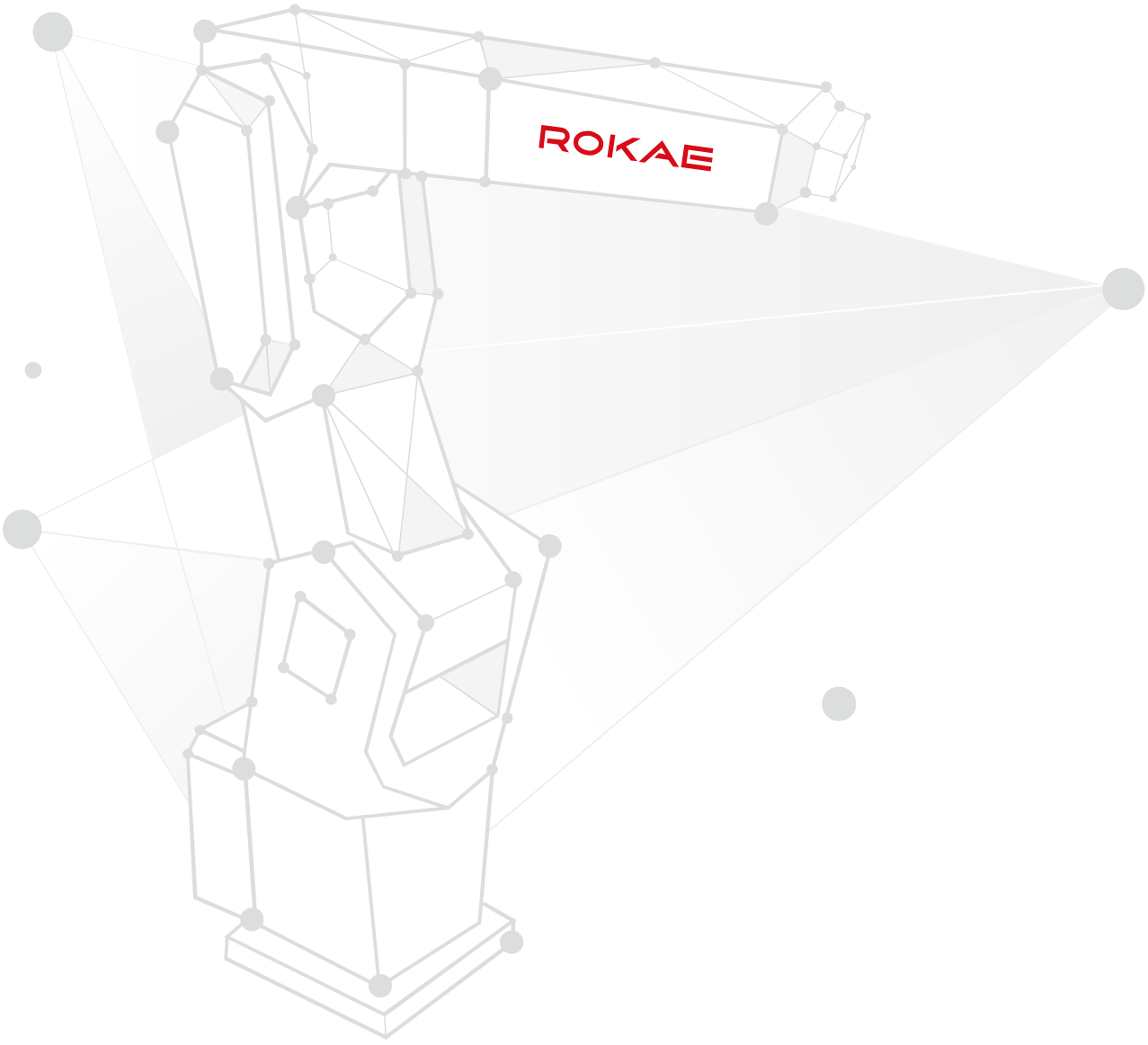
ROKAE 珞石
轻 型 机 器 人 专 家

XB12系列

机器人控制系统操作手册



- 用机器人技术让生活更美好 -

XB12 系列

机器人控制系统操作手册

文档编号：20110000004

文档版本：A

本手册中记载的内容如有变更，恕不事先通告。本公司对手册中可能出现的错误均不承担任何责任。

本公司对因使用本手册及其中所述产品而引起的意外或间接伤害均不承担任何责任，敬请谅解。

本公司不可能预见所有的危险和后果，因此本手册不能警告用户所有可能的危险。

如您发现本手册的内容有误或需要改进抑或补充之处，请不吝指正。

本手册的原始语言为中文，所有其他语言版本均翻译自中文版本。

珞石（山东）智能科技有限公司

中国.山东

目录

1 手册概述 1
1.1 关于本手册 1
1.2 手册对象 1
1.3 如何阅读产品手册 1
1.4 本手册中的插图 1
1.5 垂询方式 1
2 安全 3
2.1 简介 3
2.1.1 安全责任说明 3
2.1.2 按规定使用机器人 3
2.2 安全术语 3
2.2.1 安全标识 3
2.2.2 风险说明 4
2.2.3 安全特性 5
2.2.4 运动使能与安全停止 6
2.2.5 安全装置 6
2.3 工作中的安全事项 7
2.3.1 概述 7
2.3.2 关注自身安全 7
2.3.3 操作示教器的安全事项 8
2.3.4 使用控制柜的安全事项 9
2.3.5 急停按钮测试 9
2.3.6 从紧急停止状态恢复 9
2.3.7 抱闸测试 10
2.3.8 手动释放抱闸 10
2.3.9 手动模式的安全事项 10
2.3.10 自动模式的安全事项 11
2.3.11 生产线上的安全处理 11
2.3.12 火灾事故的安全处理 11
2.3.13 触电事故的安全处理 12
2.4 作业人员及内容要求 12
2.4.1 作业人员定义 12
2.4.2 作业人员要求 13
2.4.3 作业内容要求 13

2.5 安全培训 14
2.5.1 概述 14
2.5.2 作业人员的安全 14
3 使用 Titanite 17
3.1 关于本章 17
3.2 控制柜 18
3.2.1 关于控制柜 18
3.2.2 控制柜布局 19
3.2.3 按钮与接口 19
3.2.4 IO 及网口定义 21
3.3 示教器 25
3.3.1 关于示教器 25
3.3.2 按钮与接口 26
3.3.3 如何握持示教器 27
3.4 启动系统 27
3.5 维护 31
3.5.1 关于维护时的安全 31
3.5.2 维修计划 31
3.5.3 检修间隔 31
4 操作界面 33
4.1 界面布局 33
4.1.1 状态栏 33
4.1.2 快捷设置 34
4.1.3 主要页面 34
4.1.4 界面切换 40
4.2 基本设置 41
4.2.1 用户登录 41
4.2.2 设置软限位 41
4.2.3 系统升级 42
4.2.4 备份恢复 43
4.2.5 恢复出厂设置 45
4.2.6 姿态调整 46
5 Jog 操作 49
5.1 什么是 Jog? 49
5.2 机器人坐标系统 50
5.3 Jog 时的注意事项 52

5.4 开始 Jog....53
5.4.1 选择 Jog 参数....53
5.4.2 关节空间 Jog....54
5.4.3 笛卡尔空间 Jog....55
5.4.4 查看位置与 IO....55
6 编程和调试....59
6.1 关于本章....59
6.2 工作模式....60
6.2.1 手动模式....60
6.2.2 自动模式....60
6.2.3 模式切换....62
6.3 工具....65
6.3.1 什么是工具？....65
6.3.2 什么是工具中心点？....65
6.3.3 定义工具....65
6.3.4 标定工具坐标系....66
6.3.5 定义工具负载....67
6.3.6 使用工具....67
6.4 工件....67
6.4.1 什么是工件？....67
6.4.2 定义工件....68
6.4.3 标定工件坐标系....68
6.4.4 使用工件....69
6.5 基坐标系....70
6.6 机械零点....70
6.6.1 关于机械零点....70
6.6.2 机械标定....70
6.6.3 软标定....73
6.7 配置 IO....74
6.8 可编程按键....76
6.9 开始编程....78
6.9.1 管理程序文件....78
6.9.2 创建一个工程....79
6.9.3 编写程序....80
6.9.4 视觉....83
6.10 程序调试....90

6.10.1 调试功能菜单 90
6.10.2 运行指针 90
6.10.3 前瞻机制 92
6.10.4 移动程序指针 92
6.10.5 单步运行 93
6.10.6 重回路径 93
6.10.7 变量管理 94
6.10.8 机器人的奇异性 95
7 RL 编程语言 97
7.1 关于 RL 语言 97
7.2 程序结构 99
7.2.1 概述 99
7.2.2 程序模块 99
7.2.3 函数 100
7.3 变量 102
7.3.1 基本概念 102
7.3.2 变量声明 104
7.3.3 变量类型 105
7.4 运算符 126
7.4.1 优先级 128
7.5 指令 129
7.5.1 AccSet 129
7.5.2 ActUnit 130
7.5.3 BitAnd 130
7.5.4 BitCheck 131
7.5.5 BitClear 131
7.5.6 BitLSh 132
7.5.7 BitNeg 132
7.5.8 BitOr 133
7.5.9 BitRSh 134
7.5.10 BitSet 134
7.5.11 BitXOr 135
7.5.12 Break 135
7.5.13 ByteToStr 136
7.5.14 CalcJointT 137
7.5.15 CalcRobT 137

7.5.16 ChkNet 138
7.5.17 CJointT 138
7.5.18 ClkRead 139
7.5.19 ClkReset 139
7.5.20 ClkStart 140
7.5.21 ClkStop 140
7.5.22 Continue 141
7.5.23 CRobT 141
7.5.24 DeactUnit 142
7.5.25 DecToHex 142
7.5.26 DoubleToByte 142
7.5.27 DoubleToRobtarget 143
7.5.28 DoubleToSpeed 143
7.5.29 DoubleToStr 144
7.5.30 EulerToQuaternion 144
7.5.31 EXIT 145
7.5.32 HexToDec 145
7.5.33 HomeSet 145
7.5.34 HomeSetAt 146
7.5.35 Hordr 146
7.5.36 HordrAt 146
7.5.37 HomeDef 147
7.5.38 HomeClr 147
7.5.39 Home 147
7.5.40 IF…ELSE… 147
7.5.41 IntToByte 148
7.5.42 IntToStr 149
7.5.43 MemIn 149
7.5.44 MemSw 149
7.5.45 MemOff 150
7.5.46 MemOn 150
7.5.47 MemOut 150
7.5.48 MotionSup 151
7.5.49 MoveAbsJ 152
7.5.50 MoveJ 153
7.5.51 MoveL 154

7.5.52 Offs 156
7.5.53 Pause 156
7.5.54 Print 157
7.5.55 PulseDO 157
7.5.56 QuaternionToEuler 158
7.5.57 RelTool 158
7.5.58 RETURN 159
7.5.59 SearchL 159
7.5.60 SetDO 161
7.5.61 SetGO 161
7.5.62 SocketCreate 162
7.5.63 SocketClose 162
7.5.64 SocketSendByte 163
7.5.65 SocketSendString 163
7.5.66 SocketReadBit 164
7.5.67 SocketReadByte 165
7.5.68 SocketReadDouble 165
7.5.69 SocketReadInt 166
7.5.70 SocketReadString 166
7.5.71 StrFind 167
7.5.72 StrLen 168
7.5.73 StrMap 168
7.5.74 StrMatch 169
7.5.75 StrMemb 169
7.5.76 StrOrder 170
7.5.77 StrPart 171
7.5.78 StrSplit 171
7.5.79 StrToByte 172
7.5.80 StrToDouble 172
7.5.81 TestAndSet 173
7.5.82 WHILE 174
7.5.83 Wait 174
7.5.84 WaitSyncTask 174
7.5.85 WaitUntil 176
7.5.86 WaitWobj 176
7.5.87 XYLim 176
7.5.88 XYLimClr 177

7.5.89 XYLimDef 177
7.5.90 数学指令 177
7.6 起始点 180
7.6.1 概述 180
7.6.2 参数配置 180
7.6.3 系统输出 181
7.6.4 姿态调整 181
7.7 外部通信 181
8 可选功能 187
8.1 功能授权 187
8.2 多任务 187
8.2.1 概述 187
8.2.2 新建任务 187
8.2.3 任务看板 188
8.2.4 启动和运行任务 189
8.2.5 任务间通信 190
9 故障排查 191
9.1 按故障现象 192
修订记录 193

1 手册概述

1.1 关于本手册

感谢您购买本公司的机器人系统。

本手册记载了如何正确操作机器人控制系统。

安装使用该机器人系统前，请仔细阅读本手册与其他相关手册。

阅读之后，请妥善保管，以便随时取阅。

1.2 手册对象

本手册面向：

- 机器人编程调试人员。

请务必保证以上人员具备控制系统操作所需的知识，并已接受本公司的相关培训。

1.3 如何阅读产品手册

本手册包含单独的安全章节，必须在阅读安全章节后，才能进行安装或维护作业。

1.4 本手册中的插图

由于产品升级或其他原因，产品手册中的一些图片可能会与实际产品存在差异，但操作步骤是正确的。

同时，对于某些通用的信息，可能会使用其他型号机器人的图片进行说明。

1.5 垂询方式

机器人维护、维修等相关事项，请与本公司售后部门或当地经销商联系。

联系时，请准备好如下信息：

- 控制器型号/序列号
- 机器人型号/序列号
- 软件名称/版本
- 系统出现的问题

2 安全

2.1 简介

本章介绍在使用机器人时需要注意的安全原则和流程。

与机器人外部安全防护装置的设计、安装有关的内容不在本章描述范围之内，请与您的系统集成商联系以获得此类信息。

2.1.1 安全责任说明

珞石机器人致力于提供可靠的安全信息，但不对此承担责任。即使一切操作都按照安全操作说明进行，也不能确保工业机器人不会造成人身和财产方面的损失。

除安全章节外，请注意在文档的必要部分有其他的安全提示。

2.1.2 按规定使用机器人

工业机器人的使用应符合当地的法律法规，不允许违规使用在违背法律法规的用途上。

按规定使用机器人还包括遵守各单个部件的产品手册说明，包括对其描述的操作、安装、维护说明等内容。

以下违规的使用应被禁止：

- 运输人员和动物
- 在有爆炸危险的环境中使用
- 在可燃性环境中使用
- 在允许的范围之外使用
- 在井下使用

2.2 安全术语

2.2.1 安全标识

2.2.1.1 关于安全标识

按照本手册内容操作机器人时可能会遇到不同程度的危险状况，因此在可能会造成危险的操作说明附近会有专门的安全标识提示框，重点提示用户注意防范，提示框中的内容包括：

- 一个表示安全级别的图标和对应的名称，例如警告、危险、提示等。
- 一段简单的描述，用于说明如果操作人员不消除该危险可能会造成的后果。
- 有关如何消除危险的操作说明。

2.2.1.2 安全级别

图标	名称	说明
	危险	带有该标识的内容如果没有按照规定操作，将会对人员造成严重甚至致命的伤害，同时将会/可能会对机器人造成严重损坏。与此类危险有关的操作包括接触控制柜内高压器件、在机器人运行时进入其工作区域等。

	警告	带有该标识的内容如果没有按照规定操作，可能会导致严重人身伤害，甚至可能致命，对机器人本身也将造成较大损坏。
	警示	带有该标识的内容如果没有按照规定操作，可能会导致人身伤害，对机器人本身可能也会造成损坏。
	提示	用于提示一些重要信息或者前提条件。

2.2.2 风险说明

2.2.2.1 风险描述

图标	名称	说明
	挤压	操作人员、维护人员在调试、维修、检修、安装工具时进入机器人运动范围，可能会产生伤害。
	夹手	维护人员在进行维护操作时，接近带传动部件或其他运动部件时，存在夹手的风险。
	撞击	操作人员、维护人员在调试、维修、检修、安装工具时进入机器人运动范围，可能会产生严重伤害。
	摩擦	操作人员、维护人员在调试、维修、检修、安装工具时进入机器人运动范围，可能会产生伤害。
	零件飞出	操作人员、维护人员在调试、维修、检修、安装工具时进入机器人运动范围，工具或工件可能因夹持松懈飞出，此时可能会产生严重伤害。
	火灾	电路发生短路、导线或器件着火时可能发生火灾，可能会产生严重伤害。
	高温表面	维护人员在进行设备检修、维护时，接触机器人高温表面，可能会导致烫伤。
	触电危险	提示当前操作可能会有人员触电风险，造成严重甚至是致命的伤害。
	防静电（ESD）	提示当前操作涉及的零部件对静电敏感，不按规范操作可能会造成器件损坏。

警告

任何正在运动中的机器人都是潜在的致命机械!

机器人在运行时，可能会执行与期望不符甚至是不合理的运动。此外，机器人在运动时会携带巨大的能量，当发生碰撞时，会对其工作范围内的人员和设备造成严重伤害/损害。

2.2.2.2 消除危险

	操作	参考信息
1	在开始运行机器人之前，确保已经正确配置和安装所有的安全设备。	安全设备包括急停按钮、安全门、安全光栅等。
2	机器人编程过程中，必须保证由进入机器人工作区域的人员持有示教器。	工作区域之外的人员须避免在没有观察到工作区域内人员的情况下使用示教器操作机器人。
3	在开始运行机器人程序之前，确保机器人工作区域中没有人员存在。	
4	对机器人进行运动编程时，请务必在第一次测试运行之前找到潜在的碰撞风险。	

2.2.3 安全特性

2.2.3.1 说明

本机器人系统配备了专门的安全控制器用来处理安全相关信号，并提供了安全门、急停按钮等外部安全信号接口。

由安全控制器处理或输出的信号包括：

- 急停按钮信号。
- 安全门信号。
- 使能开关信号。
- 模式选择信号。
- 急停状态信号。

2.2.3.2 适用安全标准

机器人系统的设计符合以下相关标准：

标准	描述
2006/42/EC	机械指令
2014/30/EU	电磁兼容指令
EN ISO 12100:2010	机械安全 设计通则 风险评估与风险减小
EN ISO 10218-1:2011	工业机器人 安全要求 第 1 部分：机器人
EN ISO 13849-1:2015	机械安全 控制系统安全相关部件 第 1 部分：设计通则
IEC 60204-1:2016	机械电气安全 机械电气设备 第 1 部分：通用技术条件
IEC 61508-2:2010	电气/电子/可编程电子安全相关系统的功能安全 第 2 部分：电气/电子/可编程电子安全相关系统的要求
IEC 62061:2010	机械电气安全 安全相关电气/电子/可编程电子控制系统的功能安全

IEC 61000-6-2:2016	电磁兼容 通用标准 工业环境中的抗扰度
IEC 61000-6-4:2011	电磁兼容 通用标准 工业环境中的发射

2.2.4 运动使能与安全停止

2.2.4.1 运动使能

机器人控制系统的运动控制功能应由安全控制器使能，安全控制器通过内部逻辑判断当前使用环境安全时，通过安全输出信号控制驱动器 STO（安全转矩关断）的通断。控制系统只能在安全控制器判断此时刻为安全时，才允许用户手动操作机器人或自动运行程序。

2.2.4.2 安全停止

机器人的停止方式可分为三种，STOP 0、STOP 1 和 STOP 2。

安全停止是指在由安全控制器触发的停止，安全控制器只触发 STOP 0、STOP 1 两种停止方式，STOP 2 只由控制系统负责。

- STOP 0 停止

STOP 0 停止被触发后，立即切断电机的动力电源并闭合各关节抱闸，是安全级别最高的停止方式，但停止过程中机器人处于非受控状态，停止后可能会偏离编程路径。

手动模式的安全停止属于 STOP 0，自动模式下切换操作模式而引起的安全停止也属于 STOP 0。

- STOP 1 停止

STOP 1 停止被触发后，控制系统立刻沿编程路径执行减速过程，之后不论机器人是否完全停下，安全控制器将切断电机的动力电源并闭合各关节抱闸。在绝大多数情况下，由于是受控停止，机器人最终将停在编程路径上，因此该种急停方式对周边设备的保护性最好。

自动模式下安全门/安全光栅打开、自动模式下急停按钮被按下而发生的安全停止，均属于 STOP 1。

- STOP 2 停止

STOP 2 停止被触发后，控制系统立刻沿编程路径执行减速，直到机器人完全停止运动。此时电机的动力电源仍然保持，抱闸仍然打开，机器人保持在当前位置上。

2.2.4.3 紧急停止

紧急停止属于安全停止的一种，是机器人系统中优先级最高的功能。按下急停按钮将触发紧急停止功能，此时所有其他的机器人控制功能将停止，机器人停止运动且各关节电机的动力电将被切断，控制系统切换至紧急停止状态，在被复位之前将一直保持该状态。

紧急停止状态意味着除手动抱闸释放电路外，其他所有通往机器人本体的电源将被切断，必须执行复位操作才能将系统恢复到正常状态。

> 提示
>
> 紧急停止仅用于在危险情况下立刻停止机器人运行，不能将紧急停止作为正常的程序停止，否则将对机器人的抱闸系统和传动系统造成额外而不必要的磨损，降低机器人的使用寿命。

2.2.5 安全装置

2.2.5.1 急停按钮

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

还配合使用黄色的衬底、保护外壳或警示牌。按下急停时，按钮靠机械锁定，这是急停按钮的安全锁机制，此时必须通过手动释放来复位装置。大多数急停按钮都采用旋转释放方式，旋转方向会标在按钮的表面，也有一部分按钮支持直接向上拔起的释放方法。

2.2.5.2 使能开关

使能装置（Enabling Device）是一个具有 2 段按压 3 个位置的特殊开关，又称三位使能开关（以下简称使能开关），用于在手动模式下控制机器人动力电源的通断，由此来实现机器人的运动使能。

使能开关安装在示教器的背面，如图 1 所示，只有按下使能开关并保持在中间位置时才会接通电机电源，使机器人处于允许运动的状态，可以进行 Jog 或者运行程序。松手放开或者用力按压到底都将切断电机动力电电源。

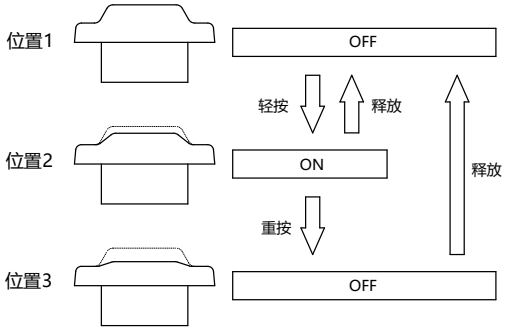


图 1 使能开关

> **警告**
>
> 严禁使用任何外部装置将使能开关卡住使其停留在中间位置！

> **提示**
>
> 在任何情况下都必须保证使能开关可以正常工作。
>
> 在编程和调试期间，当不需要机器人运动时应尽快松开使能开关。

2.3 工作中的安全事项

2.3.1 概述

2.3.1.1 关于机器人

不论运动速度如何，工业机器人都具有很大的潜在危险性。程序运行中的一个暂停或者等待之后可能接着一个快速、危险的运动指令。即便您已经熟悉当前机器人的运动轨迹和模式，但是在自动模式下机器人的运动轨迹仍然可能会被外部信号在毫无预警的情况下改变。

因此，在进入机器人的工作范围时必须要遵守安全规范。

2.3.1.2 关于本节

本节将介绍一些面向机器人最终用户的基本安全规范。但是限于篇幅，无法覆盖每一种特定的情形。

2.3.2 关注自身安全

2.3.2.1 基本原则

为了保证使用机器人时的安全，必须严格遵守以下几条原则：

- 当有工作人员处于机器人的安全防护区域内时，只能使用手动模式操作机器人。
- 当您进入机器人的安全防护区域时，必须将示教器拿在手上，以确保机器人在您的控制之下。
- 留意安装在机器人上的可活动的工具，例如电钻、电锯等。在靠近机器人之前，要确保这些工具已经停止运行。
- 留意工件表面或者机器人本体，在长时间工作后，机器人的电机和外壳温度可能会非常高。
- 留意机器人抓手及所抓持的物品。如果抓手打开，工件有可能会掉落造成人员受伤或者设备损坏。此外机器人使用的抓手可能非常强力，如果不按规范使用也可能会造成伤害。
- 留意机器人控制柜内的电力部件。即使已经断电，器件内留存的能量仍然非常危险。

2.3.3 操作示教器的安全事项

2.3.3.1 安全存放示教器

当不使用示教器时，应将从控制柜上取下来的示教器妥善存放在远离机器人工作站或者控制柜的地方，不能让操作人员误认为这个示教器仍然连接在控制柜上，以免出现危险时试图使用没有连接到控制柜上的示教器来停止机器人。

2.3.3.2 示教器电缆

示教器与控制柜之间通过示教器电缆进行连接，在使用示教器时，为了避免发生人员伤害或设备损坏，请遵守以下要求：

- 确保工作人员不会绊到示教器电缆，并导致示教器跌落或人员摔倒。
- 不要挤压示教器电缆，否则可能会损坏内部线芯。
- 不要把示教器电缆放在尖锐的边缘处，否则可能会损坏电缆护套。
- 确保示教器电缆的弯曲半径大于 100mm，否则可能会损坏电缆。

2.3.3.3 示教器使用权限

标准示教器的模式选择开关是配备钥匙的，即手动/自动模式只能通过钥匙进行切换，请妥善保管钥匙，且仔细考虑钥匙的使用权限。通常经过安全培训、基础操作培训的人员才可认为有权限使用钥匙。

警告

示教器上的模式选择开关钥匙是按标准为同一型号的所有示教器设计的。

请确保所有的钥匙均由符合权限的人员保管，以防止误用。

操作示教器界面的人员也应该有使用权限的区分，以保证调试人员、维护人员根据自身所负责的事务来正确、合理的使用示教器。

控制系统内置了三个级别的用户，根据操作权限从低到高分别是 operator，admin 和 god。从低权限用户切换到高权限用户需要输入密码，反之则不用。高权限的用户可以修改相同或更低级别用户的密码，operator 级别用户密码不能修改。对于不同权限的操作内容划分，请查看对应的控制系统操作手册。

2.3.3.4 无示教器模式

当控制系统选择为无示教器模式时，需特别注意调试、编程时的安全，需保证操作人员的近距离范围内安装或放置有急停按钮设备，且急停按钮的信号接入到机器人系统的安全 IO 接口中，

以便人员在遇到紧急情况时可及时按下急停按钮来保护自身和设备的安全。

2.3.4 使用控制柜的安全事项

2.3.4.1 确保控制柜内无导电异物

当控制柜进行了维修、器件更换等操作后，请务必检查柜内是否有可导电的异物，这些物体很可能在机器人使用过程中造成控制柜的短路，进而引发其他危险。

2.3.4.2 禁止在柜门打开时给控制柜供电

- 机器人控制柜开机前，必须确保已经关闭了柜门。
- 控制柜内部的带电器件或节点并未全部进行防护，故禁止在柜门打开的情况下使用控制柜，很可能给操作人员或设备带来致命危险！
- 当柜门打开时，控制柜不能达到声称的防护等级。
- 当柜门打开时，柜内器件更易受到电磁环境的干扰，且可能对外产生超出标准的辐射，很可能直接影响机器人系统的使用。

2.3.4.3 禁止将控制柜用于其他用途

控制柜仅用于控制机器人本体运动，应禁止将其用于其他用途，如人员站立在柜体上、人员在控制柜上办公、人员将柜体用作梯子等。

2.3.5 急停按钮测试

急停按钮是触发紧急停止状态的唯一手段，也是紧急情况下保障操作人员和设备的安全的最重要装置。

因此，在机器人第一次投入使用、机器人检修完成后的第一次启动等时刻，需首先对急停按钮进行测试，包括对集成商接入机器人系统的外部急停按钮进行测试，以确认按下急停按钮可使设备进入紧急停止状态、进行复位操作可解除紧急停止状态。

急停按钮确认无异常后，才可以对机器人进行配置或者编程。

2.3.6 从紧急停止状态恢复

2.3.6.1 说明

系统处于紧急停止状态时必须执行复位操作才能恢复到正常状态。复位过程非常简单但是非常重要，它保证了机器人系统不会以危险状态投入到生产运行中。

2.3.6.2 复位急停按钮

所有按钮形式的急停装置都有一个安全锁机制，被按下后必须手动释放来复位装置的急停状态。大多数急停按钮都采用旋转释放方式，旋转方向会标在按钮的表面。也有一部分按钮支持直接向上拔起的释放方法。

2.3.6.3 从紧急停止状态恢复的操作步骤

序号	操作
1	确认造成急停的危险状况已经被处理，危险源已经不存在。
2	复位引起急停的安全装置。
3	按下控制柜上的复位按钮，或点击示教器/PC 端界面上的复位按钮，使系统从急停状态中恢复，自动模式时也可使用系统输入信号来复位系统急停状态。

	注意！此步操作与控制柜型号相关，具体操作可查看对应的控制柜产品手册和控制系统操作手册。

2.3.7 抱闸测试

系统进入紧急停止状态时，电机的动力电源将被切断，各关节抱闸将闭合。因此，抱闸的正常与否影响着进入紧急停止状态的机器人是否能保障操作人员的安全、降低风险。

在日常的机器人使用过程中，各关节抱闸会出现正常的磨损，进行抱闸测试以确认其仍具备正常的功能十分必要。

测试方法如下：

序号	操作
1	手动模式下，将每个轴依次运行到其负载最大的位置。
2	按下使能开关，使使能开关保持在中间位置。 此时电机动力电已供应，抱闸已打开。
3	松开使能开关，此时抱闸闭合。 观察机器人本体是否保持之前的位置。 可通过示教器观察各轴角度值，以确认各关节位置是否保持不变。
4	依次测试各个轴。 如果各轴位置保持不变，则认为抱闸功能可用。

2.3.8 手动释放抱闸

当机器人处于紧急停止状态时，除手动抱闸释放电路外，其他所有通往机器人本体的电源将被切断。遇到紧急情况时，可通过手动释放抱闸来移动机器人本体。

标准控制柜上安装有 1 个抱闸释放按钮，当不应用此功能时，请保持此按钮的保护罩为盖住的状态，以避免此功能被误触发。

部分型号的机器人本体上也安装有抱闸释放按钮，不同型号本体的此功能触发方式不同，请详细查看对应本体的产品手册。

危险

在手动释放抱闸前，请务必确认移动本体的过程中不会对受困人员、操作人员造成伤害！

警告

手动释放抱闸以移动机器人本体时，请注意：

小负载机器人型号可手动移动本体各轴，中负载和大负载机器人型号需要使用行车、起重机等设备辅助移动本体各轴。

2.3.9 手动模式的安全事项

2.3.9.1 关于手动模式

在手动模式下，机器人的运动处于手动控制状态。只有在使能开关处于中间位置时，才能对机器人进行 Jog 或者运行程序。

手动模式用于编写、调试机器人程序以及参与工作站试运行调试。

2.3.9.2 手动模式下的速度限制

在手动模式下，机器人末端的运动速度被限制在 250mm/s 以下，即无论是 Jog 机器人还是运行程序，不论程序中设置的速度是多少，机器人末端的最大运动速度不会超过 250mm/s。

2.3.9.3 旁路外部安全信号

在手动模式下，外部安全装置如安全门、安全光栅等信号将被旁路，即在手动模式下即使安全门被打开系统也可以进行电机使能的操作，且不会有安全门打开的信息提示，以方便进行调试。

2.3.10 自动模式的安全事项

2.3.10.1 关于自动模式

自动模式用于在正式生产过程中机器人程序的运行。

自动模式下使能开关将被旁路，因此机器人可以在没有人员参与的情况下自动运行。

2.3.10.2 启用外部安全信号

外部安全装置如安全门、安全光栅等在自动模式下会启用，安全门打开会使电机断开电源且闭合抱闸。

2.3.11 生产线上的安全处理

绝大多数情况下，机器人属于生产线的一部分，因此机器人出现故障往往不只影响机器人本身，而会影响整个生产线，同样，生产线的其他部分出现问题时，也可能会影响到机器人。因此应由对整个生产线非常熟悉的人员来设计故障补救方案，以提高整个系统的安全性。

- 需关注与机器人进行交互的其他设备

例如，当某机器人需要维护时，需将此机器人从生产线上先脱离出来，也必须同时脱离与其交互的其他设备，例如为其上料的机器人。

- 需关注机器人周围仍保持运行的其他设备

例如，生产线上的机器人需要从传送带上抓取工件，当机器人出现故障时，为保证生产过程不中断，在检修机器人的同时，传送带可能仍然保持运行，此时机器人维修人员应额外注意安全，需提前考虑运行中的传送带可能带来的风险，并制定详细的在此环境中工作的安全措施。

2.3.12 火灾事故的安全处理

2.3.12.1 轻度火灾的处理措施

在即将发生火灾危险或火灾已经发生但尚未蔓延开来的情况下，不要惊慌，保持镇定，使用现场提供的灭火装置将火焰扑灭。严禁用水扑灭因短路导致的火灾。

警告

机器人工作现场使用的灭火装置需由用户提供，用户需根据现场实际情况，选择合适的灭火装置。如果是控制器发生火灾，请使用二氧化碳（CO_2）灭火器。

2.3.12.2 重度火灾的处理措施

当火灾已蔓延开来、处于不可控状态时，现场工作人员不要再试图灭火，应立即通知其他工作人员，放弃私人物品，尽快从紧急出口向外撤离，撤离时禁止使用电梯，撤离过程中同时呼叫消防队。

若有人员衣物着火，不要让他/她跑动，应让他/她迅速平躺在地上，用衣服或其它合适物品、

方式将火扑灭。

2.3.13 触电事故的安全处理

2.3.13.1 触电事故的处理

当发现有人触电，不要惊慌，首先要尽快切断电源，根据现场具体条件，果断采取适当的方法和措施：

- 如果电源开关或按钮距离触电地点很近，应迅速拉开开关，切断电源。
- 如果电源开关或按钮距离触电地点很远，可用绝缘手钳或用干燥木柄的斧、刀、铁锹等切断电源侧（即来电侧）的电线，切断的电线不可触及人体。
- 当导线搭在触电人身上或压在身下时，可用干燥的木棒、木板、竹杆或其它带有绝缘柄（手握绝缘柄）的工具，迅速将电线挑开，不能使用任何金属棒或湿的东西去挑电线，以免救护人触电。

警告

救护人不要直接接触触电人员，否则救护人也可能触电！

2.3.13.2 触电伤员脱离电源后的处理

- 如果触电伤员神志清醒，应使其就地仰面躺开，严密监视，暂时不要站立或走动。
- 如果触点伤员神志不清，应使其就地仰面躺开，确保气道通畅，并以 5 秒的时间间隔呼叫伤员或轻拍其肩部，以判断伤员是否意识丧失。禁止摆动伤员头部呼叫伤员。就地抢救的同时尽快联系医院。
- 如果触电伤员意识丧失，应在 10 秒内判断伤员呼吸、心跳情况。若既无呼吸又无动脉搏动，可判定呼吸心跳已停止，应立即用心肺复苏法对其进行抢救。

2.4 作业人员及内容要求

2.4.1 作业人员定义

作业人员可分为以下三类：

- 操作人员

操作人员可进行机器人电源的开关，可通过示教器或其他界面启动机器人程序，不可进入安全防护区域内。

- 调试人员

调试人员可进行机器人操作，可进入安全防护区域内，可对机器人进行设置、示教、编程等操作。

- 维护人员

维护人员可进行机器人操作，可进入安全防护区域内，可对机器人进行设置、示教等操作，可对机器人进行调整、维修等操作。

警告

可进入安全防护区域内的调试、维护人员，必须提前接受并通过机器人的专业培训。

警告

在进行机器人操作、编程、维护时，作业人员必须注意安全，应根据实际情况，选择穿戴必要的物品进行作业，包括适合作业内容的工作服、安全鞋、安全帽等。

2.4.2 作业人员要求

2.4.2.1 操作人员要求

操作人员应满足如下条件：

- 操作人员的年龄应该在当地合法用工年龄范围内。
- 操作人员应具备良好的身体条件。良好的身体条件包括：良好的视力（可佩戴眼镜或隐形眼镜）、良好的听力、良好的协调能力。操作人员在工作期间不能服用可能降低心智水平的物品（如药物、酒精、毒品等）。
- 理解当地适用的安全法规，如工作安全卫生法规、工伤事故预防法规等。

2.4.2.2 调试人员要求

调试人员应符合操作人员的标准，另外，调试人员还应满足如下条件：

- 调试人员应具备基本的技术知识，能理解机器人相关的技术文件和技术图纸，能按手册文件完成其工作任务。
- 调试人员需对机器人系统的使用非常熟悉，能根据实际需求通过操作机器人合理的实现目的。

2.4.2.3 维护人员要求

维护人员应符合操作人员的标准，另外，维护人员还应具备一定的其他专业知识（如电气、机械、气动等），能按手册文件完成其工作任务。

2.4.3 作业内容要求

2.4.3.1 安装、操作的安全要求

- 在搬运、安装机器人设备时，需按照本公司手册说明的方法进行，否则有可能由于错误操作导致机器人翻倒，进而导致作业人员伤亡或设备损坏。
- 机器人设备安装好后首次使用时，务必先以低速进行，然后逐渐加快速度，不可首次就使用高速运行。
- 程序和系统变量等信息默认保存在控制柜存储设备中，为了预防由于意外引起的数据丢失，建议用户定期进行数据备份。

2.4.3.2 调试的安全要求

调试时尽可能在安全防护区域外进行，当必须在安全防护区域内进行调试时，应着重注意下列事项：

- 仔细查看安全防护区域内的情况，确认没有危险再进入安全防护区域。
- 应确认安全防护区域内的所有调试人员的位置。
- 应在确认整个系统的状态后进行作业。
- 要做到随时都可以按下急停按钮。
- 应以低速运行机器人。

调试结束后，调试人员务必在安全防护区域外进行操作。

2.4.3.3 维护的安全要求

- 仔细查看安全防护区域内的情况，确认没有危险再进入安全防护区域。
- 应确认安全防护区域内的所有维护人员的位置。
- 当接通电源时，部分维护作业有触电的危险，应尽可能先断开机器人设备及系统电源，再进行维护作业。
- 维护作业时应避免其他人员无意中接通电源。
- 在进行作业时，不要将身体任何部位搭放在机器人设备的任何部分，以免造成不必要的人身伤害或对设备造成不良影响。
- 进行维护作业时，应配备适当的照明器具。
- 如需更换部件，务必使用本公司指定部件。若使用指定部件以外的部件，有可能导致机器人设备的损坏。
- 在更换部件时拆下来的零件（如螺钉），应正确装回其原来部位，如果发现零件不够或零件有剩余，则应再次确认并正确安装。

2.5 安全培训

2.5.1 概述

现场操作人员、调试人员、维护人员必须经过正规的机器人安全及操作培训，并考核合格后，才能对机器人进行操作、调试和维护。禁止非专业人员、培训未合格的人员操作、调试或维护机器人，以免对作业人员和机器人设备造成严重损害。

设备的所有作业人员都应做到：

- 判断设备的当前状态，保证设备当前处于无故障的情况，才对机器人设备进行操作、调试等作业。
- 当遇到紧急事件时，能选择最安全的方式处理，最大限度降低生命和财产损失。
- 充分理解本公司的产品手册文件，按文件的要求对设备进行作业。

2.5.2 作业人员的安全

下面列出一般性注意事项，请考虑采取以确保作业人员的安全：

- 在设备运行时，即使机器人看上去已经停止，也有可能是因为机器人在等待启动信号而处在即将运行的状态。此状态也应该视为设备处在操作状态。
- 外围设备均应进行良好的接地。
- 应尽可能地将外围设备安装在机器人工作范围之外。
- 应采用在地板上画线等方式来标清机器人设备的动作范围。

2.5.2.1 操作人员的安全

操作人员不可进入安全防护区域内作业：

- 应在安全防护区域外进行机器人操作。
- 为了防止无关人员误入安全防护区域，应设置防护栏或安全门。
- 不需要操作机器人时，应断开控制柜电源，或者按下急停按钮。
- 应在操作人员伸手可及范围之内设置急停按钮。

2.5.2.2 调试人员的安全

在进行调试作业时，某些情况下需要进入机器人的工作范围内，此时尤其要注意安全：

- 在进行调试作业之前，应确认设备处在安全状态。

- 应事先确认安全装置（如急停按钮）的位置和状态。
- 应特别注意，勿使其他人员进入机器人工作范围内。
- 在机器人启动前，应充分确认机器人工作范围内没有人员。

在调试结束后，务必按照下列步骤执行测试运转：

- 在低速下，单步执行程序至少一个循环，确认没有异常。
- 在低速下，连续运行程序至少一个循环，确认没有异常。
- 在实际应用的运转速度下，连续运行程序至少一个循环，确认没有异常。

2.5.2.3 维护人员的安全

为了确保维修人员的安全，应充分注意下列事项：

- 进行维修作业前，应确认外围设备处在安全状态。
- 进行维修作业前，应尽可能先断开设备电源。应根据需要先用锁等锁住主断路器，以避免其他人员无意中接通电源。
- 当迫不得已必须要在通电的情况下进入机器人工作范围内时，应在按下急停按钮再进入。维护人员应挂上“正在维修”的标牌，以避免其他人员无意中操作设备。
- 在机器人运动过程中，切勿进入机器人的工作范围内。
- 当机器人工作范围内有其他人员时，切勿执行自动程序运行。
- 进行维护作业时应在设备旁边配置一名熟悉机器人系统、能够察觉危险的人员，使其在紧急情况下可以按下急停按钮。
- 在更换部件或重新组装时，应注意避免异物粘附或者异物混入。
- 在检修控制柜内部时，如要接触到电源单元、印刷电路板等时，为了预防触电，务必先断开控制柜主断路器的电源，再进行作

3 使用 Titanite

3.1 关于本章

说明

本章将从整体上介绍有关 Titanite 机器人控制系统的组成和基本概念。

Titanite 控制系统由控制柜和示教器两个主要部件和相关的配套电缆组成。

3.2 控制柜

3.2.1 关于控制柜

说明

Titanite 机器人控制系统的所有主要组件安装在一个控制柜中，该控制柜也称为 XBC（xBot Control Cabinet 的缩写），在您机器人控制柜的铭牌上一般使用 XBC 的命名方式。

XBC 控制柜内包含了控制机器人运动的所有必须组件，例如主控制器、IO 模块、安全模块、伺服驱动器、电源模块等，Titanite 系统的控制软件运行在主控制器上。

