ADTECH众为兴

把工作交给机器 将思想留给自己

QC410 控制器用户手册

深圳众为兴技术股份有限公司

地址：深圳市南山区艺园路田厦 IC 产业园 5 楼　　邮编：518052

电话：+86-755-26722719　　传真：0755-26722718-816

Email：tech@adtechcn.com　　Web：http://www.adtechcn.com

版权声明

本手册的所有部分，著作财产权归深圳众为兴技术股份有限公司（以下简称众为兴）所有，未经众为兴许可，任何单位或个人不可任意仿制、拷贝、撰抄或转译。本手册无任何形式的担保、立场表达或其它暗示。如有本手册或其所提到的产品的信息，所引起的直接或间接的资料流出，利益损失或事业终止，众为兴及其所属员工不承担任何责任。除此以外，本手册提到的产品及其资料仅供参考，内容如有更新，恕不另行通知。

版权所有，不得翻印。

深圳众为兴技术股份有限公司

说明书基本信息

本说明书由深圳众为兴技术股份有限公司组织编写。

本说明书主要编写人：众为兴

版本号：V2.5.6

说明书范围

本说明书是关于QC410驱控一体相关的操作手册，包含ARStudio操作、电气接线、语言编程以及伺服调试四个章节。

安全注意事项

※运输与储存

- ☞ 产品包装箱堆迭不可超过六层
- ☞ 不可在产品包装箱上攀爬、站立或放置重物
- ☞ 不可使用与产品相连的电缆拖动或搬运产品
- ☞ 严禁碰撞、划伤面板和显示屏
- ☞ 产品包装箱应避免潮湿、暴晒以及雨淋

※开箱检查

- ☞ 打开包装后请确认是否是您所购买的产品
- ☞ 检查产品在运输途中是否有损坏
- ☞ 对照清单确认各部件是否齐全，有无损伤
- ☞ 如存在产品型号不符、缺少附件或运输损坏等情况，请及时与我公司联系

※接线

- ☞ 参加接线与检查的人员必须是具有相应技术的专业人员
- ☞ 产品必须可靠接地，接地电阻应小于4 欧姆，不能使用中性线（零线）代替地线
- ☞ 接线必须正确、牢固，以免导致产品故障或意想不到的后果
- ☞ 与产品连接的浪涌吸收二极管必须按规定方向连接，否则会损坏产品
- ☞ 插拔插头或打开产品机箱前，必须切断产品电源

※检修

- ☞ 检修或更换元器件前必须切断电源
- ☞ 发生短路或过载时应检查故障，故障排除后方可重新启动
- ☞ 不可对产品频繁通断电，断电后若须重新通电，相隔时间至少1分钟

※其它

- ☞ 未经允许，请勿擅自打开机壳。
- ☞ 长时间不用，请切断电源。

- ☞ 特别注意不要让粉尘，铁粉进入产品。
- ☞ 输出继电器若使用非固态继电器，则须在继电器线圈上并联续流二极管。检查所接电源是否符合要求，杜绝将控制器烧坏。
- ☞ 控制器的寿命与环境温度有很大关系，若加工现场温度过高，请安装散热风扇。控制器允许工作的环境温度范围在0℃-60℃之间。
- ☞ 避免在高温、潮湿、多尘或有腐蚀性气体的环境中使用。
- ☞ 在震动强烈的地方，应加橡胶防震垫进行缓冲。

※保养

- ☞ 在一般的使用条件下（环境条件：日平均30℃，负载率80%，运行率每天12小时），请按如下项目进行日常检查和定期检查。

日常检查	日常	● 确认环境温度、温度、尘埃异物 ● 有无异常震动、声音 ● 通风孔有无被纱线等塞住
定期检查	1 年	● 坚固部件是否松动 ● 端子台是否损伤

由于机器人系统较为复杂，危险性大。本说明书记录了与安全有关的注意事项，请严格遵守所记录的事项。

● 安全注意事项与标记

标志		标志的含义
	危险	表示错误使用时，会引发危险状况，造成人员重伤或死亡
	注意	表示错误使用时，会引发危险状况，可能造成人员伤害或损坏设备造成物质损失。
	禁止	表示绝对不可实施

!	强制	表示必须要实施的

- 危险

请不要在易燃、易爆的环境下使用本系统。	
!	容易造成伤害或引起火灾。

请按照图纸或者说明书接线。	
!	易发生触电、损坏电机。

在通电状态下，不要随意拔插头，在运行状态下，不要触碰机器人运转部位。	
!	易触电、造成人身伤害。

通电状态下，不要进行接线、维修等操作，请务必断电 5 分钟以上再进行操作。	
!	易触电。

请务必将驱控一体机与机器人本体之间可靠接地	
!	发生故障漏电时易造成触电、火灾事件，易引发误动作。

非专业人员请不要拆开驱控一体机外壳，请不要用手触摸驱控一体机内部部件	
!	容易发生触电

通电情况下，不要接触驱控一体机电源插头	
!	容易发生触电

请不要损伤，重压线缆或在线缆上悬挂重物	
⚠	容易发生触电

通电状态下，不要拔插驱控一体机上的端子	
⚠	容易发生触电及短路

运行状态下，不要拔出驱控一体机上的端子	
⚠	容易发生触电及短路

- **注意** ⚠

请注意驱控一体、机器人本体上的电机及周边设备发热情况。	
⚠	易被烫伤。

发生故障时，先切断电源，查明原因并清除，完全恢复之后，应低速运行设备。	
⚠	若有不良因素残留，易造成误动作。

使用控制器及机器人本体时，不能超出其规格范围。	
⚠	易造成产品损坏。

机器人搬运时，需用附属的固定工具加以固定。	
⚠	防止起吊时，因手臂移动发生伤残事故。

安装、运行、维护点检前，务必仔细阅读说明书，按说明书中步骤操作	
⚠	易发生触电、火灾

电源电压、电源容量一定要采用本公司指定的规格。	
⚠	不正确使用易发生设备故障、易造成火灾。

请正确使用相互配对的正确的驱控一体机及机器人	
⚠	易发生故障

要定期对驱控一体机实施保养和检查作业	
⚠	忽视保养和检查，是造成设备故障及事故的重要原因。

请不要在产品上放置重物	
⚠	易造成伤害

请按说明书里的接线方式正确接线	
⚠	错误的接线方式易造成机器人或驱控一体机损坏或引起火灾

发生异常时，请立马停机	
⚠	易发生触电、人身伤害、火灾

需要修理时，请联系我公司，请勿自行拆解	
⚠	易造成故障

请勿撞击	
⚠	易造成故障

- 禁止

机器人运转过程中，任何人员都不允许站立在机器人动作区域。	
	会发生重大伤残事故。

禁止在工作场地堆放妨碍机器人动作的器材。	
	设备异常时，容易造成伤害。

禁止将手持示教器上的急停开关短接。	
	机器人在发生意外或者运行不正常时，需使用急停开关，停止运行设备。

禁止进行说明书上无记录、不正确的操作。	
	不正确的操作会带来设备的误动作。

禁止操作人员之外的其它人员靠近设备	
	触摸到危险部位会引发伤残或重大事故发生

发生事故时，要切断电源，清楚不良原因	
	有不良原因残留时，机器人可能会发生误动作，引起不良后果。

禁止用户擅自进行部件调换及改造	
	会降低系统性能且可能发生故障

请不要自行拆卸清理

	易造成火灾、易触电
请不要将产品保存在漏雨、有水、有害气体等环境中	
	易发生故障

✧ 强制

保存时请避免被阳光直射	
	易造成故障

请在规定范围内使用	
	易造成烧毁、故障

运转过程中必须关闭设备防护罩	
	打开的防护罩会有触电、伤残危险

操作人员要经过充分的培训	
	进行不正确操作，会引发设备误动作，造成伤残或重大灾害发生

手动示教时，如果机器人未按照指定的方向动作，立即按下急停，停止设备运行	
	易发生事故及故障

电源线一定要使用指定的电线

!	易发生火灾及故障

安全守则

开机运行前，须知道机器人根据所编程序将要执行的全部任务；

机器人运行在自动模式下，任何人员都不允许进入其运动所及的区域；

需要编程、测试及维修工作时，须将机器人置于手动模式之下；

调试人员进入机器人工作区域时，须随身携带示教器，防止他人误动作；

机器人长时间不运作时，夹具上不应当放置物品，须空机；

停电之后须及时关闭机器人上的主电源开关，并取走夹具上的工具；

目录

目录 10
第一章 ARStudio 操作 12
1.1 ARStudio APP 12
1.2 界面一览表 17
1.3 工程空间操作 23
1.4 机器人配置：系统 56
1.5 机器人配置：资源 77
1.6 辅助工具 90
1.7 报警错误处理 122
1.8 附录：修改记录 136
第二章电气连接篇 137
2.1 系统介绍与功能概述 137
2.2 外部接线 145
2.3 电源线以及外部急停端子接线 153
2.4 机器人本体与 QC410 控制器连线 156
2.5 数字输入输出接口 166
2.6 通信端口 172
2.7 产品安装尺寸图 179
2.8 附录：修改记录 179
第三章 AR 语言手册篇 181
3.1 AR 语言总览 181
3.2 修改记录 301
第四章伺服管理调试篇 304
4.1 伺服调试软件启动 305
4.2 参数管理 314
4.3 基本参数设置 314
4.4 示波器 323
4.5 运动调试 340

4.6 高级调试 356
4.7 伺服错误报警与处理 357

第一章 ARStudio 操作

1.1 ARStudio APP

1.1.1 ARStudio APP 介绍

ARStudio 是适用于 QC410 控制器的一款上位机软件。

ARStudio 的特征：

- 适用于微软 Windows 7, Windows 10 等操作系统；
- 集成应用开发环境：用于提供程序开发环境的应用程序，包括代码编辑器、编译器、调试器和图形用户界面工具。集成了代码编写功能、分析功能、编译功能、调 试功能等一体化的开发软件服务；
- 通过以太网协议与 QC410 控制器进行通讯；
- 允许一台电脑同时运行多个 ARStudio 软件来分别控制不同的 QC410 控制器；
- AR 语言编程；
- 支持多线程多任务，IO 控制和网络通讯协议；
- IO 系统包括 34 路输入、27 路输出以及扩展 IO（32 输入+32 输出）；
- 支持 TCP/IP、UDP、RS232 串口通讯、Modbus 通讯；
- 可视化轨迹监控；
- 通过 ARStudio 可以直接连接控制器进行编程调试；

1.1.2 ARStudio APP 安装使用说明

- **硬件安装**

首先需要一台成功安装 ARStudio APP 的个人电脑；

- 若是应用于实际生产工艺，需连接控制器和机器人才可以用 ARStudio 控制机人实际运动。
- 若是仅应用于学习，可直接在 ARStudio APP 上选择进行虚拟运动测试工艺。

打开软件后，点击连接 IP ，切换到虚拟位置模式 ，即可学习测试。

- **软件安装**

Windows 环境下：直接打开安装包点击“ARStudioInstaller.exe”双击运行。

1.1.3 ARStudio APP 启动方式

方式 1. 直接双击个人电脑桌面上 ARStudio 软件图标。

方式 2. 选择电脑开始菜单栏里的 ARStudio 软件，单击打开。

注：ADT-QC410 虚拟控制器支持 ARStudio 后台自动运行，因此无需配置随电脑开机自动启动方式。

1.1.4 ARStudio 与控制器通讯连接

➢ 情形 1：虚拟机通讯连接

使用情形：当操作人员没有 QC410 控制器，可以通过虚拟机的方式学习和熟悉 QC410 的界面、按钮操作以及功能的使用等，连接步骤如下：

（1） 电脑 PC 一台，并已成功安装 ARStudio 软件

；

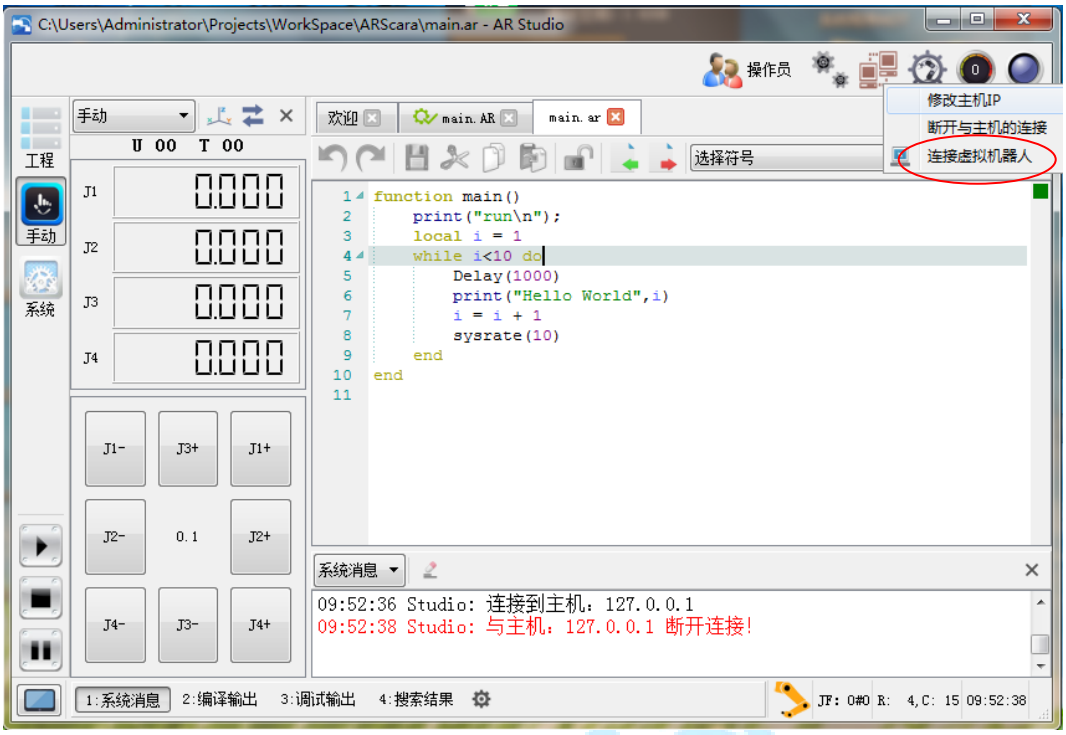
（2） 首次运行 ARStudio 时，软件将自动创建一个“工作空间”并连接到虚拟机器人（127.0.0.1）。若“连接图标”
是未连接状态（暗红色），单击该图标则软件尝试连接控制器，连接成功时变成蓝色
。

或者通过左键长按或者右击该图标
可以弹出控制配置菜单：

“修改主机 IP”：填写目标控制器的 IP，并尝试连接

“断开与主机的连接”：断开与目标控制器的通讯

“连接虚拟机器人”：连接到虚拟控制器（127.0.0.1）



也可鼠标左键点击 图标，弹出对话框，选择“连接虚拟机器人”，这样操作也可成功连接到虚拟机。

➢ 情形 2：PC 与 QC410 控制器通讯连接

使用情形：操作人员通过 PC 机控制机器人的运动，连接步骤如下：

（1） 电脑 PC 一台，并已成功安装 ARStudio 软件

；

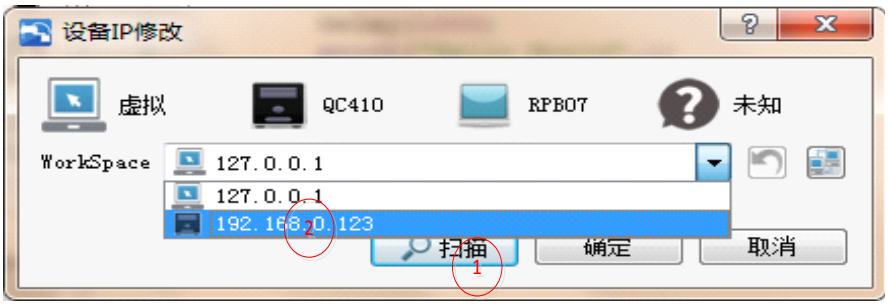
（2） 网线一根，一端插入 QC410 控制器侧面的任意一个 LAN（LAN1~LAN3）端口，另一端插入电脑网口；

