XYZ Studio Max User's Manual

XYZ ROBOTICS.INC

2023 年 09 月 12 日

软件概述

I 软件概述 2

1 简介 3

2 软件安装 4

3 功能结构 7

4 软件界面 10

II 基础操作 14

5 新建项目 15

6 连接机械臂 18

7 基本视觉操作 22

8 基本运动操作 51

9 基本任务操作 91

III 典型场景 97

10 2D/3D 坐标移动 98

11 3D 轨迹移动 132

12 拆码垛场景 253

IV 附录 271

13 IP 设置 272

14 安装机械臂驱动 280

15 PLC 连接 687

16 通信协议 706
17 术语 722
18 修订记录 724

本文档介绍 XYZ Studio Max 软件及其使用方法。在使用软件之前，请认真阅读本手册并妥善保存以便日后参考。

Part I

软件概述

CHAPTER 1

简介

XYZ Studio Max（以下简称 Max）是一款集拆码垛、无序工件上下料、免注册货品抓取、精确定位等先进功能于一体的工业软件。通过图形化的界面，无须编写代码即可使用多种前沿算法，满足复杂多样的需求。

上手快

- 一站式操作：集成视觉服务、运动规划和任务规划功能，一站式完成视觉空间设置、抓取方式和轨迹注册、轨迹规划等操作。
- 开箱即用：内置多个应用模板，深度适配行业典型场景，实现快速部署。
- 免示教：可在仿真环境中，自动生成无碰撞轨迹，自动避障并规划最优路径。
- 图形化编程：简单拖拽内置算法模块，即可实现流程的搭建，无需代码编程。
- 调试方便：支持单步运行，同时支持各种数据类型可视化。

功能强

- 全场景：采用 2D+3D 视觉结合的方式，满足多元化工业和物流应用需求。
- 适应性强：通过深度学习等领先算法，准确识别多种随机物料。

CHAPTER 2

软件安装

本章介绍 Max 软件下载和安装的步骤。

2.1 安装步骤

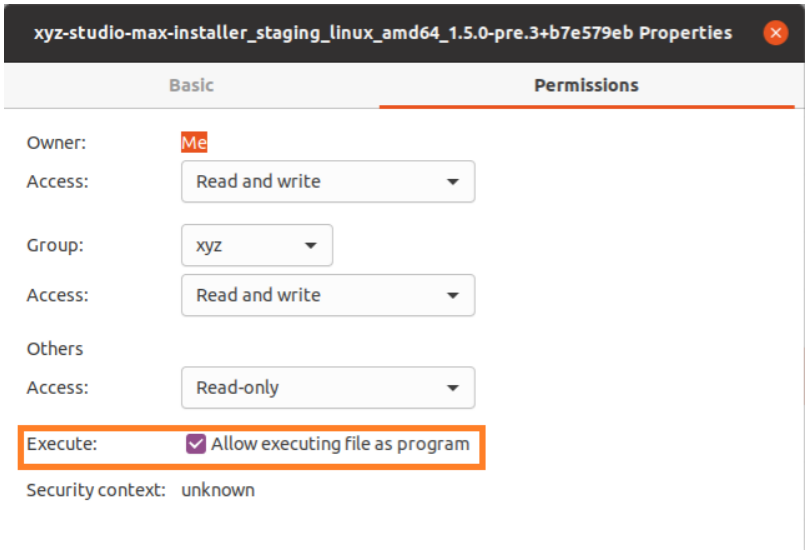
1. 登录 XYZ Studio Max 资源中心，用户名及密码与 xyz_apt/csc 账户相同，当前可在资料网站激活或者更新密码。登录成功后，界面如下。

2. 点击 下载按钮，根据需要选择下载 Max 安装包，也可单击 **更多版本**查看历史版本。

   下载安装包之后，双击 exe 文件，打开 Studio Max Installer 安装程序，单击 下一步开始安装。

提示:

- 预览版是开发过程中的软件包，仅供体验试用。
- 在 Linux 系统中，需要将安装包的 **属性**勾选 **Execute** 属性，然后方可安装，如下图所示。



3. 单击 浏览指定安装目录，然后单击 下一步。

提示: 安装目录必须是空文件夹，否则将无法继续安装。

4. 根据安装向导提示依次勾选 **XYZ Studio Max** 组件，阅读并接受许可协议，选择开始菜单快捷方式（一般保持默认即可）。
5. 完成上述安装设置后，单击 安装。
6. 安装完成后，单击 完成。在 Windows 系统的电脑桌面或者 Linux 系统的应用栏双击  即可打开软件。

2.2 安装许可

下载安装完成之后，还需要使用 CodeMeter 激活软件许可。

1. 登录 XYZ Studio Max 资源中心，下载激活软件 CodeMeter 和许可。
2. 软件许可下载完成后解压缩到本地文件夹，并双击 license.WibuCmRaU 文件，CodeMeter 会自动导入许可。

提示:

- 试用版许可有效期为 3 个月，过期后无法使用 Max，需要重新获取并导入许可。
- 实际部署项目时，请联系星猿哲工程师，获取正式版许可。

CHAPTER 3

功能结构

XYZ Studio Max 主要有三大功能：

- 视觉：获取视觉图像并根据拍照结果和工件模型，确定物体位置。
- 运动：编辑运动环境，完成运动规划所需的设置。
- 任务：执行机械臂运动规划，在不同工作空间之间完成拣选。

XYZ Studio Max 的使用流程如下。

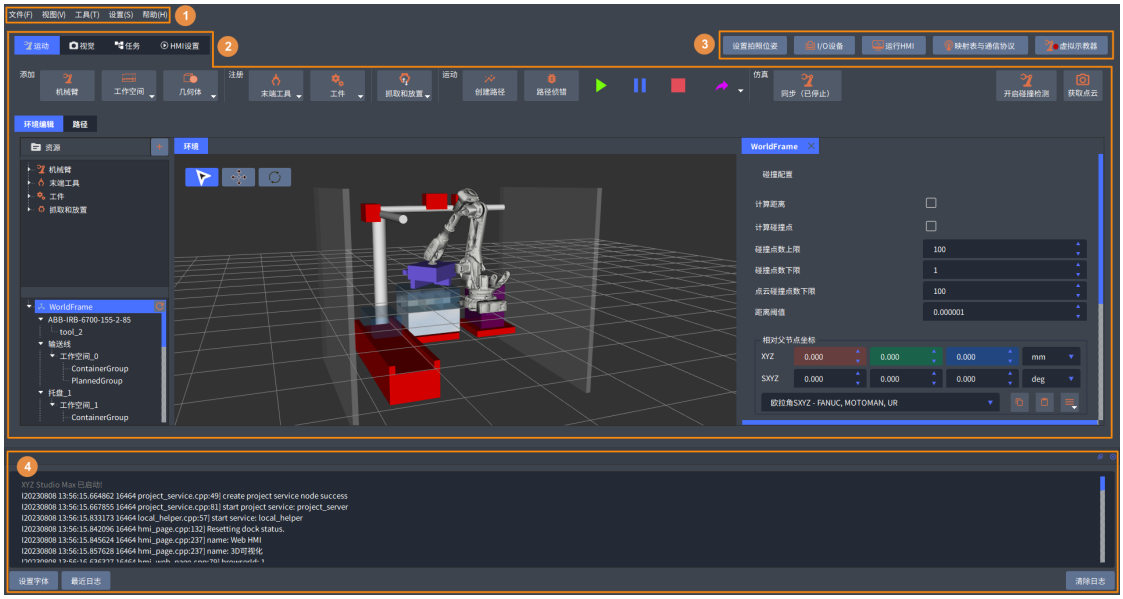
流程	步骤
	1. 新建项目。 2. 连接机械臂。 3. 视觉设置。 • 连接相机 • 创建流图 • 标定 • 服务 4. 运动设置。 • 添加工作空间、添加几何体 • 注册末端工具 • 注册工件 • 注册抓取方式、注册放置方式 • 注册抓放规划 • 设置运动路径 • 路径侦错 5. 任务设置：流图配置。

提示： 在实际项目执行过程中，运动和 视觉的设置可交替进行，具体根据实际情况灵活调整。

CHAPTER 4

软件界面

Max 的通用设置包括菜单栏、界面切换、日志模块、设置拍照位姿和虚拟示教器等功能，这些功能如下图所示。



1. 菜单栏：
    - 文件：新建、打开、保存、关闭项目。
    - 视图：开启 **节点视图**，可在 视觉页签下查看视觉流图；开启 **日志模块**，可查看操作日志。
    - 工具：提供路径侦错、模型编辑器、进程管理、视觉重复精度检测等工具。
    - 设置：设置字体、语言、启用页面等。部分设置需在软件重启后生效。
    - 帮助：查看使用手册、变更日志、机械臂代码，访问资源中心，提供用户反馈。
2. 界面切换：可切换至 视觉、运动、任务和 *HMI* 设置页签。

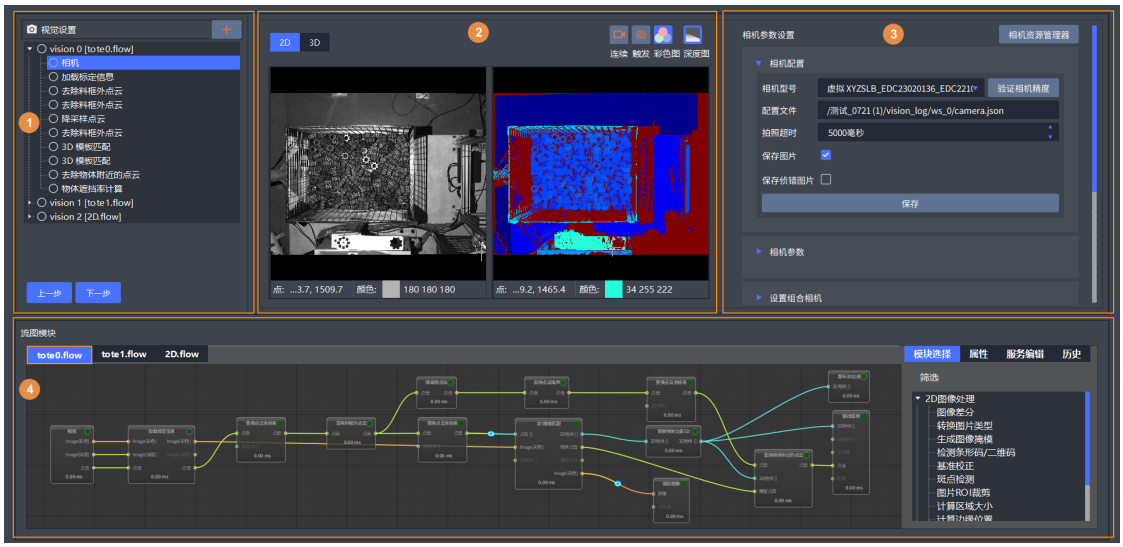
3. 机械臂设置：
    - 设置拍照位姿：当相机安装在机械臂上时，视觉服务需使用拍照位姿来校准点云位姿。
    - I/O 设备：添加并管理 I/O 设备。
    - 运行 HMI：在完成 HMI 设置后，单击此按钮，运行 HMI，单击 切换至配置模式，退出 HMI。
    - 映射表与通信协议：可设置工件代号映射表、任务流图映射表，查看并修改机械臂基础设置。
    - 虚拟示教器：连接并控制机械臂。连接机械臂前，请确保 robot_driver_node 进程已正确开启（顶部工具栏“工具 > 进程管理”）
4. 日志模块：在顶部菜单栏，选择“视图 > 日志模块”，显示日志模块，可查看用户操作日志。

通用功能设置好之后，在 视觉、运动、任务页签下分别完成视觉服务、运动空间及路径、任务流程等功能设置。

备注： 本手册中的软件界面截图仅供参考，请以实际界面为准。

4.1 视觉

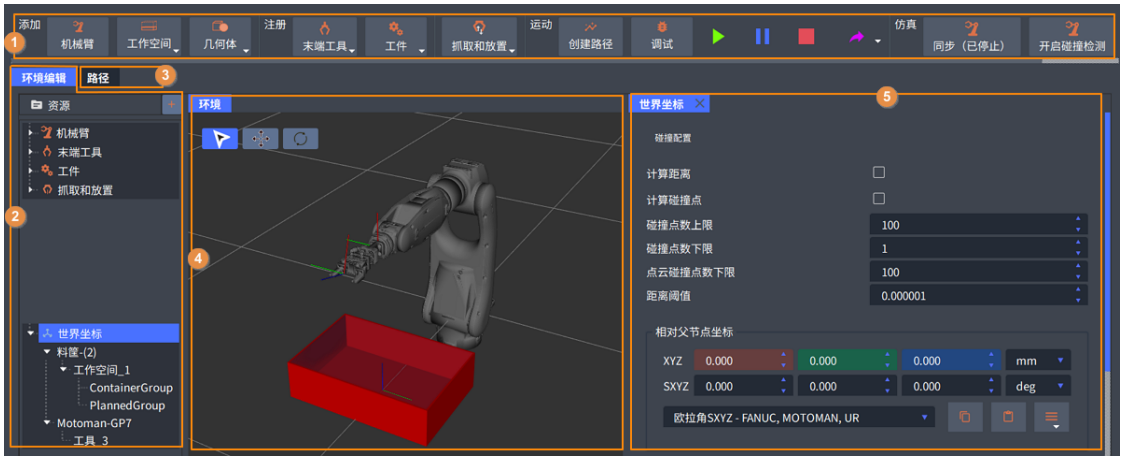
视觉界面如下图所示：



1. 关键节点列表：显示流图中添加的关键节点模块。单击任意模块，下方流图会定位至相应节点。
2. 预览：可视化预览窗口，显示彩色图、点云、深度图等。
3. 参数设置栏：设置各功能模块的参数。
4. 节点视图：在顶部工具栏，选择“视图 > 节点视图”，显示节点流图。
    - 右键单击中间的流图窗口，可选择添加、复制、删除节点，返回上一级界面，或自动调整显示比例。
    - 滑动鼠标滚轮，可缩放流图。
    - 在右侧功能配置区，可添加节点，配置参数，设置服务，查看操作记录。
    - 向上拖动节点视图上方分隔线，可隐藏其它区域。

4.2 运动

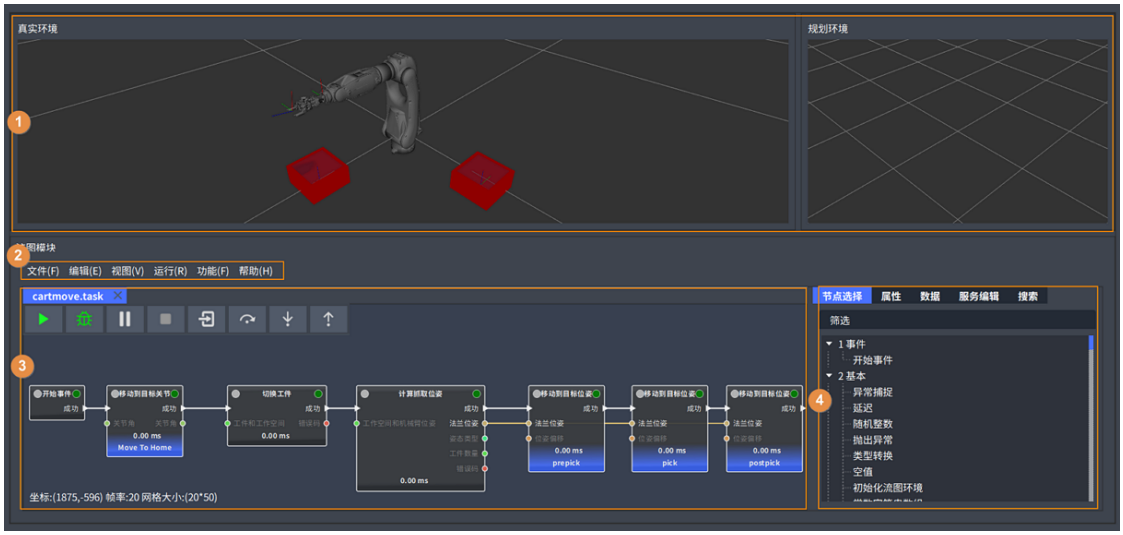
运动界面如下图所示：



1. 工具栏：单击相应按钮，完成常用运动设置，如添加机械臂、工件、夹具、障碍物，创建运动路径，进行仿真模拟等。
2. 环境编辑：分为资源树和场景树上下两部分。
   - 在资源树下可添加机械臂、工件、工具等。资源树中的 **抓取和放置**、**工件**、**机械臂**、**末端工具**分别对应“项目文件夹\resources”下的 grasps_places、objects、robots、tools 文件夹，在 Max 中添加机械臂、抓取方式等后，会在本地文件夹中生成相应文件，反之在本地文件夹添加文件后，也会同步显示在软件界面。
   - 场景树显示所有已添加元素的对应节点，右键单击某一节点，可进行复制、删除、添加子元素等操作。
3. 路径：添加运动规划路径。
4. 环境预览：显示模拟真实环境的 3D 空间，可调整空间中工件、夹具、料筐等各元素的位置、姿态。
   - 单击画面，滑动鼠标滚轮，可缩放画面。
   - 按住鼠标右键，拖动鼠标，可调整画面视角。
   - 按住鼠标滚轮，移动鼠标，可拖动画面。
5. 参数设置栏：根据选择的元素节点设置相应参数。

4.3 任务

任务界面如下图所示：



1. 展示真实环境和规划环境。
2. 任务菜单栏：提供任务相关的功能设置，主要包括文件、编辑、运行、清理、帧率、功能等方面的设置。
3. 任务流程显示区：显示配置流程图，可同时打开多个任务文件。同时左上角还有流图调试工具，用于完成开始、暂停、跳过等调试步骤。
4. 模块设置：提供常用模块的选择、属性设置等功能。
    - 节点选择：在结构树中选择所需模块，或根据关键字搜索模块名称，然后将模块拖拽至流图中。
    - 属性：选中模块之后，可在 属性页签中设置模块参数。
    - 数据：展示变量表和统计数据表。
    - 服务编辑：编辑流图服务，可以添加指令，设置根节点、叶节点等，机械臂主控时使用。
    - 搜索：可根据关键字搜索当前流图中的模块，也可查看当前流图中的函数表。

Part II

基础操作

CHAPTER 5

新建项目

使用 Max 时，首先需要新建项目。

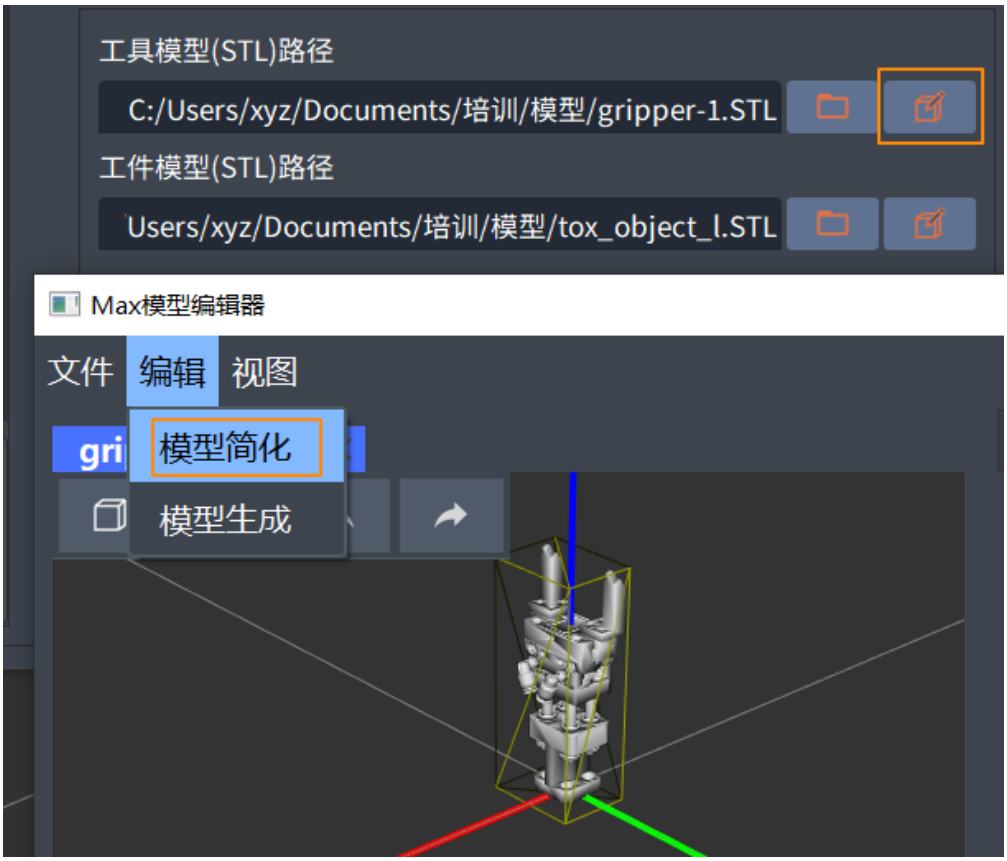
1. 在 Max 顶部工具栏，选择“文件 > 新建项目”。

2. 根据使用场景选择已有模板，预置模板包含视觉、运动、任务的基础设置，也可选择创建空白项目。
3. 设置项目名称、项目路径，单击 选择机械臂，选择机械臂类型并完成添加。

- 常规机械臂：选用 Max 内置的机械臂型模型。在左侧搜索栏搜索或根据品牌、轴数、臂展等条件筛选机械臂型号，单击待添加的机械臂型号，再单击右下角 确定。
- 导入机械臂：从本地导入机械臂 urdf 模型。
- 自定义机械臂：自定义添加机械臂时，需设置机械臂的轴数和欧拉角类型。

4. 勾选 高级参数后，设置工具、工件模型等高级参数。也可不在新建项目时设置高级参数，待项目创建完成后，在 运动、视觉、任务页签下再作设置。

    a. 添加工具和工件的 STL 模型。添加模型后，可单击  ，打开模型编辑器，选择顶部工具栏“编辑 > 模型简化”，简化模型以提高计算效率。



b. 设置工作空间类型。部分场景仅涉及抓取工作空间，部分场景需设置抓取工作空间和放置工作空间。

c. 选择任务模板。场景不同，支持的任务模板可能不同。

5. 单击 确认。

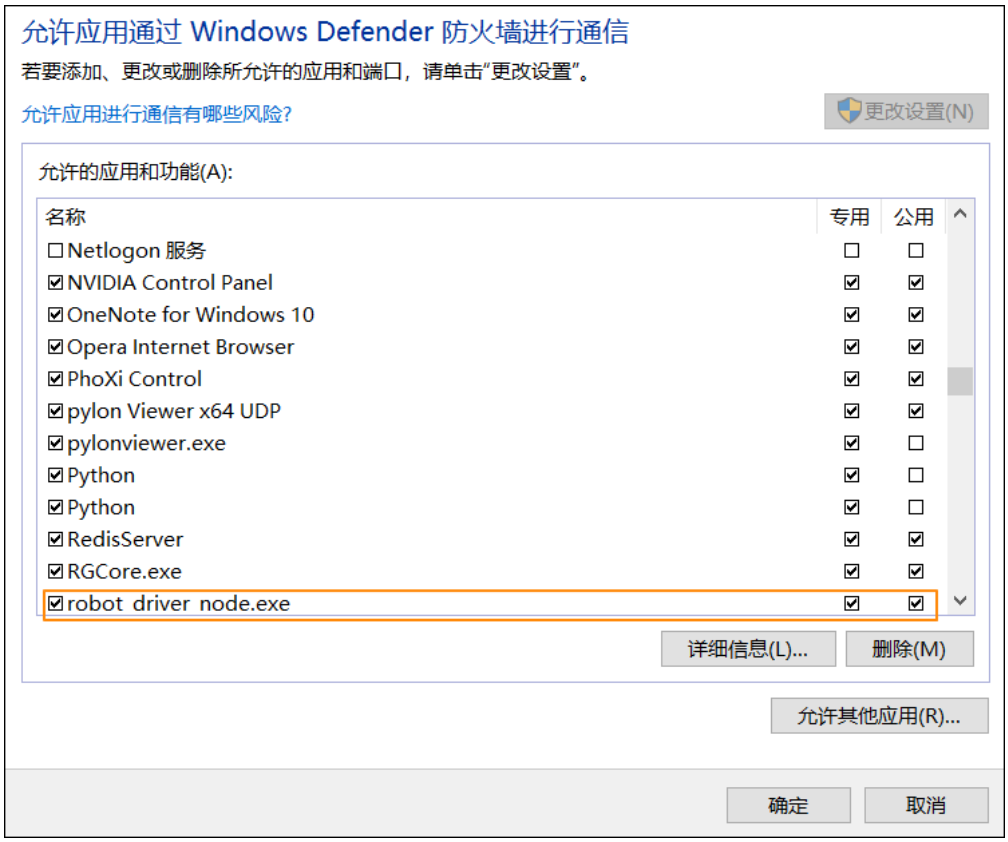
CHAPTER 6

连接机械臂

前提条件

连接机械臂前，需完成以下设置：

- 在 Windows 安全中心关闭防火墙。在 Windows 左下角搜索栏搜索并打开“防火墙和网络保护”，关闭域网络、专用网络、公用网络的防火墙。然后单击 *允许应用通过防火墙*，单击 *允许其他应用*，将“Max 安装目录\bin”文件夹下的 robot_driver_node.exe 添加到 **允许的应用和功能**列表中，勾选对应的 **专用**和 **公用**复选框，单击 *确定*，使 robot_driver_node.exe 能够通过专用和公用网络。



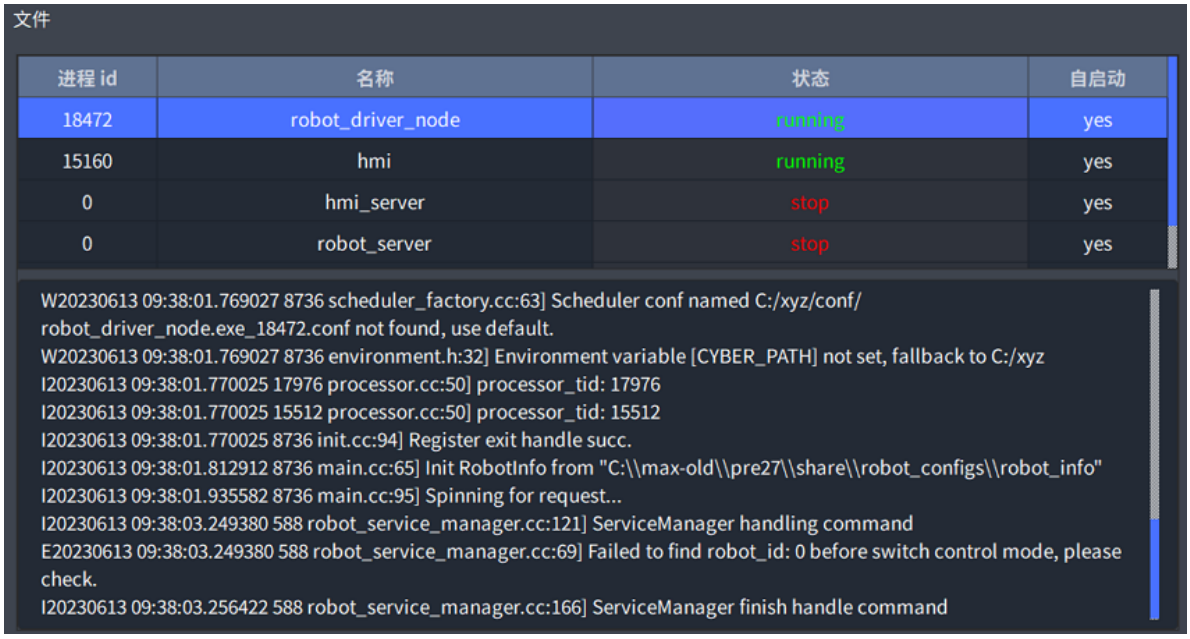
- 在 *运动*页签下添加并加载对应型号的机械臂 urdf 模型，具体操作请参见*添加机械臂*。

操作步骤

1. 在 Max 顶部工具栏，选择“工具 > 进程管理”，查看进程状态。

   - robot_driver_node：工控机主控驱动程序。在工控机主控模式下，由工控机根据任务主动下发各类指令给机械臂/PLC 等设备让其执行。
   - robot_server：机械臂主控驱动程序。在机械臂主控模式下，由机械臂（包含 PLC 等设备）根据任务主动发送各类指令给工控机获取信息。

   两者的定义和使用场景可以参考*机械臂主控（Robot Master）*，*工控机主控（IPC Master）*。



2. 如果 robot_driver_node 进程处于停止状态，右键单击进程，选择 *启动进程*，确保下方日志无报错。

   如果需要使用机械臂主控模式，则需要启动 robot_server 进程。

提示： 右键单击进程，可启动、关闭、删除进程，并设置配置文件路径等运行参数。robot_driver_node 和 robot_server 对应的配置文件分别为“项目文件夹\robot_configs”下的 robot_config.yaml 和 robot_server_config.yml。

---

3. 在机械臂上完成准备工作，请参考安装机械臂驱动。
4. 单击 Max 软件界面右上角 *虚拟示教器*，单击 *连接机械臂*。仿真运行时，选择 **仅连接到虚拟机械臂**，然后单击右上角 *连接虚拟机械臂*。


虚拟示教器
机械臂断连，请连接机械臂
机械臂连接
基础控制
I/O 控制
连接机械臂
虚拟示教器
创建
（已停止）
开启碰撞检测
连接设置
机械臂能力
机械臂ID:
0: motoman_gp7
厂商:
motoman
型号:
gp7
机械臂轴数:
6
欧拉角类型:
sxyz
机械臂控制器:
YRC1000
机械臂运动控制IP:
0.0.0.0
机械臂运动控制端口:
5000
机械臂状态反馈IP:
0.0.0.0
机械臂状态反馈端口:
5001
仅连接到虚拟机械臂
操作时请将实体示教器置于手边，必要时使用实体示教器上的急停按钮停下机械臂

CHAPTER 7

基本视觉操作

本章介绍 视觉界面的基本操作，通过配置模块和流图，完成连接相机、标定、配置服务等。

7.1 模块结构

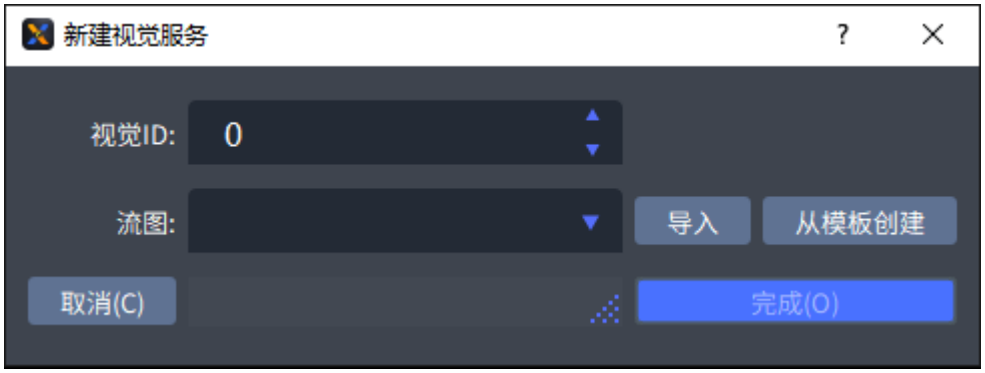
模块是视觉算法流程运行的最小单元，将各模块连接起来，可按不同需求实现各种视觉处理功能。视觉流图中的模块结构及说明如下。

1. 模块名称
2. 模块单独运行按钮
3. 输入，是一个模块运行时所需的数据
4. 输出，是一个模块经过模块中算法计算后得出的结果
5. 模块单次运行耗时

7.2 创建流图

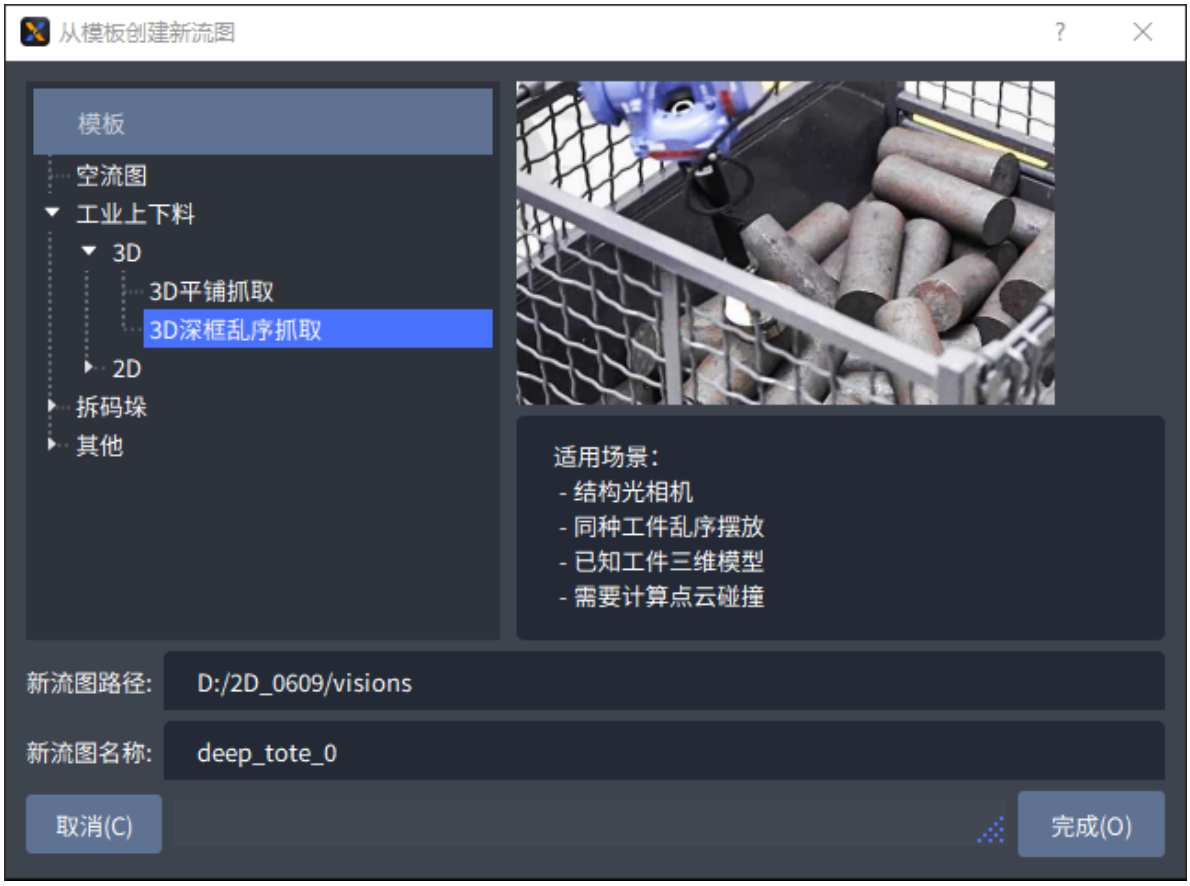
视觉内置了多种典型场景的流图模板，下面介绍创建流图的方法。

1. 单击 视觉页签，单击 视觉设置右侧  ，创建视觉服务。
2. 在弹出的 新建视觉服务窗口设置视觉 ID。



提示： 视觉 ID 不可重复，建议从 0 开始依次递增。

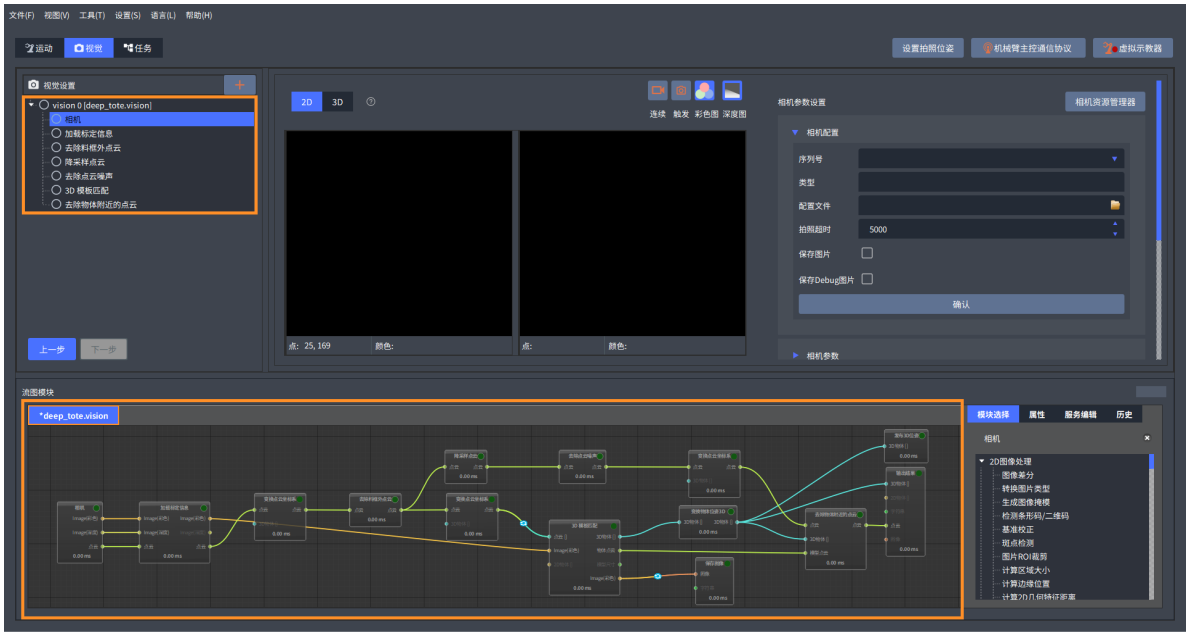
3. 单击 从模板创建，根据需要选择合适的模板，如下图中在弹出窗口选择 **工业上下料 > 3D > 3D 深筐乱序抓取**，并设置流图名称，然后单击 完成，完成视觉流图创建。



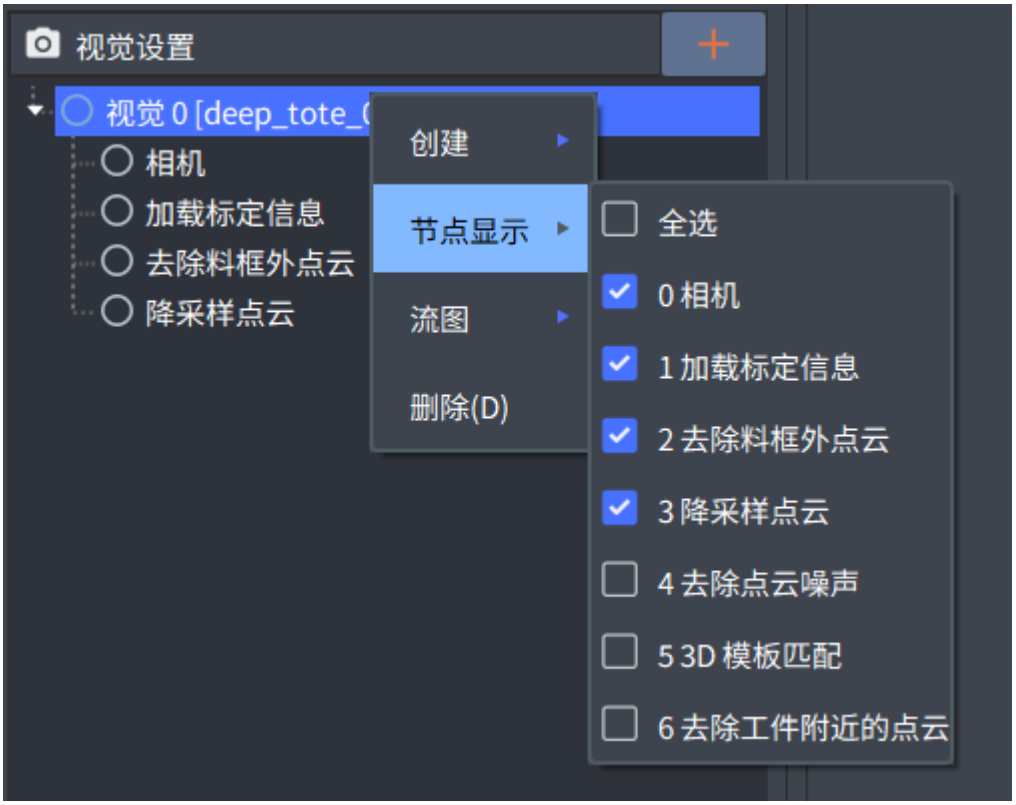
提示:

- 若要创建全新空白流图，可单击 从模板创建后选择 空流图。
- 在 新建视觉服务窗口单击 导入，打开已有的视觉流图文件。

4. 创建成功后，在界面左上角关键节点列表查看流图中的关键节点模块。在顶部工具栏选择 “视图 > 节点视图”，在界面下方的节点视图中查看所有已添加的节点。



右键单击创建好的视觉节点，可在 **节点显示**菜单中勾选想要查看的关键节点，隐藏不想查看的节点。



5. 编辑流图及模块。

   - **添加模块**：右键单击流图窗口，选择 *添加模块*，搜索添加模块。也可在界面右下角 *模块选择*页签下搜索模块，将模块拖入流图窗口。
   - **配置模块**：在流图窗口中单击需配置的模块，在界面右上角参数配置栏或右下角 *属性*页签配置参数。配置完成后，单击界面左上角 *文件*，选择 *保存项目*。

小技巧： 鼠标指向 *属性*页签下参数，可查看操作提示。

- **连接模块**：在流图中单击模块右侧端口，按住鼠标左键拖出曲线，引向要连接的模块左侧的端口，即可完成连接。

小技巧：

- 相同数据类型的端口可建立连接，不同数据类型的端口不可连接。部分端口支持隐式类型转换，虽然数据类型不同，也可建立连接。
- 鼠标指向端口名称，可查看端口的数据类型和简要说明，如下图所示。

- 双击连接线，可查看模块输出的数据。

- **复制或删除模块**：
  - 多选模块：按住【Shift】键，按住鼠标左键框选模块和连线，或按住【Ctrl】键，逐个点选模块和连线。
  - 复制并粘贴模块：单击模块，先按【Ctrl+C】再按【Ctrl+V】。
  - 删除模块或连线：单击模块或连线，按【Delete】键。

6. 运行节点。
   - 运行单个节点：单击模块右上角绿色按钮。
   - 运行命令节点：通过服务运行流图中一串指定的模块。详细操作，请参见服务。

提示： 保存视觉流图时，在 Max 左上方单击“文件 > 保存项目”即可。

关于各场景的流图配置，请分别参见“典型场景”部分的内容。

7.3 连接相机

前提条件

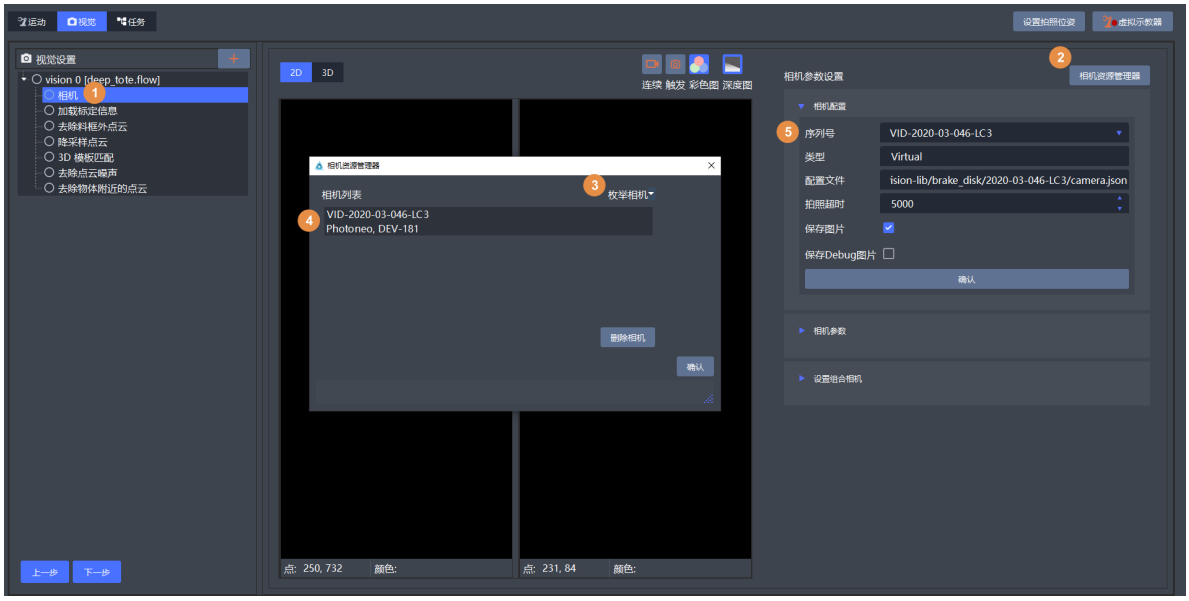
在 Max 上设置相机前，请确保相机和工控机在同一网段，相关设置请参考设置工控机 *IP* 地址 和设置相机 *IP*。

操作步骤

在 Max 上枚举并连接网络相机。

1. 在 *视觉*界面，打开 **相机**模块。
2. 单击右上角 *相机资源管理器*。

3. 单击 枚举相机，软件会查找、显示与本机同网段的相机。
4. 选择相机，单击 确认。
5. 选择相机序列号，并确认配置文件，单击 确认。



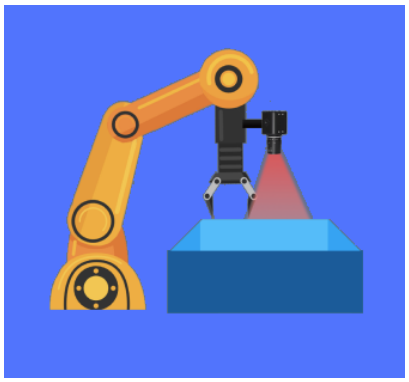
7.4 标定

标定是指获取机械臂坐标系和相机坐标系的转化关系，将视觉识别的结果转移到机械臂坐标系下，同时标定可用于校正由于镜头畸变造成的图像扭曲。

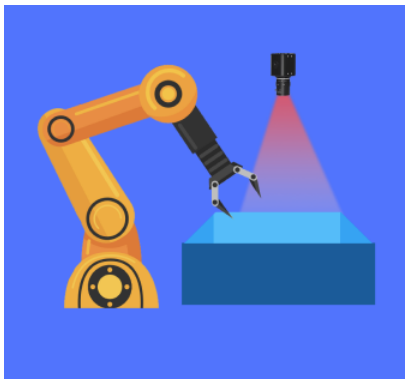
相机的安装方式

相机的安装方式有两种，安装在机械臂上和固定在场景中。

下图所示的是相机安装在机械臂上的方式，该方式又被称为眼在手上（eye on hand）。在该方式中，相机固定于机械臂末端法兰上，相机与机械臂法兰的相对位置是固定的。



下图所示的是相机固定在场景中的方式，该方式又被称为眼在手外（eye to hand）。在该方式中，相机独立安装在固定支架上。



标定的分类

标定分为内参标定和外参标定两个部分。

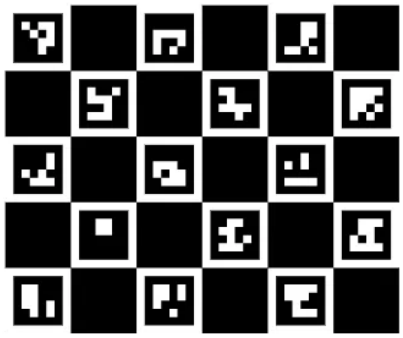
- 内参标定：内参指的是相机内部固有的基本参数，包括镜头焦距、像素大小、畸变系数等，内参标定的目的在于准确计算出这些参数。该标定方式仅针对 2D 相机，3D 相机无需内参标定。
- 外参标定：外参指的是相机在世界坐标系中的位置、旋转方向等参数，外参标定又可分为手眼标定和双目标定两种。
  - 手眼标定：获取相机和机械臂间的转换关系，可分为如下两种方式。
    - 相机在机械臂上：即眼在手上（eye on hand），可使用彩色图和点云标定。
    - 相机固定在场景：即眼在手外（eye to hand），也可使用彩色图和点云标定。
  - 双目标定：获得两台相机之间的相互位置关系的标定，用于获取物体的深度信息，并通过相机外参，将世界坐标系与相机坐标系联系起来。进行双目标定之前，需要先通过手眼标定确定主相机的手眼关系，然后再对从相机进行标定。

标定板的分类

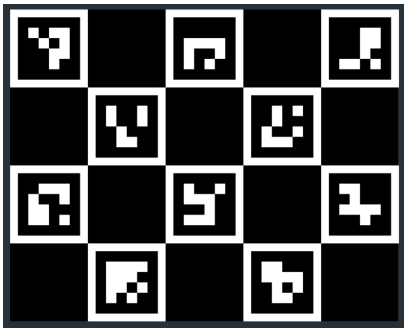
标定板是有固定间距图案阵列的平板，通过相机拍摄带有标定板的图片，然后通过计算以完成标定任务。

目前 Max 支持的常用标定板有三种：

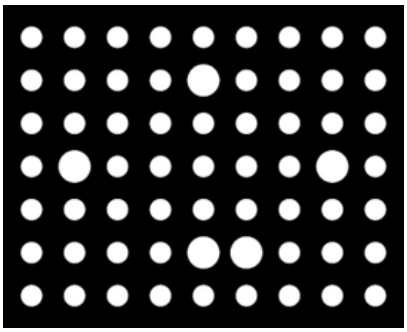
- 五行六列 _20mm：长 120mm，适用于相机视野较小的情况。



- 四行五列 _50mm：长 250mm，适用于相机视野较大的情况。



- 九行十一列 _15mm：该标定板为圆点图案，适用于 2D 相机内参标定。



此外，用户可根据需求自定义标定板，用户可设置标定板类型、行数、列数、格子边长以及二维码边长，系统可自动生成标定板。自定义标定板在配置 视觉界面时，在视觉流图中的 **加载标定信息**模块中配置。需要注意的是，用户自定义标定板是行数必须小于列数，二维码边长必须小于格子边长。

标定板的安装

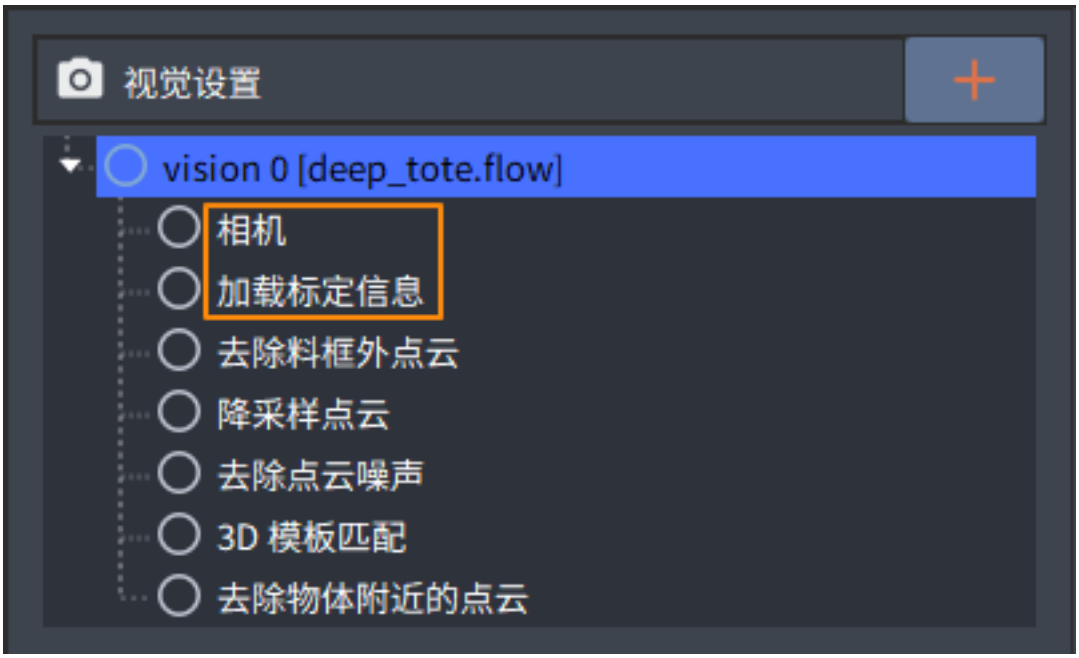
> **警告:** 安装标定板前请确保机械臂断电，以确保人员安全。

将标定板安装在机械臂的末端，将螺丝拧紧即可。

- 对于眼在手上的标定方式，请检查标定板是否在固定位置，确保标定过程中标定板不发生晃动。
- 对于眼在手外的标定方式，请检查标定板是否装紧固，确保标定过程中标定板和法兰的相对位置固定。

标定的流程

相机标定时，2D 相机需要先在 视觉界面的 **相机**模块中 相机参数区域，标定相机内参，具体方法请参见内参标定。然后在 **加载标定信息**模块中，标定相机外参，具体方法请参见手眼标定 和双目标定。



7.4.1 内参标定

内参标定的目的在于准确计算出相机内部固有的基本参数，该标定方式仅针对 2D 相机。其使用步骤如下。

1. 在 视觉界面中打开 **相机**模块，然后在界面最右边 相机参数设置区域中开始标定。

2. 在 *相机配置*区域，选择相机序列号和对应相机的配置文件。也可不选择配置文件，在 *相机参数*区域调整参数并单击 *另存为*，保存为新的配置文件使用。其他参数可根据用户需求选择，然后单击 *确认*，完成相机基本参数设置。
3. 单击 *验证相机精度*，开始标定。
    a. 在标定设置中，选择合适的标定板。2D 相机的内参标定，建议使用九行十一列的标定板以提高标定精确度。

b. 手动移动标定板至相机视野的 4 个角落和中心点，并分别单击 检测标定板得到验证数据，直到完成 5 个点的标定，进度条会实时显示标定进度。

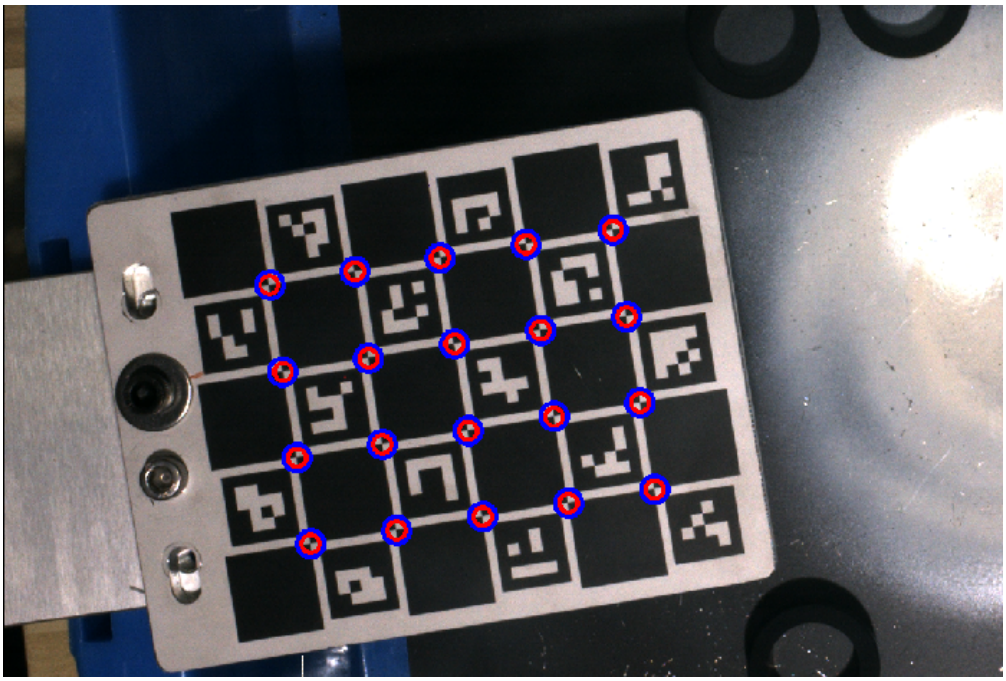
提示:

- 尽可能保证拍摄到的 5 张图片可以覆盖相机拍摄的整个画面。
- 确保每个位置都有足够的旋转角度的变化，以便不同位置的标定板都能被软件准确识别。
- 如果对标定结果不满意，可以单击 重置进度，重新开始内参数标定。

---

c. 在下方验证结果区域查看误差值，并根据界面提示检查标定误差。

d. 双击右下角图片，可以放大图片，查看大图中红点和蓝点的重合度，重合度越高标定越准确。如下图所示。



e. 单击右下角 确认，完成内参标定，Max 会自动将标定参数写入配置文件。

内参标定结束后，可根据需要选择手眼标定或者双目标定的方式，进行外参标定。

7.4.2 手眼标定

Max 提供的手眼标定方式有两种，一种是相机在机械臂上，另一种是相机固定在场景。

- 相机在机械臂上，即眼在手上（eye on hand）。在这种标定方式中，相机被固定于机械臂末端法兰上，相机与机械臂法兰的相对位置是固定的。
- 相机固定在场景，即眼在手外（eye to hand）。在这种标定方式中，相机被安装在机械臂外的一个固定位置，或者安装在滑轨上，可在若干个固定位置之间移动。

下面将详细介绍两种标定方法在 Max 中的使用步骤。

在 视觉界面中打开 **加载标定信息**模块，在界面最右边 *3D* 标定设置区域中 **相机标定**页签下，单击 新建标定数据开始标定，如下图。

1. 设置标定方法。

a. 设置 **应用场景**，根据实际情况选择 **眼在手上**或 **眼在手外**。

b. 设置 **标定算法**，根据实际情况可选择 **标准**、**触碰**或 **间接**。本章以标准标定算法为例。

c. 选择 **操作方式**，根据实际情况选择 **自动**或 **手动**，一般推荐自动标定的方式，方便快捷。

d. 根据实际情况设置机械臂轴数。若选择 4 轴机械臂则还需设置 **法兰盘 Z 轴偏置**，该值需手工测量，然后在界面中设置。

2. 标定设置。

a. 单击 选择，根据需要选择标定板类型，再单击 确认并返回即可。

b. 根据实际情况以及工作空间大小设置旋转角度，一般建议为 10~30 度。该参数仅自动标定需要设置。

c. 根据实际需求设置标定延迟时间，一般设置 1~3 秒即可。

d. 设置 **X/Y 方向移动距离**和 **Z 方向移动距离**，设置每次在各方向上的移动距离。

提示： **X/Y 方向移动距离**仅针对四轴机械臂，六轴机械臂不需要配置该参数。

---

e. 单击 注册，可实时读取当前机械臂末端位姿，完成位姿注册，初始位姿会在下面显示。也

可单击页签右下角的 *移动机械臂到初始位姿*，使机械臂回归到初始位姿。

提示： 注册前请先连接机械臂，否则无法读取机械臂位姿。

---

f. 单击 *生成标定位姿*开始自动标定。

3. 标定数据。

a. 在 **标定数据**页签中单击 *开始*，机械臂会根据设定旋转角、标定延迟等参数自动完成标定，并在进度条实时显示标定进度。

提示： 若选择手动的标定方式，则需要使用示教器将机械臂移到不同的位置，并手动输入相应点的位姿数据。每输入一组数据单击一次 *标定*，进度条会实时显示标定进度，直到完成所有数据的输入。

---

b. 若对标定过程不满意，可单击 *重置标定进度*，重新开始标定。当有效点达到 8 个时，即可计算标定结果。

c. 单击 *计算标定结果*，则显示相机位姿、角度、标定误差等信息。

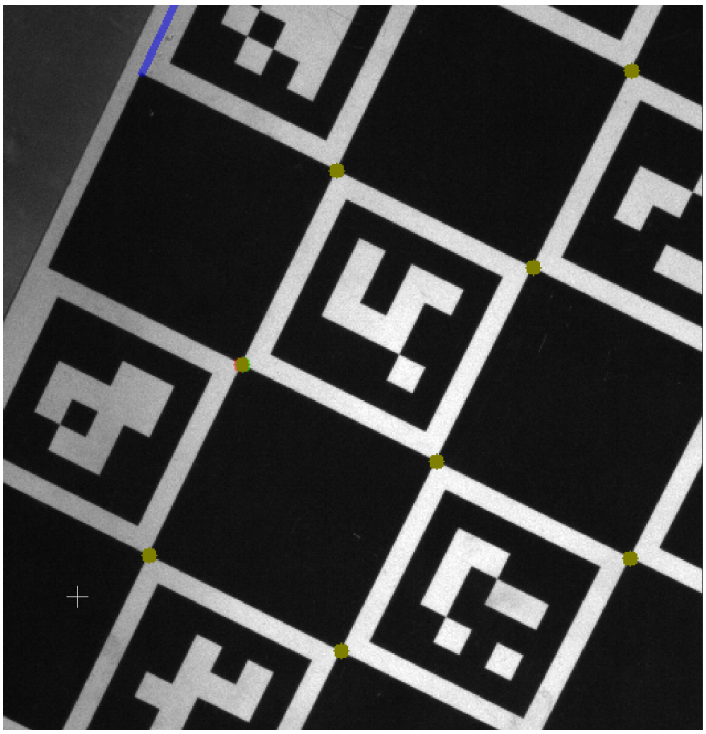
d. 在“标定数据”表格中会显示标定的点的误差，若误差较大可以单击 编辑，弹出 编辑标定数据窗口，更新位姿数据减小误差。然后单击 使用该位姿，窗口关闭，系统会重新计算并更新标定结果。

e. 单击 下一步开始检查标定结果。

4. 检查标定结果。

a. 在 **标定结果**页签中查看相机位姿、角度、标定误差等信息。同时单击问号图标，可查看标定结果的解释说明。

b. 单击 可视化结果，在彩色图中观察投影点重合度，重合度高则标定结果好。检查结果时，按【Ctrl】键滑动鼠标滚轮，可放大图片。



c. 在 **点云标定**页签中选择 **点云标定**，开始点云标定的步骤。若无需标定点云，则单击 完成即可结束相机标定。

提示:

- 在标定的图像结果中，红色点是视觉识别得到的点，绿色点是计算出来的理论值，两种点重合的部分显示为黄色。重合度越高、黄色部分越大表示标定结果越好。
- 在 3D 视图中，当坐标系原点在标定板顶点，X 轴和 Y 轴分别与该顶点两条邻边重合，且 Z 轴垂直于标定板平面向上时，标定结果较好。

---

5. 点云标定。


标定设置
操作方法
自动
手动
标定板
标定板名称 标定板图像
选择
注册初始机械臂位姿 未注册
注册
移动机械臂，使得标定板在相机视野的中央，并将此机械臂位姿作为自动标定的初始位姿。
XYZ
0.000
0.000
0.000
mm
RZYX
0.000
0.000
0.000
deg
欧拉角RZYX - ABB, ESTUN, KUKA, NACHI, TM
移动机械臂到初始位姿
X/Y方向移动距离
100毫米
Z方向移动距离
100毫米
拍照延迟
3秒
生成标定位姿
标定数据
采集标定数据
标定进度
27/27
检测成功:
27
检测失败:
0
开始
重置标定进度
标定结果
计算标定结果
0.535743 毫米
标定数据
序号 X,Y,Z EZ,EY,EX 误差 操作
1 1034.9,1173.1,277.2 83.2,19.0,-178.0 0.2780
2 1043.8,1248.2,251.2 83.2,19.0,-178.0 0.2457
3 1123.0,1237.7,248.4 83.2,19.0,-178.0 0.3807
4 1114.1,1162.6,274.6 83.2,19.0,-178.0 0.4216
5 1105.1,1087.5,300.5 83.2,19.0,-178.0 0.4318
6 1025.7,1097.9,303.1 83.2,19.0,-178.0 0.3986
7 946.5,1108.2,305.8 83.2,19.0,-178.0 0.3996
8 955.4,1183.4,279.8 83.2,19.0,-178.0 0.1677
9 964.4,1258.5,253.8 83.2,19.0,-178.0 0.2296
10 1030.4,1153.9,220.5 83.2,19.0,-178.0 0.2096
11 1039.3,1229.1,194.5 83.2,19.0,-178.0 0.2294
12 1118.6,1218.7,191.7 83.2,19.0,-178.0 0.4582
13 1109.5,1143.6,217.7 83.2,19.0,-178.0 0.2906
14 1100.6,1068.4,243.9 83.2,19.0,-178.0 0.3132
上一步
下一步

a. 点云标定的参数设置及方法请参照前文步骤 2~4。进行点云标定时，根据实际情况设置 *X/Y 方向移动距离*和 *Z 方向移动距离*，其余步骤和前面相同。

提示： 点云标定用于校准相机深度（z）方向上的精度，在能自动标定的情况下，一定要选择点云标定。

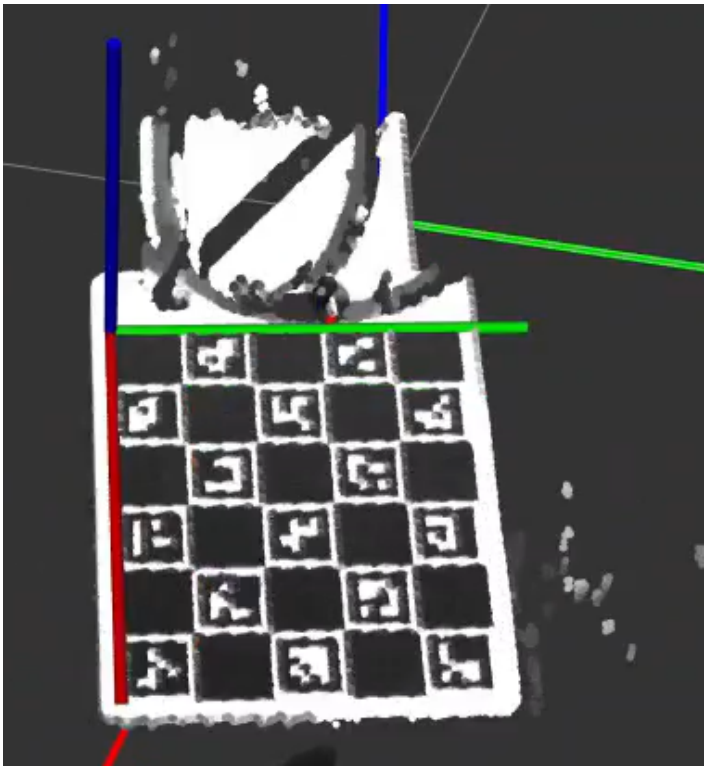
---

b. 单击 下一步，检查点云标定结果。

6. 检查点云标定结果。

a. 在 **标定结果**页签中查看相机位姿、角度、标定误差等信息。

b. 单击 可视化结果，可在彩色图中观察投影点重合度，重合度越高则标定结果越好。



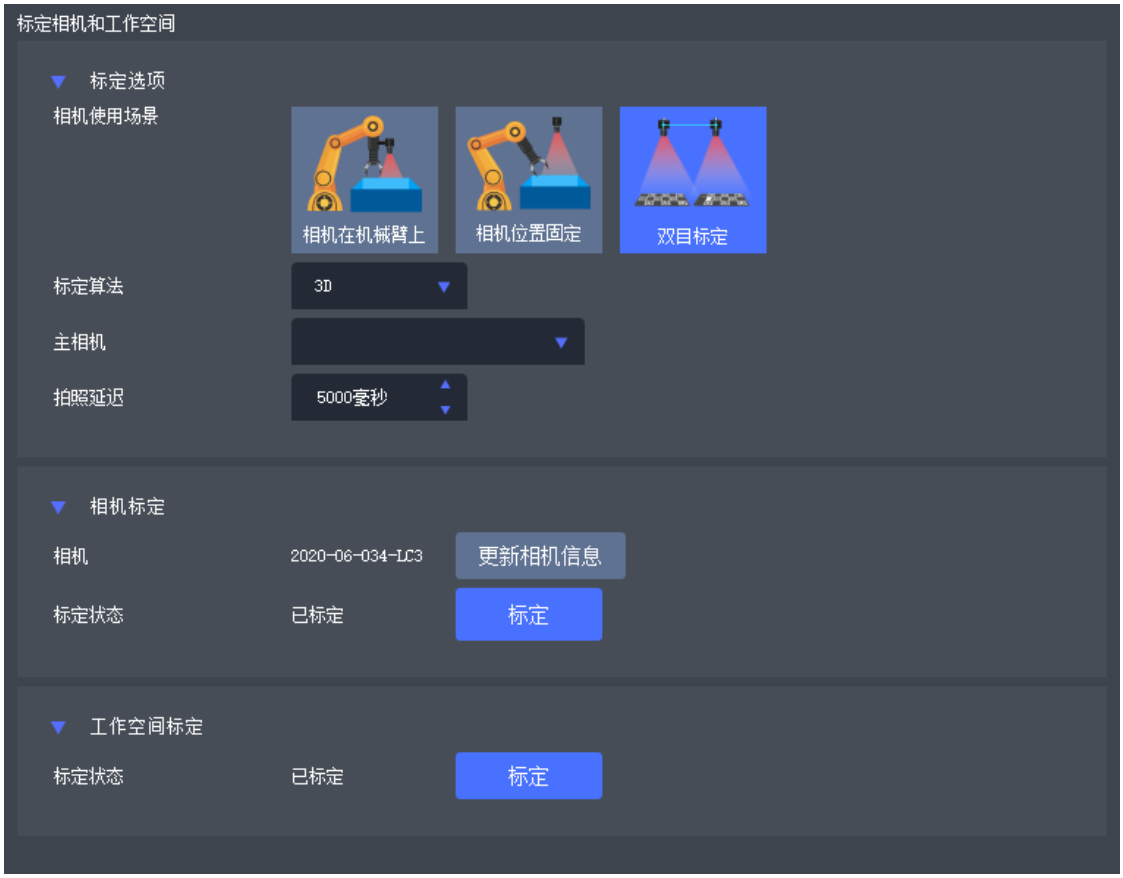
c. 单击 完成结束相机标定。

手眼标定完成后还需要进行工作空间标定，请参见工作空间标定。

7.4.3 双目标定

双目标定获得两台相机之间的相互位置关系的标定。

在 视觉界面中打开 **加载标定信息**模块，在界面最右边 标定相机和工作空间区域中完成标定，如下图。



重要:

- 双目标定时，请保证相机布局合理，尽量覆盖整个工件的边缘位置，相机距离在 500~1000 mm 处。
- 安装相机时同时要考虑光源的选择和布置，保证视野内的光线稳定，物体边缘和特征清晰。
- 双目标定前，需要先针对 2D 相机进行内参标定。

双目标定步骤如下。

1. 标定前先确定主相机，然后使用手眼标定的方式，标定主相机外参，标定方法请参见手眼标定。
2. 标定算法选择 双目标定，用来标定主从相机的相对关系。
3. 在 已标定相机设置中选择主相机，根据用户需求设置相机 ID、标定数据、拍照延迟时间、标定板。
4. 在 待标定相机设置中选择从相机，根据用户需求设置标定板、拍照延迟时间。

提示： 2D 相机的双目标定建议使用九行十一列图案为圆形的标定板。

5. 手动移动标定板位置，每移动一个位置后在 *标定数据采集区域*依次单击 *获取机械臂位姿*和 *采集数据*，直到完成所有位置的标定，进度条会实时显示标定进度。

   若图像效果不理想，单击 *重置进度*重新开始标定。

6. 单击 *计算标定结果*，显示相机位姿、角度、标定误差等信息。
7. 在 *工作空间标定*区域中单击 *标定*，开始标定工作空间。

双目标定完成后还需要进行工作空间标定，请参见工作空间标定。

7.4.4 工作空间标定

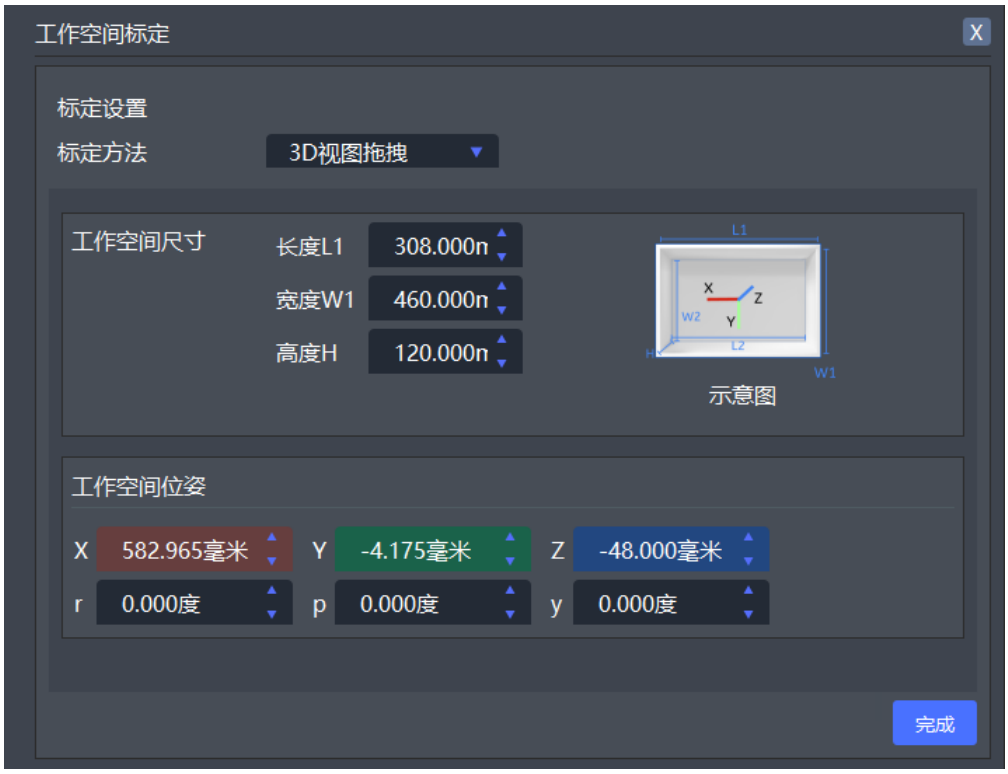
相机标定完成后，还需要标定视觉工作空间。打开 **加载标定信息**模块后，在界面最右边 *3D 标定设置*区域中**工作空间标定**页签下，单击 *新建标定数据*开始标定，如下图。

工作空间的标定有三种方式：3D 视图拖拽、角落点触碰式和边缘点触碰式。

3D 拖拽

3D 视图拖拽是指在预览视图当中，将料筐模型拖拽至与料筐点云重合的位置，从而标定工作空间。该方式适用于 3D 相机。

1. 在 *工作空间尺寸*区域中设置工作空间的长、宽、高，用户需要测量工作空间实际的长宽高数据，然后输入真实的工作空间尺寸。



2. 在预览窗口移动工作空间，使之与点云中的料筐吻合。
3. 依次单击 完成，即可完成标定。

角落点触碰式

该方式使用探针触碰工作空间（如料筐）上表面的三个角点，通过这三个点的位置标定整个工作空间。该方式适用于 2D 相机。

1. 设置法兰坐标系下的工具位姿，工具（即探针）和法兰位置保持相对固定。
2. 在 工作空间尺寸区域中设置工作空间的长、宽、高，用户需要测量工作空间实际的长宽高数据，然后输入真实的工作空间尺寸。
3. 移动机械臂，使探针的尖端分别触碰工作空间的三个角点，单击 标定分别得到三个点的位置数据。
4. 系统计算得到工作空间位姿，单击 完成，即可完成标定。

边缘点触碰式

该方式使用探针触碰工作空间（如料筐）上表面的五个点，其中工作空间的三条边上各取一个点，另外一条边上取两个点，通过这五个点的位置标定整个工作空间。该方式适用于 2D 相机。


工作空间标定
X
标定设置
标定方法
边缘点触碰式标定
法兰坐标系下的工具位姿(TCP)
X 0.000毫米
Y 0.000毫米
Z 0.000毫米
EZ 0.000度
EY 0.000度
EX 0.000度
工作空间尺寸
长度 308.000毫米
宽度 460.000毫米
高度 119.000毫米
示意图
点位1 未标定 标定
点位2 未标定 标定
点位3 未标定 标定
点位4 未标定 标定
点位5 未标定 标定
重置进度
计算标定结果
工作空间位姿
X 582.965毫米
Y -4.175毫米
Z -48.000毫米
r 0.000度
p 0.000度
y 0.000度
完成

1. 设置法兰坐标系下的工具位姿，工具（即探针）和法兰位置保持相对固定。
2. 在 工作空间尺寸区域中设置工作空间的长、宽、高，用户需要测量工作空间实际的长宽高数据，然后输入真实的工作空间尺寸。
3. 移动机械臂，使探针的尖端分别触碰工作空间的五个点，三条边上各取一个点，另一条边上取两个点，单击 标定分别得到五个点的位置数据。
4. 系统计算得到工作空间位姿，单击 完成，即可完成标定。

7.5 服务

调用服务就是运行流图中一串指定的模块。在 视觉界面右下角 服务编辑页签下，可查看并配置服务。服务名称中的编号与工作空间一一对应，如 vision_3 对应工作空间 3。

7.5.1 开启服务

不同模板流图预设不同服务指令。常见指令如下：

- capture_images：拍照获取图像。
- load_calibration_info：加载标定信息。
- get_safe_height：获得安全高度。
- calculate_object_poses：计算物体位姿。
- calculate_object_dimension：计算物体尺寸。

在 服务编辑页签下，取消选择 开启，单击不同服务命令，可查看各命令对应的起始和结束节点，单击 运行选中命令可运行对应的模块。勾选指令后，选择 开启，开启对应服务。

7.5.2 添加命令

用户也可在软件预设的命令之外添加命令，操作如下：

1. 取消选择 开启，在 命令栏输入命名，单击 添加。
2. 单击新添加的命令，在节点视图中单击要操作的起始和结束模块，右键选择 起始模块和 结束模块，分别定义起始和结束节点。

CHAPTER 8

基本运动操作

本章介绍 运动界面的基本操作，包括添加机械臂、添加工具、注册抓取方式等。

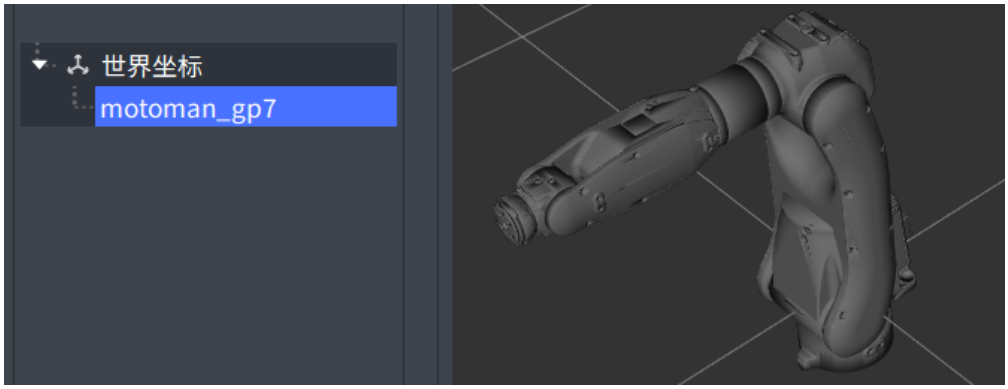
8.1 添加机械臂

如在新建项目时未添加机械臂模型，可在新建项目后，在 运动页签下添加模型。

1. 在 运动页签下快捷工具栏，单击 机械臂。
2. 选择机械臂类型并完成添加。
    - 常规机械臂：选用 Max 内置的机械臂型模型。在左侧搜索栏搜索或根据品牌、轴数、臂展等条件筛选机械臂型号，单击待添加的机械臂型号，再单击右下角 确定。

    - 导入机械臂：从本地导入机械臂 urdf 模型。
    - 自定义机械臂：自定义添加机械臂时，需设置机械臂的轴数和欧拉角类型。

3. 在中间预览窗口查看机械臂，并根据需要在右侧参数设置栏调整机械臂位姿。



4. （可选）在右侧参数设置栏 **软限位设置**页签下，修改关节限位，限制机械臂的运动范围，从而提高求解路径合理性，保护设备和人员安全。

8.2 添加工作空间

根据实际场景，添加料筐、托盘、输送线等工作空间。2D/3D 坐标移动和 3D 轨迹移动场景一般添加料筐，拆码垛场景一般添加托盘、输送线。

8.2.1 添加托盘

1. 在 *运动*页签下快捷工具栏，单击 *工作空间*，选择 **托盘**。
2. 单击界面左下角 **托盘**节点，在右侧参数设置栏设置托盘的尺寸和位姿。
   - 使用视觉标定结果：从视觉标定结果获取托盘位姿和尺寸。在使用该功能前，需先在 *视觉*页签下完成视觉标定，请参见标定。
     a. 在 **使用视觉标定结果**右侧，选择 **是**。
     b. 选择视觉服务编号。
     c. 单击 *获取标定结果*，选择获取视觉位姿或尺寸。

   - 手动设置托盘尺寸和位姿：在 **托盘尺寸**下方设置托盘的长度、宽度、高度，尺寸与技术协议规定尺寸保持一致，在 **底面内侧中心点位姿**下方设置托盘位姿。
3. 在右侧 **工作空间设置**下方，设置工作空间的编号、尺寸、位姿。
   - 单击 *所有工作空间与托盘同步*，可选择同步托盘的尺寸或位姿。

- 单击 **操作**列下方  ，在 **工作空间**下方设置工作空间的尺寸及位姿，可选择使用视觉标定结果，或者手动设置，工作空间的长宽一般与托盘长宽相同，高度取决于垛型的高度。在 **工作空间资源**下方可添加工作空间资源，本节以在托盘添加 SKU 为例，操作如下：

a. 单击 *添加坐标轴*，添加工作空间资源的坐标轴。

b. 在 *任务*页签下，新建任务流图，添加 **生成 SKU** 与 **模拟视觉结果**模块，连接两个模块的 SKU 信息端口。

c. 单击 **生成 SKU** 模块，在右侧属性栏选择 **使用常数**，设置 SKU 的长宽高等信息。

d. 单击 **模拟视觉结果**模块，在右侧属性栏选择 **手动输入工作空间编号**，输入托盘工作空间的编号。

e. 依次运行两个模块后，回到 *运动*页签，单击 *同步（已启动）*，退出同步模式，单击右下角 **工作空间**节点，可查看生成的工作空间资源。

8.2.2 添加料筐

1. 在 运动 页签下快捷工具栏，单击 工作空间，选择 **添加料筐**。
2. 单击界面左下角 **料筐**节点，在右侧参数设置栏设置料筐的尺寸和位姿。
   - 使用视觉标定结果：从视觉标定结果获取料筐位姿和尺寸。在使用该功能前，需先在 视觉 页签下完成视觉标定，请参见标定。
     a. 在 **使用视觉标定结果**右侧，选择 **是**。
     b. 选择视觉服务编号。
     c. 单击 获取标定结果，选择获取视觉位姿或尺寸。

   - 手动设置料筐尺寸和位姿：在 **料筐尺寸**下方设置料筐的长度、宽度、高度、厚度，在 **底面内侧中心点位姿**下方设置料筐位姿。如果是异形料筐，则单击 **异形料筐**右侧  ，设置料筐尺寸，并选择有效的面。

3. 在右侧 **工作空间设置**下方，设置工作空间的编号、尺寸、位姿。
   - 单击 *所有工作空间与料筐同步*，可选择同步料筐的尺寸或位姿。

   - 单击 **操作**列下方  ，可设置工作空间的尺寸及位姿，并添加工作空间资源。设置工作空间的尺寸及位姿时，可选择使用视觉标定结果，或者手动设置，操作与料筐的尺寸、位姿设置类似。

8.2.3 添加输送线

1. 在 运动 页签下快捷工具栏，单击 工作空间，选择 **输送线**。
2. 单击界面左下角 **输送线**节点，在右侧参数设置栏设置输送线的尺寸和位姿。
   - 使用视觉标定结果：从视觉标定结果获取输送线位姿和尺寸。在使用该功能前，需先在 视觉 页签下完成视觉标定，请参见标定。
     a. 在 **使用视觉标定结果**右侧，选择 **是**。
     b. 选择视觉服务编号。
     c. 单击 获取标定结果，选择获取视觉位姿或尺寸。

   - 手动设置输送线尺寸和位姿：在 **输送线尺寸**下方设置输送线的长度、宽度、高度、挡板尺寸，在 **底面内侧中心点位姿**下方设置输送线位姿。长宽可以根据实际情况设置，高度建议设置为输送线距离机械臂的高度。
3. 在右侧 **工作空间设置**下方，设置工作空间的编号、尺寸、位姿。
   - 单击 所有工作空间与输送线同步，可选择同步输送线的尺寸或位姿。

   - 单击 **操作**列下方  ，可设置工作空间的尺寸及位姿，并添加工作空间资源。设置工作空间的尺寸及位姿时，可选择使用视觉标定结果，或者手动设置。工作空间的长宽一般与输送线长宽相

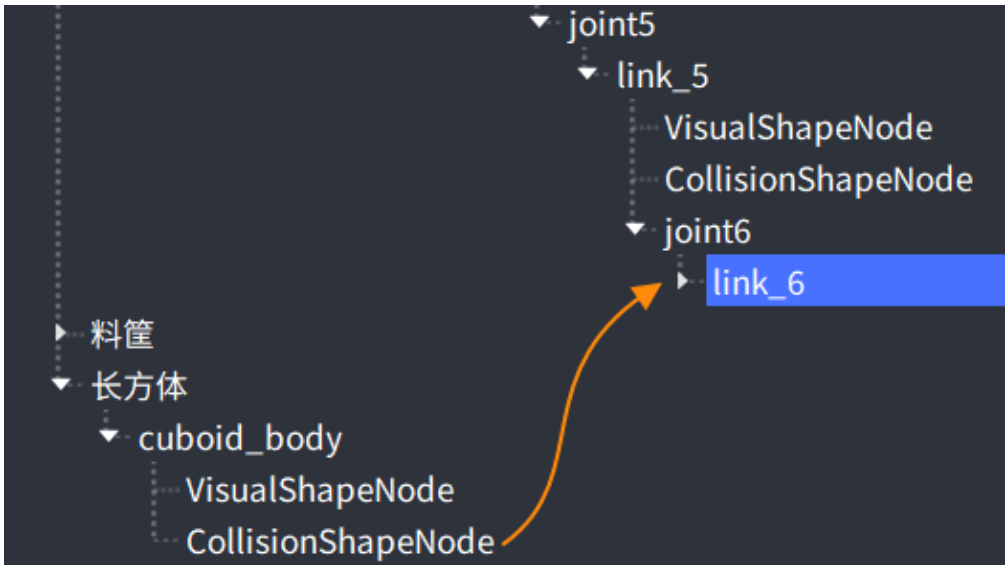
同，高度取决于垛型的高度，对于拆垛来说，高于物料高度即可。

8.3 添加几何体

根据实际场景，添加几何体，用于代表环境中的相机、障碍物等物体。碰撞体位姿、大小需要准确，且适量膨胀，以便碰撞检测。本节以添加长方体为例。

1. 在 *运动*页签下快捷工具栏，单击 *几何体*，选择 **添加长方体**。
2. 单击界面左下角 **长方体**节点，根据实际环境测量结果，在右侧参数设置栏设置长宽高、位姿等信息。

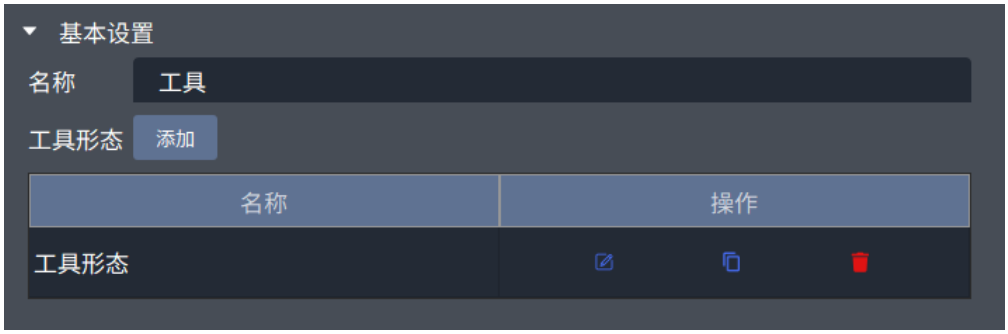
3. （可选）如需用几何体表示安装在机械臂上的相机，可右键单击左下角 **长方体**节点，选择 **展示细节**，将 **CollisionShapeNode** 拖放至机械臂模型的末轴节点之下，比如六轴机械臂就拖放至 link_6 之下。



8.4 注册末端工具

8.4.1 注册工具

1. 在 运动 页签下快捷工具栏，单击 末端工具，选择 **注册工具**。
2. 单击界面左下角 **工具** 节点，在右侧参数设置栏单击 添加，添加工具形态。



3. 在 工具形态 页签下，单击 **碰撞模型** 下方 添加，选择 **模型文件**，单击  ，选择*STL* 模型路径。

提示:

- 如果将 **可视化和碰撞模型一致** 设置为 “否”，则需分别设置碰撞模型和可视化模型。
- 除了导入 STL 模型，也可添加长方体、圆柱体或球体模型。

---

4. 查看模型基本信息，为不影响后续计算，如果模型三角面数大于 1000，建议先使用模型编辑器，将三角面数减至 1000 以下。

   a. 单击  ，确认模型信息，单击 确认。

   提示: 也可在顶部工具栏，选择 “工具 > 模型编辑器”，打开模型编辑器后导入模型。

   ---

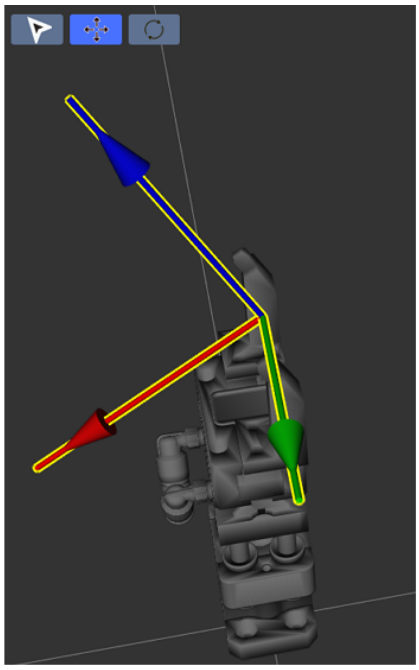
   b. 在 **Max 模型编辑器** 窗口顶部工具栏，选择 “编辑 > 模型简化”，将简化方式设置为 **减少三角面**，输入目标面数后，单击 确认，三角面数会减少至设置的数量。

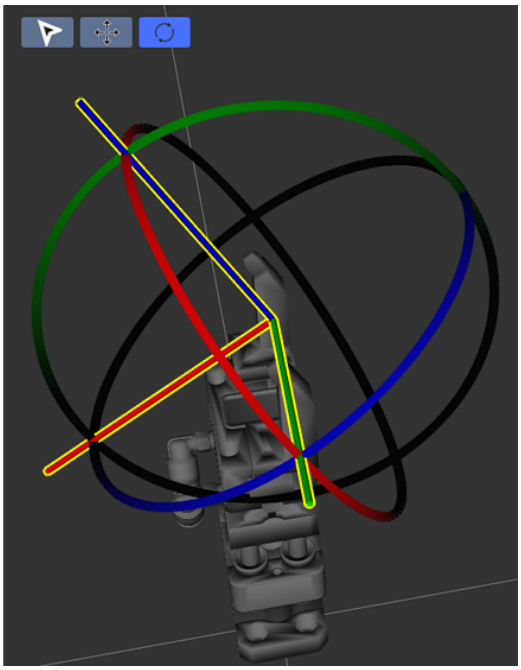
   c. 在 **Max 模型编辑器** 窗口顶部工具栏，选择 “文件 > 保存”。回到 工具形态 页签下，重新导入简化后的模型。

5. 在 工具形态页签下，设置工具参考点。单击 **工具参考点**下方 添加，在中间的预览窗口，将工具参考点拖放到工具尖端，单击右侧参数设置栏右下角 完成。

小技巧: 建议将工具参考点放置在工具两个尖端中间，Z 轴方向与尖端朝向平行。

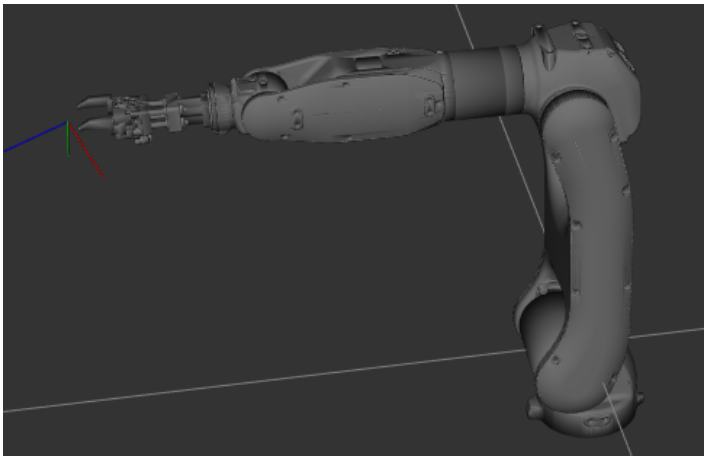
- 单击  ，按住 X、Y、Z 轴三个方向的箭头移动位置。
- 单击  ，按住圆环旋转角度。
- 单击  ，退出移动或旋转模式。在移动或旋转模式下，无法更改右侧参数设置栏信息。





6. 如果工具有多种形态，比如张开与闭合，可单击界面左下角 **工具**节点，在右侧参数设置栏单击 添加，重复以上步骤添加更多工具形态。

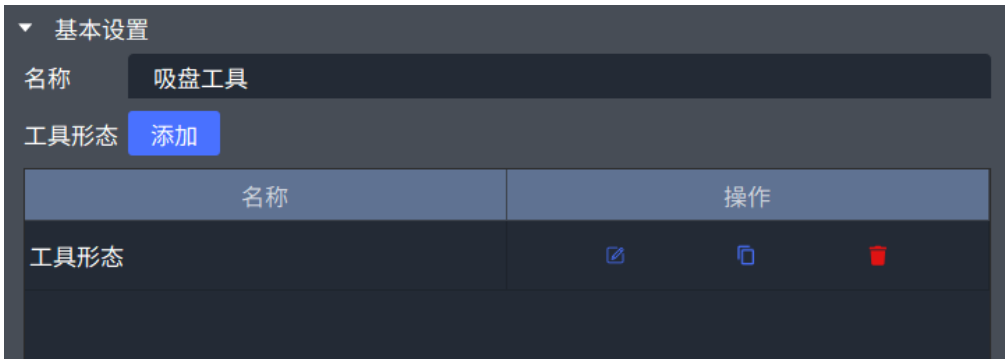
7. 单击中间窗口上方的 环境页签，将左侧环境编辑面板中的工具文件拖放到左下方机械臂模型节点下。



8.4.2 注册吸具

注册吸具，一般用于拆码垛场景。

1. 在 运动页签下快捷工具栏，单击 末端工具，选择 **注册吸具**。
2. 单击界面左下角 **吸盘工具**节点，在右侧参数设置栏单击 添加，添加工具形态。



3. 注册工具模型。在 工具形态页签下，单击 **碰撞模型**下方 添加，选择 **模型文件**，添加 STL 模型，或者选择 **长方体**、**圆柱体**或 **球体**，添加相应形状的碰撞体，然后单击 下一步。

提示:

- 如果将 **可视化和碰撞模型一致**设置为 "否"，则需分别设置碰撞模型和可视化模型。
- 导入 STL 模型时，为不影响后续计算，如果模型三角面数大于 1000，建议先使用模型编辑器，将三角面数减至 1000 以下。

4. 注册吸取单元。吸取单元指物理层面的吸取单元，即实际 IO 能够控制的最小单元。
    a. 设置吸盘基准面的尺寸与位姿。
    b. 设置吸盘物理属性，包括原始高度、吸取后高度、单位面积吸取重量。

c. 设置吸取单元。启用 **视角固定 Z 轴正视图**，使中间预览窗口的视角固定在 Z 轴正视图，设置 I/O 来源以及 X 和 Y 方向上吸取单元数量，单击 *自动生成吸取单元*。可在参数栏 **吸取单元详情**下方查看生成结果。

d. 单击每个吸取单元左侧  ，设置对应吸取单元的 I/O。I/O 端口号需与实际机械臂设置的 IO 端口保持一致，否则会造成运算与实际不符。

- 抓取 I/O：单击 **抓取 I/O** 下方 *添加 IO 端口*，设置 I/O 端口号和放置值。
- 放置 I/O：单击 **放置 I/O** 下方 *添加 IO 端口*，设置 I/O 端口号和放置值。
- 掉箱检测 I/O：启用掉箱检测后，单击 *添加 IO 端口*，设置 IO 端口号、未掉箱时的值。

提示:

- 设置掉箱检测 I/O，如果考虑传感器具体位置，则需设置传感器点位和直径。
- 如有多个 IO 端口，需设置使用逻辑，可选择 **和**或 **或**。

e. 单击 下一步。

5. 注册吸取组合。单击 生成吸取组合，创建吸取组合，单击界面右下角 完成。

8.5 注册工件

8.5.1 注册待抓工件

有工件模型

如果已有工件的 STL 模型，可导入模型以注册工件。

1. 在 运动页签下快捷工具栏，单击 工件，选择 **注册工件**。
2. 单击界面左下角 **工件**节点，单击右侧参数设置栏 **碰撞模型**下方 添加，选择 **模型文件**，单击  ，选择 STL 模型路径。

提示： 如果将 **可视化和碰撞模型一致**设置为“否”，则需分别设置碰撞模型和可视化模型。

3. 查看模型基本信息，为不影响后续计算，如果模型三角面数大于 1000，建议先使用模型编辑器，将三角面数减至 1000 以下。

   a. 单击  ，确认模型信息，单击 *OK*。

   提示： 也可在顶部工具栏，选择“工具 > 模型编辑器”，打开模型编辑器后导入模型。

   b. 在 **Max 模型编辑器**窗口顶部工具栏，选择“编辑 > 模型简化”，选择“减少三角面”，输入三角面数量后，单击 确认，三角面数会减少至设置的数量。

   c. 在 **Max 模型编辑器**窗口顶部工具栏，选择“文件 > 保存”。回到 工件页签下，导入简化后的模型。

4. （可选）单击右侧参数设置栏 **碰撞模型**下方 添加，选择 **长方体**、**圆柱体**、**球体**，在 STL 模型基础上添加碰撞体。添加后，设置每个碰撞体的尺寸和位姿。

5. 单击界面右下角 完成。

无工件模型

如无工件的 STL 模型，可用碰撞体拼接工件。

1. 在 运动页签下快捷工具栏，单击 工件，选择 **注册工件**。

2. 单击界面左下角 **工件**节点，单击右侧参数设置栏 **碰撞模型**下方 添加，选择 **长方体**、**圆柱体**、**球体**，添加一个或多个碰撞体。

3. 设置每个碰撞体的尺寸和位姿。

提示： 如果将 **可视化和碰撞模型一致**设置为“否”，则需分别设置碰撞模型和可视化模型。

4. 单击界面右下角 完成。

8.5.2 注册箱子

注册箱子模型，一般用于料箱拆码垛场景。

1. 在 运动页签下快捷工具栏，单击 工件，选择 **注册箱子**。
2. 单击界面左下角 **箱子**节点，在右侧参数设置栏设置箱子的尺寸、条码位置等参数。
3. 单击界面右下角 完成。

8.6 注册抓取方式

根据抓取物体的类型，注册抓取方式。

8.6.1 注册抓工件方式

注册抓取工件的方式。

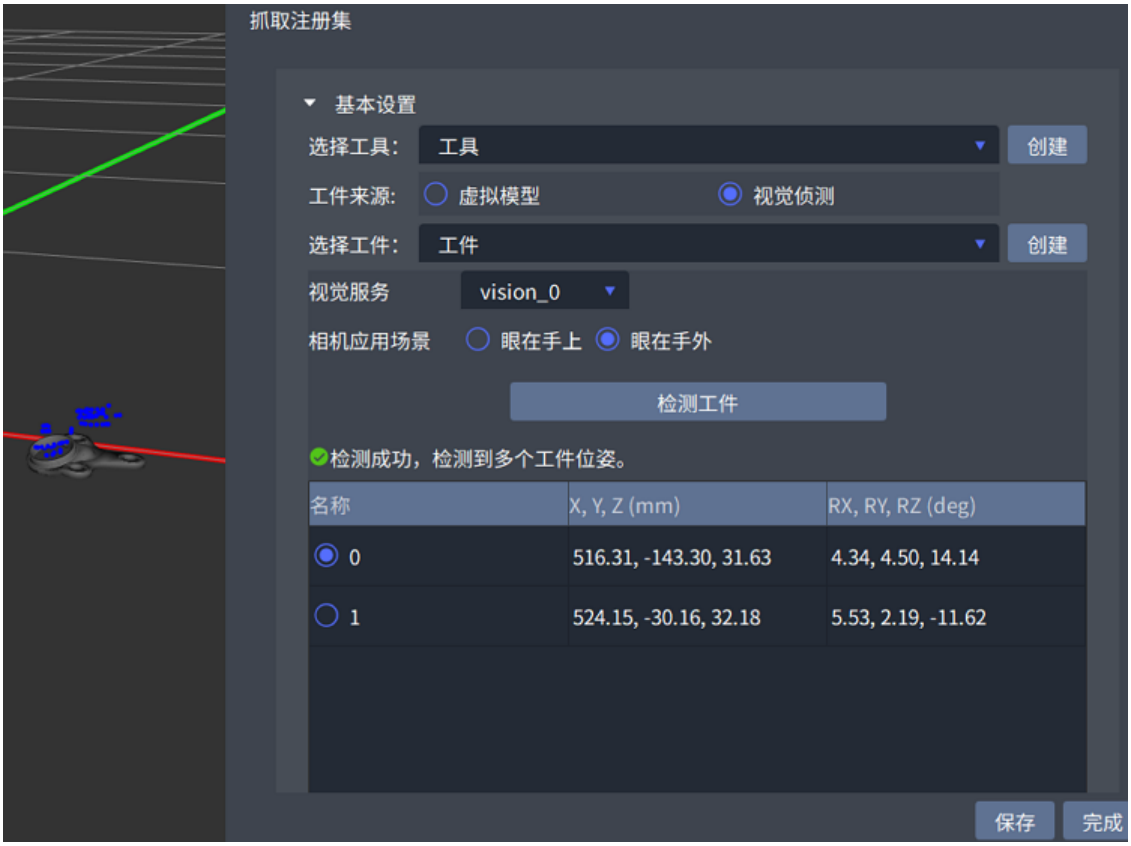
前提条件

在注册工件抓取方式前，需先注册工具和工件，操作参见注册末端工具 和注册工件。

操作步骤

1. 在 运动 页签下快捷工具栏，单击 抓取和放置，选择“注册抓取方式 > 工件抓取注册”。
2. 单击左下角 **抓取注册集**节点，在右侧参数设置栏为该抓取方式绑定工具、工件等。
   a. 从 **选择工具**右侧下拉列表中选择已注册的工具。

   b. 选择工件来源。
      - 虚拟模型：选择已注册的工件模型。
      - 视觉侦测：从视觉检测结果中选择较为稳定的工件位姿用于注册。如勾选此项，需选择工件、视觉服务、相机使用场景，然后单击 检测工件，选择检测到的工件位姿。



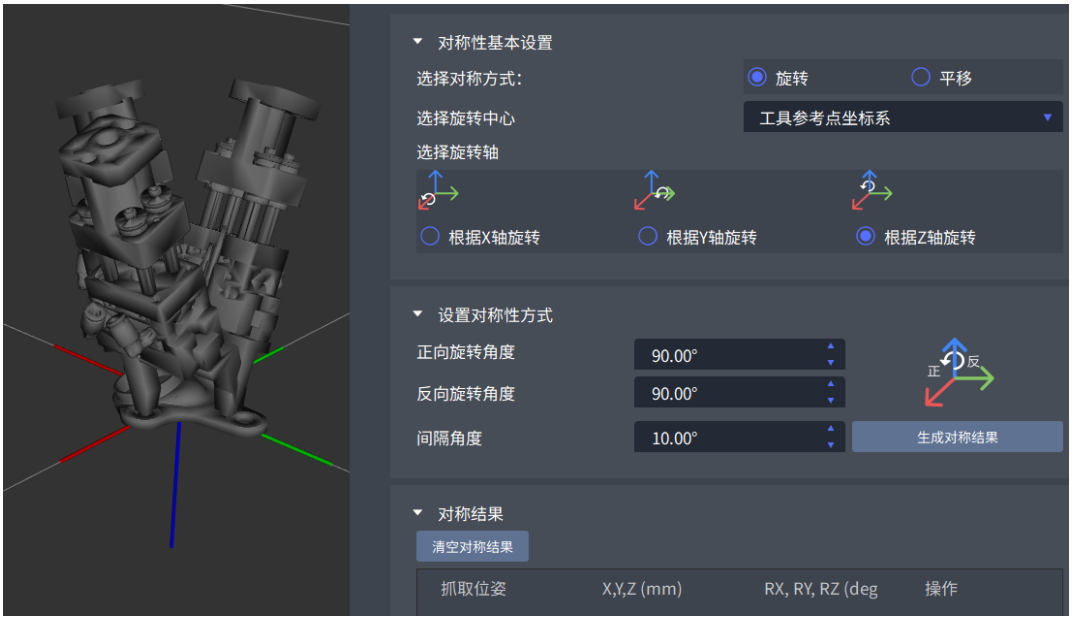
c. 单击 创建抓取注册。

3. 在 **注册抓取**下方选择工具形态，在 **注册位姿**下方设置抓取位姿。

   - 如果工件来源为虚拟模型，创建注册抓取后，选择控制轴，一般设为工具参考点即可，再单击 对齐控制轴与工件坐标，使工具参考点和工件的坐标系对齐，然后调整工具位置至合适的抓取位置和角度。

   - 如果工件来源为视觉侦测，则注册方式可选择 **现实示教**，移动真实机械臂至抓取位姿，输入机械臂位姿。

4. 单击 **对称性设置**下方 新增，通过旋转或平移变换遍历生成抓取工件所需的合理位姿。
   - 旋转遍历性设置
     a. 将对称方式设置为旋转。
     b. 选择旋转中心，默认为工具参考点坐标系。
     c. 选择旋转轴，可选择 X 轴、Y 轴或 Z 轴。
     d. 设置正向旋转角度、反向旋转角度、间隔角度。在中间的预览窗口可预览生成效果。
     e. 单击 生成对称结果。在 **对称结果**下方查看生成结果。
     f. 单击界面右下角 完成。



   - 平移遍历性设置
     a. 将遍历方式设置为平移。
     b. 选择平移中心，默认为工具参考点坐标系。
     c. 选择平移轴，可选择 X 轴、Y 轴或 Z 轴。

d. 设置正向平移角度、反向平移角度、间隔角度。在中间的预览窗口可预览生成效果。

e. 单击 生成对称结果。在 **对称结果**下方查看生成结果。

f. 单击界面右下角 完成。

5. 单击界面右下角 完成。

完成抓取方式注册后，还需将抓取方式添加到抓取处理流程中，以便任务规划时读取，请参见注册抓放规划。

8.6.2 注册抓箱子方式

注册抓取箱子的方式，一般用于拆码垛场景。

前提条件

在注册箱子抓取方式前，需先注册吸具，操作参见注册吸具。

操作步骤

1. 在 运动页签下快捷工具栏，单击 抓取和放置，选择 “注册抓取方式 > 箱子抓取注册”。
2. 单击左下角 **纸箱抓取注册**节点，在右侧参数设置栏选择吸取方法。
   - 单抓：一次抓取一个箱子。
   - 多抓（同一箱型）：在箱型一致的情况下，一次抓取多个箱子。
3. 如果选择多抓（同一箱型），需设置多抓合并参数。如使用其他抓取方法，则可跳过此步骤。
   - 允许长边合并：是否允许两个物体长边合并，即沿着物体 y 方向合并。
   - 允许短边合并：是否允许两个物体短边合并，即沿着物体 x 方向合并。
   - 最大合并数：多抓时最大合并抓取个数。
   - 高级参数：启用 **高级参数**后，可配置以下参数。
     - 合并最大允许高度差：两个箱子的高度差小于等于此阈值，才能合并。
     - 和理想合并位置最小重叠比例：合并位置与理想位置重叠比例不低于该值，才能合并。

- 计算时限：允许计算合并的最大时间。
- 正方形判断阈值：一般使用默认值即可。如果箱子的长宽差小于该阈值，则认为箱子表面为正方形。

4. 设置抓取规则。
   - 选择工具：选择已注册的吸盘工具。
   - 工具形态：选择已添加的工具形态。
   - 请选择需要使用的吸取组合：从吸取组合列表中选择需使用的组合。
   - 高级参数：启用 **高级参数**后，可配置以下参数。
     - 吸取方向：默认为 **吸顶**。
     - 允许不连续的吸取单元：选择是否允许不连续的吸取单元。
     - 最小吸取面积/物体表面积比例：吸盘与箱子重合的面积占箱子面积的比例大于此值，才能吸取成功。
     - 最小吸取面积/工具参考面积比例：吸盘与箱子重合的面积占吸盘吸取面表面积的比例大于此值，才能够吸取成功。
     - 单个吸取组合最大返回个数：单个吸取组合最多返回的吸取个数。
     - 计算时限：允许计算吸取姿态的最大时间。

5. 在 **测试参数配置**页签下，设置测试参数。选择测试方法，支持 **真实运行**和 **仿真运行**，然后配置其他参数。

   真实运行

   a. 选择是否为相同箱型。

   b. 选择视觉服务编号、相机应用场景。如果相机应用场景是眼在手上，需设置拍照位姿。

   c. 选择是否启用视觉检测工件尺寸。**相同箱型**选择 **是**后，该功能自动启用，且不可关闭。

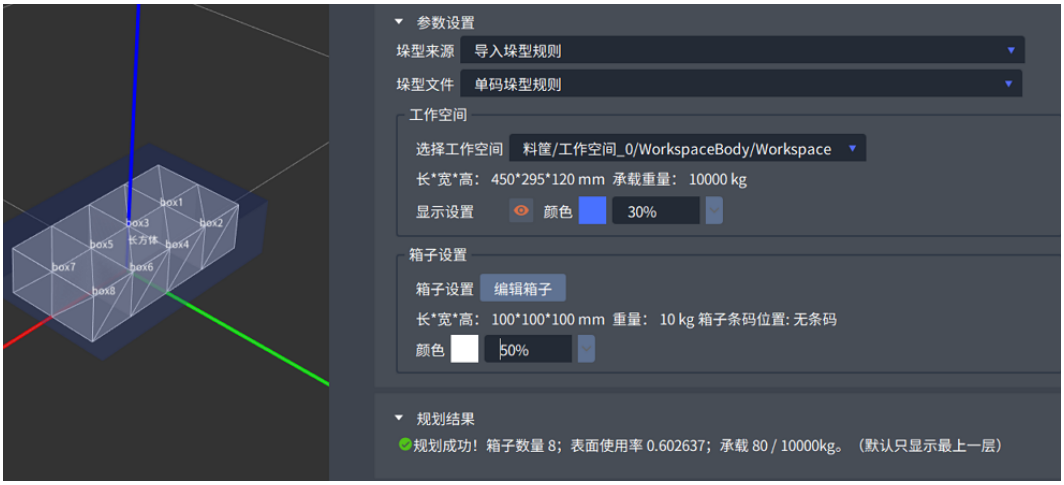
   d. 选择箱子条码位置。

   e. 单击 检测工件。

仿真运行

- 选择垛型来源为 **从环境导入垛型**，设置工作空间。
- 选择垛型来源为 **手动增加垛型**，单击 *新增箱子*，设置箱子尺寸、位姿、条码位置等参数，单击 *完成*。可重复此步骤，添加更多箱子，或单击 **操作**列的  ，复制生成新的箱子。

- 选择垛型来源为 **导入垛型规则**，选择已注册的垛型文件（请参见*注册放置方式*），选择工作空间，单击 *编辑箱子*，设置箱子尺寸和条码位置，查看规划结果。



6. 在 **运行结果**页签下，单击 *测试运行*，查看运行结果。确认无误后，单击右下角 *完成*。

完成抓取方式注册后，还需将抓取方式添加到抓取处理流程中，以便任务规划时读取，请参见*注册抓放规划*。

8.6.3 注册抓圆柱方式

注册抓取圆柱的方式。

前提条件

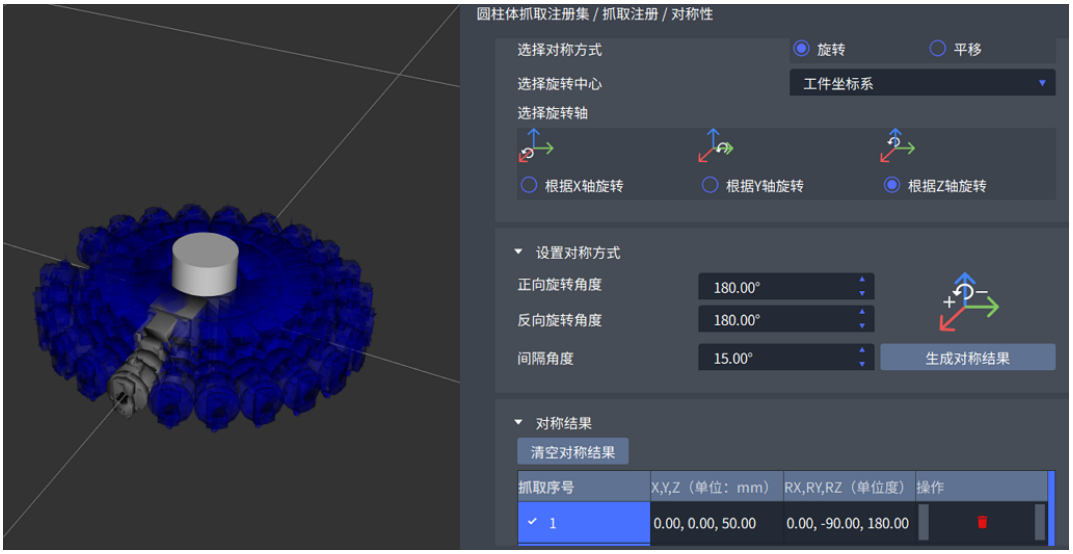
在注册工件抓取方式前，需先注册工具，操作参见*注册末端工具*。

操作步骤

1. 在 *运动*页签下快捷工具栏，单击 *抓取和放置*，选择“注册抓取方式 > 圆柱抓取注册”。
2. 单击左下角 **圆柱体抓取注册集**节点，在右侧参数设置栏 **圆柱体设置**下方设置圆柱体的长宽高，在 **工具设置**下方选择工具、工具形态、工具参考点，单击 *新增抓取注册*。

3. 选择抓取方式，可选择 **吸取侧面**、**吸取上端面**、**吸取上下端面**。然后选择控制轴坐标系，单击 对齐控制轴与基准坐标。如果抓取方式为吸取侧面，基准坐标的原点位于圆柱侧表面 Z 轴向正中间。如果抓取方式为吸取上（下）端面，基准坐标的原点位于圆柱上（下）端面的中心。

4. 单击 **对称性设置**下方 新增，通过旋转或平移变换遍历生成抓取圆柱所需的合理位姿。
   - 旋转遍历性设置
     a. 将对称方式设置为旋转。
     b. 选择旋转中心，默认为工件坐标系。
     c. 选择旋转轴，可选择 X 轴、Y 轴或 Z 轴。
     d. 设置正向旋转角度、反向旋转角度、间隔角度。在中间的预览窗口可预览生成效果。
     e. 单击 生成对称结果。在 **对称结果**下方查看生成结果。
     f. 单击界面右下角 完成。



- 平移遍历性设置

  a. 将遍历方式设置为平移。

  b. 选择对称方式。如果选择 **相对模式**，需设置 Z 轴方向偏移工件中心距离百分比；如果选择 **绝对模式**，需设置 Z 轴方向偏移中心的距离和间隔距离。

  c. 设置对称个数。

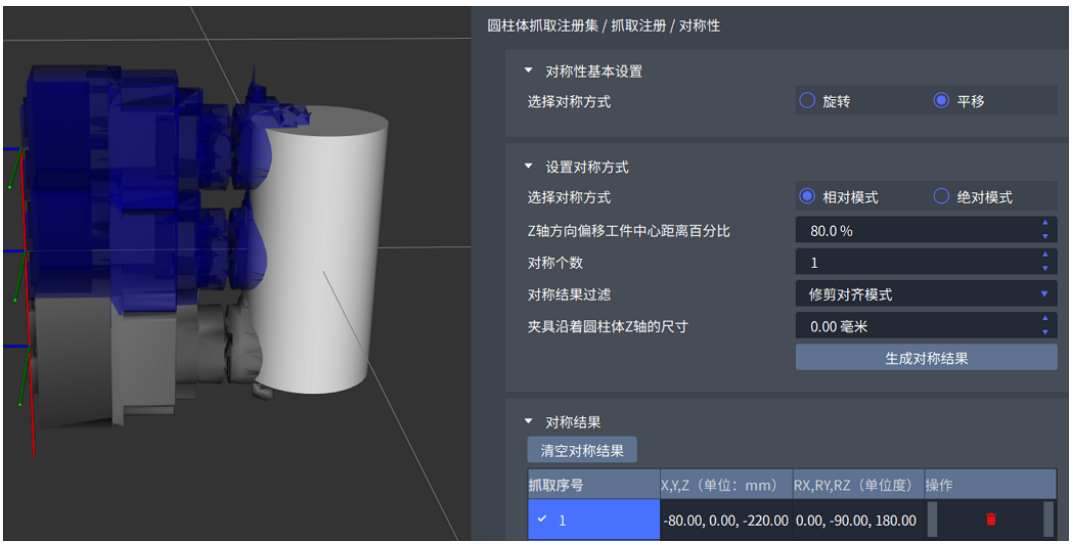
  d. 选择对称结果过滤方式，如果选择 **修剪对齐模式**或 **修建放弃模式**，还需设置夹具沿着圆柱体 Z 轴的尺寸。修剪对齐模式指修改超出圆柱侧表面高度的位姿，使之落于圆柱侧表面范围内；修剪放弃模式指舍弃超出圆柱侧表面范围的位姿。

  小技巧: 鼠标指向 **对称结果过滤**，可查看两种模式的图示。

  ---

  e. 单击 生成对称结果。在 **对称结果**下方查看生成结果。

  f. 单击界面右下角 完成。



5. 单击界面右下角 完成。

完成抓取方式注册后，还需将抓取方式添加到抓取处理流程中，以便任务规划时读取，请参见注册抓放规划。

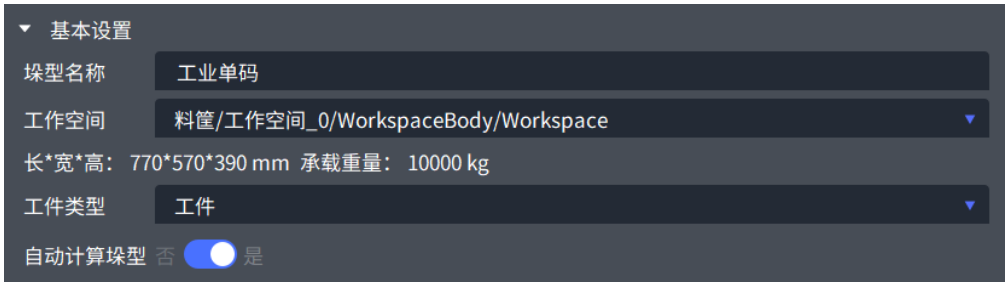
8.7 注册放置方式

根据应用场景，注册放置方式。

8.7.1 工业单码垛型注册（手动）

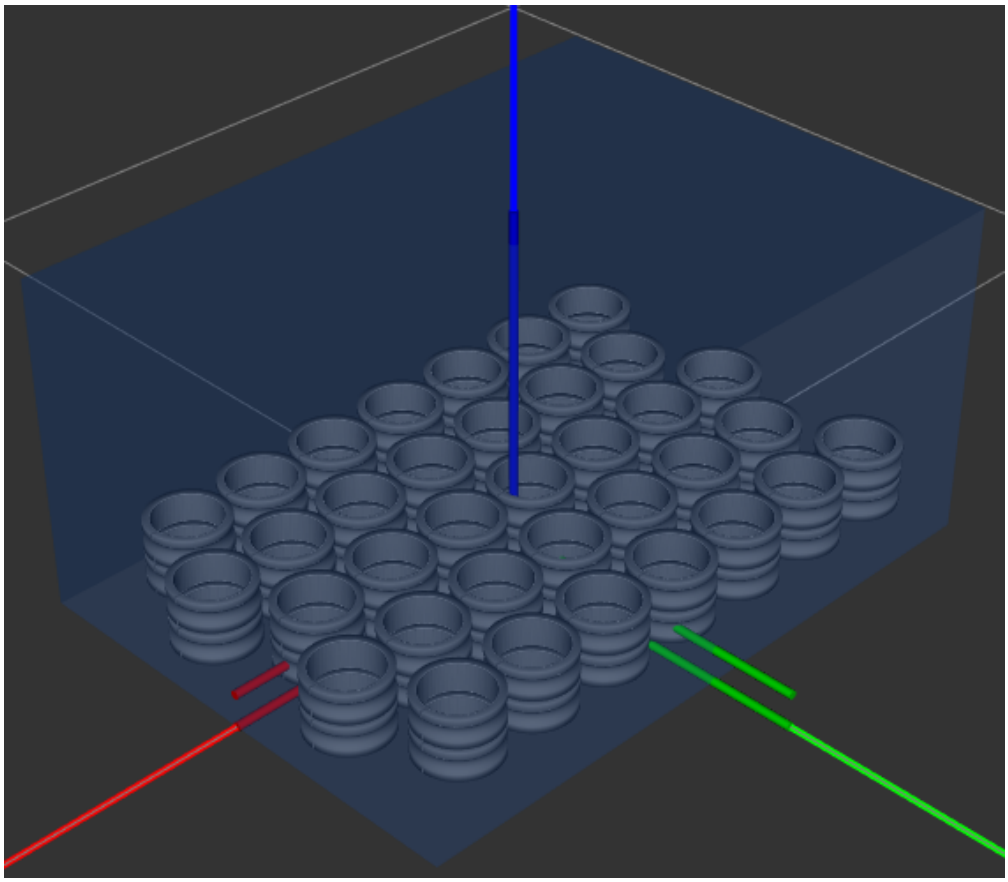
手动注册工业码垛垛型。

1. 在 运动页签下快捷工具栏，单击 抓取和放置，选择“注册放置方式 > 工业单码垛型注册（手动）”。
2. 单击界面左下角 **工业单码**节点，在右侧参数设置栏 **基本设置**下方，设置垛型名称、工作空间、工件类型等参数。



3. 在 **垛型规则设置**下方，设置垛型规则。
   - 码放空间尺寸：设置码放空间的长宽高。
   - 相邻工件中心距离：根据实际测量结果，设置相邻工件中心的距离。
   - 选择垛型：支持 **棋盘型**和 **蜂窝型**。如选择 **蜂窝型**，还需设置排列方向、相邻行位移、对齐方式。
   - 码垛层数：可选择 **自动计算**或 **自定义**，如选择 **自定义**，需手动设置层数。
   - 每层高度：设置每层码垛的高度。
   - 工件相对该层底面 z 方向值：设置工件相对于该层底面 z 方向的距离。
   - 奇偶层对应关系：如有多层，需设置奇数层和偶数层的对应关系。
   - 允许垛超出工作空间范围：允许 X 方向和 Y 方向上允许垛型超出工作空间的范围。

4. 单击 自动计算垛型，在中间预览窗口查看生成的垛型，确认无误后单击右下角 完成。



8.7.2 箱子单码垛型规则（自动）

添加箱子单码垛型规则，自动生成垛型。。

1. 在 *运动*页签下快捷工具栏，单击 *抓取和放置*，选择“注册放置方式 > 箱子单码垛型规则（自动）”。
2. 单击界面左下角 **单码垛型规则**节点，在右侧参数设置栏选择放置的工作空间，并单击 *编辑箱子*，根据实际情况修改箱子的尺寸和条码位置。

3. 在 *单码垛型规则设置*下方，选择垛型并配置相关参数。垛型可选择 **高装载**、**回字型**、**棋盘型**。
   - 高装载
     - 箱子尽量铺开：启用该功能后，在垛型规划时，会保证箱子尽量铺开。
     - 码垛层数：可选择 **自动计算**或 **自定义**，如选择 **自定义**，需手动设置层数。
     - 奇偶层对应关系：当码垛层数不只一层时，设置奇数层和偶数层的对应关系。
     - 箱子条码朝向：可选择 **不考虑条码朝向**或 **条码朝外**。
   - 回字型：设置码垛层数、奇偶层对应关系、箱子条码朝向。
   - 棋盘型
     - 码垛层数：可选择 **自动计算**或 **自定义**，如选择 **自定义**，需手动设置层数。
     - 箱子条码朝向：可选择 **不考虑条码朝向**、**条码朝外**或 **依据轴**。如选择 **不考虑条码朝向**或 **条码朝外**，还需设置箱子长边方向。如选择 **依据轴**，还需设置工作空间轴方向。
     - 自动布局方向：沿工作空间 X 轴或 Y 轴方向自动布局。
     - 沿布局方向个数：可选择 **自动计算**或 **自定义**，如选择 **自定义**，需手动设置布局个数。
     - 靠边选项：设置码垛在工作空间中的靠边位置。

4. 查看规划结果。
   - 若规划成功，可查看箱子数量、平面覆盖率、体积装载率等信息，并查看垛型。
   - 若规划失败，查看失败原因。

8.7.3 箱子单码垛型注册（手动）

手动添加箱子单码垛型。

1. 在 *运动*页签下快捷工具栏，单击 *抓取和放置*，选择“注册放置方式 > 箱子单码垛型注册（手动）”。
2. 单击界面左下角 **箱子单码垛型注册（手动）**节点，在右侧参数设置栏单击 *新增托盘*，设置托盘和工作空间的尺寸，单击 *确定*。
3. 单击 **垛型列表**下方 *新增垛型*，设置箱子尺寸、条码位置等参数，单击 *确定*。

4. 在 **垛型**页签下，单击 *使用自动计算垛型*。

5. 选择垛型并配置相关参数，参数配置参见箱子单码垛型规则（自动），然后单击 使用自动垛型生成层。
6. 单击右下角 完成。

8.8 注册抓放规划

注册抓取处理流程或组合处理流程。如项目涉及抓放组合时，选择组合处理流程。2D/3D 坐标移动和 3D 轨迹移动项目一般注册抓取处理流程即可，拆码垛项目一般需注册组合处理流程。

8.8.1 注册抓取处理流程

添加抓取处理流程，用于任务规划时读取抓取方式。

前提条件

在注册抓取处理流程前，需先注册抓取方式，操作参见注册抓工件方式.

操作步骤

1. 单击界面上方 抓取和放置，选择“注册抓放规划 > 抓取处理流程”。
2. 单击界面左下角 **抓取处理流程**节点，在右侧参数设置栏选择工件处理策略、抓取注册集、抓取规划处理策略。配置完成后，单击 下一步。

小技巧： 工件处理策略用于过滤工件，抓取规划处理策略用于对抓取规划进行修改、排序、过滤。鼠标指向前后处理策略下拉列表中的选项，可查看对应策略的简要说明。

3. （可选）选择抓取工作空间，单击 测试运行，查看运行结果，包括工件处理结果、抓取规划生成结果、抓取规划处理结果。

4. 单击右下角 完成。

8.8.2 注册组合处理流程

添加组合处理流程，用于任务规划时读取抓放组合方式。

前提条件

在注册抓取处理流程前，需先注册抓取方式，操作参见注册抓箱子方式.

操作步骤

1. 单击界面上方 抓取和放置，选择“注册抓放规划 > 组合处理流程”。
2. 单击界面左下角 **抓放组合处理流程**节点，在右侧参数设置栏选择工件处理策略、抓取注册集、抓取规划处理策略。配置完成后，单击 下一步。
3. 选择放置位姿来源、工作空间处理策略、抓放组合方式、抓放组合处理策略，单击 下一步。

4. （可选）选择抓取和放置工作空间，单击 测试运行，查看运行结果。
5. 单击右下角 完成。

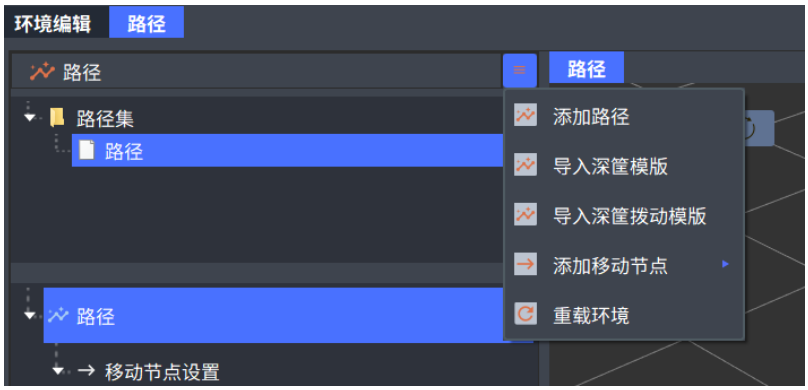
8.9 设置运动路径

机械臂运动路径一般涉及起始点、抓取点、放置点以及相对于抓取点、放置点的点位，起始点通过任务流图中的 **移动到目标关节**设置，其余点位在 路径侦错下的 **路径**面板设置。

以下为添加运动规划的简单示例，不同应用场景下不同项目添加的点位及具体设置应以项目实际情况为准。

1. 在 运动页签下的 **路径**面板，单击右上角  ，选择 **添加路径**。

> **提示：** 也可选择导入已有模板，比如深筐模板与深筐拨动模板，模板中包含预置路径节点。



2. 在右侧参数设置栏 **抓取配置**下方，设置抓取移动涉及的工作空间、工作流程、视觉服务编号，并根据需要在 **放置配置**下方设置放置相关参数。

3. 右键单击 **路径**面板左下角 **路径**，选择“添加移动节点 > 抓取移动节点”。单击左下角新生成的 **抓取移动节点**，在右侧参数设置栏选择运动类型。运动类型指前一个点到当前点的运动类型。

4. 右键单击 **路径**面板左下角 **抓取移动节点**，选择“添加移动节点在前 > 相对移动节点”。单击左下角新生成的 **相对移动节点**，在右侧参数设置栏将 **相对节点**设置为 **抓取移动节点**，单击 **目标点**右侧 添加，设置相对位姿，单击 确定。

5. 右键单击 **路径**面板左下角 **抓取移动节点**，选择“添加移动节点在后 > 相对移动节点”，再生成一个相对移动节点，代表机械臂在抓取工件后相对移动点位。单击该节点，将相对节点设置为 **抓取移动节点**，再添加目标点并设置相对位姿。

6. 右键单击上一步生成的 **相对移动节点**，选择“添加移动节点在后 > 绝对移动节点”，用该节点表示放置点位。单击左下角新生成的 **绝对移动节点**，在右侧参数设置栏将 **抓放状态**设置为 **放下工件**，设置物体掉落高度，单击 **目标点**右侧 添加，添加放置点的机械臂关节角。

7. 确认各点位的碰撞检测设置。单击各节点，在右侧参数设置栏 **碰撞检测设置**下方查看碰撞检测参数。
   - 检查目标点位碰撞：检测该点是否发生碰撞。
   - 检查插值点位碰撞：检测从上一个点到该点之间插值点位是否发生碰撞。

提示： 单击 **高级参数**右侧  ，可查看并设置碰撞检测的高级参数。

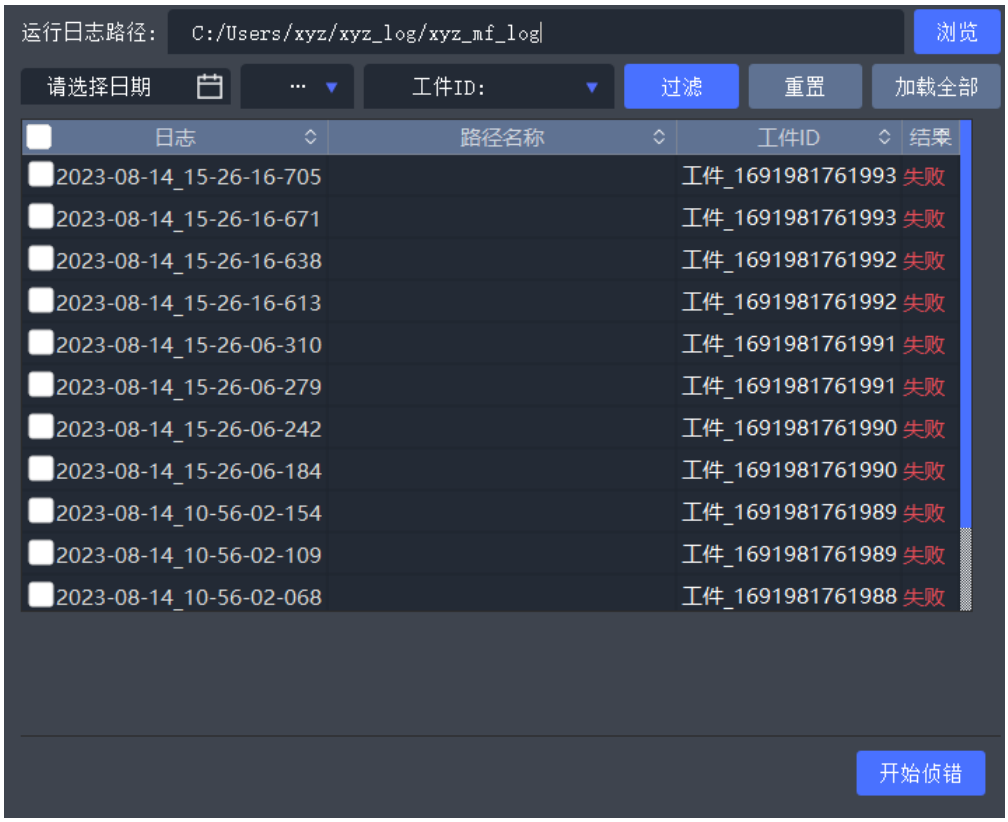
8. 右键单击中间预览窗口上方 **路径**页签，选择 **保存**。

8.10 路径侦错

如果轨迹规划失败，可使用 Max 自带的侦错工具排查问题。

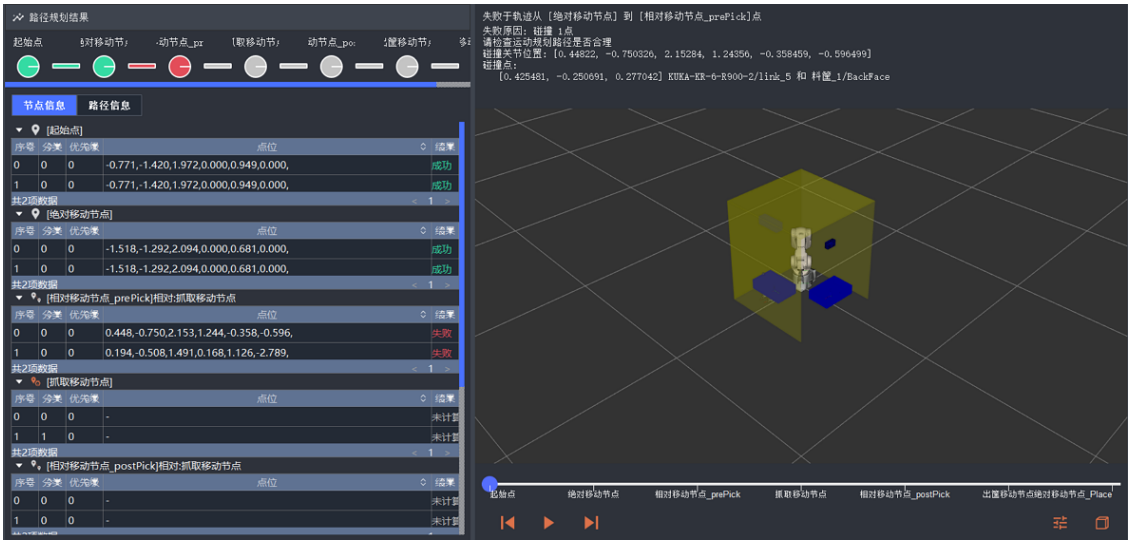
1. 在 运动页签下快捷工具栏，单击 路径侦错，或在 Max 顶部菜单栏，选择“工具 > 路径侦错”。
2. 选择“文件 > 打开最近一次日志”，或选择“文件 > 打开历史日志”，选择一条或多条日志后，单击 开始侦错。

> **提示：** 可根据日期、工件 ID 等条件对日志进行过滤。



3. 查看路径规划结果。路径节点如果为绿色，表示规划成功，红色表示失败，灰色表示未计算。单击红色节点，查看详细信息。
   - 节点信息：显示各节点的优先级、点位、规划结果等信息。

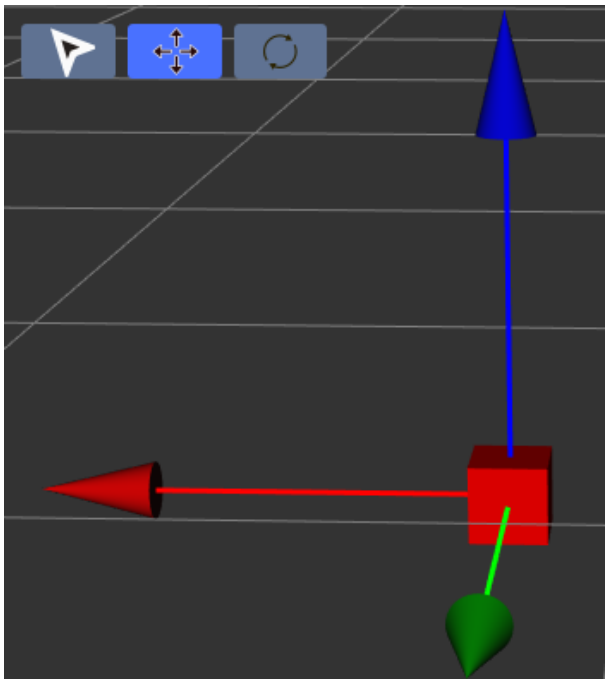
- 路径信息：显示各路径规划结果。单击路径后，单击底部  ，可自动演示轨迹。单击右下角  ，打开状态窗口，显示当前机械臂关节角。单击右下角  ，进入线框模式，隐藏可视化模型，以线框显示碰撞模型。



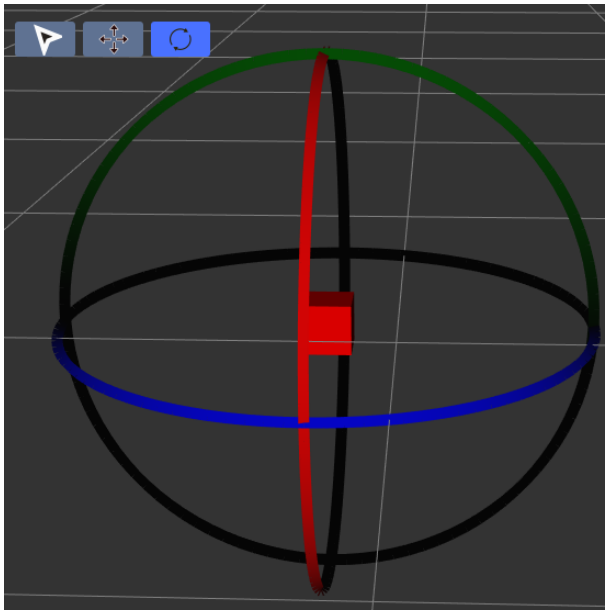
运动页签下常见的操作按钮说明如下：

可视化窗口左上角控制按钮

-  ：按住 X、Y、Z 轴三个方向的箭头移动位置。



-  ：按住圆环旋转角度。



-  ：退出移动或旋转模式。在移动或旋转模式下，无法更改右侧参数设置栏信息。

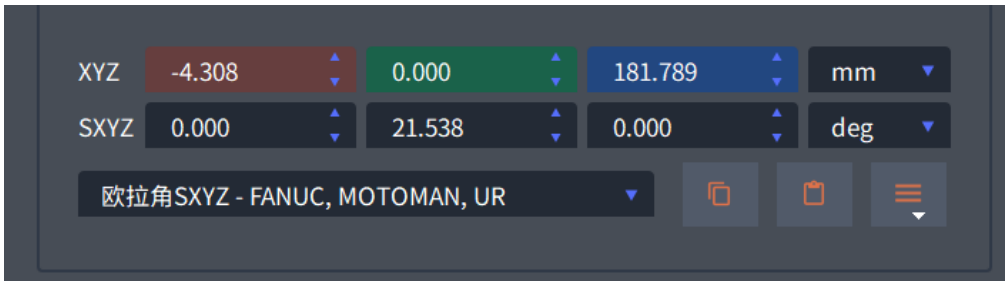
右侧参数设置栏常见按钮

-  ：浏览并选择文件路径。
-  ：控制对应模型是否可见。
-  或  ：删除。

-  ：编辑。
-  ：复制。
-  ：打开模型编辑器，修改模型。
-  ：复制当前矩阵位姿。复制后的数据示例如下：

```
[[ 0.930177, -0.000000,  0.267112, -0.004308],
[ 0.000000,  1.000000,  0.100000, -0.000000],
[-0.367112,  0.000000,  0.630177,  0.181789],
[ 0.000000,  0.000000,  0.000000,  1.000000]]
```

-  ：粘贴矩阵位姿。根据粘贴的数据调整当前位姿。
-  ：单击该图标后，可选择以当前旋转格式及单位复制、粘贴位姿或旋转角，或者将位姿进行逆向处理，也可将当前位姿设置为单位矩阵。



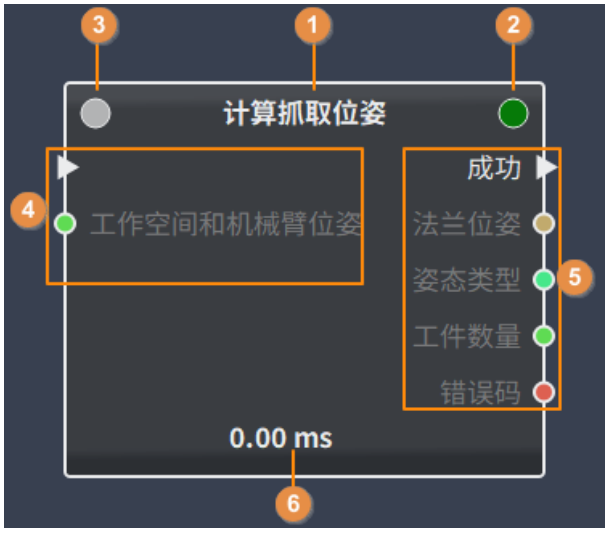
CHAPTER 9

基本任务操作

任务部分的主要功能是搭建、配置不同场景的完整的工作流程，并控制整个流程运行。该部分采用可视化编程的方法，并提供丰富的功能模块。用户无需编写代码即可根据需求灵活搭建工作流图，方便快捷。

9.1 模块结构

任务中的模块由六个部分组成，其功能结构和说明如下。



1. 模块名称
2. 模块单独运行按钮
3. 模块断点设置按钮
4. 输入，是一个模块运行时所需的事件结果和参数
5. 输出，是一个模块经过计算后得到的事件结果和数据

6. 模块单次运行耗时

9.2 流图配置

任务的流图模板可以大致分为以下四个部分：主函数、初始化（对应主函数中的初始化模块）、运动规划（对应主函数中的运动规划模块），以及运动执行（对应主函数中的运动执行）。

任务预置了多种场景的流图模板，并提供了事件、流程控制、函数、变量、运动规划、运动执行等多种类型的模块，其基本操作方法如下。

新建流图

1. 在菜单栏选择“文件 > 新建”，新建空白流图，也可在菜单栏选择“文件 > 读取”，已有文件，如下图所示。

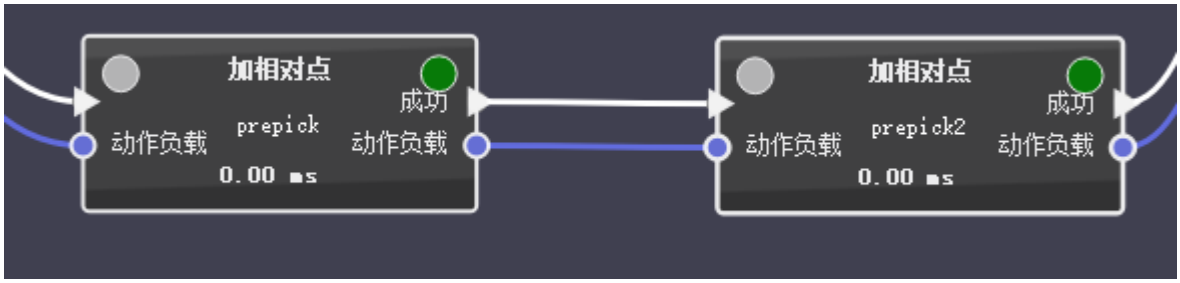
2. 在右侧 节点选择页签下，选择模块拖拽至流图配置区域，或右键单击流图窗口，选择 **添加节点**，搜索添加模块。

导入模板

在任务菜单栏选择“文件 > 新建”，根据应用场景选择模板，设置流图名称后单击 完成。

连接和删除模块

1. 从前置模块的输出端口拖出连线至下一个模块的输入端口，完成模块连接。



提示: 只有接口图形和颜色完全相同的两个模块接口才可以相互连接。三角表示逻辑流，圆圈表示数据流。

---

2. 删除连接线时，单击待删除的连线，连线边框高亮显示，表示已选中。然后按【Delete】键或右键单击连线，选择 删除。

编辑属性

选中待编辑的模块，在右侧 属性页签配置模块参数。

提示: 单击  ，运行模块。单击  ，查看功能描述。

运行模块或流图

- 运行当前模块：单击模块右上角的绿色圆点，可执行当前模块。单击模块左上角的灰色圆点，圆点变成红色，可设置检查断点。

- 运行完整流图：单击页面左上角工具栏中的开始按钮  ，也可在任务菜单栏选择“运行 > 运行”。同时，也可在 运动界面执行。

- 运行至当前模块：单击选择某个模块，模块边框高亮表示已选中。然后右键单击模块，在弹出的列表中选择 **运行直到选中节点**，或在任务菜单栏选择“运行 > 运行到选中节点”。

保存流图

右键单击当前页签，选择 保存，或在任务菜单栏选择 “文件 > 保存”。

Part III

典型场景

CHAPTER 10

2D/3D 坐标移动

该场景使用视觉识别计算出工件坐标后，采用坐标移动的方式进行抓取，无需轨迹规划、碰撞检测，使用方便高效。

使用资源包

在配置该场景前，可先登录 XYZ Studio Max 资源中心，下载 2D/3D 坐标移动的资源包。下载完成后，按照以下步骤熟悉该场景的视觉、运动、任务配置。

1. 打开 Max，在顶部菜单栏选择“文件 > 打开项目”，选择并打开资源包里的.max 文件。
2. 加载虚拟相机。工控机可通过虚拟相机，在不连接相机的情况下从本地获取图片等数据。
   a. 在 视觉页签下，打开相机模块。
   b. 单击界面右上角 相机资源管理器，单击 添加虚拟相机，选择并打开资源包虚拟相机文件夹中的 camera.json 配置文件，单击 确定。

c. 右上角 **相机型号**选择资源包中的虚拟相机型号，单击 保存。单击中间预览窗口上方  ，可查看虚拟相机中的图片。

3. 查看视觉、运动、任务的配置。

案例讲解

本章以一个简单的圆环抓取项目为例，介绍 2D/3D 坐标移动场景的配置。在本项目中，圆环工件平铺摆放在料筐内，相机安装在机械臂上，要求使用 ABB-IRB-1200-5-0-9 机械臂，配合直爪夹具，将工件抓取并放入另一个料筐中。配置教程如下：

10.1 新建项目

根据项目场景，选择模板，创建项目。操作可参考新建项目。

1. 在 Max 顶部工具栏，选择“文件 > 新建项目”。
2. 选择场景模板，在本项目中，选择 **2D-坐标移动-眼在手上**或 **3D-坐标移动-眼在手上**模板。
3. 添加机械臂，本项目使用 ABB-IRB-1200-5-0-9 机械臂。
4. 选择 **高级参数**后，添加夹具和工件的 STL 模型，将 **抓取工作空间**设置为 **料筐**，其他参数保持默认。也可不添加夹具和工件模型，使用现实示教方式注册抓取。
5. 单击 确认。

10.2 视觉

本章节将针对具体场景详细介绍视觉服务的配置过程。在进行视觉配置前，请先参见连接机械臂，完成机械臂连接。

10.2.1 2D 单层平面多物体抓取

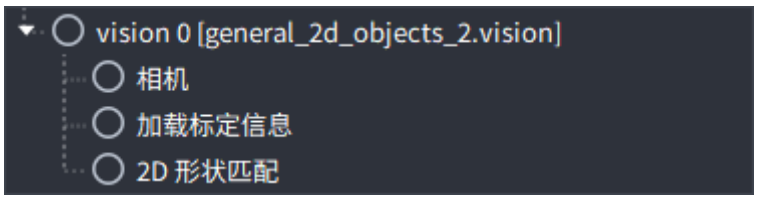
适用于多种已知物体，需要注意的是物体种类越多速度越慢，场景要求如下。

- 物体在同一平面稳定摆放。
- 使用工业 2D 相机，相机平行于平面拍照。

本章节描述的相关配置仅供参考，具体操作请以实际项目为准。

基本配置

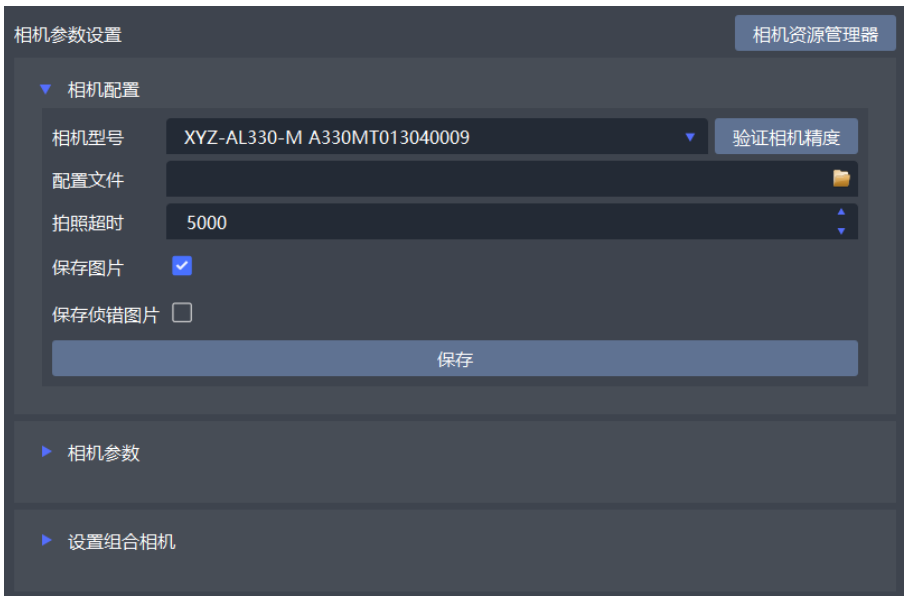
配置前，请依次完成新建项目、添加工作空间，并使用 **2D 单层平面多物体抓取**模板创建视觉服务，请参见创建流图。创建完成之后会在左上角生成节点树，如下图所示。



根据需求，依次配置这三个模块。

步骤 1. 配置相机模块

1. 单击左侧节点树中的 **相机**模块，然后在右侧单击 相机资源管理器以连接相机，接着在 **相机配置**区域设置相机基本信息，请参见连接相机。
2. 在 **相机参数**区域单击 验证相机精度，开始 2D 相机的标定。



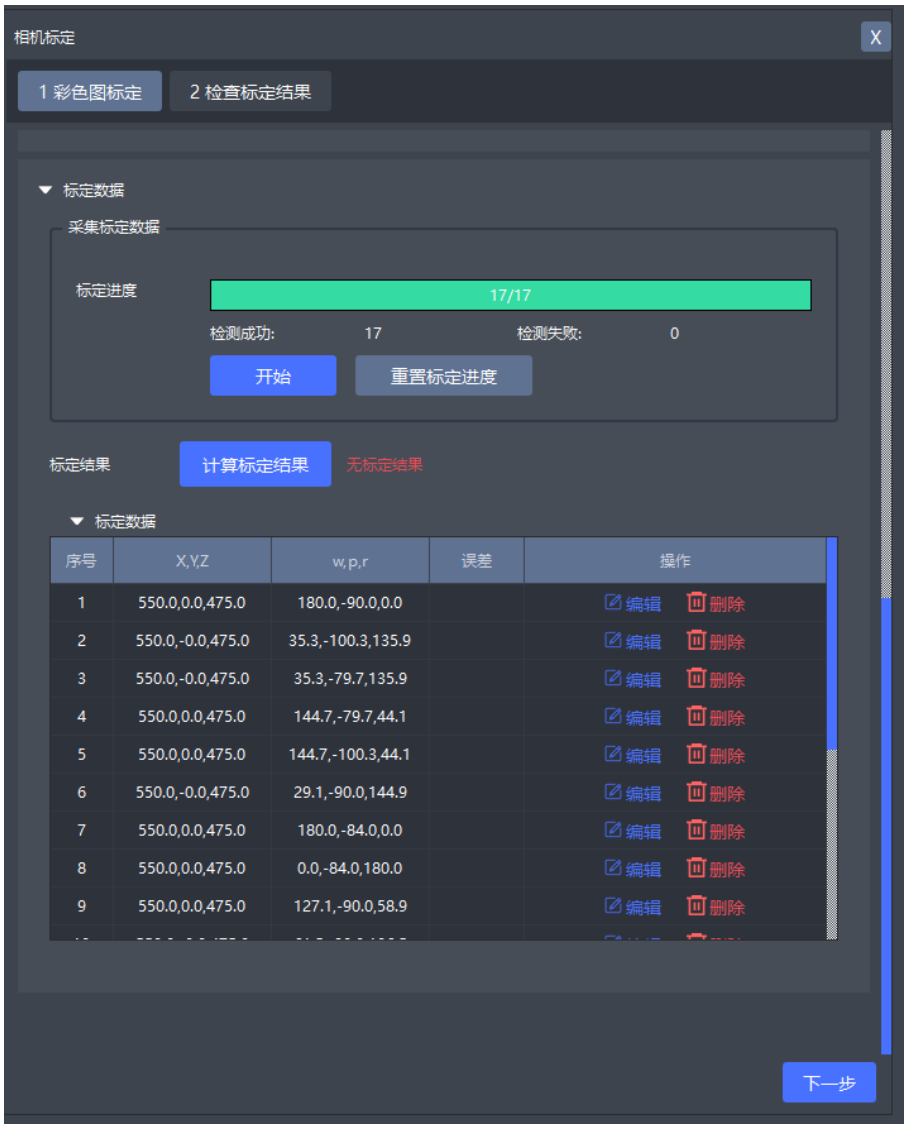
3. 依次选择标定板、标定内参数、验证标定结果等步骤，请参见内参标定。

步骤 2. 配置加载标定信息模块

1. 单击左侧节点树中的 **加载标定信息**模块，软件界面右侧出现 **3D 标定设置**区域，在该区域依次标定相机和工作空间。

2. 单击 **相机标定**区域中的 *新建标定数据*，设置标定数据名称，然后单击 *编辑标定数据*，设置标定方法和其它标定信息，请参见手眼标定 。

3. 完成数据标定（标定数据时，有 8 个点采集成功即可）并计算标定结果，单击 下一步进入 **检查标定结果**页签。



4. 检查并根据界面提示确认标定误差后，单击 **完成**，即可完成相机标定。

5. 标定工作空间，请参见工作空间标定。2D 相机使用角触碰或者边缘触碰的方式标定工作空间。

步骤 3. 配置 2D 形状匹配模块

1. 创建模板。

   a. （可选，该步骤仅对 **眼在手上**的场景生效，眼在手外的场景请跳过此步骤。）单击 Max 右上角的 设置拍照位姿，在弹出的窗口中设置视觉服务、相机 ID，单击 获取机械臂位姿，然后单击 设置拍照位姿即可获取最新的拍照位姿。

b. 单击左侧节点树中的 **2D 形状匹配**模块，在 **检测物体**页签下单击 创建模板开始训练工件模板。如果有已训练好的模板，可以直接选择模板检测物体。

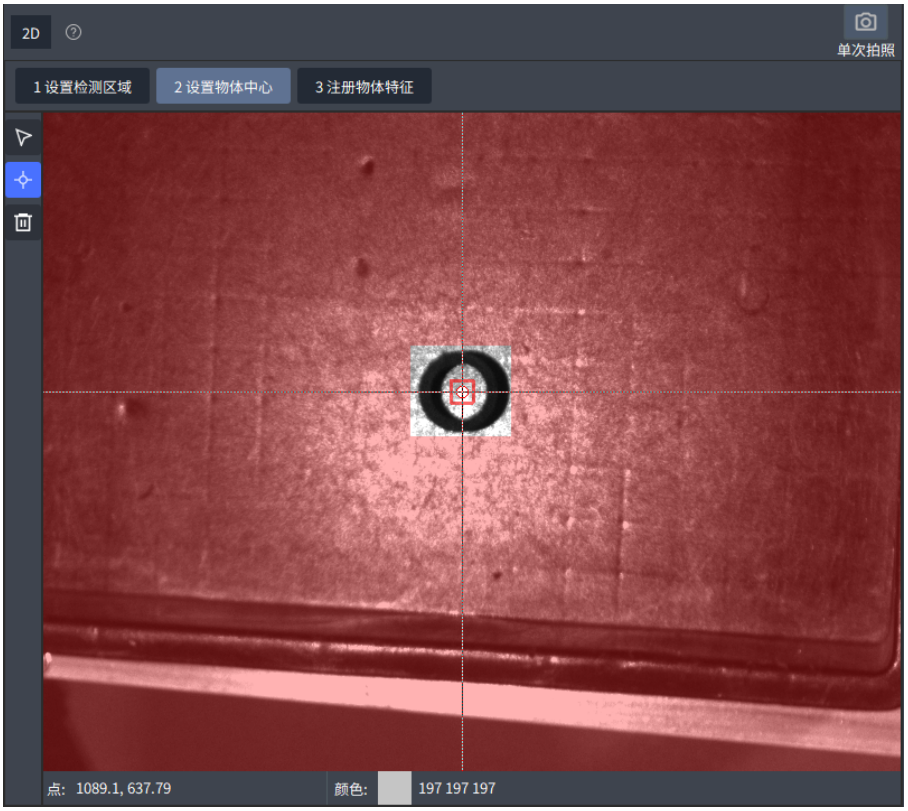
c. 在 **训练工件模板**页签单击 创建工件类型并给工件类型命名，工件模板名称默认与工件类型名称相同，用户可自行修改。测量工件尺寸，并输入工件长、宽、高。然后根据需求设置基础参数，开始训练工件模板。

2. 训练模板。

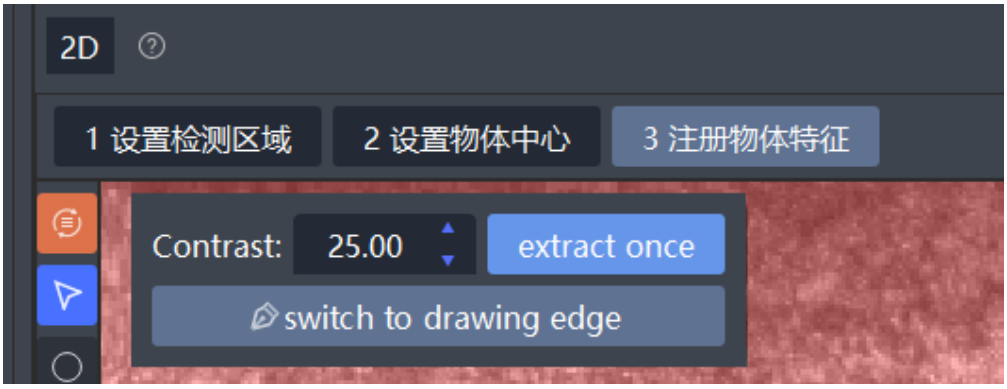
   a. 在预览窗口单击 单次拍照获取图像，然后设置检测区域，检测区域尽量贴合工件边缘。如下图所示，是设置好检测区域的结果。

   在选中  时，可单击并拖动图中的白色圆点，修改矩形框的尺寸和位置。如果不满意，可单击预览区左侧的  ，重新设置检测区域。

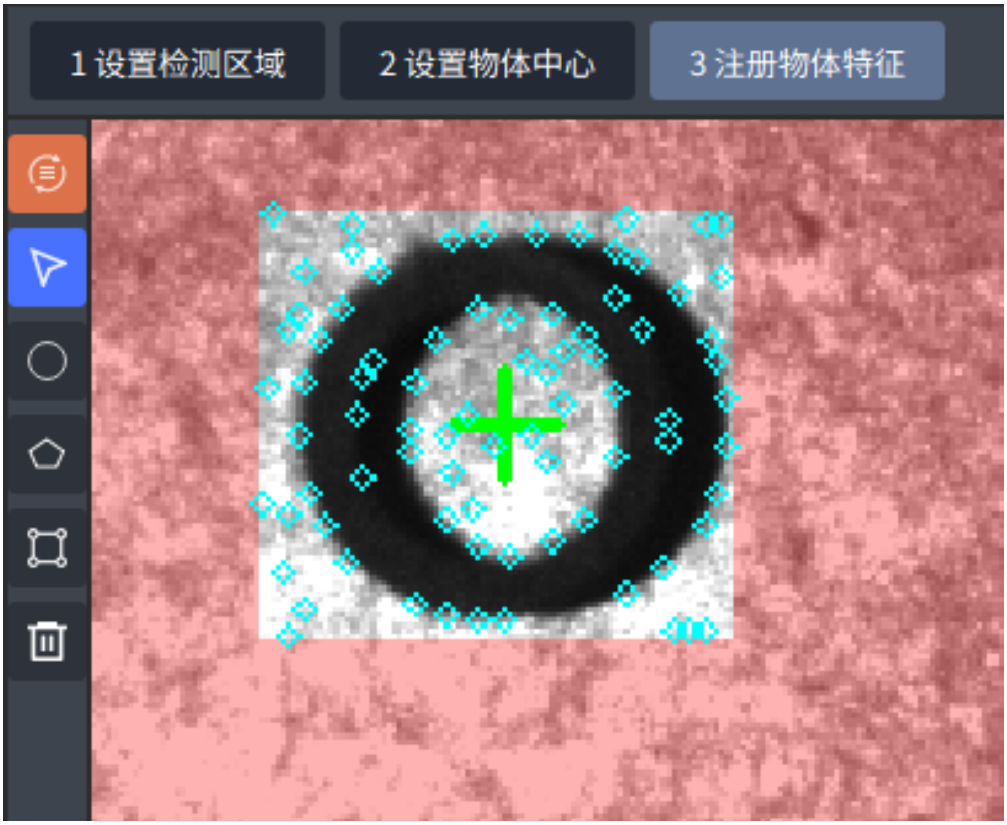
b. 设置物体中心，对于圆环形工件，尽量贴近圆心。



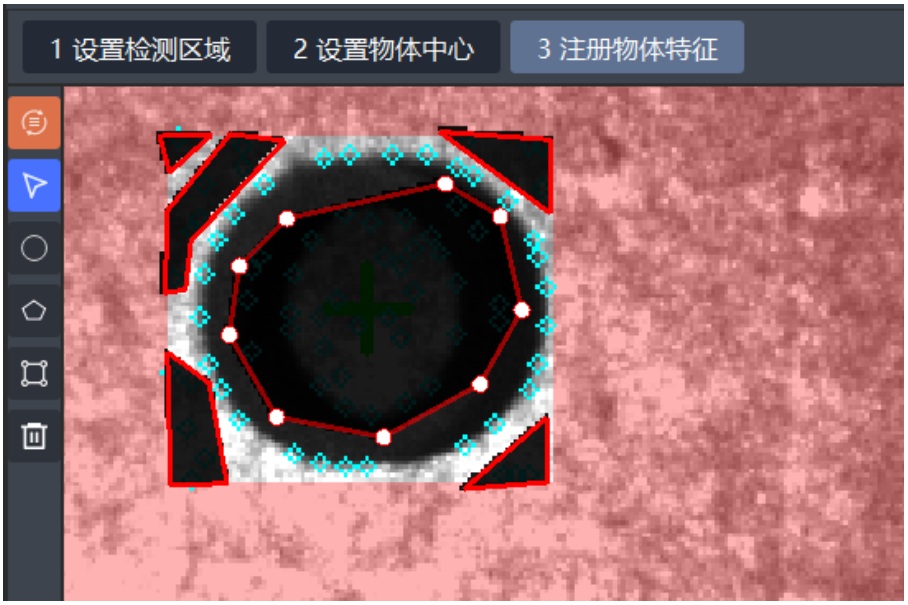
c. 单击  ，根据需要调整对比度，然后单击 *extract once* 获取特征点。



