**

Halt Index;

■ **参数**

Index：任务号。取 0 和 3。

■ **范例**

Halt 3;

6.12.3 Resume

■ **功能**

恢复任务。

■ **描述**

继续启动（恢复）选定的任务。

■ **格式**

Resume Index;

■ **参数**

Index：任务号。取 0 和 3。

■ **范例**

Resume 3;

6.12.4 Quit

■ **功能**

停止并退出任务。

■ **描述**

停止并退出选定的任务。

■ **格式**

Quit Index;

■ **参数**

Index：任务号。取 0 和 3。

■ **范例**

Quit 3;

6.13 GetRunState

■ **功能**

获取系统运行状态。

■ **描述**

获取系统运行状态，值存到到 B/LB/R/LR 变量中。

■ **格式**

GetRunState(iState);

■ **参数**

iState: 利用 B/LB/R/LR 变量存储系统运行状态结果。

0- 停止

1- 启动

2- 单步前进

3- 单步后退

4- 暂停

■ **范例**

```
GetRunState(B1);
Switch B0
Case 0:
   Print “State is stop”;
Break;
Case 1:
   Print “State is run”;
Break;
Case 2 To 3:
   Print “State is step”;
Break;
Case 4:
   Print “State is pause”;
Break;
Default:
   Print “State is undefine”;
Break;
EndSwitch;
```

6.14 GetAlarmNo

■ **功能**

报警号获取。

■ **描述**

获取报警号，值存到到 B/LB/R/LR 变量中。

■ **格式**

GetAlarmNo(AlarmNo);

■ **参数**

AlarmNo: 利用 B/LB/R/LR 变量存储获取到的报警号。报警列表见附录一。

■ **范例**

```
GetAlarmNo(B1);
```

第 7 章 信息交互指令

7.1 Alarm

■ **功能**

显示自定义报警

■ **描述**

在窗口消息栏显示用户自定义的报警信息。

■ **格式**

Alarm [Index];

■ **参数**

Index：自定义报警序号（0~15）。

■ **范例**

Alarm [2];

7.2 Print

■ **功能**

打印信息。

■ **描述**

在窗口消息栏打印变量或者字符串。

■ **格式**

Print (Object)；

■ **参数**

Object：需要打印的变量或字符串（B\R\D\P\PR\LB\LR\LD\ 字符串变量或直接字符串）。

支持最多 4 个“+”将信息连起来打印。

说明：

1. 打印的内容不超过 99 个字符。

■ **范例**

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

7.3 GetPlcVar

■ **功能**

获取 PLC 传输的变量值。

■ **描述**

从 PLC 获取变量的值并存储在变量中，可获取的 PLC 变量类型：Byte、Int、DInt、LReal（参照“ITC61131-3”标准）。

7.3.1 GetPlcVarByte

■ **功能**

获取 Byte 类型的变量值。

■ **描述**

从 PLC 获取 Byte 类型变量的值并存储在 B/LB 类型的变量中。

■ **格式**

B = GetPlcVarByte(VarIndex);

■ **参数**

VarIndex：PLC Byte 类型变量的编号（0~255），可以是数字或 B/R/LB/LR 变量。

■ **返回值**

B：B/LB 变量，读取到的 PLC Byte 变量的值存储在该变量中。

■ **范例**

B1= GetPlcVarByte(3);

7.3.2 GetPlcVarInt

■ **功能**

获取 Int 类型的变量值。

■ **描述**

从 PLC 获取 Int 类型变量的值并存储在 R/LR 类型的变量中。

■ **格式**

R = GetPlcVarInt(VarIndex);

■ **参数**

VarIndex：PLC Int 类型变量的编号（0~255），可以是数字或 B/R/LB/LR 变量。

■ **返回值**

R：R/LR 变量，读取到的 PLC Int 变量的值存储在该变量中。

■ **范例**

R1= GetPlcVarInt(3);

7.3.3 GetPlcVarDInt

■ **功能**

获取 DInt 类型的变量值。

■ **描述**

从 PLC 获取 DInt 类型变量的值并存储在 R/LR 类型的变量中。

■ **格式**

R = GetPlcVarDInt(VarIndex);

■ **参数**

VarIndex：PLC DInt 类型变量的编号（0~255），可以是数字或 B/R/LB/LR 变量。

■ **返回值**

R：R/LR 变量，读取到的 PLC DInt 变量的值存储在该变量中。

■ **范例**

R1= GetPlcVarDInt(3);

7.3.4 GetPlcVarLReal

■ **功能**

获取 LReal 类型的变量值。

■ **描述**

从 PLC 获取 LReal 类型变量的值并存储在 D/LD 类型的变量中。

■ **格式**

D = GetPlcVarLReal(VarIndex);

■ **参数**

VarIndex：PLC LReal 类型变量的编号（0~255），可以是数字或 B/R/LB/LR 变量。

■ **返回值**

D：D/LD 变量，读取到的 PLC LReal 变量的值存储在该变量中。

■ **范例**

D1= GetPlcVarLReal(3);

7.4 注释

■ **功能**

输入注释。

■ **描述**

在程序中插入注释。

■ **格式**

Content

■ **参数**

Content：注释内容

■ **范例**

##Program is maded at 10.

7.5 TimeStart

■ **功能**

启动计时器。

■ **描述**

启动定时器并开始计时。

■ **格式**

TimeStart(TimerIndex)；

■ **参数**

TimerIndex：计时器编号（0~4）。

■ **范例**

```
TimeStart(0);
```

7.6 TimeOut

■ **功能**

输出计时器时长。

■ **描述**

输出计时器时长到指定变量中。

■ **格式**

TimeOut(TimerIndex, D);

■ **参数**

TimerIndex：计时器编号（0~4）

D：D/LD 变量，计时器时长（s）保存在该变量中。

注意：

1） 计时器开启后，数值一直累积，除非重新使用 TimeStart 开启；

2） 计时器是在控制器中计时，不会因为切换程序而重新计时。除非重新使用 TimeStart 开启；

3） 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;
```

7.7 GetModBusCoil

■ **功能**

获取 ModBus 从站线圈值。

■ **描述**

从 ModBus 从站获取线圈值并保存在 B/LB 类型的变量中。

■ **格式**

GetModBusCoil(StartAddr, CoilNum, B);

■ **参数**

StartAddr：线圈的起始地址（2048~4095 或 6144~8191），单位 bit。

CoilNum：需要读取的线圈个数（1~8）。

B：B/LB 变量，用于存储读取到的线圈值。

注意：

1） 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。

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

3） 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，以此类推。

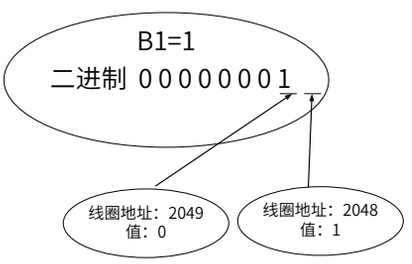
4） 当少于 8 个线圈时，多余线圈值不会读取，对应 bit 位上值取 0。

5） 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

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



7.8 SetModBusCoil

■ **功能**

设置 ModBus 从站线圈值。

■ **描述**

根据 B/LB 变量的值设置 ModBus 从站的线圈值。

■ **格式**

SetModBusCoil(StartAddr, CoilNum, B);

■ **参数**

StartAddr：线圈的起始地址（2048~4095 或 6144~8191），单位 bit。

CoilNum：需要写入的线圈个数（1~8）。

B：B/LB 变量，根据该变量的值设置 ModBus 从站的线圈值。

说明：

1） 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。

2） B/LB 变量是 BYTE 类型，占用 8bit，能设置最多 8 个线圈。

3） 变量从低位到高位设置，如 bit0 设置到 “起始地址” 的线圈，bit1 设置到 “起始地址 +1” 的线圈，以此类推。

4） 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。

5） 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。

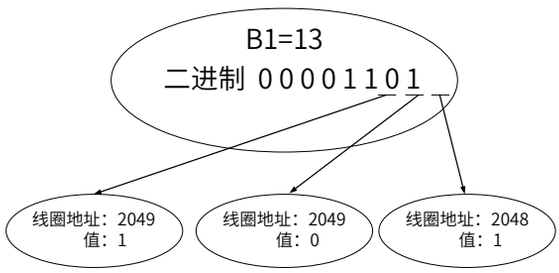
■ **范例**

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

```
B1=13;
## 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

SetModBusCoil(2048,2, B1);



7.9 GetModBusReg

■ **功能**

读取 ModBus 从站寄存器值。

■ **描述**

从 ModBus 寄存器中，按照 DataType 指定的数据类型，读取 VarNum 个该类型的数据，强制转换成变量 VarName 的数据类型，并存储于 VarName 指定的变量数组中。

■ **格式**

GetModBusReg(StartAddr,VarName,DataType,VarNum);

■ **参数**

StartAddr：寄存器的起始地址（0~65535）。

VarName：变量名。读取到的数据存储在以该变量为首的数组中。变量可以为 R 变量 /LR 变量 /D 变量 /LD 变量。

DataType：数据类型。按照以下指定的数据类型读取数据：

1：代表 short 类型，占用 2 个字节，即 1 个寄存器

2：代表 int 类型，占用 4 个字节，即 2 个寄存器

3：代表 float 类型，占用 4 个字节，即 2 个寄存器

4：代表 double 类型，占用 8 个字节，即 4 个寄存器

VarNum：读取数据的个数。范围 1-256。

注意：由于 B\R\D 变量最多只有 256 个，因此 VarNum + VarName 中的变量号 <=256

■ **范例**

GetModBusReg(16384, R1,1,2);

*## 从寄存器起始地址 16384 上读取 2 个 Short 类型数据（2*1 个寄存器），并强制转换成 R 变量类型 (int) 存储于 R1、R2 中。*

GetModBusReg(16384, R1,2,3);

*## 从寄存器起始地址 16384 上读取 3 个 int 类型数据（3*2 个寄存器），并强制转换成 R 变量类型 (int) 存储于 R1、R2、R3 中。*

GetModBusReg(16384,R1,3,3);

*## 从寄存器起始地址 16384 上读取 3 个 float 类型数据（3*2 个寄存器），并强制转换成 R 变量类型 (int) 存储于 R1、R2、R3 中。*

GetModBusReg(16384,R1,4,3);

*## 从寄存器起始地址 16384 上读取 3 个 double 类型数据（3*4 个寄存器），并强制转换成 R 变量类型 (int) 存储于 R1、R2、R3 中。*

GetModBusReg(16384,D1,1,3);

*## 从寄存器起始地址 16384 上读取 3 个 short 类型数据（3*1 个寄存器），并强制转换成 D 变量类型 (double) 存储于 D1、D2、D3 中。*

7.10 SetModBusReg

■ **功能**

设置 ModBus 从站寄存器的值。

■ **描述**

把 VarName 指定的变量数组中的数据，强制转换成 DataType 对应的数据类型，并存储于 StartAddr 指定的寄存器起始地址中。

■ **格式**

SetModBusReg(StartAddr,VarName, DataType,VarNum);

■ **参数**

StartAddr：寄存器起始地址（16384~32767 或 49152~65535）。

VarName：变量名。数据存储在以该变量为首的数组中。变量可以为 R 变量 /LR 变量 /D 变量 /LD 变量。

DataType：数据类型。按照以下指定的数据类型转换数据：

1：代表 short 类型，占用 2 个字节，即 1 个寄存器

2：代表 int 类型，占用 4 个字节，即 2 个寄存器

3：代表 float 类型，占用 4 个字节，即 2 个寄存器

4：代表 double 类型，占用 8 个字节，即 4 个寄存器

VarNum：读取数据的个数。范围 1-256。

注意：由于 B\R\D 变量最多只有 256 个，因此 VarNum + VarName 中的变量号 <=256

■ **范例**

SetModBusReg(16384, R1,1,2);
*## 把 R1、R2 的值强制转换成 short 类型存储在寄存器起始地址 16384 到 16384 + 2*1 的地址上。*
SetModBusReg(16384, R1,2,3);
*## 把 R1、R2、R3 的值强制转换成 int 类型存储在寄存器起始地址 16384 到 16384 + 3*2 的地址上。*
SetModBusReg(16384,R1,3,3);
*## 把 R1、R2、R3 的值强制转换成 float 类型存储在寄存器起始地址 16384 到 16384 + 3*2 的地址上。*
SetModBusReg(16384,R1,4,3);
*## 把 R1、R2、R3 的值强制转换成 double 类型存储在寄存器起始地址 16384 到 16384 + 3*4 的地址上。*

7.11 GetCommVal

■ **功能**

读取共享内存公共区的值。

■ **描述**

从指定的偏移地址开始读取共享内存公共区的若干变量值并保存到指定的变量中。

■ **格式**

GetCommVal(OffsetAddr, &B, Num);

■ **参数**

OffsetAddr：公共区相对于起始地址的偏移量（0~8191），单位 BYTE。

&B：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。

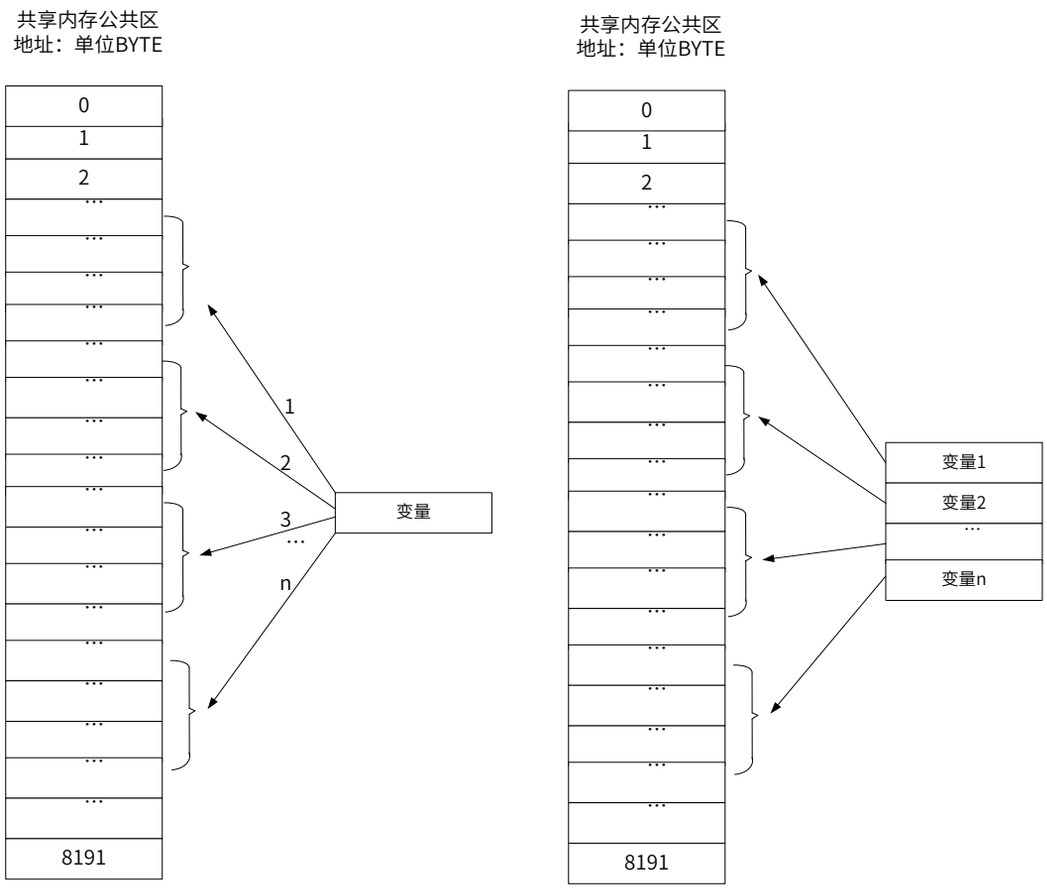
Num：读取变量的个数（0~255）。

说明：

1） 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。

2） 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);



3） 变量的类型不同，会占据公共区地址大小也不同。详见下表“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);                    ## 结果 LR1=45，LR2=45
GetCommVal (8000, LR1, 2);
R1=46;
R2=47;
SetCommVal (8000,&R1, 2);                   ## 结果 LR1=46，LR2=47
GetCommVal (8000, LR1, 2);
CString str1 = “abc”;
SetCommVal (500,str1, 3);                   ## 结果 str1=”ab”
GetCommVal (500, &str1, 2);
```

7.12 SetCommVal

■ **功能**

设置公共区指定地址的值。

■ **描述**

根据参数 2 的值设置共享内存公共区的值。

■ **格式**

SetCommVal (OffsetAddr, &B, Num);

■ **参数**

OffsetAddr：公共区相对于起始地址的偏移量（0~8191），单位 BYTE。

&B：数值（-2147483648~2147483647）/B/LB/R/LR/D/LD/P/PR/LPR/ 字符串变量，或者 &B/&R/&D/&PR，需要写入到共享内存公共区的值。

Num：写入变量的个数（0~255）。

说明：

1） 详情参见 GetCommVal 说明。

2） 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。

视觉指令是控制器与外部视觉设备在通讯时所用的指令，可以通过网口或串口通信。

网口通信，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；但需设置一个端口号，并在视觉设备上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 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;
PR0=(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;
Send Port[1026],TxBuf;
L[1]:
Get Port[1026], T[10], Goto L[1];
RxBuf = GetPorbuf(0,100,1026);
B1 = StrGetData(RxBuf,”#”,P10);
Cnvrt(P[10],P[20],World]);
Movl Offset(P[20],PR0),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:
```

(打开)

↓

发送请求

↓

接收数据

↓

处理接收数据
并控制运动

↓

(关闭)

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];
Send Port[1026],TxBuf;
L[1]:
Get Port[1026], T[10], Goto L[1];
RxBuf = GetPorbuf(0,100,1026);
B1 = StrGetData(RxBuf,”#”,P10);
Cnvrt(P[10],P[20],World]);
Movl Offset(P[20],PR0),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;
```

对于串口通信，插上串口即可直接编程使用，示例如下：

```
START;
B0 =0;
While  B0 <> 1
Open Com(1,19200,B0) ;
EndWhile;
Send Port[1],"TA";
L[1]:
Get Port[1],T[0],Goto L[1];
Str[1] =GetPortbuf(0,100,1);
Print Str[1];
Close Com,1;
END;
```

对于跟随工艺，可先使用 CnvVision 指令打开传送带的视觉端口，视觉端口开启后视觉数据将自动传送到控制器并处理，获取传送带上指令物体的数据后实现动态跟随及抓取动作，抓取成功后关闭传送带的视觉端口。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];
EndIf;
CnvVision (Conveyor[1],ON);
## 定义 P[30] 为在 1 号传送带物体的正上方 10mm 处。注意坐标系号为 7，为一个跟随传动带运动的点。
P[30]=(0,0,10,0,0,0),(0,0,0,0),(7,0,1);
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;
```

第 8 章 视觉指令

8.1 Open

8.1.1 Open Socket

■ **功能**

连接到服务器。

■ **描述**

指定服务器 IP 地址及端口号，连接到服务器。

■ **格式**

格式 1：Open Socket(IP, SvrPort, ClientPort);

格式 2：Open Socket(IP, SvrPort, ClientPort,B);

■ **参数**

IP：服务器 IP 地址。

SvrPort：服务器端口号（1024~9999，其中 3333 不可使用）。

ClientPort：客户端端口号（1024~9999，其中 3333 不可使用）。

B：（可选项）B/LB 变量，返回值（打开成功返回 1，否则返回 0）。

说明：

1） Open Socket 用于本地控制器作客户端时与外部通讯使用。

2） 客户端与服务器不得使用相同的端口号

3） Open Socket 一旦启动，端口会一直打开；除非使用 close 或手动打开其它程序时自动关闭。

4） 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;
```

8.1.2 Open Com

■ **功能**

连接到串口。

■ **描述**

指定串口号和波特率，连接到指定串口。

■ **格式**

格式 1：Open Com(Port, Baudrate);

格式 2：Open Com(Port, Baudrate,B);

■ **参数**

Port：串口号，可为 1-15。对于 IMC100 控制器，只有 1 个串口，该项不起作用。

Baudrate：波特率，可选 9600、19200、38400、43000、56000、57600、115200。

B：（可选项）B/LB 变量，返回值（打开成功返回 1，否则返回 0）。

■ **说明**

1） Open Com 一旦启动，端口会一直打开；除非使用 close 或手动打开其它程序时自动关闭。

2） 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 Com(1, 9600,LB1);
EndWhile;
```

8.2 Close

8.2.1 Close Socket

■ **功能**

关闭以太网端口。

■ **描述**

关闭打开的以太网端口，切断以太网连接。

■ **格式**

Close Socket,ClientPort;

■ **参数**

ClientPort：待关闭的客户端端口号（1024~9999，其中 3333 不可使用）。

说明：Close Socket 关闭客户端端口号为当前值的连接。

■ **范例**

```
Close Socket,1025;
```

8.2.2 Close Com

■ **功能**

关闭串口。

■ **描述**

关闭打开的串口，切断串口连接。

■ **格式**

Close Com, Port;

■ **参数**

Port：串口号可为 1-15。对于 IMC100 控制器，只有 1 个串口，该项不起作用，调用 Close 即会关闭唯一的串口连接。

■ **范例**

```
Close Com,1;
```

8.3 GetSocketNo

■ **功能**

获取通讯的客户端端口号。

■ **描述**

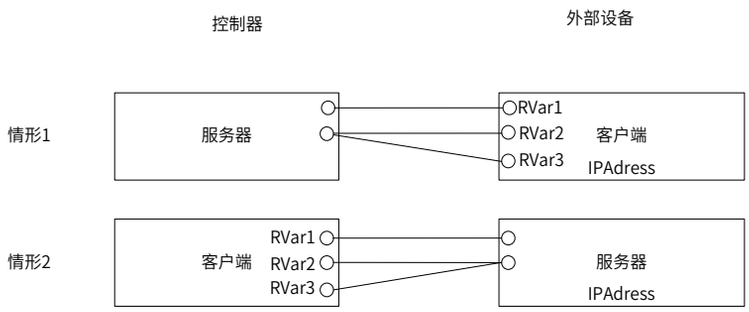
在控制器与外部设备的连接中，指定外部设备的 IP 地址后，就能获取客户端端口号。需要注意两点。

1） 获取总是客户端的端口号。

当控制器作为服务器、外部设备作为客户端时，获取的是此连接中外部设备的端口号；

当控制器作为客户端、外部设备作为服务器时，获取的是此连接中控制器的端口号。

2） 当控制器与外部设备有多个连接时，则能获取这些所有连接的客户端端口号。



■ **格式**

GetSocketNo(IPAdress , RVar);

■ **参数**

IPAdress: 外部设备的 IP 地址

RVar： R/LR 变量，用于存储获取的客户端的 IP 地址。当控制器与外部设备有多个连接时，则客户端上所有连接的端口号，存储在以 R/LR 变量为起始地址的变量数组中

■ **返回值**

返回连接的客户端的个数

■ **范例**

```
B1 = GetSocketNo(“192.168.23.100”,R3);
If(B1>0)
## 取第一个连接
Get Port[R3],T[1],Goto L[1];                    ## 获取数据
Endif;
```

8.4 GetPortbuf

■ **功能**

获取接收缓冲区的字符串。

■ **描述**

从接收缓冲区指定位置开始读取指定数量的字符并存储到指定的字符串变量中。

■ **格式**

StrVar = GetPortbuf(StartBit , Num, Port);

■ **参数**

StartBit：数字 /B/LB/R/LR 变量（0~1023），接收缓冲区的起始读取位置。

Num：数字 /B/LB 变量（0~100），读取的字符个数。

Port：当使用客户端 - 服务器通信时，此 Port 为客户端端口号，应满足 1024-9999，其中 3333 不可使用，因为它为系统占用。当使用串口通信时，此 Port 为串口号，应满足 0-15。然而由于该指令可配对上述两种通信，因此利用示教器输入时，只限制输入范围 0-9999。

■ **返回值**

StrVar：保存从接收缓冲区读取的字符串。

说明：在接收缓冲区中从某一位 StartBit 开始取连续 Num 个字符，并传给指定的字符串变量。失败时传 0。

■ **范例**

```
## 从第 2 位开始，取 3 个字符
L[2]:
Get Port[1025],T[1],Goto L[2];
Str1 = GetPortbuf(2,3,1025);
```

8.5 Send Port

■ **描述**

将字符串发送到网络的远端端口。

■ **格式**

格式 1：Send Port[Port],String;

格式 2：Send Port[Port],String,Type;

■ **参数**

Port：当使用客户端 - 服务器通信时，此 Port 为客户端端口号，应满足 1024-9999，其中 3333 不可使用，因为它为系统占用。当使用串口通信时，此 Port 为串口号，应满足 0-15。然而由于该指令可配对上述两种通信，因此利用示教器输入时，只限制输入范围 0-9999。

String：发送的字符串。

Type：发送数据类型，可以为 String/Binary/Hex。没有此参数时，默认为以字符串形式发送，相当于使用 String。

说明：

1） 当使用客户端 - 服务器通信时，既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。但这里的端口号恒为客户端端口号。

2） 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。

3） 当使用串口通信时，此 Port 为串口号 .

■ **范例**

```
Send Port[1025],”ABC”;                  ## 直接发送字符串”ABC”
Str[1] = “ABC”;                         ## 发送全局字符串变量
Send Port[1025],Str[1];
String ss = “ABC”;                      ## 发送局部字符串变量
Send Port[1025], ss;
```

8.6 Get Port

■ **功能**

从远端端口接收数据。

■ **描述**

在一定时间内，接收远端端口的数据，放入缓冲区。当超过指定的时间而未接收到数据，则跳转到指定的标签处。

■ **格式**

Get Port[Port],T[Time],Goto L[Index];

■ **参数**

Port：当使用客户端 - 服务器通信时，此 Port 为客户端端口号，应满足 1024-9999，其中 3333 不可使用，因为它为系统占用。当使用串口通信时，此 Port 为串口号，应满足 0-15。然而由于该指令可配对上述两种通信，因此利用示教器输入时，只限制输入范围 0-9999。

Time：等待时间（0~65535），单位 s。

Index：标签号，当等待超时时跳转到的标签号。

说明：

1） 当使用客户端 - 服务器通信时，既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。但这里的端口号恒为客户端端口号。

2） 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。

3） 当使用串口通信时，此 Port 为串口号。

4） 在 T[Time] 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[Index]）；若未接收到数据，则执行该句的 Goto L[Index]，即跳转至 L[Index] 处。

■ **范例**

```
Get Port[1025],T[1],Goto L[2];
Str1 = GetPortbuf(2,3,1025);
```

8.7 传送带系列

8.7.1 CnvVision

■ **功能**

打开 / 关闭传送带的视觉端口。

■ **描述**

打开 / 关闭指定传送带的视觉端口，该语句要在 Open 指令之后，GetCnvObject 前（打开视觉端口）后（关闭视觉端口）用到。

■ **格式**

CnvVision(Conveyor[Index],ON|OFF,ClientPort);

■ **参数**

Index：传送带编号（0~3）。

ON|OFF：打开或者关闭视觉端口

ClientPort：客户端端口号。

■ **范例**

```
CnvVision (Conveyor[0],ON,1024);
GetCnvObject(0,0), Goto L[0];
……
CnvVision (Conveyor[0],OFF,1024);
```

8.7.2 GetCnvObject

■ **功能**

接收传送带上物体的视觉数据。

■ **描述**

从指定的传送带上获取指定类型物体的数据，对象在队列中被删除。

■ **格式**

GetCnvObject(CnvID, ObjID),Goto L[Index];

■ **参数**

CnvID：传送带编号（0~3）。

ObjID：物体类型编号（0~15）。

Index：标签号，指定时间内未接收成功跳转到的标签号，接收成功则跳转到下一行。

■ **说明**

1） 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至 DSP 处理，直至关闭视觉端口。

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

3） 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。

注意

1. 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。

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),(0,0,0,0,0),(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,1026;
END;
```

8.7.3 CopyCnvObject

■ **功能**

复制传送带上物体的视觉数据。

■ **描述**

复制指定传送带上指定类型的物体对象，对象在队列中仍然保留。

■ **格式**

CopyCnvObject(CnvID, ObjID),Goto L[Index];

■ **参数**

CnvID：传送带编号（0~3）。

ObjID：物体类型编号（0~15）。

Index：标签号，指定时间内未接收成功跳转到的标签号，接收成功，跳转至下一行。

■ **范例**

```
CopyCnvObject(1, 0),Goto L[1];
```

8.7.4 ClearCnvObject

■ **功能**

传送带物体删除。

■ **描述**

删除指定传送带上的物体，可以删除若干个或者全部删除。被检测到的传送带物体在程序暂停、停止时不会自动清除，若需要清除需要通过该指令编程实现。

■ **格式**

ClearCnvObject(CnvID，Num|ALL);

■ **参数**

CnvID：传送带编号（0~3）。

Num|ALL：要删除的物体个数，可以是 Num 个，如果为 ALL，则全部删除。

■ **范例**

```
ClearCnvObject(1，2);                    ## 删除传送带 1 上的 2 个对象数据
ClearCnvObject(1，ALL);                  ## 删除传送带 2 上的全部对象数据
```

第 9 章 其他指令

9.1 USING MAIN

■ **功能**

声明调用主程序的位置变量。

■ **描述**

在子程序中声明主程序中的 P 及其后 VarNum 个位置变量，以便调用。

■ **格式**

USING MAIN,P,VarNum；

■ **参数**

P：主程序中位置变量。

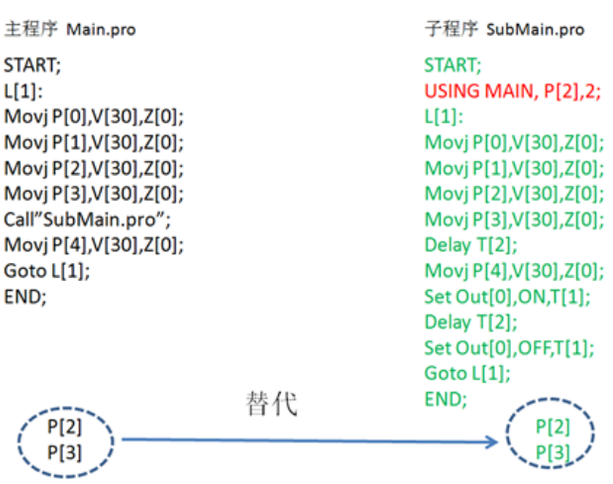
VarNum：从 P 开始要使用的连续变量的个数（1~9999）。

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：



9.2 锁螺丝系列

9.2.1 LoadScrewParm

■ **功能**

加载螺丝工艺参数至伺服。

■ **描述**

加载某个产品所使用的螺丝工艺文件到伺服控制器。

■ **格式**

LoadScrewParm(TechFile，B);

■ **参数**

TechFile：锁付某个产品时所使用的螺丝工艺文件。

B：B/LB 变量，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。

9.2.2 LockScrew

■ **功能**

螺丝锁付。

■ **描述**

完成螺丝的锁付操作。

■ **格式**

LockScrew(TechID，P，V[Vel]);

■ **参数描述**

TechID：锁螺丝工艺号，即锁螺丝所使用的工艺参数号（0~15）。

P：锁付的位置变量编号。

Vel：滑台下降速度，用户根据节拍自设。

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];
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;
```

9.2.3 CheckLock

■ **功能**

螺丝锁付结果检测。

■ **描述**

在螺丝锁付完成后对锁付结果进行检测。

■ **格式**

CheckLock(B);

■ **参数描述**

B：B/LB 变量，锁付结果检测指令的返回值，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 指令后再执行 CheckLock。

9.2.4 UnLockScrew

■ **功能**

松螺丝。

■ **描述**

从指定的位置拆除一个螺丝。

■ **格式**

UnLockScrew(TechID，P，V[Vel]，RetreatDist，UnlockVel);

■ **参数描述**

TechID：松螺丝工艺号，即拆螺丝所使用的工艺参数号（0~15）。

P：松螺丝的起始位置编号。

Vel：滑台下降的速度，用户可根据节拍自设。

RetreatDist：松螺丝时滑台的回退距离，单位 mm。

UnlockVel：松螺丝速度，单位 mm/s。

说明：

1） 拆卸过程的工艺参数均能通过工艺号选取；

2） 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] 到位之后就开始执行。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] 点后就开始边拆边回退，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。

```
START;
LoadScrewParm(“aa.stp”,B0);
Movj P[0],V[30],Z[0];
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];
Set Out[7],OFF;
Delay(0.1);
END;
```

9.2.5 CheckUnLock

■ **功能**

松螺丝结果检测。

■ **描述**

检测松螺丝结果，保存在指定的变量中。

■ **格式**

CheckUnLock(B);

■ **参数**

B：B/LB 变量，松螺丝结果检测指令返回值，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。

h) ScrewStop

■ **功能**

停止锁螺丝或松螺丝。

■ **描述**

用户根据一些报警信息执行电批停止指令。

■ **格式**

ScrewStop;

9.3 干涉区系列

通过指令可设置额外 8 个干涉区，相对于设置界面中配置的干涉区，他们是独立的，可以与界面设置的干涉区同时生效、互不影响。

通过指令 WZBoxDef、WZSphDef、WZCylDef、WZLimJDef 可分别将干涉区定义为矩形、球形、圆柱形、关节型。WZDOSet 指令定义干涉后的动作，同时激活干涉区，开启干涉检查。WZEnable、WZDisable 分别用来激活、禁用干涉区。

注意：进入干涉区后，由于减速需要一个过程，不能立即停止，建议设置的干涉区比实际大一些。

9.3.1 WZBoxDef

■ **功能**

定义矩形干涉区。

■ **描述**

利用长方体的对角两点，定义基坐标系下的矩形干涉区。

■ **格式**

WZBoxDef Box[AreaID], nType,P1,P2;

■ **参数**

AreaID：干涉区号，整型数据（0~7）。

nType：干涉区类型，0 表示干涉区为内部，1 表示干涉区为外部。

P1、P2：矩形体的对角端点。

9.3.2 WZSphDef

■ **功能**

定义球形干涉区。

■ **描述**

利用球的球心和半径，定义基坐标系下的球形干涉区

■ **格式**

WZSphDef Box[AreaID], nType,P,Radius;

■ **参数**

AreaID：干涉区号，整型数据（0~7）。

nType：干涉区类型，0 表示干涉区为内部，1 表示干涉区为外部。

P：球形干涉区的球心位置变量。

Radius：double 型数据，定义干涉区球体的半径。

9.3.3 WZCylDef

■ **功能**

定义圆柱干涉区。

■ **描述**

利用圆柱的底面圆心、半径和圆柱高度，定义基坐标系下的圆柱干涉区。

■ **格式**

WZCylDef Box[AreaID], nType,P,Radius,Height;

■ **参数**

AreaID：干涉区号，整型数据（0~7）。

nType：干涉区类型，0 表示干涉区为内部，1 表示干涉区为外部。

P：圆柱形干涉区的底面圆心位置变量。

Radius：double 型数据，定义圆柱形干涉区的底面圆半径。

Height：double 形数据，定义圆柱形干涉区的高度。

9.3.4 WZLimJDef

■ **功能**

定义关节干涉区。

■ **描述**

定义关节坐标系下的干涉区

■ **格式**

WZLimJDef Box[AreaID], nType,PR1,PR2;

■ **参数**

AreaID：干涉区号，整型数据（0~7）。

nType：干涉区类型，0 表示干涉区为内部，1 表示干涉区为外部。

PR1、PR2：关节极限的一小一大两个量，平移变量里面是各关节角度。其中如果两个平移变量的关节角一致，则该关节不做限制，比如：都为 0。

9.3.5 WZDOSet

■ **功能**

定义干涉后的动作并激活干涉区。

■ **描述**

定义干涉后的动作，如干涉后置变量、置信号或是产生报警等等。同时它也会激活一个干涉区。

■ **格式**

WZDOSet Box[AreaID],ActionValue;

■ **参数**

AreaID：干涉区号，整型数据（0~7）。

ActionValue：有以下三种形式：

B：发生干涉后，指定的 B 变量值置 1，但不会停止运动，未干涉保持原来 B 值。

Out：发生干涉后，指定的 Out 置 ON，，但不会停止运动，未干涉保持原来 Out 值。

Alarm: 发生干涉后，触发一个自定义报警，停止运动并掉使能。

注意：

1） WZDOSet 指令定义干涉动作的同时，也会默认激活这个干涉区，相当于启用了 WZEnable 指令。

2） 当激活了干涉区，但未使用 WZDOSet 定义干涉动作时，默认在发生干涉时使用用户自定义报警 0。

■ **范例**

WZDOSet Box[1],B2;
WZDOSet Box[1],Out[3];
WZDOSet Box[1],Alarm[4];

9.3.6 WZDisable

■ **功能**

禁用干涉区。

■ **描述**

不再激活指定的干涉区，关闭此干涉区的干涉检查。

■ **格式**

WZDisableBox[AreaID];

■ **参数**

AreaID：干涉区号，整型数据（0~7）。

9.3.7 WZEnable

■ **功能**

激活干涉区

■ **描述**

激活指定的指令型干涉区，开启此干涉区的干涉检查。

■ **格式**

WZEnable Box[AreaID];

■ **参数**

AreaID：干涉区号，整型数据（0~7）。

注意：当激活了干涉区，但未使用 WZDOSet 定义干涉动作时，默认在发生干涉时使用用户自定义报警 0。

9.4 动力学系列

9.4.1 Load Obj

■ **功能**

加载工件动力学参数。

■ **描述**

加载工件动力学参数。

■ **格式**

Load Obj[ObjID];

■ **参数**

ObjID：数字 /B/R/LB/LR 变量，工件号（0~15）。

9.4.2 UnLoad Obj

■ **功能**

卸载工件动力学参数。

■ **描述**

卸载工件动力学参数。

■ **格式**

UnLoad Obj[ObjID];

■ **参数**

ObjID：数字 /B/R/LB/LR 变量，工件号（0~15）。

9.4.3 etObjMass

■ **功能**

设置工件的质量。

■ **描述**

设置工件的质量。

■ **格式**

SetObjMass(ObjID，ObjMess);

■ **参数**

ObjID：数字 /B/R/LB/LR 变量，工件号（0~15）。

ObjMess：double/B/R/LB/LR/D/LD 变量，工件质量，单位 kg。

9.4.4 SetObjCog

■ **功能**

设置工件的质心。

■ **描述**

设置工件的质心。

■ **格式**

SetObjCog(ObjID，dx,dy,dz);

■ **参数**

ObjID：数字 /B/R/LB/LR 变量，工件号（0~15）。

dx,dy,dz：double/B/R/LB/LR/D/LD 变量，质心偏移，单位 mm。

9.4.5 SetObjOrient

■ **功能**

设置工件的姿态

■ **描述**

设置工件的姿态。

■ **格式**

SetObjOrient(ObjID，A,B,C);

■ **参数**

ObjID：数字 /B/R/LB/LR 变量，工件号（0~15）。

A,B,C：double/B/R/LB/LR/D/LD 变量，工件的姿态，单位°。

9.4.6 SetObjInertia

■ **功能**

设置工件的转动惯量。

■ **描述**

设置工件的转动惯量。

■ **格式**

SetObjInertia(ObjID，Ix,Iy,Iz);

■ **参数**

ObjID：数字 /B/R/LB/LR 变量，工件号（0~15）。

Ix,Iy,Iz：double/B/R/LB/LR/D/LD 变量，绕 x,y,z 轴的转动惯量，单位 (***)。

9.4.7 SetToolMass

■ **功能**

设置工具的质量。

■ **描述**

设置工具的质量。

■ **格式**

SetObjMass(ToolID，ToolMess);

■ **参数**

ToolID：数字 /B/R/LB/LR 变量，工具号（0~15）。

ToolMess：double/B/R/LB/LR/D/LD 变量，工具质量，单位 kg。

9.4.8 SetToolCog

■ **功能**

设置工具的质心。

■ **描述**

设置工具的质心。

■ **格式**

SetToolCog(ToolID，dx,dy,dz);

■ **参数**

ToolID：数字 /B/R/LB/LR 变量，工具号（0~15）。

dx,dy,dz：double/B/R/LB/LR/D/LD 变量，工具质心偏移，单位 mm。

9.4.9 SetToolOrient

■ **功能**

设置工具的姿态。

■ **描述**

设置工具的质心。

■ **格式**

SetToolOrient(ToolID，A,B,C);

■ **参数**

ToolID：数字 /B/R/LB/LR 变量，工具号（0~15）。

A,B,C：double/B/R/LB/LR/D/LD 变量，工件的姿态，单位°。

9.4.10 SetToolInertia

■ **功能**

设置工具的转动惯量。

■ **描述**

设置工具的转动惯量。

■ **格式**

SetObjInertia(ToolID，A,B,C);

■ **参数**

ToolID：数字 /B/R/LB/LR 变量，工具号（0~15）。

A,B,C：double/B/R/LB/LR/D/LD 变量，转动惯量，单位 (***)。

9.5 位置锁存系列

位置锁存功能是指：

利用 Out[15] 信号触发系统保存此时机器人的位置，在需要使用时调用相应指令取用该位置。

该功能在飞拍中有应用，位置锁存开启后，在运动过程中使用 Out[15] 信号，同时触发拍照和记录当前位置，在执行后续运动的过程中，依据拍到信息，同步计算出应到达的最终位置，并最终运动到此点。

注意：

触发位置锁存的信号固定为 Out[15]，在使用指令控制时应特别注意。

对于 IRCB300 系列电柜，支持位置锁存功能。用户只需将 Out[15] 接到相机上作为触发拍照使用。

对于对于 IRCB10 系列电柜，不支持该功能。

9.5.1 LatchEnable

■ **功能**

位置锁存功能开启 / 关闭

■ **描述**

位置锁存开启后，系统监控指定的信号（Out[15]）变化，当检测到由 OFF 变为 ON 时，机器人位置被锁存下来。锁存完成后可通过 GetLatchPos 读取被锁存的位置坐标。

■ **格式**

LatchEnable ON|OFF;

■ **参数**

ON|OFF：ON 开启，OFF 关闭

■ **范例**

```
START;
LatchEnable ON;
ClearLatchPos;
Movl P[0],V[30],Z[0];
Movl P[1],V[30],Z[0],NWait,Out(15,OFF,D[0]),Out(15,ON,D[50]);
B1=0;
While B1==0
        B1 =GetLatchPos(P[10],2,0,0);
EndWhile;
Print P[10];
LatchEnable OFF;
End;
```

注意：

在使用 LatchEnable ON 开启并使用完成后，需 LatchEnable OFF 关闭。

9.5.2 ClearLatchPos

■ **功能**

锁存位置清除。

■ **描述**

将之前锁存的机器人位置清除。

■ **格式**

ClearLatchPos;

■ **范例**

参考 LatchEnable

9.5.3 GetLatchPos

■ **功能**

锁存位置读取。

■ **描述**

读取最后一次锁存的机器人位置

■ **格式**

GetLatchPos(P, CoordID,ToolID,UserID);

■ **参数**

P：位置变量，获取结果存储在位置变量中。

CoordID：获取的位置变量的坐标系号（1~4）。

ToolID：获取的位置变量的工具号（0~15）。

UserID：获取的位置变量的用户号（0~15）。

■ **返回值**

无锁存数据 0，有锁存数据 1，当返回 1 时 P 的值有效。

范例：

参考 LatchEnable

维护篇

维护篇 - 目录

第 1 章 机器人的安全

1.1 安全等级定义

在本手册中有三种安全等级定义。

危 险	“危险”表示如果不按规定操作，则导致死亡或严重身体伤害。
警 告	“警告”表示如果不按规定操作，则可能导致死亡或严重身体伤害。
注 意	“注意”表示如果不按规定操作，则可能导致轻微身体伤害或设备损坏。

1.2 安全注意事项

系统设计时
危 险
◆ 请务必对系统设置安全护栏，防止人员进入系统的动作区域内，否则可能造成严重的安全问题。 ◆ 设计和制造末端执行器时，请确保其不会因为动力变化而产生危险。
开箱验收时
注 意
◆ 开箱前请检查设备的外包装是否完好 , 有无破损、浸湿、受潮、变形等情况。 ◆ 请按照层次顺序打开包装，严禁猛烈敲打！ ◆ 开箱时请检查设备及附件表面有无残损、锈蚀、碰伤等情况。 ◆ 开箱后请仔细对照装箱清单，查验设备及附件数量、资料是否齐全。
警 告
◆ 开箱时发现设备及附件有损伤、锈蚀、使用过的迹象等问题，请勿安装！ ◆ 开箱时发现设备有内部进水、部件缺少或部件损坏的情况时，请勿安装！ ◆ 请仔细对照装箱清单，发现装箱清单与设备名称不符时，请勿安装！ ◆ 请按照包装箱指示的开箱方向进行开箱。
储存与运输时
危 险
◆ 请由具有资格的作业人员进行司索、起重机起吊作业或叉车驾驶等搬运作业，否则可能造成重伤或重大损害。
警 告
◆ 请尽可能在原包装状态下用吊车和叉车等进行搬运。 ◆ 使用吊车、起重机等搬运设备时，作业者需穿戴个人防护装置，搬运路线周围禁止人员站立或停留。 ◆ 吊起设备时，请用手扶住以确保平衡，起吊不稳可能会导致设备掉落，造成重伤或重大损害。
注 意
◆ 请按照设备的储存与运输条件进行储存与运输，储存温度、湿度满足要求。 ◆ 避免在水溅雨淋、阳光直射、强电场、强磁场、强烈振动等场所储存与运输。 ◆ 避免设备储存时间超过 3 个月，储存时间过长时，请进行更严密的防护和必要的检验。 ◆ 请将设备进行严格包装后再进行车辆运输，长途运输时必须使用封闭的箱体。 ◆ 严禁将本设备与可能对本设备构成影响或损害的设备或物品一起混装运输。 ◆ 在运输含锂电池的设备时，必须遵守运输规定。

安装时

危 险

- 严禁非专业人员进行设备安装、接线、保养维护、检查或部件更换！
- 请务必对系统安装安全护栏，否则可能造成严重的安全问题。
- 安装系统时，请勿与周围的建筑物、结构件或设备等产生干扰，否则可能会因工具或工件撞到外围设备造成重伤或重大损害。