（3） 打开 ARStudio 软件，QC410 控制器上电；

（4） 鼠标右键点击 或 图标，弹出对话框选择【修改主机 IP】弹出【修改 IP 地址】对话框；

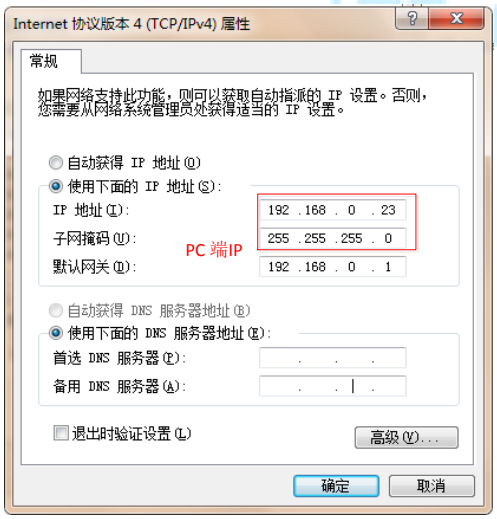
（5） 点击“扫描”下拉框可以自动罗列出当前局域网中可用 QC410 控制器（或 RPB07）的 IP 地址。选择 QC410 图标 对应的 IP--单击右侧小图标--点击“确定”，即可建立连接。

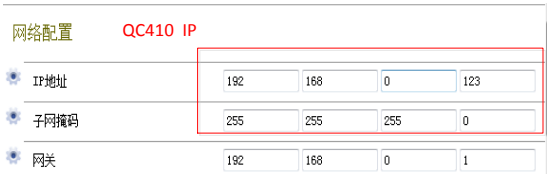
连接成功后，可以在【参数配置】--“串口/网络配置”中查看 QC410 控制器具体的网络配置



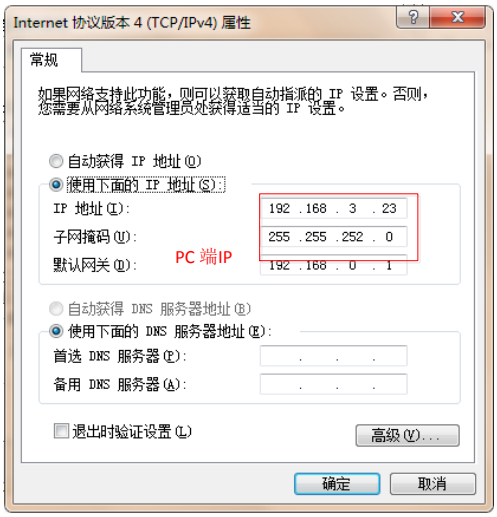
注意 1：PC 机与 QC410 的 IP 地址需设置在同一个网段。如果子网掩码为 255.255.255.0；那么在同一网段内的 IP 可设置范围为：192.168.0.1-192.168.0.255；如果子网掩码为 255.255.252.0，那么在同一网段内的 IP 可设置为 192.168.0.1-192.168.0.255、192.168.1.1-192.168.1.255、192.168.2.1-192.168.2.255、192.168.3.1-192.168.3.255。例如：

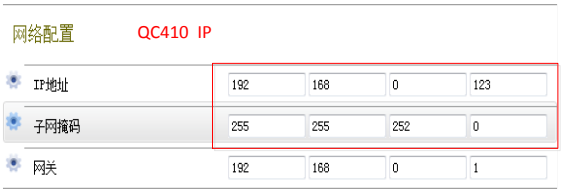
情形1：子网掩码 255.255.255.0



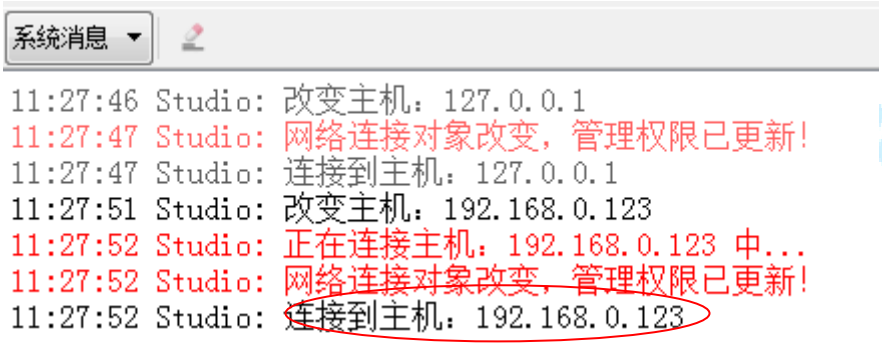


情形2：子网掩码 255.255.252.0

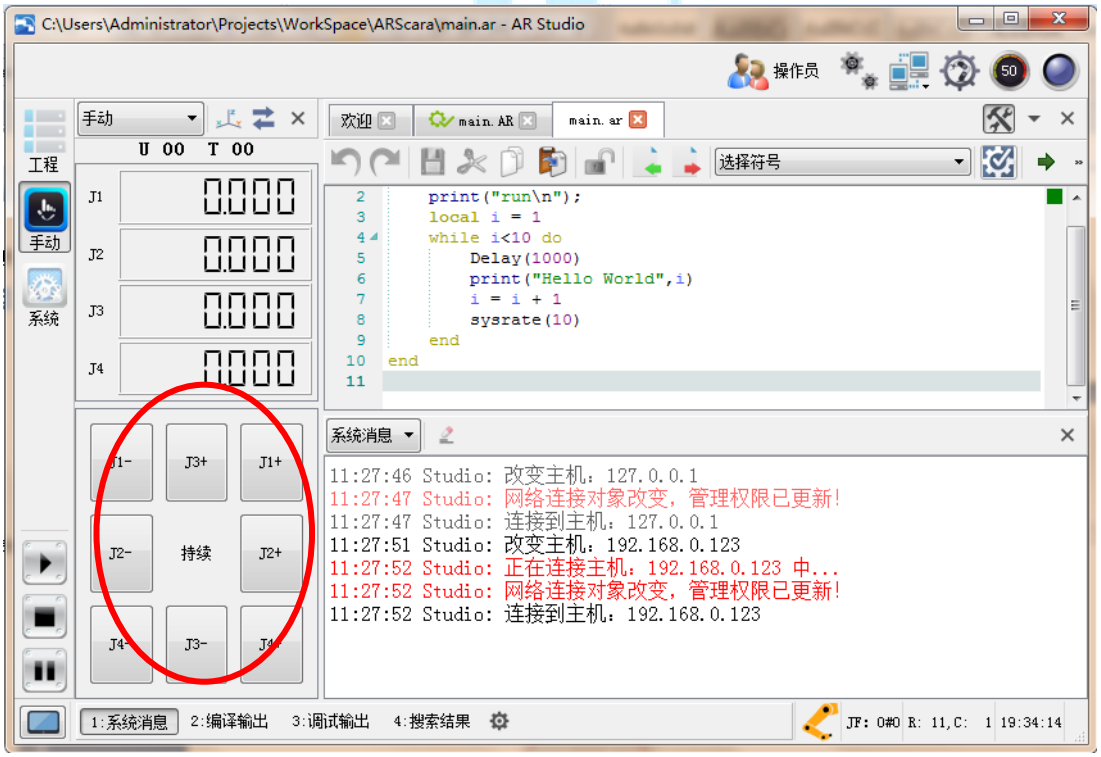




（6） 此时将鼠标箭头移至蓝色图标，并且在【系统消息】中显示连接到主机：192.168.0.123，则表示电脑已经成功连接 QC410 控制器。若控制器没有连接机器人，须点击齿轮图标的闭合状态切换为分开状态，进入虚拟位置模式。



（7） 点击手动图标进入轴运动界面，测试能否正常运动。



注意：IP 地址冲突

情形：当有多台 QC410 控制器或示教器和一台 PC 在同一个局域网（即连接到同一个交换机），且多台 QC410 控制器或示教器配置有相同的 IP 地址，有可能造成

PC 与这些设备是随机连接的，造成安全隐患。

解决办法：已保护（弹窗消息提示），当 PC 与控制器建立连接的时候，若 PC 扫描到有多台相同的 IP 地址的设备时会提示报错。此时须将 QC410 控制器的 IP 地址或示教器的 IP 地址设置为不同。

1.2 界面一览表

注：J1 轴的标志位是指，当-180<J1<180，J1 标志位为 0；当 180<J1<360 or -360<J1<-180，J1 标志位为 1。J2 轴的标志位是指，当-180<J1<180，J2 标志位为 0；当 180<J1<360 or -360<J1<-180，J2 标志位为 1。

本说明书中的一些图标的命名：

图标	名称	功能
	工程	可以调出工程管理界面，工程、程序、点位等在这里设置
	手动	手动操作机器人笛卡尔或关节运行
	系统	机器人参数配置，小工具等功能界面

	运行 停止 暂停 复位	机器人自动运行、停止、暂停、复位 注：复位仅用于清除报警，不能终止程序的运行
	隐藏工具栏	隐藏系统左侧工具栏
操作员	权限管理	设定不同的操作权限，包括工人、操作员、管理员、厂家
	位置模式	实际位置模式、离线位置模式
	后台程序状态	监控当前是否有后台运行程序
	使能	使能、断使能、拖拽三种模式
83	速度倍率	显示或修改运行的速度倍率
	系统状态	运行（绿色）/暂停（黄色）/停止（蓝色）/报警（红色）
	伺服管理	与之前 QC400 的伺服调试软件 QX 功能相同

1.2.1 手动笛卡尔/关节

笛卡尔位置指当前坐标系下的 X、Y、Z、C 相对于零点的实时位置；关节位置指当前坐标系下的 J1、J2、J3、J4 轴实时位置。通过点击笛卡尔或关节所在的区域即可实现手动笛卡尔和关节坐标系的切换。手动笛卡尔/关节运动的三步基本操作，如下图所示：

（1）在已有的用户和工具中选择对应的用户号和工具号；

（2）笛卡尔坐标系和关节坐标系的选择；

（3）对应的轴操作;

对于手动笛卡尔/关节运动，还涉及到一些细节操作，例如连续/单步切换、倍率的修改等操作。

> ✧ **小技巧：**手动连续、单步运动决定手动定位的精度；倍率修改决定手动移动的速度。

1.2.2 手动连续/单步移动

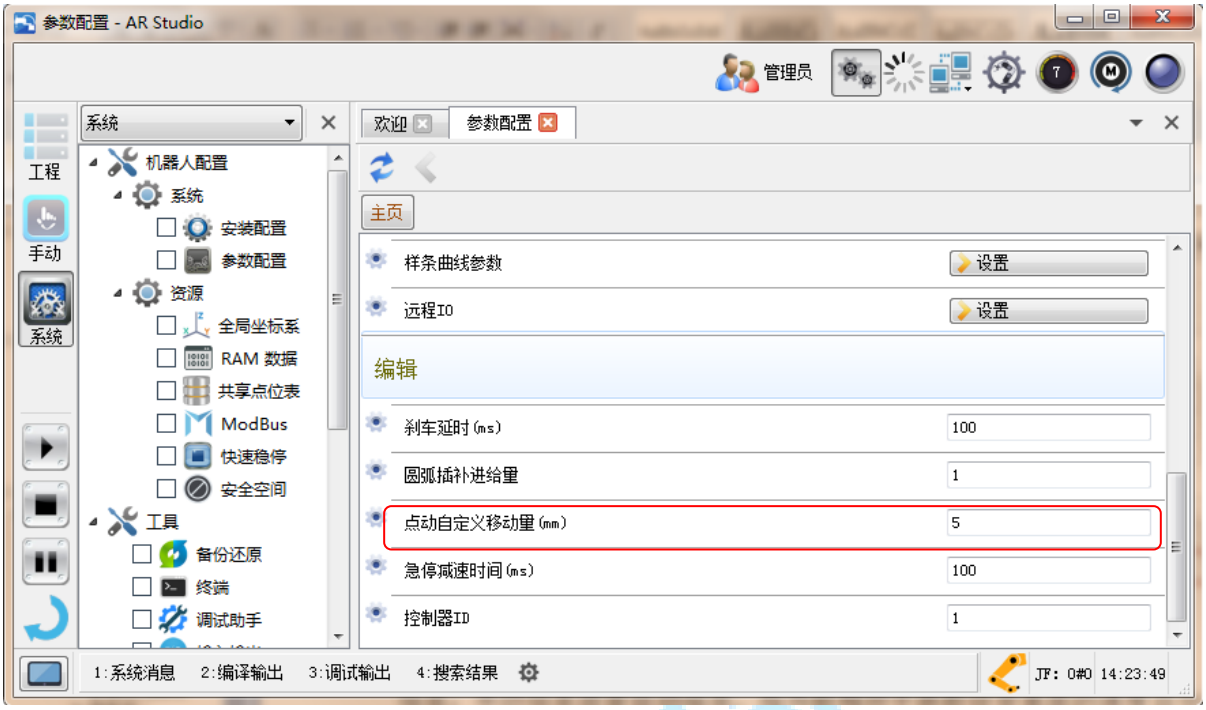
点击“持续”按钮可实现连续/单步移动切换。关于此操作的几点说明：

（1）单步涉及到参数中点动自定义移动量的设定（默认为 5.00），如图 4.5 所示，单步模式中按距离分为三种：0.10、1.00 和 5.00（自定义）；

（2）在笛卡尔坐标系下，对于 X、Y、Z 轴，三个参数的单位是 mm，对于 C 轴，单位是°；

（3）在关节坐标系下，对于 J1、J2、J4 轴，三个参数的单位是°，对于 J3 轴，单位是 mm。

> ✧ **小技巧：**单步功能用于手动模式下高精度教导位置，连续功能用于手动模式下粗略教导位置。



1.2.3 倍率获取/修改

倍率：手动倍率或者自动倍率，指当前相对于参数设定里面的速度百分比。倍率的使用说明：

（1）倍率变量是全局的，即手动和自动运行都调用同一倍率；

（2）倍率影响手动和自动运行的实际速度；

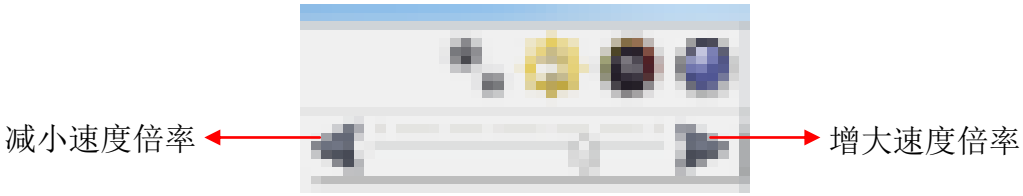
（3）手动实际速度是手动速度乘以手动倍率，例如：参数里设置的 J2 轴的手动速度为 200，如当前的手动倍率为 50%，则当前 J2 的手动速度为 200*50%=100；

