Feel U

新一代柔性协作机器人

xMate

工业技术进步是人类社会发展变革的推动力

生产的自动化、智能化正在改变当今全球工业的生产效率

工业机器人的诞生与进化——由简至精、由刚到柔

正在不断缩短与人的距离

珞石新一代柔性协作机器人——xMate

以高灵敏度力感知&高动态力控

成为人类工业生产得力的合作伙伴，为柔性智造赋能

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 界面介绍

界面功能划分

A	菜单栏，各功能模块操作入口，包括工程、机器人、诊断、演示、选项和帮助。
B	主操作区，各功能模块操作界面。
C	顶部状态栏，显示当前工程、即时日志、工具工件信息、状态监控和运动操作界面入口按钮。
D	底部状态栏，显示连接状态，运行速度、运行状态、用户信息及机器人型号信息。
E	状态监控窗口，可切换显示运动状态、任务信息以及 IO 信号等监控状态。
F	运动操作窗口，用于拖动或者 JOG 机器人。

> 提示
>
> 1、 HMI 与控制器交互较多，频繁改变窗口大小可能导致界面停止更新，此时点击 alt+tab 切换窗口即可恢复。
> 2、 HMI 软件仅支持 64 位系统，可支持 win7、win8、win10。

1.1 顶部状态栏

说明

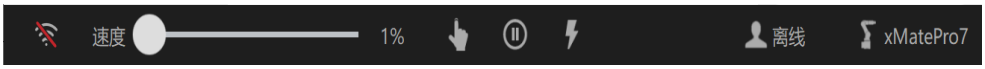
顶部状态栏显示菜单栏、当前工程、即时日志、工具工件信息、状态监控和运动操作界面入口按钮。

A	菜单栏按钮，点击按钮可选择需要切换的功能模块。
B	当前工程，点击进入当前工程相关信息。
C	即时日志，显示最近一条系统日志。
D	工具信息，显示、设置当前使用工具信息。
E	工件信息，显示、设置当前使用工件信息。
F	状态监控按钮，用于打开关闭状态监控窗口。
G	运动窗口按钮，用于打开关闭运动窗口。

1.2 底部状态栏

说明

底部状态栏显示机器人连接状态、程序运行速度、机器人工作模式、机器人运行状态、电机上电状态、当前用户信息和机器人型号信息。



A	HMID 与机器人控制器的连接状态，出现红色斜杆为未连接，全灰色为已连接。
B	程序运行速度调整滑块。
C	工作模式，分为手动(手指图标)，自动。
D	机器人运动状态，包括运动、停止等。
E	机器人电机上电状态，红色为上电，灰色为下电。
F	当前用户信息：操作员、管理员、超级管理员。
G	机器人型号信息。

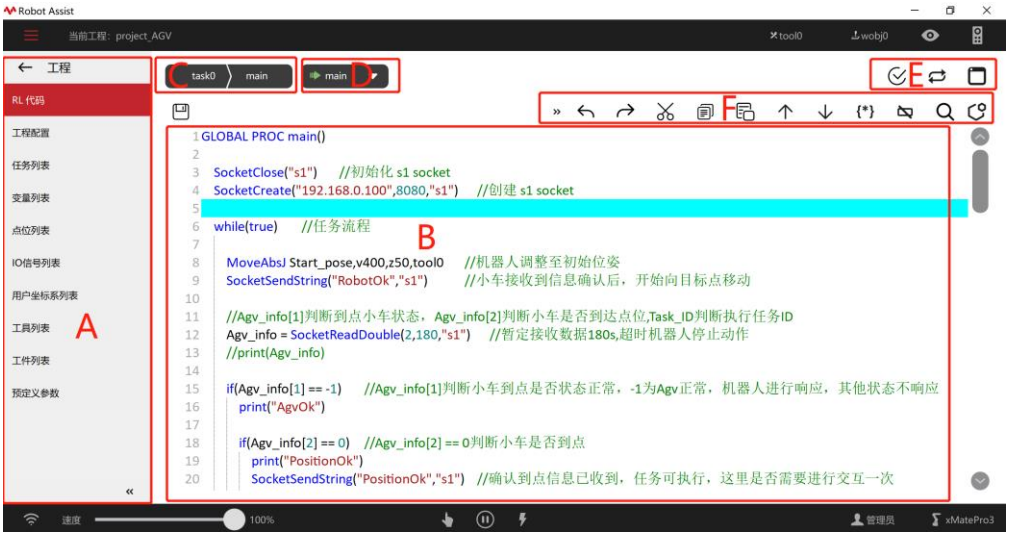
机器人运行状态

底部状态栏最中间图标，展示当前机器人运行状态，包括程序运行状态、机器人状态、控制器所处模式等。

空闲状态		程序停止，机器人未运动。
程序运行		程序运行，如果机器人运动会变成红色。
拖动示教		控制器处于拖动模式，可以进行拖动，如果机器人运动会变成红色。
demo 模式		控制器处于 demo 打开模式，可以进行 demo 演示，如果机器人运动会变成红色。
辨识模式		辨识状态，如果机器人运动会变成红色。
JOG 模式		jog 模式，如果机器人运动会变成红色，随 jog 按键开始结束而改变。
RCI 模式		RCI 开启状态，如果机器人运动会变成红色。
协作模式		协作模式开启状态，会与其他状态进行组合显示再图标的右上角，如
错误状态		机器人处于错误状态
调试模式		机器人处于调试模式，如果机器人运动会变成红色

1.3 工程

界面功能划分



A	侧边栏，用于切换工程子对象设置界面，包括 RL 程序、工程配置、任务列表、变量列表、点位列表、IO 信号列表、用户坐标系列表、工具列表、工件列表。
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、向上移动光标、向下移动光标、批量注释、搜索、辅助编程、快速删除行、上移一行、下移一行。

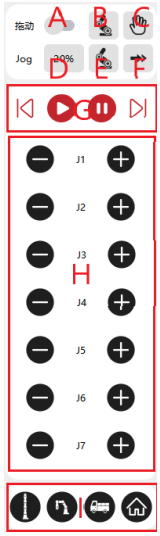
1.4 运动窗口

说明

运动界面用于示教调试机器人位姿，并显示机器人运动状态。

xMate 机器人支持两种方式示教机器人位姿：JOG 和拖动方式。

- JOG 方式，通过 JOG 按钮控制相应方向运动。
- 拖动方式，通过拖动 Pilot 手柄，直接引导机器人运动。



A	拖动启用按钮，用于开启拖动功能。
B	拖动空间选择，可选择轴空间拖动、笛卡尔空间拖动。

C	笛卡尔空间拖动自由度设置，可选择仅平移拖动、仅旋转拖动以及自由拖动。
D	JOG 目标速度调整，调整范围 1~100%，相对 JOG 最高速度限制 250mm/s 的百分比。
E	JOG 坐标系选择，包括：世界坐标系、基坐标系、法兰坐标系、工具坐标系、工件坐标系和轴空间。
F	JOG 模式设置：连续模式、点动模式。
G	程序运行调试按键，包括向上一步、启动/停止、向下一步。
H	JOG 按键，轴空间 JOG 时显示 J1~J7，笛卡尔空间 JOG 时显示 X/Y/Z/AX/AY/AZ 及 Elbow。
I	快速位姿调整，包括运动到机械零位、拖动准备位姿、发货姿态以及自定义 home 点。

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：包括世界坐标系、法兰坐标系等。

JOG 方式可选连续运动或者点动：

- 连续运动模式下，使能和 JOG 按键同时按下，机器人在设定 JOG 速度下连续运动直至松开使能或者 JOG 按键。
- 点动模式下，每点击一次 JOG 按键，机器人在上电状态，机器人以固定步长运动。并可通过设置点动模式的 JOG 步长，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）

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，只能手动引导机器人绕 TCP 做姿态旋转，仅平移模式，可控制机器人沿笛卡尔空间做平移运动。

快速调整

对于常用的机器人位姿，HMI 运动界面提供便捷调整功能，目标位姿包括：机械零位、拖动准备位姿、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，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。

1.5 状态监控

说明

状态监控用于监控机器人 3D 模型、任务运行状态和 IO 信号。

机器人 3D 模型界面显示机器人当前的三维模型、实时关节角度、实时关节力矩、机器人臂角、机器人末端在基坐标系下的位置、机器人末端相对于世界坐标系的旋转矩阵的 RPY 角和四元数。

任务界面显示各个任务的运行状态。

IO 信号界面显示 xMate 底座上的 4 路 DI/DO 信号以及末端 2 路 DI/DO 信号，界面上支持选择 IO 仿真模式测试 DI/DO 信号值。

1.5.1 3D 模型

说明

机器人 3D 模型界面显示机器人当前的三维模型、实时关节角度、实时关节力矩、机器人臂角、机器人末端在基坐标系下的位置、机器人末端相对于世界坐标系的旋转矩阵的 RPY 角和四元数以及参考坐标系。

1.5.2 任务

说明

实时观察各任务的运行状态。

任务操作

	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。

1.5.3 IO 信号

说明

实时监测所有 IO 值，并能够进行仿真操作。

IO 仿真操作

	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，可以对其进行仿真。	注意，非仿真模式 DO 也可以强制输出。
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。

1.5.4 网络连接

说明

实时监测已建立的网络连接，包括 socket、modbus、rci 等。

网络连接状态

	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。

1.5.5 寄存器变量

说明

实时监测 modbus 寄存器变量的值。

寄存器变量

	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。	

2 连接

说明

运动界面用于示教调试机器人位姿，并显示机器人运动状态。

xMate 机器人支持两种方式示教机器人位姿：JOG 和拖动方式。

2.1 连接方式

说明

- xMate 机器人系统只支持有线形式接入到以太网中。
- 对于 1 台 HMID 调试一台机器人的情况，HMID 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，HMID 通过同网段探测可连接的机器人。
- 对于 AGV 搭载等不方便有线连的场景，可以通过机器人与无线路由连接后，再与 HMID 通过无线连接。

网线直连

机器人底座上的 J1 网口，配置了一个固定 IP 地址，该地址在所有控制器上均相同，且不可更改，您可以将 HMI 所在 PC 通过网线直连，并配置连接 ip 为 192.168.0.160。

外网口连接

机器人底座上的 J2 网口支持两种连接模式：dhcp、Static。

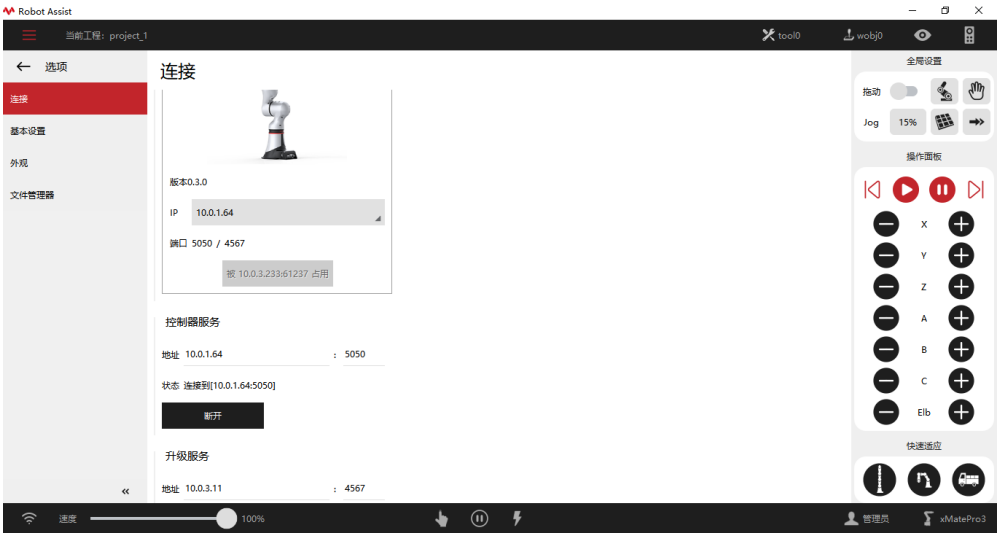
J2 网口默认 dhcp 连接模式，机器人通过 J2 网口接入到 DHCP 功能的路由中，用于自动分配 IP 地址给机器人，这时可以通过机器人探测功能探测到机器人并连接。

用户可在控制器设置>系统 IP 属性栏，选择 static 模式，配置机器人静态 IP，此时机器人 J2 网口的 IP 地址为用户配置的固定 IP。

> **警告**
>
> 使用 static 模式为机器人 J2 网口配置固定 IP 时，切勿将 IP 地址配置成与 J1 网口相同的网段（192.168.0.xxx）。

将 HMI 连接控制器

进入 HMI->选项->连接界面，依次输入机器的 IP 地址



> **提示**

IP 和端口是连接控制器的途径，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。

2.2 搜索可连接机器人

说明

HMI 可检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 HMID 和机器人处于同一网络中，网络互通。
2、 HMID 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 HMID 防火墙屏蔽机器人发送的 UDP 信号。

2.3 切除与恢复连接

说明

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

2.4 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，HMI 一直重连控制器。

3 运行模式

3.1 安全管理

3.1.1 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。

3.1.2 安全术语

3.1.2.1 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。

3.1.2.2 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；

3.1.2.3 停止处理

停止处理类型

xMate 支持两种停止处理类型：紧急停止和可控性停止。

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。

可控性停止

可控停止是机器人在不断电的情况下，停止程序运行。

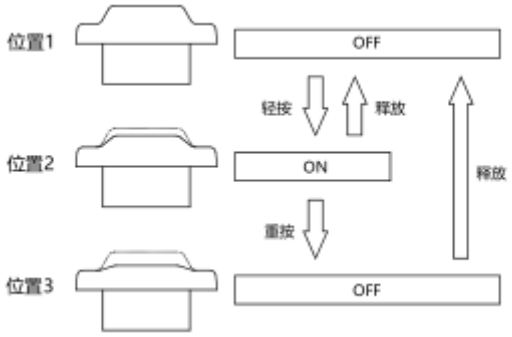
- STOP 2 停止，机器人通电时的可控性停止。安全评定控制系统的操控可使机器人停留在停止的位置。

3.1.2.4 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

手持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！

3.1.3 工作中的安全事项

3.1.3.1 概述

关于机器人

3.1.4 xMate 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。

3.1.5 力控参数

说明

力控参数即根据基座刚度等级（高、低）提供的两组适用的力控制参数，如果设置基座刚度，机器人将根据基座刚度等级适应性的调节控制参数，否则使用默认的控制参数。

绝大多使用场景下均无需调整此参数，只有当因为基座不稳，导致机器人在拖动或是阻抗运动过程中严重的存在抖动现象时，可以将基座刚度等级调节为低。例如：AGV 小车底座使用场景。

3.1.6 快速调整

说明

对于常用的机器人位姿， HMI 运动界面提供便捷调整功能，目标位姿包括：机械零位、拖动、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

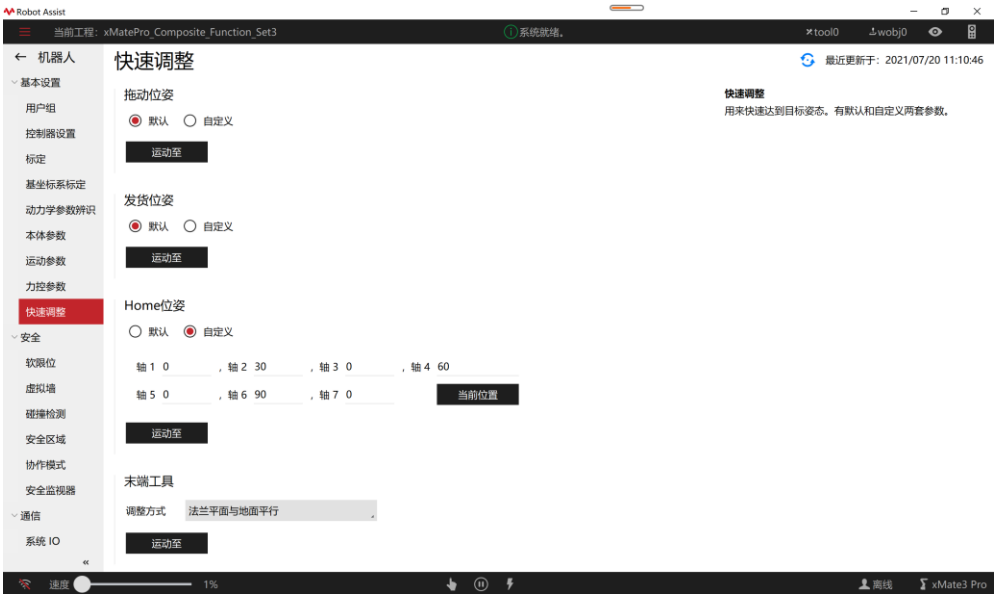
快速位姿调整在手动模式下使用，使用方式与 JOG 操作类似，手动模式下通过使能将机器人上电，按下相应目标位姿按钮，机器人将通过轴空间运动至目标位姿。

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。



3.1.7 安全功能）来保障用户与 xMate 协作时的人身安全。请确保您已经熟悉阅读运动参数

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。

3.1.8 力控参数

说明

力控参数即根据基座刚度等级（高、低）提供的两组适用的力控制参数，如果设置基座刚度，机器人将根据基座刚度等级适应性的调节控制参数，否则使用默认的控制参数。

绝大多使用场景下均无需调整此参数，只有当因为基座不稳，导致机器人在拖动或是阻抗运动过程中严重的存在抖动现象时，可以将基座刚度等级调节为低。例如：AGV 小车底座使用场景。

3.1.9 快速调整

说明

对于常用的机器人位姿， HMI 运动界面提供便捷调整功能，目标位姿包括：机械零位、拖动、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

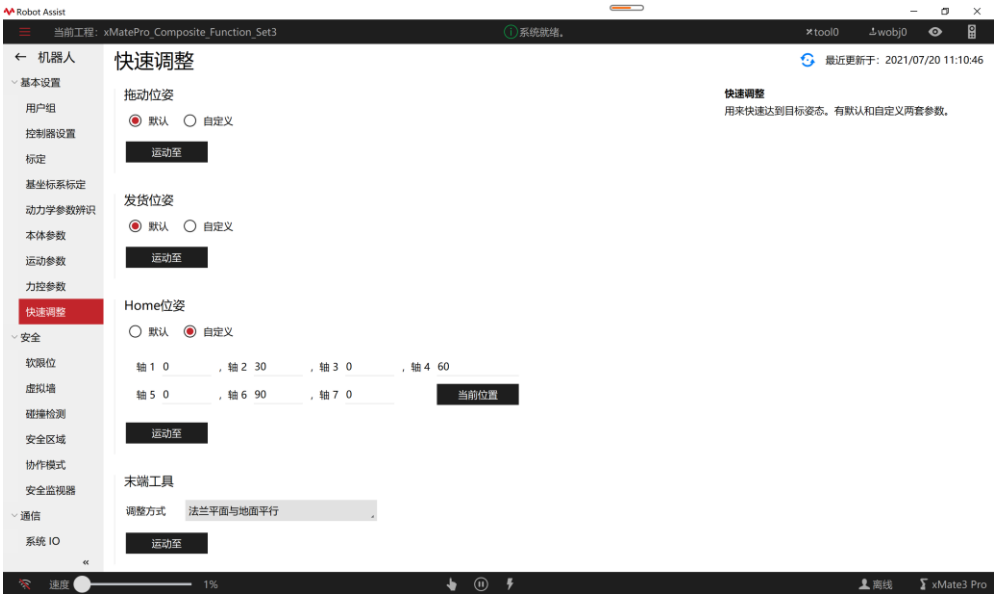
快速位姿调整在手动模式下使用，使用方式与 JOG 操作类似，手动模式下通过使能将机器人上电，按下相应目标位姿按钮，机器人将通过轴空间运动至目标位姿。

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。



安全功能的安全功能介绍，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。

3.1.9.1 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。

3.1.9.2 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。

3.1.9.3 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。

3.1.9.4 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。

3.1.9.5 紧急情况的处理

3.1.9.5.1 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。

3.1.9.5.2 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。

3.2 运行模式

3.2.1 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（ROKAE Robot Language）程序；
- 调试 RL 程序，包括但不限于启动、停止、单步运行、更新程序位置等；
- 设置控制系统参数、标定坐标系等；
- 查看、修改变量；

3.2.2 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，可以通过系统 IO 设置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 列表参见系统 IO 设置。

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；

通常在自动模式下执行的任务

1. 自动模式下，将无法进行 Jog。
2. 自动模式下不允许修改配置文件，配置 IO 板个数，设置机器人安装方式。
3. 自动模式下不允许备份恢复。
4. 自动模式下不允许功能授权。

5. 自动模式下不允许设置软限位。
6. 自动模式下不允许新建，修改，删除 IO 及配置系统 IO 设置。
7. 自动模式下不允许参数辨识。
8. 自动模式下不允许开启/关闭碰撞检测。
9. 自动模式下不允许开启/关闭协作模式。
10. 自动模式下不允许开启/关闭拖动示教。
11. 自动模式下不允许标定。
12. 自动模式下不允许新建变量。
13. 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。

3.2.3 模式切换

3.2.3.1 关于模式切换

当前模式

通过观察 HMI 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）。

3.2.3.2 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。

3.2.3.3 自动切换到手动

从自动模式切换到手动模式

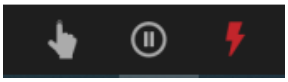
点击 HMI 界面上的  图标将自动模式切换到手动模式，观察图标是否切换成  。如果是，那么模式切换已经完成。如果切换失败，根据顶部状态栏的实时日志信息排除故障。

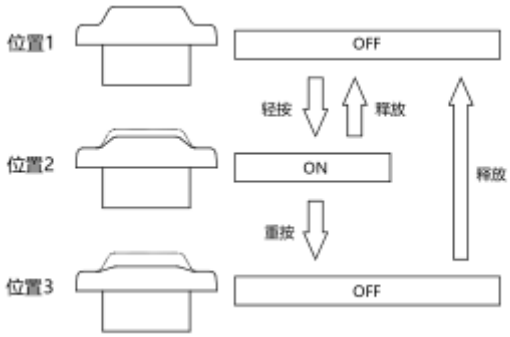
3.3 机器人上下电

3.3.1 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 HMI 界面上的底部状态栏上电按钮变成红表示上电成功。



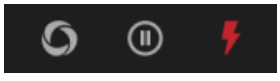


提示

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 HMI 界面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。



3.3.2 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 HMI 界面底部状态栏的上下电按钮给电机上电。

> 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。

4 机器人

4.1.1 基本设置

4.1.2 用户组及权限

说明

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，admin 和 god，分别对应操作员、管理员、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，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

