ESTUN 机器人 SCARA 系列

PC 软件使用说明书

感谢您使用 ESTUN 机器人产品。

在使用机器人之前，务必仔细阅读机器人安全使用须知，并在理解该内容的基础上使用机器人。

本公司致力于不断提升产品品质，本手册中与产品有关的规格和信息如有改动，恕不另行通知。

本手册中所有陈述、信息和建议均已经过慎重处理，但不保证完全正确。本公司对于因使用本手册而造成的直接或间接损失不负任何责任。

用户必须对其应用任何产品负全部责任，须谨慎使用本手册及产品。

本手册所有内容的解释权属南京埃斯顿机器人工程有限公司。

产品服务热线：400-025-3336

地址：南京市江宁经济开发区吉印大道 1888 号　　邮编：211102

电话：025-81031813

公司主页：www.estun.com

电子邮箱：robot@estun.com

安全使用须知

本章说明为安全使用机器人而需要遵守的内容。在使用机器人之前，务必熟读并理解本章中所述内容。

使用埃斯顿机器人的公司、个人应该熟读所在地区、国家的标准和法律，并且安装适当的安全设施保护机器人的使用人员。使用前（安装、运转、保养、检修），请务必熟读并全部掌握本说明书和其他附属资料，在熟知全部设备知识、安全知识及注意事项后再开始使用。但是使用人员即使完全按照手册中给出的所有安全信息进行，埃斯顿公司也无法保证使用人员不会受到任何伤害。

使用人员的定义

使用人员的定义如下所示。

- 操作人员
  进行机器人的电源 ON/OFF 操作。
  从操作面板启动机器人程序。
- 程序人员
  进行机器人的操作。
  在安全区域内进行机器人的示教等。
- 维修人员
  进行机器人的操作。
  在安全区域内进行机器人的示教等。
  进行机器人的维护（修理、调整、更换）作业。

操作人员不能在安全区域内进行作业。

程序人员和维修人员可以在安全区域内进行作业。

在进行机器人的操作、编程、维护时，操作人员、程序人员、维修人员必须注意安全，至少应穿戴以下物品进行作业。

- 适合于作业内容的作业服
- 安全鞋
- 安全帽

专门培训

安全区域内的作业，包括搬运、设置、示教、调整、维护等。

在安全区域内进行作业，必须接受过机器人的专业培训。

关于培训的更多信息，请咨询南京埃斯顿机器人工程有限公司。

安全标示

本手册中若出现如下标示的说明内容，用户必须仔细阅读并严格遵守。

标示	定义
 危险	**危险标示** 如果用户不遵守该标示随后的安全说明，将有可能造成人员伤亡。
 注意	**注意标示** 如果用户不遵守该标示随后的安全说明，将有损坏设备或人体受伤。
IN FO	**说明或要点** 该标志随后的说明有助于用户更好的理解或有效的操作。

使用人员的安全事项

(1) 搬运和安装机器人时，务必按照埃斯顿公司所示的方法进行。错误的方法可能导致机器人翻倒，引发事故。

(2) 务必在机器人安装前划分出安全区域。可在机器人工作区域周围安装栅栏及警示牌保证机器人安全工作，防止闲杂人等进入以及防止机器人伤人。

(3) 机器人上方不能有悬挂物，以防掉落砸坏机器人等设备。

(4) 严禁倚靠电控柜，或者随意触动按钮，以防机器人产生未预料的动作，引起人身伤害或者设备损坏。

(5) 拆分机器人时，注意机器人上可能掉落的零件砸伤人员。

(6) 在进行外围设备的个别调试时，务必断开机器人电源后执行。

(7) 外围设备均应连接适当的地线。

(8) 首次使用机器人操作时，务必以低速进行。然后逐渐加速，并确认是否有异常。

(9) 在使用示教器时，带上手套可能导致操作上的失误，务必摘下手套后操作。

(10) 程序和系统变量等信息，可以保存到存储卡等介质中。为了防止因意外而丢失数据，建议用户定期保存数据。

(11) 严禁搬动机器人各轴，否则可能造成人身伤害和设备损坏。

(12) 在进行电控柜与机器人、外围设备间的配线及配管时须采取防护措施，如将管、线或电缆从坑内穿过或加保护盖予以遮盖，以免被人踩坏或被叉车辗压而坏。

(13) 任何工作的机器人都可能有不可预料的动作，对工作范围内的人员造成严重的伤害或者对设备造成破坏。在准备机器人工作前，需测试各安全措施（栅栏门、抱闸、安全指示灯）的可靠性。在开启机器人前，确保机器人工作范围内没有其他人员。

(14) 通过软件设定的动作范围及负载条件切勿超出产品规格表中的规定值，设置不当可能造成人员伤害或机器损坏。

(15) 如果工作必须要在机器人工作范围内进行，需要遵循以下规则：

- 模式选为手动模式后才能连接使能，断开计算机控制等其他自动控制。
- 当机器人处于手动模式时，速度必须限制在 250mm/s 以下；机器人需要调到手动全速度时，只有对风险充分了解的专业人员才能操作。
- 注意机器人的转动关节，防止头发、衣服被卷入关节；同时要注意机器人或者其他的附属设备运动可能造成的其他危险。
- 测试电机抱闸是否正常工作，以防机器人异常造成人身伤害。
- 考虑机器人突然向自己所处方位运动时的应变方案。
- 确保设置躲避场所，以防万一。

危险

在任何情况下，都不要站在任何机器人臂下方，以防机器人异常运动或者其他人连接使能。

注意

在现场需要放置一个二氧化碳灭火器，以防机器人系统失火。

操作人员：

(1) 在操作机器人前，应先按电控柜前门及示教器右上方的急停键，以检查“伺服准备”的指示灯是否熄灭，并确认其电源确已关闭。

(2) 在操作期间，绝不允许非工作人员触动电控柜。否则可能会造成机器人产生未预料的动作，从而引起人身伤害和设备损坏。

(3) 当往机器人上安装一个工具时，务必先切断（OFF）控制柜及所装工具上的电源，并且悬挂警示牌。安装过程中如接通电源，可能造成电击，或产生机器人的非正常运动，从而引起伤害。

(4) 急停

急停独立于所有机器人电气控制，可以停止所有机器人运动。

急停意味着连接到机器人上的所有电源断开，但是伺服电机上抱闸的电源没有断开，必须释放急停按钮并且重新开启机器人，机器人才能重新运作。

机器人系统里有几个急停按钮可以来紧急停止机器人，示教器和电控柜上都有一个红色的按钮（如左图所示）。当然用户也可以根据需要自己设置急停按钮。
急停按钮必须安装在容易触碰到的位置，以便出现意外时可以紧急停止机器人。

注意

急停只能被用于紧急情况下急停机器人，不能用于平常的程序停止，关闭机器人等。

危险

操作者需要注意伺服电机的动力线、连接夹具和其他装置的动力线的高电压危险。

程序人员：

在进行机器人的示教作业时，程序员在某些情况下需要进入机器人的动作范围内，尤其应注意安全。

注意

接通、断开使能是通过操作一个在示教器上的 Mot 按钮，当按下时，伺服电机上使能；当断开时，伺服电机断开使能。

为了确保安全使用示教器，需要遵守下面规则：

- 确保使能按钮在任何时候都有效。
- 在暂时停止机器人、编程或者测试时，使能需要及时断开。
- 示教者在进入机器人工作区域时，需要带着示教器，避免其他人在编程者不知情时操作机器人。
- 示教器不得放在机器人工作范围内，以防机器人运动时碰到示教器引起异常动作。

维修人员：

(1) 注意机器人中容易发热的部件

正常运作的机器人部分部件会发热，尤其是伺服电机，减速机部分，靠近或触碰容易造成烫伤。在发热的状态下必须触碰部件时，应佩戴耐热手套等保护用具。

用手触摸这些部分前先用手靠近这些部分感受其温度，以防烫伤。
在停机后等待足够时间让高温部分冷却下来再进行维修工作。

(2) 关于拆卸部件的安全注意事项

在确认齿轮等内部零件不再旋转、运动后打开盖子或保护装置，在齿轮、轴承等旋转时不能打开保护装置。如果有必要，使用辅助装置使内部不再固定的零件保持它的原来的位置。

在维修、安装、保养等服务后的第一次测试需要遵循下面的步骤：

a) 清理机器人和机器人工作范围内的所有维修、安装工具。
b) 安装好所有的安全措施。
c) 确保所有人站在机器人的安全范围之外。
d) 测试时要特别要注意维修的部件的工作情况。

在维修机器人时，禁止把机器人作为梯子，不要爬上机器人，以防摔落。

(3) 关于气动/液压的安全注意事项

在关闭气源或者液压泵后，气压/液压系统中存在残留的气体/液体，这些气体/液体有一定的能量，要采取一定的措施防止残留的能量对人体和设备造成伤害，在维修气压和液压元件前，需要把系统中残留的能量释放掉。

为防意外，需要安装安全阀。

(4) 虽然故障诊断时需要开启电源，但在维修机器人时务必要关闭电源，切断其他电源连接。

(5) 抱闸检测

正常运行中，抱闸通常会磨损，这时需要对抱闸进行检测。具体步骤如下。

a) 让机器人各个关节动到关节承受最大负载的位置。
b) 关闭机器人，抱闸工作。
c) 对各关节做标记。
d) 过段时间看机器人各关节是否活动。

(6) 加润滑油时的安全事项

当给减速机加润滑油时，对人身、设备都有可能造成伤害，所以在进行加油工作以前，必须遵循以下的安全信息。

- 在进行加油或放油工作时要戴防护措施（手套等），以防高温油液或者减速机对维修人员造成伤害。
- 打开油腔盖时需谨慎，油腔内可能存在压力造成溅射心，务必远离开口。
- 加油应根据油量表操作，禁止加满，完成后应检查油液指示口。
- 不同型号的油不能加入同一减速机，更换不同型号油前，需将残余油液清理干净。
- 放油要放完全或者在加完油后要检查油液指示口。

在放空减速机内油液前，可以先运行机器人一段时间加热油液，放油更容易。

刀具、外围设备的安全事项

在机器人关闭后，机器人外接设备有可能还在运行，所以外接设备的电源线或者动力电缆损坏也会对人身造成伤害。

机器人的安全事项

在紧急的情况下，机器人的任何一个臂夹到操作人员了，均需要拆除。安全拆除相关问题详情请询问我司技术人员。

小型机器人手臂可以手动移除，但是大型机器人需要用到吊车或者其他小型设备。

在释放关节抱闸之前，机械臂需要先固定，确保机械臂不会在重力作用下对受困者造成二次伤害。

机器人的停止方法

机器人有如下 3 种停止方法。

断电停止

这是断开伺服电源，使得机器人的动作在一瞬间停止的机器人停止方法。由于在机器人动作时断开伺服电源，减速动作的轨迹得不到控制。

通过断电停止操作，执行如下处理：

- 发出报警后，断开伺服电源。机器人的动作在一瞬间停止。
- 暂停程序的执行。

对于动作中的机器人，通过急停按钮等频繁进行断电操作，会导致机器人的故障。应避免日常情况下断电停止的系统配置。

报警停止

这是机器人系统发出报警（断电报警除外）后，通过控制指令使机器人的动作减速停止的机器人停止方法。

通过控制停止，执行如下处理：

- 机器人系统因过载、故障等原因发出报警（断电报警除外）。
- 伺服系统发出“控制停止”指令，减速停止机器人的动作，暂停程序的执行。
- 断开伺服电源。

保持

这是维持伺服电源，使得机器人的动作减速停止的机器人停止方法。

通过保持，执行如下处理：

- 使机器人的动作减速停止，暂停程序的执行。

警告、注意标签

（1）电击警示标识

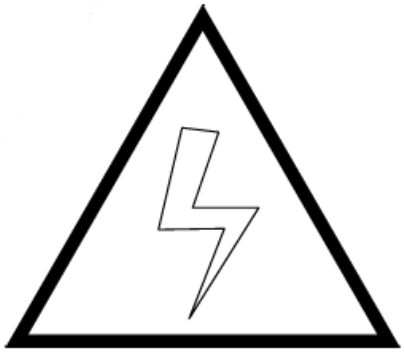


图 0.1 电击警示标识

贴有此标签处有高压、电击危险，应予注意。

（2）高温警示标识

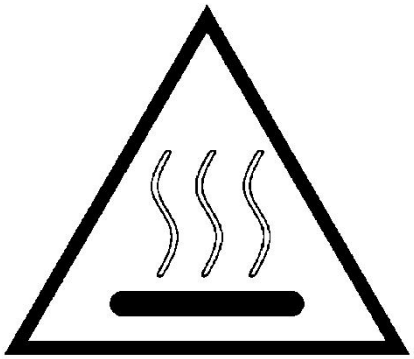


图 0.2 高温警示标识

贴有此标签处会发热，应予注意。在发热状态下必须接触设备时，应佩戴耐热手套等防护用具。

（3）禁止踩踏标识

图 0.3 禁止踩踏标识

不要将脚搭放在机器人上，或爬到机器人上面。踩踏会造成设备不良影响，也可能造成作业人员伤害事故。

（4）机器人伤人警示标识

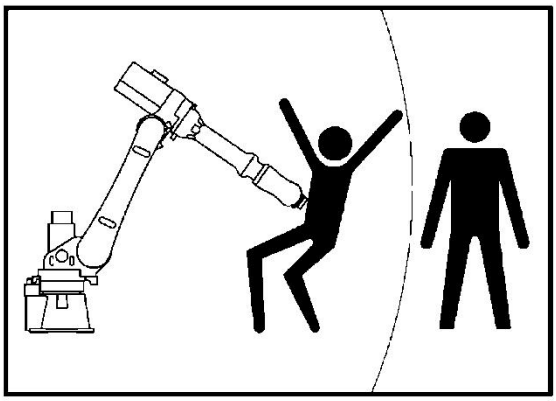


图 0.4 机器人伤人警示标识

在机器人动作范围内作业有受到机器人伤害的危险。

（5）禁止拆卸标识

图 0.5 禁止拆卸标识

贴有此标志的部位禁止用户对其进行拆卸。应由专业人员使用专业工具进行拆卸。

前言

本文档对 PC 端编程软件进行软件功能的分解，对每个功能模块进行简要说明。旨在帮助操作人员了解该软件。

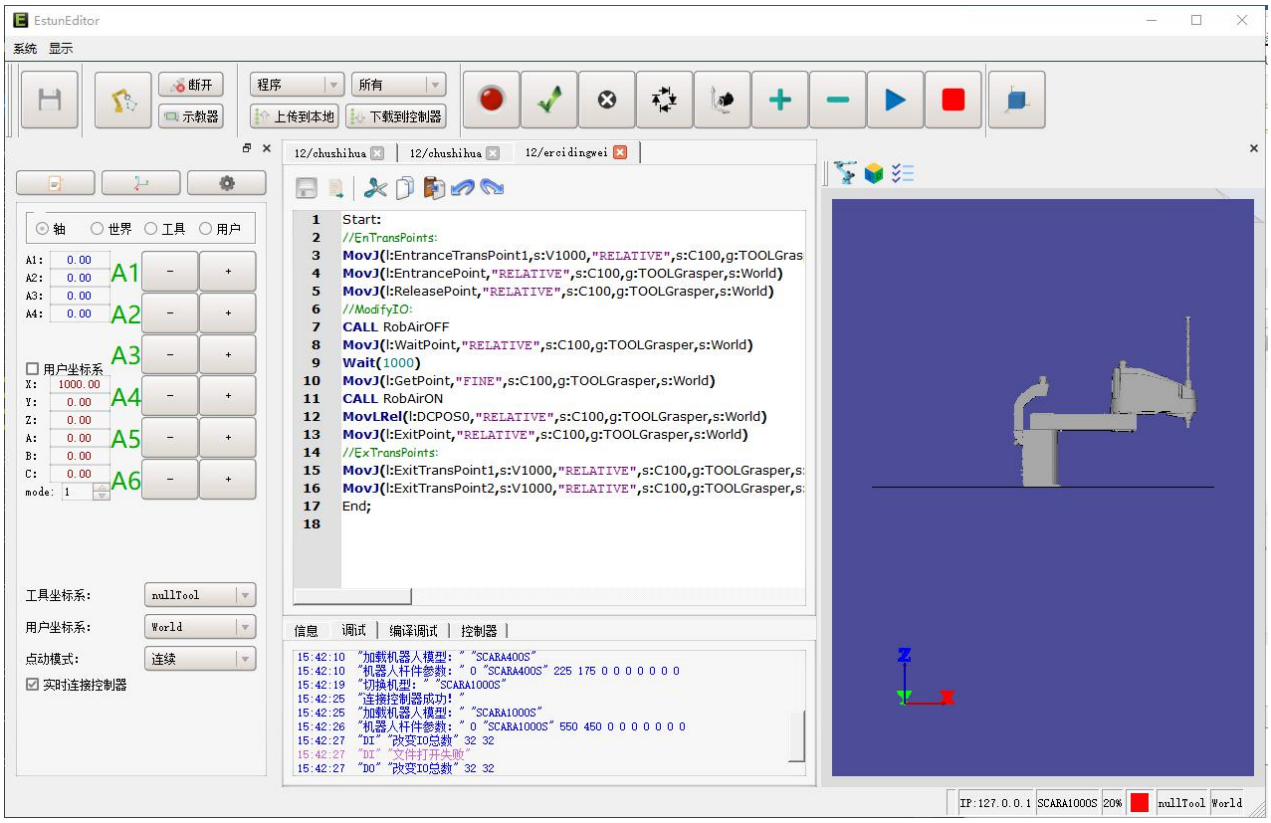
目录

目录

1.启动系统

1.1. 系统

打开软件，点击界面左上角的“系统” 系统 按钮，可设置语言、权限、打开说明书。



1.1.1.中英文切换

点击“语言”，选择对应的语言，重启程序生效。

1.1.2.权限

点击权限，选择“管理员”或者“用户”。

选择管理员后，需要输入密码，初始密码是“000000”。

再次选择管理员后，可以修改密码。

选择用户，权限直接下降到用户权限。

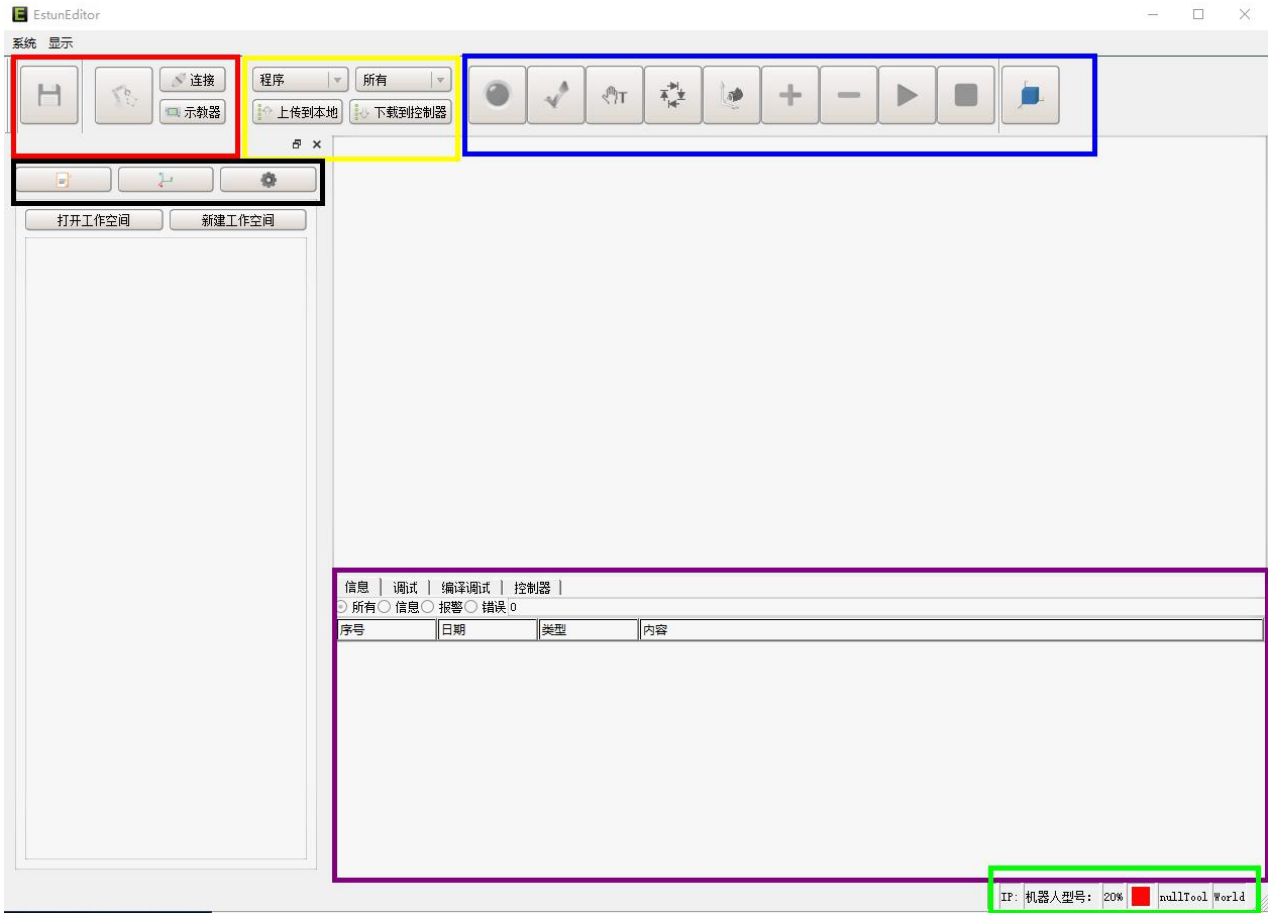
1.1.3.说明书

点击说明书，即可弹出软件的使用说明。

2.工程管理功能

2.1. PC 端软件主 UI 各功能简介

下图为软件的主 UI 界面：



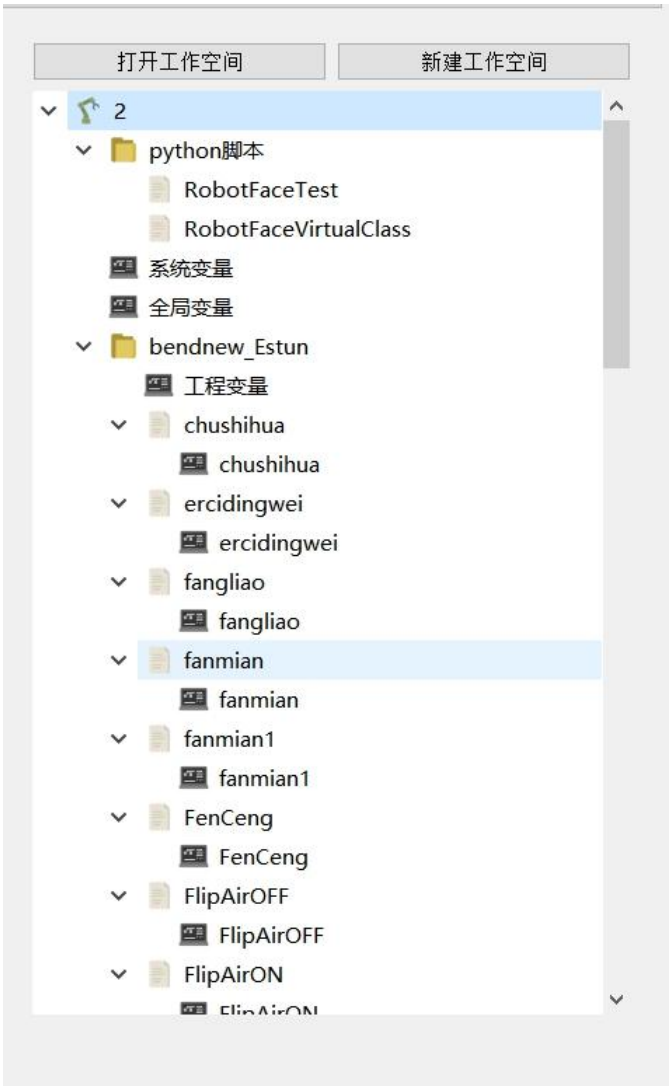
1. 红色标注区域为工具栏区域：分别是工程编译保存按钮、控制器连接断开按钮、虚拟示教器按钮。
2. 黄色标注区域为 FTP 上传下载模块。
3. 蓝色区域为虚拟控制器交互模块，包括：上使能、复位、系统模块切换、运行模式切换、坐标系切换、机器人速度设置、启动停止操作、三维显示按钮等。
4. 紫色标注区域为日志模块，包含不同的日志输出类别，用户可以根据需要进行切换显示。
5. 黑色部分是导航栏模块 分别是工程管理，手动，IO 设置模块的切换按钮。点击相应的按钮，会展示不同的导航模块。点击 UI 左上角的显示可以选择弹出或者隐藏导航页面
6. 绿色的是部分信息显示模块。包含 IP 机器人型号 机器人速度 当前坐标系信息。

