DOBOT

Dobot CRStudio 用户手册

（CR & Nova）

深圳市越疆科技有限公司

目录

前言

目的

本手册介绍了Dobot协作机械臂的移动端控制APP Dobot CR Studio，方便用户了解和使用CR Studio控制协作机械臂。

读者对象

本手册适用于：

- 客户
- 销售工程师
- 安装调测工程师
- 技术支持工程师

修订记录

时间	修订记录
2023/05/29	更新至Android 4.12.0/iOS 2.13.0版本
2023/01/13	更新至Android 4.11.0/iOS 2.12.0版本
2022/12/09	更新至Android 4.10.0/iOS 2.11.0版本
2022/10/31	更新至Android 4.9.0/iOS 2.10.0版本
2022/08/25	更新至Android 4.8.0/iOS 2.9.0版本

符号约定

在本手册中可能出现下列标志，它们所代表的含义如下。

符号	说明
⚠危险	表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害
⚠警告	表示有中度或低度潜在危害，如果不能避免，可能导致人员轻微伤害、机械臂毁坏等情况
⚠注意	表示有潜在风险，如果忽视这些文本，可能导致机械臂损坏、数据丢失或不可预知的结果
🕮说明	表示是正文的附加信息，是对正文的强调和补充

1 从这里开始

欢迎使用CR Studio。CR Studio是越疆针对Dobot机械臂开发的移动端控制软件，具有功能简单易用，实用性强，界面简洁易懂等特点，可以帮助用户快速掌握Dobot机械臂的用法。

本手册主要介绍如何使用CR Studio控制CR系列和Nova系列机械臂。控制CR系列和Nova系列的方法大致相同，本手册主要以CR系列为例进行说明。

如果您是初次使用CR Studio，我们建议您按照如下顺序阅读本手册。

1. 连接机器人：将CR Studio连接至机械臂。
2. 主界面说明：了解CR Studio的主界面，大致了解CR Studio有哪些功能。
3. 设置：根据实际需求对机械臂进行配置。如果机械臂采用吊顶式、壁挂式安装或者呈一定角度安装，则需要在下使能状态下设置旋转角度和倾斜角度，请先参考本体安装进行配置。
4. 监控：了解CR Studio提供的监控能力。如果需要安装末端插件，请参考末端插件进行配置。
5. 编程/工艺：了解CR Studio提供的编程和工艺能力，尝试创建自己的工程吧。
6. 远程控制：工程开发完后，尝试通过远程控制来运行工程吧。

2 连接机器人

在连接机器人前，需确保控制柜已安装WiFi模块。

1. 在平板搜索并连接机器人的WiFi，WiFi的默认SSID为“Dobot_WIFI_XXX”，其中XXX为位于机械臂底座上的机械臂编号；初始WiFi密码为：1234567890。

   用户可在管理员权限下，在软件设置中修改WiFi的SSID和密码，重新启动控制柜后生效。

2. 在主界面单击“开始连接”按钮，连接控制柜。

   连接成功后，“开始连接”按钮变为“断开连接”按钮，同时按钮左侧显示WiFi图标，如下图所示。

3 主界面说明

3.1 主界面总览

CR Studio支持安卓和iOS两个平台，建议安装在安卓平板/iPad上使用。两个平台的APP略有差异，但基本功能与UI大致相同。本手册以安卓平板为例对CR Studio的使用进行说明。

启动APP后，首先会显示主界面，如下图所示。单击屏幕下方的 可查看主界面功能简介。

序号	说明
1	顶部工具栏
2	机器人仿真面板
3	主要功能面板
4	点动板

3.2 顶部工具栏

顶部工具栏在APP使用过程中始终位于屏幕顶部（可通过  按钮隐藏），功能如下图所示。

序号	名称	说明
1	菜单按钮	单击后弹出下述的菜单。 • 帮助：查看帮助文档。 • 锁屏：强制锁屏。仅为APP锁屏，和系统锁屏无关，解锁需要输入管理员密码或锁屏密码。 • 切换语言：切换APP的显示语言，设置后需重启APP才会生效。 • 崩溃日志：上传APP崩溃日志，协助我们排查崩溃问题。 • 版本信息：查看机器人和APP相关的版本信息。 • 关于我们：查看越疆科技公司信息。 • 退出app：退出CR Studio。
2	主页按钮	单击该按钮可返回主界面。从其它界面通过单击该按钮返回主界面时，那个页面未保存的内容不会丢失。
3	连接按钮	连接机器人的WiFi后，单击“开始连接”按钮使APP与该机器人相连，详见连接机器人。连接成功后按钮变为“断开连接”，单击可断开与机器人之间的连接。
4	全局速率	调整全局速率。全局速率为机械臂实际运行速度的计算因子，速度计算方法详见点动设置与再现设置。
5	报警信息	机械臂使用过程中出现异常会触发报警。无报警时图标为白色，有报警时图标会变为红色。单击可打开报警信息界面。
6	使能按钮	单击可切换机械臂使能状态，详见使能状态说明。
7	监控按钮	单击可打开监控界面，详见IO监控。
8	急停按钮	机械臂运行过程中出现突发情况时可按下急停开关，机械臂紧急停止并断电，详见急停按钮说明。

报警信息

报警信息界面如下图所示。

您可在此界面中查看报警的信息，其中类型用于区分报警是机械臂还是控制柜发出的。

存在报警时，机械臂如果未下电，末端的指示灯会变为红色常亮。

解决报警的原因后，单击“清除报警”按钮可以清除报警。

使能状态说明

机械臂需要进入上使能状态后才可以工作。

- 使能按钮图标为蓝色（  ）时，表示机械臂处于下使能状态。单击按钮会先后弹出角度确认和负载设置窗口（末端负载的偏心坐标需在J6轴为0°时设置，负载重量（含末端治具重量）不得超过CR机械臂允许的最大负载重量），确认后机械臂会开始上使能。此时机械臂会开始小幅度运动，然后末端指示灯变为绿色常亮，表示机械臂已上使能。同时，使能按钮图标也会变为绿色（  ）。

  > **注意：**
  >
  > 负载设置不正确可能导致出现碰撞检测异常报警或者拖拽时本体不受控制。

①

②

- 使能按钮图标为绿色时，表示机械臂处于上使能状态。单击按钮会弹出下使能的确认框，确认后机械臂会开始下使能。此时机械臂末端的指示灯会变为蓝色常亮，表示机械臂已下使能。同时，使能按钮图标也会变为蓝色。

- 使能按钮为蓝色闪烁状态状态时，表示机械臂处于拖拽模式，此时不能下使能和通过控制软件控制机械臂运动（运行工程、点动、RunTo指定姿态等）。
- 如果在CR Studio和机械臂连接状态中，机械臂被下电，使能按钮会变为白色，要改变使能状态需要先给机器人上电。

急停按钮说明

急停按钮被按下后，机械臂会急停并断电，按钮图标变为被按下（  ）状态，APP显示如下界面。

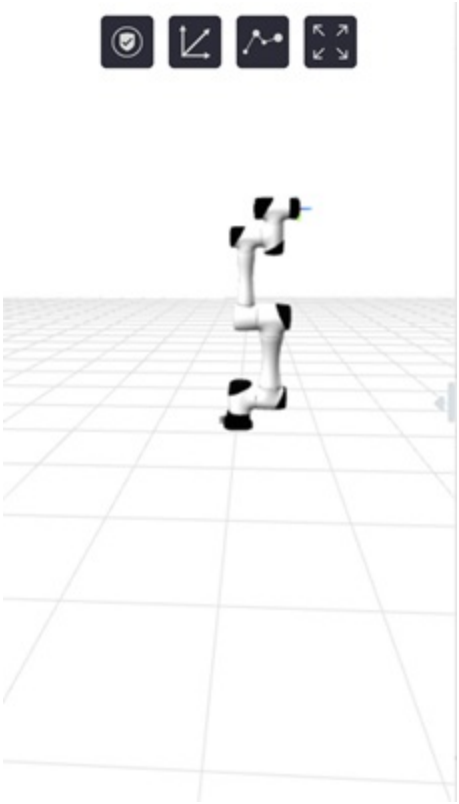
如需重新使能机械臂，请先再次单击急停按钮将其复位，再依次清除报警，上电和使能。

> **说明**
>
> - 如果是实体急停按钮被按下，软件上急停按钮的图标不会变，清除报警前需要先复位实体急停按钮（一般是通过顺时针旋转按钮复位）。
> - 远程控制模式下不会显示此界面。

3.3 机器人仿真面板

机器人仿真面板位于主界面左侧，用于显示机器人当前姿态，按住面板右边缘向左拖动可隐藏该面板。

> 📖**说明** 该面板也可在图形编程界面中使用，默认隐藏，当屏幕左边缘显示时可按住并向右拖动可显示该面板。



按住面板中央区域拖动可改变观察视角，使用双指手势可放大/缩小。

面板顶部有四个功能按钮，功能如下：

 : 显示/隐藏机器人工作区域。

 : 显示/隐藏参考平面坐标系。

 : 显示/隐藏机器人近十秒的运动轨迹。

 : 扩大/缩小仿真面板显示区域。

3.4 主要功能面板

主要功能面板位于主界面中央，显示CR Studio各个主要功能的入口。

图形编程：单击可打开图形编程界面，详见图形编程。

脚本编程：单击可打开脚本编程界面，详见脚本编程。

设置：单击可打开设置界面，详见设置。

工艺：单击可打开工艺界面，详见工艺。

3.5 点动板

点动板位于主界面右侧，用于控制机器人进行点动，按住面板向右拖动可隐藏该面板。

> 📖**说明** 该面板也可在其他界面中使用，默认隐藏，当屏幕右边缘显示 时按住并向左拖动可显示该面板。

1 回到自定义位姿按钮

长按“回到自定义位姿”按钮可使机器人运动至自定义位姿。自定义位姿支持设置。

2 坐标系和步进值选择

可分别选择工具坐标系，用户坐标系和步进值。

1. **工具坐标系**：请参考工具坐标系设置。
2. **用户坐标系**：请参考用户坐标系设置。

3. **步进值**：单次点动的位移值。
    - “持续”表示用户按住点动按钮时机器人持续运动，松开时停止运动。
    - 选择具体的数值（如0.1）则表示单击点动按钮时机器人会位移这个值，然后停止运动。

      在笛卡尔坐标系下，该值的单位是mm，0.1表示每次单击位移0.1mm。

      在关节坐标系下，该值的单位是°，0.1表示每次单击位移0.1°。

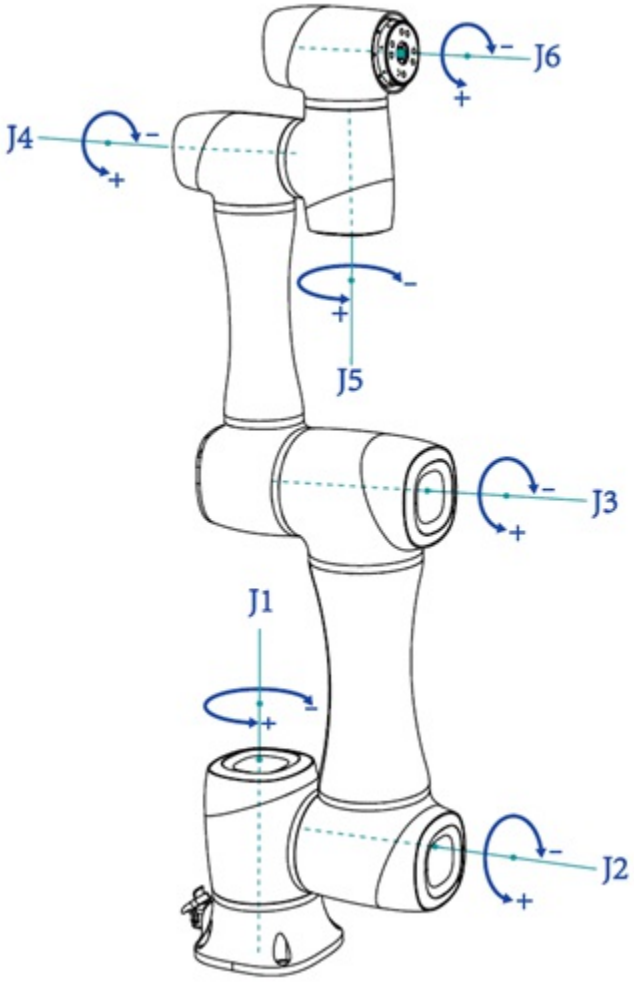
3 点动操作板

CR Studio支持基于三种坐标系进行点动。单击或长按各个坐标轴的"+"或者"-"即可进行点动。

点动板下方的R/D/N/Cfg表示机器人各臂朝向，详细说明可点击其右侧的 查看。

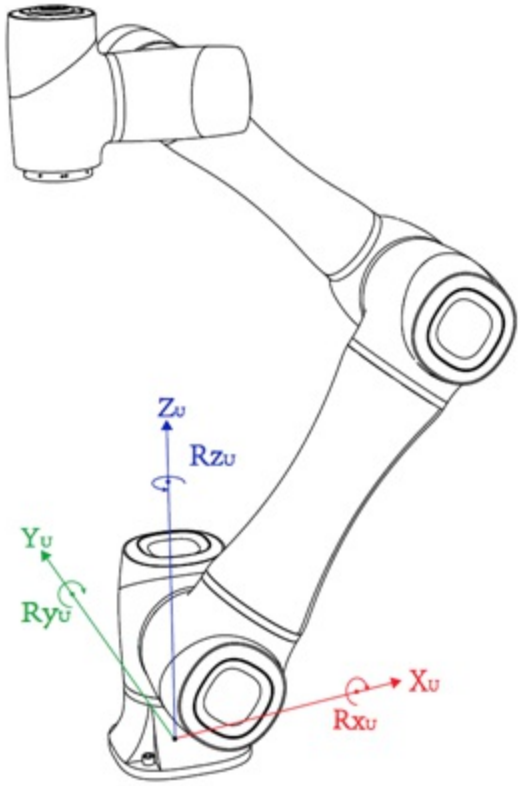
关节坐标系

关节坐标系是以各运动关节为参照确定的坐标系。各关节均为旋转关节，如下图所示。



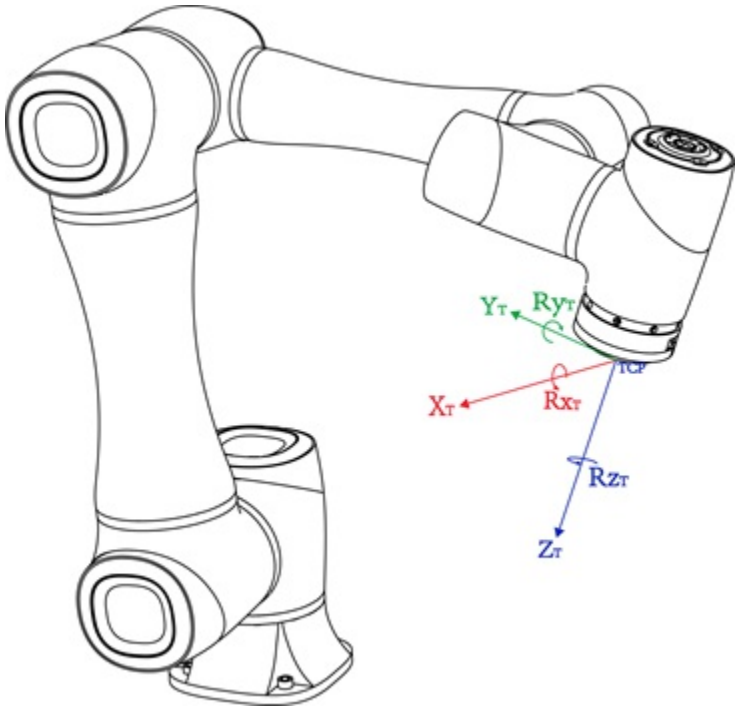
用户坐标系

用户坐标系是用户自定义的工作台坐标系或工件坐标系，其原点及各轴方向可根据实际需要确定。默认用户坐标系如下图所示，Y轴正方向为重载插座面对的方向。



工具坐标系

工具坐标系是定义工具中心点TCP（Tool Center Point）的位置和工具姿态的坐标系，其原点和方向都是随着末端工件位置与角度不断变化的。默认工具坐标系如下图所示，Y轴正方向为航空插座的反对方向。



4 设置

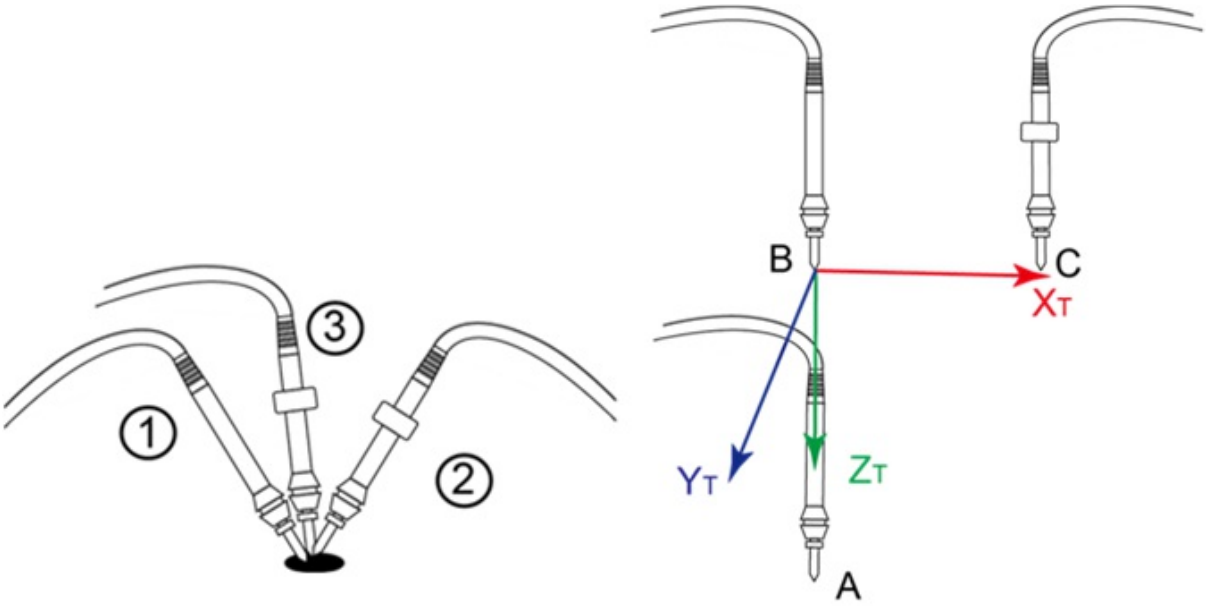
4.1 工具坐标系设置

当机械臂末端安装了工具（如焊枪、喷嘴、夹具等）后，为了编程和机械臂运行的需要，此时需要设置工具坐标系。例如，利用多个夹具同时搬运多个工件，可将每个夹具设置为独立的工具坐标系以提高搬运效率。

当前系统支持最多10个工具坐标，工具坐标系**0**为默认工具坐标系，即不使用工具，不可更改。

> 📖**说明**
>
> 建立工具坐标系时，请确保参考坐标系为默认工具坐标系。

六轴工具坐标系采用三点示教法“TCP+ZX”生成：机械臂末端安装工具后，调整工具的位姿，使TCP（Tool Center Point）以三种不同的方向对准空间中同一点（参考点），获取工具位置偏移。再根据另外三点（**A**、**B**、**C**）获取工具姿态偏移。

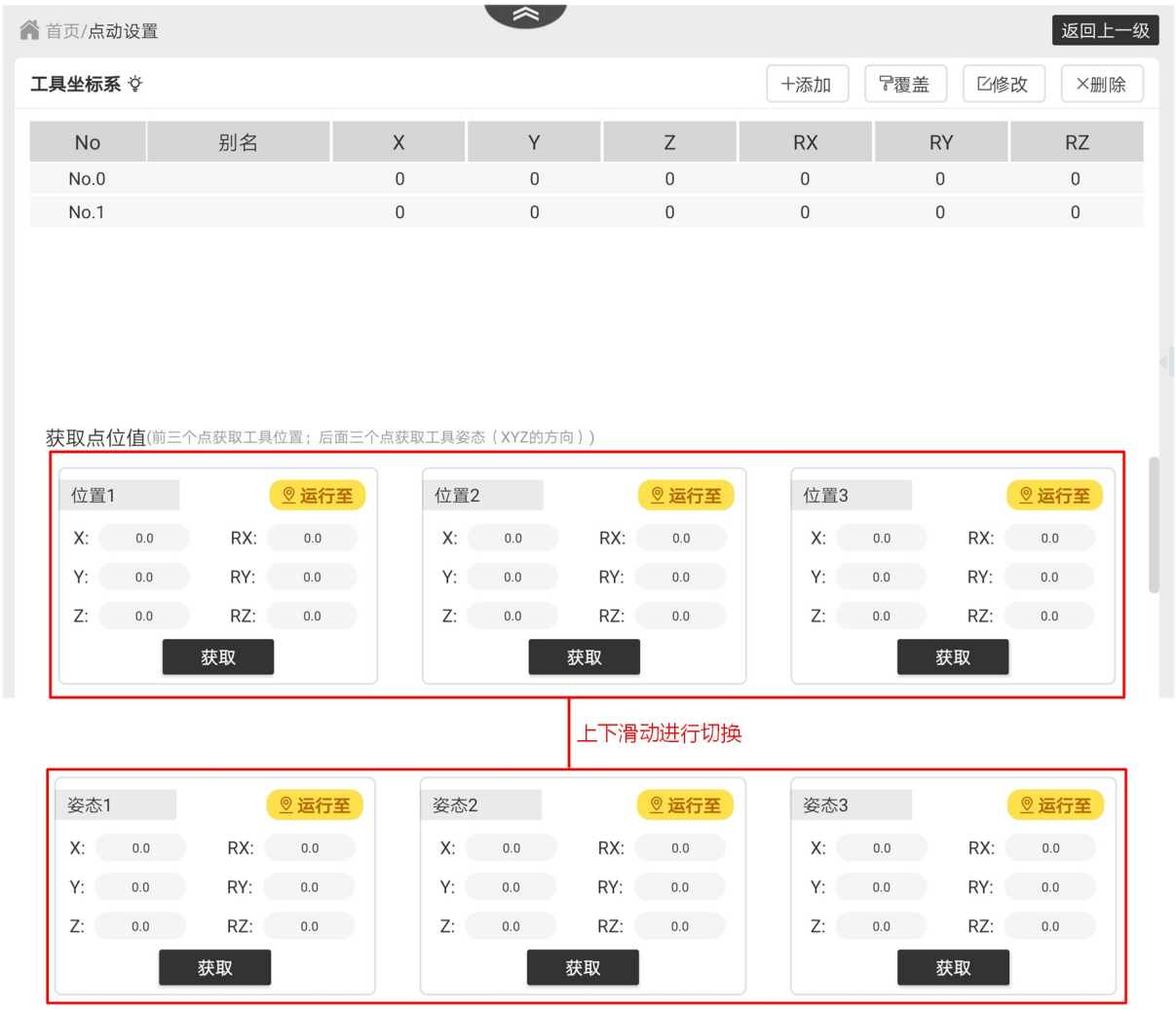


添加工具坐标系

前提条件

- 机械臂已上电。
- 机械臂处于使能状态。

操作步骤



说明

1. 单击界面左上的图标可查看本界面的使用教程，较文档更加直观，建议查看。
2. 若已添加了9个坐标系，“添加”按钮会消失，表示不可再添加新的坐标系。

1. 在机械臂末端安装工具，本节不做详细说明。
2. 点动机械臂，使TCP（Tool Center Point）以第一种姿态对准参考点（位姿①），在“位置1”区域单击“获取”获取第一个点的坐标。
3. 点动机械臂，使TCP以第二种姿态对准参考点（位姿②），在“位置2”区域单击“获取”获取第二个点的坐标。
4. 点动机械臂，使TCP以第三种姿态对准参考点（位姿③），在“位置3”区域单击“获取”获取第三个点的坐标。
5. 点动机械臂，使TCP以垂直姿态对准参考点，在“姿态1”区域单击“获取”获取第四个点的坐标（A点）。

6. 正向点动Z轴，使机械臂移动至另一点，在“姿态2”区域单击“获取”获取第五个点的坐标（B点）。

> 📖**说明**
>
> 该步骤确定Z轴正方向。

7. 正向点动X轴，使机械臂移动至C点（不能与A、B两点在同一直线上），在“姿态3”区域单击“获取”获取第六个点的坐标。

> 📖**说明**
>
> 该步骤确定X轴正方向，然后根据右手定则确定Y轴方向。

8. 单击“添加”，APP会计算得到新工具坐标系的参数并添加一条新的记录。

其他操作

除No.0外，单击对应工具坐标系的“No”列可以选中该坐标系，单击其他列可以修改该列对应参数的值。

运行至：长按“运行至”按钮可让机械臂运动至已获取的点位。

覆盖：获取到六点的坐标后，选中指定坐标系，单击“覆盖”按钮，可以使用新标定的坐标系覆盖该坐标系。

修改：选中指定坐标系，单击“修改”按钮，可将下方过程点位修改为选中的坐标系对应的点位。

删除：选中指定坐标系，单击“删除”按钮，可以删除该坐标系。

> 📖**说明**
>
> 删除坐标系只是清除该条记录的内容，并不会删除该条记录本身。用户需要重新标定该坐标系时，可使用覆盖操作。No.0坐标系不可删除。

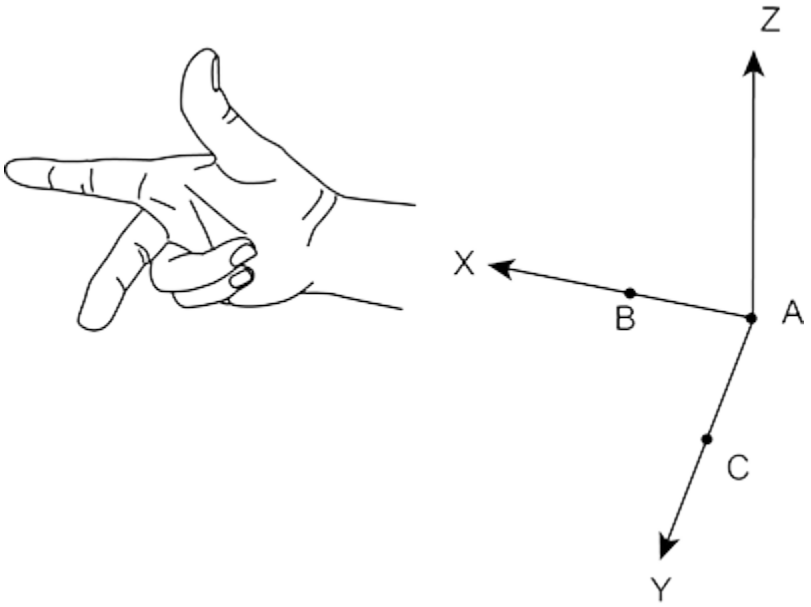
4.2 用户坐标系设置

当工件的位置发生变化或机械臂的运行程序需要在多个同类型的加工系统中重复使用时，此时需要设置用户坐标系，使所有路径都跟随用户坐标同步更新，极大的简化了示教编程。

当前系统支持最多10个用户坐标系，用户坐标系**0**为基坐标系，不可更改。

> 📖**说明**
>
> 建立用户坐标系时，请确保参考坐标系为基坐标系。

用户坐标系采用三点示教法生成：将机械臂移动至任意三点**A**，**B**和**C**。其中**A**点作为原点，**AB**两点之间的连线确定用户坐标系X轴正方向，C点沿X轴做垂线确定Y轴正方向（实际标定时，C点不一定要在Y轴上，落在X轴和Y轴正方向确定的平面象限内，且不在X轴上即可，系统可自动计算Y轴正方向），根据右手法则，确定Z轴方向。



添加用户坐标系

前提条件

- 机械臂已上电。
- 机械臂处于使能状态。

操作步骤

> **说明**
>
> 1. 单击界面左上的图标可查看本界面的使用教程，较文档更加直观，建议查看。
> 2. 若已添加了9个坐标系，“添加”按钮会消失，表示不可再添加新的坐标系。

1. 点动机械臂至任意一点（用户坐标系原点），在“第一个点”区域单击“获取”得到第一个点的坐标。
2. 点动机械臂至用户坐标系X轴上的第二个任意点，在“第二个点”区域单击“获取”得到第二个点的坐标。
3. 点动机械臂至用户坐标系XY轴正方向确定的平面象限上除X轴外的第三个任意点，在“第三个点”区域单击“获取”得到第三个点的坐标。
4. 单击“添加”，APP会计算得到新用户坐标系的参数并添加一条新的记录。

其他操作

除No.0外，单击对应用户坐标系的“No”列可以选中该坐标系，单击其他列可以修改该列对应参数的值。

运行至：长按“运行至”按钮可让机械臂运动至已获取的点位。

覆盖：获取到三点的坐标后，选中指定坐标系，单击“覆盖”按钮，可以使用新标定的坐标系覆盖该坐标系。

修改：选中指定坐标系，单击“修改”按钮，可将下方过程点位修改为选中的坐标系对应的点位。

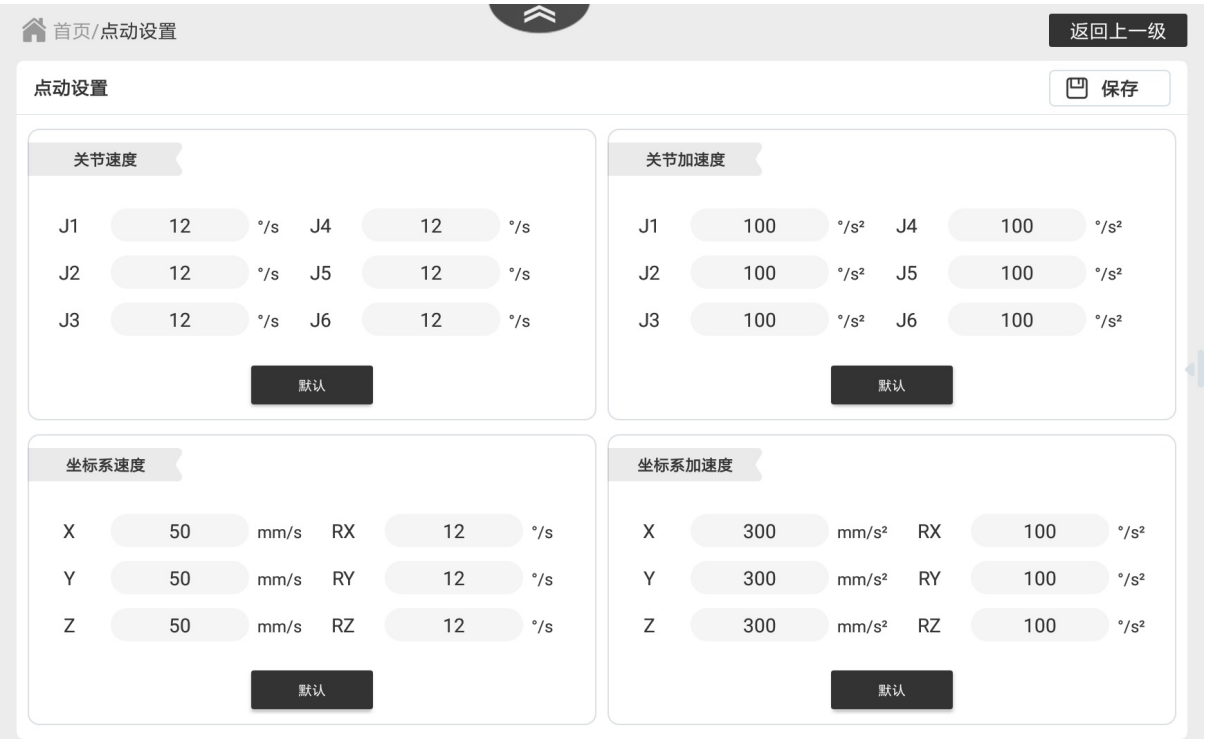
删除：选中指定坐标系，单击“删除”按钮，可以删除该坐标系。

> **📖说明**
>
> 删除坐标系只是清除该条记录的内容，并不会删除该条记录本身。用户需要重新标定该坐标系时，可使用覆盖操作。No.0坐标系不可删除。

4.3 点动设置

> 📖**说明**
>
> - 本功能需要管理员权限。管理员默认密码为000000。
> - 机械臂出厂的时候已经设置了最优的运动参数，如无特殊需求，不建议修改。如果工况要求更高的运动速度，建议小幅度更改，否则过高的运动速度可能会损害关节寿命并造成安全隐患。

请根据实际需要，设置点动时关节或坐标系下各轴的最大速度、最大加速度，设置完成后请单击“保存”。



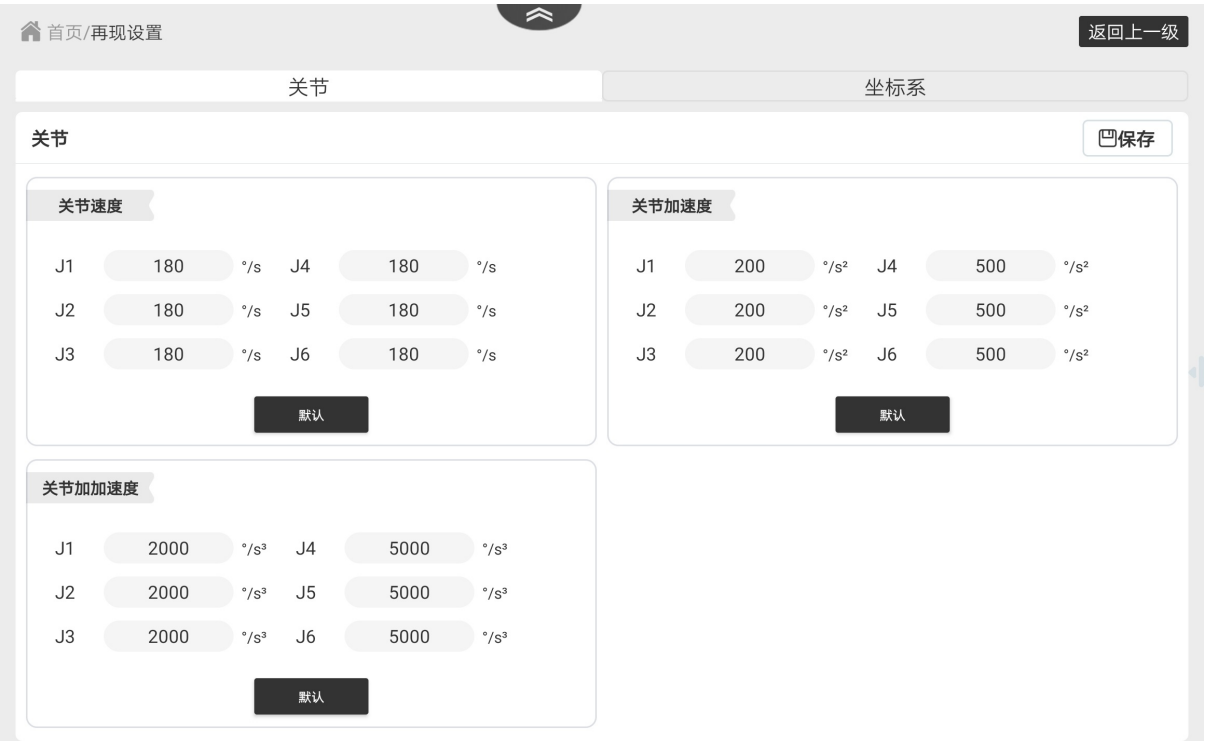
机器人实际运动速度/加速度 = 此处设置的速度/加速度 x 全局速率。

单击“默认”按钮会将对应版块的设置全部恢复为默认值。

4.4 再现设置

> **说明**
>
> - 本功能需要管理员权限。管理员默认密码为000000。
> - 机械臂出厂的时候已经设置了最优的运动参数，如无特殊需求，不建议修改。如果工况要求更高的运动速度，建议小幅度更改，否则过高的运动速度可能会损害关节寿命并造成安全隐患。