基本参数

下方表格列出了 XBC 控制柜的基本参数：

参数	值/说明
外形尺寸	XBC3：469 mm(含导轨模块则 544) × 497 mm × 261 mm （宽 × 深 × 高） XBC3N：493mm(含导轨模块则 568) × 557mm × 312mm XBC3E：560mm × 485mm × 758mm
自重	XBC3：30kg（含导轨扩展模块则为 32kg） XBC3N：33kg（含导轨扩展模块则为 35kg） XBC3E：75kg
IP 防护等级	IP40
供电类型	XBC3、XBC3N：230VAC，电压波动范围不超过-15%～+10%，频率变化范围不超过±2% XBC3E：3 x 380VAC，电压波动范围不超过-15%～+10%，频率变化范围不超过±2%
额定功率	1.5kw（XB4、XB4s、XB4h），1.7kw（XB12），2.5kw（XB7、XB7s、XB7h、XB7Ls、XB7Lh、XB7XL、XB7XLh），6.5kw（XB10），9.5kw（XB20）
工作环境温度	0℃ ～ +40℃
储存温度	-10℃～ +55℃
工作/储存最大环境湿度	≤80%，无凝露，无结霜
安装方式	XBC3、XBC3N：桌面式，应放置于高度在 0.6m-1.2m 之间的桌面上 XBC3E：落地式
工作环境	➢ 室内安装 ➢ 避免阳光照射 ➢ 远离灰尘油烟盐分铁屑等 ➢ 远离易燃性、腐蚀性液体与气体 ➢ 不得与水接触 ➢ 不传递冲击与振动 ➢ 远离电气干扰源

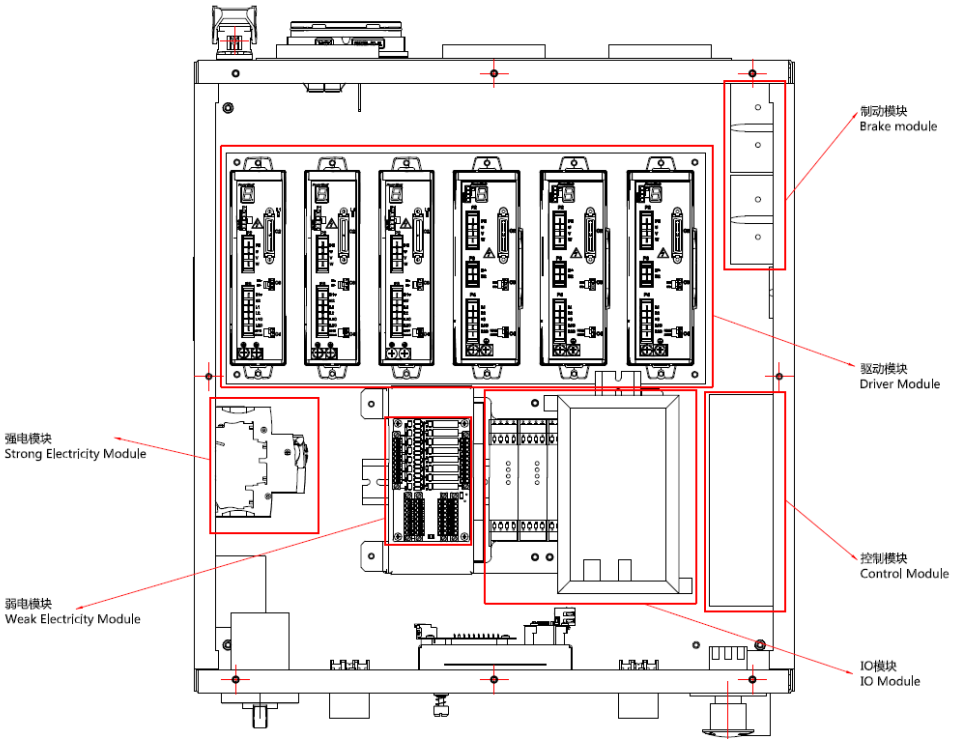
警告

控制柜电源线的过电流保护需由用户提供，用户需根据所选购的控制柜的额定功率，慎重选择合适的过电流保护器件。

> **提示**
>
> 为保证进行有效散热以避免控制系统出现过热的情况，在放置 XBC 控制柜时请确保控制柜前后各保留不低于 300 mm 的空间，左右各保留不低于 100 mm 的空间。

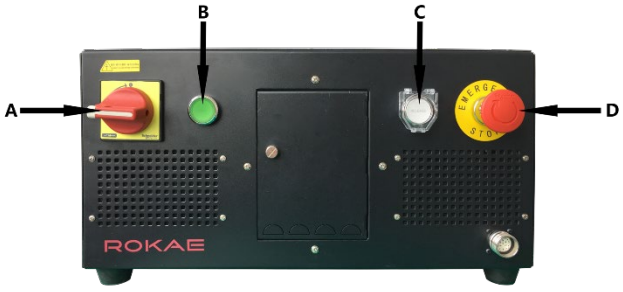
3.2.2 控制柜布局

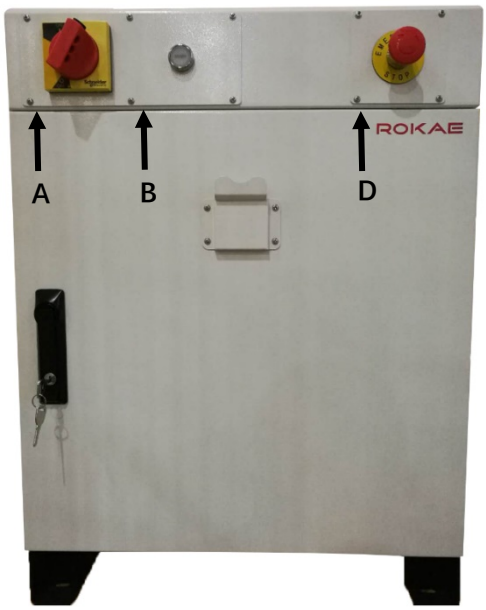
示意图



3.2.3 按钮与接口

按钮



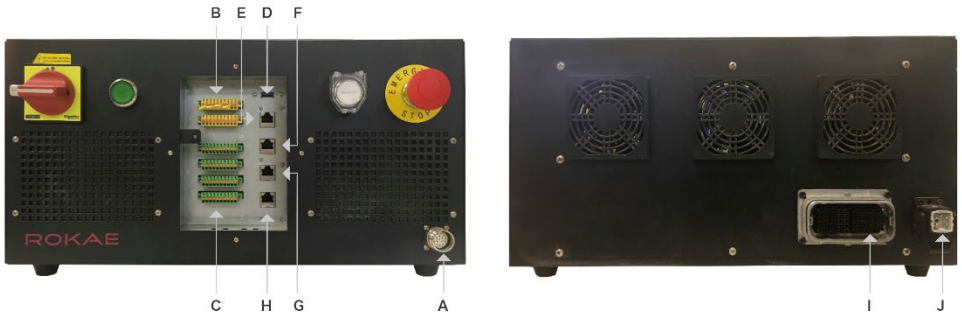


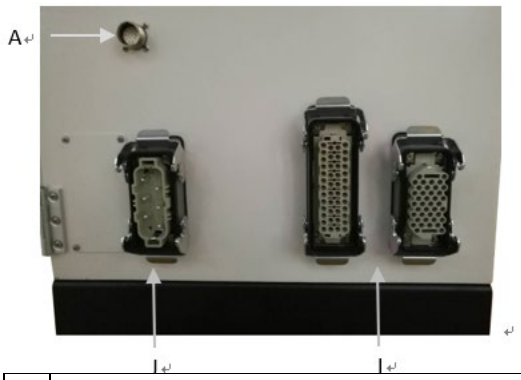
A	主电源开关，控制整个机器人系统的电源通断。
B	Motor On 按钮，用来复位急停状态或者在自动模式下接通伺服。
C	抱闸释放按钮，按下后机器人 6 个轴的抱闸都将打开，机器人可由外部力量进行移动。
D	急停按钮，按下后机器人将紧急停止，需要执行手动复位操作才能切换回正常模式。

危险

按下抱闸释放按钮后，机器人各关节的保持制动器（又称抱闸）将全部打开，在没有外力辅助的情况下机器人将自由下落，必须保证在按下抱闸释放之前已使用适当的方式固定机器人以免造成人员受伤或者机器人损坏。
在不使用抱闸释放功能时，禁止打开按钮上方的保护盖。

接口





A	示教器接口，用于连接 xPad 示教器。
B	安全 IO 接线端子，详细的针脚定义见 IO 接口定义。
C	通用 IO 接线端子，详细的针脚定义见 IO 接口定义。
D	USB 接口，用于内部调试或数据导入导出。
E	EtherCAT 设备扩展网口，用于扩展 EtherCAT 接口的 IO 模块。
F	视觉接口，用于连接 RJ45 接口的工业相机。
G	调试接口，用于售后人员进行内部调试。
H	Profinet 或 EtherNet/IP 接口，用于连接外部 Profinet 或 EtherNet/IP 总线（选配）。
I	中继电缆重载接头，包括动力线和信号线，连接到机器人本体。
J	电源插头，连接外部供电线。

危险

控制柜内包含高压部件，严禁非授权人员私自打开控制柜外壳，否则将可能造成严重甚至致命的人身伤害，珞石科技不对此类事故承担任何责任。

3.2.4 IO 及网口定义

说明

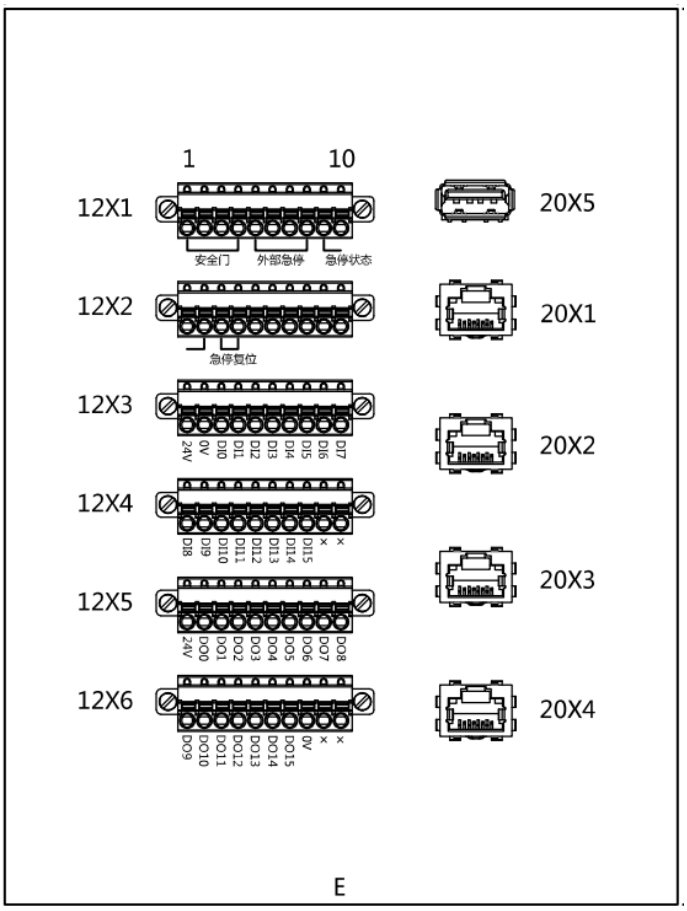
XBC 控制柜的 IO 及网口编号如下图所示，详细的接点定义见安全 IO 和通用 IO。

警告

控制柜 IO 接口需要在控制柜断电的时候才能进行接线的操作，控制柜开机时接线，如操作不当，可能引起内部电路板短路烧坏，使部分功能无法使用。
珞石有可能会对控制柜内硬件进行升级，若操作手册与控制柜接口窗背面 IO 示意图接线方式不一致，请以接口窗背面 IO 示意图为准。

提示

安全 IO 和通用 IO 接线端子具有防反插设计，当插接不顺利时应检查是否插反，避免蛮插损坏端子。



安全 IO

每一个安全输入或输出信号均为双路控制，因此每一个信号均包含“回路 1”和“回路 2”两组 IO 点，共 4 个接线端子，实际使用时应按要求接线。

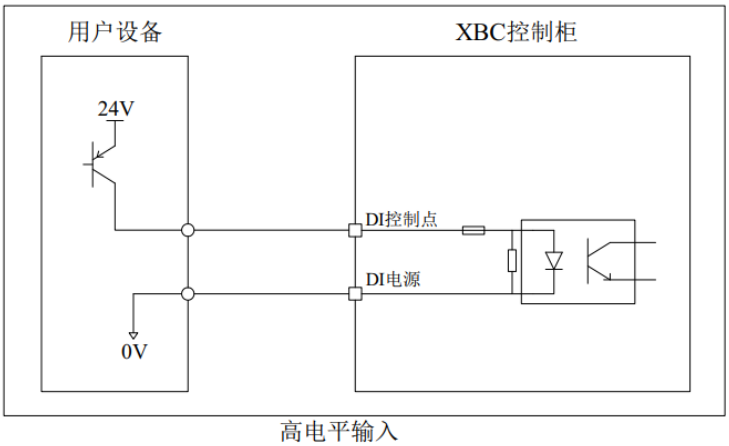
端子号	功能定义	说明
12X1-1	安全门 1	安全门回路 1，未使用时需短接。
12X1-2	安全门 1	
12X1-3	安全门 2	安全门回路 2，未使用时需短接。
12X1-4	安全门 2	
12X1-5	外部急停 1	外部急停回路 1，未使用时需短接。
12X1-6	外部急停 1	
12X1-7	外部急停 2	外部急停回路 2，未使用时需短接。
12X1-8	外部急停 2	
12X1-9	急停状态 1	机器人急停状态输出回路 1，没有急停时为常开。
12X1-10	急停状态 1	
12X2-1	急停状态 2	机器人急停状态输出回路 2，没有急停时为常开。
12X2-2	急停状态 2	
12X2-3	急停复位	机器人急停状态的复位信号输入端。
12X2-4	急停复位	
12X2-5—12X2-10	预留	

通用 IO

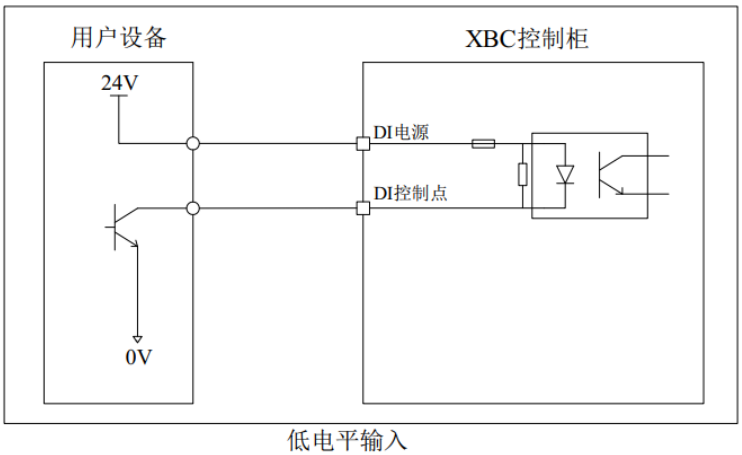
XBC 控制柜默认带有 16 个数字输入和 16 个数字输出接点，如现场需要更多 IO 信号，可通过 EtherCAT 接口扩充新的 IO 模块。

通用 IO 不具备直接驱动负载的能力，请根据现场负载的情况决定是否需要使用中间继电器，其中 DI 为高电平、低电平输入均有效，DO 为高电平输出，每个触点容量为 100mA。

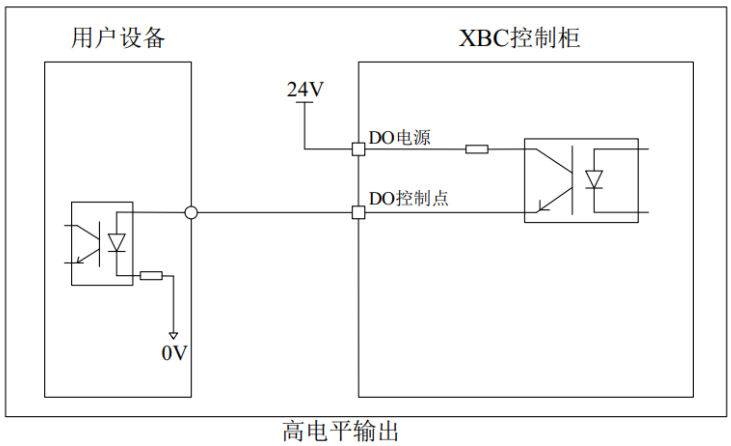
DIDO 的接线示意图如下图：



高电平输入



低电平输入



高电平输出

详细的 IO 接口定义见下表：

端子号	功能定义	说明
12X3-1	DC24V	DI 电源，需现场提供
12X3-2	DC0V	DI 电源，需现场提供

12X3-3	DI0	用户自定义
12X3-4	DI1	用户自定义
12X3-5	DI2	用户自定义
12X3-6	DI3	用户自定义
12X3-7	DI4	用户自定义
12X3-8	DI5	用户自定义
12X3-9	DI6	用户自定义
12X3-10	DI7	用户自定义
12X4-1	DI8	用户自定义
12X4-2	DI9	用户自定义
12X4-3	DI10	用户自定义
12X4-4	DI11	用户自定义
12X4-5	DI12	用户自定义
12X4-6	DI13	用户自定义
12X4-7	DI14	用户自定义
12X4-8	DI15	用户自定义
12X4-9	x	预留
12X4-10	x	预留
12X5-1	DC24V	DO 电源，需现场提供
12X5-2	DO0	用户自定义
12X5-3	DO1	用户自定义
12X5-4	DO2	用户自定义
12X5-5	DO3	用户自定义
12X5-6	DO4	用户自定义
12X5-7	DO5	用户自定义
12X5-8	DO6	用户自定义
12X5-9	DO7	用户自定义
12X5-10	DO8	用户自定义
12X6-1	DO9	用户自定义
12X6-2	DO10	用户自定义
12X6-3	DO11	用户自定义
12X6-4	DO12	用户自定义
12X6-5	DO13	用户自定义
12X6-6	DO14	用户自定义
12X6-7	DO15	用户自定义
12X6-8	0V	DO 电源，需现场提供
12X6-9	x	预留
12X6-10	x	预留

USB 接口

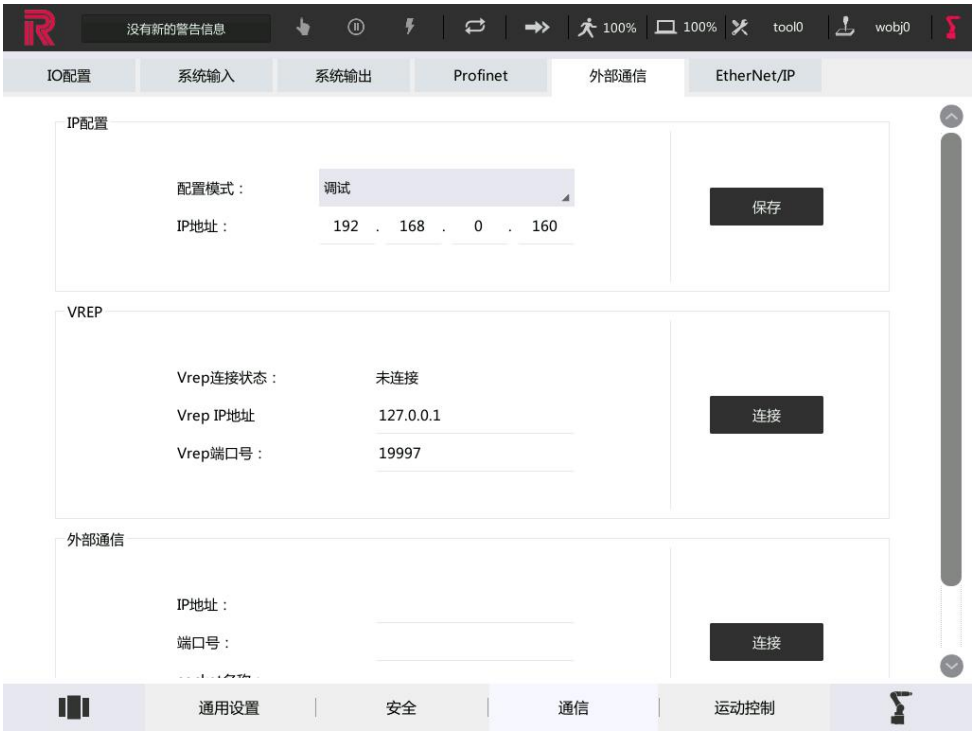
XBC 控制柜默认带有 1 个 USB 接口，方便现场调试，端口号为 20X5。

网口

XBC 控制柜默认带有 4 个网口，包括 1 个 EtherCAT 扩展网口，1 个调试网口，1 个视觉网口和 1 个 Profinet 网口。

端子号	功能定义	说明
20X1	EtherCAT 扩展网口	用于扩展 EtherCAT 从站。
20X2	视觉网口	用于和视觉通信，默认 ip 地址：192.168.2.160
20X3	调试网口	用于开发调试，默认 ip 地址：192.168.0.160
20X4	Profinet 网口	用于连接 Profinet。

用户可通过控制面板-通信-外部通信修改调试 IP 和视觉 IP，修改完成点击保存，即可立即生效。



警告

调试 IP 和视觉 IP 网段不允许为 192.168.1.XXX，且调试和视觉网段不允许重复，比如视觉 IP 和调试 IP 不能同时为 192.168.2.XXX。

3.3 示教器

3.3.1 关于示教器

说明

示教器是一个自带完整硬件和软件的嵌入式手持设备，您可以使用示教器来完成与机器人有关的所有功能，例如编写和调试程序、查看系统状态、设置系统参数等等。

Titanite 系统配备的示教器称为 xPad，是一款精心设计的可靠易用的产品，熟练使用 xPad 将有效提高机器人的使用效率。

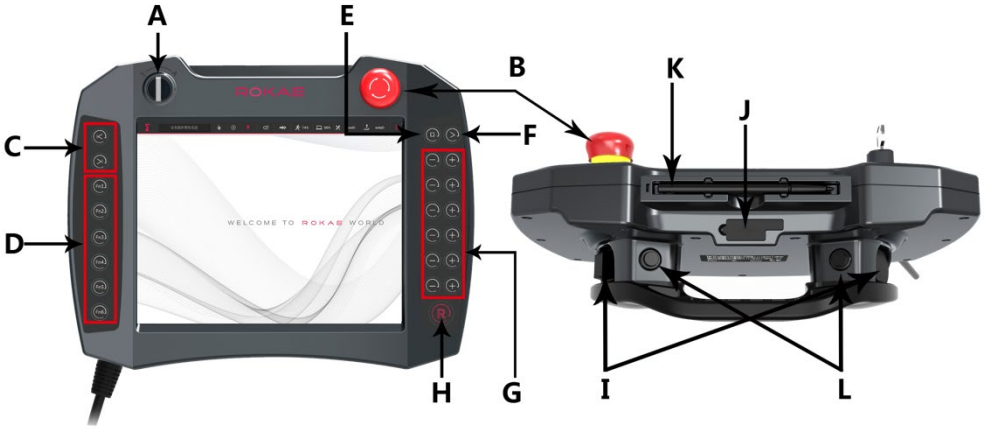
基本参数

下方表格列出了 xPad 示教器的基本参数：

参数	值/说明
屏幕尺寸	9.7 英寸
重量	1.15kg（不含电缆）
屏幕分辨率	1024 * 768
电缆最小折弯半径	64 mm
IP 防护等级	IP65
工作环境温度	0°C ~ +40°C
储存温度	-10°C~ +55°C
工作/储存最大环境湿度	≤80%，无凝露，无结霜
噪音水平	≤70dB(A)

3.3.2 按钮与接口

说明



A	**模式选择钥匙开关**，用来切换机器人的工作模式。
B	**急停按钮**，用来在危险情况下触发紧急停止。
C	**单步运行按钮**，用来向前或者向后运行 1 条指令，用来调试机器人程序。
D	**自定义功能按钮**，用户可为每个按钮单独制定功能，其中 fn6 已占用，不可以定义。
E	**停止按钮**，用来停止机器人程序的运行。
F	**开始按钮**，用来启动机器人程序的运行。
G	**Jog 按钮**，共 6 组 12 个，对应机器人的 6 个关节或者笛卡尔空间 6 个自由度。
H	**R 按钮**，用于在自动模式时对电机上电。
I	**三位使能开关**，用于在手动模式下对机器人进行运动使能。
J	**USB 接口**，用来连接 U 盘，采用橡胶盖保护。
K	**触摸笔凹槽**，用来存放触摸笔。
L	**预留按钮**，用于后续功能扩展。

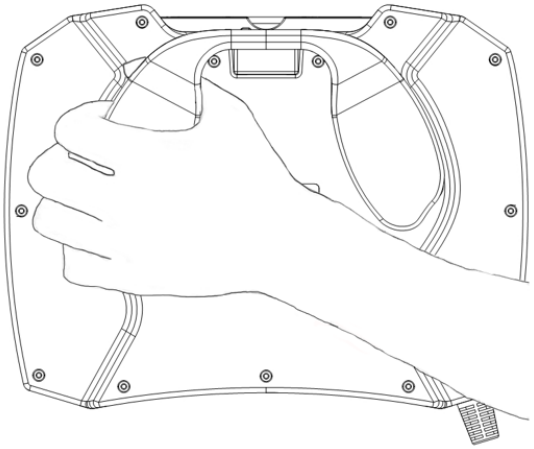
警告

请妥善保管模式选择开关配备的钥匙，以免丢失后影响使用或者非授权人员更改操作模式导致机器人停止造成生产损失。

3.3.3 如何握持示教器

说明

通常情况下示教器都采用手持方式进行操作。习惯于右手操作用户需要使用左手握持示教器，然后使用右手操作示教器上的按钮和触摸屏，推荐的握持方式如下图所示：

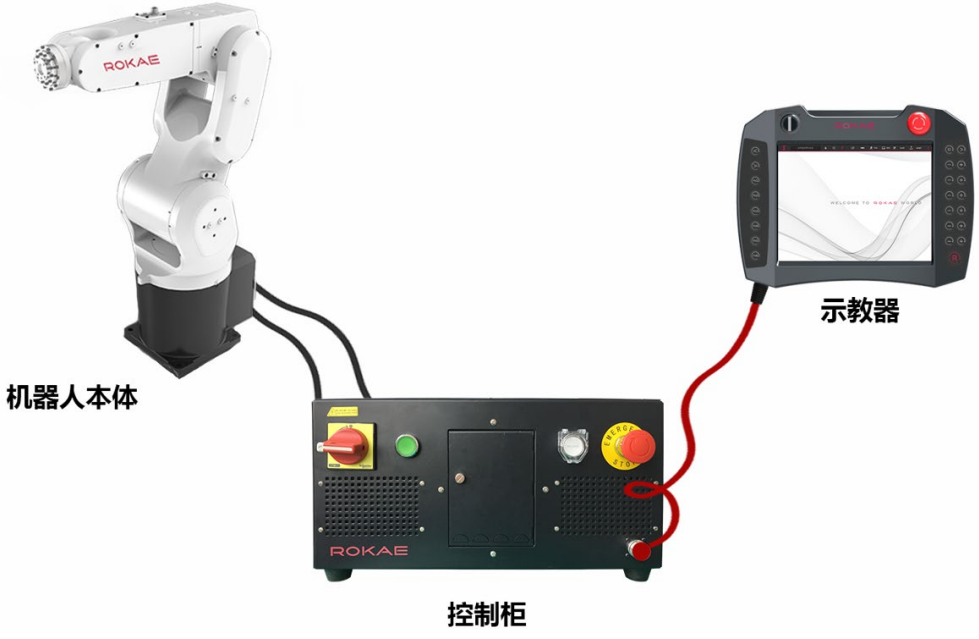


3.4 启动系统

说明

机器人的包装箱中含有所需的全部中间电缆，包括：

- 机器人中继电缆，包括动力电缆和信号电缆，带重载接头；
- 控制柜供电电缆，黑色，带有 4 针重载接头；
- 示教器电缆，红色，固定在示教器上。



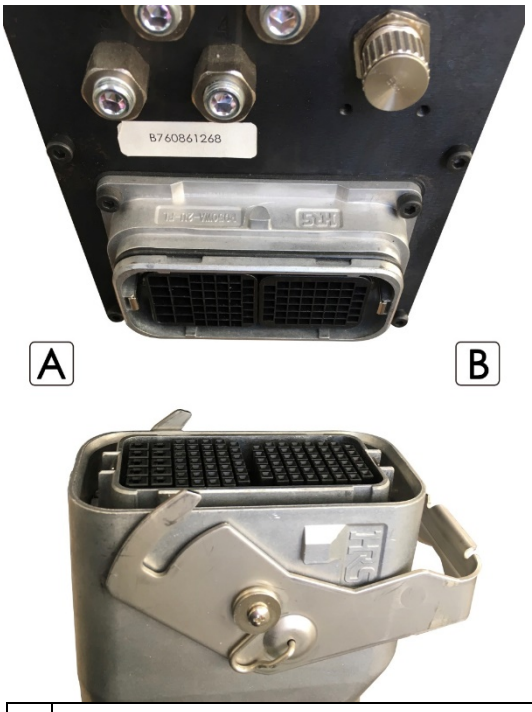
连接示教器

请参照下图连接示教器电缆：

> 提示
>
> 示教器电缆接头上的每个插针都标有编号，请务必按照正确的角度连接插头并旋紧。
> 请注意，在机器人通电时拔下示教器电缆会造成机器人急停！

连接本体

请参照下图连接本体电缆，此为本体底座后侧接线面板：



A	动力电缆。
B	信号电缆。

警示

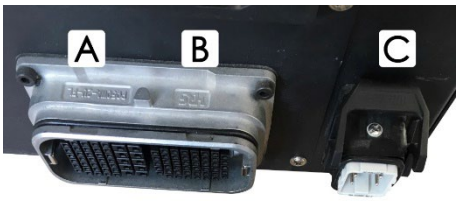
机器人本体的接地回路需由用户提供，用户需根据现场实际情况、空间位置等因素，使用随设备提供的端子将机器人本体良好接地，尽可能选择较短的接地回路。

提示

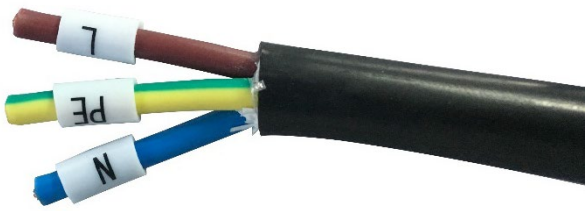
控制柜与机器人中间电缆的重载连接器内的插针具有一定的活动余量，因此连接重载插头时动作要轻柔，遇到阻力时应检查时候插针有歪斜情况，如有则需纠正后再进行连接，以免损坏重载插针。

连接控制柜

请参照下图连接控制柜电缆，此为控制柜 XBC 后侧接线面板：



A	动力电缆。
B	信号电缆。
C	外部电源电缆



对于外部电源电缆，请根据下表接线：

型号	针号/线缆颜色	定义
XBC3、XBC3N	1	火线（L）
	2	预留
	3	零线（N）
	4	预留
	PE	地线（PE）
XBC3E	黄/棕	L1（A 相）
	绿/黑	L2（B 相）

	红/灰	L3（C 相）
	蓝	零线（N）
	黄绿	地线（PE）

警示

机器人控制柜的接地回路需由用户提供，用户需根据现场实际情况、空间位置等因素，使用随设备提供的端子将控制柜良好接地，尽可能选择较短的接地回路。

提示

控制柜与机器人中间电缆的重载连接器内的插针具有一定的活动余量，因此连接重载插头时动作要轻柔，遇到阻力时应检查时候插针有歪斜情况，如有则需纠正后再进行连接，以免损坏重载插针。

启动系统

电气连接确认无误后，使用控制柜上的主电源开关来启动系统。如果一切正常，系统在启动完毕后会在示教器上显示欢迎界面，接下来就可以对机器人进行配置或者编程。
如果启动完成后系统出现报警或者根本无法启动，请查阅故障排查章节有关内容。

3.5 维护

3.5.1 关于维护时的安全

说明

警告

- 请严格遵守维护步骤，勿随意拆卸设备零部件。
- 维护作业需由指定的专业人员完成。
- 如果未接受过培训，请在电源接通时远离机器人。另外，请勿进入到机器人运动范围内，即使看到机器人似乎停止了动作，但处于通电状态的机器人可能还会意外进行动作，并可能造成严重的安全问题。
- 进入正规运转之前，请确认紧急停止开关与安全围栏开关动作状态正常。如果在开关不能正常动作的状态下进行运转，发生紧急状况时则无法发挥安全功能，可能会导致重伤或重大损害，非常危险。

触电危险

请务必在关闭控制器与相关装置电源并拔出电源插头之后进行维护、更换及配线作业，否则可能会导致触电或故障。

3.5.2 维修计划

说明

为了使机器人能够长期保持高效的性能，必须定期进行维护。检修人员须编制检修计划并严格执行。

3.5.3 检修间隔

说明

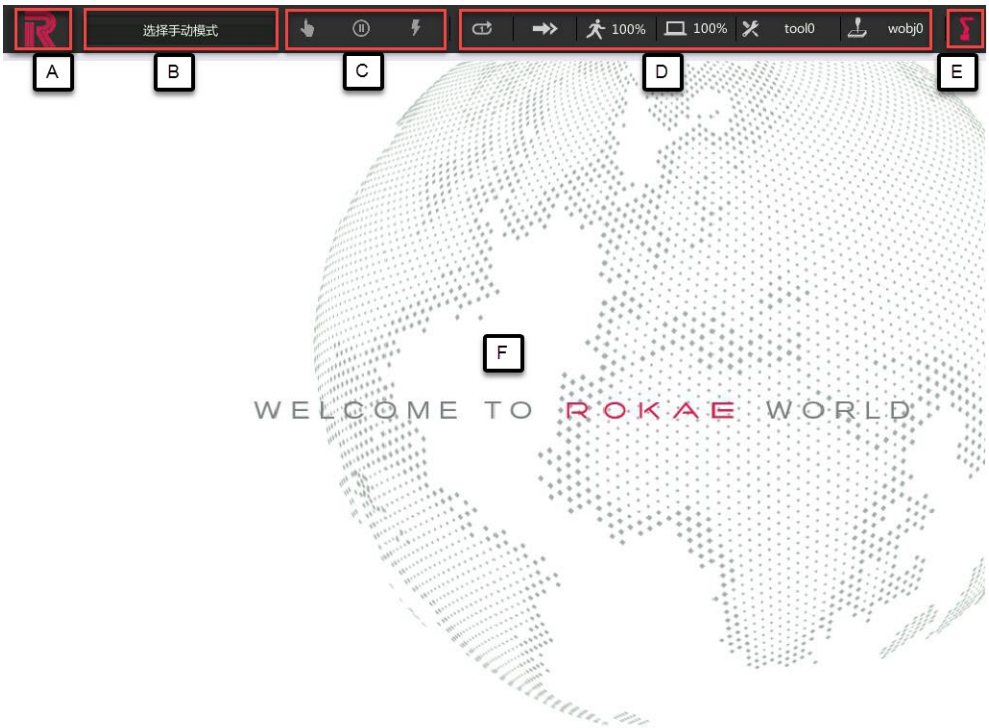
序号	检查项目	检查位置	检查间隔				
			日常	1 个月	3 个月	6 个月	12 个月
1	螺钉如有松动，请紧固	外部可见螺钉	●				
2	插头如有松动，请插紧	控制柜上的外部连接器	●				
3	检查是否存在外部损伤，清除表面	控制柜外观	●				
		示教器外观	●				
		外部电缆		●			

	的灰尘						

4 操作界面

4.1 界面布局

界面功能划分

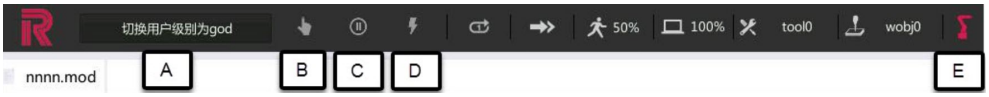


A	主菜单按钮。
B	日志提示框，用于显示最新的一条日志的标题。
C	状态栏，详细信息请参考状态栏。
D	快捷设置栏，详细信息请参考快捷设置。
E	机械单元和 Jog 指示，详细信息请参考 Jog 导轨。
F	主显示区，根据功能操作不同显示不同的内容，详细信息请参考主要页面。

4.1.1 状态栏

说明

Titanite 系统的操作模式、控制器工作状态、机器人运行状态、最新的日志等重要信息可通过状态栏图标来进行判断。



A	日志提示框，用来显示 warning 以上级别的日志
B	工作模式，分为手动，等待和自动。
C	机器人运动状态，包括运动、停止。
D	控制器状态，包括初始化、伺服使能、伺服禁止、紧急停止、常规停止、机器人故障。

E	当前激活的机械单元和操作模式。操作模式包括零力模式和位置模式。当位置模式时，显示机器人或者导轨图标，只有激活的机械单元才可以进行 Jog。当零力模式时，显示拖动机器人图标。

4.1.2 快捷设置

说明

快捷设置按钮用于快速调节 Jog 和程序运行参数，同时还可指示当前参数的设置状态。

A	程序循环模式，支持单次运行和循环运行两种方式。
B	Jog 增量模式，可选连续 Jog 和增量 Jog，并且可以调整增量步进的大小。
C	Jog 速率设置，用来调节 Jog 时的运动速度，可调范围 1%~100%。
D	程序运行速率设置，用来调节程序运行时的运动速度，可调范围 1%~100%，该参数同时影响手动和自动两种模式的程序运行速率。
E	工具设置，用来选择 Jog 和编程时使用的工具。
F	工件设置，用来选择 Jog 和编程时使用的工件。
G	Jog 参考系设置，用来选择 Jog 时的单轴和笛卡尔模式以及在笛卡尔模式下的参考坐标系。

4.1.3 主要页面

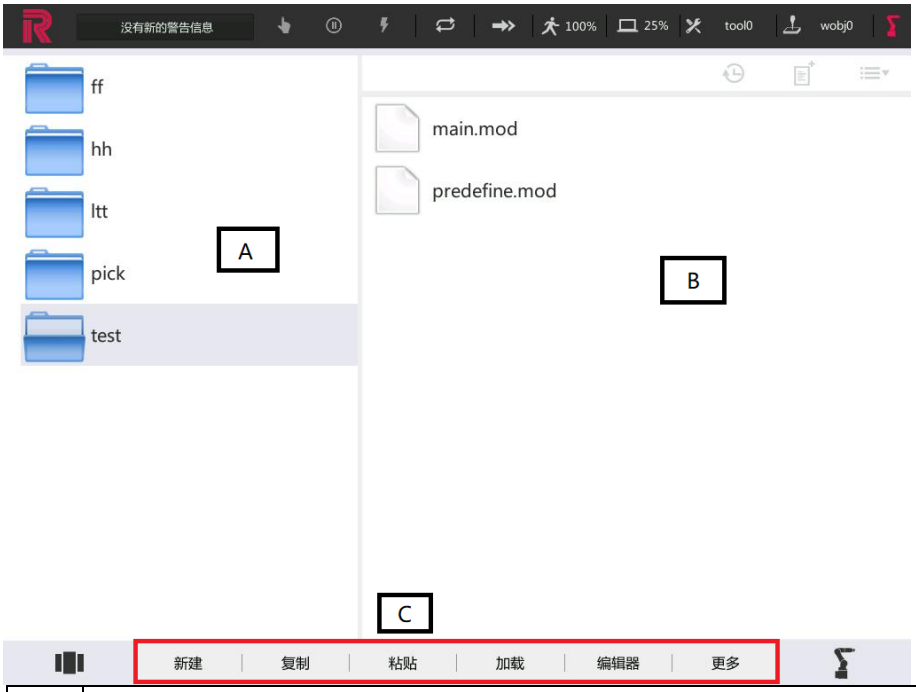
4.1.3.1 资源管理器

说明

Titanite 系统使用资源管理器页面管理用户程序。

用户的所有程序均存储在主控制器上，可以使用资源管理器进行新建、删除、复制、粘贴、重命名等文件操作。

图示



A	目录树，显示主控制器 workspace 目录下的所有用户工程。
B	文件列表，显示左侧选中文件夹下的所有程序文件。
C	菜单栏，显示常用的功能按钮。

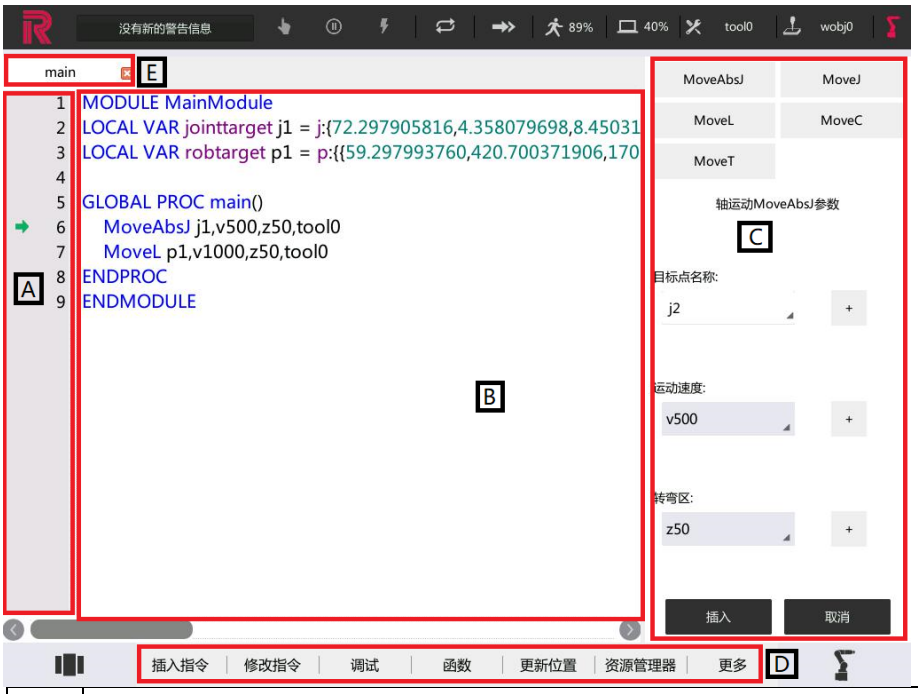
有关资源管理器的更多内容，请参考管理程序文件。

4.1.3.2 程序编辑器

说明

程序编辑器用来编写机器人程序文件，编辑窗口上方的 Tab 页显示当前编辑文件的名称。

图示



A	运行指示区，显示行号、程序指针以及运动指针。
B	主编辑区，关键字高亮显示。
C	指令编辑区，Titanite 系统提供了便捷的插入指令和修改指令界面，支持绝大部分 RL 指令的快速插入和修改。
D	工具按钮栏。
E	当前加载的文件名。

