ROKAE 珞石
轻型工业机器人专家

xCore

机器人控制系统使用手册

文档版本：V1.6.1

发布日期：2022.09

本手册适用于 xCore V 1.6.1，详见本手册版本信息章节，使用前请仔细核对实际产品版本信息，确保一致

本手册中包含的信息如有变更，恕不另行通知，且不应视为珞石的承诺。珞石对本手册中可能出现的错误概不负责。

除本手册中有明确陈述之外，本手册中的任何内容不应解释为珞石对个人损失、财产损坏或具体适用性等做出的任何担保或保证。

珞石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概不负责。

可通过联系珞石技术支持工程师以获取此手册的纸质复印件。

目录

1 文档清单

xCore 控制系统包含完善的基本功能，以及丰富的扩展功能。为了让您更快掌握 xCore 的使用，珞石机器人可以提供以下文档，如有需要，可以联系官方获取。

ID	名称	最新版本号	简介
1	《xCore 机器人控制系统使用手册》	V1.6.1	介绍 xCore 控制系统基本功能使用；
2	《xVision 使用手册》	V1.0.0	介绍 xVision 机器人视觉基本使用；
3	《激光焊接工艺包使用手册》	V3.0	介绍激光焊接工艺包使用；
4	《xCore 控制系统扩展功能使用手册》		介绍 RCI、xCore-SDK 的使用；
5	《电镀线跟踪功能操作手册》	V1.2	介绍电镀线跟踪功能的使用；
6	《RokaeStudio 用户手册》	V1.1.0	介绍离线编程软件使用；

2 名词术语

- HMI：Human Machine Interface 人机交互界面；
- HMID：HMI device 人机交互设备；
- RCI：Rokae Control Interface 珞石机器人外部控制接口，支持底层实时控制；
- SDK：Software Development Kit，软件开发工具包，支持通过 C++等语言实现对机器人的底层控制，后续将逐步取代 RCI；
- Project：工程，控制机器人运行的程序、任务等对象的管理集合；一个工程的数据对象可以导出，在其他工程或机器人复用；
- Task，在 xCore 中我们称为任务；
- Module，在 xCore 中我们称为程序文件；
- Elbow：臂角，指的是臂平面与参考平面之间的夹角，臂平面指的是机器人大臂与小臂所组成的平面，参考平面是指，三轴为零时末端达到同样位姿时所形成的臂平面；
- RL：Rokae Robot Language，珞石机器人语言，包含丰富的指令，可以基于 RL 搭建机器人任务；
- RobotAssist：珞石机器人推出的一款集示教器功能为一体的软件，搭配新一代控制系统 xCore，具有机器人移动控制、程序编写、参数配置、状态监控等功能，可以运行于 xPad2 示教器、PC 等设备上；

3 简介

概述

Robot Assist 是珞石机器人推出的一款集示教器功能为一体的软件，搭配新一代控制系统 xCore，具有机器人移动控制、程序编写、参数配置、状态监控等功能，界面友好，适用于珞石机器人推出的全系列机器人（包括工业机器人和协作机器人），并持续更新以支持后续开发的新型号。该软件可安装于 PC、Surface、珞石示教器 xPad2 上，只要与机器人处于同一网段内，连接机器人便可操控机器人。

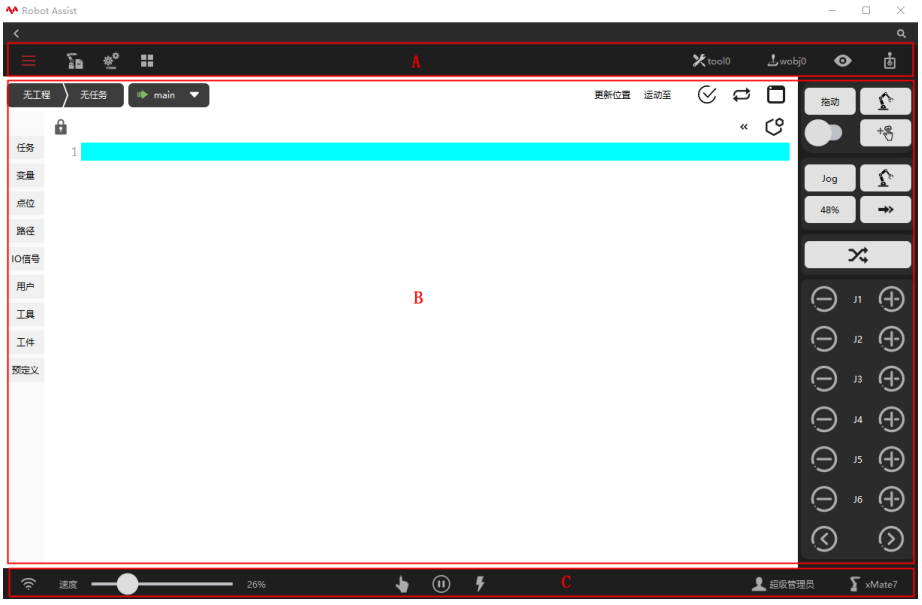
运行环境

若选择不使用 xPad2，建议您使用平板电脑或者笔记本电脑作为操作终端，推荐配置如下。

终端类型	平板电脑	终端类型	电脑
存储容量	32GB	存储容量	32GB
系统内存	4GB	系统内存	4GB
屏幕尺寸	8.0 寸及以上	显卡	Intel HD Graphics 4000 或更高
网络通信	Wi-Fi	网络通信	Wi-Fi 或有线网卡
操作系统	Windows7 64bit ， Windows10 64bit，Ubuntu16.04，Ubuntu18.04		
处理器	Intel Core I3 及以上		

3.1 主界面

操作主界面有三个区域，包括顶部状态栏、主显示区、底部状态栏。



A	顶部状态栏，包括：显示菜单按钮、机器人编程按钮、机器人配置按钮、示教器选项按钮、即时日

	志、工具工件信息、状态监控面板入口按钮和操控面板入口按钮。
B	主显示区，根据不同功能模块显示具体的操作界面。
C	底部状态栏，显示连接状态、程序运行速度倍率、工作模式、运动状态、登录用户信息及机器人型号信息。

提示

RobotAssist 软件与控制器实时交互，频繁改变窗口大小可能导致监控数据停止更新，此时点击 Alt+Tab 切换窗口即可恢复。

3.1.1 顶部状态栏

说明

顶部状态栏，显示主菜单入口按钮、当前工程、即时日志、工具信息、工件信息、机器人状态监控面板入口按钮和机器人操控面板入口按钮。

A	菜单按钮，点击可选择诊断日志、帮助、演示功能模块，跳转到功能子界面。
B	机器人编程按钮，点击可跳转到当前工程子界面。
C	机器人配置按钮，点击可跳转到设置、安全、通信、授权功能模块。
D	示教器选项按钮，点击可跳转到连接、基本设置、外观调整、文件管理功能模块。
E	工具选择，显示当前使用工具信息，可选择使用工具。
F	工件选择，显示当前使用工件信息，可选择使用工件。
G	状态监控面板入口按钮，点击可打开、关闭状态监控面板。
H	操控面板入口按钮，点击可打开、关闭操控面板。

3.1.2 底部状态栏

说明

底部状态栏显示 RobotAssist 软件与机器人连接状态、程序运行速率、机器人工作模式、机器人状态、机器人电机状态、当前用户登录信息和机器人型号信息。

A	RobotAssist 软件与机器人的连接状态，出现红色斜杆为未连接，全灰色为已连接。
B	程序运行速率调整滑块，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。
C	机器人工作模式，分为手动和自动，点击按钮可进行模式切换。
D	机器人状态，包括机器人运动状态、机器人系统状态、控制器模式等。
E	机器人电机状态，红色为上电，灰色为下电，其他包括紧急停止、安全门打开。自动模式下点击按钮可给机器人上电。
F	当前登录用户信息：操作员、管理员、超级管理员。点击按钮可跳转用户登录界面，默认登录密码 123456。
G	机器人型号信息。

机器人工作模式

 手动模式下进行程序编写及调试。

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。

 自动模式下进行连续自动化生产。

在该模式下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 DI 信号来控制电机上电。

机器人状态

底部状态栏  显示当前机器人状态，包括机器人运动状态、机器人系统状态、控制器模式等。

空闲状态		程序停止状态，机器人未运动。
程序运行		程序运行中，当机器人运动会变成红色。
拖动模式		控制器处于拖动模式下，可以进行拖动，当机器人运动会变成红色。
Demo 模式		控制器处于 Demo 演示模式下，可以开始 Demo 演示，当机器人运动会变成红色。
辨识模式		控制器处于辨识模式下，当机器人运动会变成红色。
Jog 模式		控制器处于 Jog 模式下，随 Jog 按钮开始、结束而改变。
RCI 模式		控制器处于 RCI 模式下，当机器人运动会变成红色。
协作模式		控制器处于协作模式下，会与其他状态进行组合显示再图标的右上角
错误状态		机器人系统处于错误状态
调试模式		机器人系统处于调试模式，当机器人运动会变成红色

机器人电机状态

上电状态		机器人电机处于上电状态
下电状态		机器人电机处于下电状态
急停状态	E	机器人处于紧急停止状态，此时无法给机器人电机上电
安全门状态	G	机器人处于安全门打开状态，此时无法给机器人电机上电

3.2 状态监控

说明

通过顶部状态栏状态  可打开浮动状态监控面板，状态监控用于监控机器人 3D 模型、任务运行状态、IO 信号、网络连接状态、寄存器变量，方便 Jog 和编程时使用。

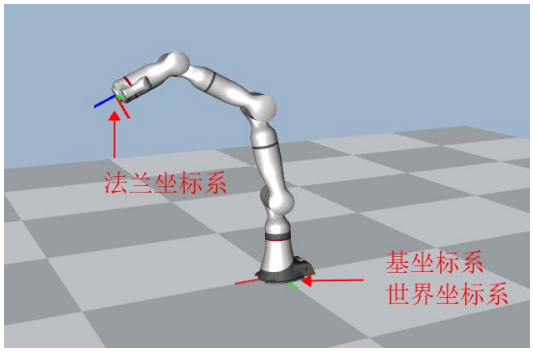
3.2.1 3D 模型监控

说明

3D 模型监控界面显示机器人当前的三维模型、关节角度、关节转矩、机器人臂角、机器人末端在某坐标系下的位置、机器人末端相对于某坐标系的旋转矩阵的 RPY 角和四元数。

机器人三维模型上叠加有三个坐标系的显示：法兰坐标系、基坐标系、世界坐标系。当进行机器人基坐标系标定后，3D 模型监控界面，基坐标系相对世界坐标系会根据标定结果相应变化。

末端位置可以选择在三个坐标系（工件坐标系、基坐标系、世界坐标系）下显示；默认显示基坐标系下的监控数据。



3.2.2 多任务监控

说明

多任务监控界面显示当前工程中各个任务的任务类型、运行状态。

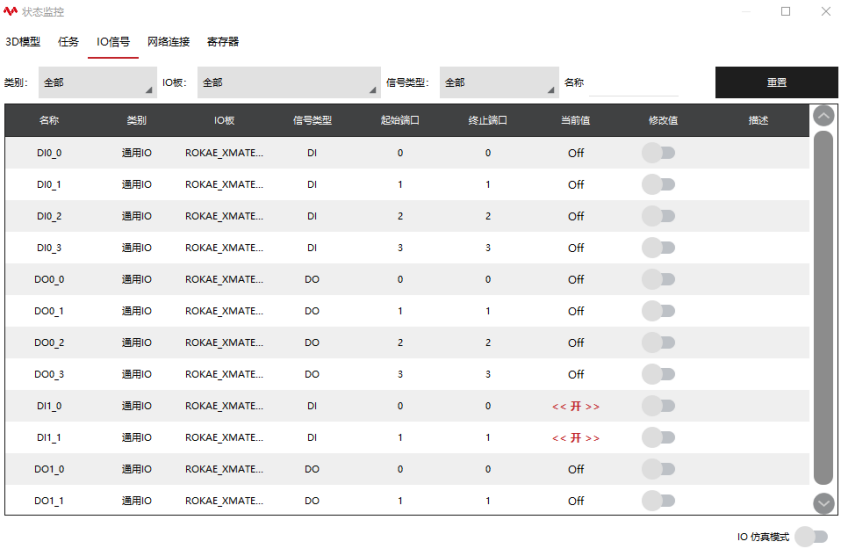
	操作	说明
1	任务名称	在工程中添加的任务，在此处显示。
2	任务类型	可以在工程->任务列表中进行配置。
3	开始运行程序，能够观察到任务的运行状态。	任务停止：STOP 任务运行：GO

3.2.3 IO 信号监控

说明

对于 xMate 协作机器人，IO 信号监控界面默认显示本体底座上的 4 路 DI、DO 信号以及末端 2 路 DI、DO 信号；对于工业机器人，IO 信号监控界面默认显示控制柜内已配置的 IO 信号。

可打开 IO 仿真模式，测试 DI/DO 信号值。



	操作	说明
1	进入 IO 信号页面，点击【IO 仿真模式】使能按钮，开启仿真模式。	有权限限制，需要是 Admin 或者 God 用户才能使用。
2	点击相应 DI 或者 DO 所对应的修改值按钮，可以对	注意，非仿真模式 DO 也可以强制输出。

	其进行仿真。	
3	点击【IO 仿真模式】使能按钮，关闭仿真模式。	实际值和修改值按钮没有强关联，关闭仿真模式后，修改值按钮会统一置为 false。

3.2.4 网络连接监控

说明

网络连接监控界面显示当前与控制器已建立的网络连接信息（IP、端口号）和状态，默认显示 SOCKET、 MODBUS、 RCI 等连接状态。

	操作	说明
1	【类别】可支持显示 MODBUS、RCI、SOCKET 等类型连接。	相应连接可以在相关界面进行增加或配置，SYS_SOCKET 特指外部通信对应连接。
2	【名称】连接的名称。	MODBUS、RCI、SYS_SOCKET 为固有名称，属于系统默认，用户新建的会显示其自定义名称。
3	【IP】连接对应的 IP。	如果是客户端性质的连接会显示目标服务端的 IP，如果是服务端会显示自身 IP。
4	【端口】连接对应的端口。	如果是客户端性质的连接会显示目标服务端的端口，如果是服务端会显示自身端口。
5	【状态】连接当前状态。	一般有三种，已连接、已关闭、正在连接；如果是服务端性质连接未被连接时会显示正在监听。

3.2.5 寄存器变量监控

说明

寄存器变量监控界面显示各寄存器的信息，可根据内容过滤器进行快速查找。

	操作	说明
1	【内容过滤】用户可以自定义想要展示的内容。	通过选择连接、变量类型、名称、描述等可以筛选包含相应内容的变量，方便用户查看。
2	各列含义可参考 modbus 寄存器变量配置。	

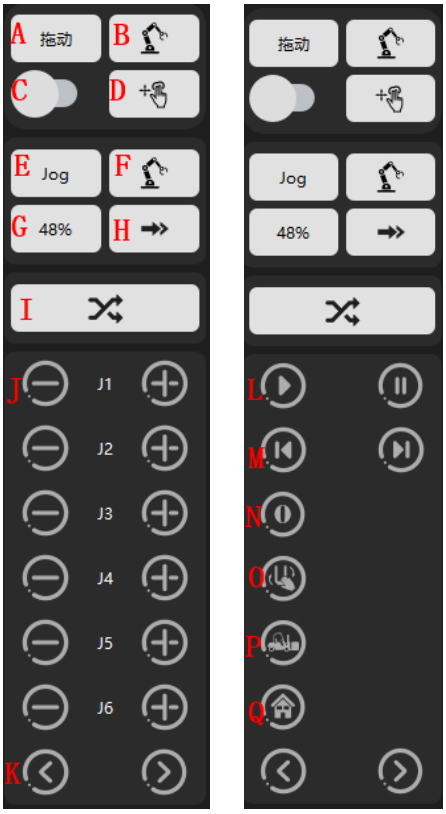
3.3 操控面板

说明

通过顶部状态栏  或  可打开操控面板，操控面板用于切换机器人控制模式，对机器人进行移动控制，以及位姿示教。

机器人支持两种方式示教机器人位姿：Jog 模式和拖动模式（仅限 xMate 协作机器人）。

- Jog 模式，通过 Jog 按钮控制机器人在相应方向运动。
- 拖动模式，通过末端拖动 Pilot/xPanel 手柄，直接手动引导机器人运动。



编号	说明
A	图标，说明 BCD 跟拖动相关。
B	拖动空间设置：轴空间拖动、笛卡尔空间拖动。
C	拖动使能开关按钮，用于开启、关闭拖动模式。
D	拖动方式设置。 轴空间仅支持：自由拖动；笛卡尔空间支持三种：仅平移拖动、仅旋转拖动以及自由拖动。
E	图标，说明 FGH 跟 Jog 相关。
F	Jog 参考系设置，用来选择 Jog 时的单轴模式和笛卡尔模式以及在笛卡尔模式下的参考坐标系，包括：世界坐标系、基坐标系、法兰坐标系、工具坐标系、工件坐标系。
G	Jog 速率设置，用来调节 Jog 时的运动速率，调整范围 1~100%，为相对 Jog 最高速度限制 250mm/s 的百分比。
H	Jog 步进模式设置，可选连续 Jog 和步进 Jog，并且可以调整增量步进的大小。
I、K	功能区切换，可以进行 Jog 按钮与 L~Q 按钮的切换。
J	Jog 按钮。对于 7 轴机器人，轴空间 Jog 时显示 J1~J7，笛卡尔空间 Jog 时显示 X/Y/Z/A/B/C 及 Elbow；对于 6 轴机器人，轴空间 Jog 时显示 J1~J6，笛卡尔空间 Jog 时显示 X/Y/Z/A/B/C。
L	程序启动、停止按钮。
M	程序运行上一步、下一步按钮。
N	运动至“回零位姿”按钮。
O	运动至“拖动位姿”按钮。
P	运动至“发货位姿”按钮。
Q	运动至“自定义 home 点位姿”按钮。

提示

请您在 JOG 和打开拖动使能开关前，确认当前机器人工作模式为  手动模式且  下电状态。

3.4 功能模块

说明

通过菜单按钮  可打开功能选项卡，功能模块主要包括工程模块、机器人模块、诊断模块、演示模块、选项模块、帮助模块，点击主菜单按钮可切换不同功能模块选项卡。

3.4.1 菜单模块

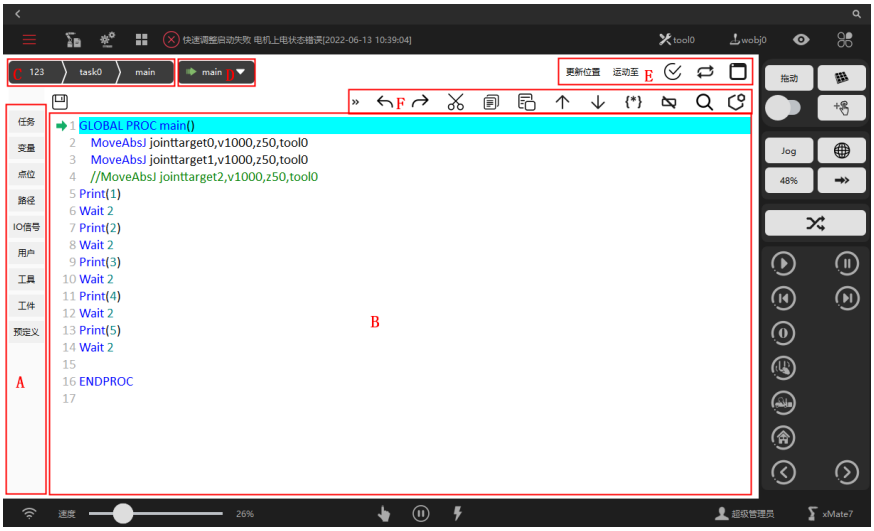
诊断日志	
示教器日志	显示当前操作界面日志信息；
控制器日志	显示当前连接机器人控制器运行日志；
日志时间轴	通过时间轴直观显示日志历程；
内部日志	显示用于显示控制器的底层日志信息。机器人出故障时，技术支持可通过查看内部日志快速定位问题原因；
高级选项	辅助用于辅助开发人员进行诊断伺服、ECAT 等设备问题，及开启实时线程告警监测等功能。由于开启诊断功能后会加重控制器运行负荷，所以在实际生产环境中非必要不要打开。

帮助	
关于珞石	介绍了界面、控制器版本以及珞石机器人的简要介绍
产品介绍	包含工业机器人、协作机器人全系列技术参数
软件升级	提供了控制器升级、控制器备份、恢复出厂设置、重启机器人、抹除机器人配置信息等功能

演示
提供四个演示 demo 以展示 xMate 7 自由度设计，冗余运动控制的灵动性；一指触停的灵敏性；变阻抗刚度的柔顺性等特性。

3.4.2 机器人编程模块

界面功能划分



A	界面选项卡，用于切换工程子对象设置界面，包括任务、变量、点位、路径、IO 信号、用户坐标系、工具坐标系、工件坐标系、预定义等；
B	程序编辑区，对机器人进行辅助编程以及程序命令编辑；
C	程序文件选择，用于不同任务不同程序文件之间的编辑调试切换；
D	程序调试快速定位按钮，可选择快速定位至 main 函数、快速定位至光标；
E	程序语法检查、循环模式、输出终端；
F	程序编辑工具栏：撤销、重复、剪切、粘贴、复制、上移一行、下移一行、批量注释、删除当前行、搜索替换、辅助编程；

3.4.3 机器人配置模块

设置	用户组、控制器设置、零点标定、基坐标系标定、动力学参数辨识、本体参数、运动参数、力控参数、快速调整；
安全	软限位、虚拟墙、碰撞检测、安全区域、安全监视器、协作模式；
通信	系统 IO、外部通信、寄存器、IO 设备、总线设备、末端工具、RCI 设置、串口设置；
工艺包	激光焊接等；
授权	EtherCAT 授权；

3.4.4 示教器选项模块

连接	包括机器人探测、机器人连接、自动重连设置；
基本设置	包括软件语言设置、开机自启动设置、绑定 IP、工作区路径选择、图形性能调整、关闭 3D 显示；
外观调整	包括主题切换、控件调节、字体调节；
文件管理	可以打开缓存文件夹、日志文件夹、工作区文件夹进行浏览查看；

4 连接机器人

说明

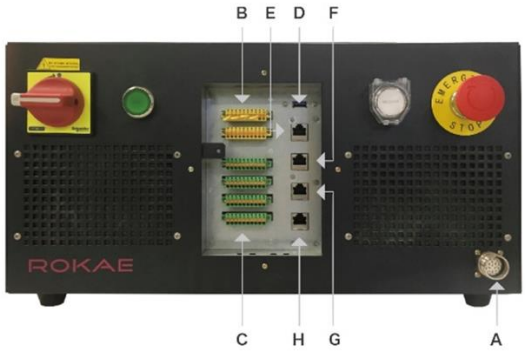
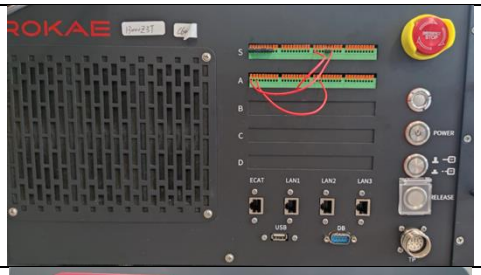
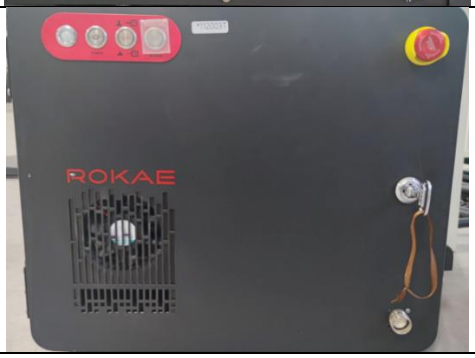
通过运行 Robot Assist 软件的设备(xPad2 示教器、PC)可连接任意一台珞石机器人，只需将 Robot Assist 软件所在设备、以及机器人接入同一局域网，可选择机器人探测、手动输入控制器服务地址等方式，与相连接的机器人建立连接。

机器人系统只支持有线形式接入到局域网中，或使用网线直接连接：

- 对于使用示教器 xPad2 的情况，直接将 xPad2 连接到机器人对应的插口即可；
- 对于使用运行 RobotAssist 的 PC 的调试机器人的情况，PC 可以直接通过网线与机器人系统连接；
- 对于需要切换多台机器人连接的，可以将多台机器人接入同一局域网，运行 RobotAssist 的 PC 通过同网段探测，寻找可连接的机器人；
- 对于 AGV 搭载等不方便有线连的场景，可以通过机器人控制柜预留的网口（xMate 协作机器人底座网口，工业机器人控制柜视觉/调试网口）与无线路由连接后，再与 HMID 通过无线连接；

4.1 机器人网口及 IP

ID	简介	配图
1	**xMate ER** 系列协作机器人，底座配有两个以太网口，其中 J2 网口出厂默认配置为固定 IP 192.168.0.160，J1 网口出厂默认配置为自动获取。	
2	**xMate CR** 系列协作机器人，底座标配的以太网口只有一个 J1，出厂默认配置为固定 IP 192.168.0.160	

3	采用 **XBC3** 控制柜的工业机器人，控制柜上有四个以太网口纵向排列，自上而下依次为：EtherCAT 设备扩展网口，用于从站拓展；视觉网口，用于连接工业相机，默认配置为固定 IP 192.168.2.160；调试网口，默认配置为固定 IP 192.168.0.160；总线扩展网口，选配；	
4	采用 **XBC5** 控制柜的工业机器人，控制柜上有四个以太网口横向排列，自左至右依次为：EtherCAT 设备扩展网口，用于从站拓展；视觉网口，用于连接工业相机，默认配置为固定 IP 192.168.2.160；调试网口，默认配置为固定 IP 192.168.0.160；总线扩展网口，选配；	
5	采用高防护等级的 **XBC5** 控制柜，控制柜上无预留网口；	

4.2 终端设备连接准备

ID	终端设备	描述
1	xPad2	示教器 xPad2 支持适配 xMate CR 系列协作机器人和采用 XBC5 控制柜的工业机器人。您可以将示教器 xPad2 通过线缆插头连接到机器人上（对于工业机器人，插口位于 XBC5 控制柜上；对于 xMate 协作机器人，插口位于底座上），即可完成示教器的连接。机器人开机后，示教器 xPad2 将自动开机，并自动启动预装的 Robot Assist 软件。
2	PC	方式一：网线直连 机器人底座或控制柜上都有一个网口预设为调试网口，配置固定 IP 地址为 192.168.0.160，该地址在所有机器人上均相同，且不建议轻易修改。您可以将运行 RobotAssist 的 PC 通过网线与此网口直连，控制机器人。 方式二：外网口连接 外网口连接机器人支持两种方式，自动获取 IP 地址或者设置静态 IP 地址的方式。 需要自动获取地址时，协作机器人可将 J1 网口设置为 DHCP 模式，工业机器人可将视觉网口设置为 DHCP 模式，机器人通过此网口接入到 DHCP 功能的路由中，用于自动分配 IP 地址给机器人，这时可以通过机器人探测功能探测到机器人并连接。 需要使用静态 IP 地址时，协作机器人可将 J1 网口设置为所需网段的 IP 地址，工业机器人可将视觉网口设置为所需网段 IP 地址，机器人通过此网口接入到路由中，这时可以通过机器人的 IP 地址访问和控制机器人。

		关于 IP 修改： 以 Windows10 操作系统为例，将网线一端连接至机器人的 J2 接口，另一端连接至终端设备（PC）；在终端设备（PC）上单击"开始 > 控制面板"菜单选择"网络和共享中心"；弹出"网络和共享中心"窗口；在"网络和共享中心"窗口单击"本地连接"；弹出"本地连接状态"界面；在"本地连接状态"界面单击"属性"'弹出"本地连接属性"界面；在"本地连接属性"界面双击"Internet 协议版本 4（TCP/IPv4）"；弹出"Internet 协议版本 4（TCP/IPv4）属性"界面；在"Internet 协议版本 4（TCP/IPv4）属性"界面选择"使用下面的 IP 地址"，修改终端设备（PC）的 IP 地址、子网掩码以及默认网关并确认。（可将终端设备（PC）的 IP 地址修改为与机器人同一网段未被占用的任意 IP 地址，其子网掩码和默认网关与机器人的一致） 

警告

手动修改机器人的网口 IP 地址时，切勿将不同的网口设置为同一网段的静态 IP 地址；不要轻易修改调试网口的网络模式和 IP 地址（192.168.0.160）；不要轻易修改示教器 xPad 适配网卡的网络模式和 IP 地址（192.168.1.160）。

4.3 连接机器人

连接机器人

进入选项->连接界面，依次输入机器的 IP 地址

> **提示**
>
> IP 和端口是识别目标控制器的途径，当连不上机器人时，应首先排查网络问题，检测网络是否互通。

机器人探测

说明

HMI 可探测同一网段内所有可以连接的机器人并显示。

点击底部状态栏网络图标按钮  ，可以进入到搜索机器人界面，点击搜索机器人可以探测到可用的机器人。

> **提示**
>
> 1、 搜索机器人时请确保运行 RobotAssist 的设备和机器人处于同一网络中，网络互通。
> 2、 运行 RobotAssist 的设备和机器人网络互通时，使用搜索可用机器人功能还是无法搜索到机器人时，可能是因为设备防火墙屏蔽了 RobotAssist 软件发送的连接请求。

4.4 用户登录

说明

成功连接机器人后，默认用户为操作员，点击底部状态栏 超级管理员 可进行用户切换，默认密码 123456。

用户登录及权限详情可见 5.1.1.1 用户组及权限章节。

4.5 切除与恢复连接

说明

- 在连接界面点击断开按钮，可切除 RobotAssist 软件和控制器的连接。
- 不支持多 RobotAssist 软件同时连接，当前 RobotAssist 软件断开连接确认后或者机器人重启，其他 RobotAssist 软件可连接。
- 恢复 RobotAssist 软件连接与初始连接过程一致，通过用户登陆建立连接。

4.5.1 自动重连

说明

在连接界面，通过打开按钮可以启用自动重连功能。重连方式有两种：

- 复选框勾选，使用重连时长和次数，可以指定重连间隔和次数，重连总时长=单次时长*重连次数。
- 复选框未勾选，不使用重连时长和次数，RobotAssist 软件一直重连控制器。

4.5.2 热插拔示教器 xPad2

说明

在机器人开机使用过程中，如果需要断开示教器 xPad2 与机器人的物理连接，需要按照如下步骤进行操作，否则直接断开物理连接将会导致机器人进入急停状态。

- 进入基本设置界面，在“示教器模式设置”中，可以看的当前示教器模式的状态显示，默认为“有示教器模式”，单击“切换模式”按钮将尝试将模式切换为“无示教器模式”，切换成功后，界面显示模式为“无示教器模式”。
- 当示教器模式切换为“无示教器模式”后，可以断开示教器 xPad2 与机器人的物理连接，此时机器人将不会进入急停状态。

如果需要重新连接示教器 xPad2 至机器人，按照以下步骤操作即可：

- 将示教器 xPad2 与机器人建立好物理连接；
- 进入基本设置界面，在“示教器模式设置”中，可以看的当前示教器模式的状态显示，应该为“无示教器模式”，单击“切换模式”按钮将尝试将模式切换为“有示教器模式”，切换成功后，界面显示模式为“有示教器模式”。

> 提示
>
> 热插拔示教器的功能仅支持部分机型，对于不支持此功能的机型将提示示教器模式切换失败，具体机型配置建议咨询技术支持。

5 工作模式与安全

5.1 安全管理

5.1.1 关于本章

本章将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，请联系您的系统集成商以获得此类信息。

5.1.2 安全术语

5.1.2.1 安全标识

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。

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。
	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。
	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。
	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。
	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期间损坏。
	提示	用于提示一些重要信息或者前提条件。

风险说明

图标	名称	说明
	挤压	操作人员、维护人员在调试、维修、检修、工具装夹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、工具装夹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、工具装夹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、工具装夹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，可能会导致烫伤危害。

5.1.2.2 安全特性

安全级别

xCore 系统配备了专门的安全模块用来处理安全相关信号，并提供了安全门、安全光栅等外部安全信号接口。

由安全模块处理的信号包括：

- 紧急停止信号；
- 安全门信号；
- 使能开关信号；

5.1.2.3 停止处理

停止处理类型

珞石机器人支持两种停止处理类型：紧急停止和受控停止。

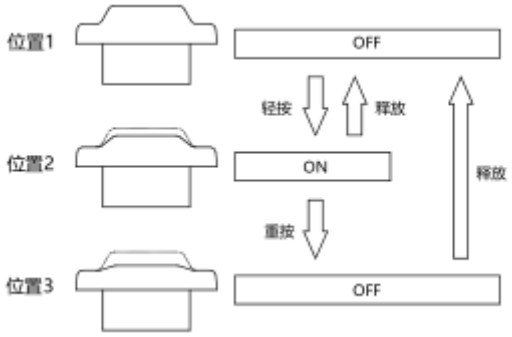
紧急停止	紧急停止是机器人系统中优先级最高的功能。按下紧急停止按钮将触发急停，此时所有其他的机器人控制功能将停止，机器人停止运动且各关节电机的动力电将被切断，控制系统切换紧急停止状态，在被手动复位之前该状态将一直保持。 触发急停后，根据不同的工作状态，系统可能会采取两种不同的停止方式中的任意一种： ➢ STOP 0 停止，当机器人的电源被切断后，机器人立刻停止工作。这是不可控的停止，由于每个关节会以最快的速度制动，因此机器人可能偏离程序设定的路径。当超过安全评定极限，或当控制系统的安全评定部分出现错误的情况下方可使用这种保护性停止； ➢ STOP 1 停止，当为机器人供电使其停止时，机器人就停止，当机器人实现停止后切断电源。这是可控性停止，机器人会遵循程序编制的路径。机器人运动停止电源切断； 提示 1、紧急停止仅用于在危险情况下立刻停止机器人运行。 2、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。
受性停止	受控停止是机器人在不断电的情况下，停止程序运行。 ➢ STOP 2 停止，机器人通电时的可控性停止。安全评定控制系统的操控可使机器人停留在停止的位置。

5.1.2.4 使能开关

使能装置

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。

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，使得机器人处于允许运动的状态。松手放开或者用力按压到底都会将电源切断。



提示

手动使能的黄色按钮为使能开关，当按压到中间位置时电机动力电源接通并自动使能，系统处于 Motor On 状态，可以进行 Jog 或者运行程序。松开或者按到底时电机动力电切断，系统回到 Motor Off 状态。

为了安全的使用示教器，必须遵守以下要求：

1. 在任何情况下都必须保证使能开关可以正常工作；
2. 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；
3. 任何进入机器人工作空间的人员必须随身携带手持使能，以避免其他人在内部人员不知情的情况下启动机器人 。

警告

严禁使用外部装置将使能开关卡住使其停留在中间位置！

5.1.3 工作中的安全事项

5.1.3.1 概述

关于机器人

xCore 在人机协作方面提供了协作模式、碰撞检测等安全功能（具体参见错误!未找到引用源。）来保障用户与机器人配合工作时的人身安全。请确保您已经熟悉阅读 7.2 节的安全功能介绍，再操作机器人。

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。

5.1.3.2 关注自身安全

基本原则

必须遵守以下几条简单的原则以便安全的操作机器人：

➢ 留意安装在机器人上的会活动的工具，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的问题，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。

5.1.3.3 从急停状态恢复

说明

系统处于急停状态时必须执行复位操作以便恢复到正常状态。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。

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。

5.1.3.4 手动模式的安全事项

关于手动模式

在手动模式机器人的运动处于手动控制下。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。

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 250 mm/s 以下，即无论是 Jog 机器人还是运行程序，机器人的最大运动速度不会超过 250 mm/s，不论程序中设置的速度是多少。

旁路外露安全信号

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不会处于急停状态，以方便进行调试。

5.1.3.5 自动模式的安全事项

关于自动模式

自动模式用于在正式生产过程中运行机器人程序。

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。

启用外部安全信号

外部安全信号如安全门、安全光栅等在自动模式下会启用，安全门打开会触发紧急停止。

安全处理生产中的故障

绝大多数情况下，机器人都属于一条生产线的一部分，因此机器人出现故障往往不只影响机器人工作站本身，同样的当生产线其他部分出现问题时也可能会影响到机器人工作站。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，以提高安全性。

例如在某条生产线上，机器人需要从传送带上抓取工件。如果机器人出现故障，为了保证生产过程不中断，在检修机器人的同时传送带保持运行，此时机器人维修人员应该额外考虑在运行中的传送带旁边工作的安全措施。

再比如一个焊接机器人需要进行例行维护而将该机器人从生产线上脱离出来时，也必须停止为该机器人上料的机器人，以免造成人员伤害。

5.1.3.6 紧急情况的处理

5.1.3.6.1 火灾处理

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。

重度火灾的处理措施

当火灾已蔓延开来、处于不可控阶段时，现场工作人员不要再试图灭火，而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，而应让他/她迅速平躺在地上，用衣服或其它合适物品、方式将火扑灭。

5.1.3.6.2 触电处理

切断电源

当发现有人触电，不要惊慌，首先要尽快切断电源。

应根据现场具体条件，果断采取适当的方法和措施，一般有以下几种方法和措施：

1） 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。

2） 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。

3） 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。

触电伤员脱离电源后的处理

1） 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。

2） 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，并用 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。

3） 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。若即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。

5.2 机器人工作模式

5.2.1 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。

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 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；

5.2.2 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。xCore 系统支持的系统 IO 列表参见。

通常在自动模式下执行的任务

自动模式下通常被用来执行以下任务：

- 加载，启动及停止 RL 程序；
- 急停后返回原编程路径；
- 备份系统；
- 清洗工具（根据工艺要求）；
- 对工件进行加工、处理；

自动模式下使用限制

- 自动模式下，将无法进行 Jog。
- 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。
- 自动模式下不允许备份恢复。
- 自动模式下不允许功能授权。
- 自动模式下不允许设置软限位。
- 自动模式下不允许新建，修改，删除 IO。
- 自动模式下不允许参数辨识。
- 自动模式下不允许开启/关闭碰撞检测。
- 自动模式下不允许开启/关闭协作模式。
- 自动模式下不允许开启/关闭拖动示教。
- 自动模式下不允许标定。
- 自动模式下不允许新建变量。
- 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。

5.2.3 模式切换

5.2.3.1 关于模式切换

当前模式

通过观察 RobotAssist 软件界面上的底部状态栏工作模式图标，即可知道当前控制系统所处的工作模式。

 表示当前控制器处于手动模式下，  表示当前控制器处于自动模式下。用户可以点击 HMI 界面上的工作模式图标切换工作模式。

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)。

5.2.3.2 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，请确认碰撞检测功能已开启，防止机器人与工作人员发生意外碰撞出现人身伤害！

注意事项

> 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。

5.2.3.3 自动切换到手动

从自动模式切换到手动模式

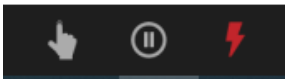
点击 RobotAssist 界面上的  图标将自动模式切换到手动模式，观察图标是否切换成  。如果是，那么模式切换已经完成。如果切换失败，根据顶部状态栏的实时日志信息排除故障。

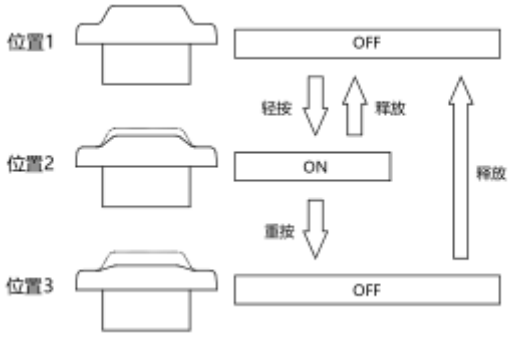
5.3 机器人上下电

5.3.1 机器人上电

手动模式上电

手动模式下，用户通过按住手动使能手柄上的黄色三位使能开关保持在中间位置给电机上电。可以通过听机器人上电声音或者观察 RobotAssist 软件界面上的底部状态栏上电按钮变成红表示上电成功。

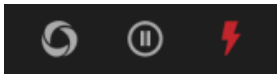




> 提示
>
> 若上电失败，观察实时日志判断机器人此时状态，调整至支持上电的状态再上电。

自动模式上电

自动模式下，点击 RobotAssist 软件底部状态栏的上电按钮给电机上电，判断电机是否正常上电方法与手动模式相同。



5.3.2 机器人下电

手动模式下电

手动模式下，用户通过松开或者用力按压黄色三位使能开关使其保持在位置 1 或位置 3 给电机下电。

自动模式下电

自动模式下，点击 RobotAssist 软件底部状态栏的上下电按钮给电机下电。

> 警告
>
> 紧急情况下，按下手动使能上的急停按钮给机器人紧急断电，需要再次上电时，请先手动复位急停开关。

6 移动控制

6.1 Jog 模式

Jog 设置

Jog 运动可选择轴空间 Jog 或者笛卡尔空间 Jog。

笛卡尔空间 Jog 可以选择 Jog 运动的坐标系，包括五种：世界坐标系、基坐标系、法兰坐标

系、工具坐标系、工件坐标系。

Jog 模式可选连续运动或者增量步进：

- 连续运动模式下，通过使能将机器人上电，按下 Jog 按钮，机器人在设定 Jog 速度下连续运动直至松开使能或者 Jog 按钮；
- 增量步进模式下，长按一次 Jog 按钮，机器人在上电状态，将以固定步长运动；可通过设置步长大小，来精确调整机器人位姿；
- Jog 速度设置，用来控制 Jog 时机器人的运动速度，可选范围 1%~100%，100%时对应机器人最大 TCP 速度 250 mm/s。（笛卡尔空间 Jog 和轴空间 Jog 都以 TCP 线速度 250mm 为最大 Jog 速度）

> 提示
> 增量步进模式下，需长按 Jog 按钮，等待机器人运行指定步长后方可松开 Jog 按钮，点按操作将可能使机器人提前停下来。

快速调整

对于常用的机器人位姿，HMI 运动界面提供快捷调整功能，目标位姿包括：机械零位、拖动位姿、发货位姿及 Home 点位姿等。

快速位姿调整在手动模式下使用，使用方式与 Jog 操作类似，手动模式下通过使能将机器人上电，按下相应目标位姿按钮，机器人将通过轴空间运动至目标位姿。

运动过程的速度可通过 Jog 速度调整。

快速调整提供了参数配置功能，用户可使用自定义的发货位姿、拖动位姿或 Home 点位姿，可以在机器人配置->设置->快速调整页面进行参数配置。未启用自定义设置，则使用默认配置。

6.2 拖动模式

说明

xMate 协作机器人提供拖动模式，用户可以快速实现轨迹录制、轨迹复现功能，在编写程序时，可以轻易拖动机器人进行定位，大大缩短编程时间。

拖动设置

拖动方式可设置为轴空间拖动和笛卡尔空间拖动。

轴空间拖动，各轴运动独立，直接调整各轴位置到期望位姿。

笛卡尔空间拖动，可选仅旋转和仅平移，仅旋转模式，可直接手动引导机器人绕 TCP 做姿态旋转，仅平移模式，可引导机器人沿笛卡尔空间各方向做平移运动。

手动模式下电状态下，在操作面板打开拖动使能开关，机器人自动上电并启用拖动模式，用手同时按下末端拖动把手使能按钮，即可拖动机器人进行点位示教、轨迹录制。

提示

1、 请在启用拖动模式前，设置拖动方式、拖动空间。
2、 启用拖动模式后，机器人自动上电，此时无法设置拖动方式、拖动空间，请关闭拖动后进行设置。

警告

1、 在启用拖动模式前，必须准确设置机器人的动力学参数以及负载参数，否则拖动模式可能打开失败，拖动过程可能存在飘动感。
2、 可使用系统提供的动力学参数辨识功能以及负载辨识功能来设置参数。

危险

在使用拖动示教之前必须要确保以下参数已经正确设置：
1. 机器人的型号；
2. 机器人安装方式，地面安装或者吊装；
3. 本体和负载的动力学参数；
4. 机械零点标定；

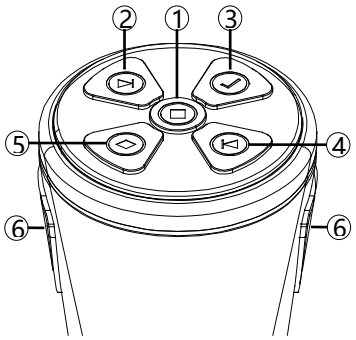
否则各轴角度处于错误状态时，控制器无法计算正确的输出力矩，机器人拖动功能不能正常使用。

适用机型：

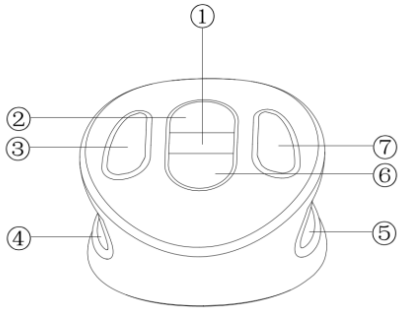
拖动功能及其扩展功能（末端把手、点位示教、连续轨迹示教、轨迹复现）仅适用于 xMate 系列协作机器人。

6.2.1 末端把手

说明

系列	简介	定义
xMate ER	xMate ER 系列机器人末端集成智能交互面板 Pilot 把手，在拖动模式下，通过手柄上按键可快速进行点位示教、连续轨迹示教，实现更好的人机交互。	末端 Pilot 把手按键定义： 

标号	定义
1	以当前位姿更新示教点，开始/停止轨迹录制
2	下一个

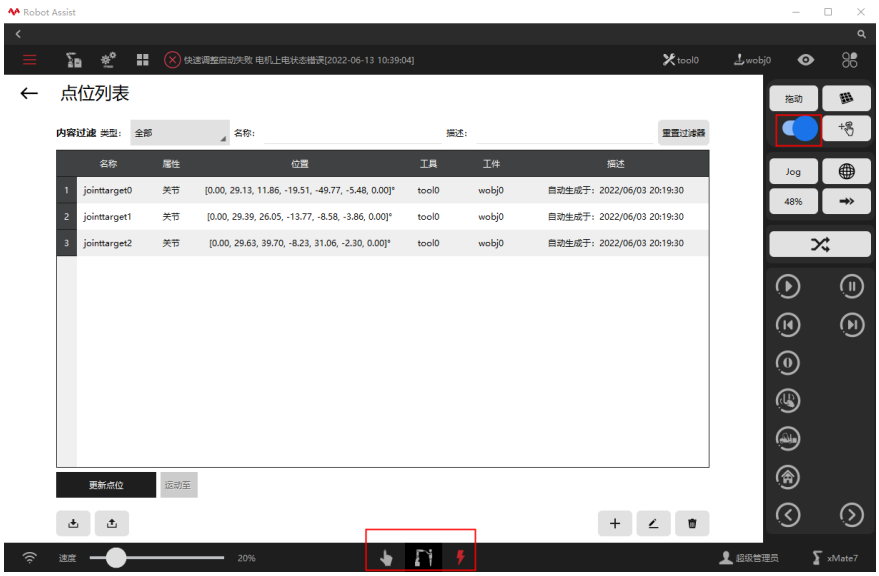
<table>
<tr><td></td><td></td><td><table>
<tr><td>3</td><td>增加列表中点位/轨迹，弹窗提示确认</td></tr>
<tr><td>4</td><td>上一个</td></tr>
<tr><td>5</td><td>删除列表中点位/轨迹，弹窗提示取消</td></tr>
<tr><td>6</td><td>拖动模式下，两个使能按键同时按下开启拖动功能</td></tr>
</table></td></tr>
<tr><td>xMate CR</td><td>xMate CR 系列机器人末端集成智能交互面板 xPanel 把手，在拖动模式下，通过手柄上按键可快速进行点位示教、连续轨迹示教，实现更好的人机交互。</td><td>末端 xPanel 把手按键定义：
 

<table>
<tr><th>标号</th><th>定义</th></tr>
<tr><td>1</td><td>以当前位姿更新示教点，开始/停止轨迹录制</td></tr>
<tr><td>2</td><td>列表中光标向前方向</td></tr>
<tr><td>3</td><td>删除列表中点位/轨迹，取消弹窗提示</td></tr>
<tr><td>4</td><td rowspan="2">拖动模式下，两个使能按键同时按下开启拖动功能</td></tr>
<tr><td>5</td></tr>
<tr><td>6</td><td>列表中光标向后方向</td></tr>
<tr><td>7</td><td>增加列表中点位/轨迹，确认弹窗提示</td></tr>
</table></td></tr>
</table>

6.2.2 点位示教

说明

在操作面板打开拖动使能开关，机器人自动上电并启用拖动模式，通过 RobotAssist 软件和末端拖动把手进行以下操作：



	操作	说明

1	创建/加载一个工程，进入点位列表界面	末端按键只在 RobotAssist 软件当前打开页面为点位列表、路径列表时响应
2	同时按下末端把手两个使能按键，拖动机器人至任意一个位置后松开拖动使能按键，按下末端把手按键"点位增加"按钮	点位列表末尾新增一个当前位姿信息的示教点，光标定位到新增示教点
3	按下末端把手按键"上一个/下一个"按钮	点位列表中光标向上一个/下一个点位移动并选中
4	在点位列表中选择一个点位更新位置，拖动机器人至另一个位置后松开拖动使能按键，按下末端把手按键"点位更新"按钮	点位列表中当前选中点位以当前位姿更新
5	在点位列表中选择一个点位进行删除，按下末端把手按键"点位删除"按钮并确认	删除点位时有弹窗提示，按下末端按键"确认"按钮，当前选择点位从点位列表中删除；按下末端按键"取消"按钮，弹窗提示取消，当前选择点位保留

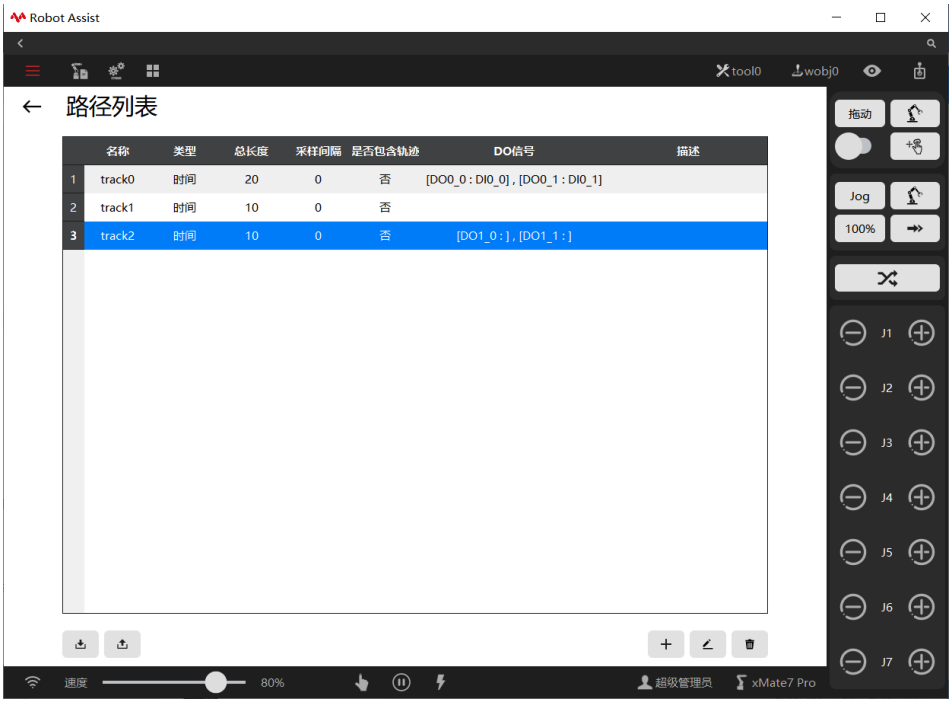
6.2.3 连续轨迹示教

HMI 流程式操作

在操作面板打开拖动使能开关，机器人自动上电并启用拖动模式，通过 RobotAssist 软件进行以下操作：

步骤 **1** 创建/加载一个工程，调整机器人至任意初始位置，进入"工程 > 路径"界面。

步骤 **2** 在路径列表界面，点击 + 新建一个路径，进入新路径设置子界面



步骤 **3** 设置新路径名称、描述、录制总时长，如需录制 DO 信号，点击"选择 DO"按钮，在弹出的对话框中选择 DO 信号，每个 DO 信号还可以设置一个映射 DI 信号，当一个 DO 信号设置 DI 映射信号时，录制路径时会将 DI 信号的变化当作 DO 信号的变化进行记录，并将 DI 信号输出到 DO 信号；若不关联 DI 信号，则会直接记录 DO 信号的变化，此时可以在状态监控-