请根据实际需要，设置关节或坐标系下各轴的运动速度、加速度、加加速度，设置完成后请单击“保存”。



机器人实际运动速度/加速度 = 此处设置的速度/加速度 x 全局速率 x 编程时速度指令设置的百分比。

单击“默认”按钮会将对应版块的设置全部恢复为默认值。

4.5 安全设置

4.5.1 碰撞安全

碰撞安全功能主要用于减少碰撞力对机器人本体的影响，避免机器人本体或者外围设备损坏：当机械臂碰撞到工件或其他障碍物时，机械臂会自动停止运行，以防机器或操作人员碰撞受伤。

此页面不支持开关碰撞检测功能，如果碰撞等级最低时依然经常发生误报，希望关闭碰撞检测，请联系技术支持。

碰撞等级为安全等级设置，共有5个安全等级，5级最高。等级越高，机械臂碰撞检测停止所需的力越小。

碰撞后处理方式包括：

- 停止：机械臂停止运行工程。
- 暂停：机械臂暂停，需要根据实际情况选择解决碰撞原因后恢复工程运行，或者停止工程运行。
- 进入拖拽模式：机械臂停止运行工程并自动进入拖拽模式。

> ⚠**注意：**
>
> 如果负载设置不正确，可能导致机械臂运动时误触发碰撞检测并自动进入拖拽模式，运动速度较快时存在安全风险。

- 进入回退模式：机械臂根据碰撞前的轨迹自动回退指定距离，回退距离的设置范围为0~50，单位：mm。

机械臂点动中检测到停止所需的力后，会弹窗提示"检测碰撞停止"，此时您需要解决碰撞的原因，并单击“复位”。如果解决碰撞原因需要操作APP，可单击“一分钟后提醒我”暂时关闭弹窗（一分钟后会再次弹窗提示）。

修改设置后，单击“保存”按钮才会生效。

4.5.2 关节抱闸

机械臂处于下使能状态时，为防止关节运动，关节会自动抱闸使电机保持位置锁定，确保机械的运动部分不会因为自重或外力移动。

如果用户需进行关节拖拽操作，可开启抱闸，即在机械臂下使能后，手动扶住关节，然后单击想要移动的关节对应的开关。

> **注意**
>
> 当开启抱闸时，需手动扶住关节，避免机械臂因自重下砸。

4.5.3 末端负载

为充分发挥机械臂自身具备的性能，需将末端负载质量与偏心坐标设定在允许的范围内，确保机器运行时J6轴不会产生偏心。通过合理的设定，可优化机械臂的运动，抑制振动，缩短作业时间。

> **说明** 机械臂每次上使能时都会弹窗要求设置负载参数，设置的参数也会同步到此界面。

- 末端负载的偏心坐标需在J6轴为0°时设置。
- 末端负载重量即夹具末端重量与工件重量之和，不能超过机械臂允许的最大负载。负载重量设置为0时会以黄色底色和感叹号进行提示，且使能后使能图标右下也会出现黄色感叹号提示。

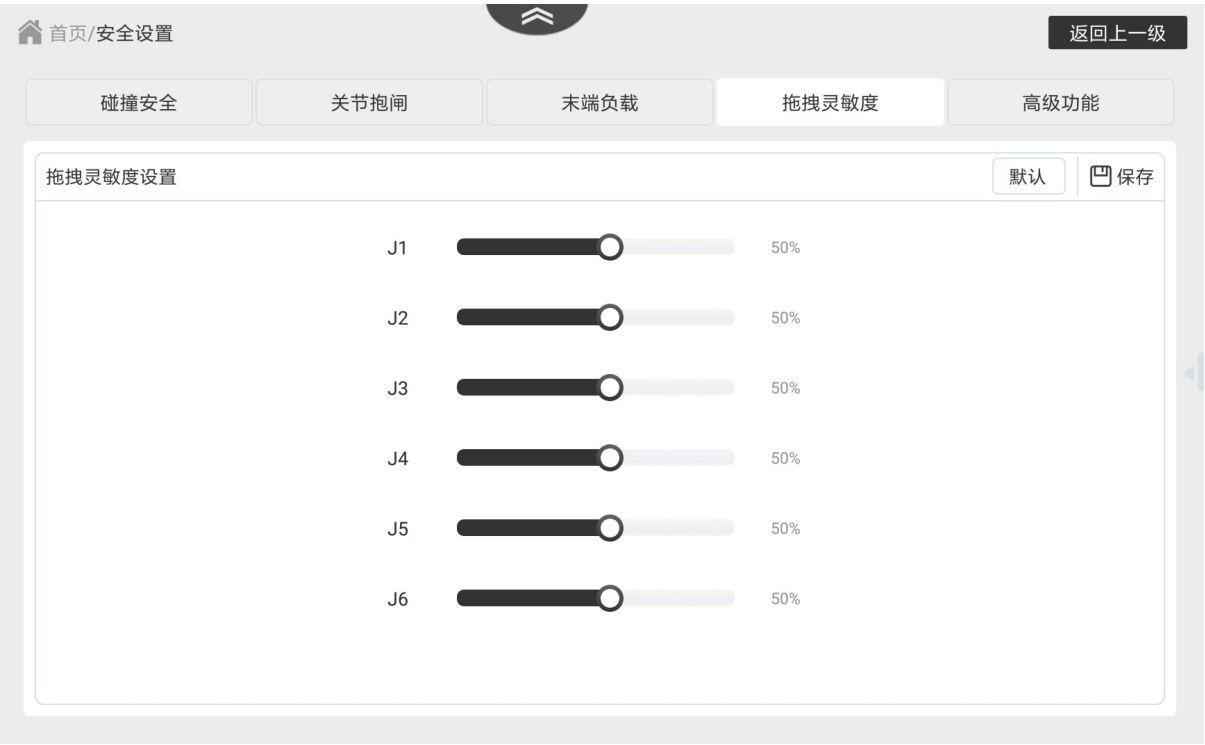
> **注意：**
>
> 负载设置不正确可能导致出现碰撞检测异常报警或者拖拽时本体不受控制。

修改设置后，单击“保存”按钮才会生效。

请务必设定合适的负载质量与偏心坐标，否则可能会导致发生错误或冲击，缩短机器使用寿命。

4.5.4 拖拽灵敏度

灵敏度设置主要用于调节机械臂各关节在拖拽操作中的灵敏程度。



灵敏度越低，拖拽过程中的阻力越大。

单击“默认”可以将所有关节的灵敏度恢复到默认值。

修改设置后，单击“保存”按钮才会生效。

4.5.5 高级功能

> 📖**说明** 本功能需要管理员权限。管理员默认密码为000000。

当因为现场情况需要对高级功能进行开关时，可在此界面进行配置。

如无特殊需求，建议保持默认值。

4.6 远程控制

外部设备可以通过不同的远程控制模式如远程I/O模式、远程Modbus模式下发指令控制机械臂（即对示教好的程序文件进行控制运行）。可通过APP设置远程控制模式。

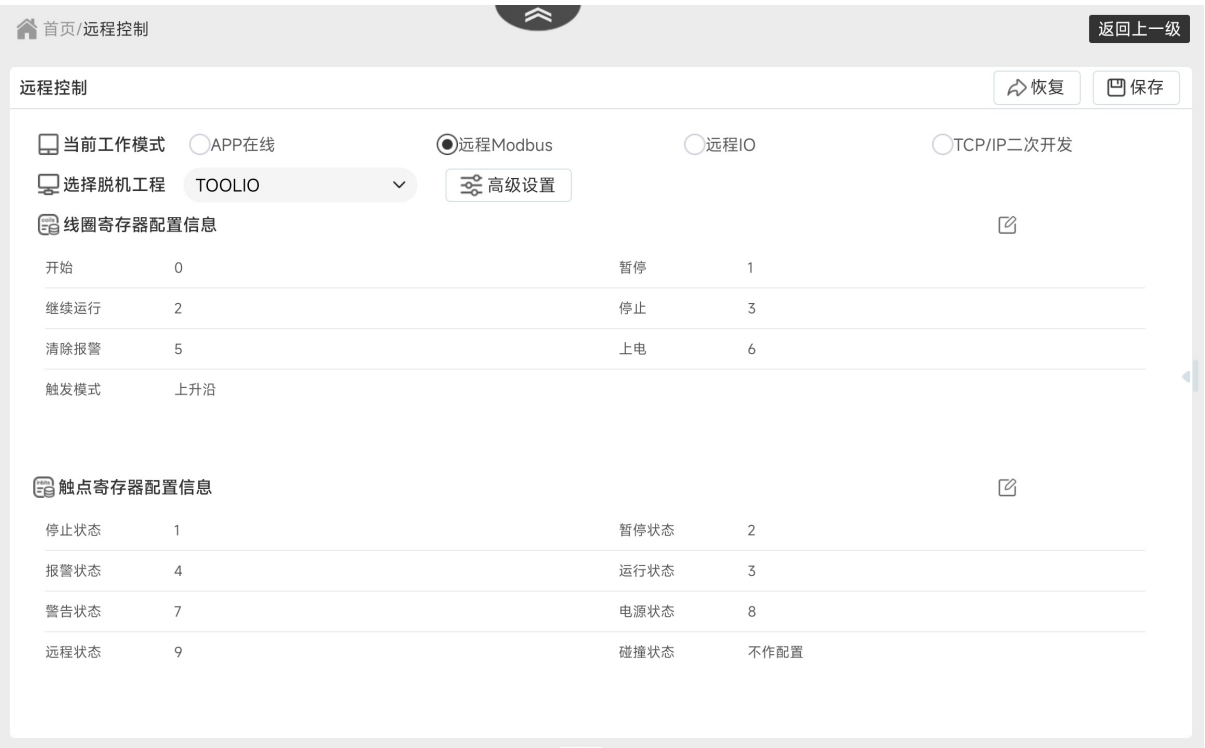
> **说明**
>
> - 远程控制模式切换不需重新启动机器人控制系统。
> - 无论机器人控制系统处于哪种模式，急停开关始终有效。
> - 若机械臂在远程控制模式下运行，此时切换为其他工作模式工程会自动停止。
> - 单击“恢复”按钮可以将界面自动切换到当前设置中的工作模式。

APP在线

默认的控制模式，使用APP对机械臂进行控制。

远程Modbus

当远程控制模式为远程Modbus时，外部设备可通过远程Modbus来控制机器人。



Modbus寄存器特定功能如上图所示，可单击进行编辑。

通过远程Modbus模式运行工程的步骤如下。

前提条件

- 已准备好远程运行的工程文件，且能正确运行。
- 已通过LAN接口将外部设备与机械臂连接。可直接连接或通过路由器连接，请根据实际情况选择。其中，机械臂和外部设备的IP地址需在同一网段。
- 机械臂已上电。

操作步骤

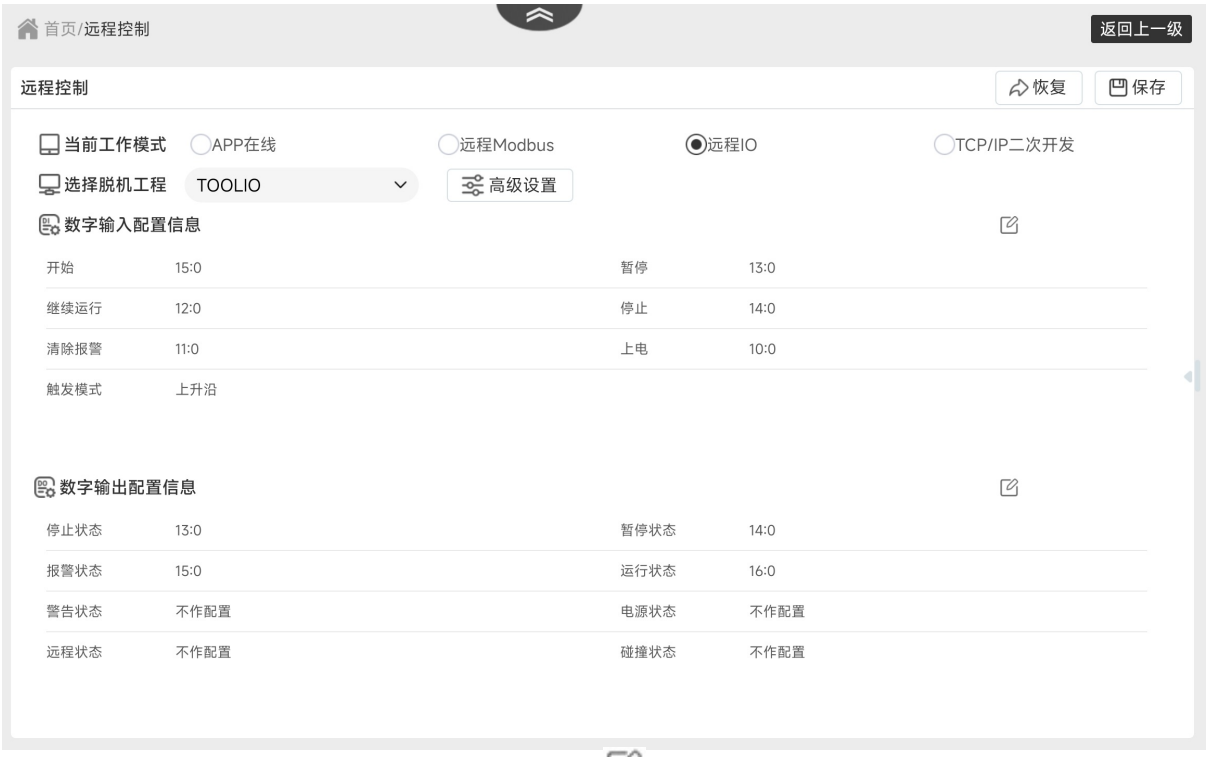
1. 在“远程控制”界面选中“远程Modbus”，并选择待远程运行的脱机工程。
2. 如果需要通过Modbus启动多个不同的工程，可点击高级设置。在高级设置中，可以设置备选工程的保存寄存器地址，以及配置备选工程列表，如下图所示。

3. 单击“保存”按钮，此时机械臂进入远程Modbus控制模式，不再接受APP急停以外的指令。同时，界面右上会出现“调试信息”按钮，单击可查看机械臂运行过程打印出来的调试信息。

4. 在外部设备上触发启动信号，此时机械臂会按照选择的工程文件进行运行。
5. 若触发停止信号，则停止运动。

远程IO

当远程控制模式为远程I/O时，外部设备可通过远程I/O来控制机械臂。



控制系统特定的I/O接口定义如上图所示，可单击进行编辑。控制柜类型为CCBOX（小型控制柜）时，安全IO与通用IO共用端子，已被配置为安全IO的端子不可再被配置为远程IO。

单击“高级设置”可以通过组I/O的方式配置多个备选工程。

1. 单击“+”或“-”可以增加或减少分配给组I/O的地址的数量，分配地址越多，可配置的备选工程越多。
    - 0个地址：不可配置备选工程。
    - 1个地址：可配置1个备选工程

    - 2个地址：可配置3个备选工程
    - 3个地址：可配置7个备选工程
    - 4个地址：可配置15个备选工程
2. 通过下拉框可修改分配的地址，分配給组I/O的地址不能与远程I/O或CCBOX的安全I/O重复。
3. 分配地址后，可针对各个组IO数值设置备选工程，至少设置一个。
    - 在远程I/O模式下，运行工程前，通过设置对应（黑底表示ON，白底表示OFF）的组IO数值，选择对应的备选工程。
    - 以上图中分配DI1~DI4四个地址为例：

      DI1为ON，DI2、DI3、DI4为OFF表示选择第1个备选工程；

      DI1和DI2位ON，DI3、DI4为OFF表示选择第3个备选工程；

      DI1~DI4全部为ON表示选择第15个备选工程。
    - 组IO全部设置为OFF时表示选择上一级页面配置的主工程。
4. 单击“保存”完成配置。

通过远程IO模式运行工程的步骤如下。

前提条件

- 已准备好远程运行的工程文件，且能正确运行。
- 已通过通用IO接口将外部设备与机械臂连接。
- 机械臂已上电。

操作步骤

1. 在“远程控制”界面选中“远程IO”，并选择待远程运行的脱机工程。
2. 单击“保存”按钮，此时机械臂进入远程IO控制模式，不再接受APP急停以外的指令。同时，界面右上会出现“调试信息”按钮，单击可查看机械臂运行过程打印出来的调试信息。

3. 在外部设备上触发启动信号，此时机械臂会按照选择的工程文件进行运行。
4. 若触发停止信号，则停止运动。

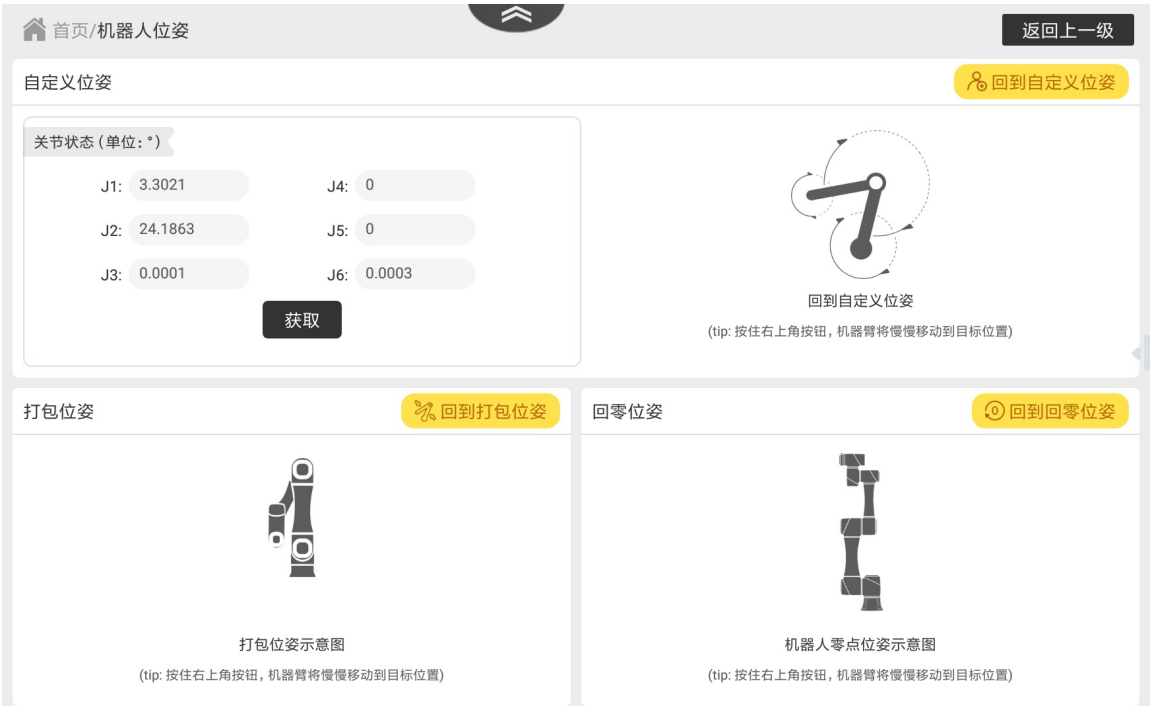
TCP/IP二次开发

此模式仅用于客户基于TCP自行开发控制软件的场合。如果您需要自行开发控制软件，请参考 Dobot TCP/IP协议（托管于Github）。

4.7 机器人位姿

CR Studio支持将机械臂运动到常用的位姿：打包位姿、回零位姿、自定义位姿。

- 打包姿态可缩小机器人占用空间，方便打包运送。
- 回零姿态即零点位置。
- 自定义姿态则根据用户需求自定义常用的姿态，方便快速移动到该位置。

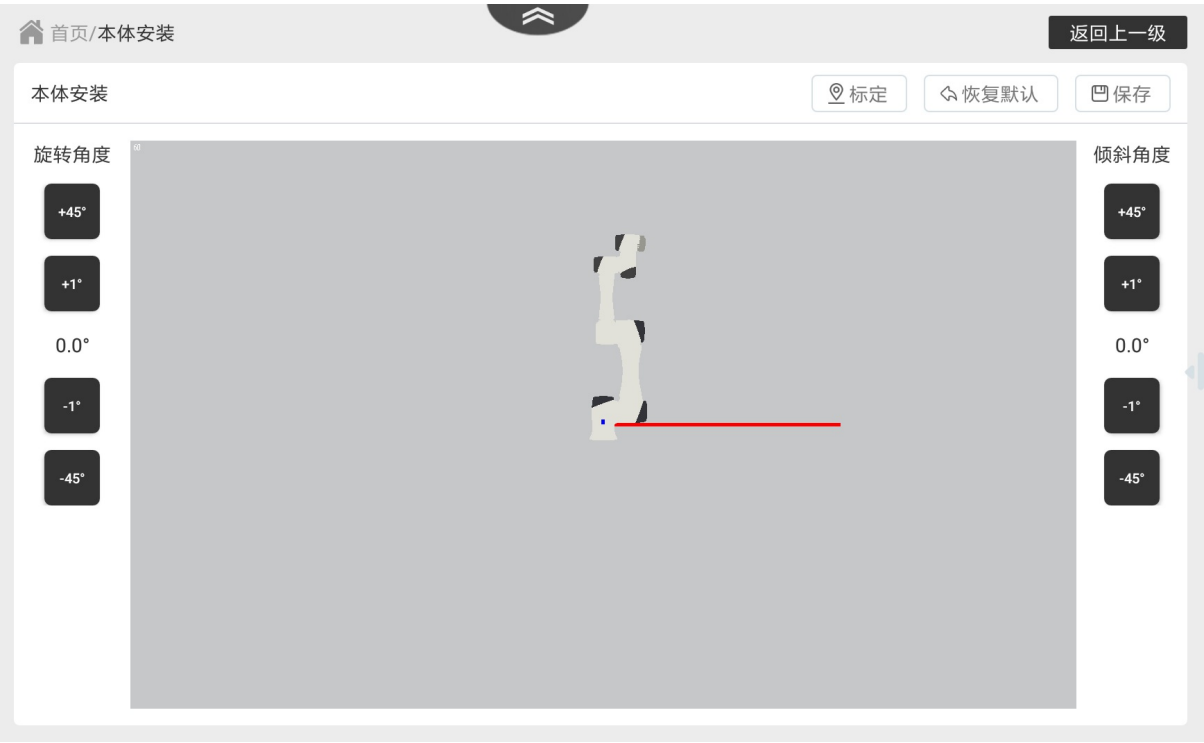


- 打包位姿：长按“回到打包位姿”，使机械臂移动至出厂位姿。
- 回零位姿：长按“回到回零位姿”，使机械臂移动至零点位姿。
- 自定义位姿：点动机械臂至自定义位置，单击“获取”，可确定自定义位姿。长按“回到自定义位姿”，使机械臂移动至用户自定义的姿态。

4.8 本体安装

> 📖**说明** 本功能需要管理员权限。管理员默认密码为000000。

一般情况下机械臂安装在平稳的台面或者地面上，此种情况无需进行任何操作。如果机械臂采用吊顶式、壁挂式安装或者呈一定角度安装，则需要在下使能状态下设置旋转角度和倾斜角度。



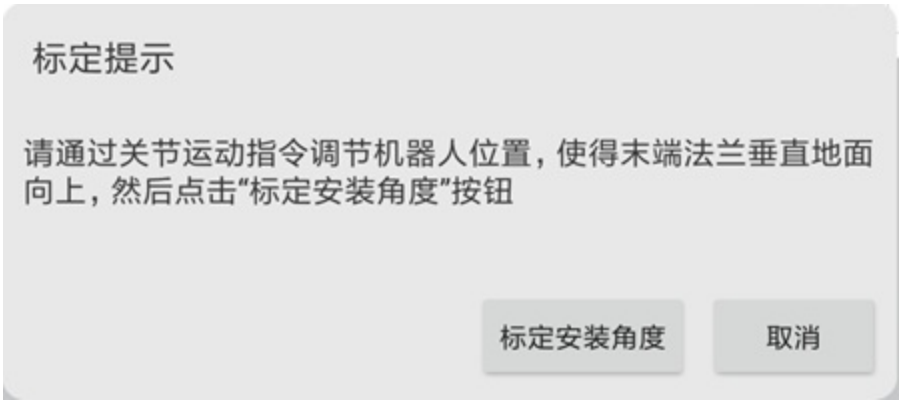
手动标定

用户根据实际的安装角度，通过页面左右的按钮调整中间模型的角度。

- 倾斜角度是指本体在原点位置绕X轴逆时针旋转的角度。
- 旋转角度是指本体在原点位置绕Z轴逆时针旋转的角度。

自动标定

在机械臂安装并使能后，单击“标定”按钮，并按照弹出的对话框进行操作，以获取倾斜角度和旋转角度。



标定完成后，单击“保存”，设置才会生效。

单击”恢复默认“则会将已标定的角度恢复为默认值。

4.9 软件设置

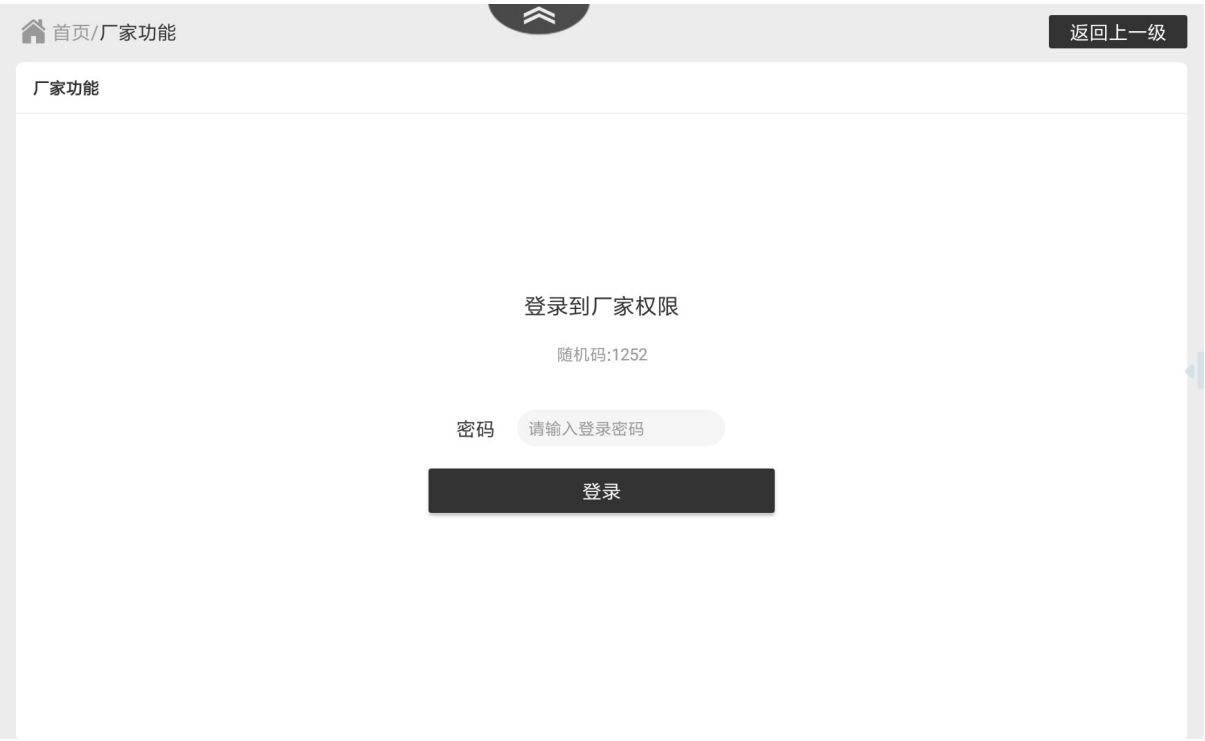
- 锁屏设置：用户可根据需要设置APP锁屏时间以及修改锁屏密码。若设置的时间段内未操作软件，系统将自动锁屏。锁屏后可通过管理员密码或锁屏密码解锁，默认为：000000。
- 修改Wi-Fi：机器人可通过WiFi与外部设备通信。用户可在“修改 Wi-Fi”页面修改WiFi名称和密码，重新启动控制柜后生效。其中，WiFi初始密码为1234567890。
- IP设置：机器人可通过LAN接口与外部设备通信，支持TCP、UDP或Modbus协议。用户可修改机器人的IP地址，子网掩码和网关。连接外部设备时，IP地址必须与外部设备的IP地址在同一网段，且不冲突。

  控制柜为大型控制柜（CC162）时，只有一个LAN接口，修改的是该接口的地址。控制柜为小型控制柜（CCBOX）时，有两个LAN接口，修改的是LAN1接口的地址。两种控制柜LAN1接口的默认IP地址都为192.168.5.1。

  - 如果机械臂与外部设备直接连接或通过交换机连接，则需勾选“手动获取”，修改IP地址、子网掩码以及默认网关后单击“确认修改”。
- 如果机械臂与外部设备通过路由器，则勾选“自动获取”（由路由器自动分配IP地址），然后单击“确认修改”。
- 用户管理：用户可在“ 用户管理”页面修改管理员的用户密码。管理员密码默认为：000000。

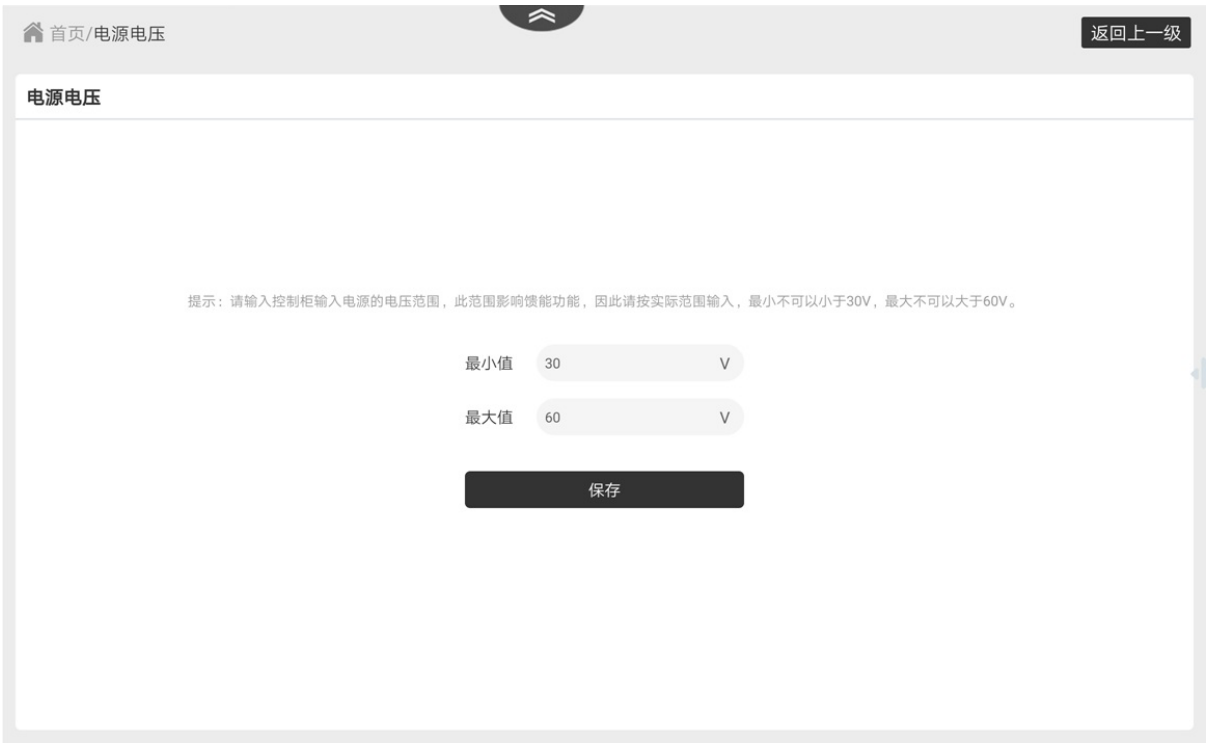
4.10 厂家功能

厂家功能一般由Dobot技术支持或工厂人员使用，本节不做详细说明。



4.11 电源电压（CCBOX）

控制柜类型为CCBOX（小型控制柜）时，需要设置控制柜的输入电源电压范围。



请按输入电压实际范围进行设置，最小值不小于30V，最大值不大于60V。

5 监控

5.1 I/O监控

1 控制柜和末端IO

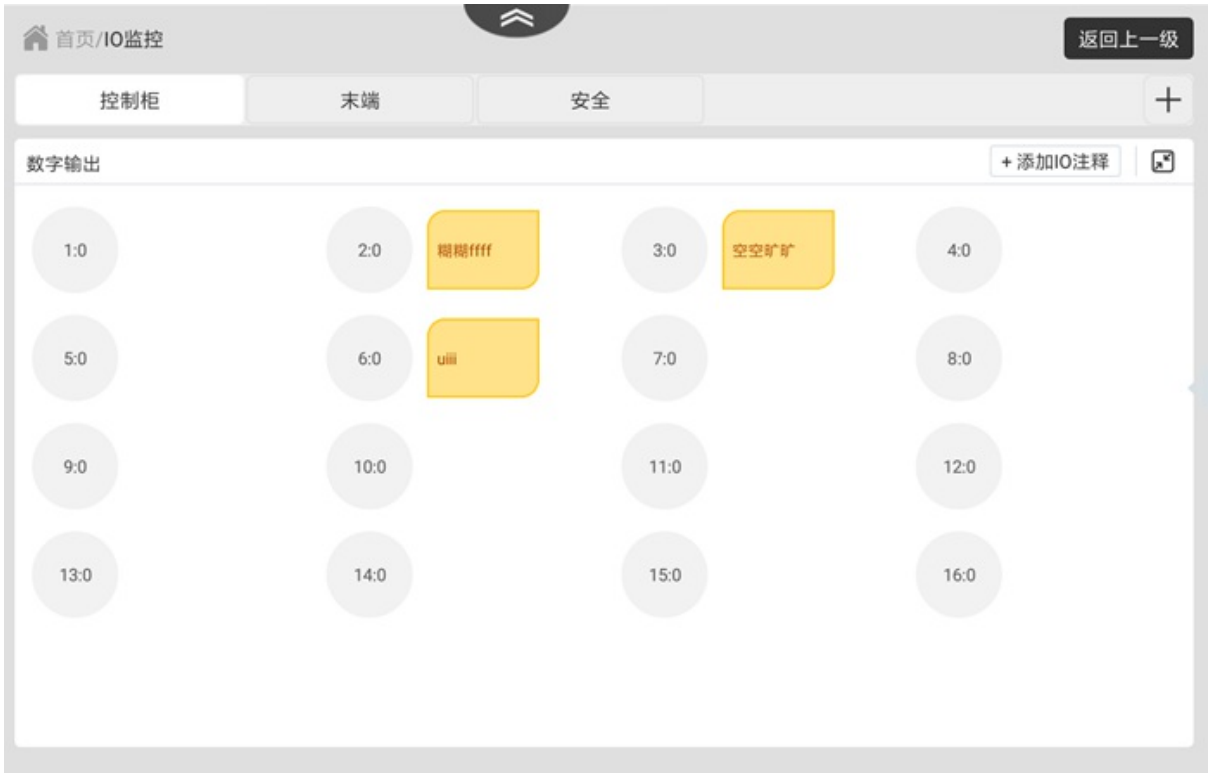
APP可以监控和设置控制柜和本体末端各个I/O状态。各个IO的具体定义请参考对应型号机械臂硬件手册中的IO接口说明，下文主要以CR系列机械臂为例进行介绍。

I/O监控具有三个功能：输出、监控以及模拟。

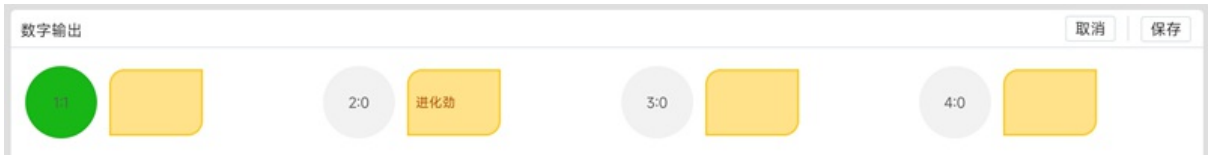
- 输出：设置数字输出或模拟输出的值（模拟量输出设置后需保存才会生效）。单击数字输出模块的“重置”按钮可重置所有修改过的输出值。
- 监控：查看输入、输出的真实状态。灰色表示DI未触发，绿色表示DI已触发。
- 模拟：模拟数字输入状态，方便调试运行程序。单击要模拟的数字输入后弹出设置窗口，如下图所示；选择虚拟并勾选“DI输入”，可以模拟该DI输入的触发状态。