自动提取的特征点如下图所示。



d. 使用自定义图形遮盖非物体特征点，仅保留物体轮廓的特征点。



提示:

- Max 提供三种注册物体特征的方式：圆形、矩形、自定义图形。对于不规则工件，用户可选择自定义图形，手动勾画特征点并连接成线，尽量与工件边缘保持贴合以更好地注册物体特征。
- 对于圆环形工件，也可使用圆形勾画出工件的轮廓，且圆心尽量与物体中心重合。
- 在训练模板时，注意打光方式，使得工件的阴影区域越小越好。

e. 单击 *保存并训练* Max 自动完成工件模板训练，然后单击 *完成* 可使用训练好的模板。

3. 检测物体。

   a. 在 *检测物体* 页签选择已训练完成的工件模板。

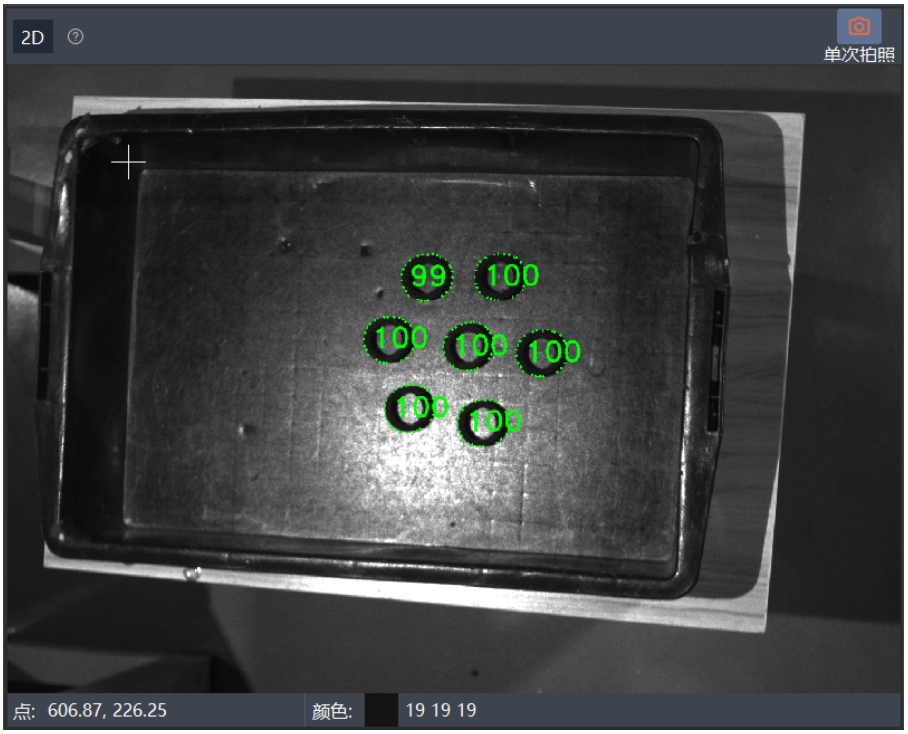
b. 设置显示参数，用户可根据需要选择是否显示匹配点、分数和边缘。

c. 设置检测参数，各参数含义如下。

- 最小检测分数：最大为 100，分数越高匹配越准确，但匹配的数量可能减少。
- 有效检测数量：检测物体的最大数量，最小值为 1。
- 检测对比度：数值越小，边缘越多，检测速度越慢，精度不一定提升。
- 弱检测对比度：处于该值与对比度之间的点会根据连接性进行选择，可增加边缘点连续性。
- 非极大值抑制分数：检测出来的两个物体的交并比 (重叠区域面积/两物体总面积) 大于该值时，检测分数低的物体会被略去。

- 启用所有物体非极大值抑制：是否启用所有类型物体的非极大值抑制。
- 扩增像素和金字塔：图像金字塔梯度扩增像素数。

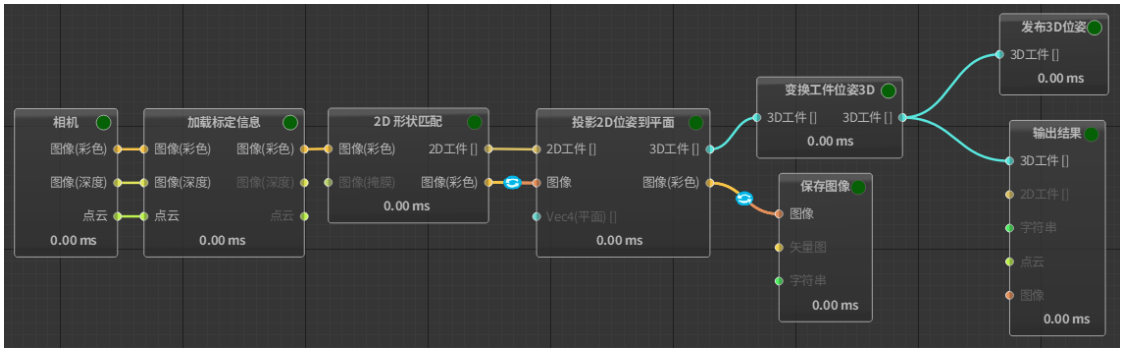
d. 单击 运行检测，开始检测物体，根据显示参数的设置会在图像预览区显示结果。



流图详解

视觉流图的配置是 Max 中的高级用法，对于一般用户而言，配置完节点树中的模块即可运行整个视觉流程，无需再配置视觉流图。视觉流图中包含了节点树中没有的模块，可以配置一些高级参数，如有需要可以按照以下内容操作。

单击“视图 > 节点视图”可查看创建成功的视觉流图，**2D 单层平面多物体抓取**场景的流图如下。



2D 单层平面多物体抓取场景的流图可大致分成三个功能组：图像获取、图像计算处理和图像输出，各模块的配置及使用流程如下。

1. 获取待处理图像。
   a. **相机**模块完成内参标定，并获取彩色图。
   b. **加载标定信息**模块完成相机和工作空间的标定。

2. 对图像进行计算处理。
    a. **2D 形状匹配**模块，训练工件模板供 2D 匹配使用。
    b. **投影 2D 位姿到平面**模块将输入的 2D 位姿投影到平面上，生成 3D 位姿。
3. 输出图像处理结果。
    a. 通过 **保存图像**模块保存 **投影 2D 位姿到平面**模块的输出结果。
    b. **变换工件位姿 3D** 模块将物体位姿从相机坐标系转换到世界坐标系。然后通过 **发布 3D 位姿**和 **输出结果**模块输出 3D 物体。

10.2.2 3D 平铺抓取

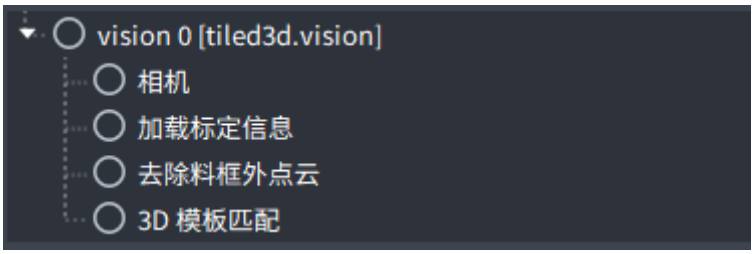
该场景适用于工作空间内同种物体平铺有序摆放，且拥有物体 STL 模型的情况。目前广泛应用于汽车零件、机加工件等多种场景，场景要求如下。

- 相机为结构光相机
- 已知工件的三维模型

本章节描述的相关配置仅供参考，具体操作请以实际项目为准。

基本配置

配置前，请依次完成新建项目、添加工作空间，并使用 **3D 平铺抓取**模板创建视觉服务，请参见创建流图。创建完成之后会在左上角生成节点树，如下图所示。



根据需求，依次配置这四个模块。

步骤 1. 配置相机模块（请参见连接相机。）

步骤 2. 配置加载标定信息模块（请参见手眼标定。）

提示: 对 3D 相机进行标定时，推荐进行点云标定。

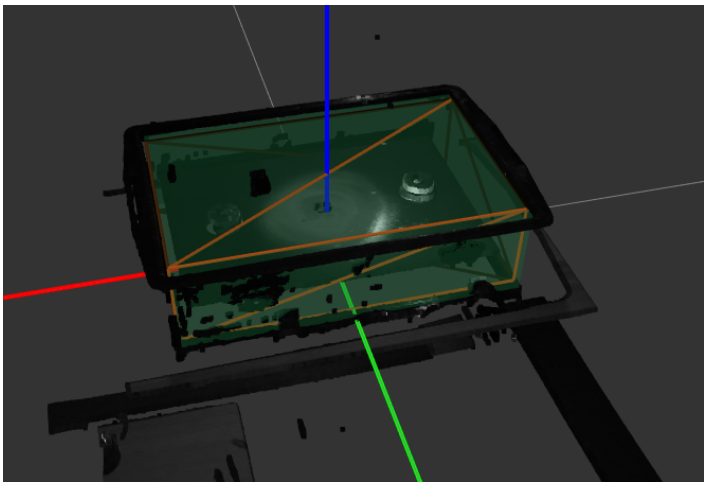
步骤 3. 配置去除料筐外点云模块

1. 单击左侧节点树中的 **去除料筐外点云**模块，然后在右侧配置点云切割参数。

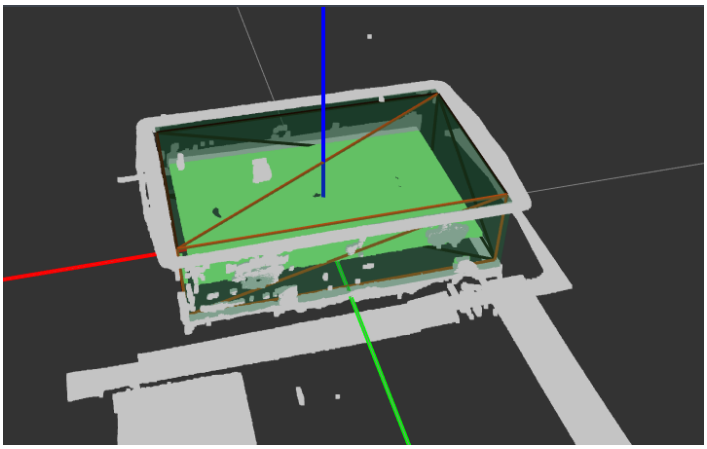
- X 方向收缩长度：料筐沿 X 方向的收缩长度，负值则意味着扩张。
- Y 方向收缩长度：料筐沿 Y 方向的收缩长度，负值则意味着扩张。
- 底部抬升高度：料筐底部抬升高度，负值则意味着下降。
- 顶部下降高度：料筐顶部下降高度，负值则意味着抬升。

单击 **触发**拍照后可获取点云。需要注意的是，这里设置的参数使得工作空间中完整包含工件点云即可，点云不是越多越好，点的数量过多会影响计算速度。

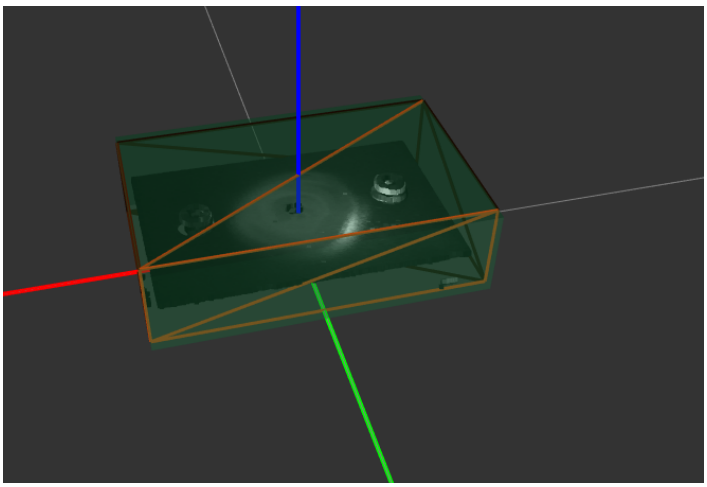
2. 在预览视图中移动料框模型，使之与点云中的料筐位置吻合。
3. 在 **可视化显示**区域选择需要查看的图像。
   - 未切割点云图：仅显示未去除料筐外点云的图像。



- 切割对比图：对比显示切割前后的图像，被切割的点云用白色显示。



- 切割完点云图：仅显示去除料筐外点云的图像。



步骤 4. 配置 3D 模板匹配模块

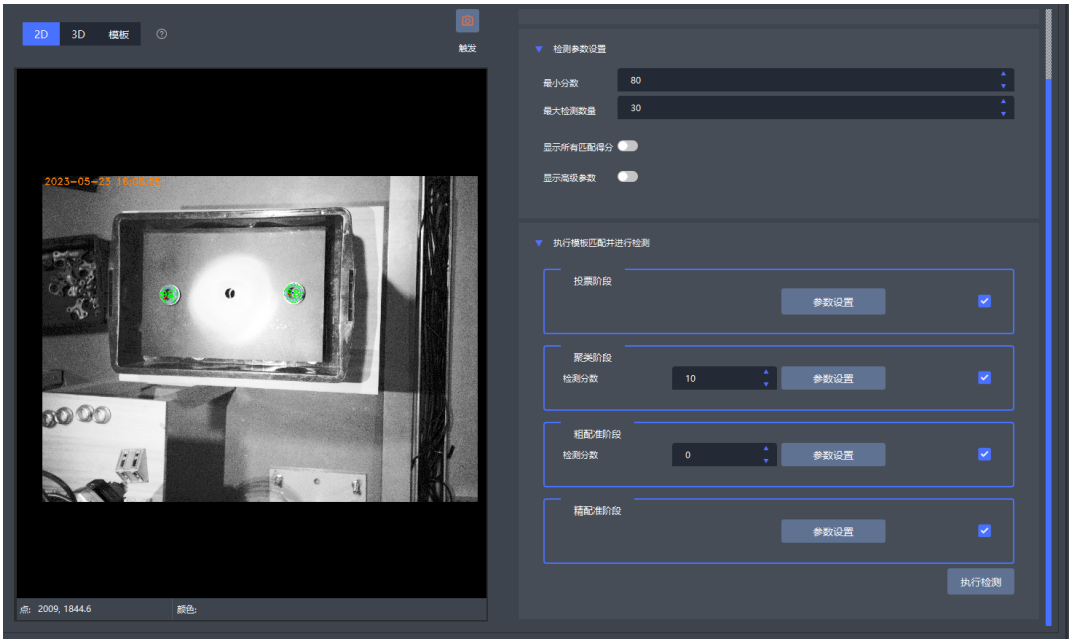
1. 单击 创建模板，在弹出窗口单击 选择模板，选择模板文件，并输入模板名称。工件类型必须和注册抓取时使用的工件名称一致。

提示： 在右下角 *属性*页签下，将 **是否使用已创建的模板**设置为 true，再次打开 Max 时会自动读取已创建的模板，不必重新创建。

2. 根据需要设置模板参数，单击 *确认*，完成参数设置。

提示： 根据需要选择 *特征类型*，如果模型表面有棱有角或者整体较扁平，建议尝试 edge-I/edge-II 模式。*限制物体朝向*建议使用 no_limit。

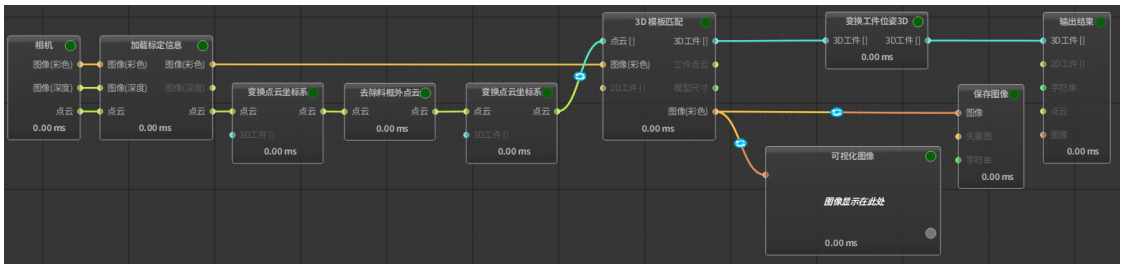
3. 根据需要设置检测参数。模板匹配时，小于 *最小分数*的结果将被过滤。
4. 设置不同检测阶段的参数，系统会根据物体模型在点云中寻找匹配结果，单击 *执行检测*，然后在左侧查看检测结果。如下图所示，检测出的工件上有分数显示则表明检测成功。



流图详解

视觉流图的配置是 Max 中的高级用法，对于一般用户而言，配置完节点树中的模块即可运行整个视觉流程，无需再配置视觉流图。视觉流图中包含了节点树中没有的模块，可以配置一些高级参数，如有需要可以按照以下内容操作。

单击“视图 > 节点视图”可查看创建成功的视觉流图，**3D 平铺抓取**场景的流图如下。



3D 平铺抓取场景的流图可大致分成四个功能组：图像获取、图像预处理、图像计算处理和图像输出，各模块的配置及使用流程如下。

1. 获取待处理图像。
    a. **相机**模块完成内参标定，并获取彩色图。
    b. **加载标定信息**模块完成相机和工作空间的标定。
2. 对图像进行预处理。
    a. 使用 **变换点云坐标系**和 **去除料筐外点云**模块，将点云从相机坐标系转换到工作空间坐标系，并去除料筐以外的点云。
    b. 基于上一步得到的料筐内点云，配置 **变换点云坐标系**模块，将点云从工作空间坐标系转换到相机坐标系。
3. 对图像进行计算处理，结合步骤 1.b 得到的彩色图和步骤 2.b 得到的点云，配置 **3D 模板匹配**模块，根据物体模型在点云中寻找匹配结果。

4. 针对步骤 3 的计算结果，输出或保存图像。
    a. 配置 **变换工件位姿 3D** 模块，将物体位姿从相机坐标系转换到世界坐标系。然后配置 **输出结果**模块，查看 3D 物体位姿。
    b. 也可使用 **保存图像**模块保存结果，或使用 **可视化图像**模块查看可视化彩色图。
5. 重复上述步骤，完成其它料筐的视觉流图的设置。

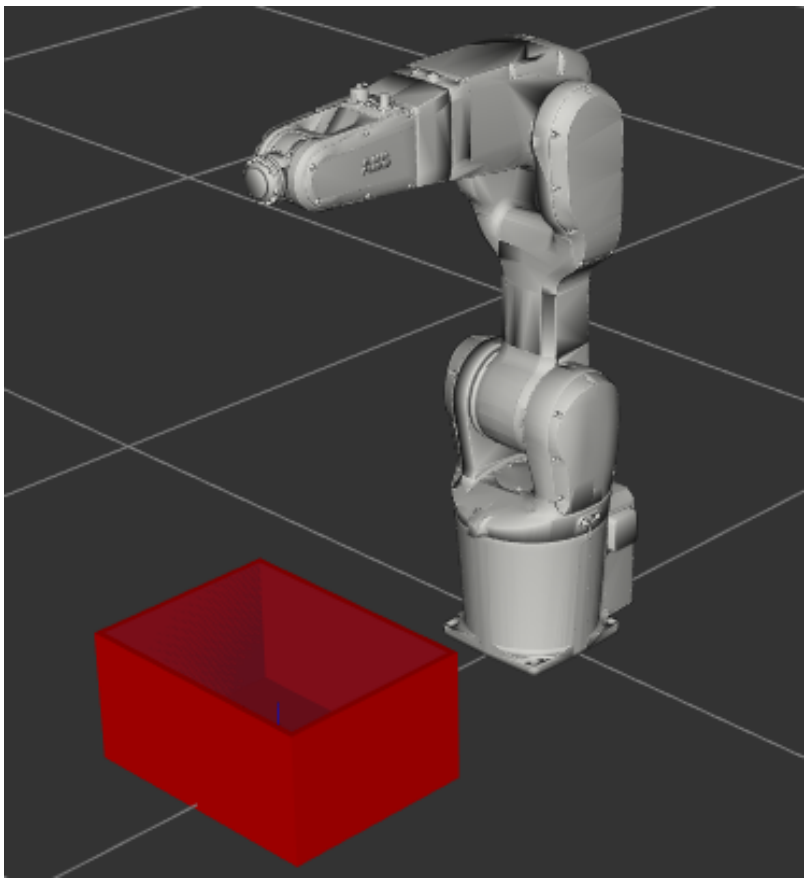
10.3 运动

在 运动界面，设置机械臂的抓取处理流程。

提示： 在开始设置运动相关内容前，请先根据机械臂和相机品牌连接并配置真实环境中的机械臂和相机。相关内容请参考连接机械臂 和设置相机 *IP*。在仿真环境中运行无需配置机械臂和相机。

---

1. 创建项目时，已添加机械臂。如未添加，可在 运动页签下快捷工具栏，单击 机械臂，添加机械臂，请参见添加机械臂。
2. 创建项目时，已添加料筐。如未添加，可在顶部快捷工具栏，单击 工作空间，选择 **添加料筐**，添加料筐后设置工作空间的尺寸、位姿、编号，请参见添加料筐。



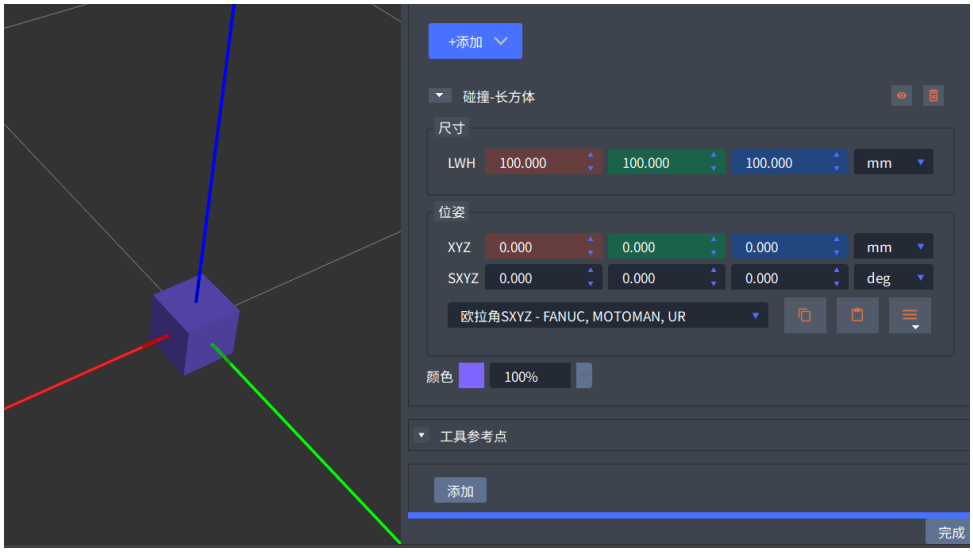
3. 注册工具、工件，再注册抓取注册集。2D/3D 坐标移动不涉及碰撞检测，在使用现实示教方式的前提下，工具和工件碰撞模型可不使用 STL 模型，比如可用长方体代替，软件从视觉检测结果中获取工件。如使用虚拟模型注册抓取，则需导入 STL 模型文件。

使用现实示教方式注册抓取

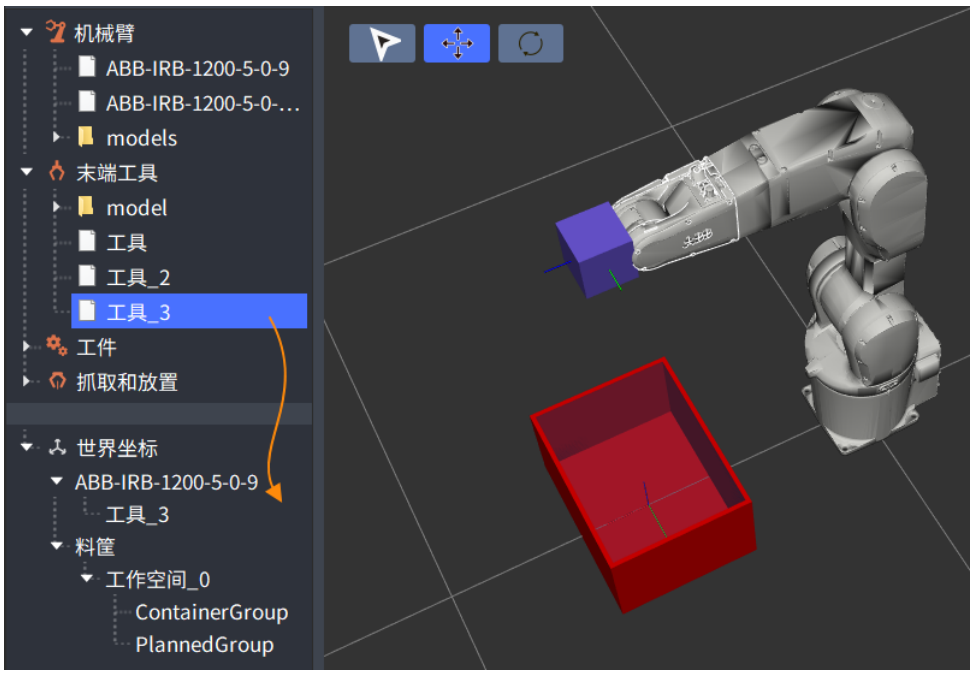
a. 在 运动页签下快捷工具栏，单击 末端工具，选择 **注册工具**。单击界面左下角 **工具**节点，在右侧参数设置栏单击 添加，添加工具形态。

b. 单击 **碰撞模型**下方 添加，可不添加模型文件，用长方体代替，再单击 **工具参考点**下方 添加，添加工具参考点，工具参考点位姿保持默认值即可。

c. 单击界面右下角 完成，工具注册完成。



d. 单击中间窗口上方的 **环境**页签，将左侧环境编辑面板中的工具文件拖放到左下方机械臂模型节点下。



e. 在顶部快捷工具栏，单击 工件，选择 **注册工件**。单击界面左下角 **工件**节点，单击右侧参数设置栏 **碰撞模型**下方添加，选择 **长方体**。

f. 在顶部快捷工具栏，单击 *抓取和放置*，选择 "注册抓取方式 > 工件抓取注册"。单击界面左下角 **抓取注册集**节点，在右侧参数设置栏 **基本设置**下方选择已注册的工具和工件，将工件来源设置为 **视觉侦测**，选择视觉服务、相机使用场景，然后单击 *检测工件*，选择检测到的工件位姿，单击 *创建注册抓取*。

g. 将注册方式设置为 **现实示教**，移动真实机械臂至抓取位姿，输入机械臂位姿。

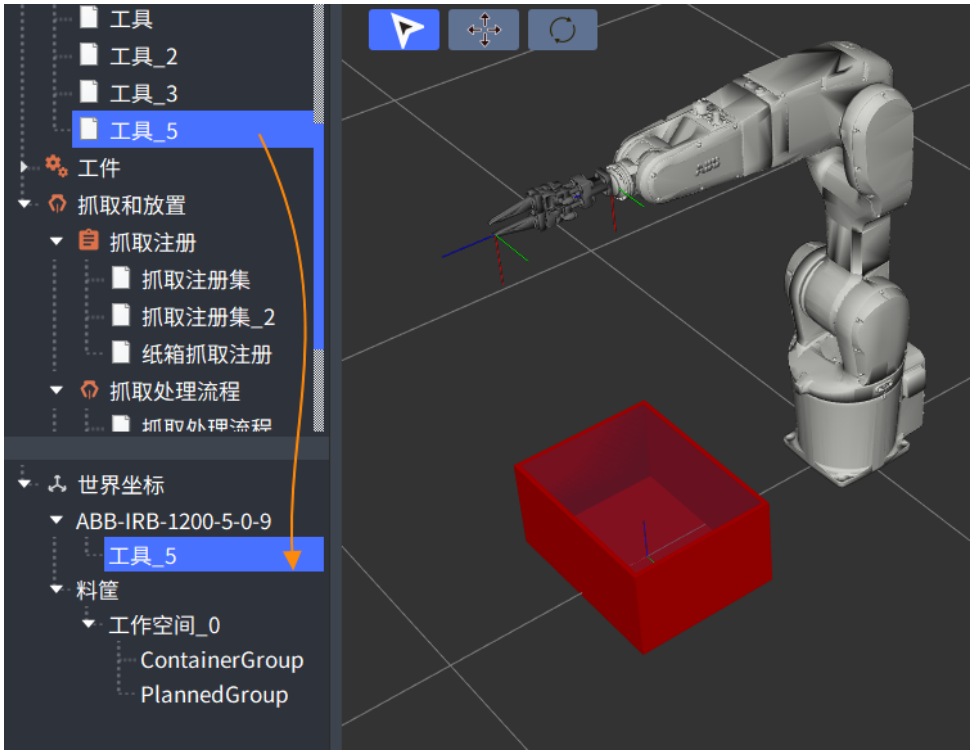
h. 单击 **对称性设置**下方 *新增*，通过旋转或平移变换遍历生成抓取位姿，单击界面右下角 *完成*。

使用虚拟模型注册抓取

a. 在 *运动*页签下快捷工具栏，单击 *末端工具*，选择 **注册工具**，再添加工具形态。碰撞模型使用 STL 模型，并添加工具参考点。请参见*注册工具*。如果在创建项目时已添加工具的 STL 模型，则只需配置工具参考点。

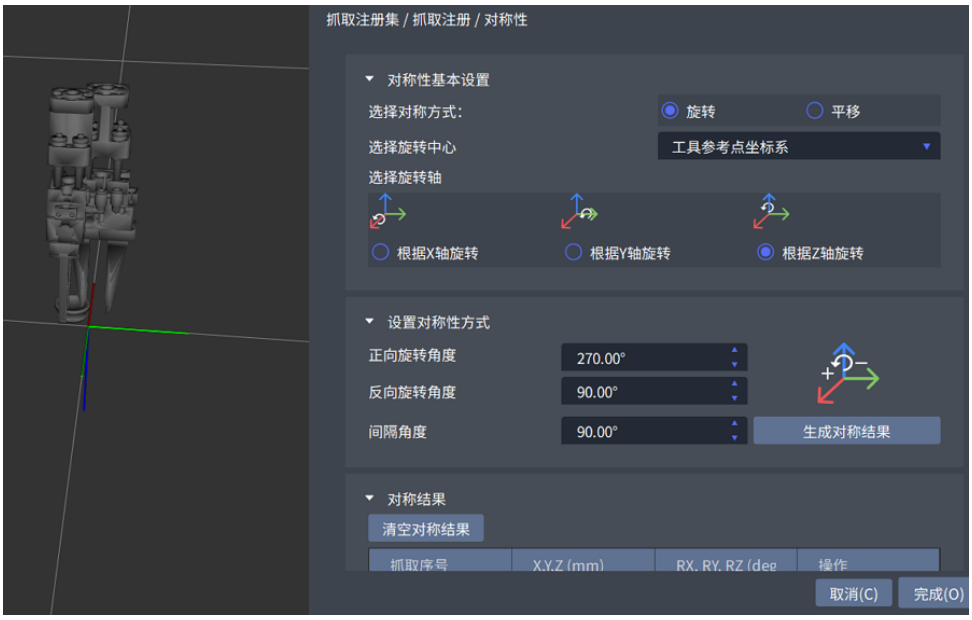
b. 单击中间窗口上方的 **环境**页签，将左侧环境编辑面板中的工具文件拖放到左下方机械臂

模型节点下。



c. 在顶部快捷工具栏，单击 工件，选择 **注册工件**。使用 STL 模型注册工件，请参见注册工件。如果在创建项目时已添加工件的 STL 模型，可跳过此步骤。

d. 在顶部快捷工具栏，单击 抓取和放置，选择“注册抓取方式 > 工件抓取注册”。使用已注册的工具和工件，设置抓取注册集，请参见注册抓取方式。



4. 注册抓取处理流程，参见注册抓取处理流程。

   a. 在顶部工具栏，单击 抓取和放置，选择 "注册抓放规划 > 添加抓取处理流程"。

   b. 单击界面左下角 **抓取处理流程**节点，在右侧参数设置栏选择前处理策略、抓取注册集、后处理策略。配置完成后，单击 下一步。

   c. 选择抓取工作空间，单击 测试运行，查看运行结果。确认无误后，单击右下角 完成。

10.4 任务

设置完视觉和运动部分之后、运行任务流图之前，需要添加工件代号映射关系，以便任务流图调用视觉和运动设置的工件、视觉服务编号、抓取处理流程等内容，具体步骤如下。

1. 单击 Max 右上角 映射表与通信协议，在弹出的 **映射表与通信协议**窗口选择“工件代号映射表 > 坐标移动”页签。

2. 单击 添加映射关系，在弹出的 **添加映射关系**窗口设置键、值和参数。

- 工件代号：用户自定义工件代号，仅支持英文字母和数字，长度不超过 15 个字符。可添加多个工件。
- 视觉服务编号：在下拉菜单选择视觉服务编号，编号和视觉部分的设置保持一致。
- 视觉流图文件：与视觉空间 ID 下的流图保持一致。
- 处理流程：在下拉菜单中选择抓取处理流程，这些抓取处理流程已在运动部分设置好。运动流程编号从 0 开始自动增加，可单击 或 新建或删除处理流程。
- 缓存多个视觉结果：可根据需要是否勾选。

3. 单击 确认完成添加，列表中会显示已添加的映射关系，用户可对列表中的映射关系进行操作。

-  ：修改映射关系。
-  ：复制并修改映射关系，注意键值不能重复。
-  ：删除映射关系。

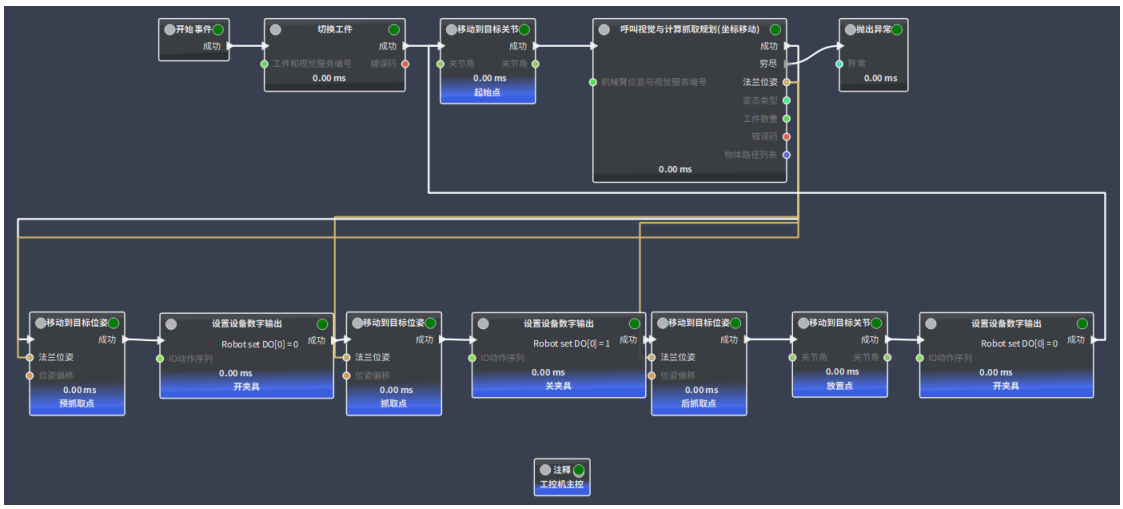
4. 单击 确认完成映射表设置。

提示： 用户可单击 自动生成映射关系，系统自动根据视觉和运动的设置生成工件代号映射表，但有多个抓取处理流程时无法自动生成映射关系。

设置完工件代号映射表之后，在 任务中导入 2D/3D 坐标移动中的基础模板，请参见流图配置。任务流图的设置分为工控机主控和机械臂主控两种方式。

10.4.1 工控机主控

添加好基础模板之后，用户可根据需求添加或修改模块。以抓取圆环项目为例，配置完成后的流图如下。



下面详细介绍该任务流图的配置步骤。

1. 在任务流图模块当中选择“功能 > 一键更新节点机械臂属性”，系统自动根据当前设置更新机械臂路径。也可在模块的属性中设置机械臂路径。

2. 在 **切换工件**模块中设置工件代号和视觉服务编号，其它参数可根据需求修改。该工件代号在映射表中设置，请参考本页面开头的内容。

3. 在 **移动到目标关节**模块中设置关节角度、位置和速度等参数，指定一个起始点位置。

4. 在 **呼叫视觉与计算抓取规划（坐标移动）** 模块中设置工作空间编号、机械臂路径、工作空间和机械臂位姿等参数，其它参数可根据需求修改，该模块会计算出最终的抓取位姿。
5. 设置目标关节和数字输出。
    a. 根据实际需要，设置预抓取点（prepick）、后抓取点（postpick）的位姿偏移，分别表示抓取前和抓取后机械臂停留的位置。例如在本项目中，该位姿是相对于抓取位姿的偏移量。模块中参考点路径为实际环境中工具参考点的路径。

b. 在预抓取点之后添加一个 **设置设备数字输出**模块。单击该模块，在右侧 **属性**页签，设置端口编号和值，此时值变为 0，表示工具张开。

提示： 单击 Max 右上角的 虚拟示教器，在 **I/O 控制**页签根据机械臂实际情况设置数字输出端口号。例如在本项目中端口 3 控制的是工具的开合，信号 1 表示工具闭合，信号 0 表示工具张开。端口名称可自行定义，然后单击 确认。在 **设置设备数字输出**模块设置时，选择 I/O 控制里的端口即可。

c. 在抓取点（pick）之后添加一个 **设置设备数字输出**模块。单击该模块，在右侧 **属性**页签，勾选 **值**，此时值变为 1，表示工具闭合。

d. 添加两个 **移动到目标关节**模块，分别表示后抓取点（postpick）和放置点（place），并在 **属性**页签中设置在这些点的目标关节角。

e. 在放置点之后添加一个 **设置设备数字输出**模块。单击该模块，在右侧 **属性**页签，勾选 **值**，此时值变为 0，表示工具张开。

f. 从流图末尾引出连线至表示起始点的 **移动到目标关节**模块（切换工件模块之后），以设置抓取多个工件的动作循环，系统会自动根据视觉计算结果判断是否有待抓取工件。

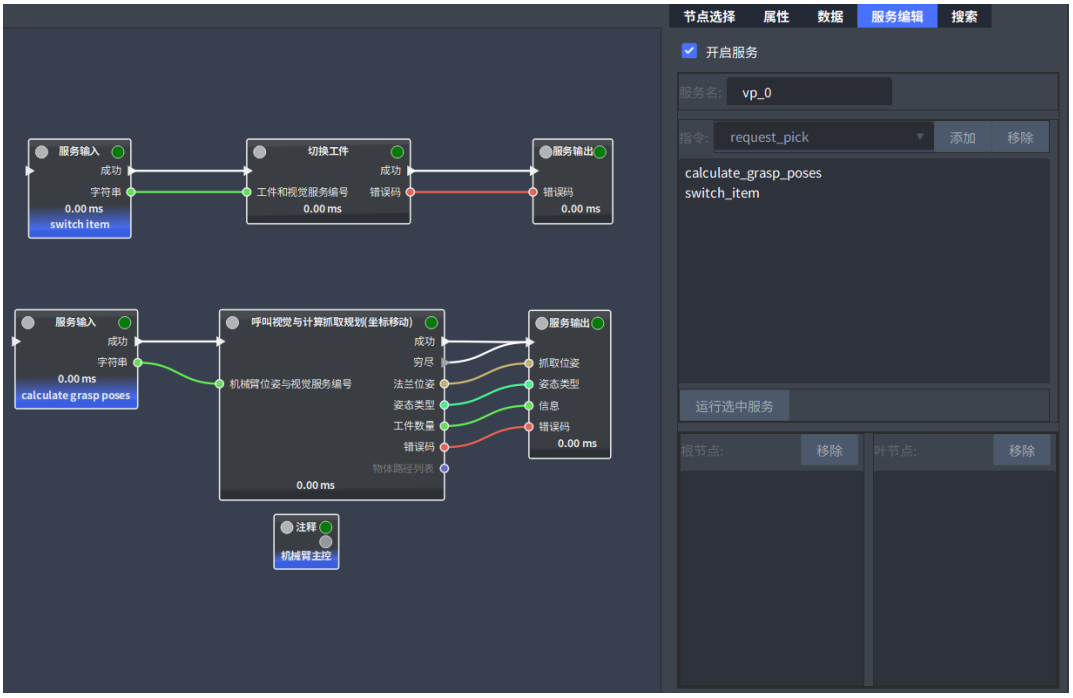
8. 单击任务流图左上角的  ，开始执行整个抓取流程。

10.4.2 机械臂主控

1. 在 任务界面右侧 服务编辑页签下，单击服务指令，编辑相应模块。

   在 2D/3D 坐标移动场景中，常见的服务如下：

   - switch_item：切换工件。
   - calculate_grasp_poses：计算抓取位姿。



2. 选择 *开启服务*，开启服务。工控机收到机械臂请求后，调用相应服务。

机械臂主控模式下，ABB 机械臂案例和示例代码可参考*案例/模板说明*，其他品牌的案例和代码可在*安装机械臂驱动* 中选择相应的品牌查看。

在 Max 中，单击右上方 *映射表与通信协议*，在弹出窗口的 *机械臂基础设置*页签下，勾选 **开启**打开测试模式。此时右侧会出现 *打开文件*和 。单击 *打开文件*可查看通信指令的 yml 文件，单击 可在浏览器中打开*机械臂主控通讯协议 v2* 页面，并查看机械臂主控通信协议、指令等的内容。

CHAPTER 11

3D 轨迹移动

通过精准视觉识别、稳定高速抓取、极速应用设置，Max 可有效应对稳定性要求高、作业环境差、强度大的 3D 轨迹移动场景。

本章以三个不同的项目为例，介绍 3D 轨迹移动场景的配置。在配置该场景前，可先登录 XYZ Studio Max 资源中心，下载对应场景的资源包。资源包的使用方法，包括虚拟相机的配置，可参见*2D/3D 坐标移动*。

11.1 蝴蝶件抓取

本章以一个简单的蝴蝶件抓取项目为例，介绍 3D 轨迹移动场景的配置。在本项目中，蝴蝶件工件乱序摆放在料筐内，相机安装在机械臂上，要求使用 KUKA-KR-6-R900-sixx 机械臂，配合夹具，将工件抓取并放入另一个料筐中。配置教程如下：

11.1.1 新建项目

根据项目场景，选择模板，创建项目。操作可参考新建项目。

1. 在 Max 顶部工具栏，选择“文件 > 新建项目”。
2. 选择场景模板，在本项目中，选择 **3D-轨迹移动-眼在手上**模板。
3. 添加机械臂，本项目使用 KUKA-KR-6-R900-sixx 机械臂。
4. 选择 **高级参数**后，添加夹具和工件模型，其他参数保持默认。也可不在新建项目时设置高级参数，待项目创建完成后，在 运动、视觉、任务页签下再作设置。
5. 单击 确认。

11.1.2 视觉

本章节将针对具体场景详细介绍视觉服务的配置过程。在进行视觉配置前，请先参见连接机械臂，完成机械臂连接。

场景要求

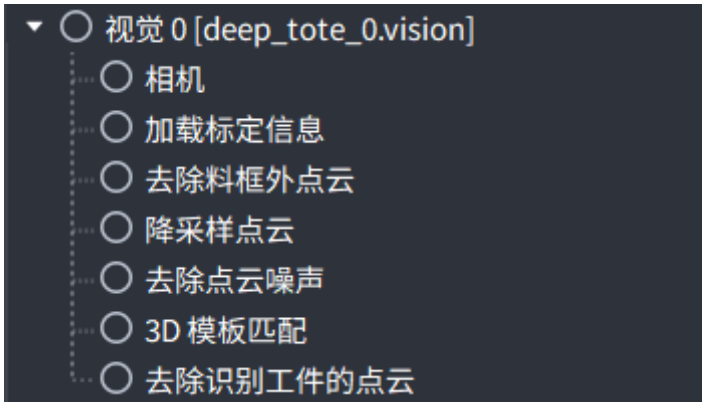
蝴蝶件抓取的场景采用 **3D 深筐乱序抓取**模板，该流图适用于工作空间内同种物体乱序摆放，且拥有物体 STL 模型的情况。目前广泛应用于汽车零件、机加工件 等多种场景。

- 相机为结构光相机
- 需要计算点云碰撞

本章节描述的相关配置仅供参考，具体操作请以实际项目为准。

基本配置

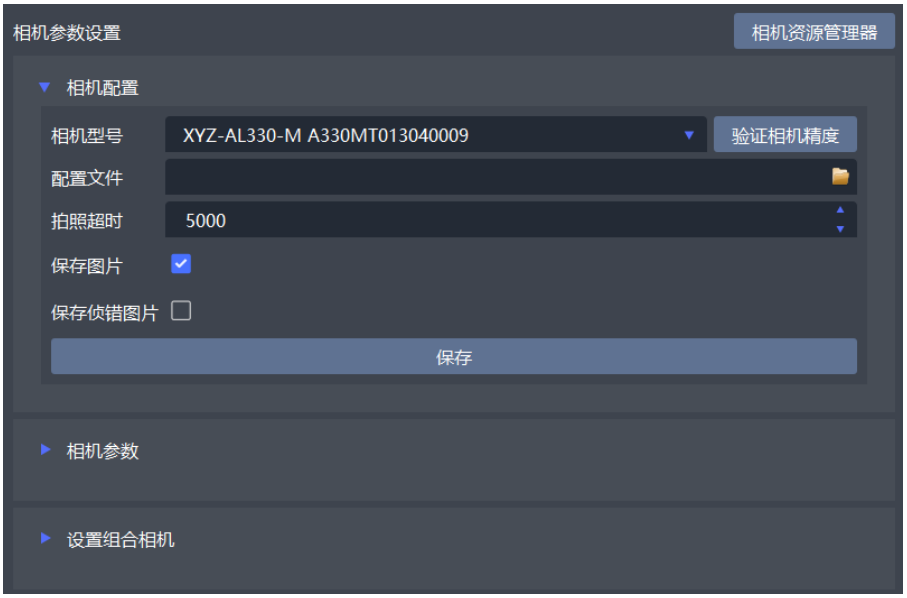
配置前，请依次完成新建项目、添加工作空间，并使用 **3D 深筐乱序抓取**模板创建视觉服务，请参见创建流图。创建完成之后会在左上角生成节点树，如下图所示。



根据需求，依次配置这些模块。

步骤 1. 配置相机模块

1. 单击左侧节点树中的 **相机**模块，然后在右侧单击 相机资源管理器以连接相机，接着在 **相机配置**区域设置相机基本信息，请参见连接相机。
2. 在 **相机参数**区域单击 验证相机精度。



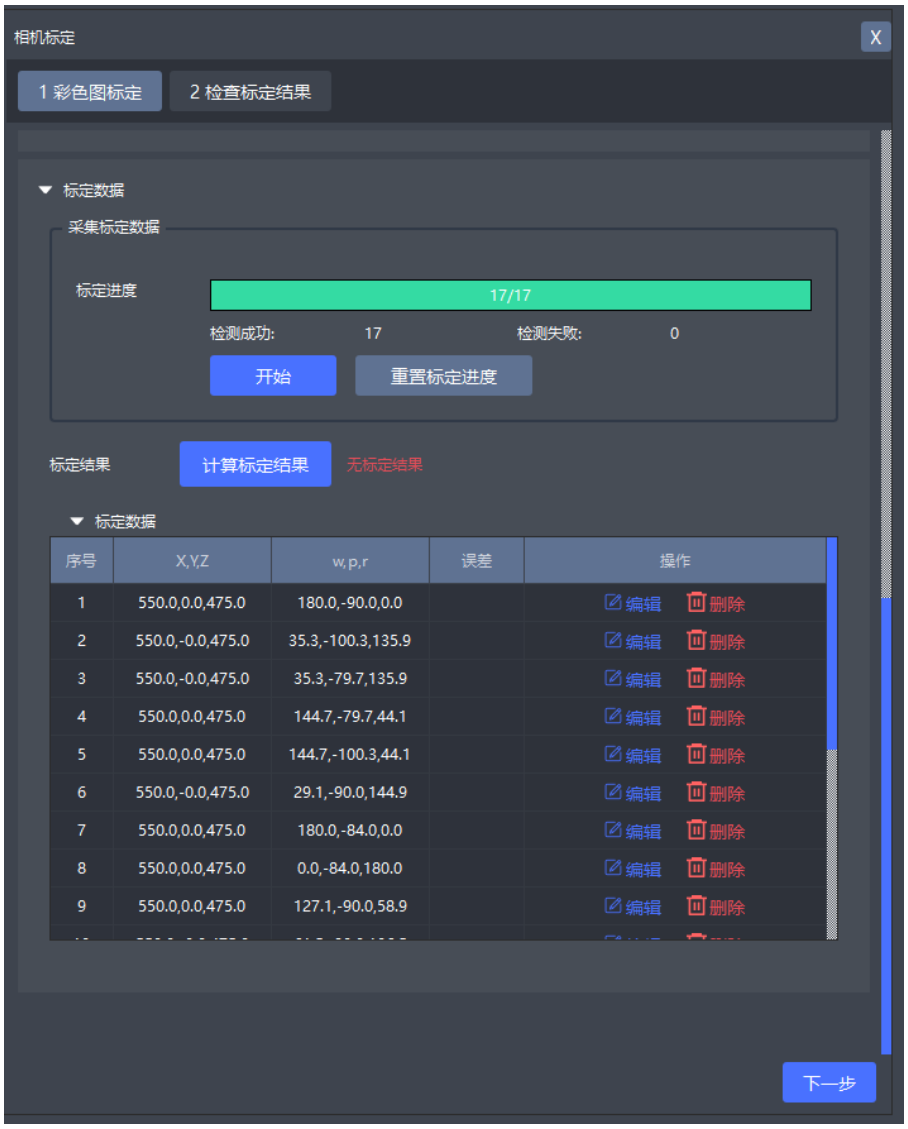
3. 依次选择标定板、标定内参数、验证标定结果等步骤，以验证相机精度。

步骤 2. 配置加载标定信息模块

1. 单击左侧节点树中的 **加载标定信息**模块，然后在右侧依次标定相机和工作空间。

2. 单击 **相机标定**区域中的 新建标定数据，设置标定数据名称，然后在 相机标定页签设置标定方法和其它标定信息，请参见手眼标定 。

3. 依此单击 开始和 计算标定结果完成数据标定，然后单击 下一步进入 **检查标定结果**页签。



4. 检查并根据界面提示确认标定误差后，单击 **完成**，即可完成相机标定。

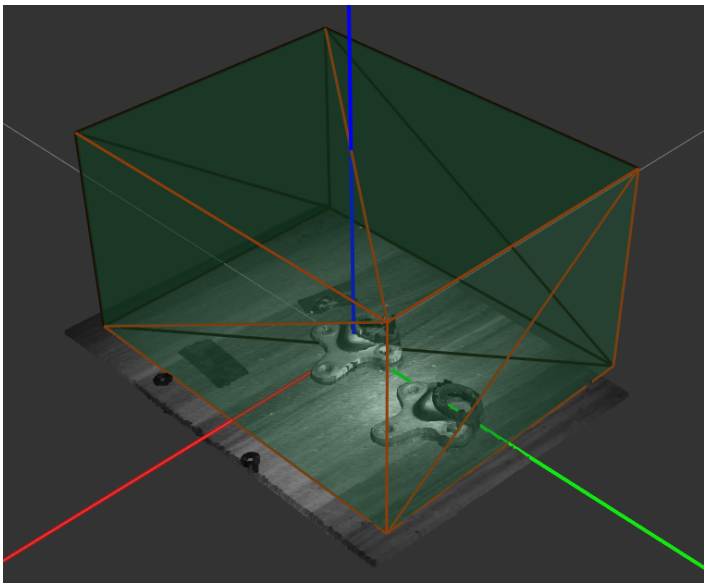
5. 若要提高精度，则勾选 点云标定，完成点云标定，请参见手眼标定 中的步骤 5 点云标定的相关内容。
6. 标定工作空间，请参见工作空间标定。3D 相机使用 3D 拖拽的方式标定工作空间。

步骤 3. 配置去除料筐外点云模块

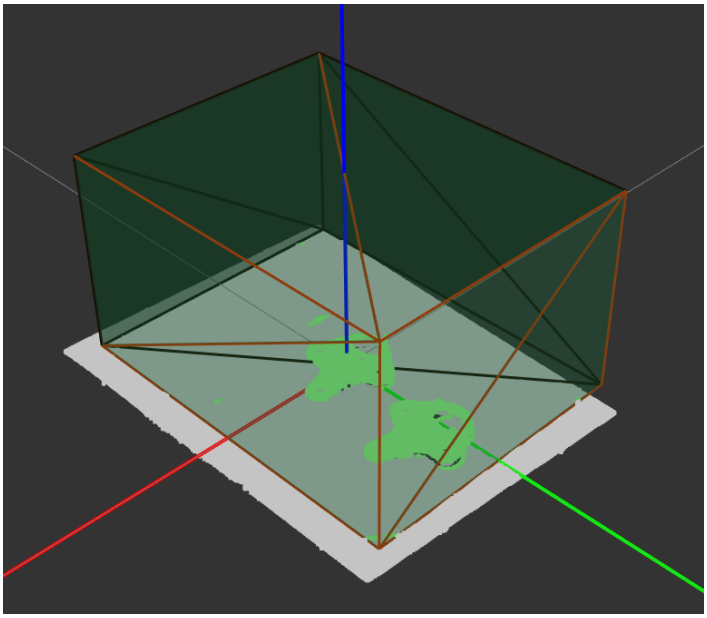
1. 单击左侧节点树中的 **去除料框外点云**模块，然后在右侧配置点云切割参数，然后单击预览窗口中的 触发。

- X 方向收缩长度：料筐沿 X 方向的收缩长度，负值则意味着扩张。
- Y 方向收缩长度：料筐沿 Y 方向的收缩长度，负值则意味着扩张。
- 底部抬升高度：料筐底部抬升高度，负值则意味着下降。
- 顶部下降高度：料筐顶部下降高度，负值则意味着抬升。

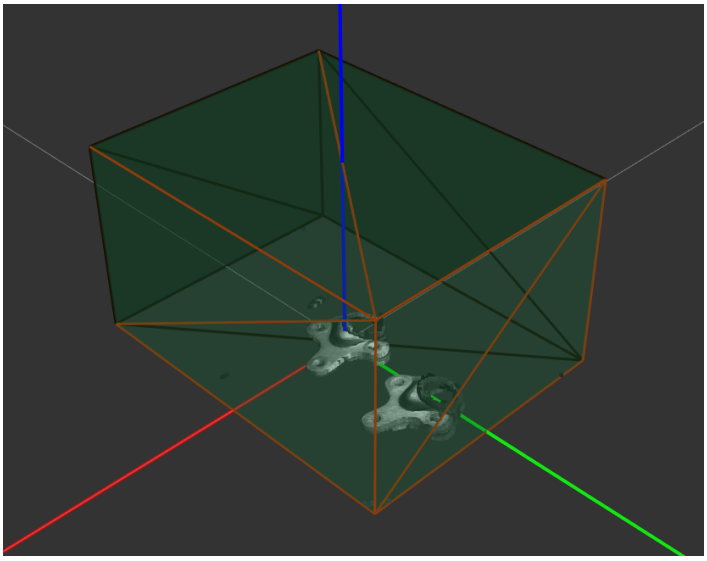
2. 在预览视图中移动料筐模型，使之与点云中的料筐位置吻合。
3. 在 **可视化显示**区域选择需要查看的图像。
   - 未切割点云图：仅显示未去除料筐外点云的图像。



- 切割对比图：对比显示切割前后的图像，被切割的点云用白色显示。

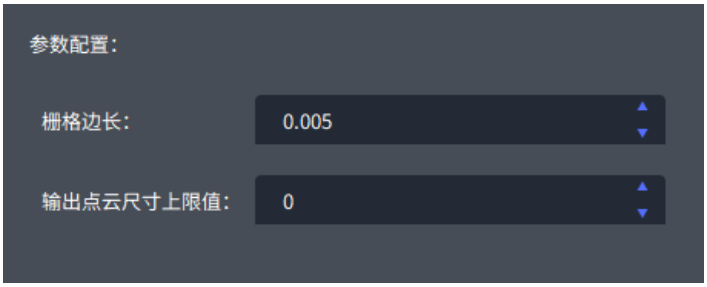


- 切割完点云图：仅显示去除料筐外点云的图像。



步骤 4. 配置降采样点云模块

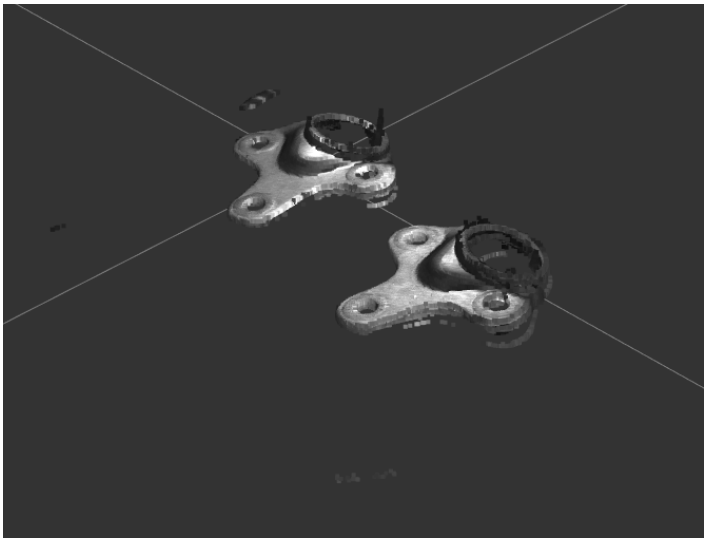
1. 单击左侧节点树中的 **降采样点云**模块，然后在右侧配置降采样参数。



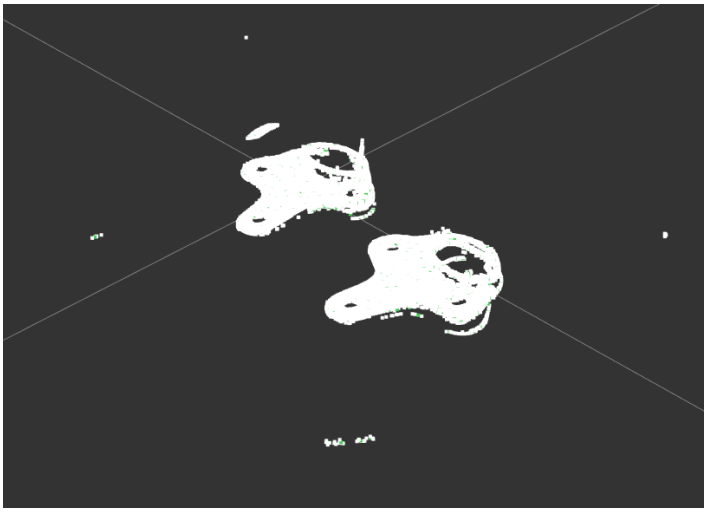
- 栅格边长：将点云中的点划分到指定大小的三维栅格（即正方体）中，栅格边长越大，输出的点云越稀疏。

- 输出点云尺寸上限值：如果该值大于 0，将会进行两次采样，第二次降采样操作将会基于第一次进行。

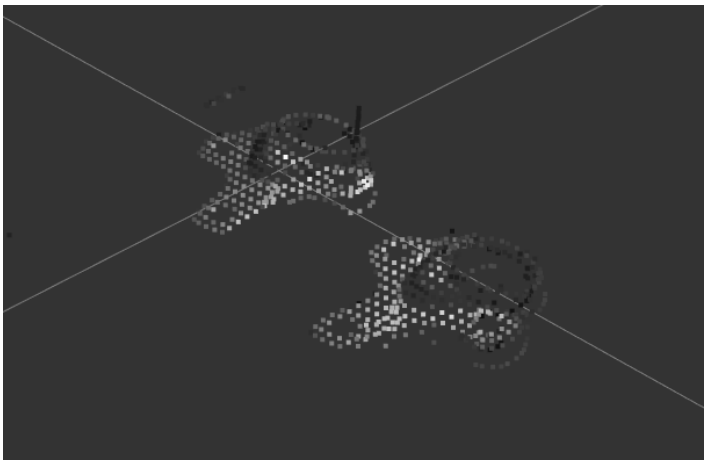
2. 在 **可视化显示**区域选择需要查看的图像。
   - 降采样点云前：仅显示降采样前的图像。



   - 降采样点云对比图：对比显示降采样前后的图像，被去除的点云部分用白色显示。



   - 降采样点云后：仅显示降采样后的图像。



步骤 5. 配置去除点云噪声模块

1. 单击左侧节点树中的 **去除点云噪声**模块，然后在右侧配置去噪声参数，噪声即离群点。过滤方法分为 Statistical 和 Radius 两种。

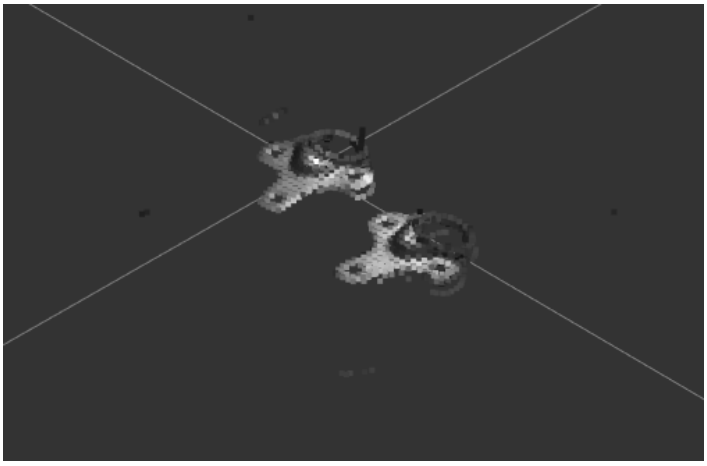
   Statistical 是基于统计的离群点过滤，运行时间比基于距离的离群点过滤方法更久，其参数配置如下：

   - 标准差倍数：过滤掉与最邻近点平均距离大于平均值 + 倍标准数 * 标准差以外的点，该值越小则过滤点越多。
   - 用于统计的临近点个数：该值越大则统计越准确，但计算时间更久。

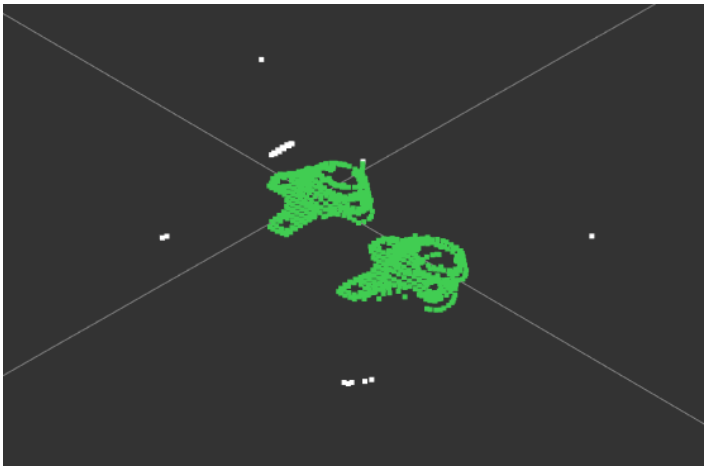
   Radius 表示基于距离的离群点过滤，建议经验丰富的工程师使用，其参数配置如下：

   - 临近点搜索半径：统计临近点数量时，只考虑该半径范围内的点，该值越大则计算时间越长。
   - 最小临近点个数：如果一个点附近的临近点数量小于该值，则会被视为噪声过滤，该值越大则过滤强度越高。

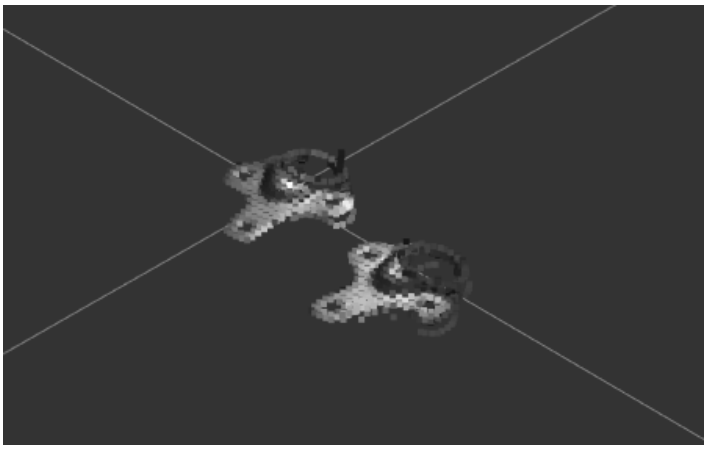
2. 在 **可视化显示**区域选择需要查看的图像。
   - 去除点云噪声前：仅显示去除点云噪声前的图像。



- 去除点云噪声对比图：对比显示去除噪声前后的图像，被去除的点云用白色显示。

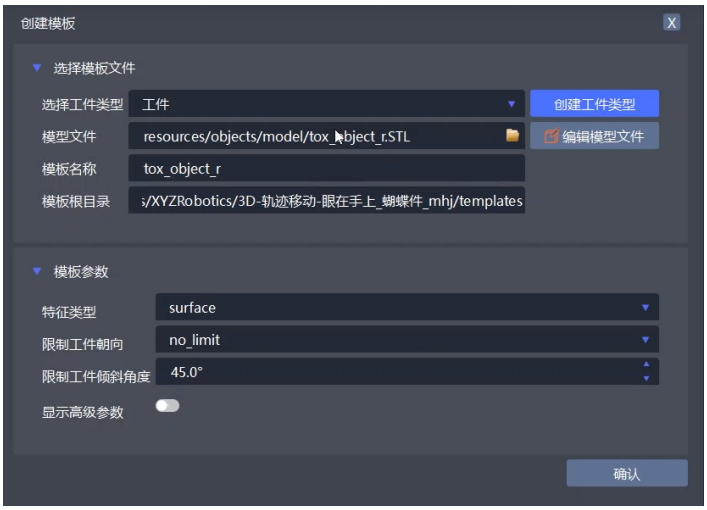


- 去除点云噪声后：仅显示去除点云噪声后的图像。



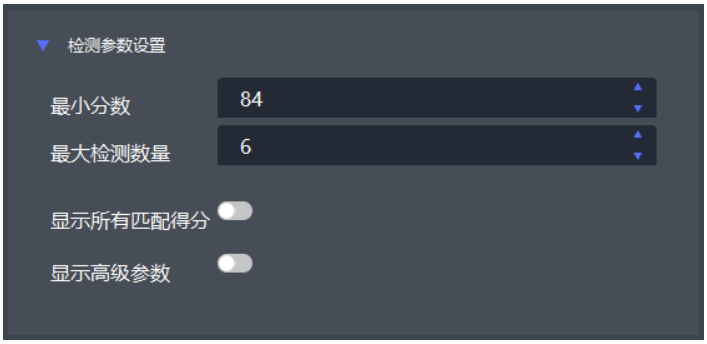
步骤 6. 配置 3D 模板匹配模块

1. 单击 创建模板，选择一个已有的或者新建一个工具类型，新建工具类型时不可重名，且工件类型必须和注册抓取时使用的工件名称一致。然后选择模型文件，并给模板命名即可。需要修改模型时，可单击 编辑模型文件编辑模型。



提示: 如果已有模板，单击 选择模板，直接选择包含模型的文件夹即可。

2. 根据需要选择 特征类型，如果模型表面整体较为扁平，且平面和直线较多，建议尝试 edge-I/edge-II 模式。限制物体朝向建议使用 no_limit，其作用是增加可抓取的朝向和角度，提升抓取概率。
3. 根据需要设置检测参数。模板匹配时，小于 最小分数的结果将被过滤。超过 最大检测数量的结果，也将被过滤。

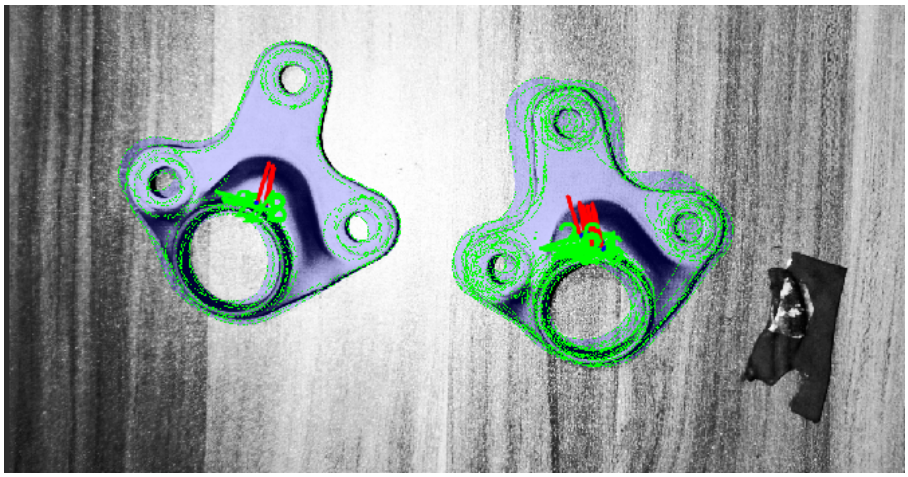


4. 设置不同检测阶段的参数，系统会根据物体模型在点云中寻找匹配结果。

用户也可以查看分步执行的检测结果，各阶段参数设置如下。

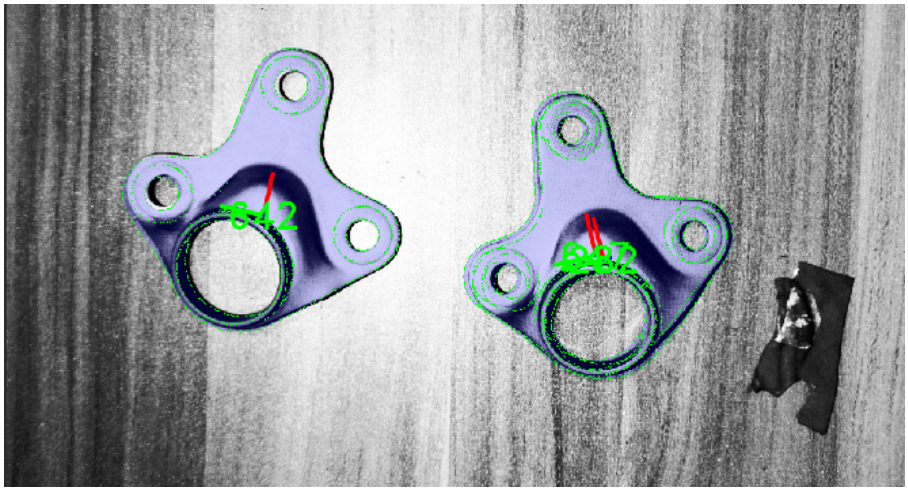
- 通过 **投票**阶段，点云中的每个关键点都会得到一个最可能对应的模型关键点和对应的位姿结果，该阶段各参数配置如下。

该步骤运行结果如下图所示。



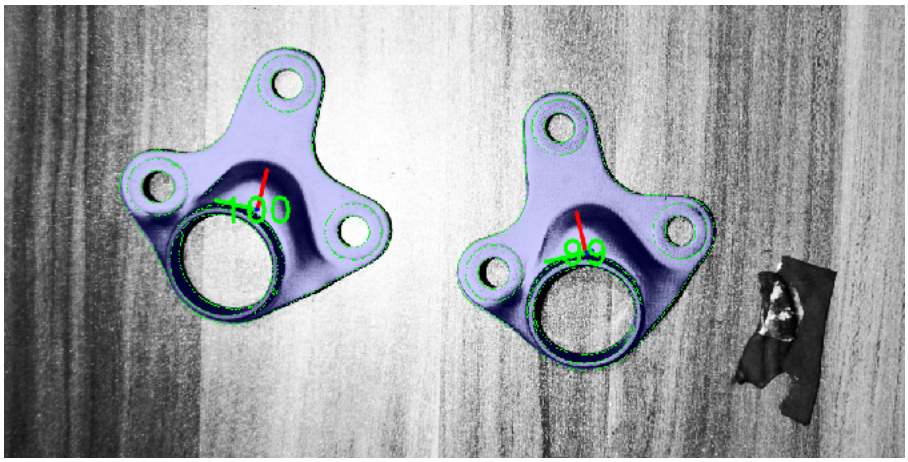
- 通过 **聚类**阶段，合并相近的位姿结果，去除得票数较少的结果，该阶段各参数配置如下。

该步骤运行结果如下图所示。



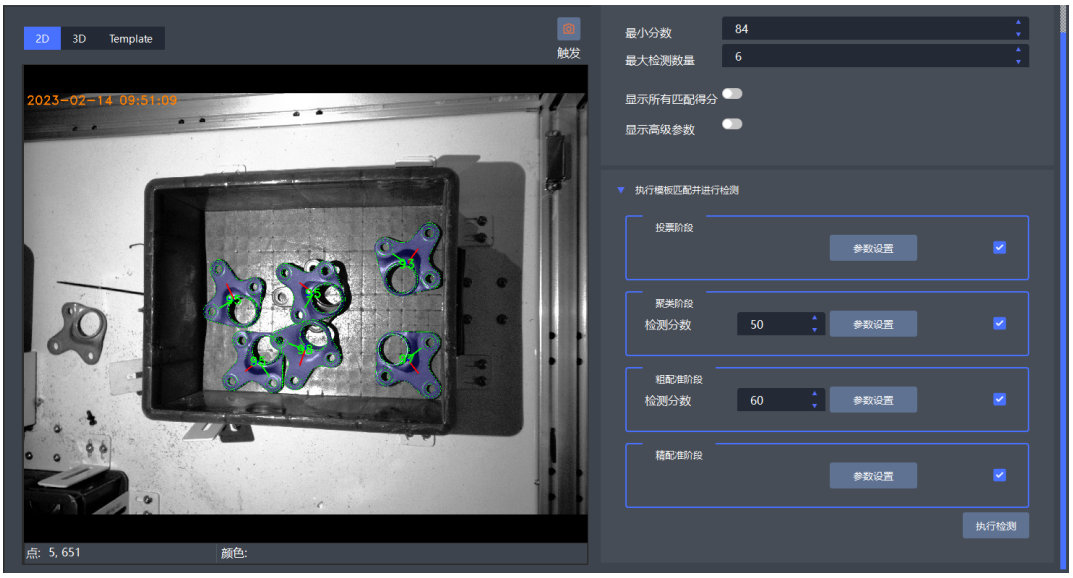
- 通过 **粗配准**阶段，使用 ICP 迭代，优化结果的精度、去除误差较大的结果，该阶段各参数配置如下。

该步骤运行结果如下图所示。



- 通过 **精配准**阶段，使用 ICP 迭代，优化结果的精度、去除误差较大的结果，该阶段各参数配置如下。

5. 单击 执行检测，然后在预览区域查看检测结果。



步骤 7. 配置去除工件附近的点云模块

1. 单击左侧节点树中的 **去除工件附近的点云**模块，然后在右侧配置参数。

- 最大去除半径：工件表面附近一定半径内的点将被去除，通常该值为 2~10 毫米。
- 去除长方体 primitive 点云时的 XY 阈值：去除 primitive 长方体内部点云时使用的 XY 阈值。
- 去除长方体 primitive 点云时的 Z 阈值：去除 primitive 长方体内部点云时使用的 Z 阈值。

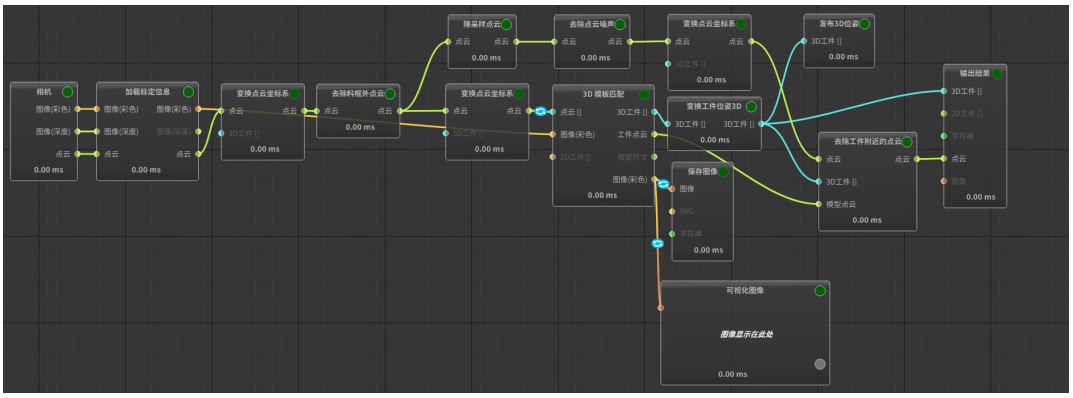
2. 单击 运行，然后在预览区选择输入、输出，查看处理后的点云图像。单击 保存可以存储图像。

流图详解

视觉流图的配置是 Max 中的高级用法，对于一般用户而言，配置完节点树中的模块即可运行整个视觉流程，无需再配置视觉流图。视觉流图中包含了节点树中没有的模块，可以配置一些高级参数，如有需要可以按照以下内容操作。

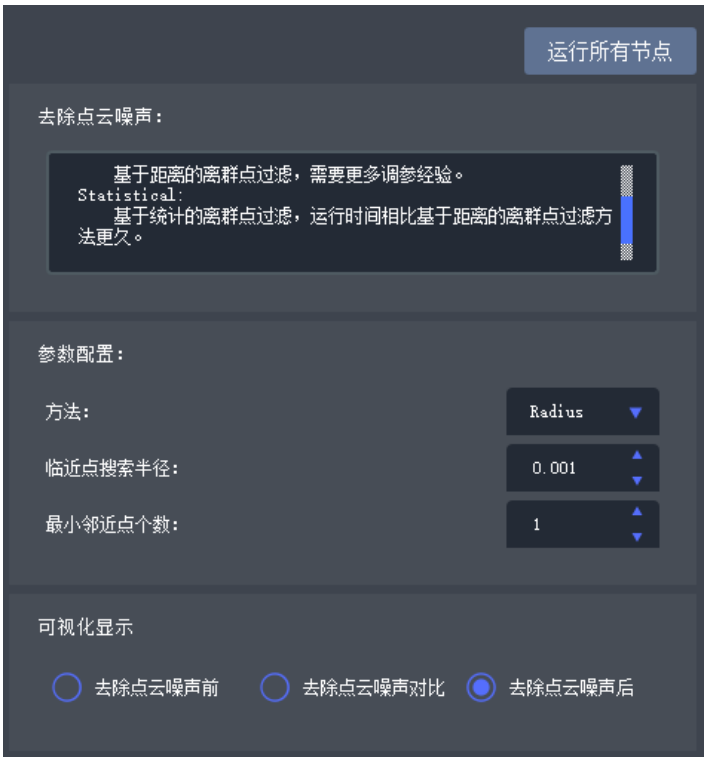
3D 深筐乱序抓取场景的标准视觉流图配置如下。

1. 创建工作空间，使用 **3D 深筐乱序抓取**模板创建流图，请参见创建流图。创建成功的视觉流图如下。



2. 配置 **相机**模块，获取彩色图、深度图及点云。请参见连接相机。
3. 配置 **加载标定信息**模块，请参见标定。
4. 配置 **变换点云坐标系**和 **去除料筐外点云**模块，将点云从相机坐标系转换到工作空间坐标系，并去除料筐以外的点云。

5. 配置 **降采样点云**、**去除点云噪声**、**变换点云坐标系**模块，合理降低点云数量，去除噪声，并将点云从工作空间坐标系转换到世界坐标系。详细请参见基本配置 中的步骤 4、步骤 5。



6. 基于步骤 4 得到的料筐内点云，配置 **变换点云坐标系**模块，将点云从工作空间坐标系转换到相机坐标系。
7. 结合步骤 3 得到的彩色图，配置 **3D 模板匹配**模块，根据物体模型在点云中寻找匹配结果。
8. 配置 **变换工件位姿 3D** 模块，将物体位姿从相机坐标系转换到世界坐标系。
9. 基于步骤 5 得到的世界坐标系下料筐内点云、步骤 6 中 3D 模板匹配得出的模型点云，以及步骤 8 得到的世界坐标系下物体位姿，配置 **去除识别工件的点云**模块并运行。
10. 配置 **输出结果**模块，输出步骤 8 得到的世界坐标系下物体位姿，及步骤 9 得到的物体点云。
11. 重复上述步骤，完成其它料筐的视觉流图的设置。

11.1.3 运动

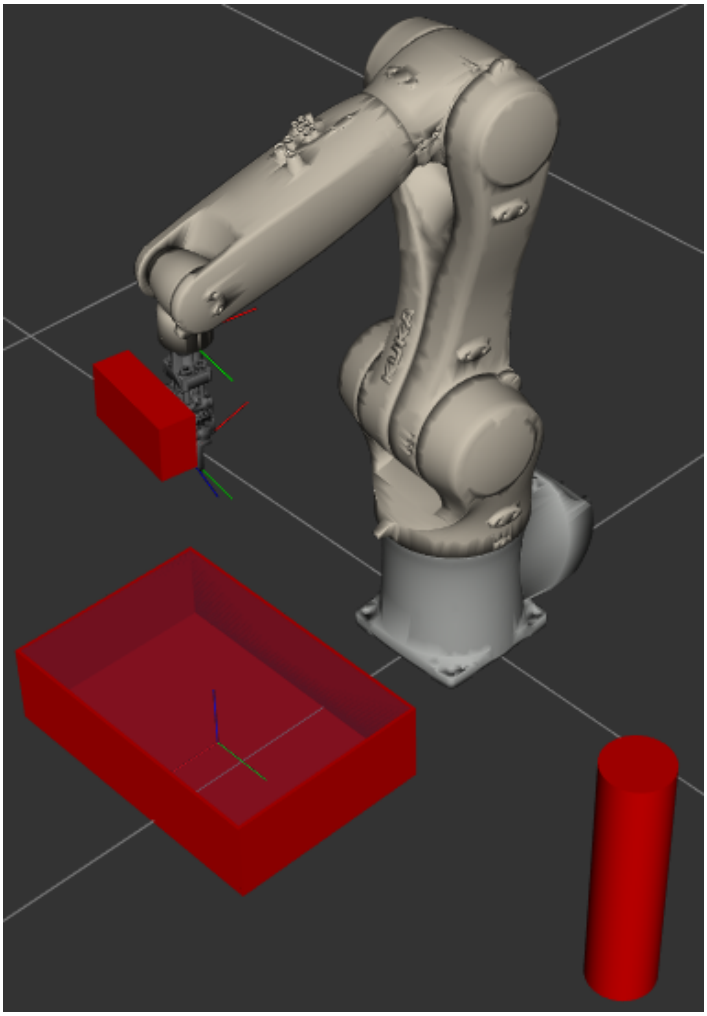
在 *运动*界面，配置仿真环境并规划运动路径。

提示： 在开始设置运动相关内容前，请先根据机械臂和相机品牌连接并配置真实环境中的机械臂和相机。相关内容请参考连接机械臂 和设置相机 *IP*。如果仅在仿真环境中运行，则无需配置机械臂和相机。

配置仿真环境

根据实际场景，在仿真环境中添加机械臂、料筐、几何体、工具。

1. 创建项目时，已添加机械臂。如未添加，可在 运动 页签下快捷工具栏，单击 机械臂，添加机械臂，请参见添加机械臂。
2. 创建项目时，已添加料筐。可在顶部快捷工具栏，单击 工作空间，选择 **添加料筐**，继续添加料筐。本项目需添加两个料筐，添加料筐后需设置料筐及工作空间的尺寸、位姿等参数，请参见添加料筐。
3. 创建项目时，已添加工具，工具自动加载至机械臂末端。如未添加，可在 运动 页签下快捷工具栏，单击 末端工具，选择 **注册工具**，注册工具并将工具添加至机械臂，具体参见下一节。
4. 在顶部快捷工具栏，单击 几何体，根据实际场景，添加几何体，用于表示环境中的相机、障碍物等物体。

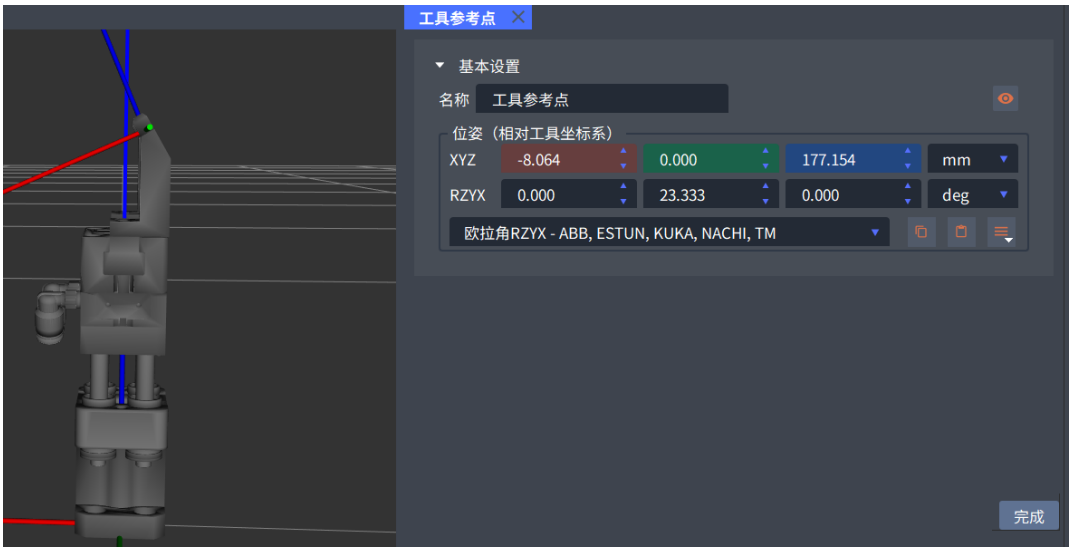


注册抓取

利用工具和工件，注册抓取方式和抓取处理流程。

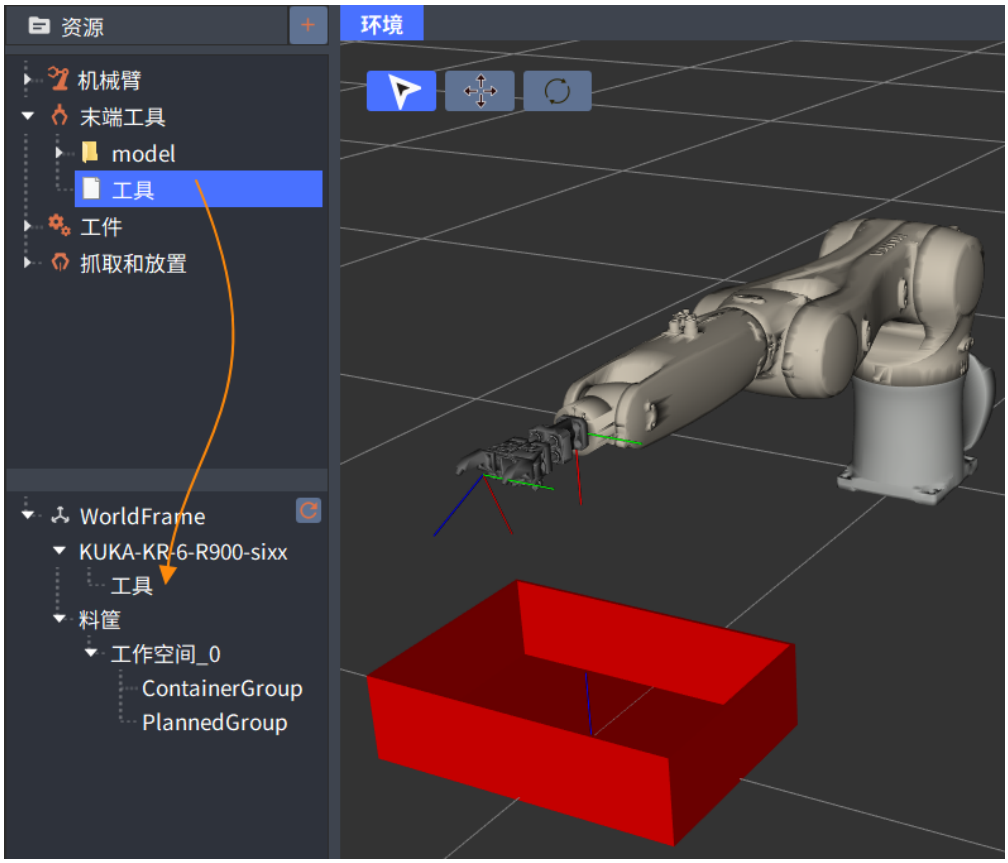
1. 注册工具。

a. 在 运动页签下快捷工具栏，单击 末端工具，选择 **注册工具**，再添加工具形态。碰撞模型使用 STL 模型，并添加工具参考点。请参见注册工具。



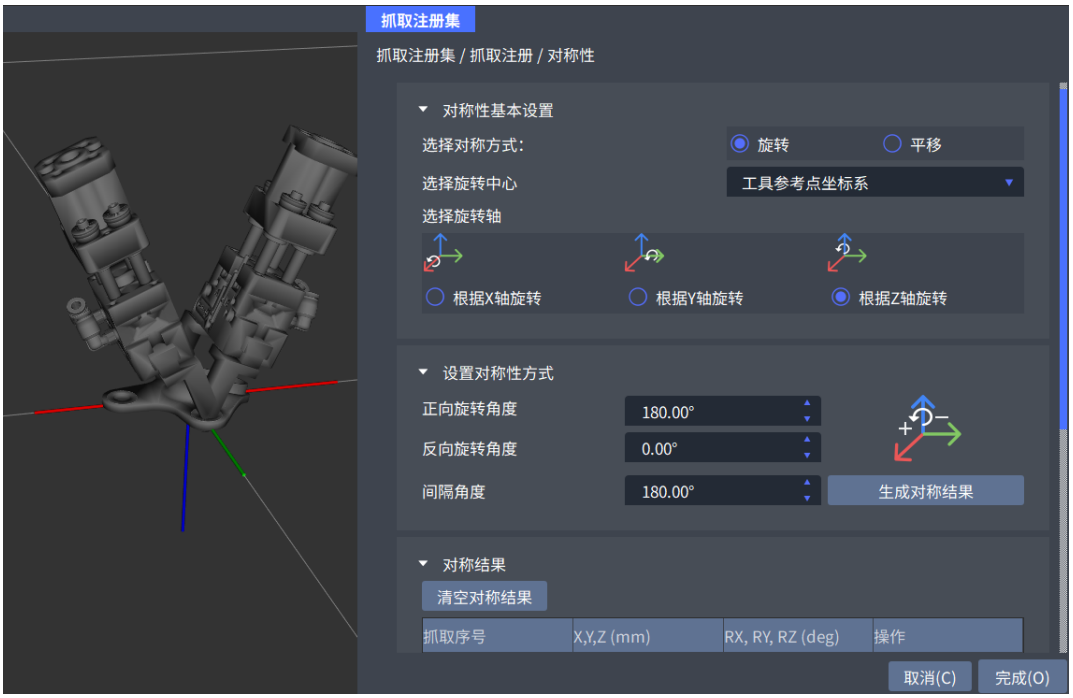
b. 单击中间窗口上方的 **环境**页签，将左侧环境编辑面板中的工具文件拖放到左下方机械臂模型节点下。

提示： 如果在创建项目时已添加工具的 STL 模型，则只需配置工具参考点。



2. 注册工件。在顶部快捷工具栏，单击 工件，选择 **注册工件**。使用 STL 模型注册工件，请参见注册工件。如果在创建项目时已添加工件的 STL 模型，可跳过此步骤。

3. 注册抓取方式。在顶部快捷工具栏，单击 抓取和放置，选择 “注册抓取方式 > 工件抓取注册”。使用已注册的工具和工件，设置抓取注册集，请参见注册抓取方式。



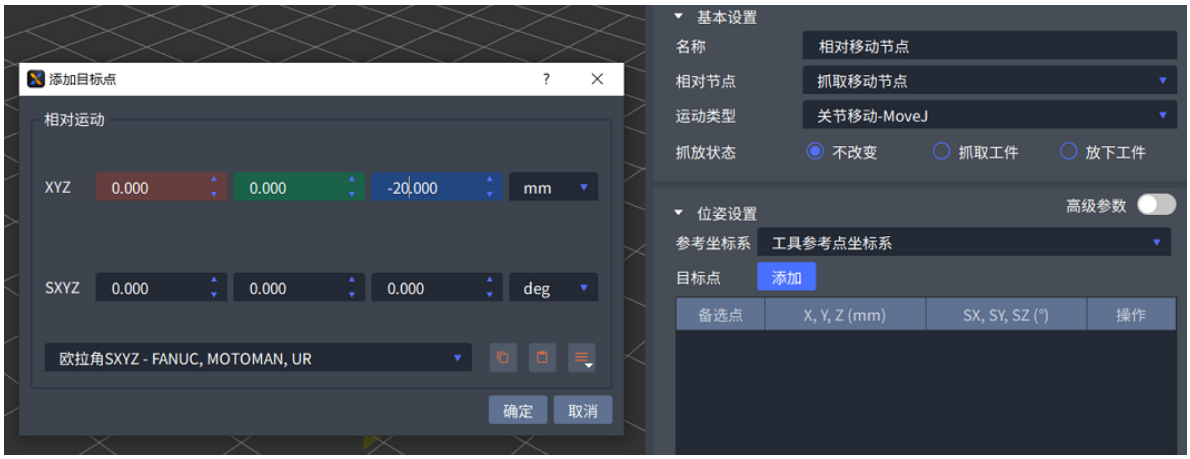
4. 注册抓取处理流程。在顶部快捷工具栏，单击 抓取和放置，选择 “注册抓放规划 > 抓取处理流程”，设置前后处理策略并添加抓取注册集。请参见注册抓取处理流程。

配置运动路径

添加点位，配置机械臂运动路径。一般需添加抓取点、放置点以及相对于抓取点、放置点的点位。

1. 在 *运动*页签下的 **路径**面板，单击右上角  ，选择 **添加路径**新建路径，或导入预置模板，本节以新建路径为例。
2. 在右侧参数设置栏 **抓取配置**下方，设置抓取移动涉及的工作空间、处理流程、视觉服务编号，并根据需要在 **放置配置**下方设置放置相关参数。

3. 右键单击 **路径**面板左下角 **路径**，选择“添加移动节点 > 抓取移动节点”。单击左下角新生成的 **抓取移动节点**，在右侧参数设置栏将 **运动类型**设置为 **线性移动-MoveL**。
4. 右键单击 **路径**面板左下角 **抓取移动节点**，选择“添加移动节点在前 > 相对移动节点”。单击左下角新生成的 **相对移动节点**，在右侧参数设置栏将 **相对节点**设置为 **抓取移动节点**，单击 **目标点**右侧 *添加*，设置相对位姿。在本项目中，设置相对位姿时将 Z 值设为-20mm，表示该点位在抓取点上方 20mm 处。



5. 右键单击 **路径**面板左下角 **抓取移动节点**，选择“添加移动节点在后 > 相对移动节点”，再生成一个相对移动节点，代表机械臂在抓取工件后抬起的点位。单击该节点，将相对节点设置为 **抓取移动节点**，再添加目标点并设置相对位姿，在本项目中，将 Z 值设为-40mm。

6. 右键单击上一步生成的 **相对移动节点**，选择“添加移动节点在后 > 绝对移动节点”，用该节点表示放置点位。单击左下角新生成的 **绝对移动节点**，在右侧参数设置栏将 **抓放状态**设置为 **放下工件**，单击 **目标点**右侧 添加，添加放置点的机械臂关节角。

7. 确认各点位的碰撞检测设置。单击各节点，在右侧参数设置栏 **碰撞检测设置**下方查看碰撞检测参数。
   - 检查目标点位碰撞：检测该点是否发生碰撞。
   - 检查插值点位碰撞：检测从上一个点到该点之间插值点位是否发生碰撞。
8. 右键单击中间预览窗口上方 **路径**页签，选择 **保存**。

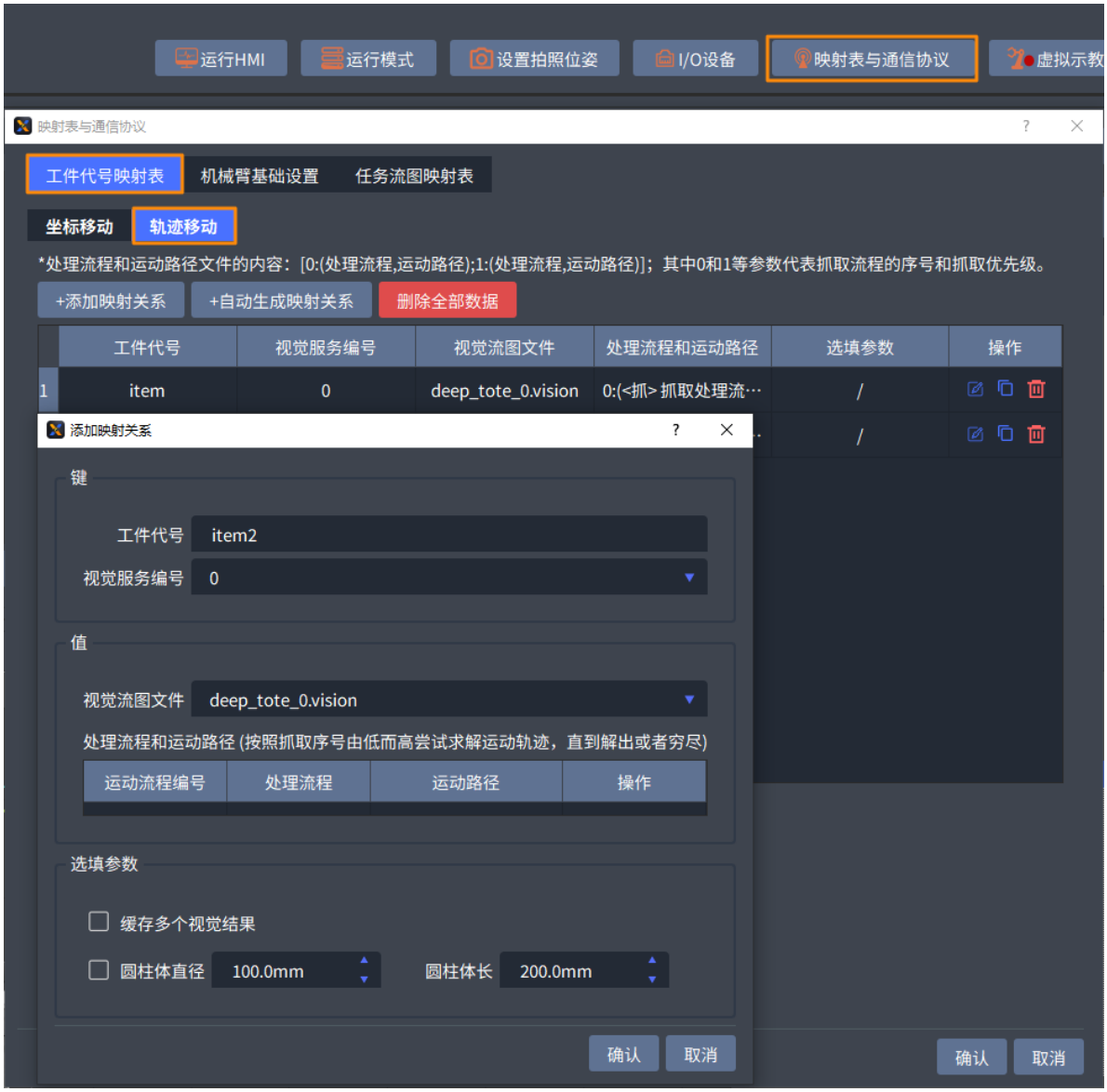
11.1.4 任务

任务部分利用视觉部分的计算结果和运动部分的运动规划，控制整个项目的流程运行。本章节将详细介绍任务部分的配置。

工件代号映射表

设置完视觉和运动部分之后、运行任务流图之前，需要添加工件代号映射关系，以便任务流图调用视觉和运动设置的工件、视觉服务编号、抓取处理流程等内容，具体步骤如下。

1. 单击 Max 右上角 *映射表与通信协议*，在弹出的 **映射表与通信协议**窗口选择“工件代号映射表 > 轨迹移动”页签。
2. 单击 *添加映射关系*，在弹出的 **添加映射关系**窗口设置键、值和参数。



- 工件代号：用户自定义工件代号，仅支持英文字母和数字，长度不超过 15 个字符。可添加多个工件。

- 视觉服务编号：在下拉菜单选择视觉服务编号，编号和视觉部分的设置保持一致。
- 视觉流图文件：与视觉服务编号下的流图保持一致。
- 处理流程：在下拉菜单中选择抓取处理流程，这些抓取处理流程已在运动部分设置好。运动流程编号从 0 开始自动增加，可单击  或  新建或删除处理流程。
- 运动路径：选择在 *运动*中设置好的一条路径。
- 缓存多个视觉结果：可根据需要是否勾选。
- 圆柱体：本项目无需勾选。与圆柱体相关的项目可根据需求，设置圆柱体的直径和长，无需重新注册圆柱体的抓取方式，方便使用。

3. 单击 *确认*完成添加，列表中会显示已添加的映射关系，用户可对列表中的映射关系进行操作。

-  ：修改映射关系。
-  ：复制并修改映射关系，注意键值不能重复。
-  ：删除映射关系。

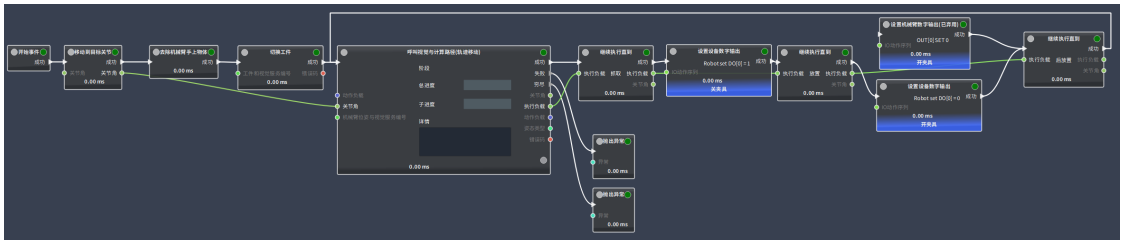
4. 单击 *确认*完成映射表设置。

提示： 用户可单击 *自动生成映射关系*，系统自动根据视觉和运动的设置生成工件代号映射表，但有多个抓取处理流程时无法自动生成映射关系。

设置完工件关系映射表之后，在 *任务*中导入 3D 轨迹移动中的模板，请参见流图配置。任务流图的设置分为工控机主控和机械臂主控两种方式，每种方式下含多个模板，可根据需要选择。在本章节工控机主控使用 **抓取同步（粗颗粒度）**模板，机械臂主控使用 **抓取异步前端**和 **抓取异步后端**模板。

工控机主控

添加好基础模板之后，用户可根据需求添加或修改模块。以抓取蝴蝶件项目为例，配置完成后的流图如下。



下面详细介绍该任务流图的配置步骤。

1. 在任务流图模块当中选择 “功能 > 一键更新节点机械臂属性”，系统自动根据当前设置更新机械臂路径。也可在模块的属性中设置机械臂路径。

2. 单击 **移动到目标关节**模块，在右侧 **属性**页签中设置目标关节角、位置和速度等参数，即起始点位置。

3. 单击 **呼叫视觉与计算路径（轨迹移动）**模块，在右侧 **属性**页签中设置法兰位姿，该位姿仅在“眼在手上”的场景中需要设置。

4. 分别单击两个 **继续执行直到**模块，在右侧 **属性**页签中将 **目标点名称**设置为与 *运动*中路径规划设置的点相同的名称，例如该项目中两个模块分别设为的“抓取”和“放置”。

5. 分别设置 **设置设备数字输出**模块。单击该模块，在右侧 **属性**页签，设置端口编号和值。值为 0，表示工具张开；值为 1，表示工具闭合。

提示： 单击 Max 右上角的 虚拟示教器，在 **I/O 控制**页签根据机械臂实际情况设置数字输出端口号。例如在本项目中端口 3 控制的是工具的开合，信号 1 表示工具闭合，信号 0 表示工具张开。端口名称可自行定义，然后单击 确认。在 **设置设备数字输出**模块设置时，选择 I/O 控制里的端口即可。

6. 将后一个 **继续执行直到**模块的“成功”输出与 **呼叫视觉与计算路径（轨迹移动）**模块的输入相连接。

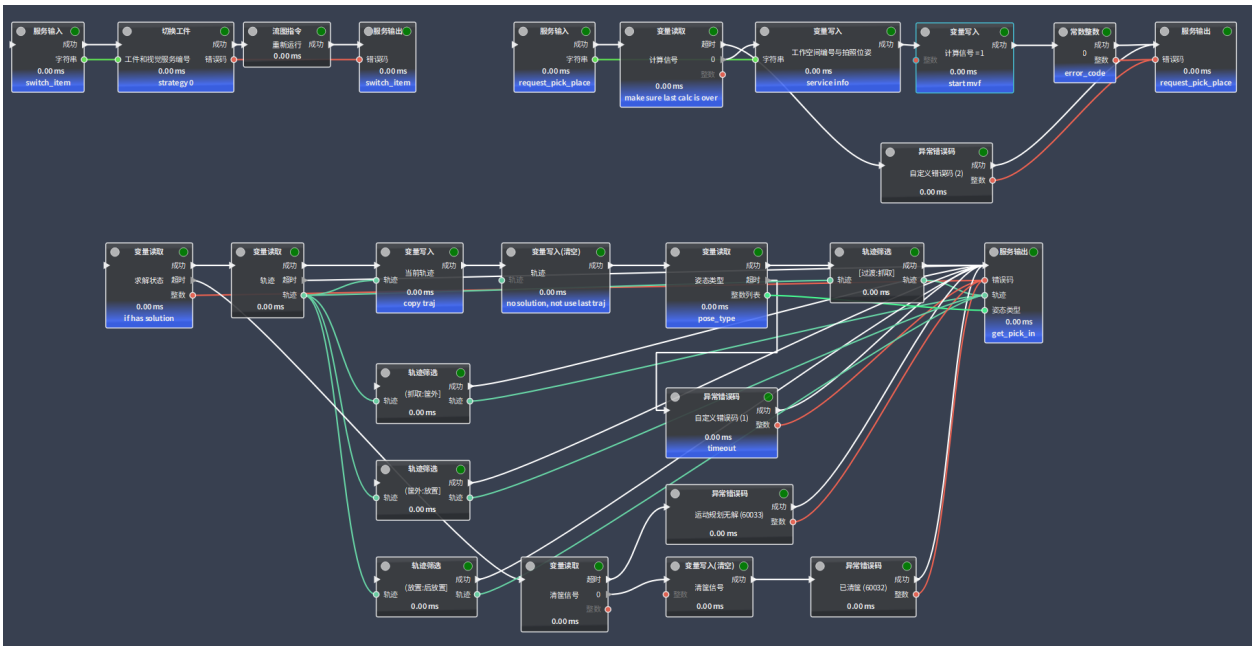
7. 单击任务流图左上角的  ，开始执行整个抓取流程。

机械臂主控

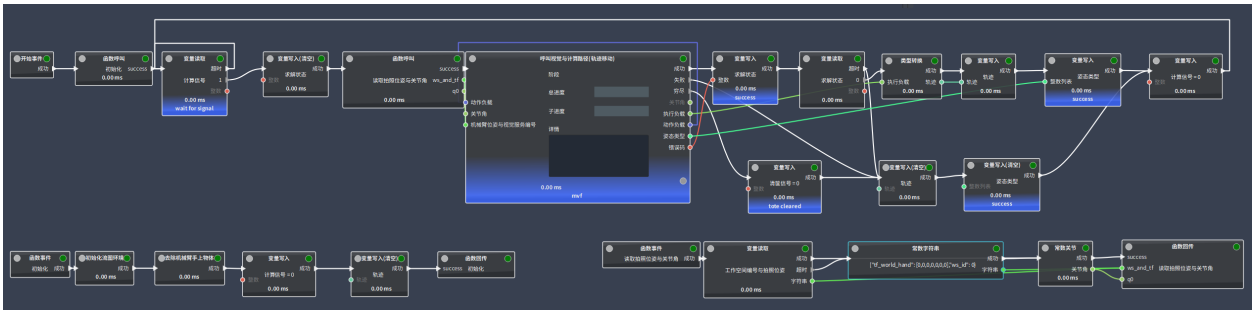
在机械臂主控模板中，需要同时使用 **抓取异步前端**和 **抓取异步后端**两个模板。

其中前端模板主要负责整体抓放流程的控制，而后端完成大量的计算工作，例如机械臂抓取出筐后可以立即拍照，后端可以立刻执行计算任务，这样机械臂完成一次抓放流程之后可以直接获取计算结果进行下一次抓取，以提升执行效率。前端被机械臂呼叫，对指令的响应速度更快，但前端不负责具体的计算任务，仅从后端获取计算结果。

下图是 **抓取异步前端**模板示意图，其中左上部分控制切换工件的流程，右上部分控制请求抓放的流程，下面的流程控制抓放节点筛选。

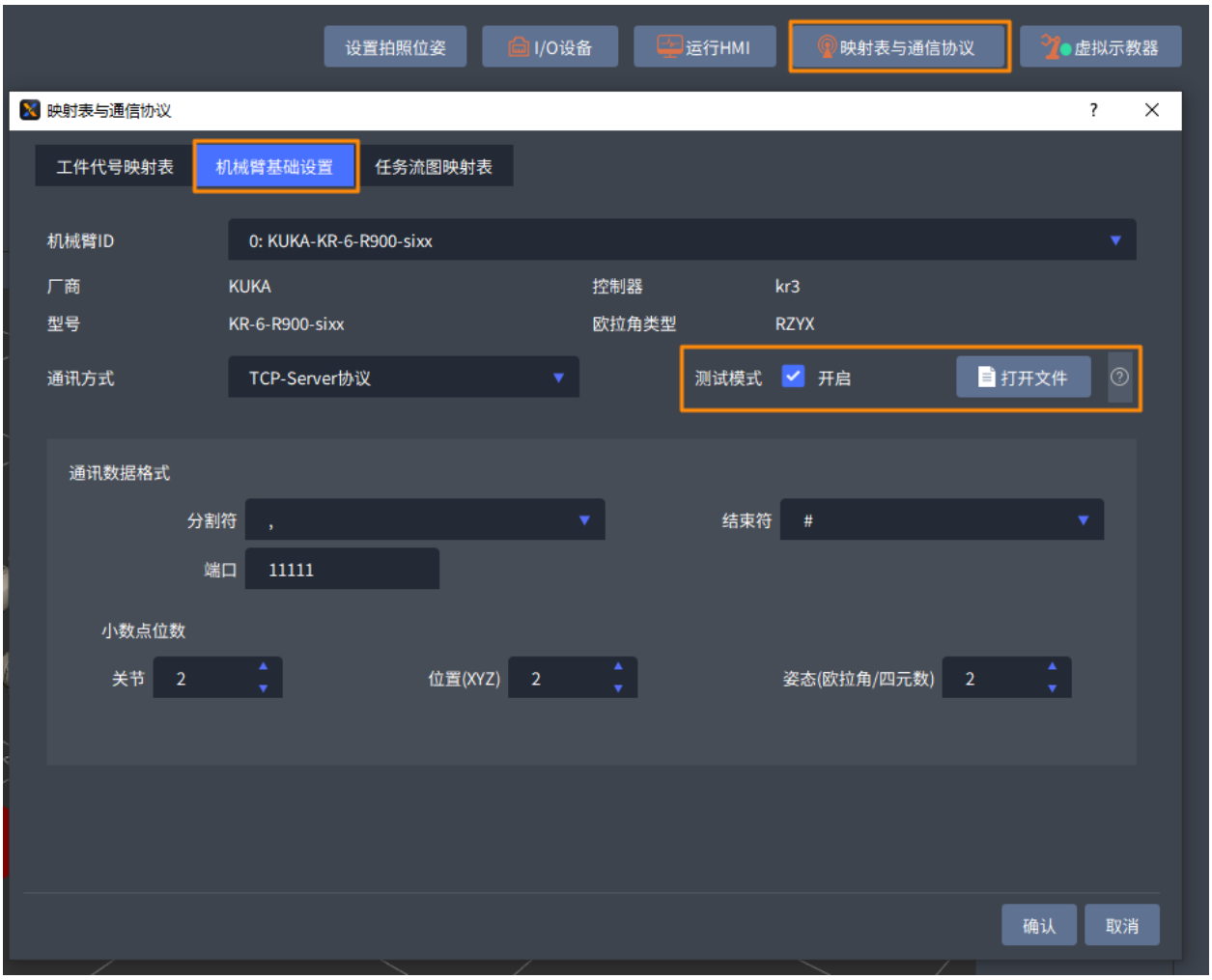


下图是 **抓取异步后端**模板示意图，其中上面是计算抓取轨迹的流程，左下部分控制工作空间的切换，右下部分控制坐标系转换。



机械臂主控模式下，Kuka 机械臂案例和示例代码可参考案例/模板说明，其他品牌的案例和代码可在安装机械臂驱动 中选择相应的品牌查看。

在 Max 中，单击右上方 映射表与通信协议，在弹出窗口的 机械臂基础设置页签下，勾选 **开启**打开测试模式。此时右侧会出现 打开文件和  。单击 打开文件可查看通信指令的 yml 文件，单击  可在浏览器中打开机械臂主控通讯协议 *v2* 页面，并查看机械臂主控通信协议、指令等的内容。



11.2 圆柱体抓取

本章以一个圆柱体（铁棒）抓取项目为例，介绍 3D 轨迹移动场景的配置。在本项目中，圆柱体工件乱序摆放在料筐内，相机安装在机械臂外固定的位置，要求使用 Motoman-GP25 机械臂，配合夹具，将工件抓取并放入另一个料筐中。配置教程如下：

11.2.1 新建项目

根据项目场景，选择模板，创建项目。操作可参考新建项目。

1. 在 Max 顶部工具栏，选择“文件 > 新建项目”。
2. 选择场景模板，在本项目中，选择 **3D-轨迹移动-眼在手外**模板。
3. 添加机械臂，本项目使用 Motoman-GP25 机械臂。
4. 选择 **高级参数**后，添加夹具模型，其他参数保持默认。也可不在新建项目时设置高级参数，待项目创建完成后，在 运动、视觉、任务页签下再作设置。
5. 单击 确认。

11.2.2 视觉

本章节将针对具体场景详细介绍视觉服务的配置过程。在进行视觉配置前，请先参见连接机械臂，完成机械臂连接。

场景要求

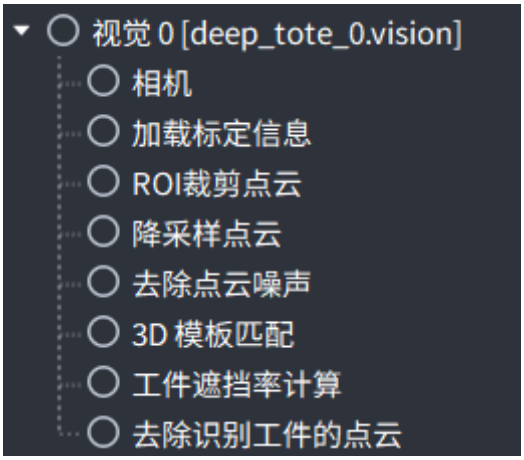
圆柱体抓取的场景采用 **3D 深筐乱序抓取**模板，该流图适用于工作空间内同种物体乱序摆放，且拥有物体 STL 模型的情况。目前广泛应用于汽车零件、机加工件 等多种场景。

- 相机为结构光相机
- 需要计算点云碰撞

本章节描述的相关配置仅供参考，具体操作请以实际项目为准。

基本配置

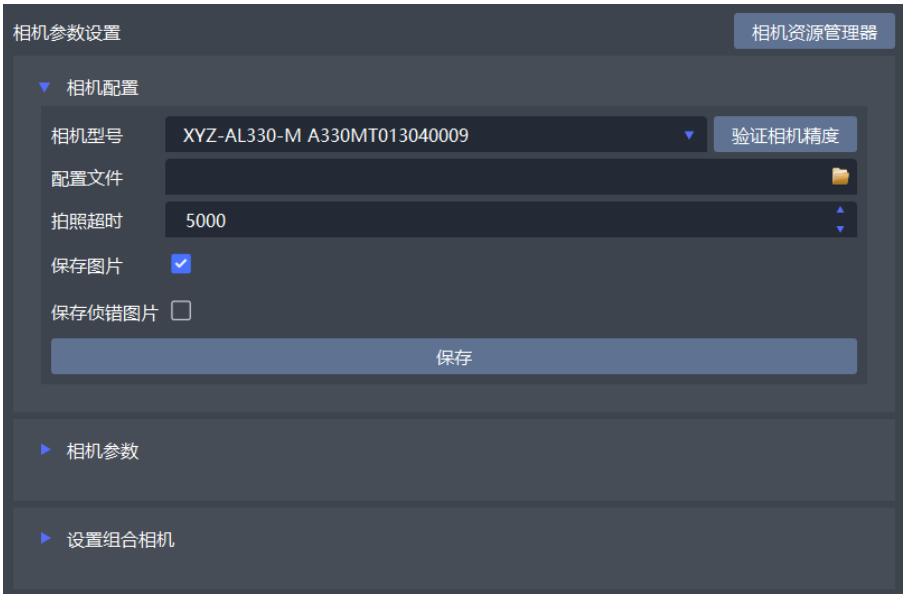
配置前，请依次完成新建项目、添加工作空间，并使用 **3D 深筐乱序抓取**模板创建视觉服务，请参见创建流图。创建完成之后会在左上角生成节点树，如下图所示。



根据需求，依次配置这些模块。

步骤 1. 配置相机模块

1. 单击左侧节点树中的 **相机**模块，然后在右侧单击 相机资源管理器以连接相机，接着在 **相机配置**区域设置相机基本信息，请参见连接相机。
2. 在 **相机参数**区域单击 验证相机精度。



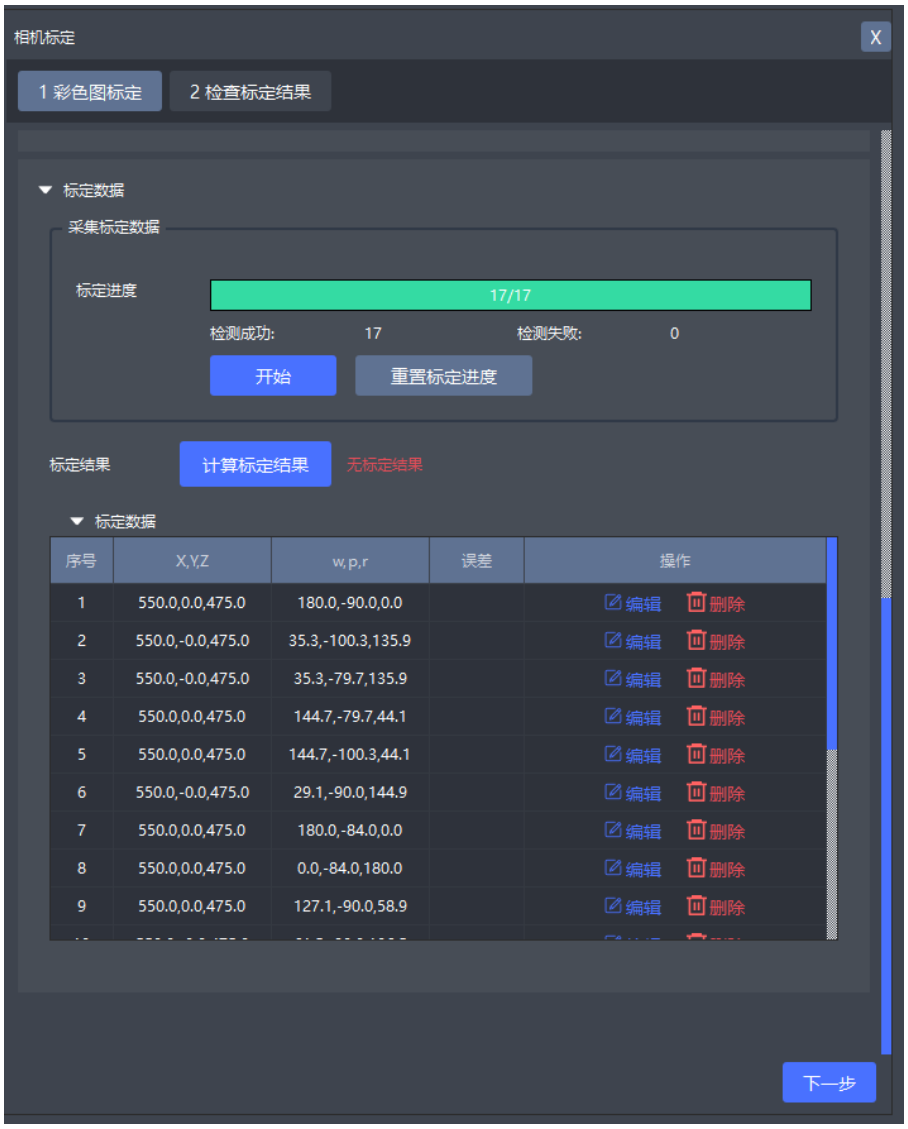
3. 依次选择标定板、标定内参数、验证标定结果等步骤，以验证相机精度。

步骤 2. 配置加载标定信息模块

1. 单击左侧节点树中的 **加载标定信息**模块，然后在右侧依次标定相机和工作空间。

2. 单击 **相机标定**区域中的 新建标定数据，设置标定数据名称，然后单击 编辑标定数据，设置标定方法和其它标定信息，请参见手眼标定 。

3. 依此单击 开始和 计算标定结果完成数据标定，然后单击 下一步进入 **检查标定结果**页签。



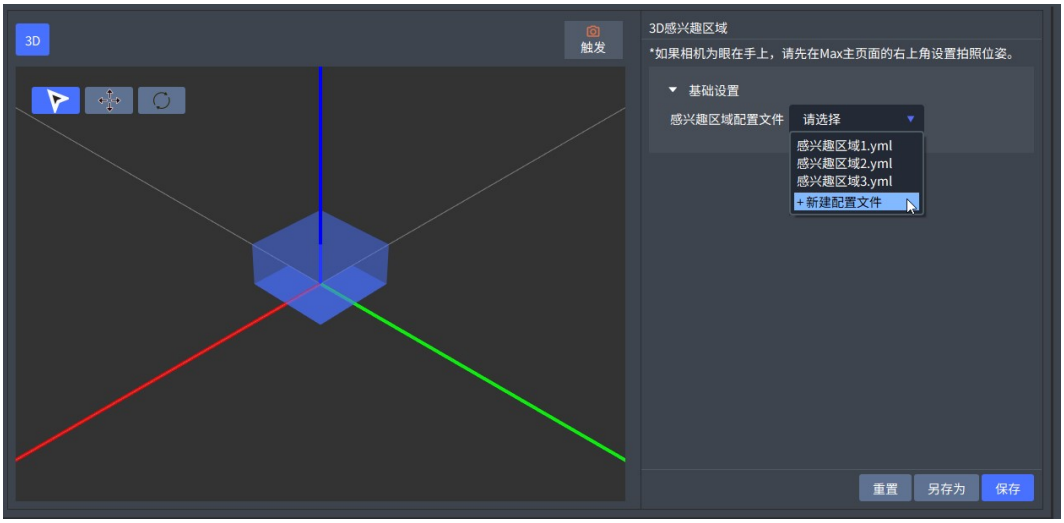
4. 检查并根据界面提示确认标定误差后，单击 **完成**，即可完成相机标定。

5. 若要提高精度，则勾选 点云标定，完成点云标定，请参见手眼标定 中的步骤 5 点云标定的相关内容。
6. 标定工作空间，请参见工作空间标定。3D 相机使用 3D 拖拽的方式标定工作空间。

步骤 3. 配置 ROI 裁剪点云模块

对于圆柱体抓取的项目，可以在加载标定信息之后增加 **ROI 裁剪点云**模块，使识别的区域更加准确。

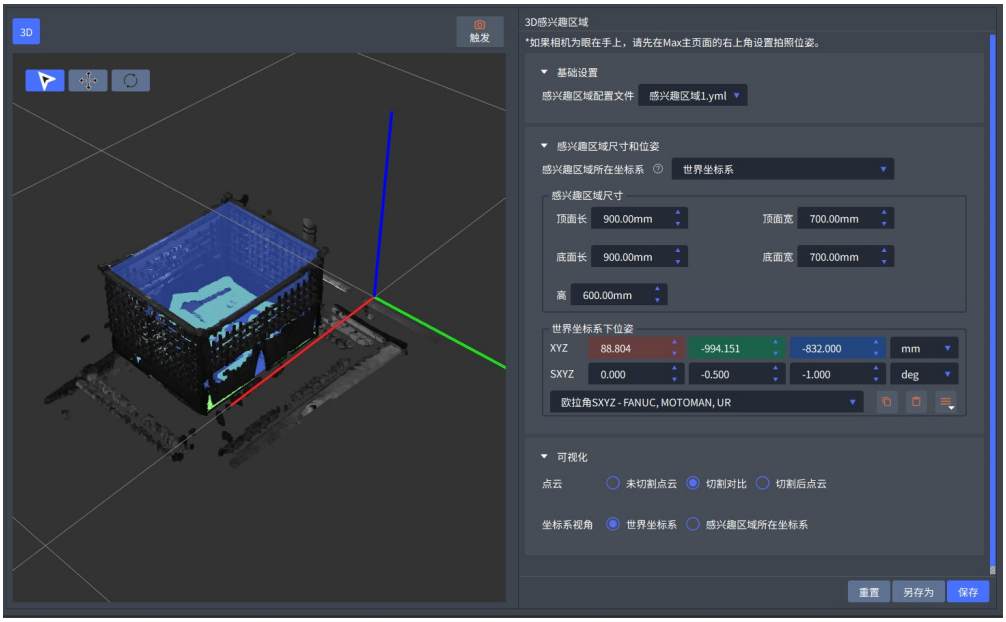
1. 单击左侧节点树中的 **ROI 裁剪点云**模块，然后在右侧单击 触发获取点云图像。
2. 在右侧基础设置里单击 **感兴趣区域配置文件**下拉菜单中的 **新建配置文件**，然后在弹出的 **新建**窗口给配置文件命名。也可选择并修改已有的配置文件。选择配置文件完成后如下图所示。



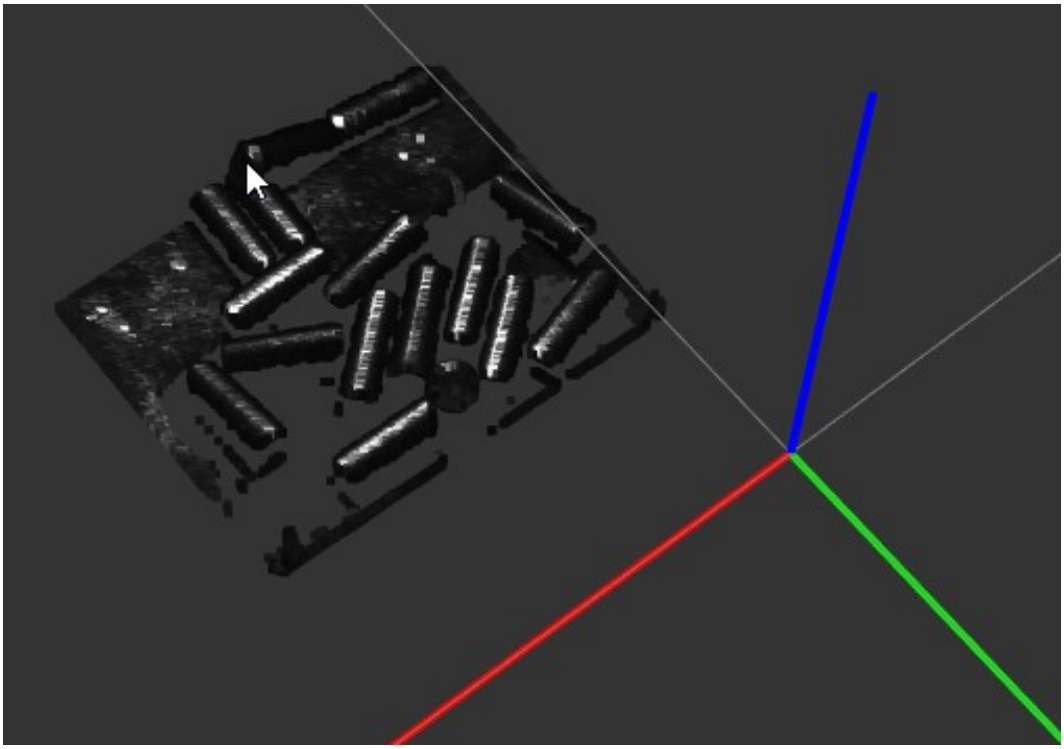
3. 根据实际需求，选择 **感兴趣区域所在坐标系**，Max 提供 **世界坐标系**和 **相机坐标系**两种选项，本章节以世界坐标系为例。

   a. 设置 **感兴趣区域尺寸**，可手动测量工作空间实际尺寸，并在此基础上做调整，一般能够覆盖整个工作空间即可。

   b. 在预览视图中拖动感兴趣区域，使之与点云中的工作空间吻合，可在手动输入数字调整位姿。设置好 ROI 裁剪点云后如下图所示。



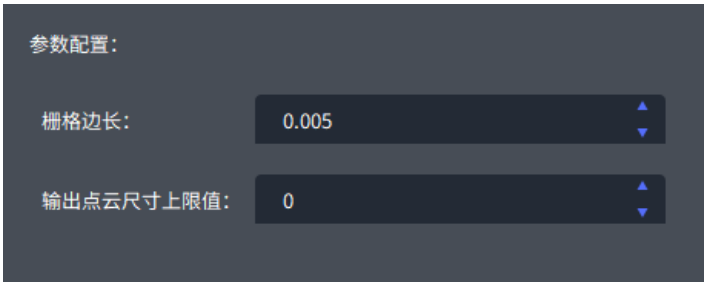
4. 在 **可视化**区域查看点云图，Max 提供三种类型的点云图，未切割点云、切割对比、切割后点云，如下是切割完成后的 ROI 区域点云。



5. 单击 保存，保存感兴趣区域配置文件，也可单击 重置修改配置文件。

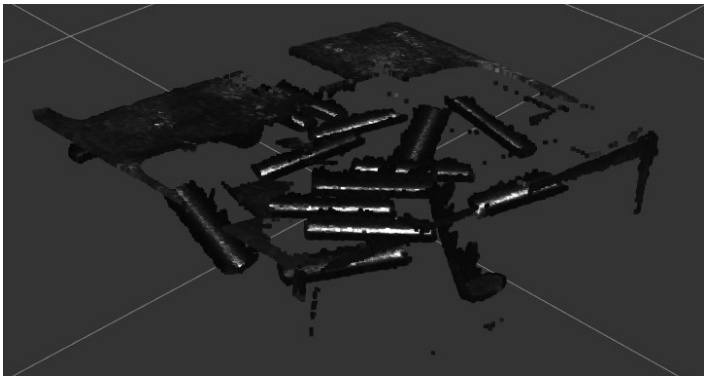
步骤 4. 配置降采样点云模块

1. 单击左侧节点树中的 **降采样点云**模块，然后在右侧配置降采样参数。

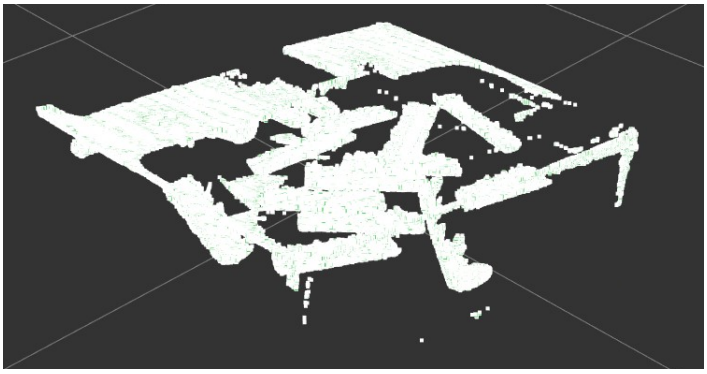


- 栅格边长：将点云中的点划分到指定大小的三维栅格（即正方体）中，栅格边长越大，输出的点云越稀疏。
- 输出点云尺寸上限值：如果该值大于 0，将会进行两次采样，第二次降采样操作将会基于第一次进行。

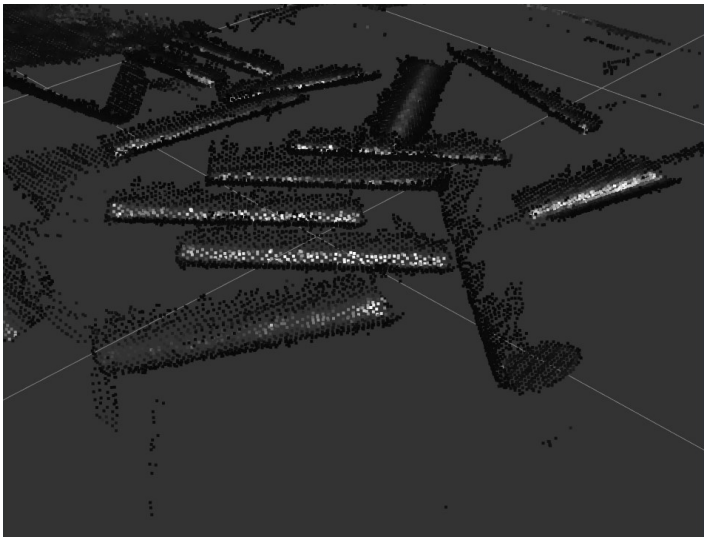
2. 在 **可视化显示**区域选择需要查看的图像。
   - 降采样点云前：仅显示降采样前的图像。



- 降采样点云对比图：对比显示降采样前后的图像，被去除的点云部分用白色显示。



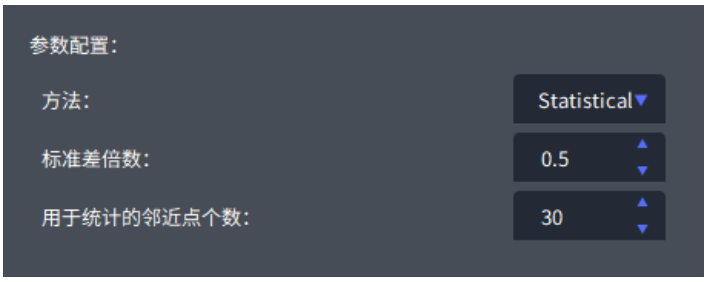
- 降采样点云后：仅显示降采样后的图像，如下是放大细节后的降采样后点云图。



步骤 5. 配置去除点云噪声模块

1. 单击左侧节点树中的 **去除点云噪声**模块，然后在右侧配置去噪声参数，噪声即离群点。过滤方法分为 Statistical 和 Radius 两种。

   Statistical 是基于统计的离群点过滤，运行时间比基于距离的离群点过滤方法更久，其参数配置如下：

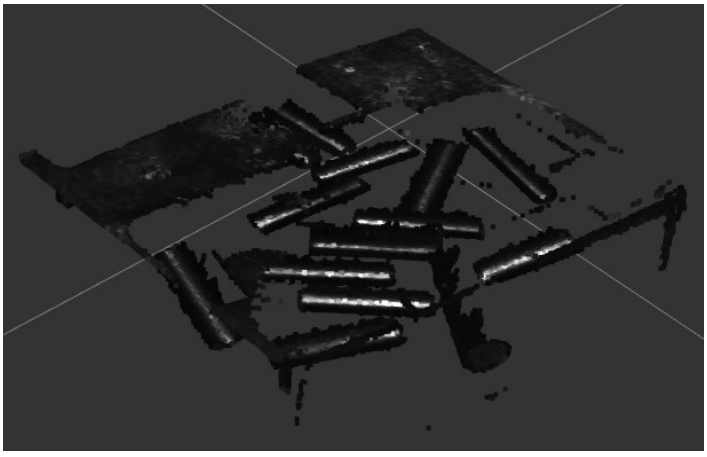


- 标准差倍数：过滤掉与最邻近点平均距离大于平均值 + 倍标准数 * 标准差以外的点，该值越小则过滤点越多。
- 用于统计的临近点个数：该值越大则统计越准确，但计算时间更久。

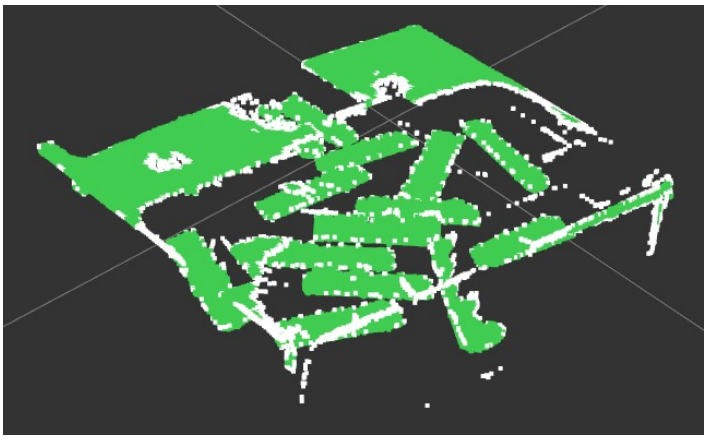
Radius 表示基于距离的离群点过滤，建议经验丰富的工程师使用，其参数配置如下：

- 临近点搜索半径：统计临近点数量时，只考虑该半径范围内的点，该值越大则计算时间越长。
- 最小临近点个数：如果一个点附近的临近点数量小于该值，则会被视为噪声过滤，该值越大则过滤强度越高。

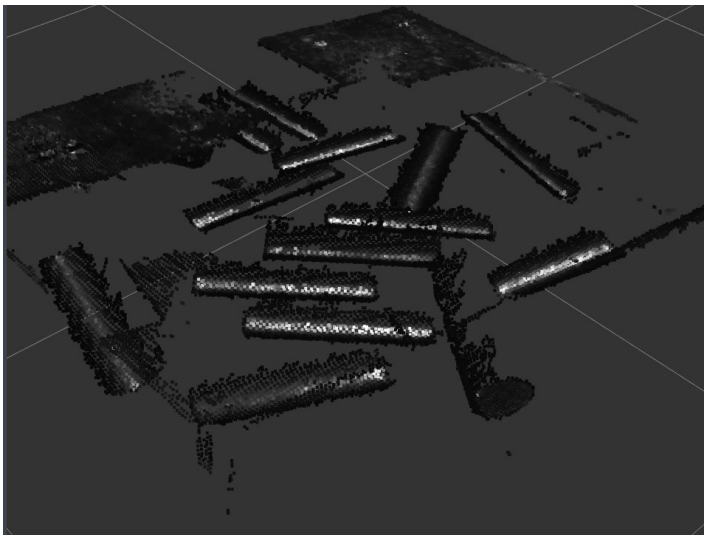
2. 在 **可视化显示**区域选择需要查看的图像。
   - 去除点云噪声前：仅显示去除点云噪声前的图像。



   - 去除点云噪声对比图：对比显示去除噪声前后的图像，被去除的点云用白色显示。

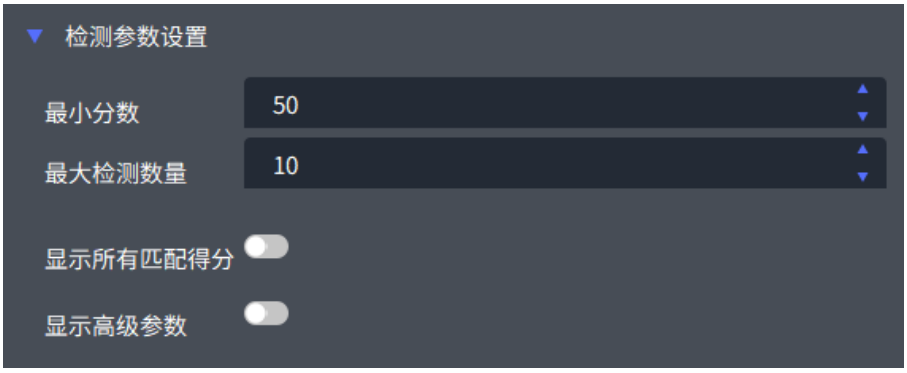


- 去除点云噪声后：仅显示去除点云噪声后的图像，如下是放大细节之后的图像。



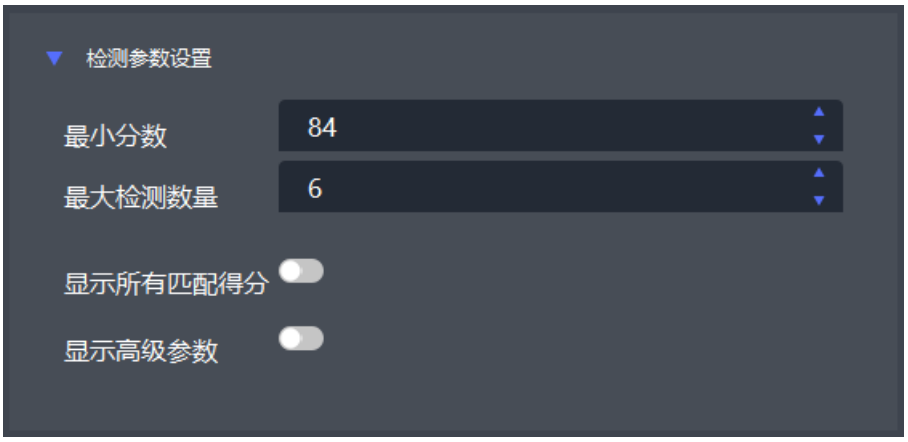
步骤 6. 配置 3D 模板匹配模块

1. 单击 *创建模板*，选择一个已有的或者新建一个工具类型，新建工具类型时不可重名，且工件类型必须和注册抓取时使用的工件名称一致。然后选择模型文件，并给模板命名即可。需要修改模型时，可单击 *编辑模型文件*编辑模型。



提示： 如果已有模板，单击 *选择模板*，直接选择包含模型的文件夹即可。

2. 根据需要选择 特征类型，如果模型表面整体较为扁平，且平面和直线较多，建议尝试 edge-I/edge-II 模式。限制物体朝向建议使用 no_limit，其作用是增加可抓取的朝向和角度，提升抓取概率。
3. 根据需要设置检测参数。模板匹配时，小于 最小分数的结果将被过滤。超过 最大检测数量的结果，也将被过滤。

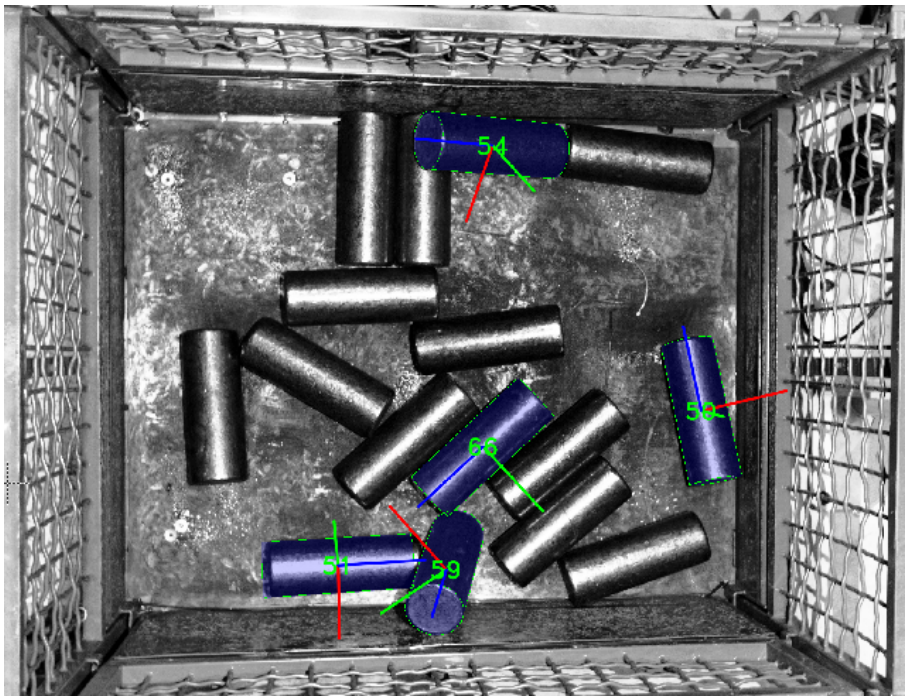


4. 设置不同检测阶段的参数，系统会根据物体模型在点云中寻找匹配结果。

用户也可以查看分步执行的检测结果，各阶段参数设置如下。

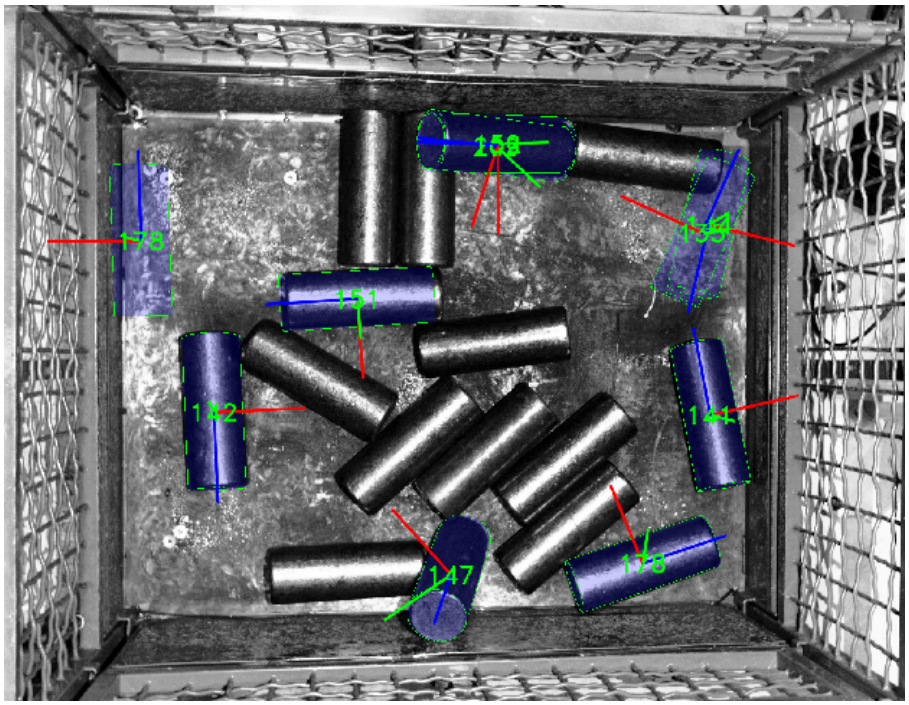
- 通过 **投票**阶段，点云中的每个关键点都会得到一个最可能对应的模型关键点和对应的位姿结果，该阶段各参数配置如下。

该步骤运行结果如下图所示。



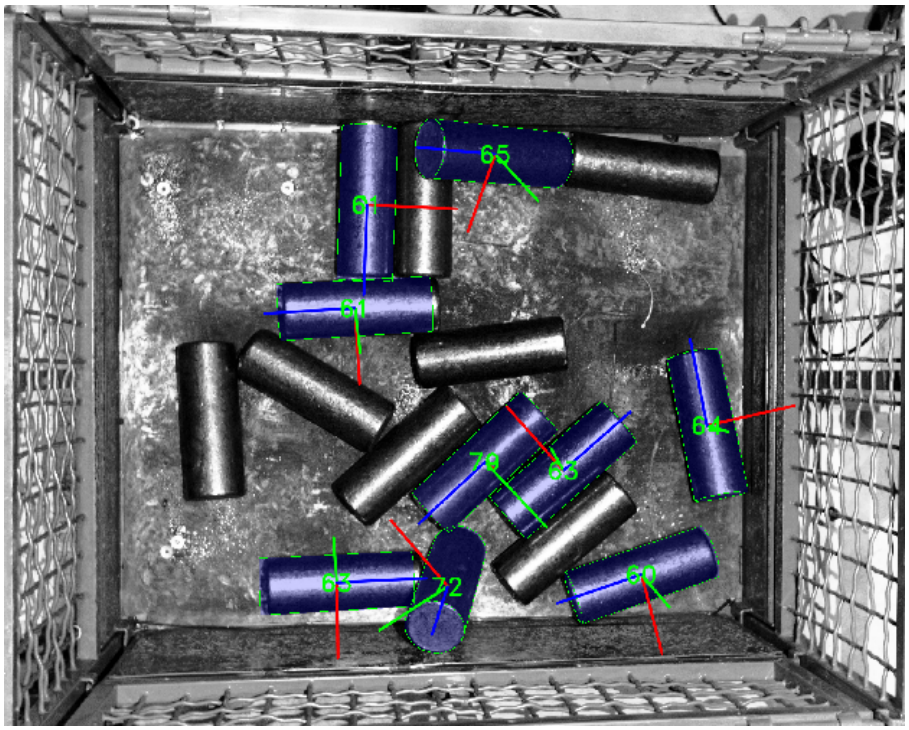
- 通过 **聚类**阶段，合并相近的位姿结果，去除得票数较少的结果，该阶段各参数配置如下。

该步骤运行结果如下图所示。



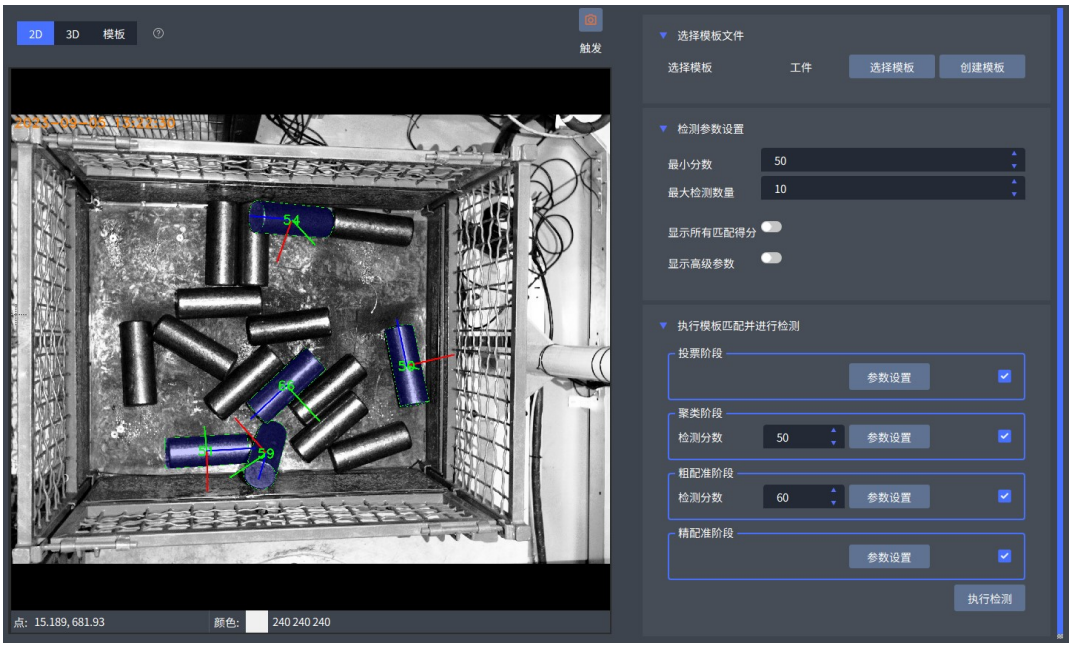
- 通过 **粗配准**阶段，使用 ICP 迭代，优化结果的精度、去除误差较大的结果，该阶段各参数配置如下。

该步骤运行结果如下图所示。



- 通过 **精配准**阶段，使用 ICP 迭代，优化结果的精度、去除误差较大的结果，该阶段各参数配置如下。

5. 单击 执行检测，然后在预览区域查看检测结果。



步骤 7. 配置工件遮挡率模块

1. 单击左侧节点树中的 **工件遮挡率**模块，然后在右侧配置参数。

   - 遮挡率计算方法：有两种计算方法，UseDetected 仅考虑已检测出的工件，UseCloud 使用点云计算遮挡率。
   - 遮挡距离阈值：只有遮挡点与被遮挡点之间的距离大于该阈值才被认为是遮挡关系，否则会被认为是噪声而忽略。该值一般与工件的尺寸相关。
   - 遮挡图尺寸大小：增大尺寸可以提高遮挡率计算的精度，但需要提高输入点云的密度，否则会出现计算错误。

2. 单击 运行，然后在预览区选择输入、输出，查看处理后数据。单击 保存可以存储数据。

步骤 8. 配置去除识别工件的点云模块

1. 单击左侧节点树中的 **去除识别工件的点云**模块，然后在右侧配置参数。

   - 最大去除半径：工件表面附近一定半径内的点将被去除，通常该值为 2~10 毫米。
   - 去除长方体 primitive 点云时的 XY 阈值：去除 primitive 长方体内部点云时使用的 XY 阈值。
   - 去除长方体 primitive 点云时的 Z 阈值：去除 primitive 长方体内部点云时使用的 Z 阈值。

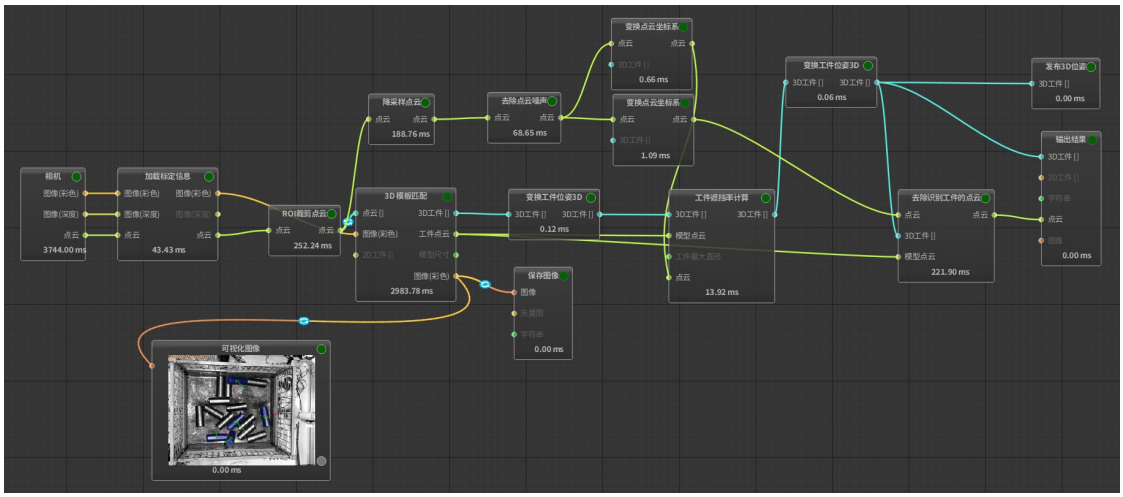
2. 单击 运行，然后在预览区选择输入、输出，查看处理后的点云图像。单击 保存可以存储图像。

流图详解

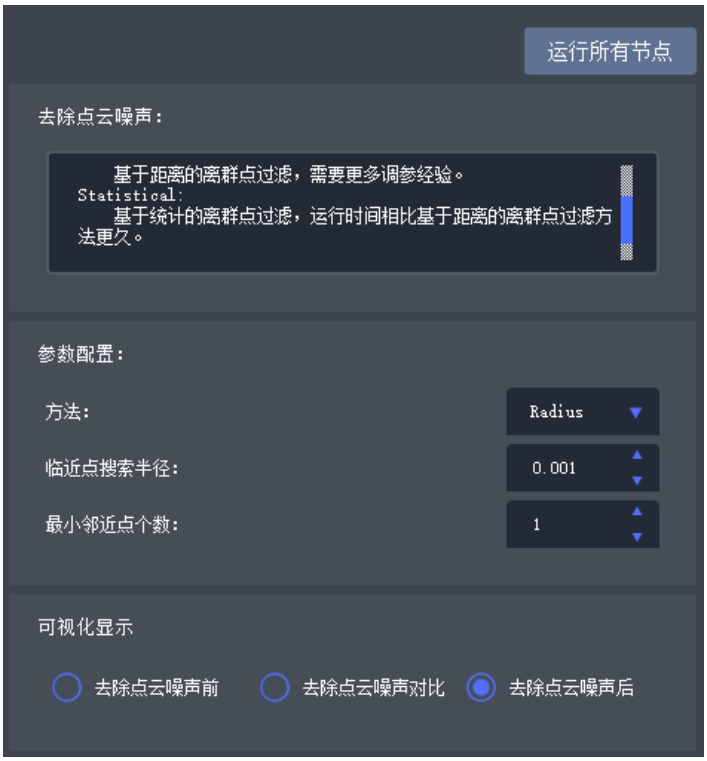
视觉流图的配置是 Max 中的高级用法，对于一般用户而言，配置完节点树中的模块即可运行整个视觉流程，无需再配置视觉流图。视觉流图中包含了节点树中没有的模块，可以配置一些高级参数，如有需要可以按照以下内容操作。

3D 深筐乱序抓取场景的标准视觉流图配置如下。

1. 创建工作空间，使用 **3D 深筐乱序抓取**模板创建流图，请参见创建流图。创建成功的视觉流图如下。



2. 配置 **相机**模块，获取彩色图、深度图及点云。请参见连接相机。
3. 配置 **加载标定信息**模块，请参见标定。
4. 基于步骤 3 得到的点云，配置 **ROI 裁剪点云**模块，仅保留裁剪框内的点云。
5. 基于步骤 4 得到的点云，配置 **降采样点云**、**去除点云噪声**，合理降低点云数量，去除噪声。然后配置 2 个 **变换点云坐标系**模块，分别如下。
   a. 配置一个 **变换点云坐标系**模块，得到工作空间坐标系下的点云，用于计算工件遮挡率。
   b. 再配置一个 **变换点云坐标系**模块，得到世界坐标系下的点云，用于去除识别工件的点云。



6. 结合步骤 3 得到的彩色图和步骤 4 得到的点云，配置 **3D 模板匹配**，得到 3D 工件位姿、工件点云，也可通过 **保存图像**、**可视化图像**模块保存或查看彩色图。
7. 基于步骤 4 得到的 3D 工件位姿，配置 **变换工件位姿 3D** ，得到世界坐标系下的 3D 工件位姿。
8. 基于步骤 7 的位姿，再结合步骤 4 得到的工件模型点云和步骤 5.a 得到的工作空间坐标系下的点云，配置 **工件遮挡率计算**模块，得到工件的 3D 位姿。然后通过 **变换工件位姿 3D** 模块的到世界坐标系下的工件 3D 位姿，并使用 **发布 3D 位姿**模块查看 3D 位姿。
9. 结合步骤 5.b 得到世界坐标系下的点云、步骤 6 得到的工件点云、步骤 8 得到的工件 3D 位姿，配置 **去除识别工件的点云**模块用于去除已检测工件所在位置的点云。
10. 结合步骤 8 得到的工件 3D 位姿、步骤 9 得到的点云，使用 **输出结果**模块输出识别到的工件的位姿和点云图像。
11. 重复上述步骤，完成其它料筐的视觉流图的设置。

11.2.3 运动

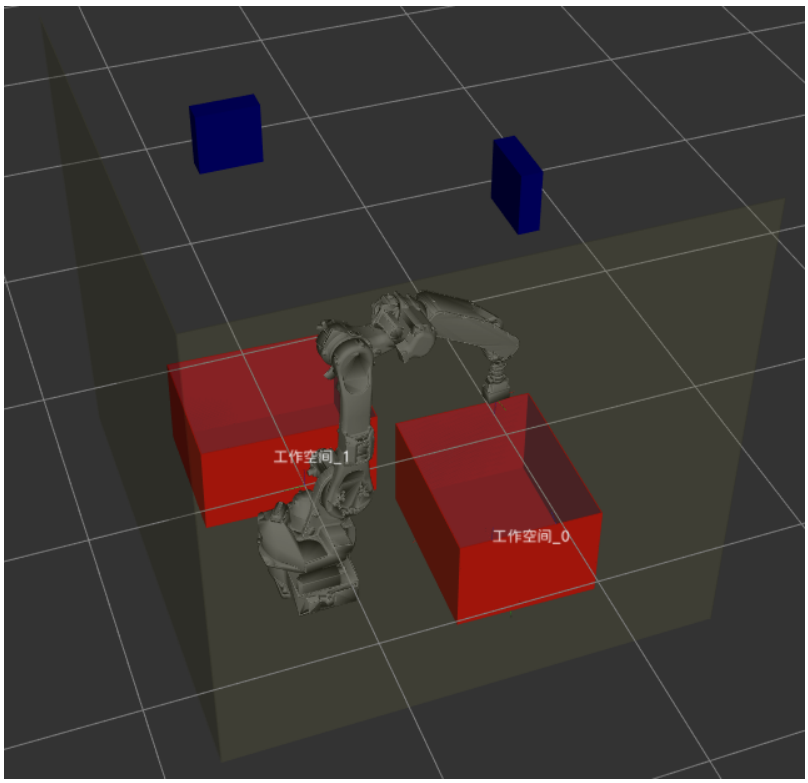
在 运动界面，配置仿真环境并规划运动路径。

提示： 在开始设置运动相关内容前，请先根据机械臂和相机品牌连接并配置真实环境中的机械臂和相机。相关内容请参考连接机械臂 和设置相机 *IP*。如果仅在仿真环境中运行，则无需配置机械臂和相机。

配置仿真环境

根据实际场景，在仿真环境中添加机械臂、料筐、几何体、工具。

1. 创建项目时，已添加机械臂。如未添加，可在 运动页签下快捷工具栏，单击 机械臂，添加机械臂，请参见添加机械臂。
2. 创建项目时，已添加料筐。可在顶部快捷工具栏，单击 工作空间，选择 **添加料筐**，继续添加料筐。本项目需添加两个料筐，添加料筐后需设置料筐及工作空间的尺寸、位姿等参数，请参见添加料筐。
3. 创建项目时，已添加工具，工具自动加载至机械臂末端。如未添加，可在 运动页签下快捷工具栏，单击 末端工具，选择 **注册工具**，注册工具并将工具添加至机械臂，具体参见下一节。
4. 在顶部快捷工具栏，单击 几何体，根据实际场景，添加几何体，用于表示环境中的相机、障碍物等物体。



注册抓取

利用工具和工件，注册抓取方式和抓取处理流程。

1. 注册工具，具体操作参见注册工具。

a. 在 运动页签快捷工具栏，单击 末端工具，选择 **注册工具**，

b. 单击左下角 **工具**节点，在右侧参数设置栏添加工具形态，设置可视化模型和碰撞模型。

c. 添加工具参考点，然后单击右下角 完成，回到 **工具形态**页签，再单击右下角 完成，回到 **工具**页签，单击 完成并添加到机械臂。

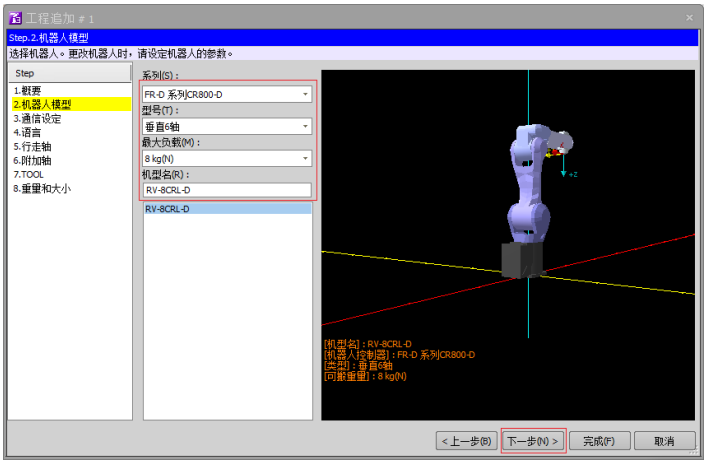
2. 注册工件。在顶部快捷工具栏，单击 工件，选择 **注册工件**，无需添加模型，只需确保工件名称与视觉中 **3D 模板匹配**模块使用的模板名称相同。

3. 注册抓取方式。在顶部快捷工具栏，单击 抓取和放置，选择 "注册抓取方式 > 圆柱抓取注册"。设置圆柱体尺寸，使用已注册的工具设置抓取注册集，请参见注册抓圆柱方式。