2.2. 工程管理功能介绍

方便用户以及技服人员现场进行操作，通过下载控制器工程（连接控制器之后）以及通过编辑电脑

本地工程可以方便地实现工程程序的增删查改等功能。待修改编辑完成后，根据需求，可通过上传功能，将本地工程上传到控制器进行运行。单击  图标即可弹出工程管理页面。

功能页面如图所示。



1. 可以打开以及新建相应的工作空间文件夹（每个工作空间可以理解为不同的机型以及控制器参数）工程管理模块的根节点就是一个工作空间，右击节点可以新建工程。
2. 工作空间内可以显示目前控制器里面的 python 脚本文件，用户可以根据需要进行查看以及编辑。
3. 进行工程以及程序的增删查改（重命名、删除、新建）。
4. 双击程序以及变量相应节点进行程序以及变量编辑页面的查看。
5. 右击程序节点 可以进行程序的加载以及程序卸载操作，当加载成功后，程序节点图标会切换为选中状态。

2.3. 工程管理图标动态变换说明

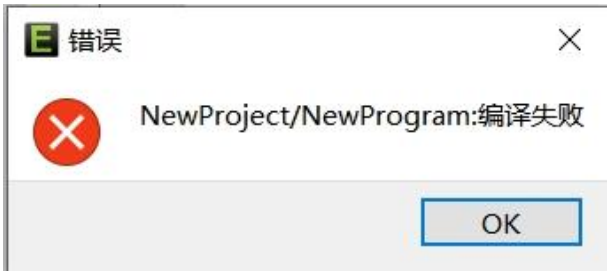
在进行当前工作空间内的编辑操作的时候，如果用户对变量节点以及程序节点进行编辑，当前工作管理的相应节点的图标会发生动态变化：

1. 变量编辑状态图标：当打开变量节点进行变量的增删改操作，此时工程管理相应的相应节点图标将会变成  。

2. 程序编辑状态图标：当进行指令程序编辑操作的时候，如果程序未报错，则图标会切成  。如果程序有错误，则会切成  。

3. 当前工作空间的节点，只要有节点变成上述图标，则软件左上角的编译保存图标会从灰化状态  变成可操作状态  ，此时用户点击图标进行编译保存操作，即可进行所有编辑未保存节点的编译保存。

4. 如若保存成功，则所有的变量以及程序编辑节点都会成初始加载图标状态；如果弹窗提示有程序编译失败，如图所示：



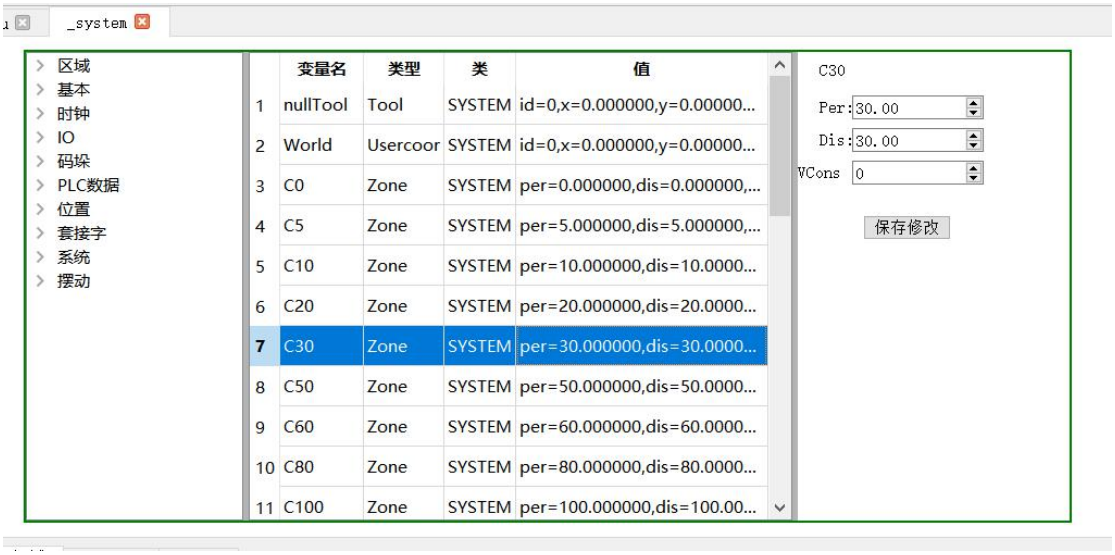
则用户需要对相应的程序进行重新编辑，无编译错误之后，再进行保存操作。

2.4. 变量模块说明

以下所有的变量功能说明，如果产生了变量的增删改的操作，最后都需要用 ctrl+s 快捷键、点击同步修改到文件、或者点击软件左上角的工程编译保存按钮进行保存。用户亦可通过观察变量节点图标变化，确认当前变量节点的是否保存。

2.5. 变量查看

打开或者新建一个工作空间之后，通过鼠标双击资源树相应的变量节点图标，即可在右侧弹出该变量文件相对应的 UI 页面。



2.6. 变量新建/删除

在变量管理界面，可通过变量编辑 UI 区域的左侧树状节点进行变量类型选择，进行双击添加，新建变量。

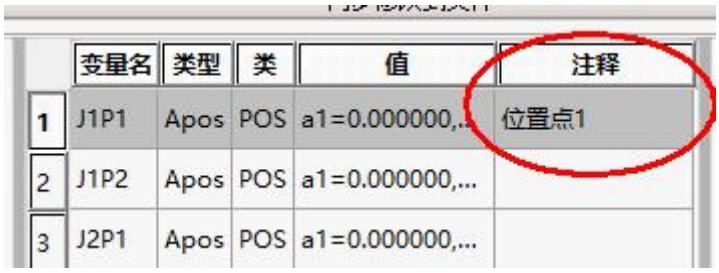
在变量管理界面，可选中相应的变量行，鼠标右击进行删除，删除已有变量。

2.7. 变量重命名

可通过鼠标双击表格内的变量名单元格进行变量名重命名操作。

在相同作用域内，已存在的变量名无法用于变量重命名操作。

2.8. 变量注释



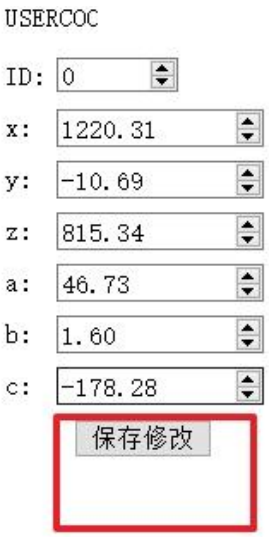
可通过鼠标双击表格内的变量注释所在的单元列，进行相应的变量注释。最后进行保存，即可完成变量注释，后续用户打开该工程的时候 该注释都能看见

2.9. 位置变量示教

在变量页面的属性编辑页面栏，CPOS 或者 APOS 变量会有一个示教按钮，点击示教之后，就会同步示教操作页面的位置信息到该变量页面，然后点击保存完成变量修改。

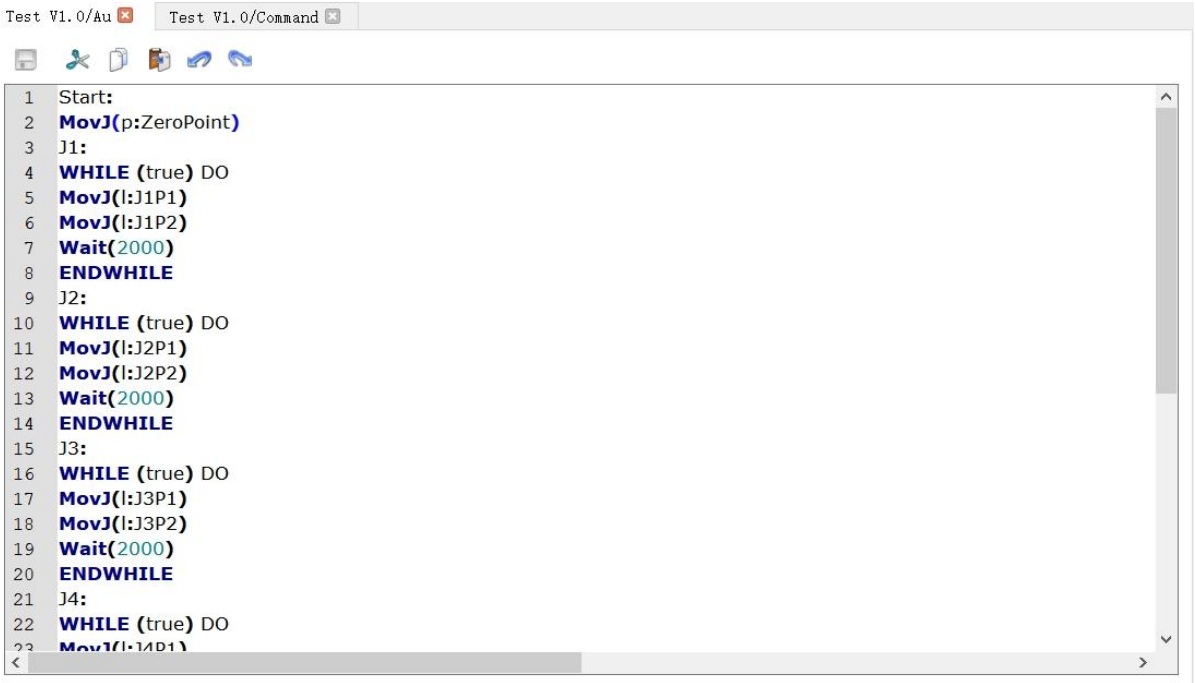
2.10. 变量属性编辑

鼠标单击相应变量行，页面右侧会显示该变量的当前属性值，用户可以在属性页面可以方便对相应的属性进行修改；修改完成后，需要点击保存按钮，保存到变量表格数据中（进行属性页面修改之后，一定要点击保存修改按钮，否则不生效）如图所示。



2.11. 程序编辑

用户可以通过此软件方便地进行程序编辑修改。程序编辑 UI 如下所示。

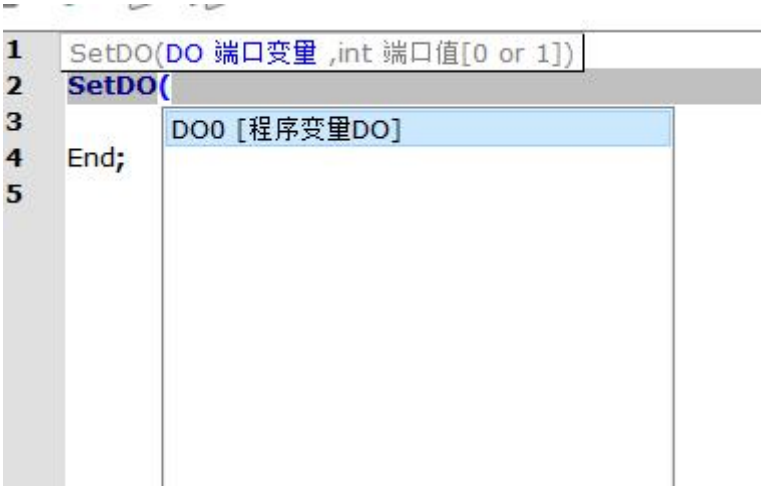


1. 用户可以通过双击工程管理树的程序节点打开相应的程序进行编辑
2. 程序编辑区域提供灵活方便的文本编程编辑操作，支持常用的文本快捷操作：ctrl+c 复制、ctrl+x 剪切 ctrl+v 粘贴 、ctrl+z 回退、ctrl+y 前进等操作。也可以通过点击程序编辑 UI 区域的工具栏进行相应操作。

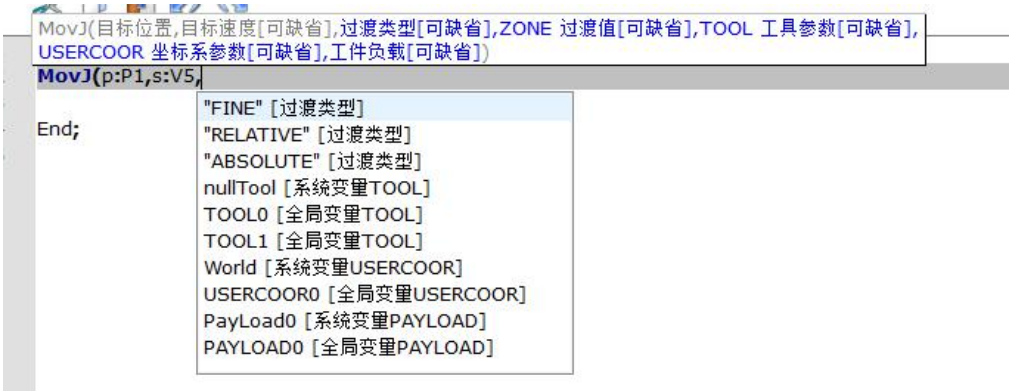
3. 支持程序指令、关键字等高亮显示。

4. 支持代码输入自动提示功能（工程范围内的变量、指令等列表供选择）并进行 tip 提示，让用户一目了然进行选择输入。

5. 在进行指令函数的参数输入时，输入左括号后，会弹出该函数参数输入提示，以及自动筛选当前输入参数的自动选择列表。每次输入逗号进行参数输入，都会更新当前指令参数的提示列表，并且会点亮函数提示框相应的参数提示，供用户进行参考输入



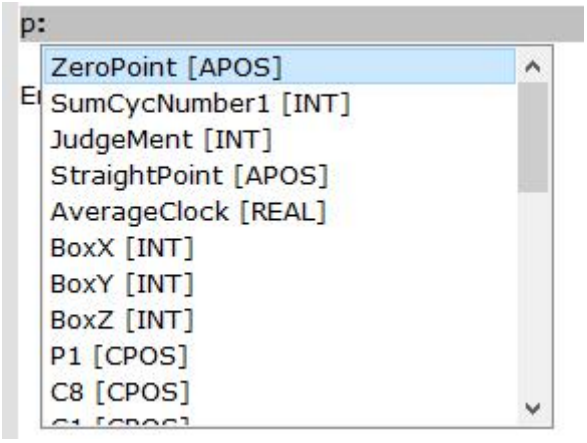
6. 对于多种参数输入组合的指令 例如 movJ movL 等指令，里面会包含各种可选以及必选参数（虽然是可选，但是各参数类型的顺序都是有所依赖的）每输入一个参数之后，编辑器都会更新当前可输入参数列表的输入提示，并且会在函数提示窗口点亮目前可输入的所有参数列表。这样能及其高效，一目了然地帮助用户应付此类参数组合的指令编辑。



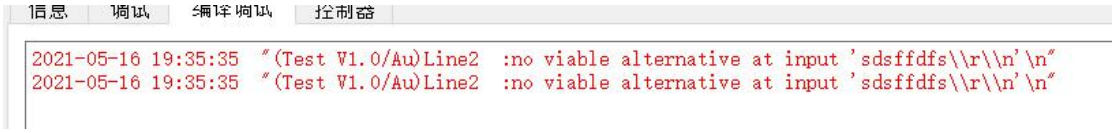
7. PC 端软件区别于示教器编程的 UI 界面显示有个不同点，就是变量的显示。例如我从输入提示框选择

一个变量进行输入的话，那么在编辑器中他呈现的应该是 s:，p:，g:,l:开头的形式。s:代表系统变量，p:代表工程变量，g:代表全局变量，l:代表程序变量。

另外为方便用户筛选不同域里面的所有变量，可支持在用户手动输入 s:，p:，g:,l:之后，自动在输入提示框更新弹出相应域里面的所有变量供用户选择输入。



8. 该程序编辑器能够在用户输入的时候实时报错一些基本错误信息,在下方的编译调试区域进行显示报错信息，或者能在错误行进行红色高亮显示提醒。



```
1  Start:
2  MovJ(p:ZeroPoint)
3  Wait(l:J1P1)
4  J1:
5  WHILE (true) DO
6  MovJ(l:J1P1)
7  MovJ(l:J1P2)
8  Wait(2000)
9  ENDWHILE
10 J2:
11 WHILE (true) DO
12 MovJ(l:J2P1)
13 MovJ(l:J2P2)
14 Wait(2000)
15 ENDWHILE
16 J3:
17 WHILE (true) DO
18 MovJ(l:J3P1)
19 MovJ(l:J3P2)
20 Wait(2000)
21 ENDWHILE
22 J4:
23 WHILE (true) DO
```

信息 调试 编译调试 控制器

```
2021-05-16 19:38:06 "(Test V1.0/Au)Line3 :参数[1]是无效类型[APOS],正确类型应该是如下类型{ ConstInt.}\n"
2021-05-16 19:38:06 "(Test V1.0/Au)Line3 :参数[1]是无效类型[APOS],正确类型应该是如下类型{ ConstInt.}\n"
```

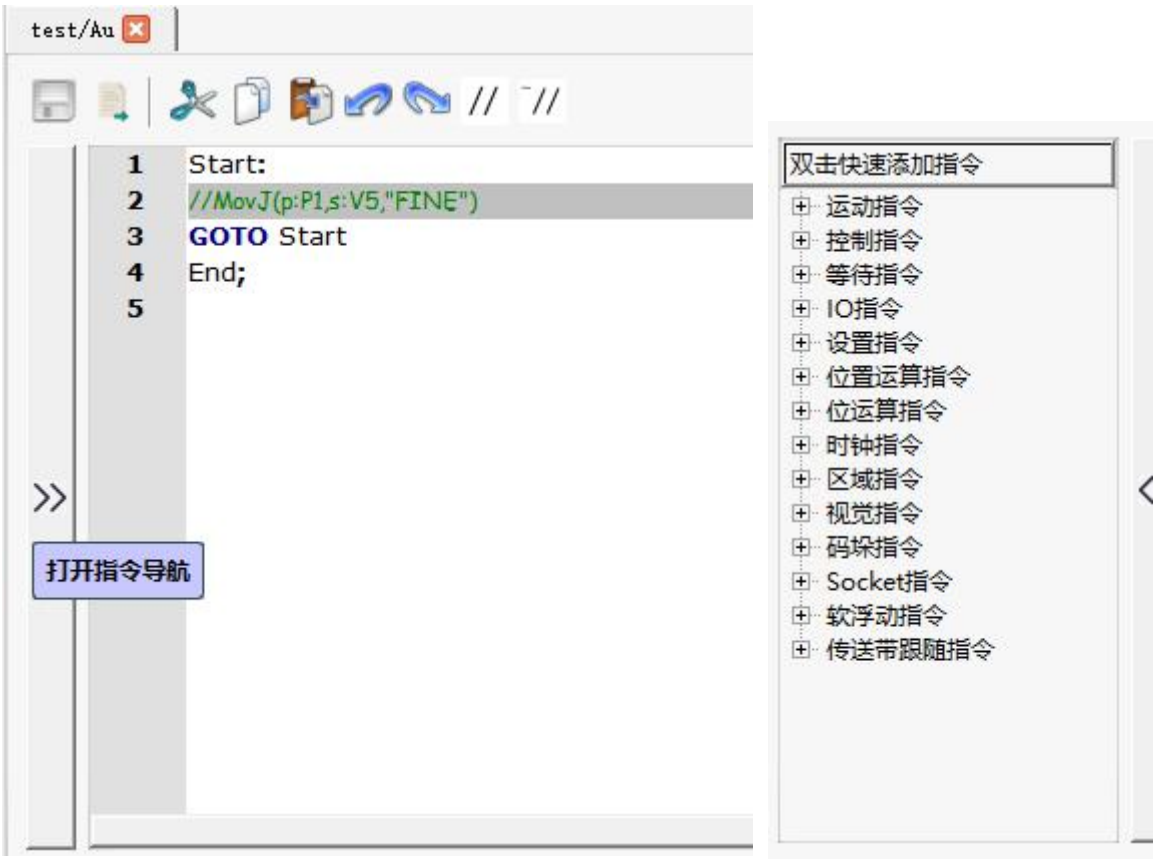
9. 编辑输入完成后 点击  图标，即可对当前编辑的程序进行编译保存，此时也能校验一些语法错误，如有错误显示，保存不成功，用户需要按照提示进行相应的程序修改，待检查无误后，就能编译保存成功。

10. 编辑区域可通过// 或者/* */进行代码注释。

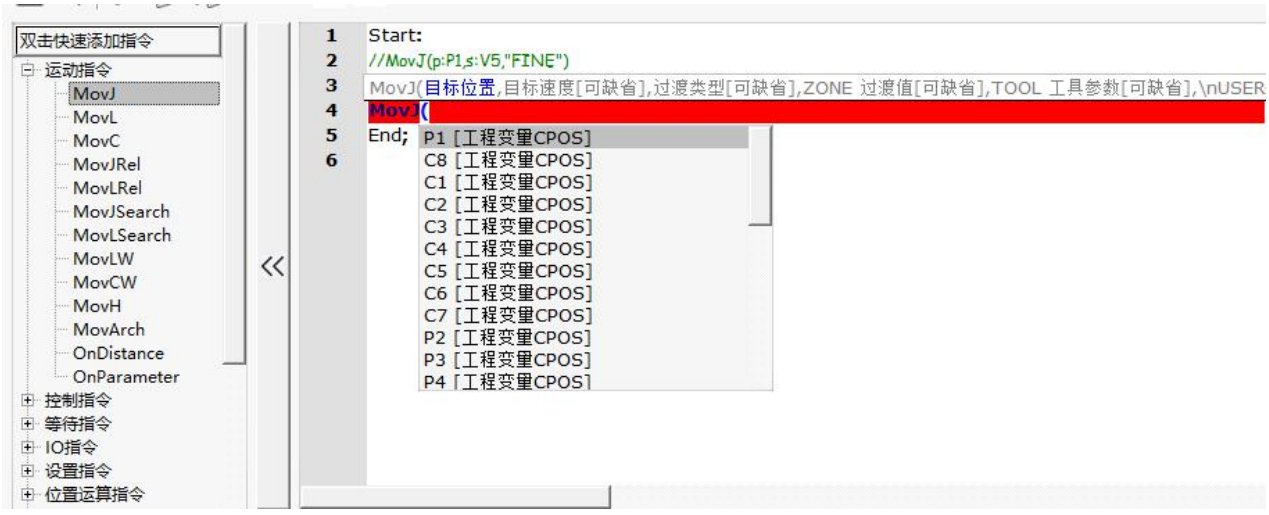
```
Start:
//先运行Scope
/*循环变量初始化*/
l:CycX.value = 1
l:CycY.value = 1
l:CycZ.value = 1
l:CycCircle.value = 1
/*设定间隔*/
l:SpaceX.value = 10
l:SpaceY.value = 10
l:SpaceZ.value = 10
l:SpaceCircle.value = 10
/*设定循环上限*/
l:MaxCycX.value = p:BoxX.value / l:SpaceX.value
```

2.12. 指令导航

单击如下图所示的按钮，打开指令导航窗口



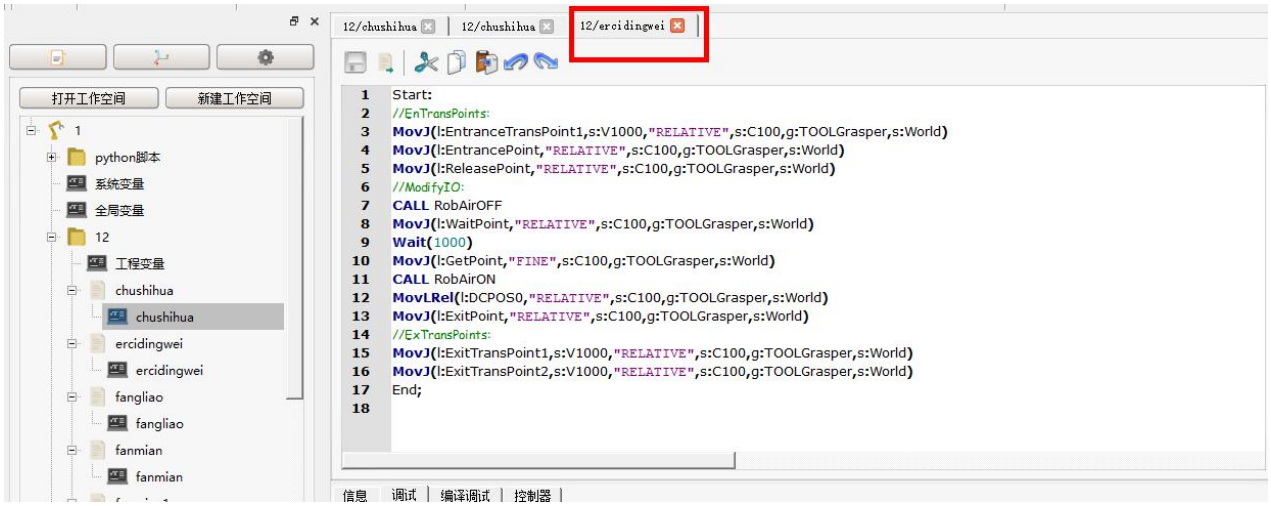
然后双击展开各类指令列表的节点，会有相应的指令。光标移动到编辑区域相应位置，然后双击该指令，即可完成快速添加的操作。