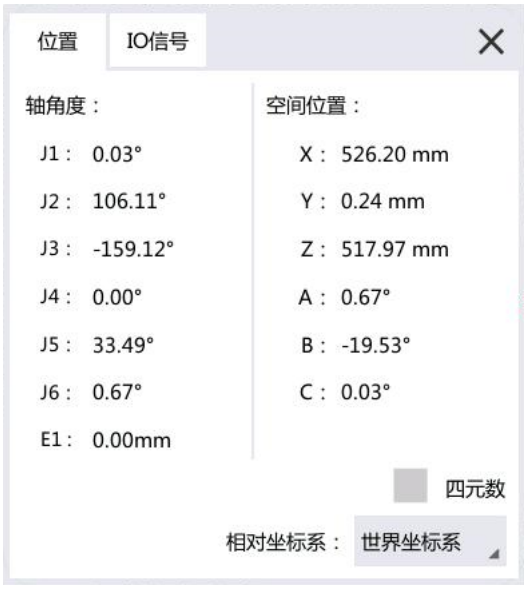
有关程序编辑器的更多信息，请参考编写程序。

4.1.3.3 状态监控

说明

用来查看机器人当前关节角度、空间位置、IO 状态及 socket 状态。

图示



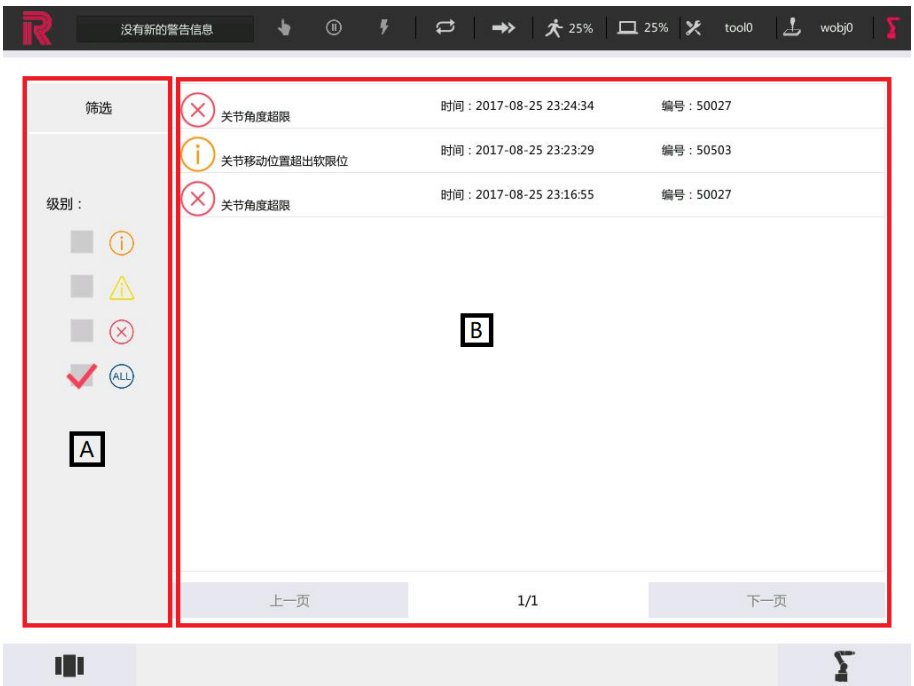
状态监控页面是一个浮动窗口，可以拖动到示教器屏幕的任何地方。有关状态监控的更多信息，请参考查看位置与 IO。

4.1.3.4 日志管理

说明

Titanite 系统提供了详细的运行日志，可以用来追溯机器人的运行状况，查找故障原因。使用日志管理界面对产生的日志进行筛选、查看等操作。

图示



A	筛选条件区，可选择日志级别进行筛选。
B	日志显示区，只显示日志的名称、级别、产生时间以及编号等基本信息，每页 10 条，使用翻页按钮切换上下页。
详情页	日志详情，点击某条日志后会进入日志详情页，内容包括日志的详细说

	明、产品该日志的原因、推荐的解决方法等。

补充说明

Titanite 系统最多保存 9999 条日志，超过这个数量后，会自动删除日期最老的日志以便存储新的日志信息。

4.1.3.5 标定

说明

标定页面用来标定工具坐标系、工件坐标系、用户坐标系以及机械零点。

图示

工具坐标系	有关工具坐标系标定，请参考标定工具坐标系。
工件坐标系	有关工件坐标系标定，请参考定义工件。
用户坐标系	有关用户坐标系标定，请参考定义工件。
零点标定	有关零点标定，请参考机械零点。

4.1.3.6 变量管理

说明

用来对机器人运动程序中使用的变量进行统一查看和修改，相比在程序编辑器中操作更简单、直观。

图示

变量类型	已使用类型显示黑色，未使用类型显示蓝色。
变量列表	根据左侧选中类型，显示当前程序所用到的所有该类型变量。
工具栏	支持对变量的新建、修改、复制、删除等操作。

关于变量管理的更多信息，请参考变量管理。

4.1.3.7 控制面板

说明

用来配置系统参数，包括通用设置、安全、通信、运动控制等。

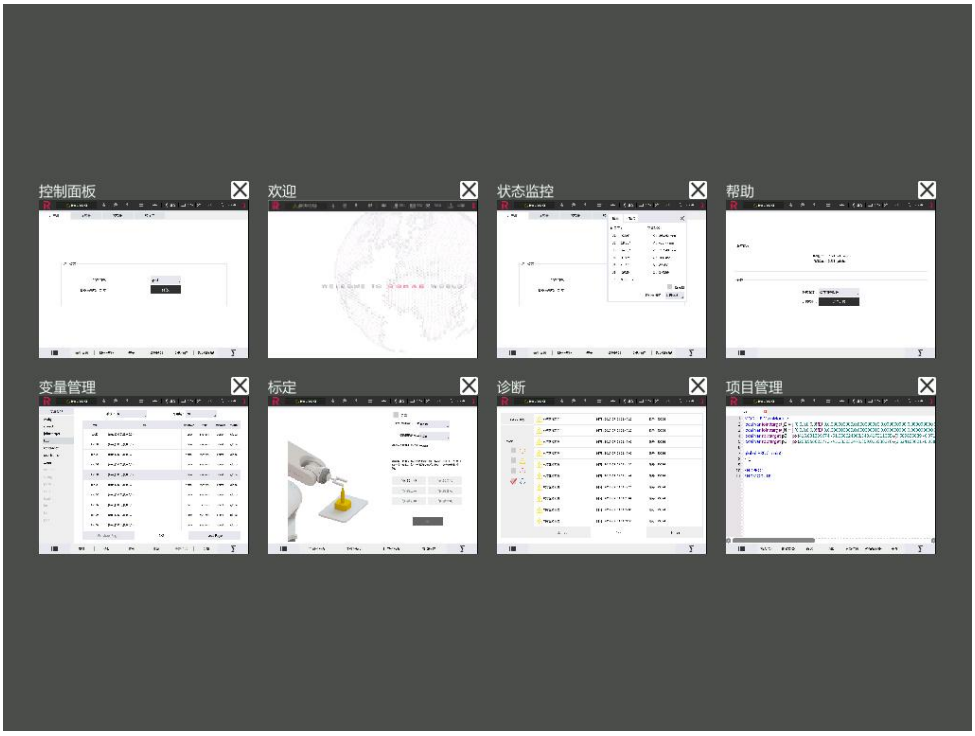
图示

通用设置	用于用户设置，示教器设置，控制器设置，备份恢复和功能授权。
安全	用来设置与安全相关的信号，包括软限位，协作模式和碰撞检测。
通信	用来设置 IO，外部控制模式等信息，更多信息请参考系统 IO。
运动控制	用来负载辨识，设置拖动示教，导轨，传送带等信息，更多信息清参考拖动示教，导轨和传送带。

4.1.4 界面切换

说明

使用屏幕左下角的“界面切换”按钮，可以快速在已经打开的界面之间切换、关闭不再需要的界面或者返回初始界面，界面列表效果如下：



4.2 基本设置

4.2.1 用户登录

说明

Titanite 系统内置了三个级别的用户，根据操作权限从低到高分别是 operator，admin 和 god。

从低权限用户切换到高权限用户需要输入密码，反之则不用。高权限的用户可以修改相同或更低级别用户的密码。operator 级别用户密码不能修改。

操作权限划分

系统功能名称	Operator	Admin	God
文件管理（新建、删除、复制、重命名等）	×	√	√
查看程序内容	√	√	√
编辑程序	×	√	√
加载、运行/停止程序	√	√	√
Jog 机器人	×	√	√
状态监控	√	√	√
坐标系标定	×	√	√
零点标定	×	√	√
日志管理	√（仅查看）	√	√
示教器设置	√	√	√
可编程按键	×	√	√
控制器设置	√	√	√
系统升级	×	√	√
恢复出厂值设置	×	√	√
备份/恢复	√（仅备份）	√	√
软限位	√（仅查看）	√	√
碰撞检测设置	×	√	√
负载参数辨识	×	√	√
超级管理员界面	×	×	√
功能授权界面	×	√	√
状态发布界面	×	√	√
查看 Debug 级别日志	×	×	√
通信界面	×	√	√
运动控制界面	×	√	√
功能授权界面	×	√	√
变量管理	√（仅查看）	√	√

4.2.2 设置软限位

说明

软限位是从软件层面来设置各轴最大运动范围的功能，用户可以根据现场情况设置软限位，避免机器人与周边的设备发生干涉或者碰撞。

> 警告
>
> 软限位范围不能超过机器人本体所允许的机械硬限位范围。

图示

admin 权限可修改关节软限位值，但不能超出软限位最大值。

用户可通过软限位开关启用或禁用软限位。当启用软限位时，Jog 和运行程序都会检测是否超出软限位，保护机器人与人员安全；当禁用软限位时，用户只能 Jog，不允许运行程序。

当机器人处于软限位之外时

在一些极少见的情况下，机器人可能会运动到软限位之外，例如机器人在运动到限位边界时触发急停，机器人在执行 STOP0 时可能会超出软限位。机器人有一个或者多个关节在软限位之外时将无法进行 Jog 和运行程序，此时需要首先取消软限位，然后将超限的关节 Jog 回软限位范围内，然后再次启用软限位。

> 警告
>
> 取消软限位功能只能用来在机器人关节超出软限位时将超限关节 Jog 回到正常范围内，软限位取消时将无法运行程序。

4.2.3 系统升级

说明

新版本控制系统软件会带来更多新功能，修复已知 bug，在 HMI 帮助页面可以查看当前使用的控制系统和 HMI 软件版本。

Titanite 系统升级非常简单，将获取的新版本软件包拷贝至 U 盘根目录并连接到示教器的 USB 接口上，在**帮助**页面选择系统升级即可。

升级软件版本时可选择需要保存的数据进行升级：

1. 工程数据，包括用户创建的所有工程，回放工程等。
2. IO 数据，包括用户添加的 IO 及 IO 配置。
3. 授权数据，即授权码。
4. 机械数据，包括机器人类型，机器人参数，软限位，零点等。
5. 日志数据，包括所有日志信息。
6. 状态数据，包括上一次加载的工程，使用的工具，工件，运动参考坐标系，设置的 Jog 速度，运行速度，选择的各种模式等。
7. 示教器配置数据，包括可编程按键。

如果用户勾选以上各项，系统将在保留勾选各项数据的前提下完成系统升级。

如果系统没有异常情况，建议保留各项进行升级。

警告

1. 由于 IO 机制改变，如果需要自行从 v2.4 版本往更高版本升级，请首先咨询珞石的技术支持人员获取完善的备份方案之后再进行升级，否则可能会造成用户数据丢失。
2. 由于 2.8.3 版本对升级机制进行了优化，如果需要由 2.8.3 及以上版本向 2.8.3 以下版本降级或由 2.8.3 以下版本升级到 2.8.3 及以上版本时，请联系珞石的技术支持人员完成备份和降级或升级，否则可能会造成用户数据丢失。

4.2.4 备份恢复

说明

Titanite 系统提供了易于使用的备份和恢复功能，可用于日常系统备份或者多台机器人之间拷贝配置数据和程序文件。

备份恢复功能页面位于控制面板内，目前仅支持使用 U 盘进行备份和恢复。

4.2.4.1 备份

图示

步骤

序号	操作	参考信息
1	将 U 盘插入到示教器的 USB 接口上	仅支持 FAT32 格式的 U 盘，不支持 NTFS 格式。
2	从控制面板进入备份恢复页面，选择备份。	如果 U 盘没有插入或者格式不正确，系统会提示插入 U 盘
3	如果 U 盘信息正常，那么 HMI 会弹出备份文件夹命名框，默认名称格式为 XBx_Backup_年月日-时间，精确到分钟。	Titanite 系统会将需要备份的信息存放到一个文件夹内，系统会自动生成一个默认文件夹名称。 您可以修改为其他任意合法的文件夹名称
4	点击确定后，系统即开始执行备份操作。	需要备份的文件包括： 工作空间文件夹 workspace 内所有文件，系统配置文件，以及必要的信息校验文件。

4.2.4.2 恢复

图示

步骤

序号	操作	参考信息
1	将包含备份文件的 U 盘插入到示教器的 USB 接口上	仅支持 FAT32 格式的 U 盘，不支持 NTFS 格式。
2	从控制面板进入备份恢复页面，选择恢复。	如果 U 盘没有插入或者格式不正确，系统会提示插入 U 盘
3	如果 U 盘信息正常，那么 HMI 会弹出一个指向 U 盘根目录的文件浏览窗口，请选择需要恢复的文件夹。	
4	手动选择需要恢复的数据。	默认不选，如果直接点击恢复，会提示请选择需要恢复的数据。
4	点击确定后，系统即开始执行恢复操作。	系统会检查如下信息： 所选文件夹中包含的文件是否合法。 当前机器人本体型号与备份文件中是否一致。 当前版本是否一致。
5	恢复进度条完成后，系统会提示重启。	恢复完成后，必须重启才能生效。

4.2.5 恢复出厂设置

说明

通过该功能，用户可以方便地恢复机器人到出厂状态，包括机器人型号，本体参数，授权状态等。

备份出厂值设置

只有 god 权限可用。

机器人出厂前，完成各项测试及参数辨识后，需执行该操作，备份机器人当前状态，以备将来恢复。

恢复出厂值设置

admin 权限可用。

机器人需要恢复到出厂状态时，可执行该操作，重启生效。

> 警告
>
> 由于恢复出厂设置操作会删除用户之前的程序和配置，请务必谨慎操作，一旦恢复无法还原。

注意事项

1. 上电状态不允许执行恢复出厂设置操作。

4.2.6 姿态调整

说明

通过该功能，用户可以方便地调整机器人姿态，包括将机器人调整到法兰与地面平行，工具坐标系 X 轴与地面垂直，工具坐标系 Y 轴与地面垂直，工具坐标系 Z 轴与地面垂直，零点姿态，转运姿态。

当需要调整姿态时，请确认机器人使用工况，手动模式上电，按住开始运动按钮，机器人开始执行姿态调整。当需要停止机器人时，直接松开按钮即可。

使用限制

1. 开启导轨情况下，不支持姿态调整。
2. 姿态调整只允许手动模式操作。

警告

使用姿态调整功能时，请务必确认机器人工况，确保机器人工作环境的安全，当察觉危险或碰撞时，请立即松开按钮或松开使能或拍下急停以使机器人停止。

5 Jog 操作

5.1 什么是 Jog?

什么是 Jog

Jog 操作是指在手动模式下通过示教器手动控制机器人的关节或者外部导轨运动。

如何开始 Jog

> **警告**
>
> 当您第一次操作机器人、机器人第一次投入生产、机器人或控制系统更换或增加了新的部件时，请先在手动模式下进行低速的 Jog 运动。可先对控制系统的紧急停止功能进行测试，即在手动按下使能按钮的情况下，观察控制柜指示灯是否处于常亮状态、示教器界面是否显示为上电状态，而后拍下示教器或控制柜上的急停按钮，观察控制柜指示灯是否处于熄灭状态、示教器界面是否显示为急停状态；确认紧急停止功能可用的前提下，将急停状态复位后，在手动模式下进行低速的 Jog 运动。

以下内容描述了进行 Jog 的主要步骤。

序号	操作	参考信息
1	Jog 前需要确认的前提条件包括： ➢ 系统以 Admin 级别用户登录并处于手动模式下； ➢ 当前并未运行任何程序； ➢ 三位使能开关正常按下且系统处于电机上电状态；	手动模式的信息可参考手动模式。
2	选择要 Jog 的对象是机器人还是导轨。	使用“切换机械单元”按钮来选择，更多信息请参考 Jog 导轨。
3	设置必要的 Jog 参数。	选择 Jog 参数。
4	选择在关节空间 Jog 还是在笛卡尔空间 Jog	更多信息可参考： 关节空间 Jog。 笛卡尔空间 Jog。 机器人坐标系统。
5	如果需要在 Jog 过程中查看机器人位置，则可以使用状态监控浮动窗口	查看位置与 IO。

5.2 机器人坐标系统

说明

机器人的运动包含位置、速度、加速度等信息，这些信息必须要指定参考系才具有实际意义，因此在开始 Jog 之前我们需要了解机器人所使用的坐标系统。
此外，定义并使用合适的坐标系有助于简化编程过程，提升机器人的使用效率。

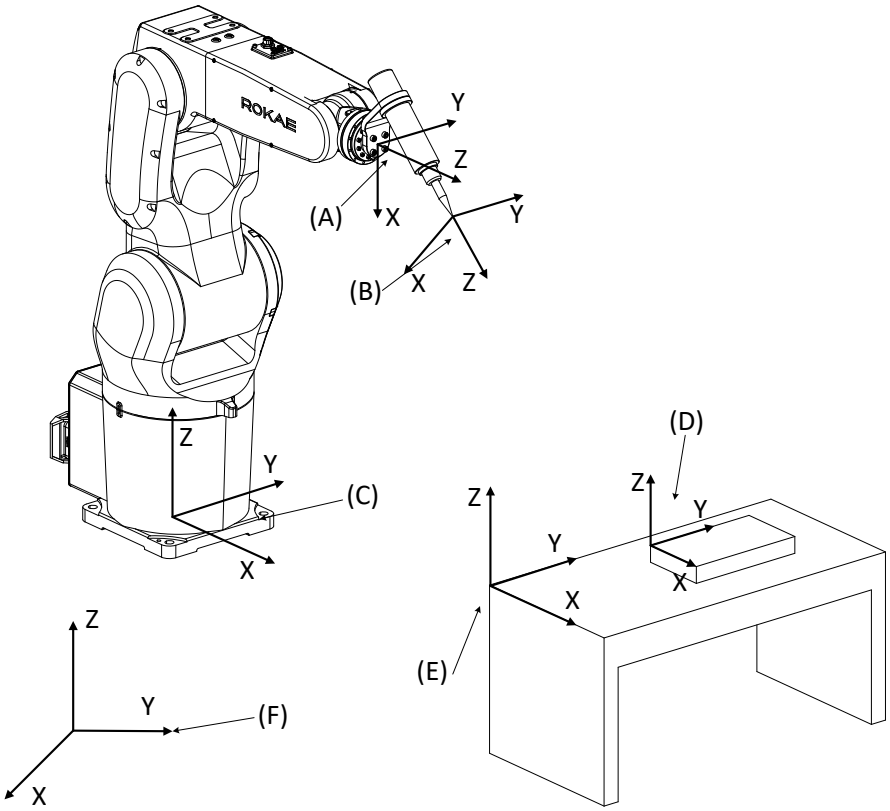
欧拉角

欧拉角用来描述刚体在三维空间的取向。对于任何坐标系，一个刚体的取向，是依照顺序，从参考坐标系做三个欧拉角的旋转而设定的。所以刚体的取向可以用三个基本旋转矩阵来决定。换句话说，任何关于刚体旋转的旋转矩阵是由三个基本旋转矩阵复合而成的。需要支持 2 种顺规，分别是 XYZ，ZYX。

当选择新的欧拉角顺规，程序和 Jog 将按新顺规执行。

坐标系

Titanite 系统使用直角坐标系（即笛卡尔坐标系，之后的文档中将统一称为笛卡尔坐标系）来描述三维空间中的位置和姿态。
目前 Titanite 系统中使用的坐标系参见下图：



A. **法兰坐标系**，定义在机器人末端法兰盘的中心位置，不具有实际意义，仅在定义工具/工件坐标系时充当参考系。

B. **工具坐标系**，定义在工具上的坐标系，机器人编程的位置指的是工具坐标系的位置，有关工具坐标系的进一步信息请参考工具。

C. **基坐标系**，定义在机器人底座的中心位置，用于确定机器人的摆放位置。

D. **工件坐标系**，定义在工件上的坐标系，良好定义的工件坐标系可以大幅降低编程复杂度并提高程序复用性有关工件坐标系的进一步信息请参考工件。

E. **用户坐标系**，在定义工件坐标系时充当参考系，不单独使用。

F. **世界坐标系**，该坐标系并没有具体的位置，当只有一个机器人时，该坐标系可认为就在机器人底座中心，与基坐标系重合；当有多个需要协调运动的机器人或外部设备时，世界坐标系可为这些设备提供一个唯一的参考系，在满足方便标定其他设备基坐标系的前提下，其具体位置可任意指定。

5.3 Jog 时的注意事项

未标定零点时

警告

机器人零点丢失后，所有的限位功能将失效，机器人可能会运动到危险的位置，因此强烈建议在发现零点丢失后执行机械零点标定操作。

速度限制

为保证安全，机器人末端在笛卡尔空间的最大运动速度将被限制在 250 mm/s。
请注意，当机器人腕心靠近一轴轴线时，由于奇异性存在可能会导致 TCP 速度很小而肘部速度很大，因此肘部的最大运动速度也会被限制在 250 mm/s。

Jog 外部轴

外部轴只能以单轴模式进行 Jog。
在 Jog 外部轴时，机器人的反应取决于当时所选的 Jog 参考坐标系，详见 Jog 导轨。

未定义工具动力学参数时

如果所使用工具的动力学参数未定义，那么的 Jog 时可能会导致伺服系统过载，因此在正式 Jog 之前请正确定义工具的动力学参数，详见标定工具坐标系或者定义工具负载。

5.4 开始 Jog

5.4.1 选择 Jog 参数

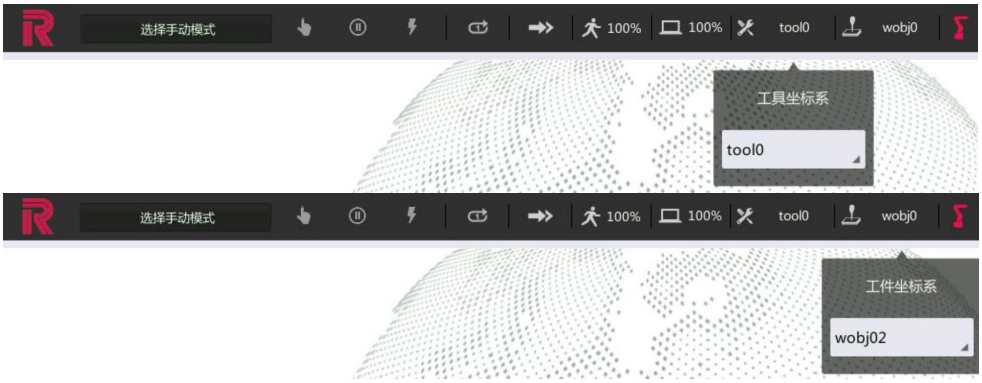
选择合适的 Jog 方式

可以使用两种方式对机器人进行 Jog。

1. 关节空间 Jog，即单独转动机器人某个关节，例如 1 轴，2 轴，5 轴等，详见关节空间 Jog；
2. 笛卡尔空间 Jog，即使机器人末端沿着三维空间路径运动，例如 xyz，或者绕这三个轴的旋转笛卡尔空间 Jog；

选择合适的工具工件

Jog 开始前需要选择合适的工具和工件，因为编写程序时插入的运动指令中使用的工具和工件与 Jog 时选择的是相同的。



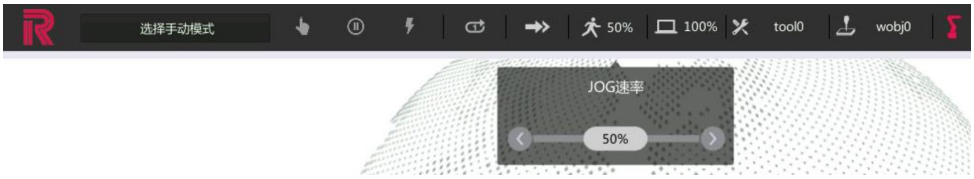
选择关节空间 Jog 时，选中的工具会被用来计算 250 mm/s 的速度约束，即工具的 TCP 平动速度不会超过 250 mm/s。

选择笛卡尔空间 Jog 时，选中的工具除被用来计算速度外，合理定义的工具和工件坐标系还可以提升编程效率。

调整 Jog 速率

根据安全规定，在手动模式下无论是关节空间 Jog 还是笛卡尔空间 Jog，机器人的最大运动速度都被限制为 250 mm/s。

Jog 速率用来控制 Jog 时机器人的运动速度，可选范围 1%~100%，100%时对应 250 mm/s。



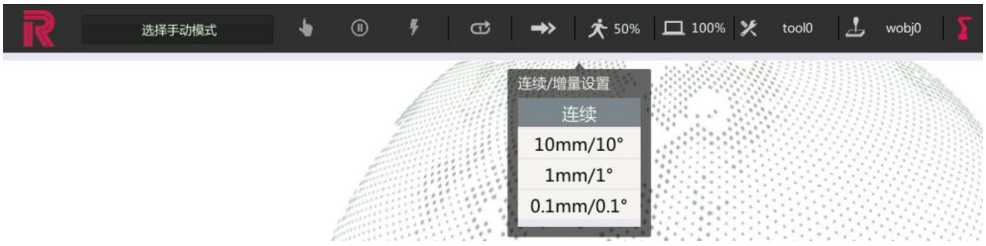
选择增量或连续模式

增量和连续模式定义了机器人如何响应单次 Jog 命令。

在连续模式下，按下某个 Jog 键后（保持按下不松开），机器人将以指定速度连续运动到关节限位处。

在增量模式下，按下某个 Jog 键后（保持按下不松开），机器人移动选定的距离/角度后会自动停止，要想继续运动必须松开按键并重新按下。该模式可用于微调示教点或定量的移

动特定距离。



5.4.2 关节空间 Jog

说明

关节空间 Jog 是最基本的手动控制机器人方式，使用示教器上的 Jog 按键一次控制一个机器人关节的转动。

关节空间 jog

	操作	说明
1	在示教器界面的右下角选择 Jog 模式为单轴	
2	选择合适的工具工件，设定期望的 Jog 速度和增量模式。	具体操作可参考选择 Jog 参数。
3	按下三位使能开关，等待电机上电后，即可使用示教器右侧的六组“+” “-”号对机器人的各轴进行 Jog 操作。	观察示教器 HMI 上方的状态栏来判断电机是否上电，详见状态栏。 六组“+” “-”号从上到下分别对应机器人的 1 轴到 6 轴。

使用限制

在关节空间 Jog 时，选择不同的工具会对关节转动的速度产生影响，这是因为在 Jog 时系

统只会限制 TCP 的运动速度不超过 250 mm/s，因此在相同的 TCP 速度下，几何尺寸更大的工具会导致关节转动速度变慢。
但是选择不同的工具不会影响关节转动的正反方向。

5.4.3 笛卡尔空间 Jog

说明

笛卡尔空间 Jog 是指在我们熟悉的三维空间移动机器人，在这个模式下机器人的移动方向是可预测的，因此更适合进行编程。
笛卡尔空间 Jog 时可选择 6 个不同的 Jog 方向，分别是沿所选参考坐标系 X、Y、Z 三个坐标轴方向的平移，以及沿这三个坐标轴的旋转 Rx、Ry、Rz。

> 提示
>
> 通常情况下，在工具或工件坐标系下 Jog 机器人更方便于编程，因此推荐始终根据现场情况定义合适的工具和工件坐标系。

笛卡尔空间 Jog

	操作	说明
1	在示教器界面的右下角选择笛卡尔空间 Jog 的参考坐标系（世界坐标系、基坐标系、工具坐标系、工件坐标系任意一个）。	
2	选择合适的工具工件，设定期望的 Jog 速度和增量模式。	具体操作可参考选择 Jog 参数。
3	按下三位使能开关，等待电机上电后，即可使用示教器右侧的六组“+”“-”号对机器人进行 Jog 操作。	观察示教器 HMI 上方的状态栏来判断电机是否上电，详见状态栏。 六组“+” “-”号从上到下分别对应参考坐标系的 X、Y、Z 三个坐标轴以及沿三个坐标轴的旋转 Rx、Ry、Rz。

使用限制

由于奇异性的存在，在笛卡尔空间 Jog 时有些位置机器人是无法到达的，当机器人位于奇异点或者非常靠近奇异点时，只能通过关节 Jog 来移动机器人。详细信息请参考机器人的奇异性。

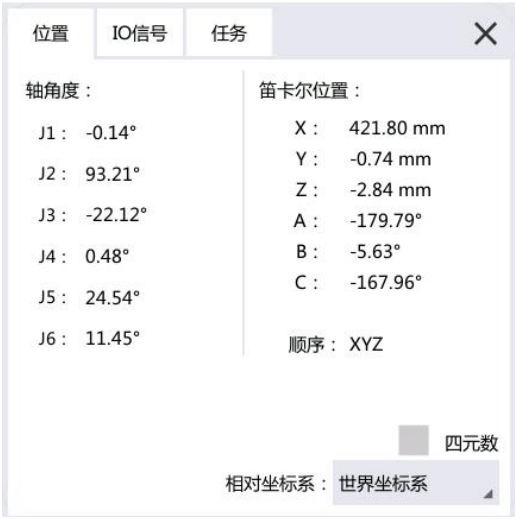
5.4.4 查看位置与 IO

说明

示教器 HMI 界面提供了专门查看机器人位置和 IO 状态的浮动窗口，方便在 Jog 和编程时使用。

机器人位置

可分别查看机器人当前各关节角度和末端的空间位置。



末端的空间位置可以选择在三个坐标系下显示，分别是工件坐标系、基坐标系以及世界坐标系。

手持工具或外部工具时，末端空间位置相对不同坐标系，表示的意义如下表：

相对坐标系	手持工具时	外部工具时
世界坐标系	工具相对世界坐标系	工具相对法兰坐标系
基坐标系	工具相对基坐标系	工具相对法兰坐标系
工件坐标系	工具相对工件坐标系	工具相对工件坐标系

> 提示
>
> 请注意此处的参考坐标系仅用于显示，与 Jog 时选择的运动参考坐标系无关，与 RL 程序中存储的目标点位置无关。

查看和调试 IO

通过勾选框，可分别查看全部 IO 信号、数字输入信号（Digital Input Signals，DI）、数字输出信号（Digital Output Signals，DO）、数字输入信号组（Groups Of Digital Input Signals，GI）和数字输出信号组（Groups Of Digital Output Signals，GO）的类型，名称，值和属性。

其中，DI（DO）分别包括普通 DI（DO）和系统 DI（DO）。系统 DI（DO）不能仿真，不能强制输出。

点击开启仿真，即可对 IO 信号进行仿真设置或强制输出。当选中 DI 或 DO 时，显示 true 和 false 按钮，用户可直接点击按钮设置 DI 或 DO 的值。

当选中 GI 或 GO 时，true 和 false 按钮隐藏，显示 123 按钮，点击 123 按钮弹出输入框，可输入 GI 或 GO 的值，注意，GI/GO 的值范围应满足要求，比如地址位选择 3-5，则允许输入的范围为 0~7。

提示

1. 一旦开启仿真模式，则会一直处于仿真状态，不接收外部 DI 和 GI 的值，关闭状态监控窗口也不会关闭仿真状态，除非点击取消仿真。
2. 重启不保存仿真状态。默认重启后仿真状态是关闭的。

6 编程和调试

6.1 关于本章

说明

本章将介绍如何对机器人进行编程以完成特定的任务。

为了更好的了解工业机器人编程的原理和方法，本章前半部分将对编程过程中涉及到的一些基本概念例如机器人的工作模式，机器人使用的工具和工件，机械零点等进行解释。在正式开始编程前，建议首先阅读该部分内容。

6.2 工作模式

6.2.1 手动模式

说明

手动模式主要用于机器人程序编写及调试。

在手动模式下，机器人的所有运动均由用户手动控制，机器人只有在运动使能时（三位使能开关处于中间位置）才会给电机通电并响应运动指令。

手动模式下的安全性

操作工业机器人是一项具有潜在危险性的工作，而在手动模式下操作人员需要在与机器人非常接近的情况下执行操作，因此操作人员必须手持示教器以便随时取消运动使能或按下急停开关，并严格遵守现场的安全规范。

手动模式下，安全光栅、安全门等外部安全设施不起作用，以方便近距离调试。

警告

只要有可能，手动模式下的操作应该在所有人员处于安全防护区域之外时进行。

通常在手动模式下执行的任务

手动模式通常用来执行以下任务：

- 紧急停止后将机器人 Jog 回路径附近以便继续运行程序；
- 创建、编写 RL（ROKAE Robot Language）程序；
- 调试 RL 程序，包括但不限于启动、停止、单步运行、更新程序位置等；
- 设置控制系统参数、标定坐标系等；
- 查看、修改变量；
- 配置 IO，比如添加、修改、删除 IO，设置系统输入，系统输出；
- 姿态调整；
- 配置可编程按键；
- 重启控制系统，修改配置文件等；
- 备份，恢复；
- 多任务设置，如新建任务，修改任务，删除任务，以及通过任务看板设置选择启动或停止的任务。
- 功能授权；
- 软限位设置；
- 导轨、多任务等参数设置；
- 系统升级，恢复出厂值设置；

6.2.2 自动模式

说明

自动模式用于连续的自动化生产，在该模式下外部安全设施（如安全光栅、安全门等）启用，而三位使能开关将不再起作用，机器人可以在没有人员参与的情况下正常工作。

机器人处于自动模式时，可以通过系统 IO 信号来控制机器人或者获取机器人工作状态。例如可以使用一个 DI 信号来启动/停止 RL 程序，使用另一个来控制电机上电。Titanite 系统支持的系统 IO 列表参见系统 IO。

警告

在切换到自动模式之前，任何挂起/停用的防护设置（如安全光栅、安全门等）都应该恢复为正常工作状态。

请注意，切换到自动模式后，机器人随时可能被外部信号启动，严禁在机器人工作区域内逗留！

通常在自动模式下执行的任务

自动模式通常被用来执行以下任务：

- 加载，启动及停止 RL 程序。
- 执行回放工程。
- 负载辨识。
- 本体参数辨识。

自动模式下的使用限制

1. 自动模式下，将无法进行 Jog。
2. 自动模式下不允许修改配置文件，配置 IO 板个数，设置机器人安装方式。
3. 自动模式下不允许配置可编程按键。
4. 自动模式下不允许备份恢复。
5. 自动模式下不允许功能授权。
6. 自动模式下不允许设置软限位。
7. 自动模式下不允许新建，修改，删除 IO 及配置系统 IO。
8. 自动模式下不允许开启/关闭碰撞检测。
9. 自动模式下不允许开启/关闭传送带。
10. 自动模式下不允许开启/关闭导轨。
11. 自动模式下不允许设置网络发布接口。
12. 自动模式下不允许开启/关闭协作模式。
13. 自动模式下不允许开启/关闭拖动示教。
14. 自动模式下不允许标定。
15. 自动模式下不允许升级/恢复出厂设置。

此外根据现场情况不同可能还存在其他使用限制，请咨询您的系统集成商获得进一步信息。

6.2.2.1 系统 IO

说明

系统 IO 分为系统输入（System Digital Input）和系统输出（System Digital Output）两种，外部控制器可通过系统输入给 Titanite 控制系统发送各种指令，如电机上电，启动程序，急停复位等，Titanite 系统也可使用系统输出功能对外发送各种状态。

系统输入

Titanite 系统支持的系统输入包括：

序号	系统输入名称
1	电机上电指令
2	电机下电指令
3	程序启动指令
4	程序停止指令
5	急停复位指令
6	程序指针到 main
7	清除报警

所有的系统输入均为脉冲触发，为了保证 Titanite 系统正确接收外部的指令，请保证外部输入的脉冲宽度不小于 300 毫秒。

> 提示
>
> 系统输入功能仅在自动模式下有效，手动模式下从系统输入传来的信号将被忽略。

系统输出

Titanite 系统支持的系统输出包括：

序号	系统输出名称	输出有效	输出无效
1	电机上电状态	电机上电	电机下电
2	程序运行状态	程序运行	程序未运行
3	工作模式	自动模式	手动模式/等待模式
4	急停状态	急停状态	非急停状态
5	路径碰撞检测状态	打开	未打开
6	Jog 碰撞检测状态	打开	未打开
7	碰撞触发状态	触发	未触发
8	故障状态	故障状态	非故障状态
9	报警状态	报警状态	非报警状态
10	Home 输出状态	在 home 点	未在 home 点

> 提示
>
> 系统输出状态在手动和自动模式下均有效，但出于安全和可用性考虑，建议仅在 Titanite 处于自动模式时使用这些信号。

使用限制

某个 IO 点与系统 IO 绑定后，将无法对其进行强制输出或者仿真输入操作。

6.2.3 模式切换

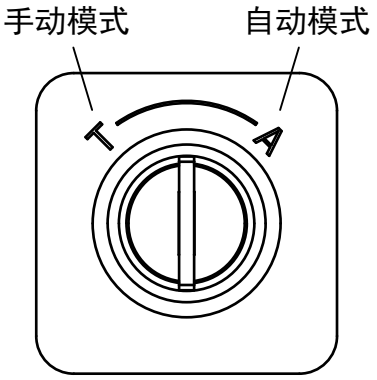
6.2.3.1 关于模式切换

当前模式

通过观察示教器上的模式选择开关指向或者 HMI 界面的状态栏，即可知道当前控制系统所处的工作模式。

模式选择开关

模式选择开关用来选择机器人的工作模式，切换模式时需要使用配套的钥匙，应禁止非授权人员使用该钥匙随意切换运行模式。



安全性

为安全起见，模式切换时系统将会切断动力电（这表示如果系统正在执行 RL 程序，机器人正在运动过程中，系统将触发 STOP 1 停止）。

手动模式转换到自动模式后，需要按下控制柜上的复位/上电按钮，动力电才会恢复到正常供电状态。

6.2.3.2 手动切换到自动

何时需要切换到自动模式

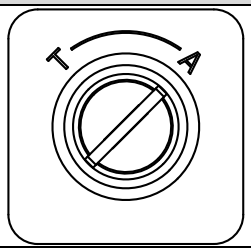
当操作人员需要对程序进行全状态、全速验证时或者程序已经准备好进行正式生产活动时，可切换到自动模式。

危险

当处于自动模式时，机器人可能在无任何警告的情况下由外部信号触发运动。

在切换到自动模式之前，必须确保没有人员位于安全防护区域内！

从手动模式切换到自动模式

	操作	说明
1	将模式选择开关旋转到自动位置，此时示教器上将弹出一个确认对话框，系统处于“等待”状态。 处于“等待”状态时，无法执行电机上电和执行 RL 程序。	
2	点击“确定”，系统将切换到自动模式。 如果在点击“确定”之前又将模式选	

	择开关旋转回手动模式，那么该确认对话框将自动关闭，系统保持手动模式。	
3	确认系统是否已顺利切换到自动模式。 如果是，那么可以继续运行之前的程序或者加载新的 RL 程序。 如果切换失败，那么需要根据错误提示排除故障。	

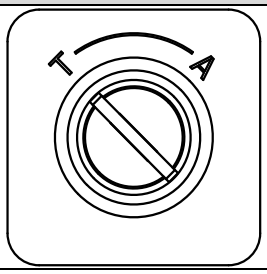
警告

在自动模式下，机器人可以被远程上电并启动 RL 程序，这意味着机器人将随时可能启动。
请咨询您的系统集成商以获取机器人系统的详细配置情况。

6.2.3.3 自动切换到手动

从自动模式切换到手动模式

从自动模式切换到手动模式非常简单，只需要一步操作。

	操作	说明
1	将模式选择开关旋转到手动位置。	
2	确认系统是否已顺利切换到手动模式。 如果是，那么模式切换已经完成。 如果切换失败，那么需要根据错误提示排除故障。	

6.3 工具

6.3.1 什么是工具?

定义

工具是指安装在机器人末端法兰上用来完成特定加工工序的器具，常见的工具有**气动/电动手爪**、**焊枪**、**喷头**等。机器人出厂时没有附带任何工具，您需要根据实际情况选择外购或者自行设计合适的工具并完成安装和设置，才可使用机器人进行工作。

使用任何一个工具之前都必须先进行标定，以获取 TCP（什么是工具中心点？）的数据。

当使用**错误!未找到引用源。**时，工具安装在机器人工作范围内的固定位置而不是机器人上。

6.3.2 什么是工具中心点?