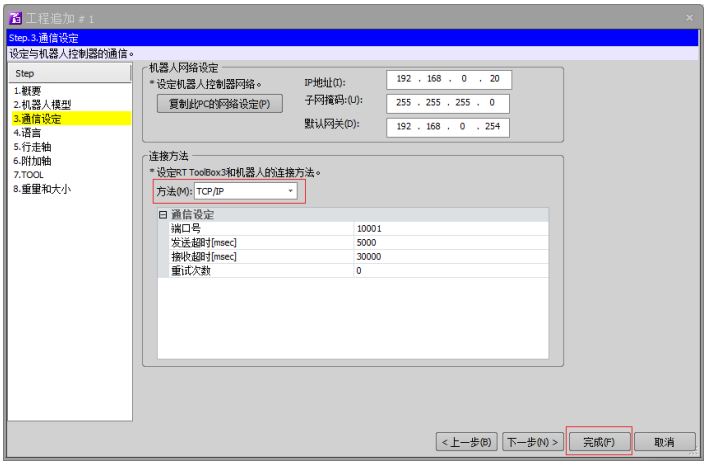
4. 注册抓取处理流程。在顶部快捷工具栏，单击 抓取和放置，选择“注册抓放规划 > 抓取处理流程”，单击左下角 **抓取处理流程**，设置抓取处理流程。在本项目中，可按以下步骤设置：

   a. 工件处理策略选择 **无处理**，抓取注册集选择之前注册的圆柱抓取注册集。

   b. 抓取规划处理策略选择 **[倍增] 根据抓取姿态拆分抓取规划**，对抓取规划进行拆分，再使用 **[过滤] 根据抓取角度过滤抓取规划**加以过滤。



   c. 在抓取规划处理策略中添加 **[排序] 根据混合策略排序抓取规划**，然后将 **[排序] 根据物体遮挡率排序抓取规划**和 **[排序] 根据抓取角度排序抓取规划**添加至排序处理列表中，根据物体遮挡率和抓取角度对抓取规划进行排序。



d. 在抓取规划处理策略中添加 **[合并] 合并抓取物体相同的抓取规划**，将多个抓取规划合而为一，然后单击 下一步。

5. （可选）选择抓取工作空间，单击 测试运行，查看运行结果，包括工件处理结果、抓取规划生成结果、抓取规划处理结果。
6. 单击右下角 完成。

配置运动路径

添加点位，配置机械臂运动路径。一般需添加抓取点、放置点以及相对于抓取点、放置点的点位。

1. 在 运动页签下的 **路径**面板，单击右上角  ，选择 **导入深筐模板**。
2. 在右侧参数设置栏 **抓取配置**下方，设置抓取移动涉及的工作空间、处理流程、视觉服务编号，在 **放置配置**下方设置放置工作空间。

3. 单击左下角 **过渡**节点，在右侧参数设置栏单击 **目标点**右侧 添加，使用示教器控制机械臂运动到指定点位后，单击 获取机械臂关节角，单击 确定。

4. 单击左下角 **预抓取**节点，在右侧参数设置栏单击 **目标点**右侧 添加，设置预抓取点相对于抓取点的相对位姿。

5. 单击左下角 **抓取**节点，在右侧参数设置栏设置运动类型，建议使用 **线性移动 & 关节过奇点-MoveLJL**。在该模式下，机械臂默认采取线性移动（MoveL）方式，并自动判断是否经过奇异点，经过奇异点时切换成关节移动（MoveJ）方式，经过后再切换回线性移动方式。
6. 单击左下角 **后抓取**节点，在右侧参数设置栏设置运动类型，建议使用 **线性移动 & 关节过奇点-MoveLJL**，单击 **目标点**右侧 添加，设置后抓取点相对于抓取点的相对位姿。
7. 单击左下角 **筐外**节点，在右侧参数设置栏单击 **目标点**右侧 添加，设置该点位的机械臂关节角。
8. 单击左下角 **预放置**节点，在右侧参数设置栏单击 **目标点**右侧 添加，设置预放置点的机械臂关节角。
9. 单击左下角 **放置**节点，在右侧参数设置栏将 **抓放状态**设置为 **放下工件**，单击 **目标点**右侧 添加，添加放置点的机械臂关节角。

10. 单击左下角 **后放置**节点，在右侧参数设置栏单击 **目标点**右侧 添加，设置后放置点的机械臂关节角。
11. 右键单击中间预览窗口上方 **深筐路径**页签，选择 **保存**。

11.2.4 任务

任务部分利用视觉部分的计算结果和运动部分的运动规划，控制整个项目的流程运行。本章节将详细介绍任务部分的配置。

工件代号映射表

设置完视觉和运动部分之后、运行任务流图之前，需要添加工件代号映射关系，以便任务流图调用视觉和运动设置的工件、视觉服务编号、抓取处理流程等内容，具体步骤如下。

1. 单击 Max 右上角 映射表与通信协议，在弹出的 **映射表与通信协议**窗口选择“工件代号映射表 > 轨迹移动”页签。
2. 单击 添加映射关系，在弹出的 **添加映射关系**窗口设置键、值和参数。

- 工件代号：用户自定义工件代号，仅支持英文字母和数字，长度不超过 15 个字符。可添加多个工件。
- 视觉服务编号：在下拉菜单选择视觉服务编号，编号和视觉部分的设置保持一致。
- 视觉流图文件：与视觉空间 ID 下的流图保持一致。
- 处理流程：在下拉菜单中选择抓取处理流程，这些抓取处理流程已在运动部分设置好。运动流程编号从 0 开始自动增加，可单击  或  新建或删除处理流程。
- 运动路径：选择在 *运动* 中设置好的一条路径。
- 缓存多个视觉结果：可根据需要是否勾选。
- 圆柱体：本项目是针对圆柱体（铁棒）的项目，需要勾选该参数并设置圆柱体的直径和长，重新注册圆柱体的抓取方式。

3. 单击 *确认* 完成添加，列表中会显示已添加的映射关系，用户可对列表中的映射关系进行操作。

-  ：修改映射关系。
-  ：复制并修改映射关系，注意键值不能重复。
-  ：删除映射关系。

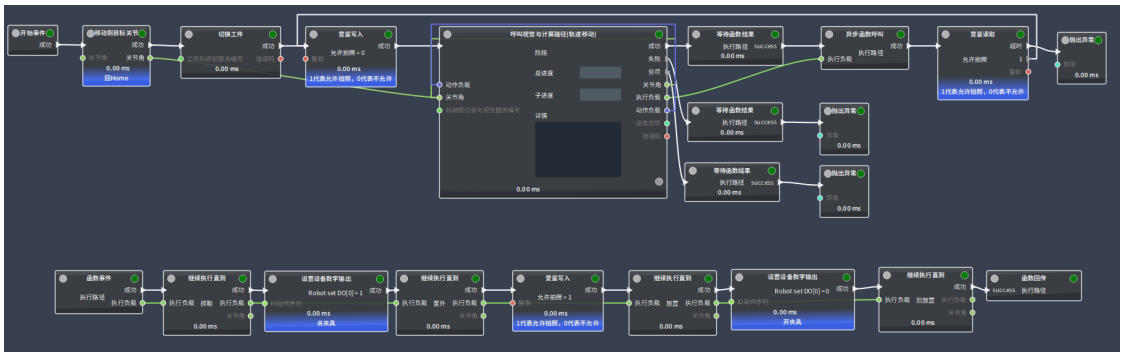
4. 单击 *确认*完成映射表设置。

提示： 用户可单击 *自动生成映射关系*，系统自动根据视觉和运动的设置生成工件代号映射表，但有多个抓取处理流程时无法自动生成映射关系。

设置完工件关系映射表之后，在 *任务*中导入 3D 轨迹移动中的模板，请参见流图配置。任务流图的设置分为工控机主控和机械臂主控两种方式，每种方式下含多个模板，可根据需要选择。在本章节工控机主控使用 **抓取异步（粗颗粒度）**模板，机械臂主控使用 **抓取异步前端**和 **抓取异步后端**模板。

工控机主控

添加好基础模板之后，用户可根据需求添加或修改模块。以抓取圆柱体（铁棒）项目为例，配置完成后的流图如下。



下面详细介绍该任务流图的配置步骤。

1. 在任务流图模块当中选择 “功能 > 一键更新节点机械臂属性”，系统自动根据当前设置更新机械臂路径。也可在模块的属性中设置机械臂路径。

2. 单击 **切换工件**模块，在右侧设置 **工件代号**，与上述映射表中的工件代号保持一致。

3. 单击 **移动到目标关节**模块，在右侧 **属性**页签中设置目标关节角、位置和速度等参数，即起始点位置。

4. 分别单击四个 **继续执行直到**模块，在右侧 **属性**页签中将 **目标点名称**设置为与 运动中路径规划设置的点相同的名称，例如该项目中四个模块分别设为“抓取”、“筐外”、“放置”和“后放置”。

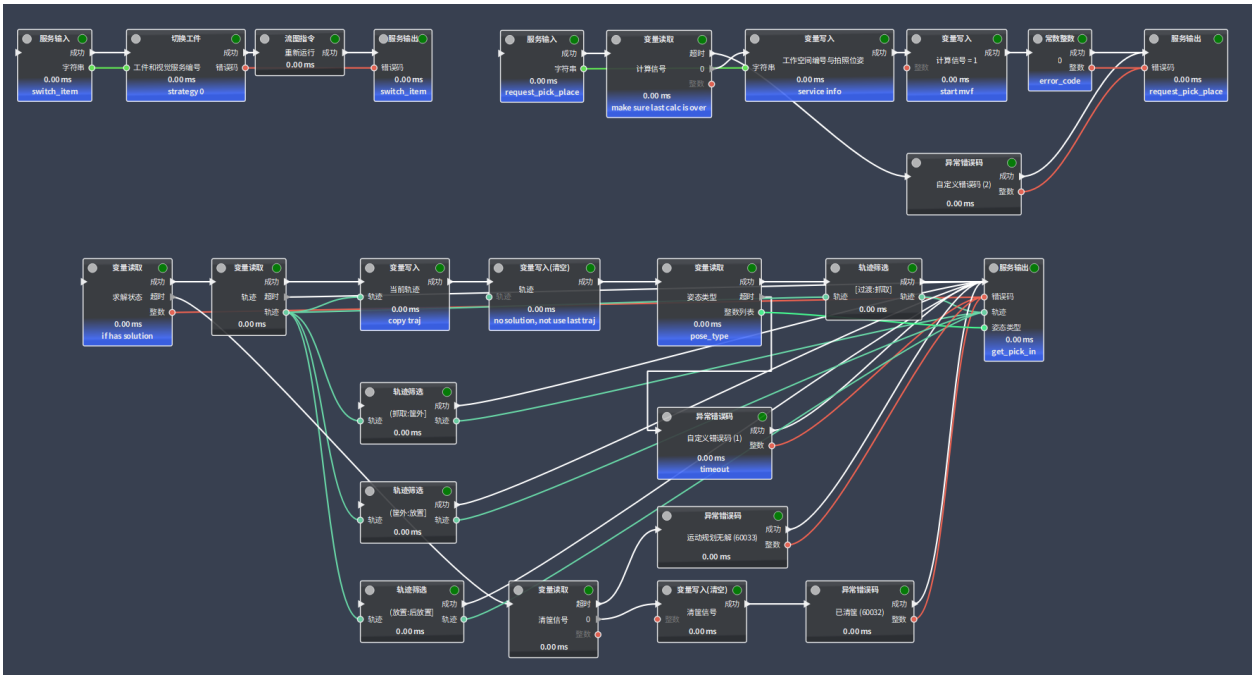
5. 单击任务流图左上角的  ，开始执行整个抓取流程。

机械臂主控

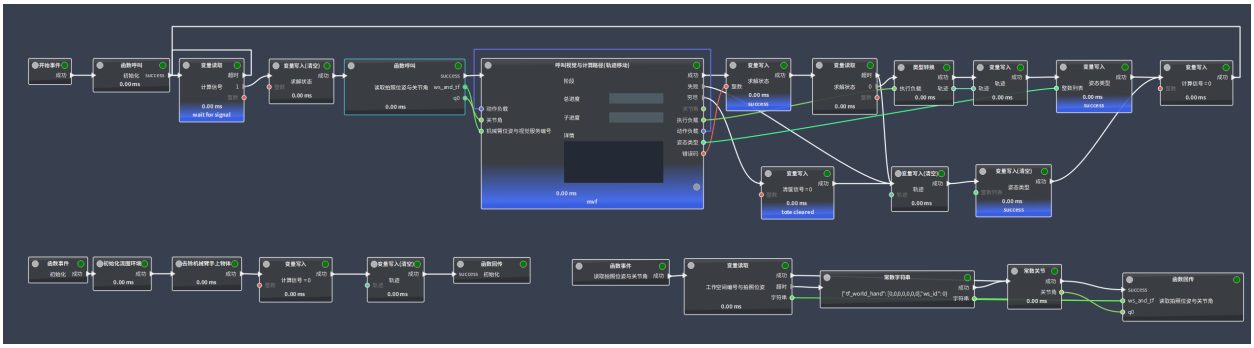
在机械臂主控模板中，需要同时使用 **抓取异步前端**和 **抓取异步后端**两个模板。

其中前端模板主要负责整体抓放流程的控制，而后端完成大量的计算工作，例如机械臂抓取出筐后可以立即拍照，后端可以立刻执行计算任务，这样机械臂完成一次抓放流程之后可以直接获取计算结果进行下一次抓取，以提升执行效率。前端被机械臂呼叫，对指令的响应速度更快，但前端不负责具体的计算任务，仅从后端获取计算结果。

下图是 **抓取异步前端**模板示意图，其中左上部分控制切换工件的流程，右上部分控制请求抓放的流程，下面的流程控制抓放节点筛选。



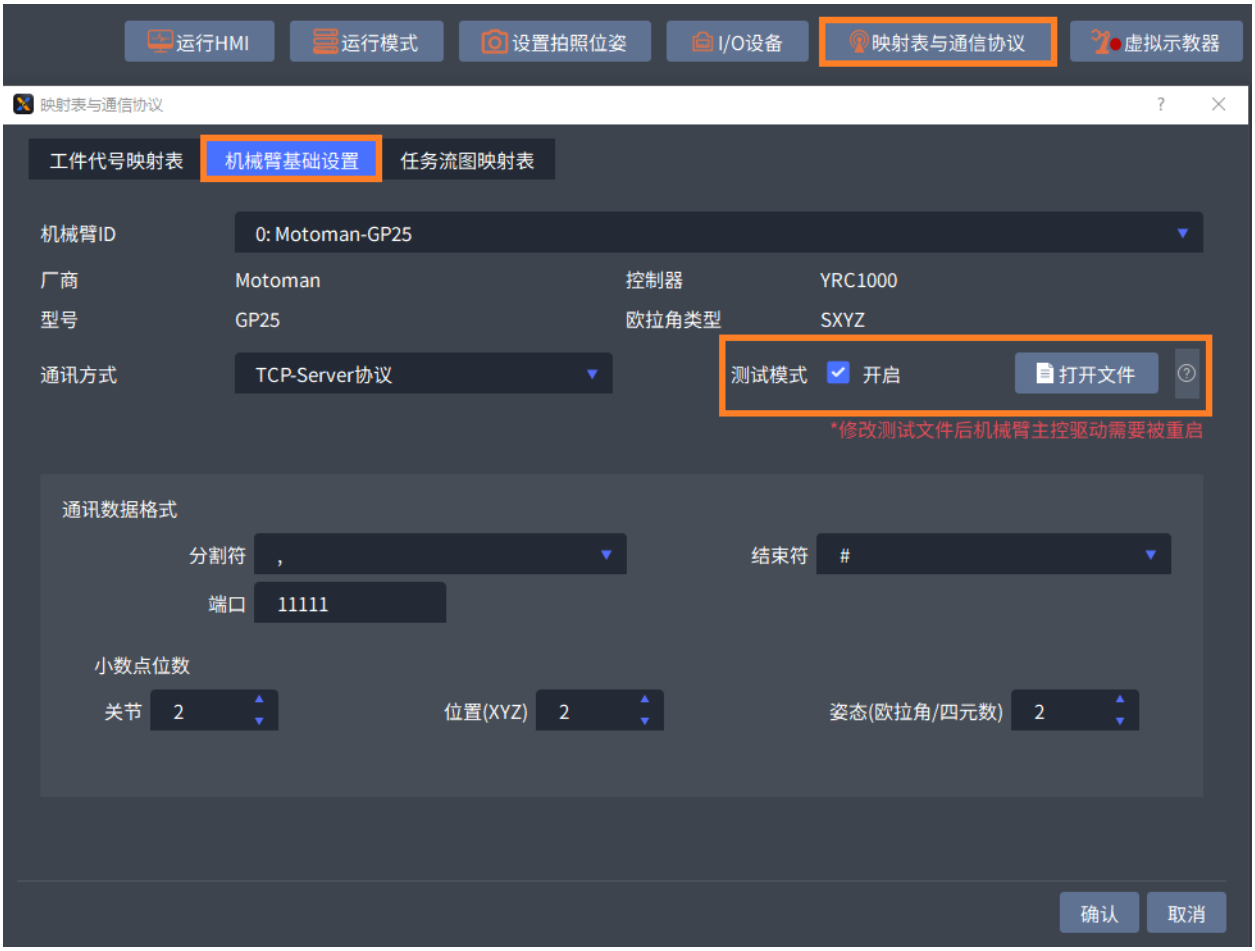
下图是 **抓取异步后端**模板示意图，其中上面是计算抓取轨迹的流程，左下部分控制工作空间的切换，右下部分控制坐标系转换。



机械臂主控模式下，Yaskawa Motoman 机械臂案例和示例代码可参考案例/模板说明，其他品牌的案例和代码可在安装机械臂驱动 中选择相应的品牌查看。

在 Max 中，单击右上方 映射表与通信协议，在弹出窗口的 机械臂基础设置页签下，勾选 **开启**打开测试模

式。此时右侧会出现 打开文件和 。单击 打开文件可查看通信指令的 yml 文件，单击 可在浏览器中打开机械臂主控通讯协议 v2 页面，并查看机械臂主控通信协议、指令等的内容。



11.3 工业码垛

本章以一个工业码垛项目为例，介绍 3D 轨迹移动场景的配置。在本项目中，机械臂将轴承从一个料筐抓取至另一个料筐内，并码放整齐。相机安装在机械臂外固定的位置，要求使用 Fanuc-M-10iD-8L 机械臂。配置教程如下：

11.3.1 新建项目

根据项目场景，选择模板，创建项目。操作可参考新建项目。

1. 在 Max 顶部工具栏，选择“文件 > 新建项目”。
2. 选择场景模板，在本项目中，选择 **3D-轨迹移动-眼在手外**模板。
3. 添加机械臂，本项目使用 Fanuc-M-10iD-10L 机械臂。
4. 选择 **高级参数**后，添加夹具和工件模型，其他参数保持默认。也可不在新建项目时设置高级参数，待项目创建完成后，在 运动、视觉、任务页签下再作设置。

5. 单击 确认。

11.3.2 视觉

本章节将针对具体场景详细介绍视觉服务的配置过程。在进行视觉配置前，请先参见连接机械臂，完成机械臂连接。

场景要求

工业码垛场景采用 **3D 深筐乱序抓取**模板，该流图适用于工作空间内同种物体乱序摆放，且拥有物体 STL 模型的情况。目前广泛应用于汽车零件、机加工件 等多种场景。

- 相机为结构光相机
- 需要计算点云碰撞

本章节描述的相关配置仅供参考，具体操作请以实际项目为准。

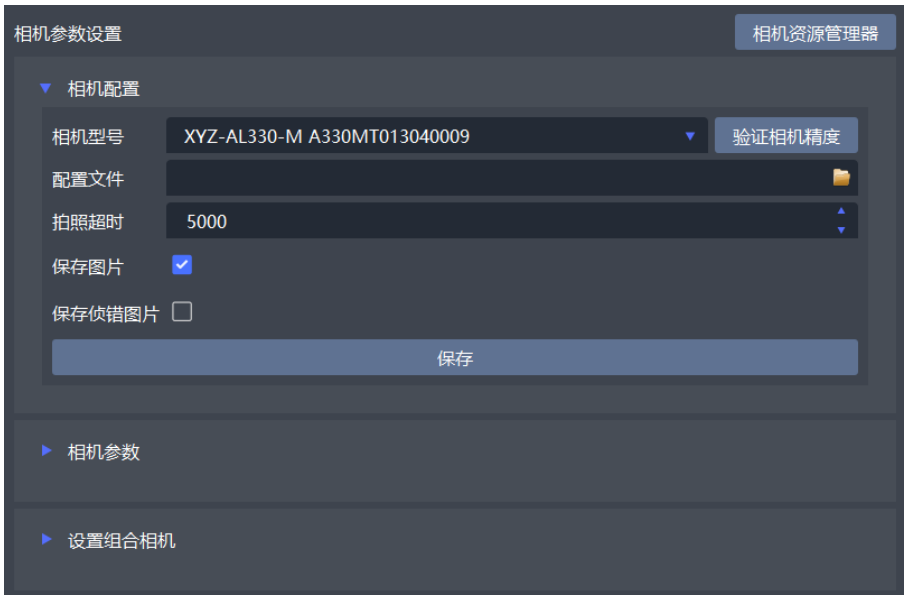
基本配置

配置前，请依次完成新建项目、添加工作空间，并使用 **3D 深筐乱序抓取**模板创建视觉服务，请参见创建流图。创建完成之后会在左上角生成节点树，如下图所示。

根据需求，依次配置这些模块。

步骤 1. 配置相机模块

1. 单击左侧节点树中的 **相机**模块，然后在右侧单击 相机资源管理器以连接相机，接着在 **相机配置**区域设置相机基本信息，请参见连接相机。
2. 在 **相机参数**区域单击 验证相机精度。



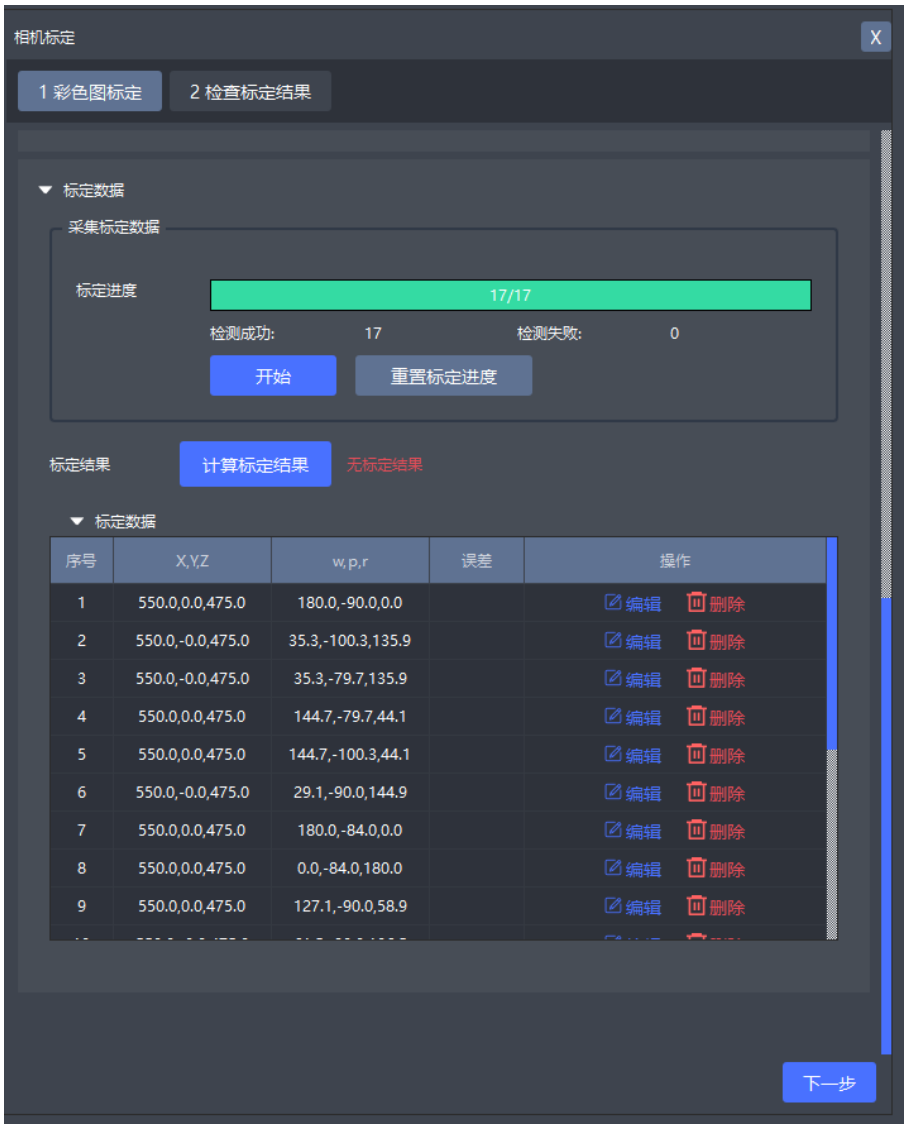
3. 依次选择标定板、标定内参数、验证标定结果等步骤，以验证相机精度。

步骤 2. 配置加载标定信息模块

1. 单击左侧节点树中的 **加载标定信息**模块，然后在右侧依次标定相机和工作空间。

2. 单击 **相机标定**区域中的 *新建标定数据*，设置标定数据名称，然后单击 *编辑标定数据*，设置标定方法和其它标定信息，请参见手眼标定 。

3. 依此单击 开始和 计算标定结果完成数据标定，然后单击 下一步进入 **检查标定结果**页签。



4. 检查并根据界面提示确认标定误差后，单击 **完成**，即可完成相机标定。

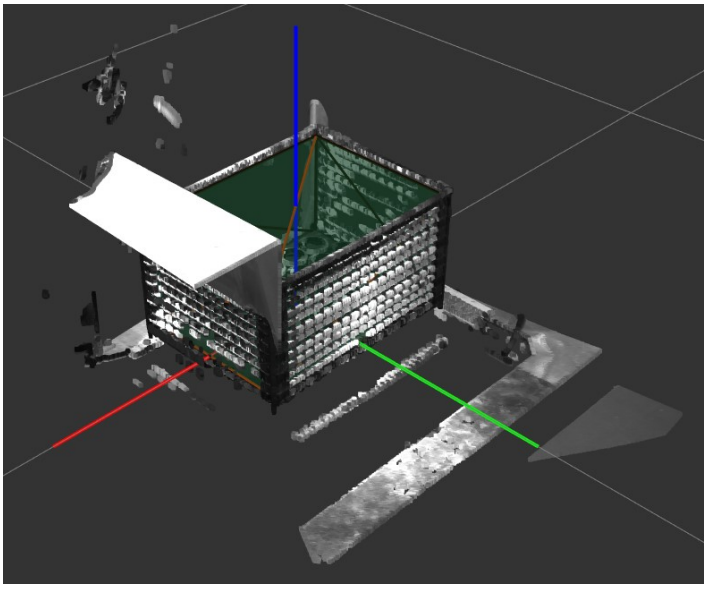
5. 若要提高精度，则勾选 点云标定，完成点云标定，请参见手眼标定 中的步骤 5 点云标定的相关内容。
6. 标定工作空间，请参见工作空间标定。3D 相机使用 3D 拖拽的方式标定工作空间。

步骤 3. 配置去除料筐外点云模块

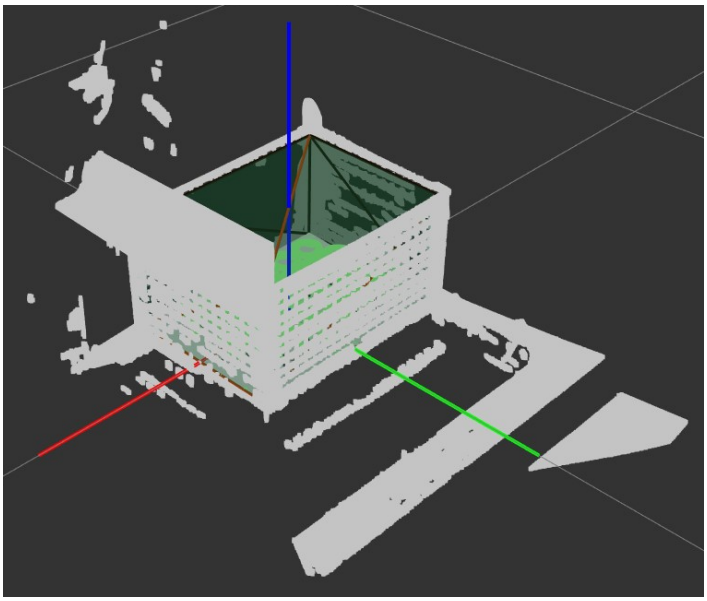
1. 单击左侧节点树中的 **去除料框外点云**模块，然后在右侧配置点云切割参数，然后单击预览窗口中的 触发。

- X 方向收缩长度：料筐沿 X 方向的收缩长度，负值则意味着扩张。
- Y 方向收缩长度：料筐沿 Y 方向的收缩长度，负值则意味着扩张。
- 底部抬升高度：料筐底部抬升高度，负值则意味着下降。
- 顶部下降高度：料筐顶部下降高度，负值则意味着抬升。

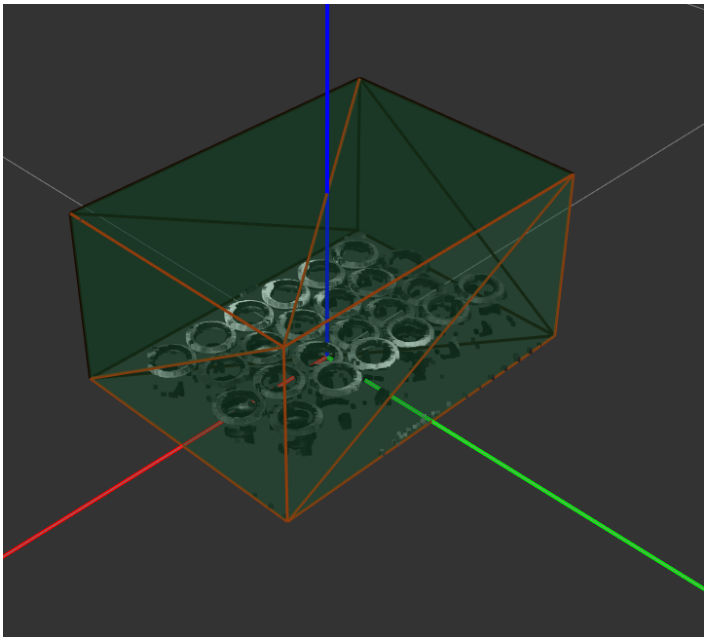
2. 在预览视图中移动料筐模型，使之与点云中的料筐位置吻合。
3. 在 **可视化显示**区域选择需要查看的图像。
   - 未切割点云图：仅显示未去除料筐外点云的图像。



- 切割对比图：对比显示切割前后的图像，被切割的点云用白色显示。

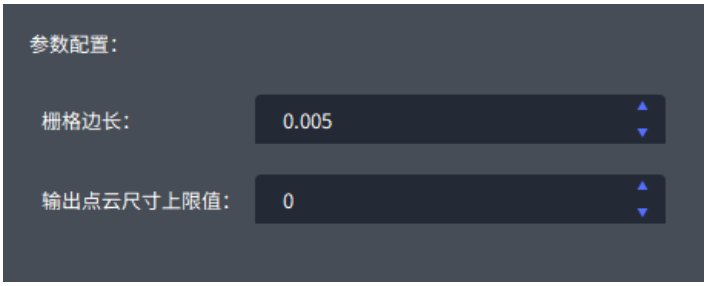


- 切割完点云图：仅显示去除料筐外点云的图像。



步骤 4. 配置降采样点云模块

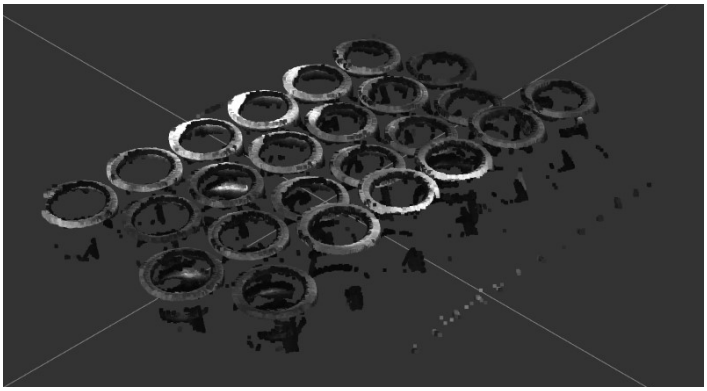
1. 单击左侧节点树中的 **降采样点云**模块，然后在右侧配置降采样参数。



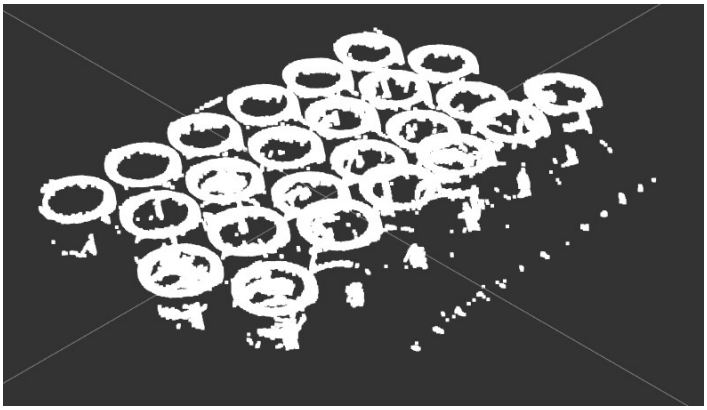
- 栅格边长：将点云中的点划分到指定大小的三维栅格（即正方体）中，栅格边长越大，输出的点云越稀疏。
- 输出点云尺寸上限值：如果该值大于 0，将会进行两次采样，第二次降采样操作将会基于第一次进行。

2. 在 **可视化显示**区域选择需要查看的图像。

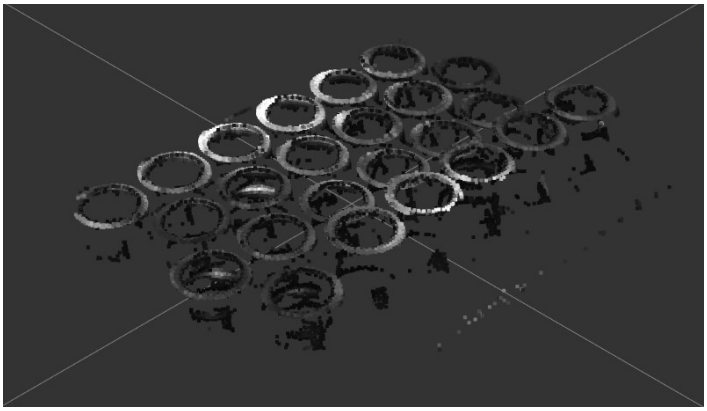
- 降采样点云前：仅显示降采样前的图像。



- 降采样点云对比图：对比显示降采样前后的图像，被去除的点云部分用白色显示。



- 降采样点云后：仅显示降采样后的图像。



步骤 5. 配置去除点云噪声模块

1. 单击左侧节点树中的 **去除点云噪声**模块，然后在右侧配置去噪声参数，噪声即离群点。过滤方法分为 Statistical 和 Radius 两种。

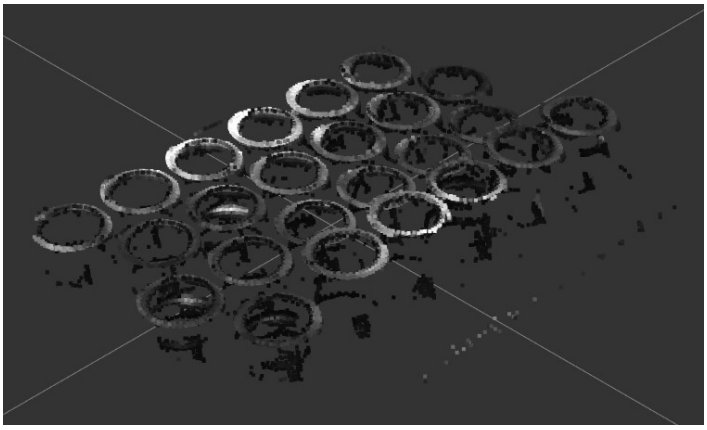
Statistical 是基于统计的离群点过滤，运行时间比基于距离的离群点过滤方法更久，其参数配置如下：

- 标准差倍数：过滤掉与最邻近点平均距离大于平均值 + 倍标准数 * 标准差以外的点，该值越小则过滤点越多。
- 用于统计的临近点个数：该值越大则统计越准确，但计算时间更久。

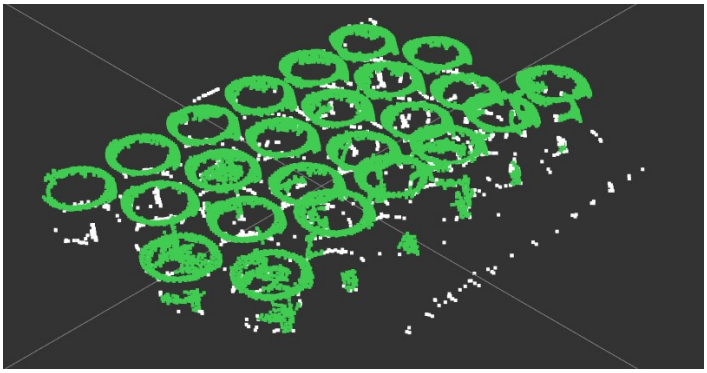
Radius 表示基于距离的离群点过滤，建议经验丰富的工程师使用，其参数配置如下：

- 临近点搜索半径：统计临近点数量时，只考虑该半径范围内的点，该值越大则计算时间越长。
- 最小临近点个数：如果一个点附近的临近点数量小于该值，则会被视为噪声过滤，该值越大则过滤强度越高。

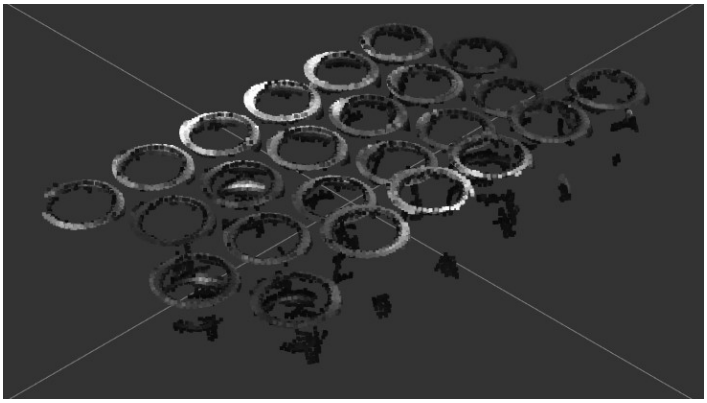
2. 在 **可视化显示**区域选择需要查看的图像。
   - 去除点云噪声前：仅显示去除点云噪声前的图像。



   - 去除点云噪声对比图：对比显示去除噪声前后的图像，被去除的点云用白色显示。



- 去除点云噪声后：仅显示去除点云噪声后的图像。

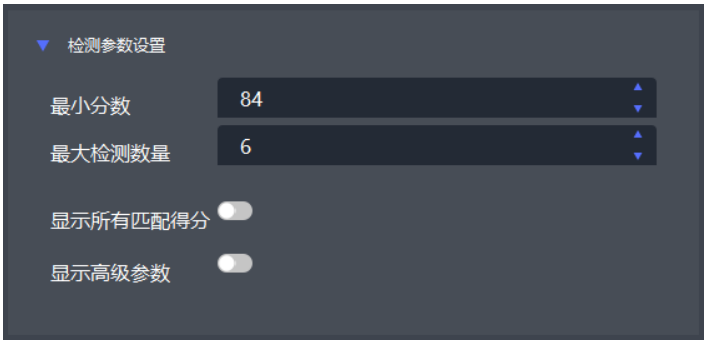


步骤 6. 配置 3D 模板匹配模块

1. 单击 创建模板，选择一个已有的或者新建一个工具类型，新建工具类型时不可重名，且工件类型必须和注册抓取时使用的工件名称一致。然后选择模型文件，并给模板命名即可。需要修改模型时，可单击 编辑模型文件编辑模型。

提示: 如果已有模板，单击 选择模板，直接选择包含模型的文件夹即可。

2. 根据需要选择 特征类型，如果模型表面整体较为扁平，且平面和直线较多，建议尝试 edge-I/edge-II 模式。限制物体朝向建议使用 no_limit，其作用是增加可抓取的朝向和角度，提升抓取概率。
3. 根据需要设置检测参数。模板匹配时，小于 最小分数的结果将被过滤。超过 最大检测数量的结果，也将被过滤。

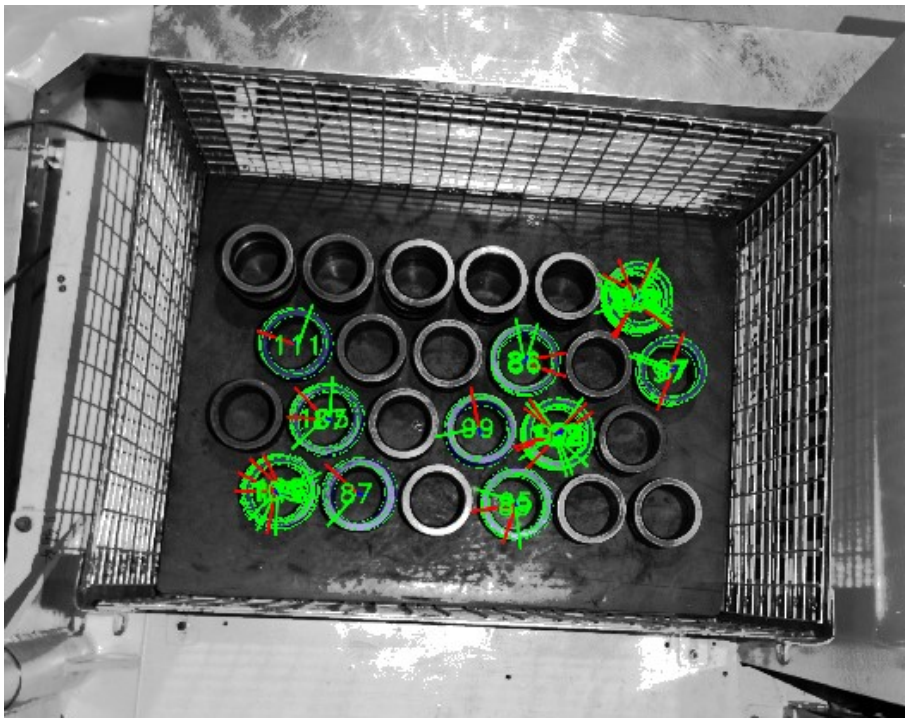


4. 设置不同检测阶段的参数，系统会根据物体模型在点云中寻找匹配结果。

用户也可以查看分步执行的检测结果，各阶段参数设置如下。

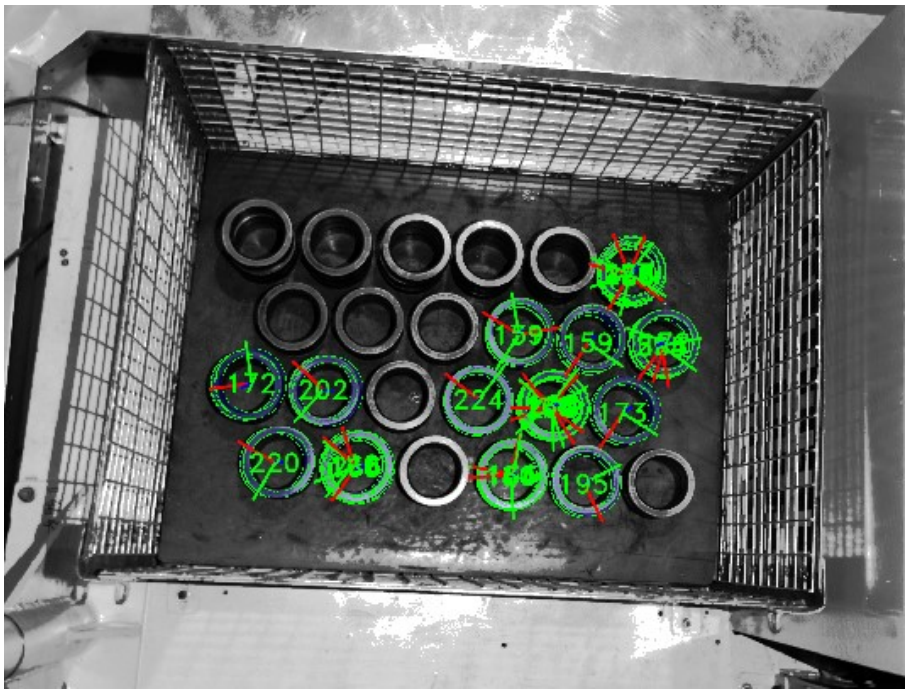
- 通过 **投票**阶段，点云中的每个关键点都会得到一个最可能对应的模型关键点和对应的位姿结果，该阶段各参数配置如下。

该步骤运行结果如下图所示。



- 通过 **聚类**阶段，合并相近的位姿结果，去除得票数较少的结果，该阶段各参数配置如下。

该步骤运行结果如下图所示。

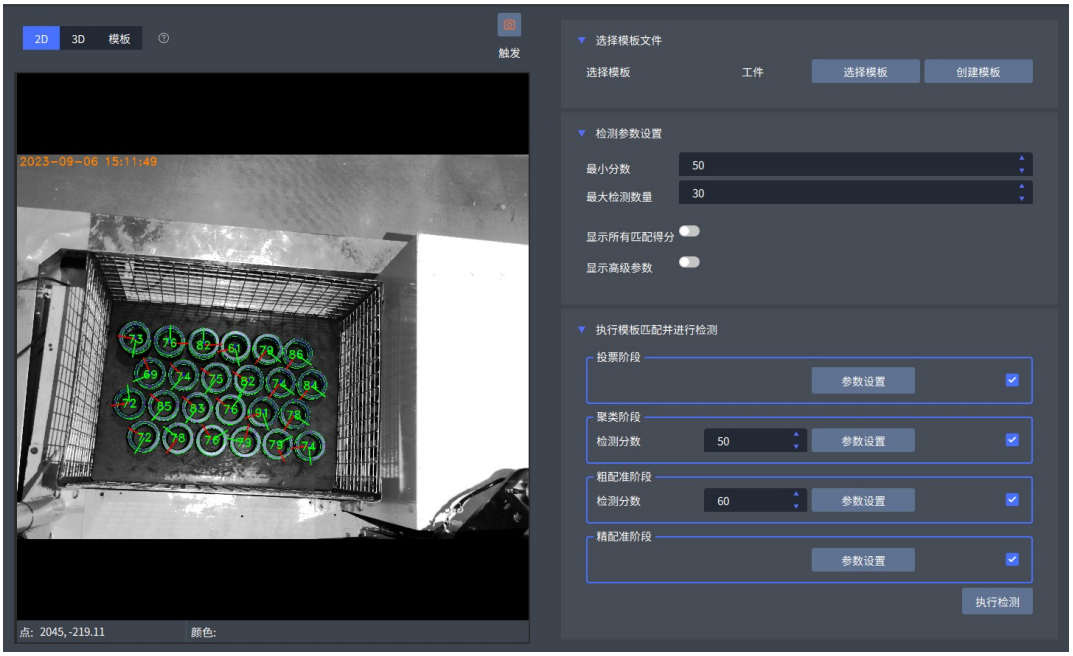


- 通过 **粗配准**阶段，使用 ICP 迭代，优化结果的精度、去除误差较大的结果，该阶段各参数配置如下。

该步骤运行结果如下图所示。

- 通过 **精配准**阶段，使用 ICP 迭代，优化结果的精度、去除误差较大的结果，该阶段各参数配置如下。

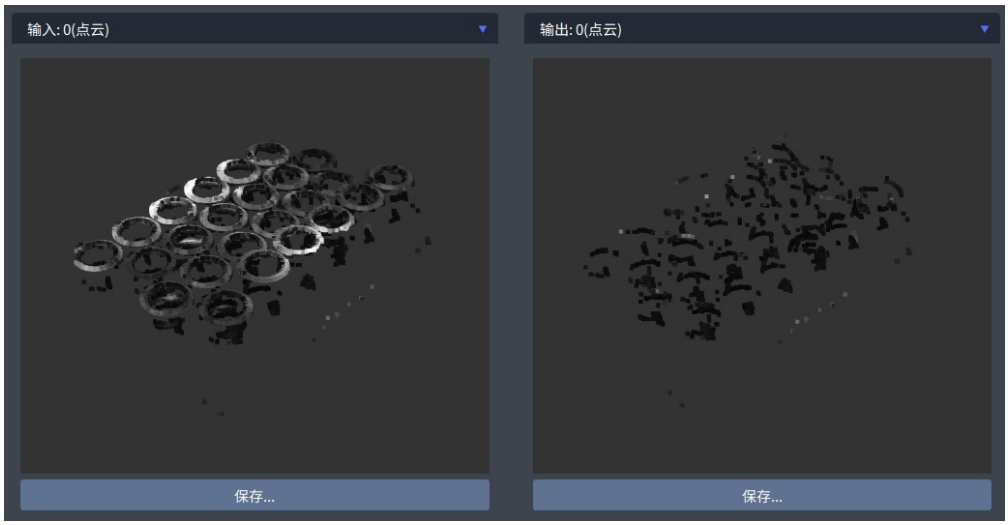
5. 单击 执行检测，然后在预览区域查看检测结果。



步骤 7. 配置去除识别工件的点云模块

1. 单击左侧节点树中的 **去除工件附近的点云**模块，然后在右侧配置参数。

   - 最大去除半径：工件表面附近一定半径内的点将被去除，通常该值为 2~10 毫米。
   - 去除长方体 primitive 点云时的 XY 阈值：去除 primitive 长方体内部点云时使用的 XY 阈值。
   - 去除长方体 primitive 点云时的 Z 阈值：去除 primitive 长方体内部点云时使用的 Z 阈值。
2. 单击 运行，然后在预览区选择输入、输出，查看处理后的点云图像。单击 保存可以存储图像。

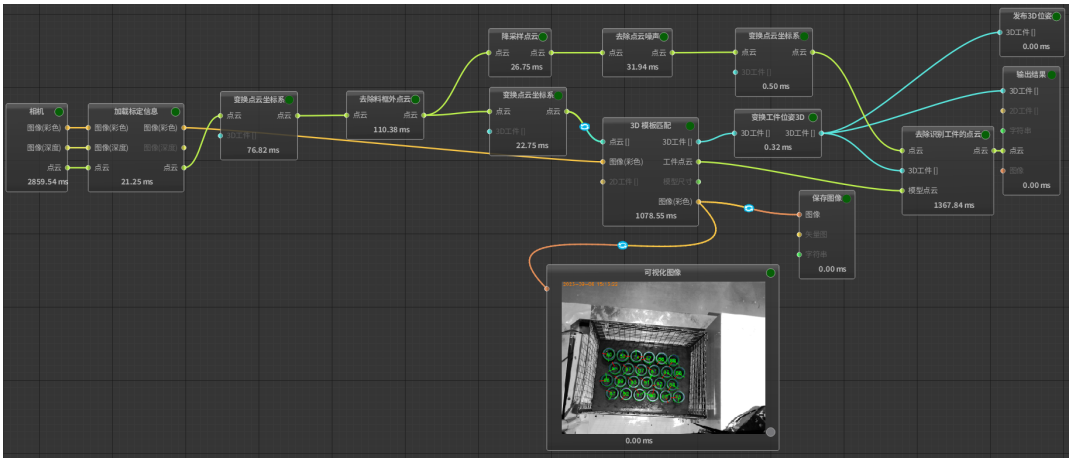


流图详解

视觉流图的配置是 Max 中的高级用法，对于一般用户而言，配置完节点树中的模块即可运行整个视觉流程，无需再配置视觉流图。视觉流图中包含了节点树中没有的模块，可以配置一些高级参数，如有需要可以按照以下内容操作。

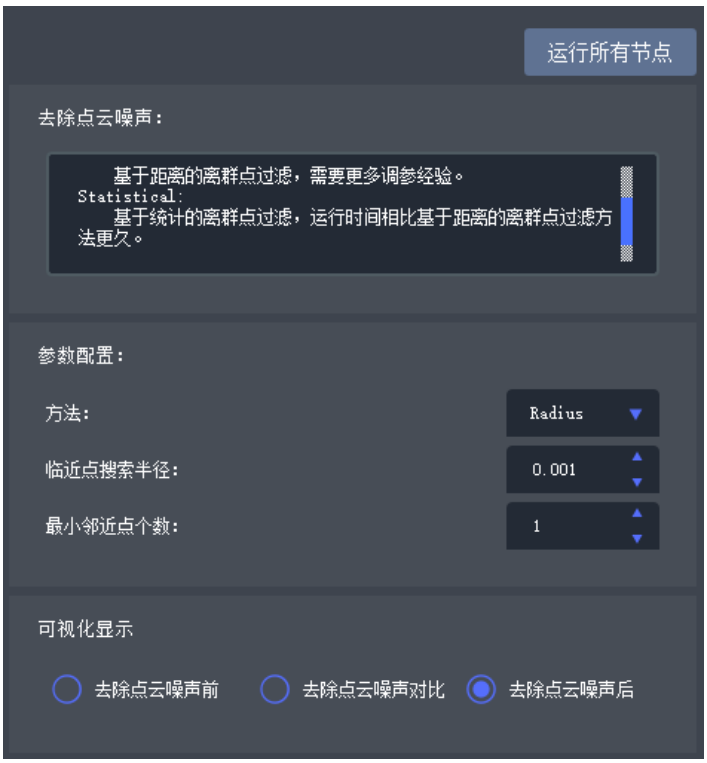
3D 深筐乱序抓取场景的标准视觉流图配置如下。

1. 创建工作空间，使用 **3D 深筐乱序抓取**模板创建流图，请参见创建流图。创建成功的视觉流图如下。



2. 配置 **相机**模块，获取彩色图、深度图及点云。请参见连接相机。
3. 配置 **加载标定信息**模块，请参见标定。
4. 配置 **变换点云坐标系**和 **去除料筐外点云**模块，将点云从相机坐标系转换到工作空间坐标系，并去除料筐以外的点云。

5. 配置 **降采样点云**、**去除点云噪声**、**变换点云坐标系**模块，合理降低点云数量，去除噪声，并将点云从工作空间坐标系转换到世界坐标系。



6. 基于步骤 4 得到的料筐内点云，配置 **变换点云坐标系**模块，将点云从工作空间坐标系转换到相机坐标系。
7. 结合步骤 3 得到的彩色图，配置 **3D 模板匹配**模块，根据物体模型在点云中寻找匹配结果。
8. 配置 **变换物体位姿 3D** 模块，将物体位姿从相机坐标系转换到世界坐标系。
9. 基于步骤 5 得到的世界坐标系下料筐内点云、步骤 6 中 3D 模板匹配得出的模型点云，以及步骤 8 得到的世界坐标系下 3D 物体位姿，配置 **去除识别工件的点云**模块并运行。
10. 配置 **输出结果**模块，输出步骤 8 得到的世界坐标系下物体位姿，及步骤 9 得到的物体点云。

11. 重复上述步骤，完成其它料筐的视觉流图的设置。

11.3.3 运动

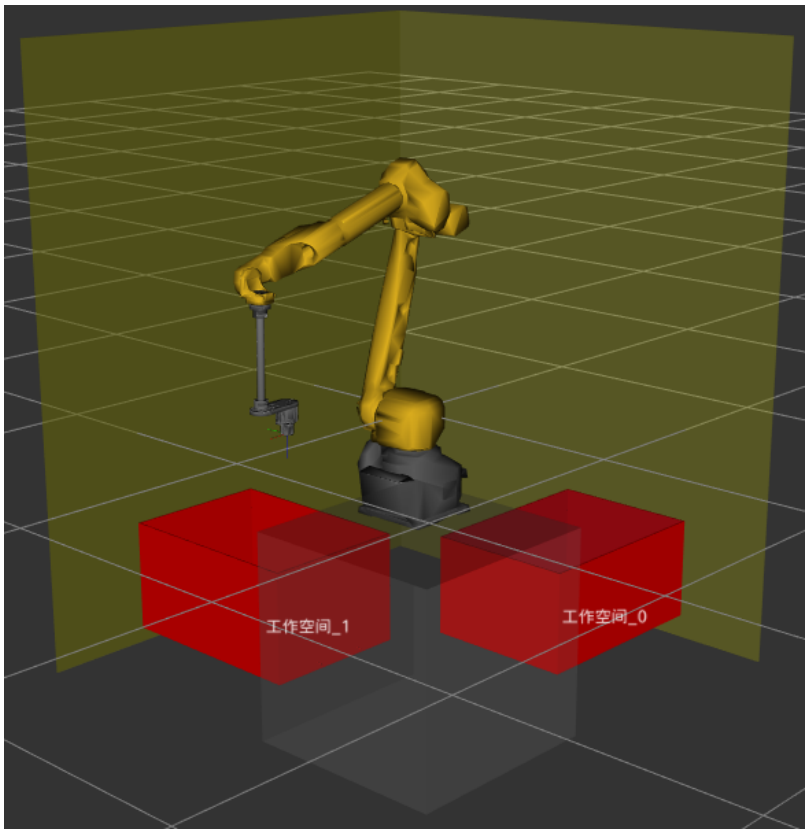
在 运动界面，配置仿真环境并规划运动路径。

提示： 在开始设置运动相关内容前，请先根据机械臂和相机品牌连接并配置真实环境中的机械臂和相机。相关内容请参考连接机械臂 和设置相机 *IP*。如果仅在仿真环境中运行，则无需配置机械臂和相机。

配置仿真环境

根据实际场景，在仿真环境中添加机械臂、料筐、几何体、工具。

1. 创建项目时，已添加机械臂。如未添加，可在 运动页签下快捷工具栏，单击 机械臂，添加机械臂，请参见添加机械臂。
2. 创建项目时，已添加料筐。可在顶部快捷工具栏，单击 工作空间，选择 **添加料筐**，继续添加料筐。本项目需添加两个料筐，添加料筐后需设置料筐及工作空间的尺寸、位姿等参数，请参见添加料筐。
3. 创建项目时，已添加工具，工具自动加载至机械臂末端。如未添加，可在 运动页签下快捷工具栏，单击 末端工具，选择 **注册工具**，注册工具并将工具添加至机械臂，具体参见下一节。
4. 在顶部快捷工具栏，单击 几何体，根据实际场景，添加几何体，用于表示环境中的相机、障碍物等物体。



注册抓取方式

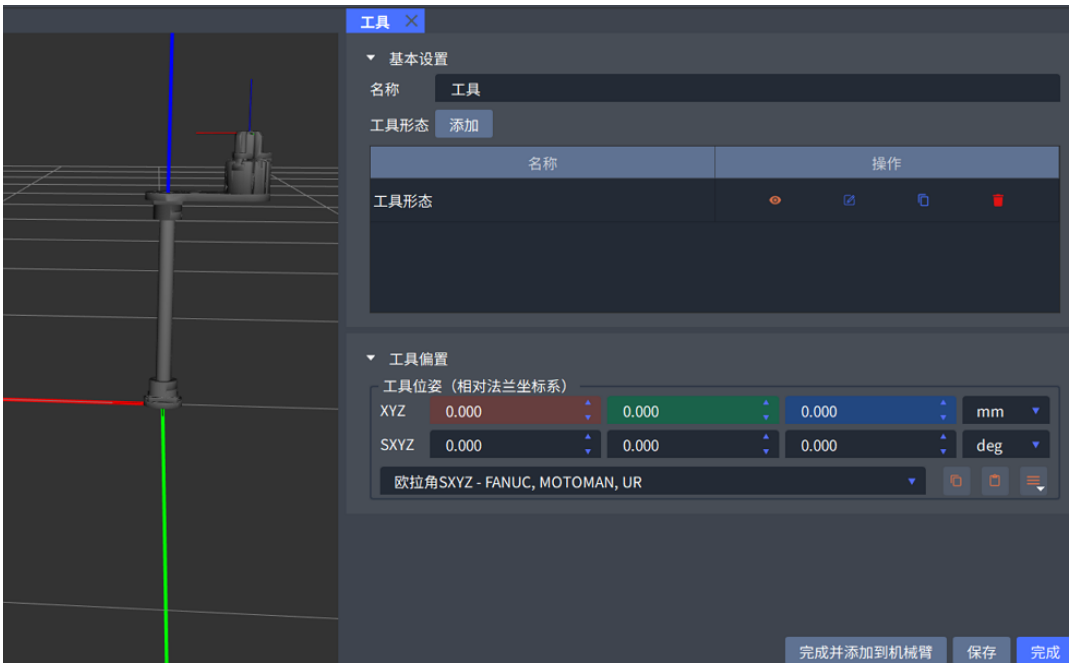
利用工具和工件，注册抓取方式。

1. 注册工具，具体操作参见注册工具。

a. 在 运动页签下快捷工具栏，单击 末端工具，选择 **注册工具**，

b. 单击左下角 **工具**节点，在右侧参数设置栏添加工具形态，设置可视化模型和碰撞模型。

c. 添加工具参考点，然后单击右下角 完成，回到 **工具形态**页签，再单击右下角 完成，回到 **工具**页签，单击 完成并添加到机械臂。



2. 注册工件。在顶部快捷工具栏，单击 工件，选择 **注册工件**。使用 STL 模型注册工件，请参见注册工件。如果在创建项目时已添加工件的 STL 模型，可跳过此步骤。

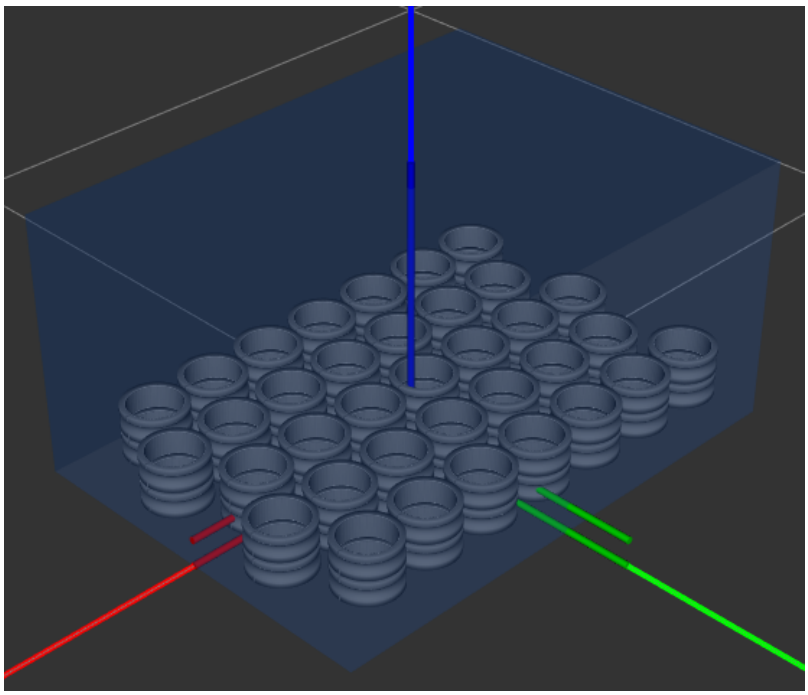
3. 注册抓取方式。在顶部快捷工具栏，单击 抓取和放置，选择 “注册抓取方式 > 工件抓取注册”。使用已注册的工具和工件，设置抓取注册集，请参见注册抓取方式。

规划垛型

注册码垛垛型。

1. 在 *运动*页签下快捷工具栏，单击 *抓取和放置*，选择“注册放置方式 > 工业单码垛型注册（手动）”。
2. 单击界面左下角 **工业单码**节点，在右侧参数设置栏 **基本设置**下方，设置垛型名称、工作空间、工件类型等基本参数。
3. 在 **垛型规则设置**下方，参考*工业单码垛型注册（手动）*，设置垛型规则。主要需设置以下参数：
   a. 设置码放空间尺寸、相邻工件中心距离。
   b. 选择蜂窝型垛型。
   c. 设置码垛层数、每层高度、工件相对该层底面 z 方向值。

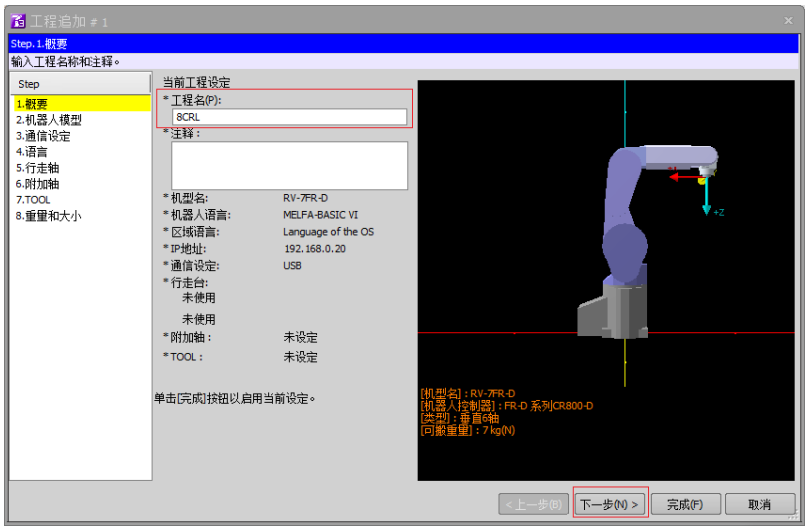
4. 单击 自动计算垛型，在中间预览窗口查看生成的垛型，确认无误后单击右下角 完成。



设置抓放组合规划

注册抓放组合处理流程，用于任务规划时读取抓放方式。

1. 在 运动页签下快捷工具栏，单击 抓取和放置，选择“注册抓放规划 > 组合处理流程”。
2. 单击界面左下角 **抓放组合处理流程**节点，在右侧参数设置栏 **抓取设置**页签下完成抓取相关设置。
   a. 工件处理策略选择 **无处理**，抓取注册集选择之前注册的抓取注册集。
   b. 抓取规划处理策略选择 **[排序] 根据混合策略排序抓取规划**，然后在排序处理列表中添加 **[排序] 根据物体坐标排序抓取规划**，修改排序精度，将物体坐标轴设置为 Z 轴，将参考坐标系设置为工作空间。
   c. 参考上一步骤，在排序处理列表中再添加两次 **[排序] 根据物体坐标排序抓取规划**，将物体坐标系分别设置为 X 轴和 Y 轴，并配置其余参数。



   d. 在抓取规划处理策略中添加 **[合并] 合并抓取物体相同的抓取规划**，将多个抓取规划合而为一，然后单击 下一步。
3. 在 **放置设置**页签下，完成放置相关设置。
   a. 放置位姿来源和工作空间处理策略保持默认即可。
   b. 将 **抓放组合方式**设置为 **所有组合**。
   c. 抓放组合处理策略保持默认即可。
   d. 单击 下一步。

4. （可选）选择抓取和放置工作空间，单击 测试运行，查看运行结果。

5. 单击右下角 完成。

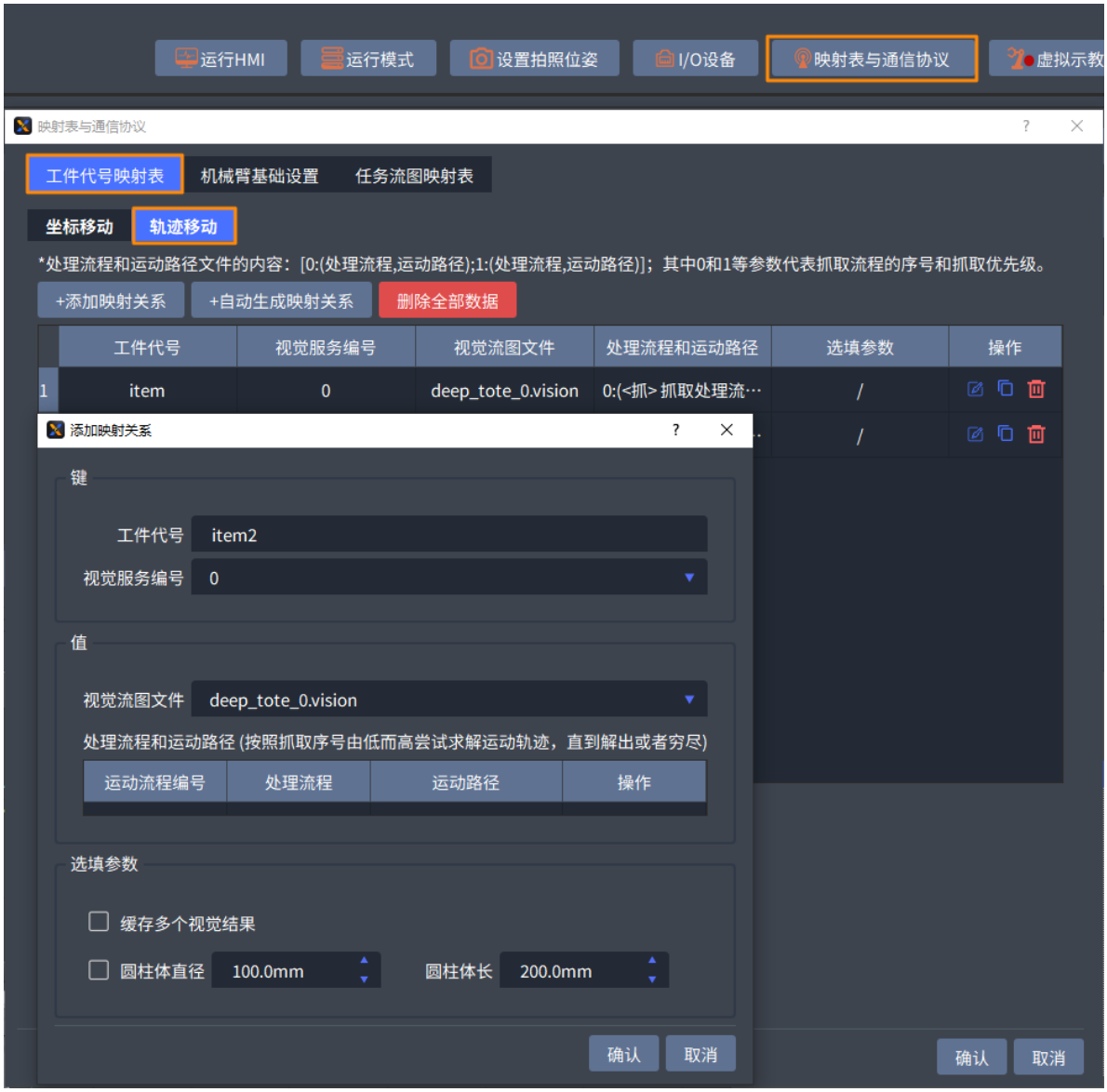
11.3.4 任务

任务部分利用视觉部分的计算结果和运动部分的运动规划，控制整个项目的流程运行。本章节将详细介绍任务部分的配置。

工件代号映射表

设置完视觉和运动部分之后、运行任务流图之前，需要添加工件代号映射关系，以便任务流图调用视觉和运动设置的工件、视觉服务编号、抓取处理流程等内容，具体步骤如下。

1. 单击 Max 右上角 *映射表与通信协议*，在弹出的 **映射表与通信协议**窗口选择“工件代号映射表 > 轨迹移动”页签。
2. 单击 *添加映射关系*，在弹出的 **添加映射关系**窗口设置键、值和参数。



- 工件代号：用户自定义工件代号，仅支持英文字母和数字，长度不超过 15 个字符。可添加多个工件。

- 视觉服务编号：在下拉菜单选择视觉服务编号，编号和视觉部分的设置保持一致。
- 视觉流图文件：与视觉空间 ID 下的流图保持一致。
- 处理流程：在下拉菜单中选择抓取处理流程，这些抓取处理流程已在运动部分设置好。运动流程编号从 0 开始自动增加，可单击  或  新建或删除处理流程。
- 运动路径：选择在 *运动*中设置好的一条路径。
- 缓存多个视觉结果：可根据需要是否勾选。
- 圆柱体：本项目无需勾选。与圆柱体相关的项目可根据需求，设置圆柱体的直径和长，无需重新注册圆柱体的抓取方式，方便使用。

3. 单击 *确认*完成添加，列表中会显示已添加的映射关系，用户可对列表中的映射关系进行操作。

-  ：修改映射关系。
-  ：复制并修改映射关系，注意键值不能重复。
-  ：删除映射关系。

4. 单击 *确认*完成映射表设置。

提示： 用户可单击 *自动生成映射关系*，系统自动根据视觉和运动的设置生成工件代号映射表，但有多个抓取处理流程时无法自动生成映射关系。

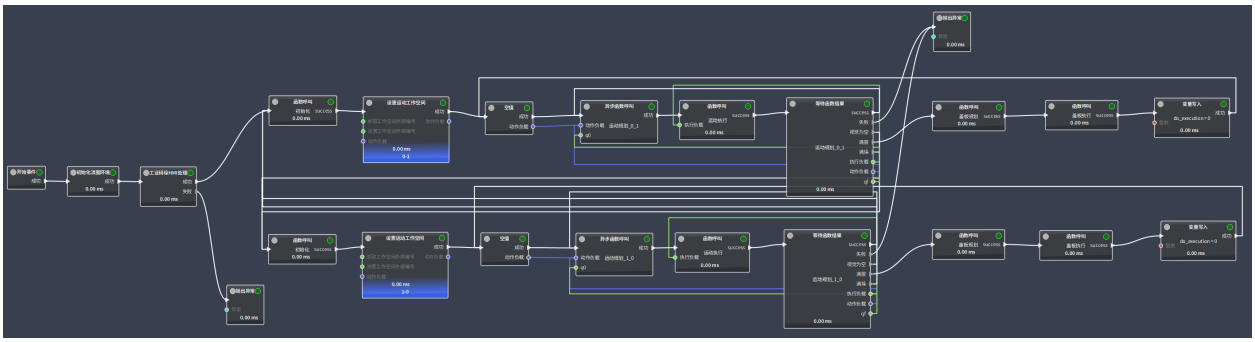
设置完工件关系映射表之后，在 *任务*中导入 3D 轨迹移动中的模板，请参见流图配置。任务流图的设置分为工控机主控和机械臂主控两种方式，每种方式下含多个模板，可根据需要选择。在本章节工控机主控使用 **抓取 + 工业码垛异步**模板，机械臂主控使用 **抓取异步前端**和 **抓取异步后端**模板。

工控机主控

添加好基础模板之后，用户可根据需求添加或修改模块。以工业码垛项目为例，任务流图主要分为四个部分：主程序、运动规划、运动执行和其它功能，下面依次介绍这四个部分。主程序控制整体流程，运行过程中会调用另外三个部分执行相应功能。

主程序

主程序控制整个任务流程，主程序配置好之后如下图所示。



下面详细介绍主程序的配置步骤。

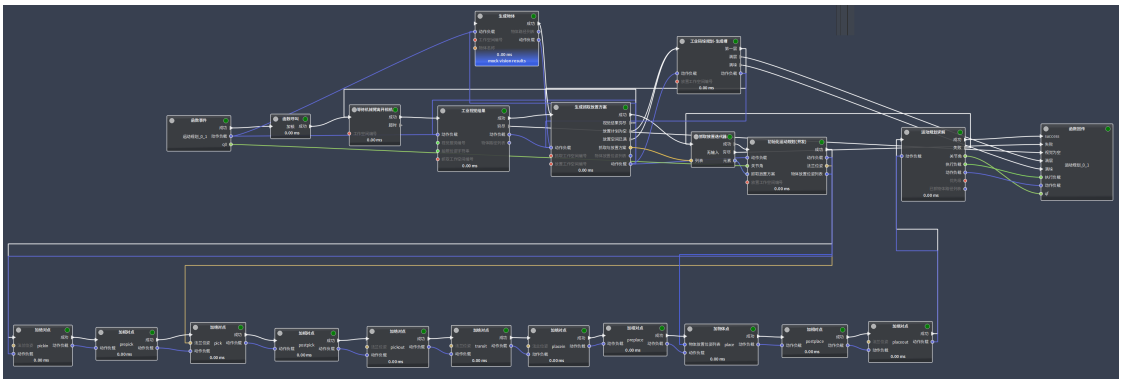
1. 在任务流图模块当中选择“功能 > 一键更新节点机械臂属性”，系统自动根据当前设置更新机械臂路径。也可在模块的属性中设置机械臂路径。

2. 单击 **设置运动工作空间**模块，在右侧 **属性**页签中设置 **抓取工作空间外部编**和 **放置工作空间外部编号**，分别与抓取、放置的视觉 ID 保持一致。

3. 单击 **异步函数呼叫**模块，在右侧 **属性**页签中选择 **函数名**。选择时注意抓取的方向，例如“运动规划_0_1”是从工作空间 0 抓取到工作空间 1。

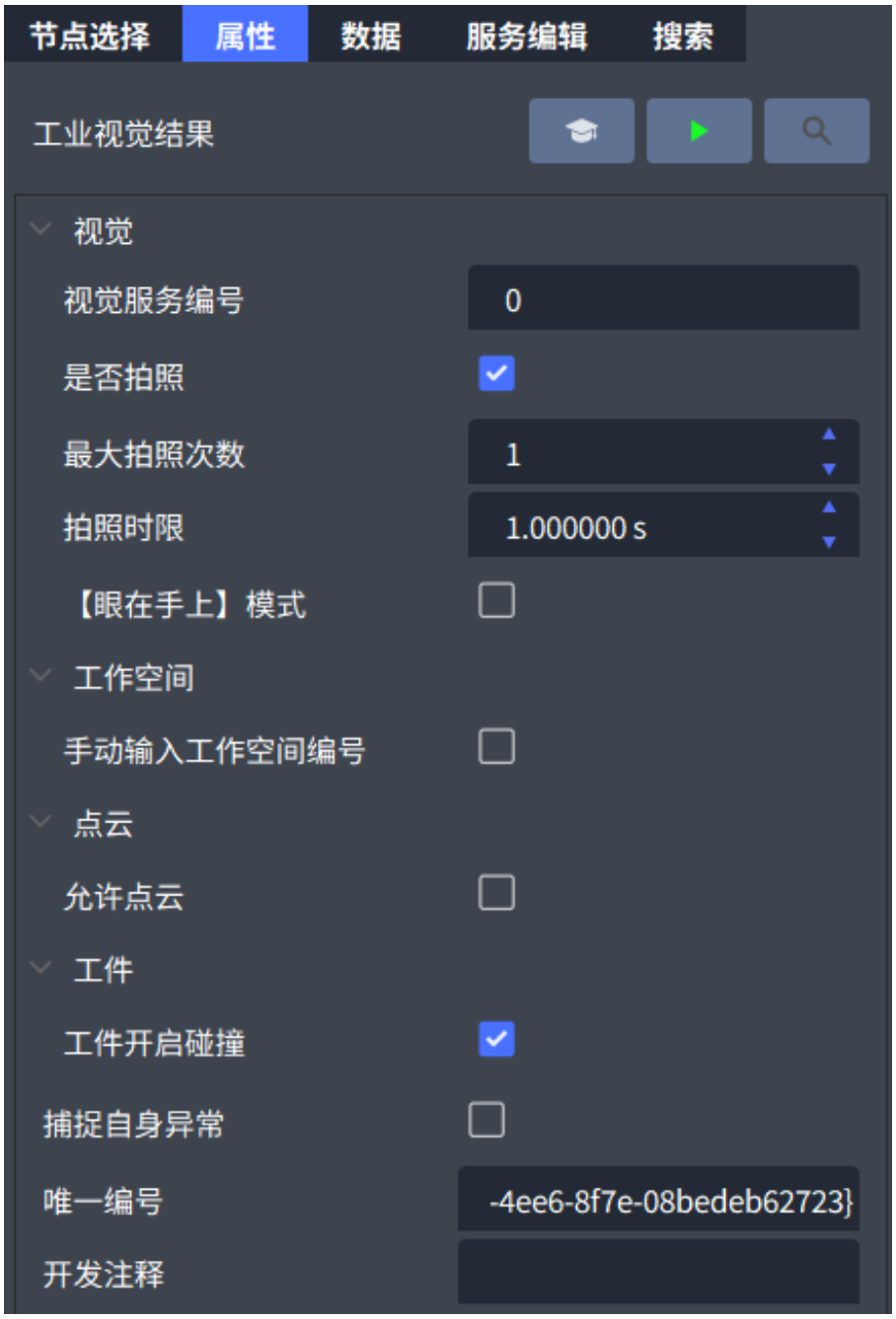
运动规划

运动规划主要负责机械臂路径控制，运动规划配置好之后如下图所示。



下面详细介绍运动规划的配置步骤。

1. 单击 **工业视觉结果**模块，在右侧 **属性**页签设置视觉服务编号，这里设置为抓取工作空间的视觉服务编号。



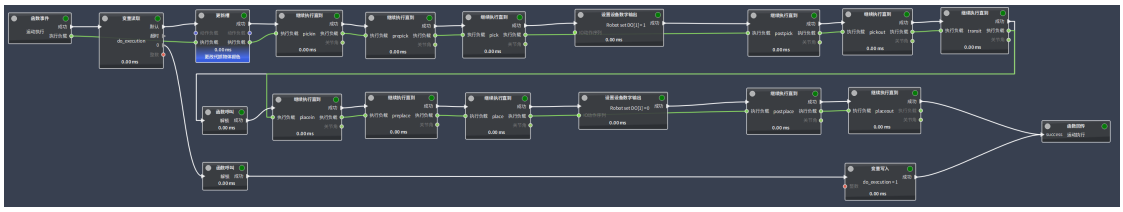
2. 单击 **生成抓取放置方案**模块，在右侧 **属性**页签设置 **组合规划配置文件路径**，这里直接选择在 运动部分设置好的抓放组合处理流程文件路径即可。

3. 单击 **工业码垛规划-生成槽**模块，在右侧 **属性**页签设置 **工件文件路径**，选择工件模型文件所在路径。再设置 **手动码垛注册文件路径**，这里直接选择在 运动部分设置好的放置注册文件路径即可。

提示： 如果在两个料筐中来回抓取，则需按上述方法再配置一条运动规划流图。

运动执行

运动执行主要负责机控制械臂运动过程中的具体点位以及夹具的开合，运动执行配置好之后如下图所示。



下面详细介绍运动执行的配置步骤。

1. 根据 *运动*中的路径规划，依次设置 **继续执行直到**模块，在右侧 **属性**页签中将 **目标点名称**设置为与路径规划中相同的名称，模块数量可根据实际情况修改。

2. 单击 **设置设备数字输出**模块，在右侧 **属性**页签设置 IO 动作序列，选择端口号和值。例如在本项目中，控制工具开合的端口号为 1，值为 1 表示工具张开，值为 0 表示工具闭合。

提示： 单击 Max 右上角的 虚拟示教器，在 **I/O 控制**页签根据机械臂实际情况设置数字输出端口号。例如在本项目中端口 3 控制的是工具的开合，信号 0 表示工具闭合，信号 1 表示工具张开。端口名称可自行定义，然后单击 确认。在 **设置设备数字输出**模块设置时，选择 I/O 控制里的端口即可。

其它功能

该部分主要包含初始化、盖板规划和执行、加锁解锁的功能，主程序会调用这些功能。如下是该部分任务流图。

1. 在“初始化”的任务流图中，完成如下设置。
    a. 单击 **移动到目标关节**模块，在右侧 **属性**页签中设置目标关节角、位置和速度等参数，即起始点位置。
    b. 单击 **创建运动工作空间编号到视觉服务编号映射**设置映射内容，一般与视觉服务编号保持一致即可。

2. 在“盖板规划和执行”的任务流图中，单击 **延迟**模块设置 **延迟秒数**。
3. 在“加锁解锁”的任务流图中，完成如下设置。
    a. 单击 **变量读取**设置变量参数，包括变量名、类型等。

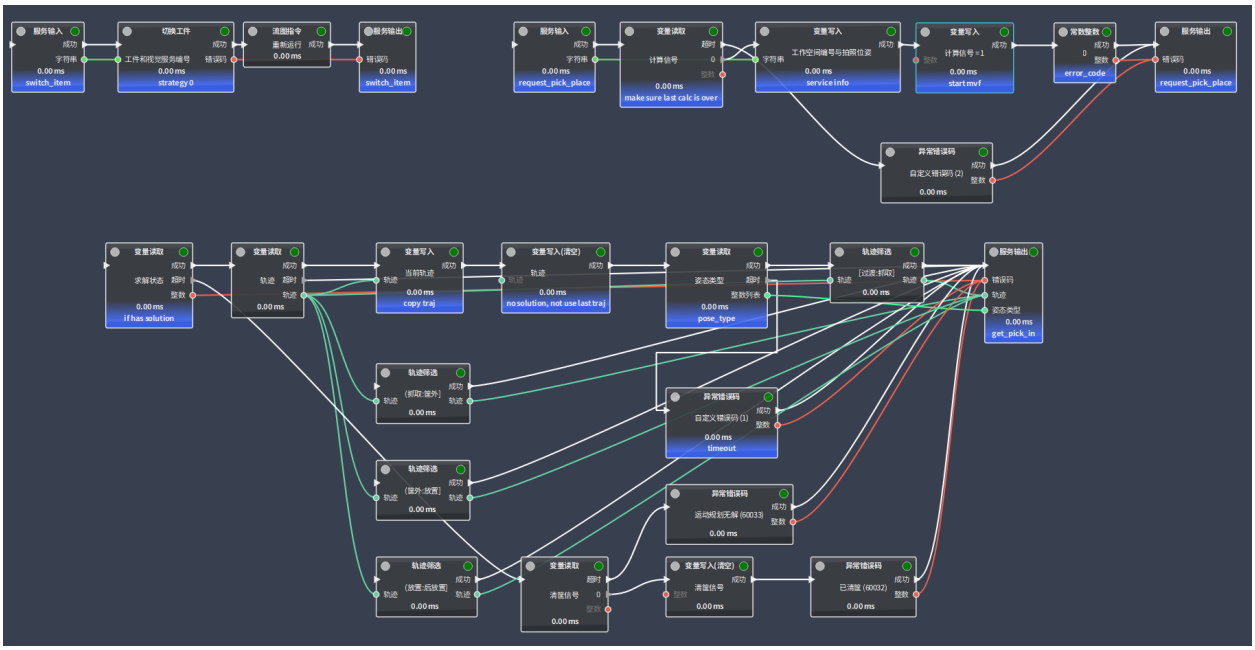
b. 单击 **变量写入**设置变量默认值，例如在本项目中 0 表示加锁，1 表示解锁。

机械臂主控

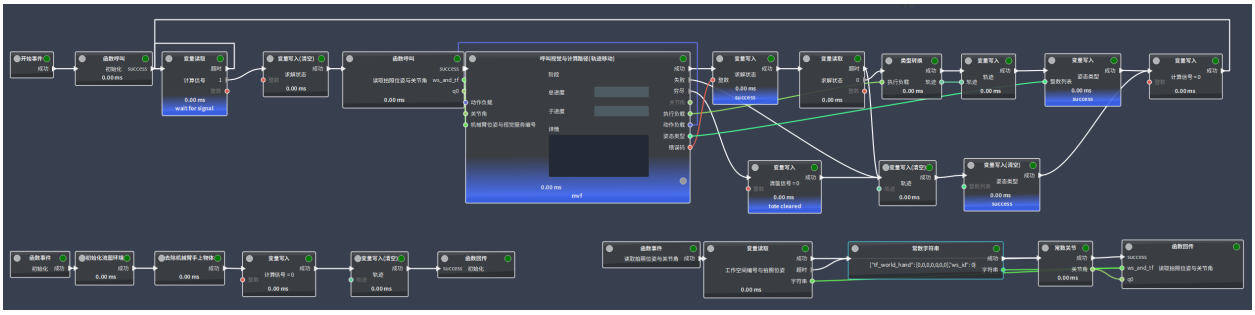
在机械臂主控模板中，需要同时使用 **抓取异步前端**和 **抓取异步后端**两个模板。

其中前端模板主要负责整体抓放流程的控制，而后端完成大量的计算工作，例如机械臂抓取出筐后可以立即拍照，后端可以立刻执行计算任务，这样机械臂完成一次抓放流程之后可以直接获取计算结果进行下一次抓取，以提升执行效率。前端被机械臂呼叫，对指令的响应速度更快，但前端不负责具体的计算任务，仅从后端获取计算结果。

下图是 **抓取异步前端**模板示意图，其中左上部分控制切换工件的流程，右上部分控制请求抓放的流程，下面的流程控制抓放节点筛选。



下图是 **抓取异步后端**模板示意图，其中上面是计算抓取轨迹的流程，左下部分控制工作空间的切换，右下部分控制坐标系转换。



机械臂主控模式下，Fanuc 机械臂案例和示例代码可参考案例/模板说明，其他品牌的案例和代码可在安装机械臂驱动 中选择相应的品牌查看。

在 Max 中，单击右上方 映射表与通信协议，在弹出窗口的 机械臂基础设置页签下，勾选 **开启**打开测试模式。此时右侧会出现 打开文件和  。单击 打开文件可查看通信指令的 yml 文件，单击  可在浏览器中打开机械臂主控通讯协议 v2 页面，并查看机械臂主控通信协议、指令等的内容。


运行HMI
运行模式
设置拍照位姿
I/O设备
映射表与通信协议
虚拟示教器
映射表与通信协议
工件代号映射表
机械臂基础设置
任务流图映射表
机械臂ID
0: Fanuc-M-10iD-8L
厂商
Fanuc
控制器
R30iA
型号
M-10iD-8L
欧拉角类型
SXYZ
通讯方式
TCP-Server协议
测试模式
开启
打开文件
*修改测试文件后机械臂主控驱动需要被重启
通讯数据格式
分割符
,
结束符
#
端口
11111
小数点位数
关节
2
位置(XYZ)
2
姿态(欧拉角/四元数)
2
确认
取消

CHAPTER 12

拆码垛场景

本章介绍拆码垛场景的参数和流程配置。

12.1 视觉

本章节将针对具体场景详细介绍视觉配置过程。

12.1.1 纸箱单拆

纸箱单拆是对托盘上单一种类的物体进行拆垛的场景，视觉功能实现对托盘物体信息的感知，辅助机械臂定位*SKU* 的位姿、尺寸等信息。

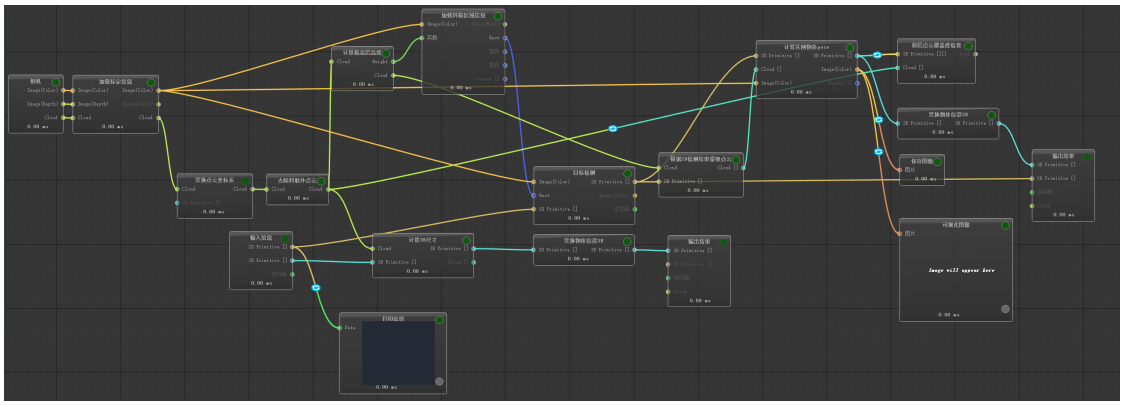
场景要求

- 纸箱紧密摆放
- 纸箱表面纹理未知
- 纸箱尺寸大小相同

该场景需要使用 3D 相机得到的彩色图进行深度学习识别，同时使用 3D 相机得到的点云进行位姿计算。

流图配置

纸箱单拆场景的视觉标准流图如下。



1. 创建工作空间，使用 **纸箱单拆**模板创建流图，具体方法请参见模块结构。
2. 配置 **相机**模块，获取彩色图、深度图及点云。详细设置请参考连接相机。
3. 配置 **加载标定信息**模块，加载 env 文件中的标定信息。详细设置请参考标定。
4. 配置 **变换点云坐标系**和 **去除料筐外点云**模块，将点云从相机坐标系转换到料筐坐标系，并去除料筐以外的点云。在进行点云切割时，需要测量料筐的实际尺寸，并输入 Max。

5. 配置 **计算最高层高度**模块，根据层厚对点云进行分层，计算出最高层点云的高度，再结合步骤 3 输出的彩色图，配置 **加载料箱区域信息**模块，计算 2D 图像中给定高度上的工作空间。
6. 通过 **输入信息**模块输入客户端提供的 2D primitive，结合步骤 3 输出的彩色图和步骤 5 输出的矩形图像，配置 **目标检测**模块以检测目标物体，并输出物体的 2D Primitive 计算结果。
7. 计算实例物体位姿。
    a. 结合步骤 5 中 **计算最高层高度**模块输出的点云和步骤 6 输出的 2D primitive 计算结果，使用 **根据 2D 检测结果提取点云**模块输出点云。
    b. 利用该点云，再结合步骤 3 输出的彩色图和步骤 6 输出的 2D primitive，配置 **计算实例物体 pose** 模块，使用深度学习结果计算位姿，并输出 3D primitive 计算结果，如下图所示。

8. 检查并输出结果。
   - 结合步骤 4 输出的点云和步骤 7 输出的 3D primitive，配置 **顶层点云覆盖度检测**模块，对物体进行检测。
   - 使用 **变换点云坐标系**模块，将步骤 7 输出的 3D primitive 计算结果由料筐坐标系转换到世界坐标系，并结合步骤 6 输出的 2D Primitive，在 **输出结果**模块查看输出。
   - 使用 **保存图像**和 **可视化图像**模块，保存或查看步骤 7 **计算实例物体 pose** 模块输出的彩色图。
9. 通过 **输入信息**模块输入客户端提供的 3D primitive，结合步骤 4 输出的点云，通过 **计算 3D 尺寸**模块计算物体尺寸，然后使用 **变换物体位姿 3D** 模块将计算结果由料筐坐标系转换到世界坐标系，最后通过 **输出结果**模块查看处理后的 3D primitive。

12.1.2 拆麻袋

拆麻袋是将托盘上的麻袋拆放到指定位置，托盘上的麻袋大小、种类、形状均不固定。

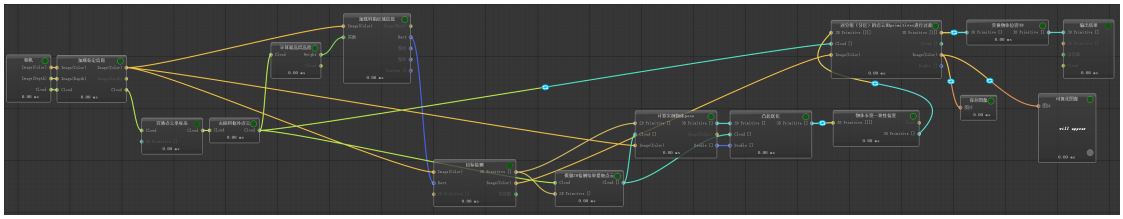
场景要求

- 麻袋紧密摆放
- 麻袋表面纹理未知

该场景需要使用 3D 相机得到的彩色图进行深度学习识别，同时使用 3D 相机得到的点云进行位姿计算。

流图配置

拆麻袋场景的视觉标准流图如下。



1. 创建工作空间，使用 **拆麻袋**模板创建流图，具体方法请参见模块结构。
2. 配置 **相机**模块，获取彩色图、深度图及点云。详细设置请参考连接相机。
3. 配置 **加载标定信息**模块，加载 env 文件中的标定信息。详细设置请参考标定。
4. 配置 **变换点云坐标系**和 **去除料筐外点云**模块，将点云从相机坐标系转换到料筐坐标系，并去除料筐以外的点云。在进行点云切割时，需要测量料筐的实际尺寸，并输入 Max，如下图所示。

5. 配置 **计算最高层高度**模块，根据层厚对点云进行分层，计算出最高层点云的高度，再结合步骤 3 输出的彩色图，配置 **加载料箱区域信息**模块，计算 2D 图像中给定高度上的工作空间。
6. 结合步骤 3 输出的彩色图和步骤 5 输出的矩形图像，配置 **目标检测**模块以检测目标物体，并输出物体的 2D Primitive 计算结果和彩色图。
7. 计算实例物体位姿。
    a. 结合步骤 4 中 **去除料筐外点云**模块输出的点云和步骤 6 输出的 2D primitive 计算结果，使用 **根据 2D 检测结果提取点云**模块输出点云。
    b. 利用该点云，再结合步骤 3 输出的彩色图和步骤 6 输出的 2D primitive，配置 **计算实例物体 pose** 模块，使用深度学习结果计算位姿，并输出 3D primitive 计算结果，如下图所示。

8. 综合 **根据 2D 检测结果提取点云**模块输出的点云、**计算实例物体 pose** 模块输出的 3D primitive 和物体内点比例，使用 **凸起优化**模块对图片进行处理，并输出 3D primitive 计算结果。
9. 使用 **物体长宽一致性检查**模块对上一步的输出结果进行检查，并输出 3D primitive。
10. 结合步骤 4 输出的点云和步骤 6 输出的彩色图，以及上一步的输出结果，配置 **对分组（分区）的点云和 primitives 进行过滤**模块，输出 3D Primitive 计算结果和彩色图。
11. 输出或保存结果。
    - 针对步骤 10 中输出的 3D primitive 使用 **变换物体位姿 3D** 模块，将计算结果从料筐坐标系转换到世界坐标系，并使用 **输出结果**模块查看。
    - 使用 **保存图像**和 **可视化图像**模块，保存或查看步骤 10 输出的彩色图即可。

12.1.3 类料箱拆垛

类料箱是指形似料箱的物体，例如料箱、圆桶、轮胎等等。类料箱拆垛是将托盘上的该类物体拆放到指定位置。

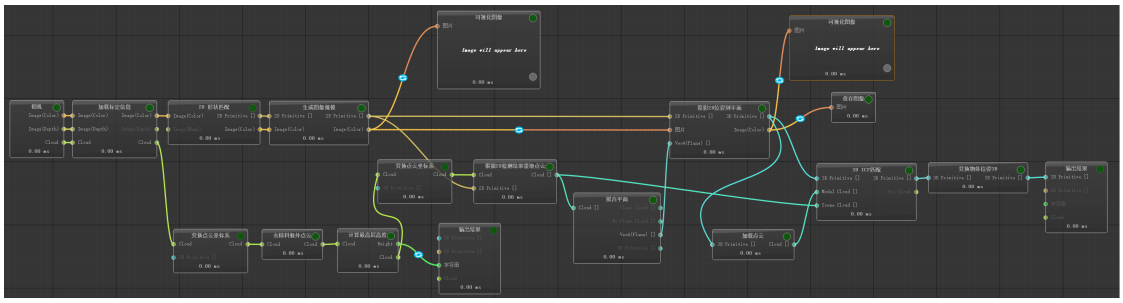
场景要求

- 类料箱物体平铺，且紧密摆放
- 物体为同种已知型号的类料箱物体

该场景需要使用 3D 相机得到的彩色图进行 2D 模板匹配，同时使用 3D 相机得到的点云和物体 CAD 模型进行点云匹配。

流图配置

类料箱拆垛场景的视觉标准流图如下。



1. 创建工作空间，使用 **类料箱拆垛**模板创建流图，具体方法请参见模块结构。
2. 配置 **相机**模块，获取彩色图、深度图及点云。详细设置请参考连接相机。
3. 配置 **加载标定信息**模块，加载 env 文件中的标定信息。详细设置请参考标定。
4. 配置 **2D 形状匹配**和 **生成图像掩模**模块，对步骤 3 得到的彩色图像进行处理，得到新的彩色图像和 2D primitive 计算结果。若有需要，可使用 **可视化图像**查看处理后的彩色图。
5. 基于步骤 3 得到的标定后点云，配置 **变换点云坐标系**和 **去除料筐外点云**模块，将点云从相机坐标系转换到料筐坐标系，并去除料筐以外的点云。在进行点云切割时，需要测量料筐的实际尺寸，并输入 Max，如下图所示。

6. 配置 **计算最高层高度**模块，根据层厚对点云进行分层，计算出最高层点云的高度。如有需要，可通过 **输出结果**模块查看计算结果。
7. 将步骤 6 得到的计算结果，输入 **变换点云坐标系**，将点云从料筐坐标系转换到相机坐标系。然后将点云输入到 **根据 2D 检测结果提取点云**模块，再结合步骤 4 生成的 2D primitive 图像掩模，将输入点云提取成和物体一一对应的点云数组。
8. 将步骤 7 得到的点云输入 **拟合平面**模块，通过拟合计算得到平面向量。
9. 将步骤 4 得到的彩色图和 2D primitive 计算结果，以及步骤 8 得到的平面向量，输入到 **投影 2D 位姿到平面**模块，计算得到 3D primitive 和彩色图像。若有需要，可使用 **可视化图像**和 **保存图像**模块查看或保存处理后的彩色图。
10. 过滤并输出检测结果。

a. 将步骤 9 得到的 3D primitive 计算结果输入 **加载点云**模块，生成点云。

b. 将步骤 7 得到的点云（作为场景点云）、步骤 9 得到的 3D primitive 计算结果、上一步 **加载点云**模块得到的点云（作为模型点云）输入 **3D ICP 匹配**，模块会将模型点云对齐到场景点云，并计算得到 3D primitive。

c. 配置 **变换点云坐标系**，将上一步 **3D ICP 匹配**模块得到的 3D primitive 从相机坐标系转换到世界坐标系，然后通过 **输出结果**模块查看最终结果。

12.1.4 类料箱拆垛（高精度）

类料箱是指形似料箱的物体，例如料箱、圆桶、轮胎等等。类料箱拆垛是将托盘上的该类物体拆放到指定位置。与普通类料箱拆垛场景相比，类料箱拆垛（高精度）场景需要装备高精度相机（如结构光相机），且支持混拆。

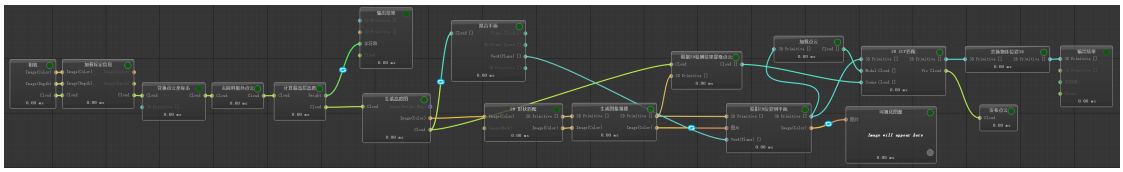
场景要求

- 类料箱物体平铺，且紧密摆放
- 物体为同种已知型号的类料箱物体

该场景需要使用 3D 相机得到的彩色图进行 2D 模板匹配，同时使用 3D 相机得到的点云和物体 CAD 模型进行点云匹配。

流图配置

类料箱拆垛（高精度）场景的视觉标准流图如下。



1. 创建工作空间，使用 **类料箱拆垛**模板创建流图，具体方法请参见模块结构。
2. 配置 **相机**模块，获取彩色图、深度图及点云。详细设置请参考连接相机。
3. 配置 **加载标定信息**模块，加载 env 文件中的标定信息。详细设置请参考标定。
4. 配置 **变换点云坐标系**和 **去除料筐外点云**模块，将点云从相机坐标系转换到料筐坐标系，并去除料筐以外的点云。在进行点云切割时，需要测量料筐的实际尺寸，并输入 Max，如下图所示。

5. 配置 **计算最高层高度**模块，根据层厚对点云进行分层，使用步骤 4 得出的点云计算出最高层点云的高度。如有需要，可通过 **输出结果**模块查看计算结果。
6. 配置 **生成高度图**模块，使用步骤 5 得到的点云，生成可视化高度彩色图和高度图点云。
7. 将步骤 6 得到的点云输入 **拟合平面**模块，通过拟合计算得到平面向量。
8. 将步骤 6 得到的彩色图输入 **2D 形状匹配**和 **生成图像掩模**模块，得到新的彩色图像和 2D primitive 计算结果。
9. 将步骤 8 得到的彩色图和 2D primitive 计算结果，以及步骤 7 得到的平面向量，输入到 **投影 2D 位姿到平面**模块，计算得到 3D primitive 和彩色图像。若有需要，可使用 **可视化图像**模块查看处理后的彩色图。
10. 将步骤 9 得到的 3D primitive 计算结果输入 **加载点云**模块，生成点云。
11. 将步骤 6 得到的点云输入 **根据 2D 检测结果提取点云**模块，再结合步骤 8 生成的 2D primitive 图像掩模，将输入点云提取成和物体一一对应的点云数组。
12. 将步骤 11 得到的点云（作为场景点云）、步骤 9 得到的 3D primitive 计算结果、步骤 10 得到的点云（作为模型点云）输入 **3D ICP 匹配**，模块会将模型点云对齐到场景点云，并计算得到 3D primitive 和可视化点云。该点云由 **发布点云**模块发布。
13. 将步骤 12 得到的 3D primitive 计算结果，通过 **变换物体位姿 3D** 模块，从料筐坐标系转换到世界坐标系，并通过 **输出结果**模块查看最终结果。

12.2 运动

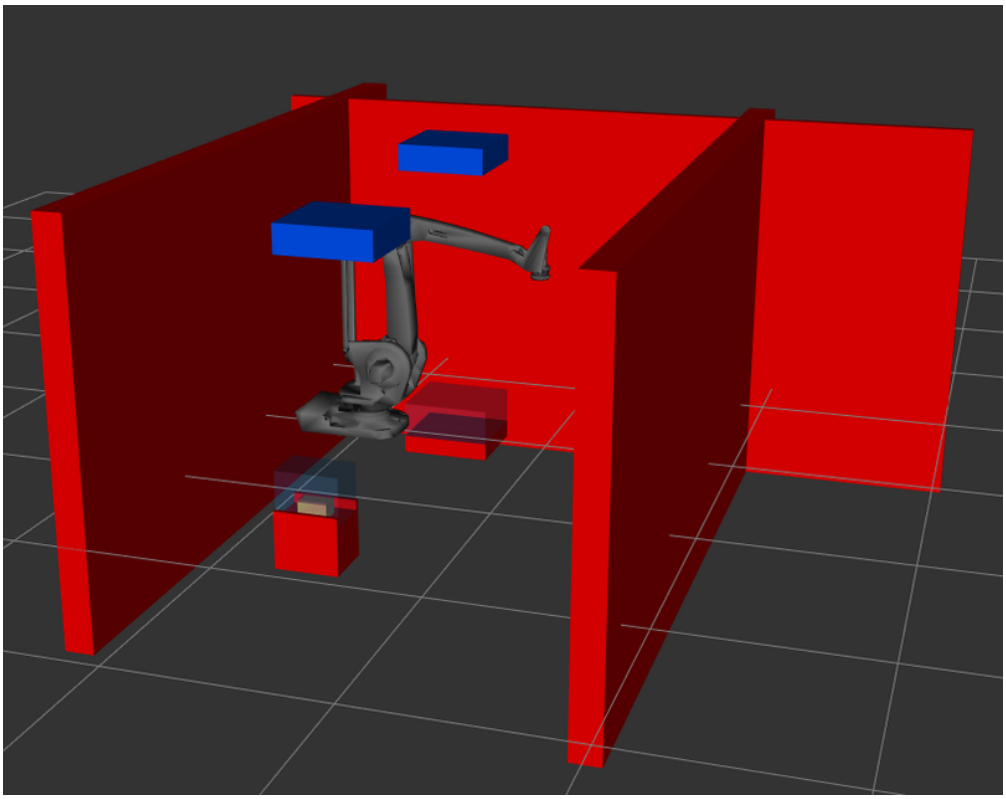
在 运动界面，设置机械臂夹取、放置、碰撞检测、运动路径等。

提示:

- 在开始设置运动相关内容前请先根据机械臂和相机品牌连接并配置机械臂和相机。相关内容请参考连接机械臂 和设置相机 *IP*。
- 配置前，请准备好机械臂等物体的模型文件。

确认机械臂和相机连接无误后，开始配置运动相关设置，具体步骤如下。

1. 在 运动页签下，完成机械臂模型添加，具体方法请参见添加机械臂。添加时注意确认实际所使用的机械臂和模型文件是否一致。
2. 添加吸盘，具体方法请参见注册吸具。
3. 注册抓取和放置方式，具体方法请参见注册抓箱子方式 和注册放置方式。
4. 在世界坐标系中添加托盘、传送带。托盘或传送带的尺寸、位置需要用户根据实际情况设置。操作可参考添加输送线 和添加托盘。
5. 在世界坐标系中添加环境障碍物，环境障碍物的尺寸、位置需要用户手动测量，然后在世界坐标系中设置。最终配置的示例如下图所示。



提示: 在实际项目执行过程中，运动和 视觉界面的设置可交替进行，具体可根据实际情况灵活调整。

12.3 任务

根据实际需要，导入拆码垛的模板，详细操作请参见基本任务操作中的流图配置，并修改模块配置。

本章节仅介绍拆码垛场景特有的模块和流程，其它常用模块的使用和配置请参见任务。

小技巧： 因 任务中模块数量较多且流程图较为复杂，可以在右侧 搜索页签搜索关键字定位模块。

12.3.1 纸箱单码

根据实际需要导入纸箱单码模板，并配置模块参数。去除机械臂手上物体、移动到目标关节、清空工作空间、等待机械臂离开相机、获取视觉结果、初始化运动规划、异步函数呼叫、加绝对点、加相对点、加物体点、动作规划求解、运动执行等模块的配置，可参考任务。此外，该场景还需配置以下模块：

SKU 信息

该模块主要用于生成 SKU 信息，以识别条码朝向。条码朝向主要有以下几种情况：无码朝向、一长边有码、两长边有码、一短边有码、两短边有码。

该模块主要包含 SKU 的长、宽、高，以及重量等信息。对于存在输入的情况下，默认保留输入的数据，并在此基础上增加条码朝向选项。对于没有输入的情况下，默认选择模块设置的 SKU 信息。

单码垛型规划模块

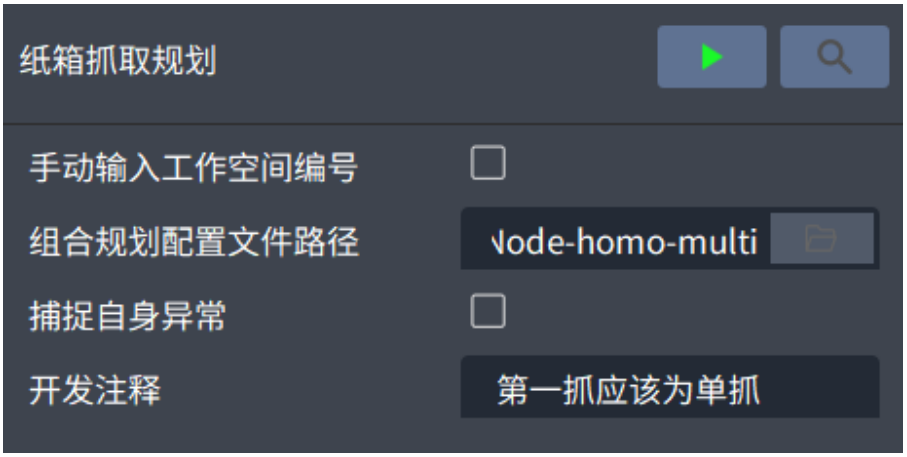
该模块用于计算码垛位置，为 **纸箱抓取规划**模块提供信息。

使用时，需设置工作空间编号、SKU 信息、单码配置文件路径等参数。

纸箱抓取规划

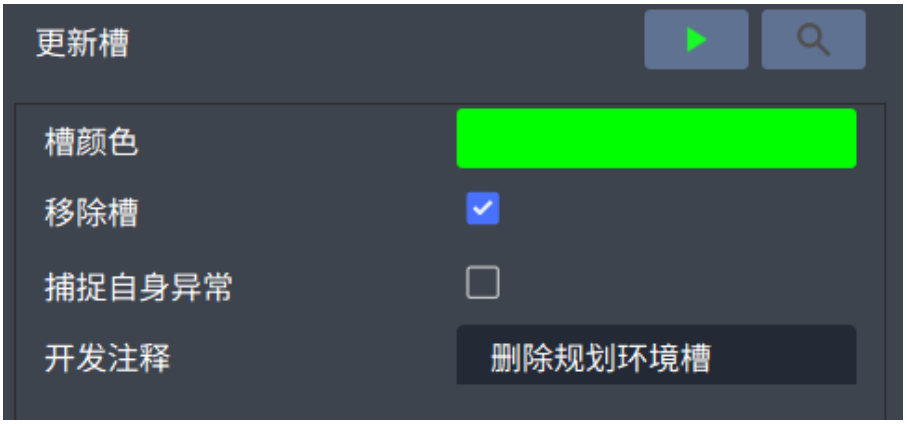
该模块的主要任务是从环境中提取待抓取物体，并计算其抓取规划，然后从环境中提出待放置的位置，并匹配抓取规划和环境中可放置位置。

使用时，需设置工作空间编号、组合规划配置文件路径等参数。



更新槽

该模块的主要任务是更新槽信息，根据实际情况更新物体槽颜色，或者选择是否移除规划环境槽。



模拟视觉结果

该模块的主要功能是模拟在传送带或者托盘上产生 SKU，以便无视觉码垛或者仿真使用。

使用时，需设置工作空间编号，并启用 **物体开启碰撞**，设置机械臂路径等参数。

12.3.2 纸箱单拆

根据实际需要导入纸箱单拆模板，并配置模块参数。去除机械臂手上物体、移动到目标关节、清空工作空间、获取视觉结果、初始化运动规划、异步函数呼叫、加绝对点、加相对点、加物体点、动作规划求解、运动执行等模块的配置，可参考任务。此外，该场景还需配置以下模块：

初始化流图环境

根据需要选择是否初始化变量表以及缓存数据，完成环境初始化。

清空环境

该模块的主要功能是清空环境中的数据，包括可视化及碰撞检测等数据。

使用时，设置 **工作空间路径**、**机械臂路径**，还可根据需要选择是否去掉机械臂手上物体。

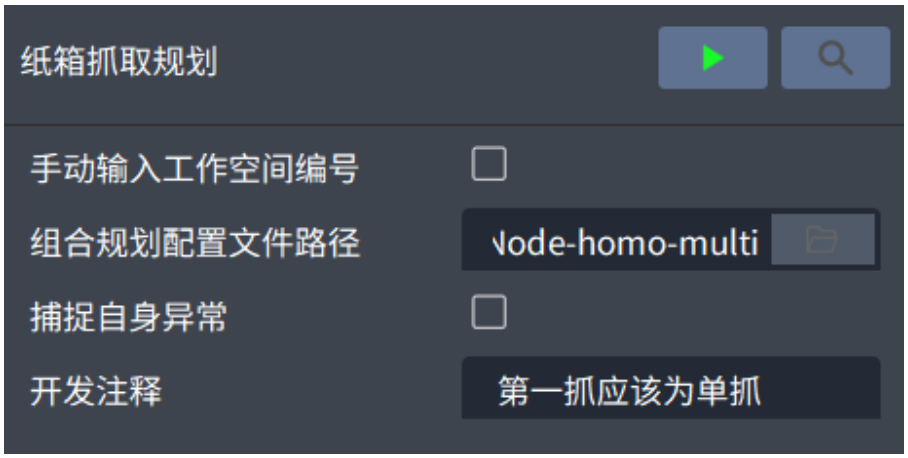
更新最大可抓取数量

更新夹爪本身属性所允许的最大可抓取数量，根据实际情况设置。

纸箱抓取规划

该模块的主要任务是从环境中提取待抓取物体，并计算其抓取规划，然后从环境中提出待放置的位置，并匹配抓取规划和环境中可放置位置。

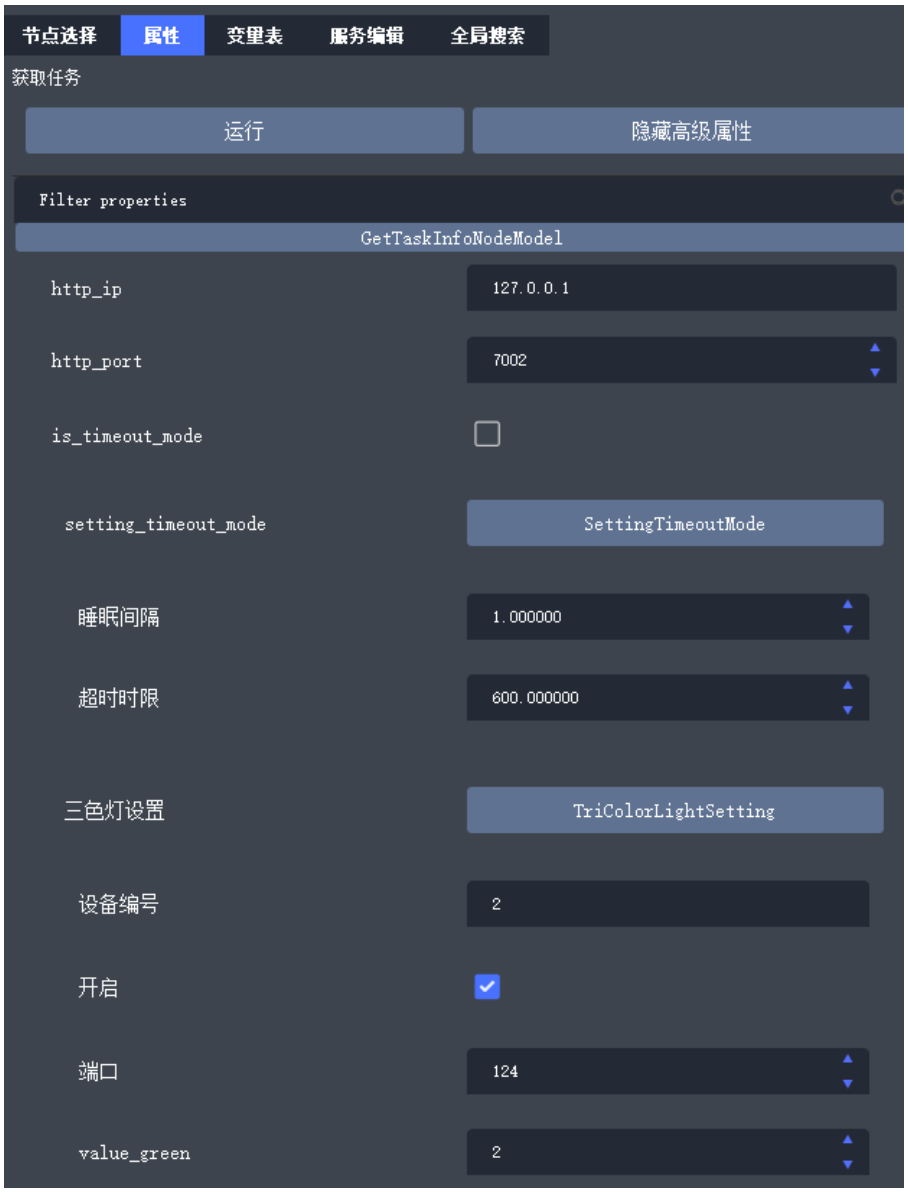
使用时，设置 **抓取工作空间路径、放置工作空间路径、组合规划配置文件路径**。



获取任务

当由 WCS 下发任务时，执行此模块，成功则开始执行任务。

使用时，根据需要设置获取任务的信息，例如超时时限、设备编号、端口等信息。



单码垛型规划模块

该模块用于计算码垛位置，完成的主要任务是计算环境中可放置位置，并给纸箱抓取规划模块提供信息。使用时，设置 **目标工作空间路径、SKU 信息、单码配置文件路径**。

SKU 信息

该模块的主要任务是对 SKU 的信息进行设置和改动，以识别条码朝向。条码朝向主要有以下几种情况：无码朝向、一长边有码、两长边有码、一短边有码、两短边有码。

该模块主要包含 SKU 的长、宽、高，以及重量等信息。对于存在输入的情况下，默认保留输入的数据，并在此基础上增加条码朝向选项。对于没有输入的情况下，默认选择模块设置的 SKU 信息，如下图所示。

模拟视觉结果

该模块的主要功能是模拟在输送线或者托盘上产生 SKU，以便于无视觉码垛或者仿真使用。

使用时，设置 **工作空间路径**，并开启 **物体开启碰撞**，其它参数根据实际需要设置。


节点选择
属性
变量表
服务编辑
全局搜索
模拟视觉结果
运行
隐藏高级属性
Filter properties
MockVisionResultNodeModel
工作空间路径
Assign...
物体开启碰撞
传送带设置
ConveyorConfigParams
barcode_point_to
kAlongX
conveyor_box_alignment
kLowerRight
customized_pose
CustomizedPose
quaternion
x 0.000000
y 0.000000
z 0.000000
w 1.000000
Edit
x_offset
0.000000
y_offset
0.000000
z_offset
0.000000

Part IV

附录

CHAPTER 13

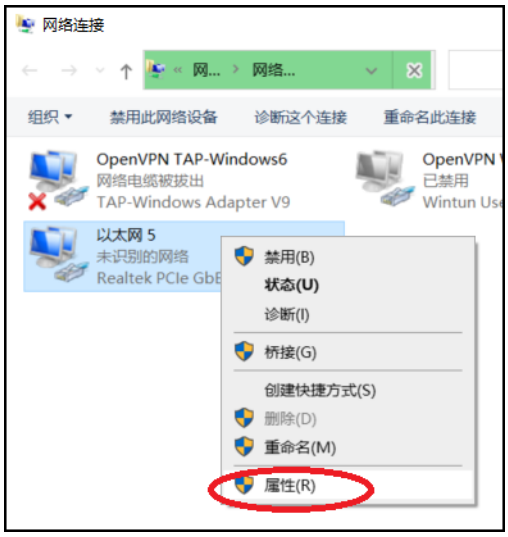
IP 设置

设置工控机和相机 IP 地址，确保两者 IP 在同一网段，以便在 Max 软件上枚举、连接对应相机。

13.1 设置工控机 IP 地址

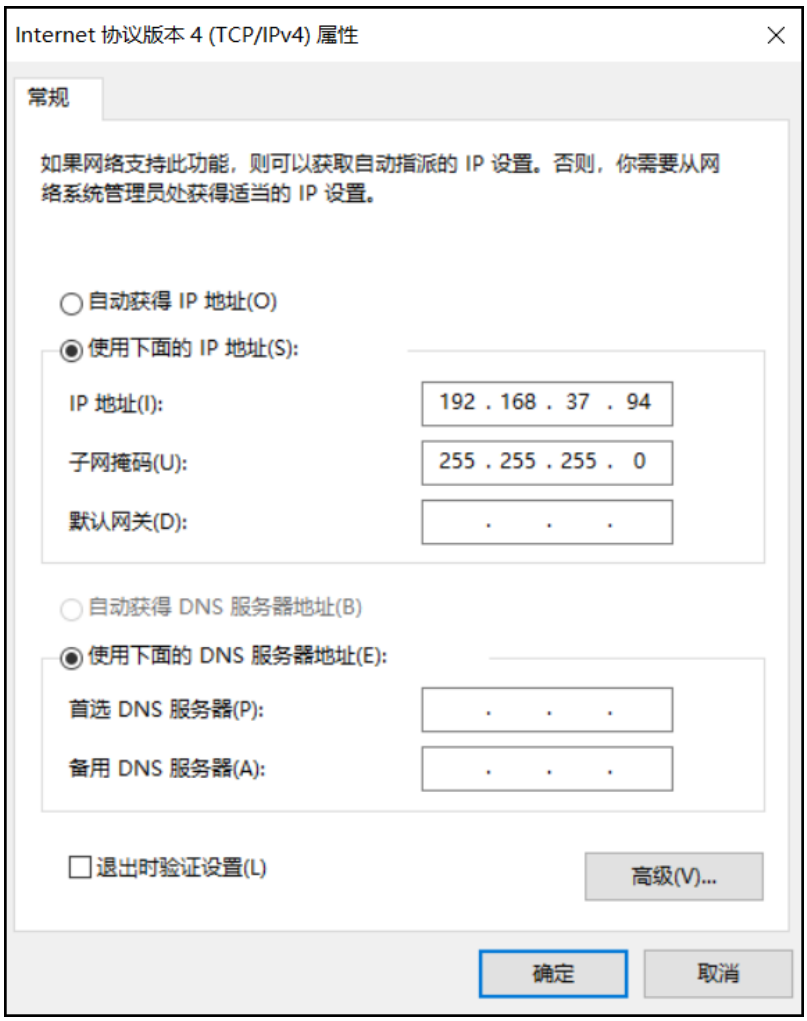
在设置相机 IP 前，需先修改连接相机的工控机网卡的静态 IP，确保网卡 IP 处于相机的目标网段。例如：相机目标 IP 为 192.168.37.111/255.255.255.0，则可将工控机网卡 IP 设置为 192.168.37.101/255.255.255.0，注意不要与其他设备 IP 冲突。

1. 在 Windows 桌面左下角搜索框搜索并打开“控制面板”。
2. 选择“网络和 Internet > 网络和共享中心 > 更改适配器设置”。
3. 右键单击要修改的网卡，选择 **属性**。



4. 单击 *Internet 协议版本 4(TCP/IPv4)*，再单击 *属性*。

5. 选择 **使用下面的 IP 地址**，设置 IP 地址、子网掩码，网关和 DNS 建议不作设置。



13.2 设置相机 IP

设置完工控机 IP 后，修改相机 IP，使两者处于同一网段。Max 支持多种品牌的网络相机，不同品牌的相机，IP 设置方法有所不同。

13.2.1 通用配置

包括 XYZ-ST-M、XYZ-ST-L、XYZ-AL-M、XYZ-GW-S 及标准 2D 相机在内，大部分相机的 IP 配置方法参照以下步骤。

1. 打开 MVS 客户端，在左侧设备栏右键单击要设置的相机，选择 **修改 IP**。

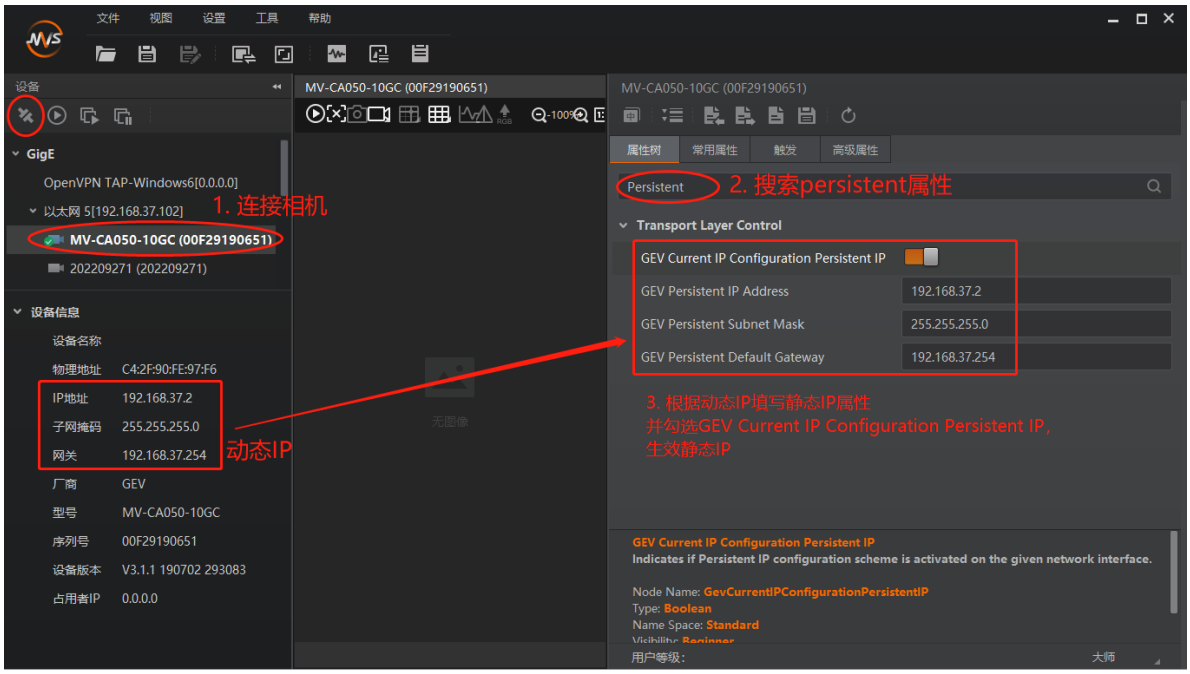
2. 选择 **静态 IP**，设置 IP 地址、子网掩码、默认网关，单击 确定。

3. 回到左侧设备栏，单击相机，查看 IP 地址是否修改成功。

4. 固定 IP 配置，避免相机 IP 在断电后重置。

   a. 在左侧设备栏双击相机或单击  ，在右侧 属性树页签下搜索栏输入 persistent。

   b. 根据动态 IP 设置静态 IP，并启用 *GEV Current IP Configuration Persistent IP*，使静态 IP 生效。



提示： 部分相机在配置 IP 后会自动重启使配置生效，此时 MVS 会报错，然后约半分钟找不到相机。待 MVS 再次找到相机后，确保 IP 正确即可。

13.2.2 XYZ-SP-L

1. 记录要设置的相机 MAC 地址（机身侧面）。

2. 打开终端，进入 xyz-studio-max/bin 目录，输入：

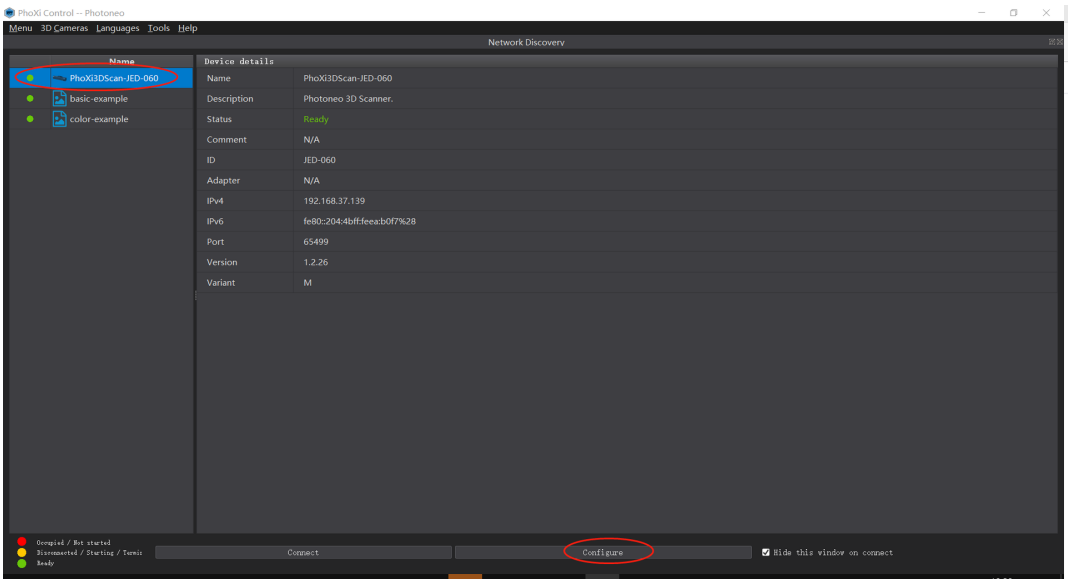
```
.\ForceDeviceIP.exe -static MAC地址 IP地址 子网掩码 网关
```

3. 打开 MVS，在左侧设备栏单击相机查看设备信息，检查动态 IP 设置是否成功。
4. 固定 IP 配置。用 MVS 连接相机并检查静态 IP 是否与动态 IP 一致，若不一致，在 MVS 中直接修改即可。操作参考通用配置 。

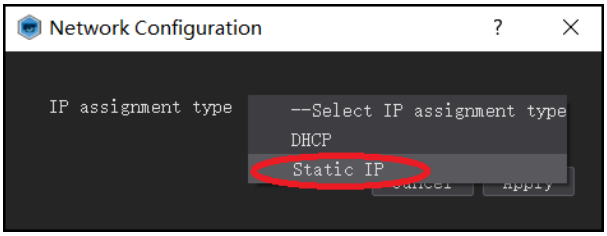
13.2.3 XYZ-GX 系列

该系列相机优先采用 IPv6 的 link local address，因此若网卡属性的 Internet 协议版本 6(TCP/IPv6) 设置为自动获取 IP，就能正常发现和使用相机。但考虑到可能存在工控机网卡不支持 IPv6 的情况，建议先设置工控机 IP 地址，参见设置工控机 *IP* 地址，再使用 PhoXi Control 设置相机 IP，操作如下：

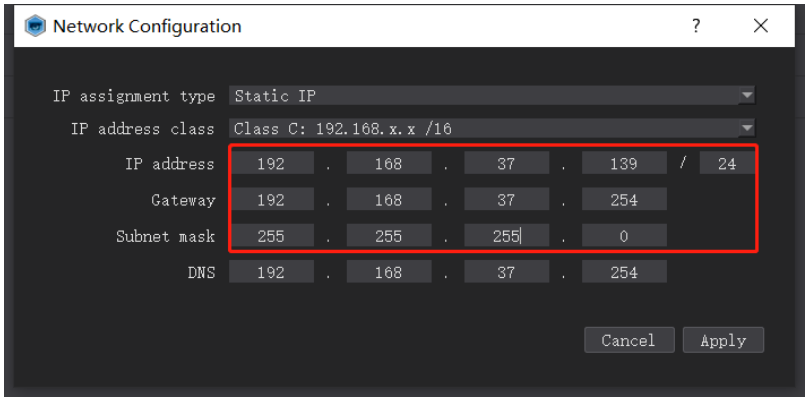
1. 打开 PhoXi Control 软件，自动进入 *Network Discovery* 界面。从左侧相机列表中选择要配置 IP 的相机，单击下方 *Configure*。



2. 从 **IP assignment type** 右侧下拉列表中选择 **Static IP**。



3. 从 **IP address class** 右侧下拉列表中选择 **192.168.x.x**，在 **IP address** 右侧设置 IP 地址，格式如下：192（预设）.168（预设）.37（手动输入，固定）.131（手动输入，介于 131-150 间，与其余视觉设备 IP 不重复）/ 16（预设），设置完毕后单击右下方 *Apply*。

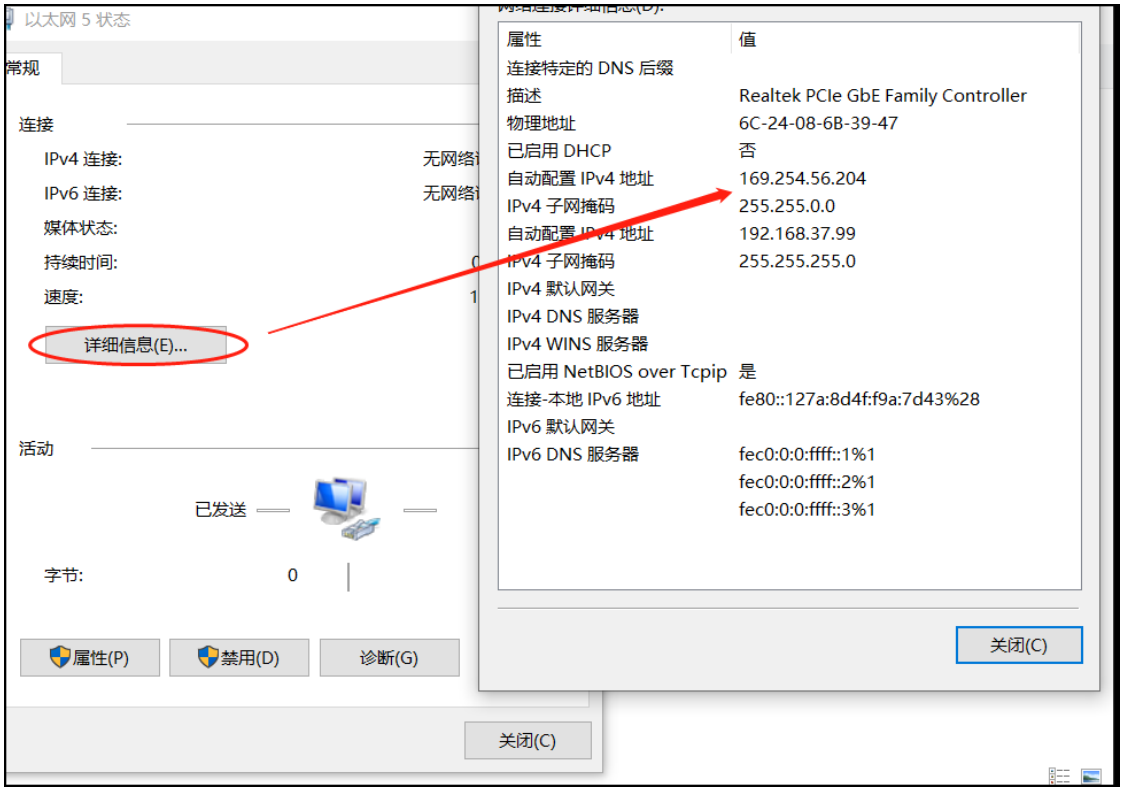


4. 界面弹出软件重启提示，单击 *Yes*。

13.3 FAQ

Q: 设置工控机 IP 后，IP 未生效，以太网出现了 169.254.xx.xx 的 IP 地址，导致相机无法连接。

A: 正常配置静态 IP，仅有 192.168.37.xx 网段，出现 169.254.xx.xx，可能是因为 IP 冲突，修改 IP 后解决。



CHAPTER 14

安装机械臂驱动

驱动程序包含机械臂主控（*Robot Master*） 和工控机主控（*IPC Master*） 两种。

14.1 驱动文件位置

位于 MAX 安装目录的 `share/robot_code` 路径下

14.2 已经适配驱动的机械臂

14.2.1 Abb

此处介绍安装 abb 机械臂驱动的相关事项。

驱动版本和使用要求

支持的机械臂类型

Abb 六轴机械臂、四轴机械臂（拆码垛机械臂）

控制器型号

IRC5

机械臂需要开通的功能

Abb 机械臂本身需要具有的功能：

- 616-1 PC Interface
- 623-1 Multitasking
- 613-1 Collision Detection
- 709-1 DeviceNet Master/Slave
- 888-3 PROFINET Device

其中 `616-1 PC Interface` 和 `623-1 Multitasking` 为程序所必须的选项。如果缺少这两个选项程序将不能正常运行。

`613-1 Collision Detection` 提供碰撞检测功能，主要作用是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损坏。可以根据需求选择是否添加该选项。

`709-1 DeviceNet Master/Slave` 提供 DeviceNet 通讯。Abb 机器人控制器和 DSQC652 IO ，板进行通讯的时需要该功能。如果是 Local_IO 板卡则不需要此选项。

`888-3 PROFINET Device` 提供 PROFINET 通讯。可以根据需求选择是否添加该选项。

可在机械臂示教器主界面 `系统信息` -> `系统属性` -> `控制模块` -> `选项`查看控制器是否具有相应功能。如果没有，请自行刷入。

图 1: Abb 控制器功能查看

安装驱动

abb 机械臂驱动文件列表

abb 机械臂的标准驱动文件如下所示

–**BACKINFO** –backinfo.txt

–**RAPID**

–**TASK1（运动相关程序）**

–**PROGMOD**

- xyz_libconst.mod （函数用到的常数变量）
- xyz_libmotpkt.mod （数据包解析处理函数）
- xyz_libsock.mod （socket 相关函数）
- xyz_main.mod （程序运行起点）
- xyz_master.mod （机械臂主控函数主体）
- xyz_motion.mod （工控机主控函数主体）

–**SYSMOD**

- user.sys

–**TASK2（机械臂发送状态程序）**

–**PROGMOD**

- xyz_libsock.mod （socket 相关函数）
- xyz_libstapkt.mod （数据包解析处理函数）
- xyz_status.mod （机械臂发送状态函数主体）
- xyz_status_task.mod （程序运行起点）

–**SYSMOD**

- user.sys

–**SYSPAR**

- EIO.cfg
- SIO.cfg
- SYS.cfg

除了上述文件外，在 SYSPAR 文件夹下还有其他文件，这些文件不是 ABB 系统所必须，也不必导入到控制器里。这些文件的作用是为了方便 IO 信号的设置，具体使用方法会在“导入程序”环节讲到。

–**SYSPAR**

- eio_d652.cfg (DSQC 652 IO 板卡的 IO 信号配置模板)
- eio_local_io.cfg (Local_IO 的 IO 信号配置模板)

设定机械臂 IP

MENU-> 控制面板-> 配置，切换主题为 Communication，进入 IP Setting。若有配置，修改如下；如果没有任何配置，则点击 添加，修改如下：

- "IP" 设置为：192.168.37.100
- "Subnet" 设置为：255.255.255.0

图 2: ABB 设定机械臂 IP

网络设置完成后重启机械臂生效。

导入程序

1. 导入程序之前需要先备份 ABB 机器人控制器里的程序。可以通过 RobotStudio 备份或者通过 U 盘来备份

- 使用 RobotStudio 来备份机器人程序

  [1] 准备好一台安装了 RobotStudio 的电脑。用网线连接装好 RobotStudio 的电脑和 ABB 控制柜上的 **X2 服务（Service）口**。电脑使用自动取得 IP（DHCP），不要手动设置，X2 口会自动分配 192.168.125.* 的 IP 给电脑。

  [2] 启动 RobotStudio，依次点击 文件-> 在线-> 一键连接。

  [3] 点击 控制器 (C)-> 备份-> 创建备份，存到指定位置。

- 使用 U 盘备份机器人程序

  [1] 准备一个文件系统为 FAT32 格式的 U 盘 (其他格式的文件系统可能不会被 ABB 控制器识别)

  [2] 将 U 盘插到示教器的 U 盘口上

图 3: RobotStudio 连接 ABB 机械臂

图 4: ABB RobotStudio 备份程序

[3] 在示教器上依次点击 MENU -> 备份与恢复 -> 备份当前系统选择备份路径到 U 盘所在的位置后，点击备份

2. 使用文件比对工具，比较备份的文件夹和要导入的 ABB 代码文件夹，并对相应部分进行修改和替换。

   文件对比工具推荐使用 Meld (下载地址：https://meldmerge.org/) 或者 Beyond Compare (下载地址：https://www.beyondcomparepro.com/download)

   为了防止修改备份的过程中误修改其他文件破坏备份，可以将备份文件重新拷贝一份。

   修改备份文件的时候请不要直接拷贝该文档中的配置内容片段，生成文档时这些片段的格式会被改动，直接拷贝文档中的代码到备份文件，会出现备份文件由于格式问题无法被 ABB 控制器识别的问题

   [1] 删除备份文件夹下的 RAPID 目录，并拷贝 ABB 标准文件夹中 RAPID 文件目录到备份文件夹

   [2] 修改 backupinfo.txt 中的 TASK 内容为：

```
>>TASK1: (T_ROB1,,)
PROGMOD\xyz_motion.mod @
SYSMOD\user.sys @
PROGMOD\xyz_libsock.mod @
PROGMOD\xyz_main.mod @
PROGMOD\xyz_libmotpkt.mod @
PROGMOD\xyz_master.mod @
PROGMOD\xyz_libconst.mod @

>>TASK2: (STATUS,,)
SYSMOD\user.sys @
PROGMOD\xyz_libsock.mod @
PROGMOD\xyz_status_task.mod @
```

(下页继续)

图 5: abb 示教器备份程序

(续上页)

```
PROGMOD\xyz_libstapkt.mod @
PROGMOD\xyz_status.mod @
```

请不要直接拷贝上述配置内容片段来修改备份文件中的 backupinfo.txt 。请根据 ABB 标准驱动文件中的 backupinfo.txt 来修改。

[3] 修改 SYS.cfg 中的 CAB_TASKS 为:

```
CAB_TASKS:

-Name "T_ROB1" -Type "NORMAL" -MotionTask

-Name "STATUS" -Task_in_forground "STATUS" -Type "NORMAL"
```

请不要直接拷贝上述配置内容片段来修改备份文件中的 SYS.cfg 。请根据 ABB 标准驱动文件中的 SYS.cfg 来修改。

[4] 修改 EIO.cfg 文件。需要在 EIO.cfg 文件中的 EIO_SIGNAL: 条目下添加如下的 IO 信号定义

```
EIO_SIGNAL:

  -Name "output0" -SignalType "DO"

  -Name "output1" -SignalType "DO"

  -Name "output2" -SignalType "DO"

  -Name "output3" -SignalType "DO"

  -Name "output4" -SignalType "DO"

  -Name "output5" -SignalType "DO"
```

(下页继续)

(续上页)

```
-Name "output6" -SignalType "DO"

-Name "output7" -SignalType "DO"

-Name "output8" -SignalType "DO"

-Name "output9" -SignalType "DO"

-Name "output10" -SignalType "DO"

-Name "output11" -SignalType "DO"

-Name "output12" -SignalType "DO"

-Name "output13" -SignalType "DO"

-Name "output14" -SignalType "DO"

-Name "output15" -SignalType "DO"

-Name "input0" -SignalType "DI"

-Name "input1" -SignalType "DI"

-Name "input2" -SignalType "DI"

-Name "input3" -SignalType "DI"

-Name "input4" -SignalType "DI"

-Name "input5" -SignalType "DI"

-Name "input6" -SignalType "DI"

-Name "input7" -SignalType "DI"

-Name "input8" -SignalType "DI"

-Name "input9" -SignalType "DI"

-Name "input10" -SignalType "DI"

-Name "input11" -SignalType "DI"

-Name "input12" -SignalType "DI"

-Name "input13" -SignalType "DI"

-Name "input14" -SignalType "DI"

-Name "input15" -SignalType "DI"

-Name "GroupDO" -SignalType "GO"
```

以上只是定义了程序所使用的 IO 信号，并没有把这些 IO 信号和物理设备上的信号做映射。如果需要把 IO 信号映射到如理设备，可以在每个 IO 信号的定义后面加上“-Device”字段并填入 IO 设备的名称，加上“-DeviceMap”字段并填入映射到 IO 设备上的端口号。下面片段展示的是 IO

信号和 DSQC 652 IO 板卡做好映射后样子。

```
EIO_SIGNAL:

  -Name "output0" -SignalType "DO" -Device "d652" -DeviceMap "0"

  -Name "output1" -SignalType "DO" -Device "d652" -DeviceMap "1"

  -Name "output2" -SignalType "DO" -Device "d652" -DeviceMap "2"

  -Name "output3" -SignalType "DO" -Device "d652" -DeviceMap "3"

  -Name "output4" -SignalType "DO" -Device "d652" -DeviceMap "4"

  -Name "output5" -SignalType "DO" -Device "d652" -DeviceMap "5"

  -Name "output6" -SignalType "DO" -Device "d652" -DeviceMap "6"

  -Name "output7" -SignalType "DO" -Device "d652" -DeviceMap "7"

  -Name "output8" -SignalType "DO" -Device "d652" -DeviceMap "8"

  -Name "output9" -SignalType "DO" -Device "d652" -DeviceMap "9"

  -Name "output10" -SignalType "DO" -Device "d652" -DeviceMap "10"

  -Name "output11" -SignalType "DO" -Device "d652" -DeviceMap "11"

  -Name "output12" -SignalType "DO" -Device "d652" -DeviceMap "12"

  -Name "output13" -SignalType "DO" -Device "d652" -DeviceMap "13"

  -Name "output14" -SignalType "DO" -Device "d652" -DeviceMap "14"

  -Name "output15" -SignalType "DO" -Device "d652" -DeviceMap "15"

  -Name "input0" -SignalType "DI" -Device "d652" -DeviceMap "0"

  -Name "input1" -SignalType "DI" -Device "d652" -DeviceMap "1"

  -Name "input2" -SignalType "DI" -Device "d652" -DeviceMap "2"

  -Name "input3" -SignalType "DI" -Device "d652" -DeviceMap "3"

  -Name "input4" -SignalType "DI" -Device "d652" -DeviceMap "4"

  -Name "input5" -SignalType "DI" -Device "d652" -DeviceMap "5"

  -Name "input6" -SignalType "DI" -Device "d652" -DeviceMap "6"

  -Name "input7" -SignalType "DI" -Device "d652" -DeviceMap "7"

  -Name "input8" -SignalType "DI" -Device "d652" -DeviceMap "8"

  -Name "input9" -SignalType "DI" -Device "d652" -DeviceMap "9"

  -Name "input10" -SignalType "DI" -Device "d652" -DeviceMap "10"
```

(下页继续)

(续上页)

```
  -Name "input11" -SignalType "DI" -Device "d652" -DeviceMap "11"

  -Name "input12" -SignalType "DI" -Device "d652" -DeviceMap "12"

  -Name "input13" -SignalType "DI" -Device "d652" -DeviceMap "13"

  -Name "input14" -SignalType "DI" -Device "d652" -DeviceMap "14"

  -Name "input15" -SignalType "DI" -Device "d652" -DeviceMap "15"

  -Name "GroupDO" -SignalType "GO" -Device "d652" -DeviceMap "0-15"
```

如果需要配置的 ABB 机器人使用的是 DSQC 652 IO 板卡，请不要直接拷贝上述配置内容片段来修改备份文件中的 EIO.cfg 。而是根据 ABB 标准驱动文件中的 eio_d652.cfg 来修改。如果需要配置的 ABB 机器人使用的是 Local_IO 板卡，请根据 ABB 标准驱动文件中的 eio_local_io.cfg 来修改 EIO.cfg 文件。修改时请一定注意格式，每个 IO 信号的定义之间会有空行，EIO.cfg 文件的最后一行也需要有空行。

关于 IO 设备的名称可以在 EIO.cfg 文件和设备相关的条目下查看。以下代码展示 ABB 机器人上常用的两种 IO 设备，一种是 DEVICENET_DEVICE: 条目下定义通过 DeviceNet 通信的 DSQC 652 IO 板卡，一种是 ETHERNETIP_DEVICE 条目下定义的 ABB Local I/O Device。在设备信息中"-Name"后的名字是 IO 信号需要映射到的"-Device"的设备名。

```
DEVICENET_DEVICE:

  -Name "d652" -VendorName "ABB Robotics" -ProductName "24 VDC I/O Device"\
  -Label "DSQC 652 24 VDC I/O Device" -Address 10 -ProductCode 26\
  -DeviceType 7 -ConnectionType "COS" -OutputSize 2 -InputSize 2

ETHERNETIP_DEVICE:

  -Name "Local_IO" -VendorName "ABB Robotics" -ProductName "DSQC1030"\
  -Label "ABB Local I/O Device" -Address "192.168.125.100" -VendorId 75\
  -ProductCode 29 -DeviceType 12 -OutputAssembly 100 -InputAssembly 101\
  -ConfigurationAssembly 102 -InputConnectionType "POINT2POINT"\
  -ConnectionPriority "SCHEDULE" -OutputSize 2 -InputSize 2\
  -ConfigurationSize 8\
  -ConfigurationData00 "01 02 09 10 01 08 10 01 00 00 00 00 00 00 00 00"\
  -O2T_RPI 40000 -T2O_RPI 40000
```

3. 替换修改完成后，需要把修改后的备份恢复到控制器中。

- 使用 RobotStudio 恢复备份

  [1] 按照"导入程序"中"使用 RobotStudio 来备份机器人程序"的前两步将 RobotStudio 和机器人连接

  [2] 连接后点击 控制器 (C)-> 备份-> 从备份中恢复，选择修改后的备份文件

  [3] 点击确定后，机器人将自动重启进行备份恢复。

- 使用 U 盘恢复备份

  [1] 将修改的备份程序拷贝到 U 盘中

  [2] 将 U 盘插到示教器的 U 盘口上

  [3] 在示教器上依次点击 MENU -> 备份与恢复 -> 恢复系统选择 U 盘中修改后的备份文件路径，点击 确定

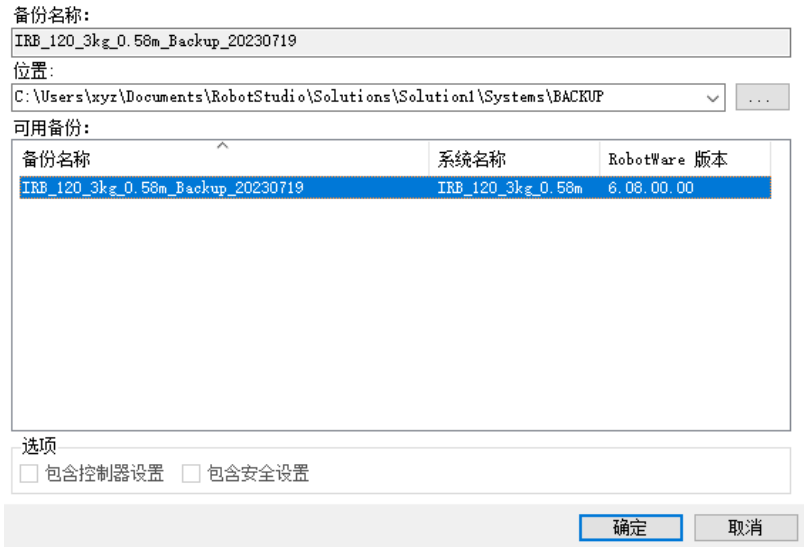


图 6: RobotStudio 恢复备份

图 7: U 盘恢复备份

[4] 点击 恢复机器人将自动重启进行备份恢复.