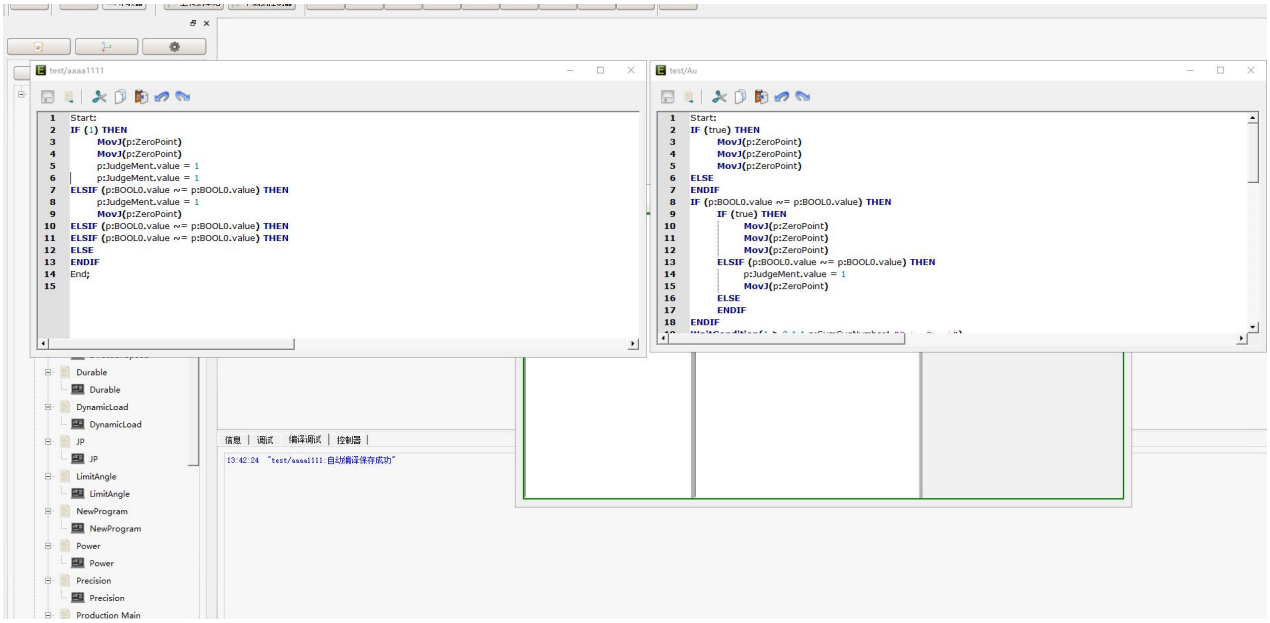
再次点击打开关闭按钮 即可关闭指令导航窗口。

2.13. 多文档多窗口编辑功能

在双击树节点相应的程序之后，为方便用户进行各个编辑窗口的单独编辑，用户可以选择将鼠标移动到如下图所示的红色区域，鼠标单击按住不放，可以将该页面拖拽出主窗口进行单独的编辑查看。

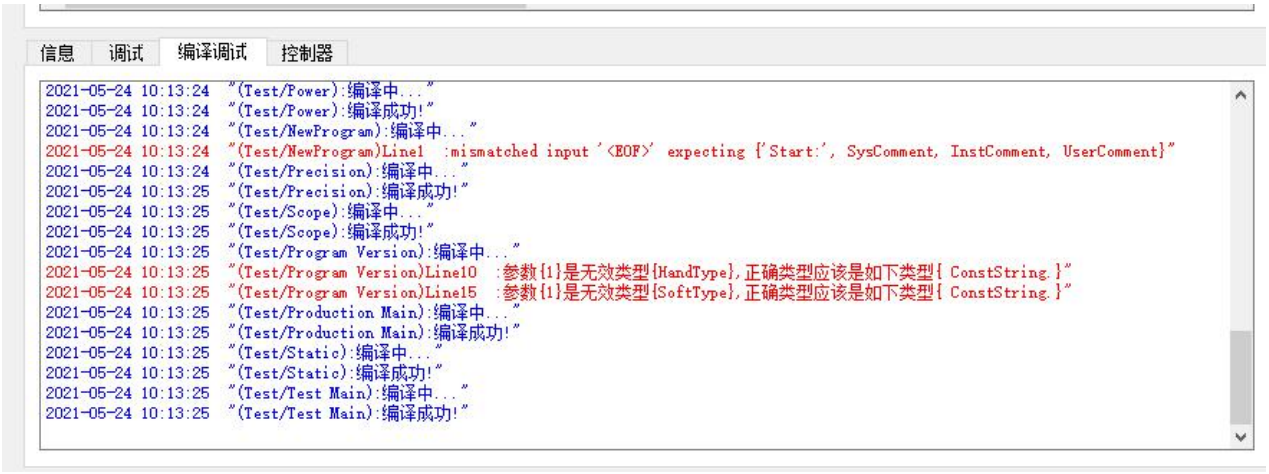


下图就是拖拽出主 UI 的窗口显示，此时就可以同时进行多文档的编辑了。



2.14. 程序编译

除了支持程序单独编译交错，为方便用户在进行工程编辑的过程中，整体进行工程的校验编译。用户可以鼠标右击工程管理模块的工程名称节点， 在弹出的菜单中选择“编译工程”，这样就能进行工程的整体编译，编译信息会在编译调试窗口进行输出提示，方便用户查看各程序的编译结果。



2.15. 运行中程序指针显示

系统单步运行或者连续运行中，可显示当前运行程序指针行。

```
1  Start:
2  MovJ(p:ZeroPoint)
3  J1:
4  l:TemporaryCycNumber.value = 1
5  WHILE (l:TemporaryCycNumber.value <= 10) DO
6  MovJ(l:J1P1)
7  MovJ(l:J1P2)
8  Wait(2000)
9  l:TemporaryCycNumber.value = l:TemporaryCycNumber.value + 1
10 ENDWHILE
11 J2:
12 l:TemporaryCycNumber.value = 1
13 WHILE (l:TemporaryCycNumber.value <= 10) DO
```

程序运行过程中，可通过打开虚拟示教器同步查看指针行显示，对比有无差异。

2.16. 程序指针重设

在程序编辑 UI 的上方菜单栏有个设置指针的操作按钮，如图所示。用户可以将鼠标光标放到当前已加载程序或者正在执行程序的相应行，然后点击该按钮，即可实现指针的重设。

2.17. 工程保存

1. 点击主程序左上角的保存图标，将正在修改的变量以及程序一起同步保存到本地。
2. 可将修改的变量页面以及正在编辑程序页面统一编译保存。
3. 如果在编译调试栏有错误报错信息，则表示程序有错误编辑输出，则保存编译失败。提示用户进行修改后在进行保存操作。

3.设置

3.1. IO 状态显示

在导航栏界面上点击设置按钮，然后点击“IO”进入 IO 检测界面

如下图所示。IO 的相关数据是通过主界面传送过来的。

3.1.1.数字量

进入 IO 检测界面，默认显示的是“数字量”列表的界面，可查看 IO 的工作情况。

用户可通过该界面查看控制模块上的 IO 状态：绿色表示为有信号；灰色表示为无信号。

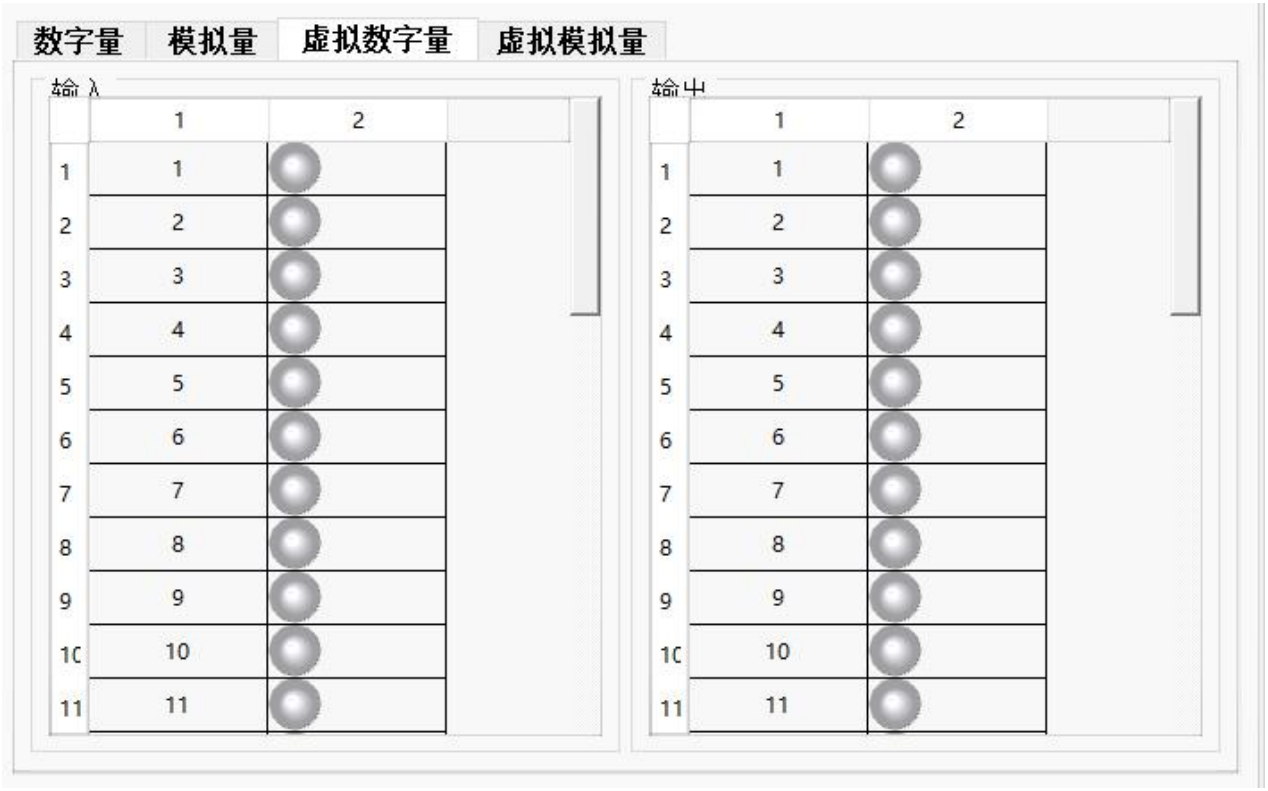
用户可通过该界面查看实际 IO 的状态。

3.1.2.模拟量

用户可以通过该界面查看实际模拟量状态。

3.1.3.虚拟数字量

用户可以通过该界面查看虚拟 IO 的状态。



3.1.4.虚拟模拟量

用户可以通过该界面查看虚拟模拟量状态。

3.1.5 设置标签

所有的信号都可以设置标签。

3.2. 配置文件设置

3.2.1.查看设置主界面

在主页界面上点击“配置文件设置”进入系统设置界面，如下图所示。

当前界面的参数都是从配置文件中读取出来的 conveyor.ini，motionCfg.ini，systemCfg.ini。这三个配置文件中获取，后续修改界面中的数据在点击 save 按钮后也会保存进配置文件中。

3.2.2.传送带配置

点击“传送带配置”，如下图所示，用户可以修改对应的参数并保存。

3.2.3.网络设置

点击“网络设置”，如下图所示，用户可以修改对应的参数并保存。

3.2.4.视觉设置

点击“视觉设置”，如下图所示，用户可以修改对应的参数并保存。

3.2.5.碰撞检测

点击“碰撞检测”，如下图所示，用户可以修改对应的参数并保存。其中的数量是可以动态设置的，需要主界面调用的时候设置对应的数量就行了。

3.2.6.振动抑制

点击“振动抑制”，如下图所示，用户可以修改对应的参数并保存。其中的数量是可以动态设置的，需要主界面调用的时候设置对应的数量就行了。

3.2.7 寸动设置

点击寸动设置设置每个轴的寸动距离或角度。

4.控制器模块

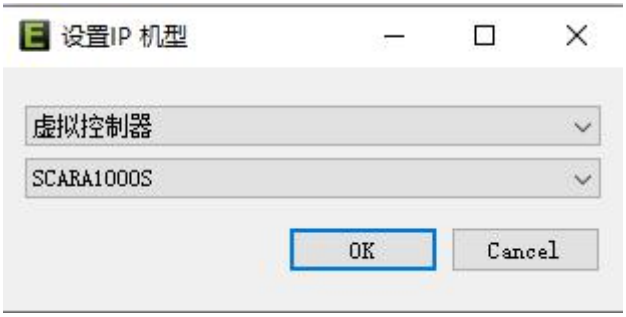
4.1. 控制器启动

4.1.1.虚拟控制器

点击红色区域以及黄色区域的控制器启动连接按钮：

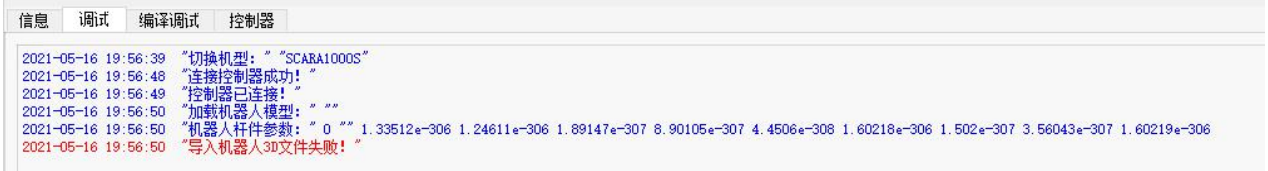
选择“虚拟控制器”或者“实际控制器”。

虚拟控制器，选择需要的机型，点击确定。



虚拟控制器的启动时间要比实际控制长，因为需要在电脑中启动一个虚拟控制器，而实际控制不需要。所以，有时会遇到虚拟控制虽然启动了，“控制器操作模块”变为可编辑状态。

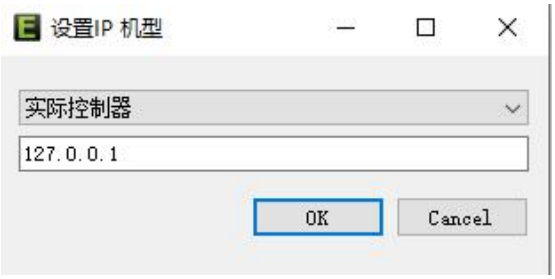
但是日志显示“加载机器人模型”为空，导致三维模块不能确定加载什么机型，所以报错“导入机器人 3D 文件失败”。



此时需要点击“三维”模块，重新更新控制器状态，加载对应机型。有时需要点击几次，直到三维刷新出对应机器人模型。

4.1.2.实际控制器

实际控制器，修改实际控制器的 IP，点击确定。



如果 IP 正确，则耐心等待几秒，让三维模块加载对应机型。

备注：无论虚拟控制器还是实际控制器，启动成功以后，工具栏的“控制器操作模块”会变成可编辑状态。此模块后面会有详细介绍。

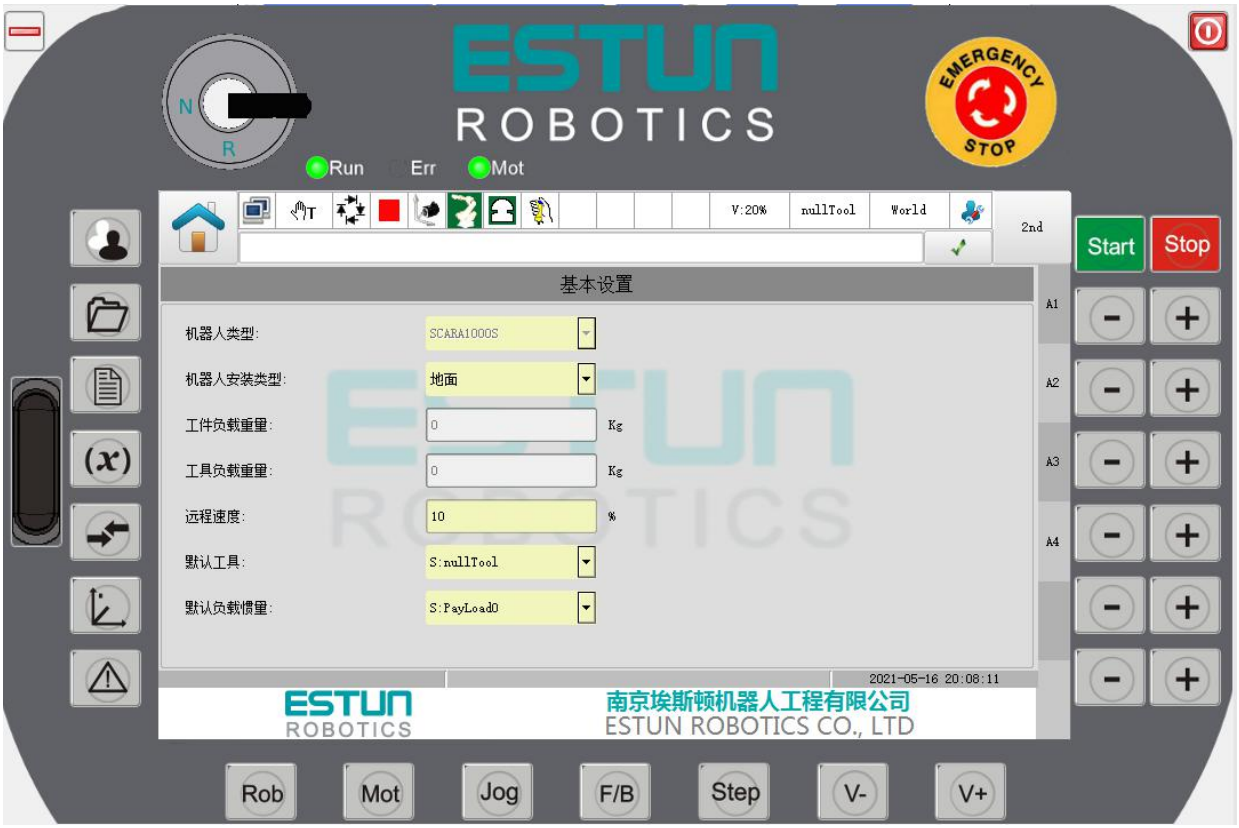
无论虚拟控制器还是实际控制器，启动成功以后，可以点击“示教器”来打开虚拟示教器，辅助查看控制器的状态。通常我们无需打开，只要通过“控制器操作模块”和底部的状态栏，即可同样操作看制器。

4.2. 控制器断开

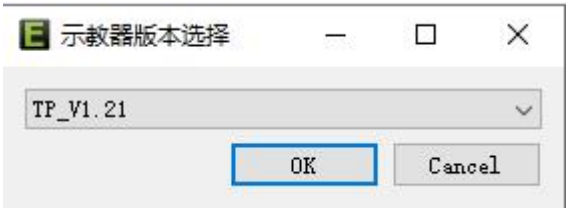
在启动控制器以后，可以点击“断开”按钮，断开与控制器的连接。如果是虚拟控制器，还会关闭后台的虚拟控制器。

4.3. 虚拟示教器

点击“示教器”按钮，启动虚拟示教器。



右键“示教器”按钮，可以选择示教器的版本。



如果启动的是实际示教器，需要修改 IP，可以在“设置”界面里面修改对应控制器 intime 的 IP，修改完，点击保存，关闭重启“虚拟示教器”。

4.4. 控制器操作模块

在连接控制器以后，控制器操作模块可以进行对控制的操作。控制器操作模块会和程序底部状态栏一起查看当前控制器的状态。

控制器操作模块有：

“伺服使能”：给控制器上下使能。

“复位”：复位控制器的错误。

“运行模式”：切换手动、自动、远程运行模式，其中远程模式是右键点击触发。

“单步/连续”：点动运行是单步还是连续。

“坐标系”：切换控制器的坐标系。

“加速”：控制器速度增加，右键出发，直接加速到 100%。

“减速”：控制器减速，右键出发，直接减速到 1%。

“启动”：启动当前加载程序。

“停止”：停止当前加载程序。

备注：这里的按钮功能，与虚拟控制器上对应的按钮功能基本一致。

状态栏含有：

1 控制器错误状态，上面“复位”按钮，即为复位此处控制器异常。

2 控制器 IP。

3 控制器机型。

4 控制器速度。

5 控制器运行状态，分为运行、暂停、停止，上面“启动”、“停止”按钮即关联到此处显示控制的运行状态。

6 控制器工具坐标系，可在手动示教器或者程序中设置。

7 控制器用户坐标系，可在手动示教器或者程序中设置。

5.示教器模块

示教器模块通常与三维模块，程序编辑模块一起使用。如：点动时，可以与三维模块一起查看机器人位姿；程序编辑时，给变量示教时，就是赋值当前示教器当前的坐标；在连接控制器后，“实时连接控制器”勾选时，示教器的点动即为虚拟示教器的点动等等。

5.1. 未连接控制器状态

未连接控制器状态是指没有连接控制器，或者连接控制器“实时连接控制器”没有勾选上。

首先，仍然是点击“手动”：

打开手动示教器。在手动示教器的顶部，是“轴”、“世界”、“工具”、“用户”坐标系的选择，选择不同坐标系，点动效果也会跟着对应改变。

坐标系选择的下方，就是点动按钮和坐标显示。特别提出的是，在没有连接控制器，或者连接控制器“实时连接控制器”没有勾选上时，可以点击“mode”来切换当前世界坐标系，在三维中可以清晰的看到此 mode 值时的机器人关节姿态。勾选用户坐标系，即切换世界坐标系坐标为当前用户坐标系下的用户坐标。

点动按钮下面的“工具坐标系”、“用户坐标系”。点击后可选择此时的工作空间下的用户坐标系和工具坐标系。

点动模式：可以选择“连续”或者“寸动”两种模式，寸动的参数可在设置中查看。

最下面的“实时连接控制器”，即在连接控制的情况下，去掉勾选，也会使示教器进入没有连接控制器的状态。

5.2. 连接虚拟控制器

连接虚拟示教器必须勾选“实时连接控制器”。与没有连接控制器状态不同的是：

1、顶部的坐标系已经不可以编辑，只能通过“控制操作模块”的“坐标系”按钮进行切换。

2、此时点动的速度也可以在“控制操作模块”中进行修改。

3、在切换底部坐标系时，控制器的坐标系也会随之切换。

5.3 连接实际控制器

与连接虚拟控制的区别是：

此时控制就是实际机器人，所以一切操作，都会实时生效到实际机器人。比如点动，机器人会跟着

跑。所以慎用！建议实际机器人选择去掉勾选“实时连接实际控制器”。

6.上传和下载功能

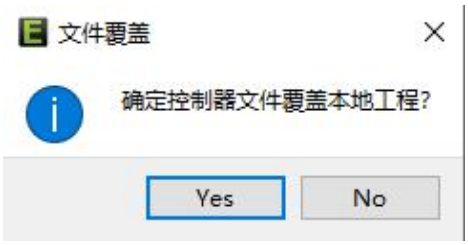
上传下载模块是在连接控制器以后，上传控制器的程序、脚本或者其他的系统配置，编辑完后下载到控制器中使用。

6.1. 文件上传

左上角可以选择上传控制器的程序部分、脚本部分、配置文件部分还是所有都上传。下载时同样依据此处的选择。

点击“上传到控制器”，就会将选中的部分，从控制器上传到本地。

上传完成后，弹出是否覆盖本地工程，如果选择是，则本地的文件相同的部分会被覆盖。比如相同程序名下的同名程序文件会被覆盖掉。选择否，则不对本地工程文件生效，只是上传控制器文件状态，此时可在“下载到控制器”按下以后，看到本地与远程控制器文件的所有差异。



6.2. 文件下载

可以选择文件下载类型（只对程序生效，配置文件和脚本都是直接覆盖，不会对比）：

覆盖：只显示程序文件差异的部分。

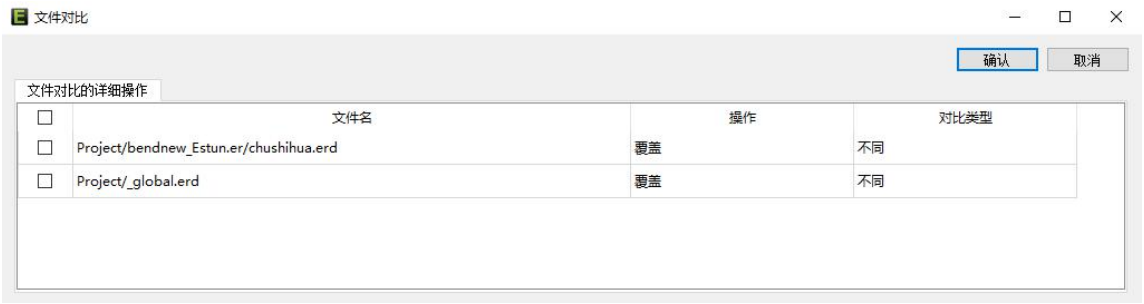
添加：只显示本地工程多出来的部分，选择是否添加到控制器。（谨慎添加，比如只添加程序文件，不添加程序文件对应的变量文件，控制器程序运行会报错）