（4）自动实际加速度为当前程序的速度乘以此倍率；

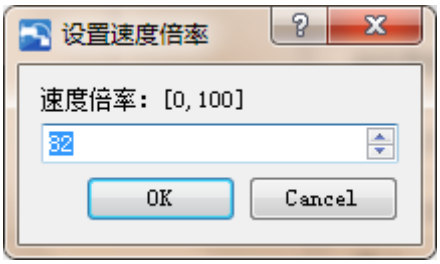
（5）手动和自动速度需在参数界面设置，包括插补速度以及关节速度。

倍率的修改可通过以下两种方式改变：

方式 1：鼠标左键点击“ ”图标，则可弹出倍率设置菜单，通过左右按键实现倍率的增大或减小，如图所示。



方式 2：鼠标右键点击“ ”图标，弹出“设置速度倍率”弹窗。可在弹框手动输入 0~100 的整型数字，也可以通过 按键微调倍率。



> ✧ **小技巧：**当前程序的速度是程序中设定的速度百分比乘以参数中设定的速度。

1.2.4 实际/离线位置切换

两齿轮的分开和闭合分别对应了机器人的离线位置和实际位置两种状态。

点击“ ”按钮，可切换机器人离线位置和实际位置转换。两齿轮分开（ ），则记录的是离线位置；两齿轮闭合（ ），则记录机器人实际位置。

> ✧ **小技巧：**两齿轮闭合应用于机器人处于轻拽模式下记录机器人实际位置。

1.2.5 机器人模式切换

机器人有三种模式：非使能模式、使能模式和轻拽模式；非使能模式和使能模式可用于自动也可用于手动运行模式；轻拽模式只适用于手动运行模式。机器人“ ”图标，用于切换机器人三种模式：

（1）默认模式为非使能模式（图标颜色为灰色）；

（2）轻触机器人图标，可切换到使能模式（图标由灰色变成绿色）；

（3）长按机器人图标，可切换到轻拽模式（图标由灰色变成黄色）；

（4）三种模式的切换都是相对于非使能模式而言的。

非使能模式：机器人电机处于非工作状态

使能模式：机器人电机处于工作状态

轻拽模式：机器人易于手动操作移动

> ✧ **小技巧：**手动模式下若想快速示教位置，可将机器人切换到轻拽模式。轻拽模式切换到非使能模式可通过单击该图标。

1.2.6 权限管理

众为兴机器人系统规定了不同用户对机器人的操作权限。众为兴机器人权限管理包含工人、操作员、管理员、厂家 4 个等级，其中工人权限最低，厂家权限最高。点击权限管理“ ”可进入【权限管理】设置界面，如下图所示：

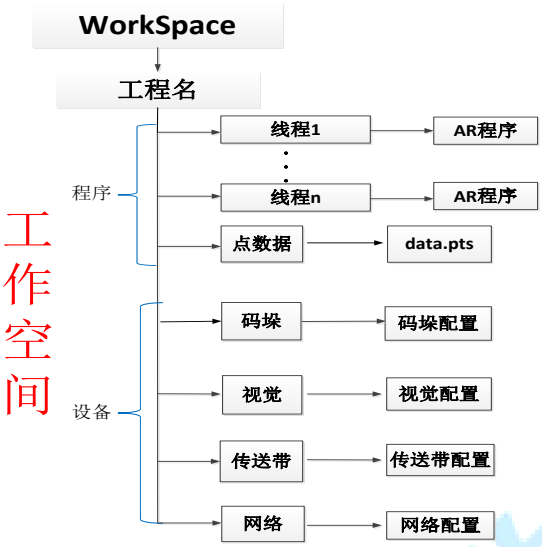
- 工人只能操作示教器上的一些按钮操作（启动、暂停、停止程序运行，修改运行倍率, 备份参数等），不能修改任何参数；注意：众为兴机器人若关机前管理权限为操作员及以上的权限，开机默认为操作员模式；若关机前管理权限为工人模式，开机保持为工人模式。
- 操作员在工人权限的基础上可以修改部分参数以及系统升级更新，例如：【参数配置】中的参数，但不能修改【安装配置】中的参数；
- 管理员在操作的权限基础上可以修改安装向导里的参数以及备份还原操作，但不能进行清原点、运动学标定等操作。
- 厂家为最高权限，可以享有所有操作权。

✧ **小技巧：**注销操作只能从高权限降级到工人权限；在高权限可以获取低权限的登录密码信息。

✧ **小技巧：**登录权限密码可咨询厂家获取。

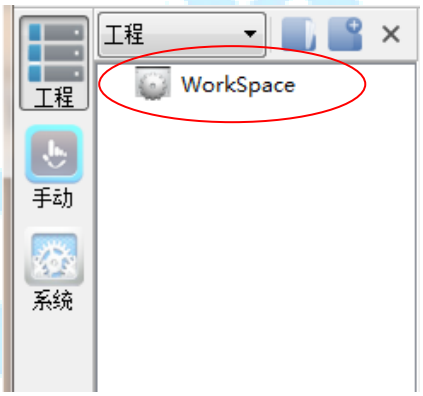
1.3 工程空间操作

QC410 控制器编程主要围绕工作空间展开，如下图所示：



1.3.1 工作空间：WorkSpace

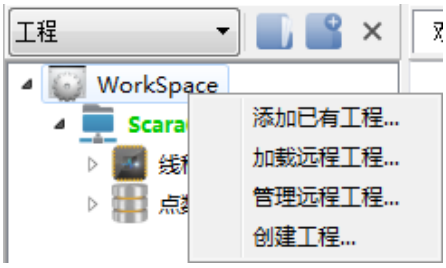
点击 ARStudio 软件界面左侧的工具 即可进入 WorkSpace 界面：



每个工作空间中默认包含一个工程“Scara0”，每个工程内默认包含一个线程和一个 data.pts 点位表。机器人项目是以工程形式来管理，工程包含有外围设备的配置（视觉、网络、编码器、码垛、传送带），程序的编写（各个线程任务属性）。

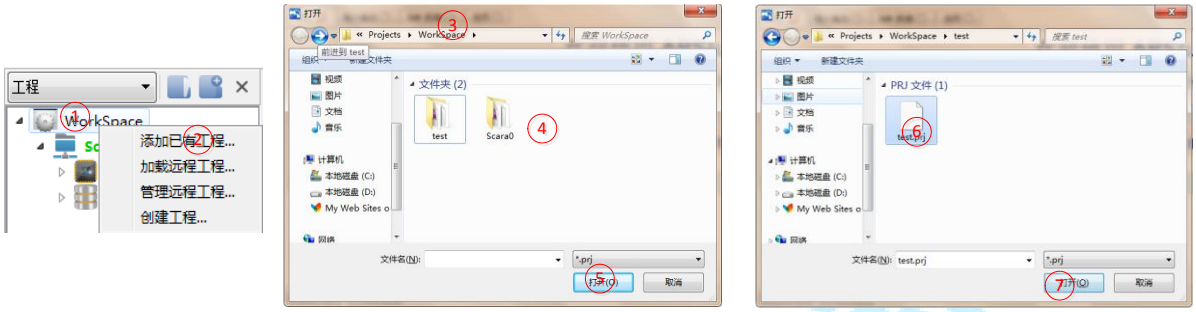
1.3.1.1 工作空间 WorkSpace 操作

工作空间操作包含添加已有工程、加载远程工程、管理远程工程、创建工程 4 项。

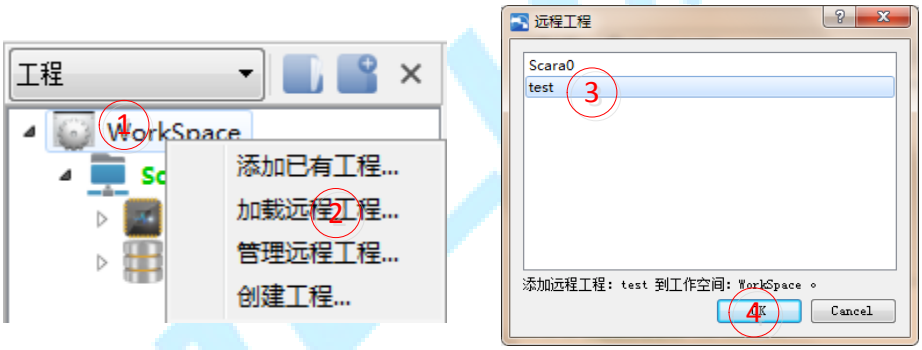


➢ 添加已有工程

“添加已有工程”是将指定路径下的机器人工程导入到 WorkSpace 中。步骤为：鼠标右键单击工作空间名字。例如 WorkSpace ----- 选择“添加已有工程”弹出【打开】对话框 ----- 搜索被添加工程所在的路径 ----- 选中**.prj** 文件 ----- 点击“打开”。

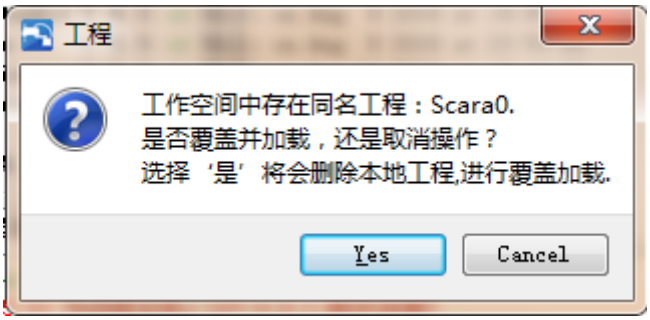


➢ 加载远程工程

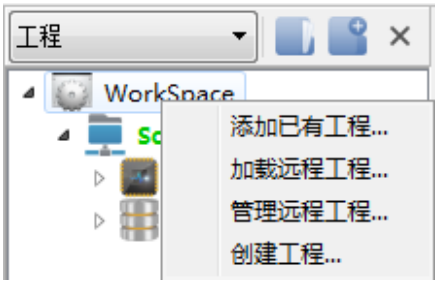


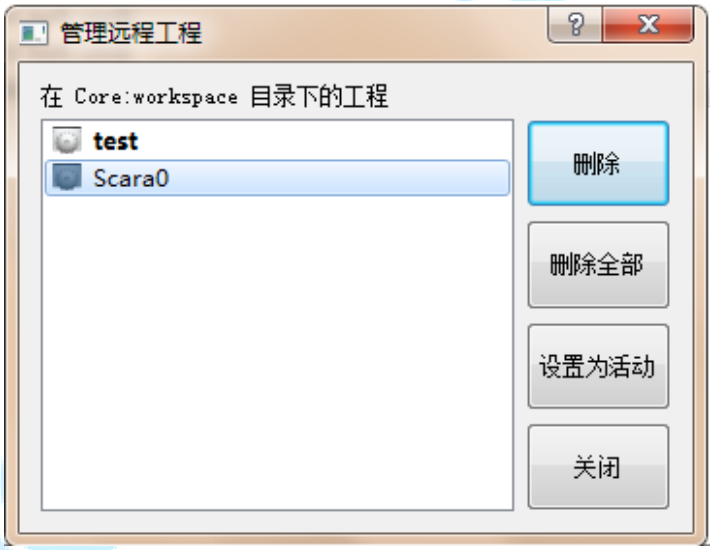
“加载远程工程”指的是从 QC410 控制器中加载指定的工程到 ARStudio (本地)中运行调试。远程工程存放在 QC410 控制器的 /home/ARApp/bin/workspace 中（可通过 WinSCP 软件访问 QC410 控制器中的磁盘）。

> ✧ **小技巧：**加载远程工程时若 ARStudio 的工作空间 WorkSpace 中存在有同名工程，则会给出“工程”弹框提示，用户可选择“是覆盖并加载还是取消操作”。



➢ 管理远程工程





删除：删除 QC410 控制器 Core:workspace 目录下指定的某一个工程；

删除全部：删除 QC410 控制器 Core:workspace 目录下所有的工程；

设置为活动：适用于 QC410 控制器脱离 ARStudio 单独运行时使用，使用的条件：

（1） 要运行的工艺工程在 Core.workspace 存在，即将 ARStudio 工作空间中的工程上传到 QC410 控制器中；

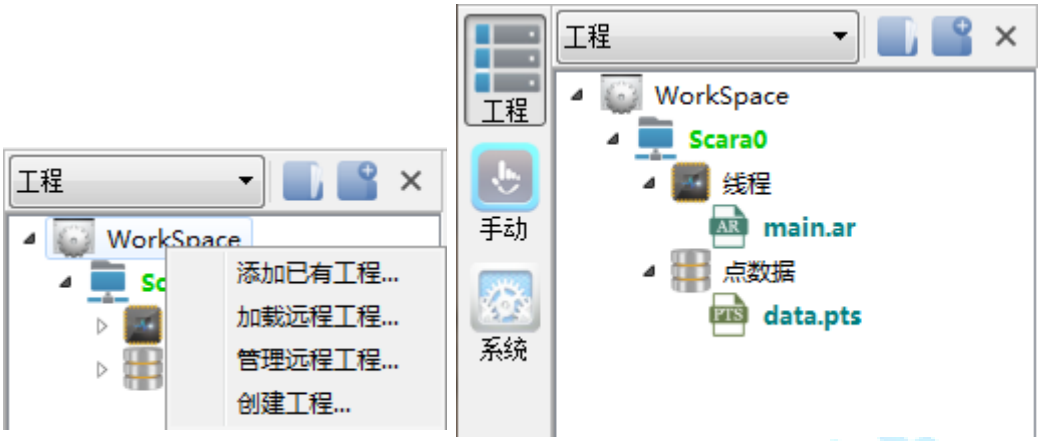
（2） 此工程设置为活动；

（3） 此工程中的线程设置为“开机自启动”或通过外部输入信号启动工程运行。

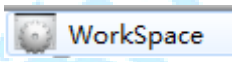
关闭：关闭【管理远程工程】对话框。

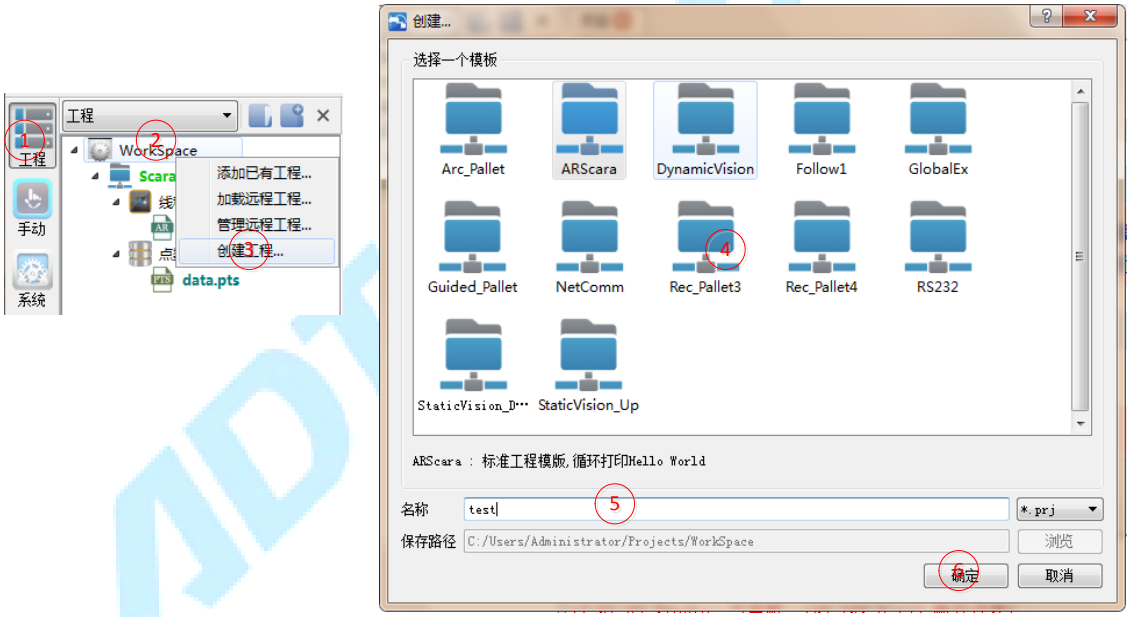
➢ 创建工程

在这里，我们通过建立一个最小工程来演示工程的创建，最小工程包含有一个线程以及一个点位数据表。

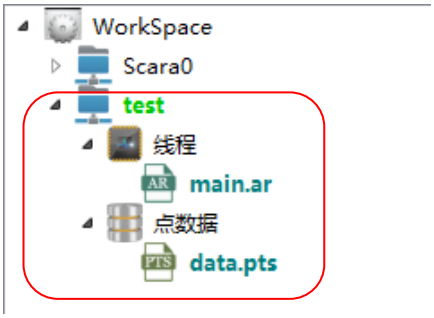


新建最小工程步骤如下：

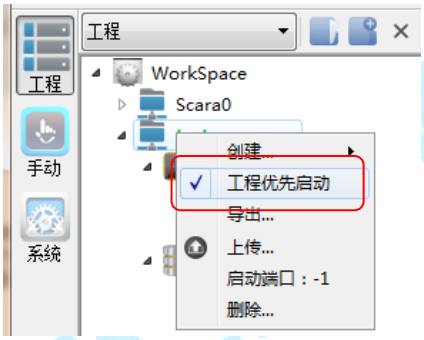
（1）点击 图标进入【工程】界面-----鼠标右键单击“  ”弹出菜单-----选择“创建工程”进入【创建】界面-----从 QC410 自带的模板程序中自选一个模板工程-----修改选中模板的名字（可选项）-----点击“确定”。