定义

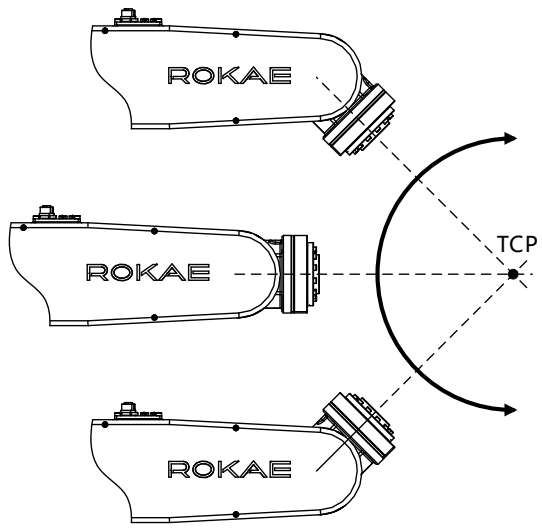
工具中心点（Tool Center Point，TCP）是位于工具上的一个特定点，通常情况下机器人使用该“点”进行加工作业，例如一个焊枪的焊丝尖端，气动手爪的某一个手指顶端等。机器人可绕 TCP 点旋转变换姿态而保持 TCP 的位置不变。

不同的工具有不同的 TCP，根据实际情况定义合适的 TCP 可以大幅提升编程效率。

TCP 也是工具坐标系的原点，详细内容可参考 tool 变量介绍。

> 提示
>
> 如无特殊说明，凡是本手册中提到“机器人位置、速度、加速度”的地方，均是指 TCP 相对于工件坐标系的位置、速度、加速度。

示意图



6.3.3 定义工具

说明

使用工具前首先需要定义新工具。在 Titanite 控制系统中，工具通过 tool 数据类型进行存

储和使用。要定义一个工具，就意味着要创建一个 tool 型的变量，关于 tool 的详细描述请参考 RL 编程语言章节的介绍（tool）。

Titanite 系统内置了 16 个预定义的工具变量 tool1~tool16，每一个 tool 变量的初始定义与 tool0 重合，用户可选择任意一个进行标定或修改之后使用，不支持用户新建 tool 型变量。

简单来讲，我们需要获得关于工具的以下参数：

1. 工具的 TCP 和姿态，即标定工具坐标系；
2. 工具的重量、质心以及旋转惯量，即工具的动力学参数；

修改工具定义只能通过 HMI 的坐标系标定界面来进行，详细步骤请参考标定工具坐标系。

对于 tool 类型的变量，在变量管理界面只能查看，不能新建或者修改。

在标定界面完成新坐标系定义之后，可以通过手动输入功能修改工具的动力学参数，也可以通过参数辨识界面来对工具的动力学参数进行辨识。

> 提示
>
> tool0 是系统预定义的工具变量，其工具坐标系与法兰坐标系重合，动力学参数都是 0。
>
> tool0 变量不允许修改。

6.3.4 标定工具坐标系

说明

工具坐标系标定指的是测量出工具坐标系相对于法兰坐标系的位置和姿态偏移量的过程。针对平面搬运的特殊应用，通过示教器选择“手动输入”，输入数据，确定工具坐标系相对于法兰坐标系的位姿即可，无需执行其他标定过程。

目前 XB12 系列机器人不支持其他方式进行工具坐标系的标定。

标定的具体流程如下

序号	操作	说明
1	使用 admin 级别的用户登录系统，并切换到工件坐标系标定界面。	
2	在标定界面选择将要标定的工具。	可通过下拉框选择要标定的工具。可选择 tool01~tool16，也可对工具重命名。
3	确认该工具是普通工具还是外部工具。	通过选择工具安装方式是外部还是手持来切换普通工具/外部工具。
4	选择标定方法为手动输入。	
5	在相应的输入框中输入工具坐标系相对于法兰的坐标系的位置和姿态。	位置包括 3 个变量，姿态包括 4 个变量，一般情况下保持姿态数据默认即可。具体坐标系定义参考 5.2 机器人坐标系统
6	点击“保存”按钮对数据进行保	

	存。	

6.3.5 定义工具负载

说明

如前所述，完整定义一个工具需要确定工具的运动学参数和动力学参数，在 Titanite 系统中使用 load 型变量存储对象的动力学参数，因此工具的动力学参数又称工具负载，详细信息请参考 tool 和 wobj 变量介绍。

工具的动力学参数需要单独指定，需在工具坐标系标定页面上，直接输入相应的数据，包括质量和质心位置。

> 提示
>
> 请务必准确定义新工具的动力学参数，否则将影响机器人的运动性能，严重情况下甚至会造成机器人过载而损坏。

6.3.6 使用工具

在 Jog 时使用

如需要使用特定工具进行 Jog 操作，只需要在示教器界面上方快捷操作栏的“工具”下拉框中选择需要的工具即可。

在程序中使用

在程序中使用特定工具非常简单，只需要在运动语句的“工具”参数中使用目标工具即可。使用示教器的“插入指令”界面编写运动指令时，其默认用的“工具”和“工件”与 Jog 时使用的一致，即界面上方快捷操作栏当前选中的“工具”和“工件”，具体操作步骤请参考插入指令。

6.4 工件

6.4.1 什么是工件?

说明

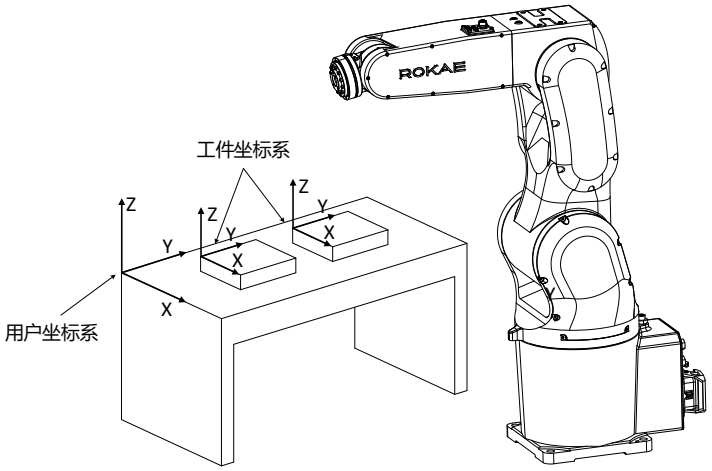
工件是指机器人使用工具进行加工或者处理的物品，在 Titanite 系统中使用 wobj（Work Object）类型的变量来描述一个实际的工件。

引入工件的概念是为了简化编程步骤，提高效率。

机器人的运动轨迹都是在工件坐标系下定义的，这样主要有两个好处：

- 当工件发生移动或者加工多个相同工件时，只需要重新标定工件坐标系，程序中的所有路径即可随之更新，而不需要重新编写程序。
- 允许加工被外部轴(如导轨，变位机等)移动的工件。

每一个工件实际上使用包含两个坐标系，一个是工件相对的用户坐标系，可以理解为摆放工件的工作台/桌子，在处理多个相同工件时非常有用；另一个是固连在工件上的工件坐标系，所有的程序路径都是在工件坐标系下描述的。



6.4.2 定义工件

说明

使用工件前首先需要定义新工件。在 Titanite 控制系统中，工件通过 wobj 数据类型进行存储和使用。要定义一个工件，就意味着要创建一个 wobj 型的变量。

Titanite 系统内置了 16 个预定义的工件变量 wobj1~wobj16，每一个 wobj 变量的初始定义与 wobj0 重合，用户可选择任意一个进行标定或修改之后使用，不支持用户新建 wobj 型变量。

工件 wobj 并没有包含动力学参数，因此定义工件的过程就是标定工件坐标系的过程。

> 提示
>
> wobj0 是系统预定义的工件变量，其用户坐标系和工件坐标系都与世界坐标系重合。与 tool0 一样，wobj0 也不允许修改。

6.4.3 标定工件坐标系

说明

在示教器上选择“手动输入”方式直接输入数据，确定工件坐标系位姿。具体流程如下：

序号	操作	说明
1	使用 admin 级别的用户登录系统，并切换到工件坐标系标定界面。	
2	在下拉框选择将要标定的工件坐标系。	
3	手动输入工件坐标系相对于用户坐标系的位置和姿态。	默认的用户坐标系与基坐标系重合，有关工件坐标系和世界坐标系的说明见 5.2 机器人坐标系统
4	点击“保存”按钮，对标定数据进行保存。	

6.4.4 使用工件

在 Jog 时使用

如需要在特定的工件坐标系下进行 Jog，只需要在示教器界面上方快捷操作栏的“工件”下拉框中选择需要的工件即可。

在程序中使用

在程序中使用特定工件非常简单，只需要在运动语句的“工件”参数中使用目标工件即可。使用示教器的“插入指令”界面编写运动指令时，其默认用的“工具”和“工件”与 Jog 时使用的一致，即界面上方快捷操作栏当前选中的“工具”和“工件”，具体操作步骤请参考插入指令。

> 提示
>
> 通常情况下，运动指令的工件参数是可选的，因此如果不特意指定的话，系统会默认使用 wobj0，默认的 wobj0 与世界坐标系重合。
> 如果使用外部工具功能，则必须指定与所用工具配套的工件参数。

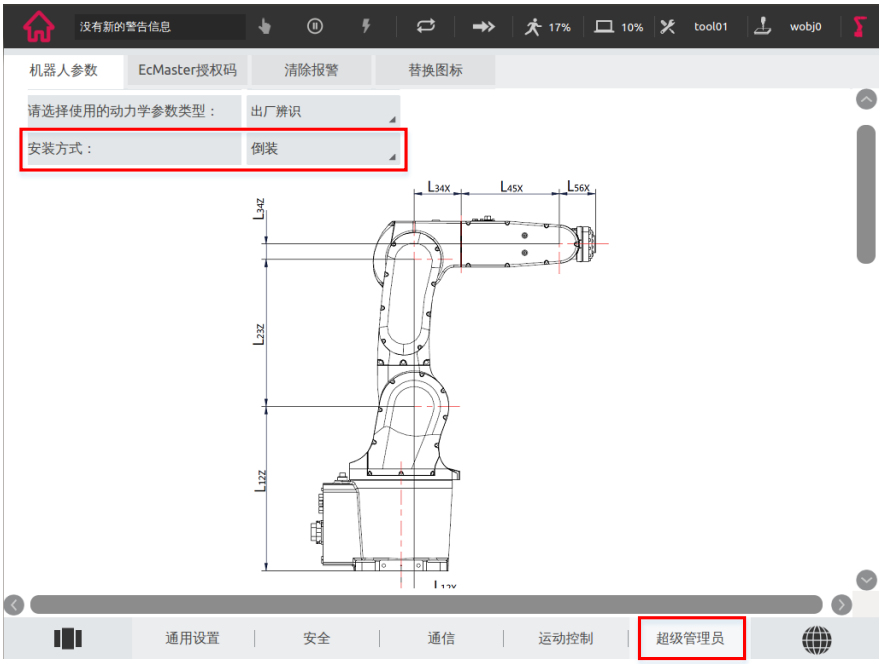
6.5 基坐标系

什么是基坐标系?

基坐标系原点定义在机器人底座的中心位置，用于确定机器人的摆放位置，默认情况下与世界坐标系的位置相同。

基坐标系的方向与机器人的安装方式直接相关，机器人正装时，基坐标系与世界坐标系完全重合，机器人倒装时，基坐标系 Z 轴与世界坐标系的 Z 轴方向相反。

机器人在使用之前，需依据安装方式，在示教器的[控制面板]-[超级管理员]-[机器人参数]-[安装方式]中进行选择，以保证基坐标系的正确性。



6.6 机械零点

6.6.1 关于机械零点

什么是机械零点

机器人在设计时会预先定义一个初始姿态，在这个姿态下各个关节的角度为 0。从机械角度来讲，零点姿态指的是相邻连杆之间形成某个特定的夹角时的姿态。从软件角度来讲，由于机器人使用编码器来记录关节角度，零点姿态是指伺服电机转动到特定编码器值时机器人的姿态。

因此机械零点实际上可以有两种解释。从观察人员的角度来看，机械零点是机器人各个关节运动到特定位置时所对应的机器人姿态；从控制系统的角度来看，机械零点是若干个编码器的数值。

6.6.2 机械标定

说明

零点标定的目的是为了让控制算法中的理论零点与实际机械零点重合，使得机械连杆系统

可以正确的反应控制系统的位置和速度指令。

更通俗的讲，零点标定是利用机械本体上预先设计好的某些定位装置将机器人的各个关节旋转到特定的角度，并通知控制系统记录此时各关节电机编码器数值的过程。

零点标定包括软标定和机械标定，使用机械标定情况如下：

- 本体维修：更换皮带、更换电机。
- 更换本体。
- 强制升级。

图示

标定步骤

序号	操作	说明
1	使用 admin 级别的用户登录系统，进入机械标定界面。	只能在手动模式下且没有运行程序时进行机械标定。 机械标定界面位于“零点标定”分类中。
2	以单轴模式 Jog 机器人或者使用抱闸释放按钮移动机器人关节，使丢失零点的轴到达标定位置。	不同的机器人具有不同的零点标定辅助装置，请查阅对应机器人的机械本体说明书。 “标定角度”是标定位置相对机械零点旋转角度值
3	当对应的轴移动到零点位置时，点击 HMI 上的对应关节的“标定”按钮完成该关节的标定。	存在耦合关系的关节需要同时进行标定。
4	重复执行步骤 3，直到机器人所有	

	的关节都完成标定。	

6.6.3 软标定

说明

使用软标定情况如下：

- 编码器电池欠压。

图示

标定步骤

	操作	说明
1	使用 admin 级别的用户登录系统，进入软标定界面。	只能在手动模式下且没有运行程序时进行软标定。 软标定界面位于“零点标定”分类中。
2	以单轴模式 Jog 机器人或者使用抱闸释放按钮移动机器人关节，使需要标定的轴移至宽窄标定槽重合，且窄槽完全位于宽槽内。	
3	当对应的轴移动到零点位置时，点击 HMI 上的对应关节的“标定”按钮完成该关节的标定。	存在耦合关系的关节需要同时进行标定。
4	重复执行步骤 3，直到机器人所有的关节都完成标定。	

6.7 配置 IO

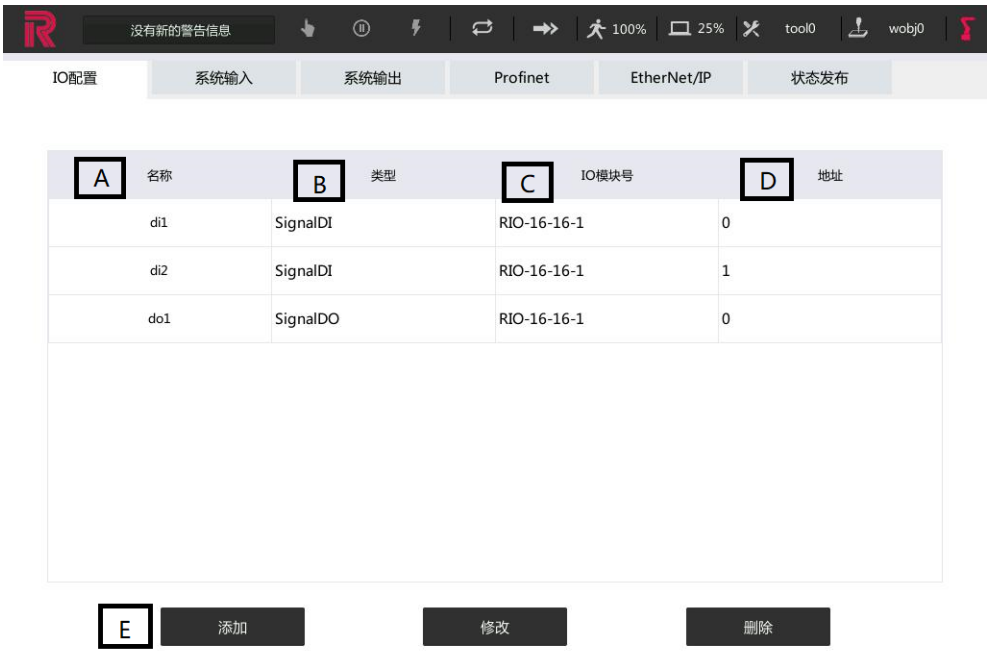
说明

Titanite 系统中，除系统 IO 信号外，其他所有通用 IO 信号都需要在控制面板的输入输出界面进行配置后才可以在程序中使用。

在 RL 程序中使用 signalxx 类型的变量来存储和访问 IO 信号，详细信息请参考 RL 章节的介绍。

添加、修改、删除 IO

所有通用 IO 信号的配置都在控制面板的相关页面进行，如下图所示：



A	输入输出信号名称，可以在“新建”或者“修改”时更改
B	信号的类型，包括 signaldi、signalgi、signaldo、signalgo 等
C	IO 模块号，珞石提供的标准 IO 模块
D	地址号，IO 信用映射对应的物理地址号，从 0 开始计算
E	功能按钮区，可以对 IO 信号执行新建、修改和删除操作

警告

如果 IO 配置出现错误，如映射 IO 端口超出物理限制或者出现端口重复分配等情况，控制系统启动时将进入 SYS_ERR 状态并在 HMI 上给出报警信息，在此种情况下只允许用户进入系统配置界面修正错误的配置，不允许其他操作。

查看 IO

配置完成的通用 IO 可以在状态监控界面进行查看，且只能看到已经配置好的 IO，支持对

IO 的强制输出或者仿真输入。
在变量管理界面不能查看通用 IO。

使用 IO

对于输入信号来讲（DI/GI），在 RL 程序中可直接使用输入信号的变量名读取输入节点的状态。

Example 1

```
//使用数字输入的状态作为判断条件
IF（di1 == true）
   do something…
ENDIF
```

对于输出信号（DO/GO），在 RL 中可以使用专门的指令 SetDO 和 SetGO 来处理，详见各指令的说明章节。

使用限制

不同的 GI/DI 信号可重复映射相同的物理端口，不同的 GO/DO 信号不可映射重复的物理端口，如果 GO/DO 映射了重复的物理端口，将会导致系统错误状态，需手动删除 DO 或 GO，或更改 DO/GO 的地址解决。

6.8 可编程按键

说明

Titanite 系统中，用户可以对示教器按键进行配置，从而方便快捷的使用配置的功能。

可通过控制面板-通用设置-示教器-可编程按键配置我们需要的按键功能。

目前支持可配置的按键包括 Fn1，Fn2，Fn3，Fn4 和 Fn5。

目前支持的可配置功能包括键盘，状态监控，截图到 U 盘，Print 框，切换机械单元，位置/零力模式，增加/减少 Jog 速率，增加/减少程序运行速率，强制输出。

提示

可编程按键配置完成后，无需重启立即生效。

功能说明

功能	说明
键盘	仅在程序编辑器界面可通过按键调出或关闭。
状态监控	所有界面均可通过按键调出或关闭。
截图到 U 盘	通过按键可将当前屏幕界面截图，并保存到 U 盘 printscreen 文件夹下，并以自增数字命名。
Print 框	仅在程序编辑器界面可通过按键调出或关闭。
切换机械单元	可通过按键在机器人和导轨之间切换，当没有导轨时，提示“导轨未开启”。
位置/零力模式	可通过按键切换位置模式和零力模式，前提是必须开启拖动示教功能且处于下电状态，以保证机器人和操作人员的安全性。

	切换成功后，可手动使能上电，拖动机器人动作。
增加/减少 Jog 速率	可通过右侧滚动条选择 Jog 速率变化幅度，从 1%~20%，当用户操作按键，状态栏 Jog 速率随之变化。
增加/减少运行速率	可通过右侧滚动条选择运行速率变化幅度，从 1%~20%，当用户操作按键，状态栏运行速率随之变化。
强制输出	可通过右侧下拉条选择 DO 信号，通过按键可将 DO 状态置反。

R+Fn6

示教器屏幕校准。

当示教器屏幕点击位置不准确时，可通过该功能对示教器屏幕进行校准，校准后会自动重启示教器和控制器。

如果碰到屏幕点击无反应的情况，则连续两次点击 R+Fn6，即可直接进入屏幕校准界面。

使用限制

强制输出不允许系统 DO，只允许普通 DO。

6.9 开始编程

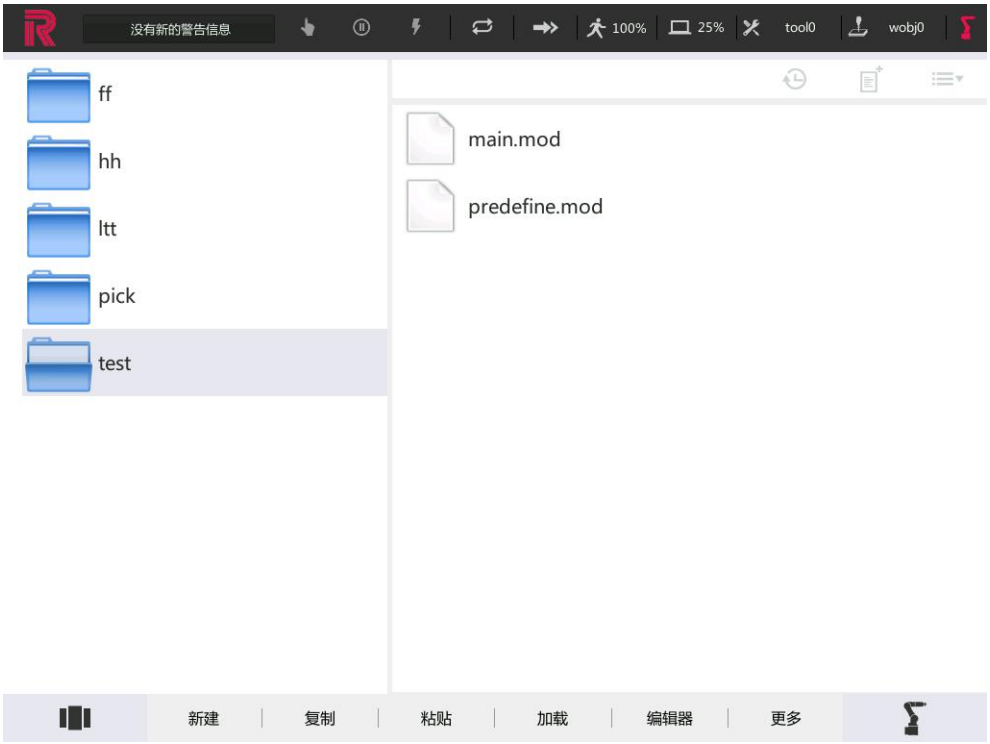
6.9.1 管理程序文件

说明

所有机器人程序文件按照“工程”的概念组织起来，每一个工程中包含若干程序模块文件，用户编写的机器人语言代码就存储在这些程序模块文件中，更多信息请参考程序结构。
所有文件存储在控制器的硬盘上，可以通过示教器进行查看和编辑。

资源管理器

资源管理器是为方便用户管理项目文件而设计，采用类似 WINDOWS 风格资源管理器风格的设计，左侧展示当前控制器上已经存在的所有工程（一个文件夹即一个工程），右侧展示选中工程内的程序模块文件。



加载与查看

在一个工程中的程序可以被运行之前，必须要先加载该工程。选中期望的工程文件夹后，点击资源管理器下方的“加载”按钮即可完成加载。只有被加载工程内包含的文件才可以编辑。
未加载工程内包含的文件可以使用只读的“查看”功能打开，用于查看其内容，但是不能修改，也不能运行。

程序文件操作

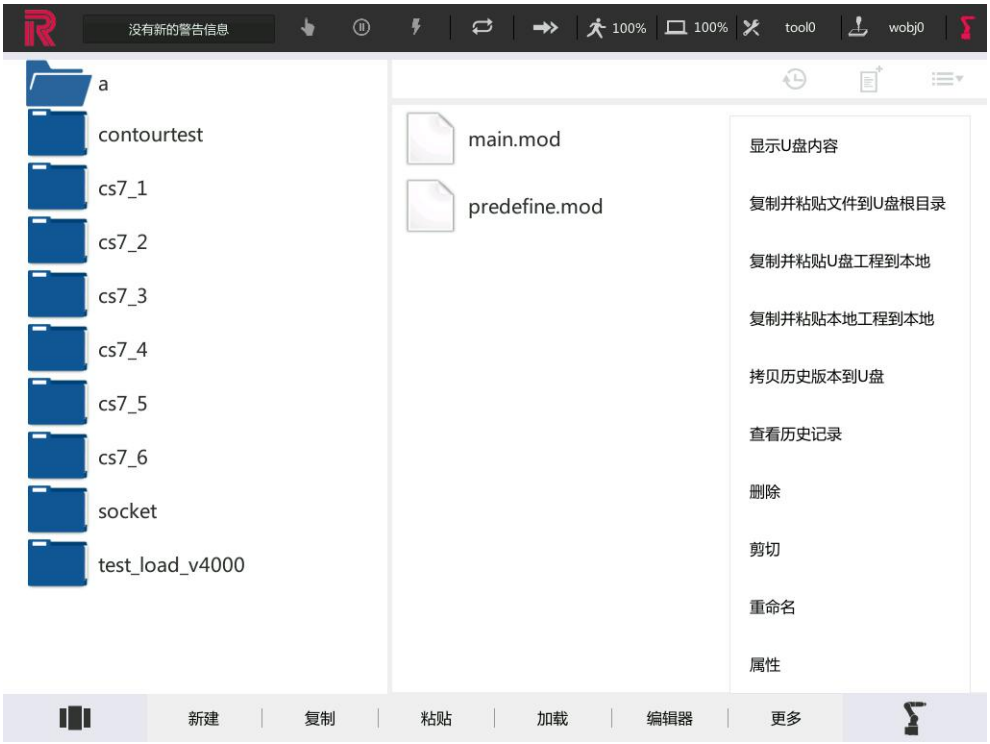
示教器上配备的资源管理器使用方法与 Windows 系统中的资源管理器非常类似，支持常见的新建、删除、重命名、复制、粘贴、剪切等。
由于程序文件都存储在控制器上，因此更换示教器不会造成程序文件丢失。

如果需要在不同机器人间拷贝程序，请使用 U 盘，示教器上提供了标准 USB 接口。

> 提示
>
> 在每个工程文件夹中，都有一个自动生成的与当前工程对应的 predefine.mod，里面存储了一些必要的系统变量，该文件仅支持查看操作，不可修改和删除。

历史版本找回

示教器会定期（每隔 10 分钟）备份工程数据，以防丢失。当用户工程数据丢失时，可通过资源管理器-更多-拷贝历史版本到 U 盘功能找回。



6.9.2 创建一个工程

说明

创建一个可用工程的基本步骤如下。

	操作	说明
1	在资源管理器界面点击下方的“新建“按钮选择“工程”，输入想要的工程名称，点击确定。	工程名仅支持字母、数字和下划线，且不能以数字开头。
2	工程创建完毕后，系统会自动生成 main.mod 和 predefine.mod，打开 main.mod 后可以进行编辑。	只有被加载的工程才能运行。
3	使用键盘或者辅助编程功能插入想要的机器人指令。	有关插入指令的更多内容，请参考编写程序。
4	在手动模式下试运行程序，并进行必要的调整。	有关程序调试的更多内容，请参考程序调试。

5	调试无误后，就可以切换到自动模式运行了。	有关自动模式的更多内容，请参考自动模式。

6.9.3 编写程序

编写机器人程序即编写机器人指令组合的过程，Titanite 系统支持有两种方式。

一种是直接键盘输入，适用于长度较短且参数较少的指令，为避免出现错误，只建议熟悉 Titanite 系统的人员使用。

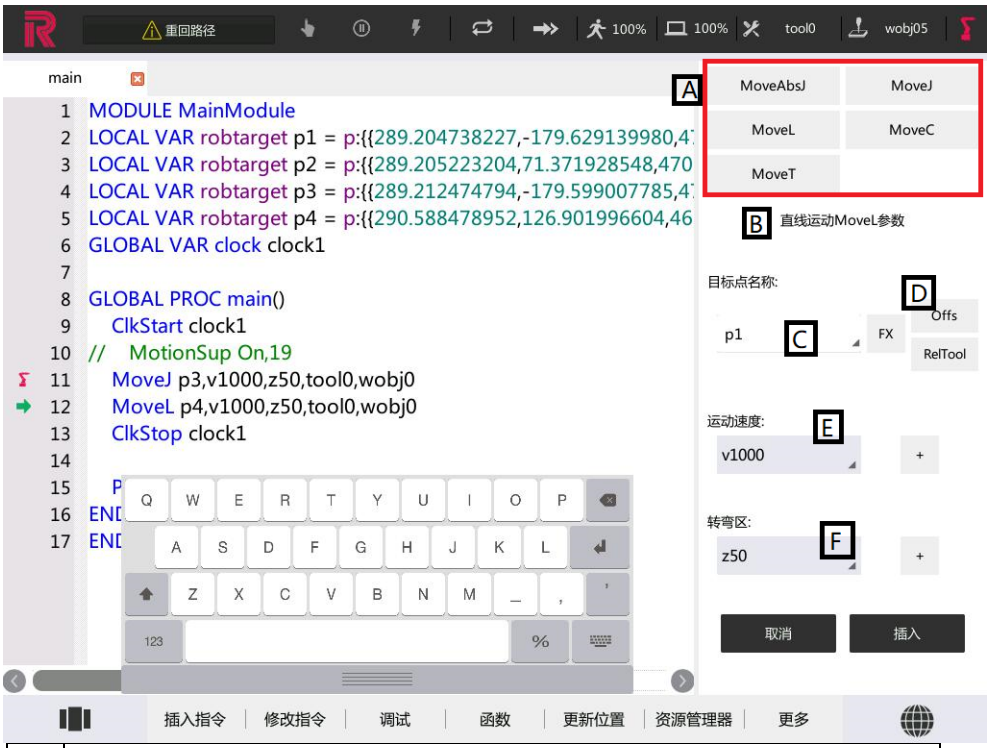
另一种是使用示教器提供的“插入指令”界面来快速编写指令，适用于所有的 RL 语言指令，推荐所有人使用。

6.9.3.1 插入指令

辅助编程

为缩短编程时间，提高编程效率，Titanite 控制系统针对绝大部分 RL 指令设计了专门的编辑界面，我们称之为“辅助编程”，在程序编辑器界面点击**插入指令**按钮即可弹出辅助编程界面，位于示教器的右侧。

下面以直线指令 MoveL 为例，对辅助编程页面的参数和操作方法进行说明。



A	**同类指令快速选择区**，用来显示与当前指令属于同一类的指令。例如在插入运动指令时，此处显示所有的运动指令按钮；插入网络指令时，显示所有与 Socket 有关的指令，方便用户快速选择切换。
B	将要插入的指令名称。
C	**目标点名称**，以机器人的当前位姿作为目标点，系统会自动为该点设置默认名称，用户也可以不使用系统指定的名称而选择输入自定义的名称。 当然也可以使用下拉框选择之前示教的点作为新指令的目标点。
D	**对目标点操作的函数按钮**，包括 Offs 和 RelTool 两个函数，目前仅对已有的笛卡

	尔空间点有效。请注意，默认情况下**目标点名称**下拉框内显示的是待插入的新点名称，实际上在控制器中还不存在该点，因此 FX 按钮不会出现，必须要使用下拉框选择一个之前示教的点之后 Fx 才会出现。
E	**指令运动速度**，使用下拉条来选择新指令的期望速度，有关运动速度的更多信息请参考 speed 变量介绍。
F	**转弯区大小**，使用下拉条来选择新指令与后一个运动指令之间的转弯区大小，有关转弯区的更多信息请参考 zone 变量介绍。

根据实际需要设置完所有参数后，点击确认，将在光标所在行插入该条指令。如果光标所在行已经有一条指令，那么系统会自动换行在下一行插入。

由于非运动指令的参数较少且在辅助编程界面均有使用方法提示，此处不再单独介绍，更多内容请参考 RL 编程语言。

> 提示
>
> 虽然在辅助编程界面针对每一条指令都有简要提示，但受限于显示区域大小，无法描述全部细节，因此在开始编程之前，强烈推荐首先阅读 RL 编程语言章节以便熟悉编程指令的用法。

6.9.3.2 修改指令

说明

如果需要对已经编写的指令进行修改，可以调出软键盘直接修改，也可以使用**修改指令**按钮重新调出辅助编程界面来进行修改。

使用键盘修改

RL 程序使用文本存储，因此可以直接使用软键盘进行修改。

对于指令的简单修改，可以直接使用键盘来进行，包括删除字符、输入字符、删除某行等。

> 提示
>
> 使用键盘进行指令编写和修改，在程序正式运行前系统不会检测语法是否正确，因此请注意检查输入的内容是否符合语法规范，以免造成未知错误。

修改指令

Titanite 系统针对 RL 指令提供了方便的辅助修改功能，将光标放置在需要修改的指令所在行，点击**修改指令**按钮，系统会自动将光标所在行指令的参数显示在辅助编程页面，用户可以直接在界面上修改，操作方法与插入新指令一致。

被注释的指令不能使用修改指令功能。

部分特殊格式的指令不能使用修改指令功能。

更新程序路径点位置

程序在编写和调试过程中，经常会需要调整某些路径点的位置，Titanite 系统提供了两种方式来修改程序的路径点。

1. 在变量管理界面找到对应的目标点变量，直接修改变量内的值。使用该方法需要对待修改的变量名称以及增加的数值偏移量比较熟悉，否则容易改错。
2. 使用程序编辑器的**更新位置**按钮，操作步骤见下表。

	操作	说明
1	将光标放置在需要修改位置的指令所在行。	仅支持运动指令，且该指令不能处于注释状态。
2	Jog 机器人到期望的新位置。	
3	点击**更新位置**按钮，系统会自动将当前位置信息更新到所选指令中。	

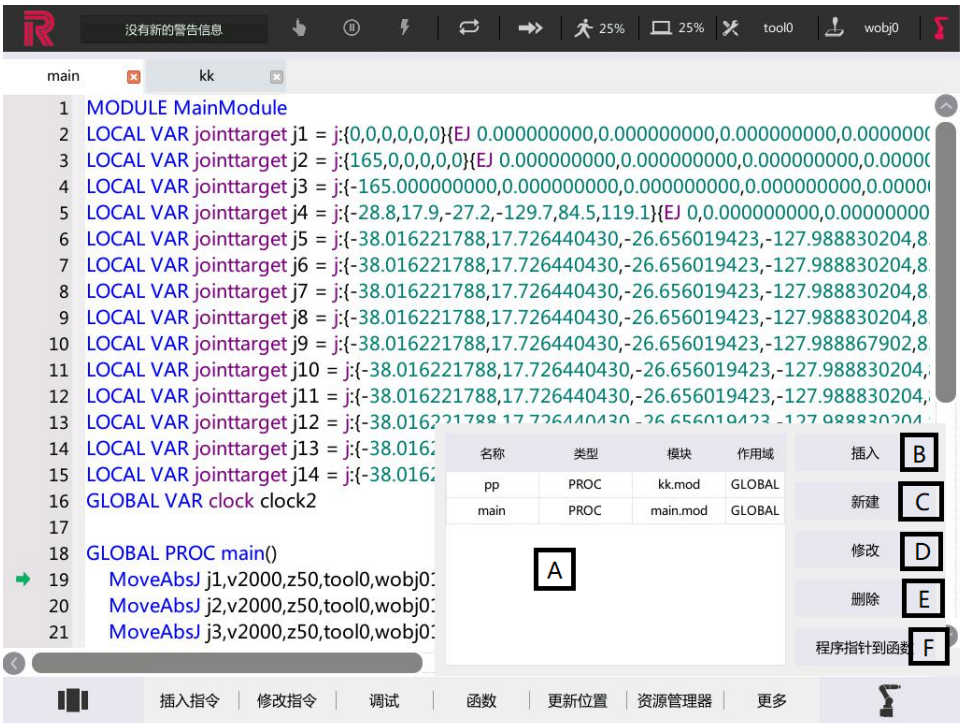
6.9.3.3 函数操作

说明

用户可以将某些代码定义为函数，以便实现代码复用，提高编程效率，简化代码结构。Titanite 系统提供了完善的图形化函数操作界面，包括但不限于新建、删除、修改、插入、前瞻等函数操作。

函数可以放在同一个模块中，也可以放在不同的模块中。

函数菜单



A	**函数列表**，显示当前已加载工程中所有函数，并给出每个函数所属的模块名称。
B	**插入**，将当前选中的函数插入到当前光标所在处，用于函数调用。
C	**新建**，创建新的函数，需要输入函数名称以及函数的作用域。
D	**修改**，修改函数的作用域。
E	**删除**，删除当前选中的函数。

F	**程序指针到函数**，移动程序指针到选中的函数首行，之后可以直接运行该函数内包含的代码。

注意事项

1. 在同一程序模块中，不允许任何变量(GLOBAL，LOCAL 级别，不包括 ROUTINE 级别)与本模块内的函数发生命名冲突；
2. 在不同的模块中，不允许任何 GLOBAL 级别的函数与变量发生命名冲突；

6.9.4 视觉

6.9.4.1 概述

说明

在很多应用场合，机器人需要搭配工业相机来实现更多功能，本节将介绍如何配置 Titanite 系统与相机之间的连接，以及如何使用 RL 语言编写视觉运动程序。

支持的相机种类

由于市面上主流的工业相机均支持使用网络的方式进行通讯，因此可以使用 RL 语言提供的 Socket 指令编写程序与相机交换数据，例如触发拍照指令或者从相机获取目标位置等。换句话说，只要您选择的工业相机**支持 TCP/IP 通信方式**，那么就可以和 Titanite 系统进行集成，编写视觉应用程序。

安全事项

请注意，启用视觉系统后，机器人会按照相机提供的数据进行移动，因此其运动轨迹可能难以预测。

即使机器人处于静止状态，也必须总是假设视觉系统处于活动状态，机器人随时可能根据视觉系统的指令开始移动。

警告

进入机器人工作区域之前，操作人员必须遵守以下步骤以防止突然移动的机器人对人身造成伤害：

- 机器人应处于手动工作模式，以防止机器人被视觉系统或者外部信号意外启动；
- 任何进入机器人工作区域的人员必须随身携带示教器；
- 编程和调试时，如果暂时不需要机器人进行移动，以尽快松开使能开关。

使用工业相机的一般步骤

下面的表格中列出了在 Titanite 系统中使用工业相机的一般步骤。

序号	操作	说明
1	按照相机的说明书对相机进行连接和基本设置，需保证相机工作在 TCP/IP 通信模式下。	包括设置期望的图像特征，交互命令字等，不同品牌相机的设置过程不尽相同，请咨询您的相机供应商获取更多信息。
2	使用网线将相机连接至控制柜的视觉用网口上。	关于控制柜网络接口的更多信息请参考按钮与接口。

		为保证通信质量，请使用超五类（Cat.5e）或以上级别的网络电缆。
3	对相机进行标定。	获取像素点与实际距离的对应关系，更多信息请参考相机标定。
4	执行相机到机器人的标定。	确定相机坐标系与用户坐标系的关系，更多信息请参考相机到机器人的标定。
5	编写 Socket 程序控制相机拍摄，并从相机读取期望特征的位置，然后将获取的位置信息用于运动指令中。	更多信息请参考视觉编程。

6.9.4.2 视觉标定

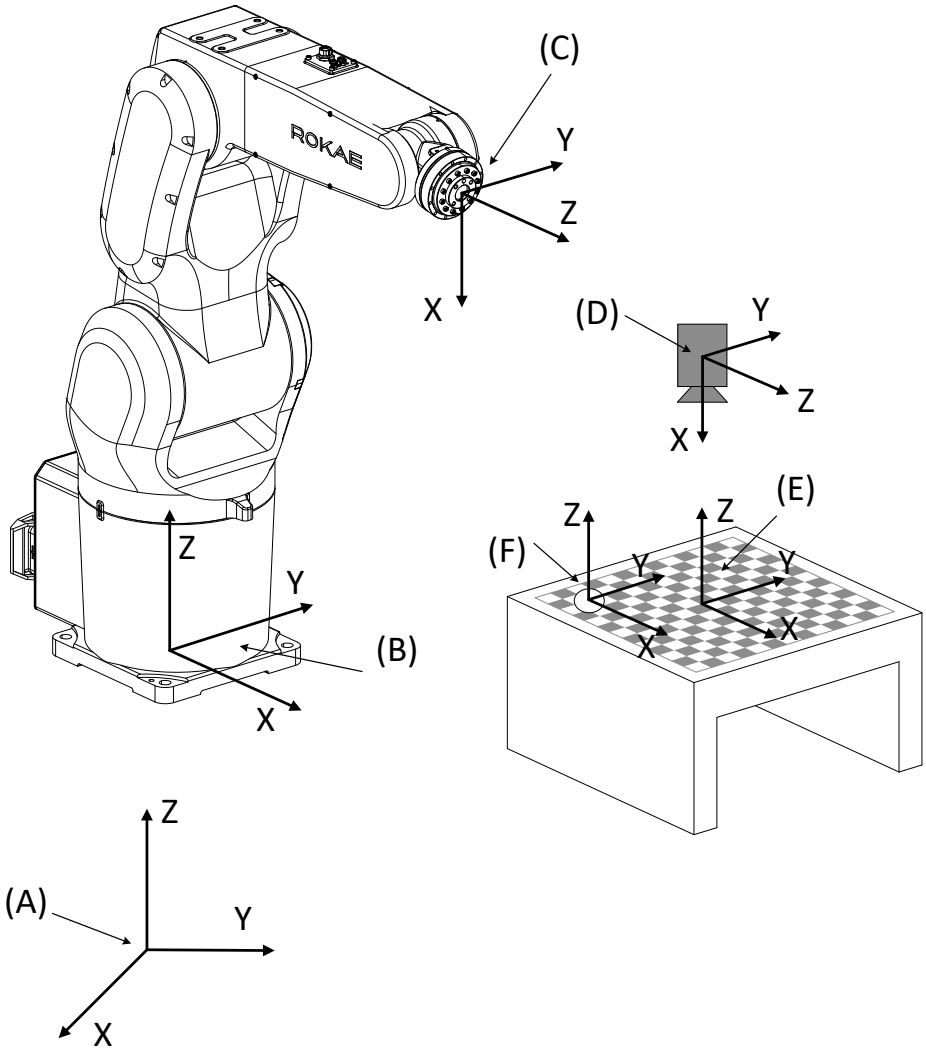
说明

大部分使用视觉的机器人应用都是从相机获取目标位置信息，然后用于引动机器人运动。这要求相机可以从其拍摄的图像中获取机器人可用的坐标位置，在这些坐标信息可以被 RL 程序正确使用之前需要首先对相机进行配置，其过程主要分成两个部分：

1. 相机标定，确定像素点与空间距离的对应关系。
2. 相机到机器人的标定，把相机坐标系和机器人的工件坐标系关联起来。

视觉应用中的坐标系统

视觉系统一个主要的作用是为机器人的运动指令提供目标位置信息，常见的安装方式如下图所示：

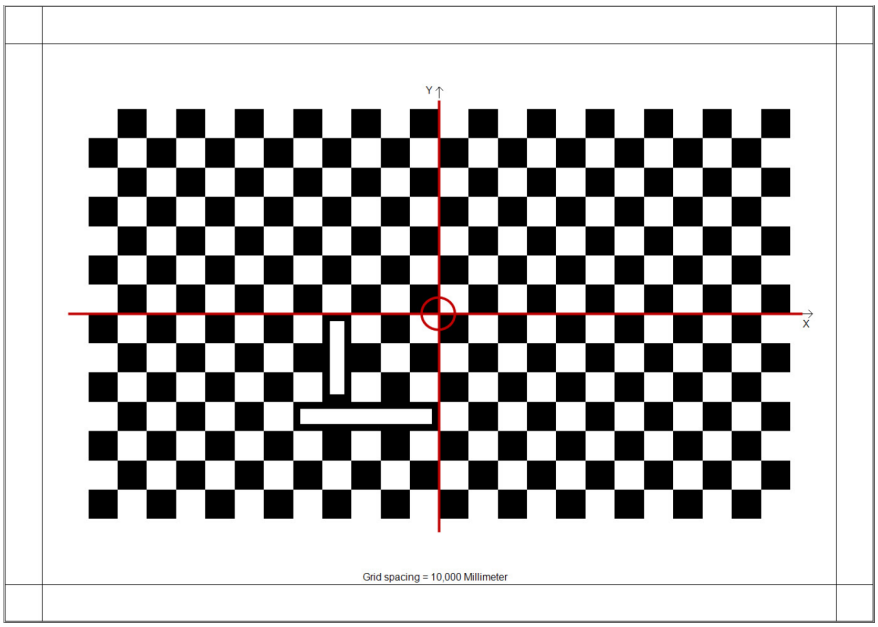


A	世界坐标系。
B	基坐标系。
C	法兰坐标系。
D	相机安装的位置，该位置不影响机器人的运动控制，因此无需设置，此处仅做示意。
E	用户坐标系，与相机坐标系重合，有关相机坐标系的更多信息请参考相机到机器人的标定。
F	工件坐标系，用来定位实际工件的位置。