控制柜类型为CCBOX（小型控制柜）时，可以设置数字输入/输出的类型为PNP（高电平有效）或者NPN（低电平有效），如下图所示。

单击数字输出或数字输入模块右上角的“编辑”按钮可扩大显示该IO模块，以数字输出为例，如下图所示。

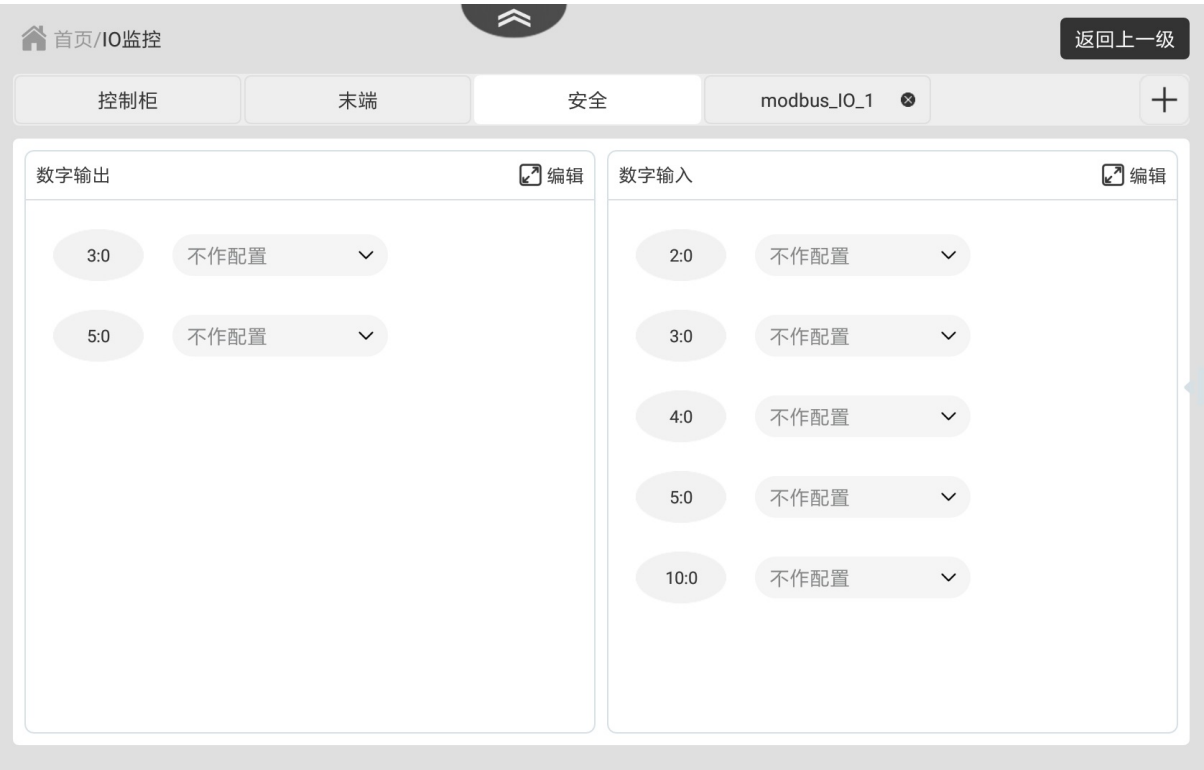


单击“添加IO注释”可以进入设置注释界面。单击对应端口右侧的注释框即可添加/修改注释，设置后单击“保存”。



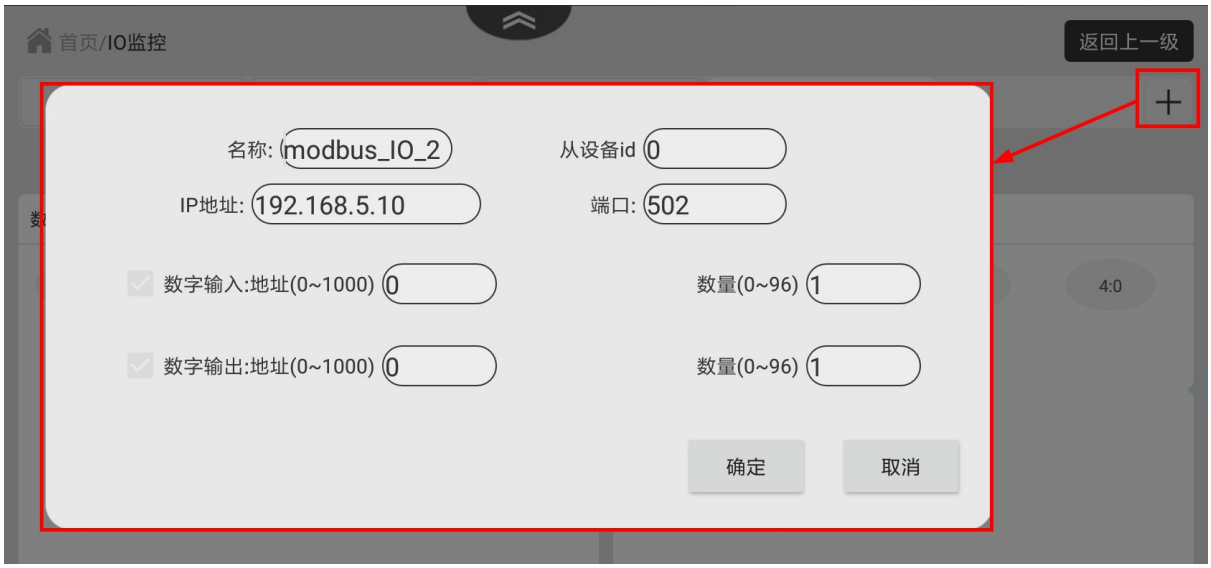
2 安全IO

安全界面用于设置各个安全IO接口的功能，单击编辑可进行修改，编辑完成后需要保存。安全IO接口详情请参考对应型号机械臂硬件手册中的安全IO接口说明。控制柜类型为CCBOX（小型控制柜）时，安全IO与通用IO共用端子，已被配置为远程IO的端子（包括高级设置）不可再被配置为安全IO。



3 拓展IO

除默认页签外，用户可以通过单击 + 添加新的自定义页签，可用于监控Modbus通信的IO，如下图所示。



- 名称：页签的名称。
- 从设备ID：填写设备号。
- IP地址：输入Modbus设备的地址。

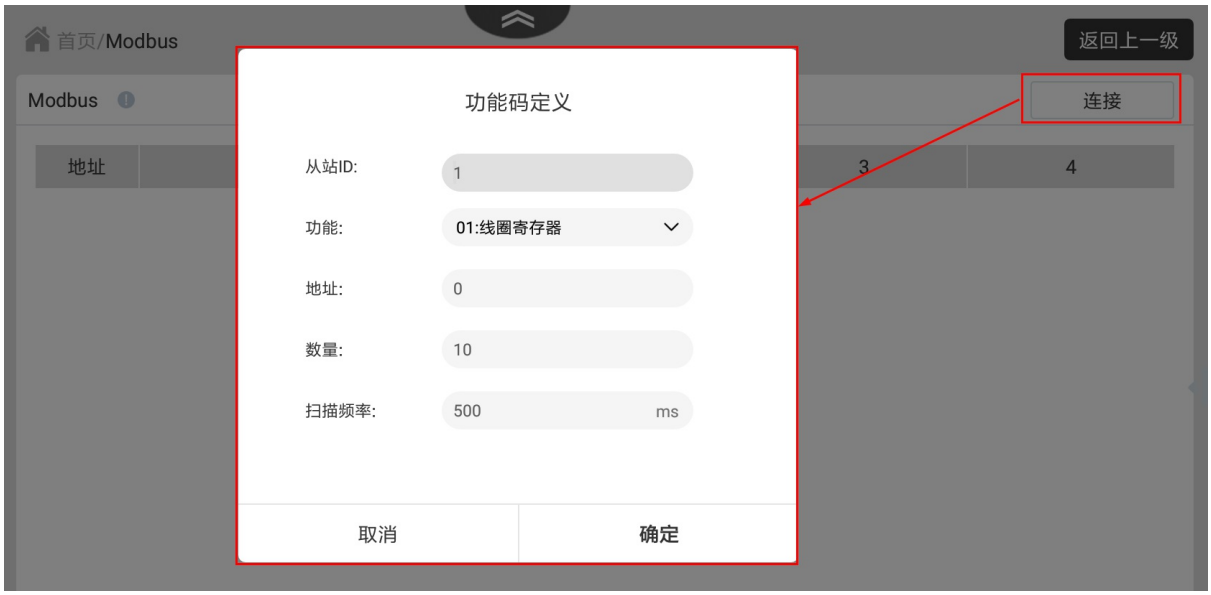
- 端口：填写Modbus通讯的端口号。
- 数字输入/数字输出：勾选后配置DI/DO的寄存器地址和数量。

单击确认后会在IO监控界面出现新的页签，但监控功能需要重启控制柜后才会生效。

单击自定义页签右侧的  可以删除该页签。

5.2 Modbus

该界面用于连接Modbus从站。



- 从站ID：设置Modbus从站的ID。
- 功能：设置从站的功能。支持线圈寄存器，离散输入，保持寄存器和输入寄存器。
- 地址/数量：寄存器的地址和数量。寄存器定义详见附录A Modbus寄存器定义。
- 扫描频率：机械臂扫描从站的时间间隔。

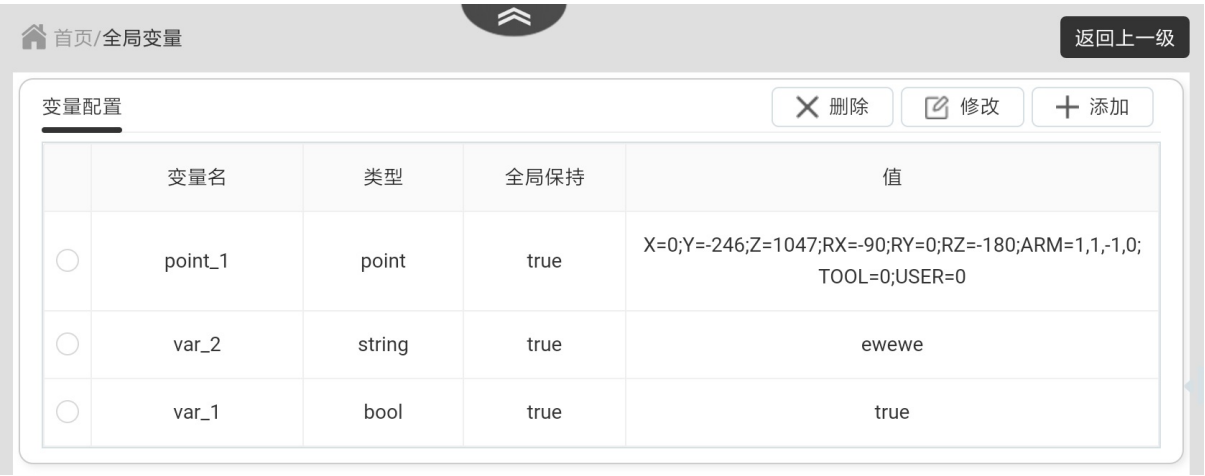
单击“确定”开始连接，如下图所示，单击“断开连接”可断开与从站的连接。

- 从站功能为线圈寄存器或保持寄存器时，单击表格内的单元格可以修改对应地址的值和别名。
- 从站功能为离散输入或输入寄存器时，单击表格内的单元格可以修改对应地址的别名。

5.3 全局变量

该界面用于配置与查看全局变量。

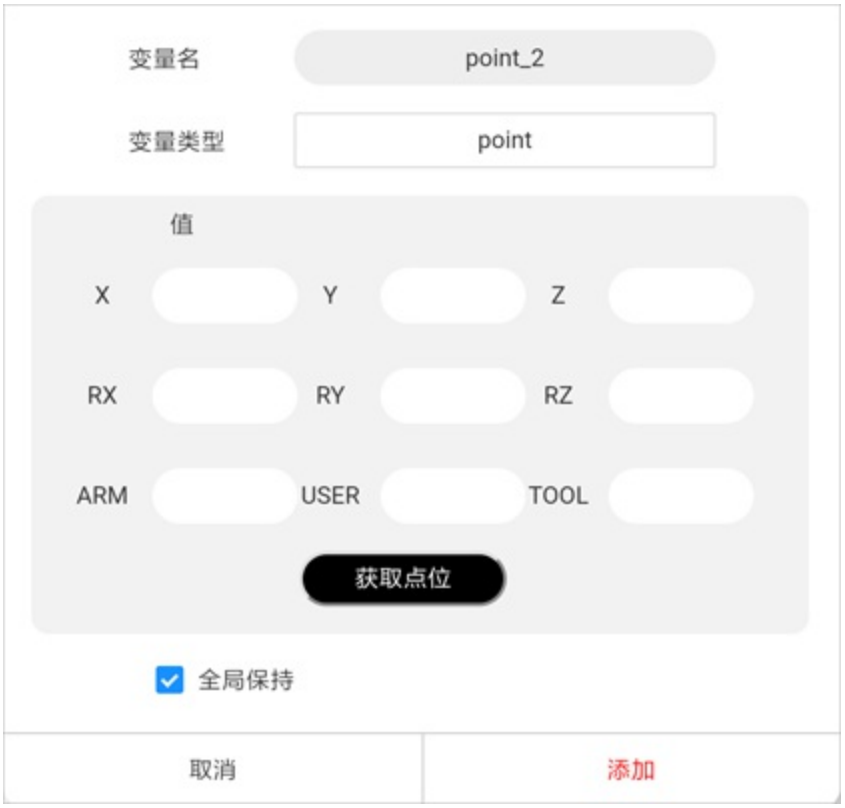
设置全局变量后，在图形编程中可使用全局变量相关的积木调用，在脚本编程中可直接通过变量名进行调用。



APP 支持以下类型的全局变量：

- bool：布尔值
- String：字符串
- int：整型数
- float：双精度浮点数
- point：机械臂点位，可通过将机械臂运动到指定点位后获取，如下图所示。

变量设置为全局保持时，在图形化或脚本编程中修改了全局变量的值后会同步到此处；否则修改的值仅在对应工程运行中生效，不会全局同步。


变量名
point_2
变量类型
point
值
X
Y
Z
RX
RY
RZ
ARM
USER
TOOL
获取点位
全局保持
取消
添加

5.4 机器状态

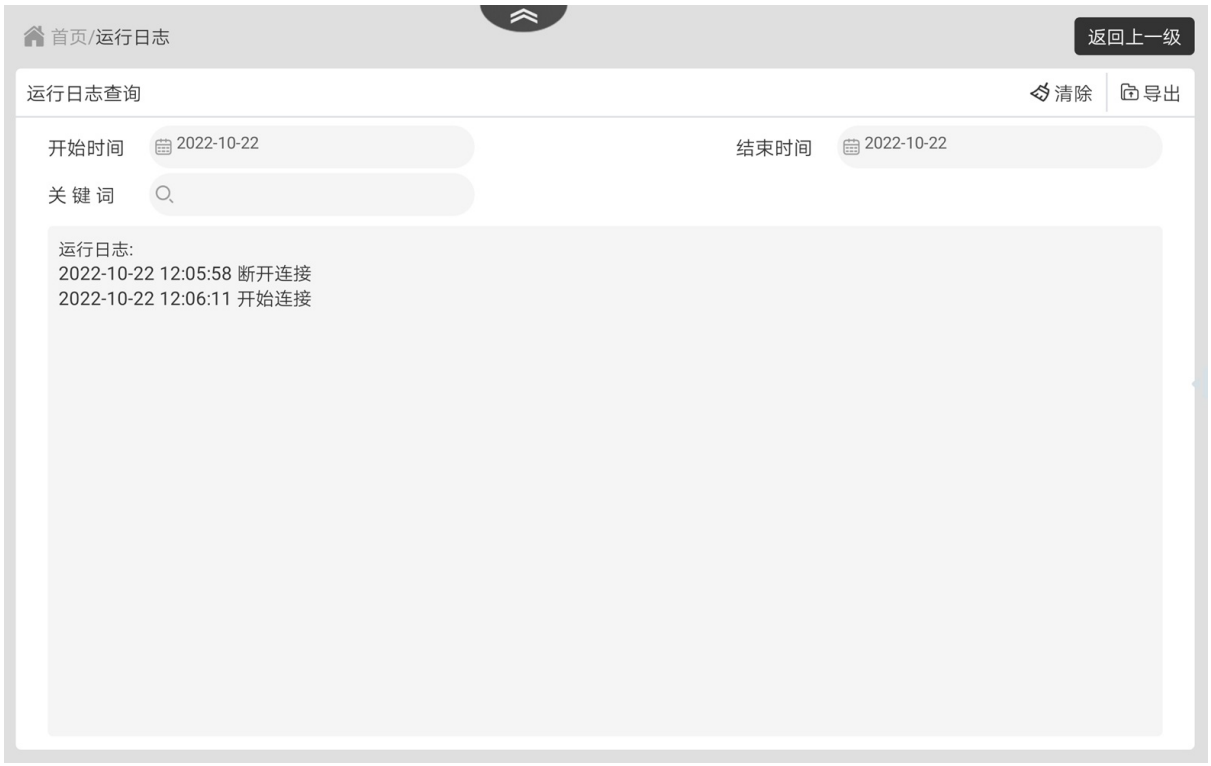
该界面可查看机器人的各类实时状态。当控制柜类型为CCBOX（小型控制柜）时，不显示平均功耗和机器人电流。

> **说明** 仅管理员权限可查看设备维护页面。管理员默认密码000000。


首页/机器状态
返回上一级
机器数据
关节状态
设备维护
关节1
电流 0.0A
电压 0.0V
温度 0℃
关节2
电流 0.0A
电压 0.0V
温度 0℃
关节3
电流 0.0A
电压 0.0V
温度 0℃
关节4
电流 0.0A
电压 0.0V
温度 0℃
关节5
电流 0.0A
电压 0.0V
温度 0℃
关节6
电流 0.0A
电压 0.0V
温度 0℃
200
160
120
80
40
0℃
首页/机器状态
返回上一级
机器数据
关节状态
设备维护
运行时间
清除
机器运行时间 0天13小时13分钟14秒
伺服运行时间 0天9小时25分钟4秒
控制器系统时间
修改
控制器系统时间 2022-08-02 14:25:46

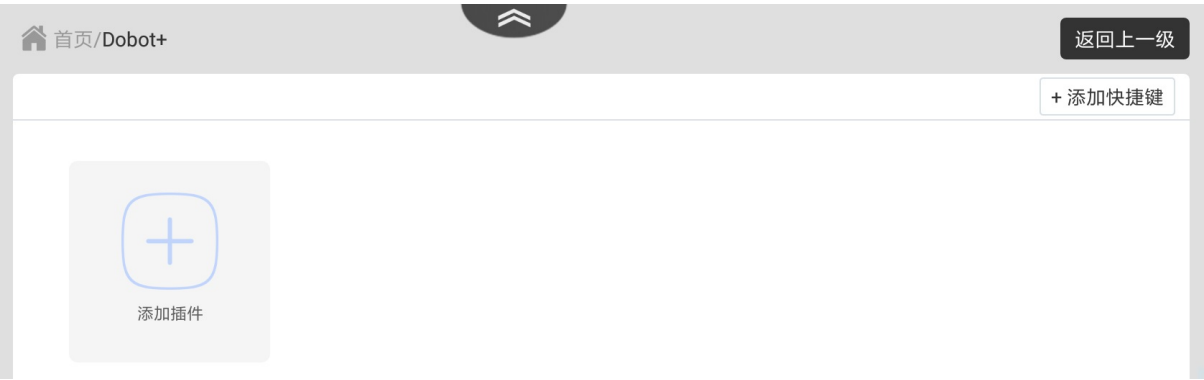
5.5 运行日志

用户可以通过查看日志了解机械臂历史操作。



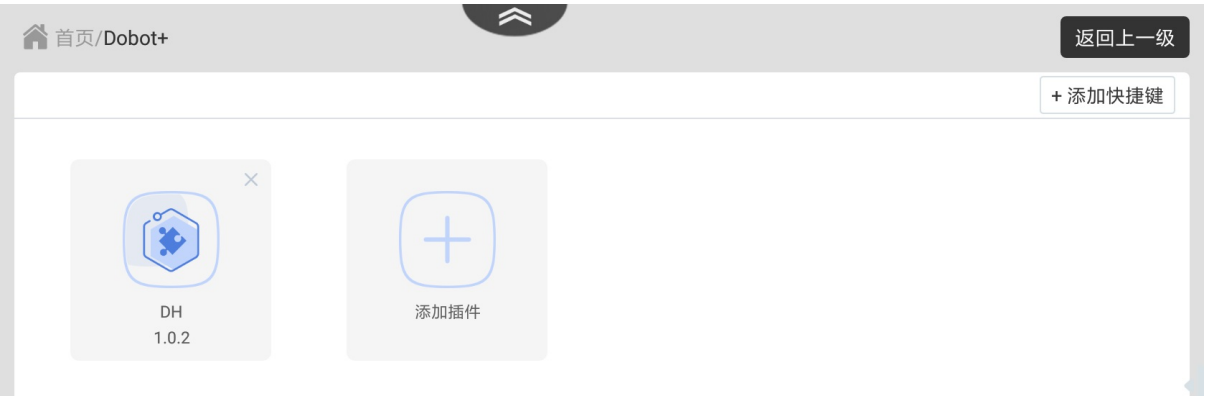
5.6 Dobot+

该页面用于安装与配置机械臂末端插件，插件可用于控制末端工具。



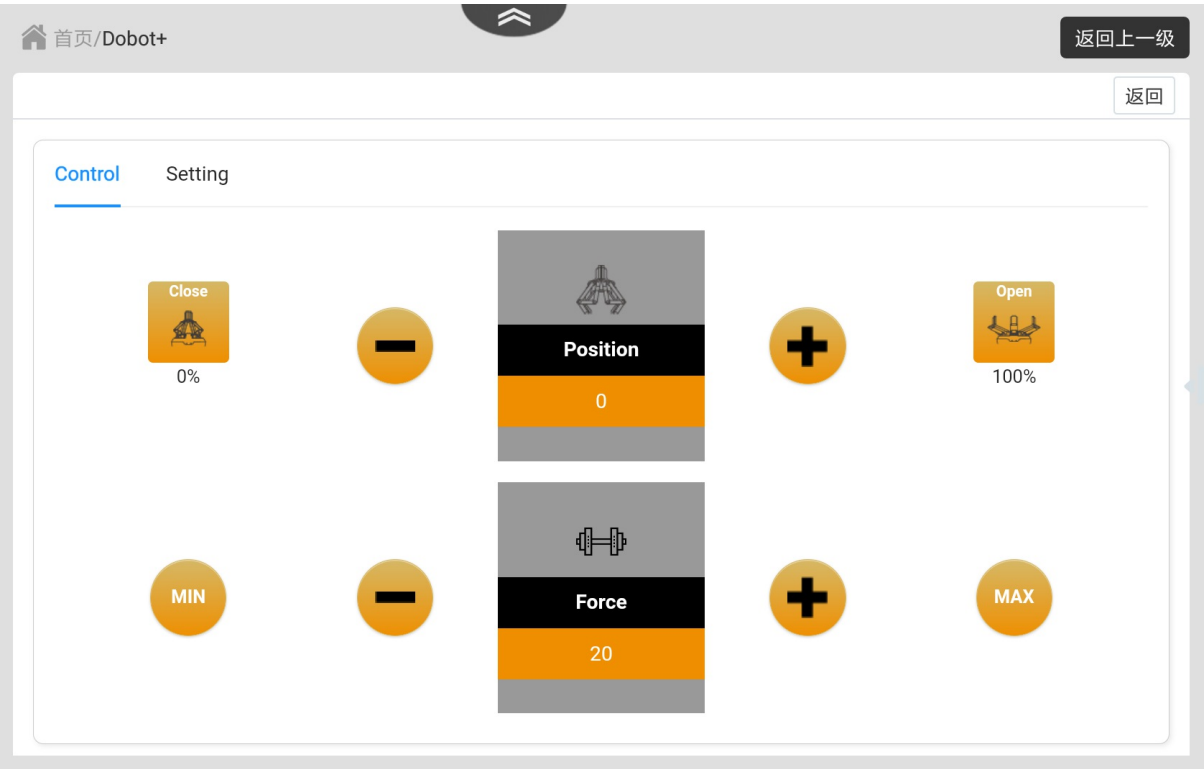
单击“添加插件”可以查看插件列表。单击插件左侧的 + 可以安装该插件，也可单击“导入”上传自定义插件。

插件安装完成后会显示在Dobot+主界面。



同时，图形编程和脚本编程中也都会新增插件相关的积木/指令。下图以DH夹爪插件为例。

单击插件图标可进行插件基本功能的配置。不同插件的配置方式不同，本文不进行详细介绍。下图以DH夹爪插件为例。



单击界面右上角的“添加快捷键”可以为插件设置快捷键，保存后可通过机械臂末端按键控制末端工具。

例如，选择DH夹爪的插件，设置快捷键1为夹爪张开操作，快捷键2为夹爪闭合操作，并保存。之后，按下图示的末端按钮夹爪会张开，再次按下末端按钮夹爪会闭合，此后循环。

6 编程

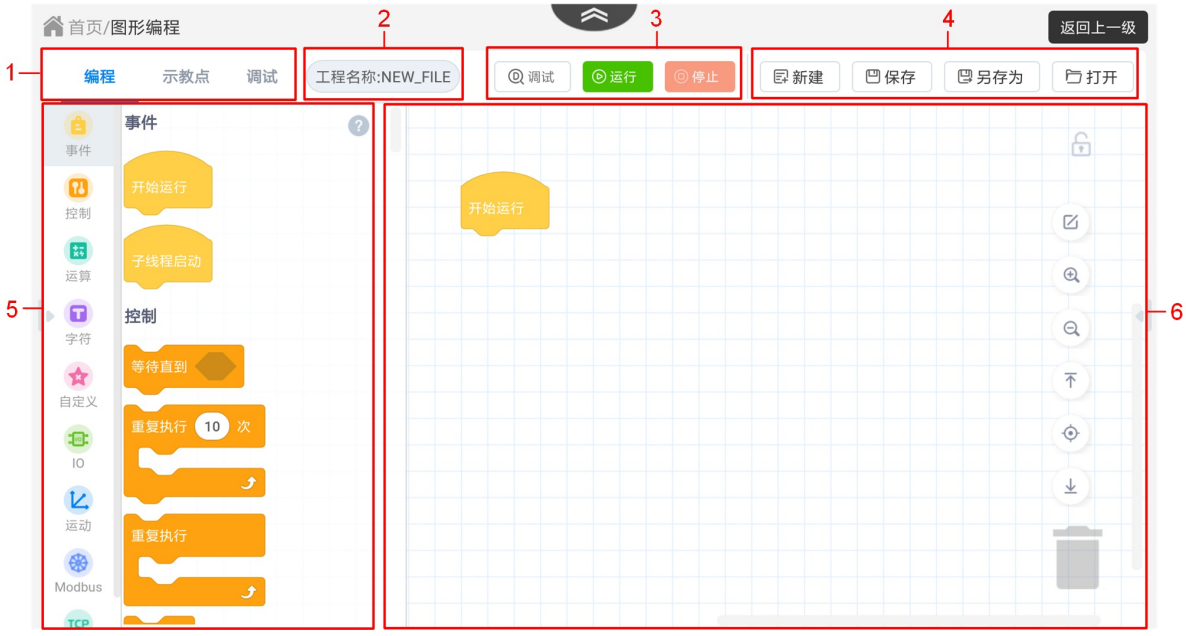
6.1 图形编程

CR Studio提供图形化编程的能力，用户可通过拖拽积木块的方式进行编程，从而控制机械臂的运行，直观易懂。

> **说明** 本章主要介绍如何使用图形编程界面，关于积木的具体说明请参考附录B。

编程界面

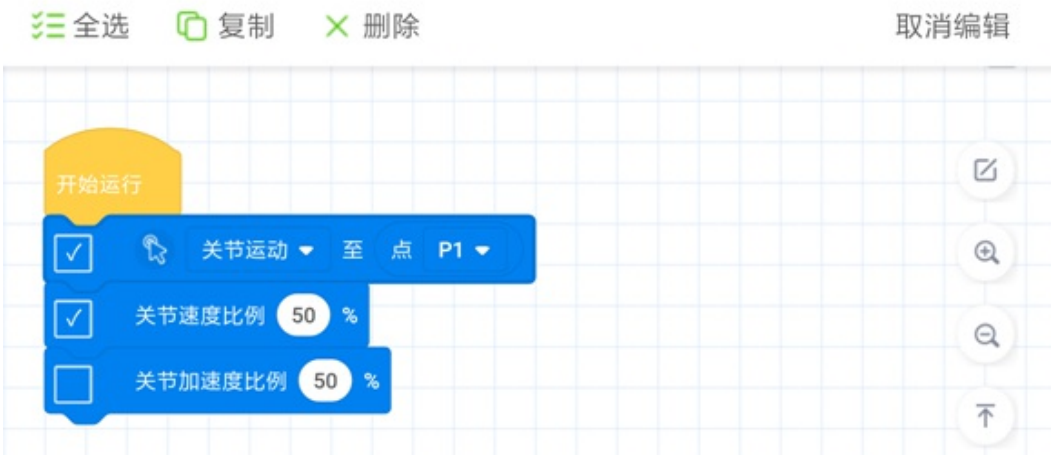
编程界面是图形编程功能的主界面，如下图所示。



序号	名称	说明
1	功能页签	用于切换不同图形编程功能界面。
2	工程名称	显示当前打开的工程的名称。
3	工程控制按钮	用于控制当前工程的调试（单步运行）、运行、暂停和停止。
4	工程管理按钮	用于管理工程文件。 在“打开”界面中，除了可以打开已保存的工程外，还支持以下功能。 • 将选中的图形编程工程转换为脚本编程的工程，转换成功后可在脚本编程界面打开转换后的工程。 • 将选中的工程文件导出到本地路径，以及从本地路径导入工程文件（包括APP自动备份的工程文件）。
		提供编程所需的积木，可以按照分类及颜色查找所需的积木。

		单击模块右上角的 可查看积木说明文档。
6	编程区	程序的编写区域，可以将积木拖放至该区域来编写程序。 在已放进该区域的积木上长按可打开菜单，支持复制积木、删除积木以及将不包含事件积木的一组积木变为子程序。 已修改且未保存的积木左端会出现 图标，提示用户该积木已修改。

编程区右侧的图标说明如下。

图标	说明
	用于进入编辑状态。 编辑状态下，用户可多选或全选积木进行复制或删除。 单击“取消编辑”或者在编程区进行其他操作都会退出编辑状态。 
	用于锁定/解锁编程区域。
	分别用于放大/缩小编程区域。
	分别用于回到积木的顶部/居中显示积木/回到积木的底部。
	将编程区中的积木拖到此处进行删除。用户也可长按积木选择删除。

示教点界面

示教点界面用于管理编程过程中要用到的机械臂点位，如下图所示。

序号	名称	说明
1	点位管理按钮	• 将机械臂运动到指定点位后，单击“添加”可以添加一条点位记录； • 勾选单个点位后单击“覆盖”则会用机械臂当前点位覆盖选中的点位； • 勾选点位后单击“删除”可以删除选中的点位； • 单击“保存”可以保存当前点位设置。
2	点位列表	显示已添加点位的信息，其中“工具”表示工具坐标系，“用户”表示用户坐标系。
3	“运行至”功能	勾选单个点位后，选择运动模式，再长按“运行至”按钮，机械臂会运行至勾选的点位。

调试界面

调试界面用于查看积木对应的脚本，工程运行日志，以及机器人运动的仿真模型，确认工程运行过程与结果是否符合预期，如下图所示。

- 主线程页签显示图形编程工程转换为脚本后的命令，运行后可查看当前运行的是哪行命令。

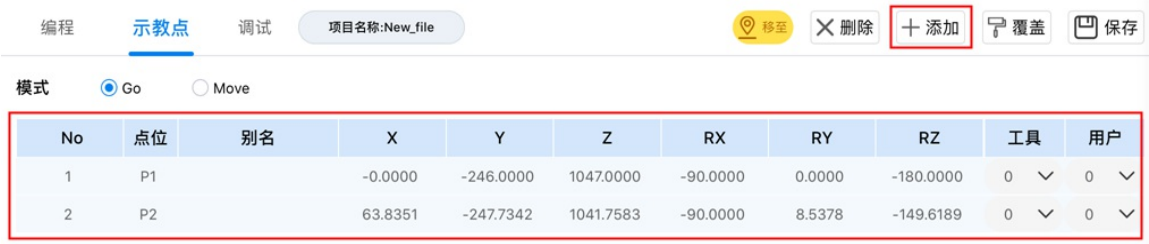
- 变量页签显示在图形编程中自定义的变量。
- 单击“调试”按钮后，该按钮变为“单步”按钮，单击可单步运行程序（单步是指单个积木，可能对应多行脚本）。

单击代码编写区域下方的 可显示控制台，用于输出运行信息，运行或调试过程中的错误信息也会在此输出。

操作步骤

下文以编写一个程序控制机械臂在两个点之间循环运动为例，介绍如何使用图形编程的主要功能。

1. 打开示教点页签，将机械臂运动至任意点P1后，单击"添加"保存点P1的位置信息。
2. 将机械臂运动至另一任意点P2后，单击"添加"保存点P2的位置信息。



3. 打开编程页签，拖动“重复执行”积木到“开始运行”积木的下方。
4. 拖动点运动积木到“重复执行”积木中，目标点选择P1。
5. 再拖动一个点运动积木到上一个点运动积木下，目标点选择P2。

6. 单击“保存”，自定义工程名称并单击“确定”。

7. 单击“运行”，机械臂开始运动。

6.2 脚本编程

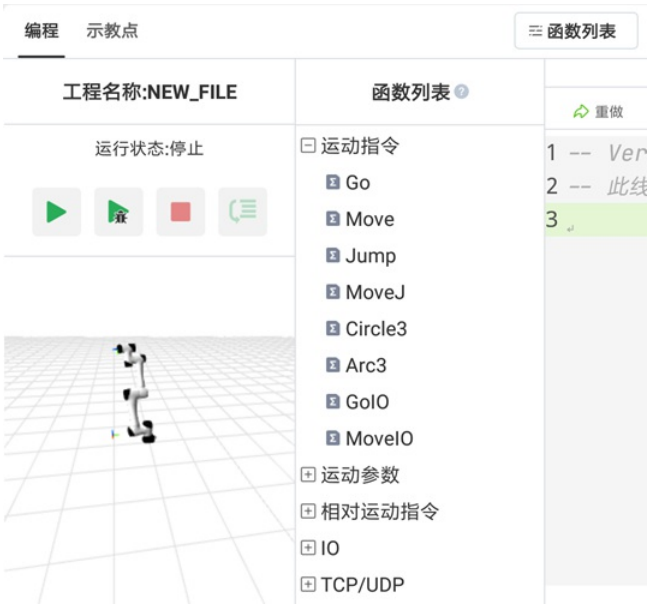
CR系列提供丰富的Dobot API接口，例如运动指令、TCP/UDP指令等，采用Lua语言，方便用户进行二次开发时调用。CR Studio提供了Lua脚本的编程环境，用户可以编写自己的Lua脚本控制机器人运行。