这样，一个最小工程的框架就创建成功。接下来要做的就是程序的编写以及点位的示教。

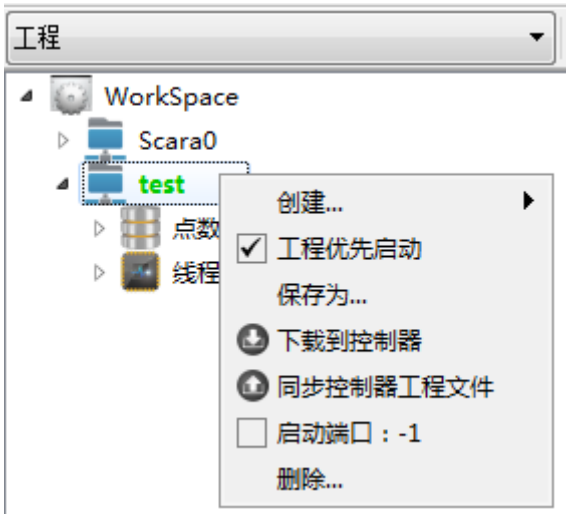


> ✧ **小技巧：**新建的工程默认会被优先启动，优先启动的工程名为高量绿色显示，未被优先启动的工程名为黑色字体显示。
>
> ✧ **小技巧：**当 WorkSpace 中有多个工程，一次有且只有一个工程会被优先启动。



1.3.1.2 工程操作

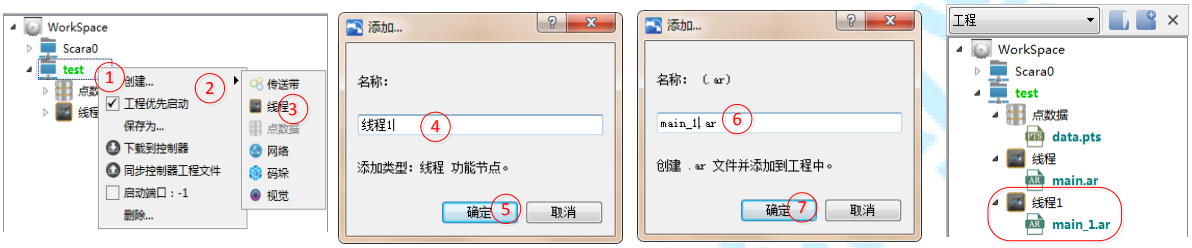
工程操作创建（线程、点数据、码垛、视觉、编码器、传送带、网络）、工程优先启动、保存为、下载到控制器、同步控制器工程文件、启动端口、删除等操作。接下来依次对这几个操作做个详细说明：



➢ 创建

“创建”操作用于添加一些外部设备到当前工程下，例如，对于多线程工艺，则须在当前工程下创建一个新的线程，步骤如下：

鼠标右键单击工程名，例如：test ----- 选择“创建”----- 选择“线程”，弹出【添加】对话框 ----- 输入新建线程的名字 ----- 点击“确定”弹出另一个【添加】对话框 ----- 输入新建线程中 AR 程序的名字（新建线程默认会自动生成一个 AR 程序，名字须人为指定）----- 点击“确定”，这样新的线程就创建成功了。

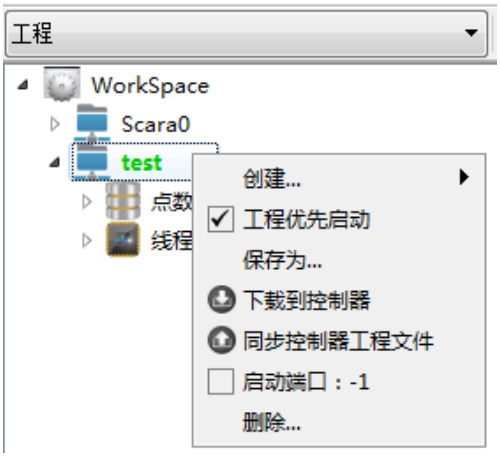


其它创建操作跟创建线程步骤基本一致，这里不再一一列举。

> ✧ **小技巧：**每个工程中可以包含多个线程，但是只能有一个点数据、码垛、视觉、编码器、传送带或网络（若当前工程中已包含其中的某一个或多个，创建菜单中对应的那一项就为灰色显示，无法操作。）

➢ 工程优先启动

“工程优先启动”操作针对于当前工作空间 WorkSpace 中包含有多个工程，优先启动哪一个工程。

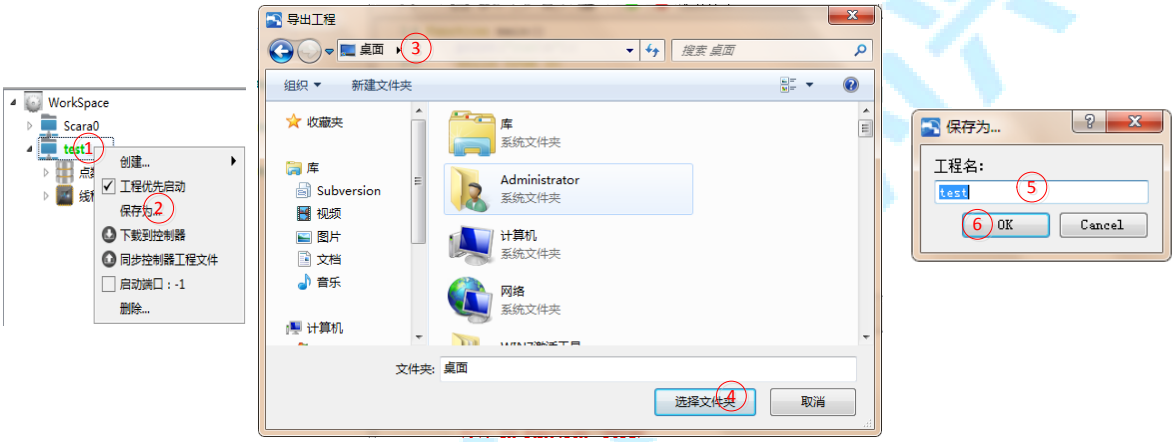


> ✧ **小技巧：**新建的工程默认会被优先启动。

➢ 保存为

“保存为”操作用于将已调试好的工程打包备份保存，可用于到其它相同工艺的控制器中。导出步骤如下图所示：

鼠标右键单击工程名，例如：test ----- 选择“导出”弹出【导出工程】对话框 ----- 选择被导出工程要保存的路径 ----- 点击“选择文件夹”弹出【保存为】对话框 ----- 给被导出的工程重新命名保存(也可不修改,默认就是被导出工程的名字)-----点击“OK”。这样，工程就成功导出保存。



➢ 下载到控制器

“下载到控制器”操作是将 ARStudio 中的工程下载到 QC410 控制器的 Core:WorkSpace 路径。在 ARStudio 中运行一次当前优先启动的工程也会自动将该工程下载到到 QC410 控制器的 Core:WorkSpace 路径。

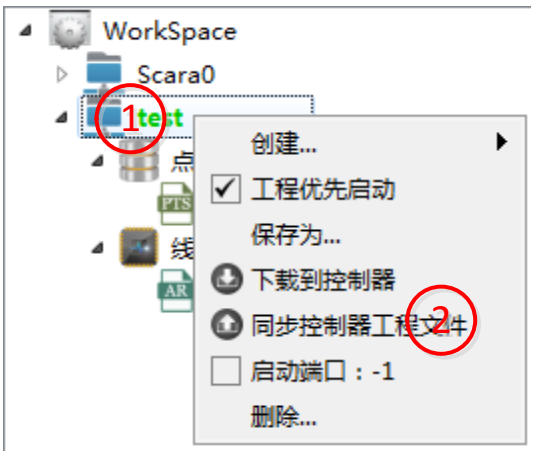
➢ 同步控制器工程文件

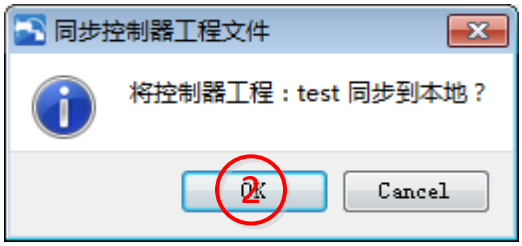
“同步控制器工程文件”操作为了避免在多台设备修改工程文件时，出现已做出的修改丢失的问题。例如：当使用示教器修改 AR 文件并传输到控制器后，这时若切换到电脑连接控制器（电脑中这个 AR 文件还是上一次修改的文件，与控制器中的被示教器修改过的文件不一致），若用户忘记了上一次在示教器中做出了更改而直接运行工程或下载工程到控制器，就会使得示教器修改的数据丢失。

此操作包含两种情形，一是主动同步，二是被动提示

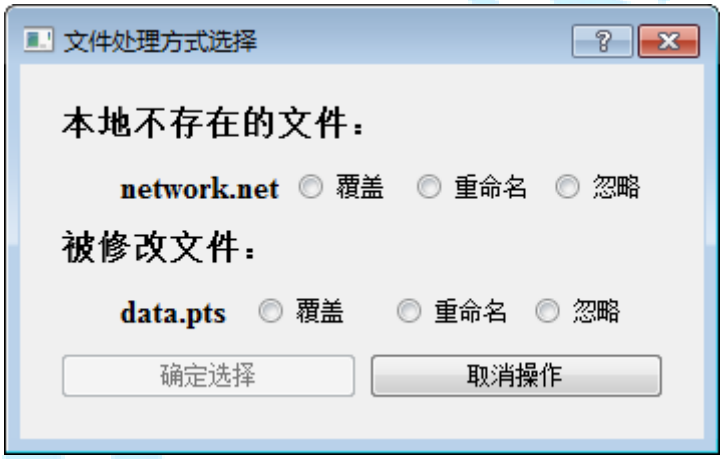
情形 1：主动同步

右键工程节点，再点击同步控制器工程文件：





点击“OK”，若控制器中的工程有通过其它设备（示教器或电脑）上做过更改，则会弹出下列对话框，对话框中显示的内容有**本地不存在的文件**和**被修改文件**，以及对文件的处理方式：



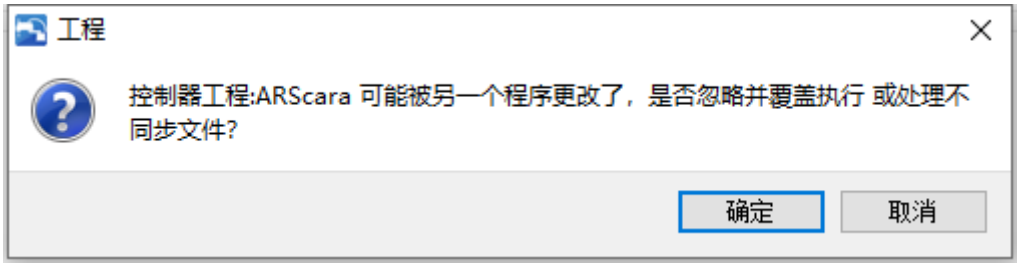
覆盖：下载控制器文件并覆盖此文件；

重命名：下载控制器文件到工程目录，并修改文件名。修改策略为名称+_temp。例如 data.pts，重命名为 data_temp.pts.

忽略：不同步控制器中被修改过的文件到工程目录；

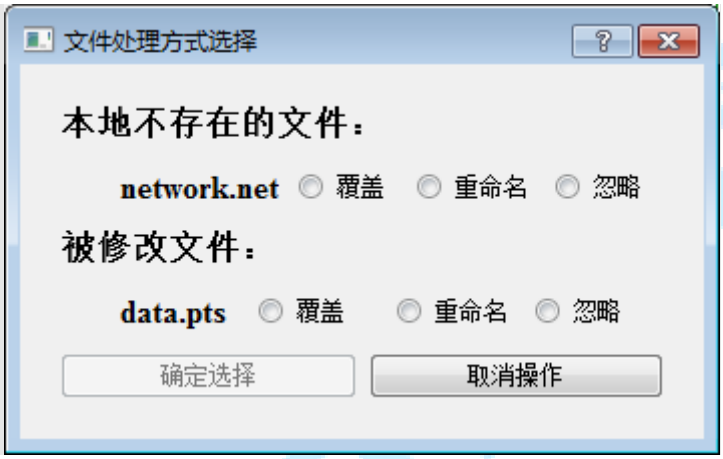
情形 2：被动提示

为避免用户忘记做出的修改或误操作，在 ARStudio 中，点击运行 或 debug 或下载到控制器操作时，若检测到文件不同会做出以下提示：



确定：忽略提示，继续运行本地工程文件；

取消：处理不同步文件，会弹出下列处理框：



覆盖：下载控制器文件并覆盖此文件；

重命名：下载控制器文件到工程目录，并修改文件名。修改策略为名称+_temp。例如 data.pts，重命名为 data_temp.pts.