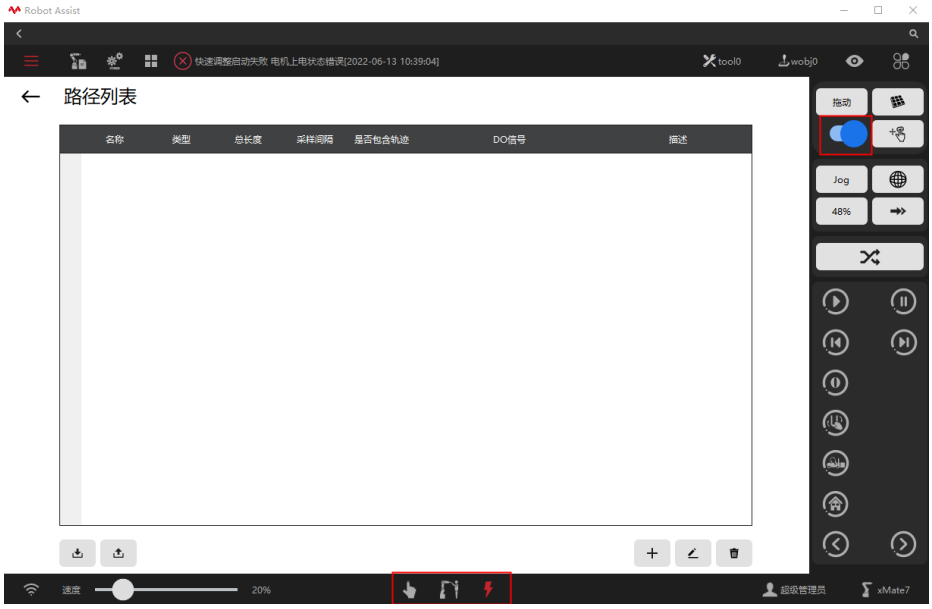
IO 信号界面手动设置 DO 信号输出。

步骤 **4** 设置好参数后，点击录制按钮，开始轨迹录制。开始录制后，在倒计时时间内拖动机器人并设置相关的 DIO 信号完成轨迹录制，录制过程中点击"停止"按钮，停止后记录停止时刻前轨迹。

> **提示**
>
> 轨迹录制后暂存在缓存中，可以选择丢弃或者保存至文件。页面保存处显示当前"有/无录制轨迹"。

末端按键操作

在操作面板打开拖动使能开关，机器人自动上电并启用拖动模式，通过 RobotAssist 软件和末端拖动把手进行以下操作：



	操作	说明
1	创建/加载一个工程，调整机器人至任意初始位置，进入路径列表界面	末端按键只在 RobotAssist 软件当前打开页面为点位列表、路径列表时响应
2	按下末端把手按键"路径增加"按钮	路径列表末尾新增一个路径，光标定位到新增路径
3	按下末端把手按键"上一个/下一个"按钮	路径列表中光标向上一个/下一个路径移动并选中
4	在点位列表中选择一个路径开始录制，按下末端把手按键"开始轨迹录制"按钮，同时按下末端把手两个使能按键拖动机器人进行轨迹录制	按下"开始轨迹录制"按钮后，轨迹录制开始；直到按下"停止轨迹录制"按钮，录制结束并自动保存
5	在点位列表中选择一个路径进行删除，按下末端把手按键"点位删除"按钮并确认	删除点位时有弹窗提示，按下末端按键"确认"按钮，当前选择点位从点位列表中删除；按下末端按键"取消"按钮，弹窗提示取消，当前选择点位保留

6.2.4 轨迹复现

说明

连续轨迹示教成功后完成轨迹录制，可在录制界面回放轨迹进行确认，确认后手动保存录制轨迹。

回放设置

回放模式可勾选循环模式，回放时循环回放。

回放速率可设置为 1%至 300%。

> 提示
>
> 建议用户回放速率设置范围 1%至 300%，当回放速率大于 100%，有可能产生驱动器跟随错误。

回放操作

步骤 **1** 关闭拖动使能开关，切换至自动模式，点击上电按钮，机器人上电。

步骤 **2** 点击回放，机器人回放录制轨迹。

步骤 **3** 轨迹回放完成后，完成轨迹确认，点击保存，录制轨迹成功保存到文件。

步骤 **4** 完成轨迹录制、轨迹回放确认、轨迹保存，点击下一步返回路径列表，路径列表显示已保存路径

7 机器人配置

7.1 基本设置

7.1.1 用户组及权限

用户级别

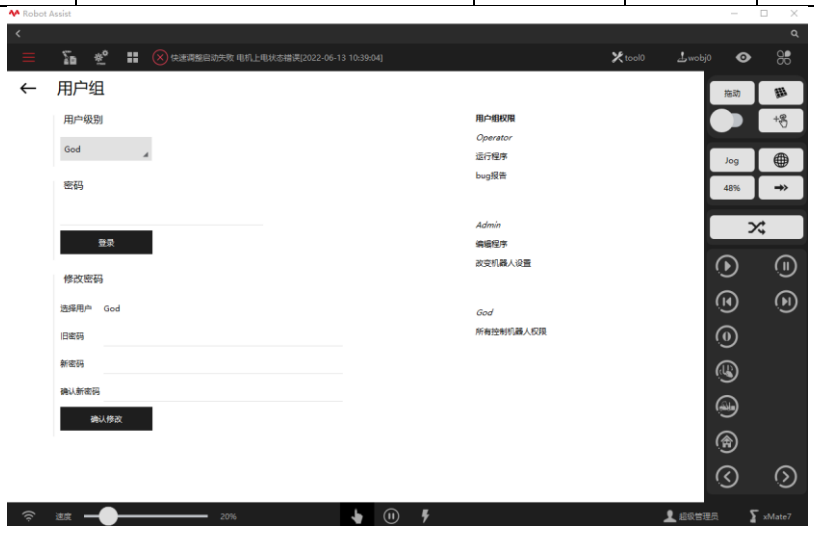
xCore 系统内置了三个级别的用户，根据操作权限从低到高分别是 Operator（操作员），Admin（管理员）和 God（超级管理员）。

从低权限用户切换到高权限用户需要输入密码，默认密码 123456，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

类别	功能	操作员	管理员	超级管理员
工程相关	工程管理 （新建、导入、导出）	N	Y	Y
	查看工程 （包括程序和 IO、变量等对象数据）	Y	Y	Y
	编辑工程 （包括程序编辑和工具等对象设置）	N	Y	Y
机器人运动与程序运行	模式切换	N	Y	Y
	上/下电	N	Y	Y
	启动/停止程序	Y	Y	Y
	调整程序运行速度	N	Y	Y
	单步调试程序	N	Y	Y
	APP 运行	Y	Y	Y
	Jog/拖动界面	N	Y	Y

运行界面	设置自动加载工程	N	Y	Y
	查看运行数据	Y	Y	Y
系统设置	系统升级	N	Y	Y
	数据备份/恢复	N	Y	Y
	用户权限管理	N	Y	Y
	功能授权	N	Y	Y
	系统时间设置	N	Y	Y
	系统语言设置	N	Y	Y
	控制器重启	N	Y	Y
机器人设置	本体参数设置	N	N	Y
	机器人安装	N	Y	Y
	零点标定	N	Y	Y
	运动参数辨识	N	N	Y
	扩展 IO 模块配置	N	Y	Y
	系统 IO 设置	N	Y	Y
	末端工具通信设置	N	Y	Y
	外部通信设置	N	Y	Y
	安全设置	N	Y	Y
	清除伺服告警	N	Y	Y
	RCI 功能设置	N	Y	Y
日志管理	查看日志	Y	Y	Y
	删除日志	N	Y	Y
	查看/删除 debug 级别日志	N	N	Y
	日志备份	N	Y	Y
帮助界面	查看帮助	Y	Y	Y



7.1.2 控制器设置

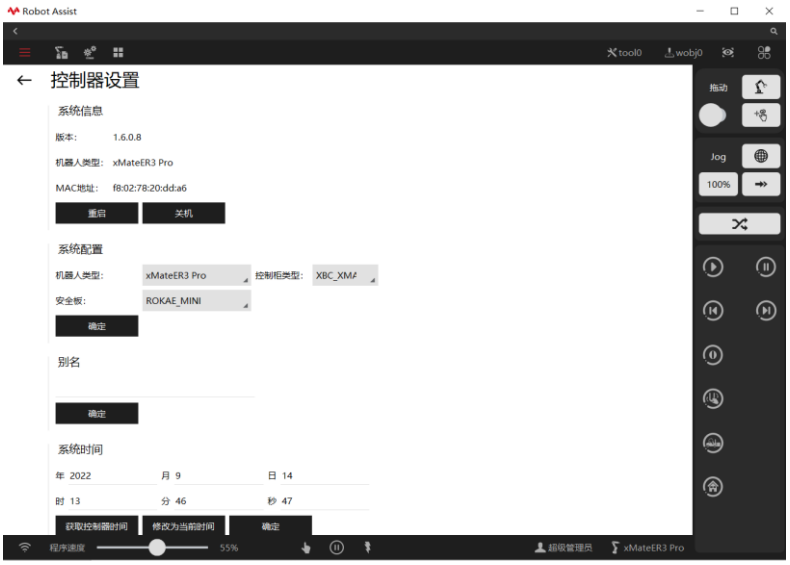
系统信息

xCore 系统提供了控制器设置选项，可对控制器进行软重启和关机功能，您需在重启前保存所

有配置信息。如果进行了控制器关机，则需要控制柜断电后重新上电才能重新启动控制器软件。

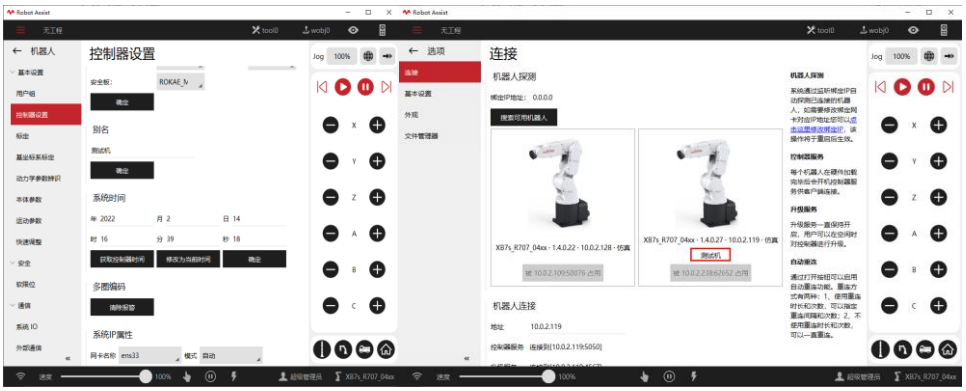
系统配置

应当正确配置当前机器人的机器人类型、控制柜类型、安全板类型，方可保证机器人运作正常。



别名

对控制器设置别名，方便在同一片局域网内存在多台机器人设备时进行区分。别名会在机器人探测时显示在界面上，如下图所示。



系统时间

系统时间显示的是控制器的系统时间。

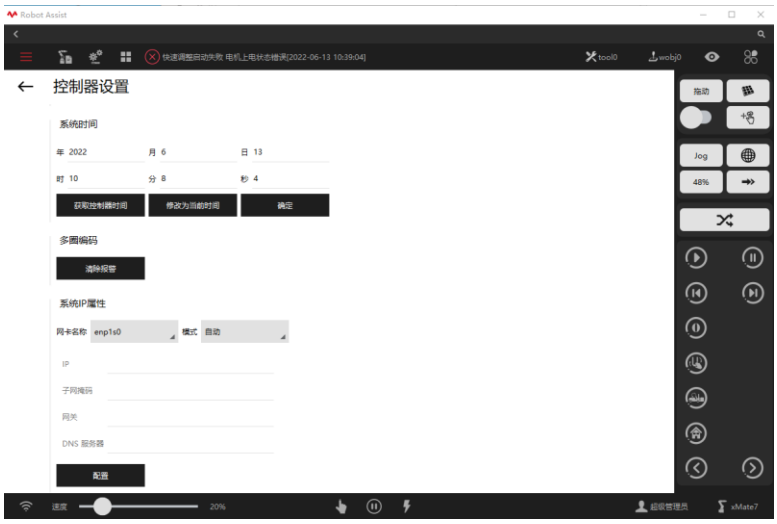
系统时间为日志等功能提供绝对时间基准，用以追溯相关事件的发生时刻。

用户可以通过 获取控制器时间 来观察当前机器人的时间基准，以检查是否和本机时间对齐。

支持手动修改和修改为 RobotAssist 软件所在设备当前的系统时间两种修改方式，用户可以直

接手动修改控制器系统时间；或点击 修改为当前时间 控制器系统时间更新为 RobotAssist 软件所在设备当前的系统时间。

当 RobotAssist 软件上显示的系统时间与 RobotAssist 软件所在设备右下角的系统时间不一致时，用户可点击 修改为当前时间 将控制器系统时间更新为 RobotAssist 软件所在设备的系统时间。



警告

1、系统时间是日志信息的绝对时间标准，请勿随意修改，错误的修改系统时间会导致用户无法根据日志追溯相关事件发生的时刻。

2、该操作不宜频繁执行，“获取控制器时间”、“修改为当前时间”这两个操作或者同操作本身的执行间隔需要大于 5 秒。

多圈编码

清除编码器多圈报错。

提示

1、 该功能仅对工业机器人有用。

2、 机器人本体更换编码器电池之后，需要使用该功能清除报错，再重新进行标定。

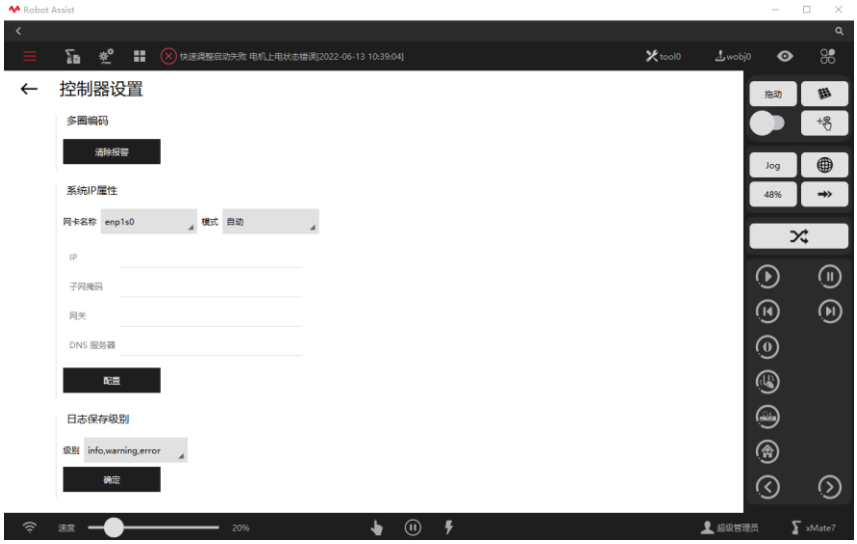
系统 IP 属性

对机器人外网口连接模式进行配置，具体使用说明参见第 4 章，连接机器人章节。

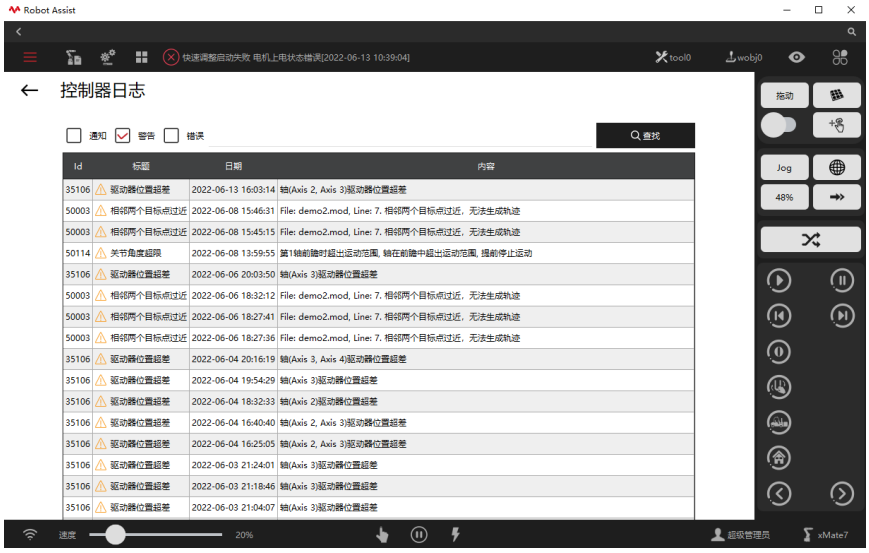
日志保存级别

设置日志保存级别。日志级别从低到高分为“信息”、“警告”、“错误”三个级别，对应显示为“info”、“warning”和“error”。可以设置从某个级别开始进行日志保留，这样低于设置级别的日

志将仅进行在线展示，不会进行历史留存。



例如，选择” warning,error”，则 info 级别的日志只会在出现的时候显示一次，在“诊断”中或者控制器重启之后，无法查询到。而 warning 和 error 的日志除了在线显示，也可以在“诊断”中或者控制器重启之后查询到。历史记录查询位置在“诊断”->“控制器日志”中。



7.1.3 零点标定

说明

xCore 系统提供了机器人标定功能，包括机械零点标定和力传感器零点标定。标定功能可以执行一键标定，也可以进行单轴标定；

机械零点标定

机械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。

> 警告
>
> 1、 机械零点是机器人控制算法中的理论零点，请勿随意标定，标定前请使用机械标定块确认机器人各关节都处于零点。
> 2、 机器人出厂经过激光跟踪仪标定之后，机器人不可进行机械零点标定，否则，激光跟踪仪标定后的零点数据会丢失，影响机器人精度。当机器人零点丢失时，请联系机器人厂商进行零点数据还原。

力矩零点标定

力传感器零点标定的目的是为了让控制算法中的理论关节扭矩零点与实际关节扭矩零点重合，使得机械连杆系统可以正确的获取关节的实际扭矩。

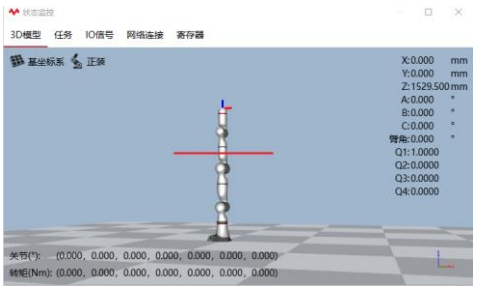
更通俗的讲，力传感器零点标定就是将机器人的各关节运动到不受重力影响的位置，并通知控制系统记录此时各关节力传感器数值的过程。

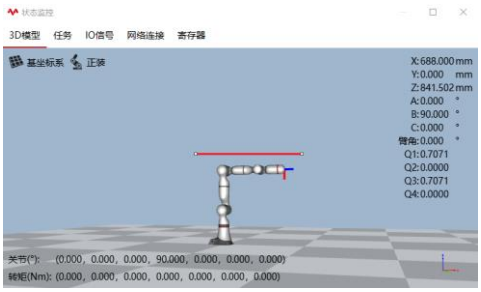
> 警告
>
> 进行力矩零点标定之前，请确保机器人处于机械零点的位置！

角度标定设定

在某些空间受限制的场景，机器人无法回到机械零点。此时如果需要标定，则需要使用“角度标定设定”功能。该功能可以在已知当前角度的情况下，输入已知的角度，达到与“机械零点标定”相同的效果。

下面以 xMate7 Pro 七轴机器人为例，假设在 4 轴空间之上有障碍物，机器人无法达到机械零点竖直状态，而又需要进行零点标定，可以先通过各个轴单独 jog 达到 4 轴 90 度，机器人成为直角状态。然后在“角度设定”中，输入当前对应的角度信息来进行“机械零点标定”。





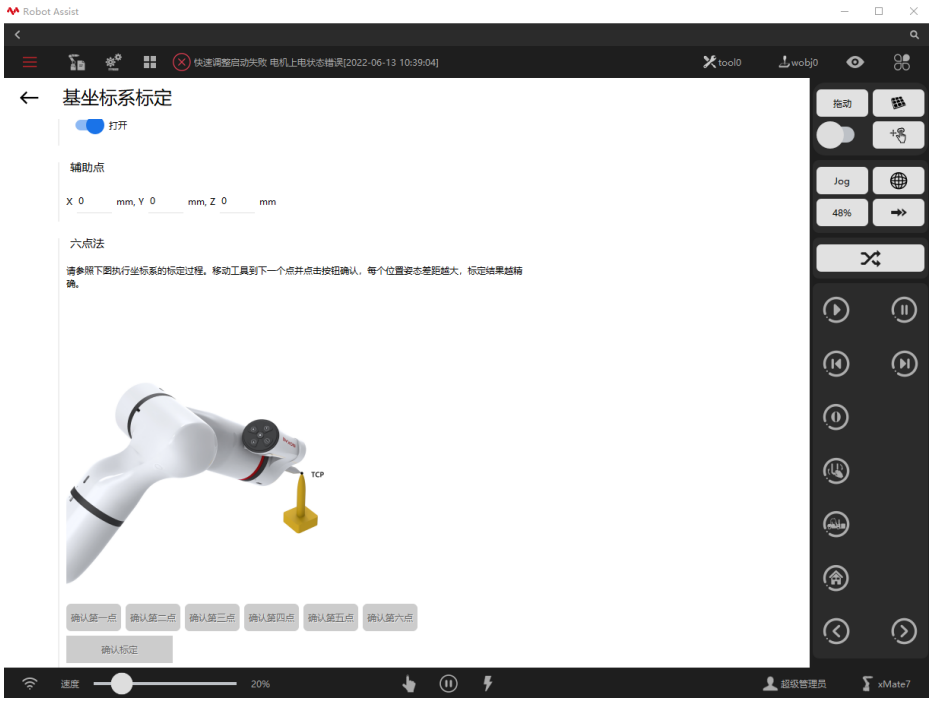
注意，在上面这个例子中，虽然是不同姿态标定，但机器人的零点都是竖直状态，因此在输入 4 轴 90 度标定成功后，如果直接使用 “快速调整至零点”功能，则机器人还是会往机械零点竖直状态运动，与障碍物碰撞！所以一定要注意，该功能标定的是零点，而不是指零点为当前角度。

7.1.4 基坐标系标定

什么是基坐标系?

基坐标系位于机器人底座的中心，相对世界坐标系描述，用于确定机器人的摆放位置。在机器人任意角度安装、多机器人场景下，通常需进行基坐标系标定。

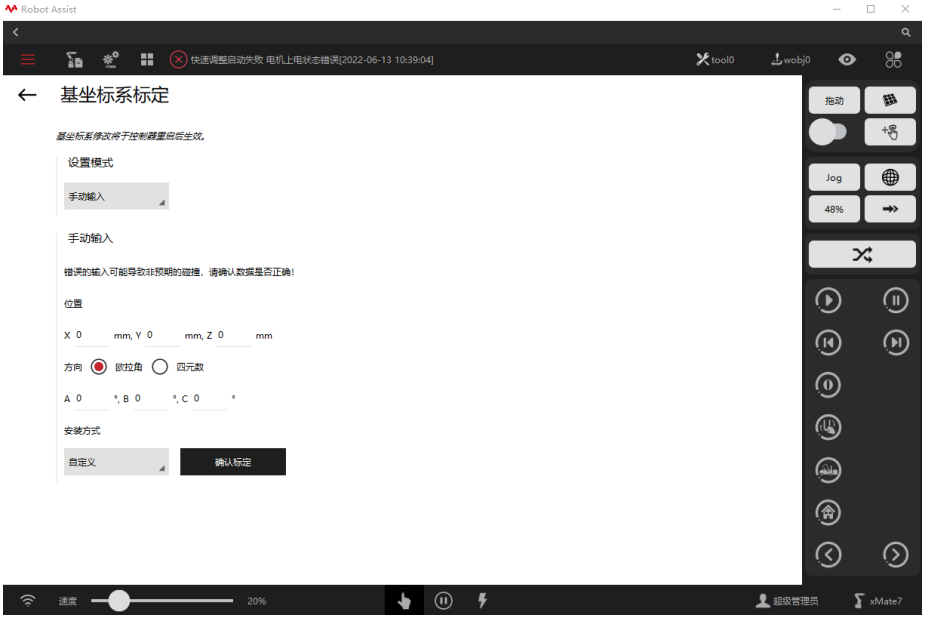
基坐标系标定



如上图所示，标定基坐标系的一般步骤如下：

	操作	说明
1	使用 admin 级别的用户登录系统，并完成工具坐标系标定。	当前选择的工具与法兰端安装的工具保持一致。
2	确认基坐标系标定方法。	系统支持六点法和手动输入，默认使用六点法标定，若已知基坐标系相对世界坐标系的偏移，可直接手动输入。
3	定义辅助点位置。	当机器人离世界坐标系距离较远，工具末端够不着时，可通过辅助点来标定基坐标系。 辅助点位置相对于世界坐标系定义。
4	Jog 机器人，依次确认各示教点。	根据标定结果，用户可选择是否保存基坐标系数据。

手动输入



可通过手动输入的方式来设置基坐标系，手动输入基坐标相对于世界坐标系的位置和姿态，姿态可以通过欧拉角或四元数来表示。

安装方式	说明	A	B	C
正装		0	0	0
墙装 A	电源出线口在基座上方	0	90	0
墙装 B	电源出线口在基座右边	-90	0	-90
墙装 C	电源出线口在基座下方	180	-90	0
墙装 D	电源出线口在基座上方	90	0	90
倒装				

警告

手动输入参数标定基坐标系时，请确保数据准确，错误的输入可能导致非预期的碰撞。

7.1.5 动力学参数辨识

说明

动力学参数辨识功能用来辨识机器人的动力学模型参数。动力学模型参数主要用于机器人力控、拖动示教、虚拟墙和碰撞检测等功能。请确保机器人动力学模型参数正确辨识，否则会出现上述功能无法正常使用，或者机器人异常抖动的现象。

使用流程

步骤一 机器人配置->设置->动力学参数辨识

步骤二 在左侧列表中选中动力学参数辨识。

步骤三 请将机器人周围的障碍物移除，保证机器人可达空间内无任何障碍物（底座除外）。机器人可达空间请参考各机器人型号所对应的机器人安装手册。

步骤四 “预热时间”是机器辨识持续运行时间，运行时间越长，辨识效果越好。预热时间可选择范围为 0/1/2/4 小时。如果选择 0 小时，则辨识只运行一次完整的轨迹就完成，持续时间大约 1 分钟。

步骤五 点击“开始运行”。之后，机器人将自动开始执行动力学参数辨识程序。

步骤六 等待辨识结果。若显示辨识成功，则表明辨识过程正常完成。若显示辨识失败，请参考后续错误处理部分。

步骤七 动力学辨识后，需要进行摩擦力辨识，对机器人的摩擦力系数进行修正。点击“开始运行”。之后，机器人将自动开始执行摩擦力辨识程序。

步骤八 等待辨识结果。若显示辨识成功，则表明辨识过程正常完成。重启后，摩擦力系数生效，并在当前界面看到辨识后的各轴摩擦力系数。若显示辨识失败，请参考后续错误处理部分。

使用限制

1.动力学参数辨识功能在以下机型不支持：XB12s-3、XB12s-4。

2.请在试运行前确保机器人可达空间范围内无障碍物。

3.使用前请确保机器人零点正确标定，关于机器人零点标定请参考章节 7.1.3。

4.请不要带负载进行动力学辨识以及摩擦力辨识

5.辨识结果不会立即生效，需要重启机器人后方才生效

错误处理

1. 辨识过程中可以通过 RobotAssist 软件点击”停止”按钮来停止辨识。

2.紧急情况可以通过急停按钮来停止机器人运行。

3.中断辨识后，辨识结果不会被记录，需要重新辨识。

4.摩擦力辨识结果异常会启用标称值，并在界面上会有提示。如果标称值不合适，可以在界面下方手动修改摩擦力系数。

7.1.6 本体参数

说明

本体参数包含 RD 参数、减速比、耦合系数。这些参数都是机器人本体相关的数据，包括本体机械机构以及本体零部件属性。该页面的参数会直接影响运动的准确性，请谨慎修改。

RD 参数

RD 参数是用来描述机器人连杆坐标系之间相对位姿关系的一组参数，是机器人运动学的基础。

- RD 参数需匹配机器人实际的连杆参数，配置时需注意，否则可能出现超过工作空间等报错，请谨慎修改；
- 机器人出厂前会存在一组默认参数，正确标定机器人连杆参数后可修改 RD 参数来提高机器人绝对精度；用户在修改参数或者导入参数之前应该检查标定后的 RD 参数合理性；
- 修改 RD 参数之后需要重启控制器才能使参数生效。

减速比

减速比是机器人各轴内减速器的参数。请保持出厂设置，不要更改。仅在机器人更换了不同型号的减速器之后，按照厂商提供的减速比进行设置。

耦合系数

机器人 4 轴、5 轴、6 轴三个轴运动时存在耦合关系，耦合系数描述了这些关节运动时其他关

节耦合运动的参数。请保持出厂设置，不要更改。仅在机器人更换了不同型号的减速器或传动部件之后，由厂商提供数据和修改。

使用限制

God 权限可用。

7.1.7 运动参数

说明

运动参数包含机器人各个轴的最大速度、最大加速度和最大加加速度，运动参数影响机器人运动时可能达到的最大速度、最大加速度、最大加加速度，影响机器人运行节拍和平顺性。机器人出厂前存在默认一组参数，修改运动参数可能造成机器人异常抖动、报错，影响机器人使用寿命，请谨慎修改。

- 最大轴速度：指的是机器人运动时，每个轴允许的最大速度，主要由电机速度限制。一般使用出厂参数，不需要修改。
- 最大轴加速度：指的是机器人运动时，每个轴允许的最大加速度，受电机力矩能力限

制。当关闭动力学约束时，该参数生效，机器人运动时每个轴的最大加速度由该参数限制；当开启动力学约束时，该参数失效，机器人运动时每个轴的最大加速度由动力学模块计算得到。

设置机器人最大加速度时，数值不应小于该轴的最大速度的 3-5 倍。

- 最大轴加加速度：指的是机器人运动时，每个轴允许的最大加加速度。加加速度指的是加速度对时间的导数。通常情况下，加加速度越大，机器人运动时越容易抖动，加加速度越小，机器人越不容易抖动。另外，当机器人经过转弯区时，机器人实际的加加速度会增大，此时加加速度对机器人节拍的限制较为明显。当机器人运行的程序中，小转弯区较多时，可以适当地增大加加速度，从而加快节拍，调整时需要额外注意机器人的抖动情况。

  设置机器人最大加加速度时，数值不应小于该轴的最大加速度的 3-5 倍。
- 加速度倍率：该参数用于缩放机器人运行时的加速度，数值越大，机器人运行时的加速度越大，数值越小，机器人运行时的加速度越小。
- 加速度上升时间：机器人加速度从最小值增加到最大值所消耗的时间，数值越小，机器人加速越快，数值越大，机器人加速越平缓。
- 安全控制：机器人从收到停止信号，到机器人完全停止运动，该过程所消耗的时间。数值越小，机器人停止越快，数值越大，机器人停止越慢。
- Search 指令最大停止距离：使用 Search 类指令时，从机器人收到停止运动信号，到机器人完全停止运动，该过程中机器人 tcp 点走过的距离不超过该数值。

使用限制

God 权限可用。

7.1.8 力控参数

说明

力控参数，即根据实际硬件设备和环境进行适配的控制参数。

控制器中已内置两套控制参数，按照基座刚度等级（高、低）进行区分。设置与实际安装环境相符的基座刚度等级，机器人将切换对应的基础控制参数。初始化及未启用状态下，使用高刚度基座控制参数作为默认值。

常见固定底座的使用场景，无需调整此参数；当机器人安装在柔性底座、移动平台上，需要将基座刚度等级调节为低。例如：AGV 小车搭载机器人的使用场景。

力控参数辨识，即借助外加负载进行控制参数精确标定的操作。

注意：该功能仅供有经验者使用。辨识过程应严格按照操作指导分步进行；负载优先选择专用标定块，限于工具条件也可通过自定义方法，输入实际负载信息。

7.1.9 快速调整设置

说明

HMI 界面支持便捷调整至某些常用位姿的功能。常用位姿支持用户进行自定义设置，设定完成后，HMI 界面上可快速调整至新的设定姿态。支持自定义的位姿包括：拖动位姿、发货位姿及 Home 点位姿等。

同时使用该功能还可以在保持机器人 TCP 位置和臂角（仅 7 轴机器人有臂角的概念）不变的情况下，快速调整机器人到一些特殊的姿态，包括：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。

操作方式

快速位姿调整在手动模式下使用，使用方式与 JOG 操作类似，手动模式下通过使能将机器人上电，按下相应目标位姿按钮，机器人将通过轴空间运动至目标位姿。

运动过程的速度可通过 JOG 速度调整。

参数配置

快速调整提供了参数配置功能，对于一些特定用户，想使用其它的发货位姿、拖动位姿或 Home 点位姿，可以在机器人->快速调整页面进行参数配置。

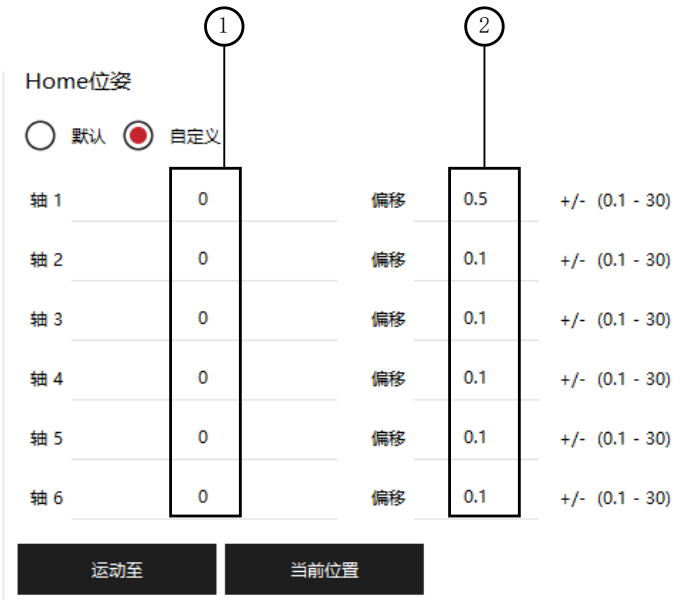
打开启用按钮，点击运动窗口中对应的快速调整按钮，机器人会运动到修改后的位置。如果未启用参数配置，快速调整使用默认的位姿。

Home 位姿

这里详细介绍 Home 位姿的设置及参数含义。Home 位姿支持设置为以关节角度为基准的一个范围，机器人关节在这个范围区间内时，均处在 Home 位姿，因此相应的 Home 状态也会输出信号。

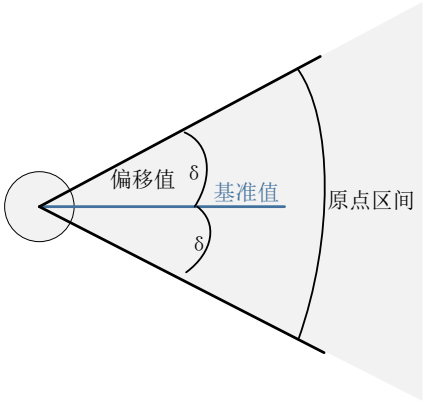
Home 基准点支持通过"当前位置"进行示教更新。

相关参数如下：



序号	名称	含义
1	基准值	各个关节的原点基准值。
2	偏移值	以基准值为中心，原点区间的上下对称浮动值，支持设定的范围为[0.1,30]。如基准值为 1°，偏移值为 3°时，原点的区间范围为[-2°,4°]。

原点区间、基准值和偏移值之间的关系如下图：



7.1.10 电子铭牌

说明

电子铭牌配置于工业机器人，安装于本体，主要用于保存和本体相关的数据，避免更换工控机或者更换控制柜导致的基本数据丢失。

电子铭牌的软件功能主要分为控制器和 RobotAssist 软件两部分，控制器负责电子铭牌的数据读取、校验、覆盖等功能，RobotAssist 软件负责电子铭牌相关操作命令的下发及数据展示等功能。控制器在开机启动后，会首先检测电子铭牌是否存在：若存在，则正常读取数据，进行数据校验，并将校验结果进行存储；若不存在且用户未选择使用电子铭牌，则直接使用控制器数据正常运行，若不存在且用户选择了使用电子铭牌，则会给出电子铭牌不存在的提

示。RobotAssist 软件在连接上控制器后，会首先检测控制器内电子铭牌数据校验的结果，并根据不同的校验结果进行弹窗提示，用户只需根据弹窗提示进行操作即可，具体操作步骤可详见 1.1.2.2 节。

开机选择

开机后，控制器会检测电子铭牌中的数据是否与控制器中的数据相同；

若相同，则会直接使用控制器内的数据，且无任何提示信息；

若不同，则会弹框提示是否使用电子铭牌中的数据，如下图；

若成功使用电子铭牌中的数据，则会默认使用电子铭牌的数据覆盖控制器中的数据。

几种出现弹框提示的情况：

1）若检测到存在电子铭牌且数据与控制器不同，则会弹框提示“是否使用电子铭牌数据”；若选择是，则使用电子铭牌中的信息；若选择否，会直接使用控制器数据；

2）选择过一次使用电子铭牌数据后，再次开机，会默认使用电子铭牌信息；当控制器信息再次与电子铭牌信息不同时，才会重新弹框提示“是否使用电子铭牌数据”；

3）若开机初始就不存在电子铭牌，则会默认使用控制器数据；但使用过电子铭牌信息后，若下次开机检测不到电子铭牌，则会提示“是否使用控制器数据”；选择是，正常使用控制器数据；选择否，控制器整体处于故障状态，无法进行操作，需要重启解决；

> 提示
>
> 1、当电子铭牌中的机型与控制器中的机型不符时，会使用电子铭牌数据失败；若想成功使用电子铭牌中的数据，需注意保持控制器中的机型数据与电子铭牌中的机型数据相同。

电子铭牌界面显示

RobotAssist 软件界面依次点击：机器人配置->设置->电子铭牌，此界面会显示电子铭牌中的相关信息；若控制器检测到电子铭牌，无论使用电子铭牌与否，此界面都会显示电子铭牌中相关参数段的信息；

电子铭牌的状态可以通过此界面的状态栏位进行判断；三个参数分别表示：电子铭牌状态，电子铭牌数据与控制器数据的匹配情况，开机是否“点击使用电子铭牌”按钮，如下图：

开机若未检测到电子铭牌，界面显示则如下图：

> 提示
>
> 1、状态栏第三个参数，只要点击使用电子铭牌数据，无论数据使用成功与否，此位置都会显示为使用。

电子铭牌界面功能介绍

功能	说明
导出控制器数据	将控制器中相关参数段的数据导出至文件中
导出电子铭牌数据	将电子铭牌中的数据导出至文件中
刷新	同步电子铭牌信息状态
基础信息	显示电子铭牌中基础信息的参数段；不可手动修改
编码器电池电压	显示编码器电池电压；每次开机或开机后的每 24h 会检测一次编码器电池电压，并显示实际的电压值；不可手动修改
运动时间	当电机有动作时，会累加运动时间；界面显示值会每 1h 刷新一次；不可手动修改
机械零点参数、运动学参数	界面会分别显示控制器的目前值及电子铭牌的目前值；不可手动修改
动力学参数	此参数段在界面隐藏显示
覆盖电子铭牌数据	使用控制器中的数据覆盖电子铭牌中的数据

提示

1、导出的所有数据均为加密显示。

2、若选择使用电子铭牌，当机器人在进行零点标定、修改本体参数、动力学辨识操作后，控制器会自动向电子铭牌中同步修改后的数据。

7.2 安全功能

7.2.1 适用范围

安全功能	工业机器人	xMate 协作机器人
软限位	Y	Y
虚拟墙	N	Y
碰撞检测	Y	Y
安全区域	N	Y
安全监视器	N	Y
协作模式	N	Y

7.2.2 软限位

功能说明

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。注意，下图仅做七轴机器人的示例，实际出现几个轴、每个轴的软限位值与机型相关。

> **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。

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 STOP 0 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。

7.2.3 虚拟墙

功能说明

针对一些医疗场景，xMate 协作机器人提供虚拟墙功能。例如将 xMate 作为医生的助力工具，用户可通过拖动进行手术操作，并通过手术导航系统设置虚拟墙，限定 xMate 法兰末端的操作空间范围。

- 虚拟墙形状类型：长方体、球形；
- 虚拟墙中心位置和限制范围可设置：
  - 中心位置以基坐标系为参考坐标系设置，单位 mm，支持以当前法兰位置设置虚拟墙中心位置；
  - 球形限制范围采用球半径设置，单位 mm；
  - 长方体限制范围采用长宽高限制，分别对应长方体在 XYZ 方向的长度，单位 mm；

操作步骤如下：

	操作	说明
1	使用 admin 级别的用户登录系统，并打开拖动模式。	虚拟墙只能在拖动模式开启时有效。
2	选择虚拟墙形状类型。	支持长方体和球形。
3	确定虚拟墙中心点。	将机器人拖动到某个位置，点击界面上的当前位置按钮，将机器人法兰中心设置为虚拟墙。
4	设置虚拟墙范围。	球形限制范围采用球半径设置，单位 mm，长方体限制范围采用长宽高限制，分别对应长方体在 XYZ 方向的长度，单位 mm。
5	打开虚拟墙。	点击打开按钮，开启虚拟墙。

7.2.4 碰撞检测

功能说明

碰撞检测功能是一种基于机器人动力学模型参数估计的被动检测功能，机器人运行过程中与外界发生意外碰撞时，碰撞检测能够及时检测出碰撞并执行预先设置的处置措施。

- 碰撞检测功能可启用或者关闭，默认关闭状态。
- 碰撞检测模式分为级别设置和单轴设置。

级别设置：针对不同的应用场景，提供了低、中、高三种不同碰撞检测灵敏度。灵敏度越高，触发碰撞检测的外力越小。低灵敏度对应满负载全速情况，中灵敏度对应半负载 50%自动运行状态，高灵敏度对应 JOG 或是协作模式。例如用户在使用 jog 功能移动机器人时，想要开启碰撞检测功能，可以选择高灵敏度。在自动模式下满负载全速运行程序时，选择低灵敏度。

单轴设置：提供了针对特定应用场景，精调碰撞检测灵敏度的接口。用户可以根据 HMI 提示的碰撞信息，依次调节各轴的灵敏度。设置适用于当前应用场景的灵敏度，同时兼顾碰撞的灵敏度和稳定性。

- 碰撞检测提供了触发暂停和安全停止两种触发行为

触发暂停：在检测到碰撞时，机器人会暂停当前运动，当我们给机器人一个下压的力时，机器人能够继续之前的运动；

安全停止：在检测到碰撞时，机器人以一种安全的方式停止当前运动。

注意事项

1.在程序执行过程中机器人高速运动并与外部设备发生了硬碰撞（stiff collision），碰撞力过大直接导致伺服驱动器报警停机，清除碰撞后重启机器人复位伺服报警，才能重新运行机器人。

2.灵敏度模式选择不对，可能会引起机器人碰撞误报，请根据不同的应用场景，选择不同的灵敏度阈值。

3.碰撞检测灵敏度受机器人硬件影响，不同的机器人之间灵敏度阈值存在差异。目前三种灵敏度模式只是提供了一套标称值。用户若对碰撞检测灵敏度有更高需求，可通过单轴设置基于特定应用场景精调各轴灵敏度，或是通过 RL 指令来在线调整检测灵敏度。

⚠	警告

在使用碰撞检测之前必须要确保以下参数已经正确设置，否则控制器无法计算正确的输出力矩，可能会产生误报。

1、 机器人的型号

2、 机器人的安装方式

3、 负载信息（工具）

4、 机械和传感器零点

5、 本体参数信息

7.2.5 安全区域

功能说明

安全区域限制了机器人活动空间，用户可以定义空间中某一片区域使得机器人在进入该区域时触发协作模式或者 Stop1 停机。

参数说明

1.安全区域提供立方体或者平面两种定义方式：立方体或点面矢量区域。

2.进入安全区域的行为模式有触发协作模式或触发 Stop1 急停。

3.安全区域方面支持正/负方向设置，正向表示当机器人进入了用户设置的安全区域触发用户定义的行为模式；负向表示机器人离开用户设置的安全区域触发用户定义的行为模式。

安全区域设置方式

1.当选择立方体类型时，用户可通过设置基坐标系下 xyz 的范围来设置空间立方体区域。

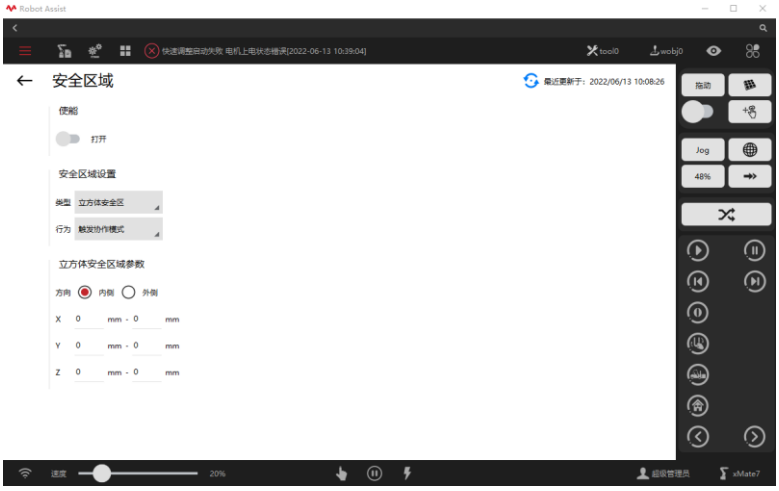
2.当选择点面矢量区域时，支持三点法构建平面，用户 Jog 机器人到点 a，点击界面上的点 1，则将 a 点记录为平面的第一个点，同理依次 Jog 机器人到点 b、c，存为点 2、3。按照 abc 点序，根据右手定则可确定 abc 组成平面的法线正向，点击保存当前平面参数，则构建好 abc 三点组成的平面。若用户想继续添加平面，点击点面安全区参数方框左下角的 "+" 即可。

示例

示例 1

假设用户想设置一个立方体区域（如下图红框），使得机器人工具末端进入红框之后，触发 Stop1 急停。以此为例，安全区域开启步骤如下：

	操作	说明
1	使用 admin 级别的用户登录系统，打开安全区域使能开关。	安全区域默认关闭状态。
2	选择安全区域空间类型为立方体。	支持立方体和点面矢量区域。
3	选择进入安全区域触发的行为 Stop1 急停。	支持协作模式和 Stop1 急停，触发协作模式需要协作模式功能开启才能生效。
4	选择立方体方向。	正向为进入安全区域，机器人触发行为。
5	设置立方体范围。	xyz 坐标为基坐标系下的坐标。



示例 2

假设用户想通过点面矢量法，构造由两个平行的平面 M,N 组成的空间区域，使得机器人工具末端进入这片区域之后，触发 Stop1 急停。以此为例，安全区域开启步骤如下：

	操作	说明
1	使用 admin 级别的用户登录系统，打开安全区域使能开关	安全区域默认关闭状态。
2	选择安全区域空间类型为点面矢量区域	支持立方体和点面矢量区域。
3	选择进入安全区域触发的行为为 Stop1 急停	支持协作模式和 Stop1 急停，触发协作模式需要协作模式功能开启才能生效。
4	选择立方体方向。	正向为进入安全区域，机器人触发行为。
5	点击安全区参数框左下角的“+”	新建平面 M
6	将机器人依次 Jog 到平面 M 的三个点 a,b,c	按照 abc 点的排列顺序，依据右手定则确定平面 M 的法线正向
7	点击页面左下角的保存当前平面参数	将平面 M 更新为 abc 点构成的平面
8	重复步骤 5，6，7，构建平面 N	注意平面 N 的法线方向要与平面 M 相反

7.2.6 安全监视器

说明

安全监视是对机器人正常运行状态下的安全监控器，该模式相较于协作模式各监控项目阈值要更大。

参数配置

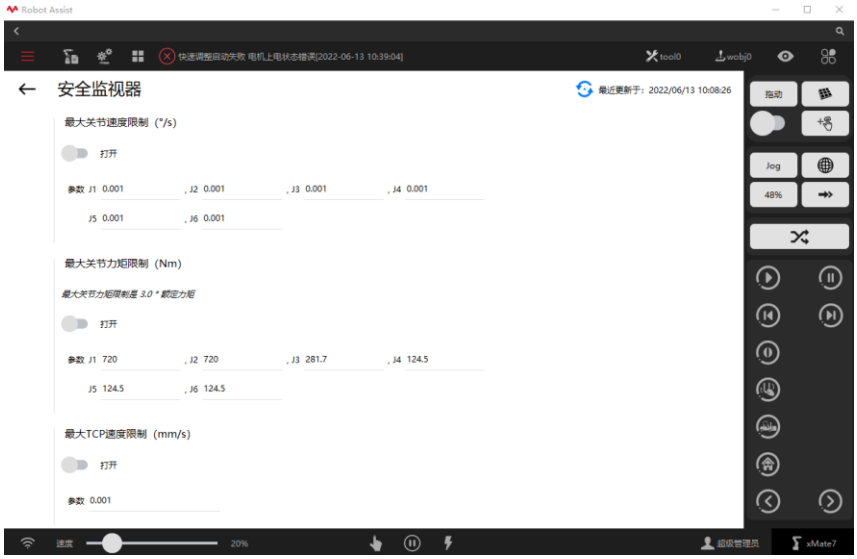
需要获得 admin 以上的权限才能对安全监控功能进行配置，界面如下所示：

1.设置安全监控功能界面，用以设置各监控项目及参数，包括：

A	关节最大速度限制，各轴独立设置， 各关节最大速度：【180， 150， 180， 180， 225， 225， 225】°/s
B	TCP 最大线速度限制：1m/s, X/Y/Z 共用一个限制阈值，参数设置范围 0.0~1m/s
C	关节最大力矩限制，各轴独立设置， xMate3 Pro 各关节最大力矩：【281.7，338.1，281.7，281.7，99.6， 99.6， 99.6】Nm. xMate7 Pro 各关节最大力矩：【720，720，281.7，281.7，124.5， 124.5， 124.5】Nm.
D	总功率限制：4476W,参数设置范围：0~4476W

2.监控项目触发，机器人采用 STOP1 处理，

3.各监控项目可独立启用/关闭，默认关闭状态。



7.2.7 协作模式

说明

协作模式是一种在人员和机器人共享工作区域时的工作模式，进入该模式，机器人运行速度会根据协作模式 TCP 最大速度监控参数设置进行降速处理。

参数配置

需要获得 admin 以上的权限才能对协作模式功能进行配置，界面如下所示：

1.协作模式下设置的监控参数范围如下:

A	关节最大速度限制：15°/s，各轴独立设置，参数设置范围：0.0~15.0°/s
B	TCP 最大线速度限制：0.25m/s, X/Y/Z 共用一个限制阈值，参数设置范围 0.0~0.25m/s
C	关节最大力矩限制，各轴独立设置， 各关节最大力矩：【140.9，169.0，140.9，140.9，49.8，49.8，49.8】Nm.
D	总功率限制：192.7 W,参数设置范围：0~192.7 W

2.协作模式下，各监控参数超出限定值，采用 STOP1 停止处理。

7.3 通信配置

7.3.1 系统 IO 配置

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 xCore 控制系统发送各种指令，如电机上电，启动程序，急停复位等，xCore 系统也可使用系统输出功能对外发送各种状态。

系统输入

xCore 系统支持的系统输入包括:

编号	系统输入名称	备注
1	上电	
2	下电	
3	启动程序	
4	暂停程序	
5	指针到 main	
6	进入协作模式	仅协作机器人
7	退出协作模式	仅协作机器人
8	清除报警	

9	上电且运行	依次执行上电、pptomain、运行
10	上电并继续运行	依次执行上电、运行
11	下电并暂停	暂停，待机器人停止后下电
12	急停复位并清除报警	
13	手动模式	
14	自动模式	只有该功能在手动模式生效

所有的系统输入均为脉冲触发，为了保证 xCore 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

xCore 系统支持的系统输出包括：

编号	系统输出名称	输出有效	输出无效	备注
1	上下电状态	电机上电	电机下电	
2	运行状态	程序运行	程序未运行	
3	操作模式	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	碰撞触发	触发	未触发	仅协作机器人
6	协作模式	协作模式状态	非协作模式状态	仅协作机器人
7	报警状态	报警状态	无报警	
8	Home 状态	机器人 TCP 到 home 点	机器人 TCP 未处于 home 点	

除"工作模式"信号外，其他所有的系统输出均是高电平有效。

对于"工作模式"信号，自动模式时输出为高电平，手动模式输出为低电平。

> 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 xCore 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。