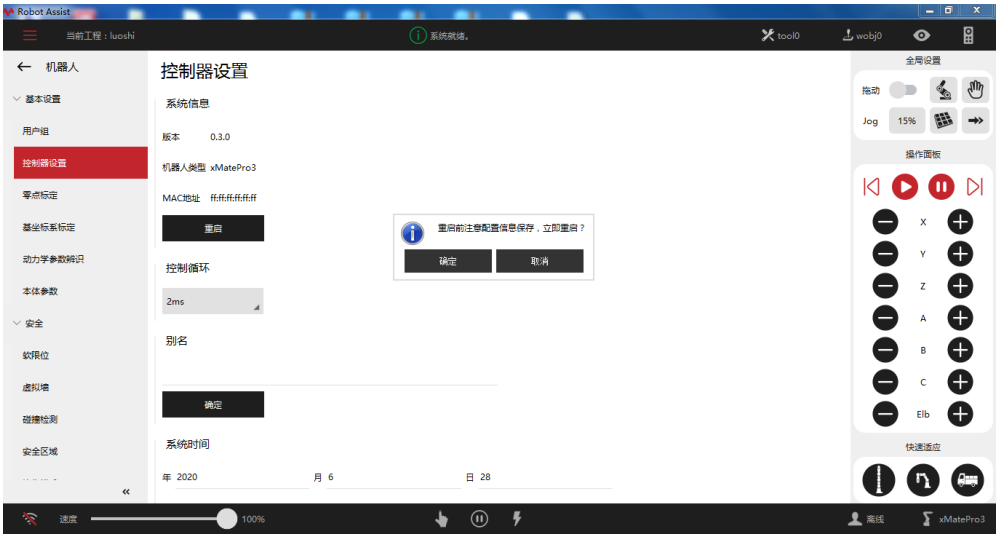
类别	功能	操作员	管理员	超级管理员
工程相关	工程管理（新建、导入、导出）	N	Y	Y
	查看工程（包括程序和 IO、变量等对象数据）	Y	Y	Y
	编辑工程（包括程序编辑和工具等对象设置）	N	Y	Y
机器人运动与程序运行	模式切换	N	Y	Y
	上/下电	N	Y	Y
	启动/停止程序	Y	Y	Y
	调整程序运行速度	N	Y	Y
	单步调试程序	N	Y	Y
	APP 运行	Y	Y	Y

	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

4.1.3 控制器设置

系统信息

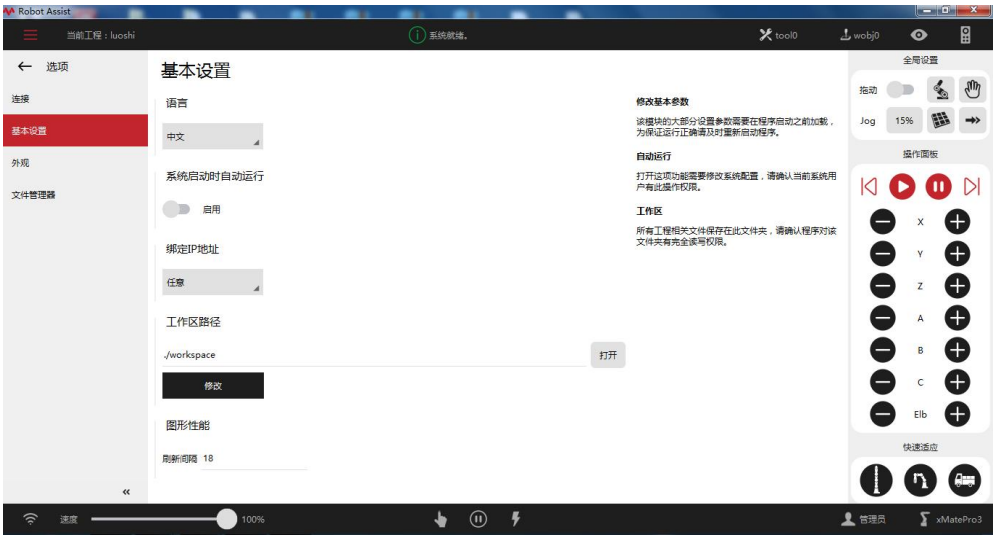
对控制器进行重启，用户在重启前应保存配置信息。



系统配置

HMI 系统语言支持中、英文。

用户点击切换后需要重启 HMI 软件



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。

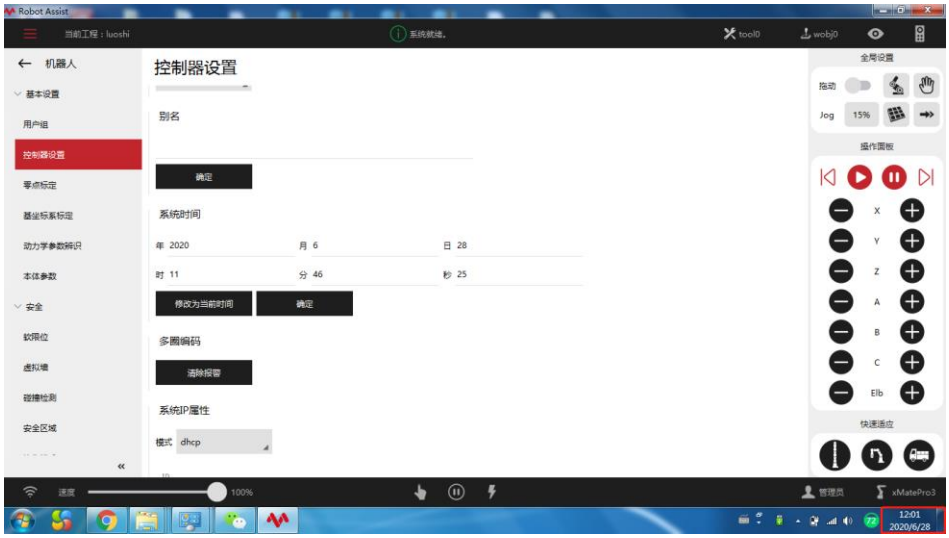
系统时间

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。

支持手动修改和修改为 HMID 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 HMID 当前的系统时间。

当 HMI 界面上显示的系统时间与 HMID 界面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 HMID 界面右下角的系统时间。



警告

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。

多圈编码

清除编码器多圈报错。

系统 IP 属性

对机器人外网口连接模式进行配置，具体使用说明参见连接方式。

日志保存级别

设置日志保存级别。

4.1.4 零点标定

说明

xMate 零点标定功能包括机械零点标定和力传感器零点标定，零点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

1、 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！

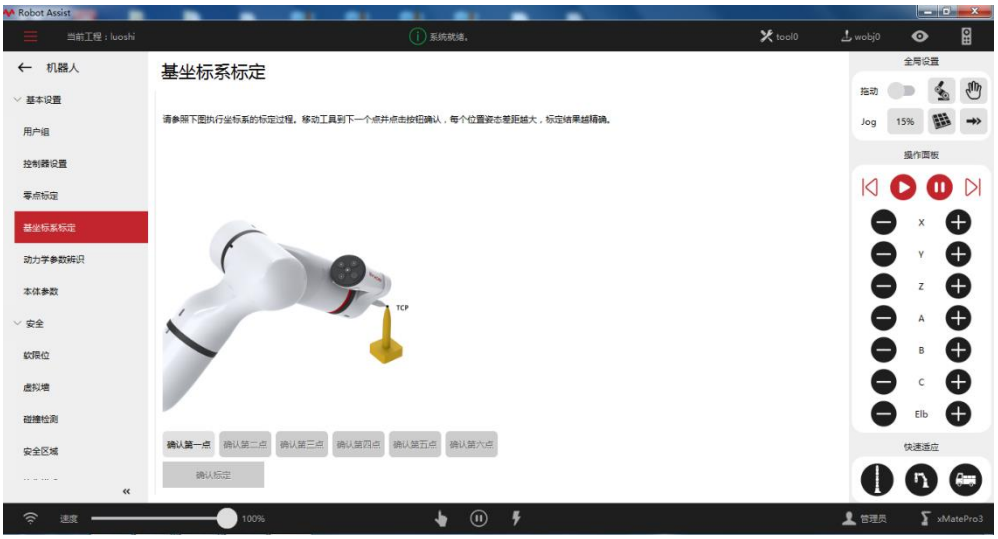
4.1.5 基坐标系标定

什么是基坐标系?

机器人支持正装、墙装、倒装的安装方式，默认正装方式。如果用户需要更改安装方向，HMI界面上会提示用户重新设置基坐标系。

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。

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 6 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。

4.1.6 动力学参数辨识

说明

xMate 动力学参数指出厂设置或是动力学参数辨识功能获取的机器人动力学模型参数，动力学模型参数主要用于机器人力控、拖动示教、虚拟墙和碰撞检测功能。运行这些功能前，请确保机器人动力学参数已经正确辨识，否则会出现上述功能无法正常使用，或是异常抖动的现象。

4.1.7 本体参数

说明

DH 参数是用来描述机器人连杆之间相对位姿关系的一组参数，是机器人运动学的基础。机器人出厂前存在默认一组参数，修改 DH 参数可能造成无法运动的情况，请谨慎修改。用户在修改参数或者导入参数之前应该检查标定后的 DH 参数合理性。参数修改完成后点击保存或者启用 DH 参数，再重启控制器。

4.1.8 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。

4.1.9 力控参数

说明

力控参数即根据基座刚度等级（高、低）提供的两组适用的力控制参数，如果设置基座刚度，机器人将根据基座刚度等级适应性的调节控制参数，否则使用默认的控制参数。

绝大多使用场景下均无需调整此参数，只有当因为基座不稳，导致机器人在拖动或是阻抗运动

过程中严重的存在抖动现象时，可以将基座刚度等级调节为低。例如：AGV 小车底座使用场景。

4.1.10 快速调整

说明

对于常用的机器人位姿， HMI 运动界面提供便捷调整功能，目标位姿包括：机械零位、拖动、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

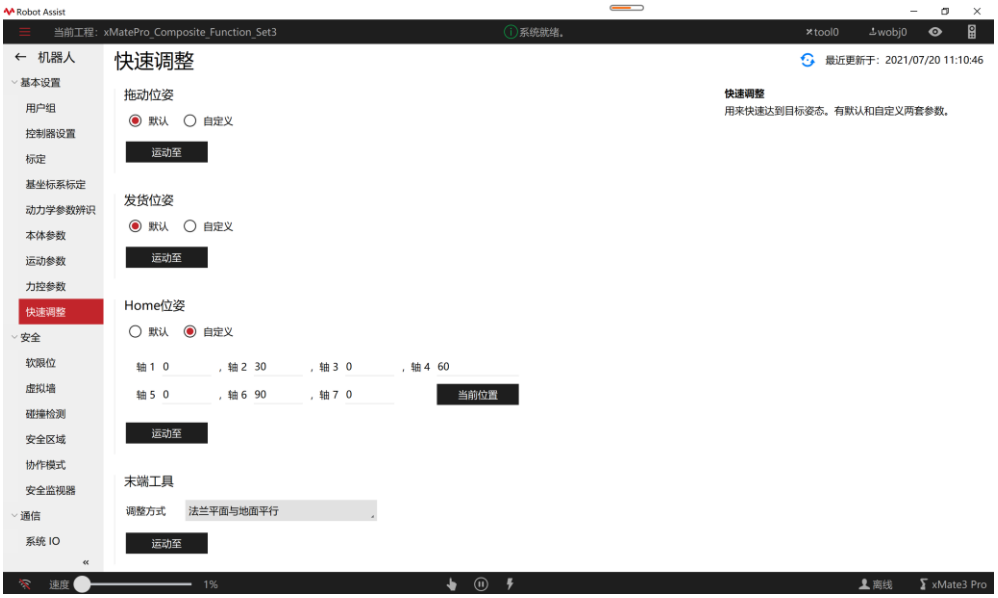
快速位姿调整在手动模式下使用，使用方式与 JOG 操作类似，手动模式下通过使能将机器人上电，按下相应目标位姿按钮，机器人将通过轴空间运动至目标位姿。

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。



4.2 安全功能

4.2.1 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。

警告

1. 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。
2. 基于安全考虑，一般在设置软限位范围时，需要在机械硬限位的正负边界各预留 5°的活动范围。
3. 机械臂的机械硬限位范围参见各机型产品手册。

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
>
> 1. 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，
> 2. 软限位取消时，RL 程序无法运行。

4.2.2 虚拟墙

功能说明

针对一些医疗场景，xMate 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。

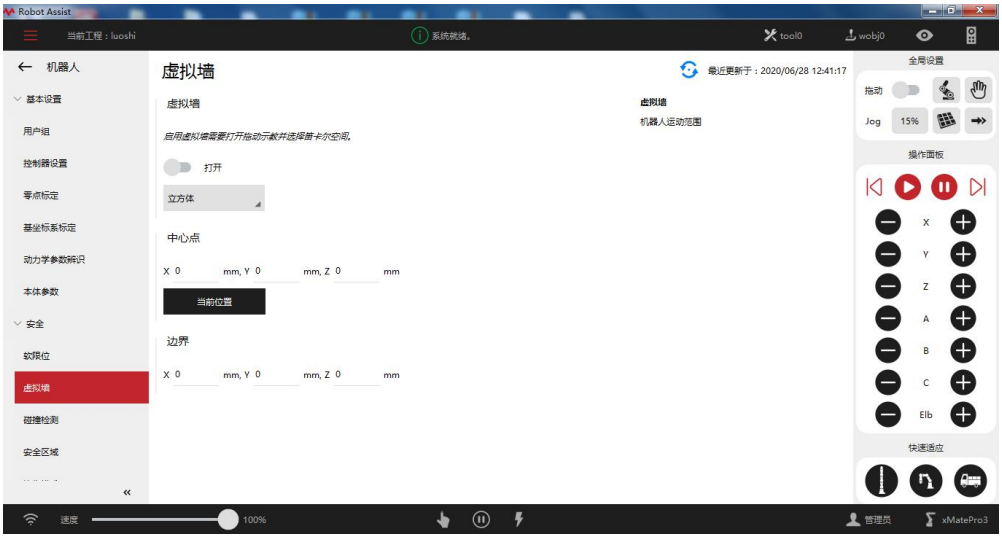
1.支持虚拟墙形状类型长方体、球形。

2.虚拟墙中心位置和限制范围可设置：

Ø 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。

Ø 球形限制范围采用球半径设置，单位 mm。

Ø 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。

4.2.3 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，xMate 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。

1.碰撞检测功能可启用或者关闭，默认关闭状态。

2.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。

3.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

4.2.4 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。

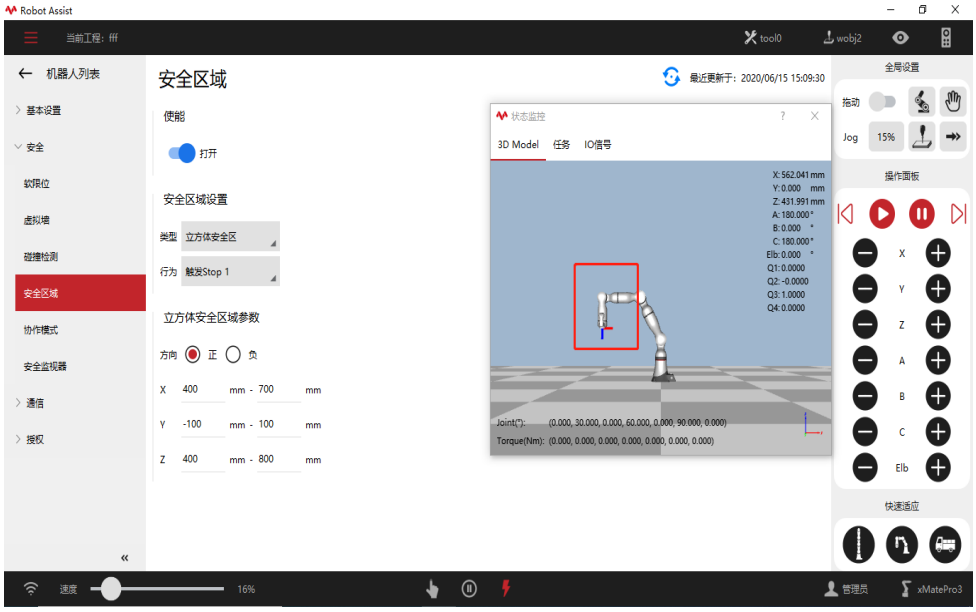
示例

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

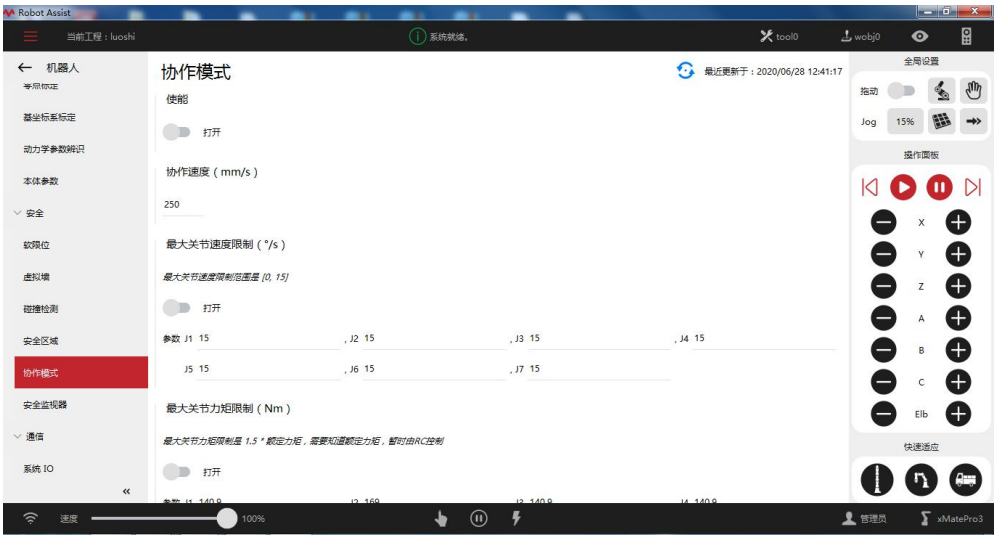
4.2.5 协作模式

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。

4.2.6 安全监视

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。

4.3 通信配置

4.3.1 系统 IO 设置

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
1	电机上电指令
2	电机下电指令
3	程序启动指令
4	程序停止指令
5	程序指针到 main
6	进入协作模式

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
1	电机上电状态	电机上电	电机下电
2	程序运行状态	程序运行	程序未运行
3	工作模式	自动模式	手动模式/等待模式
4	急停状态	急停状态	非急停状态
5	碰撞触发状态	触发	未触发
6	协作模式状态	协作模式状态	非协作模式状态

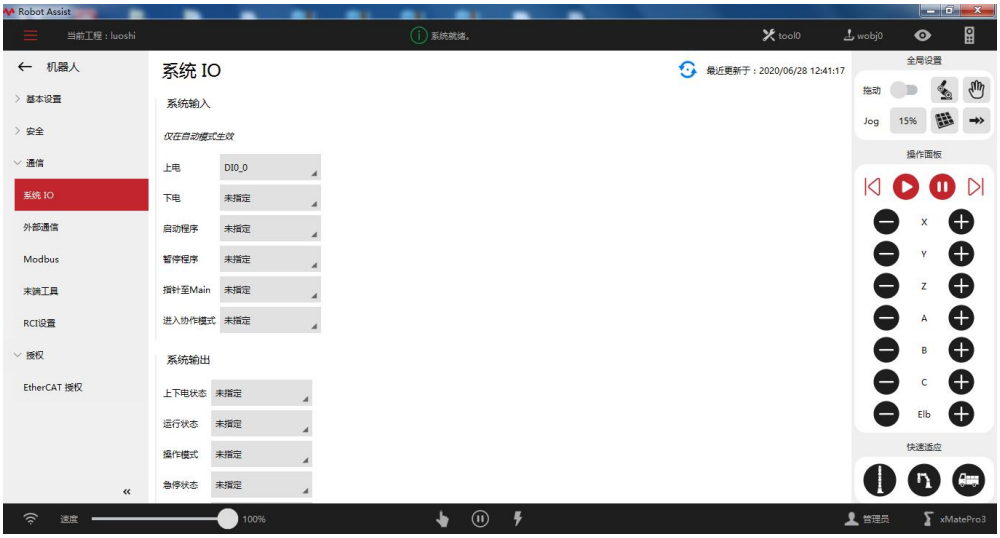
除"工作模式"信号外，其他所有的系统输出均是高电平有效。

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。



4.3.2 Socket 通信外部控制

说明

1.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。

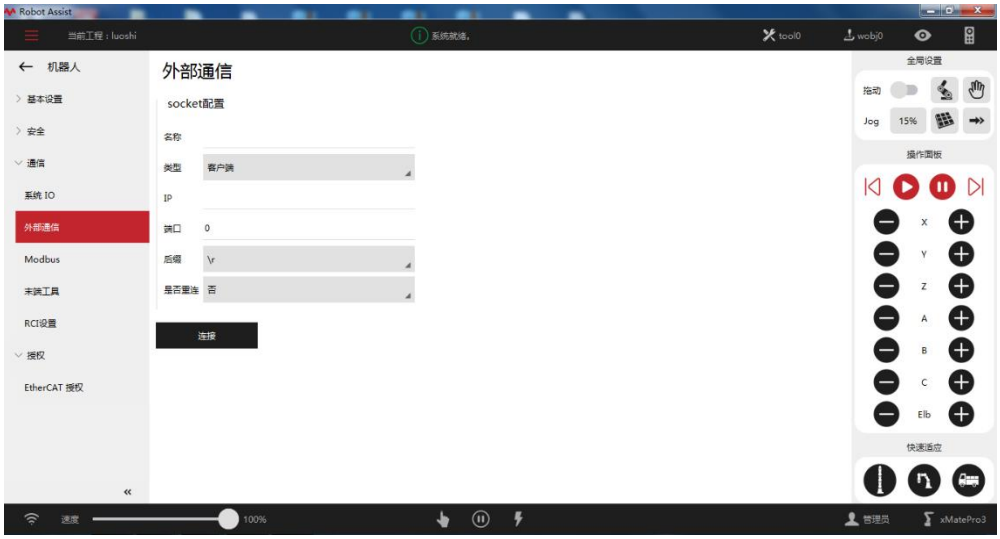
2.Socket 通信接口支持界面：配置 IP 地址、端口号、Socket 名称。

3.支持 Client 模式。

4.支持 Socket 断线重连功能。

启动接口

在使用各项交互指令之前，首先需要创建用于信息交互的 Socket，该项操作在示教器的界面上操作，需要输入的参数包括上位机 IP、端口以及要创建 Socket 的名称，如下图所示：



交互命令列表

下表中给出了外部通信接口支持的信息内容及对应的命令格式，xCore 系统使用"\r"作为指定的命令结束符（"\r"是转义字符，表示回车，十进制值是 13）。交互命令包括控制命令，配

置命令和监控命令。

控制命令包括：

	指令名称	发送的字符串	返回值
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	PPToMain	″pp_to_main″+″\r″	成功返回″true″ ;失败返回″false″
6	电机上电指令	″motor_on″+″\r″	成功返回″true″;失败返回″false″
7	电机下电指令	“motor_off” + “\r”	成功返回″true″;失败返回″false″
8	切换至手动模式	“switch_mode:manual”+”\r”	成功返回″true″;失败返回″false″
9	切换至自动模式	“switch_mode:auto”+”\r”	成功返回″true″;失败返回″false″

支持监控命令：

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″

备注：

1.笛卡尔位置字符串格式：x、y、z、a、b、c、q1、q2、q3、q4；其中 x、y、z 单位为 mm，a、b、c 单位为度，q1~q4 是姿态的四元数表示；

2. 轴位置字符串格式：j1、j2、j3、j4、j5、j6、j7；其中机器人轴角度单位是弧度，导轨位置单位是米；

3. 轴速度字符串格式：vj1、vj2、vj3、vj4、vj5、vj6、vj7；其中机器人轴速度单位是弧度/秒，导轨速度单位是米/秒；
4. 轴力矩字符串格式：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；

4.3.3 Modbus 寄存器

说明

Modbus 协议是一种工业现场总线协议标准。Modbus-TCP 是基于以太网，因此 xMate 机器人与外界进行通信时需要通过网线连接，在 Modbus-TCP 网络中，xMate 机器人作为服务器存在。

4.3.3.1 连接 Modbus

说明

xMate 机器人系统只支持有线形式接入到以太网中。

xMate 机器人底座配置了两个网络接口可将机器人连接到外部网络，分别为"J1"与"J2"网络接口。机器人端配置了一个固定 IP 地址（192.168.0.160），可通过"J2"网络接口与外设直连方式连接；也可以通过"J1"网络接口接入到 DHCP 功能的路由中，用于自动分配 IP 地址给机器人。