> **说明** 本手册仅介绍CR Studio脚本编程界面的使用方法，图形编程指南请参考附录C 。

编程界面

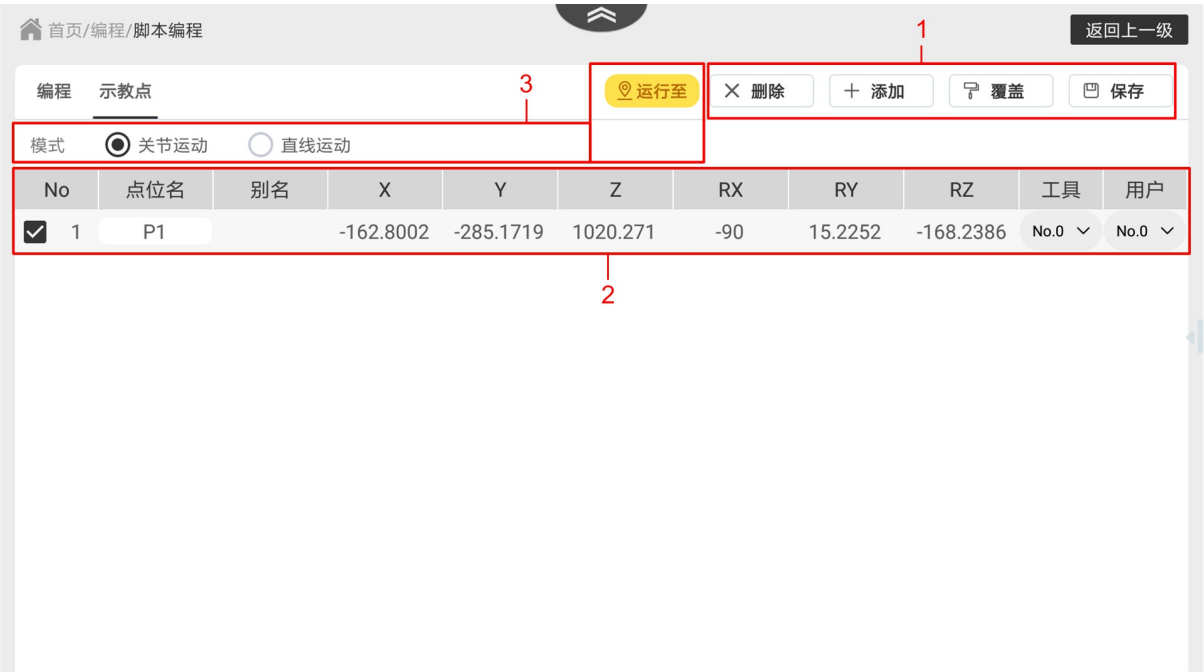
编程界面是脚本编程功能的主界面，如下图所示。

序号	名称	说明
1	功能页签	用于切换不同脚本编程功能界面。
		用于显示/隐藏函数列表。用户单击函数列表中的Dobot API后，可以通过配置参数的方式生成对应代码。单击列表标题右侧的 可查看API帮助文档。

2	函数列表显示开关	
3	工程管理按钮	用于管理工程文件。在“打开”界面中还支持将选中的工程文件导出到本地路径，以及从本地路径导入工程文件（包括APP自动备份的工程文件）。
4	工程名称	显示当前打开的工程的名称。
5	工程控制区	用于显示工程当前状态和控制工程运行。 ：单击开始运行工程。 ：单击进入调试模式。调试模式下该按钮会变为，此时可以断点运行，或者单步调试。 ：单击可以停止运行工程。 ：仅调试模式下可用，单击可以单步执行代码。
6	机器人仿真区	显示机器人实时仿真模型。
7	代码区	Lua代码编写区域。 • 主线程用于运行机械臂的主要控制代码。 • 变量页签用于定义该工程使用的变量。 • （仅安卓支持）单击代码行左侧的序号可以添加断点，调试模式下单击会开始断点运行，运行至该断点会自动暂停。 • 单击右上角的+可以添加子线程，最多可添加五个。子线程是与主线程并行运行的程序，例如I/O控制。 • 单击代码编写区域下方的可显示控制台，用于输出运行信息，运行或调试过程中的错误信息也会在此输出。 • 单击可以锁定代码编写区域，再次单击可以解锁。

示教点界面

示教点界面用于管理编程过程中要用到的机械臂点位，如下图所示。



序号	名称	说明
1	点位管理按钮	• 将机械臂运动到指定点位后，单击“添加”可以添加一条点位记录； • 勾选单个点位后单击“覆盖”则会用机械臂当前点位覆盖选中的点位； • 勾选点位后单击“删除”可以删除选中的点位； • 单击“保存”可以保存当前点位设置。
2	点位列表	显示已添加点位的信息，其中“工具”表示工具坐标系，“用户”表示用户坐标系。
3	“运行至”功能	勾选单个点位后，选择运动模式，再长按“运行至”按钮，机械臂会运行至勾选的点位。

操作步骤

下文以编写一个程序控制机械臂在两个点之间循环运动为例，介绍如何使用脚本编程的主要功能。

1. 打开示教点页签，将机械臂运动至任意点P1后，单击"添加"保存点P1的位置信息。
2. 将机械臂运动至另一任意点P2后，单击"添加"保存点P2的位置信息。

3. 打开编程页签，添加循环指令。
4. 在循环代码的"end"前添加一条移动指令，目标点为P1。
5. 再添加一条移动指令，目标点选择P2。

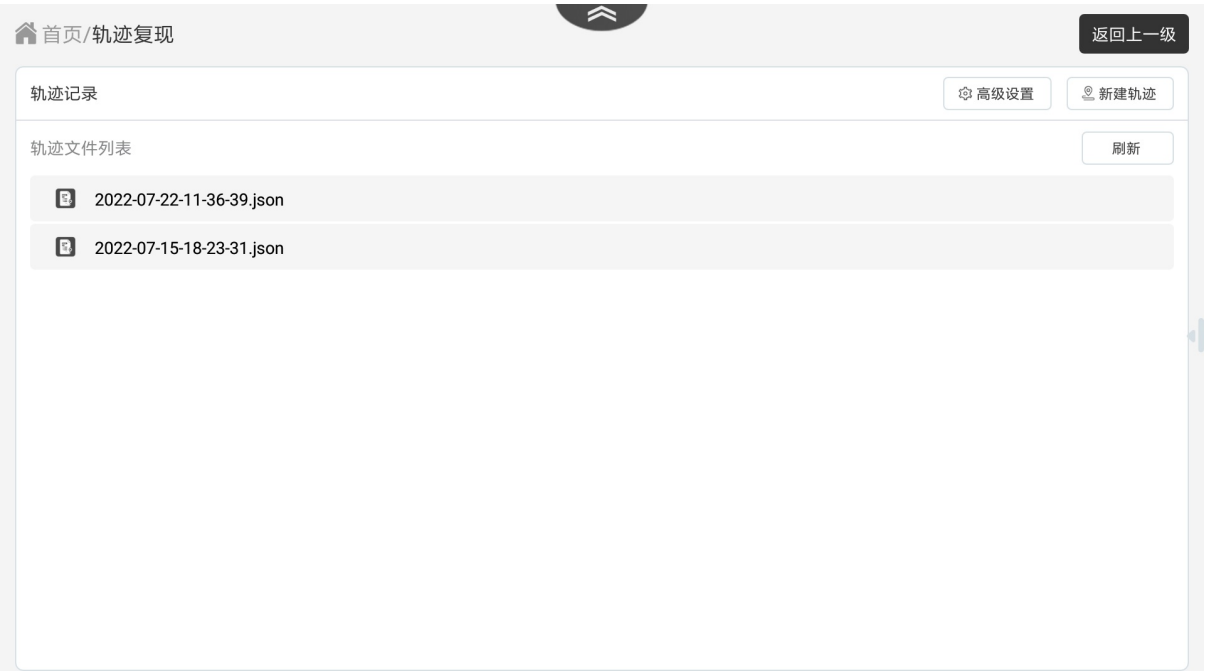
```
while(true)
do
    Go(P1)
    Go(P2)
end
```

6. 单击“保存”，自定义工程名称并单击“确定”。
7. 单击  ，机械臂开始运动。

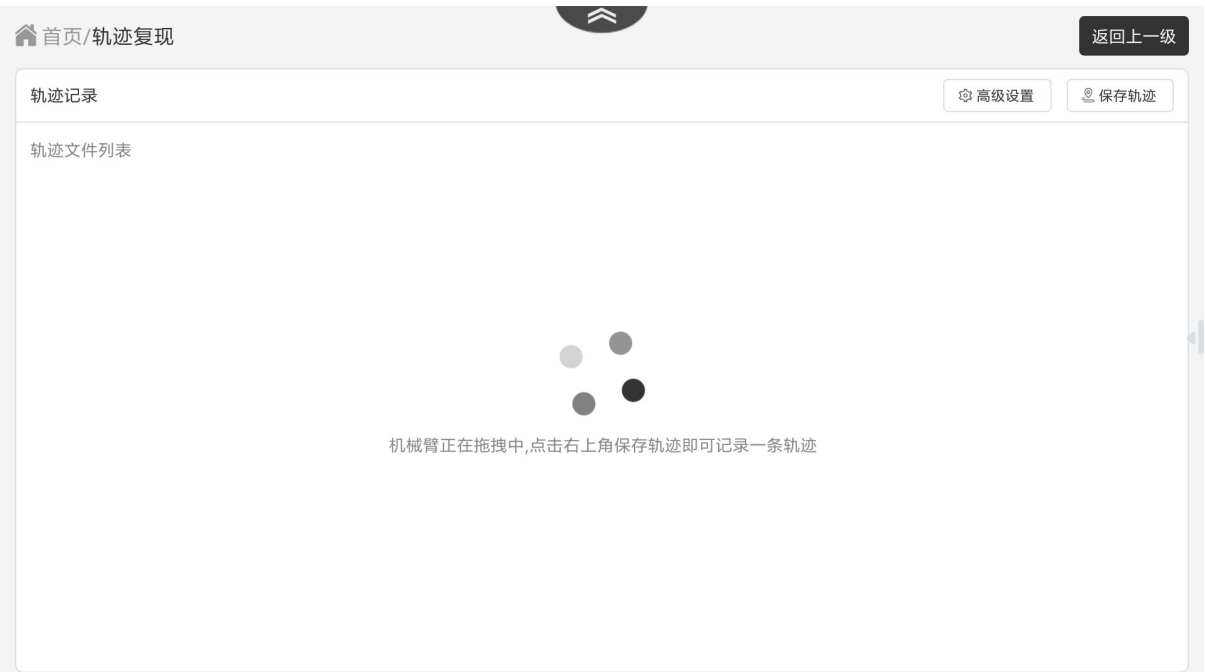
7 工艺

7.1 轨迹复现

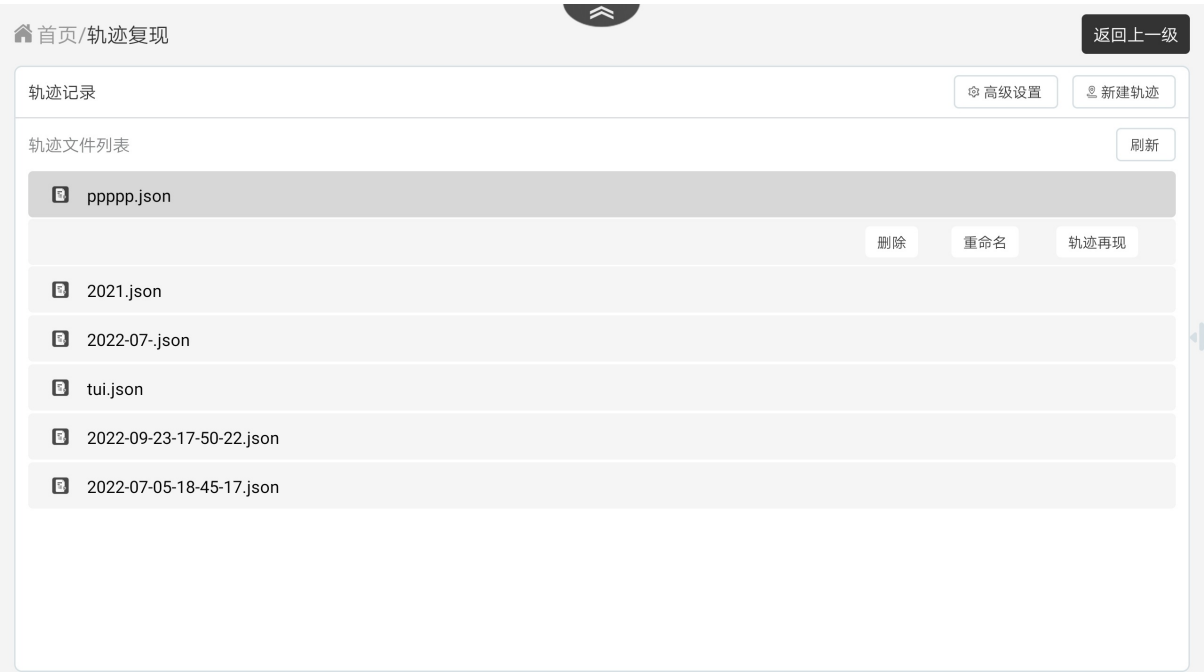
轨迹复现用于复现用户拖拽机械臂示教的轨迹，其主界面如下图所示。



单击右上角的“新建轨迹”后，机械臂末端指示灯变为黄色常亮，就可以开始拖拽机械臂本体，拖拽的轨迹会被记录，用于以后复现。



想要记录的轨迹完成后，单击右上角的“保存轨迹”，机械臂末端指示灯变为绿色常亮，轨迹文件列表中会新增一条以录制时间命名的记录。



单击选中已保存的轨迹后，会出现“删除”、“重命名”和“轨迹再现”按钮。

- 单击“轨迹复现”可以让机械臂复现记录的轨迹。此时机械臂末端指示灯变为黄色闪烁并开始自动运动。轨迹复现完成后机械臂会停止运动，末端指示灯变为绿色常亮。
- 单击“重命名”可以修改轨迹文件的名称。
- 单击“删除”则可以删除这条轨迹。

> 📖**说明** 已保存的轨迹文件也可在图形编程和脚本编程中通过轨迹复现相关指令调用。

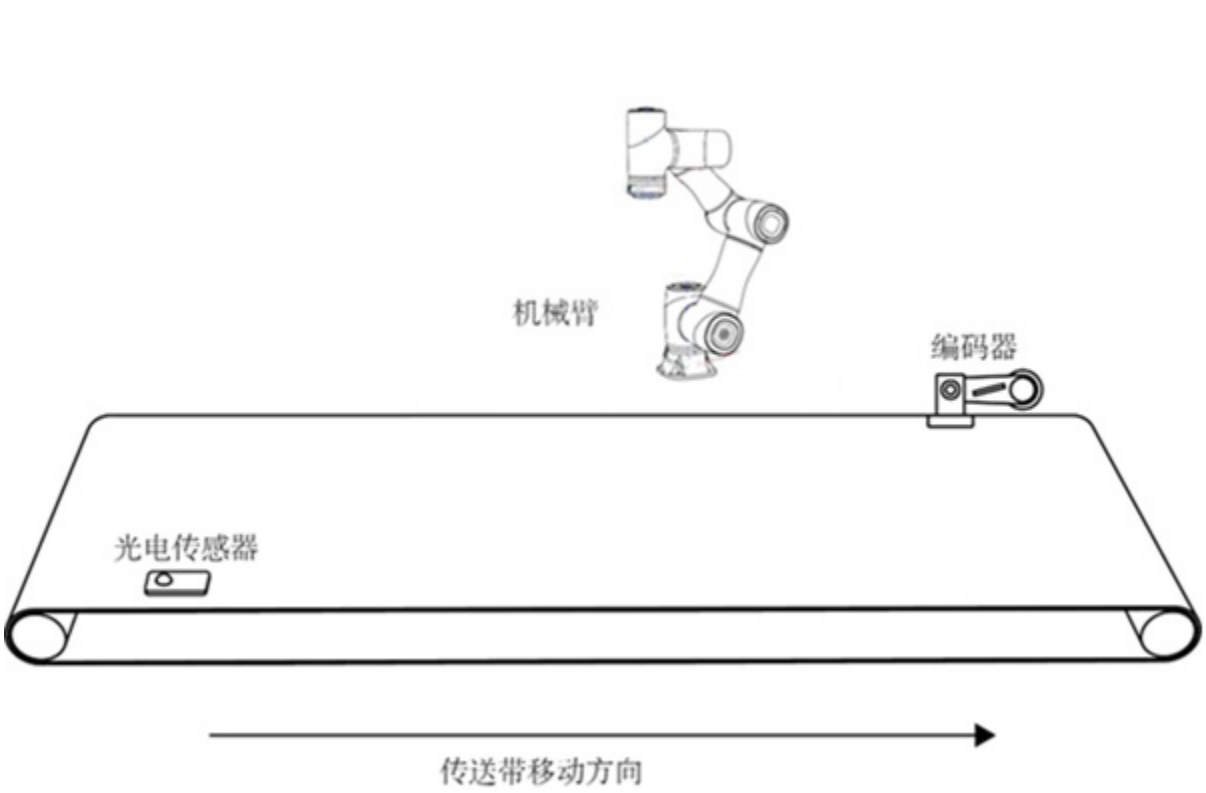
单击高级设置可以针对轨迹复现的方式进行设置。不勾选“匀速复现”时可以设置复现的速率。

7.2 传送带

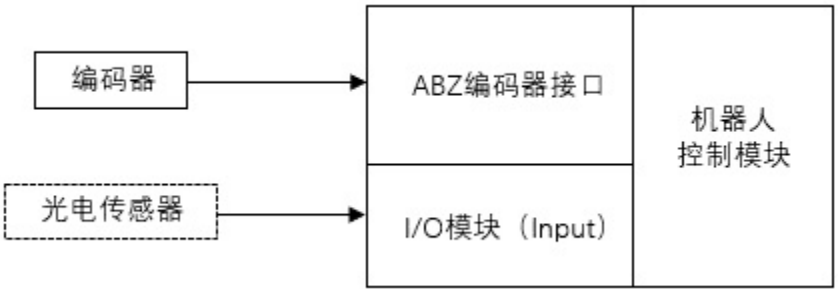
传送带工艺可通过光电传感器检测到传送带上的物体，CR在物体移动过程中对其进行准确的抓取或其他应用。

1 搭建环境

一套完整的传送带工艺环境如下图所示。



通信示意图如下。



编码器

编码器用于记录传送带移动距离和工件位置，并反馈给机器人控制系统，方便机械臂进行抓取。编码器与控制柜的ABZ编码器接口连接。建议选用E6B2-CWZ1X（1000P/R）编码器。

以E6B2-CWZ1X（1000P/R）编码器为例，引脚连接如下表所示。

颜色	端口
黑色	A+
黑色和红色	A-
白色	B+
白色和红色	B-
橙色	Z+
橙色和红色	Z-
棕色	I/O 5V
蓝色	I/O 0V

光电传感器

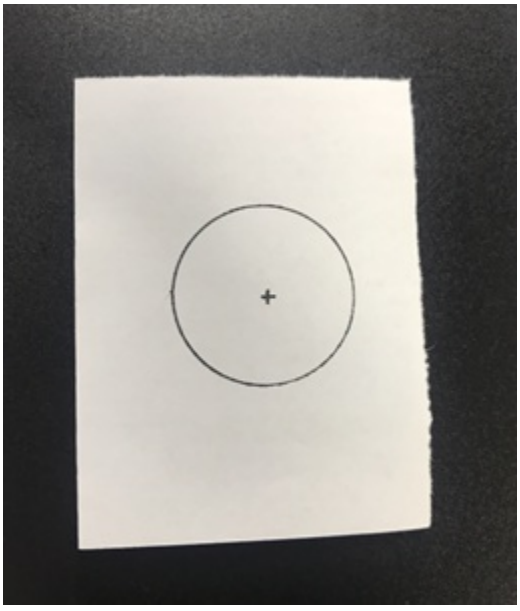
光电传感器检测到物体有无时会输出不同的电平信号，控制系统可通过信号边沿检测物体。将光电传感器连接至控制柜的I/O模块的任意DI接口。

2 标定传送带

配置传送带工艺前，需设置坐标系对传送带进行标定，用于描述传送带与机械臂之间的相对关系。下述标定过程中，采用用户坐标系进行标定。

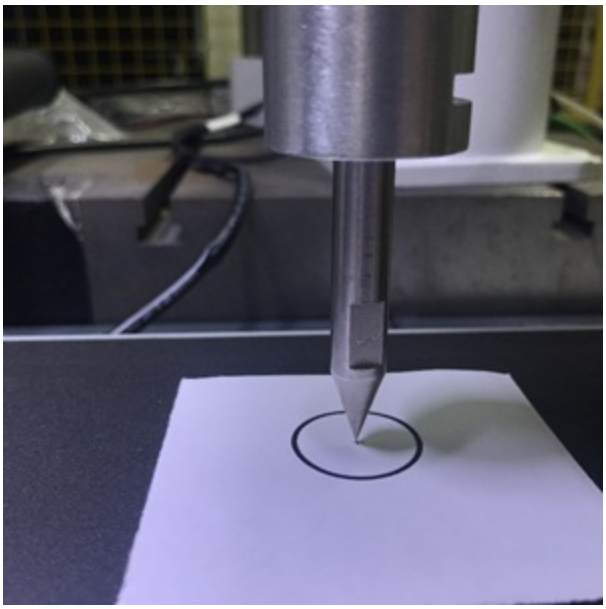
标定前需要在机械臂末端安装标定针，并准备好标定标签，本文不做详述。

1. 在传送带上贴上标签，如下图所示。

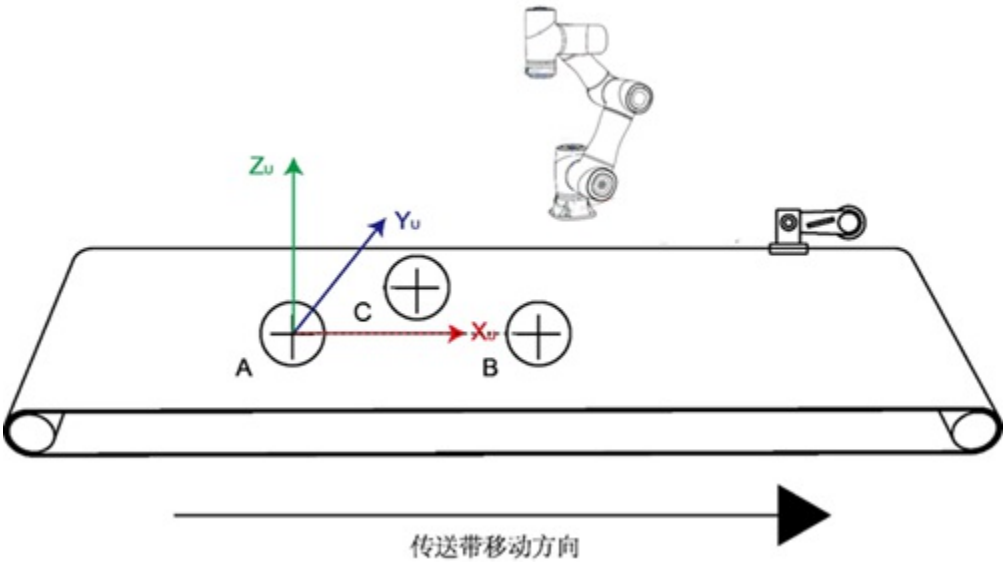


2. 进入用户坐标系设置界面。

3. 使能机械臂，并点动至传送带上的标签处，获取第一个点（点A，坐标系原点）。



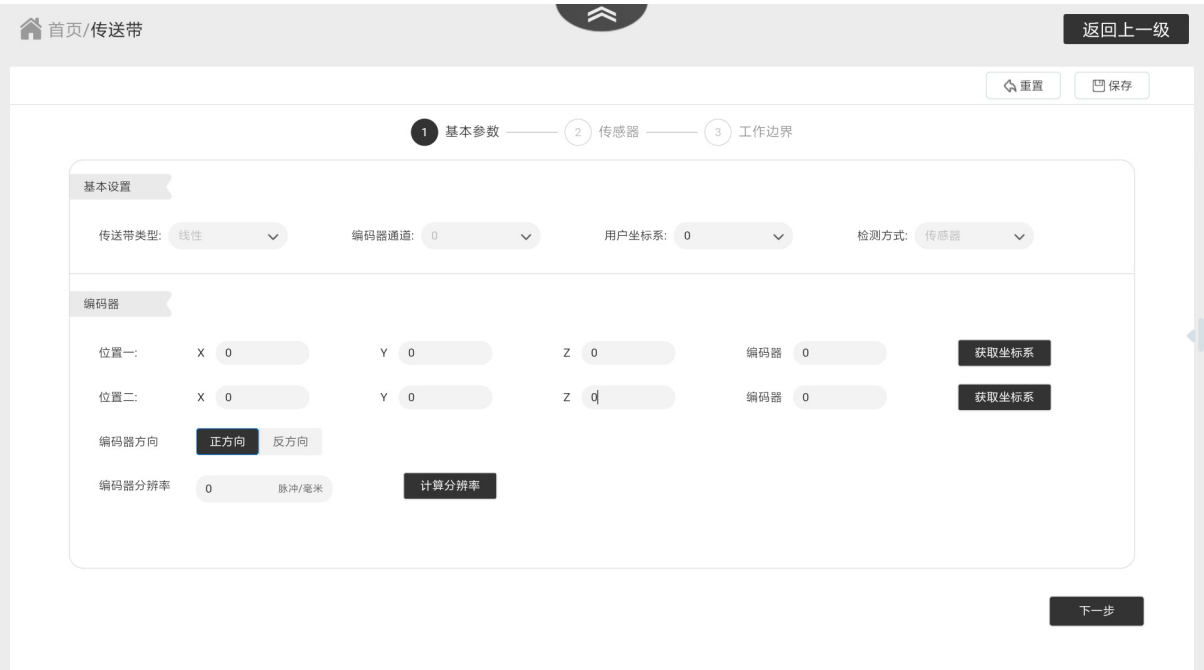
4. 控制传送带移动一段距离后停止，标签也随之移动。
5. 点动机械臂至标签处，获取第二个点（点B，用于确定X轴）。
6. 将机械臂点动至传送带上除AB点确定的直线外的任意位置，获取第三个点（点C，用于确定Y轴）。



7. 添加或覆盖坐标系，并保存。

3 配置传送带

单击工艺界面中的传送带图标开始配置传送带。



基本设置

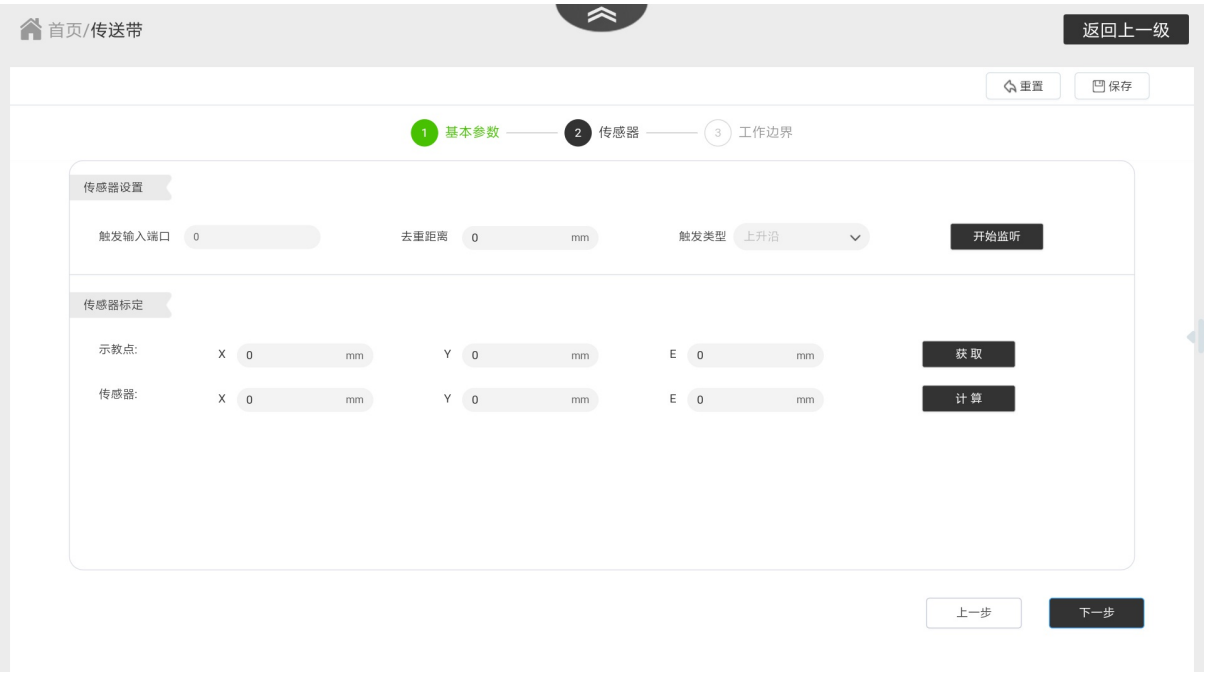
- 传送带类型：当前仅支持线性传送带，不可配置。
- 编码器通道：当前仅支持默认通道，不可配置。
- 用户坐标系：选择上一步保存的传送带坐标系。
- 检测方式：当前仅支持通过传感器检测，不可配置。

编码器

标定编码器的分辨率。编码器分辨率即传送带移动单位长度（mm）时编码器的脉冲增量。

1. 在传送带上贴上标签，并点动机械臂至传送带上的标签处，然后单击“位置一”所在行的“获取坐标系”，获取位置一的值。
2. 控制传送带移动一段距离并停止，标签也随之移动。
3. 点动机械臂至传送带上的标签处，然后单击“位置二”所在行的“获取坐标系”，获取位置二的值。
4. 根据实际情况设置编码器方向，本文以正方向为例。
5. 单击“计算分辨率”得出编码器的分辨率。

完成本页的设定后，单击“下一步”设置传感器。



传感器设置

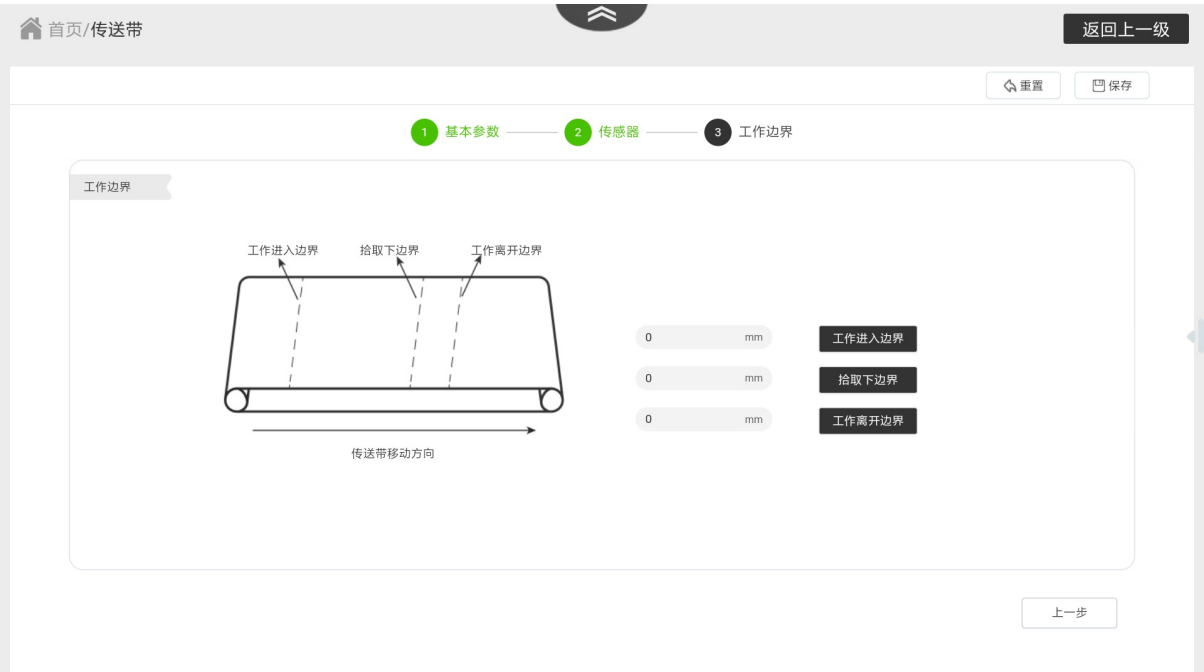
- 触发输入端口：设置传感器连接的控制柜DI端口的编号。
- 去重距离：去重即检测一个有效信号后，在后续一段距离内如果再检测到该信号变化，则认为是该信号为无效信号，会自动剔除。需根据实际情况设置，一般可设置为2mm~5mm。
- 触发类型：当前仅支持信号上升沿触发，不可配置。

传感器标定

传感器标定是为了在传感器被工件触发的瞬间，获取工件的位置（即传感器的位置），以便工件随传送带移动时，根据坐标的偏移来计算出每一时刻工件所在的用户坐标系下的位置。

1. 单击传感器设置中的“开始监听”，开启对传感器信号的监听。
2. 把一个工件放置在传感器上游，开启传送带，工件随传送带移动，待工件经过传感器且移动至机械工作范围内后停止传送带。
3. 点动机械臂至工件的中心位置，单击“获取”读取工件当前的位置。
4. 单击“计算”得出传感器的位置。

完成本页的设定后，单击“下一步”设置工作边界。



工作边界用于设置机械臂在传送带上可工作的范围。

- 工作进入边界：工件越过这个边界后，机械臂开始跟踪并拾取工件。
- 拾取下边界：工件越过这个边界后，机械臂如果还未开始拾取该工件，则判断为无法完成该工件的处理流程，不会尝试去拾取该工件。该范围需要根据传送带的运动速度以及实际经验设置，建议多次调试获取最佳值。
- 工作离开边界：工件越过这个边界后，机械臂会停止跟踪工件（为了防止跟踪至机器人运动区域外），并触发报警。