警 告

- 严禁改装本设备！
- 请勿在强电场或强电磁波干扰的场所安装本设备！
- 拆卸设备安装螺钉，请扶住设备防止设备翻倒。

接线时

危 险

- 严禁非专业人员进行设备安装、接线、保养维护、检查或部件更换！
- 请务必断开电源后进行接线作业，否则可能会有触电的危险或导致系统故障。
- 接线前，请切断所有设备的电源。切断电源后设备内部电容有残余电压，请至少等待 10 分钟再进行接线等操作。
- 接线时，请务必保证紧急停止开关和安全门等安全相关输入信号正确接入，否则系统发生紧急状况时将无法发挥安全功能，可能会导致重伤或重大损害。
- 请务必保证设备的良好接地，否则会有电击的危险。
- 请遵守静电防止措施（ESD）规定的步骤，并佩戴静电手环进行接线等操作，避免损坏设备内部的电路。

警 告

- 请将线缆连接牢固。请勿在电缆上放置重物，请勿强行弯曲或拉拽电缆，否则可能造成电缆损坏、断线或接触不良，有触电的危险或导致系统故障。
- 接线时使用到的线缆必须符合相应的线径和屏蔽等要求，使用屏蔽线缆时屏蔽层需要单端可靠接地！
- 接线时请勿弄错连接关系，否则系统将无法正常工作，还可能造成安全问题。
- 接线完成后，请确保设备内部没有掉落的螺钉或裸露线缆。

操作时

危 险

- 操作前，请确认紧急停止开关与安全门开关工作正常，否则系统发生紧急状况时将无法发挥安全功能，可能会导致重伤或重大损害。
- 请在规定的环境条件下使用系统，否则可能会缩短设备的使用寿命，还可能会造成严重的安全问题。
- 请在规格范围内使用系统，否则可能会缩短设备的使用寿命，还可能会造成严重的安全问题。

警 告

- 操作前，请检查是否正确安装系统、是否正确连接线缆和气源、是否正确与周边设备进行连接。
- 操作前，请确认安全护栏内侧没有人。系统动作期间，请勿进入其动作区域内，否则可能造成严重的安全问题。
- 请勿随意更改出厂设置。
- 如果在操作期间系统进行异常动作，请立即按下紧急停止开关，否则可能产生严重的安全问题。
- 系统正常工作时，请勿通过断开电源的方法来停止系统或随意按下紧急停止开关，否则容易造成设备损坏。

保养和维修时
警 告 ◆ 请按照产品保修协议进行设备报修。 ◆ 请按照设备维护和保养要求对设备进行日常和定期检查与保养，并做好保养记录。 ◆ 设备出现故障或损坏时，由专业人员按照维修指导对设备进行故障排除和维修，并做好维修记录。 ◆ 请按照手册中的更换指导进行部件更换。 ◆ 维护时请勿使异物进入到设备内部与连接端子中。 ◆ 除进行保养和维护作业时以外，请不要打开设备的盖子。 ◆ 更换设备后，请务必重新对设备接线检查与参数设置。
报废时
警 告 ◆ 请按照国家有关规定与标准进行设备的报废，以免造成财产损失或人员伤亡！ ◆ 报废的设备请按照工业废弃物处理标准进行处理回收，避免污染环境。

1.3 警告标签

机器人本体上粘贴有下述警告标签。在粘贴这些标签的位置附近存在特有的危险，操作时请充分注意。为了安全地操作、维护机器人系统，请务必遵守警告标签上记载的注意与警告内容。

请勿损坏、损伤或剥下这些标签。

警告标签	安全说明
	由于机器人重心前置，为防止机器前倾造成机器损坏或人身危险，请将机器人固定后再拆除底座安装螺钉。
	机器人运转期间，请勿进入到动作区域内。否则可能会撞到机器人，造成严重的安全问题，非常危险
	如果通电期间触摸内部通电部分，可能会有触电的危险。

-539-

1.4 紧急移动和紧急停止操作

1） 关于紧急移动

当系统处于紧急模式时，可以用手移动关节臂，具体方法如下：

警 告

◆ 按下制动解除开关开关时，请注意因末端执行器自重而产生下垂！

■ IRS100 系列

一轴 用手顺、逆时针旋转

二轴 用手顺、逆时针旋转

三轴 按住制动解除开关，用手推动丝杠上下滑动

四轴 按住制动解除开关，用手推动丝杠顺、逆时针旋转

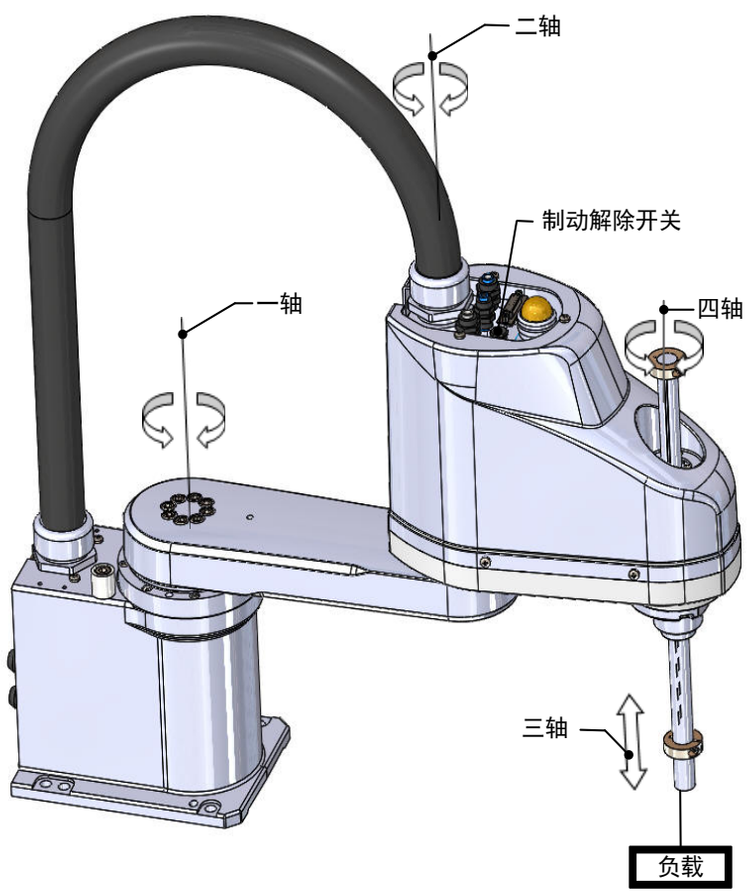


图 1-15 IRS100 系列制动解除开关位置示意图

-540-

■ IRS100-20 系列

一轴 用手顺、逆时针旋转

二轴 用手顺、逆时针旋转

三轴 按住制动解除开关，用手推动丝杠上下滑动

四轴 按住制动解除开关，用手推动丝杠顺、逆时针旋转

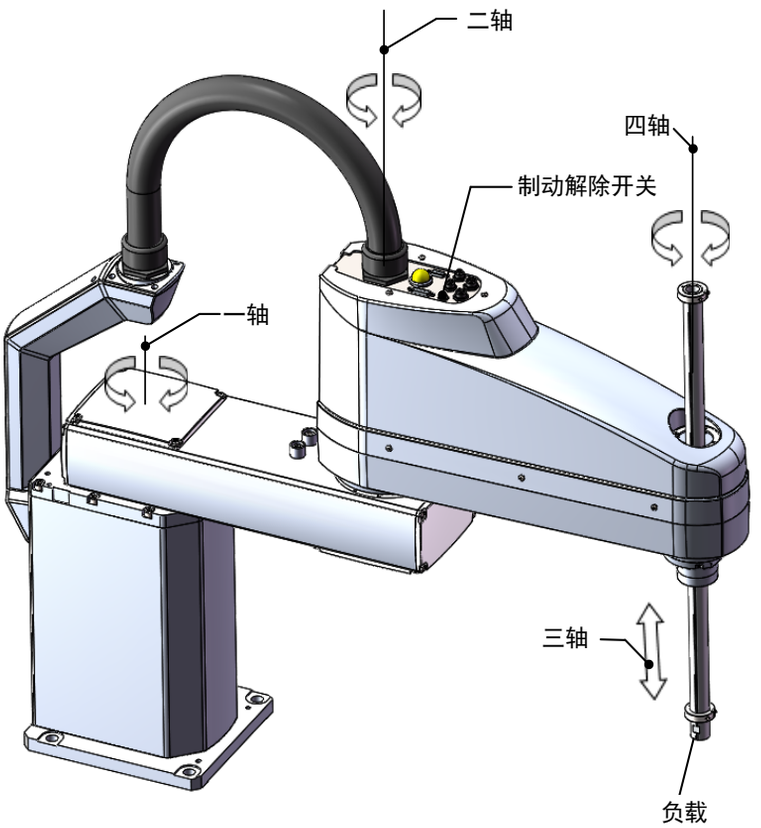


图 1-16 IRS100-20 系列制动解除开关位置示意图

2） 关于紧急移动

a）当机器人在工作过程中有异常或是突发紧急情况需要停止时，请立即按下急停开关。当按下开关时，电机电源被切断，制动系统使机器人在最短距离内停止动作。

b）机器人正常运动时，不建议使用急停开关，一方面，频繁的刹车会使电机抱闸寿命缩短，另一方面，机器人正常运行轨迹被破坏，有可能动作执行出错，严重时还可能撞击到附近的物体。

c）非紧急情况时的停止，请按下机器人停止按钮，此时，电机电源不切断，抱闸也不刹车，因而不影响抱闸寿命。

1.5 机器人可动范围说明

为了便于理解，下图标识了机器人实际的可动范围。

请注意：机器人机械臂等所有可动区域都属于可动范围。

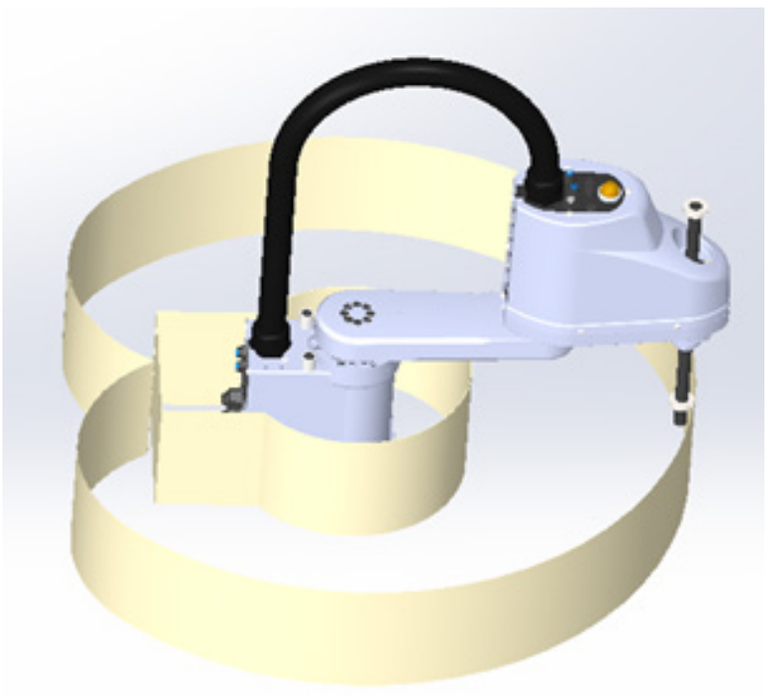


图 1-17 机器人可动范围说明

第 2 章 外壳的拆卸与安装

2.1 小臂外壳的拆卸与安装

警 告

- ◆ 外壳拆卸与安装过程中，切勿用力拉拽外壳，否则可能造成线缆损坏、断线或接触不良，有触电的危险或导致系统故障。
- ◆ 因维护作业等情况拆下外壳后，在维护作业等完成后，请及时将外壳装回原位。
- ◆ 如果固定外壳的螺钉遗失，请参考下图，必须使用规定的螺钉类型和数量固定外壳。
- ◆ 请将外壳固定充分，否则外壳可能在机器人运行时产生异响，甚至可能脱落飞出，造成严重的安全问题。

■ IRS100 系列

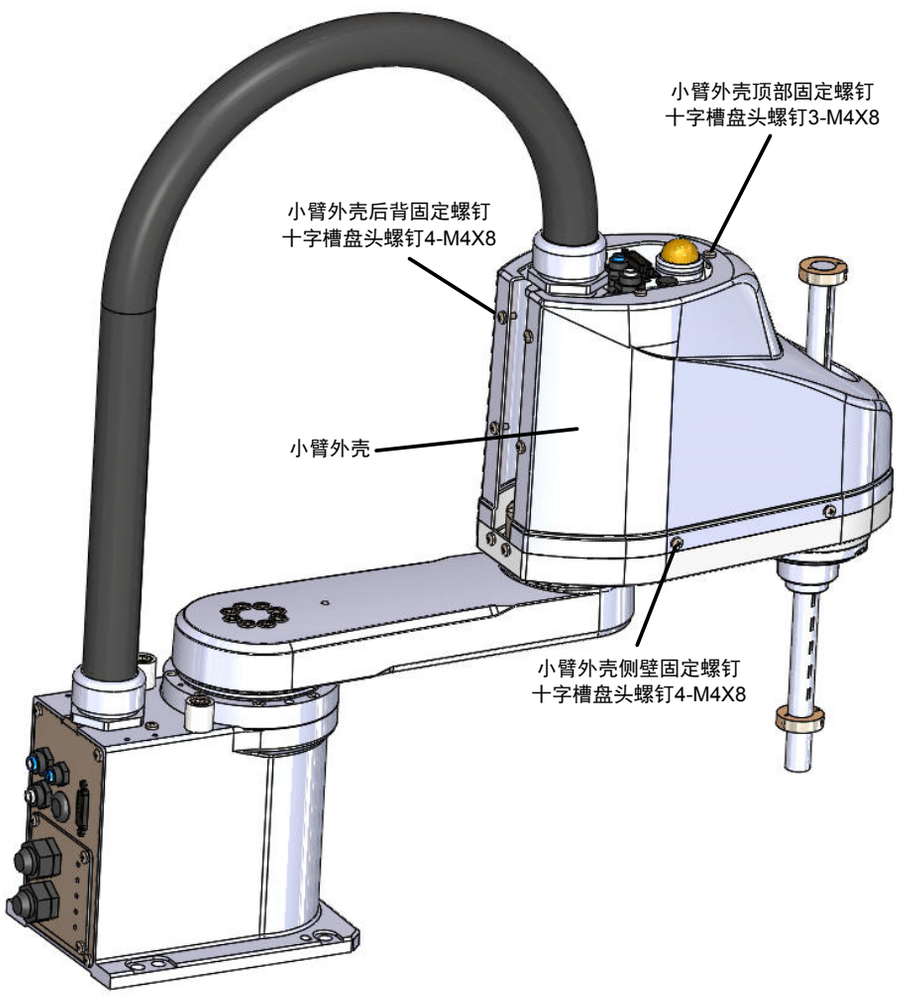


图 2-1小臂外壳拆卸与安装（IRS100 系列）

拆卸步骤：

第 1 步： 将顶部、侧壁、后背的 11 颗十字盘头螺钉取下；

第 2 步： 抬起并取下小臂外壳；

安装时，先扣好外壳，然后由上到下手动拧紧 11 颗螺钉；

■ IRS100-20 系列

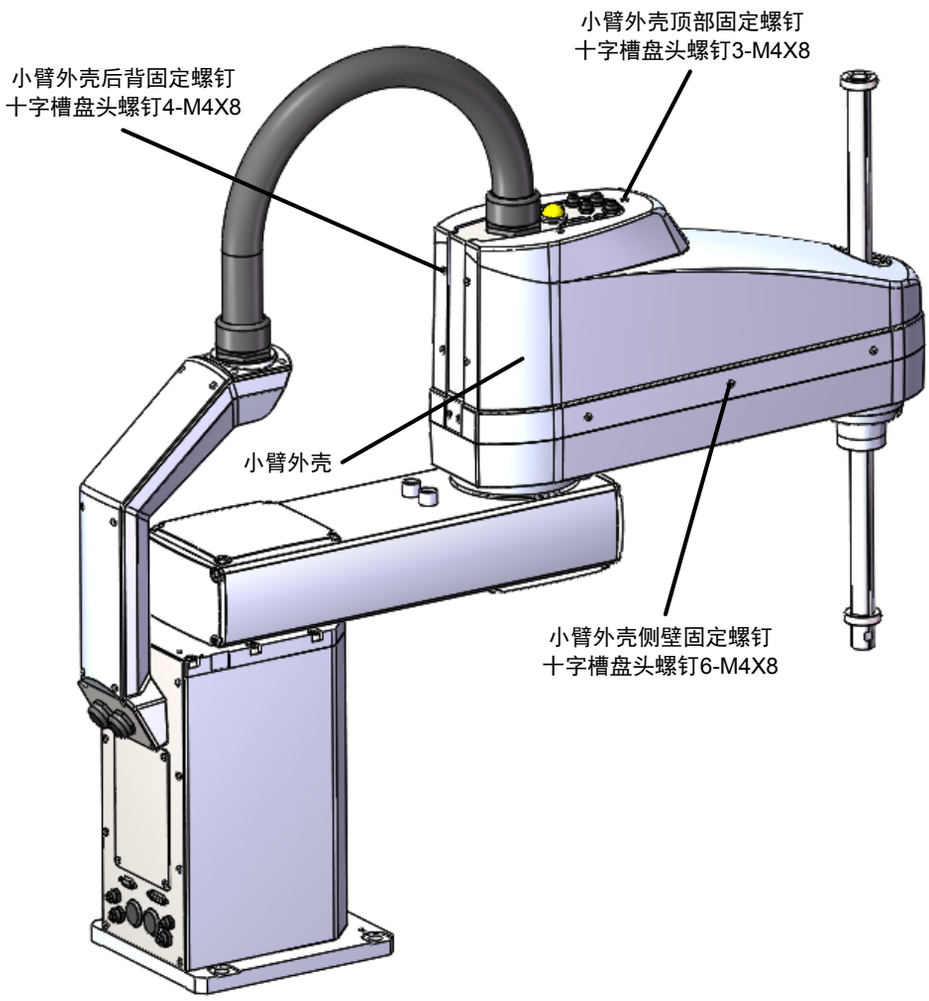


图 2-2小臂外壳拆卸与安装（IRS100-20 系列）

拆卸步骤：

第 1 步： 将顶部、侧壁、后背的 13 颗十字盘头螺钉取下；

第 2 步： 抬起并取下小臂外壳；

安装时，先扣好外壳，然后由上到下手动拧紧 13 颗螺钉；

2.2 底部花键母外壳的拆卸与安装

警 告

- 安装花键母外壳过程中，注意限位环的位置不可随意更改！
- 因维护作业等情况拆下外壳后，在维护作业等完成后，请及时将外壳装回原位。
- 如果固定外壳的螺钉遗失，请参考下图，必须使用规定的螺钉类型和数量固定外壳。
- 请将外壳充分固定好，否则外壳可能在机器人运行时产生异响，甚至可能脱落飞出，造成严重的安全问题。

■ IRS100 系列

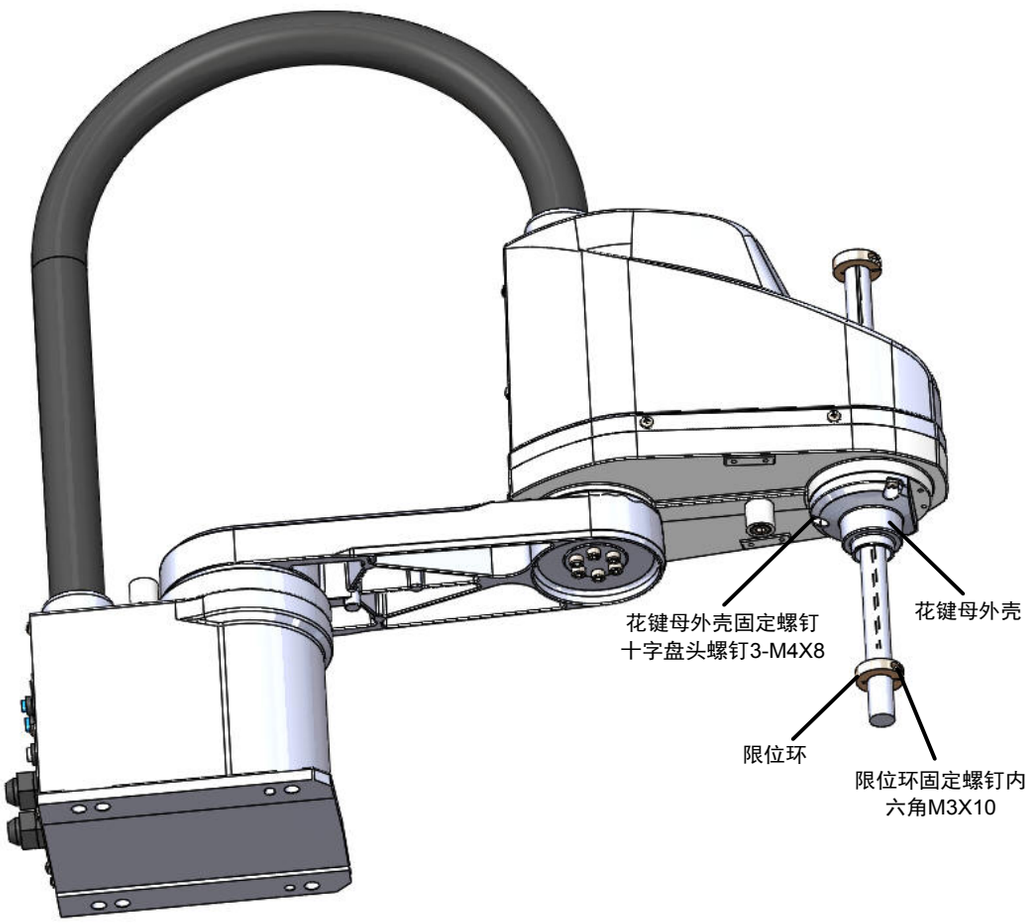


图 2-3 底部花键母外壳的拆卸与安装（IRS100 系列）

花键母外壳拆卸步骤：

第 1 步： 松开限位环固定螺钉（限位环紧固螺钉 M3X10），取下限位环；

第 2 步： 松开花键母外壳螺钉（花键母外壳固定螺钉十字盘头螺钉 3-M4X8）；

第 3 步： 取下花键母外壳；

安装时，先装好底部外壳，再装限位环。

■ IRS100-20 系列

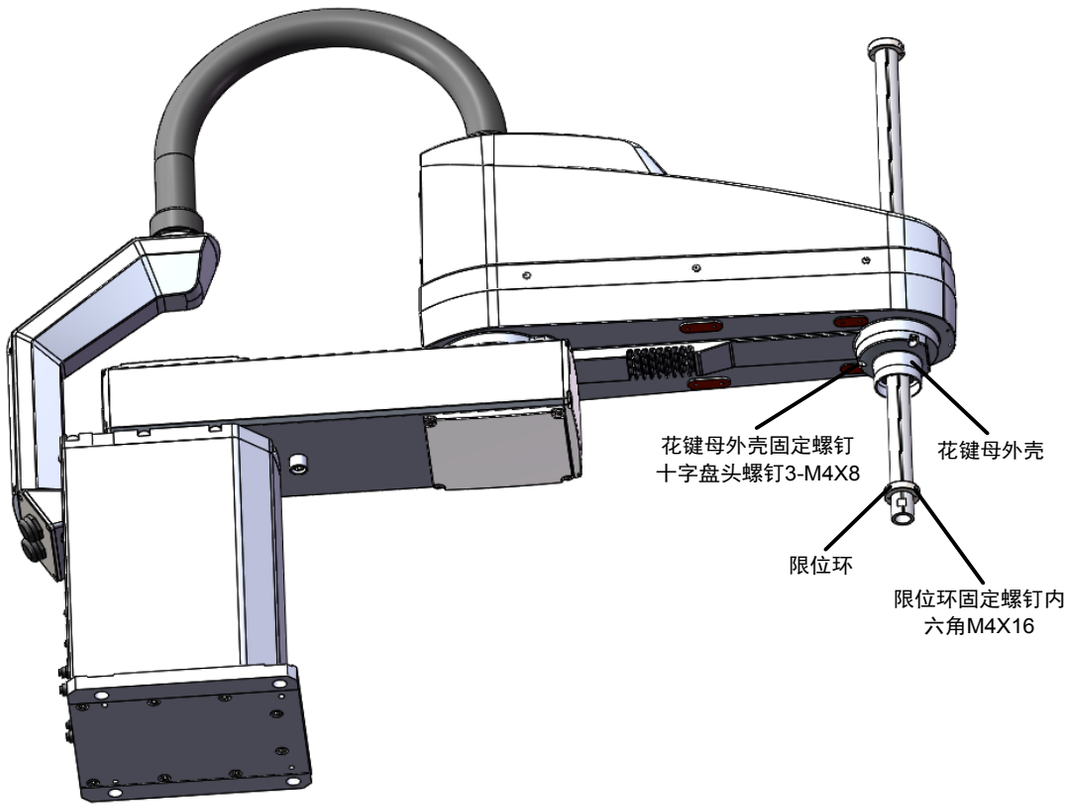


图 2-4 底部花键母外壳的拆卸与安装（IRS100-20 系列）

花键母外壳拆卸步骤：

第 1 步： 松开限位环固定螺钉（限位环紧固螺钉 M4X16），取下限位环；

第 2 步： 松开花键母外壳螺钉（花键母外壳固定螺钉十字盘头螺钉 3-M4X8）；

第 3 步： 取下花键母外壳；

安装时，先装好底部外壳，再装限位环。

第 3 章 定期点检

3.1 定期检查项目

为了能够使机器人更加安全高效的运行，需要对机器人定期的进行检测及保养。

机器人系统要在充分确定安全的情况下进行维护，并且只能经过机器人安全培训的人员才能维护该系统。

机器人维护按时间分五个阶段：日常、1 个月、3 个月、6 个月、12 个月；

每到一定阶段，追加相应的点检项目。

1） 电源 OFF 时（不动作时）的检查

检查项目	检查位置	日常 检查	1 个月检查	3 个月检查	6 个月检查	12 个月检查
确认螺钉有无松动 / 晃动， 如有，则进行加紧。	末端执行器安装螺钉	√	√	√	√	√
	机器人的底座安装螺钉	√	√	√	√	√
	各关节	√	√	√	√	√
	轴周边的螺钉					√
	电机、减速机等的螺钉					√
确认连接器有无松动， 如有，则压入 / 加紧。	机器人侧外部（连接器板等）	√	√	√	√	√
	机器人线缆单元		√	√	√	√
伤痕检查 清除附着的灰尘等	机器人本体	√	√	√	√	√
	外部线缆		√	√	√	√
变形、位置偏移的修正	安全护栏等	√	√	√	√	√
确认同步皮带有无松弛， 如有，则重新张紧	第 2 机械臂内部				√	√
润滑脂的状态是否足够润滑，根据需要添加适量用润滑脂	滚珠丝杠花键			√	√	√

2） 电源 ON 时（不动作时）的检查

检查项目	检查位置	日常检查	1 个月检查	3 个月检查	6 个月检查	12 个月检查
用手轻轻摇晃线缆，确认有无断线	机器人侧外部（连接器板等）				√	√
在使能状态下用手按压各机械臂，确认有无晃动	各机械臂					√

3） 电源 ON 时（动作时）的检查

检查项目	检查位置	日常检查	1 个月检查	3 个月检查	6 个月检查	12 个月检查
作业区域的确认	各关节					√
确认有无动作异常声音、异常振动	全体	√	√	√	√	√
利用量规重复测量精度	全体					√

3.2 螺钉紧固力矩和拧紧方法

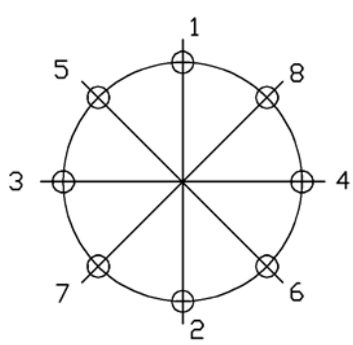
连接部位的内六角螺栓，如果没有特殊说明，请按照以下力矩紧固：

螺栓	力矩（N.m)
M3	1.5
M4	3
M5	7
M6	9
M8	30
M10	58
M12	127.5

紧定螺钉，如果没有特殊说明，请按照以下力矩紧固：

螺栓	力矩（N.m)
M3	0.8
M4	2

螺钉环形拧紧时，采用对角拧紧法：



切勿一次将力矩拧到最大，先用内六角扳手按图示稍作拧紧，最后用扭矩扳手按照上述表格的力值拧紧至最终力值。

3.3 润滑脂加注

润滑脂加注周期说明：

关节	部位	周期
第一关节	减速机	更换电机时
第二关节	减速机	更换电机时
第三关节	滚珠丝杠花键	每月

注意以下三点：

1） 减速机和滚珠丝杠加注时，必须使用原装润滑脂，润滑脂型号请参考*“4.10 部件清单”*。

2） 严禁丝杠润滑脂用光才加注，否则会产生不可逆的机械损坏！

3） 通常情况下，减速机不需要加注润滑脂，但在极限（占空比、速度、载荷）条件下，可以每 10000 小时加注一次润滑脂。

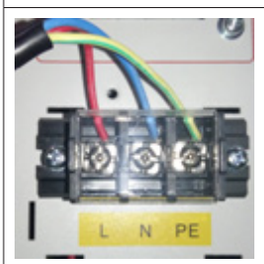
NOTE

◆ 一旦润滑脂进入眼睛、口腔或沾染皮肤，请按如下流程处理：

1. 进入眼睛时，请先用清水清洗，然后就医；
2. 进入口腔时，请充分漱口，如果吞咽下去则需要立即就医；
3. 沾染皮肤时，请用肥皂水冲洗。

3.4 控制柜连接线的日常点检

控制柜连接线缆日常检查项目如下：

检查点	检查内容	检查频次
	①检查机器人本体动力输出线、编码器线是否与控制柜连接完好；	每天
	②检查示教器线缆是否与控制柜接好；	每天
	③检查控制柜与外部电源线是否接好（L、N、PE 分别对应棕、蓝、黄绿）；	每天
所有线束	④擦拭纸或干抹布擦干净，不能粘酸、碱、油或腐蚀性液体	每天

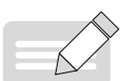
第 4 章 更换与维护

4.1 J1 轴维护

	名称	数量
维护部件	J1 轴电机	1
	减速机润滑脂	/
工具	内六角扳手一套	1
	扭力批（0-10N.m）	1
	六角批头一套	1
	吸油纸	若干
	抹布	若干
	十字螺丝刀	2
	拉力计	1
	张力仪	1

螺钉拧紧力矩参照：

螺栓	力矩（N.m)
M5	7
M6	9



NOTE

- 批头规格推荐：直径 3/4/5/6/8mm，长度：15cm；直径 1.5/2/2.5，长度：10cm。
- 十字螺丝刀规格推荐：直径 5mm，长度：20cm；直径 3mm，长度：15cm。

危 险

- 严禁非专业人员进行设备安装、接线、保养维护、检查或部件更换！
- 请勿在通电状态下装卸电机连接器，否则可能导致机器人进行异常动作，造成严重的安全问题。此外，如果在通电状态下进行维护作业，可能有触电危险。
- 请勿对本手册未记载的部位进行拆卸，或按照与记载不同的方法进行维护。
- 请在安全护栏外确认更换部件后机器人的动作，否则可能因机器人进行异常的动作造成严重的安全问题。
- 进入正常运转前，请确认紧急停止开关与安全门开关工作正常，否则系统发生紧急状况时将无法发挥安全功能，可能会导致重伤或重大损害。
- 维护电机后，必须要标定零点，否则重新开机后容易飞车，造成严重的安全问题！

警 告

- 设备出现故障或损坏时，由专业人员按照维修指导对设备进行故障排除和维修，并做好维修记录。
- 维护时请勿使异物进入到设备内部与连接端子中。
- 更换设备后，请务必重新对设备接线检查与参数设置。
- 进行更换作业时，请勿向电机轴施加过大的冲击，否则可能导致电机或编码器使用寿命缩短或损坏。
- 请勿拆卸电机和编码器，否则可能因重新安装时错位等导致电机无法使用。

4.1.1 J1 轴电机维护

■ IRS100 系列

第 1 步： 取下大臂和小臂；

松开大臂紧固螺钉（内六角螺钉 8-M6 X16），将大臂及整个小臂一起取下。

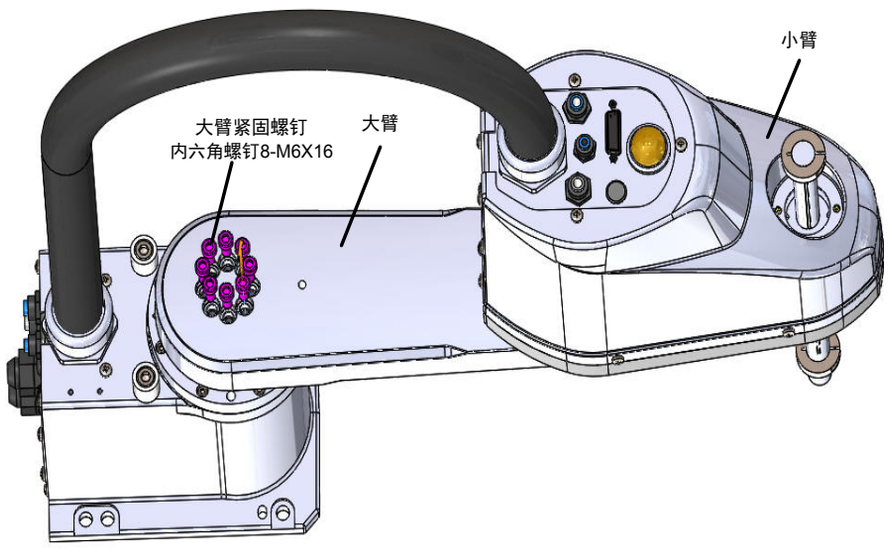


图 4-1 松开大臂螺钉（IRS100 系列）

注 意

◆ 拆下大臂螺钉时，请扶住小臂，防止重心不稳而倾倒。

第 2 步： 取下 J1 轴电机组件；

松开 J1 轴电机组件紧固螺钉，并取下 J1 轴电机组件。

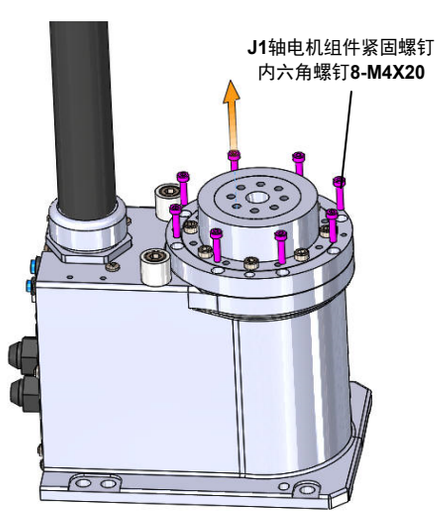


图 4-2 松开 J1 轴电机组件紧固螺钉（IRS100 系列）

第 3 步： 更换 J1 轴电机组件。还原 J1 轴电机时，请按照与拆卸相反的步骤进行安装。

■ IRS100-20 系列

第 1 步： 取出上下盖板；

松开大臂紧固螺钉（十字盘头螺钉 4-M4x8），将上下盖板取出；

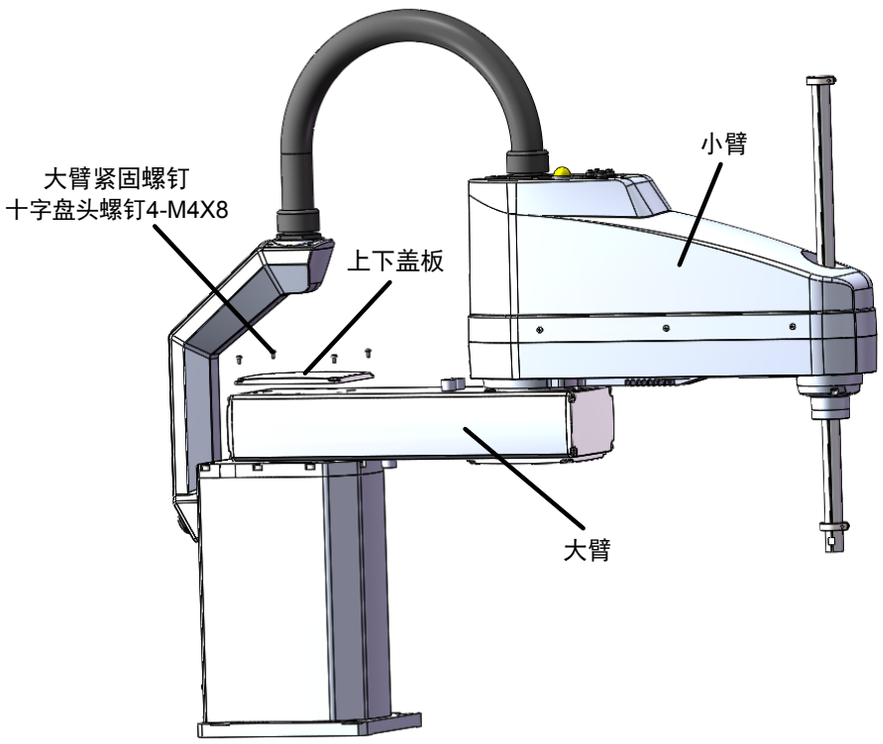


图 4-3 松开大臂上下盖板螺钉（IRS100-20 系列）

第 2 步： 取下大臂和小臂；

拆下大臂紧固螺钉（内六角螺钉 8-M6 X16），将大臂及整个小臂一起取下。

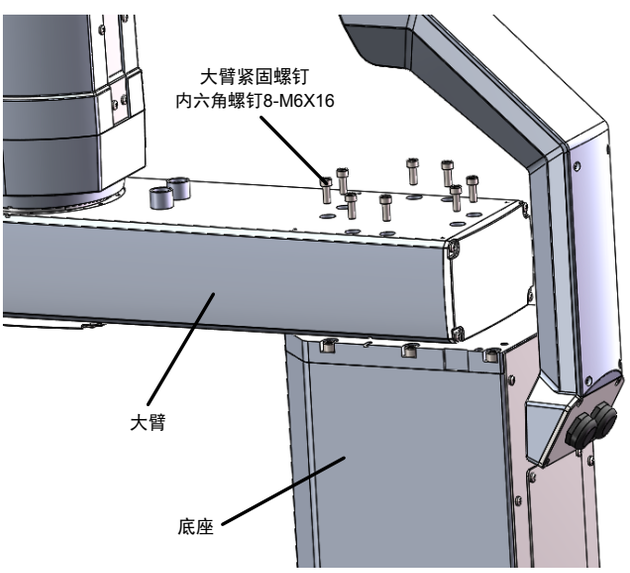


图 4-4 松开大臂螺钉（IRS100-20 系列）

注 意

◆ 拆下大臂螺钉时，请扶住小臂，防止重心不稳而倾倒。

第 3 步： 取下 J1 轴电机组件；

松开 J1 轴电机组件紧固螺钉（内六角螺钉 8-M8X20），并取下 J1 轴电机组件；

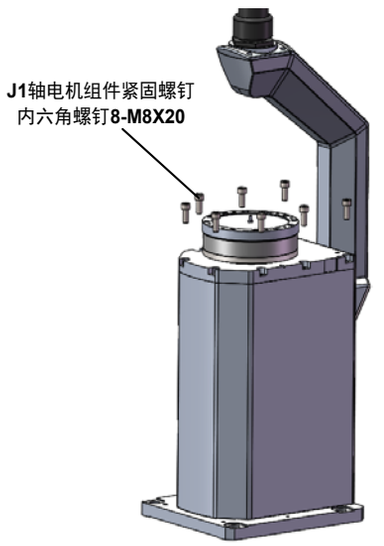


图 4-5 松开 J1 轴电机组件紧固螺钉（IRS100-20 系列）

第 4 步： 更换 J1 轴电机组件。还原 J1 轴电机时，请按照与拆卸相反的步骤进行安装。

4.1.2 J1 轴减速机润滑油脂更换

■ IRS100 系列

第 1 步： 取下大臂和小臂；

按照*“4.1.1 J1 轴电机维护”*中 IRS100 系列的第 1 步进行操作，取下大臂和小臂。

（参考：*“图 4-1 松开大臂螺钉（IRS100 系列）”*）。

第 2 步： 取下谐波减速机钢轮；

拆卸谐波减速机紧固螺钉（内六角螺钉 8-M5 X20），将谐波减速机钢轮取下。

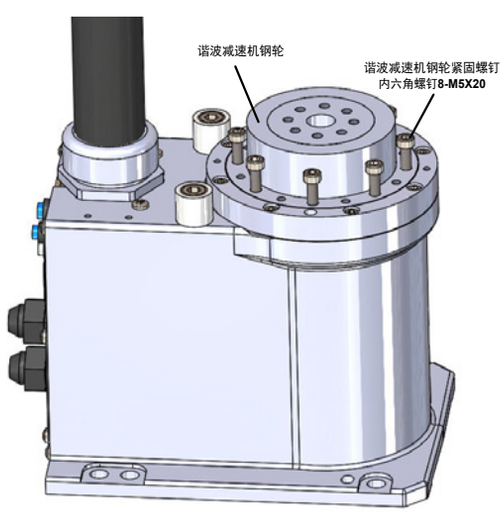


图 4-6 拆卸 J1 轴电机（IRS100 系列）

第 3 步： 波发生器涂抹润滑油；

清理谐波减速机波发生器及谐波减速机钢轮中的旧油脂，并将润滑脂均匀涂抹在波发生器表面，保证减速机内油腔油脂量约 30mL。

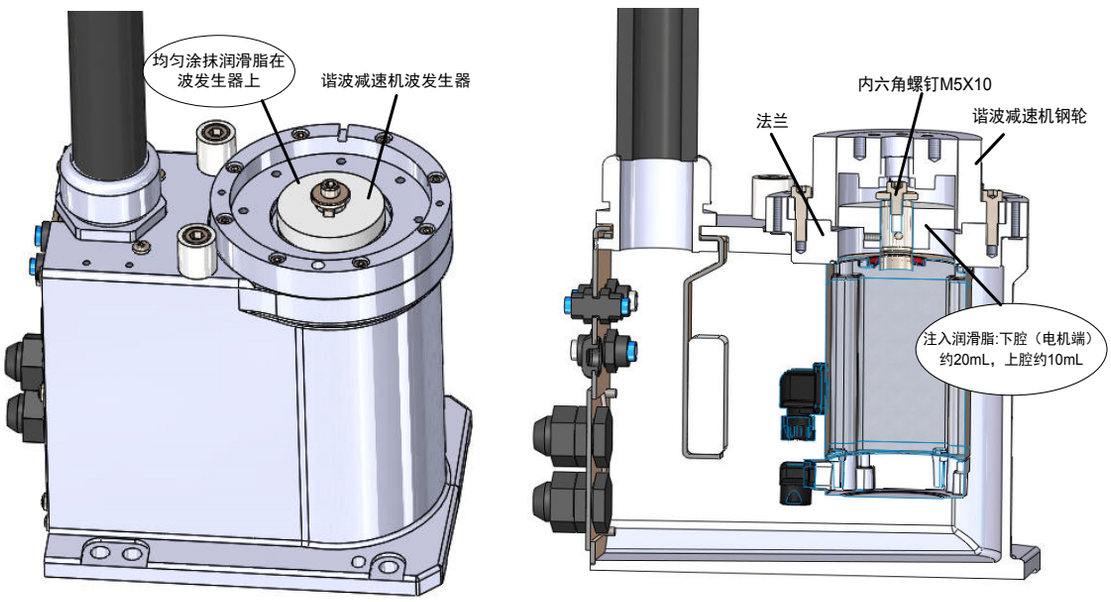


图 4-7　波发生器涂抹润滑油（IRS100 系列）

第 4 步： 重新装配复原；

使用内六角扳手旋转内六角螺钉 M5X10，同时将谐波减速机钢轮轻轻压入，带钢轮完全贴合上法兰后。拧紧内六角螺钉 8-M5X20（分三次对角拧紧，力矩依次增加 3 N.m、5 N.m、7 N.m）；

将大臂安装在谐波减速机钢轮表面，保证大臂与谐波减速贴实不得倾斜，紧固内六角螺钉 8-M6 X16（分三次对角拧紧，力矩依次增加 5 N.m、7 N.m、9 N.m）；