4.3.3.2 开启 Modbus

说明

从站模式，打开 HMI，连接到机器人，并切换到 Admin 以上的用户权限，进入机器人设置界面，打开通讯>Modbus 设置界面。

选择从站模式并输入对应 IP 和端口号，随后点击打开按钮，启动 Modbus 功能

IP 设置成 0.0.0.0 时，机器人根据外设连接的网口自动分配 IP 地址，用户也可根据机器人与外设连接的端口输入对应的 IP 地址，例如：

- 机器人通过“J2”网络接口与外设直连方式连接时，输入“192.168.0.160”，端口号默认“502”；
- 机器人通过“J1”网络接口与外设直连方式连接时，输入无线路由自动分配的 IP，端口号默认“502”；

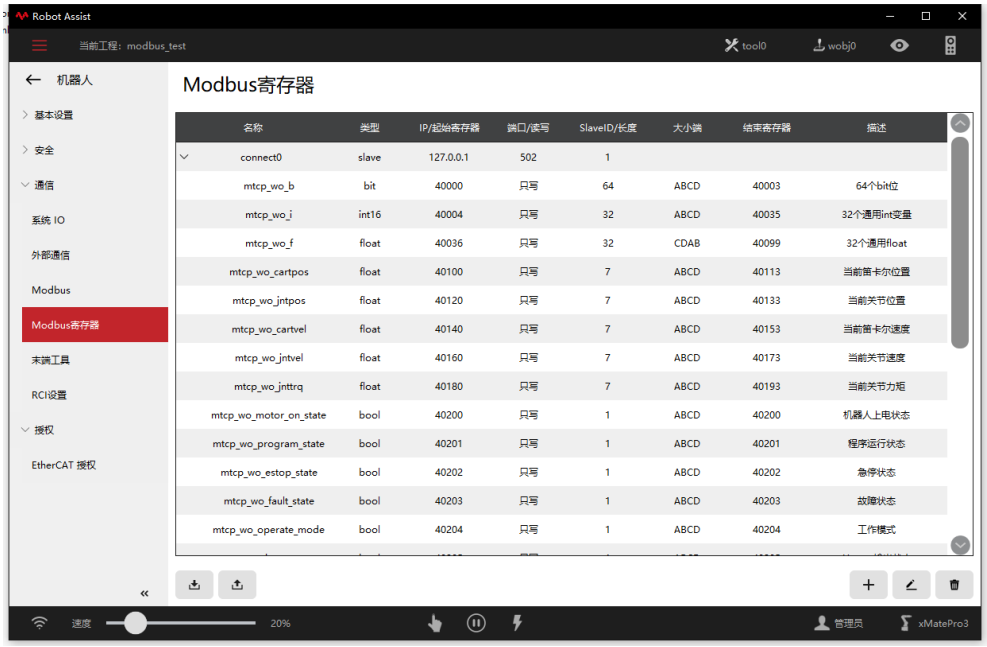
4.3.3.3 配置 Modbus

Modbus 属性

默认配置文件中可以看到每个 Modbus 寄存器有以下几个属性：

名称	类型	首地址	读/写	长度	大小端	结束地址	描述

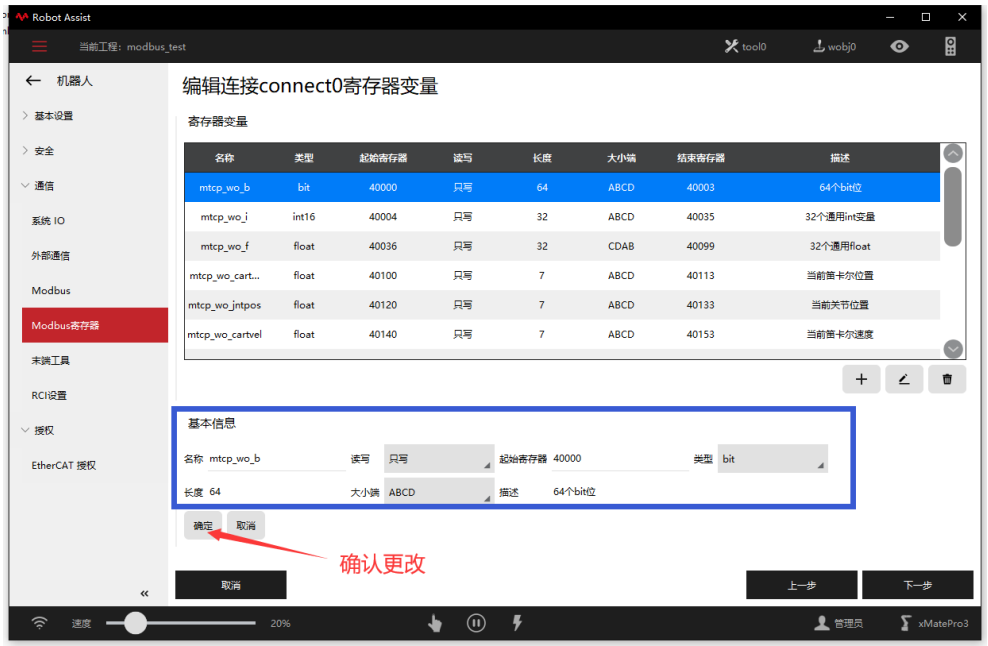
默认配置见下图



用户可以根据自己的需求，手动修改该配置文件，具体修改步骤如下：

1. 点击右下角的编辑按钮，进入编辑界面
2. 修改寄存器参数操作步骤见下图

3. 更改完成后，点击“确定”确认更改。



> **提示**
>
> 寄存器之间的名称以及地址不能冲突，即不同寄存器之间不能重名，不同寄存器的起始寄存器和结束寄存器之间不能交叉。

4.3.3.4 获取机器人状态

说明

参考默认配置文件，xCore 程序会将机器人的实时状态输出到 Modbus 设备中，外部设备可以通过 Modbus 获取机器人的状态信息。

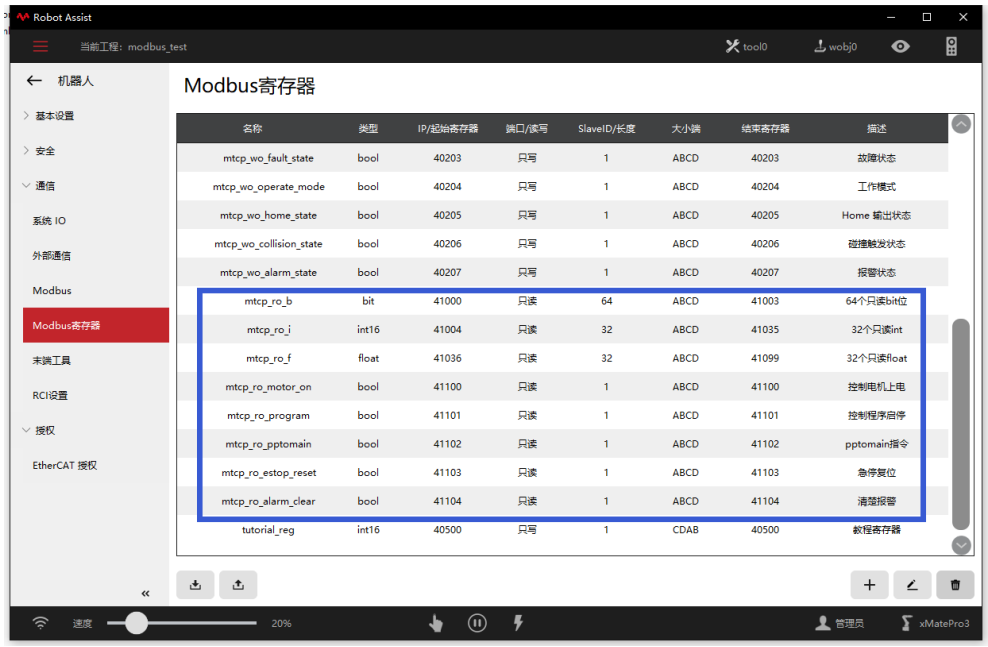
Modbus寄存器

名称	类型	IP/起始寄存器	端口/读写	SlaveID/长度	大小端	结束寄存器	描述
mtcp_wo_f	float	40036	只写	32	CDAB	40099	32个通用float
mtcp_wo_cartpos	float	40100	只写	7	ABCD	40113	当前笛卡尔位置
mtcp_wo_jntpos	float	40120	只写	7	ABCD	40133	当前关节位置
mtcp_wo_cartvel	float	40140	只写	7	ABCD	40153	当前笛卡尔速度
mtcp_wo_jntvel	float	40160	只写	7	ABCD	40173	当前关节速度
mtcp_wo_jnttrq	float	40180	只写	7	ABCD	40193	当前关节力矩
mtcp_wo_motor_on_state	bool	40200	只写	1	ABCD	40200	机器人上电状态
mtcp_wo_program_state	bool	40201	只写	1	ABCD	40201	程序运行状态
mtcp_wo_estop_state	bool	40202	只写	1	ABCD	40202	急停状态
mtcp_wo_fault_state	bool	40203	只写	1	ABCD	40203	故障状态
mtcp_wo_operate_mode	bool	40204	只写	1	ABCD	40204	工作模式
mtcp_wo_home_state	bool	40205	只写	1	ABCD	40205	Home 输出状态
mtcp_wo_collision_state	bool	40206	只写	1	ABCD	40206	碰撞触发状态
mtcp_wo_alarm_state	bool	40207	只写	1	ABCD	40207	报警状态

4.3.3.5 控制机器人

说明

见配置文件，可以通过 Modbus 的寄存器控制机器人的状态，例如上下电、程序的运行。



名称	类型	IP/起始寄存器	端口/读写	SlaveID/长度	大小端	结束寄存器	描述
mtcp_wo_fault_state	bool	40203	只写	1	ABCD	40203	故障状态
mtcp_wo_operate_mode	bool	40204	只写	1	ABCD	40204	工作模式
mtcp_wo_home_state	bool	40205	只写	1	ABCD	40205	Home 输出状态
mtcp_wo_collision_state	bool	40206	只写	1	ABCD	40206	碰撞触发状态
mtcp_wo_alarm_state	bool	40207	只写	1	ABCD	40207	报警状态
mtcp_ro_b	bit	41000	只读	64	ABCD	41003	64个只读bit位
mtcp_ro_i	int16	41004	只读	32	ABCD	41035	32个只读int
mtcp_ro_f	float	41036	只读	32	ABCD	41099	32个只读float
mtcp_ro_motor_on	bool	41100	只读	1	ABCD	41100	控制电机上电
mtcp_ro_program	bool	41101	只读	1	ABCD	41101	控制程序启停
mtcp_ro_pptomain	bool	41102	只读	1	ABCD	41102	pptomain指令
mtcp_ro_estop_reset	bool	41103	只读	1	ABCD	41103	急停复位
mtcp_ro_alarm_clear	bool	41104	只读	1	ABCD	41104	清除报警
tutorial_reg	int16	40500	只写	1	CDAB	40500	教程寄存器

4.3.3.6 Modbus 指令

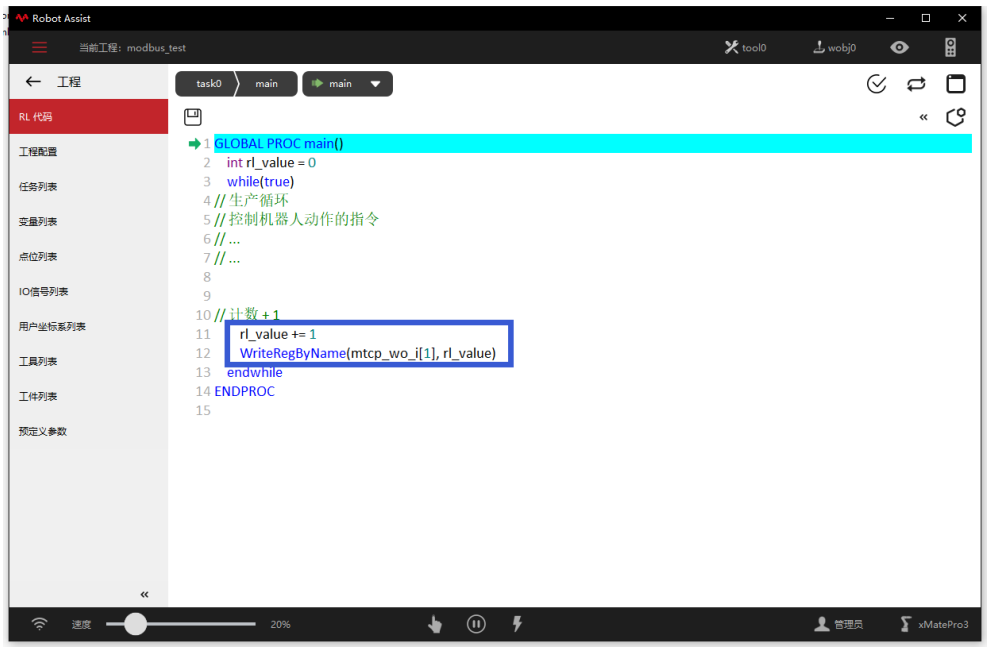
解释器使用 WriteRegByName 和 ReadRegByName 两条指令读取、修改 Modbus 的数据值，指令格式如下：

- WriteRegByName(modbus_reg[index], rl_symbol)

第一个参数是 Modbus 寄存器在配置文件中的名字，可以使用[index]在对应寄存器的首地址再

进行偏移，限制 1 <= index <= Modbus.len，默认 index = 1

解释器中的数据可以输出到 Modbus 设备中（比如 RL 语言的循环次数），假设它在解释器内部定义为“int rl_value”，如果要把它输出到 Modbus，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，则只需要在解释器中添加一条 WriteRegByName 指令即可，就可以通过 Modbus 获得解释器内部的信息。



- ReadRegByName(modbus_reg[index], rl_symbol)

和 Write 指令十分类似，不过其功能是将 Modbus 寄存器内部的变量读取到解释器内部，可以用来控制 RL 程序的执行流程、运动参数等等。

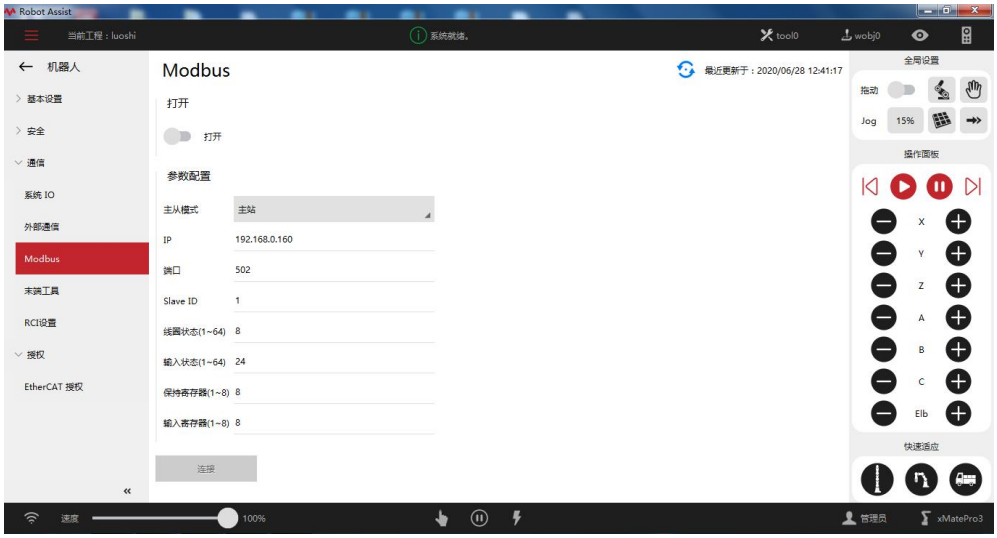
- RL 程序中也支持 Modbus 变量直接赋值的方式，例如“mtcp_wo_i[1] = 1“

4.3.4 Modbus IO 模块配置

说明

xMate 控制器只支持以主站方式接入到 Modbus 网络中，暂不支持 SLAVER 功能。IO 模块需支持 Modbus TCP 功能，建议选择珞石推荐的 Modbus IO 模块。

参数配置



参数	值/说明
启用 Modbus 模块	功能未打开时，连接界面置灰；打开时，可进行参数配置。
模块状态	显示模块连接状态：断开、连接。
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	从站 ID
线圈状态（1~64）	Modbus 模块“线圈状态”寄存器。范围：1~64，数据类型：int。
输入状态（1~64）	Modbus 模块“输入状态”寄存器。范围：1~64，数据类型：int。
连接	参数配置完成后，点击“连接”，完成参数配置。

4.3.5 末端工具通信

说明

xMate 控制器支持对大寰爪手的开合控制，爪手支持 IO 通讯和 RS485 通讯。

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

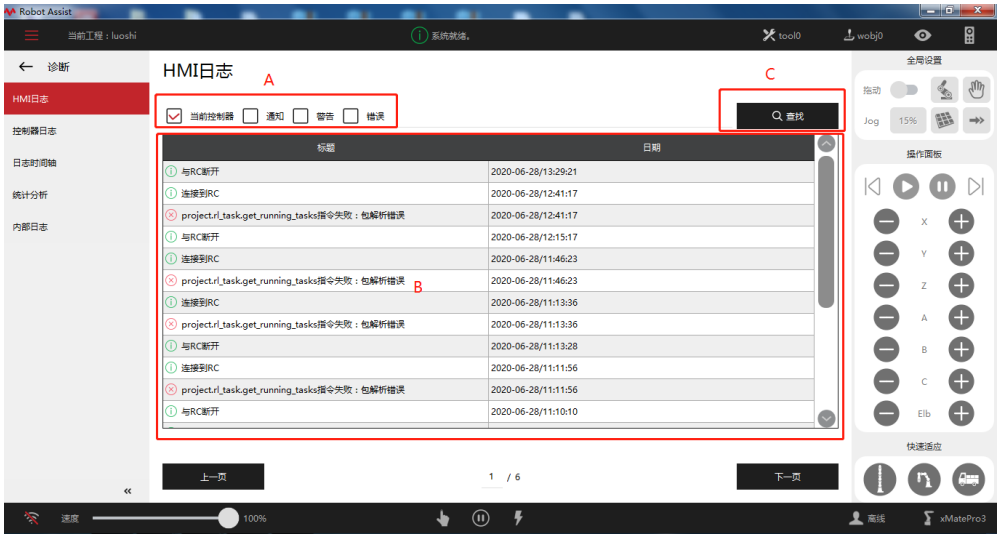
RS485 支持对夹爪 trip 参数设置，IO 控制时，其 trip 参数只能通过大寰通讯转接盒设置。

5 诊断

说明

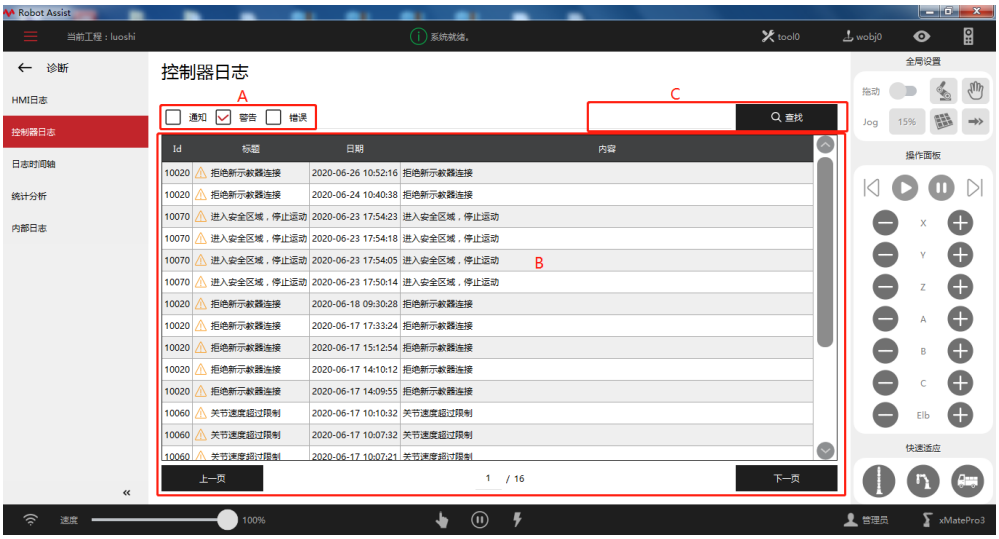
xCore 系统提供了详细的运行日志，可以用来追溯机器人的运行状况，查找故障原因。使用日志管理界面对产生的日志进行筛选、查看等操作。

5.1 HMI 日志



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

5.2 控制器日志



操作说明

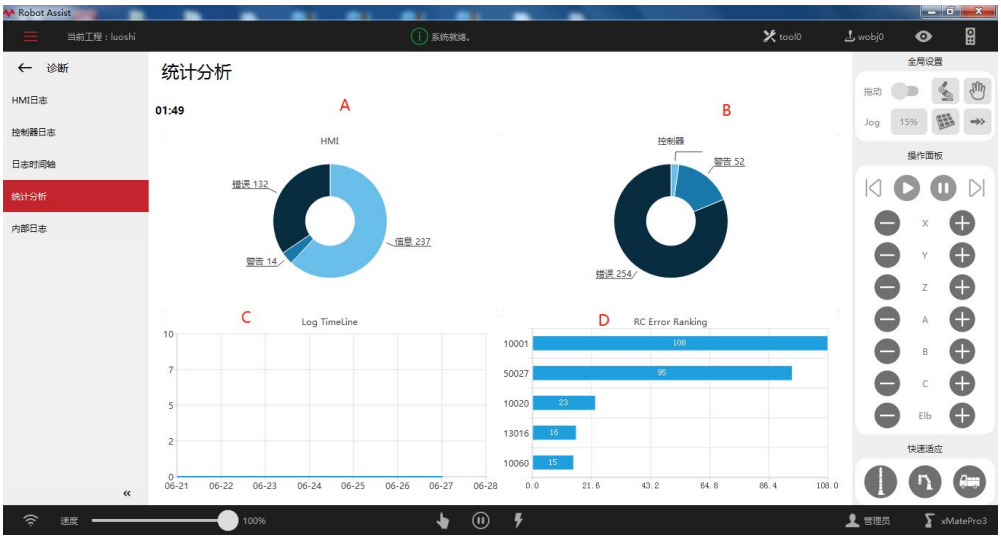
A	筛选条件区，可选择日志级别进行筛选
B	日志显示页，显示日志编号、标题、发生时间以及内容等基本信息。使用翻页按钮切换上下页
C	查找区，可根据关键字对日志进行搜索