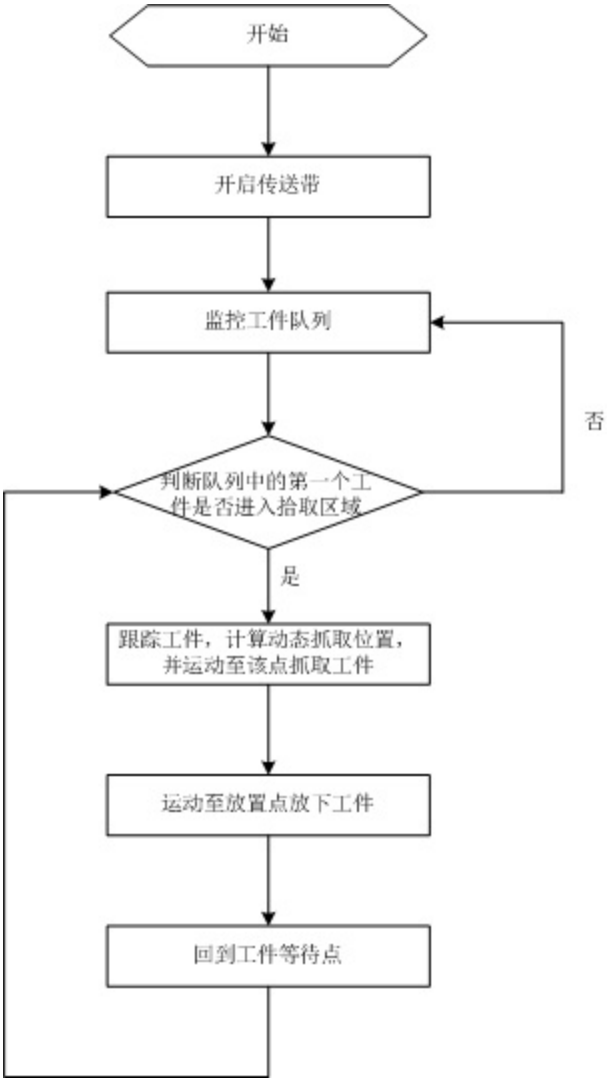
分别点动机械臂到各个边界的对应位置，并点击按钮进行设置。

设置完成后，单击右上角的“保存”，之后便可在工程中使用该传送带配置。

4 应用案例

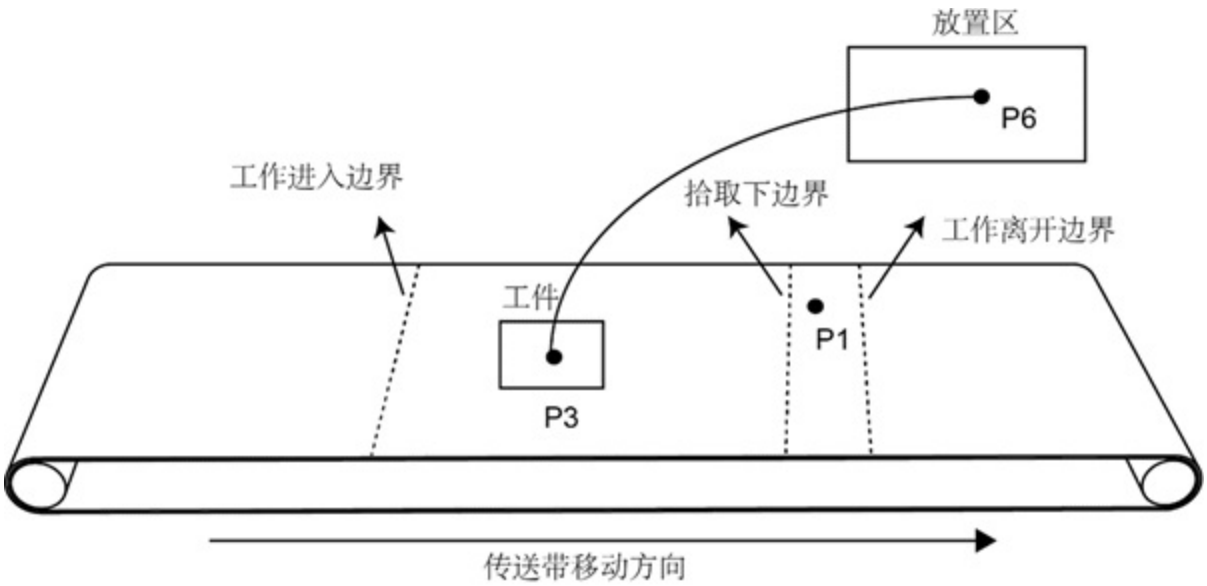
配置完传送带参数后，可通过调用传送带跟踪API编写脚本，实现传送带跟踪作业。

传送带跟踪作业的工作流程如下：



编写本案例的工程脚本时，需要在配置传送带时设置的用户坐标系下示教六个点：

- 等待工件点：P1
- 跟踪工件上方点：P2
- 抓取工件点：P3
- 抬升工件点：P4
- 放置上方点：P5
- 放置点：P6



案例脚本如下：

```
CnvVison(0) ----------------------//开启传送带

DO(9,0) ----------------------//通过 DO1 和 DO2 控制气泵启停和状态

DO(2,0)

local flag  ----------------------//是否存在工件标识

local typeObject  ----------------------//工件的属性分类

local point = {0,0,0}  ----------------------//跟踪队列的工件的坐标

while true do

      Go(P1,"Speed=100 Accel=100 SYNC=1") ---//机械臂等待点，一般设置在工作空间内

      print("Test")

      while true do

            flag,typeObject,point = GetCnvObject(0,0) --//查询是否有工件，如果有则跳出循环

            if flag == true then

                  break

            end

            Sleep(20)

      end

      SyncCnv(0) ----------------------//同步传送带，开始跟踪

      Move(P2,"SpeedS=100 AccelS=100") ------------------//跟踪工件上方点
```

```
    Move(P3,"SpeedS=100 AccelS=100") ------------------//抓取工件点
    Wait(100)
    DO(9,1) ----------------------//开启气泵，抓取工件
    --DO(2,1)
    Wait(100)
    Move(P4,"SpeedS=100 AccelS=100 SYNC=1")  ---------//抬升工件点
    StopSyncCnv() ----------------------//停止传送带跟踪
    Sleep(20)
    Go(P5,"Speed=100 Accel=100")  ----------------//放置点上方
    Go(P6,"Speed=100 Accel=100 SYNC=1")  --------//放置点
    Sleep(1000)
    DO(1,0) ----------------------//关闭气泵
    DO(2,0,"SYNC=1")
    Sleep(1000)
    Go(P5,"Speed=100 Accel=100")
end
```

8 最佳实践

本章将会介绍通过远程I/O控制机械臂的完整流程，以帮助您理解Dobot CRStudio的各个功能是如何配合使用的。

为方便您理解流程，我们先预设一个示例场景：按下开始按钮后，运行指示灯亮起，然后机械臂通过末端的夹爪从取料点抓取物料，运动至下料点放下物料，然后再回到取料点抓取物料，重复循环。

为达成上述场景，需要给机械臂末端安装夹爪（以DH系列为例，安装方法请参考DH系列夹爪使用手册），并将按钮和指示灯连接至控制柜I/O接口（假设开始按钮连接至DI11，停止按钮连接至DI12；运行指示灯连接至DO11，报警指示灯连接至DO12；连线方法请参考机器人对应的硬件使用手册）。

整体流程

硬件安装完成且机械臂上电后，软件操作的整体流程如下：

1. 连接机器人
2. 安装Dobot+插件（可选）
3. 设置并选择工具坐标系
4. 编写工程
5. 配置并进入远程I/O模式

操作步骤

连接并使能机器人

连接机械臂的详细说明请参考连接机器人，此处以无线连接为例简单说明。

1. 在平板上搜索并连接机器人的WiFi，WiFi的默认SSID为“Dobot_WIFI_XXX”，其中XXX为位于机械臂底座上的机械臂编号；初始WiFi密码为：1234567890。
2. 在Dobot CRStudio界面上方选择单击“开始连接”,连接成功后，“开始连接”按钮变为“断开连接”按钮。

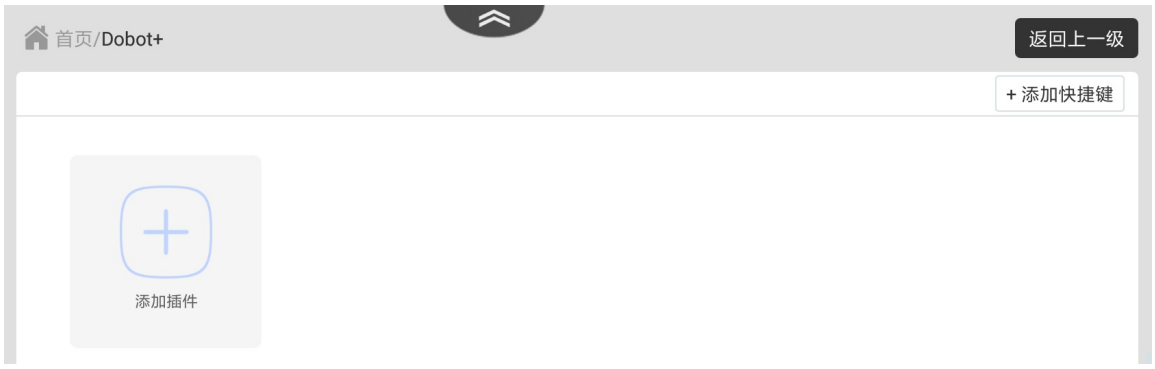
3. 单击使能按钮，设置负载参数后使能机械臂。

安装Dobot+插件（可选）

Dobot+插件的详细说明请参考末端插件，此处以安装DH插件为例简单说明。

如果安装的末端工具无对应插件，则跳过该步骤，后续编程时直接通过I/O指令控制末端工具。

1. 打开“监控 > Dobot+”面板。
2. 单击“添加插件”，然后单击DH插件左侧的"+“即可完成安装。



设置并选择工具坐标系

标定夹爪的工具坐标系，具体操作请参考工具坐标系。

编写工程

编程的详细说明请参考图形编程和脚本编程，此处以图形编程为例简单说明。

为达成本章开头描述的示例场景，我们需要先示教四个点，分别是取料点P1，取料点上方过渡点P2，下料点上方过渡点P3，以及下料点P4。

1. 打开存点页面，将机械臂运动至P1，然后单击“新增”。

2. 用同样方法新增示教P2，P3，P4。

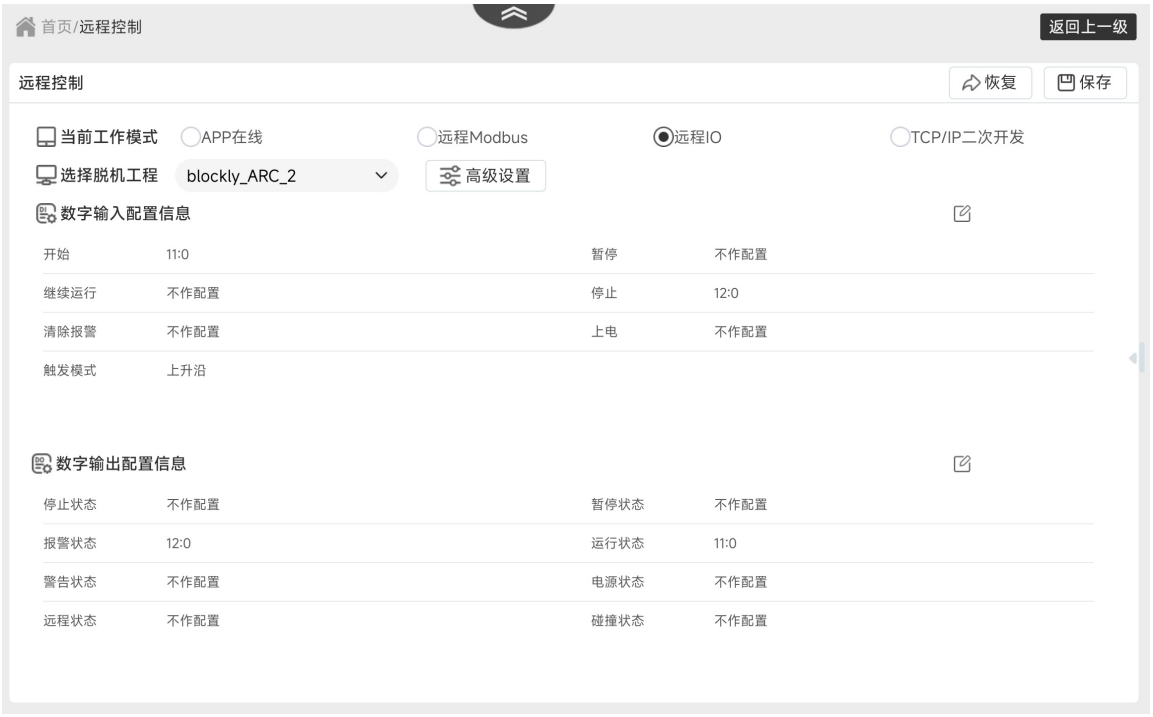
3. 拖动积木编程实现取料和下料。一个简单的程序示例如下图。

4. 保存工程。

配置并进入远程I/O模式

远程控制的详细说明请参考远程控制，此处仅基于示例场景做简单说明。

1. 打开“设置 >远程控制”页面。
2. 修改当前工作模式为“远程IO”。
3. 选择上一节保存的图形编程工程。
4. 单击编辑图标，根据本章开头所述场景修改I/O配置。
5. 单击”保存“，进入远程IO模式。



进入远程I/O模式后，按下已连接到机械臂控制柜的开始按钮，机械臂就会开始运行工程。

附录A Modbus寄存器定义

1 Modbus介绍

Modbus是一种串行通信协议，通过此协议，CR系列机器人可与外部设备进行通信。要通过外部设备如PLC控制CR系列机器人时，外部设备作为Modbus主站，CR系列机器人作为Modbus从站。

Modbus目前存在以下版本：

- Modbus-RTU：紧凑的，采用二进制表示数据的方式。使用循环冗余校验的校验和的方式。
- Modbus-ASCII：基于ASCII码的字符串的表示方式，是一种可读的、冗余的表示方式。使用纵向冗余校验的校验和的方式。
- Modbus-TCP：通过TCP/IP的TCP方式连接。这种方式不需要校验和计算。

基于我们现有的硬件接口，目前我们采用Modbus-TCP的方案。

通常Modbus地址由5位数字组成，包括起始的数据类型代号，以及后面的偏移地址。Modbus Master协议库把标准的Modbus地址映射为Modbus功能号，读写从站的数据。

Modbus Master协议库支持如下地址：

- 00001-09999：数字量输出（线圈）
- 10001-19999：数字量输入（触点）
- 30001-39999：输入数据寄存器（通常为模拟量输入）
- 40001-49999：数据保持寄存器

Modbus协议的数据主要分为四类：线圈状态、离散量输入、输入寄存器、保持寄存器，基于CR系列机器人的内存空间，我们定义了线圈、触点（离散输入）、输入、保持寄存器，用于外部设备与机器人控制系统进行数据交互。其中，每个寄存器地址为4096个。详细说明如下所示。

2 线圈寄存器（控制机器人）

PLC地址	脚本地址（Get/SetCoils）	寄存器类型	功能
00001	0	Bit	启动
00002	1	Bit	暂停
00003	2	Bit	继续
00004	3	Bit	停止

00005	4	Bit	急停
00006	5	Bit	清除报警
00007	6	Bit	上电
03096~04096	3095~4095	Bit	用户自定义

3 离散输入定义（机器人状态）

PLC地址	脚本地址(GetInBits)	寄存器类型	功能
10002	1	Bit	停止状态
10003	2	Bit	暂停中
10004	3	Bit	运行状态
10005	4	Bit	报警状态
10006	5	Bit	系统预留
10008	7	Bit	远程模式
13096~14096	3095~4095	Bit	用户自定义

4 输入寄存器定义

PLC地址	数据类型	值的数目	字节大小	脚本地址(GetInRegs)	寄存器类型	功能
31000	unsigned short	1	2	999	U16	数据有效性
31001	unsigned short	1	2	1000	U16	消息字节总长度
31002~31004	unsigned short	3	6	1001~1003	U16	保留位
31005~31008	uint64	1	8	1004~1007	U64	数字输入
31009~31012	uint64	1	8	1008~1011	U64	数字输出
31013~31016	uint64	1	8	1012~1015	U64	机器人模式
31017~31020	uint64	1	8	1016~1019	U64	时间戳
31021~31024	uint64	1	8	1020~1023	U64	保留位

31025~31028	uint64	1	8	1024~1027	U64	内存结构测试标准值 0x0123 4567 89AB CDEF
31029~31032	double	1	8	1028~1031	F64	保留位
31033~31036	double	1	8	1032~1035	F64	速度比例
31037~31040	double	1	8	1036~1039	F64	机器人当前动量
31041~31044	double	1	8	1040~1043	F64	控制板电压
31045~31048	double	1	8	1044~1047	F64	机器人电压
31049~31052	double	1	8	1048~1051	F64	机器人电流
31053~31056	double	1	8	1052~1055	F64	保留位
31057~31060	double	1	8	1056~1059	F64	保留位
31061~31072	double	3	24	1060~1071	F64	TCP加速度
31073~31084	double	3	24	1072~1083	F64	肘位置
31085~31096	double	3	24	1084~1095	F64	肘速度
31097~31120	double	6	48	1096~1119	F64	目标关节位置
31121~31144	double	6	48	1120~1143	F64	目标关节速度
31145~31168	double	6	48	1144~1167	F64	目标关节加速度
31169~31192	double	6	48	1168~1191	F64	目标关节电流
31193~31216	double	6	48	1192~1215	F64	目标关节扭矩
31217~31240	double	6	48	1216~1239	F64	实际关节位置
31241~31264	double	6	48	1240~1263	F64	实际关节速度
31265~31288	double	6	48	1264~1287	F64	实际关节电流
31289~31312	double	6	48	1288~1311	F64	TCP传感器力值 (通过六维力计算)
31313~31336	double	6	48	1312~1335	F64	TCP笛卡尔实际坐标值
31337~31360	double	6	48	1336~1359	F64	TCP笛卡尔实际速度值
31361~31384	double	6	48	1360~1383	F64	TCP力值 (通过关节电

						流计算)
31384~31408	double	6	48	1384~1407	F64	TCP笛卡尔目标坐标值
31409~31432	double	6	48	1408~1431	F64	TCP笛卡尔目标速度值
31433~31456	double	6	48	1432~1455	F64	关节温度
31456~31480	double	6	48	1456~1479	F64	关节控制模式
31481~31504	double	6	48	1480~1503	F64	关节电压
31505~31506	char	4	4	1504~1505	U16	手系
31507	char	1	1	1506低字节	U8	用户坐标
31507	char	1	1	1506高字节	U8	工具坐标
31508	char	1	1	1507低字节	U8	算法队列运行标志
31508	char	1	1	1507高字节	U8	算法队列暂停标志
31509	char	1	1	1508低字节	U8	关节速度比例
31509	char	1	1	1508高字节	U8	关节加速度比例
31510	char	1	1	1509低字节	U8	关节加加速度比例
31510	char	1	1	1509高字节	U8	笛卡尔位置速度比例
31511	char	1	1	1510低字节	U8	笛卡尔姿态速度比例
31511	char	1	1	1510高字节	U8	笛卡尔位置加速度比例
31512	char	1	1	1511低字节	U8	笛卡尔姿态加速度比例
31512	char	1	1	1511高字节	U8	笛卡尔位置加加速度比例
31513	char	1	1	1512低字节	U8	笛卡尔姿态加加速度比例
31513	char	1	1	1512高字节	U8	机器人抱闸状态
31514	char	1	1	1513低字节	U8	机器人使能状态

31514	char	1	1	1513高字节	U8	机器人拖拽状态
31515	char	1	1	1514低字节	U8	机器人运行状态
31515	char	1	1	1514高字节	U8	机器人报警状态
31516	char	1	1	1515低字节	U8	机器人点动状态
31516	char	1	1	1515高字节	U8	机器类型
31517	char	1	1	1516低字节	U8	按钮板拖拽信号
31517	char	1	1	1516高字节	U8	按钮板使能信号
31518	char	1	1	1517低字节	U8	按钮板录制信号
31518	char	1	1	1517高字节	U8	按钮板复现信号
31519	char	1	1	1518低字节	U8	按钮板夹爪控制信号
31519	char	1	1	1518高字节	U8	六维力在线状态
31520~31560	char	1	82	1519~1559	U16	保留位
31561~31584	double	6	48	1560~1583	F64	实际扭矩
31585~31588	double	1	8	1584~1587	F64	负载重量（kg）
31589~31592	double	1	8	1588~1591	F64	X方向偏心距离（mm）
31593~31596	double	1	8	1592~1595	F64	Y方向偏心距离（mm）
31597~31600	double	1	8	1596~1599	F64	Z方向偏心距离（mm）
31601~31624	double	6	48	1600~1623	F64	用户坐标值
31625~31648	double	6	48	1624~1647	F64	工具坐标值
31649~31652	double	1	8	1648~1651	F64	轨迹复现运行索引
31653~31676	double	6	48	1652~1675	F64	当前六维力数据原始值

31677~31692	double	4	32	1676~1691	F64	[qw,qx,qy,qz] 目标四元数
31693~31708	double	4	32	1692~1707	F64	[qw,qx,qy,qz] 实际四元数
31709~31721	double	1	24	1708~1720	F64	保留位
			1440			共1440字节

5 保持寄存器定义（机器人PLC交互）

PLC地址	脚本地址 (Get/SetHoldRegs)	寄存器类型	功能
40001~41281	0~1280	U16	码垛
43095~44095	3095~4095	U16	用户自定义
49001	9000	U16	当前螺丝序号
49002	9001	U16	指定要执行的螺丝序号： 序号从1开始，0代表没有指定螺丝序号
49003	9002	U16	螺丝锁附结果： 0代表失败；1代表成功；2代表None
49004	9003	U16	锁附产品（工位）上的螺丝数
49005~49054	9004~9053	U16	产品锁附结果： 最大支持50组；0代表失败；1代表成功；2代表None
49055	9054	U16	报警代码： 0没有报警；1取料时螺丝机物料；2取料前批头已有螺丝；3螺丝取料失败；4螺丝锁附异常；5电批异常
49056	9055	U16	锁附的螺丝总数
49057	9056	U16	锁附的不良螺丝总数
49058	9057	U16	锁附的产品总数
49201	9220	F64	螺丝锁附扭力（Nm)
49205	9224	F64	螺丝锁附角度（°）
49209	9228	F64	螺丝锁附节拍（s）

49213~49412	9232~9431	F64	产品锁附扭力（Nm）； 最大支持50组
49413~49612	9432~9631	F64	产品锁附角度（°） 最大支持50组

附录B 图形编程积木说明

快速体验

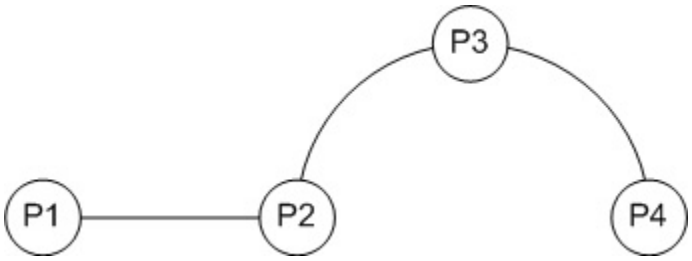
- 控制机械臂运动
- 读写Modbus寄存器数据
- 通过TCP通信传递数据
- 使用托盘积木进行码垛

控制机械臂运动

场景说明

为了方便体验如何用图形编程控制机械臂运动，我们先假设一个简单的上下料场景：

机械臂从P1取料，直线运动到P2后，经由P3圆弧运动到P4放料，然后再原路返回取料，重复十次。



为实现这个场景，我们需要示教这四个点。

然后我们假设：

- 机械臂已安装夹爪，通过末端的DO1进行控制（设置为开触发夹爪进行抓取，设置为关触发夹爪松开）。
- 取料点有传感器，连接到控制柜DI1（开表示有料，关表示无料）。
- 外接报警装置连接到控制柜DO1（设置为开触发报警）。

调试时可以通过监控页面模拟DI和监视DO。

编程步骤

接下来，让我们开始编程。

1. 首先，设置机械臂的运动参数。此处仅为示例，请根据实际需要设置。
2. 然后，将机械臂关节运动至P1，并设置同步指令，等机械臂运动完成再执行后续指令。

3. 接下来设置一个十次的循环，并在其中嵌套其他积木。

4. 嵌套一个条件判断积木，当取料点有料时控制夹爪进行抓取，无料时则触发报警并暂停工程。

5. 设置下发夹爪抓取指令后等待3秒，等取料成功再运动机械臂。实际等待时间请根据实际情况设置。

6. 然后将机械臂直线运动至P2，接着再经由P3圆弧运动至P4，然后设置同步指令。

7. 下发夹爪松开指令，等待3秒，再让机械臂原路返回，准备执行下一次循环。

读写Modbus寄存器数据

场景说明

为了体验如何通过图形编程读写Modbus数据，我们先假设要实现以下场景：

1. 先分别给一个线圈寄存器和一个保持寄存器写入值。
2. 读取一个输入寄存器的正整数值，赋给一个变量。
3. 设置一个循环，当上述变量的值为0时停止循环。
4. 在循环中读取一个线圈寄存器的值，如果为1则执行运动指令，并把上述变量减1。

编程步骤

接下来，让我们开始编程。

1. 首先，创建一个主站并连接子站。子站的IP、端口和ID根据实际情况设置。
2. 然后，判断主站创建是否成功，不成功的时候打印“Create failed!”。
3. 创建主站成功的场合，分别给一个线圈寄存器和一个保持寄存器写入值，地址和值都请根据实际情况设置。

4. 接下来建立一个新数值变量，变量名称自定义。

5. 读取一个输入寄存器的值并赋给该变量。
6. 设置一个循环，循环终止条件设置为该变量为0。
7. 在循环中，当指定线圈寄存器的值为1时，运动机械臂（下图仅使用一条运动指令作为示例，实际需要是可循环执行的多条运动指令）并将该变量减一。
8. 循环结束后，关闭Modbus主站。


如果
获取创建主站结果
=
0
那么
设置线圈寄存器 地址
1001
数值
1
设置保持寄存器 地址
1
数值
50
类型
U16
将
test
设为
获取输入寄存器 地址
1
类型
U16
重复执行直到
test
=
0
如果
获取线圈寄存器 地址
1001
=
1
那么
关节运动
至
点
P1
将
test
增加
-1
关闭主站

通过TCP通信传递数据

场景说明

为了体验如何通过图形编程进行TCP通信，我们先假设要实现以下场景：

1. 打开TCP端口，作为TCP客户端连接TCP服务端（我们也支持创建TCP服务端，此处以客户端为例）。
2. 发送一个字符串到服务端。
3. 从服务端接收一个字符串，假设这个字符串是逗号分割的多个数值，例如“10,15,20,25”。
4. 控制机械臂循环运动，循环次数为这个字符串的第一个逗号前的数值。

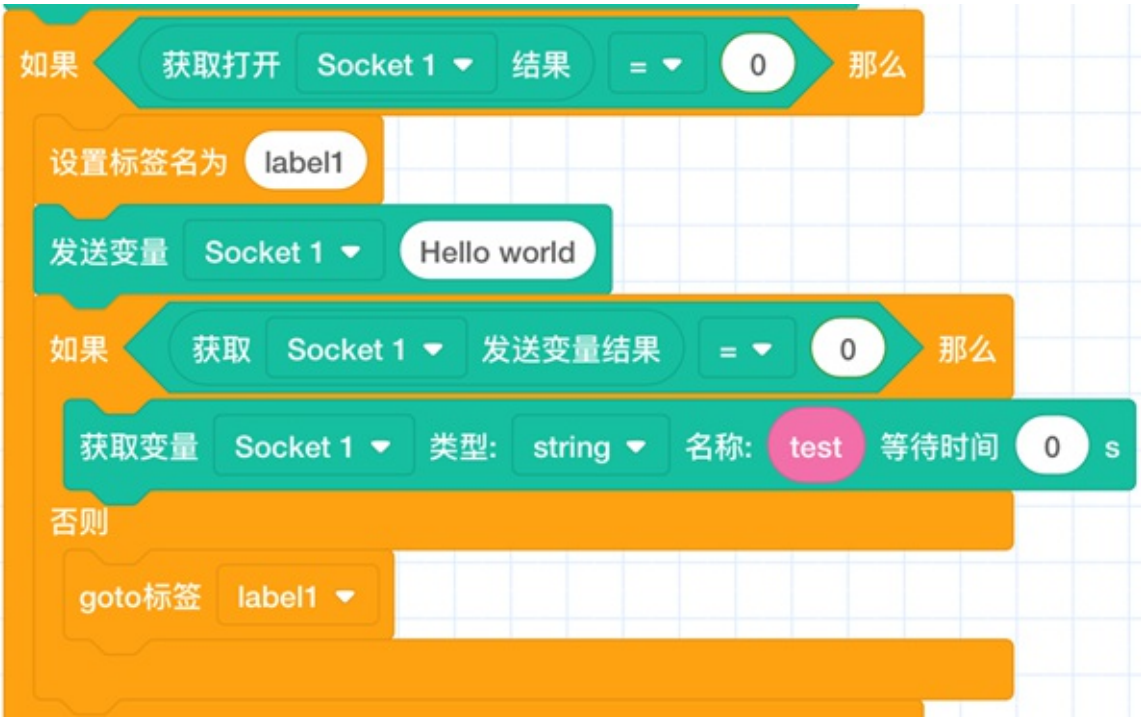
编程步骤

接下来，让我们开始编程。

1. 首先，连接Socket 1，连接TCP服务端。服务端的IP和端口根据实际情况设置。
2. 然后，判断连接是否成功，连接失败时打印“failed:”加连接结果的返回值。
3. 连接成功时，发送一个变量给服务端，例如"Hello world"。

4. 接下来建立一个新字符串变量，变量名称自定义（例如"test"）。

5. 在发送变量的积木前面插入一个标签。
6. 判断发送变量的结果，如果发送失败，则跳转到标签处，重新发送变量。
7. 如果发送成功，则从服务端获取字符串类型的变量并保存到“test”中。



8. 将变量“test”转换为数组。
9. 设定一个循环，循环次数为上述的数组的第一个元素。
10. 在循环中运动机械臂（下图仅使用一条运动指令作为示例，实际需要是可循环执行的多条运动指令）。

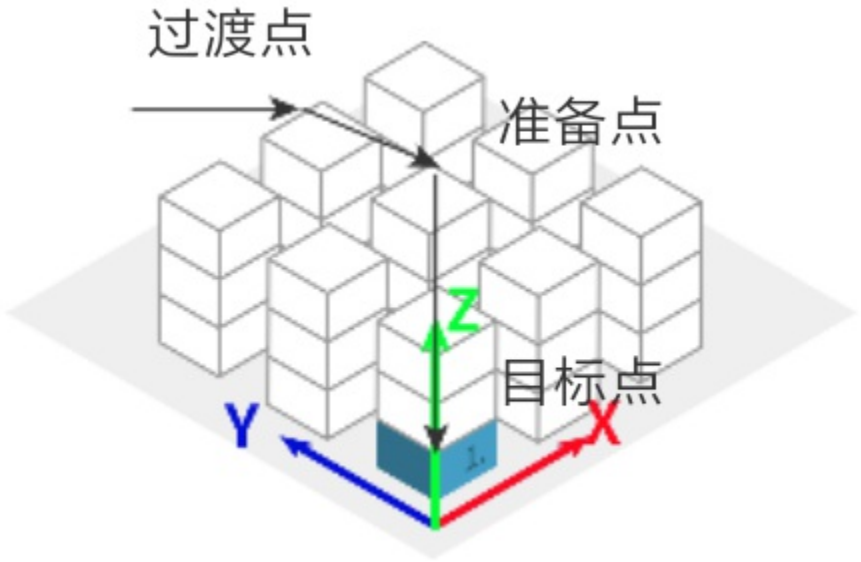
11. 在程序的最后，关闭Socket 1。

使用托盘积木进行码垛

场景说明

在搬运应用中，有些搬运的物料排列规则且间距均匀，若一一示教各个物料的位置会存在误差大、效率低的问题。使用托盘积木可通过设置跺型自动计算出各物料的位置，有效的解决该类问题。

首先，我们假设需要将物料堆放成一个立方体，我们需要先手动码垛一个目标跺型出来，然后示教相关点位：



- 示教过渡点为P1。过渡点是机械臂进出码垛区域都会经过的一个点位，用于安全 过渡，可示教为取料点上方。
- 示教取料箱的点位为P2。
- 准备点和目标点不需要逐个示教，请参考后续配置跺型。

然后，我们假设机械臂末端已安装夹爪或吸盘等机构，通过控制柜的DO1控制抓取或放置物料。

配置跺型

将创建托盘积木拖动到编程区后，单击积木可打开托盘面板。

托盘维度

- 一维：物料排成一行，物料总数等于X方向上的个数。
- 二维：物料排成方形，物料总数等于X方向和Y方向上的个数的乘积。
- 三维：物料堆放成立方体，物料总数等于三个方向上的个数的乘积。

本文以三维跺型为例，分别设置各个方向上的物料个数为10，则一个完整的跺型总共有1000个物料。

点位配置

以三维跺型为例，此处需要配置八个点，分别对应立方体的八个角上的物料的目标点（放料时的机械臂点位），控制系统会通过这八个点和物料的数量自动计算出每个物料的目标点，然后按照X->Y->Z的坐标轴顺序进行排序。

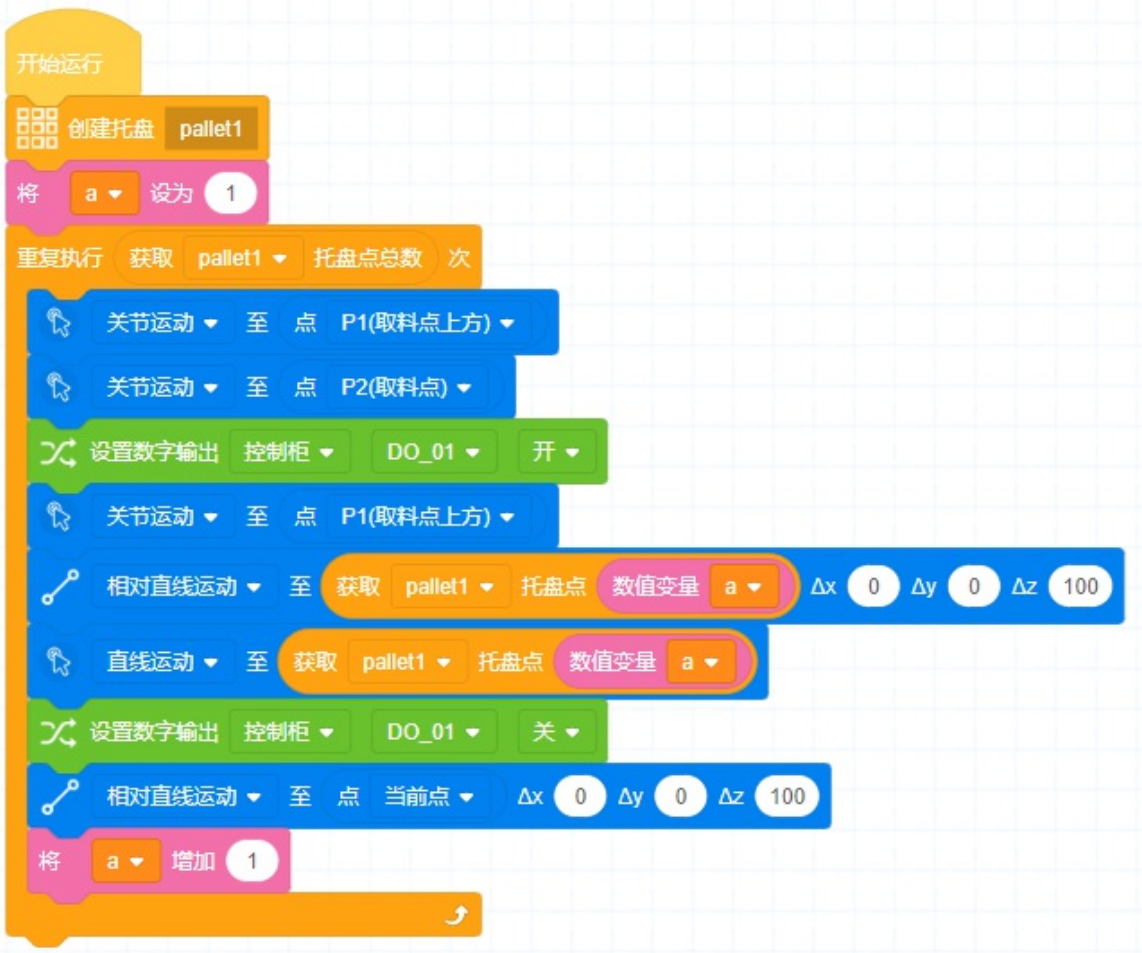
配置点位时可以选择工程中已示教的存点，也可以单击“自定义”后获取机械臂当前点位坐标。已配置的点位图标会变为绿色。

编程步骤

接下来，让我们开始编程。

1. 新建一个自定义数值变量，将值设为1，用于记录循环次数。
2. 添加一个循环积木，循环次数为对应托盘的点总数。
3. 添加运动和IO积木，控制机械臂进行取料。