删除：只显示控制器程序多出来的部分，选择是否删除。（谨慎删除，比如只删除程序文件，不删除

程序文件对应的变量文件，控制器程序运行会报错）

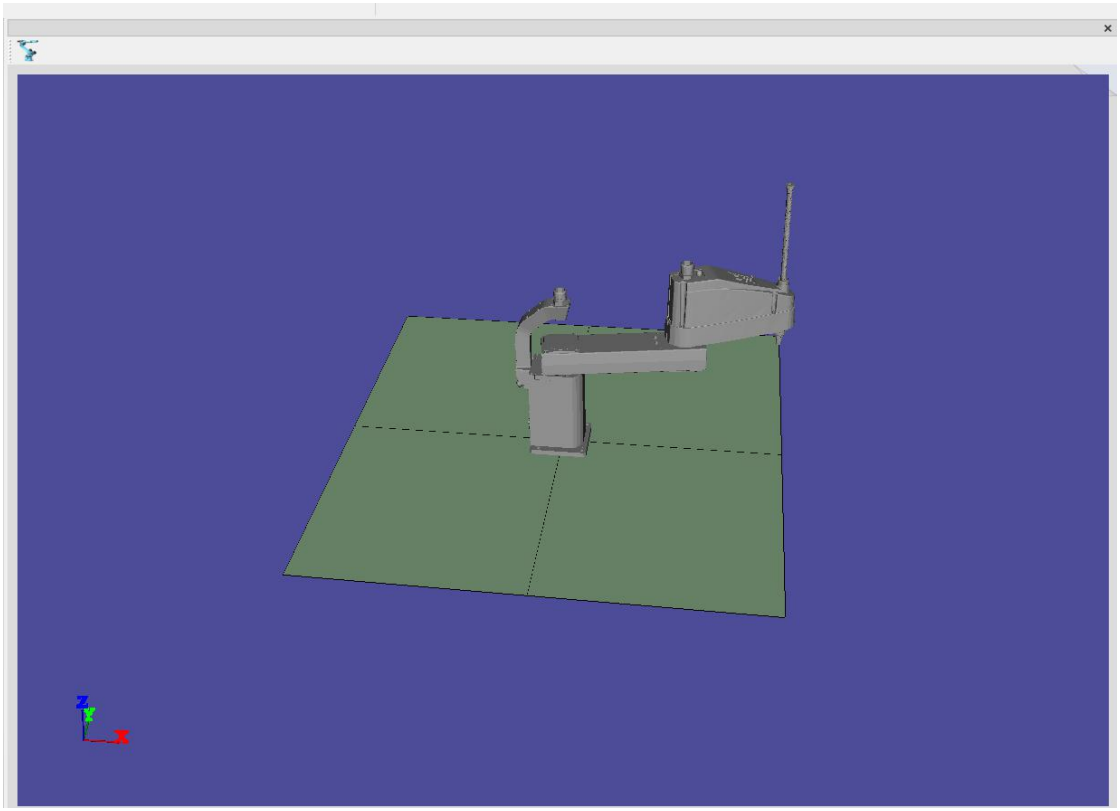
所有：三种同时对比显示。

对比弹窗弹出对应的文件对比，需要手动勾选是否下载。



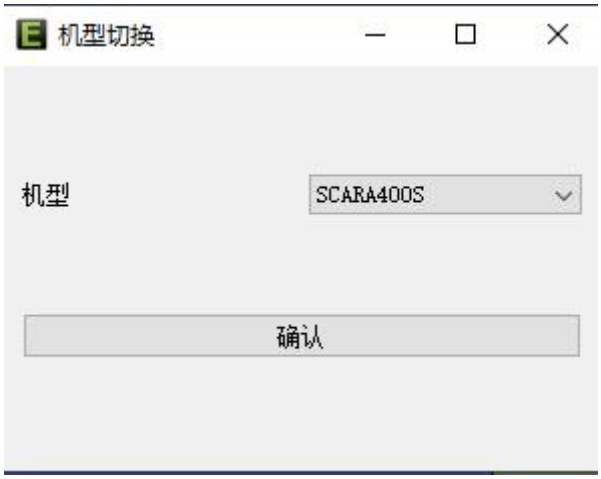
7.三维功能

点击“三维”  按钮，即可加载三维模块。

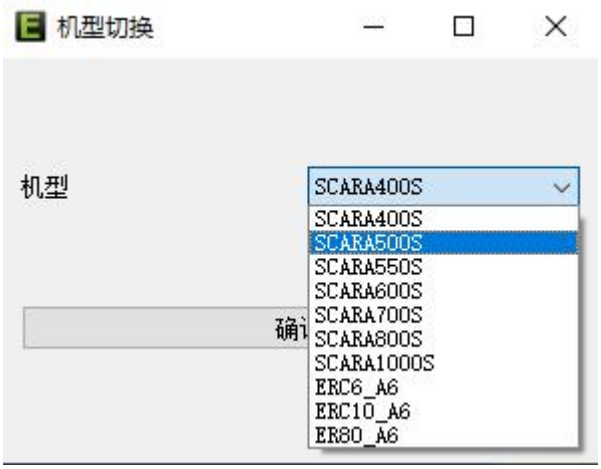


7.1. 机型加载和切换

点击三维中的“机型”，弹出机型切换的对话框。



目前支持所有 ESTUN 的 SCARA 机型，选择需要的机型，点击确定，即可切换三维到对应的机型。



备注：切换机型是离线状态下操作机器人。如果连接控制器，无论是虚拟还是实际控制器，只需要点击“三维”按钮，就会自动刷新到与控制器对应的机型。需要注意，连接控制器以后，不可随意切换机型，否则会造成三维空间中的机器人与实际机器人不一致！

7.2. 显示机器人的位姿

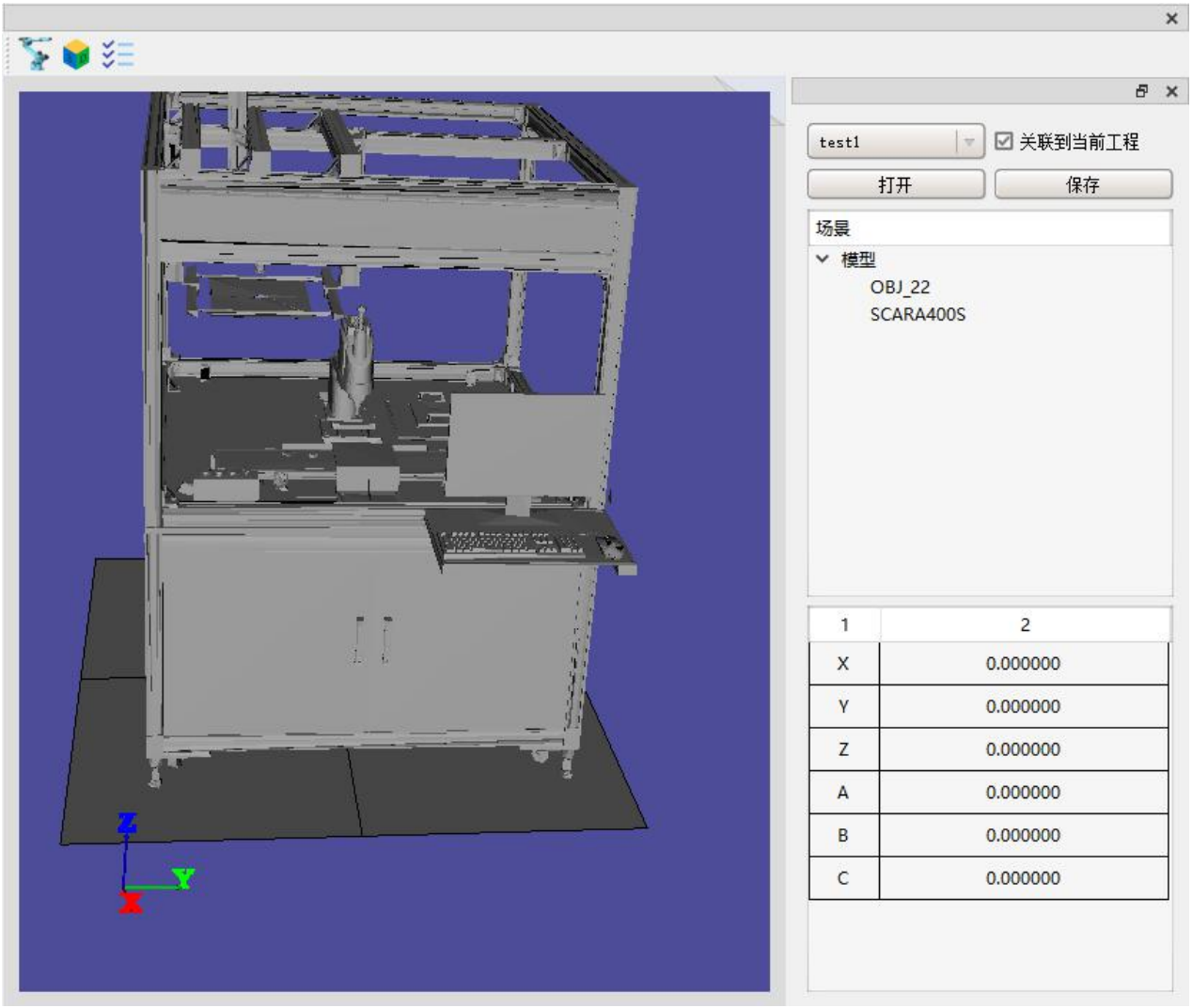
点击手动按钮：

左侧对话窗口即可显示手动示教器，可以通过此窗口，操作 3D 环境中的机器人。同样，在手动示教器的“实时连接控制”勾选上以后，通过“控制器模块”启动运行程序时，三维空间中的机器人将会实时显示控制器运行程序时的实时位姿。后面会有对手动示教器的详细介绍。


轴
世界
工具
用户
A1: 0.00
A2: 0.00
A3: 0.00
A4: 0.00
A1
A2
A3
A4
A5
A6
−
+
用户坐标系
X: 1000.00
Y: 0.00
Z: 0.00
A: 0.00
B: 0.00
C: 0.00
mode: 1
工作坐标系:
用户坐标系:
实时连接控制器

7.3 模型导入

对于需要搭建简单现场场景的，可以在三维中导入相关模型，来搭建三维环境。



点击“模型”按钮，

弹出“导入模型”弹窗：

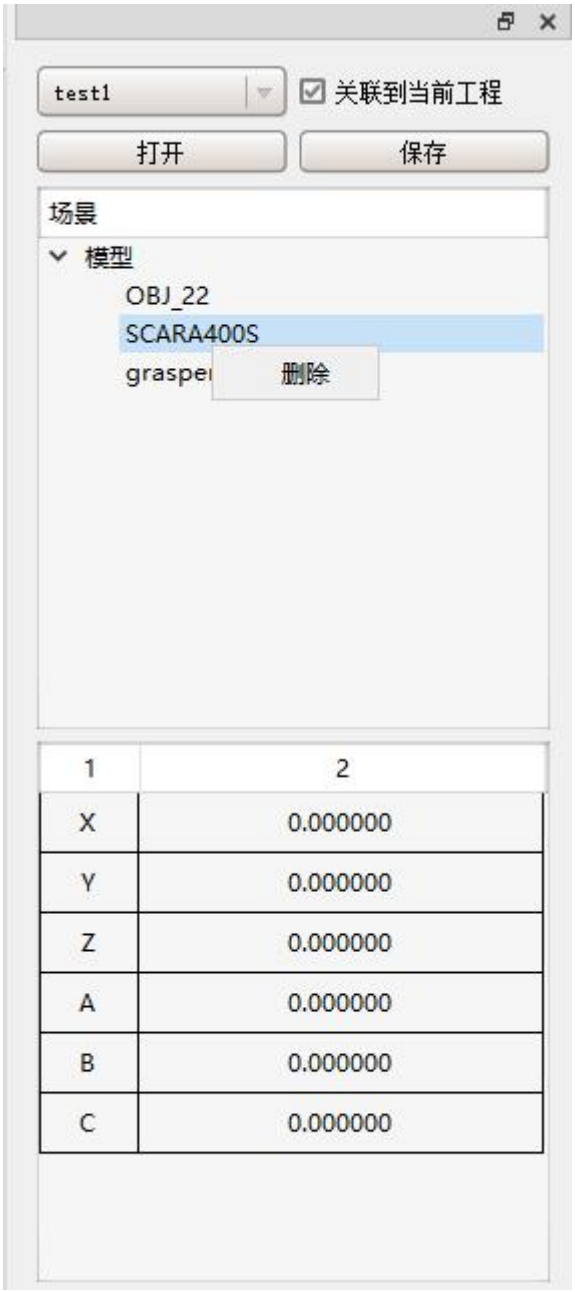
然后选择类型：其他表示普通的场景模型，工具代表可以安装到机器人法兰末端的夹爪模型，只能安装一个夹爪。
模型名称：不能为空。
在“模型文件”里，点击“选择文件”，选择需要导入的模型文件。（仅支持 STL 格式的文件，文件大小尽量控制在 2M 以下）

填写完参数以后，即可导入模型到三维场景中。

7.4 工程管理

点击工程管理按钮 

，右侧弹出工程管理树



关联到当前工程：勾选后，会将三维场景和当前打开的工程关联起来，下次打开工程后，可以选择自动加载此三维场景。

场景列表：在 Models 节点下面显示当前三维环境中的模型列表，双击列表中的模型名，会在底部的坐标栏中显示该选中模型在三维中的位置，可以在底部坐标栏中修改模型位置。右击列表种的模型，可以删除；如果右击列表顶部节点“Models”，可以删除整个项目。

打开：打开一个已有的三维场景，打开后，会在顶部显示当前打开的三维场景名称。

保存：保存当前场景到三维工程中。如果是临时工程“tmp”，需要重新命名。

坐标栏：显示场景中的模型位置，并可以修改。

8 日志功能

日志模块记录了用户的一些操作异常，给予相关的提示，对于我们使用软件提供一个帮助。同时，在遇到问题时，可以提供查看到的对应的窗口提示，向专业人员提供信息来得到帮助，解决问题。

8.1 控制器信息

点击日志窗口的“控制器”标签，即可显示控制器的一些输出，这里的输出暂时都是英文，作为较专业人员查看控制器问题的信息窗口。

8.2 编译调试信息

点击日志窗口的“编译调试”标签，即可显示程序编辑时的一些提示。通过这些提示，我们可以更轻松的定位到程序编辑时的问题。在编辑程序时，建议打开此窗口。

8.3 普通调试信息

点击日志窗口的“调试”标签，即可显示我们除了程序编辑提示外的一些操作提示，比如点动机器人，连接控制器等等。建议在非程序编辑操作时，打开此窗口。

8.4 所有信息

点击日志窗口的“信息”标签，即可显示软件对于所有的信息的一个分类。可以在这里点击“所有”查看所有提示信息，点击“信息”查看所有普通提示信息，点击“报警”查看所有警告，点击“错误”产看所有严重报警。在右侧的数字是此类信息的条数。

附录

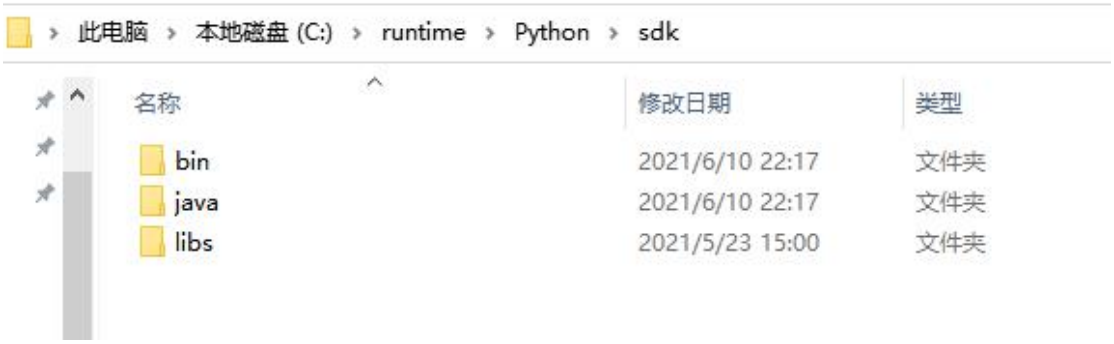
实际控制器升级说明：

为了使用 PC 端离线软件与实际控制器通讯，需要升级控制器软件。

需要:压缩包：sdK_release_20210526_for_pc.zip

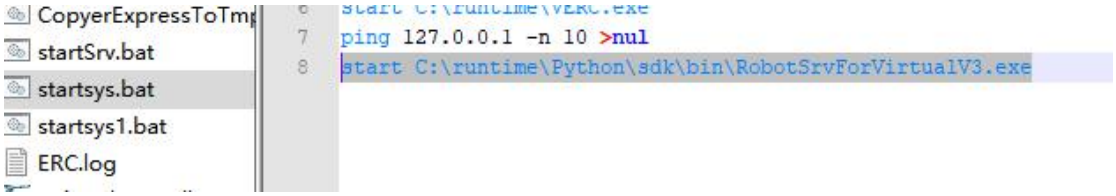
升级步骤如下：

1、将压缩包 sdK_release_20210526_for_pc.zip 解压，将 sdk 文件夹拷贝到 Python 文件夹下。



2、可在 C:\runtime\ sysConfig\ systemCfg 打开查看版本号,在 C:\runtime\python\sdk\PythonSdkDll 下有对应的版本，替换 rsl 库到 runtime 根目录。

3、在启动项中添加 start C:\runtime\Python\sdk\bin\RobotSrvForRealV3.exe



示教器权限分配表

功能		√有权限×无权限★二次密码确认			
		调试	管理员	用户	游客
面板按键	急停按键	√	√	√	√
	模式开关	√	√	√	√
	点动按键	√	√	√	×
	界面切换键	√	√	√	√
	功能按键	√	√	√	√
主界面	设置全局速度	√	√	√	√
	清除报警	√	√	√	√
	切换多轴	√	√	√	√
	界面切按钮	√	√	√	√
工程管理	新建工程	√	√	√	×

功能		√有权限×无权限★二次密码确认			
		调试	管理员	用户	游客
	复制/粘贴工程	√	√	√	×
	删除工程	√	√	√	×
	工程导入导出	√	√	√	×
	加载	√	√	√	√
	注销	√	√	√	√
	打开	√	√	√	√
	设置自启动	√	√	√	×
	重命名	√	√	√	×
	刷新	√	√	√	√
程序编辑	新建指令（包含快捷指令）	√	√	√	×
	修改指令（包含快捷指令）	√	√	√	×
	设置 PC 指针	√	√	√	×
	编辑功能 （复制/粘贴/剪切/删除/注释/折叠）	√	√	√	×
	监视	√	√	√	√
	刷新	√	√	√	√
	指令行选择（单选/多选）	√	√	√	√
	撤销	√	√	√	×
程序数据	新建数据	√	√	√	×
	删除数据	√	√	√	×
	修改数据（包含标定等）	√	√	√	×
	刷新	√	√	√	√
	重命名	√	√	√	×
系统日志	当前报警查看	√	√	√	√
	历史报警查看	√	√	√	√
	获取历史报警	√	√	√	√
	清除报警	√	√	√	√
手动检测	关节坐标系查看	√	√	√	√
	世界坐标系查看	√	√	√	√
	用户坐标系查看	√	√	√	√
	电机力矩值查看	√	√	√	√
	单圈值查看	√	√	√	√
	单圈值设置	√	√	×	×
	机器人回零	√★	√★	√★	×
	工具切换	√	√	√	×
	参考坐标系切换	√	√	√	×
	寸动模式切换	√	√	√	×
	寸动坐标系	√	√	√	×
	寸动设置	√	√	√	×
系统设置	基本设置	√	√	√	×
	碰撞检测	√	√	√	×

功能		√有权限×无权限★二次密码确认			
		调试	管理员	用户	游客
	IP 设置	√	√	×	×
	视觉配置	√	√	√	×
	跟随配置	√	√	√	×
	振动抑制	√	√	√	×
	本地设置	√	√	×	×
	高级设置	×	×	×	×
	系统状态	×	×	×	×
	硬件测试	×	√	×	×
	维护	√	√	×	×
	调试设置	√	×	×	×
IO 检测	物理 IO 查看	√	√	√	√
	物理输出修改	√	√	√	×
	虚拟 IO 查看	√	√	√	√
	虚拟输出修改	√	√	√	×
	系统 IO 查看及触发设置	√★	√★	×	×
插件导航	新建组	√	×	×	×
	删除组	√	×	×	×
	加载插件	√	×	×	×
	卸载插件	√	×	×	×
	移动	√	×	×	×
	打开	√	×	×	×
插件管理	安装	√	×	×	×
	导出	√	×	×	×
	卸载	√	×	×	×
	权限管理	√	×	×	×

ModbusTCP 控制接口数据表

	本地地址	寄存器地址	定义	说明	注释
Send	MBDataBuffer[0]	40001	预留	心跳检测值	1-65535 循环变化
	MBDataBuffer[1]	40002		全局速度	
	MBDataBuffer[2]	40003		读写标志位应答	

	本地地址	寄存器地址	定义	说明	注释
	MBDataBuffer[3]	40004	Rob 状态信息	bit0：手动操作模式 bit1：自动操作模式 bit2：远程操作模式 bit3：使能状态 bit4：运行状态 bit5：错误状态 bit6：程序运行状态 bit7：机器人正在动作	
	MBDataBuffer[4]	40005	当前加载工程名	20 个字节	例如加载的工程文件名为 :estun.test,则各寄存器的值如下: [4]0x6573, [5]0x7475, [6]0x6E2E, [7]0x6D61, [8]0x696E,
	MBDataBuffer[5]	40006			
	MBDataBuffer[6]	40007			
	MBDataBuffer[7]	40008			
	MBDataBuffer[8]	40009			
	MBDataBuffer[9]	40010			
	MBDataBuffer[10]	40011			
	MBDataBuffer[11]	40012			
	MBDataBuffer[12]	40013			
	MBDataBuffer[13]	40014			
	MBDataBuffer[14]	40015	SimDout[1-16]	DO 1-16	
	MBDataBuffer[15]	40016	SimDout[17-32]	DO 17-32	
	MBDataBuffer[16]	40017	SimDout[33-48]	DO 33-48	
	MBDataBuffer[17]	40018	SimDout[49-64]	DO 48-64	
	MBDataBuffer[18]	40019	Rob 执行命令状态	bit0: 命令为 0 bit1：急停命令执行成功 bit2：启动命令执行成功 bit3：停止命令执行成功 bit4：复位命令执行成功 bit5：上使能命令成功 bit6：下使能命令成功 bit7：加载工程命令成功 bit8：注销工程命令成功 bit9：设置全局速度成功 bit10：等待控制权 bit11：等待命令	命令寄存器为 0 时，bit[0]为 1，命令寄存器由命令时，bit[0]为 0 命令执行成功后，相应成功位置 1 当重新获取控制权且命令为 0 时，之前的成功位会被清零 因此，可下发命令的状态码为 0x801

	本地地址	寄存器地址	定义	说明	注释
				bit12：等待命令执行完成 bit13：命令执行错误 bit14：保留 bit15：保留	
	MBDataBuffer[19]	40020	用户用	AO 1-32	
	……				
	……				
	……				
	MBDataBuffer[50]	40051			
Receive	MBDataBuffer[51]	40052	Rob 操作指令	bit2（0→0x4）：机器人程序启动 bit3（0→0x8）：机器人程序停止 bit4（0→0x10）：机器人错误复位 bit7（0→0x80）：加载工程文件 bit8（0→0x100）：注销当前工程文件 bit9（0→0x200）：设置全局速度 bit10（0→0x400）：指令状态机重置	所有指令上升沿触发，配合读写标志位 0x11（命令状态位位 0x801 时可下发指令） 注意：当遇到命令响应失败时，必须利用 bit10 复位状态机，方可以下发新命令
	MBDataBuffer[52]	40053	全局速度值		
	MBDataBuffer[53]	40054	设置工程名	20 个字节	
	MBDataBuffer[54]	40055			
	MBDataBuffer[55]	40056			
	MBDataBuffer[56]	40057			
	MBDataBuffer[57]	40058			
	MBDataBuffer[58]	40059			
	MBDataBuffer[59]	40060			