相机标定

相机标定又称相机校准（Camera Calibration），指的是计算像素坐标系与物理空间坐标系之间的对应关系，通俗来讲就是告诉相机其拍摄的照片上两个距离 n 个像素点位置换算到实际中距离是多少毫米，之后相机就可以把像素信息转换成距离/位置信息发送给机器人控制器。

最常见的相机标定方法是采用棋盘格板（Checkerboard），如下图所示，主流的工业相机都支持使用这种方法对相机进行标定，具体步骤请查阅您所选用相机的操作手册。



此外，如果对精度要求不高，也可以不使用棋盘格来标定，请咨询您的相机供应商或者查阅相机的说明手册获取更多信息。

> 提示
>
> 相机标定是对相机内部参数进行设置的过程，一般通过相机提供的软件进行（不属于机器人控制器功能范围），因此不同品牌的相机有不同的设置流程，请参阅相机供应商提供的相关资料。

相机到机器人的标定

相机完成标定后，可以输出其视场范围内任意两个特征点在 XY 两个方向上的准确距离，有的相机还可以输出绕 Z 轴的旋转角度 R。通常情况下这些相机返回的距离/角度可以看做是位置或者坐标系的偏移量，因此接下来只要告诉机器人控制器这些偏移量是相对哪个坐标系定义的，就可以在 RL 程序中正常使用了。

由于相机输出的这些偏移量是在相机坐标系下定义的（通过相机标定过程来配置），因此确定所需参考系最简单的方式就是让机器人的某个工件坐标系与相机坐标系重合，这样在 RL 程序中可以不用任何转换直接使用这些偏移量来引导机器人的运动。

这个把相机坐标系与机器人工件坐标系建立联系的过程称为相机到机器人的标定，在 Titanite 系统中，常用的做法是通过标定一个与相机坐标系重合的工件坐标系来实现这一过程。

> 提示
>
> 对于精度要求不高的场合，很多工业相机也提供了通过在视场中采集若干个（一般为 3~30）点来进行快速标定的功能，本质上这项功能是把相机标定和相机到机器人的标定合二为一并全都放在了相机内进行，使用默认的工件坐标系（即世界坐标系）计算偏移量。
>
> 请根据现场情况选择合适的标定方法，Titanite 系统均可提供良好的支持。

6.9.4.3 视觉编程

说明

本节将使用一个视觉应用的实例来演示如何使用 RL 编写视觉程序。

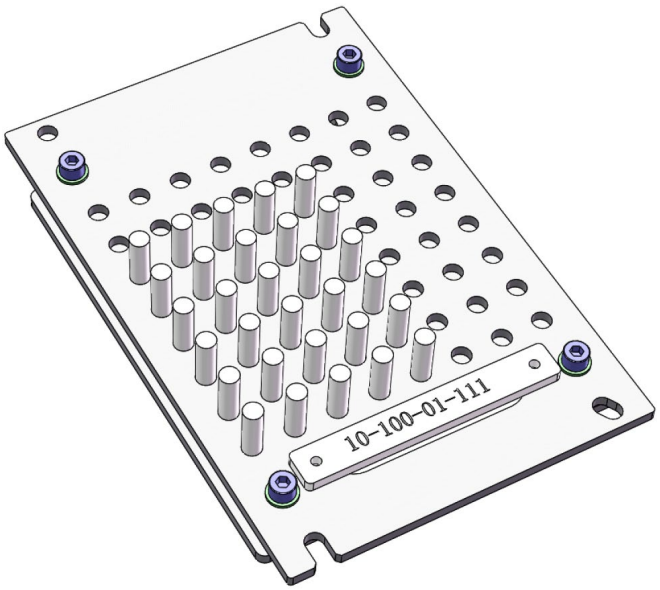
应用背景介绍

本实例中使用的工业相机为 Panasonic ANPVC1210，您可以根据实际情况更换其他支持 TCP/IP 通信的工业相机。

下图表示的是一个放置机床刀具毛坯的托盘，机器人需要从该托盘上拾取毛坯然后送往加工中心进行处理，最后将处理完毕的刀具插回到另外的成品托盘上。

有关托盘的一些数据如下：

- 不论刀具毛坯的粗细和长短，每个托盘上固定有 8*8 = 64 个孔位；
- 每两个孔位圆心之间的距离都是 23 毫米；



在本例中使用相机主要有两个目的：

1. 识别刀盘上的编号来确定不同刀具的长度和直径，以便调用对应的 RL 程序；
2. 获取刀具毛坯的位置和托盘空闲孔的位置，以便准确的拾取和放置工件；

由于拾取和放置的程序是类似的，因此我们单独针对拾取毛坯来进行实例介绍。

对于这样一个应用，可采用的方案有两种：

1. 在刀盘上设置一个基准点，每次抓取前由相机获取目标刀具毛坯相对于基准点的偏移量，然后控制机器人按照偏移量移动到目标位置抓取；
2. 编写程序让机器人每次从工件坐标系的原点位置抓取毛坯，然后在应用过程中每次所用的空间坐标系定义更新到相机返回的位置上，即每次抓取之前首先通过相机返回的数据更新工件坐标系。

在下面的实例中，我们采用第一种方案。

RL 程序示例

以下是 RL 代码实例，为了提升代码可读性，隐藏了一部分变量声明和简单逻辑的代码。如果需要在实际系统上运行该段代码，需要根据情况补充完整。

```
VAR int countPick=0
VAR int row=0
VAR int col=0
VAR robtarget pickOrigin    //第一个孔的圆心位置，用作整个程序的参考点
VAR robtarget pickTarget
VAR bool TrayStatus[64]    //标识孔位是否有刀具毛坯，共 64 个 bit 对应 64 个孔，True 代表有
VAR signaldi isCNCReady    //从机床获取信息，以判断机床就绪
VAR signaldo gripClose    //控制手爪的 DO 信号
VAR triggData triGC
//以下是主程序
PROC main()
MoveAbsJ j1,v200,fine,tool02,cutter208    //移动到起始位置等待
SocketCreate("192.168.0.5",8601,s0)    //与相机的发送端口建立连接，用于接收相机数据
SocketCreate("192.168.0.5",8604,s1)    //与相机的数据接收端口建立连接，用于给相机发送指令
Wait 0.5    //等待相机
//等待 CNC 准备好
IF(isCNCReady == TRUE)
  //给相机发送程序指令调用 X003 号程序，请注意不同品牌相机调用的程序或者指令不同!
SocketSendString("%X003**",s1)
SocketSendByte(13,s1)    //发送 ACSII 码“13”，代表回车
SocketSendString("%S**",s1)    //相机指定的拍摄指令
SocketSendByte(13,s1)
Wait 0.5    //等待相机准备数据
TrayStatus =SocketReadBit(64,s0)    //从相机获取 1 个刀盘共 64 个孔位的占用信息
WHILE (countPick < 64)
countPick++
IF(TrayStatus [countPick] == TRUE)
row=( countPick -1)/8    //获取目标抓取位置相对于参考位置相差多少行
col= countPick - 8*row – 1    //获取目标抓取位置相对于参考位置相差多少列
pickTarget.X= pickOrigin.X + col*23
pickTarget.Y= pickOrigin.Y + row*23
MoveL Offs(pickTarget,0,0,80), v300, fine, tool02, cutter208 //移动到目标夹持位置的上方
TriggIO triGC,5,gripClose,True //设置运动中 IO 触发条件，在距离终点 5mm 处关闭手爪
TriggL pickTarget, v100, triGC, fine, tool02, cutter208    //关闭气爪，夹住毛坯件
MoveL Offs(pickTarget,0,0,100), v200, fine, tool02, cutter208    //向上移动拿起工件
MoveL p3, v300, fine, tool02, cutter208
Wait 0.2
```

```
.....    //继续其他与机床协同的非视觉相关操作
Endif
Endwhile
Endif
ENDPROC
```

6.9.4.4 提高视觉精度

说明

本节将介绍影响视觉精度的几个重要因素，以及如何优化这些因素以提高视觉应用的整体精度。

选取特征与设置照明

使用相机获取位置之前，我们首先需要在零件上寻找合适的特征，选取这些特征的时候需要平衡可用性和一致性两个因素。特征的一致性尤为重要，所选取的特征在其他相同零件上是否表现一致，以及工件移动时是否仍然可以被相机所正确捕获都需要考虑。

影响特征一致性的主要因素是照明条件，照明不足时会导致相机识别率降低，在零件摆放位置或姿态发生变化时可能无法准备识别。

因此，成功应用视觉系统的先决条件是在工件上找到合适的特征并为其提供充足的照明条件，以便在指定的使用环境下获得持续的高识别率。

确定视场

一般情况下，视场（The field of view）大小取决于相机与被拍摄工件的距离和相对角度。当相机的像素值固定时，视场越大，图像的精度也就越低。提高精度的简单方法是采用更高像素的相机或者减小视场的范围。

在多数情况下没有必要把整个工件都放置在视场中来进行图像识别或者测量。因为我们只关心想要识别或者测量的零件特征，只要保证这些特征可以清晰且持续的可以被拍摄到即可。使用较小的视场不仅可以提高视觉精度（每毫米对应的像素变多），还可以降低对照明设备的要求（需要提供照明的范围变小了）。

相机拍摄工件的角度也会影响最终的精度，因此一般来讲，在现场环境允许的情况下应尽可能的让相机从垂直于所选特征的角度拍摄工件。如果现场环境不允许，那么应使用相机提供的算法来补偿因为视角不垂直引起的误差。

6.10 程序调试

6.10.1 调试功能菜单

说明

为方便程序调试，Titanite 系统在程序编辑器界面提供了若干强大的调试功能。

菜单功能

程序指针到 Main	移动程序指针到 Main 函数，相当于程序复位。
移动程序指针	把程序指针移动到光标所在行，更多信息见移动程序指针。
检查程序	用于检查当前程序中是否存在特定明显错误，例如函数重名、关键标识符缺失。不能检查出所有语法错误，执行程序指针到 Main 也可以达到检查程序的效果。
切换 MoveL/J	用于将当前语句快速的在 MoveL 和 MoveJ 之间切换。
剪切整行	剪切光标所在当前行。
粘贴整行	将剪切到的整行内容，插入光标所在下一行。

6.10.2 运行指针

说明

运行指针位于编辑器的左侧，用来显示 RL 程序的执行状态。

程序指针

绿色小箭头即程序指针（Program Pointer，简称 PP），指向编译器已经前瞻编译完成的语

	句。根据编程轨迹的执行时间长短不同，程序指针和运动指针可能指向同一行，也可能指向不同的行。 有关前瞻机制的相关信息，可参考前瞻机制。
运动指针	
	红色的小机器人是运动指针（Motion Pointer，简称 MP），指向当前机器人正在执行的运动语句。
光标	
	光标是指输入提示符“|”，光标所在的行会高亮显示。 请注意，绝大部分与程序文本有关的操作，例如修改指令、更新位置、复制、剪切、注释行等，都是针对光标所在行操作的。

6.10.3 前瞻机制

说明

前瞻（Lookahead）是指在机器人运动过程中，控制系统提前处理当前机器人正在执行指令之后的程序指令，引入前瞻机制有如下好处：

- 可以获得前方轨迹的速度、加速度要求以及机器人本身的限制条件信息，以便规划性能最优的控制策略；
- 根据编程的转弯区设置规划转弯区轨迹；
- 获取靠近软限位/边界、靠近奇异点等异常状态，以便提前进行处理；

前瞻机制无法手动关闭，系统在运行程序时会自动进行前瞻，可以使用程序指针（Program Pointer）来查看前瞻的位置。

有些 RL 指令会打断前瞻，编译器遇到此类指令后会停止继续编译，直到机器人把对应的指令执行完毕后才会继续编译，会造成前瞻停止的指令有 Wait，RETURN，EXIT，Print，Pause，CRobT，SocketCreate，SocketClose，SocketSendByte，SocketSendInt，SocketSendString，SocketReadString，SocketReadBit，SocketReadInt，SocketReadDouble 等。

6.10.4 移动程序指针

说明

如果需要从程序中间的某一行开始执行之后的程序，那么可以使用该功能将程序指针移动到光标所在行，之后程序即可从新的位置开始执行。

	操作	说明
1	暂停正在运行的程序，点击屏幕将光标移动到期望的行上。	
2	在程序编辑器界面，点击“调试”按钮，选择“程序指针到光标”。	
3	程序指针 PP 将移动到选中的行上。	
4	程序指针 PP 指向目标行后，点击程序开始或者下一步，机器人会从当前位置以关节插补方式缓慢的运动到指定行的目标位置，执行过程与“重回路径”过程一致。	有关重回路径功能的详细介绍，请参考重回路径。
5	重回路径完成后，机器人将会停止在指定行的目标点位置，再次点击“开始”，机器人才会继续执行之后的程序。	

使用限制

移动程序指针存在如下限制：

- 使用该功能时，以下指令会被忽略，编译器的编译位置会直接移动到目标行，除此之

外，其他所有指令都会被执行：

a) 所有的运动指令；

b) SetDO、SetGO、Return、Wait、Print 以及所有的 Socket 指令；

c) 函数调用行；

- 移动程序指针时，忽略流程控制指令的判断条件。
- 不能跨函数移动程序指针，需要先使用“程序指针到函数“功能将程序指针移动到某个函数的开头，再使用移动程序指针功能；
- 移动程序指针只能移动到运动指令行。

6.10.5 单步运行

说明

单步运行主要用来做程序调试，机器人可以每次仅执行一条运动指令并停止在该条指令的目标点上，便于确认每一个示教点是否符合要求。
此外在单步运行时，机器人可以沿着编程的路径反向运行，方便用户在相邻的示教点之间进行快速跳转。

单步运行

单步运行有两种，按照程序执行方向不同分为正向单步和逆向单步。
正向单步又称“下一步”，即让机器人沿路径正常执行一条指令，并停止在当前指令的终点上。“下一步”按钮的作用与“启动”按钮非常类似，最大的区别就是使用下一步按钮启动的程序运行到指令的终点后会自动停止。
逆向单步又称“上一步”，即让机器人沿路径往回走一条指令，并停止在当前指令的**起点**上。
单步运行期间可随时使用“停止”按钮来停止机器人的运行，再次点击启动或者单步运行按钮程序会继续沿选定的方向运行。

使用限制

由于逆向单步仅用于在某个程序段对示教点进行检查或者调试，因此相比正向单步有如下使用上的限制：

1. 逆向运行时系统将只会执行运动指令；
2. 逆向运行时无法越过逻辑指令，例如 WHILE、IF、Wait 等，遇到这些指令将无法继续逆向运行；
3. 当上下两条指令使用的工具或者工件不相同时，无法逆向运行；
4. 逆向运行到程序首行时，无法再继续逆向运行；
5. 正向运行与逆向运行之间切换时，需要点击两次“上一步”或者“下一步”。第一次点击用来改变程序指针的运动方向，第二次点击后机器人才会开始运动。

6.10.6 重回路径

说明

在某些特定情况下，机器人的位置会偏离其编程的路径，例如：

- 在程序运行中被停止的期间（程序复位造成的程序停止不在此列），机器人被 Jog 到

其他位置；

- 程序运行期时触发紧急停止，机器人执行 STOP 0 后；

当程序再次从停止位置启动时，如果系统检测到机器人已经偏离编程路径，那么机器人会首先执行重回路径（Regain Path）运动以返回到原来的编程路径上。

为保证安全，重回路径时机器人的运动速度比较慢，并且可随时按示教器上的“停止”按钮来停止机器人的运动。

使用限制

重回路径时机器人执行的是关节轨迹，因此末端路径无法预测，请注意观察是否会与周边环境发生碰撞。

只有当机器人从程序中间停止的地方继续执行时，控制系统才会检测是否偏离路径，如果偏离则会执行重回路径操作。

如果程序被复位，那么系统将不会检测是否偏离路径，而是直接从第一行开始执行，请注意防范可能发生的碰撞。

6.10.7 变量管理

说明

变量管理界面提供了对机器人系统内几乎所有变量的新建、查看、修改及删除操作，目前支持的变量类型包括：

序号	变量类型	描述
1	系统预定义变量	指用户不可修改的变量，用于存储某些系统参数，例如 tool0/wobj0 等。
2	用户预定义变量	用户可以进行修改，并且可以在多个程序中进行使用的变量，例如用户标定的工具、工件等。
3	程序变量	用户在程序中进行定义的变量，一般仅在当前程序及其子程序中使用，包含了绝大部分系统支持的变量类型。

对于一些某些有专门定义步骤的变量类型，例如 tool/wobj（使用标定界面进行定义和修改），robtarget/jointtarget/speed/zone（使用辅助编程界面进行定义和修改），虽然也可在变量查看界面进行查看和修改，但是出于方便性和减少错误的考虑，仍然推荐使用专用界面进行修改，在变量管理界面只进行查看操作。

提示

可以在变量管理界面查看和修改的变量仅限于当前所加载的机器人程序中使用的变量，因此加载其他程序后显示的变量会发生变化。

变量编辑

如果需要新增变量或者对某个存在的变量进行修改，可通过点击“新建”或者“修改”按钮进入变量编辑页面进行操作。

变量类型	在新建变量时用于选择变量的类型，所有支持的类型都列在左侧边栏内。
变量名称	将要插入变量的名称。
数组维数	用于创建或者修改数组，最大支持 3 维数组。需要注意的时，数组维数过高时系统会花费较多时间进行数据处理，HMI 响应可能会变慢。
模块名称	默认为 main.mod，也可以选择存储在其他 mod 中。
存储类型	可选择 const、pers 和 var，更多信息请参考变量声明。
作用域	可选择 global 和 local，更多信息请参考变量声明。

6.10.8 机器人的奇异性

说明

正常情况下，机器人最多可以使用 8 种不同的关节配置到达同一个工作空间中的位姿，详细信息可以参考 confdata 变量介绍。

但是在机器人的工作空间中还存在若干特殊的位姿，机器人可以使用无数种不同的关节配置到达，这些位姿被称为奇异点。奇异点会导致控制系统在基于空间位置计算关节角度时出现问题。

一般来讲，机器人有三种类型的奇异点：

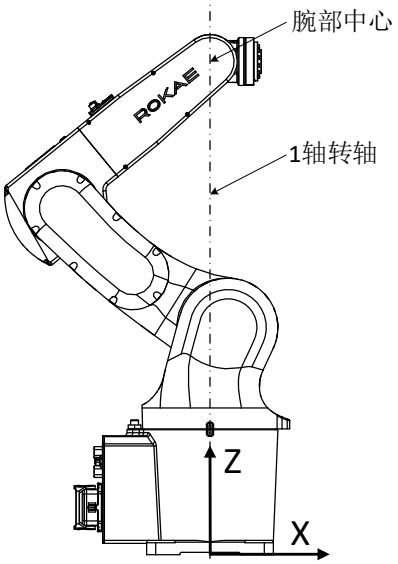
1. 肩部奇异点
2. 腕部奇异点
3. 肘部奇异点

机器人执行关节运动时，不存在奇异性问题。

当机器人执行靠近奇异点的笛卡尔空间轨迹时，某些关节的速度可能会非常快，为了保证不超过关节最大速度，末端轨迹的速度将会被自动降低。

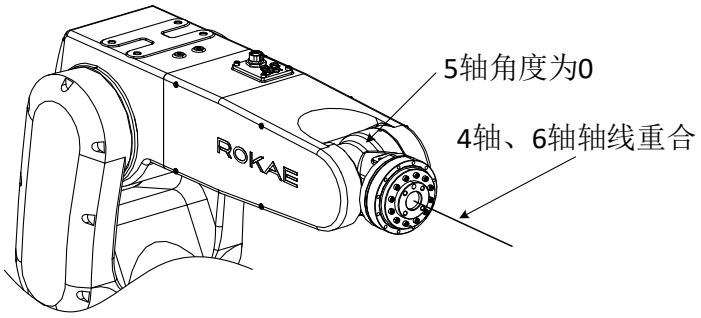
肩部奇异点

当腕心位于 1 轴轴线上时，机器人处于肩部奇异点上：



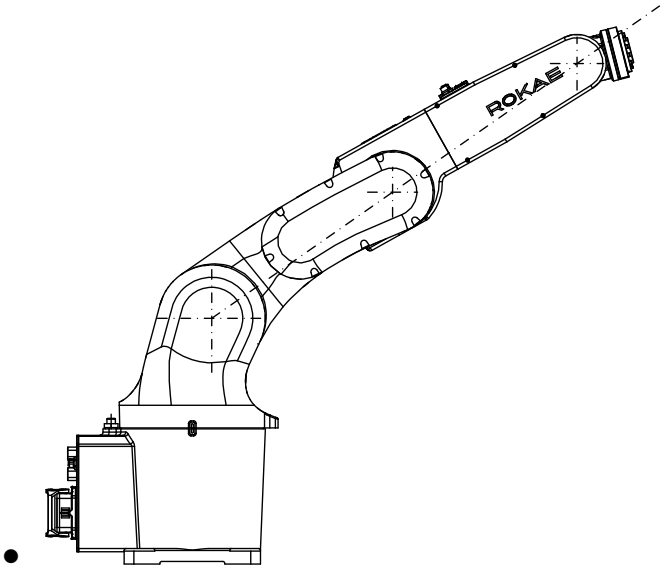
腕部奇异点

当 4 轴和 6 轴轴线重合（5 轴角度为 0）时，机器人处于腕部奇异点上：



肘部奇异点

当腕心、2 轴转轴、3 轴转轴位于一条直线上时，机器人处于边界奇异点上：



7 RL 编程语言

7.1 关于 RL 语言

概述

工业机器人是一种适用于多种场合的可编程设备，用来给机器人编写任务程序的语言被称为机器人语言（Robot Language），Titanite 系统使用 RL 语言作为 ROKAE 机器人的编程语言。

RL 语言为 ROKAE Robot Language 的缩写，即 ROKAE 机器人编程语言，用户可以通过示教器编写控制机器人的程序。

RL 语言程序文件后缀名为.mod，例如：MoveObj.mod 或 PickSomething.mod，每一个程序文件构成一个程序模块。

RL 语言的指令不区分大小写，例如对于 MoveAbsJ 、moveabsj、MOVEABSJ 三种写法，RL 均视为正确的 MoveAbsJ 指令，但是为保持统一的语言风格，建议采用单词首字母大写的格式。

示例

为了说明 RL 程序语言的特点，先看一个简单的程序，了解 ROCKL 的基本结构和格式：

```
//程序开始，首先进行变量声明
MODULE MainModule
GLOBAL VAR int counter = 3
GLOBAL VAR jointtarget home= J:{0,0,0,0,0,0}{EJ 0,0,0,0,0,0}
GLOBAL VAR robtarget startpoint= p:{{48,54.9,4.3},{0.352,0.99,0.11,-0.387}}{cfg 0,-1,2,0}{EJ 0,0,0,0,0,0}
GLOBAL VAR robtarget endpoint = p:{{48,54.9,4.3},{0.352,0.99,0.11,-0.387}}{cfg 0,-1,2,0}{EJ 0,0,0,0,0,0}
GLOBAL VAR speed v40 = v:{40,2,5,10,2}
LOCAL CONST zone z100 = z:{100,50}

//声明完毕，以下是功能指令
PROC main()
MoveAbsJ home, v40, z100, tool3
counter++
IF(counter > 6)
   MoveL startpoint, v40, z100, gripper       //移动机器人到 startpoint 点
Else
   MoveJ endpoint, v40, z100, gripper
ENDIF
```

其中：

- 整个程序分为声明区和实现区两大部分，在每个 Mod 文件中第一个函数之前的区域为声明区，例如 main.mod 中，PROC main 之前的部分为声明区；
- VAR 表示变量的存储类型，表示一个可变量；
- int、robtarget、speed、zone、tool 是 RL 语言的专用变量类型；
- MoveJ 是 RL 语言中一个标准的运动指令；
- “//”后面是注释内容；

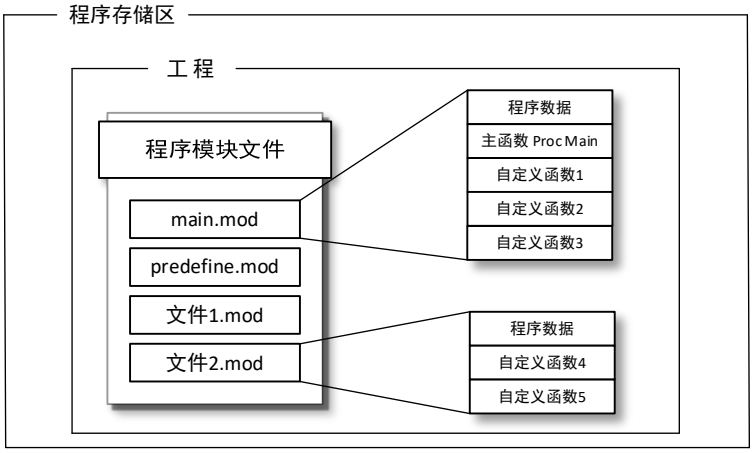
7.2 程序结构

7.2.1 概述

说明

Titanite 系统中所有的程序文件都按照“工程”的概念组合在一起，包含如下特性：

1. RL 程序根据范围大小分为三个层次：
   a) 工程，即 Program，一个工程中包含若干个程序模块文件；
   b) 程序模块，即 Module，分为程序模块（.mod）与系统模块（.sys）两种，一个程序文件就是一个模块；
   c) 函数，即 ROUTINE，用户自定义的用于重复调用的程序块；
2. 每个工程使用一个独立的文件夹存储，在资源管理器界面左侧，每一个单独的文件夹就代表一个独立的工程；
3. 一个工程中可以包含多个程序模块，但有且仅有 1 个 main.mod，在 main.mod 中包含一个 PROC main，该 PROC main 作为整个工程的入口函数；
4. RL 语言支持用户自定义函数，不同的自定义函数可以存储在同一个程序文件中，也可以存储在不同的程序文件中；
5. 工程、程序文件以及函数的相互关系如下图所示：



7.2.2 程序模块

说明

程序模块即.mod 或.sys 文件，每个程序模块中都包含了若干数据变量以及函数，用于实现具体的机器人功能，一个项目中可以包含多个程序文件。每一个程序文件都可以执行拷贝、删除等常规的文件操作。

在每个工程中，必须有一个程序模块中要包含 main 函数，用作整个工程的入口函数。加载并执行某个工程，本质上是在执行 main 函数。

程序模块分类

程序模块扩展名为.mod，每一个模块文件即为一个程序模块。

模块的定义

模块的定义方式为：

```
MODULE Name
声明区
PROC main()
…
ENDPROC
PROC test1()
…
ENDPROC
PROC test2()
…
ENDPROC
ENDMODULE
```

> 提示
>
> 每一个模块中，位于文件头部与第一个函数之前的代码区称为声明区，用来存放 GLOBAL 和 LOCAL 作用域的变量声明，该区域不允许用户直接在编辑器中修改。

7.2.3 函数

说明

使用函数功能可以简化代码结构，提高代码可读性和复用率。用户可以将需要经常执行的程序段定义成新的函数，这样在主程序中就可以方便的随时调用。

目前不支持带返回值的函数。

函数定义

函数的定义方式为：

```
SCOPE PROC RoutineName()
…
…
//do something
…
…
ENDPROC
```

其中，

1. SCOPE 为函数作用域，支持 GLOBAL 和 LOCAL 两种；
2. PROC 是函数的定义关键字；
3. RoutineName 为函数名称，命名规则与变量命名规则相同，详见变量命名规则。

函数调用

调用函数时，直接在程序编辑器输入函数名即可：

```
RoutineName()
```

仅能调用本工程内的其它 GLOBAL 级别的函数或本模块文件中 LOCAL 级别的函数，不支持递归调用，不支持两个子函数之间相互交叉调用。

对于函数的调用在编译器中被视为一条独立的程序指令。

注意事项

不允许在函数中定义函数；

7.3 变量

7.3.1 基本概念

变量命名规则

RL 语言中的变量名称可由字母，下划线，数字组成，必须以字母或者下划线“_”开头，但变量名不能与系统关键字重名，RL 系统关键字详见预定义关键字。

除此之外，还存在以下注意事项：

1. 在同一模块中，不允许出现重名的 GLOBAL、LOCAL 级别变量；
2. 在不同的模块中，不允许出现重名的 GLOBAL 变量；
3. 在不同的模块中，允许出现重名的 LOCAL 级别变量；
4. 在同一模块中，不允许任何变量（GLOBAL，LOCAL 级别，不包括 ROUTINE 级别）与本模块内的函数发生命名冲突；
5. 在不同的模块中，不允许任何 GLOBAL 级别的函数与变量发生命名冲突；

> 提示
>
> 当变量名只包含两个字符时，需要注意第二个字符不要为“h”、“b”，否则该变量可能会被转换为十六进制或者二进制，更多信息请参考进制转换。

变量作用域

RL 语言系统定义了三种作用域：

1. 全局（GLOBAL），对当前工程中的所有模块可见，可在模块声明区内声明；
2. 局部（LOCAL），仅对当前模块可见，可在模块声明区内声明；
3. 函数（ROUTINE），仅在当前函数内可见，只能在函数体内部声明，且声明该作用域变量时不允许指定作用域类型（GLOBAL 或者 LOCAL）；

> 提示
>
> 函数（ROUTINE）作用域仅适用于变量，不适用于自定义函数。

存储类型

每个变量根据其是否可以在程序执行过程中被修改分为可变量（VAR）、持续性变量（PERS）常量（CONST）。

- VAR（Variable），可变量，可以在程序运行过程中进行重新赋值的变量。
- CONST（Const Variable），常量变量，不能在程序运行过程中重新赋值的变量，此类型变量的值必须在其初始化时指定。
- PERS（Persisten Variable），持续性变量，在程序执行过程中如果此类型变量的值发生了变化，将会自动将该变量的初始值修改为当前值，由此达到“持续”存储的效果。

> 提示
>
> 即使某个 PERS 类型变量的值在程序运行过程中被更改，其在程序编辑器声明区内显示的初始值也不会立即刷新，只有当程序重新加载或程序停止时在程序编辑器声明区显示的初始值才会更新

到最新值。
不论程序运行与否，都可以在“变量管理”界面随时查看 PERS 变量的最新值。

预定义关键字

以下为 RL 语言预定义的保留关键字（不区分大小写）：
Module、EndModule、Proc、EndProc、Func、EndFunc、TRAP、ENDTRAP、SetDO、DO_ALL、SetGO、SetAO、WaitDI、Wait、WaitUntil、WaitWObj、WBID、Q、P、J、V、W、T、S、L、CA、DURA、IGNORELEFT、EJ、1J、FCBV、FCCV、FCOL、FCXYZ、FCCART、PE、PER、TCP、ORI、EXJ、CFG、PDIS、JDIS、MoveAbsJ、MoveJ、MoveL、MoveC、MoveT、LOCAL、TASK、GLOBAL、VAR、CONST、PERS、INV、DOT、CROSS、sin、cos、tan、asin、cot、acos、atan、atan2、sinh、cosh、tanh、ln、log10、pow、exp、sqrt、ceil、floor、abs、rand、GetCurPos、Print、PrintToFile、ClkRead、TestAndSet、IF 、Else、Endif、WHILE、ENDWHILE、for、from、to、endfor、Break、Continue、Del、Int、Double、Bool、String、BYTE、Robtarget、Speed、Zone、Tool、Wobj、Jointtarget、TriggData、Load、FCBoxVol、FCSphereVol、FCCylinderVol、FCXyzNum、FCCartNum、Pose、CLOCK、INTNUM、SYNCIDENT、TASKS、Call、Return、EXIT、Pause、StopMove、StartMove、StorePath、RestoPath、True、False、Interrupt、When、Offs、CalcJointT、CalcRobT、CRobT、RelTool、SocketCreate、SocketClose、SocketSendByte、SocketSendInt、SocketSendString 、 SocketReadString 、 SocketReadBit 、 SocketReadInt 、 SocketReadDouble、AccSet、MotionSup、TriggIO、TriggJ、TriggL、TriggC、On、Off、clock 、 intnum 、 userframe 、 pinf 、 ninf 、 FCFRAME_WORLD 、 FCFRAME_TOOL 、 FCFRAME_WOBJ 、 FCFRAME_PATH 、 FCPLANE_XY 、 FCPLANE_XZ 、 FCPLANE_YZ 、 FC_LINE_X、FC_LINE_Y、FC_LINE_Z、FC_ROT_X、FC_ROT_Y、FC_ROT_Z、Offs、CalcJoinT、CalcRobT、CRobT、RelTool、\\Start、\\Time、ClkReset、ClkStart、ClkStop、CONNECT、WITH、IDisable、IEnable、ISignalDI、\\Single、\\SingleSafe、WaitWobj、DropWobj、WobjIdentifier 、 WobjAngle 、 ActUnit 、 DeactUnit 、 INTNO 、 \\Exp 、 DoubleToStr 、 WaitSyncTask、FCAct、FCDeact、
FCLoadID、FCCalib、FCSupvForce、FCSupvTorque、FCSupvPosBox、FCSupvPosSphere、FCSupvPosCylinder 、 FCSupvOrient 、 FCSupvOrient 、 FCSupvReoriSpeed 、 FCSupvTCPSpeed、FCCondForce、FCCondTorque、FCCondOrient、FCCondReoriSpeed、FCCondPosBox 、 FCCondPosCylinder 、 FCCondPosSphere 、 FCCondTCPSpeed 、 FCCondWaitWhile、 FCRefLine、 FCRefRot、 FCRefSpiral、 FCRefCircle、 FCRefForce、 FCRefTorque、FCRefStart、FCRefStop、FCSetSDPara

进制转换

RL 语言支持在数字或者字母后面增加进制标识符的方式来直接输入十六进制、二进制字符或者科学计数法的数值。

Example 1

在 0~9、a~f 以及 A~F 后面增加“h”后缀，RL 编译器会将相应的数字或者字母当做十六进制来处理，并在编译器内部转换成十进制处理：
8h，代表十六进制的 8，十进制的 8；

bh，代表十六进制的 b，十进制中的 11；

25h，代表十六进制的 25，十进制的 37；

Example 2

在 0~9、a~f 以及 A~F 后面增加“b”后缀，RL 编译器会将相应的数字或者字母当做二进制来处理：

1b，代表二进制的 1，十进制的 1；

10b，代表二进制的 10，十进制的 2；

1010b，代表二进制的 1010，十进制的 10；

Example 3

在数字后面增加“e±x”，表示该数字乘以 10 的 x 次方，例如：

5e+20，代表 5×10^{20}；

26e-15，代表 26×10^{-15}；

112e-10，代表 112×10^{-10}；

7.3.2 变量声明

说明

在使用一个变量之前必须要先进行声明，变量声明语句格式为：

SCOPE STORAGE TYPE varname [= value]

其中：

1. SCOPE 为变量作用域，详见变量作用域；
2. STORAGE 为变量存储类型，详见存储类型；
3. TYPE 为变量类型，可以是基本类型，也可以是专用类型，详见变量类型；
4. varname 为变量名，详见变量命名规则；

中括号[ ]内为可选内容，可以在声明变量的时候进行初始化，也可以不初始化。对于在声明时没有进行显式初始化的变量，系统会自动根据变量的类型赋予不同的初值。默认的初值可能会在某些情况下造成程序执行问题，因此建议对每一个手动新增的变量进行初始化。

示例

以下是几个变量声明示例：

Example 1

```
VAR int counter = 8          //声明整型变量 count，并赋初值为 8
VAR double time = 2.5        //声明浮点型变量 time，并赋初值为 2.5
VAR bool ifOpen = true       //声明 bool 型变量 ifOpen，并赋初值为 true
```

Example 2

一般情况下，变量不允许重名：

```
VAR int counter = 8
VAR double counter = 2.5
```

此时编译器将报错，提示“添加变量失败”。

Example 3

但是全局变量和局部变量变量可以使用相同的变量名：

```
VAR int counter = 1
```

GLOBAL int counter = 555

虽然不同作用域的变量允许重名，但是为了避免引起混淆和误用，除非使用重名的变量在工艺上有特别的好处，否则不建议使用重名变量。

> 提示
>
> 不可在 while 等循环语句块内部声明变量，否则在该部分代码重复执行时会造成重复声明，导致出现“添加变量失败”错误。
> 请在循环体外部声明需要使用的变量。

使用限制

- 不支持声明 PERS 存储类型的 ROUTINE 变量；
- 当不同级别的变量或函数存在重名情况时，编译器会根据作用域的优先级来决定选择使用哪个变量，具有最高优先级顺序的变量将优先被选中，而低优先级顺序的将被遮蔽隐藏，各作用域的优先顺序如下：

1. 当出现变量重名时，作用域的优先顺序为：ROUTINE > LOCAL > GLOBAL；
2. 当出现函数重名时，作用域的优先顺序为：LOCAL > GLOBAL；

7.3.3 变量类型

7.3.3.1 基本变量

整型 int

整型 int 变量的取值范围是-2^{31}~2^{31}，建议赋值在规定范围内，如果超过范围则会随机赋值，因此使用时请不要超过最大取值范围。

例如：

GLOBAL VAR int counter = 4

表示定义了一个整型的全局变量类型的数据 counter，且初始值等于 4。

浮点型 double

浮点数使用 8 个字节存储，使用时请不要超出取值范围。

例如：

LOCAL VAR double time = 1.5

表示定义了一个浮点型的局部变量 time，且初始值等于 1.5。

布尔型 bool

bool 型变量主要用于状态或逻辑判断，取值为 true 或者 false。

当被赋值 int 或 double 值时，非 0 取值为 true，0 取值为 false。

例如：

GLOBAL VAR bool ifClose = true

表示定义了一个 bool 型的全局变量 ifClose，且初始值为 true。

字符串类型 string

字符串类型变量由多个字母或者数字组成，且定义时必须放在双引号“”内。

例如：VAR string name = “rokae”
表示定义了一个字符串变量 name，并初始化为“rokae”。
字符串类型变量支持“+”操作，可实现字符串拼接。
例如：name = “Rok” + “ae”
表示变量 name 被赋值为”Rokae”。

数组 Array

数组是由相同类型的变量组成的集合，可以是一维也可以是多维。数组中的元素使用下标来访问，每一维的下标都从 1 开始。
例如：

```
VAR int table[1][16]          //声明了一个一维数组，包含 16 个整型变量。
table[1][6] = 8               //将数组的第一行第 6 个元素赋值为 8。
```

也可在声明时对数组进行赋值：

```
VAR int temp[2][3] = {{1,2,3},{4,5,6}}。
```

矩阵可认为是二维数组，声明和定义的形式与数组一样。
例如声明一个整型的 2 行 2 列的矩阵：

```
VAR int matrix [2][2] = {{1,2},{3,4}}
```

> 提示
>
> 数组总长度不允许超过 1000。

类型隐式转换

为方便使用，某些基本类型变量之间可以进行隐式类型转换，见下表：

初始数据类型	可隐式转换类型
浮点型（double）	布尔型（bool）
整型（int）	布尔型（bool）
整型（int）	浮点型（double）

Example 1

```
VAR int a1 = 3
VAR double a2 = 4
VAR double a3 = a1 * a2          //编译器将 a1 隐式地把转换成浮点型数据。
VAR double counter = 5
while(counter)
counter--
endwhile                         //编译器将 counter 隐式地转换成 bool 型数据。
```

7.3.3.2 byte

说明

byte 表示 RL 语言中的无符号字节，相当于 C++中的 unsigned char。取值范围 0~255，不允许为负值。一般用于 SocketSendByte 指令。
当 byte 赋值超限时，不会报错，会自动截断取低 8 位字节。

示例

下面的例子显示了如何使用 byte：

Example 1

```
VAR byte data = 177
```

定义了一个 byte 类型变量 data，它的值为 177。

> 提示
>
> 当 byte 变量值超过 255 时，会自动进行截断，只保留低 8 位的值，比如 var byte data2=288，截断后 data2 的值为 32。

7.3.3.3 clock

说明

clock 用于计时，clock 指令就像用于计时的秒表。

clock 类型存储的时间精度是 0.001s，最大时间间隔 45 天（即 45×24×3600 秒）。

示例

下面的例子显示了如何使用 clock：

Example 1

```
VAR clock clock1
ClkStart clock1
ClkStop clock1
VAR double interval=ClkRead(clock1)
ClkReset clock1
```

interval 读到的是 ClkStart 和 ClkStop 之间的时间间隔，单位是 s。

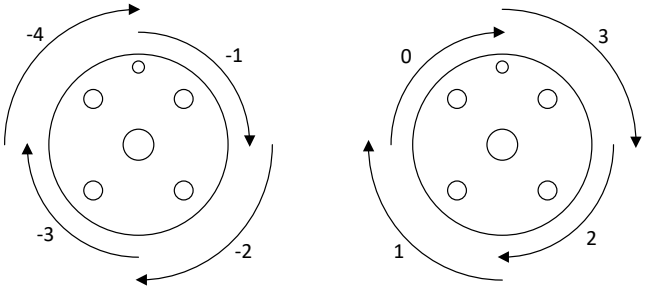
7.3.3.4 confdata

说明

confdata（Robot Configuration Data）用于定义空间目标点对应的形态配置数据。

对于常见的串联 6 轴机器人来讲，同一个笛卡尔空间目标点最多对应 8 组不同的运动学逆解，因此需要使用 confdata 来明确指定选择哪一个形态。

此外，由于机器人使用的是旋转关节，任何一个关节在 1°和 361°时表现出来的状态是一样的，因此在选定机器人的形态后，还需要采取其他措施来处理关节的多圈性问题。此处我们以象限数的方法来标记关节角度的大致范围，例如当关节角度位于 0~90 度时其象限数为 0，关节角度为 90~180 度时记为 1，每隔 90 度增减 1。角度为负数时，相应的象限数也为负数，如下图所示（顺时针旋转为正向）：



对于串联 6 轴工业机器人来讲，需要 4 个参数来构成完整的 confdata，包括：

- cf1，记录 1 轴的象限数；
- cf4，记录 4 轴的象限数；
- cf6，记录 6 轴的象限数；
- cfx，记录机器人使用哪个位形到达目标位置，详见后续解释。

定义

cf1

数据类型：int

1 轴角度对应的象限。

cf4

数据类型：int

4 轴角度对应的象限。

cf6

数据类型：int

6 轴角度对应的象限。

cfx

数据类型：int

机器人使用的形态配置编号，取值范围 0~7。

补充说明

对于工业上常见的六轴串联球腕或 L 形手腕机器人来讲，同一个末端笛卡尔位姿最多有 8 组不同的逆解，cfx 使用 0~7 这 8 个整数代表 8 组逆解，详细解释如下。

cfx	腕心位于 1 轴……	腕心位于大臂……	5 轴角度为……
0	前面	前面	正
1	前面	前面	负
2	前面	后面	正
3	前面	后面	负
4	后面	前面	正
5	后面	前面	负
6	后面	后面	正
7	后面	后面	负