7.3.2 外部通信

说明

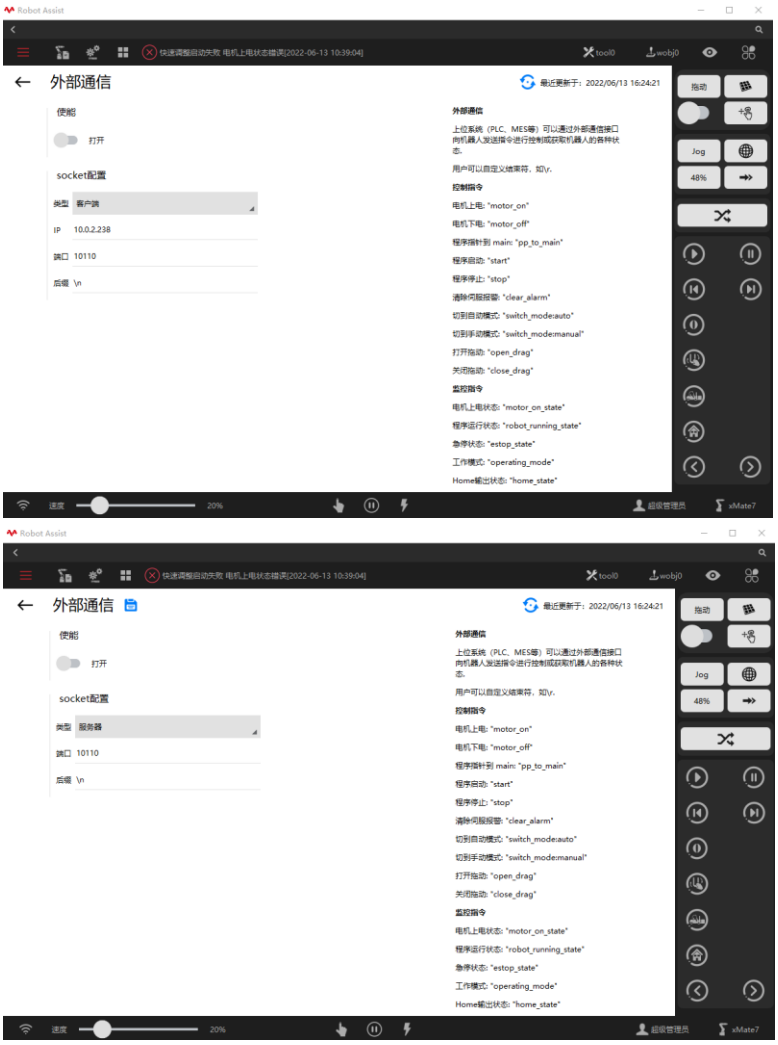
xCore 系统提供了基于 Socket 的外部通信接口，上位系统（PLC、MES 等）可以通过该接口向机器人发送控制指令或者获取机器人的各种状态。

Socket 通信接口支持界面：配置 IP 地址、端口号、通讯结束符（后缀）。

Socket 通信接口支持机器人作为客户端和服务端，但同一时刻仅能作为一种状态使用。

启动接口

在使用各项交互指令之前，首先需要配置 Socket 通信相关参数，并打开该功能。该项操作在示教器的界面上操作，通过“机器人”->“通信”->“外部通信”页面开启，如下图所示：



配置参数

当机器人作为客户端使用时，需要配置参数如下：

序号	配置参数	说明
1	IP	服务端 IP，比如连接的 PLM、MES 系统 IP 地址。
2	端口	服务端监听端口
3	后缀	服务端向机器人发送控制指令或者监控指令时，指令结束处需要附加的后缀字符。通常使用简单结束符\r 或者\n 或者\t。注意此处可以使用组合后缀且长度不限制，例如\r\n、\r\t、\r\n\t；也可以使用可见字符，例如字母等。

当机器人作为服务端使用时，支持多连接，如果使用多连接需要客户端注意处理控制时序可

冲突。需要配置参数如下：

序号	配置参数	说明
1	端口	服务端监听端口
2	后缀	服务端向机器人发送控制指令或者监控指令时，指令结束处需要附加的后缀字符。通常使用简单结束符\r 或者\n 或者\t。注意此处可以使用组合后缀且长度不限制，例如\r\n、\r\t、\r\n\t；也可以使用可见字符，例如字母等。

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，假设用户使用"\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：

	指令名称	发送的字符串	返回值	备注
1	关闭 socket 接口	"xCore::SocketInterface::Disable" +"\r"	无返回值	
2	启动 socket 接口	"xCore::SocketInterface::Enable" +"\r"	无返回值	
3	启动程序	"start"+"\r"	成功返回"true"; 失败返回"false"	
4	停止程序	"stop"+"\r"	成功返回"true"; 失败返回"false"	
5	清除伺服报警	"clear_alarm"+"\r"	成功返回"true"; 失败返回"false"	
6	程序指针到 main	"pp_to_main"+"\r"	成功返回"true" ;失败返回"false"	
7	电机上电指令	"motor_on"+"\r"	成功返回"true"; 失败返回"false"	
8	电机下电指令	"motor_off" + "\r"	成功返回"true"; 失败返回"false"	
9	切换至手动模式	"switch_mode:manual"+"\r"	成功返回"true"; 失败返回"false"	
10	切换至自动模式	"switch_mode:auto"+"\r"	成功返回"true"; 失败返回"false"	
11	打开拖动	"open_drag"+"\r"	成功返回"true"; 失败返回"false"	仅协作机器人
12	关闭拖动	"close_drag"+"\r"	成功返回"true"; 失败返回"false"	仅协作机器人

支持监控命令：

	指令名称	发送的字符串	返回值	备注
1	电机上电状态	"motor_on_state" + "\r"	成功返回"true";失败返回"false" true:电机上电，false:电机下电	
2	程序运行状态	"robot_running_state" + "\r"	成功返回"true";失败返回"false"	

			true:运行状态，false:非运行状态	
3	急停状态	“estop_state” + “\r”	成功返回”true”;失败返回”false” true:急停状态，false:非急停状态	
4	故障状态	“fault_state” + “\r”	成功返回”true”;失败返回”false” true:故障状态，false:非故障状态	
5	工作模式	“operating_mode” + “\r”	成功返回”true”;失败返回”false” true:自动模式，false:手动模式/等待模式	
6	获取笛卡尔位置	“cart_pos” + “\r”	笛卡尔位置字符串+”\r”	
7	获取笛卡尔位置	“cart_pos_name” + “\r”	“cart_pos : ”+笛卡尔位置字符串+”\r”	
8	获取轴位置	“jnt_pos” + “\r”	轴位置字符串+”\r”	
9	获取轴位置	“jnt_pos_name” + “\r”	“jnt_pos : ”+轴位置字符串+”\r”	
10	获取轴速度	“jnt_vel” + “\r”	轴速度字符串+”\r”	
11	获取轴速度	“jnt_vel_name” + “\r”	“Jnt_vel : ”轴速度字符串+”\r”	
12	获取轴力矩	“jnt_trq” + “\r”	轴力矩字符串+”\r”	
13	获取轴力矩	“jnt_trq_name” + “\r”	“Jnt_trq : ”轴力矩字符串+”\r”	
14	Home 状态输出	“home_state” + “\r”	有输出返回”true”;无输出返回”false”	
16	碰撞检测状态	“collision_state” + “\r”	触发碰撞返回”true”;无碰撞返回”false”	仅协作机器人
17	获取机器人任务状态	“task_state” + “\r”	机器人当前在进行的任务。包括: ● 就绪：ready ● Jog：jog ● 负载辨识：load_identify ● 动力学辨识：dynamic_identify ● 拖动打开：drag ● 程序运行中：program ● Demo：demo ● RCI：rci ● 调试：debug	详见 3.1.2”底部状态栏”章节中关于机器人当前状态的图标和描述

备注:

	字符串格式	单位
笛卡尔位置	x、y、z、a、b、c、q1、q2、q3、q4	x、y、z 单位 mm；a、b、c 单位为度；q1~q4 是姿态四元数
轴位置	j1、j2、j3、j4、j5、j6、j7	机器人轴角度单位 rad；导轨位置单位 m；
轴速度	vj1、vj2、vj3、vj4、vj5、vj6、vj7	机器人轴速度单位 rad/s，导轨速度单位是 m/s；
轴力矩	tj1、tj2、tj3、tj4、tj5、tj6、tj7	机器人轴与导轨力矩的单位都是电机额定力矩的千分比；

7.3.3 总线设备

说明

目前支持 CC-Link、Modbus、EtherCAT、PROFINET 四种总线。CC-Link 包括了 CC-Link 设备（通过 EtherCAT 转接）和 CC-Link IE Field Basic；EtherCAT 可用于扩展 IO 模块、PROFINET、EtherNet/IP 等其他总线模块。

支持总线	协议	支持方式	备注
Modbus	TCP	master 和 slave	

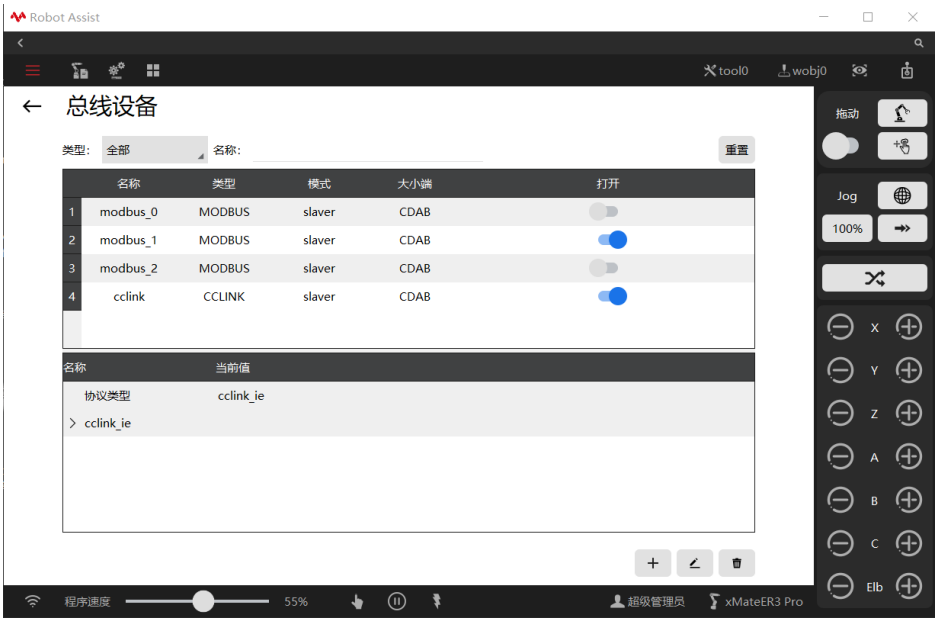
	UDP	不支持	
	RTU	master 和 slave	仅工业机器人
CC-Link	485	远程设备站（slave）	仅工业机器人
	IE Field Basic	远程设备站（slave）	

Modbus 支持功能码如下：

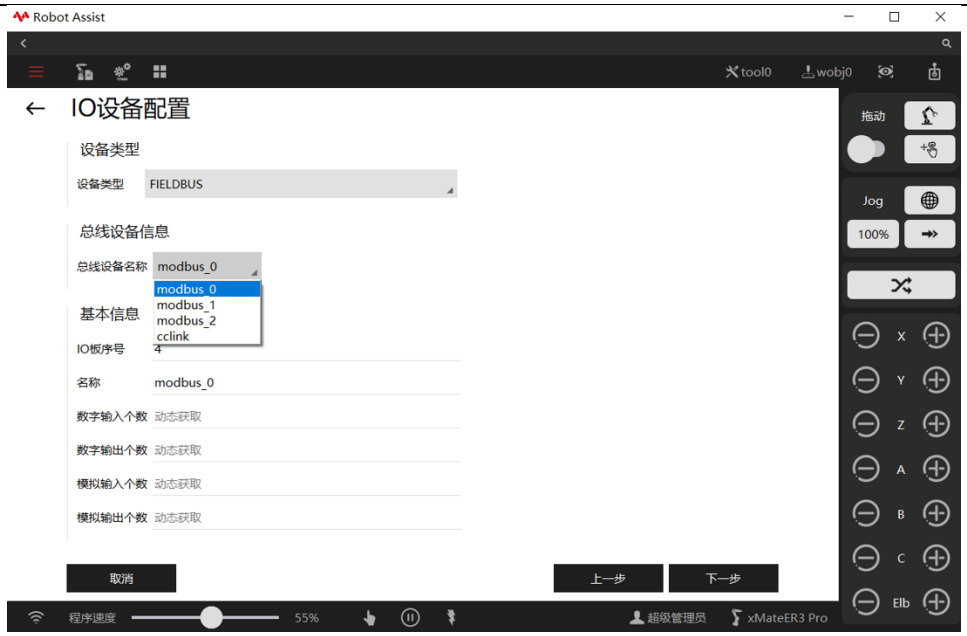
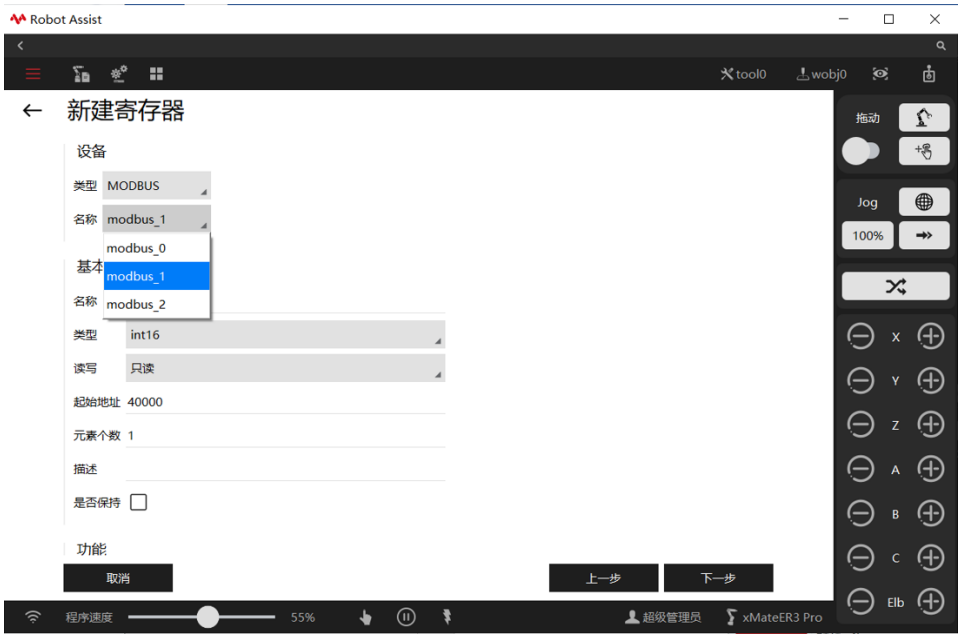
功能码	含义	支持情况
0x01	读线圈	支持
0x05	写单个线圈	支持
0x0F	写多个线圈	支持
0x02	读离散量输入	支持
0x04	读输入寄存器	不支持
0x03	读保持寄存器	支持
0x06	写单个保持寄存器	支持
0x10	写多个保持寄存器	支持

配置方式

通过“机器人配置”->“通信”->“总线设备”进入。页面分为上下两部分，上半部分管理所有的总线连接，下半部分是某一个总线连接的属性参数。上半部分的表格中，可以对每条总线连接单独进行打开和关闭操作。当总线连接关闭状态时，”状态监控”->”IO 信号”中将不会显示这个连接配置的 IO。



参数名	参数说明
名称	管理列表第一列是名称，这个名称在配置”IO 设备”和”寄存器”时均会使用到。比如上图的名称分别为 modbus_0、modbus_1、modbus2、 cclink，引用处如下图所示： ➢ “IO 设备配置”引用该名称字段，用来表示 IO 设备跟哪个总线相关； ➢ “寄存器”引用该字段，表示寄存器与哪条总线相关；

	 
类型	管理列表的第二列是类型，在新增/编辑总线设备时可以选择，目前支持 CC-Link、Modbus、EtherCAT、PROFINET 四种。CC-Link 包括了 CC-Link 设备（通过 EtherCAT 转接）和 CC-Link IE Field Basic；EtherCAT 可用于扩展 IO 模块、Profinet、EtherNet/IP 等其他总线模块。
模式	管理列表的第三列是模式，显示当前机器人是作为总线上的主站还是从站。
大小端	管理列表的第四列是大小端，主要用于寄存器。由于每个寄存器占用 2 个字节，两个字节的十六进制排列顺序有多种，此属性需要主站和从站对应，否则数据将不符合预期。目前大小端支持 ABCD、CDAB、BADC、DCBA 四种，控制系统默认使用 CDAB 的方式。
打开按钮	管理列表的第五列是打开按钮。可以操作该按钮，用来启用总线功能或者关闭总线功能。每个总线设备可以单独打开和关闭。注意关闭某个总线设备后，该总线设备上配置的 IO 将不会显示在”状态监控”->”IO 信号”中。

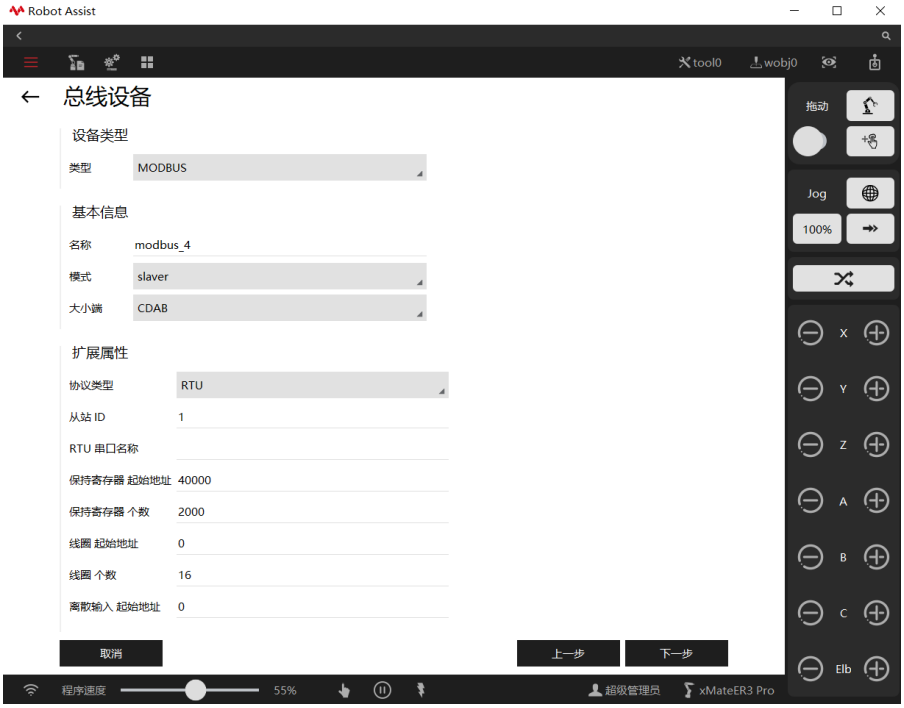
7.3.3.1 新增 modbus 通信

在总线设备页面，点击右下角  ，进入新增通信总线设备页面，选择设备类型为”MODBUS”。支持 TCP 和 RTU 两种协议方式，支持主站和从站两种总线配置。

7.3.3.1.1 Modbus TCP 配置

参数	简介
模式	可以选择机器人作为主站”master”或者从站”slave”；
从站 ID	机器人做从站时，注意总线整体配置，不要与其他从站冲突。机器人做主站时，此配置表示机器人期望通讯的目标从站 ID。注意目前机器人作为主站时，仅支持与外部设备单从站通信；
TCP/IP	机器人作从站时，填入 0.0.0.0 即可，表示监听所有网卡。机器人做主站时，此项需要填入对应通信的从站的 IP 地址；
TCP 端口	从站使用 TCP 协议时的端口号；
保持寄存器起始地址	功能码 0x03、0x06、0x10 所影响的寄存器的起始地址，1 个寄存器占 2 个字节；
线圈起始地址	功能码 0x01、0x05、0x0F 所影响的寄存器的起始地址；
离散输入起始地址	功能码 0x02 所影响的寄存器的起始地址

7.3.3.1.2 Modbus RTU 配置



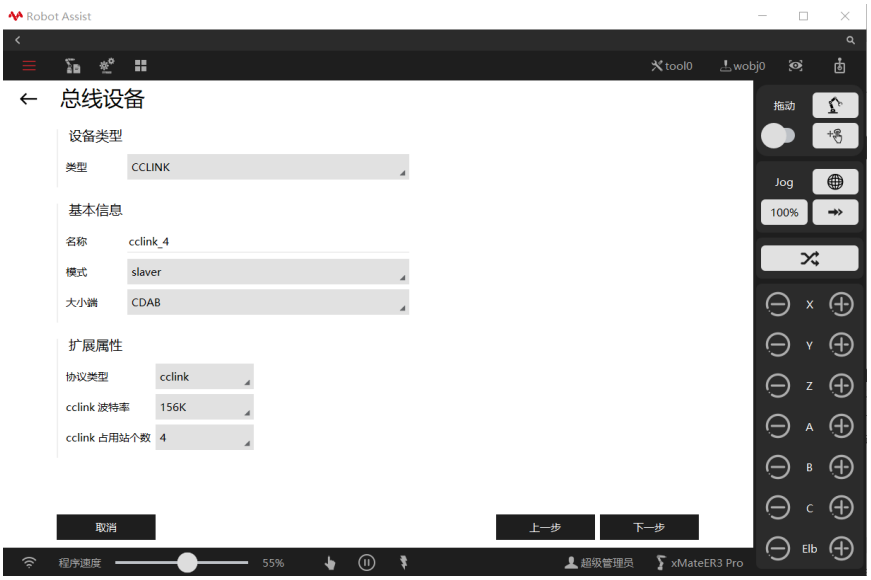
部分概念与 Modbus TCP 相同，不另行说明，下面仅介绍不同之处。

RTU 串口名称：表示总线通信应用的串口介质，此项需要再“机器人配置”->“通信”->“串口设置”中进行配置，包括通信使用的参数。

7.3.3.2 新增 CC-Link 通信

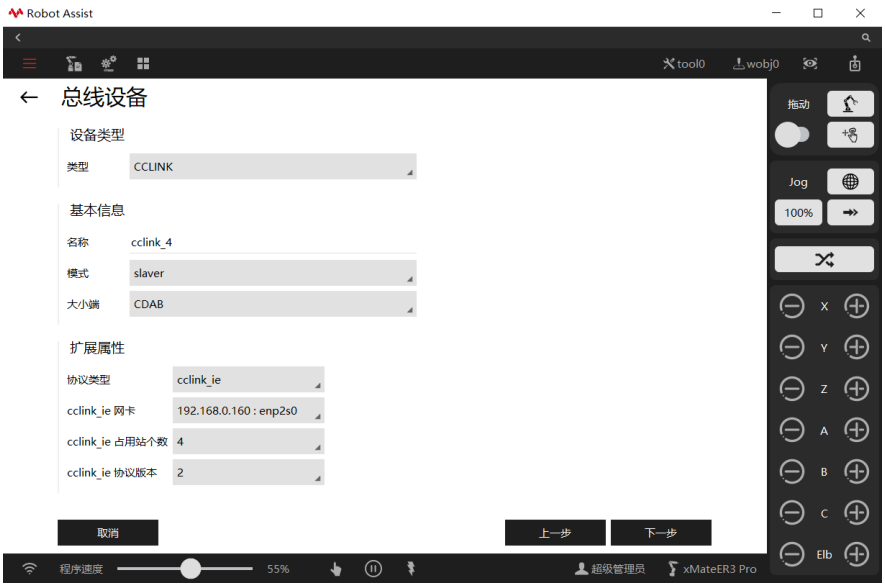
在总线设备页面，点击右下角  ，进入新增通信总线设备页面，选择设备类型为”CCLINK”。支持 CC-Link 和 CC-Link IE Field Basic 两种协议方式，仅支持从站总线配置。

7.3.3.2.1 CC-Link 配置



参数	简介
模式	CC-Link 和 CC-Link IE Field Basic 仅支持作为从站使用
协议类型	cclink 表示使用 EtherCAT 扩展的 CC-Link 模块
cclink 波特率	通信波特率，注意主站与从站配置要匹配
cclink 占用站个数	可以配置 1~16 个占用站，默认为 4 个。推荐使用模式设置

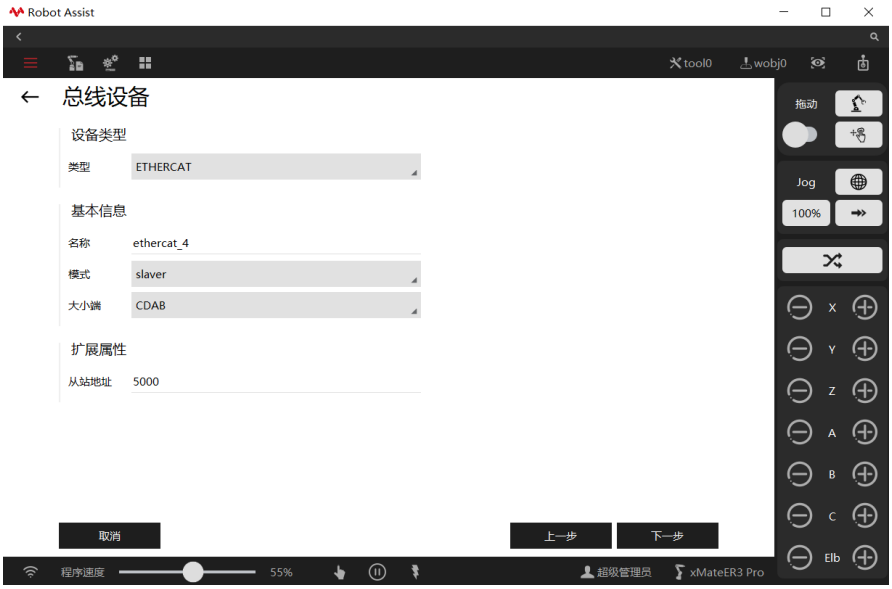
7.3.3.2.2 CC-Link IE Field Basic 配置



参数	简介
协议类型	cclink_ie 表示直接使用机器人以太网口进行的 CC-Link IE Field Basic 通信协议
cclink_ie 网卡	配置通信使用哪个以太网卡进行
cclink_ie 占用站个数	可以配置 1~16 个占用站，默认为 4 个。推荐使用模式设置
cclink_ie 协议版本	可选 Ver1 或者 Ver2，此选项需要与主站保持一致

7.3.3.3 新增 EtherCAT 通信

在总线设备页面，点击右下角  ，进入新增通信总线设备页面，选择设备类型为”ETHERCAT”。EtherCAT 扩展设备可以用来接入 PROFINET、EtherNet/IP 等模块。



从站地址：此参数为 EtherCAT 拓扑中的从站地址号。由于机器人内部设备已经占用了 EtherCAT 从站编号 1000~4000，为了避免设备地址冲突，用户自行扩展的 EtherCAT 设备从站编号要求不小于 5000。

7.3.3.4 新增 PROFINET 通信

配置方式

在总线设备页面，点击右小角  ，进入新增通信总线设备页面，选择设备类型为”PROFINET”。仅支持从站总线配置。槽 1 至槽 6 的模型选择须和通信伙伴侧配置保持一致。

参数说明：

参数	说明
设备类型	就是 PROFINET
模式	仅支持 slaver
大小端	一般选择 DCBA，具体选项根据通信双方约定而定

网卡	选择跟通信伙伴连接的网口；包含了网卡 IP 和网卡名称
更新周期(ms)	默认是 10ms，最小支持 2ms
槽 1 类型	只能选择 DO_256 模型，表示有 256 个数字量通过槽 1 从机器人输出到通信伙伴
槽 2 类型	只能选择 DI_256 模型，表示有 256 个数字量通过槽 2 从通信伙伴输入到机器人
槽 3 类型	选项模型有 AO_Int16_8/ AO_Int16_16/ AO_Int16_32/ AO_Int16_64/ AO_Int16_128/ AO_Int16_256。其中，AO_Int16_8 表示有 8 个 int16 的模拟量通过槽 3 从机器人输出到通信伙伴，其他依次类推
槽 4 类型	选项模型有 AI_Int16_8/ AI_Int16_16/ AI_Int16_32/ AI_Int16_64/ AI_Int16_128/ AI_Int16_256。其中，AI_Int16_8 表示有 8 个 int16 的模拟量通过槽 4 从通信伙伴输入到机器人，其他依次类推
槽 5 类型	选项模型有 AO_Float32_8/ AO_Float32_16/ AO_Float32_32/ AO_Float32_64/ AO_Float32_128/ AO_Float32_256。其中，AO_Float32_8 表示有 8 个 float32 的模拟量通过槽 5 从机器人输出到通信伙伴，其他模型依次类推
槽 6 类型	选项模型有 AI_Float32_8/ AI_Float32_16/ AI_Float32_32/ AI_Float32_64/ AI_Float32_128/ AI_Float32_256。其中，AI_Float32_8 表示有 8 个 float32 的模拟量通过槽 6 从通信伙伴输入到机器人，其他模型依次类推

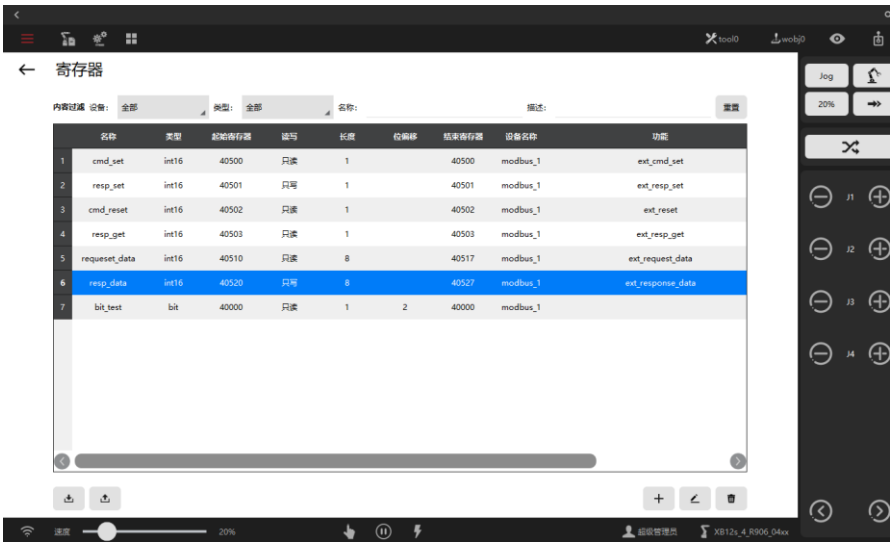
7.3.4 寄存器

说明

寄存器表示机器人内的可用变量，可以用来与外部设备进行数据交换，从而达到控制机器人的作用。寄存器也可以在当前 RL 工程中作为变量使用，支持两种操作寄存器变量的方式：指令式和赋值式。寄存器是机器人本身的概念，而不是属于总线设备，具体到某个寄存器是可以单独绑定到某个总线设备上来进行通信和数据交换的，需要在新建或者编辑寄存器时指定这种绑定关系。每个寄存器占 2 个字节。不同类型的变量占用的寄存器数量不同。

配置方式

可以在“机器人配置”->“通信”->“寄存器”页面，通过右下角的三个按钮对寄存器变量进行添加。



名称

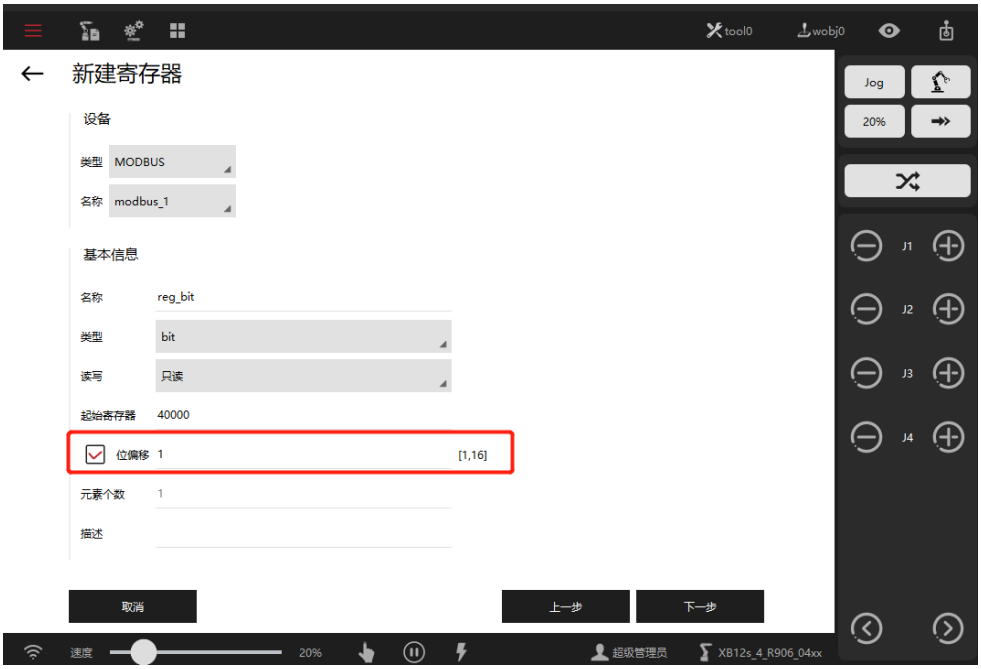
寄存器列表第一列是名称，该名称将会用于 RL 访问寄存器变量，注意列表中不可重名，也不可以和 RL 中任何列表中的变量重名，否则 RL 将出现变量冲突。

类型

寄存器列表第二列是类型，支持 bit、bool、int16、float 四种。具体说明如下

序号	类型	说明
1	bit	bit 类型的寄存器变量仅占用一个寄存器某 1 位，其中 bit 数组需要以 16 位的整数倍出现。如上图所示，寄存器 41000 位 bit 类型寄存器，长度为 64，则占用 41000~41003 这 4 个寄存器。
	bool	一个 bool 变量占用 1 个寄存器
	int16	一个 int16 变量占用 1 个寄存器
	float	float 类型的寄存器占用 2 个寄存器

bit 类型寄存器的几点说明：



- 如上图所示，如果勾选了位偏移，表示占用一个寄存器某一位，可选值为 1-16
- bit 类型的寄存器只有勾选了位偏移，才允许功能绑定。
- bit 类型的寄存器勾选了位偏移后，元素个数会被自动修改成 1

参数说明

参数	说明
起始寄存器	寄存器列表第三列是起始寄存器，用于表示寄存器的起始地址。所有寄存器的地址不可交叉占用，否则会发生寄存器冲突错误。例如定义了寄存器占用 41000~41003，则不能定义另一个寄存器从 41002 开始。
读写	寄存器列表第四列是读写属性，表示该寄存器在机器人的视角（而不是主站或从站视角）下是对其进行读操作还是写操作。机器人需要对外输出的状态，就是只写寄存器（机器人做从站时为保持寄存器功能码 0x03，机器人做主站时为保持寄存器功能码 0x06 或 0x10）；机器人需要获取从外部设备发过来的命令，就是只读寄存器（机器人做从站时为保持寄存器功能码 0x06 或 0x10，机器人做主站时为保持寄存器功能码 0x03）。
长度	寄存器列表第五列是长度，表示该变量的数量，对于大于 1 的变量，可以使用数组的方式进行变量引用，下标从 1 开始。注意此项要区别于寄存器数量。例如上图，寄存器 40140~40153，该变量类型为 float，每个 float 占

<table>
<tr><td></td><td>用 2 个寄存器。因此长度为 7，占用寄存器数量为 2*7=14 个。示例可以在 RL 程序中通过 mtcp_wo_cartvel[1]~mtcp_wo_cartvel[7]来引用这个寄存器的内容。</td></tr>
<tr><td>位偏移</td><td>bit 类型寄存器映射到寄存器的位置，一个寄存器占两个字节既 16 位，位偏移是指对应寄存器的位置，偏移值为 1-16。创建 bit 类型寄存器时，如果没有勾选位偏移时显示为空。</td></tr>
<tr><td>结束寄存器</td><td>寄存器列表第六列是结束寄存器，用于表示寄存器变量所占用的最后一个寄存器地址。在寄存器变量连续排布时，此列将有助于用户快速对寄存器分配进行规划。例如可以通过此项的数值加 1 来确定下一个寄存器的起始地址。</td></tr>
<tr><td>设备名称</td><td>寄存器列表第七列是设备名称。设备名称是通过总线设备创建时定义的名称，表示寄存器绑定到哪一个总线设备商。支持将寄存器绑定到 CC-Link、CC-Link IE Field Basic、Modbus、EtherCAT 设备上。</td></tr>
<tr><td>功能</td><td>寄存器列表第八列是功能，这一列中的内容是一些固定的功能码，用于表示该寄存器对应的机器人功能。功能码是固定不可修改的，描述如下表所示。
功能码分为只读和只写功能码。
只读类型
<table>
<tr><th>ID</th><th>功能码名称</th><th>支持绑定类型</th><th>对应功能</th></tr>
<tr><td>1</td><td>空白</td><td>无</td><td>无功能，用户自定义输入</td></tr>
<tr><td>2</td><td>ctrl_clear_alarm</td><td>bit,int16，bool</td><td>清除伺服报警</td></tr>
<tr><td>3</td><td>ctrl_estop_reset</td><td>bit,int16，bool</td><td>急停复位</td></tr>
<tr><td>4</td><td>ctrl_motor_on_off</td><td>bit,int16，bool</td><td>电机上电和下电
1：上电；0：下电</td></tr>
<tr><td>5</td><td>ctrl_pptomain</td><td>bit,int16，bool</td><td>程序指针到 main</td></tr>
<tr><td>6</td><td>ctrl_program_start_stop</td><td>bit,int16，bool</td><td>程序运行/停止</td></tr>
<tr><td>7</td><td>ctrl_set_program_speed</td><td>bit,int16，bool</td><td>设置程序运行速率</td></tr>
<tr><td>8</td><td>ctrl_switch_operation_auto_manual</td><td>bit,int16，bool</td><td>切换自动模式和手动模式
1：自动模式；0：手动模式</td></tr>
<tr><td>9</td><td>ext_cmd_set</td><td>bit,int16，bool</td><td>远程控制功能：下发指令
见“远程控制”</td></tr>
<tr><td>10</td><td>ext_reset</td><td>bit,int16，bool</td><td>远程控制功能：整体功能重置
见“远程控制”</td></tr>
<tr><td>11</td><td>ext_resp_get</td><td>bit,int16，bool</td><td>远程控制功能：确认并清除上一条命令的响应</td></tr>
<tr><td>12</td><td>ext_request_data</td><td>int16 数组</td><td>远程控制功能：指令功能码。数组，固定长度 8 寄存器。</td></tr>
</table>
说明：以上系统寄存器，所有的系统输入均为脉冲触发，为了保证 xCore 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 60 毫秒。

只写类型
<table>
<tr><th>ID</th><th>功能码名称</th><th>支持绑定类型</th><th>对应功能</th></tr>
<tr><td>1</td><td>空白</td><td></td><td>无功能，用户自定义输出</td></tr>
<tr><td>2</td><td>ext_error_code</td><td>int16</td><td>远程控制功能：错误码</td></tr>
<tr><td>3</td><td>ext_resp_set</td><td>bit,int16，bool</td><td>远程控制功能：指令执行完毕的响应</td></tr>
<tr><td>4</td><td>ext_response_data</td><td>int16 数组</td><td>远程控制功能：需要反馈的数据。数组，固定长度 8 寄存器。</td></tr>
<tr><td>5</td><td>sta_alarm</td><td>bit,int16，bool</td><td>伺服报警状态
1：有伺服报警；0：无报警</td></tr>
</table></td></tr>
</table>

		6	sta_error_code	int16	机器人错误码。 读到的错误码=机器人实际错误码-30000	
		7	sta_collision	bit,int16，bool	碰撞检测状态 1：检测到碰撞；0：无碰撞	
		8	sta_error_code	int16	机器人上报错误码 该错误码仅为编号，实际错误需要此编号减去 30000 以后查表获得	
		9	sta_estop	bit,int16，bool	急停状态 1：当前触发急停；0：正常	
		10	sta_home	bit,int16，bool	机器人法兰中心是否在 home 点 1：处于 home 点；0：不在 home 点	
		11	sta_motor	bit,int16，bool	电机上电状态 1：已上电；0：未上电	
		12	sta_operation_mode	bit,int16，bool	当前操作模式 1：自动模式；0：手动模式	
		13	sta_program	bit,int16，bool	当前是否处于程序运行状态 1：程序运行中；0：空闲	
		14	sta_program_speed	int16	查询当前程序运行速度，百分比的数值。	
		15	sta_cart_pose	float 数组	查询当前机器人笛卡尔位姿，绑定寄存器的要求：float 数组，长度为 8	
		16	sta_jnt_pose	float 数组	查询当前机器关节角度，绑定寄存器的要求：float 数组，长度为 8	
		17	sta_jnt_trq	float 数组	查询当前机器人关节力矩，绑定寄存器的要求：float 数组，长度为 8	
		18	sta_jnt_vel	float 数组	查询当前机器人关节速度，绑定寄存器的要求：float 数组，长度为 8	
		19	sta_robot_is_busy	int,bit,bool	当前机器人是否在执行 pptomain 等耗时操作: 1：正在执行； 0：空闲	

使用方式

控制系统提供两种方式来对寄存器进行读取、修改操作，分别是指令式和赋值式。

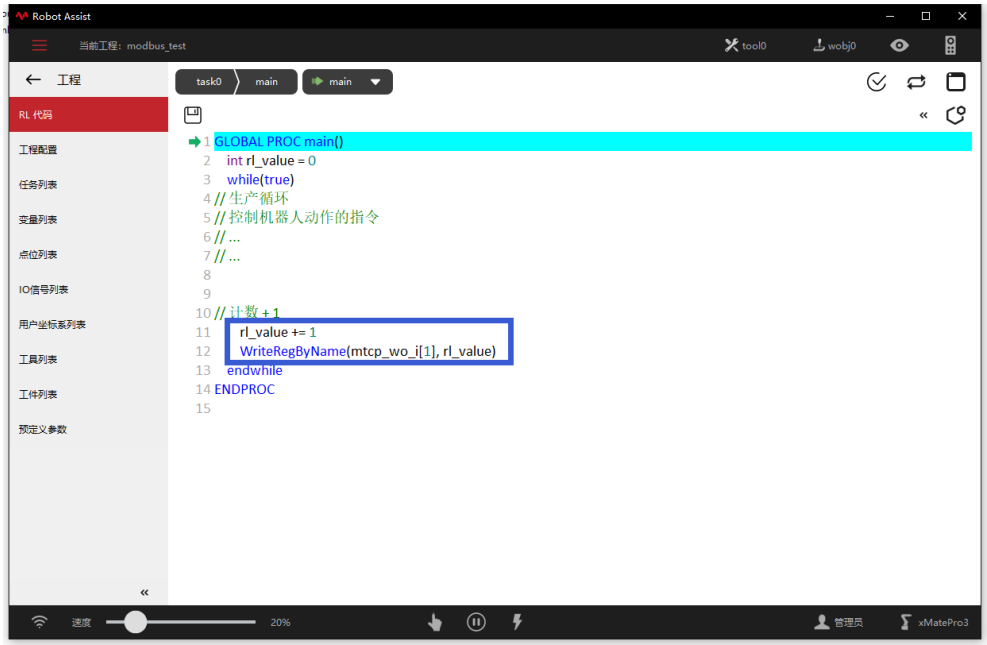
指令式提供 WriteRegByName 和 ReadRegByName 两条指令；赋值式则更直观简单些，使用操作符"="。

1）指令式:

- WriteRegByName(modbus_reg[index], rl_symbol)

第一个参数是寄存器在"机器人"->"寄存器"配置中的名字，可以使用[index]在对应寄存器的首地址再进行偏移，限制 1 <= index <= 寄存器最大长度，默认 index = 1

控制系统中的数据可以通过寄存器输出到其绑定的设备中（比如 RL 语言的循环次数），假设它在控制系统中定义为"int rl_value"，如果要把它输出到外部设备，可以约定一个寄存器，比如默认配置中的"mtcp_wo_i"的第一个寄存，则只需要在 RL 语言中添加一条 WriteRegByName 指令，参数将通过"寄存器"-"总线设备"渠道发送到外部设备。



➢ ReadRegByName(modbus_reg[index], rl_symbol)

和 WriteRegByName 指令十分类似，其功能是将寄存器内部的值更新到 RL 程序的变量中，可以用来控制 RL 程序的执行流程、运动参数等等。

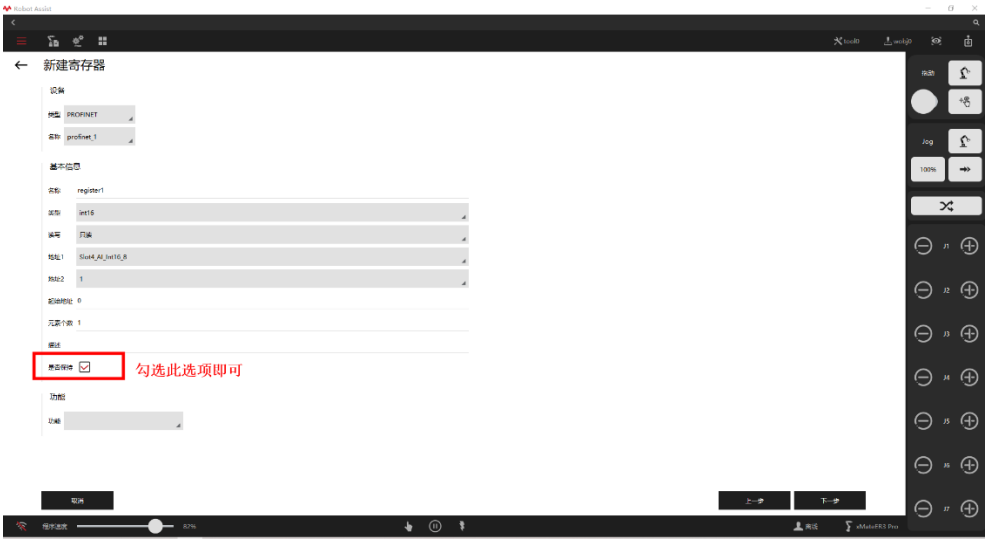
2）赋值式

直接使用操作符"="，例如"mtcp_wo_i[1] = 1"是将寄存器 mtcp_wo_i 的第一个元素的值更新为 1。同理，"a = mtcp_wo_i[1]" 是将寄存器 mtcp_wo_i 的第一个元素的值更新到 RL 程序的变量 a 中。

寄存器保持配置

寄存器保持：创建一个具有保持属性的寄存器 a，该寄存器 a 的当前值在机器人重启、关机或者断电、RL 停止时会保持在非易失的存储介质上。当机器人重新上电或者 RL 重新运行时，寄存器 a 的值就会恢复成关机、RL 停止运行前的保持值。

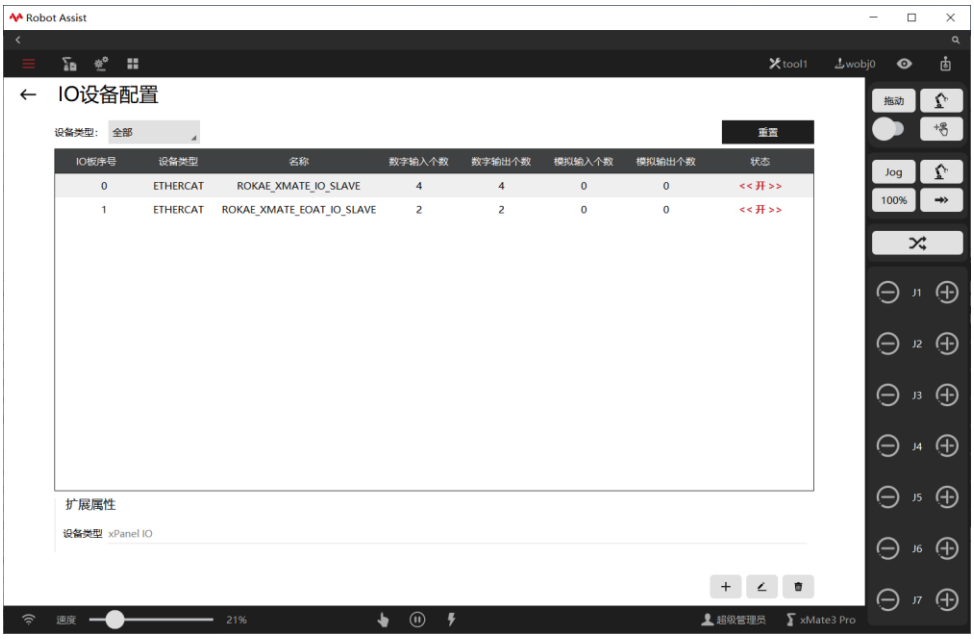
寄存器保持配置界面如下：



7.3.5 IO 设备

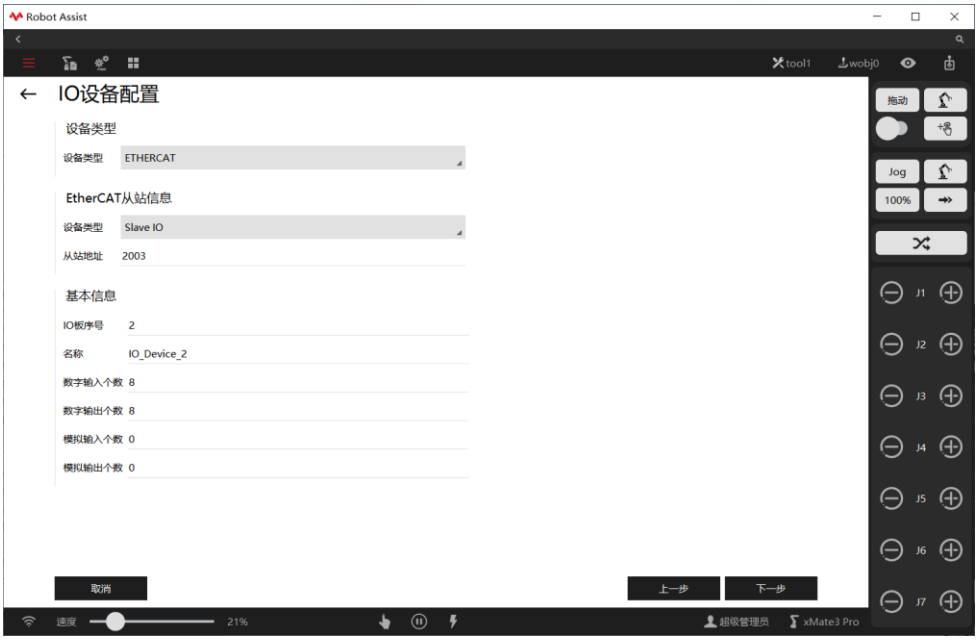
说明

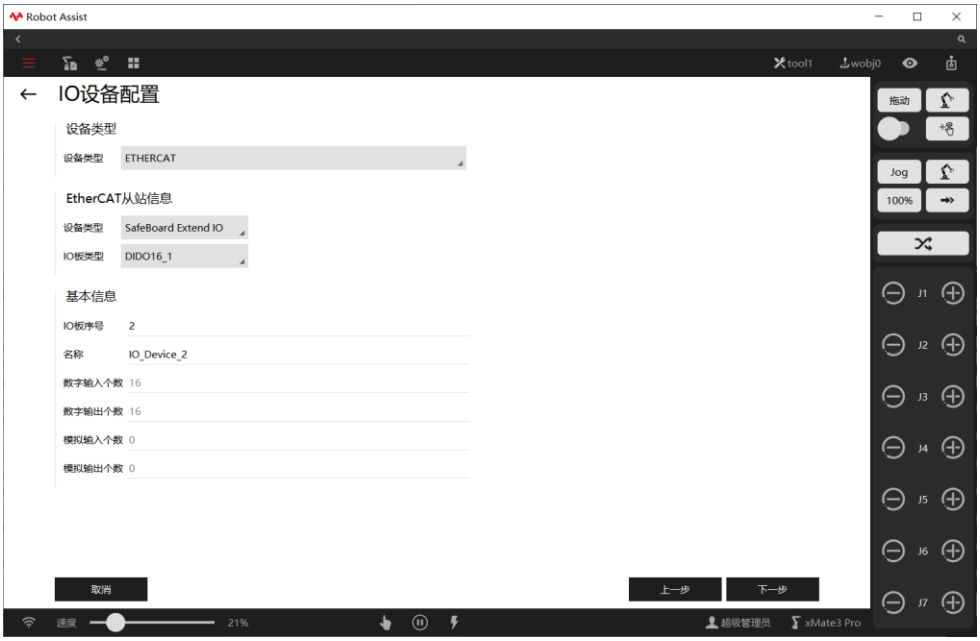
IO 包括 DI、DO、AI、AO。信号来源有控制柜自带、EtherCAT 扩展、现场总线扩展三种。工业机器人控制柜上自带若干 DI 和 DO。协作机器人底座和末端自带若干 DI 和 DO。工业机器人控制柜上预留了 EtherCAT 扩展接口，可以外接扩展 EtherCAT 模块，产生新的 DI、DO、AI、AO。Modbus 总线扩展也可以配置 IO。