	本地地址	寄存器地址	定义	说明	注释
	MBDataBuffer[60]	40061			
	MBDataBuffer[61]	40062			
	MBDataBuffer[62]	40063			
	MBDataBuffer[63]	40064	SimDI[1-16]	DI 1-16	
	MBDataBuffer[64]	40065	SimDI[17-32]	DI 17-32	
	MBDataBuffer[65]	40066	SimDI[33-48]	DI 33-48	
	MBDataBuffer[66]	40067	SimDI[49-64]	DI 48-64	
	MBDataBuffer[67]	40068	用户用	AI 1-32	
	……				
	……				
	……				
	MBDataBuffer[98]	40099			
	MBDataBuffer[99]	40100	读写标志位		0x11 打开 rob 命令下发权限

部署

软件需要 64 位系统。

安装在无中文目录下。

如果三维插件无法使用，很可能是显卡驱动需要更新。

运动指令

运动指令列表如下图所示：

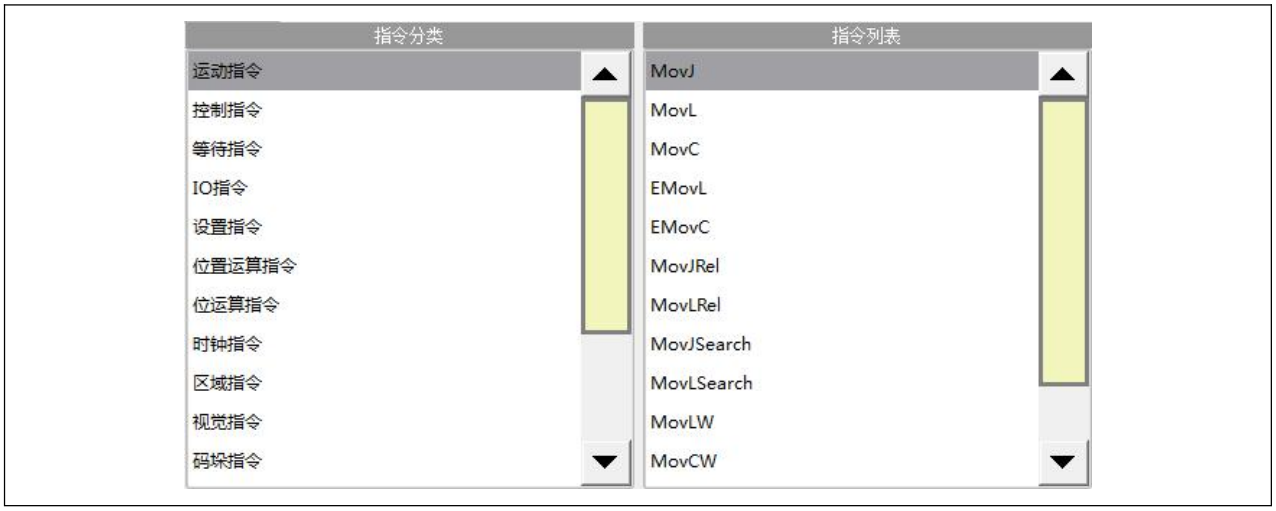


图 1 运动指令列表

MovJ

该指令表示机器人各个关节进行点到点的运动（point to point），机器人的末端轨迹为不规则的曲线，并且在该指令运行结束时可进行 IO 指令的操作，具体如下图：

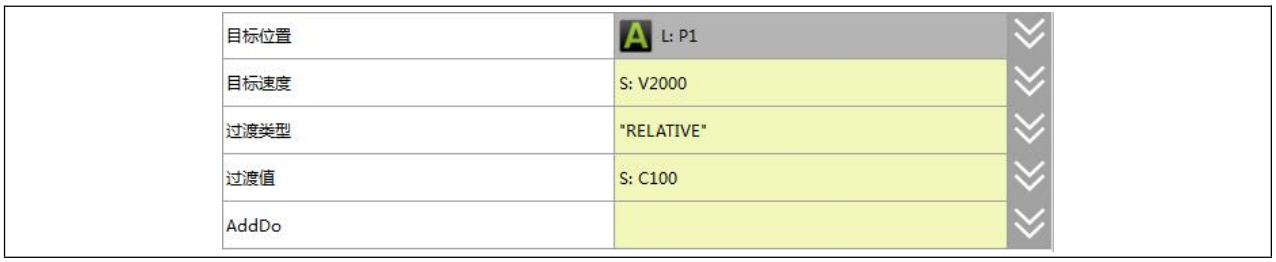


图 2 MovJ 参数编辑界面

- 目标位置：指令终点位置，类型可为APOS或CPOS。在下拉列表中选择已有的点或在弹出的对话框中新建并示教一个变量。
- 目标速度：指令运行速度，为SPEED类型变量。该变量可以选用系统预定义，也可以自己创建；其中，per项对MovJ指令起调控作用，即为全局速度的百分比。
- 过渡类型：机器人逼近终点时的过渡方式。
  - RELATIVE：相对过渡。
  - ABSOLUTE：绝对过渡。
- 过渡值：机器人逼近终点时的过渡值，为ZONE类型变量。该变量可以选用系统预定义，也可以自己创建；其中，当变量值为0时表示不过渡，过渡值越大过渡半径越大。
- AddDo：机器人执行完该指令后，可进行IO操作。
  - NULL：无任何操作。
  - IO 指令：执行 IO 操作，目前支持的 IO 指令有 SetDo、SetAo、SetSimDo、SetSimAo、PulseOut、PulseSimOut、SetDO8421、SetDIEdge、SetSimDIEdge。

说明	SPEED 类型变量有如下参数： per：百分比，主要影响关节坐标系下运动，如 MovJ 等 tcp：末端直线速度，主要影响笛卡尔坐标系下运动，如 MovL、MovC 等 ori：末端姿态速度，主要影响笛卡尔坐标系下运动，如 MovL、MovC 等 exj_l：外部轴直线速度(当无外部轴时无效) exj_r：外部轴旋转速度（当无外部轴时无效） ZONE 类型变量变量有如下几个参数： per：百分比，相对过渡类型时转弯区 dis：绝对距离，绝对过渡类型是转弯区

示例 1：

```
MovJ (P1, V50, "RELATIVE", C100)
```

机器人以 V50 的速度运行到 P1 点，并与下一轨迹存在过渡，过渡值为 100%。

示例 2：

```
MovJ (P2, V50, "RELATIVE", C0)Do SetDo(DO0,0)
```

机器人以 V50 的速度运行到 P2 点，且无过渡，运行至 P2 点时，设置 DO0 为 0。

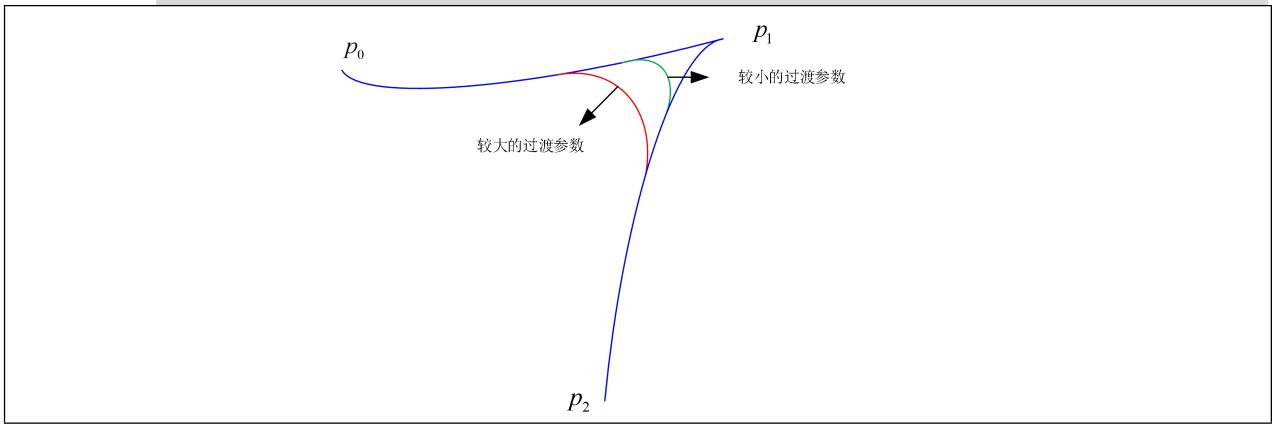


图 3 轨迹示意图

MovL

MovL指令为直线运动指令，通过该指令可以使机器人TCP点以设定的速度直线运动到目标位置，若运动的起止姿态不同，则运行过程中姿态随位置同步地旋转到终点的姿态。与MovJ相同，在执行完该指令时可以附加进行IO操作，MovL的参数设置与MovJ雷同，不同之处为目标速度为绝对速度值，不是百分比。

MovC

圆弧指令指机器人TCP点从起始位置，经过中间位置到目标位置做圆弧运动，若运动的起止姿态不同，则运行过程中姿态随位置同步地旋转到终点的姿态，但不一定经过中间位置的姿态。

中间位置	L: P1
目标位置	L: P2
目标速度	S: V2000
过渡类型	"RELATIVE"
过渡值	S: C100
AddDo	

图 4 MovC 参数编辑界面

- 中间位置：圆弧中间辅助点位置。
- 目标位置：圆弧终点位置。
- 目标速度：指令运行速度。
- 过渡类型：机器人逼近终点时的过渡方式。
  - RELATIVE：相对过渡。
  - ABSOLUTE：绝对过渡。
- 过渡值：机器人逼近终点时的过渡值。
- AddDo：机器人执行完该指令后，可进行IO操作。
  - NULL：无任何操作。
  - IO 指令：执行 IO 操作。

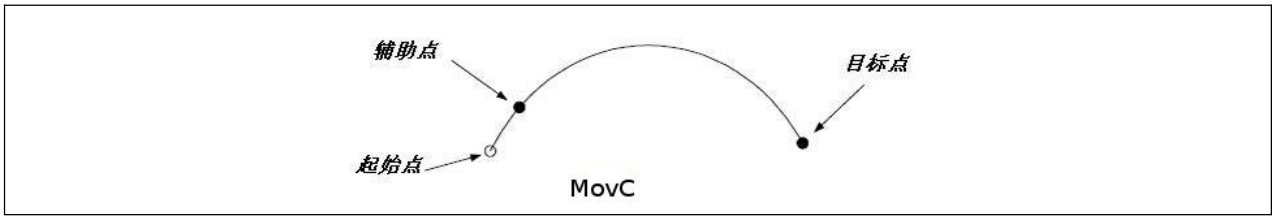


图 5 MovC 圆弧关键点示意

该指令必须遵循以下规定：机器人TCP末端做整圆运动，必须执行两个圆弧运动指令。

EMovL

带坐标系参数的直线运动指令，通过该指令可以使机器人TCP点自动切换到设定的坐标系并以设定的速度直线运动到目标位置。

目标位置	L: P1
目标速度	S: V2000
过渡类型	"RELATIVE"
过渡值	S: C100
工具参数	S: nullTool
坐标系参数	S: World
AddDo	

图 6 EMovL 参数编辑界面

若直线运动的起止姿态不同，则运行过程中姿态按设定的姿态速度随位置同步地旋转到终点的姿态。在执行完该指令时可以附加进行IO操作，

- 目标位置：指令终点位置。在下拉列表中选择已有的点或在弹出的对话框中新建并示教一个变量。
- 目标速度：指令运行速度。该参数值是相对于全局速度的一个百分比。
- 过渡类型：机器人逼近终点时的过渡方式。
  - RELATIVE：相对过渡。
  - ABSOLUTE：绝对过渡。
- 过渡值：机器人逼近终点时的过渡值。
- 工具参数：示教终点位置时参考的工具坐标系。
- 坐标系参数：示教终点位置时参考的用户坐标系。
- AddDo：机器人执行完该指令后，可进行IO操作。
  - NULL：无任何操作。
  - IO 指令：执行 IO 操作。

EMovC

圆弧运动指令，通过该指令可以使机器人TCP点以设定的速度做圆弧运动到目标位置。

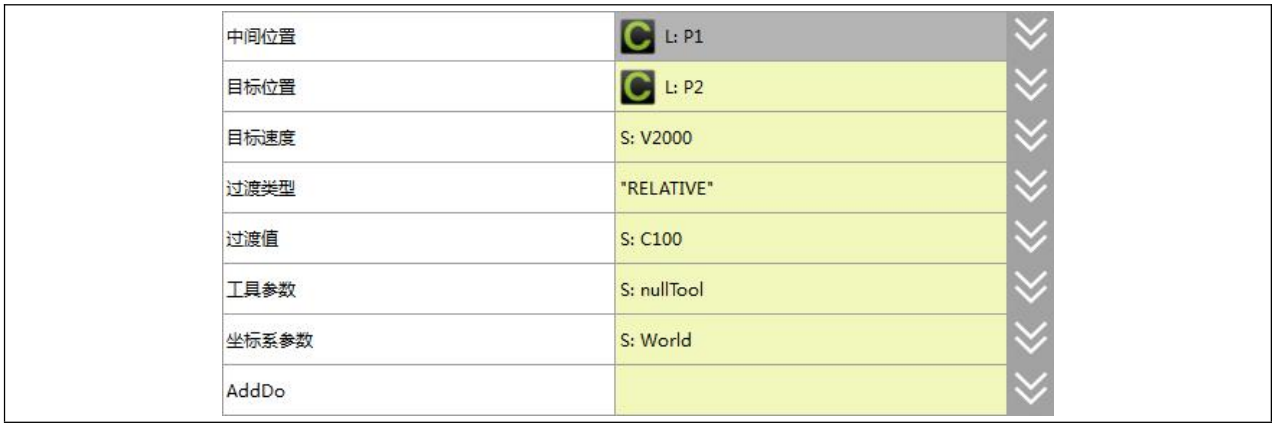


图 7 EMovC 数编辑界面

若运动的起止姿态不同，则运行过程中姿态随位置同步地旋转到终点的姿态，但不一定经过中间位置的姿态。

- 中间位置：圆弧中间辅助点位置。
- 目标位置：指令终点位置。在下拉列表中选择已有的点或在弹出的对话框中新建并示教一个变量。
- 目标速度：指令运行速度。该参数值是相对于全局速度的一个百分比。
- 过渡类型：机器人逼近终点时的过渡方式。
  - RELATIVE：相对过渡。
  - ABSOLUTE：绝对过渡。
- 过渡值：机器人逼近终点时的过渡值。
- 工具参数：示教终点位置时参考的工具坐标系。
- 坐标系参数：示教终点位置时参考的用户坐标系。
- AddDo：机器人执行完该指令后，可进行IO操作。
  - NULL：无任何操作。
  - IO 指令：执行 IO 操作。

MovJRel

MovJ插补相对偏移指令。该指令总是以当前机器人位置或者上一步运动指令的目标位置为起点位置，然后机器人相对移动位移偏移。

与MovJ等指令不同，其使用的目标相对位置值无法通过示教获取，用户需要事先在变量列表中新建关节相对位置类型的变量，然后在下拉列表中选择。

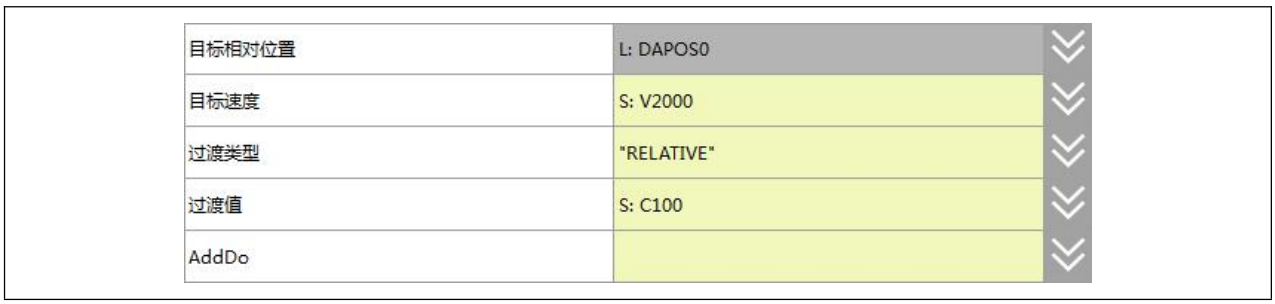


图 8 MovJRel 数编辑界面

- 目标相对位置：执行该指令时机器人要移动的位置增量。
- 目标速度：执行该指令时机器人的移动速度。该参数值是相对于全局速度的一个百分比。
- 过渡类型：机器人逼近终点时的过渡方式。
  - RELATIVE：相对式过渡。
  - ABSOLUTE：绝对过渡。
- 过渡值：机器人逼近终点时的过渡值。
- AddDo：机器人执行完该指令后，进行IO操作。

MovLRel

与MovJRel类似。

MovJSearch

寻位指令，指的是在执行这条MovJ指令时，进行IO检测或力矩检测。

当检索条件为IO检测时，索引IO的值达到触发值时，停止运行该条指令，继续执行下条指令。

当检索条件为力矩检测时，索引轴的力矩值达到触发值时，停止运行该条指令，继续执行下条指令。

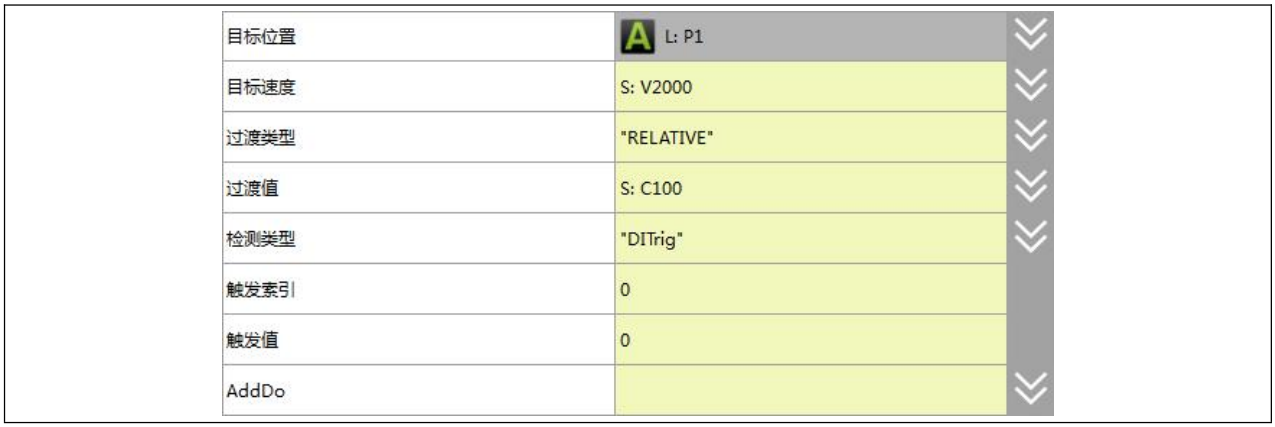


图 9 MovJSearch 数编辑界面

- 目标位置、目标速度、过渡类型、过渡值、AddDo：与MovJ指令含义相同。
- 检测类型
  - DITrig：物理数字量 IO 输入检测。
  - SIMDITrig: 虚拟数字量 IO 输入检测。
  - TorqTrig：力矩检测。
- 触发索引
  - 对于 InputTrig，该参数表示需要检测的 IO 端口号。
  - 对于 TorqTirg，该参数表示需要检测的轴号。
- 触发值：
  - 对于 InputTrig， IO 检测的阈值。
  - 对于 TorqTirg，力矩检测的阈值，单位为额定转矩的千分比。
- AddDo：机器人执行完该指令后，进行IO操作。

MovLSearch

寻位指令，指的是在执行这条MovL指令时，进行IO检测或力矩检测，与MovJSearch类似。

MovLW

摆动指令，指的是根据设定的摆频、摆幅，在前进方向上做正弦摆运动。

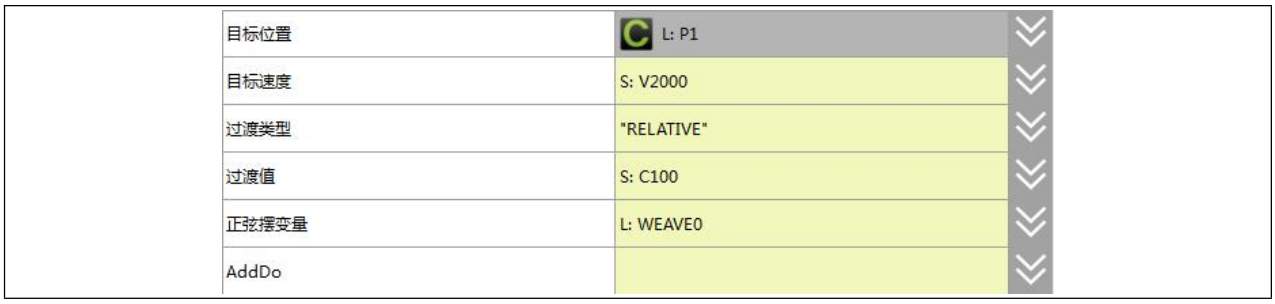
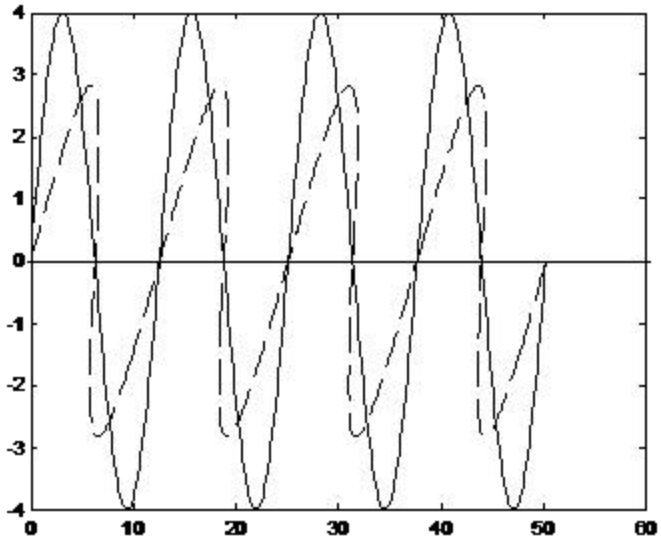


图 10 MovLW 编辑界面

- 目标位置：指令终点位置（APOS或CPOS类型位置点）。在下拉列表中选择已有的点或在弹出的对话框中新建并示教一个变量。
- 目标速度：指令运行速度（详细见MovJ注释）。
- 过渡类型：机器人逼近终点时的过渡方式。
  - RELATIVE：相对过渡。
  - ABSOLUTE：绝对过渡。
- 过渡值：机器人逼近终点时的过渡值（详细见MovJ注释）。
- 正弦摆变量：摆动参数。可以设置摆频、摆幅、停止时间、旋转等参数。
- AddDo：机器人执行完该指令后，可进行IO操作。
  - NULL：无任何操作。
  - IO 指令：执行 IO 操作。

说明	SPEED类型变量有如下参数： Freq：摆频，单位：Hz Amp：摆幅，单位：mm StopTime：停止时间，指在摆动到波峰、波谷停止时间，单位：ms RotAngle_X：摆动基准平面绕摆动行进方向旋转的角度，该参数用于决定最终的摆动平面，单位：deg RotAngle_Z：绕摆动平面的法向量旋转的角度，决定正弦摆的形状，单位：deg

注意	由行进路线与工具坐标系的Tz方向确定的平面的法向量，与行进路线构成的平面为摆动基准平面。 RotAngle_Z的作用，如下图所示：RotAngle_Z=0为实线，不为0为虚线所示。 

MovCW

与MovLW类似。

OnDistance

距离触发指令，在不打断过渡的前提下，控制器可以从起点运动一段距离或者距离终点一段距离时触发执行一些操作，如赋值，设置IO等操作。

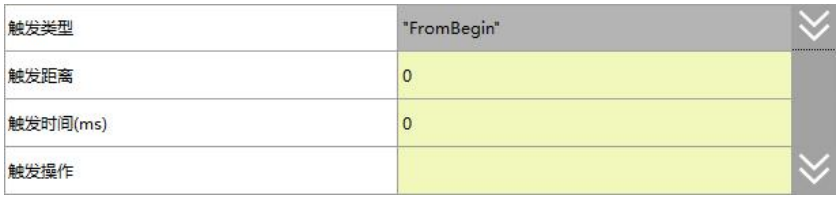


图 11 OnDistance 编辑界面

当触发类型设定的是从起点判断时，机器人运动到设定的距离后，控制器会进入触发等待，延时触

发时间后，开始执行相关的触发操作。

当触发类型设定的是从末点判断时，机器人运动到设定的距离后，控制器会进入触发等待，延时触发时间后，开始执行相关的触发操作。

- 触发类型：指定当前触发是距离起点一段距离触发还是距离终点一段距离触发。
- 触发距离：触发的距离参数。
- 触发时间：满足触发条件后触发延时触发的时间。
- 触发的操作：满足触发条件后，机器人需要执行的触发操作。

示例 1：

在P1到P2轨迹的起点距离为200ms处，延时100ms执行Do操作。

```
MovL (P1, V50, "RELATIVE", C0)
OnDistance("FromBegin",200,100) DO SetDO(DO0,1);
MovL (P2, V50, "RELATIVE", C0)
```