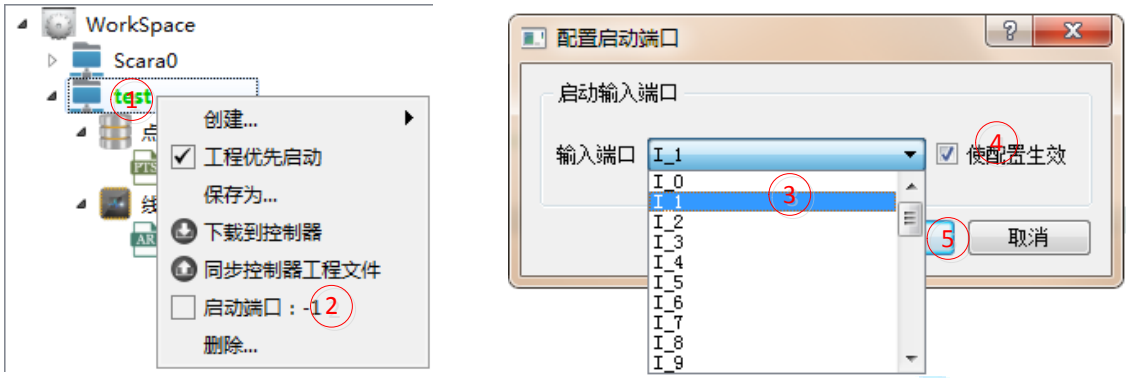
忽略：不同步控制器中被修改过的文件到工程目录；

> ✧ **小技巧：**当执行“同步控制器工程文件”操作时，线程中新建的 AR 文件以及新创建的线程中的 AR 文件没有“添加到工程”操作，只能选择“下载到本体”或“忽略”操作。

➢ **启动端口**

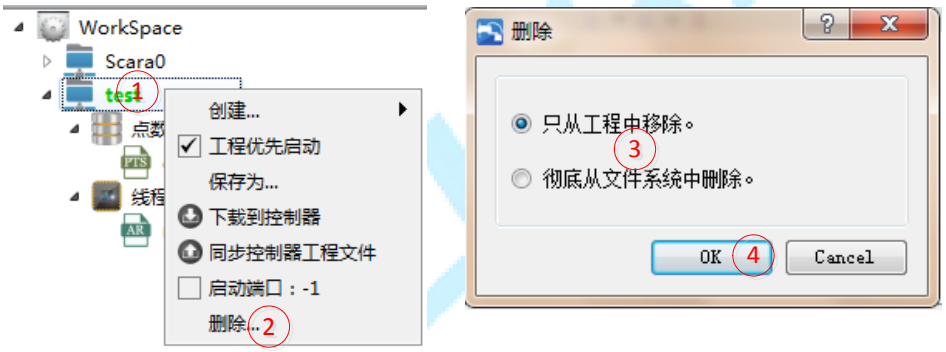
“启动端口”使用场景：通过设定的外部启动信号（输入信号）用来切换优先启动的工程。

操作步骤：鼠标右键单击工程名，例如：test ----- 选择“启动端口”弹出【配置启动端口】对话框 ----- 设定输入端口号（根据实际外部接线来设定）----- 勾选“使配置生效”----- 点击“确定”。这样，工程启动输入端口配置成功了。



> ✧ **小技巧：**工程中设定的启动输入端口仅对此工程有用。

➢ **删除**

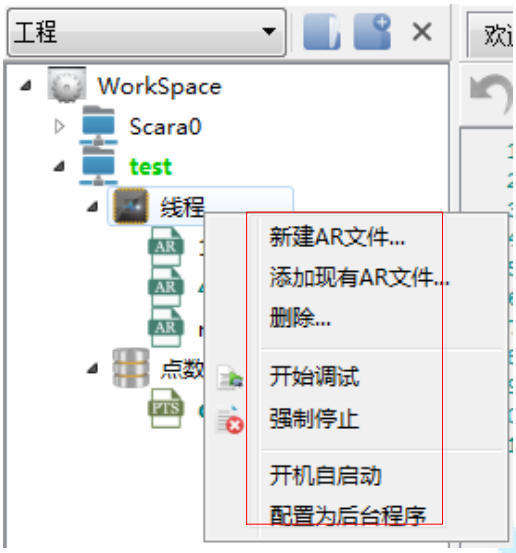


只从工程中移除：被操作的工程只是从 ARStudio 显示的工作空间 WorkSpace 中不再显示，但是工程仍然被保留在 WorkSpace 所在的文件系统中。

彻底从文件系统中删除：被操作的工程从 WorkSpace 所在的文件系统中被彻底删除。

1.3.1.3 线程操作

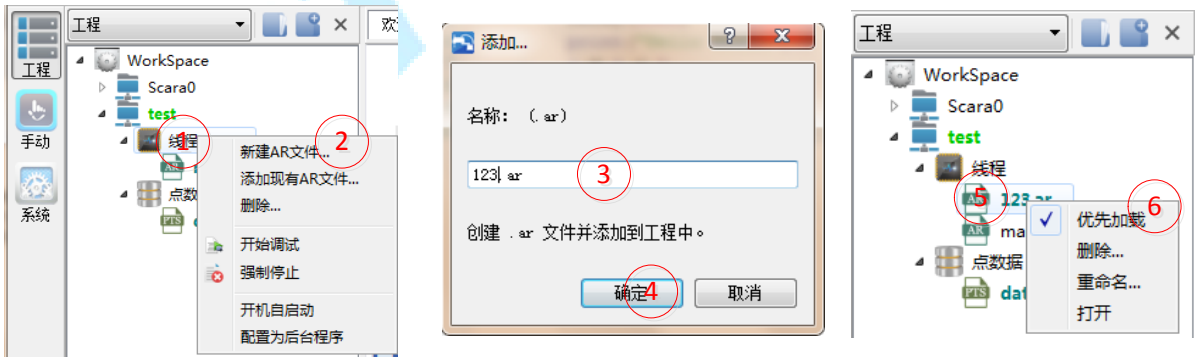
工程中线程的操作包含有新建 AR 文件、添加现有 AR 文件、删除、开始调试、强制停止、开机自启动、配置为后台程序等操作。



➢ **新建 AR 文件**

“新建 AR 文件”操作适用一个线程下添加多个 AR 程序，**注意，线程下多个 AR 程序一次只有一个会被优先加载**。新建 AR 程序操作，会在当前工程的线程下新增一个 AR 程序，之前 AR 程序保留不会被覆盖。新建步骤如下：

鼠标右键单击“线程”弹出菜单-----选择“新建 AR 文件”弹出【添加】界面-----输入新建 AR 程序文件的名字，例如 123，后缀为 .AR-----点击“确定”按钮-----鼠标右键新建程序名字，例如 123.ar ,弹出菜单-----选择“优先加载”（可选项，默认是之前使用过的 AR 程序）。这样新的 AR 程序就创建成功了。

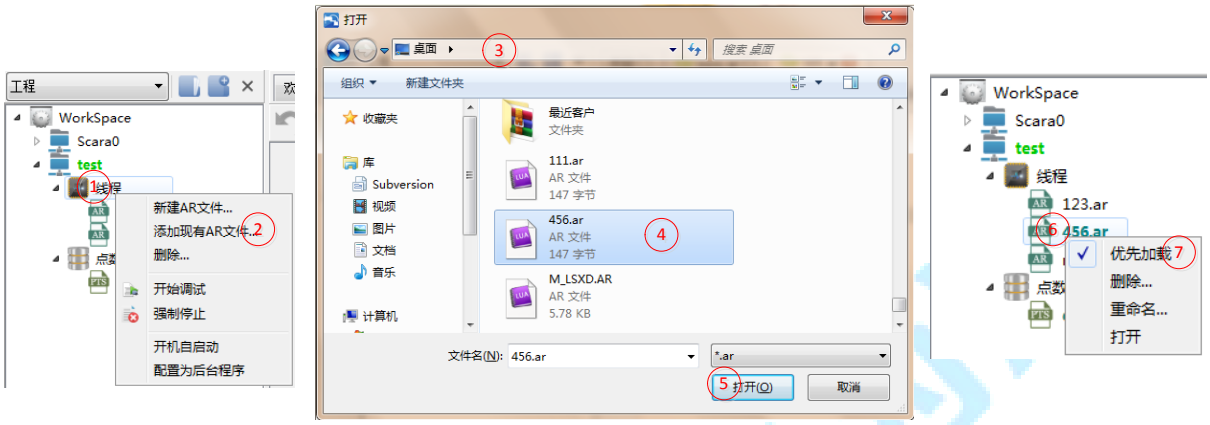


➢ **添加现有 AR 文件**

“添加现有 AR 文件”操作，适用于现场有多台进行相同工艺调试的 QC410 控制器。要加载的 AR 程序已经在某一台已成功调试，可以直接通过加载 AR 程序的方式应用于其它的控制器。步骤如下：

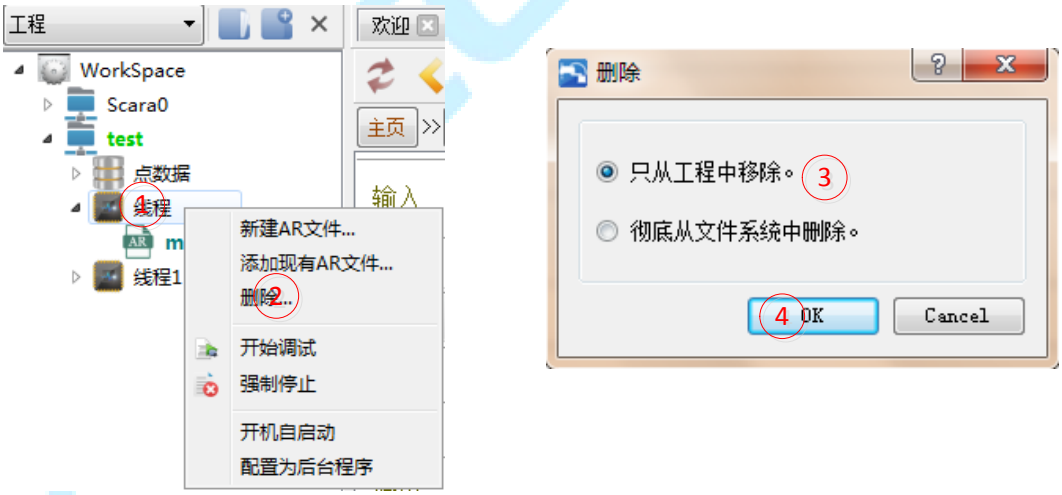
鼠标右键单击“线程”弹出菜单-----选择“添加现有 AR 文件”弹出【打开】界面

-----搜索要加载的 AR 程序路径-----选中要加载的 AR 程序，例如 456，后缀为 .AR-----点击“打开”按钮-----鼠标右键被加载的 AR 程序名字，例如 456.ar ,弹出菜单-----选择“优先加载”（可选项，默认是之前使用过的 AR 程序），这样加载 AR 程序就成功了。



> ✧ **小技巧：**新建或加载的 AR 程序不会自动指定为优先加载，须人为手动加载；被优先加载的 AR 程序为绿色高量显示，未被优先加载的 AR 程序为黑色字体显示；一个线程下一次只能有一个 AR 程序被加载。

➢ **删除**



只从工程中移除：被操作的线程只是从 ARStudio 显示的工作空间 WorkSpace 中不再显示，但是工程仍然被保留在 WorkSpace 所在的文件系统中。

彻底从文件系统中删除：被操作的线程从 WorkSpace 所在的文件系统中被彻底删除。

➢ **开始调试**

此操作启动 AR 程序进入调试模式。

➢ **强制停止**

“强制停止”使用情形为：当前线程配置为后台程序，可通过“强制停止”在停止后台线程的运行。

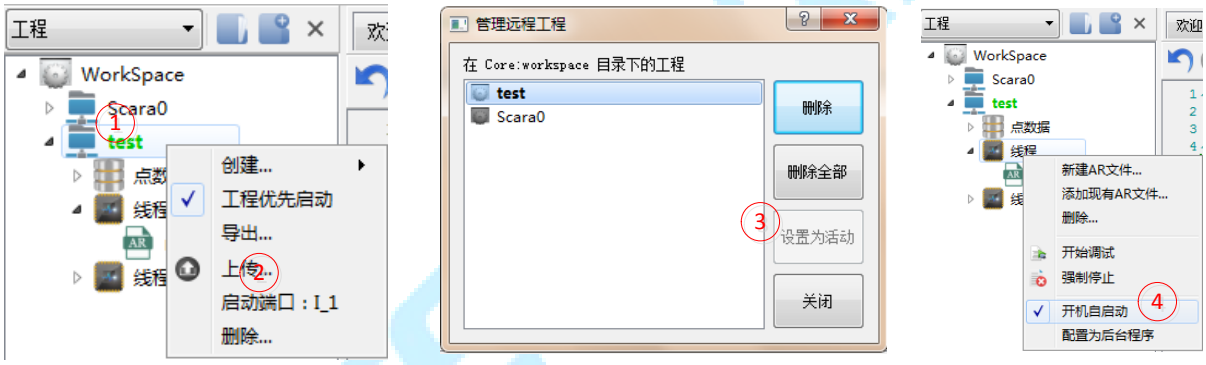
➢ **开机自启动**

“开机自启动”适用于 QC410 控制器脱离 ARStudio 单独运行时使用，使用的条件：

（1）要运行的工艺工程在 Core.workspace 存在，即将 ARStudio 工作空间中的工程上传到 QC410 控制器中；

（2）此工程设置为活动；

（3）此工程中的线程设置为“开机自启动”。这样控制器启动后设置为活动的工程会自动运行。



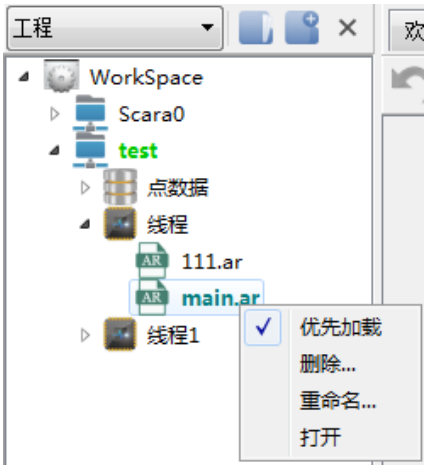
> ✧ **小技巧：**如果需要暂停或急停需利用 PC 重新连接才可以改变当前状态。因此，请在拔掉电脑之前反复测试工艺，确保可以安全正常进行时才能拔掉运行。
>
> ✧ **小技巧：**因为设置开机自启动以后，齿轮默认闭合，所以如果需要电机实际运转起来的话，需要在程序里添加 MotOn()指令。此时开机上电的话，一体机是与上位机软件脱离独立运行的。

➢ **配置为后台程序**

“配置为后台程序”是指将当前工程停止运行的情况下，该线程继续在后台运行。常见的使用情况有后台程序用来与外部设备通讯或 IO 交互等。

> ✧ **小技巧：**后台程序可通过重启 QC410 控制器或线程操作中的“强制停止”来终止后台程序的运行；
>
> ✧ **小技巧：**“开机自启动”和“配置为后台程序”可同时使用。

1.3.1.4 AR 程序操作

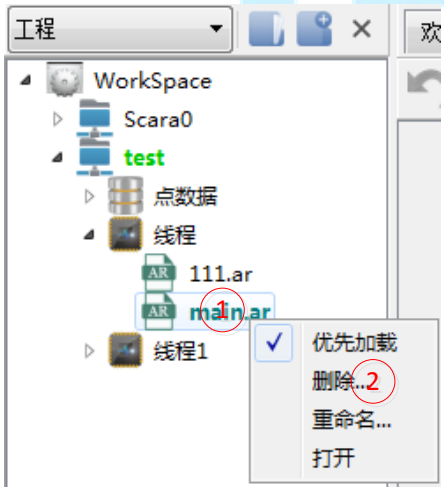


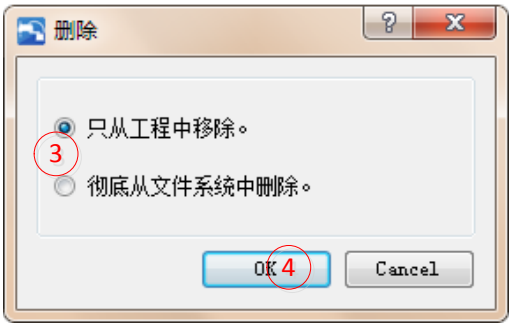
AR 程序包含优先加载、删除、重命名、打开四个选项，接下来依次作一下说明：

➢ **优先加载**

一个线程下可以包含有多个 AR 程序，但是每次有且只有一个 AR 程序会被加载运行，其它的 AR 程序可通过被优先加载的 AR 程序使用，例如，在优先加载的 AR 程序中通过 dofile 指令打开其它的 AR 程序。可通过“优先加载”用来选择需要运行的 AR 程序。

➢ **删除**



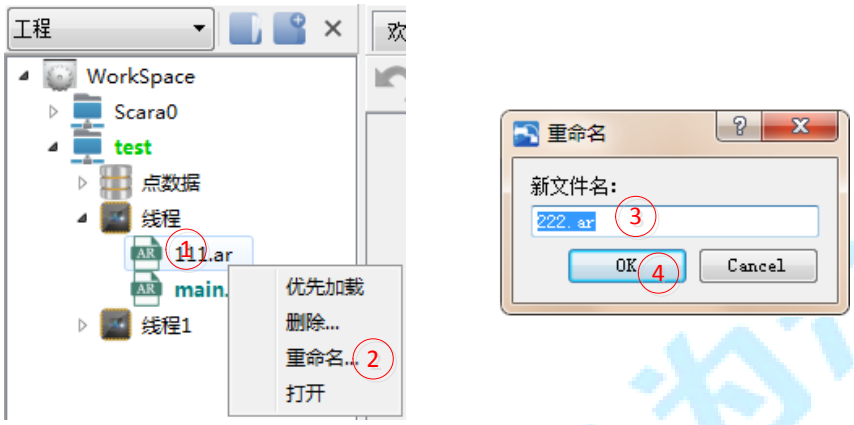


只从工程中移除：被操作的 AR 程序只是从 ARStudio 显示的工作空间 WorkSpace 中不再显示，但是 AR 程序仍然被保留在 WorkSpace 所在的文件系统中。