5.3 日志时间轴

操作说明

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

5.4 统计分析



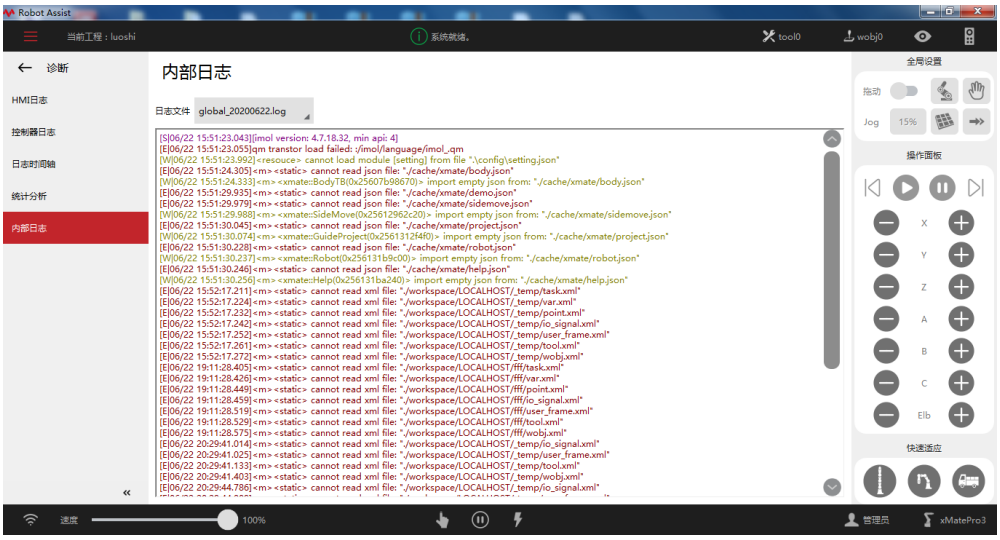
操作说明

A	显示 HMI 各级别日志比例
B	显示控制器各级别日志比例
C	显示控制器每天打印的日志个数趋势图
D	显示引起控制器崩溃的错误排行榜，纵坐标为错误编号，横坐标为错误产生数量

5.5 内部日志

说明

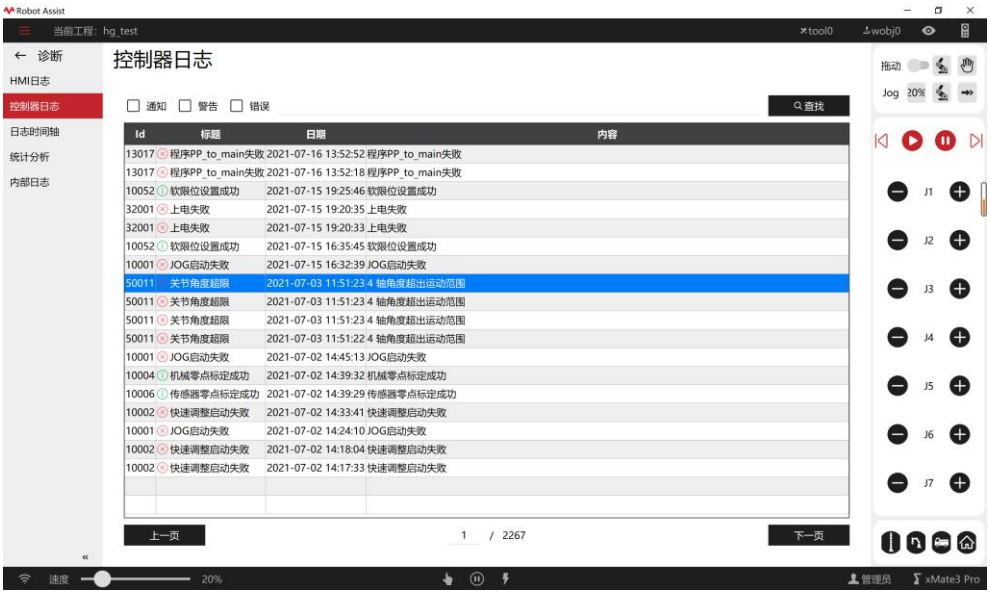
客户现场遇到问题时，技术支持可根据日志文件判断出问题原因。



5.6 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。



5.7 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。

操作说明

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。

6 演示

6.1 七轴冗余运动

说明

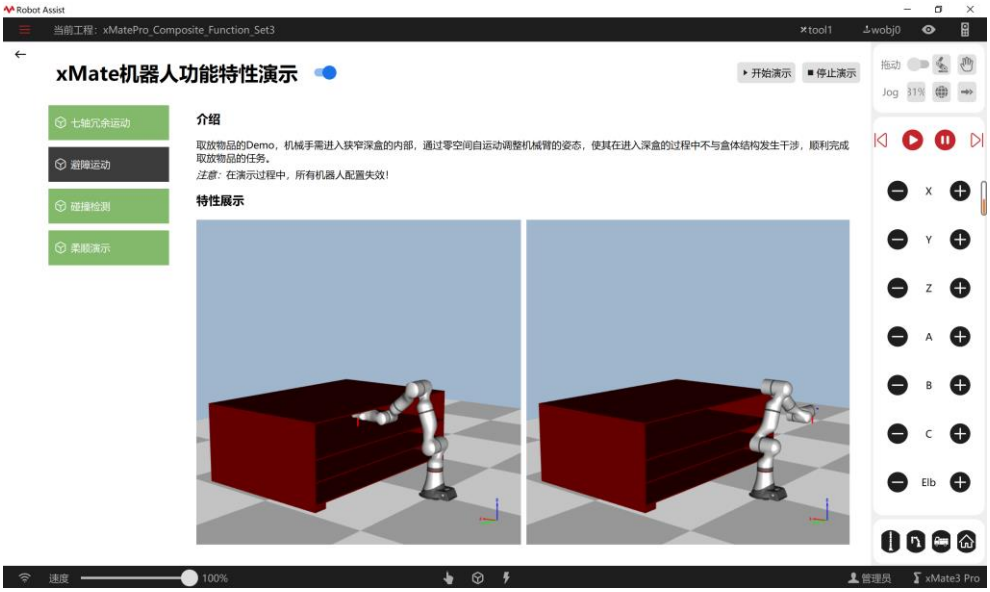
七轴冗余机械手的一般运动示例，包括圆弧、直线、转弯曲、零空间自运动等。

	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。

6.2 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。



6.3 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。

6.4 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。

7 选项

连接

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 HMI 间的网络断开后，HMI 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 HMI 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。

基本设置

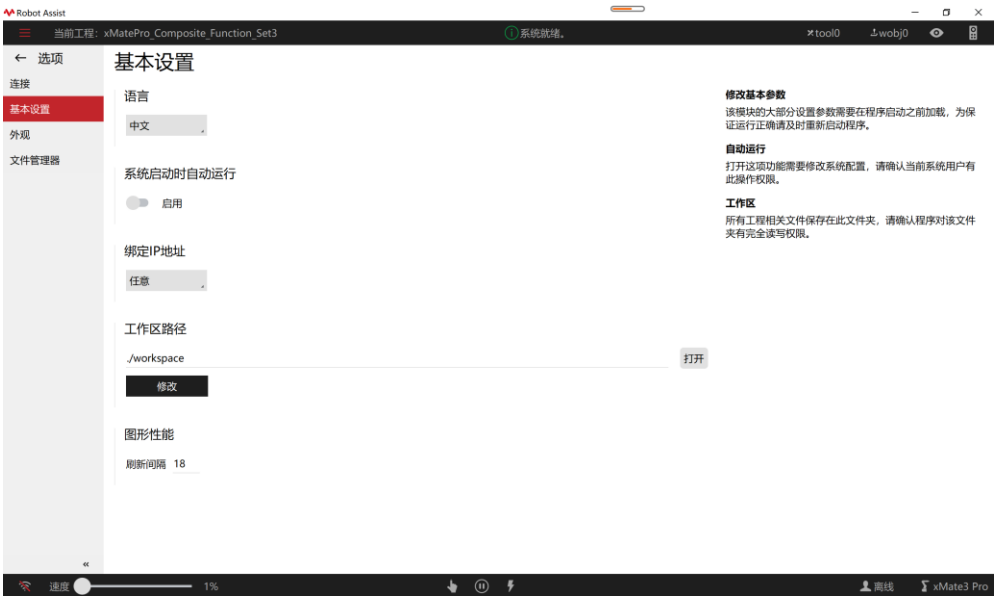
语言设置：支持中文和英文。

系统启动时自动运行：勾选后 HMI 软件在开机时会自动运行

绑定 IP 地址：设置 HMI 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。



外观

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 HMI 界面上全部文字的大小，点击应用后立即生效。

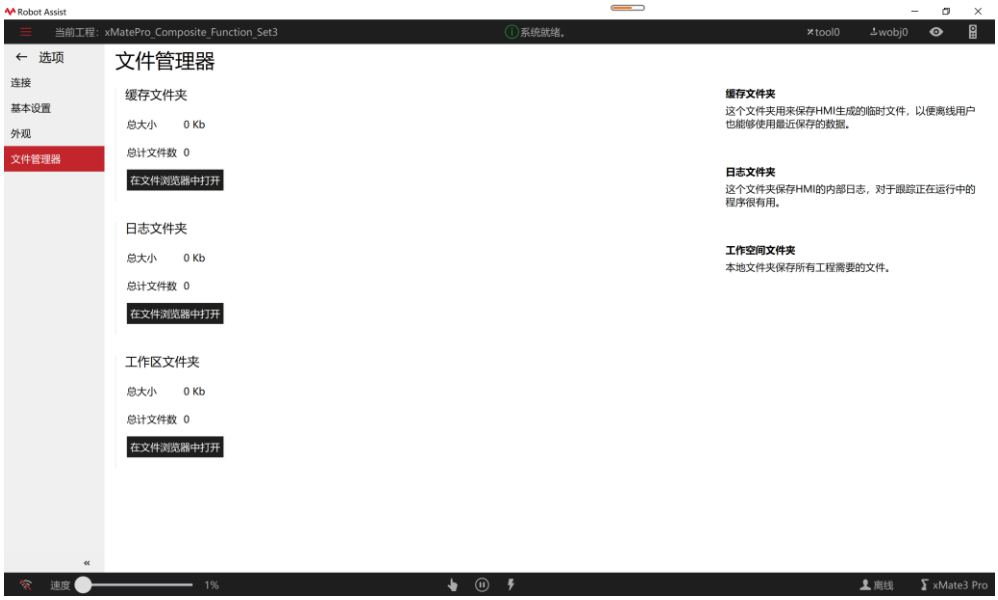
文件管理器

文件管理器界面主要用于快捷打开 HMI 软件包中的文件夹。

缓存文件夹：用于存储 HMI 软件的缓存。

日志文件夹：用于打开存储 HMI 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



8 帮助

说明

帮助界面主要用于软件升级备份和恢复等操作。

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。

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。

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。

9 RCI

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》。

10 工程

10.1 工程介绍

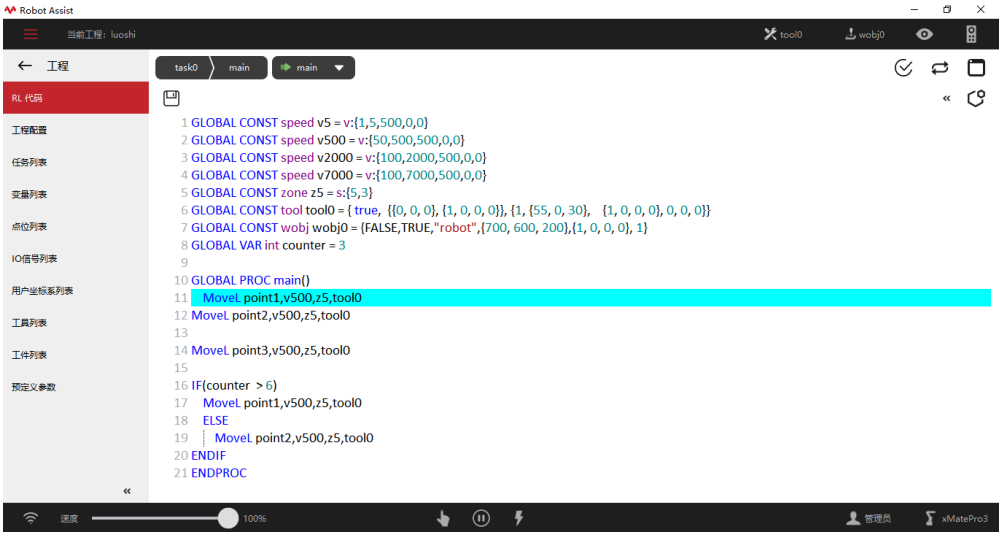
工程概述

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；
- RL 程序；
- 示教点位列表；
- 变量列表；
- IO 信号列表；
- 用户坐标系列表；
- 工具列表；
- 工件列表；
- 预定义参数。

打开工程

点击左上角的≡按钮打开扩展菜单栏，选择底下的工程进入到工程界面；或者直接点击 当前工程：luoshi 可以快捷进入工程界面



10.2 工程配置

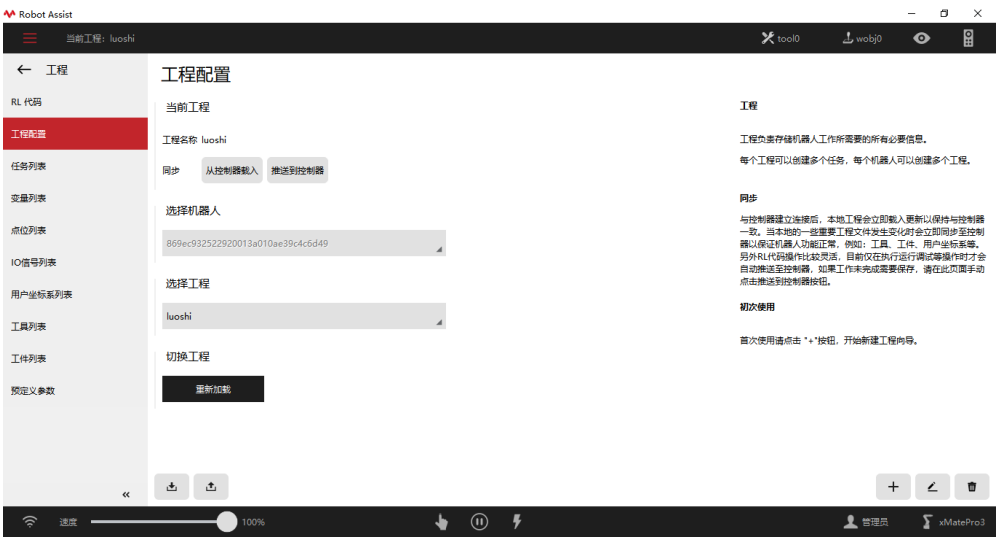
说明

工程配置界面用于对当前工程进行相关配置，包括：

- HMI 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

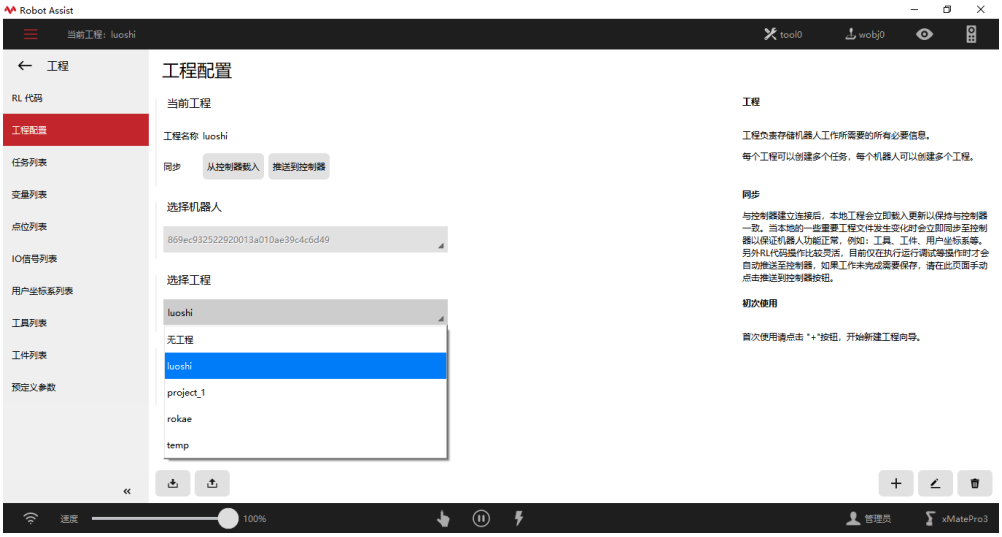
HMI 与控制器工程同步

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

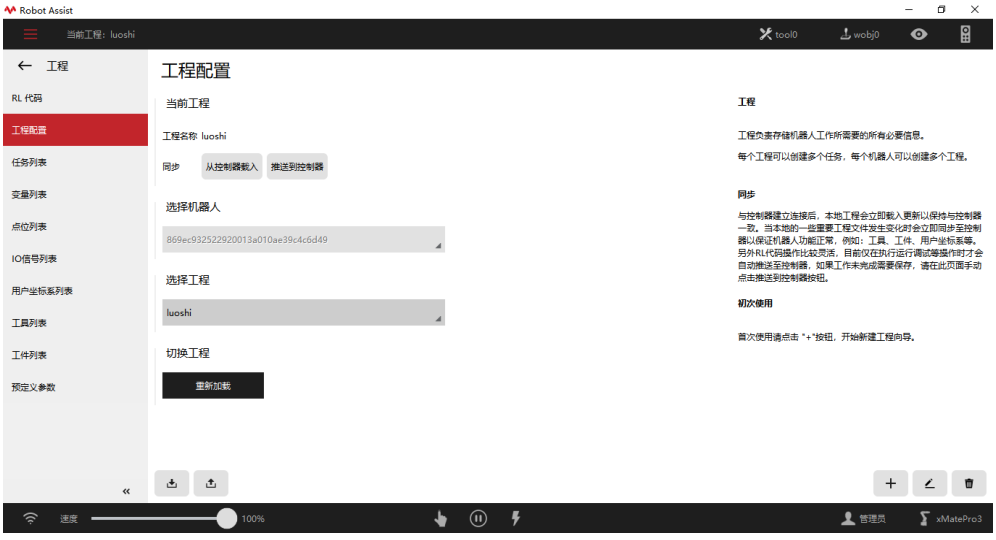
点击选择工程下方的下拉菜单，显示所有工程，选择想要切换的工程，点击下方的重新加载对工程进行切换。



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！

10.3 任务列表

10.3.1 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。

名词解释

Project，在 xCore 中我们称之为工程。

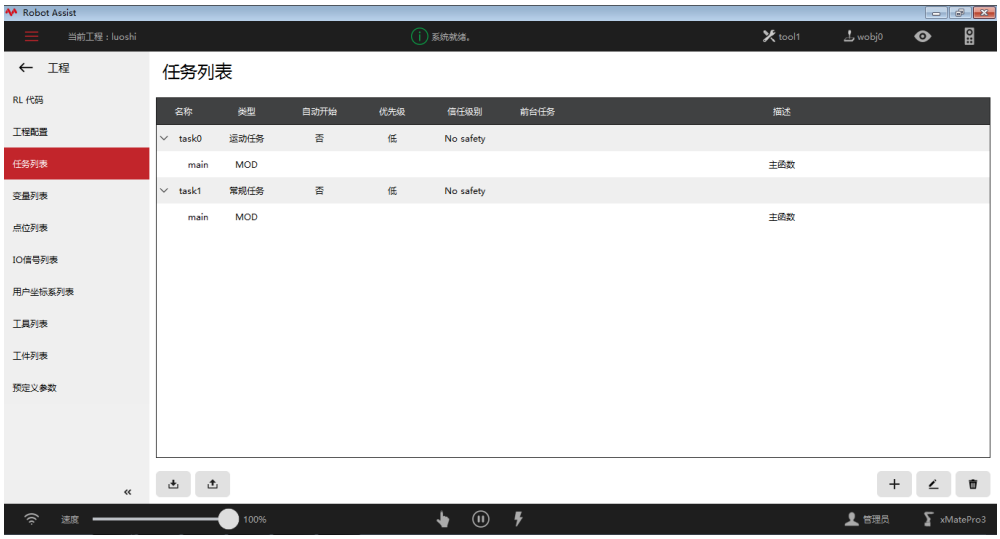
Task，在 xCore 中我们称为任务。

Module，在 xCore 中我们称为程序文件。

10.3.2 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。



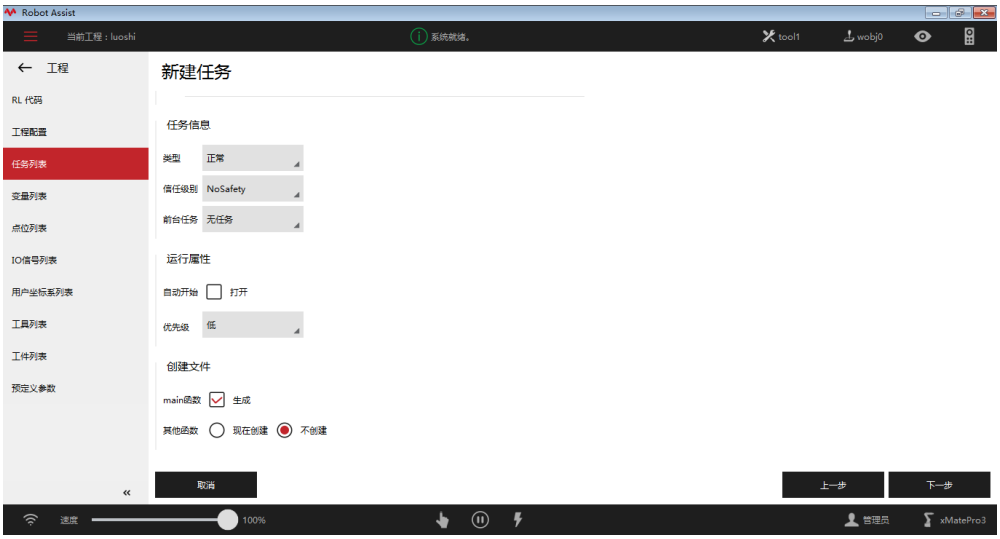
10.3.3 新建任务

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。



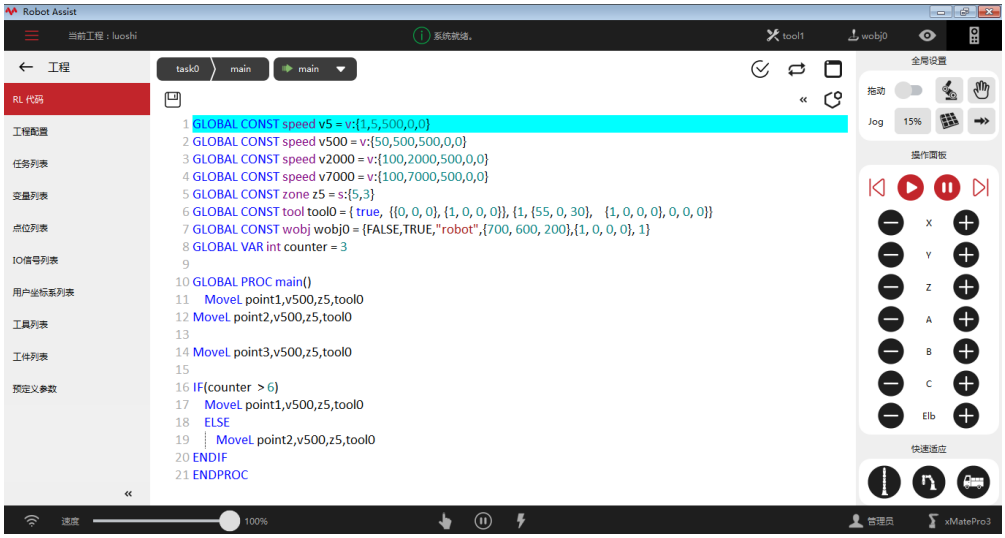
使用限制

- 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。

10.3.4 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。

10.3.5 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。

10.4 RL 程序

10.4.1 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。

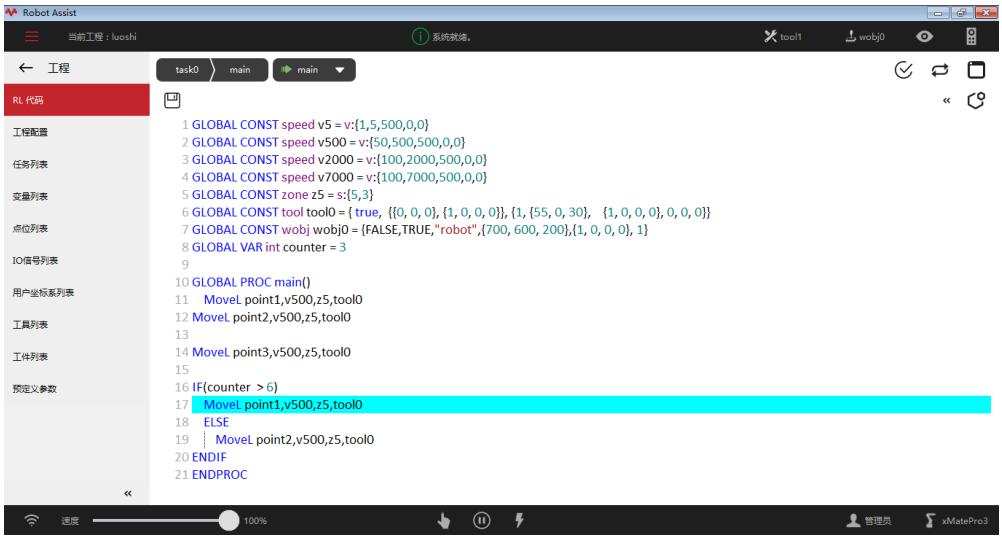
RL 语言为 ROKAE Robot Language 的缩写，即 ROKAE 机器人编程语言，用户可以通过示教器编写控制机器人的程序。