示例 2：

在P1到P2轨迹的终点距离为200ms处，延时100ms执行Do操作。

```
MovL (P1, V50, "RELATIVE", C0)
OnDistance("FromEnd",200,100) DO SetDO(DO0,1);
MovL (P2, V50, "RELATIVE", C0)
```

⚠ 注意	• 当距离触发满足且时间触发还在延时时，如果当前轨迹已经运行完，触发将延后执行，等待时间触发满足后再执行。 • 当设定的触发距离超出轨迹长度时，会有两种情况：触发类型为起点触发时，运动到末点时将执行触发判断。触发类型为末点触发时，在起点处将执行触发判断。满足条件则立刻触发。 • 两段轨迹之间有多条触发指令时，触发各自检测，满足条件后各自触发执行，互不干扰。 • 执行OnDistance之前必须执行笛卡尔坐标系下运动指令，否则OnDistance将不起作用，同时状态栏提示该操作无效，但不影响系统的运行。 • OnDistance与下一条笛卡尔坐标系下运动指令之间可插入除等待以外指令，否则执行报错。等待指令如wait，waitcondition，waitfinish等。

OnParameter

距离百分比触发指令，用于在不打断过渡的前提下，在直线轨迹的某一处触发执行一些操作，比如赋值，设置io等快捷操作。

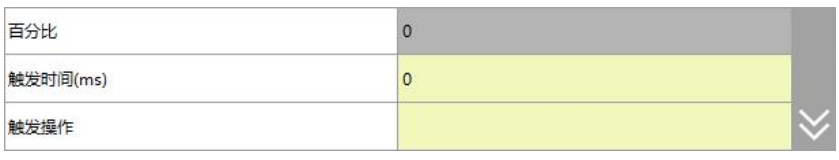


图 12 OnParameter 编辑界面

当机器人运动到设定的触发百分比路径时，控制器进入触发等待，延时触发时间后，开始执行相关的触发操作。

- 路径百分比：需要触发的路径所在占比。
- 触发时间：满足触发条件后触发延时触发的时间。
- 触发的操作：满足触发条件后，机器人需要执行的触发操作。

示例：

在P1到P2轨迹运行距离为60%处，延时500ms执行Do操作。

```
MovJ (P1, V50, "RELATIVE", C0)
OnParameter(60,500) DO SetDO(DO0,1);
MovL (P2, V50, "RELATIVE", C0)
```

控制指令

示教编程器控制指令列表如图所示：

图 13 控制指令列表界面

LABEL

Label指令用于定义GOTO跳转目标。

GOTO

GOTO指令用于跳转到程序不同部分。跳转目标通过LABEL指令定义。不允许从外部跳转进入内部程序块。内部程序块可能是WHILE循环程序块或者IF程序块。

IF

IF指令用于条件判断表达式跳转控制。当条件判断表达式结果为真时，程序执行IF下程序块的内容。类似于c++中的IF语句。IF条件判断表达式结果必须是BOOL类型。每一个IF指令必须以关键字ENDIF做为条件判断程序块的结束。

当选择“控制指令→ IF”后，将进入IF语句的设置界面，如下图所示。

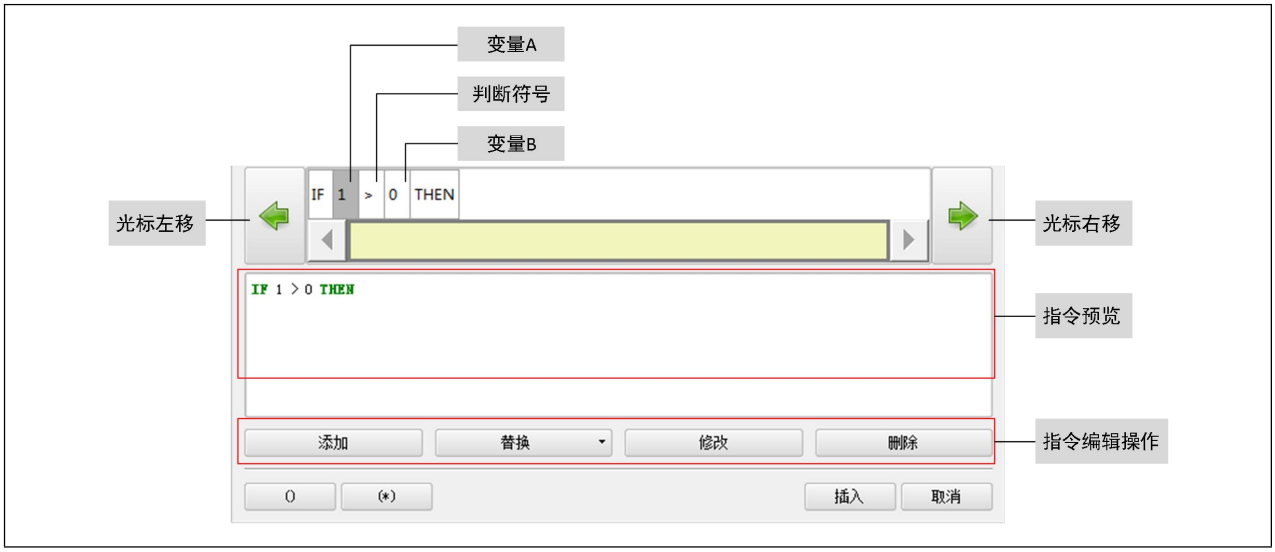


图 14 IF 语句设置界面

- 变量A和变量B可设定为常数或参数：点击“替换”然后选择想要替换的常数、参数或函数（包括端口号）。若要修改变量的值，可将光标移至需要修改的变量上，再点击“修改”，并在弹出的对话框中设置想要修改的数值。
- 判断符号用来判断变量A和变量B之间的关系，若为真，则执行THEN后的指令，若为假，则跳过THEN后的指令。若要修改判断符号，可将光标移至判断符号上，再点击“修改”，并在弹出的对话框中选择需要的符号。
- 点击“添加”，可在光标的当前位置新增一个“+<Exp>”，其中，“+”表示运算符号，“<Exp>”表示操作数。
  - 点击“+”，然后点击“修改”并在弹出的软键盘中选择想要的运算符号，如下图**错误!未找到引用源。**所示。

图 15 可修改的运算符号

  - 点击“<Exp>”，然后点击“替换”或“修改”，可变更操作数为想要的常数、参数或函数（包括端口号）。
- 点击想要删除的运算符号或操作数，然后点击“删除”，可删除该运算符号及随后的操作数。
- 点击某个运算符号，再点击“()”，可为当前运算符号的前后操作数增加一个括号。
点击某个操作数，再点击“()”，可为当前的操作数增加一个括号。
- 若要删除括号，点击运算符号或操作数，再点击“(*)”，可删除被选中的运算符号或操作数响应的括号。

```
IF   x >1 and y == 1and z < 1 THEN
     y = 10
ENDIF
```

说明	若要删除IF指令，需在程序编辑界面中点击“多行选择”，然后点击IF和END_IF之间的控制语句，再点击右侧菜单“编辑→ 删除”。 在“替换”时，可选择“数学函数”或“字符函数”。关于函数的说明，请参见“数学运算函数”和“字符运算函数”。

Else

Else指令常与IF语言联合使用，表示IF条件判断的不成立时将执行的语句。

WHILE

WHILE 指令在满足条件的时候循环执行子语句。循环控制表达式必须是BOOL 类型。该指令必须以关键字ENDWHILE作为循环控制结束。

设置WHILE表达式的方法与设置IF表达式的方法相同。

例如：

```
l= 1
WHILE 1==1DO
MovJ (P1, V50, ”RELATIVE”, C0)
MovJ (P2, V50, ”RELATIVE”, C0)
l=l+1
ENDWHILE
```

该指令执行两点之间的循环运动100次。

CALL

调用指令，当前程序跳转至同一工程下另一个子程序，子程序执行完后跳转回当前程序。假如需要调用的程序为program1，在程序中生成命令为：

```
CALL program1
```

程序执行流程如下图所示。

图 16 CALL 指令执行流程

RUN

程序并行运行指令，使得机器人能够在运行当前程序的同时，运行其它程序，（当前程序继续运行不跳转的同时启动另一个程序）且运行的程序必须是同一个工程下。假如需要运行的程序为program1，在程序中生成命令为：

```
RUN program1
```

说明	目前，控制器可支持运行2个并行程序。

程序执行流程如下图所示。

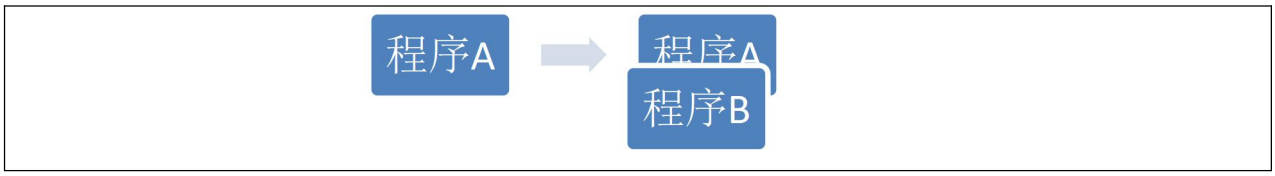


图 17 RUN 指令执行流程

KILL

停止并行运行的指令，使得机器人能够在运行当前程序的同时，停止运行其它程序，且停止运行的程序必须是同一个工程下且为运行状态。假如需要停止运行的程序为program1，在程序中生成命令为：

```
KILL program1
```

RETURN

返回指令。一般情况下，执行该指令后，程序会返回至程序的开头，如果是在CALL指令中使用了RETURN指令，则返回至CALL指令的上一级程序。

…=…

建立一个表达式，用来给某个变量赋值，如**错误!未找到引用源。**所示。

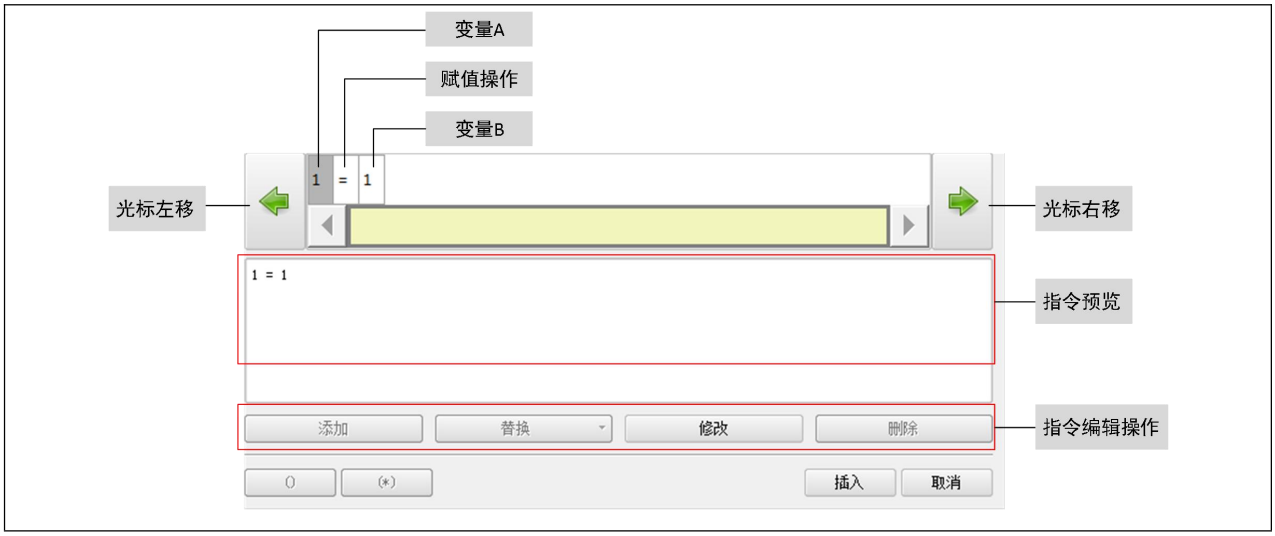


图 18 建立表达式说明

- 变量A和变量B可设定为常数或参数：点击“替换”然后选择想要替换的常数、参数或函数（包括端口号）。若要修改变量的值，可将光标移至需要修改的变量上，再点击“修改”，并在弹出的对话框中设置想要修改的数值。
- 若要添加一个赋值操作，将光标移动至“变量B”，然后点击“添加”，表达式会自动添加一个新的空数值，默认的运算符号为“+”。
- 点击运算符号“+”，再点击“修改”，可选择想要的运算符号，如下图所示。

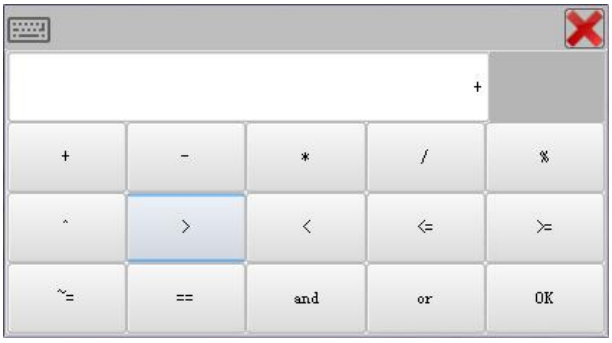


图 19 可修改的运算符号

- 选择某一新增运算符号或随后的变量，然后点击“删除”，可删除该新增的运算符号和随后的变量。

说明	在“替换”时，可选择“数学函数”或“字符函数”，关于函数的说明，请参见请参见“数学运算函数”和“字符运算函数”。

/*…*/

注释指令，为程序添加释义或备注。
该指令不会影响程序的运行，仅仅是便于用户阅读和理解程序。

等待指令

等待指令列表如图所示：

图 20 等待指令列表

Wait

用于设置机器人等待时间，时间单位为ms，假如设置等待1s，生成命令为：

```
Wait(1000)
```

WaitFinish

用于同步机器人的运动以及程序执行。

如示例程序所示，SetDO指令将在第一条MovL指令执行到20%时触发。第一条MovL指令执行完后直接过渡执行第二条MovL指令。

若删去WaitFinish (20)，则SerDO将在第一条MovL指令执行完后执行，然后再执行第二条MovL指令。此情况下两条MovL指令间无过渡，且有停顿。

```
MovL (P1, 1000,”RELATIVE”, C100)
WaitFinish (20)
SetDO (DO1, 1)
MovL (P2, 1000,”RELATIVE”, C100)
```

WaitCondition

设置机器人执行等待的条件。若在设定时间内未满足条件，则返回超时状态。

- 当“判别条件”为真时，才执行下一步指令，否则程序将继续等待直到表达式为真。
- “时长”表示执行等待时所需的时间，单位为ms。
  - 如果该参数的值为 0，将强制等待判别条件为真时才继续执行下一条指令。
  - 如果该参数的值非 0，即使判别条件仍未为真，系统将在等待给定时长后跳过该指令，并继续执行下一条指令。
- 中断使能：设定等待过程中是否在程序恢复后继续计时。
  - 0：遇到程序停止时，停止计时，并在恢复时继续程序停止前的计时。
  - 1：遇到程序停止时，停止计时，并在恢复时重新计时。
- 超时判断值：选择一个变量并在如下两种情况下为其赋值。
  - 若输入条件为真切该指令执行完成时，对该变量赋值 0；
  - 当该指令由于超时执行完成时，对该变量赋值 1。

```
WaitCondition (“ 1 > 0 ”, 1000, 0, INT0)
```

IO 指令

IO指令列表如图所示：

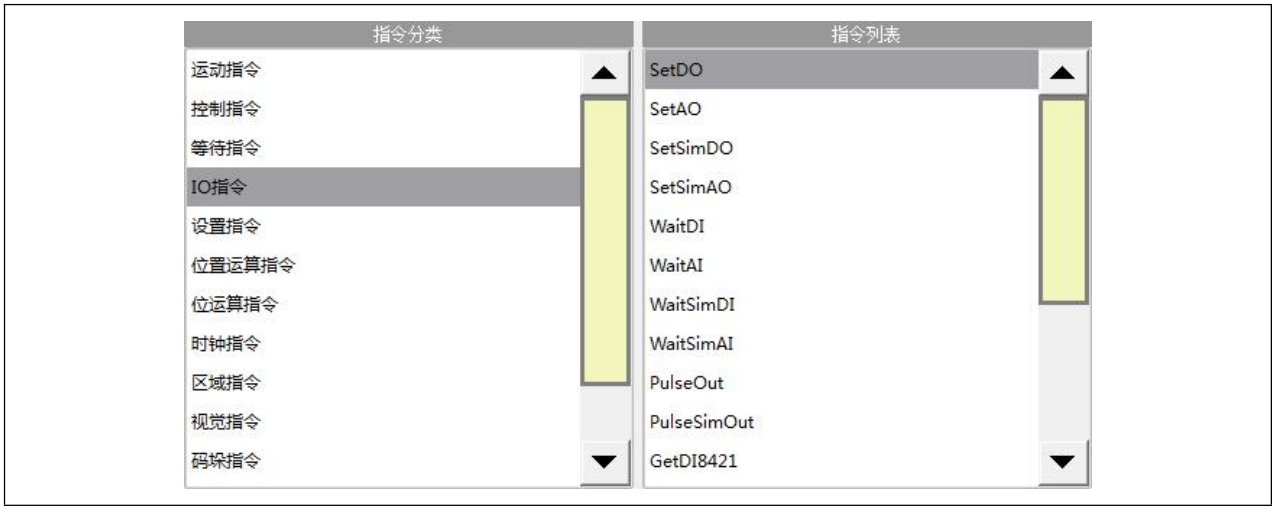


图 21 IO 指令列表

这些指令运用到输入输出模块的数字信号和模拟信号，数字与模拟信号经常与触发信息一起配合使用。

SetDO

将数字量输出端口设置为TRUE（1）或者FALSE（0）状态，例如：

```
SetDO (DO1, 1)
```

SetAO

将模拟量输出端口设置为一定值，例如：

```
SetAO (AO2, 5.000000)
```

SetSimDO

将虚拟数字量输出端口设置为TRUE（1）或者FALSE（0）状态，例如：

```
SetSimDO (SimDO0, 0)
```

SetSimAO

将虚拟模拟量输出端口设置为一定值，例如：

```
SetSimAO (SimAO1, 5.000000)
```

WaitDI

等待直到数字量输入端口被设置或复位。

- 端口变量：需要等待的输入端口编号。

- 端口值：等待的数字量输入端口电平。
  - 0：低电平
  - 1：高电平
- 时长(ms)：等待信号变换的时间。
- 中断使能：设定等待过程中是否在程序恢复后继续计时。
  - 0：遇到程序停止时，停止计时，并在恢复时继续程序停止前的计时。
  - 1：遇到程序停止时，停止计时，并在恢复时重新计时。
- 超时判断值：返回该指令执行结束后的状态
  - 0：成功检测到信号
  - 1：未检测到信号，指令超时返回。

```
WaitDI (DI0, 1, 1000, 0, INT1)
```

WaitAI

等待直到模拟量输入端口值与给定值相等，参数说明与WaitDI相同。

```
WaitAI (AI0, 4.500000)
```

WaitSimDI

等待直到虚拟数字量输入端口被设置或复位，参数说明与WaitDI相同。

```
WaitSimDI (SimDI0, 0)
```

WaitSimAI

等待直到虚拟模拟量输入端口值与给定值相等，参数说明与WaitDI相同。

```
WaitSimAI (SimAI0, 4.500000)
```

PulseOut

设定IO变量输出的脉冲时间与条件。

- 端口变量：需要设定的端口编号。
- 端口值：设定输出的脉冲电平。
  - 0：低电平
  - 1：高电平
- 持续时间：设定脉冲输出的时间。
- 中断使能：设定脉冲输出是否在中断后继续输出。
  - 0：遇到程序停止时，脉冲会持续输出到设定时间后停止输出，即不受程序停止而停止其功能。
  - 1：遇到程序停止时，停止输出，当程序继续运行时才会继续输出，即与程序运行状态同步，程序停则输出暂停，程序运行则输出继续。

```
PulseOut (DO1, 1, 500, 1)
```

PulseSimOut

设定SimIO变量输出的脉冲时间与条件。

- 端口变量：需要设定的端口编号。
- 端口值：设定输出的脉冲电平。
  - 0：低电平
  - 1：高电平
- 持续时间：设定脉冲输出的时间。
- 中断使能：设定脉冲输出是否在中断后继续输出。
  - 0：遇到程序停止时，脉冲会持续输出到设定时间后停止输出，即不受程序停止而停止其功能。
  - 1：遇到程序停止时，停止输出，当程序继续运行时才会继续输出，即与程序运行状态同步，程序停则输出暂停，程序运行则输出继续。

```
PulseSimOut (SimDO1, 1, 500, 1)
```

GetDI8421

获取一段连续的DI口状态（将其看成一段二进制数据）并以十进制数返回。

- 起始DI端口：想要获取的起始DI端口号。
- 结束DI端口：想要获取的结束DI端口号。
- 结束-起始端口值不应大于32。
- 返回值：将获取到的端口状态值存储到传入的int变量中。

```
GetDI8421 (9, 12, IN0)
```

SetDO8421

设置一段连续的DO口状态（将其看成一段二进制数据），将传入的十进制数转换成二进制数设置到指定的DO口上。

- 起始DO端口：想要设置的起始DO端口号。
- 结束DO端口：想要设置的结束DO端口号。
- 结束-起始端口值不应大于32。
- 设定值：想要端口输出的十进制设定值。

以下述指令为例，3转换成二进制是0011,那9号跟10号会分别输出高电平。为2代表9号低电平，10号设为高电平。

```
SetDO8421 (9, 10, 3)
```

GetSimDIToVar

将虚拟数字量输入信号映射到某个变量上。

- 端口变量：获取虚拟数字量输入信号值的变量，类型为虚拟数字量输入类型变量。
- 变量：被获取的数字量信号值赋值的变量，类型为BOOL类型。

例如，将虚拟数字量输入类型变量SIMDI1的值赋值给布尔型变量BOOL0的指令如下：

```
GetSimDIToVar(SimDI1,BOOL0)
```

SetSimDOByVar

将某个布尔型变量的值映射到一个数字量输出变量上。

- 端口变量：数字量输出类型变量，接收变量的值并输出。
- 变量：布尔型变量，将其值输出给端口变量。

例如，将布尔型变量BOOL0的值输出给虚拟数字量输出型变量SimDO1的指令如下：

```
SetSimDOByVar(SimDO1,BOOL0)
```

GetSimAIToVar

将虚拟模拟量输入信号映射到某个变量上。

- 端口变量：获取虚拟模拟量输入信号值的变量，类型为虚拟模拟量输入类型变量。
- 变量：被获取的模拟量信号值赋值的变量，类型为REAL类型。

例如，将虚拟模拟量输入类型变量SIMAI1的值赋值给实数型变量REAL0的指令如下：

```
GetSimAIToVar(SimAI1,REAL0)
```

SetSimAOByVar

将某个REAL型变量的值映射到一个虚拟模拟量输出变量上。

- 端口变量：虚拟模拟量输出类型变量，接收变量的值并输出。
- 变量：REAL型变量，将其值输出给端口变量。

例如，将实数类型变量REAL0的值输出给虚拟模拟量输出型变量SimDO0的指令如下：

```
SetSimAOByVar(SimAO0,REAL0)
```

SetDIEdge

通过指令强制设置某个数字量输入端口的边沿信号状态。

- 端口变量：数字量输出类型变量，设置该端口的上升沿或下降沿状态。
- 边沿类型：选择要设置端口的边沿信号状态。
  - riseEdge：上升沿状态。
  - downEdge：下降沿状态。
- 边沿值：需要设置的边沿状态值。