彻底从文件系统中删除：被操作的 AR 程序从 WorkSpace 所在的文件系统中被彻底删除。

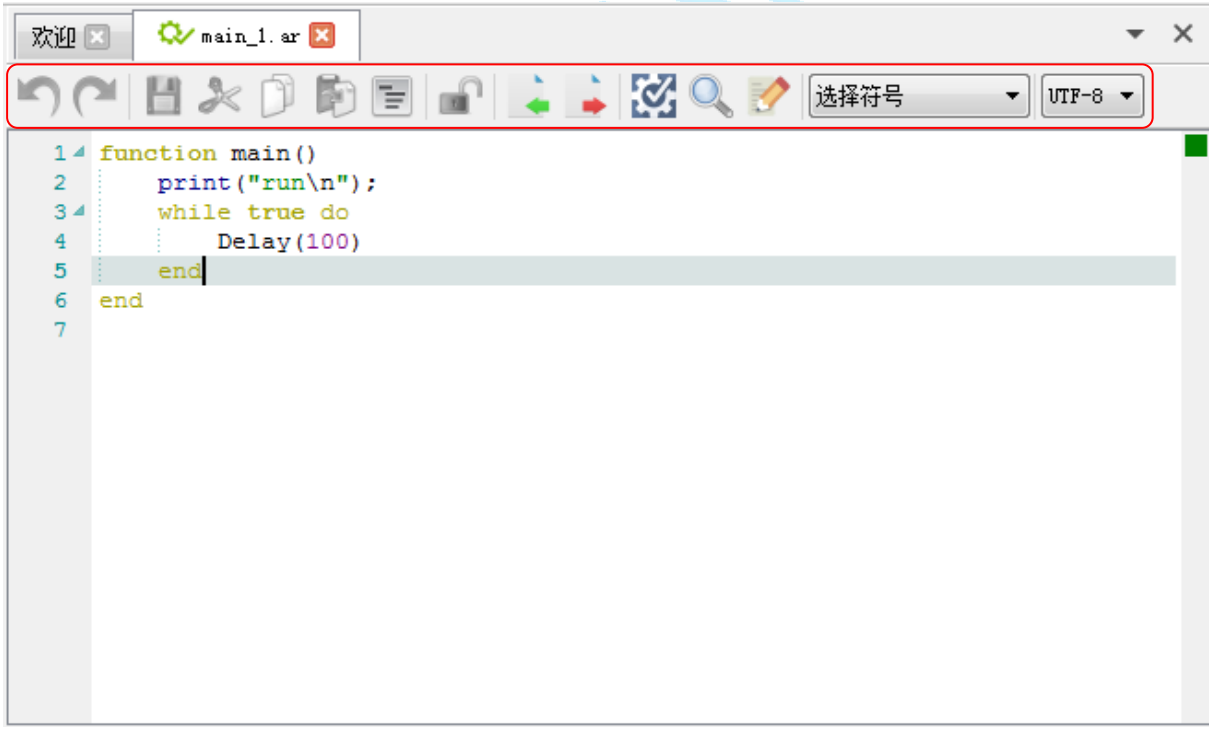
➢ **重命名**

给指定的 AR 程序重新命名，AR 程序中内容不会被修改。被修改之前的 AR 程序被覆盖。



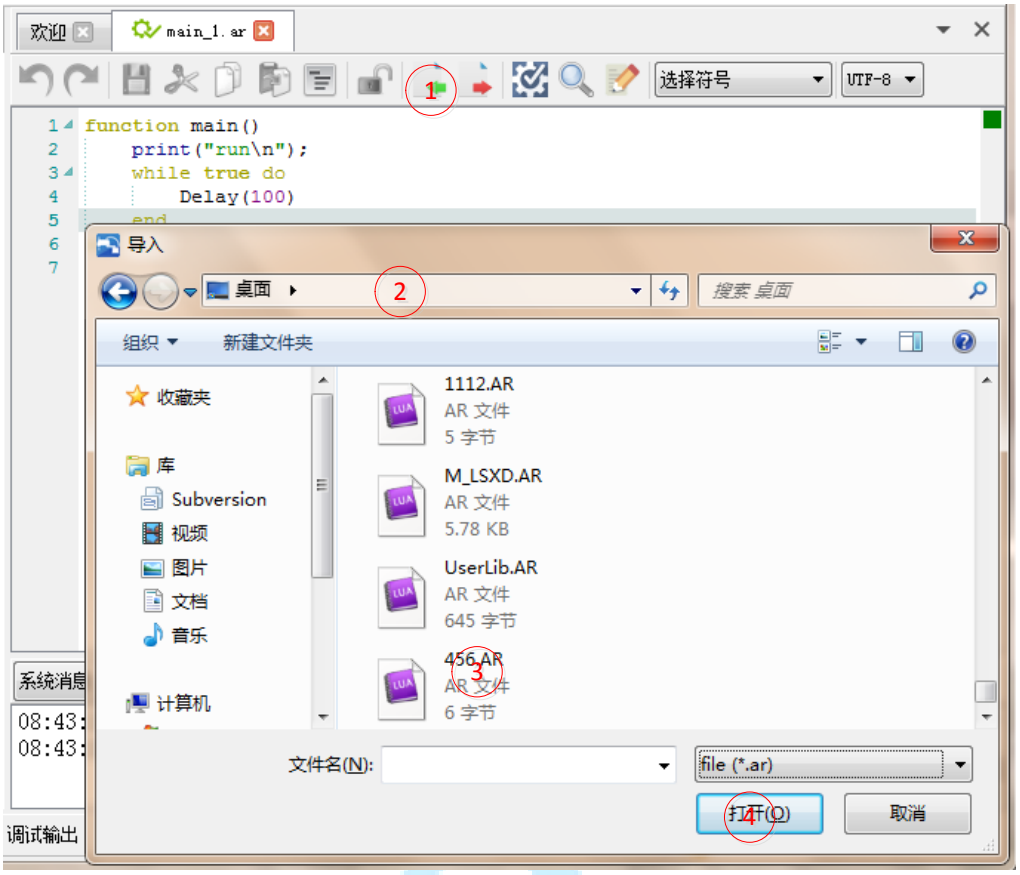
➢ **打开**

打开 AR 程序显示到 AR 编辑界面：

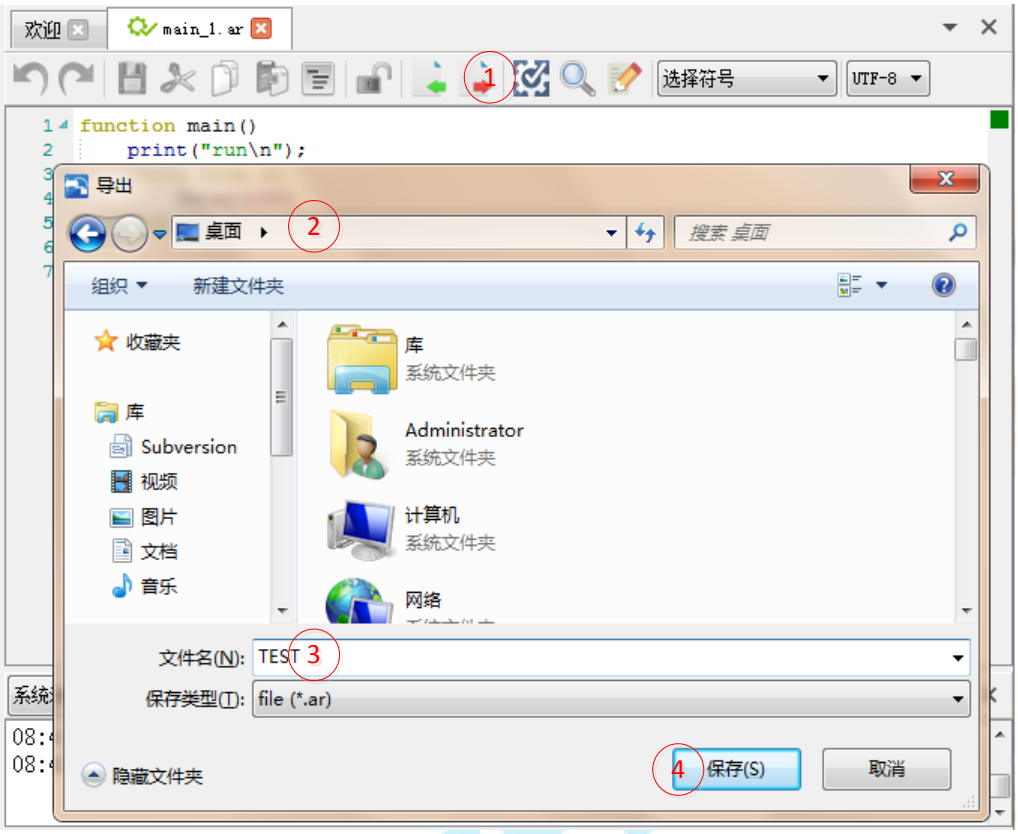


这个界面只介绍其中 4 个工具：导入、导出、插入，其它工具跟其它编程软件的功能都是通用的，这里不再赘述：

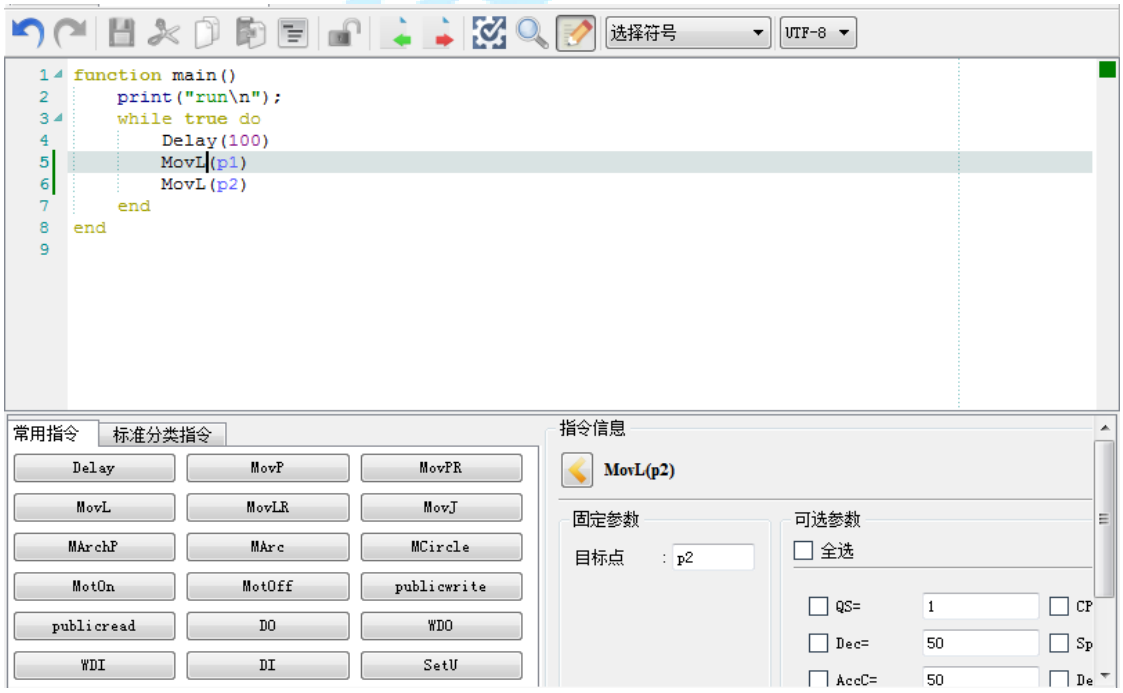
导入：从 PC 机或其它路径导入 AR 程序。注意，导入的程序只会覆盖当前正在操作的程序内容，程序的名字不会被覆盖。操作步骤如下图所标序号所示：



导出：将当前操作的 AR 程序保存到 PC 或其它设备。注意，导出的程序需要重新自定义一个名字，操作步骤如下图所标序号所示：



插入：此功能也叫做向导式编程。当编程人员初次接触机器人编程指令或不是很熟悉的情况下，可以通过插入指令的方式来编程。

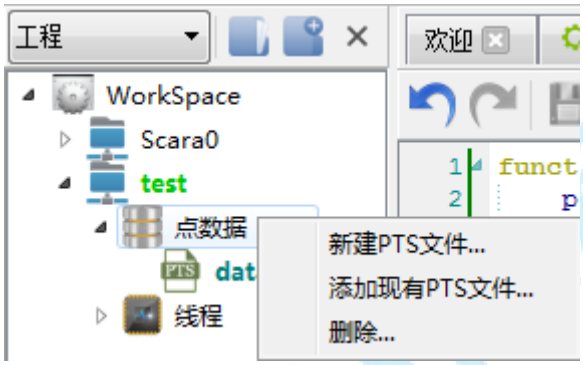


✧ **小技巧：**当文件为可读或格式不对时，此时文件是不可编辑的；

> ✧ **小技巧：**AR 程序中的指令区分大小写；
>
> ✧ **小技巧：**输入指令的首字母可以关联出指令的一些关键信息；指令的具体用法请参考 AR 语言手册说明书。

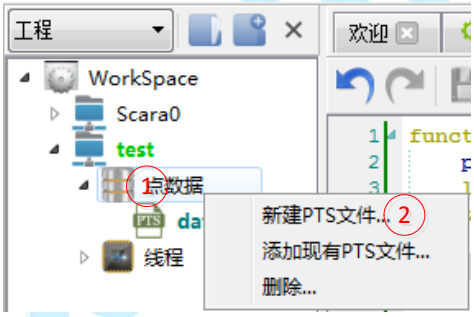
1.3.1.5 点数据操作

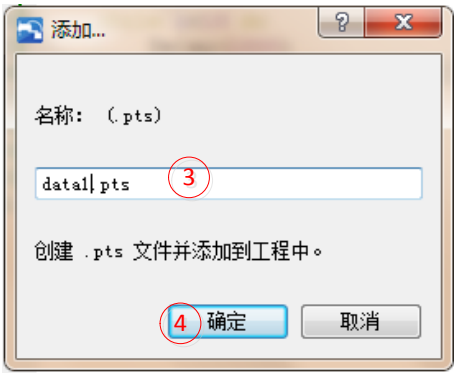
“点数据”包含新建 PTS 文件、添加现有 PTS 文件、删除 3 项操作。



➢ **新建 PTS 文件**

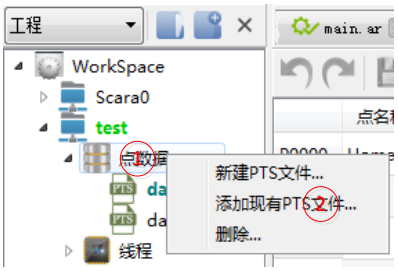
“点数据”列表中可以包含多个 .PTS 数据表，用户可通过“新建 PTS 文件”操作来添加新的数据表。