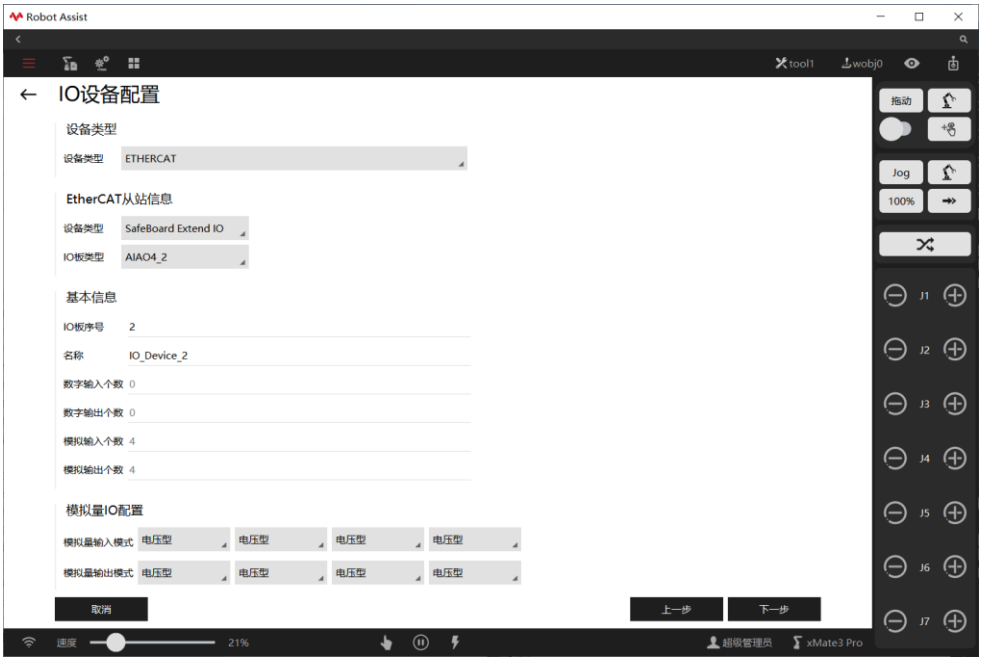


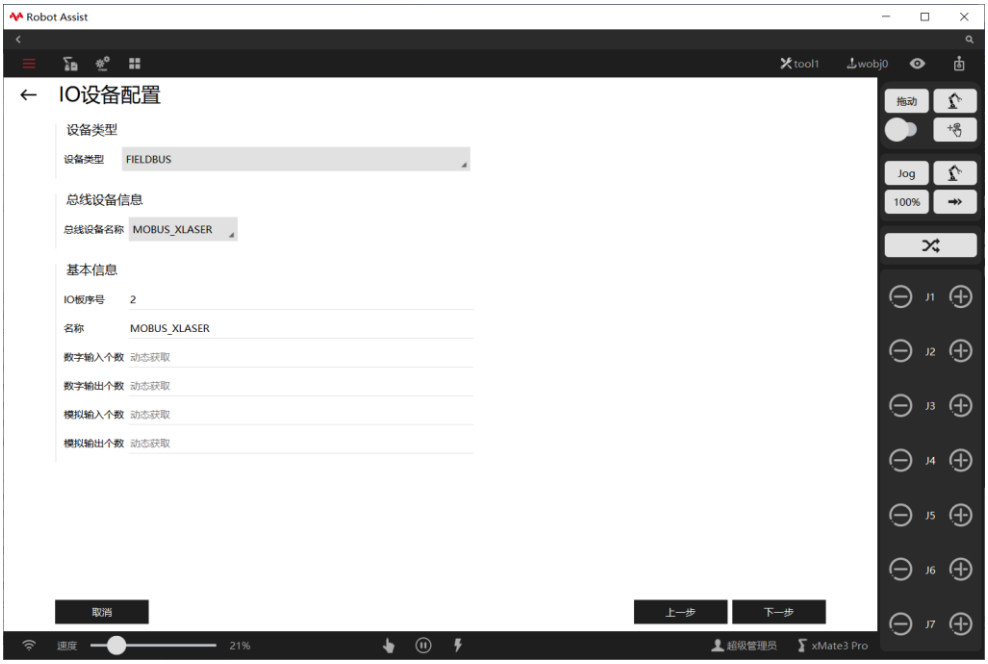
参数

在 IO 设备配置页面，点击右下角  ，进入 IO 设备配置页面，根据设备类型不同，页面参数有所区别。







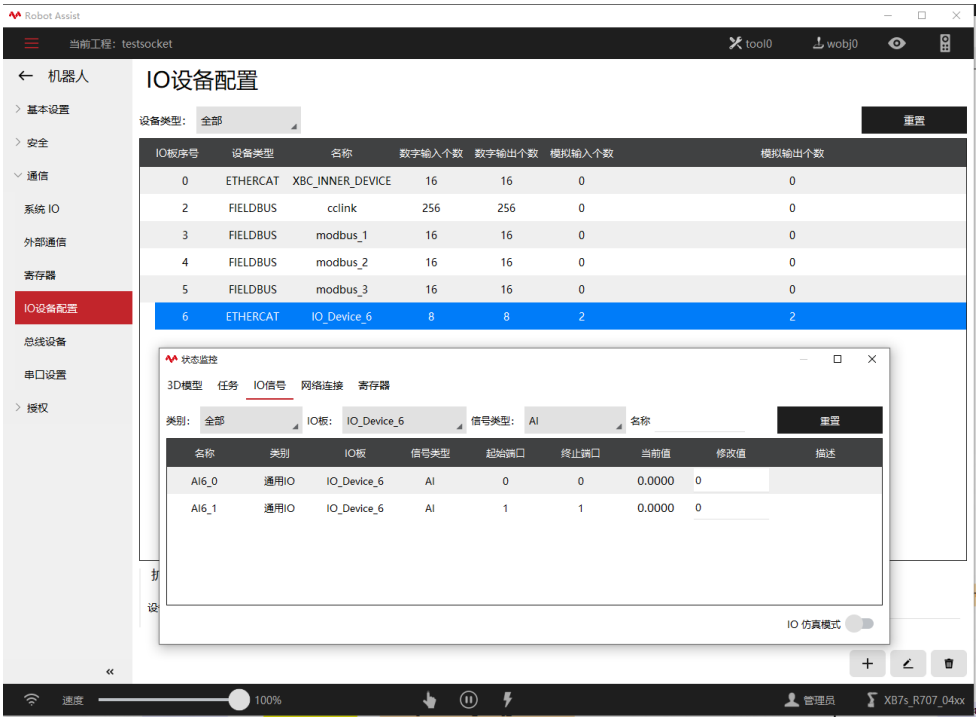


参数	说明
设备类型	可选 EtherCAT 和 FIELDBUS。EtherCAT 表示使用 EtherCAT 总线结合扩展模块来实现 IO 扩展。此时仅支持扩展模块作为从站使用，需要配置从站地址。
EtherCAT 从站信息-设备类型	可选 SafeBoard IO、SafeBoard Extend IO、xPanel IO、Slave IO。SafeBoard IO 表示机器人安全板上自带的 DI 和 DO。SafeBoard Extend IO 表示机器人安全板上的扩展 IO，一般是指 XBC_5 控制柜安全板的扩展 IO。xPanel IO 仅适用于协作机器人，表示末端的的 DI 和 DO。Slave IO 表示 EtherCAT 扩展模块。
EtherCAT 从站信息-IO 板类型	当 EtherCAT 从站信息-设备类型选择为 SafeBoard Extend IO 时，会出现 IO 板类型选项，提供可供选择的安全板扩展 IO 板，可选 DIO16_1、DIO16_4、DIO16_5、DIO16_6、AIAO4_2、AIAO4_3，选项的最后一位编号代表安全板扩展 IO 板的地址。根据实际接入的安全板扩展 IO 选择即可。除了用户手动编辑外，控制器可以自动识别并添加安全板扩展 IO。
EtherCAT 从站信息-从站地址	扩展模块在 EtherCAT 总线拓扑中的从站地址。不可以与安全板、关节、协作机器人末端的地址冲突。
FIELDBUS-总线设备名称	“总线设备”页面中，创建总线连接时用户自定义的名称，用来和“总线设备”进行关联。
IO 板序号	每一个 IO 设备配置都将生成一块虚拟的 IO 板，控制系统内部对 IO 板进行分类和管理。IO 板序号是进行虚拟 IO 板管理的唯一编号。
名称	虚拟 IO 板的自定义名称。改名称会在“状态监控”->“IO 信号”中用于筛选条件。
数字输入个数	DI 数量。
数字输出个数	DO 数量。
模拟输入个数	AI 数量。
模拟输出个数	AO 数量。
模拟量 IO 配置	当 EtherCAT 从站信息-设备类型选择为 SafeBoard Extend IO 时， IO 板类型选择为 AIAO4_2 或 AIAO4_3，会出现模拟量 IO 配置选项，可将每个模拟量通道配置为电压型或电流型。

状态监控

“状态监控”->“IO 信号”中，可以对创建的 DI、DO、AI、AO 进行观察。可以通过“虚拟 IO 板名

称”对 IO 信号进行过滤，此时仅显示当前虚拟 IO 板所配置的 IO 信号。可以通过信号类型进行过滤，只显示 DI、DO、AI、AO 中某一类型的信号。

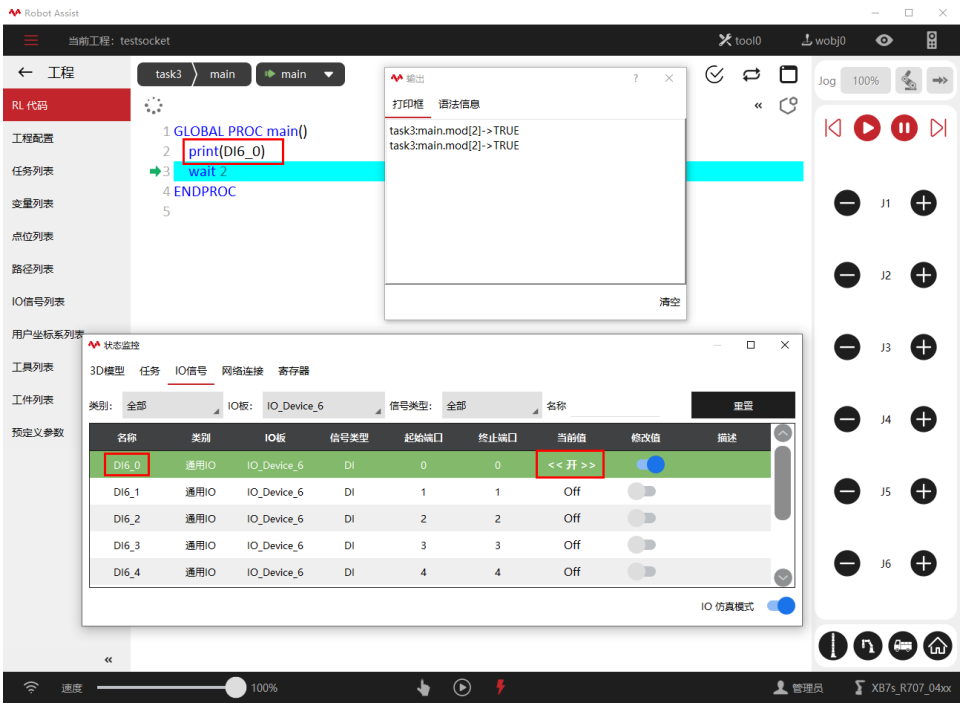


使用方式

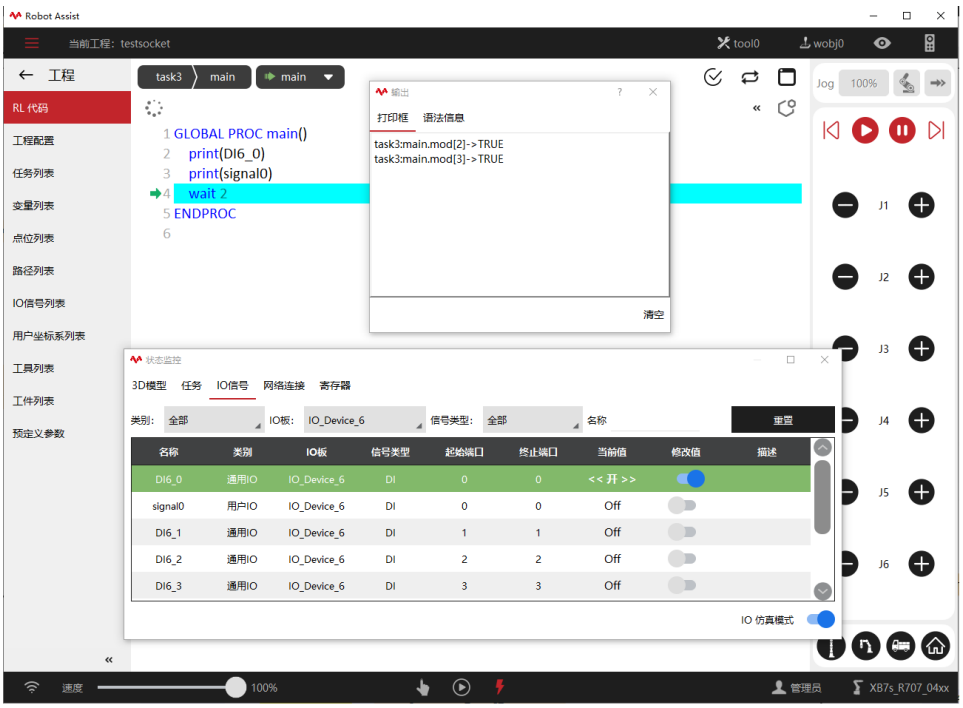
虚拟 IO 板配置好，IO 信号会生成默认的名称。此默认名称可以直接在 RL 程序中使用，也可以在“工程”->”IO 信号”中对创建的 IO 信号进行绑定和起名，然后再 RL 中使用新的名称。

直接使用默认值

6 号 IO 板默认生成了 DI6_X 的 IO 信号，如图在 RL 程序中直接使用 DI6_0 进行处理



绑定到 RL 工程中起别名



7.3.5.1 寄存器远程控制

说明

远程控制是一项组合功能，由 7 种不同功能的寄存器配合使用，按照特定的时序可完成复杂业务逻辑交互。外部设备可以通过远程控制功能实现机器人 Jog、更新点位、获取机器人位置、状态等功能。

寄存器功能

外部设备对机器人进行控制时，需要使用 4 种寄存器，对机器人而言属于只读寄存器。

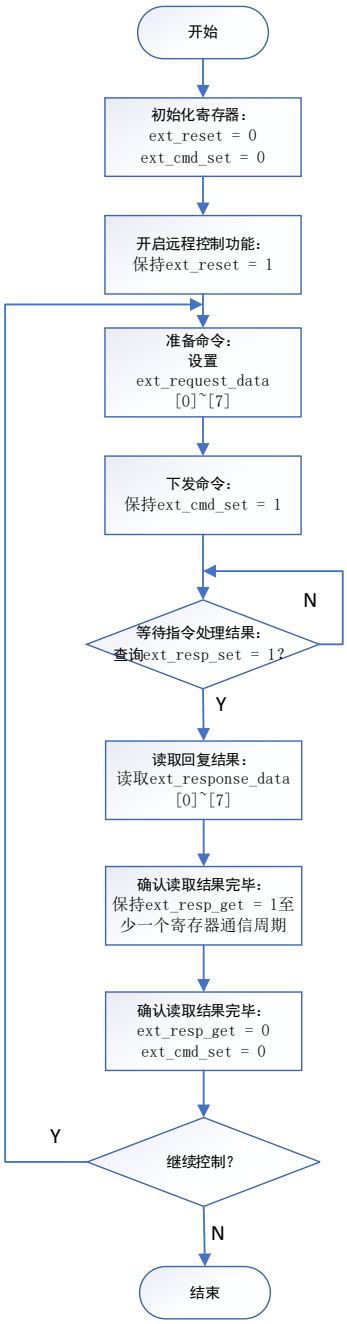
序号	功能码名称	属性	类型	长度	对应功能
1	ext_cmd_set	只读	int16, bool, bit	1	下发指令 1、命令执行时通过将请求信号设定为 1 进行请求，当命令为 1 时才响应请求。 2、为了防止误动作，请务必将命令数据设定到数据区域后再执行。（数据暂时放到缓存区，只有该位变为 1 才响应数据）。 3、命令执行完毕后，将该位进行清除(置为 0)。
2	ext_reset	只读	int16, bool, bit	1	功能重置： 1、用于开启远程控制功能的信号。功能使用过程中，请始终保持该寄存器为 1 状态。 2、0 状态下功能处于停止状态。 3、接口功能发生异常进行重置或中断动作命令时也使用该信号
3	ext_resp_get	只读	int16, bool, bit	1	确认并清除上一条命令的响应，重置 ext_resp_set 信号为 0。
4	ext_request_data	只读	int16	8	指令功能码。数组，固定长度 8 寄存器。具体功能详见功能码部分介绍

外部设备希望获取机器人的状态时，需要使用 3 种寄存器，对机器人而言是只写寄存器。

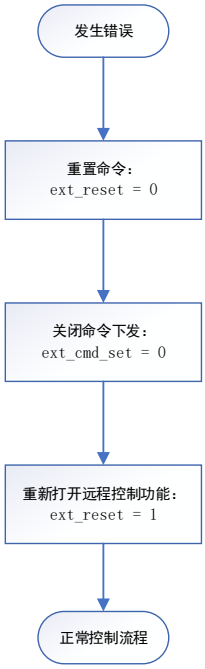
序号	功能码名称	属性	类型	长度	对应功能
1	ext_error_code	只写	int16	1	远程控制功能：错误码
2	ext_resp_set	只写	int16, bool, bit	1	机器人响应控制指令后，会设定该寄存器为 1，表示指令执行完成。
3	ext_response_data	只写	int16	8	远程控制功能：需要反馈的数据。数组，固定长度 8 寄存器。

使用流程

7 种寄存器使用配合及流程控制如下图所示。



正常控制流程



错误处理流程

指令格式

指令和响应分别使用 8 个寄存器实现。

命令信号 ext_request_data (占用八个寄存器，reg0 — reg7)，用于指定命令和相关参数的数据信号区域。命令由多字构成：

(1) 字：一个寄存器（16 位） 构成。

(2) 命令形式：命令最多由 8 个字构成。可根据使用的命令变化。最小的命令为 1 个字。

命令编号	命令编号 1	命令编号 2	.....	命令编号 7

响应信号 ext_response_data (占用八个寄存器，reg0 — reg7))，用于获取响应的数据信号区域。响应由多字构成：

(1) 字：一个寄存器（16 位） 构成。

(2) 响应形式：响应最多由 8 个字构成。可根据收到的命令变化。 最小的响应为 1 个字。但是，如果是异常响应则固定占用 3 个字。

命令编号	响应 1	响应 2	.....	响应 7

具体可用的命令编号如下表：

命令类别	说明	命令码	命令长度	
			命令	响应
JOG	设置 jog 空间	1	2	3
	获取 jog 空间	2	1	4
	设置 jog 速度	3	2	3
	获取 jog 速度	4	1	4
	设置 jog 步长	5	2	3
	获取 jog 步长	6	1	4
	开始 jog	7	4	2
	停止 jog（不带参数）	8	1	2
	更新点位	9	2	2
	运动到点	10	2	2
设置信息	设置工具	11	2	3
	获取当前工具 id	12	1	4
	设置工件	13	2	3
	获取当前工件 id	14	1	4

指令详解

1）设置 jog 空间：

命令/回复	命令码	参数 1	参数 2
设置 jog 空间	1	坐标系: 1: 轴空间 2: 世界坐标系 3: 法兰坐标系 4: 基坐标系 5: 工具坐标系 6: 工件坐标系	N/A
回复	1	设置结果：0 成功；1 失败。	错误码

2）获取 jog 空间：

命令	命令码	参数 1	参数 2	参数 3
获取 jog 空间	2	N/A	N/A	N/A
回复	2	设置结果: 0 成功；1 失败	错误码	坐标系: 1: 轴空间 2: 世界坐标系 3: 法兰坐标系 4: 基坐标系 5: 工具坐标系

				6: 工件坐标系

3）设置 jog 速度：

命令/回复	命令码	参数 1	参数 2
设置 jog 速度	3	Jog 速度（1~100）	N/A
回复	3	设置结果：0 成功；1 失败	错误码

4）获取 jog 速度：

命令	命令码	参数 1	参数 2	参数 3
获取 jog 速度	4	N/A	N/A	N/A
回复	4	设置结果： 0 成功；1 失败	错误码	Jog 速度（1~100）

5）设置 jog 步长：

命令/回复	命令码	参数 1	参数 2
设置 jog 步长	5	1: 连续 2: 10mm 步长 3: 1mm 步长 4: 0.1mm 步长 5: 0.01mm 步长	N/A
回复	5	设置结果：0 成功；1 失败	错误码

6）获取 jog 步长：

命令	命令码	参数 1	参数 2	参数 3
获取 jog 步长	6	N/A	N/A	N/A
回复	6	设置结果： 0 成功；1 失败	错误码	1: 连续 2: 10mm 步长 3: 1mm 步长 4: 0.1mm 步长 5: 0.01mm 步长

7）开始 jog：

该指令依赖命令码 1：设置 jog 空间。如果是轴空间，参数 1 的值表示的是关节编号（J1~J7，1 代表 J1，7 代表 J7）；如果是笛卡尔空间，参数 1 的值表示的是（x,y,z,a,b,c,elb）的编号（1 表示 x，7 表示 elb）。

命令/回复	命令码	参数 1	参数 2
开始 jog	7	运行方式： 轴空间表示关节编号； 笛卡尔空间（x,y,z,a,b,c,elb）	Jog 方向： 1：负方向 2：正方向
回复	7	设置结果：0 成功；1 失败	错误码

8）停止 jog：

命令/回复	命令码	参数 1
停止 jog	8	N/A
回复	8	设置结果：0 成功；1 失败。

9）更新点位：

命令/回复	命令码	参数 1	参数 2

更新点位	9	RL 工程点位列表中的编号	N/A
回复	9	设置结果：0 成功；1 失败	错误码

10）运动到点位：

命令/回复	命令码	参数 1	参数 2
运动到点位	10	运动方式： 1：MoveAbsj；2：MoveJ；3：MoveL	RL 工程点位列表中的编号
回复	10	设置结果：0 成功；1 失败	错误码

11）设置当前工具：

命令/回复	命令码	参数 1	参数 2
设置当前工具	11	RL 工程工具列表中的编号	N/A
回复	11	设置结果：0 成功；1 失败	错误码

12）获取当前工具的编号：

命令/回复	命令码	参数 1	参数 2	参数 3
获取当前工具的编号	12	N/A	N/A	N/A
回复	12	设置结果：0 成功；1 失败	错误码	当前工具编号

13）设置当前工件：

命令/回复	命令码	参数 1	参数 2
设置当前工件	13	RL 工程工件列表中的编号	N/A
回复	13	设置结果：0 成功；1 失败	错误码

14）获取当前工件的编号：

命令/回复	命令码	参数 1	参数 2	参数 3
获取当前工件的编号	14	N/A	N/A	N/A
回复	14	设置结果：0 成功；1 失败	错误码	当前工件编号

错误码

在进行命令设置时，如参数错误、机器人状态不匹配或者其他情况下，可能发生设置失败。此时可以通过错误码观察机器人出现问题。

控制系统有三种错误码：

1） 指令执行结果的错误码(ext_response_data 中)（命令执行结果）。

2） ext_error_code 错误码(命令无法执行，比如机器人忙碌，远程控制标志位不正确等)。

3） sta_error_code，机器人错误码，（jog 运动过程中出现的停止报错，可以读这个寄存器）。

正常情况下用户需要按照流程来使用错误码：

1） 发完执行指令之后（ext_cmd_set=1），先读 ext_error_code, 如果没有错误码，那么再读 ext_repose_data 的返回值，如果返回值不为零，那么就读 ext_repose_data 的错误码。

2） 对于运动操作（jog 和运动到点位），如果上面返回值都成功了，还应该读一下 sta_error_code ,看运动中是否有错导致的停止（比如奇异点和超限位）。

ext_error_code 含义：

错误码	含义	备注
01	不支持的命令	
02	参数不合法	
03	控制标志不正确	需要确认：ExtRespSet 为 0，ExtRespSet 为 1
04	机器人忙碌	机器人正在执行指令中，禁止响应别的指令
05	找不到对应的编号	工具、点位，工件的 id
06	点位类型和运动类型不匹配	针对运动至到某个点的指令，点位类型和运动类型不匹配，比如关节空间只能用 MoveAbsJ 指令，笛卡尔类型点只能用 MoveJ 或 MoveL
07	输入的轴数和机型不匹配	
11	手自动模式不对	
12	机器人状态不对，请假查机器人是否是 jog 状态	机器人处于 jog 状态时可以被 jog，若处于拖动模式等非 jog 状态，则不允许被 jog。
13	上电状态不对	机器人只有在上电之后才能 Jog。
14	机器人处于非位置模式，不允许 jog	与 12 类似。
15	启动 jog 失败，算法报错	由于诸多原因导致无法 jog 时会报该错误。
20	遇到奇异点	
21	已经运动到目标点	若已经运动至某点再次运动至该点时，会报该错误。

7.3.5.2 Modbus 拓展 IO

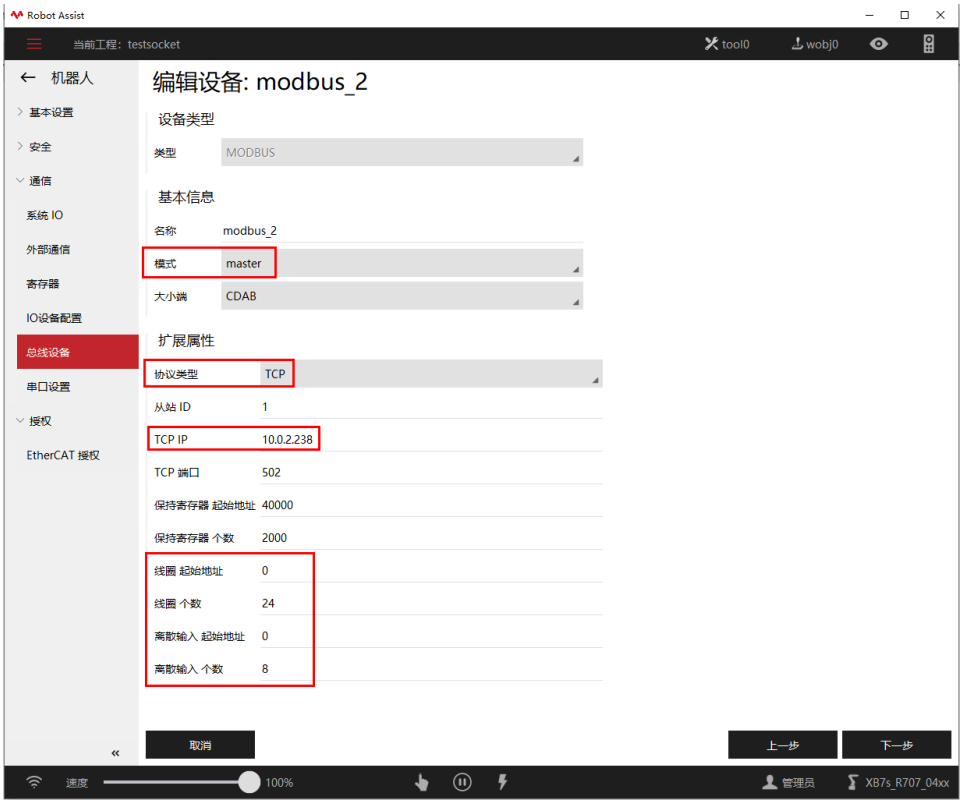
说明

当需要使用真实的 IO 电信号来与外部设备进行交互时，推荐使用转接模块。将转换模块连接到控制柜上，通过控制系统支持的现场总线方式对转换模块进行操作，建议选择珞石推荐的 Modbus IO 模块。该模块为 Modbus TCP 从站，以线圈方式对其进行控制，因此机器人需要配置为 Modbus Master，并开启线圈功能。

参数配置

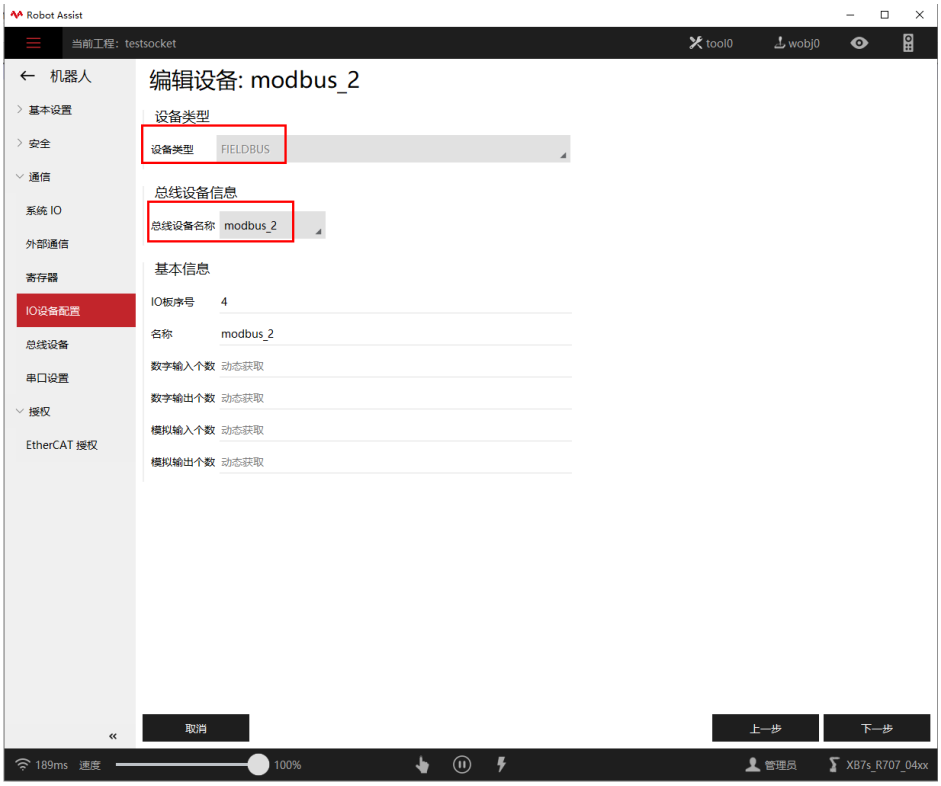
按照现场总线和扩展 IO 的配置方式，使用 Modbus 扩展模块扩展真实 IO 时，需要先配置“总线设备”，然后配置“IO 设备配置”。

1）配置“总线设备”



参数	值/说明
主从模式	选择“主站”，机器人作为主站。
IP	机器人作为主站时，IP 填写 Modbus IO 模块的 IP。
端口号	机器人作为主站时，端口号填写 Modbus IO 模块的端口号。
Slave ID	Modbus IO 模块从站 ID
线圈状态	Modbus 模块“线圈状态”寄存器。范围：1~32，数据类型：int。
输入状态	Modbus 模块“输入状态”寄存器。范围：1~32，数据类型：int。
连接	参数配置完成后，点击“连接”，完成参数配置。

2）配置“IO 设备配置”

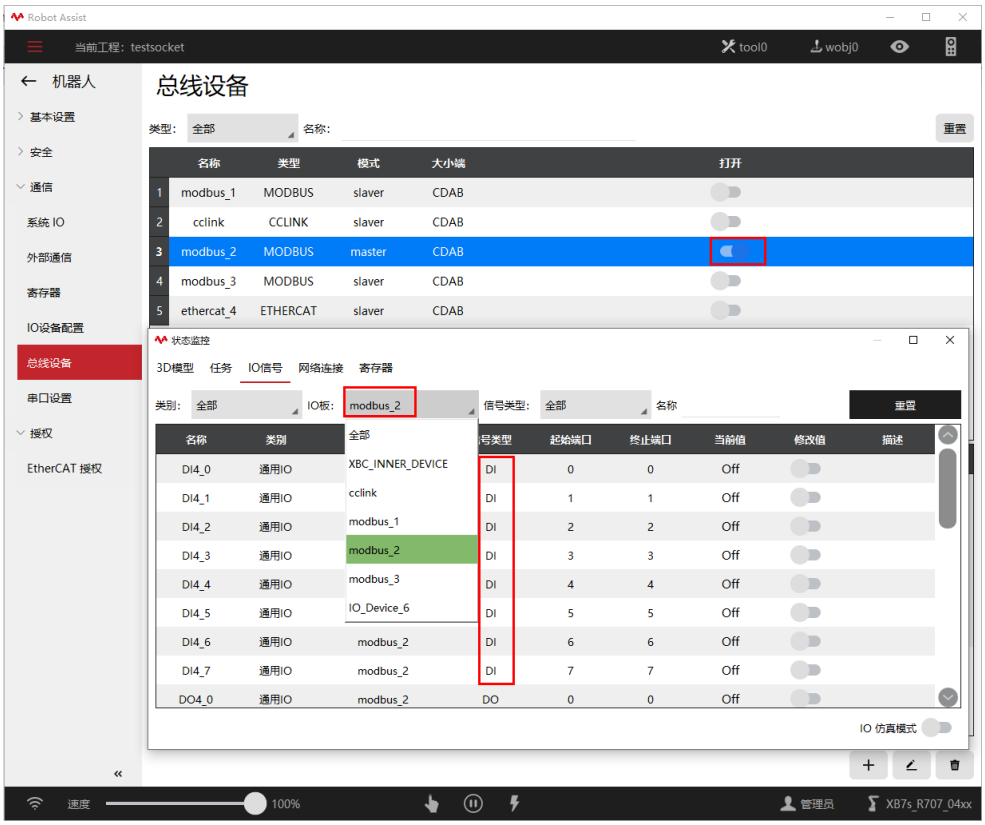


参数	值/说明
设备类型	选择“FIELDBUS”。
总线设备名称	选择刚才进行“总线设备”配置时，自定义的名称。
基本信息	数字输入/输出、模拟输入/输出，会在总线设备配置页面配置好，因此采用自动获取，无需配置。

3）开启功能和状态监控

注意刚配置好“总线设备”和“IO 设备配置”时，状态监控中是不会展示配置好的扩展 IO 的，因为此时还未开启总线连接。如果要使用这些 IO，必须开启总线连接，使控制柜与扩展 IO 模块正确建立连接和保持通信。

如图，开启总线设备后，控制器与扩展模块建立连接，正常工作，此时 IO 信号的状态监控中将 可以看到 modbus_2 这个总线设备的 IO，并且其配置的 DI、DO 数量与总线设备中配置的一致。



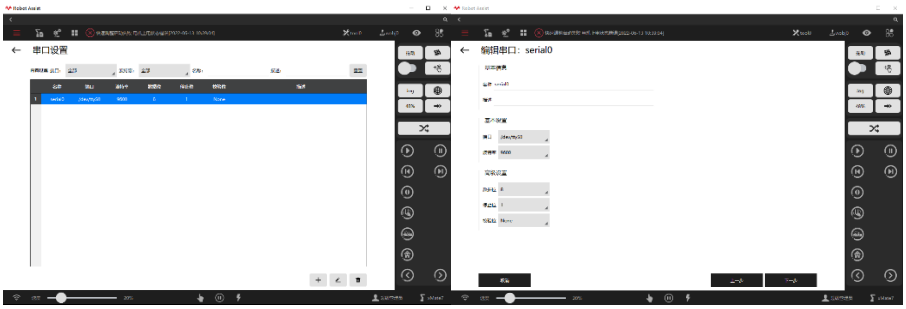
7.3.6 串口通信

说明

用户可以使用串口的方式和外部设备进行通信。使用串口需要硬件设备支持，工业机器人如果选用 XBC5 控制柜，则柜体上预留了一路 RS-232 串口。另一个方案则是借用控制柜预留的 USB 接口，使用 USB 转 RS-232 的接口模块来进行串口通信。协作机器人对外暂未预留硬件接口，因此暂时不支持该功能。

配置

串口使用前，需要配置的参数有：串口名称（RL 中使用）、串口端口、波特率、数据位、停止位、校验位。配置界面在“机器人配置”->“通信”->“串口设置”中，如图所示。请尽量保证串口通信两端的参数设置一致。否则可能会导致收发数据异常。



参数	说明
名称	用户自定义名称，在 RL 中操作该串口资源使用的唯一标识符。需要注意这个名称受工程中名称冲突限制，不可以和工程中已有的网络标识符同名，不可以和已有的其他串口标识符同名。
端口	系统端口。控制系统会将扫描到的串口资源（包括 USB 形式转换的串口）列举出来，供用户选择使用。此项就是操作系统识别到的串口资源的名称。
波特率	可选 1200， 2400， 4800， 9600， 19200， 38400， 57600， 115200。
数据位	可选 5， 6， 7， 8 位。
停止位	可选 1， 1.5， 2 位。
校验位	可选奇校验（Odd），偶校验（Even），全 1 校验（Mark），全 0 校验（Space），无校验（None）

使用串口

配置完串口以后，无需重启即可在 RL 程序中使用串口。串口功能包含一系列指令，详见 RL 指令中关于串口指令的详细说明。

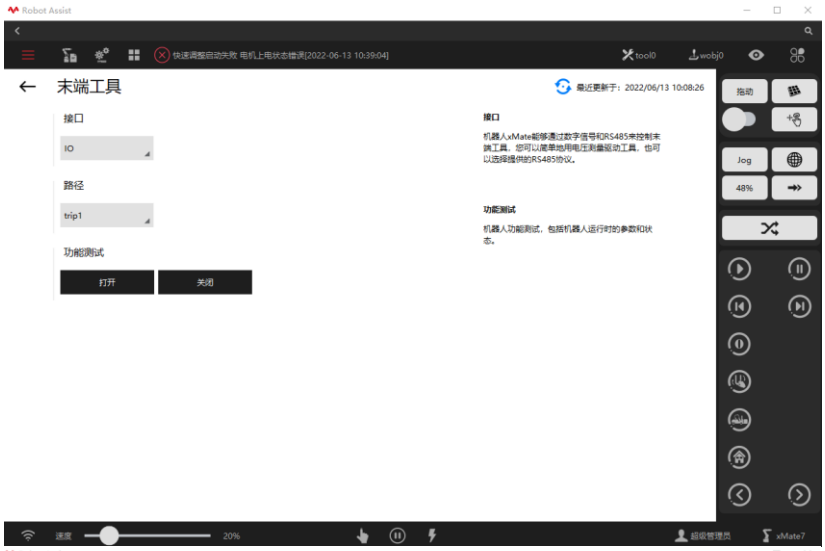
7.3.7 末端工具通信

说明

xMate 支持对大寰爪手的开合控制，末端工具接口支持 IO 通讯和 RS485 通讯。

该功能仅适用于协作机器人，包括 xMate ER 系列、xMate CR 系列。

参数设置



参数设置	值/说明
接口	通讯协议，可选 IO 或者 RS485 控制。
路径	爪手包括两组行程属性 trip1 和 trip2，包含张开闭合位置和张开闭合力。
最大位置	爪手张开的最大位置，单位是百分比
最小位置	爪手张开的最小位置，单位是百分比。
外撑力	爪手张开时使用的力，单位是百分比。
夹持力	爪手闭合时使用的力，单位是百分比。

提示

RS485 支持对夹爪 trip 参数设置，IO 控制时，其 trip 参数只能通过大寰通讯转接盒设置。

7.3.8 RCI 配置

说明

RCI 是外部控制接口，使用前需要进行 RCI 通信设置。

参数设置

使能开关打开之前填写用户 PC 的 IP 地址，如果用户 PC 与机器人通过网线直连，那么用户 PC 的 IP 地址应该与机器人的 IP 处于同一网段；如果用户 PC 与机器人通过无线或者交换机的方式连接，那用户 PC 应该与机器人处于同一局域网内。端口号默认设置为 1337。

包丢失阈值为百分比，代表 RCI 通信过程中的丢包率，例如，包丢失阈值设置为 10，代表 RCI 使用过程中的丢包率不得超过 10%。建议丢包阈值设置范围在 10~20 之间。

开启 RCI

通信设置完成后，打开使能开关，按下保存按钮，即打开 RCI 功能。

RCI 使用结束后，关闭使能开关，按下保存按钮，即关闭 RCI 功能。

详细的 RCI 使用方法和例程参考《RCI 用户使用手册》。

7.4 工艺包

7.4.1 激光焊接

参考手册《激光焊接工艺包使用手册》。

7.4.2 电镀线跟踪

参考手册《电镀线跟踪功能操作手册》。

7.5 授权

7.5.1 EtherCAT 授权

说明

通过授权码对 EtherCAT 通信授权。

> 提示
>
> 当授权码失效或者授权失败，将无法使能让机器人上电。

8 菜单模块

8.1 诊断

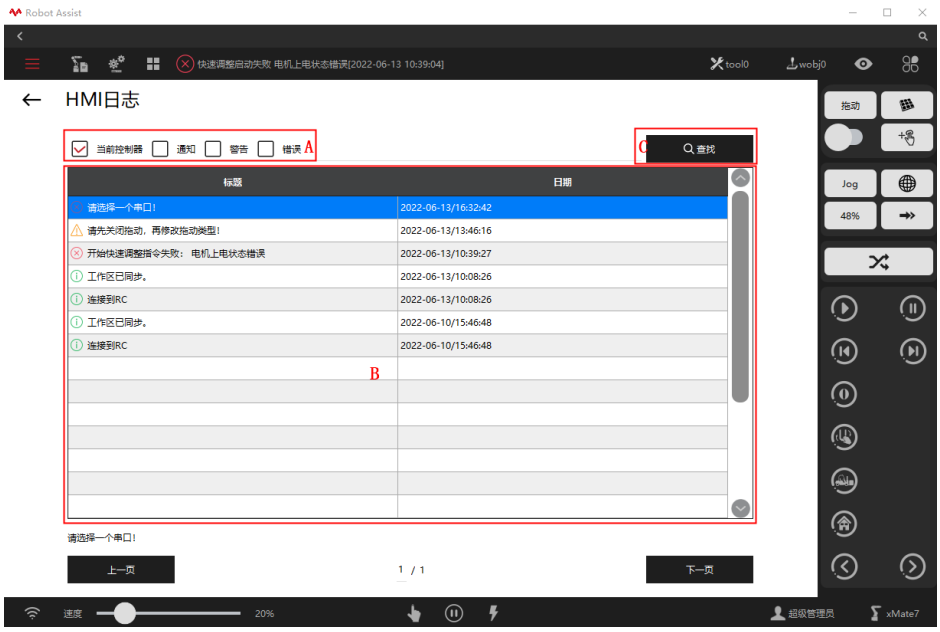
说明

xCore 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。

8.1.1 示教器日志

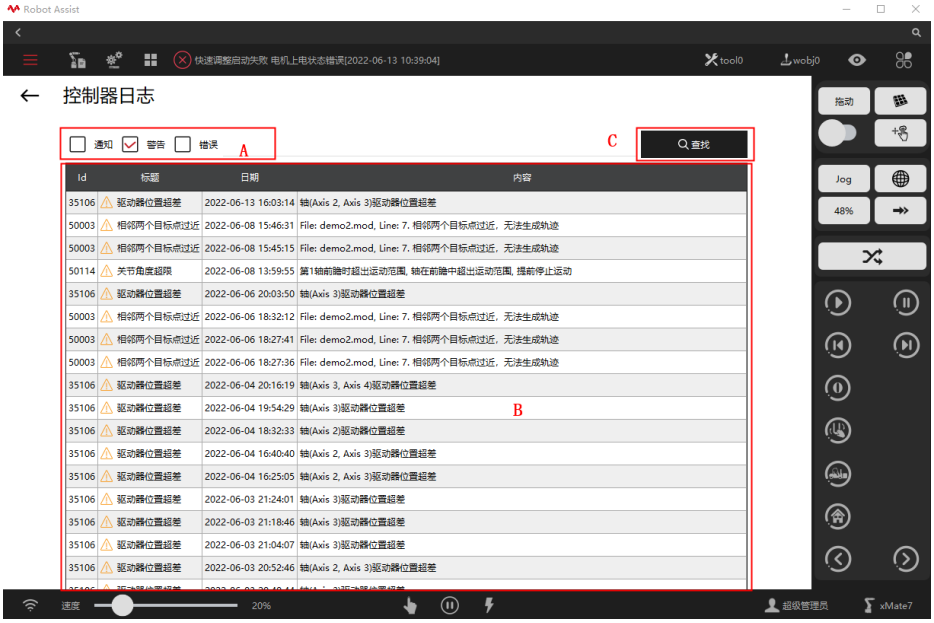
说明



A	筛选条件区，可选择只查看当前控制器或者与 HMI 连接过的控制器日志；可选择日志级别进行筛选
B	日志显示页，显示日志标题和发生时间，使用翻页按钮切换上下页
C	查找区，可根据关键字对日志进行搜索

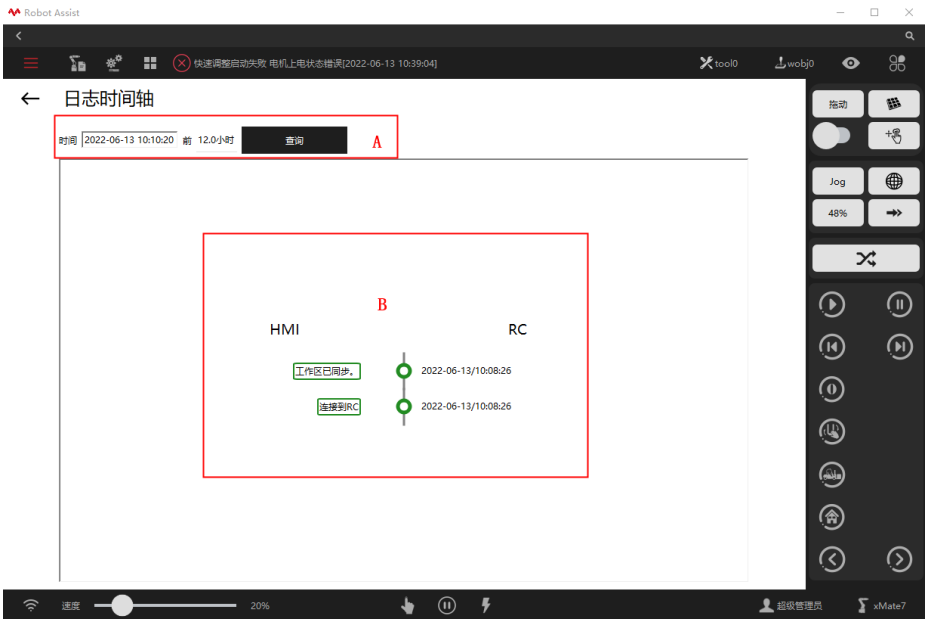
8.1.2 控制器日志

说明



A	筛选条件区，可选择日志级别进行筛选
B	日志显示页，显示日志编号、标题、发生时间以及内容等基本信息。使用翻页按钮切换上下页
C	查找区，可根据关键字对日志进行搜索

8.1.3 日志时间轴

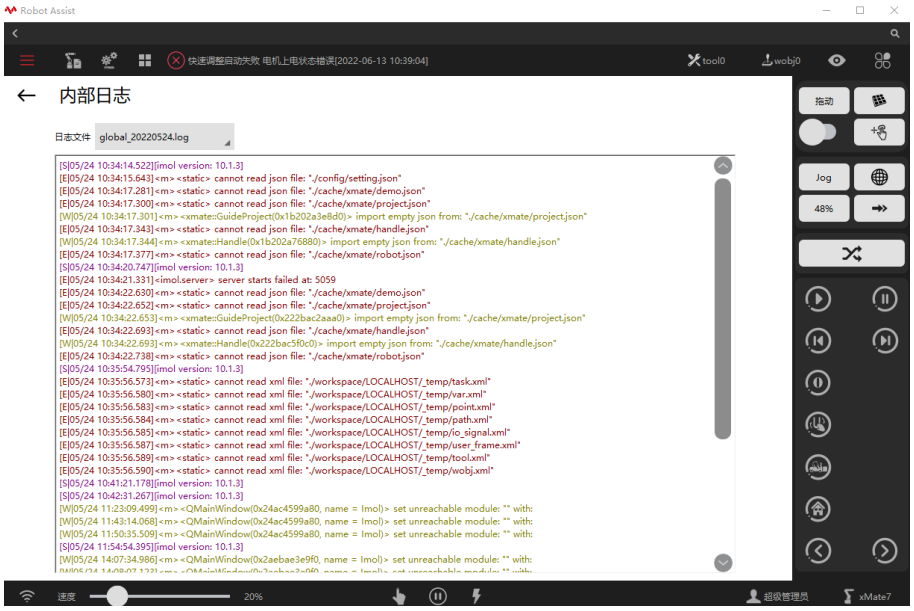


A	查找区，可设置时间区间对日志进行搜索
B	日志显示页，左侧为 HMI 日志，右侧为 RC 日志

8.1.4 内部日志

说明

客户现场遇到问题时，技术支持可根据日志文件判断出问题原因。



8.1.5 高级选项

说明

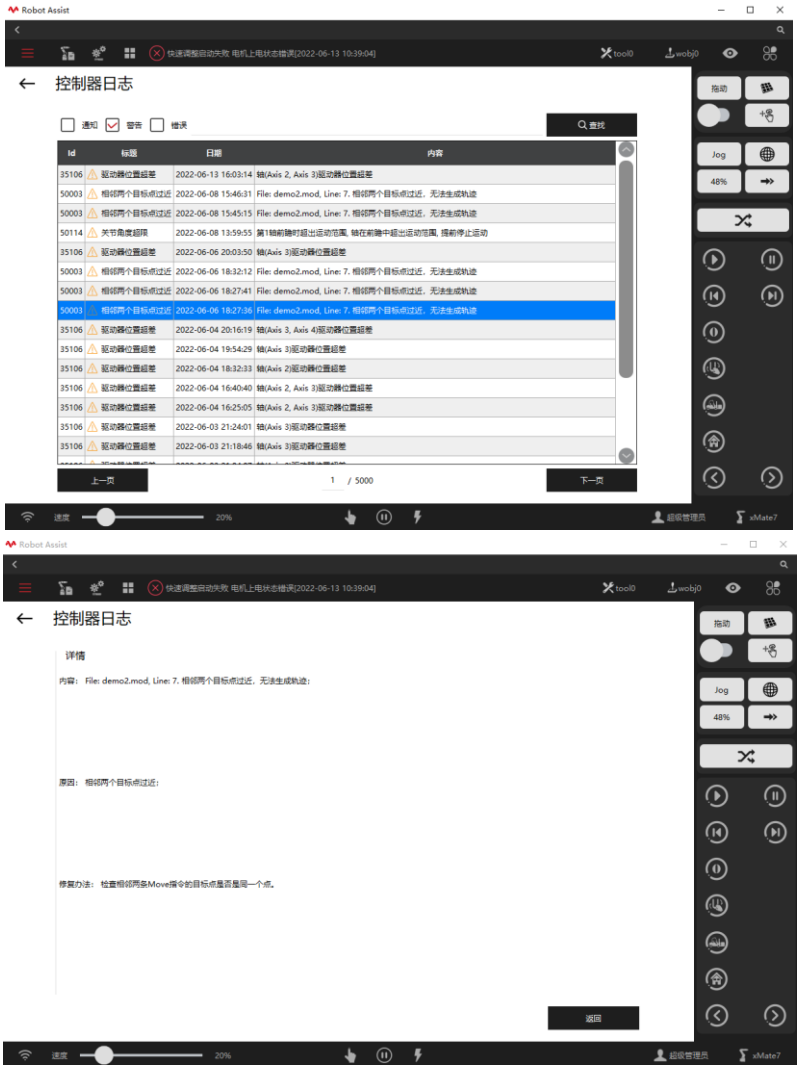
此界面功能用于辅助开发人员进行诊断伺服、ECAT 等设备问题，开启实时线程告警监测等功能。由于开启诊断功能后会加重控制器运行负荷，所以在实际生产环境中非必要不要打开。

伺服诊断	伺服诊断模块用来保存伺服出现错误的数据。点击保存按钮，5s 后可以导出诊断数据。
EC 诊断	EC 软件诊断功能可以辅助排查 ECAT 设备问题。
超时警告	开启后可以上送实时线程超时警告。

8.1.6 错误恢复

说明

机器人报错时，用户可根据错误详情中提示的修复方法进行错误恢复，遇到无法恢复的错误时，联系厂商售后支持。



8.2 帮助

说明

软件升级界面提供了控制器升级，控制器备份，抹除所有配置以及恢复出厂设置相关的功能。

控制器升级：用于上传升级包及数据恢复的操作。文件支持加密文件和未加密文件的压缩包两种类型。上传升级文件成功后界面会提示“上传成功”，根据弹窗提示重启即可。

数据恢复操作：通过控制器升级选择要恢复的数据包，勾选要恢复的数据后点击上传，根据弹窗提示重启。

控制器备份：用于控制器中的数据备份。勾选需要备份的文件，点击“打开”选择备份路径后点击“导出”。导出的文件为加密文件。

重启机器人：用于重启工控机系统，执行该操作需要建立升级服务连接。

抹除所有配置：用于一键抹除机器人配置文件，自定义配置，以及用户的工程文件等。但是会保留控制系统相关运行日志。点击此按钮后，需要手动重启机器人。执行该操作需要建立升级服务连接。

恢复出厂配置：用于控制系统恢复到出厂默认状态，会重置控制系统的版本到默认出厂状态。控制系统相关配置文件，用户的工程文件会被重置。但是会保留控制系统相关运行日志。执行该操作需要建立升级服务连接。

当用户想升级到最新版本的控制器时，可联系我们申请新版控制器安装包以及 HMI 软件包。

- 将新版控制器安装包以及 HMI 软件包下载到本地。
- 打开新版本的 HMI，连接控制器，连接升级服务。
- 选中备份选项中的自定义配置、机器人配置、控制器日志、工程数据，选择想要备份的本地文件夹目录，导出备份。
- 请勿选中控制器升级选项中的自定义配置、机器人配置、控制器日志、工程数据演示案例、伺服。只需要选择下载到本地的控制器安装包，升级选项会根据安装包配置升级选项，点击上传即可。
- 上传成功后， HMI 会提示重启控制器，重启之后控制器升级成功。