7.3.3.5 jointtarget

说明

存储机器人关节角度和外部轴位置。

关节角度的单位是度 Degree，外部导轨的单位是毫米 mm。

定义

robax

机器人关节角度（Robot Axes）

数据类型：double

robax 包含 6 个 double 型的成员，分别存储机器人 6 个关节的角度 Degree。

extax

外部轴位置（External Axes）

数据类型：double

extax 包含 6 个 double 型的成员，最多可存储 6 个外部轴的位置信息。

如果外部轴为旋转轴，则单位是角度 Degree；如果外部轴为直线轴，则单位是毫米 mm。

示例

Example 1

VAR jointtarget home = J:{0,0,0,0,90,0}{EJ 0,0,0,0,0,0}

上述语句定义了一个名为“home”的点，除五轴为 90 度外，机器人其他轴角度均为 0 度，所有的外部轴回到零位。

Example 2

VAR jointtarget start = J:{10,20,30,0,90,0}{EJ 325.6,0,0,0,0,0}

上述语句定义了一个名为“start”的点，机器人一轴为 10 度，二轴为 20 度，三轴为 30 度，五轴为 90 度，第一个外部轴位置为 325.6 度或 325.6 毫米，取决于外部轴类型。

7.3.3.6 load

说明

load 变量类型用于存储机器人负载的动力学参数。

机器人的负载主要有两种：

- 安装在机器人末端的工具或工件本身；
- 工具所抓起/吸起来的物体。

load 型变量不支持单独创建，只能作为 tool 型变量的成员在标定工具界面手动修改。

正确定义负载的动力学参数可以使得机器人获得最佳性能。

警告

请务必正确定义机器人末端负载的动力学参数，包括工具本身以及由工具抓取的物体两部分。错误的定义可能会导致如下后果：

- 机器人无法最大化利用伺服系统的能力，导致性能下降；
- 路径精度降低，定位误差变大；
- 机械部件过载导致寿命降低或损坏。

定义

在 Titanite 系统中负载被当做刚体来处理，用于描述负载的参数有 4 个。

参数定义：

PERS load piece1 = l:{mass,{cogx，cogy，cogz}, {q1,q2,q3,q4}, ix, iy, iz}

mass

质量（Mass）

数据类型：double

用于描述负载的质量，单位是千克 kg。

cogx

质心（Center of Gravity）在 x 方向的偏移量。

数据类型：double

如果工具安装在机器人上，则 cogx 记录负载质心在工具坐标系下 X 方向的偏移量；如果使用外部工具功能，则 cogx 记录被抓手夹持的负载质心在工件坐标系下 X 方向的偏移量。

cogy

质心（Center of Gravity）在 y 方向的偏移量。

数据类型：double

如果工具安装在机器人上，则 cogy 记录负载质心在工具坐标系下 Y 方向的偏移量；如果使用外部工具功能，则 cogy 记录被抓手夹持的负载质心在工件坐标系下 Y 方向的偏移量。

cogz

质心（Center of Gravity）在 Z 方向的偏移量。

数据类型：double

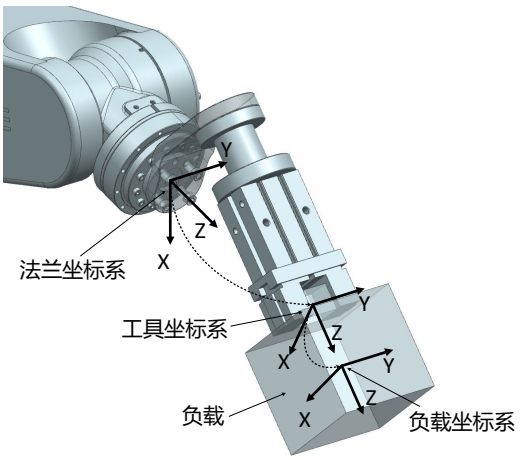
如果工具安装在机器人上，则 cogz 记录负载质心在工具坐标系下 Z 方向的偏移量；如果使用外部工具功能，则 cogz 记录被抓手夹持的负载质心在工件坐标系下 Z 方向的偏移量。

q1~q4

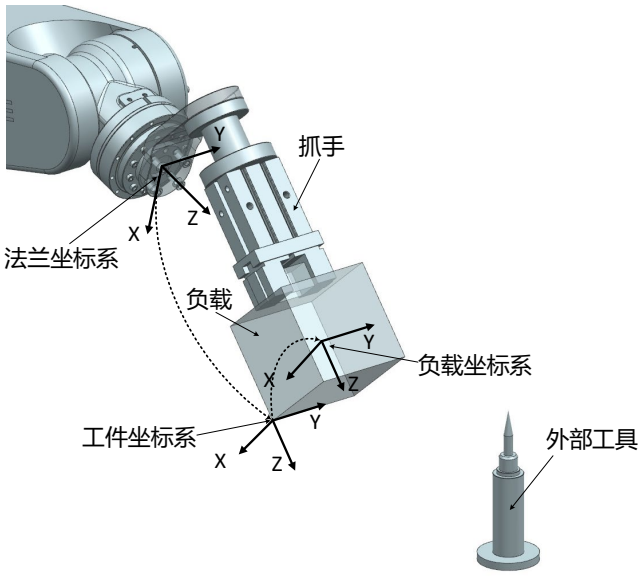
四元数，记录负载惯量主轴的方向。

数据类型：double

当工具安装在机器人上时，惯量主轴的方向是在工具坐标系下描述的，见下图：



当使用外部工具时，惯量主轴的方向是在工件坐标系下描述的，见下图：



ix

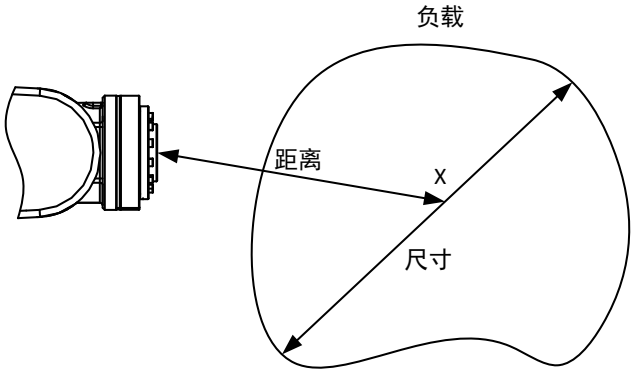
惯量 x

数据类型：double

负载沿 x 主轴的惯量，单位 kgm^2。

正确定义负载惯量有助于提升机器人运动精度，尤其是搬运较大物体的情况下。如果 ix、iy、iz 都被设置为零，则负载将会被当做一个质点来处理。

通常情况下，如果负载质心到法兰中心点的距离小于负载本身的最大尺寸时，就应该对负载惯量进行定义，如下图：



iy

惯量 y

数据类型：double

负载沿 y 主轴的惯量，单位 kgm^2。

iz

惯量 z

数据类型：double

负载沿 z 主轴的惯量，单位 kgm^2。

7.3.3.7 orient

说明

存储坐标系或空间刚体的姿态信息。

orient 类型的变量不支持单独创建或者修改，仅作为某些变量的成员变量。

定义

RL 语言系统使用四元数来表示姿态，因此共有 4 个分量，表示形式为：

q1

数据类型：double

四元数的第一个分量。

q2

数据类型：double

四元数的第二个分量。

q3

数据类型：double

四元数的第三个分量。

q4

数据类型：double

四元数的第四个分量。

关于四元数

通常情况下我们使用旋转矩阵来描述刚体的姿态，四元数是另一种更为简洁的姿态描述方式。

四元数的四个分量满足如下关系：

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$$

旋转矩阵和四元数之间可以相互转换，假设有一个旋转矩阵 R：

$$\mathrm{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

则:

$q_1 = \frac{\sqrt{r_{11} + r_{22} + r_{33} + 1}}{2}$	无
$q_2 = \frac{\sqrt{r_{11} - r_{22} - r_{33} + 1}}{2}$	$sign\ q_2 = sign(r_{32} - r_{23})$
$q_3 = \frac{\sqrt{r_{22} - r_{11} - r_{33} + 1}}{2}$	$sign\ q_3 = sign(r_{13} - r_{31})$
$q_4 = \frac{\sqrt{r_{33} - r_{11} - r_{22} + 1}}{2}$	$sign\ q_4 = sign(r_{21} - r_{12})$

7.3.3.8 pos

说明

用来存储三维空间的位置信息。

pos 类型的变量不支持单独创建或者修改，仅作为某些变量的成员变量。

定义

RL 语言系统使用笛卡尔坐标系来描述三维空间，因此 pos 变量有 x、y、z 三个分量。

x

数据类型：double

位置的 x 坐标。

y

数据类型：double

位置的 y 坐标。

z

数据类型：double

位置的 z 坐标。

7.3.3.9 pose

说明

存储笛卡尔空间的位置和姿态。

定义

参数定义：

VAR pose pose1 = pe:{x, y, z},{q1,q2,q3,q4}

x

数据类型：double

位置的 X 坐标。

y

数据类型：double

位置的 Y 坐标。

z

数据类型：double

位置的 Z 坐标。

q1

数据类型：double

四元数的第一个分量。

q2

数据类型：double

四元数的第二个分量。

q3

数据类型：double

四元数的第三个分量。

q4

数据类型：double

四元数的第四个分量。

7.3.3.10 robtarget

说明

存储三维空间的笛卡尔位置和姿态，用于 MoveJ 和 MoveL 等指令。

由于机器人运动学逆解存在多解性，对于同一个目标位姿，机器人往往可以采用多种不同

的形态到达，为了明确的指定采用哪一种配置形态，robtarget 变量中还包含了机器人配置（Robot Configuration）数据。

robtarget 类型的变量在通过辅助编程插入运动指令时自动创建，手动更改变量内部的的值可能会导致 Pose 和 ConfData 不对应，机器人无法正常执行运动指令。

警告

机器人程序中使用笛卡尔位置和姿态均是在**工件坐标系**下定义的。如果最终使用的工件与最初编程时使用的工件不一样，会导致机器人的运动偏离期望路径。因此对于以下两种情况要确认工件的变更不会造成危险：

- 使用“修改指令”功能更改指令的 wobj 参数时；
- 实际使用的工件与程序指令中使用的工件不一样时。

不当使用会导致人员受伤或设备损坏！

定义

trans

空间位置

数据类型：pos

保存在参考坐标系下的位置偏移量。

rot

姿态

数据类型：orient

保存在参考坐标系下的姿态。

conf

机器人配置数据（Robot Configuration）

数据类型：confdata

保存机器人的位形配置数据，详见 confdata 介绍。

extax

外部轴信息（External Axes）

数据类型：double

extax 包含 6 个 double 型的成员，最多可存储 6 个外部轴的位置信息。

如果外部轴为旋转轴，则单位是角度 Degree；如果外部轴为直线轴，则单位是毫米 mm。

示例

VAR robtarget pB1=P:{{1289.491,0.001,3102.876},{0.987,0.000,-0.162,0.000}}{cfg 0,0,0,1}{EJ 0,0,0,0,0,0}

定义了一个名为 pB1 的笛卡尔位姿，所有外部轴处于零位。

7.3.3.11 signalxx

说明

signalxx 类型的变量用于描述输入与输出信号。

所有 signalxx 类型的变量都需要在“控制面板”的“输入输出”界面进行定义，然后在程

序中使用，不支持在程序中直接声明。

描述

signalxx 目前仅支持数字输入输出，包括如下变量类型：

变量类型	用于描述…	说明
signaldi	数字输入信号	值为 True 或者 False，仅表示状态
signaldo	数字输出信号	值为 True 或者 False，对输出赋值
signalgi	数字组输入信号	将一段连续的物理输入端口定义为一个二进制数，在 RL 中可转换为十进制使用，最多支持 16 个 DI 构成组输入，因此 signalgi 的取值范围是 0～（2^n -1），n 是组输入包含的 DI 点个数
signalgo	数字组输出信号	将一段连续的物理输出端口定义为一个二进制数，在 RL 中可转换为十进制使用，最多支持 16 个 DO 构成组输出，因此 signalgo 的取值范围是 0～（2^n -1），n 是组输出包含的 DO 点个数

signaldo 和 signalgo 类型仅包含信号的引用，可使用单独的指令（例如 SetDO、SetGO 等）来进行赋值。

signaldi 和 signalgi 可用于在程序中直接获取所对应输入信号的值。

示例

Example 1

```
//使用数字输入的状态作为判断条件
IF（di1 == true）
   do something…
ENDIF
```

Example 2

```
//使用数字组输入的状态作为判断条件
例如定义组输入 gi2 映射了 Profinet IO 的第一个 byte 的前三个 bit，那么当 bit0~bit2 的值分别为 0,1,1 时，gi2 的值为 110（换算成 int 型为 6），组输出（signalgo）同理。
IF（gi2 == 8）
   do something…
endif
```

注意事项

不支持在程序中定义/声明 signalxx 类型的变量，如果出现这种用法，程序将会报错。在使用 signalxx 类型的变量之前，请首先在控制面板中进行配置。

提示

signalxx 型变量的作用域为 System，与其他作用域类型的关系为 System > GLOBAL >

LOCAL。
如果 IO 配置界面的 Signal 与 RL 程序中声明的变量重名，那么较低作用域级别的变量将被选中

7.3.3.12 speed

说明

用来定义机器人和外部轴的运动速度。
为方便用户使用，系统预设了常用的速度变量，可通过辅助编程直接选择使用，详见插入指令。

定义

speed 型变量包含 5 个成员变量，分别是 per、tcp、ori、exj_r、exj_l。

per

关节速度百分比（Joint Velocity Percentage）
数据类型：double
用于指定关节运动指令时的运动速度，适用于 MoveAbsJ 和 MoveJ 指令，取值范围 1%~100%。

tcp

TCP 线速度（Linear Velocity）
数据类型：double
定义工具中心点的线速度，取值范围 0.001 毫米/秒 ~ 7000 毫米/秒。

ori

空间旋转速度（Orientation Velocity）
数据类型：double
定义工具的旋转速度，取值范围 0.001 度/秒 ~ 500 度/秒。

exj_l

外部轴线速度
数据类型：double
定义外部直线轴的运动速度，取值范围 0 毫米/秒 ~ 2000 毫米/秒。

exj_r

外部轴角速度
数据类型：double
定义外部旋转轴的运动速度，取值范围 0 度/秒 ~ 300 度/秒。

示例

VAR speed v1 = v:{40,300,100,200,1000}
定义了一个名为 v1 的 speed 变量，其中关节旋转速度为最大允许速度的 40%，TCP 线速度为 300 mm/s，空间旋转速度为 100°/s，外部轴角速度为 200°/s，外部轴线速度为 1000 mm/s。

预定义的速度变量

系统预定义了常用的速度变量，具体如下表所示。

名称	关节速度百分比	TCP 线速度	空间旋转速度	外部轴角速度	外部轴线速度
v5	1%	5 mm/s	200°/s	0°/s	0 mm/s
v10	3%	10 mm/s	200°/s	0°/s	0 mm/s
v25	5%	25 mm/s	200°/s	0°/s	0 mm/s
v30	5%	30 mm/s	200°/s	0°/s	0 mm/s
v40	5%	40 mm/s	200°/s	0°/s	0 mm/s
v50	8%	50 mm/s	200°/s	0°/s	0 mm/s
v60	8%	60 mm/s	200°/s	0°/s	0 mm/s
v80	8%	80 mm/s	200°/s	0°/s	0 mm/s
v100	10%	100 mm/s	200°/s	0°/s	0 mm/s
v150	15%	150 mm/s	200°/s	0°/s	0 mm/s
v200	20%	200 mm/s	200°/s	0°/s	0 mm/s
v300	30%	300 mm/s	200°/s	0°/s	0 mm/s
v400	40%	400 mm/s	200°/s	0°/s	0 mm/s
v500	50%	500 mm/s	200°/s	0°/s	0 mm/s
v600	60%	600 mm/s	200°/s	0°/s	0 mm/s
v800	70%	800 mm/s	200°/s	0°/s	0 mm/s
v1000	100%	1000 mm/s	200°/s	0°/s	0 mm/s
v1500	100%	1500 mm/s	200°/s	0°/s	0 mm/s
v2000	100%	2000 mm/s	200°/s	0°/s	0 mm/s
V3000	100%	3000 mm/s	200°/s	0°/s	0 mm/s
v4000	100%	4000 mm/s	200°/s	0°/s	0 mm/s
v5000	100%	5000 mm/s	200°/s	0°/s	0 mm/s
v6000	100%	6000 mm/s	200°/s	0°/s	0 mm/s
v7000	100%	7000 mm/s	200°/s	0°/s	0 mm/s

提示

系统预定义的 speed 变量中所有的空间旋转速度都是 200°/s，如果对机器人末端的旋转速度有特殊要求，可根据工艺要求自行定义新的 speed 变量以便使用。

7.3.3.13 syncident

说明

同步标识符用于定义任务之间配合时的等待点，与 WaitSyncTask 指令配合使用。

所有需要在同一个点同步的任务中的 syncident 变量名称应该都一样。

在多个任务需要协同配合执行时，syncident 用来定义一个程序执行位置，在所有任务都运行到包含相同 syncident 参数的 WaitSyncTask 指令之前，任何一个任务都不能继续执行；或者说只有全部需要协同的任务都运动到相同的 WaitSyncTask 指令行时，所有的任务才会继续执行。

syncident 型变量属于非值变量（Non-Value），因此不支持任何数学运算操作。
该类型在变量管理中显示，不能手动修改，只能删除。

示例

GLOBAL PERS tasks task_list[2] = {“ROB1”, “STN1”, “ROB2”}
GLOBAL VAR syncident sync1
WaitSyncTask sync1, task_list
任务 ROB1 中执行到 WaitSyncTask 时，会等待 STN1 和 ROB2 两个任务都执行到有着相同 syncident 参数的 WaitSyncTask 指令处，然后再同时向下执行。

7.3.3.14 tasks

说明

在多任务时，tasks 型变量用来指定若干个需要同步的任务名称。
该类型在变量管理中显示，也可以直接手动修改。

定义

任务的名称以字符串的形式给定，然后多个 RL 任务组成一个数组，构成一个 tasks 变量。之后该变量可以被 WaitSyncTask 指令使用。

示例

GLOBAL PERS tasks task_list[2] = {“task_1”, “task_2”}
定义了一个名为 task_list 的 tasks 变量，包含“task_1”和“task_2”两个任务。

7.3.3.15 tool

说明

tool 型变量用来记录工具参数，包括机器人所用工具的 TCP、姿态以及动力学参数。
机器人使用工具与外部环境交互，因此 tool 变量会从以下几个方面影响机器人的运动：

- 只有工具中心点 TCP 会按照编程的路径和速度运动，当机器人执行一个空间纯旋转运动时，只有 TCP 会保持不动；
- 编程时指定的运动路径和速度均是指工具坐标系相对于工件坐标系运动的路径和速度，因此更换良好标定后的工具或工件不影响路径的形状和速度大小。
- 当使用外部工具时，编程的速度指的是工件的速度（相对于外部工具）；

注意，当使用外部工具时，tool 变量中的 tframe 记录外部工具的零点位置和姿态偏移量，而 tload 则用来记录机器人末端所安装的用于抓取工件的手抓的动力学参数。
tool 型变量的数据都存储在数据库中，加载程序时由程序编辑器从数据库中读出，因此请不要尝试在程序编辑器中直接对 tool 型变量直接修改，以免造成不可预知的错误。如果需要修改 tool 型变量，请通过标定界面进行修改，详见标定工具坐标系。

警告

请务必正确定义机器人末端负载的动力学参数，包括工具本身以及由工具抓取的物体两部分。错

误的定义可能会导致如下后果：

- 机器人无法最大化利用伺服系统的能力，导致性能下降；
- 路径精度降低，定位误差变大；
- 机械部件过载导致寿命降低或损坏。

定义

robhold

数据类型：bool

定义工具是否安装在机器人上，True 表示工具安装在机器人上，False 表示工具没有安装在机器人上，当前正在使用外部工具。

在进行 Jog 或者执行程序时，同时使用的工具/工件组合中，robhold 参数只能有一个为 True，即如果工具的 robhold 为 True，则对应的工件 robhold 就必须为 false，反之亦然，否则机器人会给出错误提示，并无法进行 Jog 或执行相应的程序语句。

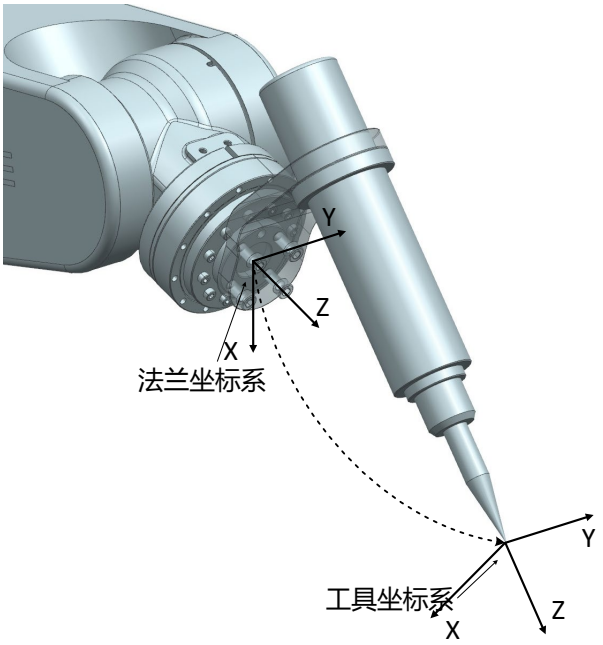
tframe

工具坐标系（Tool Frame）

数据类型：pose

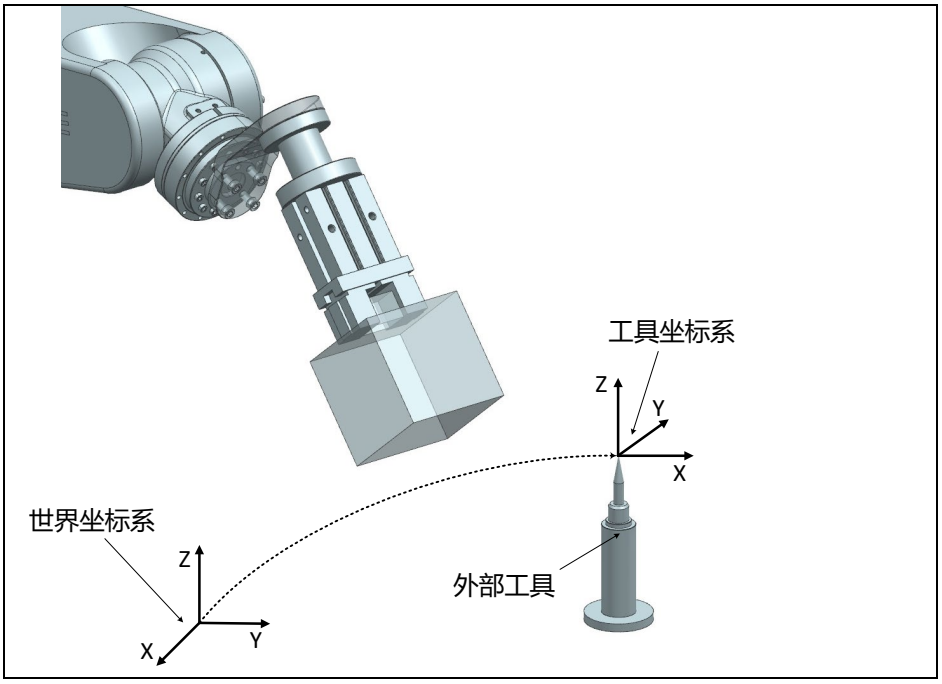
记录所用工具的工具坐标系，包括：

- TCP 表示相对于机器人末端法兰坐标系在 x、y、z 三个方向的偏移量，单位是毫米。
- 工具坐标系相对于法兰坐标系的姿态偏移量，用四元数表示，见下图：



提示

当使用外部工具功能时，工具的 TCP 和姿态是相对于世界坐标系定义的：



tload

工具的动力学参数

数据类型：load

记录工具的动力学参数，对于普通工具来讲，tload 描述整个工具的动力学参数，对于外部工具来讲，tload 描述机器人用来（夹持工件的）手抓的动力学参数。

对于安装在机器人上的普通工具来讲，其 load 参数包括：

- 工具的质量（重量），单位 kg；
- 工具的重心，在法兰坐标系下描述，单位毫米；
- 惯量主轴的方向，在法兰坐标系下描述；
- 以及工具沿惯量主轴的惯量大小，单位 kgm^2。如果所有的惯量分量都定义为 0 kgm^2，则该工具会被当做一个质点（Point Mass）来处理。

提示

如果机器人使用的是外部工具，那 tload 成员则用来记录安装在机器人上的手抓的动力学参数，具体的参数含义保持不变。

提示

请注意 tload 成员仅仅定义机器人用来（夹持工件的）手抓的动力学参数，被夹持的工件的动力学参数并不包含在内，为了保证机器人在任何情况下都获得最佳性能，需要定义两个 tool 变量来处理这种情况：

- 一个 tool 保存手抓本身的所有参数；
- 另一个 tool 保存手抓+被夹持工件的所有参数；

在使用时，通过在运动语句中使用不同的工具来实现有无负载的切换功能。

示例

CONST tool tool2 := { true, {{100, 0, 220}, {1, 0, 0 ,0}}, {2, {20, 0, 50}, {1, 0, 0, 0}, 0, 0, 0}}
定义了一个名为 tool2 的工件，其中的各项参数为：

- 该工具安装在机器人上；
- TCP 相对于法兰坐标系 XYZ 方向偏移量分别为 100，0，220，姿态与法兰坐标系相同；
- 该工具的质量为 2kg，质心相对于法兰坐标系原点在 XYZ 方向上的偏移为 20，0，50mm；
- 该工具当作质点处理，惯量数据为 0。

7.3.3.16 wobj

说明

wobj 是工件（Work Object）的缩写，工件指被机器人加工、处理、搬运的物体。
所有运动指令中使用的位置都是在工件坐标系下定义的（如果没有指定工件坐标系，则默认在世界坐标系下定义，世界坐标系可以被看做是 wobj0），这样做有如下几个好处：

- 很多加工点的位置可以从工件的设计图纸中获得并直接使用；
- 当机器人被重新安装或者工件被移动后，只需要重新标定工件坐标系就可以直接复用之前的程序，避免了重新编程；
- 在配备合适传感器的情况下，可以自动补偿工件的震动或者轻微移动。

正常情况下，如果不定义专门的工件坐标系，那么控制系统将把世界坐标系当做默认的工件坐标系 wobj0。但是当使用外部工具时，必须要定义工件坐标系，因为此时编程的路径和速度是指工件的路径和速度，而不是工具的。
通常工件坐标系是相对于用户坐标系定义的，但是如果用户未指定用户坐标系，则工件坐标系默认相对于世界坐标系定义，详见机器人坐标系统。
工件实际上由两个坐标系组成，分别是用户坐标系和工件坐标系，在工件坐标系的上层插入一个用户坐标系，是为了支持有多个相同工件需要加工的情况，有关坐标系定义关系的解释详见“定义”部分有关 oframe 的解释。

提示

wobj 型变量的数据都存储在数据库中，加载程序时由程序编辑器从数据库中读出，因此请不要尝试在程序编辑器中直接对 wobj 型变量直接修改，以免造成不可预知的错误。如果需要修改 wobj 型变量，请通过标定界面进行修改，详见定义工件。

定义

robhold

定义该工件是否安装在机器人上。True 表示工件安装在机器人上，当前正在使用外部工具，False 表示工件没有安装在机器人上，当前正在使用普通工具。

ufprog

用户坐标系是否移动（User Frame Programmed）

变量类型：bool

定义用户坐标系是固定的还是移动的，True 表示用户坐标系是固定的，False 表示用户坐标系是移动的，e.g 定义在外部变位机或者其他机器人上。

该值多用于当需要机器人与变位机或其他机器人协调运动时。

ufmec

用户坐标系关联的机械单元（User Frame Mechanical Unit）

数据类型：string

用机械单元名称的方式来指定用户坐标系与哪个机械单元绑定，只有当 ufprog 为 false 时才有用。

oframe

工件坐标系（Work Object Frame）

数据类型：pose

存储工件坐标系的原点和姿态。

uframe_id

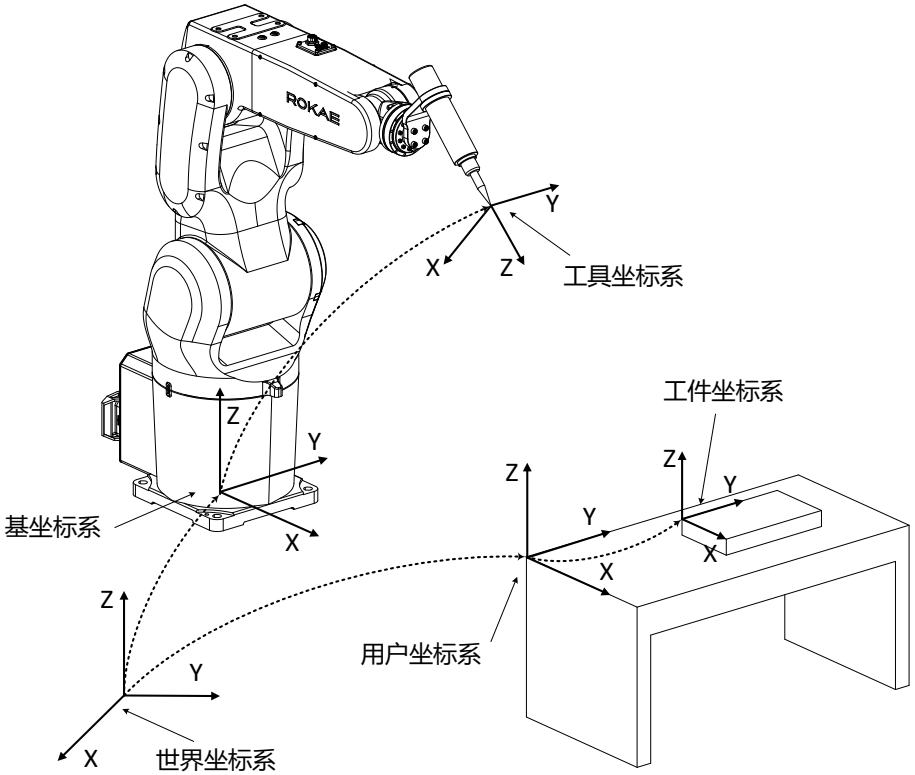
用户坐标系（User Frame）id

数据类型：int

存储用户坐标系的 id。可通过 id 找到对应的用户坐标系。

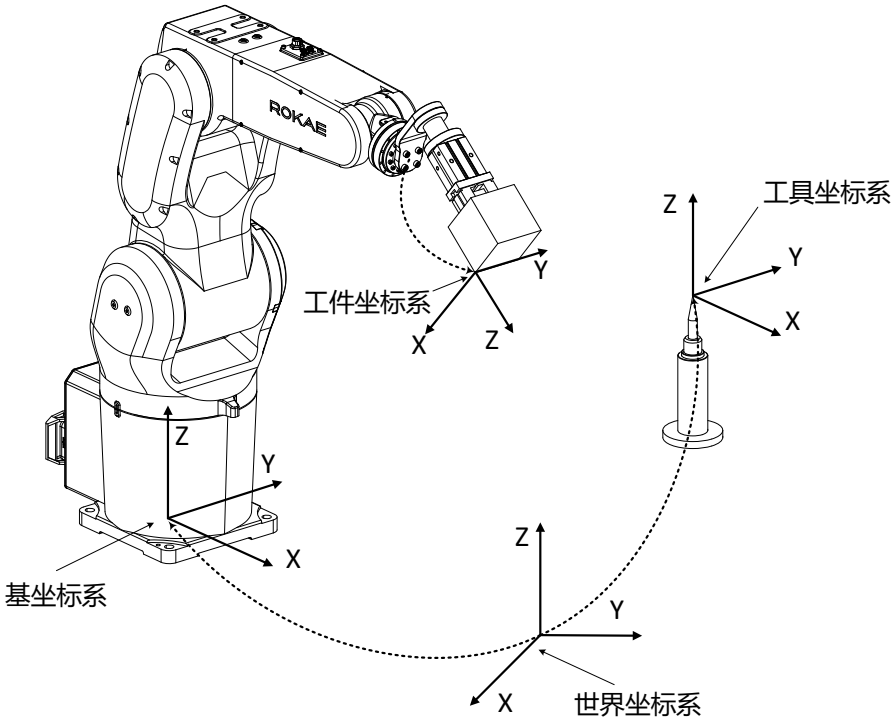
在使用**普通工具**时（非外部工具），坐标系的定义链为：

- 工件坐标系相对于用户坐标系定义；
- 用户坐标系相对于世界坐标系定义。



当使用**外部工具**时，坐标系的定义链为：

- 工件坐标系相对于用户坐标系定义；
- 用户坐标系相对于法兰坐标系定义。



示例

CONST wobj wobj2 :={FALSE, TRUE, "robot", {300, 600, 200}, {1, 0, 0, 0}, 1};

定义了一个名为 wobj2 的工件，其中的各项参数为：

1. 该工件不是安装在机器人上；
2. 工件坐标系是固定的，不会随外部变位机或者其他机器人运动；
3. 工件坐标系原点在用户坐标系下的坐标值为 300 mm、600 mm、200 mm，姿态与用户坐标系一致；
4. 用户坐标系 id 为 1。

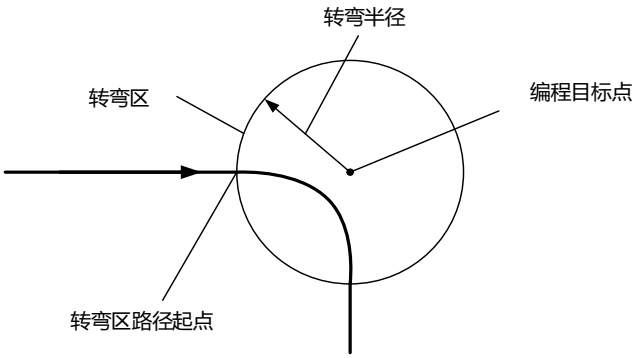
7.3.3.17 zone

说明

zone 变量用于定义某一个运动如何结束或者说定义两条运动轨迹之间转弯区的大小。

对于同一个机器人指令目标点，在运动指令中有两种处理方式：

1. 当做**停止点**处理，机器人将运动到目标点且到达目标点时的速度为 0，之后才会继续执行下一条指令；
2. 当做**过渡点**处理，机器人不会运动到目标点，而是从距离该目标点若干毫米的地方开始转向往下一个目标点运动，转弯路径会偏离编程路径。两条轨迹之间的过渡区域我们称之为转弯区，见下图：



转弯区的大小不能超过路径长度的一半，如果超过，系统会自动将转弯区缩小到总路径长度一半大小。

使用转弯区可以避免机器人频繁启停，显著减少节拍时间。

定义

关节轨迹和笛卡尔空间轨迹使用不同的参数来定义转弯区。

distance

笛卡尔空间转弯区大小

数据类型：double

用于 MoveL 指令，定义笛卡尔空间轨迹的转弯区大小，即当机器人运动到距离目标点还有 distance 毫米的地方时，开始转向往下一个目标点运动，单位是毫米，取值范围 0 ~ 200 mm。

percent

转弯百分比

数据类型：double

用于 MoveJ 和 MoveAbsJ，表示距离目标角度还有多远时开始转弯，100%表示整个转动角度值的一半；对于只有空间纯旋转运动的 MoveL 指令，也会使用 Percent 参数，而不使用 Distance。

示例

VAR zone z100 = s:{100,50}

定义了一个 zone 变量，其中笛卡尔空间转弯区大小为 100 mm，关节空间转弯区大小为 50%。

预定义的转弯区变量

系统预定义了常用的转弯区变量，具体如下表所示。

名称	笛卡尔空间转弯区大小	转弯百分比
fine	0 mm	0%
z1	1 mm	1%
z5	5 mm	3%
z10	10 mm	5%
z15	15 mm	8%

z20	20 mm	10%
z30	30 mm	15%
z40	40 mm	20%
z50	50 mm	25%
z60	60 mm	30%
z80	80 mm	40%
z100	100 mm	50%
z150	150 mm	75%
z200	200 mm	100%

使用限制

在某些特定情况下转弯区会被取消，系统会上报“转弯区被取消 Corner Path Failed”日志。

- 前后两条轨迹中的至少一条长度过小（1 mm/0.001 rad）；
- 前后两条轨迹接近平行且运动方向相反；
- 前后两条轨迹有一条轨迹为纯旋转的运动（前后两条如果均为纯旋转，则会生成转弯区）；

前后两条轨迹纯旋转运动转轴反向，如前一条只有六轴正转，后一条只有六轴反转。

产生“转弯区被取消”警告时，程序自动将受到影响的语句目标点当做停止点处理。

除了以上几种特殊情况外，所有的逻辑指令也会导致其上一条运动指令的转弯区被取消。

7.4 运算符

算数运算符

算数运算符包括：

运算符	作用
+	加
-	减/负号
*	乘
/	除
%	取余
--	自减
++	自加

算数运算操作符支持 int，double 类型数据的操作，各种算术运算符用法示例如下：

Example 1

```
VAR int a = 1
VAR int b = 2
VAR int c = -b    //取负
VAR int ac = a * c //乘法
```

Example 2

矩阵和基本数据类型可以进行乘法运算：

```
VAR double counter = 2
VAR double matrix1[3][3] = {{1,2,3},{3,4,5},{3,4,6}}
VAR double matrix2[3][3] = counter * matrix1
```

Example 3

++，--两个运算符又称单目运算符，是指对一个操作数进行操作的运算符：

```
x = n++              //表示将 n 的值赋给 x 后，n 再加 1
x = --n              //表示 n 先减 1 后，再将新值赋给 x
```

逻辑运算符

逻辑运算符支持基本数据类型的运算，包括：

运算符	作用
&&	逻辑与
||	逻辑或
<	小于
>	大于
<=	小于等于
>=	大于等于
==	等于
!=	不等于
!	取逻辑非

逻辑与&&表达式为真的条件是两边的结果都为真，而逻辑或||为真的条件是两边只要有一个条件为真即可。

Example 1

其他逻辑运算符用法示例如下：

```
VAR int res = 1
while(res < 3)                    //比较 res 是否小于 3
res++
endwhile
di5 = !di6                //取逻辑非
VAR int counter = 4
while(di7 && di8)     //求逻辑与
if(counter == 5)                  //是否相等
        break
   endif
endwhile
```

赋值运算符

赋值运算符包括：

运算符	作用
=	赋值
+=	加等
-=	减等
*=	乘等
/=	除等
%=	取模等

各种赋值运算符用法示例如下

```
VAR int num1 = 3
VAR int num2 = 4
num1 += num2          //等同于  num1 = num1 + num2  则 num1 = 7。
num1 -= num2          //等同于  num1 = num1  –  num2，则 num1 = -1。
num1 *= num2          //等同于  num1 = num1 * num2  则 num1 = 12。
num1 /= num2          //等同于  num1 = num1 / num2  则 num1 = 0。
num1 %= num2          //等同于  num1 = num1 % num2  则 num1 = 3。
```

其他运算符

运算符	作用
()	圆括号
.	点操作符

各运算符用法示例如下：

Example 1

```
VAR int num = arr[1]          //取数组第一个元素赋给 num
```

```
VAR int num2 = (1+2)*3     //使用括号可以改变运算顺序，这里 num2 的值为 9
```

Example 2

```
定义一个 robtarget 变量 pt1
pt1.x = 200                //使用“.”操作符将 pt1 点的 x 坐标改为 200
```

使用限制

“.”操作符不支持对 robtarget 变量 A，B，C 成员的修改。

7.4.1 优先级

优先级	运算符	使用形式	结合方向
1	( )	（表达式）/函数名(形参表)	
	.	变量名.	
2	-	-表达式	右到左
	++	++变量名/变量名++	
	--	--变量名/变量名--	
	!	!表达式	
3	/	表达式/表达式	左到右
	*	表达式*表达式	
	%	整型表达式/整型表达式	
4	+	表达式+表达式	左到右
	-	表达式-表达式	
5	>	表达式>表达式	左到右
	>=	表达式>=表达式	
	<	表达式<表达式	
	<=	表达式<=表达式	
6	==	表达式==表达式	左到右
	!=	表达式!= 表达式	
7	&&	表达式&&表达式	左到右
8	||	表达式||表达式	左到右
9	=	变量=表达式	右到左
	/=	变量/=表达式	
	*=	变量*=表达式	
	%=	变量%=表达式	
	+=	变量+=表达式	
	-=	变量-=表达式	

7.5 指令

7.5.1 AccSet

说明

当机器人搬运易碎物品时，可使用 AccSet 指令来降低加速度和减速度以达到更加平顺的运动效果。

示例

以下是一些 AccSet 的基本用法。

Example 1

AccSet 50,100

加速度大小被降低为系统预设值的 50%，加速度变化率（即加加速度或 jerk）保持不变。

Example 2

AccSet 100,50

加速度大小保持不变，但加加速度降低为系统预设值的 50%。

参数

AccSet Acc, Ramp

Acc

数据类型：int

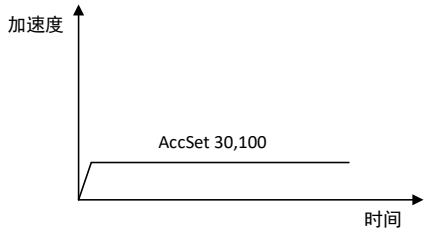
按照系统预设值的百分比指定加速度和减速度的大小，取值范围 20%~100%，其中 100%对应系统默认的最大加速度，小于 20%的参数系统会自动按照最大加速度的 20%处理。

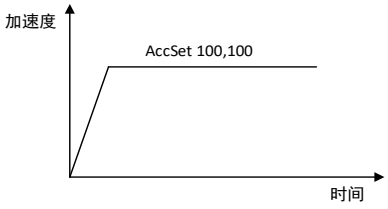
Ramp

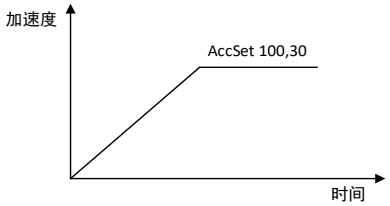
数据类型：int

按照系统预设值的百分比指定加加速度（Jerk）的大小，取值范围 10%~100%，其中 100%对应系统默认的最大加加速度，小于 10%的参数系统会自动按照最大加加速度的 10%处理。