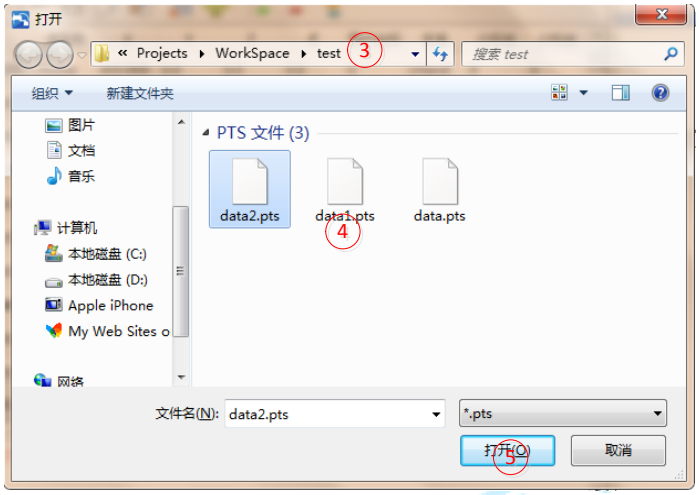




➢ **添加现有 PTS 文件**

“添加现有 PTS 文件”是指从指定路径导入 .PTS 格式的数据表到当前工程下。

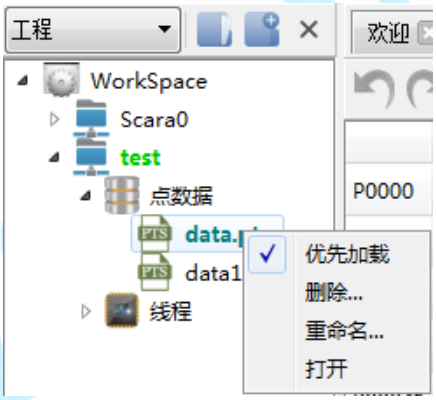




➢ **删除**

与删除工程、线程操作一致，这里不再赘述。

➢ **数据表 .PTS 操作**

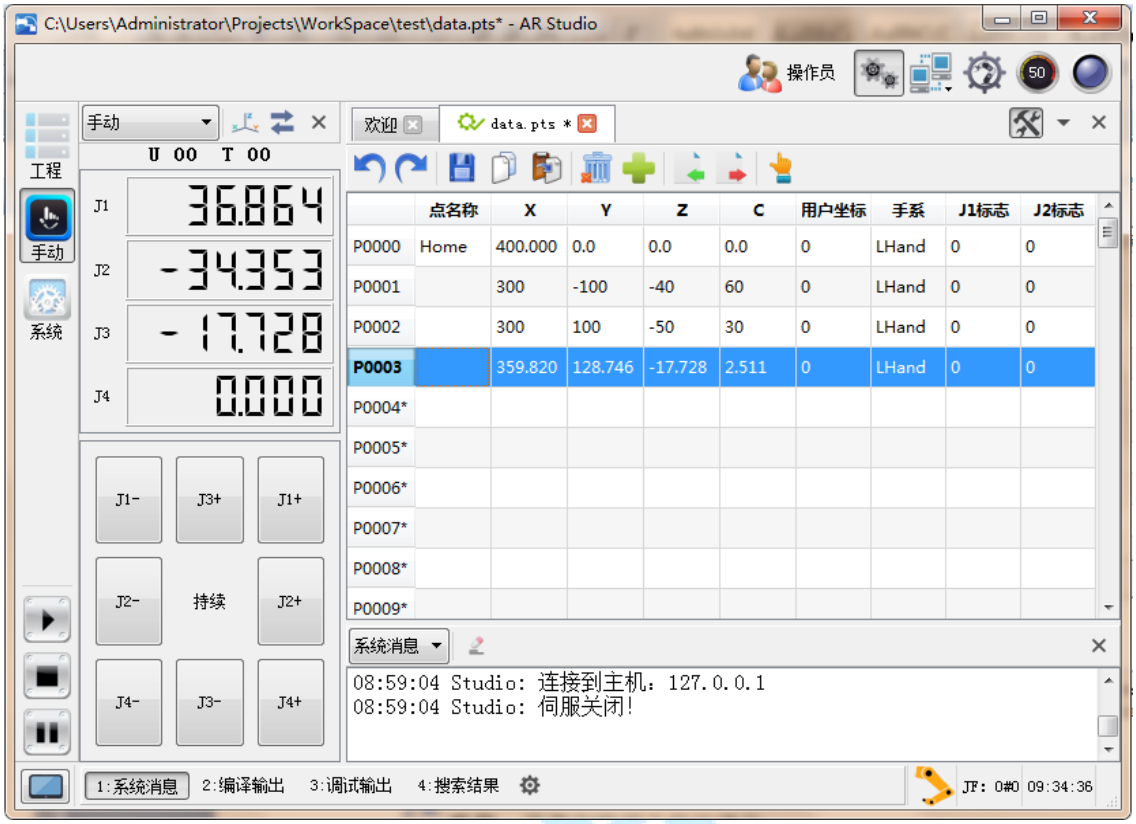


优先加载：一个点数据下可以包含有多个.pts 点位表，但是每次有且只有一个点位数据表会被加载运行。可通过“优先加载”用来选择目标数据表。

删除：删除点数据中某一个.pts 数据表

重命名：给指定的.pts 数据表重新命名，数据表中内容不会被修改。被修改之前的数据表被覆盖。

打开：打开指定的某一个.pts 数据表：



撤销：撤销之前的操作；

重做：恢复到撤销之前的操作;

保存：保存点位表的操作生效；

复制：复制点位表中某一行、某一列或某一个数据；

粘贴：将复制的内容粘贴到点位表中的指定位置；

删除：删除点位表中某一行或某几行的数据；选中要删除的行（蓝色高量显示），点击此按钮即可删除。

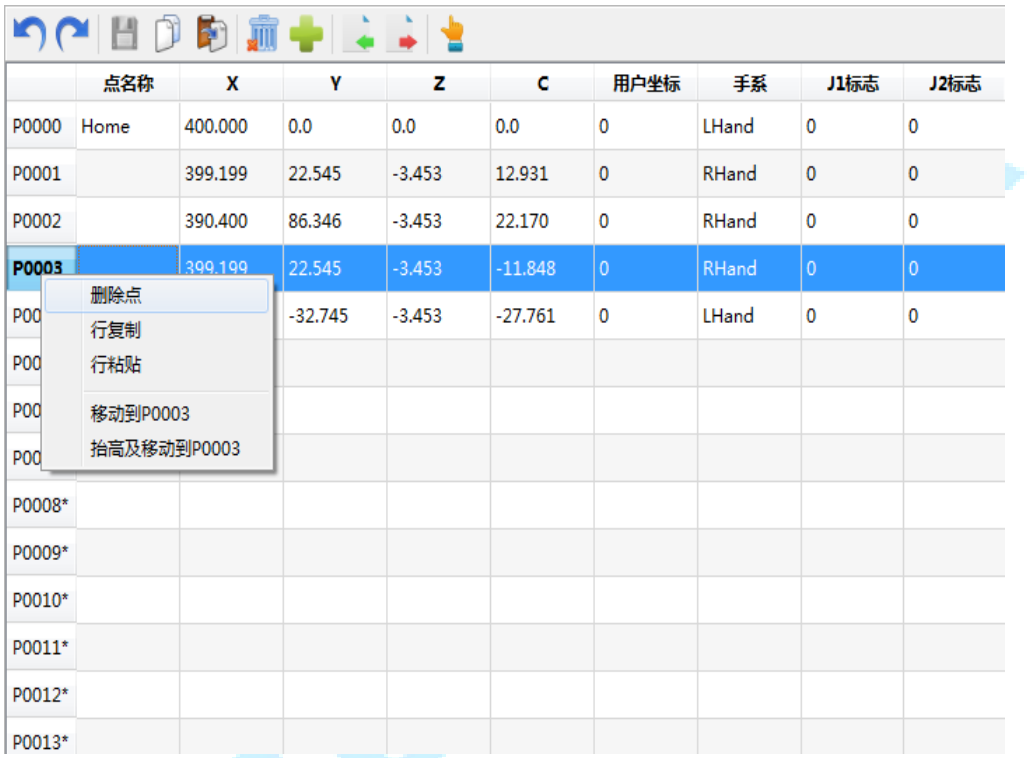
增加新行：.pts 数据表中默认只有 P0000~P0099 共 100 个数据点位，若点位数量不足，可通过点击此按钮 来增加数据点位，点击一次可添加 20 行数据；

导入：将指定的.pts 数据表从 PC 机或 U 盘导入到当前的点位表。当前点位表中的数据会覆盖，但是点位表的名字保留不变。

导出：将当前操作的点位表导出到 PC 机或其它设备备份保存。

示教：将机器人当前的笛卡尔坐标写入到正在操作的点位数据表中。选中某一个点位（蓝色高量显示），点击 即可示教成功。

对已示教的点可以进行点处理。例如点击“P0003”，则弹出“点处理”对话框，包括删除该点，行复制，行粘贴和移动到该点。



✧ **小技巧：**P0000 点是固定点，不可修改，对应的是机器人的原点位置。可通过移动到 P0000 点快速回到到机器人零点位置；

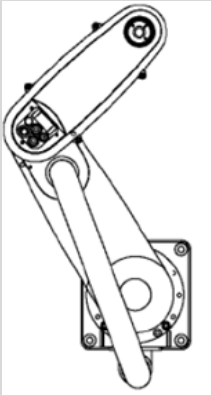
✧ **小技巧：**对点位表进行操作后，须点击保存 按钮生效；

✧ **小技巧：**点位数据表中一个点位信息包括点名称（别名）、**X**（x 坐标）、**Y**（y 坐标）、**Z**（z 坐标）、**C**（c 坐标）、**用户坐标**(坐标基于的用户)、**手系**、**J1 标志**、**J2 标志**、**描述**，对应的说明如下：

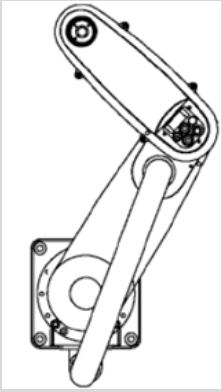
手系：左手系（标志位为 0）、右手系（标志位为 1）

点名称：须以数字、字母或者两者组合来命名，点名称可直接在 AR 程序中使用，例如运动指令；

描述：点位的说明备注，可以用中文备注；



左手系



右手系

J1 轴的标志位定义：

当 -180<J1<180 时， J1 Flag=0 ;

当 180<J1<360 或 -360<J1<-180 时， J1 Flag = 1

J2 轴的标志位定义：

当 -180<J2<180 时， J2 Flag=0 ;

当 180<J2<360 或 -360<J2<-180 时， J2 Flag = 1

1.3.2 试运行

程序编译无误，则可试运行。

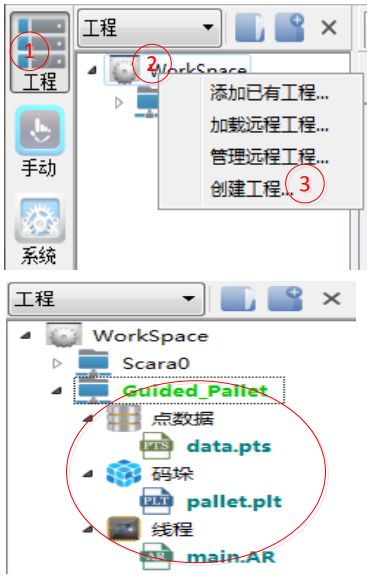
运行一个程序，安全起见的话，首先应该离线仿真，即程序运行，机器人不运动，可以通过轨迹跟踪界面监控程序逻辑性以及点位是否能够到达；同时机器人运动速度不要设置过大，速度倍率设定 20%适宜。

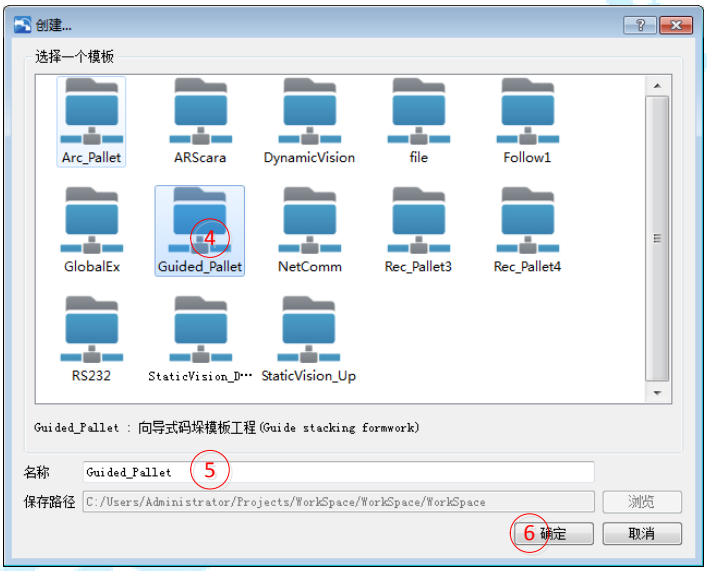
（1）离线运行：不使能，机器人图标“ ”为灰色，则为离线仿真，点击“启动” 按钮，若程序运行正常则可切换为在线运行；

（2）在线运行：上使能 ，点击“ ”，程序运行的同时，机器人运动到对应的目标点。可通过“ ”按钮暂停程序运行，或者“ ”按钮停止程序运行。

1.3.3 配置码垛

1. 点击 进入工程界面 ----- 鼠标右键点击“工作空间 WorkSpace”弹出菜单列表 -----选中“创建工程”弹出【创建对话框】-----选中向导式码垛模板工程-----修改要导入工程模板的名字（可选项，默认名字为 Guided_Pallet）----- 点击“确定”，则配置码垛工程即可导入成功。





2. 鼠标右键选中“pallet.plt”，在弹出的菜单中选中“打开”，即可进入【码垛配置界面】。码垛配置界面如下，其中 分别代表：【保存】、【添加码垛】、【删除当前码垛】、【验证当前码垛配置】；

3. 选择需要配置的码垛号，默认只有一个 PLT0。【添加码垛】之后可以看到对应码垛名称的选项。码垛个数最多不超过 10 个，当满十个时会提示“到达码垛最大配置数”；
4. 【当前个数】默认为 1，可以修改此参数修改当前码垛的个数；
5. 根据当前码垛形状选择【矩形码垛】或者【圆弧码垛】。

1.3.3.1 三点矩形码垛

Porg	码垛原点位置
Px	码垛行方向的最后一个点
Py	码垛列方向的最后一个点
Pz	码垛层方向的最后一个点（根据是否多层进行勾选）

注：其中每个点位均含 X，Y，Z，C，H 五个元素，当 H=0 时是左手系，H=1 时是右手系。通过【选择】可以从当前点位表中直接选择已示教好的点，或者当下进行【示教】即示教一个新的点位。填写对应的 X 方向个数，Y 方向个数，多层时填写 Z 方向个数。

其中默认标志位为“0”，即保留为码垛点 ;选中不码垛点，点击移除，标志位更改为“1”。

当需要某一码垛点的实际位置时，可以通过【保存&移动到该点】看区域是否到达或者是否需要码垛。设置完成后请点击【保存】。

1.3.3.2 四点矩形码垛

Porg	码垛原点位置
Px	码垛行方向的最后一个点
Py	码垛列方向的最后一个点
Pxy	码垛对角线方向的最后一个点
Pz	码垛层方向的最后一个点（根据是否多层进行勾选）

注：其中每个点位均含 X，Y，Z，C，H 五个标志位，当 H=0 时是左手系，H=1 时是右手系。通过【选择】可以从当前点位表中直接选择已示教好的点，或者当下进行【示教】即示教一个新的点位。填写对应的 X 方向个数，Y 方向个数，多层时填写 Z 方向个数。