配置 IO 信号

如果在导入程序的步骤中对 EIO.cfg 文件中的 IO 信号和 IO 板卡做了映射，那么就不需要在配置 IO 信号。另外如果只是 **机械臂主控**，进行 **标定**操作，也可以不把 IO 信号映射到物理设备上。本步骤适合 EIO.cfg 文件只定义了 IO 信号，没有把 IO 信号映射到 IO 板卡上，或者修改已经映射好的 IO 信号。

- 使用 RobotStudio 配置 IO 信号（**建议**）

  [1] 用网线连接装好 RobotStudio 的电脑和控制柜上的 **X2 服务（Service）口**。电脑使用自动取得 IP（DHCP），不要手动设置，X2 口会自动分配 192.168.125.* 的 IP 给电脑。

  [2] 启动 RobotStudio，依次点击 文件-> 在线-> 一键连接。

  [3] 点击 控制器 （C）中的 请求写权限，然后在示教器上点击 同意。

图 8: RobotStudio 请求写权限

[4] 选择 配置中的 I/O System, 然后单击类型栏里的 Signal，开始配置信号

[5] 开始 IO 信号和硬件之间的映射。

- IO 信号和数字量输出端口的映射。下面以 IO 信号 “output0” 和 DSQC 652 IO 板卡为例做说明:

  双击 “Name” 一栏中的 “output0”，弹出 “output0” 的配置界面。

  在 “Assign to Device” 中选择此 IO 信号使用的 IO 板卡。

  在 “Device Mapping” 中输入具体映射到 IO 板卡上的端口。

  单击配置界面底部的 “确定” 完成配置。

  完成后即可将 “output0” 映射到 DSQC 652 数字量输出的 0 号端口上。按照此方法完成 “output0” 到 “output15” 这十六个数字量输出 IO 信号的配置。

- IO 信号和数字量输入端口的映射，下面以 IO 信号 input0 和 DSQC 652 IO 板卡为例做说明:

  双击 “Name” 一栏中的 “input0”，弹出 “input0” 的配置界面。

  在 “Assign to Device” 中选择此 IO 信号使用的 IO 板卡。

  在 “Device Mapping” 中输入具体映射到 IO 板卡上的端口。

  单击配置界面底部的 “确定” 完成配置。

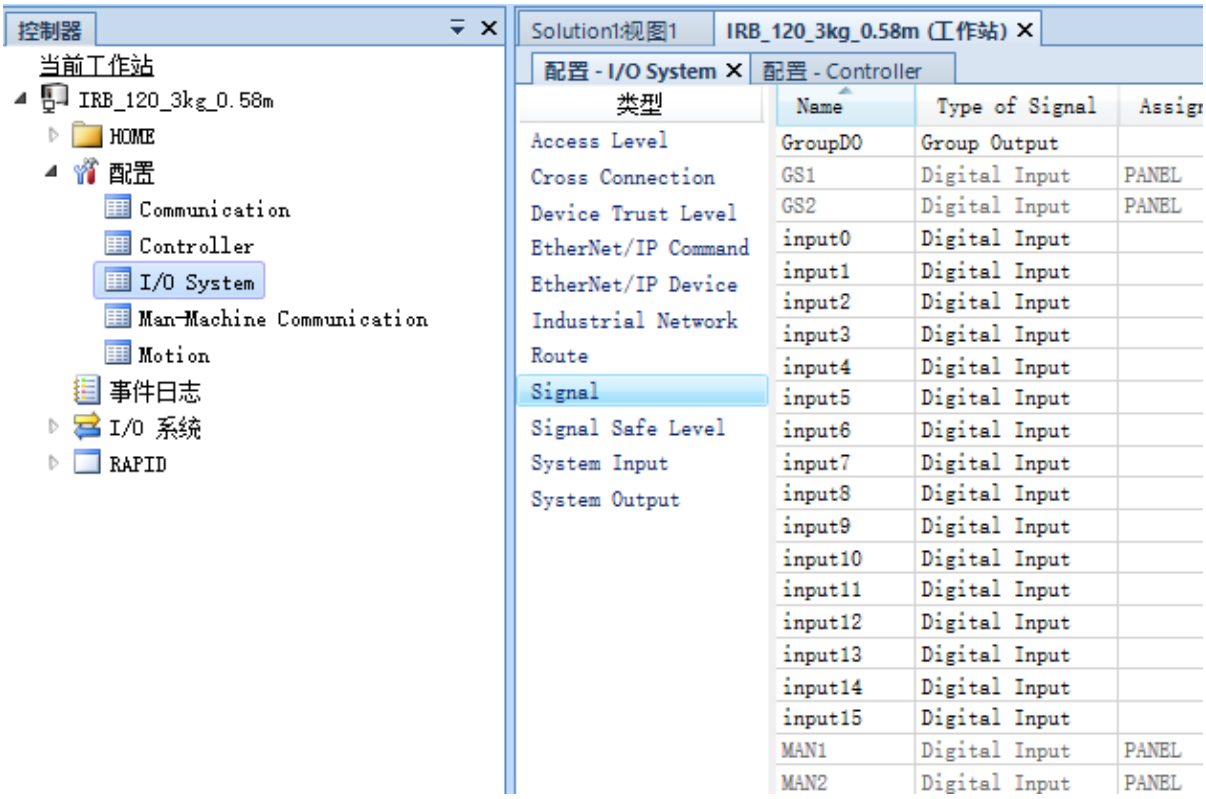


图 9: RobotStudio 配置 I/O System

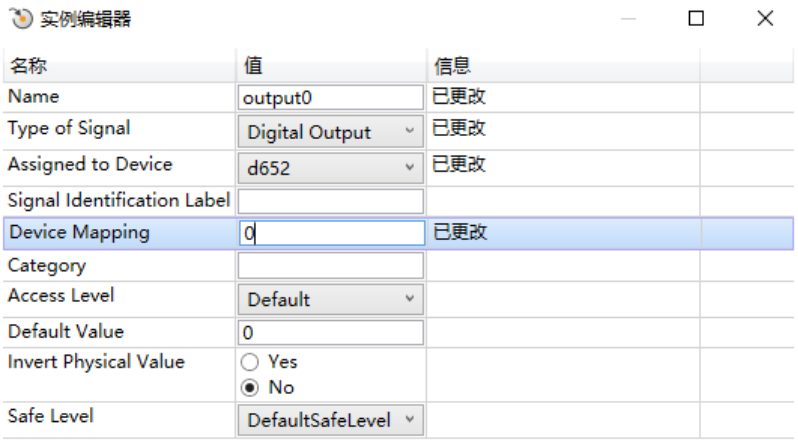


图 10: RobotStudio 添加“output0”端口

完成后即可将“input0”映射到 DSQC 652 数字量输入的 0 号端口上。按照此方法完成“input0”到“input15”这十六个数字量输入 IO 信号的配置。

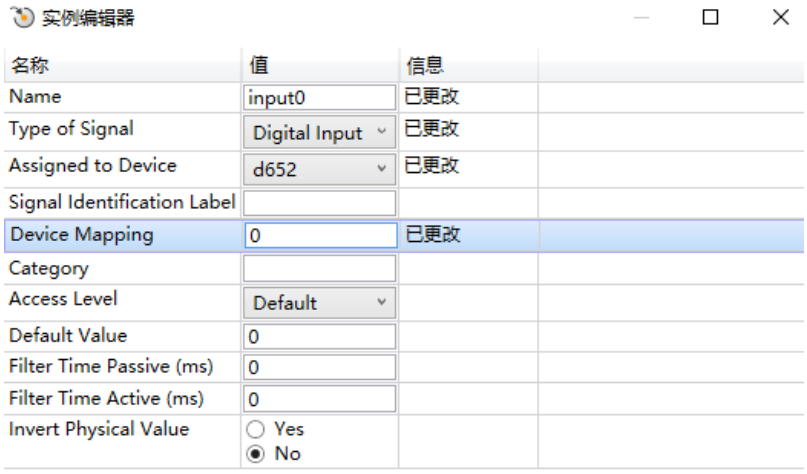


图 11: RobotStudio 添加“input0”端口

– GroupDO 信号配置

双击“Name”一栏中的“GroupDO”，弹出“GroupDO”的配置界面。

在“Assign to Device”中选择此 GroupDO 信号使用的 IO 板卡。

在“Device Mapping”中输入具体映射到 IO 板卡上端口的范围。

单击配置界面底部的“确定”完成配置。

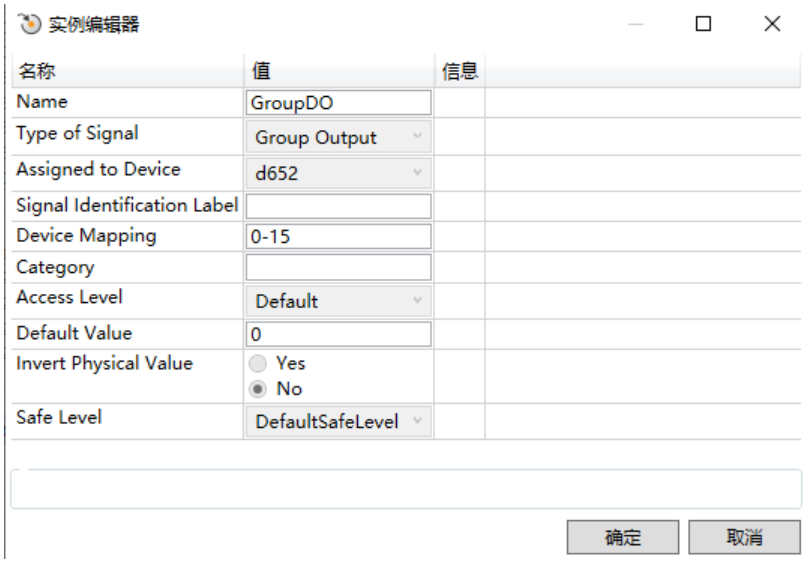


图 12: RobotStudio 添加“GroupDO”端口

可以完成所有信号的配置后再重启控制器，这样可以节省时间。另外信号配置完成后必须重启控制器才能生效。

- 使用示教器设定

  [1] 单击示教器左上角的主菜单按钮，进入 控制面板（首先将机械臂运行模式切换至手动模式，在自动或手动全速模式下无法对信号进行配置操作）。

图 13: 示教器主菜单

[2] 选择“配置”。

[3] 双击“Signal”（或选中“Signal”，然后单击示教器右下角的“显示全部”）。

[4] 开始 IO 信号和硬件之间的映射。

- IO 信号和数字量输出端口的映射。下面以 IO 信号“output0”和 DSQC 652 IO 板卡为例做说明:

  在“Signal”中找到“output0”并选中，然后点击 编辑切换到“output0”的配置界面。

  在“Assign to Device”中选择此 IO 信号使用的 IO 板卡。

  在“Device Mapping”中输入具体映射到 IO 板卡上的端口。

  单击配置界面底部的“确定”完成配置。

- IO 信号和数字量输入端口的映射，下面以 IO 信号“input0”和 DSQC 652 IO 板卡为例做说明:

  在“Signal”中找到“input0”并选中，然后点击 编辑切换到“input0”的配置界面。

  在“Assign to Device”中选择此 IO 信号使用的 IO 板卡。

  在“Device Mapping”中输入具体映射到 IO 板卡上的端口。

  单击配置界面底部的“确定”完成配置。

- GroupDO 信号配置

  在“Signal”中找到“GroupDO”并选中，然后点击 编辑切换到“GroupDO”的配置界面。

  在“Assign to Device”中选择此 GroupDO 信号使用的 IO 板卡。

图 14: 示教器配置

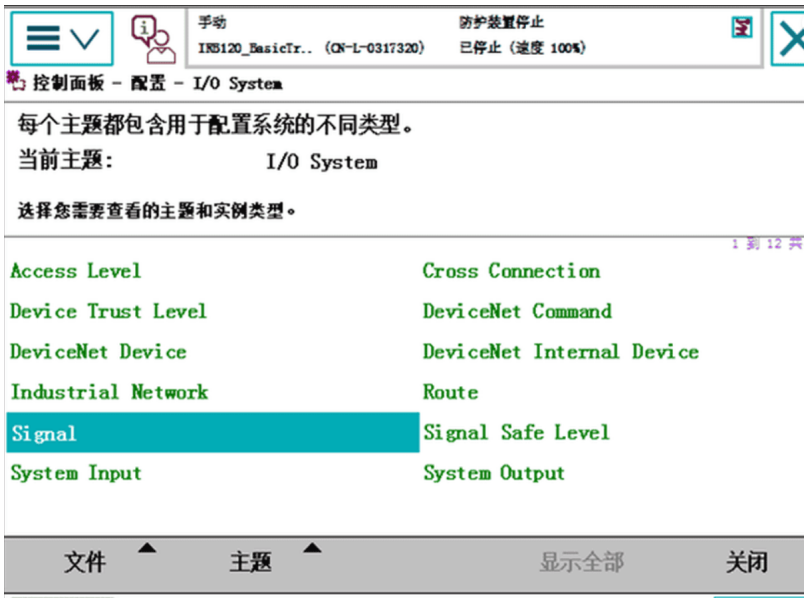


图 15: 示教器信号

图 16: 示教器添加“output0”端口

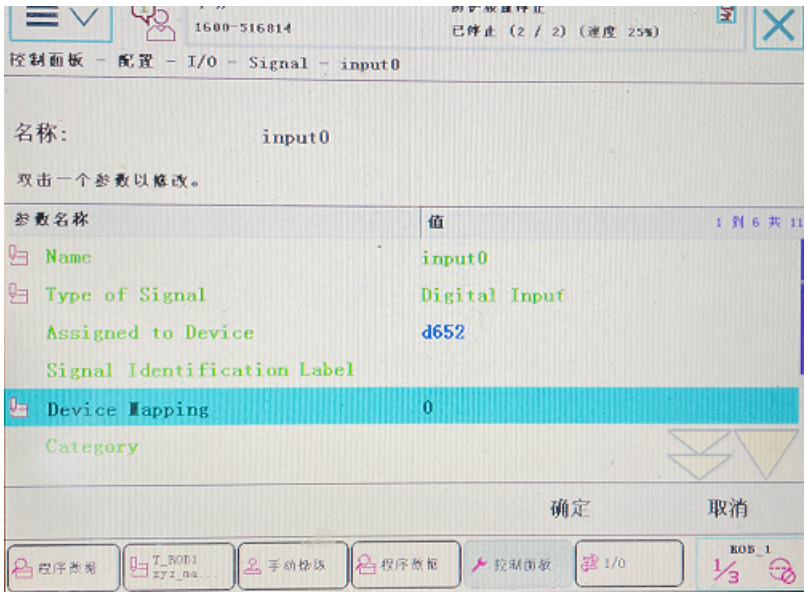


图 17: 示教器添加“input0”端口

在“Device Mapping”中输入具体映射到 IO 板卡上端口的范围。

单击配置界面底部的“确定”完成配置。

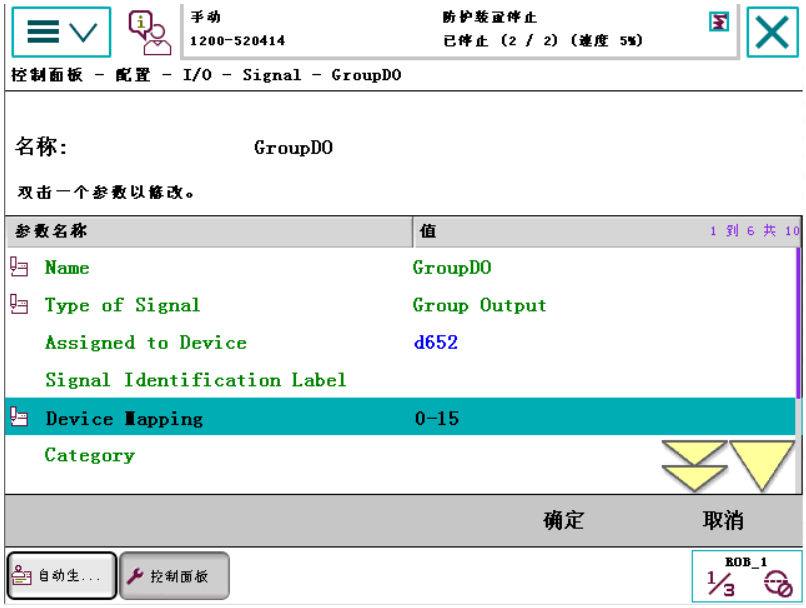


图 18: 示教器添加 GroupDO 端口

四轴机械臂需要进行的操作

四轴机械臂运行前，需要注释 RAPID 代码中机械臂 Move 命令前后的：`SingArea \Wrist`，`ConfL \On`，`ConfL \Off` 代码。

例如：

```
! ConfL \On;
! SingArea \Wrist;
MoveL target, gMotSpeed, gMotZone, gMotTool \WObj := gMotWobj;
! ConfL \Off;
```

运行程序

通讯说明

工控机和 ABB 机械臂的通讯方式为 TCP/IP： 工控机作为 TCP/IP 的服务端，机械臂作为 TCP/IP 的客户端。

启动程序

运行工控机主控

修改 T_ROB1 中的 main 函数，将 xyzMaster 注释掉，保留 xyzMotion ，表示运行工控机主控。

MENU -> 自动生产窗口, 点击 PP 移至 Main，运行程序即可。

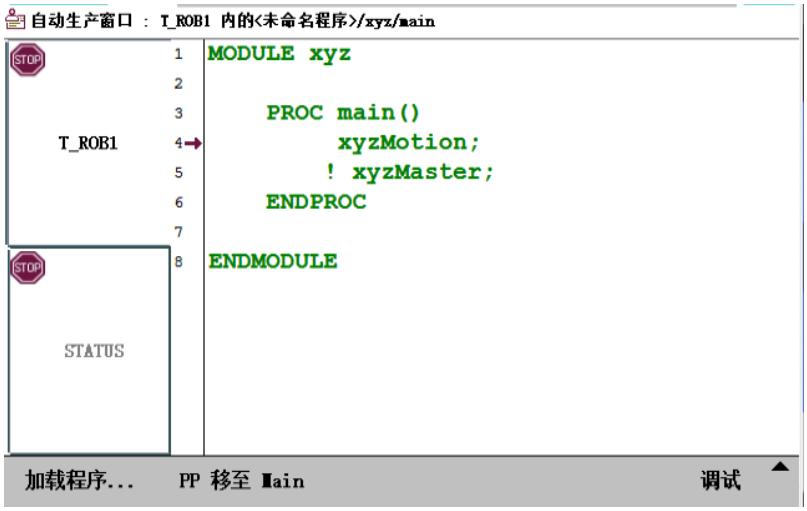


图 19: 启动程序

注：如果运行时发现 **转角路径故障**，通常是 MAX 任务流图设置的圆滑度（ZONE ）过大，可以调整到零或较小值。

运行机械臂主控

修改 T_ROB1 中的 main 函数，将 xyzMotion 注释掉，保留 xyzMaster ，表示运行机械臂主控。

MENU -> 自动生产窗口, 点击 PP 移至 Main ，运行程序即可。

xyzMaster 需要根据项目具体流程和需求进行修改。修改时可以根据项目需求和流程参考 xyz_master.mod 中提供的机械臂主控的四个模板函数。或者把 xyzMaster 注释掉，然后把需要调用的模板函数添加到 main 函数中。

请注意模板函数并不能直接运行，请一定根据项目现场环境和流程需求进行修改

模板函数分别是“CartMoveBasic”，“CartMoveRepo”，“TrajMoveSync”，“TrajMoveAsync”。在这四个模板中:

“CartMoveBasic”是坐标运动的基础模板，包含了眼在手上和眼在手外的基础运行逻辑。

“CartMoveRepo”提供了坐标移动带工件二次定位的运行逻辑。

"TrajMoveSync" 是轨迹移动的同步方式模板，即每次请求获取轨迹后，都会立即执行获取的轨迹。

"TrajMoveAsync" 是轨迹移动的异步方式模板，会在抓取到工件执行完出抓取区域轨迹后开始下一次请求规划指令，然后再继续请求未执行完的放置轨迹。

API 说明

abb 工控机主控支持的 API

指令号	命令	支持情况
100	获取机械臂程序版本号	支持
101	发送速度数据	支持
102	发送加速度数据	支持
103	发送圆滑过渡参数	支持
104	发送工具坐标系 (TCP) 参数	支持
105	设置数字量输出	支持
106	SetJointsMovej	支持
107	SetCartMovel	支持
108	SetJoinsMovel	支持
109	SetCartMovej	支持
110	MovejSequence	支持
111	MovelSequence	支持
112	SetJointsMovejDo	支持
113	SetCartMovelDo	支持
114	SetJointsMovelDo	支持
115	SetJointsMovejGroupDo	支持
116	SetCartMovelGroupDo	支持
117	SetJointsMovelGroupDo	支持
118	MovelUntil	支持
119	获取数字量输入状态	支持
120	获取数字量输出状态	支持
121	获取模拟量输入口数值	不支持
122	获取机械臂当前角度	支持
123	获取机械臂当前位姿	支持
200	机械臂后台发送状态	支持

abb 机械臂主控支持的 API

PROC xyzCreateSocket(VAR socketdev sock)

创建套接字

参数 sock (*VAR socketdev*) –套接字

PROC xyzConnect(VAR socketdev sock, string ip, num port)

连接到服务器

参数

- **sock** (*num*) –套接字
- **ip** (*string*) –服务器 ip 地址

- **port** –服务器端口号

PROC xyzClose(VAR socketdev sock)

关闭套接字

参数 sock(*VAR socketdev*) –套接字

FUNC num xyzSwitchApp(string app_name)

切换应用

参数 app_name(*string*) –应用名称

返回 err_code

返回类型 num

FUNC num xyzSwitchFlow(string flow_name)

切换 flow

参数 flow_name(*string*) –flow 名称

返回 err_code

返回类型 num

FUNC num xyzSwitchItem(int ws_id, string item_codename)

切换工件

参数

- **ws_id**(*num*) –工作空间 id
- **item_codename**(*string*) –工件名称

返回 err_code

返回类型 num

FUNC num xyzSwitchTool(string tool_name)

切换工具

参数 tool_name(*string*) –工具名称

返回 err_code

返回类型 num

FUNC num xyzReqCapImg(num ws_id, VAR num token)

请求拍照

参数

- **ws_id**(*num*) –工作空间 id
- **token**(*num*) –请求拍照结果

返回 err_code

返回类型 num

FUNC num xyzGetCapImg(num token)

获取拍照结果

参数 token(*num*) –请求拍照时返回的 token

返回 err_code

返回类型 num

FUNC num xyzCapImg(num ws_id)

拍照

参数 **ws_id** (*num*) –需要进行拍照操作的工作空间 id

返回 err_code

返回类型 num

FUNC num xyzReqGraspPose(num ws_id, VAR num token)

请求抓取位姿

参数

- **ws_id** (*INnumT*) –需要获取抓取点位的工作空间 id
- **token** (*num*) –返回的用于获取目标点位时使用的 token

返回 err_code

返回类型 num

FUNC num xyzGetGraspPose(num token, VAR num pipeline_num, VAR num register_num, VAR num pose_type, VAR pose grasp_pose)

获取抓取位姿

参数

- **token** (*num*) –求抓取目标点位时返回的 token
- **pose_num** (*num*) –可供抓取的点数量
- **pipeline_num** (*num*) –pipeline 编号
- **register_num** (*num*) –注册编号
- **grasp_pose** (*pose*) –抓取位姿

返回 err_code

返回类型 num

FUNC num xyzReqObjPose(num ws_id, VAR num token)

请求物体位姿

参数

- **ws_id** (*num*) –需要获取物体位姿的工作空间 id
- **obj_token** (*num*) –物体位姿识别的 token

返回 err_code

返回类型 num

FUNC num xyzGetObjPose(num token, VAR num pose_num, VAR num pose_type, VAR pose obj_pose)

获取物体位姿

参数

- **token** (*num*) –请求物体位姿时得到的 token
- **pose_num** (*num*) –物体数量
- **pose_type** (*num*) –当前返回的物体 pose 类型

- **obj_pose** (*pose*) –物体位姿

返回 err_code

返回类型 num

FUNC num xyzResetVision(num ws_id)

重置视觉

参数 ws_id (*num*) –需要重置视觉的工作空间 id

返回 err_code

返回类型 num

FUNC num xyzSendCurrentJoints(num joints{*})

发送特定角度数组

参数 joint{*} (*num*) –传入角度数组

返回 err_code

返回类型 num

FUNC num xyzSendCurrentCartPose(num cart_pose{*})

发送特定位姿

参数 cart_pose{*} (*num*) –传入位姿数组

返回 err_code

返回类型 num

FUNC num xyzSendCurrentExtJoints(num ext_joints{*})

发送特定角度数组

参数 ext_joints{*} (*num*) –传入角度数组

返回 err_code

返回类型 num

FUNC num xyzReqPick()

请求 pick 动作规划

返回 err_code

返回类型 num

FUNC num xyzReqPlace()

请求 place 动作规划

返回 err_code

返回类型 num

FUNC num xyzReqPickPlace()

请求 pick 和 place 规划

返回 err_code

返回类型 num

FUNC num xyzGetPickin(VAR num pose_num, VAR num pose_type, VAR num wp_type{*}, VAR jointtarget joint_wp{*}, VAR robtarget cart_wp{*})

获取取料入框轨迹

参数

- **pose_num** (*num*) –轨迹点数
- **pipeline_num** (*num*) –pipeline 编号
- **register_num** (*num*) –注册编号
- **wp_type{*}** (*num*) –轨迹点类型数组
- **joint_wp{*}** (*jointtarget*) –轨迹点角度数组
- **cart_wp{*}** (*robtarget*) –轨迹点位姿数组

返回 err_code

返回类型 num

FUNC num xyzGetPickout(VAR num pose_num, VAR num pose_type, VAR num wp_type{*}, VAR jointtarget joint_wp{*}, VAR robtarget cart_wp{*})

获取取料出框轨迹

参数

- **pose_num** (*num*) –轨迹点数
- **pipeline_num** (*num*) –pipeline 编号
- **register_num** (*num*) –注册编号
- **wp_type{*}** (*num*) –轨迹点类型数组
- **joint_wp{*}** (*jointtarget*) –轨迹点角度数组
- **cart_wp{*}** (*robtarget*) –轨迹点位姿数组

返回 err_code

返回类型 num

FUNC num xyzGetPlacein(VAR num pose_num, VAR num pose_type, VAR num wp_type{*}, VAR jointtarget joint_wp{*}, VAR robtarget cart_wp{*})

获取放料入框轨迹

参数

- **pose_num** (*num*) –轨迹点数
- **pipeline_num** (*num*) –pipelin 编号
- **register_num** (*num*) –注册编号
- **wp_type{*}** (*num*) –轨迹点类型数组
- **joint_wp{*}** (*jointtarget*) –轨迹点角度数组
- **cart_wp{*}** (*robtarget*) –轨迹点位姿数组

返回 err_code

返回类型 num

FUNC num xyzGetPlaceout(VAR num pose_num, VAR num pose_type, VAR num wp_type{*}, VAR jointtarget joint_wp{*}, VAR robtarget cart_wp{*})

获取放料出框轨迹

参数

- **pose_num** (*num*) –轨迹点数
- **pipeline_num** (*num*) –pipelin 编号
- **register_num** (*num*) –注册编号
- **wp_type{*}** (*num*) –轨迹点类型数组
- **joint_wp{*}** (*jointtarget*) –轨迹点角度数组
- **cart_wp{*}** (*robtarget*) –轨迹点位姿数组

返回 err_code

返回类型 num

FUNC num xyzSwitchStrat(string strat_name)

请求切换策略

参数 strat_name (*string*) –策略名称

返回 err_code

返回类型 num

FUNC num xyzUpdateTotePose(VAR pose tote_pose)

料箱重定位

参数 tote_pose (*pose*) –料箱位姿

返回 err_code

返回类型 num

FUNC num xyzUpdateObjPoseOnHand()

工件在上手的二次定位

返回 err_code

返回类型 num

FUNC num xyzUpdateObjPoseToHand(VAR num pose_num, VAR num pose_type, VAR num wp_type{*}, VAR jointtarget joint_wp{*}, VAR robtarget cart_wp{*})

工件不在手上的二次定位

参数

- **pose_num** (*num*) –工件点位的数量
- **pipeline_num** (*num*) –pipeline 编号
- **register_num** (*num*) –注册编号
- **wp_type{*}** (*num*) –工件点位类型数组
- **joint_wp{*}** (*jointtarget*) –工件点位角度数组
- **cart_wp{*}** (*robtarget*) –工件位姿数组

返回 err_code

返回类型 num

FUNC num xyzGetObjPoseType(VAR num pose_type)

获取工件姿态类型

参数 pose_type (*num*) –工件姿态类型

返回 err_code

返回类型 num

FUNC num xyzResetPalletStatus()

重置工业码垛状态

返回 err_code

返回类型 num

FUNC num xyzCalculateGraspPose(num ws_id, VAR num pipeline_num, VAR num register_num, VAR num pose_type, VAR pose grasp_pose)

计算抓取位姿

参数

- **ws_id** (*num*) –工作空间 id
- **pose_num** (*num*) –可供抓取的点数量
- **pipeline_num** (*num*) –pipeline 编号
- **register_num** (*num*) –注册编号
- **grasp_pose** (*pose*) –抓取位姿

返回 err_code

返回类型 num

FUNC num xyzCalculateObjectPose(num ws_id, VAR num pose_num, VAR num pose_type, VAR pose obj_pose)

计算物体位姿

参数

- **ws_id** (*num*) –工作空间 id
- **pose_num** (*num*) –物体数量
- **pose_type** (*num*) –当前返回的物体 pose 类型
- **obj_pose** (*pose*) –物体位姿

返回 err_code

返回类型 num

FUNC num xyzUsrCmd(string out_strings{*}, num out_ints{*}, num out_floats{*}, robtarget out_cart, jointtarget out_joints, VAR string in_strings{*}, VAR num in_ints{*}, VAR num in_floats{*}, VAR robtarget in_cart, VAR jointtarget in_joints)

自定义请求

参数

- **out_strings** (*string{5}*) –发送给工控机的字符串数组，数组长度为 5
- **out_ints** (*num{10}*) –发送给工控机的整型数组，数组长度需为 10

- **out_floats** (*num{10}*) –发送给工控机的浮点数数组，数组长度为 10
- **out_cart** (*robtarget*) –发送给工控机的笛卡尔位姿
- **out_joints** (*jointtarget*) –发送给工控机的关节角度
- **in_strings** (*string{5}*) –工控机发送给机械臂的字符串数组要存放的变量，数组长度为 5
- **in_ints** (*num{10}*) –工控机发送给机械臂的整型数组要存放的变量，数组长度为 10
- **in_floats** (*num{10}*) –工控机发送给机械臂的浮点数数组要存放的变量，数组长度为 10
- **in_cart** –工控机发送给机械臂的笛卡尔位姿要存放的变量
- **in_joints** (*jointtarget*) –工控机发送给机械臂的关节角度值要存放的变量

案例/模板说明

机械臂主控模板说明

以下为机械臂主控模板代码，包含坐标移动基础模板, 坐标移动二次定位模板，轨迹移动同步模板以及轨迹移动异步模板。关于模板中的 API 可以查阅 ABB 的“API 说明”部分。

请注意模板函数并不能直接运行，请一定根据项目现场环境和流程需求进行修改。建议使用 RobotStudio 来修改。

坐标移动基础模板

```
PROC CartMoveBasic()
! S1: 初始化
VAR num err_code;            ! 错误代码
VAR bool eye_on_hand;        ! true 表示眼在手上，false 表示眼在手外
VAR num token;               ! 获取请求结果凭证
VAR jointtarget home_pose;   ! home 点
VAR robtarget scan_pose;     ! 眼在手上时，需要发送的拍照位
VAR pose grasp_pose;         ! 抓取位姿
VAR num cart_array{6};       ! 用于发送拍照点位姿
VAR num pose_num;            ! 识别出的位姿个数
VAR num pipeline_num;        ! 处理流程编号
VAR num register_num;        ! 抓取序号

Reset output0; ! 重置数字输出
WaitTime 0.5;

! S2: 连接到工控机
xyzCreateSocket master_socket;                                    ! 创建 socket client
xyzConnect master_socket, master_server_ip, master_server_port; ! 连接上位机 server

! err_code := xyzSwitchFlow("cart_basic.t");  ! 切换任务流图␣
↪, 默认被注释掉，如果需求可以取消注释
!␣
↪检查工控机返回的错误码，如果有错误程序将直接停止。可以根据项目需求自定义错误代码处理方式
! xyzCheckErrorCode(err_code);
```

(下页继续)

(续上页)

```
! S3: 切换成当前工件
! 切换工件，两个参数分别表示视觉服务ID，和工件代号
! 工件代号为MAX中映射表与通讯协议设置中的参数
err_code := xyzSwitchItem(0,"item1");
xyzCheckErrorCode(err_code);

! S4: 运动到 home 位
! 首先移动到 home 点。注意需要先定义 home_pose 的关节角度值，可以通过 RobotStudio␣
↪来修改
! 或者用使用其他点来替代
MoveAbsJ home_pose, gMotSpeed, gMotZone, gMotTool \Wobj:=gMotWobj;

! 如果是眼在机械臂上的应用形式，请把 eye_on_hand 赋值为 true
eye_on_hand := false;
WHILE TRUE DO
    LOOPSTART:
    ! S5: 眼在机械臂上
    IF eye_on_hand THEN
        ! S6: 运动到拍照位姿
        ! 注意需要先定义 scan_pose 的位姿，可以通过 RobotStudio 来修改
        ! 或者用使用其他点来替代
        MoveL scan_pose, gMotSpeed, fine, gMotTool \Wobj:=gMotWobj; ! 移动到 scan pose
        ! S7: 发送拍照位姿
        err_code := xyzSendCurrentCartPose(scan_pose);
        xyzCheckErrorCode(err_code);
    ENDIF

    ! S8: 请求抓取点位
    err_code = xyzReqGraspPose(0, token);   ! 请求 grasp pose
    xyzCheckErrorCode(err_code);

    ! S9: 获取抓取点位
    err_code = xyzGetGraspPose(token, pose_num, pipeline_num, register_num, grasp_
↪pose);
    xyzCheckErrorCode(err_code);
    IF pose_num < 1 THEN  ! 处理没有识别到工件的情况
        WaitTime 5
        GOTO LOOPSTART;
    ENDIF

    ! S10: 运动到抓取点并抓取工件
    ! 可能需要添加其他路径点作为过渡

    ! 移动到预抓取点，偏移值可以根据实际需求修改
    MoveL Offs(grasp_pose, 0, 0, 100), gMotSpeed, gMotZone, gMotTool \WObj:=gMotWobj;
    ! 移动到抓取位姿
    MoveL grasp_pose, gMotSpeed, gMotZone, gMotTool \Wobj:=gMotWobj;
    Set output0;   ! 控制数字输出
    WaitTime 0.5;

    ! S11: 运动到放置点并放置工件
    ! 移会预抓取点，偏移值可以根据实际需求修改
    MoveL Offs(grasp_pose, 0, 0, 100), gMotSpeed, gMotZone, gMotTool \WObj:=gMotWobj;

    ! 需要添加其他路径点作为过渡
```

(下页继续)

(续上页)

```
    ! 移动到放置点之后
    Reset output0;  ! 重置数字输出
ENDWHILE
ENDPROC
```

坐标移动二次定位模板

```
PROC CartMoveRepo()
! S1: 初始化
VAR num err_code;            ! 错误代码
VAR num token;               ! 获取请求结果凭证1
VAR num token1;              ! 获取请求结果凭证2
VAR jointtarget home_pose;   ! home 点
VAR robtarget scan_pose;     ! 拍照位姿
VAR robtarget grasp_pose;    ! 抓取位姿
VAR num cart_array{6};       ! 用于发送拍照点位姿
VAR num pose_num;            ! 识别出的位姿个数
VAR num pipeline_num;        ! 处理流程编号
VAR num register_num;        ! 抓取序号

! S2: 连接到工控机
xyzCreateSocket master_socket; 创建 socket
xyzConnect master_socket, master_server_ip, master_server_port; ! 连接上位机 server

! err_code := xyzSwitchFlow("cart_repo.t"); !␣
↪切换任务流图，默认被注释掉，如果需求可以取消注释
!␣
↪检查工控机返回的错误码，如果有错误程序将直接停止。可以根据项目需求自定义错误代码处理方式
! xyzCheckErrorCode(err_code);

Reset output0; ! 重置数字输出
Reset output1;
WaitTime 0.5;

! S3: 机械臂外的相机切换成识别当前工件
! 切换工件，两个参数分别表示视觉服务 ID ，和工件代号
! 工件代号为 MAX 中映射表与通讯协议设置中的参数
err_code := xyzSwitchItem(0,"item1");
xyzCheckErrorCode(err_code);

! S4: 运动到 home 位
! 首先移动到 home 点。注意需要先定义 home_pose 的关节角度值，可以通过 RobotStudio␣
↪来修改
! 或者用使用其他点来替代
MoveAbsJ home_pose, gMotSpeed, gMotZone, gMotTool \Wobj:=gMotWobj;

WHILE TRUE DO
    ! S5: 请求工件的抓取位姿
    err_code = xyzReqGraspPose(0, token);   ! 请求抓取位姿
    xyzCheckErrorCode(err_code);

    ! S6: 获取工件的抓取位姿
    err_code = xyzGetGraspPose(token, pose_num, pipeline_num, register_num, grasp_
↪pose);
```

(下页继续)

(续上页)

```
    xyzCheckErrorCode(err_code);

    ! S7: 处理是否识别到工件
    IF pose_num < 1 THEN                          ! 处理没有识别到工件的情况
        ! 工作空间内没有工件，需要先移除隔板
        ! S15: 没有识别到工件，机械臂外的相机切换识别隔板
        err_code := xyzSwitchItem(0,"board"); ! 切换隔板
        xyzCheckErrorCode(err_code);

        ! S16: 请求隔板抓取位姿
        err_code = xyzReqGraspPose(0, token);    ! 请求隔板的抓取位姿
        xyzCheckErrorCode(err_code);

        ! S17: 获取隔板的抓取位姿
        err_code = xyzGetGraspPose(token, pose_num, pipeline_num, register_num, grasp_
↪pose);
        xyzCheckErrorCode(err_code);
        IF pose_num < 1 THEN   ! 未识别到隔板的处理
            TPWrite "Get board grasp pose failed";
            Stop;
        ENDIF

        ! S18: 运动到隔板抓取位姿，并抓取隔板
        ! 注意: 可能需要添加其他路径点作为过渡
        ! 移动到预抓取位姿，偏移值可以根据实际需求修改
        MoveL Offs(grasp_pose, 0, 0, 100), gMotSpeed, gMotZone, gMotTool \
↪WObj:=gMotWobj;
        ! 移动到抓取点
        MoveL grasp_pose, gMotSpeed, gMotZone, gMotTool \Wobj:=gMotWobj;
        Set output0;   ! 控制数字输出
        WaitTime 0.5;

        ! S19: 运动到隔板的放置位姿
        ! 注意: 可能需要添加其他路径点作为过渡
        ! 注意需要先定义隔板放置位姿
        ! 移回预抓取位姿，偏移值可以根据实际需求修改
        MoveL Offs(grasp_pose, 0, 0, 100), gMotSpeed, gMotZone, gMotTool \
↪WObj:=gMotWobj;

        ! 此处需要添加放置隔板的路径点

        ! S20: 机械臂外的相机切换成识别当前工件
        err_code := xyzSwitchItem(0,"item1");  ! 切换工件
        xyzCheckErrorCode(err_code);

    ELSE    ! 如果识别数量不为0
        ! S8: 运动到拍照位姿
        ! 注意需要先定义 scan_pose 的位姿，可以通过 RobotStudio 来修改
        ! 或者用使用其他点来替代
        MoveL scan_pose, gMotSpeed, gMotZone, gMotTool \Wobj:=gMotWobj;  ! 运动到␣
↪scan pose

        ! S9: 发送拍照位姿
        err_code := xyzSendCurrentCartPose(scan_pose);
        xyzCheckErrorCode(err_code);
```

(下页继续)

(续上页)

```
        ! S10: 机械臂上的相机切换成识别当前工件
        err_code := xyzSwitchItem(1,"item1");  ! 切换工作空间与工件
        xyzCheckErrorCode(err_code);

        ! S11: 请求机械臂上的相机获取工件抓取位姿
        err_code = xyzReqGraspPose(1, token1);  ! 请求抓取位姿
        xyzCheckErrorCode(err_code);

        ! S12: 获取工件的抓取位姿
        err_code = xyzGetGraspPose(token1, pose_num, pipeline_num, register_num,␣
↪grasp_pose); ! 获取抓取位姿
        xyzCheckErrorCode(err_code);
        IF pose_num < 1 THEN  !  未识别到工件的处理
            TPWrite "Get grasp pose failed";
            Stop;
        ENDIF

        ! S13: 运动到工件的抓取位姿并进行抓取
        ! 注意: 可能需要添加其他路径点作为过渡
        ! 移动到预抓取位姿, 偏移值可以根据实际需求修改
        MoveL Offs(grasp_pose, 0, 0, 100), gMotSpeed, gMotZone, gMotTool \
↪WObj:=gMotWobj;
        ! 移动到抓取点
        MoveL grasp_pose, gMotSpeed, gMotZone, gMotTool \Wobj:=gMotWobj;
        Set output1;   ! 控制数字输出
        WaitTime 0.5;

        ! S14: 运动到工件的放置位姿
        ! 注意: 可能需要添加其他路径点作为过渡
        ! 注意: 需要定义工件放置位姿
        ! 移回预抓取位姿, 偏移值可以根据实际需求修改
        MoveL Offs(grasp_pose, 0, 0, 100), gMotSpeed, gMotZone, gMotTool \
↪WObj:=gMotWobj;

        ! 此处需要添加放置工件的路径点

        Reset output1;   ! 重置数字输出

    ENDIF
ENDWHILE
ENDPROC
```

轨迹移动同步模板

```
PROC TrajMoveSync()
! S1: 初始化
VAR num err_code;           ! 错误代码
VAR num ws_id;              ! 工作空间
VAR num pose_num;           ! 轨迹点的个数
VAR num pipeline_num;       ! 处理流程编号
VAR num register_num;       ! 抓取序号
VAR num wp_type{30};
VAR jointtarget home_pose;  ! home 点位
VAR jointtarget joint_wp{30};
```

(下页继续)

(续上页)

```
VAR robtarget cart_wp{30};

! S2: 连接到工控机
xyzCreateSocket master_socket;
xyzConnect master_socket, master_server_ip, master_server_port;

ws_id := 0;
err_code := 0;

! err_code := xyzSwitchFlow("traj_sync.t"); !␣
↪切换任务流图，默认被注释掉，如果需求可以取消注释
! xyzCheckErrorCode(err_code);

! S3: 切换成当前工件
! 切换工件，两个参数分别表示视觉服务 ID ，和工件代号
! 工件代号为 MAX 中映射表与通讯协议设置中的参数
err_code := xyzSwitchItem(0,"item1");
xyzCheckErrorCode(err_code);

Reset output0;
WaitTime 0.5;

! S4: 运动到 home 位
! 首先移动到 home 点。注意需要先定义 home_pose 的关节角度值，可以通过 RobotStudio␣
↪来修改
! 或者用使用其他点来替代
MoveAbsJ home_pose, gMotSpeed, gMotZone, gMotTool \Wobj:=gMotWobj;

WHILE TRUE DO
    ! S5: 请求抓取和放置规划
    err_code := xyzReqPickPlace(ws_id);
    xyzCheckErrorCode(err_code);

    ! S6: 获取 pick-in 轨迹
    err_code := xyzGetPickin(ws_id, pose_num, pipeline_num, register_num, wp_type,␣
↪joint_wp, cart_wp);
    xyzCheckErrorCode(err_code);

    ! S7: 判断是否识别到工件
    IF pose_num < 1 THEN
        TPWrite "Tote Cleared";
        Stop;
    ENDIF

    ! S8: 执行 pick-in 轨迹
    xyzExecuteTraj pose_num, pipeline_num, register_num, wp_type, joint_wp, cart_wp;
    Set output0;

    ! S9: 获取 pick-out 轨迹
    err_code := xyzGetPickout(ws_id, pose_num, pipeline_num, register_num, wp_type,␣
↪joint_wp, cart_wp);
    xyzCheckErrorCode(err_code);

    ! S10: 执行 pick-out 轨迹
    xyzExecuteTraj pose_num, pipeline_num, register_num, wp_type, joint_wp, cart_wp;
```

(下页继续)

(续上页)

```
    ! S11: 获取 place-in 轨迹
    err_code := xyzGetPlacein(ws_id, pose_num, pipeline_num, register_num, wp_type,␣
↪joint_wp, cart_wp);
    xyzCheckErrorCode(err_code);

    ! S12: 执行 place-in 轨迹
    xyzExecuteTraj pose_num, pipeline_num, register_num, wp_type, joint_wp, cart_wp;
    Reset output0;

    ! S13: 获取 place-out 轨迹
    err_code := xyzGetPlaceout(ws_id, pose_num, pipeline_num, register_num, wp_type,␣
↪joint_wp, cart_wp);
    xyzCheckErrorCode(err_code);

    ! S14: 执行 place-out 轨迹
    xyzExecuteTraj pose_num, pipeline_num, register_num, wp_type, joint_wp, cart_wp;
ENDWHILE
ENDPROC
```

轨迹移动异步模板

```
PROC TrajMoveAsync()
! S1: 初始化
VAR num err_code;          ! 错误代码
VAR num ws_id;             ! 工作空间 id
VAR num pose_num;          ! 轨迹点的个数
VAR num pipeline_num;      ! 处理流程编号
VAR num register_num;      ! 抓取序号
VAR num wp_type{30};
VAR jointtarget home_pose;      ! 示教器的home点位
VAR jointtarget joint_wp{30};
VAR robtarget cart_wp{30};

! S2: 连接到工控机
xyzCreateSocket master_socket;
xyzConnect master_socket, master_server_ip, master_server_port;

ws_id := 0;
err_code := 0;

! err_code := xyzSwitchFlow("traj_async.t"); !␣
↪切换任务流图，默认被注释掉，如果需求可以取消注释
! xyzCheckErrorCode(err_code);

! S3: 切换成当前工件
! 切换工件，两个参数分别表示视觉服务 ID ，和工件代号
! 工件代号为 MAX 中映射表与通讯协议设置中的参数
err_code := xyzSwitchItem(0,"item1");
xyzCheckErrorCode(err_code);

Reset output0;
WaitTime 0.5;

! S4: 运动到 home 位
```

(下页继续)

(续上页)

```
! 首先移动到 home 点。注意需要先定义 home_pose 的关节角度值，可以通过 RobotStudio␣
↪来修改
! 或者用使用其他点来替代
MoveAbsJ home_pose, gMotSpeed, gMotZone, gMotTool \Wobj:=gMotWobj;

! S5: 请求抓取和放置规划
err_code := xyzReqPickPlace(ws_id);  ! 请求抓取和放置规划
xyzCheckErrorCode(err_code);

WHILE TRUE DO
    ! S6: 获取 pick-in 轨迹
    err_code := xyzGetPickin(ws_id, pose_num, pipeline_num, register_num, wp_type,␣
↪joint_wp, cart_wp);
    xyzCheckErrorCode(err_code);

    ! S7: 判断是否识别到工件
    IF pose_num < 1 THEN
        TPWrite "Tote Cleared";
        Stop;
    ENDIF

    ! S8: 执行 pick-in 轨迹
    xyzExecuteTraj pose_num, pipeline_num, register_num, wp_type, joint_wp, cart_wp;
    Set output0;

    ! S9: 获取 pick-out 轨迹
    err_code := xyzGetPickout(ws_id, pose_num, pipeline_num, register_num, wp_type,␣
↪joint_wp, cart_wp);
    xyzCheckErrorCode(err_code);

    ! S10: 执行 pick-out 轨迹
    xyzExecuteTraj pose_num, pipeline_num, register_num, wp_type, joint_wp, cart_wp;

    ! S11: 请求下一次的抓取和放置规划
    ! 运动到相机视野外，开始请求下一次的抓取和放置规划
    err_code := xyzReqPickPlace(ws_id);
    xyzCheckErrorCode(err_code);

    ! S12: 获取本次 place-in 轨迹
    ! 继续请求本次 place-in 轨迹
    err_code := xyzGetPlacein(ws_id, pose_num, pipeline_num, register_num, wp_type,␣
↪joint_wp, cart_wp);
    xyzCheckErrorCode(err_code);

    ! S13: 执行本次 place-in 轨迹
    xyzExecuteTraj pose_num, pipeline_num, register_num, wp_type, joint_wp, cart_wp;
    Reset output0;

    ! S14: 获取本次 place-out 轨迹
    err_code := xyzGetPlaceout(ws_id, pose_num, pipeline_num, register_num, wp_type,␣
↪joint_wp, cart_wp);
    xyzCheckErrorCode(err_code);

    ! S15: 执行本次 place-out 轨迹
    xyzExecuteTraj pose_num, pipeline_num, register_num, wp_type, joint_wp, cart_wp;
ENDWHILE
ENDPROC
```

常见问题

1. 如果运行时发现 **转角路径故障**，通常是 MAX 任务流图设置的圆滑度（ZONE）过大，可以调整到零或较小值。

附录

14.2.2 ae

此处介绍安装 ae 配天机械臂驱动的相关事项。

驱动版本和使用要求

支持的机械臂类型

配天六轴机械臂

控制器型号

配天 icube 控制器

控制器需要开通的功能

1. 自带 Socket 功能，无需额外开通
2. 控制器版本需要为 2.6.5, 其他版本使用遇到问题请联系开发人员。

   下面是使用示教器查看版本的方法，

   - 使用真实示教器，登录用户解锁，各权限密码如下：
     - 集成商：GRACE
     - 示教员：PEACE
     - 操作员：LOVE
   - 在示教器，进入 系统-> 系统与更新->“系统信息 “ 查看版本，需要查看的是 HMI 软件和 ARCS 软件版本。

使用 RPsim PC 示教器

RPsim 是 ae 官方的编写代码、运行和调试程序的软件，可在 ae 官网下载或向 ae 售后索要软件，使用同真实示教器一致。**推荐使用！**

使用过程参照 RPsim 安装包的 txt 文档，若出现乱码，可以通过 notepad++ 或其他高级文本软件进行查看。

按照操作说明，若仍不能打开 ARCS 共享目录：

1. 需要修改 windows 服务。

图 20: 示教器查看当前版本 1

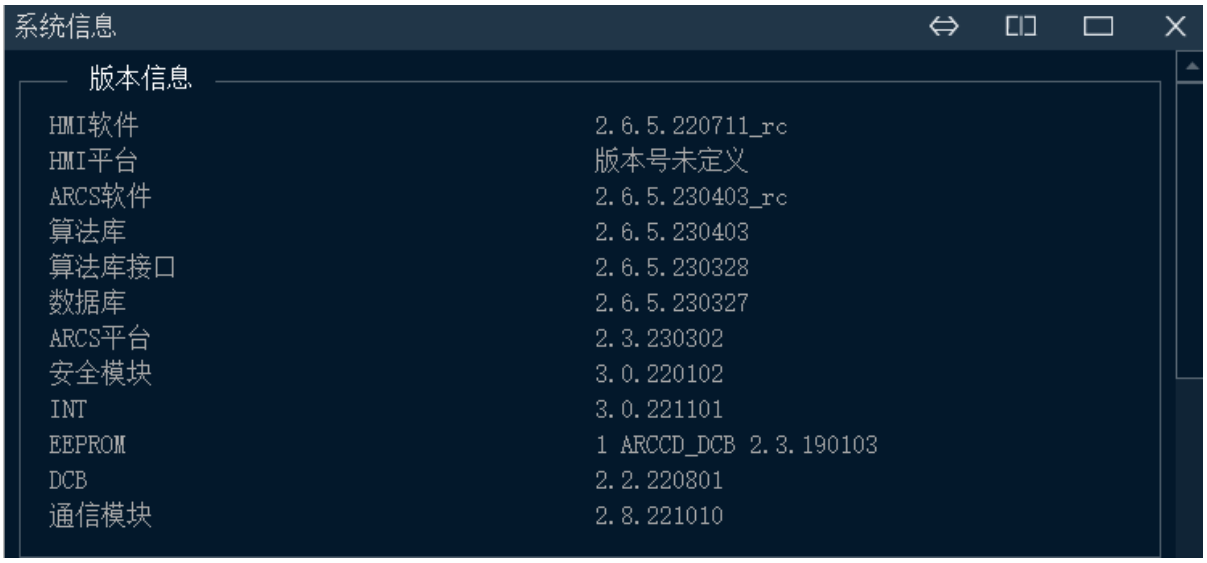


图 21: 示教器查看当前版本 2

具体操作为 `win + R` 键，输入 services.msc，找到 **Server** 和 **DNS Client**，两者都必须是 **自动**，若不是，需要双击进入每一个服务，将启动类型改为自动。

若 DNS Client 无法修改为自动，则需要以管理员权限打开 windows terminal 或 powershell，输入

```
REG add "HKLM\SYSTEM\CurrentControlSet\services\dnscache" /v Start␣
↪/t REG_DWORD /d 2 /f
```

2. `win + s` 搜索框输入控制面板，进入 网络和共享中心 -> 高级共享设置，勾选启用网络发现，点击保存更改。

图 22: 启用网络发现

设置好，再次尝试打开共享目录，成功的话，就可以双击 RPsim 软件，打开 PC 示教器了。

安装驱动

ae 机械臂驱动文件列表

机械臂代码可从 MAX 安装目录下的 share/robot_code/inovance 中获得。

├── xyz_motion.arl **工控机主控程序**
├── xyz_master_api.arl **机械臂主控测试程序**
├── xyzData.pro 常数定义
├── xyz_status.pro **后台通道任务**

设定机械臂 IP

1. 集成商权限登录进示教器后，进入 系统 -> 系统配置 -> 网络配置，根据网线连接的控制柜的网口进行配置，EtherNet 网口 A 对应用户网口 1，EtherNet 网口 B 对应用户网口 2。
2. 配置网络如下所示

- IP 地址：192.168.37.100
- 子网掩码：255.25.255.0
- 网关：192.168.37.1

3. 保存后，重启示教器。

使用 RPsim 导入程序

- 进入 文件 -> 文件管理，进入 `/script/` 目录，新建 `xyz` 文件夹
- 依次按照 robot_code/ae 中的代码，新建同名文件，将代码复制粘贴到文件中。

使用 U 盘导入程序

1. 将 MAX 安装目录下的 share/robot_code 中 ae 的机械臂代码拷入 fat32 U 盘，插入示教器 USB 口
2. 进入 `script` 文件夹，新建 xyz 文件夹
3. 在示教器主界面，点击 文件 -> 恢复备份 -> 程序文件，弹出如图所示的 导入程序列表框。

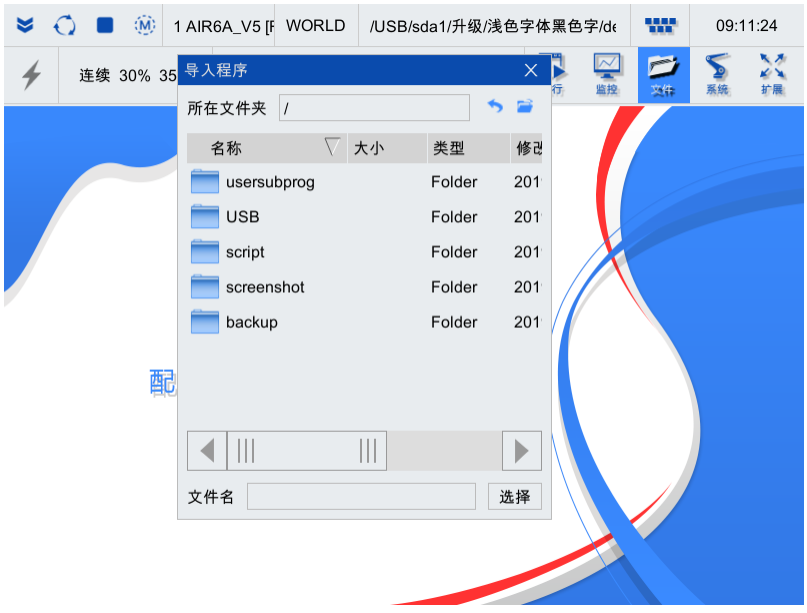


图 23: U 盘导入程序

4. 选择所需的文件，即可将程序导入到系统中。

运行程序

通讯说明

工控机和 ae 机械臂的通讯方式为 socket：工控机作为 socket server(服务端)，机械臂作为 socket client（客户端）。

运行程序（推荐使用 RPsim）

1. 配置工控机主控

   配置前台任务

   - 点击 文件 -> 文件管理, 进入到存放代码的路径，打开 xyz-motion.arl

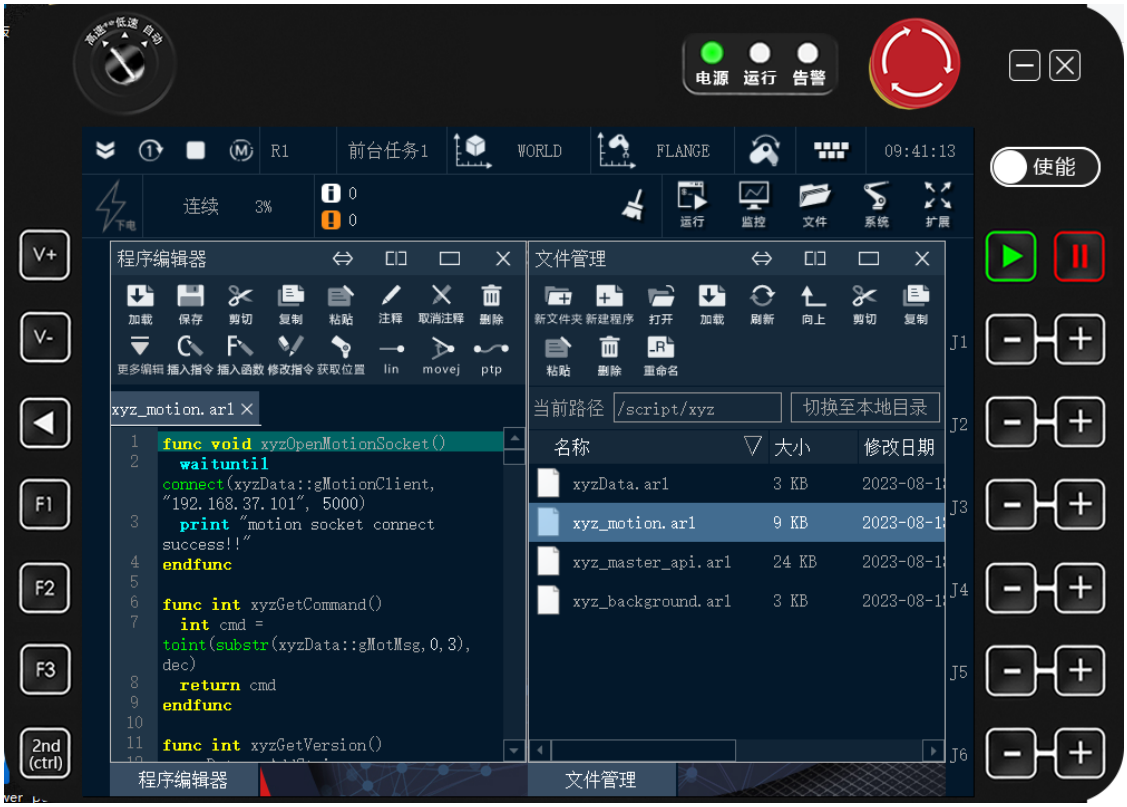


图 24: 打开 xyz_motion.arl

   - 查看当前通道是否为前台任务 1。如果不是，点击 后台任务 1，勾选 前台任务，即可切换为前台。
   - 在 程序编辑器界面，点击加载，如果没有任何错误提醒，则会弹出 程序调试器界面。

   配置后台任务

   - 点击 文件 -> 文件管理, 进入到存放代码的路径，打开 xyz_status.arl
   - 查看当前通道是否为后台任务 1。如果不是，点击 后台任务 1，勾选 后台任务，即可切换为后台。

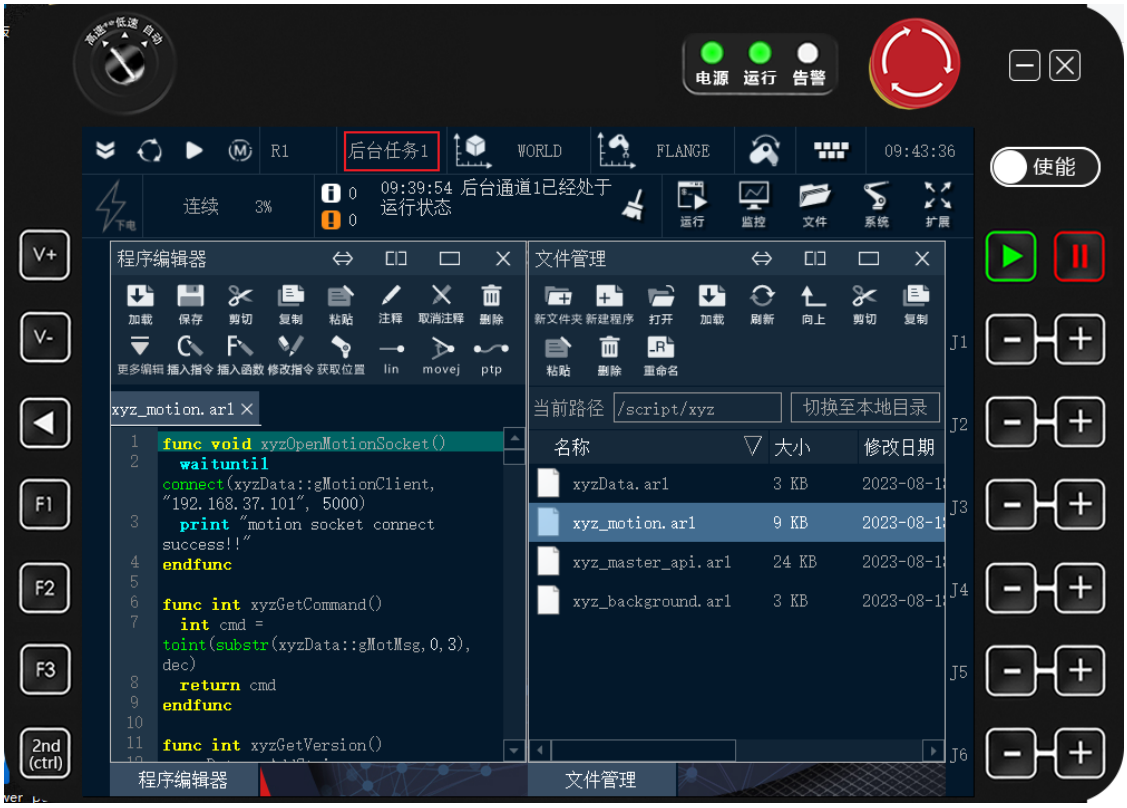


图 25: 切换为前台任务

- 在 程序编辑器界面，点击加载，如果没有任何错误提醒，则会弹出 程序调试器界面，这时后台任务已经配置成功。

工控机主控运行

- 在 程序调试器界面，复位前台程序
- 点击前台任务选择界面，对后台任务，依次点击 暂停，复位
- 工控机点击连接机器人，示教器上电后，点击示教器上的运行按钮。

2. 配置机械臂主控

- 点击 文件 -> 文件管理，进入到存放代码的路径，打开 xyz-master_api.arl
- 修改代码后，点击 加载配置为前台任务，如果没有任何错误提醒，则会弹出 程序调试器界面。
- MAX 打开 robot server 后，示教器点击运行按钮。
- 如果程序意外停止，复位程序，再重新运行。

API 说明

ae 工控机主控支持的 API

指令号	命令	支持情况
100	获取机械臂程序版本号	支持
101	发送速度数据	支持
102	发送加速度数据	支持
103	发送圆滑过渡参数	支持
104	发送工具坐标系 (TCP) 参数	支持
105	设置数字量输出	支持
106	SetJointsMovej	支持
107	SetCartMovel	支持
108	SetJoinsMovel	支持
109	SetCartMovej	支持
110	MovejSequence	支持
111	MovelSequence	支持
112	SetJointsMovejDo	不支持
113	SetCartMovelDo	不支持
114	SetJointsMovelDo	不支持
115	SetJointsMovejGroupDo	不支持
116	SetCartMovelGroupDo	不支持
117	SetJointsMovelGroupDo	不支持
118	MovelUntil	支持
119	获取数字量输入状态	支持
120	获取数字量输出状态	支持
121	获取模拟量输入口数值	不支持
122	获取机械臂当前角度	支持
123	获取机械臂当前位姿	支持
200	机械臂后台发送状态	支持

ae 机械臂主控支持的 API

xyzSwitchAPP(*app_name*)

切换应用

参数 **app_name** –应用名称

返回 err_code 为 0 表示成功

xyzSwitchFlow(*flow_name*)

切换工件

参数 **flow_name** –流图名称

返回 err_code 为 0 表示成功

xyzSwitchTool(*tool_name*)

切换工具

参数 **tool_name** –工具名称

返回 err_code 为 0 表示成功

xyzReqCapImg(*vision_service_id*)

请求拍照

参数 vision_service_id –视觉服务 id

返回

err_code 为 0 表示成功

xyzData::gCapToken 返回的 token

xyzGetCapImg(*xyzData::gCapToken*)

获取拍照结果

参数 xyzData::gCapToken –请求拍照时返回的 token

返回 err_code 为 0 表示成功

xyzCapImg(*vision_service_id*)

拍照

参数 vision_service_id –需要进行拍照操作的工作空间 id

返回 err_code 为 0 表示成功

xyzReqGraspPose(*ws_id*)

请求抓取位姿

参数 ws_id –需要获取抓取点位的工作空间 id

返回

err_code 为 0 表示成功

xyzData::gGraspToken: 返回的用于获取目标点位时使用的 token

xyzGetGraspPose(*xyzData::gGraspToken*)

获取抓取位姿

参数 grasp_pose_token –求抓取目标点位时返回的 token

返回

err_code 为 0 表示成功

xyzData::gGraspPose: 抓取位姿

xyzData::gGraspPoseNum: 可供抓取的点数量

xyzData::gPipelineNum: pipeline 文件 number

xyzData::gRegisterNum: 用到的注册文件的注册 number

xyzReqObjPose(*ws_id*)

请求物体位姿

参数 ws_id –需要获取物体位姿的工作空间 id

返回

err_code 为 0 表示成功

xyzData::gObjToken: 返回的用于物体位姿识别的 token

xyzGetObjPose(*xyzData::gObjToken*)

获取物体位姿

参数 obj_pose_token –请求物体位姿时得到的 token

返回

err_code 为 0 表示成功

xyzData::gObjPose: 物体位姿

xyzData::gObjPoseNum: 物体数量

xyzData::gObjPoseType: 当前返回的物体 pose 类型

xyzResetTask()

重置视觉

参数 ws_id –需要重置的任务 id

返回 err_code 为 0 表示成功

xyzSendCurrentJoints()

发送机械臂当前角度：j1~j6: 机械臂当前的角度信息，如果机械臂轴数小于 6 轴，则需要对应位置补零，依旧发送六个数。

返回 err_code 为 0 表示成功

xyzSendCurrentCartPose()

发送机械臂法兰当前位姿

返回 err_code 为 0 表示成功

xyzSendCurrentExtJoints()

暂不支持。

发送机械臂当前扩展轴位置：j1~j6: 机械臂当前扩展轴的角度信息，如果扩展轴数不足 6 的, 需要补零后发送六个数。

返回 err_code 为 0 表示成功

xyzReqPick()

暂不支持。请求 pick 动作规划

返回 err_code 为 0 表示成功

xyzReqPlace()

暂不支持。请求 place 动作规划

返回 err_code 为 0 表示成功

xyzReqPickPlace(*ws_id*)

请求 pick 和 place 规划

参数 ws_id –需要抓取的工作空间 id

返回 err_code 为 0 表示成功

xyzGetPickin(*ws_id*)

获取取料入框轨迹

参数 ws_id –规划空间 id，默认填 0

返回

err_code 为 0 表示成功

xyzData::gPipelineNum pipeline 文件 number

xyzData::gRegisterNum 用到的注册文件的注册 number

xyzGetPickout(*ws_id*)

获取取料出框轨迹

参数 ws_id –规划空间 id，默认填 0

返回

err_code 为 0 表示成功

xyzData::gPipelineNum pipeline 文件 number

xyzData::gRegisterNum 用到的注册文件的注册 number

xyzGetPlacein(*ws_id*)

获取放料入框轨迹

参数 ws_id –规划空间 id，默认填 0

返回

err_code 为 0 表示成功

xyzData::gPipelineNum pipeline 文件 number

xyzData::gRegisterNum 用到的注册文件的注册 number

xyzGetPlaceout(*ws_id*)

获取放料出框轨迹

参数 ws_id –规划空间 id，默认填 0

返回

err_code 为 0 表示成功

xyzData::gPipelineNum pipeline 文件 number

xyzData::gRegisterNum 用到的注册文件的注册 number

xyzSwitchStrat(*strategy_name*)

请求切换策略

参数 strategy_name –策略名称

返回 err_code 为 0 表示成功

xyzUpdateTotePose(*err_code*)

料箱重定位

返回

err_code 为 0 表示成功

tote_pose: 料箱位姿

xyzUpdateObjPoseOnHand(*err_code*)

工件在上手的二次定位

返回 err_code 为 0 表示成功

xyzUpdateObjPoseToHand()

工件不在手上的二次定位

返回

err_code 为 0 表示成功

xyzData::gPipelineNum pipeline 文件 number

xyzData::gRegisterNum 用到的注册文件的注册 number

xyzGetObjPoseType()

获取工件姿态类型

返回

err_code 为 0 表示成功

xyzData::gObjPoseType: 工件姿态类型

xyzResetPalletStatus()

重置工业码垛状态

返回 err_code 为 0 表示成功

xyzSwitchItem(*item_codename*)

切换工件

参数

- **xyzData::gWsId** –工作空间 id
- **item_codename** –工件名称

返回 err_code 为 0 表示成功

xyzCalculateGraspPose(*ws_id*)

计算抓取位姿

参数 ws_id –工作空间 id

返回

err_code 为 0 表示成功

xyzData::gGraspPose: 抓取位姿

xyzData::gGraspPoseNum: 可供抓取的点数量

xyzData::gPipelineNum: pipeline 文件 number

xyzData::gRegisterNum: 用到的注册文件的注册 number

xyzCalculateObjPose(*ws_id*)

计算物体位姿

参数 ws_id –工作空间 id

返回

err_code 为 0 表示成功

xyzData::gObjPose: 物体位姿

xyzData::gObjPoseNum: 物体数量

xyzData::gObjPoseType: 当前返回的物体 pose 类型

xyzUserCommand()

用户自定义发送数据。使用请按照参照 xyz-master_api

发送给工控机：

xyzData::out_str： 提供 5 个字符串供用户输入

xyzData::out_double：提供 10 个整数类型供用户输入

xyzData::out_double：提供 10 个浮点数供用户输入

xyzData::out_pose： cart_pose 点位坐标，如果机器人是欧拉角形式，则只需给 a,b,c 赋值，d 赋值 0 即可

xyzData::out_joint：关节角坐标值

工控机返回：

xyzData::out_str： 返回 5 个字符串

xyzData::in_int： 提供 10 个整数类型供用户输入：返回 10 个整数值

xyzData::in_double： 提供 10 个浮点数供用户输入：返回 10 个浮点数

xyzData::in_pose： cart_pose 点位坐标，如果机器人是欧拉角形式，则只需给 a：返回一个笛卡尔位姿数据

xyzData::in_joint： 关节角坐标值：返回一个关节坐标

返回 err_code 为 0 表示成功

案例/模板说明

常见问题

附录

14.2.3 Aubo

此处介绍安装 aubo 机械臂驱动的相关事项。

驱动版本和使用要求

支持的机械臂类型

Aubo 六轴协作机械臂

控制器型号

AUBO-CB-M

机械臂需要开通的功能

Aubo 自带 socket 通讯功能，无需开通。

安装驱动

aubo 机械臂驱动文件列表

–**aubo**

- xyz_master.aubo （机械臂主控程序）
- xyz_motion.aubo （工控机主控程序）
- xyz_status.aubo （工控机主控程序）

设定机械臂 IP

设置-> 系统-> 网络

- “IP”设置为：192.168.37.101

设定工控机 IP

1. 用网线连接电脑和控制柜上的 **网线接口**。
2. 将电脑的 ipv4 网络设定为:

- “IP”设置为：192.168.37.100
- “子网掩码”设置为：255.255.255.0

导入程序

1. 将装有 aubo 机械臂驱动程序的 u 盘插入控制柜上的 usb 接口
2. 将 u 盘插入机械臂控制柜上的 usb 接口后会弹出 u 盘的窗口，找到所需脚本后长按选择 Copy to recent-script 文件夹下即可

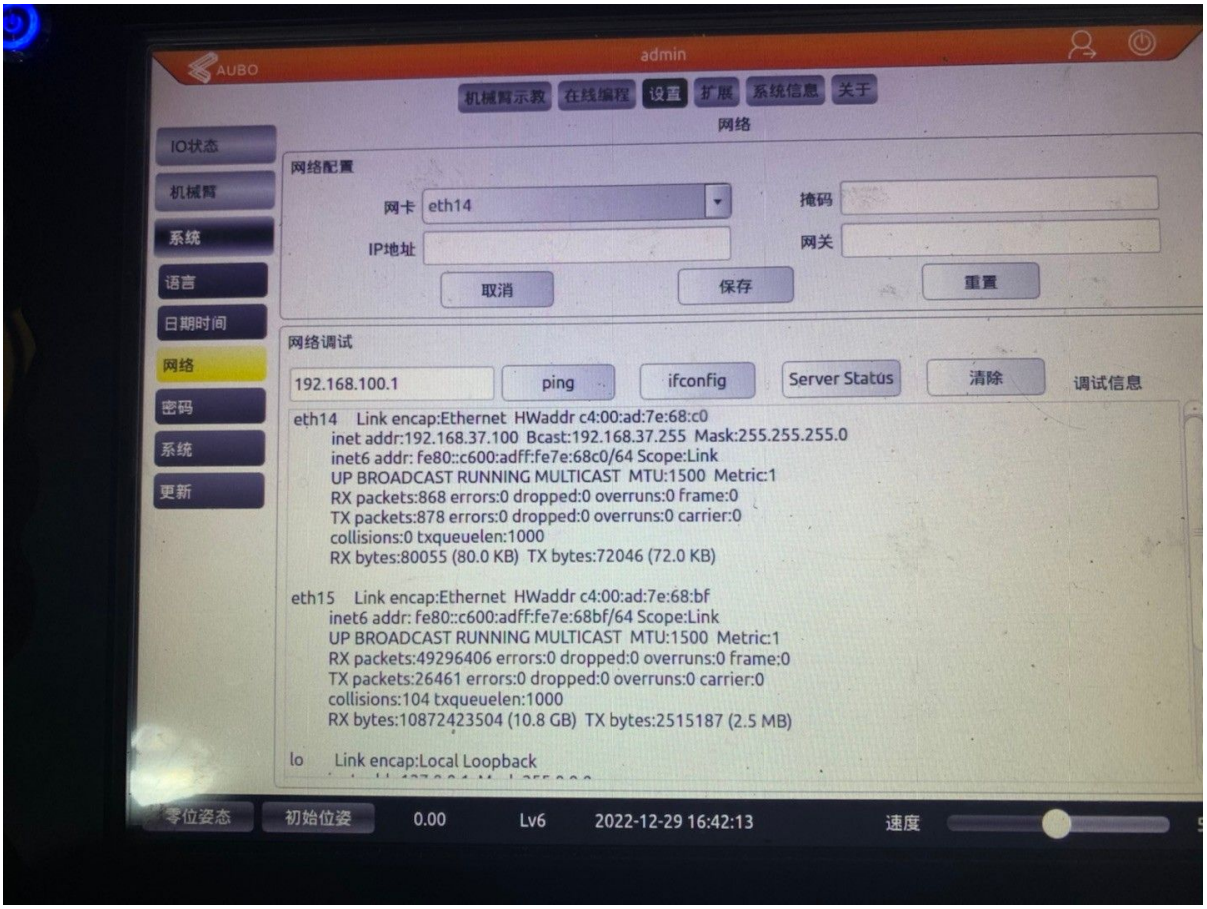


图 26: aubo 设定机械臂 IP

运行程序

通讯说明

工控机和 aubo 机械臂的通讯方式为 TCP/IP：xyz_motion 中工控机作为 socket server（服务端），机械臂作为 socket client（客户端），xyz_status 中工控机作为 socket client（客户端），机械臂作为 socket server（服务端）。

启动程序

工控机主控设置

条件-> 高级条件->Thread 加入一个新线程

Project_Program 和 Thread 下分别加入一个高级条件->script-> 脚本文件，分别导入 xyz_status 和 xyz_motion 后即可，工控机开启后点击开始即可。

示教器设置

机械臂主控设置

同工控机主控，此时不需要添加线程，在机械臂程序下添加 `xyz_master` ，简单的用法示例如下图只要在 Project_Program 下面先导入一下 xyz_master

后续通过条件-> 高级条件-> 脚本-> 行脚本来调用 xyz_master 中的函数即可，昵称可随意设定，下面的内容填写要调用的函数即可

API 说明

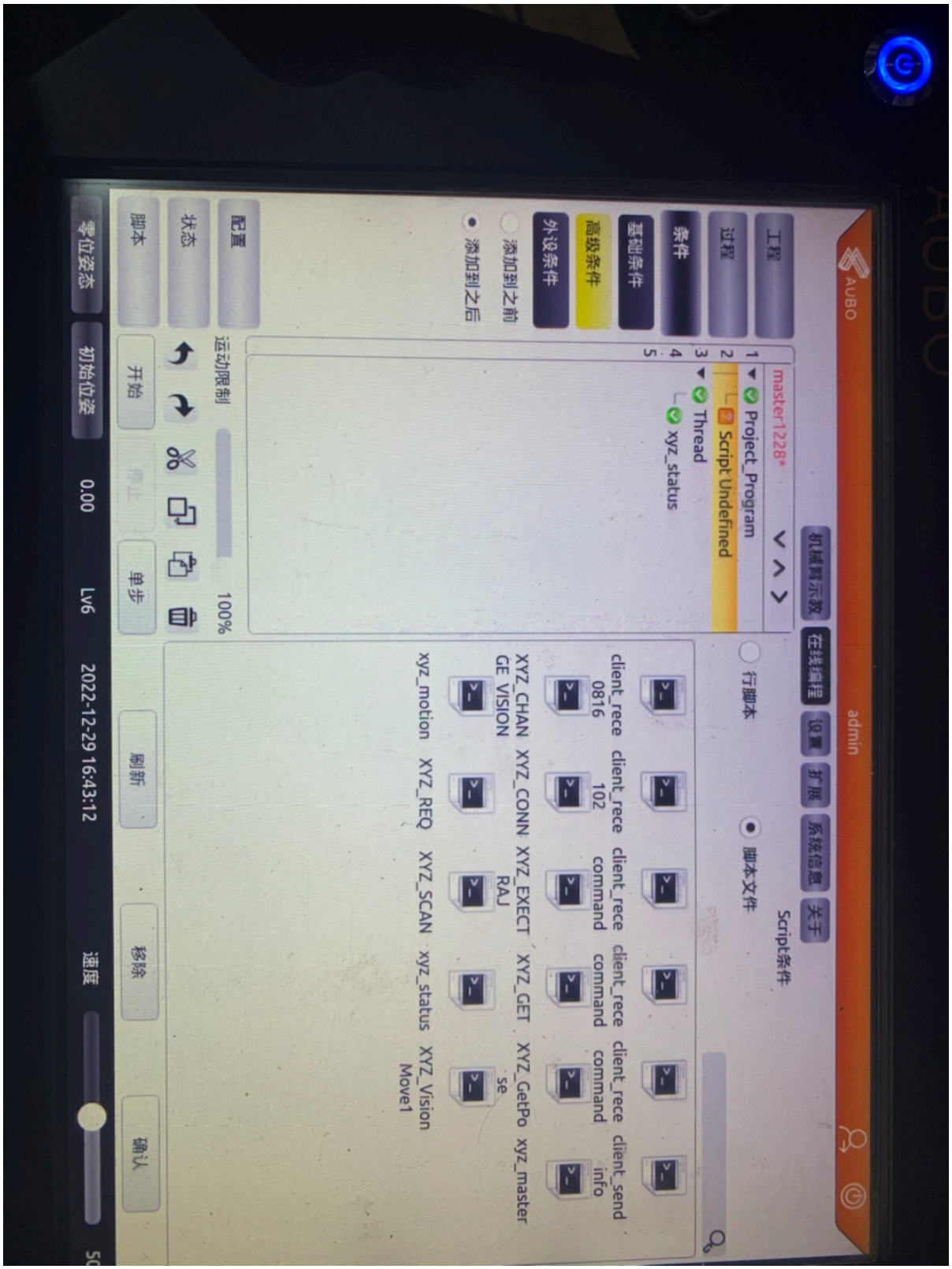


图 27: 工控机主控程序设置

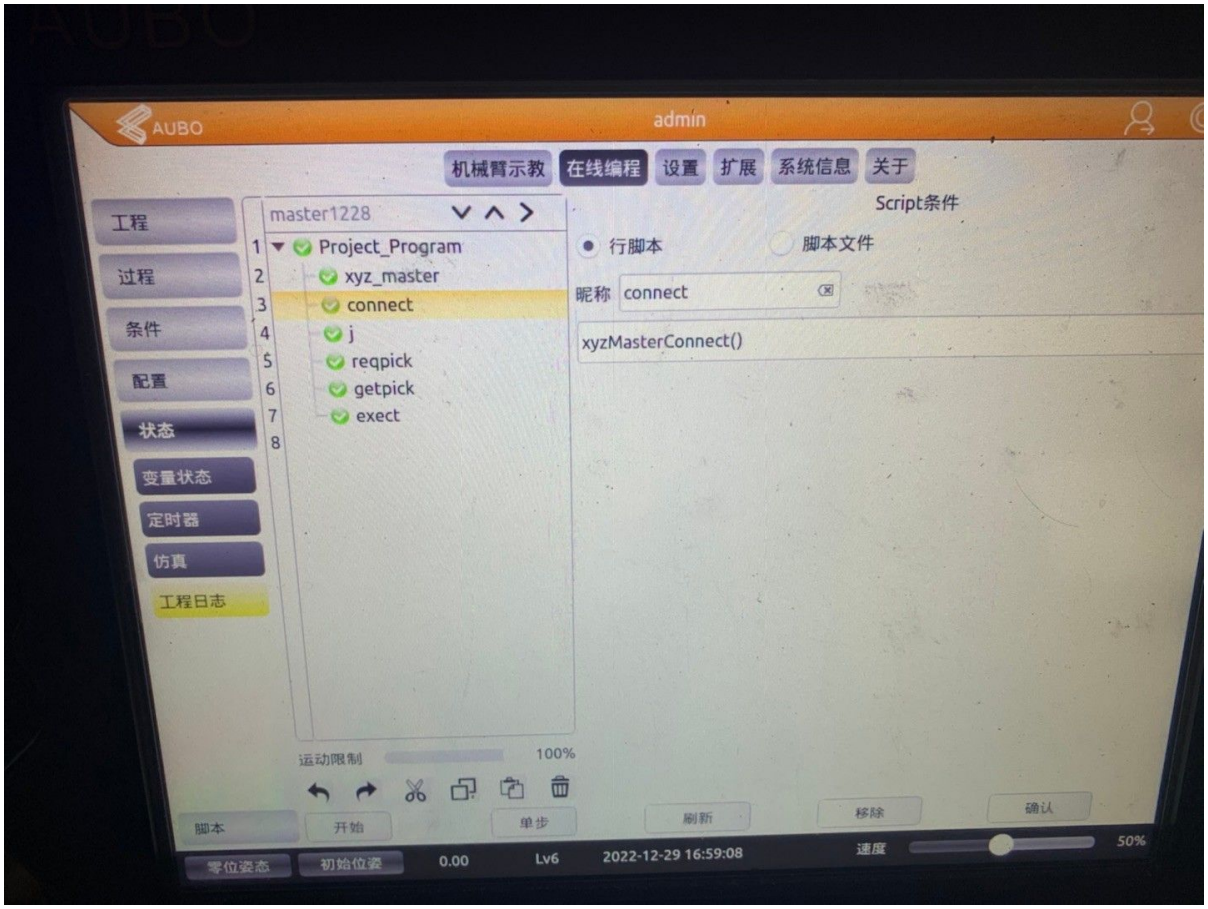


图 28: 机械臂主控使用示例

aubo 工控机主控支持的 API

指令号	命令	支持情况
100	获取机械臂程序版本号	支持
101	发送速度数据	支持
102	发送加速度数据	支持
103	发送圆滑过渡参数	支持
104	发送工具坐标系 (TCP) 参数	支持
105	设置数字量输出	支持
106	SetJointsMovej	支持
107	SetCartMovel	支持
108	SetJoinsMovel	支持
109	SetCartMovej	支持
110	MovejSequence	支持
111	MovelSequence	支持
112	SetJointsMovejDo	不支持
113	SetCartMovelDo	不支持
114	SetJointsMovelDo	不支持
115	SetJointsMovejGroupDo	不支持
116	SetCartMovelGroupDo	不支持
117	SetJointsMovelGroupDo	不支持
118	MovelUntil	不支持
119	获取数字量输入状态	支持
120	获取数字量输出状态	支持
121	获取模拟量输入口数值	支持
122	获取机械臂当前角度	支持
123	获取机械臂当前位姿	支持
200	机械臂后台发送状态	支持

aubo 机械臂主控支持的 API

xyzSwitchApp(*app_name*)

切换应用

参数 **app_name**(*string*) –应用名称

返回 error_code

返回类型 num

xyzSwitchObj(*obj_name*)

切换工件

参数 **obj_name**(*string*) –工件名称

返回 error_code

返回类型 num

xyzSwitchTool(*tool_name*)

切换工具

参数 **tool_name**(*string*) –工具名称

返回 error_code

返回类型 num

xyzReqCapImg(*ws_id*)

请求拍照 (注：该函数更新全局变量)

参数

- **ws_id** (*num*) –工作空间 id
- **V_TOKEN** (*num*) –获取拍照结果的凭据 (全局变量)

返回 error_code

返回类型 num

xyzGetCapImg(*token*)

获取拍照结果

参数 **token** (*num*) –请求拍照时返回的 token

返回 error_code

返回类型 num

xyzCapImg(*ws_id*)

拍照

参数 **ws_id** (*num*) –需要进行拍照操作的工作空间 id

返回 error_code

返回类型 num

xyzReqGraspPose(*ws_id*)

请求抓取位姿 (注：该函数更新全局变量)

参数

- **ws_id** (*num*) –需要获取抓取点位的工作空间 id
- **V_TOKEN** (*num*) –返回的用于获取目标点位时使用的 token(全局变量)

返回 error_code

返回类型 num

xyzGetGraspPose(*token*)

获取抓取位姿 (注：该函数更新全局变量)

参数

- **token** (*num*) –请求抓取目标点位时返回的 token
- **V_POSE_NUM** (*num*) –可供抓取的点数量 (全局变量)
- **V_POSE_TYPE** (*num*) –当前返回的抓取点的 pose 类型 (全局变量)
- **V_POSE** (*{{px, py, pz}, {w, x, y, z}}*) –抓取位姿 (全局变量)

返回 error_code

返回类型 num

xyzReqObjPose(*ws_id*)

请求物体位姿 (注：该函数更新全局变量)

参数

- **ws_id** (*num*) –需要获取物体位姿的工作空间 id
- **V_TOKEN** (*num*) –物体位姿识别的 token(全局变量)

返回 error_code

返回类型 num

xyzGetObjPose(*token*)

获取物体位姿 (注：该函数更新全局变量)

参数

- **token** (*num*) –请求物体位姿时得到的 token
- **V_POSE_NUM** (*num*) –物体数量 (全局变量)
- **V_POSE_TYPE** –当前返回的物体 pose 类型 (全局变量)
- **V_POSE** (*{{px, py, pz}, {w, x, y, z}}*) –物体位姿 (全局变量)

返回 error_code

返回类型 num

xyzResetVision(*ws_id*)

重置视觉

参数 **ws_id** (*num*) –需要重置视觉的工作空间 id

返回 error_code

返回类型 num

xyzSendCurrentJoints()

发送当前关节位姿

返回 error_code

返回类型 num

xyzSendCurrentCartPose()

发送当前笛卡尔空间位姿

返回 error_code

返回类型 num

xyzReqPick()

请求 pick 动作规划

返回 error_code

返回类型 num

xyzReqPlace()

请求 place 动作规划

返回 error_code

返回类型 num

xyzReqPickPlace()

请求 pick 和 place 规划

返回 error_code

返回类型 num

xyzGetPickin()

获取取料入框轨迹 (注：该函数不需要传入参数，仅更新全局变量)

参数

- **V_POSE_TYPE** (*num*) –轨迹点 pose 类型 (全局变量)
- **V_WP_TYPE[]** (*array*) –轨迹点类型数组 (全局变量)
- **V_WAY_POINT[]** (*2d-array*) –轨迹点关节位姿数组 (全局变量)

返回 error_code

返回类型 num

xyzGetPickout()

获取取料出框轨迹 (注：该函数不需要传入参数，仅更新全局变量)

参数

- **V_POSE_TYPE** (*num*) –轨迹点 pose 类型 (全局变量)
- **V_WP_TYPE[]** (*array*) –轨迹点类型数组 (全局变量)
- **V_WAY_POINT[]** (*2d-array*) –轨迹点关节位姿数组 (全局变量)

返回 error_code

返回类型 num

xyzGetPlacein()

获取放料入框轨迹 (注：该函数不需要传入参数，仅更新全局变量)

参数

- **V_POSE_TYPE** (*num*) –轨迹点 pose 类型 (全局变量)
- **V_WP_TYPE[]** (*array*) –轨迹点类型数组 (全局变量)
- **V_WAY_POINT[]** (*2d-array*) –轨迹点关节位姿数组 (全局变量)

返回 error_code

返回类型 num

xyzGetPlaceout()

获取放料出框轨迹 (注：该函数不需要传入参数，仅更新全局变量)

参数

- **V_POSE_TYPE** (*num*) –轨迹点 pose 类型 (全局变量)
- **V_WP_TYPE[]** (*array*) –轨迹点类型数组 (全局变量)
- **V_WAY_POINT[]** (*2d-array*) –轨迹点关节位姿数组 (全局变量)

返回 error_code

返回类型 num

xyzSwitchStrat(*strat_name*)

请求切换策略

参数 **strat_name** (*string*) –策略名称

返回 error_code

返回类型 num

xyzUpdateTotePose()

料箱重定位 (注：该函数不需要传入参数，仅更新全局变量)

参数 **V_POSE** (*{{px, py, pz}, {w, x, y, z}}*) –料箱位姿 (全局变量)

返回 error_code

返回类型 num

xyzUpdateObjPoseOnHand()

工件在上手的二次定位

返回 error_code

返回类型 num

xyzUpdateObjPoseToHand()

工件不在手上的二次定位，获取二次抓取的轨迹 (注：该函数不需要传入参数，仅更新全局变量)

参数

- **V_POSE_TYPE** (*num*) –轨迹点 pose 类型 (全局变量)
- **V_WP_TYPE[]** (*array*) –轨迹点类型数组 (全局变量)
- **V_WAY_POINT[]** (*2d-array*) –轨迹点关节位姿数组 (全局变量)

返回 error_code

返回类型 num

xyzGetObjPoseType()

获取工件姿态类型 (注：该函数不需要传入参数，仅更新全局变量)

参数 **V_POSE_TYPE** (*num*) –工件姿态类型 (全局变量)

返回 error_code

返回类型 num

xyzResetPalletStatus()

重置工业码垛状态

返回 error_code

返回类型 num

xyzExecuteTraj()

用于执行一段轨迹，注意所有笛卡尔空间位姿与轨迹点皆为全局变量 (注：该函数不需要传入参数，调用的均为全局变量)

参数

- **V_POSE_TYPE** (*num*) –轨迹点 pose 类型 (全局变量)
- **V_WP_TYPE[]** (*array*) –轨迹点类型数组 (全局变量)
- **V_WAY_POINT[]** (*2d-array*) –轨迹点关节位姿数组 (全局变量)

返回 error_code

返回类型 num

案例/模板说明

机械臂主控主函数说明

以下为机械臂主控模板代码，注意对工控机返回的 err_code 进行判断。模板用途为所有接口的集成测试，用作实际使用的参考。

```
------------------------Test Code----------------------------------

-- This function is for testing all api
function xyzTest()
xyzTestSwitchApp()
xyzTestSwitchObj()
xyzTestSwitchTool()
-- Because of token, the next 3 test sequence can not be changed
xyzTestImg()
xyzTestGrasp()
xyzTestObj()
xyzTestResetVision()
xyzTestSendJointCartExtJoint()
xyzTestReqPickPlace()
xyzTestGetPickPlace()
xyzTestSwitchStrat()
xyzTestUpdate()
xyzTestGetObjPoseType()
xyzTestRestPalletStatus()
end

function xyzTestSwitchApp()
local app_name = "111"
local error_code = xyzSwitchApp(app_name)
app_name = "error_app"
error_code = xyzSwitchApp(app_name)
end

function xyzTestSwitchObj()
local obj_name = "222"
local error_code = xyzSwitchObj(obj_name)
obj_name = "error_obj"
error_code = xyzSwitchObj(obj_name)
end

function xyzTestSwitchTool()
local tool_name = "333"
local error_code = xyzSwitchTool(tool_name)
tool_name = "error_tool"
error_code = xyzSwitchTool(tool_name)
end

function xyzTestImg()
local ws_id = 0
local token = 1
-- token in robot server will increase after every call
local error_code = xyzReqCapImg(ws_id)
error_code = xyzGetCapImg(token)
ws_id = 0
error_code = xyzCapImg(ws_id)
```

(下页继续)

(续上页)

```
end

function xyzTestGrasp()
local ws_id = 0
local pose_num = 0
local pose_type = 0
local token = 0
-- token in robot server will increase after every call
local error_code = xyzReqGraspPose(ws_id)
token = 3
error_code = xyzGetGraspPose(token)
end

function xyzTestObj()
local pose_num = 0
local pose_type = 0

local ws_id = 0
local token = 0
-- token in robot server will increase after every call
local error_code = xyzReqObjPose(ws_id)
token = 4
error_code = xyzGetObjPose(token)
end

function xyzTestResetVision()
local ws_id = 0
local error_code = xyzResetVision(ws_id)
end

function xyzTestSendJointCartExtJoint()
local error_code = xyzSendCurrentJoints()
error_code = xyzSendCurrentCartPose()
end

function xyzTestReqPickPlace()
local error_code = xyzReqPick()
error_code = xyzReqPlace()
error_code = xyzReqPickPlace()
end

function xyzTestGetPickPlace()
local pose_type = 0
local pose_num = 0
--   please check the waypoint data before xyzExecuteTraj() !
local error_code = xyzGetPickin()
xyzExecuteTraj()
error_code = xyzGetPickout()
xyzExecuteTraj()
error_code = xyzGetPlacein()
xyzExecuteTraj()
error_code = xyzGetPlaceout()
xyzExecuteTraj()
end

function xyzTestSwitchStrat()
```

(下页继续)

(续上页)

```
local strat_name = "strat1"
local error_code = xyzSwitchStrat(strat_name)
strat_name = "error_strat"
error_code = xyzSwitchStrat(strat_name)
end

function xyzTestUpdate()
local pose_type = 0
local pose_num = 0
local error_code = xyzUpdateTotePose()
error_code = xyzUpdateObjPoseOnHand()
error_code = xyzUpdateObjPoseToHand()
end

function xyzTestGetObjPoseType()
local pose_type = 0
local error_code = xyzGetObjPoseType()
end

function xyzTestRestPalletStatus()
local error_code = xyzResetPalletStatus()
end

--------------------------------Main Function--------------------------------------

function xyzMain()
xyzMasterConnect()
xyzTest()
xyzMasterClose()
end


-- In real situation, xyzMain() should be commented or deleted
-- Call function in script instead
xyzMain()
```

常见问题

附录

14.2.4 Dobot

此处介绍安装 Dobot 机械臂驱动的相关事项。

驱动版本和使用要求

支持的机械臂类型

Dobot 六轴协作机械臂

控制器型号

DT-CR-6R162-00I

机械臂需要开通的功能

Dobot 自带 socket 通讯功能，无需开通。

安装驱动

Dobot 机械臂驱动文件列表

–**Master**

- global.lua (机械臂主控功能接口库)
- test_src0.lua
- xyz_CartMove_Basic.lua (坐标运动基础应用模板)
- xyz_CartMove_Reposition.lua (坐标运动二次定位应用模板)
- xyz_TrajMove_ASync.lua (轨迹移动异步模板)
- xyz_TrajMove_Sync.lua (轨迹移动同步模板)

–**Motion**

- motion_src0.lua (工控机主控运动控制程序)
- status_src0.lua (工控机主控机械臂状态发送程序)

设定机械臂 IP

DobotStudio Pro 是越疆针对 Dobot 机器人开发的控制软件，软件支持 Windows 和 Android 操作系统，使用 DobotStudio Pro 操作控制 Dobot 机器人非常方便快捷。

- 安装 DobotStudio Pro

  1. 下载 DobotStudio Pro 安装包

  可以从 Dobot 官网的下载中心 (https://www.dobot.cn/service/download-center) 下载最新的 DobotStudio Pro 安装包。DobotStudio Pro 是免费的，只需要注册一个 Dobot 账号便可以下载，下载中心也有 DobotStudio Pro 用户使用手册，以及其他手册，可以根据需求下载。

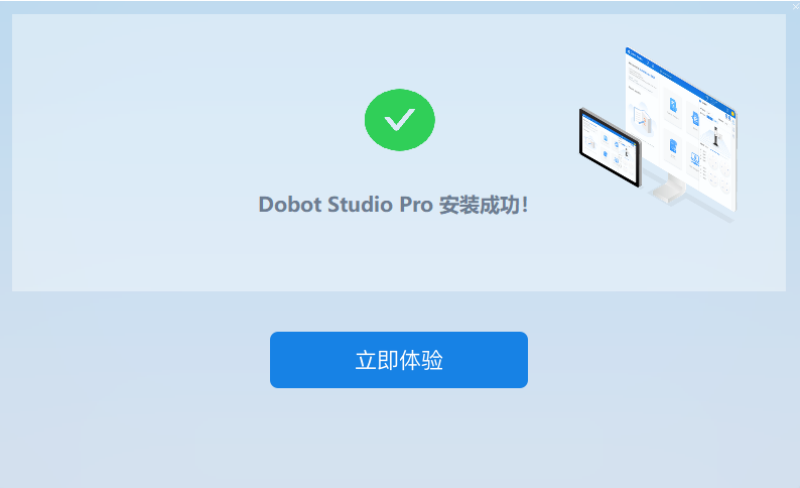
  2. 双击打开安装包后，选择安装语言，并单击下一步。

  3. 在安装界面中单击一键安装，或在自定义选项中设置安装路径后开始安装。


欢迎来到DobotStudio Pro
安装指引!
中文
下一步


多机控制
Dobot Studio Pro是越疆科技推出的一款协作机器人的电脑端的控制软件，适用于越疆M1Pro/MG400/CR系列的协作机器人。
一键安装
阅读并同意软件许可协议
自定义选项

4. 软件安装成功后在提示界面单击立即体验可以直接打开软件。



- 连接 Dobot 机器人并修改机器人 IP 地址

  Dobot 控制柜开机过程机械臂五轴上的蓝色灯会一直闪烁，表示控制柜内置 WiFi 路由器在初始化中，LAN 、WiFi 、示教器均无法连接机器人。等待蓝色灯变常亮时，表示初始化完成。Dobot 机器人控制器支持通过 wifi 和有线连接，这里我们只说明通过有线连接的方式，无线连接可以查阅 DobotStudio Pro 用户使用手册。

  1. 请将网线一端连接至控制柜的 LAN 网口，另一端连接至 PC 或者交换机，然后修改 PC 的 IP 地址，使其与控制柜的 IP 地址在同一网段。

     控制柜上 LAN1 网口的默认 IP 为 192.168.5.1，LAN2 网口的默认 IP 为 192.168.200.1。LAN1 网口的 IP 地址可在通讯设置中修改，LAN2 网口的 IP 地址不可修改。

  2. 等待机器人初始化完成后，打开 DobotStudio Pro ， DobotStudio Pro 启动后会自动搜索同网段的机器人。

  3. 等待搜索到机器人后，选择要连接的机器人，然后点击 连接。

  4. 连接到机器人后会右侧界面会自动弹出控制界面，可以点击右侧边栏的 控制收回控制界面，然后点击主界面上的 菜单 -> 设置弹出设置界面。

  5. 点击 通讯设置然后修改 IP 地址和子网掩码，修改完成后点击 应用。然后关闭 DobotStudio Pro ，重启机器人控制器便可以完成设置。

     – “IP 地址”设置为：192.168.37.100


CR7
192.168.5.1
未连接
连接
CR7
CR7
192.168.5.1
未连接
连接
虚拟控制器 CR5
127.0.0.1
未连接
连接
搜索设备
手动添加


DobotStudio Pro
设置...
语言
帮助
检查更新...
关于...
欢迎来到 DobotStu
- 全新的用户界面
- 易于使用和用户友好
- 将来将与更多Dobot产品兼容
VERSION: 2.7.1-x86-64-stable.2023
如果您有任何建议，请发送电子邮件至pm@dobot.cc。

– "子网掩码"设置为：255.255.255.0

重启控制器后如果想要在连接机器人，需要设置电脑的 IP 地址和机器人在同一网段。

导入工程

在越疆的脚本编程中，脚本文件的名字是无法被修改，并且在运行程序时如果脚本中有需要执行的代码，那么脚本程序将单独在一个线程中启动运行。为了更好的区分工控机主控程序和机械臂主控程序，同时又不使脚本文件过于庞大，我们需要先创建两个 DobotStudio Pro 工程，分别时工控机主控工程和机械臂主控工程。

- 工控机主控工程

1. 打开 DobotStudio Pro 并和机器人连接成功后，点击主界面上的 脚本编程图标，然后界面会跳转到脚本编程界面。

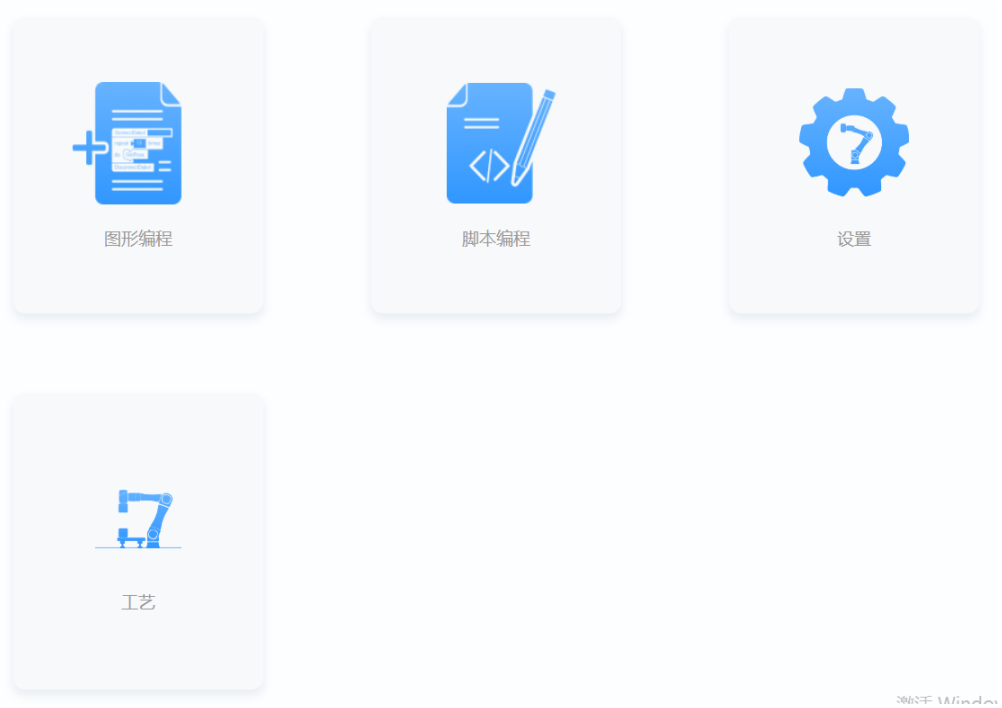
2. 点击 文件 -> 新建，选择 空白工程并点击 确定。

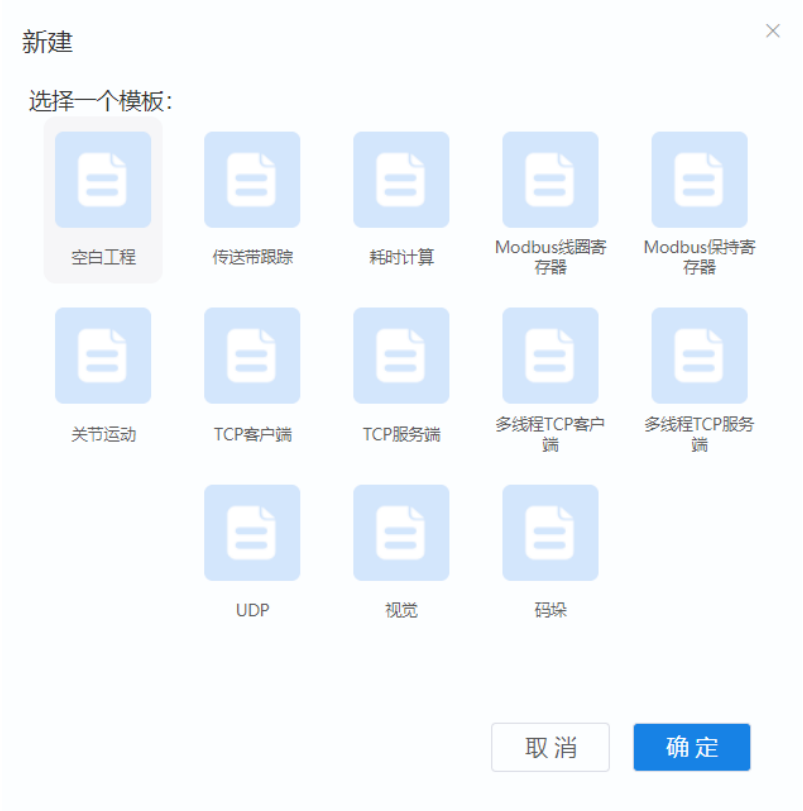
3. 编辑界面选中 `src0.lua`，并删掉里面已有的内容。

4. 通过文本编辑工具打开 Dobot 驱动文件中的 motion_src0.lua ，并拷贝其中所有内容，复制到 `src0.lua` 中。

   复制 motion_src0.lua 里代码的时候，可以先通过快捷键 (ctrl+A) 全选所有的代码，然后再复制 (ctrl+C)，这样可以保证所有的代码没有被遗漏。

   motion_src0.lua 代码中的 C_MOTION_SERVER_IP 和 C_MOTION_PORT 表示的是工控机的 IP 地址和端口号，如果工控机 IP 地址或者端口变更请修改 C_MOTION_SERVER_IP 和 C_MOTION_PORT

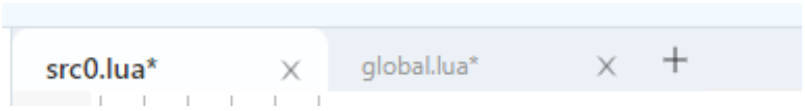

图形编程
脚本编程
设置
工艺


新建
选择一个模板:
空白工程
传送带跟踪
耗时计算
Modbus线圈寄存器
Modbus保持寄存器
关节运动
TCP客户端
TCP服务端
多线程TCP客户端
多线程TCP服务端
UDP
视觉
码垛
取消
确定

```
src0.lua*   global.lua*
                reply = xyzSetJointsMoveJ()
            elseif cmd == C_CMD_SET_CARTESIAN_MOVEL then
                reply = xyzSetCartMoveL()
            elseif cmd == C_CMD_SET_CARTESIAN_MOVEJ then
                reply = xyzSetCartMoveJ()
            elseif cmd == C_CMD_GET_ANALOG_INPUT then
                reply = xyzGetAnalogInput()
            elseif cmd == C_CMD_GET_JOINTS then
                reply = xyzGetJoints()
            elseif cmd == C_CMD_GET_CARTESIAN then
                reply = xyzGetCartesian()
            elseif cmd == C_CMD_MOVEJ_SEQUENCE then
                reply = xyzMoveJSequence()
            elseif cmd == C_CMD_MOVEL_SEQUENCE then
                reply = xyzMoveLSequence()
            elseif cmd == C_CMD_MOVEL_DO then
                reply = xyzMoveLDO()
            else
                reply = C_INVALID_CMD
            end
                reply = tostring(cmd)..C_DELIMITER..reply..C_DELIMITER..C_TERMINATOR
        end
        xyzSendString(reply)
    end
  end
end

-- This is main function
xyzMotion()
```

5. 点击编辑界面上的 + 按钮新增一个脚本，新增的脚本会自动命名为 `src1.lua`，选中 `src1.lua`，并删掉里面的已有内容。



6. 通过文本编辑工具打开 Dobot 驱动文件中的 status_src0.lua，并拷贝其中所有的内容，并复制到 `src1.lua` 中。

   复制 status_src0.lua 里代码的时候，可以先通过快捷键 (ctrl+A) 全选所有的代码，然后再复制 (ctrl+C)，这样可以保证所有的代码没有被遗漏。

   status_src0.lua 代码中 C_STATUS_SERVER_IP 和 C_STATUS_PORT 表示机械臂的 IP 地址和端口号，如果机械臂 IP 地址或者端口号变更请修改 C_STATUS_SERVER_IP 和 C_STATUS_PORT

7. 点击 保存，在弹出的对话框中输入工程名 "xyz_motion"，然后点击 确定保存工程。

至此工控机主控的工程便创建完成并保存到了 dobot 控制器中。

- 机械臂主控工程

  1. 打开 DobotStudio Pro 并和机器人连接成功后，点击主界面上的 脚本编程图标，然后界面会跳转到脚本编程界面。

src0.lua* × global.lua* × src1.lua* × +

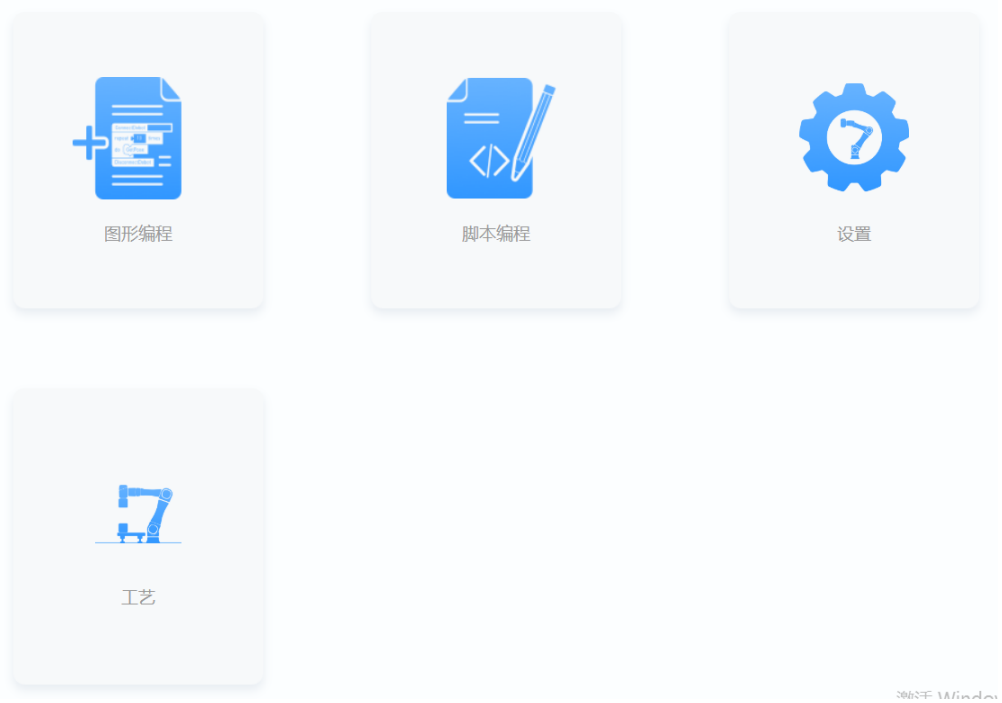
```lua
end

function xyzGetDIString()
    local di_str = ""
    for i = 1, 32 do
        local signal = DI(i)
        if signal == ON then
            di_str = di_str .. "1" .. C_DELIMITER
        else
            di_str = di_str .. "0" .. C_DELIMITER
        end
    end
    return di_str
end

function xyzStatusSend()
    xyzStatusConnect()
    while (true) do
        local reply = ""
        local joint_string = xyzGetJointString()
        local pose_string = xyzGetPoseString()
        local di_string = xyzGetDIString()
        local reply = C_CMD_STATUS..C_DELIMITER..C_JOINT_FLAG..C_DELIMITER..jo
        xyzSendString(reply)
        Sleep(50)     --50ms
    end
    xyzStatusClose()
end


-- This is the main function
xyzStatusSend()
```

如果此时是刚创建完工控机主控工程，可以直接跳转到第二步。



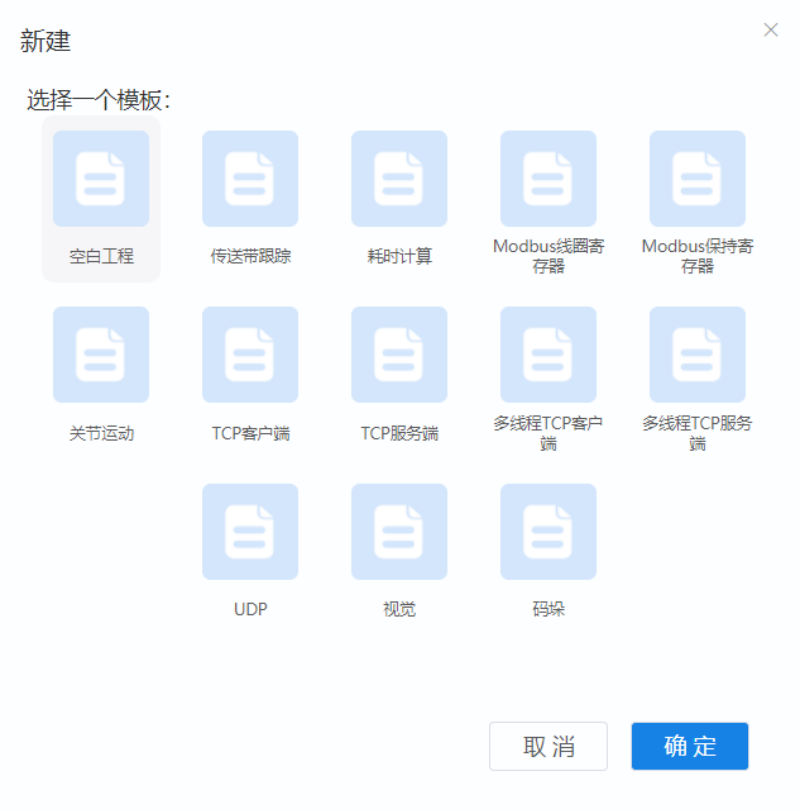
2. 点击 文件 -> 新建，选择 空白工程并点击 确定。

3. 编辑界面选中 `global.lua`，并删掉里面已有的内容。
4. 通过文本编辑工具打开 Dobot 驱动文件中的 global.lua ，并拷贝其中所有的内容，并复制到 `global.lua` 中。

   复制 global.lua 里代码的时候，可以先通过快捷键 (ctrl+A) 全选所有的代码，然后再复制 (ctrl+C)，这样可以保证所有的代码没有被遗漏
5. 编辑界面选中 `src0.lua`，并删掉里面已有的内容。
6. 根据项目需求通过文本编辑工具打开 dobot 程序文件夹下的模板程序，并拷贝其中的所有内容，并复制到 `src1.lua` 中。这里以坐标运动基础应用模板 (xyz_CartMove_Basic.lua) 为例

7. 点击 保存，在弹出的对话框中输入工程名 "xyz_master"，然后点击 确定保存工程。

至此机械臂主控的工程便创建完成并保存到了 dobot 控制器中。



运行程序

通讯说明

工控机和 Dobot 机械臂的通讯方式为 TCP/IP, 工控机作为 TCP/IP 服务端，机械臂作为 TCP/IP 客户端。

启动程序

工控机主控设置

1. DobotStudio Pro 和机器人连接成功后，选择主界面上 最近工程中的“xyz_motion”

2. 待工程加载完成跳转到脚本编程界面后，点击 开始按钮，开始运行程序

src0.lua* global.lua*

```
491         local wp = {}
492
493         while n < num_of_waypoints do
494             table.insert(wp_type, xyzGetNum(7 * n + 5, data_vec))
495             local tmp = {}
496             for i = 1, 6, 1
497             do
498                 table.insert(tmp, xyzGetNum(7 * n + 5 + i, data_vec))
499             end
500             table.insert(wp, tmp)
501             n = n + 1
502         end
503
504         return num_of_waypoints, wp_type, wp
505 end
506
507 function xyzExecuteTraj(num_of_waypoints, type_of_waypoints, waypoints)
508     for i = 1, num_of_waypoints, 1 do
509         local err = xyzHandleWayPoint(waypoints[i], type_of_waypoints[i])
510         if err ~= 0 then
511             return err
512         end
513     end
514
515     return 0
516 end
```

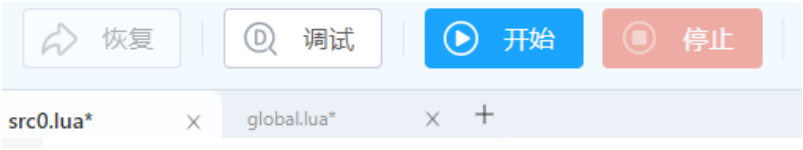
src0.lua* global.lua*

```
93                  Pick = {user=0,tool=1,coordinate=grasp_pose}
94                  SpeedS(V_TCP_SPEED)
95                  AccelS(V_TCP_ACC)
96                  Move(Pick)
97                  Sync()
98                  DO(1, ON)
99              end
100
101             do
102                 SpeedS(V_TCP_SPEED)
103                 AccelS(V_TCP_ACC)
104                 Move(V_PLACE_POSE)
105                 Sync()
106                 DO(1, OFF)
107             end
108
109         ::Continue::
110     end
111
112     ::err_exit:: do
113         print("Error_Exit. Please Check the error code")
114         return
115     end
116 end
117
118 xyzMasterConnect()
119 xyzMain()
120 xyzMasterClose()
121
```

机械臂主控设置

1. DobotStudio Pro 和机器人连接成功后，选择主界面上 最近工程中的“xyz_master”

2. 待工程加载完成跳转到脚本编程界面后，点击 开始按钮，开始运行程序



API 说明

Dobot 工控机主控支持的 API

指令号	命令	支持情况
100	获取机械臂程序版本号	支持
101	发送速度数据	支持
102	发送加速度数据	支持
103	发送圆滑过渡参数	支持
104	发送工具坐标系 (TCP) 参数	支持
105	设置数字量输出	支持
106	SetJointsMovej	支持
107	SetCartMovel	支持
108	SetJoinsMovel	支持
109	SetCartMovej	支持
110	MovejSequence	支持
111	MovelSequence	支持
112	SetJointsMovejDo	支持
113	SetCartMovelDo	支持
114	SetJointsMovelDo	不支持
115	SetJointsMovejGroupDo	不支持
116	SetCartMovelGroupDo	不支持
117	SetJointsMovelGroupDo	不支持
118	MovelUntil	不支持
119	获取数字量输入状态	支持
120	获取数字量输出状态	不支持
121	获取模拟量输入口数值	支持
122	获取机械臂当前角度	支持
123	获取机械臂当前位姿	支持
200	机械臂后台发送状态	支持

Dobot 机械臂主控支持的 API

xyzMasterConnect()

连接到工控机服务器

xyzMasterClose()

断开与工控机服务器的连接

xyzSwitchApp(*app_name*)

切换应用

参数 **app_name**(*string*) –应用名称

返回 error_code

返回类型 number

xyzSwitchFlow(*flow_name*)

切换流图

参数 **flow_name**(*string*) –流图名称

返回 error_code

返回类型 number

xyzSwitchItem(*ws_id*, *obj_name*)

切换工件

参数

- **ws_id**(*number*) –工作空间 id
- **obj_name**(*string*) –工件名称

返回 error_code

返回类型 number

xyzSwitchTool(*tool_name*)

切换工具

参数 **tool_name**(*string*) –工具名称

返回 error_code

返回类型 number

xyzReqCapImg(*vision_service_id*)

请求拍照

参数 **vision_service_id**(*number*) –需要进行拍照的视觉服务 id

返回 error_code

返回类型 number

Return token 获取拍照结果的凭据

xyzGetCapImg(*token*)

获取拍照结果

参数 **token**(*number*) –请求拍照时返回的 token

返回 error_code

返回类型 number

xyzCapImg(*vision_service_id*)

拍照

参数 **vision_service_id**(*number*) –需要进行拍照操作的工作空间 id

返回 error_code

返回类型 number

xyzReqGraspPose(*ws_id*)

请求抓取位姿

参数 **ws_id**(*number*) –需要获取抓取位姿的工作空间 id

返回 error_code

返回类型 number

Return token 获取抓取位姿结果的凭据

xyzGetGraspPose(*token*)

获取抓取位姿

参数 **token** (*number*) –请求抓取目标位姿时返回的 token

返回 error_code

返回类型 number

返回 pose

返回类型 table

返回 pose_num

返回类型 number

返回 pipeline_num

返回类型 number

返回 register_num

返回类型 number

xyzReqObjPose(*ws_id*)

请求物体位姿

参数 **ws_id** (*number*) –需要获取物体位姿的工作空间 id

返回 error_code

返回类型 number

Return token 获取物体位姿结果的凭据

xyzGetObjPose(*token*)

获取物体位姿

参数 **token** (*num*) –请求物体位姿时得到的 token

返回 error_code

返回类型 number

返回 pose

返回类型 table

返回 pose_num

返回类型 number

返回 pose_type

返回类型 number

xyzResetTask()

重置视觉

返回 error_code

返回类型 number

xyzSendCurrentJoints()

发送当前关节位姿

返回 error_code

返回类型 number

xyzSendCurrentCartPose()

发送当前笛卡尔空间位姿

返回 error_code

返回类型 number

xyzReqPick()

请求 pick 动作规划

返回 error_code

返回类型 number

xyzReqPlace()

请求 place 动作规划

返回 error_code

返回类型 number

xyzReqPickPlace(*ws_id*)

请求 pick 和 place 规划

参数 **ws_id** (*number*) –需要请求 pick 和 place 规划的工作空间 id

返回 error_code

返回类型 number

xyzGetPickin(*ws_id*)

获取取料入框轨迹

参数 **ws_id** (*number*) –需要获取 pick in 轨迹的工作空间 id

返回 error_code

返回类型 number

返回 pipeline_num

返回类型 number

返回 register_num

返回类型 number

返回 wp_num

返回类型 number

返回 wp_type

返回类型 array

返回 wp

返回类型 2d-array

xyzGetPickout(*ws_id*)

获取取料出框轨迹

参数 ws_id(*number*) –需要获取 pick out 轨迹的工作空间 id

返回 error_code

返回类型 number

返回 pipeline_num

返回类型 number

返回 register_num

返回类型 number

返回 wp_num

返回类型 number

返回 wp_type

返回类型 array

返回 wp

返回类型 2d-array

xyzGetPlacein(*ws_id*)

获取放料入框轨迹

参数 ws_id(*number*) –需要获取 place in 轨迹的工作空间 id

返回 error_code

返回类型 number

返回 pipeline_num

返回类型 number

返回 register_num

返回类型 number

返回 wp_num

返回类型 number

返回 wp_type

返回类型 array

返回 wp

返回类型 2d-array

xyzGetPlaceout(*ws_id*)

获取放料出框轨迹 (注：该函数不需要传入参数，仅更新全局变量)

参数 ws_id(*number*) –需要获取 place out 轨迹的工作空间 id

返回 error_code

返回类型 number

返回 pipeline_num

返回类型 number

返回 register_num

返回类型 number

返回 wp_num

返回类型 number

返回 wp_type

返回类型 array

返回 wp

返回类型 2d-array

xyzSwitchStrat(*strat_name*)

请求切换策略

参数 **strat_name**(*string*)–策略名称

返回 error_code

返回类型 number

xyzUpdateTotePose()

料箱重定位

参数 **V_POSE**–料箱位姿 (全局变量)

返回 error_code

返回类型 number

返回 pose

返回类型 table

xyzUpdateObjPoseOnHand()

工件在上手的二次定位

返回 error_code

返回类型 number

xyzUpdateObjPoseToHand()

工件不在手上的二次定位，获取二次抓取的轨迹

返回 error_code

返回类型 number

返回 pipeline_num

返回类型 number

返回 register_num

返回类型 number

返回 wp_num

返回类型 number

返回 wp_type

返回类型 array

返回 wp

返回类型 2d-array

xyzGetObjPoseType()

获取工件姿态类型

返回 error_code

返回类型 number

返回 pose_type

返回类型 number

xyzResetPalletStatus()

重置工业码垛状态

返回 error_code

返回类型 number

xyzExecuteTraj(*num_of_waypoints*, *type_of_waypoints*, *waypoints*)

执行段轨迹

参数

- **num_of_waypoints** (*number*) –轨迹点的个数
- **type_of_waypoints** (*array*) –轨迹点类型数组
- **waypoints[]** (*2d-array*) –轨迹点数组

返回 error_code

返回类型 number

xyzCalculateGraspPose(*ws_id*)

计算抓取位姿

参数 **ws_id** (*number*) –工作空间 id

返回 error_code

返回类型 number

返回 pose

返回类型 table

返回 pose_num

返回类型 number

返回 pipeline_num

返回类型 number

返回 register_num

返回类型 number

FUNC num xyzCalculateObjectPose(ws_id)

计算物体位姿

参数 **ws_id** (*number*) –工作空间 id

返回 error_code

返回类型 number

返回 pose

返回类型 table

返回 obj_num

返回类型 number

返回 pose_type

返回类型 number

案例/模板说明

机械臂主控程序说明

以下为机械臂主控的四种应用形式的模板代码，注意对工控机返回的 err_code 进行判断。

座标移动基础模板

```
--------------------------------CartMove Basic Template--------------------------------
↪-----

V_HOME_POSE = {user=0,tool=1,coordinate={240, 560, 90, -153, -14, 16}, joint={80, -50,
↪ -100, 90, 90, -30}}
V_PLACE_POSE = {user=0,tool=1,coordinate={-230, 540, -120, 175, 30, 0}}
V_SCAN_POSE = {user=0,tool=1,coordinate={-100, 650, 240, -180, 0, 0}}

function xyzMain()
-- S1: 初始化
local err_code = 0
local return_token = 0
local ws_id = 0
local flow_name = "cart_basic.t"
local item_codename = "item1"

--IO operation
DO(1, OFF)

-- 切换任务流图 ，默认被注释掉，如果需求可以取消注释
-- do
--   err_code = xyzSwitchFlow(flow_name)
--   xyzCheckErrorCode(err_code)
-- end

-- S3: 切换成当前工件
do
    err_code = xyzSwitchItem(ws_id, item_codename)
    xyzCheckErrorCode(err_code)
end

-- S4: 运动到 home 位
-- 首先移动到 home 点。注意需要先定义 V_HOME_POSE 的关节角度值
-- 或者用使用其他点来替代 V_HOME_POSE
do
    --Speed(V_J_SPEED)
```

(下页继续)

(续上页)

```
    --Accel(V_J_ACC)
    --MoveJ(V_HOME_POSE)
    --Sync()

    SpeedS(V_TCP_SPEED)
    AccelS(V_TCP_ACC)
    Move(V_HOME_POSE)
    Sync()
end

-- 如果是眼在机械臂上的应用形式，请把 eye_on_hand 赋值为 true
eye_on_hand = false

while(true)
do
    -- S5: 眼在机械臂上
    if eye_on_hand then
    -- S6: 运动到拍照位姿
    -- 注意：需要先定义拍照位姿，
    -- 可以直接修改 V_SCAN_POSE 或者使用其他点位来替换
    --走到眼在手上的拍照点，并发送当前拍照点位姿给上位机
    SpeedS(V_TCP_SPEED)
    AccelS(V_TCP_ACC)
    Move(V_SCAN_POSE)
    Sync()

    -- S7: 发送拍照位姿
    err_code = xyzSendCurrentCartPose()
    xyzCheckErrorCode(err_code)
    end

    -- S8: 请求抓取点位
    --请求计算抓取点（自带拍照功能）
    err_code, return_token = xyzReqGraspPose(ws_id)
    xyzCheckErrorCode(err_code)

    local grasp_pose = {}
    local grasp_pose_num = 0
    local pipeline_num = 0
    local register_num = 0

    -- S9: 获取抓取点位
    err_code, grasp_pose, grasp_pose_num, pipeline_num, register_num =␣
↪xyzGetGraspPose(return_token)
    xyzCheckErrorCode(err_code)

    if grasp_pose_num < 1 then
        --当前无可抓取工件，进行下一次拍照请求
        print("No grasp_pose, continue requesting")
        Sleep(5000) --5S
        goto Continue
    end

    -- S10: 运动到抓取点并抓取工件
    -- 注意：可能需要添加其他路径点作为过渡
    do
```

(下页继续)

(续上页)

```
    Pick = {user=0,tool=1,coordinate=grasp_pose}
    SpeedS(V_TCP_SPEED)
    AccelS(V_TCP_ACC)
    -- 移动到预抓取点，偏移值可以根据实际需求修改
    Move(RP(Pick, {0, 0, 100}))
    -- 运动到抓取点
    Move(Pick)
    Sync()
    DO(1, ON)
    -- 移回预抓取点，偏移值可以根据实际需求修改
    Move(RP(Pick, {0, 0, 100}))
    end

    -- S11: 运动到放置点并放置工件
    -- 注意: 可能需要添加其他路径点作为过渡
    -- 注意: 首先需要定义放置点位姿,
    -- 可以直接修改 V_PLACE_POSE 或者使用其他点来替换
    do
    SpeedS(V_TCP_SPEED)
    AccelS(V_TCP_ACC)
    Move(V_PLACE_POSE)
    Sync()
    DO(1, OFF)
    end
end
end


-- S2: 连接到工控机
xyzMasterConnect()
xyzMain()
xyzMasterClose()
```

座标移动二次定位模板:

```
------------------------------CartMove Reposition Template--------------------------
↪----------

V_HOME_POSE = {user=0,tool=1,coordinate={240, 560, 90, -153, -14, 16}, joint={80, -50,
↪ -100, 90, 90, -30}}
V_PLACE_POSE = {user=0,tool=1,coordinate={-230, 540, -120, 175, 30, 0}}
V_BOARD_PLACE_POSE = {user=0,tool=1,coordinate={-100, 650, 240, -180, 0, 0}}

function xyzMain()
-- S1: 初始化
local err_code = 0
local return_token = 0

local flow_name = "cart_repo.t"
local item_codename = "item1"

local ws_id = 0   --相机1
local ws_id_2 = 1 --相机2
```

(下页继续)

(续上页)

```
--IO operation
DO(1,OFF)

-- 切换任务流图 ，默认被注释掉，如果需求可以取消注释
-- do
--   err_code = xyzSwitchFlow(flow_name)
--   xyzCheckErrorCode(err_code)
-- end

-- S3: 机械臂外的相机切换成识别当前工件
do
    err_code = xyzSwitchItem(ws_id, item_codename)
    xyzCheckErrorCode(err_code)
end

--IO operation
DO(1,OFF)

-- S4: 运动到 home 位
-- 首先移动到 home 点。注意需要先定义 V_HOME_POSE 的关节角度值
-- 或者用使用其他点来替代 V_HOME_POSE
do
    --Speed(V_J_SPEED)
    --Accel(V_J_ACC)
    --MoveJ(V_HOME_POSE)
    --Sync()

    SpeedS(V_TCP_SPEED)
    AccelS(V_TCP_ACC)
    Move(V_HOME_POSE)
    Sync()
end

while(true)
do
    local rough_grasp_pose = {}
    local rough_grasp_pose_num = 0
    local rough_pipeline_num = 0
    local rough_register_num = 0

    -- S5: 请求工件的抓取位姿
    err_code, return_token = xyzReqGraspPose(ws_id)
    xyzCheckErrorCode(err_code)

    -- S6: 获取工件的抓取位姿
    --相机1获取工件粗定位抓取点
    err_code, rough_grasp_pose, rough_grasp_pose_num, rough_pipeline_num, rough_
↪register_num = xyzGetGraspPose(return_token)
    xyzCheckErrorCode(err_code)

    -- S7: 处理是否识别到工件
    --无工件，抓隔板
    if rough_grasp_pose_num < 1 then
        local board_grasp_pose = {}
        local board_grasp_pose_num = 0
        local board_pipeline_num = 0
```

(下页继续)

(续上页)

```
        local board_register_num = 0

        -- S15: 没有识别到工件，机械臂外的相机切换识别隔板
        --相机1拍隔板
        err_code = xyzSwitchItem(ws_id, "board")
        xyzCheckErrorCode(err_code)

        -- S16: 请求隔板抓取位姿
        err_code, return_token = xyzReqGraspPose(ws_id)
        xyzCheckErrorCode(err_code)

        -- S17: 获取隔板的抓取位姿
        --相机1获取隔板抓取点
        err_code, board_grasp_pose, board_grasp_pose_num, board_pipeline_num, board_
↪register_num = xyzGetGraspPose(return_token)
        xyzCheckErrorCode(err_code)
        if board_grasp_pose_num < 1 then
        -- no board
        --无隔板，此时也可以换料框
        goto err_exit
        end

        -- S18: 运动到隔板抓取位姿，并抓取隔板
        -- 注意: 可能需要添加其他路径点作为过渡
        --抓隔板
        local P = {user=0,tool=1,coordinate=board_grasp_pose}
        SpeedS(V_TCP_SPEED)
        AccelS(V_TCP_ACC)
        -- 移动到隔板预抓取点，偏移值可以根据实际需求修改
        Move(RP(P, {0, 0, 100}))
        -- move to board grasp pose
        Move(P)
        Sync()
        DO(1,ON)
        -- 移回隔板预抓取点，偏移值可以根据实际需求修改
        Move(RP(P, {0, 0, 100}))

        -- S19: 运动到隔板的放置位姿
        -- 注意: 可能需要添加其他路径点作为过渡
        --放置隔板
        SpeedS(V_TCP_SPEED)
        AccelS(V_TCP_ACC)
        Move(V_BOARD_PLACE_POSE)
        Sync()
        DO(1,OFF)

        -- S20: 机械臂外的相机切换成识别当前工件
        err_code = xyzSwitchItem(ws_id, item_codename)
        xyzCheckErrorCode(err_code)

    else --有工件，二次定位
        -- S8: 运动到拍照位姿
        -- 注意: 需要实现定义拍照位，或者根据机械臂外相机提供的抓取点位来确定，
        -- 可以直接修改 rough_grasp_pose 或者使用其他点位替换
        --走到工件粗定位点
        local P = {user=0,tool=1,coordinate=rough_grasp_pose}
```

(下页继续)

(续上页)

```
        SpeedS(V_TCP_SPEED)
        AccelS(V_TCP_ACC)
        Move(P)
        Sync()

        -- S9: 发送拍照位姿
        --发送当前眼在手上的拍照点位姿给上位机
        err_code = xyzSendCurrentCartPose()
        xyzCheckErrorCode(err_code)

        -- S10: 机械臂上的相机切换成识别当前工件
        --相机2对工件进行精定位拍照
        err_code = xyzSwitchItem(ws_id_2, item_codename)
        xyzCheckErrorCode(err_code)

        -- S11: 请求机械臂上的相机获取工件抓取位姿
        err_code, return_token = xyzReqGraspPose(ws_id_2)
        xyzCheckErrorCode(err_code)

        local fine_grasp_pose = {}
        local fine_grasp_pose_num = 0
        local fine_pipeline_num = 0
        local fine_register_num = 0

        -- S12: 获取工件的抓取位姿
        --获取工件精定位抓取点
        err_code, fine_grasp_pose, fine_grasp_pose_num, fine_pipeline_num, fine_
↪register_num = xyzGetGraspPose(return_token)
        xyzCheckErrorCode(err_code)

        if fine_grasp_pose_num < 1 then
        -- no object
        goto err_exit
        end

        -- S13: 运动到工件的抓取位姿并进行抓取
        -- 注意: 可能需要添加其他路径点作为过渡
        --抓工件
        local P = {user=0,tool=1,coordinate=fine_grasp_pose}
        SpeedS(V_TCP_SPEED)
        AccelS(V_TCP_ACC)
        -- 移动到工件预抓取点, 偏移值可以根据实际需求修改
        Move(RP(P, {0, 0, 100}))
        -- move to grasp pose
        Move(P)
        Sync()
        DO(2, ON)
        -- 移回工件预抓取点, 偏移值可以根据实际需求修改
        Move(RP(P, {0, 0, 100}))

        -- S14: 运动到工件的放置位姿
        -- 注意: 可能需要添加其他路径点作为过渡
        --放置工件
        SpeedS(V_TCP_SPEED)
        AccelS(V_TCP_ACC)
        Move(V_PLACE_POSE)
```

(下页继续)

(续上页)

```
        Sync()

        DO(2, OFF)
    end
end

::err_exit:: do
    print("Error_Exit. Please Check")
    return
end
end

-- S2: 连接到工控机
xyzMasterConnect()
xyzMain()
xyzMasterClose()
```

轨迹移动同步模板:

```
--------------------------------TrajMove Sync Template---------------------------------
↪----

-- 注意: 使用 Sync() 函数可以使程序等待机械臂完成运动指令
-- 注意: 使用 Sync() 函数可以使程序等待机械臂完成运动指令
-- 注意: 使用 Sync() 函数可以使程序等待机械臂完成运动指令

function xyzMain()
    -- S1: 初始化
    local err_code = 0
    local ws_id = 0
    local flow_name = "traj_sync.t"
    local item_codename = "item1"

    --IO operation
    DO(1,OFF)

    -- 切换任务流图 ，默认被注释掉，如果需求可以取消注释
    -- do
    --     err_code = xyzSwitchFlow(flow_name)
    --     xyzCheckErrorCode(err_code)
    -- end

    -- S3: 切换成当前工件
    do
        err_code = xyzSwitchItem(ws_id, item_codename)
        xyzCheckErrorCode(err_code)
    end

    -- S4: 运动到 home 位
    -- 首先移动到 home 点。注意需要先定义 home 的关节角度值
    -- 或者用使用其他点来替代 home
    do
    --local home = {joint={80, -50, -100, 90, 90, -30}}
    --Speed(V_J_SPEED)
```

(下页继续)

(续上页)

```
    --Accel(V_J_ACC)
    --MoveJ(home)
    --Sync()

    local home = {coordinate={240, 560, 90, -153, -14, 16}}
    SpeedS(V_TCP_SPEED)
    AccelS(V_TCP_ACC)
    Move(home)
    Sync()
    end

    while(true)
    do
        local pipeline_num = 0
        local register_num = 0
        local wp_num = 0
        local wp_type = {}
        local wp = {}

        -- S5: 请求抓取和放置规划
        err_code = xyzReqPickPlace(ws_id)
        xyzCheckErrorCode(err_code)

        -- S6: 获取 pick-in 轨迹
        err_code, pipeline_num, register_num, wp_num, wp_type, wp = xyzGetPickin(ws_
↪id)
        xyzCheckErrorCode(err_code)

        -- S7: 判断是否识别到工件
        --if tote cleared, exit
        if wp_num == 0 then
            print("tote clear")
            goto clear_tore_exit
        end

        -- S8: 执行 pick-in 轨迹
        if xyzExecuteTraj(wp_num, wp_type, wp) ~= 0 then
            print("Execute pick in error")
            goto err_exit
        end
        Sync()
        --IO operation
        DO(1, ON)

        -- S9: 获取 pick-out 轨迹
        err_code, pipeline_num, register_num, wp_num, wp_type, wp = xyzGetPickout(ws_
↪id)
        xyzCheckErrorCode(err_code)

        -- S10: 执行 pick-out 轨迹
        if xyzExecuteTraj(wp_num, wp_type, wp) ~= 0 then
            print("Execute pick out error")
            goto err_exit
        end

        -- S11: 获取 place-in 轨迹
```

(下页继续)

(续上页)

```
        err_code, pipeline_num, register_num, wp_num, wp_type, wp = xyzGetPlacein(ws_
↪id)
        xyzCheckErrorCode(err_code)

        -- S12: 执行 place-in 轨迹
        if xyzExecuteTraj(wp_num, wp_type, wp) ~= 0 then
            print("Execute place in error")
            goto err_exit
        end
        Sync()
        --IO operation
        DO(1, OFF)

        -- S13: 获取 place-out 轨迹
        err_code, pipeline_num, register_num, wp_num, wp_type, wp = xyzGetPlaceout(ws_
↪id)
        xyzCheckErrorCode(err_code)

        -- S14: 执行 place-out 轨迹
        if xyzExecuteTraj(wp_num, wp_type, wp) ~= 0 then
            print("Execute place out error")
            goto err_exit
        end
    end

    -- 处理其他异常情况
    ::err_exit:: do
        print("Error_Exit. Please Check the error code")
        return
    end

    ::clear_tore_exit:: do
        print("work done: tote has been cleared")
        return
    end

end

-- S2: 连接到工控机
xyzMasterConnect()
xyzMain()
xyzMasterClose()
```

轨迹移动异步模板:

```
--------------------------------TrajMove ASync Template--------------------------------
↪-----

-- 注意: 使用 Sync() 函数可以使程序等待机械臂完成运动指令
-- 注意: 使用 Sync() 函数可以使程序等待机械臂完成运动指令
-- 注意: 使用 Sync() 函数可以使程序等待机械臂完成运动指令

function xyzMain()
    -- S1: 初始化
```

(下页继续)

(续上页)

```
    local err_code = 0
    local ws_id = 0
    local flow_name = "traj_async.t"
    local item_codename = "item1"

    --IO operation
    DO(1,OFF)

    -- 切换任务流图 ，默认被注释掉，如果需求可以取消注释
    -- do
    --   err_code = xyzSwitchFlow(flow_name)
    --   xyzCheckErrorCode(err_code)
    -- end

    -- S3: 切换成当前工件
    do
    err_code = xyzSwitchItem(ws_id, item_codename)
    xyzCheckErrorCode(err_code)
    end



    -- S4: 运动到 home 位
    -- 首先移动到 home 点。注意需要先定义 home 的关节角度值
    -- 或者用使用其他点来替代 home
    do
    --local home = {joint={80, -50, -100, 90, 90, -30}}
    --Speed(V_J_SPEED)
    --Accel(V_J_ACC)
    --MoveJ(home)
    --Sync()

    local home = {coordinate={240, 560, 90, -153, -14, 16}}
    SpeedS(V_TCP_SPEED)
    AccelS(V_TCP_ACC)
    Move(home)
    Sync()
    end

    -- S5: 请求抓取和放置规划
    err_code = xyzReqPickPlace(ws_id)
    xyzCheckErrorCode(err_code)

while(true)
    do
    local pipeline_num = 0
    local register_num = 0
    local wp_num = 0
    local wp_type = {}
    local wp = {}

    -- S6: 获取 pick-in 轨迹
    err_code, pipeline_num, register_num, wp_num, wp_type, wp = xyzGetPickin(ws_id)
    xyzCheckErrorCode(err_code)

    -- S7: 判断是否识别到工件
```

(下页继续)

(续上页)

```
--if tote cleared, exit
if wp_num == 0 then
    print("tote clear")
    goto clear_tore_exit
end

-- S8: 执行 pick-in 轨迹
if xyzExecuteTraj(wp_num, wp_type, wp) ~= 0 then
    print("Execute pick in error")
    goto err_exit
end
Sync()
--IO operation
DO(1, ON)

-- S9: 获取 pick-out 轨迹
err_code, pipeline_num, register_num, wp_num, wp_type, wp = xyzGetPickout(ws_id)
xyzCheckErrorCode(err_code)

-- S10: 执行 pick-out 轨迹
if xyzExecuteTraj(wp_num, wp_type, wp) ~= 0 then
    print("Execute pick out error")
    goto err_exit
end
Sync()

-- S11: 请求下一次的抓取和放置规划
-- request next pick and place plan in advance
err_code = xyzReqPickPlace(ws_id)
xyzCheckErrorCode(err_code)

-- S12: 获取本次 place-in 轨迹
err_code, pipeline_num, register_num, wp_num, wp_type, wp = xyzGetPlacein(ws_id)
xyzCheckErrorCode(err_code)

-- S13: 执行本次 place-in 轨迹
if xyzExecuteTraj(wp_num, wp_type, wp) ~= 0 then
    print("Execute place in error")
    goto err_exit
end
Sync()
--IO operation
DO(1, OFF)

-- S14: 获取本次 place-out 轨迹
err_code, pipeline_num, register_num, wp_num, wp_type, wp = xyzGetPlaceout(ws_id)
xyzCheckErrorCode(err_code)

-- S15: 执行本次 place-out 轨迹
if xyzExecuteTraj(wp_num, wp_type, wp) ~= 0 then
    print("Execute place out error")
    goto err_exit
end
end

-- 处理其他异常情况
```

(下页继续)

(续上页)

```
    ::err_exit:: do
    print("Error_Exit. Please Check the error code")
    return
    end

    ::clear_tore_exit:: do
    print("work done: tote has been cleared")
    return
    end
end

-- S2: 连接到工控机
xyzMasterConnect()
xyzMain()
xyzMasterClose()
```

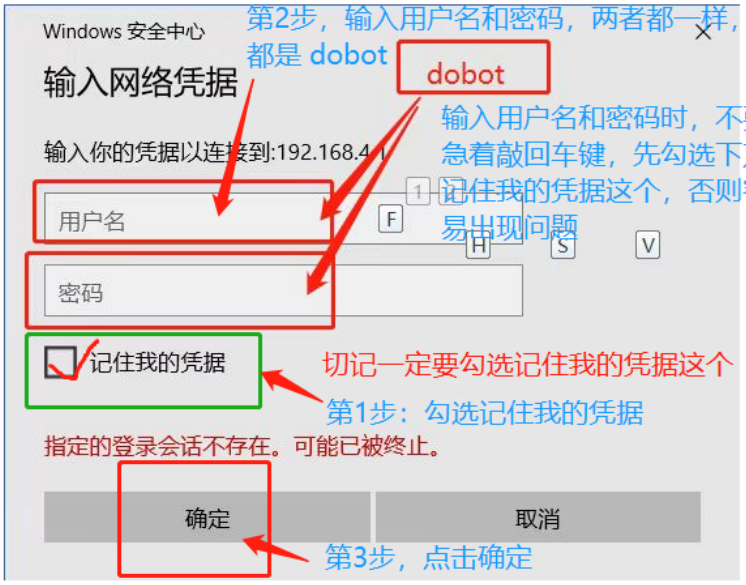
常见问题

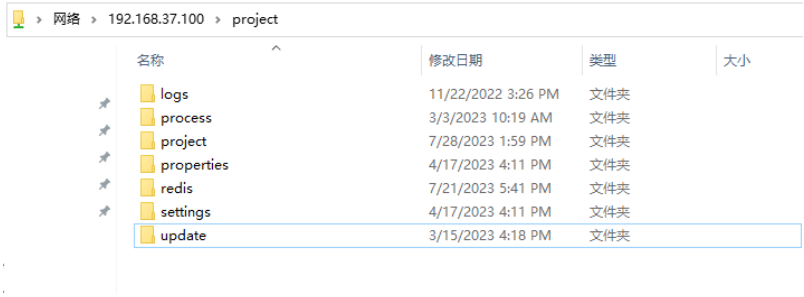
附录

备份 Dobot 控制器

1. 准备一台 Windows 系统的电脑，修改电脑的 IP 地址使其和机器人控制器在同一网段。
   - 网线连接（默认）：192.168.5.1 ，如有更改机械臂默认的 IP ，请输入所改后的 IP 。
   - WIFI 连接（默认）：192.168.1.6。
2. 同时按住键盘上的 Windows+R 键，在弹出的“运行”对话框中，输入两次反斜线和机械臂 IP 地址，然后点击“确定”按钮，即可看到一个 project 的共享文件夹。

   如果点击“确认”之后弹出如下对话框，只需要输入用户名和密码，然后勾选“记住我的凭据”，最后点击对话框上的“确定”即可。
   - 用户名：dobot
   - 密码：dobot
3. 打开的 project 共享文件夹中的所有文件即为所要备份的文件。在本地电脑新建一个文件夹，然后拷贝所有的文件到本地新建的文件夹即可完成备份。


Windows 安全中心
输入网络凭据
输入你的凭据以连接到:192.168.4
用户名
密码
记住我的凭据
指定的登录会话不存在。可能已被终止。
确定
取消
第2步，输入用户名和密码，两者都一样，都是 dobot
dobot
输入用户名和密码时，不要急着敲回车键，先勾选下方记住我的凭据这个，否则容易出现问题
切记一定要勾选记住我的凭据这个
第1步：勾选记住我的凭据
第3步，点击确定


网络 › 192.168.37.100 › project
名称
修改日期
类型
大小
logs 11/22/2022 3:26 PM 文件夹
process 3/3/2023 10:19 AM 文件夹
project 7/28/2023 1:59 PM 文件夹
properties 4/17/2023 4:11 PM 文件夹
redis 7/21/2023 5:41 PM 文件夹
settings 4/17/2023 4:11 PM 文件夹
update 3/15/2023 4:18 PM 文件夹

请注意不要剪切 project 共享文件夹里的文件，不然系统将无法正常启动

4. 项目工程文件存放在 project 共享文件夹下的 project 目录下，如果只是拷贝工程文件，可以到该目录下把工程文件拷贝出来。

   在 DobotStudio Pro 上只能看到最近使用的工程，不能看到机器人控制器里的工程，如果想要加载机器人控制器里已有的工程可以使用该方法先把 project 目录下的工程剪切到本地电脑，然后通过 DobotStudio Pro 导入剪切到本地电脑的工程。另外如果保存工程时 DobotStudio Pro 提示有同名的工程，那么也可以到 project 目录把同名的工程删除或者剪切到其他位置。

14.2.5 Efort

此处介绍安装 Efort 机械臂驱动的相关事项。

驱动版本和使用要求

支持的机械臂类型

Efort 六轴机械臂

控制器型号

EC2-S6

机械臂需要开通的功能

自带 Socket 功能，无需开通。

安装驱动

Efort 机械臂驱动文件列表

需要导入机械臂的程序是 `XYZ_MASTER.XPL` `XYZ_MOTION.XPL` `XYZ_STATUS.XPL`。XPL 文件是以 XML 格式记录的程序，直接打开查看会难以阅读，需要通过埃夫特提供的 PC Editor 软件 (下载地址：https://www.yuque.com/efort/guide/suggko) 加载查看。

`XYZ_MASTER` 文件夹下的代码是机械臂主控示例代码，不能导入到示教器中，仅供查阅参考使用，可以通过 PC Editor 或者 vscode 查看。

├── XYZ_MASTER **此文件夹下的代码是示例代码，不能直接导入到示教器中，仅供参考使用**
│ ├── Program
│ │ ├── Main.pgm **主程序，选定需要运行的任务**
│ │ ├── Main_example.pgm **机械臂主控 api 调用示例**
│ │ ├── xyzCalculaeGraspPose.pgm
│ │ ├── xyzCalculateObjectPose.pgm
│ │ ├── xyzCapImg.pgm
│ │ ├── xyzCartMoveBasic.pgm **座标移动基础模板**
│ │ ├── xyzCartMoveReposition.pgm **座标移动二次定位模板**

```
│   │   ├── xyzClearUserCommandData.pgm
│   │   ├── xyzConnectSocket.pgm
│   │   ├── xyzExecutePickInTraj.pgm 执行抓取入筐子程序
│   │   ├── xyzExecutePickOutTraj.pgm 执行抓取出筐子程序
│   │   ├── xyzExecuteTraj.pgm 通用执行轨迹函数
│   │   ├── xyzGetCapImg.pgm
│   │   ├── xyzGetGraspPose.pgm
│   │   ├── xyzGetObjPose.pgm
│   │   ├── xyzGetObjPoseType.pgm
│   │   ├── xyzGetPickIn.pgm
│   │   ├── xyzGetPickOut.pgm
│   │   ├── xyzGetPlaceIn.pgm
│   │   ├── xyzGetPlaceOut.pgm
│   │   ├── xyzParseTraj.pgm
│   │   ├── xyzParseUserCommand.pgm
│   │   ├── xyzRecvAndParse.pgm
│   │   ├── xyzReqCapImg.pgm
│   │   ├── xyzReqGraspPose.pgm
│   │   ├── xyzReqObjPose.pgm
│   │   ├── xyzReqPick.pgm
│   │   ├── xyzReqPickPlace.pgm
│   │   ├── xyzReqPlace.pgm
│   │   ├── xyzResetPalletStatus.pgm
│   │   ├── xyzResetTask.pgm
│   │   ├── xyzSendCurrentCartPose.pgm
│   │   ├── xyzSendCurrentExtJoints.pgm
│   │   ├── xyzSendCurrentJoints.pgm
│   │   ├── xyzSwitchApp.pgm
│   │   ├── xyzSwitchFlow.pgm
│   │   ├── xyzSwitchItem.pgm
│   │   ├── xyzSwitchStrat.pgm
│   │   ├── xyzSwitchTool.pgm
│   │   ├── xyzTrajMoveAsync.pgm 轨迹移动异步模板
│   │   ├── xyzTrajMoveSync.pgm 轨迹移动同步模板
│   │   ├── xyzUpdateObjPoseOnHand.pgm
│   │   ├── xyzUpdateObjPoseToHand.pgm
│   │   ├── xyzUpdateTotePose.pgm
│   │   └── xyzUserCommand.pgm
│   ├── Var 每个程序对应的变量文件
│   │   ├── Main.var
│   │   .
│   │   .
│   │   .
│   └── XYZ_MASTER.cfg
└── XYZ_MOTION 工控机主控运动控制程序，示例代码仅供查看
└── XYZ_STATUS 工控机主控机械臂状态发送程序，示例代码仅供查看
└── XYZ_MASTER.XPL 机械臂主控程序
```

└── XYZ_MOTION.XPL **工控机主控运动控制程序**
└── XYZ_STATUS.XPL **工控机主控机械臂状态发送程序**

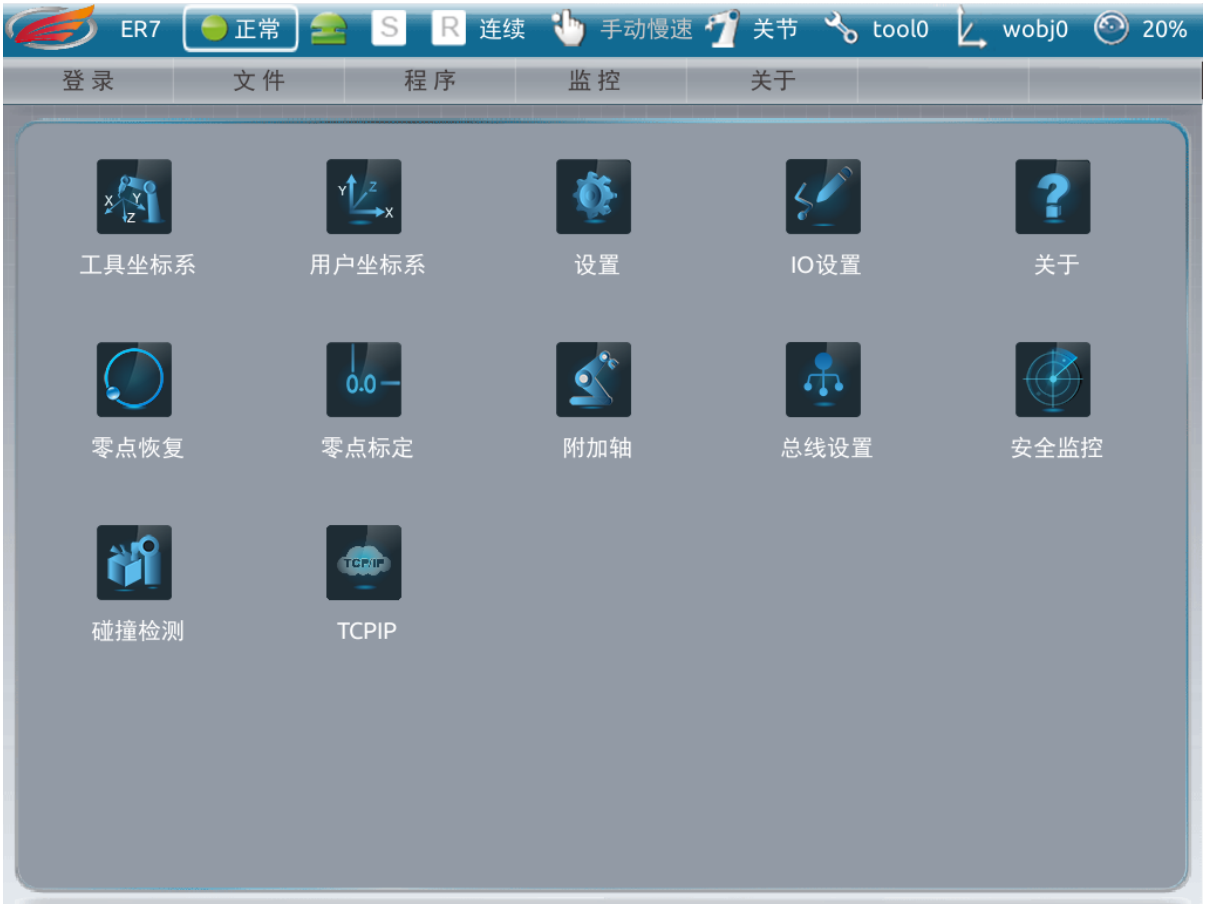
设定机械臂 IP

1. 机械臂开机后到登录界面，输入管理员登录密码然后点击登录

   EFORT 机器人提供操作员、工程师、管理员三个权限等级的账号，默认登录账号为操作者。可以进行账号切换：在密码弹窗中输入对应权限等级的密码，然后点击 登录按钮，即可登录相应账号。各权限等级的密码如下：

   - 管理员：999999
   - 工程师：666666
   - 操作者：111111

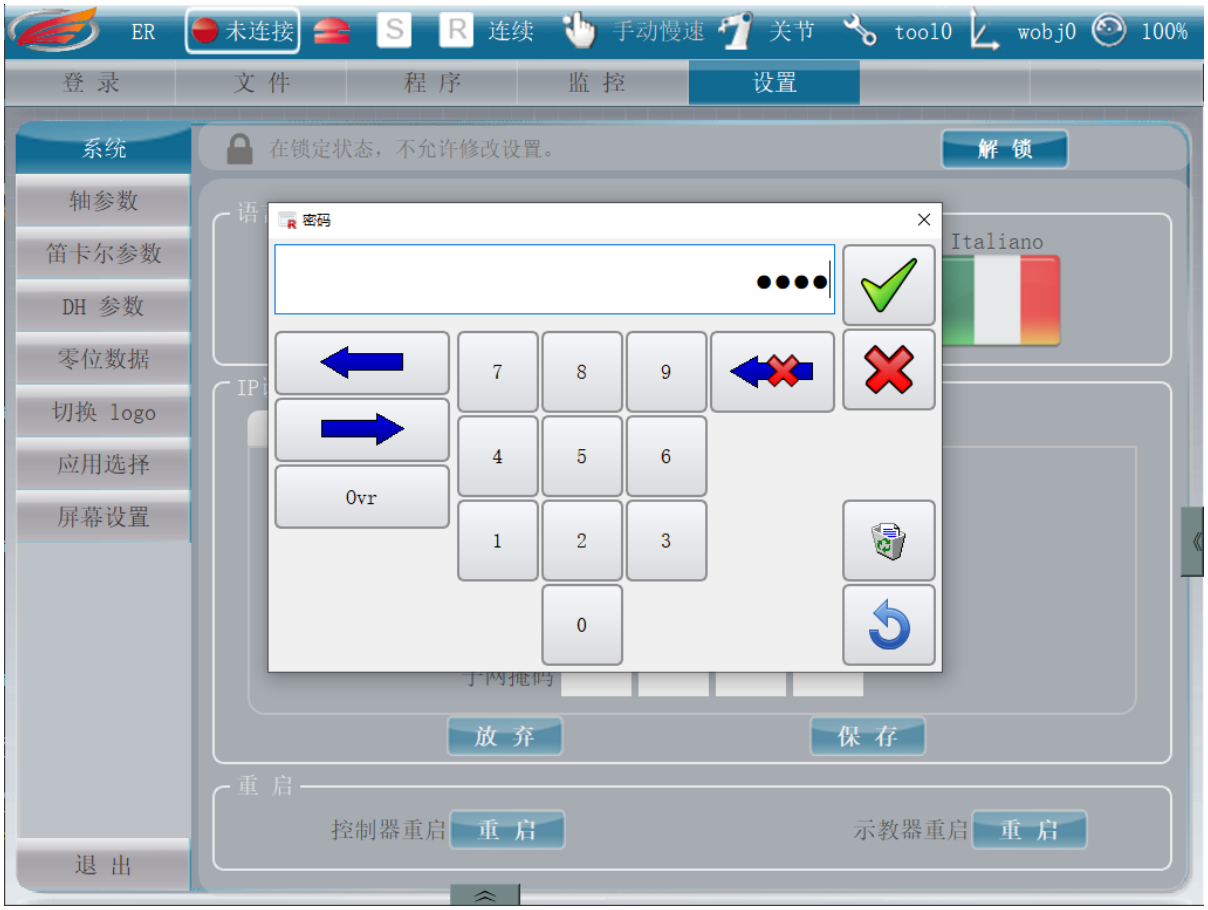
2. 登录后，按下示教器上的 HOME 按键，进入到桌面



3. 在桌面上点击 设置图标，进入设置界面

4. 进入设置界面后点击右上角的 解锁按钮，然后输入密码：1975 ，进行解锁


ER7
正常
S
R
连续
手动慢速
关节
tool0
wobj0
20%
登 录
文 件
程 序
监 控
关于
设置
系统
轴参数
笛卡尔参数
DH 参数
零位数据
切换 logo
应用选择
屏幕设置
退 出
在锁定状态，不允许修改设置。
解 锁
语言
中文
English
Italiano
IP设置
ETH2
ETH3
WAN
使能IP协议
使能TPU IP
ETH3 IP 192 168 37 100
示教器IP 192 168 37 102
网关 192 168 37 1
子网掩码 255 255 255 0
放 弃
保 存
重 启
控制器重启
重 启
示教器重启
重 启


ER
未连接
S
R
连续
手动慢速
关节
tool0
wobj0
100%
登 录
文 件
程 序
监 控
设置
系统
在锁定状态，不允许修改设置。
解 锁
轴参数
笛卡尔参数
DH 参数
零位数据
切换 logo
应用选择
屏幕设置
退 出
密码
Italiano
7
8
9
4
5
6
1
2
3
0
Ovr
放 弃
保 存
重 启
控制器重启
重 启
示教器重启
重 启

5. 设置 ETH3 的 ip
   - 控制器 IP：192.168.37.100
   - 示教器 IP：192.168.37.102
   - 网关：192.168.37.1
   - 子网隐码：255.255.255.0

网络设置完成后重启机械臂生效。

配置 SOCKET 通信

1. 机械臂开机后会到登录界面，输入管理员登录密码然后点击登录
2. 登录后，按下示教器上的 HOME 按键，进入到桌面
3. 在桌面找到 TCPIP 图标，并点击图标进入到 TCPIP 设置界面

   如果桌面没有 TCPIP 图标，则需先进入设置里的 应用选择界面，勾选 TCPIP ，然后桌面上便会有 TCPIP 图标了。
4. 进入 TCPIP ，在 通路选择里通过左右切换选择 #01 然后进行如下设置，设置完成后点击
   - 协议类型: 客户端
   - 服务器 IP : 192.168.37.101 （工控机 IP ）


ER7
正常
S
R
连续
手动慢速
关节
tool0
wobj0
20%
登 录
文 件
程 序
监 控
关于
设置
TCPIP
通信参数
通路选择
#01
协议类型
客户端
套接字号
套接字1
服务器IP
192.168.37.101
端口
10142
超时时间
10000
心跳时间
10000
首字符
!
结束符
!
CR($R)
LF($N)
连接状态
退出
保存
TPU网络调试助手
协议类型
客户端
在线
服务器IP
192.168.1.200
连接
端口
9000
清空接收区
接收区
按十六进制接收
发送区
按十六进制发送
hello world!
发送
清空发送区

图 29: Efort 示教器开启 TCPIP 功能

- 端口: 10142 (10142 是工控机主控所使用的端口号，如果是机械臂主控请修改端口为 11111)
- 超时时间: 1 (1 是工控机主控的超时时间，如果是机械臂主控请修改超时时间为 10000)
- 心跳时间: 10000
- 首字符: !
- 尾字符: !

图 30: Efort 示教器配置机械臂主控 SOCKET

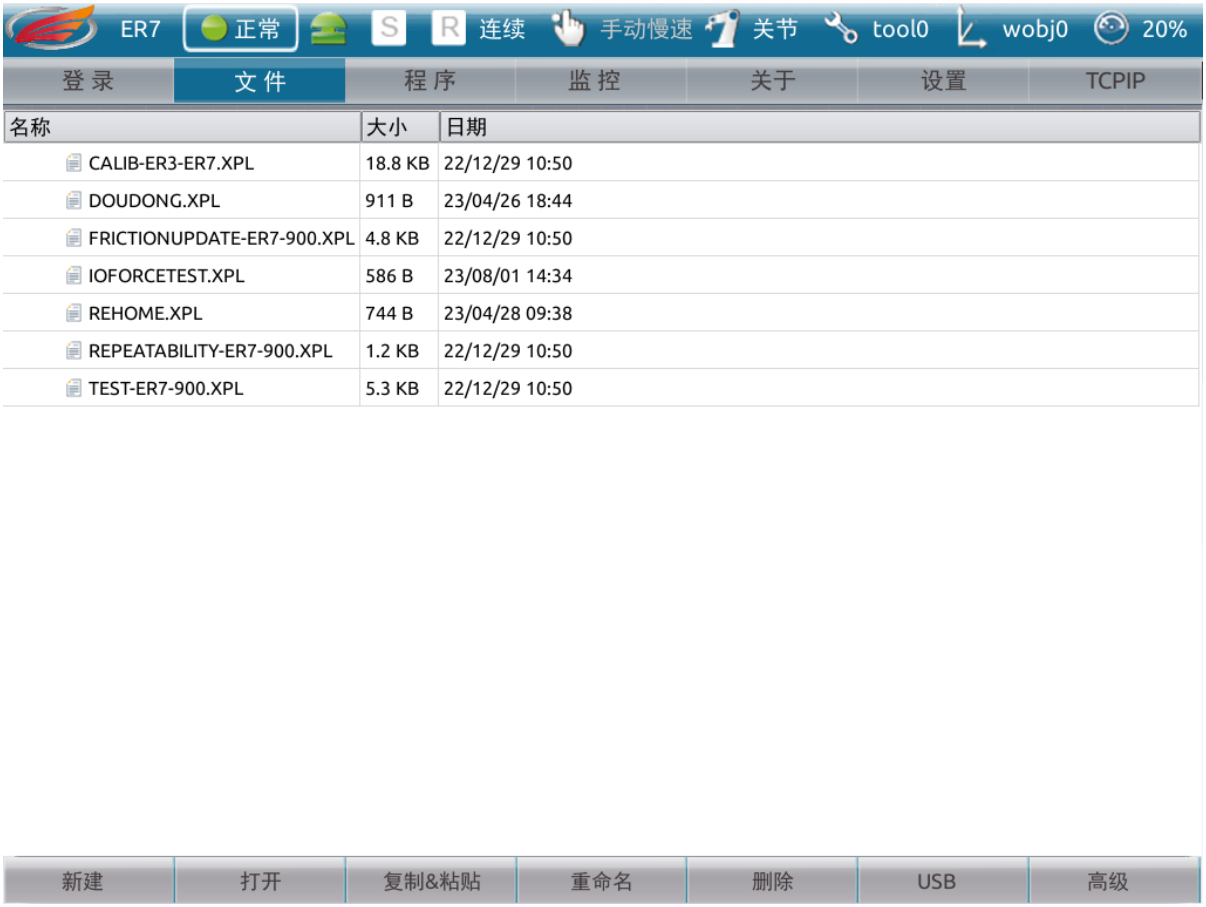
注意：工控机主控和机械臂主控均使用了 #01 以上设置是针对工控机主控的形式，如果是机械臂主控只需把把端口的值修改为 11111，超时时间修改为 10000

5. 在 通路选择里通过左右切换选择 #02 然后进行如下设置
   - 协议类型: 客户端
   - 服务器 IP : 192.168.37.101 （工控机 IP）
   - 端口: 10144
   - 超时时间: 10000
   - 心跳时间: 10000
   - 首字符: !
   - 尾字符: !