> **提示**
>
> 1. 为了防止升级过程中数据丢失，我们建议升级之前先将控制器进行备份。
> 2. 软件升级过程中请勿选择自定义配置、机器人配置、控制器日志、工程数据、演示案例、伺服等选项，否则会将当前的机器人零点，动力学模型，工程数据等信息覆盖掉。
> 3. 点击抹除所有配置或恢复出厂设置按钮会重置机器人关键运行配置文件和删除用户自定义的工程数据等，操作之前请反复确认后再执行操作！

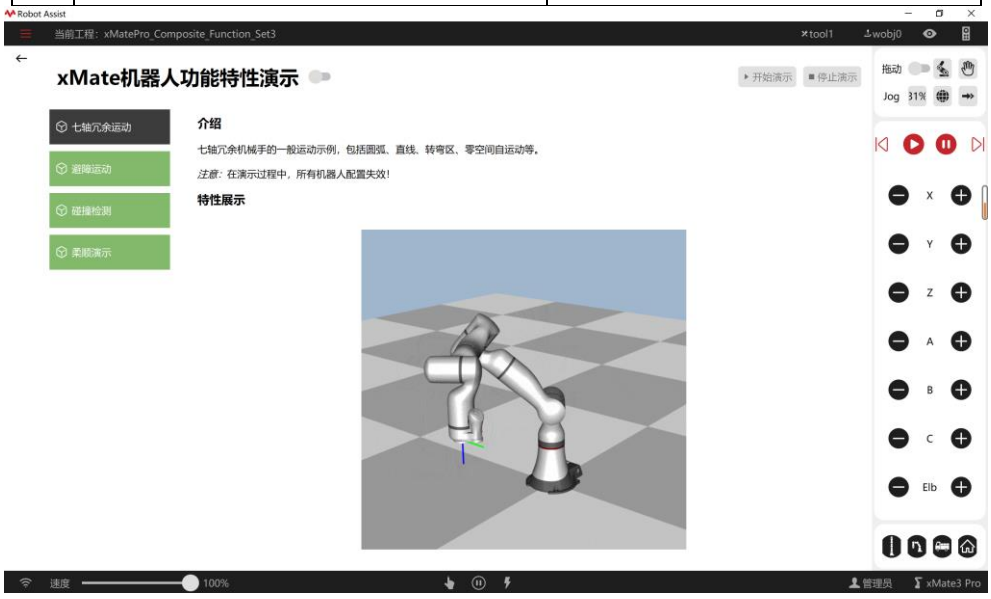
8.3 演示

5.3.1 七轴冗余运动

说明

七轴冗余机械手的一般运动示例，包括圆弧、直线、转弯曲、零空间自运动等。

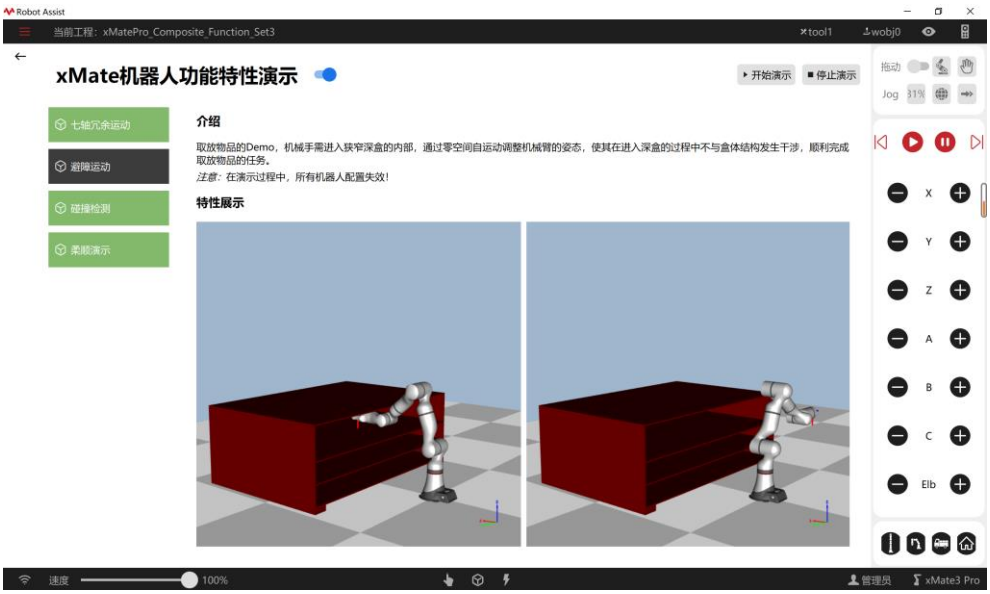
	操作	说明
1	使用 admin 级别的用户登录系统，并切换到演示界面。	
2	在左侧 Demo 列表栏，选择演示的特性功能。	
3	点击底部状态栏操作模式切换按钮，将机器人切换至自动模式。	
4	点击底部状态栏上电按钮上电。	
5	点击右上角的开始演示。	
6	Demo 演示完毕后，点击右上角的停止演示，再进行 Demo 切换。	演示过程中，需要调整 DEMO 阈值时，例如：碰撞检测灵敏度、柔顺演示刚度，先点击停止演示按钮，调整完阈值之后，再开始演示。



8.3.2 避障运动

说明

机械手进入窄深盒的内部，通过零空间自运动调整机械臂的姿态，使其在进入深盒的过程中不会与盒体结构发生干涉，顺利完成取放物品的任务。



8.3.3 碰撞检测

说明

碰撞检测演示支持单轴灵敏度设置和高中低三档灵敏度两种设置模式。

检测碰撞并停止后向下按压机器人，可从碰撞停止状态返回，继续运行。

8.3.4 柔顺演示

说明

柔顺演示 Demo 展示了 xMate 在笛卡尔空间的不同刚度的力控特性，该特性适用于磨削力较大的打磨、抛光、去毛刺等工艺。

柔顺演示 Demo 支持末端平移、旋转及臂角刚度单独设置和平移旋转组合两种设置方式。

警告

Demo 演示过程中，所有机器人配置失效，例如：

1、 机器人基坐标系默认与世界坐标系重合。

2、 机器人默认末端无负载，机器人末端安装负载时会影响碰撞检测和柔顺演示等 Demo 演示。

9 示教器选项

9.1 连接设置

说明

连接界面主要探测和连接机器人等操作。

搜索可用机器人：搜索在同一局域网中全部的机器人（直连情况除外），当连接机器人后会显示控制器服务及升级服务均连接。

自动重连：当机器人与 RobotAssist 软件间的网络断开后，RobotAssist 软件会尝试自动重连，超过设定的重连时间后将不再尝试连接。

如 RobotAssist 软件与机器人处于同一网络中仍搜不到机器人，或连接机器人后 **3D** 界面不显示机器人的实时位置，点击图中蓝色字体或进入"基本设置"界面，在绑定 IP 地址下拉框中选择局域网所分配的 IP，就可以解决上述两个问题。

9.2 基本设置

说明

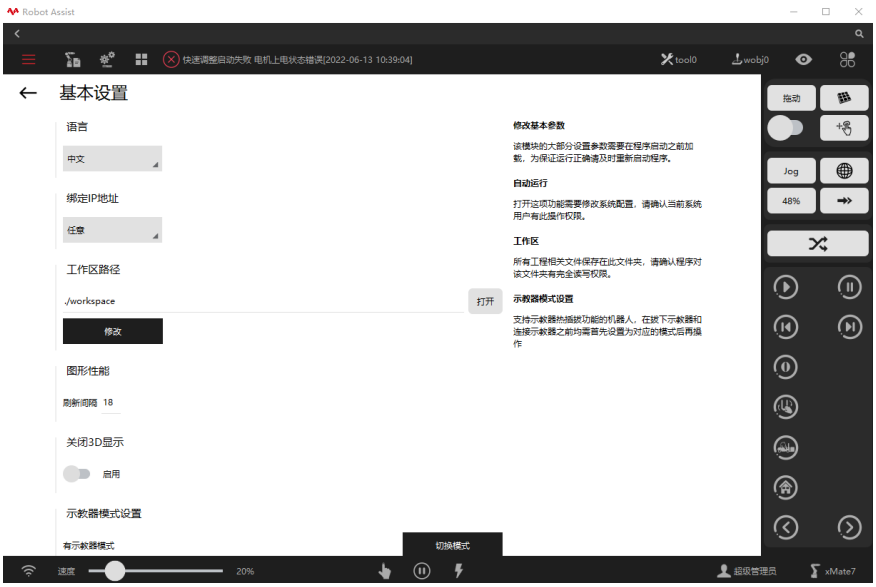
语言设置：支持中文和英文。

系统启动时自动运行：勾选后 RobotAssist 软件在开机时会自动运行

绑定 **IP** 地址：设置 RobotAssist 软件和机器人连接时使用哪个网卡

工作区路径：用于设置工程文件存储的位置

图形性能：用于设置 3D 模型刷新的时间间隔，单位为 ms，上限 100。



9.3 外观设置

说明

主题切换：默认风格和 win7 适配。两个风格只有字体不一样。

控件调节：设置界面按钮、图标及状态栏等的大小，点击应用后立即生效。

字体调节：用于设置 RobotAssist 软件界面上全部文字的大小，点击应用后立即生效。

9.4 文件管理器

说明

文件管理器界面主要用于快捷打开 RobotAssist 软件包中的文件夹。

缓存文件夹：用于存储 RobotAssist 软件的缓存。

日志文件夹：用于打开存储 RobotAssist 软件日志的文件夹，此处的日志与诊断界面的内部日志内容一致，可以点击此处进入文件夹进行日志的单独拷贝。

工作区文件夹：用于快捷打开机器人工程文件存储位置。

10 机器人运动基础

10.1 坐标系系统

说明

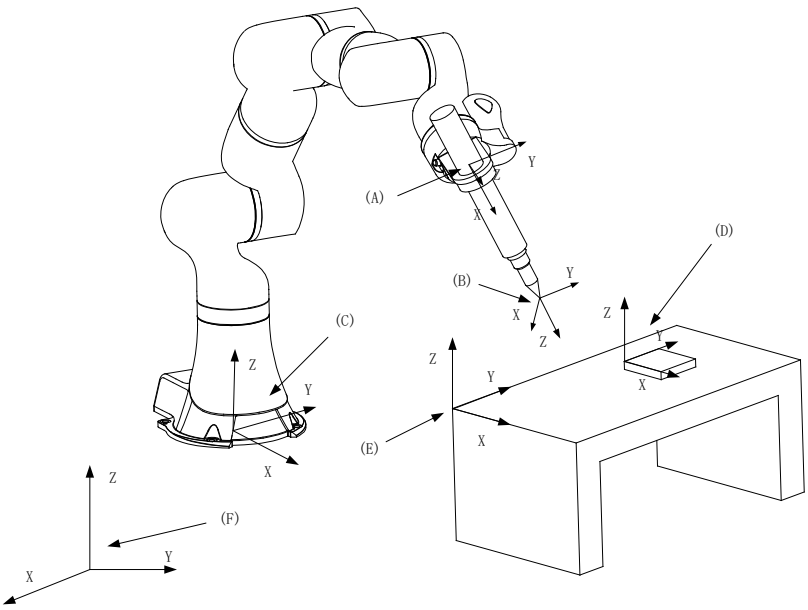
机器人的运动包含位置、速度、加速度等信息，这些信息必须要指定参考系才具有实际意义，因此在开始 Jog 之前我们需要了解机器人所使用的坐标系统。

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。

坐标系

xCore 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 xCore 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。

不同坐标系之间的依赖关系详见变量 tool 和 wobj。

10.2 机器人奇异点

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，xMate 机器人的奇异点可以分为如下情况：

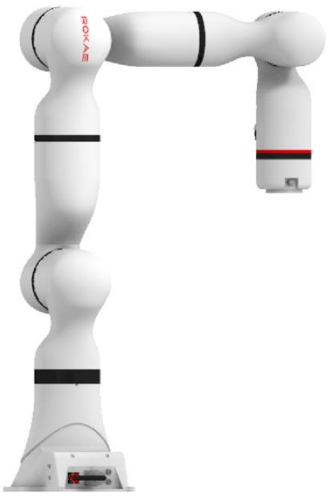
1. 2 轴奇异点
2. 4 轴奇异点
3. 6 轴奇异点
4. 腕心奇异点

另外，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。

2 轴奇异点

当 2 轴角度等于 0°，机器人处于 2 轴奇异状态：



此时，机器人求解逆解时，无法区分 1 轴与 3 轴角度。

4 轴奇异点

当机器人 4 轴角度为 0，机器人处于奇异状态，将这种位姿称为伸展位置：

当机器人处于这种位姿时，机器人沿平行于 3 轴或 5 轴方向的运动受到限制。在机器人运动到工作空间边界位置时，通常会导致机器人处于这种奇异状态。

这种奇异状态会使机器人的手腕根部丧失一个运动自由度（无法使得手腕根部产生沿手臂轴向的运动）。此时求解逆解时，3 轴和 5 轴的位置无法求解。

6 轴奇异点

当机器人 6 轴角度等于 0°时，机器人处于 6 轴奇异状态：

此时，机器人求解逆解时，无法区分 5 轴与 7 轴角度。

腕心奇异点

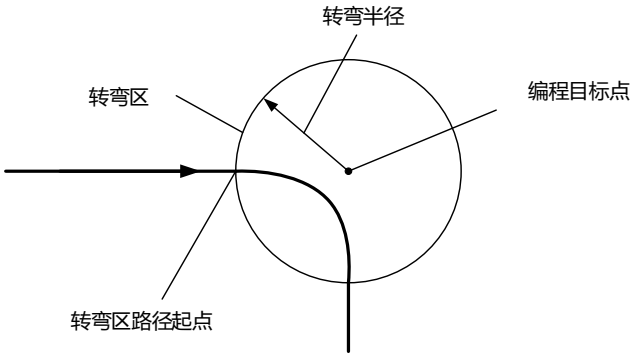
当机器人腕心位于一轴正上方时，机器人处于腕心奇异状态：

此时，机器人求解逆解时，无法准确给定 1 轴角度。

10.2.1 转弯区

说明

对于机械臂来说，其运动通常是按照使用者指定的多条轨迹依次执行的。然而，使用者指定的不同轨迹间通常并非光滑连接，而是存在各种各样的"尖点"。这些"尖点"的存在使得机器人在执行一条轨迹时，必须在该轨迹末尾停止运动，才能运动到下一条轨迹上。为了使机器人能够在不同轨迹间连续运动，必须消除上述尖点""，生成转弯区轨迹将用户指定的不同轨迹平滑地连接起来。见下图：



另外，机械臂在关节空间运动时，其运动同样是在沿着不同轨迹运行的。不同的是，此时的轨迹不再是由笛卡尔位姿定义的，而是由机器人各轴角度直接定义的，所以机器人在关节空间运动时同样存在上述转弯区。

对于转弯区的具体参数设置详见变量 zone。

10.2.2 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，目前只有 Print 指令，逻辑判断指令，用户自定义函数不会打断前瞻机制，其余函数都会打断前瞻。

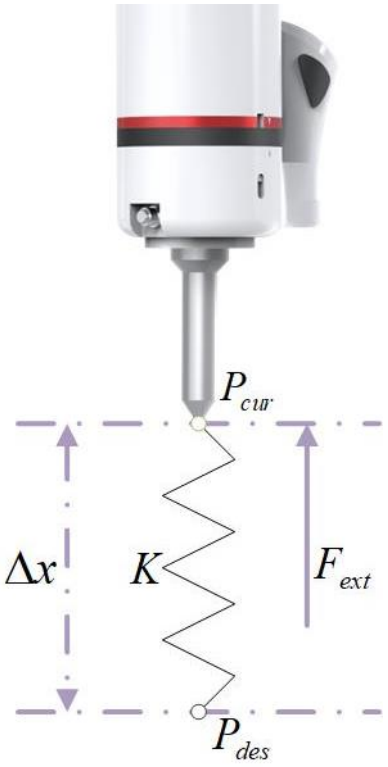
10.3 机器人力控

10.3.1 力控功能简介

机器人力控制是机器人末端与外部环境存在力交互的控制过程。在非接触类机器人运动控制过程中，只关注位置控制过程（速度与精度）。当与环境存在接触时，纯位置控制要求机器人和环境必须具有非常高的精度，避免位置偏差引起的接触力对机器人和环境造成损坏。与单纯位置控制不同，机器人力控制在与环境交互过程中引入力/力矩反馈回路，并通过力/力矩反馈回路改变机器人的运动特性，从而起到与外部环境动态交互的作用。在机器人与外部环境存在偏差或不确定性时，力控会智能地调整预设的位置轨迹，消除位置偏差引起的内力，保证交互过程的平稳安全。

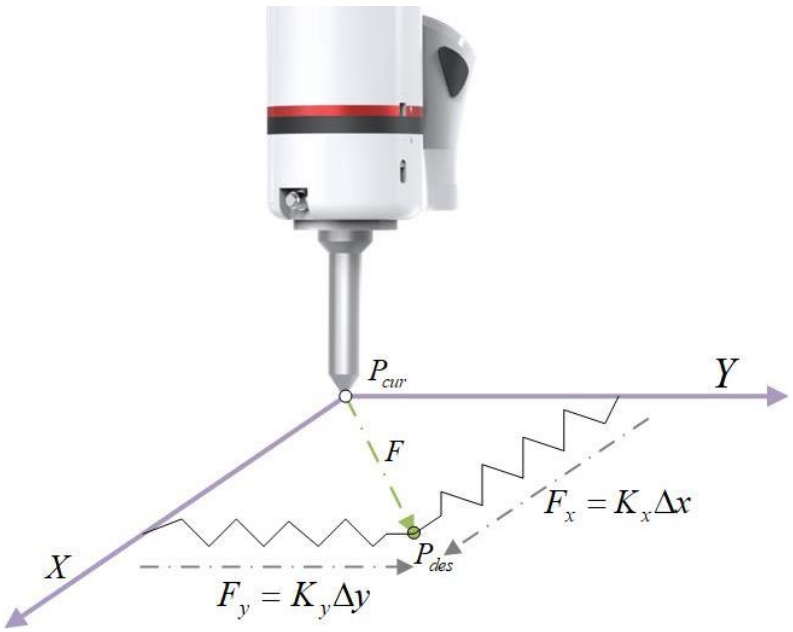
10.3.2 阻抗控制

相比传统工业机器人，xMate 关节中装载有关节扭矩传感器，这使其能够精准的感知关节扭矩。关节的扭矩信息使得 xMate 能够通过阻抗来实现力控制，阻抗控制使得机器人具有非常柔顺的交互行为。这意味着机器人与环境的交互相当于一个虚拟的弹簧刚度、阻尼系统。此时，机器人对外力具有敏感性，外力的作用能够使机器人偏离预先设定的轨迹。而外力的作用消失后，机器人能够具有一定的回弹效果。



阻抗运动过程中，在环境中的外力作用下，机器人的实际位置会偏离期望位置一定距离，偏离的距离取决于阻抗刚度和外力的大小，偏差的具体数值可以通过外力和阻抗刚度的比值得到。如上图，阻抗控制模式下，阻抗刚度设置为 K，外力 F_{ext} 的作用下，机器人的当前位置 P_{cur} 会偏离期望位置 P_{des}，位置偏差为Δx。此偏差引起的阻抗力和外力会达到最终的平衡。

阻抗各方向的刚度可以单独设置，各方向阻抗力为此方向阻抗刚度和位置偏差的乘积，各方向阻抗力最终合成总的阻抗力。如下图，阻抗模式下，在外力的作用下机器人当前位置 P_{cur} 偏离期望位置 P_{des}。X、Y 方向的偏差分别为Δx、Δy，阻抗刚度分别为 K_x、K_y，阻抗力分别为 F_x、F_y。总的阻抗力 $F = F_x + F_y$。



10.3.3 搜索运动

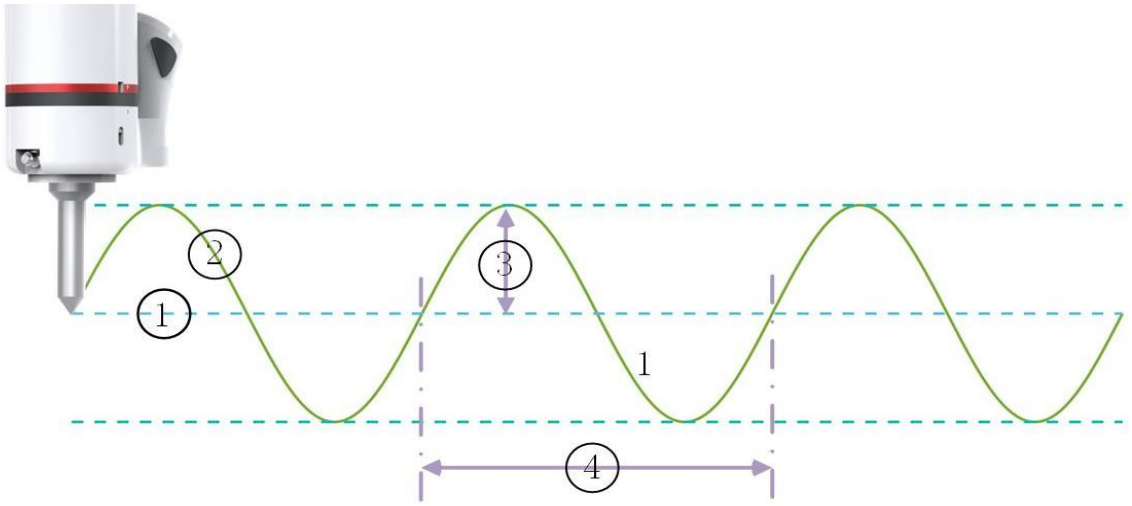
人手在工件装配的过程中，会感受装配过程中力的变化，当感受到阻碍的时候（工件卡住），会尝试通过抖动，来使工件顺利安装。力控制使得 xMate 也具有这样的手法，也即搜索运动，xMate 支持绕轴旋转的正弦搜索运动和平面内的莉萨如搜素运动。搜索运动是在机器人既定运动的基础上叠加的额外运动，搜索运动使得机器人表现出一定的抖动，从而能够在装配过程中更好的克服阻碍。下图是一个正弦搜索运动过程：

1、期望轨迹

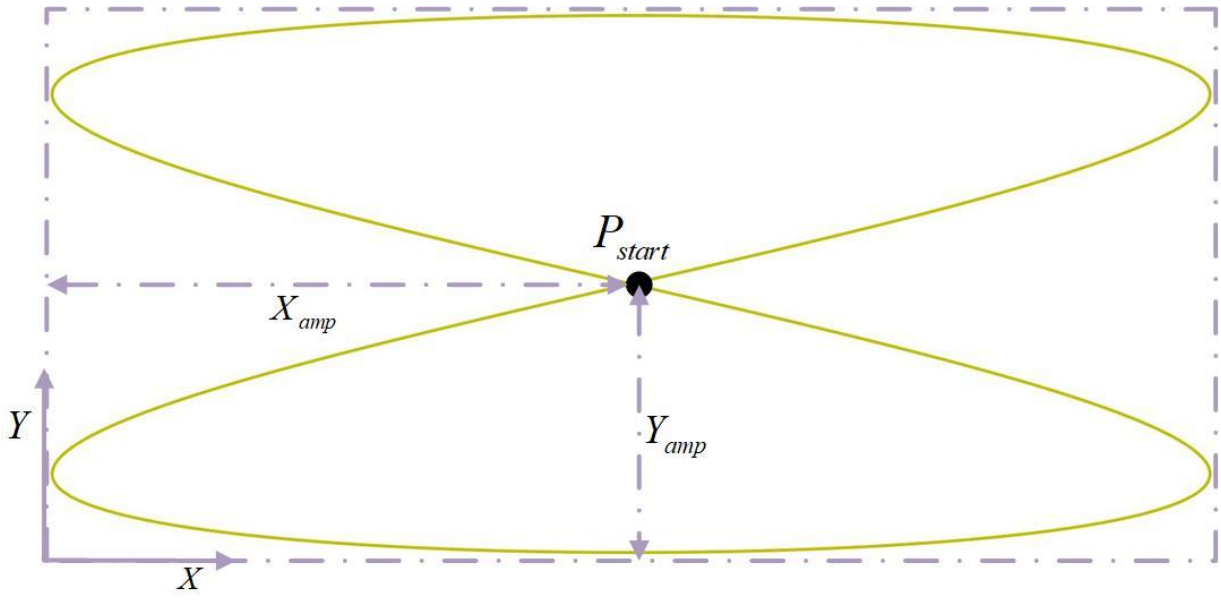
2、实际轨迹（期望轨迹+搜索运动）

3、搜索运动幅值

4、搜索运动周期



所谓莉萨如搜索运动，指的是平面内两个相垂直的方向分别施加一个正弦搜索运动，两方向正弦搜索运动的频率往往成一定的比例。例如下图所示为 XY 平面类的莉萨如搜索运动，其中 X 和 Y 方向搜索运动频率的比值为 2:1，中心点 P_{start} 为期望位姿点，X_{amp} 为 X 方向搜索运动的幅值，Y_{amp} 为 Y 方向搜索运动的幅值。

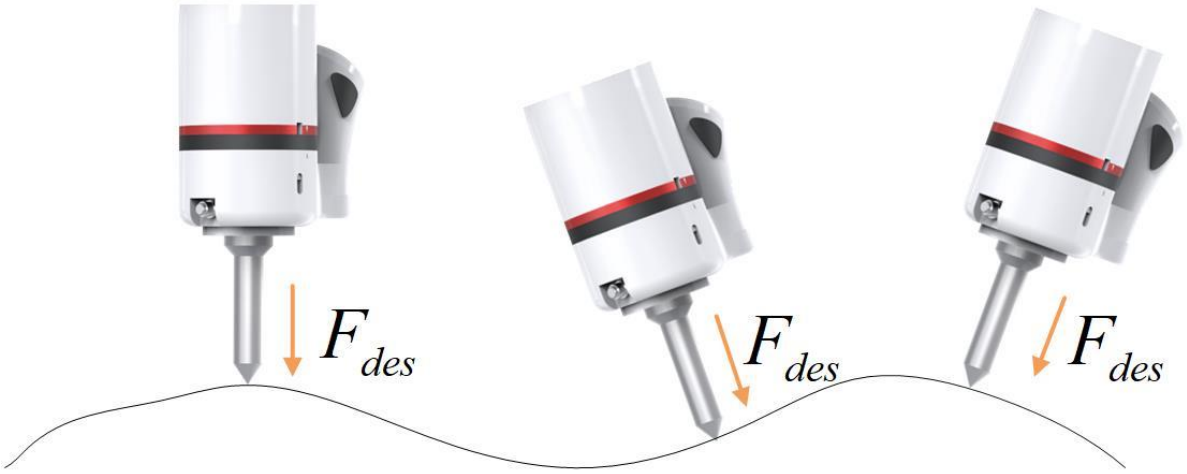


10.3.4 应用场景

工业机器人的力控应用场景大致可以分为两类：恒力跟踪和力控装配

1) 恒力跟踪

如下图，在恒力跟踪应用场景中，机器人保证与曲面之间的接触力保持一个恒定值 F_{des}，同时机器人能够顺应曲面起伏的变化。恒力跟踪主要应用在打磨、去毛刺等应用场景。



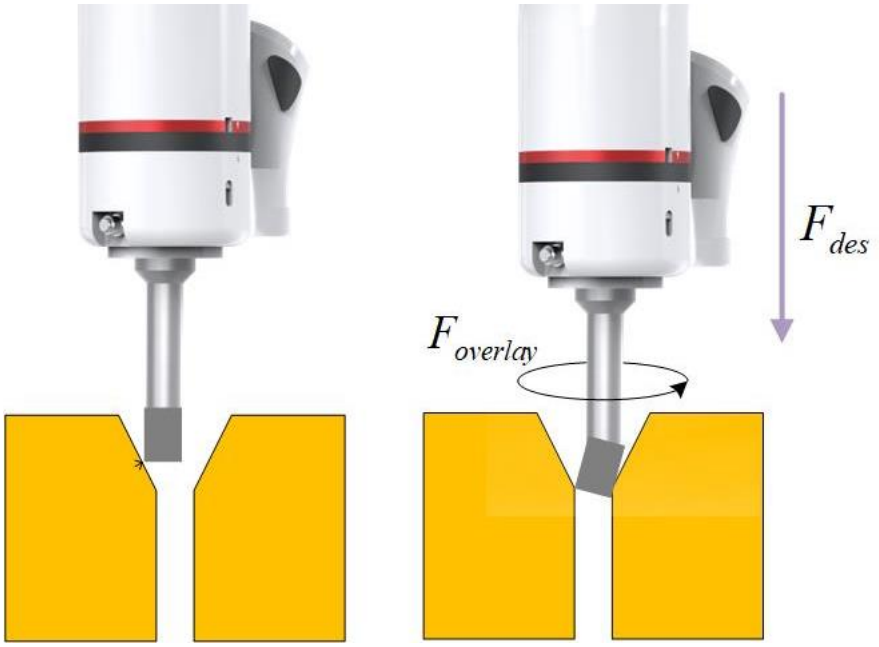
恒力打磨示例程序：程序的含义是，机器人设置为笛卡尔阻抗模式，设置阻抗刚度和负载信息，同时开启力控。通过施加 z 方向的期望力将工件压到打磨面上，在下压的过程中监控 z 方向的观测力，当此观测力超过一定的阈值，就认为工具已经接触曲面，这时施加一个打磨方向的期望轨迹。运动过程中机器人保持恒力，从而实现恒力打磨的效果。

```
VAR POSE T_POSE = PE:{0, 0, 0},{1, 0, 0, 0}  //工具坐标系
VAR POSE W_POSE = PE:{0, 0, 0},{1, 0, 0, 0}  //工件坐标系
FcInit T_POSE, W_POSE, 0 //力控初始化，0 代表力控坐标系为基座标系
SetControlType 1 //设置阻抗类型为笛卡尔阻抗，0：关节阻抗 1：笛卡尔阻抗
SetCartCtrlStiffVec 500, 500, 0, 100, 100, 100 //设置笛卡尔阻抗刚度，前三项是平移刚度(0~1500)，后三项旋转刚度(0~100)
SetCartNsStiff 2.0 //设置零空间刚度（0~4）
SetLoad 0.82,0,0,0.041,0,0,0 //设置负载信息
FcStart //开启力控
SetCartForceDes 0, 0, -15, 0, 0, 0 //设置笛卡尔空间期望力，Z 方向施加一个-15N 的期望
FcCondForce -100, 100, -100, 100, -100, 10, true, 20.0 //笛卡尔空间力监控，Z 方向需要施加一个 10N 的力来触发条件
FcCondWaitWhile //开启前面设置的监控条件
MoveL p0,v800,z50,tool0 //运动指令(期望轨迹)
FcStop //关闭力控模块
```

2) 力控装配

机器人装配工艺中，如果使用纯位置控制，由于位置、建模误差，机器人很容易就会碰撞到工件上，从而造成工件或机器人的损伤。

而在基于力控制的装配工艺中，机器人在感受到外力超过一定范围（工件卡住），会尝试施加额外的搜索运动（抖动）来克服阻碍，从而使工件得以顺利安装。如下图，左边纯位置控制在装配过程中会发生碰撞，右边力控制在装配过程中，通过期望力 F_{des} 将机器人推入装配孔，通过搜索运动 $F_{overlay}$ 克服安装过程工件卡死的情况。



力控装配示例程序：程序的含义是，机器人设置为笛卡尔阻抗模式，设置阻抗刚度和负载信息，同时开启力控。通过施加 z 方向的期望力将工件压入安装孔，在压入的过程中监控 z 方向的观测力，当此观测力超过一定的阈值，就认为工件卡住了，这时开启预先设置的莉萨如搜索运动来保证工件的顺利推入，同时通过动态监控 z 方向的位置来判断工件是否已经完成安装。

```
VAR POSE T_POSE = PE:{0, 0, 0},{1, 0, 0, 0} //工具坐标系
VAR POSE W_POSE = PE:{0, 0, 0},{1, 0, 0, 0} //工件坐标系
FcInit T_POSE, W_POSE, 0 //力控初始化
SetControlType 1 //设置阻抗类型为笛卡尔阻抗，0：关节阻抗 1：笛卡尔阻抗
SetCartCtrlStiffVec 500, 500, 0, 100, 100, 100 //设置阻抗刚度，前三项是平移刚度(0~1500)，后三项旋转刚度(0~100)
SetCartNsStiff 2.0 //设置零空间刚度（0~4）
SetLoad 0.82,0,0,0.041,0,0,0 //设置负载信息
SetLissajousOverlay 0, 5, 5, 5, 5, 0 //设置莉萨如搜索运动 XY 平面, 5N, 5Hz,5N, 5Hz, 0rad
FcStart //开启力控
SetCartForceDes 0, 0, -15, 0, 0, 0 //设置笛卡尔空间期望力，Z 方向存在一个-15N 的期望力
FcCondForce -100, 100, -100, 100, -100, 10, true, 20.0 //笛卡尔空间力监控，Z 方向需要施加一个 10N 的力来触发条件
FcCondWaitWhile //开启前面设置的监控条件
StartOverlay // 开启搜索运动
VAR fcboxvol box1 = fcbv:{-1000.0, 1000.0, -1000.0, 1000.0, 250.0, 500.0} //定义盒状区域
FcCondPosBox F_POSE, box1, true, 20.0 //盒状区域监控，超出此盒状区域，触发条件
FcCondWaitWhile //开启前面设置的监控条件
FcStop //关闭力控模块
```

11 编程与调试

11.1 编程准备

编程工具

您可以使用 RobotAssist 和 RokaeStudio 编程。RobotAssist 最合适用于在线修改程序，如位置和路径变量。RokaeStudio 适用于离线编程与仿真，包括方案设计、设备选型、轨迹生成、轨迹优化、仿真调试和代码生成。

有关如何使用 RokaeStudio 离线编程的详情，请参阅 RokaeStudio 使用手册。

定义工具、有效载荷和工件

在开始编程前需定义工具、有效载荷和工件，默认工具为 tool0，工件为 wobj0。然后，您可以随时定义更多所需对象。

定义坐标系

确认机器人的安装过程中正确设置了基坐标系。

在开始编程前，您可以根据需要定义工具坐标系和工件坐标系。以后添加更多对象时，也需要定义相应坐标系。

11.2 认识工程

11.2.1 工程介绍

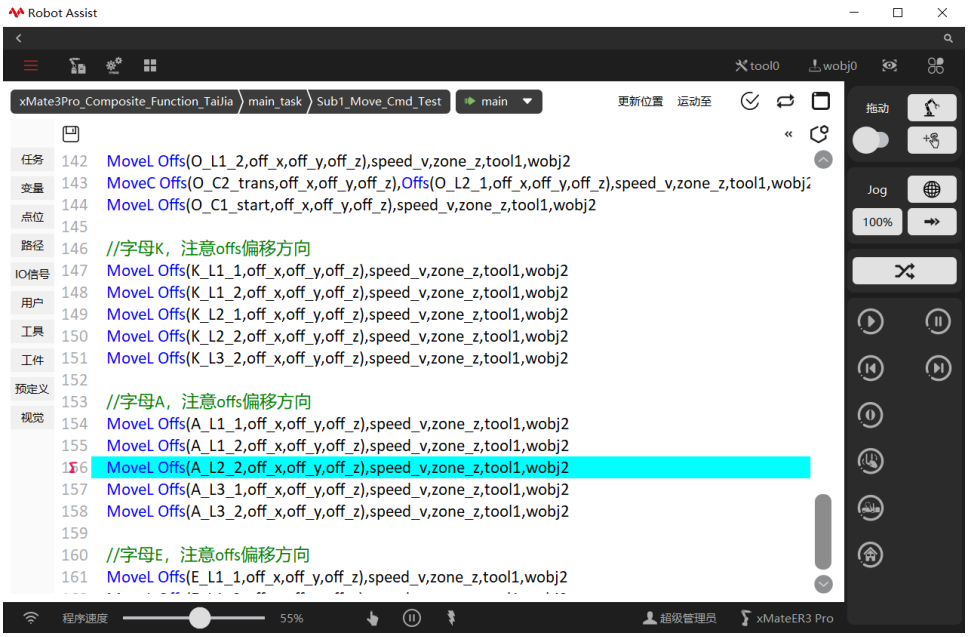
工程概述

在 xCore 控制器中，工程指的是控制机器人运行的程序、任务等对象的管理集合，其负责存储机器人工作所需要的所有必要信息，具体包括：

- 任务列表；
- 变量列表；
- 点位列表；
- 路径列表；
- IO 信号列表；
- 用户坐标系列表；
- 工具坐标系列表；
- 工件坐标系列表；
- 预定义参数；
- 视觉；

打开工程

点击左上角的  进入到工程界面；



11.2.2 工程配置

说明

工程配置界面用于对当前工程进行相关配置，包括：

- RobotAssist 软件与控制器工程同步；
- 工程切换；
- 工程导入/导出；
- 新建，修改，删除工程

RobotAssist 软件与控制器工程同步

与控制器建立连接后，本地工程会立即载入更新以保持与控制器一致。当本地的一些重要工程文件发生变化时会立即同步至控制器以保证机器人功能正常，例如：工具、工件、用户坐标系等。另外 RL 代码操作比较灵活，目前仅在执行运行调试等操作时才会自动推送至控制器，如果工作未完成需要保存，点击推送至控制器按钮，将工程手动推送至控制器。

工程切换

点击选择工程下方的下拉菜单，显示所有工程，选择想要切换的工程，点击下方的重新加载对工程进行切换。

新建工程

点击 + 创建新的工程，工程名称只能是英文字母，数字和下划线"_"的集合。

点击新建按钮之后会进入新建工程向导界面支持轻松创建和导入相关配置。新建工程默认任务是任务 0。

修改工程

点击 对当前工程进行修改。

删除工程

点击 删除当前工程。

> **警告**
>
> 文件一旦删除，无法恢复！

11.2.3 任务列表

11.2.3.1 什么是多任务？

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

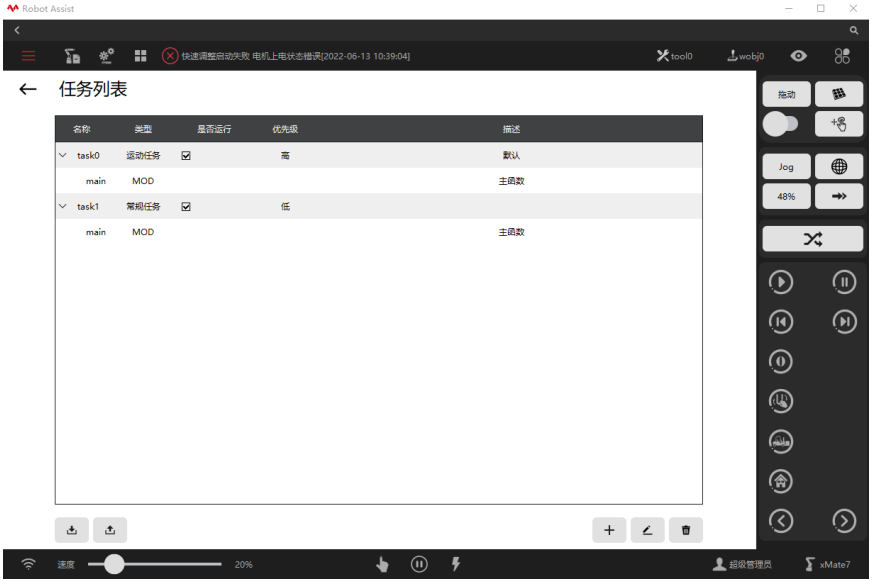
辑的限制。

- 机器人工作的时候通过示教器获取一些输入。
- 控制、激活/反激活外部设备。

11.2.3.2 任务列表

概述

xCore 系统提供了对并行处理任务的管理界面，界面上会显示各个并行任务的任务属性，每个任务的控制逻辑在 Task 中实现，用户可以通过这个界面新建，设置，删除任务。



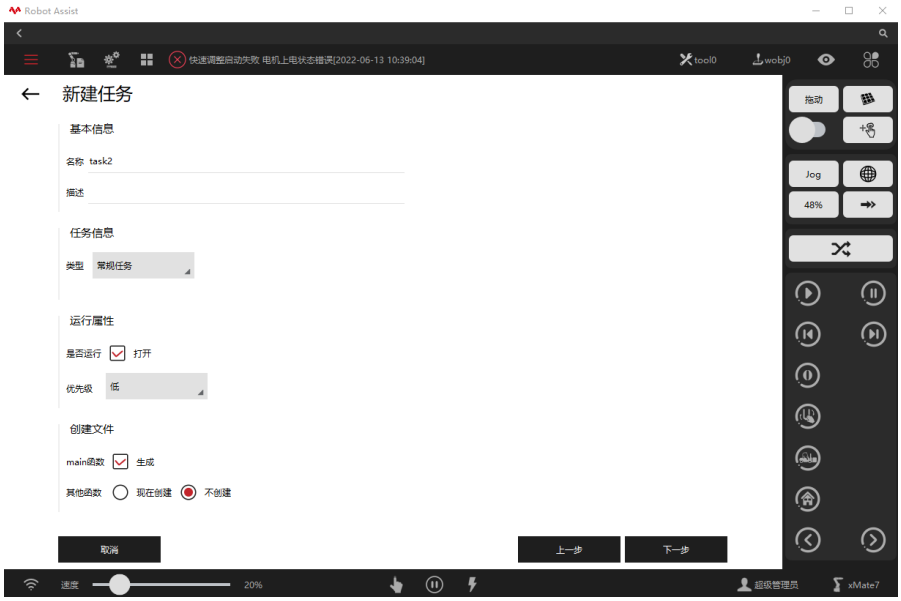
11.2.3.3 新建任务

任务属性

任务属性	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。
任务类型	运动任务指可以使用 RL 指令来控制机器人的运动。只能有一个任务是运动任务。
自动开始	配合生产模式使用，选中自动开始表示系统重启时程序开始重新执行。通常情况下不会被示教器或者急停停止。
优先级	设置任务运行的优先级
创建文件	勾中生成 main 函数时，创建任务之后会自动生成 main 函数。其他函数同理。

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。



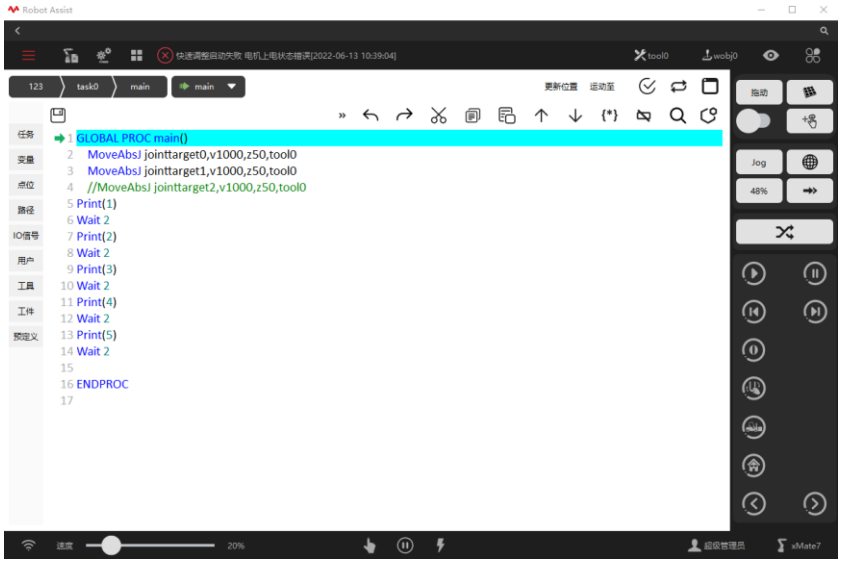
使用限制

- 支持 10 个任务。
- 最多只有一个运动任务。
- 更改任务类型、任务入口函数、是否运动任务属性即时生效。

11.2.3.4 启动和运行任务

说明

在 RL 代码界面中点击 task0 选择任务。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。



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 PPToMain 时，所有没有运行的任务都执行 PPToMain。
- 有任务运行时，禁止修改任务列表中的内容。

11.2.3.5 任务间通信

说明

任务间通信支持两种方式，PERS 变量和中断。

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。

任务间使用中断协调执行顺序

- 在需要等待的任务中定义中断以及对应的中断处理函数。
- 在被等待的任务的合适地方设置中断触发信号。

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。

11.2.4 变量列表

11.2.4.1 变量

11.2.4.1.1 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；

1. 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
>
> 1. 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 。
> 2. 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。