4. 添加运动和IO积木，控制机械臂到当前托盘点放料，当前托盘点的索引号等于当前的循环次数。
5. 循环次数加一。



本文编写的程序仅作为一个简单的示例，实际应用可以根据实际情况加上更多的IO控制和情况判断，例如没取到物料时就不执行后续动作。

积木说明

事件积木组

事件积木组用于作为程序开始运行的标识。

开始运行

描述：程序主线程的标识。创建新工程后编程区默认会有一个开始运行标识，请将其他非事件积木接在其下方进行编程。

使用限制：一个工程中只能有一个“开始运行”积木。

子线程启动

描述：程序子线程的标识。子线程会和主线程同步运行，但子线程不可以使用机械臂控制命令，仅可进行变量运算或IO控制等。请根据程序逻辑需要选择是否使用子线程。

使用限制：一个工程中最多只能有5个子线程。

控制积木组

控制积木组用于控制程序的运行路径。

等待直到满足条件

描述：程序暂停，直到参数为true再继续运行。

参数：使用其他六边形积木作为参数。

重复执行n次

描述：把其他积木嵌套到该积木中间，被嵌套的积木指令会被重复执行指定的次数。

参数：重复执行的次数。

持续重复执行

描述：把其他积木嵌套到该积木中间，被嵌套的积木指令会被一直重复执行，直到遇到结束重复积木。

结束重复

描述：用于嵌套在重复执行类的积木中，程序执行到该基本的时候会直接结束重复，执行重复积木之后的积木指令。

满足条件后执行

描述：如果参数为true，则执行嵌套的积木指令。参数为false时直接跳到下一个积木指令。

参数：使用其他六边形积木（返回值为布尔值，即true或者false）作为参数。

满足或不满足条件后分别执行

描述：如果参数为true，则执行“否则”之前嵌套的积木指令。参数为false时，执行“否则”之后嵌套的积木指令。

参数：使用其他六边形积木（返回值为布尔值，即true或者false）作为参数。

重复执行直到满足条件

描述：重复执行嵌套的积木指令直到参数为true。

参数：使用其他六边形积木（返回值为布尔值，即true或者false）作为参数。

设置标签

描述：设置一个标签，设置后可以通过标签跳转积木进行跳转。

参数：标签名称，必须以字母开头，不能使用空格等特殊字符。

标签跳转

描述：程序执行到该积木后，会直接跳转到指定的标签，执行标签之后的积木指令。

参数：已设置的标签名称。

指令折叠

描述：可以将嵌套的积木折叠显示。不起控制作用，仅用于使程序更加美观和易读。

参数：描述被折叠的积木块，建议起一个间接且直观的名称。

暂停

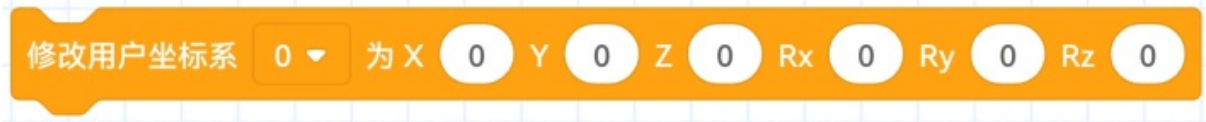
描述：程序运行到该积木后自动暂停，需要通过控制软件或者远程控制操作才可继续运行。

设置碰撞检测功能

描述：设置碰撞检测功能。通过该积木设置的碰撞检测等级仅在工程运行期间生效，工程停止后会恢复为修改前的值。

参数：选择碰撞检测功能的灵敏度，可选择关闭或等级1到5，等级数字越大碰撞检测越灵敏。

修改用户坐标系



描述：修改指定的用户坐标系。该修改仅在当前工程运行中生效，工程停止后坐标系会恢复为修改前的值。

参数：

- 指定要修改的用户坐标系的编号。
- 指定修改后的用户坐标系参数。

修改工具坐标系

描述：修改指定的工具坐标系。该修改仅在当前工程运行中生效，工程停止后坐标系会恢复为修改前的值。

参数：

- 指定要修改的工具坐标系的编号。
- 指定修改后的工具坐标系参数。

创建托盘

描述：用于创建一个托盘的跺型，详见使用托盘积木进行码垛。

参数：要创建的托盘的名称。

获取指定托盘总点数

描述：获取指定的托盘的目标垛点的总数。

参数：托盘的名称。

获取指定托盘点位坐标

描述：获取指定的托盘的指定点位坐标。

参数：

- 托盘的名称。
- 点位的索引序号，从1开始。

延时执行指令

描述：程序运行到该积木后，会暂停指定的时间后再继续运行。

参数：程序暂停的时间。

运动等待指令

描述：接在运动积木前后，起到延时下发运动指令或运动完成后延时下发下一条指令的作用。

参数：指令延时下发的时间。

获取系统时间

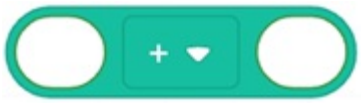
描述：获取系统当前的时间。

返回值：系统当前时间的Unix时间戳，单位：毫秒。

运算积木组

运算积木组用于对变量或常量进行运算。

四则运算



描述：对参数进行四则运算。

参数：

- 两边填写参与运算的变量或常量，可以使用返回值为数值的椭圆形积木或者直接填写。
- 中间选择四则运算符。

返回值：运算结果的数值。

比较运算

描述：对参数进行比较运算。

参数：

- 两边填写参与运算的变量或常量，可以使用返回值为数值的椭圆形积木或者直接填写。
- 中间选择比较运算符。

返回值：比较结果为真时返回true，为假时返回false。

与运算

描述：对参数进行与运算。

参数：两边填写参与运算的变量，使用其他六边形积木。

返回值：两个参数都为真时返回true，任一个为假返回false。

或运算

描述：对参数进行或运算。

参数：两边填写参与运算的变量，使用其他六边形积木。

返回值：两个参数任一个为真时返回true，都为假时false。

非运算

描述：对参数进行非运算。

参数：填写参与运算的变量，使用其他六边形积木。

返回值：参数为真时返回false，为假时true。

取余运算

描述：对参数进行取余运算。

参数：两边填写参与运算的变量或常量，可以使用返回值为数值的椭圆形积木或者直接填写。

返回值：运算结果的数值。

四舍五入运算

描述：对参数进行四舍五入运算。

参数：填写参与运算的变量或常量，可以使用返回值为数值的椭圆形积木或者直接填写。

返回值：运算结果的数值。

单值运算

描述：对参数进行各种单值运算。

参数：

- 选择运算方式。
    - 绝对值
    - 向下取整
    - 向上取整
    - 平方根
    - sin
    - cos
    - tan
    - asin
    - acos
    - atan
    - ln
    - loh
    - e^
    - 10^
- 填写参与运算的变量或常量，可以使用返回值为数值的椭圆形积木或者直接填写。

返回值：运算结果的数值。

打印

描述：将参数输出到控制台查看，主要用于调试。

参数：

- 选择同步打印还是异步打印。同步打印会等已下发的全部指令执行完毕后再打印信息，异步打印则程序执行到该积木时立即打印信息。
- 要输出到控制台的变量或常量，可以使用其他椭圆形积木或者直接填写。

字符积木组

字符积木组包含了字符串和数组的常用功能。。

获取字符串的第n个字符

描述：获取作为变量的字符串的第n个字符。

参数：

- 第一个参数填写字符串，可以使用其他椭圆形积木或者直接填写。
- 第二个参数指定要返回字符串的第几个字符。

返回值：字符串指定位置的字符。

判断字符串一是否包含字符串二

描述：判断第一个参数的字符串中是否包含了第二个参数的字符串。

参数：要判断的两个字符串，可以使用返回值为字符串的椭圆形积木或者直接填写。

返回值：字符串一包含了字符串二时返回true，否则返回false。

拼接两个字符串

描述：将两个字符串拼接成一个字符串，第二个字符串会接在第一个字符串后面。

参数：要拼接的两个字符串，可以使用返回值为字符串的椭圆形积木或者直接填写。

返回值：拼接后的字符串。

获取字符串或数组长度

描述：获取指定的字符串或数组的长度。字符串的长度是指该字符串有多少个字符，数组的长度是指该数组有多少个元素。

参数：要计算长度的字符串或数组，可以使用返回值为字符串或数组的椭圆形积木。

返回值：字符串或数组的长度数值。

比较两个字符串

描述：根据ACSII码比较两个字符串的大小。

参数：要比较的两个字符串，可以使用返回值为字符串的椭圆形积木或者直接填写。

返回值：字符串一和字符串二相等时返回0，字符串一小于字符串二时返回-1，字符串一大于字符串二时返回1。

将数组转换为字符串

描述：将指定数组转换为字符串，字符串中不同数组元素使用指定分隔符分隔。例如数组为{1,2,3}，分隔符为|，则转换后的字符串为“1|2|3”。

参数：

- 要转成字符串的数组，使用返回值为数组的椭圆形积木。
- 转换使用的分隔符。

返回值：转换后的字符串。

将字符串转换为数组

描述：将指定字符串转换为数组，根据指定的分隔符对字符串进行分隔。例如字符串为“1|2|3”，分隔符为|，则转换后的数组为{[1]=1,[2]=2,[3]=3}。

参数：

- 要转成数组的字符串，可以使用返回值为字符串的椭圆形积木或直接输入。
- 转换使用的分隔符。

返回值：转换后的数组。

获取数组指定下标的元素

描述：获取指定数组中指定下标位置的元素。下标表示元素在数组中的位置，例如数组{7,8,9}中8的下标就是2。

参数：

- 目标数组，使用返回值为数组的椭圆形积木。
- 指定元素的下标。

返回值：数组指定位置的元素的值。

获取数组中多个指定元素

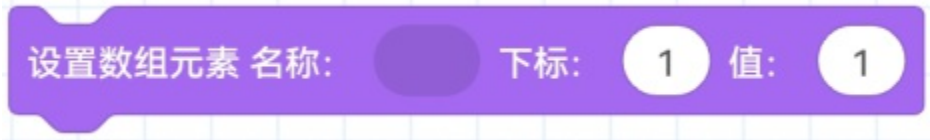
描述：获取指定数组中多个指定下标位置的元素。在开始下标和结束下标的范围内，根据步进值获取元素。

参数：

- 目标数组，使用返回值为数组的椭圆形积木。
- 通过开始下标和结束下标指定获取元素的范围。
- 步进值用于确定获取元素的频率，1就是全部获取，2就是隔一个元素进行获取，以此类推。

返回值：指定的元素组成的新数组。

设置数组中的指定元素



描述：设置数组中指定下标位置的元素的值。

参数：

- 目标数组，使用返回值为数组的椭圆形积木。
- 指定元素的下标。
- 指定元素的值。

自定义积木组

自定义积木组用于新建和管理自定义积木，以及调用全局变量。

调用全局变量

描述：调用控制软件中设置的全局变量。

参数：选择全局变量的名称。

返回值：全局变量的值。

设置全局变量

描述：设置指定的变量。需要注意的是，设置全局变量和设置自定义变量的积木外形相同，但功能略有不同。

参数：

- 选择要修改的变量的名称。
- 修改后的值，可以直接填写或使用其他椭圆形积木。

新建自定义变量

单击可新建一个自定义变量，变量类型可选择数值或者字符串，变量名称必须以字母开头，不能使用空格等特殊字符。创建至少一个变量后，积木列表中会出现下述自定义变量相关积木。

自定义数值变量

描述：新建的自定义数值变量，默认值为nil，建议赋值后再使用。可以通过变量选择下拉框修改现在选中的变量的名称或删除该变量。

返回值：变量的值。

设置自定义数值变量的值

描述：设置指定数值变量。需要注意的是，设置全局变量和设置自定义变量的积木外形相同，但功能略有不同。

参数：

- 选择要修改的变量的名称。
- 修改后的值，可以直接填写或使用其他椭圆形积木。

增减自定义数值变量的值

描述：将指定数值变量增加指定的值。

参数：

- 选择要修改的变量的名称。
- 要增加的值，可以直接填写或使用其他椭圆形积木。设置为负数时可实现减少值。

自定义字符串变量

描述：新建的自定义字符串变量，默认值为nil，建议赋值后再使用。可以通过变量选择下拉框修改现在选中的变量的名称或删除该变量。

返回值：变量的值。

设置自定义字符串变量的值

描述：设置指定字符串变量。

参数：

- 选择要修改的变量的名称。
- 修改后的值，直接输入字符串。

建立一个数组

单击可新建一个自定义数组，数组名称必须以字母开头，不能使用空格等特殊字符。创建至少一个数组后，积木列表中会出现下述数组相关积木。

自定义数组

描述：新建的自定义数组，默认为空数组，建议赋值后再使用。在积木列表中右键（PC端）/长按（移动端）积木可以修改数组的名称或删除数组。也可以通过其他数组积木中数组选择下拉框修改现在选中的数组的名称或删除该数组。数组积木前的勾选框暂无作用，可忽略。

返回值：数组的值。

将变量加入数组

描述：将变量加入指定数组。新加入的变量会成为数组的最后一项。

参数：

- 要添加的变量，可以直接填写或使用其他椭圆形积木。
- 选择要修改的数组。

删除数组指定项

描述：删除指定数组的指定项。

参数：

- 选择要修改的数组。
- 要删除的项的编号，可以直接填写或使用其他返回值为数值的椭圆形积木。

删除数组全部项

描述：删除指定数组的全部项。

参数：选择要修改的数组。

在数组中插入项

描述：在数组中的指定位置插入变量。

参数：

- 选择要修改的数组。
- 插入的位置，可以直接填写或使用其他返回值为数值的椭圆形积木。
- 要添加的变量，可以直接填写或使用其他椭圆形积木。

替换数组中的指定项

描述：将数组中的指定项替换为指定变量。

参数：

- 选择要修改的数组。
- 要替换的项的编号，可以直接填写或使用其他返回值为数值的椭圆形积木。
- 替换后的变量，可以直接填写或使用其他椭圆形积木。

获取数组中的指定项

描述：获取数组中指定项的值。

参数：

- 选择要获取项的数组。
- 要获取的项的编号，可以直接填写或使用其他返回值为数值的椭圆形积木。

返回值：指定项的值。

获取数组中的总项数

描述：获取数组中项目的总数。

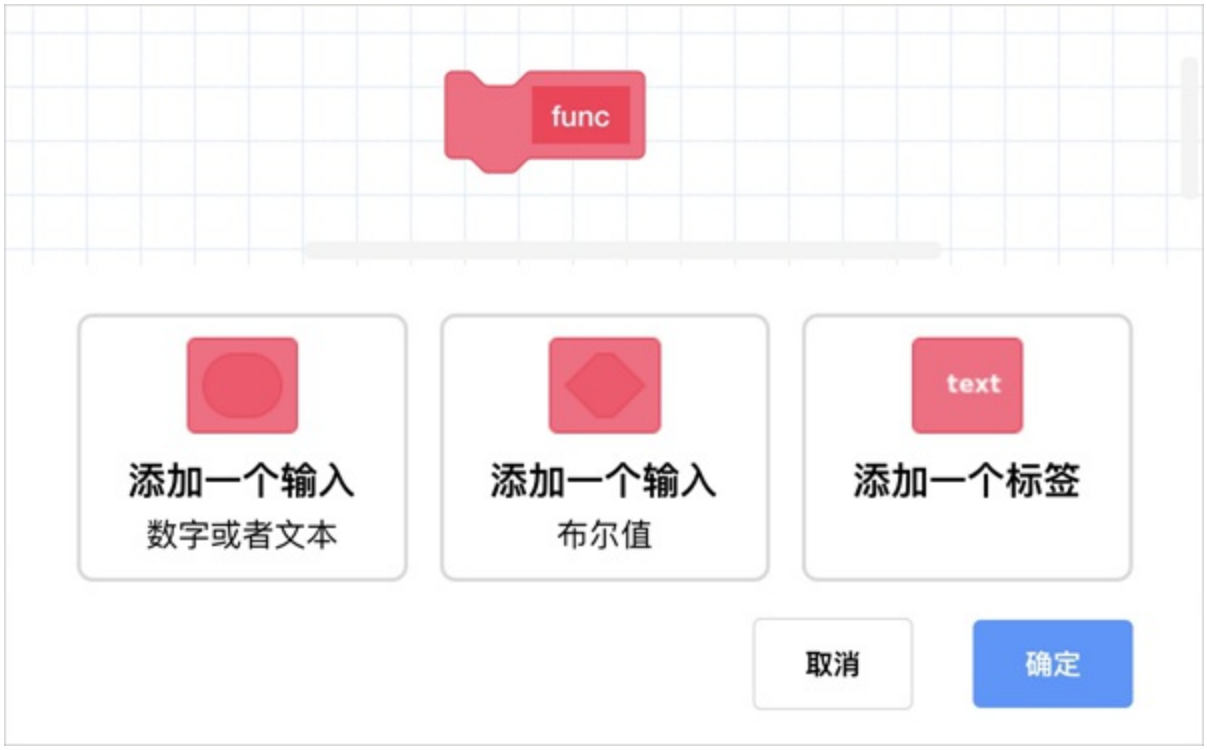
参数：选择要获取项的数组。

返回值：指定数组的项目总数。

新建一个函数

单击可新建一个函数。函数是一个固定的程序段，您可以将一组实现特定常用功能的积木定义为一个函数，之后每次要用到该功能时只需调用这个函数，不需要再重复搭建相同积木组。新建一个函数需要声明并定义该函数。新建函数成功后，积木列表中会出现对应的函数积木。

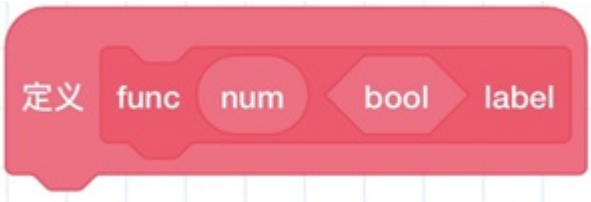
1. **声明函数**



在这个界面，需要定义函数的名称，输入（参数）的类型、数量与名称。函数和参数名称不能包含空格等特殊字符。您还可以给函数添加标签，标签可以作为注释使用，对函数或输入进行补充说明。

1. **定义函数**

完成函数声明后，编程区域内会出现该函数的定义头积木。



您需要在该积木下方接续积木进行编程，定义该函数的功能。

定义头中的输入可拖出来在下方的积木中使用，表示使用实际调用该函数时的输入作为参数。

自定义函数

描述：用户自定义的函数积木，名称与输入参数由用户自定义，用于调用已定义的函数。在积木列表中右键（PC端）/长按（移动端）积木可以修改该函数的声明。如果需要删除该函数，请删除该函数的定义头积木。

新建一个子程序

单击可新建一个子程序。图形编程支持嵌套调用子程序，子程序支持图形编程和脚本编程，最大嵌套两层。新建子程序成功后，积木列表中会出现对应的子程序积木。

- 选择积木编程后页面变为子程序积木编程页面，可以设置子程序描述和编写子程序。

- 选择脚本编程后弹出子程序脚本编程窗口，可以设置子程序描述和编写子程序。

子程序

- 图形化编程子程序

- 脚本编程子程序

描述：子程序积木，描述由用户在创建子程序时定义，用于调用已保存的子程序。在积木列表中右键（PC端）/长按（移动端）积木可以修改或删除该子程序。

IO积木组

IO积木组用于管理机械臂IO端子的输入输出。输入输出端口的取值范围由机械臂的对应端子数量决定，请参考对应机械臂的硬件手册。

设置数字输出

描述：设置指定DO的开启或关闭。

参数：

- 选择DO端子的位置，包括控制柜和末端。
- 选择DO端子的编号。
- 选择要输出的状态（开或者关）。

设置数字输出（子程序用）

描述：设置指定DO的开启或关闭。在子程序中设置DO请使用此积木。

参数：

- 选择DO端子的位置，包括控制柜和末端。
- 选择DO端子的编号。
- 选择要输出的状态（开或者关）。

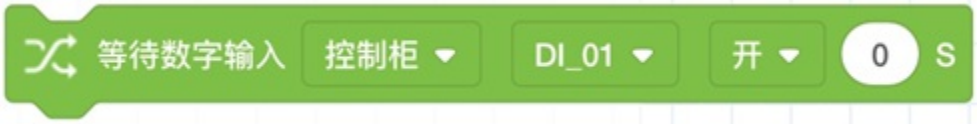
设置一组数字输出

描述：设置一组DO。将积木拖进编程区后单击进行设置。

参数：

- 通过“+”或“-”可以增加或减少要设置的DO数量。
- 选择DO端子的编号。
- 选择要输出的状态（开或者关）。

等待数字输入



描述：等待指定的DI满足条件或等待超时后再执行后续的积木指令。

参数：

- 选择DI端子的位置，包括控制柜和末端。
- 选择DI端子的编号。
- 选择要等待的状态（开或者关）。
- 等待超时时间，如果设置为0，则一直等到满足条件。

设置模拟输出

描述：设置指定模拟输出的值。

参数:

- 选择模拟输出端子的编号。
- 要输出的值，可以直接填写或使用其他返回值为数值的椭圆形积木。

判断数字输入状态

描述：判断指定的DI的当前状态是否满足条件。

参数:

- 选择DI端子的位置，包括控制柜和末端。
- 选择DI端子的编号。
- 选择要判断为真的状态。

返回值：指定DI的当前状态满足条件时，返回true，否则返回false。

获取模拟输入

描述：获取指定模拟输入的值。

参数:

- 选择模拟输入端子的位置，包括控制柜和末端。
- 选择模拟输入端子的编号。

返回值：指定模拟输入的值。

运动积木组

运动积木组用于控制机械臂运动及进行运动相关设置。

控制运动的积木均为异步指令，即下发命令成功后就会执行下一条指令，不会等待机器人运动完成，机器人会按照下发顺序逐条执行指令。如果需要等待已下发的指令都执行完成后再执行后续指令，可使用同步指令。

点位参数可以在该工程的“示教点”页面添加后在此处选择；控制运动的积木中还支持将默认的变量积木拖出来，使用其他的返回值为点位笛卡尔坐标值的椭圆形积木替代。

高级配置

当预设的运动积木无法满足编程需求时，可通过高级配置方式创建控制机器人运动的积木。创建成功的积木会直接出现在编程区域。详情请参考运动高级配置。

运动至目标点

描述：控制机械臂从当前位置运动到指定点。积木拖动到编程区后，双击可进行高级配置，详情请参考运动高级配置。

参数：

- 选择运动方式，支持关节运动和直线运动。关节运动的轨迹非直线，所有关节会同时完成运动。
- 目标点。

运动至目标点（带偏移）

描述：控制机械臂从当前位置运动到指定点偏移后的坐标，目标点可设置为当前点。

参数：

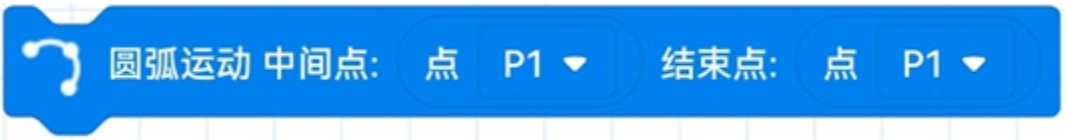
- 选择运动方式，支持相对关节运动和相对直线运动。
- 目标点。
- 笛卡尔坐标系下相对于目标点的X轴、Y轴、Z轴方向上的偏移量，单位：毫米。

关节偏移运动

描述：控制机械臂关节从当前位置运动指定的偏移量。

参数：各关节的偏移量，单位：度。

进行圆弧运动



描述：控制机械臂从当前位置以圆弧插补方式运动至笛卡尔坐标系下的指定点。当前位置的坐标必须不在中间点和结束点确定的直线上。

参数：

- 中间点指用于确定圆弧的中间点。
- 结束点为目标点。

进行整圆运动

描述：控制机械臂从当前位置进行整圆插补运动，运动指定圈数后重新回到当前位置。当前位置的坐标必须不在中间点和结束点确定的直线上。

参数：

- 中间点指用于确定整圆的中间点。
- 结束点指用于确定整圆的结束点。
- 输入进行整圆运动的圈数，取值范围1~999。

轨迹复现

描述：控制机械臂进行轨迹复现，复现的轨迹文件要在轨迹复现工艺中录制。

参数：

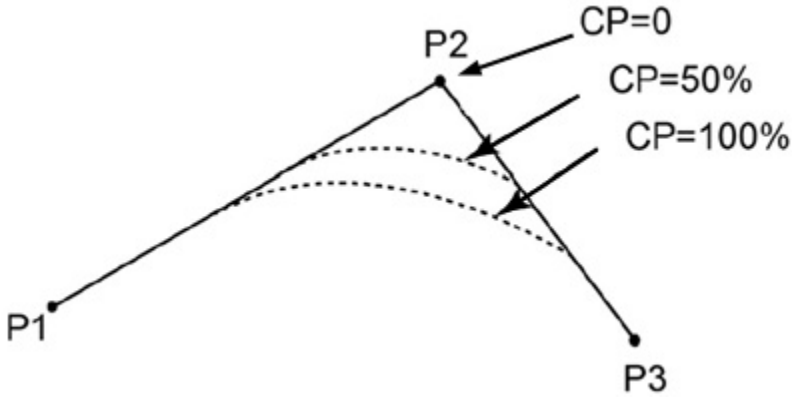
- 选择要复现的轨迹文件。
- 选择复现时的运动速度：
  - 匀速
  - 0.25倍速
  - 0.5倍速
  - 1倍速
  - 2倍速

同步指令

描述：程序运行到这条指令时，会等待机械臂将之前已下发的指令全部执行完成，再继续执行后续指令。

设置平滑过渡比例

描述：设置运动中的平滑过渡比例，即机械臂从起始点经过中间点到达终点时，经过中间点是以直角方式过渡还是以曲线方式过渡，如下图所示。



参数：平滑过渡比例，取值范围：0~100。

设置关节速度比例

描述：设置关节运动的速度比例。

参数：关节速度比例，取值范围：0~100。机械臂实际运动速度比例为参数设置的比例乘以控制软件再现设置中的值再乘以全局速率。

设置关节加速度比例

描述：设置关节运动的加速度比例。

参数：关节加速度比例，取值范围：0~100。机械臂实际运动加速度比例为参数设置的比例乘以控制软件再现设置中的值再乘以全局速率。

设置直线速度比例

描述：设置直线和弧线运动的速度比例。

参数：直线和弧线速度比例，取值范围：0~100。机械臂实际运动速度比例为参数设置的比例乘以控制软件再现设置中的值再乘以全局速率。

设置直线加速度比例

描述：设置直线和弧线运动的加速度比例。

参数：直线和弧线加速度比例，取值范围：0~100。机械臂实际运动加速度比例为参数设置的比例乘以控制软件再现设置中的值再乘以全局速率。

修改指定点的坐标值

描述：修改指定点的指定笛卡尔坐标维度的值。

参数：

- 选择要修改的点。
- 选择要修改的坐标维度。
- 选择修改后的值。

获取指定点的坐标值

描述：获取指定点的笛卡尔坐标值。

参数：选择要获取坐标值的点。

返回值：指定点的笛卡尔坐标值。

获取指定点的指定坐标维度的值

描述：获取指定点的指定笛卡尔坐标维度的值。

参数：

- 选择要获取坐标值的点。
- 选择要获取的坐标维度。

返回值：指定点的指定笛卡尔坐标维度的值。

获取当前位置的指定坐标维度的值

描述：获取TCP当前位置在指定坐标系下的指定坐标维度的值。

参数：

- 选择要获取的坐标维度。

- 选择用户坐标系和工具坐标系，返回的坐标值会换算为对应坐标系下的值。

返回值：TCP当前位置在指定坐标系下的指定坐标维度的值。

运动高级配置

通过高级配置方式创建控制机器人运动的积木。配置的内容包括积木名称，运动方式和运动参数。不同的运动方式需配置的运动参数不同。其中，机械臂实际运动速度/加速度比例为参数设置的比例乘以控制软件再现设置中的值再乘以全局速率。

MovJ

运动方式：从当前位置以关节插补方式运动至笛卡尔坐标系下的目标位置。

基础设置：P点坐标，即目标点的坐标。可在该工程的“示教点”页面添加后在此处选择，也可直接在此页面示教进行自定义。

高级设置：

勾选并配置需要设置的高级参数。

- 速度(Speed)：运动速度比例，取值范围：1~100。
- 加速度(Accel)：运动加速度比例，取值范围：1~100。
- 平滑过渡(CP)：运动中的平滑过渡比例，详见本章最后的“平滑过渡（CP）说明”。取值范围：0~100。
- 过程I/O设置：运动到指定距离或百分比时，触发指定DO。距离为正时，表示离起点的距离，距离为负时，表示离目标点的距离。单击下方的“+”可以添加过程IO，单击右侧的“-“可以删除对应过程IO。

MovL

运动方式：从当前位置以直线插补方式运动至笛卡尔坐标系下的目标位置。

基础设置：P点坐标，即目标点的坐标。可在该工程的“示教点”页面添加后在此处选择，也可直接在此页面示教进行自定义。

高级设置：

勾选并配置需要设置的高级参数。

- 速度(Speed)：运动速度比例，取值范围：1~100。
- 加速度(Accel)：运动加速度比例，取值范围：1~100。
- 平滑过渡(CP)：运动中的平滑过渡比例，详见本章最后的“平滑过渡（CP）说明”。取值范围：0~100。
- 过程I/O设置：运动到指定距离或百分比时，触发指定DO。距离为正时，表示离起点的距离，距离为负时，表示离目标点的距离。单击下方的“+”可以添加过程IO，单击右侧的“-“可以删除对应过程IO。

JointMovJ

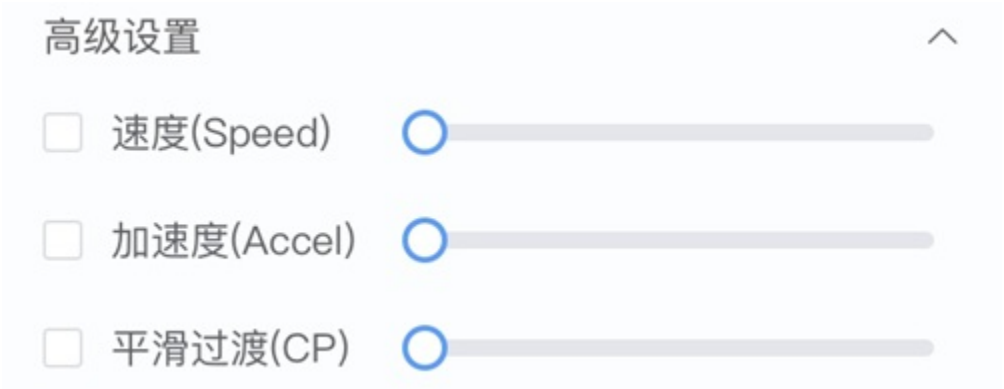
运动方式：从当前位置以关节插补方式运动至目标关节角度。

基础设置：目标关节角度，通过示教进行自定义。

高级设置：

勾选并配置需要设置的高级参数。

- 速度(Speed)：运动速度比例，取值范围：1~100。
- 加速度(Accel)：运动加速度比例，取值范围：1~100。
- 平滑过渡(CP)：运动中的平滑过渡比例，详见本章最后的“平滑过渡（CP）说明”。取值范围：0~100。



RelMovJ

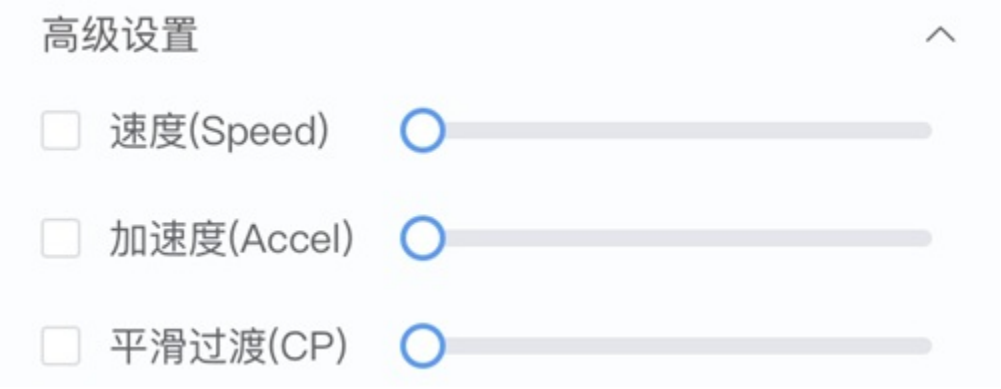
运动方式：从当前位置以关节插补方式运动至笛卡尔坐标系下的偏移位置。

基础设置：笛卡尔坐标系下X轴、Y轴、Z轴方向上的偏移量，单位：毫米。

高级设置：

勾选并配置需要设置的高级参数。

- 速度(Speed)：运动速度比例，取值范围：1~100。
- 加速度(Accel)：运动加速度比例，取值范围：1~100。
- 平滑过渡(CP)：运动中的平滑过渡比例，详见本章最后的“平滑过渡（CP）说明”。取值范围：0~100。



RelMovL

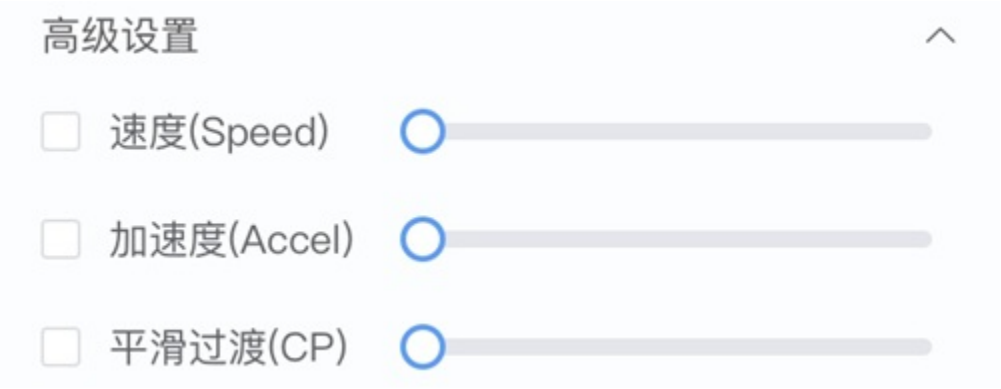
运动方式：从当前位置以直线插补方式运动至笛卡尔坐标系下的偏移位置。

基础设置：笛卡尔坐标系下X轴、Y轴、Z轴方向上的偏移量，单位：毫米。

高级设置：

勾选并配置需要设置的高级参数。

- 速度(Speed)：运动速度比例，取值范围：1~100。
- 加速度(Accel)：运动加速度比例，取值范围：1~100。
- 平滑过渡(CP)：运动中的平滑过渡比例，详见本章最后的“平滑过渡（CP）说明”。取值范围：0~100。

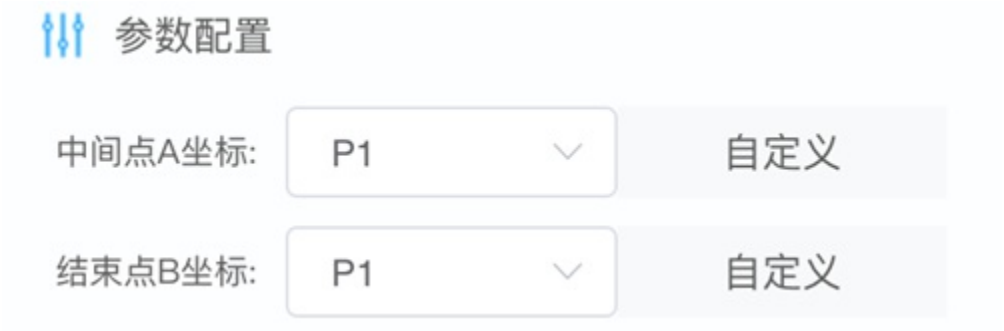


Arc

运动方式：从当前位置以圆弧插补方式运动至笛卡尔坐标系下的指定点。当前位置的坐标必须不在AB点确定的直线上。

基础设置：

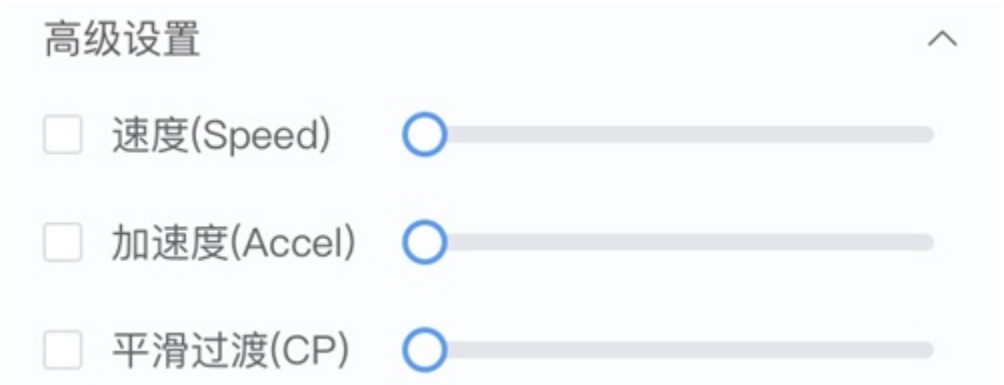
- 中间点A坐标：圆弧中间点的坐标。
- 结束点B坐标：目标点的坐标。两个点都可在该工程的“示教点”页面添加后在此处选择，或者直接在此页面示教进行自定义。