ER7
正常
S
R
连续
手动慢速
关节
tool0
wobj0
20%
登 录
文 件
程 序
监 控
关于
设置
TCPIP
通信参数
通路选择
#02
协议类型
客户端
套接字号
套接字2
服务器IP
192.168.37.101
端口
10144
超时时间
10000
心跳时间
10000
首字符
!
结束符
!
CR($R)
LF($N)
连接状态
退出
保存
TPU网络调试助手
协议类型
客户端
在线
服务器IP
192.168.1.200
连接
端口
9000
清空接收区
接收区
按十六进制接收
发送区
按十六进制发送
hello world!
发送
清空发送区

使用 U 盘导入程序

1. 机械臂开机后会到登录界面，输入管理员登录密码然后点击登录
2. 准备一个文件系统为 FAT32 格式的 U 盘，将机械臂代码拷入 U 盘，并将 U 盘插入到示教器 USB 口上
3. 在示教器上，点击 文件标签栏，点击底部 USB -> 从 USB



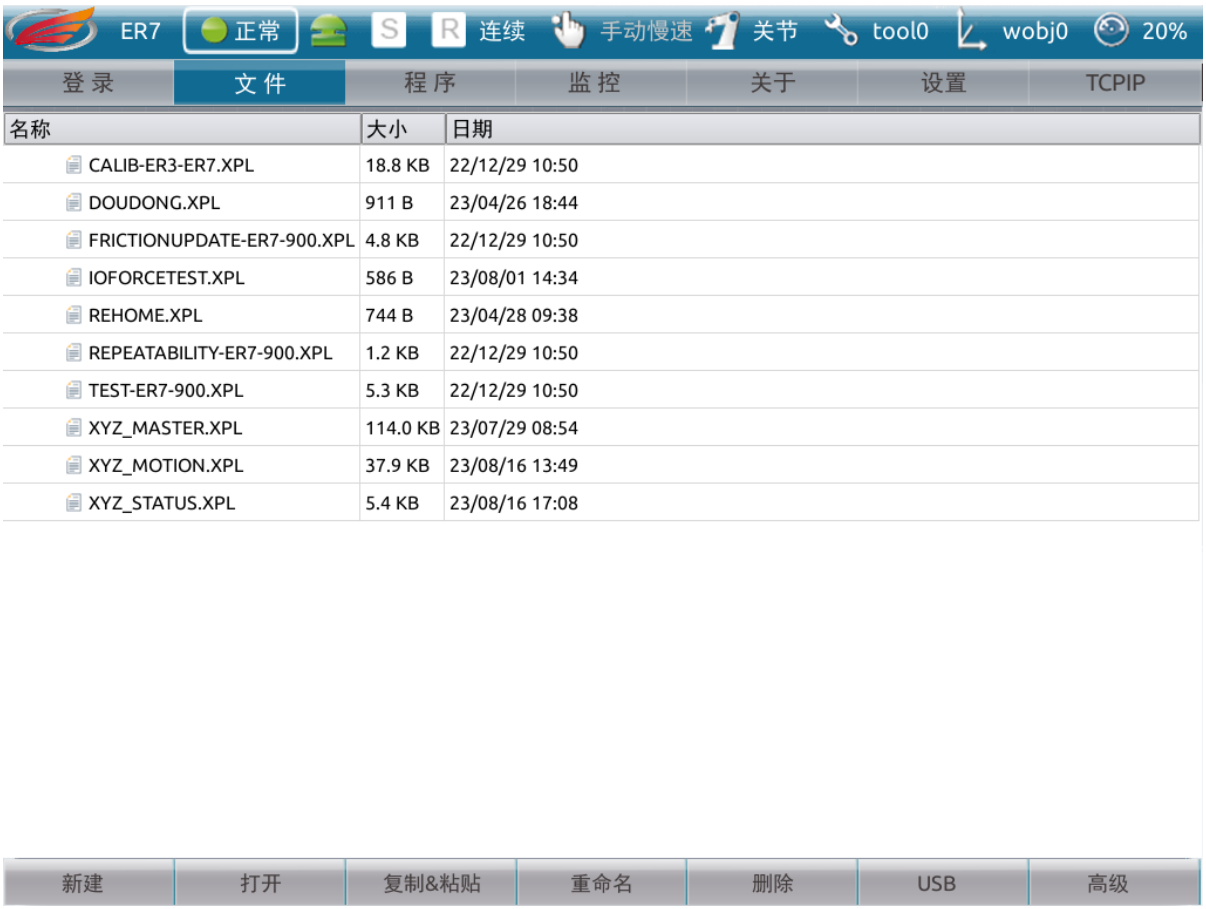
4. 在弹出的界面中，选中需要导入的 XPL 文件，然后点击 导入

   导入时依次选择 XYZ_MASTER.XPL XYZ_MOTION.XPL XYZ_STATUS.XPL 这三个文件进行导入。导入之后的 文件界面如下图所示。

运行程序

通讯说明

工控机和 Efort 机械臂的通讯方式为 TCP/IP , 工控机作为 TCP/IP 的服务端，机械臂作为 TCP/IP 的客户端。


ER7
正常
S
R
连续
手动慢速
关节
tool0
wobj0
20%
登 录
文 件
程 序
监 控
关于
设置
TCPIP
名称
大小
日期
CALIB-ER3-ER7.XPL 18.8 KB 22/12/29 10:50
DOUDONG.XPL 911 B 23/04/26 18:44
FRICTIONUPDATE-ER7-900.XPL 4.8 KB 22/12/29 10:50
IOFORCETEST.XPL 586 B 23/08/01 14:34
REHOME.XPL 744 B 23/04/28 09:38
REPEATABILITY-ER7-900.XPL 1.2 KB 22/12/29 10:50
TEST-ER7-900.XPL 5.3 KB 22/12/29 10:50
XYZ_MASTER.XPL 114.0 KB 23/07/29 08:54
XYZ_MOTION.XPL 37.9 KB 23/08/16 13:49
XYZ_STATUS.XPL 5.4 KB 23/08/16 17:08
新建
打开
复制&粘贴
重命名
删除
USB
高级

运行工控机主控

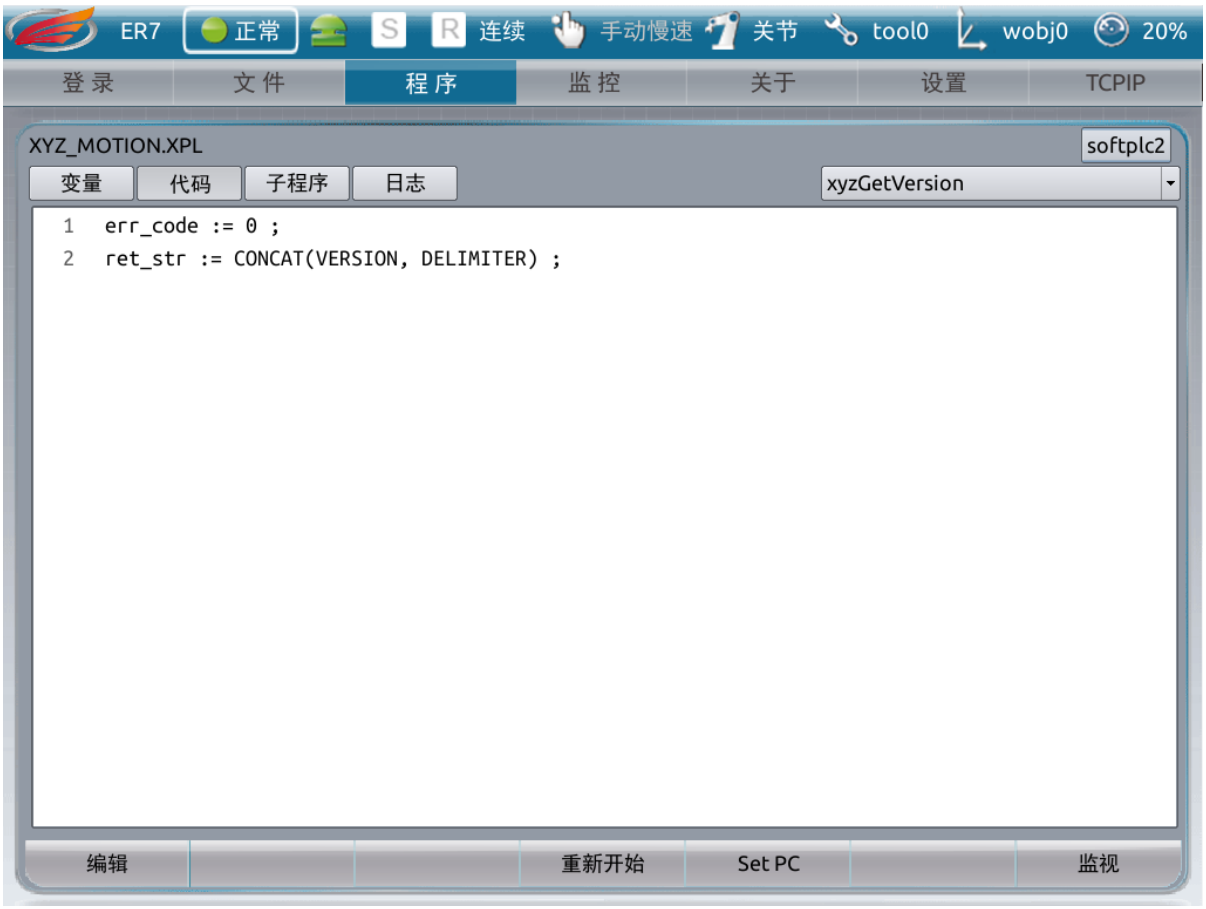
1. 以管理员登录后，点击 程序标签栏，查看当前程序选择的是否是为主程序 robot

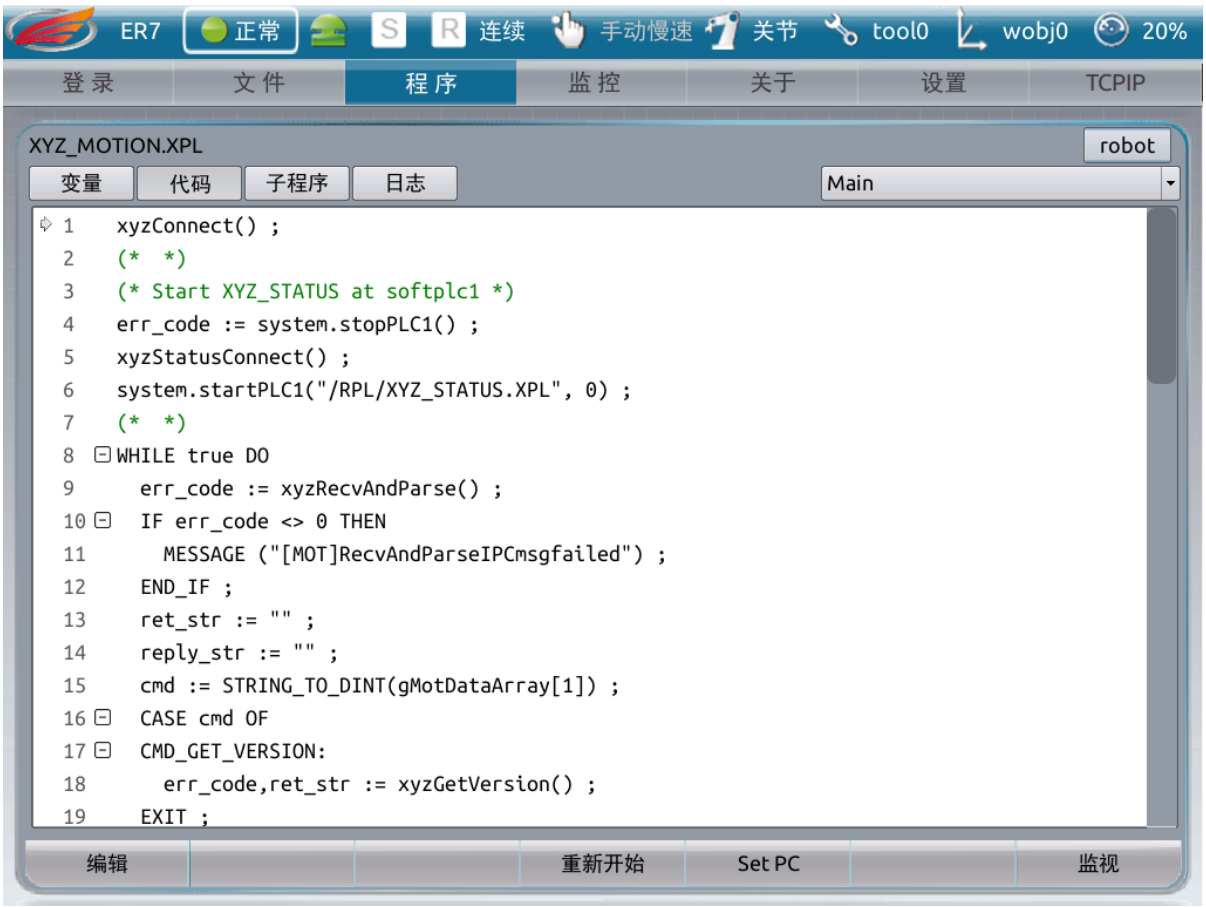
   如果不是主程序，则点击右上角的程序按钮，在弹出的窗口中选择 robot 并点击 确认。下图显示的当前选中为第二个软 PLC 程序，如果要切换到 robot 需要点击 softplc2 然后在弹出的界面中，选中 robot 所在的一栏，然后点击 确认切换换到 robot

2. 在示教器上，点击 文件标签栏，选中 XYZ_MOTION.XPL 然后点击底部的 打开，在弹出的提示界面上点击 是加载 XYZ_MOTION.XPL 程序

   程序加载完成后会自动跳到 程序标签栏

3. 将示教器的旋钮切到 Auto ，在弹出的自动运行界面点击 确定
4. 按下控制柜上绿色闪烁的 SERVO ON 按钮
5. 按下示教器上的 PWR 按键，然后按下示教器上的运行建开始运行程序


ER7
正常
S
R
连续
手动慢速
关节
tool0
wobj0
20%
登 录
文 件
程 序
监 控
关于
设置
TCPIP
XYZ_MOTION.XPL
softplc2
变量
代码
子程序
日志
xyzGetVersion
1 err_code := 0 ;
2 ret_str := CONCAT(VERSION, DELIMITER) ;
编辑
重新开始
Set PC
监视


ER7
正常
S
R
连续
手动慢速
关节
tool0
wobj0
20%
登 录
文 件
程 序
监 控
关于
设置
TCPIP
XYZ_MOTION.XPL
robot
变量
代码
子程序
日志
Main
1 xyzConnect() ;
2 (* *)
3 (* Start XYZ_STATUS at softplc1 *)
4 err_code := system.stopPLC1() ;
5 xyzStatusConnect() ;
6 system.startPLC1("/RPL/XYZ_STATUS.XPL", 0) ;
7 (* *)
8 WHILE true DO
9 err_code := xyzRecvAndParse() ;
10 IF err_code <> 0 THEN
11 MESSAGE ("[MOT]RecvAndParseIPCmsgfailed") ;
12 END_IF ;
13 ret_str := "" ;
14 reply_str := "" ;
15 cmd := STRING_TO_DINT(gMotDataArray[1]) ;
16 CASE cmd OF
17 CMD_GET_VERSION:
18 err_code,ret_str := xyzGetVersion() ;
19 EXIT ;
编辑
重新开始
Set PC
监视

机械臂主控运行

1. 以管理员登录后，点击 程序标签栏，查看当前程序选择的是否是为主程序 robot

   如果不是主程序，则点击右上角的程序按钮，在弹出的窗口中选择 robot 并点击 确认。下图显示的当前选中为第二个软 PLC 程序，如果要切换到 robot 需要点击 softplc2 然后在弹出的界面中，选中 robot 所在的一栏，然后点击 确认切换换到 robot

2. 在示教器上，点击 文件标签栏，选中 XYZ_MASTER.XPL 然后点击底部的 打开，在弹出的提示界面上点击 是加载 XYZ_MASTER.XPL 程序

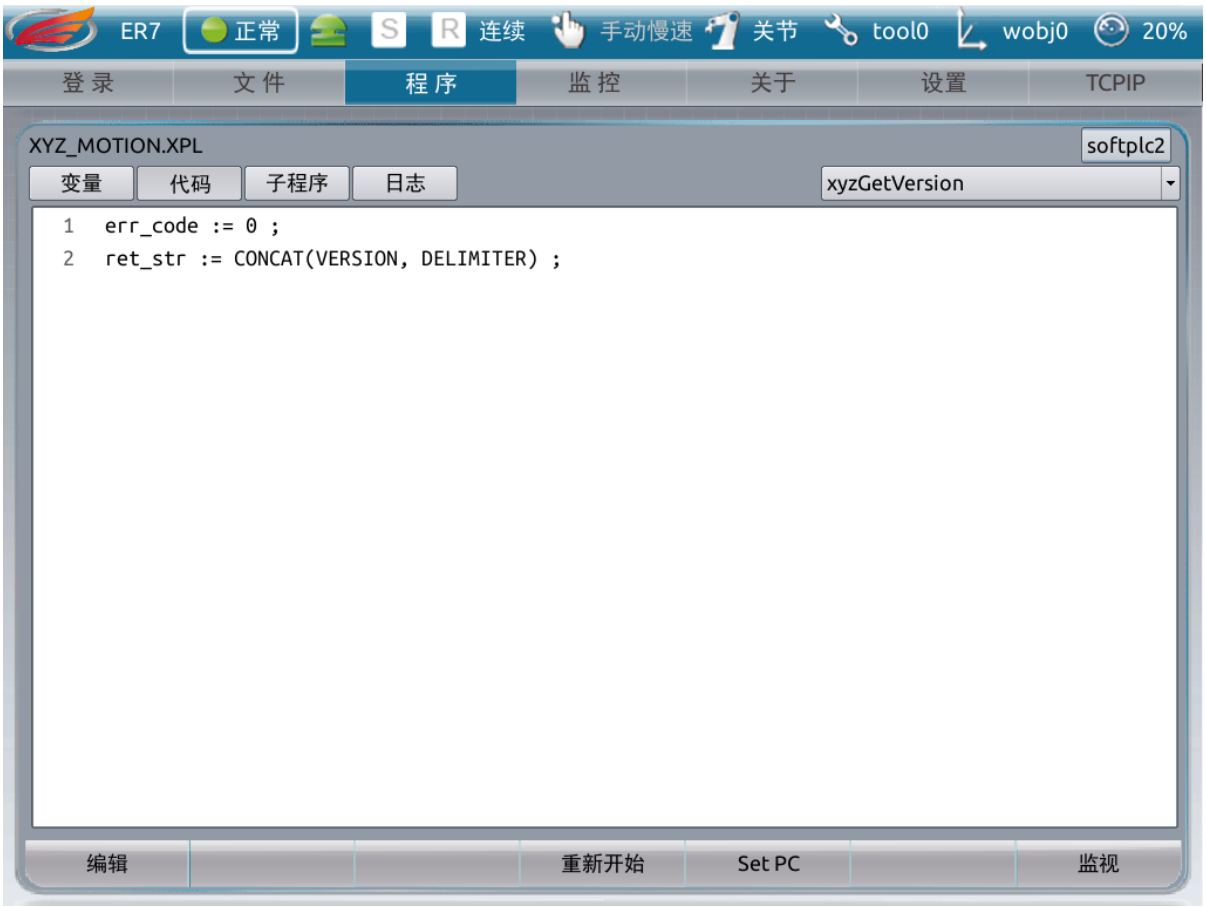
   程序加载完成后会自动跳到 程序标签栏

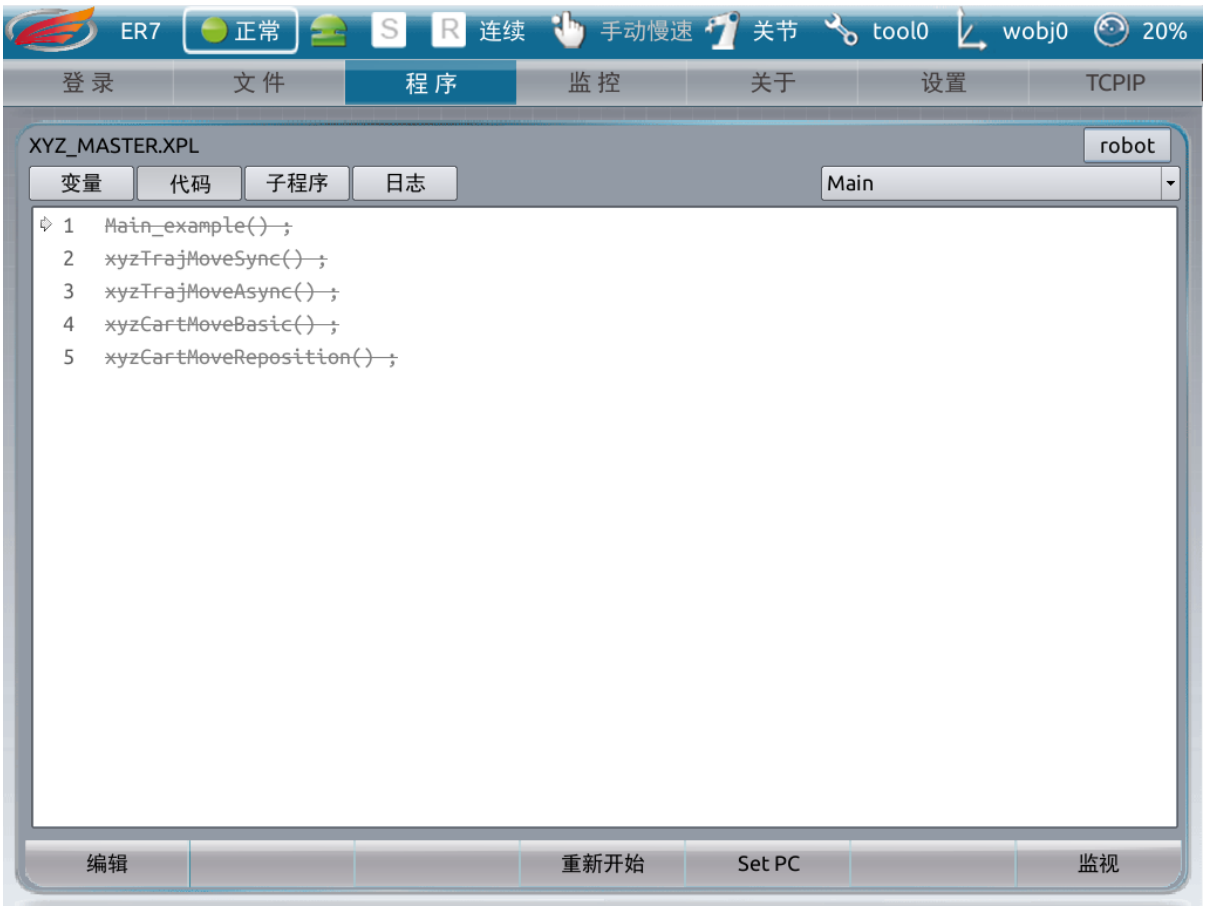
3. 进入 main 函数，根据项目所需的流程以及所提供的模板修改 main 函数，也可以直接修改模板函数，然后在 main 函数中调用修改后的模板函数。

4. 将示教器的旋钮切到 Auto ，在弹出的自动运行界面点击 确定

5. 按下控制柜上绿色闪烁的 SERVO ON 按钮

6. 按下示教器上的 PWR 按键，然后按下示教器上的运行建开始运行程序


ER7
正常
S
R
连续
手动慢速
关节
tool0
wobj0
20%
登 录
文 件
程 序
监 控
关于
设置
TCPIP
XYZ_MOTION.XPL
softplc2
变量
代码
子程序
日志
xyzGetVersion
1 err_code := 0 ;
2 ret_str := CONCAT(VERSION, DELIMITER) ;
编辑
重新开始
Set PC
监视


ER7
正常
S
R
连续
手动慢速
关节
tool0
wobj0
20%
登 录
文 件
程 序
监 控
关于
设置
TCPIP
XYZ_MASTER.XPL
robot
变量
代码
子程序
日志
Main
1 Main_example() ;
2 xyzTrajMoveSync() ;
3 xyzTrajMoveAsync() ;
4 xyzCartMoveBasic() ;
5 xyzCartMoveReposition() ;
编辑
重新开始
Set PC
监视

API 说明

efort 工控机主控支持的 API

指令号	命令	支持情况
100	获取机械臂程序版本号	支持
101	发送速度数据	支持
102	发送加速度数据	支持
103	发送圆滑过渡参数	支持
104	发送工具坐标系 (TCP) 参数	支持
105	设置数字量输出	支持
106	SetJointsMovej	支持
107	SetCartMovel	支持
108	SetJoinsMovel	支持
109	SetCartMovej	支持
110	MovejSequence	支持
111	MovelSequence	支持
112	SetJointsMovejDo	支持
113	SetCartMovelDo	不支持
114	SetJointsMovelDo	不支持
115	SetJointsMovejGroupDo	不支持
116	SetCartMovelGroupDo	不支持
117	SetJointsMovelGroupDo	不支持
118	MovelUntil	不支持
119	获取数字量输入状态	支持
120	获取数字量输出状态	支持
121	获取模拟量输入口数值	不支持
122	获取机械臂当前角度	支持
123	获取机械臂当前位姿	支持
200	机械臂后台发送状态	支持

efort 机械臂主控支持的 API

xyzSwitchApp(*app_name*)

切换应用

参数 **app_name** (*STRING*) –应用名称

返回 err_code 为 0 表示成功

xyzSwitchFlow(*flow_name*)

切换工件

参数 **flow_name** (*STRING*) –流图名称

返回 err_code 为 0 表示成功

xyzSwitchTool(*tool_name*)

切换工具

参数 **tool_name** (*STRING*) –工具名称

返回 err_code 为 0 表示成功

xyzReqCapImg(*vision_service_id*)

请求拍照

参数 vision_service_id –请求拍照的视觉服务 id

返回

err_code 为 0 表示成功

cap_img_token 返回的 token

xyzGetCapImg(*cap_img_token*)

获取拍照结果

参数 cap_img_token (*INT*) –请求拍照时返回的 token

返回 err_code 为 0 表示成功

xyzCapImg(*vision_service_id*)

拍照

参数 vision_service_id –需要进行拍照操作的视觉服务 id

返回 err_code 为 0 表示成功

xyzReqGraspPose(*ws_id*)

请求抓取位姿

参数 ws_id –需要获取抓取点位的工作空间 id

返回

err_code 为 0 表示成功

grasp_pose_token 返回的用于获取目标点位时使用的 token

xyzGetGraspPose(*grasp_pose_token*)

获取抓取位姿

参数 grasp_pose_token (*INT*) –求抓取目标点位时返回的 token

返回

err_code 为 0 表示成功

grasp_pose 抓取位姿

grasp_pose_num 可供抓取的点数量

pipeline_num pipeline 文件 number

register_num 用到的注册文件的注册 number

xyzReqObjPose(*ws_id*)

请求物体位姿

参数 ws_id (*INT*) –需要获取物体位姿的工作空间 id

返回

err_code 为 0 表示成功

obj_token 返回的用于物体位姿识别的 token

xyzGetObjPose(*obj_pose_token*)

获取物体位姿

参数 obj_pose_token(*INT*) –请求物体位姿时得到的 token

返回

err_code 为 0 表示成功

obj_pose 物体位姿

num 物体数量

pose_type 当前返回的物体 pose 类型

xyzResetTask(*ws_id*)

重置任务

参数 ws_id(*INT*) –需要重置的任务 id

返回 err_code 为 0 表示成功

xyzSendCurrentJoints()

发送机械臂当前角度：j1~j6: 机械臂当前的角度信息，如果机械臂轴数小于 6 轴，则需要对应位置补零，依旧发送六个数。

返回 err_code

返回类型 INT

xyzSendCurrentCartPose()

发送机械臂法兰当前位姿

返回 err_code

返回类型 INT

xyzSendCurrentExtJoints()

暂不支持。

发送机械臂当前扩展轴位置：j1~j6: 机械臂当前扩展轴的角度信息，如果扩展轴数不足 6 的, 需要补零后发送六个数。

返回 err_code 为 0 表示成功

xyzReqPick()

请求 pick 动作规划

返回 err_code 为 0 表示成功

xyzReqPlace()

请求 place 动作规划

返回 err_code

返回类型 INT

xyzReqPickPlace(*ws_id*)

请求 pick 和 place 规划

参数 ws_id(*INT*) –需要抓取的工作空间 id

返回 err_code 为 0 表示成功

xyzGetPickin(*ws_id*)

获取取料入框轨迹

参数 ws_id(*INT*)–规划空间 id，默认填 0

返回

err_code 为 0 表示成功

num 轨迹点数量

pipeline_num pipeline 文件 number

register_num 用到的注册文件的注册 number

因函数不支持传出数组，轨迹点位信息这里不做输出。轨迹信息保存在功能块变量（全局变量）**gInCarts**, **gWpType**, **gInJoints** 中。以下轨迹均同理。

xyzGetPickout(*ws_id*)

获取取料出框轨迹

参数 ws_id(*INT*)–规划空间 id，默认填 0

返回

err_code 为 0 表示成功

num 轨迹点数量

pipeline_num pipeline 文件 number

register_num 用到的注册文件的注册 number

xyzGetPickout(*ws_id*)

获取放料入框轨迹

参数 ws_id(*INT*)–规划空间 id，默认填 0

返回

err_code 为 0 表示成功

num 轨迹点数量

pipeline_num pipeline 文件 number

register_num 用到的注册文件的注册 number

xyzGetPlaceout(*ws_id*)

获取放料出框轨迹

参数 ws_id(*INT*)–规划空间 id，默认填 0

返回

err_code 为 0 表示成功

num 轨迹点数量

pipeline_num pipeline 文件 number

register_num 用到的注册文件的注册 number

xyzSwitchStrat(*strat_name*)

请求切换策略

参数 strat_name(*string*)–策略名称

返回 err_code 为 0 表示成功

xyzUpdateTotePose()

料箱重定位

返回

err_code 为 0 表示成功

tote_pose 料箱位姿

xyzUpdateObjPoseOnHand()

工件在上手的二次定位

返回 err_code 为 0 表示成功

xyzUpdateObjPoseToHand()

工件不在手上的二次定位

返回

err_code 为 0 表示成功

num 轨迹点数量

pipeline_num pipeline 文件 number

register_num 用到的注册文件的注册 number

xyzGetObjPoseType()

获取工件姿态类型

返回

err_code 为 0 表示成功

pose_type 工件姿态类型

xyzResetPalletStatus()

重置工业码垛状态

返回 err_code 为 0 表示成功

xyzSwitchItem(*ws_id*, *item_codename*)

切换工件

参数

- **ws_id** (*INT*) –工作空间 id
- **item_codename** (*STRING*) –工件名称

返回 err_code 为 0 表示成功

xyzCalculaeGraspPose(*ws_id*)

计算抓取位姿

参数 **ws_id** (*INT*) –工作空间 id

返回

err_code 为 0 表示成功

grasp_pose 抓取位姿

num 物体数量

pipeline_num pipeline 文件 number

register_num 用到的注册文件的注册 number

xyzCalculateObjectPose(*ws_id*)

参数 ws_id(*INT*) –工作空间 id

返回

err_code 为 0 表示成功

obj_pose 物体位姿

num 物体数量

pose_type 当前返回的物体 pose 类型

xyzClearUserCommandData()

清空用户自定义命令数据 (**使用自定义命令前，必须调用**)

xyzUserCommand()

用户自定义发送数据。使用请按照参照 Main_example.

发送给工控机：

out_str ： 提供 5 个字符串供用户输入

out_ints：提供 10 个整数类型供用户输入

out_floats：提供 10 个浮点数供用户输入

out_cart_pose：cart_pose 点位坐标，如果机器人是欧拉角形式，则只需给 a,b,c 赋值，d 赋值 0 即可

out_joints：关节角坐标值

工控机返回：

in_str：返回 5 个字符串

in_ints：提供 10 个整数类型供用户输入：返回 10 个整数值

in_floats：提供 10 个浮点数供用户输入：返回 10 个浮点数

in_cart_pose：cart_pose 点位坐标，如果机器人是欧拉角形式，则只需给 a：返回一个笛卡尔位姿数据

in_joints：关节角坐标值：返回一个关节坐标

返回 err_code 为 0 表示成功

案例/模板说明

机械臂主控主函数说明

机械臂主控的 api 调用示例在 Main_example 子程序中。

以下为机械臂主控模板代码，注意对工控机返回的 err_code 进行判断。

轨迹移动同步模板

```
CALL: xyzConnectSocket() ; 连接工控机
(* "" *)
flow_name := "traj_sync.t" ;
CALL: err_code := xyzSwitchFlow(flow_name) ; 切换任务为轨迹移动同步
IF err_code <> 0 THEN
    MESSAGE ("Failed") ;
    ENDPROG ;
END_IF ;
(* "" *)
ws_id := 0 ;
item_codename := "item1" ;
CALL: err_code := xyzSwitchItem(ws_id, item_codename) ; 切换工件
IF err_code <> 0 THEN
    MESSAGE ("Failed") ;
    ENDPROG ;
END_IF ;
(* "" *)
io.setDOut(1, false) ; io 初始化
MJOINT (POINTJ(0, 0, 0, 0, -90, 0), v50, fine, tool0) ; 机械臂回到初始位置
(* "" *)
(* "" *)
WHILE true DO
    ws_id := 0 ;
    CALL: err_code := xyzReqPickPlace(ws_id) ; 请求计算注册抓取
    IF err_code <> 0 THEN
        MESSAGE ("Failed") ;
        ENDPROG ;
    END_IF ;
    (* "" *)
    ws_id := 0 ;
    CALL: err_code,num,pipeline_num,register_num := xyzGetPickIn(ws_id) ;␣
↪获取抓取入筐轨迹
    IF err_code <> 0 THEN
        MESSAGE ("Failed") ;
        ENDPROG ;
    END_IF ;
    IF num < 1 THEN
        GOTO clear_tote_exit ; 无可抓取工件，退出
    END_IF ;
    CALL: xyzExecutePickInTraj(num) ;  执行抓取入筐轨迹
    io.setDOut(1, true) ; 抓取工件
    (* "" *)
    ws_id := 0 ;
    CALL: err_code,num,pipeline_num,register_num := xyzGetPickOut(ws_id) ;␣
↪获取抓取出筐轨迹
    IF err_code <> 0 THEN
        MESSAGE ("Failed") ;
        ENDPROG ;
    END_IF ;
    CALL: xyzExecutePickOutTraj(num) ; 执行抓取出筐轨迹
    (* "" *)
    ws_id := 0 ;
    CALL: err_code,num,pipeline_num,register_num := xyzGetPlaceIn(ws_id) ;␣
↪获取放置入筐轨迹
    IF err_code <> 0 THEN
```

(下页继续)

(续上页)

```
        MESSAGE ("Failed") ;
        ENDPROG ;
    END_IF ;
    CALL: xyzExecuteTraj(num) ; 执行放置入筐轨迹
    io.setDOut(1, false) ; 放置工件
    (* "" *)
    ws_id := 0 ;
    CALL: err_code,num,pipeline_num,register_num := xyzGetPlaceOut(ws_id) ;␣
↪获取放置出筐轨迹
    IF err_code <> 0 THEN
        MESSAGE ("Failed") ;
        ENDPROG ;
    END_IF ;
    CALL: xyzExecuteTraj(num) ; 执行放置出筐轨迹
    (* "" *)
END_WHILE ;
LABEL clear_tote_exit :
```

轨迹移动异步模板

```
CALL: xyzConnectSocket() ; 连接工控机
(* "" *)
flow_name := "traj_async.t" ;
CALL: err_code := xyzSwitchFlow(flow_name) ; 切换任务为轨迹移动异步
IF err_code <> 0 THEN
    MESSAGE ("Failed") ;
    ENDPROG ;
END_IF ;
(* "" *)
ws_id := 0 ;
item_codename := "item1" ;
CALL: err_code := xyzSwitchItem(ws_id, item_codename) ; 切换工件
IF err_code <> 0 THEN
    MESSAGE ("Failed") ;
    ENDPROG ;
END_IF ;
(* "" *)
io.setDOut(1, false) ; io 初始化
MJOINT (POINTJ(0, 0, 0, 0, -90, 0), v50, fine, tool0) ; 机械臂回到初始位置
WHILE true DO
    ws_id := 0 ;
    CALL: err_code := xyzReqPickPlace(ws_id) ; 请求计算注册抓取
    IF err_code <> 0 THEN
        MESSAGE ("Failed") ;
        ENDPROG ;
    END_IF ;
    (* "" *)
    ws_id := 0 ;
    CALL: err_code,num,pipeline_num,register_num := xyzGetPickIn(ws_id) ;␣
↪获取抓取入筐轨迹
    IF err_code <> 0 THEN
        MESSAGE ("Failed") ;
        ENDPROG ;
    END_IF ;
```

(下页继续)

(续上页)

```
    IF num < 1 THEN
        GOTO clear_tote_exit ; 无可抓取工件，退出
    END_IF ;
    CALL: xyzExecutePickInTraj(num) ; 执行抓取入筐轨迹
    io.setDOut(1, true) ;
    (* "" *)
    ws_id := 0 ;
    CALL: err_code,num,pipeline_num,register_num := xyzGetPickOut(ws_id) ;␣
↪获取抓取出筐轨迹
    IF err_code <> 0 THEN
        MESSAGE ("Failed") ;
        ENDPROG ;
    END_IF ;
    CALL: xyzExecutePickOutTraj(num) ; 执行抓取出筐轨迹
    (* "" *)
    ws_id := 0 ;
    CALL: err_code := xyzReqPickPlace(ws_id) ; 提前请求计算下一次注册放置
    IF err_code <> 0 THEN
        MESSAGE ("Failed") ;
        ENDPROG ;
    END_IF ;
    (* "" *)
    ws_id := 0 ;
    CALL: err_code,num,pipeline_num,register_num := xyzGetPlaceIn(ws_id) ;␣
↪获取放置入筐轨迹
    IF err_code <> 0 THEN
        MESSAGE ("Failed") ;
        ENDPROG ;
    END_IF ;
    CALL: xyzExecuteTraj(num) ; 执行放置入筐轨迹
    io.setDOut(1, false) ; 放置工件
    (* "" *)
    ws_id := 0 ;
    CALL: err_code,num,pipeline_num,register_num := xyzGetPlaceOut(ws_id) ;␣
↪获取放置出筐轨迹
    IF err_code <> 0 THEN
        MESSAGE ("Failed") ;
        ENDPROG ;
    END_IF ;
    CALL: xyzExecuteTraj(num) ; 执行放置出筐轨迹
END_WHILE ;
LABEL clear_tote_exit :
(* "" *)
```

座标移动基础模板

```
CALL: xyzConnectSocket() ; 连接工控机
(* "" *)
flow_name := "cart_basic.t" ;
CALL: err_code := xyzSwitchFlow(flow_name) ; 切换任务为座标移动基础任务
IF err_code <> 0 THEN
    MESSAGE ("Failed") ;
    ENDPROG ;
END_IF ;
```

(下页继续)

(续上页)

```
(* "" *)
ws_id := 0 ;
item_codename := "item1" ;
CALL: err_code := xyzSwitchItem(ws_id, item_codename) ; 切换工件
IF err_code <> 0 THEN
    MESSAGE ("Failed") ;
    ENDPROG ;
END_IF ;
(* "" *)
io.setDOut(1, false) ; io 初始化
MJOINT (POINTJ(0, 0, 0, 0, -90, 0), v50, fine, tool0) ; 机械臂回到初始位置
(* "" *)
is_eye_in_hand := true ; 是否为眼在手上
WHILE true DO
    IF is_eye_in_hand = true THEN 如果眼在手上
        (* "move to scan pose when eye is in hand" *) 机械臂移动到拍照位置
        MLIN (POINTC(476.45, -0.14, 768.17, 180, 0, 180, CFG1), v500, fine, tool0) ;
        CALL: err_code := xyzSendCurrentCartPose() ; 发送当前位姿
        IF err_code <> 0 THEN
            MESSAGE ("Failed") ;
            ENDPROG ;
        END_IF ;
        (* "" *)
    END_IF ;
    (* "" *)
    ws_id := 0 ;
    CALL: err_code,grasp_pose_token := xyzReqGraspPose(ws_id) ; 请求抓取位姿
    IF err_code <> 0 THEN
        MESSAGE ("Failed") ;
        ENDPROG ;
    END_IF ;
    获取抓取位姿
    CALL: err_code,grasp_pose,num,pipeline_num,register_num := xyzGetGraspPose(grasp_
↪pose_token) ;
    IF err_code <> 0 THEN
        MESSAGE ("Failed") ;
        ENDPROG ;
    END_IF ;
    (* "" *)
    IF num < 1 THEN
        MESSAGE ("no grasp pose, continue requesting next grasp pose") ;␣
↪无可抓取工件，继续请求下一个抓取位姿
        TIMERSTART (timer1, 5) ;
        WAIT (TIMERQ(timer1)) ; 等待 5 秒
        CONTINUE ;
    END_IF ;
    (* "" *)
    MLIN (grasp_pose, v500, fine, tool0) ; 机械臂移动到抓取位姿
    io.setDOut(1, true) ; 抓取工件
    (* "move to place pose" *) 机械臂移动到放置位姿
    MLIN (POINTC(476.45, -0.14, 768.17, 180, 0, 180, CFG1), v500, fine, tool0) ;
    io.setDOut(1, false) ; 放置工件
    (* "" *)
END_WHILE ;
(* "" *)
```

座标移动二次定位模板

```
CALL: xyzConnectSocket() ; 连接工控机
(* "" *)
flow_name := "cart_repo.t" ;
CALL: err_code := xyzSwitchFlow(flow_name) ; 切换任务为座标移动二次定位任务
IF err_code <> 0 THEN
    MESSAGE ("Failed") ;
    ENDPROG ;
END_IF ;
(* "" *)
io.setDOut(1, false) ; io 初始化
io.setDOut(2, false) ;
(* "move to home" *)
MJOINT (POINTJ(0, 0, 0, 0, -90, 0), v50, fine, tool0) ; 机械臂回到初始位置
(* "" *)
(* "camera1 rough position" *) 相机1 粗定位
ws_id := 0 ;
item_codename := "item1" ;
CALL: err_code := xyzSwitchItem(ws_id, item_codename) ; 切换工件
IF err_code <> 0 THEN
    MESSAGE ("Failed") ;
    ENDPROG ;
END_IF ;
(* "" *)
ws_id := 0 ;
CALL: err_code,grasp_pose_token := xyzReqGraspPose(ws_id) ; 请求粗抓取位姿
IF err_code <> 0 THEN
    MESSAGE ("Failed") ;
    ENDPROG ;
END_IF ;
(* "" *)
WHILE true DO
    (* "camera1 get rough position" *) 相机1 获取粗抓取位姿
    CALL: err_code,rough_grasp_pose,rough_grasp_pose_num,rough_pipeline_num,rough_
↪register_num := xyzGetGraspPose(grasp_pose_token) ;
    IF err_code <> 0 THEN
        MESSAGE ("Failed") ;
        ENDPROG ;
    END_IF ;
    (* "" *)
    IF rough_grasp_pose_num < 1 THEN 如果无可抓取工件, 抓隔板
        ws_id := 0 ;
        item_codename := "board" ;
        CALL: err_code := xyzSwitchItem(ws_id, item_codename) ; 切换工件为隔板
        IF err_code <> 0 THEN
            MESSAGE ("Failed") ;
            ENDPROG ;
        END_IF ;
        ws_id := 0 ;
        CALL: err_code,board_grasp_pose_token := xyzReqGraspPose(ws_id) ;␣
↪请求隔板抓取位姿
        IF err_code <> 0 THEN
            MESSAGE ("Failed") ;
            ENDPROG ;
        END_IF ;
        (* "camera1 grasp board" *) 相机1 抓取隔板
```

(下页继续)

(续上页)

```
        CALL: err_code,board_grasp_pose,board_grasp_pose_num,board_pipeline_num,board_
↪register_num := xyzGetGraspPose(board_grasp_pose_token) ;
        IF err_code <> 0 THEN
            MESSAGE ("Failed") ;
            ENDPROG ;
        END_IF ;
        IF board_grasp_pose_num < 1 THEN 如果无可抓取隔板
            (* "no board, you can change tote" *)
            ENDPROG ;
        END_IF ;
        MLIN (board_grasp_pose, v500, fine, tool0) ; 机械臂移动到隔板抓取位姿
        io.setDOut(1, true) ; 抓取隔板
        (* "move to board place pose" *) 机械臂移动到隔板放置位姿
        MLIN (POINTC(476.45, -0.14, 768.17, 180, 0, 180, CFG1), v500, fine, tool0) ;
        io.setDOut(1, false) ;  放置隔板
        (* "camera1 rough position" *) 相机1 粗定位
        ws_id := 0 ;
        item_codename := "item1" ;
        CALL: err_code := xyzSwitchItem(ws_id, item_codename) ; 切换工件
        IF err_code <> 0 THEN
            MESSAGE ("Failed") ;
            ENDPROG ;
        END_IF ;
        ws_id := 0 ;
        CALL: err_code,grasp_pose_token := xyzReqGraspPose(ws_id) ; 请求粗抓取位姿
        IF err_code <> 0 THEN
            MESSAGE ("Failed") ;
            ENDPROG ;
        END_IF ;
        CONTINUE ;
        (* "" *)
    END_IF ;
    (* "item exists, camera2 reposition" *) 有可抓取工件，相机2 二次定位
    (* "move to rough grasp pose" *) 机械臂移动到粗抓取位姿
    MLIN (rough_grasp_pose, v500, fine, tool0) ;
    CALL: err_code := xyzSendCurrentCartPose() ; 发送当前位姿
    IF err_code <> 0 THEN
        MESSAGE ("Failed") ;
        ENDPROG ;
    END_IF ;
    (* "" *)
    ws_id := 1 ;
    item_codename := "item1" ;
    CALL: err_code := xyzSwitchItem(ws_id, item_codename) ; 切换工件
    IF err_code <> 0 THEN
        MESSAGE ("Failed") ;
        ENDPROG ;
    END_IF ;
    ws_id := 1 ;
    CALL: err_code,grasp_pose_token := xyzReqGraspPose(ws_id) ; 请求精确抓取位姿
    IF err_code <> 0 THEN
        MESSAGE ("Failed") ;
        ENDPROG ;
    END_IF ;
    (* "" *) 获取精确抓取位姿
    CALL: err_code,fine_grasp_pose,fine_grasp_pose_num,fine_pipeline_num,fine_
↪register_num := xyzGetGraspPose(grasp_pose_token) ;
```

(下页继续)

(续上页)

```
    IF err_code <> 0 THEN
        MESSAGE ("Failed") ;
        ENDPROG ;
    END_IF ;
    IF fine_grasp_pose_num < 1 THEN 如果无可抓取工件，退出
        ENDPROG ;
    END_IF ;
    MLIN (fine_grasp_pose, v500, fine, tool0) ; 机械臂移动到精确抓取位姿
    io.setDOut(2, true) ; 抓取工件
    (* "move to item place pose" *) 机械臂移动到放置位姿
    MLIN (POINTC(476.45, -0.14, 768.17, 180, 0, 180, CFG1), v500, fine, tool0) ;
    io.setDOut(2, false) ; 放置工件
    (* "" *)
    (* "camera1 rough position" *) 相机1 粗定位
    ws_id := 0 ;
    item_codename := "item1" ;
    CALL: err_code := xyzSwitchItem(ws_id, item_codename) ; 切换工件
    IF err_code <> 0 THEN
        MESSAGE ("Failed") ;
        ENDPROG ;
    END_IF ;
    ws_id := 0 ;
    CALL: err_code,grasp_pose_token := xyzReqGraspPose(ws_id) ; 请求粗抓取位姿
    IF err_code <> 0 THEN
        MESSAGE ("Failed") ;
        ENDPROG ;
    END_IF ;
    (* "" *)
    (* "" *)
END_WHILE ;
(* "" *)
(* "" *)
```

常见问题

1. 使用 setJointsMovel 以及 SetCartMovej 很慢

   这是因为 Efort 控制器进行正、逆运动学很慢，请尽量不要使用。

附录

14.2.6 Elite

此处介绍安装 elite 机械臂驱动的相关事项。

驱动版本和使用要求

支持的机械臂类型

Elite 六轴协作机械臂

控制器型号

Elite

机械臂需要开通的功能

Elite 自带 socket 通讯功能，无需开通。

安装驱动

elite 机械臂驱动文件列表

–lua

- xyz_master.lua （机械臂主控程序）
- xyz_motion.lua （工控机主控程序）
- xyz_status.lua （工控机主控程序）

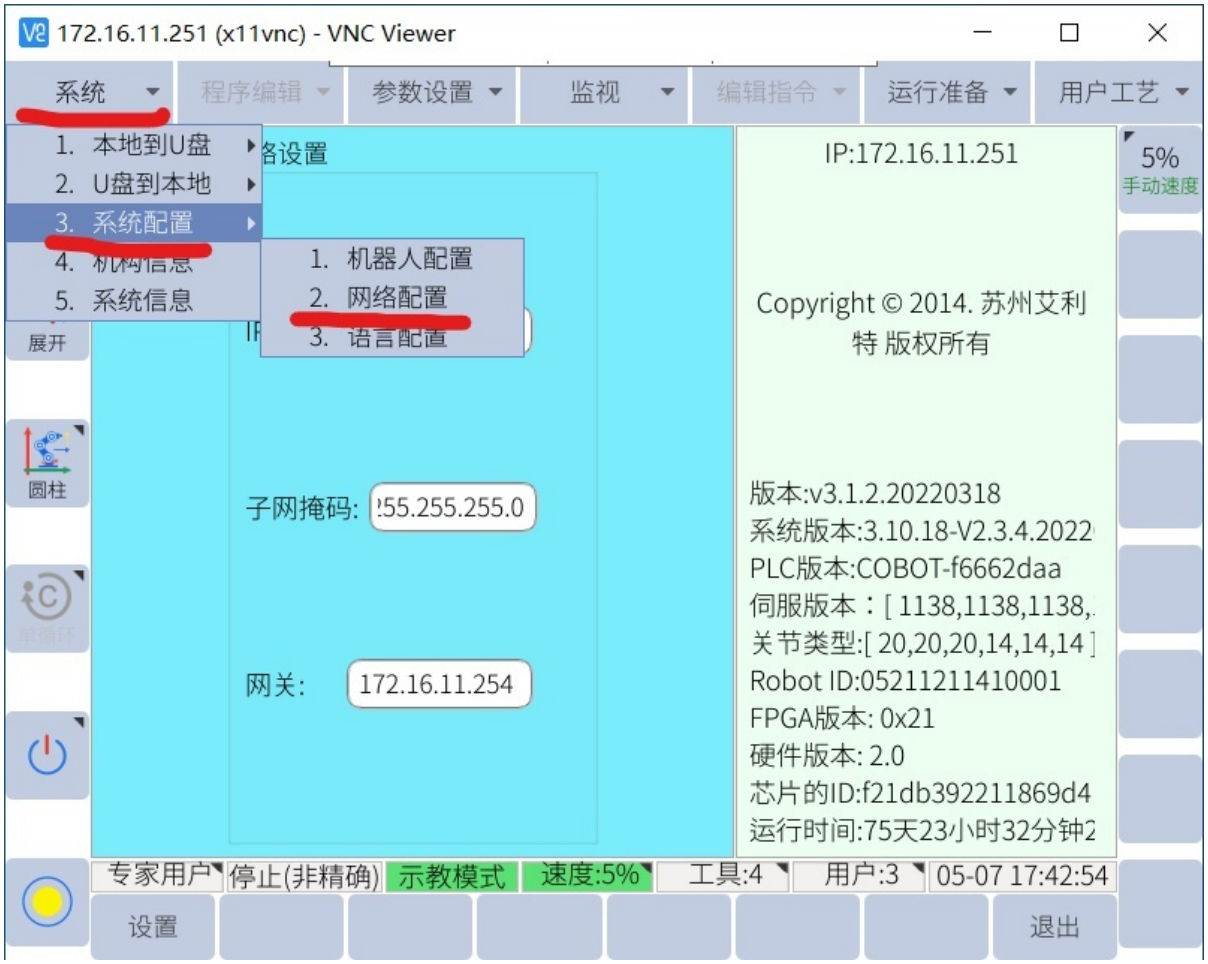
–jbi

- cart_move_basic.jbi （cart move 基础模板）
- cart_move_basic.jbi （cart move 重定位模板）
- traj_move_sync.jbi （traj move 同步模板）
- traj_move_async.jbi （traj move 异步模板）
- xyz_exectraj.jbi （机械臂主控执行轨迹程序）
- xyz_init.jbi （机械臂主控初始化程序）
- xyz_send_recv.jbi （机械臂主控通讯程序）
- xyz_motion.jbi （工控机主控前台程序）

设定机械臂 IP

系统-> 系统配置-> 网络配置

- "IP" 设置为：192.168.37.100，如果项目有其他需求可根据项目实际情况进行修改



设定工控机 IP

1. 用网线连接电脑和控制柜上的 **网线接口**。
2. 将电脑的 ipv4 网络设定为:

- "IP" 设置为：192.168.37.101，如果项目有其他需求可根据项目实际情况进行修改
- "子网掩码" 设置为：255.255.255.0

图 31: elite 设定机械臂 IP

导入程序

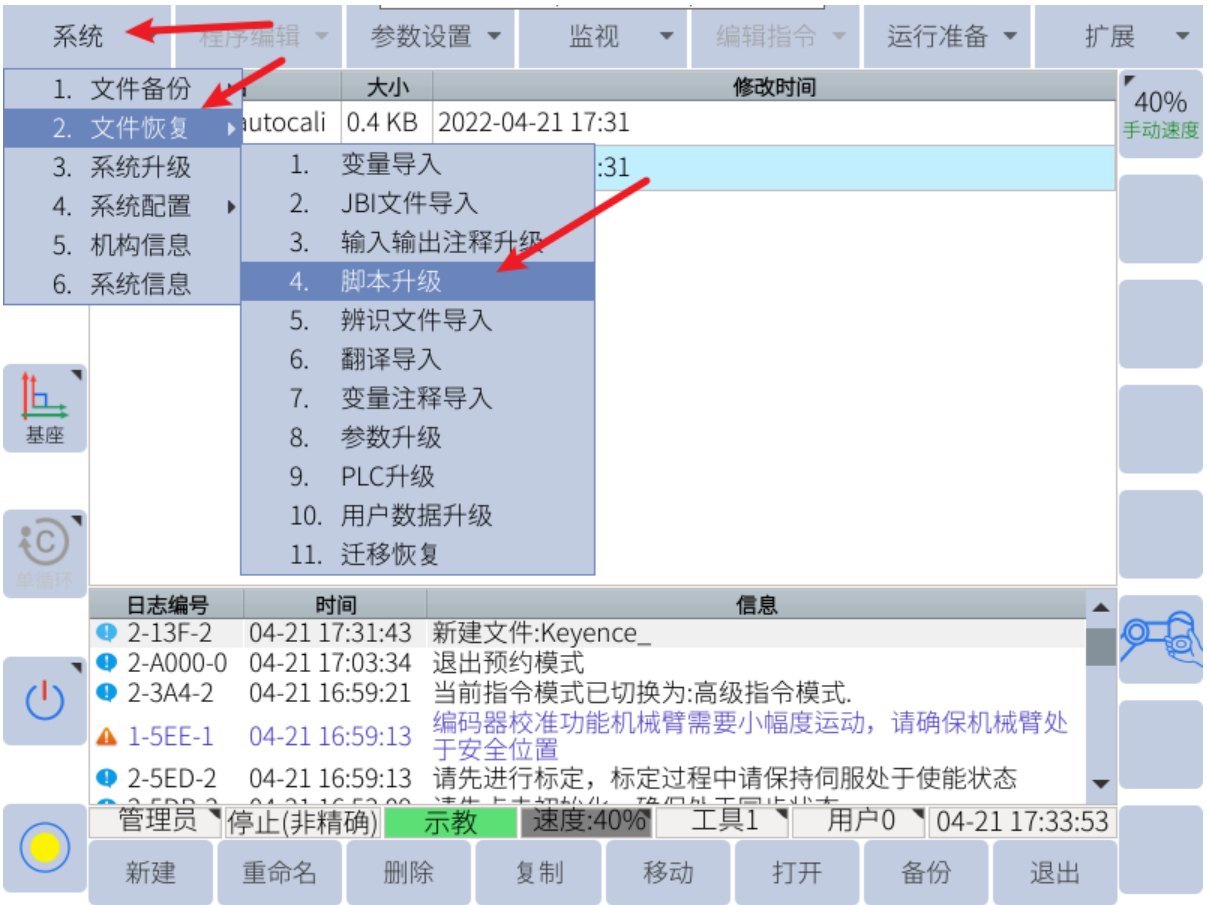
1. 将 MAX 安装目录下的 share/robot_code 中 elete/lbctrl 文件夹拷入 U 盘中，并把 U 盘插入控制柜内的 USB 接口，注意：烧录程序时需要保证当前没有 LUA 脚本正在运行，没有 JBI 程序被选中/运行。
2. 点击 系统 -> 文件恢复（U 盘到本地） -> JBI 文件导入

3. 点击 系统 -> 文件恢复（U 盘到本地） -> 脚本升级

运行程序

通讯说明

工控机和 elite 机械臂的通讯方式为 TCP/IP：xyz_master, xyz_motion 和 xyz_status 中工控机作为 socket server（服务端），机械臂作为 socket client（客户端）。

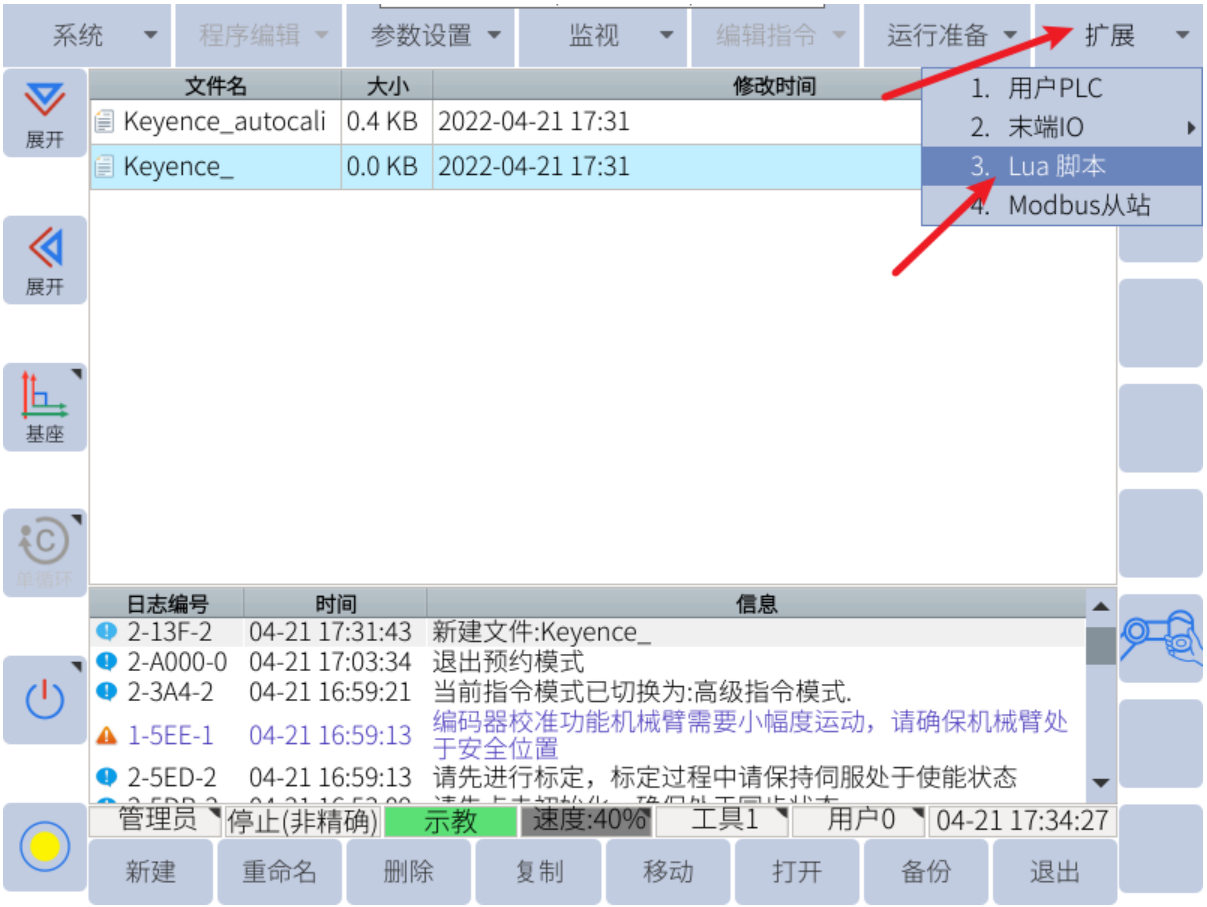

系统
程序编辑
参数设置
监视
编辑指令
运行准备
扩展
1. 文件备份
2. 文件恢复
3. 系统升级
4. 系统配置
5. 机构信息
6. 系统信息
大小
修改时间
autocali
0.4 KB
2022-04-21 17:31
:31
1. 变量导入
2. JBI文件导入
3. 输入输出注释升级
4. 脚本升级
5. 辨识文件导入
6. 翻译导入
7. 变量注释导入
8. 参数升级
9. PLC升级
10. 用户数据升级
11. 迁移恢复
40%
手动速度
基座
单循环
日志编号
时间
信息
2-13F-2 04-21 17:31:43 新建文件:Keyence_
2-A000-0 04-21 17:03:34 退出预约模式
2-3A4-2 04-21 16:59:21 当前指令模式已切换为:高级指令模式.
1-5EE-1 04-21 16:59:13 编码器校准功能机械臂需要小幅度运动，请确保机械臂处于安全位置
2-5ED-2 04-21 16:59:13 请先进行标定，标定过程中请保持伺服处于使能状态
管理员
停止(非精确)
示教
速度:40%
工具1
用户0
04-21 17:33:53
新建
重命名
删除
复制
移动
打开
备份
退出

启动程序

无论是工控机主控还是机械臂主控，都需要先设置 lua 脚本，方法如下。

脚本配置方法

选择 扩展（用户工艺）-> LUA 脚本



将脚本 1, 脚本 2，这两个脚本的开机运行设置为 否，开关设置为 打开。

此外，运行前要检查示教器左下角的状态栏是否为黄色，如果是黄色，需要先按住使能按钮，再点击下图中的黄色按钮，进行自动校准，待黄色按钮变绿，方可运行程序。

图 32: elite 设置 lua 脚本


系统
程序编辑
参数设置
监视
编辑指令
运行准备
扩展
展开
展开
基座
单指环
编号 开机运行 开关 文件 状态
脚本1 否 打开 Keyence_AutoCali
脚本2 否 关闭
脚本3 否 关闭
脚本4 否 关闭
脚本5 否 关闭
脚本6 否 关闭
脚本7 否 关闭
脚本8 否 关闭
40%
手动速度
日志编号 时间 信息
2-13F-2 04-21 17:31:43 新建文件:Keyence_
2-A000-0 04-21 17:03:34 退出预约模式
2-3A4-2 04-21 16:59:21 当前指令模式已切换为:高级指令模式.
1-5EE-1 04-21 16:59:13 编码器校准功能机械臂需要小幅度运动，请确保机械臂处于安全位置
2-5ED-2 04-21 16:59:13 请先进行标定，标定过程中请保持伺服处于使能状态
管理员
停止(非精确)
示教
速度:40%
工具1
用户0
04-21 17:34:40
保存
运行
停止
退出

图 33: 示教器自动校准

工控机主控设置

1. 参照上述脚本设置方法，将脚本 1 设置为 `xyz_motion.lua`，脚本 2 设置为 `xyz_status.lua`。
2. 示教器中选择 `xyz_motion.jbi` 运行即可，每次运行程序前，要运行的 LUA 脚本状态需要设置为 停止。
3. 运行方法：在示教器主界面找到 `xyz_motion.jbi`，然后转动示教器右上角的钥匙至循环模式，此时程序指针会自动回到改程序的第一行，再点击示教器右下角的黄色伺服按钮，最后点击示教器右下角的绿色运行按钮即可。

机械臂主控设置

1. 参照上述脚本设置方法，将脚本 1 设置为 `xyz_master.lua`，脚本 2 设置为 `xyz_status.lua`。
2. 依据项目实际需求，示教器中选择对应的模板 jbi 程序（例如 `cart_move_basic.jbi`）运行即可，每次运行程序时，要运行 LUA 脚本状态需要设置为 停止。
3. 运行方法同工控机主控模式。

API 说明

elite 工控机主控支持的 API

指令号	命令	支持情况
100	获取机械臂程序版本号	支持
101	发送速度数据	支持
102	发送加速度数据	不支持
103	发送圆滑过渡参数	不支持
104	发送工具坐标系 (TCP) 参数	支持
105	设置数字量输出	支持
106	SetJointsMovej	支持
107	SetCartMovel	支持
108	SetJoinsMovel	支持
109	SetCartMovej	不支持
110	MovejSequence	不支持
111	MovelSequence	不支持
112	SetJointsMovejDo	不支持
113	SetCartMovelDo	不支持
114	SetJointsMovelDo	不支持
115	SetJointsMovejGroupDo	不支持
116	SetCartMovelGroupDo	不支持
117	SetJointsMovelGroupDo	不支持
118	MovelUntil	不支持
119	获取数字量输入状态	不支持
120	获取数字量输出状态	支持
121	获取模拟量输入口数值	不支持
122	获取机械臂当前角度	支持
123	获取机械臂当前位姿	支持
200	机械臂后台发送状态	支持

注意：由于 elite 工控机主控程序不支持设置 ACC 与 ZONE，故 ACC 与 ZONE 需要在下位机 `xyz_motion.jbi` 中修改。

elite 工控机主控定义了一些全局变量，不允许用户使用，占用情况如下 (`xyz_motion.jbi`)：另请注意：elite 机器人的各种类型寄存器（包括 tool 寄存器），在使用前均需要打开，否则无法读写。

B 字节型		
地址	含义	说明 (是否用户可写)
B005~B009	内部使用	x
B018	读写标志位	x
B019	运动标志位	x
B020	运动角速度	x

I 整数型		
地址	含义	说明 (是否用户可写)
I010	运动线速度	x

P 位置型		
地址	含义	说明 (是否用户可写)
P001	接收的点	x
P002	发送的点	x
P003	中转的点	x

V 位置型		
地址	含义	说明 (是否用户可写)
V001	接收的点	x
V002	接收的点	x
V003	发送的点	x

elite 机械臂主控支持的 API

elite 机械臂主控定义了一些全局变量，不允许用户占用，占用情况如下 (如 `cart_move_basic.jbi`)：

B 字节型		
地址	含义	说明 (是否用户可写)
B001	通讯标志位	x
B002	发送命令标志位	x
B003	错误代码	x
B004	字符串编号	x
B011~B017	预留，请勿使用	x
B021~B050	轨迹点类型	x

I 整数型		
I011~I021	自定义请求发送整数	x
I021~I030	自定义请求接收整数	x

P 位置型		
地址	含义	说明 (是否用户可写)
P001	接收的点	x
P002	发送的点	x
P011~P040	轨迹点	read only

V 位置型		
地址	含义	说明 (是否用户可写)
V001	接收的点	x
V003	发送的点	x
V011~V040	轨迹点	read only

切换应用 ()

```
SET B002 501
SET B004 1
CALL JOB:xyz_send_recv
```

输入:

- B004: 为后台字符串序号，为应用名称

输出:

- B003: 返回 error code

切换流图 ()

```
SET B002 502
SET B004 1
CALL JOB:xyz_send_recv
```

输入:

- B004: 为后台字符串序号，为流图名称

输出:

- B003: 返回 error code

切换工具 ()

```
SET B002 503
SET B004 1
CALL JOB:xyz_send_recv
```

输入:

- B004: 为后台字符串序号，为工具名称

输出:

- B003: 返回 error code

请求拍照 ()

```
SET B002 504
SET B012 0
CALL JOB:xyz_send_recv
```

输入：

- B012：视觉服务 id

输出：

- B003：返回 error code
- B013：返回 token

获取拍照结果 ()

```
SET B002 505
SET B013 0
CALL JOB:xyz_send_recv
```

输入：

- B013: token 值

输出：

- B003：返回 error code

拍照 ()

```
SET B002 506
SET B012 0
CALL JOB:xyz_send_recv
```

输入：

- B012：视觉服务 id

输出：

- B003：返回 error code

请求抓取目标点位 ()

```
SET B002 507
SET B011 0
CALL JOB:xyz_send_recv
```

输入：

- B011：工作空间 id

输出：

- B003：返回 error code

获取抓取目标点位 ()

```
SET B002 508
SET B013 0
CALL JOB:xyz_send_recv
```

输入：

- B013：token 值

输出：

- V001：目标点位姿

- B014：当前有多少个可供抓取的点
- B015：pipeline 文件 number 值
- B016：用到的注册文件的注册 number 值
- B003：返回 error code

请求物体位姿 ()

```
SET B002 509
SET B011 0
CALL JOB:xyz_send_recv
```

输入：

- B011：工作空间 id

输出：

- B013：返回的 token
- B003：返回 error code

获取物体位姿 ()

```
SET B002 510
SET B013 0
CALL JOB:xyz_send_recv
```

输入：

- B013：工件位姿 token

输出：

- V001：物体位姿
- B014：当前物体位姿个数
- B017：物体姿态类型
- B003：返回 error code

重置任务 ()

```
SET B002 511
CALL JOB:xyz_send_recv
```

输出：

- B003：返回 error code

发送机械臂当前角度 ()

```
SET B002 512
SET P002 0,0,0,0,0,0
CALL JOB:xyz_send_recv
```

输入：

- P002：机器人当前角度

输出：

- B003：返回 error code

发送机械臂当前笛卡尔位姿 ()

```
SET B002 513
SET V003 0,0,0,0,0,0
CALL JOB:xyz_send_recv
```

输入：

- V003：机器人当前笛卡尔位姿

输出：

- B003：返回 error code

发送机械臂当前扩展轴位置 ()

```
SET B002 514
SET P002 0,0,0,0,0,0
CALL JOB:xyz_send_recv
```

输入：

- P002：机器人当前角度

输出：

- B003：返回 error code

请求 **pick** 动作规划 ()

```
SET B002 515
CALL JOB:xyz_send_recv
```

输出：

- B003：返回 error code

请求 **place** 动作规划 ()

```
SET B002 516
CALL JOB:xyz_send_recv
```

输出：

- B003：返回 error code

请求 **pick** 和 **place** 规划 ()

```
SET B002 517
SET B011 0
CALL JOB:xyz_send_recv
```

输入：

- B011: 工作空间 id

输出：

- B003：返回 error code

获取取料入框轨迹 ()

```
SET B002 518
SET B011 0
CALL JOB:xyz_send_recv
```

输入:

- B011: 工作空间 id

输出:

- B003：返回 error code
- B015：pipeline 文件 number 值
- B016：用到的注册文件的注册 number 值
- 轨迹点

获取取料出框轨迹 ()

```
SET B002 519
SET B011 0
CALL JOB:xyz_send_recv
```

输入:

- B011: 工作空间 id

输出:

- B003：返回 error code
- B015：pipeline 文件 number 值
- B016：用到的注册文件的注册 number 值
- 轨迹点

获取放料入框轨迹 ()

```
SET B002 520
SET B011 0
CALL JOB:xyz_send_recv
```

输入:

- B011: 工作空间 id

输出:

- B003：返回 error code
- B015：pipeline 文件 number 值
- B016：用到的注册文件的注册 number 值
- 轨迹点

获取放料出框轨迹 ()

```
SET B002 521
SET B011 0
CALL JOB:xyz_send_recv
```

输入：

- B011: 工作空间 id

输出：

- B003：返回 error code
- B015：pipeline 文件 number 值
- B016：用到的注册文件的注册 number 值
- 轨迹点

请求切换策略 ()

```
SET B002 522
SET B004 1
CALL JOB:xyz_send_recv
```

输入：

- B004：为后台字符串序号，为策略名称

输出：

- B003：返回 error code

料箱重定位 ()

```
SET B002 523
CALL JOB:xyz_send_recv
```

输出：

- V001：料框位姿
- B003：返回 error code

工件在上手的二次定位 ()

```
SET B002 524
CALL JOB:xyz_send_recv
```

输出：

- B003：返回 error code

工件不在手上的二次定位 ()

```
SET B002 525
CALL JOB:xyz_send_recv
' 随后可以直接调用轨迹执行
CALL JOB:xyz_exectraj
```

输出：

- B003：返回 error code
- B015：pipeline 文件 number 值
- B016：用到的注册文件的注册 number 值
- 轨迹点

获取工件姿态类型 ()

```
SET B002 526
CALL JOB:xyz_send_recv
```

输出:

- B003: 返回 error code
- B017: 工件姿态类型

重置工业码垛状态 ()

```
SET B002 527
CALL JOB:xyz_send_recv
```

输出:

- B003: 返回 error code

切换工件 ()

```
SET B002 528
SET B004 0
SET B011 0
CALL JOB:xyz_send_recv
```

输入:

- B004: 字符串序号，工件名称
- B011: 工作空间 id

输出:

- B003: 返回 error code

计算抓取目标点位 ()

该指令等价于请求抓取目标点位 + 获取抓取目标点位

```
SET B002 529
SET B011 0
CALL JOB:xyz_send_recv
```

输入:

- B011: 工作空间 id

输出:

- V001: 目标点位姿
- B014: 当前有多少个可供抓取的点
- B015: pipeline 文件 number 值
- B016: 用到的注册文件的注册 number 值
- B003: 返回 error code

获取物体位姿 ()

```
SET B002 530
SET B011 0
CALL JOB:xyz_send_recv
```

输入：

- B011：工作空间 id

输出：

- V001：物体位姿
- B014：当前物体位姿个数
- B017：物体姿态类型
- B003：返回 error code

案例/模板说明

机械臂主控主函数说明

以下为机械臂主控模板，这里介绍前台程序的用法以及相关寄存器的用途，寄存器的全部定义请参照 lua 文件的开头部分。

```
-------------------------cart_move_basic.jbi---------------------------------

//
NOP

S1:连接到上位机
// start backend connection
// xyz_bkgd is the first lua script
STARTLUA INDEX=1  //启用xyz_master.lua脚本

S2: 初始化参数
// init communication
CALL JOB:xyz_init   //调用初始化程序

// switch flow
SET B002 502            //设置cmd寄存器值为502
SET B004 1              //elite没有字符串寄存器，后台预留了五个字符串
                        //B004的值代表选用的是第几个字符串，现在选用第一个作为flow␣
↪name
CALL JOB:xyz_send_recv  //调用通讯程序

S3: 切换工件
// switch item
SET B002 528            //设置cmd寄存器值为502
SET B004 2              //选用后台预留的第二个作字符串为item name
SET B011 0              //设置ws_id
CALL JOB:xyz_send_recv  //调用通讯程序

DOUT OT# (1) OFF        //设置DO

S4: 移动到home点，需要提前设置好home点位置
MOVJ P001 VJ=50 PL=0 ACC=50 DEC=50 //运动到某一示教点
```

(下页继续)

(续上页)

```
SET LB000 0
SET LB001 0

// loop start           //循环开始
WHILE LB000=0 DO
LABEL *loop_start

S5: 眼在手上
// send current pose
IF LB001=0 THEN
    S6: 移动到拍照位
    MOVJ P001 VJ=50 PL=0 ACC=50 DEC=50
    S7: 发送拍照位姿给上位机
    SET B002 513            //设置cmd寄存器值为513
    SET V002 0,0,0,0,0,0    //设置要发送给上位机的cart pose
    CALL JOB:xyz_send_recv  //调用通讯程序
ENDIF

S8: 请求抓取位姿
// request grasp pose
SET B002 507                //设置cmd寄存器值为507
SET B011 0                  //设置ws_id
CALL JOB:xyz_send_recv      //调用通讯程序

S9: 获取抓取位姿
// get grasp pose
SET B002 508                //设置cmd寄存器值为508
SET B013 0                  //设置token
CALL JOB:xyz_send_recv      //调用通讯程序

IF B014 < 1 THEN
    TIMER T=5               //未识别，重新拍照
    JUMP *loop_start
ENDIF

S10: 移动到抓取位并抓取
注意，可能需要添加一些过渡点
MOVL V001 AV=400.0 MM/S CR =10.0MM   //移动到grasp pose
DOUT OT# (1) ON                      //触发DO抓取
TIMER T=0.5
S11: 移动到放置位并放置
MOVL V003 AV=400.0 MM/S CR =10.0MM   //移动至放置点
DOUT OT# (1) OFF                     //触发DO放置

ENDWHILE

END


-------------------------cart_move_repo.jbi---------------------------------
//
NOP
S1: 连接到上位机
// start backend connection
// xyz_bkgd is the first lua script
```

(下页继续)

(续上页)

```
STARTLUA INDEX=1          //启用xyz_master.lua脚本

S2: 初始化参数
// init communication
CALL JOB:xyz_init         //调用初始化程序

// switch flow
SET B002 502              //设置cmd寄存器值为502
SET B004 1                //elite没有字符串寄存器，后台预留了五个字符串
                          //B004的值代表选用的是第几个字符串，现在选用第一个作为flow␣
↪name
CALL JOB:xyz_send_recv    //调用通讯程序

S3: 切换工件
// switch item
SET B002 528              //设置cmd寄存器值为502
SET B004 2                //选用后台预留的第二个作字符串为item name
SET B011 0                //设置ws_id
CALL JOB:xyz_send_recv    //调用通讯程序

DOUT OT# (1) OFF          //设置DO
DOUT OT# (2) OFF

S4: 移动到home点，需要提前设置好home点位置
MOVJ P001 VJ=50 PL=1 ACC=50 DEC=50   //运动到某一示教点

S5: 请求抓取位姿
// request grasp pose
SET B002 507              //设置cmd寄存器值为507
SET B011 0                //设置ws_id
CALL JOB:xyz_send_recv    //调用通讯程序

SET LB000 0

// loop start             //循环开始
WHILE LB000=0 DO
LABEL *loop_start

S6: 获取抓取位姿
// get grasp pose
SET B002 508                  //设置cmd寄存器值为508
SET B013 0                    //设置token
CALL JOB:xyz_send_recv        //调用通讯程序

S7:
IF B014 < 1 THEN
    S15: 上位机切换至识别隔板，工作空间中已经没有工件，需要先取走隔板
    // switch item
    SET B002 528              //设置cmd寄存器值为502
    SET B004 3                //选用后台预留的第三个作字符串为item name
    SET B011 0                //设置ws_id
    CALL JOB:xyz_send_recv    //调用通讯程序

    S16: 请求隔板抓取位姿
    // request grasp pose
    SET B002 507              //设置cmd寄存器值为507
```

(下页继续)

(续上页)

```
    SET B011 0              //设置ws_id
    CALL JOB:xyz_send_recv  //调用通讯程序

    S17: 获取隔板抓取位姿
    // get grasp pose
    SET B002 508                //设置cmd寄存器值为508
    SET B013 0                  //设置token
    CALL JOB:xyz_send_recv      //调用通讯程序

    S18: 移动到抓取位并抓取隔板
    MOVL V001 AV=400.0 MM/S CR =10.0MM      //移动到抓取点
    DOUT OT# (1) ON

    S19: 移动到放置位并放置
    MOVL V001 AV=400.0 MM/S CR =10.0MM      //移动到放置点
    DOUT OT# (1) OFF

    S20: 上位机切换至识别工件
    // switch item
    SET B002 528            //设置cmd寄存器值为502
    SET B004 2              //选用后台预留的第三个作字符串为item name
    SET B011 0              //设置ws_id
    CALL JOB:xyz_send_recv  //调用通讯程序

    // request grasp pose
    SET B002 507            //设置cmd寄存器值为507
    SET B011 0              //设置ws_id
    CALL JOB:xyz_send_recv  //调用通讯程序

    JUMP *loop_start
ELSE
    S8: 移动到拍照位，需要先示教该拍照位
    MOVL V003 AV=400.0 MM/S CR =10.0MM

    S9: 发送拍照位姿
    // send current pose
    SET B002 513            //设置cmd寄存器值为513
    SET V002 0,0,0,0,0,0    //设置要发送给上位机的cart pose
    CALL JOB:xyz_send_recv  //调用通讯程序

    S10: 上位机切换至识别工件
    // switch item
    SET B002 528            //设置cmd寄存器值为502
    SET B004 2              //选用后台预留的第三个作字符串为item name
    SET B011 1              //设置ws_id
    CALL JOB:xyz_send_recv  //调用通讯程序

    S11: 请求抓取位姿
    // request grasp pose
    SET B002 507            //设置cmd寄存器值为507
    SET B011 1              //设置ws_id
    CALL JOB:xyz_send_recv  //调用通讯程序

    S12: 获取抓取位姿
    // get grasp pose
    SET B002 508                //设置cmd寄存器值为508
```

(下页继续)

(续上页)

```
    SET B013 0                  //设置token
    CALL JOB:xyz_send_recv      //调用通讯程序

    S13: 移动到抓取位并抓取
    MOVL V001 AV=400.0 MM/S CR =10.0MM      //移动到放置点
    DOUT OT# (2) ON

    S14: 移动到放置位并放置
    MOVL V004 AV=400.0 MM/S CR =10.0MM
    DOUT OT# (2) OFF
ENDIF
// switch item
SET B002 528            //设置cmd寄存器值为502
SET B004 2              //选用后台预留的第二个作字符串为item name
SET B011 0              //设置ws_id
CALL JOB:xyz_send_recv  //调用通讯程序

// request grasp pose
SET B002 507            //设置cmd寄存器值为507
SET B011 0              //设置ws_id
CALL JOB:xyz_send_recv  //调用通讯程序

ENDWHILE

END


-------------------------traj_move_sync.jbi---------------------------------
//
NOP
S1: 连接上位机
// start backend connection
// xyz_bkgd is the first lua script
STARTLUA INDEX=1  //启用xyz_master.lua脚本

S2: 初始化参数
// init communication
CALL JOB:xyz_init   //调用初始化程序

// switch flow
SET B002 502            //设置cmd寄存器值为502
SET B004 1              //elite没有字符串寄存器，后台预留了五个字符串
                        //B004的值代表选用的是第几个字符串，现在选用第一个作为flow␣
↪name
CALL JOB:xyz_send_recv  //调用通讯程序

S3: 切换工件
// switch item
SET B002 528            //设置cmd寄存器值为502
SET B004 2              //选用后台预留的第二个作字符串为item name
SET B011 0              //设置ws_id
CALL JOB:xyz_send_recv  //调用通讯程序

DOUT OT# (1) OFF        //设置DO

S4: 移动到home点，需要提前设置好home点位置
```

(下页继续)

(续上页)

```
MOVJ P001 VJ=50 PL=0 ACC=50 DEC=50 //运动到某一示教点

SET LB000 0

// loop start           //循环开始
WHILE LB000=0 DO
LABEL *loop_start

S5: 请求抓取和放置规划
// request pick and place
SET B002 517               //设置cmd寄存器值为517
SET B011 0                 //设置ws_id
CALL JOB:xyz_send_recv     //调用通讯程序

S6: 获取抓取入筐轨迹
// get pick in result
SET B002 518               //设置cmd寄存器值为518
SET B011 0                 //设置ws_id
CALL JOB:xyz_send_recv     //调用通讯程序

PAUSE IF B014 < 1          //清筐

S7: 执行抓取入筐轨迹
CALL JOB:xyz_exectraj      //执行轨迹
DOUT OT# (1) ON

S8: 获取抓取出筐轨迹
// get pick out result
SET B002 519               //设置cmd寄存器值为519
SET B011 0                 //设置ws_id
CALL JOB:xyz_send_recv     //调用通讯程序

S9: 执行抓取出筐轨迹
CALL JOB:xyz_exectraj      //执行轨迹

S10: 获取放置入筐轨迹
// get place in result
SET B002 520               //设置cmd寄存器值为520
SET B011 0                 //设置ws_id
CALL JOB:xyz_send_recv     //调用通讯程序

S11: 执行放置入筐轨迹
CALL JOB:xyz_exectraj      //执行轨迹
DOUT OT# (1) OFF

S12: 获取放置出筐轨迹
// get place out result
SET B002 521               //设置cmd寄存器值为521
SET B011 0                 //设置ws_id
CALL JOB:xyz_send_recv     //调用通讯程序

S13: 执行放置出筐轨迹
CALL JOB:xyz_exectraj      //执行轨迹

ENDWHILE
```

(下页继续)

(续上页)

```
END


-------------------------traj_move_async.jbi---------------------------------
//
NOP

S1: 连接到上位机
// start backend connection
// xyz_bkgd is the first lua script
STARTLUA INDEX=1  //启用xyz_master.lua脚本

S2: 初始化参数
// init communication
CALL JOB:xyz_init   //调用初始化程序

// switch flow
SET B002 502            //设置cmd寄存器值为502
SET B004 1              //elite没有字符串寄存器，后台预留了五个字符串
                        //B004的值代表选用的是第几个字符串，现在选用第一个作为flow␣
↪name
CALL JOB:xyz_send_recv  //调用通讯程序

S3: 切换工件
// switch item
SET B002 528            //设置cmd寄存器值为502
SET B004 2              //选用后台预留的第二个作字符串为item name
SET B011 0              //设置ws_id
CALL JOB:xyz_send_recv  //调用通讯程序

DOUT OT# (1) OFF        //设置DO

S4: 移动到home点，需要提前设置好home点位置
MOVJ P001 VJ=50 PL=0 ACC=50 DEC=50 //运动到某一示教点

S5: 请求抓取和放置规划
// request pick and place
SET B002 517                //设置cmd寄存器值为517
SET B011 0                  //设置ws_id
CALL JOB:xyz_send_recv      //调用通讯程序

SET LB000 0

// loop start           //循环开始
WHILE LB000=0 DO
LABEL *loop_start

S6: 获取抓取入筐轨迹
// get pick in result
SET B002 518                //设置cmd寄存器值为518
SET B011 0                  //设置ws_id
CALL JOB:xyz_send_recv      //调用通讯程序

PAUSE IF B014 < 1           //清筐

S7: 执行抓取入筐轨迹
```

(下页继续)

(续上页)

```
CALL JOB:xyz_exectraj        //执行轨迹
DOUT OT# (1) ON

S8: 获取抓取出筐轨迹
// get pick out result
SET B002 519                 //设置cmd寄存器值为519
SET B011 0                   //设置ws_id
CALL JOB:xyz_send_recv       //调用通讯程序

S9: 执行抓取出筐轨迹
CALL JOB:xyz_exectraj        //执行轨迹

S10: 请求下一次的抓取和放置规划
// request pick and place
SET B002 517                 //设置cmd寄存器值为517
SET B011 0                   //设置ws_id
CALL JOB:xyz_send_recv       //调用通讯程序

S11: 获取放置入框轨迹
// get place in result
SET B002 520                 //设置cmd寄存器值为520
SET B011 0                   //设置ws_id
CALL JOB:xyz_send_recv       //调用通讯程序

S12: 执行放置入筐轨迹
CALL JOB:xyz_exectraj        //执行轨迹
DOUT OT# (1) OFF

S13: 获取放置出筐轨迹
// get place out result
SET B002 521                 //设置cmd寄存器值为521
SET B011 0                   //设置ws_id
CALL JOB:xyz_send_recv       //调用通讯程序

S14: 执行放置出筐轨迹
CALL JOB:xyz_exectraj        //执行轨迹

ENDWHILE

END
```

常见问题

附录

14.2.7 Fanuc

此处介绍安装 fanuc 机械臂驱动的相关事项。

驱动版本和使用要求

支持的机械臂类型

fanuc 六轴机械臂、四轴机械臂（拆码垛机械臂）

控制器型号

R30iA，R30iB，R30iB_Plus

机械臂需要开通的功能

Fanuc 机械臂本身需要具有的功能：

- R648 User Socket Msg
- R632 KAREL

可在机械臂示教器主界面 MENU -> 下页 -> 状态 -> 版本 ID -> NEXT -> 订购文件查看控制器是否具有相应功能

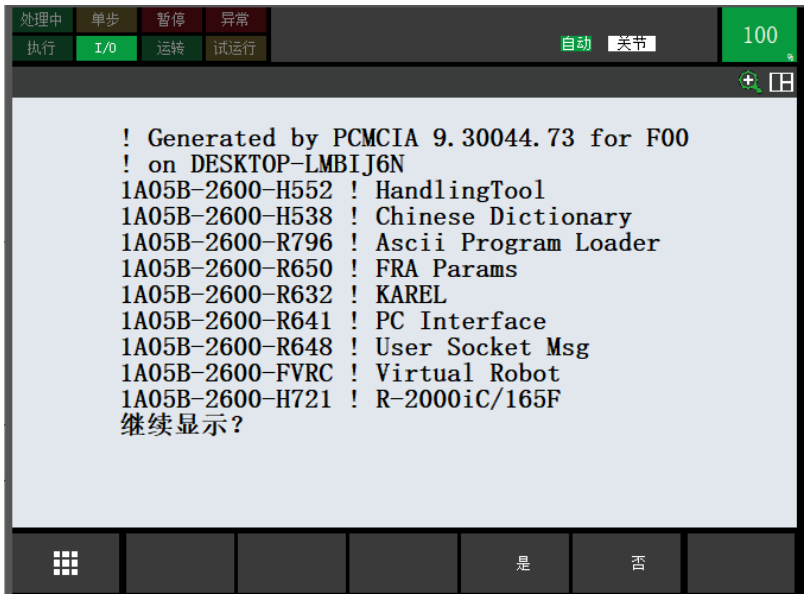


图 34: Fanuc 控制器功能查看

安装驱动

机器人版本

烧录程序前需要先判断当前机器人的系统版本，在示教器上按 MENU 键，进入 实用工具-> 声明。

图片左上角圈出的部分即为机器人版本。

fanuc 机械臂驱动文件列表

驱动文件位于 MAX 安装目录的 `share/robot_code/fanuc/` 路径下，文件包括:

- 编译后的 pc,tp 文件位于 `share/robot_code/fanuc/compiled/` ，该目录下存放了 v8.3，v9.0，v9.3 这三个 fanuc 机器人控制器版本的程序。
- 源代码 kl 文件位于 `share/robot_code/fanuc/karel/` ，源代码 ls 文件位于 `share/robot_code/fanuc/tpe/` ，源代码文件可以根据项目的特定要求或者版本需求修改并编译使用。

fanuc 控制器对于 karel 程序和 ls 程序在大版本号相同的前提下向后兼容，例如 v9.1 版本的控制器可以导入 v9.0 的程序，但是不能导入 v8.3 的程序。所以如果控制器的版本和提供的程序版本不同，可以按照向后兼容的原则来选择合适的程序，或者根据源代码文件直接编译。

以下只介绍 pc,tp 文件的含义。

–compiled 编译后的文件，这里只以 v9.3 为例

–v9.3 软件版本

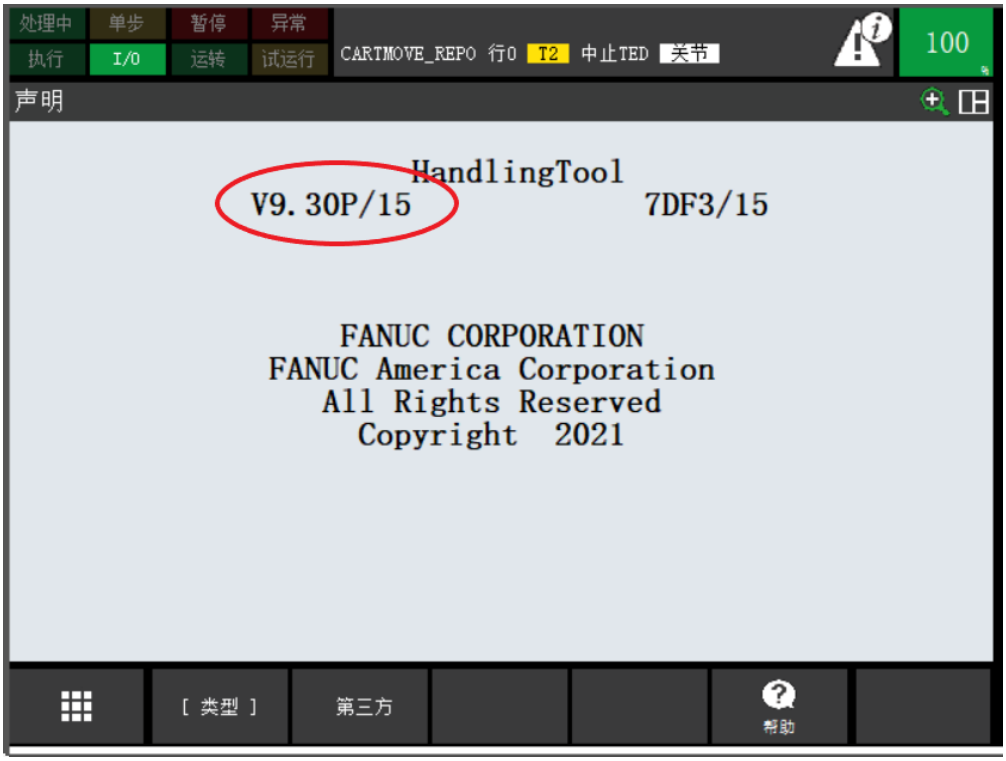


图 35: 查看机器人版本

–pc

- xyz_abt_bkgd 中止后台程序
- xyz_bkgd.pc 后台程序
- xyz_capimg.pc 拍照
- xyz_g_capimg.pc 获取拍照结果
- xyz_g_g_pose.pc 获取抓取结果
- xyz_g_obj_pt.pc 获取工件位姿类型
- xyz_g_o_pose.pc 获取工件位姿
- xyz_g_picin.pc 获取抓取入筐轨迹
- xyz_g_picout.pc 获取抓取出筐轨迹
- xyz_g_plain.pc 获取放置入筐轨迹
- xyz_g_plaout.pc 获取放置出筐轨迹
- xyz_log.pc 日志相关函数
- xyz_motion.pc 工控机主控 motion socket 函数
- xyz_pkt.pc socket package 相关函数
- xyz_r_capimg.pc 请求拍照
- xyz_reset_ps.pc 重置码垛状态
- xyz_reset_v.pc 重置视觉

- xyz_r_g_pose.pc 请求抓取位姿
- xyz_r_o_pose.pc 请求工件位姿
- xyz_r_pick.pc 请求计算抓取轨迹
- xyz_r_picpla.pc 请求计算抓取放置轨迹
- xyz_r_place.pc 请求放置轨迹
- xyz_s_c_pose.pc 发送当前法兰笛卡尔位姿
- xyz_s_ext_j.pc 发送扩展轴当前角度
- xyz_s_joints.pc 发送机械臂各轴角度
- xyz_sock.pc socket 相关函数
- xyz_status.pc 工控机主控 status socket 相关函数
- xyz_sw_app.pc 切换应用
- xyz_sw_obj.pc 切换工件
- xyz_sw_strat.pc 切换策略
- xyz_sw_tool.pc 切换工具
- xyz_sw_flow.pc 切换 flow
- xyz_u_obj_oh.pc 工件在上手的二次定位
- xyz_u_obj_th.pc 工件不在手上的二次定位
- xyz_u_t_pose.pc 料箱重定位
- xyz_c_g_pose.pc 计算抓取位姿
- xyz_c_o_pose.pc 计算物体位姿
- xyz_usr_cmd.pc 自定义请求

–tp

- xyz_exectraj.tp 执行轨迹程序
- trajmove_sync.tp traj move 同步模板
- trajmove_async.tp traj move 异步模板
- cartmove_basic.tp cart move 基础模板
- cartmove_repo.tp cart move 重定位模板
- xyzmotion.tp 工控机主控 motion socket 函数
- xyz_move.tp 工控机主控机械臂移动函数
- xyzstatus.tp 工控机主控 status 函数函数
- xyz.tp 工控机主控启动程序

–karel 此处为源代码，未经编译

–src

–include

- xyz_bkgd_h.kl
- xyz_const.kl

    - xyz_log_h.kl
    - xyz_motion_h.kl
    - xyz_pkt_h.kl
    - xyz_pkt_t.kl
    - xyz_sock_h.kl
    - xyz_sock_t.kl
    - xyz_status_h.kl
- xyz_abt_bkgd.kl
- xyz_bkgd.kl
- xyz_capimg.kl
- xyz_g_capimg.kl
- xyz_g_g_pose.kl
- xyz_g_obj_pt.kl
- xyz_g_o_pose.kl
- xyz_g_picin.kl
- xyz_g_picout.kl
- xyz_g_plain.kl
- xyz_g_plaout.kl
- xyz_log.kl
- xyz_motion.kl
- xyz_pkt.kl
- xyz_r_capimg.kl
- xyz_reset_ps.kl
- xyz_reset_v.kl
- xyz_r_g_pose.kl
- xyz_r_o_pose.kl
- xyz_r_pick.kl
- xyz_r_picpla.kl
- xyz_r_place.kl
- xyz_s_c_pose.kl
- xyz_s_ext_j.kl
- xyz_s_joints.kl
- xyz_sock.kl
- xyz_status.kl
- xyz_sw_app.kl
- xyz_sw_obj.kl

- xyz_sw_strat.kl
- xyz_sw_tool.kl
- xyz_u_obj_oh.kl
- xyz_u_obj_th.kl
- xyz_u_t_pose.kl
- xyz_sw_flow.kl
- xyz_c_g_pose.kl
- xyz_c_o_pose.kl
- xyz_usr_cmd.kl

–**tpe**

–**src**

- XYZ_EXECTRAJ.LS
- xyz.ls
- TRAJMOVE_SYNC.ls
- TRAJMOVE_ASYNC.ls
- CARTMOVE_BASIC.LS
- CARTMOVE_REPO.LS
- xyzmotion.ls
- xyz_move.ls
- xyzstatus.ls

设定机械臂 IP

1. 在示教器上按 MENU 键，进入 设置-> 主机通讯。

2. 选中协议一栏中的 1 TCP/IP，并点击示教器上的 ENTER 进入 TCP/IP 设置界面。对 TCP/IP 界面中的 IP 地址，子网掩码进行修改。

- IP 地址设置为：192.168.37.100
- 子网掩码设置为：255.255.255.0

机械臂默认 IP 地址为 192.168.37.100，如果项目有其他需求可根据项目实际情况进行修改，设置完成后界面如图所示

图 36: fanuc 设定机械臂 IP 1

图 37: fanuc 设定后的 TCP/IP 界面

配置工控机主控 socket

工控机主控模式下，fanuc 机械臂作为服务端，占用服务器的两个标签分别是 S3 和 S4。接下来介绍服务器标签的设置过程

1. 在示教器上按 MENU 键，进入 设置-> 主机通讯。

2. SHIFT + DISPALY 设置为双画面。
3. 按示教器上的 DISP 切到左侧界面，点击 显示，选择服务器。

4. 依次进入 S3 和 S4 标签并按照如下步骤进行设置
   - 如果当前状态为 已定义，则点击 动作改为 未定义；
   - 如果当前状态为 已开始，则点击 动作 -> 停止，然后再改为 未定义；
   - 如果修改状态失败，尝试重启机器人后再次修改；
   - 设定 协议为 SM；
   - 修改 服务器 IP/主机名称为 192.168.37.100；
   - 将 启动状态改为 开始（系统重启后可以自动恢复之前服务器的设置）。
5. 按示教器上的 DISP 切到右侧界面。点击 MENU, 进入 系统-> 下页-> 变量，找到并进入 $HOSTS_CFG（使用 SHIFT + 上下方向键可快速翻页）。

图 38: fanuc 设定工控机主控 socket 1

图 39: fanuc 设定工控机主控 socket 2

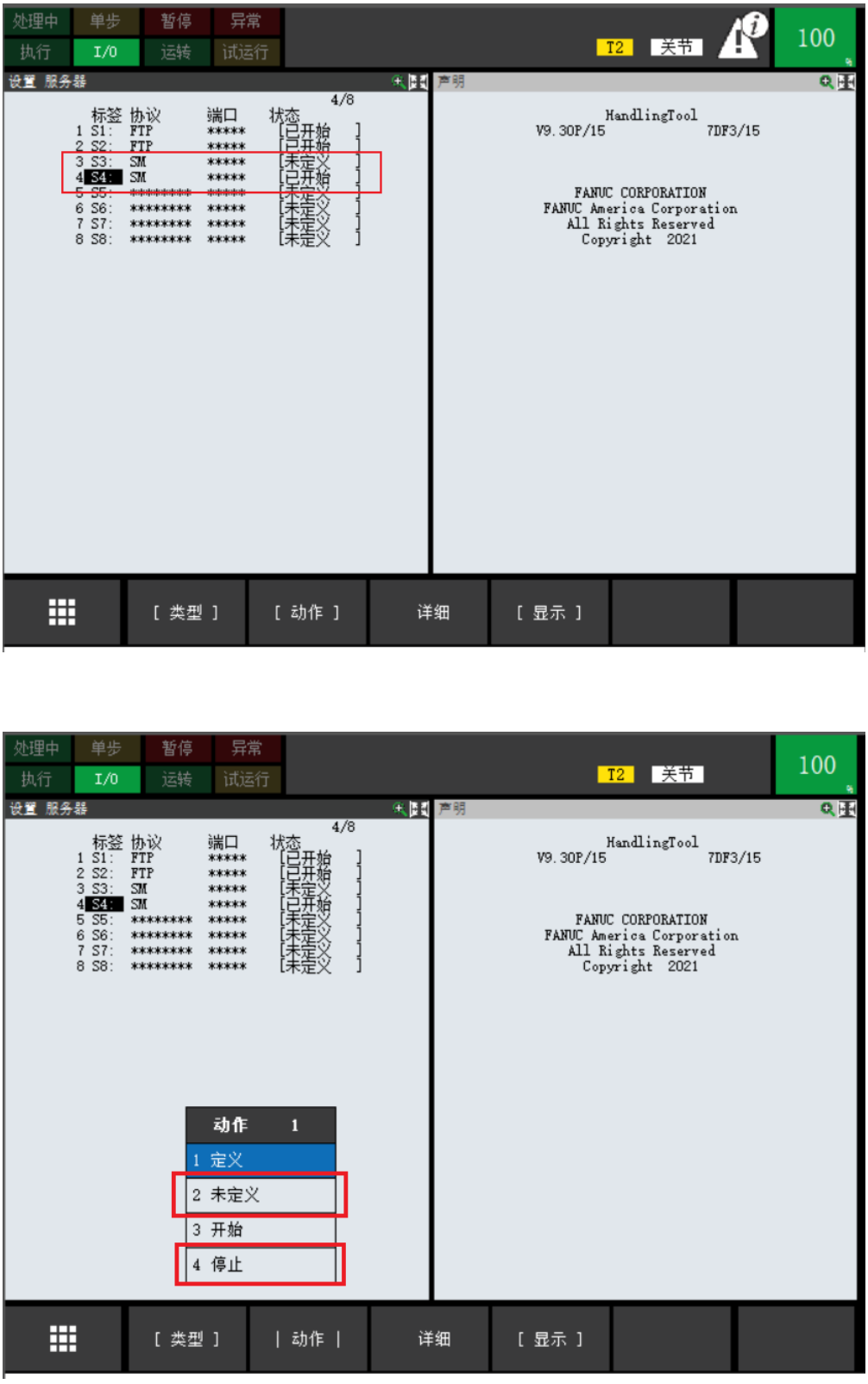


图 40: fanuc 设定工控机主控 socket 3

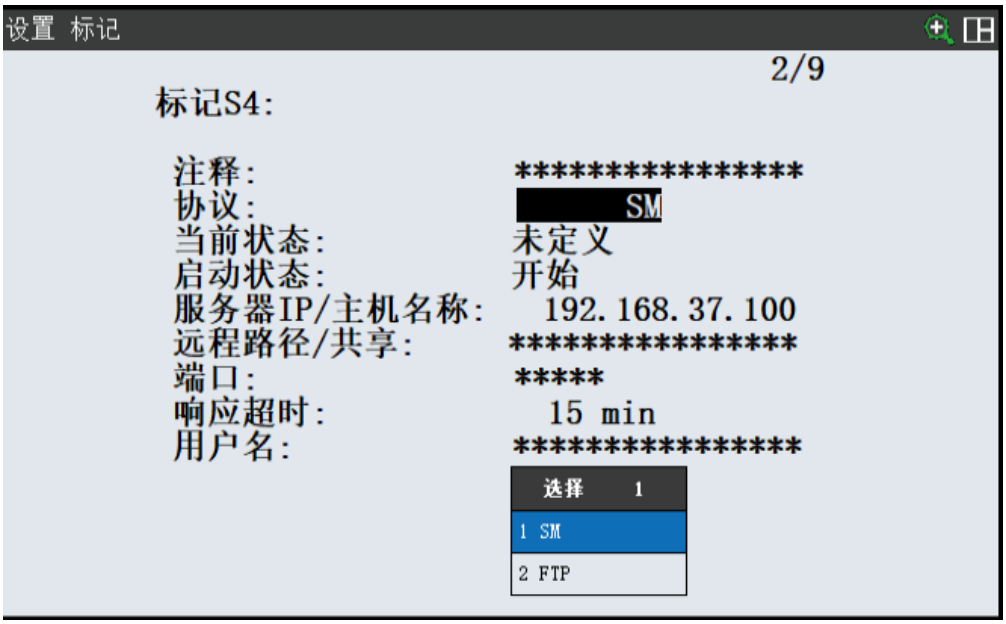


图 41: fanuc 设定工控机主控 socket 4

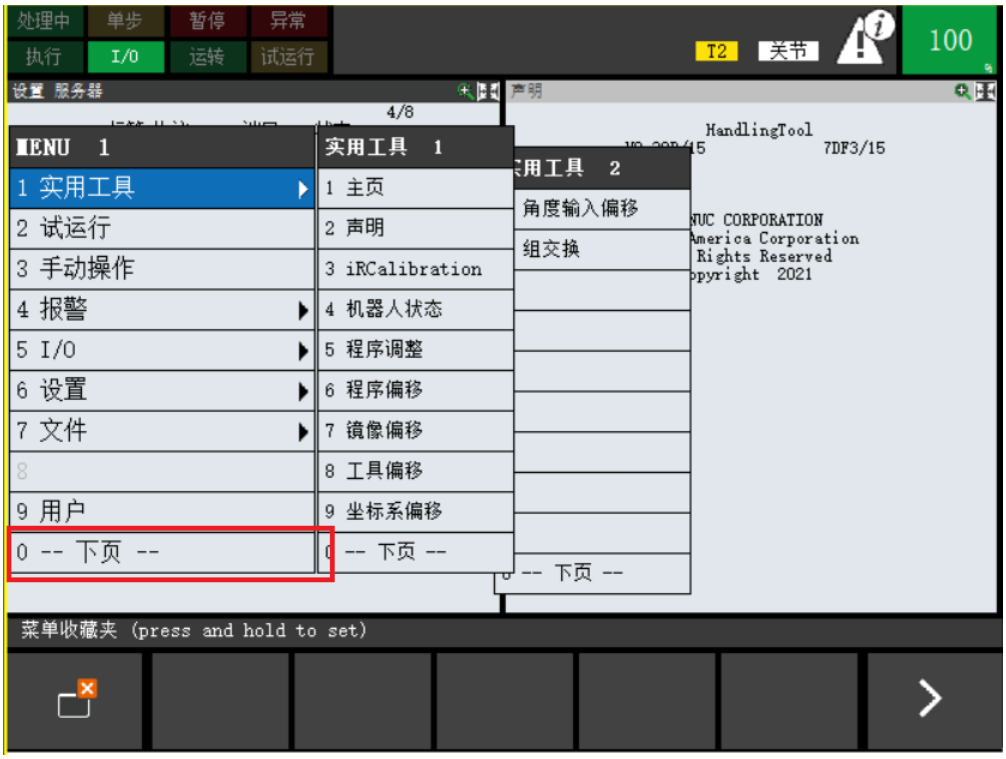


图 42: fanuc 设定工控机主控 socket 5

6. 分别修改 $HOSTS_CFG[3] 和 $HOSTS_CFG[4] 的 $SERVER_PORT 分别为 10100 和 10102。

7. 修改完成后，左侧界面回到服务器标签界面，分别移动光标选中 S3 和 S4 并进行如下操作

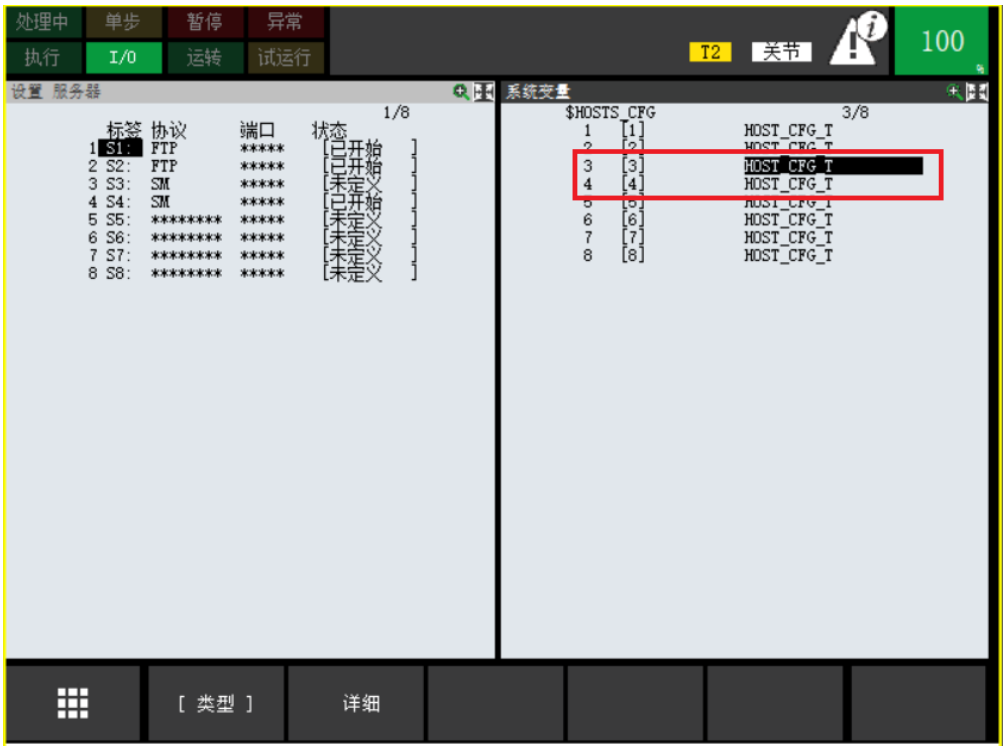
- 点击 动作选中 定义按 ENTER 将状态改为 定义
- 点击 动作选中 开始按 ENTER 将状态改为 开始
- 最后进入 S3 和 S4 查看 当前状态和 启动状态都应该被设置成了 开始。

配置机械臂主控 socket

机械臂主控模式下，fanuc 机械臂作为客户端，需要配置客服端标签页的 C2。

1. 在示教器上按 MENU 键，进入 设置-> 主机通讯。
2. SHIFT + DISPALY 设置为双画面。
3. 按示教器上的 DISP 切到左侧界面点击 显示，选择客户端。进入客户端标签页。

4. 进入 C2 并按以下步骤进行修改
    - 如果当前状态为 已定义，则点击 动作改为 未定义；
    - 如果当前状态为 已开始，则点击 动作 -> 停止，然后再改为 未定义；
    - 如果修改状态失败，尝试重启机器人后再次修改；
    - 设定 协议为 SM；



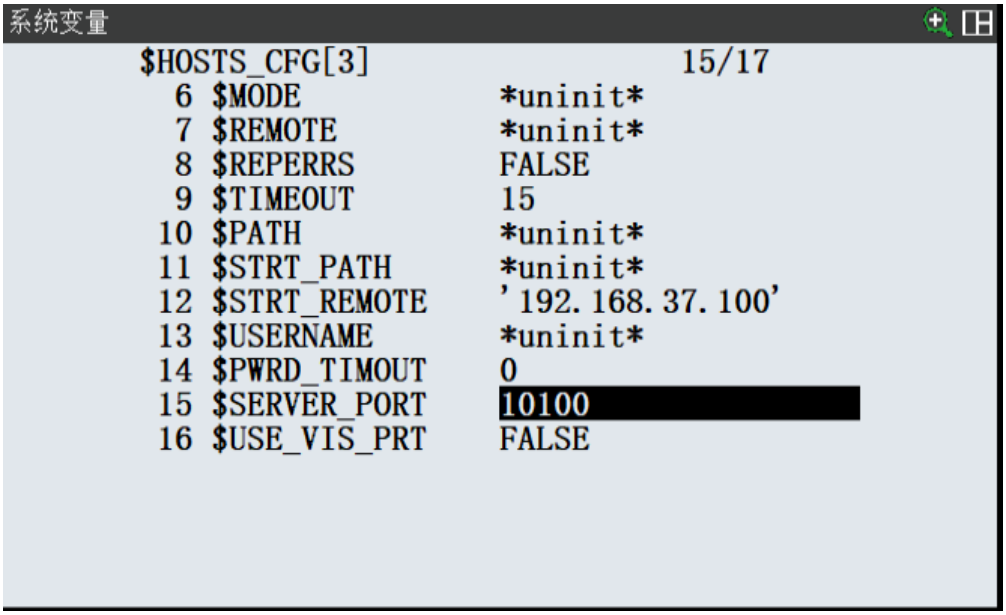


图 43: fanuc 设定工控机主控 socket 6

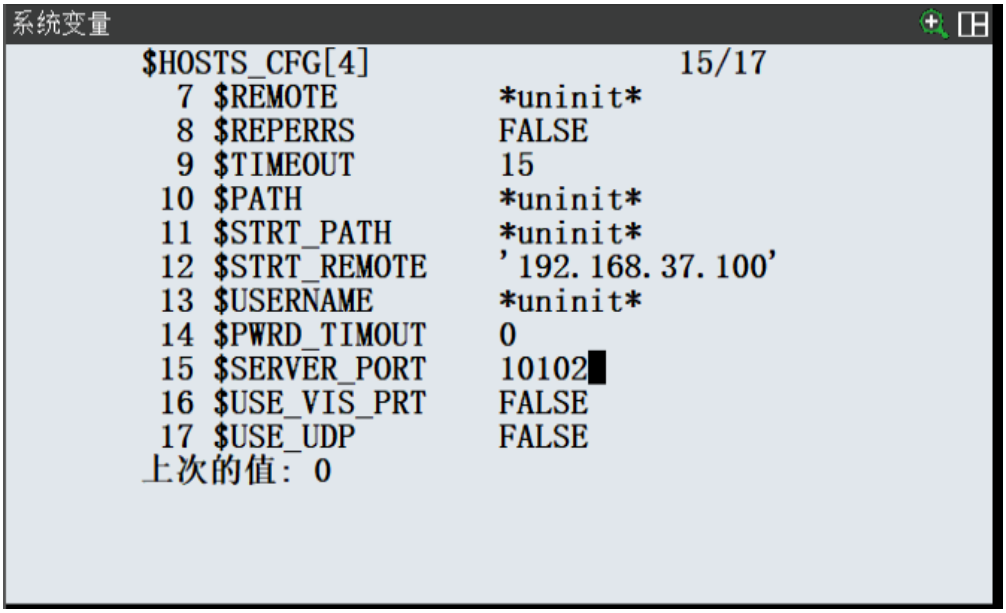


图 44: fanuc 设定工控机主控 socket 7

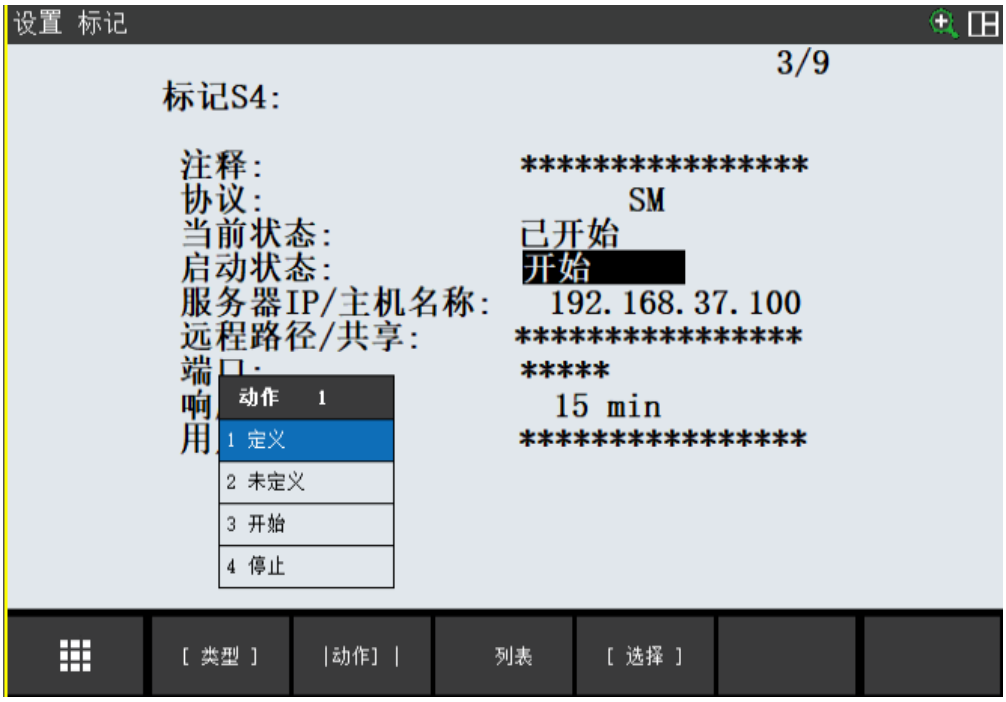


图 45: fanuc 设定工控机主控 socket 8

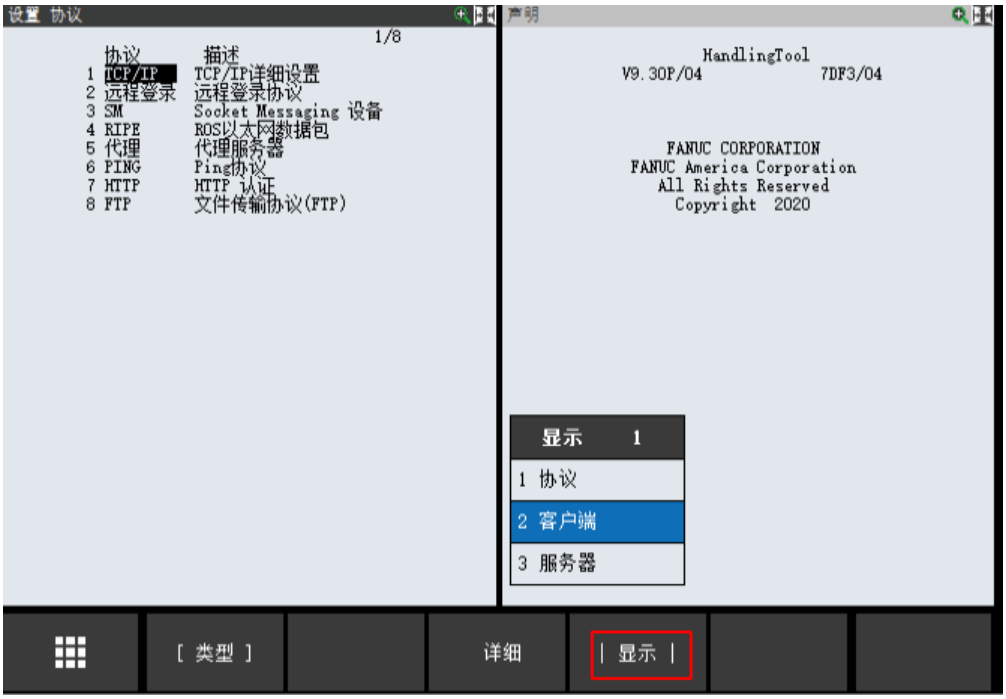


图 46: fanuc 设定机械臂机械臂主控 socket 1

- 修改 服务器 IP / 主机名称为: 192.168.37.101。

工控机的 IP 地址默认为 192.168.37.101，可根据项目需求改成其他 IP 地址。

5. 点击 MENU, 进入 系统-> 下页-> 变量，找到并进入 $HOSTC_CFG （使用 SHIFT + 上下方向键可快速翻页），将 $HOSTC_CFG[2] 中的 $SERVER_PORT 修改为 11111；

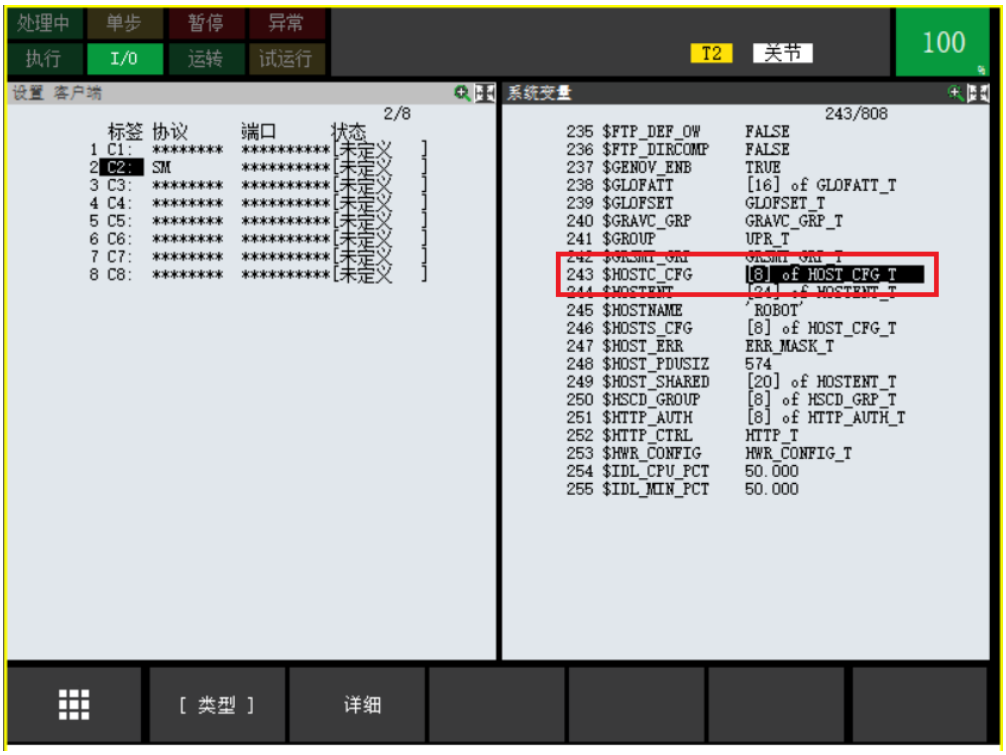
6. 修改完成后，回到客户端标签页，移动光标选中 C2 并按如下步骤操作。
   - 点击 动作选中 定义按 ENTER 将状态改为 定义
   - 点击 动作选中 开始按 ENTER 将状态改为 开始
   - 最后进入 C2 查看 当前状态和 启动状态都应该被设置成了 开始。

使用 U 盘导入程序

1. 准备一个文件系统为 FAT32 格式的 U 盘 (其他格式的文件系统可能不会被 Fanuc 控制器识别)
2. 将 Fanuc 程序.PC 与.TP 文件拷贝到 U 盘，建议全部放在一个文件夹，并且该文件夹没有其他文件
3. U 盘插入示教器或控制柜的 USB 接口
4. 示教器上 MENU-> 文件 -> 文件

5. 可通过 工具 F5 → 切换设备来选择插口：如果插在示教器上，则选 TP 上的 USB(UT1：)，如果插在控制柜上，则选 USB 盘 (UD1：)。

图 47: fanuc 设定机械臂机械臂主控 socket 2



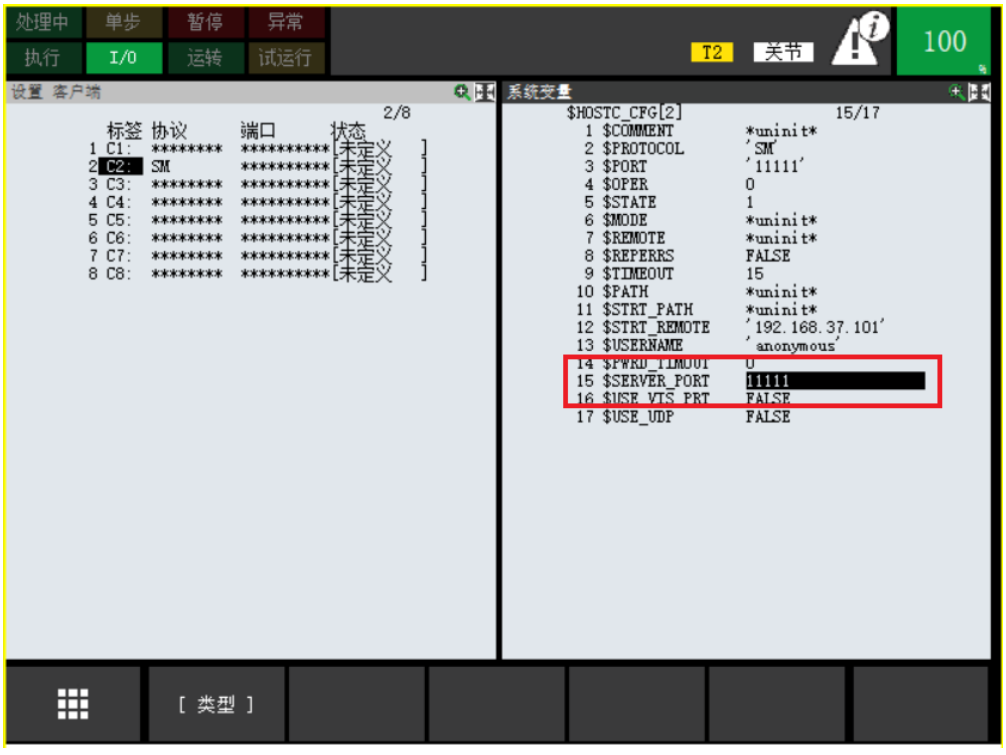


图 48: fanuc 设定机械臂机械臂主控 socket 3

图 49: fanuc 设定机械臂机械臂主控 socket 4

6. 可通过 目录 F2 来选择要显示的文件类型，通常选择 *.* ，来看到所有的文件与文件夹，选中你所存放 Fanuc PC 和 TP 程序的文件夹并进入。
7. 选中对应 PC 及 TP 的文件，或 SHIFT + 上下方向键快速翻页找到 所有文件并选中，按下 加载 F3 加载选中的文件或者加载所有文件。
8. 如果导入程序时提示是否覆盖已有程序，可以根据实际情况选择是否覆盖。如果导入程序时提示程序被占用，可以点击 SELECT，选择任意另一个不会被覆盖的程序，再次尝试导入。

至此，程序已被全部导入到机器人控制器中。

运行程序

通讯说明

工控机和 fanuc 机械臂的通讯方式为 socket：

工控机主控：工控机作为 socket client(客户端)，机械臂作为 socket server(服务端)。

机械臂主控：工控机作为 socket server(服务端)，机械臂作为 socket client(客户端)。


处理中
单步
暂停
异常
执行
I/O
运转
试运行
T2
关节
100
文件
UD1:*.*
1/32
MENU 1
1 实用工具
2 试运行
3 手动操作
4 报警
5 I/O
6 设置
7 文件
8
9 用户
0 -- 下页 --
文件 1
1 文件
2 文件存储器
3 软件安装
4 自动备份
声明
HandlingTool
V9.30P/04
7DF3/04
FANUC CORPORATION
FANUC America Corporation
All Rights Reserved
Copyright 2020
菜单收藏夹 (press and hold to set)
PREV
MENU
SELECT
EDIT
DATA
FCTN
SHIFT
NEXT

图 50: U 盘导入程序 1

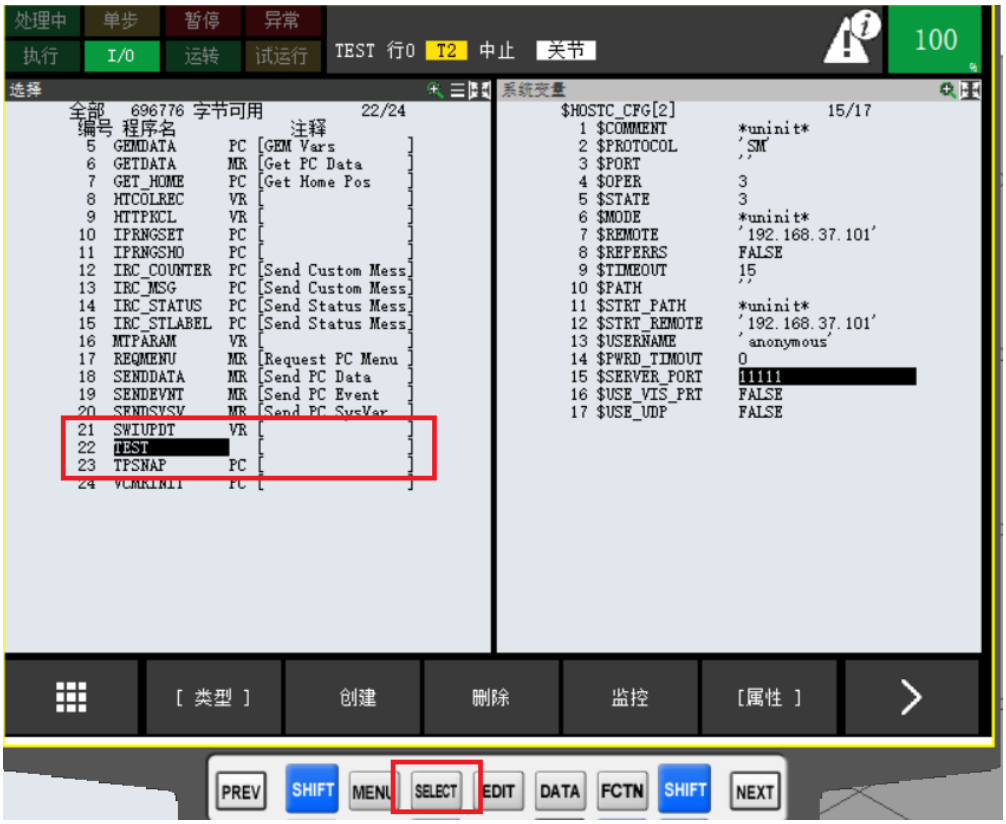

处理中
单步
暂停
异常
执行
I/O
运转
试运行
TEST 行0 T2 中止 关节
100
选择
全部 696776 字节可用 22/24
编号 程序名 注释
5 GEMDATA PC [GEM Vars]
6 GETDATA MR [Get PC Data]
7 GET_HOME PC [Get Home Pos]
8 HTCOLREC VR [ ]
9 HTTPKCL VR [ ]
10 IPRNGSET PC [ ]
11 IPRNGSHO PC [ ]
12 IRC_COUNTER PC [Send Custom Mess]
13 IRC_MSG PC [Send Custom Mess]
14 IRC_STATUS PC [Send Status Mess]
15 IRC_STLABEL PC [Send Status Mess]
16 MTPARAM VR [ ]
17 REQMENU MR [Request PC Menu]
18 SENDDATA MR [Send PC Data]
19 SENDEVNT MR [Send PC Event]
20 SENDSYSV MR [Send PC SysVar]
21 SWIUPDT VR [ ]
22 TEST [ ]
23 TPSNAP PC [ ]
24 VCMKINIT PC [ ]
系统变量
$HOSTC_CFG[2] 15/17
1 $COMMENT *uninit*
2 $PROTOCOL 'SM'
3 $PORT ''
4 $OPER 3
5 $STATE 3
6 $MODE *uninit*
7 $REMOTE '192.168.37.101'
8 $REPERRS FALSE
9 $TIMEOUT 15
10 $PATH ''
11 $STRT_PATH *uninit*
12 $STRT_REMOTE '192.168.37.101'
13 $USERNAME 'anonymous'
14 $PWRD_TIMOUT 0
15 $SERVER_PORT 1111
16 $USE_VIS_PRT FALSE
17 $USE_UDP FALSE
[类型]
创建
删除
监控
[属性]
PREV
SHIFT
MENU
SELECT
EDIT
DATA
FCTN
SHIFT
NEXT

图 51: U 盘导入程序 2

启动程序

运行工控机主控

打开工控机上的 XYZ Studio Max，加载已经配置的项目或者重新配置好一个项目后，在虚拟示教器界面上击连接机械臂，然后按以下步骤操作

- 将示教器档位切到自动
- 将控制柜档位切到自动
- 按下 Fancu 示教器上的 `FCTN -> 1 中止程序`，中止之前未关闭的程序
- 选择 tp 程序中的 **XYZ**, 并确保指针在程序的第一行
- 按下控制柜的上电按钮，运行程序

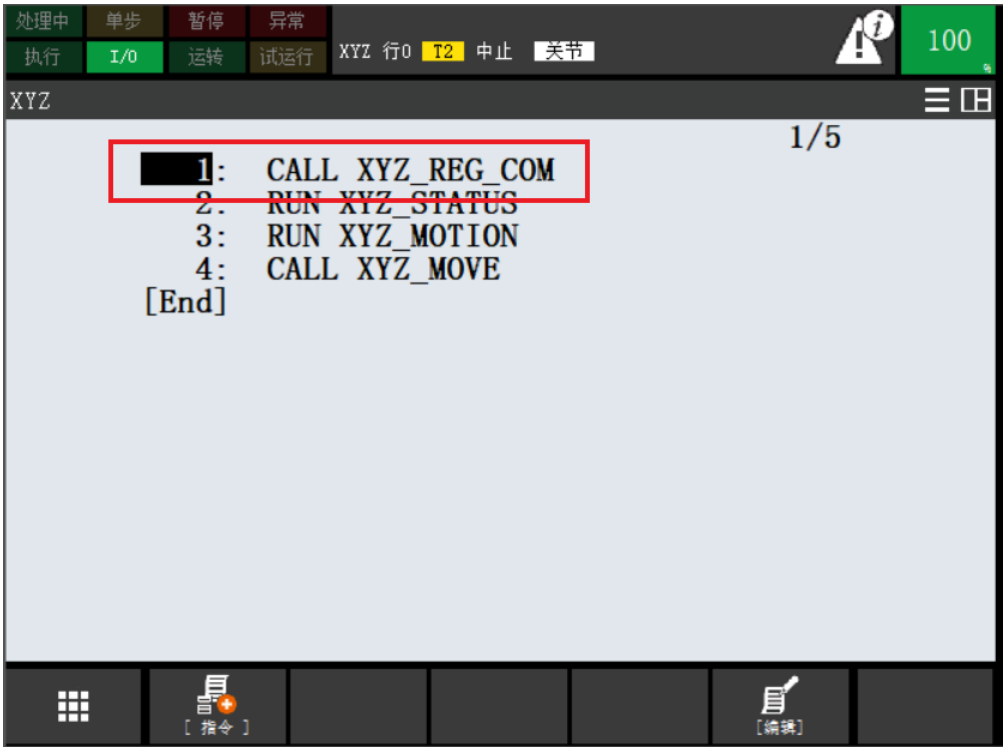


图 52: 运行程序

运行机械臂主控

在工控机的 robot_server 启动后：

- 将示教器档位切到自动
- 将控制柜档位切到自动
- 根据项目需求选择需要运行的 tp 程序，下图中选择的是 TrajMove 异步模板程序
- 按下控制柜的上电按钮，运行程序


处理中
单步
暂停
异常
执行
I/O
运转
试运行
XYZ 行0
T2
中止
关节
100
选择
全部 684128 字节可用
26/71
编号 程序名
注释
25 TPSNAP PC [ ]
26 TRAJMOVE_ASY> [ ]
27 TRAJMOVE_SYNC [ ]
28 VCMRINIT PC [ ]
29 XYZ [XYZ]
30 XYZMOTION [XYZMOTION]
31 XYZSTATUS [XYZSTATUS]
32 XYZ_BKGD PC [backgroud]
33 XYZ_CAPIMG PC [capture image]
34 XYZ_CUS_REQ PC [custom request]
TRAJMOVE_ASYNC
[类型]
创建
删除
监控
[属性]

API 说明

fanuc 工控机主控支持的 API

指令号	命令	支持情况
100	获取机械臂程序版本号	支持
101	发送速度数据	支持
102	发送加速度数据	支持
103	发送圆滑过渡参数	支持
104	发送工具坐标系 (TCP) 参数	支持
105	设置数字量输出	支持
106	SetJointsMovej	支持
107	SetCartMovel	支持
108	SetJoinsMovel	支持
109	SetCartMovej	支持
110	MovejSequence	支持
111	MovelSequence	支持
112	SetJointsMovejDo	不支持
113	SetCartMovelDo	不支持
114	SetJointsMovelDo	不支持
115	SetJointsMovejGroupDo	不支持
116	SetCartMovelGroupDo	不支持
117	SetJointsMovelGroupDo	不支持
118	MovelUntil	支持
119	获取数字量输入状态	支持
120	获取数字量输出状态	支持
121	获取模拟量输入口数值	不支持
122	获取机械臂当前角度	支持
123	获取机械臂当前位姿	支持
200	机械臂后台发送状态	支持

fanuc 机械臂主控支持的 API

RUN XYZ_BKGD

运行后台程序

后台程序会和工控机进行通信连接和收发数据。连接是否成功的的状态会写到第五个数值寄存器中，成功会给寄存器赋值为 1，失败则赋值为 0。通信过程中，如果存在收发失败会给第六个数字寄存器赋值，收发数据失败会赋值为 0，收发数据成功则会赋值为 1。每一条指令均会和工控机进行一次收发动作。

参数 none –无，状态均直接反馈到数值寄存器

R[5] 表示连接是否成功

R[6] 表示通信是否有异常

CALL XYZ_ABT_BKGD

中止后台程序 XYZ_BKGD 的运行

中止时也会关闭后台套接字和工控机的连接。

CALL XYZ_SW_APP('app_name')

切换应用

参数 app_name (*string*) –应用名称

CALL XYZ_SW_ITEM(ws_id,'item_codename')

切换工件

参数

- **ws_id** (*INT*) –工作空间 id
- **item_codename** (*string*) –工件名称

CALL XYZ_SW_FLW('flw_name')

切换 flow

参数 flw_name –flow 名称

CALL XYZ_SW_TOOL('tool_name')

切换工具

参数 tool_name (*string*) –工具名称

CALL XYZ_R_CAPIMG(vision_service_id)

请求拍照

参数 vision_service_id (*INT*) –视觉服务 id

CALL XYZ_G_CAPIMG(R[11])

获取拍照结果

参数 R[11] –请求拍照时返回的 token 寄存器地址

CALL XYZ_CAPIMG(vision_service_id)

拍照

参数 vision_service_id (*INT*) –视觉服务 id

CALL XYZ_R_G_POSE(ws_id)

请求抓取位姿

参数 ws_id (*INT*) –需要获取抓取点位的工作空间 id

CALL XYZ_G_G_POSE(R[11])

获取抓取位姿

参数 R[11] –求抓取目标点位时返回的 token 寄存器地址

CALL XYZ_R_O_POSE(ws_id)

请求物体位姿

参数 ws_id (*INT*) –需要获取物体位姿的工作空间 id

CALL XYZ_G_O_POSE(R[11])

获取物体位姿

参数 R[11] –请求物体位姿时得到的 token 寄存器地址

CALL XYZ_RESET_V(ws_id)

重置视觉

参数 ws_id (*INT*) –需要重置视觉的工作空间 id

CALL XYZ_S_JOINTS(rgx_number)

发送机械臂当前角度：j1~j6: 机械臂当前的角度信息，如果机械臂轴数小于 6 轴，则需要对应位置补零，依旧发送六个数。

参数 rgx_number (*INT*) –指定存储当前机械臂关节角度的寄存器地址，用来传给后台发送。可以任意指定，比如 50

CALL XYZ_S_C_POSE(rgx_number)

发送机械臂法兰当前位姿

参数 rgx_number (*INT*) –指定存储当前机械臂法兰位姿的寄存器地址，用来传给后台发送。可以任意指定，比如 51

CALL XYZ_S_EXT_J(rgx_number)

发送机械臂当前扩展轴位置：j1~j6: 机械臂当前扩展轴的角度信息，如果扩展轴数不足 6 的, 需要补零后发送六个数。

参数 rgx_number (*INT*) –指定存储当前机械臂扩展轴位置的寄存器地址，用来传给后台发送。可以任意指定，比如 52

CALL XYZ_R_PICK

请求 pick 动作规划

CALL XYZ_R_PLACE

请求 place 动作规划

CALL XYZ_R_PICPLA

请求 pick 和 place 规划

CALL XYZ_G_PICIN

获取取料入筐轨迹

CALL XYZ_G_PICOUT

获取取料出筐轨迹

CALL XYZ_G_PLAIN

获取放料入筐轨迹

CALL XYZ_G_PLAOUT

获取放料出筐轨迹

CALL XYZ_SW_STRAT('strat1')

请求切换策略

参数 strat1 –策略名称

Type string

CALL XYZ_U_T_POSE

料箱重定位

CALL XYZ_U_OBJ_OH

工件在上手的二次定位

CALL XYZ_U_OBJ_TH

工件不在手上的二次定位

CALL XYZ_G_OBJ_PT

获取工件姿态类型

CALL XYZ_RESET_PS

重置工业码垛状态

CALL XYZ_EXECTRAJ(traj_flag)

执行轨迹。

参数 traj_flag–执行轨迹标志位若为 2，执行抓取入筐；若为 3，执行抓取出筐；若为 4，执行放置入筐；若为 5，执行放置出筐。

CALL XYZ_C_G_POSE(ws_id)

计算抓取位姿

参数 ws_id(*INT*)–工作空间 id

CALL XYZ_C_O_POSE(ws_id)

计算物体位姿

参数 ws_id(*INT*)–工作空间 id

CALL XYZ_USR_CMD()

自定义请求

参数 none–无，所有输入输出均直接修改寄存器，寄存器的使用情况如下：

SR[6]~SR[10] 为下位机发送的字符串，不能为空

R[70]~R[79] 为下位机发送的整数

R[80]~R[89] 为下位机发送的浮点数

PR[2] 为下位机发送的一组关节角

PR[6] 为下位机发送的一个笛卡尔坐标

SR[1]~SR[5] 为下位机接收的字符串

R[50]~R[59] 为下位机接收的整数

R[60]~R[69] 为下位机接收的浮点数

PR[1] 为下位机接收的一组关节角

PR[5] 为下位机接收的一个笛卡尔坐标

案例/模板说明

机械臂主控主函数说明

以下为机械臂主控模板代码，包含坐标移动基础模板, 坐标移动二次定位模板和轨迹移动同步模板，轨迹移动异步模板。

坐标移动基础模板

```
!S1: 初始化 ;
CALL XYZ_REG_COM    ;
RO[1]=OFF ;
 ;
 ;
!S2: 连接到工控机;
RUN XYZ_BKGD ;
WAIT   1.00(sec) ;
  ;
```

(下页继续)

(续上页)

```
  IF (R[5]<>1) THEN ;
  !如果没连接到工控机，示教器会报警提示 ;
  UALM[5] ;
  ENDIF ;
   ;
   !切换任务流图 ，默认被注释掉，如果需求可以取消注释
  //CALL XYZ_SW_FLW('cart_basic.t') ;
  //CALL XYZ_CK_ERR    ;
   ;
  !S3: 切换成当前工件 ;
  CALL XYZ_SW_ITEM(0,'item1') ;
  CALL XYZ_CK_ERR    ;
   ;
  ! S4: 运动到 home 位 ;
  ! 首先移动到 home 点。注意需要先定义 home_pose 的关节角度值;
  ! 或者用使用其他点来替代 ;
J P[1:home pose] 100% CNT100    ;
   ;
  !如果是眼在机械臂上的应用形式 ;
  !请修改 R[99]=1 ;
  R[99]=0    ;
   ;
  LBL[1] ;
   ;
  !S5: 眼在机械臂上 ;
  IF (R[99]=1) THEN ;
  !S6: 运动到拍照位姿 ;
  !注意需要先定义 scan_pose 的位姿 ;
  !或者用使用其他点来替代 ;
  !scan pose 必须定义在位置寄存器里，不然后台读取不到 ;
L PR[99] 100mm/sec FINE    ;
  !S7: 发送拍照位姿 ;
  CALL XYZ_S_C_POSE(99) ;
  CALL XYZ_CK_ERR    ;
  ENDIF ;
   ;
  !S8: 请求抓取点位 ;
  CALL XYZ_R_G_POSE(0) ;
  CALL XYZ_CK_ERR    ;
   ;
  !S9: 获取抓取点位 ;
  R[98]=R[11:MAST parsed int]    ;
  CALL XYZ_G_G_POSE(R[98]) ;
  CALL XYZ_CK_ERR    ;
  IF (R[12:MAST pose num]<1) THEN ;
  !工作空间中没有工件 ;
  WAIT   5.00(sec) ;
  JMP LBL[1] ;
  ENDIF ;
   ;
  !S10: 运动到抓取点并抓取工件 ;
  !这里需要添加过渡点，防止机器人运动过程中发生碰撞 ;
L PR[5:MAST cart pose] 100mm/sec FINE    ;
  RO[1]=ON ;
   ;
  !S11: 运动到放置点并放置工件  ;
```

(下页继续)

(续上页)

```
  !这里需要添加过渡点，防止机器人运动过程中发生碰撞 ;
L P[2:place pose] 100mm/sec FINE    ;
  RO[1]=OFF ;
   ;
  JMP LBL[1] ;
   ;
```

坐标移动重定位模板

```
 !S1: 初始化 ;
 CALL XYZ_REG_COM    ;
 RO[1]=OFF ;
 RO[2]=OFF ;
  ;
 !S2: 连接到工控机 ;
 RUN XYZ_BKGD ;
 WAIT   1.00(sec) ;
   ;
  IF (R[5]<>1) THEN ;
  !如果没连接到工控机，示教器会报警提示 ;
  UALM[5] ;
  ENDIF ;
   ;
   !切换任务流图 ，默认被注释掉，如果需求可以取消注释
  //CALL XYZ_SW_FLW('cart_repo.t') ;
  //CALL XYZ_CK_ERR    ;
   ;
  !S3: 机械臂外的相机切换成识别当前工件;
  CALL XYZ_SW_ITEM(0,'item1') ;
  CALL XYZ_CK_ERR    ;
   ;
  !S4: 运动到home pose ;
  ! 首先移动到 home 点。注意需要先定义 home_pose 的关节角度值;
  ! 或者用使用其他点来替代 ;
J P[1:home pose] 100% CNT100    ;
   ;
  LBL[1] ;
   ;
  !S5: 请求工件的抓取位姿 ;
  CALL XYZ_R_G_POSE(0) ;
  CALL XYZ_CK_ERR    ;
   ;
  !S6: 获取工件的抓取位姿 ;
  R[98]=R[11:MAST parsed int]    ;
  CALL XYZ_G_G_POSE(R[98]) ;
  CALL XYZ_CK_ERR    ;
   ;
  !S7: 处理是否识别到工件;
  IF (R[12:MAST pose num]<1) THEN ;
   ;
  !S15: 没有识别到工件，机械臂外的相机切换识别隔板;
  CALL XYZ_SW_ITEM(0,'board') ;
  CALL XYZ_CK_ERR    ;
   ;
```

(下页继续)

(续上页)

```
  !S16: 请求隔板抓取位姿 ;
  CALL XYZ_R_G_POSE(0) ;
  CALL XYZ_CK_ERR    ;
   ;
  !S17: 获取隔板的抓取位姿 ;
  R[98]=R[11:MAST parsed int]    ;
  CALL XYZ_G_G_POSE(R[98]) ;
  CALL XYZ_CK_ERR    ;
  IF (R[12:MAST pose num]<1) THEN ;
  !no board.maybe tote empty ;
  JMP LBL[2]
  ENDIF ;
   ;
  !S18: 运动到隔板抓取位姿, 并抓取隔板 ;
  !这里需要添加过渡点, 防止机器人运动过程中发生碰撞 ;
L PR[5:MAST cart pose] 100mm/sec FINE    ;
  RO[1]=ON ;
   ;
  !S19: 运动到隔板的放置位姿 ;
  !这里需要添加过渡点, 防止机器人运动过程中发生碰撞 ;
L P[3:board place pose] 100mm/sec FINE    ;
  RO[1]=OFF ;
   ;
  !S20: 机械臂外的相机切换成识别当前工件;
  CALL XYZ_SW_ITEM(0,'item1') ;
  CALL XYZ_CK_ERR    ;
   ;
  ELSE ;
   ;
  !S8: 运动到拍照位姿 ;
  !注意需要先定义 scan_pose 的位姿 ;
  !或者用使用其他点来替代 ;
L PR[99] 100mm/sec FINE    ;
   ;
  !S9: 发送拍照位姿 ;
  CALL XYZ_S_C_POSE(99) ;
  CALL XYZ_CK_ERR    ;
   ;
  !S10: 机械臂上的相机切换成识别当前工件;
  CALL XYZ_SW_ITEM(1,'item1') ;
  CALL XYZ_CK_ERR    ;
   ;
  !S11: 请求机械臂上的相机获取工件抓取位姿 ;
  CALL XYZ_R_G_POSE(1) ;
   ;
  !S12: 获取工件的抓取位姿 ;
  R[98]=R[11:MAST parsed int]    ;
  CALL XYZ_G_G_POSE(R[98]) ;
   ;
  IF (R[12:MAST pose num]<1) THEN ;
  JMP LBL[2] ;
  ENDIF ;
   ;
  !S13: 运动到工件的抓取位姿并进行抓取 ;
  !这里需要添加过渡点, 防止机器人运动过程中发生碰撞 ;
L PR[5:MAST cart pose] 100mm/sec FINE    ;
```

(下页继续)

(续上页)

```
  RO[2]=ON ;
  ;
  !S14: 运动到工件的放置位姿 ;
  !这里需要添加过渡点，防止机器人运动过程中发生碰撞 ;
L P[4:object place pos] 100mm/sec FINE    ;
  RO[2]=OFF ;
  ;
  ENDIF ;
  ;
  JMP LBL[1] ;
  ;
  !处理未识别到工件的情况 ;
  LBL[2] ;
  ;
```

轨迹移动同步模板

```
  ;
 !S1: 初始化 ;
 CALL XYZ_REG_COM    ;
 RO[1]=OFF ;
  ;
 !S2: 连接到工控机 ;
 RUN XYZ_BKGD ;
 WAIT   1.00(sec) ;
  ;
  IF (R[5]<>1) THEN ;
  !如果没连接到工控机，示教器会报警提示 ;
  UALM[5] ;
  ENDIF ;
   ;
   !切换任务流图 ，默认被注释掉，如果需求可以取消注释
  //CALL XYZ_SW_FLW('traj_async.t') ;
  //CALL XYZ_CK_ERR    ;
   ;
  !S3: 切换成当前工件 ;
  CALL XYZ_SW_ITEM(0,'item1') ;
  CALL XYZ_CK_ERR    ;
   ;
  !S4: 运动到home pose ;
  ! 首先移动到 home 点。注意需要先定义 home_pose 的关节角度值;
  ! 或者用使用其他点来替代 ;
J P[1:home pose] 100% FINE    ;
   ;
  LBL[1] ;
   ;
  !S5: 请求抓取和放置规划 ;
  CALL XYZ_R_PICPLA(0) ;
  CALL XYZ_CK_ERR    ;
   ;
  !S6: 获取 pick-in 轨迹;
  CALL XYZ_G_PICIN(0) ;
  CALL XYZ_CK_ERR    ;
   ;
```

(下页继续)

(续上页)

```
!S7: 判断是否识别到工件;
IF (R[12:MAST pose num]<>0) THEN ;
!no object ;
JMP LBL[2] ;
ENDIF ;
 ;
!S8: 执行 pick-in 轨迹;
CALL XYZ_EXECTRAJ(2) ;
RO[1]=ON ;
 ;
!S9: 请求 pick-out 轨迹;
CALL XYZ_G_PICOUT(0) ;
CALL XYZ_CK_ERR    ;
 ;
!S10: 执行 pick-out 轨迹;
CALL XYZ_EXECTRAJ(3) ;
 ;
!S11: 获取 place-in 轨迹;
CALL XYZ_G_PLAIN(0) ;
CALL XYZ_CK_ERR    ;
 ;
!S12: 执行 place-in 轨迹;
CALL XYZ_EXECTRAJ(4) ;
 ;
!13: 获取 place-out 轨迹;
CALL XYZ_G_PLAOUT(0) ;
CALL XYZ_CK_ERR    ;
 ;
!S14: 执行 place-out 轨迹;
CALL XYZ_EXECTRAJ(5) ;
JMP LBL[1] ;
 ;
!处理未识别到工件的情况 ;
LBL[2] ;
 ;
```

轨迹移动异步模板

```
!S1: 初始化 ;
CALL XYZ_REG_COM    ;
RO[1]=OFF ;
 ;
!S2: 连接到工控机 ;
RUN XYZ_BKGD ;
WAIT   1.00(sec) ;
 ;
IF (R[5]<>1) THEN ;
!如果没连接到工控机，示教器会报警提示 ;
UALM[5] ;
ENDIF ;
 ;
 !切换任务流图 ，默认被注释掉，如果需求可以取消注释
//CALL XYZ_SW_FLW('traj_sync.t') ;
//CALL XYZ_CK_ERR    ;
```

(下页继续)

(续上页)

```
   ;
  !S3: 切换成当前工件 ;
  CALL XYZ_SW_ITEM(0,'item1') ;
  CALL XYZ_CK_ERR    ;
   ;
  !S4: 运动到home pose ;
  ! 首先移动到 home 点。注意需要先定义 home_pose 的关节角度值;
  ! 或者用使用其他点来替代 ;
J P[1:home pose] 100% CNT100    ;
   ;
  !S5: 请求抓取和放置规划 ;
  CALL XYZ_R_PICPLA(0) ;
  CALL XYZ_CK_ERR    ;
   ;
   ;
  LBL[1] ;
   ;
  !S6: 获取 pick-in  估计;
  CALL XYZ_G_PICIN(0) ;
  CALL XYZ_CK_ERR    ;
   ;
  !S7 判断是否识别到工件;
  IF (R[12:MAST pose num]=0) THEN ;
  !no object ;
  JMP LBL[2] ;
  ENDIF ;
   ;
  !S8: 执行 pick-in 轨迹;
  CALL XYZ_EXECTRAJ(2) ;
  RO[1]=ON ;
   ;
  !S9: 获取 pick-out 轨迹;
  CALL XYZ_G_PICOUT(0) ;
  CALL XYZ_CK_ERR    ;
   ;
  !S10: 执行 pick-out 轨迹;
  CALL XYZ_EXECTRAJ(3) ;
   ;
  !S11: 请求下一次的抓取和放置规划 ;
  CALL XYZ_R_PICPLA(0) ;
  CALL XYZ_CK_ERR    ;
   ;
  !S12: 获取本次 place-in 轨迹;
  CALL XYZ_G_PLAIN(0) ;
  CALL XYZ_CK_ERR    ;
   ;
  !S13: 执行本次 place-in 轨迹;
  CALL XYZ_EXECTRAJ(4) ;
   ;
  !S14: 获取本次 place-out 轨迹;
  CALL XYZ_G_PLAOUT(0) ;
  CALL XYZ_CK_ERR    ;
   ;
  !S15: 执行本次 place-out 轨迹;
  CALL XYZ_EXECTRAJ(5) ;
  JMP LBL[1] ;
```

(下页继续)

(续上页)

```
 ;
!处理未识别到工件的情况 ;
LBL[2] ;
 ;
```

常见问题

- 如果 FANUC 系统软件版本要早于 8.9，可能存在兼容性问题，目前已知的解决方案为，将所有函数名、变量名改到 12 位以内，重新编译即可。
- FANUC 的寄存机定义见附录

附录

寄存器含义

类型	位数	含义
R	1	角速度（工控机主控）
R	2	线速度（工控机主控）
R	3	加速度（工控机主控）
R	4	计数位（工控机主控）
R	5	连接工控机是否成功 (机械臂主控)
R	10	错误代码（机械臂主控）
R	11	工控机发来的整数，例如 token、posetype（机械臂主控）
R	12	位姿数量（机械臂主控）
R	20	位姿类型的第一位（机械臂主控，如果多于一个位姿，R21 及后面的寄存器存有剩余的位姿类型）
PR	1	关节位姿（工控机主控）
PR	5	笛卡尔位姿（工控机主控）
PR	10	位姿的第一位（机械臂主控，如果多于一个位姿，PR11 及后面的寄存器存有剩余的位姿）

14.2.8 iNexBot

此处介绍安装 iNexBot 机械臂驱动的相关事项。

驱动版本和使用要求

支持的机械臂类型

可经 inexbot 控制器配置的机械臂

控制器型号

iNexBot 控制器

机械臂需要开通的功能

- 自带 Socket 功能，无需额外开通
- 控制器最低支持版本：rtl2-22.07.05-2023061416
- 示教器最低支持版本：rtl-22.07.05-2023061519

  可进入示教器 设置 -> 系统设置-> 版本升级, 查看版本：

图 53: 示教器查看版本

安装驱动

iNexBot 机械臂驱动文件列表

代码可从 MAX 安装目录下的 share/robot_code 中获取。

```
├── lua
│   └── xyz_status.lua
└── robotJob
    └── R1
        ├── AxyzCartMoveBas.JBR 座标移动基础模板
        ├── AxyzCartRepo.JBR 座标移动二次定位模板
        ├── AxyzMasterTest.JBR 机械臂主控 api 测试程序
        ├── AxyzMotion.JBR 工控机主控程序
        ├── AxyzTrajAsync.JBR 轨迹移动异步模板
        ├── AxyzTrajSync.JBR 轨迹移动同步模板
```

├── NxyzGetCartPose.JBR **以下以 N 开头的函数，均为工控机主控函数，不能调用**
├── NxyzGetCurJoints.JBR
├── NxyzGetDI.JBR
├── NxyzGetVersion.JBR
├── NxyzMotionParse.JBR
├── NxyzMovejSeq.JBR
├── NxyzMovelSeq.JBR
├── NxyzMovelUntil.JBR
├── NxyzSetAcc.JBR
├── NxyzSetCartMovej.JBR
├── NxyzSetCartMovel.JBR
├── NxyzSetDO.JBR
├── NxyzSetJntsMovej.JBR
├── NxyzSetJntsMovel.JBR
├── NxyzSetSpeed.JBR
├── NxyzSetTool.JBR
├── NxyzSetZone.JBR
├── RobotResetProgram.ResetPro
├── xyzCalGraspPose.JBR **以下以 xyz 开头的函数，均为机械臂主控函数，可以调用**
├── xyzCalObjPose.JBR
├── xyzCapImg.JBR
├── xyzExeTraj.JBR
├── xyzGetCapImg.JBR
├── xyzGetGraspPose.JBR
├── xyzGetObjPose.JBR
├── xyzGetObjType.JBR
├── xyzGetPickIn.JBR
├── xyzGetPickOut.JBR
├── xyzGetPlaceIn.JBR
├── xyzGetPlaceOut.JBR
├── xyzNormalParse.JBR
├── xyzReqCapImg.JBR
├── xyzReqGraspPose.JBR
├── xyzReqObjPose.JBR
├── xyzReqPick.JBR
├── xyzReqPickPlace.JBR
├── xyzReqPlace.JBR
├── xyzResetPalStat.JBR
├── xyzResetTask.JBR
├── xyzSendCurCart.JBR
├── xyzSendCurJoints.JBR
├── xyzSendExtJoints.JBR
├── xyzStatus.JBPG
├── xyzSwitchApp.JBR
├── xyzSwitchFlow.JBR
├── xyzSwitchItem.JBR
├── xyzSwitchStrat.JBR

```
├── xyzSwitchTool.JBR
├── xyzUpdObjOnHand.JBR
├── xyzUpdObjToHand.JBR
└── xyzUpdTotePose.JBR
```

设定机械臂 IP

在示教器，将操作用户改为管理员，密码为“123456”。

进入 设置 -> 系统设置-> ip 设置，修改网口 1 ip，修改控制器 ip。为降低出错可能，建议使用默认 ip，注意工控机也要将 ip 改为相同网段（192.168.1.101）。

- 网口 1 “控制器 IP”设置为：192.168.1.13
- 本机 ip 设为：192.168.1.102 或同网段其他值

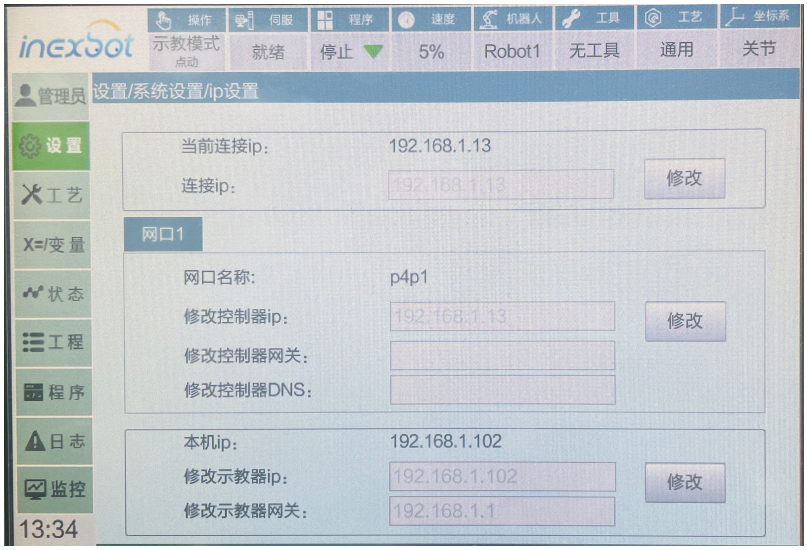


图 54: iNexBot 设定机械臂 IP

网络设置完成后重启机械臂生效。

配置 SOCKET 通信

打开 设置 -> TCP 通讯设置：

- 机械臂主控使用的是工艺号 1，修改方式为 客户端, IP 为 192.168.1.101，端口号为 11111, 帧头为空，分割符为 “,”，结束符为 “#”，进制为十进制。
- 工控机主控使用的工艺号 2，修改方式为 客户端, IP 为 192.168.1.101，端口号为 5000, 帧头为空，分割符为 “,”，结束符为 “#”，进制为十进制。
- 工控机主控还需要使用 lua 脚本中的 socket，已在 lua 文件定义，无需修改，只需导入 lua 程序即可。

使用 U 盘导入程序

- 将 MAX 安装目录下的 share/robot_code 中 inexbot 的机械臂代码拷入 U 盘，插入示教器 USB 口
- 在示教器上，点击 设置-> 系统设置 -> 导入程序，选择需要导入的文件夹导入。

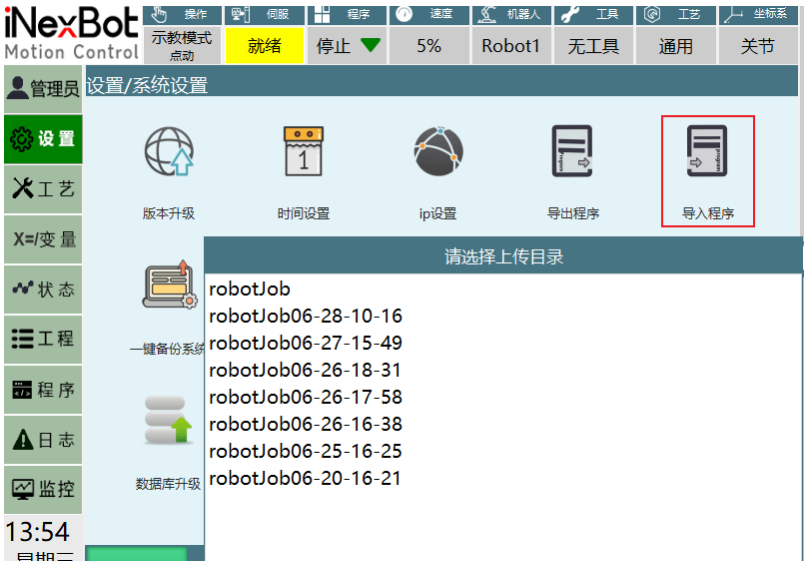


图 55: U 盘导入程序

- 导入 lua 程序

  将 xyz_status.lua 脚本放到 u 盘的 upgrade 文件夹中，或者 pc 版的 updateTmp 中，同样是在 导入程序处，上传 xyz_status.lua 文件。

修改控制器角度显示为角度

- 在示教器上，点击 设置-> 操作参数，将 姿态值`` 改为 `` 角度制。

图 56: 修改控制器角度显示为角度制

运行程序

通讯说明

工控机和 iNexBot 控制器的通讯方式为 socket：工控机作为 socket server(服务端)，机械臂作为 socket client（客户端）。

机械臂主控，使用的是工艺号 1；

工控机主控，使用的是工艺号 2 以及 lua 程序。

启动程序

1. 机械臂主控运行

- 在示教模式下，注意需要 设置 -> 后台任务 -> 全局后台 xyzStatus 任务停止。
- 按照模板，编写好机械臂主控程序后，选择工程中的 Axyz* 程序，打开，切换运行模式，上电运行。

2. 工控机主控运行

- 在示教模式下，注意每次工控机主控运行前，都要将 设置 -> 后台任务 -> 全局后台 xyzStatus 任务 **停止**，然后 **启动**
- 此时，在 MAX 中连接机械臂，然后切换为运行模式，运行工控机主控 AxyzMotion 程序。

3. 注意！后台任务是全局任务的情况下，不设为开机自启，无法启动。

   后台有可能无法停止，需要重启控制器。

寄存器说明

1. 机械臂主控寄存器说明

socket 接收到的数据，是 GD499 开始，GD499 为接收校验位，GD500 为错误代码。

- GS001: app_name 应用名称
- GS002: flow_name 任务名称
- GS003: tool_name 工具名称
- GS004: strat_name 策略名称
- GS005: item_codename 工件代号
- GI001: vision_service_id 视觉服务 ID
- GI002: req_cap_img_token 请求拍照 token
- GI003: ws_id 工作空间 id
- GI004: pose_type 姿态类型
- GP0001: grasp_pose 抓取位姿
- GP0002: obj_pose 工件位姿
- GP0003: tote_pose 料筐位姿

接收轨迹时:

- GD501:pipeline_num pipeline 文件 number

- GD502:register_num 用到的注册文件的注册 number
- GD503:way_point_num 轨迹点数

注意需要提前把全局 GP0001, GP0002, GP0003 改为直角类型

2. 工控机主控寄存器说明

工控机主控 socket 接收数据，是 GD500 开始。

- GB999 允许后台连接的标志位，1 为允许，0 为不允许
- GB998 后台 status socket 连接成功的标志位
- GI001 线速度
- GI002 角速度
- GI003 加速度
- GI004 减速度
- GS001 VERSION 版本 id
- GI005 圆滑等级 zone

API 说明

工控机主控支持的 API

指令号	命令	支持情况
100	获取机械臂程序版本号	支持
101	发送速度数据	支持
102	发送加速度数据	支持
103	发送圆滑过渡参数	支持
104	发送工具坐标系 (TCP) 参数	支持
105	设置数字量输出	支持
106	SetJointsMovej	支持
107	SetCartMovel	支持
108	SetJoinsMovel	支持
109	SetCartMovej	支持
110	MovejSequence	支持
111	MovelSequence	支持
112	SetJointsMovejDo	不支持
113	SetCartMovelDo	不支持
114	SetJointsMovelDo	不支持
115	SetJointsMovejGroupDo	不支持
116	SetCartMovelGroupDo	不支持
117	SetJointsMovelGroupDo	不支持
118	MovelUntil	不支持
119	获取数字量输入状态	支持
120	获取数字量输出状态	支持
121	获取模拟量输入口数值	不支持
122	获取机械臂当前角度	支持
123	获取机械臂当前位姿	支持
200	机械臂后台发送状态	支持

iNexBot 机械臂主控支持的 API

xyzSwitchApp()
切换应用

参数 **GS001** –应用名称

返回 err_code: GD500 为 0 表示成功

xyzSwitchFlow()
切换工件

参数 **GS002** –流图名称

返回 err_code: GD500 为 0 表示成功

xyzSwitchTool()
切换工具

参数 **GS003** –工具名称

返回 err_code: GD500 为 0 表示成功

xyzReqCapImg()
请求拍照

参数 **GI001** (*vision_service_id*) –视觉服务 id

返回

err_code: GD500 为 0 表示成功

cap_img_token: GI002 返回的 token

xyzGetCapImg()
获取拍照结果

参数 **GI002** (*cap_img_token*) –请求拍照时返回的 token

返回 err_code: GD500 为 0 表示成功

xyzCapImg()
拍照

参数 **GI001** (*vision_service_id*) –需要进行拍照操作的工作空间 id

返回 err_code: GD500 为 0 表示成功

xyzReqGraspPose()
请求抓取位姿

参数 **GI003** (*ws_id*) –需要获取抓取点位的工作空间 id

返回

err_code: GD500 为 0 表示成功

grasp_pose_token: GD501: 返回的用于获取目标点位时使用的 token

xyzGetGraspPose()
获取抓取位姿

参数 **GD501** (*grasp_pose_token*) –求抓取目标点位时返回的 token

返回

err_code: GD500 为 0 表示成功

grasp_pose: GD501~GD506 抓取位姿

grasp_pose_num: GD508 可供抓取的点数量

grasp_pose_type: GD509 当前返回的抓取点的 pose 类型

xyzReqObjPose()

请求物体位姿

参数 GI003 (*ws_id*) –需要获取物体位姿的工作空间 id

返回

err_code: GD500 为 0 表示成功

obj_token: GD501 返回的用于物体位姿识别的 token

xyzGetObjPose()

获取物体位姿

参数 GD501 (*obj_pose_token*) –请求物体位姿时得到的 token

返回

err_code: GD500 为 0 表示成功

obj_pose: GD501~GD506 物体位姿

obj_num: GD508 物体数量

obj_pose_type: GD509 当前返回的物体 pose 类型

xyzResetTask()

重置任务:return: err_code: GD500 为 0 表示成功

xyzSendCurrentJoints()

发送机械臂当前角度：j1~j6: 机械臂当前的角度信息，如果机械臂轴数小于 6 轴，则需要对应位置补零，依旧发送六个数。

返回 err_code: GD500 为 0 表示成功

xyzSendCurrentCartPose()

发送机械臂法兰当前位姿

返回 err_code: GD500 为 0 表示成功

xyzSendCurrentExtJoints()

暂不支持。

发送机械臂当前扩展轴位置：j1~j6: 机械臂当前扩展轴的角度信息，如果扩展轴数不足 6 的, 需要补零后发送六个数。

返回 err_code: GD500 为 0 表示成功

xyzReqPick()

暂不支持。请求 pick 动作规划

返回 err_code: GD500 为 0 表示成功

xyzReqPlace()