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；

10.4.2 程序结构

10.4.2.1 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；

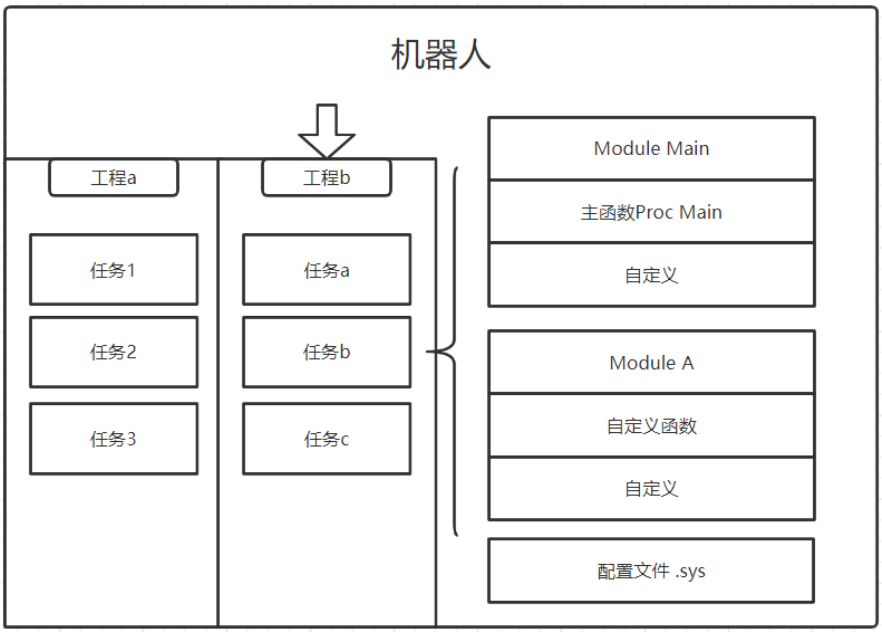
2. 一个工程中可以包含多个任务，原则上每个任务互相独立，仅靠提供的接口进行交互；

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：



10.4.2.2 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。

10.4.2.3 函数

说明

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

目前不支持带返回值的函数。

函数定义

函数的定义方式为：

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

其中，

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

函数调用

调用函数时，直接在程序编辑器输入函数名即可：

RoutineName()

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

对于函数的调用在编译器中被视为一条独立的程序指令。

注意事项

1. 不允许在函数中定义函数。

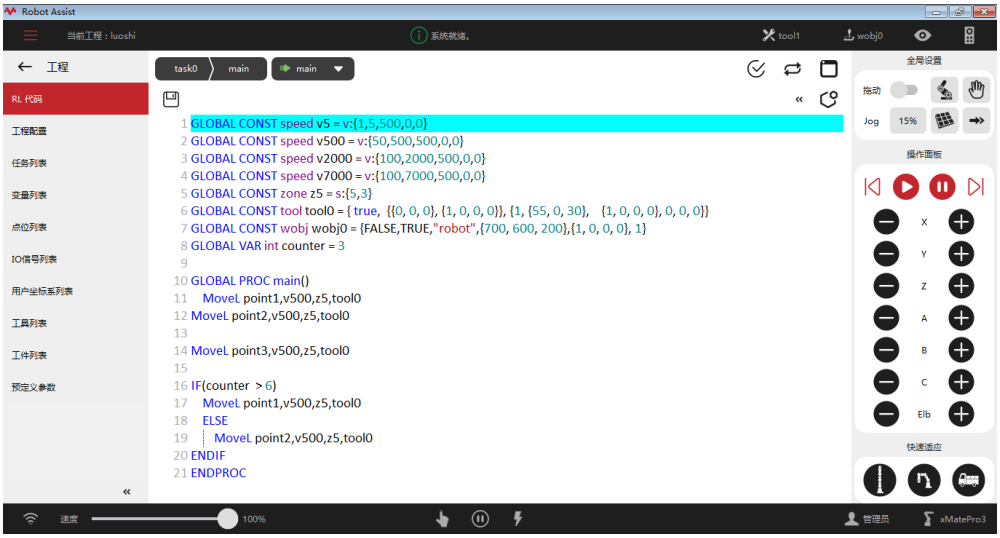
10.4.3 程序编辑

10.4.3.1 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。

10.4.4 程序调试

10.4.4.1 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）。

10.4.4.2 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。

10.4.4.3 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。

10.4.4.4 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。

10.4.4.5 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。

10.4.4.6 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。

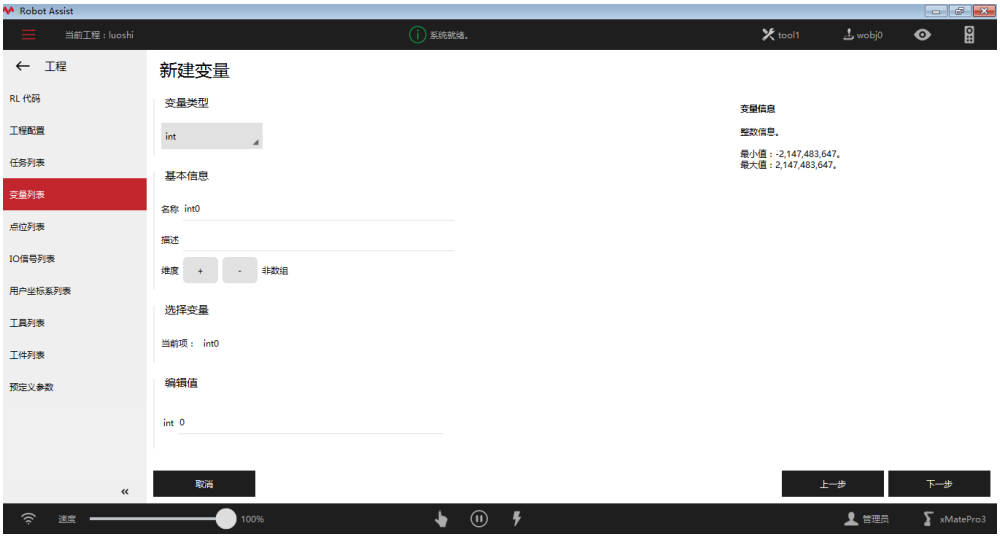
10.4.4.7 变量管理

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：



10.5 点位列表

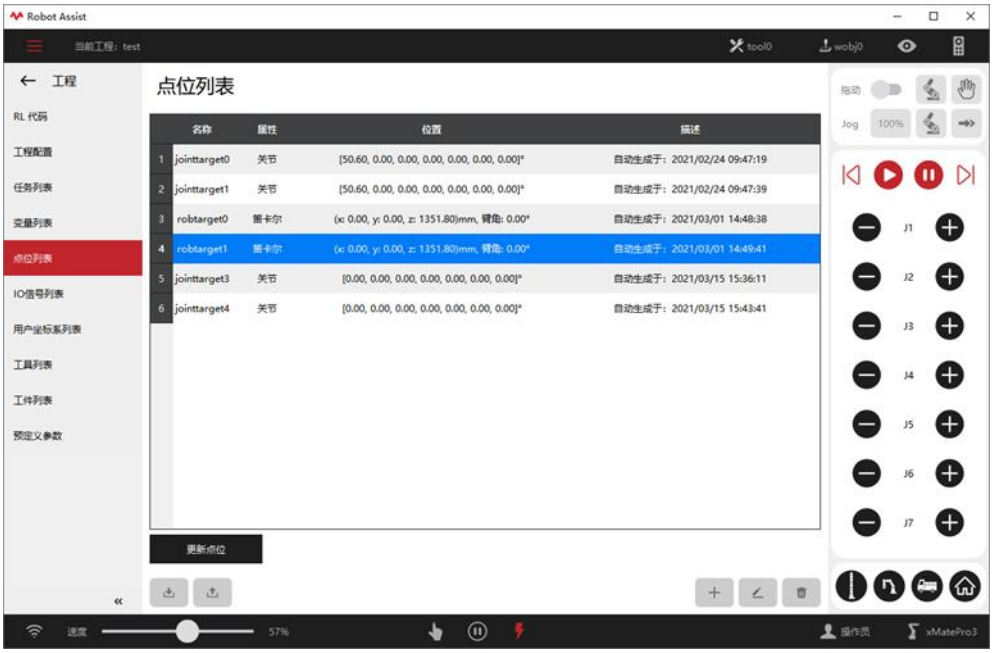
说明

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。

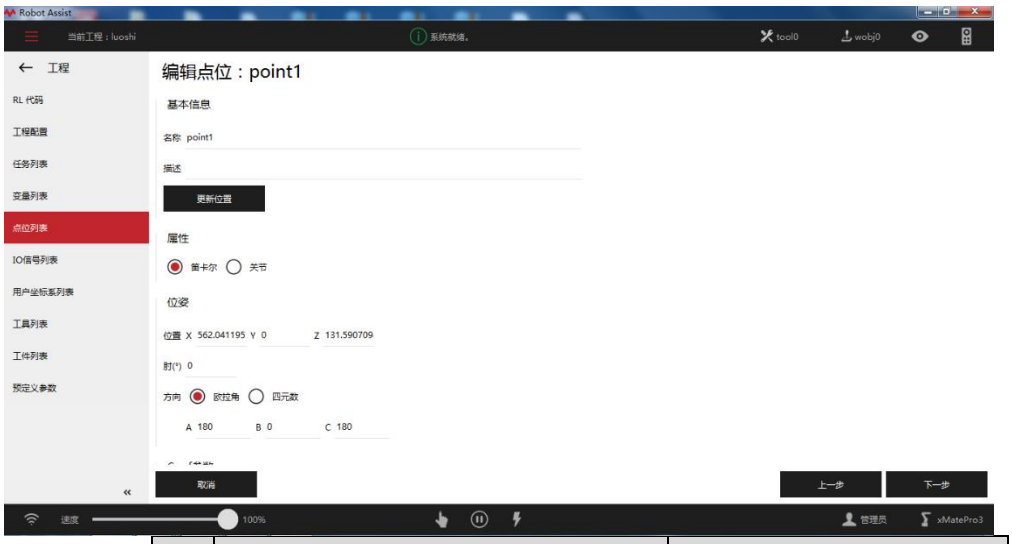
添加、修改、删除 点位

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	

10.6 IO 列表

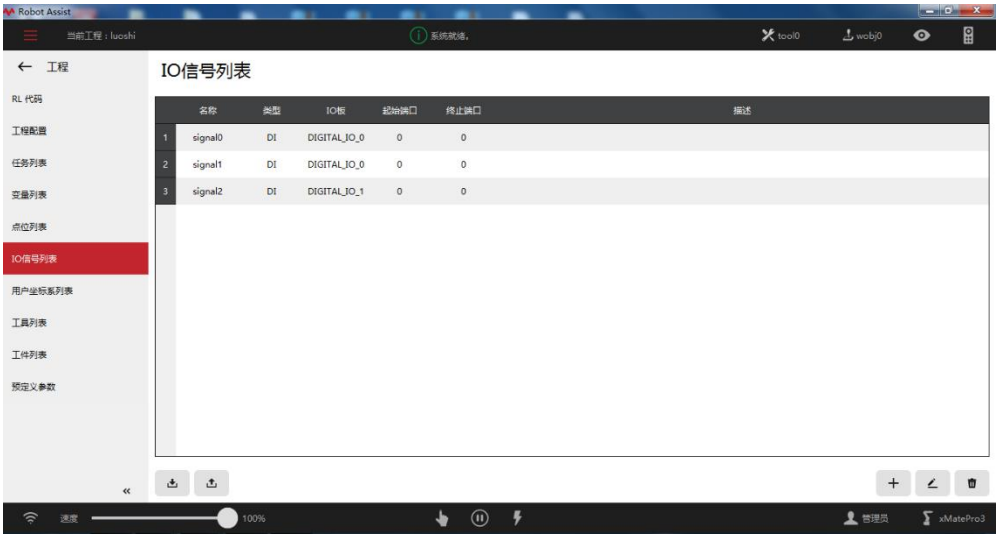
说明

xCore 系统中，除系统 IO 设置 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。

Example 1

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

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

使用限制

用户自定义 IO 不可映射到系统输出上。

10.7 工具

10.7.1 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：

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

修改工具定义只能通过 HMI 的坐标系标定界面来进行，详细步骤请参考标定工具坐标系。对于 tool 类型的变量，在变量管理界面只能查看，不能新建或者修改。

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

提示

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

10.7.2 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。

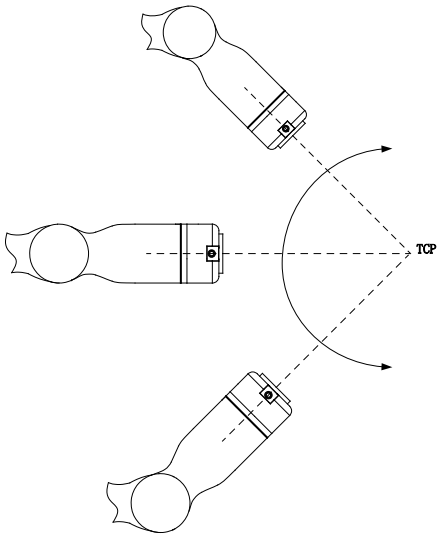
不同的工具有不同的 TCP，根据实际情况定义合适的 TCP 可以大幅提升编程效率。

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



10.7.3 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。

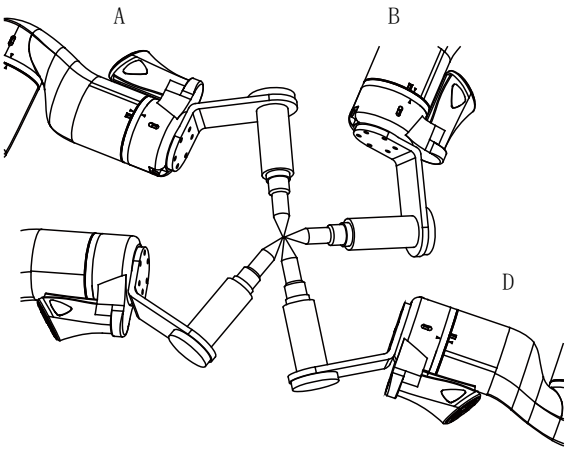
如果您使用的工具生产商提供了这些偏移量数据，那么可以在示教器上选择"手动输入"方式直接输入而不必执行标定过程。

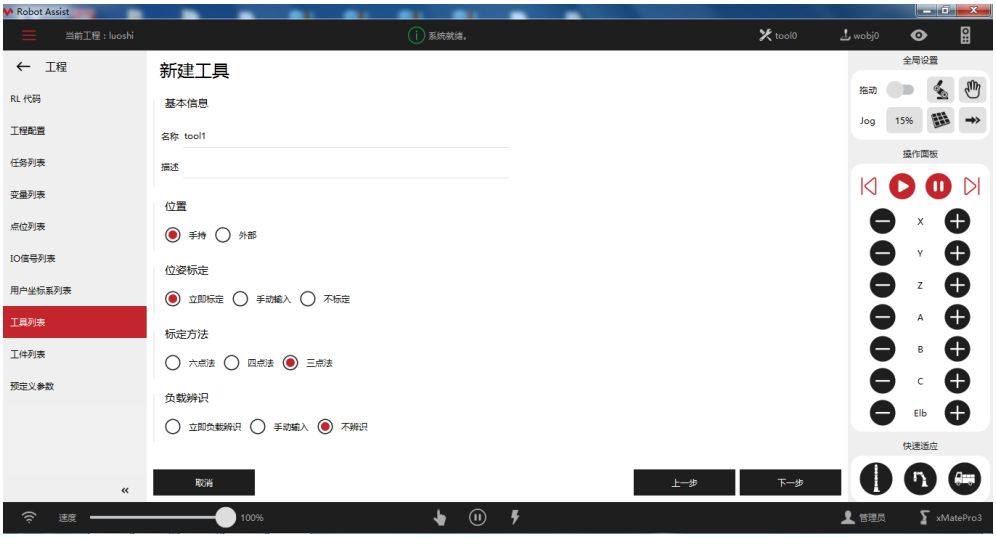
对于没有尺寸数据的工具，则需要使用 xCore 提供三种的方法来进行工具坐标系标定：

- 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。

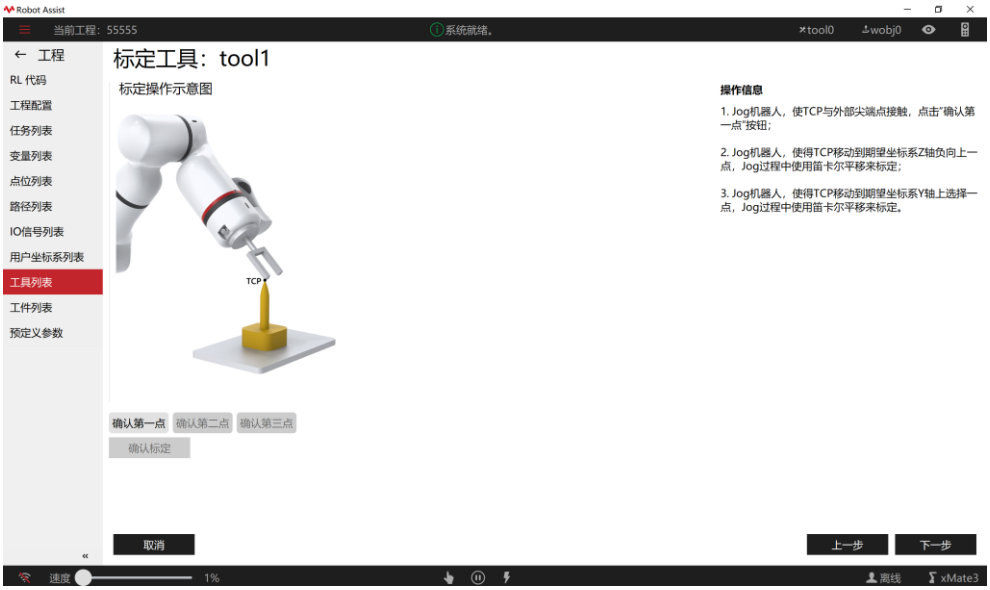
标定工具坐标系位姿

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。

10.7.4 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：

1. 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；
2. 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。

在示教器的 HMI 界面上提供了非常详尽的操作步骤，请根据提示进行操作。

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。

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。

10.7.5 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。

10.7.6 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：

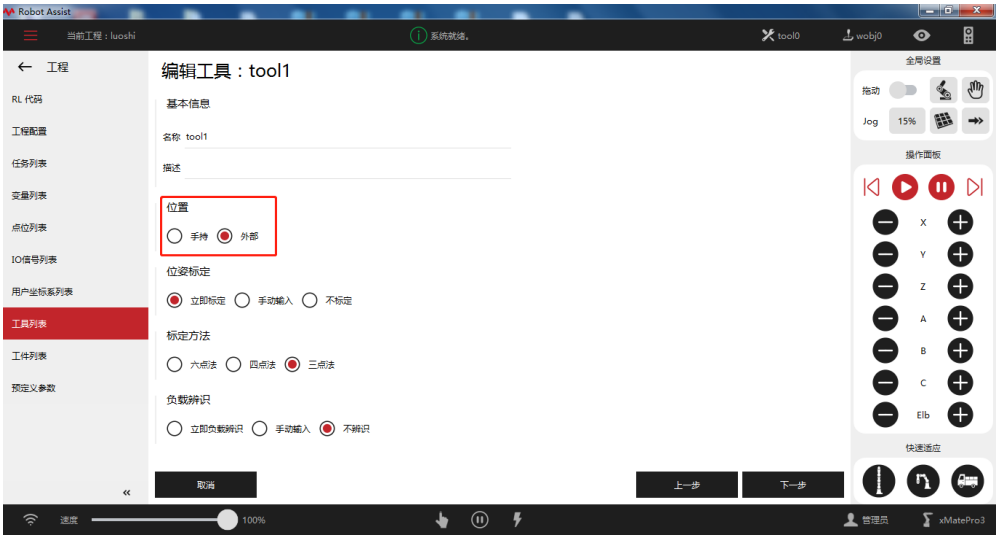
1. 要使用的工具尺寸比较大或者比较重，安装到机器人上不方便或者可能会影响到机器人的运动性能；
2. 待加工的工件尺寸比较大，正常情况下机器人工作范围无法很好的覆盖；
3. 需要完成一些特定的工艺动作，例如打磨一个方形物体时，需要分别绕 4 个角做自旋动作。

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。

10.8 工件列表

10.8.1 什么是工件？

说明

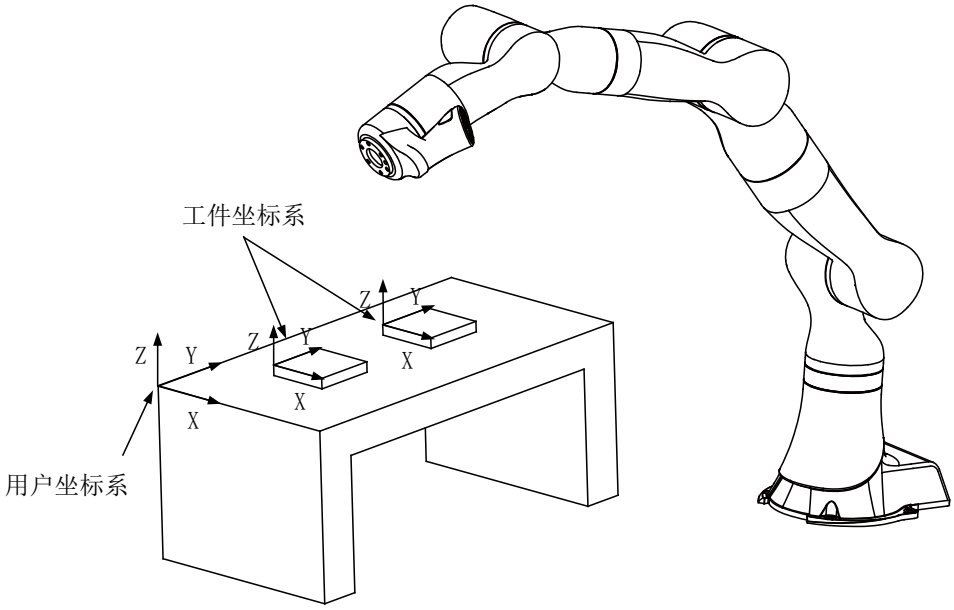
工件是指机器人使用工具进行加工或者处理的物品，在 xCore 系统中使用 wobj（Work Object）类型的变量来描述一个实际的工件。