例 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；

例 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；

例 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)；

11.2.4.1.2 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：

例 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
```

例 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"。

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

11.2.4.1.3 用户变量保持

说明

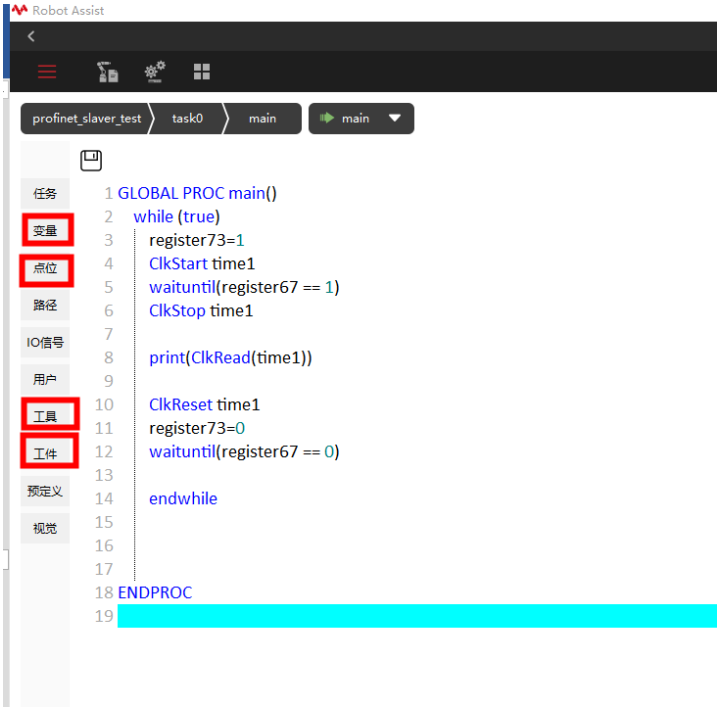
在 RL 的同一个工程中创建一个带保持的用户变量 a，该用户变量被标记成 pers 变量，那么该变量的值在 RL 停止、机器人重启、机器人关机、机器人断电时保持在非易失存储介质上，当机器人重新上电或者 RL 重新运行时，该变量 a 的值就会恢复成保持值。只有变量第一次创建或者变量重新编辑，变量才会被赋初始值。

用户变量目前支持保持的数据类型

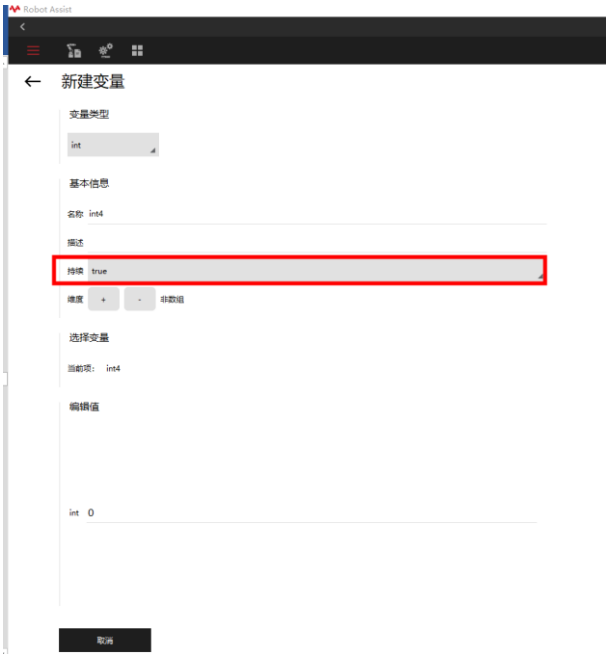
Int、byte、double、bool、string、robtarget、jointtarget、pose、speed、zone、fcboxvol、Fcspherevol、fccylindervol、fcxyznum、fccartnum、torqueinfo、tool、wobj

用户变量保持配置

在 RL 工程界面，可以创建保持变量的入口如下红框所示：



点击变量、点位、工具、工件任一按钮，进入创建该类型的用户变量。所有可以创建保持属性的变量都有一个"持续"下拉框，选择 true 表示该变量是保持变量，即标记为 pers 变量。比如创建一个 int 类型的 pers 变量，配置如下：（其他类型以此类推）



11.2.4.2 变量列表

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：

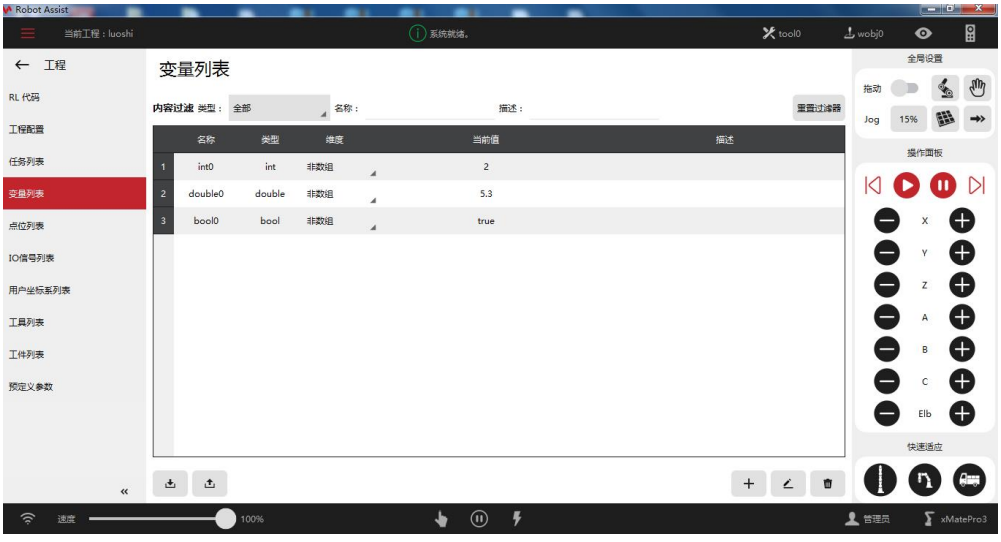
序号	变量类型	描述
1	系统预定义变量	指用户不可修改的变量，用于存储某些系统参数，例如 tool0/wobj0 等。
2	用户预定义变量	用户可以进行修改，并且可以在多个程序中进行使用的变量，例如用户标定的工具、工件等。
3	程序变量	用户在程序中进行定义的变量，一般仅在当前程序及其子程序中使用，包含了绝大部分系统支持的变量类型。

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。
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。

11.2.5 点位列表

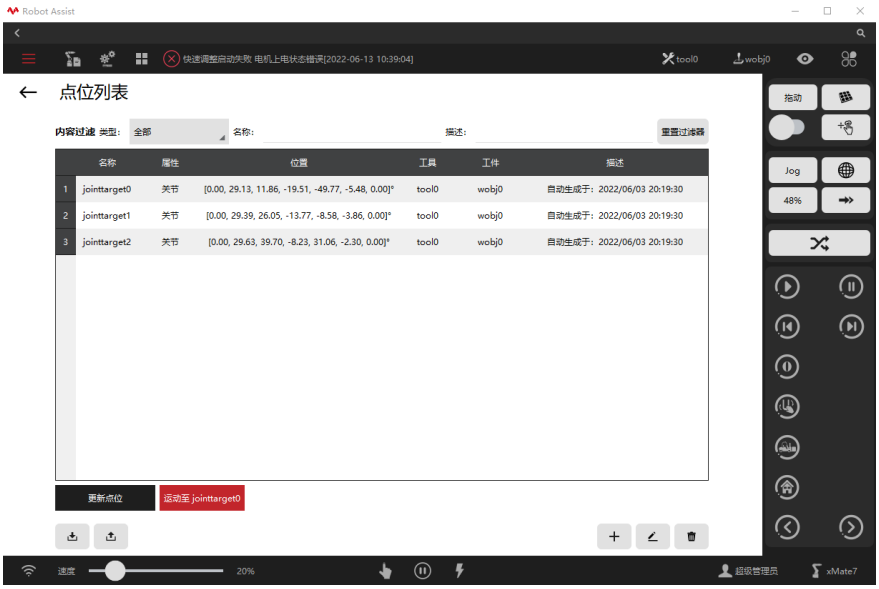
说明

xCore 系统提供对示教点位管理的界面，RL 程序中用到的点位信息都需要在点位列表进行配置后才可以在程序中使用。

支持在编辑点位页面以当前位姿更新示教点位置；支持在点位列表页面已当前位姿更新示教点位置。

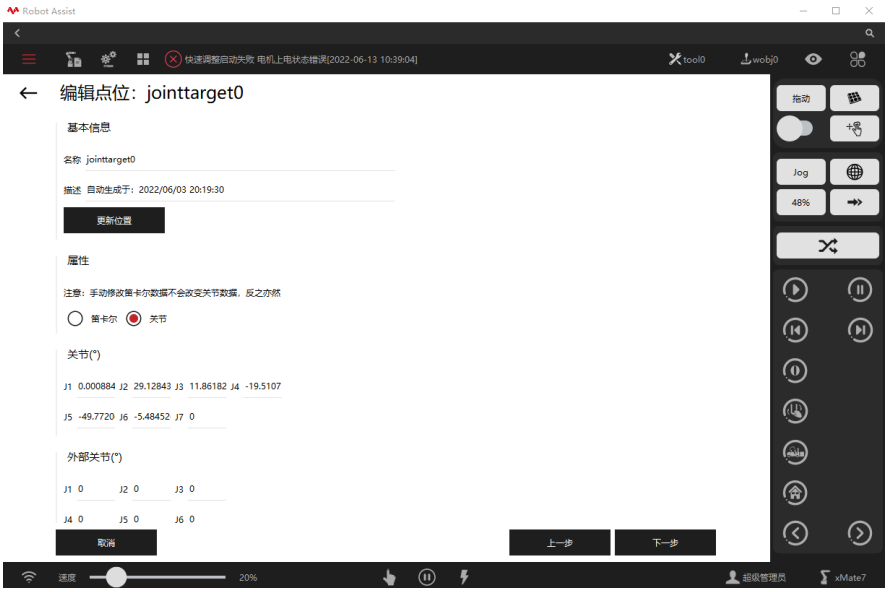
添加、修改、删除 点位

所有的点位信息的配置都在点位列表页面进行，如下图所示：



A	点位名称，可以在“新建”或者“修改”时更改。
B	点位属性，包括关节空间和笛卡尔空间。
C	位置，关节空间点，显示七个轴的关节角；笛卡尔空间点，显示基坐标系下 xyz 坐标和机器人臂角。
D	描述，用户可对点位进行相应描述，可以在“新建”或者“修改”时更改。

编辑点位



	操作	说明
1	使用 admin 级别的用户登录系统，并切换到点位列表界面	
2	点击右下角的"+"进入点位新建引导界面。	也可点击 对点位进行修改或者点击 删除点位。
3	在名称栏对当前点位进行命名	
4	在描述栏对当前点位添加描述	非必要
5	点击更新位置。	以当前位姿更新示教点信息。
6	选择笛卡尔空间点或是关节空间点	用来手动更新点位信息，如果使用步骤 5 中更新点位方式则可以不选择此项。
7	根据点位属性手动输入点位位姿	

11.2.6 路径列表

TBD

11.2.7 IO 信号列表

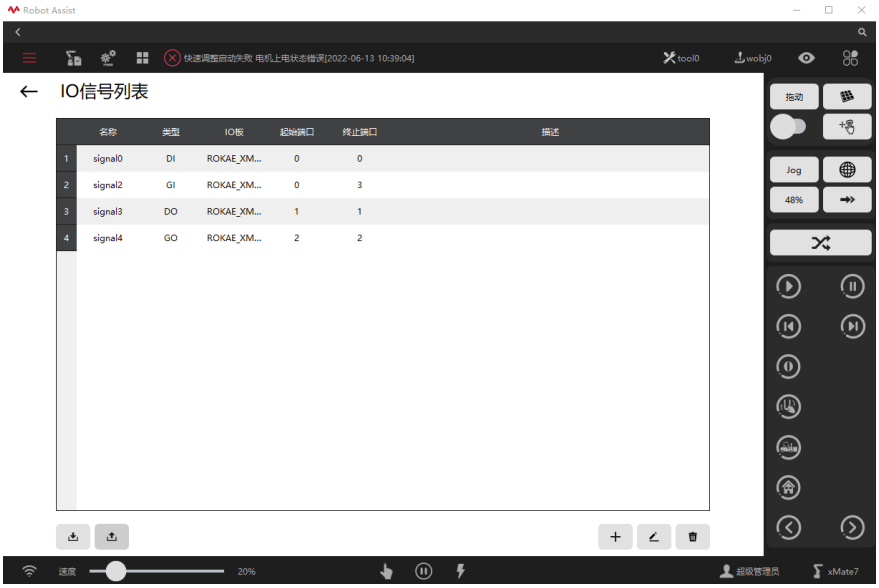
说明

xCore 系统中，除默认信号外，其他所有通用 IO 信号（包括 Profinet 信号）都需要在 IO 信号列表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 IO 列表页面进行，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，可以是珞石提供的标准 IO 模块，也可以是 Profinet 总线或 Ehternet/IP 总线。
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。

例 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

- 用户自定义 IO 不可映射到系统输出上。

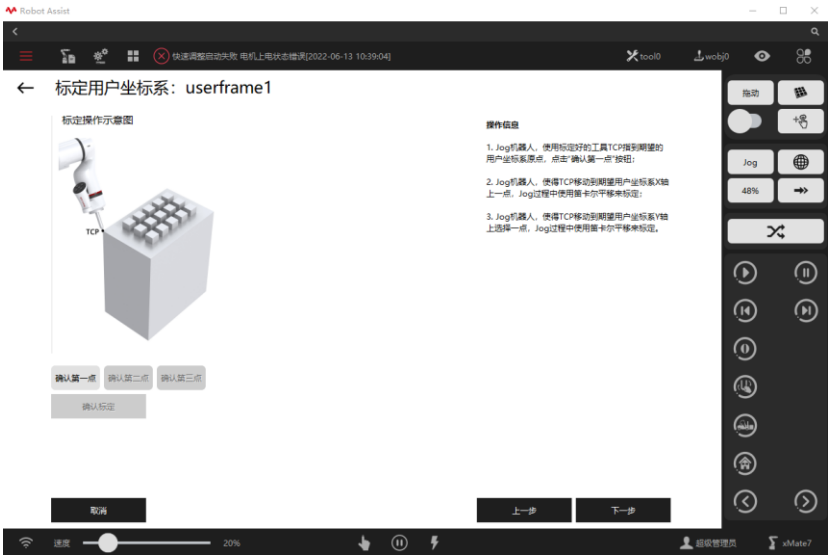
11.2.8 用户坐标系列表

说明

用户坐标系，在定义工件坐标系时充当参考系，不单独使用。

标定用户坐标系

标定用户坐标系使用的方法也称为 3 点法，操作步骤与标定工具坐标系的 3 点法基本类似。在标定用户坐标系之前，需要首先标定一个工具，然后使用该工具的 TCP 来进行工件坐标系标定，为方便操作，推荐使用带有尖端的工具。



	操作	说明
1	使用 admin 级别的用户登录系统，并切换到用户坐标系列表界面	
2	在名称栏对将要标定的用户坐标系进行命名。	
3	在位姿标定选项中选择立刻标定	如果提前知道用户坐标系，可以手动输入。也可选择不标定，默认用户坐标系是世界坐标系。
4	Jog 机器人，使标定好的工具 TCP 指到期望的工件坐标系原点，点击"确认第一点"按钮	世界坐标系
5	Jog 机器人，使标定好的工具 TCP 指到期望的工件坐标系 X 轴上一点，然后确认第二个点。	第二个点与第一个点的连线即工件坐标系的 X 轴。
6	Jog 机器人，使标定好的工具 TCP 指到期望的工件坐标系 XY 平面上一点，然后确认第三个点	也可以在期望的 Y 轴上选择一点，因为 Y 轴上的点也在 XY 平面上。

11.2.9 工具坐标系列表

11.2.9.1 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（工具中心点）的数据。

当使用工具时，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。

说明

使用工具前首先需要定义新工具。在 xCore 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）。

简单来讲，我们需要获得关于工具的以下参数：

- 工具的 TCP 和姿态，即标定工具坐标系；
- 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
>
> 1. 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。
> 2. tool0 变量不允许修改。

11.2.9.2 工具中心点

定义

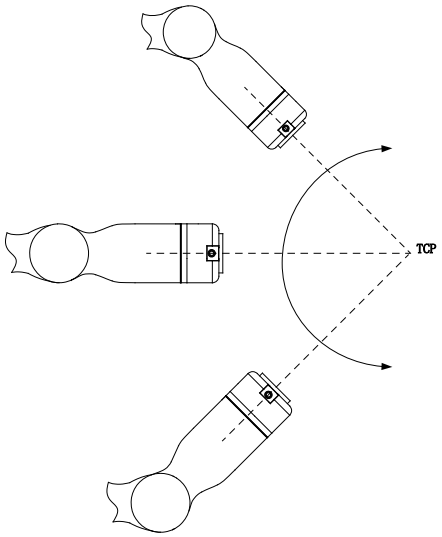
工具中心点（Tool Center Point，TCP）是位于工具上的一个特定点，通常情况下机器人使用该“点”进行加工作业，例如一个焊枪的焊丝尖端，气动手爪的某一个手指顶端等。机器人可绕 TCP 点旋转变换姿态而保持 TCP 的位置不变。

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



11.2.9.3 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。

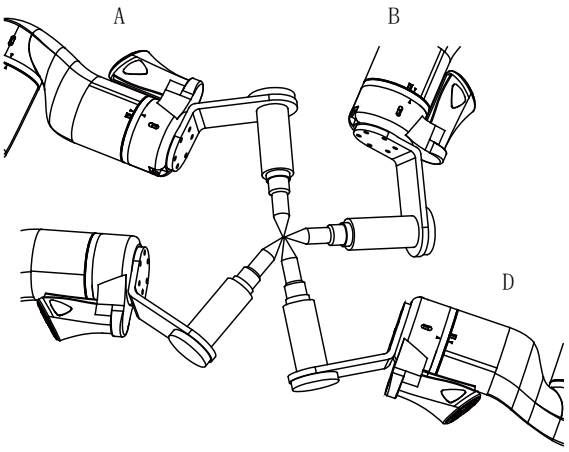
如果您使用的工具生产商提供了这些偏移量数据，那么可以在示教器上选择"手动输入"方式直接输入而不必执行标定过程。

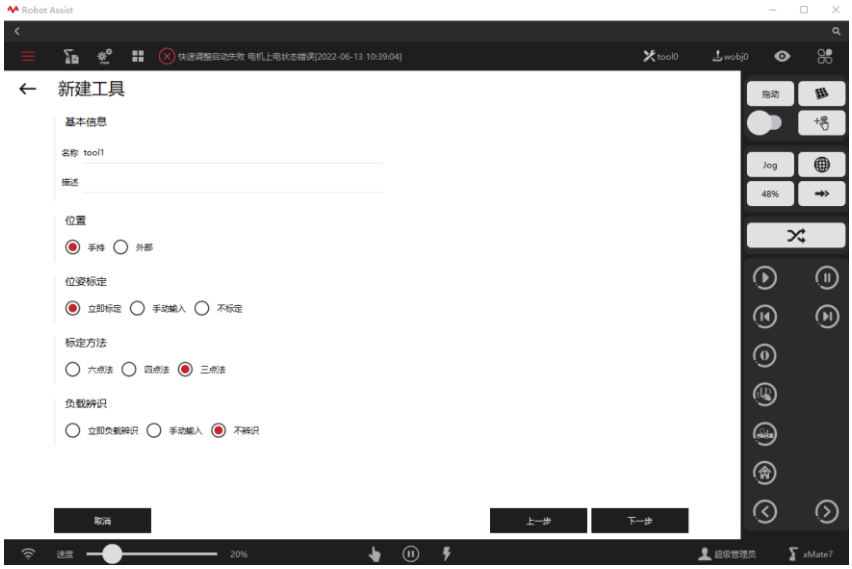
对于没有尺寸数据的工具，则需要使用 xCore 提供三种的方法来进行工具坐标系标定：

- 4 点法，用来标定工具坐标系的原点；
- 3 点法，使用 4 点法完成坐标系原点的标定后用 3 点法来标定坐标系姿态；
- 6 点法，同时标定坐标系的原点和姿态，相当于 4 点法和 3 点法的集合。

标定工具坐标系位姿

标定工具坐标系之前，需要准备一个固定的外部尖端点，要求该点位于机器人工作范围之内，并且使用待标定的工具能以比较灵活的姿态接触到该点。在 HMI 的工具标定界面有更为详细的示意图可供参考。





	操作	说明
1	在待标定的工具上选择一个合适的点，这个点将作为工具坐标系的原点，即 TCP。	通常情况下 TCP 都选择在工艺加工点上，例如弧焊枪的焊丝尖端、手爪的指尖等。 当然也可以根据实际情况将 TCP 放置在工具上的任何地方。
2	在工具列表界面点击右下角的"+"进入新建工具向导界面。	对标定工具进行命名。
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 6 点法。	也可以选择 4 点法或 3 点法。 4 点法只标定工具原点，三点法只标定工具坐标系姿态。
5	选择负载辨识为立即负载辨识。	如果客户不要工具负载参数，则可以选择不辨识。
6	Jog 机器人，使得选定的 TCP 与外部尖端点接触，之后点击"确认第一点"按钮。	当两个点靠近时，使用增量模式可以更好的调整位置。
7	重复步骤 6，直到 4 个点全部确认完毕。	为了获得更高的标定精度，4 个点之间的姿态差距应尽可能的大，即机器人应尽量以不同的

		姿态接触外部尖端点。

警告

如果机器人安装在导轨上，那么在标定过程中禁止操作导轨，否则将造成标定失败。

11.2.9.4 工具负载参数

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 xCore 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，尤其注意在使用外部工具时，tool 变量中的 load 参数存储的是对应的工件负载。

使用 4 点法或者 6 点法只能确定工具的运动学参数，工具的动力学参数需要单独指定，定义工具的负载参数有两种方法：

- 如果手头有工具的负载数据，那么可以在工具坐标系标定页面上选择手动输入方法，直接输入对应的数据；
- 如果不知道工具的负载数据，那么需要使用 xCore 系统提供的负载辨识功能来进行辨识。

进行负载辨识

负载辨识功能可以方便的计算工具的动力学参数。

进行工具负载辨识的步骤是:

- 1 将机器人切换到自动模式，并执行上电操作；

- 2 在空载的状态下运行无负载辨识程序，等待辨识程序完成；
- 3 装上工具负载，运行有负载辨识程序，等待辨识程序完成；
- 4 辨识完成出现辨识结果弹窗，点击保存。

提示

1. 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。
2. 开始辨识之前先将机器人预热半小时以上可提高辨识准确度。
3. 负载惯量计算基于法兰坐标系。
4. 只支持机器人正装情况下的负载辨识。

注意事项

在辨识过程中出现以下情况，将会导致辨识过程终止，所有已获得的辨识数据丢失，需要重新开始辨识：

- 辨识操作进行到中间时，选择了其他工具或切换到其他界面；
- 辨识程序运行过程中触发急停或者外部安全停止；
- 辨识程序运行过程中，从自动模式转换到手动模式会导致辨识失败。

警告

辨识程序需要在自动模式下执行，因此各项防护措施均应处于有效状态，外部控制信号可能随时启动机器人，因此请在完成安装且人员全部退到安全区域时再切换到自动模式执行。

11.2.9.5 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。

11.2.9.6 外部工具

什么是外部工具

通常情况下，我们把工具安装在机器人上，工具随机器人运动来完成指定的工作，我们把这样的工具叫做普通工具，常见的工具如抓手，吸盘，焊枪等都是普通工具。

但是在某些特定情况下把工具安装到机器人上会影响正常使用，例如：

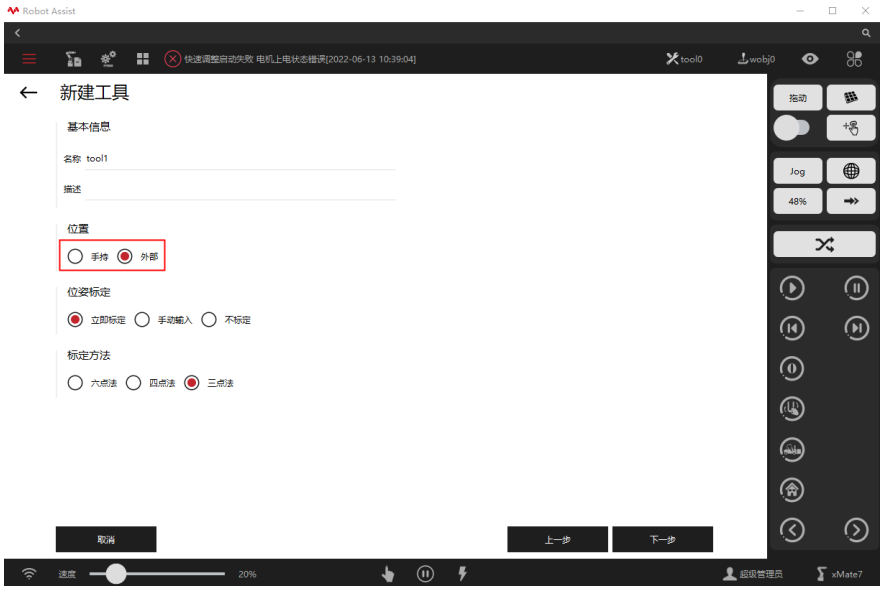
2. 要使用的工具尺寸比较大或者比较重，安装到机器人上不方便或者可能会影响到机器人的运动性能；
3. 待加工的工件尺寸比较大，正常情况下机器人工作范围无法很好的覆盖；
4. 需要完成一些特定的工艺动作，例如打磨一个方形物体时，需要分别绕 4 个角做自旋动作。

在这些情况下，选择把工件安装到机器人上而把工具固定在外部的某个位置会更合适更方便。我们把这些安装在机器人之外，固定在某个位置不动的工具称为外部工具（有些品牌称为 Stationary Tool 或者 Remote TCP）。

创建外部工具

在 xCore 系统中，外部工具同样使用 tool 型的变量来描述，在 tool 型变量中使用一个专门的标志位 robhold 来定义某个工具是普通工具还是外部工具。

使用示教器创建外部工具非常简单，只需要在工具标定界面选定某个工具，从工具安装位置中选择外部即可。



标定外部工具坐标系

外部工具的标定方法与普通工具一致，支持 4 点法、6 点法和手动输入三种方式，但标定外部工具坐标系需要使用已标定好的普通工具来进行，此处仅以 4 点法为例进行介绍。

	操作	说明
1	使用 admin 级别的用户登录系统，并切换到工具坐标系标定界面。	
2	标定一个带尖端的普通工具，或者选择一个已经标定好的普通工具，标定精度应尽可能的高。	该普通工具用于之后的外部工具标定，此步骤可有效提高外部工具标定的精度。
3	Jog 机器人，使标定好的工具 TCP 指到期望的外部工具坐标系原点，点击“确认第一点”按钮。	
4	Jog 机器人，使标定好的工具 TCP 以不同的姿态指到期望的工件坐标系原点上，然后分别确认第二、三、四个点。	4 个点的选取原则与普通工具标定的 4 点法一致。
5	标定完毕后，系统会弹出标定误差，根据误差大小来选择是否需要重新标定。	有关标定精度的信息请参考确认标定精度。

注意事项

外部工具必须与对应的工件一起使用，即同时选中的工具和工件中的 robhold 参数必须有一个为 False 而另一个为 True，否则系统会给出错误提示，并禁止 Jog 机器人。

使用外部工具时，定义工具坐标系和工件坐标系的参考系与普通工具是不一样的，见下表：

坐标系名称	普通工具时相对于……定义	外部工具时相对于……定义
工件坐标系	用户坐标系	用户坐标系
用户坐标系	世界坐标系	法兰坐标系
工具坐标系	法兰坐标系	世界坐标系

更多信息请参考 tool 变量介绍。

11.2.10 工件坐标系列表

11.2.10.1 什么是工件？

说明

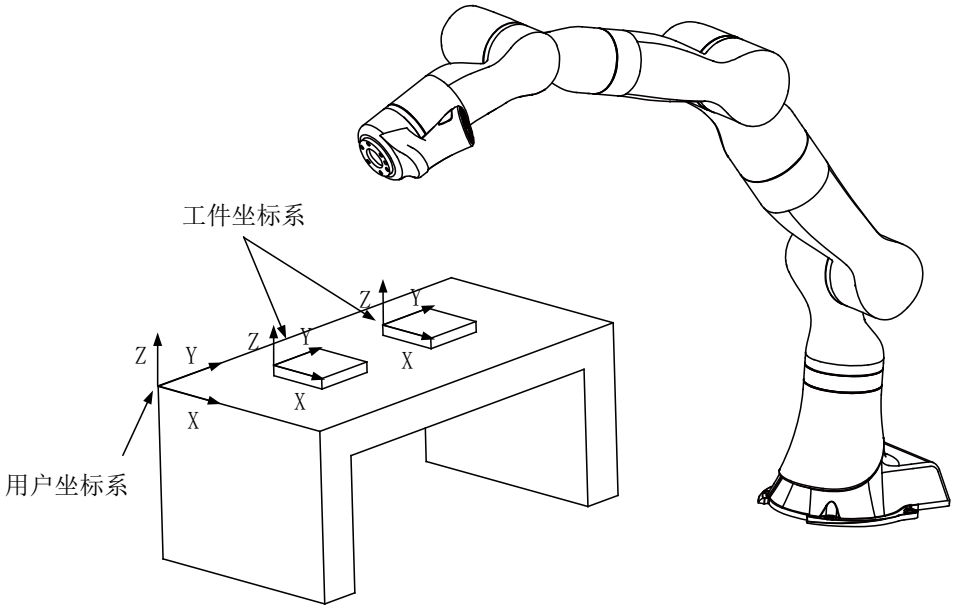
工件是指机器人使用工具进行加工或者处理的物品，在 xCore 系统中使用 wobj（Work Object）类型的变量来描述一个实际的工件。

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。



11.2.10.2 定义工件

说明

使用工件前首先需要定义新工件。在 xCore 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。

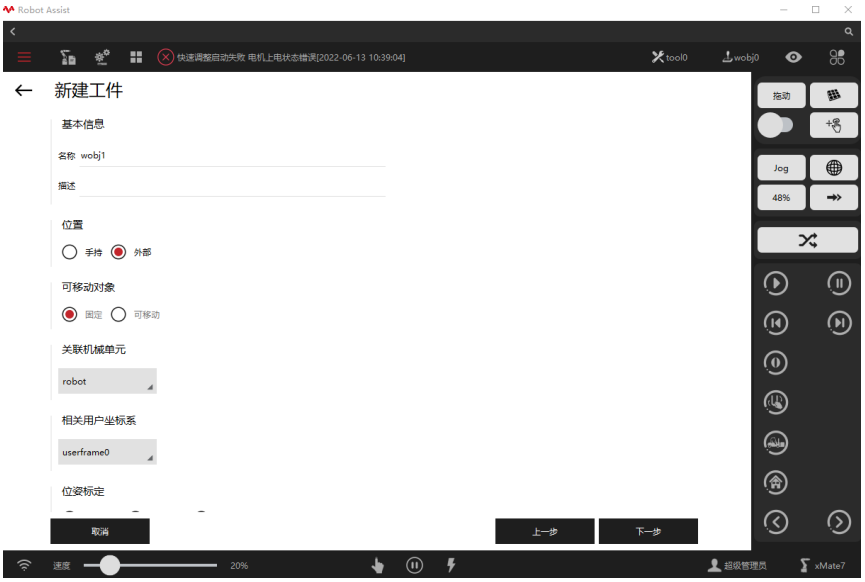
工件 wobj 并没有包含动力学参数，因此定义工件的过程就是标定工件坐标系的过程。

提示

1. 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。
2. 与 tool0 一样，wobj0 也不允许修改。

标定工件坐标系

标定工件坐标系使用的方法也称为 3 点法，操作步骤与标定工具坐标系的 3 点法基本类似。



在标定工件之前，需要首先标定一个工具，然后使用该工具的 TCP 来进行工件坐标系标定，为方便操作，推荐使用带有尖端的工具。

	操作	说明
1	使用 admin 级别的用户登录系统，并切换到工件列表界面	
2	在名称栏对将要标定的工件坐标系进行命名。	用户坐标系不是必选项，默认选择 userframe0，即世界坐标系。
3	位置选择为外部	手持工件标定见下文
4	Jog 机器人，使标定好的工具 TCP 指到期望的工件坐标系原点，点击"确认第一点"按钮	世界坐标系
5	Jog 机器人，使标定好的工具 TCP 指到期望的工件坐标系 X 轴上一点，然后确认第二个点。	第二个点与第一个点的连线即工件坐标系的 X 轴。
6	Jog 机器人，使标定好的工具 TCP 指到期望的工件坐标系 XY 平面上一点，然后确认第三个点	也可以在期望的 Y 轴上选择一点，因为 Y 轴上的点也在 XY 平面上。

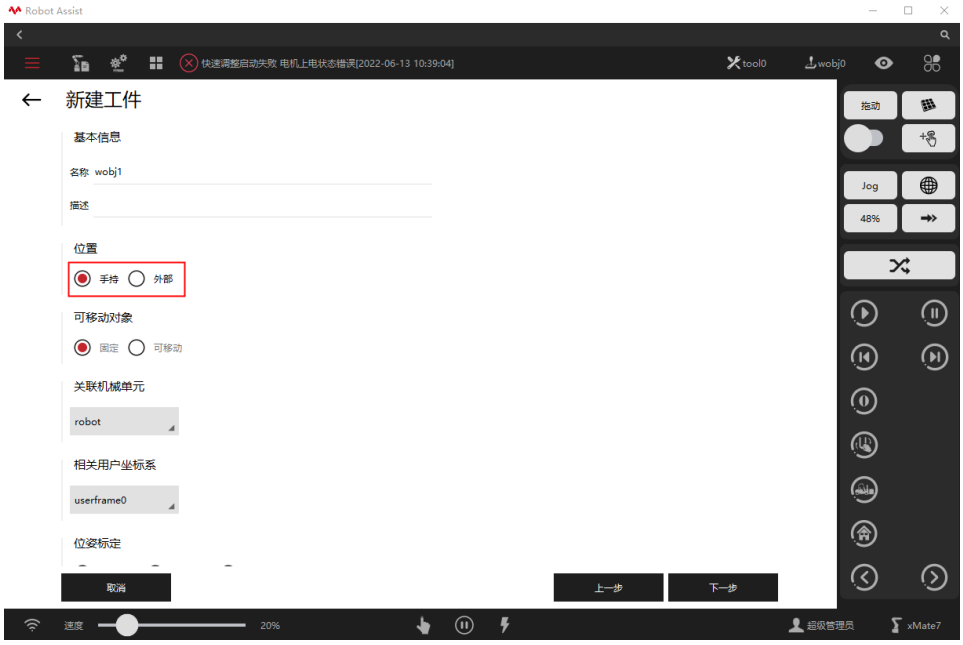
标定手持工件坐标系

如果使用外部工具功能，那么对应的工件将安装在机器人上，我们称之为手持工件。

手持工件同样需要标定工件坐标系，且必须使用已经定义好的外部工具进行标定，更多信息请参考外部工具功能。

标定手持工件坐标系的一般步骤如下：

	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	在名称栏对将要标定的工件坐标系进行命名。	用户坐标系不是必选项，默认选择 userframe0，即世界坐标系。
3	位置选择为手持	
4	Jog 机器人，使标定好的外部工具 TCP 指到期望的工件坐标系原点，点击“确认第一点”按钮。	第二个点与第一个点的连线即工件坐标系的 X 轴。
5	Jog 机器人，使标定好的外部工具 TCP 指到期望的工件坐标系 X 轴上一点，然后确认第二个点。	第二个点与第一个点的连线即工件坐标系的 X 轴。
6	Jog 机器人，使标定好的外部工具 TCP 指到期望的工件坐标系 XY 平面上一点，然后确认第三个点	也可以在期望的 Y 轴上选择一点，因为 Y 轴上的点也在 XY 平面上。



11.2.10.3 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 wobj0 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。

11.2.10.4 外部工具/工件使用说明

定义

为了减少默认工具工件的定义，默认工具 tool0 和默认工件 wobj0 是否手持根据用户选择的工具和工件确定：

1）当用户选择的工具坐标系如 tool1 为手持时，选择默认工件坐标系 wobj0 则自动为外部，且 wobj0 与用户坐标系重合；当用户选择的工具坐标系如 tool1 为外部时，选择默认工件坐标系 wobj0 则自动为手持，且 wobj0 与法兰坐标系重合；

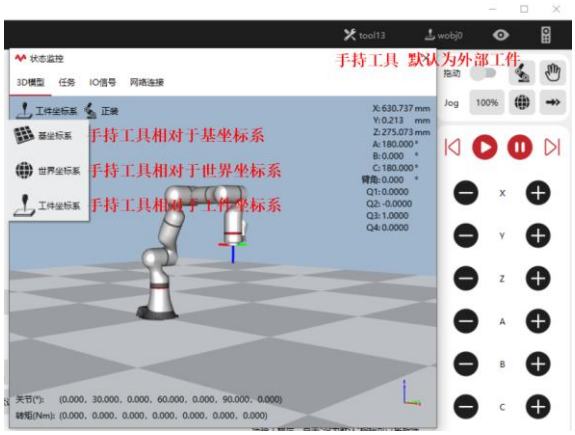
2）工件坐标系同理，当选择工件坐标系如 wobj1 为外部时，选择默认工具坐标系 tool0 为手持，且与法兰坐标系重合；选择工件坐标系如 wobj1 为手持时，选择默认工具坐标系 tool0 为外部，且与用户坐标系重合；

3）同时选择默认工具坐标系 tool0 和工件坐标系 wobj0 时，tool0 为手持且与法兰坐标系重合，wobj0 为外部且与用户坐标系重合。

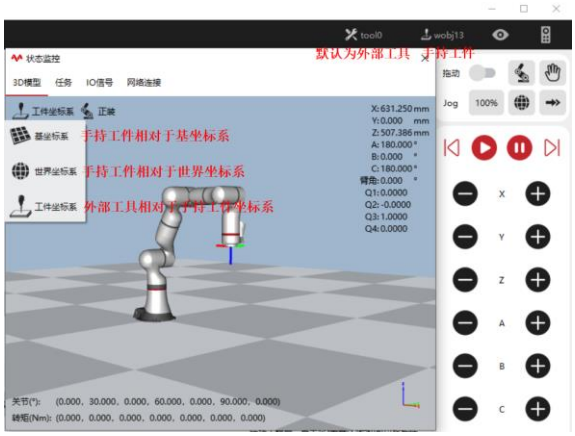
3D 界面显示

一般的，用户希望显示机械臂末端在不同坐标系下的位姿，因此：

1） 当使用手持工具时，3D 界面显示选择的工具坐标系相对于基坐标系/世界坐标系/工件坐标系的位姿；



2） 当使用外部工具时，3D 界面显示选择基坐标系/世界坐标系时，显示选择的（手持）工件坐标系相对于基坐标系/世界坐标系的位姿；3D 界面显示选择工件坐标系时，显示选择（外部）工具坐标系相对于（手持）工件坐标系的位姿。



用户坐标系

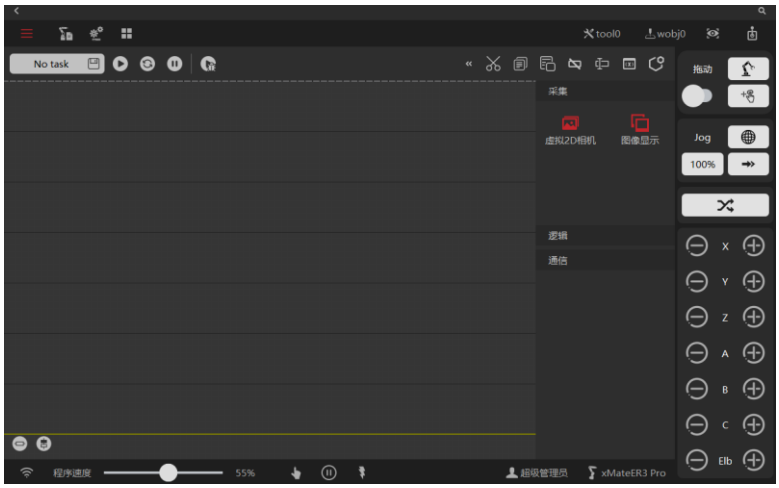
按照定义，用户坐标系也分为外部和手持，根据所配合的工件坐标系进行区分。例如，当使用外部工件时（对应的工具坐标系为手持），则配合使用的用户坐标系自动为外部，表示在世界坐标系下；当使用手持工件时（对应的工具坐标系为外部），则配合使用的用户坐标系自动为手持，表示在法兰坐标系下。

使用用户坐标系时需要仔细区分配合使用的工件坐标系，如果标定的用户坐标系表示在世界坐标系下，配合使用手持工件时则可能出现预想之外的错误。

11.2.11 视觉

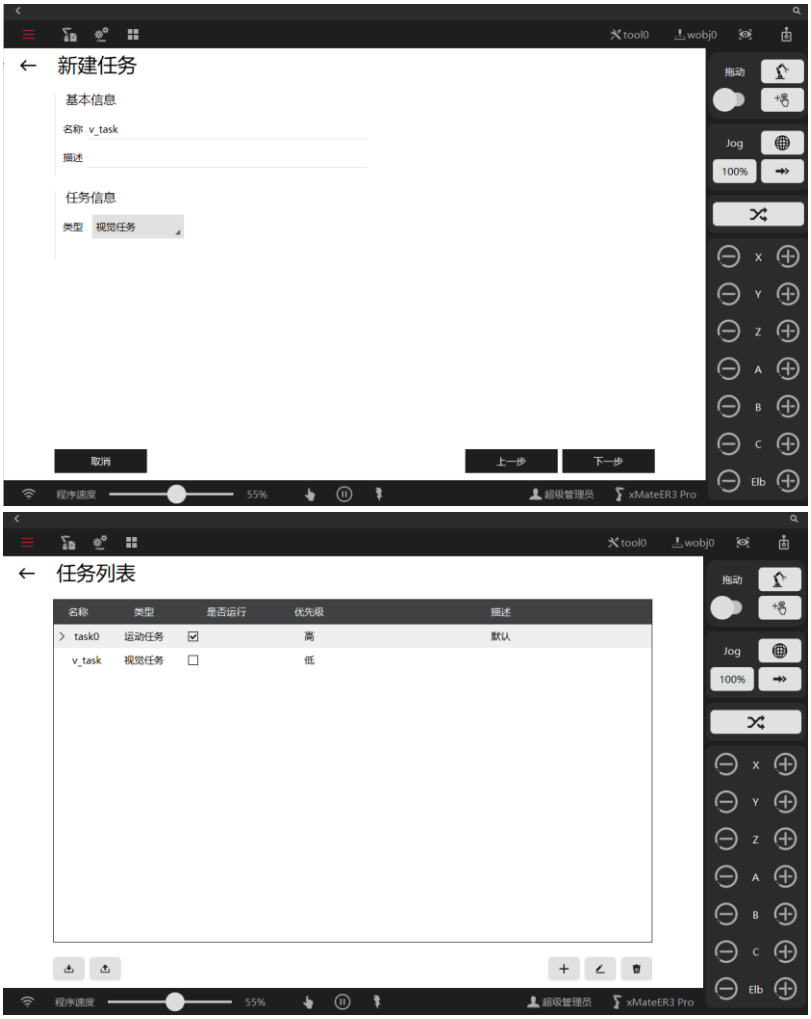
说明

视觉任务编程，可以视为跟 RL 运动任务编程同样级别。当新建工程后，可以在左侧单击“视觉”，打开视觉任务编辑界面。



视觉任务新建、打开、重命名

视觉任务也属于“任务”的一种，单击“任务”，可以查看视觉任务，并进行视觉任务的新建、打开、重命名。



更多关于视觉功能的使用，请参考手册《xVision 使用手册》。

11.2 RL 程序

11.2.1 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），xCore 系统使用 RL 语言作为 ROKAE 机器人的编程语言。

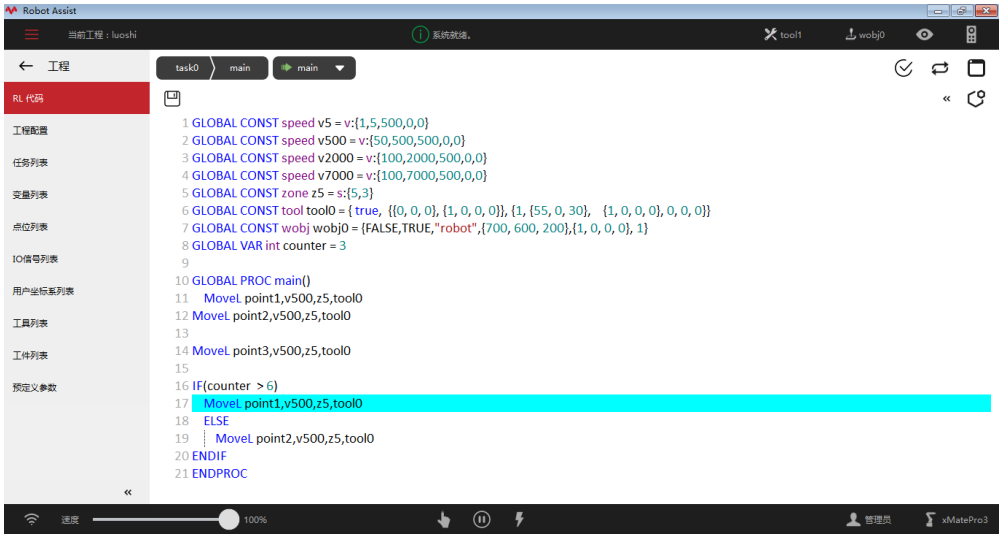
RL 语言为 ROKAE Robot Language 的缩写，即 ROKAE 机器人编程语言，用户可以通过示教器编写控制机器人的程序。

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 RLs 的基本结构和格式：



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 GLOBAL 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，若不显示声明变量的存储类型，RL 程序默认变量为可变量；
- int、robtarget、speed、zone、tool 是 RL 语言的专用变量类型；
- MoveJ 、MoveAbsj、MoveL 等是 RL 语言中一个标准的运动指令；
- “//” ”/**/”后面是注释内容；

11.2.2 程序结构

11.2.2.1 概述

说明

xCore 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四个层次：

a) 工程，即 Project，RL 程序的最大级别，配备机器人的各种默认参数，管理子对象，任务；

b) 任务，即 Task，包含若干个程序模块；

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，用户可在函数内按自己的需求调用机器人功能或者其他模块；

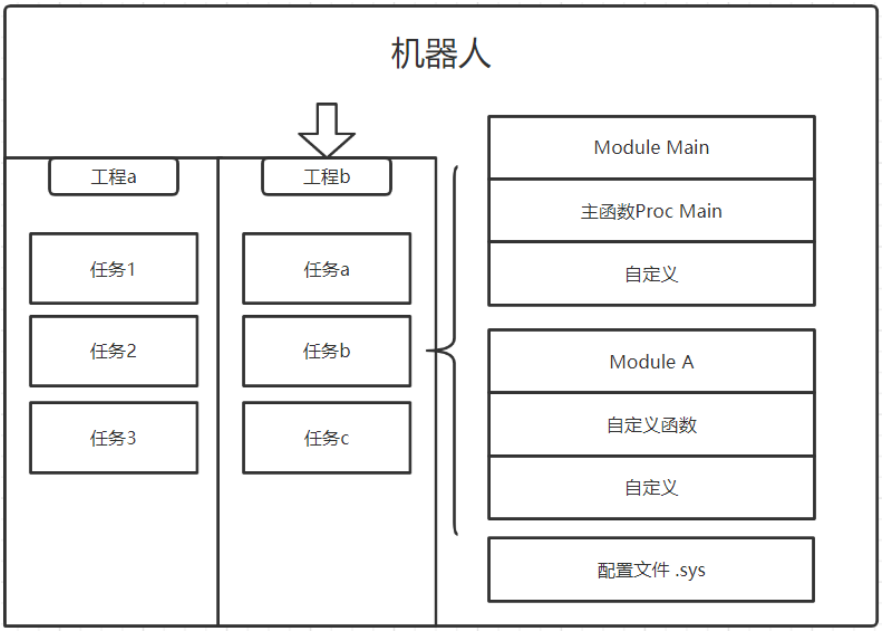
2. 一个工程中可以包含多个任务，原则上每个任务互相独立，仅靠提供的接口进行交互；

3. 一个任务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 GLOBAL PROC main，该 GLOBAL 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. 机器人一次只能选择一个工程执行。

工程、程序文件以及函数的相互关系如下图所示：



11.2.2.2 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。

模块的定义

模块的定义方式为：

```
PROC main()
...
ENDPROC
PROC test1()
...
ENDPROC
PROC test2()
...
ENDPROC
```

> **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。

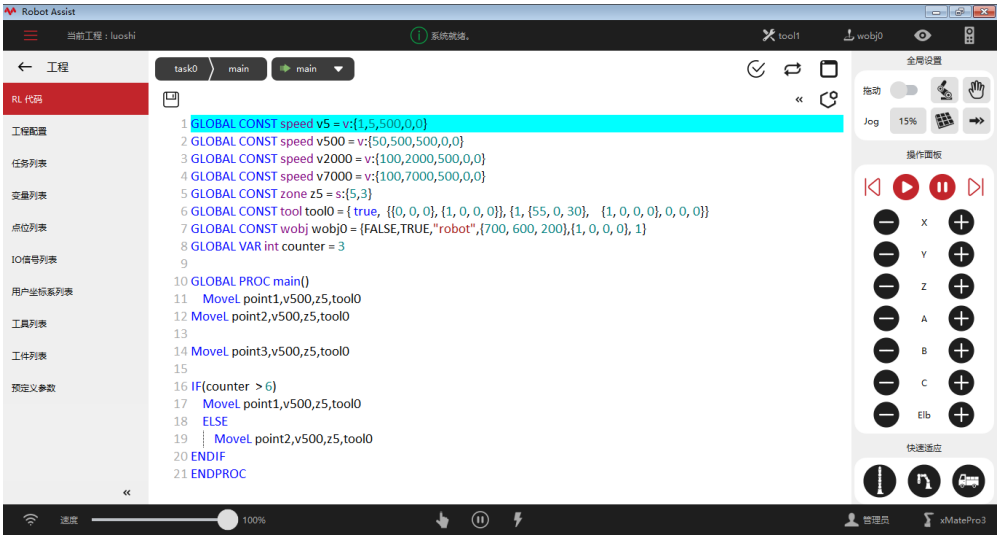
11.2.3 程序编辑

11.2.3.1 功能菜单

说明

为方便程序调试，xCore 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	点击 main，移动程序指针到 Main 函数，相当于程序复位。
程序指针到光标	点击 光标，把程序指针移动到光标所在行。
检查程序	执行程序指针到 Main 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。
插入指令	点击，可以插入运动指令和其他指令。
搜索程序	点击，对程序进行关键字搜索。
注释指令	点击，对选中代码行进行注释，支持多行注释。
向下移动代码行	点击，将选中的代码行下一行，支持多行同时下移。
向上移动代码行	点击，将选中的代码行上一行，支持多行同时上移。
粘贴整行	点击，将剪切或复制到的整行内容，插入光标所在行，
复制整行	点击，复制选中的代码行，支持多行同时复制。
剪切整行	点击，剪切选中的代码行，支持多行同时剪切。
撤销操作	点击，对上一步操作进行撤销。
恢复操作	点击，对上一步撤销的操作进行恢复。
循环模式	点击，选择循环或者只跑一次。
输出框	点击，显示打印信息和语法信息。

11.2.4 程序调试

11.2.4.1 程序指针

说明

程序指针指向程序已解析并运行的行号。

在 HMI 界面上，程序指针使用绿色小箭头表示（又称绿指针）。

11.2.4.2 运动指针

说明

运动指针表示机器人实际正在执行的指令；
在 HMI 界面上，运动指针使用红色箭头表示。

11.2.4.3 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，解释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，目前只有 Print 指令，逻辑判断指令，用户自定义函数不会打断前瞻机制，其余函数都会打断前瞻。

11.2.4.4 单步调试

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指令并在该指令完成后停止，便于确认每一个示教点、每一条指令是否符合要求。
单步运行状态又称为单步模式，与之对应的是连续模式，机器人可以在单步模式和连续模式之间随意进行切换。

使用限制

1. 连续模式程序自动执行，需要处理转弯区，可以有运动前瞻。

2. 单步模式直接执行指令，不处理转弯区，无运动前瞻。
3. 连续模式下，前瞻点够了才运动，到位后才可以继续解析指令。
4. 单步模式下，所有下一步信号由界面触发，不处理转弯区，不前瞻。
5. 单步模式下，运动过程中点击下一步，不响应。
6. 连续模式下，运动过程中的回调，按照前瞻逻辑响应。
7. 下一步可到任意行，执行字面意义。对于 RL 程序而言，只有"程序指令"，不区分运动指令和逻辑指令。
10. 连续运行，暂停在转弯区上，下一步均先回到当前转弯区所对应的目标点。

11.2.4.5 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。

11.2.4.6 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
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。

使用限制

移动程序指针存在如下限制：

1. 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，其他所有指令都不执行：

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；

2. 移动程序指针时，忽略流程控制指令的判断条件。

3. 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。

4. 移动程序指针只能移动到运动指令行。

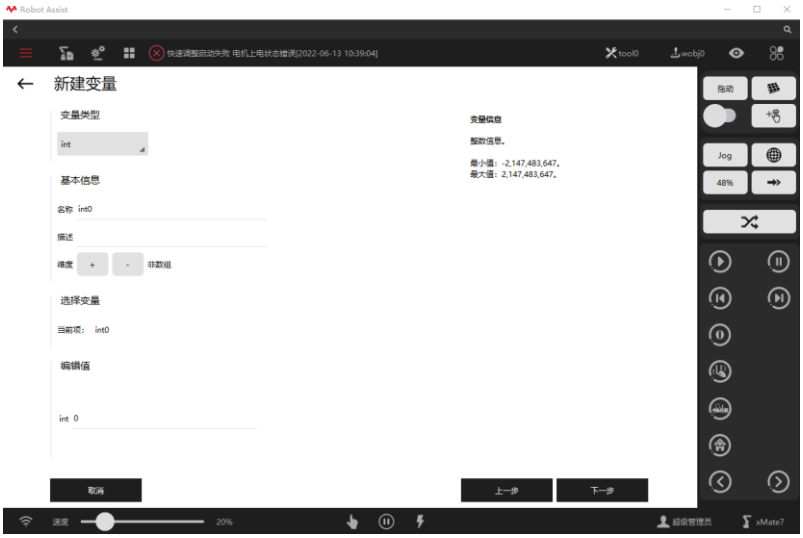
11.2.4.7 变量管理

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为（int/ byte/ bool/ double/ string/ robtarget/ jointtarget/ speed/ zone/ clock/ pose/fcboxvol/intnum tasks）。

说明

虽然所有类型的变量用户都可以在编程界面手动输入，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。
变量管理界面如下：



12 RL 编程指令

12.1 变量

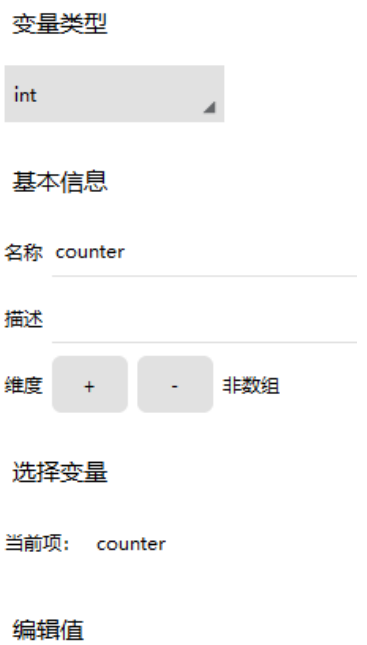
12.1.1 Int

说明

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，使用时请不要超过最大取值范围。

示例

例如，在变量列表中定义：



```
int counter = 4
```

表示定义了一个整型的变量类型的数据 counter，且初始值等于 4。

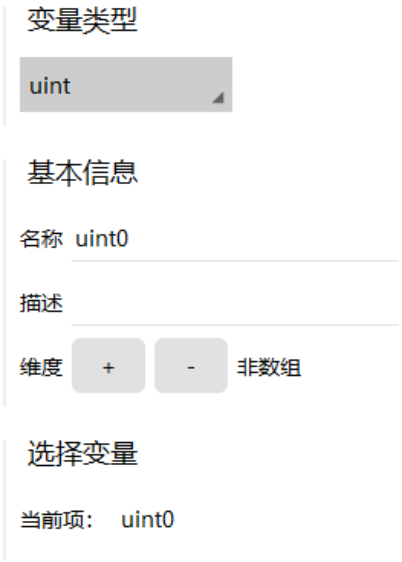
12.1.2 uint

说明

整型 uint 变量的取值范围是 $0 \sim 2^{32}-1$，使用时请不要超过最大取值范围。

示例

与有符号整型类似，在变量列表中定义：



12.1.3 Double

说明

浮点数使用 8 个字节存储，使用时请不要超出取值范围。

示例

例如，在变量列表中定义：

表示定义了一个浮点型的变量 time，且初始值等于 1.5。

12.1.4 Bool

说明

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。

示例

例如，在变量列表中定义：

```
bool true
```

表示定义了一个 bool 型的变量 ifClose，且初始值为 true。

12.1.5 String

说明

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。

示例

例如，在变量列表中定义：

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"。

12.1.6 Array

说明

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。

示例

例如，在变量列表中定义：

表示定义了一个二维数组，包含 16 个整型变量，将数组的第一行第 6 个元素赋值为 8。

> 提示
> 数组总长度不允许超过 1000。

12.1.7 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

例如，在变量列表中定义：

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。

12.1.8 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

例如，在变量列表中定义：

下面的例子显示了如何使用变量类型 clock：

例 1