■　IRS100-20 系列

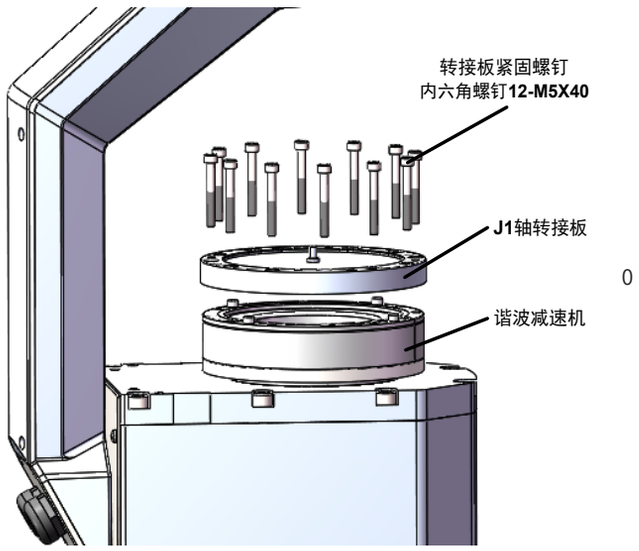
第 1 步： 取出上下盖板，取下大臂和小臂；

按照*“4.1.1 J1 轴电机维护”*中 IRS100-20 系列的第 1~2 步进行操作，取下大臂和小臂。

（参考：*“图 4-3 松开大臂上下盖板螺钉（IRS100-20 系列）”* *“图 4-4 松开大臂螺钉（IRS100-20 系列）”*）。

第 2 步： 拆卸 J1 轴转接板；

拆卸 J1 轴转接板紧固螺钉（内六角螺钉 12-M5 X40），将转接板取下。



0

图 4-8　拆卸 J1 轴转接板（IRS100-20 系列）

第 3 步： 谐波减速机涂抹润滑油；

清理谐波减速机空腔中的旧油脂，并在波发生器与柔轮形成的内腔中注入润滑脂，保证减速机内油腔油脂量约 150mL。

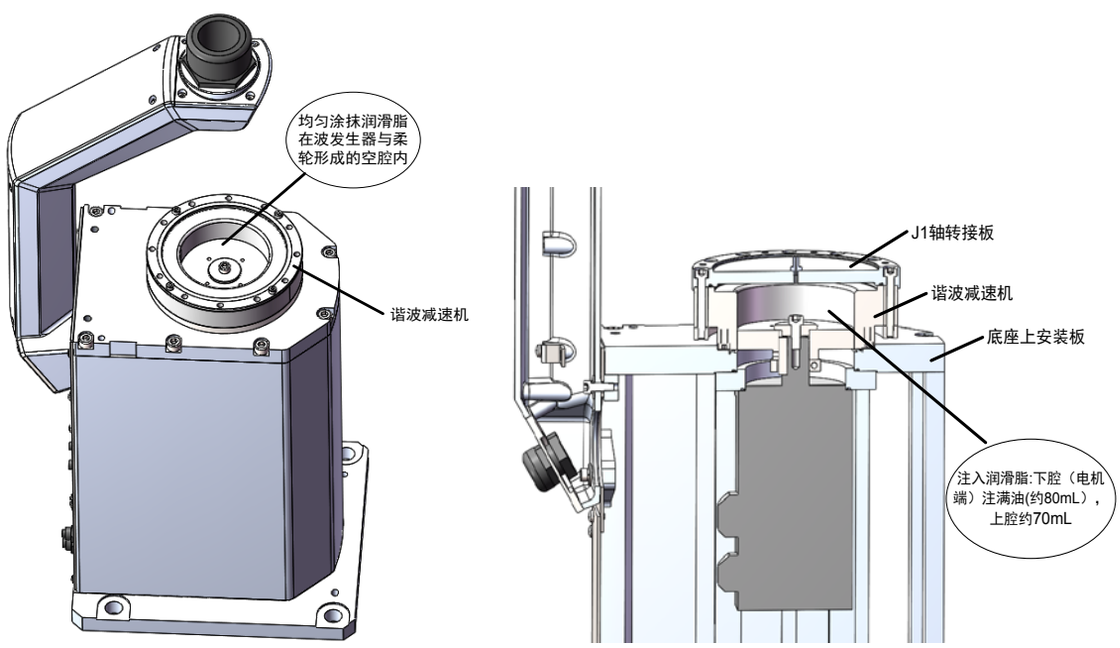


图 4-9　谐波减速机涂抹润滑油（IRS100-20 系列）

第 5 步： 重新装配复原；

安装 J1 轴转接板时，拧紧内六角螺钉 8-M5X20（分三次对角拧紧，力矩依次增加 3 N.m、5 N.m、7 N.m）。

将大臂安装在谐波减速机钢轮表面，保证大臂与谐波减速贴实不得倾斜，紧固内六角螺钉 8-M6×16（分三次对角拧紧，力矩依次增加 5 N.m、7 N.m、9 N.m）。

NOTE

- 拆卸下的螺钉不允许重复使用，需使用新的螺钉；
- 安装螺钉时螺钉表面使用乐泰 243 螺纹胶。

4.2 J2 轴维护

	名称	数量
维护部件	J1 轴电机	1
	减速机润滑脂	/
工具	内六角扳手一套	1
	扭力批（0-10N.m）	1
	六角批头一套	1
	吸油纸	若干
	抹布	若干
	十字螺丝刀	2
	拉力计	1
	张力仪	1

螺钉拧紧力矩参照

螺栓	力矩（N.m)
M3	1.5
M4	3

NOTE

- 批头规格推荐：直径 3/4/5/6/8mm，长度：15cm；直径 1.5/2/2.5，长度：10cm。
- 十字螺丝刀规格推荐：直径 5mm，长度：20cm；直径 3mm，长度：15cm。

危 险

- 严禁非专业人员进行设备安装、接线、保养维护、检查或部件更换！
- 请勿在通电状态下装卸电机连接器，否则可能导致机器人进行异常动作，造成严重的安全问题。此外，如果在通电状态下进行维护作业，可能有触电危险。
- 请勿对本手册未记载的部位进行拆卸，或按照与记载不同的方法进行维护。
- 请在安全护栏外确认更换部件后机器人的动作，否则可能因机器人进行异常的动作造成严重的安全问题。
- 进入正常运转前，请确认紧急停止开关与安全门开关工作正常，否则系统发生紧急状况时将无法发挥安全功能，可能会导致重伤或重大损害。
- 维护电机后，必须要标定零点，否则重新开机后容易飞车，造成严重的安全问题！

警 告

- 设备出现故障或损坏时，由专业人员按照维修指导对设备进行故障排除和维修，并做好维修记录。
- 维护时请勿使异物进入到设备内部与连接端子中。
- 更换设备后，请务必重新对设备接线检查与参数设置。
- 进行更换作业时，请勿向电机轴施加过大的冲击，否则可能导致电机或编码器使用寿命缩短或损坏。
- 请勿拆卸电机和编码器，否则可能因重新安装时错位等导致电机无法使用。

4.2.1 J2 轴电机维护

■ IRS100 系列

第 1 步： 拆卸小臂外壳；

按照*“2.1 小臂外壳的拆卸与安装”*进行操作，拆下小臂外壳。

（参考：*“图 2-1 小臂外壳拆卸与安装（IRS100 系列）”*）。

第 2 步： 松开钣金件螺钉；

松开小臂钣金件后的十字盘头螺钉 2-M4X8。

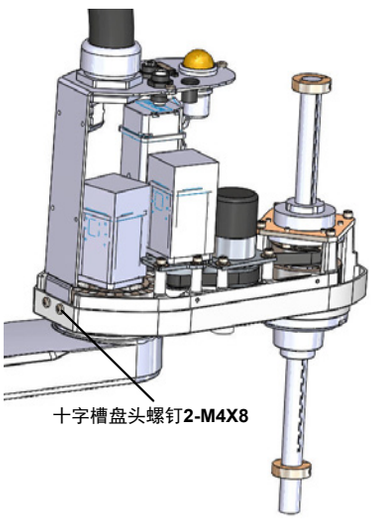


图 4-10 松开钣金件螺钉（IRS100 系列）

第 3 步： 松开 J2 轴电机法兰盘，必要时松开临近有干涉的电机，取下 J2 轴电机组件；

拧松 J3 轴电机安装板和 J4 轴电机安装板上的螺钉，小心放置 J3 轴和 J4 轴电机，切勿用力拽拉电机连接线缆。
拧松 J2 轴法兰紧固螺钉内六角螺钉 8-M4X16。

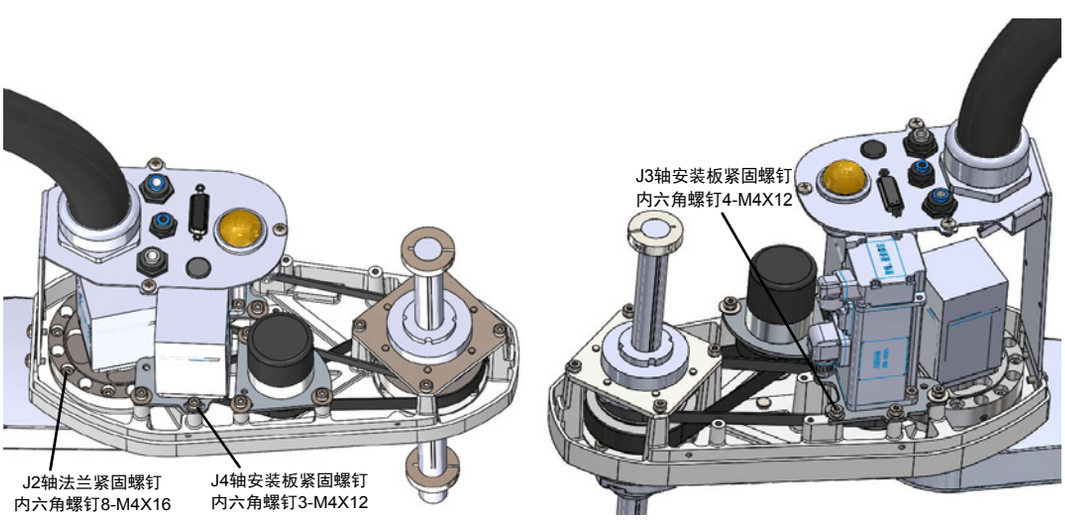


图 4-11 松开 J2 轴电机法兰盘（IRS100 系列）

NOTE

◆ 还原装配 J2 轴电机过程中，电机法兰平台需要抹上平面密封胶（推荐使用 5699 密封胶）。

第 4 步： 更换 J2 轴电机组件。还原 J2 轴电机时，请按照与拆卸相反的步骤进行安装。

■ IRS100-20 系列

第 1 步： 拆卸小臂外壳；

按照*“2.1 小臂外壳的拆卸与安装”*进行操作，拆下小臂外壳。

（参考：*“图 2-2 小臂外壳拆卸与安装（IRS100-20 系列）”*）。

第 2 步： 松开钣金件螺钉；

松开小臂钣金件后的十字盘头螺钉 2-M4X8。

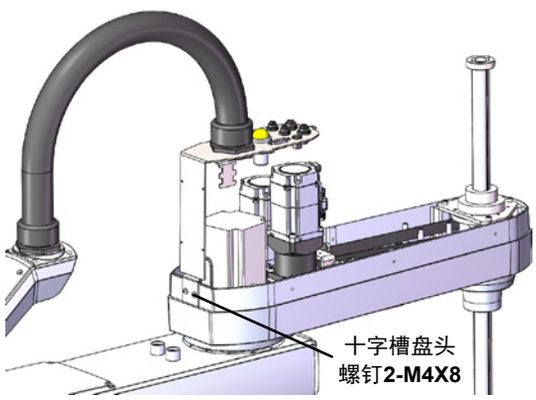


图 4-12 松开钣金件螺钉（IRS100-20 系列）

第 3 步： 松开 J2 轴电机法兰盘，必要时松开临近有干涉的电机，取下 J2 轴电机组件；

拧松 J3 轴电机安装板和 J4 轴电机安装板上的螺钉，小心放置 J3 轴和 J4 轴电机，切勿用力拽拉电机链接线缆。拧松 J2 轴法兰紧固螺钉内六角螺钉 8-M4X16，注意保留所带的 O 型圈。

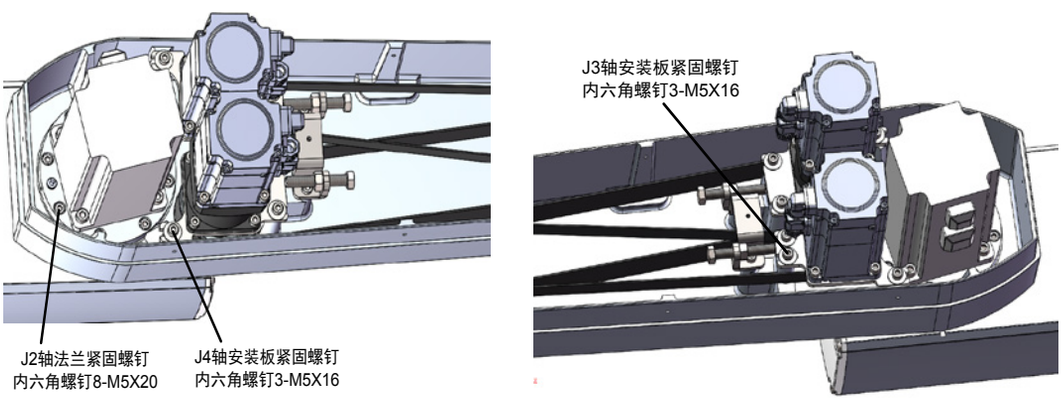


图 4-13 松开二轴电机法兰盘（IRS100-20 系列）

第 4 步： 还原 J2 轴电机时，请按照与拆卸相反的步骤进行安装。

4.2.2 J2 油脂更换

■ IRS100 系列

第 1 步： 取下 J2 轴电机组件。

按照*“4.2.1 J2 轴电机维护”*中 IRS100 系列第 1~3 步进行操作，从小臂上取下 J2 轴电机组件。

（参考：*“图 2-1 小臂外壳拆卸与安装（IRS100 系列）”* *“图 4-10 松开钣金件螺钉（IRS100 系列）”* *“图 4-11 松开二轴电机法兰盘（IRS100 系列）”*）。

*第 2 步：*波发生器涂抹润滑油，

清理谐波减速机波发生器及谐波减速机钢轮中的旧油脂，并将润滑脂均匀涂抹在波发生器表面，钢轮内腔注入润滑脂：下腔约 7mL，上腔约 5mL。

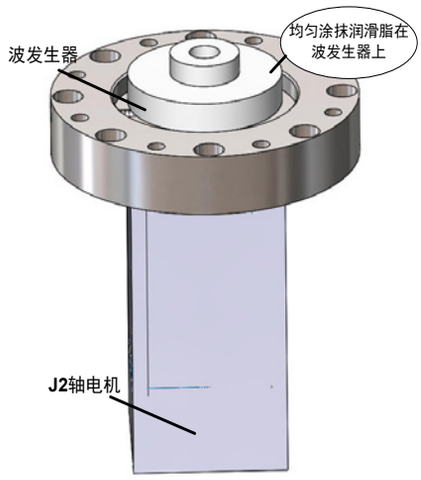


图 4-14 波发生器涂抹润滑油（IRS100 系列）

■ IRS100-20 系列

*第 1 步：*取下 J2 轴电机组件。

按照*“4.2.1 J2 轴电机维护”*中 IRS100-20 系列第 1~3 步进行操作，从小臂上取下 J2 轴电机组件。

（参考：*“图 2-2 小臂外壳拆卸与安装（IRS100-20 系列）”“图 4-12 松开钣金件螺钉（IRS100-20 系列）”“图 4-13 松开二轴电机法兰盘（IRS100-20 系列）”*）。

*第 2 步：*波发生器涂抹润滑油。

清理谐波减速机波发生器及谐波减速机钢轮中的旧油脂，并将润滑脂均匀涂抹在波发生器表面，钢轮内腔注入润滑脂，保证减速机内油腔油脂量约 230mL。

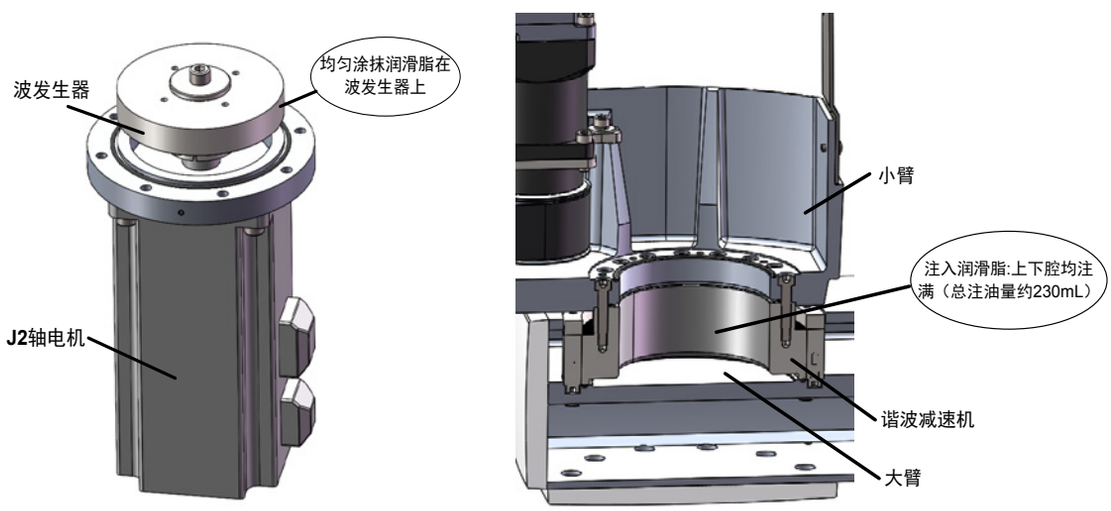


图 4-15 波发生器涂抹润滑油（IRS100-20 系列）

NOTE

- 拆卸下的螺钉不允许重复使用，需使用新的螺钉；
- 安装螺钉时螺钉表面使用乐泰 243 螺纹胶。

4.3 J3 轴维护

	名称	数量
维护部件	J1 轴电机	1
	减速机润滑脂	/
工具	内六角扳手一套	1
	扭力批（0-10N.m）	1
	六角批头一套	1
	吸油纸	若干
	抹布	若干
	十字螺丝刀	2
	拉力计	1
	张力仪	1

螺钉拧紧力矩参照

螺栓	力矩（N.m)
M3	1.5
M4	3

NOTE

- 批头规格推荐：直径 3/4/5/6/8mm，长度：15cm；直径 1.5/2/2.5，长度：10cm。
- 十字螺丝刀规格推荐：直径 5mm，长度：20cm；直径 3mm，长度：15cm。

危 险

- 严禁非专业人员进行设备安装、接线、保养维护、检查或部件更换！
- 请勿在通电状态下装卸电机连接器，否则可能导致机器人进行异常动作，造成严重的安全问题。此外，如果在通电状态下进行维护作业，可能有触电危险。
- 请勿对本手册未记载的部位进行拆卸，或按照与记载不同的方法进行维护。
- 请在安全护栏外确认更换部件后机器人的动作，否则可能因机器人进行异常的动作造成严重的安全问题。
- 进入正常运转前，请确认紧急停止开关与安全门开关工作正常，否则系统发生紧急状况时将无法发挥安全功能，可能会导致重伤或重大损害。
- 维护电机后，必须要标定零点，否则重新开机后容易飞车，造成严重的安全问题！

警 告

- 设备出现故障或损坏时，由专业人员按照维修指导对设备进行故障排除和维修，并做好维修记录。
- 维护时请勿使异物进入到设备内部与连接端子中。
- 更换设备后，请务必重新对设备接线检查与参数设置。
- 进行更换作业时，请勿向电机轴施加过大的冲击，否则可能导致电机或编码器使用寿命缩短或损坏。
- 请勿拆卸电机和编码器，否则可能因重新安装时错位等导致电机无法使用。

4.3.1 J3 轴同步带维护

■ IRS100 系列

第 1 步： 拆卸小臂外壳；

按照*“2.1 小臂外壳的拆卸与安装”*进行操作，拆下小臂外壳。

（参考：*“图 2-1 小臂外壳拆卸与安装（IRS100 系列）”*）。

第 2 步： 松开 J3 轴电机组件和丝杆螺母；

松开 J3 轴安装板紧固螺钉 4-M4X12，松开丝杆螺母安装板紧固螺钉 4-M4X12，松开 J3 轴电机组件和丝杆螺母。

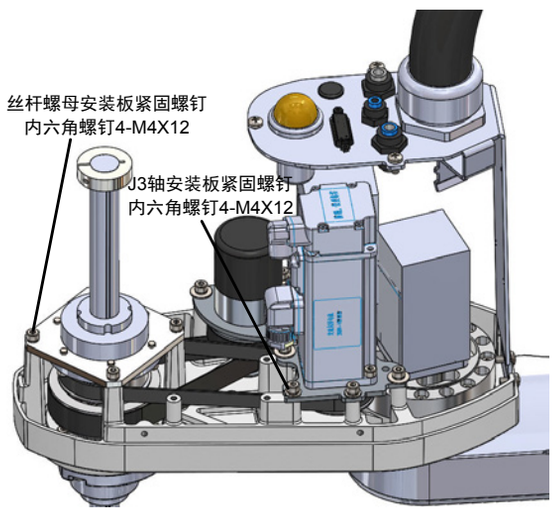


图 4-16 松开 J3 轴电机组件和丝杆螺母（IRS100 系列）

第 3 步： 更换 J3 轴同步带。

取出 J3 轴电机组件，升高丝杆螺母后，可将 J3 轴同步带抽出并进行更换。

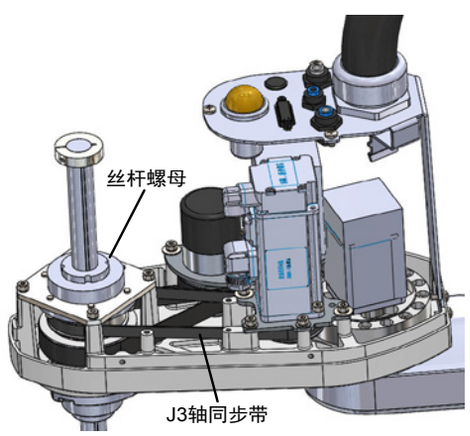


图 4-17 更换 J3 轴同步带（IRS100 系列）

第 4 步： 拧紧丝杆螺母安装板紧固螺钉 4-M4X12；

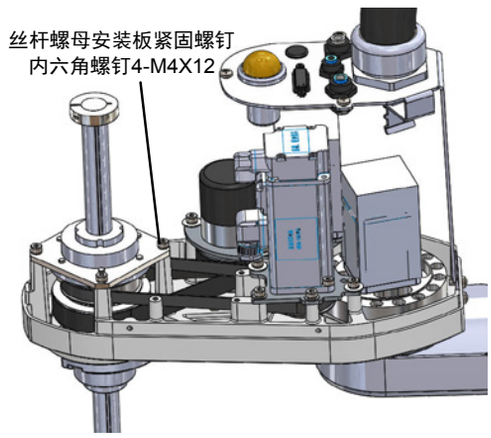


图 4-18 拧紧丝杆螺母安装板紧固螺钉（IRS100 系列）

第 5 步： 张紧同步带，拧紧电机螺钉；

张紧 J3 轴同步带，保证同步带装入同步带轮内，使用拉力计水平拉 J3 轴电机安装板（使用张力仪进行检测，保证同步带检测频率 110-140Hz）。

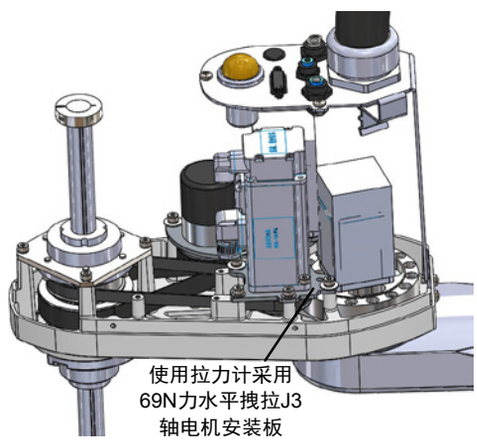


图 4-19 张紧 J3 轴同步带（IRS100 系列）

第 6 步： 锁紧 J3 轴安装板紧固螺钉；

锁紧 J3 安装板紧固螺钉内六角螺钉 4-M4X12。

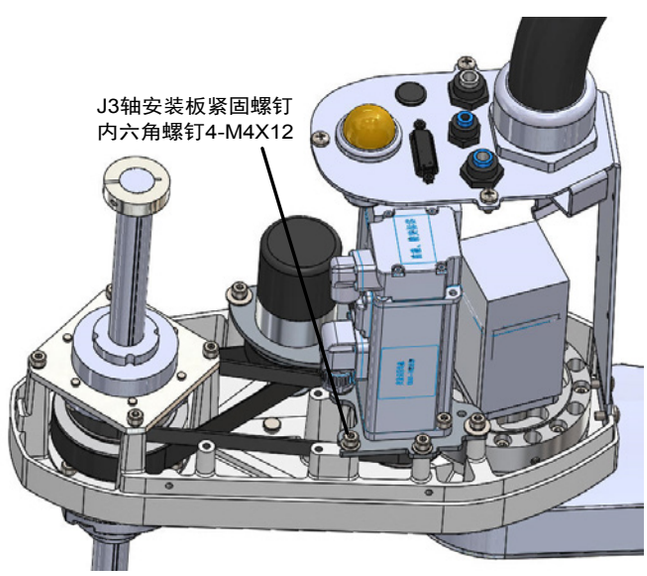


图 4-20 锁紧 J3 轴安装板紧固螺钉（IRS100 系列）

第 7 步： 还原小臂外壳，对小臂外壳的 11 颗螺钉进行拧紧。

■ IRS100-20 系列

*第 1 步：*拆卸小臂外壳；

按照*“2.1 小臂外壳的拆卸与安装”*进行操作，拆下小臂外壳。

（参考：*“图 2-2 小臂外壳拆卸与安装（IRS100-20 系列）”*）。

*第 2 步：*拆卸同步带张紧力调整装置；

拧松 J3、J4 轴同步带张紧力调整螺栓，然后松开张紧力调整装置螺钉。

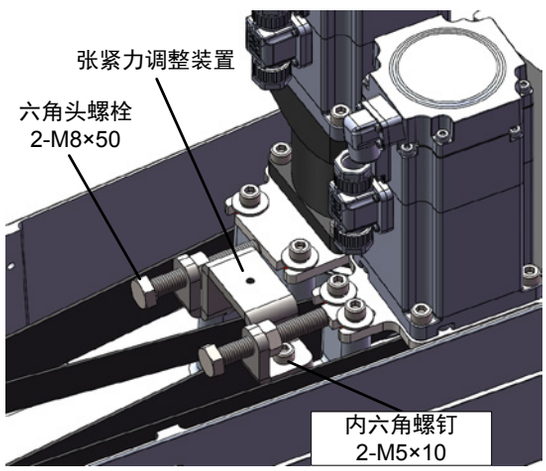


图 4-21 拆卸同步带张紧力调整装置（IRS100-20 系列）

*第 3 步：*松开 J3 轴电机组件和丝杆螺母；

松开 J3 轴安装板紧固螺钉 3-M5X10，松开丝杆螺母安装板紧固螺钉 3-M5X16。

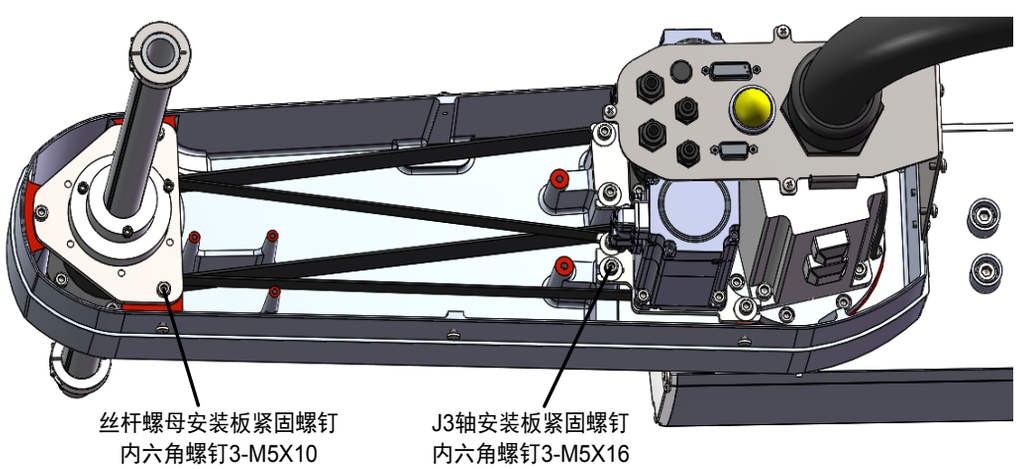


图 4-22 松开 J3 轴电机组件和丝杆螺母（IRS100-20 系列）

第 4 步： 更换 J3 轴同步带。

取出 J3 轴电机组件，升高丝杆螺母后，可将 J3 轴同步带抽出并进行更换。

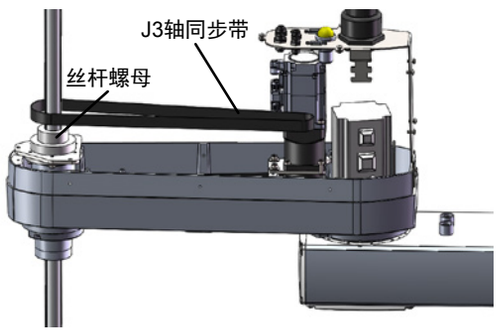


图 4-23 更换 J3 轴同步带（IRS100-20 系列）

第 5 步： 拧紧丝杆螺母安装板紧固螺钉。

更换好同步带并放置好丝杆螺母及安装板后，拧紧螺钉 3-M5X12。

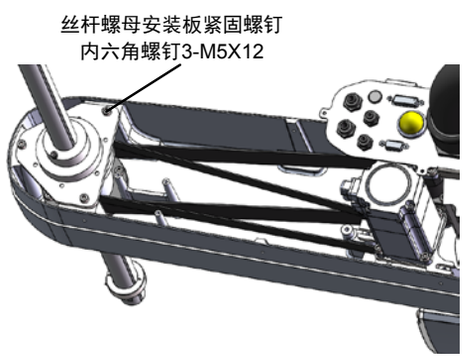


图 4-24 拧紧丝杆螺母安装板紧固螺钉（IRS100-20 系列）

第 6 步： 张紧同步带，锁紧 J3 安装板紧固螺钉；

张紧 J3 轴同步带，保证同步带装入同步带轮内，安装张紧力调整装置后，拧紧调节螺钉或螺母，使用张力仪进行检测，保证同步带检测频率 65-75Hz，完成后锁紧 J3 安装板紧固螺钉内六角螺钉 3-M5X16。

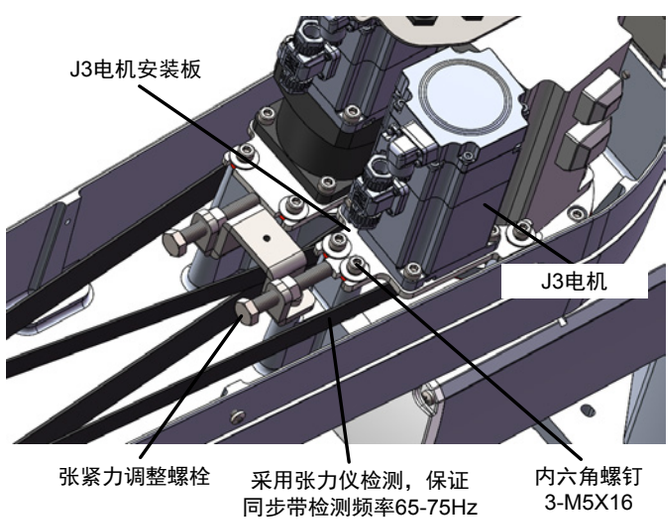


图 4-25 张紧 J3 轴同步带（IRS100-20 系列）

第 7 步： 还原小臂外壳，对小臂外壳的 13 颗螺钉进行拧紧。

NOTE

- 拆卸下的螺钉不允许重复使用，需使用新的螺钉；
- 安装螺钉时螺钉表面使用乐泰 243 螺纹胶。

4.3.2 J3 轴电机维护

■ IRS100 系列

第 1 步： 取下 J3 轴电机组件；

按照*“4.3.1 J3 轴同步带维护”*中的 IRS100 系列第 1~2 步取下 J3 轴电机组件。

（参考：*“图 2-2 小臂外壳拆卸与安装（IRS100-20 系列）”“图 4-16 松开 J3 轴电机组件和丝杆螺母（IRS100 系列）”*）。

第 2 步： 更换 J3 轴电机；

松开紧定螺钉 2-M4X5，拆下 J3 同步带轮，拆卸平键，松开紧固螺钉内六角螺钉 2-M4X10，取下 J3 轴电机安装板，取下 J3 轴电机。

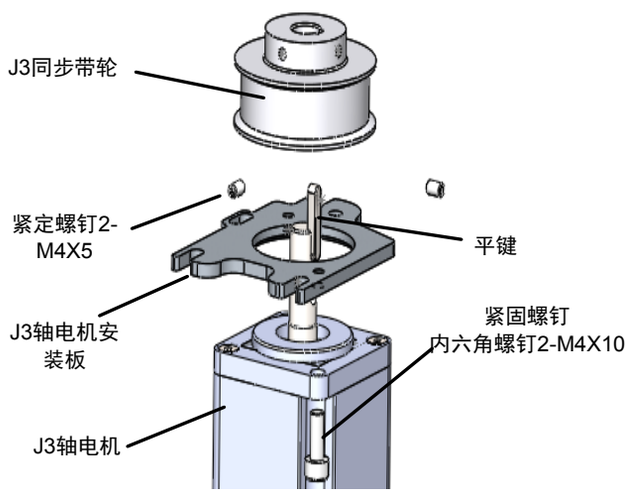


图 4-26 更换 J3 轴电机（IRS100 系列）

还原 J3 轴电机时，请按照与拆卸相反的步骤进行安装。

■ IRS100-20 系列

第 1 步： 取下 J3 轴电机组件；

按照*“4.3.1 J3 轴同步带维护”*中的 IRS100-20 系列第 1~3 步取下 J3 轴电机组件。

（参考：*“图 2-2 小臂外壳拆卸与安装（IRS100-20 系列）”“图 4-21 拆卸同步带张紧力调整装置（IRS100-20 系列）”“图 4-22 松开 J3 轴电机组件和丝杆螺母（IRS100-20 系列）”*）。

第 2 步： 拆卸 J3 轴电机端同步带轮；

松开内六角螺钉 2-M4X10，松开内六角螺钉 2-M4X5，拆下同步带轮限位块，J3 轴同步带轮。

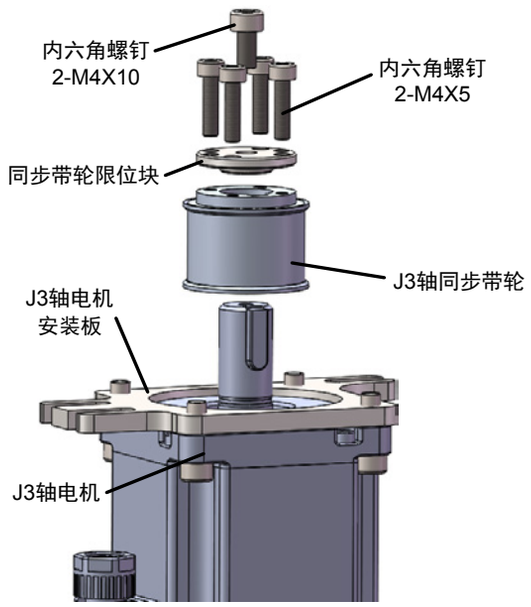


图 4-27 拆卸 J3 轴电机端同步带轮（IRS100-20 系列）

第 5 步： 更换 J3 轴电机；

拆下内六角螺钉 2-M4X10，取下 J3 轴电机安装板，取下 J3 轴电机。

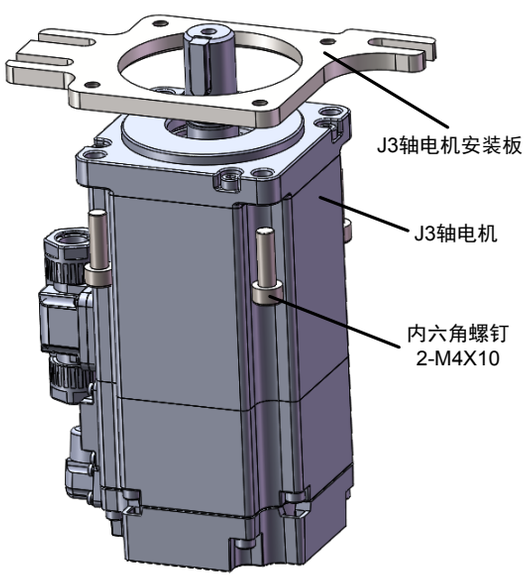


图 4-28 取下 J3 轴电机（IRS100-20 系列）

还原 J3 电机时，请按照与拆卸相反的步骤进行安装。

4.4 J4 轴维护

	名称	数量
维护部件	J1 轴电机	1
	减速机润滑脂	/
工具	内六角扳手一套	1
	扭力批（0-10N.m）	1
	六角批头一套	1
	吸油纸	若干
	抹布	若干
	十字螺丝刀	2
	拉力计	1
	张力仪	1

螺钉拧紧力矩参照

螺栓	力矩（N.m)
M3	1.5
M4	3

NOTE

- 批头规格推荐：直径 3/4/5/6/8mm，长度：15cm；直径 1.5/2/2.5，长度：10cm。
- 十字螺丝刀规格推荐：直径 5mm，长度：20cm；直径 3mm，长度：15cm。

危 险

- 严禁非专业人员进行设备安装、接线、保养维护、检查或部件更换！
- 请勿在通电状态下装卸电机连接器，否则可能导致机器人进行异常动作，造成严重的安全问题。此外，如果在通电状态下进行维护作业，可能有触电危险。
- 请勿对本手册未记载的部位进行拆卸，或按照与记载不同的方法进行维护。
- 请在安全护栏外确认更换部件后机器人的动作，否则可能因机器人进行异常的动作造成严重的安全问题。
- 进入正常运转前，请确认紧急停止开关与安全门开关工作正常，否则系统发生紧急状况时将无法发挥安全功能，可能会导致重伤或重大损害。
- 维护电机后，必须要标定零点，否则重新开机后容易飞车，造成严重的安全问题！

警 告

- 设备出现故障或损坏时，由专业人员按照维修指导对设备进行故障排除和维修，并做好维修记录。
- 维护时请勿使异物进入到设备内部与连接端子中。
- 更换设备后，请务必重新对设备接线检查与参数设置。
- 进行更换作业时，请勿向电机轴施加过大的冲击，否则可能导致电机或编码器使用寿命缩短或损坏。
- 请勿拆卸电机和编码器，否则可能因重新安装时错位等导致电机无法使用。

4.4.1 J4 轴同步带维护

■ IRS100 系列

第 1 步： 拆卸小臂外壳；

按照*“2.1 小臂外壳的拆卸与安装”*进行操作，拆下小臂外壳。

（参考：*“图 2-1 小臂外壳拆卸与安装（IRS100 系列）”*）。）

第 2 步： 松开 J3 轴电机组件和丝杆螺母；

松开 J3 轴安装板紧固螺钉 4-M4X12，松开丝杆螺母安装板紧固螺钉 4-M4X12，并取下 J3 轴同步带。

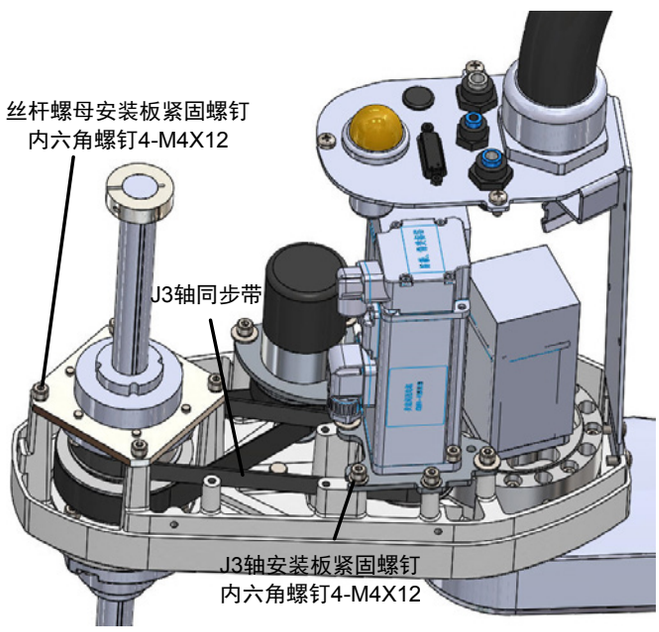


图 4-29 取下 J3 轴电机组件（IRS100 系列）

第 3 步： 松开 J4 轴电机组件和中间轴组件；

松开 J4 轴电机安装板内六角螺钉 3-M4X12，并松开中间轴组件紧固螺钉内六角 3-M4X12，更换 J4 轴二级同步带和 J4 轴一级同步带。

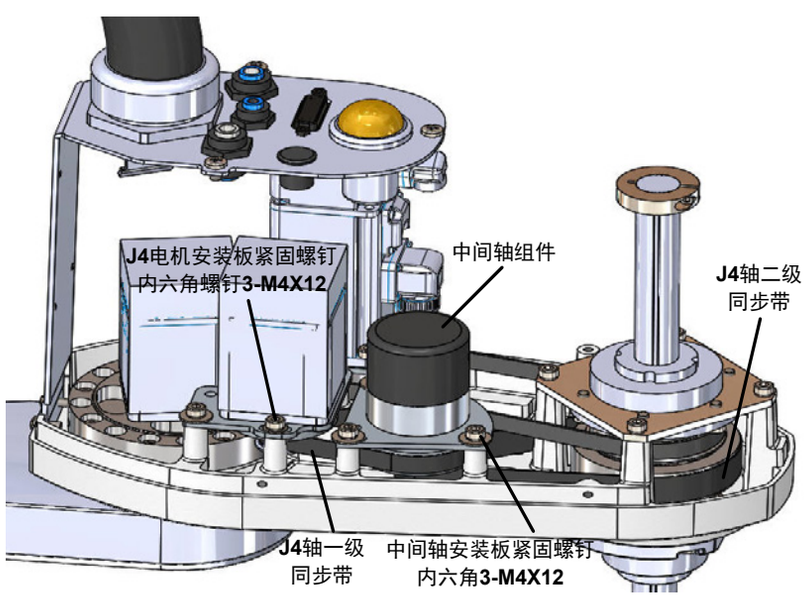


图 4-30 松开 J4 轴电机组件（IRS100 系列）

*第 4 步：*拧紧丝杆螺母螺钉；

将拆卸下来的 J3 轴同步带重新装入，拧紧丝杆螺母安装板内六角螺钉 4-M4X12，调整 J4 轴二级皮带张力拧紧中间轴组件紧固螺钉内六角 3-M4X12（使用张力仪进行检测，保证同步带检测频率 260-310Hz）。