例如，将DI1端口的上升沿状态设置为0的指令如下：

```
SetDIEdge(DI1 , "riseEdge" , 0)
```

SetSimDIEdge

通过指令强制设置某个虚拟数字量输入端口的边沿信号状态。

- 端口变量：虚拟数字量输出类型变量，设置该端口的上升沿或下降沿状态。

- 边沿类型：选择要设置端口的边沿信号状态。
  - riseEdge：上升沿状态。
  - downEdge：下降沿状态。
- 边沿值：需要设置的边沿状态值。

例如，将SimDI1端口的上升沿状态设置为0的指令如下：

```
SetSimDIEdge(SimDI1 , "riseEdge" , 0)
```

设置指令

设置指令列表如图所示：

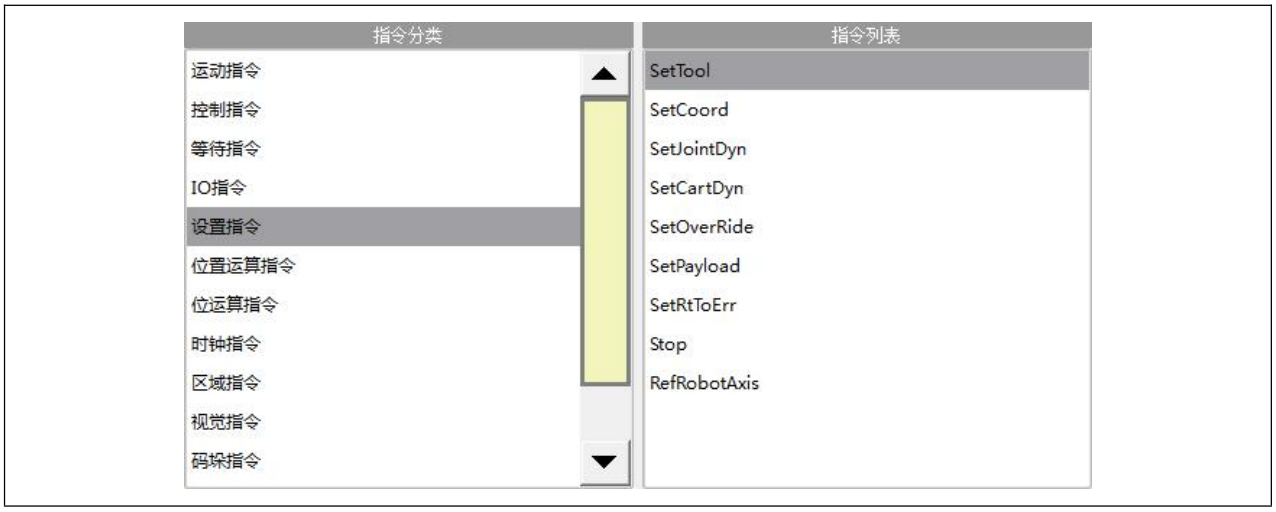


图 22 设置指令列表

SetTool

选择工具坐标系指令。设置工具坐标系的方法请参见“工具坐标系”小节的描述。例如：

```
SetTool (TOOL0)
```

SetCoord

选择用户坐标系指令。设置用户坐标系的方法请参见“用户坐标系”小节的描述。例如：

```
SetCoord (USERCOOR0)
```

SetJointDyn

该指令用于设置关节动态参数，包括加速度、减速度、加加速度。运行该指令后，后续运动指令均以该动态参数运行。

- 加速度：关节最大加速度的百分比。
- 减速度：关节最大减速度的百分比。

- 加加速度：关节最大加加速度的百分比。

SetCartDyn

该指令用于设置笛卡尔空间的动态参数，运行该指令后，后续运动指令均以该动态参数运行。

- 加速度：直线运动最大加速度的百分比。
- 减速度：直线运动最大减速度的百分比。
- 加加速度：直线运动最大加加速度的百分比。
- 姿态加速度：姿态最大旋转加速度的百分比。
- 姿态减速度：姿态最大旋转减速度的百分比。
- 姿态加加速度：姿态最大旋转加加速度的百分比。

SetOverRide

全局速度设置功能。设置“全局速度”为一个合适的百分比数值。例如：

```
SetOverRide (20)
```

SetPayload

设置负载质量。例如：

```
SetPayload (6.0)
```

SetRtToErr

设置机器人停止运行并发出错误信息，例如：

```
SetRtToErr (“halt”, 10000)
```

Stop

该命令用于停止所有激活程序的执行。

```
Stop ()
```

RefRobotAxis

设置机器人单轴回零的指令。

```
RefRobotAxis (1)
```

位置运算指令

IO指令列表如下图所示。

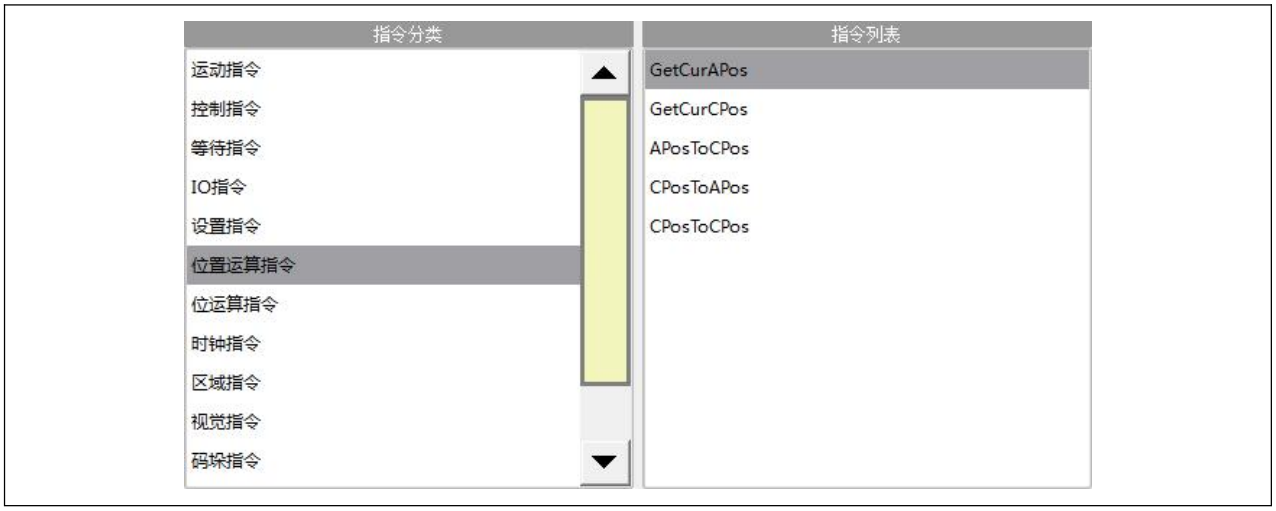


图 23 位置运算指令列表

GetCurAPos

该指令用于获取当前关节坐标系下的位置，赋值给Apos类型变量。

GetCurCPos

该指令用于获取当前坐标系（世界坐标系或用户坐标系）下的笛卡尔空间位置，赋值给Cpos类型变量。

APosToCPos

机器人位置点转换指令，通过该指令可以将APos点转换为CPos点。
给定APos点，以及要转换的目标CPos点的用户坐标系及工具参数，可以得到目标CPos点的值。

CPosToAPos

机器人位置点转换指令，通过该指令可以将CPos点转换为APos点。
给定CPos点及其所在的用户坐标系及工具参数，可以得到目标APos点的值。

CPosToCPos

机器人位置点转换指令，通过该指令可以将CPos点转换为CPos点。
给定CPos点及其所在的用户坐标系及工具参数，以及要转换的目标CPos点的用户坐标系及工具参

数，可以得到目标CPos点的值。

位运算指令

位运算指令列表如下图所示。

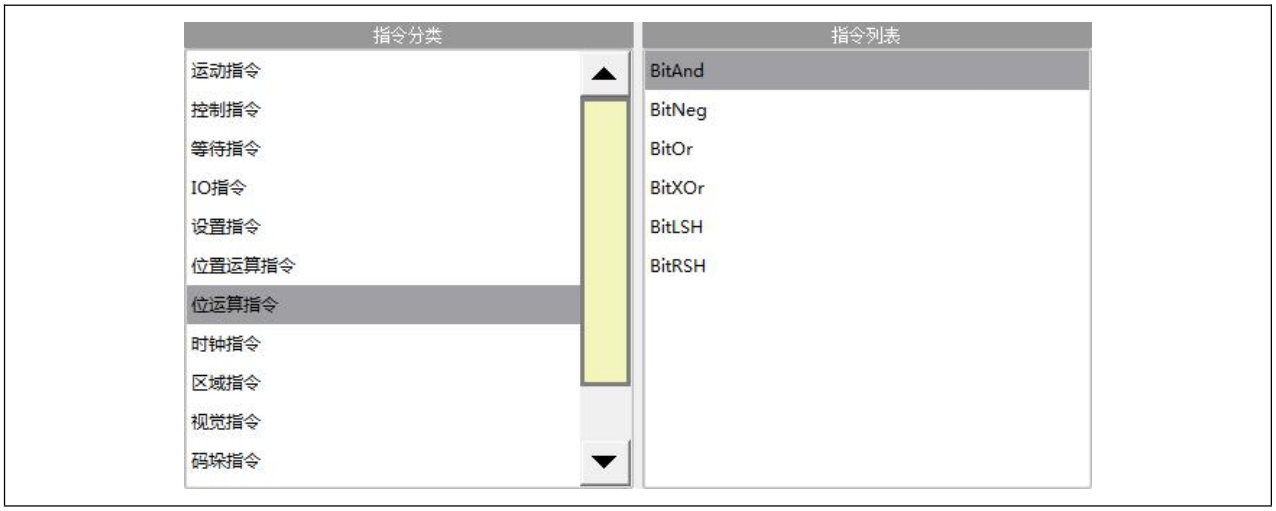


图 24 位运算指令列表

BitAnd

实现按位与的运算。该指令将两个操作数进行按位与的运算，并将结果赋值给第一个操作数。

- 操作数1：INT型变量；运算结果也赋值到该操作数。
- 操作数2：INT型变量。

BitNeg

实现按位取反的运算。该指令将两个操作数进行按位取反的运算，并将结果赋值给第一个操作数。此外，取反操作仅操作数后16位。

- 操作数1：INT型变量；运算结果也赋值到该操作数。

BitOr

实现按位或的运算。该指令将两个操作数进行按位或的运算，并将结果赋值给第一个操作数。

- 操作数1：INT型变量；运算结果也赋值到该操作数。
- 操作数2：INT型变量。

BitXOr

实现按位异或的运算。该指令将两个操作数进行按位异或的运算，并将结果赋值给第一个操作数。

- 操作数1：INT型变量；运算结果也赋值到该操作数。

- 操作数2：INT型变量。

BitLSH

实现按位左移的运算。该指令将两个操作数进行按位左移的运算，并将结果赋值给第一个操作数。

- 操作数1：INT型变量；运算结果也赋值到该操作数。
- 操作数2：INT型变量。

BitRSH

实现按位右移的运算。该指令将两个操作数进行按位右移的运算，并将结果赋值给第一个操作数。

- 操作数1：INT型变量；运算结果也赋值到该操作数。
- 操作数2：INT型变量。

时钟指令

时钟指令列表如下图所示。

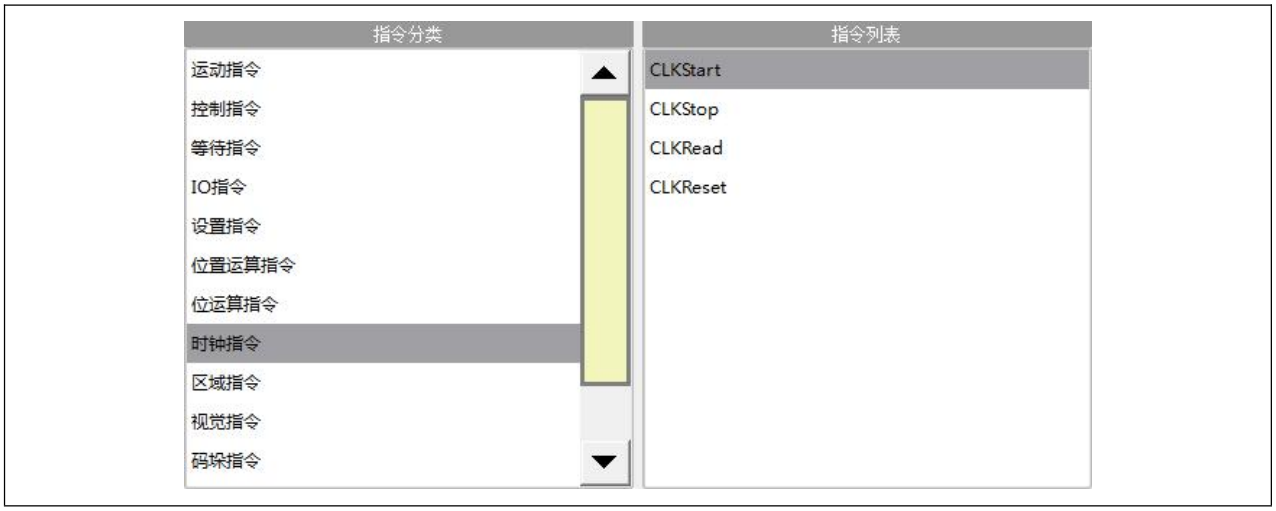


图 25 时钟指令列表

CLKStart

启动指定时钟（启动后，可以从变量列表中看到指定时钟变量的state为true）。例如：

```
CLKStart (CLOCK0)
```

CLKStop

停止指定时钟（其state为false，但不会复位）。例如：

```
CLKStop (CLOCK0)
```

CLKRead

读取指定时钟的值，在变量列表查看变量对应的value即可。例如：

```
CLKRead (CLOCK0)
```

CLKReset

复位指定时钟的状态、值。例如：

```
CLKReset (CLOCK0)
```

区域指令

区域指令列表如下图所示。

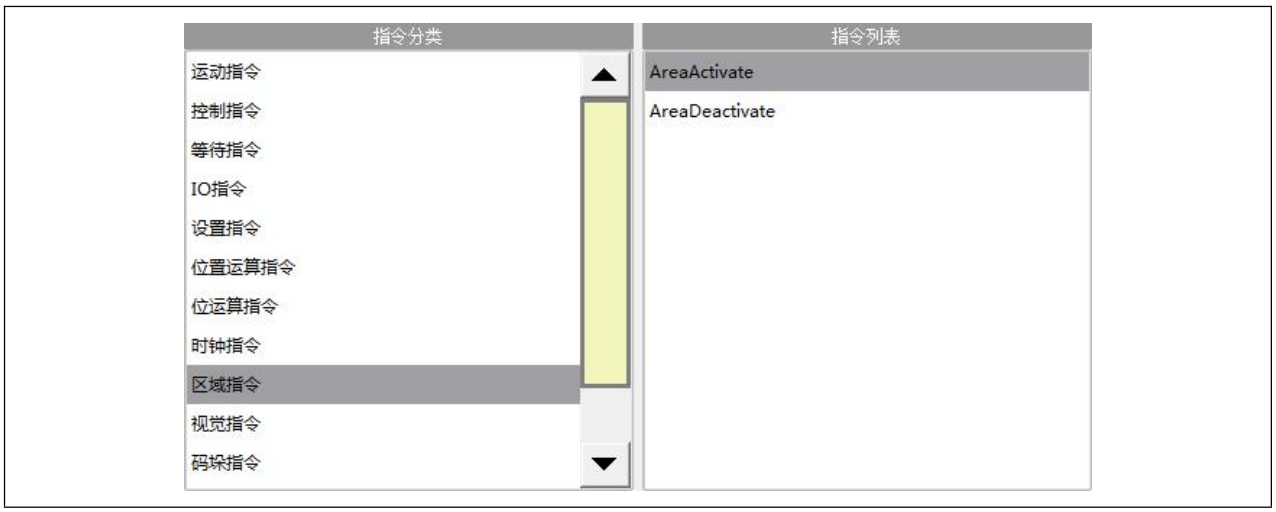


图 26 区域指令列表

AreaActivate

激活区域指令，使指定的区域变量生效。

AreaDeactivate

冻结区域指令，使指定的区域变量失效。

视觉指令

视觉指令列表如下图所示。

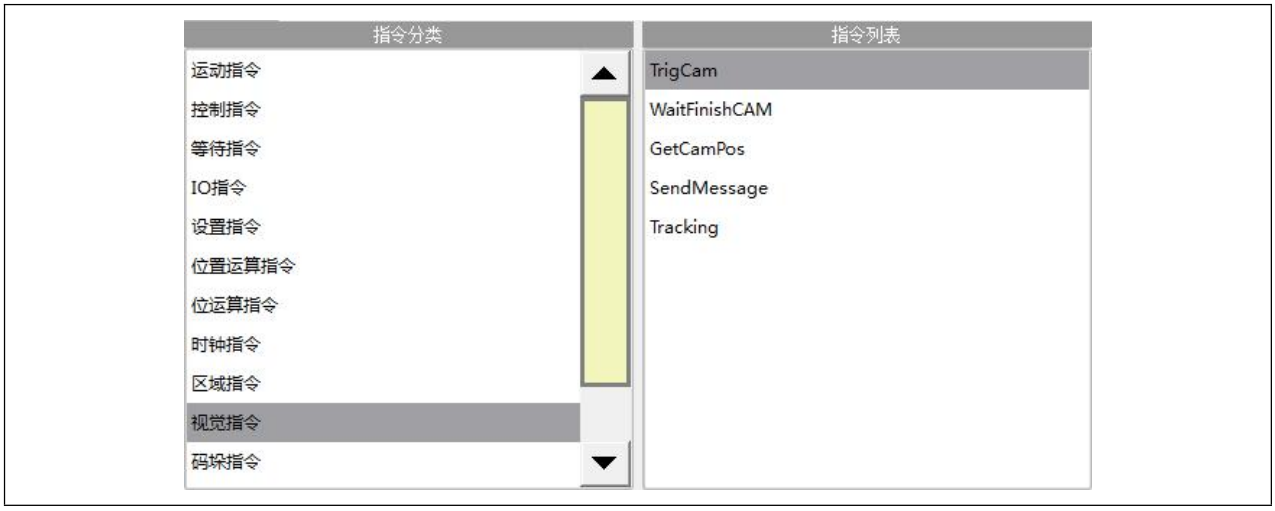


图 27 视觉指令列表

TrigCam

触发相机拍摄指令。

```
TrigCam (“OK”)
```

WaitFinishCAM

等待相机拍摄完成的指令。

```
WaitFinishCAM (1000)
```

GetCamPos

获取相机位置的指令。

```
GetCamPos (P6, INT0, INI1)
```

SendMessage

发送字符串消息，可用于3D视觉指令，给相机发送指定的字符串，让相机拍照。例如：

```
SendMessage (“OK”)
```

Tracking

跟踪传送带指令，关于该指令的详细说明，请参见“传动带跟随”章节。

```
Tracking (10.0, 10.0, DO2, DO1, INT0,INT1)
```

码垛指令

码垛指令列表如下图所示。

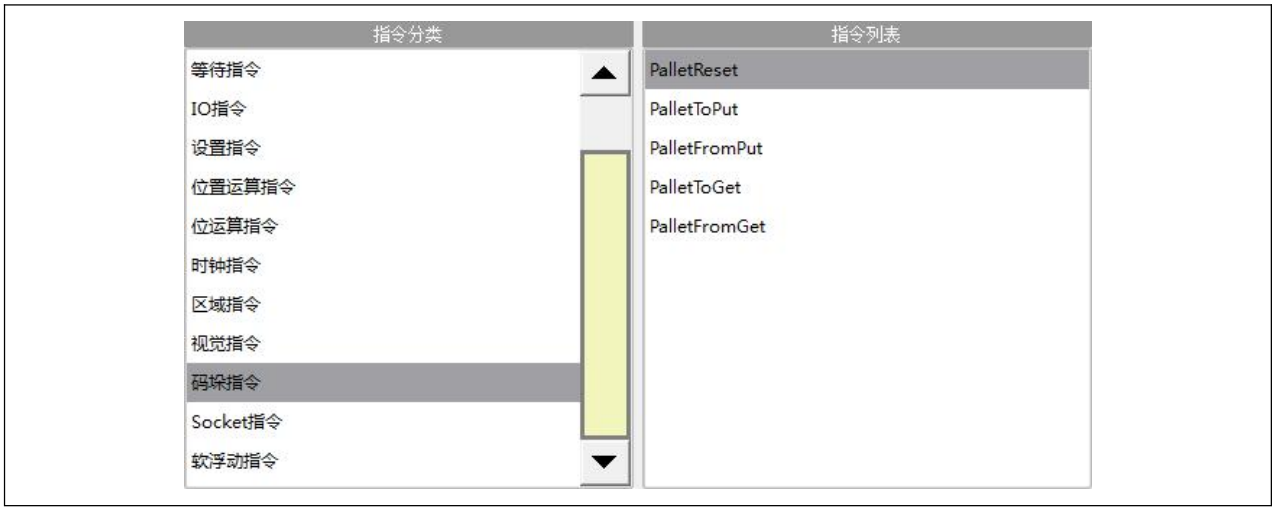


图 28 码垛指令列表

PalletReset

垛重置指令。当一个跺体在未码放完成而被迫停止时，可调用该指令来还原。

- 码垛名：当前码垛的PALLET变量。
- 已码工件数：当前码垛上已经放置的工件数量。一般必须设置为0至该跺体的最大码垛数。

```
PalletReset (PALLET0,0)
```

例如，最大工件数为60的跺体Pallet0，在放置完8个工件时中止。可调用“PalletReset(Pallet0, 8);”重新开始后续工件的码放。

PalletToPut

放置工件指令。

- 码垛名：当前码垛的PALLET变量。
- 目标速度：机器人运行速度。

```
PalletToPut (PALLET1, 2000)
```

码垛功能的垛位计算取决于示教的第一个工件位置和配置的码垛的距离。在前一个工件的位置基础上添加设定的码垛距离，从而得到下一个工件的位置，如下图所示。

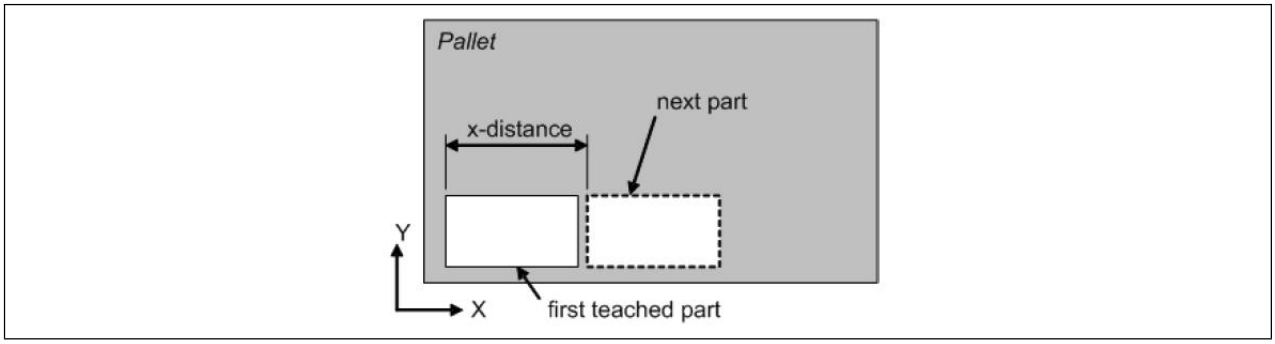


图 29 计算下一个工件的位置

在下图中，默认码垛的顺序从X开始。首先放置x方向里的所有部件，然后再开始放置下一行。在x方向上放置工件直到达到设置好的x方向工件数。

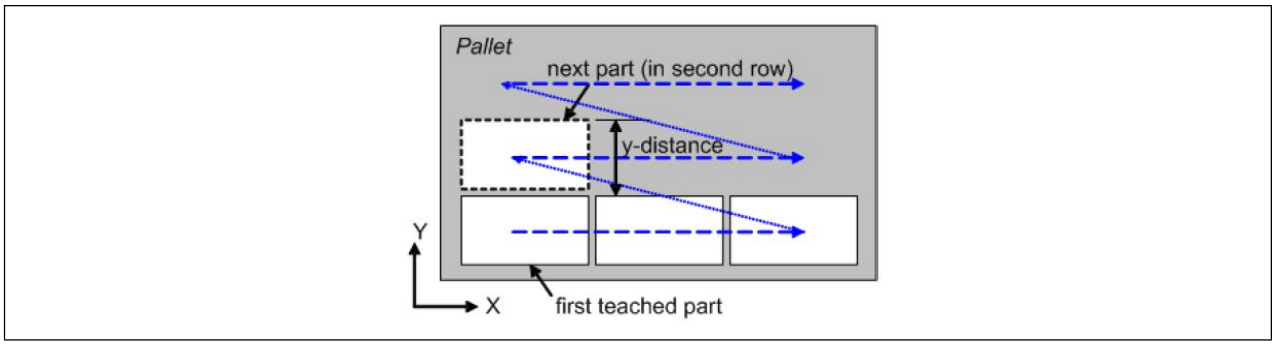


图 30 默认码垛顺序

当一个新行开始放置时，在第一行工件位置的基础上添加y方向上的偏移即生成了新行的工件的位置数据。

这个例子里默认的码垛顺序是x y z. 首先在x方向放置工件然后再进行第二行。新的一行的第一个部件从图中所示的位置开始。只要行满的，它们就会被放在下一行。在这一层的所有行都被填满后，xy层在z方向重复平移，直到跺满。