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。



10.8.2 定义工件

说明

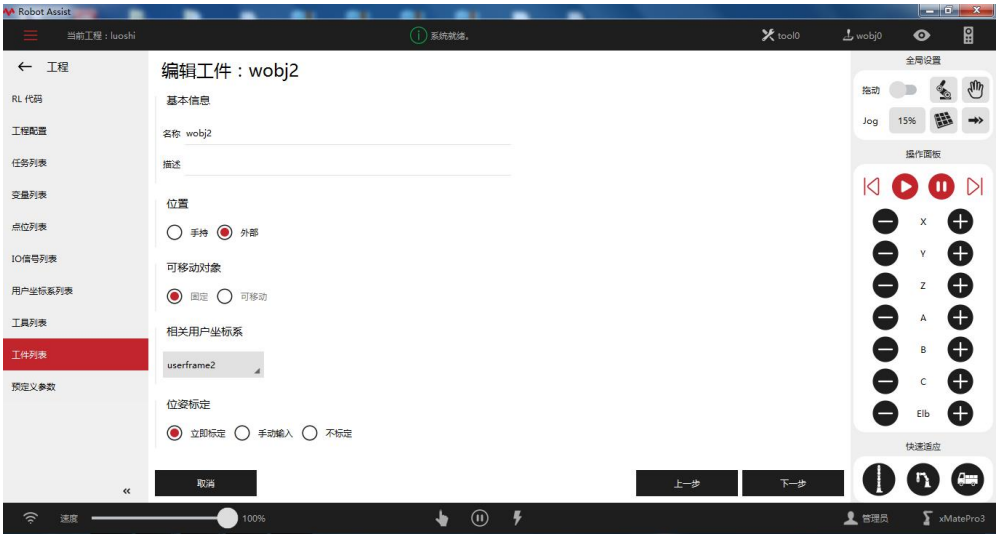
使用工件前首先需要定义新工件。在 xCore 控制系统中，工件通过 wobj 数据类型进行存储和使用。要定义一个工件，就意味着要创建一个 wobj 型的变量。

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

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。

10.8.3 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  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。

10.9 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；

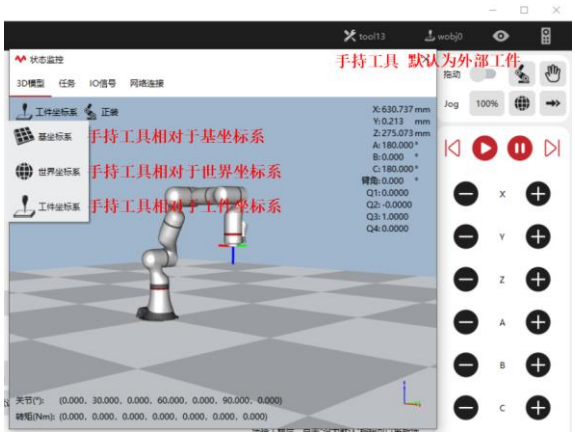
2）工件坐标系同理，当选择工件坐标系如 wobj1 为外部时，选择默认工具坐标系 tool0 为手持，且与法兰坐标系重合；选择工件坐标系如 wobj1 为手持时，选择默认工具坐标系 tool0 为外部，且与用户坐标系重合；

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。

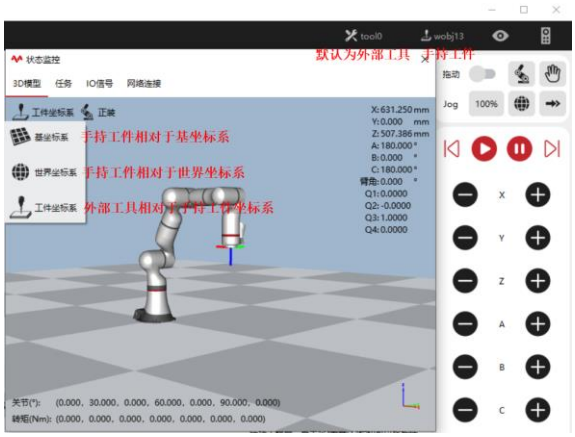
3D 界面显示

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。

10.10 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。

10.11 变量列表

10.11.1 变量

10.11.1.1 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；

4. 在不同的模块中，不允许任何 GLOBAL 级别的函数与变量发生命名冲突；

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

变量作用域

RL 语言系统定义了三种作用域：

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

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

存储类型

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

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

> 提示
>
> 即使某个 PERS 类型变量的值在程序运行过程中被更改，其在程序编辑器声明区内显示的初始值也不会立即刷新，只有当程序重新加载或程序停止时在程序编辑器声明区显示的初始值才会更新到最新值 。
> 不论程序运行与否，都可以在"变量管理"界面随时查看 PERS 变量的最新值。

预定义关键字

以下为 RL 语言预定义的保留关键字（不区分大小写）：

Module、EndModule、Proc、EndProc、Func、EndFunc、TRAP、ENDTRAP、SetDO、DO_ALL、SetGO、SetAO、WaitDI、Wait、WaitUntil、WaitWObj、WBID、Q、P、J、V、W、T、S、L、CA、DURA、IGNORELEFT、EJ、1J、FCBV、FCCV、FCOL、FCXYZ、FCCART、PE、PER、TCP、ORI、EXJ、CFG、PDIS、JDIS、MoveAbsJ、MoveJ、MoveL、MoveC、MoveT、LOCAL、TASK、GLOBAL、VAR、CONST、PERS、INV、DOT、CROSS、sin、cos、tan、asin、cot、acos、atan、atan2、sinh、cosh、tanh、ln、log10、pow、exp、sqrt、ceil、floor、abs、rand、GetCurPos、Print、PrintToFile、ClkRead、TestAndSet、IF 、Else、Endif、WHILE、ENDWHILE、for、from、

to、endfor、Break、Continue、Del、Int、Double、Bool、String、BYTE、Robtarget、Speed、Zone、Tool、Wobj、Jointtarget、TriggData、Load、FCBoxVol、FCSphereVol、FCCylinderVol、FCXyzNum、FCCartNum、Pose、CLOCK、INTNUM、SYNCIDENT、TASKS、Call、Return、EXIT、Pause、StopMove、StartMove、StorePath、RestoPath、True、False、Interrupt、When、Offs、CalcJointT、CalcRobT、CRobT、RelTool、SocketCreate、SocketClose、SocketSendByte、SocketSendInt、SocketSendString、SocketReadString、SocketReadBit、SocketReadInt、SocketReadDouble、AccSet、MotionSup、TriggIO、TriggJ、TriggL、TriggC、On、Off、clock、intnum、userframe、pinf、ninf、FCFRAME_WORLD、FCFRAME_TOOL、FCFRAME_WOBJ、FCFRAME_PATH、FCPLANE_XY、FCPLANE_XZ、FCPLANE_YZ、FC_LINE_X、FC_LINE_Y、FC_LINE_Z、FC_ROT_X、FC_ROT_Y、FC_ROT_Z、Offs、CalcJoinT、CalcRobT、CRobT、RelTool、\\Start、\\Time、ClkReset、ClkStart、ClkStop、CONNECT、WITH、IDisable、IEnable、ISignalDI、\\Single、\\SingleSafe、WaitWobj、DropWobj、WobjIdentifier、WobjAngle、ActUnit、DeactUnit、INTNO、\\Exp、DoubleToStr、WaitSyncTask、FCAct、FCDeact、FCLoadID、FCCalib、FCSupvForce、FCSupvTorque、FCSupvPosBox、FCSupvPosSphere、FCSupvPosCylinder、FCSupvOrient、FCSupvOrient、FCSupvReoriSpeed、FCSupvTCPSpeed、FCCondForce、FCCondTorque、FCCondOrient、FCCondReoriSpeed、FCCondPosBox、FCCondPosCylinder、FCCondPosSphere、FCCondTCPSpeed、FCCondWaitWhile、FCRefLine、FCRefRot、FCRefSpiral、FCRefCircle、FCRefForce、FCRefTorque、FCRefStart、FCRefStop、FCSetSDPara

---

进制转换

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

Example 1

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

8h，代表十六进制的 8，十进制的 8；

bh，代表十六进制的 b，十进制中的 11；

25h，代表十六进制的 25，十进制的 37；

Example 2

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

1b，代表二进制的 1，十进制的 1；

10b，代表二进制的 10，十进制的 2；

1010b，代表二进制的 1010，十进制的 10；

Example 3

在数字后面增加“e±x”，表示该数字乘以 10 的 x 次方，例如：

5e+20，代表 5×10^20 ；

26e-15，代表 26×10^(-15)；

112e-10，代表 112×10^(-10)；

10.11.1.2 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：

Example 1

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

Example 2

一般情况下，变量不允许重名：

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

此时编译器将报错，提示"添加变量失败"。

Example 3

但是全局变量和局部变量变量可以使用相同的变量名：

```
VAR int counter = 1
GLOBAL int counter = 555
```

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

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

使用限制

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

10.11.2 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。

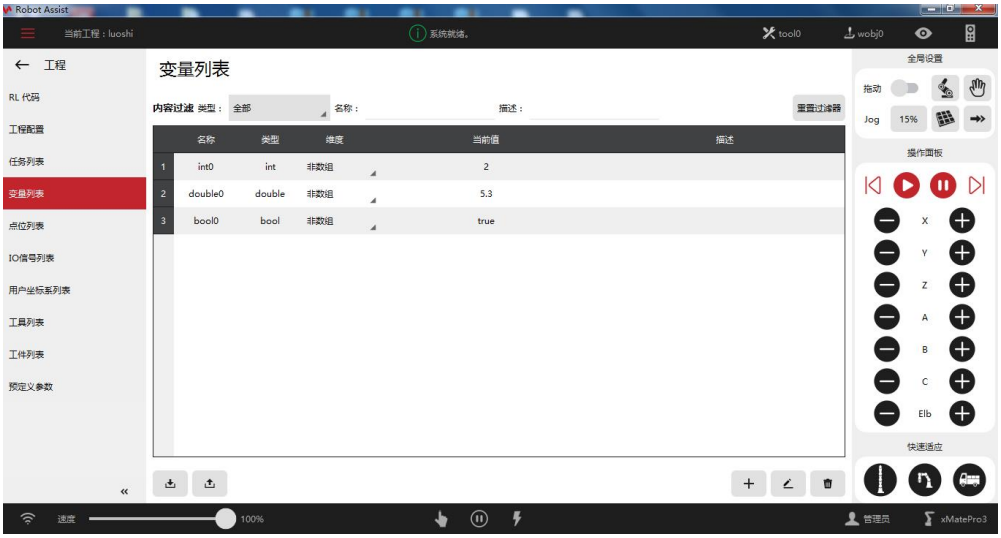
对于一些某些有专门定义步骤的变量类型，例如 tool/wobj（使用标定界面进行定义和修改），robtarget/jointtarget/speed/zone（使用辅助编程界面进行定义和修改），虽然也可在变量

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 机器人运动基础

11.1 坐标系系统

说明

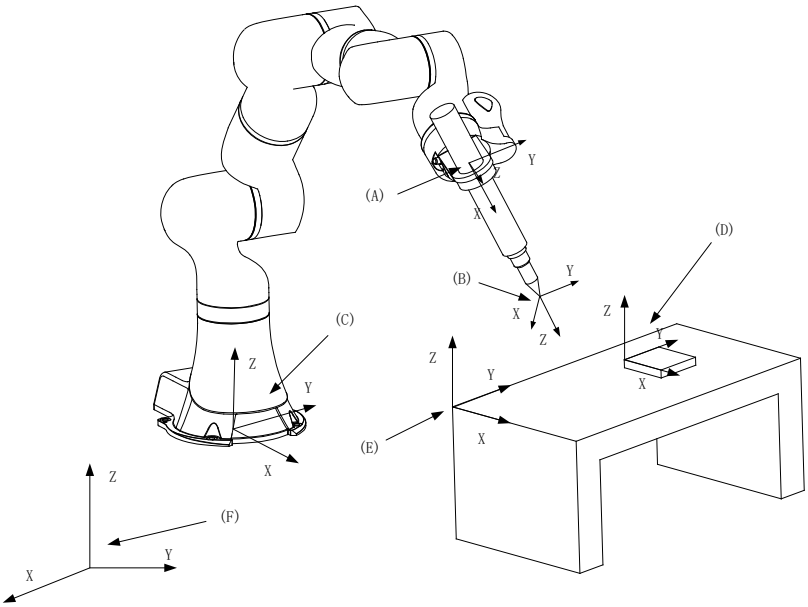
机器人的运动包含位置、速度、加速度等信息，这些信息必须要指定参考系才具有实际意义，因此在开始 Jog 之前我们需要了解机器人所使用的坐标系统。

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。

11.2 机器人奇异

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

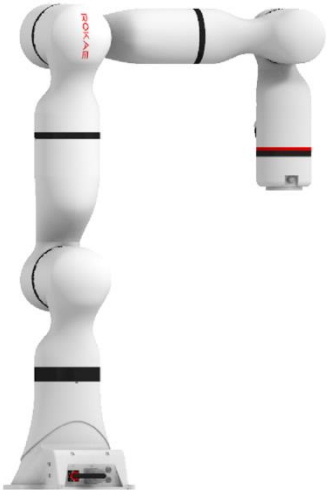
4. 腕心奇异点

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

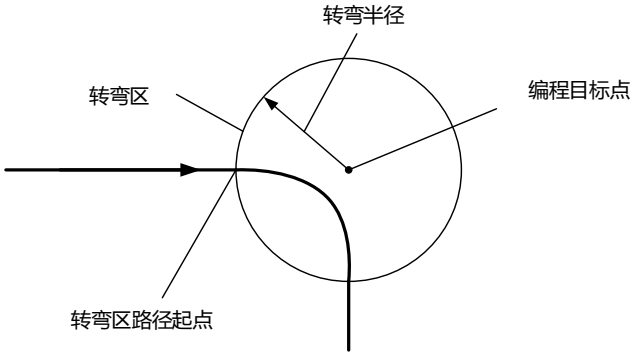
当机器人腕心位于一轴正上方时，机器人处于腕心奇异状态：

此时，机器人求解逆解时，无法准确给定 1 轴角度。

11.2.1 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。

11.2.2 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。

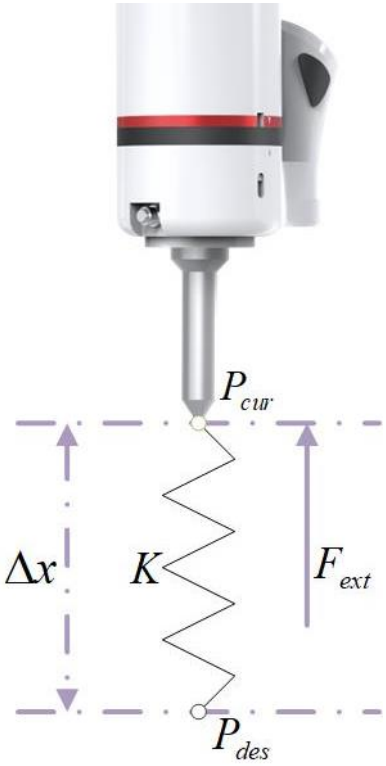
11.3 机器人力控

11.3.1 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。

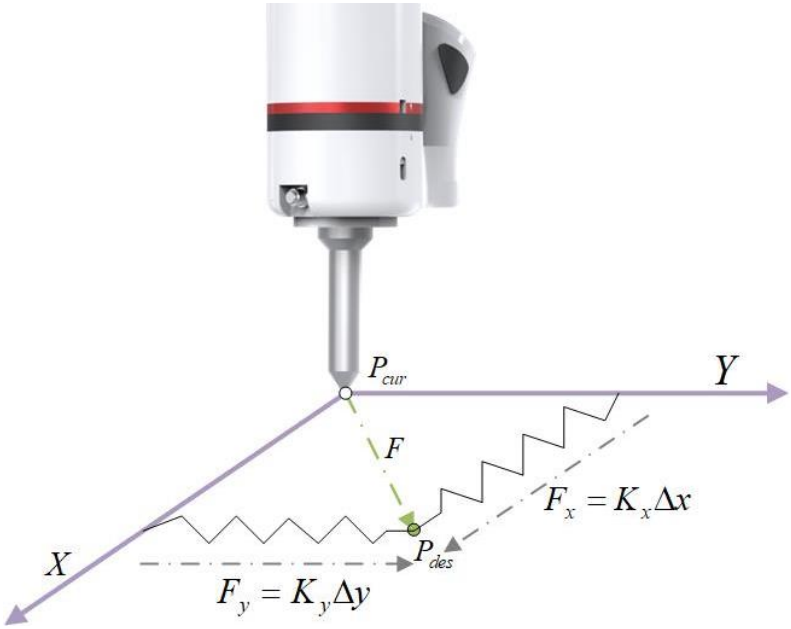
11.3.2 阻抗控制

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 $F = F_x + F_y$。



11.3.3 搜索运动

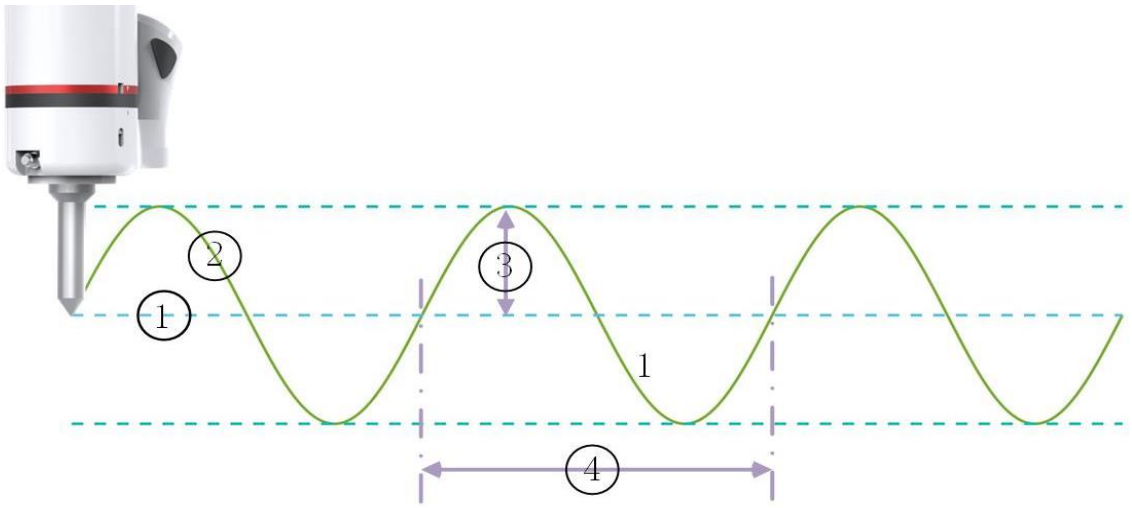
人手在工件装配的过程中，会感受装配过程中力的变化，当感受到阻碍的时候（工件卡住），会尝试通过抖动，来使工件顺利安装。力控制使得 xMate 也具有这样的手法，也即搜索运动，xMate 支持绕轴旋转的正弦搜索运动和平面内的莉萨如搜素运动。搜索运动是在机器人既定运动的基础上叠加的额外运动，搜索运动使得机器人表现出一定的抖动，从而能够在装配过程中更好的克服阻碍。下图是一个正弦搜索运动过程：

1、期望轨迹

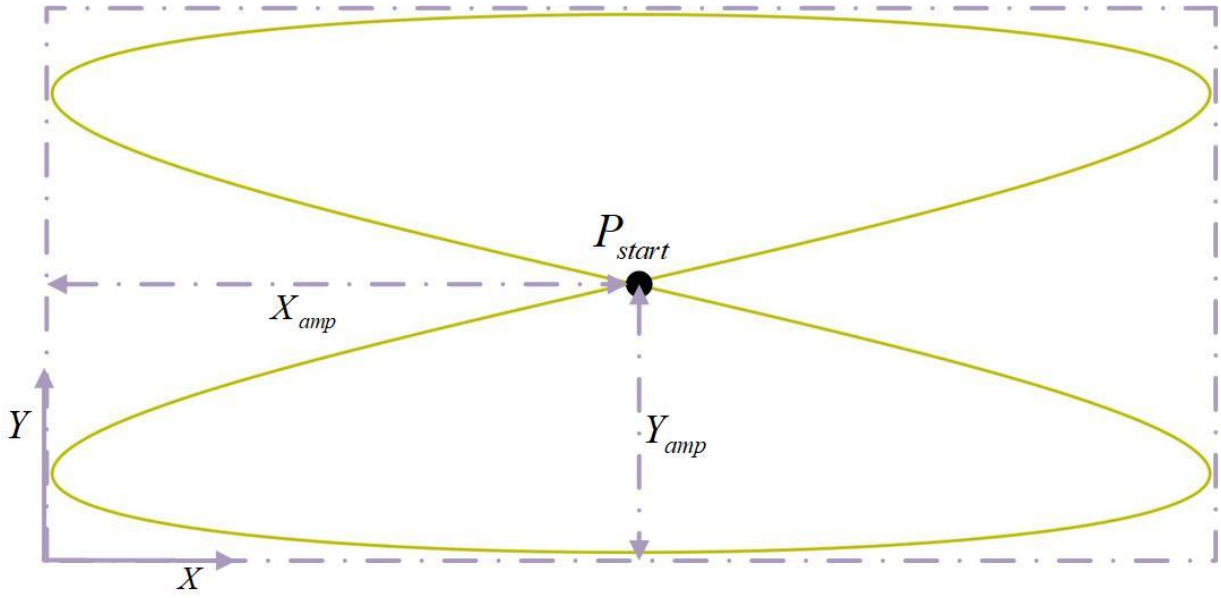
2、实际轨迹（期望轨迹+搜索运动）

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。

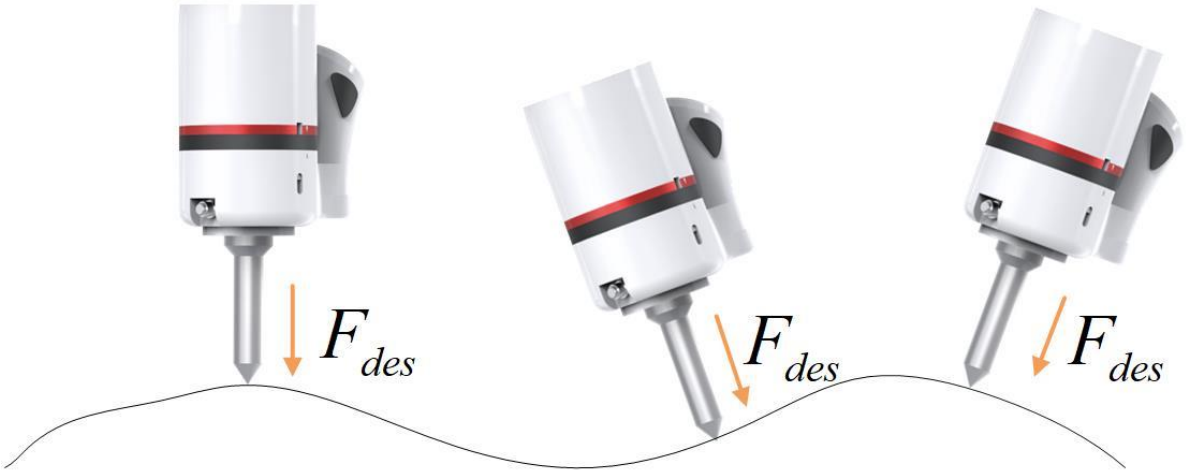


11.3.4 应用场景

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。



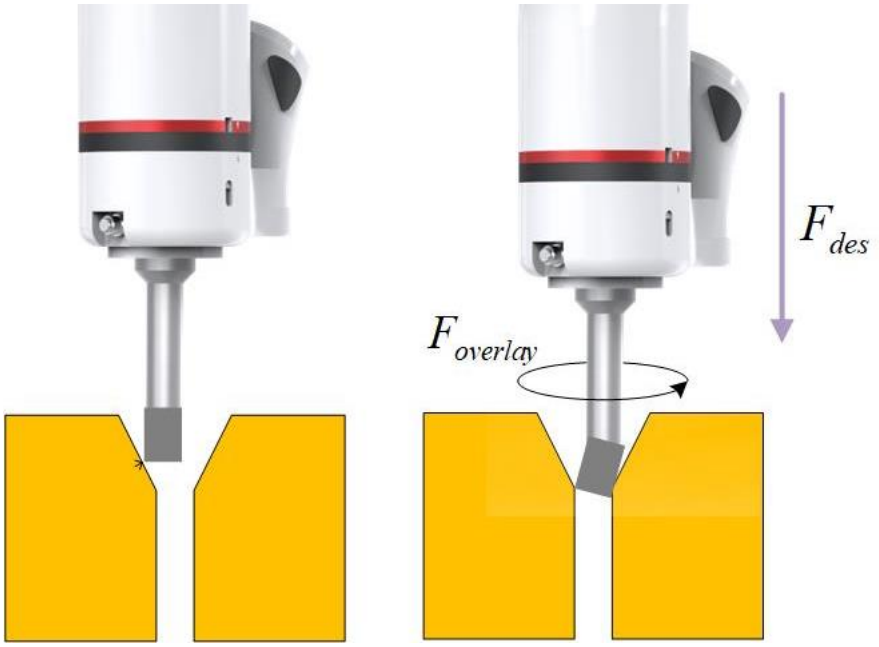
恒力打磨示例程序：程序的含义是，机器人设置为笛卡尔阻抗模式，设置阻抗刚度和负载信息，同时开启力控。通过施加 z 方向的期望力将工件压到打磨面上，在下压的过程中监控 z 方向的观测力，当此观测力超过一定的阈值，就认为工具已经接触曲面，这时施加一个打磨方向的期望轨迹。运动过程中机器人保持恒力，从而实现恒力打磨的效果。

```
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
```