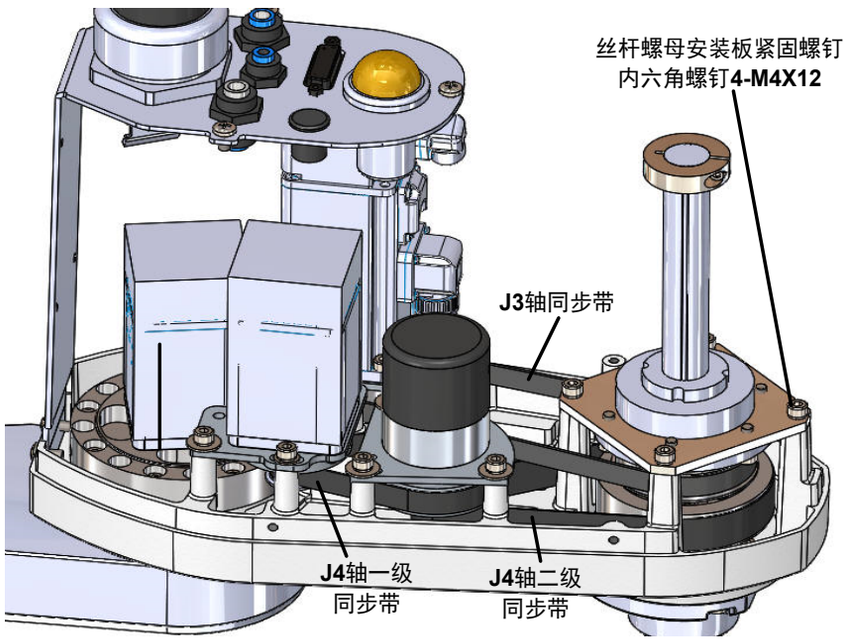


图 4-31 拧紧丝杆螺母螺钉（IRS100 系列）

*第 5 步：*拧紧 J4 轴电机安装板螺钉；

调整 J4 轴一级皮带张力拧紧 J4 轴电机安装板紧固螺钉内六角 3-M4X12（使用张力仪进行检测，保证同步带检测频率 380-430Hz）。

*第 6 步：*装好 J3 轴电机；

按照*“4.3.1 J3 轴同步带维护”*中 IRS100 系列第 5~6 步，对 J3 轴的皮带进行重新装配即可。

（参考：*“图 4-19 张紧 J3 轴同步带（IRS100 系列）”“图 4-20 锁紧 J3 轴安装板紧固螺钉（IRS100 系列）”*）

*第 7 步：*还原小臂外壳，对小臂外壳的 11 颗螺钉进行拧紧。

■ IRS100-20 系列

*第 1 步：*拆卸小臂外壳和同步带张紧力调整装置。

按照*“4.3.1 J3 轴同步带维护”*中 IRS100-20 系列第 1~2 步进行操作，拆卸小臂外壳和同步带张紧力调整装置。

（参考：*“图 2-2 小臂外壳拆卸与安装（IRS100-20 系列）”“图 4-21 拆卸同步带张紧力调整装置（IRS100-20 系列）”*）

第 2 步： 松开 J3 轴和 J4 轴电机组件螺钉；

松开 J3 轴安装板紧固螺钉 3-M5X16，松开 J4 轴安装板紧固螺钉 3-M5X16，取下 J3 轴和 J4 轴电机组件。

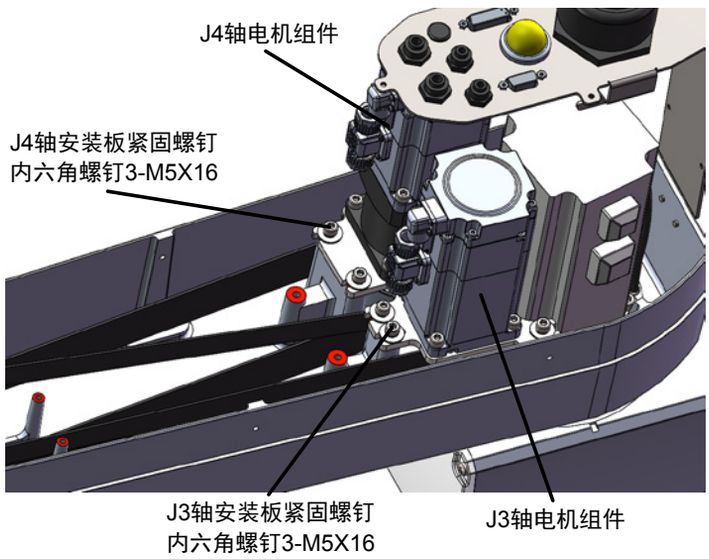


图 4-32 松开 J3 轴和 J4 轴电机组件螺钉（IRS100-20 系列）

第 3 步： 松开丝杆螺母安装板螺钉；

松开丝杆螺母安装板紧固螺钉 3-M5X10，并取下 J3 轴同步带，然后取出 J4 轴同步带进行维护。

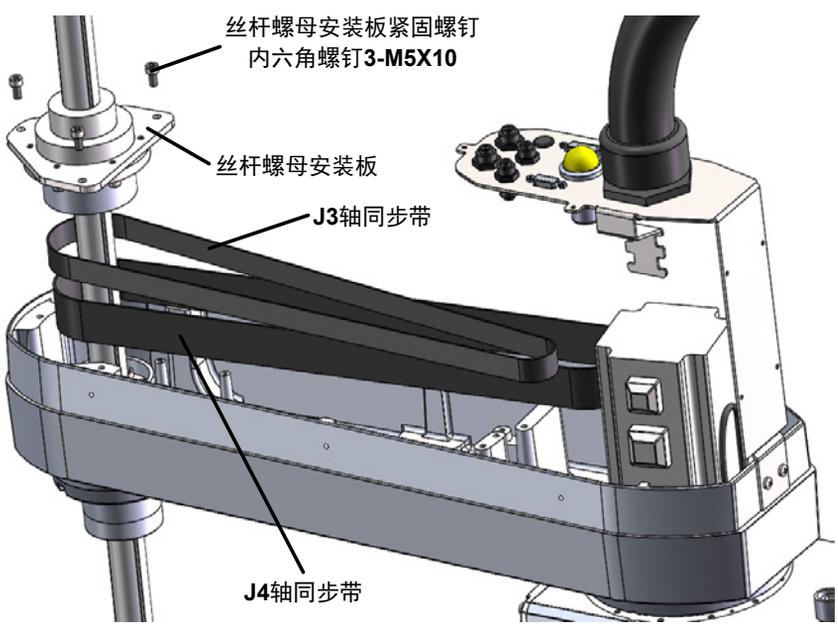


图 4-33 松开丝杆螺母安装板螺钉（IRS100-20 系列）

第 4 步： 拧紧丝杆螺母螺钉；

将拆卸下来的 J3 轴同步带和 J4 轴同步带重新装入，拧紧丝杆螺母安装板内六角螺钉 3-M5X10。

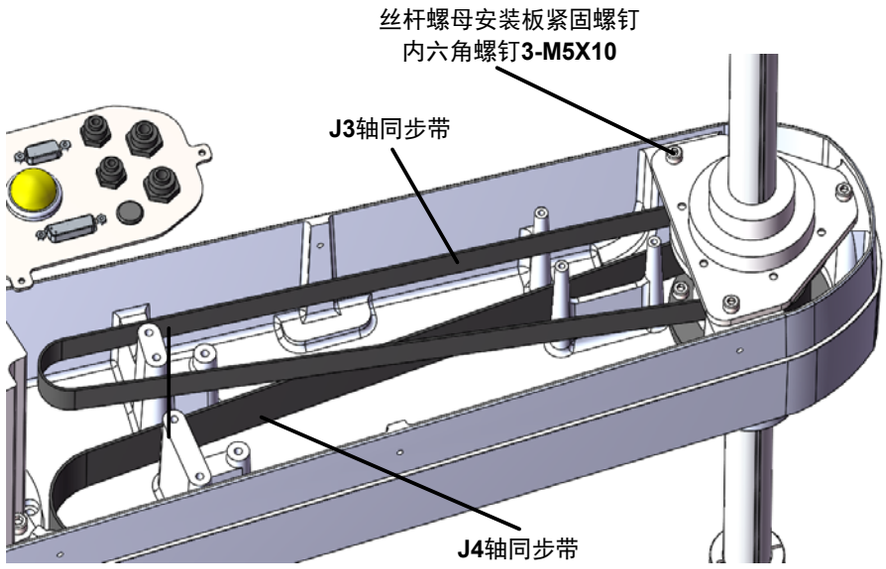


图 4-34 拧紧丝杆螺母螺钉（IRS100-20 系列）

第 5 步： 拧紧 J4 轴电机安装板螺钉；

安装同步带张紧力调整装置，拧动调整螺栓，使用张力仪进行检测，保证 J4 同步带检测频率 85-95Hz，然后拧紧 J4 轴电机安装板螺钉。

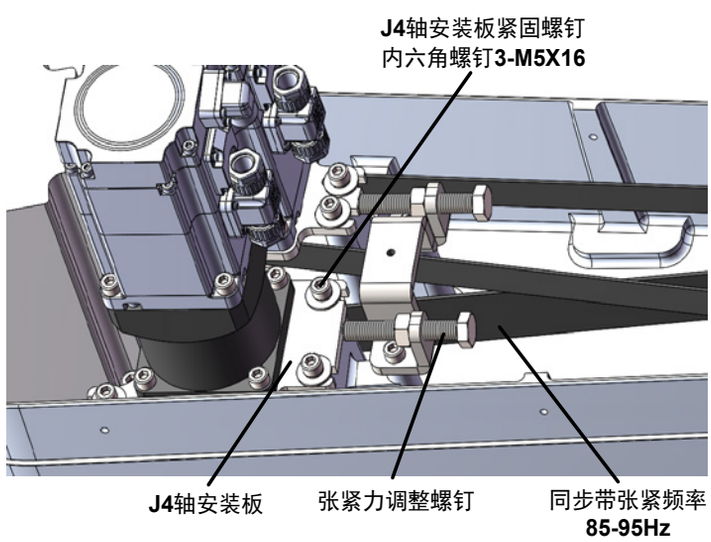


图 4-35 拧紧 J4 轴电机安装板螺钉（IRS100-20 系列）

第 6 步： 装好 J3 轴电机；

按照*“4.3.1 J3 轴同步带维护”*中的 IRS100-20 系列第 6 步，对 J3 轴的皮带进行重新装配即可。

（参考：*“图 4-25 张紧 J3 轴同步带（IRS100-20 系列）”*）

第 7 步： 还原小臂外壳，，对小臂外壳的 13 颗螺钉进行拧紧。

4.4.2 J4 轴电机维护

■ IRS100 系列

第 1 步： 取下 J4 轴电机组件；

按照*“4.4.1 J4 轴同步带维护”*中 IRS100 系列第 1~3 的步骤先取下 J4 轴电机组件；

（参考：*“图 2-1 小臂外壳拆卸与安装（IRS100 系列）”“图 4-29 取下 J3 轴电机组件（IRS100 系列）”“图 4-30 松开 J4 轴电机组件（IRS100 系列）”*）

第 2 步： 更换 J4 轴电机；

松开紧定螺钉 2-M4X5，取下 J4 同步带轮，拆下平键，松开紧固螺钉内六角螺钉 2-M4X10，取下 J4 轴电机安装板。

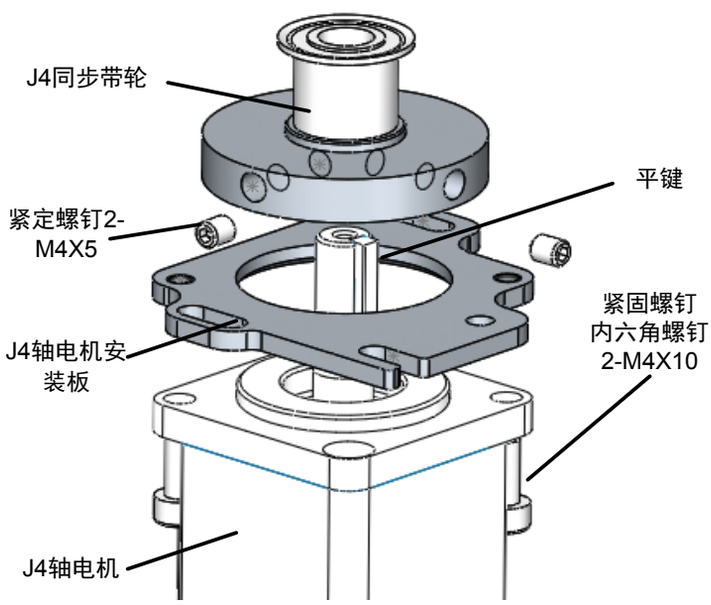


图 4-36 更换 J4 轴电机（IRS100 系列）

还原 J4 轴电机时，请按照与拆卸相反的步骤进行安装。

■ IRS100-20 系列

第 1 步： 拆卸小臂外壳；

按照*“4.2.1 J2 轴电机与减速机维护”*中 IRS100-20 系列第 1 步进行操作，抬起并取下小臂外壳。

（参考：*“图 2-2 小臂外壳拆卸与安装（IRS100-20 系列）”*）

第 2 步： 松开行星减速机与电机输出轴的紧固螺钉；

取下电机输出轴固定孔的橡胶壳，松开电机轴固定螺钉。如果螺钉处于不合适的位置，则需要轻轻转动电机轴，使螺钉头处于可操作的位置。

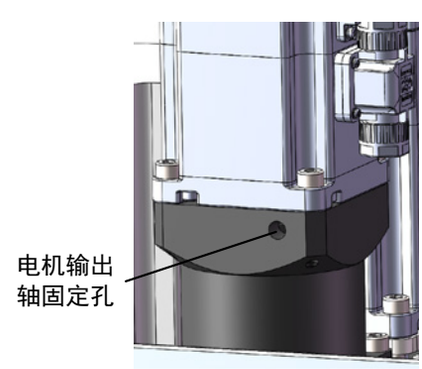


图 4-37 松开行星减速机与电机输出轴的紧固螺钉（IRS100-20 系列）

第 3 步： 更换 J4 轴电机；

松开 J4 电机与行星减速机的安装螺钉（内六角螺钉 4-M5X16），取下 J4 轴电机。

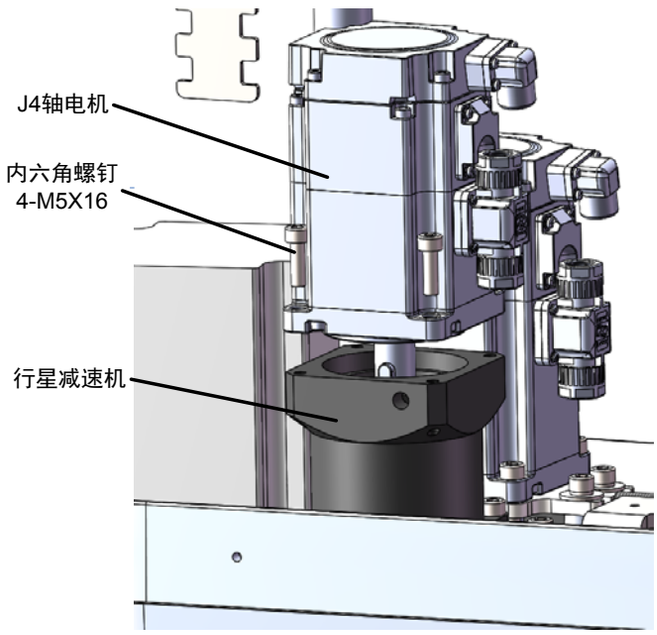


图 4-38 松开 J4 电机紧固螺钉（IRS100-20 系列）

还原 J4 轴电机时，请按照与拆卸相反的步骤进行安装。

4.4.3 J4 轴行星减速机维护

■ IRS100-20 系列

第 1 步： 取下 J4 轴电机组件；

按照*“4.4.1 J4 轴同步带维护”*中 IRS100-20 系列第 1~3 的步骤先取下 J4 轴电机组件；

（参考：*“图 2-2 小臂外壳拆卸与安装（IRS100-20 系列）”“图 4-21 拆卸同步带张紧力调整装置（IRS100-20 系列）”“图 4-32 松开 J3 轴和 J4 轴电机组件螺钉（IRS100-20 系列）”*）

第 2 步： 拆卸 J4 轴电机；

松开内六角螺钉 4-M5X16，拆下 J4 轴电机。

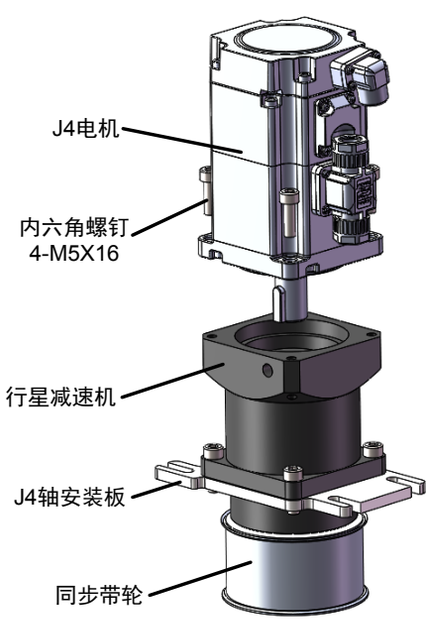


图 4-39 拆卸 J4 轴电机（IRS100-20 系列）

第 3 步： 拆卸同步带轮；

松开内六角螺钉 6-M4X16，拆下同步带轮。

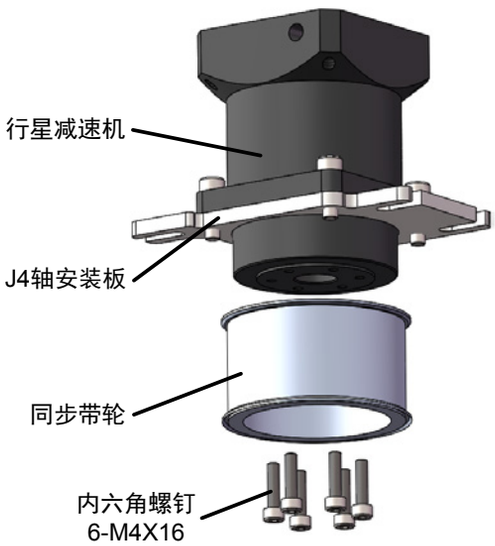


图 4-40 拆卸同步带轮（IRS100-20 系列）

第 4 步： 拆卸 J4 轴安装板。

松开内六角螺钉 4-M5X16，拆下 J4 轴安装板。

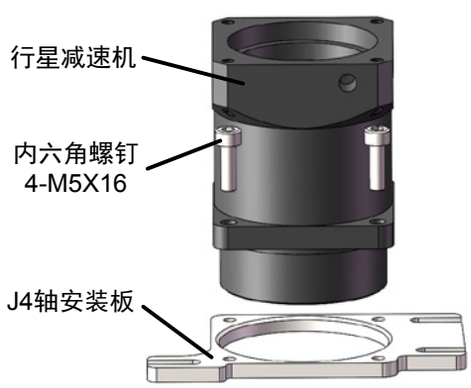


图 4-41 拆卸 J4 轴安装板（IRS100-20 系列）

第 5 步： 重新装配还原。

还原 J4 轴行星减速机时，请按照与拆卸相反的步骤进行安装。

4.5 滚珠丝杠花键保养

滚珠丝杆花键润滑脂加注：

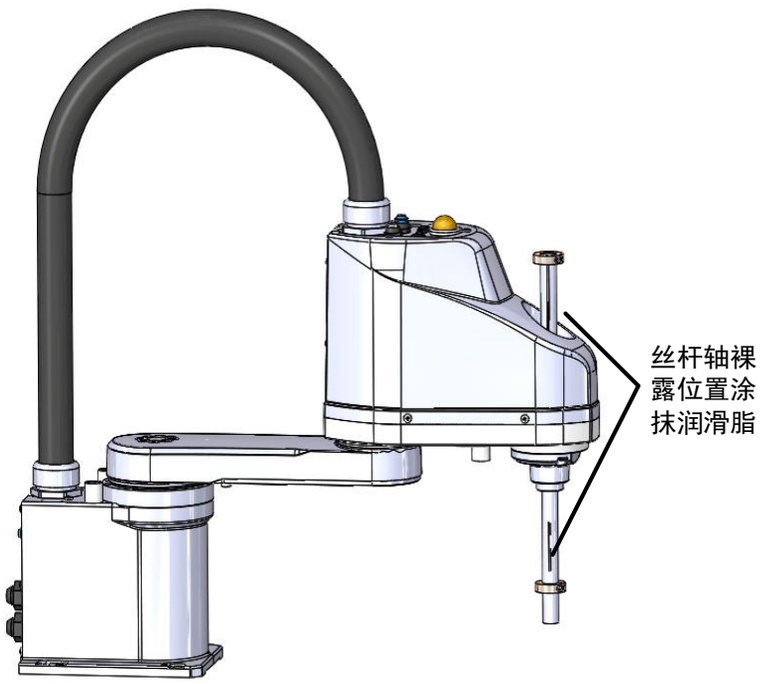


图 4-42 滚珠丝杆花键润滑脂加注

用干净抹布擦净丝杆轴表面，并在丝杆轴裸露的位置涂抹润滑脂（丝杆润滑油 -MUL TEMP LRL No.3）。

4.6 机器人线束维护

危 险

- 严禁非专业人员进行设备安装、接线、保养维护、检查或部件更换！
- 请勿在通电状态下装卸电机连接器，否则可能导致机器人进行异常动作，造成严重的安全问题。此外，如果在通电状态下进行维护作业，可能有触电危险。
- 请勿对本手册未记载的部位进行拆卸，或按照与记载不同的方法进行维护。
- 请在安全护栏外确认更换部件后机器人的动作，否则可能因机器人进行异常的动作造成严重的安全问题。
- 进入正常运转前，请确认紧急停止开关与安全门开关工作正常，否则系统发生紧急状况时将无法发挥安全功能，可能会导致重伤或重大损害。

警 告

- 设备出现故障或损坏时，由专业人员按照维修指导对设备进行故障排除和维修，并做好维修记录。
- 维护时请勿使异物进入到设备内部与连接端子中。
- 更换设备后，请务必重新对设备接线检查与参数设置。
- 请将线缆连接牢固，请勿在线缆上放置重物，请勿强行弯曲或拉拽线缆，否则可能造成线缆损坏、断线或接触不良，有触电的危险或导致系统故障。
- 接线时使用到的线缆必须符合相应的线径和屏蔽等要求，使用屏蔽线缆时屏蔽层需要单端可靠接地！
- 接线时请勿弄错连接关系，否则系统将无法正常工作，还可能造成安全问题。
- 接线完成后，请确保设备内部没有掉落的螺钉或裸露线缆。

4.6.1 更换底座至小臂的连接线束

1） 底座至小臂连接电缆单元的拆卸

■ IRS100 系列

请按照以下步骤进行拆卸：

第 1 步： 将控制柜的电源设为 ON，将控制柜设为上电状态。

第 2 步： 挂出“作业中”标识。请挂出“作业中”标识，以防止其他作业者操作控制柜或操作面板。

第 3 步： 进入安全防护栏内。

第 4 步： 按下抱闸按钮开关，解除第三轴电机的制动，将轴降到下限位置。确保留有足够的空间，防止夹具末端碰撞外围装置等。

第 5 步： 关闭控制柜的电源，并断开连接至控制柜上的动力输出和编码器线缆的连接器。

第 6 步： 取下固定底座前盖板四个对角的螺丝（十字盘头螺钉 4-M4X8），拆下底座前盖板。

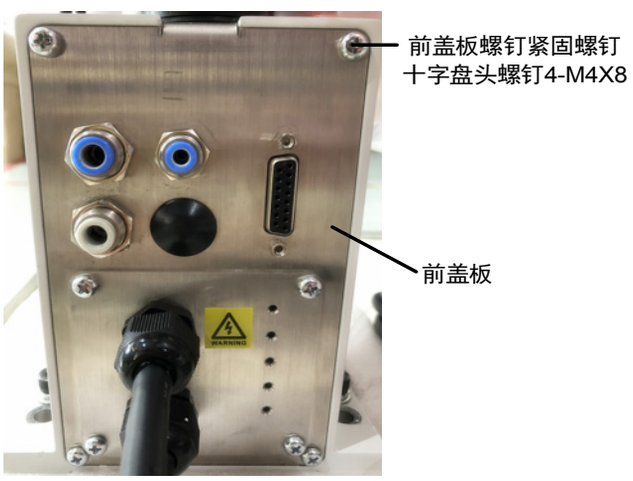


图 4-43 拆下底座前盖板（IRS100 系列）

第 7 步： 将底座内部线缆轻轻从底座里面拉出，断开底座内部对接的动力和编码器连接器，松开连接至底座盖板上的三根气管和 DB-15 接头。

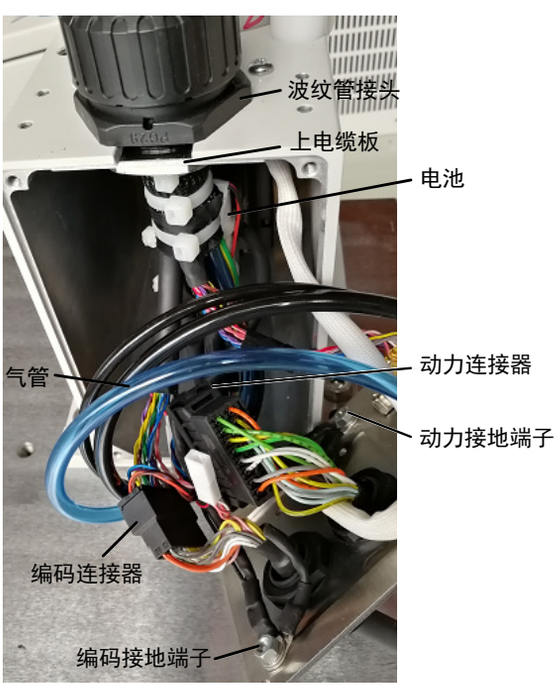


图 4-44 底座内部线缆（IRS100 系列）

备注：D-sub 电缆的安装六角镀镍螺丝（4-40 UNC）非常小，请务必妥善保管螺丝。按压气管接头的卡环，然后可拉出空气管（ø6X2、ø4X1）。

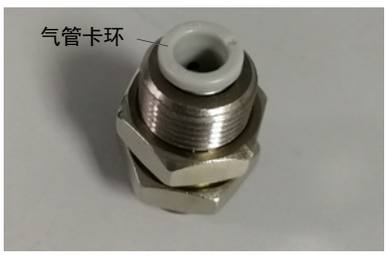


图 4-45 气管接头卡环

第 8 步： 拆下固定在底座前盖板上的接地端子。

第 9 步： 剪断绑扎在底座上电缆板上的两根固定电池的扎带，断开电池与线缆连接的白色 2PIN 连接器，将电池从上电缆板上取出。

（参考：*“图 4-44 底座内部线缆（IRS100 系列）”*）

第 10 步： 按照*“2.1 顶部外壳的拆卸与安装”*中 IRS100 系列步骤拆下顶部外壳。

（参考：*“图 2-1 顶部外壳拆卸与安装（IRS100 系列）”*）

第 11 步： 剪断电机上用于固定线缆的扎带。

第 12 步： 用十字螺丝刀将二轴、三轴和四轴动力和编码器接头四个脚固定的十螺丝（十字盘头 4-M2X8.5）松开，轻轻将接头从电机端公座上取出，注意要避免出现弯针情况。断开抱闸按钮和指示灯连接器，拆开固定在小臂上的接地线。

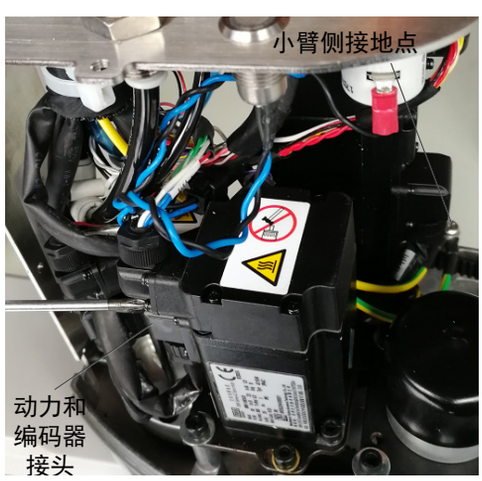


图 4-46 小臂接地点和动力 / 编码器接头（IRS100 系列）

图 4-47 编码器 / 动力接头

第 13 步： 拆开小臂端固定小臂钣金的螺丝（十字盘头螺钉 4-M4X8），可将线缆从小臂上取下。

第 14 步： 用扳手松开固定至底座上的波纹管接头螺母，将线缆从底座拉出，用十字螺丝刀将一轴编码器接头四个脚固定的螺丝（十字盘头 4-M2X8.5）取下来，轻轻将接头从电机端公座上取出，注意要避免出现弯针情况。

第 15 步： 将线缆从本体上取下。

■ IRS100-20 系列

请按照以下步骤进行拆卸：

第 1 步： 将控制柜的电源设为 ON，将控制柜设为上电状态。

第 2 步： 挂出“作业中”标识。请挂出“作业中”标识，以防止其他作业者操作控制柜或操作面板。

第 3 步： 进入安全防护栏内。

第 4 步： 按下抱闸按钮开关，解除第三、四轴电机的制动，将轴降到下限位置。确保留有足够的空间，防止夹具末端碰撞外围装置等。

第 5 步： 关闭控制柜的电源，并断开连接至控制柜上的动力输出和编码器线缆的连接器。

第 6 步： 取下固定底座前盖板四个对角的螺丝（十字盘头螺钉 4-M4X8），拆下底座前盖板。

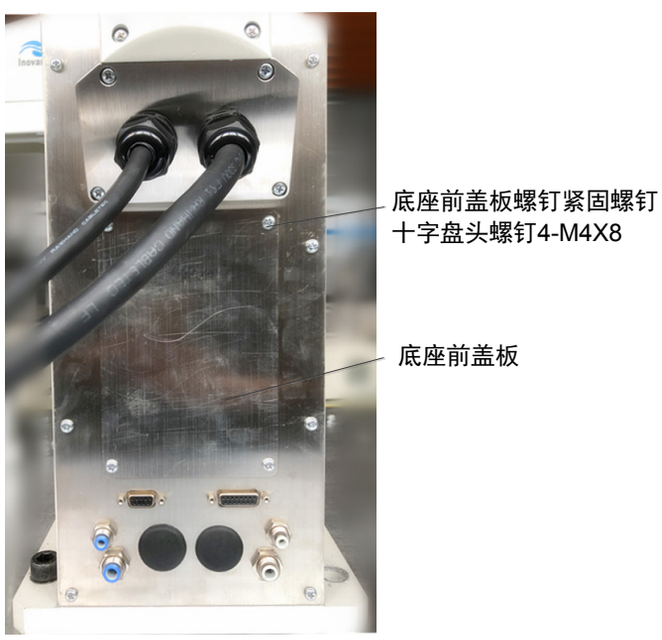


图 4-48 拆下底座前盖板（IRS100-20 系列）

第 7 步： 断开底座内部对接的动力和编码器连接器，松开连接至底座前盖板上的四根气管和 DB-9、DB-15 接头。

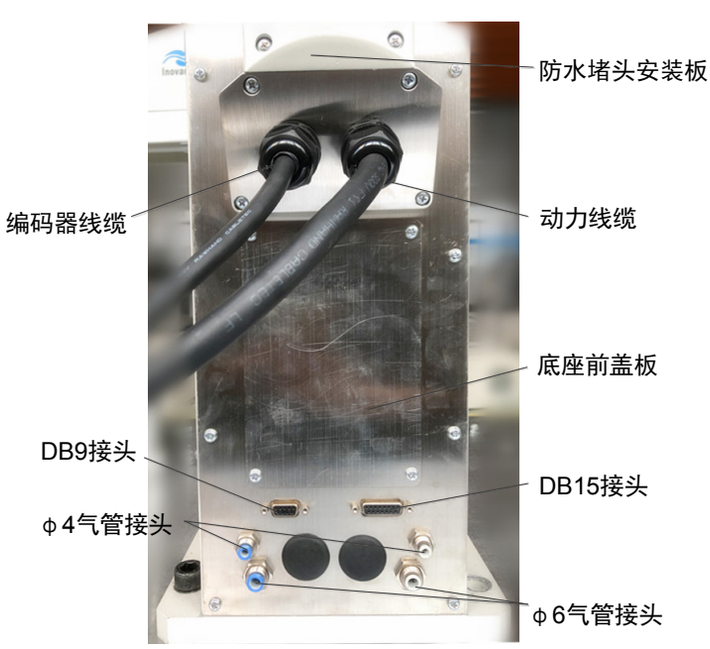


图 4-49 底座前盖板（IRS100-20 系列）

第 8 步： 拆开底座延伸臂的后盖，避免刮花表面喷漆。

第 9 步： 剪断绑扎在底座延伸臂固定线束和电池的扎带，断开动力和编码连接器断开，将动力和编码器接地线的 M4 螺丝松开，断开电池与线缆连接的白色 2PIN 连接器，将电池取出。

第 10 步： 按照*“2.1 顶部外壳的拆卸与安装”*中 IRS100-20 系列步骤拆下小臂端外壳。

（参考：*“图 2-2 顶部外壳拆卸与安装（IRS100-20 系列）”*）

第 11 步： 剪断电机上用于固定线缆的扎带。

第 12 步： 用十字螺丝刀将二轴、三轴和四轴动力和编码器接头四个脚固定的十螺丝（十字盘头 4-M2X8.5）松开，轻轻将接头从电机端公座上取出，注意要避免出现弯针情况。断开抱闸按钮和指示灯连接器，拆开固定在小臂上的接地线。

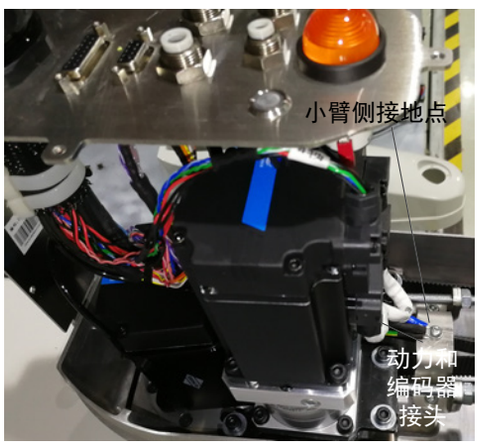


图 4-50 小臂接地点和动力 / 编码器接头（IRS100-20 系列）

第 13 步： 拆开小臂端固定小臂钣金的螺丝（十字盘头螺钉 4-M4X8），可将线缆从小臂上取下。

第 14 步： 用扳手松开固定至底座上的波纹管接头螺母，将线缆从底座拉出，用十字螺丝刀将一轴编码器接头四个脚固定的螺丝（十字盘头 4-M2X8.5）取下来，轻轻将接头从电机端公座上取出，注意要避免出现弯针情况。

第 15 步： 将线缆从本体上取下。

2） 底座至小臂连接电缆单元的安装

■ IRS100 系列

请按照与拆卸相反的操作步骤安装新的机器线束。

第 1 步： 将底座处一轴的编码连接器对接在电机上，用螺丝刀将四个对角的螺丝（十字盘头 4-M2X8.5）锁紧。

第 2 步： 将线缆穿过底座的过孔，并用扳手将波纹管接头螺母（LNB-PG36）锁紧在底座上。

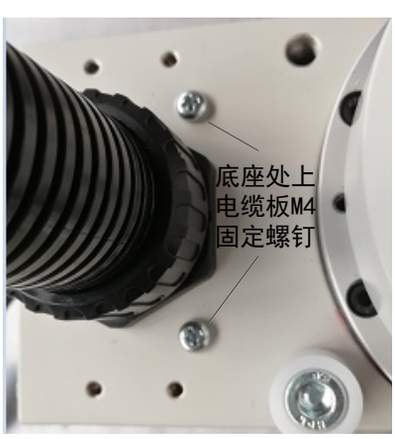


图 4-51 波纹管接头螺母（IRS100 系列）

第 3 步： 将电池自带接头与线缆上接头对接，并将电池放在上电缆板钣金上，用与拆卸第 9 步相同规格的扎带绑扎。

第 4 步： 将底座内动力和编码器连接器对接，并将动力和编码器的地线一一对应锁紧在底座前盖板上。

第 5 步： 将三根气管和 DB-15 接头安装前盖板上。

第 6 步： 将线缆慢慢推入底座内，盖上底座盖板并将四个对角的螺丝（十字盘头螺钉 4-M4X8）拧紧固定。

（参考：*“图 4-43 拆下底座前盖板（IRS100 系列）”*）

备注：盖上前盖板的过程中注意不要夹到线缆，否则容易造成短路触电、断路等问题。

第 7 步： 用螺丝刀将小臂端线缆钣金安装至小臂出两个螺丝（十字盘头螺钉 2-M4X8）拧紧，固定小臂钣金。

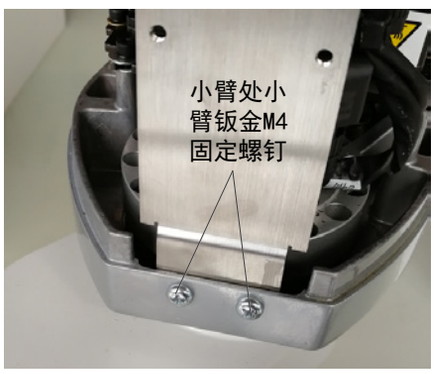


图 4-52 小臂钣金固定螺钉（IRS100 系列）

第 8 步： 将二轴、三轴和四轴动力和编码器接头与对应各轴电机上的插座轻轻对接，用螺丝刀将连接器四个脚的螺丝（十字盘头 4-M2X8.5）拧紧。

（参考：*“图 4-47 编码器 / 动力接头”*）

第 9 步： 接好指示灯和抱闸按钮的连接器。

第 10 步： 将接地线锁紧在小臂上的接地位置。

第 11 步： 使用与拆卸第 11 步相同的扎带在原位置将电缆绑扎固定。

第 12 步： 盖上小臂端外壳并用螺丝（十字盘头螺钉 4-M4X8）将外壳固定。

（参考：*“图 2-1 小臂外壳拆卸与安装（IRS100 系列）”*）

第 13 步： 调整机器人 1 关节以外的原点，如果拆卸过程中电池有断开，则需调整所有轴的原点。

第 14 步： 离开安全防护栏。

第 15 步： 将连接至控制柜上的动力和编码器线缆的连接器对接好，并用螺丝刀锁紧编码器的 DB-25 接头。

■ IRS100-20 系列

请按照与拆卸相反的操作步骤安装新的机器线束。

第 1 步： 将线缆穿过底座延伸臂端盖的过孔，并用扳手将波纹管接头螺母（LNB-PG29）锁紧在端盖上，波纹管接头螺母处点螺纹胶（乐泰 243）紧固。

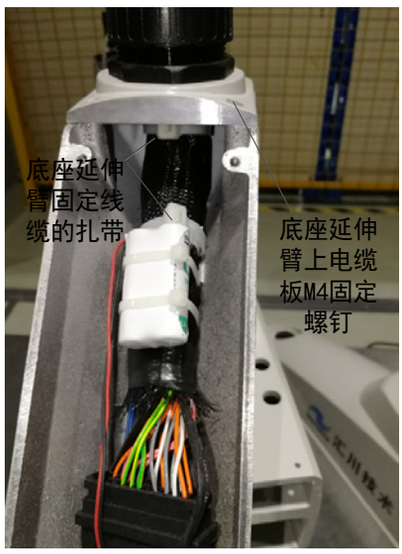


图 4-53 底座延伸臂（IRS100-20 系列）

第 2 步： 将底座处一轴的编码连接器对接在电机上，用螺丝刀将四个对角的螺丝锁紧。

第 3 步： 将电池自带接头与线缆上接头对接，并将电池放在上电缆板钣金上，用与拆卸第 9 步相同规格的扎带绑扎。

第 4 步： 将底座内动力和编码器连接器对接，并将动力和编码器的地线一一对应锁紧在底座前盖板上。

第 5 步： 将四根气管和 DB-9、DB-15 接头安装前盖板上。

第 6 步： 盖上底座盖板并将四个对角的螺丝拧紧固定，盖上底座延伸臂的后盖并将螺丝拧紧固定。

第 7 步： 用螺丝刀将小臂端线缆钣金安装至小臂出两个螺丝（十字盘头螺钉 2-M4X8）拧紧，固定小臂钣金。

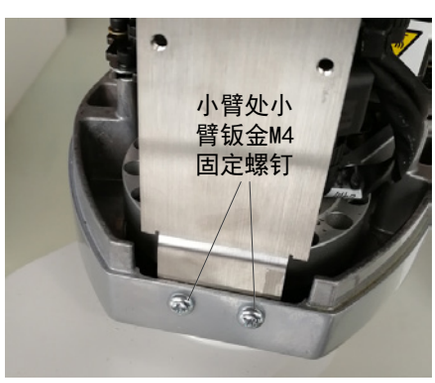


图 4-54 小臂钣金固定螺钉（IRS100-20 系列）

第 8 步： 将二轴、三轴和四轴动力和编码器接头与对应各轴电机上的插座轻轻对接，用螺丝刀将连接器四个脚的螺丝（十字盘头 4-M2X8.5）拧紧。

（参考：*“图 4-47 编码器 / 动力接头”*）

第 9 步： 接好指示灯和抱闸按钮的连接器。

第 10 步： 将接地线锁紧在小臂上的接地位置。

第 11 步： 使用与拆卸第 11 步相同的扎带在原位置将电缆绑扎固定。

第 12 步： 盖上小臂端外壳并用螺丝（十字盘头螺钉 4-M4X8）将外壳固定。

（参考：*“图 2-2 小臂外壳拆卸与安装（IRS100-20 系列）”*）

第 13 步： 调整机器人 1 关节以外的原点，如果拆卸过程中电池有断开，则需调整所有轴的原点。

第 14 步： 离开安全防护栏。

第 15 步： 将连接至控制柜上的动力和编码器线缆的连接器对接好，并用螺丝刀锁紧编码器的 DB-25 接头。

4.6.2 更换本体至控制柜的连接线缆

1） 本体至控制柜连接线束单元的拆卸

■ IRS100 系列

请按照以下步骤进行拆卸：

第 1 步： 将控制柜断电。

第 2 步： 断开连接至控制柜上的动力和编码器线缆的连接器。

第 3 步： 挂出“作业中”标识。

请挂出“作业中”标识，以防止其他作业者操作控制柜或操作面板。

第 4 步： 进入安全防护栏内。

第 5 步： 取下四个对角的固定底座前盖板螺丝（十字盘头螺钉 4-M4X8），拆下底座前盖板。

（参考：*“图 4-43 拆下底座前盖板（IRS100 系列）”*）

第 6 步： 将底座内部线缆轻轻从底座里面拉出，用十字螺丝刀将一轴动力接头四个脚固定的螺丝（十字盘头 4-M2X8.5）松开，轻轻将接头从电机端公座上取出，注意要避免出现弯针情况，断开底座内部对接的动力和编码器连接器，拆下固定至底座前盖板上的接地端子。

第 7 步： 将电缆板固定在底座前盖板上的四个对角螺丝（十字盘头螺钉 4-M4X8）松开。

图 4-55 电缆板固定螺钉（IRS100 系列）

第 8 步： 取下本体至控制柜的连接线束。

■ IRS100-20 系列

请按照以下步骤进行拆卸：

第 1 步： 将控制柜断电。

第 2 步： 断开连接至控制柜上的动力和编码器线缆的连接器。

第 3 步： 挂出“作业中”标识。

请挂出“作业中”标识，以防止其他作业者操作控制柜或操作面板。

第 4 步： 进入安全防护栏内。

第 5 步： 取下底座延伸臂盖板螺钉（十字盘头螺钉 6-M4X8），拆下底座延伸臂盖板。

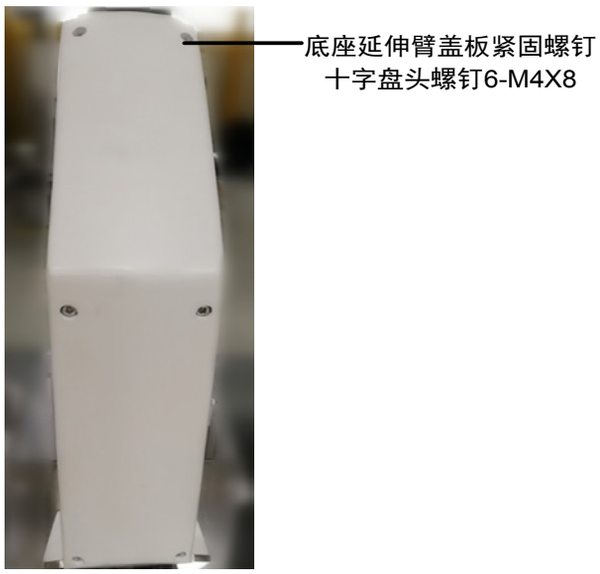


图 4-56 底座延伸臂盖板（IRS100-20 系列）

第 6 步： 用十字螺丝刀将一轴动力接头四个脚固定的螺丝（十字盘头 4-M2X8.5）松开，轻轻将接头从电机端公座上取出，注意要避免出现弯针情况，断开底座延伸臂上对接的动力和编码器连接器，拆下固定至底座盖板上的接地端子，剪断固定线缆的扎带。

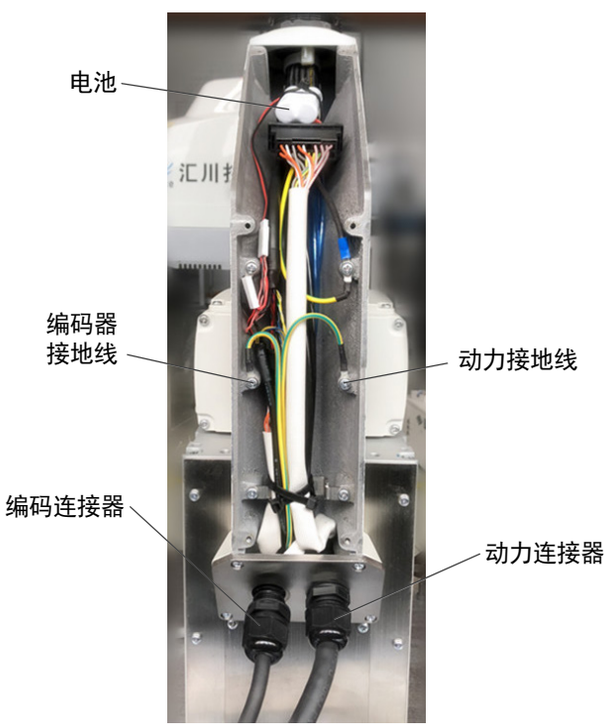


图 4-57 底座延伸臂内图示（IRS100-20 系列）

第 7 步： 将防水堵头安装板上的四个对角螺丝（十字盘头螺钉 4-M4X8）松开。

图 4-58 防水堵头安装板（IRS100-20 系列）

第 8 步： 取下本体至控制柜的连接线束。

2） 本体至控制柜连接线束单元的安装

■ IRS100 系列

请按照以下步骤进行安装：

第 1 步： 将电缆板的四个对角螺丝（十字盘头螺钉 4-M4X8）拧紧，固定在底座前盖板上。

第 2 步： 将一轴动力接头与电机上的插座轻轻对接，用螺丝刀将连接器四个脚的螺丝（十字盘头 4-M2X8.5）拧紧，接好底座内动力和编码器线缆的连接器。

（参考：*“图 4-47 编码器 / 动力接头”*）

第 3 步： 将动力和编码的接地端子一一对应锁在底座前盖板螺丝（十字盘头螺丝 M4X8）上。

第 4 步： 将线缆慢慢塞入底座内，盖上底座盖板并将四个对角的螺丝（十字盘头螺钉 4-M4X8）拧紧固定。

NOTE

◆ 盖上前盖板的过程中注意不要夹到线缆，否则容易造成短路触电、断路等问题。

第 5 步： 将连接至控制柜上的动力和编码器线缆的连接器对接好，并用螺丝刀锁紧编码器的 DB-25 接头。

■ IRS100-20 系列

请按照以下步骤进行安装：

第 1 步： 将防水堵头安装板的四个对角螺丝拧紧，固定在底座上。

第 2 步： 将一轴动力接头与电机上的插座轻轻对接，用螺丝刀将连接器四个脚的螺丝（十字盘头 4-M2X8.5）拧紧，接好底座内动力和编码器线缆的连接器。

（参考：*“图 4-47 编码器 / 动力接头”*）

第 3 步： 将动力和编码的接地端子一一对应所在底座延伸臂接地点上。

第 4 步： 盖上底座盖板并将四个对角的螺丝（十字盘头螺丝 M4X8）拧紧固定，盖上底座延伸臂的后盖并将螺丝拧紧固定。

NOTE

◆ 盖上延伸臂后盖的过程中注意不要夹到线缆，否则容易造成短路触电、断路等问题。

第 5 步： 将连接至控制柜上的动力和编码器线缆的连接器对接好，并用螺丝刀锁紧编码器的 DB-25 接头。

4.7 电池更换与维护

警 告

◆ 除进行维护作业时以外，请不要打开控制柜的盖子。控制柜内部有高电压的充电部位，即使在电源关闭的状态下也有触电的危险。

◆ 请务必在关闭控制柜与相关装置电源并拔出电源插头之后进行更换作业。如果在通电的状态下进行作业，则可能会导致触电或故障。

◆ 请勿在保持电源打开的状态下装卸电动机连接器。否则可能会导致机械手进行异常动作，非常危险。另外，如果在通电的状态下进行作业，则可能会导致触电或故障。

◆ 在得到停电通知时，要预先关断机器人的主电源及气源。

◆ 突然停电后，要赶在再次来电之前预先关闭机器人的主电源开关，并及时取下夹具上的工件。

◆ 请勿在保持电源打开的状态下装卸电机连接器。否则可能会导致机器人进行异常动作，非常危险。另外，如果在通电的状态下进行作业，则可能会导致触电或故障

◆ 通过拔下电源插头来关闭机器人系统的电源。请务必将 AC 电源电缆连接到电源插头上，切勿直接连到工厂电源上。

◆ 请务必在关闭控制柜与相关装置电源并拔出电源插头之后进行更换作业。 如果在通电的状态下进行作业，则可能会导致触电或故障。

◆ 请充分注意锂电池的使用。如果采取下述错误使用方法，则可能会导致发热、漏液、爆炸或起火等，非常危险。另外，也可能会造成安全问题。
1. 充电；2. 加压变形；3. 拆卸；4. 短路；5. 装反；6. 加热；7. 火烧；8. 焊接；9. 强制放电；