```
ClkStart clock1
ClkStop clock1
interval=ClkRead(clock1)
ClkReset clock1
```

Interval(预先声明的 double 变量)读到的是 ClkStart 和 ClkStop 之间的时间间隔，单位是 s。

12.1.9 隐式类型转换

说明

目前，在变量列表中设置数据时，限制了数据类型，即不符合变量类型的值不能成功输入，从而避免类型隐式转化。

示例

例如，在定义整型变量 counter 时，只能输入整数，而不能输入小数。

12.1.10 confdata

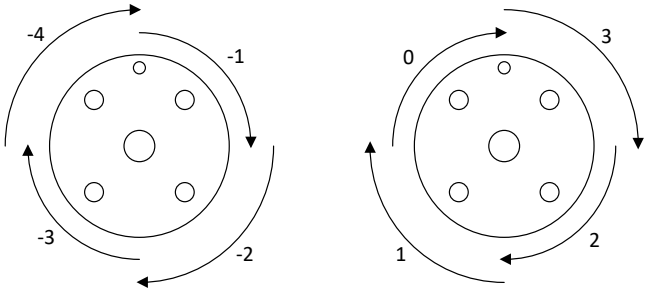
说明

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

对于带有冗余度的 7 轴机器人来讲，在臂角相同的情况下，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：左图为关节角度为负时的情形，右图为关节角度为正时的情形。对于机器人关节而言，逆时针转动角度增加，顺时针转动角度减小。在下图中，某关节顺时针转动，关节角度减小，则关节角度对应的 confdata 变化分别为-1->-2->-3->-4 和 3->2->1->0。



对于 xMate 来讲，需要 7 个参数来构成完整的 confdata，包括：

- cf1，记录 1 轴的象限数；
- cf2，记录 2 轴的象限数；
- cf3，记录 3 轴的象限数；
- cf3，记录 4 轴的象限数；
- cf5，记录 5 轴的象限数；
- cf6，记录 6 轴的象限数；
- cf7，记录 7 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。

Cf2

数据类型：int

2 轴角度对应的象限。

Cf3

数据类型：int

3 轴角度对应的象限。

Cf4

数据类型：int

4 轴角度对应的象限。

Cf5

数据类型：int

5 轴角度对应的象限。

Cf6

数据类型：int

6 轴角度对应的象限。

Cf7

数据类型：int

7 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 xMate 类型的带有冗余自由度的机器人来讲，在臂角保持不变的情况下，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……	6 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

12.1.11 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 Axis）

数据类型：double

robax 包含 7 个 double 型的成员，分别存储机器人 7 个关节的角度 Degree。

extax

外部轴位置（External Axis）

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

例如，在变量列表中定义：

上述语句定义了一个名为"jointtarget0"的关节空间的点，除五轴为 90 度外，机器人其他轴角度均为 0 度，第一个外部轴位置为 10 度或 10 毫米，取决于外部轴类型。

12.1.12 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，或者使用负载辨识功能由控制系统自动修改。

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 xCore 系统中负载被当做刚体来处理，用于描述负载的参数如下：

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

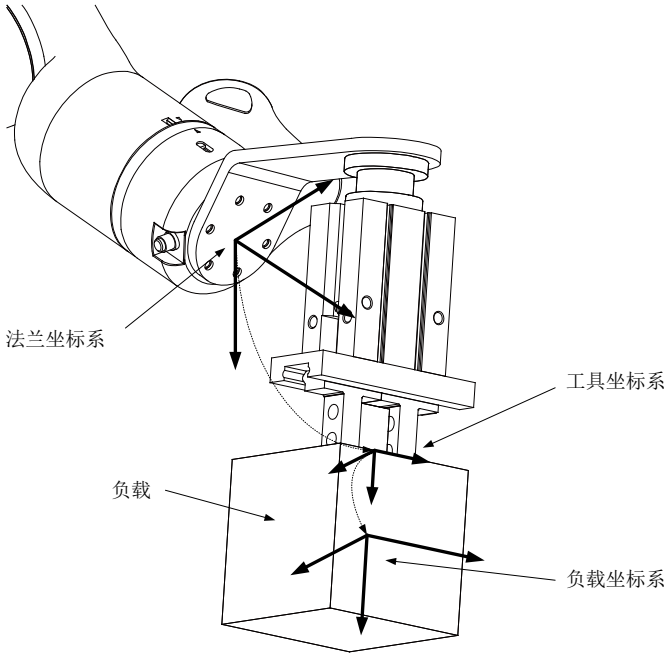
如果工具安装在机器人上，则 cogz 记录负载质心在工具坐标系下 Z 方向的偏移量；如果使用外部工具功能，则 cogz 记录被抓手夹持的负载质心在工件坐标系下 Z 方向的偏移量。

q1~q4

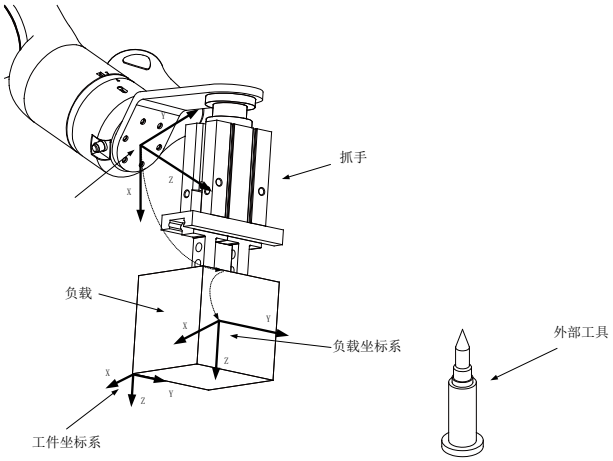
四元数，记录负载惯量主轴的方向。

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

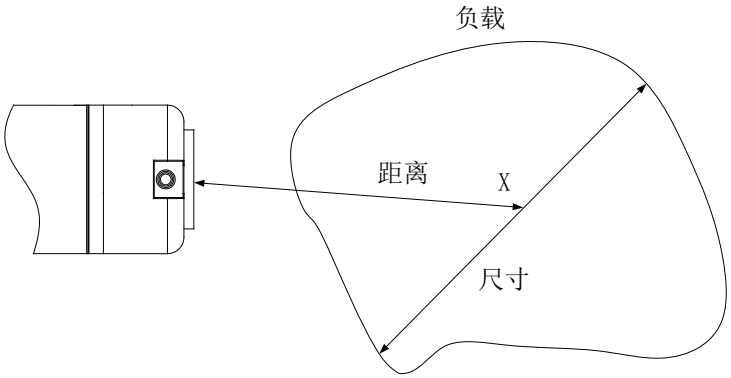
惯量 x

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 kgm2。

12.1.13 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:

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

12.1.14 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。

12.1.15 pose

说明

存储笛卡尔空间的位置和姿态。

定义

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。

12.1.16 robtarget

说明

存储三维空间的笛卡尔位置和姿态，用于 MoveJ、MoveL、MoveC 以及 MoveT 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

例如，在变量列表中定义：

定义了一个名为 p1 的笛卡尔位姿，其位置与姿态（四元数表示）如上所示，臂角角度为 10°，1,3,5,7 轴的角度均处于 0~90°之间，机器人对应第一组位型构型（详见 confdata），所有外部轴处于零位。

12.1.17 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"IO 信号列表"中进行定义，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

例 1

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

例 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 IO 信号列表中进行配置。

> 提示
>
> 1. 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。
> 2. 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

12.1.18 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，分别是关节速度百分比，TCP 线速度，空间旋转速度，外轴线速度，外轴角速度。

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 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。

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。

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。

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

在变量列表中定义：

定义了一个名为 speed0 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。

12.1.19 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。

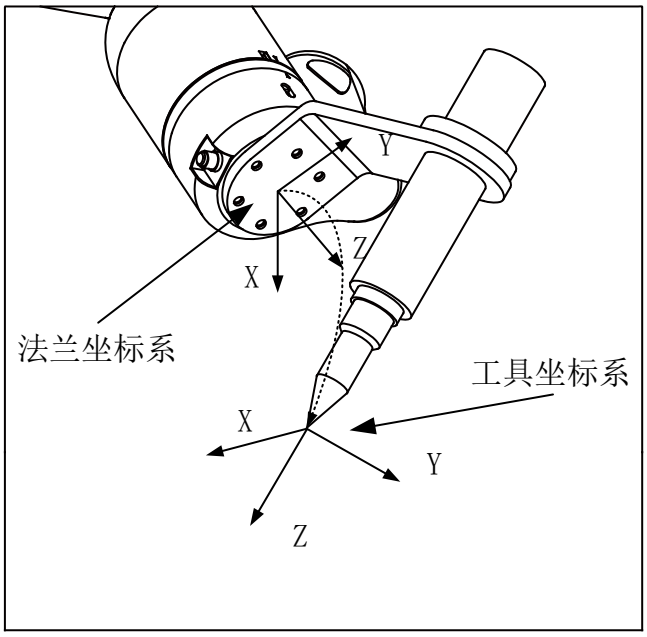
tframe

工具坐标系（Tool Frame）

数据类型：pose

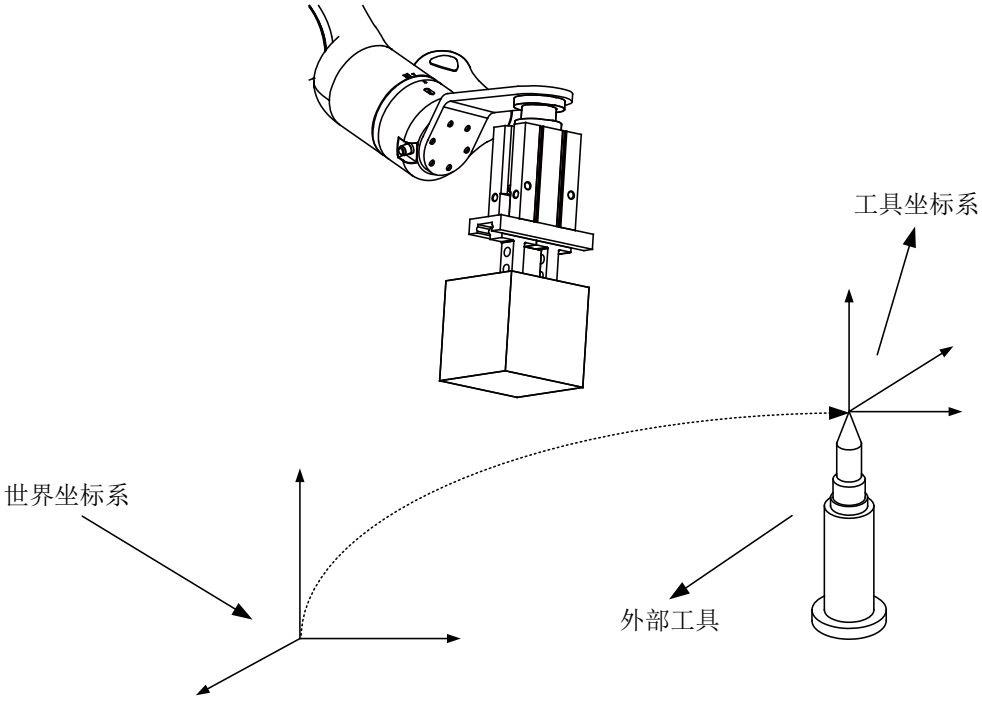
记录所用工具的工具坐标系，包括：

- 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

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。

12.1.20 trigdata

说明

trigdata 用于存储有关机器人移动期间触发事件的信息数据，包括触发条件和触发动作。触发条件，通常是机器人到达路径上的指定位置点；触发动作，具体形式可以是设置 IO、设置变量等。

trigdata 类型的变量不能通过赋值操作符进行定义，只能通过特定的 RL 指令来定义，因此每一个 trigdata 类型的变量内存储什么样的信息，取决于所使用的 Trig 指令，例如 TrigIO 等；最终在运动指令 TrigL、TrigC、TrigJ 中，以该数据作为参数使用。

示例

下面的例子显示了如何使用 trigdata：

例 1

```
VAR trigdata gripopen
TrigIO gripopen,0.5,do1,true
TrigL p1,v500,gripopen,fine,tool1
```

12.1.21 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作是 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作默认的工件坐标系 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

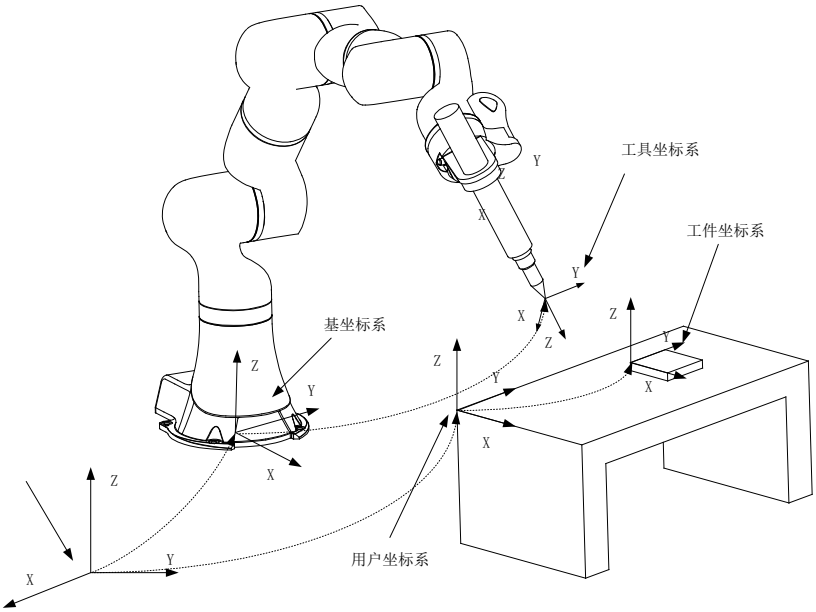
用户坐标系（User Frame）id

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。

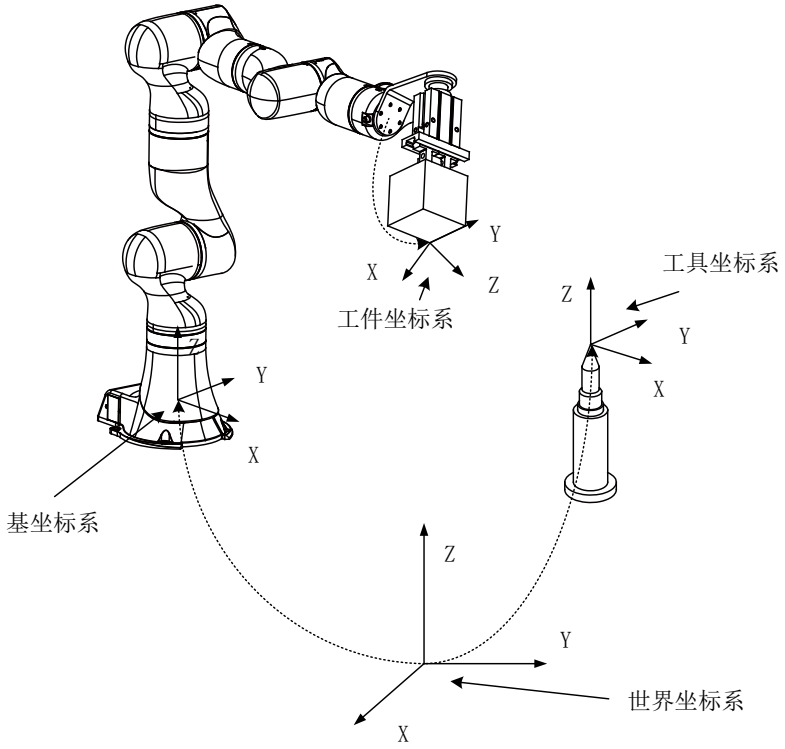
在使用普通工具时（非外部工具），坐标系的定义链为：

- 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

在变量列表中定义：

定义了一个名为 wobj2 的工件，其中的各项参数为：

- 该工件安装在机器人上；
- 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；
- 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；

用户坐标系 id 为 1。

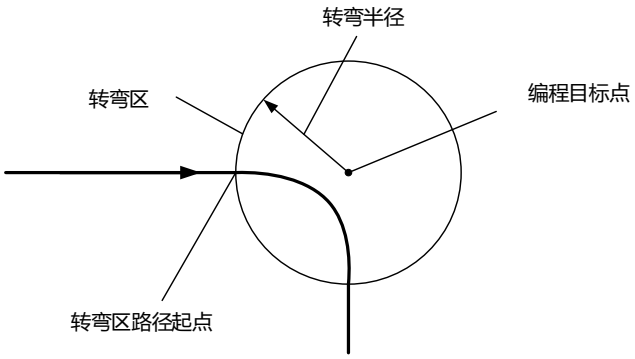
12.1.22 zone

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：



定义的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，所以该变量包括 distance 和 percent 两部分。

distance

笛卡尔空间转弯区大小

数据类型：double

用于 MoveL、MoveC 和 MoveT 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

例如，在变量列表中定义：

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，可能的原因有：

- 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；
- 前后两条轨迹纯旋转运动转轴反向，如前一条只有末端轴正转，后一条只有末端轴反转。

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。

12.1.23 torqueinfo

说明

用来描述机器人受到力和力矩信息

内部包括轴空间力矩信息以及笛卡尔空间力矩信息两个部分

定义

joint_torque

数据类型：轴空间力矩信息

cart_torque

数据类型：笛卡尔空间力矩信息

joint_torque.measure_torque

数据类型：double 数组

轴空间测量力信息，力传感器测量到的各轴所受力矩

joint_torque.external_torque

数据类型：double 数组

轴空间外部力信息，控制器根据机器人模型和测量力计算出的各轴所受力矩信息

cart_torque.m_force

数据类型：double 数组

笛卡尔空间(xyz)各个方向受到的力的大小

cart_torque.m_torque

数据类型：double 数组

笛卡尔空间(xyz)各个方向受到的力矩的大小

基本示例

在变量列表中定义：

下面的例子显示了如何使用变量类型 torqueinfo：

例 1

```
TorqueInfo tmp_info = GetEndtoolTorque(tool1, wobj1)
//获得在 tool1 wobj1 条件下机器人末端工具的力矩信息结构体
…
print(tmp_info.joint_torque.measure_torque)
print(tmp_info.joint_torque.external_torque)
//打印各个轴的测量力和外部力
…
print(tmp_info.cart_torque.m_torque)
//打印笛卡尔空间力矩
…
print(tmp_info.cart_torque.m_force[0])
print(tmp_info.cart_torque.m_torque[0])
//打印 X 方向的力和力矩信息
```

12.1.24 SocketServer

说明

在控制器上建立 Socket TCP 服务器，用来监听外部设备以客户端身份发起的连接。该服务器仅用来监听连接请求，支持多连接。当连接建立之后，会生成一个新的 SocketConn 对象来进行通信。

> 提示
>
> 1、 系统资源申请和释放需要一定时间来处理，不要频繁创建（OpenDev）和销毁（CloseDev）服务器资源，创建和销毁资源期间推荐保留至少 500ms 时间间隔，否则系统资源会超负荷导致出现问题。
> 2、 该语句仅仅创建了一个服务器资源对象，但是并没有完成服务器创建。服务器需要通过 OpenDev 和 SocketAccept 来进入监听状态。
> 3、 支持服务器一对多连接。

定义

ip

数据类型：string

控制系统使用 ip 参数来匹配网卡，并使用对应的网卡进行网络监听。该参数如果设置为"0.0.0.0"则表示监听所有网卡的连接。通常设置为"0.0.0.0"即可。

port

数据类型：int

监听端口。外部客户端发起连接时，需要指定服务器端口为此设置的数值。

name

数据类型：string

RL 程序中使用，此服务器的唯一标识。工程内唯一，多任务间可共享使用，不可命名冲突。

基本示例

例 1

```
SocketServer ss = {"192.168.0.160", 8090, "svr"}  //仅监听 ip 配置为 192.168.0.160 的网卡
SocketConn conn = SocketAccept( "svr")
```

例 2

```
SocketServer ss = {"0.0.0.0", 8090, "svr"} //监听机器人所有网卡
SocketConn conn = SocketAccept( "svr")
```

12.1.25 SocketConn

说明

Socket TCP 连接对象，可用于对外部设备的通信。分为两种类型：

1） 机器人作为客户端，通过该对象发起向外部设备 TCP 服务器的连接和通信。

2） 机器人作为服务器，当外部设备的 TCP 客户端发起连接之后，产生的与对端设备的通信连接。当外部不同设备发起多个 TCP 客户端连接，则对每个连接产生一个连接。

定义

ip

数据类型：string

机器人作为客户端使用时，该参数表示外部设备服务器的 ip。

机器人作为服务器使用时，当外部设备建立连接后，该参数表示外部客户端的 ip。

port

数据类型：int

监听端口。机器人发起连接时，需要指定外部设备服务器端口。

name

数据类型：string

RL 程序中使用，此连接的唯一标识。工程内唯一，多任务间可共享使用，连接之间以及连接与服务器的命名不能有冲突。

cache

数据类型：int

缓存大小，表示接收数据最多可以缓存的数量。可缺省，默认值为 1。

suffix

数据类型：string

结束符，表示一条消息的结束。可缺省，默认值为”\r”。

attr

数据类型：string

连接属性。

"incoming"： 本地为服务器，由对端客户端连入。ip、port 标识客户端信息。

"outgoing"： 本地为客户端，连接外部服务器。ip、port 为连接对端服务器的信息。

""及其他： 不可用连接，表示还没有打开连接或者查找到未建立过的连接。

state

数据类型：string

当前通信连接状态。Closed：连接已关闭；establish：连接已建立正常使用中。

> 提示
>
> 1、 做客户端使用时，ip、port 信息应该由用户编程设置。做服务端使用时，ip、port 应该从 accept 指令自动获取。连接建立以后不要轻易修改这两个值，除非您非常清楚这两项的使用，避免程序逻辑使用出错。
> 2、 suffix 随时可以重新设置，在下一次 read 之前可以生效。谨慎使用该功能，否则会造成通信数据错误。suffix 应该在进行通信之前设置好，就不要再更改了。

基本示例

例 1

```
//服务器 ip 为"192.168.0.202"，端口号 8090，此连接名称为“clt”，缓存缺省为 1，后缀缺省为”\r”
SocketConn scnn1 = {"192.168.0.202", 8090, "clt"}
```

例 2

```
//服务器 ip 为"192.168.0.203"，端口号 8091，此连接名称为“clt1”，缓存为 2，后缀缺省为”\ r”
```

```
SocketConn scnn2 = {"192.168.0.203", 8091, "clt1", 2}
```

例 3

```
//服务器 ip 为"192.168.0.204"，端口号 8092，此连接名称为"clt2"，缓存为 2，后缀为"\n"
SocketConn scnn3 = {"192.168.0.204", 8092, "clt2", 2, "\n"}
```

例 4

```
//作为服务器使用，外部设备建立连接以后
//服务器 ip 为"192.168.0.204"，端口号 8092，此连接名称为"clt2"，缓存为 2，后缀为"\n"
SocketConn conn = SocketAccept( "svr1")
Print(conn.ip)  //外部设备的 ip
Print(conn.port)  //外部设备产生该连接所使用的端口
Print(conn.cache)  //接收消息缓冲区队列
Print(conn.suffix)  //收发后缀
```

12.2 函数

12.2.1 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。

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。

12.3 指令

12.3.1 变量类型转换

12.3.1.1.1 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

例 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

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

12.3.1.1.2 StrToDouble

说明

StrToDouble 用于将一个字符串转换为浮点数据。

返回值

数据类型： double

输入字符串表示的浮点变量。

定义

StrToDouble (ConStr)

ConStr

数据类型： string

表示要被转换的字符串。

示例

例 1

```
VAR double db_data = StrToDouble("-10")    // -10.0
db_data = StrToDouble("45.678")            // 45.678
```

使用限制

- 输入非十进制浮点数报错。

12.3.1.1.3 StrToInt

说明

StrToInt 用于将一个字符串转换为整型数据。

返回值

数据类型： int

输入字符串表示的整型变量。

定义

StrToInt (ConStr)

ConStr

数据类型： string

表示要被转换的十进制数字字符串。

示例

例 1

```
VAR int int_data = StrToInt("-10")      // -10
int_data = StrToInt("45678")            // 45678
```

使用限制

- 转换的变量范围是 -2^{31}~2^{31}，超出范围则报错。
- 若输入不为十进制，报错。

12.3.1.1.4 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

例 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”。

12.3.1.1.5 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

数据类型：string

表示要被转换的十进制数据，用 0-9 表示。

使用限制

- 数据范围从 0~2147483647 或 0~7fffffff。

12.3.1.1.6 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

数据类型：double

要转换的 double 变量。

12.3.1.1.7 DoubleToStr

说明

将 double 类型的变量转换为 string。

参数

DoubleToStr(Val, Dec)

Val

数据类型：double

要转换的 double 变量。

Dec

数据类型：int

要保留的小数点位数。

使用限制

➢ 小数点位数最多 15 位。

12.3.1.1.8 HexToDec

说明

用于将 16 进制的数转换成 10 进制数。

返回值

返回转换得到的 10 进制整型数据，用 0-9 表示。

参数

HexToDec(str)

str

数据类型：string

要转换的 16 进制数据，用 0-9，a-f，A-F 表示。

使用限制

➢ 数据范围从 0~2147483647 或 0~7fffffff。

12.3.1.1.9 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

数据类型：int 或 int 数组

要被转换的整数变量或数组。

使用限制

- 数据范围从-2147483647~2147483647。

12.3.1.1.10 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

数据类型：int

要被转换的整数变量。

使用限制

- 数据范围从-2147483647~2147483647。

12.3.2 运动指令

12.3.2.1 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。

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。

示例

以下是一些 MoveAbsJ 用例：

例 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。

例 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。

12.3.2.2 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。

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。

示例

以下是一些 MoveJ 用例：

例 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。

例 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。

12.3.2.3 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、位置和臂角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，请检查转动速度设置是否合理。

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。

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。

示例

以下是一些 MoveL 用例：

例 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。

例 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。

12.3.2.4 MoveC

说明

MoveC （ Move Circle） 用于将工具中心点 TCP 沿圆弧经过中间辅助点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，辅助点的姿态不影响圆弧运动过程。

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，MoveC 指令最终的运动时间取决于姿态、位置和臂角变化时间较长的那一个。 因此在某些特定轨迹（例如位移很小而姿态变化很大）下，如果机器人运动速度明显过慢或过快，请检查转动速度设置是否合理。

参数

MoveC AuxPoint， ToPoint， Speed， Zone， Tool， [Wobj]

其中，带[ ]的参数为可选参数，可不写。

AuxPoint

辅助点（Auxiliary Point）

数据类型： robtarget

在笛卡尔空间描述的辅助点位置，用于确定圆弧大小和运动方向，该点的姿态不会影响最终轨迹的执行。

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 MoveC 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。

示例

以下是一些 MoveC 用例：

例 1

MoveC p10， p20， v1000， z50， tool0

机器人使用工具 tool0， TCP 以 v1000 的速度沿圆弧经过 p10 点运动到 p20 定义的目标位置，转弯区大小为 50 mm。

例 2

MoveC auxpoint， endpoint， v500， z50， gripper， wobj2

机器人使用工具 gripper，在工件坐标系 wobj2 下， TCP 以 v500 的速度沿圆弧经过 auxpoint 点运动到 endpoint 定义的目标位置，转弯区大小为 50 mm。

12.3.2.5 MoveT

说明

MoveT (Move trochoid)用于将工具中心点 TCP 按照指定步进长度，经过辅助点画次摆线旋转步进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，辅助点的姿态不影响螺旋运动过程。

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，MoveT 指令最终的运

动时间取决于姿态、位置和臂角变化时间较长的那一个。 因此在某些特定轨迹（例如位移很小而姿态变化很大）下，如果机器人运动速度明显过慢或过快，请检查转动速度设置是否合理。

参数

MoveT AuxPoint, ToPoint, Step, Speed, Zone, Tool, [Wobj]

其中，带[ ]的参数为可选参数，可不写。

AuxPoint

辅助点（Auxiliary Point）

数据类型： robtarget

在笛卡尔空间描述的辅助点位置，用于确定圆弧大小和运动方向，该点的姿态不会影响最终轨迹的执行。

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。

Step

步进长度

数据类型： double

次摆线前进的步进长度，单位 mm

Speed

运动速度（Move Speed）

数据类型： speed

用于指定机器人执行 MoveT 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。

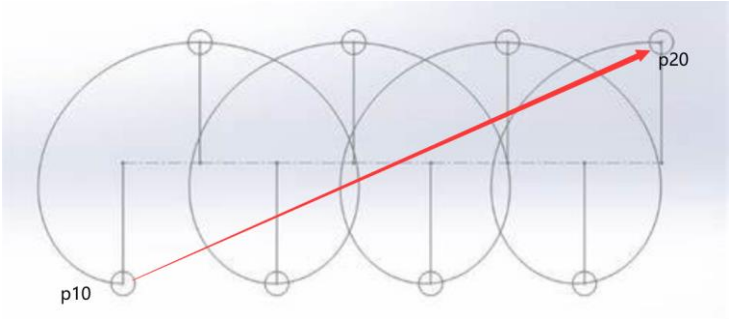
示例

以下是一些 MoveT 用例：

例 1

MoveT p10, p20, 50, v1000, z50, tool0

机器人使用工具 tool0， TCP 以 v1000 的速度沿圆弧经过 p10 点画次摆线，每次摆线步进 50mm，最后运动到 p20 定义的目标位置，转弯区大小为 50 mm。



12.3.2.6 SearchL

说明

SearchL（Search Liner） 用于沿直线移动工具中心点时搜索位置。

在移动时，机器人会监控一个数字输入信号（DI）。当监测的信号状态符合触发模式时，机器人立即读取当前位置。

当由机械臂固定的工具为用于表面探测的探针时，通常可使用该指令。使用 SearchL 指令，可获得工件的概略坐标。

本指令仅可用于运动任务。

参数

SearchL [action,] di, [trigger_mode,] save_rob, target_rob, Speed, Tool [,Wobj]

其中，带[ ]的参数为可选参数，可不写。

action

触发 DI 后的行为

数据类型：关键字

无参数：不停止

\Stop：快速停止，可能导致机器人偏离路径，但停止速度更快，仅当速度小于 v100 时可用

\PStop：规划停止，机器人会在预定轨迹上停止，不限制速度

di

数据类型：DI 信号

SearchL 指令触发特定行为的信号，使用用户自定义 DI 信号

trigger_mode

DI 信号触发模式

数据类型：关键字

无参数：默认为上升沿触发

\Flanks：边缘触发（上升沿或下降沿）

\Posflank：上升沿触发

\Negflank：下降沿触发

\Highlevel：高电平触发

\Lowlevel：低电平触发

save_rob

数据类型： robtarget

存储机器人触发信号时位置数据的点位

target_rob

数据类型： robtarget

直线运动的目标点位

Speed

运动速度（Move Speed）

数据类型： speed

用于指定机器人执行 Search 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

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。

示例

以下是一些 SearchL 用例：

例 1

SearchL di0, save_rob, target_rob, v500, tool0

机器人使用工具 tool0， TCP 以 v500 的速度朝 target_rob 直线运动，如果运动过程中 di0 跳变为高电平，将信号跳变时机器人的坐标信息记录在 save_rob。

例 2

SearchL \PStop, di0, \Lowlevel, save_rob, target_rob, v500, tool0

机器人使用工具 tool0， TCP 以 v500 的速度朝 target_rob 直线运动，如果运动过程中 di0 为低电平，机器人将立即规划停止，并在检测到信号为低电平时将机器人的坐标信息记录

save_rob。

12.3.2.7 SearchC

说明

SearchC（Search Circle） 用于沿圆周移动工具中心点时搜索位置。

在移动时，机器人会监控一个数字输入信号（DI）。当监测的信号状态符合触发模式时，机器人立即读取当前位置。

当由机械臂固定的工具为用于表面探测的探针时，通常可使用该指令。使用 SearchC 指令，可获得工件的概略坐标。

本指令仅可用于运动任务。

参数

SearchC [action,] di, [trigger_mode,] save_rob, aux_rob, target_rob, Speed, Tool [,Wobj]

其中，带[ ]的参数为可选参数，可不写。

action

触发 DI 后的行为

数据类型：关键字

无参数：不停止

\Stop：快速停止，可能导致机器人偏离路径，但停止速度更快，仅当速度小于 v100 时可用

\PStop：规划停止，机器人会在预定轨迹上停止，不限制速度

di

数据类型： DI 信号

SearchC 指令触发特定行为的信号，使用用户自定义 DI 信号

trigger_mode

DI 信号触发模式

数据类型：关键字

无参数：默认为上升沿触发

\Flanks：边缘触发（上升沿或下降沿）

\Posflank：上升沿触发

\Negflank：下降沿触发

\Highlevel：高电平触发

\Lowlevel：低电平触发

save_rob

数据类型： robtarget

存储机器人触发信号时位置数据的点位

aux_rob

数据类型： robtarget

圆周运动的辅助点

target_rob

数据类型： robtarget

圆周运动的目标点位

Speed

运动速度（Move Speed）

数据类型： speed

用于指定机器人执行 Search 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

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。

示例

以下是一些 SearchC 用例：

例 1

SearchC di0, save_rob, aux_rob, target_rob, v500, tool0

机器人使用工具 tool0， TCP 以 v500 的速度经过辅助点 aux_rob 朝 target_rob 圆周运动，如果运动过程中 di0 跳变为高电平，将信号跳变时机器人的坐标信息记录在 save_rob。

例 2

SearchC \PStop, di0, \Flanks, save_rob, target_rob, v500, tool0

机器人使用工具 tool0， TCP 以 v500 的速度经过辅助点 aux_rob 朝 target_rob 直线运动，如果运动过程中 di0 由低电平变为高电平，或高电平变为低电平，机器人将立即规划停止，并在信号跳变时将机器人的坐标信息记录在 save_rob。

12.3.3 Trigger 指令

12.3.3.1 TrigIO

说明

TriggIO 用来将一个 trigdata 设置为在运动过程中输出 I/O 的触发器，支持数字输出 DO 和数字

组输出 GO。

定义

TrigIO TrigData,Distance,RefStart,SignalName,Value

参数：

TrigData 数据类型 trigdata，用于存储本条 TrigIO 设置的触发数据的变量；

Distance 数据类型 double，非负数(负数当成 0 处理)，定义触发事件在路径上的位置偏移，至于该位置偏移是相对路径的起点 or 终点，由 RefStart 定义；

RefStart 数据类型 bool，定义触发位置相对起点(true)还是终点(false)；

SignalName 数据类型 signaldo 或 signalgo，与本次定义的 IO 事件关联的数字输出或者数字组输出的信号名称，该信号必须是已经正确设置的输出信号；//add

Value 数据类型：bool 或 int，定义触发 IO 事件时，指定输出信号的目标值，给定值的数据类型应该与 SignalName 类型相匹配。

示例

例 1

参考 TrigL 的示例；

12.3.3.2 TrigReg

说明

TrigReg 用来将一个 trigdata 设置为在运动过程中修改寄存器值；寄存器类型支持 int16,bool,float,bit 四种。

定义

TrigReg TrigData,Distance,RefStart,RegName,Value

参数：

TrigData 数据类型 trigdata，用于存储本条 TrigIO 设置的触发数据的变量；

Distance 数据类型 double，非负数(负数当成 0 处理)，定义触发事件在路径上的位置偏移，至于该位置偏移是相对路径的起点 or 终点，由 RefStart 定义；

RefStart 数据类型 bool，定义触发位置相对起点(true)还是终点(false)；

RegName 寄存器名字，无数据类型。注意：寄存器无法在 RL 中新建，需要用户在"机器人-通信-寄存器"页面新建；

Value 数据类型：需支持 int16、bool、float、bit，定义了触发寄存器修改事件时，指定寄存器的目标值，给定值的数据类型应该与 RegName 类型相匹配；如果用户给的值与寄存器类型不符，进行强制类型转换；。

示例

例 1

参考 TrigL 的示例；

12.3.3.3 TrigL

说明

TrigL 与 MoveL 类似，都是执行空间直线运动的指令，区别是 TrigL 可以在运动过程中的若干指定位置执行预定义的操作，其他参数的数量与含义二者之间并无差异。

定义

TrigL ToPoint,Speed,Trigger,Zone,Tool,[Wobj]

参数：

ToPoint 目标位姿(To Point),类型:robtarget，笛卡尔空间描述的目标位姿；

Speed 运动速度，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；

Trigger 触发条件和动作，类型：trigdata；trigdata 必须是使用 TrigX 指令处理过之后的 trigdata，否则运动到该行时编译器会给出错误提示；

Zone 转弯区，类型：zone，用来定义当前轨迹的转弯区大小；

Tool 工具，类型：tool；

[Wobj] 工件，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。

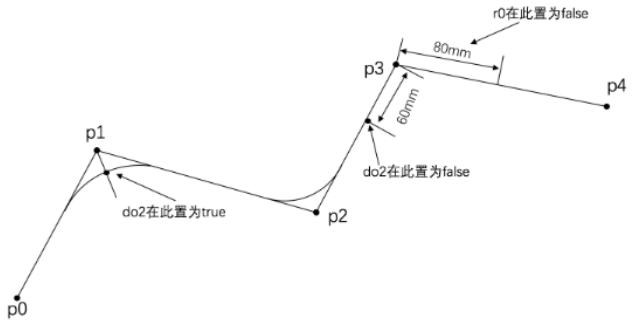
示例

例 1

```
VAR trigdata tc1
VAR trigdata tc2
VAR trigdata tc3
...
//设置 tc1、tc2、tc3
TrigIO tc1,0,true,do2,true
TrigIO tc2,60,false,do2,false
TrigReg tc3,80,true,r0,false    //r0 为 bool 类型的寄存器
...
//运动
MoveL p1,v500,z50,tool1
TrigL p2,v500,tc1,z50,tool1
```

TrigL p3,v500,tc2,fine,tool1

TrigL p4,v500,tc3,fine,tool1



12.3.3.4 TrigC

说明

TriggC 与 MoveC 类似，都是执行圆弧运动的指令，区别是 TriggC 可以在运动过程中的若干指定位置执行预定义的操作，其他参数的数量与含义二者之间并无差异。

定义

TrigC AuxPoint,ToPoint,Speed,Trigger,Zone,Tool,[Wobj]

参数：

AuxPoint 辅助点(Auxiliary Point)，类型:robtarget，笛卡尔空间描述的目标位姿；

ToPoint 目标位姿(To Point),类型:robtarget，笛卡尔空间描述的目标位姿；

Speed 运动速度，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；

Trigger 触发条件和动作，类型：trigdata；trigdata 必须是使用 TrigX 指令处理过之后的 trigdata，否则运动到该行时编译器会给出错误提示；

Zone 转弯区，类型：zone，用来定义当前轨迹的转弯区大小；

Tool 工具，类型：tool；

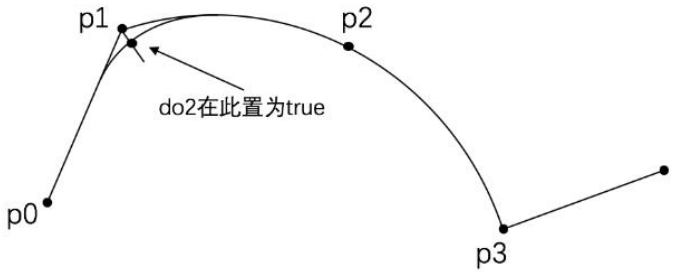
[Wobj] 工件，类型：wobj；执行该轨迹时使用的工件。当工具安装在机器人上时，该参数可以忽略; 当使用外部工具时，必须要指定该参数，且机器人将使用 wobj 中存储的数据来计算运动速度和转弯区大小。

示例

例 1

```
VAR trigdata tc1
...
//设置 tc1
```

```
TrigIO tc1,0,true,do2,true
...
//运动
MoveL p1,v500,z50,tool1
TrigC p2,p3,v500,tc1,fine,tool1
```



12.3.4 力控指令

12.3.4.1 CalibSensorError

说明

清除末端六维力测量值，以当前测量值作为零点。

定义

CalibSensorError

没有参数，直接使用。

示例

例 1

```
FcInit Tool1, Wobj0, 0
FcStart
SetSensorUseType 1
CalibSensorError
```

设置六维力测量值的使用方式为软件清零的方式，而后清除零点。

使用限制

- SetSensorUseType 1 后，也就是设置六维力测量值的使用方式为软件清零的方式后。方可调用此接口，不满足此限制，末端六维力测量值清除不成功。

12.3.4.2 FcInit

说明

用于力控开启前的一些初始化工作，如设置工件、工具和力控坐标系。

定义

FcInit Tool, Wobj, ForceFrameRef

Tool

数据类型：pose

力控所使用的工具，力控坐标系的原点是该工具的 TCP（姿态与第三个参数中所选择的坐标系姿态相同）。需要注意的是，所有使用的转接法兰都需要包含在工具的定义中。

Wobj

数据类型：pose

力控所使用的工件，很多力控功能的定义是相对于工件坐标系来的，例如力控坐标系的姿态、搜索模式和终止条件等。该参数默认为 Wobj0。

ForceFrameRef

数据类型：int

定义力控坐标系相对于哪个坐标系定义，支持

0：世界坐标系

1：工件坐标系

2：工具坐标系

默认值为世界坐标系（0）。

示例

例 1

FcInit Tool1, Wobj0, 0

力控初始化，并且定义力控所使用的工具 tool1 和工件 wobj0，以及力控坐标系相对于世界坐标系定义。

使用限制

- FcInit 和 FcStop 之间不允许再次调用 FcInit。

12.3.4.3 SetControlType

说明

设置阻抗控制类型。

定义

SetControlType ctrl_type

ctrl_type

数据类型：int

阻抗控制类型，支持

0：关节阻抗

1：笛卡尔阻抗

示例

例 1

FcInit Tool1, Wobj0, 0
SetControlType 0
FcInit 之后设置阻抗控制模式为关节阻抗。

使用限制

- FcInit 之后，FcStart 之前，不满足此限制，阻抗类型将无法设置成功。

12.3.4.4 SetSensorUseType

说明

设置末端六维力测量值的使用方式。

定义

SetSensorUseType sensor_use_type

sensor_use_type

数据类型：int

传感器使用方式，支持：

0：动态补偿

1：软件清零

示例

例 1

FcInit Tool1, Wobj0, 0
FcStart
SetSensorUseType 0
设置六维力测量值的使用方式为动态补偿，也即无需标零，直接使用。

使用限制

- FcInit 之后，FcStop 之前，方可调用此接口，不满足此限制，末端六维力测量值使用方式设置不成功。

12.3.4.5 SetCartNSStiff

说明

设置零空间阻抗刚度。

定义

SetCartNSStiff cart_ns_stiff

cart_ns_stiff

数据类型：double

笛卡尔零空间阻抗刚度，取值范围（0~4），单位：N.m/rad。

示例

例 1

FcInit Tool1, Wobj0, 0
SetControlType 1
SetCartNSStiff 2
设置阻抗控制模式为笛卡尔阻抗，零空间刚度设置为 2。

使用限制

- SetControlType 1 之后，也即设置阻抗控制类型为笛卡尔阻抗之后，方可调用此接口，不满足此限制，零空间阻抗参数将无法设置成功。

12.3.4.6 SetJntCtrlStiffVec

说明

设置关节阻抗刚度。

定义

SetJntCtrlStiffVec jnt1_stiff，jnt2_stiff，jnt3_stiff，jnt4_stiff，jnt5_stiff， jnt6_stiff，jnt7_stiff

jnt1_stiff

数据类型：double

关节 1 阻抗刚度，取值范围（0~1500），单位：Nm/rad。

Jnt2_stiff

数据类型：double

关节 2 阻抗刚度，取值范围（0~1500），单位：Nm/rad。

Jnt3_stiff

数据类型：double

关节 3 阻抗刚度，取值范围（0~1500），单位：Nm/rad。

Jnt4_stiff

数据类型：double

关节 4 阻抗刚度，取值范围（0~1500），单位：Nm/rad。

Jnt5_stiff

数据类型：double

关节 5 阻抗刚度，取值范围（0~100），单位：Nm/rad。

Jnt6_stiff

数据类型：double

关节 6 阻抗刚度，取值范围（0~100），单位：Nm/rad。

Jnt7_stiff

数据类型：double

关节 7 阻抗刚度，取值范围（0~100），单位：Nm/rad。

示例

例 1

FcInit Tool1, Wobj0, 0
SetControlType 0
SetJntCtrlStiffVec 1500,1500, 1500,1500,100,100,100
设置阻抗控制模式为关节阻抗，1~7 关节的阻抗刚度分别设置为 1500、1500、1500、1500、100、100、100。

使用限制

- SetControlType 0 之后，也即设置阻抗控制类型为关节阻抗之后，方可调用此接口，不满足此限制，关节阻抗参数将无法设置成功。

12.3.4.7 SetCartCtrlStiffVec

说明

设置笛卡尔阻抗刚度。

定义

SetCartCtrlStiffVec trans_stiff_x，trans_stiff_y，trans_stiff_z，rot_stiff_x，rot_stiff_y，rot_stiff_z

trans_stiff_x

数据类型：double

X 方向笛卡尔阻抗力刚度，取值范围（0~1500），单位：N/m。

trans_stiff_y

数据类型：double

Y 方向笛卡尔阻抗力刚度，取值范围（0~1500），单位：N/m。

trans_stiff_z

数据类型：double

Z 方向笛卡尔阻抗力刚度，取值范围（0~1500），单位：N/m。

rot_stiff_x

数据类型：double

X 方向笛卡尔阻抗力矩刚度，取值范围（0~100），单位：N.m/rad。

rot_stiff_y

数据类型：double

Y 方向笛卡尔阻抗力矩刚度，取值范围（0~100），单位：N.m/rad。

rot_stiff_z

数据类型：double

Z 方向笛卡尔阻抗力矩刚度，取值范围（0~100），单位：N.m/rad。

示例

例 1

FcInit Tool1, Wobj0, 0
SetControlType 1
SetCartCtrlStiffVec 1000，1000，1000，100，100，100

设置阻抗控制模式为笛卡尔阻抗，X/Y/Z 方向的阻抗力刚度设置为 1000，阻抗力矩刚度设置为 100。

使用限制

- SetControlType 1 之后，也即设置阻抗控制类型为笛卡尔阻抗之后，方可调用此接口，不满足此限制，笛卡尔阻抗参数将无法设置成功。

8.3.1.8 SetJntTrqDes

说明

设置关节期望力矩。

定义

SetJntTrqDes tau_d1,tau_d2,tau_d3,tau_d4,tau_d5,tau_d6,tau_d7

tau_d1

数据类型：double

关节 1 期望力矩，取值范围（0~20），单位 N.m。

tau_d2

数据类型：double

关节 2 期望力矩，取值范围（0~20），单位 N.m。

tau_d3

数据类型：double

关节 3 期望力矩，取值范围（0~20），单位 N.m。

tau_d4

数据类型：double

关节 4 期望力矩，取值范围（0~20），单位 N.m。

tau_d5

数据类型：double

关节 5 期望力矩，取值范围（0~20），单位 N.m。

tau_d6

数据类型：double

关节 6 期望力矩，取值范围（0~20），单位 N.m。

tau_d7

数据类型：double

关节 7 期望力矩，取值范围（0~20），单位 N.m。

示例

例 1

```
FcInit Tool1, Wobj0, 0
SetControlType 0
FcStart
SetJntTrqDes 5,5,5,5,5,5,5
```

FcStop
设置关节期望力矩，所有关节的期望力矩设置为 5N.m。

使用限制

- FcStart 之后，FcStop 之前，方可调用此接口，不满足此限制，关节期望力矩设置不成功。

8.3.1.9 SetCartForceDes

说明

设置笛卡尔期望力/力矩。

定义

SetCartForceDes force_x，force_y，force_z，torque_x，torque_y，torque_z

force_x

数据类型：double

X 方向笛卡尔期望力，单位 N。

force_y

数据类型：double

Y 方向笛卡尔期望力，单位 N。

force_z

数据类型：double

Z 方向笛卡尔期望力，单位 N。

torque_x

数据类型：double

X 方向笛卡尔期望力矩，取值范围（0~20），单位 N.m。

torque_y

数据类型：double

Y 方向笛卡尔期望力矩，取值范围（0~20），单位 N.m。

torque_z

数据类型：double

Z 方向笛卡尔期望力矩，取值范围（0~20），单位 N.m。

示例

例 1

```
FcInit Tool1, Wobj0, 0
SetControlType 1
FcStart
SetCartForceDes 0,0,5,0,0,0
FcStop
```

设置笛卡尔期望力/力矩，将 z 方向的期望力设置为 5N。

使用限制

- FcStart 之后，FcStop 之前，方可调用此接口，不满足此限制，笛卡尔期望力/力矩设置不成功。

12.3.4.10 SetSineOverlay

说明

设置绕单轴旋转的正弦搜索运动。

定义

SetSineOverlay line_dir, amplify, frequncy, phase, bias

line_dir

数据类型：int

line_dir：搜索运动参考轴，支持

0：参考方向为 X 轴

1：参考方向为 Y 轴

2：参考方向为 Z 轴

amplify

数据类型：double

搜索运动幅值，取值范围（0~10），单位 N.m。

frequncy

数据类型：double

搜索运动频率，取值范围（0~5），单位 Hz。

phase

数据类型：double

搜索运动相位，取值范围（0~3.14），单位 rad。

bias

数据类型：double

搜索运动偏置，取值范围（0~10），单位 N.m。

示例

例 1

```
FcInit Tool1, Wobj0, 0
SetControlType 1
SetSineOverlay 0，10，5，3.14，2
```

设置绕 x 轴（0）的旋转搜索运动，幅值为 10N.m，频率为 5Hz，相位为 3.14，偏置为 2N.m。

使用限制

- SetControlType 1 之后，也即设置阻抗控制类型为笛卡尔阻抗之后，StartOverlay 之前。方可调用此接口，不满足此限制，正弦搜索运动将无法设置成功。

12.3.4.11 SetLissajousOverlay

说明

设置平面内的莉萨如搜索运动。

定义

SetLissajousOverlay plane, amplify_one, frequncy_one, amplify_two, frequncy_two, phase_diff

plane

数据类型：int

搜索运动参考平面，支持

0：参考平面为 XY 平面

1：参考平面为 XZ 平面

2：参考平面为 YZ 平面

amplify_one

数据类型：double

amplify_one：搜索运动一方向幅值，取值范围（0~10），单位 N.m。

frequncy_one

数据类型：double

frequncy_one：搜索运动一方向频率，取值范围（0~5），单位 Hz。

amplify_two

数据类型：double

amplify_two：搜索运动二方向幅值，取值范围（0~10），单位 N.m。

frequncy_two

数据类型：double

frequncy_two：搜索运动二方向频率，取值范围（0~5），单位 Hz。

phase_diff

数据类型：double

phase_diff：搜索运动两个方向相位偏差，取值范围（0~3.14），单位 rad。

示例

例 1

FcInit Tool1，Wobj0，0

SetControlType 1

SetLissajousOverlay 0，5，2.5，10，5，3.14

设置 xy 平面（0）内的莉萨如搜索运动，x 方向的幅值为 5N.m，频率为 2.5Hz；x 方向的幅值为 10N.m，频率为 5Hz，y 方向相对于 x 方向的相位偏差为 3.14。

使用限制

- SetControlType 1 之后，也即设置阻抗控制类型为笛卡尔阻抗之后，StartOverlay 之前。不满足此限制，搜索运动将无法设置成功。

12.3.4.12 SetLoad

说明

设置力控模块使用的负载信息。

定义

SetLoad m,rx,ry,rz,Ixx,Iyy,Izz

m

数据类型：double

负载质量，单位：kg

rx

数据类型：double

负载质心在法兰坐标系 x 方向的距离，单位：mm。

ry

数据类型：double

负载质心在法兰坐标系 y 方向的距离，单位：mm。

rz

数据类型：double

负载质心在法兰坐标系 z 方向的距离，单位：mm。

Ixx

数据类型：double

负载质心 x 轴的主轴惯量，单位：kg*mm^2。

Iyy

数据类型：double

负载质心 y 轴的主轴惯量，单位：kg*mm^2。

Izz

数据类型：double

负载质心 z 轴的主轴惯量，单位：kg*mm^2。

示例

例 1

```
FcInit Tool1, Wobj0, 0
FcStart
SetLoad 1,0,0,10,0.001,0.001,0.0001
```

设置末端负载，此负载具有这样的信息：质量 1kg，质心在法兰坐标系下的分量分别为 0，0，10mm，负载相对于负载质心坐标系的主轴转动惯量分别为 0.001kg*mm^2，0.001kg*mm^2，0.0001kg*mm^2。

使用限制

- FcStart 之后，方可调用此接口，不满足此限制，负载参数将无法设置成功。

8.3.1.13 FcStart

说明

用于开启力控操作，此操作将机器人从单纯的位置控制切换到力控制。

定义

FcStart
没有参数，直接使用。

示例

例 1

FcInit Tool1, Wobj0, 0
FcStart
FcInit 之后通过 FcStart 开启力控，此时机器人处于力控模式。

使用限制

- FcInit 之后调用此接口，且调用之前应确保机器人机械零点、力传感器零点、负载信息已经正确设置，本体参数已经正确辨识。否则，将影响力控功能的使用效果，甚至无法开启力控功能。

12.3.4.14 FcStop

说明

用于停止力控，此操作将机器人从力控制切换到位置控制。执行该指令内部会自动停止所有搜索运动。

定义

FcStop
没有参数，直接使用。

示例

例 1

FcInit Tool1, Wobj0, 0
FcStart
FcStop
FcStop 停止力控，将机器人从力控制切换到位置控制。执行该指令会清空所有的力控状态。

使用限制

- FcStart 之后调用此接口，调用此接口时会清空力控状态，例如：力控负载信息、阻抗参数、搜索运动、期望力等。如需再次开启力控，则需重新 FcInit。

12.3.4.15 StartOverlay

说明

开启前面设置的搜索运动。

定义

StartOverlay
没有参数，直接使用。

示例

例 1

```
FcInit Tool1, Wobj0, 0
SetControlType 1
SetSineOverlay 0，10，5，3.14，2
SetLissajousOverlay 0，5，2.5，10，5，3.14
FcStart
StartOverlay
```

开启搜索运动，搜索运动为之前设置的各种搜索运动的叠加。示例中的搜索运动为绕 x 轴的正弦搜索运动和 xy 平面内的莉萨如搜索运动的叠加。

使用限制

- FcStart 之后，方可调用此接口，不满足此限制，搜索运动将无法开启成功。

12.3.4.16 PauseOverlay

说明

暂停搜索运动。

定义

PauseOverlay
没有参数，直接使用。

示例

例 1

```
FcInit Tool1, Wobj0, 0
SetControlType 1
SetSineOverlay 0，10，5，3.14，2
FcStart
StartOverlay
PauseOverlay
```

暂停搜索运动。

使用限制

- StartOverlay 之后，方可调用此接口。

12.3.4.17 RestartOverlay

说明

重新开启暂停的搜索运动。

定义

RestartOverlay
没有参数，直接使用。

示例

例 1

```
FcInit Tool1, Wobj0, 0
SetControlType 1
SetSineOverlay 0，10，5，3.14，2
FcStart
StartOverlay
PauseOverlay
RestartOverlay
```

重新开启搜索运动。

使用限制

- PauseOverlay 之后，方可调用此接口。此接口和 PauseOverlay 搭配使用，用于重新启动暂停的搜索运动。

12.3.4.18 StopOverlay

说明

停止搜索运动。

定义

StopOverlay
没有参数，直接使用。

示例

例 1

```
FcInit Tool1, Wobj0, 0
SetControlType 1
SetSineOverlay 0，10，5，3.14，2
FcStart
StartOverlay
StopOverlay
```

停止搜索运动。

使用限制

- StartOverlay 之后，调用此接口才具有实际意义。

12.3.4.19 FcCondForce

说明

用于定义与接触力有关的终止条件。

定义

FcCondForce xmin, xmax, ymin, ymax, zmin, zmax, IsInside, TimeOut

xmin

定义 X 方向上的力限制下限，为负值时表示 X 负方向上的最大值。单位为 N，默认值为负无穷。

数据类型： double

xmax

定义 X 方向上的力限制上限，为负值时表示 X 负方向上的最小值。单位为 N，默认值为正无穷。

数据类型： double

ymin

定义 Y 方向上的力限制下限，为负值时表示 Y 负方向上的最大值。单位为 N，默认值为负无穷。

数据类型： double

ymax

定义 Y 方向上的力限制上限，为负值时表示 Y 负方向上的最小值。单位为 N，默认值为正无穷。

数据类型： double

zmin

定义 Z 方向上的力限制下限，为负值时表示 Z 负方向上的最大值。单位为 N，默认值为负无穷。

数据类型： double

zmax

定义 Z 方向上的力限制上限，为负值时表示 Z 负方向上的最小值。单位为 N，默认值为正无穷。

数据类型： double

IsInside

用于定义限制条件内部为 true 还是外部为 true。

数据类型： bool

TimeOut

定义超时时间，单位为秒，取值范围 1~600。

数据类型： double

示例

例 1