12 机器人编程

12.1 变量

12.1.1 变量类型

12.1.1.1 基本变量

整型 int

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

例如，在变量列表中定义：

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。

浮点型 double

浮点数使用 8 个字节存储，使用时请不要超出取值范围。

例如，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。

布尔型 bool

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

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

表示定义了一个 bool 型的变量 ifClose，且初始值为 true。

字符串类型 string

字符串类型变量由多个字母或者数字组成，且定义时必须放在双引号""内。

表示定义了一个字符串变量 name，并初始化为"rokae"。

字符串类型变量支持"+"操作，可实现字符串拼接。

例如：name = "Rok" + "ae"

表示变量 name 被赋值为"Rokae"。

数组 Array

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

例如：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。

类型隐式转换

目前，在变量列表中进行数据赋值时，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。

例如，在定义整型变量 counter 时，只能输入整数，而不能输入小数。

12.1.1.2 byte

说明

byte 表示 RL 语言中的无符号字节，相当于 C++中的 unsigned char。取值范围 0~255，不允许为负值。一般用于 SocketSendByte 指令。

当 byte 赋值超限时，不会报错，会自动截断取低 8 位字节。

示例

下面的例子显示了如何使用 byte：

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。

12.1.1.3 clock

说明

clock 用于计时，clock 指令就像用于计时的秒表。

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

示例

下面的例子显示了如何使用 clock：

Example 1

```
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。

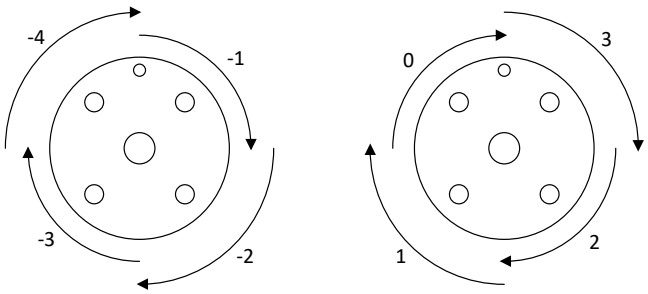
12.1.1.4 confdata

说明

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

对于带有冗余度的 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

12.1.1.5 jointtarget

说明

存储机器人关节角度和外部轴位置。

关节角度的单位是度 Degree，外部导轨的单位是毫米 mm。

定义

robax

机器人关节角度（Robot 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

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。

12.1.1.6 load

说明

load 变量类型用于存储机器人负载的动力学参数。

机器人的负载主要有两种：

- 安装在机器人末端的工具或工件本身；
- 工具所抓起/吸起来的物体。

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

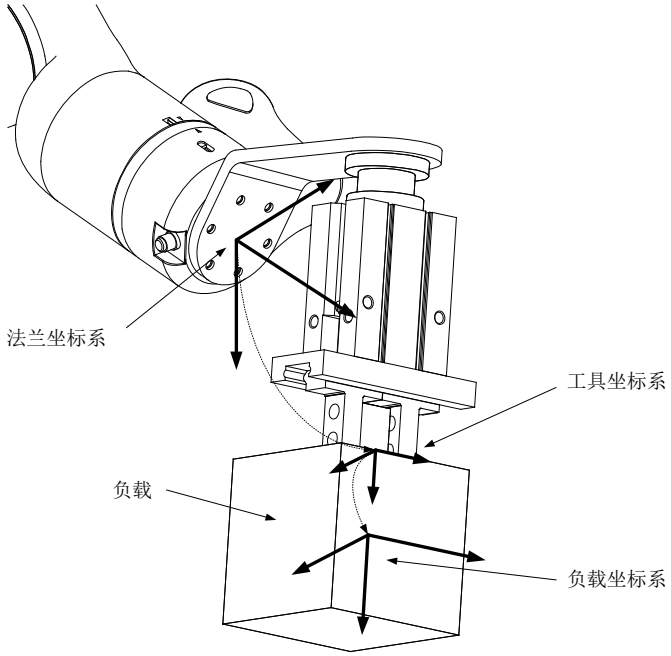
如果工具安装在机器人上，则 cogz 记录负载质心在工具坐标系下 Z 方向的偏移量；如果使用外部工具功能，则 cogz 记录被抓手夹持的负载质心在工件坐标系下 Z 方向的偏移量。

q1~q4

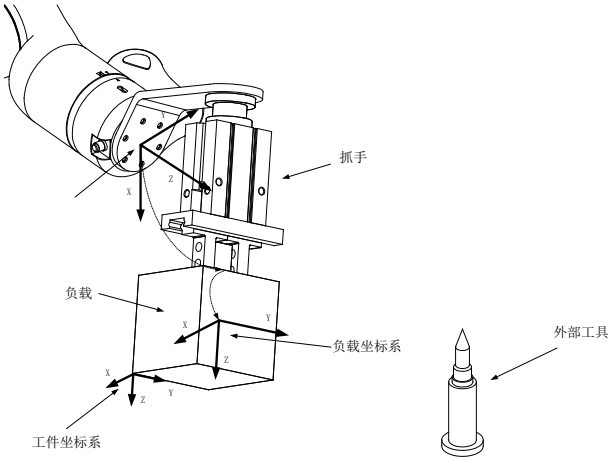
四元数，记录负载惯量主轴的方向。

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

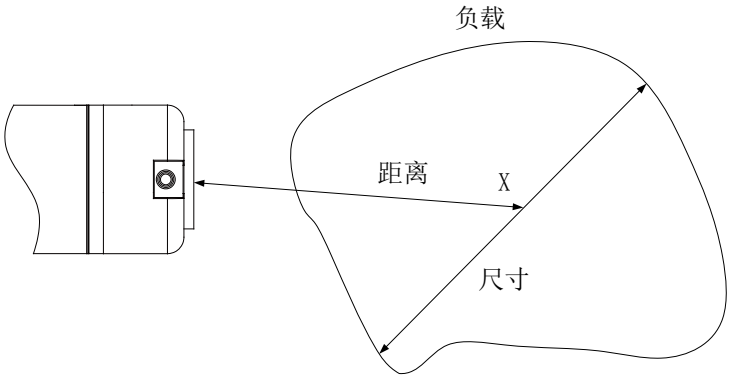
惯量 x

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。

12.1.1.7 orient

说明

存储坐标系或空间刚体的姿态信息。

orient 类型的变量不支持单独创建或者修改，仅作为某些变量的成员变量。

定义

RL 语言系统使用四元数来表示姿态，因此共有 4 个分量，表示形式为：

q1

数据类型：double

四元数的第一个分量。

q2

数据类型：double

四元数的第二个分量。

q3

数据类型：double

四元数的第三个分量。

q4

数据类型：double

四元数的第四个分量。

关于四元数

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

四元数的四个分量满足如下关系：

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

旋转矩阵和四元数之间可以相互转换，假设有一个旋转矩阵 R：

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

则:

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

12.1.1.8 pos

说明

用来存储三维空间的位置信息。

pos 类型的变量不支持单独创建或者修改，仅作为某些变量的成员变量。

定义

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

x

数据类型：double

位置的 x 坐标。

y

数据类型：double

位置的 y 坐标。

z

数据类型：double

位置的 z 坐标。

12.1.1.9 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。

12.1.1.10 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

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。

12.1.1.11 signalxx

说明

signalxx 类型的变量用于描述输入与输出信号。

所有 signalxx 类型的变量都需要在"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

Example 1

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

Example 2

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

注意事项

不支持在程序中定义/声明 signalxx 类型的变量，如果出现这种用法，程序将会报错。在使用 signalxx 类型的变量之前，请首先在 IO 信号列表中进行配置。

> 提示
>
> signalxx 型变量的作用域为 System，与其他作用域类型的关系为 System > GLOBAL > LOCAL。
> 如果 IO 配置界面的 Signal 与 RL 程序中声明的变量重名，那么较低作用域级别的变量将被选中

12.1.1.12 speed

说明

用来定义机器人和外部轴的运动速度。

为方便用户使用，系统预设了常用的速度变量，可通过辅助编程直接选择使用，详见插入指令。

定义

speed 型变量包含 5 个成员变量，分别是关节速度百分比，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。

12.1.1.13 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。

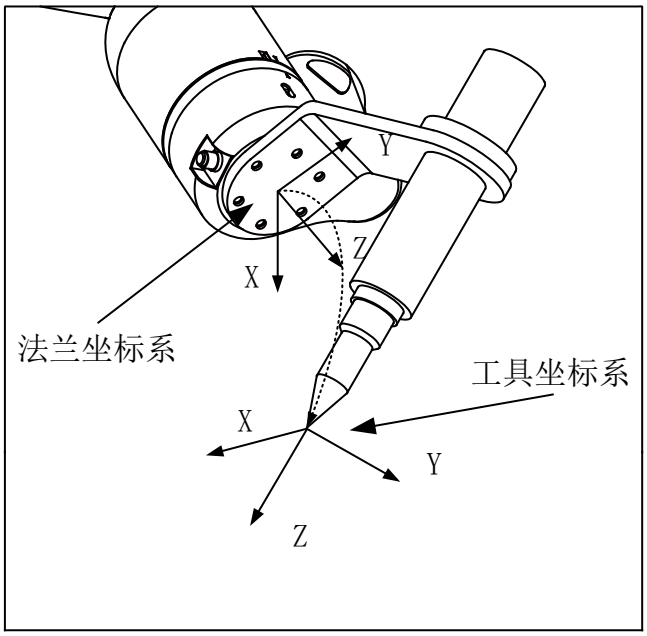
tframe

工具坐标系（Tool Frame）

数据类型：pose

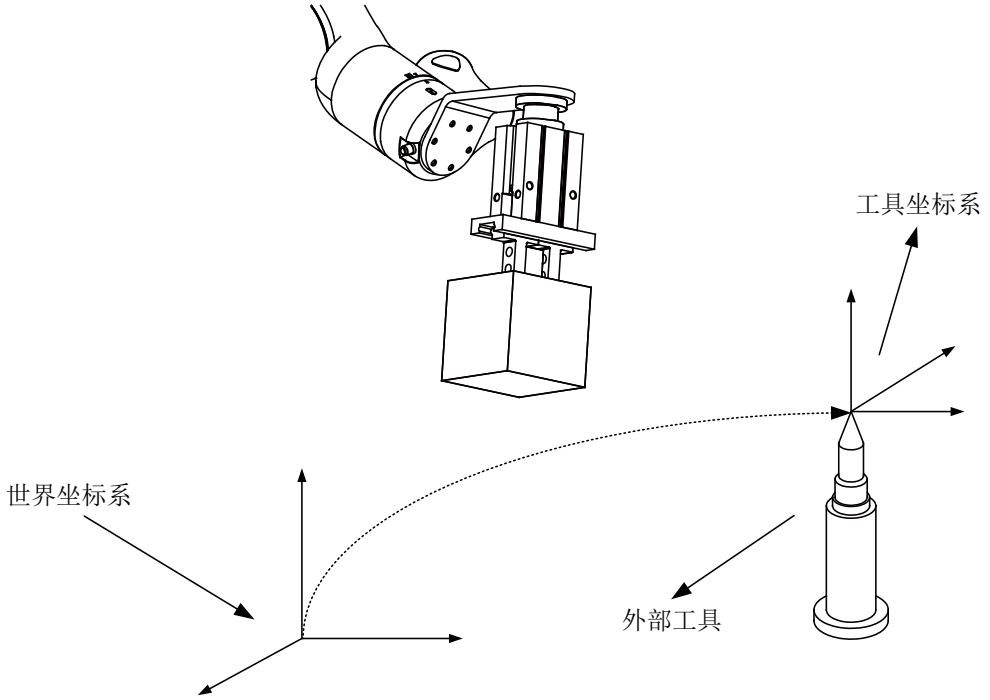
记录所用工具的工具坐标系，包括：

- 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。

12.1.1.14 triggdata

说明

triggdata 用于存储机器人定位事件信息（Positioning Event）。

定位事件指的是在运动过程中的某个特定位置触发一个输出信号或者运行一个中断程序。

triggdata 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 triggdata 类型的变量内存储什么样的信息，取决于所使用的 Trigg 指令，例如 TriggIO 等，然后就可以被对应的运动指令 TriggL、TriggC、TriggJ 等所使用。

示例

下面的例子显示了如何使用 triggdata：

Example 1

```
TriggIO gripopen, 0.5, gripopen, true
TriggL p2, v500, gripopen, fine, gripper
```

当机器人运动到距离 p2 点 0.5mm 的位置时，将 DO 信号 gripopen 置为 true。

12.1.1.15 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

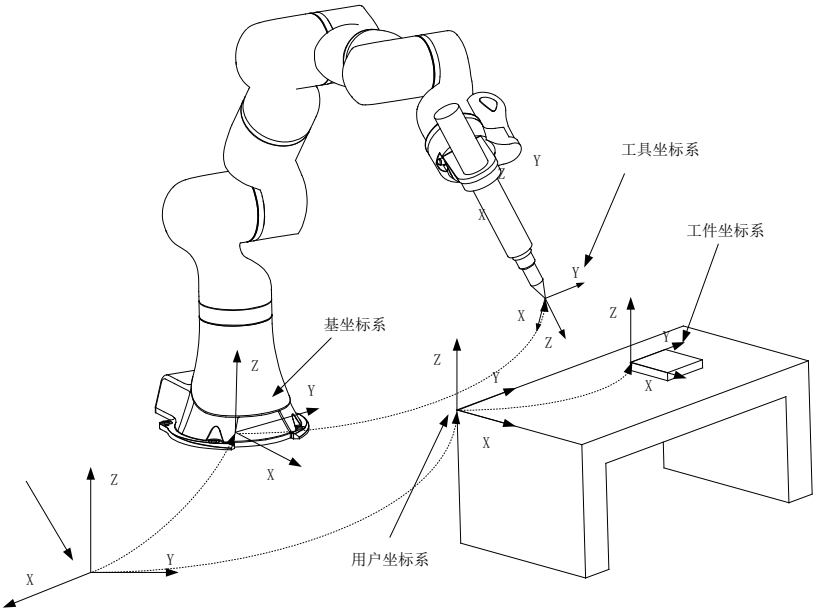
用户坐标系（User Frame）id

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。

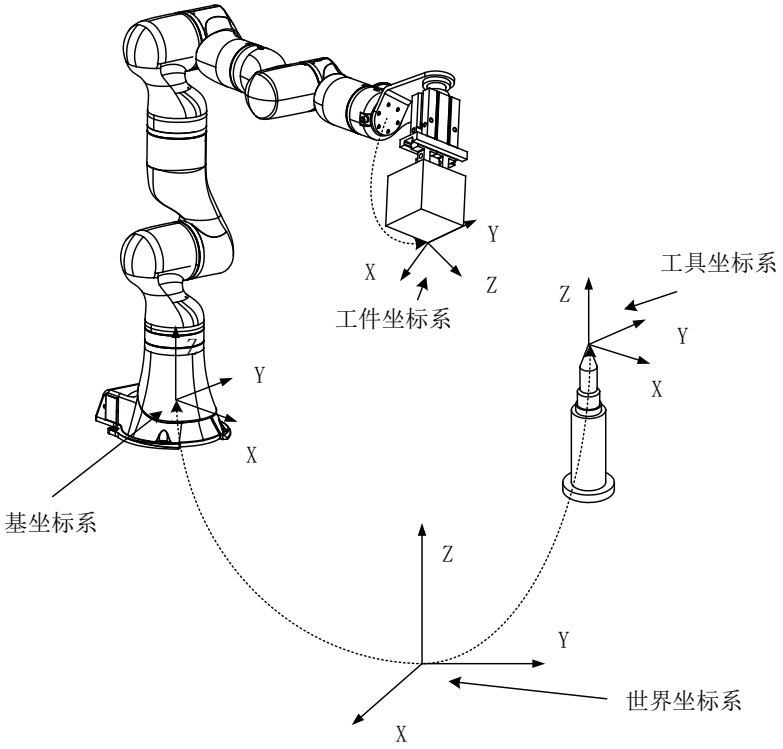
在使用普通工具时（非外部工具），坐标系的定义链为：

- 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。



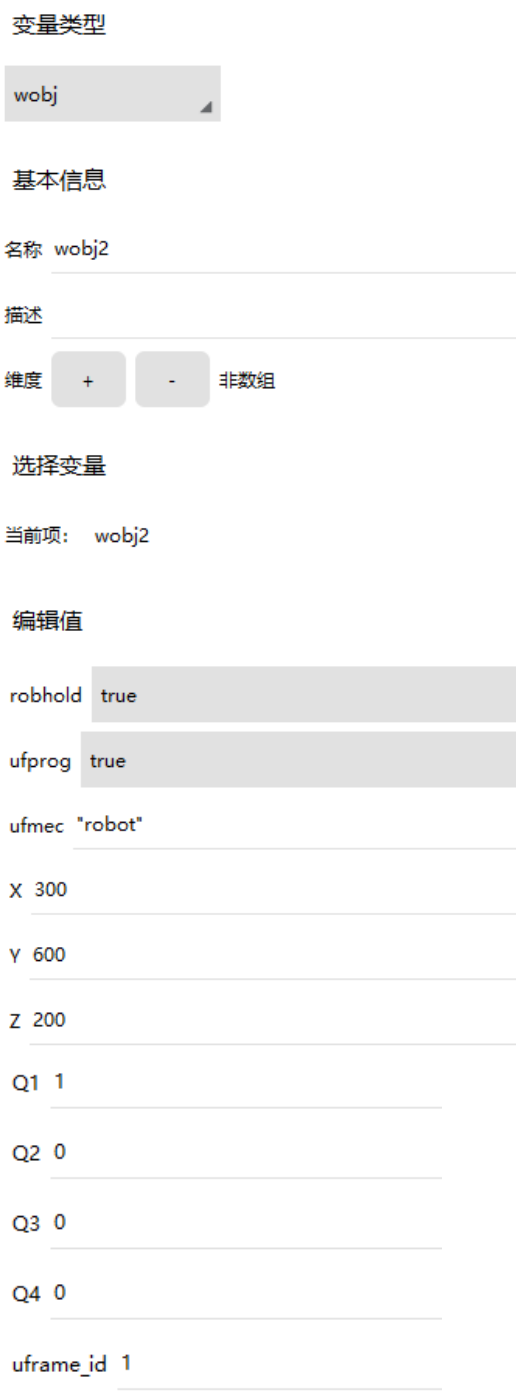
当使用外部工具时，坐标系的定义链为：

- 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。

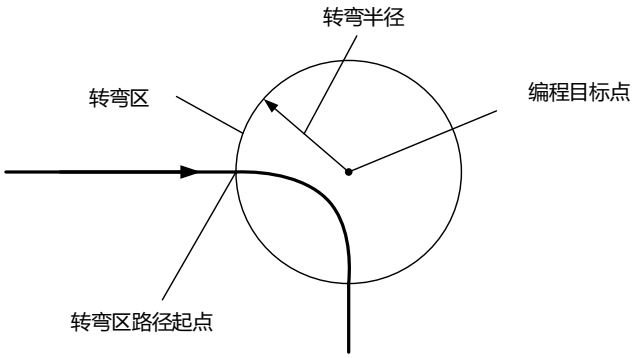
12.1.1.16 zone

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。

12.1.1.17 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

示例

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

12.1.2 运算

12.1.2.1 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：

Example 1

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

Example 2

++，--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。

Example 1

其他逻辑运算符用法示例如下：

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

```
    endif
endwhile
```

赋值运算符

赋值运算符包括：

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

各种赋值运算符用法示例如下

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

其他运算符

运算符	作用
()	圆括号
.	点操作符

各运算符用法示例如下：

Example 1

```
VAR int num = arr[1]          //取数组第一个元素赋给 num
VAR int num2 = (1+2)*3     //使用括号可以改变运算顺序，这里 num2 的值为 9
```

Example 2

定义一个 robtarget 变量 pt1

```
pt1.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。

12.1.2.2 基础运算优先级

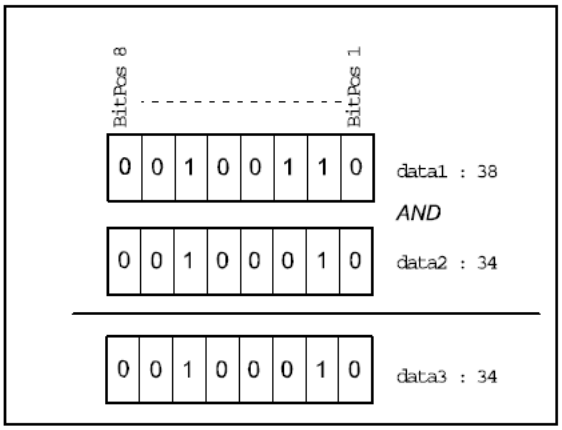
优先级	运算符	使用形式	结合方向
1	()	（表达式）/函数名(形参表)	
	.	变量名.	
2	-	-表达式	右到左

	++	++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	

12.1.2.3 BitAnd

说明

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



返回值

数据类型：byte

表示 2 个 byte 类型数据执行逻辑与返回的结果。

定义

BitAnd (BitData1, BitData2)