◆ 废弃电池时，请咨询专业处理公司，或根据各国各地区的相关法律法规进行废弃。

◆ 废弃时，即使是已使用完毕的电池，也请务必进行端子绝缘。如果接触其它金属或电池端子，则可能会形成短路，从而导致发热、漏液、爆炸或起火。

如果锂电池耗尽，打开控制柜电源时（软件启动时），则会发生警告电压过低的错误， 电机的位置数据消失，需要对所有关节进行原点位置调整。

锂电池的使用寿命为 1.5 年。即使在机器人连续通电的状态下，也请每 1.5 年更换一次锂电池。

请务必使用本公司指定的锂电池。请正确进行连接电池，不要弄错极性。

NOTE

◆ 锂电池每组 2 个，更换电池时请依次进行更换。（同时取下 2 个电池时，电机的位置数据消失，需要对所有关节进行原点位置调整。）

电池更换步骤：

■ IRS100 系列

第 1 步： 松开前盖板前盖板螺钉紧固螺钉十字盘头螺钉 4-M4X8，取下前盖板。

（参考：*“图 4-58 拆下底座前盖板（IRS100 系列）”*）

第 2 步： 换电池

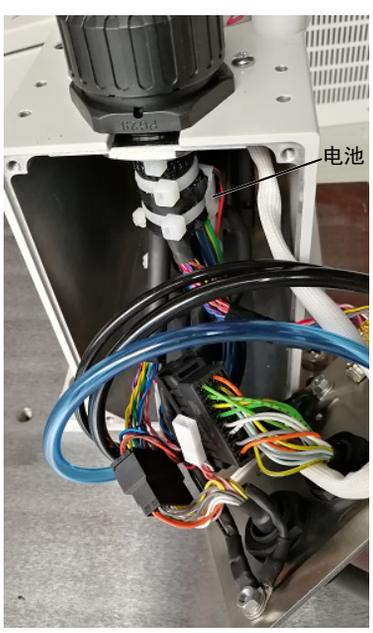


图 4-59 电池位置示意（IRS100 系列）

剪开扎带，更换电池，并扎上扎带，缕好内部电线和气管。并装好前盖板。

■ IRS100-20 系列

第 1 步： 取下底座延伸臂盖板螺钉（十字盘头螺钉 6-M4X8），拆下底座延伸臂盖板。

（参考：*“图 4-71 底座延伸臂盖板（IRS100-20 系列）”*）

第 2 步： 换电池

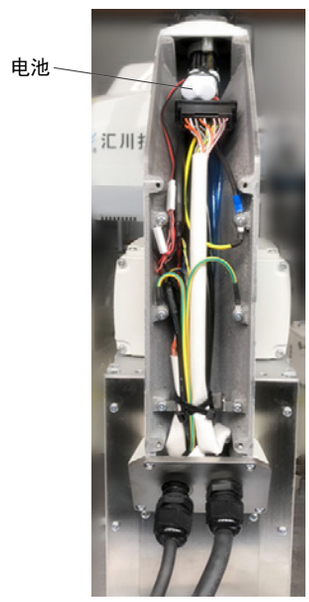


图 4-60 电池位置示意（IRS100-20 系列）

剪开扎带，更换电池，并扎上扎带，缕好内部电线和气管。并装好延伸臂盖板。

4.8 零点调整

4.8.1 零点调整方法

零点是机器人工作的参考点及基准点。如果更换机器人的部件（电机、减速机、同步皮带、线缆等），电机侧保存的零点与控制器侧保存的零点之间则会产生偏差，无法进行正确的定位。 因此，部件更换之后，进行零点调整。

NOTE

◆ 零点调整后，机器人绝对精度与出厂时的绝对精度可能存在偏差。

危 险

◆ 请务必对系统设置安全护栏，防止人员进入系统的动作区域内，否则可能造成严重的安全问题。

◆ 操作前，请确认安全护栏内侧没有人。系统动作期间，请勿进入其动作区域内，否则可能造成严重的安全问题。

◆ 在示教模式下操作机器人系统，虽然动作处于受限状态（低速、低功率），这样在一定程度上可以保证作业人员的安全，但在机器人进行意想不到的动作时，可能会造成严重的安全问题。

4.8.2 零点调整步骤

PC 版的示教软件及手持示教盒均配备有用于进行零点调整的操作界面。下面以 PC 版的示教软件为例进行说明，手持示教盒操作类似。

需对机器人的作业点进行坐标计算时，第 2 关节的精度是非常重要的。进行第 2 关节 的零点调整时，根据向导，“利用右手腕 / 左手腕法则进行零点调整”。详情请参阅 “第 2 关节的正确零点调整”。从机器人结构上讲，不能进行仅限于第 4 关节的零点调整，第 4 关节与第 3 关节请同时进行。

1） 登录用户权限

■ i. 在示教软件或手持示教盒主界面，点击【用户设置】快捷键，打开用户设置界面。

■ ii. 在【密码输入框】输入密码，并点击【登录按钮】。

2） 切换到绝对零点设置界面

■ i. 在示教软件或手持示教盒主界面，选择【设置】-【零点设置】-【绝对零点】，打开绝对零点设置界面。

3） 手动将机器人各个关节移动到零点附近

■ i. 可以使用手持示教盒上的摇杆或者示教软件上的操作界面，将机器人移动到零点附近。参考“示教软件手册 / 编程手册”

■ ii. 也可以在未上使能的状态下，用手推动机器人，使其到达零点附近。

4） 切换到紧急停止状态

■ i. 对于 PC 版的示教软件，点击虚拟的【急停按钮】；对于手持示教盒，按下红色急停按钮；

■ ii. 此时示教软件（或手持示教盒显示屏）右上角的状态指示灯显示为“急停状态”（红色状态）。

5） 获取零点信息并保存

■ i. 移动到零点位置时，点击【取当前值】按钮，获取机器人处于零点位置时的编码器脉冲数；

■ ii. 点击【保存】按钮，完成零点调整。

4.8.3 各关节零点位置

1） J1 轴零点位置： 与机器人坐标系的 Y 坐标轴重叠的位置

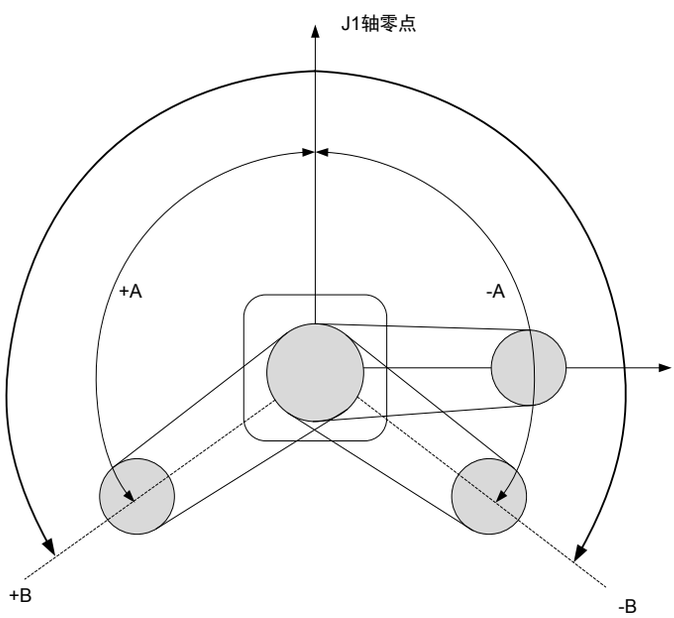


图 4-61 J1 轴零点位置示意图

2） J2 轴零点位置：与机器人坐标系的 Y 坐标轴重叠的位置

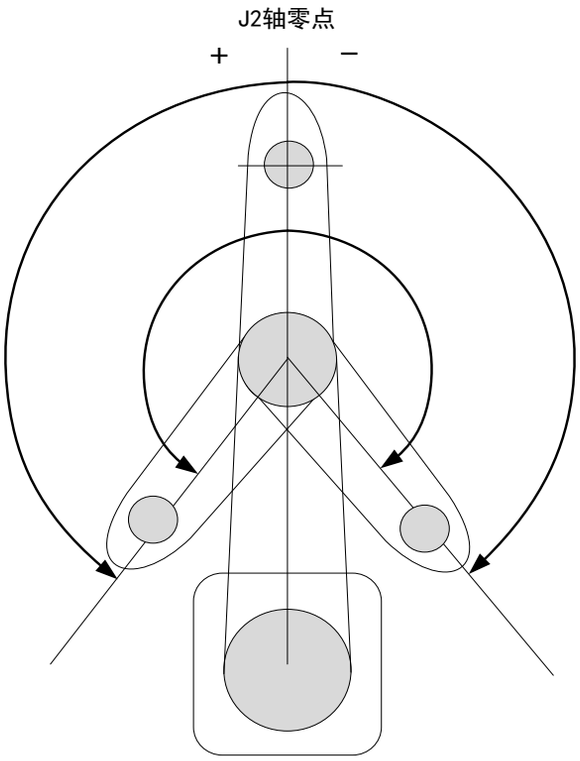


图 4-62 J2 轴零点位置示意图

3） J3轴零点位置：J3 轴在上限位置时为零点位置，如下图所示：

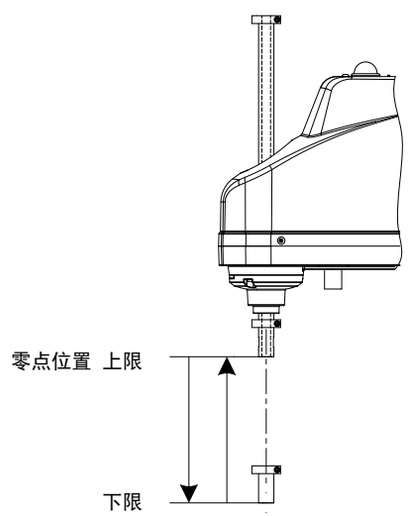


图 4-63 J3 轴零点位置示意图

J3 轴的高度因机器人的规格而异，可动区域的上限位置请参考各机型机械篇手册。

4） J4 轴零点位置： 轴的平面（或上下机械挡块的槽）朝向第 2 机械臂顶端方向的位置

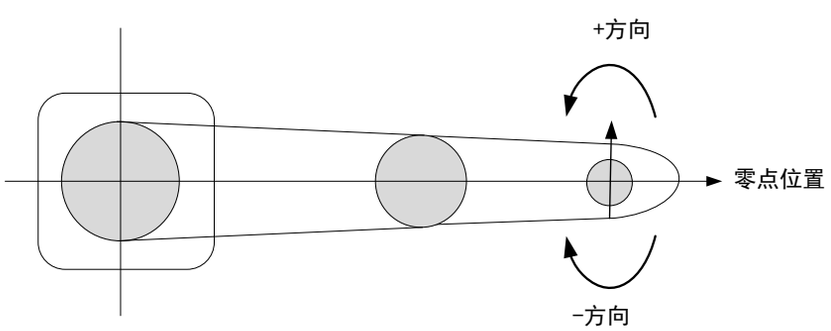


图 4-64 J4 轴零点位置示意图

4.8.4 第 2 关节的零点调整方法

需对机器人的作业点进行坐标计算时，第 2 关节的精度是非常重要的。如果在“零点调整步骤”中进行第 2 关节的零点调整，则通过向导“利用右手腕左手腕法则进行零点调整”。

步骤：

1） 进行零点调整时，基准点为滚珠丝杠花键轴的中心。

■ 当末端执行器的中心偏离滚珠丝杠花键轴的中心时，需要拆下末端执行器进行零点调整。

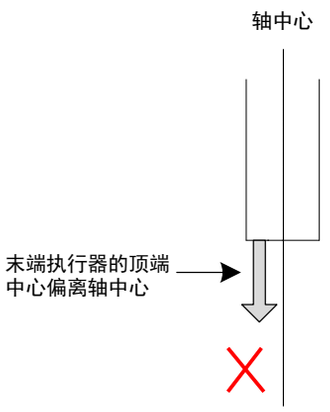


图 4-65 偏离轴中心示意图

2） 在轴顶端侧制作如下图所示的零点调整夹具（例），以明确轴中心。将变更右手腕 / 左手腕姿势时易于确认的位置作为目标点，然后在装置侧打上 × 号。

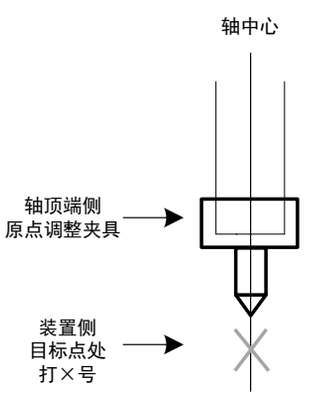


图 4-66 调整轴中心示意图

3） 利用右手腕 / 左手腕进行零点调整

■ 1、利用右手腕（或左手腕）姿势将机器人末端调整到目标点位置，在示教器主界面 -【设置】-【零点设置】-【绝对零点】中，点击【取当前值】按钮，记录当前状态下第 2 关节的度数，记为 Enc1。

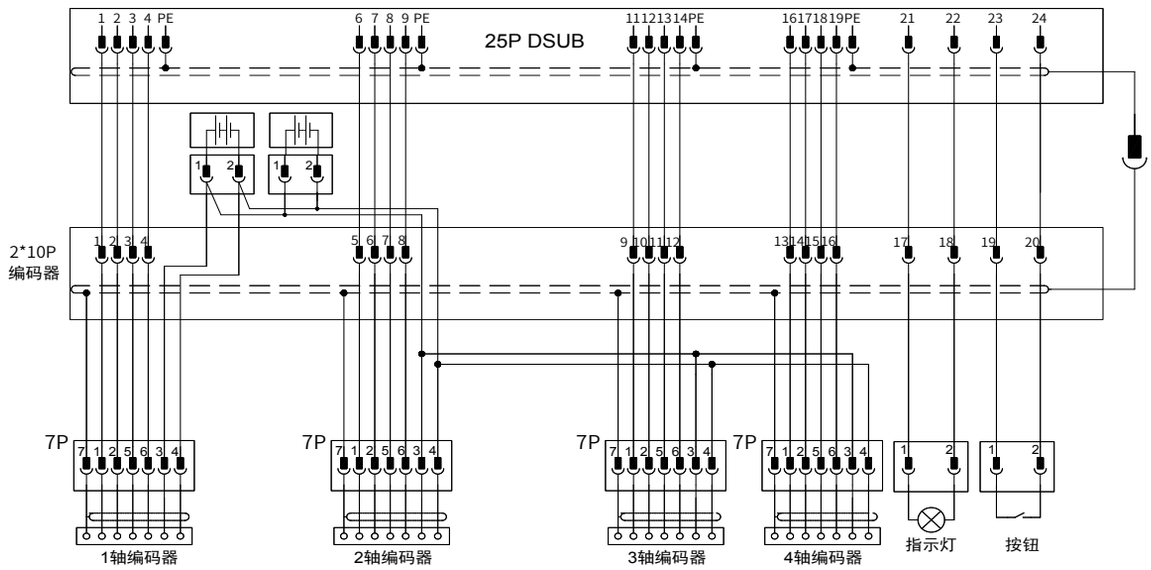
■ 2、手动利用左手腕（或右手腕）姿势将机器人调整到目标点位置，、在示教器主界面 -【设置】-【零点设置】-【绝对零点】中，点击【取当前值】按钮，记录当前状态下第 2 关节的度数，记为 Enc2。

■ 3、示教器主界面 -【设置】-【零点设置】-【绝对零点】中，点击【默认值】按钮，获取当前保存的零点信息。计算 (Enc1+Enc2)/2，并将计算结果填到第 2 关节的本界面下的第 2 关节输入框，点击【保存按钮】，完成第 2 关节的零点调整。

4） 拆下末端执行器调整零点之后，安装末端执行器，将机器人移动到示教点，确认位置偏移。出现位置偏移时，请对末端执行器安装位置进行微调，然后再次进行示教。

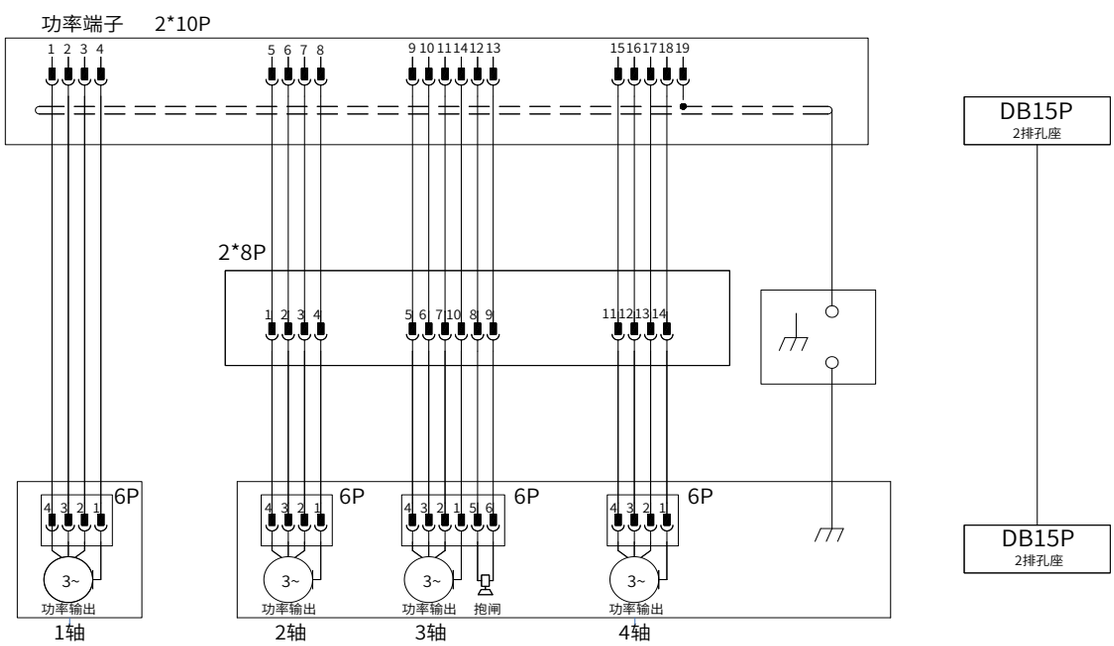
4.9 机器人本体电气连接图

■ IRS100 系列



编码器引脚	7	1	2	5	6	3	4
定义	屏蔽	PS+	PS-	5V	GND	BT+	BT-

图 4-67 机器人本体电气连接图（编码器连接）

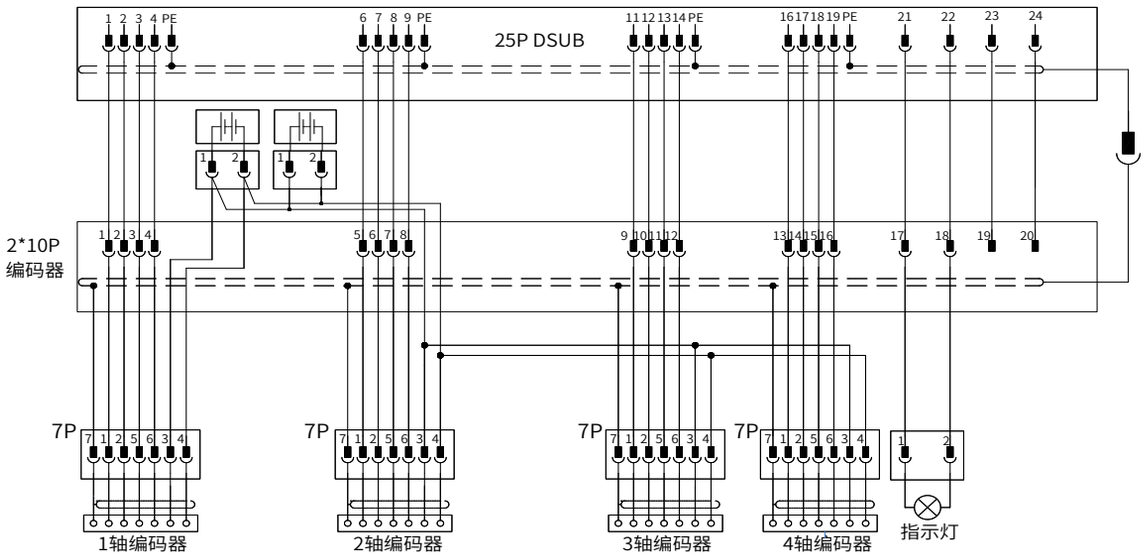


功率输出引脚	4	3	2	1	5	6
定义	U	V	W	PE	BK+	BK-

图 4-68 机器人本体电气连接图（动力线连接）

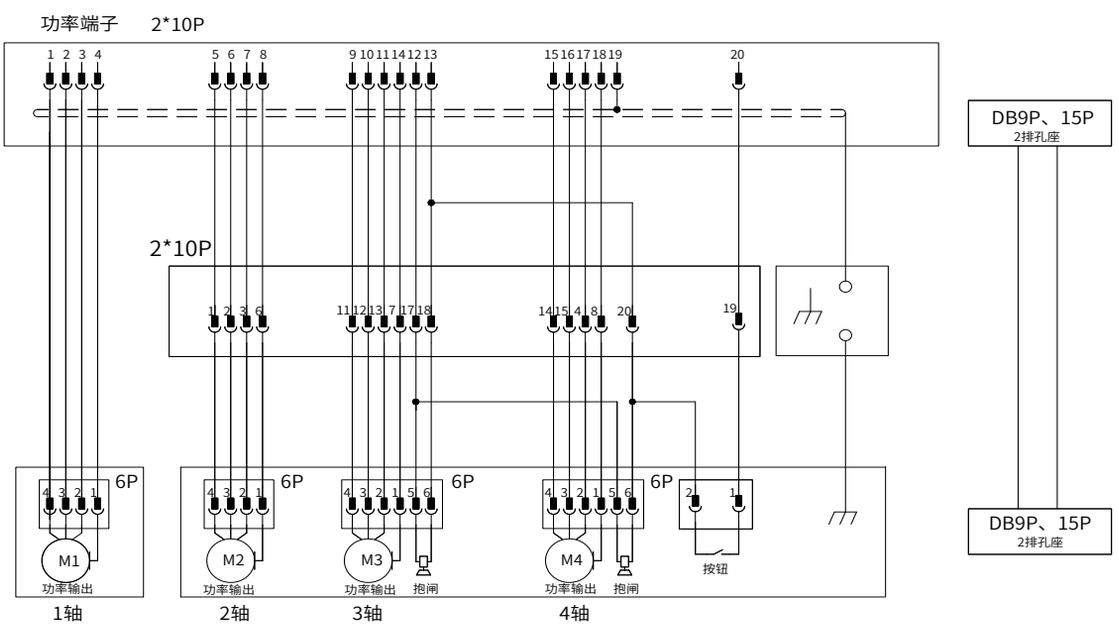
■ IRS100-20 系列

保养和维修时
危 险 ◆ 操作前，请确认紧急停止开关与安全门开关工作正常，否则系统发生紧急状况时将无法发挥安全功能，可能会导致重伤或重大损害。



编码器引脚	7	1	2	5	6	3	4
定义	屏蔽	PS+	PS-	5V	GND	BT+	BT-

图 4-69 机器人本体电气连接图（编码器连接）



功率输出引脚	4	3	2	1	5	6
定义	U	V	W	PE	BK+	BK-

图 4-70 机器人本体电气连接图（动力线连接）

4.10 部件清单

IRB100-3-40Z15TS3			
部件名称		部件信息	参阅：维护篇
本体电缆			*“4.6 机器人线束维护”*
AC 伺服电机	第 1 关节	400W	*“4.1.1 J1 轴电机维护”*
	第 2/4 关节	100W	*“4.2.1 J2 轴电机维护”* *“4.4.2 J4 轴电机维护”*
	第 3 关节	100W	*“4.3.2 J3 轴电机维护”*
减速机	第 1 关节	72100175（汇川物料编码)/ 72100053（汇川物料编码）	
	第 2 关节	72100169（汇川物料编码)/ 72100054（汇川物料编码）	
O 型圈	第 1 关节	53*1.8	*“4.1.1 J1 轴电机维护”*
同步带	第 3 关节	72100048	*“4.3.1 J3 轴同步带维护”*
	第 4 关节	72100050（汇川物料编码）	*“4.4.1 J4 轴同步带维护”*
		72100049（汇川物料编码）	
制动解除开关		J3 轴	
滚珠丝杠花键		150mm	
电池		更换用锂电池（每组 2 个）	*“4.7 电池更换与维护”*
指示灯		24V- 外径 22mm	
润滑脂	滚珠丝杠花键	MUL TEMP LRL No.3	*“4.5 滚珠丝杠花键保养”*
	减速机	SK-1A	
		SK-2A	
塑胶件	小臂外壳		
	花键母外壳		
控制柜到本体电缆		3m	

IRB100-6-*			
部件名称		部件信息	参阅：维护篇
本体电缆			*“4.6 机器人线束维护”*
AC 伺服电机	第 1/2 关节	400W	*“4.1.1 J1 轴电机维护”* *“4.2.1 J2 轴电机维护”*
	第 3 关节	100W	*“4.3.2 J3 轴电机维护”*
	第 4 关节	100W	*“4.4.2 J4 轴电机维护”*
减速机	第 1 关节	72100034（汇川物料编码)/ 72100051（汇川物料编码)	
	第 2 关节	72100035（汇川物料编码)/ 72100052（汇川物料编码)	

IRB100-6-*			
部件名称		部件信息	参阅：维护篇
O 型圈	第 1 关节	53*1.8	*“4.1.1 J1 轴电机维护”*
		90mmm*1.8mm	*“4.3.1 J3 轴同步带维护”*
	第 2 关节	73mm*1.8mm	*“4.4.1 J4 轴同步带维护”*
同步带	第 3 关节	72100068（汇川物料编码 ）	*“4.4.1 J4 轴同步带维护”*
	第 4 关节	72100069（汇川物料编码 ）	
		72100070（汇川物料编码 ）	
制动解除开关		J3 轴	
滚珠丝杠花键		200mm	
电池		更换用锂电池（每组 2 个）	*“4.7 电池更换与维护”*
指示灯		24V- 外径 22mm	
润滑脂	滚珠丝杠花键	MUL TEMP LRL No.3	*“4.5 滚珠丝杠花键保养”*
	减速机	SK-1A	
塑胶件	小臂外壳		
	花键母外壳		
控制柜到本体电缆		3m	

IRB100-20-*			
部件名称		部件信息	参阅：维护篇
本体电缆			*“4.6 机器人线束维护”*
AC 伺服电机	第 1/2 关节	750W	*“4.1.1 J1 轴电机维护”*
	第 3 关节	400W	*“4.3.2 J3 轴电机维护”*
	第 4 关节	200W	*“4.4.2 J4 轴电机维护”*
减速机	第 1 关节	72100058（汇川物料编码)/ 72100062（汇川物料编码 ）	
	第 2 关节	72100059（汇川物料编码)/ 72100063（汇川物料编码 ）	
	第 4 关节	72100060（汇川物料编码)/ 72100061（汇川物料编码 ）	*“4.4.3 J4 轴行星减速机维护”*
O 型圈	第 1/2 关节	73*1.8	*“4.1.1 J1 轴电机维护”* *“4.2.1 J2 轴电机维护”*
		85*1.8	
		90*1.8	
		118*1.8	

IRB100-20-*			
部件名称		部件信息	参阅：维护篇
同步带	第 3 关节	臂长 600/700 机型： 72100176（汇川物料编码 ） 臂长 800/1000 机型： 72100122（汇川物料编码 ）	*“4.3.1 J3 轴同步带维护”*
	第 4 关节	臂长 600/700 机型： 72100177（汇川物料编码 ） 臂长 800/1000 机型： 72100123（汇川物料编码 ）	*“4.4.1 J4 轴同步带维护”*
制动解除开关		J3/J4 轴	
滚珠丝杠花键		RBS2525	
电池		更换用锂电池（每组 2 个）	*“4.7 电池更换与维护”*
指示灯		24V- 外径 22mm	
润滑脂	滚珠丝杠花键	MUL TEMP LRL No.3	*“4.5 滚珠丝杠花键保养”*
	减速机	SK-1A	
塑胶件	小臂外壳		
	花键母外壳		
	大臂前后端盖		
	小臂上下端盖		
控制柜到本体电缆		3m/5m/10m	

附录篇

附录篇 - 目录

附录 1：机器人伺服调试

1.1 调试流程

1.1.1 基本流程

第 1 步： 确定伺服驱动器电源线，UVW 线，编码器，CN1 端子上信号线，EtherCAT 网线等是否连接正确。

第 2 步： 确认伺服参数是否设置正确。

第 3 步： 伺服 JOG 点动试运行

第 4 步： 确认旋转方向是否正确。

第 5 步： 进行各个轴的负载惯量辨识、

第 6 步： 通过刚性表粗调各轴增益。

第 7 步： 机器人控制器设置结构参数，轴限位，零点，运动参数等，并示教运行。

第 8 步： 机器人再现运行，由低速逐渐到高速来回往复运行。

第 9 步： 根据定位要求，细调伺服增益。

第 10 步： 根据节拍要求，调节机器人速度或加速度等运动参数。

1.1.2 机器人常用伺服参数设置

第 1 步： 根据实际使用的电机类型，设置伺服驱动器上的电机型号参数。当电机为标准绝对式电机，即铭牌上的 Motor Code：14101 时，请将伺服驱动器的参数 H0000 设置为 14101；若实际电机为机器人专用超短电机，即电机铭牌上的 Motor Code：14120 时，请请将伺服驱动器的参数 H0000 设置为 14120。

第 2 步： 设置编码器类型，由于机器人上的电机都采用绝对式电机，所有需要设置电机编码器类型为绝对位置线性模式，即 H0201=1。

第 3 步： 根据电机型号参数，设置伺服驱动器上的电子齿轮比参数。机器人控制器位置当量为 2^{20}，当 H0000=14101 时，此时电机编码器当量为 223，因此需要设置电子齿轮比 6091h-1h 设置为 8，若采用的是 620N 伺服，也可以等效设置 H1C36=8。 当 H0000=14120 时，此时电机编码器当量为 220，此时电子齿轮比 6091h-1h 设置为 8，若采用的是 620N 伺服，也可以等效设置 H1C36=8。

第 4 步： 设置伺服驱动器的控制模式，由于机器人控制器和对应伺服驱动器采用的是 EtherCAT 通讯方式，所以控制模式应该选择 EtherCAT 模式，即 H0200=9。

第 5 步： 根据机器人各个轴规定的正方向，设置电机的旋转方向，即设置 H0202=1 或 0 ，从而保证电机旋转方向和机器人规定正方向一致。

第 6 步： 根据是否接有外置泄放电阻设置相应的伺服参数，若电控柜内部伺服驱动器上接有外置泄放电阻，根据电阻规格及散热方式，设置 H0225，H0226 和 H0227。注意，外置泄放电阻的实际阻值不能小于 50 欧，H0227 不能小于 50，否则伺服驱动器内部的泄放管存在烧毁的风险。

第 6 步： 根据电机是否带有抱闸，设置伺服驱动器上相应的抱闸参数，例如若伺服为 IS620N 系列，当 CN1 端子上 DO2 接到抱闸继电器上时，则需要设置 H0402=9，H0403=1。若伺服为 SV820N 系列，则只需要设置 H0216=1。

第 7 步： 根据各个轴的负载惯量结果，设置伺服驱动器的 H0815 参数。

第 8 步： 设置伺服驱动器增益参数，根据实际伺服跟随响应的需求，设置 H0800~H0802 以及 H0705 参数。另外 IS620N 和 SV820N 伺服都需要设置 H0806=2。

第 9 步： 倘若机器人对运动轨迹有要求，根据需要设置伺服驱动器的速度前馈 H0818 和 H0819 相关参数。

1.1.3 伺服 JOG 点动试运行

请使用 JOG 点动运行确认伺服电机是否可以正常旋转，转动时无异常振动和异常声响。可以通过伺服调试软件两种方式操作。电机以当前功能码 H0604 存储值作为点动速度。

■ 伺服驱动调试平台点动运行

打开我司伺服驱动调试平台点动运行界面，设置 H0604 点动速度值，点击界面伺服 ON 按钮后，通过界面上正反转按钮实现点动正反转运行功能。当关闭点动运行界面，退出点动运行模式时，之前设置的 H0604 点动运行速度值并不保存，重新还原成默认值。

1.1.4 负载惯量辨识操作

伺服驱动器支持离线和在线两种惯量辨识方法，其操作分别如下：

1 离线惯量辨识

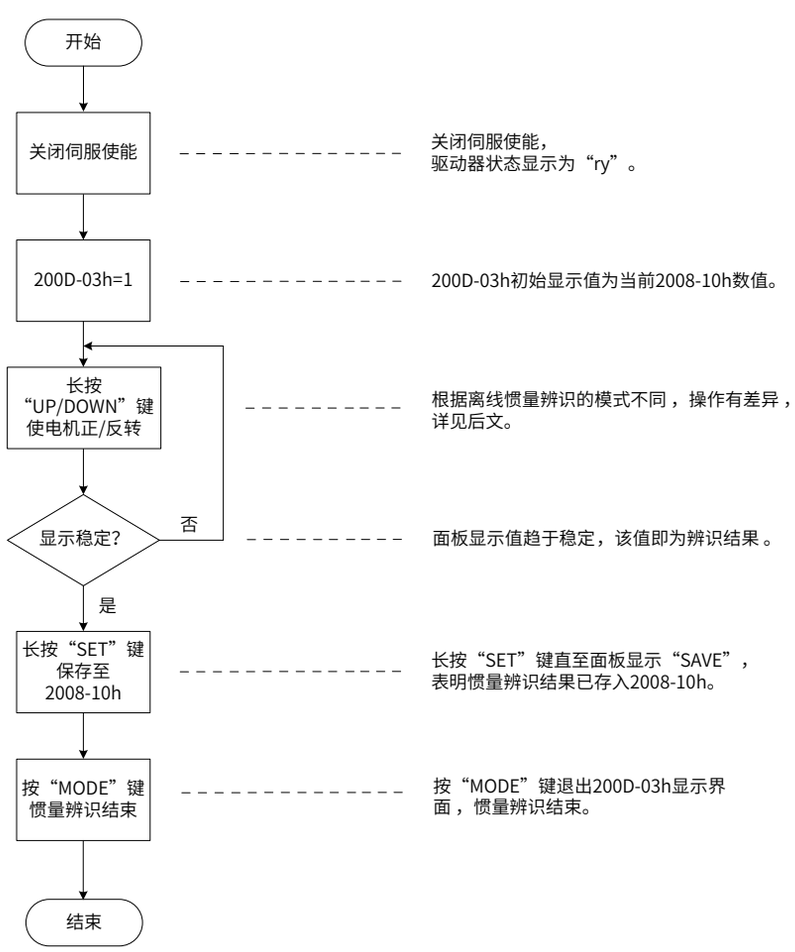
进行离线惯量辨识前，首先确认如下内容：

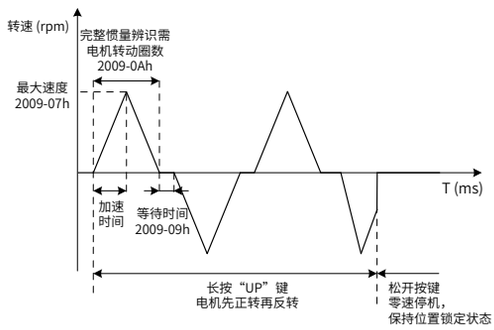
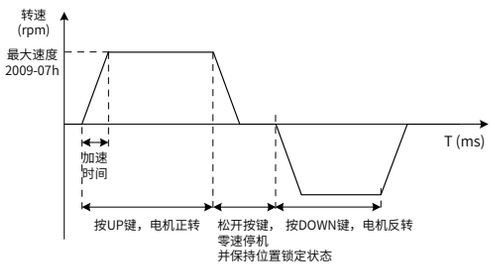
1） 在机械限位范围内有正反各 1 圈以上的可运动行程。

2） 满足 H0909(完成单次惯量辨识所需电机转动圈数) 要求。查看当前惯量辨识最大速度 H0906，惯量辨识时加速至最大速度时间 H0907，以及完成惯量辨识所需电机转动圈数 H0909，确保电机在此停止位置处的运行行程大于 H0909 设置值，否则应适当减小 H0906 或 H0907 设置值，直至满足该要求。

3） 预估负载惯量比 H0815 数值， H0815 默认值为 1，如果实际负载惯量比大于 30.00，可能会发生电机动作迟缓而导致辨识失败，此时可采取预置 H0815 为一较大的初始值，建议以 5.00 为起始值。或者适当增加伺服驱动器增益，以使电机实际转速能够达到惯量辨识最大速度 H0906.

离线惯量辨识的一般操作流程：

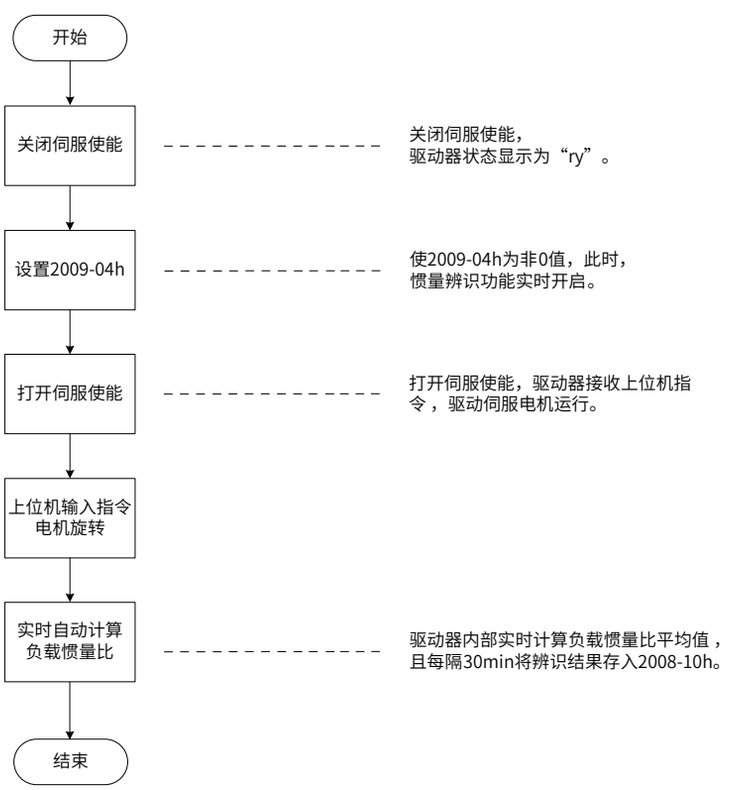


项目	正反三角波形式 (2009-06h=0)	JOG 点动模式 (2009-06h=1)
指令形式	对称三角波  	梯形波 
最大速度	2009-07h	2009-07h
加减速时间	2009-08h	2009-08h
按键说明	长按 UP 键：电机先正转后反转 长按 DOWN 键：电机先反转后正转 松开按键：零速停机，保持位置锁定状态	按 UP 键：电机正转 按 DOWN 键：电机反转 松开按键：零速停机，保持位置锁定状态
间隔时间	2009-09h	前后两次按键操作时间间隔
电机旋转圈数	≤ 2009-0Ah	人为控制
适用场合	电机行程较短的场合	电机行程较长，可人为控制的场合

也可以通过伺服后台软件进行离线惯量辨识，即点击进行“辅助功能 -> 惯量辨识 (New)”界面，然后按照提示逐步操作，当辨识完成后，再将辨识结果手动输入到 H0815 参数中。

2 在线惯量辨识

在线惯量辨识一般操作流程如下：



H0903 设置 1~3 的区别在于负载惯量比 H0815 的实时更新速度不同：

-601-

H0903=1，适用于实际负载惯量比几乎不会发生变化的场合，如机床，雕铣机等。

H0903=2，适用于实际负载惯量比发生缓慢变化的场合。

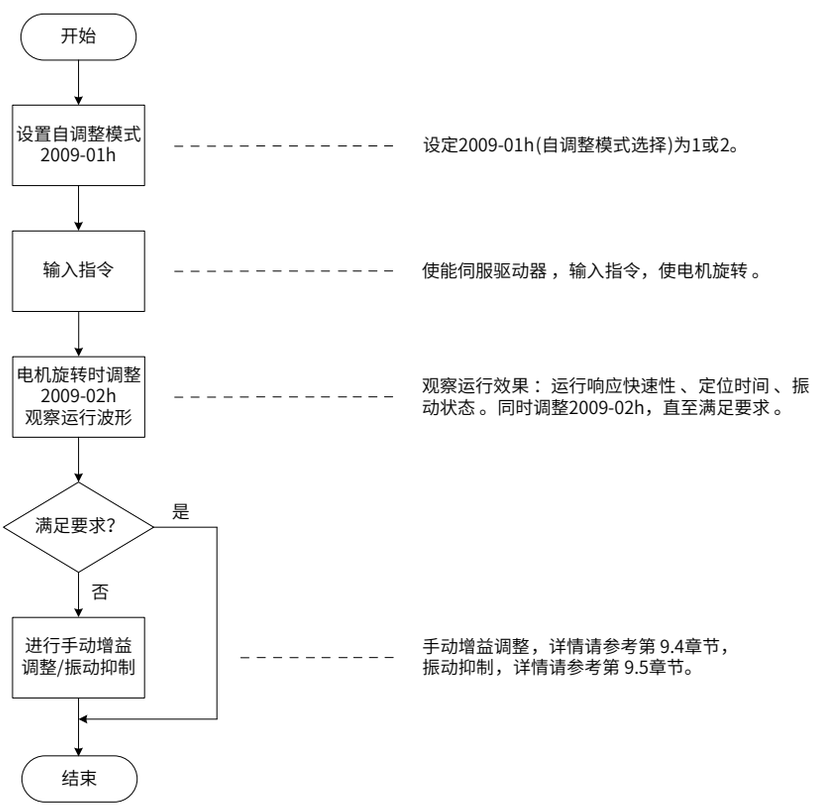
H0903=3，适用于实际负载惯量比会发生快速变化的场合，如搬运机械手等。

通过伺服驱动器面板或者后台软件，将伺服驱动器功能码 H09_03 设置成 3，即选择“2- 开启在线辨识，快速变化”。在机器人示教器上编写一个小程序，仅仅只让相应轴正负旋转，旋转的角度范围以不要和周边设备有干涉碰撞为前提，然后在“在线模式下”循环往复运行。旋转角度越大越好，运行速度越大越好，加速度越大越好，其在线惯量辨识出来的结果越准确。在“在线模式”下，正负往复运行过程中，多次反复读取伺服功能码 H0815，观察伺服功能码 H08_15 数值是否有变化，以及 H0815 数值变化趋于稳定 (一般仅仅是小数点后面数字变化)。此时该轴的惯量值已经辨识出来，并自动保存到了 H0815 功能码中。然后再将该轴伺服驱动器的功能码 H09_03 设置 0，即“0_ 关闭在线惯量辨识”。最后在读取查看该 轴伺服驱动器的功能码 H08_15 的数值，确认之前“在线辨识出来的结果”是否正确保存到 H08_15 功能码中。

1.1.5 伺服增益调试

1 自动增益调整

自动增益调整是指通过刚性等级选择功能 H0901，伺服驱动器将自动产生一组匹配的增益参数，满足快速性欲稳定性需求。在使用自动增益调整功能前，务必正确获得负载惯量比。



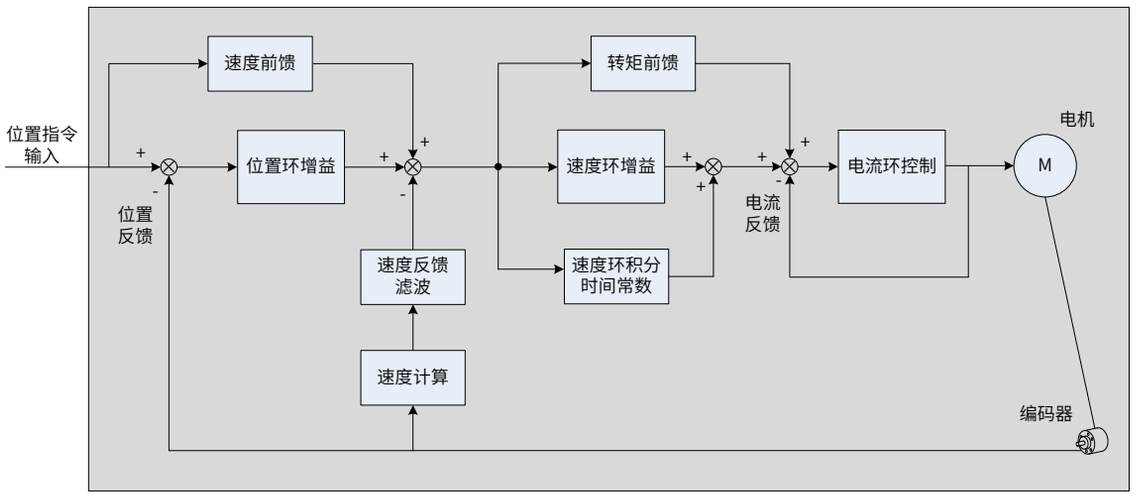
刚性等级 H0901 的取值范围在 0~31 级之间。0 级对应的刚性最弱，增益最小；31 级对应的刚性最强，增益最大。根据不同的负载类型，以下经验值可供参考。

推荐刚性等级	负载机构类型
4 级～ 8 级	一些大型机械
8 级～ 15 级	皮带等刚性较低的应用
15 级～ 20 级	滚珠丝杠、直连等刚性较高的应用

-602-

2 手动增益调整

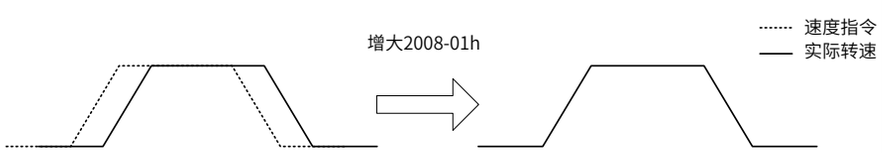
在自动增益调整达不到预期效果时，可以手动微调增益。通过更细致的调整，优化效果。伺服系统由三个控制环路构成，从外向内依次是位置环，速度环和电流环，基本控制框图如下图所示。