进行放置时，建议路径如下图所示。

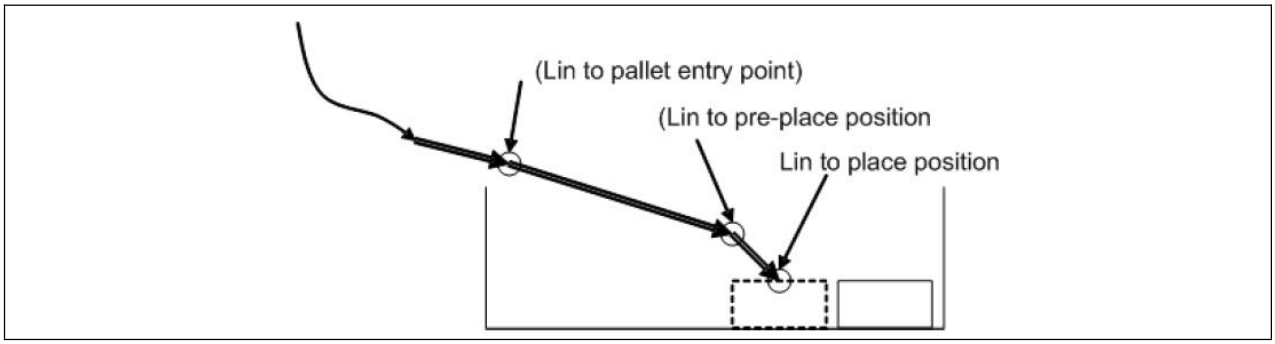


图 31 放置路径

在调用PalletToPut 指令时，执行了以下运动指令：

- 机器人以直线的运动方式移动至跺的进入位置（可选）
- 机器人以直线的运动方式移动至垛点的前点位置（可选）
- 机器人以直线的运动方式移动至工件的放置位置（必选）

PalletFromPut

放置工件完成后，机器人从放置位离离开的指令。

- 码垛名：当前码垛的PALLET变量。
- 目标速度：机器人运行速度。

```
PalletFromPut (PALLET2, 2000)
```

该指令是放置工件的相反动作，从放置好的工件位置离开。离开时，建议路径如下图所示。

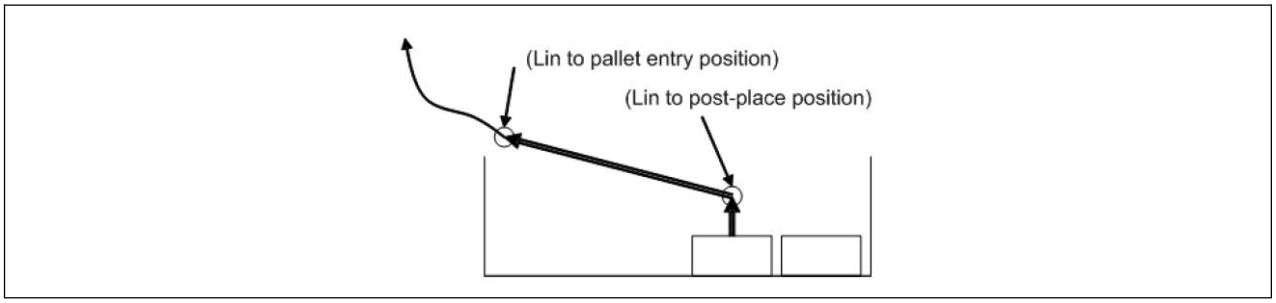


图 32 放置后离开的路径

在调用PalletFromPut 指令时，执行了以下运动指令：

- 机器人以直线的运动方式移动至垛点的后点位置（可选）
- 机器人以直线的运动方式移动至跺的进入位置（可选）

如果跺体的后点和进入位置都没有被设置，执行PalletFromPut 指令时，机器人保持当前的姿态不变。

PalletToGet

该指令是进行抓取工件动作，机器人执行该指令时，会从当前点移动到目标工件的码垛点进行抓取工件。

- 码垛名：当前码垛的PALLET变量。
- 目标速度：机器人运行速度。

```
PalletToGet (PALLET3, 2000)
```

抓取过程中，路径的规划建议如下图所示。

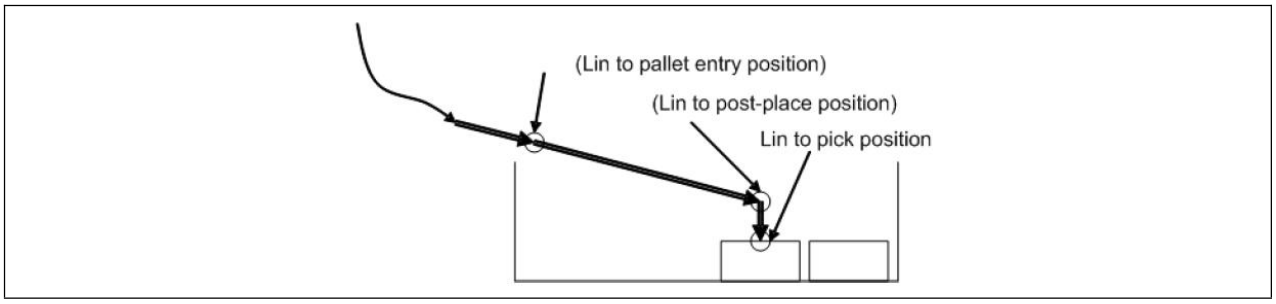


图 33 抓取工件路径

在调用PalletToGet 指令时，执行了以下运动指令：

- 机器人以直线的运动方式移动至跺的进入位置（可选）
- 机器人以直线的运动方式移动至垛点的前点位置（可选）

- 机器人以直线的运动方式移动至抓取位置（必选）

在这些运动序列之后，机器人到达工件的放置位置，然后关闭夹具，抓取工件。

PalletFromGet

该指令是进行抓取工件动作后离开的过程，机器人执行该指令时，会从抓取工件的码垛点离开。

- 码垛名：当前码垛的PALLET变量。
- 目标速度：机器人运行速度。

```
PalletFromGet (PALLET4, 2000)
```

抓取过程中，路径的规划建议如下图所示。

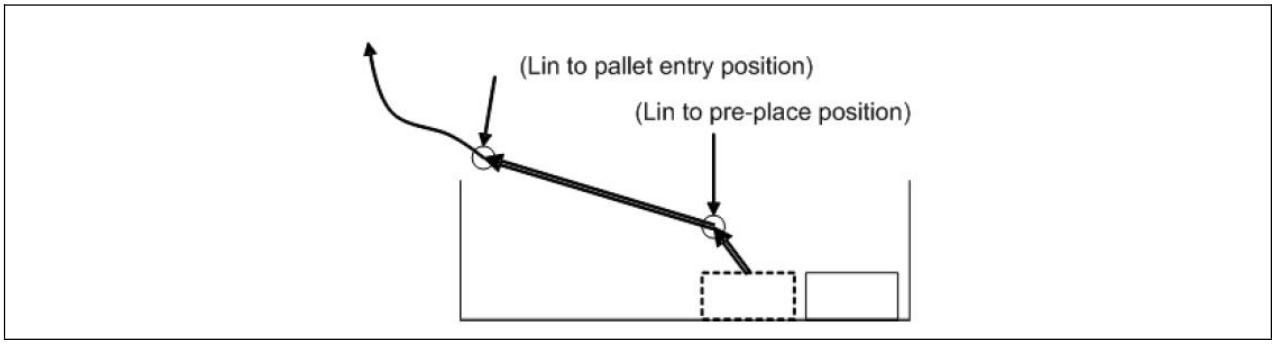


图 34 抓取后离开的路径

在调用PalletFromGet 指令时，执行了以下运动指令：

- 机器人以直线的运动方式移动至垛点的后点位置（可选）
- 机器人以直线的运动方式移动至码垛的进入位置（可选）

如果垛点的后点和进入位置都没有被设置，执行PalletFromGet 指令时，机器人保持当前的姿态不变。

Socket 指令

Socket指令列表如下图所示。

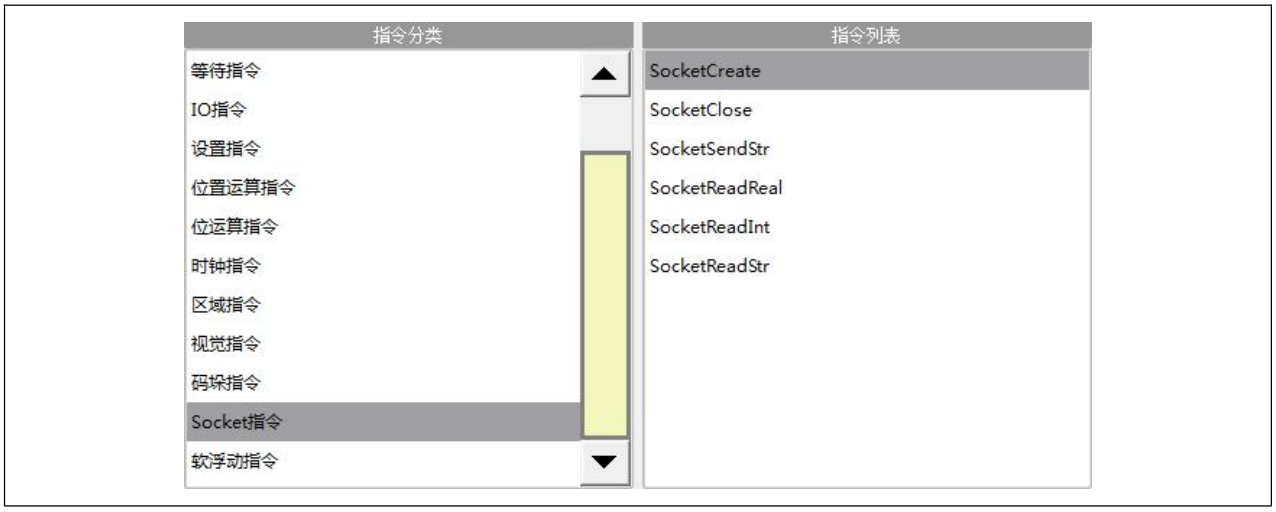


图 35 Socket 指令列表

SocketCreate

创建socket客户端，以便于对服务器进行数据的交互。

根据传入的服务器端参数，在本地创建客户端并与服务器端建立连接。

- socket名称：需要创建的socket名称，该值传入的是一个socket变量。
- IP地址：需要连接的服务器ip地址。
- 端口号：需要连接的服务器端口号。
- 操作返回值：创建操作的返回值变量，返回值为0表示成功，返回值为1则为失败。

SocketClose

关闭之前创建过的socket客户端。

根据传入的socket名，关闭创建过的客户端，并将操作的成功与否返回。

- socket名称：需要关闭的socket名称，该值传入的是一个socket变量。
- 操作返回值：创建操作的返回值变量，返回值为0表示成功，返回值为1则为失败。

SocketSendStr

向服务器端发送字符串，进行命令的交互。

向已经建立连接的服务器端发送字符串，并将成功的与否返回。

- socket名称：需要执行发送操作的socket名称，该值传入的是一个socket变量。
- 发送的字符串：需要发给服务器端的字符串数据。
- 操作返回值：创建操作的返回值变量，返回值为0表示成功，返回值为1则为失败。

SocketReadReal

读取服务器端发来的字符串，并以real数组的形式存储。

等待并接收服务器端发送的字符串，其格式为[1.1,2.2,3.3,4.4](即数据起末为'[]'，数字与数字之间用','隔开)，当接收到该字符串后，机器人系统会将其拆分解析并按顺序存放到数组中。

- socket名称：需要执行读取操作的socket名称，该值传入的是一个socket变量。
- 读取的数据个数：需要读取的数据存储到数组的个数。
- 数据返回值：将读取并转换后的值存入到数组变量中，并将该数组变量返回。
- 检测时间：等待服务器端发送数据的等待时间。超时报警。
- 操作返回值：创建操作的返回值变量，返回值为0表示成功，返回值为1则为失败。

SocketReadInt

读取服务器端发来的字符串，并以int数组的形式存储。

等待并接收服务器端发送的字符串，其格式为[1,2,3,4](即数据起末为'[]'，数字与数字之间用','隔开)，当接收到该字符串后，机器人系统会将其拆分解析并按顺序存放到数组中。

- socket名称：需要执行读取操作的socket名称，该值传入的是一个socket变量。

- 读取的数据个数：需要读取的数据存储到数组的个数。
- 数据返回值：将读取并转换后的值存入到数组变量中，并将该数组变量返回。
- 检测时间：等待服务器端发送数据的等待时间。超时报警。
- 操作返回值：创建操作的返回值变量，返回值为0表示成功，返回值为1则为失败。

SocketReadStr

读取服务器端发来的字符串，并以string变量的形式存储。

等待并接收服务器端发送的字符串，其格式为盘[Hello world!](即数据起末为'[]')，当接收到该字符串后，机器人系统会将其存放到字符串变量中。

- socket名称：需要执行读取操作的socket名称，该值传入的是一个socket变量。
- 读取的字符串：服务器端发送来的字符串数据，该值以一个string变量的形式返回。
- 检测时间：等待服务器端发送数据的等待时间。超时报警。
- 操作返回值：创建操作的返回值变量，返回值为0表示成功，返回值为1则为失败。

软浮动指令

软浮动指令列表如下图所示。

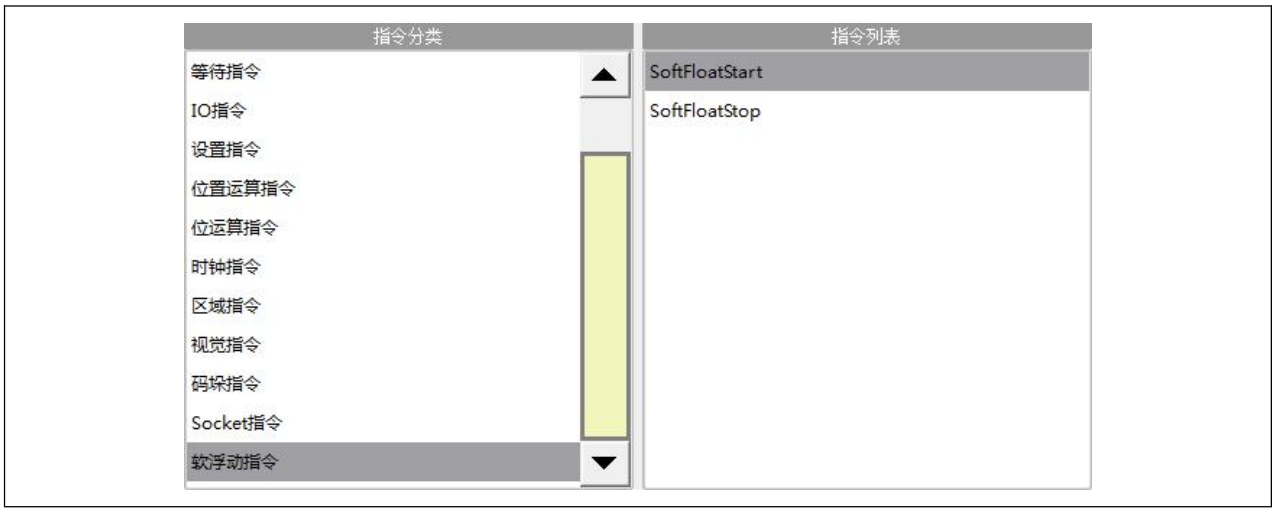


图 36 软浮动指令列表

SoftFloatStart

软浮动开始指令。开启软浮动功能并设置软浮动参数。

- 软浮动类型：
  - 直线软浮动。目前只支持直线软浮动。
- 软浮动参考坐标系：
  - WORLD：世界坐标系
  - USER：工具坐标系
  - TOOL：用户坐标系
- 坐标系号：当选择参考坐标系为WORLD时，坐标系号默认为空，不显示。当参考坐标系为TOOl

或者USER时，在下拉列表中选择已有的坐标系号或在弹出的对话框中新建并示教一个坐标系号。

- 软浮动方向：
  - X：所选参考坐标系或坐标系号的 X 方向
  - Y：所选参考坐标系或坐标系号的 Y 方向
  - Z：所选参考坐标系或坐标系号的 Z 方向
- 软浮动灵敏度：软浮动柔顺度参数。
  - HIGH：柔顺等级最高
  - MidHigh：柔顺等级次高
  - MID：中等柔顺
  - MidLow：柔顺等级次低
  - LOW：柔顺等级最低

其中，设定软浮动方向的说明（详见软浮动功能说明）。

SoftFloatStop

软浮动结束指令，关闭软浮动功能。

完成软浮动功能后，调用此指令关闭软浮动功能。

数学运算函数

在“IF”指令和“...=...”指令中，可能会使用到数学运算函数或字符串运算函数。本节是对所能使用到的“数学函数”进行说明。

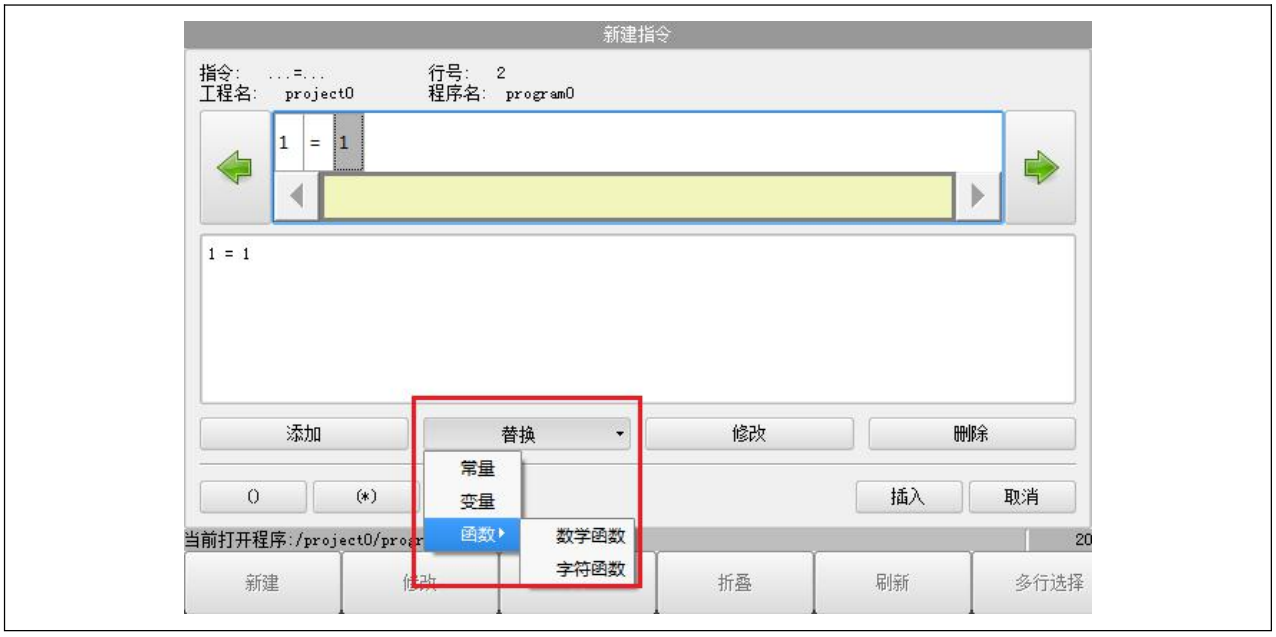


图 37 数学运算函数界面

sin

正弦三角函数。

- 参数1：int或real型变量或常量。

- 函数返回值：real型常量。

cos

余弦三角函数。

- 参数1：int或real型变量或常量。
- 函数返回值：real型常量。

tan

正切三角函数。

- 参数1：int或real型变量或常量。
- 函数返回值：real型常量。

asin

反正弦三角函数。

- 参数1：int或real型变量或常量。
- 函数返回值：real型常量。

acos

反余弦三角函数。

- 参数1：int或real型变量或常量。
- 函数返回值：real型常量。

atan

反正切三角函数。

- 参数1：int或real型变量或常量。
- 函数返回值：real型常量。

atan2

返回X轴到(x, y)点的弧度值。

- 参数1：int或real型变量或常量。
- 参数2：int或real型变量或常量。
- 函数返回值：real型常量。

sinh

双曲正弦函数。

- 参数1：int或real型变量或常量。
- 函数返回值：real型常量。

cosh

双曲余弦函数。

- 参数1：int或real型变量或常量。
- 函数返回值：real型常量。

tanh

双曲正切函数。

- 参数1：int或real型变量或常量。
- 函数返回值：real型常量。

log

自然对数函数。

- 参数1：int或real型变量或常量。
- 函数返回值：real型常量。

log10

以10为底的对数函数。

- 参数1：int或real型变量或常量。
- 函数返回值：real型常量。

sqrt

开平方根函数。

- 参数1：int或real型变量或常量。
- 函数返回值：real型常量。

exp

以e为底的指数函数。

- 参数1：int或real型变量或常量。

- 函数返回值：real型常量。

pow

指数函数。

- 参数1：int或real型变量或常量，底数。
- 参数1：int或real型变量或常量，指数。
- 函数返回值：real型常量。

deg

弧度转角度函数。

- 参数1：int或real型变量或常量。
- 函数返回值：real型常量。

rad

角度转弧度函数。

- 参数1：int或real型变量或常量。
- 函数返回值：real型常量。

fmod

取余函数。

- 参数1：int或real型变量或常量，被除数。
- 参数2：int或real型变量或常量，除数。
- 函数返回值：real型常量。

floor

向下取整函数。

- 参数1：int或real型变量或常量。
- 函数返回值：int型常量。

random

2个参数之间取随机数。

- 参数1：int或real型变量或常量。
- 参数2：int或real型变量或常量。
- 函数返回值：int型常量。

字符运算函数

在“IF”指令和“...=...”指令中，可能会使用到数学运算函数或字符串运算函数。本节是对所能使用到的“字符函数”进行说明。

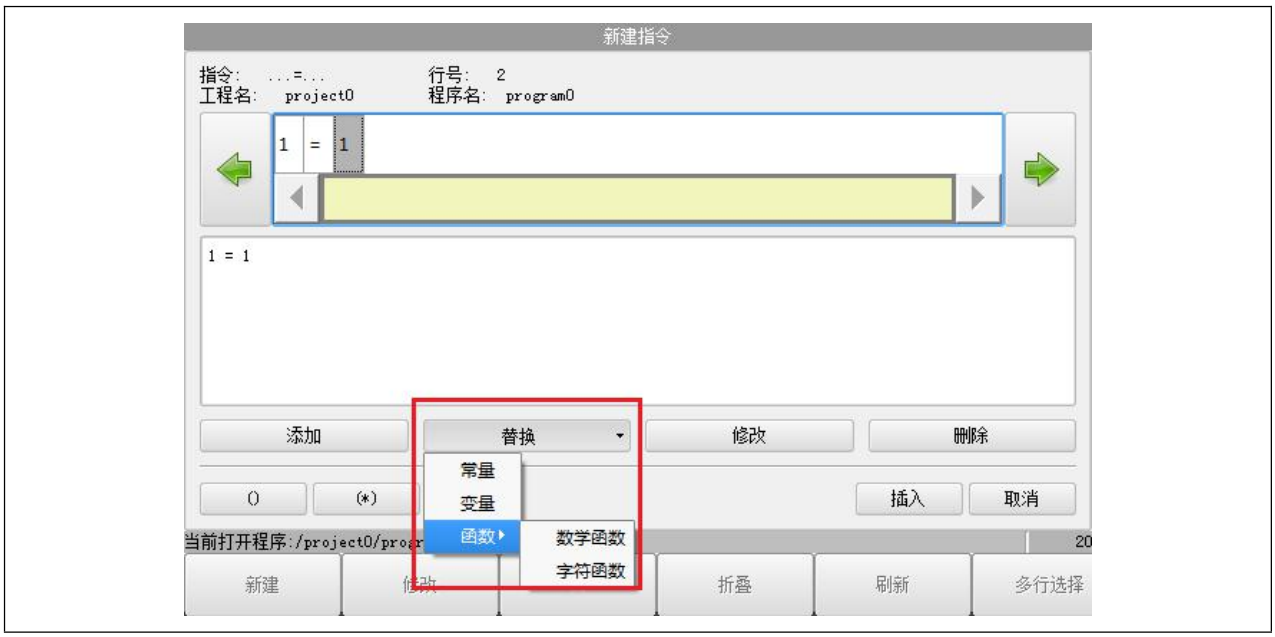


图 38 字符运算函数界面

len

计算字符串长度。

- 参数1：string型变量或常量。
- 函数返回值：int型常量。

byte

取字符串第n位的字符的ASCCII码。

- 参数1：string型变量或常量。
- 参数2：int型变量或常量。
- 函数返回值：int型常量。

char

返回ASCCII码对应的字符。

- 参数1：int型变量或常量。
- 函数返回值：string型常量。

find

返回字符串中子字符串的位置。

- 参数1：string型变量或常量。
- 参数2：string型变量或常量
- 函数返回值：int型常量。(当找不到对应字符或字符串时，返回值为-1)

sub

返回字符串s的第a到b位。

- 参数1：string型变量或常量。
- 参数2：int型变量或常量。
- 参数3：int型变量或常量。
- 函数返回值：string型常量。

gsub

在字符串s内搜索a子字符串，并用字符串b替换a。

- 参数1：string型变量或常量。
- 参数2：string型变量或常量。
- 参数3：string型变量或常量。
- 函数返回值：string型常量。

lower

返回字符串的小写格式。

- 参数1：string型变量或常量。
- 函数返回值：string型常量。

upper

返回字符串的大写格式。

- 参数1：string型变量或常量。
- 函数返回值：string型常量。