暂不支持。请求 place 动作规划

返回 err_code: GD500 为 0 表示成功

xyzReqPickPlace()

请求 pick 和 place 规划

参数 **GI003** (*ws_id*) –需要抓取的工作空间 id

返回 err_code: GD500 为 0 表示成功

xyzGetPickin()

获取取料入框轨迹

参数 **GI003** (*ws_id*) –规划空间 id，默认填 0

返回 err_code: GD500 为 0 表示成功

xyzGetPickout()

获取取料出框轨迹

参数 **GI003** (*ws_id*) –规划空间 id，默认填 0

返回 err_code: GD500 为 0 表示成功

xyzGetPlacein()

获取放料入框轨迹

参数 **GI003** (*ws_id*) –规划空间 id，默认填 0

返回 err_code: GD500 为 0 表示成功

xyzGetPlaceout()

获取放料出框轨迹

参数 **GI003** (*ws_id*) –规划空间 id，默认填 0

返回 err_code: GD500 为 0 表示成功

xyzSwitchStrat()

请求切换策略

参数 **GS004** (*strat_name*) –策略名称

返回 err_code: GD500 为 0 表示成功

xyzUpdTotePose()

料箱重定位

返回

err_code: GD500 为 0 表示成功

tote_pose: GP0003: GD501~GD506 料箱位姿

xyzUpdObjOnHand()

工件在上手的二次定位

返回 err_code: GD500 为 0 表示成功

xyzUpdObjToHand()

工件不在手上的二次定位

返回 err_code: GD500 为 0 表示成功

xyzGetObjType()

获取工件姿态类型

返回

err_code: GD500 为 0 表示成功

pose_type: GD501 工件姿态类型

xyzResetPalStat()

重置工业码垛状态

返回 err_code: GD500 为 0 表示成功

xyzSwitchItem()

切换工件

参数 **GS005** (*item_codename*) –工件名称

返回 err_code: GD500 为 0 表示成功

xyzCalGraspPose()

计算抓取位姿

参数 **GI003** (*ws_id*) –工作空间 id

返回 err_code: GD500 为 0 表示成功 grasp_pose GD501~GD506 抓取位姿

xyzCalObjPose()

计算物体位姿

参数 **GI003** (*ws_id*) –工作空间 id

返回 err_code: GD500 为 0 表示成功 obj_pose GD501~GD506 物体位姿

案例/模板说明

机械臂主控主函数说明

注意对工控机返回的 err_code 进行判断。

轨迹移动同步模板:

```
NOP
CLOSEMSG ID = 1 关闭socket连接
OPENMSG ID = 1 打开socket连接
SET B001 = 1
TIMER T = 1
MSG_CONN_ST 1 B001
IF {(B001 != 1)}
PRINTMSG 2 #socket connection failed!#
STOPRUN
ENDIF
SET GS002 = #traj_sync.t# 切换任务
CALL [$xyzSwitchFlow$]
IF {(GD500 != 0)}
STOPRUN
ENDIF
```

(下页继续)

(续上页)

```
SET GI003 = 0
SET GS005 = #item1# 切换工件
CALL [$xyzSwitchItem$]
IF {(GD500 != 0)}
STOPRUN
ENDIF
DOUT OT#(1) 1 T = 0 0  初始化 IO
MOVJ P0002 VJ = 10 % PL = 0 ACC = 10 DEC = 10 0 移动到初始位姿
LABEL [$loop$]
SET GI003 = 0
CALL [$xyzReqPickPlace$] 请求计算抓取放置轨迹
IF {(GD500 != 0)}
STOPRUN
ENDIF
SET GI003 = 0
CALL [$xyzGetPickIn$] 获取抓取入筐轨迹
IF {(GD500 != 0)}
STOPRUN
ENDIF
CALL [$xyzExePickinTraj$] 执行抓取入筐轨迹
SET GI003 = 0
CALL [$xyzGetPickOut$] 获取抓取出筐轨迹
IF {(GD500 != 0)}
STOPRUN
ENDIF
CALL [$xyzExePikoutTraj$] 执行抓取出筐轨迹
SET GI003 = 0
CALL [$xyzGetPlaceIn$] 获取放置入筐轨迹
IF {(GD500 != 0)}
STOPRUN
ENDIF
DOUT OT#(1) 0 T = 0 0
CALL [$xyzExeTraj$] 执行放置入筐轨迹
SET GI003 = 0
CALL [$xyzGetPlaceOut$] 获取放置出筐轨迹
IF {(GD500 != 0)}
STOPRUN
ENDIF
CALL [$xyzExeTraj$] 执行放置出筐轨迹
JUMP [$loop$]
END
```

轨迹移动异步模板

```
NOP
CLOSEMSG ID = 1 关闭socket连接
OPENMSG ID = 1  打开socket连接
SET B001 = 1
TIMER T = 1
MSG_CONN_ST 1 B001
IF {(B001 != 1)}
PRINTMSG 2 #socket connection failed!#
STOPRUN
ENDIF
```

(下页继续)

(续上页)

```
SET GS002 = #traj_async.t# 切换任务
CALL [$xyzSwitchFlow$]
IF {(GD500 != 0)}
STOPRUN
ENDIF
SET GI003 = 0
SET GS005 = #item1# 切换工件
CALL [$xyzSwitchItem$]
IF {(GD500 != 0)}
STOPRUN
ENDIF
DOUT OT#(1) 1 T = 0 0
MOVJ P0001 VJ = 10 % PL = 0 ACC = 10 DEC = 10 0
SET GI003 = 0
CALL [$xyzReqPickPlace$] 请求计算抓取放置轨迹
IF {(GD500 != 0)}
STOPRUN
ENDIF
LABEL [$loop$]
SET GI003 = 0
CALL [$xyzGetPickIn$] 获取抓取入筐轨迹
IF {(GD500 != 0)}
STOPRUN
ENDIF
IF {(GD503 < 1)}
STOPRUN
ENDIF
CALL [$xyzExePickinTraj$]  执行抓取入筐轨迹
SET GI003 = 0
CALL [$xyzGetPickOut$] 获取抓取出筐轨迹
IF {(GD500 != 0)}
STOPRUN
ENDIF
CALL [$xyzExePikoutTraj$] 执行抓取出筐轨迹
SET GI003 = 0
CALL [$xyzReqPickPlace$] 提前请求计算抓取放置轨迹
IF {(GD500 != 0)}
STOPRUN
SET GI003 = 0
CALL [$xyzGetPlaceIn$] 获取放置入筐轨迹
IF {(GD500 != 0)}
STOPRUN
ENDIF
DOUT OT#(1) 0 T = 0 0
CALL [$xyzExeTraj$] 执行放置入筐轨迹
SET GI003 = 0
CALL [$xyzGetPlaceOut$] 获取放置出筐轨迹
IF {(GD500 != 0)}
STOPRUN
ENDIF
CALL [$xyzExeTraj$] 执行放置出筐轨迹
JUMP [$loop$]
END
```

座标移动基础模板:

```
NOP
CLOSEMSG ID = 1 关闭socket连接
OPENMSG ID = 1 打开socket连接
SET B001 = 1
TIMER T = 1
MSG_CONN_ST 1 B001
IF {(B001 != 1)}
PRINTMSG 2 #socket connection failed!#
STOPRUN
ENDIF
SET GS002 = #cart_basic.t# 切换任务
CALL [$xyzSwitchFlow$]
IF {(GD500 != 0)}
STOPRUN
ENDIF
SET GI003 = 0
SET GS005 = #item1# 切换工件
CALL [$xyzSwitchItem$]
IF {(GD500 != 0)}
STOPRUN
ENDIF
DOUT OT#(1) 1 T = 0 0 初始化IO
POSSETALL P0001 RF  0 0 0 0 0 0 0 0 0 0 0 0 0 0
MOVJ P0001 VJ = 10 % PL = 0 ACC = 10 DEC = 10 0 移动到初始位姿
SET B001 = 1
LABEL [$loop$]
IF {(B001 == 1)}
POSSETALL P0002 RF  0 0 0 0 0 0 0 0 0 0 0 0 0 0
MOVL P0002 V = 10 mm/s PL = 0 ACC = 1 DEC = 1 0 移动到拍照位姿
CALL [$xyzSendCurCart$] 发送当前位姿
IF {(GD500 != 0)}
STOPRUN
ENDIF
ENDIF
SET GI003 = 0
CALL [$xyzReqGraspPose$] 请求抓取位姿
CALL [$xyzGetGraspPose$] 获取抓取位姿
POSSETALL GP0001 BF  GD501 GD502 GD503 GD504 GD505 GD506 0 0 0 0 0 0 0 0
IF {(GD508 < 1)}
PRINTMSG 1 #No grasp pose, continue requesting#
JUMP [$loop$]
ENDIF
MOVL GP0001 V = 10 mm/s PL = 0 ACC = 1 DEC = 1 0 移动到抓取位姿
DOUT OT#(1) 0 T = 0 0 抓取
POSSETALL P0003 RF  0 0 0 0 0 0 0 0 0 0 0 0 0 0
MOVL P0003 V = 10 mm/s PL = 0 ACC = 1 DEC = 1 0 移动到放置位姿
DOUT OT#(1) 1 T = 0 0 放置
JUMP [$loop$]
END
```

座标移动二次定位模板

```
NOP
CLOSEMSG ID = 1 关闭socket连接
OPENMSG ID = 1 打开socket连接
SET B001 = 1
TIMER T = 1
MSG_CONN_ST 1 B001
IF {(B001 != 1)}
PRINTMSG 2 #socket connection failed!#
STOPRUN
ENDIF
SET GS002 = #cart_repo.t# 切换任务
CALL [$xyzSwitchFlow$]
IF {(GD500 != 0)}
STOPRUN
ENDIF
DOUT OT#(1) 0 T = 0 0 初始化IO
DOUT OT#(2) 0 T = 0 0
POSSETALL P0001 RF  0 0 0 0 0 0 0 0 0 0 0 0 0 0
MOVJ P0001 VJ = 10 % PL = 0 ACC = 10 DEC = 10 0 移动到初始位姿
SET GI003 = 0
SET GS005 = #item1# 切换工件
CALL [$xyzSwitchItem$]
IF {(GD500 != 0)}
STOPRUN
ENDIF
SET GI003 = 0
CALL [$xyzReqGraspPose$] 请求粗抓取位姿
IF {(GD500 != 0)}
STOPRUN
ENDIF
LABEL [$loop$]
CALL [$xyzGetGraspPose$] 获取粗抓取位姿
POSSETALL GP0001 BF  GD501 GD502 GD503 GD504 GD505 GD506 0 0 0 0 0 0 0 0
IF {(GD500 != 0)}
STOPRUN
ENDIF
IF {(GD508 < 1)} 如果没有抓取到工件，尝试抓隔板
SET GI003 = 0
SET GS005 = #board# 切换抓取物体为隔板
CALL [$xyzSwitchItem$]
IF {(GD500 != 0)}
STOPRUN
ENDIF
SET GI003 = 0
CALL [$xyzReqGraspPose$] 请求隔板抓取位姿
CALL [$xyzGetGraspPose$] 获取隔板抓取位姿
IF {(GD500 != 0)}
STOPRUN
ENDIF
POSSETALL GP0001 BF  GD501 GD502 GD503 GD504 GD505 GD506 0 0 0 0 0 0 0 0
IF {(GD508 < 1)}
PRINTMSG 1 #No board, you can change grasp tote# 没有隔板，可以换筐
STOPRUN 停止运行
ENDIF
POSSETALL GP0001 BF  GD501 GD502 GD503 GD504 GD505 GD506 0 0 0 0 0 0 0 0
```

(下页继续)

(续上页)

```
MOVL GP0001 V = 10 mm/s PL = 0 ACC = 1 DEC = 1 0 移动到隔板抓取位姿
DOUT OT#(1) 1 T = 0 0 抓取
POSSETALL GP0001 BF  111 222 333 1.5708 0 0 0 0 0 0 0 0 0 0
MOVL GP0001 V = 10 mm/s PL = 0 ACC = 1 DEC = 1 0 移动到隔板放置位姿
DOUT OT#(1) 0 T = 0 0 放置
SET GI003 = 0
SET GS005 = #item1# 切换抓取物体为工件
CALL [$xyzSwitchItem$]
IF {(GD500 != 0)}
STOPRUN
ENDIF
SET GI003 = 0
CALL [$xyzReqGraspPose$] 请求粗抓取位姿
IF {(GD500 != 0)}
STOPRUN
ENDIF
JUMP [$loop$]
ENDIF
MOVL GP0001 V = 10 mm/s PL = 0 ACC = 1 DEC = 1 0 移动到粗抓取位姿
CALL [$xyzSendCurCart$] 发送当前位姿
IF {(GD500 != 0)}
STOPRUN
ENDIF
SET GI003 = 1
SET GS005 = #item1# 切换抓取物体为工件
CALL [$xyzSwitchItem$]
IF {(GD500 != 0)}
STOPRUN
ENDIF
SET GI003 = 1
CALL [$xyzReqGraspPose$] 请求精确抓取位姿
CALL [$xyzGetGraspPose$] 获取精确抓取位姿
IF {(GD500 != 0)}
STOPRUN
ENDIF
POSSETALL GP0001 BF  GD501 GD502 GD503 GD504 GD505 GD506 0 0 0 0 0 0 0 0
IF {(GD508 < 1)}
PRINTMSG 1 #No fine grasp pose, stop# 没有精确抓取位姿，停止
STOPRUN
ENDIF
MOVL GP0001 V = 10 mm/s PL = 0 ACC = 1 DEC = 1 0 移动到精确抓取位姿
DOUT OT#(2) 1 T = 0 0 抓取
POSSETALL P0002 RF  0 0 0 0 0 0 0 0 0 0 0 0 0 0
MOVL P0002 V = 10 mm/s PL = 0 ACC = 1 DEC = 1 0 移动到放置位姿
DOUT OT#(2) 0 T = 0 0 放置
SET GI003 = 0
SET GS005 = #item1# 切换抓取物体为工件
CALL [$xyzSwitchItem$]
IF {(GD500 != 0)}
STOPRUN
ENDIF
SET GI003 = 0
CALL [$xyzReqGraspPose$] 请求粗抓取位姿
IF {(GD500 != 0)}
STOPRUN
ENDIF
```

(下页继续)

(续上页)

```
JUMP [$loop$]
END
```

常见问题

1. 后台程序无法停止

   解决方法：重启控制器

附录

14.2.9 inovance

此处介绍安装 inovance 汇川机械臂驱动的相关事项。

驱动版本和使用要求

支持的机械臂类型

inovance 六轴机械臂

控制器型号

inovance 控制器

控制器需要开通的功能

1. 自带 Socket 功能，无需额外开通
2. 控制器版本需要为 S03.21R15 或更高版本。

- 如果版本较低，函数不支持传出变量，代码无法正确运行，需要联系汇川售后升级版本。**注意控制器版本和示教器版本、InoRobotLab 版本需要保持一致**。
- 同一时间，只能示教器、虚拟示教器、InoRobotLab 其中一种连接机器人。若另一软件想连接，必须先断开当前连接。

3. 下面是使用示教器和 InoRobotLab 查看版本的方法：

- 示教器查看版本
  - 首先断开其他软件与机器人控制器的连接。
  - 进入 系统设置 -> 通讯设置 -> 连接控制器，选择正确的 IP，点击 连接。
  - 在连接控制器成功后，进入 监控 -> 版本信息查看版本：
- InoRobotLab 查看版本

  InoRobotLab 是汇川官方的编写代码、运行和调试程序的软件，可在汇川官网下载 **与示教器版本一致的 InoRobotLab，推荐使用**。

  在连接机械臂后，可以在 InoRobotLab 中 关于 -> 版本信息查看当前版本：

对象	版本
示教器版本	S03.21R16
控制器版本	--
InoRobShop版本	--

图 57: 示教器查看当前版本

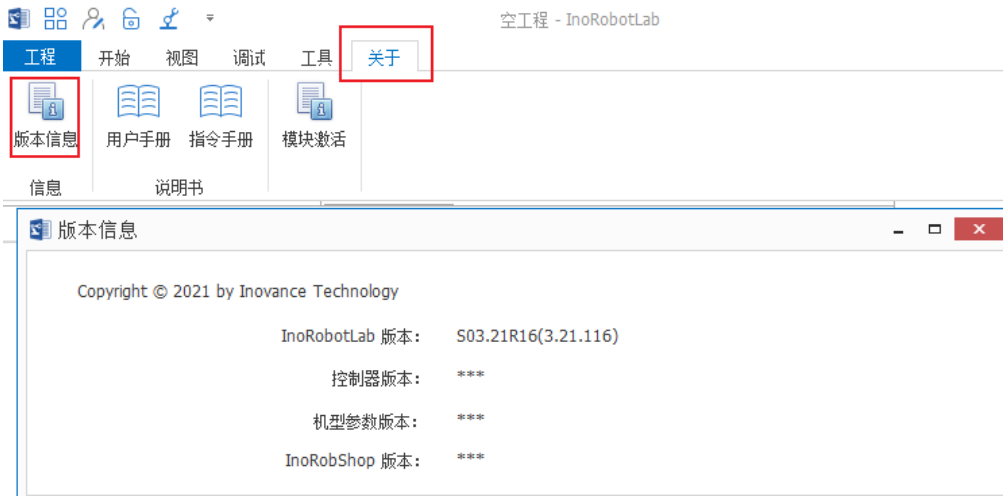


图 58: InoRobotLab 查看当前版本

安装驱动

inovance 机械臂驱动文件列表

机械臂代码可从 MAX 安装目录下的 share/robot_code/inovance 中获得。

```
└── xyz
    ├── Data 数据文件夹（部分文件可能不存在）
    │   ├── BreakPoints.jsn
    │   ├── GripLoadParam.jsn
    │   ├── Labels.jsn
    │   ├── MonitorVars.jsn
    │   ├── P.pts
    │   ├── ToolCoordSysParam.jsn
    │   ├── UserCoordSysParam.jsn
    │   └── UserDefineWarning.jsn
    ├── api.pro 机械臂主控用到的 api
    ├── background.pro 后台程序，用于机械臂主控发送心跳信号，及工控机主控不断发送当前状态
    ├── lib.pro 数据处理相关函数
    ├── main.pro 主程序
    ├── project 项目文件夹（部分文件可能不存在）
    │   ├── Device
    │   │   └── Device.dat
    │   ├── RealTimeMonitor
    │   │   └── monitor.dat
    │   ├── Variables
    │   │   ├── IRC500-L6D_0A36742D-B012-467C-9D55-B8C7D8A352B7
    │   │   │   └── Variables.dat
    │   │   ├── IR_SV660N_4EBA2E57-4319-4767-95A2-4389CA3320DB
    │   │   │   └── Variables.dat
    │   │   ├── IR_SV660N_5FAA612D-EB88-472D-B41B-1FD796152192
    │   │   │   └── Variables.dat
    │   │   └── IR_SV660N_C6F29084-5D29-4FA5-B678-1FD40E7B395A
    │   │       └── Variables.dat
    │   └── project.inopro
    ├── xyz.prj 工程文件
    ├── xyzCartMoveBasic.pro 座标移动基础模板
    ├── xyzCartMoveReposition.pro 座标移动二次定位模板
    ├── xyzConst.pro 常数定义
    ├── xyzMaster.pro 机械臂主控测试程序
    ├── xyzMotion.pro 工控机主控程序
    ├── xyzTrajMoveAsync.pro 轨迹移动异步模板
    └── xyzTrajMoveSync.pro 轨迹移动同步模板
```

设定机械臂 IP

1. 进入 设置 -> 系统设置 -> 通讯设置 -> 连接到控制器，IP 地址为 192.168.23.25，连接控制器。
2. 连接成功后，在真实示教器最上方，切换到管理模式，密码为 “000000”。
3. 进入 设置 -> 系统设置 -> 通讯设置 -> 网络配置，将 ethernet 网口 1 ip 改为 192.168.37.100

图 59: inovance 设定机械臂 IP

4. 将 ethernet 网口 1 连接到工控机上或者交换机上，工控机 ip 设为 192.168.37.101。

使用 InoRobotLab 导入程序

推荐使用 InoRobotLab 导入、运行程序。示教器导入可能会有问题。

1. 在汇川官网下载好对应控制器版本的 InoRobotLab, 在 控制器处右键单击，选中 属性，配置 IP 地址为 192.168.37.100，保存连接。

2. 进入工程，点击打开本地工程，选择 MAX 安装目录下的 share/robot_code/inovance 中的 xyz.prj

3. 进入 调试，点击 同步到控制器。

注 1：在使用 InoRobotLab 导入下位机程序时如果出现乱码报错，则关闭系统的 UTF-8 再重新启动打开软件查看错误 (注意使用 Max 软件时需要打开 UTF-8)

注 2：若出现 程序文件机型匹配错误或 全局点文件机型匹配错误，可能是汇川机型匹配问题，需要修改机型匹配文件。下载下面文档，按文档操作：

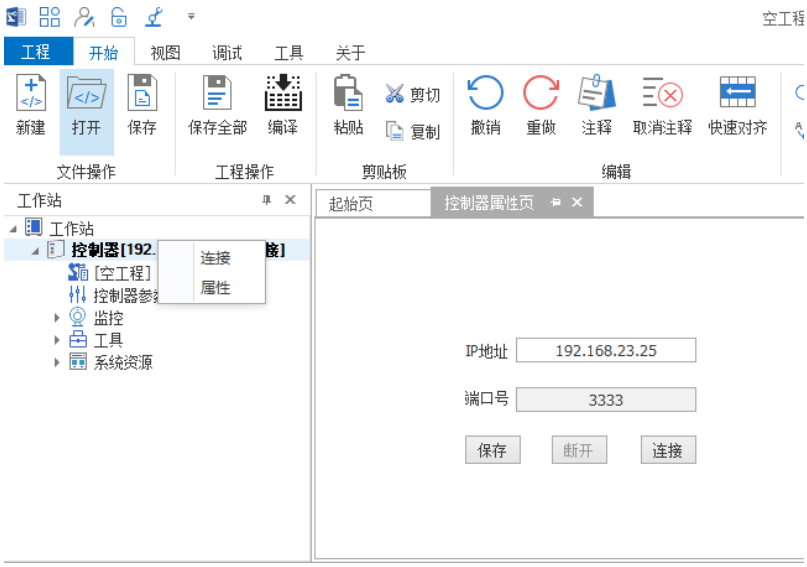


图 60: InoRobotLab 连接控制器

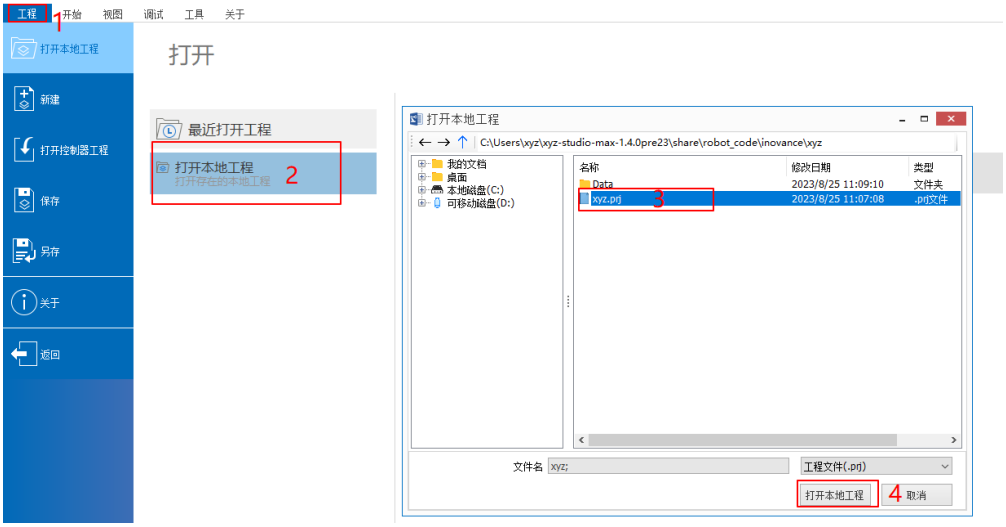


图 61: InoRobotLab 导入程序 1

图 62: InoRobotLab 导入程序 2

使用虚拟示教器导入程序

1. 在汇川官网下载好对应控制器版本的虚拟示教器，打开后进入 设置 -> 系统设置 -> 通讯设置 -> 连接到控制器，连接控制器。注意此时 IP 地址为 192.168.37.100.
2. **按照图片选择，进入导入程序界面，选择需要导入的工程文件，即 MAX 安装目录下的 share/robot_code/inovance 中的 x**

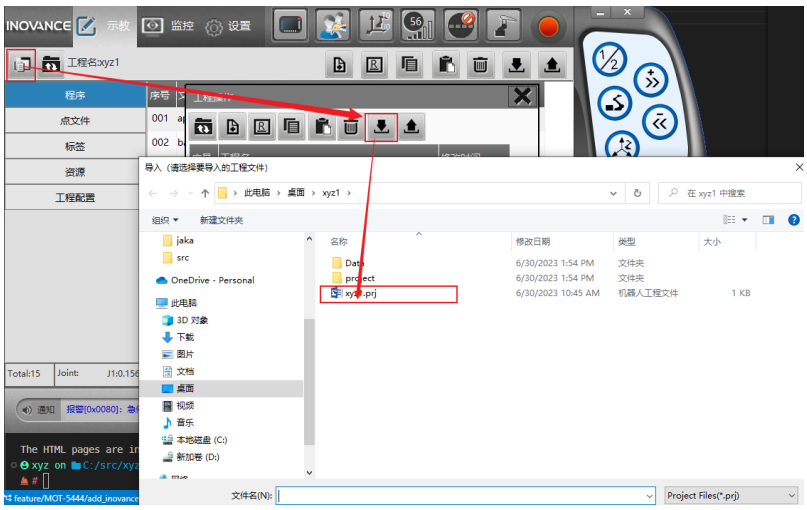


图 63: 虚拟示教器导入程序

使用 U 盘导入程序

1. 将 MAX 安装目录下的 share/robot_code 中 inovance 的机械臂代码拷入 U 盘，插入示教器 USB 口
2. 在示教器上，按照图片选择，进入导入程序界面，选择需要导入的工程文件 xyz.prj

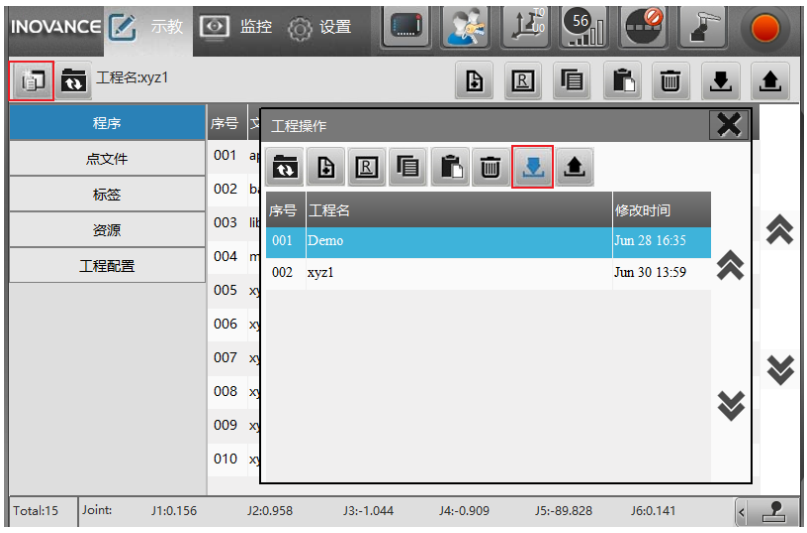


图 64: U 盘导入程序

运行程序

通讯说明

工控机和 inovance 控制器的通讯方式为 socket：工控机作为 socket server(服务端)，机械臂作为 socket client（客户端）。

- 机械臂主控，使用的是第一个 socket 连接，代码中对应的是 2, 即代码中的 Open Socket(“192.168.37.101” ,11111,2,Single, LB[1]) 中的 2.
- 工控机主控，使用的是第二个和第三个 socket 连接，代码中对应的是端口 3 和端口 4。

注：按照汇川控制器的设定，端口 2-5 为自由分配端口。

运行前 socket 设定（在真实示教器上操作）

需要在 真实示教器 -> 设置 -> 系统设置 -> 通讯设置 -> 通讯服务管理将 #2，#3，#4 都改为客户端，点击右上角 保存。

InoRobotLab 运行程序（目前推荐使用）

进入调试界面，点击 返回起始行，打开 main 函数，反注释需要运行的程序，注释不使用的程序。

右侧注意清除错误，上使能。

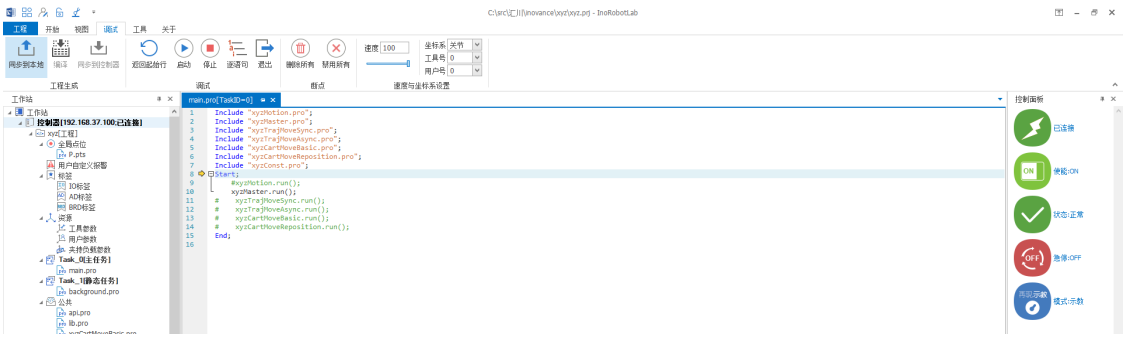


图 65: InoRobotLab 运行程序

示教器运行程序

1. 机械臂主控运行

   打开 main 函数，在修改机械臂主控模板程序后，反注释需要运行的机械臂主控程序，注释不使用的程序。

   按下示教器模式切换按钮。切换为再现模式，右上角如有！注意清楚错误，随后按下启动按钮即可运行。

2. 工控机主控运行

   修改 main.pro, 反注释 xyzMotion.run()，注释其他行。

   按下示教器模式切换按钮。切换为再现模式，右上角如有！注意清楚错误，随后按下启动按钮即可运行。

API 说明

inovance 工控机主控支持的 API

指令号	命令	支持情况
100	获取机械臂程序版本号	支持
101	发送速度数据	支持
102	发送加速度数据	支持
103	发送圆滑过渡参数	支持
104	发送工具坐标系 (TCP) 参数	支持
105	设置数字量输出	支持
106	SetJointsMovej	支持
107	SetCartMovel	支持
108	SetJoinsMovel	支持
109	SetCartMovej	支持
110	MovejSequence	支持
111	MovelSequence	支持
112	SetJointsMovejDo	不支持
113	SetCartMovelDo	不支持
114	SetJointsMovelDo	不支持
115	SetJointsMovejGroupDo	不支持
116	SetCartMovelGroupDo	不支持
117	SetJointsMovelGroupDo	不支持
118	MovelUntil	支持
119	获取数字量输入状态	支持
120	获取数字量输出状态	支持
121	获取模拟量输入口数值	不支持
122	获取机械臂当前角度	支持
123	获取机械臂当前位姿	支持
200	机械臂后台发送状态	支持

inovance 机械臂主控支持的 API

xyzSwitchAPP(*app_name*, *err_code*)

切换应用

参数 **app_name** –应用名称

返回 err_code 为 0 表示成功

xyzSwitchFlow(*flow_name*, *err_code*)

切换工件

参数 **flow_name** –流图名称

返回 err_code 为 0 表示成功

xyzSwitchTool(*tool_name*, *err_code*)

切换工具

参数 **tool_name** –工具名称

返回 err_code 为 0 表示成功

xyzReqCapImg(*vision_service_id*, *err_code*, *cap_img_token*)

请求拍照

参数 vision_service_id –视觉服务 id

返回

err_code 为 0 表示成功

cap_img_token 返回的 token

xyzGetCapImg(*cap_img_token*, *err_code*)

获取拍照结果

参数 cap_img_token –请求拍照时返回的 token

返回 err_code 为 0 表示成功

xyzCapImg(*vision_service_id*, *err_code*)

拍照

参数 vision_service_id –需要进行拍照操作的工作空间 id

返回 err_code 为 0 表示成功

xyzReqGraspPose(*ws_id*, *err_code*, *grasp_pose_token*)

请求抓取位姿

参数 ws_id –需要获取抓取点位的工作空间 id

返回

err_code 为 0 表示成功

grasp_pose_token: 返回的用于获取目标点位时使用的 token

xyzGetGraspPose(*grasp_pose_token*, *err_code*, *grasp_pose*, *grasp_pose_num*, *pipeline_num*, *register_num*)

获取抓取位姿

参数 grasp_pose_token –求抓取目标点位时返回的 token

返回

err_code 为 0 表示成功

grasp_pose: 抓取位姿

grasp_pose_num: 可供抓取的点数量

pipeline_num: pipeline 文件 number

register_num: 用到的注册文件的注册 number

xyzReqObjPose(*ws_id*, *err_code*, *obj_pose_token*)

请求物体位姿

参数 ws_id –需要获取物体位姿的工作空间 id

返回

err_code 为 0 表示成功

obj_pose_token: 返回的用于物体位姿识别的 token

xyzGetObjPose(*obj_pose_token*, *err_code*, *obj_pose*, *obj_pose_num*, *obj_pose_type*)

获取物体位姿

参数 obj_pose_token –请求物体位姿时得到的 token

返回

err_code 为 0 表示成功

obj_pose: 物体位姿

obj_pose_num: 物体数量

obj_pose_type: 当前返回的物体 pose 类型

xyzResetTask(*err_code*)

重置视觉

参数 ws_id –需要重置的任务 id

返回 err_code 为 0 表示成功

xyzSendCurrentJoints(*err_code*)

发送机械臂当前角度：j1~j6: 机械臂当前的角度信息，如果机械臂轴数小于 6 轴，则需要对应位置补零，依旧发送六个数。

返回 err_code 为 0 表示成功

xyzSendCurrentCartPose(*err_code*)

发送机械臂法兰当前位姿

返回 err_code 为 0 表示成功

xyzSendCurrentExtJoints()

暂不支持。

发送机械臂当前扩展轴位置：j1~j6: 机械臂当前扩展轴的角度信息，如果扩展轴数不足 6 的, 需要补零后发送六个数。

返回 err_code 为 0 表示成功

xyzReqPick()

暂不支持。请求 pick 动作规划

返回 err_code 为 0 表示成功

xyzReqPlace()

暂不支持。请求 place 动作规划

返回 err_code 为 0 表示成功

xyzReqPickPlace(*ws_id*, *err_code*)

请求 pick 和 place 规划

参数 ws_id –需要抓取的工作空间 id

返回 err_code 为 0 表示成功

xyzGetPickin(*ws_id*, *err_code*, *pipeline_num*, *register_num*, *num*, *wp_type*, *pos*)

获取取料入框轨迹

参数 ws_id –规划空间 id，默认填 0

返回

err_code 为 0 表示成功

pipeline_num pipeline 文件 number

register_num 用到的注册文件的注册 number

num 轨迹点数

wp_type 姿态数组

pos 轨迹点数组

xyzGetPickout(*ws_id*, *err_code*, *pipeline_num*, *register_num*, *num*, *wp_type*, *pos*)

获取取料出框轨迹

参数 ws_id–规划空间 id，默认填 0

返回

err_code 为 0 表示成功

pipeline_num pipeline 文件 number

register_num 用到的注册文件的注册 number

num 轨迹点数

wp_type 姿态数组

pos 轨迹点数组

xyzGetPlacein(*ws_id*, *err_code*, *pipeline_num*, *register_num*, *num*, *wp_type*, *pos*)

获取放料入框轨迹

参数 ws_id–规划空间 id，默认填 0

返回

err_code 为 0 表示成功

pipeline_num pipeline 文件 number

register_num 用到的注册文件的注册 number

num 轨迹点数

wp_type 姿态数组

pos 轨迹点数组

xyzGetPlaceout(*ws_id*, *err_code*, *pipeline_num*, *register_num*, *num*, *wp_type*, *pos*)

获取放料出框轨迹

参数 ws_id–规划空间 id，默认填 0

返回

err_code 为 0 表示成功

pipeline_num pipeline 文件 number

register_num 用到的注册文件的注册 number

num 轨迹点数

wp_type 姿态数组

pos 轨迹点数组

xyzSwitchStrat(*strategy_name*, *err_code*)

请求切换策略

参数 **strategy_name** –策略名称

返回 err_code 为 0 表示成功

xyzUpdateTotePose(*err_code*, *tote_pose*)

料箱重定位

返回

err_code 为 0 表示成功

tote_pose: 料箱位姿

xyzUpdateObjPoseOnHand(*err_code*)

工件在上手的二次定位

返回 err_code 为 0 表示成功

xyzUpdateObjPoseToHand(*err_code*, *pipeline_num*, *register_num*, *num*, *wp_type*, *pos*)

工件不在手上的二次定位

返回

err_code 为 0 表示成功

pipeline_num pipeline 文件 number

register_num 用到的注册文件的注册 number

num 轨迹点数

wp_type 姿态数组

pos 轨迹点数组

xyzGetObjPoseType(*err_code*, *obj_pose_type*)

获取工件姿态类型

返回

err_code 为 0 表示成功

obj_pose_type: 工件姿态类型

xyzResetPalletStatus(*err_code*)

重置工业码垛状态

返回 err_code 为 0 表示成功

xyzSwitchItem(*ws_id*, *item_codename*, *err_code*)

切换工件

参数

- **ws_id** –工作空间 id
- **item_codename** –工件名称

返回 err_code 为 0 表示成功

xyzCalculateGraspPose(*ws_id*, *err_code*, *grasp_pose*, *grap_pose_num*, *pipeline_num*, *register_num*)

计算抓取位姿

参数 ws_id –工作空间 id

返回

err_code 为 0 表示成功

grasp_pose 抓取位姿

grap_pose_num 可供抓取的点数量

pipeline_num pipeline 文件 number

register_num 用到的注册文件的注册 number

xyzCalculateObjPose(*ws_id*, *err_code*, *obj_pose*, *obj_pose_num*, *obj_pose_type*)

计算物体位姿

参数 ws_id –工作空间 id

返回

err_code 为 0 表示成功

obj_pose 物体位姿

obj_pose_num 物体数量

obj_pose_type 当前返回的物体 pose 类型

案例/模板说明

机械臂主控主函数说明

注意对工控机返回的 err_code 进行判断。

座标移动基础模板:

```
Include "api.pro";
Func run()
    # S1, S2 : Init and conect To IPC 初始化以及连接工控机
    LB[1] = 0;
    While LB[1] <>1
        Open Socket("192.168.37.101",11111,2,Single, LB[1]);
    EndWhile;
    gMasterPort = 2;
    Get Port[gMasterPort],T[0.2],Goto L[2];
    L[2]:
    If GetPortState(gMasterPort) <> 1
        Halt 3;
        Goto L[1];
    EndIf;
    Int err_code;
    gMasterFlag = True;
    #-----------------------------------------#
    #    String flow_name ="cart_basic.t";
    #    api.xyzSwitchFlow(flow_name, err_code);
```

(下页继续)

(续上页)

```
    #    If err_code <> 0
    #        Halt 3;
    #        Goto L[1];
    #    EndIf;
    # S3 切换工件
    Int ws_id = 0;
    String item_codename = "item1";
    api.xyzSwitchItem(ws_id, item_codename, err_code);
    If err_code <> 0
        Halt 3;
        Goto L[1];
    EndIf;
    # io operation
    Set Out[1],ON;
    # S4: move To Home 回原点
    LP[1] = (0, 0, 0, 0, -90, 0),(0,0,0,1),(1,0,0);
    Movj  LP[1], V[30], Z[0], Tool[0];
    Bool is_eye_in_hand = True;
    Int grasp_pose_token = 0;
    Double grasp_pose[6] = {0,0,0,0,0,0};
    Int grasp_pose_num = 0;
    Int pipeline_num = 0;
    Int register_num = 0;
    While True
        # S5
        If is_eye_in_hand
            # S6: Move To scan pose 移动到拍照位姿
            LP[1] = (0, 0, 0, 0, -90, 0),(0,0,0,1),(1,0,0);
            Movj  LP[1], V[30], Z[0], Tool[0];
            # S7  发送当前位姿
            api.xyzSendCurrentCartPose(err_code);
            If err_code <> 0
                Halt 3;
                Goto L[1];
            EndIf;
        EndIf;
        # S8: req grasp pose(including cap image) 请求抓取位姿
        grasp_pose_token = 0;
        ws_id = 0;
        api.xyzReqGraspPose(ws_id, err_code, grasp_pose_token);
        If err_code <> 0
            Halt 3;
            Goto L[1];
        EndIf;
        # S9 获取抓取位姿
        grasp_pose_num = 0;
        pipeline_num = 0;
        register_num = 0;
        api.xyzGetGraspPose(grasp_pose_token, err_code, grasp_pose, grasp_pose_num,␣
↪pipeline_num, register_num);
        If err_code <> 0
            Halt 3;
            Goto L[1];
        EndIf;
        LP[2] = (grasp_pose[0],grasp_pose[1],grasp_pose[2],grasp_pose[3],grasp_
↪pose[4],grasp_pose[5]),(0,0,0,1),(3,0,0);
```

(下页继续)

(续上页)

```
        If grasp_pose_num < 1
            Wait T[5];
            Continue;
        EndIf;
        # S10: Move To grasp pose 移动到抓取位姿
        # add more path pose, modify the offset PR[], unit: mm
        # 添加额外路径点
        PR[1] = (0,0,100,0,0,0);
        Movj Offset(LP[2],PR[1]),V[30],Z[0],Tool[0];
        Movl LP[2], V[30], Z[0], Tool[0];
        Movj Offset(LP[2],PR[1]),V[30],Z[0],Tool[0];
        # io operation
        Set Out[1],OFF;
        # S11: Move To place pose 移动到放置位姿
        # add more path pose, modify the offset PR[], unit: mm
        # 添加额外路径点
        PR[2] = (0,0,100,0,0,0);
        LP[3] = (0, 0, 0, 0, -90, 0),(0,0,0,1),(3,0,0);
        Movj Offset(LP[2],PR[2]),V[30],Z[0],Tool[0];
        Movl  LP[3], V[30], Z[0], Tool[0];
        Movj Offset(LP[2],PR[2]),V[30],Z[0],Tool[0];
        # io operation
        Set Out[1],ON;
    EndWhile;
    L[1]:
    Close Socket, gMasterPort;
EndFunc;
```

座标移动二次定位模板

```
Include "api.pro";
Func run()
    # S1, S2 : Init and conect To IPC 初始化以及连接工控机
    LB[1] = 0;
    While LB[1] <>1
        Open Socket("192.168.37.101",11111,2,Single, LB[1]);
    EndWhile;
    gMasterPort = 2;
    Get Port[gMasterPort],T[0.2],Goto L[2];
    L[2]:
    If GetPortState(gMasterPort) <> 1
        Halt 3;
        Goto L[1];
    EndIf;
    Int err_code;
    gMasterFlag = True;
    #-----------------------------------------#
    #    String flow_name ="cart_repo.t";
    #    api.xyzSwitchFlow(flow_name, err_code);
    #    If err_code <> 0
    #        Halt 3;
    #        Goto L[1];
    #    EndIf;
    # S3 切换工件
```

(下页继续)

(续上页)

```
    Int ws_id = 0;
    String item_codename = "item1";
    api.xyzSwitchItem(ws_id, item_codename, err_code);
    If err_code <> 0
        Halt 3;
        Goto L[1];
    EndIf;
    # S4: Move To Home 回原点
    LP[1] = (0, 0, 0, 0, -90, 0),(0,0,0,1),(1,0,0);
    Movj  LP[1], V[30], Z[0], Tool[0];
    # io operation
    Set Out[1],ON;
    Int rough_grasp_pose_token = 0;
    Double rough_grasp_pose[6] = {0,0,0,0,0,0};
    Int rough_grasp_pose_num = 0;
    Int rough_pipeline_num = 0;
    Int rough_register_num = 0;
    String board_name;
    Int board_grasp_pose_token = 0;
    Double board_grasp_pose[6] = {0,0,0,0,0,0};
    Int board_grasp_pose_num = 0;
    Int board_pipeline_num = 0;
    Int board_register_num = 0;
    Int fine_grasp_pose_token = 0;
    Double fine_grasp_pose[6] = {0,0,0,0,0,0};
    Int fine_grasp_pose_num = 0;
    Int fine_pipeline_num = 0;
    Int fine_register_num = 0;
    While True
        # S5 请求抓取位姿
        rough_grasp_pose_token = 0;
        ws_id = 0;
        api.xyzReqGraspPose(ws_id, err_code, rough_grasp_pose_token);
        If err_code <> 0
            Halt 3;
            Goto L[1];
        EndIf;
        # S6: camera1 get rough grasp pose 获取粗抓取位姿
        api.xyzGetGraspPose(rough_grasp_pose_token, err_code, rough_grasp_pose, rough_
↪grasp_pose_num, rough_pipeline_num, rough_register_num);
        If err_code <> 0
            Halt 3;
            Goto L[1];
        EndIf;
        # S7: grasp board if no object exists on board 如果没有物体在隔板上，抓取隔板
        If (rough_grasp_pose_num < 1)
            # S15: switch board; 切换抓取物体为隔板
            ws_id = 0;
            board_name = "board";
            api.xyzSwitchItem(ws_id, board_name, err_code);
            If err_code <> 0
                Halt 3;
                Goto L[1];
            EndIf;
            # S16 请求隔板抓取位姿
            ws_id = 0;
```

(下页继续)

(续上页)

```
            api.xyzReqGraspPose(ws_id, err_code, board_grasp_pose_token);
            If err_code <> 0
                Halt 3;
                Goto L[1];
            EndIf;
            # S17 获取隔板抓取位姿
            api.xyzGetGraspPose(board_grasp_pose_token, err_code, board_grasp_pose,␣
↪board_grasp_pose_num, board_pipeline_num, board_register_num);
            If err_code <> 0
                Halt 3;
                Goto L[1];
            EndIf;
            LP[2] = (board_grasp_pose[0],board_grasp_pose[1],board_grasp_pose[2],
↪board_grasp_pose[3],board_grasp_pose[4],board_grasp_pose[5]),(0,0,0,1),(3,0,0);
            If (board_grasp_pose_num < 1)
                Halt 3;
                Goto L[1];
            EndIf;
            # S18: Move To board grasp pose 移动到隔板抓取位姿
            PR[1] = (0,0,100,0,0,0);
            # add more path pose, modify the offset PR[], unit: mm
            # 添加额外路径点
            Movj Offset(LP[2],PR[1]),V[30],Z[0],Tool[0];
            Movl LP[2], V[30], Z[0], Tool[0];
            Movj Offset(LP[2],PR[1]),V[30],Z[0],Tool[0];
            # io operation
            Set Out[1],OFF;
            # S19: Move To board place pose 移动到隔板放置位姿
            # add more path pose, modify the offset PR[], unit: mm
            # 添加额外路径点
            PR[2] = (0,0,100,0,0,0);
            LP[3] = (0, 0, 0, 0, -90, 0),(0,0,0,1),(3,0,0);
            Movj Offset(LP[3],PR[2]),V[30],Z[0],Tool[0];
            Movl LP[3], V[30], Z[0], Tool[0];
            Movj Offset(LP[3],PR[2]),V[30],Z[0],Tool[0];
            # io operation
            Set Out[1],ON;
            # S20 切换工件
            item_codename = "item1";
            ws_id = 0;
            api.xyzSwitchItem(ws_id, item_codename, err_code);
            If err_code <> 0
                Halt 3;
                Goto L[1];
            EndIf;
            Continue;
        EndIf;
        # S9: move To rough_grasp_pose 移动到粗抓取位姿
        LP[4] = (rough_grasp_pose[0],rough_grasp_pose[1],rough_grasp_pose[2],rough_
↪grasp_pose[3],rough_grasp_pose[4],rough_grasp_pose[5]),(0,0,0,1),(2,0,0);
        Movj  LP[4], V[30], Z[0], Tool[0];
        # S9: Send cart pose with eye In hand
        api.xyzSendCurrentCartPose(err_code);
        If err_code <> 0
            Halt 3;
            Goto L[1];
```

(下页继续)

(续上页)

```
        EndIf;
        # S10 使用机械臂上相机识别工件
        ws_id = 1;
        item_codename = "item1";
        api.xyzSwitchItem(ws_id, item_codename, err_code);
        If err_code <> 0
            Halt 3;
            Goto L[1];
        EndIf;
        # S11 请求精确抓取位姿
        fine_grasp_pose_token = 0;
        ws_id = 0;
        api.xyzReqGraspPose(ws_id, err_code, fine_grasp_pose_token);
        If err_code <> 0
            Halt 3;
            Goto L[1];
        EndIf;
        # S12 获取精确抓取位姿
        api.xyzGetGraspPose(fine_grasp_pose_token, err_code, fine_grasp_pose, fine_
↪grasp_pose_num, fine_pipeline_num, fine_register_num);
        If err_code <> 0
            Halt 3;
            Goto L[1];
        EndIf;
        If (fine_grasp_pose_num < 1)
            Halt 3;
            Goto L[1];
        EndIf;
        LP[5] = (fine_grasp_pose[0],fine_grasp_pose[1],fine_grasp_pose[2],fine_grasp_
↪pose[3],fine_grasp_pose[4],fine_grasp_pose[5]),(0,0,0,1),(3,0,0);
        # S13: Move To Fine grasp pose 移动到精确抓取位姿
        # add more path pose, modify the offset PR[], unit: mm
        # 添加额外路径点
        PR[3] = (0,0,100,0,0,0);
        Movj Offset(LP[5],PR[3]),V[30],Z[0],Tool[0];
        Movj  LP[5], V[30], Z[0], Tool[0];
        Movj Offset(LP[5],PR[3]),V[30],Z[0],Tool[0];
        # io operation
        Set   Out[1],OFF;
        # S14: Move To place pose 移动到放置位姿
        # add more path pose, modify the offset PR[], unit: mm
        # 添加额外路径点
        PR[4] = (0,0,100,0,0,0);
        LP[6] = (0, 0, 0, 0, -90, 0),(0,0,0,1),(1,0,0);
        Movj Offset(LP[6],PR[4]),V[30],Z[0],Tool[0];
        Movj  LP[6], V[30], Z[0], Tool[0];
        Movj Offset(LP[6],PR[4]),V[30],Z[0],Tool[0];
        # io operation
        Set Out[1],ON;
    EndWhile;
    L[1]:
    Close Socket, gMasterPort;
EndFunc;
```

轨迹移动同步模板:

```
Include "api.pro";
Func run()
    # S1,S2: Init And conect To IPC 初始化以及连接工控机
    LB[1] = 0;
    While LB[1] <>1
        Open Socket("192.168.37.101",11111,2,Single, LB[1]);
    EndWhile;
    gMasterPort = 2;
    Get Port[gMasterPort],T[0.2],Goto L[2];
    L[2]:
    If GetPortState(gMasterPort) <> 1
        Halt 3;
        Goto L[1];
    EndIf;
    Int err_code;
    gMasterFlag = True;
#-------------------------------------------#
#    String flow_name ="traj_sync.t"; # 切换任务
#    api.xyzSwitchFlow(flow_name, err_code);
#    If err_code <> 0
#        Halt 3;
#        Goto L[1];
#    EndIf;
#   S3
    Int ws_id = 0;
    String item_codename = "item1"; 切换工件
    api.xyzSwitchItem(ws_id, item_codename, err_code);
    If err_code <> 0
        Halt 3;
        Goto L[1];
    EndIf;
    # io operation
    Set   Out[1],ON;
    # S4: move To Home 回原点
    LP[1] = (0, 0, 0, 0, -90, 0),(0,0,0,1),(1,0,0);
    Movj  LP[1], V[30], Z[0], Tool[0];
    Int pipeline_num = 0;
    Int register_num = 0;
    Int num = 0;
    Int wp_type[30]={};
    Double pos[30][6]={};
    While True
        # S5 请求计算抓取放置
        ws_id = 0;
        api.xyzReqPickPlace(ws_id,err_code);
        If err_code <> 0
            Halt 3;
            Goto L[1];
        EndIf;
        # S6 获取抓取入筐轨迹
        ws_id = 0;
        api.xyzGetPickin(ws_id,err_code,pipeline_num,register_num,num,wp_type,pos);
        If err_code <> 0
            Halt 3;
            Goto L[1];
```

(下页继续)

(续上页)

```
        EndIf;
        # S7
        If num < 1
            # S15 如果没有可以抓取的工件
            Goto L[1];
        EndIf;
        # S8 执行抓取入筐轨迹
        api.xyzExecutePickInTraj(num, wp_type, pos);
        # io operation
        Set Out[1],OFF;
        # S9 获取抓取出筐轨迹
        ws_id = 0;
        api.xyzGetPickout(ws_id,err_code,pipeline_num,register_num,num,wp_type,pos);
        If err_code <> 0
            Halt 3;
            Goto L[1];
        EndIf;
        # S10 执行抓取出筐轨迹
        api.xyzExecutePickOutTraj(num, wp_type, pos);
        # S11 获取放置入筐轨迹
        ws_id = 0;
        api.xyzGetPlacein(ws_id,err_code,pipeline_num,register_num,num,wp_type,pos);
        If err_code <> 0
            Halt 3;
            Goto L[1];
        EndIf;
        # S12 执行放置入筐轨迹
        api.xyzExecuteTraj(num, wp_type, pos);
        # io operation
        Set Out[1],ON;
        # S13 获取放置出筐轨迹
        ws_id = 0;
        api.xyzGetPlaceout(ws_id,err_code,pipeline_num,register_num,num,wp_type,pos);
        If err_code <> 0
            Halt 3;
            Goto L[1];
        EndIf;
        # S14 执行放置出筐轨迹
        api.xyzExecuteTraj(num, wp_type, pos);
    EndWhile;
    L[1]:
    Close Socket, gMasterPort;
EndFunc;
```

轨迹移动异步模板

```
Include "api.pro";
Func run()
    # S1,S2: Init And conect To IPC 初始化以及连接工控机
    LB[1] = 0;
    While LB[1] <>1
        Open Socket("192.168.37.101",11111,2,Single, LB[1]);
    EndWhile;
    gMasterPort = 2;
```

(下页继续)

(续上页)

```
    Get Port[gMasterPort],T[0.2],Goto L[2];
    L[2]:
    If GetPortState(gMasterPort) <> 1
        Halt 3;
        Goto L[1];
    EndIf;
    Int err_code;
    gMasterFlag = True;
    #-----------------------------------------#
    #    String flow_name ="traj_async.t";
    #    api.xyzSwitchFlow(flow_name, err_code);
    #    If err_code <> 0
    #        Halt 3;
    #        Goto L[1];
    #    EndIf;
    # S3 切换工件
    Int ws_id = 0;
    String item_codename = "item1";
    api.xyzSwitchItem(ws_id, item_codename, err_code);
    If err_code <> 0
        Halt 3;
        Goto L[1];
    EndIf;
    # io operation
    Set   Out[1],ON;
    # S4: move To Home 回原点
    LP[1] = (0, 0, 0, 0, -90, 0),(0,0,0,1),(1,0,0);
    Movj  LP[1], V[30], Z[0], Tool[0];
    Int pipeline_num = 0;
    Int register_num = 0;
    Int num = 0;
    Int wp_type[30]={};
    Double pos[30][6]={};
    While True
        # S5 请求计算抓取放置
        ws_id = 0;
        api.xyzReqPickPlace(ws_id,err_code);
        If err_code <> 0
            Halt 3;
            Goto L[1];
        EndIf;
        # S6 获取抓取入筐轨迹
        ws_id = 0;
        api.xyzGetPickin(ws_id,err_code,pipeline_num,register_num,num,wp_type,pos);
        If err_code <> 0
            Halt 3;
            Goto L[1];
        EndIf;
        # S7
        If num < 1
            # S16 如果没有可以抓取的工件
            Goto L[1];
        EndIf;
        # S8 执行抓取入筐轨迹
        api.xyzExecutePickInTraj(num, wp_type, pos);
        # io operation
```

(下页继续)

(续上页)

```
        Set   Out[1],OFF;
        # S9 请求抓取出筐轨迹
        ws_id = 0;
        api.xyzGetPickout(ws_id,err_code,pipeline_num,register_num,num,wp_type,pos);
        If err_code <> 0
            Halt 3;
            Goto L[1];
        EndIf;
        # S10 执行抓取出筐轨迹
        api.xyzExecutePickOutTraj(num, wp_type, pos);
        # S11: request next pick and place plan in advance 请求下一次抓取放置
        ws_id = 0;
        api.xyzReqPickPlace(ws_id,err_code);
        If err_code <> 0
            Halt 3;
            Goto L[1];
        EndIf;
        # S12 获取放置入筐轨迹
        ws_id = 0;
        api.xyzGetPlacein(ws_id,err_code,pipeline_num,register_num,num,wp_type,pos);
        If err_code <> 0
            Halt 3;
            Goto L[1];
        EndIf;
        # S13 执行放置入筐轨迹
        api.xyzExecuteTraj(num, wp_type, pos);
        # io operation
        Set Out[1],ON;
        # S14 获取放置出筐轨迹
        ws_id = 0;
        api.xyzGetPlaceout(ws_id,err_code,pipeline_num,register_num,num,wp_type,pos);
        If err_code <> 0
            Halt 3;
            Goto L[1];
        EndIf;
        # S15 执行放置出筐轨迹
        api.xyzExecuteTraj(num, wp_type, pos);
    EndWhile;
    L[1]:
    Close Socket, gMasterPort;
EndFunc;
```

常见问题

1. 程序运行过后，再次运行程序时，机械臂后台程序未完全归位导致无法正常连接。

   InoRobotLab 或示教器需要进行如下操作：

   - 查看 background.pro 是否卡在某一行。
   - 如果使用 InoRobotLab 出现此情况，则重复切换右侧控制面板的模式为 示教，看到 调试页 中的 同步到控制器为可点击状态时，点击 同步到控制器，再运行程序；

     或者在示教器上 示教-> 工程配置处，将此任务由 静态任务切换为 动态任务，再切换回 静态任务，
   - 确认 background.pro 代码已经同步到控制器。如果没有卡住，开始工控机主控的连接。

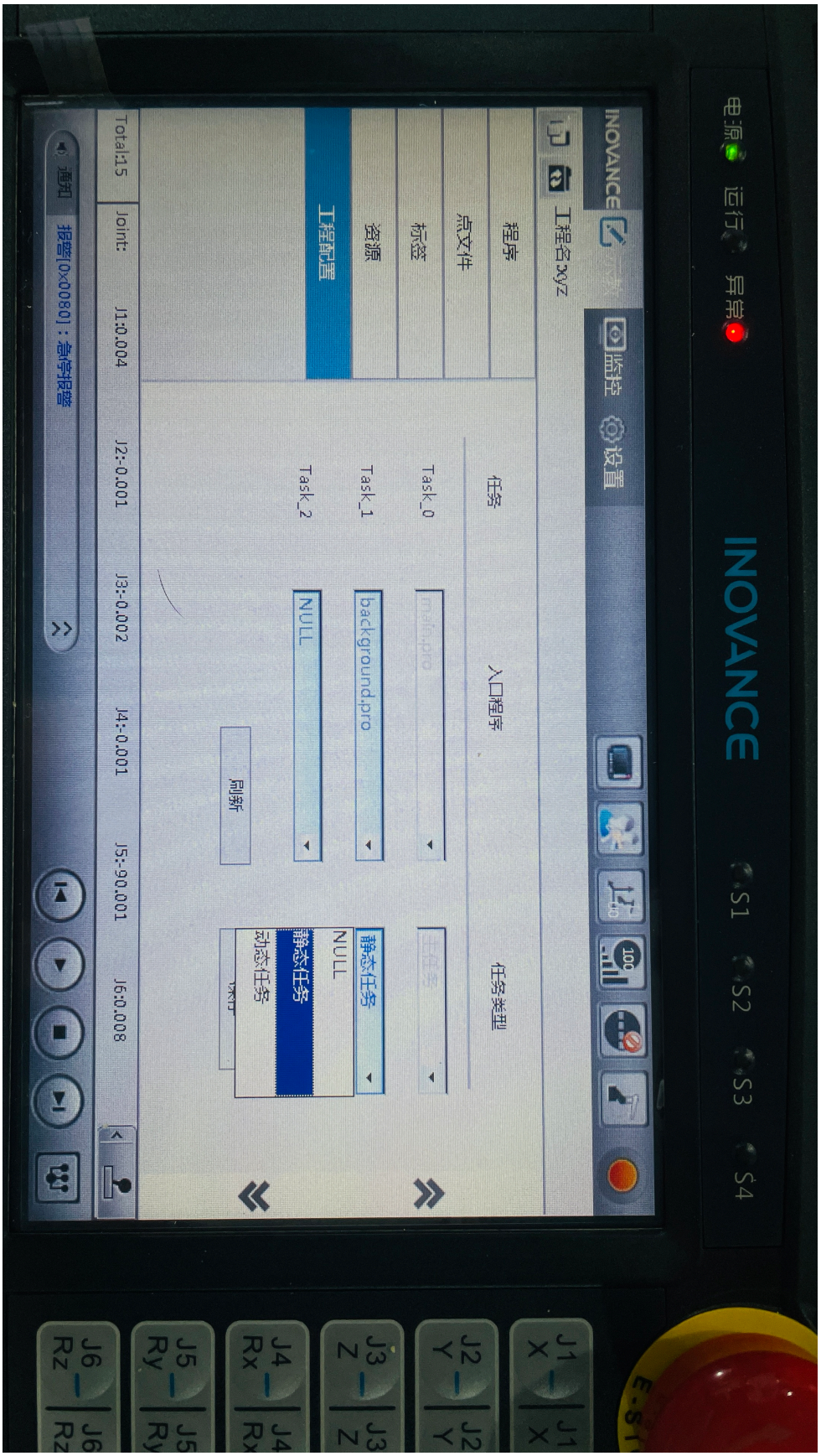

电源
运行
异常
INOVANCE
S1
S2
S3
S4
INOVANCE
监控
设置
工程名:xyz
程序
点文件
标签
资源
工程配置
任务
入口程序
任务类型
Task_0
Task_1
Task_2
main.pro
background.pro
NULL
刷新
静态任务
动态任务
NULL
Total:15
Joint:
J1:0.004
J2:-0.001
J3:-0.002
J4:-0.001
J5:90.001
J6:0.008
通知
报警(0x0080):急停报警
J1 X
J2 Y
J3 Z
J4 Rx
J5 Ry
J6 Rz

图 66: 示教器切换任务

2. 若程序导入示教器，出现大量报错，尤其是在函数定义或者函数调用处出错，可能是控制器版本太低
   解决方法：联系汇川售后，升级控制器版本为 S03.21R15 或更高版本.

附录

14.2.10 Jaka

此处介绍安装 jaka 机械臂驱动的相关事项。

驱动版本和使用要求

支持的机械臂类型

jaka 六轴机械臂

控制器型号

CBA7

机械臂需要开通的功能

控制器自带 socket 功能，无需单独设置。

安装驱动

jaka 机械臂驱动文件列表

jaka/
├── const.py —常量定义
├── jakaAPI.dll —jaka 机械臂 dll(64 位)
├── jakaAPI.lib —jaka 机械臂 lib(64 位)
├── jkrc.pyd —jaka 机械臂 python 接口 (64 位)
├── lib32 —32 位文件
│ ├── jakaAPI.dll —jaka 机械臂 dll(32 位)
│ ├── jakaAPI.lib —jaka 机械臂 lib(32 位)
│ └── jkrc.pyd —jaka 机械臂 python 接口 (32 位)
├── xyzCartMoveBasic.py —**座标移动基础模板**
├── xyzCartMoveReposition.py —**座标移动二次定位模板**
├── xyzTrajMoveAsync.py —**轨迹移动异步模板**
├── xyzTrajMoveSync.py —**轨迹移动同步模板**
├── xyz_master.py —机械臂主控 api 调用示例
└── xyz_master_object.py —jaka 机械臂类

设定机械臂 IP

1. 启动电控柜，长按手柄开关按键 1~2s，启动电控柜，待电控柜启动成功（手柄 JAKA 呼吸灯变为蓝色）。

2. 使用装好 JAKA Zu 软件的电脑或者安卓手机，接入 CAB 开头的 WIFI 网络。

3. 打开 JAKA Zu 软件，点击右上角的 未连接，弹出界面会显示可以连接的机械臂，点击连接，以管理员权限登陆，密码为：jakazuadmin。

图 67: JAKA Zu 软件连接机械臂

4. 进入 设置 -> 系统设置 -> 网络设置，依次设置 IP 地址、子网掩码、网关，配置完成后重启机械臂。

图 68: JAKA Zu 软件设定机械臂 IP

运行程序

通讯说明

目前程序只适配了机械臂主控代码。

工控机 robot server 和 xyz_master 的通讯方式为 socket：工控机作为 socket server(服务端)，xyz_master 作为 socket client（客户端）。

xyz_master 调用节卡 SDK 来控制机械臂，服务端 ip 和端口均在代码中指定。

运行机械臂主控

将节卡机械臂代码拷贝到任意文件夹下，打开 MAX 的 robot server 进程后，运行 **自行修改**后的模板代码。

API 说明

jaka 机械臂主控支持的 API

xyzSwitchApp(*app_id*)

切换应用

参数 **app_id**(*string*) –应用名称

返回 err_code

返回类型 INT

xyzSwitchFlow(*flow_name*)

切换任务

参数 **flow_name**(*string*) –任务 id

返回 err_code

返回类型 INT

xyzSwitchTool(*tool_id*)

切换工具

参数 **tool_id** –工具名称

返回 err_code

返回类型 INT

xyzReqCapImg(*vision_service_id*)

请求拍照

参数 **vision_service_id**(*INT*) –视觉服务 id

返回

err_code 错误代码

token 请求拍照结果时使用的 token

xyzGetCapImg(*token*)

获取拍照结果

参数 token(*INT*) –请求拍照时返回的 token

返回 err_code

返回类型 INT

xyzCapImg(*vision_service_id*)

拍照

参数 vision_service_id(*INT*) –需要进行拍照操作的视觉服务 id

返回 err_code

返回类型 INT

xyzReqGraspPose(*workspace_id*)

请求抓取位姿

参数 workspace_id(*INT*) –需要获取抓取点位的工作空间 id

返回

err_code 错误代码

token 返回的用于获取目标点位时使用的 token

xyzGetGraspPose(*token*)

获取抓取位姿

参数 token(*INT*) –求抓取目标点位时返回的 token

返回

err_code 错误代码

grasp_pose 抓取位姿

grasp_pose_num 可供抓取的点数量

pipeline_num pipeline 文件 number

register_num 用到的注册文件的注册 number

xyzReqObjPose(*workspace_id*)

请求物体位姿

参数 workspace_id(*INT*) –需要获取物体位姿的工作空间 id

返回

err_code 错误代码

obj_token 物体位姿识别的 token

xyzGetObjPose(*self*, *token*)

获取物体位姿

参数 token(*INT*) –请求物体位姿时得到的 token

返回

err_code 错误代码

obj_pose 物体位姿

obj_pose_num 物体数量

obj_pose_type 当前返回的物体 pose 类型

xyzResetTask()

重置任务

返回 err_code

返回类型 INT

xyzSendCurrentJoints()

发送机械臂当前角度：j1~j6: 机械臂当前的角度信息，如果机械臂轴数小于 6 轴，则需要对应位置补零，依旧发送六个数。

返回 err_code

返回类型 INT

xyzSendCurrentPose()

发送机械臂法兰当前位姿

返回 err_code

返回类型 INT

xyzSendCurrentExtJoints()

发送机械臂当前扩展轴位置：j1~j6: 机械臂当前扩展轴的角度信息，如果扩展轴数不足 6 的, 需要补零后发送六个数。

返回 err_code

返回类型 INT

xyzReqPick()

请求 pick 动作规划

返回 err_code

返回类型 INT

xyzReqPlace()

请求 place 动作规划

返回 err_code

返回类型 INT

xyzReqPickPlace(*ws_id*)

请求 pick 和 place 规划

参数 **ws_id**(*INT*) –工作空间 id

返回 err_code

返回类型 INT

xyzGetPickin(*ws_id*)

获取取料入框轨迹

参数 **ws_id**(*INT*) –工作空间 id

返回

err_code 错误代码

pipeline_num pipeline 文件 number

register_num 用到的注册文件的注册 number

traj_num 轨迹点数

wp_type 轨迹点类型（数组）

wp 轨迹点（数组）

xyzGetPickout(*ws_id*)

获取取料出框轨迹

参数 ws_id(*INT*) –工作空间 id

返回

err_code 错误代码

pipeline_num pipeline 文件 number

register_num 用到的注册文件的注册 number

traj_num 轨迹点数

wp_type 轨迹点类型（数组）

wp 轨迹点（数组）

xyzGetPlacein(*ws_id*)

获取放料入框轨迹

参数 ws_id(*INT*) –工作空间 id

返回

err_code 错误代码

pipeline_num pipeline 文件 number

register_num 用到的注册文件的注册 number

traj_num 轨迹点数

wp_type 轨迹点类型（数组）

wp 轨迹点（数组）

xyzGetPlaceout(*ws_id*)

获取放料出框轨迹

参数 ws_id(*INT*) –工作空间 id

返回

err_code 错误代码

pipeline_num pipeline 文件 number

register_num 用到的注册文件的注册 number

traj_num 轨迹点数

wp_type 轨迹点类型（数组）

wp 轨迹点（数组）

xyzSwitchStrat(*strat_name*)

请求切换策略

参数 strat_name(*string*) –策略名称

返回 err_code

返回类型 INT

xyzUpdateTotePose()

料箱重定位

返回 err_code pose 料箱位姿

xyzUpdateObjPoseOnHand()

工件在上手的二次定位

返回 err_code

返回类型 INT

xyzUpdateObjPoseToHand()

工件不在手上的二次定位

返回

err_code 错误代码

pipeline_num pipeline 文件 number

register_num 用到的注册文件的注册 number

traj_num 轨迹点数

wp_type 轨迹点类型（数组）

wp 轨迹点（数组）

xyzGetObjPoseType()

获取工件姿态类型

返回

err_code

pose_type 姿态类型

xyzResetPalletStatus()

重置工业码垛状态

返回 err_code

返回类型 INT

xyzSwitchItem(*item_codename*)

切换工件

参数 item_codename(*string*) –工件名称

返回 err_code

返回类型 INT

xyzCalculateGraspPose(*ws_id*)

计算抓取位姿

参数 ws_id(*INT*) –工作空间 id

返回

err_code 错误代码

grasp_pose 抓取位姿

grasp_pose_num 可供抓取的点数量

pipeline_num pipeline 文件 number

register_num 用到的注册文件的注册 number

xyzCalculateObjPose(*ws_id*)

计算物体位姿

参数 ws_id(*INT*) –工作空间 id

返回

err_code 错误代码

obj_pose 物体位姿

obj_pose_num 物体数量

obj_pose_type 当前返回的物体 pose 类型

案例/模板说明

机械臂主控主函数说明

轨迹移动同步模板

```
import os
import sys

JAKA_PYTHONPATH = os.path.join(os.path.dirname(__file__)) # 添加当前路径到环境变量
sys.path.append(JAKA_PYTHONPATH)
import socket

import const
import jkrc
import xyz_master_object

jk = xyz_master_object.jk_master("192.168.37.100") # 创建机械臂对象，传入机械臂 ip␣
↪地址
server_address = ("127.0.0.1", 11111) # 指定 robot server 的 ip 和端口
jk.socketConnect(server_address)
try:
    err_code = jk.xyzSwitchFlow("traj_sync.t") # 切换任务
    if err_code != 0:
        os.system("pause") # 错误代码不为零，暂停程序
    ws_id = 0
    err_code = jk.xyzSwitchItem(ws_id,"item1") # 切换工件
    if err_code != 0:
```

(下页继续)

(续上页)

```
        os.system("pause")
    IO_CABINET = 0
    IO_TOOL = 1 #
    IO_EXTEND = 2
    jk.robot.set_digital_output(IO_CABINET, 0, 0) # 设置数字输出
    home_joints = [0,0,0,0,-90,0]
    move_mode = 0
    is_block = True
    speed = 0.1 # unit: rad/s
    jk.robot.joint_move(home_joints, move_mode, is_block, speed) # 移动到初始位置
    while True:
        ws_id = 0
        err_code = jk.xyzReqPickPlace(ws_id) # 请求计算抓取放置轨迹
        if err_code != 0:
            os.system("pause")

        err_code, pipeline_num, register_num, traj_num, wp_type, wp = jk.
↪xyzGetPickin(0) # 获取抓取入筐轨迹
        if err_code != 0:
            os.system("pause")
        if traj_num < 1: # 如果没有轨迹，退出循环
            break

        jk.xyzExecutePickinTraj(traj_num, wp_type, wp) # 执行抓取入筐轨迹
        jk.robot.set_digital_output(IO_CABINET, 0, 1) # 抓取

        err_code, pipeline_num, register_num, traj_num, wp_type, wp = jk.
↪xyzGetPickout(0) # 获取抓取出筐轨迹
        if err_code != 0:
            os.system("pause")
        jk.xyzExecutePickoutTraj(traj_num, wp_type, wp) # 执行抓取出筐轨迹

        err_code, pipeline_num, register_num, traj_num, wp_type, wp = jk.
↪xyzGetPlacein(0) # 获取放置入筐轨迹
        if err_code != 0:
            os.system("pause")
        jk.xyzExecuteTraj(traj_num, wp_type, wp) # 执行放置入筐轨迹
        jk.robot.set_digital_output(IO_CABINET, 0, 0) # 放置

        err_code, pipeline_num, register_num, traj_num, wp_type, wp = jk.
↪xyzGetPlaceout(0) # 获取放置出筐轨迹
        if err_code != 0:
            os.system("pause")
        jk.xyzExecuteTraj(traj_num, wp_type, wp) # 执行放置出筐轨迹

finally:
    jk.robot.logout()
    print("robot disconnected")
```

轨迹移动异步模板

```
import os
import sys

JAKA_PYTHONPATH = os.path.join(os.path.dirname(__file__)) # 添加当前路径到环境变量
sys.path.append(JAKA_PYTHONPATH)
import socket

import const
import jkrc
import xyz_master_object

jk = xyz_master_object.jk_master("192.168.37.100") # 创建机械臂对象，传入机械臂 ip␣
↪地址
server_address = ("127.0.0.1", 11111) # 指定 robot server 的 ip 和端口
jk.socketConnect(server_address)
try:
    err_code = jk.xyzSwitchFlow("traj_async.t")  # 切换任务
    if err_code != 0:
        os.system("pause")
    ws_id = 0
    err_code = jk.xyzSwitchItem(ws_id,"item1") # 切换工件
    if err_code != 0:
        os.system("pause")
    IO_CABINET = 0
    IO_TOOL = 1 #
    IO_EXTEND = 2
    jk.robot.set_digital_output(IO_CABINET, 0, 0) # 设置数字输出
    home_joints = [0,0,0,0,-90,0]
    move_mode = 0
    is_block = True
    speed = 0.1 # unit: rad/s
    jk.robot.joint_move(home_joints, move_mode, is_block, speed) # 移动到初始位置
    while True:
        ws_id = 0
        err_code = jk.xyzReqPickPlace(ws_id) # 请求计算抓取放置轨迹
        if err_code != 0:
            os.system("pause")

        ws_id = 0
        err_code, pipeline_num, register_num, traj_num, wp_type, wp = jk.
↪xyzGetPickin(ws_id) # 获取抓取入筐轨迹
        if err_code != 0:
            os.system("pause")
        if traj_num < 1: # 如果没有轨迹，退出循环
            break

        jk.xyzExecutePickinTraj(traj_num, wp_type, wp) # 执行抓取入筐轨迹
        jk.robot.set_digital_output(IO_CABINET, 0, 1) # 抓取

        ws_id = 0
        err_code, pipeline_num, register_num, traj_num, wp_type, wp = jk.
↪xyzGetPickout(ws_id) # 获取抓取出筐轨迹
        if err_code != 0:
            os.system("pause")
        jk.xyzExecutePickoutTraj(traj_num, wp_type, wp) # 执行抓取出筐轨迹
```

(下页继续)

(续上页)

```
        ws_id = 0
        err_code = jk.xyzReqPickPlace(ws_id) # 提前请求下一次抓取放置轨迹
        if err_code != 0:
            os.system("pause")

        ws_id = 0
        err_code, pipeline_num, register_num, traj_num, wp_type, wp = jk.
↪xyzGetPlacein(ws_id) # 获取放置入筐轨迹
        if err_code != 0:
            os.system("pause")
        jk.xyzExecuteTraj(traj_num, wp_type, wp)     # 执行放置入筐轨迹
        jk.robot.set_digital_output(IO_CABINET, 0, 0) # 放置

        ws_id = 0
        err_code, pipeline_num, register_num, traj_num, wp_type, wp = jk.
↪xyzGetPlaceout(ws_id) # 获取放置出筐轨迹
        if err_code != 0:
            os.system("pause")
        jk.xyzExecuteTraj(traj_num, wp_type, wp) # 执行放置出筐轨迹

finally:
    jk.robot.logout()
    print("robot disconnected")
```

座标移动基础模板

```
import os
import sys

JAKA_PYTHONPATH = os.path.join(os.path.dirname(__file__)) # 添加当前路径到环境变量
sys.path.append(JAKA_PYTHONPATH)
import socket

import const
import jkrc
import xyz_master_object

jk = xyz_master_object.jk_master("192.168.37.100") # 创建机械臂对象，传入机械臂 ip␣
↪地址
server_address = ("127.0.0.1", 11111) # 指定 robot server 的 ip 和端口
jk.socketConnect(server_address)
try:
    err_code = jk.xyzSwitchFlow("catr_basic.t") # 切换任务
    if err_code != 0:
        os.system("pause")
    ws_id = 0
    err_code = jk.xyzSwitchItem(ws_id,"item1") # 切换工件
    if err_code != 0:
        os.system("pause")
    IO_CABINET = 0
    IO_TOOL = 1 #
    IO_EXTEND = 2
    jk.robot.set_digital_output(IO_CABINET, 0, 0) # 设置数字输出
```

(下页继续)

(续上页)

```
    home_joints = [0,0,0,0,-90,0]
    move_mode = 0
    is_block = True
    speed = 0.1 # unit: rad/s
    jk.robot.joint_move(home_joints, move_mode, is_block, speed) # 移动到初始位置

    is_eye_in_hand = True

    while True:

        if is_eye_in_hand:
            scan_pose = [400,0,400,0,0,0]
            speed = 10 # unit: mm/s
            jk.robot.linear_move(scan_pose, move_mode, is_block, speed) #␣
↪移动到拍照位姿
            err_code = jk.xyzSendCurrentPose() # 发送当前位姿
            if err_code != 0:
                os.system("pause")

        err_code, grasp_token = jk.xyzReqGraspPose(ws_id) # 请求计算抓取位姿
        if err_code != 0:
            os.system("pause")
        err_code, grasp_pose, grasp_pose_num, pipeline_num, register_num = jk.
↪xyzGetGraspPose(grasp_token) # 获取抓取位姿
        if err_code != 0:
            os.system("pause")

        if grasp_pose_num < 1: # 如果没有位姿，继续请求
            print("No grasp pose, continure requesting...")
            continue

        speed = 10 # unit: mm/s
        jk.robot.linear_move(grasp_pose, move_mode, is_block, speed) # 移动到抓取位姿
        jk.robot.set_digital_output(IO_CABINET, 0, 1) # 抓取

        place_pose = [400,0,400,0,0,0]
        speed = 10 # unit: mm/s
        jk.robot.linear_move(place_pose, move_mode, is_block, speed) # 移动到放置位姿
        jk.robot.set_digital_output(IO_CABINET, 0, 0) # 放置

finally:
    jk.robot.logout()
    print("robot disconnected")
```

座标移动二次定位模板

```
import os
import sys

JAKA_PYTHONPATH = os.path.join(os.path.dirname(__file__)) # 添加当前路径到环境变量
sys.path.append(JAKA_PYTHONPATH)
import socket

import const
```

(下页继续)

(续上页)

```
import jkrc
import xyz_master_object

jk = xyz_master_object.jk_master("192.168.37.100") # 创建机械臂对象，传入机械臂 ip 地址
server_address = ("127.0.0.1", 11111) # 指定 robot server 的 ip 和端口
jk.socketConnect(server_address)
try:
    err_code = jk.xyzSwitchFlow("cart_repo.t") # 切换任务
    if err_code != 0:
        os.system("pause")

    IO_CABINET = 0
    IO_TOOL = 1 #
    IO_EXTEND = 2
    jk.robot.set_digital_output(IO_CABINET, 0, 0) # 设置数字输出0状态
    jk.robot.set_digital_output(IO_CABINET, 1, 0) # 设置数字输出1状态
    home_joints = [0,0,0,0,-90,0]
    move_mode = 0
    is_block = True
    speed = 0.1 # unit: rad/s
    jk.robot.joint_move(home_joints, move_mode, is_block, speed) # 移动到初始位置

    ws_id = 0
    err_code = jk.xyzSwitchItem(ws_id,"item1") # 切换工件
    if err_code != 0:
        os.system("pause")
    err_code, rough_grasp_token = jk.xyzReqGraspPose(ws_id) # 请求计算抓取位姿
    if err_code != 0:
        os.system("pause")

    while True:
        # camera 1 get rough grasp pose 相机1获取粗抓取位姿
        err_code, rough_grasp_pose, rough_grasp_pose_num, rough_pipeline_num, rough_register_num = jk.xyzGetGraspPose(rough_grasp_token)
        if err_code != 0:
            os.system("pause")

        # no item, grasp board 没有工件，抓取隔板
        if rough_grasp_pose_num < 1:
            ws_id = 0
            err_code = jk.xyzSwitchItem(ws_id,"board") # 切换工件
            if err_code != 0:
                os.system("pause")
            err_code, board_grasp_token = jk.xyzReqGraspPose(ws_id) # 请求计算抓取位姿
            if err_code != 0:
                os.system("pause")
            # 获取隔板抓取位姿
            err_code, board_grasp_pose, board_grasp_pose_num, board_pipeline_num, board_register_num = jk.xyzGetGraspPose(board_grasp_token)
            if err_code != 0:
                os.system("pause")

            if board_grasp_pose_num < 1:
                # no board, you can change tote # 没有隔板，可以切换料箱
                break
```

(下页继续)

(续上页)

```
            speed = 10 # unit: mm/s
            jk.robot.linear_move(board_grasp_pose, move_mode, is_block, speed)    #␣
↪移动到隔板抓取位姿
            jk.robot.set_digital_output(IO_CABINET, 0, 1) # 抓取
            # place board 放置隔板
            board_place_pose = [400,0,400,0,0,0]
            speed = 10 # unit: mm/s
            jk.robot.linear_move(board_place_pose, move_mode, is_block, speed) #␣
↪移动到隔板放置位姿
            jk.robot.set_digital_output(IO_CABINET, 0, 0) # 放置

            ws_id = 0
            err_code = jk.xyzSwitchItem(ws_id,"item1") # 切换工件
            if err_code != 0:
                os.system("pause")
            err_code, rough_grasp_token = jk.xyzReqGraspPose(ws_id) # 请求粗抓取位姿
            if err_code != 0:
                os.system("pause")
            continue

        # move to rough grasp pose 移动到粗抓取位姿
        speed = 10 # unit: mm/s
        jk.robot.linear_move(rough_grasp_pose, move_mode, is_block, speed)
        err_code = jk.xyzSendCurrentPose() # 发送当前位姿

        # camera2 get grasp pose 相机2获取精确抓取位姿
        ws_id = 1
        err_code = jk.xyzSwitchItem(ws_id,"item1")
        if err_code != 0:
            os.system("pause")
        err_code, fine_grasp_token = jk.xyzReqGraspPose(ws_id) # 请求精确抓取位姿
        if err_code != 0:
            os.system("pause")
        # 获取精确抓取位姿
        err_code, fine_grasp_pose, fine_grasp_pose_num, fine_pipeline_num, fine_
↪register_num = jk.xyzGetGraspPose(fine_grasp_token)
        if err_code != 0:
            os.system("pause")
        if fine_grasp_pose_num < 1: # 如果没有位姿，退出循环
            break

        # move to fine grasp pose 移动到精确抓取位姿
        speed = 10 # unit: mm/s
        jk.robot.linear_move(fine_grasp_pose, move_mode, is_block, speed) #␣
↪移动到精确抓取位姿
        jk.robot.set_digital_output(IO_CABINET, 1, 1) # 抓取

        # place item 放置工件
        item_place_pose = [400,0,400,0,0,0]
        speed = 10 # unit: mm/s
        jk.robot.linear_move(item_place_pose, move_mode, is_block, speed) #␣
↪移动到放置位姿
        jk.robot.set_digital_output(IO_CABINET, 1, 0) # 放置

        ws_id = 0
```

(下页继续)

(续上页)

```
        err_code = jk.xyzSwitchItem(ws_id,"item1") # 切换工件
        err_code, rough_grasp_token = jk.xyzReqGraspPose(ws_id) # 请求相机1粗抓取位姿


finally:
    jk.robot.logout()
    print("robot disconnected")
```

常见问题

附录

14.2.11 Kawasaki

此处介绍安装 kawasaki 机械臂驱动的相关事项。

驱动版本和使用要求

支持的机械臂类型

Kawasaki 六轴机械臂、四轴机械臂（拆码垛机械臂）

控制器型号

E20

机械臂需要开通的功能

Kawasaki 自带 socket 通讯功能，无需开通。

安装驱动

kawasaki 机械臂驱动文件列表

–**kawasaki**

- xyz_lib.as （全局常量、通讯函数）
- xyz_master.as （机械臂主控程序）
- xyz_motion.as （工控机主控前台程序）
- xyz_status.as （工控机主控后台程序）

设定机械臂 IP

菜单-> 辅助功能-> 系统-> 网络设定

- "IP" 设置为：192.168.37.100
- "子网掩码" 设置为：255.255.255.0

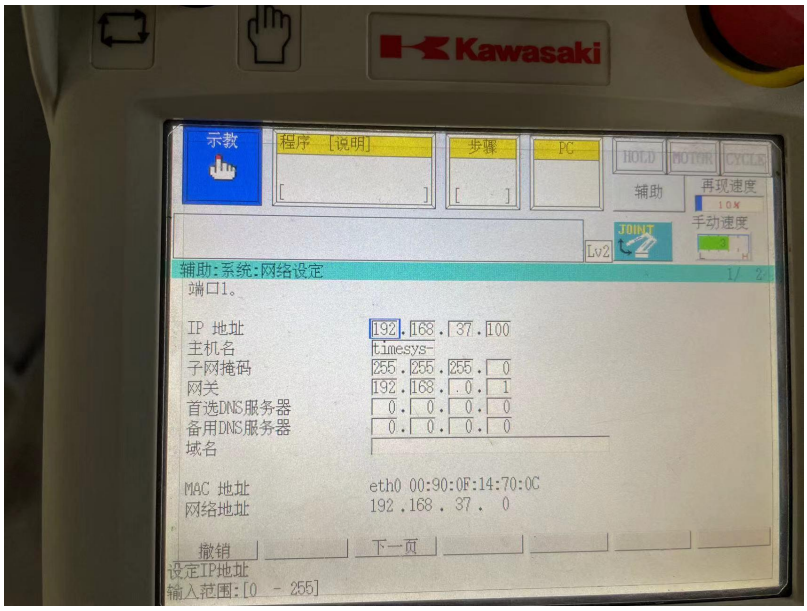


图 69: kawasaki 设定机械臂 IP

使用 KRterm 导入程序

1. 用网线连接装好 KRterm 的电脑和控制柜上的 **网线接口**。
2. 将电脑的 ipv4 网络设定为:

- "IP" 设置为：192.168.37.100
- "子网掩码" 设置为：255.255.255.0
- "默认网关" 设置为：192.168.37.0

3. 启动 KRterm，依次点击 `Com-> Options-> TCP/IP`, 将 IP 设为和机械臂 IP 一致，请 **不要修改** Port No, 填写完后依次点击 `Add-> Ok`。

3. 点击 `Com-> Connect by list`，选择刚刚设定的配置，接着在 terminal 中输入 `as` 即可登录：

4. 在 terminal 中输入 `load 文件路径`，即可将程序载入到示教器。

   ！注意：

   - 文件目录中的四个文件要严格按照 `xyz_lib.as-> xyz_status.as-> xyz_motion.as-> xyz_master.as` 的顺序导入。

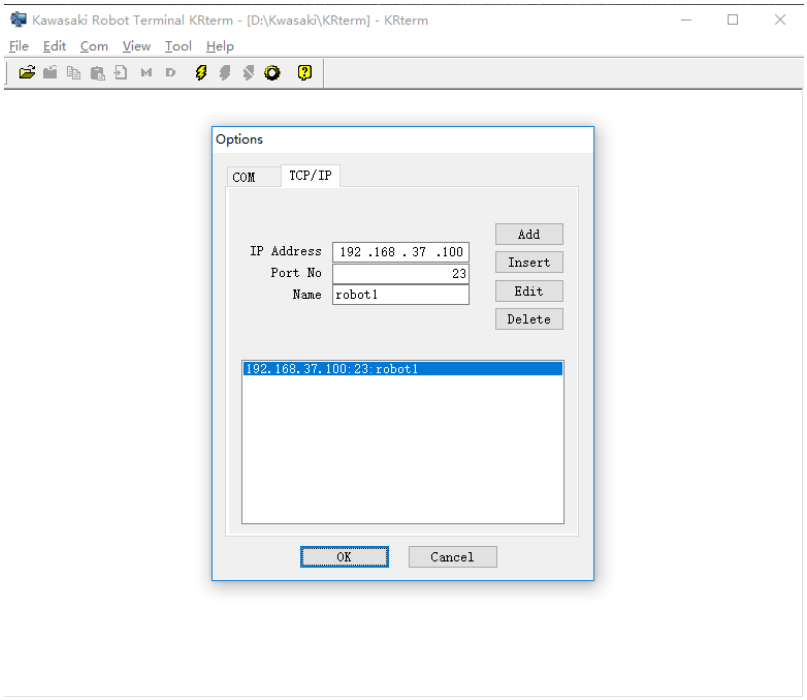


图 70: KRterm 连接 kawasaki 机械臂

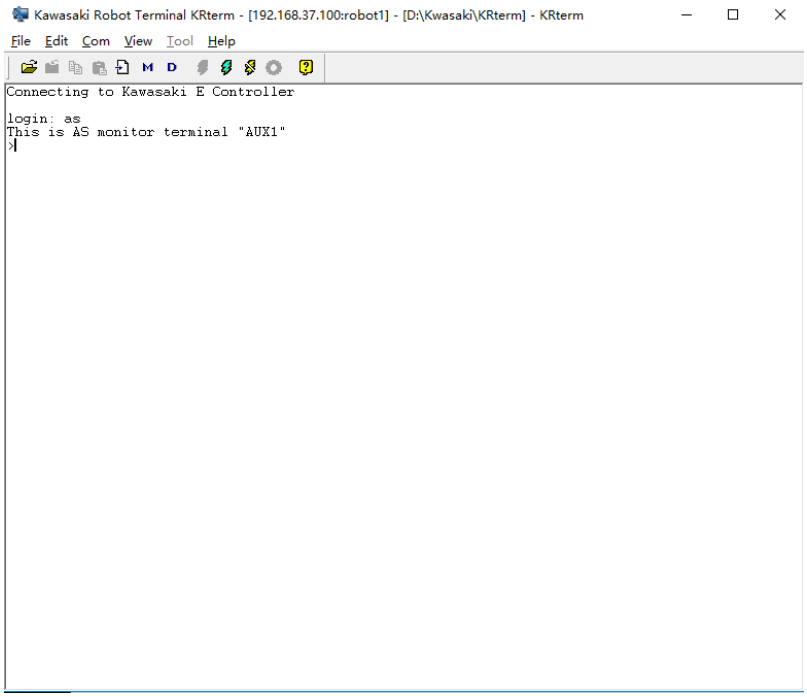


图 71: KRterm 连接 kawasaki 机械臂

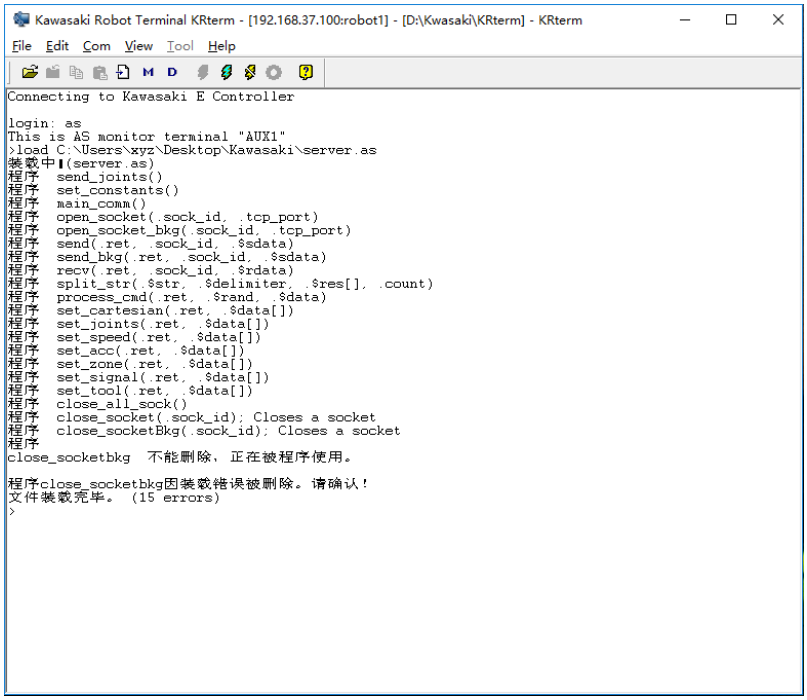


图 72: KRterm 载入 kawasaki 机械臂程序

运行程序

通讯说明

工控机和 kawasaki 机械臂的通讯方式为 TCP/IP：工控机作为 socket server（服务端），机械臂作为 socket client（客户端）。

启动程序

运行工控机主控

启动程序，程序 **->** 登录程序**在程序清单中选中 xyz_motion。然后在控制柜和示教器切换再现模式（左上角变绿），解除锁**

！注意：

- 川崎机械臂执行多任务，任务 1 为主任务，任务 2 为后台任务并由任务 1 启动。其中任务 1 可以直接在示教器的程序栏停止并取消登录。任务 2 需要在系统后台关闭，可以操作示教器，也可以在 KRterm 上在线关闭（仅在需要时清除后台任务）。
- 进入 菜单 -> 辅助 -> 系统 -> PC 程序启动/停止，分别进行 执行中断（PCABORT） -> 执行停止（PCEND） -> 登录注销（PCKILL）并选中任务 2，即可关闭程序。
- KRterm 也可以关闭后台任务，即在登录状态下，输入 PCABORT 2:、PCEND 2: 和 PCKILL 2: 命令，查看示教器可以看到 PC 栏的任务已经清除。

图 73: 启动程序

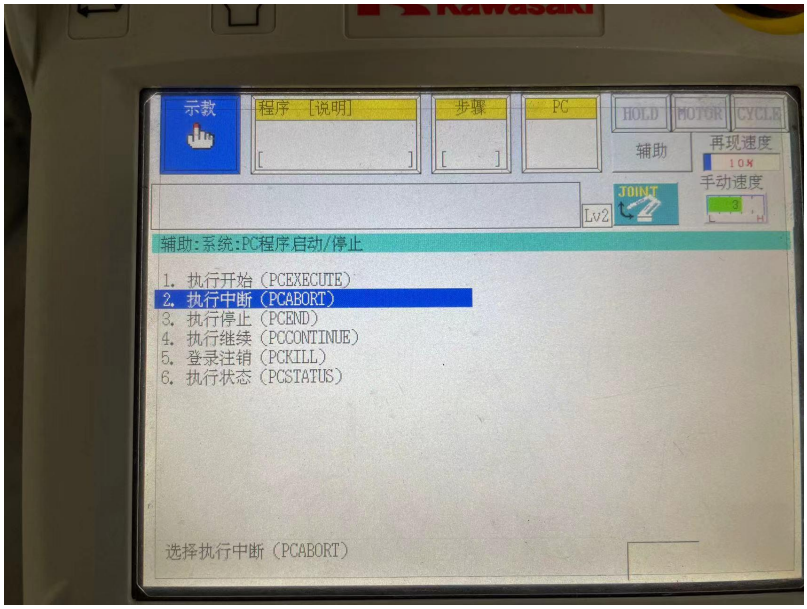


图 74: 启动程序

运行机械臂主控

同工控机主控，只需要将选择的程序换为 `xyz_master`, 机械臂主控不是多任务，不需要按工控机主控的方式终止程序。

API 说明

kawasaki 工控机主控支持的 API

指令号	命令	支持情况
100	获取机械臂程序版本号	支持
101	发送速度数据	支持
102	发送加速度数据	支持
103	发送圆滑过渡参数	支持
104	发送工具坐标系 (TCP) 参数	支持
105	设置数字量输出	支持
106	SetJointsMovej	支持
107	SetCartMovel	支持
108	SetJoinsMovel	支持
109	SetCartMovej	不支持
110	MovejSequence	不支持
111	MovelSequence	不支持
112	SetJointsMovejDo	不支持
113	SetCartMovelDo	不支持
114	SetJointsMovelDo	不支持
115	SetJointsMovejGroupDo	不支持
116	SetCartMovelGroupDo	不支持
117	SetJointsMovelGroupDo	不支持
118	MovelUntil	不支持
119	获取数字量输入状态	不支持
120	获取数字量输出状态	支持
121	获取模拟量输入口数值	不支持
122	获取机械臂当前角度	支持
123	获取机械臂当前位姿	支持
200	机械臂后台发送状态	支持

kawasaki 机械臂主控支持的 API

.PROGRAM xyz_sw_app(.error_code, .$app_name)

切换应用

参数 app_name (*string*) –应用名称

返回 .error_code

返回类型 num

.PROGRAM xyz_sw_item(.error_code, .$item_name)

切换工件

参数 item_name –工件名称

返回 .error_code

返回类型 num

.PROGRAM xyz_sw_tool(.error_code, .$tool_name)

切换工具

参数 **tool_name** (*string*) –工具名称

返回 .error_code

返回类型 num

.PROGRAM xyz_r_capimg(.error_code, .ws_id, .token)

请求拍照

参数

- **ws_id** (*num*) –工作空间 id
- **token** (*num*) –获取拍照结果的凭据

返回 error_code

返回类型 num

.PROGRAM xyz_g_capimg(.error_code, .token)

获取拍照结果

参数 **token** (*num*) –请求拍照时返回的 token

返回 error_code

返回类型 num

.PROGRAM xyz_capimg(.error_code)

拍照

参数 **ws_id** (*num*) –需要进行拍照操作的工作空间 id

返回 error_code

返回类型 num

.PROGRAM xyz_r_g_pose(.error_code, .ws_id, .token)

请求抓取位姿

参数

- **ws_id** (*num*) –需要获取抓取点位的工作空间 id
- **token** (*num*) –返回的用于获取目标点位时使用的 token

返回 error_code

返回类型 num

.PROGRAM xyz_g_g_pose(.error_code, .token, .pose_num, .pose_type)

获取抓取位姿

参数

- **token** (*num*) –请求抓取目标点位时返回的 token
- **pose_num** (*num*) –可供抓取的点数量
- **pose_type** (*num*) –当前返回的抓取点的 pose 类型

- **grasp_pose** (*XYZOAT*) –抓取位姿（全局变量）

返回 error_code

返回类型 num

.PROGRAM xyz_r_o_pose(.error_code, .ws_id, .token)

请求物体位姿

参数

- **ws_id** (*num*) –需要获取物体位姿的工作空间 id
- **token** (*num*) –物体位姿识别的 token

返回 error_code

返回类型 num

.PROGRAM xyz_g_o_pose(.error_code, .token, .pose_num, .pose_type)

获取物体位姿

参数

- **token** (*num*) –请求物体位姿时得到的 token
- **pose_num** (*num*) –物体数量
- **pose_type** (*num*) –当前返回的物体 pose 类型
- **object_pose** (*XYZOAT*) –物体位姿（全局变量）

返回 error_code

返回类型 num

.PROGRAM xyz_reset_v(.error_code, .ws_id)

重置视觉

参数 **ws_id** (*num*) –需要重置视觉的工作空间 id

返回 error_code

返回类型 num

.PROGRAM xyz_s_joints(.error_code, .#joints)

发送特定关节位姿

参数 **joints** (关节位姿变量) –需要发送的关节位姿

返回 error_code

返回类型 num

.PROGRAM xyz_s_c_pose(.error_code)

发送特定笛卡尔空间位姿

参数 **sending_pose** (*XYZOAT*) –需要发送的笛卡尔空间位姿（全局变量）

返回 error_code

返回类型 num

.PROGRAM xyz_s_ext_j(.error_code, .#ext_joints)

发送特定外部轴关节位姿

参数 **ext_joints** (关节位姿变量) –需要发送的关节位姿

返回 error_code

返回类型 num

.PROGRAM xyz_r_pick(.error_code)

请求 pick 动作规划

返回 error_code

返回类型 num

.PROGRAM xyz_r_place(.error_code)

请求 place 动作规划

返回 error_code

返回类型 num

.PROGRAM xyz_r_picpla(.error_code)

请求 pick 和 place 规划

返回 error_code

返回类型 num

.PROGRAM xyz_g_picin(.error_code, .pose_type, .wps_num, .wps_type[], .#jnts_traj[])

获取取料入框轨迹

参数

- **pose_num** (*num*) –轨迹点数
- **pose_type** (*num*) –轨迹点 pose 类型
- **wps_type[]** (*array*) –轨迹点类型数组
- **jnts_traj[]** (*array*) –轨迹点关节位姿数组
- **cart_traj[]** (*array*) –轨迹点笛卡尔空间位姿数组（全局变量）

返回 error_code

返回类型 num

.PROGRAM xyz_g_picout(.error_code, .pose_type, .wps_num, .wps_type[], .#jnts_traj[])

获取取料出框轨迹

参数

- **pose_num** (*num*) –轨迹点数
- **pose_type** (*num*) –轨迹点 pose 类型
- **wps_type[]** (*array*) –轨迹点类型数组
- **jnts_traj[]** (*array*) –轨迹点关节位姿数组
- **cart_traj[]** (*array*) –轨迹点笛卡尔空间位姿数组（全局变量）

返回 error_code

返回类型 num

.PROGRAM xyz_g_plain(.error_code, .pose_type, .wps_num, .wps_type[], .#jnts_traj[])

获取放料入框轨迹

参数

- **pose_num** (*num*) –轨迹点数
- **pose_type** (*num*) –轨迹点 pose 类型
- **wps_type[]** (*array*) –轨迹点类型数组
- **jnts_traj[]** (*array*) –轨迹点关节位姿数组
- **cart_traj[]** (*array*) –轨迹点笛卡尔空间位姿数组（全局变量）

返回 error_code

返回类型 num

.PROGRAM xyz_g_plaout(.error_code, .pose_type, .wps_num, .wps_type[], .#jnts_traj[])

获取放料出框轨迹

参数

- **pose_num** (*num*) –轨迹点数
- **pose_type** (*num*) –轨迹点 pose 类型
- **wps_type[]** (*array*) –轨迹点类型数组
- **jnts_traj[]** (*array*) –轨迹点关节位姿数组
- **cart_traj[]** (*array*) –轨迹点笛卡尔空间位姿数组（全局变量）

返回 error_code

返回类型 num

.PROGRAM xyz_sw_strat(.error_code, .$strat_name)

请求切换策略

参数 **strat_name** (*string*) –策略名称

返回 error_code

返回类型 num

.PROGRAM xyz_u_t_pose(.error_code)

料箱重定位

参数 **tote_pose** (*pose*) –料箱位姿 (全局变量)

返回 error_code

返回类型 num

.PROGRAM xyz_u_obj_oh(.error_code)

工件在上手的二次定位

返回 error_code

返回类型 num

.PROGRAM xyz_u_obj_th(.error_code, .pose_type, .wps_num, .wps_type[], .#jnts_traj[], .cart_traj[])

工件不在手上的二次定位，获取二次抓取的轨迹

参数

- **pose_num** (*num*) –轨迹点数
- **pose_type** (*num*) –轨迹点 pose 类型
- **wps_type[]** (*array*) –轨迹点类型数组
- **jnts_traj[]** (*array*) –轨迹点关节位姿数组
- **cart_traj[]** (*array*) –轨迹点笛卡尔空间位姿数组（全局变量）

返回 error_code

返回类型 num

.PROGRAM xyz_g_obj_pt(.error_code, .pose_type)

获取工件姿态类型

参数 **pose_type** (*num*) –工件姿态类型

返回 error_code

返回类型 num

.PROGRAM xyz_reset_ps(.error_code)

重置工业码垛状态

返回 error_code

返回类型 num

.PROGRAM xyz_exectraj(.cmd, .pose_type, .wps_num, .wps_type[], .#jnts_traj[])

用于执行一段轨迹，注意所有笛卡尔空间位姿与轨迹点皆为全局变量

参数

- **pose_num** (*num*) –轨迹点数
- **pose_type** (*num*) –轨迹点 pose 类型
- **wps_type[]** (*array*) –轨迹点类型数组
- **jnts_traj[]** (*array*) –轨迹点关节位姿数组
- **cart_traj[]** (*array*) –轨迹点笛卡尔空间位姿数组（全局变量）

返回 error_code

返回类型 num

案例/模板说明

机械臂主控主函数说明

以下为机械臂主控模板代码，注意对工控机返回的 err_code 进行判断。

```
.PROGRAM cartmovebasic()
;-------------------------------------------------------------------------
;cart move 基础模板
;-------------------------------------------------------------------------
    ; S1: 初始化参数
    CALL set_const
    CP ON
    MESSAGE ON
    CALL close_all_sock
    ; 相关运动参数需要合理设置
    SPEED 100 ALWAYS
    ACCEL 100 ALWAYS
    ACCURACY 10 ALWAYS
    ; 数字输出信号需要合理设置
    SIGNAL -1
    ; 一些示教点位需要合理设置
    POINT #HOME = #PPOINT(0,0,0,0,0,0)
    POINT cart_wp1 = TRANS(0,0,0,0,0,0) ; scan pose
    POINT cart_wp2 = TRANS(0,0,0,0,0,0)
    .eye_on_hand = 0 ; 眼在手上标志位

    ; S2: 连接到上位机
    CALL xyz_conn(sock_id, IP[], PORT)

    ; CALL xyz_sw_flw(.error_code, "test.t")
    ; IF .error_code <> 0 GOTO LABEL_FAIL

    ; S3: 切换工件
    CALL xyz_sw_item(.error_code, 0, "obj1")
    IF .error_code <> 0 GOTO LABEL_FAIL

    ; S4: 移动到home点，需要提前设置好home点位置
    JMOVE #HOME
    WHILE 1 DO
        LABEL_START:
        ; S5: 眼在手上
        IF .eye_on_hand == 1 THEN
            ; S6: 移动到拍照位
            ; you should define scan_pose first
            LMOVE cart_wp1
            POINT sending_pose = cart_wp1
            ; S7: 发送拍照位姿给上位机
            CALL xyz_s_c_pose(.error_code)
            IF .error_code <> 0 GOTO LABEL_FAIL
        END

        ; S8: 请求抓取位姿
        CALL xyz_r_g_pose(.error_code, 0, .token)
        IF .error_code <> 0 GOTO LABEL_FAIL

        ; S9: 获取抓取位姿
```

(下页继续)

(续上页)

```
        CALL xyz_g_g_pose(.error_code, .token, .pose_num, .pipe_num, .reg_num)
        IF .error_code <> 0 GOTO LABEL_FAIL
        IF .pose_num < 1 THEN
            TWAIT 5
            GOTO LABEL_START
        END
        ; S10: 移动到抓取位并抓取
        ; 注意，可能需要添加一些过渡点
        LMOVE grasp_pose
        SIGNAL 1

        ; S11: 移动到放置位并放置
        LMOVE cart_wp2
        SIGNAL -1
    END
LABEL_FAIL:
    CALL xyz_disconn(.ret, sock_id)
.END


.PROGRAM cartmoverepo()
;-------------------------------------------------------------------------
;cart move 重定位模板
;-------------------------------------------------------------------------
    ; S1: 初始化参数
    CALL set_const
    CP ON
    MESSAGE ON
    CALL close_all_sock
    ; 相关运动参数需要合理设置
    SPEED 100 ALWAYS
    ACCEL 100 ALWAYS
    ACCURACY 10 ALWAYS
    ; 数字输出信号需要合理设置
    SIGNAL -1
    SIGNAL -2
    ; 一些示教点位需要合理设置
    POINT #HOME = #PPOINT(0,0,0,0,0,0)
    POINT cart_wp1 = TRANS(0,0,0,0,0,0) ; scan pose
    POINT cart_wp2 = TRANS(0,0,0,0,0,0)

    ; S2: 连接到上位机
    CALL xyz_conn(sock_id, IP[], PORT)

    ;CALL xyz_sw_flw(.error_code, "test.t")
    ;IF .error_code <> 0 GOTO LABEL_FAIL

    ; S3: 切换工件
    CALL xyz_sw_item(.error_code, 0, "obj1")
    IF .error_code <> 0 GOTO LABEL_FAIL

    ; S4: 移动到home点，需要提前设置好home点位置
    JMOVE #HOME

    ; S5: 请求抓取位姿
    CALL xyz_r_g_pose(.error_code, 0, .token)
```

(下页继续)

(续上页)

```
IF .error_code <> 0 GOTO LABEL_FAIL

WHILE 1 DO
    LABEL_START:
    ; S6: 获取抓取位姿
    CALL xyz_g_g_pose(.error_code, .token, .pose_num, .pipe_num, .reg_num)
    IF .error_code <> 0 GOTO LABEL_FAIL

    ; S7:
    IF .pose_num < 1 THEN
        ;S15: 上位机切换至识别隔板
        ;工作空间中已经没有工件，需要先取走隔板
        CALL xyz_sw_item(.error_code, 0, "board")
        IF .error_code <> 0 GOTO LABEL_FAIL

        ; S16: 请求隔板抓取位姿
        CALL xyz_r_g_pose(.error_code, 0, .token)
        IF .error_code <> 0 GOTO LABEL_FAIL

        ; S17: 获取隔板抓取位姿
        CALL xyz_g_g_pose(.error_code, .token, .pose_num, .pipe_num, .reg_num)
        IF (.error_code <> 0) OR (.pose_num < 1) GOTO LABEL_FAIL

        ; S18: 移动到抓取位并抓取隔板
        LMOVE grasp_pose
        SIGNAL 1

        ; S19: 移动到放置位并放置
        LMOVE cart_wp2
        SIGNAL -1

        ;注意，可能需要添加一些过渡点

        ;S20: 上位机切换至识别工件
        CALL xyz_sw_item(.error_code, 0, "obj1")
        IF .error_code <> 0 GOTO LABEL_FAIL
        CALL xyz_r_g_pose(.error_code, 0, .token)
        IF .error_code <> 0 GOTO LABEL_FAIL
        GOTO LABEL_START
    ELSE
        ; S8: 移动到拍照位，需要先示教该拍照位
        LMOVE cart_wp1
        POINT sending_pose = cart_wp1

        ; S9: 发送拍照位姿
        CALL xyz_s_c_pose(.error_code)
        IF .error_code <> 0 GOTO LABEL_FAIL

        ; S10: 上位机切换至识别工件
        CALL xyz_sw_item(.error_code, 1, "obj1")
        IF .error_code <> 0 GOTO LABEL_FAIL

        ; S11: 请求抓取位姿
        CALL xyz_r_g_pose(.error_code, 1, .token)
        IF .error_code <> 0 GOTO LABEL_FAIL
```

(下页继续)

(续上页)

```
            ; S12: 获取抓取位姿
            CALL xyz_g_g_pose(.error_code, .token, .pose_num, .pipe_num, .reg_num)
            IF (.error_code <> 0) OR (.pose_num < 1) GOTO LABEL_FAIL

            ; S13: 移动到抓取位并抓取
            LMOVE grasp_pose
            SIGNAL 1

            ; S14: 移动到放置位并放置
            ; Attention: Additional waypoints should be added for place obj
            LMOVE cart_wp2
            SIGNAL -1
        END
        CALL xyz_sw_item(.error_code, 0, "obj1")
        IF .error_code <> 0 GOTO LABEL_FAIL
        CALL xyz_r_g_pose(.error_code, 0, .token)
        IF .error_code <> 0 GOTO LABEL_FAIL
    END
LABEL_FAIL:
    CALL xyz_disconn(.ret, sock_id)
.END


.PROGRAM trajmovesync()
;-------------------------------------------------------------------------
;traj move 同步模板
;-------------------------------------------------------------------------
    ; S1: 初始化参数
    CALL set_const
    CP ON
    MESSAGE ON
    CALL close_all_sock

    ; S2: 连接上位机
    CALL xyz_conn(sock_id, IP[], PORT)

    ; CALL xyz_sw_flw(.error_code, "test.t")
    ; IF .error_code <> 0 GOTO LABEL_FAIL

    ; S3: 切换工件
    CALL xyz_sw_item(.error_code, 0, "item1")
    IF .error_code <> 0 GOTO LABEL_FAIL

    ; 以下参数需要设置合理的初始值
    SIGNAL -1
    .error_code = 0
    .ws_id = 0

    ; S4: 移动到home点，需要提前设置好home点位置
    POINT #HOME = #PPOINT(0,0,0,0,0,0)
    JMOVE #HOME

    WHILE 1 DO
        ; S5: 请求抓取和放置规划
        CALL xyz_r_picpla(.error_code, .ws_id)
        IF .error_code <> 0 GOTO LABEL_FAIL
```

(下页继续)

(续上页)

```
        ; S6: 获取抓取入筐轨迹
        CALL xyz_g_picin(.error_code, .ws_id, .pose_type, .wps_num, .wps_type[], .
↪#jnts_traj[])
        IF .error_code <> 0 GOTO LABEL_FAIL
        ; S7: 执行抓取入筐轨迹
        CALL xyz_exectraj(2, .pose_type, .wps_num, .wps_type[], .#jnts_traj[])
        ; S8: 获取抓取出筐轨迹
        CALL xyz_g_picout(.error_code, .ws_id, .pose_type, .wps_num, .wps_type[], .
↪#jnts_traj[])
        IF .error_code <> 0 GOTO LABEL_FAIL
        ; S9: 执行抓取出筐轨迹
        CALL xyz_exectraj(3, .pose_type, .wps_num, .wps_type[], .#jnts_traj[])
        ; S10: 获取放置入筐轨迹
        CALL xyz_g_plain(.error_code, .ws_id, .pose_type, .wps_num, .wps_type[], .
↪#jnts_traj[])
        IF .error_code <> 0 GOTO LABEL_FAIL
        ; S11: 执行放置入筐轨迹
        CALL xyz_exectraj(4, .pose_type, .wps_num, .wps_type[], .#jnts_traj[])
        ; S12: 获取放置出筐轨迹
        CALL xyz_g_plaout(.error_code, .ws_id, .pose_type, .wps_num, .wps_type[], .
↪#jnts_traj[])
        IF .error_code <> 0 GOTO LABEL_FAIL
        ; S13: 执行放置出筐轨迹
        CALL xyz_exectraj(5, .pose_type, .wps_num, .wps_type[], .#jnts_traj[])
    END
LABEL_FAIL:
    CALL xyz_disconn(.ret, sock_id)
.END


.PROGRAM trajmoveasync()
;-----------------------------------------------------------------------
;traj move 异步模板
;-----------------------------------------------------------------------
    ; S1: 初始化参数
    CALL set_const
    CP ON
    MESSAGE ON
    CALL close_all_sock

    ; S2: 连接到上位机
    CALL xyz_conn(sock_id, IP[], PORT)

    ; CALL xyz_sw_flw(.error_code, "test.t")
    ; IF .error_code <> 0 GOTO LABEL_FAIL

    ; S3: 切换工件
    CALL xyz_sw_item(.error_code, 0, "item1")
    IF .error_code <> 0 GOTO LABEL_FAIL

    ; 以下参数需要设置合理的初始值
    SIGNAL -1
    .error_code = 0
    .ws_id = 0
```

(下页继续)

(续上页)

```
    ; S4: 移动到home点，需要提前设置好home点位置
    POINT #HOME = #PPOINT(0,0,0,0,0,0)
    JMOVE #HOME

    ; S5: 请求抓取和放置规划
    CALL xyz_r_picpla(.error_code, .ws_id)
    IF .error_code <> 0 GOTO LABEL_FAIL
    WHILE 1 DO
        ; S6: 获取抓取入筐轨迹
        CALL xyz_g_picin(.error_code, .ws_id, .pose_type, .wps_num, .wps_type[], .
↪#jnts_traj[])
        IF .error_code <> 0 GOTO LABEL_FAIL
        ; S7: 执行抓取入筐轨迹
        CALL xyz_exectraj(2, .pose_type, .wps_num, .wps_type[], .#jnts_traj[])
        ; S8: 获取抓取出筐轨迹
        CALL xyz_g_picout(.error_code, .ws_id, .pose_type, .wps_num, .wps_type[], .
↪#jnts_traj[])
        IF .error_code <> 0 GOTO LABEL_FAIL
        ; S9: 执行抓取出筐轨迹
        CALL xyz_exectraj(3, .pose_type, .wps_num, .wps_type[], .#jnts_traj[])
        ; S10: 请求下一次的抓取和放置规划
        CALL xyz_r_picpla(.error_code, .ws_id)
        IF .error_code <> 0 GOTO LABEL_FAIL
        ; S11: 获取放置入框轨迹
        CALL xyz_g_plain(.error_code, .ws_id, .pose_type, .wps_num, .wps_type[], .
↪#jnts_traj[])
        IF .error_code <> 0 GOTO LABEL_FAIL
        ; S12: 执行放置入筐轨迹
        CALL xyz_exectraj(4, .pose_type, .wps_num, .wps_type[], .#jnts_traj[])
        ; S13: 获取放置出筐轨迹
        CALL xyz_g_plaout(.error_code, .ws_id, .pose_type, .wps_num, .wps_type[], .
↪#jnts_traj[])
        IF .error_code <> 0 GOTO LABEL_FAIL
        ; S14: 执行放置出筐轨迹
        CALL xyz_exectraj(5, .pose_type, .wps_num, .wps_type[], .#jnts_traj[])
    END
LABEL_FAIL:
    CALL xyz_disconn(.ret, sock_id)
.END
```

常见问题

附录

14.2.12 Kuka

此处介绍安装 kuka 机械臂驱动的相关事项。

驱动版本和使用要求

支持的机械臂类型

Kuka 六轴机械臂、五轴机械臂（kuka 拆码垛机械臂）

控制器型号

kr4，同时 KSS 版本需为 8.6 以上

查看 KSS 版本：点击示教器左上角机械臂图标处，点击 帮助 -> 信息。

机械臂需要开通的功能

Kuka 机械臂需要安装 EthernetKRL 功能包

查看功能是否安装：

1. 点击左侧人头标识处，切换用户为管理员，密码为 kuka

2. 进入 投入运行 -> 辅助软件，查看是否有 EthernetKRL

安装驱动

kuka 机械臂驱动文件列表

–src

- xyz_motion.src （工控机主控运行程序）
- xyz_master.src （机械臂主控运行程序）
- xyz_CartMove_template.src （座标移动模板程序）
- xyz_CartMove_reposition.src （座标移动二次定位模板程序）
- xyz_libmotpkt.src （协议处理相关函数）
- xyz_libsock.src （socket 相关函数）
- RobotData.dat （存放的是 src 文件使用的全局变量）
- sps.sub （后台运行程序）

–xml

- MasterClient.xml （机械臂主控 socket 配置文件）
- MotionClient.xml （工控机主控 motion socket 配置文件）
- StatusClient.xml （工控机主控 status socket 配置文件）

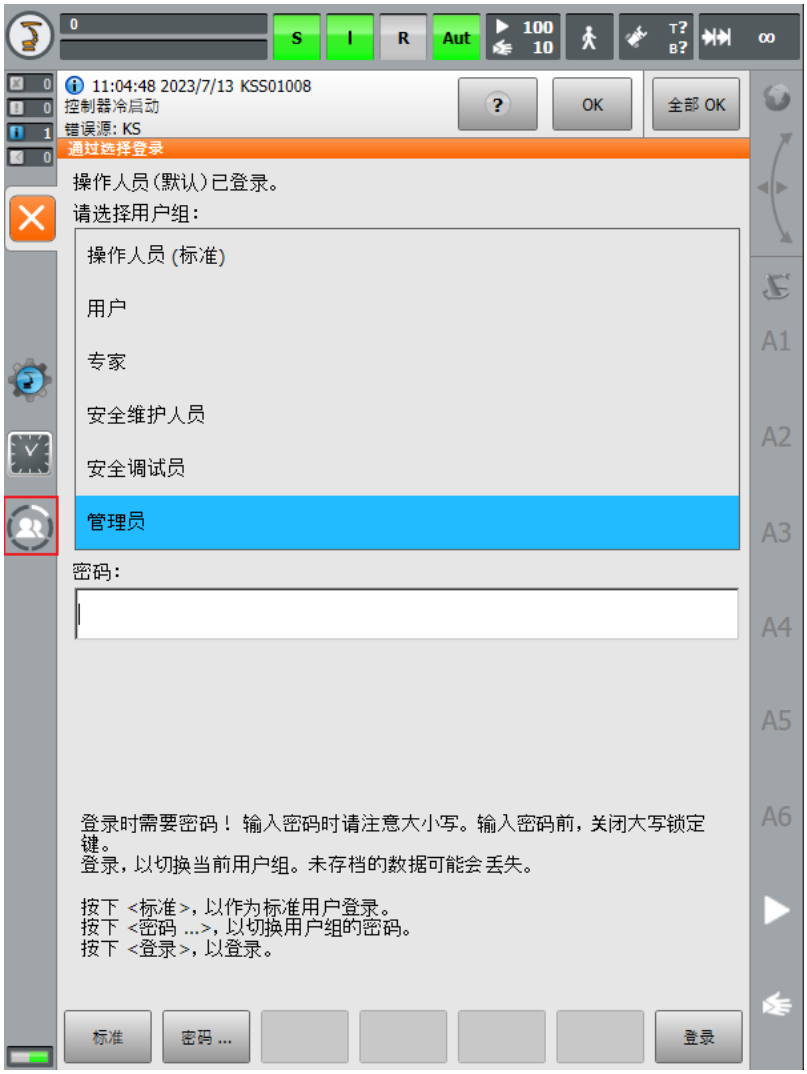


图 75: kuka 切换用户权限

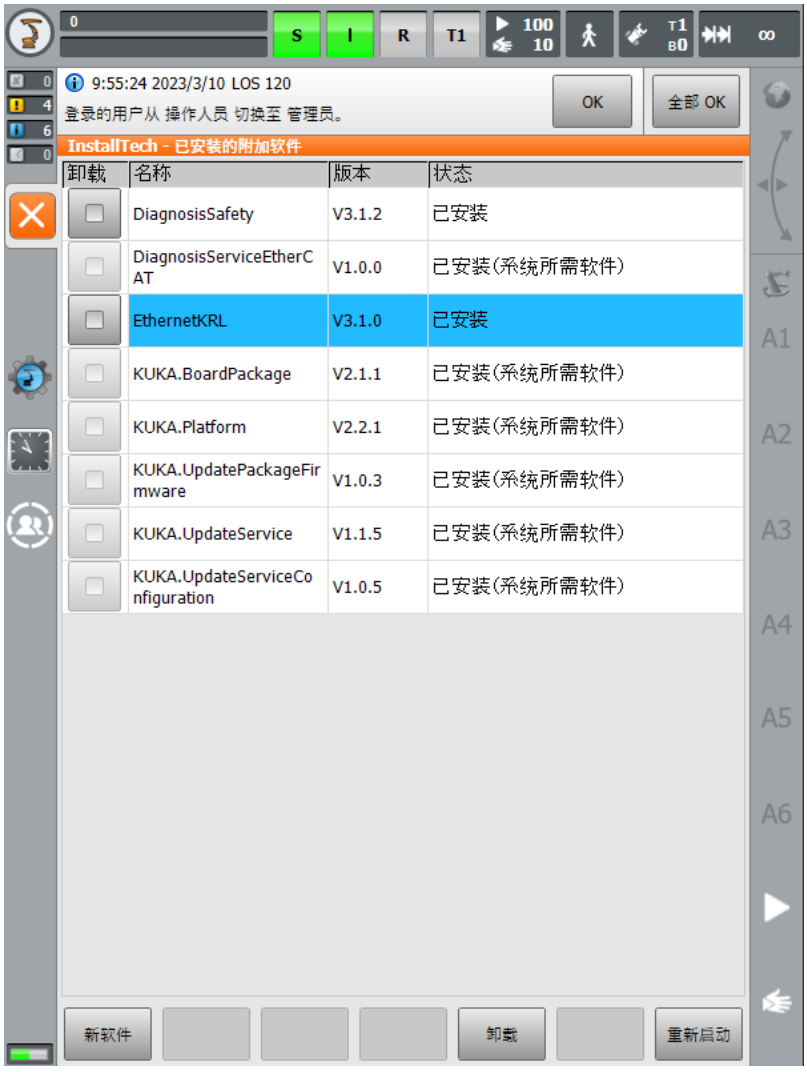


图 76: Kuka 控制器功能查看

设定机械臂 IP

进入用户组选择界面，选择专家登录，密码为 kuka。以下设定均需在 **专家模式**下。

从“主菜单”中选择“投入运行”-> “网络配置”，然后分别选择“IP 地址”，“子网隐码”，“标准网关”进行修改。

- “IP 地址”设置为：192.168.37.100
- “子网隐码”设置为：255.255.255.0
- “标准网关”设置为：0.0.0.0

图 77: kuka 设定机械臂 IP 1

网络设置完成后重启机械臂生效（也可在导入程序完成后进行重启）。

注：若”网络配置”无法按下，或无法删除文件，尝试按”R”或”S” -> 取消选择程序。

拆码垛机器人额外需要进行的操作

在程序导入到示教器之前，修改 xyz_motion 代码，修改 xyzSetJointsMovej() 函数，按照提示，将注释的三行代码反注释, 即：

```
DEFFCT INT xyzSetJointsMovej()

  ; for palleting, uncomment the following sentences
  curJoint = $AXIS_ACT
  Target_Jnt.a4 = curJoint.A4
  Target_Jnt.a5 = curJoint.A5
```

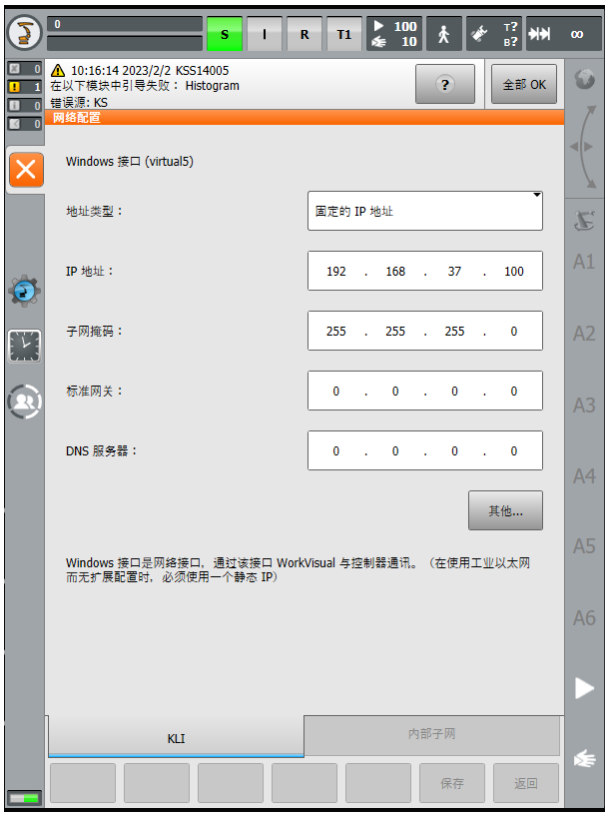


图 78: kuka 设定机械臂 IP 2

使用 U 盘导入程序

- 切换用户为专家或更高权限
- 将 MAX 安装目录下的 share/robot_code 中 kuka 文件夹拷入 U 盘，插入示教器 USB 口或控制柜 U 盘口
- 在 U 盘找到 src 程序，选中程序 `*.src`, `*.dat`。点击示教器底部菜单“编辑”->”复制”

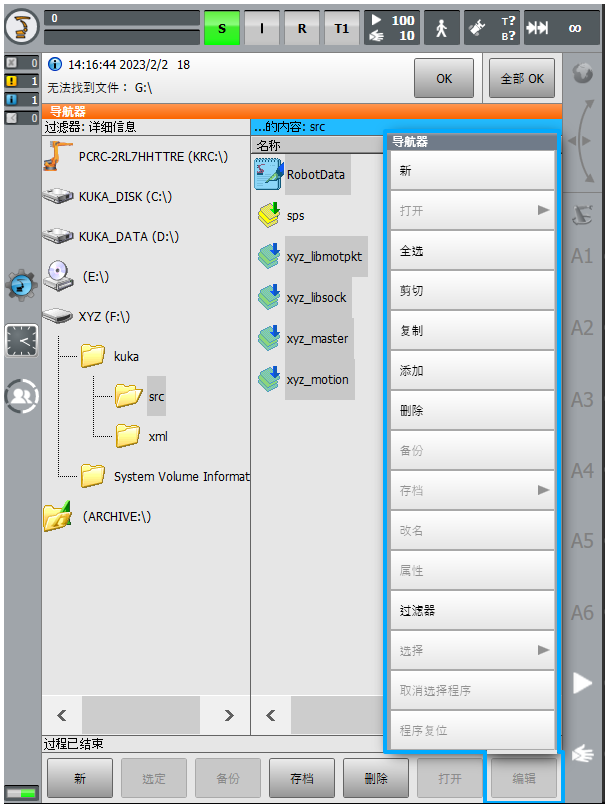


图 79: U 盘导入程序 1

- 在 R1 文件夹下，点击底部菜单“新”,” 新建 xyz_robotics 文件夹
- 在 xyz_robotics 文件夹下，点击底部菜单“编辑” -> “添加”，这时文件开始粘贴
- 按照相同拷贝过程，将 U 盘中的 sps.sub 拷贝到 `R1/System` 路径下
- 将 xml 文件夹下的所有 xml 文件拷贝到 `C:\KRC\ROBOTER\Config\User\Common\EthernetKRL\`

此时，程序已经全部导入。

工控机主控时 IO 相关设定

- 设定数字输出 xyzSetDigitalOutput 不需要进行端口绑定，直接在 MAX 中输入实际的端口号。
- 获取数字输入 DigitalInput 不需要进行端口绑定, 直接在 MAX 中输入实际的端口号。
- 另外，xyzSendRobotStatus() 也会在后台定时不断发送机械臂关节角、位姿、数组输入，要注意进行 di 绑定设置。

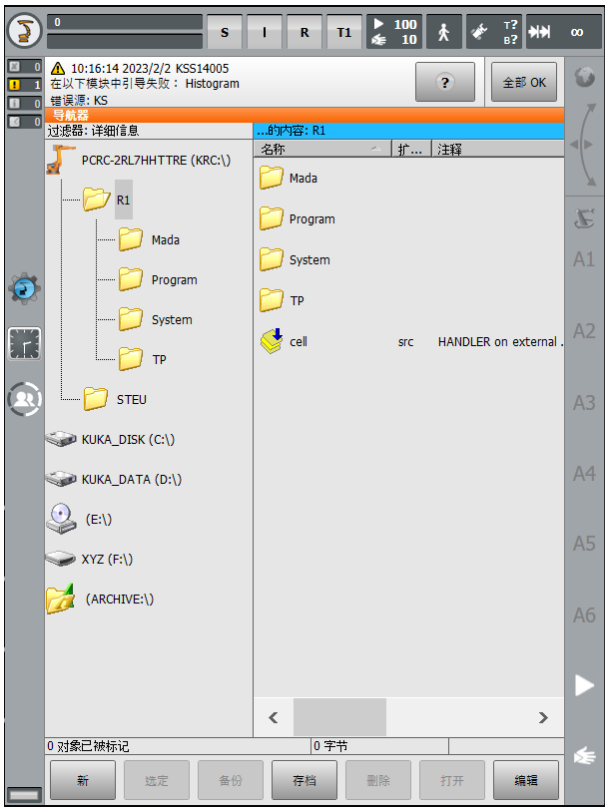

S
I
R
T1
100
10
10:16:14 2023/2/2 KSS14005
在以下模块中引导失败： Histogram
错误源: KS
全部 OK
导航器
过滤器: 详细信息
...的内容: R1
名称
扩...
注释
PCRC-2RL7HHTTRE (KRC:\)
R1
Mada
Program
System
TP
STEU
KUKA_DISK (C:\)
KUKA_DATA (D:\)
(E:\)
XYZ (F:\)
(ARCHIVE:\)
Mada
Program
System
TP
cell
src
HANDLER on external .
A1
A2
A3
A4
A5
A6
0 对象已被标记
0 字节
新
选定
备份
存档
删除
打开
编辑

图 80: U 盘导入程序 2

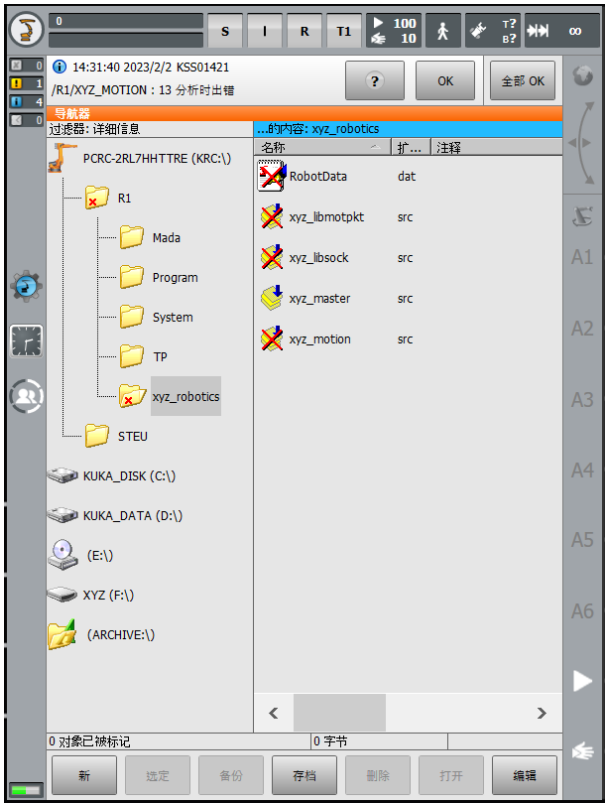


图 81: U 盘导入程序 3

```
GLOBAL DEF xyzSendRobotStatus()
  DECL STATE_T status_state
  DECL EKI_STATUS status_ret
  E6AXIS curJoint
  INT i
  CHAR Str[256]
  BOOL flag
  REAL J1, J2, J3, J4, J5, J6
  CHAR Port_state[16]
  CHAR RECV_STR[128]
  INT status_offset

  status_offset = 0
  IF $FLAG[3] == FALSE THEN
      gRet = EKI_Open("StatusClient")
      WAIT FOR $FLAG[3]
  ENDIF

  IF $FLAG[3] == TRUE THEN
      gRet = EKI_ClearBuffer("StatusClient","Buffer")
      curJoint = $AXIS_ACT
      J1 = curJoint.a1
      J2 = curJoint.a2
      J3 = curJoint.a3
      J4 = curJoint.a4
      J5 = curJoint.a5
```

(下页继续)

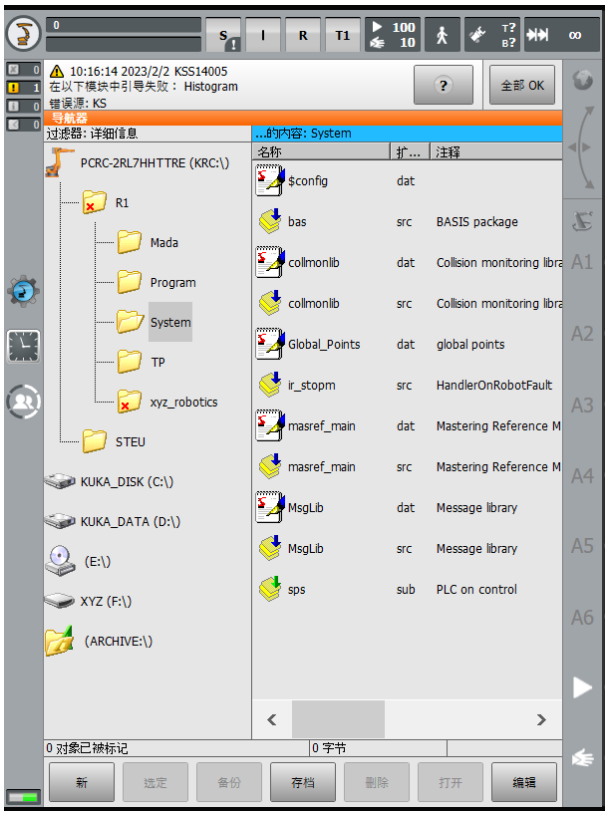

10:16:14 2023/2/2 KSS14005
在以下模块中引导失败： Histogram
错误源: KS
全部 OK
导航器
过滤器: 详细信息
...的内容: System
名称
扩...
注释
PCRC-2RL7HHTTRE (KRC:\)
R1
Mada
Program
System
TP
xyz_robotics
STEU
KUKA_DISK (C:\)
KUKA_DATA (D:\)
(E:\)
XYZ (F:\)
(ARCHIVE:\)
$config dat
bas src BASIS package
collmonlib dat Collision monitoring libra
collmonlib src Collision monitoring libra
Global_Points dat global points
ir_stopm src HandlerOnRobotFault
masref_main dat Mastering Reference M
masref_main src Mastering Reference M
MsgLib dat Message library
MsgLib src Message library
sps sub PLC on control
0 对象已被标记
0 字节
新
选定
备份
存档
删除
打开
编辑

图 82: U 盘导入程序 4

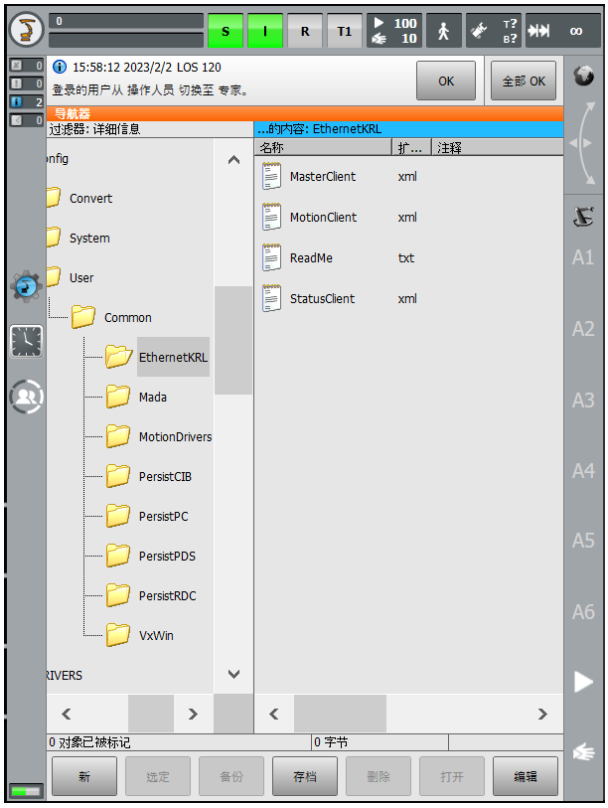


图 83: U 盘导入程序 5

(续上页)

```
      J6 = curJoint.a6
      flag = STRCLEAR(Str[])
      status_offset = 0
      FOR i=1 TO 16 STEP 1
         Port_state[i] = 0
      ENDFOR
      ; Bind DigitalInput with port here !!! 在这里进行 DI 的绑定
      ;For example, $IN[500] 绑定端口 0
      ;IF $IN[500] == True THEN
          ;Port_state[0] = 1 ; port number:0
      ;ELSE
          ;Port_state[0] = 0
      ;ENDIF
      SWRITE(Str[], status_state, status_offset, "200,")
      SWRITE(Str[], status_state, status_offset, "201,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,
↪%d,%d,", J1, J2, J3, J4, J5, J6, 0, 0)
      SWRITE(Str[], status_state, status_offset, "202,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,
↪%d,", $pos_act.x, $pos_act.y, $pos_act.z, $pos_act.A, $pos_act.B, $pos_act.C, 0)
      SWRITE(Str[], status_state, status_offset, "203,%d,%d,%d,%d,%d,%d,%d,%d,", Port_
↪state[1], Port_state[2], Port_state[3], Port_state[4], Port_state[5], Port_state[6],
↪ Port_state[7], Port_state[8])
      SWRITE(Str[], status_state, status_offset, "%d,%d,%d,%d,%d,%d,%d,%d,", Port_
↪state[9], Port_state[10], Port_state[11], Port_state[12], Port_state[13], Port_
↪state[14], Port_state[15], Port_state[16])
      gRet = Eki_Send("StatusClient", Str[])
```

(下页继续)

(续上页)

```
      FOR i = 1 TO 128 STEP 1
          RECV_STR[i] = 0
      ENDFOR
      WAIT FOR $FLAG[4]
      status_ret = EKI_GetString("StatusClient","Buffer", RECV_STR[])
      $FLAG[4] = FALSE
  ENDIF
END
```

运行程序

通讯说明

工控机和 kuka 机械臂的通讯方式为 socket：工控机作为 socket server(服务端)，机械臂作为 socket client（客户端)。

运行后台程序 sps.sub

工控机主控时，后台程序用于定时发送机械臂的各轴角度、末端位姿、I/O 状态。机械臂主控时，后台程序用于发送心跳信号

进入 **管理员**模式，选择顶部菜单栏 "**S** "，点击 选择/启动，应该可以看到行数在一直滚动。

如果机械臂发生意外停止，可以看到行数会停在某一数字，此时需要 取消选择，再点击 选择/启动。

图 84: kuka 运行后台 sub 程序

机械臂上电

- 机械臂顶部菜单栏第二个应该为 |，表示已上电。
- 若为 O，则需要点击 O, 点击驱动装置 |。

启动程序

工控机主控运行

- 转动示教器档位切换，点击 AUT 后，旋回。
- 选中 xyz_robotics 文件夹下的 xyz_motion, 点击底部菜单栏的 选定。
- 工控机启动 robot_driver_node 后，按动示教器左侧实体按键：|> 运行按钮，需按动两次。

机械臂主控运行

- 转动示教器档位切换，点击 AUT 后，旋回。
- 选中 xyz_robotics 文件夹下的 xyz_master, 点击底部菜单栏的 选定。
- 工控机启动 robot_server 后，按动示教器左侧实体按键：|> 运行按钮。

如果机械臂运行中发生意外停止，切回 T1 模式，再点击 R -> 取消选择程序或 程序复位。

API 说明

kuka 工控机主控支持的 API

指令号	命令	支持情况
100	获取机械臂程序版本号	支持
101	发送速度数据	支持
102	发送加速度数据	支持
103	发送圆滑过渡参数	支持
104	发送工具坐标系 (TCP) 参数	支持
105	设置数字量输出	支持
106	SetJointsMovej	支持
107	SetCartMovel	支持
108	SetJoinsMovel	支持
109	SetCartMovej	支持
110	MovejSequence	支持
111	MovelSequence	支持
112	SetJointsMovejDo	不支持
113	SetCartMovelDo	不支持
114	SetJointsMovelDo	不支持
115	SetJointsMovejGroupDo	不支持
116	SetCartMovelGroupDo	不支持
117	SetJointsMovelGroupDo	不支持
118	MovelUntil	支持
119	获取数字量输入状态	支持
120	获取数字量输出状态	支持
121	获取模拟量输入口数值	不支持
122	获取机械臂当前角度	支持
123	获取机械臂当前位姿	支持
200	机械臂后台发送状态	支持

kuka 机械臂主控支持的 API

- kuka 支持传入变量传出

GLOBAL DEF xyzOpenSocket(sock[]:IN)

根据 xml 连接指定 socket

参数 sock[] (*CHAR[]*) –socket 名称

返回 err_code

返回类型 INT

GLOBAL DEF xyzHeartBeat()

定期发送心跳信号，不支持

DEFFCT INT xyzSwitchApp(app_name[]:IN)

切换应用

参数 app_name[] (*CHAR[]*) –应用名称

返回 err_code

返回类型 INT

DEFFCT INT xyzSwitchFlow(flow_name[]:IN)

切换流图

参数 flow_name[] (*CHAR[]*) –流图名称

返回 err_code

返回类型 INT

DEFFCT INT xyzSwitchTool(tool[]:IN)

切换工具

参数 tool[] (*CHAR[]*) –工具名称

返回 err_code

返回类型 INT

DEFFCT INT xyzReqCapImg(vision_service_id:IN, token:OUT)

请求拍照

参数

- **vision_service_id** (*INT*) –需要进行拍照的视觉服务 id
- **token** (*INT*) –请求拍照结果

返回 err_code

返回类型 INT

DEFFCT INT xyzGetCapImg(token:IN)

获取拍照结果

参数 token (*INT*) –请求拍照时返回的 token

返回 err_code

返回类型 INT

DEFFCT INT xyzCapImg(vision_service_id:IN)

拍照

参数 **vision_service_id** (*INT*) –需要进行拍照的视觉服务 id

返回 err_code

返回类型 INT

DEFFCT INT xyzReqGraspPose(ws_id:IN, token:OUT)

请求抓取位姿

参数

- **ws_id** (*INT*) –需要获取抓取点位的工作空间 id
- **token** (*INT*) –返回的用于获取目标点位时使用的 token

返回 err_code

返回类型 INT

DEFFCT INT xyzGetGraspPose(token:IN, grasp_pose:OUT, grasp_pose_num:OUT, pipeline_num:OUT, register_num:OUT)

获取抓取位姿

参数

- **token** (*INT*) –求抓取目标点位时返回的 token
- **grasp_pose** (*E6POS*) –抓取位姿
- **grasp_pose_num** –可供抓取的点数量
- **grasp_pose_num** –INT
- **pipeline_num** –pipeline 编号
- **pipeline_id** –INT
- **register_num** –注册编号

返回 err_code

返回类型 INT

DEFFCT INT xyzReqObjPose(ws_id:IN, obj_token:OUT)

请求物体位姿

参数

- **ws_id** (*INT*) –需要获取物体位姿的工作空间 id
- **obj_token** (*INT*) –物体位姿识别的 token

返回 err_code

返回类型 INT

DEFFCT INT xyzGetObjPose(token:IN, obj_pose:OUT, obj_pose_num:OUT, obj_pose_type:OUT)

获取物体位姿

参数

- **token** (*INT*) –请求物体位姿时得到的 token
- **obj_pose** (*E6POS*) –物体位姿

- **obj_pose_num** (*INT*) –物体数量
- **obj_pose_type** (*INT*) –当前返回的物体 pose 类型

返回 err_code

返回类型 INT

DEFFCT INT Task()

重置任务

返回 err_code

返回类型 INT

DEFFCT INT xyzSendCurrentJoints()

发送机械臂当前角度：j1~j6: 机械臂当前的角度信息，如果机械臂轴数小于 6 轴，则需要对应位置补零，依旧发送六个数。

返回 err_code

返回类型 INT

DEFFCT INT xyzSendCurrentCartPose()

发送机械臂法兰当前位姿

返回 err_code

返回类型 INT

DEFFCT INT xyzSendCurrentCartPose()

发送机械臂当前扩展轴位置：j1~j6: 机械臂当前扩展轴的角度信息，如果扩展轴数不足 6 的, 需要补零后发送六个数。

返回 err_code

返回类型 INT

DEFFCT INT xyzReqPick()

请求 pick 动作规划

返回 err_code

返回类型 INT

DEFFCT INT xyzReqPlace()

请求 place 动作规划

返回 err_code

返回类型 INT

DEFFCT INT xyzReqPickPlace()

请求 pick 和 place 规划

返回 err_code

返回类型 INT

GLOBAL DEFFCT INT xyzGetPickin(ws_id:IN, pipeline_num:OUT, register_num:OUT, num:OUT, wp_type[]:OUT, joints[]:OUT, carts[]:OUT)

参数

- **uws_id** (*INT*) –需要获取取料入框点位的工作空间 id

- **pipeline_num** (*INT*) –pipeline 编号
- **register_num** (*INT*) –注册编号
- **num** (*INT*) –可供抓取的点数量
- **wp_type[]** (*INT*) –轨迹点位类型数组
- **joints[]** (*E6AXIS*) –关节点位数组
- **carts[]** (*E6POS*) –笛卡尔点位数组

获取取料入框轨迹

返回 err_code

返回类型 INT

GLOBAL DEFFCT INT xyzGetPickout(ws_id:IN, pipeline_num:OUT, register_num:OUT, num:OUT, wp_type[]:OUT, joints[]:OUT, carts[]:OUT)

参数

- **uws_id** (*INT*) –需要获取取料出框点位的工作空间 id
- **pipeline_num** (*INT*) –pipeline 编号
- **register_num** (*INT*) –注册编号
- **num** (*INT*) –可供抓取的点数量
- **wp_type[]** (*INT*) –轨迹点位类型数组
- **joints[]** (*E6AXIS*) –关节点位数组
- **carts[]** (*E6POS*) –笛卡尔点位数组

获取取料出框轨迹

返回 err_code

返回类型 INT

GLOBAL DEFFCT INT xyzGetPlacein(ws_id:IN, pipeline_num:OUT, register_num:OUT, num:OUT, wp_type[]:OUT, joints[]:OUT, carts[]:OUT)

参数

- **uws_id** (*INT*) –需要获取放料入框点位的工作空间 id
- **pipeline_num** (*INT*) –pipeline 编号
- **register_num** (*INT*) –注册编号
- **num** (*INT*) –可供抓取的点数量
- **wp_type[]** (*INT*) –轨迹点位类型数组
- **joints[]** (*E6AXIS*) –关节点位数组
- **carts[]** (*E6POS*) –笛卡尔点位数组

获取放料入框轨迹

:param

返回 err_code

返回类型 INT

GLOBAL DEFFCT INT xyzGetPlaceout(ws_id:IN, pipeline_num:OUT, register_num:OUT, num:OUT, wp_type[]:OUT, joints[]:OUT, carts[]:OUT)

参数

- **uws_id** (*INT*) –需要获取放料出框点位的工作空间 id
- **pipeline_num** (*INT*) –pipeline 编号
- **register_num** (*INT*) –注册编号
- **num** (*INT*) –可供抓取的点数量
- **wp_type[]** (*INT*) –轨迹点位类型数组
- **joints[]** (*E6AXIS*) –关节点位数组
- **carts[]** (*E6POS*) –笛卡尔点位数组

获取放料出框轨迹

返回 err_code

返回类型 INT

GLOBAL DEF xyzParseTraj(pipeline_num:OUT, register_num:OUT, num:OUT, wp_type[]:OUT, joints[]:OUT, carts[]:OUT)

解析轨迹

返回

err_code : 错误码

pipeline_num : pipeline 编号

register_num : 注册编号

num : 轨迹点数

wp_type[] : 轨迹点类型

joints[] : 关节坐标数组

carts[] : 笛卡尔坐标数组

GLOBAL DEF xyzExecuteTraj(num:IN, wp_type[]:OUT, joints[]:OUT, carts[]:OUT)

执行轨迹

返回

err_code : 错误码

num : 轨迹点数

wp_type[] : 轨迹点类型

joints[] : 关节坐标数组

carts[] : 笛卡尔坐标数组

GLOBAL DEF xyzExecutePickInTraj(num:IN, wp_type[]:OUT, joints[]:OUT, carts[]:OUT)

执行取料入框轨迹

返回

err_code : 错误码

num : 轨迹点数

wp_type[] : 轨迹点类型

joints[] : 关节坐标数组

carts[] : 笛卡尔坐标数组

GLOBAL DEF xyzExecutePickOutTraj(num:IN, wp_type[]:OUT, joints[]:OUT, carts[]:OUT)

执行取料出框轨迹

返回

err_code : 错误码

num : 轨迹点数

wp_type[] : 轨迹点类型

joints[] : 关节坐标数组

carts[] : 笛卡尔坐标数组

GLOBAL DEF xyzSetMoveJParameters(movej_vel:IN, movej_acc:IN, movej_apo_dist:IN)

设置关节移动参数

参数

- **movej_vel** –关节移动速度, 范围 0~100
- **movej_acc** –关节移动加速度, 范围 0~100
- **movej_apo_dist** –圆滑过渡半径, 范围 0~500 mm

GLOBAL DEF xyzSetMoveLParameters(movel_vel:IN, movel_acc:IN, movel_apo_dist:IN)

设置笛卡尔移动参数

参数

- **movel_vel** –笛卡尔移动速度, 范围 0~100
- **movel_acc** –笛卡尔移动加速度, 范围 0~100
- **movel_apo_dist** –圆滑过渡半径, 范围 0~500 mm

GLOBAL DEFFCT INT xyzRecvAndParse()

接收并解析数据

返回 err_code

返回类型 INT

DEFFCT INT xyzSwitchStrat(in_strat[]:IN)

请求切换策略

参数 in_strat[] (*CHAR[]*) –策略名称

返回 err_code

返回类型 INT

DEFFCT INT xyzUpdateTotePose()

料箱重定位（需手动扩展）

返回 err_code

返回类型 INT

DEFFCT INT xyzUpdateObjPoseOnHand()

工件在上手的二次定位

返回 err_code

返回类型 INT

GLOBAL DEFFCT INT xyzUpdateObjPoseToHand(pipeline_num:OUT, register_num:OUT, num:OUT, wp_type[]:OUT, joints[]:OUT, carts[]:OUT)

参数

- **pipeline_num** (*INT*) –pipeline 编号
- **register_num** (*INT*) –注册编号
- **num** (*INT*) –可供抓取的点数量
- **wp_type[]** (*INT*) –轨迹点位类型数组
- **joints[]** (*E6AXIS*) –关节点位数组
- **carts[]** (*E6POS*) –笛卡尔点位数组

工件不在手上的二次定位（需手动扩展）

返回 err_code

返回类型 INT

DEFFCT INT xyzSwitchObj(obj_name[]:IN)

切换工件

参数 **obj_name[]** (*CHAR[]*) –流图名称

返回 err_code

返回类型 INT

GLOBAL DEFFCT INT xyzCalculateGraspPose(ws_id:IN, grasp_pose:OUT, grasp_pose_num:OUT, pipeline_num:OUT, register_num:OUT)

计算抓取目标点位

参数

- **ws_id** (*INT*) –需要进行抓取的工作空间 id
- **grasp_pose** (*E6POS*) –抓取目标点位
- **grasp_pose_num** (*INT*) –可供抓取的点数量
- **pipeline_num** –pipeline 编号
- **register_num** –注册编号

GLOBAL DEFFCT INT xyzCalculateObjPose(ws_id:IN, obj_pose:OUT, obj_pose_num:OUT, pose_type:OUT)

计算物体位姿

参数

- **ws_id** (*INT*) –需要进行抓取的工作空间 id
- **obj_pose** (*E6POS*) –物体位姿
- **obj_pose_num** (*INT*) –可供抓取的点数量
- **pose_type** (*INT*) –物体位姿类型

案例/模板说明

机械臂主控程序说明

以下为机械臂主控模板代码，注意对工控机返回的 err_code 进行判断。

座标移动基础模板

```
DEF xyz_CartMove_template()
    ; S1: Initialization
    ; S1: 初始化
    INT err_code ; 错误码
    INT grasp_pose_token ; 抓取位姿 token
    E6POS grasp_pose ; 抓取位姿
    E6POS home_pose ; home 位姿
    E6POS scan_pose ; 拍照位姿
    E6POS place_pose ; 放置位姿
    INT grasp_pose_num ; 抓取位姿数量
    BOOL is_eye_in_hand ; 是否为眼在手上
    INT obj_pipeline_num ; 物体 pipeline 编号
    INT obj_register_num ; 物体注册编号

    grasp_pose_token = 0
    gMasterFlag = TRUE
    err_code = 0
    grasp_pose_num = 0
    obj_pipeline_num = 0
    obj_register_num = 0

    ; 以下为中断处理等代码，不能删除
    GLOBAL INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO IR_STOPM ( )
    INTERRUPT ON 3
    BAS (#INITMOV,0 )
    ; The first movement for Kuka must be a PTP movement
    $BWDSTART = FALSE
    PDAT_ACT = PPDAT_1
    FDAT_ACT = FPDAT_2
    BAS(#PTP_PARAMS, PPDAT_1.vel)
    ; Comment out this line if there is an error
    SET_CD_PARAMS (0)
    PTP $AXIS_ACT

    ; S2: Connect to IPC
    ; S2: 连接 IPC
    xyzOpenSocket("MasterClient") ; 连接 IPC

    ; change the flow name here!
```

(下页继续)

(续上页)

```
; err_code = xyzSwitchFlow("cart_move.t") ; 切换任务
; IF (err_code <> 0) THEN  ;
;     GOTO END_PROGRAM;
; ENDIF

; S3: Switch object
; S3: 切换物体
; 0: ws_id, "item1": item_codename
; change the item_codename here!
err_code = xyzSwitchItem(0,"item1") ; 切换物体
IF (err_code <> 0) THEN  ;
    HALT;
    GOTO END_PROGRAM;
ENDIF

; S4: Move to home pose
; S4: 移动到 home 位姿
; you should use WorkVisual to set the home_pose,
; or use pendant to set the home_pose
; 使用 WorkVisual 设置 home 位姿，或者使用示教器设置 home 位姿
; home_pose = {X 578.80, Y -9.12, Z 541.47, A 173.34,B -4.38,C -177.88}
; xyzSetMoveJParameters(10,20,0) ; 设置关节移动参数：速度、加速度、减速度
; PTP home_pose ; 关节移动到 home 位姿
; xyzSetMoveLParameters(10,20,0) ; 设置直线移动参数：速度、加速度、减速度
; LIN home_pose ; 直线移动到 home 位姿
; $OUT[500] = TRUE; IO 操作

; if application is eye_on_hand, please make eye_on_hand := true;
; 如果是眼在手上的应用，请将 eye_on_hand := true;
is_eye_in_hand = false;

BEGIN_WHILE:
WHILE TRUE
    ; S5: eye on hand application
    ; S5: 眼在手上的应用
    IF is_eye_in_hand THEN
        ; S6: Move to scan pose
        ; S6: 移动到拍照位姿
        ; you should use WorkVisual to set the scan_pose,
        ; or use pendant to set the scan_pose
        ; 使用 WorkVisual 设置拍照位姿，或者使用示教器设置拍照位姿
        ; set movel or moveJ parameters(vel, acc, zone)
        ; scan_pose = {X 578.80, Y -9.12, Z 541.47, A 173.34,B -4.38,C -177.88}
        ; xyzSetMoveJParameters(10,20,0) ; 设置关节移动参数：速度、加速度、减速度
        ; PTP scan_pose ; 关节移动到拍照位姿
        ; xyzSetMoveLParameters(10,20,0) ; 设置直线移动参数：速度、加速度、减速度
        ; LIN scan_pose ; 直线移动到拍照位姿

        ; S7: Send scan pose
        ; S7: 发送拍照位姿
        err_code = xyzSendCurrentCartPose()
        IF (err_code <> 0) THEN  ;
            HALT;
            GOTO END_PROGRAM;
        ENDIF
    ENDIF
```

(下页继续)

(续上页)

```
        ; S8: Request grasp pose(including cap image)
        ; S8: 请求抓取位姿
        err_code = xyzReqGraspPose(0,grasp_pose_token)
        IF (err_code <> 0) THEN  ;
            HALT;
            GOTO END_PROGRAM;
        ENDIF

        ; S9: Get grasp pose
        ; S9: 获取抓取位姿
        err_code = xyzGetGraspPose(grasp_pose_token, grasp_pose, grasp_pose_num, obj_
↪pipeline_num, obj_register_num)
        IF (err_code <> 0) THEN  ;
            HALT;
            GOTO END_PROGRAM;
        ENDIF

        ; 如果没有可以抓取的物体
        IF (grasp_pose_num < 1) THEN
            WAIT SEC 5
            GOTO BEGIN_WHILE;
        ENDIF

        ; S10: Move to grasp pose and grasp object
        ; S10: 移动到抓取位姿并抓取物体
        ; Attention: Additional waypoints maybe need added
        ; 注意：可能需要添加额外的路径点
        ; xyzSetMoveJParameters(10,20,0) ; 设置关节移动参数：速度、加速度、减速度
        ; PTP grasp_pose ; 关节移动到抓取位姿
        ; xyzSetMoveLParameters(10,20,0) ; 设置直线移动参数：速度、加速度、减速度
        ; LIN grasp_pose ; 直线移动到抓取位姿
        ; $OUT[500] = TRUE; grasp object 抓取物体
        ; WAIT SEC 0.5

        ; S11: Move to place pose and place object
        ; S11: 移动到放置位姿并放置物体
        ; you should use WorkVisual to set the place_pose,
        ; or use pendant to set the place_pose
        ; 使用 WorkVisual 设置放置位姿，或者使用示教器设置放置位姿
        ; place_pose = {X 578.80, Y -9.12, Z 541.47, A 173.34,B -4.38,C -177.88}
        ; xyzSetMoveJParameters(10,20,0) ; 设置关节移动参数：速度、加速度、减速度
        ; PTP place_pose ; 关节移动到放置位姿
        ; xyzSetMoveLParameters(10,20,0) ; 设置直线移动参数：速度、加速度、减速度
        ; LIN place_pose ; 直线移动到放置位姿

        ; Attention: Additional waypoints should be added
        ; 注意：需要添加额外的路径点

        ; after move to place pose
        ; 在移动到放置位姿之后
        ; $OUT[500] = FALSE; place object 放置物体

    ENDWHILE

    END_PROGRAM:
END
```

座标移动二次定位模板

```
DEF xyz_CartMove_reposition()
    ; S1: Initialization
    ; S1: 初始化
    INT err_code ; 错误码
    INT  rough_grasp_pose_token ; 粗定位抓取位姿 token
    E6POS  rough_grasp_pose ; 粗定位抓取位姿
    INT  rough_grasp_pose_num ; 粗定位抓取位姿数量
    INT  board_grasp_pose_token ; 精定位抓取位姿 token
    E6POS  board_grasp_pose ; 精定位抓取位姿
    INT  board_grasp_pose_num ; 精定位抓取位姿数量
    INT  fine_grasp_pose_token ; 精定位抓取位姿 token
    E6POS  fine_grasp_pose ; 精定位抓取位姿
    INT  fine_grasp_pose_num ; 精定位抓取位姿数量
    E6POS home_pose ; home 位姿
    INT obj_pipeline_num ; 物体 pipeline 编号
    INT obj_register_num ; 物体注册编号
    INT board_pipeline_num ; board pipeline 编号
    INT board_register_num ; board 注册编号


    gMasterFlag = TRUE
    err_code = 0
    obj_pipeline_num = 0
    obj_register_num = 0
    board_pipeline_num = 0
    board_register_num = 0
    fine_grasp_pose_token = 0

    ; 以下为中断处理等代码，不能删除
    GLOBAL INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO IR_STOPM ( )
    INTERRUPT ON 3
    BAS (#INITMOV,0 )
    ; The first movement for Kuka must be a PTP movement
    $BWDSTART = FALSE
    PDAT_ACT = PPDAT_1
    FDAT_ACT = FPDAT_2
    BAS(#PTP_PARAMS, PPDAT_1.vel)
    ; Comment out this line if there is an error
    SET_CD_PARAMS (0)
    PTP $AXIS_ACT

    ; $OUT[500] = TRUE; io initialization IO 初始化

    ; S2: Connect to IPC
    ; S2: 连接 IPC
    xyzOpenSocket("MasterClient") ; 连接 IPC

    ; change the flow name here! The name should be less than 13 characters.
    ; err_code = xyzSwitchFlow("cart_repo.t") ; 切换任务
    ; IF (err_code <> 0) THEN  ;
    ;    HALT;
    ;    GOTO END_PROGRAM;
    ; ENDIF

    ; S3: The camera outside the robotic arm is switched to recognize the current␣
↪workpiece
```

(下页继续)