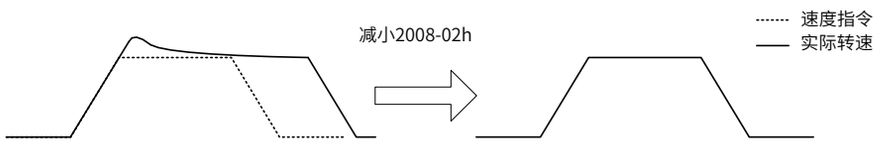
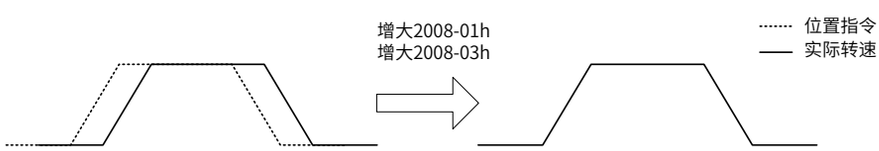
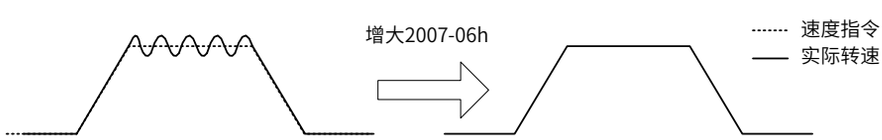


越是内侧的环路，要求响应性越高。不遵守该原则，可能导致系统不稳定。

伺服驱动器默认的电流环增益已确保了充分的响应性，一般无需调整，需要调整的只有位置环增益、速度环增益及其它辅助增益。因此，位置模式模式下进行增益调整时，为保证系统稳定，提高位置环增益的同时，需提高速度环增益，并确保位置环的响应低于速度环的响应。

基本增益调整方法如下：

步骤	索引码	名称	调整说明
1	2008-01h	速度环增益	参数作用： 决定速度环能够跟随的变化的速度指令最高频率。 在负载惯量比平均值 (2008-10h) 设置正确的前提下，可认为： 速度环最高跟随频率 =2008-01h  调整方法： 在不产生噪声、振动的情况下，增大此参数，可加快定位时间，带来更好的速度稳定性和跟随性； 产生噪音，则降低参数设定值；

步骤	索引码	名称	调整说明
2	2008-02h	速度环积分时间常数	参数作用： 消除速度环偏差。  调整方法： 建议按以下关系取值： $500 \leqslant 2008\text{-}01h \times 2008\text{-}02h \leqslant 1000$ 例如，速度环增益 2008-01h=40.0Hz 时，速度环积分时间常数应满足：12.50ms ⩽ 2008-02h ⩽ 25.00ms。 减小设定值可加强积分作用，加快定位时间，但设定值过小易引起机械振动。 设定值过高，将导致速度环偏差总不能归零。 当 2008-02h=512.00ms 时，积分无效。
3	2008-03h	位置环增益	参数作用： 决定位置环能够跟随的变化的位置指令最高频率。 位置环最高跟随角频率 =2008-03h  调整方法： 为保证系统稳定，应保证速度环最高跟随频率是位置环最高跟随频率的 3~5 倍，因此： $3 \leqslant \frac{2\times\pi\times 2008\text{-}01h}{2008\text{-}03h} \leqslant 5$ 例如，速度环增益 2008-01h=40.0Hz 时，位置环增益应满足：50.2Hz ⩽ 2008-03h ⩽ 83.7Hz。 根据定位时间进行调整。加大此参数，可加快定位时间，并提高电机静止时抵抗外界扰动的能力。 设定值过高可能导致系统不稳定，发生振荡。
4	2007-06h	转矩指令滤波时间常数	参数作用： 消除高频噪声，抑制机械共振。  调整方法： 应保证转矩指令低通滤波器的截止频率高于速度环最高跟随频率的 4 倍，因此： $\frac{1000}{2\times\pi\times 2007\text{-}06h} \geqslant (2008\text{-}01h)\times 4$ 例如，速度环增益 2008-01h=40.0Hz 时，转矩指令滤波时间常数应满足：2007-06h ⩽ 1.00ms。 增大 2008-01h 发生振动时，可通过调整 2007-06h 抑制振动； 设定值过大，将导致电流环的响应降低； 需抑制停机时的振动，可尝试加大 2008-01h，减小 2007-06h； 电机停止状态振动过大，可尝试减小 2007-06h 设定值。

1.2 伺服调试软件操作指导

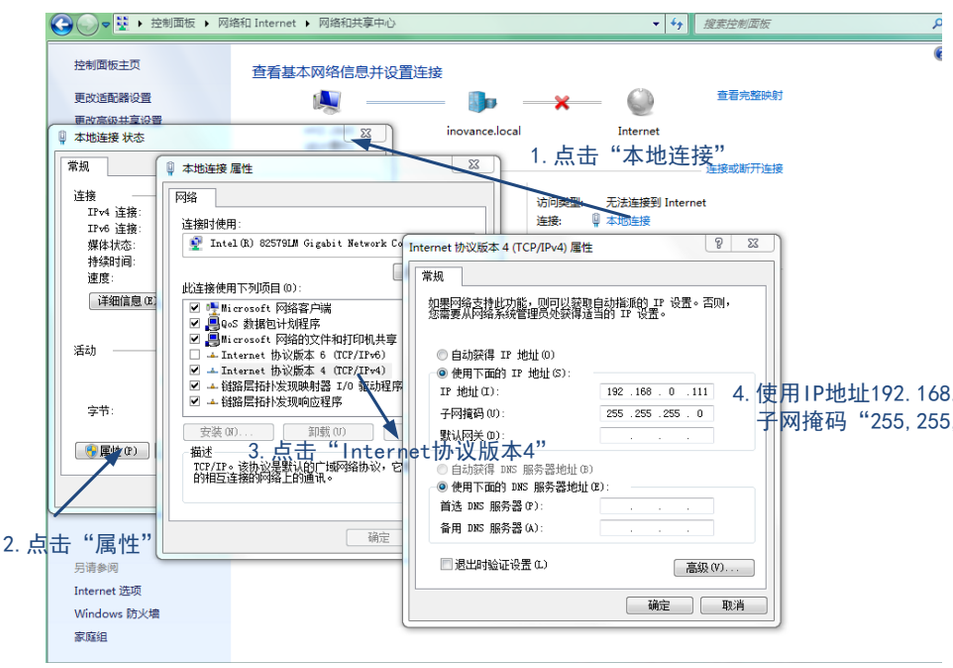
IRCB300 系列机器人控制柜提供 SERVO-NET 端口，可接入伺服后台调试软件进行参数设置、在线调试诊断等。

1.2.1 伺服调试软件连接

步骤 1：通过网线，将 IRCB300 系列机器人控制柜伺服调试端口“SERVO-NET”连接到电脑的网口上。

注：对于 IRCB300 系列 6 轴机器人控制柜，调试 J1~J4 轴时，请将网线连接到电控柜的 SERVO-NET1 端口，调试 J5~J6 轴时，请将网线连接到电控柜的 SERVO-NET2 端口。

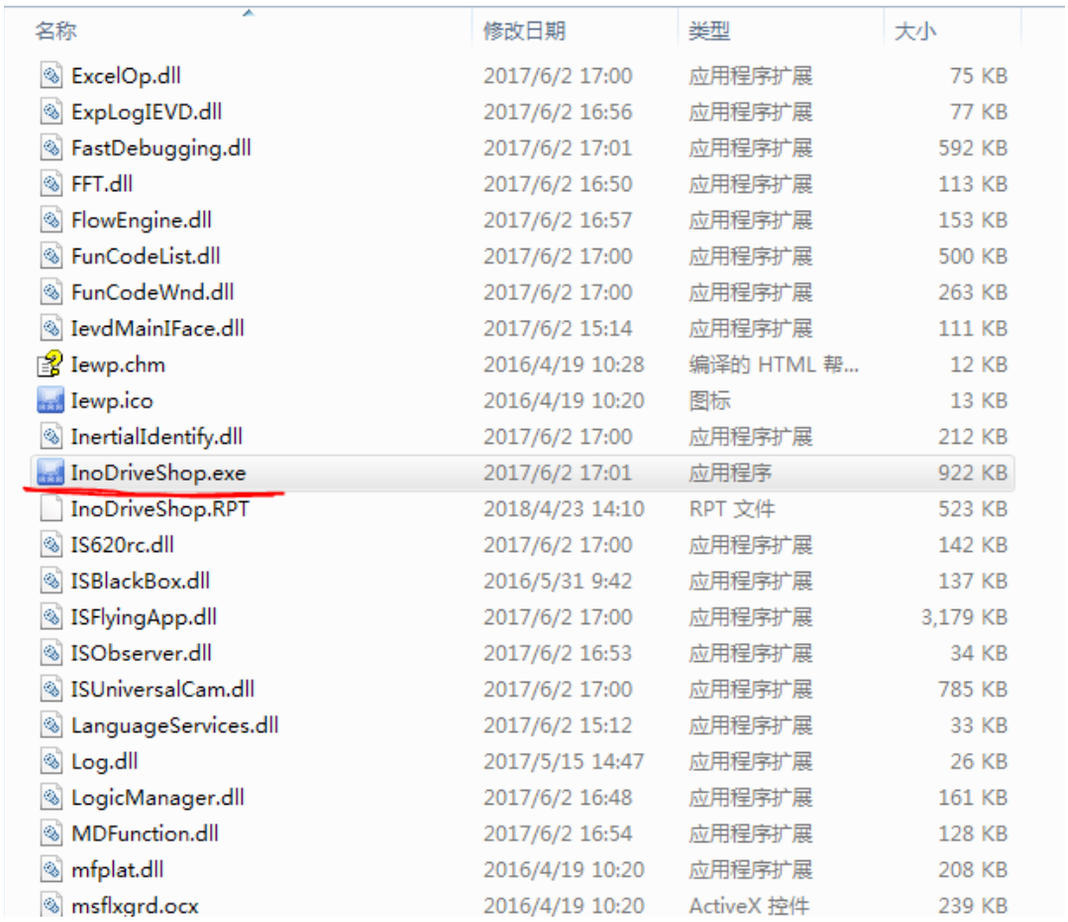
步骤 2：在网络和共享中心中，选取本地连接，配置计算机本地 IP 为 192.168.0.XXX；



NOTE

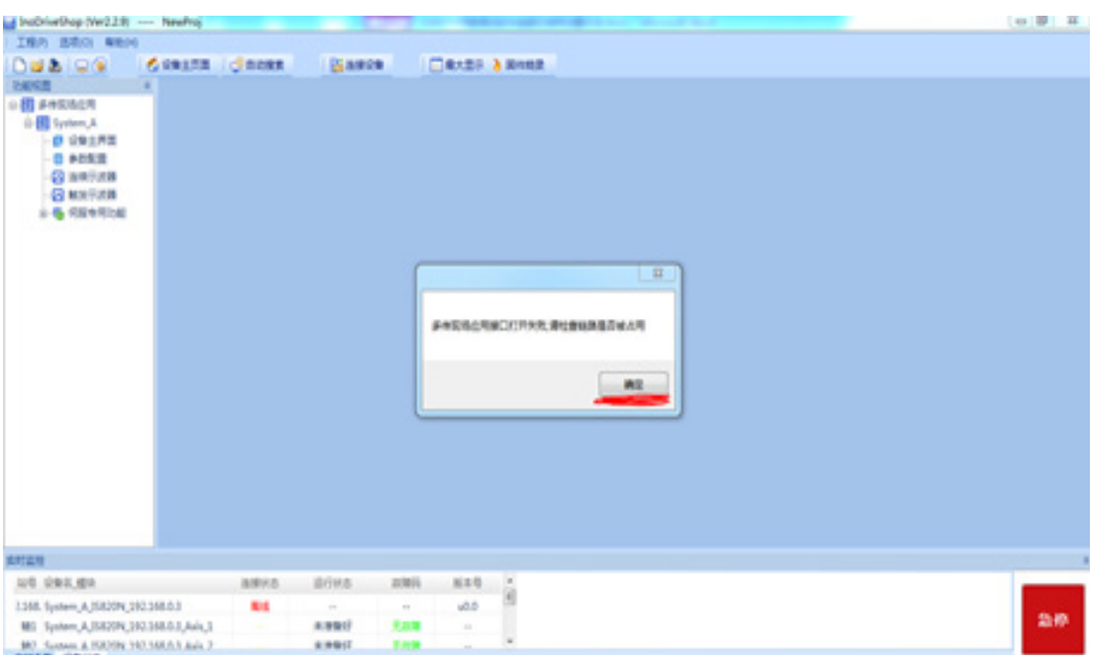
◆ 请保证设备端 IP 地址和电脑本机连接 IP 地址处在同一网段内并且不重合，才可正常扫描连接。伺服设备端默认 IP 为：192.168.0.2；电脑端 IP 可以在控制面板 --> 网络和共享中心中查询和修改。

步骤 3：打开伺服调试软件压缩包，双击“InoDriveShop.exe”，打开伺服调试软件；

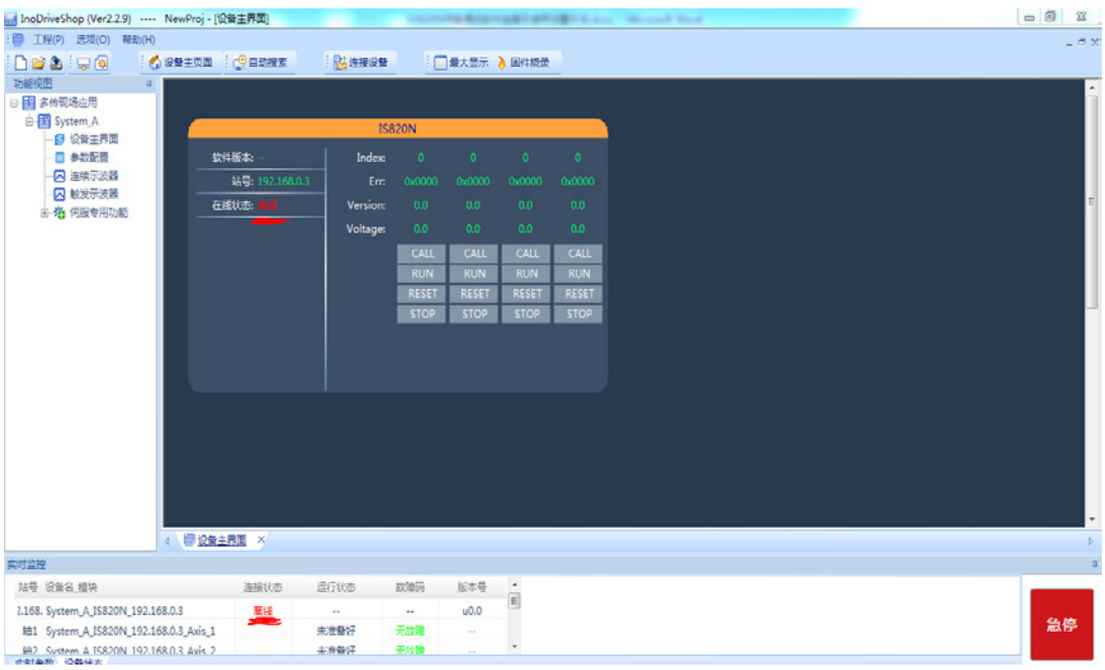


名称	修改日期	类型	大小
ExcelOp.dll	2017/6/2 17:00	应用程序扩展	75 KB
ExpLogIEVD.dll	2017/6/2 16:56	应用程序扩展	77 KB
FastDebugging.dll	2017/6/2 17:01	应用程序扩展	592 KB
FFT.dll	2017/6/2 16:50	应用程序扩展	113 KB
FlowEngine.dll	2017/6/2 16:57	应用程序扩展	153 KB
FunCodeList.dll	2017/6/2 17:00	应用程序扩展	500 KB
FunCodeWnd.dll	2017/6/2 17:00	应用程序扩展	263 KB
IevdMainIFace.dll	2017/6/2 15:14	应用程序扩展	111 KB
Iewp.chm	2016/4/19 10:28	编译的 HTML 帮...	12 KB
Iewp.ico	2016/4/19 10:20	图标	13 KB
InertialIdentify.dll	2017/6/2 17:00	应用程序扩展	212 KB
InoDriveShop.exe	2017/6/2 17:01	应用程序	922 KB
InoDriveShop.RPT	2018/4/23 14:10	RPT 文件	523 KB
IS620rc.dll	2017/6/2 17:00	应用程序扩展	142 KB
ISBlackBox.dll	2016/5/31 9:42	应用程序扩展	137 KB
ISFlyingApp.dll	2017/6/2 17:00	应用程序扩展	3,179 KB
ISObserver.dll	2017/6/2 16:53	应用程序扩展	34 KB
ISUniversalCam.dll	2017/6/2 17:00	应用程序扩展	785 KB
LanguageServices.dll	2017/6/2 15:12	应用程序扩展	33 KB
Log.dll	2017/5/15 14:47	应用程序扩展	26 KB
LogicManager.dll	2017/6/2 16:48	应用程序扩展	161 KB
MDFunction.dll	2017/6/2 16:54	应用程序扩展	128 KB
mfplat.dll	2016/4/19 10:20	应用程序扩展	208 KB
msflxgrd.ocx	2016/4/19 10:20	ActiveX 控件	239 KB

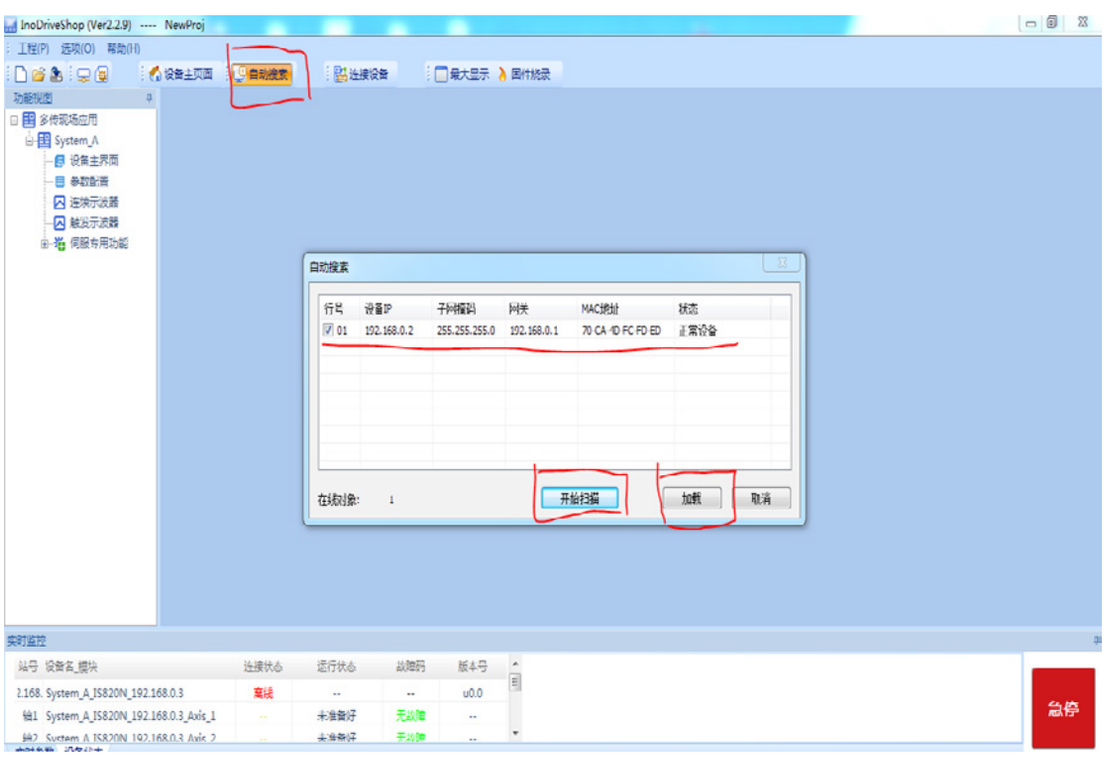
步骤 4：打开伺服调试软件后，会弹出“多传现场应用接口打开失败，请检查链路是否被占用”的提示，此时可忽略该提示，点击“确认”按钮。



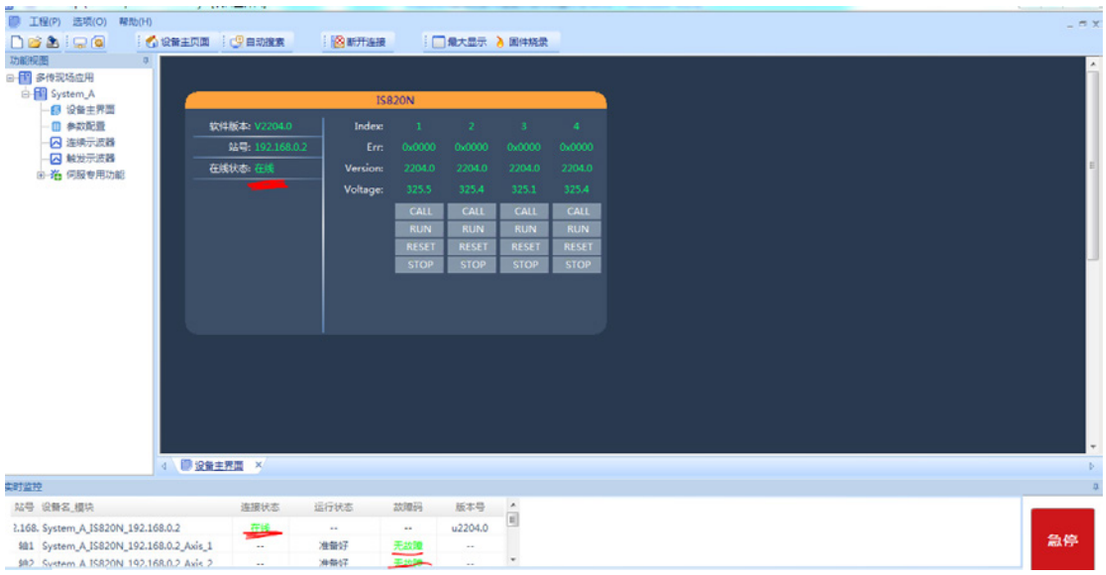
步骤 5：打开伺服调试软件后，如下图红线标识所示，此处提示“离线”，表示该软件没有和伺服驱动器连接上。



步骤 6：如下图所示，点击“自动搜索”按钮，弹出如下图所示的对话框，然后再点击对话框中的“开始扫描”按钮，会自动搜索出相关的伺服设备显示在对话框的表格中，最后再点击对话框中的“加载”按钮。进行调试软件和伺服设备连接。

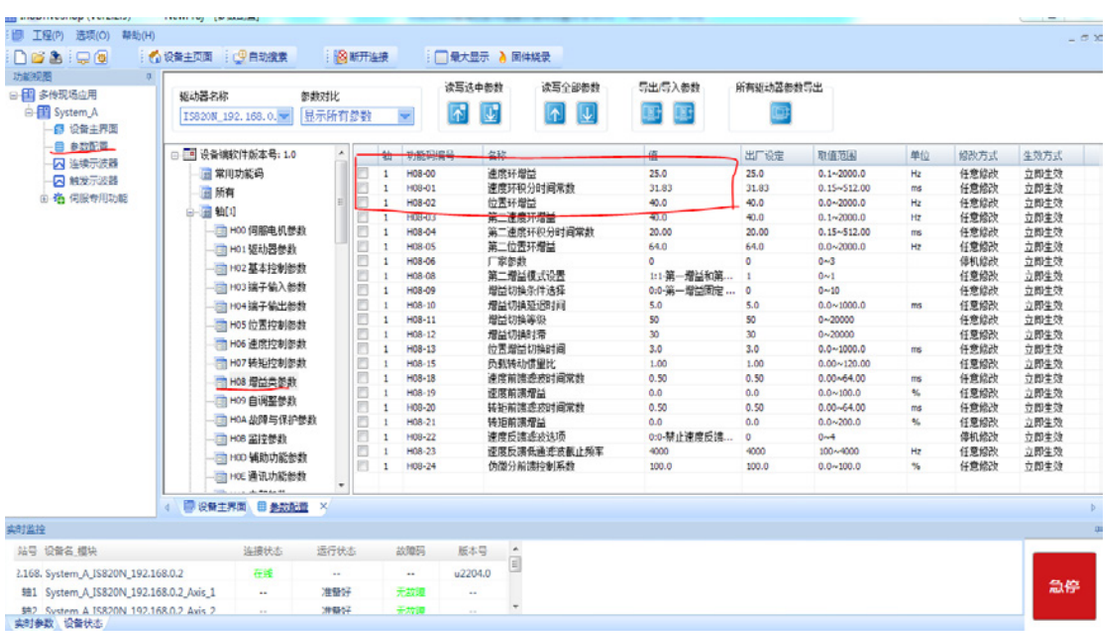


步骤 7：正常连接后调试软件，如下截图所示，红色标识位置会显示“在线”。当伺服驱动器无故障时，其软件下方“实时监控”栏里的故障码为“无故障”。至此，伺服调试软件和伺服驱动器正常连接，可以进行相关参数的设置和波形采集。

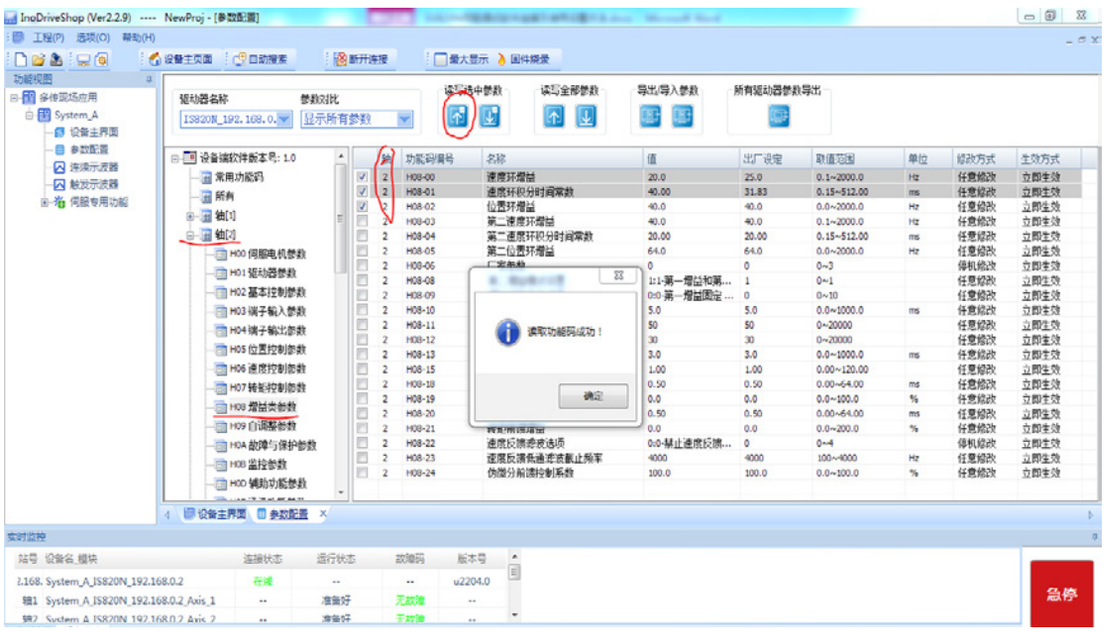


1.2.2 伺服调试软件的参数设置操作

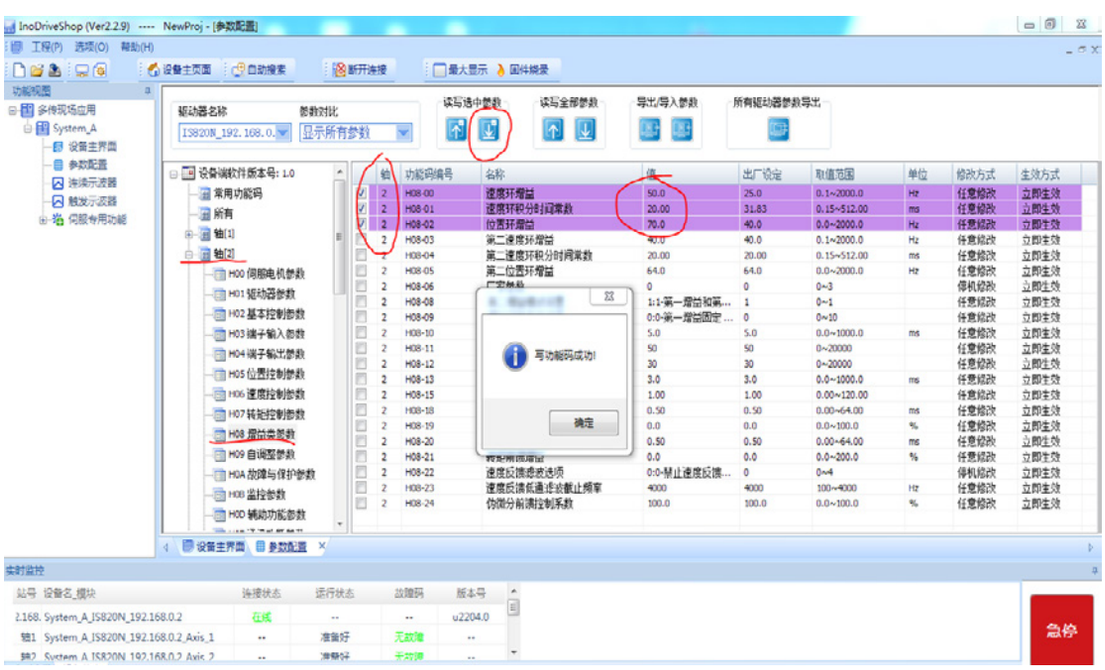
步骤 1：参数设置，点击左侧“System_A 项目下的参数配置”，其右侧可以看到相关伺服参数功能码，同时可以看到 4 个轴，即：轴 1~ 轴 4。每个轴的伺服参数又被分别 H00~H18，共计 15 个组，并进行了组的分类。其中 H08 组为调试中用的最多的 PID 增益内参数。每个组的具体功能码参数设置界面为最右侧的大窗口。其中第一列为轴号，第 2 列为功能码编号，第 3 列为名称，第 4 列则为需要设置数值。



步骤 2：进行功能码参数读取，以轴 2，H08 组增益类参数，H0800~H0802 为例。先将 H0800~H0802 三个参数左侧的□，勾选上；然后点击“读写选中参数中的向上箭头的按钮”进行伺服驱动器参数上传读取，也可以采用另一种操作方式，即点击鼠标右键，在弹出的对话框中，单击“读取当前选中参数”，进行伺服参数读取，读取成功后，软件会自动弹出“读取功能码成功”的提示框。

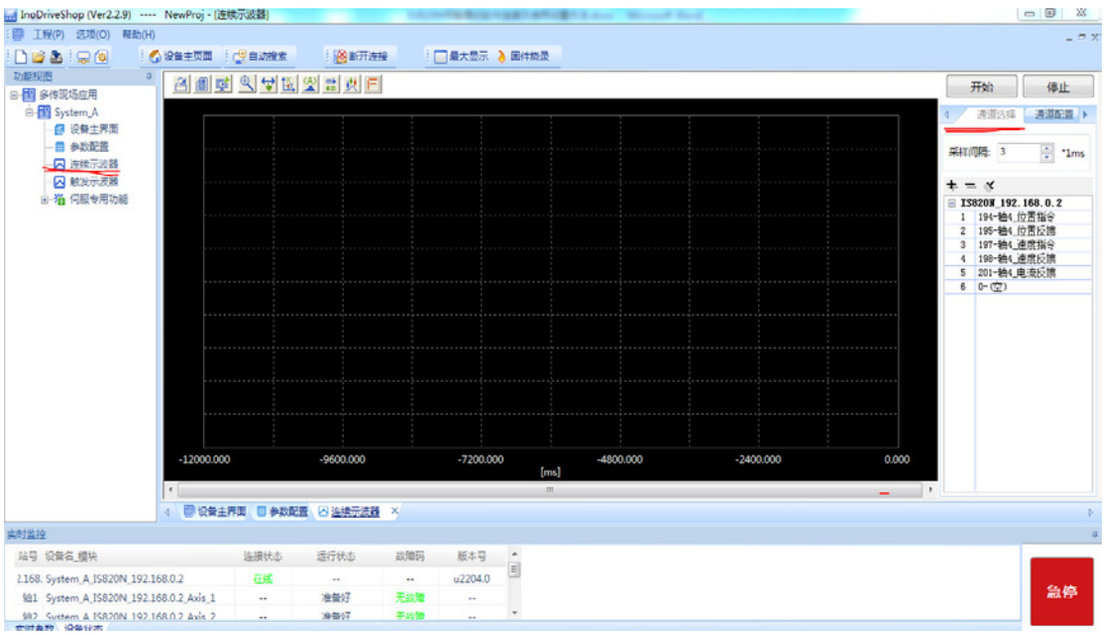


步骤 3：进行功能码参数写入，以轴 2，H08 组增益类参数，H0800~H0802 为例。分别在 H0800~H0802 三个参数的“值”列中，输入自己想要设置的具体参数数值，然后按下键盘上的“回车”键确认，此时新输入参数值被预录入，且和之前参数值进行比较，当不相同时，其背景色自动由灰色变为高亮紫色。同时左侧的□，自动勾选上；然后只需点击“读写选中参数中的向下箭头的按钮”进行伺服驱动器参数下载写入，也可以采用另一种操作方式，即点击鼠标右键，在弹出的对话框中，单击“写入当前选中参数”，进行伺服参数下载写入，写入成功后，软件会自动弹出“写入功能码成功”的提示框。

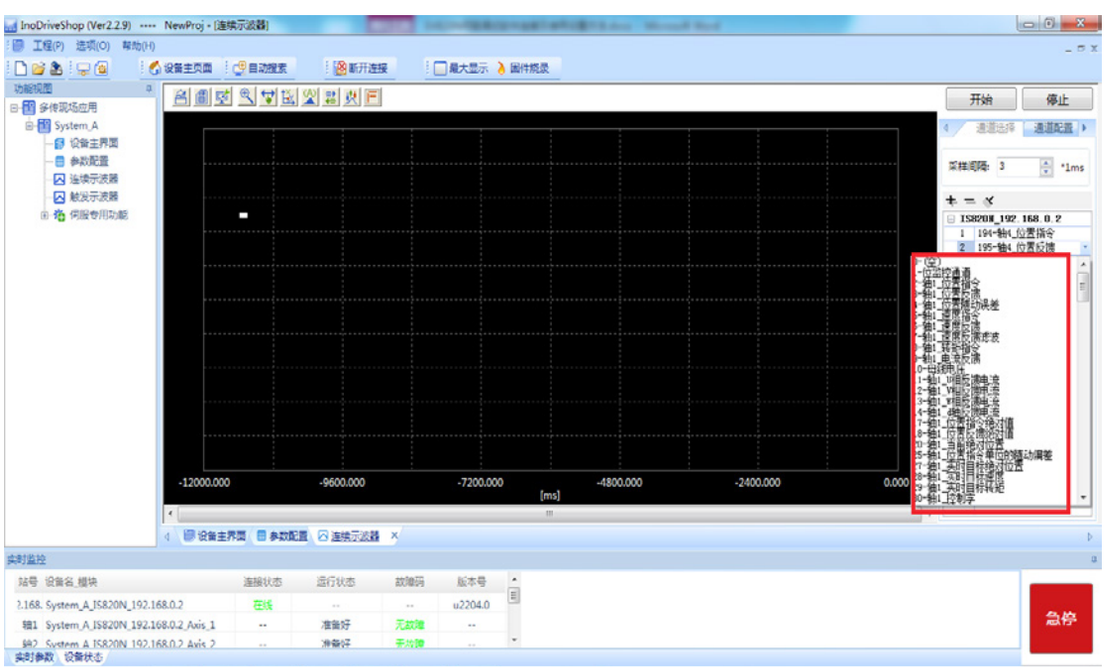


1.2.3 伺服调试软件的波形采集

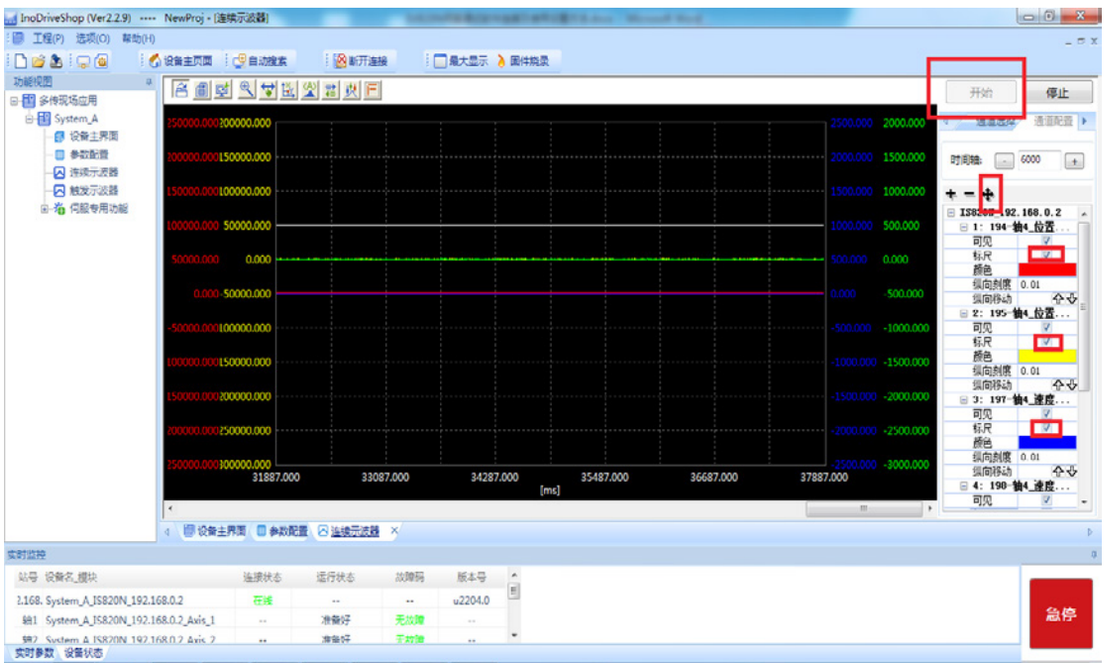
步骤 1：如下截图所示，波形采集共有 2 种方式，分别为：“连续示波器”和“触发示波器”，其中“连续示波器”类似物理示波器的自动滚动连续采集，而“触发示波器”则为根据设置的触发条件是否满足，来触发单次采集。



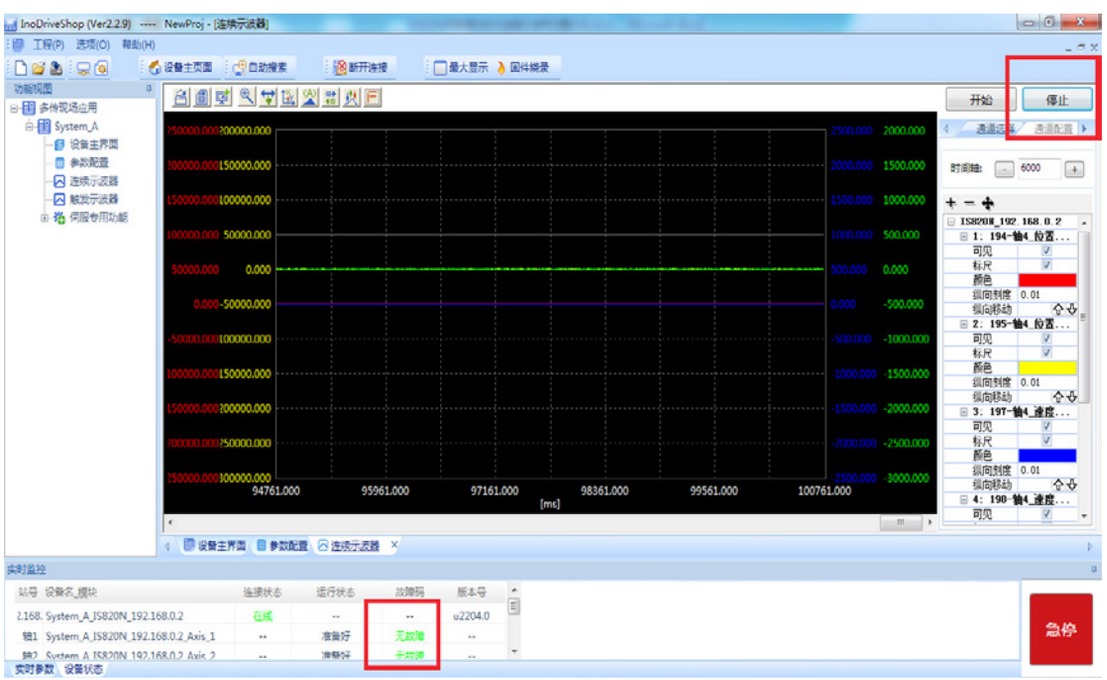
步骤 2：以“连续示波器”为例，如下图所示，该波形采集，共有 6 个通道，即同时可以采集 6 个不同的对象，这个 6 个不同的对象，可以是同一个轴，也可以是 4 个轴的任何一个轴。其 6 个采集对象的选择方法为：单击右侧 6 个采集通道中的表格，则会自动弹出下拉菜单，在相应的下拉菜单中，可以选择所需要的“不同轴的不同对象”。也可以根据需要设置“采样间隔时间”。



步骤 3：以“连续示波器”为例，如下图所示，在设置完采样通道的采集对象和间隔时间后，点击“开始”按键，就可以自动滚动采集了，此时右侧的 6 个采集对象里分别可以设置“是否可见”；“是否显示标尺”；以及对应的“曲线颜色”和“纵轴刻度”，□里打勾，代表启用。也可点击“四个箭头”图标，进行波形曲线的屏幕自动调整。



步骤 4：以“连续示波器”为例，如下图所示，当需要停止采集时，只需要点击“停止”按键就可以了，当需要保存采集的波形时，只需点击上方的第 2 个保存图标，就可以进行波形保存了，另外当伺服驱动器有报错时，在下方的实时监控中的故障码，就可以读取相应的各个轴具体故障码数值。



1.3 常见伺服故障处理方法

Er.136 : 电机编码器 ROM 中的存储参数数据异常

产生机理：

伺服驱动器在上电时刻读取编码器 ROM 区存储的参数时，出现异常，即：未正确读取出相关参数。

原因	确认方法	处理措施
1. 伺服驱动器和电机类型不匹配	确认电机铭牌上的 Motor Code 数值是否和驱动器 H0000 参数一致。	如果不一致，请以电机铭牌上的 Motor Code 数值为准，修改伺服驱动器上的 H0000 数值，使之与电机保持一致。 （注： 电机的 Motor Code 有三种规格：14000，14101， 14120， 务必保证伺服驱动器 H0000 数值和电机 Motor Code 一致）
2. 伺服驱动器编码器接线异常	采用万用表测量伺服驱动器 DB 端子上的编码器各线芯和电机侧的 9 孔插头里的编码器各线芯是否接通良好，是否断线。 检查编码器上各个接插件或接头处各线芯和插针是否接触不良，退针，脱线。	如果测出编码器线芯接触不良，断线，或线芯插针处接触不良，退针，脱线等情况，请更换编码器线缆或接插件，务必保证编码器线缆可靠连接。
3. 伺服驱动器侧和电机侧，编码器接线端子内的线序错误。	检查伺服驱动器侧的编码器 DB 端子内的各芯的线序和管脚是否正确。 检查电机侧的编码器接插件内的各芯的线序和管脚是否正确。	伺服驱动器侧或电机侧编码器端子内的各芯的线序或管脚错误，请参考伺服手册上编码器信号定义，进行改接。
4. 编码器 ROM 内未存储参数或存储参数异常	伺服驱动器 H0A35 设置 0，H0241 设置 1430 后，，读取 H00 组电机参数，检查该组参数内有没有和对应电机不匹配的参数。	若读取 H00 组电机参数异常，请更换电机。
5. 标准 IS620P 伺服驱动器不支持 Motor Code 14120 电机	请确认标准的 IS620P 伺服驱动器其所匹配电机 Motor Code 只能为 14000 或 14101，不能为 14120。	如果需要标准的 IS620P 伺服驱动器匹配 Motor Code 为 14120 的电机话，请联系伺服部门市场联系人，索要相应的非标软件。
6. 伺服驱动器或电机故障	排除上述 5 中情况后，如果伺服驱动器重启后仍持续报 Er.136 故障	采用交叉验证法，定位问题。 1. 用一个已确认好的伺服驱动器去带该报错电机，重新上电后，如果故障消除，则表明该电机故障，请更换电机。 2. 用一个已确认好的电机去搭载该报错驱动器，重新上电后，如果故障消除，则表明该伺服驱动器故障，请更换伺服驱动器。

Er.400: 主回路电过压

产生机理：

P、 – 之间直流母线电压超过故障阈值

220V 驱动器：母线电压正常值：310V，母线电压故障值：420V

380V 驱动器：母线电压正常值：540V，母线电压故障值：760V

原因	确认方法	处理措施
1. 电机运行于急加减状态，且速度，加速度过大，导致制动能量回馈到伺服驱动器母线上超过了母线电压故障阈值。	通过伺服后台软件的示教器界面，监控伺服母线电压波形，观察发生故障时刻的母线电压实际值是否超过故障阈值。	请在伺服驱动器的 P 和 C 端子之间，接入合适的外置泄放电阻，并根据外置泄放电阻的阻值，功率，散热方式，设置相应伺服驱动器参数 H0225、H0226、H0227。外置泄放电阻的选型及参数设置方法，请参考伺服手册。
2. 电机运行于急加减状态，且速度，加速度过大，伺服 P、C 端已接入外置泄动电阻，但伺服驱动器未正确设置。	请检查伺服驱动器关于外置泄放电阻的相关参数 H0225、 H0226、H0227，是否正确设置，是否启用外置泄放功能。	根据外置泄放电阻的规格，正确设置伺服驱动器关于外置泄放电阻的相关参数 H0225、H0226、H0227
3. 外置泄放电阻失效	请在伺服驱动器断电情况下，用万用表测量伺服驱动器 P、C 之间的外接制动电阻的阻值是否为电阻规格值，若阻值过大，则电阻烧坏。	请更换功率更大的外置泄放电阻。
4. 电机运行于急加减状态，且速度，加速度过大，负载过大，导致制动能量超过外置泄放电阻最大泄放能力。	当外置泄放电阻的阻值为 50 欧 (不能低于 50 欧) 时，达到外置泄放电阻最大泄放能力，此种情况急加减速时，仍报 Er400 过压故障	请降低各电机轴加速度或速度数值。
5. 伺服驱动器主回路供电电源处于不稳定状态，或遭受雷击影响。	测量伺服驱动器主回路供电电源输入端上的的交流电压是否稳定。	接入浪涌抑制器后，再接通控制电和主回路电，若仍然发生故障，则更换伺服驱动器。
6. 伺服驱动器故障，母线电压采样异常	查看 H0B26 数值和伺服驱动器 P、C 端子母线电压实测值是否一致。	H0B26 数值和伺服驱动器 P、C 端子母线电压实测值不一致，请更换伺服驱动器。