```
FcInit Tool1, Wobj0, 0
FcStart
FcCondForce -100，100，-100，100，-100，100，true，60
```

定义一个终止条件，当接触力在力控坐标系 X/Y/Z 轴方向的大小在正负 100N 的范围之内时，条件为 true，当超出 100N 范围时终止。超时时间为 60 秒。

使用限制

- FcStart 之后，FcStop 之前，方可调用此接口，不满足此限制，接触力终止条件无法设置成功。

12.3.4.20 FcCondPosBox

说明

用于定义与接触位置有关的终止条件。

定义

```
FcCondPosBox SupvFrame, Box, IsInside, Timeout
```

SupvFrame

用于选择监控的空间体在哪个坐标系定义。该坐标系是从工件坐标系上叠加一个坐标系转换得来的，转换坐标系由 pose 定义，默认使用 pose0，即不使用任何转换，直接使用工件坐标系。

数据类型： pose

Box

定义一个长方体。

数据类型： fcboxvol

IsInside

用于定义限制条件内部为 true 还是外部为 true。

数据类型： bool

TimeOut

定义超时时间，单位为秒，取值范围 1~600。

数据类型： double

示例

例 1

```
FcInit Tool1, Wobj0, 0
FcStart
VAR fcboxvol box1 = fcbv:{-100.0, 100.0, -200.0, 200.0, -300.0, 300.0}
```

```
VAR pose pose1 = pe:{0, 0, 0},{1, 0, 0, 0}
FCCondPosBox pose1, box1, false, 60
```

定义一个终止条件，当机器人 TCP 进入定义的长方体内部或等待超过 60 秒 ，触发终止条件。

使用限制

- FcStart 之后，FcStop 之前，方可调用此接口，不满足此限制，立方体位置终止条件无法设置成功。。

12.3.4.21 FcCondTorque

说明

用于定义与接触力矩有关的终止条件。

定义

```
FcCondTorque xmin, xmax, ymin, ymax, zmin, zmax, IsInside, TimeOut
```

xmin

定义 X 方向上的力矩限制下限，为负值时表示 X 负方向上的最大值。单位为 N.m，默认值为负无穷。

数据类型： double

xmax

定义 X 方向上的力矩限制上限，为负值时表示 X 负方向上的最小值。单位为 N.m，默认值为正无穷。

数据类型： double

ymin

定义 Y 方向上的力矩限制下限，为负值时表示 Y 负方向上的最大值。单位为 N.m，默认值为负无穷。

数据类型： double

ymax

定义 Y 方向上的力矩限制上限，为负值时表示 Y 负方向上的最小值。单位为 N.m，默认值为正无穷。

数据类型： double

zmin

定义 Z 方向上的力矩限制下限，为负值时表示 Z 负方向上的最大值。单位为 N.m，默认值为负无穷。

数据类型： double

zmax

定义 Z 方向上的力矩限制上限，为负值时表示 Z 负方向上的最小值。单位为 N.m，默认值为正无穷。

数据类型： double

IsInside

用于定义限制条件内部为 true 还是外部为 true。

数据类型： bool

TimeOut

定义超时时间，单位为秒，取值范围 1~600。

数据类型： double

示例

例 1

FcInit Tool1, Wobj0, 0
FcStart
FcCondTorque -10，10，-10，10，-10，10，true，60

定义一个终止条件，当接触力矩在力控坐标系各轴方向的大小大于 10Nm，或时间超时 60 秒，则触发终止条件。

使用限制

- FcStart 之后，FcStop 之前，方可调用此接口，不满足此限制，接触力矩终止条件无法设置成功。

12.3.4.22 FcCondWaitWhile

说明

用于激活在之前定义的终止条件，并在当前行等待直到这些条件变为 False 或者超时。

定义

FcCondWaitWhile
没有参数，直接使用。

示例

例 1

FcInit Tool1, Wobj0, 0
FcStart
FcCondTorque -10，10，-10，10，-10，10，true，60
FcCondForce -100，100，-100，100，-100，100，true，60
FcCondWaitWhile
激活终止条件，程序阻塞在当前位置，等待触发终止条件。

使用限制

- 定义力控终止条件之后。

12.3.4.23 GetEndToolTorque

说明

用于获取机器人当前的力矩信息

定义

GetEndToolTorque Tool, Wobj [, RefType]

[]表示该参数可以省略

返回值

力矩信息

数据类型：TorqueInfo

参数

Tool

当前使用的工具信息。

数据类型： Tool

Wobj

当前使用的工件信息。

数据类型： Wobj

RefType

力矩相对的参考系

数据类型： Int

0：默认值，末端相对世界坐标系的力矩信息

1：末端相对于法兰盘的力矩信息

2：末端相对于 TCP 点的力矩信息

示例

例 1

FcInit Tool1, Wobj0, 0

FcStart

FcCondTorque -10，10，-10，10，-10，10，true，60

FcCondForce -100，100，-100，100，-100，100，true，60

FcCondWaitWhile

激活终止条件，程序阻塞在当前位置，等待触发终止条件。

使用限制

- 定义力控终止条件之后。

12.3.5 拖动回放

12.3.5.1 ReplayPath

说明

对使用拖动示教进行录制的轨迹进行回放复现。可以控制回放时的运行速率。参考 4.2.4 节轨迹复现。

定义

ReplayPath path [, rate] [, wobj/tool]

path

数据类型：path
拖动回放路径类型，在路径列表中定义、拖动示教录制。

rate

数据类型：double
回放百分比，范围 0.01~3.00。0.01 表示按照拖动时 1%运行速率回放；1.00 表示 100%运行速率原速回放；3.00 表示 300%运行速率回放。

wobj/tool

数据类型：工具|工件
指定多动回放指令的末端设备是某个工具或者工件，回放时机器人会按照对应设备的工具更改回放的控制参数，提高运行的稳定性

示例

例 1

ReplayPath path , 1, tool1
使用原始速率进行录制回放。

12.3.6 IO 指令

12.3.6.1 SetDO

说明

设置某个数字输出信号的值。

定义

SetDO DoName，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。

Value

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

示例

例 1

SetDO do2，true
将 do2 对应的数字输出点置为高电平。

12.3.6.2 SetAllDO

说明

设置所有数字输出信号的值。

定义

SetAllDO Value

Value

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

示例

例 1

SetAllDO true
将所有数字输出电压置为高电平，绑定了系统功能的 DO 除外。

12.3.6.3 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

例 1

SetGO go3，8
将 go3 对应的一组物理端口的值设置为 8。

12.3.6.4 SetAO

说明

设置某个模拟输出信号的值。

定义

SetAO AoName，Value

AoName

数据类型：signalao

指定需要改变值的 ao 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：double
ao 信号的目标值.

示例

例 1

SetAO ao3，5.123
将 ao3 对应的一组物理端口的值设置为 5.123。

12.3.6.5 PulseDO

说明

用于产生一个脉冲的 DO 信号。

定义

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。

12.3.7 通信指令

RL 程序中机器人与外部设备通信，支持以太网和串口两种方式。在指令集上，设计了一套统一的指令进行资源管理和数据收发，以提供一致的使用体验。

指令集	TCP 客户端	TCP 服务器	串口
OpenDev	√	√	√
SocketAccept	N/A	√	N/A
CloseDev	√	√	√
SendString	√	√	√
SendByte	√	√	√
ReadBit	√	√	√
ReadByte	√	√	√

ReadDouble	√	√	N/A
ReadInt	√	√	N/A
ReadString	√	√	√
GetSocketConn	√	N/A	N/A
GetSocketServer	N/A	√	N/A
GetBufSize	N/A	N/A	√
ClearBuffer	N/A	N/A	√

12.3.7.1 OpenDev

说明

用于打开监听服务器、作为客户端发起连接、打开串口资源，取决于参数所表示的对象。

1) 打开 SocketServer 对象时，将初始化资源，并且完成端口绑定和端口监听。
2) 打开 SocketConn 对象时，机器人将作为 TCP 客户端，按照预设的 ip 和端口尝试连接外部服务器。
3) 打开串口资源，将按照窗口参数初始化串口，准备好通信条件。

返回值

无。

定义

OpenDev(name)

name

数据类型：string
客户端对象或者服务端对象或者串口资源的名称。

示例

例 1

```
SocketConn scnn3 = {"192.168.0.200", 8090, "clt1", 2, "\n"}
try
  OpenDev("clt1")  // 尝试连接远程服务器，连接成功则 clt1 的 attr 自动修改为 outgoing。
  string readstr = ReadString(30, "clt1")
  .....  //对 readstr 的逻辑处理
  string sendstr = "hello server！"
  SendString(sendstr， "clt1")  //使用 clt1 的客户端连接来发送数据
  ...  // 一系列代码
catch(ERROR e)  // ERROR 错误类型，包括产生错误的文件、行号、错误码、错误内容等
  ...  //一系列异常处理
Endtry
```

例 2

```
SocketServer listener1  = {"192.168.0.200", 8090, "svr1"}
global pers bool exit = false
try
  OpenDev( "svr1" ) //绑定端口、监听端口
```

```
  while(exit != true)
    SocketConn conn = SocketAccept( "svr1")  //阻塞式接收连入的客户端
  Endwhile
catch(ERROR e)
...    //一系列异常处理
Endtry
```

错误处理

若发生报错，控制系统将抛出异常，并上报错误原因。如果异常没有被 try 代码块捕获，则控制系统将停止程序运行。

12.3.7.2 SocketAccept

说明

阻塞等待客户端连接到来，并完成客户端连接处理。该指令仅在机器人作为 TCP 服务器时使用。

> 提示
>
> 1、 指令将阻塞当前任务，因此正确的使用方法是在多任务中使用，单独作为一个低优先级任务持续不断接收并产生通信连接对象 SocketConn。
> 2、 指令返回一个连接操作对象，并且拥有客户端连接 ip 和 port 信息，可供程序其他部分使用。返回的连接对象是一个 SocketConn 结构体，名称由系统随机分配。拿到连接对象以后，请更改连接对象的名称，避免连接丢失。
> 3、 支持服务器一对多连接。

返回值

数据类型：SocketConn

外部设备以 TCP 客户端形式连接上机器人后，控制系统产生一个通信对象，用于 RL 程序来控制通信读写。

定义

```
SocketConn conn  = SocketAccept(name)
```

name

数据类型：string

已经准备好的 SocketServer 对象的名称，并且已经使用 OpenDev 打开成功。

基本示例

例 1

```
SocketServer listener1  = {"192.168.0.200", 8090, "svr1"}
global pers bool exit = false
try
  OpenDev( "svr1" ) //绑定端口、监听端口
  while(exit != true)
    SocketConn conn = SocketAccept( "svr1")  //阻塞式接收连入的客户端
    conn.name = "client1"  // 重要！ 给通信连接起个名，否则后续难以按名称进行数据读写
    conn.suffix = "\n"  //可选，设置数据包结束符
```

```
  Endwhile
catch(ERROR e)
...    //一系列异常处理
Endtry
```

错误处理

若发生报错，控制系统将抛出异常，并上报错误原因。如果异常没有被 try 代码块捕获，则控制系统将停止程序运行。

12.3.7.3 CloseDev

说明

关闭资源，可用于关闭 TCP 通信链接、TCP 监听服务器或者串口资源。

返回值

无。

定义

```
CloseDev(name)
```

name

数据类型：string

用于通信的 SocketConn 连接、监听服务器 SocketServer 对象或者串口资源。

基本示例

例 1

```
SocketConn scnn3 = {"192.168.0.200", 8090, "clt1", 2, "\n"}
try
  OpenDev("clt1")
  string readstr = ReadString(30, "clt1")
  .....  //对 readstr 的逻辑处理
  string sendstr = "hello server！"
  SendString（sendstr， "clt1"） //使用 clt1 的客户端连接来发送数据
  ...    // 一系列代码
catch(ERROR e)
  ...    //一系列异常处理
endtry
CloseDev("clt")  //无论是否发生错误，最后关闭 socket 客户端。
```

例 2

```
SocketServer listener1  = {"192.168.0.200", 8090, "svr1"}
global pers bool exit = false
try
  OpenDev( "svr1" ) //绑定端口、监听端口
  while(exit != true)
    SocketConn conn = SocketAccept( "svr1")  //阻塞式接收连入的客户端
    conn.name = "client1"  // 重要！ 给通信连接起个名，否则后续难以按名称进行数据读写
    conn.suffix = "\n"  //可选，设置数据包结束符
  Endwhile
catch(ERROR e)
...    //一系列异常处理
Endtry

CloseDev("client1")  //关闭与外部 TCP 客户端通信。重要！
```

```
CloseDev("svr1")      //关闭监听服务器
```

提示

1、 示例 2 中，有两个网络对象存在，必须按照顺序，先关闭通信连接，再关闭服务器对象，否则将产生资源释放不彻底的状态（TCP TIME_WAIT 状态）。
2、 如果机器人做服务器时，已经与外部建立了多路通信连接，则需要在关闭服务器之前，先将这些通信连接依次关闭掉。
3、 如果发生了资源释放不彻底，需要重启控制系统才能恢复。不过不必过于担心，控制系统中允许的资源数量有冗余，可以保证少量的资源被占用后，依然可以正常工作。但还是需要避免大量资源因为使用方式不正确而处于占用状态。

12.3.7.4 SendString

说明

对外发送一个字符串。可以是通过网络或串口，取决于参数中的标识符所表示的硬件资源。

定义

SendString(StringData，name)

StringData

数据类型：string
待发送的字符串数据。

name

数据类型：string
用于发送数据的硬件资源名称。 可以是建立好的 TCP 通信连接 SocketConn 对象，也可以是已经成功打开的串口资源。

示例

例 1

SendString(“Hello World”，” Socket0”)
通过 Socket0 对外发送 Hello World 字符串。 其中 Socket0 是已经定义并连接成功的 SocketConn 类型。

例 2

```
VAR String str1 =“Hello World”
SocketSendString(str1，” Serial1”)
```

通过 Serial1 对外发送 str1 存储的字符串 Hello World。 其中 Serial1 是已经定义并成功打开的串口。

12.3.7.5 SendByte

说明

对外发送一个字节 byte，在需要发送 ASCII 字符时非常有用。

返回值

无。

定义

SendByte(ByteData, name)

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。

name

数据类型：string
用于发送数据的 Socket 或者串口名称。

示例

例 1

SendByte(13, "socket0")
通过 socket0 对外发送一个回车符。

例 2

VAR byte data1 = 13
SendByte(data1, "serial0")
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 serial0 对外发送。

例 3

VAR byte data2[2] = {13,17}
SendByte(data2, "socket0")
通过 socket0 发送一个 byte 类型数组变量 data2。会将数组中的内容全部发出去。

例 4

VAR byte data2[2] = {13,17,20}
SendByte(data2[2], "socket0")
通过 socket0 发送一个 byte 类型变量，该变量为 data2[2]，仅表示数组中的第 2 个元素。此时会将 data2[2]的值 17 发出去，而不会多发任何其他元素。

12.3.7.6 ReadBit

说明

控制系统按 bit 接收数据。

1）通过网络通信以 TCP 形式接收，外部发送的数据需以 SocketConn 配置好的结束符结尾。

2）通过串口通信方式接收，外部设备只需要发送数据部分即可，无需考虑结束符。

返回值

数据类型：bool 数组
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

Ret = ReadBit(BitNum, TimeOut, name)

BitNum

数据类型：int
需要读取的 bit 数量，大小应该为 8 的整数倍。

TimeOut

数据类型：int
超时时间。单位 s，范围 0~86400，默认 60s。

name

数据类型：string
通信连接 SocketConn 的名称或者串口的名称。

Ret

数据类型：bool 数组
接收到的数据，数组第一个元素表示最低位。

示例

例 1

```
bool groupio[16]
groupio = ReadBit(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。
假设外部设备发送 ASCII 字符：95+结束符，则机器人接收到"95"。其中"9"的十六进制为 0x39，"5"的十六进制为 0x35，因此用户接收到的数据为 0x3935。此时 groupio 数组从[1]~[16]表示为：1001 1100 1010 1100。[1]是数据低位，与 0x3935 符合。

12.3.7.7 ReadByte

说明

接收一定字节的数据。注意数据之间需要使用逗号隔开。

返回值

数据类型：byte 数组
使用 byte 型数组存储接收到的数据。

定义

```
Ret = ReadByte(ByteNum, TimeOut, name)
```

ByteNum

数据类型：int
需要读取的 bit 数量，大小应该为 8 的整数倍。

TimeOut

数据类型：int
超时时间。单位 s，范围 0~86400，默认 60s。

name

数据类型：string
通信连接 SocketConn 的名称或者串口的名称。

Ret

数据类型：byte 数组
接收到的数据。

示例

例 1

byte rets[6] = {0,0,0,0,0,0}
rets = ReadByte(6,60,"clt1")
读取 6 个 byte 的数据存储到名为 rets 的 byte 类型数组中，超时时间 60s。
注意外部设备的 byte 数据之间需要使用逗号隔开，比如发送”1,2,3,4,5,6”
通过 TCP 发送数据，需要在数据后面加上预设好的结束符。
通过串口发送数据，不需要结束符。

12.3.7.8 ReadDouble

说明

通过 Socket 接收 double 型数据，外部发送的数据需以配置好的结束符结尾。
注意，该指令仅对 TCP 网络通信生效，适用于机器人做客户端和服务端，但不适用于串口。

返回值

数据类型：double 数组
使用 double 型数组存储接收到的数据。

定义

Ret = ReadDouble(DoubleNum, TimeOut, name)

DoubleNum

数据类型：double
需要读取的 double 数个数，最大为 30 个。

TimeOut

数据类型：int
超时时间。单位 s，范围 0~86400，默认 60s。

name

数据类型：string
用于接收数据的 Socket 名称。

示例

例 1

double dd[10]
dd =ReadDouble(10, 60, "Socket0")
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。

12.3.7.9 ReadInt

说明

通过 Socket 接收 int 型数据，外部发送的数据需以配置好的结束符结尾。
注意，该指令仅对 TCP 网络通信生效，适用于机器人做客户端和服务端，但不适用于串口。

返回值

数据类型：int
使用 int 型数组存储接收到的数据。

定义

Ret = ReadInt(IntNum, TimeOut, name)

IntNum

数据类型：int
需要读取的 int 数个数，最大为 30 个。

TimeOut

数据类型：int
超时时间。单位 s，范围 0~86400，默认 60s。

name

数据类型：string
用于接收数据的 Socket 名称。

示例

例 1

```
int ii[10]
ii = ReadInt(10, 60, "Socket0")
```

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。

12.3.7.10 ReadString

说明

读取字符串并返回，外部发送的数据需以配置好的结束符结尾。

返回值

数据类型： string
存储接收到的字符串。

定义

Ret = ReadString(TimeOut, name, [len])

TimeOut

数据类型： int
超时时间。 单位 s， 范围 0~86400，默认 60s。

name

数据类型：string
用于接收数据的 Socket 或者串口名称。

len

数据类型：int
可选参数，使用串口读取时才使用。由于串口中没有加入结束符的概念，因此需要指定长度后才能正确读取和解析。

示例

例 1

```
VAR String str1
str1 = ReadString(60, "Socket1")
```

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

例 2

```
VAR String str1
str1 = ReadString(60, “serial0”,5)
```

从 serial0 中接收一个长度为 5 的字符串，并存储到 str1 中，超时时间 60s。串口通信方式。

说明

接收一定字节的数据。注意数据之间需要使用逗号隔开。

返回值

数据类型：byte 数组
使用 byte 型数组存储接收到的数据。

定义

Ret = ReadByte(ByteNum, TimeOut, name)

ByteNum

数据类型：int
需要读取的 bit 数量，大小应该为 8 的整数倍。

TimeOut

数据类型：int
超时时间。单位 s，范围 0~86400，默认 60s。

name

数据类型：string
通信连接 SocketConn 的名称或者串口的名称。

Ret

数据类型：byte 数组
接收到的数据。

示例

例 1

```
byte rets[6] = {0,0,0,0,0,0}
rets = ReadByte(6,60,"clt1")
```

读取 6 个 byte 的数据存储到名为 rets 的 byte 类型数组中，超时时间 60s。
注意外部设备的 byte 数据之间需要使用逗号隔开，比如发送”1,2,3,4,5,6”
通过 TCP 发送数据，需要在数据后面加上预设好的结束符。
通过串口发送数据，不需要结束符。

12.3.7.11 GetSocketConn

说明

从 socket 连接名称查找对应的 socket 属性集对象。该指令获取的结果可以用来进行判断和处理逻辑。仅应该将其当作只读对象使用。该指令仅适用于通信连接（包括机器人作为客户端，或者作为服务端已经连接上的用于通信的通道），不适用于监听服务器和串口。

返回值

数据类型：SocketConn
通过字面名称查找到的 socket 属性对象。

定义

Ret = GetSocketConn(name)

name

数据类型：string
通信连接 SocketConn 的名称。

Ret

数据类型：SocketConn
通过字面名称查找到的 socket 属性对象。

示例

例 1

SocketConn ret= GetSocketConn(“client0”)
查找名称为“client0”的 SocketConn 对象。可以使用 ret 来获得这个连接的属性，包括 ip 地址、端口号、通信结束符、连接状态等信息。

可查询属性	查询方法	含义和示例
ip 地址	ret.ip	字符串，如”192.168.0.161”
端口号	ret.port	整型，如 8090
属性	ret.attr	机器人做服务器：“incoming”； 机器人做客户端：“outgoing”； 如果连接未建立：””或者其他值，通常为空
缓存大小	ret.cache	1~100
名称	ret.name	示例为“client0”
连接状态	ret.state	closed、establish

12.3.7.12 GetSocketServer

说明

从用户定名称查找对应的服务器属性集对象。该指令获取的结果可以用来进行判断和处理逻辑。仅应该将其当作只读对象使用。该指令仅适用于监听服务器（SocketServer 对象），不适用于通信连接（包括机器人作为客户端，或者作为服务端已经连接上的用于通信的通道）和串口。

返回值

数据类型：SocketServer
通过字面名称查找到的服务器属性对象。

定义

Ret = GetSocketServer(name)

name

数据类型：string
通信连接 SocketServer 的名称。

Ret

数据类型：SocketServer
通过字面名称查找到的 socket 属性对象。

示例

例 1

```
SocketServer listener1  = {"192.168.0.200", 8090, "svr1"}
OpenDev( "svr1" ) //绑定端口、监听端口
//根据连接标识 "svr1" 获取 SocketServer 对象，此时 ret 将复制任务 1 中 listener1 的全部状态
SocketServer  ret= GetSocketServer("svr1")
if(svrfindout.state == "listening")  //使用 SocketServer 的 attr 属性，判断是否监听中
  //逻辑处理
endif
```

可查询属性	查询方法	含义和示例
ip 地址	ret.ip	字符串，如”192.168.0.161”
端口号	ret.port	整型，如 8090
名称	ret.name	示例为 “svr1”
连接状态	ret.state	closed、listening、error

12.3.7.13 GetBufSize

说明

获取串口缓冲区中还剩余多少数据未读，单位为字节。指令仅适用于串口，不适用于 TCP 服务器和客户端。

返回值

数据类型：int
缓冲区中未处理数据量，以字节为单位。

定义

```
Ret = GetBufSize(name)
```

name

数据类型：string
串口资源的字面名称。

Ret

数据类型：int
缓冲区中未处理数据量，以字节为单位。

示例

例 1

```
OpenDev("serial0")
int a = GetBufSize("serial0")
print(a)
```

12.3.7.14 ClearBuffer

说明

清除缓冲区，未读取完毕的字符将丢失。指令仅适用于串口，不适用于 TCP 服务器和客户端。

返回值

无。

定义

ClearBuffer(name)

name

数据类型：string
串口资源的字面名称。

Ret

数据类型：int
缓冲区中未处理数据量，以字节为单位。

示例

例 1

```
OpenDev("serial0")
int a = GetBufSize("serial0")
print(a)
```

12.3.8 网络指令

12.3.8.1 SocketCreate（过期的）

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"回车"结尾，在接收到"回车"之前的所有数据都将合并做为同一条数据处理。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。
注意，该指令被标注为"过期的"，为 xCore 控制系统 1.3 版本使用指令，在更高版本中依然有效，但不再继续维护，也不推荐继续使用。

返回值

数据类型：bool

如果创建成功返回 true，创建失败返回 false

定义

SocketCreate（"ip_Address", Port, "Name" [,Cache] [, "Terminator"]）

ip_Address

数据类型：string 定义需要连接 server 的 ipv4 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。

Cache

数据类型：int

定义 Socket 缓存大小，通信数据存在缓存队列中，可省略。

Terminator

数据类型：string

定义 Socket 通信的结束符类型，可省略，默认是"\r"。

示例

例 1

```
if (SocketCreate("10.0.6.11",8080,"S1",10,"\r"))
  // 创建成功
else
  // 错误处理
endif
```

提示

3、 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。

4、 为避免循环模式下，频繁调用 SocketCreate 和 SocketClose 指令，两条指令之间最好增加时间延时，例如：

```
SocketClose("S1")
wait 0.1
SocketCreate("10.0.6.11",8080,"S1",10,"\r")
```

12.3.8.2 SocketClose（过期的）

说明

关闭 Socket。

注意，该指令被标注为"过期的"，为 xCore 控制系统 1.3 版本使用指令，在更高版本中依然有效，但不再继续维护，也不推荐继续使用。

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

例 1

SocketClose（”Socket0”）

12.3.8.3 SocketSendString（过期的）

说明

通过 Socket 对外发送一个字符串。
注意，该指令被标注为”过期的”，为 xCore 控制系统 1.3 版本使用指令，在更高版本中依然有效，但不再继续维护，也不推荐继续使用。

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

例 1

SocketSendString（”Hello World”，” Socket0”）
通过 Socket0 对外发送 Hello World 字符串。

例 2

VAR String str1 =”Hello World”
SocketSendString（str1，” Socket0”）
通过 Socket0 发送 str1 存储的字符串。

12.3.8.4 SocketSendByte（过期的）

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。
注意，该指令被标注为"过期的"，为 xCore 控制系统 1.3 版本使用指令，在更高版本中依然有效，但不再继续维护，也不推荐继续使用。

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

例 1

SocketSendByte(13, "socket0")
通过 socket0 对外发送一个回车符。

例 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。

例 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。

12.3.8.5 SocketReadBit（过期的）

说明

通过 Socket 按 bit 接收数据，外部发送的数据需以回车结尾。
注意，该指令被标注为"过期的"，为 xCore 控制系统 1.3 版本使用指令，在更高版本中依然有效，但不再继续维护，也不推荐继续使用。

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

数据类型：int

需要读取的 bit 数量，大小应该为 8 的整数倍。

TimeOut

数据类型：int
超时时间。单位 s，范围 0~86400，默认 60s。

SocketName

数据类型：string
用于接收数据的 Socket 名称。

示例

例 1

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。

12.3.8.6 SocketReadDouble（过期的）

说明

通过 Socket 接收 double 型数据，外部发送的数据需以回车结尾。
注意，该指令被标注为"过期的"，为 xCore 控制系统 1.3 版本使用指令，在更高版本中依然有效，但不再继续维护，也不推荐继续使用。

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

数据类型：double
需要读取的 double 数个数，最大为 30 个。

TimeOut

数据类型：int
超时时间。单位 s，范围 0~86400，默认 60s。

SocketName

数据类型：string
用于接收数据的 Socket 名称。

示例

例 1

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。

12.3.8.7 SocketReadInt（过期的）

说明

通过 Socket 接收 int 型数据，外部发送的数据需以回车结尾。
注意，该指令被标注为"过期的"，为 xCore 控制系统 1.3 版本使用指令，在更高版本中依然有效，但不再继续维护，也不推荐继续使用。

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

数据类型：int
需要读取的 int 数个数，最大为 30 个。

TimeOut

数据类型：int
超时时间。单位 s，范围 0~86400，默认 60s。

SocketName

数据类型：string
用于接收数据的 Socket 名称。

示例

例 1

```
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。

12.3.8.8 SocketReadString（过期的）

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。
注意，该指令被标注为"过期的"，为 xCore 控制系统 1.3 版本使用指令，在更高版本中依然有效，但不再继续维护，也不推荐继续使用。

返回值

数据类型： string
存储接收到的字符串。

定义

SocketReadString(TimeOut, "SocketName")

TimeOut

数据类型： int

超时时间。 单位 s， 范围 0~86400，默认 60s。

SocketName

数据类型：string

用于接收数据的 Socket 名称。

示例

例 1

```
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。

12.3.9 逻辑指令

12.3.9.1 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。

12.3.9.2 Wait

说明

程序等待一段时间，范围是 0~2147484 秒。

示例

例 1

```
Wait 2
```

表示等待 2s 的时间。

12.3.9.3 WaitUntil

说明

程序等待某个条件成立。

示例

例 1

```
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。

12.3.9.4 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

例 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。

12.3.9.5 IF…Else if…Else

说明

条件判断语句。

示例

例 1

```
IF(condition1)
     //a
Else if (condition2)
     //b
Else if (condition3)
     //c
Else
//d
Endif
```

condition1 成立时执行逻辑 a，condition2 成立时执行逻辑 b，以此类推。

12.3.9.6 Goto

说明

Goto 语句允许把指针跳转到被标记的语句。

示例

例 1

```
int a = 0
int b = 9
Goto end
printf(a)
end:
printf(b)
```

先定义两个变量 a 和 b，然后用 printf 函数打印两句话，直接用 Goto 语句强制跳转到打印 b 语句的 end 标记位置，此时 a 的打印就不会执行了。

12.3.9.7 For

说明

For 循环允许您编写一个执行指定次数的循环控制结构。

示例

例 1

```
For(int i from 1 to 10)
    printf("i = %d\n", i)
endfor
```

该程序把 i 从 1 到 10 每次加 1 依次打印 10 次。

例 2

```
For(int i from 1 to 10 step 3)
    printf("i = %d\n", i)
Endfor
```

该程序把 i 从 1 到 10 每次加 3 依次打印 4 次。

补充说明

Continue 和 Break 可以用来控制 For 的流程,详细操作见 Continue 和 Break 指令说明。

12.3.9.8 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

例 1

```
VAR int count = 0
WHILE(1)
   count++
   IF(count == 1)
    Continue
   Else
    break
    MoveAbsJ j10， v500， fine， tool1
   Endif
ENDWHILE。
```

MoveAbsJ 的代码将不会被执行到。

12.3.9.9 Inzone

说明

该指令和 SetDO 或者 modbus、cclink 等 IO 操作或指令配合使用，可保证信号在确定的点位触发，不会被前瞻指针提前触发。

示例

```
MoveL p1
MoveL p2
Inzone
  SetDO dox, true
  print(123)
EndInzone
MoveL p3
```

补充说明

在示例中，使用了一个 Inzone 指令，解释器前瞻到 Inzone 之后，并不会立即执行，而是生成了一个附加函数，函数内容是 SetDO 以及 print 指令，这个附加函数会在运动指令 move p2 完成之后生效。

1、如果 p2 p3 两条运动指令之间存在转弯区，则附加函数会在机器人进入两段运动的转弯区的时刻开始执行

2、如果没有转弯区，则附加函数会在机器人到达 p2 的时刻开始执行

12.3.9.10 WHILE

说明

While 循环允许您编写一个在条件满足前不断执行的循环控制结构。

示例

例 1

```
int count = 0
while(count < 10)
    count++
    print(count)
endwhile
```

该程序实现一个 count 从 0 到 10 每次加 1 并打印的循环。

补充说明

Continue 和 Break 可以用来控制 While 的流程,详细操作见 Continue 和 Break 指令说明。

12.3.9.11 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。

提示

该指令暂时不支持辅助编程。

12.3.9.12 try/catch

说明

Try/Catch 指令允许程序编写者自主决定程序错误后的处理方式。

示例

例 1

```
ReadOnce:
Try
    Double xyz[3] = ReadDouble(3, timeout, socketname)
    Robtarget_0.trans.x = xyz[1]
    Robtarget_0.trans.y = xyz[2]
    Robtarget_0.trans.z = xyz[3]

    MoveL Robtarget_0, v2000, fine, tool0
Catch(error e)
    SendString("Recv rob xyz error", socketname)
    Goto ReadOnce
endtry
```

该程序实现了一个简单的应用场景，使用通信指令 ReadDouble 从 TcpSocket 读取一个三维数组作为运动点位的 xyz 参数，然后使用 MoveL 指令运动到对应笛卡尔点。
如果没有使用 try/catch 指令并且从 TcpSocket 收到点位是错误数据，则机器人会报错"超出运动范围"或者"规划错误"，并且停止程序的运行。
如果使用了 try/catch 指令，虽然依然会报告运动指令错误，但是程序不会停止，而是跳转到 catch 到 endtry 的代码段，执行用户想要的错误处理。本样例中就是通过 SendString 告诉 Socket 上位机收到的点位错误，再由上位机决定如何处理，并执行 goto 指令重新执行 ReadDouble 等待下一次的位置。

目前能被 try/catch 指令捕获错误的指令：

所有运动指令（见 12.3.2 运动指令）
所有通信指令（见 12.3.7 通信指令）

12.3.10 起始点指令

12.3.10.1 Home

说明

以轴空间运动让机器人回到设定好的初始点。

定义

Home
指令无参数

示例

例 1

```
HomeSet 0,30,0,60,0,90,0
Home
```

通过 HomeSet 指令设置初始点，再通过 Home 指令让机器人运动到轴空间的拖拽位姿。

使用限制

- 必须在机器人设置>快速调整 界面开启 Home 位姿设置或者通过 HomeSet 指令开启 Home 位姿设置，才可以使用 Home 指令，否则 Home 指令会报错。

12.3.10.2 HomeSet

说明

设定机器人的轴空间初始点位置

定义

HomeSet axis1,axis2,axis3,axis4,axis5,axis6,axis7

axisx

数据类型：Double

设定在初始点各个轴的角度

示例

例 1

HomeSet 0,30,0,60,0,90,0
Home

通过 HomeSet 指令设置初始点，再通过 Home 指令让机器人运动到轴空间的拖拽位姿。

12.3.10.3 HomeSetAt

说明

获得机器人的初始点的设定数据

定义

HomeSetAt(index)

返回值

数据类型：double

关节角，单位°

index

数据类型：int

获得初始点指定轴的关节角，当 index 为 0 时，返回是否开启了 Home 设置，1 表示已开启，0 表示未开启

示例

例 1

HomeSet 0,30,0,60,0,90,0
double angle2 = HomeSetAt(2)

angle2 获得关节 2 的关节角 30°。

12.3.10.4 HomeDef

说明

判断是否设置了初始点

定义

HomeDef()

返回值

数据类型：bool

true 已设置初始点

false 未设置初始点

12.3.10.5 HomeSpeed

说明

设定 Home 指令的运行速度

定义

HomeSpeed Speed

示例

例 1

HomeSpeed v1000
Home

设定初始点运动的速度为 V1000，随后 Home 指令让机器人按照 V1000 的速度往初始点运动。

12.3.10.6 HomeClr

说明

清除初始点设置

定义

HomeClr

示例

例 1

HomeClr

清除程序中设定的起始点。清楚后 Home 指令将无法执行。

12.3.11 数学指令

12.3.11.1 sin

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。

12.3.11.2 cos

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。

12.3.11.3 tan

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。

12.3.11.4 cot

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。

12.3.11.5 asin

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

12.3.11.6 acos

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。

12.3.11.7 atan

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

12.3.11.8 sinh

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。

12.3.11.9 cosh

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。

12.3.11.10 tanh

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。

12.3.11.11 exp

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。

12.3.11.12 log

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。

12.3.11.13 log10

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。

12.3.11.14 pow

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。

12.3.11.15 sqrt

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。

12.3.11.16 ceil

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。

12.3.11.17 floor

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。

12.3.11.18 abs

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。

12.3.11.19 rand

函数定义： rand()

函数说明：产生一个整型随机数；

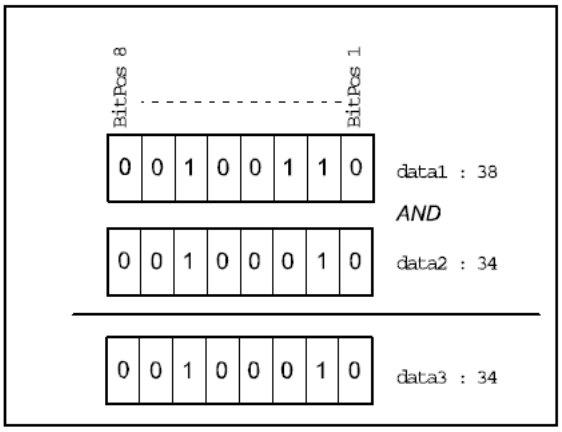
返回值：一个整型随机数，范围为 0~2147483647。

12.3.12 位操作

12.3.12.1 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

例 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。

12.3.12.2 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

例 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。

12.3.12.3 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

例 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。

12.3.12.4 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

例 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。

12.3.12.5 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

例 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。

12.3.12.6 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

例 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。

12.3.12.7 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

例 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。

12.3.12.8 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

例 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。

12.3.12.9 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

例 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。

12.3.13 字符串操作

12.3.13.1 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

例 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。

12.3.13.2 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

例 1

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

字符串"Robotics"长度为 8。

12.3.13.3 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

例 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。

12.3.13.4 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

例 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。

12.3.13.5 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

例 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。

12.3.13.6 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

例 1

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

12.3.13.7 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

例 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”。

12.3.13.8 StrSplit

说明

StrSplit 可以对字符串进行分割，通过指定分隔符，将字符串分割成字符串数组

返回值

数据类型：string 数组

表示分割得到的字符串数组

定义

StrSplit (Str [, separator])

Str

数据类型：string

表示要被分割的原字符串。

separator

数据类型：string

分隔符，该字符串中的所有字符都被看做分隔符，可以缺省，当缺省时，空格作为默认分隔符

示例

例

string str_arr[4] = StrSplit("test1,test2;test3\test4", "\,;")

字符串分割结果为四个子串(test1 test2 test3 test4)。

使用限制

- 输入字符串为空时会报错。
- 分割结果和定义字符串的长度不匹配时会报错。

12.3.13.9 StrToByte

说明

StrToByte 可以将字符串转换为 byte 类型数据

返回值

数据类型：byte

字符串的转换结果。

定义

StrToByte (Str [, trans])

Str

数据类型：string

将要进行数据转换的字符串。

trans

数据类型：枚举

表示字符串的数学进制格式，可用参数为 \Bin（二进制），\Okt（八进制），\Hex（十六进制），\Char（字符）以及默认参数（无参数，十进制）

示例

例 1

Byte NumBin = StrToByte(“10”, \Bin)
Byte NumOkt = StrToByte(“10”, \Okt)
Byte NumBin = StrToByte(“10”)
Byte NumHex = StrToByte(“10”, \Hex)

依次将字符串 ”10” 按照二进制、八进制、十进制、16 进制转换为 byte 数字，结果依次为

2, 8, 10, 16。

例 2

Byte NumChar = StrToByte(“0”, \Char)

将字符”0”按照字符与 ASCII 的转换关系，转为 48。

使用限制

- 输入字符串不符合指定的数据格式时会报错。

12.3.13.10 StrToDouble

说明

StrToDouble 可以将字符串转换为 double 类型数据

返回值

数据类型：double

字符串的转换结果。

定义

StrToDouble (Str)

Str

数据类型：string

将要进行数据转换的字符串。

示例

例

Double NumDouble = StrToDouble(“3.1415926”)

将”3.1415926”字符串转换为 double 类型数据。

使用限制

- 输入字符串不符合指定的数据格式时会报错。

12.3.13.11 StrToInt

说明

StrToInt 可以将字符串转换为 Int 类型数据

返回值

数据类型：Int

字符串的转换结果。

定义

StrToDouble (Str)

Str

数据类型：string

将要进行数据转换的字符串。

示例

例

Int NumInt = StrToInt(“99”)

将”99”字符串转换为 Int 类型数据。

使用限制

- 输入字符串不符合指定的数据格式时会报错。

12.3.14 运算符

8.3.11.1 基础运算符

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：

例 1

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

例 2

++，--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，RL 不区分前后自加自减：

```
x = n++       //表示 n 先加 1，再将 n 的值赋给 x
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。

例 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：

例 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
```

例 2

定义一个 robtarget 变量 pt1

```
pt1.trans.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。

8.3.11.2 运算优先级

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	

12.3.15 时钟指令

12.3.15.1 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

例 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。

12.3.15.2 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

例 1

```
VAR clock clock1
ClkReset clock1
```

重置 clock1。

12.3.15.3 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

例 1

```
VAR clock clock1
ClkStart clock1
```

声明 clock1，启动计时器 clock1。

12.3.15.4 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

例 1

```
VAR clock clock1
ClkStart clock1
...
ClkStop clock1
```

停止计时器 clock1。

12.3.16 高级指令

12.3.16.1 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 [, Tool, Wobj])

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

Tool

数据类型： tool 工具

包含描述 Point 点位的工具坐标系信息

Wobj

数据类型： wobj 工件

包含描述 Point 点位的工件坐标系信息

示例

例 1

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

由于未指定工具工件，默认使用 tool0,wobj0 作为工具工件，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。

例 2

p2=RelTool(p1,100,0,30,20,0,0, tool5, wobj6)

将 p1 点沿 wobj6 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。

例 3

MoveL RelTool(p1, 100,0,30,20,0,0), v4000, fine, tool2, wobj4

RelTool 和 Move 指令配合使用，未指定特定的工具工件坐标系，将使用运动指令的 tool 和

wobj,将 p1 点沿 wobj4 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。

> 提示
> 该指令的可选参数 Tool 和 Wobj 暂时不支持辅助编程。

12.3.16.2 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，平移偏移量由 x、 y、 z 三个量来表示，姿态旋转偏移量由 Rx、 Ry、 Rz 表示。

返回值

数据类型： robtarget

偏移后的新位姿。

定义

Offs（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。

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

例 1

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。

提示

该指令暂时不支持辅助编程。

12.3.16.1 ConfL On/Off

说明

xMate 笛卡尔坐标有一组 conf 参数(cf1~7, cfx)，用户手动更改或写入的笛卡尔坐标点对应的 conf 数据可能时错误的，导致控制器无法解析出目标点的路径，但是存在部分场景，用户只关心机器人 TCP 点的位置而不关心姿态，此时可以使用 ConfL Off 解除 conf 限制，让控制器尝试计算出一组可行的 conf 参数（可能算不出来，导致运动指令失败）

示例

例 1

```
p1.trans.x = ....
MoveJ p1, v1000 ....
只修改了坐标，未修改 cf 参数，该指令很可能导致执行失败
...
ConfL Off
MoveJ p1, v1000 ....
关闭 conf 检查，机器人能够运动到 p1 点，但是姿态不确定
```

例 2

```
ConfL On
打开 conf 检查
```

12.3.16.2 AccSet

说明

当机器人搬运易碎物品时，可使用 AccSet 指令调节加速度了减速度以达到更加平顺的运动效果。

定义

AccSet acc, ramp

acc

数据类型：int

按照系统预设值的百分比大小指定加速度和减速度的大小，取值范围 30%~100%，其中 100%对应最大加速度，超过限制范围机器人会停止并报错。

ramp

数据类型：int

按照系统预设值的百分比大小指定加加速度（Jerk），取值范围 10%~100%，其中 100%对应最大加加速度，超过限制范围机器人会停止并报错。

示例

例 1

AccSet 50,15

加速度、加加速度设置为默认的一半。

注意事项

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

12.3.16.3 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

12.3.16.4 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

12.3.16.5 GetEndtoolTorque

说明

获取当前指令指定的工具坐标系下的末端工具力矩信息，用于力控任务的控制。

返回值

数据类型： TorqueInfo

末端力矩信息

定义

GetEndtoolTorque(tool, wobj [, type])

tool

数据类型： 工具参数

机器人工作工程中，手持的负载可能会随时变化，该参数应该提供当前机器人使用的工具

wobj

数据类型： 工件参数

机器人工作工程中，手持的负载可能会随时变化，该参数应该提供当前机器人使用的工件

type

数据类型： int 枚举

0 末端相对于世界坐标系的力矩信息

1 末端相对于法兰坐标系的力矩信息

2 末端相对于 tcp 点的力矩信息

示例

例 1

TorqueInfo tmp_info = GetEndtoolTorque(tool1, wobj1)
获得在 tool1 wobj1 条件下机器人末端工具的力矩信息结构体

print(tmp_info.joint_torque.measure_torque)
print(tmp_info.joint_torque.external_torque)
打印各个轴的测量力和外部力

print(tmp_info.cart_torque.m_torque)
打印笛卡尔空间力矩

print(tmp_info.cart_torque.m_force[0])
print(tmp_info.cart_torque.m_torque[0])
打印 X 方向的力和力矩信息

12.3.16.6 MotionSup

说明

用于打开、关闭碰撞检测功能

定义

MotionSup type [, level]

type

数据类型：关键字

on 打开，off 关闭

level

数据类型：string

MotionSup On 的额外参数，修改碰撞检测灵敏度

"High" 高碰撞灵敏度

"Medium" 中等灵敏度

"Low" 低灵敏度

示例

例 1

MotionSup On
//... 其他指令
MotionSup Off
开启碰撞检测后执行其他指令，执行完成后使用 MotionSup Off 关闭碰撞检测

例 2

MotionSup On, "High"
开启碰撞检测并设置检测灵敏度为高

12.3.16.7 CONNECT（过期）

说明

用于将中断标识和 TRAP 作用域关联。
中断是通过定制一个中断事件和分配一个中断标识来定义。因此，当事件发生时， TRAP 作用域执行。

定义

CONNECT Interrupt WITH TRAP

Interrupt

数据类型： intnum
中断描述符。
中断描述符必须是全局变量。

TRAP

数据类型： string
TRAP 作用域名称。

示例

例 1

```
VAR intnum test_int
PROC main()
CONNECT test_int WITH test_TRAP
ISignalDI di1, 1 , test_int
```

中断描述 test_int 被连接到 TRAP 作用域 test_TRAP。当 di1 变为高时，将会产生中断。换句话说，当信号 di1 变高时，作用域 test_TRAP 被执行

12.3.16.8 BreakLookAhead

说明

该指令通知控制系统取消前瞻，强制取消上一条运动指令和下一条运动指令之间的转弯区。
机器人 TCP 将运行至上一条运动指令的目标点后，再向下一个点位运动，不衔接转弯区。程

序指针也将等待 TCP 运行至上一条运动指令的目标点后，再继续往下前瞻扫描。

定义

BreakLookAhead
指令无参数和返回值。

示例

例 1

```
MoveL  P1,v1000,z50,tool0
BreakLookAhead
MoveL  P2,v1000,z50,tool0
MoveL  P3,v1000,z50,tool0
```

1）点位 P1 的转弯区设置为 z50，由于 BreakLookAhead 指令存在，此处前瞻将取消，转弯区也将取消，机器人 TCP 会精确运动至 P1 点，再向 P2 运动。而 P2 和 P3 之间没有 BreakLookAhead 指令，因此在 P2 点会前瞻并且保留 z50 转弯区向 P3 运动。

2）BreakLookAhead 指令与 wait 0 指令效果完全相同。

12.3.16.9 GetRobotMaxLoad

说明

获取当前型号机器人的最大负载值。

定义

```
Ret = GetRobotMaxLoad()
```

Ret

数据类型：int

最大负载

示例

例 1

```
int maxload = GetRobotMaxLoad()
print(maxload)
```

以 xMate 7 为例，返回值为 7。

12.3.16.10 GetRobotState

说明

获取控制系统当前的运行状态。用 4 个字节的 bit 信息来表示控制系统的状态，包括故障、紧急停止、安全门、操作模式、伺服模式、运动状态等信息。具体如下表：

序号	状态位	含义
1	Byte[1].bit[1]	1：控制系统未授权
2	Byte[1].bit[2]	1：控制系统可恢复故障
3	Byte[1].bit[3]	1：控制系统致命错误

4	Byte[1].bit[4]	1：伺服系统故障
5	Byte[1].bit[5]	1：伺服系统致命故障
6	Byte[1].bit[6]	1：紧急停止
7	Byte[1].bit[7]	1：安全门停止
8	Byte[1].bit[8]	保留
9	Byte[2].bit[1]	上电状态 ， 0 ：电机未上电；1 ：电机已上电
10	Byte[2].bit[2]	机器人运动状态 ， 0：空闲；1：运动中
11	Byte[2].bit[3]	操作模式， 0：手动模式；1：自动模式
12	Byte[2].bit[4]	伺服模式， 0：位置模式；1：力矩模式
13	Byte[2].bit[5]	保留
14	Byte[2].bit[6]	保留
15	Byte[2].bit[7]	保留
16	Byte[2].bit[8]	保留
17	Byte[3]	保留
18	Byte[4]	保留

定义

Ret = GetRobotState()

Ret

数据类型：byte 数组

用四个 byte 类型来表示机器人状态。

示例

例 1

```
byte st[4] = GetRobotMaxLoad()
print(st)
```

返回{0,5,0,0}，查表可知当前状态为：无故障，电机已上电，自动模式，伺服处于位置模式。

12.3.17 函数指令

12.3.17.1 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

例 1

```
p2 = CRobT( tool1， wobj2 )
```

12.3.17.2 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

例 1

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

12.3.17.3 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），旋转外部轴的单位是度（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

例 1

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。

12.3.17.4 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

例 1

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。

12.3.17.5 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

例 1

```
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。

12.3.17.6 PoseMult

说明

PoseMult 用于计算两个位姿变换的乘积。

定义

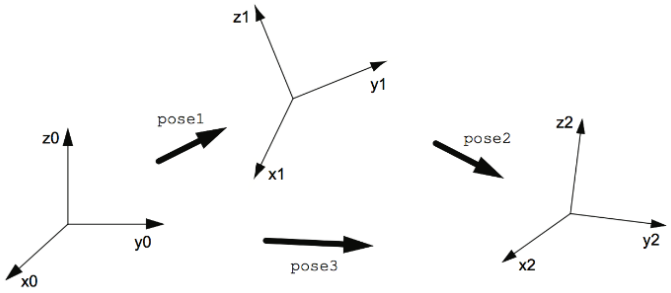
pose3 = PoseMult(pose1, pose2)

参数说明：pose1 和 pose2 为 pose 类型输入，pose3 为 pose 类型的返回值。

示例

pose1 代表坐标系 1 相对坐标系 0 的位姿，pose2 代表坐标系 2 相对坐标系 1 的位姿，坐标系 2 相对坐标系 0 的位姿 pose3 可以通过以下方式计算：

```
VAR pose pose1
VAR pose pose2
VAR pose pose3
...
pose3 = PoseMult(pose1, pose2)
```



12.3.17.7 PoseInv

说明

PoseInv 用于计算一个位姿变换的逆。

定义

pose2 = PoseInv(pose1)

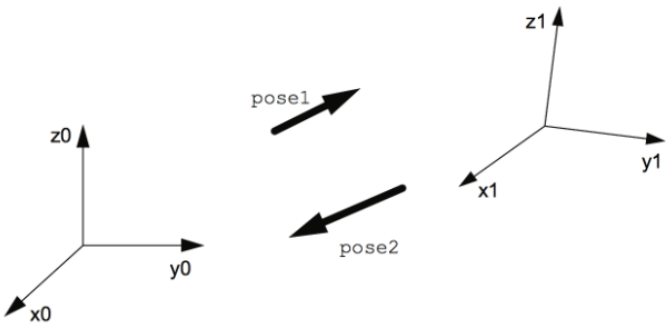
参数说明：pose1 为 pose 类型输入，pose2 为 pose 类型的返回值。

示例

pose1 代表坐标系 1 相对坐标系 0 的位姿，pose2 代表坐标系 0 相对坐标系 1 的位姿。

如果已知 pose1，pose2 可以通过以下方式计算：

```
VAR pose pose1
VAR pose pose2
...
pose2 = PoseInv(pose1)
```



12.3.18 寄存器指令

12.3.18.1 ReadRegByName

说明	
	根据寄存器的名字，读取对应寄存器的变量值
定义	
	ReadRegByName(RegData，Value)
RegData	
	数据类型：可读的寄存器变量 设置>通信>寄存器界面功能，寄存器变量。
Value	
	数据类型：bool/int/double 寄存器数据将写入 Value，如果寄存器的变量类型和解释器变量不一致，将自动进行格式转换
示例	
例 1	
	int tmp_int ReadRegByName(modbus_int_read[6], tmp_int) 将名为 modbus_int_read 的下标为 6 的数据，读取到 tmp_int 变量中

12.3.18.2 WriteRegByName

说明	
	根据寄存器的名字，读取对应寄存器的变量值
定义	

WriteRegByName(RegData，Value)

RegData

数据类型：可写的寄存器变量
设置>通信>寄存器界面功能，寄存器变量。

Value

数据类型：bool/int/double
将 Value 数据写入寄存器中，如果寄存器的变量类型和解释器变量不一致，将自动进行格式转换

示例

例 1

WriteRegByName(modbus_int_write[6], 200)
将 INT 200 的数据，写入 modbus_int_write[6] 对应的寄存器中。

公众号：ROKAE珞石 微信号：Rokae-tech

网址：http://www.rokae.com

2小时售后服务电话

010-62967922

北京市海淀区农科院西路6号海青大厦A座7层

济宁市邹城市中心店镇机电产业园恒丰路888号

苏州工业园区星湖街328号创意产业园1-A1F

深圳市宝安区中粮福安机器人智造产业园10栋1楼