(续上页)

```
    ; S3: 切换机械臂外部的相机，识别当前工件
    ; camera 1 rough location, 0: ws_id, "item1": item_codename
    ; change item_codename here!
    err_code = xyzSwitchItem(0,"item1") ; 切换工件
    IF (err_code <> 0) THEN  ;
        HALT;
        GOTO END_PROGRAM;
    ENDIF

    ; S4: Move to home pose
    ; S4: 移动到 home 位姿
    ; Additional: should define home_pose first,
    ; you can modify home_pose with WorkVisual,
    ; or use pendant to set the home_pose
    ; 需要先定义 home 位姿，
    ; 可以使用 WorkVisual 修改 home 位姿，或者使用示教器设置 home 位姿
    ; home_pose = {X 578.80, Y -9.12, Z 541.47, A 173.34,B -4.38,C -177.88}
    ; xyzSetMoveJParameters(10,20,0) ; vel, acc, zone␣
↪设置关节移动参数：速度、加速度、减速度
    ; PTP home_pose ; 关节移动到 home 位姿
    ; xyzSetMoveLParameters(10,20,0) ; vel, acc, zone␣
↪设置直线移动参数：速度、加速度、减速度
    ; LIN home_pose ; 直线移动到 home 位姿

    ; S5: Request grasp pose
    ; S5: 请求抓取位姿
    err_code = xyzReqGraspPose(0,rough_grasp_pose_token) ; 请求抓取位姿
    IF (err_code <> 0) THEN  ;
        HALT;
        GOTO END_PROGRAM;
    ENDIF

    BEGIN_WHILE:
    WHILE TRUE
        ; S6: get grasp pose
        ; S6: 获取抓取位姿
        err_code = xyzGetGraspPose(rough_grasp_pose_token, rough_grasp_pose,  rough_
↪grasp_pose_num,  obj_pipeline_num, obj_register_num)
        IF (err_code <> 0) THEN  ;
            HALT;
            GOTO END_PROGRAM;
        ENDIF

        ; S7: grasp board if no object exists on board
        ; S7: 如果board上没有物体，就抓取board
        IF (rough_grasp_pose_num < 1) THEN
            ; S15: The camera outside the robotic arm is switched to recognize the␣
↪board
            ; S15: 切换机械臂外部的相机，识别board
            ; no object in workspace, should remove board first
            ; 没有物体在工作区，应该先移除board
            ; switch board; O:ws_id, board: board_name
            err_code = xyzSwitchItem(0,"board") ; 切换工件为 board
            IF (err_code <> 0) THEN  ;
                HALT;
                GOTO END_PROGRAM;
```

(下页继续)

(续上页)

```
            ENDIF

            ; S16: Request board grasp pose
            ; S16: 请求 board 抓取位姿
            err_code = xyzReqGraspPose(0,board_grasp_pose_token)
            IF (err_code <> 0) THEN  ;
                HALT;
                GOTO END_PROGRAM;
            ENDIF

            ; S17: Get board grasp pose
            ; S17: 获取 board 抓取位姿
            err_code = xyzGetGraspPose(board_grasp_pose_token, board_grasp_pose, ␣
↪board_grasp_pose_num,  board_pipeline_num, board_register_num)
            IF (err_code <> 0) THEN  ;
                HALT;
                GOTO END_PROGRAM;
            ENDIF

            ; 如果没有可以抓取的 board
            IF (board_grasp_pose_num < 1) THEN
                ; no board exists, you can change tote
                HALT;
                GOTO END_PROGRAM;
            ENDIF

            ; S18: Move to board grasp pose and grasp board
            ; S18: 移动到 board 抓取位姿并抓取 board
            ; xyzSetMoveJParameters(10,20,0) ; vel, acc, zone␣
↪设置关节移动参数：速度、加速度、减速度
            ; PTP grasp_pose ; 关节移动到抓取位姿
            ; xyzSetMoveLParameters(10,20,0) ; vel, acc, zone␣
↪设置直线移动参数：速度、加速度、减速度
            ; LIN grasp_pose ; 直线移动到抓取位姿

            ; $OUT[500] = TRUE; grasp board ; 抓取 board
            WAIT SEC 0.5

            ; S19: Move to board place pose
            ; S19: 移动到 board 放置位姿
            ; you should use WorkVisual to set the place_pose,
            ; or use pendant to set the place_pose
            ; 需要先定义 place 位姿，使用 WorkVisual 或者示教器设置
            ; place_pose = {X 578.80, Y -9.12, Z 541.47, A 173.34,B -4.38,C -177.88}
            ; xyzSetMoveJParameters(10,20,0) ; vel, acc, zone␣
↪设置关节移动参数：速度、加速度、减速度
            ; PTP place_pose ; 关节移动到放置位姿
            ; xyzSetMoveLParameters(10,20,0) ; vel, acc, zone␣
↪设置直线移动参数：速度、加速度、减速度
            ; LIN place_pose ; 直线移动到放置位姿

            ; $OUT[500] = FALSE; place board ; 放置 board

            ; S20: The camera outside the robotic arm is switched to recognize the␣
↪current workpiece
            ; S20: 切换机械臂外部的相机，识别当前工件
```

(下页继续)

(续上页)

```
            ; 0: ws_id, "item1": item_codename
            ; change the item_codename here!
            err_code = xyzSwitchItem(0,"item1") ; 切换工件为 item1
            IF (err_code <> 0) THEN  ;
                HALT;
                GOTO END_PROGRAM;
            ENDIF

            GOTO BEGIN_WHILE;
        ELSE

            ; S8: Move to scan pose
            ; S8: 移动到拍照位姿
            ; Additional: should define scan_pose first,
            ; youcan modify scan_pose with WorkVisual or
            ; use pendant to set scan_pose
            ; 需要先定义 scan 位姿，使用 WorkVisual 或者示教器设置
            ; scan_pose = {X 578.80, Y -9.12, Z 541.47, A 173.34,B -4.38,C -177.88}
            ; xyzSetMoveJParameters(10,20,0) ; 设置关节移动参数：速度、加速度、减速度
            ; PTP scan_pose ; 关节移动到拍照位姿
            ; xyzSetMoveLParameters(10,20,0) ; 设置直线移动参数：速度、加速度、减速度
            ; LIN scan_pose ; 直线移动到拍照位姿

            ; S9: Send scan pose
            ; S9: 发送拍照位姿
            err_code = xyzSendCurrentCartPose()
            IF (err_code <> 0) THEN  ;
                HALT;
                GOTO END_PROGRAM;
            ENDIF

            ; S10: The camera on the robotic arm is switched to recognize the current␣
↪workpiece
            ; S10: 切换机械臂上的相机，识别当前工件
            ; camera2 cap precise image, 1: ws_id, "item1": item_codename
            ; change the item_codename here!
            err_code = xyzSwitchItem(1,"item1") ; 切换工件为 item1
            IF (err_code <> 0) THEN  ;
                GOTO END_PROGRAM;
            ENDIF

            ; S11: Request grasp pose
            ; S11: 请求抓取位姿
            err_code = xyzReqGraspPose(1,fine_grasp_pose_token)
            IF (err_code <> 0) THEN  ;
                HALT;
                GOTO END_PROGRAM;
            ENDIF

            ; S12: Get grasp pose
            ; S12: 获取抓取位姿
            err_code = xyzGetGraspPose(fine_grasp_pose_token ,fine_grasp_pose, fine_
↪grasp_pose_num, obj_pipeline_num, obj_register_num)
            IF (err_code <> 0) THEN  ;
                HALT;
                GOTO END_PROGRAM;
```

(下页继续)

(续上页)

```
            ENDIF

            ; 如果没有可以抓取的工件
            IF (fine_grasp_pose_num < 1) THEN
                HALT;
                GOTO END_PROGRAM;
            ENDIF

            ; S13: Move to grasp pose and grasp workpiece
            ; S13: 移动到抓取位姿并抓取工件
            ; xyzSetMoveJParameters(10,20,0) ; vel, acc, zone␣
↪设置关节移动参数：速度、加速度、减速度
            ; PTP fine_grasp_pose ; 关节移动到抓取位姿
            ; xyzSetMoveLParameters(10,20,0) ; vel, acc, zone␣
↪设置直线移动参数：速度、加速度、减速度
            ; LIN fine_grasp_pose ; 直线移动到抓取位姿

            ; $OUT[500] = TRUE; grasp workpiece ; 抓取工件
            WAIT SEC 0.5

            ; S14: Move to place pose and place workpiece
            ; S14: 移动到放置位姿并放置工件
            ; you should use WorkVisual to define place_pose,
            ; or use pendant to set place_pose
            ; 需要先定义 place 位姿，使用 WorkVisual 或者示教器设置
            ; place_pose = {X 578.80, Y -9.12, Z 541.47, A 173.34,B -4.38,C -177.88}
            ; xyzSetMoveJParameters(10,20,0) ; vel, acc, zone␣
↪设置关节移动参数：速度、加速度、减速度
            ; PTP place_pose ; 关节移动到放置位姿
            ; xyzSetMoveLParameters(10,20,0) ; vel, acc, zone␣
↪设置直线移动参数：速度、加速度、减速度
            ; LIN place_pose ; 直线移动到放置位姿
            ; $OUT[500] = FALSE; place workpiece ; 放置工件
        ENDIF
    ENDWHILE

    END_PROGRAM:
END
```

3D 轨迹移动同步程序

```
DEF xyz_TrajMove_Sync()

   ; S1: Initialization
   ; S1: 初始化
   INT err_code
   CHAR flow_name[20]
   INT ws_id
   CHAR item_codename[20]
   E6POS home_pose
   INT pipeline_num
   INT register_num
   INT way_point_num
   INT wp_type[30]
```

(下页继续)

(续上页)

```
E6AXIS joints[50]
E6POS carts[50]

gMasterFlag = TRUE

GLOBAL INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO IR_STOPM ( )
INTERRUPT ON 3
BAS (#INITMOV,0 )
; The first movement for Kuka must be a PTP movement
$BWDSTART = FALSE
PDAT_ACT = PPDAT_1
FDAT_ACT = FPDAT_2
BAS(#PTP_PARAMS, PPDAT_1.vel)

; Comment out this line if there is an error
SET_CD_PARAMS (0)
PTP $AXIS_ACT

; S2: Connect to IPC
; S2: 连接工控机
xyzOpenSocket("MasterClient")
gStrOpSuccess = STRCLEAR(gSendgMotMsg[])

; flow_name[] = "traj_sync.t"
; err_code = xyzSwitchFlow(flow_name[])
; IF (err_code <> 0) THEN  ;
;    HALT;
;    GOTO err_exit;
; ENDIF

; S3: Switch to the current item
; S3: 切换到当前工件
ws_id = 0
item_codename[] = "item1"
err_code = xyzSwitchItem(ws_id, item_codename[])
IF (err_code <> 0) THEN  ;
    HALT;
    GOTO err_exit;
ENDIF

; IO operation
; $OUT[500] = False
WAIT SEC 0.5

; S4: Move to home pose
; S4: 移动到 home 位姿
; you should use WorkVisual to set the home pose,
; or use pendant to set the home pose
; home_pose = {X 578.80, Y -9.12, Z 541.47, A 173.34,B -4.38,C -177.88}
; xyzSetMoveJParameters(10,20,0) ; vel, acc, zone
; PTP home_pose
; xyzSetMoveLParameters(10,20,0)
; LIN home_pose

WHILE True
; S5: Request Pick and place trajectory
```

(下页继续)

(续上页)

```
   ; S5: 请求抓取放置轨迹
   ws_id = 0
   err_code = xyzReqPickPlace(ws_id)
   IF (err_code <> 0) THEN  ;
       HALT;
       GOTO err_exit;
   ENDIF

   ; S6: Get Pick in trajectory
   ; S6: 获取抓取入筐轨迹
   err_code = xyzGetPickin(0, pipeline_num, register_num, way_point_num, wp_type[],␣
↪joints[], carts[])
   IF (err_code <> 0) THEN
       ; S15
       HALT;
       GOTO err_exit;
   ENDIF

   ; if tote cleared, exit
   ; 如果没有可以抓取的物体，退出
   IF (way_point_num < 1) THEN
       GOTO clear_tore_exit
   ENDIF
   ; S8 Execute Pick in trajectory
   ; S8: 执行抓取入筐轨迹
   xyzExecutePickInTraj(way_point_num, wp_type[], joints[], carts[])
   ; IO operation IO 操作
   ; $OUT[500] = False

   ; S9: Get Pick out trajectory
   ; S9: 获取抓取出筐轨迹
   err_code = xyzGetPickout(0, pipeline_num, register_num, way_point_num, wp_type[],␣
↪joints[], carts[])
   IF (err_code <> 0) THEN  ;
       HALT;
       GOTO err_exit;
   ENDIF
   ; S10: Execute Pick out trajectory
   ; S10: 执行抓取出筐轨迹
   xyzExecutePickOutTraj(way_point_num, wp_type[], joints[], carts[])

   ; S11: Get Place in trajectory
   ; S11: 获取放置入筐轨迹
   err_code = xyzGetPlacein(0, pipeline_num, register_num, way_point_num, wp_type[],␣
↪joints[], carts[])
   IF (err_code <> 0) THEN  ;
       HALT;
       GOTO err_exit;
   ENDIF
   ; S12: Execute Place in trajectory
   ; S12: 执行放置入筐轨迹
   xyzExecuteTraj(way_point_num, wp_type[], joints[], carts[])

   ; IO operation IO 操作
   ; $OUT[500] = True
```

(下页继续)

(续上页)

```
   ; S13: Get Place out trajectory
   ; S13: 获取放置出筐轨迹
   err_code = xyzGetPlaceout(0, pipeline_num, register_num, way_point_num, wp_type[],␣
↪joints[], carts[])
   IF (err_code <> 0) THEN  ;
       HALT;
       GOTO err_exit;
   ENDIF
   ; S14: Execute Place out trajectory
   ; S14: 执行放置出筐轨迹
   xyzExecuteTraj(way_point_num, wp_type[], joints[], carts[])

   ENDWHILE

   err_exit:

   clear_tore_exit:
```

END

3D 轨迹移动异步运行程序

```
DEF xyz_TrajMove_Async()

    ; S1: Initialization
    ; S1: 初始化
    INT err_code
    CHAR flow_name[20]
    INT ws_id
    CHAR item_codename[20]
    E6POS home_pose
    INT pipeline_num
    INT register_num
    INT way_point_num
    INT wp_type[30]
    E6AXIS joints[50]
    E6POS carts[50]

    gMasterFlag = TRUE ; 表示正在运行机械臂主控

    ; 以下为中断处理等代码，不能删除
    GLOBAL INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO IR_STOPM ( )
    INTERRUPT ON 3
    BAS (#INITMOV,0 )
    ; The first movement for Kuka must be a PTP movement
    $BWDSTART = FALSE
    PDAT_ACT = PPDAT_1
    FDAT_ACT = FPDAT_2
    BAS(#PTP_PARAMS, PPDAT_1.vel)

    ; Comment out this line if there is an error
    SET_CD_PARAMS (0)
    PTP $AXIS_ACT

    ; S2: Connect to IPC
```

(下页继续)

(续上页)

```
    ; S2: 连接工控机
    xyzOpenSocket("MasterClient")
    gStrOpSuccess = STRCLEAR(gSendgMotMsg[])

    ; flow_name[] = "traj_async.t"
    ; err_code = xyzSwitchFlow(flow_name[])
    ; IF (err_code <> 0) THEN  ;
    ;    HALT;
    ;    GOTO err_exit;
    ; ENDIF

    ; S3: Switch to the current item
    ; S3: 切换到当前工件
    ws_id = 0
    item_codename[] = "item1"
    err_code = xyzSwitchItem(ws_id, item_codename[])
    IF (err_code <> 0) THEN  ;
        HALT;
        GOTO err_exit;
    ENDIF

    ; IO operation IO 操作
    ; $OUT[500] = False

    ; S4: Move to home pose
    ; S4: 移动到 home 位姿
    ; you should use WorkVisual to set the home_pose,
    ; or use pendant to set the home_pose
    ; home_pose = {X 578.80, Y -9.12, Z 541.47, A 173.34,B -4.38,C -177.88}
    ; xyzSetMoveJParameters(10,20,0) ; vel, acc, zone
    ; PTP home_pose
    ; xyzSetMoveLParameters(10,20,0)
    ; LIN home_pose

    WHILE True
        ; S5: Request Pick and place trajectory
        ; S5: 请求抓取放置轨迹
        ws_id = 0
        err_code = xyzReqPickPlace(ws_id)
        IF (err_code <> 0) THEN  ;
            HALT;
            GOTO err_exit;
        ENDIF

        ; S6: Get Pick in trajectory
        ; S6: 获取抓取入筐轨迹
        ws_id = 0
        err_code = xyzGetPickin(ws_id, pipeline_num, register_num, way_point_num, wp_
↪type[], joints[], carts[])
        IF (err_code <> 0) THEN  ;
            HALT;
            GOTO err_exit;
        ENDIF

        ; if tote cleared, exit
        ; 如果没有可以抓取的物体，退出
```

(下页继续)

(续上页)

```
        IF (way_point_num < 1) THEN
            GOTO clear_tore_exit
        ENDIF
        ; S8 Execute Pick in trajectory
        ; S8: 执行抓取入筐轨迹
        xyzExecutePickInTraj(way_point_num, wp_type[], joints[], carts[])
        ; IO operation IO 操作
        ; $OUT[500] = False

        ; S9: Get Pick out trajectory
        ; S9: 获取抓取出筐轨迹
        ws_id = 0
        err_code = xyzGetPickout(ws_id, pipeline_num, register_num, way_point_num, wp_
↪type[], joints[], carts[])
        IF (err_code <> 0) THEN  ;
            HALT;
            GOTO err_exit;
        ENDIF
        ; S10: Execute Pick out trajectory
        ; S10: 执行抓取出筐轨迹
        xyzExecutePickOutTraj(way_point_num, wp_type[], joints[], carts[])

        ; S11: request next pick and place plan in advance
        ; S11: 提前请求下一次抓取放置轨迹
        ws_id = 0
        err_code = xyzReqPickPlace(ws_id)
        IF (err_code <> 0) THEN  ;
            HALT;
            GOTO err_exit;
        ENDIF

        ; S12: Get Place in trajectory
        ; S12: 获取放置入筐轨迹
        ws_id = 0
        err_code = xyzGetPlacein(ws_id, pipeline_num, register_num, way_point_num, wp_
↪type[], joints[], carts[])
        IF (err_code <> 0) THEN  ;
            HALT;
            GOTO err_exit;
        ENDIF
        ; S13: Execute Place in trajectory
        ; S13: 执行放置入筐轨迹
        xyzExecuteTraj(way_point_num, wp_type[], joints[], carts[])

        ; IO operation IO 操作
        ; $OUT[500] = True

        ; S14: Get Place out trajectory
        ; S14: 获取放置出筐轨迹
        ws_id = 0
        err_code = xyzGetPlaceout(ws_id, pipeline_num, register_num, way_point_num,␣
↪wp_type[], joints[], carts[])
        IF (err_code <> 0) THEN  ;
            HALT;
            GOTO err_exit;
        ENDIF
```

(下页继续)

(续上页)

```
        ; S15: Execute Place out trajectory
        ; S15: 执行放置出筐轨迹
        xyzExecuteTraj(way_point_num, wp_type[], joints[], carts[])

    ENDWHILE

    err_exit:

    clear_tore_exit:

END
```

常见问题

工控机主控运行时，工控机接收不到数据

解决方法：检查 sps.sub 文件是否卡在某一行。如果卡住，需要在专家模式下重新启动 sps.sub 文件。

附录

问题：

1. 程序导入后，发现文件出现很多错误，点开错误列表，出现 未定义的子程序 SET_CD_PARAMS(0)

   解决方法：在任意文本编辑器，注释所有的 SET_CD_PARAMS(0) 语句，然后将代码重新导入示教器。

2. 工控机主控时，发现 MAX 机械臂位姿不会随着机械臂移动而更新

   排查方法：点击示教器顶部菜单栏的 S，检查 SUB 文件是否正常运行。正常运行时，应该是代码行号不断改变。如果代码行号不变，说明 SUB 文件卡住。需要：

   - 切换为手动模式
   - 更改用户权限为管理员
   - 在菜单栏 S 处，依次点击 停止，取消选择,选择/启动。
   - 然后重新启动工控机主控。

   注：出现这种情况，往往是上一次断开连接后，没有点击顶部信息栏的 OK 或 全部 OK, 请注意操作习惯。

14.2.13 Yaskawa motoman

此处介绍安装安川 motoman 机械臂驱动的相关事项。

驱动版本和使用要求

支持的机械臂类型

motoman 六轴机械臂、四轴机械臂（motoman 拆码垛机械臂）

控制器型号

YRC1000，YRC1000-MICRO

机械臂需要开通的功能

Motoman 机械臂出厂时需要激活以太网功能包

查看是否激活步骤：

1. 在设置中选择 CPU 重置，长按 主菜单重启机械臂
2. 进入 设置 -> 选项功能 -> 网络功能设定查看以太网是否可以使用

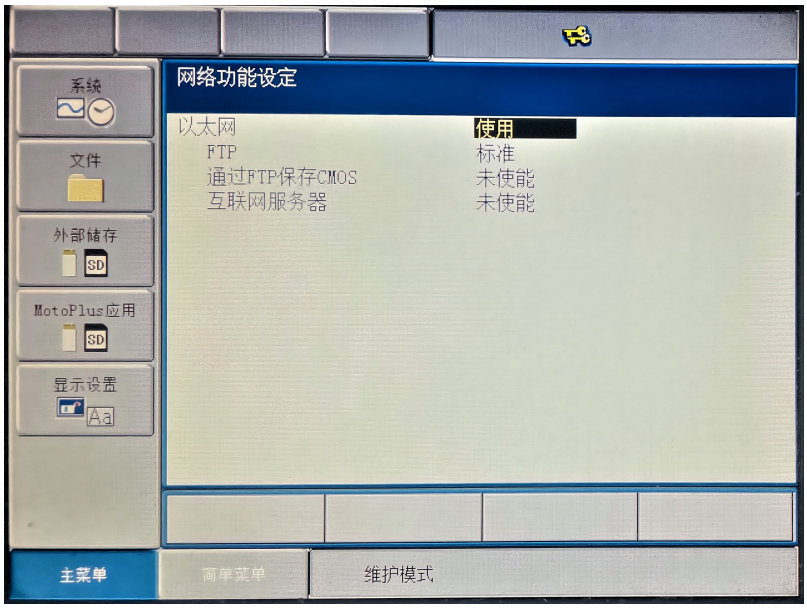


图 85: Motoman 控制器功能查看

安装驱动

motoman 机械臂驱动文件列表

–inform 安川机械臂前台运行函数

–xyz_master 机械臂主控运行函数

- XYZCartMoveBasicTpl.JBI **CartMove 基础模板程序**
- XYZCartMoveRepoTpl.JBI **CartMove 二次定位模板程序**
- XYZMasterConnect.JBI 机械臂主控 socket 连接函数
- XYZMasterExecTraj.JBI 机械臂主控执行轨迹函数
- XYZMasterInit.JBI 机械臂主控初始化函数
- XYZMasterProcCmd.JBI 机械臂主控命令处理函数
- XYZMasterTaskTest.JBI 测试程序

–xyz_motion 工控机主控运行函数

- XYZMotionConnect.JBI 工控机主控 motion socket 连接函数
- XYZMotionExecTraj.JBI 工控机主控执行轨迹函数
- XYZMotionInit.JBI 工控机主控初始化函数
- XYZMotionTask.JBI **工控机主控运行函数**
- XYZStatusConnect.JBI 工控机主控 status socket 连接函数

–motoplus_out 编译好的 motoplus .out 文件

- xyz_main_master.out 机械臂主控.out
- xyz_main_motion.out 工控机主控.out

–motoplus_source_code motoplus 源码

- xyz_const.h
- xyz_main_master.c
- xyz_main_motion.c
- xyz_master.c
- xyz_master.h
- xyz_motion.c
- xyz_motion.h
- xyz_sock_lib.c
- xyz_sock_lib.h
- xyz_utils.c
- xyz_utils.h

维护模式网络设定和烧录.out 文件

1. 以 **维护模式**开机：在按下 主菜单的状态下启动机械臂
2. 点击 系统-> 安全模式，在下拉菜单中选择 管理模式，密码为 "99999999999999" 输到头
3. 网络设置

- 从主菜单中选择 系统 -> 设置，按下 选项功能 -> LAN 接口设置，按下 选择按钮。
- 将 主机设置改为 手动设置, IP 地址改为 手动设置，修改 IP 和子网屏蔽：
  - IP 地址：192.168.37.100
  - 子网屏蔽：255.255.255.0
- 按下实体键盘右下 回车按钮，在所显示的确认对话框中选择 是

4. 启用 Motoplus 功能

- 从主菜单中选择 设置，进入 选项功能
- 在选择 MotoPlus 功能后选择 使用
- 在所显示的各确认对话框中选择 是

5. 将程序复制到 USB 存储器，将 USB 存储器插入示教器的 USB 端口
6. Motoplus(.out) 程序导入

- 在主菜单的 MotoPlus 应用 -> 装置中，选择已保存文件的 USB 设备
- 从主菜单中选择 MotoPlus 应用 -> 安装（用户应用程序）。注：若已存在.out，则需要先删除
- 在选择 .out 后按下 选择按钮

导入 INFORM 文件

1. 以正常方式开机（不按住主菜单）
2. 主菜单-> 外部储存 -> 装置 -> 选择装置，如 USB 示教编程器。
3. 主菜单 -> 外部储存 -> 安装-> 程序-> 选择所有的.JBI-> 是。

此时，程序已经全部导入。

运行程序

通讯说明

工控机和 motoman 机械臂的通讯方式为 socket：工控机作为 socket server(服务端)，机械臂作为 socket client（客户端）。

启动程序

运行工控机主控

- 确认已在维护模式下，烧录过 xyz_main_motion.out。
- 工控机启动 robot_driver_node 后，上伺服，启动 XYZMotionTask.JBI，即可。

运行机械臂主控

- 确认已在维护模式下，烧录过 xyz_main_master.out。
- 针对机械臂工作站的运行逻辑，新建机械臂主控运行程序。
- 工控机启动 robot_server 后，上伺服，启动新建立的机械臂主控程序即可。

API 说明

motoman 工控机主控支持的 API

指令号	命令	支持情况
100	获取机械臂程序版本号	支持
101	发送速度数据	支持
102	发送加速度数据	支持
103	发送圆滑过渡参数	支持
104	发送工具坐标系 (TCP) 参数	不支持
105	设置数字量输出	支持
106	SetJointsMovej	支持
107	SetCartMovel	支持
108	SetJoinsMovel	支持
109	SetCartMovej	支持
110	MovejSequence	支持
111	MovelSequence	支持
112	SetJointsMovejDo	不支持
113	SetCartMovelDo	不支持
114	SetJointsMovelDo	不支持
115	SetJointsMovejGroupDo	不支持
116	SetCartMovelGroupDo	不支持
117	SetJointsMovelGroupDo	不支持
118	MovelUntil	不支持
119	获取数字量输入状态	支持
120	获取数字量输出状态	支持
121	获取模拟量输入口数值	不支持
122	获取机械臂当前角度	支持
123	获取机械臂当前位姿	支持
200	机械臂后台发送状态	支持

motoman 工控机主控定义了一些全局变量，不允许用户使用，占用情况如下 (`XYZMotionInit.JBI`):

B 字节型		
地址	含义	说明 (是否用户可写)
B000	motion 通讯标志位	x
B001	status 通讯标志位	x
B002	内部使用	x
B003	内部使用	x
B004	内部使用	x
B005~B009	预留，请勿使用	x

I 整数型		
地址	含义	说明 (是否用户可写)
I000	motion 通讯端口号	x
I001	status 通讯端口号	x
I002	speed_joint	x
I003	speed_cart	x
I004	override pl	x
I005	acc	x
I006	dec	x
I007~I019	预留，请勿使用	x
I020	接收到的 cmd	x
I021	下发轨迹的点位数	x
I022	单点运动还是走轨迹	x
I023	运动类型	x
I024	error_code	x
I025~I029	预留，请勿使用	x
I030	执行轨迹中的标志位	x
I031	执行轨迹中的标志位	x
I032~I039	预留，请勿使用	x

P 位置型		
地址	含义	说明 (是否用户可写)
P040	单点运动	x
P041~P049	预留，请勿使用	x
P050~P100	走轨迹	x

S 字符串		
地址	含义	说明 (是否用户可写)
S000	服务器 IP 地址	x

motoman 机械臂主控支持的 API

motoman 机械臂主控定义了一些全局变量，不允许用户占用，占用情况如下 (`XYZMasterInit.JBI`):

B 字节型		
地址	含义	说明 (是否用户可写)
B000	通讯标志位	x
B001	发送命令标志位	x
B002~B005	预留，请勿使用	x

I 整数型		
地址	含义	说明 (是否用户可写)
I000	服务器端口号	√
I001	视觉服务 id	√
I002	设置的 token	√
I003	uws_id	√
I004	ws_id	√
I005~I009	预留，请勿使用	x
I010	请求的命令封装好了，调函数即可	x
I011	error_code	read only
I012	工控机返回的 token	read only
I013	(grasp_pose/object_pose/轨迹点) 数量	read only
I014	object_pose_type	read only
I015	心跳值	read only
I016	pipeline_num	read only
I017	register_num	read only
I018~I019	预留，请勿使用	x
I020~I060	轨迹点中每个点的类型	read only

P 位置型		
地址	含义	说明 (是否用户可写)
P000	grasp_pose	read only
P001	object_pose	read only
P002	tote_pose	read only
P003~P019	预留，请勿使用	x
P020~P060	轨迹点	read only

S 字符串		
地址	含义	说明 (是否用户可写)
S000	服务器 IP 地址	√
S001	app_name	√
S002	flow_name	√
S003	tool_name	√
S004	strategy_name	√
S005	object_name	√
S006~S010	预留，请勿使用	x

心跳信号 ()

motoplus 后台定期发送心跳信号，无需主动调用，100s 更新一次心跳值到 I015。

切换应用 ()

```
SET S001 "111"
CALL JOB:XYZMasterProcCmd ARGF501
```

输入:

- S001: 应用名称

输出:

- I011: 返回 error code

切换 **flow**()

```
SET S002 "222"
CALL JOB:XYZMasterProcCmd ARGF502
```

输入:

- S002: 任务 flow 名称

输出:

- I011: 返回 error code

切换工具 ()

```
SET S003 "333"
CALL JOB:XYZMasterProcCmd ARGF503
```

输入:

- S003: 工具 id

输出:

- I011: 返回 error code

请求拍照 ()

```
SET I001 0
CALL JOB:XYZMasterProcCmd ARGF504
SET LI000 I012
```

输入:

- I001: 视觉服务 id

输出:

- I012: 拍照返回的 token，随后将 I012 赋给 LI000
- I011: 返回 error code

获取拍照结果 ()

```
SET I002 LI000
CALL JOB:XYZMasterProcCmd ARGF505
```

输入:

- I002: 设置 token 值，由拍照返回的 token LI000 传来

输出：

- I011：返回 error code

拍照 ()

```
SET I001 0
CALL JOB:XYZMasterProcCmd ARGF506
```

输入：

- I001：视觉服务 id

输出：

- I011：返回 error code

请求抓取目标点位 ()

```
SET I004 0
CALL JOB:XYZMasterProcCmd ARGF507
SET LI000 I012
```

输入：

- I004：工作空间 id

输出：

- I012：返回的 token，随后将 I012 赋给 LI000
- I011：返回 error code

获取抓取目标点位 ()

```
SET I002 LI000
CALL JOB:XYZMasterProcCmd ARGF508
```

输入：

- I002：设置 token 值，来自于之前“请求抓取目标点位”的 token 返回值

输出：

- P000：目标点位姿
- I013：当前有多少个可供抓取的点
- I016：pipeline 文件 number 值
- I017：用到的注册文件的注册 number 值
- I011：返回 error code

请求物体位姿 ()

```
SET I004 0
CALL JOB:XYZMasterProcCmd ARGF509
SET LI000 I012
```

输入：

- I004：工作空间 id

输出：

- I012：返回的 token，随后将 I012 赋给 LI000
- I011：返回 error code

获取物体位姿 ()

```
SET I002 LI000
CALL JOB:XYZMasterProcCmd ARGF510
```

输入：

- I002：工件位姿 token

输出：

- P001：物体位姿
- I013：当前物体位姿个数
- I014：物体姿态类型
- I011：返回 error code

重置任务 ()

```
CALL JOB:XYZMasterProcCmd ARGF511
```

输出：

- I011：返回 error code

发送机械臂当前角度 ()

```
CALL JOB:XYZMasterProcCmd ARGF512
```

输出：

- I011：返回 error code

发送机械臂当前笛卡尔位姿 ()

```
CALL JOB:XYZMasterProcCmd ARGF513
```

输出：

- I011：返回 error code

发送机械臂当前扩展轴位置 ()

```
CALL JOB:XYZMasterProcCmd ARGF514
```

输出：

- I011：返回 error code

请求 **pick** 动作规划 ()

暂不支持

```
CALL JOB:XYZMasterProcCmd ARGF515
```

输出：

- I011：返回 error code

请求 **place** 动作规划 ()

暂不支持

```
CALL JOB:XYZMasterProcCmd ARGF516
```

输出:

- I011: 返回 error code

请求 **pick** 和 **place** 规划 ()

```
SET I003 0
CALL JOB:XYZMasterProcCmd ARGF517
```

输入:

- I003: uws_id，目前先设置为 0 即可

输出:

- I011: 返回 error code

获取取料入框轨迹 ()

```
SET I003 0
CALL JOB:XYZMasterProcCmd ARGF518
' 随后可以直接调用轨迹执行
CALL JOB:XYZMasterExecTraj ARGF518
```

输入:

- I003: uws_id，目前先设置为 0 即可

输出:

- I011: 返回 error code
- I016: pipeline 文件 number 值
- I017: 用到的注册文件的注册 number 值
- 轨迹点

获取取料出框轨迹 ()

```
SET I003 0
CALL JOB:XYZMasterProcCmd ARGF519
' 随后可以直接调用轨迹执行
CALL JOB:XYZMasterExecTraj ARGF519
```

输入:

- I003: uws_id，目前先设置为 0 即可

输出:

- I011: 返回 error code
- I016: pipeline 文件 number 值
- I017: 用到的注册文件的注册 number 值
- 轨迹点

获取放料入框轨迹 ()

```
SET I003 0
CALL JOB:XYZMasterProcCmd ARGF520
' 随后可以直接调用轨迹执行
CALL JOB:XYZMasterExecTraj ARGF520
```

输入:

- I003: uws_id，目前先设置为 0 即可

输出:

- I011：返回 error code
- I016：pipeline 文件 number 值
- I017：用到的注册文件的注册 number 值
- 轨迹点

获取放料出框轨迹 ()

```
SET I003 0
CALL JOB:XYZMasterProcCmd ARGF521
' 随后可以直接调用轨迹执行
CALL JOB:XYZMasterExecTraj ARGF521
```

输入:

- I003: uws_id，目前先设置为 0 即可

输出:

- I011：返回 error code
- I016：pipeline 文件 number 值
- I017：用到的注册文件的注册 number 值
- 轨迹点

请求切换策略 ()

```
SET S004 "strat1"
CALL JOB:XYZMasterProcCmd ARGF522
```

输入:

- S004：策略名

输出:

- I011：返回 error code

料箱重定位 ()

```
CALL JOB:XYZMasterProcCmd ARGF523
```

输出:

- P002：料框位姿
- I011：返回 error code

工件在上手的二次定位 ()

```
CALL JOB:XYZMasterProcCmd ARGF524
```

输出:

- I011: 返回 error code

工件不在手上的二次定位 ()

```
CALL JOB:XYZMasterProcCmd ARGF525
' 随后可以直接调用轨迹执行
CALL JOB:XYZMasterExecTraj ARGF525
```

输出:

- I011: 返回 error code
- I016: pipeline 文件 number 值
- I017: 用到的注册文件的注册 number 值
- 轨迹点

获取工件姿态类型 ()

```
CALL JOB:XYZMasterProcCmd ARGF526
```

输出:

- I014: 工件类型信息
- I011: 返回 error code

重置工业码垛状态 ()

```
CALL JOB:XYZMasterProcCmd ARGF527
```

输出:

- I011: 返回 error code

切换工件 ()

```
SET I004 0
SET S005 "obj1"
CALL JOB:XYZMasterProcCmd ARGF528
```

输入:

- I004: 工作空间 id
- S005: 工件名称

输出:

- I011: 返回 error code

计算抓取目标点位 ()

该指令等价于请求抓取目标点位 + 获取抓取目标点位

```
SET I004 0
CALL JOB:XYZMasterProcCmd ARGF529
```

输入：

- I004：工作空间 id

输出：

- P000：目标点位姿
- I013：当前有多少个可供抓取的点
- I016：pipeline 文件 number 值
- I017：用到的注册文件的注册 number 值
- I011：返回 error code

获取物体位姿 ()

```
SET I004 0
CALL JOB:XYZMasterProcCmd ARGF530
```

输入：

- I004：工作空间 id

输出：

- P001：物体位姿
- I013：当前物体位姿个数
- I014：物体姿态类型
- I011：返回 error code

案例/模板说明

CartMove 模板

以下为机械臂主控 CartMove 模板程序，注意对工控机返回的 error_code 进行判断。

CartMove 基本模板：XYZCartMoveBasicTpl.JBI

CartMove 二次定位模板：XYZCartMoveRepoTpl.JBI

TrajMove 模板

```
/JOB
//NAME XYZMasterTaskTest
//POS
///NPOS 0,0,0,0,0,0
//INST
///DATE 2022/11/05 09:40
///ATTR SC,RW
///GROUP1 RB1
///LVARS 5,12,5,5,5,5,0,0
NOP
' 以上为编辑程序过程中，自动生成的内容
' init 机械臂主控初始化
CALL JOB:XYZMasterInit
```

(下页继续)

(续上页)

```
'
' connect to server  连接机械臂主控 server
CALL JOB:XYZMasterConnect ARGF"192.168.37.101" ARGF11111
'
'
' switch_strat  请求切换策略
'   set strat_name 设定策略名
SET S004 "strat1"
CALL JOB:XYZMasterProcCmd ARGF522
IFTHENEXP I011<>0
    MSG "switch_obj NG"
    ABORT
ENDIF
'
'
' req_pick_place 请求pick和place规划
SET I003 0
CALL JOB:XYZMasterProcCmd ARGF517
IFTHENEXP I011<>0
    MSG "req_pick_place NG"
    ABORT
ENDIF
'
' get_pick_in 获取取料入框轨迹
SET I003 0
CALL JOB:XYZMasterProcCmd ARGF518
IFTHENEXP I011<>0
    MSG "get_pick_in NG"
    ABORT
ENDIF
'   check pose_type
IFTHENEXP I014<>1
    MSG "get_pick_in:pose_type NG"
    ABORT
ENDIF
' 执行取料入框轨迹
CALL JOB:XYZMasterExecTraj ARGF518
'
' get_pick_out 获取取料出框轨迹
SET I003 0
CALL JOB:XYZMasterProcCmd ARGF519
IFTHENEXP I011<>0
    MSG "get_pick_out NG"
    ABORT
ENDIF
'   check pose_type
IFTHENEXP I014<>1
    MSG "get_pick_out:pose_type NG"
    ABORT
ENDIF
' 执行取料出框轨迹
CALL JOB:XYZMasterExecTraj ARGF519
'
' get_place_in 获取放料入框轨迹
SET I003 0
CALL JOB:XYZMasterProcCmd ARGF520
```

(下页继续)

(续上页)

```
IFTHENEXP I011<>0
    MSG "get_place_in NG"
    ABORT
ENDIF
'   check pose_type
IFTHENEXP I014<>1
    MSG "get_place_in:pose_type NG"
    ABORT
ENDIF
' 执行放料入框轨迹
CALL JOB:XYZMasterExecTraj ARGF520
'
' get_place_out  放料入框轨迹
SET I003 0
CALL JOB:XYZMasterProcCmd ARGF521
IFTHENEXP I011<>0
    MSG "get_place_out NG"
    ABORT
ENDIF
'   check pose_type
IFTHENEXP I014<>1
    MSG "get_place_out:pose_type NG"
    ABORT
ENDIF
' 执行放料出框轨迹
CALL JOB:XYZMasterExecTraj ARGF521
'
'
END
```

常见问题

工控机主控运行时，连接不上工控机

原因：motoplus 后台卡在某个位置，需要重启机械臂。

附录

14.2.14 Nachi

此处介绍安装 nachi 机械臂驱动的相关事项。

驱动版本和使用要求

支持的机械臂类型

nachi 六轴机械臂

控制器型号

CFD

机械臂需要开通的功能

自带 socket 通讯功能，无需开通

安装驱动

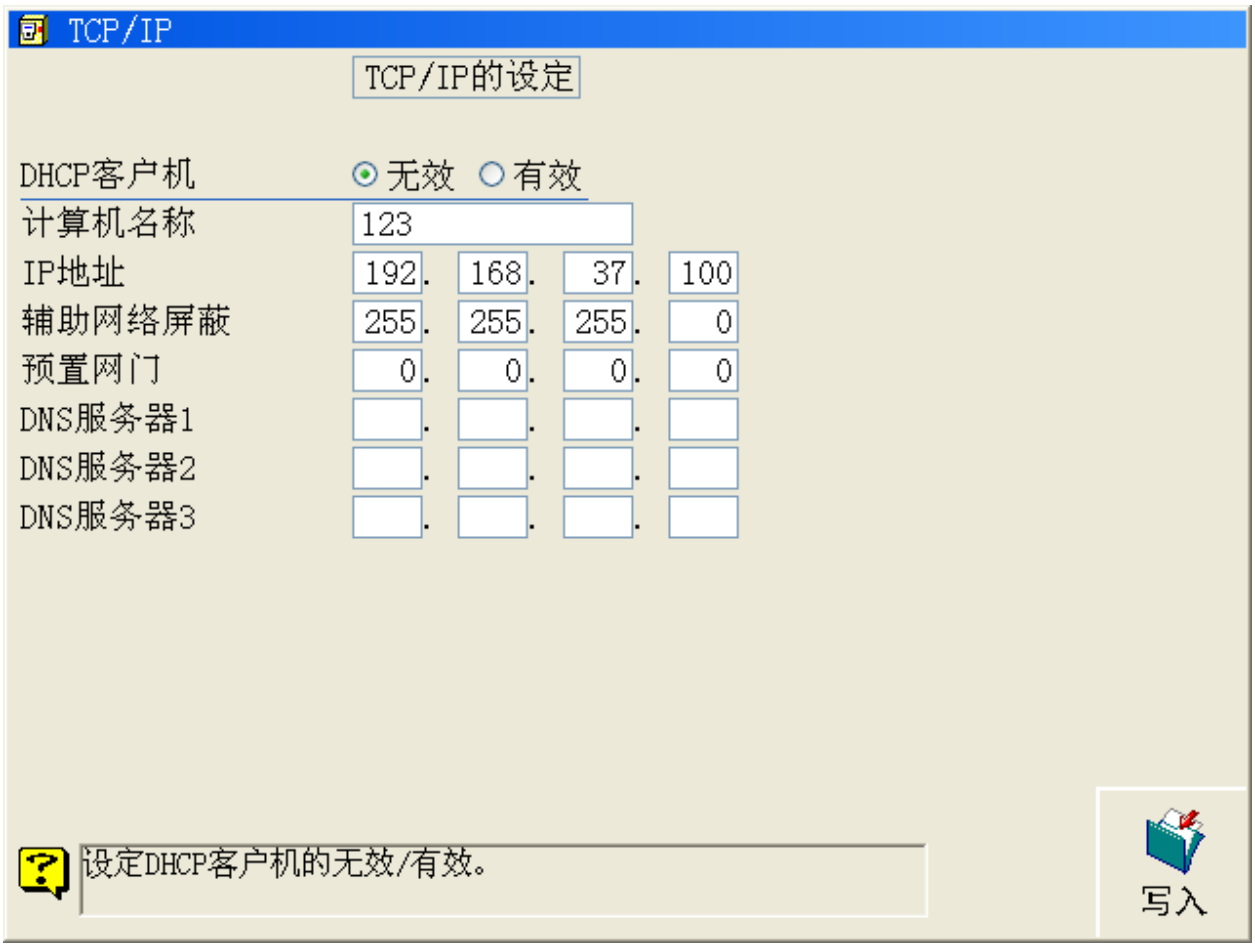
nachi 机械臂驱动文件列表

–**nachi**

- MZ04-01-A.000(工控机主控前台源程序)
- MZ04-01-A.001(机械臂主控前台模板源程序，用户根据此模板进行修改)
- MZ04-01-A.100(机械臂主控测试源程序，可以不导入)
- PUBLIC.INC(全局变量定义区)
- USERTASK-A.100(工控机主控用户任务源程序)
- USERTASK-A.200(工控机主控用户任务源程序)
- USERTASK-A.300(机械臂主控用户任务源程序)
- USRPROC.INC(全局函数定义区)
- VARIABLE.INC(全局变量、寄存器使用范围定义区)

设定机械臂 IP

1. 用钥匙开关在控制柜上将机械臂切换到手动模式。
2. 切换用户权限: 示教器 R 按钮 -> 输入数字 314 -> 输入密码，默认密码是 123456 或者 86055，或者咨询机械臂提供商。
3. 示教器上执行：常数设定 -> 通讯 -> 以太网 -> TCP/IP, 按如下进行设置后 -> 写入


TCP/IP
TCP/IP的设定
DHCP客户机
无效
有效
计算机名称
123
IP地址
192. 168. 37. 100
辅助网络屏蔽
255. 255. 255. 0
预置网门
0. 0. 0. 0
DNS服务器1
DNS服务器2
DNS服务器3
设定DHCP客户机的无效/有效。
写入

导入程序

那智机械臂程序导入有两种方法，一种是使用 FD 软件，一种是使用 U 盘。这里介绍使用 U 盘导入的方法。

程序文件准备:

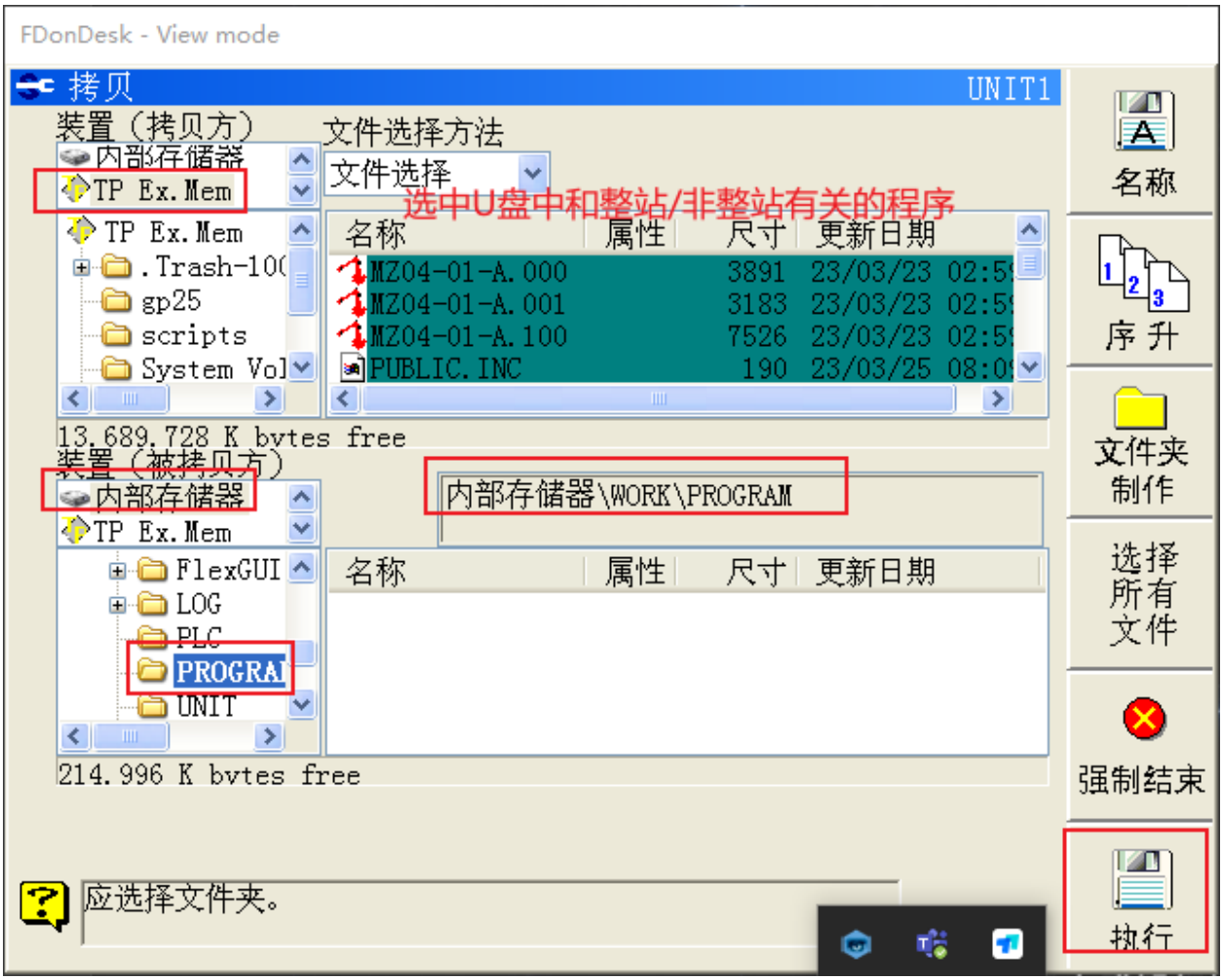
1. 那智机械臂程序源文件名称中的 `MZ04-01` 是机械臂的型号，不同机械臂需要自行修改为相应的程序文件名称。
2. 确保那智机械臂程序文件尾行格式是 CRLF。

U 盘准备:

1. 确保 U 盘被格式为为 `FAT32`。
2. 将那智的程序文件拷贝到 U 盘根目录下，其中的 `MZ04-01-A.100` 可以导入，也可以不导入。

导入并编译:

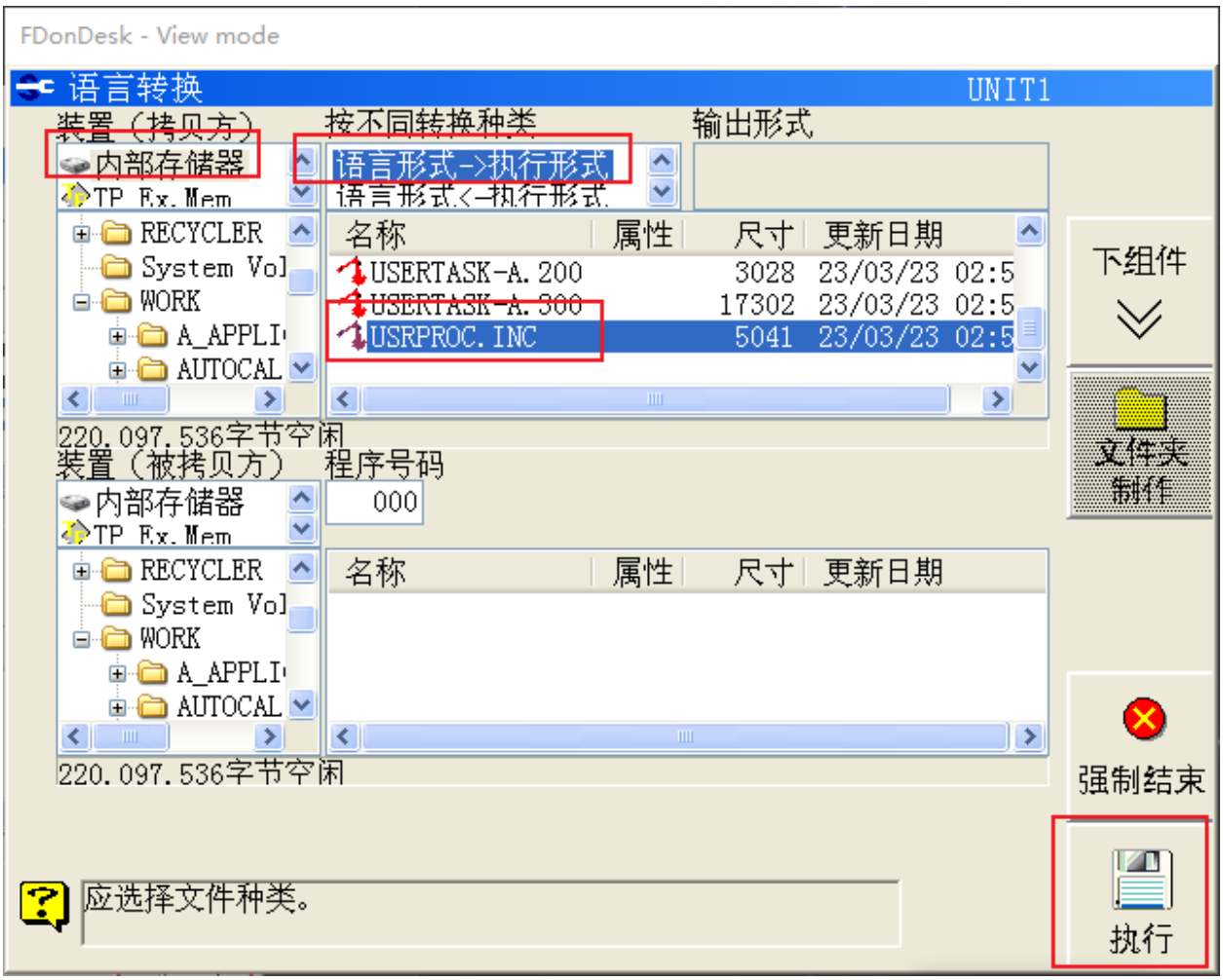
1. 机械臂切换到手动档，并切换用户权限
2. 将 U 盘插入到机械臂示教器 USB 接口
3. 示教器上执行: 维修-> 文件操作 -> 拷贝 -> 选择 U 盘中所有程序文件并拷贝到 `/WORK/PROGRAM` 下:



4. 重启机械臂!!! 否则后续程序编译会有问题!

因为 PUBLIC.INC 需要重启后会才能自动编译成 PUBLIC.DAT

5. 进行程序编译：维修 -> 程序转换 -> 语言转换 -> 先对 USPRPOC.INC 进行编译，方法如下：



注意编译有顺序性，必须首先对 USRPROC.INC 进行编译。

6. 同样的编译方法，按 **顺序** 对剩下的程序进行编译：USERTASK-A.100，USERTASK-A.200，USERTASK-A.300，MZ04-01-A.000，MZ04-01-A.001，MZ04-01-A.100。

运行程序

通讯说明

工控机和 nachi 机械臂的通讯方式为 socket：工控机作为 socket server(服务端)，机械臂作为 socket client（客户端)。

启动程序

运行机械臂主控

用户根据项目实际需要，修改机械臂主控模板程序 `MZ04-01-A.001` ，然后导入到机械臂后重新编译并加载并运行。

注：也可以直接在示教器上直接修改程序内容。

运行步骤:

1. 切换到手动档，并切换用户权限
2. 停止后台程序，并设置后台程序的优先级别

   示教器上打开 **用户任务监视器**窗口: 维修 -> 监视器 2 (或其他监视器) -> 用户任务 -> 用户任务监视器

   如果是第一次运行，则此时看不到如上所述的后台任务 `100`, `200`, `300`，这时需要加载这 3 个后台任务，方法为:

   将光标移动到 用户任务监视器任意处，然后按示教器上的 编辑按钮，进入到编辑模式:

   然后按如下方法添加 3 个后台任务并设定优先级别为 6 -> 写入

   如果 执行状态那一列有正在启动中的后台程序，则需要先停止这些后台任务，方法为：移动光标到 **启动中**的后台程序

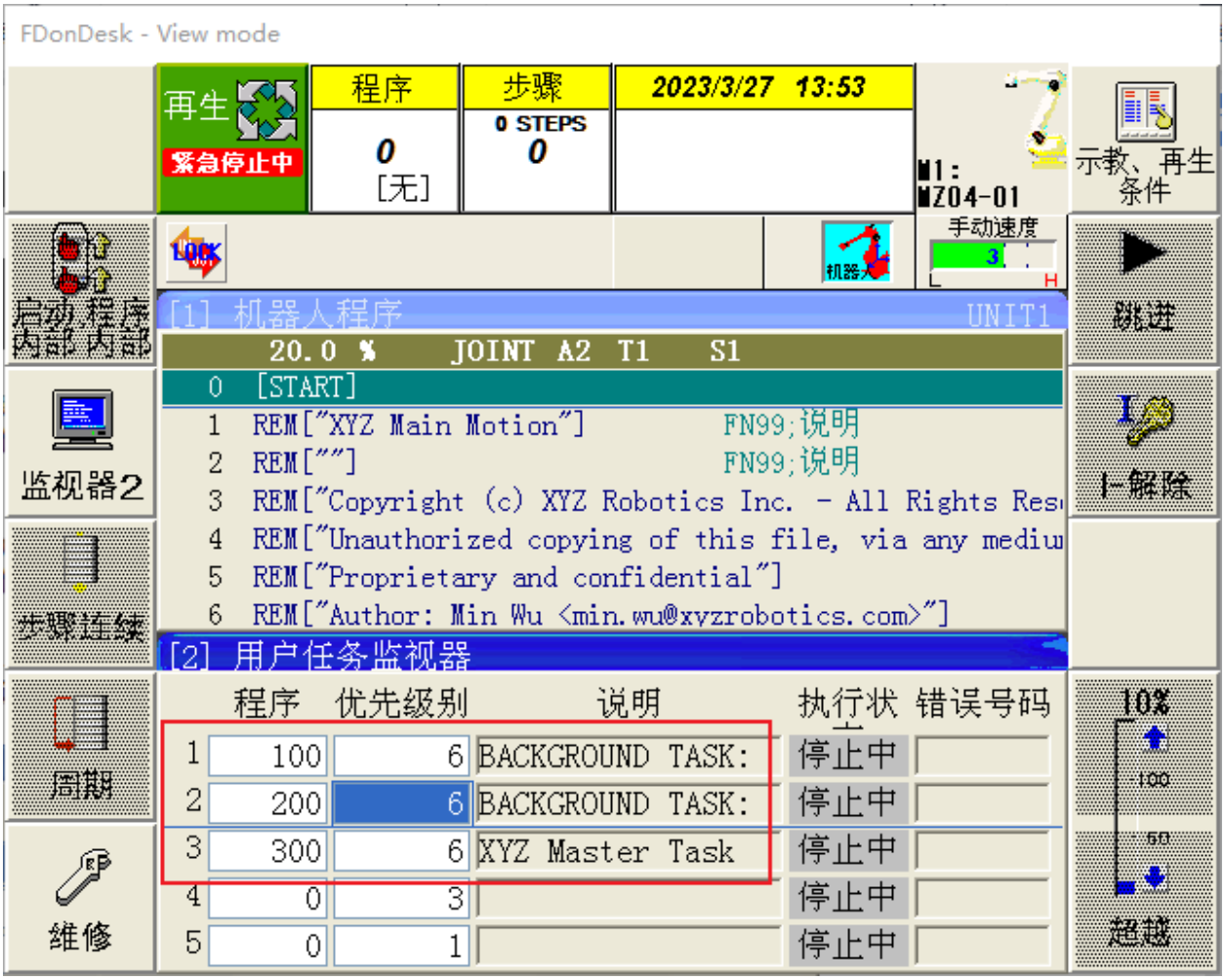
   然后按示教器的 `ENABLE` + `OFF`
3. 加载机械臂程序 `001`
4. 切换到自动档
5. 上使能：同时按住示教器左上方的的 `ENABLE` + `MOTOR ON`
6. 运行程序：同时按住示教器的 `ENABLE` + `SHIFT` + `GO`

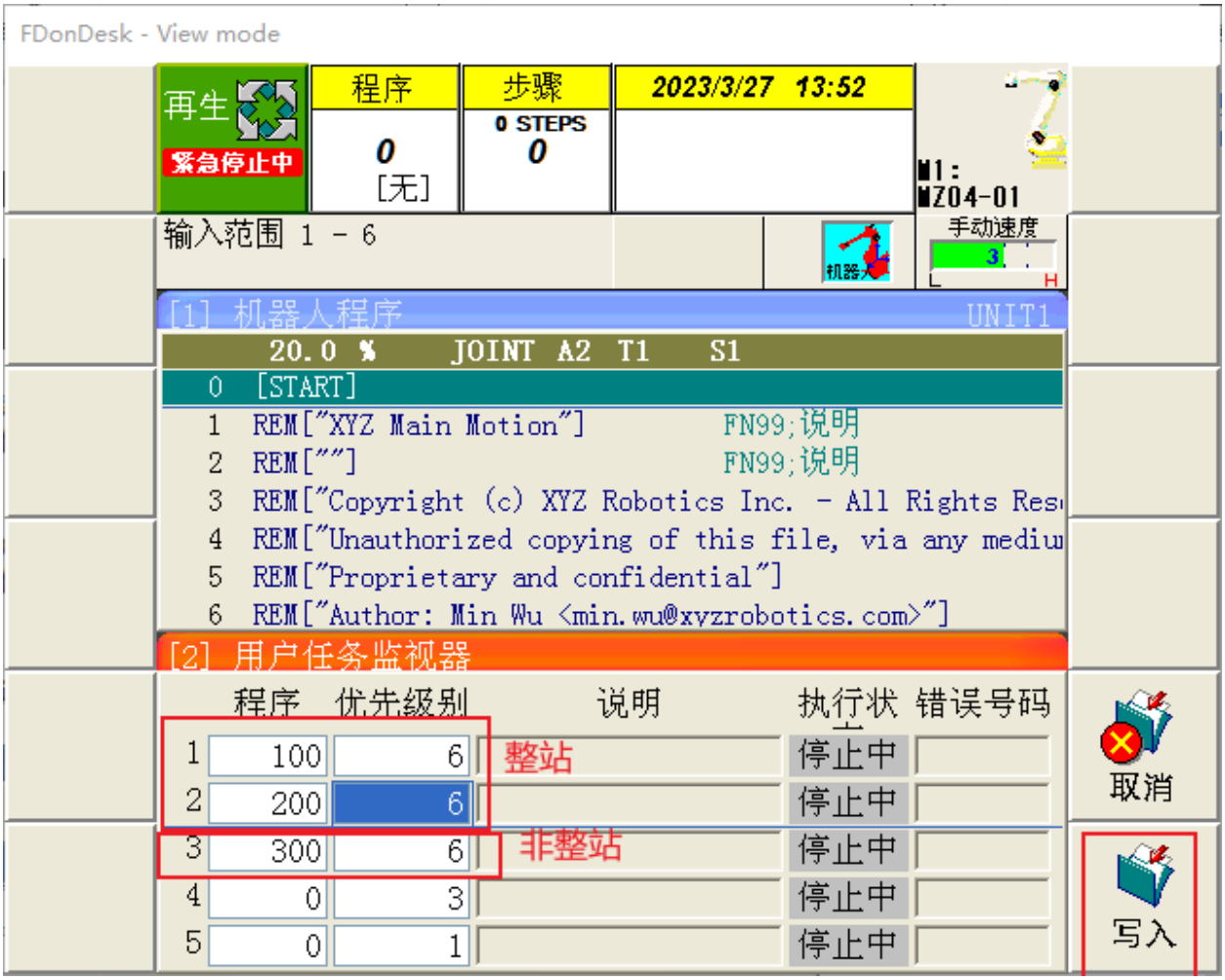
运行工控机主控

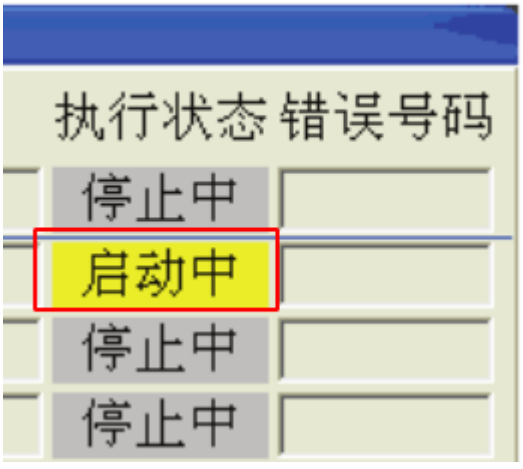
用户根据项目实际使用的机械臂修改工控机主控前台程序名 `MZ04-01-A.000` ，然后导入到机械臂后重新编译并加载并运行。

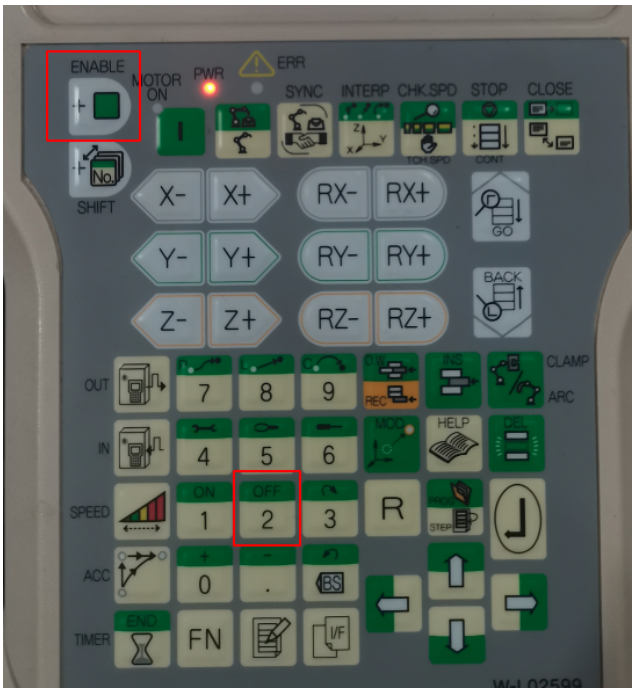
运行步骤:

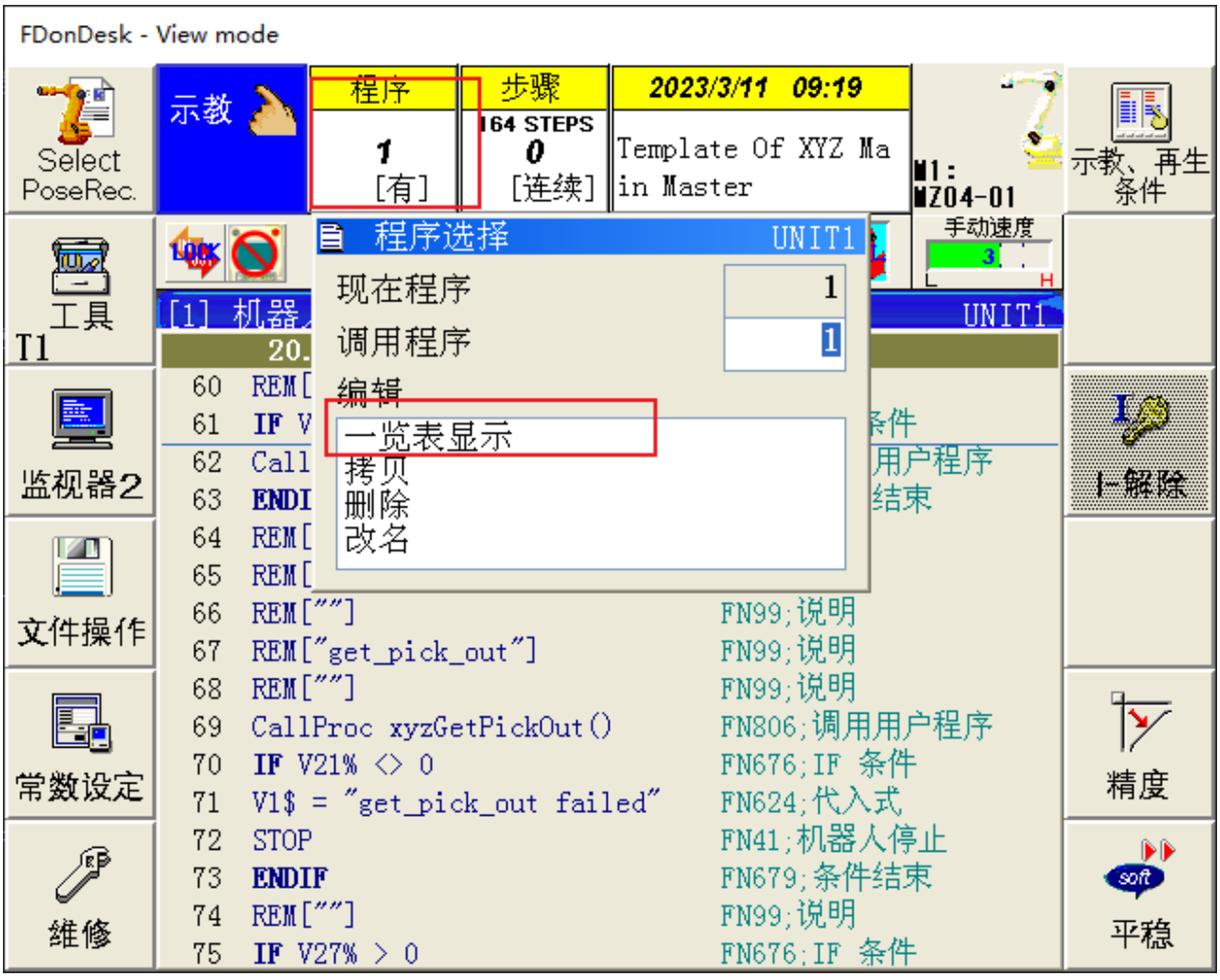
方法同机械臂主控的运行步骤，执行前需要确认后台程序优先级为 6，然后前台程序为 `000`


FDonDesk - View mode
再生
紧急停止中
程序
0
[无]
步骤
0 STEPS
0
2023/3/27 13:53
M1:
MZ04-01
示教、再生
条件
手动速度
3
L
H
LOCK
机器人
启动.程序
内部.内部
跳进
监视器2
I-解除
步骤连续
周期
维修
10%
100
50
超越
[1] 机器人程序
UNIT1
20.0 % JOINT A2 T1 S1
0 [START]
1 REM["XYZ Main Motion"] FN99;说明
2 REM[""] FN99;说明
3 REM["Copyright (c) XYZ Robotics Inc. - All Rights Res
4 REM["Unauthorized copying of this file, via any mediu
5 REM["Proprietary and confidential"]
6 REM["Author: Min Wu <min.wu@xyzrobotics.com>"]
[2] 用户任务监视器
程序 优先级别 说明 执行状 错误号码
1 100 6 BACKGROUND TASK: 停止中
2 200 6 BACKGROUND TASK: 停止中
3 300 6 XYZ Master Task 停止中
4 0 3 停止中
5 0 1 停止中


FDonDesk - View mode
再生
紧急停止中
程序
0
[无]
步骤
0 STEPS
0
2023/3/27 13:52
M1:
MZ04-01
输入范围 1 - 6
机器人
手动速度
3
[1] 机器人程序
UNIT1
20.0 % JOINT A2 T1 S1
0 [START]
1 REM["XYZ Main Motion"] FN99;说明
2 REM[""] FN99;说明
3 REM["Copyright (c) XYZ Robotics Inc. - All Rights Res
4 REM["Unauthorized copying of this file, via any mediu
5 REM["Proprietary and confidential"]
6 REM["Author: Min Wu <min.wu@xyzrobotics.com>"]
[2] 用户任务监视器
程序 优先级别 说明 执行状 错误号码
1 100 6 整站 停止中
2 200 6 停止中
3 300 6 非整站 停止中
4 0 3 停止中
5 0 1 停止中
取消
写入


执行状态 错误号码
停止中
启动中
停止中
停止中


ENABLE
MOTOR ON
PWR
ERR
SYNC
INTERP
CHK SPD
STOP
CLOSE
SHIFT
X-
X+
RX-
RX+
Y-
Y+
RY-
RY+
Z-
Z+
RZ-
RZ+
GO
BACK
CLAMP
ARC
HELP
SPEED
ACC
TIMER
FN
R


FDonDesk - View mode
Select PoseRec.
示教
程序
1
[有]
步骤
164 STEPS
0
[连续]
2023/3/11 09:19
Template Of XYZ Ma
in Master
M1:
MZ04-01
手动速度
3
示教、再生条件
工具
T1
监视器2
文件操作
常数设定
维修
程序选择 UNIT1
现在程序 1
调用程序 1
编辑
一览表显示
拷贝
删除
改名
[1] 机器人
UNIT1
20.
60 REM[
61 IF V
62 Call
63 ENDI
64 REM[
65 REM[
66 REM[""] FN99;说明
67 REM["get_pick_out"] FN99;说明
68 REM[""] FN99;说明
69 CallProc xyzGetPickOut() FN806;调用用户程序
70 IF V21% <> 0 FN676;IF 条件
71 V1$ = "get_pick_out failed" FN624;代入式
72 STOP FN41;机器人停止
73 ENDIF FN679;条件结束
74 REM[""] FN99;说明
75 IF V27% > 0 FN676;IF 条件
条件
用户程序
结束
I-解除
精度
soft
平稳

API 说明

nachi 工控机主控支持的 API

指令号	命令	支持情况
100	获取机械臂程序版本号	支持
101	发送速度数据	支持
102	发送加速度数据	支持
103	发送圆滑过渡参数	支持
104	发送工具坐标系 (TCP) 参数	不支持
105	设置数字量输出	支持
106	SetJointsMovej	支持
107	SetCartMovel	支持
108	SetJoinsMovel	支持
109	SetCartMovej	支持
110	MovejSequence	不支持
111	MovelSequence	不支持
112	SetJointsMovejDo	不支持
113	SetCartMovelDo	不支持
114	SetJointsMovelDo	不支持
115	SetJointsMovejGroupDo	不支持
116	SetCartMovelGroupDo	不支持
117	SetJointsMovelGroupDo	不支持
118	MovelUntil	不支持
119	获取数字量输入状态	支持
120	获取数字量输出状态	支持
121	获取模拟量输入口数值	不支持
122	获取机械臂当前角度	支持
123	获取机械臂当前位姿	支持
200	机械臂后台发送状态	支持

那智工控机主控定义了一些全局变量，不允许用户使用，占用情况如下 (`VARIABLE.INC`):

整数变量 V%		
地址	含义	说明 (是否用户可写)
V50%	通讯连接标志位	read only
V51%	motion_socket 报错 1	read only
V52%	motion_socket 报错 2	read only
V53%	通讯接收长度	read only
V54%	内部使用	read only
V55%	内部使用	read only
V56%	通讯发送长度	read only
V57%	内部使用	read only
V58%	内部使用	read only
V60%	内部使用	read only
V61%	status_socket 报错 1	read only
V62%	status_socket 报错 2	read only
V63%	内部使用	read only
V64%	内部使用	read only
V65%	通讯发送长度	read only
V80%	服务端发送的 cmd	read only
V81%	服务端发送的 acc	read only
V82%	服务端发送的 zone	read only
V83%	服务端发送的 port	read only
V84%	服务端发送的 signal	read only

字符串变量 V$		
地址	含义	说明 (是否用户可写)
V10$	内部使用	x
V11$	所有收到信息	read only
V21$	内部使用	x
V22$	所有发送信息	read only
V30$	显示错误消息	read only
V31$	显示错误消息	read only

实数变量 V!		
地址	含义	说明 (是否用户可写)
V80!~V96!	内部使用	x

nachi 机械臂主控支持的 API

那智部分全局变量已经被占用，不允许用户使用，占用情况如下 (`VARIABLE.INC`):

整数变量 V%		
地址	含义	说明 (是否用户可写)
V1%	通讯连接标志位	x
V2%	发送命令标志位	x
V3%	服务端 ip 地址	可写
V4%	服务端端口号	可写
V5%	是否通讯失败	read only
V6%	socket 报错 1	read only
V7%	socket 报错 2	read only
V8%	内部使用	x
V9%	内部使用	x
V10%	内部使用	x
V11%	内部使用	x
V12%	发送的命令码	read only
V13%	设置发送给服务端的 ws_id	可写
V14%	设置发送给服务端的 token	可写
V15%~V20%	系统预留，勿用	x
V21%	服务端返回的 error_code	read only
V22%	服务端返回的 token	read only
V23%	服务端返回的 grasp_pose_num	read only
V24%	服务端返回的 grasp_pose_type	read only
V25%	服务端返回的 obj_pose_num	read only
V26%	服务端返回的 obj_pose_type	read only
V27%	服务端返回的轨迹点数量	read only
V28%	服务端返回的轨迹点 pose_type	read only
V29%~V58%	服务端返回的每个轨迹点的类型 (最多支持 30 个点)	read only

字符串变量 V$		
地址	含义	说明 (是否用户可写)
V1$	显示错误消息	read only
V2$	内部使用	x
V3$	内部使用	x
V4$~V13$	内部使用	x
V14$~V15$	系统预留，勿用	x

实数变量 V!		
地址	含义	说明 (是否用户可写)
V1!	x of grasp_pose	read only
V2!	y of grasp_pose	read only
V3!	z of grasp_pose	read only
V4!	a of grasp_pose	read only
V5!	b of grasp_pose	read only
V6!	c of grasp_pose	read only
V7!	x of object_pose	read only
V8!	y of object_pose	read only
V9!	z of object_pose	read only
V10!	a of object_pose	read only
V11!	b of object_pose	read only
V12!	c of object_pose	read only
V13!	x of tote_pose	read only
V14!	y of tote_pose	read only
V15!	z of tote_pose	read only
V16!	a of tote_pose	read only
V17!	b of tote_pose	read only
V18!	c of tote_pose	read only
V19!~V26!	内部使用	x
V301!~V306!	轨迹点 1	x
V307!~V312!	轨迹点 2	x
···	轨迹点 n	x
V475!~V480!	轨迹点 30	x

机械臂主控支持的 api 定义在 `USRPROC.INC`:

MasterInit()

初始化各变量

xyzMasterConnect()

连接 server，需要事先将 ip 地址设置到 `V3%`, 端口号设置到 `V4%`

xyzSwitchApp(*app_name*)

切换 app

参数 app_name(*STRING*) –app 的名称

xyzSwitchObj(*obj_name*)

切换工件

参数 obj_name(*STRING*) –app 的名称

xyzSwitchTool(*tool_id*)

切换工具

参数 tool_id(*STRING*) –工具名称

xyzReqCapImg()

请求拍照，需要事先将 ws_id 设置到 `V13%`

返回的 token 存放在 `V22%`

xyzGetCapImg()

获取拍照结果，需要事先将想查找的 token 设置到 `V14%`

xyzCapImg()

拍照并获取结果，等价于 xyzReqCapImg() + xyzGetCapImg()。需要事先将 ws_id 设置到 V13%

xyzReqGraspPose()

请求抓取目标点位，需要事先将 ws_id 设置到 V13%

返回的 token 存放在 V22%

xyzGetGraspPose()

获取抓取点位，需要事先将想查找的 token 设置到 V14%

返回的抓取点信息存放在 V23%, V24%, V1!~V6!

xyzReqObjPose()

请求物体位姿，需要事先将 ws_id 设置到 V13%

返回的 token 存放在 V22%

xyzGetObjPose()

获取物体位姿，需要事先将想查找的 token 设置到 V14%

返回的物体位姿存放在 V25%, V26%, V7!~V12!

xyzResetVision()

重置视觉，需要事先将 ws_id 设置到 V13%

xyzSendCurrentJoints()

发送机械臂当前角度

xyzSendCurrentCartPose()

发送机械臂当前 Cartesian

xyzSendCurrentExtJoints()

发送机械臂当前扩展轴位置，该指令暂不支持

xyzReqPick()

请求 pick 动作规划

xyzReqPlace()

请求 place 动作规划

xyzReqPickPlace()

请求 pick 和 place 规划

xyzGetPickIn()

获取取料入框轨迹，轨迹点信息存放在 V27%, V28% 等

xyzGetPickOut()

获取取料出框轨迹，轨迹点信息存放在 V27%, V28% 等

xyzGetPlaceIn()

获取放料入框轨迹，轨迹点信息存放在 V27%, V28% 等

xyzGetPlaceOut()

获取放料出框轨迹，轨迹点信息存放在 V27%, V28% 等

xyzSwitchStrat(*strat_name*)

请求切换策略

参数 strat_name(*STRING*)–策略名称

xyzUpdateTotePose()

料箱重定位，新的料箱位姿存放在 V13! ~ V18!

xyzUpdateObjPoseOnHand()

工件在手上的二次定位

xyzUpdateObjPoseToHand()

工件不在手上的二次定位，轨迹点信息存放在 V27%, V28% 等

xyzGetObjPoseType()

获取工件姿态类型，返回的姿态类型存放在 V26%

xyzResetPalletStatus()

重置工业码垛状态

案例/模板说明

机械臂主控主函数说明

用户需要根据项目情况，编写机械臂前台主程序。

nachi 的一部分寄存器已经被占用，用户不允许使用这些寄存器，详见 nachi api 界面。

以下为机械臂主控模板代码 (MZ04-01-A.001)，注意对工控机返回的 err_code 进行判断。

```
INCLUDE "VARIABLE"

' 初始化并启动后台程序
CallProc MasterInit()
KILLMCR 300, 0, 10000
DELAY 2
FORKMCR 300, 10000
DELAY 2

'
' 连接服务端
'
' 设置服务端ip地址(ip地址的最后一段)
V3% = 101
' 设置服务端端口
V4% = 11111
CallProc xyzMasterConnect()

'
' 切换app
'
```

(下页继续)

(续上页)

```
' 设置项目名称
L1$ = "111"
CallProc xyzSwitchApp(L1$)
' 需要对返回的error_code(V21%)进行判断，以下每条语句同理
IF V21% <> 0
    STOP
ENDIF

*MAIN_LOOP

'
' 请求pick
'
CallProc xyzReqPick()
IF V21% <> 0
    STOP
ENDIF

'
' 获取pick_in轨迹
'
CallProc xyzGetPickIn()
IF V21% <> 0
    STOP
ENDIF

' 轨迹点数量>0时运行轨迹
IF V27% > 0
    CallProc ExecTraj()
ENDIF


'
' 获取pick_out轨迹
'
CallProc xyzGetPickOut()
IF V21% <> 0
    STOP
ENDIF

' 轨迹点数量>0时运行轨迹
IF V27% > 0
    CallProc ExecTraj()
ENDIF

'
' 获取place_in轨迹
'
CallProc xyzGetPlaceIn()
IF V21% <> 0
    STOP
ENDIF

' 轨迹点数量>0时运行轨迹
IF V27% > 0
    CallProc ExecTraj()
```

(下页继续)

(续上页)

```
ENDIF


'
' 获取pick_out轨迹
'
CallProc xyzGetPlaceOut()
IF V21% <> 0
    STOP
ENDIF

' 轨迹点数量>0时运行轨迹
IF V27% > 0
    CallProc ExecTraj()
ENDIF

GOTO *MAIN_LOOP



END



'␣
↪*****************************************************************************************************
' SUB FUNCTIONS
'␣
↪*****************************************************************************************************

UsrProc ExecTraj()
    '
    ' set your own speed/override... here
    ' 如果需要分别调整出筐/入筐等的轨迹、圆滑值，可以在这里自定义调整
    '
    ' IF V12% = 518
    '     ' set pick_in speed/override...
    ' ELSEIF V12% = 519
    '     ' set pick_out speed/override...
    ' ELSEIF V12% = 520
    '     ' set place_in speed/override...
    ' ELSEIF V12% = 521
    '     ' set place_out speed/override...
    ' ELSEIF V12% = 525
    '     ' set update_obj_pose_to_hand speed/override...
    ' ELSE
    '     STOP
    ' ENDIF


    ' V27%: traj pose num
    L100% = V27% - 1
    FOR L101% = 0 TO L100% STEP 1
        V19! = V![301+6*L101%]
        V20! = V![302+6*L101%]
        V21! = V![303+6*L101%]
```

(下页继续)

(续上页)

```
        V22! = V![304+6*L101%]
        V23! = V![305+6*L101%]
        V24! = V![306+6*L101%]
        IF V%[29+L101%] = 11
            ' JOINTS_MOVEJ
            MOVEX A=3, AC=0, SM=0, M1J, P, (V19!,V20!,V21!,V22!,V23!,V24!),R=100.0,MS
        ELSEIF V%[29+L101%] = 12
            ' JOINTS_MOVEL
            MOVEX A=3, AC=0, SM=0, M1J, L, (V19!,V20!,V21!,V22!,V23!,V24!),R=100.0,MS
        ELSEIF V%[29+L101%] = 21
            ' CART_MOVEJ
            MOVEX A=3, AC=0, SM=0, M1X, P, (V19!,V20!,V21!,V22!,V23!,V24!),R=100.0,MS
        ELSEIF V%[29+L101%] = 22
            ' CART_MOVEL
            MOVEX A=3, AC=0, SM=0, M1X, L, (V19!,V20!,V21!,V22!,V23!,V24!),R=100.0,MS
        ELSE
            STOP
        ENDIF
    NEXT
EndProc
```

常见问题

1. 机械臂程序编译出错

   确保导入到机械臂中的源程序文件是 CRLF 结尾。上传 `PUBLIC.INC` 以后，需要重启机械臂才能生效。编译机械臂主控程序具有顺序性，必须首先编译 `USRPROC.INC` 。

2. 工控机主控手动运行时 socket 报错

   那智机械臂的问题，需要打到自动档进行测试。

附录

14.2.15 otc

此处介绍安装 otc 机械臂驱动的相关事项。

驱动版本和使用要求

支持的机械臂类型

otc 六轴机械臂

控制器型号

FD19

机械臂需要开通的功能

自带 socket 通讯功能，无需开通

安装驱动

otc 机械臂驱动文件列表

–otc

- MZ04-01-A.000(工控机主控前台源程序)
- MZ04-01-A.001(机械臂主控前台 CartMove 基础模板源程序)
- MZ04-01-A.002(机械臂主控前台 CartMove 二次定位模板源程序)
- MZ04-01-A.100(机械臂主控测试源程序，可以不导入)
- PUBLIC.INC(全局变量定义区)
- USERTASK-A.100(工控机主控用户任务源程序)
- USERTASK-A.200(工控机主控用户任务源程序)
- USERTASK-A.300(机械臂主控用户任务源程序)
- VARIABLE.INC(全局变量、寄存器使用范围定义区)

设定机械臂 IP

1. 用钥匙开关在控制柜上将机械臂切换到手动模式。
2. 切换用户权限: 示教器 `R` 按钮 -> 输入数字 `314` -> 输入密码，默认密码是 `123456` 或者 `86055`，或者咨询机械臂提供商。
3. 示教器上执行：`常数设定` -> `通讯` -> `以太网` -> `TCP/IP`, 按如下进行设置后 -> `写入`

导入程序

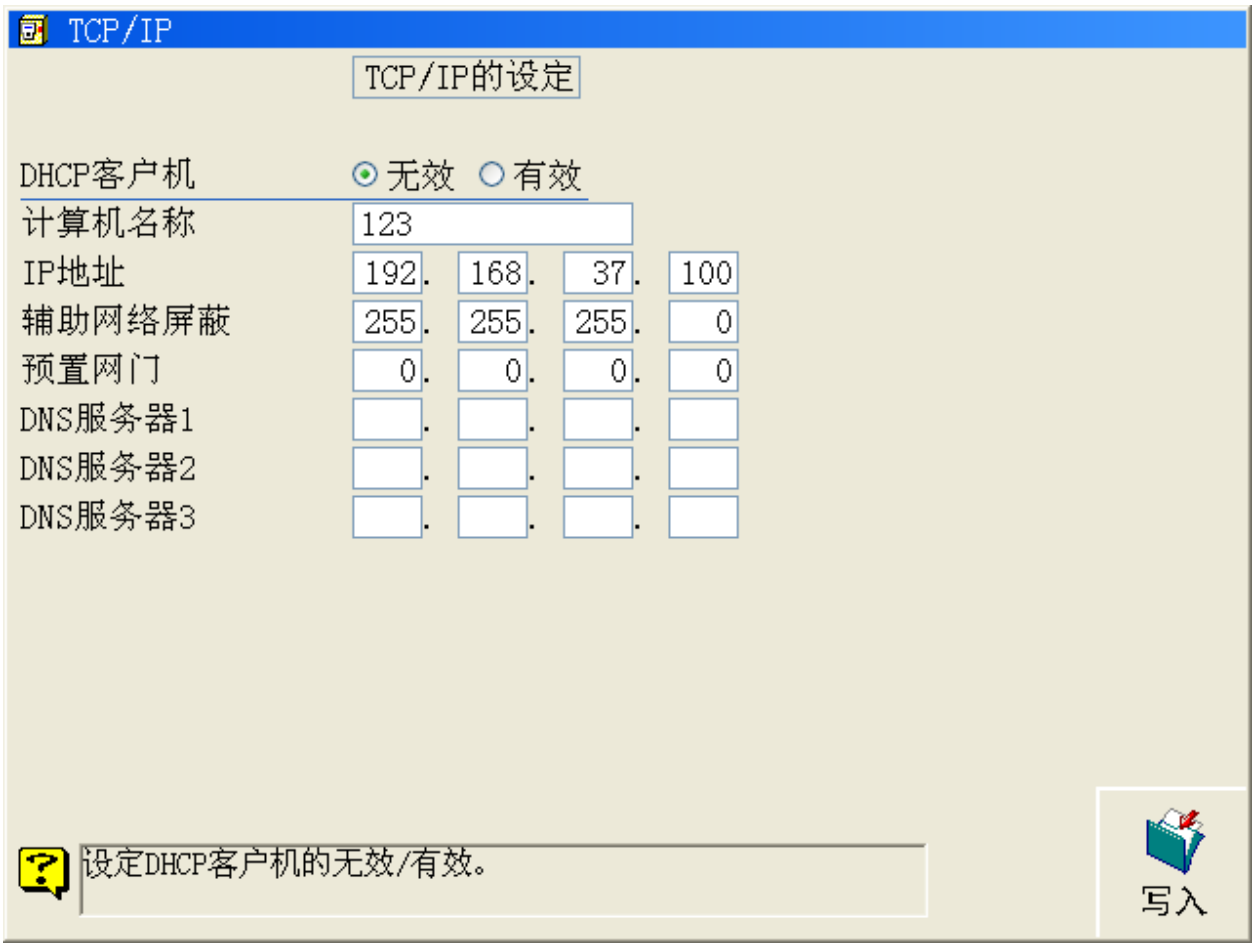
otc 机械臂程序导入有两种方法，一种是使用 FD 软件，一种是使用 U 盘。这里介绍使用 U 盘导入的方法。

程序文件准备:

1. otc 机械臂程序源文件名称中的 `MZ04-01` 是机械臂的型号，不同机械臂需要自行修改为相应的程序文件名称。
2. 确保 otc 机械臂程序文件尾行格式是 CRLF。

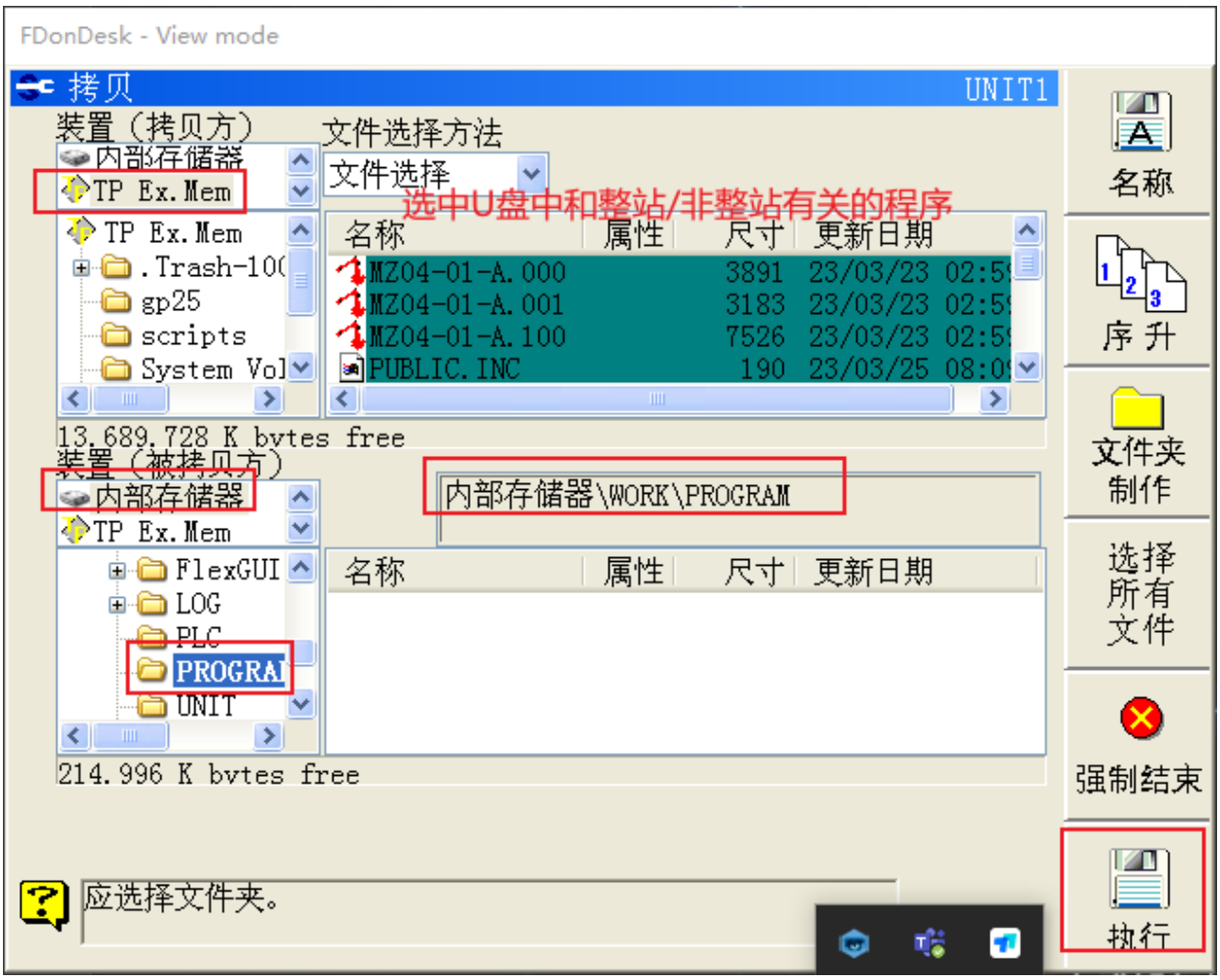
U 盘准备:

1. 确保 U 盘被格式为为 `FAT32`。
2. 将 otc 的程序文件拷贝到 U 盘根目录下，其中的 `MZ04-01-A.100` 可以导入，也可以不导入。


TCP/IP
TCP/IP的设定
DHCP客户机
无效
有效
计算机名称
123
IP地址
192. 168. 37. 100
辅助网络屏蔽
255. 255. 255. 0
预置网门
0. 0. 0. 0
DNS服务器1
DNS服务器2
DNS服务器3
设定DHCP客户机的无效/有效。
写入

导入并编译:

1. 机械臂切换到手动档，并切换用户权限
2. 将 U 盘插入到机械臂示教器 USB 接口
3. 示教器上执行: 维修-> 文件操作 -> 拷贝 -> 选择 U 盘中所有程序文件并拷贝到 /WORK/PROGRAM 下:



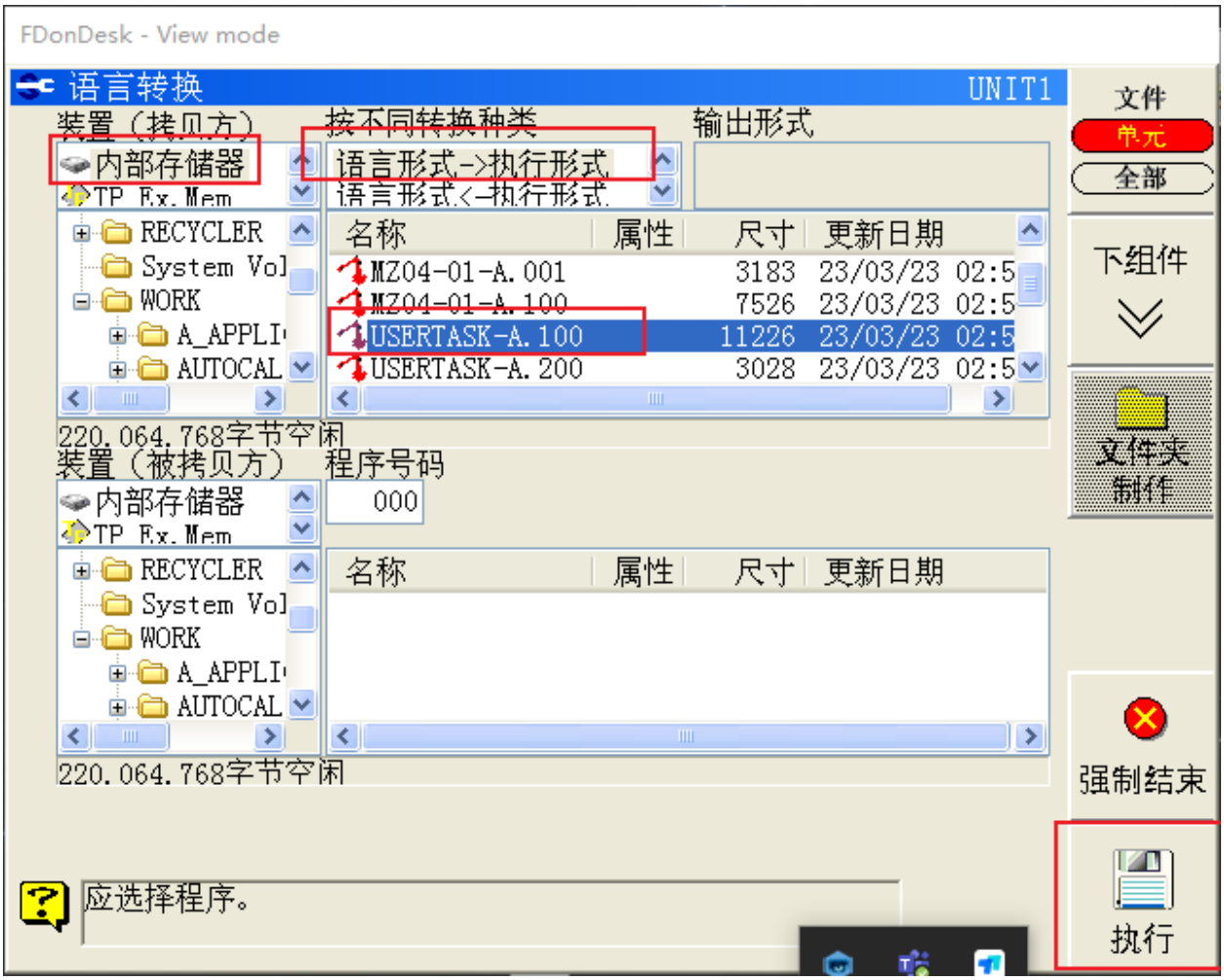
4. 重启机械臂!!! 否则后续程序编译会有问题!

   因为 PUBLIC.INC 需要重启后会才能自动编译成 PUBLIC.DAT

5. 进行程序编译: 维修 -> 程序转换 -> 语言转换 -> 先对 USERTASK-A.100 进行编译，方法如下:

   注意编译有顺序性，必须先编译后台程序

6. 同样的编译方法，按 **顺序** 对剩下的程序进行编译: USERTASK-A.200, USERTASK-A.300, MZ04-01-A.000, MZ04-01-A.001, MZ04-01-A.002, MZ04-01-A.100 (这是测试程序，可不编译)。


FDonDesk - View mode
语言转换
UNIT1
装置（拷贝方）
按不同转换种类
输出形式
内部存储器
TP Ex.Mem
语言形式->执行形式
语言形式<-执行形式
RECYCLER
System Vol
WORK
A_APPLI
AUTOCAL
名称
属性
尺寸
更新日期
MZ04-01-A.001 3183 23/03/23 02:5
MZ04-01-A.100 7526 23/03/23 02:5
USERTASK-A.100 11226 23/03/23 02:5
USERTASK-A.200 3028 23/03/23 02:5
220,064,768字节空闲
装置（被拷贝方）
程序号码
000
应选择程序。
文件
单元
全部
下组件
文件夹制作
强制结束
执行

运行程序

通讯说明

工控机和 otc 机械臂的通讯方式为 socket：工控机作为 socket server(服务端)，机械臂作为 socket client（客户端)。

启动程序

运行机械臂主控

用户根据项目实际需要，参考机械臂主控模板程序 `MZ04-01-A.001`, `MZ04-01-A.002` 等，修改生成自己的程序，然后导入到机械臂后重新编译、加载并运行。

注：也可以在示教器上直接修改程序内容。

运行步骤:

1. 切换到手动档，并切换用户权限
2. 停止后台程序，并设置后台程序的优先级别

   示教器上打开 **用户任务监视器**窗口: 维修 -> 监视器 2 (或其他监视器) -> 用户任务 -> 用户任务监视器

   如果是第一次运行，则此时看不到如上所述的后台任务 `100`, `200`, `300`，这时需要加载这 3 个后台任务，方法为:

   将光标移动到 用户任务监视器任意处，然后按示教器上的 编辑按钮，进入到编辑模式:

   然后按如下方法添加 3 个后台任务并设定优先级别为 6 -> 写入

   如果 执行状态那一列有正在启动中的后台程序，则需要先停止这些后台任务，方法为：移动光标到 **启动中**的后台程序

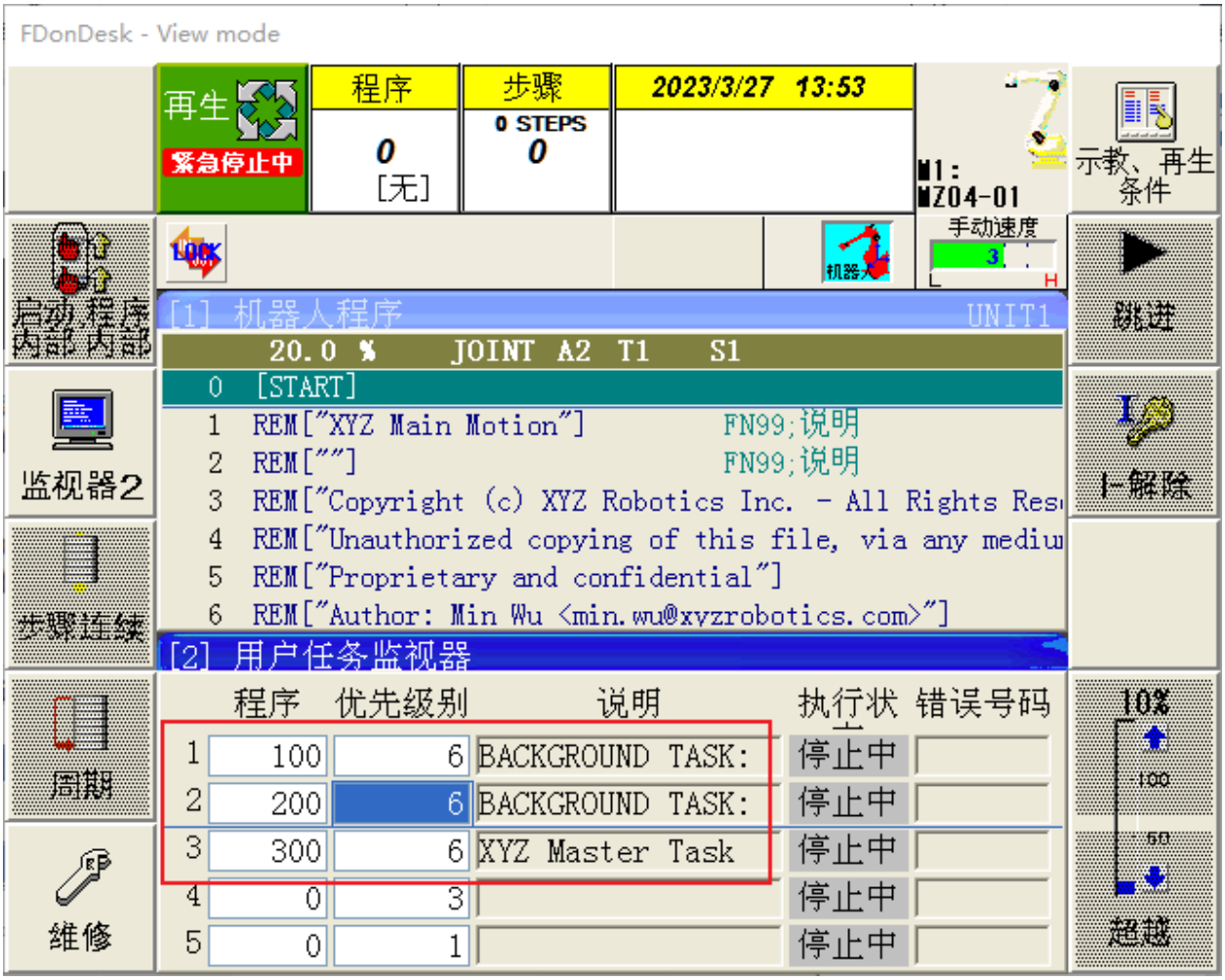
   然后按示教器的 `ENABLE` + `OFF`
3. 加载自己的机械臂程序，如 `001`:
4. 切换到自动档
5. 上使能：同时按住示教器左上方的的 `ENABLE` + `MOTOR ON`
6. 运行程序：同时按住示教器的 `ENABLE` + `SHIFT` + `GO`

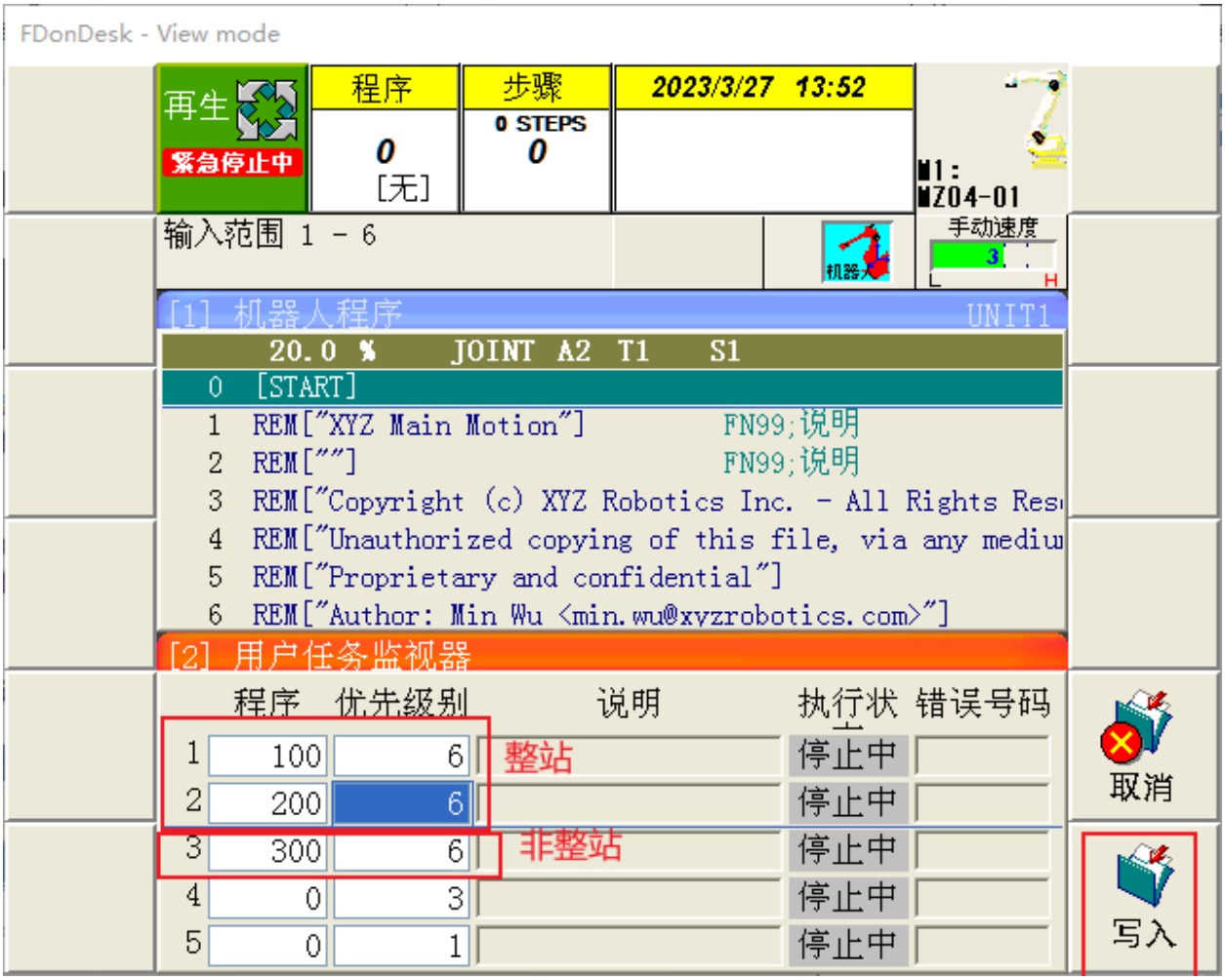
运行工控机主控

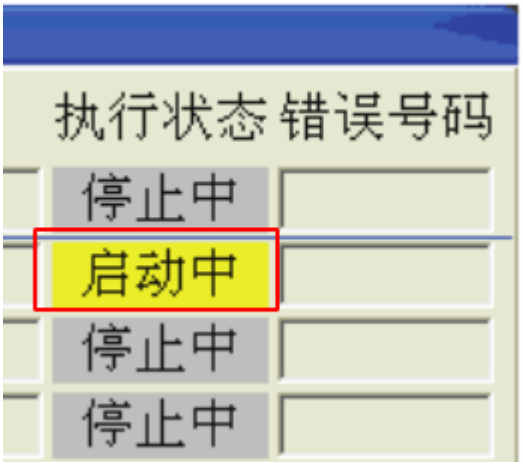
用户根据项目实际使用的机械臂修改工控机主控前台程序名 `MZ04-01-A.000` ，然后导入到机械臂后重新编译并加载并运行。

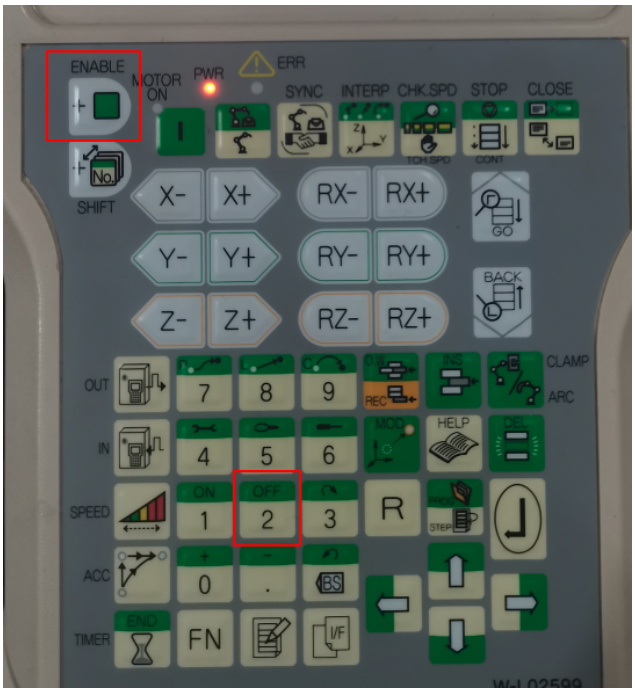
运行步骤:

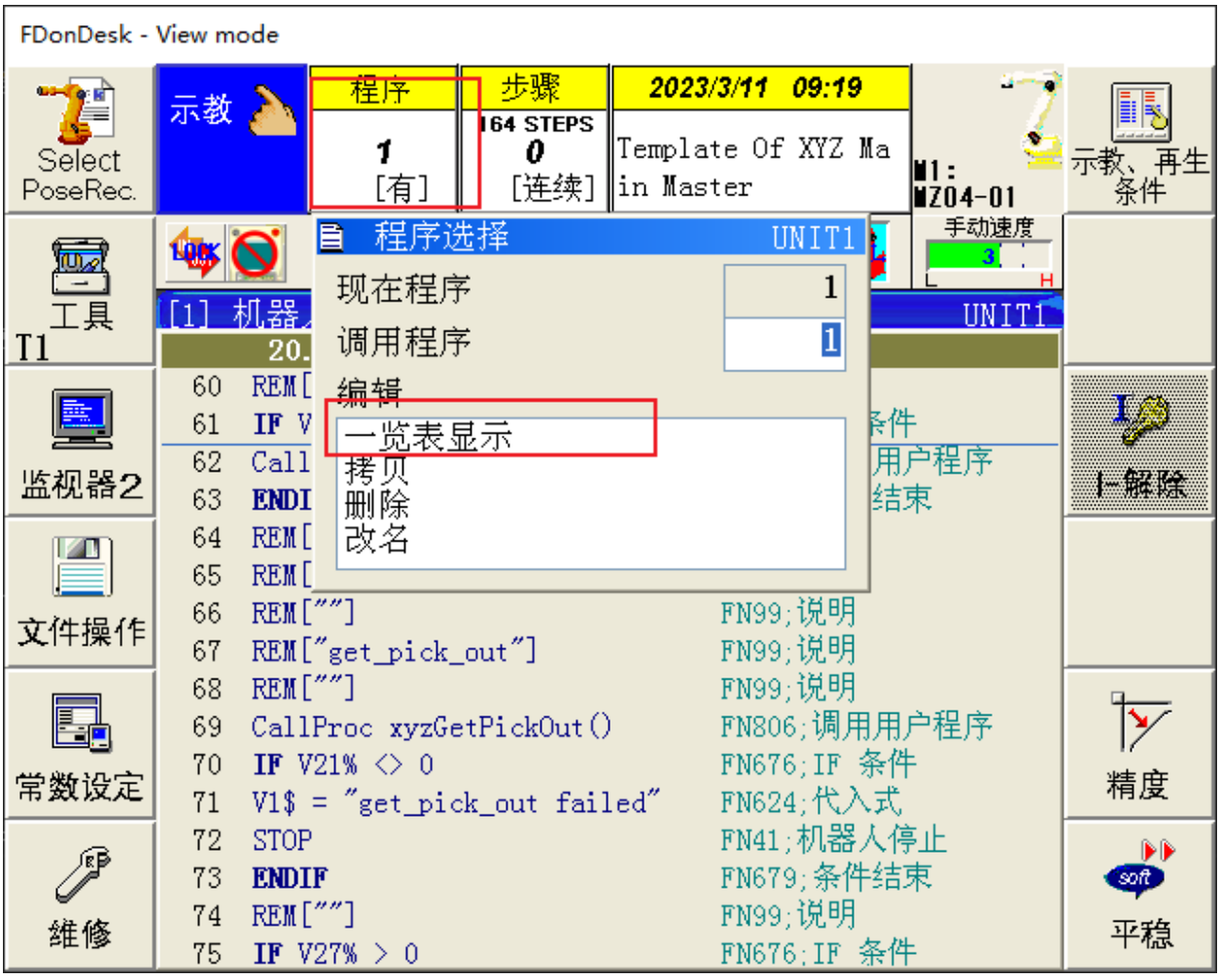
方法同机械臂主控的运行步骤，每次执行前需要确认后台程序优先级为 6 并停止正在运行的后台程序，然后前台程序为 `000`


FDonDesk - View mode
再生
紧急停止中
程序
0
[无]
步骤
0 STEPS
0
2023/3/27 13:53
M1:
MZ04-01
示教、再生
条件
手动速度
3
L
H
LOCK
机器人
启动.程序
内部.内部
跳进
监视器2
I-解除
步骤连续
周期
维修
10%
100
50
超越
[1] 机器人程序
UNIT1
20.0 % JOINT A2 T1 S1
0 [START]
1 REM["XYZ Main Motion"] FN99;说明
2 REM[""] FN99;说明
3 REM["Copyright (c) XYZ Robotics Inc. - All Rights Res
4 REM["Unauthorized copying of this file, via any mediu
5 REM["Proprietary and confidential"]
6 REM["Author: Min Wu <min.wu@xyzrobotics.com>"]
[2] 用户任务监视器
程序 优先级别 说明 执行状 错误号码
1 100 6 BACKGROUND TASK: 停止中
2 200 6 BACKGROUND TASK: 停止中
3 300 6 XYZ Master Task 停止中
4 0 3 停止中
5 0 1 停止中


FDonDesk - View mode
再生
紧急停止中
程序
0
[无]
步骤
0 STEPS
0
2023/3/27 13:52
M1:
MZ04-01
输入范围 1 - 6
机器人
手动速度
3
[1] 机器人程序
UNIT1
20.0 % JOINT A2 T1 S1
0 [START]
1 REM["XYZ Main Motion"] FN99;说明
2 REM[""] FN99;说明
3 REM["Copyright (c) XYZ Robotics Inc. - All Rights Res
4 REM["Unauthorized copying of this file, via any mediu
5 REM["Proprietary and confidential"]
6 REM["Author: Min Wu <min.wu@xyzrobotics.com>"]
[2] 用户任务监视器
程序 优先级别 说明 执行状 错误号码
1 100 6 整站 停止中
2 200 6 停止中
3 300 6 非整站 停止中
4 0 3 停止中
5 0 1 停止中
取消
写入


执行状态 错误号码
停止中
启动中
停止中
停止中


ENABLE
MOTOR ON
PWR
ERR
SYNC
INTERP
CHK SPD
STOP
CLOSE
SHIFT
X-
X+
RX-
RX+
Y-
Y+
RY-
RY+
Z-
Z+
RZ-
RZ+
GO
BACK
OUT
IN
SPEED
ACC
TIMER
CLAMP
ARC
HELP
7
8
9
4
5
6
1
2
3
R
0
FN


FDonDesk - View mode
Select PoseRec.
示教
程序
1
[有]
步骤
164 STEPS
0
[连续]
2023/3/11 09:19
Template Of XYZ Ma
in Master
M1:
MZ04-01
手动速度
3
示教、再生条件
工具
T1
监视器2
文件操作
常数设定
维修
程序选择
UNIT1
现在程序
1
调用程序
1
编辑
一览表显示
拷贝
删除
改名
[1] 机器人
UNIT1
20.
60 REM[
61 IF V
62 Call
63 ENDI
64 REM[
65 REM[
66 REM[""]
67 REM["get_pick_out"]
68 REM[""]
69 CallProc xyzGetPickOut()
70 IF V21% <> 0
71 V1$ = "get_pick_out failed"
72 STOP
73 ENDIF
74 REM[""]
75 IF V27% > 0
条件
用户程序
结束
FN99;说明
FN99;说明
FN99;说明
FN806;调用用户程序
FN676;IF 条件
FN624;代入式
FN41;机器人停止
FN679;条件结束
FN99;说明
FN676;IF 条件
I-解除
精度
soft
平稳

API 说明

otc 工控机主控支持的 API

指令号	命令	支持情况
100	获取机械臂程序版本号	支持
101	发送速度数据	支持
102	发送加速度数据	支持
103	发送圆滑过渡参数	支持
104	发送工具坐标系 (TCP) 参数	不支持
105	设置数字量输出	支持
106	SetJointsMovej	支持
107	SetCartMovel	支持
108	SetJoinsMovel	支持
109	SetCartMovej	支持
110	MovejSequence	不支持
111	MovelSequence	不支持
112	SetJointsMovejDo	不支持
113	SetCartMovelDo	不支持
114	SetJointsMovelDo	不支持
115	SetJointsMovejGroupDo	不支持
116	SetCartMovelGroupDo	不支持
117	SetJointsMovelGroupDo	不支持
118	MovelUntil	不支持
119	获取数字量输入状态	支持
120	获取数字量输出状态	支持
121	获取模拟量输入口数值	不支持
122	获取机械臂当前角度	支持
123	获取机械臂当前位姿	支持
200	机械臂后台发送状态	支持

otc 工控机主控定义了一些全局变量，不允许用户使用，占用情况如下 (`VARIABLE.INC`):

姿势变量 Pn		
地址	含义	说明 (是否用户可写)
P1	内部使用	x

整数变量 V%		
地址	含义	说明 (是否用户可写)
V50%	通讯连接标志位	read only
V51%	motion_socket 报错 1	read only
V52%	motion_socket 报错 2	read only
V53%	motion 通讯接收长度	read only
V54%	内部使用	read only
V55%	内部使用	read only
V56%	motion 通讯发送长度	read only
V57%	内部使用	read only
V58%	内部使用	read only
V60%	内部使用	read only
V61%	status_socket 报错 1	read only
V62%	status_socket 报错 2	read only
V63%	内部使用	read only
V64%	内部使用	read only
V65%	status 通讯发送长度	read only
V80%	服务端发送的 cmd	read only
V81%	服务端发送的 acc	read only
V82%	服务端发送的 zone	read only
V83%	服务端发送的 port	read only
V84%	服务端发送的 signal	read only

字符串变量 V$		
地址	含义	说明 (是否用户可写)
V10$	内部使用	x
V11$	所有收到信息	read only
V21$	内部使用	x
V22$	所有发送信息	read only
V30$	显示错误消息	read only
V31$	显示错误消息	read only

实数变量 V!		
地址	含义	说明 (是否用户可写)
V80!~V96!	内部使用	x

otc 机械臂主控支持的 API

otc 部分全局变量已经被占用，不允许用户使用，占用情况如下 (`VARIABLE.INC`):

姿势变量 Pn		
地址	含义	说明 (是否用户可写)
P1	内部使用	x

整数变量 V%		
地址	含义	说明 (是否用户可写)
V1%	通讯连接标志位	x
V2%	发送命令标志位	x
V3%	服务端 ip 地址	可写
V4%	服务端端口号	可写
V5%	是否通讯失败	read only
V6%	socket 报错 1	read only
V7%	socket 报错 2	read only
V8%	内部使用	x
V9%	内部使用	x
V10%	内部使用	x
V11%	内部使用	x
V12%	发送的命令码	read only
V13%	设置发送给服务端的 ws_id	可写
V14%	设置发送给服务端的 token	可写
V15%	设置发送给服务端的视觉服务 id	可写
V16%	设置发送给服务端的 uws_id	可写
V17%~V20%	系统预留，勿用	x
V21%	服务端返回的 error_code	read only
V22%	服务端返回的 token	read only
V23%	服务端返回的 grasp_pose_num	read only
V24%	系统预留，勿用	x
V25%	服务端返回的 obj_pose_num	read only
V26%	服务端返回的 obj_pose_type	read only
V27%	服务端返回的轨迹点数量	read only
V28%	系统预留，勿用	x
V29%~V58%	服务端返回的每个轨迹点的类型 (最多支持 30 个点)	read only
V59%	服务端返回的 pipeline_num	read only
V60%	服务端返回的 register_num	read only

字符串变量 V$		
地址	含义	说明 (是否用户可写)
V1$	显示错误消息	read only
V2$	内部使用	x
V3$	内部使用	x
V4$~V13$	内部使用	x
V14$~V15$	系统预留，勿用	x

实数变量 V!		
地址	含义	说明 (是否用户可写)
V1!	x of grasp_pose	read only
V2!	y of grasp_pose	read only
V3!	z of grasp_pose	read only
V4!	a of grasp_pose	read only
V5!	b of grasp_pose	read only
V6!	c of grasp_pose	read only
V7!	x of object_pose	read only
V8!	y of object_pose	read only
V9!	z of object_pose	read only
V10!	a of object_pose	read only
V11!	b of object_pose	read only
V12!	c of object_pose	read only
V13!	x of tote_pose	read only
V14!	y of tote_pose	read only
V15!	z of tote_pose	read only
V16!	a of tote_pose	read only
V17!	b of tote_pose	read only
V18!	c of tote_pose	read only
V19!~V26!	内部使用	x
V301!~V306!	轨迹点 1	x
V307!~V312!	轨迹点 2	x
···	轨迹点 n	x
V475!~V480!	轨迹点 30	x

机械臂主控支持的 api 定义在放在各个前台模板程序里面:

MasterInit()

初始化各变量

xyzMasterConnect()

连接服务端

参数

- **ip** 地址 (V3%) –服务器的 ip 地址，填最后 3 位即可
- 端口号 (V4%) –

xyzSwitchApp(*app_name*)

切换应用

参数 **app_name** (*STRING*) –应用的名称

返回 error_code(V21%):

xyzSwitchFlow(*flow_name*)

切换 flow

参数 **flow_name** (*STRING*) –需要切换到的 task flow 名称，以”.t“结尾，如“1.t”

返回 error_code(V21%):

xyzSwitchTool(*tool_id*)

切换工具

参数 **tool_id** (*STRING*) –工具名称

返回 error_code(V21%):

xyzReqCapImg()

请求拍照

参数 vision_service_id (V15%) –视觉服务 id

返回

error_code(V21%):

token(V22%): 用于获取对应拍照结果使用的 token

xyzGetCapImg()

获取拍照结果

参数 token (V14%) –请求拍照时返回的 token

返回 error_code(V21%):

xyzCapImg()

拍照并获取结果，等价于 xyzReqCapImg() + xyzGetCapImg()

参数 vision_service_id (V15%) –视觉服务 id

返回 error_code(V21%):

xyzReqGraspPose()

请求抓取目标点位

参数 ws_id (V13%) –工作空间 id

返回

error_code(V21%):

token(V22%): 用于获取目标点位时使用的 token

xyzGetGraspPose()

获取抓取目标点位

参数 token (V14%) –请求抓取目标点位时返回的 token

返回

error_code(V21%):

grasp_pose_num(V23%): 当前有多少个可供抓取的点

pipeline_num(V59%): pipeline 文件 number

register_num(V60%): 用到的注册文件的注册 number

grasp_pose(V1!~V6!): 抓取目标点的位姿数据

xyzReqObjPose()

请求物体位姿

参数 ws_id (V13%) –工作空间 id

返回

error_code(V21%):

token(V22%): 用于获取物体位姿时作为识别 token

xyzGetObjPose()

获取物体位姿

参数 token(V14%) –请求抓取目标点位时返回的 token

返回

error_code(V21%):

object_pose_num(V25%): 当前物体位姿个数

pose_type(V26%): 物体的位姿类型

object_pose(V7!~V12!): 物体的位姿数据

xyzResetTask()

重置任务, 一般用来初始化任务内部变量

返回 error_code(V21%):

xyzSendCurrentJoints()

发送机械臂当前角度

返回 error_code(V21%):

xyzSendCurrentCartPose()

发送机械臂当前 Cartesian 值

返回 error_code(V21%):

xyzSendCurrentExtJoints()

发送机械臂当前扩展轴位置，该指令暂不支持

返回 error_code(V21%):

xyzReqPick()

请求 pick 动作规划，该指令暂不支持

返回 error_code(V21%):

xyzReqPlace()

请求 place 动作规划，该指令暂不支持

返回 error_code(V21%):

xyzReqPickPlace()

请求 pick 和 place 规划

参数 uws_id(V16%) –目前该值填 0 即可

返回 error_code(V21%):

xyzGetPickIn()

获取取料入框轨迹

参数 uws_id(V16%) –目前该值填 0 即可

返回

error_code(V21%):

waypoint_num(V27%): 轨迹点中点位数量

pipeline_num(V59%): pipeline 文件 number

register_num(V60%): 用到的注册文件的注册 number

轨迹点 ()：在 V27% 大于 0 的情况下，直接调用 ExecTraj() 走轨迹即可

xyzGetPickOut()

获取取料出框轨迹

参数 uws_id(V16%) –目前该值填 0 即可

返回

error_code(V21%):

waypoint_num(V27%): 轨迹点中点位数量

pipeline_num(V59%): pipeline 文件 number

register_num(V60%): 用到的注册文件的注册 number

轨迹点 ()：在 V27% 大于 0 的情况下，直接调用 ExecTraj() 走轨迹即可

xyzGetPlaceIn()

获取放料入框轨迹

参数 uws_id(V16%) –目前该值填 0 即可

返回

error_code(V21%):

waypoint_num(V27%): 轨迹点中点位数量

pipeline_num(V59%): pipeline 文件 number

register_num(V60%): 用到的注册文件的注册 number

轨迹点 ()：在 V27% 大于 0 的情况下，直接调用 ExecTraj() 走轨迹即可

xyzGetPlaceOut()

获取放料出框轨迹

参数 uws_id(V16%) –目前该值填 0 即可

返回

error_code(V21%):

waypoint_num(V27%): 轨迹点中点位数量

pipeline_num(V59%): pipeline 文件 number

register_num(V60%): 用到的注册文件的注册 number

轨迹点 ()：在 V27% 大于 0 的情况下，直接调用 ExecTraj() 走轨迹即可

xyzSwitchStrat(*strat_name*)

请求切换策略

参数 strat_name(*STRING*) –策略名称

返回 error_code(V21%):

xyzUpdateTotePose()

料箱重定位

返回

error_code(V21%):

tote_pose(V13! ~ V18!)：重定位后的料箱位姿数据

xyzUpdateObjPoseOnHand()

工件在上手的二次定位

返回 error_code(V21%):

xyzUpdateObjPoseToHand()

工件不在手上的二次定位

返回

error_code(V21%):

waypoint_num(V27%): 轨迹点中点位数量

pipeline_num(V59%): pipeline 文件 number

register_num(V60%): 用到的注册文件的注册 number

轨迹点 ()：在 V27% 大于 0 的情况下，直接调用 ExecTraj() 走轨迹即可

xyzGetObjPoseType()

获取工件姿态类型

返回

error_code(V21%):

pose_type(V26%): 工件的 pose 类型信息

xyzResetPalletStatus()

重置工业码垛状态

返回 error_code(V21%):

xyzSwitchObj(*obj_name*)

切换工件

参数

- **ws_id** (V13%) –工作空间 id
- **obj_name** (*STRING*) –工件名称

返回 error_code(V21%):

xyzCalculateGraspPose()

计算抓取目标点位，该指令等价于 xyzReqGraspPose() + xyzGetGraspPose()

参数 ws_id (V13%) –工作空间 id

返回

error_code(V21%):

grasp_pose_num(V23%): 当前有多少个可供抓取的点

pipeline_num(V59%): pipeline 文件 number

register_num(V60%): 用到的注册文件的注册 number

grasp_pose(V1!~V6!)：抓取目标点的位姿数据

xyzCalculateObjectPose()

计算物体位姿，该指令等价于 xyzReqObjPose() + xyzGetObjPose()

参数 ws_id (V13%) –工作空间 id

返回

error_code(V21%):

object_pose_num(V25%): 当前物体位姿个数

pose_type(V26%): 物体的位姿类型

object_pose(V7!~V12!): 物体的位姿数据

案例/模板说明

用户需要根据项目情况，编写机械臂前台主程序。

otc 的一部分寄存器已经被占用，用户不允许使用这些寄存器，详见*API 说明*。

CartMove 模板

以下为机械臂主控 CartMove 模板程序，注意对工控机返回的 error_code 进行判断。

用户需要根据项目情况，编写自己的机械臂前台主程序。

CartMove 基础模板：MZ04-01-A.001

CartMove 二次定位模板：MZ04-01-A.002

TrajMove 模板

```
INCLUDE "VARIABLE"

' 初始化并启动后台程序
CallProc MasterInit()
KILLMCR 300, 0, 10000
DELAY 2
FORKMCR 300, 10000
DELAY 2

'
' 连接服务端
'
' 设置服务端ip地址(ip地址的最后一段)
V3% = 101
' 设置服务端端口
V4% = 11111
CallProc xyzMasterConnect()

'
' 切换app
'
' 设置项目名称
L1$ = "111"
CallProc xyzSwitchApp(L1$)
' 需要对返回的error_code(V21%)进行判断，以下每条语句同理
IF V21% <> 0
    STOP
ENDIF
```

(下页继续)

(续上页)

```
*MAIN_LOOP

'
' 请求pick
'
CallProc xyzReqPick()
IF V21% <> 0
    STOP
ENDIF

'
' 获取pick_in轨迹
'
CallProc xyzGetPickIn()
IF V21% <> 0
    STOP
ENDIF

' 轨迹点数量>0时运行轨迹
IF V27% > 0
    CallProc ExecTraj()
ENDIF


'
' 获取pick_out轨迹
'
CallProc xyzGetPickOut()
IF V21% <> 0
    STOP
ENDIF

' 轨迹点数量>0时运行轨迹
IF V27% > 0
    CallProc ExecTraj()
ENDIF

'
' 获取place_in轨迹
'
CallProc xyzGetPlaceIn()
IF V21% <> 0
    STOP
ENDIF

' 轨迹点数量>0时运行轨迹
IF V27% > 0
    CallProc ExecTraj()
ENDIF


'
' 获取pick_out轨迹
'
CallProc xyzGetPlaceOut()
```

(下页继续)

(续上页)

```
IF V21% <> 0
    STOP
ENDIF

' 轨迹点数量>0时运行轨迹
IF V27% > 0
    CallProc ExecTraj()
ENDIF

GOTO *MAIN_LOOP



END



'␣
↪*****************************************************************************************************
' SUB FUNCTIONS
'␣
↪*****************************************************************************************************

UsrProc ExecTraj()
    '
    ' set your own speed/override... here
    ' 如果需要分别调整出筐/入筐等的轨迹、圆滑值，可以在这里自定义调整
    '
    ' IF V12% = 518
    '     ' set pick_in speed/override...
    ' ELSEIF V12% = 519
    '     ' set pick_out speed/override...
    ' ELSEIF V12% = 520
    '     ' set place_in speed/override...
    ' ELSEIF V12% = 521
    '     ' set place_out speed/override...
    ' ELSEIF V12% = 525
    '     ' set update_obj_pose_to_hand speed/override...
    ' ELSE
    '     STOP
    ' ENDIF


    ' V27%: traj pose num
    L100% = V27% - 1
    FOR L101% = 0 TO L100% STEP 1
        V19! = V![301+6*L101%]
        V20! = V![302+6*L101%]
        V21! = V![303+6*L101%]
        V22! = V![304+6*L101%]
        V23! = V![305+6*L101%]
        V24! = V![306+6*L101%]
        IF V%[29+L101%] = 11
            ' JOINTS_MOVEJ
            MOVEX A=3, AC=0, SM=0, M1J, P, (V19!,V20!,V21!,V22!,V23!,V24!),R=100.0,MS
        ELSEIF V%[29+L101%] = 12
```

(下页继续)

(续上页)

```
            ' JOINTS_MOVEL
            MOVEX A=3, AC=0, SM=0, M1J, L, (V19!,V20!,V21!,V22!,V23!,V24!),R=100.0,MS
        ELSEIF V%[29+L101%] = 21
            ' CART_MOVEJ
            MOVEX A=3, AC=0, SM=0, M1X, P, (V19!,V20!,V21!,V22!,V23!,V24!),R=100.0,MS
        ELSEIF V%[29+L101%] = 22
            ' CART_MOVEL
            MOVEX A=3, AC=0, SM=0, M1X, L, (V19!,V20!,V21!,V22!,V23!,V24!),R=100.0,MS
        ELSE
            STOP
        ENDIF
    NEXT
EndProc

...
...
```

常见问题

1. 机械臂程序编译出错

   确保导入到机械臂中的源程序文件是 CRLF 结尾。上传 `PUBLIC.INC` 以后，需要重启机械臂才能生效。编译机械臂主控程序具有顺序性，需要先编译后台程序。

2. 工控机主控手动运行时 socket 报错

   otc 机械臂的问题，需要打到自动档进行测试。

附录

14.2.16 Rokae

此处介绍安装 rokae 机械臂驱动的相关事项。

驱动版本和使用要求

支持的机械臂类型

Rokae 六轴机械臂

控制器型号

Xcore

机械臂需要开通的功能

Rokae 机械臂控制器版本要保证为 1.7.4

安装驱动

rokae 机械臂驱动文件列表

rokae 机械臂的用于烧录的驱动文件只有一个压缩文件：xyz.zip

获取权限

在对机器人做出任何修改时会遇到权限不足的情况，按以下步骤获取权限

1. 点击图中的设置按钮，选择 设置 -> 用户组，然后选择用户组为 God，输入默认密码 123456，即可。

设定机械臂 IP

示教器右下角点选机器人名称，进入控制器设置界面，向下拖动找到系统 IP 属性，设置为：

- “IP”设置为：192.168.37.100
- “子网掩码”设置为：255.255.255.0
- “网关”设置为：192.168.37.1

网络设置完成后重启机械臂生效。

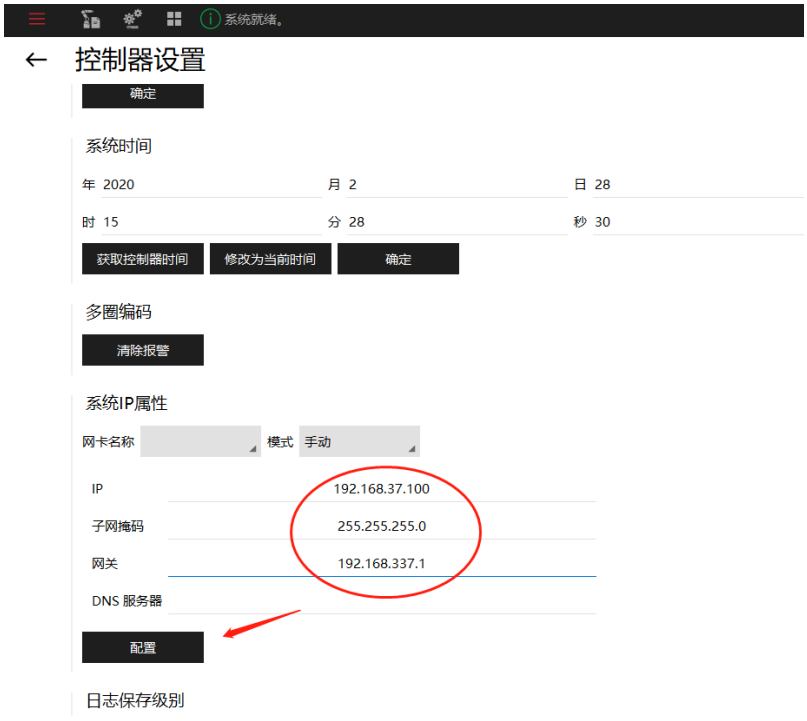


图 86: ROKAE 设定机械臂 IP

使用 U 盘导入程序

1. 将 MAX 安装目录下的 share/robot_code 中 rokae 的机械臂程序文件 `xyz.zip` 拷入 U 盘，插入示教器右上方的 USB 口。
2. 在示教器上，按照图片选择，点击圆圈中的图标，再点击当前的工程名称，即可进入工程配置界面，见下图。

图 87: 配置工程

3. 在示教器上，点击上图中的箭头指向的按钮，即可进入烧录程序界面，在箭头指向的地方选择 U 盘中存放程序的路径, 即可导入。

图 88: 烧录程序

4. 回到配置工程界面后选取刚刚导入的 xyz 工程，点击重新加载，即可将当前工程设置为烧录的程序。

5. 至此，烧录程序完成。

运行程序

通讯说明

工控机和 Rokae 机械臂的通讯方式为 TCP/IP： 工控机作为 TCP/IP 的服务端，机械臂作为 TCP/IP 的客户端。

启动程序

运行前，需要检查当前的控制器版本。

1. 点击示教器左上角的三条横线，选择 帮助-> 关于珞石
2. 即可查看当前控制器版本，需要为 1.7.4。

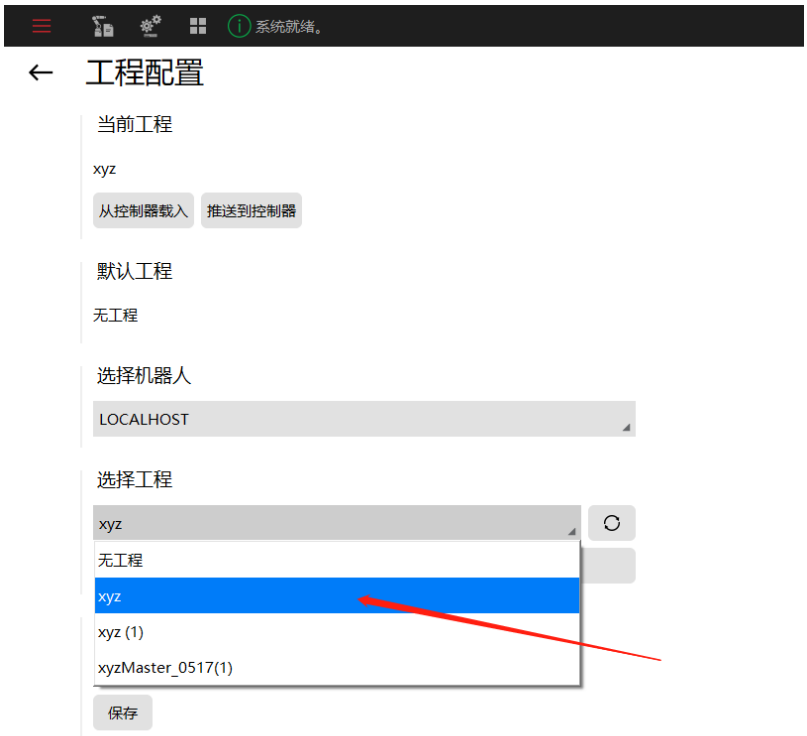
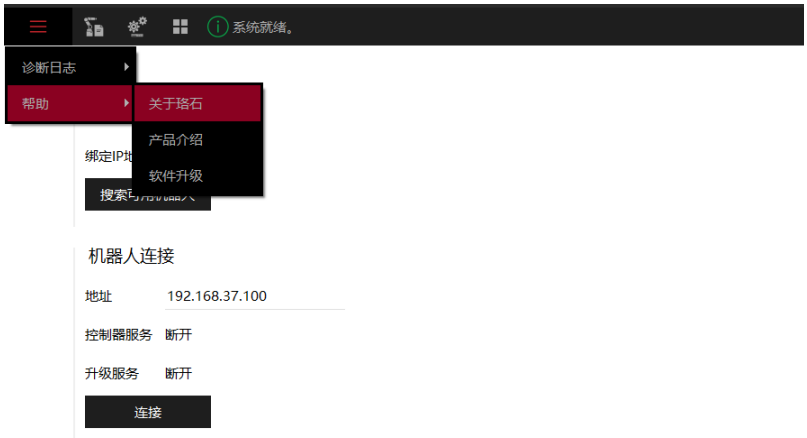


图 89: 选择工程

图 90: 加载工程



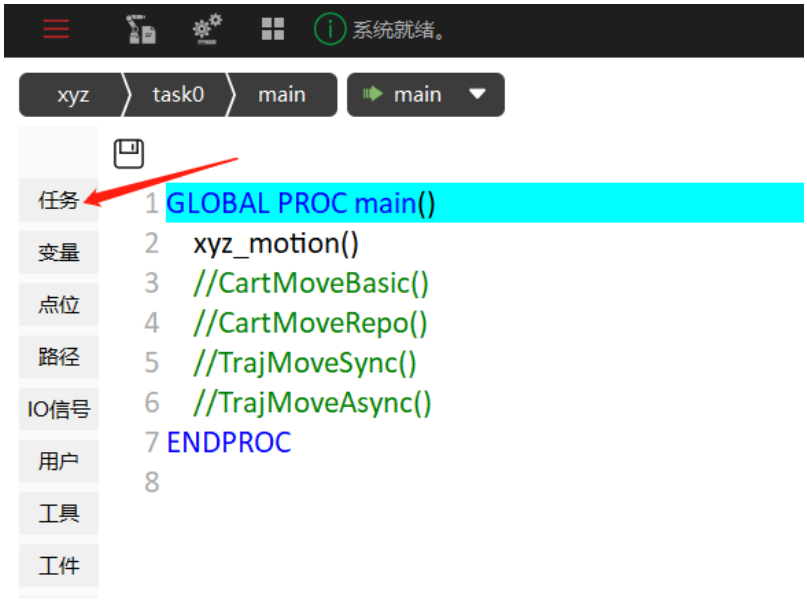
运行工控机主控

1. 下图为烧录完程序的工程界面，点击左侧的 任务图标。
2. 将 `task0` 与 `task1` 的 是否运行这一选项都勾选上。
3. 回到上一界面，取消注释需要运行的 `xyz_motion()` ，再点击图中的程序指针移至 main。
4. 点击示教器最下方的上电按钮，再点击示教器右侧的运行按钮，即可运行程序。

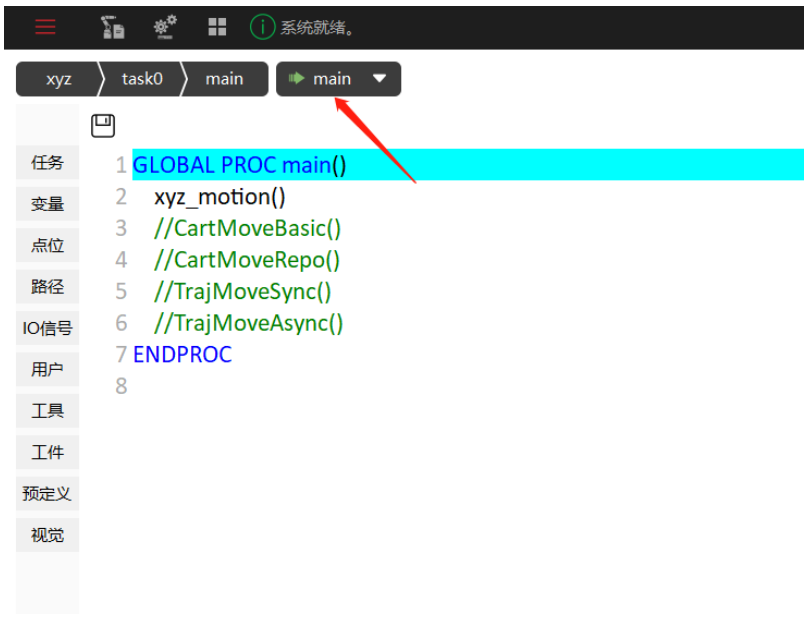
运行机械臂主控

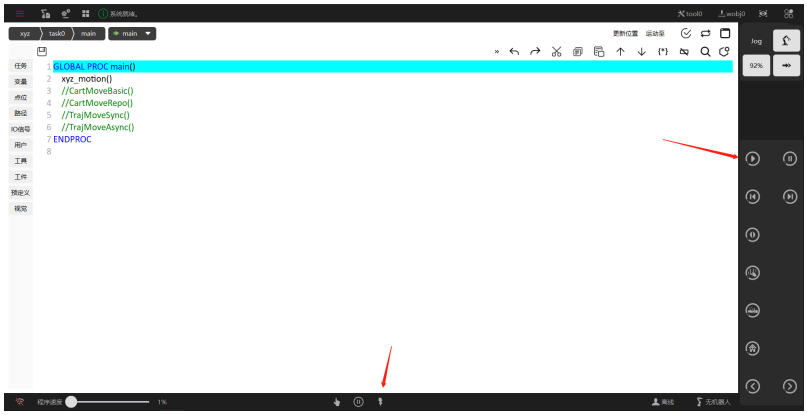
同工控机主控程序运行方法，只有两处需要修改：

1. 需要将下图中的 `task1` 的 是否运行取消勾选。
2. 根据实际项目需求，选用合适的模板，将下图中 `main` 程序适当程序取消注释，例如图中将 `CartMoveBasic` 取消注释。

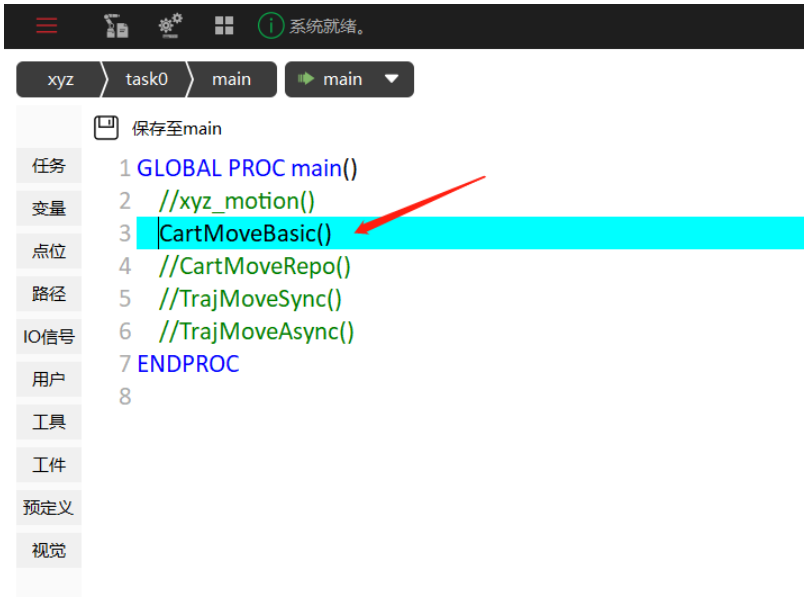

系统就绪。
xyz
task0
main
main
任务
变量
点位
路径
IO信号
用户
工具
工件
1 GLOBAL PROC main()
2 xyz_motion()
3 //CartMoveBasic()
4 //CartMoveRepo()
5 //TrajMoveSync()
6 //TrajMoveAsync()
7 ENDPROC
8


系统就绪。
任务列表
名称
类型
是否运行
优先级
task0
运动任务
高
task1
常规任务
低


系统就绪。
xyz
task0
main
main
任务
变量
点位
路径
IO信号
用户
工具
工件
预定义
视觉
1 GLOBAL PROC main()
2 xyz_motion()
3 //CartMoveBasic()
4 //CartMoveRepo()
5 //TrajMoveSync()
6 //TrajMoveAsync()
7 ENDPROC
8




系统就绪。
任务列表
名称
类型
是否运行
优先级
task0
运动任务
高
task1
常规任务
低


系统就绪。
xyz
task0
main
main
保存至main
任务
变量
点位
路径
IO信号
用户
工具
工件
预定义
视觉
1 GLOBAL PROC main()
2 //xyz_motion()
3 CartMoveBasic()
4 //CartMoveRepo()
5 //TrajMoveSync()
6 //TrajMoveAsync()
7 ENDPROC
8

API 说明

rokae 工控机主控支持的 API

指令号	命令	支持情况
100	获取机械臂程序版本号	支持
101	发送速度数据	支持
102	发送加速度数据	支持
103	发送圆滑过渡参数	支持
104	发送工具坐标系 (TCP) 参数	支持
105	设置数字量输出	支持
106	SetJointsMovej	支持
107	SetCartMovel	支持
108	SetJoinsMovel	支持
109	SetCartMovej	支持
110	MovejSequence	支持
111	MovelSequence	支持
112	SetJointsMovejDo	不支持
113	SetCartMovelDo	不支持
114	SetJointsMovelDo	不支持
115	SetJointsMovejGroupDo	不支持
116	SetCartMovelGroupDo	不支持
117	SetJointsMovelGroupDo	不支持
118	MovelUntil	不支持
119	获取数字量输入状态	不支持
120	获取数字量输出状态	不支持
121	获取模拟量输入口数值	不支持
122	获取机械臂当前角度	支持
123	获取机械臂当前位姿	支持
200	机械臂后台发送状态	支持

rokae 机械臂主控支持的 API

PROC xyzConnect(string sock)

连接指定 socket

参数 sock (*VAR string*) –socket 名称，定义在全局变量中

PROC xyzClose(string sock)

关闭指定 socket

参数 sock (*VAR string*) –socket 名称，定义在全局变量中

FUNC int xyzSwitchApp(string app_name)

切换应用

参数 app_name (*string*) –应用名称

返回 err_code

返回类型 int

FUNC int xyzSwitchFlow(string flow_name)

切换 flow

参数 **flow_name** (*string*) –flow 名称

返回 err_code

返回类型 int

FUNC int xyzSwitchItem(int ws_id, string item_codename)

切换工件

参数

- **ws_id** (*int*) –工作空间 id
- **item_codename** (*string*) –工件名称

返回 err_code

返回类型 int

FUNC int xyzSwitchTool(string tool_name)

切换工具

参数 **tool_name** (*string*) –工具名称

返回 err_code

返回类型 int

FUNC int xyzReqCapImg(int ws_id)

请求拍照

参数

- **ws_id** (*int*) –工作空间 id
- **Token** (*int*) –请求拍照结果, 全局变量

返回 err_code

返回类型 int

FUNC int xyzGetCapImg(int token)

获取拍照结果

参数 **token** (*int*) –请求拍照时返回的 token

返回 err_code

返回类型 int

FUNC int xyzCapImg(int ws_id)

拍照

参数 **ws_id** (*int*) –需要进行拍照操作的工作空间 id

返回 err_code

返回类型 int

FUNC int xyzReqGraspPose(int ws_id)

请求抓取位姿

参数

- **ws_id** (*INintT*) –需要获取抓取点位的工作空间 id
- **Token** –返回的用于获取目标点位时使用的 Token, 全局变量

返回 err_code

返回类型 int

FUNC int xyzGetGraspPose(int token)

获取抓取位姿

参数

- **token** (*int*) –求抓取目标点位时返回的 token
- **PoseNum** (*int*) –可供抓取的点数量
- **PipeNum** (*int*) –pipeline 编号
- **RegNum** (*int*) –注册编号
- **GraspPose** (*pose*) –抓取位姿

返回 err_code

返回类型 int

FUNC int xyzReqObjPose(int ws_id)

请求物体位姿

参数

- **ws_id** (*int*) –需要获取物体位姿的工作空间 id
- **Token** (*int*) –物体位姿识别的 token

返回 err_code

返回类型 int

FUNC int xyzGetObjPose(int token)

获取物体位姿

参数

- **token** (*int*) –请求物体位姿时得到的 token
- **PoseNum** (*int*) –可供抓取的点数量
- **PipeNum** (*int*) –pipeline 编号
- **RegNum** (*int*) –注册编号
- **ObjectPose** (*pose*) –抓取位姿

返回 err_code

返回类型 int

FUNC int xyzResetVision(int ws_id)

重置视觉

参数 **ws_id** (*int*) –需要重置视觉的工作空间 id

返回 err_code

返回类型 int

FUNC int xyzSendCurrentJoints()

发送当前机器人角度

返回 err_code

返回类型 int

FUNC int xyzSendCurrentCartPose()

发送当前机器人位姿

返回 err_code

返回类型 int

FUNC int xyzSendCurrentExtJoints()

发送当前机器人外部角

返回 err_code

返回类型 int

FUNC int xyzReqPick()

请求 pick 动作规划

返回 err_code

返回类型 int

FUNC int xyzReqPlace()

请求 place 动作规划

返回 err_code

返回类型 int

FUNC int xyzReqPickPlace(int ws_id)

请求 pick 和 place 规划

参数 **ws_id** (*int*) –工作空间 id

返回 err_code

返回类型 int

FUNC int xyzGetPickin(int ws_id)

获取取料入框轨迹

参数

- **PoseNum** (*int*) –可供抓取的点数量
- **PipeNum** (*int*) –pipeline 编号
- **RegNum** (*int*) –注册编号
- **WaypointType** (*int array*) –轨迹点类型
- **JointWaypoints** (*joints array*) –joints 类型轨迹点
- **CartWaypoints** (*cart array*) –cart 类型轨迹点

返回 err_code

返回类型 int

FUNC int xyzGetPickout(int ws_id)

获取取料出框轨迹

参数

- **PoseNum** (*int*) –可供抓取的点数量

- **PipeNum** (*int*) –pipeline 编号
- **RegNum** (*int*) –注册编号
- **WaypointType** (*int array*) –轨迹点类型
- **JointWaypoints** (*joints array*) –joints 类型轨迹点
- **CartWaypoints** (*cart array*) –cart 类型轨迹点

返回 err_code

返回类型 int

FUNC int xyzGetPlacein(int ws_id)

获取放料入框轨迹

参数

- **PoseNum** (*int*) –可供抓取的点数量
- **PipeNum** (*int*) –pipeline 编号
- **RegNum** (*int*) –注册编号
- **WaypointType** (*int array*) –轨迹点类型
- **JointWaypoints** (*joints array*) –joints 类型轨迹点
- **CartWaypoints** (*cart array*) –cart 类型轨迹点

返回 err_code

返回类型 int

FUNC int xyzGetPlaceout(int ws_id)

获取放料出框轨迹

参数

- **PoseNum** (*int*) –可供抓取的点数量
- **PipeNum** (*int*) –pipeline 编号
- **RegNum** (*int*) –注册编号
- **WaypointType** (*int array*) –轨迹点类型
- **JointWaypoints** (*joints array*) –joints 类型轨迹点
- **CartWaypoints** (*cart array*) –cart 类型轨迹点

返回 err_code

返回类型 int

FUNC int xyzSwitchStrat(string strat_name)

请求切换策略

参数 strat_name (*string*) –策略名称

返回 err_code

返回类型 int

FUNC int xyzUpdateTotePose()

料箱重定位

参数 TotePose (*pose*) –料箱位姿

返回 err_code

返回类型 int

FUNC int xyzUpdateObjPoseOnHand()

工件在上手的二次定位

返回 err_code

返回类型 int

FUNC int xyzUpdateObjPoseToHand()

工件不在手上的二次定位

参数

- **PoseNum** (*int*) –可供抓取的点数量
- **PipeNum** (*int*) –pipeline 编号
- **RegNum** (*int*) –注册编号
- **WaypointType** (*int array*) –轨迹点类型
- **JointWaypoints** (*joints array*) –joints 类型轨迹点
- **CartWaypoints** (*cart array*) –cart 类型轨迹点

返回 err_code

返回类型 int

FUNC int xyzGetObjPoseType()

获取工件姿态类型

参数 PoseType (*int*) –工件姿态类型

返回 err_code

返回类型 int

FUNC int xyzResetPalletStatus()

重置工业码垛状态

返回 err_code

返回类型 int

FUNC int xyzCalculateGraspPose(int ws_id)

计算抓取位姿

参数

- **ws_id** (*int*) –工作空间 id
- **PoseNum** (*int*) –可供抓取的点数量
- **PipeNum** (*int*) –pipeline 编号
- **RegNum** (*int*) –注册编号
- **GraspPose** (*pose*) –抓取位姿

返回 err_code

返回类型 int

FUNC int xyzCalculateObjectPose(int ws_id)

计算物体位姿

参数

- **ws_id** (*int*) –工作空间 id
- **PoseNum** (*int*) –可供抓取的点数量
- **PipeNum** (*int*) –pipeline 编号
- **RegNum** (*int*) –注册编号
- **ObjectPose** (*pose*) –物体位姿

返回 err_code

返回类型 int

案例/模板说明

机械臂主控主函数说明

以下为机械臂主控模板代码，包含坐标移动基础模板, 坐标移动二次定位模板，轨迹移动同步模板以及轨迹移动异步模板。关于模板中的 API 可以查阅 ROKAE 的“API 说明”部分。

请注意模板函数并不能直接运行，请一定根据项目现场环境和流程需求进行修改。建议使用 RobotStudio 来修改。

坐标移动基础模板

```
GLOBAL PROC CartMoveBasic()
    VAR int err_code;           // 错误代码
    VAR int flag                // 0 表示眼在手上，1表示眼在手外
    VAR jointtarget home_pose   // home 点
    VAR robtarget scan_pose     // 眼在手上时，需要发送的拍照位
    VAR robtarget place_pose    // 放置位姿

    xyzClose("MasterSock")   // 断开可能已存在的socket
    xyzConnect("MasterSock") // 连接上位机 server

    // err_code = xyzSwitchFlow("test.t")     // 切换 flow␣
↪, 默认被注释掉，如果需求可以取消注释
    // if (err_code != 0)                     // 异常处理
    //     xyzClose("MasterSock")
    // endif

    // 切换工件，两个参数分别表示视觉服务ID，和工件代号
    // 工件代号为MAX中映射表与通讯协议设置中的参数
    err_code = xyzSwitchObject(0, "obj1")
    if (err_code != 0)                    // 异常处理
        xyzClose("MasterSock")
    endif
```

(下页继续)

(续上页)

```
    SetDO DO0_0, false // 重置数字输出

    // 首先移动到 home 点。注意需要先定义 home_pose 的关节角度值
    // 或者用使用其他点来替代
    MoveAbsJ home_pose,v1000,z50,tool0

    flag = 1    // 修改手眼关系标志位（是否需要发送拍照点位姿）
    While (true)
                if (flag==0)
            // 注意需要先定义 scan_pose 的位姿,或者用使用其他点来替代
            MoveL scan_pose,v1000,z50,tool0 // 移动到 scan pose
            err_code = xyzSendCurrentCartPose()  // 发送拍照位姿
            if (err_code != 0)                   // 异常处理
                xyzClose("MasterSock")
            endif
        endif

        err_code = xyzReqGraspPose(0)   // 请求 grasp pose
        if (err_code != 0)              // 异常处理
            xyzClose("MasterSock")
        endif

        // 获取grasp pose
        err_code = xyzGetGraspPose(Token)
        if (err_code != 0)              // 异常处理
            xyzClose("MasterSock")
        endif
        if (PoseNum < 1)    // 识别失败处理
            Wait 5
                        continue
            endif

        // 可能需要添加其他路径点作为过渡

        // 移动到抓取位姿
        MoveL GraspPose,v1000,z50,tool0
        SetDO DO0_0, true   // 控制数字输出

        // 需要添加其他路径点作为过渡

        // 移动到放置点
        MoveL place_pose,v1000,z50,tool0
        SetDO DO0_0, false  // 重置数字输出
    Endwhile
ENDPROC
```

坐标移动二次定位模板

```
GLOBAL PROC CartMoveRepo()
    VAR int err_code             //错误代码
    VAR jointtarget home_pose    // home 点
    VAR robtarget scan_pose      // 拍照位姿
    VAR robtarget place_pose     // 放置位姿

    xyzClose("MasterSock")   // 断开可能已存在的socket
    xyzConnect("MasterSock") // 连接上位机 server

    // err_code = xyzSwitchFlow("test.t")     // 切换 flow␣
↪, 默认被注释掉, 如果需求可以取消注释
    // if (err_code != 0)                     // 异常处理
    //     xyzClose("MasterSock")
    // endif

    // 切换工件, 两个参数分别表示视觉服务ID, 和工件代号
    // 工件代号为MAX中映射表与通讯协议设置中的参数
    err_code = xyzSwitchObject(0, "obj1")
    if (err_code != 0)                   // 异常处理
        xyzClose("MasterSock")
    endif

    SetDO DO0_0, false // 重置数字输出
        SetDO DO0_1, false

    // 首先移动到 home 点。注意需要先定义 home_pose 的关节角度值,␣
↪或者用使用其他点来替代
    MoveAbsJ home_pose,v1000,z50,tool0
    err_code = xyzReqGraspPose(0)    // 请求 grasp pose
    if (err_code != 0)               // 异常处理
        xyzClose("MasterSock")
    endif

    While (true)
        // 获取抓取位姿
        err_code = xyzGetGraspPose(Token)
        if (err_code != 0)               // 异常处理
            xyzClose("MasterSock")
        endif
        if (PoseNum < 1)                                // 如果识别数量为0
            // 工作空间内没有工件, 需要先移除隔板
            err_code = xyzSwitchObject(0, "board") // 切换隔板
            if (err_code != 0)               // 异常处理
                xyzClose("MasterSock")
            endif

            err_code = xyzReqGraspPose(0);       // 请求隔板的抓取位姿
            if (err_code != 0)               // 异常处理
                xyzClose("MasterSock")
            endif

            // 获取隔板抓取位姿
            err_code = xyzGetGraspPose(Token)
            if (err_code != 0) OR (pose_num < 1)  // 异常处理与未识别处理
                xyzClose("MasterSock")
```

(下页继续)

(续上页)

```
        endif

        // 可能需要添加其他路径点作为过渡

        // 移动到抓取位姿
        MoveL GraspPose,v1000,z50,tool0
        SetDO DO0_0, true   // 控制数字输出

        // 此处需要添加放置隔板的路径点

        err_code = xyzSwitchObject(0, "obj1")    // 切换工件
        if (err_code != 0)              // 异常处理
            xyzClose("MasterSock")
        endif

        err_code = xyzReqGraspPose(0)   // 请求抓取位姿
        if (err_code != 0)              // 异常处理
            xyzClose("MasterSock")
        endif
        continue
    else    // 如果识别数量不为0
        // 注意需要先定义 scan_pose 的位姿，或者用使用其他点来替代
        MoveL scan_pose,v1000,z50,tool0   // 运动到 scan pose
                    err_code = xyzSendCurrentCartPose()    // 发送 scan pose
        if (err_code != 0)              // 异常处理
            xyzClose("MasterSock")
        endif

        err_code = xyzSwitchObject(1, "obj1") // 切换工作空间与工件
        if (err_code != 0)              // 异常处理
            xyzClose("MasterSock")
        endif

        err_code = xyzReqGraspPose(1)  // 请求抓取位姿
        if (err_code != 0)              // 异常处理
            xyzClose("MasterSock")
        endif

        err_code = xyzGetGraspPose(Token) // 获取抓取位姿
        if (err_code != 0) OR (pose_num < 1)  // 异常处理与未识别处理
            xyzClose("MasterSock")
        endif

        // 可能需要添加其他路径点作为过渡

        // 移动到抓取位姿
        MoveL GraspPose,v1000,z50,tool0
                    SetDO DO0_1,true   // 控制数字输出

        // 可能需要添加其他路径点作为过渡

        // 移动到放置位姿
        MoveL place_pose,v1000,z50,tool0
        SetDO DO0_1,false   // 重置数字输出

    ENDIF
```