Er.430: 控制电欠压

产生机理：

220V 驱动器：正常值：310V，故障值：190V

380V 驱动器：正常值：540V，故障值：350V

原因	确认方法	处理措施
1. 伺服驱动器控制电 (L1C、L2C) 切断过程中发生	伺服驱动器的控制电 L1C、L2C 在切断过程中，掉电速度较慢，可能会报出 Er.430。重新上电后该故障自动清除。倘若此时机器人控制器断电更慢，该故障可能会被控制器捕获并保存，导致重新上电时，伺服驱动器无故障，但控制器却显示驱动器故障报警 (实则为掉电过程控制器捕获到伺服驱动的该 Er430 故障)	重新上电后，伺服驱动器无故障，此时可以直接清除机器人控制器上的故障报警，不必理会该 Er.430 故障。
2. 伺服驱动器控制电线缆接触不良	检测线缆是否连通，并测量控制电线缆驱动器侧 (L1C、L2C) 的电压是否符合以上要求。	重新接线或更换线缆

Er.620: 电机过载

产生机理：

电机累积热量过高，且达到故障阈值。

原因	确认方法	处理措施
1. 电机抱闸没有松开，电机拖着没松开的抱闸一起旋转，造成负载太重，电机输出有效转矩长时间超过额定值。	检查电机抱闸，确认伺服驱动器上使能后，抱闸是否能可靠松开。 在保证安全的前提下，可手动控制抱闸 24V 继电器吸合，查看电机抱闸是否可以正常打开。	确保电机抱闸没有损坏。 确保电机抱闸电气接线正确，在上使能时电机抱闸可以正常松开。 检查伺服驱动内抱闸参数是否正确设置。
2．加速度太大，速度太大，加减速太过频繁，导致电机输出有效转矩长时间持续超过额定转矩。	查看机器人控制器中运动参数中的加速度或速度是否设置过大，查看伺服驱动器的最大转矩，速度波形是否过大。 查看伺服驱动器平局负载率 H0B12 是否长时间大于 100.0%	降低加减速度或速度 或减轻负载，或更换大容量伺服驱动器及匹配的电机。
3. 负载惯量过大，加减速太频繁	计算机械惯量比或进行伺服惯量辨识，查看 H0815 惯量比设置是否合理。 确定运动节拍时间是否过快。	更改负载夹治具设计或安装，减小负载夹治具的偏心程度。 降低加速度或速度，增大节拍运行时间。
4. 因机械因素而导致的卡顿，或摩擦过大，从而导致运行时负载过大	手动旋转机械，检查机械在运动过程中，是否有明显的卡顿或摩擦阻力变大的现象。 与合格机械进行交叉对比测试，查看机械上的阻力是否不同。	排除机械因素
5. 伺服驱动器故障	在排除上述 4 中情况下，重新上下电后，低速缓慢运行仍报该故障	更换伺服驱动器

Er.630: 电机堵转

产生机理：

电机实际转速低于 10rpm，但转矩指令达到限定值，且持续超过 H0A32 设定值。

原因	确认方法	处理措施
1. 驱动器到电机间的 U V W 动力线的相序接错。	检查驱动器侧、电机侧以及中间接插件的 U V W 动力线的相序是否正确。	按照正确配线相序重新接线。 注：U V W 共有 6 中排序组合，只有一种相序正确，其它相序会导致 Er.630 堵转或 Er.234 飞车，其中 Er.234 飞车存在损坏机械或伤人风险，操作时注意安全。
2. 驱动器到电机间的 U V W 动力线任意一项或多项出现断线现象。	检查驱动器侧、电机侧以及中间接插件的 U V W 动力线是否可靠连接，检查机器人底座及电控柜端航插内的动力线是否可靠连接，是否发生退针，脱线，虚焊等现象。	按照正确配线重新接线，或更换线缆或接插件。
3. 电机旋转时，抱闸没有松开。	检查电机抱闸，确认伺服驱动器上使能后，抱闸是否能可靠松开。 在保证安全的前提下，可手动控制抱闸 24V 继电器吸合，查看电机抱闸是否可以正常打开。	确保电机抱闸没有损坏。 确保电机抱闸电气接线正确，在上使能时电机抱闸可以正常松开。 检查伺服驱动内抱闸参数是否正确设置。
4. 多台伺服驱动器和电机间编码器或动力线相互交叉接错。	多台伺服驱动器和电机间编码器或动力线相互交叉接错，即这台驱动器的动力线或编码器线接到其它的电机上。	检查多台伺服驱动器和电机间编码器或动力线的连接是不是一一对应。
5. 因机械因素导致电机堵转	检查机械上是否把电机轴卡死，不能旋转。	排查机械因素

Er.730: 电机的编码器电池断线

产生机理：

该编码器电池断线故障，属于编码器故障的一种，在该故障产生，机器人各轴不动且电

控柜不重新上电的状态下，可以通过编码器子故障查询码 H0B28 和 H0B68，查询该故障产生的机理和对象。H0B28 和 H0B68 编码器子故障类型见最后的附录。

注意：该故障下，编码器的零点丢失，需要在机器人机械零点位置，H0d20 设 2，清除

编码器多圈数据和故障后，重新标定零点。

原因	确认方法	处理措施
1. 编码器电池线缆断线	用万用表测量电机侧编码器电池电压是否为接近 0V	更换或维修电机编码器电池线缆，重新上电后，H0d20 设置 1，清除故障后，再重新上电。
2. 编码器电池线缆接触不良	检查机器人底座处电池接插件是否接触良好 检查电机侧编码器接插件是否接触良好 检查机器人底座和小臂波纹管线缆捆扎处线缆是否有磨损或断线线缆现象	更换或维修编码器线缆或接插件，重新上电后，H0d20 设置 1，清除故障后，再重新上电。
3. 编码器电池正负极接反	检查编码器电池正负极是否接反，红色为正极，黑色为负极。	按照正确电池极性重新接入，重新上电后，H0d20 设置 1，清除故障后，再重新上电。
4. 编码器电池电压脱落	检查机器人底座编码器电池是否因震动而脱落	重新正确接入电池后，重新上电，H0d20 设置 1，清除故障后，再重新上电。
5. 编码器电池电压过低	用万用表测量电机侧编码器电池电压是否远低于 3.6V	更换编码器电池，重新上电后，H0d20 设置 1，清除故障后，再重新上电。
6. 编码器损坏	编码器电池电压正常，电池线缆接触良好，反复上下电，会持续报该故障，且 H0B28 编码器子故障码的 Bit4 或 Bit11 位被置 1 或 H0B68 编码器子故障的 Bit12 被置 1	更换电机

Er.731 电机的编码器电池欠压

产生机理：

当编码器电池电压低于 3.2V 时产生该故障，该编码器电池欠压故障属于编码器故障

的一种，在该故障产生，机器人各轴不动且电控柜不重新上电的状态下，可通过编码器子故障查询码 H0B28 和 H0B68，查询该故障产生的机理和对象。H0B28 和 H0B68 编码器子故障类型见最后的附录。

注意：该故障下，编码器的零点丢失，需要在机器人机械零点位置，H0d20 设 2，清除

编码器多圈数据和故障后，重新标定零点。

原因	确认方法	处理措施
1. 在机器人电控柜断电期间，发生过编码器电池线缆接触不良。	检查机器人底座处电池接插件是否接触良好 检查电机侧编码器接插件是否接触良好 检查机器人底座和小臂波纹管线缆捆扎处线缆是否有磨损或断线线缆现象	更换或维修编码器线缆或接插件，重新上电后，H0d20 设置 1，清除故障后，再重新上电。
2. 编码器电池正负极接反	检查编码器电池正负极是否接反，红色为正极，黑色为负极。	按照正确电池极性重新接入，重新上电后，H0d20 设置 1，清除故障后，再重新上电。
3. 编码器电池电压脱落	检查机器人底座编码器电池是否因震动而脱落	重新正确接入电池后，重新上电，H0d20 设置 1，清除故障后，再重新上电。
4. 编码器电池电压过低	用万用表测量电机侧编码器电池电压是否低于 3.2V	更换编码器电池，重新上电后，H0d20 设置 1，清除故障后，再重新上电。
5. 编码器损坏	编码器电池电压正常，电池线缆接触良好，反复上下电，会持续报该故障，且 H0B28 编码器子故障码的 Bit4 或 Bit11 位被置 1 或 H0B68 编码器子故障的 Bit12 被置 1	更换电机

Er.740 编码器干扰

产生机理：

编码器 Z 信号被干扰，导致 Z 信号对应的电角度偏差过大。

原因	确认方法	处理措施
1. 编码器 Z 信号受干扰	检查周围是否有大型设备产生干扰。 让伺服处于人“rdy”状态，手动逆时针旋转电机轴，监控 H0B10 是否平滑增大或减小，且一圈对应 5 个 0~360°。 若转动过程中 H0B10 有异常突变，则编码器损坏。 若转动过程中不报警，但伺服运行过程中报警，则干扰可能性大。	除去干扰源，电机线缆和编码器线线缆，分开走线。
2. 编码器故障	将电机处于同一位置，多次上电并查看 H0B10，电角度偏差应该在 ±30°内。	更换伺服电机

Er.755 尼康编码器故障

产生机理：

该故障是机器人专用超短电机（尼康编码器）的所有编码器故障种类的总故障码（即：

总称）。在该故障产生，机器人各轴不动且电控柜不重新上电的状态下，可通过编码器子故障查询码 H0B28 和 H0B68，查询该故障产生的机理和对象。H0B28 和 H0B68 编码器子故障类型见最后的附录。

原因	确认方法	处理措施
1. 在伺服驱动器断电时，更换过编码器电池后，第一次上电	进行过更换电池的操作。 子故障码 H0B28 的 Bit4=1 或 Bit10=1 或 Bit11=1. 子故障码 H0B68 为 0	H0d20 置 1 后，重新上电，故障被清除。 注意：该工况下，编码器的零点丢失，需要在机器人机械零点位置，H0d20 设 2，清除编码器多圈数据后，重新标定零点。
2. 在伺服驱动器断电时，编码器电池线缆发生过接触不良或断线，接插件退针的问题。	用万用表检测电机侧的电池电压是否为 3.6V。 子故障码 H0B28 的 Bit4=1 或 Bit10=1 或 Bit11=1. 子故障码 H0B68=0	维修或更换好编码器电池线缆，H0d20 置 1 后，重新上电，故障被清除。 注意：该工况下，编码器的零点丢失，需要在机器人机械零点位置，H0d20 设 2，清除编码器多圈数据后，重新标定零点。
3. 编码器线缆断线、接触不良。编码器接插件未接通、接触不良或退针。	采用万用表测量伺服驱动器 DB 端子上的编码器各线芯和电机侧的 9 孔插头里的编码器各线芯是否接通良好，是否断线。 检查编码器上各个接插件或接头处各线芯和插针是否接触不良，退针，脱线。 子故障码 H0B28 的 Bit4=1，Bit0=1 子故障码 H0B68=0	维修或更换编码器电池线缆，H0d20 置 1 后，重新上电，故障被清除。
4. 编码器屏蔽线未连接，断线，或接触不良情况下，外界干扰导致编码通讯帧错误。	采用万用表测量伺服驱动器 DB 端子上外壳和电机侧的 9 孔插头里的屏蔽层线芯是否接通良好，是否断线。 子故障码 H0B28 的 Bit1=1，Bit2=1 或 Bit3=1 子故障码 H0B68=0	维修或更换编码器屏蔽层线缆。 H0d20 置 1 后，重新上电，故障被清除。
5. 编码器电池正负极接反	检查编码器电池正负极是否接反，红色为正极，黑色为负极。 子故障码 H0B28 的 Bit11=1， 子故障码 H0B68=0	按照正确电池极性重新接入，重新上电后，H0d20 设置 1，清除故障后，再重新上电。 注意：该工况下，编码器的零点丢失，需要在机器人机械零点位置，H0d20 设 2，清除编码器多圈数据后，重新标定零点。
6. 编码器电池电压过低	用万用表测量电机侧编码器电池电压是否低于 3.2V。 子故障码 H0B28 的 Bit10=1 或 Bit11=1 子故障码 H0B68=0	更换编码器电池，重新上电后，H0d20 设置 1，清除故障后，再重新上电。 注意：该工况下，编码器的零点丢失，需要在机器人机械零点位置，H0d20 设 2，清除编码器多圈数据后，重新标定零点。
7. 编码器故障	子故障码 H0B28 的 Bit0=1、Bit5=1、Bit9=1，Bit12=1、Bit13=1 子故障码 H0B68 ≠ 0	更换伺服电机

Er.765 编码器温度过高

产生机理：

该故障码为编码器警告，起到提示作用，告知此时编码器温度过高，产生该警告时，电机可以正常运行，不会停机。编码器温度降低，或者重新上下电后，该警告清除。

原因	确认方法	处理措施
1. 编码器温度过高（警告）	子故障码 H0B28 的 Bit6=1 子故障码 H0B68=0	编码器温度降低，或者重新上下电后，该警告清除。

Er.B00: 位置偏差过大

产生机理：

位置控制模式下，位置偏差大于 6065h 设定值。

原因	确认方法	处理措施
1. 位置指令规划异常，导致伺服接收的位置指令过大，从而引起的位置偏差过大。	通过伺服后台调试软件查看控制器规划的位置指令 609A 是否出现异常	检查机器人运动指令是否正确。
2. 机器人加速度或速度过大，且负载较大，惯量较大，运行过程位置偏差过大。	检查机器人控制器各轴速度和加速度等运动参数设置是否合理。 通过伺服后台调试软件查看位置指令，位置反馈，以及位置偏差，位置跟随响应是否滞后过大，导致位置偏差过大。	根据负载大小，惯量大小，合理设置速度和加速度。
3. 伺服驱动器增益较弱，跟随响应不够。	检查伺服驱动器位置环增益和速度环增益设置是否合理。	根据负载情况，调节伺服速度环和位置增益，以及速度前馈增益，使之伺服跟随响应满足要求。
4. 位置偏差故障阈值 6066h 设置过小	确认位置偏差故障阈值 6065h 是否设置过小。	增大 6965h 设定值。
5. 因机械因素导致的电机轴卡顿或堵转	检查电机轴是否可以平滑转动	排查机械因素

Er.B01: 位置指令过速

产生机理：

机器人控制器规划的并发送给伺服的位置指令，换算成转速后，大于电机的最大转速。

电机无法跟随该位置指令。

原因	确认方法	处理措施
1. 机器人控制器位置指令规划异常，导致伺服接收位置指令过大，电机已最大转速运行都无法跟随该位置指令。	通过伺服后台调试软件查看控制器规划的位置指令 609A 是否出现异常	检查机器人运动指令是否正确。

1.4 H0B28 和 H0B68 编码器子故障查询表

H0B28 是伺服驱动器报出的编码器故障，是 16 进制显示的数值，将其转化成二进制，

由低位到高位分别对应下述表格里的编码器子故障类型，相应故障位二进行数值被置 1，表明该类型编码器子故障产生。

编码器子故障查询功能码	编码器子故障		子故障产生的可能原因
H0B28	Bit0	通讯超时	1. 编码器线缆断线 2. 编码器线缆接触不良 3. 编码器线缆屏蔽层未接好，通讯受干扰 4. 电机型号设置不对 5. 编码器损坏。
	Bit1	帧停止位错误	1. 编码器线缆屏蔽层未接好，通讯受干扰。 2. 编码器通讯线受强干扰 3. 机器人本体和电控柜地线未接好
	Bit2	CRC 校验错误	
	Bit3	数据字段错误	
	Bit4	位置计数错误	1. 编码器电池断线 2. 编码器电池发生过接触不良 3. 编码器损坏
	Bit5	编码器超速	1. 编码器损坏
	Bit6	编码器过热	1. 该故障为警告，重启可消除。 2. 反复报出表明编码器存在损坏的风险
	Bit7	完全绝对状态	1. 编码器损坏
	Bit8	通讯校验错误	1. 编码器线缆断线 2. 编码器线缆接触不良 3. 编码器屏蔽线未可靠连接
	Bit9	多圈计数溢出	1. 编码器电池异常 2. 编码器损坏
	Bit10	电池报警	1. 编码器电池线缆接触不良 2. 编码器电池线缆断线 3. 编码器电池电压低于 3.2V 4. 编码器损坏
	Bit11	电池失效	1. 编码器电池未接入 2. 编码器电池线缆断线 3. 编码器损坏
	Bit12	多圈计数错误	1. 编码器损坏
	Bit13	计数增量异常	1. 编码器线缆通讯受干扰 2. 编码器损坏
	Bit14	零点搜索失败	无此功能
	Bit15	分频计算溢出	无此功能

H0B68 是编码器 MCU 芯片自身报出的编码器故障，是 16 进制显示的数值，将其转化成二进制，由低位到高位分别对应下述表格里的编码器子故障类型，相应故障位二进行数值被置 1，表明该类型编码器子故障产生。

编码器子故障查询功能码	编码器子故障		子故障产生的可能原因
H0B68	Bit0	MCU 运算超时	编码器损坏
	Bit1	串行编码器超时	
	Bit2	电流采样超时	
	Bit3	保留位	
	Bit4	直线编码器断线	
	Bit5~Bit7	保留位	
	Bit8	单圈解算错误 (TX 端)	
	Bit9	EEPROM 繁忙 (TX 端)	
	Bit10	位置计数错误 (TX 端)	
	Bit11	多圈计数错误 (TX 端)	
	Bit12	电池失效 (TX 端)	
	Bit13	编码器超速 (TX 端)	
	Bit14	编码器过热 (TX 端)	
	Bit15	EEPROM 故障 (TX 端)	

附录 2：机器人报警及处理方法

说明：系统故障分为 3 个等级：
0：报警，掉使能
1：报警，不掉使能
2：警告（不掉使能）

故障码	等级	中文注释	故障原因	处理方法
0x0001	0	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	0	示教器通讯模块调度失败	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	0	视觉通讯模块调度失败	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	0	内部通信模块调度失败	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	0	再现、示教功能模块调度失败	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	0	数据插补模块调度失败	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	0	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	0	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	0	译码错误	程序语法错误	检查程序编写规范
0x000A	0	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	0	IO 等待时间超时	IO 等待的时间超出设置时间	1. 检查 IO 端口； 2. 重新设置等待时间
0x000C	0	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	0	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	0	运动指令译码错误	运动指令译码错误	检查示教程序
0x000F	0	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	0	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	0	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	0	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	0	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；	从新确认 IRLink 配置
0x0014	2	示教程序保存失败	SD 卡松动或无法识别	检查 SD 卡
0x0015	0	系统内部与运动模块通信错误	DSP 软件运行错误	控制器重新上电
0x0016	0	内部使能错误	DSP 软件运行错误	控制器重新上电
0x0017	0	系统运动模块程序运行错误	DSP 软件运行错误	控制器重新上电
0x0018	0	回零失败	相对编码器回零失败	重新进行回零
0x0019	0	使能缺失	运行状态丢失使能	检查系统是否使能状态
0x001A	1	数据恢复失败	铁电数据恢复不成功	检查系统是否存在铁电
0x0020	2	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	0	传入参数错误	译码传入参数错误	检查指令参数
0x0022	0	找不到指令行	输入行号超出范围	选择运行的指令行
0x0023	0	找不到点数据	点未定义	检查点是否定义
0x0024	0	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	0	点数据坐标系参数错误	坐标系值超出范围	重新取点
0x0026	0	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	0	V 参数超出范围	V 参数超出范围（1-100）	修改 V 参数
0x0028	0	Z 参数超出范围	Z 参数超出范围（0-5）	修改 Z 参数

故障码	等级	中文注释	故障原因	处理方法
0x0029	0	TOOL 参数超出范围	TOOL 参数超出范围 (0-15)	修改 Tool 参数
0x002A	0	USER 参数超出范围	USER 参数超出范围 (0-15)	修改 User 参数
0x002B	0	ACC 参数超出范围	ACC 参数超出范围 (1-100)	修改 Acc 参数
0x002C	0	until 参数超出范围	IO 号超出范围 (0-255)	修改 Until In 参数
0x002D	0	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	0	托盘未定义	托盘序号未定义	托盘定义后使用
0x002F	0	Repeat 参数错误	Repeat 参数超出范围	修改 Repeat 参数
0x0030	0	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	0	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	0	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	0	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	2	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	0	网络被干扰	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	0	视觉指令错误	视觉指令打开错误	检查视觉指令
0x0050	1	SD 卡拔出	SD 卡连接后移除	检查 SD 卡的连接情况
0x0051	0	EtherCAT 断开	EtherCAT 连接后断开	检查 EtherCAT 通信情况
0x0052	0	IRLink 断开	IRLink 连接后断开	检查 IRLink 通信情况
0x0053	0	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	0	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	0	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	0	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	0	系统固件初始化错误	FPGA 固件错误或启动异常	重启或检查 FPGA 固件
0x0058	0	系统固件初始化错误	FPGA 固件错误或启动异常	重启或检查 DSP 固件
0x0059	2	运行状态下不允许切换控制设备	运行时切换控制设备	停止后才能切换控制设备
0x005A	0	设置 IO 参数错误	IO 相关参数设置错误	检查 IO 参数设置情况
0x005B	0	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	2	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	0	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	0	（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	0	（用户自定义报警 2）	同上	同上
0x0060	0	（用户自定义报警 3）	同上	同上
0x0061	0	（用户自定义报警 4）	同上	同上
0x0062	0	（用户自定义报警 5）	同上	同上
0x0063	0	（用户自定义报警 6）	同上	同上
0x0064	0	（用户自定义报警 7）	同上	同上
0x0065	0	（用户自定义报警 8）	同上	同上
0x0066	0	（用户自定义报警 9）	同上	同上
0x0067	0	（用户自定义报警 10）	同上	同上
0x0068	0	（用户自定义报警 11）	同上	同上
0x0069	0	（用户自定义报警 12）	同上	同上
0x006A	0	（用户自定义报警 13）	同上	同上
0x006B	0	（用户自定义报警 14）	同上	同上
0x006C	0	（用户自定义报警 15）	同上	同上
0x006D	0	（用户自定义报警 16）	同上	同上
0x006E	0	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	0	干涉区 2 报警	同上	同上
0x0070	0	干涉区 3 报警	同上	同上
0x0071	0	干涉区 4 报警	同上	同上

故障码	等级	中文注释	故障原因	处理方法
0x0072	0	干涉区 5 报警	同上	同上
0x0073	0	干涉区 6 报警	同上	同上
0x0074	0	干涉区 7 报警	同上	同上
0x0075	0	干涉区 8 报警	同上	同上
0x0076	0	干涉区 9 报警	同上	同上
0x0077	0	干涉区 10 报警	同上	同上
0x0078	0	干涉区 11 报警	同上	同上
0x0079	0	干涉区 12 报警	同上	同上
0x007A	0	干涉区 13 报警	同上	同上
0x007B	0	干涉区 14 报警	同上	同上
0x007C	0	干涉区 15 报警	同上	同上
0x007D	0	干涉区 16 报警	同上	同上
0x007F	2	数据流模式未关闭	同上	同上
0x0080	0	运行模式按下急停	急停键被按下	解除急停，清除报警
0x0081	0	无后退数据	后退数据已经执行完毕	终止后退操作
0x0082	0	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	0	TCP 端口溢出	外围 TCP 应用连接过多	关闭无用的 TCP 连接
0x0084	0	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	0	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	2	IP 冲突	IP 设置冲突	重新设置 IP
0x008A	0	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	0	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	0	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	0	不合法的 IO 配置	配置的 IO 被 PLC 控制或不存在	重新配置 IO
0x008E	2	不合法的 IO 设置操作	设置的 IO 缺少 IO 控制权	检查 IO 控制权
0x008F	2	IO 不存在	使用的 IO 号超过了其配置的 IO 总数	检查 IO 是否存在： 检查实物连接 检查 IRlink 配置 检查程序使用的 IO 是否正确
0x0090	0	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	0	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	0	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	0	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	0	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	0	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	0	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	0	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	0	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	0	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	0	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	0	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	0	读写的 TCP 端口未打开	读写的 TCP 端口与开启的 TCP 端口不对应	检查指令，匹配端口
0x00A1	0	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

故障码	等级	中文注释	故障原因	处理方法
0x00A2	0	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	0	动态视觉坐标转换错误	像素坐标转换为机器人坐标出错	检查相应视觉编号的标定结果是否正确
0x00A4	0	无后退数据	没有可以后退的数据	清除报错
0x00A5	0	圆弧运动前缺乏 Movj 或 Movl	圆弧运动前一条指令不是 Movj 或 Movl（仅在单步示教时出现）	保证圆弧指令前一条指令是为 Movj 或 Movl
0x00A6	0	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	2	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	2	导出状态错误	锁螺丝机导出状态错误	重新导出
0x00A9	2	文件数量超限	当前文件夹下文件（含文件夹）数量过多	删除过多的文件，保证单个文件夹下的文件 + 文件夹数目不超过 100 个
0x00AA	0	暂停缓冲区数据发送失败	从暂停到启动时，缓冲数据发送失败	重新从停止开始启动
0x00AB	0	多任务编号不存在	任务编号写错或使用范围不对	检查任务编号
0x00AC	0	多任务编号错误或被占用	任务编号错误或已经被占用	检查任务编号
0x00AD	0	多任务创建错误	系统忙或其它未知原因导致系统创建不成功	重新创建
0x00AE	0	开启或关闭位置锁存功能失败	开启或关闭位置锁存功能失败	检查硬件连接或重启系统
0x00AF	2	控制器调试功能调用失败	控制器调试功能模块失效	检查控制器调试开关的配置是否正确
0x00B0	2	负载参数设置失败	负载参数失败不成功或超出范围	重新设置或检查参数
0x00B1		读写的串口未打开	读写的串口与开启的串口不对应	检查指令，匹配串口
0x1001	2	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	2	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	2	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	2	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	2	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	2	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	2	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	2	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	2	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	2	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	2	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	2	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	0	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	0	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	0	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	0	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	2	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	2	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	0	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	0	内部交换的数据长度无效	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	0	内部交换的数据校验错误	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	0	系统内部写入快数据错误	GPMC 通道异常	检查 DSP 软件版本或更换硬件
0x1104	0	系统内部读入快数据错误	GPMC 通道异常	检查 DSP 软件版本或更换硬件
0x1105	0	系统内部块数据缓冲区满	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

故障码	等级	中文注释	故障原因	处理方法
0x1106	0	内部通道打开错误：异常或被占用	GPMC 通道异常或已经被占用	重启
0x1107	0	内部通道打开错误：异常或被关闭	GPMC 通道异常或已经被关闭	重启
0x1108	0	系统内部 BUSY	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	0	系统内部试图获取通道资源错误	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	0	系统内部等待运动模块应答超时	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	0	系统运动模块未能有效执行指令	DSP 对 ARM 指令执行不成功	检查 DSP 软件版本或更换硬件
0x110C	0	设置给运动模块的参数非法	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	0	设置给运动模块的命令非法	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	0	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	0	系统设置的轴数据和再次从运动模块获取的轴数据不一致	数据校验出错	检查软件版本或更换硬件
0x1110	0	系统设置的 IO 数据和再次从运动模块获取的 IO 数据不一致	数据校验出错	检查软件版本或更换硬件
0x1111	0	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	0	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	0	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	0	系统设置参数校验错误	数据校验出错	检查软件版本或更换硬件
0x1115	0	打开数据通道错误	GPMC 通道号不正确	默认通道为 0，确保通道号正确
0x1116	0	数据通道映射错误	GPMC 数据通道无法映射至内存	检查软件版本或更换硬件
0x1117	0	数据通道映射错误	GPMC 通道异常	检查软件版本或更换硬件
0x1118	0	数据通道映射错误	GPMC 通道异常	检查软件版本或更换硬件
0x1119	0	设备错误	GPMC 通道异常	检查软件版本或更换硬件
0x111A	0	数据通道打开错误	GPMC 通道异常	检查软件版本或更换硬件
0x111B	0	数据通信不一致	GPMC 通道异常	检查软件版本或更换硬件
0x111C	0	数据通信不一致	GPMC 通道异常	检查软件版本或更换硬件
0x111D	0	系统申请内存出错	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	0	IRlink 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	0	IRlink 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	0	IRlink 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	0	IRlink 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	0	IRlink 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	0	IRlink 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	0	IRlink 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	0	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

故障码	等级	中文注释	故障原因	处理方法
0x2001	0	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	0	圆弧输入参数计算错误	无法计算出圆弧插补信息： （1）至少两点距离太近 （2）三点近似共线 （3）姿态变换太大 （4）与其他的轨迹过渡存在错误，如在奇异点附近过渡。	重新示教其他点计算圆弧
0x2003	0	直线输入参数计算错误	无法计算出直线插补信息： （1）姿态变换太大 （2）与其他的轨迹过渡存在错误，如在奇异点附近过渡	重新示教其他点计算直线
0x2004	0	逆解运算错误	机器人处于奇异区域或可达范围外。	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	0	奇异位置错误报警	机器人运动到奇异区域。	切换至关节模式，然后移动机器人离开奇异位置点
0x2006	0	再现运动中出现掉伺服	（1）某个的驱动器掉电 （2）某个的驱动器接线错误 （3）某个的驱动器故障	检查驱动器是否出现异常
0x2007	0	保留	姿态变化太大	重新示教其他点计算本段规划
0x2008	0	IO 的 Index 访问范围超出	IO 访问物理端不存在	检查是否有没有对应的物理 IO 模块
0x2009	0	jump 参数设置错误	MH 在第 3 轴限位范围外	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	0	臂型参数错误	直线或者圆弧运动时，终点前三个臂型参数与起点不一致。	将后条运动指令改成 movJ 或者重新取点，保证臂型一致
0x200B	0	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	0	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	0	发出伺服使能命令，但实际反馈未使能	（1）伺服主电未上 （2）关节是否在减速过程中 （3）关节还处在运动状态，没有到位。	（1）检查控制柜的强电按钮是否按下。 （2）停止和启动或上伺服之间的间隔时间太短。 （2）检查关节是否到位。放大到位误差，或调整伺服参数。
0x200E	0	关节运动输入参数错误	（1）针对 delta 机型，在空间范围外； （2）MoveJ 与 MOVL 或 MOVC 在奇异附近过渡。	（1）针对 delta 机型，检查点位是否在空间范围外； （2）检查 MoveJ 与 MOVL 或 MOVC 是否在奇异附近过渡。
0x200F	0	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	0	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	0	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	0	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	0	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	0	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	0	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	0	机器人速度异常	机器人关节速度超出了允许的最大速度的两倍	1) 确保设置界面的关节最大允许速度和关节最大允许加速度设置合理； 2) 确保机器人不在在奇异位置附近； 3）降低直线运动的速度
0x2019	0	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

故障码	等级	中文注释	故障原因	处理方法
0x201A	0	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
0x201D	0	换臂型错误	在奇异附近，不支持换臂型指令（SCARA 专用）	替换成一个远离奇异区域的点，再调用换臂型指令
0x201E	0	机器人加速度异常	机器人关节速度超出了允许的最大加速度的 50 倍	1. 确保设置页面的关节最大允许速度和关节最大允许加速度设置合理 2. 确保机器人不在奇异位置附近 3. 降低直线运动的速度和加速度
0x201F	0	速度设置异常	关节速度设置太小，或空间轨迹设置速度过大，造成约束点过多	1. 增大关节速度或减小空间轨迹设置速度 2．确保机器人不在在奇异位置附近做空间轨迹的运动
0x2021	0	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	0	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，调整工作下界、拾取上界；提高机器人速度；降低传送带速度。
0x2023	0	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	0	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	0	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	视觉处理时间是否大于拍照触发周期
0x2026	0	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。	检查跟随过程中指令点位类型是否是 7（动态物体坐标系），或在非跟随期间使用了类型 7 的点位。
0x2027	0	动态点坐标错误	给定的动态目标位置错误、奇异或出界	检查视觉给出的做坐标是否在合理范围内。
0x2028	0	传送带跟随命令语法错误	连续使用了 RefConvyor 或 RefBase	检查是否 RefConvyor 和 RefBase 配对使用
0x2029	0	建立抓取工件坐标系失败	没执行 GetCnvObject 就执行了 RefConvyor。	先调用 GetCnvObject 指令，后跟随
0x202A	0	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	0	不允许单步示教	传送带给跟随相关的指令不允许单步示教	不允许对 RefConvyor 和 RefBase 之间的指令进行单步操作
0x202C	0	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	检查程序中使用的传输带编号，是否未使能
0x202D	0	跟随过程中不允许 PTP 运动	跟随工艺中使用了 MovJ，JumP 等关节运动	检查 RefConvyor 和 RefBase 之间是否使用了 MovJ 或者 jump 指令（注：delta 中可使用 jump）
0x202E	0	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的方向参数是否正确。 2. 检查传输带是否有打滑现象。
0x202F	0	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，无法逆解	关节移除奇异位置
0x2030	0	各轴锁存计数器不一致	锁存机器人位置时，各轴锁存计数器不一致	请检查各轴的锁存信号是否有效
0x2031	0	锁存缓存已满	锁存机器人位置时，锁存缓存已满	锁存速度太快，或者锁存太多并未使用
0x2041	0	保留（15D 以后再定义）	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	0	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	0	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

故障码	等级	中文注释	故障原因	处理方法
0x2044	0	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
0x2045	0	规划轨迹中第 1 关节触发限位	轨迹中第 1 关节触发限位	检查对应段的规划轨迹，中间过程点是否有触发限位
0x2046	0	规划轨迹中第 2 关节触发限位	轨迹中第 2 关节触发限位	检查对应段的规划轨迹，中间过程点是否有触发限位
0x2047	0	规划轨迹中第 3 关节触发限位	轨迹中第 3 关节触发限位	检查对应段的规划轨迹，中间过程点是否有触发限位
0x2048	0	规划轨迹中第 4 关节触发限位	轨迹中第 4 关节触发限位	检查对应段的规划轨迹，中间过程点是否有触发限位
0x2049	0	规划轨迹中第 5 关节触发限位	轨迹中第 5 关节触发限位	检查对应段的规划轨迹，中间过程点是否有触发限位
0x204A	0	规划轨迹中第 6 关节触发限位	轨迹中第 6 关节触发限位	检查对应段的规划轨迹，中间过程点是否有触发限位
0x2071	0	示教时，规划轨迹中第 1 关节触发限位	轨迹中第 1 关节触发限位	检查对应段的规划轨迹，中间过程点是否有触发限位
0x2072	0	示教时，规划轨迹中第 2 关节触发限位	轨迹中第 2 关节触发限位	检查对应段的规划轨迹，中间过程点是否有触发限位
0x2073	0	示教时，规划轨迹中第 3 关节触发限位	轨迹中第 3 关节触发限位	检查对应段的规划轨迹，中间过程点是否有触发限位
0x2074	0	示教时，规划轨迹中第 4 关节触发限位	轨迹中第 4 关节触发限位	检查对应段的规划轨迹，中间过程点是否有触发限位
0x2075	0	示教时，规划轨迹中第 5 关节触发限位	轨迹中第 5 关节触发限位	检查对应段的规划轨迹，中间过程点是否有触发限位
0x2076	0	示教时，规划轨迹中第 6 关节触发限位	轨迹中第 6 关节触发限位	检查对应段的规划轨迹，中间过程点是否有触发限位
0x2077	0	示教状态中，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
0x2078	0	示教状态中，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
0x2079	0	示教状态中，预处理时发现臂型与当前点不一致，不能进行空间插补。	预处理时发现 ARM_TYPE 与当前点不一致，不能进行空间插补。	修改出错段的位置取点
0x207A	0	示教状态中，发现姿态运动过大	规划轨迹中发现姿态运动角度大于 180 度	减小单次运动的姿态变动范围
0x207B	0	示教状态中，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
0x2101	0	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	0	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	0	轴 1 驱动报警	驱动器出现报警	伺服报警码详见监控 -> 伺服状态 ->J1-No.12、13 行，根据报警码查阅伺服手册做相应的故障排除
0x2104	0	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	0	轴 1 跟随误差过大报警	规划位置和实际位置差过大	调整伺服参数，将相应滞后减小
0x2106	0	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	0	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

故障码	等级	中文注释	故障原因	处理方法
0x2111	0	轴 2 正限位报警	同轴 1	同轴 1
0x2112	0	轴 2 负限位报警		
0x2113	0	轴 2 驱动报警		
0x2114	0	轴 2 规划溢出报警		
0x2115	0	轴 2 跟随误差过大报警		
0x2116	0	轴 2 速度过大报警		
0x2117	0	轴 2 的驱动强电未上		
0x2201	0	轴 3 正限位报警	同轴 1	同轴 1
0x2202	0	轴 3 负限位报警		
0x2203	0	轴 3 驱动报警		
0x2204	0	轴 3 规划溢出报警		
0x2205	0	轴 3 跟随误差过大报警		
0x2206	0	轴 3 速度过大报警		
0x2207	0	轴 3 的驱动强电未上		
0x2211	0	轴 4 正限位报警	同轴 1	同轴 1
0x2212	0	轴 4 负限位报警		
0x2213	0	轴 4 驱动报警		
0x2214	0	轴 4 规划溢出报警		
0x2215	0	轴 4 跟随误差过大报警		
0x2216	0	轴 4 速度过大报警		
0x2217	0	轴 4 的驱动强电未上		
0x2301	0	轴 5 正限位报警	同轴 1	同轴 1
0x2302	0	轴 5 负限位报警		
0x2303	0	轴 5 驱动报警		
0x2304	0	轴 5 规划溢出报警		
0x2305	0	轴 5 跟随误差过大报警		
0x2306	0	轴 5 速度过大报警		
0x2307	0	轴 5 的驱动强电未上		
0x2311	0	轴 6 正限位报警	同轴 1	同轴 1
0x2312	0	轴 6 负限位报警		
0x2313	0	轴 6 驱动报警		
0x2314	0	轴 6 规划溢出报警		
0x2315	0	轴 6 跟随误差过大报警		
0x2316	0	轴 6 速度过大报警		
0x2317	0	轴 6 的驱动强电未上		
0x8001	0	无网络设备错误	EtherCAT 通信初始化错误	检查 FPGA 及 DSP 固件加载是否成功
0x8002	0	无主站错误		
0x8003	0	无效域错误		
0x8004	0	无此从站错误		
0x8005	0	无效过程数据错误		
0x8006	0	无效服务数据错误		
0x8007	0	无效入口对象错误		
0x8008	0	域内存地址分配错误		
0x8009	0	激活主站失败错误		
0x800A	0	服务数据公共错误		
0x800B	0	注册周期回调错误		
0x800C	0	过程通信配置错误		
0x800D	0	初始化模块错误		
0x800E	0	解析配置错误		
0x800F	0	配置通道参数错误		
0x8010	0	域注册错误		
0x8011	0	创建定时器错误		
0x8012	0	启动定时器错误		

故障码	等级	中文注释	故障原因	处理方法
0x8013	0	配置 EtherCAT 通信周期错误	EtherCAT 从站配置错误	检查 EtherCAT 配置信息 确保配置正确保存后重启
0x8014	0	配置 EtherCAT 版本选择错误		
0x8015	0	配置 EtherCAT 伺服从站数错误		
0x8016	0	配置 EtherCAT IO 从站数错误		
0x8017	0	配置 EtherCAT IO 模块数错误		
0x8018	0	配置 EtherCAT IO 类型错误		
0x8019	0	配置 EtherCAT IO 不支持错误		
0x801A	0	配置 EtherCAT 内存申请错误		
0x801B	0	配置 EtherCAT 报警共享内存错误		
0x801C	0	配置 EtherCAT 伺服操作模式错误		
0x801D	0	配置 EtherCAT 寄存器错误		
0x801E	0	配置 EtherCAT IO 数量与在线 IO 数量不匹配错误		
0x801F	0	配置 EtherCAT 伺服数量与在线伺服数量不匹配错误		
0x8020	0	配置 EtherCAT 伺服供应商代码不支持错误		
0x8028	0	写缓冲区错误	EtherCAT 通信运行错误	检查 EtherCAT 网络连接情况 检查 EtherCAT 从站是否有掉电等
0x8029	0	写启动命令错误		
0x802A	0	读状态寄存器错误		
0x802B	0	读数据链路状态错误		
0x802C	0	读服务数据通道错误		
0x802D	0	读服务数据长度错误		
0x802E	0	服务数据长度错误		
0x802F	0	服务数据接收错误		
0x8030	0	服务数据通道忙错误		
0x8031	0	服务数据报文错误		
0x8032	0	读过程数据通信错误		
0x8033	0	读过程数据长度错误		
0x8034	0	过程数据长度错误		
0x8035	0	过程数据接收错误		
0x8036	0	网络设备打开错误		
0x8037	0	底层通信错误		
0x8038	0	底层通信错误		
0x8039	0	底层通信错误		
0x803A	0	读发送时间戳错误		
0x803B	0	读接收时间戳错误		
0x803C	0	读过程数据剩余错误		
0x803D	0	读应用时间戳错误		
0x803E	0	现场总线 LED 打开错误		
0x803F	0	现场总线 LED IOCTRL 错误		
0x8040	0	底层硬件无效		
0x8041	0	底层硬件无效		
0x8042	0	EtherCAT 从站掉线错误		确保 EtherCAT 从站网线已正确连接后； 掉电重启
0x8N42	0	EtherCAT 从站 N 掉线错误		
0x8043	0	接入了非 EtherCAT 从站设备错误		将 EtherCAT 网线插入 EtherCAT 网口
0x8044	0	EtherCAT 网口 1 未连接错误		
0x8045	0	EtherCAT 网口 2 未连接错误		
0x8046	0	设置 EtherCAT 启动时间错误	EtherCAT 周期通信启动错误	检查从站状态是否异常

故障码	等级	中文注释	故障原因	处理方法
0x805C	0	翻转位未改变错误	EtherCAT 从站服务数据通信错误	检查服务数据请求，如伺服功能码操作数据格式等
0x805D	0	SDO 协议超时错误		
0x805E	0	命令无效或未知错误		
0x805F	0	对象不可被访问错误		
0x8060	0	试图读一个只写对象错误		
0x8061	0	试图写一个只读对象错误		
0x8062	0	对象不存在对象字典中错误		
0x8063	0	对象不能被映射为过程数据错误		
0x8064	0	被映射的对象长度超出过程数据长度错误		
0x8065	0	基本参数不兼容错误		
0x8066	0	设备内部不兼容错误		
0x8067	0	硬件导致访问失败错误		
0x8068	0	服务参数长度不匹配错误		
0x8069	0	服务参数长度太长错误		
0x806A	0	服务参数长度太短错误		
0x806B	0	子索引不存在错误		
0x806C	0	参数值越界错误		
0x806D	0	所写参数值太大错误		
0x806E	0	所写参数值太小错误		
0x806F	0	最大值小于最小值错误		
0x8070	0	数据无法传输或存储错误		
0x8071	0	本地控制导致数据无法存储错误		
0x8072	0	设备状态导致数据无法存储错误		
0x8073	0	对象字典动态总错误或对象字典不存在错误		
0x807A	0	配置 IRlink 通信周期错误	IRlink 从站配置错误	检查 IRlink 配置信息 确保配置正确保存后重启
0x807B	0	配置 IRlink 版本选择错误		
0x807C	0	配置 IRlink 伺服从站数错误		
0x807D	0	配置 IRlink 从站数错误		
0x807E	0	配置 IRlink 模块数错误		
0x807F	0	配置 IRlink 类型错误		
0x8080	0	配置 IRlink 不支持错误		
0x8081	0	配置 IRlink 内存申请错误		
0x8082	0	配置 IRlink 报警共享内存错误		
0x8083	0	配置 IRlink 伺服操作模式错误		
0x8084	0	配置 IRlink 寄存器错误		
0x8085	0	配置 IRlink IO 数量与在线 IO 数量不匹配错误		
0x8086	0	配置 IRlink 从站数量与在线从站数量不匹配错误		
0x8087	0	配置 IRlink 伺服供应商代码不支持错误		

故障码	等级	中文注释	故障原因	处理方法
0x8090	0	写缓冲区错误	IRlink 运行错误	检查 IRlink 网络连接情况； 检查 IRlink 从站是否掉电等
0x8091	0	读服务数据通道错误		
0x8092	0	读服务数据长度错误		
0x8093	0	服务数据长度错误		
0x8094	0	服务数据接收错误		
0x8095	0	服务数据通道忙错误		
0x8096	0	服务数据报文错误		
0x8097	0	读过程数据通信错误		
0x8098	0	读过程数据长度错误		
0x8099	0	过程数据长度错误		
0x809A	0	过程数据接收错误		
0x809B	0	网络设备打开错误		
0x809C	0	GPMC IRlink 读错误		
0x809D	0	GPMC IRlink 写错误		
0x809E	0	IRlink 从站掉线错误		确保 IRlink 从站网线已正确连接后； 掉电重启
0x8N9E	0	IRlink 从站 N 掉线错误		
0x809F	0	接入了非 IRlink 从站设备错误		
0x80A0	0	IRlink 网口 0 未连接错误	IRlink 网口未连接	将 IRlink 网线插入 IRlink 网口
0x80A1	0	IRlink 网口 1 未连接错误		
0x80A2	0	设置 IRlink 启动时间错误	IRlink 周期信启动错误	检查从站状态是否异常
0xE001	0		正常运行告警	无需处理
0xE002	0	固件加载失败	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	0	固件加载失败	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	0	固件复位失败	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	0	固件加载失败	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	0	固件加载失败	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	0	控制通道 1 运行正常	正常运行告警	无需处理
0xE008	0	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	0	固件加载失败	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	0	固件加载失败	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	0	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	0	固件加载失败	FPGA 可能没有正常工作	检查 FPGA 运行灯是否正常闪烁
0xE00D	0	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	0	SPI 通信失败	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	0	SPI 通信失败	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	0	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	0	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	0	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	0	控制通道 0 运行正常	正常运行告警	无需处理
0xE014	0	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