两个参数的作用可参考下图。







程序执行

当执行 AccSet 指令时，机器人和外部轴的加速度都会被改变，且后面的 AccSet 指令效果

会覆盖前一个 AccSet 指令。

发生以下操作时，加速度自动回复为默认大小（100%）：

- RL 程序被手动重置时（PP to Main）
- 加载新的 RL 程序时

7.5.2 ActUnit

说明

用于激活一个机械单元。

示例

Example 1

ActUnit CNV1

MoveL waitp，v1000，fine，tool

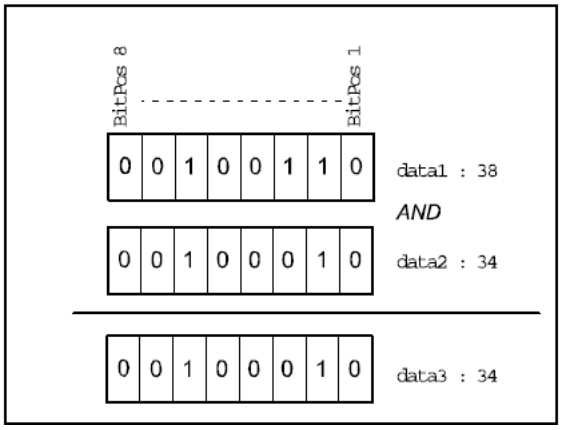
WaitWObj wobjcnv1

激活传送带 CNV1。

7.5.3 BitAnd

说明

BitAnd 用于生成一个逻辑位与的操作，针对 byte 类型数据。如下表：



返回值

数据类型：byte

表示 2 个 byte 类型数据执行逻辑与返回的结果。

定义

BitAnd (BitData1, BitData2)

BitData1

数据类型：byte

要操作的字节数据 1。

BitData2

数据类型：byte

要操作的字节数据 2。

示例

Example 1

```
VAR byte data1 = 34
VAR byte data2 = 38
VAR byte byte3 = BitAnd(data1, data2) //34
```

定义 byte 类型变量 data1，赋值 34，定义 byte 类型变量 data2，赋值 38，对 data1 和 data2 执行逻辑与操作，得到 34，赋值给 byte3。

7.5.4 BitCheck

说明

BitCheck 用于检查定义的 byte 类型数据的某一位是否为 1，若为 1 则返回 true，否则返回 false。

返回值

数据类型：bool

true 表示指定位为 1，false 表示指定位为 0。

定义

```
BitCheck (BitData, BitPos)
```

BitData

数据类型：byte

要操作的字节数据。

BitPos

数据类型：int

要操作位的位置，范围 1~8。

示例

Example 1

```
VAR byte data1 = 130
VAR bool b1 = BitCheck(data1, 8) //true
```

定义 byte 类型变量 data1，赋值 130，检测 data1 第 8 位是否为 1，返回 true。

7.5.5 BitClear

说明

通过 BitClear 可将 byte 或 int 类型的数据某一位置为 0。位数从 1 开始。

定义

```
BitClear BitData | IntData, BitPos
```

BitData

数据类型：byte

要操作的字节数据。

IntData

数据类型：int

要操作的整型数据。

BitPos

数据类型：int

要操作位的位置，对于 byte 数据来说是（1-8），对于 int 数据来说是 1-32。

示例

Example 1

```
VAR byte data1 = 255
BitClear data1 1 //254
BitClear data1 2 //252
```

定义 byte 类型变量 data1，赋值 255，对 data1 进行 BitClear 操作，将第 1 位置为 0，得到 254，将第 2 位置为 0，得到 252。

7.5.6 BitLSh

说明

BitLSh 用于对 byte 数据执行逻辑左移操作。

返回值

数据类型：byte

表示执行左移操作得到的 byte 数据。

定义

BitLSh (BitData, ShiftSteps)

BitData

数据类型：byte

要操作的字节数据。

ShiftSteps

数据类型：int

要左移的位数，范围 1~8。

示例

Example 1

```
VAR int left_shift = 3
VAR byte data1 = 38
VAR byte data2
data2 = BiLSh(data1, left_shift) //48
```

定义 byte 类型变量 data1，赋值 38，对 data1 进行左移 3 位操作，得到 48。

7.5.7 BitNeg

说明

BitNeg 用于对 byte 数据执行逻辑非操作。

返回值

数据类型：byte

表示执行逻辑非操作得到的 byte 数据。

定义

BitNeg (BitData)

BitData

数据类型：byte

要操作的字节数据。

示例

Example 1

```
VAR byte data1 = 38
VAR byte data2
data2 = BitNeg(data1) //217
```

定义 byte 类型变量 data1，赋值 38，对 data1 执行逻辑非操作，得到 217。fbyte

7.5.8 BitOr

说明

BitOr 用于对 byte 数据执行逻辑或操作。

返回值

数据类型：byte

表示执行逻辑或操作得到的 byte 数据。

定义

BitOr (BitData1, BitData2)

BitData1

数据类型：byte

要操作的字节数据 1。

BitData2

数据类型：byte

要操作的字节数据 2。

示例

Example 1

```
VAR byte data1 = 39
VAR byte data2 = 162
VAR byte data3
data3 = BitOr(data1, data2) //167
```

定义 byte 类型变量 data1，赋值 39，定义 byte 类型变量 data2，赋值 162，对 data1 和 data2 执行逻辑或操作，得到 167。

7.5.9 BitRSh

说明

BitRSh 用于对 byte 数据执行逻辑右移操作。

返回值

数据类型：byte

表示执行右移操作得到的 byte 数据。

定义

BitLSh (BitData, ShiftSteps)

BitData

数据类型：byte

要操作的字节数据。

ShiftSteps

数据类型：int

要右移的位数，范围 1~8。

示例

Example 1

```
VAR int right_shift = 3
VAR byte data1 = 38
VAR byte data2
data2 = BiRSh(data1, right_shift) //4
```

定义 byte 类型变量 data1，赋值 38，对 data1 进行右移 3 位操作，得到 4。

7.5.10 BitSet

说明

通过 BitSet 可将 byte 或 int 类型的数据某一位置为 1，位数从 1 开始。

定义

BitSet BitData | IntData, BitPos

BitData

数据类型：byte

要操作的字节数据。

IntData

数据类型：int

要操作的整型数据。

BitPos

数据类型：int

要操作位的位置，对于 byte 数据来说是（1-8），对于 int 数据来说是 1-32。

示例

Example 1

```
VAR byte data1 = 0
BitSet data1 1 //1
BitSet data1 2 //3
```

定义 byte 类型变量 data1，赋值 255，对 data1 进行 BitSet 操作，将第 1 位置为 1，得到 1，将第 2 位置为 1，得到 3。

7.5.11 BitXOr

说明

BitXOr 用于对 byte 数据执行逻辑异或操作。

返回值

数据类型：byte

表示执行逻辑或操作得到的 byte 数据。

定义

```
BitXOr (BitData1, BitData2)
```

BitData1

数据类型：byte

要操作的字节数据 1。

BitData2

数据类型：byte

要操作的字节数据 2。

示例

Example 1

```
VAR byte data1 = 39
VAR byte data2 = 162
VAR byte data3
data3 = BitOr(data1, data2) //133
```

定义 byte 类型变量 data1，赋值 39，定义 byte 类型变量 data2，赋值 162，对 data1 和 data2 执行逻辑异或操作，得到 133。

7.5.12 Break

说明

跳出当前循环，在 RL 语言中在 WHILE 循环中使用，当 WHILE 循环执行到 Break 时，不管 WHILE 的 CONDITION 如何，都会直接跳出 WHILE 循环。

示例

Example 1

```
VAR int counter = 0
WHILE(1)
```

```
IF(counter == 5)
break
Endif
counter++
ENDWHILE
```

该程序在执行到 counter 等于 5 时会跳出 WHILE 循环。

7.5.13 ByteToStr

说明

ByteToStr 用于将一个 byte 数据按照指定的格式转换成 string 数据。

返回值

数据类型：string

转换得到的 string 数据。

定义

ByteToStr (BitData [\Hex] | [\Okt] | [\Bin] | [\Char])

BitData

数据类型：byte

要转换的 byte 数据，默认按照十进制转换。

\Hex

标识符，按 16 进制转换。

\Okt

标识符，按 8 进制转换。

\Bin

标识符，按 2 进制转换。

\Char

标识符，按 Ascii 码字符格式转换。

示例

Example 1

```
VAR byte data1 = 122
VAR string str1
str1 = ByteToStr(data1) //”122”
str1 = ByteToStr(data1 \Hex) //”7A”
str1 = ByteToStr(data1 \Okt) //”172”
str1 = ByteToStr(data1 \Bin) //”01111010”
str1 = ByteToStr(data1 \Char) //”z”
```

定义 byte 类型变量 data1，赋值 122，将 data1 按不同格式转换得到不同的 string，按十进制转换得到“122”，按十六进制转换得到“7A”，按八进制转换得到“172”，按二进制转换得到“01111010”，按字符转换得到“z”。

7.5.14 CalcJointT

说明

根据指定的 robtarget 变量计算对应的关节角度。

返回值

数据类型：jointtarget
返回输入位置对应的关节角度和外部轴位置。
关节角度的单位是度（Degree），直线外部轴的单位是毫米（ mm），旋转外部轴的单位的度（Degree）。

定义

CalcJointT（Rob_Target，Tool，Wobj）

Rob_Target

数据类型：robtarget
指定的笛卡尔空间目标点，请注意该点定义时使用的工具和工件应和 CalcJointT 指令中使用的工具/工件保持一致，否则可能会导致错误的结果。

Tool

数据类型：tool
计算关节角度时使用的工具，注意需要与定义所用的 robtarget 时使用的工具一致。

Wobj

数据类型：wobj
计算关节角度时使用的工件，注意需要与定义所用的 robtarget 时使用的工件一致。

示例

Example 1

```
VAR jointtarget jpos2
CONST robtarget pt1 := [...]
jpos2 = CalcJointT(pt1, tool1,wobj2)
```

计算 tool1 到达 pt1 点时对应的关节角度，并赋值给 jpos2，pt1 点是在工件 wobj2 下定义的。

7.5.15 CalcRobT

说明

根据指定的关节角度计算对应的笛卡尔空间位姿。

返回值

数据类型：robtarget
返回给定关节角对应的笛卡尔空间位姿。

定义

CalcRobT（Joint_Target，Tool，Wobj）

Joint_Target

数据类型：jointtarget

给定的用来计算笛卡尔空间位姿的关节角度。

Tool

数据类型：tool

计算笛卡尔空间位姿时使用的工具。

Wobj

数据类型：wobj

计算笛卡尔空间位姿时使用的工件。

示例

Example 1

```
VAR robtarget pt1
CONST jointtarget jpos1 = [...]
pt1 = CalcRobT（jpos1, tool2,wobj1）
```

根据关节角度 jpos1 来计算笛卡尔位姿，并赋值给 pt1。

pt1 是工具坐标系 tool2 在工件坐标系 wobj1 下描述的位姿。

7.5.16 ChkNet

说明

用于返回网络端口 的接收缓冲器内的字符数。

定义

```
ChkNet(socket_name)
```

socket_name

socket 名称。

数据类型：string

返回值

返回接收字符数

数据类型：int

限制

Socket 未连接时，返回值为“-1”。

7.5.17 CJointT

说明

CJointT 用于读取机器人轴和外部轴当前角度。

使用 CJointT 时，机器人应处于停止状态，即 CRobT 之前的运动语句转弯区设置应为 fine。

返回值

数据类型：jointtarget

旋转轴单位：度，线性轴单位：mm
返回机器人轴和外部轴的当前角度值

定义

CJointT（）
数据类型：函数

示例

Example 1

```
VAR jointtarget j2;
j2 = CJointT（）;
```

7.5.18 ClkRead

说明

ClkRead 用于读取计时器的值。

返回值

数据类型：double
返回计时器停止时刻或当前时刻距启动 clock 的时间间隔，精度 0.001s。

定义

ClkRead（Clock）

Clock

数据类型：clock
计时器名称。

示例

Example 1

```
VAR clock clock1
ClkStart clock1
ClkStop clock1
VAR double interval=ClkRead(clock1)
```

interval 存储 clock1 在启动和停止之间的时间间隔。

7.5.19 ClkReset

说明

ClkReset 用于重置一个计时器。
使用一个计时器前可通过 ClkReset 保证计数为 0。

定义

ClkReset Clock

Clock

数据类型：clock

计时器名称。

示例

Example 1

VAR clock clock1

ClkReset clock1

重置 clock1。

7.5.20 ClkStart

说明

ClkStart 用于启动一个计时器。

当一个计时器启动后，它会不断的运行计数，直到计时器停止或程序重置。即使程序停止或机器人下电，计时器仍会继续运行。

定义

ClkStart Clock

Clock

数据类型：clock

计时器名称。

示例

Example 1

VAR clock clock1

ClkStart clock1

声明 clock1，启动计时器 clock1。

7.5.21 ClkStop

说明

ClkStop 用于停止一个计时器。

当计时器停止后，它会停止计数。计时器停止后，可以被读取间隔，重新启动或重置。

定义

ClkStop Clock

Clock

数据类型：clock

计时器名称。

示例

Example 1

VAR clock clock1

ClkStart clock1

…
ClkStop clock1
停止计时器 clock1。

7.5.22 Continue

说明

跳出本次循环。
继续从循环起始处执行下条语句，但不退出循环体，仅仅结束本次循环。

示例

Example 1

```
VAR int count = 0
WHILE(1)
count++
IF(count == 1)
Continue
Else
break
MoveAbsJ j:{ 10, 20, 30, 40, 50, 60}
Endif
ENDWHILE
```

MoveAbsJ 的代码将不会被执行到。

7.5.23 CRobT

说明

用于获取机器人位姿。
使用该函数时，需要给定工具名称和工件名称，返回指定工具坐标系的 pose，当前的轴配置信息以及外部轴位置。
使用 CRobT 时，机器人应处于停止状态，即 CRobT 之前的运动语句转弯区设置应为 fine。

返回值

数据类型：robtarget
返回当前机器人的位置、姿态、轴配置数据以及外部轴信息。

定义

CRobT（Tool，Wobj）

Tool

数据类型：tool
计算位置时使用的工具。

Wobj

数据类型：wobj

计算位置时使用的工件。

示例

Example 1

```
VAR robtarget p2;
p2 = CRobT( tool1，wobj2 );
```

7.5.24 DeactUnit

说明

用于抑制一个机械单元。

7.5.25 DecToHex

说明

将十进制的数转换成十六进制数。

返回值

数据类型：string

表示转换得到的 16 进制数据，用 0-9，a-f，A-F 表示。

参数

DecToHex(str)

str

表示要被转换的十进制数据，用 0-9 表示。

数据类型：string

使用限制

数据范围从 0~2147483647 或 0~7fffffff。

7.5.26 DoubleToByte

说明

用于将 double 或 double 数组转换成为 byte 数组。

返回值

数据类型：byte 数组

double 或 double 数组转换得到的 byte 数组，每 1 个 double 转换得到 8 个 byte。

参数

DoubleToByte(dou1)

dou1

要转换的 double 变量。

数据类型：double

7.5.27 DoubleToRobtarget

说明

将 double 类型数组赋值给 robtarget 类型数组。

返回值

数据类型：robtarget 类型数组。

参数

DoubleToRobtarget(DoubleArray,s1,e1)

DoubleArray

要转换的 double 数组。

数据类型：double

s1,e1

double 数组中索引范围。

数据类型：int

使用限制

1<=s1<=e1<=SizeOf(DoubleArray) 。

缺省则默认范围是 1~SizeOf(DoubleArray)。

示例

Example 1

```
VAR double double0[173]
VAR robtarget p1[10]
p1=DoubleToRobtarget(double0,2,171)
```

7.5.28 DoubleToSpeed

说明

将 double 类型数组赋值给 speed 类型数组。

返回值

数据类型：speed 类型数组。

参数

DoubleToSpeed(DoubleArray,s1,e1)

DoubleArray

要转换的 double 数组。

数据类型：double

s1,e1

double 数组中索引范围。

数据类型：int

使用限制

1<=s1<=e1<=SizeOf(DoubleArray) 。

缺省则默认范围是 1~SizeOf(DoubleArray)。

示例

Example 1

VAR double double0[53]

VAR speed v[10]

v=DoubleToSpeed(double0,2,51)

7.5.29 DoubleToStr

说明

将 double 类型的变量转换为 string。

参数

DoubleToStr(Val, Dec)

Val

要转换的 double 变量。

数据类型：double

Dec

要保留的小数点位数。

数据类型：int

使用限制

小数点位数最多 15 位。

7.5.30 EulerToQuaternion

说明

用于将欧拉角转成四元数。

返回值

表示转换结果，0 表示正常转换，其他-异常情况。

参数

EulerToQuaternion (type,A,B,C,q1,q2,q3,q4)

type

欧拉角顺规类型，包括 EULER_XYZ 和 EULER_ZYX。

A,B,C

要转换的欧拉角。

数据类型：double

q1~q4

转换得到的四元数。

数据类型：double

7.5.31 EXIT

说明

程序退出。

在任何用户希望程序直接退出的地方，可以使用 EXIT 指令。

EXIT 与 Return 的区别为：Return 是返回到上一级函数继续执行，而 EXIT 则不管当前在执行哪个函数，程序都将直接退出。

7.5.32 HexToDec

说明

用于将 16 进制的数转换成 10 进制数。

返回值

表示转换得到的 10 进制数据，用 0-9 表示。

参数

HexToDec(str)

str

要转换的 16 进制数据，用 0-9，a-f，A-F 表示。

数据类型：string

使用限制

- 数据范围从 0~2147483647 或 0~7fffffff。

7.5.33 HomeSet

说明

用于设置原点位置（home）的各轴角度值。

定义

（1）HomeSet axis_val1, axis_val2, axis_val3, axis_val4, axis_val5, axis_val6[,axis_val7]

（2）HomeSet()

Axis_val1~axis_val6

数据类型：int 或 double

第 1~6 轴角度值，单位度。

Axis_val7

数据类型：int 或 double

表示导轨外部轴距离值，单位 mm。

返回值

数据类型： string

如果缺省，则显示当前设置的各轴角度值，以，分隔，例如“2.2,3,1,222,0.5,99.3” 。

限制

1. HomeSet 语句执行后，会影响起始点配置界面状态；
2. 当没有设置 HomeSet 值时，执行 Home，报错“未定义原点的轴角度值”；

7.5.34 HomeSetAt

说明

用于显示当前设置原点位置（home）的各轴角度值。

定义

HomeSetAt(index)

index

关节编号， 1~9 分别表示 1~9 轴。

返回值

数据类型：int 或 double
指定关节的原点角度值。

7.5.35 Hordr

说明

用于设置执行 Home 命令时的各关节的动作顺序。用户可利用通过 Hordr 指定各步骤恢复 的关节。也可以指定多个要在各步骤进行恢复的关节。

定义

（1）Hordr val1, val2, val3, val4, val5, val6[,val7] [,val8] [,val9]
（2）Hordr()

Val1~9

数据类型：int
关节 1~9 执行恢复的步骤次序。

返回值

数据类型：string
如果省略参数，则显示当前的恢复顺序设置。

7.5.36 HordrAt

说明

用于获取指定关节在执行 Home 指令时的动作次序。

定义

HordrAt(index)

index

数据类型：int

关节编号，从 1~9 分别表示 1~9 轴。

返回值

数据类型：int

指定关节在执行 Home 指令时的动作次序。

7.5.37 HomeDef

说明

用于判断是否设置了原点位置。

定义

HomeDef()

返回值

数据类型：bool

已设置时返回“True”，未设置时返回“False”。

7.5.38 HomeClr

说明

用于清除原点位置的设置。

定义

HomeClr

提示

如果当前没有设置原点，执行 HomeClr，无效果；

7.5.39 Home

说明

将机器人机械臂移动到用户定义的原点位置

定义

Home

限制

1. 当没有设置 HomeSet 值时，执行 Home，报错“未定义原点的轴角度值”。
2. 执行 Home 操作的速度为手动模式最大速度，受运行速率百分比影响

7.5.40 IF…ELSE…

说明

条件执行语句。

定义

IF (Condition)
  program block 1
[ELSE]
  program block 2
ENDIF

如果 Condition 的条件为真，则将进入 IF 语句体内执行 program block 1 的语句，如果 Condition 的条件为假，则程序跳转到 ELSE 语句体内执行 program block 2 的语句

其中 Condition 为 bool 型变量或者表达式，ELSE 可缺省，IF…ELSE 语句中可以嵌套 IF…ELSE。

示例

Example 1

```
VAR int t = 1
t++
IF(t > 1)
t = 3
ENDIF
```

该程序执行完毕后 t 的值将变成 3。

Example 2

```
VAR int t = 1
t++
IF(t > 2)
t = 3
ELSE
t = 5
ENDIF
```

该程序执行完毕后 t 的值将变为 5。

7.5.41 IntToByte

说明

用于将 int 或 int 数组转换成为 byte 数组。

返回值

转换得到的 byte 数组，每 1 个 int 转换得到 4 个 byte。数据类型为 byte 数组。

参数

IntToByte(int1)

int1

要被转换的整数变量或数组。

数据类型：int 或 int 数组

使用限制

数据范围从-2147483647~2147483647。

7.5.42 IntToStr

说明

用于将整数转换成字符串。

返回值

转换得到的字符串。

参数

IntToStr(int1)

int1

要被转换的整数变量。

数据类型：int

使用限制

数据范围从-2147483647~2147483647。

7.5.43 MemIn

说明

用于读取一个字节的内存 IO。

返回值

表示读取得到的 byte 值，范围 0~255。数据类型：byte。

参数

MemIn(index)

index

表示内存 IO 端口编号，也就是第几个字节，最多支持 128 个字节，index 范围 0~127。

数据类型：int

示例

```
VAR byte byte1
byte1=MemIn(12)
//byte1 存储读取到的第 12 个 IO 端口的 byte 值。
```

7.5.44 MemSw

说明

用于读取某一位的内存 IO 的状态。

返回值

表示读取得到的该位的内存 IO 的状态。数据类型：bool。

参数

MemSw(index)

index

表示内存 IO 的位编号，由于最多存储 128 个字节，所以位编号范围 0~1023。
数据类型：int

示例

```
VAR bool b1
b1=MemSw(344)
//b1 存储读取到的第 344 位 IO 的状态。
```

7.5.45 MemOff

说明

用于关闭一个内存 IO 位。

参数

MemOff index

index

表示表示内存 IO 的位编号，由于最多存储 128 个字节，所以位编号范围 0~1023。
数据类型：int

示例

```
MemOff 344
//将第 344 位 IO 状态置为 0。
```

7.5.46 MemOn

说明

用于打开一个内存 IO 位。

参数

MemOn index

index

表示表示内存 IO 的位编号，由于最多存储 128 个字节，所以位编号范围 0~1023。
数据类型：int

示例

```
MemOn 344
//将第 344 位 IO 状态置为 1。
```

7.5.47 MemOut

说明

用于设置一个字节的输出数据。

参数

MemOut index, val

index

表示表示内存 IO 的位编号，由于最多存储 128 个字节，所以端口编号范围 0~127。

数据类型：int

val

表示表示要设置的值。范围 0-255。

数据类型：int 或 byte

示例

MemOn 344

//将第 344 位 IO 状态置为 1。

7.5.48 MotionSup

说明

MotionSup（*Motion Su*pervision）运动监控指令用于在 RL 程序中随时调整碰撞检测功能的各项参数。

警告

使用该指令时，必须注意以下几个事项：

- 运动监控是一系列高灵敏度的基于模型的监控功能的总称，包括负载监控、阻塞监控以及碰撞检测。由于监控功能被设计的非常灵敏，因此在正常的工艺中如果出现比较大的外力作用在机器人上，有可能引起误报；
- 如果负载没有被正确设置，需要使用负载辨识功能来定义负载以避免出现误报；
- 如果机器人使用过程中有会碰到很大的外力（例如去毛刺、打磨）时，需要通过该指令降低灵敏度或者关闭监控功能。

示例

以下是一些 MotionSup 用例：

Example 1

MotionSup On ,200

当机器人执行到本行代码时，将路径碰撞检测水平调整到 200%。

Example 2

MotionSup Off

使用指令临时关闭碰撞检测功能，直到下一次执行 MotionSup On 或者遇到其他导致参数被重置的情况。

提示

如果在系统参数中启用了碰撞检测功能，那么当出现以下情况时，所有的参数都会被

恢复为默认值：

1. RL 程序被复位，即 PP to Main；
2. 加载了新的程序；
3. 程序从头开始执行；

参数

MotionSup On（Off），TuneValue

On（Off）

碰撞检测开关

数据类型：无

On，Off 参数用于激活或者禁用碰撞检测功能。

TuneValue

检测灵敏度

数据类型：int

用于设置运动监控的灵敏度水平，可调范围为 1%~300%，数值越大灵敏度越低。

只有在使用 On 参数时，才能使用 TuneValue 参数设置灵敏度大小；使用 Off 参数时，不能使用 TuneValue。

使用限制

运动监控功能有如下使用限制：

1. 运动监控对于外部轴无效；
2. 系统参数中和 RL 指令两个设置值都会被用于计算最终的灵敏度，例如系统参数中的灵敏度水平设置为 150%，RL 指令中设置为 200%，那么实际生效的最后灵敏度水平为 300%；
3. 在系统参数中关闭碰撞检测功能后，无法在程序中使用 MotionSup 指令启用；

提示

系统会自动根据机器人的运动速度调整内部灵敏度，因此无需针对运动速度单独调整灵敏度

7.5.49 MoveAbsJ

说明

MoveAbsJ （*Move Absolute Joint*）用于把机器人和外部轴运动到一个以轴角度定义的位置上，用于快速定位或者移动机器人到某个精确的轴角度。所有的轴同步运动，机器人末端沿一条不规则的曲线移动，请注意是否有发生碰撞的危险。

MoveAbsJ 指令中使用的 tool 参数不会影响机器人的终点位置，但控制器仍然需要使用 tool 参数进行动力学的计算。

示例

以下是一些 MoveAbsJ 用例：

Example 1

MoveAbsJ j10，v500，fine，tool1
机器人使用工具 tool1，以 v500 的速度沿不规则的路径运动到 j10 定义的绝对关节角度，转弯区大小为 0。

Example 2

MoveAbsJ startpoint，v1000，z100，gripper，phone
机器人使用工具 gripper，在工件坐标系 phone 下，以 v1000 的速度沿不规则的路径运动到 startpoint 定义的绝对关节角度，转弯区大小为 100 mm。

参数

MoveabsJ ToJointPos，Speed，Zone，Tool，[Wobj]
其中，带[ ]的参数为可选参数，可不写。

TojointPos

目标关节角度（To Joint Position）
数据类型：jointtarget
机器人和外部轴的目标角度和位置值。

Speed

运动速度（Move Speed）
数据类型：speed
用于指定机器人执行 MoveAbsJ 时的运动速度，包括机器人末端的平移速度、旋转速度以及外部轴的运动速度。

Zone

转弯区（Turning Zone）
数据类型：zone
用来定义当前轨迹的转弯区大小。

Tool

数据类型：tool
执行该轨迹时使用的工具。
MoveAbsJ 指令使用工具的 TCP 数据来计算运动速度和转弯区大小。

[Wobj]

工件（Work Object）
数据类型：wobj
执行该轨迹时使用的工件。
当工具安装在机器人上时，该参数可以忽略；
当使用外部工具时，必须要指定该参数，且机器人将使用 wobj 内存储的数据来计算运动速度和转弯区大小。

7.5.50 MoveJ

说明

MoveJ （*Move The Robot By Joint Movement*）用于对机器人末端运动轨迹没有要求的场合，使机器人快速的从一个点运动到另一个点。所有的轴同步运动，机器人末端沿一条不规则的曲线移动，请注意是否有发生碰撞的危险。
MoveJ 指令与 MoveAbsJ 最大的区别在于给定的目标点格式不同。MoveJ 的目标点是工具

（TCP）的空间位姿而不是关节轴角度。

示例

以下是一些 MoveJ 用例：

Example 1

MoveJ p30，v100，z50，tool1
机器人使用工具 tool1，TCP 以 v100 的速度沿不规则的路径运动到 p30 定义的目标点，转弯区大小为 50 mm。

Example 2

MoveJ endpoint，v500，z50，gripper，wobj2
机器人使用工具 gripper，在工件坐标系 wobj2 下，TCP 以 v500 的速度沿不规则的路径运动到 endpoint 定义的目标位置，转弯区大小为 50 mm。

参数

MoveJ ToPoint，Speed，Zone，Tool，[Wobj]
其中，带[ ]的参数为可选参数，可不写。

ToPoint

目标位姿（To Point）
数据类型：robtarget
在笛卡尔空间描述的目标位置。

Speed

运动速度（Move Speed）
数据类型：speed
用于指定机器人执行 MoveJ 时的运动速度，包括机器人末端的平移速度、旋转速度以及外部轴的运动速度。

Zone

转弯区（*Turning Zone*）
数据类型：zone
用来定义当前轨迹的转弯区大小。

Tool

数据类型：tool
执行该轨迹时使用的工具。
MoveJ 指令使用工具的 TCP 数据来计算运动速度和转弯区大小。

[Wobj]

工件（Work Object）
数据类型：wobj
执行该轨迹时使用的工件。
当工具安装在机器人上时，该参数可以忽略；
当使用外部工具时，必须要指定该参数，且机器人将使用 wobj 中存储的数据来计算运动速度和转弯区大小。

7.5.51 MoveL

说明

MoveL （*Move Line*）用于将工具中心点 TCP 沿直线移动到给定的目标位置。
当起点和终点姿态不同时，姿态将与位置同步旋转到终点的姿态。
由于平移和旋转速度是分开指定的，为了保证不超出指定速度的限制，MoveL 指令最终的运动时间取决于姿态和位置变化时间较长的那一个。因此在执行某些特定轨迹（例如位移很小而姿态变化很大）时，如果机器人运动速度明显过慢或过快，请检查旋转速度设置是否合理。
当需要保持工具中心点 TCP 静止，而只调整工具姿态时，可以通过为 MoveL 指定位置相同但姿态不同的起点和终点来实现。
注意：使用 MoveL 指令无法控制第四轴转动。

示例

以下是一些 MoveL 用例：

Example 1

MoveL p10，v1000，z50，tool0
机器人使用工具 tool0，TCP 以 v1000 的速度沿直线路径运动到 p10 定义的目标点，转弯区大小为 50 mm。

Example 2

MoveL endpoint，v500，z50，gripper，wobj2
机器人使用工具 gripper，在工件坐标系 wobj2 下，TCP 以 v500 的速度沿直线路径运动到 endpoint 定义的目标位置，转弯区大小为 50 mm。

参数

MoveL ToPoint，Speed，Zone，Tool，[Wobj]
其中，带[ ]的参数为可选参数，可不写。

ToPoint

目标位姿（*To Point*）
数据类型：robtarget
在笛卡尔空间描述的目标位置。

Speed

运动速度（*Move Speed*）
数据类型：speed
用于指定机器人执行 MoveL 时的运动速度，包括机器人末端的平移速度、旋转速度以及外部轴的运动速度。

Zone

转弯区（*Turning Zone*）
数据类型：zone
用来定义当前轨迹的转弯区大小。

Tool

数据类型：tool
执行该轨迹时使用的工具，指令中的速度指的是该工具的 TCP 速度及旋转速度。

[Wobj]

工件（Work Object）
数据类型：wobj
执行该轨迹时使用的工件。
当工具安装在机器人上时，该参数可以忽略；
当使用外部工具时，必须要指定该参数，且机器人将使用 wobj 中存储的数据来计算运动速度和转弯区大小。

7.5.52 Offs

说明

位置偏移函数，用于把某个点在当前指令中指定的工件坐标系下偏移一段距离并返回新点的位置值，偏移量由 x、y、z 三个量来表示，仅支持位置平移，不支持姿态旋转。

返回值

数据类型：robtarget
偏移后的新位姿。

定义

Offs（Point，XOffset，YOffset，ZOffset）

Point

数据类型：robtarget
待偏移的位置点，或者说偏移指令的初始点。

XOffset

数据类型：double
沿工件坐标系 x 方向上的偏移量。

YOffset

数据类型：double
沿工件坐标系 y 方向上的偏移量。

ZOffset

数据类型：double
沿工件坐标系 z 方向上的偏移量。

示例

p11=Offs(p10,100,200,300)
将 p10 点沿工件坐标系的 x 方向偏移 100 mm，y 方向偏移 200 mm，z 方向偏移 300 mm，并将新的目标点位置赋给 p11 点。

7.5.53 Pause

说明

暂停程序运行。
程序会在 pause 语句的前一句执行完毕后进入暂停状态，必须使用示教器点击运行或者通过外部程序启动信号才可恢复程序运行。

7.5.54 Print

说明

将用户定义的内容打印输出到示教器，用户可以使用该函数对程序进行调试。

Print 函数的输入参数比较特殊，输入参数的个数不限，但至少要有一个，且每个参数必须为一个定义过的变量或者常量。

系统将这些变量转换为字符串并串接在一起，最后输出到程序编辑器的调试窗口。

定义

Print（var1，var2，……）

示例

Example 1

```
VAR int counter = 0
while(true)
counter++
Print(“counter = ”,counter)
endwhile
```

该程序段执行后，HMI 的程序调试窗口将打印出如下信息：

```
counter = 1
counter = 2
counter = 3
counter = 4
……
```

> 提示
>
> 当需要输出字符串时可以使用双引号“”来包含想要显示的字符，但不支持在双引号中嵌套双引号。

7.5.55 PulseDO

说明

用于产生一个脉冲的 DO 信号。

参数

PulseDO [\High,] [length,] signal

[\High]

当指令执行时，不论当前状态，始终将 signal 状态置为高（1）。

[length]

指定脉冲长度（0.001-2000s）。缺失时默认 0.2s。

数据类型：double 或 int

signal

要产生脉冲的信号。

数据类型：signaldo

使用限制

PulseDO 过程中如果执行 SetDO/SetGO，PulseDO 失效，按 SetDO/SetGO 执行。

7.5.56 QuaternionToEuler

说明

用于将四元数转成欧拉角。

返回值

表示转换结果，0 表示正常转换，其他-异常情况。

参数

QuaternionToEuler (type,q1,q2,q3,q4,A,B,C)

type

欧拉角顺规类型，包括 EULER_XYZ 和 EULER_ZYX。

q1~q4

要转换的四元数。

数据类型：double

A,B,C

转换得到的欧拉角。

数据类型：double

7.5.57 RelTool

说明

在当前指令指定的工具坐标系下对空间位置进行平移或者旋转。

与 Offs 主要有两个区别：

- Offs 是相对于工件坐标系偏移，RelTool 是相对于工具坐标系偏移；
- Offs 函数不支持对姿态进行偏移，RelTool 支持。

返回值

数据类型：robtarget

返回偏移后的新位姿。

定义

RelTool（Point，XOffset，YOffset，ZOffset，Rx，Ry，Rz）

Point

数据类型：robtarget

待偏移的位置点，或者说偏移指令的初始点。

XOffset

数据类型：double

沿工具坐标系 x 方向上的偏移量。

YOffset

数据类型：double

沿工具坐标系 y 方向上的偏移量。

ZOffset

数据类型：double

沿工具坐标系 z 方向上的偏移量。

Rx

数据类型：double

绕工具坐标系 x 轴的转动角度。

Ry

数据类型：double

绕工具坐标系 y 轴的转动角度。

Rz

数据类型：double

绕工具坐标系 z 轴的转动角度。

示例

p2=RelTool(p1,100,0,30,20,0,0)

将 p1 点沿工件坐标系的 x 方向偏移 100 mm，y 方向偏移 0 mm，z 方向偏移 30 mm，绕 x 轴旋转 20 度后，将新的目标点位置赋给 p2 点。

7.5.58 RETURN

说明

函数或 TRAP 返回。

程序遇到 RETURN 指令时，如果程序当前处于子函数或 TRAP 中，则程序将返回到上一级函数中。如果程序当前处于主函数中，则程序直接结束。

7.5.59 SearchL

说明

SearchL 用于直线运动时搜索位置。在运动时，机器人监控某 DI 信号，当信号的值改变为期望值时，机器人立即记录当前的位置，并根据开关控制机器人是否停止。

本指令只可用于运动任务。

注意：使用 SearchL 指令不支持第四轴转动的运动。

示例

例 1:

SearchL di0，p1，p2，v100，tool01

以 v100 的速度，使用 tool01 坐标系的 TCP 向目标点 p2 移动。当信号 di0 变为 True 时，将位置存储在 p1 中。

例 2:

SearchL \stop，di1，p3，p4，v100，tool01

以 v100 的速度，使用 tool01 坐标系的 TCP 向目标点 p2 移动。当信号 di0 变为 True 时，

将位置存储在 p1 中。

定义

SearchL [\Stop]|[\PStop]，Signal，SearchPoint，ToPoint，Speed，Tool [, Wobj]

[\Stop]

数据类型：标志符

当机器人搜索信号的状态变为 True 时，机器人尽快停止运动（急停），不会保持在 TCP 路径上，机器人在停止前会移动一小段距离，且不会运动回搜索位置，即信号状态改变的位置。

如果后面有指令，机器人停止后，直接跳过当前指令，执行下一条指令。

如果没有[\Stop]|[\PStop]，机器人不会停止，继续运动到目标点 ToPoint。

[\PStop]

数据类型：标志符

当机器人搜索信号的状态变为 True 时，机器人尽快停止运动，同时会保持在 TCP 路径上，机器人在停止前会移动一小段距离，且不会运动回搜索位置，即信号状态改变的位置。

[\QStop]

数据类型：标志符

当机器人搜索信号的状态变为 True 时，机器人尽快停止运动（用最快速度停止），同时会保持在 TCP 路径上，机器人在停止前会移动较小距离，且不会运动回搜索位置，即信号状态改变的位置。

QStop 相对 PStop 模式停止距离更短。

注意：QStop 能快速停止的最大速度为 v150，超速后状态不可控。

Signal

数据类型：signaldi

监控的 DI 信号名称。

SearchPoint

数据类型：robtarget

TCP 在已经触发搜索信号时的位置。

ToPoint

目标位姿（*To Point*）

数据类型：robtarget

在笛卡尔空间描述的目标位置。

Speed

运动速度（Move Speed）

数据类型：speed

用于指定机器人执行 SearchL 时的运动速度，包括机器人末端的平移速度、旋转速度以及外部轴的运动速度。

Tool

数据类型：tool

指令中机器人位置关联的工具（坐标系），工具中心点是移向指定目标点的点。

Wobj

数据类型：wobj

指令中机器人位置关联的工件（坐标系）。
缺省值为 wobj0，与世界坐标系重合。
当使用外部工具时，必须要指定该参数，且机器人将使用 wobj 中存储的数据来计算运动速度和转弯区大小。

7.5.60 SetDO

说明

设置某个数字输出信号的值。

定义

SetDO DoName | DO_ALL，Value

DoName

数据类型：signaldo
指定需要改变状态的 DO 信号名称，必须是已经在输入输出界面定义过的变量。

DO_ALL

数据类型：标识符。
指定所有的 DO 信号，必须是已经在输入输出界面定义过的变量，但不包括系统输出信号。

Value

数据类型：bool
DO 信号的目标状态，仅支持 true 和 false。

示例

SetDO do2，true
将 do2 对应的数字输出点置为高电平。

7.5.61 SetGO

说明

设置某个组输出的值。

定义

SetGO GoName，Value

GoName

数据类型：signalgo
指定需要改变值的 go 信号名称，必须是已经在输入输出界面定义过的变量。

Value

数据类型：int
go 信号的目标值。

示例

SetGO go3，8
将 go3 对应的一组物理端口的值设置为 8.