BitData1

数据类型：byte

要操作的字节数据 1。

BitData2

数据类型：byte

要操作的字节数据 2。

示例

Example 1

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

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

12.1.2.4 BitCheck

说明

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

返回值

数据类型：bool

true 表示指定位为 1，false 表示指定位为 0。

定义

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

BitData

数据类型：byte

要操作的字节数据。

BitPos

数据类型：int

要操作位的位置，范围 1~8。

示例

Example 1

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

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

12.1.2.5 BitClear

说明

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

定义

BitClear BitData | IntData, BitPos

BitData

数据类型：byte

要操作的字节数据。

IntData

数据类型：int

要操作的整型数据。

BitPos

数据类型：int

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

示例

Example 1

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

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

12.1.2.6 BitLSh

说明

BitLSh 用于对 byte 数据执行逻辑左移操作。

返回值

数据类型：byte

表示执行左移操作得到的 byte 数据。

定义

BitLSh (BitData, ShiftSteps)

BitData

数据类型：byte

要操作的字节数据。

ShiftSteps

数据类型：int

要左移的位数，范围 1~8。

示例

Example 1

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

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

12.1.2.7 BitNeg

说明

BitNeg 用于对 byte 数据执行逻辑非操作。

返回值

数据类型：byte

表示执行逻辑非操作得到的 byte 数据。

定义

BitNeg (BitData)

BitData

数据类型：byte

要操作的字节数据。

示例

Example 1

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

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

12.1.2.8 BitOr

说明

BitOr 用于对 byte 数据执行逻辑或操作。

返回值

数据类型：byte

表示执行逻辑或操作得到的 byte 数据。

定义

BitOr (BitData1, BitData2)

BitData1

数据类型：byte

要操作的字节数据 1。

BitData2

数据类型：byte

要操作的字节数据 2。

示例

Example 1

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

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

12.1.2.9 BitRSh

说明

BitRSh 用于对 byte 数据执行逻辑右移操作。

返回值

数据类型：byte

表示执行右移操作得到的 byte 数据。

定义

```
BitLSh (BitData, ShiftSteps)
```

BitData

数据类型：byte

要操作的字节数据。

ShiftSteps

数据类型：int

要右移的位数，范围 1~8。

示例

Example 1

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

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

12.1.2.10 BitSet

说明

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

定义

```
BitSet BitData | IntData, BitPos
```

BitData

数据类型：byte

要操作的字节数据。

IntData

数据类型：int

要操作的整型数据。

BitPos

数据类型：int

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

示例

Example 1

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

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

12.1.2.11 BitXOr

说明

BitXOr 用于对 byte 数据执行逻辑异或操作。

返回值

数据类型：byte

表示执行逻辑或操作得到的 byte 数据。

定义

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

BitData1

数据类型：byte

要操作的字节数据 1。

BitData2

数据类型：byte

要操作的字节数据 2。

示例

Example 1

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

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

12.1.2.12 ByteToStr

说明

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

返回值

数据类型：string

转换得到的 string 数据。

定义

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

BitData

数据类型：byte

要转换的 byte 数据，默认按照十进制转换。

\Hex

标识符，按 16 进制转换。

\Okt

标识符，按 8 进制转换。

\Bin

标识符，按 2 进制转换。

\Char

标识符，按 Ascii 码字符格式转换。

示例

Example 1

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

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

12.1.2.13 ClkRead

说明

ClkRead 用于读取计时器的值。

返回值

数据类型：double

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

定义

ClkRead (Clock)

Clock

数据类型：clock

计时器名称。

示例

Example 1

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

interval 存储 clock1 在启动和停止之间的时间间隔。

12.1.2.14 ClkReset

说明

ClkReset 用于重置一个计时器。

使用一个计时器前可通过 ClkReset 保证计数为 0。

定义

ClkReset Clock

Clock

数据类型：clock

计时器名称。

示例

Example 1

```
VAR clock clock1
ClkReset clock1
```

重置 clock1。

12.1.2.15 ClkStart

说明

ClkStart 用于启动一个计时器。

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

定义

ClkStart Clock

Clock

数据类型：clock

计时器名称。

示例

Example 1

```
VAR clock clock1
ClkStart clock1
```

声明 clock1，启动计时器 clock1。

12.1.2.16 ClkStop

说明

ClkStop 用于停止一个计时器。

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

定义

```
ClkStop Clock
```

Clock

数据类型：clock

计时器名称。

示例

Example 1

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

停止计时器 clock1。

12.1.2.17 CONNECT

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

Example 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

12.1.2.18 DecToHex

说明

将十进制的数转换成十六进制数。

返回值

数据类型：string

表示转换得到的 16 进制数据，用 0-9，a-f，A-F 表示。

参数

DecToHex(str)

str

表示要被转换的十进制数据，用 0-9 表示。

数据类型：string

使用限制

- 数据范围从 0~2147483647 或 0~7fffffff

12.1.2.19 DoubleToByte

说明

用于将 double 或 double 数组转换成为 byte 数组。

返回值

数据类型：byte 数组

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

参数

DoubleToByte(dou1)

dou1

要转换的 double 变量。

数据类型：double

12.1.2.20 DoubleToStr

说明

将 double 类型的变量转换为 string。

参数

DoubleToStr(Val, Dec)

Val

要转换的 double 变量。

数据类型：double

Dec

要保留的小数点位数。

数据类型：int

使用限制

➢ 小数点位数最多 15 位。

12.1.2.21 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

要转换的 16 进制数据，用 0-9，a-f，A-F 表示。

数据类型：string

使用限制

➢ 数据范围从 0~2147483647 或 0~7fffffff。

12.1.2.22 IntToByte

说明

用于将 int 或 int 数组转换成为 byte 数组。

返回值

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

参数

IntToByte(int1)

int1

要被转换的整数变量或数组。

数据类型：int 或 int 数组

使用限制

数据范围从-2147483647~2147483647。

12.1.2.23 IntToStr

说明

用于将整数转换成字符串。

返回值

转换得到的字符串。

参数

IntToStr(int1)

int1

要被转换的整数变量。

数据类型：int

使用限制

数据范围从-2147483647~2147483647。

12.1.2.24 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。

12.1.2.25 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。

12.1.2.26 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。

12.1.2.27 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。

12.1.2.28 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

12.1.2.29 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。

12.1.2.30 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

12.1.2.31 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。

12.1.2.32 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。

12.1.2.33 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。

12.1.2.34 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。

12.1.2.35 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

12.1.2.36 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

12.1.2.37 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

12.1.2.38 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。

12.1.2.39 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。

12.1.2.40 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

12.1.2.41 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。

12.1.2.42 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。

12.2 字符串操作

12.2.1 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

Example 1

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

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

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

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

12.2.2 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

Example 1

VAR int num
num = StrLen(“Robotics”) //8
字符串“Robotics”长度为 8。

12.2.3 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

Example 1

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

使用限制

FromMap 和 ToMap 必须匹配，长度一致。

12.2.4 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

Example 1

```
VAR int found
Found = StrMatch(“Robotics”, 1, “bo”) //3
```

从第 1 个字符开始搜索“bo“，发现第 3 个位置匹配，返回匹配的首字符所在位置 3。

12.2.5 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

Example 1

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

第 2 个字符 o 属于字符集合”aeiou”中一员，返回 true。

12.2.6 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

Example 1

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

12.2.7 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

Example 1

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

从第 1 个位置开始截取长度为 5 的字符串，得到″Robot″。

12.2.8 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

Example

```
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

1. 输入字符串为空时报错

2. 分割结果和定义字符串的长度不匹配时报错

12.2.9 StrToByte

说明

StrToByte 用于将一个特定格式的字符串转换为 byte 数据。。

返回值

数据类型： byte

表示转化得到的 byte 数据。

定义

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

ConStr

数据类型： string

表示要被转换的字符串，如果可选参数不存在，默认转化为十进制。

\Hex

标识符，按 16 进制转换。

\Okt

标识符，按 8 进制转换。

\Bin

标识符，按 2 进制转换。

\Char

标识符，按 Ascii 码字符格式转换。

示例

Example 1

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

使用限制

1、按十进制，不能超过 0~255，超出则报错；

2、按 16 进制，不能超过 FF，超出则报错；

3、按 8 进制，不能超过 377，超出则报错；

4、按 2 进制，不能超过 11111111，超出则报错；

12.2.10 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

Example 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

12.2.11 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

Example 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 运动指令

12.3.1 MoveJ

说明

MoveJ （Move The Robot By Joint Movement） 用于对机器人末端运动轨迹没有要求的场合，使机器人快速的从一个点运动到另一个点。所有的轴同步运动，机器人末端沿一条不规则的曲线移动，请注意是否有发生碰撞的危险。

MoveJ 指令与 MoveAbsJ 最大的区别在于给定的目标点格式不同。 MoveJ 的目标点是工具

（TCP）的空间位姿而不是关节轴角度。

示例

以下是一些 MoveJ 用例：

Example 1

MoveJ p30， v100， z50， tool1

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

Example 2

MoveJ endpoint， v500， z50， gripper， wobj2

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

参数

MoveJ ToPoint， Speed， Zone， Tool， [Wobj]

其中，带[ ]的参数为可选参数，可不写。

ToPoint

目标位姿（To Point）

数据类型： robtarget

在笛卡尔空间描述的目标位置。

Speed

运动速度（Move Speed）

数据类型： speed

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

Zone

转弯区（Turning Zone）

数据类型： zone

用来定义当前轨迹的转弯区大小。

Tool

数据类型： tool

执行该轨迹时使用的工具。

MoveJ 指令使用工具的 TCP 数据来计算运动速度和转弯区大小。

[Wobj]

工件（Work Object）

数据类型： wobj

执行该轨迹时使用的工件。

当工具安装在机器人上时，该参数可以忽略；

当使用外部工具时，必须要指定该参数，且机器人将使用 wobj 中存储的数据来计算运动速度和转弯区大小。

12.3.2 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。

示例

以下是一些 MoveL 用例：

Example 1

MoveL p10， v1000， z50， tool0

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

Example 2

MoveL endpoint， v500， z50， gripper， wobj2

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

参数

MoveL ToPoint， Speed， Zone， Tool， [Wobj]

其中，带[ ]的参数为可选参数，可不写。

ToPoint

目标位姿（*To Point*）

数据类型： robtarget

在笛卡尔空间描述的目标位置。

Speed

运动速度（*Move Speed*）

数据类型： speed

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

Zone

转弯区（*Turning Zone*）

数据类型： zone

用来定义当前轨迹的转弯区大小。

Tool

数据类型： tool

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

[Wobj]

工件（*Work Object*）

数据类型： wobj

执行该轨迹时使用的工件。

当工具安装在机器人上时，该参数可以忽略；

当使用外部工具时，必须要指定该参数，且机器人将使用 wobj 中存储的数据来计算运动速度和转弯区大小。

12.3.3 MoveAbsJ

说明

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

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

示例

以下是一些 MoveAbsJ 用例：

Example 1

MoveAbsJ j10， v500， fine， tool1

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

Example 2

MoveAbsJ startpoint， v1000， z100， gripper， phone

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

参数

MoveAbsJ ToJointPos， Speed， Zone， Tool， [Wobj]

其中，带[ ]的参数为可选参数，可不写。

TojointPos

目标关节角度（*To Joint Position*）

数据类型： jointtarget

机器人和外部轴的目标角度和位置值。

Speed

运动速度（*Move Speed*）

数据类型： speed

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

Zone

转弯区（*Turning Zone*）

数据类型： zone

用来定义当前轨迹的转弯区大小。

Tool

数据类型： tool

执行该轨迹时使用的工具。

MoveAbsJ 指令使用工具的 TCP 数据来计算运动速度和转弯区大小

[Wobj]

工件（*Work Object*）

数据类型： wobj

执行该轨迹时使用的工件。

当工具安装在机器人上时，该参数可以忽略；

当使用外部工具时，必须要指定该参数，且机器人将使用 wobj 内存储的数据来计算运动速度和转弯区大小。

12.3.4 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。

示例

以下是一些 MoveC 用例：

Example 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。

Example 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。

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。

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）。当信号持续变量的值变为所需值时，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。

示例

以下是一些 SearchL 用例：

Example 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

Example 2

SearchL \pstop, 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，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

12.3.5 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）。当信号持续变量的值变为所需值时，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。

示例

以下是一些 SearchL 用例：

Example1

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

Example2

SearchL \pstop, 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，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 save_rob

参数

SearchL [arg,] di, save_rob, target_rob, Speed, Tool [,Wobj]

其中，带[ ]的参数为可选参数，可不写。

arg

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

Search 指令触发特定行为的信号，使用用户自定义 DI 信号

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。

12.3.6 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）。当信号持续变量的值变为所需值时，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。

示例

以下是一些 SearchL 用例：

Example 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

Example 2

SearchC \pstop, di0, 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 跳变为高电平，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

参数

SearchC [arg,] di, save_rob, aux_rob, target_rob, Speed, Tool [,Wobj]
其中，带[ ]的参数为可选参数，可不写。

arg

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

Search 指令触发特定行为的信号，使用用户自定义 DI 信号

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。

12.3.7 Offs

说明

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

返回值

数据类型： robtarget

偏移后的新位姿。

定义

Offs（Point， XOffset， YOffset， ZOffset）

Point

数据类型： robtarget

待偏移的位置点，或者说偏移指令的初始点。

XOffset

数据类型： double

沿工件坐标系 x 方向上的偏移量。

YOffset

数据类型： double

沿工件坐标系 y 方向上的偏移量。

ZOffset

数据类型： double

沿工件坐标系 z 方向上的偏移量。

示例

```
p11=Offs(p10,100,200,300)
```

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

> 提示
>
> 该指令暂时不支持辅助编程。

12.3.8 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 接触 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

```
//用户手动输入 robtarget
//p1.trans.x = ….
//只修改了坐标，未修改 cf 参数

//该指令很可能导致执行失败
MoveJ p1, v1000 ….

//关闭 conf 检查
ConfL Off
MoveJ p1, v1000 ….
//机器人能够运动到 p1 点，但是姿态不确定
//打开 conf 检查
ConfL On
```

12.3.9 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
```

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。

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

```
// 加速度、加加速度设置为默认的一半
AccSet  50,15
```

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

12.4 逻辑指令

12.4.1 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。

12.4.2 Wait

说明

程序等待一段时间，范围是 0~2147484 秒。

示例

Example1

Wait 2

表示等待 2s 的时间。

12.4.3 WaitUntil

说明

程序等待某个条件成立。

示例

Example1

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。

12.4.4 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

Example 1

```
VAR int counter = 0
WHILE(1)
    IF(counter == 5)
        break
    Endif
counter++
ENDWHILE
```

该程序在执行到 counter 等于 5 时会跳出 WHILE 循环。

12.4.5 IF…Else if…Else

说明

条件判断语句。

示例

Example 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。

12.4.6 Goto

说明

Goto 语句允许把指针跳转到被标记的语句。

示例

Example 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。

12.4.7 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

Example 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。

Example 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。

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。

12.4.8 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

Example 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。

12.5 起始点指令

12.5.1 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

Example 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。

12.5.2 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

Example 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。

12.5.3 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

Example 1

HomeSet 0,30,0,60,0,90,0
double angle2 = HomeSetAt(2)
angle2 获得关节 2 的关节角 30°

12.5.4 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

12.5.5 HomeSpeed

说明

设定 Home 指令的运行速度

定义

HomeSpeed Speed

示例

Example 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。

12.5.6 HomeClr

说明

清除初始点设置

定义

HomeClr

示例

Example 1

```
HomeClr
```

清除程序中设定的起始点。清楚后 Home 指令将无法执行。

12.6 力控指令

12.6.1 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

Example 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。

12.6.2 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

Example 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。

12.6.3 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

Example 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.

12.6.4 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

Example 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。

12.6.5 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

Example 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。

12.6.6 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

Example 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。

12.6.7 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

Example 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。

12.6.8 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

Example 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。

12.6.9 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

Example 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。

12.6.10 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

Example 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。

12.6.11 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

Example 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。

12.6.12 PauseOverlay

说明

暂停搜索运动。

定义

PauseOverlay
没有参数，直接使用。

示例

Example 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。

12.6.13 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

Example 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。

12.6.14 StopOverlay

说明

停止搜索运动。

定义

StopOverlay
没有参数，直接使用。

示例

Example 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。

12.6.15 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

Example 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。

12.6.16 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

Example 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。

12.6.17 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

Example 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。

12.6.18 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

Example 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。

12.6.19 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

Example 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。

12.6.20 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

Example 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。。

12.6.21 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

Example 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。

12.6.22 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

Example 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
```

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。

12.6.23 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

Example 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

Example 2

MotionSup On, "High"
开启碰撞检测并设置检测灵敏度为高

12.7 通信指令

12.7.1 SocketCreate

说明

建立一个 Socket 连接，通过使用 Socket 指令，RL 程序可以从外部设备获取数据或者向外发送 程序数据。 RL 语言支持同时建立多个不同的 Socket 以便于连接多个外部设备，不同 Socket 之间采用不同的名称来进行区分。Socket 指令是基于 TCP/IP 协议的，因此理论上任何支持 TCP/IP 的外部设备都可以和 RL 程序通信以交换数据。所有发送给 RL Socket 指令的数据（即使用 SocketRead 系列指令接收的数据），都应该以"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。

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

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

提示

1、 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。
2、 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")
```

12.7.2 SocketClose

说明

关闭 Socket。

定义

SocketClose（”SocketName”）

SocketName

数据类型：string
需要关闭的 Socket 名称。

提示

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

示例

SocketClose（″Socket0”）

12.7.3 SocketSendString

说明

通过 Socket 对外发送一个字符串。

定义

SocketSendString（StringData，”SocketName”）

StringData

数据类型：string
待发送的 string 数据。

SocketName

数据类型：string
用于发送数据的 Socket 名称。

示例

Example 1

SocketSendString（″Hello World″，”Socket0”）

通过 Socket0 对外发送 Hello World 字符串。

Example 2

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

12.7.4 SocketSendByte

说明

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

定义

SocketSendByte(ByteData, ″SocketName″)

ByteData

数据类型：int 或 byte 或 byte 数组
发送一个 0~255 的无符号字节或数组，主要用于发送 ASCII 码。

SocketName

数据类型：string
用于发送数据的 Socket 名称。

示例

Example 1

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

Example 2

VAR byte data1 = 13
SocketSendByte(data1, ″socket0″)
首先定义一个 byte 类型变量 data1，它其实是一个回车符。然后通过 socket0 对外发送。

Example 3

VAR byte data2[2] = {13,17}
SocketSendByte(data2, ″socket0″)
通过 socket0 发送一个 byte 类型数组变量 data2。

12.7.5 SocketReadBit

说明

通过 Socket 按 bit 接收数据，外部发送的数据需以回车结尾。

返回值

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

定义

SocketReadBit(BitNum, TimeOut, ″SocketName″)

BitNum

需要读取的 bit 数量，大小应该为 8 的整数倍。

数据类型：int

TimeOut

超时时间。单位 s，范围 0~86400，默认 60s。

数据类型：int

SocketName

用于接收数据的 Socket 名称。

数据类型：string

示例

```
bool groupio[16]
groupio = SocketReadBit(16, 60, "Socket0")
```

通过 SocketReadBit 指令读取 16 个 bit 的数据存储到名为 groupio 的布尔型数组中，超时时间 60s。

12.7.6 SocketReadDouble

说明

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

返回值

数据类型：double

使用 double 型数组存储接收到的数据。

定义

```
SocketReadDouble(DoubleNum, TimeOut, "SocketName")
```

DoubleNum

需要读取的 double 数个数，最大为 30 个。

数据类型：double

TimeOut

超时时间。单位 s，范围 0~86400，默认 60s。

数据类型：int

SocketName

用于接收数据的 Socket 名称。

数据类型：string

示例

```
double dd[10]
dd = SocketReadDouble(10, 60, "Socket0")
```

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

12.7.7 SocketReadInt

说明

通过 Socket 接收 int 型数据，外部发送的数据需以回车结尾。

返回值

数据类型：int

使用 int 型数组存储接收到的数据。

定义

SocketReadInt(IntNum, TimeOut, ″SocketName″)

IntNum

需要读取的 int 数个数，最大为 30 个。

数据类型：int

TimeOut

超时时间。单位 s，范围 0~86400，默认 60s。

数据类型：int

SocketName

用于接收数据的 Socket 名称。

数据类型：string

示例

```
int ii[10]
ii = SocketReadInt(10, 60, ″Socket0″)
```

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

12.7.8 SocketReadString

说明

从 Socket 读取一个字符串并返回，外部发送的数据应以回车结尾。

返回值

存储接收到的字符串

数据类型： string。

定义

SocketReadString(TimeOut, ″SocketName″)

TimeOut

超时时间。 单位 s， 范围 0~86400，默认 60s。

数据类型： int

SocketName

用于接收数据的 Socket 名称。

数据类型：string

示例

```
VAR String str1
str1 = SocketReadString(60, ″Socket1″)
```

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

12.8 IO 指令

12.8.1 SetDO

说明

设置某个数字输出信号的值。

定义

SetDO DoName | DO_ALL，Value

DoName

数据类型：signaldo

指定需要改变状态的 DO 信号名称，必须是已经在输入输出界面定义过的变量。

DO_ALL

数据类型：标识符。

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

Value

数据类型：bool

DO 信号的目标状态，仅支持 true 和 false。

示例

SetDO do2，true

将 do2 对应的数字输出点置为高电平。

12.8.2 SetGO

说明

设置某个组输出的值。

定义

SetGO GoName，Value

GoName

数据类型：signalgo

指定需要改变值的 go 信号名称，必须是已经在输入输出界面定义过的变量。

Value

数据类型：int

go 信号的目标值

示例

SetGO go3，8

将 go3 对应的一组物理端口的值设置为 8.

12.8.3 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。

12.9 函数

12.9.1 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。

12.9.2 Print

说明

将用户定义的内容打印输出到示教器，用户可以使用该函数对程序进行调试。

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

系统将这些变量转换为字符串并串接在一起，最后输出到程序编辑器的调试窗口。

定义

Print（ var1， var2，……）

示例

Example 1

```
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。

12.9.3 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

Example 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。

12.9.4 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

Example 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。

12.9.5 CRobt

说明

用于获取机器人位姿。

使用该函数时，需要给定工具名称和工件名称，返回指定工具坐标系的 pose，当前的轴配置信息以及外部轴位置。

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

返回值

数据类型： robtarget

返回当前机器人的位置、姿态、轴配置数据以及外部轴信息。

定义

CRobT（ Tool， Wobj）

Tool

数据类型： tool

计算位置时使用的工具。

Wobj

数据类型： wobj

计算位置时使用的工件。

示例

Example 1

```
p2 = CRobT( tool1， wobj2 );
```

12.9.6 CJointT

说明

CJointT 用于读取机器人轴和外部轴当前角度。

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

返回值

数据类型：jointtarget

旋转轴单位：度，线性轴单位：mm

返回机器人轴和外部轴的当前角度值

定义

CJointT（）

数据类型：函数

示例

Example 1

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

12.9.7 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

12.9.8 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

12.9.9 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, Tobj])

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

example1

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。

example2

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。

example3

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
>
> 该指令暂时不支持辅助编程。

12.9.10 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

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

13 附录-名词解释

- 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。
- Project 工程，控制机器人运行的程序、任务等对象的管理集合，具体包括：Task List-任务列表、Program-程序、Point List-示教点位列表、Path List-示教路径列表、Variables-变量列表、IO-用户 IO、Tool-工具、Wobj-工件、Coordinates-坐标系。一个工程的数据对象可以导出，在其他工程或机器人复用。
- 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。

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