故障码	等级	中文注释	故障原因	处理方法
0xE015	0	固件加载失败	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	0	固件加载失败	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	0	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	0	固件加载失败	FPGA 可能没有正常工作	检查 FPGA 运行灯是否正常闪烁
0xE019	0	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	0	SPI 通信失败	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	0	SPI 通信失败	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	0	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	0	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	0	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 3： API 查询

3.1 API 故障连接表

序号	函数返回值	故障打印信息	说明	故障处理方法
0	0	/	(指令正常)	/
1	-1	e1:syntax error	指令语法错误	动态链接库版本问题，寻求技术支持。
2	-2	e2:number of parameter unmatched	指令参数个数不匹配	动态链接库版本问题，寻求技术支持。
3	-3	e3:parameter value illegal	指令参数值不合理	依据函数说明，重新给定函数参数。
4	-4	e4:not allowed in current mode	当前模式下不允许该指令	根据实际情况切换示教或再现模式后重新调用。
5	-5	e5:not allowed when robot is running	机器人运行时不允许该指令	机器人停止后重新调用。
6	-6	e6:system in emergency	系统处于急停状态	系统处于急停松开后重新调用。
7	-7	e7:system fault	系统有故障	清除系统故障后重新调用。
8	-8	e8:motion mode closed	数据流模式关闭	打开数据流模式后重新调用
9	-9	e9:motor off	电机未使能	电机使能后重新调用。
10	-10	e10:rsv	保留	保留
11	-11	e11:instruction unfinished	运动指令缓冲区仍有未完成的指令	待运动指令缓冲区空闲时重新调用。
12	-12	e12:ouput unavaiable	用户无权限控制输出端口	获取该端口权限后重新调用，或者控制其他可用端口。
13	-13	e13:read AD failed	读取 AD 通道失败	确认模块正确配置后重新调用。
14	-14	e14:write DA failed	写入 AD 通道失败	确认模块正确配置后重新调用。
15	-15	e15:write DO failed	写入 DO 通道失败	确认模块正确配置后重新调用。
16	-16	e16:command invalid	暂停指令无效	程序停止状态下不调用暂停函数。
17	-17	e17:not allowed in current coordinate	当前坐标系下不允许该指令	切换坐标系后重新调用。
18	-18	e18:mode conflict	运动模式冲突	关闭数据流模式后重新调用函数。
19	-19	e19:rsv	保留	保留
20	-20	e20:program non-existent	设置的程序路径不存在	重新输入路径或者重新编辑示教器程序。
21	-21	e21:point non-existent	位置点 P 或偏移 LPR 不存在	确认点位无误并重新打开示教程序后调用。
22	-22	e22:calc error	内部计算错误	确认不在奇异位置后重新调用。
23	-23	e23:rsv	保留	保留
24	-24	e24:without permit	当前以太网设备未获得控制许可	切换控制权并获取控制许可后重新调用。
25	-25	e25:ETH without authorization	当前控制权不是以太网设备	切换控制权后重新调用。
26	-26	e26:rsv	保留	保留
27	-27	e27:low user grade	当前用户权限等级不够	登陆更高用户等级后重新调用
28	-28	e28:permit occupied	控制许可已被其它以太网设备申请	强制获取控制许可后重新调用。

序号	函数返回值	故障打印信息	说明	故障处理方法
29	-29	e29:internal fault	内部调用错误	动态链接库版本问题，寻求技术支持。
30	-30	e30:modbus unavailable	未配置 modbus 从站	配置 modbus 从站后重新调用。
31	-31	e31:condition unfulfilled to write	当前不可设置系统参数	机器人停止后重新调用。
32	-32	e32:rsv	保留	保留
33	-253	/	内部计算异常	动态链接库版本问题，寻求技术支持。
34	-254	/	函数输入参数错误	依据函数说明，重新给定函数参数。
35	-255	/	网络通信异常	检查网络连接确认无误后重新调用。

3.2 API 函数

序号	函数名称	说明	参数	返回值	备注
1	int IMC100_Init_ETH(unsigned int ipAddr,unsigned short 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，主机字节序 ipPort：机器人控制器网络端口号，默认 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	
2	int IMC100_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	int IMC100_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	
4	int IMC100_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	
5	int IMC100_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
6	int IMC100_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	
7	int IMC100_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	
8	int IMC100_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	
9	int IMC100_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	
10	int IMC100_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	
11	int IMC100_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	

序号	函数名称	说明	参数	返回值	备注
12	int IMC100_Set_DOGroup(int num, int status, int comId=0)	按组设置 DO 的输出状态	num：DO 组序号 , 最大范围 0-7，依据实际配置情况 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	
13	int IMC100_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	
14	int IMC100_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	
15	int IMC100_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	
16	int IMC100_Jog(int mode, int axis, int cmd, int comId=0)	示教运动命令	mode：示教方式，0- 关节示教，1- 直角坐标示教 axis：轴序号，范围 1-6，关节示教时分别对应 J1-J6 轴，直角坐标示教时分别对应 X/Y/Z/RZ/RY/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	示教模式下数据流模式关闭后有效
17	int IMC100_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	仅在数据流模式下有效
18	int IMC100_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	仅在数据流模式下有效
19	int IMC100_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	仅在数据流模式下有效
20	int IMC100_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，详见定义 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	仅在数据流模式下有效
21	int IMC100_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，详见定义 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	仅在数据流模式下有效
22	int IMC100_MovJ_IO(int posNum, int vel, int zone, MOV_IO *movIo, int ioNum, int comId=0)	关节插补方式运动到指定序号的位置点，同时控制对应 IO	posNum：目标位置点序号 vel：运动速度，范围 1-100 zone：插补精度，范围 0-5 movIo：IO 控制结构体，详见定义 ioNum：IO 控制结构体的组数，范围 1-3	返回 0 表示命令发送成功，小于 0 表示失败	仅在数据流模式下有效
23	int IMC100_MovL_IO(int posNum, int vel, int zone, MOV_IO *movIo, int ioNum,int comId=0)	直线插补方式运动到指定序号的位置点，同时控制对应 IO	posNum：目标位置点序号 vel：运动速度，范围 1-100 zone：插补精度，范围 0-5 movIo：IO 控制结构体，详见定义 ioNum：IO 控制结构体的组数，范围 1-3	返回 0 表示命令发送成功，小于 0 表示失败	仅在数据流模式下有效
24	int IMC100_MovJ2_IO(ROBOT_POS pos, int vel, int zone, MOV_IO *movIo, int ioNum, int comId=0)	关节插补方式运动到指定值的位置点，同时控制对应 IO	pos：位置参数结构体，详见定义 vel：运动速度，范围 1-100 zone：插补精度，范围 0-5 movIo：IO 控制结构体，详见定义 ioNum：IO 控制结构体的组数，范围 1-3	返回 0 表示命令发送成功，小于 0 表示失败	仅在数据流模式下有效
25	int IMC100_MovL2_IO(ROBOT_POS pos, int vel, int zone, MOV_IO *movIo, int ioNum, int comId=0)	直线插补方式运动到指定值的位置点，同时控制对应 IO	pos：位置参数结构体，详见定义 vel：运动速度，范围 1-100 zone：插补精度，范围 0-5 movIo：IO 控制结构体，详见定义 ioNum：IO 控制结构体的组数，范围 1-3	返回 0 表示命令发送成功，小于 0 表示失败	仅在数据流模式下有效
26	int IMC100_Jump(int posNum, int vel=100, int zone=0, int comId=0)	跳跃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	仅在数据流模式下有效

序号	函数名称	说明	参数	返回值	备注
27	int IMC100_JumpL(int posNum, int vel=100, int zone=0, int comId=0)	直线跳跃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	仅在数据流模式下有效
28	int IMC100_Jump2(ROBOT_POS pos, int vel=100, int zone=0, int comId=0)	跳跃方式运动到指定值的位置点	pos：位置参数结构体，详见定义 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	仅在数据流模式下有效
29	int IMC100_JumpL2(ROBOT_POS pos, int vel=100, int zone=0, int comId=0)	直线跳跃方式运动到指定值的位置点	pos：位置参数结构体，详见定义 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	仅在数据流模式下有效
30	int IMC100_Jump_IO(int posNum, int vel, int zone, MOV_IO *movIo, int ioNum, int comId=0)	跳跃方式运动到指定序号的位置点，同时控制对应 IO	posNum：目标位置点序号 vel：运动速度，范围 1-100 zone：插补精度，范围 0-5 movIo：IO 控制结构体，详见定义 ioNum：IO 控制结构体的组数，范围 1-3	返回 0 表示命令发送成功，小于 0 表示失败	仅在数据流模式下有效
31	int IMC100_JumpL_IO(int posNum, int vel, int zone, MOV_IO *movIo, int ioNum,int comId=0)	直线跳跃方式运动到指定序号的位置点，同时控制对应 IO	posNum：目标位置点序号 vel：运动速度，范围 1-100 zone：插补精度，范围 0-5 movIo：IO 控制结构体，详见定义 ioNum：IO 控制结构体的组数，范围 1-3	返回 0 表示命令发送成功，小于 0 表示失败	仅在数据流模式下有效
32	int IMC100_Jump2_IO(ROBOT_POS pos, int vel, int zone, MOV_IO *movIo, int ioNum, int comId=0)	跳跃方式运动到指定值的位置点，同时控制对应 IO	pos：位置参数结构体，详见定义 vel：运动速度，范围 1-100 zone：插补精度，范围 0-5 movIo：IO 控制结构体，详见定义 ioNum：IO 控制结构体的组数，范围 1-3	返回 0 表示命令发送成功，小于 0 表示失败	仅在数据流模式下有效
33	int IMC100_JumpL2_IO(ROBOT_POS pos, int vel, int zone, MOV_IO *movIo, int ioNum, int comId=0)	直线跳跃方式运动到指定值的位置点，同时控制对应 IO	pos：位置参数结构体，详见定义 vel：运动速度，范围 1-100 zone：插补精度，范围 0-5 movIo：IO 控制结构体，详见定义 ioNum：IO 控制结构体的组数，范围 1-3	返回 0 表示命令发送成功，小于 0 表示失败	仅在数据流模式下有效
34	int IMC100_Get_PosHere(ROBOT_POS *pos, int comId=0)	查询当前位置点的位置参数（与当前坐标系相关）	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	
35	int IMC100_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	
36	int IMC100_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	
37	int IMC100_Get_PosCnvt(ROBOT_POS *posSrc, ROBOT_POS *posDst, int comId=0)	查询位置参数的坐标系转换结果	posSrc：原始坐标参数结构体 posDst：目标坐标参数结构体，代表转换的结果，其中 coord、toolNo、userNo 表示转换所参考的坐标系参数，需用户提前写入	返回 0 表示查询成功，小于 0 表示失败	
38	int IMC100_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	
39	int IMC100_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	
40	int IMC100_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	
41	int IMC100_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	
42	int IMC100_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	
43	int IMC100_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	

序号	函数名称	说明	参数	返回值	备注
44	int IMC100_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	
45	int IMC100_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	
46	int IMC100_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	
47	int IMC100_Get_DsMode(int *val, int comId=0)	查询数据流模式是否开启	val：数据流模式开启情况，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	
48	int IMC100_Get_InchMode(int *val, int comId=0)	查询示教的方式	val：示教的方式，代表查询的结果，0- 连续示教，1- 寸动示教	返回 0 表示查询成功，小于 0 表示失败	
49	int IMC100_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	
50	int IMC100_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	
51	int IMC100_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- 运动中断	返回 0 表示查询成功，小于 0 表示失败	
52	int IMC100_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	
53	int IMC100_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	
54	int IMC100_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
55	int IMC100_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
56	int IMC100_Get_CmdSts(int num, int *sts, int comId=0)	查询指定编号的运动指令实际完成状态	num：指令编号 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
57	int IMC100_Get_DINum(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	
58	int IMC100_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	
59	int IMC100_Get_ADNum(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	
60	int IMC100_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	
61	int IMC100_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	
62	int IMC100_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	

序号	函数名称	说明	参数	返回值	备注
63	int IMC100_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	
64	int IMC100_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	
65	int IMC100_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	
66	int IMC100_Get_DO(int num, int *sts, int comId=0)	按位查询 DO 的输出状态	num：DO 序号（不超过 DO 总数） sts：DO 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	
67	int IMC100_Get_DOGroup(int num, int *sts, int comId=0)	按组查询 DO 的输出状态	num：DO 组序号 sts：每组的 DO 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 DO 状态	返回 0 表示查询成功，小于 0 表示失败	
68	int IMC100_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	
69	int IMC100_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	
70	int IMC100_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 表文件系统格式错误；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	
71	int IMC100_Get_FwVersion(char ver[32], int comId=0)	查询系统控制器软件版本	ver[]：当前系统软件版本，代表查询的结果，例如 S01.12T01ES	返回 0 表示查询成功，小于 0 表示失败	
72	int IMC100_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	
73	int IMC100_Get_RobotType(char type[128], int comId=0)	查询当前系统机型	type[]：机型字符串，代表查询结果，例如 Scara_A_Ino1	返回 0 表示查询成功，小于 0 表示失败	
74	int IMC100_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	
75	int IMC100_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	
76	int IMC100_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	
77	int IMC100_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	管理模式及以上使用
78	int IMC100_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	

序号	函数名称	说明	参数	返回值	备注
79	int IMC100_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	管理模式及以上使用
80	int IMC100_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	
81	int IMC100_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	管理模式及以上使用
82	int IMC100_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	
83	int IMC100_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	管理模式及以上使用
84	int IMC100_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	
85	int IMC100_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	管理模式及以上使用
86	int IMC100_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	
87	int IMC100_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	管理模式及以上使用
88	int IMC100_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	
89	int IMC100_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	管理模式及以上使用
90	int IMC100_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	
91	int IMC100_Get_StepMotionJ(float *para, 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	
92	int IMC100_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	管理模式及以上使用
93	int IMC100_Get_StepMotionL(float *para, 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	
94	int IMC100_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	管理模式及以上使用
95	int IMC100_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	
96	int IMC100_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	管理模式及以上使用
97	int IMC100_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	
98	int IMC100_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	管理模式及以上使用

序号	函数名称	说明	参数	返回值	备注
99	int IMC100_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	
100	int IMC100_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	管理模式及以上使用
101	int IMC100_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	
102	int IMC100_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	管理模式及以上使用
103	int IMC100_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	
104	int IMC100_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	管理模式及以上使用
105	int IMC100_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	
106	int IMC100_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	管理模式及以上使用
107	int IMC100_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	
108	int IMC100_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	管理模式及以上使用
109	int IMC100_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	
110	int IMC100_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	管理模式及以上使用
111	int IMC100_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	
112	int IMC100_Set_StopDecLimJ(float para[6], int comId=0)	设置运行时关节减速度上限	para[]：最大允许关节减速度	返回 0 表示设置成功，小于 0 表示失败	管理模式及以上使用
113	int IMC100_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	
114	int IMC100_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	管理模式及以上使用
115	int IMC100_Get_ZonePara(float para[2], int comId=0)	查询过渡精度参数	para[]：线性和关节过渡精度，代表查询的结果	返回 0 表示设置成功，小于 0 表示失败	
116	int IMC100_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[]：线性和关节过渡精度	返回 0 表示设置成功，小于 0 表示失败	管理模式及以上使用
117	int IMC100_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	
118	int IMC100_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	管理模式及以上使用

序号	函数名称	说明	参数	返回值	备注
119	int IMC100_Get_AxisNLim(int axis, float *para, 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	
120	int IMC100_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	管理模式及以上使用
121	int IMC100_Get_AxisPLim(int axis, float *para, 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	
122	int IMC100_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	管理模式及以上使用
123	int IMC100_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	
124	int IMC100_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	管理模式及以上使用
125	int IMC100_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	
126	int IMC100_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	管理模式及以上使用
127	int IMC100_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	
128	int IMC100_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	管理模式及以上使用
129	int IMC100_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	
130	int IMC100_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	管理模式及以上使用
131	int IMC100_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	
132	int IMC100_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	
133	int IMC100_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	管理模式及以上使用
134	int IMC100_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	
135	int IMC100_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	管理模式及以上使用
136	int IMC100_Get_JumpPara(float *lh, float *mh, float *rh, int comId=0)	查询跳跃运动 (Jump、JumpL) 的高度参数	lh：起始位置提升高度，范围 0 至 2000，代表查询结果 mh: 运动最高点相对基坐标系零点的高度，范围 -2000 至 2000，代表查询结果 rh：到终止位置的下降高度，范围 0 至 2000，代表查询结果	返回 0 表示查询成功，小于 0 表示失败	

序号	函数名称	说明	参数	返回值	备注
137	int IMC100_Set_JumpPara(float lh, float mh, float rh, int comId=0)	设置跳跃运动(Jump、JumpL)的高度参数	lh：起始位置提升高度，范围 0 至 2000 mh: 运动最高点相对基坐标系零点的高度，范围 -2000 至 2000 rh：到终止位置的下降高度，范围 0 至 2000	返回 0 表示设置成功，小于 0 表示失败	只对数据流模式有效
138	int IMC100_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	管理模式及以上使用
139	int IMC100_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	管理模式及以上使用
140	int IMC100_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	
141	int IMC100_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	编辑模式及以上使用
142	int IMC100_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	编辑模式及以上使用
143	int IMC100_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	
144	int IMC100_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	编辑模式及以上使用
145	int IMC100_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	编辑模式及以上使用
146	int IMC100_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	
147	int IMC100_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	编辑模式及以上使用
148	int IMC100_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	
149	int IMC100_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	编辑模式及以上使用
150	int IMC100_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	
151	int IMC100_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	编辑模式及以上使用
152	int IMC100_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	
153	int IMC100_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	编辑模式及以上使用
154	int IMC100_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	
155	int IMC100_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	编辑模式及以上使用

序号	函数名称	说明	参数	返回值	备注
156	int IMC100_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	
157	int IMC100_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	编辑模式及以上使用
158	int IMC100_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	
159	int IMC100_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	编辑模式及以上使用
160	int IMC100_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	
161	int IMC100_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	编辑模式及以上使用
162	int IMC100_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	
163	int IMC100_Set_CommonVarChar(int address, char val, int comId=0)	设置公共变量区的 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	编辑模式及以上使用
164	int IMC100_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)	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	
165	int IMC100_Set_CommonVarUshort(int address, unsigned short val, int comId=0)	设置公共变量区的 unsigned short 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	编辑模式及以上使用
166	int IMC100_Get_CommonVarShort(int address, short *val, int comId=0)	查询公共变量区的 short 型数据 (2 bytes)	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	
167	int IMC100_Set_CommonVarShort(int address, short val, int comId=0)	设置公共变量区的 short 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	编辑模式及以上使用
168	int IMC100_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)	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	
169	int IMC100_Set_CommonVarUint(int address, unsigned int val, int comId=0)	设置公共变量区的 unsigned int 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	编辑模式及以上使用
170	int IMC100_Get_CommonVarInt(int address, int *val, int comId=0)	查询公共变量区的 int 型数据 (4 bytes)	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	
171	int IMC100_Set_CommonVarInt(int address, int val, int comId=0)	设置公共变量区的 int 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	编辑模式及以上使用
172	int IMC100_Get_CommonVarFloat(int address, float *val, int comId=0)	查询公共变量区的 float 型数据 (4 bytes)	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	
173	int IMC100_Set_CommonVarFloat(int address, float val, int comId=0)	设置公共变量区的 float 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	编辑模式及以上使用
174	int IMC100_Get_CommonVarDouble(int address, double *val, int comId=0)	查询公共变量区的 double 型数据 (8 bytes)	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	
175	int IMC100_Set_CommonVarDouble(int address, double val, int comId=0)	设置公共变量区的 double 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	编辑模式及以上使用

序号	函数名称	说明	参数	返回值	备注
176	int IMC100_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	adress：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	
177	int IMC100_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	adress：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	编辑模式及以上使用
178	int IMC100_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	
179	int IMC100_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	编辑模式及以上使用
180	int IMC100_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 , 范围 1-8 val：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 IMC100_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 , 范围 1-8 val：代表查询的结果	返回 0 表示设置成功，小于 0 表示失败	编辑模式及以上使用
182	int IMC100_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 float	address：modbus 区寄存器地址，范围 0-65535 sum：读取的寄存器总个数 , 范围 1-8 val：代表查询的结果 (一个 float 数据占用 2 个寄存器)	返回 0 表示查询成功，小于 0 表示失败	
183	int IMC100_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 , 范围 1-8 val：代表查询的结果	返回 0 表示设置成功，小于 0 表示失败	编辑模式及以上使用
184	int IMC100_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	
185	int IMC100_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	
186	int IMC100_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	
187	int IMC100_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	
188	int IMC100_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	
189	int IMC100_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	管理模式及以上使用
190	int IMC100_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	
191	int IMC100_Get_InCfg(int func, int *diNum, int comId=0)	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	

序号	函数名称	说明	参数	返回值	备注
192	int IMC100_Set_InCfg(int func, int diNum, int comId=0)	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	管理模式及以上使用
193	int IMC100_Get_OutCfg(int func, int *doNum, int comId=0)	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	
194	int IMC100_Set_OutCfg(int func, int doNum, int comId=0)	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	管理模式及以上使用
195	int IMC100_CurCtrlDev(int *dev, int comId=0)	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 IO，4- 远端 modbus	返回 0 表示查询成功，小于 0 表示失败	
196	int IMC100_CurPermit(int *owner, unsigned int *ipAddr, unsigned short *ipPort, int comId=0)	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	
197	int IMC100_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	
198	int IMC100_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	
199	int IMC100_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	
200	int IMC100_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	
201	int IMC100_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	
202	int IMC100_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	管理模式及以上使用
203	int IMC100_LatchEnable(int cmd, int comId=0)	锁存功能开启控制	cmd：控制命令，1- 开启，0- 关闭	返回 0 表示急停命令完成，小于 0 表示失败	
204	int IMC100_Get_LatchSts(int *sts, int comId=0)	查询锁存功能开启状态	sts：锁存功能状态，1- 开启，0- 关闭	返回 0 表示急停命令完成，小于 0 表示失败	
205	int IMC100_Get_LatchSum(int *sum, int comId=0)	查询锁存位置点的总数	sum：查询结果	返回 0 表示急停命令完成，小于 0 表示失败	
206	int IMC100_Get_LatchPos(int index, int *sts, double *pos, int comId=0)	读取对应锁存点的位置参数 (按顺序读取，每个位置只能读到一次)	index：预留参数 sts：返回状态，表示有无锁存数据（0- 无，1- 有） pos：返回锁存值，表示关节坐标	返回 0 表示急停命令完成，小于 0 表示失败	
207	int IMC100_Clr_LatchPos(int comId=0)	清除锁存位置		返回 0 表示急停命令完成，小于 0 表示失败	

附录 4：Modbus 从站地址表

地址 (10 进制)	地址 (16 进制)	变量名称	数据类型	内容	备注
主站通讯属性：比特访问 , 只读 (4096 个) 物理离散量输入 , 功能码：0x02					
0	0x0000	QW65024,bit0	位	使能状态 (0-OFF,1-ON)	
1	0x0001	QW65024,bit1	位	程序运行状态 (0- 停止 ,1- 运行)	
2	0x0002	QW65024,bit2	位	急停状态 (0-OFF,1-ON)	
3	0x0003	QW65024,bit3	位	系统故障状态 (0- 无故障 ,1- 有故障)	
4	0x0004	QW65024,bit4	位	伺服故障 0- 无故障 ,1- 有故障)	
5	0x0005	QW65024,bit5	位	系统启动完成状态 (0- 未完成 ,1- 完成)	
6	0x0006	QW65024,bit6	位	保留	
…	…	…..	位	保留	
16	0x0010	QW65025,bit0	位	P 变量读取是否成功 (0- 失败 ,1- 成功)	
17	0x0011	QW65025,bit1	位	P 变量写入是否成功 (0- 失败 ,1- 成功)	
18	0x0012	QW65025,bit2	位	直接运动到位状态 (0- 无效或未到位 ,1- 到位)	
…	…	…..	位	保留	
64	0x0040	QW65028,bit0	位	IN[000] 状态 (0-OFF,1-ON, 下同)	
65	0x0041	QW65028,bit1	位	IN[001]	
66	0x0042	QW65028,bit2	位	IN[002]	
67	0x0043	QW65028,bit3	位	IN[003]	
68	0x0044	QW65028,bit4	位	IN[004]	
69	0x0045	QW65028,bit5	位	IN[005]	
70	0x0046	QW65028,bit6	位	IN[006]	
71	0x0047	QW65028,bit7	位	IN[007]	
72	0x0048	QW65028,bit8	位	IN[008]	
73	0x0049	QW65028,bit9	位	IN[009]	
74	0x004A	QW65028,bit10	位	IN[010]	
75	0x004B	QW65028,bit11	位	IN[011]	
76	0x004C	QW65028,bit12	位	IN[012]	
77	0x004D	QW65028,bit13	位	IN[013]	
78	0x004E	QW65028,bit14	位	IN[014]	
79	0x004F	QW65028,bit15	位	IN[015]	
80	0x0050	QW65029,bit0	位	IN[016]	
81	0x0051	QW65029,bit1	位	IN[017]	
82	0x0052	QW65029,bit2	位	IN[018]	
83	0x0053	QW65029,bit3	位	IN[019]	
84	0x0054	QW65029,bit4	位	IN[020]	
85	0x0055	QW65029,bit5	位	IN[021]	
86	0x0056	QW65029,bit6	位	IN[022]	
87	0x0057	QW65029,bit7	位	IN[023]	
88	0x0058	QW65029,bit8	位	IN[024]	
89	0x0059	QW65029,bit9	位	IN[025]	
90	0x005A	QW65029,bit10	位	IN[026]	
91	0x005B	QW65029,bit11	位	IN[027]	
92	0x005C	QW65029,bit12	位	IN[028]	
93	0x005D	QW65029,bit13	位	IN[029]	
94	0x005E	QW65029,bit14	位	IN[030]	
95	0x005F	QW65029,bit15	位	IN[031]	
96	0x0060	QW65030,bit0	位	IN[032]	
97	0x0061	QW65030,bit1	位	IN[033]	
98	0x0062	QW65030,bit2	位	IN[034]	
99	0x0063	QW65030,bit3	位	IN[035]	
100	0x0064	QW65030,bit4	位	IN[036]	
101	0x0065	QW65030,bit5	位	IN[037]	
102	0x0066	QW65030,bit6	位	IN[038]	
103	0x0067	QW65030,bit7	位	IN[039]	

地址 (10 进制)	地址 (16 进制)	变量名称	数据类型	内容	备注
104	0x0068	QW65030,bit8	位	IN[040]	
105	0x0069	QW65030,bit9	位	IN[041]	
106	0x006A	QW65030,bit10	位	IN[042]	
107	0x006B	QW65030,bit11	位	IN[043]	
108	0x006C	QW65030,bit12	位	IN[044]	
109	0x006D	QW65030,bit13	位	IN[045]	
110	0x006E	QW65030,bit14	位	IN[046]	
111	0x006F	QW65030,bit15	位	IN[047]	
112	0x0070	QW65031,bit0	位	IN[048]	
113	0x0071	QW65031,bit1	位	IN[049]	
114	0x0072	QW65031,bit2	位	IN[050]	
115	0x0073	QW65031,bit3	位	IN[051]	
116	0x0074	QW65031,bit4	位	IN[052]	
117	0x0075	QW65031,bit5	位	IN[053]	
118	0x0076	QW65031,bit6	位	IN[054]	
119	0x0077	QW65031,bit7	位	IN[055]	
120	0x0078	QW65031,bit8	位	IN[056]	
121	0x0079	QW65031,bit9	位	IN[057]	
122	0x007A	QW65031,bit10	位	IN[058]	
123	0x007B	QW65031,bit11	位	IN[059]	
124	0x007C	QW65031,bit12	位	IN[060]	
125	0x007D	QW65031,bit13	位	IN[061]	
126	0x007E	QW65031,bit14	位	IN[062]	
127	0x007F	QW65031,bit15	位	IN[063]	
128	0x0080	QW65032,bit0	位	OUT[000] 状态 (0-OFF,1-ON, 下同)	
129	0x0081	QW65032,bit1	位	OUT[001]	
130	0x0082	QW65032,bit2	位	OUT[002]	
131	0x0083	QW65032,bit3	位	OUT[003]	
132	0x0084	QW65032,bit4	位	OUT[004]	
133	0x0085	QW65032,bit5	位	OUT[005]	
134	0x0086	QW65032,bit6	位	OUT[006]	
135	0x0087	QW65032,bit7	位	OUT[007]	
136	0x0088	QW65032,bit8	位	OUT[008]	
137	0x0089	QW65032,bit9	位	OUT[009]	
138	0x008A	QW65032,bit10	位	OUT[010]	
139	0x008B	QW65032,bit11	位	OUT[011]	
140	0x008C	QW65032,bit12	位	OUT[012]	
141	0x008D	QW65032,bit13	位	OUT[013]	
142	0x008E	QW65032,bit14	位	OUT[014]	
143	0x008F	QW65032,bit15	位	OUT[015]	
144	0x0090	QW65033,bit0	位	OUT[016]	
145	0x0091	QW65033,bit1	位	OUT[017]	
146	0x0092	QW65033,bit2	位	OUT[018]	
147	0x0093	QW65033,bit3	位	OUT[019]	
148	0x0094	QW65033,bit4	位	OUT[020]	
149	0x0095	QW65033,bit5	位	OUT[021]	
150	0x0096	QW65033,bit6	位	OUT[022]	
151	0x0097	QW65033,bit7	位	OUT[023]	
152	0x0098	QW65033,bit8	位	OUT[024]	
153	0x0099	QW65033,bit9	位	OUT[025]	
154	0x009A	QW65033,bit10	位	OUT[026]	
155	0x009B	QW65033,bit11	位	OUT[027]	
156	0x009C	QW65033,bit12	位	OUT[028]	
157	0x009D	QW65033,bit13	位	OUT[029]	
158	0x009E	QW65033,bit14	位	OUT[030]	
159	0x009F	QW65033,bit15	位	OUT[031]	
160	0x00A0	QW65034,bit0	位	OUT[032]	

地址 (10 进制)	地址 (16 进制)	变量名称	数据类型	内容	备注
161	0x00A1	QW65034,bit1	位	OUT[033]	
162	0x00A2	QW65034,bit2	位	OUT[034]	
163	0x00A3	QW65034,bit3	位	OUT[035]	
164	0x00A4	QW65034,bit4	位	OUT[036]	
165	0x00A5	QW65034,bit5	位	OUT[037]	
166	0x00A6	QW65034,bit6	位	OUT[038]	
167	0x00A7	QW65034,bit7	位	OUT[039]	
168	0x00A8	QW65034,bit8	位	OUT[040]	
169	0x00A9	QW65034,bit9	位	OUT[041]	
170	0x00AA	QW65034,bit10	位	OUT[042]	
171	0x00AB	QW65034,bit11	位	OUT[043]	
172	0x00AC	QW65034,bit12	位	OUT[044]	
173	0x00AD	QW65034,bit13	位	OUT[045]	
174	0x00AE	QW65034,bit14	位	OUT[046]	
175	0x00AF	QW65034,bit15	位	OUT[047]	
176	0x00B0	QW65035,bit0	位	OUT[048]	
177	0x00B1	QW65035,bit1	位	OUT[049]	
178	0x00B2	QW65035,bit2	位	OUT[050]	
179	0x00B3	QW65035,bit3	位	OUT[051]	
180	0x00B4	QW65035,bit4	位	OUT[052]	
181	0x00B5	QW65035,bit5	位	OUT[053]	
182	0x00B6	QW65035,bit6	位	OUT[054]	
183	0x00B7	QW65035,bit7	位	OUT[055]	
184	0x00B8	QW65035,bit8	位	OUT[056]	
185	0x00B9	QW65035,bit9	位	OUT[057]	
186	0x00BA	QW65035,bit10	位	OUT[058]	
187	0x00BB	QW65035,bit11	位	OUT[059]	
188	0x00BC	QW65035,bit12	位	OUT[060]	
189	0x00BD	QW65035,bit13	位	OUT[061]	
190	0x00BE	QW65035,bit14	位	OUT[062]	
191	0x00BF	QW65035,bit15	位	OUT[063]	
192	0x00C0	QW65036,bit0	位	J1 伺服告警 (0- 无告警 ,1- 有告警 , 下同)	
193	0x00C1	QW65036,bit1	位	J2 伺服告警	
194	0x00C2	QW65036,bit2	位	J3 伺服告警	
195	0x00C3	QW65036,bit3	位	J4 伺服告警	
196	0x00C4	QW65036,bit4	位	J5 伺服告警	
197	0x00C5	QW65036,bit5	位	J6 伺服告警	
198	0x00C6	QW65036,bit6	位	J7 伺服告警	
199	0x00C7	QW65036,bit7	位	J8 伺服告警	
…	…	…..	位	-	
4095	0x0fff	QW65279,bit15	位	-	
主站通讯属性：比特访问，读写 (4096 个) 线圈功能码：0x01,0x05,0x0f					
4096	0x1000	QW65280,bit0	位	程序启动运行	（复归型）
4097	0x1001	QW65280,bit1	位	程序暂停	（复归型）
4098	0x1002	QW65280,bit2	位	程序停止	（复归型）
4099	0x1003	QW65280,bit3	位	使能开关 (0-OFF,1-ON)	
4100	0x1004	QW65280,bit4	位	急停开关 (0-OFF,1-ON)	
4101	0x1005	QW65280,bit5	位	故障复位	（复归型）
…	…	……	位	保留	
4112	0x1010	QW65281,bit0	位	示教 J1/X +	（复归型）
4113	0x1011	QW65281,bit1	位	示教 J1/Y +	（复归型）
4114	0x1012	QW65281,bit2	位	示教 J3/Z +	（复归型）
4115	0x1013	QW65281,bit3	位	示教 J4/A +	（复归型）
4116	0x1014	QW65281,bit4	位	示教 J5 +	（复归型）
4117	0x1015	QW65281,bit5	位	示教 J6 +	（复归型）
4118	0x1016	QW65281,bit6	位	保留	
4119	0x1017	QW65281,bit7	位	保留	

地址 (10 进制)	地址 (16 进制)	变量名称	数据类型	内容	备注
4120	0x1018	QW65281,bit8	位	示教 J1/X -	（复归型）
4121	0x1019	QW65281,bit9	位	示教 J2/Y -	（复归型）
4122	0x101a	QW65281,bit10	位	示教 J3/Z -	（复归型）
4123	0x101b	QW65281,bit11	位	示教 J4/A -	（复归型）
4124	0x101c	QW65281,bit12	位	示教 J5 -	（复归型）
4125	0x101d	QW65281,bit13	位	示教 J6 -	（复归型）
4126	0x101e	QW65281,bit14	位	保留	
4127	0x101f	QW65281,bit15	位	保留	
…	…	……	位	保留	
4134	0x1026	QW65282,bit6	位	工位程序 1	
4135	0x1027	QW65282,bit7	位	工位程序 2	
4136	0x1028	QW65282,bit8	位	工位程序 3	
4137	0x1029	QW65282,bit9	位	保留	
4138	0x102a	QW65282,bit10	位	保留	
4139	0x102b	QW65282,bit11	位	把机器人当前位置写入当前 P 变量	（复归型）
4140	0x102c	QW65282,bit12	位	把修改位置（MW34832 等）写入当前 P 变量	（复归型）
4141	0x102d	QW65282,bit13	位	直接运动到当前 P 变量位置	（复归型）
4142	0x102e	……	位	保留	
4143	0x102f	……	位	保留	
4144	0x1030	QW65283,bit0	位	OUT[000] 控制命令（0-OFF,1-ON, 下同）	
4145	0x1031	QW65283,bit1	位	OUT[001]	
4146	0x1032	QW65283,bit2	位	OUT[002]	
4147	0x1033	QW65283,bit3	位	OUT[003]	
4148	0x1034	QW65283,bit4	位	OUT[004]	
4149	0x1035	QW65283,bit5	位	OUT[005]	
4150	0x1036	QW65283,bit6	位	OUT[006]	
4151	0x1037	QW65283,bit7	位	OUT[007]	
4152	0x1038	QW65283,bit8	位	OUT[008]	
4153	0x1039	QW65283,bit9	位	OUT[009]	
4154	0x103a	QW65283,bit10	位	OUT[010]	
4155	0x103b	QW65283,bit11	位	OUT[011]	
4156	0x103c	QW65283,bit12	位	OUT[012]	
4157	0x103d	QW65283,bit13	位	OUT[013]	
4158	0x103e	QW65283,bit14	位	OUT[014]	
4159	0x103f	QW65283,bit15	位	OUT[015]	
4160	0x1040	QW65284,bit0	位	OUT[016]	
4161	0x1041	QW65284,bit1	位	OUT[017]	
4162	0x1042	QW65284,bit2	位	OUT[018]	
4163	0x1043	QW65284,bit3	位	OUT[019]	
4164	0x1044	QW65284,bit4	位	OUT[020]	
4165	0x1045	QW65284,bit5	位	OUT[021]	
4166	0x1046	QW65284,bit6	位	OUT[022]	
4167	0x1047	QW65284,bit7	位	OUT[023]	
4168	0x1048	QW65284,bit8	位	OUT[024]	
4169	0x1049	QW65284,bit9	位	OUT[025]	
4170	0x104a	QW65284,bit10	位	OUT[026]	
4171	0x104b	QW65284,bit11	位	OUT[027]	
4172	0x104c	QW65284,bit12	位	OUT[028]	
4173	0x104d	QW65284,bit13	位	OUT[029]	
4174	0x104e	QW65284,bit14	位	OUT[030]	
4175	0x104f	QW65284,bit15	位	OUT[031]	
4176	0x1050	QW65285,bit0	位	OUT[032]	
4177	0x1051	QW65285,bit1	位	OUT[033]	
4178	0x1052	QW65285,bit2	位	OUT[034]	
4179	0x1053	QW65285,bit3	位	OUT[035]	
4180	0x1054	QW65285,bit4	位	OUT[036]	
4181	0x1055	QW65285,bit5	位	OUT[037]	

地址 (10 进制)	地址 (16 进制)	变量名称	数据类型	内容	备注
4182	0x1056	QW65285,bit6	位	OUT[038]	
4183	0x1057	QW65285,bit7	位	OUT[039]	
4184	0x1058	QW65285,bit8	位	OUT[040]	
4185	0x1059	QW65285,bit9	位	OUT[041]	
4186	0x105a	QW65285,bit10	位	OUT[042]	
4187	0x105b	QW65285,bit11	位	OUT[043]	
4188	0x105c	QW65285,bit12	位	OUT[044]	
4189	0x105d	QW65285,bit13	位	OUT[045]	
4190	0x105e	QW65285,bit14	位	OUT[046]	
4191	0x105f	QW65285,bit15	位	OUT[047]	
4192	0x1060	QW65286,bit0	位	OUT[048]	
4193	0x1061	QW65286,bit1	位	OUT[049]	
4194	0x1062	QW65286,bit2	位	OUT[050]	
4195	0x1063	QW65286,bit3	位	OUT[051]	
4196	0x1064	QW65286,bit4	位	OUT[052]	
4197	0x1065	QW65286,bit5	位	OUT[053]	
4198	0x1066	QW65286,bit6	位	OUT[054]	
4199	0x1067	QW65286,bit7	位	OUT[055]	
4200	0x1068	QW65286,bit8	位	OUT[056]	
4201	0x1069	QW65286,bit9	位	OUT[057]	
4202	0x106a	QW65286,bit10	位	OUT[058]	
4203	0x106b	QW65286,bit11	位	OUT[059]	
4204	0x106c	QW65286,bit12	位	OUT[060]	
4205	0x106d	QW65286,bit13	位	OUT[061]	
4206	0x106e	QW65286,bit14	位	OUT[062]	
4207	0x106f	QW65286,bit15	位	OUT[063]	
…		……	位	保留	
8191	0x1fff	QW65535,bit15	位	-	
主站通讯属性：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)	
…		……	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	当前模式（1- 示教，2- 再现）	
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	单精度浮点	J6/C 坐标低位	
2063	0x080f	MW2063		J6/C 坐标高位	
…	…	……	字	保留	
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	

地址 (10 进制)	地址 (16 进制)	变量名称	数据类型	内容	备注
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	
…		……	字	保留	
2145	0x0861	MW2145	字 (无符号)	当前工位程序	
2146	0x0862	MW2146	字 (无符号)	保留	
2147	0x0863	MW2147	字 (无符号)	保留	
2148	0x0864	MW2148	单精度浮点	P 变量读取的 J1/X 坐标低位	
2149	0x0865	MW2149		P 变量读取的 J1/X 坐标高位	
2150	0x0866	MW2150	单精度浮点	P 变量读取的 J2/Y 坐标低位	
2151	0x0867	MW2151		P 变量读取的 J2/Y 坐标高位	
2152	0x0868	MW2152	单精度浮点	P 变量读取的 J3/Z 坐标低位	
2153	0x0869	MW2153		P 变量读取的 J3/Z 坐标高位	
2154	0x086A	MW2154	单精度浮点	P 变量读取的 J4/A 坐标低位	
2155	0x086B	MW2155		P 变量读取的 J4/A 坐标高位	
2156	0x086C	MW2156	单精度浮点	P 变量读取的 J5/B 坐标低位	
2157	0x086D	MW2157		P 变量读取的 J5/B 坐标高位	
2158	0x086E	MW2158	单精度浮点	P 变量读取的 J6/C 坐标低位	
2159	0x086F	MW2159		P 变量读取的 J6/C 坐标高位	
2160	0x0870	MW2160	字 (有符号)	P 变量读取的臂参数 1	
2161	0x0871	MW2161	字 (有符号)	P 变量读取的臂参数 2	
2162	0x0872	MW2162	字 (有符号)	P 变量读取的臂参数 3	
2163	0x0873	MW2163	字 (有符号)	P 变量读取的臂参数 4	
2164	0x0874	MW2164	字 (无符号)	P 变量读取的坐标系	
2165	0x0875	MW2165	字 (无符号)	P 变量读取的工具坐标系号	
2166	0x0876	MW2166	字 (无符号)	P 变量读取的用户坐标系号	
…		……	字	-	
32767	0x7fff	MW32767	字	-	
主站通讯属性：16 比特访问，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	
32769	0x8001	MW32769	字		
…		……	字		
34800	0x87F0	MW34800	字	示教方式选择 (0- 连续示教 ,1- 寸动示教)	
34801	0x87F1	MW34801	单精度浮点	寸动关节步长（单位度，关节坐标下有效）	
34802	0x87F2	MW34802			
34803	0x87F3	MW34803	单精度浮点	寸动线性步长（单位毫米，直角坐标下有效）	
34804	0x87F4	MW34804			
34805	0x87F5	MW34805	字 (无符号)	直接运动方式设置 (0-MovJ,1-MovL)	
…		……..	字	保留	
34816	0x8800	MW34816	字	坐标系选择 (1- 关节 ,2- 直角 ,3- 工具 ,4- 用户)	
34817	0x8801	MW34817	字	速度设置（1-100）	

地址 (10 进制)	地址 (16 进制)	变量名称	数据类型	内容	备注
…		…….	字	保留	
34829	0x880D	MW34829	字 (无符号)	保留	
34830	0x880E	MW34830	字 (无符号)	保留	
34831	0x880F	MW34831	字 (无符号)	当前 P 变量的序号（等待读、写、运动操作）	
34832	0x8810	MW34832	单精度浮点	P 变量写入的 J1/X 坐标低位	
34833	0x8811	MW34833		P 变量写入的 J1/X 坐标高位	
34834	0x8812	MW34834	单精度浮点	P 变量写入的 J2/Y 坐标低位	
34835	0x8813	MW34835		P 变量写入的 J2/Y 坐标高位	
34836	0x8814	MW34836	单精度浮点	P 变量写入的 J3/Z 坐标低位	
34837	0x8815	MW34837		P 变量写入的 J3/Z 坐标高位	
34838	0x8816	MW34838	单精度浮点	P 变量写入的 J4/A 坐标低位	
34839	0x8817	MW34839		P 变量写入的 J4/A 坐标高位	
34840	0x8818	MW34840	单精度浮点	P 变量写入的 J5/B 坐标低位	
34841	0x8819	MW34841		P 变量写入的 J5/B 坐标高位	
34842	0x881A	MW34842	单精度浮点	P 变量写入的 J6/C 坐标低位	
34843	0x881B	MW34843		P 变量写入的 J6/C 坐标高位	
34844	0x881C	MW34844	字 (有符号)	P 变量写入的臂参数 1	
34845	0x881D	MW34845	字 (有符号)	P 变量写入的臂参数 2	
34846	0x881E	MW34846	字 (有符号)	P 变量写入的臂参数 3	
34847	0x881F	MW34847	字 (有符号)	P 变量写入的臂参数 4	
34848	0x8820	MW34848	字 (无符号)	P 变量写入的坐标系	
34849	0x8821	MW34849	字 (无符号)	P 变量写入的工具坐标系号	
34850	0x8822	MW34850	字 (无符号)	P 变量写入的用户坐标系号	
…		…….	字	-	
65535	0xffff	MW65535	字	-	

创变·精彩

官方微信

服务与技术支持APP

销售服务联络地址

深圳市汇川技术股份有限公司
Shenzhen Inovance Technology Co., Ltd.
地址：深圳市宝安区宝城70区留仙二路鸿威工业区E栋
总机：(0755)2979 9595
传真：(0755)2961 9897
客服：400-777-1260
http://www.inovance.com

苏州汇川技术有限公司
Suzhou Inovance Technology Co., Ltd.
地址：苏州市吴中区越溪友翔路16号
总机：(0512)6637 6666
传真：(0512)6285 6720
客服：400-777-1260
http://www.inovance.com