7.5.62 SocketCreate

说明

建立一个 Socket 连接，通过使用 Socket 指令，RL 程序可以从外部设备获取数据或者向外发送程序数据。

RL 语言支持同时建立多个不同的 Socket 以便于连接多个外部设备，不同 Socket 之间采用不同的名称来进行区分。Socket 指令是基于 TCP/IP 协议的，因此理论上任何支持 TCP/IP 的外部设备都可以和 RL 程序通信以交换数据。

所有发送给 RL Socket 指令的数据（即使用 SocketRead 系列指令接收的数据），都应该以结束符结尾，在接收到结束符之前的所有数据都将合并做为同一条数据处理，支持结束符自定义。使用 Socket 功能时，机器人控制器仅支持作为 client 连接外部 server。

最多支持创建 10 个 Socket 连接。

支持 socket 连续接收，socket 可接收消息队列，即一次发送多个消息，用户可使用 socketread 依次接收。队列长度最大为 100，如超出，会自动删除较早的信息。

用户可通过状态监控观察 socket 状态，包括 socket 名称，ip，端口，类型，状态，结束符。

注意：禁止频繁创建与销毁 socket。

定义

SocketCreate（“ip_Address”，Port，Name）

ip_Address

数据类型：string

定义需要连接 server 的 ip v4 地址，需使用双引号包含。

Port

数据类型：int

定义 server 端口号。

Name

数据类型：string

定义新建 Socket 的名称，不同 Socket 之间需指定不同的名称。

示例

SocketCreate（“192.168.1.100”，2020，socket0）

提示

由于 TCP/IP 协议资源释放机制限制，请不要频繁调用 SocketCreate 和 SocketClose 指令，否则可能会造成程序运行出错。

7.5.63 SocketClose

说明

关闭 Socket。

定义

SocketClose（SocketName）

SocketName

数据类型：string

需要关闭的 Socket 名称。

> 提示
>
> 不要在 SocketSend 系列指令后直接使用 SocketClose 指令，否则可能造成数据发送失败，等待收到确认消息后再使用 SocketClose 指令。

示例

SocketClose（Socket0）

7.5.64 SocketSendByte

说明

通过 Socket 对外发送一个字节 byte，在需要发送 ASCII 字符时非常有用。

定义

SocketSendByte(ByteData, SocketName)

ByteData

数据类型：int 或 byte 或 byte 数组

发送一个 0~255 的无符号字节或数组，主要用于发送 ASCII 码。

SocketName

数据类型：string

用于发送数据的 Socket 名称。

示例

Example 1

SocketSendByte(13, socket0)

通过 socket0 对外发送一个回车符。

Example 2

VAR byte data1 = 13

SocketSendByte(data1, socket0)

首先定义一个 byte 类型变量 data1，它其实是一个回车符。然后通过 socket0 对外发送。

Example 3

VAR byte data2[2] = {13,17}

SocketSendByte(data2, socket0)

通过 socket0 发送一个 byte 类型数组变量 data2。

7.5.65 SocketSendString

说明

通过 Socket 对外发送一个字符串。

定义

SocketSendString（StringData，SocketName）

StringData

数据类型：string

待发送的 string 数据。

SocketName

数据类型：string

用于发送数据的 Socket 名称。

示例

Example 1

SocketSendString（“Hello World”，Socket0）

通过 Socket0 对外发送 Hello World 字符串。

Example 2

VAR String str1 =“Hello World”

SocketSendString（str1，Socket0）

通过 Socket0 发送 str1 存储的字符串。

7.5.66 SocketReadBit

说明

通过 Socket 按 bit 接收数据，外部发送的数据需以回车结尾。

返回值

数据类型：bool

使用 bool 型数组存储接收到的 bit 数据，每一个 bit 对应一个 bool 成员。

定义

SocketReadBit(BitNum, TimeOut, SocketName)

BitNum

需要读取的 bit 数量，大小应该为 8 的整数倍。

数据类型：int

TimeOut

超时时间。单位 s，范围 0~86400，默认 60s。

数据类型：int

SocketName

用于接收数据的 Socket 名称。

数据类型：string

示例

bool groupio[16]

groupio = SocketReadBit(16, 60, Socket0)

通过 SocketReadBit 指令读取 16 个 bit 的数据存储到名为 groupio 的布尔型数组中，超时

时间 60s。

7.5.67 SocketReadByte

说明

通过 SocketReadByte 按字节读取收到的数据，并以数组形式存储。

定义

SocketReadByte(num, [timeout,]socket_name)

num

要读取的字节个数。

数据类型：int

timeout

超时时间。

数据类型：int

单位：s

缺省值：60

socket_name

套接字名称，数据类型：string，必须提前创建好，可以是 client，也可以是 server

返回值

byte 类型数组

数组长度范围：1~1000。

7.5.68 SocketReadDouble

说明

通过 Socket 接收 double 型数据，外部发送的数据需以回车结尾。

返回值

数据类型：double

使用 double 型数组存储接收到的数据。

定义

SocketReadDouble(DoubleNum, TimeOut, SocketName)

DoubleNum

需要读取的 double 数个数，最大为 30 个。

数据类型：double

TimeOut

超时时间。单位 s，范围 0~86400，默认 60s。

数据类型：int

SocketName

用于接收数据的 Socket 名称。

数据类型：string

示例

double dd[10]
dd = SocketReadDouble(10, 60, Socket0)
通过 SocketReadDouble 指令读取 10 个 double 型的数据存储到名为 dd 的 double 型数组中，超时时间 60s。

7.5.69 SocketReadInt

说明

通过 Socket 接收 int 型数据，外部发送的数据需以回车结尾。

返回值

数据类型：int
使用 int 型数组存储接收到的数据。

定义

SocketReadInt(IntNum, TimeOut, SocketName)

IntNum

需要读取的 int 数个数，最大为 30 个。
数据类型：int

TimeOut

超时时间。单位 s，范围 0~86400，默认 60s。
数据类型：int

SocketName

用于接收数据的 Socket 名称。
数据类型：string

示例

int ii[10]
ii = SocketReadInt(10, 60, Socket0)
通过 SocketReadInt 指令读取 10 个 int 型的数据存储到名为 ii 的 int 型数组中，超时时间 60s。

7.5.70 SocketReadString

说明

从 Socket 读取一个字符串并返回，外部发送的数据应以回车结尾。

返回值

存储接收到的字符串
数据类型：string

定义

SocketReadString(TimeOut, SocketName)

TimeOut

超时时间。单位 s，范围 0~86400，默认 60s。

数据类型：int

SocketName

用于接收数据的 Socket 名称。

数据类型：string

示例

VAR String str1

str1 = SocketReadString(60, Socket1)

从 Socket1 中接收一个字符串，并存储到 str1 中，超时时间 60s。

7.5.71 StrFind

说明

StrFind 用于在字符串中查找，从一个特定位置开始查找属于另一个特定字符集合的位置。

返回值

数据类型：int

表示执行查找得到的第一个字符匹配的位置。如果没有找到，返回字符串长度+1。

定义

StrFind (Str ChPos Set [\NotInSet])

Str

数据类型：string

表示要查找的字符串。

ChPos

数据类型：int

表示开始查找的位置，从 1 开始，如果位置越界，报错提示。

Set

数据类型：string

表示要匹配的字符集合。

[\NotInSet]

标识符，标识搜索不在匹配的字符集合内的字符。

示例

Example 1

VAR int found

found = StrFind(“Robotics”, 1, “aeiou”) //2

从第 1 个字符“R”开始匹配，发现第 2 个字符“o”在字符集合“aeiou”，返回匹配位置 2。

found = StrFind(“Robotics”, 1, “aeiou” \NotInSet) //1

从第 1 个字符“R”开始匹配，发现第 1 个字符“R”不在字符集合“aeiou“，返回匹配位置 1。

7.5.72 StrLen

说明

StrLen 用于获取字符串长度。

返回值

数据类型：int
表示当前字符串长度，>=0。

定义

StrLen (Str)

Str

数据类型：string
表示要计算字符个数的字符串。

示例

Example 1

```
VAR int num
num = StrLen(“Robotics”) //8
```

字符串“Robotics“长度为 8。

7.5.73 StrMap

说明

StrMap 用于创造一份 string 备份，它其中所有字符都将按照指定映射关系进行替换。映射字符根据位置一一对应，没有映射的字符保持不变。

返回值

数据类型：string
表示替换得到的字符串。

定义

StrMap (Str, FromMap, ToMap)

Str

数据类型：string
表示原字符串。

FromMap

数据类型：string
表示映射的索引部分。

ToMap

数据类型：string

表示映射的值部分。

示例

Example 1

VAR string str
str = StrMap(“Robotics”, “aeiou”, “AEIOU”) //RObOtIcs
对字符串“Robotics“进行字符映射，”aeiou”分别映射成”AEIOU“。

使用限制

FromMap 和 ToMap 必须匹配，长度一致。

7.5.74 StrMatch

说明

StrMatch 用于在字符串中搜索，从指定位置开始，搜索特定格式或字符串，返回匹配的位置。

返回值

数据类型：int
表示匹配字符串首字符所在的位置，如果没有匹配，则返回字符串长度+1。

定义

StrMatch (Str, ChPos, Pattern)

Str

数据类型：string
表示要搜索的字符串。

ChPos

数据类型：int
表示起始位置，如果超出字符串长度范围报错。

Pattern

数据类型：string
表示要匹配的格式字符串。

示例

Example 1

VAR int found
Found = StrMatch(“Robotics”, 1, “bo”) //3
从第 1 个字符开始搜索“bo“，发现第 3 个位置匹配，返回匹配的首字符所在位置 3。

7.5.75 StrMemb

说明

StrMemb 用于检查字符串中某个字符是否属于指定的字符集合。

返回值

数据类型：bool

true 表示字符串中指定位置的字符属于指定的字符集合，否则 false。

定义

StrMemb (Str, ChPos, Set)

Str

数据类型：string

表示要检查的字符串。

ChPos

数据类型：int

表示要检查的字符位置，超出字符串范围报错。

Set

数据类型：string

表示要匹配的字符集合。

示例

Example 1

```
VAR bool memb
memb = StrMemb(“Robotics”, 2, “aeiou”) //true
```

第 2 个字符 o 属于字符集合“aeiou”中一员，返回 true。

7.5.76 StrOrder

说明

StrOrder 用于比较两个字符串，并且返回布尔值。

返回值

数据类型：bool

当 str1<=str2 时返回 true，否则 false。

定义

StrOrder (Str1, Str2)

Str1

数据类型：string

表示第一个字符串值。

Str2

数据类型：string

表示第二个字符串值。

示例

Example 1

```
VAR bool le
le = StrOrder(“FIRST”, “SECOND”) //true
```

le = StrOrder(“FIRSTB”, “FIRST”) //false

7.5.77 StrPart

说明

StrPart 用于截取字符串一部分生成一个新的字符串。

返回值

数据类型：string
表示截取得到的字符串，从指定位置开始截取指定长度的字符串。

定义

StrPart (Str, ChPos, Len)

Str

数据类型：string
表示要被截取的原字符串。

ChPos

数据类型：int
表示起始截取位置，当超出字符串范围时报错。

Len

数据类型：int
表示要截取的长度。

示例

Example 1

```
VAR string part
part = StrPart(“Robotics”, 1, 5) //Robot
```

从第 1 个位置开始截取长度为 5 的字符串，得到“Robot“。

7.5.78 StrSplit

说明

对字符串进行分割，通过指定分隔符，将字符串分割成字符串数组。

定义

StrSplit(str1[, separator])

str1

待分割的字符串。
数据类型：string

Separator

分隔符，该字符串中所有字符均可作为分隔符。
缺省：以空格为默认分隔符。

限制

1. 当 str1 为空时，报错“StrSplit 中被分割字符串不能为空”。
2. 定义的字符串数组长度和实际分割得到的个数不匹配时，报错“StrSplit 函数分割得到的字符串个数与预定义字符串数组长度不匹配”；

7.5.79 StrToByte

说明

StrToByte 用于将一个特定格式的字符串转换为 byte 数据。

返回值

数据类型：byte
表示转化得到的 byte 数据。

定义

StrToByte (ConStr, [\Hex] | [\Okt] | [\Bin] | [\Char])

ConStr

数据类型：string
表示要被转换的字符串，如果可选参数不存在，默认转化为十进制。

\Hex

标识符，按 16 进制转换。

\Okt

标识符，按 8 进制转换。

\Bin

标识符，按 2 进制转换。

\Char

标识符，按 Ascii 码字符格式转换。

示例

Example 1

```
VAR byte data
data = StrToByte(“10”) //10
data = StrToByte(“AE” \Hex) //174
data = StrToByte(“176” \Okt) //126
data = StrToByte(“00001010” \Bin) //10
data = StrToByte(“A” \Char) //65
```

使用限制

1、按十进制，不能超过 0~255，超出则报错；
2、按 16 进制，不能超过 FF，超出则报错；
3、按 8 进制，不能超过 377，超出则报错；
4、按 2 进制，不能超过 11111111，超出则报错；

7.5.80 StrToDouble

说明

可以对字符串进行类型转换，转换成 double 数据。

定义

StrToDouble(str1)

str1

待转换的字符串。

数据类型：string

限制

1. 字符串前后允许有空格，可正常转换，但字符串中间不允许有空格，否则转换失败。
2. 转换失败或含非法字符应提示报错“StrToDouble 函数转换字符串失败或含有非法字符。

7.5.81 TestAndSet

说明

TestAndSet 指令可与一个标准的 bool 型变量一起作为一个二进制信号量，用作获取某项系统资源或者某段程序代码的资格。例如在不同的任务之间共享某些资源或者在同一个任务的不同级别的代码区（Trap 和 Event Routine 之间）中共享某些资源。

该指令首先测试该标志位是否为 false，如果为 false 那么将会把该标志位设置为 true 并返回 true。否则，该指令返回 false。

像示教器、文件系统与 IO 信号这些系统资源在所有任务中都是可用的。因此，必须要有一种机制，保证这些任务轮流使用这些资源，而不是同时使用同一个资源，此时 TestAndSet 指令非常有用。

示例

Example 1

```
task1
GLOBAL PERS bool flag = false
WaitUntil(TestAndSet(flag))
print(“first from task1”)
print(“second from task1”)
print(“third from task1”)
flag = false
task2
GLOBAL PERS bool flag = false
WaitUntil(TestAndSet(flag))
print(“first from task2”)
print(“second from task2”)
print(“third from task2”)
flag = false
```

两个任务都试图向示教器发送三行文字并显示。如果不使用标志位，那么将会存在这些文字混杂在一起的风险。通过使用标志位，首先执行 TestAndSet 指令的任务将首先发送三行文字，另一个程序会一直等待直到标志位被设置成 false，然后才会发送自己的三行文字。

7.5.82 WHILE

说明

循环语句，循环执行到判断条件不为真时。

定义

```
WHILE (Condition)
    Program Block
ENDWHILE
```

首先判断 Condition 是否为真，如果为真（True）则执行 Program Block 中的语句，执行完毕后重新判断 Condition 的状态，为真（True）则继续执行 Program Block 语句，为假（False）则跳出 WHILE 循环。

其中 Condition 为 bool 型变量或者表达式，WHILE 中可嵌套 WHILE 语句。

示例

Example 1

```
VAR int aa = 1
VAR int te = 1
WHILE(aa<3)
aa++
te++
ENDWHILE
```

该程序执行完毕后，te = 3

7.5.83 Wait

说明

程序等待一段时间，范围是 0~2147484 秒。

示例

Example 1

```
Wait 2
```

表示等待 2 秒的时间。

7.5.84 WaitSyncTask

说明

令用于多个任务之间在某个特殊的程序位置同步。

参数

WaitSyncTask SyncID, TaskList [\TimeOut]

SyncID

指定多个协调任务需要到达的同步点，期望的同步点上，各个任务程序中必须使用同样名称的 syncident 变量。需要在每一个任务程序中都定义一遍 global 的 syncident 变量。

数据类型：syncident

TaskList

需要协同的任务列表，必须定义为 PERS 型。必须在所有需要协同运行的任务程序中定义相同名称且包含相同内容的 PERS 类型的 tasks 变量。

数据类型：tasks

[\TimeOut]

程序的最大等待时间，单位是秒，分辨率为 0.002s。如果不指定该参数，那么程序会永远等待下去。如果时间用完仍然有程序没有到达指定的同步点，将会报错，出错的程序将会停止运行。如果出错的是 sysstop 的半静态任务，则正在执行的常规任务也停止。

数据类型：double 或 int

示例

Example 1

```
task1
GLOBAL PERS tasks task_list[2] = {“task1”, “task2”}
GLOBAL VAR syncident sync1
…
WaitSyncTask sync1, task_list
…
task2
GLOBAL PERS tasks task_list[2] = {“task1”, “task2”}
GLOBAL VAR syncident sync1
…
WaitSyncTask sync1, task_list
…
```

如上图所示的两个任务程序，第一个执行到 WaitSyncTask sync1，task_list 指令的任务将会等待，直到另一个任务执行到这一行。然后两个任务会继续执行各自之后的程序。

注意事项及使用限制

- 如果在不同任务中 WaitSyncTask 使用的 SyncID 或者 tasks 不一样，则认为是不同的任务同步组或者同步点。
- 为了保证安全的同步功能，汇合点的 syncident 必须在每一个任务中都有一个唯一的名称，且在期望的汇合点上，每个任务中包含的 WaitSyncTask 指令的 syncident 参数名称也要相同。
- WaitSyncTask 如果放在两条有转弯区的运动指令中间，会导致转弯区取消。
- 如果需要协同的任务中，存在某个任务忘记添加与其他任务相同的 WaitSyncTask 指令，那么其他任务就会永远等待下去（如果使用了 TimeOut 选项，那么会等待直到超时）。

- 如果 Tasks 变量不包含当前任务名，则运行到 WaitSyncTask 时报错。

7.5.85 WaitUntil

说明

程序等待直到某个条件成立。

示例

WaitUntil（di2 == true）
表示等待 1 号 DI 模块的第 2 个信号值为 true，然后才开始执行后面的语句。

7.5.86 WaitWobj

说明

使用该指令可以将工件队列中的第一个工件与之前定义好的随动工件坐标系建立连接。如果队列中没有工件，那么 RL 程序将持续在该指令处等待，直到出现一个可供连接的工件。如果一个 WaitWObj 已经建立了连接，那么执行第二个 WaitWObj 时 RL 将会报错。

示例

Example 1

WaitWObj wobj_on_cnv1
RL 程序连接到工件队列中的第一个工件，如果队列为空，则等待。

7.5.87 XYLim

说明

XYLim 用于设置容许动作区域，参考点为法兰坐标系原点，参考坐标系为机器人基坐标系。这样就可以根据机器人的应用来限制运转范围。

定义

XYLim x_min, x_max, y_min, y_max[, z_min, z_max]

x_min

要设置区域的 x 下限值。

x_max

要设置区域的 x 上限值。

y_min

要设置区域的 y 下限值。

y_max

要设置区域的 y 上限值。

z_min

要设置区域的 z 下限值；可省略。

z_max

要设置区域的 z 上限值；可省略。

限制

当最小值和最大值同时设置为 0 时，不限制该方向动作范围。
只在程序运行时监测。

7.5.88 XYLimClr

说明

用于清除已设置的 XYLim。

定义

XYLimClr

7.5.89 XYLimDef

说明

用于返回是否已设置 XYLim。

定义

XYLimDef()

返回值

已设置 XYLim 时返回 true，否则返回 false。

7.5.90 数学指令

sin 正弦函数

函数定义：double sin(double x)；
函数说明：sin()用来计算参数 x 的正弦值，然后将结果返回。x 单位为弧度；
返回值：返回-1 至 1 之间的计算结果。

cos 余弦函数

函数定义：double cos(double x)；
函数说明：cos()用来计算参数 x 的余弦值，然后将结果返回。x 单位为弧度；
返回值：返回-1 至 1 之间的计算结果。

tan 正切函数

函数定义：double tan(double x)；
函数说明：tan()用来计算参数 x 的正切值，然后将结果返回。x 单位为弧度；
返回值：返回参数 x 的正切值。

cot 余切函数

函数定义：double cot(double x)；
函数说明：cot()用来计算参数 x 的余切值，然后将结果返回。x 单位为弧度；
返回值：返回参数 x 的余切值。

asin 反正弦函数

函数定义：double asin(double x)；
函数说明：asin()用来计算参数 x 的反正弦值，然后将结果返回。参数 x 范围为－1 至 1 之间，超过此范围则会报错；
返回值：返回－PI/2 之 PI/2 之间的计算结果，单位为弧度。

acos 反余弦函数

函数定义：double acos(double x)；
函数说明：acos()用来计算参数 x 的反余弦值，然后将结果返回。参数 x 范围为－1 至 1 之间，超过此范围则会报错；
返回值：返回 0 至 PI 之间的计算结果，单位为弧度。

atan 反正切函数

函数定义：double atan(double x)；
函数说明：atan()用来计算参数 x 的反正切值，然后将结果返回；
返回值：返回-PI/2 至 PI/2 之间的计算结果。

sinh 双曲正弦函数

函数定义：double sinh(double x)
函数说明：sinh()用来计算参数 x 的双曲线正弦值，然后将结果返回， 数学定义式为：(exp(x)-exp(-x))/2；
返回值：返回参数 x 的双曲线正弦值。

cosh 双曲余弦函数

函数定义：double cosh(double x)
函数说明：cosh()用来计算参数 x 的双曲线余弦值，然后将结果返回，数学定义式为：(exp(x)+exp(x))/2；
返回值：返回参数 x 的双曲线余弦值。

tanh 双曲正切函数

函数定义：double tanh(double x)；
函数说明：tanh()用来计算参数 x 的双曲线正切值，然后将结果返回。数学定义式为：sinh(x)/cosh(x)；
返回值：返回参数 x 的双曲线正切值。

exp 指数函数

函数定义：double exp(double x)；
函数说明：exp()用来计算以 e 为底的 x 次方值，即 ex 值，然后将结果返回；
返回值：返回 e 的 x 次方计算结果。

log 对数函数

函数定义：double log(double x)；
函数说明：log()用来计算以 e 为底的 x 对数值，然后将结果返回。也就是求 x 的自然对数

	ln(x)，x > 0； 返回值：返回参数 x 的自然对数值。
log10 对数函数	
	函数定义：double log10(double x)； 函数说明：log10()用来计算以 10 为底的 x 对数值，然后将结果返回。其中要求 x>0； 返回值：返回参数 x 以 10 为底的对数值。
pow 求次方函数	
	函数定义：double pow(double x, double y)； 函数说明：pow()用来计算以 x 为底的 y 次方值，即 xy 值，然后将结果返回； 返回值：返回 x 的 y 次方计算结果。
sqrt 开方函数	
	函数定义：double sqrt(double x)； 函数说明：sqrt()用来计算参数 x 的平方根，然后将结果返回。参数 x 必须为正数； 返回值：返回参数 x 的平方根值。
ceil 向上取整函数	
	函数定义：double ceil(double x)； 函数说明：ceil()会返回不小于参数 x 的最小整数值，结果以 double 形态返回； 返回值：返回不小于参数 x 的最小整数值。
floor 向下取整函数	
	函数定义：double floor(double x)； 函数说明：floor()会返回不大于参数 x 的最大整数值，结果以 double 形态返回； 返回值：返回不大于参数 x 的最大整数值。
abs 求绝对值函数	
	函数定义：int abs(int x)/double abs(double x)； 函数说明：求 x 的绝对值 |x|； 返回值：当输入参数为 int 型时，输出也为 int 型。当输入参数为 double 型时，输出也为 double 型。
rand 产生随机数函数	
	函数定义： rand() 函数说明：产生一个整型随机数； 返回值：一个整型随机数，范围为 0~2147483647。

7.6 起始点

7.6.1 概述

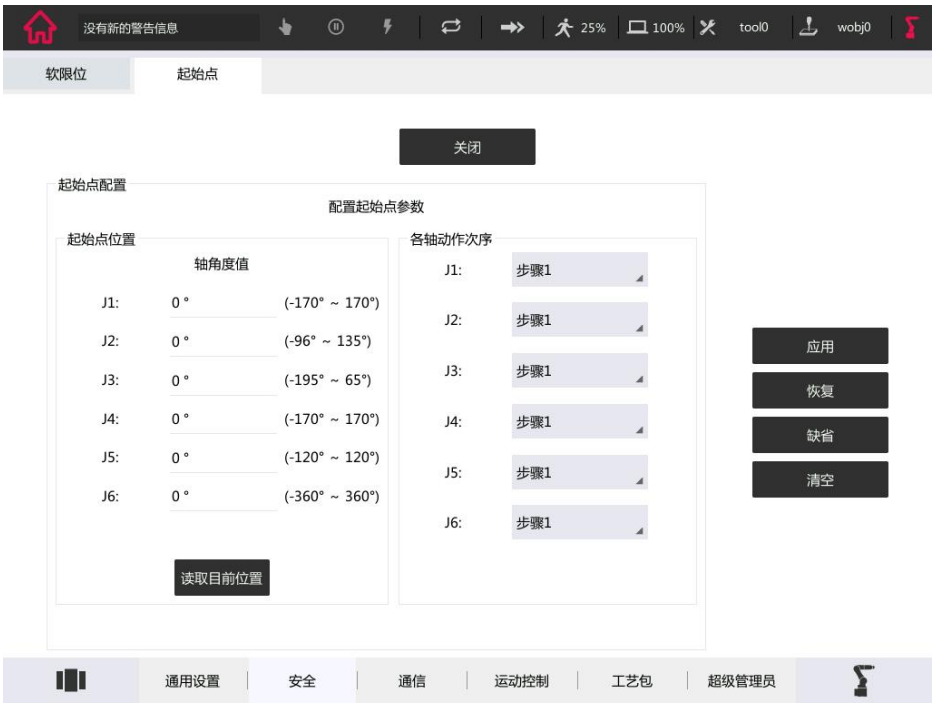
说明

用户可预先定义好“停放”或“空闲”位置，以在用户启动程序或复位重新运行程序时避免机器人与设备发生碰撞，保证初始位置的安全性。

7.6.2 参数配置

说明

需要获得 Admin 权限才可以对起始点功能进行配置，界面如下所示：



各参数的名称与含义见下表：

	参数名称	描述
1	启用按钮	确认是否启用起始点配置；如果没有启用，界面置灰，启用后，可输入轴角度值和选择各轴动作次序
2	起始点位置	通过输入框记录各轴角度值来设置起始点位置，单位：度
3	读取当前位置	点击读取目前位置按钮，可读取当前各轴角度值，并自动填入输入框
4	各轴动作次序	设置各轴动作次序
5	应用	点击“应用”后，当前配置立刻生效
6	恢复	点击“恢复”后，可恢复上一次保存的状态
7	缺省	点击“缺省”后，可恢复各轴次序缺省值
8	清空	点击“清空”后，输入框清空

提示

1.点击“应用”后，当前配置立刻生效，程序执行时以最新配置为准。
2.程序运行时，配置框不允许设置。

7.6.3 系统输出

说明

为了方便上层设备监控机器人是否在起始点位置，提供了个新的系统输出信号：

	信号名称	类型	描述
1	Home 输出状态	系统输出	当机器人到达 Home 点时，DO 输出高电平

7.6.4 姿态调整

说明

用户可通过控制面板-运动控制-姿态调整界面，点击回起始位，来验证起始位配置是否正确。

限制

执行姿态调整操作时，速度为手动模式最大速度，受运行速率百分比影响。

7.7 外部通信

说明

Titanite 系统提供了基于 Socket 的外部通信接口，上位系统（PLC、MES 等）可以通过该接口向机器人发送控制指令或者获取机器人的各种状态。

启用接口

在使用各项交互指令之前，首先需要创建用于信息交互的 Socket，该项操作在示教器的界面上操作，需要输入的参数包括上位机 IP、端口以及要创建 Socket 的名称，如下图所示：

交互命令列表

下表中给出了外部通信接口支持的信息内容及对应的命令格式，Titanite 系统使用"\r"作为指定的命令结束符（“\r”是转义字符，表示回车，十进制值是 13)。交互命令包括控制命令，配置命令和监控命令。

控制命令包括：

序号	指令名称	发送的字符串	返回值
1	电机上电指令	“motor_on”+“\r”	“true”或“false”+"\r" true :成功，false：失败
2	pptomain 指令	“pp_to_main”+“\r”	“true”或“false”+"\r" true :成功，false：失败
3	程序启动指令	“start”+“\r”	“true”或“false”+"\r" true :成功，false：失败
4	程序停止指令	“stop”+“\r”	“true”或“false”+"\r" true :成功，false：失败
5	电机下电指令	“motor_off”+“\r”	“true”或“false”+"\r" true :成功，false：失败
6	急停复位指令	“estop_reset”+“\r”	“true”或“false”+"\r" true :成功，false：失败
7	清除报警	“clear_alarm”+“\r”	“true”或“false”+"\r" true :成功，false：失败
8	关闭 socket 接口	“Titanite::SocketInterface::Disable”+“\r”	“true”或“false”+"\r" true :成功，false：失败
9	启动 socket 接口	“Titanite::SocketInterface::Enable”+“\r”	“true”或“false”+"\r" true：成功，false：失败
10	获取工程列表	“list_prog”+“\r”	程序列表，以\n 分隔，+"\r"
11	加载工程	“load_prog”+“:” +工程名+“\r”	“true”或“false”+"\r" true:成功，false：失败

配置命令包括：

序号	指令名称	发送的字符串	返回值

1	查询轴最大速度	“query_axis_max_speed”+“\r”	最大轴速度列表，以“,”分隔，+”\r”
2	设置轴最大速度	“set_axis_max_speed:”+轴最大速度列表（以:,或空格分隔，个数为 6）+“\r”	“true”或“false”+"\r"
3	查询轴最大加速度	“query_axis_max_acc”+“\r”	最大轴加速度列表，以“,”分隔，+”\r”
4	设置轴最大加速度	“set_axis_max_acc:”+轴最大加速度列表（以:,或空格分隔，个数为 6）+“\r”	“true”或“false”+"\r"
5	查询轴最大加加速度	“query_axis_max_jerk”+“\r”	最大轴加加速度列表，以“,”分隔，+”\r”
6	设置轴最大加加速度	“set_axis_max_jerk:”+轴最大加加速度列表（以:,或空格分隔，个数为 6）+“\r”	“true”或“false”+"\r"
7	查询笛卡尔空间参数	“query_space_para”+“\r”	tcp_max_speed,tcp_max_acc,tcp_max_jerk, tcp_rotate_max_speed,tcp_rotate_max_acc, tcp_rotate_max_jerk+”\r”
8	设置笛卡尔空间参数	“set_space_para:”+TCP 速度参数（以:,或空格分隔，个数为 6）+“\r”	“true”或“false”+"\r"
9	查询轴软限位	“query_axis_soft_limit”+“\r”	最大软限位列表+最小软限位列表+”\r”
10	设置轴软限位	“set_axis_soft_limit:”+最大软限位列表+最小软限位列表（以:,或空格分隔，个数为 12）+“\r”	“true”或“false”+"\r"

监控命令包括：

序号	指令名称	发送的字符串	返回值
1	电机上电状态	“motor_on_state”+“\r”	“true”或“false”+"\r" true：电机上电，false：电机未上电
2	程序运行状态	“robot_running_state”+“\r”	“true”或“false”+"\r" true：程序运行，false：程序未运行
3	急停状态	“estop_state”+“\r”	“true”或“false”+"\r" true：急停状态，false：非急停状态
4	故障状态	“fault_state”+“\r”	“true”或“false”+"\r" true：故障状态，false：非故障状态

5	工作模式	“operating_mode”+“\r”	“true”或“false”+"\r" true ：自动模式，false：手动模式/等待模式
6	路径碰撞检测状态	“path_collision_state”+“\r”	“true”或“false”+"\r" true ：功能打开，false：功能关闭
7	Jog 碰撞检测状态	“jog_collision_state”+“\r”	“true”或“false”+"\r" true ：功能打开，false：功能关闭
8	Home 输出状态	“home_state”+“\r”	“true”或“false”+"\r" true：在 home 点，false：未在 home 点
9	获取告警信息	“alarm_info”+“\r”	错误信息字符串+"\r" 错误信息字符串为当前最新报警信息的错误码，无错误时错误信息字符串为空字符串；
10	碰撞触发状态	“collision_state”+“\r”	“true”或“false”+"\r" true：发生碰撞，false：未发生碰撞或清除碰撞告警
11	获取笛卡尔位置	“cart_pos”+“\r”	笛卡尔位置字符串+"\r" 格式：x、y、z、a、b、c、q1、q2、q3、q4；其中 x、y、z 单位为 mm，a、b、c 单位为度，q1~q4 是姿态的四元数表示；
12	获取笛卡尔位置	“cart_pos_name”+“\r”	"cart_pos:"+笛卡尔位置字符串+"\r" 格式：x、y、z、a、b、c、q1、q2、q3、q4；其中 x、y、z 单位为 mm，a、b、c 单位为度，q1~q4 是姿态的四元数表示；
13	获取轴位置	“jnt_pos”+“\r”	轴位置字符串+"\r" 格式：j1、j2、j3、j4、j5、j6、[e1]，（启用导轨时，会追加导轨位置）；其中机器人轴角度单位是弧度，导轨位置单位是米；
14	获取轴位置	“jnt_pos_name”+“\r”	"jnt_pos:"+轴位置字符串+"\r" 格式：j1、j2、j3、j4、j5、j6、[e1]，（启用导轨时，会追加导轨位置）；其中机器人轴角度单位是弧度，导轨位置单位是米；
15	获取轴速度	“jnt_vel”+“\r”	轴速度字符串+"\r" 格式：vj1、vj2、vj3、vj4、vj5、vj6、[ve1]，（启用导轨时，会追加导轨速度）；其中机器人轴速度单位是弧度/秒，导轨速度单位是米/秒；
16	获取轴速度	“jnt_vel_name”+“\r”	"jnt_vel:"+轴速度字符串+"\r" 格式：vj1、vj2、vj3、vj4、vj5、vj6、[ve1]，（启用导轨时，会追加导轨速度）；其中机器人轴速度单位是弧度/秒，导轨速度单位是米/秒；

17	获取轴力矩	“jnt_trq”+“\r”	轴力矩字符串+"\r" 格式：tj1、tj2、tj3、tj4、tj5、tj6、[te1]，（启用导轨时，会追加导轨力矩），机器人轴与导轨力矩的单位都是电机额定力矩的千分比；
18	获取轴力矩	“jnt_trq_name”+“\r”	"jnt_trq:"+轴力矩字符串+"\r" 格式：tj1、tj2、tj3、tj4、tj5、tj6、[te1]，（启用导轨时，会追加导轨力矩），机器人轴与导轨力矩的单位都是电机额定力矩的千分比；

8 可选功能

8.1 功能授权

说明

Titanite 系统可选配的功能在 XB12 系列产品上，目前只支持多任务功能。
如果您需要开放可选功能，可以根据要开通的功能列表购买获取授权码，并通过通用设置-功能授权界面输入授权码，即可开启相应的可选功能。

使用限制

请勿随便输入授权码，有可能导致您已经授权的功能被关闭。如遇到上述问题，请重新输入正确的授权码即可重新开启授权功能。

8.2 多任务

8.2.1 概述

说明

多任务功能可用来在同一时间同时执行多个机器人程序，在以下场合特别有用：

- 持续监控某个信号，即使 Main 主程序已经停止运行。类似于后台 PLC 功能，但是响应速度比不上真正的 PLC。
- 在机器人执行运动主程序的时候，同时发送或者接收各种信息，而不受到主程序执行逻辑的限制。
- 机器人工作的时候通过示教器获取一些输入（待确认具体内容）。
- 控制、激活/反激活外部设备。

名词解释

Task，在 Titanite 中我们称之为任务。
Program，在 Titanite 中我们称为工程。
Module，在 Titanite 中我们称为程序文件。

8.2.2 新建任务

任务属性

参数名称	描述
任务名称	任务的名称在所有任务中必须是唯一的，任务名称仅支持字母数字下划线，首字母不能为数字，最长为 30 个字符。
前置任务	用于设置任务之间的优先级。 该参数包含了当前任务的前置任务名称。只有前置任务处于 IDLE 空闲状态时，该任务才会被执行。 如果该参数设置为空字符串，那么该任务将以最高优先级运行。
任务类型	定义任务启停和系统重启的行为，包含 2 种类型： 常规任务，系统需要手动启动或者停止（通过示教器或者外部信

	号控制），在急停时会停止运行。 半静态任务，系统重启时程序从开始重新执行。通常情况下半静态任务不会被示教器或者急停停止。 控制机械单元运动的任务类型必须为常规任务。
信任等级	信任等级定义了当一个半静态任务由于错误停止时，系统的行为： SysStop -如果该任务停止了，那么所有的常规任务将会停止，但是可以被重新启动执行，也可以 Jog 机器人。 NoSafety -只有当前的任务会停止执行。
是否运动任务	标识该任务是否可以使用 RL 指令来控制机器人的运动。 只能有一个任务是运动任务。

新建任务

新建任务时应确保资源管理器中已经存在至少 1 个工程。

使用限制

- 最多支持 10 个任务。
- 最多只有一个运动任务。
- 更改任务类型、前置任务、任务入口函数、是否运动任务属性即时生效。

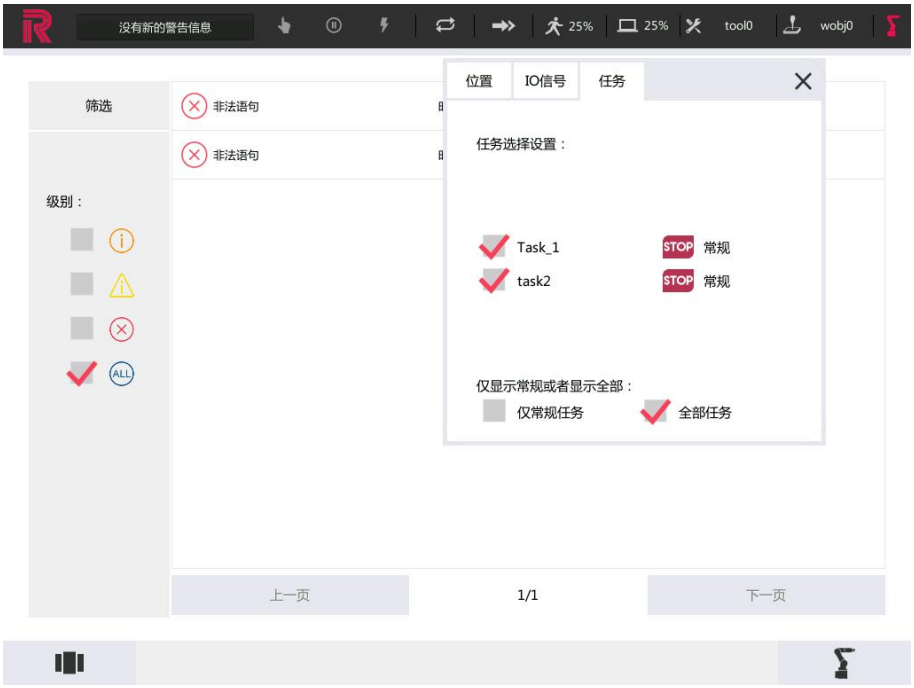
8.2.3 任务看板

说明

通过任务看板可以查看任务运行状态，设置哪些任务可以被示教器上的开始和停止按钮控制。

打开状态监控窗口，切换到任务 Tab 页面，查看任务状态或执行各项操作，选中或取消选

中状态即时生效。



使用限制

- 当控制器重启时，所有常规类型的任务会保持重启前的状态而所有的半静态任务在任务面板上将被取消选中。
- 如果在“任务看板设置”中选择了“全部任务”，那么所有半静态的任务也可以被选择是否可以被“开始”按钮启动、正向单步、逆向单步以及被“停止”按钮停止。
- 如果在“任务看板设置”中选择了“仅常规任务“，那么在“任务看板”中所有半静态任务就会被置灰不可选，自动模式下，按下“开始”按钮后，所有的半静态任务就会被启动。
- 在任务面板设置页面，可以更改成使半静态也可以被启动和停止按钮控制，但是这两种任务的信任等级 TrustLevel 必须设置为 NoSafety，否则系统会报错停止。

8.2.4 启动和运行任务

说明

在任务看板选择想要通过启动/停止按钮或者外部启动信号启动的任务，即被对勾选中的任务。在手动使能或者自动上电情况下，使用启动/停止按钮或者外部信号控制选中任务的启停。

当切换到自动模式时，在任务面板中所有的半静态任务将被取消选中。处于停止状态的半静态任务将会在下一次按下“启动”，“单步”按钮时被启动。这些任务将一直循环运行，不会被“停止”按钮以及急停按钮打断。

当控制器重启时，所有普通类型的任务会保持重启前的状态而所有的半静态任务在任务面板上将被取消选中。系统启动完成后，所有的半静态任务将会自动启动并开始周期运行。

任务运行优先级

可以通过为某项任务添加前置任务来划分任务间的优先级，来控制不同任务的执行顺序。

只有当某任务的前置任务处于空闲（IDLE）状态时，该任务才会被执行。空闲状态的可能例子包括：

- 前置任务在等待一个事件、IO 等；
- 前置任务执行了一个运动指令，在等待机器人完成运动时；

使用限制

- 通常情况下，一个后台任务会一直循环运行。如果一个任务中不包含任何等待指令，那么后台任务可能会消耗过多的计算器资源而导致控制器无法处理其他的任务。
- 变量 VARS 和常量 Const 作用域限制在各自的任务中，但是 GLOBAL 级别的 PERS 变量为全局变量。
- 当执行 PP to Main 时，所有没有运行的任务都执行 PP to Main。
- 对于半静态任务而言，只能使用循环运行模式。单次模式仅适用于普通类型的任务。
- 半静态任务的启动、单步、停止仅限于手动模式，自动模式下无法对半静态任务进行操作。
- 有任务运行时，禁止修改“多任务”界面中的内容。
- 前置任务嵌套最多支持两层。
- 运动任务不支持定义前置任务。

8.2.5 任务间通信

说明

任务间通信需通过 PERS 变量。

任务间使用 PERS 变量通信

- 在所有需要进行通信的任务工程中都定义相同名称的 GLOBAL 级别的 PERS 变量，且变量的数据类型、维数均应相同。
- 在需要的地方使用 PERS 变量来控制任务执行、传递数据等。

使用限制

- 只需在其中一个任务中为 PERS 变量指定初始值即可。如果在多个任务中为同一个 PERS 变量指定了初始值，那么将使用第一个运行的任务中定义的初始值。
- 通过 PERS 变量以及 WaitUntil 或 WHILE 指令的方式让一个任务等待另一个任务时，需要注意配合等待指令（大于 0.1s），避免程序快速执行空判断语句导致占用过多系统资源。

9 故障排查

排查方式

Titanite 系统提供了两种故障排查方式，分别为：

根据故障现象，如果错误发生在系统启动过程中，或者没有相应的日志产生，则可以按照错误现象来进行问题排查，详细内容请参考按故障现象章节。

根据日志编号，如果发生故障时在系统产生了相应的错误日志，则可以通过示教器的 HMI 界面查看日志的详细说明来确认与排除故障。

9.1 按故障现象

序号	现象描述	可能的原因	推荐的操作
1	开机后系统没有响应，指示灯不亮，散热风扇不启动	电源线没有插好 电源电压等级符合要求 供电回路的空开被断开 电路系统故障	检查电源线连接正常 检查电压等级是否符合要求 断开控制柜外部电源，检查柜内供电回路的空气开关 联系售后人员更换损坏部件
2	示教器提示“连接主控制器失败“	主控制器 ip 设置错误 示教器电缆没有插好 示教器电缆损坏	输入正确的控制器 ip，Titanite 主控制器默认 ip 为 192.168.1.160 检查示教器电缆是否插好 检查示教器电缆是否有明显的外部损伤，如有，则需要更换示教器电缆 如果可能，将示教器连接到另外的控制柜上以排除单独的控制器故障
3	示教器无法启动	示教器电缆损坏 控制柜内给示教器供电的回路故障 示教器损坏	检查示教器电缆是否有明显的外部损伤，如有，则需要更换示教器电缆 检查柜内示教器供电回路 更换示教器 如果可能，将示教器连接到另外的控制柜上以排除单独的控制器故障
4	机器人无法进行 Jog	检查机器人是否处于手动模式 模式选择开关的实际状态与示教器软件的状态不符 示教器按键故障 示教器软件故障	将机器人切换到手动模式 更改模式选择开关状态使之与软件保持一致 更换薄膜按键 使用恢复 U 盘对示教器进行恢复操作
5	机器人运动时有异常噪音	减速器出现磨损或者损坏 电机抱闸故障导致没有正确的释放 控制柜伺服驱动配置与实际机器人不符 减速器过热	检查减速器，必要时更换 检查抱闸 检查控制器软件配置是否与实际机器人相符 对节拍要求高的程序可能会使某些关节满负荷长期工作，在允许的情况下适当降低运行速度或者尝试改变机器人姿态。
6	机器人有漏油现象	关节密封圈损坏 由于安装时润滑油填充过量	参考机械维护手册，检查密封装置 如仅有少量润滑油渗出，擦拭后不再出现，则暂时不用处理，继续使用并观察即可
7	断电后，机器人无法维持位置，发生坠落	电机抱闸损坏 电机抱闸供电回路故障	联系售后人员更换电机 检查并修复供电回路故障

修订记录

版本	修订内容	修订时间
A	初版发行	2020.08.17
A	添加 XB12 系列机器人控制柜功率； 更新 4 轴相关的指令说明。	2021.01.11

ROKAE 珞石

轻 型 机 器 人 专 家

北京总部：
北京市海淀区上地四街一号院四号楼
山东分公司：
济宁市邹城市中心店镇机电产业园恒丰路 888 号
苏州分公司：
苏州工业园区星湖街 328 号创意产业园 1-A1F
深圳分公司：
深圳市宝安区中粮福安机器人智造产业园 10 栋 1 楼

网址：http://www.rokae.com
热线：400-010-8700

公众号：ROKAE 珞石
微信号：Rokae-tech