高级设置：

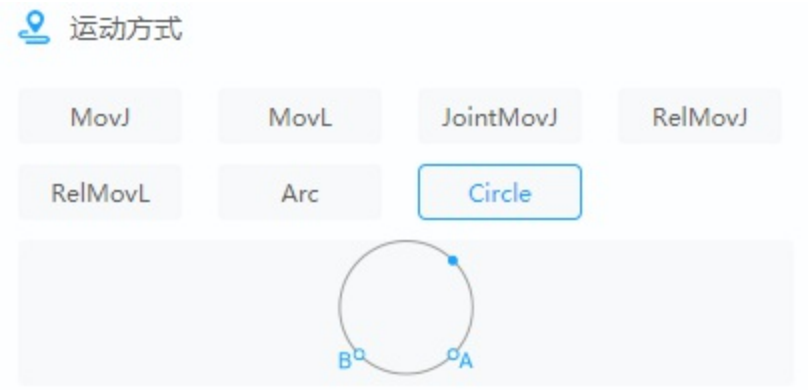
勾选并配置需要设置的高级参数。

- 速度(Speed)：运动速度比例，取值范围：1~100。
- 加速度(Accel)：运动加速度比例，取值范围：1~100。
- 平滑过渡(CP)：运动中的平滑过渡比例，详见本章最后的“平滑过渡（CP）说明”。取值范围：0~100。



Circle

运动方式：从当前位置进行整圆插补运动，运动指定圈数后重新回到当前位置。当前位置的坐标必须不在AB点确定的直线上，且三个点确定的整圆不能超出机械臂的运动范围。



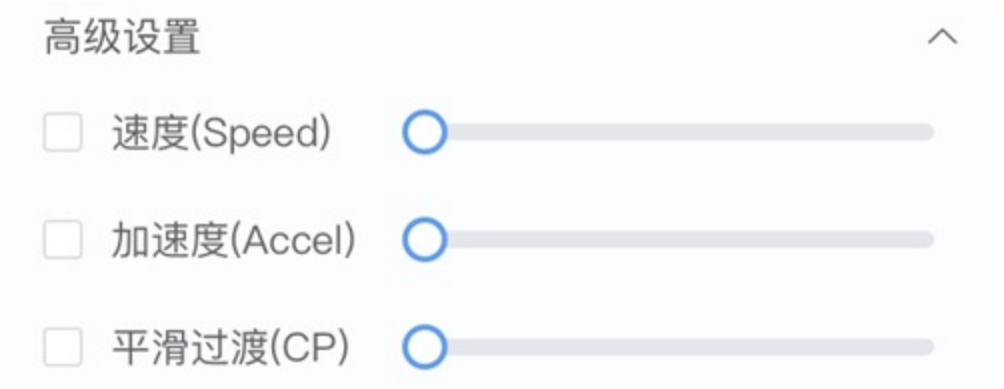
基础设置：

- 中间点A坐标：用于确定整圆的中间点的坐标。
- 结束点B坐标：用于确定整圆的结束点的坐标。两个点都可在该工程的“示教点”页面添加后在此处选择，或者直接在此页面示教进行自定义。
- 循环次数：进行整圆运动的圈数，取值范围1~999。

高级设置：

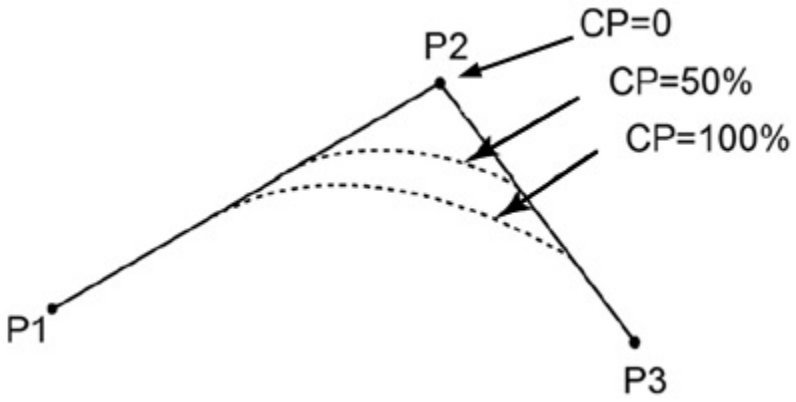
勾选并配置需要设置的高级参数。

- 速度(Speed)：运动速度比例，取值范围：1~100。
- 加速度(Accel)：运动加速度比例，取值范围：1~100。
- 平滑过渡(CP)：运动中的平滑过渡比例，详见本章最后的“平滑过渡（CP）说明”。取值范围：0~100。



平滑过渡（CP）说明

平滑过渡指机械臂从起始点经过中间点到达终点时，经过中间点是以直角方式过渡还是以曲线方式过渡，如下图所示。



Modbus积木组

Modbus积木组用于进行Modbus通信相关操作。

创建Modbus主站

描述：创建Modbus的主站，并和从站建立连接。

参数：

- Modbus从站的IP地址。
- Modbus从站的端口。
- Modbus从站的ID，取值范围：1 ~ 4。

获取创建主站结果

描述：获取创建Modbus主站的结果。

返回值：创建成功返回0，创建失败返回1。

等待输入寄存器

描述：等待输入寄存器指定地址的值满足条件后再继续执行下一条指令。

参数：

- 输入寄存器起始地址，取值范围：0 ~ 4095。
- 选择要读取的数据类型：
  - U16：16位无符号整数（2个字节，占用1个寄存器）
  - U32：32位无符号整数（4个字节，占用2个寄存器）
  - F32：32位单精度浮点数（4个字节，占用2个寄存器）
  - F64：64位双精度浮点数（8个字节，占用4个寄存器）

- 输入寄存器指定地址的值要满足的条件。

等待保持寄存器

描述：等待保持寄存器指定地址的值满足条件后再继续执行下一条指令。

参数：

- 保持寄存器起始地址，取值范围：0 ~ 4095。
- 选择要读取的数据类型：
  - U16：16位无符号整数（2个字节，占用1个寄存器）
  - U32：32位无符号整数（4个字节，占用2个寄存器）
  - F32：32位单精度浮点数（4个字节，占用2个寄存器）
  - F64：64位双精度浮点数（8个字节，占用4个寄存器）
- 输入寄存器指定地址的值要满足的条件。

等待触点寄存器

描述：等待触点寄存器指定地址的值满足条件后再继续执行下一条指令。

参数：

- 触点寄存器起始地址，取值范围：0 ~ 4095。
- 触点寄存器指定地址的值要满足的条件。

等待线圈寄存器

描述：等待线圈寄存器指定地址的值满足条件后再继续执行下一条指令。

参数：

- 线圈寄存器起始地址，取值范围：0 ~ 4095。
- 线圈寄存器指定地址的值要满足的条件。

读取输入寄存器

描述：获取输入寄存器指定地址的值。

参数：

- 输入寄存器起始地址，取值范围：0 ~ 4095。
- 选择要读取的数据类型：
  - U16：16位无符号整数（2个字节，占用1个寄存器）
  - U32：32位无符号整数（4个字节，占用2个寄存器）
  - F32：32位单精度浮点数（4个字节，占用2个寄存器）
  - F64：64位双精度浮点数（8个字节，占用4个寄存器）

返回值：输入寄存器指定地址的值。

读取保持寄存器

描述：获取保持寄存器指定地址的值。

参数：

- 保持寄存器起始地址，取值范围：0 ~ 4095。
- 选择要读取的数据类型：
  - U16：16位无符号整数（2个字节，占用1个寄存器）
  - U32：32位无符号整数（4个字节，占用2个寄存器）
  - F32：32位单精度浮点数（4个字节，占用2个寄存器）
  - F64：64位双精度浮点数（8个字节，占用4个寄存器）

返回值：输入寄存器指定地址的值。

读取触点寄存器

描述：获取触点寄存器指定地址的值。

参数：触点寄存器起始地址，取值范围：0 ~ 4095。

返回值：触点寄存器指定地址的值。

读取线圈寄存器

描述：获取线圈寄存器指定地址的值。

参数：线圈寄存器起始地址，取值范围：0 ~ 4095。

返回值：线圈寄存器指定地址的值。

连续读取线圈寄存器

描述：连续读取线圈寄存器指定地址的值。

参数：

- 线圈寄存器起始地址，取值范围：0 ~ 4095。
- 连续读取的寄存器位数。

返回值：线圈寄存器指定地址的值，存储在table中。table中第一个值对应线圈寄存器起始地址的值。

连续读取保持寄存器

描述：连续读取保持寄存器指定地址的值。

参数：

- 保持寄存器起始地址，取值范围：0 ~ 4095。
- 连续读取的值的数量。
- 选择要读取的数据类型：
  - U16：16位无符号整数（2个字节，占用1个寄存器）
  - U32：32位无符号整数（4个字节，占用2个寄存器）
  - F32：32位单精度浮点数（4个字节，占用2个寄存器）
  - F64：64位双精度浮点数（8个字节，占用4个寄存器）

返回值：线圈寄存器指定地址的值，存储在table中。table中第一个值对应线圈寄存器起始地址的值。

写入线圈寄存器

描述：将指定的值写入线圈寄存器指定的地址。

参数：

- 线圈寄存器起始地址，取值范围：6 ~ 4095。
- 选择写入的值，只能是0或1。

连续写入线圈寄存器

描述：将指定的值连续写入线圈寄存器指定的地址。

参数：

- 线圈寄存器起始地址，取值范围：0 ~ 4095。
- 要写入的值的位数。
- 输入写入的值，填写一个数组，长度与写入的位数相同，每一位只能是0或者1。

写入保持寄存器

描述：将指定的值写入保持寄存器指定的地址。

参数：

- 保持寄存器起始地址，取值范围：0 ~ 4095。
- 选择写入的值，需要与选择的数据类型对应。
- 选择要写入的数据类型：
  - U16：16位无符号整数（2个字节，占用1个寄存器）
  - U32：32位无符号整数（4个字节，占用2个寄存器）
  - F32：32位单精度浮点数（4个字节，占用2个寄存器）
  - F64：64位双精度浮点数（8个字节，占用4个寄存器）

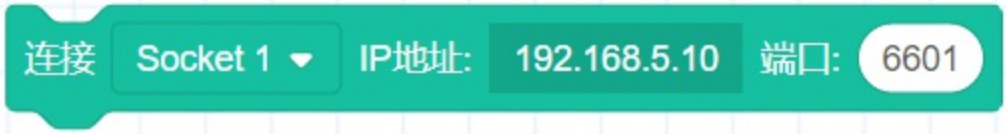
关闭主站

描述：关闭Modbus主站，断开与所有从站的连接。

TCP积木组

TCP积木组用于进行TCP通信相关的操作。

连接SOCKET



描述：创建TCP客户端，和指定的TCP服务端进行通信。

参数：

- 选择SOCKET编号，最多建立4条TCP通信链路。
- TCP服务端的IP地址。
- TCP服务端的端口。

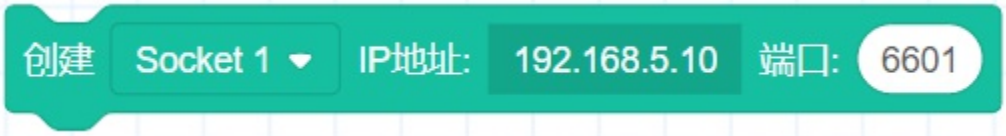
获取连接SOCKET结果

描述：获取TCP通信连接结果。

参数：选择SOCKET编号。

返回值：连接成功返回0，连接失败返回1。

创建SOCKET



描述：创建TCP服务端，等待客户端进行连接。

参数：

- 选择SOCKET编号，最多建立4条TCP通信链路。
- TCP服务端的IP地址。
- TCP服务端的端口。机械臂作为服务端时，请勿使用下述已被系统占用的端口：

22, 23, 502（0~1024之间的端口是linux系统自定义端口，被占用可能性高，请尽量避免使用），

5000~5004, 6000, 8080, 11000, 11740, 22000, 22002, 29999, 30003, 30004, 60000, 65500~65515

获取创建SOCKET结果

描述：获取TCP服务端创建结果。

参数：选择SOCKET编号。

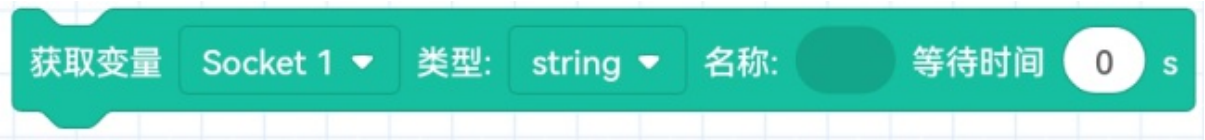
返回值：创建成功返回0，创建失败返回1。

关闭SOCKET

描述：关闭指定的SOCKET，断开该通信链路。

参数：选择SOCKET编号。

获取变量



描述：通过TCP通信获取变量并保存。

参数：

- 选择SOCKET编号。
- 选择要接收的变量的类型，支持字符串和数值。
- 用于保存接收到的数据的变量，使用已创建的变量积木。
- 填写等待超时时间，如果设置为0，则不会超时，会一直等待直到接收到变量。

发送变量

描述：通过TCP通信发送变量。

参数：

- 选择SOCKET编号。
- 发送的数据，可以直接填写或者使用其他的返回值为字符串或数值的椭圆形积木。

获取发送变量结果

描述：获取TCP发送变量的结果。

参数：选择SOCKET编号。

返回值：发送成功返回0，发送失败返回1。

附录C 脚本编程函数说明

Lua基础语法

变量与数据类型

> 如果您想系统学习Lua编程相关知识，请自行在网络上搜索Lua教程。本手册仅列举部分Lua基础语法，便于您快速查阅。

变量用于存储值，将值作为参数传递或结果返回。变量通过"="进行赋值。

Lua中的变量默认为全局变量，除非使用 local 显式声明为局部变量。局部变量的作用域为从声明位置开始到所在语句块结束。

```
a = 5               -- 全局变量
local b = 5         -- 局部变量
```

变量名称可以是任意非数字打头的字母、下划线和数字组成的字符串，Lua保留的关键字除外。

Lua 变量不要类型定义，只需要为变量赋值，Lua会根据值自动判断变量的类型。

Lua支持多种数据类型，较为常见的包括数字（number），布尔值（boolean），字符串（string）和表（table）。Lua中的数组是table的一种。

Lua中还有一种特殊的数据类型为nil，nil 表示空（没有任何有效值），例如打印一个没有赋值的变量，便会输出一个 nil 值。

数字

Lua中的number为双精度类型的实浮点数，支持各种运算，以下写法均被视为number：

- 2
- 2.2
- 0.2
- 2e+1
- 0.2e-1
- 7.8263692594256e-06

布尔值

boolean 类型只有两个可选值：true（真） 和 false（假），Lua 把 false 和 nil 看作是 false，其他的都为 true，数字 0 也是 true。

字符串

string是由数字、字母、下划线组成的一串字符。string可以使用以下三种方式来表示：

- 单引号间的一串字符。
- 双引号间的一串字符。
- **[[** 与 **]]** 间的一串字符。

在对一个数字字符串上进行算术操作时，Lua 会尝试将这个数字字符串转成一个数字。

Lua 提供了很多的方法来支持字符串的操作：

方法	说明
string.upper (argument)	字符串全部转为大写字母
string.lower (argument)	字符串全部转为小写字母
string.gsub(mainString, findString,replaceString,num)	在字符串中替换。mainString 为要操作的字符串，findString 为被替换的字符，replaceString 要替换的字符，num 替换次数（可以忽略，则全部替换）
string.find (str, substr, [init, [end]])	在一个指定的目标字符串 str 中搜索指定的内容 substr，如果找到了一个匹配的子串，就会返回这个子串的起始索引和结束索引，不存在则返回 nil。
string.reverse(arg)	字符串反转
string.format(...)	返回一个类似printf的格式化字符串
string.char(arg) 和 string.byte(arg[,int])	char 将整型数字转成字符并连接， byte 转换字符为整数值(可以指定某个字符，默认第一个字符)
string.len(arg)	计算字符串长度
string.rep(string, n)	返回字符串string的n个拷贝
..	链接两个字符串
string.gmatch(str, pattern)	回一个迭代器函数，每一次调用这个函数，返回一个在字符串 str 找到的下一个符合 pattern 描述的子串。如果参数 pattern 描述的字符串没有找到，迭代函数返回nil
string.match(str, pattern, init)	string.match()只寻找源字串str中的第一个配对. 参数init可选, 指定搜寻过程的起点, 默认为1。 在成功配对时, 函数将返回配对表达式中的所有捕获结果; 如果没有设置捕获标记, 则返回整个配对字符串. 当没有成功的配对时, 返回nil
string.sub(s, i [, j])	用于截取字符串。s为要截取的字符串，i为截取开始位置，j为截取结束位置，默认为-1，最后一个字符。

示例：

```
str = "Lua"
print(string.upper(str))        --字符串全部转为大写字母，打印结果：LUA
print(string.lower(str))        --字符串全部转为小写字母，打印结果：lua
print(string.reverse(str))     --字符串反转，打印结果：aul
print(string.len("abc"))        --计算字符串abc的长度，打印结果：3
print(string.format("the value is: %d",4))      --打印结果：the value is:4
print(string.rep(str,2))        --字符串复制2次，打印结果：LuaLua
  string1 = "cn."
  string2 = "dobot"
```

```
  string3 = ".cc"
  print("连接字符串",string1..string2..string3)  -- 使用..进行字符串连接，打印结果：cn.dobot.cc

  string1 = [[aaaa]]
  print(string.gsub(string1,"a","z",3))               --在字符串中替换，打印结果：zzza

  print(string.find("Hello Lua user", "Lua", 1))        --在字符串中搜索Lua，返回子串的起始索引和
结束索引，打印结果：7，9

  sourcestr = "prefix--runoobgoogletaobao--suffix"
  sub = string.sub(sourcestr, 1, 8)                   --取字符串前缀，第1个到第8个
  print("\n截取", string.format("%q", sub))            --打印结果：截取 "prefix--"
```

表

table是一组带索引的数据。

- 最简单的构造函数是{}，用来创建一个空表。可以直接初始化表。
- table实际为关联型数组，可以用任意类型的值来作数组的索引，但这个值不能是 nil。
- table是不固定大小的，可以根据自己需要进行扩容。
- 通过#符号可以获取表的长度。

```
tbl = {[1] = 2, [2] = 6, [3] = 34, [4] =5}
print("tbl 长度 ", #tbl)  -- 打印结果为4
```

Lua 提供了很多的方法来支持表的操作：

方法	说明
table.concat (table [, sep [, start [, end]]])	concat是concatenate(连锁, 连接)的缩写. table.concat()函数列出参数中指定table的数组部分从start位置到end位置的所有元素, 元素间以指定的分隔符(sep)隔开
table.insert (table, [pos,] value)	在table的数组部分指定位置(pos)插入值为value的一个元素. pos参数可选, 默认为表的末尾
table.remove (table [, pos])	返回table数组部分位于pos位置的元素. 其后的元素会被前移. pos参数可选, 默认为table长度, 即从最后一个元素删起
table.sort (table [, comp])	对给定的table进行升序排序

示例1：

```
fruits = {}         -- 初始化表
fruits = {"banana","orange","apple"}  -- 为表赋值

print("连接后的字符串 ",table.concat(fruits,", ", 2,3))    -- 指定索引来连接table，连接后的字符串 o
```

```
range, apple

-- 在末尾插入
table.insert(fruits,"mango")
print("索引为 4 的元素为 ",fruits[4])      --打印结果：索引为4的元素为 mango

-- 在索引为 2 的键处插入
table.insert(fruits,2,"grapes")
print("索引为 2 的元素为 ",fruits[2])      --打印结果：索引为2的元素为 grapes

print("最后一个元素为 ",fruits[5])         --打印结果：最后一个元素为 mango
table.remove(fruits)
print("移除后最后一个元素为 ",fruits[5])    --打印结果：移除后最后一个元素为 nil
```

示例2：

```
fruits = {"banana","orange","apple","grapes"}
print("排序前")
for k,v in ipairs(fruits)
do
        print(k,v)          --打印结果：banana orange apple grapes
end
--进行升序排序
table.sort(fruits)
print("排序后")
for k,v in ipairs(fruits) do
        print(k,v)          --打印结果：apple banana grapes orange
end
```

数组

数组，就是相同数据类型的元素按一定顺序排列的集合，可以是一维数组和多维数组。Lua 数组的索引键值可以使用整数表示，数组的大小不是固定的。

- 一维数组：最简单的数组，其逻辑结构是线性表。
- 多维数组：即数组中包含数组或一维数组的索引键对应一个数组。

示例1：一维数组可以用for循环出数组中的元素。使用整数索引来访问数组元素，如果索引没有值则返回nil。

```
array = {"Lua", "Tutorial"}  --创建一维数组
for i= 0, 2 do
   print(array[i])           --打印结果分别为：nil Lua Tutorial
end
```

在 Lua 索引值是以 1 为起始，但也可以指定 0 开始。除此外还可以以负数为数组索引值。

```
array = {}
for i= -2, 2 do
   array[i] = i*2+1           --为一维数组赋值
```

```
end
for i = -2,2 do
   print(array[i])            --打印结果分别为：-3 -1 1 3 5
end
```

示例2：一个三行三列的阵列多维数组

```
-- 初始化数组
array = {}
for i=1,3 do
 array[i] = {}
    for j=1,3 do
      array[i][j] = i*j
    end
end

-- 访问数组
for i=1,3 do
  for j=1,3 do
     print(array[i][j])     --打印结果分别为：1 2 3 2 4 6 3 6 9
  end
end
```

运算符

算术运算符

指令符号	说明
+	加法运算
-	减法运算
*	乘法运算
/	浮点除法运算
//	向下取整除法运算
%	取余除法运算
^	指数运算
&	按位与运算
\	按位或运算
~	按位异或运算
<<	按位左移运算
>>	按位右移运算

示例：

```
a=20
b=5
print(a+b)            --打印a加b的结果：25
print(a-b)            --打印a减b的结果：15
print(a*b)            --打印a乘b的结果：100
print(a/b)            --打印a除以b的结果：4
print(a//b)            --打印a整除b的结果：4
print(a%b)            --打印a除以b的余数结果：0
print(a^b)            --打印a的b次幂的结果：3200000
print(a&b)            --打印a和b按位与的结果：4
print(a|b)            --打印a和b按位或的结果：21
print(a~b)            --打印a异或b的结果：17
print(a<<b)            --打印a左移b个单位的结果：640
print(a>>b)            --打印a右移b个单位的结果：0
```

关系运算符

指令符号	说明

==	等于
~=	不等于
<=	小于等于
>=	大于等于
<	小于
>	大于

示例：

```
a=20                --创建变量a
b=5                    --创建变量b
print(a==b)            --打印a等于b的对比结果：false
print(a~=b)            --打印a不等于b的对比结果：true
print(a<=b)            --打印a小于等于b的对比结果：false
print(a>=b)            --打印a大于等于b的对比结果：true
print(a<b)           --打印a小于b的对比结果：false
print(a>b)           --打印a大于b的对比结果：true
```

逻辑运算符

指令符号	说明
and	逻辑与，两侧均为true，其结果才为true，只要有一侧为false，其结果即为false
or	逻辑或，一侧结果为true， 其结果即为true，如or两侧均为false，则结果为false
not	逻辑非为将判断结果直接取反

```
a=true
b=false
print(a and b)            --真与假，结果为假
print(a or b)           --真或假，结果为真
print(20 > 5 not true)    --真与非真，等于真与假，最后结果为假
```

流程控制

指令符号	说明
if...then... elseif...then... else...end	if条件判断指令。从上到下依次判断条件是否成立，如果某个判断为true，执行完对应的代码块，后面的条件判断直接忽略，不再执行
while...do...end	while循环控制指令。在条件为 true 时，让程序重复地执行某些语句。执行语句前会先检查条件是否为 true
for...do...end	for循环控制指令，重复执行指定语句，重复次数可在 for 语句中控制
repeat... until()	repeat循环控制指令。重复执行循环，直到 指定的条件为真时为止

示例：

1、if条件判断指令

```
a = 100;
b = 200;
--[ 检查条件 --]
if(a == 100)
then
   --[if条件为true时执行以下if条件判断--]
   if(b == 200)
   then
      --[if条件为true时执行该语句块--]
     print("a 的值为 :", a );
  print("b 的值为 :", b );
   end
end
```

2、while循环控制指令

```
a=10
while( a < 20 )
do
   print("a 的值为:", a)
   a = a+1
end
```

3、for循环控制指令

```
for i=10,1,-1 do
    print(i)
end
```

4、repeat循环控制指令

```
a = 10
repeat
   print("a的值为:", a)
   a = a + 1
until(a > 15)
```

函数说明

运动指令

运动指令函数用于控制机械臂进行运动。运动速度比例/加速度比例还可以在运动参数中设置，如果两处都设置了，以运动指令的值为准。机械臂实际运动速度/加速度比例为参数设置的比例乘以控制软件再现设置中的值再乘以全局速率。

Go

原型：

```
Go(P,"User=1 Tool=2 CP=1 Speed=50 Accel=20 SYNC=1")
```

描述：

从当前位置以点到点（关节运动）方式运动至笛卡尔坐标系下的目标位置。关节运动的轨迹非直线，所有关节会同时完成运动。

必选参数：

P：表示目标点，可在该工程的“示教点”页面添加后在此引用，也可自定义点位，但只支持笛卡尔坐标点位。

可选参数：

- User：用户坐标系索引，需要在设置中添加后在此引用。
- Tool：工具坐标系索引，需要在设置中添加后在此引用。
- CP：运动时设置平滑过渡（详见运动参数中的CP指令），取值范围：0~100。
- Speed：运动速度比例，取值范围：1~100。
- Accel：运动加速度比例，取值范围：1~100。
- SYNC：同步标识，取值范围：0或1。默认值为0。
  SYNC = 0表示异步执行，调用后立即返回，但不关注指令执行情况；
  SYNC = 1表示同步执行，调用后，待指令执行完才返回。

示例：

```
Go(P1)
```

机械臂以默认设置点对点运动至P1点。

Move

原型：

```
Move(P,"User=1 Tool=2 CP=1 SpeedS=50 AccelS=20 SYNC=1")
```

描述:

从当前位置以直线方式运动至笛卡尔坐标系下的目标位置。

必选参数:

P：表示目标点，可在该工程的“示教点”页面添加后在此引用，也可自定义点位，但只支持笛卡尔坐标点位。

可选参数:

- User：用户坐标系索引，需要在设置中添加后在此引用。
- Tool：工具坐标系索引，需要在设置中添加后在此引用。
- CP：运动时设置平滑过渡（详见运动参数中的CP指令），取值范围：0~100。
- SpeedS：运动速度比例，取值范围：1~100。
- AccelS：运动加速度比例，取值范围：1~100。
- SYNC：同步标识，取值范围：0或1。默认值为0。
  SYNC = 0表示异步执行，调用后立即返回，但不关注指令执行情况；
  SYNC = 1表示同步执行，调用后，待指令执行完才返回。

示例:

```
Move(P1)
```

机械臂以默认设置直线运动至P1点。

MoveJ

原型:

```
MoveJ(P,"CP=1 Speed=50 Accel=20 SYNC=1")
```

描述:

从当前位置以点到点（关节运动）方式运动至目标关节角度。

必选参数:

P：表示目标点，只能通过关节角度定义。

可选参数:

- CP：运动时设置平滑过渡（详见运动参数中的CP指令），取值范围：0~100。
- Speed：运动速度比例，取值范围：1~100。

- Accel：运动加速度比例，取值范围：1~100。
- SYNC：同步标识，取值范围：0或1。默认值为0。
  SYNC = 0表示异步执行，调用后立即返回，但不关注指令执行情况；
  SYNC = 1表示同步执行，调用后，待指令执行完才返回。

示例：

```
local P = {joint={0,-0.0674194,0,0,0,0}}
MoveJ(P)
```

自定义关节坐标点P，机械臂以默认设置运动至P点。

Circle3

原型：

```
Circle3(P1,P2,Count,"User=1 Tool=2 CP=1 SpeedS=50 AccelS=20 SYNC=1")
```

描述：

从当前位置进行整圆插补运动，运动指定圈数后重新回到当前位置。需要通过当前位置，P1，P2三个点确定一个整圆，因此当前位置不能在P1和P2确定的直线上，且三个点确定的整圆不能超出机械臂的运动范围。

必选参数：

- P1：表示整圆中间点，可在该工程的“示教点”页面添加后在此引用，也可自定义点位，但只支持笛卡尔坐标点位。
- P2：表示整圆结束点，可在该工程的“示教点”页面添加后在此引用，也可自定义点位，但只支持笛卡尔坐标点位。
- Count：进行整圆运动的圈数，取值范围1~999。

可选参数：

- User：用户坐标系索引，需要在设置中添加后在此引用。
- Tool：工具坐标系索引，需要在设置中添加后在此引用。
- CP：运动时设置平滑过渡（详见运动参数中的CP指令），取值范围：0~100。
- SpeedS：运动速度比例，取值范围：1~100。
- AccelS：运动加速度比例，取值范围：1~100。
- SYNC：同步标识，取值范围：0或1。默认值为0。
  SYNC = 0表示异步执行，调用后立即返回，但不关注指令执行情况；
  SYNC = 1表示同步执行，调用后，待指令执行完才返回。

示例：

```
Go(P1)
```

```
Circle3(P2,P3,1)
```

机械臂运动至P1，然后进行沿着P1，P2，P3确定的整圆运动一圈。

Arc3

原型：

```
Arc3(P1,P2,"User=1 Tool=2 CP=1 SpeedS=50 AccelS=20 SYNC=1")
```

描述：

从当前位置以圆弧插补方式运动至笛卡尔坐标系下的指定点。需要通过当前位置，P1，P2三个点确定一个圆弧，因此当前位置不能在P1和P2确定的直线上。

必选参数：

- P1：表示圆弧中间点，可在该工程的“示教点”页面添加后在此引用，也可自定义点位，但只支持笛卡尔坐标点位。
- P2：表示目标点，可在该工程的“示教点”页面添加后在此引用，也可自定义点位，但只支持笛卡尔坐标点位。

可选参数：

- User：用户坐标系索引，需要在设置中添加后在此引用。
- Tool：工具坐标系索引，需要在设置中添加后在此引用。
- CP：运动时设置平滑过渡（详见运动参数中的CP指令），取值范围：0~100。
- SpeedS：运动速度比例，取值范围：1~100。
- AccelS：运动加速度比例，取值范围：1~100。
- SYNC：同步标识，取值范围：0或1。默认值为0。
  SYNC = 0表示异步执行，调用后立即返回，但不关注指令执行情况；
  SYNC = 1表示同步执行，调用后，待指令执行完才返回。

示例：

```
Go(P1)
Arc3(P2,P3)
```

机械臂运动至P1，然后经由P2圆弧运动至P3。

GoIO

原型：

```
GoIO(P,{ {Mode,Distance,Index,Status},{Mode,Distance,Index,Status}...}, "User=1 Tool=2 CP=1 Sp
```

```
eed=50 Accel=20 SYNC=1")
```

描述:

从当前位置以点到点（关节运动）方式运动至笛卡尔坐标系下的目标位置，运动时并行设置数字输出端口状态。

必选参数:

- P：表示目标点，可在该工程的“示教点”页面添加后在此引用，也可自定义点位，但只支持笛卡尔坐标点位。
- 并行数字输出参数：设置当机械臂运动到指定距离或百分比时，触发指定DO。可设置多组，每组含有以下参数：
  - Mode：触发模式。0表示距离百分比，1表示距离数值。
  - Distance：指定距离。
    - Distance为正数时，表示离起点的距离。
    - Distance为负数时，表示离目标点的距离。
    - Mode为0时，Distance表示和总距离的百分比，取值1~100。
    - Mode为1时，Distance表示距离的值，单位：毫米。
  - Index：DO端子的编号。
  - Status：要设置的DO状态，0表示关，1表示开。

可选参数:

- User：用户坐标系索引，需要在设置中添加后在此引用。
- Tool：工具坐标系索引，需要在设置中添加后在此引用。
- CP：运动时设置平滑过渡（详见运动参数中的CP指令），取值范围：0~100。
- Speed：运动速度比例，取值范围：1~100。
- Accel：运动加速度比例，取值范围：1~100。
- SYNC：同步标识，取值范围：0或1。默认值为0。
  SYNC = 0表示异步执行，调用后立即返回，但不关注指令执行情况；
  SYNC = 1表示同步执行，调用后，待指令执行完才返回。

示例:

```
GoIO(P1, {0, 10, 2, 1})
```

机械臂以默认设置向P1点运动，当运动到距离起点10%的位置时，将DO2设置为开。

MoveIO

原型:

```
MoveIO(P,{ {Mode,Distance,Index,Status},{Mode,Distance,Index,Status}},"User=1 Tool=2 CP=1 Spee
dS=50 AccelS=20 SYNC=1")
```

描述:

从当前位置以直线方式运动至笛卡尔坐标系下的目标位置，运动时并行设置数字输出端口状态。

必选参数:

- P：表示目标点，可在该工程的“示教点”页面添加后在此引用，也可自定义点位，但只支持笛卡尔坐标点位。
- 并行数字输出参数：设置当机械臂运动到指定距离或百分比时，触发指定DO。可设置多组，每组含有以下参数：
  - Mode：触发模式。0表示距离百分比，1表示距离数值。
  - Distance：指定距离。
    - Distance为正数时，表示离起点的距离。
    - Distance为负数时，表示离目标点的距离。
    - Mode为0时，Distance表示和总距离的百分比，取值1~100。
    - Mode为1时，Distance表示距离的值，单位：毫米。
  - Index：DO端子的编号。
  - Status：要设置的DO状态，0表示关，1表示开。

可选参数:

- User：用户坐标系索引，需要在设置中添加后在此引用。
- Tool：工具坐标系索引，需要在设置中添加后在此引用。
- CP：运动时设置平滑过渡（详见运动参数中的CP指令），取值范围：0~100。
- SpeedS：运动速度比例，取值范围：1~100。
- AccelS：运动加速度比例，取值范围：1~100。
- SYNC：同步标识，取值范围：0或1。默认值为0。
  SYNC = 0表示异步执行，调用后立即返回，但不关注指令执行情况；
  SYNC = 1表示同步执行，调用后，待指令执行完才返回。

示例:

```
MoveIO(P1, {0, 10, 2, 1})
```

机械臂以默认设置向P1点运动，当运动到距离起点10%的位置时，将DO2设置为开。

运动参数

运动参数函数用于设置或获取机械臂运动相关参数。

Sync

原型：

```
Sync()
```

描述：

阻塞程序执行队列指令，待所有队列指令执行完才返回，执行后续指令。一般用于等待机械臂完成运动。

示例：

```
Go(P1)
Go(P2)
Sync()
```

待机械臂先运动至P1，再运动到P2后才返回，执行后续指令。

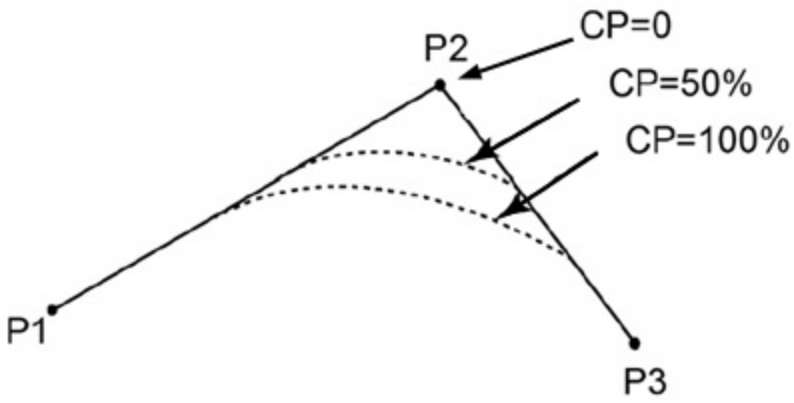
CP

原型：

```
CP(R)
```

描述：

设置平滑过渡比例，即机械臂连续运动经过多个点时，经过中间点是以直角方式过渡还是以曲线方式过渡。



必选参数：

R：平滑过渡比例。取值范围：0~100。

示例：

```
CP(50)
Move(P1)
Move(P2)
Move(P3)
```

机械臂从P1运动至P3点经过P2点时以50%平滑过渡。

Speed

原型：

```
Speed(R)
```

描述：

设置关节运动方式的速度比例。机械臂实际运动速度比例为参数设置的比例乘以控制软件再现设置中的值再乘以全局速率。

必选参数：

R：速度比例。取值范围：0~100。

示例：

```
Speed(20)
Go(P1)
```

机械臂以软件设置速度的20%的速度比例运动至P1点。

Accel

原型：

```
Accel(R)
```

描述：

设置关节运动方式的加速度比例。机械臂实际运动加速度比例为参数设置的比例乘以控制软件再现设置中的值再乘以全局速率。

必选参数：

R：加速度比例。取值范围：0~100。

示例：

```
Accel(50)
Go(P1)
```

机械臂以软件设置速度的50%的加速度比例运动至P1点。

SpeedS

原型：

```
SpeedS(R)
```

描述：

设置直线和弧线运动方式的速度比例。机械臂实际运动速度比例为参数设置的比例乘以控制软件再现设置中的值再乘以全局速率。

必选参数：

R：速度比例。取值范围：0~100。

示例：

```
SpeedS(20)
Move(P1)
```

机械臂以软件设置速度的20%的速度比例运动至P1点。

AccelS

原型：

```
AccelS(R)
```

描述:

设置直线和弧线运动方式的加速度比例。机械臂实际运动加速度比例为参数设置的比例乘以控制软件再现设置中的值再乘以全局速率。

必选参数:

R：加速度比例。取值范围：0~100。

示例:

```
AccelS(50)
Move(P1)
```

机械臂以软件设置速度的50%的加速度比例运动至P1点。

GetPose

原型:

```
GetPose()
```

描述:

获取笛卡尔坐标系下机械臂的实时位姿。如果设置了用户坐标系或工具坐标系，则获取的位姿为当前坐标系下的位姿。

返回:

机械臂当前位姿的笛卡尔坐标值。

示例:

```
local currentPose = GetPose()
Go(P1)
Go(currentPose)
```

机械臂先运动至P1，再返回当前位姿。

GetAngle

原型:

```
GetAngle()
```

描述:

获取关节坐标系下机械臂的实时位姿。

返回:

机械臂当前位姿的关节坐标值。

示例:

```
local currentAngle = GetAngle()
Go(P1)
MoveJ(currentAngle)
```

机械臂先运动至P1，再返回当前位姿。

CheckGo

原型:

```
CheckGo(P)
```

描述:

检查关节运动指令的可运行性。

必选参数:

P：表示目标点，可在该工程的“示教点”页面添加后在此引用，也可自定义点位，但只支持笛卡尔坐标点位。

返回:

检查结果。

- 0：无错误
- 16：终点接近肩部奇异
- 17：终点逆解无解
- 18：终点逆解限位
- 22：手势切换错误
- 26：终点接近腕部奇异
- 27：终点接近肘部奇异
- 29：速度参数错误
- 32：轨迹有肩部奇异点
- 33：轨迹存在逆解无解点
- 34：轨迹存在逆解限位点

- 35：轨迹有腕部奇异点
- 36：轨迹有轴部奇异点
- 37：轨迹存在关节跳变点

示例：

```
local status=CheckGo (P1)
if(status==0)
    then
        Go(P1)
    end
```

检查使用关节运动默认设置是否可达P1，如果可达，则关节运动至P1。

CheckMove

原型：

```
CheckMove(P)
```

描述：

检查直线运动指令的可运行性。

必选参数：

P：表示目标点，可在该工程的“示教点”页面添加后在此引用，也可自定义点位，但只支持笛卡尔坐标点位。

返回：

检查结果。

- 0：无错误
- 16：终点接近肩部奇异
- 17：终点逆解无解
- 18：终点逆解限位
- 22：手势切换错误
- 26：终点接近腕部奇异
- 27：终点接近肘部奇异
- 29：速度参数错误
- 32：轨迹有肩部奇异点
- 33：轨迹存在逆解无解点
- 34：轨迹存在逆解限位点
- 35：轨迹有腕部奇异点
- 36：轨迹有轴部奇异点

- 37：轨迹存在关节跳变点

示例：

```
local status=CheckMove (P1)
if(status==0)
    then
        Move(P1)
    end
```

检查使用直线运动默认设置是否可达P1，如果可达，则直线运动至P1。

相对运动指令

运动指令函数用于控制机械臂进行偏移运动。运动速度比例/加速度比例还可以在运动参数中设置，如果两处都设置了，以运动指令的值为准。机械臂实际运动速度/加速度比例为参数设置的比例乘以控制软件再现设置中的值再乘以全局速率。

RP

原型：

```
RP(P, {OffsetX, OffsetY, OffsetZ})
```

描述：

对指定点位在笛卡尔坐标系下增加X、Y、Z方向上的偏移量并返回一个新的笛卡尔坐标点。

必选参数：

- 偏移前的点位，可在该工程的“示教点”页面添加后在此引用，也可自定义点位，但只支持笛卡尔坐标点位。
- OffsetX, OffsetY, OffsetZ：笛卡尔坐标系下X、Y、Z方向上的偏移量，单位：毫米。

返回：

偏移后的笛卡尔坐标点。

示例：

```
Go(RP(P1, {30,50,10}))
```

将P1在X、Y、Z轴上分别偏移一定距离，然后运动至偏移后的点位。

RJ

原型：

```
RJ(P, {Offset1, Offset2, Offset3, Offset4, Offset5, Offset6})
```

描述：

对指定点位在关节坐标系下增加J1~J6轴偏移量并返回一个新的关节坐标点。

必选参数：

- 偏移前的点位，可在该工程的“示教点”页面添加后在此引用，也可自定义点位，但只支持关节坐标点位。
- Offset1~Offset6：关节坐标系下J1轴 ~ J6轴方向上的偏移值，单位：度。

返回：

偏移后的关节坐标点。

示例：

```
MoveJ(RJ(P1, {60,50,32,30,25,30}))
```

将P1在J1~J6轴上分别偏移一定角度，然后运动至偏移后的点位。

GoR

原型：

```
GoR({OffsetX, OffsetY, OffsetZ},"User=1 Tool=2 CP=1 Speed=50 Accel=20 SYNC=1")
```

描述：

从当前位置以点到点（关节运动）方式运动至笛卡尔坐标系下的偏移位置。关节运动的轨迹非直线，所有关节会同时完成运动。

必选参数：

OffsetX, OffsetY, OffsetZ： 笛卡尔坐标系下X轴、Y轴、Z轴方向上的偏移，单位：毫米。

可选参数：

- User：用户坐标系索引，需要在设置中添加后在此引用。
- Tool：工具坐标系索引，需要在设置中添加后在此引用。
- CP：运动时设置平滑过渡（详见运动参数中的CP指令），取值范围：0~100。
- Speed：运动速度比例，取值范围：1~100。
- Accel：运动加速度比例，取值范围：1~100。
- SYNC：同步标识，取值范围：0或1。默认值为0。
  SYNC = 0表示异步执行，调用后立即返回，但不关注指令执行情况；
  SYNC = 1表示同步执行，调用后，待指令执行完才返回。

示例：

```
GoR({10,10,10})
```

机械臂以默认设置点对点运动至偏移点。

MoveR

原型：

```
MoveR({OffsetX, OffsetY, OffsetZ},"User=1 Tool=2 CP=1 SpeedS=50 AccelS=20 SYNC=1")
```

描述：

从当前位置以直线方式运动至笛卡尔坐标系下的偏移位置。

必选参数：

OffsetX, OffsetY, OffsetZ： 笛卡尔坐标系下X轴、Y轴、Z轴方向上的偏移，单位：毫米。

可选参数：

- User：用户坐标系索引，需要在设置中添加后在此引用。
- Tool：工具坐标系索引，需要在设置中添加后在此引用。
- CP：运动时设置平滑过渡（详见运动参数中的CP指令），取值范围：0~100。
- SpeedS：运动速度比例，取值范围：1~100。
- AccelS：运动加速度比例，取值范围：1~100。
- SYNC：同步标识，取值范围：0或1。默认值为0。
  SYNC = 0表示异步执行，调用后立即返回，但不关注指令执行情况；
  SYNC = 1表示同步执行，调用后，待指令执行完才返回。

示例：

```
MoveR({10,10,10})
```

机械臂以默认设置直线运动至偏移点。

MoveJR

原型：

```
MoveJR({Offset1, Offset2, Offset3, Offset4, Offset5, Offset6},"User=1 Tool=2 CP=1 Speed=50 Acc
el=20 SYNC=1")
```

描述：

从当前位置以点到点（关节运动）方式运动至关节偏移角度。

必选参数：

Offset1~Offset6：关节坐标系下J1轴 ~ J6轴方向上的偏移值，单位：度。

可选参数：

- User：用户坐标系索引，需要在设置中添加后在此引用。
- Tool：工具坐标系索引，需要在设置中添加后在此引用。
- CP：运动时设置平滑过渡（详见运动参数中的CP指令），取值范围：0~100。
- Speed：运动速度比例，取值范围：1~100。
- Accel：运动加速度比例，取值范围：1~100。
- SYNC：同步标识，取值范围：0或1。默认值为0。
  SYNC = 0表示异步执行，调用后立即返回，但不关注指令执行情况；
  SYNC = 1表示同步执行，调用后，待指令执行完才返回。

示例：

```
MoveJR({20,20,10,0,10,0})
```

机械臂以默认设置关节运动至偏移角度。

IO

IO指令用于进行机械臂系统IO的读写和相关参数的设置。

DI

原型：

```
DI(index)
```

描述：

读取数字输入端口状态。

必选参数：

index：DI端子的编号。

返回：

对应的DI端子的电平（ON/OFF）。

示例：

```
if (DI(1)==ON) then
Move(P1)
end
```

DI1为高电平（ON）时机械臂以直线运动方式运动至P1点。

WaitDI

原型：

```
WaitDI(index,ON|OFF,period)
```

描述：

读取数字输入端口状态，若读取到的状态与指定状态一致，程序才往下执行，否则，间隔指定周期读取数字 输入端口状态。

必选参数：

- index：DI端子的编号。

- ON|OFF：等待的DI端口状态，ON：高电平；OFF：低电平。

可选参数：

period：读取周期，单位：毫秒，默认50ms。

示例：

```
WaitDI(1, ON)
Move(P1)
```

每50ms读取DI1的状态，如果DI1为高电平时，程序才往下执行，机械臂以直线运动方式运动至P1点。

DO

原型：

```
DO(index,ON|OFF)
```

描述：

设置数字输出端口状态。

必选参数：

- index：DO端子的编号。
- ON|OF：要设置的DO端口状态，ON：高电平；OFF：低电平。

示例：

```
DO(1,ON)
```

将DO1设置为高电平（ON）。

DOExecute

原型：

```
DOExecute(index,ON|OFF)
```

描述：

无视当前指令队列，立即设置数字输出端口状态。

必选参数：

- index：DO端子的编号。
- ON|OF：要设置的DO端口状态，ON：高电平；OFF：低电平。

示例：

```
DOExecute(1,ON)
```

无视当前指令队列，立即将DO1设置为高电平（ON）。

ToolDO

原型：

```
ToolDO(index,ON|OFF)
```

描述：

设置末端数字输出端口状态。

必选参数：

- index：末端DO端子的编号。
- ON|OF：要设置的DO端口状态，ON：高电平；OFF：低电平。

示例：

```
ToolDO(1,ON)
```

将末端DO1设置为高电平（ON）。

ToolDOExecute

原型：

```
ToolDOExecute(index,ON|OFF)
```

描述：

无视当前指令队列，立即设置末端数字输出端口状态。

必选参数：

- index：末端DO端子的编号。
- ON|OF：要设置的DO端口状态，ON：高电平；OFF：低电平。

示例：

```
ToolDOExecute(1,ON)
```

无视当前指令队列，立即将末端DO1设置为高电平（ON）。

ToolDI

原型：

```
ToolDI(index)
```

描述：

读取末端数字输入端口状态。

必选参数：

index：末端DI端子的编号。

返回：

对应的DI端子的电平（ON/OFF）。

示例：

```
if (ToolDI(1)==ON) then
Move(P1)
end
```

末端DI1为高电平（ON）时机械臂以直线运动方式运动至P1点。

ToolAI

原型：

```
ToolAI(index)
```

描述：

读取末端模拟输入端口的值。使用前需要通过ToolAnalogMode将端子设置为电压输入模式。

必选参数：

index：末端AI端子的编号。

返回：

对应的AI端子的值。

示例：

```
test = ToolAI(1)
```

读取末端AI1的值并赋值给变量test。

ToolAnalogMode

原型：

```
ToolAnalogMode(mode)
```

描述：

设置末端模拟输入端口的模式。

必选参数：

mode：模拟输入端口的模式

- 00：默认，485模式。
- 10：电流采集模式。
- 11：0~3.3V 电压输入模式。
- 12：0~10V 电压输入模式。

示例：

```
ToolAnalogMode(11)
```

设置末端模拟输入端口的模式为0~3.3V 电压输入模式。

AO

原型：

```
AO(index,value)
```

描述：

设置模拟输出端口的电压值。

必选参数：

- index：AO端子的编号。
- value：要设置的电压值，取值范围0~10

示例：

```
AO(1,2)
```

将AO1的电压设置为2V。

AOExecute

原型：

```
AOExecute(index,value)
```

描述：

无视当前指令队列，立即设置模拟输出端口的电压值。

必选参数：

- index：AO端子的编号。
- value：要设置的电压值，取值范围0~10

示例：

```
AOExecute(1,2)
```

无视当前指令队列，立即将AO1的电压设置为2V。

AI

原型：

```
AI(index)
```

描述：

读取模拟输入端口的值。

必选参数：

index：AI端子的编号。

返回：

对应的AI端子的值。

示例：

```
test = AI(1)
```

读取AI1的值并赋值给变量test。

SetTool485

原型：

```
SetTool485(baud,parity,stopbit)
```

描述：

设置末端工具的RS485接口对应的数据格式。

必选参数：

- baud：RS485接口的波特率
- parity：是否有奇偶校验位。"O"表示奇校验，"E"表示偶校验，"N"表示无奇偶校验位。
- stopbit：停止位长度。取值范围：1，1.5，2。

示例：

```
SetTool485(115200,"N",1)
```

将末端工具的RS485接口对应的波特率设置为115200Hz，无奇偶校验位，停止位长度为1。

SetABZPPC

原型：

```
SetABZPPC(resolution)
```

描述：

设置ABZ编码器的分辨率。

必选参数：

resolution：编码器的分辨率，单位：脉冲/毫米。

示例：

```
SetABZPPC(1000)
```

设置ABZ编码器的分辨率为1000脉冲/毫米。

GetABZ

原型：

```
GetABZ()
```

描述：

获取已设置的ABZ编码器的分辨率。

返回值：

编码器的分辨率，单位：脉冲/毫米。

示例：

```
local abz = GetABZ()
```

获取已设置的ABZ编码器的分辨率并赋值给变量abz。

TCP/UDP

TCP/UDP函数用于进行TCP或UDP通信。

TCPCreate

原型：

```
TCPCreate(isServer, IP, port)
```

描述：

创建TCP网络对象，仅可创建一个。

必选参数：

- isServer：是否创建服务端。true：表示创建服务端；false：表示创建客户端。
- IP：服务端IP地址。需与客户端IP地址在同一网段，且不冲突。创建服务端时为机械臂的IP地址，创建客户端时为对端的地址。
- port：服务端端口。机械臂作为服务端时，“port”不能设置为502或8080，否则会与Modbus默认端口或流线跟踪中使用的端口冲突，导致创建TCP网络对象失败。

返回：

- err： 0表示创建TCP网络对象成功 ， 1表示创建TCP网络对象失败。
- socket：创建的socket对象。

示例1：

```
local ip="192.168.5.1" -- 机械臂的IP地址作为服务端的IP地址
local port=6001 -- 服务端端口
local err=0
local socket=0
err, socket = TCPCreate(true, ip, port)
```

创建TCP服务端。

示例2：

```
local ip="192.168.5.25" -- 外部设备如相机的IP地址作为服务端的IP地址
local port=6001 -- 服务端端口
local err=0
local socket=0
err, socket = TCPCreate(false, ip, port)
```

创建TCP客户端。

TCPStart

原型：

```
TCPStart(socket, timeout)
```

描述：

建立TCP连接。机械臂作为服务端时，等待客户端来连接；机械臂作为客户端时，主动连接服务端。

必选参数：

- socket：已创建的socket对象。
- timeout：等待超时时间，单位：秒。如果设置为0，会一直等待到连接建立成功；如果不为0，则超过设定的时间后返回连接失败。

返回：

连接结果。

- 0：连接成功
- 1：输入参数错误
- 2：socket对象不存在
- 3：设置超时时间错误
- 4：连接失败

示例：

```
err = TCPStart(socket, 0)
```

开始建立TCP连接，一直等待直到连接建立成功。

TCPRead

原型：

```
TCPRead(socket, timeout, type)
```

描述：

接收TCP对端发送的数据。

必选参数：

socket：已创建的socket对象。

可选参数：

- timeout：等待超时时间，单位：秒。如果设置为0，会一直等待直到读取完数据再往下执行；如果不为0，则超过设定的时间后会直接往下执行。
- type：返回值类型。如果不设置，则RecBuf缓存格式为table形式；如果设置为“string”，则RecBuf缓存为字符串。

返回：

- err： 0表示接收数据成功 ， 1表示接收数据失败。
- Recbuf：接收数据缓存区。

示例：

```
err, RecBuf = TCPRead(socket,0,"string") -- RecBuf数据类型为字符串
err, RecBuf = TCPRead(socket, 0) -- RecBuf数据类型为table
```

接收TCP数据，接收的数据分别保存为字符串和table形式。

TCPWrite

原型：

```
TCPWrite(socket, buf, timeout)
```

描述：

发送数据给TCP对端。

必选参数：

- socket：已创建的socket对象。
- buf：要发送的数据。

可选参数：

timeout：等待超时时间，单位：秒。如果设置为0，会一直等待直到对端接收完数据再往下执行；如果不为0，则超过设定的时间后会直接往下执行。

返回：

发送结果。

- 0：发送成功。
- 1：发送失败

示例：

```
TCPWrite(socket, "test")
```

发送TCP数据，数据内容为“test”。

TCPDestroy

原型：

```
TCPDestroy(socket)
```

描述：

断开TCP连接并销毁socket对象。

必选参数：

socket：已创建的socket对象。

返回：

执行结果。

- 0：执行成功
- 1：执行失败

示例：

```
TCPDestroy(socket)
```

断开与TCP对端的连接。

UDPCreate

原型：

```
UDPCreate(isServer, IP, port)
```

描述：

创建UDP网络对象，仅可创建一个。

必选参数：

- isServer：固定填写为false。

- IP：对端IP地址。需与机械臂IP地址在同一网段，且不冲突。
- port：对端端口。

返回：

- err： 0表示创建UDP网络对象成功 ， 1表示创建UDP网络对象失败。
- socket：创建的socket对象。

示例：

```
local ip="192.168.5.25" -- 外部设备如相机的IP地址作为对端的IP地址
local port=6001 -- 对端端口
local err=0
local socket=0
err, socket = UDPCreate(false, ip, port)
```

创建UDP网络对象。

UDPRead

原型：

```
UDPRead(socket, timeout, type)
```

描述：

接收UDP对端发送的数据。

必选参数：

socket：已创建的socket对象。

可选参数：

- timeout：等待超时时间，单位：秒。如果设置为0，会一直等待直到读取完数据再往下执行；如果不为0，则超过设定的时间后会直接往下执行。
- type：返回值类型。如果不设置，则RecBuf缓存格式为table形式；如果设置为“string”，则RecBuf缓存为字符串。

返回：

- err： 0表示接收数据成功 ， 1表示接收数据失败。
- Recbuf：接收数据缓存区。

示例：

```
err, RecBuf = UDPRead(socket,0,"string") -- RecBuf数据类型为字符串
err, RecBuf = UDPRead(socket, 0) -- RecBuf数据类型为table
```

接收UDP数据，接收的数据分别保存为字符串和table形式。

UDPWrite

原型：

```
UDPWrite(socket, buf, timeout)
```

描述：

发送数据给UDP对端。

必选参数：

- socket：已创建的socket对象。
- buf：要发送的数据。

可选参数：

timeout：等待超时时间，单位：秒。如果设置为0，会一直等待直到对端接收完数据再往下执行；如果不为0，则超过设定的时间后会直接往下执行。

返回：

发送结果。

- 0：发送成功。
- 1：发送失败

示例：

```
UDPWrite(socket, "test")
```

发送UDP数据，数据内容为“test”。

Modbus

Modbus函数用于进行Modbus通信。

ModbusCreate

原型：

```
ModbusCreate(IP,port,slave_id)
```

描述：

创建Modbus主站，并和从站建立连接。

可选参数：

- IP：从站IP地址。不指定或指定为“127.0.0.1”或“0.0.0.1”时，表示连接本机的Modbus从站。
- port：从站端口。
- slave_id：从站ID。

返回：

- err： 0表示创建Modbus主站成功 ， 1表示创建Modbus主站失败。
- id：主站索引，取值范围0~4。

示例1：

```
local ip="192.168.5.123" -- 从站的IP地址
local port=503 -- 从站端口
local err=0
local id=0
err, id = ModbusCreate(ip, port, 1)
```

创建Modbus主站，并和指定从站建立连接。

示例1：

以下指令都表示连接本机的Modbus从站。

```
ModbusCreate()
```

```
ModbusCreate(“127.0.0.1”)
```

```
ModbusCreate(“0.0.0.1”)
```

```
ModbusCreate(“127.0.0.1”, xxx,xxx) -- xxx 任意值
```

```
ModbusCreate(“0.0.0.1”, xxx,xxx) -- xxx 任意值
```

GetInBits

原型:

```
GetInBits(id, addr, count)
```

描述:

读取Modbus从站触点寄存器地址的值。

必选参数:

- id：创建Modbus主站时返回的主站索引。
- addr：触点寄存器起始地址地址。取值范围：0 ~ 4095
- count：连续读取触点寄存器的值的数量。

返回:

触点寄存器地址的值，存储在table中。table中第一个值对应触点寄存器起始地址的值。

示例:

```
inBits = GetInBits(id,0,5)
```

从地址0开始连续读取5个地址的值。

GetInRegs

原型:

```
GetInRegs(id, addr, count, type)
```

描述:

按照指定的数据类型，读取Modbus从站输入寄存器地址的值。

必选参数:

- id：创建Modbus主站时返回的主站索引。
- addr：输入寄存器起始地址地址。取值范围：0 ~ 4095
- count：连续读取输入寄存器的值的数量。

可选参数：

type：数据类型。

- 如果为空，默认为U16
- U16：16位无符号整数（2个字节，占用1个寄存器）
- U32：32位无符号整数（4个字节，占用2个寄存器）
- F32：32位单精度浮点数（4个字节，占用2个寄存器）
- F64：64位双精度浮点数（8个字节，占用4个寄存器）

返回：

输入寄存器地址的值，存储在table中。table中第一个值对应输入寄存器起始地址的值。

示例：

```
data = GetInRegs(id, 2048, 1, “U32”)
```

从地址2048开始读取一个32位无符号整数。

GetCoils

原型：

```
GetCoils(id, addr, count)
```

描述：

读取Modbus从站线圈寄存器地址的值。

必选参数：

- id：创建Modbus主站时返回的主站索引。
- addr：线圈寄存器起始地址地址。取值范围：0 ~ 4095
- count：连续读取线圈寄存器的值的数量。

返回：

线圈寄存器地址的值，存储在table中。table中第一个值对应线圈寄存器起始地址的值。

示例：

```
Coils = GetCoils(id,0,5)
```

从地址0开始连续读取5个地址的值。

GetHoldRegs

原型：

```
GetHoldRegs(id, addr, count, type)
```

描述：

按照指定的数据类型，读取Modbus从站保持寄存器地址的值。

必选参数：

- id：创建Modbus主站时返回的主站索引。
- addr：保持寄存器起始地址地址。取值范围：0 ~ 4095
- count：连续读取保持寄存器的值的数量。

可选参数：

type：数据类型。

- 如果为空，默认为U16
- U16：16位无符号整数（2个字节，占用1个寄存器）
- U32：32位无符号整数（4个字节，占用2个寄存器）
- F32：32位单精度浮点数（4个字节，占用2个寄存器）
- F64：64位双精度浮点数（8个字节，占用4个寄存器）

返回：

保持寄存器地址的值，存储在table中。table中第一个值对应保持寄存器起始地址的值。

示例：

```
data = GetHoldRegs(id, 2048, 1, “U32”)
```

从地址2048开始读取一个32位无符号整数。

SetCoils

原型：

```
SetCoils(id, addr, count, table)
```

描述:

将指定的值写入线圈寄存器指定的地址。

必选参数:

- id：创建Modbus主站时返回的主站索引。
- addr：线圈寄存器起始地址地址。取值范围：6 ~ 4095
- count：连续写入线圈寄存器的值的数量。
- table：要写入线圈寄存器的值，存储在table中。table中第一个值对应线圈寄存器起始地址的值。

示例:

```
local Coils = {0,1,1,1,0}
SetCoils(id, 1024, #coils, Coils)
```

从地址1024开始，连续写5个线圈。

SetHoldRegs

原型:

```
SetHoldRegs(id, addr, count, table, type)
```

描述:

按照指定的数据类型，写入指定的值到保持寄存器指定的地址。

必选参数:

- id：创建Modbus主站时返回的主站索引。
- addr：保持寄存器起始地址地址。取值范围：0 ~ 4095
- count：连续写入保持寄存器的值的数量。
- table：要写入线圈寄存器的值，存储在table中。table中第一个值对应线圈寄存器起始地址的值。

可选参数:

type：数据类型。

- 如果为空，默认为U16
- U16：16位无符号整数（2个字节，占用1个寄存器）

- U32：32位无符号整数（4个字节，占用2个寄存器）
- F32：32位单精度浮点数（4个字节，占用2个寄存器）
- F64：64位双精度浮点数（8个字节，占用4个寄存器）

示例：

```
local data = {95.32105}
SetHoldRegs(id, 2048, #data, data, “F64”)
```

从地址2048开始，写入一个64位双精度浮点数。

ModbusClose

原型：

```
ModbusClose(id)
```

描述：

和Modbus从站断开连接。

可选参数：

id：创建Modbus主站时返回的主站索引。

返回：

操作结果

- 0：断开连接成功。
- 1：断开连接失败。

示例：

```
ModbusClose(id)
```

和Modbus从站断开连接。

程序控制

程序控制函数为程序运行控制相关的通用函数。其中while,if,for为lua的流程控制指令，请参考Lua基础语法 - 流程控制。print为打印指令，用于输出信息到控制台。

Sleep

原型：

```
Sleep(time)
```

描述：

脚本延时执行下一条指令。

必选参数：

time：延时时间。单位：毫秒

示例：

```
DO(1,ON)
Sleep(100)
DO(1,OFF)
```

设置DO1为ON后等待100ms再设置DO1为OFF。

Wait

原型：

```
Wait(time)
```

描述：

延时下发运动指令或运动完成后延时下发下一条指令。

必选参数：

time：延时时间。单位：毫秒

示例：

```
DO(1,ON)
```

```
Wait(100)
Go(P1)
Wait(100)
DO(1,OFF)
```

设置DO1为ON后等待100ms机械臂再运动至P1，运行至P1点后延时100ms，再设置DO1为OFF。

Pause

原型：

```
Pause()
```

描述：

暂停脚本运行。需要通过控制软件或者远程控制操作才可继续运行。

示例：

```
Go(P1)
Pause()
Go(P2)
```

机械臂运行至P1点后暂停运行，通过外部控制继续运行后才会运行至P2点。

SetCollisionLevel

原型：

```
SetCollisionLevel(level)
```

描述：

设置碰撞检测等级。通过该接口设置的碰撞检测等级仅在工程运行期间生效，工程停止后会恢复为修改前的值。

必选参数：

level：碰撞检测等级，取值范围0~5，0表示关闭碰撞检测，1~5等级越高碰撞检测越灵敏。

示例：

```
SetCollisionLevel(2)
```

设置碰撞检测等级为2级。

ResetElapsedTime

原型：

```
ResetElapsedTime()
```

描述：

待此指令前所有指令执行完成后开始计时，需配合ElapsedTime()一起使用，可用于计算运行时间。

示例：

请参考ElapsedTime的示例。

ElapsedTime

原型：

```
ElapsedTime()
```

描述：

结束计时，返回时间差，需配合ElapsedTime()一起使用。

返回：

从开始计时到结束计时的时间差。

示例：

```
Go(P2, "Speed=100 Accel=100")
ResetElapsedTime()
for i=1,10 do
Move(P1)
Move(P2)
end
print (ElapsedTime())
```

计算机械臂在P1和P2间来回运动10次的时间，打印至控制台。

Systime

原型：

```
Systime()
```

描述:

获取当前系统时间。

返回:

当前系统时间的Unix时间戳。

示例:

```
local time = Systime()
```

获取系统当前时间并保存在变量time里。

SetUser

原型:

```
SetUser(index,table)
```

描述:

修改指定的用户坐标系。该修改仅在当前工程运行中生效，工程停止后坐标系会恢复为修改前的值。

必选参数:

- index：用户坐标系序号，取值范围0~9，其中坐标系0为默认用户坐标系。
- table：修改后的用户坐标系矩阵，格式为{x, y, z, rx, ry, rz}。

示例:

```
SetUser(1,{10,10,10,0,0,0})
```

修改用户坐标系1为X=10，Y=10，Z=10，RX=0，RY=0，RZ=0。

SetTool

原型:

```
SetTool(index,table)
```

描述:

修改指定的工具坐标系。该修改仅在当前工程运行中生效，工程停止后坐标系会恢复为修改前的值。

必选参数：

- index：工具坐标系序号，取值范围0~9，其中坐标系0为默认工具坐标系。
- table：修改后的工具坐标系矩阵，格式为{x, y, z, rx, ry, rz}。

示例：

```
SetTool(1,{10,10,10,0,0,0})
```

修改工具坐标系1为X=10，Y=10，Z=10，RX=0，RY=0，RZ=0。

视觉

视觉工艺包用于配置相机通讯相关配置。相机固定于机器人工作范围内，位置固定，拍照视野范围固定，相机充当机器人的眼睛，采用以太网通讯或I/O触发方式与机器人进行数据交互。

相机的安装与配置方法根据相机不同而有所差异，本文不进行详述。

配置视觉工艺

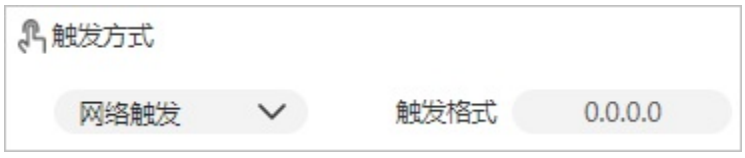
单击视觉函数组右侧的“视觉配置”按钮开始配置相机。如果您是第一次配置相机工艺，需要先单击“新建”按钮，输入相机名称后新建一个相机配置，然后软件显示如下界面。

触发方式

设定触发相机的方式。

- IO内触发：相机与机器人的DO口连接，需根据电气接线端口配置相应的输出端口号。

- 网络触发：相机与机器人的以太网口连接，需配置网络触发的字符串。需与相机侧事先约定好，机器人通过网络发送该字符串给相机会触发相机。



网络基本参数

基本参数用于设置相机和机器人的通讯模式，包含以下几种模式：

- TCP_Client：TCP通讯，机器人作为客户端，相机作为服务器。需要配置相机的IP，端口和连接超时时间。
- TCP_Server：TCP通讯，机器人作为服务器，相机作为客户端。需要配置端口和连接超时时间

接收方式包括堵塞和非堵塞两种，请根据工程脚本设计选择。

- 堵塞：发送触发信号后，程序会在堵塞时间内一直停留在接收数据的那一行函数，直到接收相机发送过来的数据或者超过堵塞时间后才会继续运行；若堵塞时间设置为 0，程序会一直停留在数据接收函数那一行直到接收到视觉发送的数据。
- 非堵塞：发送触发信号后，不管有没有接收到相机发送来的数据，程序会继续执行。

网络接受格式表示相机发送过来的数据的格式，用于解析。如果当前默认的数据位不够，还可以通过“增加数据位”的方式来增加接收数据的长度，最多到可增加到8位:
NO,D1,D2,D3,D4,D5,D6,STA，其中，NO代表起始位模板号；STA代表结束位（状态位）。

可设置多种视觉数据格式，例如：

- 没有起始位和结束位：XX,YY,CC；
- 有起始位，没有结束位：NO,XX,YY,CC；
- 没有起始位，有结束位:XX,YY,CC,STA；
- 有起始位和结束位：NO,XX,YY,CC,STA。

配置完成后，单击右上角的“保存”。

InitCam

原型：

```
InitCam(CAM)
```

描述:

与指定的相机建立连接并初始化。

必选参数：

CAM：相机的名称，需要与配置视觉工艺时配置的相机名称一致。

返回：

初始化结果。

- 0：初始化成功
- 1：初始化失败

示例：

```
InitCam("CAM0")
```

连接并初始化名称为CAM0的相机。

TriggerCam

原型：

```
TriggerCam(CAM)
```

描述：

触发已经初始化的相机拍照。

必选参数：

CAM：相机的名称，需要与配置视觉工艺时配置的相机名称一致。

返回：

触发结果。

- 0：触发成功
- 1：触发失败

示例：

```
TriggerCam("CAM0")
```

触发名称为CAM0的相机拍照。

SendCam

原型：

```
SendCam(CAM,data)
```

描述:

发送数据给已经初始化的相机。

必选参数:

- CAM：相机的名称，需要与配置视觉工艺时配置的相机名称一致。
- data：发送的数据

返回:

发送结果。

- 0：发送成功
- 1：发送失败

示例:

```
SendCam("CAM0","0,0,0,0")
```

给名称为CAM0的相机发送数据，数据内容为"0,0,0,0"。

RecvCam

原型:

```
RecvCam(CAM,type)
```

描述:

接收已经初始化的相机发送的数据。

必选参数:

CAM：相机的名称，需要与配置视觉工艺时配置的相机名称一致。

可选参数:

type：接收的数据的类型。取值范围：number或string，不设置时默认为number。

返回:

- err：错误码
  - 0：正常接收
  - 1：接收超时
  - 2：数据格式配置有误，无法解析
  - 3：网络断开
- n：数据的组数。
- data：接收到的数据，保存在二维数组中。

示例：

```
local err,n,data = RecvCam("CAM0","number")
```

接收名称为CAM0的相机发送的数据，数据类型为number。

DestroyCam

原型：

```
DestroyCam(CAM)
```

描述：

关闭与相机的连接。

必选参数：

CAM：相机的名称，需要与配置视觉工艺时配置的相机名称一致。。

返回：

关闭结果。

- 0：关闭成功
- 1：关闭失败

示例：

```
DestroyCam("CAM0")
```

关闭与名称为CAM0的相机的连接。

应用案例

配置完相机参数后，可通过调用视觉API进行编程，实现触发相机后接收相机数据。本文以脚本编程为例，编写脚本获取CAM0的数据并赋值给示教点P2。

```
while true do
    ::create_camera::
    resultInit = InitCam("CAM0")                                    --Connect CAM0 camera
    if resultInit ~= 0 then
        print("Connect camera failed, code:", resultInit)
        Sleep(1000)
        goto create_camera
    end
    while true do
        TriggerCam("CAM0")                                          --Trigger CAM0 camera  photo
        SendCam("CAM0","1,2,3,0;")                                  --Send data to CAM0 camera
        err,visionNum,visionData  = RecvCam("CAM0","number")        --Receive CAM0 camera data
        if err ~= 0 then
            print("Failed to read data")
            Sleep(1000)
            break
        end

        print("(visionNum):",(visionNum))                          --Print how many sets of CAM0 camera data received
        print("(visionData[1][1]):",(visionData[1][1]))            --Print the first data of the first group received
        i = 1
        while not ((visionNum)<i) do
        print(type(P2.coordinate[1]))
        print(P2)
        P2.coordinate[1]=(visionData[i][1])                         --visionData[i][1] is assigned to P2.X
        P2.coordinate[2]=(visionData[i][2])                        --visionData[i][2] is assigned to P2.Y
        Go(P2,"SYNC=1")
        i = i + 1
        end
        Sleep(10)
    end
    Sleep(10)
end
```