(下页继续)

(续上页)

```
        err_code = xyzSwitchObject(0, "obj1")  // 切换工作空间与工件
        if (err_code != 0)                  // 异常处理
            xyzClose("MasterSock")
        endif

        err_code = xyzReqGraspPose(0)    // 请求抓取位姿
        if (err_code != 0)                  // 异常处理
            xyzClose("MasterSock")
        endif
    Endwhile
ENDPROC
```

轨迹移动同步模板

```
GLOBAL PROC TrajMoveSync()
    VAR int err_code           // 错误代码
    VAR jointtarget home_pose  // home 点位

    xyzClose("MasterSock")   // 断开可能已存在的socket
    xyzConnect("MasterSock") // 连接上位机 server

    // err_code = xyzSwitchFlow("traj_sync.t")    // 切换 flow␣
↪, 默认被注释掉, 如果需求可以取消注释
    // if (err_code != 0)                    // 异常处理
    //     xyzClose("MasterSock")
    // endif

    // 切换工件, 两个参数分别表示视觉服务 ID , 和工件代号
    // 工件代号为 MAX 中映射表与通讯协议设置中的参数
    err_code = xyzSwitchObject(0, "obj1")
    if (err_code != 0)                    // 异常处理
        xyzClose("MasterSock")
    endif

    Reset output0; //重置数字输出

    // 首先移动到 home 点。注意需要先定义 home_pose␣
↪的关节角度值, 或者用使用其他点来替代
    MoveAbsJ home_pose,v1000,z50,tool0

    While (true)
        // 请求抓取和放置规划
        err_code = xyzReqPickPlace(0)
        if (err_code != 0)                  // 异常处理
            xyzClose("MasterSock")
        endif

        // 请求 pick in 轨迹
        err_code = xyzGetPickin(0)
        if (err_code != 0)                  // 异常处理
            xyzClose("MasterSock")
        endif
        if (PoseNum < 1)
            print("Tote cleared")
```

(下页继续)

(续上页)

```
            break
        endif

        // 执行 pick in 轨迹
        xyzExecuteTraj()
        SetDO DO0_0, true

        // 请求 pick out 轨迹
        err_code = xyzGetPickout(0)
        if (err_code != 0)              // 异常处理
            xyzClose("MasterSock")
        endif

        // 执行 pick out 轨迹
        xyzExecuteTraj()

        // 请求 place in 轨迹
        err_code = xyzGetPlacein(0)
        if (err_code != 0)              // 异常处理
            xyzClose("MasterSock")
        endif

        // 执行 place in 轨迹
        xyzExecuteTraj()
        Reset output0;

        // 请求 place out 轨迹
        err_code = xyzGetPlaceout(0)
        if (err_code != 0)              // 异常处理
            xyzClose("MasterSock")
        endif

        // 执行 place out 轨迹
        xyzExecuteTraj()
    Endwhile
ENDPROC
```

轨迹移动异步模板

```
GLOBAL PROC TrajMoveAsync()
    VAR int err_code           // 错误代码
    VAR jointtarget home_pose  // home 点位

    xyzClose("MasterSock")   // 断开可能已存在的socket
    xyzConnect("MasterSock") // 连接上位机 server

    // err_code = xyzSwitchFlow("traj_sync.t")    // 切换 flow␣
↪, 默认被注释掉, 如果需求可以取消注释
    // if (err_code != 0)                    // 异常处理
    //     xyzClose("MasterSock")
    // endif

    // 切换工件, 两个参数分别表示视觉服务 ID , 和工件代号
    // 工件代号为 MAX 中映射表与通讯协议设置中的参数
```

(下页继续)

(续上页)

```
    err_code = xyzSwitchObject(0, "obj1")
    if (err_code != 0)                  // 异常处理
        xyzClose("MasterSock")
    endif

    Reset output0; //重置数字输出

    // 首先移动到 home 点。注意需要先定义 home_pose␣
↪的关节角度值，或者用使用其他点来替代
    MoveAbsJ home_pose,v1000,z50,tool0

    err_code = xyzReqPickPlace(0)  // 请求抓取和放置规划
    if (err_code != 0)                  // 异常处理
        xyzClose("MasterSock")
    endif

    While (true)
        // 请求抓取和放置规划
        err_code = xyzReqPickPlace(0)
        if (err_code != 0)              // 异常处理
            xyzClose("MasterSock")
        endif

        // 请求 pick in 轨迹
        err_code = xyzGetPickin(0)
        if (err_code != 0)              // 异常处理
            xyzClose("MasterSock")
        endif
        if (PoseNum < 1)
            print("Tote cleared")
            break
        endif

        // 执行 pick in 轨迹
        xyzExecuteTraj()
        SetDO DO0_0, true

        // 请求 pick out 轨迹
        err_code = xyzGetPickout(0)
        if (err_code != 0)              // 异常处理
            xyzClose("MasterSock")
        endif

        // 执行 pick out 轨迹
        xyzExecuteTraj()

        // 运动到相机视野外，请求下一次轨迹运算
        err_code = xyzReqPickPlace(0)  // 请求抓取和放置规划
        if (err_code != 0)                  // 异常处理
            xyzClose("MasterSock")
        endif

        // 继续请求本次 place in 轨迹
        err_code = xyzGetPlacein(0)
        if (err_code != 0)              // 异常处理
            xyzClose("MasterSock")
```

(下页继续)

(续上页)

```
        endif

        // 执行 place in 轨迹
        xyzExecuteTraj()
        Reset output0;

        // 继续请求本次 place out 轨迹
        err_code = xyzGetPlaceout(0)
        if (err_code != 0)              // 异常处理
            xyzClose("MasterSock")
        endif

        // 执行 place out 轨迹
        xyzExecuteTraj()
    Endwhile
ENDPROC
```

常见问题

附录

14.2.17 Ur

此处介绍安装 Ur 机械臂驱动的相关事项。

驱动版本和使用要求

支持的机械臂类型

Ur 六轴协作机械臂

控制器型号

CB3.0/CB3.1

机械臂需要开通的功能

Ur 自带 socket 通讯功能，无需开通。

安装驱动

ur 机械臂驱动文件列表

–ur

- xyz_master.script （机械臂主控程序）
- xyz_motion.script （工控机主控程序）
- xyz_status.script （工控机主控程序）

设定机械臂 IP

设置-> 系统-> 网络

- “IP”设置为：192.168.37.101

图 91: ur 设定机械臂 IP

设定工控机 IP

1. 用网线连接电脑和控制柜上的 **网线接口**。
2. 将电脑的 ipv4 网络设定为:

- “IP”设置为：192.168.37.100
- “子网掩码”设置为：255.255.255.0

导入程序

1. 只需将装有 ur 机械臂驱动程序的 u 盘插入示教器上的 usb 接口即可

运行程序

通讯说明

工控机和 ur 机械臂的通讯方式为 TCP/IP：工控机作为 socket server（服务端），机械臂作为 socket client（客户端）。

启动程序

工控机主控设置

示教器设置

图 92: 工控机主控程序设置

机械臂主控设置

同工控机主控，此时不需要添加线程，在机械臂程序下添加 `xyz_master` ，简单的用法示例如下图

Ur非整站使用示例：
机器人程序一开始将xyz_master.script导入即可，后续可以通过引入加入变量直接调用非整站脚本中的函数获取返回值

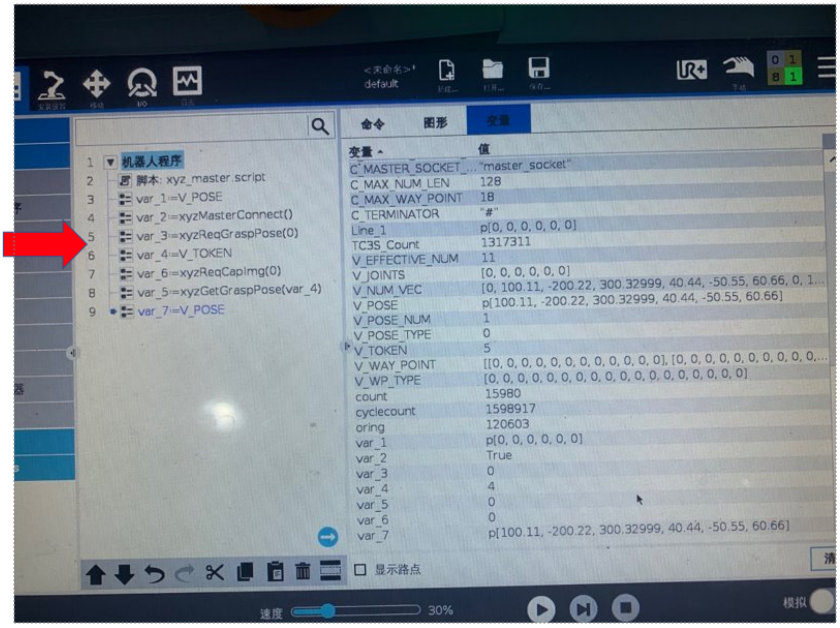


图 93: 机械臂主控使用示例

API 说明

ur 工控机主控支持的 API

指令号	命令	支持情况
100	获取机械臂程序版本号	支持
101	发送速度数据	支持
102	发送加速度数据	支持
103	发送圆滑过渡参数	支持
104	发送工具坐标系 (TCP) 参数	支持
105	设置数字量输出	支持
106	SetJointsMovej	支持
107	SetCartMovel	支持
108	SetJoinsMovel	支持
109	SetCartMovej	支持
110	MovejSequence	支持
111	MovelSequence	支持
112	SetJointsMovejDo	不支持
113	SetCartMovelDo	不支持
114	SetJointsMovelDo	不支持
115	SetJointsMovejGroupDo	不支持
116	SetCartMovelGroupDo	不支持
117	SetJointsMovelGroupDo	不支持
118	MovelUntil	不支持
119	获取数字量输入状态	支持
120	获取数字量输出状态	支持
121	获取模拟量输入口数值	支持
122	获取机械臂当前角度	支持
123	获取机械臂当前位姿	支持
200	机械臂后台发送状态	支持

ur 机械臂主控支持的 API

xyzSwitchApp(*app_name*)

切换应用

参数 **app_name**(*string*) –应用名称

返回 error_code

返回类型 num

xyzSwitchObj(*obj_name*)

切换工件

参数 **obj_name**(*string*) –工件名称

返回 error_code

返回类型 num

xyzSwitchTool(*tool_name*)

切换工具

参数 **tool_name**(*string*) –工具名称

返回 error_code

返回类型 num

xyzReqCapImg(*ws_id*)

请求拍照 (注：该函数更新全局变量)

参数

- **ws_id** (*num*) –工作空间 id
- **V_TOKEN** (*num*) –获取拍照结果的凭据 (全局变量)

返回 error_code

返回类型 num

xyzGetCapImg(*token*)

获取拍照结果

参数 **token** (*num*) –请求拍照时返回的 token

返回 error_code

返回类型 num

xyzCapImg(*ws_id*)

拍照

参数 **ws_id** (*num*) –需要进行拍照操作的工作空间 id

返回 error_code

返回类型 num

xyzReqGraspPose(*ws_id*)

请求抓取位姿 (注：该函数更新全局变量)

参数

- **ws_id** (*num*) –需要获取抓取点位的工作空间 id
- **V_TOKEN** (*num*) –返回的用于获取目标点位时使用的 token(全局变量)

返回 error_code

返回类型 num

xyzGetGraspPose(*token*)

获取抓取位姿 (注：该函数更新全局变量)

参数

- **token** (*num*) –请求抓取目标点位时返回的 token
- **V_POSE_NUM** (*num*) –可供抓取的点数量 (全局变量)
- **V_POSE_TYPE** (*num*) –当前返回的抓取点的 pose 类型 (全局变量)
- **V_POSE** (*p[x,y,z,rx,ry,rz]*) –抓取位姿 (全局变量)

返回 error_code

返回类型 num

xyzReqObjPose(*ws_id*)

请求物体位姿 (注：该函数更新全局变量)

参数

- **ws_id** (*num*) –需要获取物体位姿的工作空间 id
- **V_TOKEN** (*num*) –物体位姿识别的 token(全局变量)

返回 error_code

返回类型 num

xyzGetObjPose(*token*)

获取物体位姿 (注：该函数更新全局变量)

参数

- **token** (*num*) –请求物体位姿时得到的 token
- **V_POSE_NUM** (*num*) –物体数量 (全局变量)
- **V_POSE_TYPE** –当前返回的物体 pose 类型 (全局变量)
- **V_POSE** (*p[x,y,z,rx,ry,rz]*) –物体位姿 (全局变量)

返回 error_code

返回类型 num

xyzResetVision(*ws_id*)

重置视觉

参数 **ws_id** (*num*) –需要重置视觉的工作空间 id

返回 error_code

返回类型 num

xyzSendCurrentJoints()

发送当前关节位姿

返回 error_code

返回类型 num

xyzSendCurrentCartPose()

发送当前笛卡尔空间位姿

返回 error_code

返回类型 num

xyzReqPick()

请求 pick 动作规划

返回 error_code

返回类型 num

xyzReqPlace()

请求 place 动作规划

返回 error_code

返回类型 num

xyzReqPickPlace()

请求 pick 和 place 规划

返回 error_code

返回类型 num

xyzGetPickin()

获取取料入框轨迹 (注：该函数不需要传入参数，仅更新全局变量)

参数

- **V_POSE_NUM** (*num*) –轨迹点数 (全局变量)
- **V_POSE_TYPE** (*num*) –轨迹点 pose 类型 (全局变量)
- **V_WP_TYPE[]** (*array*) –轨迹点类型数组 (全局变量)
- **V_WAY_POINT[]** (*2d-array*) –轨迹点关节位姿数组 (全局变量, 不同类型轨迹对应数组中不同位置)

返回 error_code

返回类型 num

xyzGetPickout()

获取取料出框轨迹 (注：该函数不需要传入参数，仅更新全局变量)

参数

- **V_POSE_NUM** (*num*) –轨迹点数 (全局变量)
- **V_POSE_TYPE** (*num*) –轨迹点 pose 类型 (全局变量)
- **V_WP_TYPE[]** (*array*) –轨迹点类型数组 (全局变量)
- **V_WAY_POINT[]** (*2d-array*) –轨迹点关节位姿数组 (全局变量, 不同类型轨迹对应数组中不同位置)

返回 error_code

返回类型 num

xyzGetPlacein()

获取放料入框轨迹 (注：该函数不需要传入参数，仅更新全局变量)

参数

- **V_POSE_NUM** (*num*) –轨迹点数 (全局变量)
- **V_POSE_TYPE** (*num*) –轨迹点 pose 类型 (全局变量)
- **V_WP_TYPE[]** (*array*) –轨迹点类型数组 (全局变量)
- **V_WAY_POINT[]** (*2d-array*) –轨迹点关节位姿数组 (全局变量, 不同类型轨迹对应数组中不同位置)

返回 error_code

返回类型 num

xyzGetPlaceout()

获取放料出框轨迹 (注：该函数不需要传入参数，仅更新全局变量)

参数

- **V_POSE_NUM** (*num*) –轨迹点数 (全局变量)
- **V_POSE_TYPE** (*num*) –轨迹点 pose 类型 (全局变量)
- **V_WP_TYPE[]** (*array*) –轨迹点类型数组 (全局变量)

- **V_WAY_POINT[]** (*2d-array*) –轨迹点关节位姿数组 (全局变量, 不同类型轨迹对应数组中不同位置)

返回 error_code

返回类型 num

xyzSwitchStrat(*strat_name*)

请求切换策略

参数 **strat_name** (*string*) –策略名称

返回 error_code

返回类型 num

xyzUpdateTotePose()

料箱重定位 (注：该函数不需要传入参数，仅更新全局变量)

参数 **V_POSE** (*p[x,y,z,rx,ry,rz]*) –料箱位姿 (全局变量)

返回 error_code

返回类型 num

xyzUpdateObjPoseOnHand()

工件在上手的二次定位

返回 error_code

返回类型 num

xyzUpdateObjPoseToHand()

工件不在手上的二次定位，获取二次抓取的轨迹 (注：该函数不需要传入参数，仅更新全局变量)

参数

- **V_POSE_NUM** (*num*) –轨迹点数 (全局变量)
- **V_POSE_TYPE** (*num*) –轨迹点 pose 类型 (全局变量)
- **V_WP_TYPE[]** (*array*) –轨迹点类型数组 (全局变量)
- **V_WAY_POINT[]** (*2d-array*) –轨迹点关节位姿数组 (全局变量, 不同类型轨迹对应数组中不同位置)

返回 error_code

返回类型 num

xyzGetObjPoseType()

获取工件姿态类型 (注：该函数不需要传入参数，仅更新全局变量)

参数 **V_POSE_TYPE** (*num*) –工件姿态类型 (全局变量)

返回 error_code

返回类型 num

xyzResetPalletStatus()

重置工业码垛状态

返回 error_code

返回类型 num

xyzExecuteTraj()

用于执行一段轨迹，注意所有笛卡尔空间位姿与轨迹点皆为全局变量 (注：该函数不需要传入参数，调用的均为全局变量)

参数

- **V_POSE_NUM** (*num*) –轨迹点数 (全局变量)
- **V_POSE_TYPE** (*num*) –轨迹点 pose 类型 (全局变量)
- **V_WP_TYPE[]** (*array*) –轨迹点类型数组 (全局变量)
- **V_WAY_POINT[]** (*2d-array*) –轨迹点关节位姿数组 (全局变量, 不同类型轨迹对应数组中不同位置)

返回 error_code

返回类型 num

案例/模板说明

机械臂主控主函数说明

以下为机械臂主控模板代码，注意对工控机返回的 err_code 进行判断。

```
def xyzMain():
xyzMasterConnect() #连接工控机
xyzTest() #集成测试
xyzMasterClose() #断开连接
end

##########################Test Code#################################

# This function is for testing all api
def xyzTest():
xyzTestSwitchApp()
xyzTestSwitchObj()
xyzTestSwitchTool()
# Because of token, the next 3 test sequence can not be changed
xyzTestImg()
xyzTestGrasp()
xyzTestObj()
xyzTestResetVision()
xyzTestSendJointCartExtJoint()
xyzTestReqPickPlace()
xyzTestGetPickPlace()
xyzTestSwitchStrat()
xyzTestUpdate()
xyzTestGetObjPoseType()
xyzTestRestPalletStatus()
end

def xyzTestSwitchApp():
app_name = "111"
error_code = xyzSwitchApp(app_name)
app_name = "error_app"
error_code = xyzSwitchApp(app_name)
end
```

(下页继续)

(续上页)

```
def xyzTestSwitchObj():
obj_name = "222"
error_code = xyzSwitchObj(obj_name)
obj_name = "error_obj"
error_code = xyzSwitchObj(obj_name)
end

def xyzTestSwitchTool():
tool_name = "333"
error_code = xyzSwitchTool(tool_name)
tool_name = "error_tool"
error_code = xyzSwitchTool(tool_name)
end

def xyzTestImg():
ws_id = 0
token = 0
# token in robot server will increase after every call
error_code = xyzReqCapImg(ws_id)
error_code = xyzGetCapImg(token)
ws_id = 0
error_code = xyzCapImg(ws_id)
end

def xyzTestGrasp():
ws_id = 0
pose_num = 0
pose_type = 0
grasp_pose = p[0,0,0,0,0,0]
token = 0
# token in robot server will increase after every call
error_code = xyzReqGraspPose(ws_id)
token = 2
error_code = xyzGetGraspPose(token)
end

def xyzTestObj():
pose_num = 0
pose_type = 0
obj_pose = p[0,0,0,0,0,0]
ws_id = 0
token = 0
# token in robot server will increase after every call
error_code = xyzReqObjPose(ws_id)
token = 3
error_code = xyzGetObjPose(token)
end

def xyzTestResetVision():
ws_id = 0
error_code = xyzResetVision(ws_id)
end

def xyzTestSendJointCartExtJoint():
error_code = xyzSendCurrentJoints()
```

(下页继续)

(续上页)

```
error_code = xyzSendCurrentCartPose()
end

def xyzTestReqPickPlace():
error_code = xyzReqPick()
error_code = xyzReqPlace()
error_code = xyzReqPickPlace()
end

def xyzTestGetPickPlace():
pose_type = 0
pose_num = 0
# please check the waypoint data before xyzExecuteTraj() !
error_code = xyzGetPickin()
xyzExecuteTraj()
error_code = xyzGetPickout()
xyzExecuteTraj()
error_code = xyzGetPlacein()
xyzExecuteTraj()
error_code = xyzGetPlaceout()
xyzExecuteTraj()
end

def xyzTestSwitchStrat():
strat_name = "strat1"
error_code = xyzSwitchStrat(strat_name)
strat_name = "error_strat"
error_code = xyzSwitchStrat(strat_name)
end

def xyzTestUpdate():
tote_pose = p[0,0,0,0,0,0]
pose_type = 0
pose_num = 0
error_code = xyzUpdateTotePose()
error_code = xyzUpdateObjPoseOnHand()
error_code = xyzUpdateObjPoseToHand()
end

def xyzTestGetObjPoseType():
pose_type = 0
error_code = xyzGetObjPoseType()
end

def xyzTestRestPalletStatus():
error_code = xyzResetPalletStatus()
end

# This is the main function for test 主函数
xyzMain()
```

常见问题

附录

14.2.18 Mitsubishi

此处介绍安装三菱 Mitsubishi 机械臂驱动的相关事项。

驱动版本和使用要求

支持的机械臂类型

三菱：六轴机械臂

控制柜型号

FR-D 系列 CR800-D(CR800-CVD)

机械臂需要开通的功能

自带 Ethernet 通讯功能，无需开通

安装驱动

三菱机械臂驱动文件列表

–**mitsubishi**

- fake_motion_server.py (内部测试程序，用不到)
- fake_status_server.py (内部测试程序，用不到)
- MASTERTEST.prg (内部测试程序，用不到)
- `XYZCARTBASIC.prg` (机械臂主控：CartMove 基础模板)
- `XYZCARTREPO.prg` (机械臂主控：CartMove 二次定位模板)
- `XYZMOTION.prg` (工控机主控：前台程序)
- `XYZSTATUS.prg` (工控机主控：后台程序)
- `XYZTRAJASYNC.prg` (机械臂主控：TrajMove 异步模板)
- `XYZTRAJSYNC.prg` (机械臂主控：TrajMove 同步模板)
- `XYZUTILS.prg` (机械臂主控：api 函数文件)

安装三菱软件 RT ToolBox3

三菱 RT ToolBox3 可以在电脑上方便连接机器人、程序导入导出、程序编写运行等。

三菱 RT ToolBox3 软件的获取，请咨询客户或者相关提供商。

以下操作基于 RT ToolBox3，且示教器型号为 R32。不同示教器操作略有不同，请用户根据具体情况操作。

设定机械臂 IP

确认机器人当前 IP 地址

1. 将机器人切换到手动模式
2. 示教器上执行 F1 -> 移动箭头到 3.PARAM -> EXE -> 在 NAME 选项中输入 NETIP -> EXE -> 可以查看当前机械臂网口的 IP 地址 -> 按界面右下角 CLOSE 选项对应的 F 按钮关闭当前查询页面

图 94: 示教器查看 IP 地址

使用 RT ToolBox3　创建项目并修改机器人的 IP 地址

1. 确保机器人和电脑处于统一 IP 网段
2. 打开 RT ToolBox3 -> 菜单栏 工作区 -> 新建 -> 选择相应文件夹并输入工作区名 -> OK

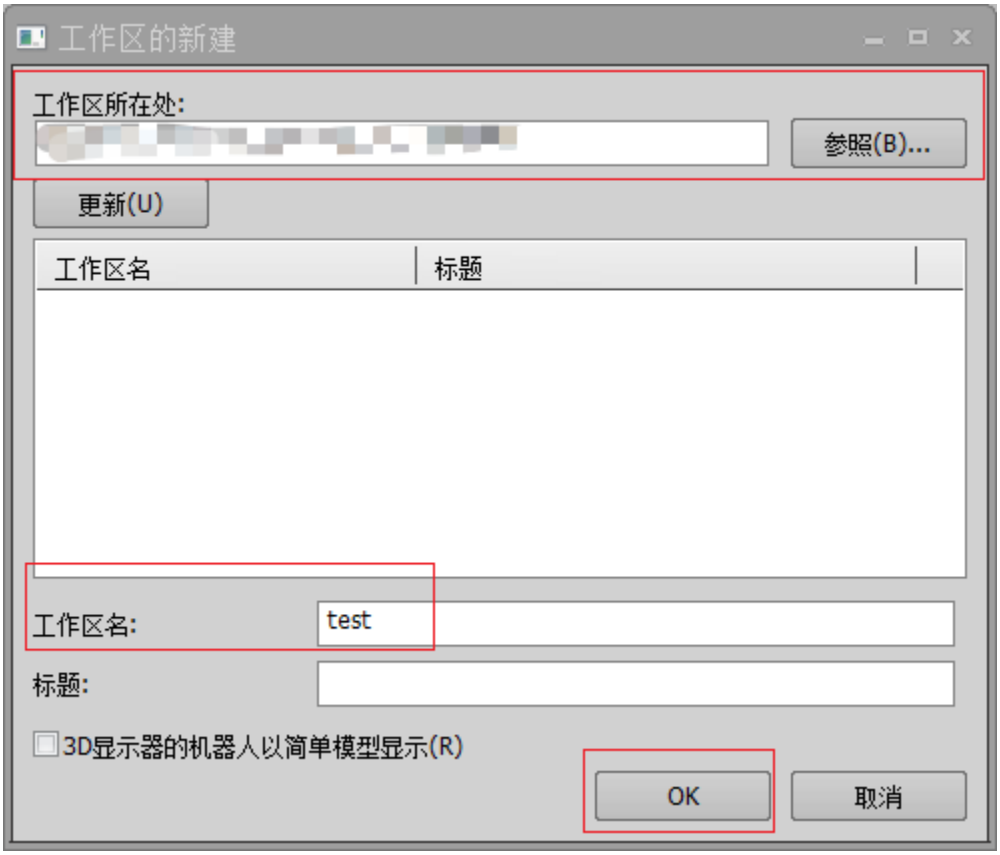


图 95: 创建工作区

3. 后续设置如下：

   在跳出的对话框中选择 在线即可连接上机器

   后续也可以通过菜单栏的 主页 -> 模式 -> 在线连接机器人

4. 通讯设置：对应的项目 -> 对应的工程名 -> 在线 -> 参数 -> 通信参数 -> Ethernet 设定，做如下设定，然后 写入 -> 等待控制器复位和软件重连

   工控机主控和机械臂主控中，机器人均做 socket client（客户端），工控机做 socket server（服务器）。

   工控机主控用到 COM1 和 COM2，机械臂主控用到 COM3，不要修改 COM 地址。

   默认设备用到 OPT15/OPT16/OPT17。用户如果需要求改设备，请至少从 OPT14 开始使用。

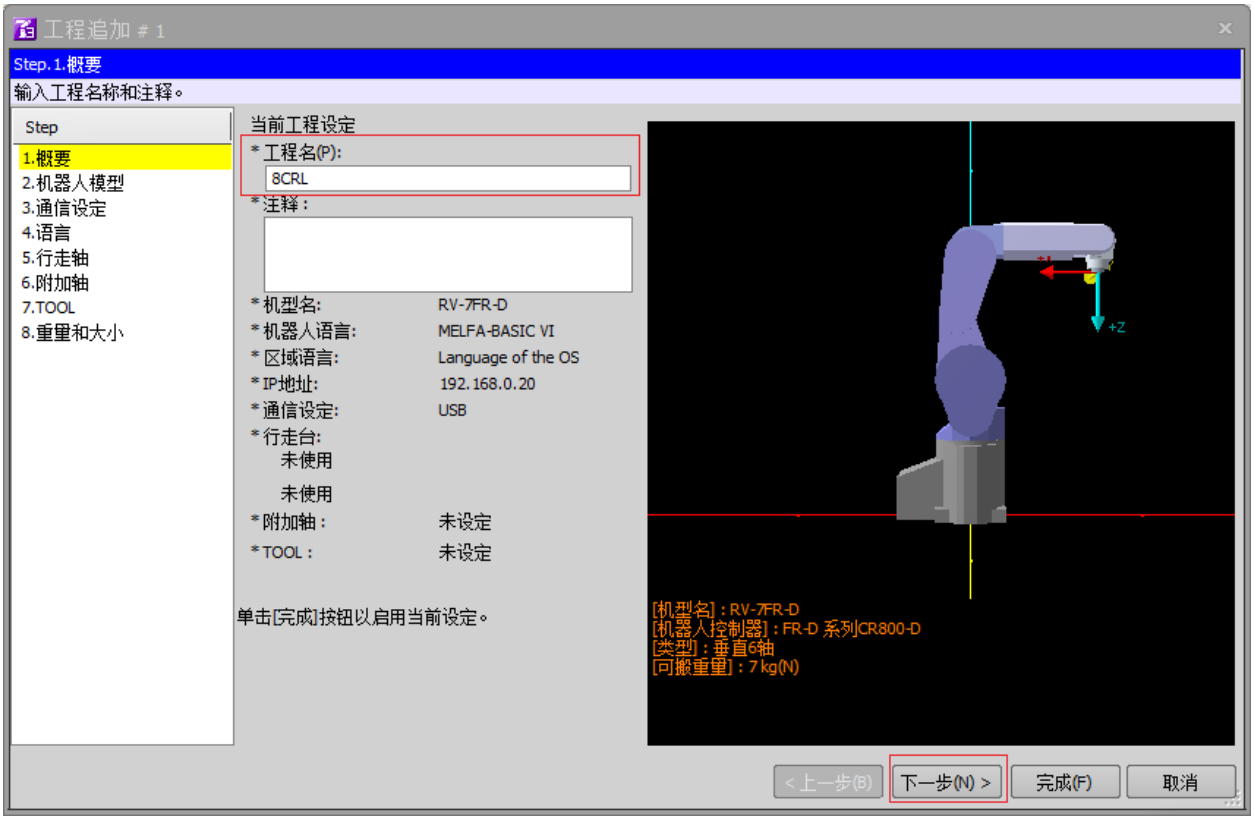

工程追加 # 1
Step. 1.概要
输入工程名称和注释。
Step
1.概要
2.机器人模型
3.通信设定
4.语言
5.行走轴
6.附加轴
7.TOOL
8.重量和大小
当前工程设定
* 工程名(P):
8CRL
* 注释：
* 机型名: RV-7FR-D
* 机器人语言: MELFA-BASIC VI
* 区域语言: Language of the OS
* IP地址: 192.168.0.20
* 通信设定: USB
* 行走台:
未使用
未使用
* 附加轴： 未设定
* TOOL： 未设定
单击[完成]按钮以启用当前设定。
+Z
[机型名]：RV-7FR-D
[机器人控制器]：FR-D 系列CR800-D
[类型]：垂直6轴
[可搬重量]：7 kg(N)
< 上一步(B)
下一步(N) >
完成(F)
取消

图 96: Step1

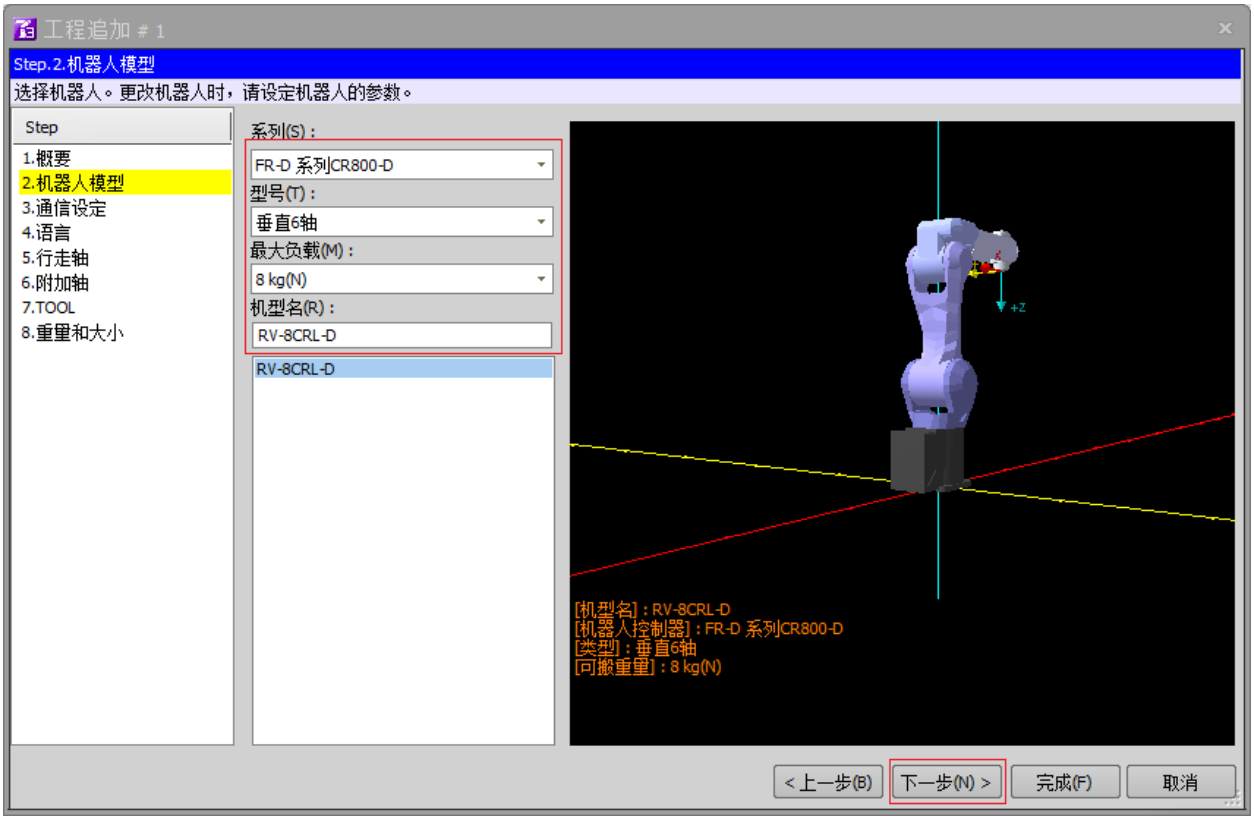

工程追加 # 1
Step.2.机器人模型
选择机器人。更改机器人时，请设定机器人的参数。
Step
1.概要
2.机器人模型
3.通信设定
4.语言
5.行走轴
6.附加轴
7.TOOL
8.重量和大小
系列(S)：
FR-D 系列CR800-D
型号(T)：
垂直6轴
最大负载(M)：
8 kg(N)
机型名(R)：
RV-8CRL-D
RV-8CRL-D
+Z
[机型名]：RV-8CRL-D
[机器人控制器]：FR-D 系列CR800-D
[类型]：垂直6轴
[可搬重量]：8 kg(N)
< 上一步(B)
下一步(N) >
完成(F)
取消

图 97: Step2

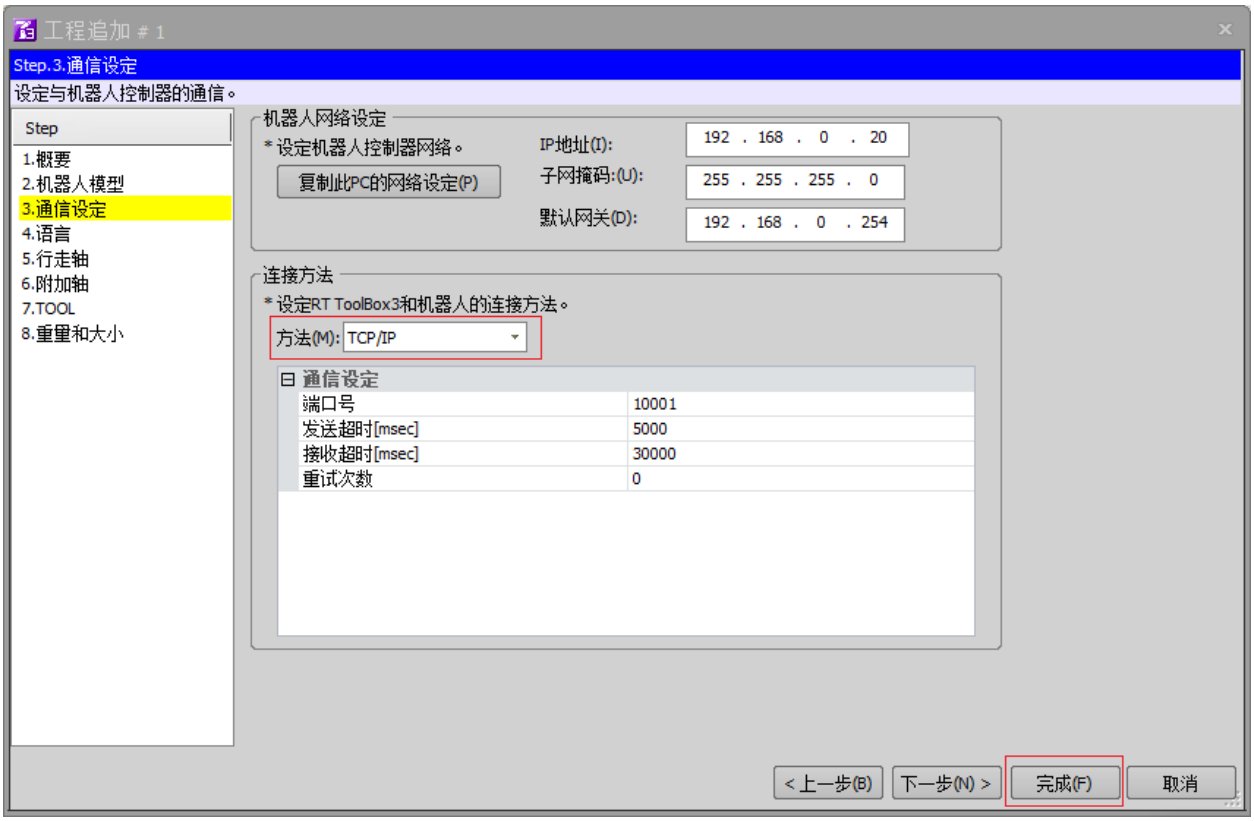


图 98: Step3

图 99: 在线

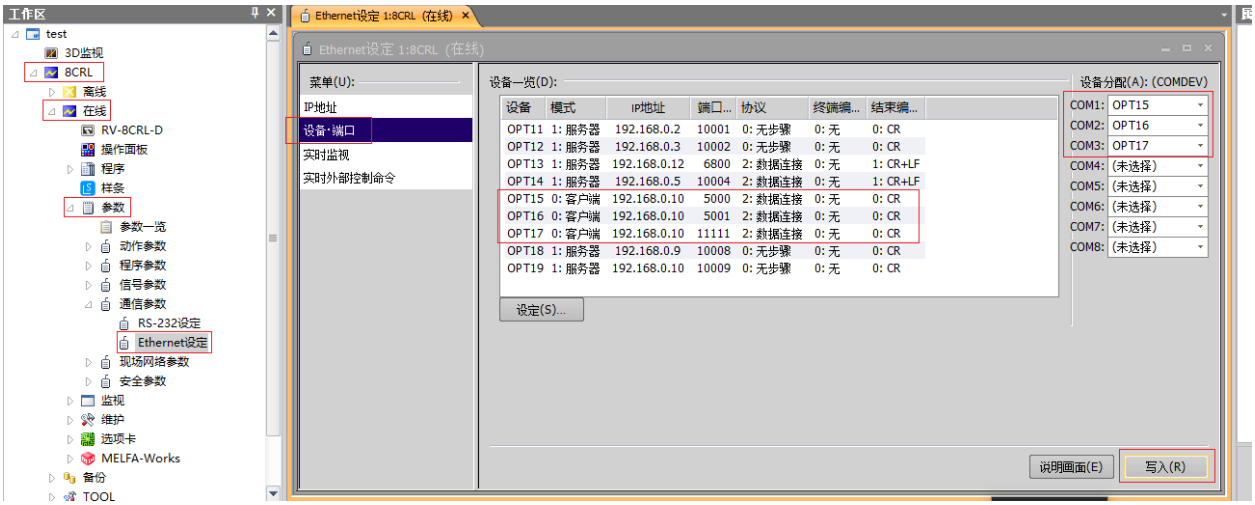


图 100: 通讯设置 1

图 101: 通讯设置 2

导入程序

1. 将三菱程序文件拷贝到项目的 `Program` 文件夹下

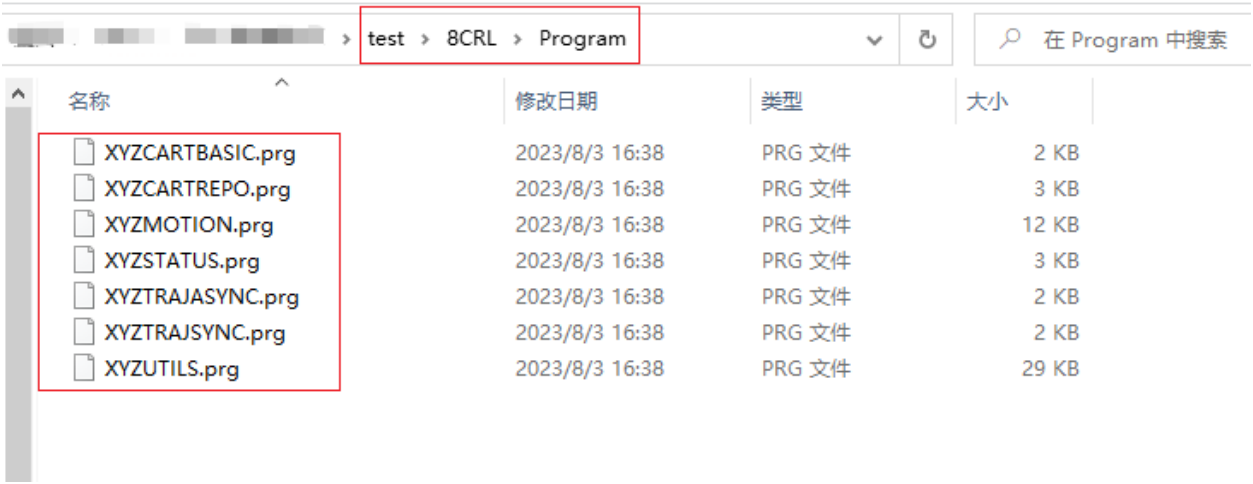


图 102: 程序复制到 `RT ToolBox3` 的项目下

2. 工作区 -> 对应的项目 -> 对应的工程名 -> 在线 -> 程序 -> 右键 程序管理

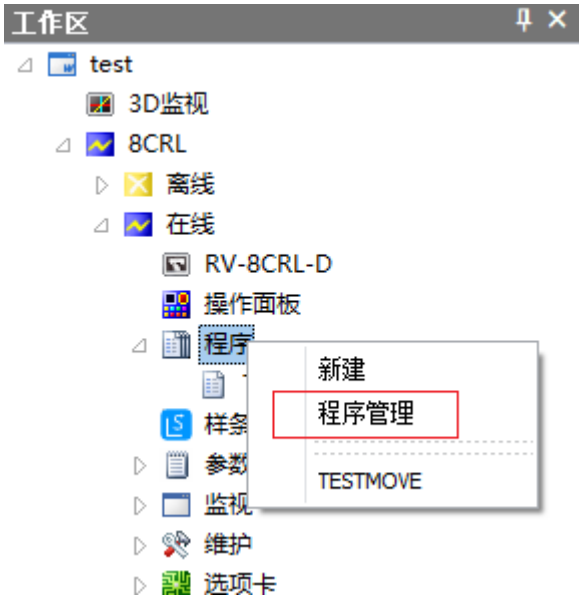


图 103: 程序管理

3. 在 程序管理弹窗中, 设置传送源为 工程，文件一栏中勾选对应的三菱程序文件。在传送目标中选择机器人，然后点 复制。等待复制完成并关闭窗口即可。

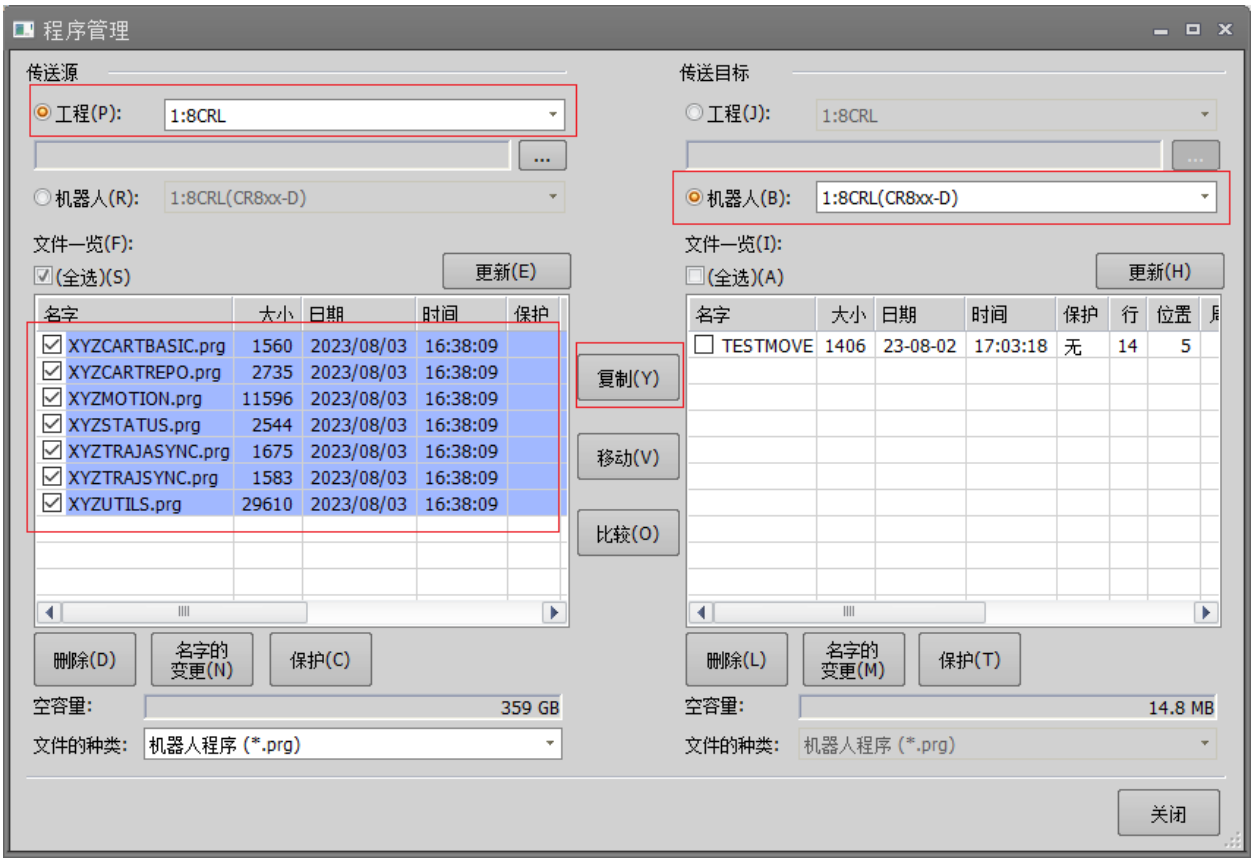


图 104: 程序管理

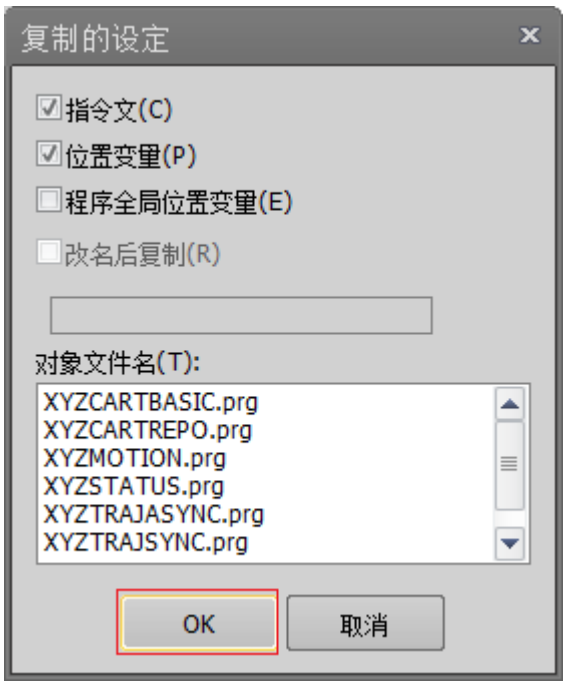


图 105: 确定

三菱机器人常见操作汇总

可以快速查阅常见操作

示教器常见操作

1. 切换到手动模式将机器人模式开关切换到手动档，并激活示教器背面的权限按钮，使示教器权限灯为亮的状态
2. 手动状态下使能按住示教器背面的使能开关 + SERVO 按键
3. 切换到自动模式将机器人模式开关切换到自动档，并熄灭示教器背面的指示灯
4. 清错

   按示教器的 RESET 按钮
5. 查看机器人当前位置

   按示教器的 JOG 按键
6. 切换菜单栏

   当菜单栏选项较多，无法在一个界面中显示完全时，可以按示教器的 FUNCTION 进行菜单栏切换

RT ToolBox3 操作面板介绍

运行程序

通讯说明

工控机和三菱机械臂的通讯方式为 socket。

工控机作为 socket server(服务端)，机械臂作为 socket client（客户端)。

启动程序

运行工控机主控

1. 机器人切换到自动模式
2. 启动操作面板：工作区 -> 对应的项目 -> 对应的工程名 -> 在线 -> 操作面板 -> 弹窗选择 OK

3. 操作面板：加载 XYZMOTION

4. 工控机启动 robot_driver_node 后
5. 操作面板：设置运行速度 -> 伺服 ON -> 开始


1:8CRL 机器人控制器
加载程序
查看当前加载的程序和运行的行号
设置运行速度
状态:
可以选择程序
程序:
选择
行号:
00000
跳转
速度比率:
50
操作面板
伺服ON
伺服OFF
开始
停止
复位
结束
上使能，断使能和运行程序
停止程序运行，复位程序，结束程序
查看报错信息
监视
3D
报警
程序
单步
后退
前进
继续
直接执行
抓手
JOG
运行速度
平时
低速率
AUTOMATIC
查看当前模式
监视当前程序的运行
抓手排列
退避点复位
1 : RV-8CRL-D
关节
Off
TOOL0
BASE0
工件1
J1:
16.010
J2:
17.240
J3:
116.710
J4:
4.470
J5:
48.700
J6:
104.170
选中后，可以进行点动，并查看当前坐标值。只有自动模式才可以进行点动操作

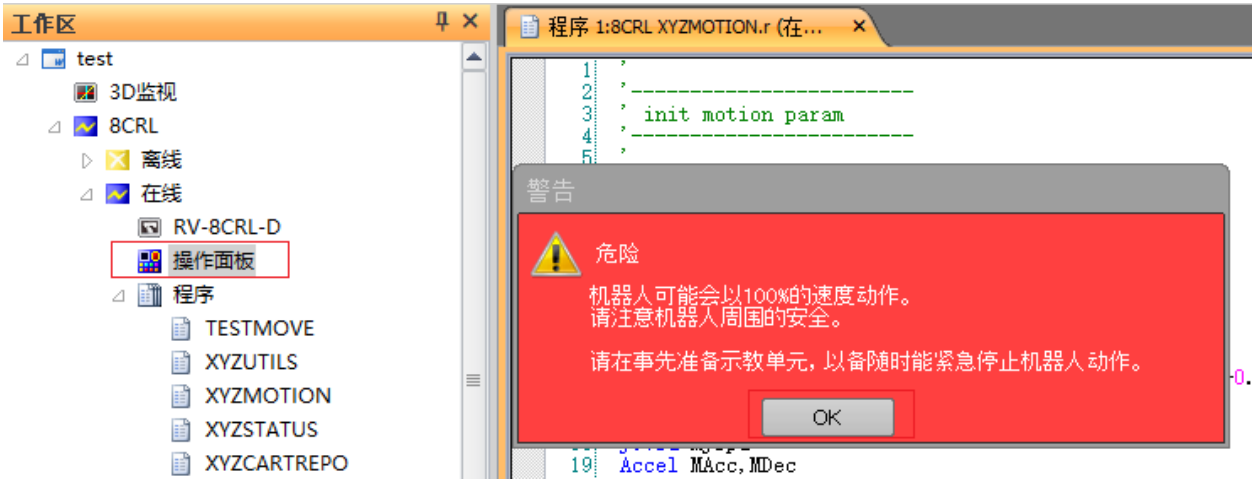

工作区
test
3D监视
8CRL
离线
在线
RV-8CRL-D
操作面板
程序
TESTMOVE
XYZUTILS
XYZMOTION
XYZSTATUS
XYZCARTREPO
程序 1:8CRL XYZMOTION.r (在...
' init motion param
Accel MAcc, MDec
警告
危险
机器人可能会以100%的速度动作。
请注意机器人周围的安全。
请在事先准备示教单元，以备随时能紧急停止机器人动作。
OK

图 106: 启动操作面板

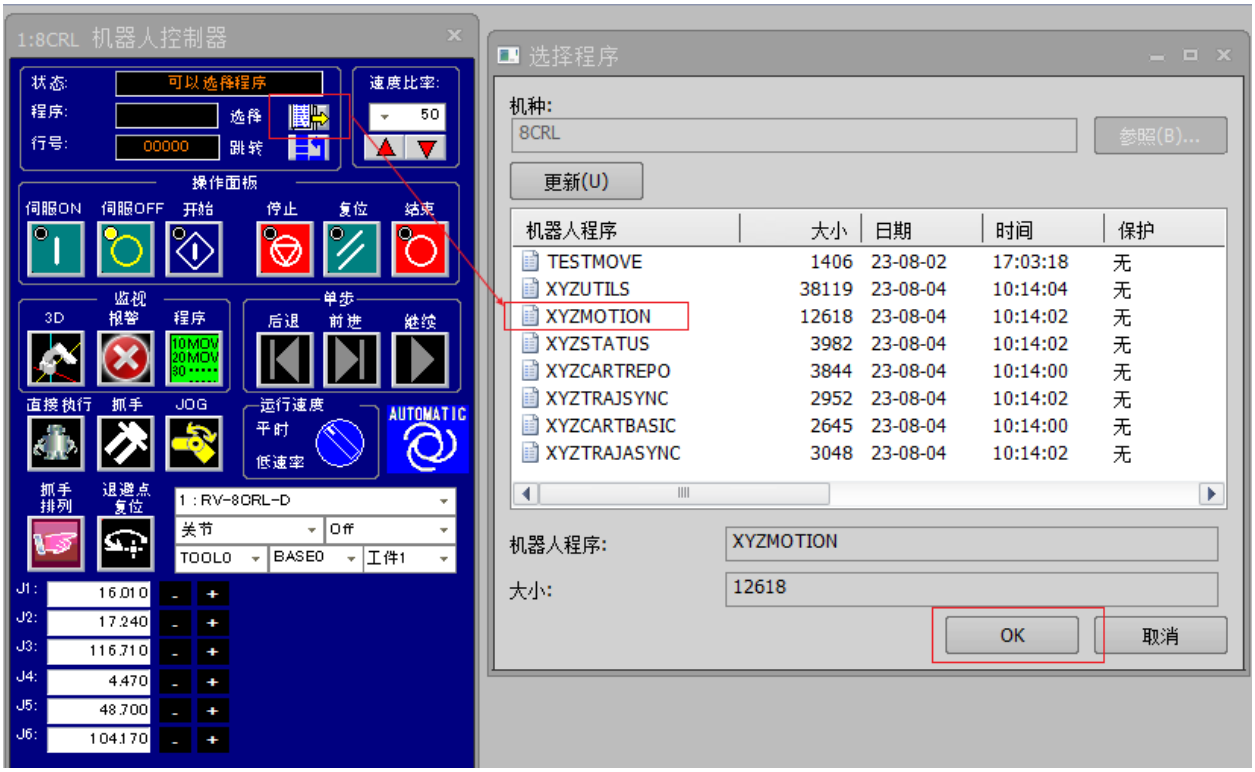


图 107: 选择机器人程序

运行机械臂主控

1. 参考下位机模板程序，根据项目运行逻辑，修改下位机运行程序：工作区 -> 对应的项目 -> 对应的工程名 -> 在线 -> 程序 -> 双击模板程序，如 XYACARTBASIC -> 弹窗选择 OK -> 修改

2. 机器人切换到自动模式
3. 启动操作面板：工作区 -> 对应的项目 -> 对应的工程名 -> 在线 -> 操作面板 -> 弹窗选择 OK
4. 加载运行程序
5. 工控机启动 robot_server
6. 操作面板：设置运行速度 -> 伺服 ON -> 开始


1:8CRL 机器人控制器
状态:
可以选择程序
程序:
XYZMOTION
选择
行号:
00001
跳转
速度比率:
20
操作面板
伺服ON
伺服OFF
开始
停止
复位
结束
监视
3D
报警
程序
单步
后退
前进
继续
直接执行
抓手
JOG
运行速度
平时
低速率
AUTOMATIC
抓手排列
退避点复位
1 : RV-8CRL-D
关节
Off
TOOL0
BASE0
工件1
J1:
16.010
J2:
17.240
J3:
116.710
J4:
4.470
J5:
48.700
J6:
104.170

图 108: 运行程序

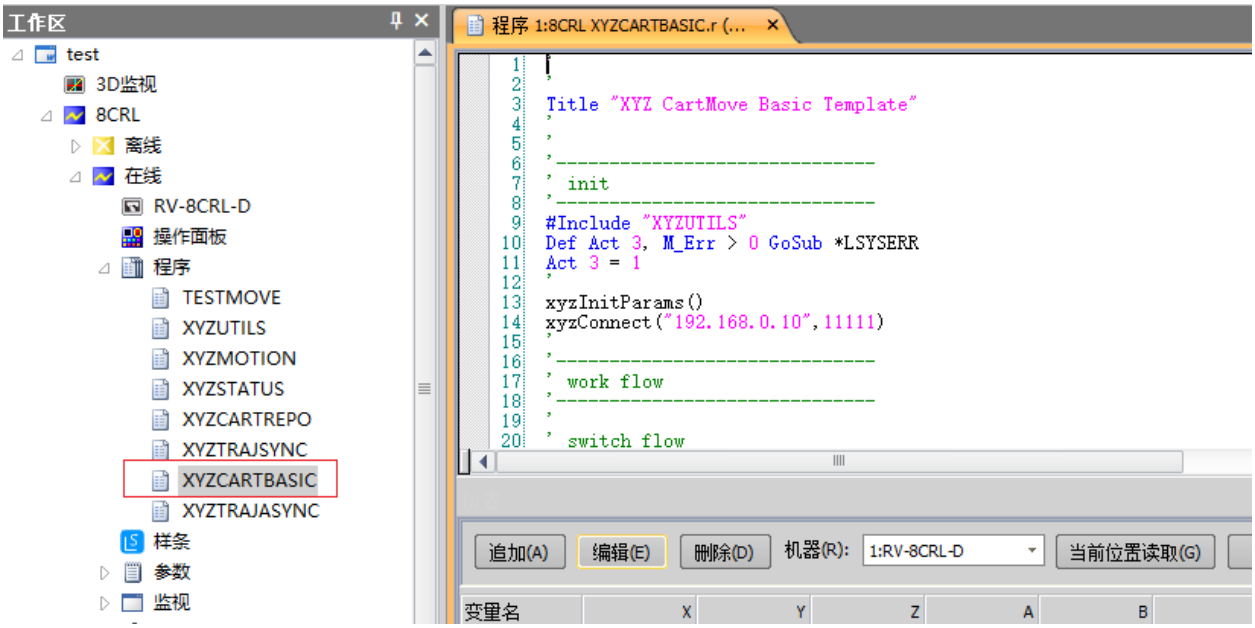


图 109: 修改模板程序

API 说明

三菱：工控机主控支持的 API

指令号	命令	支持情况
100	获取机械臂程序版本号	支持
101	发送速度数据	支持
102	发送加速度数据	支持
103	发送圆滑过渡参数	支持
104	发送工具坐标系 (TCP) 参数	支持
105	设置数字量输出	支持
106	SetJointsMovej	支持
107	SetCartMovel	支持
108	SetJoinsMovel	支持
109	SetCartMovej	支持
110	MovejSequence	支持
111	MovelSequence	支持
112	SetJointsMovejDo	不支持
113	SetCartMovelDo	不支持
114	SetJointsMovelDo	不支持
115	SetJointsMovejGroupDo	不支持
116	SetCartMovelGroupDo	不支持
117	SetJointsMovelGroupDo	不支持
118	MovelUntil	不支持
119	获取数字量输入状态	支持
120	获取数字量输出状态	支持
121	获取模拟量输入口数值	不支持
122	获取机械臂当前角度	支持
123	获取机械臂当前位姿	支持
124	调用子程序	不支持
200	机械臂后台发送状态	支持

三菱：机械臂主控支持的 API

三菱机械臂使用了一些全局变量, 不允许用户占用, 占用情况如下:

指令的返回值		
地址	含义	是否用户可写
M_00#	max 返回的 token 值	x
M_01#	grasp_pose_num 或者 object_pose_num	x
M_02#	pipeline_num	x
M_03#	register_num	x
M_04#	object_pose_type	x
P_00	grasp_pose/object_pose/tote_pose	x
M_05#	轨迹中的点的数量	x
P_100(10)~P_102(10)	轨迹中的点位和类型	x
J_100(10)~J_102(10)		
M_100(10)~M_102(10)		
C_100(6~10)	用户指令 (xyzUsrCmd) 的返回参数 C_100(6~10):5 个 string	x
M_105(10)	M_105(10):10 个 int	
M_106(10)	M_106(10):10 个 float	
P_02	P_02:1 个 cart_pose	
J_02	J_02: 一个 joints	

指令的输入参数		
地址	含义	是否用户可写
C_100(1~5)	用户指令 (xyzUsrCmd) 的输入参数 C_100(1~5):5 个 string	√
M_103(10)	M_103(10):10 个 int	
M_104(10)	M_104(10):10 个 float	
P_01	P_01:1 个 cart_pose	
J_01	J_01:1 个 joints	

以下 API 函数存放在 `XYZUTILS.prg` 。

返回值的 V 表示 void，即无返回值。

Function V xyzConnect(CIP$, MPort)

连接到服务器

参数

- **CIP$** (字符串) –服务器 IP 地址
- **MPort** (数值) –服务器端口号

返回 Void

Function M! xyzHeartBeat()

ping 上位机用

返回 err_code

返回类型 M!

Function M! xyzSwitchApp(CAppName$)

切换应用

参数 **CAppName$** (字符串) –应用名称

返回 err_code

返回类型 M!

Function M! xyzSwitchFlow(CFlowName$)

切换 flow

参数 **CFlowName$** (字符串) –flow 名称

返回 err_code

返回类型 M!

Function M! xyzSwitchTool(CTool$)

切换工具

参数 **CTool$** (字符串) –工具名称

返回 err_code

返回类型 M!

Function M! xyzReqCapImg(MVisSrvId)

请求拍照

返回的 token 值存在 M_00

参数 **MVisSrvId$** (*M*) –视觉服务 ID

返回 err_code

返回类型 M!

Function M! xyzGetCapImg(MToken)

获取拍照结果

参数 **MToken$** (*M*) –请求拍照时返回的 token

返回 err_code

返回类型 M!

Function M! xyzCapImg(MVisSrvId)

拍照

等价于同时执行:xyzReqCapImg + xyzGetCapImg

参数 **MVisSrvId$** (*M*) –视觉服务 ID

返回 err_code

返回类型 M!

Function M! xyzReqGraspPose(MWsId)

请求抓取目标点位

返回的 token 值存在 M_00

参数 **MWsId$** (*M*) –工作空间 id

返回 err_code

返回类型 M!

Function M! xyzGetGraspPose(MToken)

获取抓取目标点位

返回的 grasp_pose 存放在 P_00, grasp_pose_num 存放在 M_01, pipeline_num 存放在 M_02, register_num 存放在 M_03

参数 MToken$ (*M*) –xyzReqGraspPose 返回的 token 值

返回 err_code

返回类型 M!

Function M! xyzReqObjPose(MWsId)

请求物体位姿

返回的 token 值存在 M_00

参数 MWsId$ (*M*) –工作空间 id

返回 err_code

返回类型 M!

Function M! xyzGetObjPose(MToken)

获取物体位姿

返回的 object_pose 存放在 P_00, object_pose_num 存放在 M_01, object_pose_type 存放在 M_04

参数 MToken$ (*M*) –xyzReqObjPose 返回的 token 值

返回 err_code

返回类型 M!

Function M! xyzResetTask()

重置任务, 一般用来初始化任务内部变量

返回 err_code

返回类型 M!

Function M! xyzSendCurrentJoints()

发送机器人当前角度

返回 err_code

返回类型 M!

Function M! xyzSendCurrentCartPose()

发送机器人当前 Cartesian 坐标

返回 err_code

返回类型 M!

Function M! xyzSendCurrentExtJoints()

发送机器人当前扩展轴位置

返回 err_code

返回类型 M!

Function M! xyzReqPick()

请求 pick 动作规划

返回 err_code

返回类型 M!

Function M! xyzReqPlace()

请求 place 动作规划

返回 err_code

返回类型 M!

Function M! xyzReqPickPlace(MWsId)

请求 pick 和 place 规划

参数 MWsId$ (*M*) –工作空间 id

返回 err_code

返回类型 M!

Function M! xyzGetPickIn(MWsId)

获取取料入框轨迹

轨迹执行可以参考后面的轨迹执行函数

参数 MWsId$ (*M*) –工作空间 id

返回 err_code

返回类型 M!

Function M! xyzGetPickOut(MWsId)

获取取料出框轨迹

轨迹执行可以参考后面的轨迹执行函数

参数 MWsId$ (*M*) –工作空间 id

返回 err_code

返回类型 M!

Function M! xyzGetPlaceIn(MWsId)

获取放料入框轨迹

轨迹执行可以参考后面的轨迹执行函数

参数 MWsId$ (*M*) –工作空间 id

返回 err_code

返回类型 M!

Function M! xyzGetPlaceOut(MWsId)

获取放料出框轨迹

轨迹执行可以参考后面的轨迹执行函数

参数 MWsId$ (*M*) –工作空间 id

返回 err_code

返回类型 M!

Function M! xyzSwitchStrat(CStratName$)

请求切换策略

参数 CStratName$ (字符串) –策略名称

返回 err_code

返回类型 M!

Function M! xyzUpdateTotePose()

料箱重定位

重新定位后的料箱位置放在 P_00

返回 err_code

返回类型 M!

Function M! xyzUpdateObjPoseOnHand()

工件在手上的二次定位

返回 err_code

返回类型 M!

Function M! xyzUpdateObjPoseToHand()

工件不在手上的二次定位

返回的 pipeline_num 存放在 M_02, register_num 存放在 M_03

返回的轨迹点, 可以参考后面的轨迹执行函数

返回 err_code

返回类型 M!

Function M! xyzGetObjPoseType()

获取工件姿态类型

返回的工件位姿类型存放在 M_04

返回 err_code

返回类型 M!

Function M! xyzResetPalletStatus()

重置工业码垛状态

返回 err_code

返回类型 M!

Function M! xyzSwitchItem(MWsId, CItemCodeName$)

切换工件

参数

- **MWsId$** (*M*) –工作空间 id
- **CItemCodeName$** (字符串) –工件代号

返回 err_code

返回类型 M!

Function M! xyzCalculateGraspPose(MWsId)

计算抓取目标点位

等价于执行:xyzReqGraspPose + xyzGetGraspPose

返回的 grasp_pose 存放在 P_00, grasp_pose_num 存放在 M_01, pipeline_num 存放在 M_02, register_num 存放在 M_03

参数 MWsId$ (*M*) –工作空间 id

返回 err_code

返回类型 M!

Function M! xyzCalculateObjectPose(MWsId)

计算物体位姿

等价于执行:xyzReqGraspPose + xyzGetGraspPose

返回的 object_pose 存放在 P_00, object_pose_num 存放在 M_01, object_pose_type 存放在 M_04

参数 MWsId$ (*M*) –工作空间 id

返回 err_code

返回类型 M!

Function M! xyzUsrCmd()

用户指令

用于基础指令不支持的情况下，配合任务流图完成用户特定功能。

各个参数的含义取决于任务流图中设定的输入输出。

输入参数:

5 个字符串:C_100(1~5)

10 个 int 类型:M_103(10)

10 个 float 类型:M_104(10)

1 个 cart_pose: P_01

1 个 joints:J_01

输出参数:

5 个字符串:C_100(6~10)

10 个 int 类型:M_105(10)

10 个 float 类型:M_106(10)

1 个 cart_pose: P_02

1 个 joints:J_02

返回 err_code

返回类型 M!

其他功能 API

Function V xyzInitParams()

初始化参数，用在程序开头

返回 Void

Function V xyzExecTraj(MCntValue)

执行轨迹, 不特殊处理第一个点和最后一个点

参数 MCntValue$ (*M*) –执行轨迹时, 最大圆滑半径, 单位 mm

Function V xyzExecPickInTraj(MCntValue)

执行轨迹, 最后一个点会降速

参数 MCntValue$ (*M*) –执行轨迹时, 最大圆滑半径, 单位 mm

Function V xyzExecPickOutTraj(MCntValue)

执行轨迹, 第一个点会降速

参数 MCntValue$ (*M*) –执行轨迹时, 最大圆滑半径, 单位 mm

案例/模板说明

请注意模板函数并不能直接运行，请一定根据项目现场环境和流程需求进行修改

CartMove 模板

以下为机械臂主控 CartMove 模板程序，注意对工控机返回的 error_code 进行判断。

CartMove 基本模板：XYZCARTBASIC.prg

```
'
'
Title "XYZ CartMove Basic Template"
'
'
'-------------------------------
' init: 初始化变量+连接
'-------------------------------
#Include "XYZUTILS"
Def Act 3, M_Err > 0 GoSub *LSYSERR
Act 3 = 1
'
xyzInitParams()
xyzConnect("192.168.0.10",11111)   ' 服务器IP地址和端口号
'
'-------------------------------
' work flow: 以下为主要工作区间
'-------------------------------
'
' 切换 flow
'
M1 = xyzSwitchFlow("cart_basic.t")
If M1 <> 0 Then Hlt
'
' 切换 item
'
M1 = xyzSwitchItem(0, "item1")
If M1 <> 0 Then Hlt
'
' DO初始化，并运行到Home点
M_Out(10) = 0
Mov JHome
'
'
'
' 如果项目是眼在手上，则设置为1，否则设置为0
```

(下页继续)

(续上页)

```
MIsEyeInHand = 1
'
'
*LOOP
'
' send scan pose if in case of eye-in-hand
'
If MIsEyeInHand = 1 Then
    ' 如果是眼在手上，则
    ' 运动到拍照点并发送拍照位姿
    Mvs PScanPose
    M1 = xyzSendCurrentCartPose()
    If M1 <> 0 Then Hlt
EndIf
'
' 拍照并获取抓取点
'
M1 = xyzCalculateGraspPose(0)
If M1 <> 0 Then Hlt
' check grasp pose num
If M_01# < 1 Then
    ' no grasp pose, continue requesting
    Dly 5
    GoTo *LOOP
EndIf
'
' 运动到抓取点
'
Mvs P_00
M_Out(10) = 1
'
' 运动到示教好的放置点
'
Mvs PPlacePose
M_Out(10) = 0
'
GoTo *LOOP
'
'
'-------------------------------
' 系统报错处理
'-------------------------------
*LSYSERR:
'
' system error
'
Hlt
End
```

CartMove 二次定位模板：XYZCARTREPO.prg

```
'
'
Title "XYZ CartMove Repositioning Template"
'
'
```

(下页继续)

(续上页)

```
'------------------------------
' init: 初始化变量+连接
'------------------------------
#Include "XYZUTILS"
Def Act 3, M_Err > 0 GoSub *LSYSERR
Act 3 = 1
'
xyzInitParams()
xyzConnect("192.168.0.10",11111)  ' 服务器IP地址和端口号
'
'------------------------------
' work flow: 以下为主要工作区间
'------------------------------
'
' 切换 flow
'
M1 = xyzSwitchFlow("cart_repo.t")
If M1 <> 0 Then Hlt
'
'
M_Out(10) = 0 ' pick board
M_Out(11) = 0 ' pick item
Mov JHome
'
' 切换 item
'
M1 = xyzSwitchItem(0, "item1")
If M1 <> 0 Then Hlt
'
' 对工件粗定位拍照
'
M1 = xyzReqGraspPose(0)
If M1 <> 0 Then Hlt
MToken = M_00#
'
'
*LOOP
'
' 获取工件粗定位坐标值
'
M1 = xyzGetGraspPose(MToken)
If M1 <> 0 Then Hlt
'
' 如果识别不到工件，表示工作抓完，则抓取隔板
'
If M_01# < 1 Then ' check grasp_pose_num
    '
    ' 切换 item 到隔板
    '
    M1 = xyzSwitchItem(0, "board")
    If M1 <> 0 Then Hlt
    '
    ' 拍隔板
    '
    M1 = xyzCalculateGraspPose(0)
    If M1 <> 0 Then Hlt
```

(下页继续)

(续上页)

```
    If M_01# < 1 Then Hlt ' check grasp_pose_num
    '
    ' 抓隔板
    '
    Mvs P_00
    M_Out(10) = 1
    '
    ' 放置隔板
    '
    Mvs PBoardPlacePose
    M_Out(10) = 0
    '
    ' 切换item到工件
    '
    M1 = xyzSwitchItem(0, "item1")
    If M1 <> 0 Then Hlt
    '
    ' 对工件粗定位拍照
    '
    M1 = xyzReqGraspPose(0)
    If M1 <> 0 Then Hlt
    MToken = M_00#
    '
    GoTo *LOOP
    '
EndIf
'
'
' 二次定位
'
'
' 运动到粗定位点
'
Mvs P_00
'
' 发送拍照位姿
'
M1 = xyzSendCurrentCartPose()
If M1 <> 0 Then Hlt
'
' 切换 item
'
M1 = xyzSwitchItem(1, "item1")
If M1 <> 0 Then Hlt
'
' 对工件进行精定位拍照
'
M1 = xyzCalculateGraspPose(1)
If M1 <> 0 Then Hlt
If M_01# < 1 Then Hlt ' check grasp_pose_num
'
'
' 抓取工件
'
Mvs P_00
M_Out(11) = 1
```

(下页继续)

(续上页)

```
'
' 放置工件
Mvs PItemPlacePose
M_Out(11) = 0
'
' 切换 item
'
M1 = xyzSwitchItem(0, "item1")
If M1 <> 0 Then Hlt
'
' 对工件进行粗定位拍照
'
M1 = xyzReqGraspPose(0)
If M1 <> 0 Then Hlt
MToken = M_00#
'
'
GoTo *LOOP
'
'
'-------------------------------
' 报错处理
'-------------------------------
*LSYSERR:
'
' system error
'
Hlt
End
```

TrajMove 模板

同步模板：XYZTRAJSYNC.prg

```
'
'
Title "XYZ TrajMove Sync Template"
'
'
'-------------------------------
' init: 初始化变量+连接
'-------------------------------
#Include "XYZUTILS"
Def Act 3, M_Err > 0 GoSub *LSYSERR
Act 3 = 1
'
xyzInitParams()
xyzConnect("192.168.0.10",11111) ' 服务器IP地址和端口号
'
'-------------------------------
' work flow: 以下为主要工作区间
'-------------------------------
'
' 切换 flow
'
```

(下页继续)

(续上页)

```
M1 = xyzSwitchFlow("traj_sync.t")
If M1 <> 0 Then Hlt
'
' 切换 item
'
M1 = xyzSwitchItem(0, "item1")
If M1 <> 0 Then Hlt
'
'
M_Out(10) = 0
Mov JHome
'
'
*LOOP
'
' 请求pick place规划
'
M1 = xyzReqPickPlace(0)
If M1 <> 0 Then Hlt
'
' 获取取料入框轨迹
'
M1 = xyzGetPickIn(0)
If M1 <> 0 Then Hlt
If M_05# < 1 Then
    GoTo *CLEAREXIT
EndIf
' 执行取料入框轨迹
'   50表示走轨迹时候的圆滑半径，单位是mm
xyzExecPickInTraj(50)
M_Out(10) = 1
'
' 获取取料出框轨迹
'
M1 = xyzGetPickOut(0)
If M1 <> 0 Then Hlt
' 执行取料出框轨迹
xyzExecPickOutTraj(50)
'
' 获取放料入框轨迹
'
M1 = xyzGetPlaceIn(0)
If M1 <> 0 Then Hlt
' 执行放料入框轨迹
xyzExecTraj(50)
M_Out(10) = 0
'
' 获取放料出框轨迹
'
M1 = xyzGetPlaceOut(0)
If M1 <> 0 Then Hlt
' 执行放料出框轨迹
xyzExecTraj(50)
'
'
GoTo *LOOP
```

(下页继续)

(续上页)

```
'
'------------------------------
' normal exit
'------------------------------
'
*CLEAREXIT
' work done:tote has beeen cleared
Hlt
End
'
'
'------------------------------
' system error
'------------------------------
*LSYSERR:
'
' system error
'
Hlt
End
```

异步模板：XYZTRAJASYNC.prg

```
'
'
Title "XYZ TrajMove Async Template"
'
'
'------------------------------
' init: 初始化变量+连接
'------------------------------
#Include "XYZUTILS"
Def Act 3, M_Err > 0 GoSub *LSYSERR
Act 3 = 1
'
xyzInitParams()
xyzConnect("192.168.0.10",11111) ' 服务器IP地址和端口号
'
'------------------------------
' work flow: 以下为主要工作区间
'------------------------------
'
' 切换 flow
'
M1 = xyzSwitchFlow("traj_async.t")
If M1 <> 0 Then Hlt
'
' 切换 item
'
M1 = xyzSwitchItem(0, "item1")
If M1 <> 0 Then Hlt
'
'
M_Out(10) = 0
Mov JHome
'
```

(下页继续)

(续上页)

```
'
' 请求pick place规划
'
M1 = xyzReqPickPlace(0)
If M1 <> 0 Then Hlt
'
'
*LOOP
'
' 获取取料入框轨迹
'
M1 = xyzGetPickIn(0)
If M1 <> 0 Then Hlt
If M_05# < 1 Then
    GoTo *CLEAREXIT
EndIf
' 执行取料入框轨迹
'   50表示走轨迹时候的圆滑半径，单位是mm
xyzExecPickInTraj(50)
M_Out(10) = 1
'
' 获取取料出框轨迹
'
M1 = xyzGetPickOut(0)
If M1 <> 0 Then Hlt
' 执行取料出框轨迹
xyzExecPickOutTraj(50)
'
' 提前请求下一次的请求pick place规划
'
M1 = xyzReqPickPlace(0)
If M1 <> 0 Then Hlt
'
' 获取放料入框轨迹
'
M1 = xyzGetPlaceIn(0)
If M1 <> 0 Then Hlt
' 执行放料入框轨迹
xyzExecTraj(50)
M_Out(10) = 0
'
' 获取放料出框轨迹
'
M1 = xyzGetPlaceOut(0)
If M1 <> 0 Then Hlt
' 执行放料出框轨迹
xyzExecTraj(50)
'
'
GoTo *LOOP
'
'-------------------------------
' normal exit
'-------------------------------
'
*CLEAREXIT
```

(下页继续)

(续上页)

```
' work done:tote has beeen cleared
Hlt
End
'
'
'-------------------------------
' system error
'-------------------------------
*LSYSERR:
'
' system error
'
Hlt
End
```

常见问题

1. 重连 max 后连不上

   可能是机器人非正常退出, 尝试重新启动机器人

附录

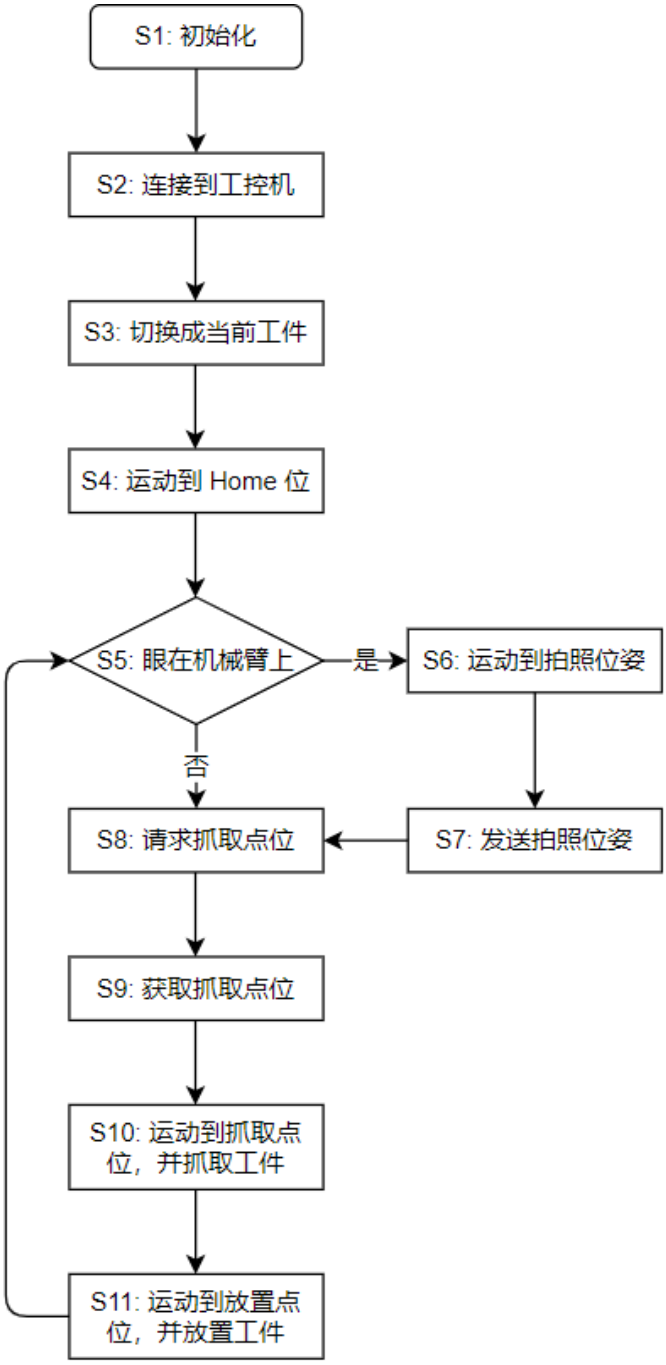
14.3 机械臂主控模板

针对工业机械臂主控应用，每种机械臂提供了四种模板函数分别是坐标移动基础模板，坐标移动二次定位模板，轨迹移动同步模板，轨迹移动异步模板。

每种模板流程图中节点说明文字上的 "S+ 数字" 表示该节点的步骤编号，例如 S4(Step4), 表示第四步。步骤编号也会在每种机械臂的模板函数中使用，方便机械臂代码的理解。

14.3.1 坐标移动基础模板

坐标移动基础模板包含了眼在手上和眼在手外两种形式的基础运行逻辑。其流程图如下所示



流程步骤说明：

S1. 首先执行初始化操作，初始整个流程需要的变量，以及 IO 信号等；

S2. 调用连接接口，在工控机服务已经打开的前提下连接到工控机服务端；

S3. 向工控机服务端发送切换工件指令，切换成当前工作的工件；

S4. 使用机器人的运动指令运动到定义好的 Home 点，待机器人运动到 Home 点之后便开始工作循环；

S5. 进入到工作循环后首先判断应用形式是否是眼在机械臂上的应用，如果是眼在机械臂上跳转到 S6 ，如果是眼在手外的机械臂应用跳转到 S8 ;

S6. 使用机器人的运动指令运动到预先定义好的拍照位姿，然后跳转到 S7 ；

S7. 待机器人运行到拍照位姿后发送拍照位姿给工控机服务端；

S8. 请求抓取点位

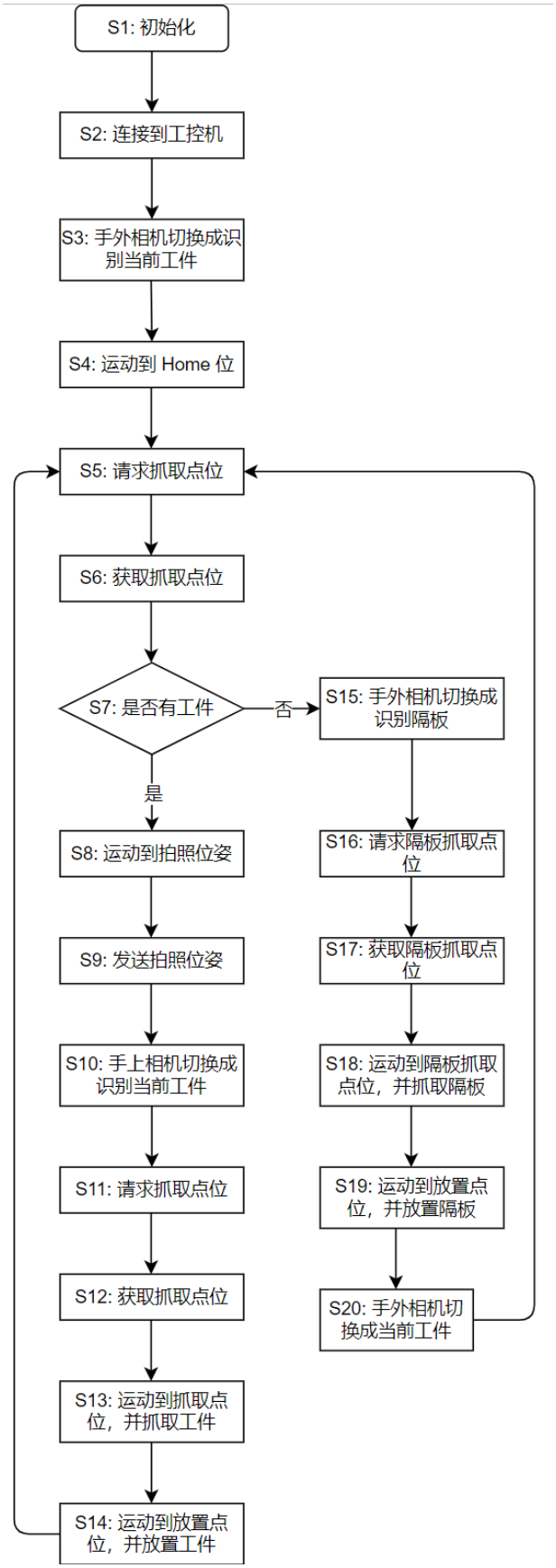
S9. 获取抓取点位

S10. 得到抓取位姿后，使用机器人的运动指令使机器人运动到抓取位姿并进行抓取 (可以插入一些过渡点位防止碰撞)；

S11. 抓取完成后，控制机械臂运动到放置位，放置工件 (可以插入一些过渡点位防止碰撞)，待放下工件后便可跳转到 S5 开始下一个循环。

14.3.2 坐标移动二次定位模板

该模板对应的情形是：环境中有两个相机，一个固定在机械臂外，一个固定在机械臂上，工件成层摆放，每层之间有隔板分割。固定在机械臂外的相机用于识别工作空间中是否有工件，识别到有工件后再移动机械臂靠近工件使用固定在机械臂上的相机对工件进行精定位抓取。如果没有识别到工件则需先移除隔板。其流程图如下所示


S1: 初始化
S2: 连接到工控机
S3: 手外相机切换成识别当前工件
S4: 运动到 Home 位
S5: 请求抓取点位
S6: 获取抓取点位
S7: 是否有工件
否
是
S15: 手外相机切换成识别隔板
S16: 请求隔板抓取点位
S17: 获取隔板抓取点位
S18: 运动到隔板抓取点位，并抓取隔板
S19: 运动到放置点位，并放置隔板
S20: 手外相机切换成当前工件
S8: 运动到拍照位姿
S9: 发送拍照位姿
S10: 手上相机切换成识别当前工件
S11: 请求抓取点位
S12: 获取抓取点位
S13: 运动到抓取点位，并抓取工件
S14: 运动到放置点位，并放置工件

流程步骤说明：

S1. 首先执行初始化操作，初始整个流程需要的变量，以及 IO 信号等；

S2. 调用连接接口, 在工控机服务已经打开的前提下连接到工控机服务端；

S3. 向工控机服务端发送切换工件指令，把机械臂外的相机切换成识别当前工作的工件；

S4. 使用机器人的运动指令运动到定义好的 Home 点, 待机器人运动到 Home 点之后便开始工作循环；

S5. 请求手外相机识别到的工件抓取位姿；

S6. 获取手外相机识别到的工件抓取位姿，以及工件个数；

S7. 判断手外相机是否识别到工件，如果识别到工件跳转到 S8 , 如果没有识别到工件跳转到 S15 ；

S8. 手外相机已经识别到工件，为了更精确定位工件位姿需要使用手上相机对工件重新拍照定位，运动到手上相机的拍照位姿处，拍照位姿可以根据手外相机提供的工件位姿来确定；

S9. 发送当前的拍照位姿；

S10. 手上相机切换成识别当前工件；

S11. 请求手上相机识别到的工件抓取位姿；

S12. 获取手上相机识别到的工件抓取位姿；

S13. 得到抓取位姿后，使用机器人的运动指令使机器人运动到抓取位姿并进行抓取 (可以插入一些过渡点位防止碰撞);

S14. 抓取完成后，控制机械臂运动到放置位，放置工件 (可以插入一些过渡点位防止碰撞)，待放下工件后便可跳转到 S5 开始下一个循环；

S15. 如果手外相机没有识别到工件，可能是工件被隔板所遮挡，需要先移除隔板。手外相机切换成识别隔板

S16. 请求手外相机识别隔板的抓取位姿；

S17. 获取手外相机识别到的隔板抓取位姿；

S18. 得到隔板抓取位姿后，使用机器人的运动指令使机器人运动到隔板抓取位姿并进行抓取 (可以插入一些过渡点位防止碰撞);

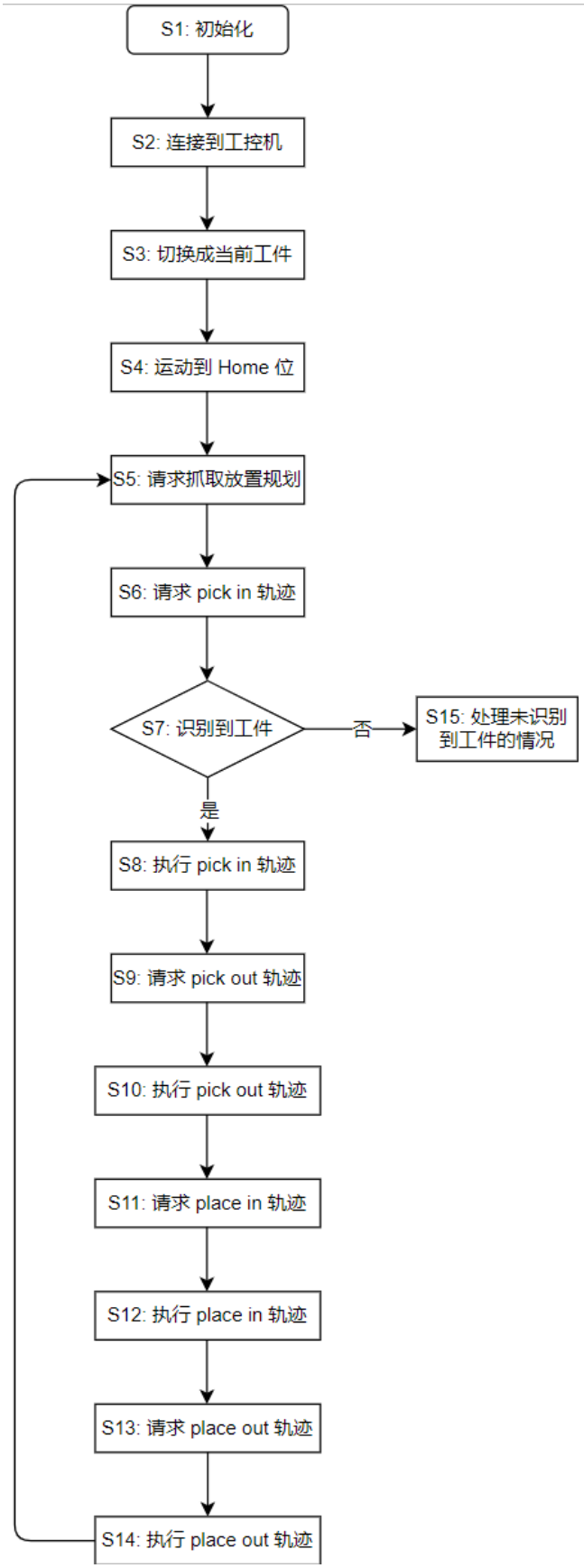
S19. 隔板抓取完成后，控制机械臂运动到隔板的放置位，放置隔板 (可以插入一些过渡点位防止碰撞)；

S20. 待放下隔板后，手外相机切换成识别当前工件，便可跳转到 S5 开始下一个循环；

在介绍轨迹移动模板之前，需要简单说明轨迹移动的应用场景和一些基本概念。在轨迹移动中，一般情况下机械臂需要走的路径点均由工控机服务端根据机械臂的工作环境计算得来。对于一般简单双筐的应用场景，工控机服务端提供的路径点位分成四个部分，分别是 pick in , pick out , place in , place out 。其中 pick in 是指抓取抓取入筐路径，从定义好的抓取入筐位到工件的抓取位；pick out 是指抓取出筐路径，从抓取点位运动到出筐位；place in 是指放置入筐轨迹，从预先定义好的入筐位到框内的放置位；place out 是指放置出筐路径，从放置位从放置出筐位。这四段路径可以根据项目需求来使用，并不一定全都需要获取并执行。

14.3.3 轨迹移动同步模板

轨迹移动同步模板即每次请求获取轨迹后，都会立即执行获取的轨迹。其流程图如下所示


S1: 初始化
S2: 连接到工控机
S3: 切换成当前工件
S4: 运动到 Home 位
S5: 请求抓取放置规划
S6: 请求 pick in 轨迹
S7: 识别到工件
否
S15: 处理未识别到工件的情况
是
S8: 执行 pick in 轨迹
S9: 请求 pick out 轨迹
S10: 执行 pick out 轨迹
S11: 请求 place in 轨迹
S12: 执行 place in 轨迹
S13: 请求 place out 轨迹
S14: 执行 place out 轨迹

流程步骤说明：

S1. 首先执行初始化操作，初始整个流程需要的变量，以及 IO 信号等；

S2. 调用连接接口, 在工控机服务已经打开的前提下连接到工控机服务端；

S3. 向工控机服务端发送切换工件指令，切换成当前工作的工件；

S4. 使用机器人的运动指令运动到定义好的 Home 点，待机器人运动到 Home 点之后便开始工作循环；

S5. 请求工控机服务端进行工件的抓取放置规划；

S6. 请求抓取的 pick in 轨迹，以及识别到的工件个数；

S7. 如果识别到工件跳转到 S8 , 如果未识别到工件跳转到 S15 ;

S8. 执行获取到的 pick in 轨迹；

S9. 获取 pick out 轨迹；

S10. 执行 pick out 轨迹；

S11. 获取 place in 轨迹；

S12. 执行 place in 轨迹；

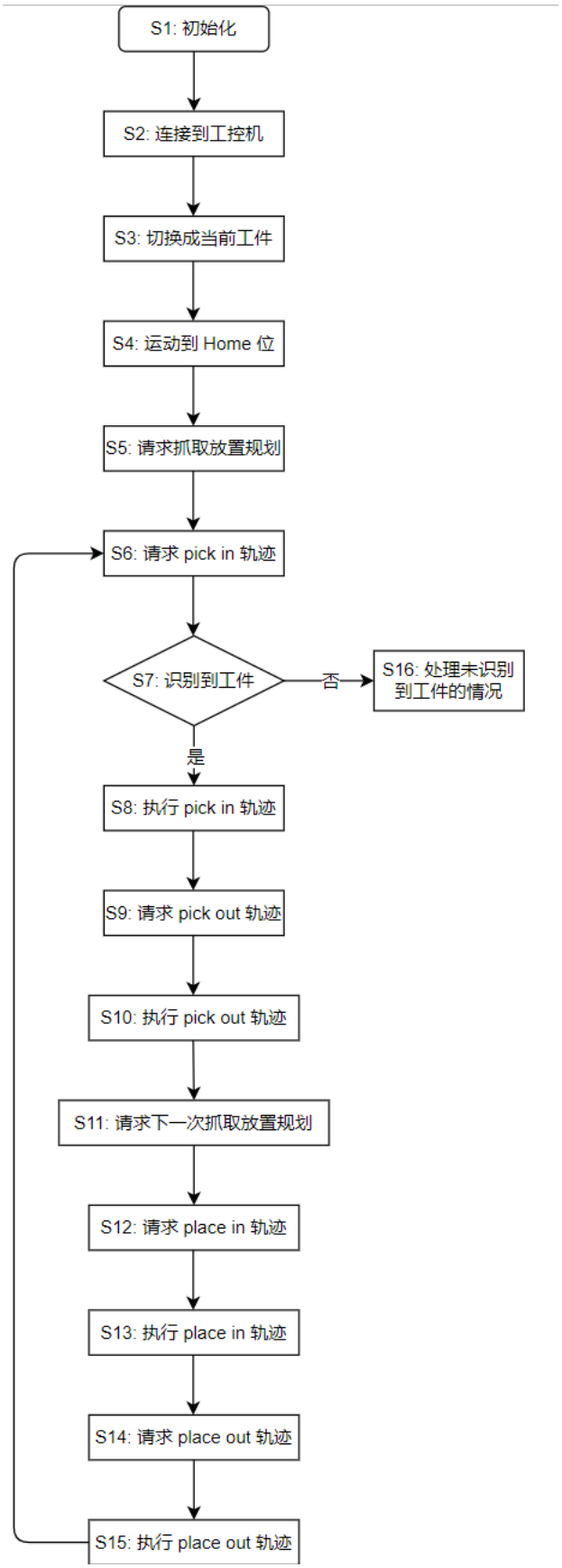
S13. 获取 place out 轨迹；

S14. 执行 place out 轨迹，执行完 place out 轨迹后跳转到 S5 开始下一次循环。

S15. 如果当前工作空间中没有识别到工件，需要根据项目需求做处理。

14.3.4 轨迹移动异步模板

轨迹移动异步模板会在抓取到工件执行完出抓取区域轨迹后开始下一次请求规划指令，然后再继续请求未执行完的放置轨迹。其流程图如下所示


S1: 初始化
S2: 连接到工控机
S3: 切换成当前工件
S4: 运动到 Home 位
S5: 请求抓取放置规划
S6: 请求 pick in 轨迹
S7: 识别到工件
否
S16: 处理未识别到工件的情况
是
S8: 执行 pick in 轨迹
S9: 请求 pick out 轨迹
S10: 执行 pick out 轨迹
S11: 请求下一次抓取放置规划
S12: 请求 place in 轨迹
S13: 执行 place in 轨迹
S14: 请求 place out 轨迹
S15: 执行 place out 轨迹

流程步骤说明：

S1. 首先执行初始化操作，初始整个流程需要的变量，以及 IO 信号等；

S2. 调用连接接口, 在工控机服务已经打开的前提下连接到工控机服务端；

S3. 向工控机服务端发送切换工件指令，切换成当前工作的工件；

S4. 使用机器人的运动指令运动到定义好的 Home 点；

S5. 请求工控机服务端进行工件的抓取放置规划，请求完之后便进入到工作循环；

S6. 请求抓取的 pick in 轨迹，以及识别到的工件个数；

S7. 如果识别到工件跳转到 S8 , 如果未识别到工件跳转到 S15 ;

S8. 执行获取到的 pick in 轨迹；

S9. 获取 pick out 轨迹；

S10. 执行 pick out 轨迹；

S11. 待执行完 pick out 轨迹，便开始请求下一次的抓取放置规划；

S12. 获取 place in 轨迹；

S13. 执行 place in 轨迹；

S14. 获取 place out 轨迹；

S15. 执行 place out 轨迹，执行完 place out 轨迹后跳转到 S5 开始下一次循环。

S16. 如果当前工作空间中没有识别到工件，需要根据项目需求做处理。

14.4 驱动连接故障排查

1. 检查机器人是否开通了 socket 通讯功能
2. ping 机器人。如 ping 不通，排查本地 ip、机器人 ip，网线是否连接正常
3. 是否按照驱动安装说明安装，确认运行了正确的文件
4. 检查下位机代码版本，应该和上位机代码版本一致才能连接
5. 查看 windows defender 是否打开，若已打开，在左边菜单栏找到允许应用通过 Windows Defender 防火墙进行通信，查看 robot driver node 的专用和公用是否都已勾选
6. 查看 windows 安全中心，域网络防火墙是否打开，需要关闭。
7. 查看 robot_config.yaml 的 timeout 时间，是否过短。需要注意的是 motion_socket 的 timeout 的第二个参数和 status_socket 的 timout 的第二个参数。这里单位是毫秒。
8. windows 的任务管理器详细信息，查看 robot driver node 是否多开
9. 按照整站或非整站通讯协议，尝试用 socket 调试工具连接机器人

CHAPTER 15

PLC 连接

目前支持的 PLC 连接是机械臂主控（*Robot Master*） 模式，暂不支持工控机主控（*IPC Master*） 模式。

15.1 已支持的 PLC 通讯协议

15.1.1 modbus_tcp

此处介绍使用 Modbus_TCP 的相关事项。

modbus_tcp 介绍

协议介绍

ModbusTcp 服务器 Server（即 Modbus Slave）	PLC
ModbusTcp 客户端 Client（即 Modbus Master）	上位机 PC：XYZ Studio Max
IP 地址	PLC 和 XYZ Studio Max 同一网段
端口	502（可配置）
响应周期	> 200ms

程序运行基本逻辑：

1. PLC 是主动请求指令的一端，通信采用周期轮寻的方式进行。
2. PLC 根据需要，在合适时机发送相关指令给 PC（如拍照、获取点位等），PC 根据 PLC 的命令执行相应功能并返回信息。
3. 为了保持长链接，PC 和 PLC 中间会有个定期运行的心跳包。

交互时序

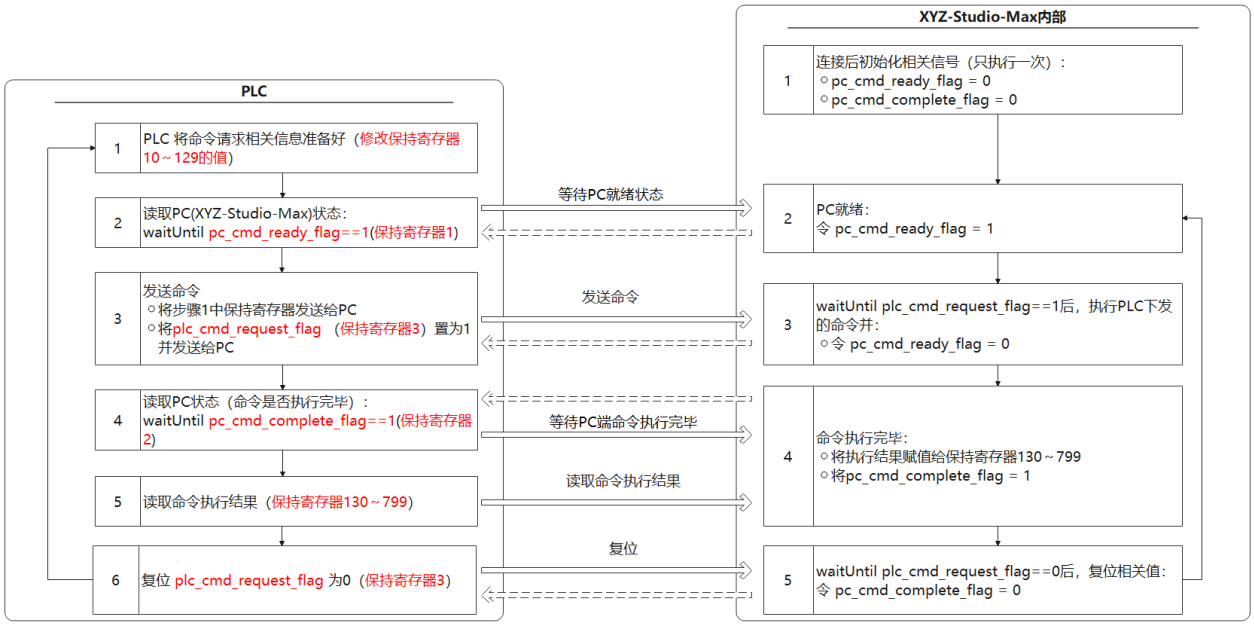


图 1: 交互时序（可放大查看）

阅读以上时序图的要点：

1. 中间实线箭头表示 PLC 向 PC 进行相关数据交互，虚线箭头表示 PC(XYZ-Studio-Max) 向 PLC 进行相关数据交互。
2. 第一次阅读时，可以主要从 plc 角度进行，而暂时忽略掉 pc 端内部细节，因为 plc 是逻辑主控。也就是可以主要查看左边 PLC 内部流程以及中间由 PLC 发起的通信部分（实线箭头部分）。
3. 上图中涉及信号复位。（本质是上升沿触发）
4. 请严格按照时序图进行信号设置和收发，否则会有时序混乱的风险。

api 接口说明

保持寄存器格式

本文所有关于 WORD, DWORD 等的数据格式描述, 均按下表所述约定。用户需要自己分辨这种描述和特定 PLC 厂家的区别并进行相应转换。

数据类型	描述
输入线圈 (Input Coils / Discrete Inputs)	只读的 BOOL 型变量 (1-bit)
保持线圈 (Holding Coils / Coils of Outputs)	可读可写的 BOOL 型变量 (1-bit)
输入寄存器 (Input Registers / Input Data)	可读可写的 WORD 型 (16-bit) 型变量
保持寄存器 (Holding Registers / Output Data)	可读可写的 WORD 型 (16-bit) 型变量

本协议只用到了 **保持寄存器**，没有用到其他寄存器或线圈。

PLC 方需要事先告知/确定其单个保持寄存器 (WORD) 和双保持寄存器 (DWORD) 的符号位和字节序，PC(XYZ-Studio-Max) 需要做相应的配置，否则数据解析会不正确。

单个保持寄存器 WORD		
符号位	有符号/无符号	这个必须是有符号的 WORD 即数据范围支持-32768 ~ 32767，因为涉及传递负数。
字节序	大端/小端/大端交换/小端交换	字节序定义见下文 根据 PLC 实际情况配置上位机 XYZ-Studio-MaX 即可

双保持寄存器 DWORD		
符号位	有符号/无符号	这个必须是有符号的 DWORD 即数据范围支持-2,147,483,648 ~ 2,147,483,647，因为涉及传递负数。
字节序	大端/小端/大端交换/小端交换	根据 PLC 实际情况配置上位机 XYZ-Studio-Max 即可

字节序的定义如下：

```
class ByteOrder(Enum):
"""
                          2Bytes(word)     4Bytes(dword/float)      8Bytes(double/4word)
    BIG_ENDIAN              AB                  AB CD                   AB CD EF GH
    LITTLE_ENDIAN           BA                  DC BA                   HG FE DC BA
    BIG_ENDIAN_SWAP         BA                  BA DC                   BA DC FE HG
    LITTLE_ENDIAN_SWAP      AB                  CD AB                   GH EF CD AB
"""
BIG_ENDIAN = 0                  # 大端
LITTLE_ENDIAN = 1               # 小端
BIG_ENDIAN_SWAP = 2             # 大端交换
LITTLE_ENDIAN_SWAP = 3          # 小端交换
```

交互标志位寄存器

	数据内容	保持寄存器地址	说明
1	heart_beat_flag 心跳检测位	0	通讯连接后，内部会定期执行 pc → plc 内部自动定期循环发送 0-1-0-1-… (周期 1s)
2	pc_cmd_ready_flag (PC command ready flag)	1	plc 发送命令前，需要查询此状态： • 为 1，表示此时 pc 准备好接收来自 plc 的命令，plc 可以发送命令。 • 为 0，表示此时 pc 为还未准备好接收来自 plc 的命令，plc 不要发送相关命令。
3	pc_cmd_complete_flag (PC command complete flag)	2	plc 发送命令后，需要查询此状态看 pc 是否命令执行完毕： • 为 1，表示此时 pc 端已经执行完 plc 发送的命令。 • 为 0，表示此时 pc 端处于命令执行的 busy 状态。
4	plc_cmd_request_flag (plc command request flag)	3	plc 发送给 pc 的命令标志位： • 为 1，表示让 pc 开始执行命令。 • plc 需要根据时序要求，进行复位为 0。
5	预留	4~9	占用，预留

PLC 发送给 PC 的寄存器

数据内容	保持寄存器地址	说明
cmd	10	命令码，支持拍照、获取点位、计算轨迹等。 各指令详细信息见下文
vision_service_id	11	视觉服务 id
ws_id	12	工作空间 id
token	13	整数值，某些命令需要用到该值

下页继续

表 1 – 续上页

item_codename	14	工件代号
app_name	15	项目名称
flow_name	16	task flow 的名称
object_name	17	工件名称（或者叫 object_id）
tool_name	18	工具名称（或者叫 tool_id）
strategy_name	19	策略名称
in_str_5	25~29	命令码 "用户指令" 中用到，具体查看下文的 "用户指令"
in_word_10	30~39	
in_float_10	40~59	
cart_pose:x	60~61	占用 2 个保持寄存器，数据解析规则是有符号类型的 DWORD 案例: • 想传递的真实数据是: -1234.56，默认小数点位数为 2 位 • 则数据 *100 后取整为 -123456，该值的 16 进制是: FF FE 1D C0 • [60] → FF FE • [61] → 1D C0 说明: • 小数点位数可以在上位机进行配置 • [60] 和 [61] 中字节顺序需要变化的，可以在上位机中进行配置。如 [60] -> FE FF, [61] -> C0 1D
cart_pose:y	62~63	同 cart_pose:x
cart_pose:z	64~65	同 cart_pose:x
cart_pose:a	66~67	同 cart_pose:x
cart_pose:b	68~69	同 cart_pose:x
cart_pose:c	70~71	同 cart_pose:x
cart_pose:d	72~73	同 cart_pose:x
joint:j1	74~75	同 cart_pose:x
joint:j2	76~77	同 cart_pose:x
joint:j3	78~79	同 cart_pose:x
joint:j4	80~81	同 cart_pose:x
joint:j5	82~83	同 cart_pose:x
joint:j6	84~85	同 cart_pose:x
系统预留	86~99	请勿使用
用户自定义区	100~129	用户自定义区

PLC 接收自上位机反馈的寄存器

数据内容	保持寄存器地址	说明
error	130	• 0：没有错误 • 1：有错误
error_code	131~132	具体的错误代码 占用 2 个保持寄存器, 数据解析规则是有符号类型的 DWORD
token	133	请求某些异步指令时会返回的值, 用户可以使用该值来获取对应的异步结果
grasp_pose_num	134	抓取点位的数量
object_pose_num	135	物体/工件点位的数量
pipeline_num	136	pipeline 文件的 number 值
register_num	137	用到的注册文件的注册 number 值
object_pose_type	138	物体/工件点位的类型
系统预留	139~159	请勿使用
cart_pose:x	160~161	根据 plc 的请求命令情况，可以是抓取点位姿 (grasp_pose) 或者物体位姿 (object_pose) 等。 数据格式同之前的 cart_pose:x
cart_pose:y	162~163	同 cart_pose:x
cart_pose:z	164~165	同 cart_pose:x
cart_pose:a	166~167	同 cart_pose:x
cart_pose:b	168~169	同 cart_pose:x
cart_pose:c	170~171	同 cart_pose:x
cart_pose:d	172~173	同 cart_pose:x
joint:j1	174~175	同 cart_pose:x
joint:j2	176~177	同 cart_pose:x
joint:j3	178~179	同 cart_pose:x
joint:j4	180~181	同 cart_pose:x
joint:j5	182~183	同 cart_pose:x
joint:j6	184~185	同 cart_pose:x
trajectory_wap_point_num	186	轨迹中的点位数量
系统预留	187~189	请勿使用

下页继续

表 2 – 续上页

trajectory_way_point_type	190~219	存放轨迹中的各个点位和点位的类型，具体分类如下： wp1_type: [190] wp1: data1: [220, 221] data2: [222, 223] data3: [224, 225] data4: [226, 227] data5: [228, 229] data6: [230, 231] wp2_type: [191] wp2: data1: [232, 233] data2: [234, 235] data3: [236, 237] data4: [238, 239] data5: [240, 241] data6: [242, 243] … wp_type: 可能是如下值： • 11：joints movej，之后的 6 个 data 就是 joints • 12：joints movel，之后的 6 个 data 就是 joints • 21：Cart movej，之后的 6 个 data 就是 cartesian • 22：Cart movel，之后的 6 个 data 就是 cartesian
trajectory_way_point	220~579	参考上面 “trajectory_way_point_type” 中的说明
out_str_5	585~589	命令码 “用户指令” 中用到，具体查看下文的 “用户指令”
out_word_10	590~599	
out_float_10	600~619	
用户自定义区	620~799	用户自定义区

支持的指令

心跳信号 (xyzHeartBeat)	
功能描述	通讯连接后，PC 自动定期循环发送 0-1-0-1-…给 PLC，周期 1s

切换应用 (xyzSwitchApp)			
功能描述	暂不支持		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 501
	app_name	15	需要切换到的 app 名字
plc 收到返回值相关寄存器	error	130	0：没有错误，1：有错误
	error_code	131~132	具体的错误代码 占用 2 个保持寄存器, 数据解析规则是有符号类型的 DWORD

切换 Flow(xyzSwitchFlow)			
功能描述	切换任务的 flow(如“1.t”)		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 502
	flow_name	16	task flow 名称填写的是 max 上映射后的 task flow 名，后缀”.t”不用填
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	

切换工具 (xyzSwitchTool)			
功能描述	暂不支持		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 503
	tool_name	18	工具名称
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	

请求拍照 (xyzReqCapImg)			
功能描述	异步请求拍照		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 504
	vision_service_id	11	视觉服务 id
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	
	token	133	可以在获取拍照结果 (xyzGetCapImg) 指令中使用该值，来查询本次拍照是否成功

获取拍照结果 (xyzGetCapImg)			
功能描述	获取异步拍照结果。使用该指令前需先调用请求拍照 (xyzReqCapImg)		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 505
	token	13	来自于用户的某一次请求拍照 (xyzReqCapImg) 返回的 token 值
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	

拍照 (xyzCapImg)			
功能描述	等价于执行：请求拍照 (xyzReqCapImg) + 获取拍照结果 (xyzGetCapImg)		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 506
	vision_service_id	11	视觉服务 id
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	

请求抓取目标点位 (xyzReqGraspPose)			
功能描述	异步请求目标抓取点的位姿		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 507
	ws_id	12	工作空间 id
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	
	token	133	可以在获取抓取目标点位 (xyzGetGraspPose) 指令中使用该值，来查询本次抓取点的值

获取抓取目标点位 (xyzGetGraspPose)			
功能描述	获取目标抓取点的位姿。使用该指令前需先调用请求抓取目标点位 (xyzReqGraspPose)		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 508
	token	13	来自于用户的某一次请求抓取目标点位 (xyzReqGraspPose) 返回的 token 值
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	
	grasp_pose_num	134	抓取点位的数量
	pipeline_num	136	pipeline 文件 number 值
	register_num	137	用到的注册文件的注册 number 值
	cart_pose:x	160~161	抓取点位姿的 x 值 占用 2 个保持寄存器，案例参考本说明书上文中的说明
	cart_pose:y	162~163	同 cart_pose:x
	cart_pose:z	164~165	同 cart_pose:x
	cart_pose:a	166~167	同 cart_pose:x
	cart_pose:b	168~169	同 cart_pose:x
	cart_pose:c	170~171	同 cart_pose:x
	cart_pose:d	172~173	同 cart_pose:x 如果机器人是欧拉角形式，则 d 用不到

请求物体位姿 (xyzReqObjPose)			
功能描述	异步请求物体位姿		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 509
	ws_id	12	工作空间 id
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	
	token	133	可以在获取物体位姿 (xyzGetObjPose) 指令中使用该值，来查询本次物体位姿

获取物体位姿 (xyzGetObjPose)			
功能描述	获取物体位姿。使用该指令前需先调用请求物体位姿 (xyzReqObjPose)		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 510
	token	13	来自于用户的某一次请求物体位姿 (xyzReqObjPose) 返回的 token 值
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	
	object_pose_num	135	物体点位的数量
	object_pose_type	138	物体点位的类型
	cart_pose:x	160~161	物体点位的 x 值 占用 2 个保持寄存器，案例参考本说明书上文中的说明
	cart_pose:y	162~163	同 cart_pose:x
	cart_pose:z	164~165	同 cart_pose:x
	cart_pose:a	166~167	同 cart_pose:x
	cart_pose:b	168~169	同 cart_pose:x
	cart_pose:c	170~171	同 cart_pose:x
	cart_pose:d	172~173	同 cart_pose:x 如果机器人是欧拉角形式，则 d 用不到

重置任务 (xyzResetTask)			
功能描述	重置任务，一般用来初始化任务内部变量		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 511
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	

发送机器人当前角度 (xyzSendCurrentJoints)			
功能描述	发送机器人当前关节角给 PC		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 512
	joint:j1	74~75	同 cart_pose:x
	joint:j2	76~77	同 cart_pose:x
	joint:j3	78~79	同 cart_pose:x
	joint:j4	80~81	同 cart_pose:x
	joint:j5	82~83	同 cart_pose:x
	joint:j6	84~85	同 cart_pose:x
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	

发送机器人当前 Cartesian(xyzSendCurrentCartPose)			
功能描述	发送机器人当前笛卡尔位姿给 PC		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 513
	cart_pose:x	60~61	同上
	cart_pose:y	62~63	同 cart_pose:x
	cart_pose:z	64~65	同 cart_pose:x
	cart_pose:a	66~67	同 cart_pose:x
	cart_pose:b	68~69	同 cart_pose:x
	cart_pose:c	70~71	同 cart_pose:x
	cart_pose:d	72~73	如果机器人是欧拉角形式，则只需给 a,b,c 赋值，d 赋值 0 即可
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	

发送机器人当前扩展轴位置 (xyzSendCurrentExtJoints)			
功能描述	发送机器人扩展轴/附加轴的关节值给 PC		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 514
	joint:j1	74~75	同 cart_pose:x
	joint:j2	76~77	同 cart_pose:x
	joint:j3	78~79	同 cart_pose:x
	joint:j4	80~81	同 cart_pose:x
	joint:j5	82~83	同 cart_pose:x
	joint:j6	84~85	同 cart_pose:x
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	

请求 pick 动作规划 (xyzReqPick)			
功能描述	暂不支持		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 515
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	

请求 place 动作规划 (xyzReqPlace)			
功能描述	暂不支持		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 516
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	

请求 pick 和 place 规划 (xyzReqPickPlace)			
功能描述	请求进行 Pick 和 Place 规划		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 517
	ws_id	12	工作空间 id
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	

获取取料入框轨迹 (xyzGetPickin)			
功能描述			
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 518
	ws_id	12	工作空间 id
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	
	pipeline_num	136	pipeline 文件 number 值
	register_num	137	用到的注册文件的注册 number
	trajectory_wap_point_num	186	轨迹中的点位数量
	各个轨迹点类型和具体的值	190~219	每个点位的类型，具体参考上面 PLC 接受自上位机反馈的寄存器
		220~579	各个点位的值，具体参考上面 PLC 接受自上位机反馈的寄存器

获取取料出框轨迹 (xyzGetPickout)			
功能描述			
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 519
	ws_id	12	工作空间 id
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	
	pipeline_num	136	pipeline 文件 number 值
	register_num	137	用到的注册文件的注册 number
	trajectory_wap_point_num	186	轨迹中的点位数量
	各个轨迹点类型和具体的值	190~219	每个点位的类型，具体参考上面 PLC 接受自上位机反馈的寄存器
		220~579	各个点位的值，具体参考上面 PLC 接受自上位机反馈的寄存器

获取放料入框轨迹 (xyzGetPlacein)			
功能描述			
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 520
	ws_id	12	工作空间 id
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	
	pipeline_num	136	pipeline 文件 number 值
	register_num	137	用到的注册文件的注册 number
	trajectory_wap_point_num	186	轨迹中的点位数量
	各个轨迹点类型和具体的值	190~219	每个点位的类型，具体参考上面 PLC 接受自上位机反馈的寄存器
		220~579	各个点位的值，具体参考上面 PLC 接受自上位机反馈的寄存器

获取放料出框轨迹 (xyzGetPlaceout)			
功能描述			
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 521
	ws_id	12	工作空间 id
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	
	pipeline_num	136	pipeline 文件 number 值
	register_num	137	用到的注册文件的注册 number
	trajectory_wap_point_num	186	轨迹中的点位数量
	各个轨迹点类型和具体的值	190~219	每个点位的类型，具体参考上面 PLC 接受自上位机反馈的寄存器
		220~579	各个点位的值，具体参考上面 PLC 接受自上位机反馈的寄存器

请求切换策略 (xyzSwitchStrat)			
功能描述			
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 522
	strat_name	19	策略名称
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	

料箱重定位 (xyzUpdateTotePose)			
功能描述	执行完毕后，环境中的料箱位姿会被自动更新		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 523
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	
	cart_pose:x	160~161	目前该值用不到，用户无需使用
	cart_pose:y	162~163	
	cart_pose:z	164~165	
	cart_pose:a	166~167	
	cart_pose:b	168~169	
	cart_pose:c	170~171	
	cart_pose:d	172~173	

工件在手上的二次定位 (xyzUpdateObjPoseOnHand)			
功能描述	暂不支持		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 524
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	

工件不在手上的二次定位 (xyzUpdateObjPoseToHand)			
功能描述	暂不支持		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 525
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	
	pipeline_num	136	pipeline 文件 number 值
	register_num	137	用到的注册文件的注册 number
	trajectory_wap_point_num	186	轨迹中的点位数量
	各个轨迹点类型和具体的值	190~219	每个点位的类型，具体参考上面 PLC 接受自上位机反馈的寄存器
		220~579	各个点位的值，具体参考上面 PLC 接受自上位机反馈的寄存器

获取工件姿态类型 (xyzGetObjPoseType)			
功能描述	暂不支持		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 526
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	
	object_pose_type	138	物体/工件的 pose 类型信息

重置工业码垛状态 (xyzResetPalletStatus)			
功能描述	暂不支持		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 527
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	

切换工件 (xyzSwitcItem)			
功能描述	切换工件		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 528
	ws_id	12	工件空间 id
	item_codename	14	工件代号
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	

计算抓取目标点位 (xyzCalculateGraspPose)			
功能描述	等价于请求抓取目标点位 (xyzReqGraspPose) + 获取抓取目标点位 (xyzGetGraspPose)		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 529
	ws_id	12	工作空间 id
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	
	grasp_pose_num	134	抓取点位的数量
	pipeline_num	136	pipeline 文件 number 值
	register_num	137	用到的注册文件的注册 number 值
	cart_pose:x	160~161	抓取点位姿的 x 值 占用 2 个保持寄存器，案例参考本说明书上文中的说明
	cart_pose:y	162~163	同 cart_pose:x
	cart_pose:z	164~165	同 cart_pose:x
	cart_pose:a	166~167	同 cart_pose:x
	cart_pose:b	168~169	同 cart_pose:x
	cart_pose:c	170~171	同 cart_pose:x
	cart_pose:d	172~173	同 cart_pose:x 如果机器人是欧拉角形式，则 d 用不到

计算物体位姿 (xyzCalculateObjectPose)			
功能描述	等价于请求物体位姿 (xyzReqObjPose) + 获取物体位姿 (xyzGetObjPose)		
PLC 输入设置值相关寄存器	数据内容	寄存器地址	说明
	cmd	10	用户需要填 530
	ws_id	12	工作空间 id
plc 收到返回值相关寄存器	error	130	同上
	error_code	131~132	
	object_pose_num	135	物体点位的数量
	object_pose_type	138	物体点位的类型
	cart_pose:x	160~161	物体点位的 x 值占用 2 个保持寄存器，案例参考本说明书上文中的说明
	cart_pose:y	162~163	同 cart_pose:x
	cart_pose:z	164~165	同 cart_pose:x
	cart_pose:a	166~167	同 cart_pose:x
	cart_pose:b	168~169	同 cart_pose:x
	cart_pose:c	170~171	同 cart_pose:x
	cart_pose:d	172~173	同 cart_pose:x 如果机器人是欧拉角形式，则 d 用不到

用户指令 (xyzUsrCmd)				
功能描述	用户自定义指令，用于基础指令不支持的情况下，配合任务流图完成用户特定功能 各个参数的含义取决于任务流图中设定的输入输出			
PLC 输入设置值相关寄存器	数据内容		寄存器地址	说明
	cmd		10	用户需要填 600
	in_str_5	[0]	25	5 个“字符串”输入 这里的“字符串”只能是 word 类型的数字
		[1]	26	
		[2]	27	
		[3]	28	
		[4]	29	
	in_word_10	[0]	30	10 个 word 类型的数值输入
		[1]~[8]	31~38	
		[9]	39	
	in_float_10	[0]	40~41	10 个“浮点数”输入 这里所谓的“浮点数”，同前面所述的 cart_pose 的 x/y/z/a/b/c/d。 每个值的类型为 dword，占 2 个保持寄存器。 如第一个值中想传递的真实数值是-1234.56，则 [0] 的值需要填入-123456，默认保留两位小数
		[1]~[8]	42~57	
		[9]	58~59	
	cart_pose	x	60~61	1 个“笛卡尔坐标值”输入，解析方式同上文
		y	62~63	

下页继续

表 3 – 续上页

		z	64~65	
		a	66~67	
		b	68~69	
		c	70~71	
		d	72~73	
	joints	j1	74~75	1 个“关节角”输入，解析方式同上文
		j2	76~77	
		j3	78~79	
		j4	80~81	
		j5	82~83	
		j6	84~85	
plc 收到返回值相关寄存器	error		130	同上
	error_code		131~132	
	out_str_5	[0]	585	5 个“字符串”输出这里的“字符串”是 word 类型的数字
		[1]	586	
		[2]	587	
		[3]	588	
		[4]	589	
	out_word_10	[0]	590	10 个 word 类型的数值输出
		[1]~[8]	591~598	
		[9]	599	
	out_float_10	[0]	600~601	10 个“浮点数”输出 解析方式同上
		[1]~[8]	602~617	
		[9]	618~619	
	cart_pose	x	160~161	1 个“笛卡尔坐标值”输出，解析方式同上文
		y	162~163	
		z	164~165	
		a	166~167	
		b	168~169	
		c	170~171	
		d	172~173	
	joints	j1	174~175	1 个“关节角”输出，解析方式同上文
		j2	176~177	
		j3	178~179	
		j4	180~181	
		j5	182~183	
		j6	184~185	

通讯配置

通讯参数配置可以在 XYZ-Studio-Max 中进行操作，设置界面如下：

图 2: 设置界面（可放大查看）

通讯地址设置

可以在界面中设置 PLC 的地址，端口和 PLC 的从站 ID

格式设置

关节小数点位数设置时用于约定寄存器中的 joint:j1~j6 的小数点位数。
位置/姿态小数点位数设置是用于约定寄存器中的 cart_pose:x~d 的小数点位数。
WORD/DWORD 下可以设置符号位和字节序，这里需要根据 PLC 的实际情况进行设置。
LWORD/Float/Double 这几个数据格式目前没有使用，可不设置

保持寄存器地址设置

保持寄存器地址分配参考*api 接口说明*。
“寄存器地址偏移”设置大于 0 的话，可以对默认的所有保持寄存器地址进行整体偏移。

常见问题

CHAPTER 16

通信协议

本章节介绍整工控机主控协议和机械臂主控通信协议。

两者的定义和使用场景可以参考机械臂主控（*Robot Master*），工控机主控（*IPC Master*）

16.1 工控机主控通讯协议

16.1.1 协议格式

“指令 (数据)+ 分隔符 + 数据 + 分隔符 + 数据 + 分隔符 +···+ 数据 + 分隔符 + 结束符”

发送和接收的数据格式如上所示，数据之间通过分隔符划分，最后以一个结束符来结尾。分隔符可以是英文逗号 (‘,’) 或者空格 (’ ‘)，结束符可以使’ #’ 或者’ \n’。分隔符和结束符根据机械臂的特性可以灵活调整。

16.1.2 指令

1. 获取机械臂程序版本号 (GetVersion)

工控机发送	“100,#”	100: 为获取机械臂程序版本指令号
工控机接收	“100,error_code,version”	version: 机械臂程序版本号，三位点分十进制字符串表示
		eg: “1.0.0 “

2. 发送速度数据 (SetSpeed)

工控机发送	“101,spd1,spd2,#”	101: 为设置机械臂速度指令号
		spd1: 第一个速度参数 (linear)
		spd2: 第二个速度参数 (angular)
工控机接收	“101,error_code,#”	

注：速度取值范围为 (0, 100]，百分比形式

3. 发送加速度数据 (SetAcc)

工控机发送	“102,acc1,acc2,# “	102: 设置机械臂加速度指令号
		acc1: 第一个加速度参数 (accel)
		acc2: 第二个加速度参数 (decel)
工控机接收	“102,error_code,# “	

注：加速度取值范围为 (0, 100]，百分比形式

4. 发送圆滑过渡参数 (SetZone)

工控机发送	“103,blend,#”	103: 设置机械臂圆滑过渡参数指令号
		blend: 圆滑过渡参数
工控机接收	“103,error_code,# “	

注：圆滑过渡参数取值范围为 [0, 100]，百分比形式

5. 发送工具坐标系 (TCP) 参数 (SetTool)

工控机发送	“104,x,y,z,a,b,c,d,# “	104: 设置机械臂工具坐标系指令号
工控机接收	“104,error_code,# “	

注：Cartesian 姿态数据，如果机械臂是欧拉角数据那么 a,b,c 中会存储数据，如果四元数数据那么 a,b,c,d 均有数据。发送数据的单位均已转换成机械臂所需要的形式

6. 设置数字量输出

上位机发送	“105,port,state,# “	105: 设置机器人数字量输出指令
		port: 机器人数字量端口号
		state: DO 要设置成的状态, 0 on OFF, 1 on ON
上位机接收	“105,error_code,# “	

7. 设置目标关节位置，机械臂执行 movej(SetJointsMovej)

工控机发送	“106,j1,j2,j3,j4,j5,j6,j7,j8,#”	106: SetJointsMovej 指令
		j1~j8: 为机械臂目标关节角
		如果机械臂为 6 轴机械臂，则 j1~j6 表示目标位置，j7，j8 为零
		四轴机械臂以实际机械臂适配为准
工控机接收	“106,error_code,# “	

8. 设置目标 Cartesian 位置，机械臂执行 movel(SetCartMovel)

工控机发送	“107,x,y,z,a,b,c,d,#”	107: SetCartMovel 指令
工控机接收	“107,error_code,# “	x: 目标 Cartesian 的 x 轴位置
		y: 目标 Cartesian 的 y 轴位置
		z: 目标 Cartesian 的 z 轴位置
		a: 目标 Cartesian 姿态数据的第一位
		b: 目标 Cartesian 姿态数据的第二位
		c: 目标 Cartesian 姿态数据的第三位
		d: 目标 Cartesian 姿态数据的第四位

注：Cartesian 姿态数据，如果机械臂是欧拉角数据那么 a,b,c 中会存储数据，如果四元数数据那么 a,b,c,d 均有数据。发送数据的单位均已转换成机械臂所需要的形式

9. 设置目标关节位置，机械臂执行 movel(SetJointsMovel)

工控机发送	“108,data1,data2,⋯,datan,#”	108: SetJointsMovel 指令号
		datax: 不同机械臂根据机需求的数据来发送。
工控机接收	“108,error_code,# “	

10. 设置目标 Cartesian 位置，机械臂执行 movej(SetCartMovej)

工控机发送	“109,data1,data2,⋯,datan,# “	109: SetCartMovej 指令号
		datax: 不同机械臂根据机需求的数据来发送。
工控机接收	“109,error_code,#”	

11. (MovejSequence)

工控机发送	“110,num,j11,j12,j13,j14,j15,j16,j17,j18, spd1,acc1,zone1,j21,j22,j23,⋯,#”	110: MovejSequence 指令号
		num: 表示共有几组关节位置数据
		jnx: 第 n 组关节数据，每 8 个一组
		spdn: 第 n 组关节对应的速度
		accn: 第 n 组关节对应的加速度
		zonen: 第 n 组关节对应的圆滑过渡
工控机接收	“110,error_code,#”	

12. (MovelSequence)

工控机发送	“111,num,x1,y1,z1,a1,b1,c1,d1,spd1,acc1,zone1, x2,y2,z2,a2,b2,c2,d2,spd2,acc2,zone2,⋯,#”	111: MovelSequence 指令号
		num: 表示共有几组 Cartesian 数据
		Cartesian 数据每 7 个一组
		spdn: 第 n 组位姿对应的速度
		accn: 第 n 组位姿对应的加速度
		zonen: 第 n 组位姿对应的圆滑过渡
工控机接收	“111,error_code,#”	

13. (SetJointsMovejDo)

工控机发送	“112,j1,j2,j3,j4,j5,j6,j7,j8,port,state,# “	112: SetJointsMovejDo 指令号
		jx: 八个关节数据
		port: 机械臂数字量端口号
		state: DO 要设置成的状态
工控机接收	“112,error_code,#”	

14. SetCartMovelDo

工控机发送	“113,data1,⋯,datan,port,state,#“	113: SetCartMovelDo 指令号
		datax: 实现 SetCartMovelDo 所必须的数据
		port: 机械臂数字量端口号
		state: DO 要设置成的状态
工控机接收	“113,error_code,#”	

15. SetJointsMovelDo

工控机发送	“114,data1,⋯,datan,port,state,#”	114: SetJointsMovelDo 指令号
		datax: 实现 SetJointsMovelDo 所必须的数据
		port: 机械臂数字量端口号
		state: DO 要设置成的状态
工控机接收	“114,error_code,#”	

16. SetJointsMovejGroupDo

工控机发送	“115,data1,⋯,datan,gval,#”	115: SetJointsMovejGroupDo 指令号
		datax: 实现 SetJointsMovejGroupDo 所必须的数据
		gval：组信号
工控机接收	“115,error_code,#“	

17. SetCartMovelGroupDo

工控机发送	“116,data1,⋯,datan,gval,#“	116: SetCartMovelGroupDo 指令号
		datax: 实现 SetCartMovelGroupDo 所必须的数据
		gval：组信号
工控机接收	“116,error_code,#“	

18. SetJointsMovelGroupDo

工控机发送	“117,data1,⋯,datan,gval,#“	117: SetJointsMovelGroupDo 指令号
		datax: 实现 SetJointsMovelGroupDo 所必须的数据
		gval：组信号
工控机接收	“117,error_code,#“	

19. MovelUntil

工控机发送	“118,data1,⋯,datan,port,#”	118: MovelUntil 指令号
		datax: 实现 MovelUntil 所必须的数据
		port：数字量输入端口号
工控机接收	“118,error_code,#”	

120. 获取数字量输入状态 (GetDigitalInput)

工控机发送	“119,port,#“	119: GetDigitalInput 指令号
		datax: 实现 MovelUntil 所必须的数据
		port：数字量输入端口号
工控机接收	“119,error_code,state,#“	state: 数字量输入端口号的状态 0 on OFF, 1 on ON

21. 获取数字量输出状态 (GetDigitalOutput)

工控机发送	“120,port,#“	120: GetDigitalOutput 指令号
		port: 数字量输出端口号
工控机接收	“120,error_code,state,#“	state: 数字量输出口的状态 0 on OFF, 1 on ON

22. 获取模拟量输入口数值 (GetAnalogInput)

工控机发送	“121,port,#”	121: GetAnalogInput 指令号
		port: 模拟量输入端口号
工控机接收	“121,error_code,val,#“	val: 模拟量输入口的值

23. 获取机械臂当前角度 (GetJoints)

工控机发送	“122,#“	122: GetJoints 指令号
工控机接收	“122,error_code,j1,j2,j3,j4,j5,j6,j7,j8,#“	j1~j8: 机械臂当前的角度值
		六轴机械臂前六个数为有效值，后两个数为零
		四轴机械臂以实际机械臂适配为准

24. 获取机械臂当前位姿 (GetCartPose)

工控机发送	“123,#“	122: GetJoints 指令号
工控机接收	“123,error_code,x,y,z,a,b,c,d,#“	x: 当前 Cartesian 的 x 轴位置
		y: 当前 Cartesian 的 y 轴位置
		z: 当前 Cartesian 的 z 轴位置
		a: 当前 Cartesian 姿态数据的第一位
		b: 当前 Cartesian 姿态数据的第二位
		c: 当前 Cartesian 姿态数据的第三位
		d: 当前 Cartesian 姿态数据的第四位

注：机械臂返回的参数的单位均为机械臂本身自己的单位，单位的转换和姿态数据的统一在工控机实现

25. 机械臂后台发送状态

机械臂发送	“200, 201,j1,⋯,j8,202,x,⋯,d,203,di1,⋯,din,#“	200: 后台发送数据指令号
		201: 关节角数据标志，其后 8 位数据 j1 到 j8 表示机械臂当前关节角度。
		202: 位姿标志位，其后的 7 位数据表示机械臂当前位姿
		203：数字量输入标志位 (可选)，其后的位数不同机械臂不同，与工控机解析相统一
机械臂接收	“200,0,#“	

16.2 机械臂主控通讯协议 v1

此版本将于 1.5 MAX 弃用

16.2.1 协议格式

“指令 (数据)+ 分隔符 + 数据 + 分隔符 + 数据 + 分隔符 +⋯+ 数据 + 分隔符 + 结束符”

发送和接收的数据格式如上所示，数据之间通过分隔符划分，最后以一个结束符来结尾。分隔符可以是英文逗号 (‘,’) 或者空格 (’ ‘)，结束符可以使’ #’ 或者’ \n’。分隔符和结束符根据机械臂的特性可以灵活调整。

16.2.2 error_code 判断

警告： 机械臂主控时，机械臂运行程序必须要对接收的返回数据进行 error_code 的判断。如果不处理 error_code，机械臂会发生碰撞或者造成其他更严重的后果，威胁使用者的人身安全！

16.2.3 指令

1. 心跳信号 (xyzHeartBeat)

机械臂发送	"500,#"	500: 机械臂 ping 工控机用
机械臂接收	"error_code,#"	

2. 切换应用 (xyzSwitchApp)

机械臂发送	"501,app_name,#"	501: 切换应用指令号
		app_name(string): 需要切换到的 app 名字
机械臂接收	"error_code,#"	

3. 切换 flow(xyzSwitchFlow)

机械臂发送	"502,flow_name,#"	502: 切换 task flow 指令号
		flow_name(string): 需要切换到的 task flow 名称，以 ".t" 结尾，如 "1.t"
机械臂接收	"error_code,#"	

4. 切换工具 (xyzSwitchTool)

机械臂发送	"503,tool,#"	503: 切换工具指令号
		tool(string): tool id
机械臂接收	"error_code,#"	

5. 请求拍照 (xyzReqCapImg)

机械臂发送	"504,vision_service_id,#"	504: 请求拍照指令号
		vision_service_id(int): 视觉 service_id
机械臂接收	"error_code,token,#"	token(int): 返回的用于请求对应拍照结果时使用的 token

6. 获取拍照结果 (xyzGetCapImg)

机械臂发送	"505,token,#"	505: 获取拍照结果指令号
		token(int): 请求拍照时返回的 token
机械臂接收	"error_code,#"	

7. 拍照 (xyzCapImg)

机械臂发送	"506,vision_service_id,#"	506: 拍照指令号
		vision_service_id(int): 视觉 service_id
机械臂接收	"error_code,#"	

8. 请求抓取目标点位 (xyzReqGraspPose)

机械臂发送	"507,ws_id,#"	507: 请求抓取目标点位指令号
		ws_id(int): 需要获取抓取点位的工作空间 id
机械臂接收	"error_code,token,#"	token(int): 返回的用于获取目标点位时使用的 token

9. 获取抓取目标点位 (xyzGetGraspPose)

机械臂发送	"508,token,#"	508: 获取抓取目标点位指令号
		token(int): 请求抓取目标点位时返回的 token
机械臂接收	"error_code,x,y,z,a,b,c,d,num,pipeline_num,register_num,#"	x,y,z,a,b,c,d: 为抓取目标点的位姿数据
		num(int): 当前有多少个可供抓取的点
		pipeline_num(int): pipeline 文件 number
		register_num(int): 用到的注册文件的注册 number

10. 请求物体位姿 (xyzReqObjPose)

机械臂发送	"509,ws_id,#"	509: 请求物体位姿指令号
		ws_id(int): 请求物体所在的空间 id
机械臂接收	"error_code,token,#"	token(int): 用于获取物体位姿时作为识别 token

11. 获取物体位姿 (xyzGetObjPose)

机械臂发送	"510,token,#"	510: 获取物体位姿指令号
		token(int) : 请求物体位姿时得到的 token
机械臂接收	"error_code,x,y,z,a,b,c,d,num,pose_type,#"	x,y,z,a,b,c,d: 物体的位姿数据
		num(int): 当前物体位姿个数
		pose_type(int) : 物体的位姿类型

12. 重置任务 (xyzResetTask)

机器人发送	"511,#"	511: 重置任务指令号 一般用来初始化任务内部变量
机器人接收	"error_code,# "	

13. 发送机械臂当前角度 (xyzSendCurrentJoints)

机械臂发送	"512,j1,j2,j3,j4,j5,j6,#"	512: 发送机械臂当前角度指令号
		j1~j6: 机械臂当前的角度信息
		如果机械臂轴数小于 6 轴，则需要对应位置补零，依旧发送六个数
机械臂接收	"error_code,#"	

14. 发送机械臂当前 Cartesian(xyzSendCurrentCartPose)

机械臂发送	"513,x,y,z,a,b,c,d,#"	513: 发送机械臂当前笛卡尔位姿指令号
		x,y,z: 机械臂当前位置数据
		a,b,c,d: 机械臂当前姿态数据
		如果机械臂是欧拉角形式，则只需给 a,b,c 赋值，d 赋值 0 即可
机械臂接收	"error_code,#"	

15. 发送机械臂当前扩展轴位置 (xyzSendCurrentExtJoints)

机械臂发送	"514,j1,j2,j3,j4,j5,j6,#"	514: 发送机械臂当前扩展轴的位置
		j1~j6: 机械臂当前扩展轴的角度信息
		如果扩展轴数不足 6 的, 需要补零后发送六个数
机械臂接收	"error_code,#"	

16. 请求 pick 动作规划 (xyzReqPick)

机械臂发送	"515,#"	515: 请求 pick 动作规划指令号 该指令暂不支持
机械臂接收	"error_code,#"	

17. 请求 place 动作规划 (xyzReqPlace)

机械臂发送	"516,#"	516: 请求 place 动作规划指令号 该指令暂不支持
机械臂接收	"error_code,#"	

18. 请求 pick 和 place 规划 (xyzReqPickPlace)

机械臂发送	"517,ws_id,#"	517: 请求 pick 和 place 规划指令号
		ws_id(int): 工作空间 id
机械臂接收	"error_code,#"	

19. 获取取料入框轨迹 (xyzGetPickin)

机械臂发送	"518,ws_id,#"	518: 获取取料入框指令号
		ws_id(int): 工作空间 id
机械臂接收	"error_code,pipeline_num,register_num,traj_number"	pipeline_num(int): pipeline 文件的注册 number
		register_num(int): 用到的注册文件的注册 number
		traj 轨迹点: num+ wp_type + wp + wp_type + wp +…. num: waypoint num 轨迹点数量 wp_type 轨迹点类型: joints or Cartesian，可能值: • 11 on joints movej • 12 on joints movel • 21 on cart movej • 22 on cart movel wp: 点位，6 个值 (j1~j6 或者 xyz-abc)

20. 获取取料出框轨迹 (xyzGetPickout)

机械臂发送	"519,ws_id,#"	519: 获取取料出框轨迹指令号
		ws_id(int): 工作空间 id
机械臂接收	"error_code,pipeline_num,register_num,traj_number,#"	pipeline_num(int): pipeline 文件
		register_num(int): 用到的注册文件的注册 number
		traj 轨迹点: num+ wp_type + wp + wp_type + wp +….
		num: waypoint num 轨迹点数量
		wp_type 轨迹点类型: joints or Cartesian，可能值: • 11 on joints movej • 12 on joints movel • 21 on cart movej • 22 on cart movel
		wp: 点位，6 个值 (j1~j6 或者 xyz-abc)

21. 获取放料入框轨迹 (xyzGetPlacein)

机械臂发送	"520,ws_id,#"	520: 获取放料入框轨迹指令号
		ws_id(int): 工作空间 id
机械臂接收	"error_code,pipeline_num,register_num,traj_number,#"	pipeline_num(int): pipeline 文件
		register_num(int): 用到的注册文件的注册 number
		traj 轨迹点: num+ wp_type + wp + wp_type + wp +….
		num: waypoint num 轨迹点数量
		wp_type 轨迹点类型: joints or Cartesian，可能值: • 11 on joints movej • 12 on joints movel • 21 on cart movej • 22 on cart movel
		wp: 点位，6 个值 (j1~j6 或者 xyz-abc)

22. 获取放料出框轨迹 (xyzGetPlaceout)

机械臂发送	"521,ws_id,#"	521: 获取放料出框轨迹指令号
		ws_id(int): 工作空间 id
机械臂接收	"error_code,pipeline_num,register_num,traj_number,#"	pipeline_num(int): pipeline 文件的注册 number
		register_num(int): 用到的注册文件的注册 number
		traj 轨迹点: num+ wp_type + wp + wp_type + wp +….
		num: waypoint num 轨迹点数量
		wp_type 轨迹点类型: joints or Cartesian，可能值: • 11 on joints movej • 12 on joints movel • 21 on cart movej • 22 on cart movel
		wp: 点位，6 个值 (j1~j6 或者 xyzabc)

23. 请求切换策略 (xyzSwitchStrat)

机器人发送	"522,strat_name,#"	522: 请求切换策略指令号
		strat_name(string): 策略名称
机器人接收	"error_code,#"	

24. 料箱重定位 (xyzUpdateTotePose)

机械臂发送	"523,#"	523: 料箱重定位指令号 执行完毕后，环境中的料箱位姿会被自动更新
机械臂接收	"error_code,x,y,z,a,b,c,d,#"	x,y,z,a,b,c,d: 重定位后的料箱位姿数据，目前该值用不到，用户无需使用

25. 工件在上手的二次定位 (xyzUpdateObjPoseOnHand)

机械臂发送	"524,#"	524: 工件在手上的二次定位指令号
机械臂接收	"error_code,#"	

26. 工件不在手上的二次定位 (xyzUpdateObjPoseToHand)

机械臂发送	"525,#"	525: 工件不在手上的二次定位指令号
机械臂接收	"error_code,pipeline_num,register_num,traj_number,#"	pipeline_num(int): pipeline 文件 number
		register_num(int): 用到的注册文件的注册 number
		traj 轨迹点: num+ wp_type + wp + wp_type + wp +…. num: waypoint num 轨迹点数量 wp_type 轨迹点类型: joints or Cartesian，可能值: • 11 on joints movej • 12 on joints movel • 21 on cart movej • 22 on cart movel
		wp: 点位，6 个值 (j1~j6 或者 xyzabc)

27. 获取工件姿态类型 (xyzGetObjPoseType)

机器人发送	"526,#"	526: 工件正反面识别指令号
机器人接收	"error_code,pose_type,#"	pose_type(int): 工件的 pose 类型信息

28. 重置工业码垛状态 (xyzResetPalletStatus)

机械臂发送	"527,#"	527: 重置码垛状态指令号
机械臂接收	"error_code,#"	

29. 切换工件 (xyzSwitchItem)

机器人发送	"528,ws_id,item_codename,#"	528: 切换工件指令号
		ws_id(int): 工作空间 id
		item_codename(string): 工件代号
机器人接收	"error_code,#"	

30. 计算抓取目标点位 (xyzCalculateGraspPose)

机械臂发送	"529,ws_id,#"	529: 计算抓取目标点位指令号 该指令等价于 xyzReqGraspPose +xyzGetGraspPose
		ws_id(int): 工作空间 id
机械臂接收	"error_code,x,y,z,a,b,c,d,num,pipeline_num,register_num,#"	x,y,z,a,b,c,d: 为抓取目标点的位姿数据 num(int): 当前有多少个可供抓取的点 pipeline_num(int): pipeline 文件 number register_num(int): 用到的注册文件的注册 number

31. 计算物体位姿 (xyzCalculateObjectPose)

机械臂发送	“530,ws_id,#”	530: 计算物体位姿指令号
		该指令等价于 xyzReqObjPose + xyzGetObjPose
		ws_id(int): 工作空间 id
机械臂接收	“error_code,x,y,z,a,b,c,d,num,pose_type,# “	x,y,z,a,b,c,d: 物体的位姿数据
		num(int): 当前物体位姿个数
		pose_type(int) : 物体的位姿类型

16.3 机械臂主控通讯协议 v2

此协议将于 1.5 MAX 启用，1.5 MAX 之前的版本请参考 v1 版本协议。

16.3.1 协议格式

“指令 (数据)+ 分隔符 + 数据 + 分隔符 + 数据 + 分隔符 +···+ 数据 + 分隔符 + 结束符”

发送和接收的数据格式如上所示，数据之间通过分隔符划分，最后以一个结束符来结尾。分隔符可以是英文逗号 (‘,’) 或者空格 (’ ‘)，结束符可以使’ #’ 或者’ \n’。分隔符和结束符根据机械臂的特性可以灵活调整。

16.3.2 error_code 判断

警告: 机械臂主控时，机械臂运行程序必须要对接收的返回数据进行 error_code 的判断。如果不处理 error_code，机械臂会发生碰撞或者造成其他更严重的后果，威胁使用者的人身安全!

16.3.3 指令

1. 心跳信号 (xyzHeartBeat)

机械臂发送	“500,#”	500: 机械臂 ping 工控机用
机械臂接收	“error_code,#”	

2. 切换任务 (xyzSwitchTask)

机器人发送	“502,task_codename,#”	502: “切换任务” 指令号
		task_codename(string): 任务代号，以” .t “结尾，如 “1.t”
机器人接收	“error_code,# “	

3. 切换工具 (xyzSwitchTool)

机器人发送	“503,tool_id,#”	503: “切换工具” 指令号
		tool_id(int): 工具 id
机器人接收	“error_code,# “	

4. 呼叫视觉命令 (xyzCallVisionCmd)

机器人发送	“504,vision_codename,#”	504: “呼叫视觉” 指令号
		vision_codename(string): 视觉命令代号
机器人接收	“error_code,# “	

5. 请求抓取目标点位 (xyzReqGraspPose)

机器人发送	“507,vs_id,#”	507: “请求抓取目标点位” 指令号
		vs_id(int): 视觉服务 id
机器人接收	“error_code,# “	

6. 获取抓取目标点位 (xyzGetGraspPose)

机器人发送	“508,token,#”	508: ” 获取抓取目标点位” 指令号
		token(int): “请求抓取目标点位” 中返回的 token
机器人接收	“error_code,x,y,z,a,b,c,d,num,pipeline_num, register_num,int1,int2,int3,int4,int5,int6,#”	x,y,z,a,b,c,d(float): 抓取目标点的位姿数据
		num(int): 当前可供抓取的点位数量
		pipeline_num(int): 运动流程编号
		register_num(int): 抓取序号
		int1~int6(int)：可在 Task 中自定义含义
		说明：眼在手上不能用异步方式

7. 请求物体位姿 (xyzReqObjPose)

机器人发送	“509,vs_id,#”	509: “请求物体位姿” 指令号
		vs_id(int): 视觉服务 id
机器人接收	“error_code,token,# “	token(int): 在执行” 获取物体位姿 “时使用到

8. 获取物体位姿 (xyzGetObjPose)

机器人发送	“510,token,#”	510: “获取物体位姿” 指令号
		token(int) : “请求物体位姿” 中返回的 token
机器人接收	“error_code,x,y,z,a,b,c,d,num,object_name, int1,int2,int3,int4,int5,int6,# “	x,y,z,a,b,c,d(float): 物体的位姿数据
		num(int): 当前物体位姿个数
		object_name(string) : 物体名称
		int1~int6(int)：可在 Task 中自定义其含义

9. 重置任务 (xyzResetTask)

机器人发送	“511,#”	511: “重置任务” 指令号
		下位机在初始化时需要调用该指令
机器人接收	“error_code,# “	token(int): 在执行” 获取物体位姿 “时使用到

10. 发送机器人当前关节坐标 (xyzSendCurrentJoints)

机器人发送	“512,j1,j2,j3,j4,j5,j6,#”	512: “发送机器人当前关节坐标” 指令号
		j1~j6(float): 机器人当前的角度信息。如果机器人轴数小于 6 轴，则需要对应位置补零，依旧发送六个数
机 器 人接收	“error_code,# “	token(int): 在执行” 获取物体位姿 “时使用到

11. 发送机器人当前笛卡尔坐标 (xyzSendCurrentCartPose)

机器人发送	“513,x,y,z,a,b,c,d,#”	513: “发送机器人当前笛卡尔坐标”指令号
		x,y,z(float): 机器人当前位置数据
		a,b,c,d(float): 机器人当前姿态数据，如果机器人是欧拉角形式，则只需给 a,b,c 赋值，d 赋值 0 即可
机器人接收	“error_code,# “	

12. 发送机器人当前扩展轴坐标 (xyzSendCurrentExtJoints)

机器人发送	“514,j1,j2,j3,j4,j5,j6,#”	514: ”发送机器人当前扩展轴坐标“指令号
		j1~j6(float): 机器人当前扩展轴的角度信息，如果扩展轴数不足 6 的, 需要补零后发送六个数
机器人接收	“error_code,# “	

13. 请求抓放规划 (xyzReqPickPlace)

机器人发送	“517,vs_id,#”	517: “请求抓放规划“指令号
		vs_id(int): 视觉服务 id
机器人接收	“error_code,# “	

14. 获取取料入框轨迹 (xyzGetPickin)

机器人发送	“518,vs_id,#”	518: ”获取取料入框轨迹“指令号
		vs_id(int): 视觉服务 id
机器人接收	“error_code,pipeline_num,register_num,traj,# “	pipeline_num(int): 运动流程编号
		register_num(int): 抓取序号
		traj(轨迹)：num+ wp_type + wp + wp_type + wp +…..
		num(int)：轨迹中的点位数量
		wp_type(int)：点位运动类型 11: moveJ(joints) 12: moveL(joints) 13: moveJ(cart_pose) 14: moveL(cart_pose)
		wp(6 个 float)：点位当 wp_type 是 11 或 12 时：joints: j1~j6 当 wp_type 是 21 或 22 时：cart_pose: xyzabc

15. 获取取料出框轨迹 (xyzGetPickout)

机器人发送	“519,vs_id,#”	519: ”获取取料出框轨迹“指令号
		vs_id(int): 视觉服务 id
机器人接收	“error_code,pipeline_num,register_num,traj,# “	pipeline_num(int): 运动流程编号
		register_num(int): 抓取序号
		traj(轨迹), 同 xyzGetPickin traj

16. 获取放料入框轨迹 (xyzGetPlacein)

机器人发送	“520,vs_id,#”	520: ”获取放料入框轨迹“指令号
		vs_id(int): 视觉服务 id
机器人接收	“error_code,pipeline_num,register_num,traj,# “	pipeline_num(int): 运动流程编号
		register_num(int): 抓取序号
		traj(轨迹), 同 xyzGetPickin traj

17. 获取放料出框轨迹 (xyzGetPlaceout)

机器人发送	“521,vs_id,#”	521: ” 获取放料出框轨迹 “指令号
		vs_id(int): 视觉服务 id
机器人接收	“error_code,pipeline_num,register_num,traj,# “	pipeline_num(int): 运动流程编号
		register_num(int): 抓取序号
		traj(轨迹), 同 xyzGetPickin traj

18. 料箱重定位 (xyzUpdateTotePose)

机器人发送	“523,vs_id,#”	523: ” 料箱重定位 “指令号
		vs_id(int): 视觉服务 id
机器人接收	“error_code,# “	

19. 工件在手上的二次定位 (xyzUpdateObjPoseInHand)

机器人发送	“524,#”	524: ” 工件在手上的二次定位 “指令号
机器人接收	“error_code,x,y,z,a,b,c,d,# “	x,y,z,a,b,c,d(float): 放置位姿数据

20. 切换工件 (xyzSwitchItem)

机器人发送	“528,vs_id,item_codename,#”	528: “切换工件” 指令号
		vs_id(int): 视觉服务 id
		item_codename(string): 工件代号
机器人接收	“error_code,# “	

21. 计算抓取目标点位 (xyzCalculateGraspPose)

机器人发送	“529,vs_id,#”	529: “计算抓取目标点位” 指令号
		该指令等价于 xyzReqGraspPose +xyzGetGraspPose
		vs_id(int): 视觉服务 id
机器人接收	“error_code,x,y,z,a,b,c,d,num,pipeline_num, register_num,int1,int2,int3,int4,int5,int6,#”	x,y,z,a,b,c,d(float): 抓取目标点的位姿数据
		num(int): 当前可供抓取的点位数量
		pipeline_num(int): 运动流程编号
		register_num(int): 抓取序号
		int1~int6(int): 可在 Task 中自定义含义

22. 计算物体位姿 (xyzCalculateObjectPose)

机器人发送	“530,vs_id,#”	530: “计算物体位姿” 指令号
		该指令等价于 xyzReqObj-Pose+xyzGetObjPose
		vs_id(int): 视觉服务 id
机器人接收	“error_code,x,y,z,a,b,c,d,num,object_name, int1,int2,int3,int4,int5,int6,# “	x,y,z,a,b,c,d(float): 物体的位姿数据
		num(int): 当前物体位姿个数
		object_name(string) : 物体名称
		int1~int6(int): 可在 Task 中自定义含义

23. 用户指令 (xyzUsrCmd)

机器人发送	“600,str_1,str_2,str_3,str_4,str_5, int_1,int_2,int_3,int_4,int_5,int_6,int_7,int_8,int_9,int_10, float_1,float_2,float_3,float_4,float_5, float_6,float_7,float_8,float_9,float_10, x,y,z,a,b,c,d, j1,j2,j3,j4,j5,j6,#”	600：用于指定“自定义指令”的指令号 用于基础指令不支持的情况下，配合任务流图完成用户特定功能。 各个参数的含义取决于任务流图中设定的输入输出。 str_1~str_5：提供 5 个字符串供用户输入 int_1~int_10：提供 10 个整数类型供用户输入 float_1~float_10：提供 10 个浮点数供用户输入 xyzabcd(float)：笛卡尔坐标值，如果机器人是欧拉角形式，则只需给 a,b,c 赋值，d 赋 0 即可 j1~j6：关节坐标值
机器人接收	“error_code,str_1,str_2,str_3,str_4,str_5, int_1,int_2,int_3,int_4,int_5,int_6,int_7,int_8,int_9,int_10, float_1,float_2,float_3,float_4,float_5, float_6,float_7,float_8,float_9,float_10, x,y,z,a,b,c,d, j1,j2,j3,j4,j5,j6,#”	str_1~str_5：返回 5 个字符串 int_1~int_10：返回 10 个整数值 float_1~float_10：返回 10 个浮点数 xyzabcd：返回一个笛卡尔位姿数据 j1~j6：返回一个关节坐标

CHAPTER 17

术语

API 应用程序接口。一组预定义的函数或过程，使应用程序能在无需提供源代码的情况下访问系统提供的功能。

ICP 迭代最近点算法，一种应用广泛的点云配准算法，主要目的是找到旋转和平移参数，将两个不同坐标系下的点云，以其中一个点云坐标系为全局坐标系，另一个点云经过旋转和平移后，两组点云重合部分完全重叠。随着发展，ICP 算法已衍生出多种变体。

OCR 字符识别，使用深度学习技术，识别图像中的文字内容并输出。

PCA 主成分分析，一种数学降维方法，利用正交变换把一系列可能线性相关的变量转换为一组线性不相关的新变量，也称为主成分，从而利用新变量在更小的维度下展示数据的特征。

primitive 指能够用于机械臂抓取的视觉计算结果。primitive 中包含了很多信息，包括物体的尺寸、姿态以及其他有用信息。

RANSAC 随机抽样一致算法，一种经典的数据过滤和参数拟合算法。该算法假设数据（内点，inlier）分布符合一定的数学模型，通过迭代计算，去除外点 (outlier，又称离群点) 和噪声点，同时获取概率上最佳的模型参数。在全局定位中，内点指正确的匹配，外点指错误的匹配，参数模型指匹配点对的空间变换矩阵。

SKU 存货单元。因某些特征 (如品牌、尺寸、颜色、型号) 使其与其他物品分开存放和计数的单位仓储物品。可以以件、盒、托盘等为单位，是物理上不可分割的最小存货单元。

STL STL 文件格式是一种为快速原型制造技术服务的三维图形文件格式。STL 文件由多个三角形面片的定义组成，每个三角形面片的定义包括三角形各个定点的三维坐标及三角形面片的法矢量。

包围盒 可将物体包含在内的二维矩形区域或者三维长方体。

点云 在同一空间参考系下表达目标空间分布和目标表面特性的海量点集合。

感兴趣区域（ROI） 从图像中圈定的需进一步处理的区域。

灰度图 以不同明暗等级的中性灰色表达信息的图像。

机加工件 用各种机械加工设备加工成型的机械零部件。一般采用的加工方式不发生化学反应或者反应很微小，如车、铣、刨、磨等。

夹具 供抓取和握持用的末端执行器。

节拍 指机械臂完成一套工艺流程所需要的时间，代表机械臂的工作效率。

机械臂主控（Robot Master） 由机械臂（包含 PLC 等设备）根据任务主动发送各类指令给工控机获取信息的一种工作模式。机械臂（包含 PLC 等设备）主动从工控机获取各类信息，如视觉信息、点位信息、轨迹信息后，其根据需要自行决定如何使用这些信息。常见的机械臂主控场景有：“2D/3D 座标移动”机械臂主控、“3D 轨迹移动”机械臂主控等。

聚类 将具有相似信息或数据特征的对象聚在一起的一种数据分析方法。被聚在一起的研究对象称为一个类(cluster)，位于同一类的对象相似度高，不同类的对象相似度较低。

栅格 一种存储、处理及显示空间数据的方法。将一给定区域分割为由行与列所形成的规则方格结构矩阵。矩阵中每一方格包含属性值及位置坐标，其位置隐含于矩阵的顺序中。

工控机 生产管理系统中，能发出操控命令的计算机。

工控机主控（IPC Master） 由工控机根据任务需求主动下发各类指令给机械臂/PLC 等设备让其执行的一种工作模式。机械臂/PLC 等设备处于等待工控机指令的状态，一旦获取了工控机指令，机械臂/PLC 会做出相应的动作，如移动到点位、读写 IO 等。常见的工控机主控场景有：使用 XYZ-Studio-Max 虚拟示教器点动机器人、相机自动标定、“2D/3D 座标移动”工控机主控、“3D 轨迹移动”工控机主控、拆码垛等。

深度图 存储三维深度特征信息的图像。

推理线程 一种多线程技术，推理线程可能需要对输入变量的值做出假设，如果这些被证明是无效的，那么依赖这些输入变量的推测线程的部分将需要被丢弃和压缩；如果假设是正确的，则程序可以在更短的时间内完成。推理线程和正常线程并行执行。

位姿 空间位置和姿态的合称。

像素 数字图像的最小构成单位。

掩膜 选定的图像、图形或物体，其作用是对处理的图像（全部或局部）进行遮挡，来控制图像处理的区域或处理过程。

置信度 在一定数据范围内，真值出现在该范围内的概率，又称置信水平。该区间被称为“置信区间”。

CHAPTER 18

修订记录

- 2023-09-12，V1.4.1，变更如下。
  - 新增圆柱体抓取和工业码垛场景。
  - 修改文档结构。
- 2023-08-24，V1.4.0，变更如下。

  V1.4.0 首次发布。