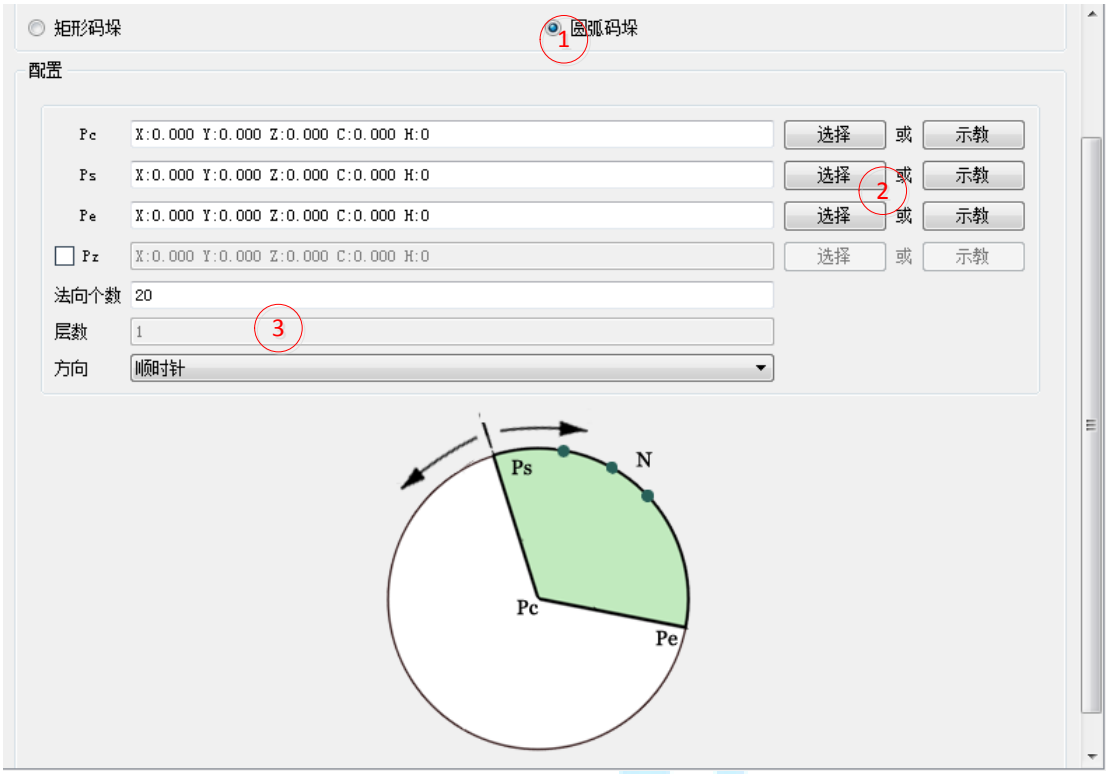
其中默认标志位为“0”，即保留为码垛点 ;选中不码垛点，点击移除，标志位更改为“1”。

当不确定是哪一点时，可以通过【保存&移动到该点】看区域是否到达或者是否需要码垛。设置完成后请点击【保存】。

1.3.3.3 圆弧码垛

Pc	圆弧码垛原心位置
Ps	圆弧码垛起始点位置
Pe	圆弧码垛终点位置
Pz	码垛层方向的最后一个点（根据是否多层进行勾选）

注：其中每个点位均含 X，Y，Z，C，H 五个标志位，当 H=0 时是左手系，H=1 时是右手系。通过【选择】可以从当前点位表中直接选择已示教好的点，或者当下进行【示教】即示教一个新的点位。法向个数即沿圆弧圆周方向个数；方向选择顺时针或逆时针。



码垛配置完成后，点击【验证配置】来验证当前配置是否合理，如出现配置不成功，请检查码垛中是否有区域无效不可到达的点位。

注：同一码垛中所有示教点位需在同一手系下进行。

1.3.3.4 配置码垛编程案例

```
-----------配置码垛----------
--在运行本程序之前需要加载码垛模块并配置码垛
local posready = p1                    --待机点
local quliao = p2                     --取料点
local flag,num
local pos ={}
InitPlts()                            --加载默认配置文件
--InitPlts("pallet1")                --加载重名后的码垛配置文件
MArchP(posready,0,10,10)              --运动到待机点
```

```
while true do
    MArchP(quliao,0,10,10)          --运动到取料点
    DO(1,ON)
    Delay(150)
    flag,num,pos =GetPLTPos("PLT0")   --获取码垛点
            --pos=Get_Plt("PLT0",num)
    if flag == 0 then          --未码完
        MArchP(pos,0,10,10)       --运动到码垛点
        DO(1,OFF)
        Delay(150)
    elseif flag ==1 then   --码满
        MArchP(pos,0,10,10)       --运动到码垛点
        DO(1,OFF)
        Delay(150)
        ResetPLT("PLT0",1)   --重置码垛
    elseif flag==2 then --码盘的最后一个点为不码垛点，码完的返回值
        ResetPLT("PLT0",1)   --重置码垛
    end
end
```

- 注意：GetPLTPos 指令、ResetPLT 指令以及 InitPlts 指令是组合配套使用的；
- 在使用 GetPltPos 指令前必须先要使用 InitPlts 加载配置的文件；
- 若使用配置码垛，除可以利用 GetPltPos 获取配置码垛点位外，还可以利用 Get_Plt 或者 Get_ArcPlt，由码垛形状决定。当使用 Get_Plt 或 Get_ArcPlt 获取指定码垛点时，遇到不码垛点返回空值；

- 配置码垛是通过在**码垛配置界面**设定码垛工艺用到的参数（例如：码垛名、行数、列数、间隔数等），然后结合 AR 程序实现码垛工艺，故称作配置码垛；
- 配置码垛须添加码垛库（GetPLTPos 与 ResetPLT 这两个指令是在 pallet.lib 库里面封装定义的）。

1.3.4 视觉

新版视觉继承了旧版视觉的大部分功能、UI 和指令用法，少数的视觉配置界面和指令用法有些改动，增加了 25 点自动标定。

1.3.4.1 静态视觉

静态视觉即相机固定于机器人工作范围内，位置固定，拍照视野范围固定，视觉充当机器人的眼睛，采用以太网与 QC410 控制器进行数据交互。静态视觉分正装和倒装两种安装方式。

➢ **正装**

视觉镜头朝下，由上往下拍照；一般应用场合是定位，即相机先拍照，然后机器人抓着物体移动到定位点或者机器人移动到拍照定位点去取物体。

➢ **倒装**

视觉镜头朝上，由下往上拍照；一般应用场合为静态纠偏，即机器人抓着物体移动到倒装相机上方拍照，然后计算出一个虚拟工具坐标，之后以虚拟工具的中心（即物体的中心）到达指定的定位点。

1.3.4.2 动态视觉

动态视觉即相机安装在机器人的手臂上，随机器人的运动而移动。视觉充当机器人的眼睛，采用以太网通讯与 QC410 控制器进行数据交互。相对于静态相机而言，动态相机安装比较方便灵活，拍照范围更广。

动态相机可安装在机器人的小臂（J2 轴）或旋转轴（J4 轴），两者区别在于：

- 安装在 J4 轴相对于 J2 轴来说，拍照范围更广、更灵活。
- 相机安装在 J2 轴的精度较高。

1.3.5 传送带

静态视觉和动态视觉的区别在于相机的实际位置是否是固定的。静态视觉的相机是固定不动的，所以在进行标定时需要选取不同位置的像素坐标，可以是一次性获取 9 个点、16 个点、25 个点不同位置的的坐标（手动标定），也可以是一个点移动到不同位置的的坐标（自动标定）。动态视觉相机是随着机器人的运动而移动，因此标定纸需固定于实际空间中。

以上两种视觉，都是相机对物料拍完照后，将数据发送给机器人，机器人去抓取。在拍完照以后到机器人运动到抓取点位的过程中，物料的位置是不发生任何变化的，即对于抓取过程来说是静态的。

传送带应用、流水线应用也叫动态抓取过程。物料在传送带上随着传送带的移动而移动。此时相机拍照获取物料当前位置，从发送数据到机器人追到拍照点的时间内物料仍在移动，如果按照原来发送的数据去抓料，位置是不准确的。因此通过计算传送带编码器的脉冲值来计算这一段距离。在计算这一段距离时，可以分为跟随和快抓两种方式。

跟随和快抓区别：传送带跟踪，是指机器人利用用户输入的物料点位置，以及物料在这一位置时对应的编码器值，实时计算物料点的位置，通过运动来追到物料。当机器人追到物料，并与物料相遇的时候，有两种策略选择。第一种策略是，机器人将相遇点作为运动的目标位置，此时返回追到目标的状态（sta=2），用户可以在这之后做取料动作，整个跟踪过程结束，这就是我们所说的快速抓取功能。第二种策略，机器人到达相遇点的同时，工具末端的合成速度与物料运动的速度在数值上相等、方向相同，此时也返回状态（sta=2），用户需要主动告诉机器人，需要结束同步过程(调用 SynOver())，机器人才会退出同步，执行其他动作，否则会一直同速度同方向地跟踪物料移动，这是我们所说的动态跟随功能。

因此，静态视觉和动态视觉的精度要高于传送带动态抓取过程。跟随的精度要高于快抓的精度，同样的，快抓的速度会高于跟随的速度。

注：具体案例请参考附件。

1.3.6 网络

当 QC410 控制器与外部设备只是进行简单的网络通讯，不涉及到视觉标定、坐标

处理等操作时，可通过添加**网络**模块的方式实现。大致步骤如下：

注意：新建网络名称默认为 NET0、NET1….NET10，用户也可自定义网络名称。

网络配置界面如下，包含网络基本参数和接收方式两个参数模块。

- （本机）网络模式（QC410 控制器）：

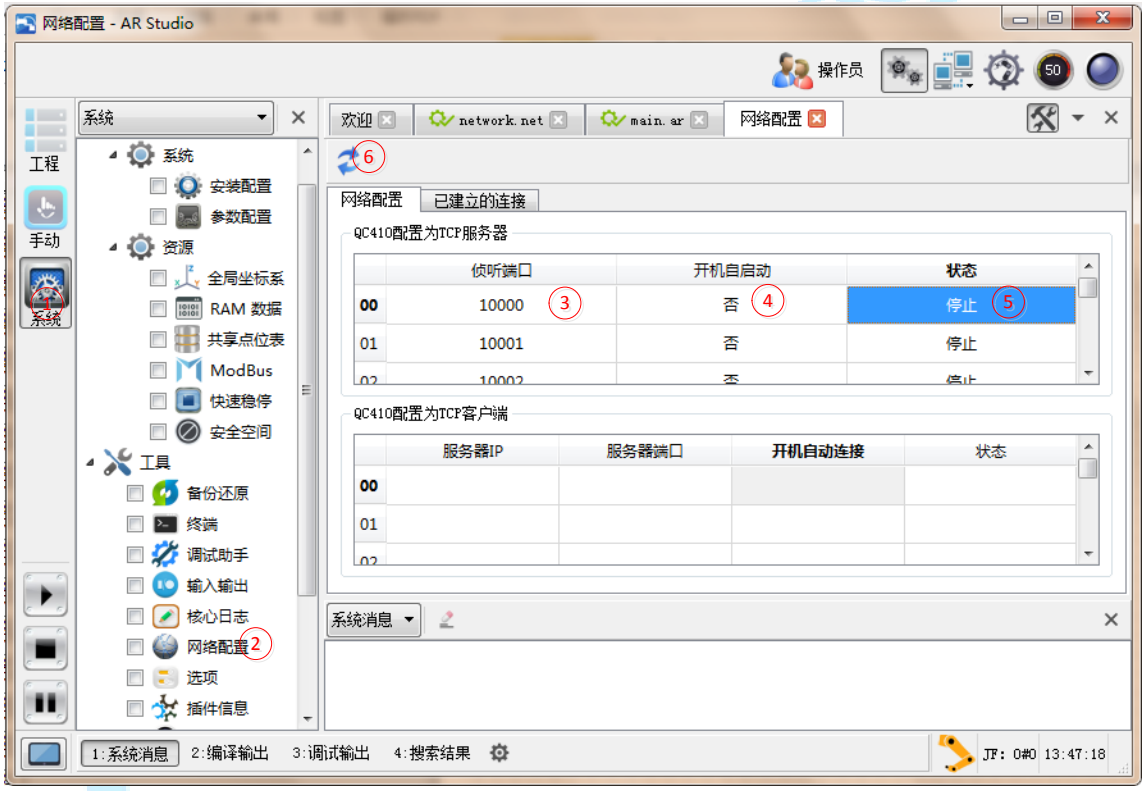
a) **UDP_Server**：QC410 控制器作为服务器，视觉作为客户端 (注:视觉的 IP 须与控制器 IP：192.168.0.123（默认）在同一网段；控制器的端口为 110)。

b) **UDP_Client**：QC410 控制器作为客户端，视觉作为服务器，此处须配置与机器人相连的连接对象的 IP 和端口号(注：连接对象的 IP 须与 QC410 控制器 IP：192.168.0.123（默认）在同一网段；控制器的端口为 110)。

c) **TCP_Client**：QC410 控制器作为客户端，视觉作为服务器；此处须配置与机

器人相连的连接对象的 IP 和端口号(注：连接对象的 IP 须与 QC410 控制器 IP：192.168.0.123（默认）在同一网段)。

d) **TCP_Server**：QC410 控制器作为服务器，视觉作为客户端；此处须配置与机器人相连的**连接对象**的 IP 和端口号(注：连接对象的 IP 须与 QC410 控制器 IP：192.168.0.123 在同一网段)。此处仍需要额外配置侦听端口序列号，首先需要在系统菜单下网络配置中的 QC410 配置为服务器选择一未使用的设置侦听端口并设置为开机自启动（可选，可手动将端口状态设置为启动或停止），设置完成以后点击同步显示当前端口状态。设置完成后，回到视觉配置界面选择刚才设置的监听端口号。步骤如下图标号所示：



● 接收方式

网络数据接收方式分堵塞式和非堵塞式两种。一般情况下，若是多任务则采用非阻塞；若是单一任务，则采用阻塞模式（采用非阻塞模式一直扫描比较占用资源，降低了 CPU 的使用效率）。

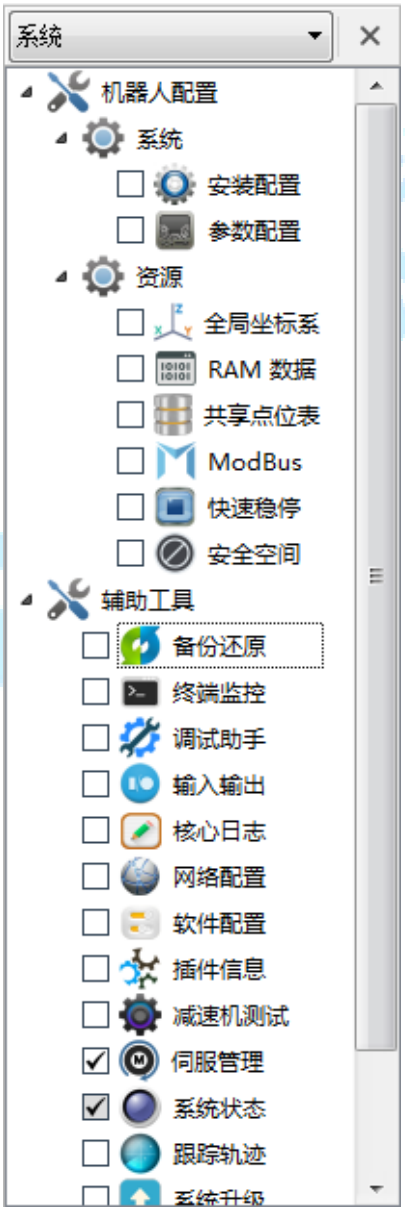
堵塞方式：发送触发信号后，程序会在阻塞时间内一直停留在接收数据的那一行函数，AR 程序会继续执行直到接收视觉发送过来的数据；若阻塞时间设置为 0，

程序会死等停留在数据接收函数那一行直到接收到视觉发送的数据。

非堵塞方式：发送触发信号后，不管有没有接收到视觉发送来的数据，AR 程序也会继续执行。

1.4 机器人配置：系统

点击图标“ ”，进入到机器人配置、工具界面。如下图：

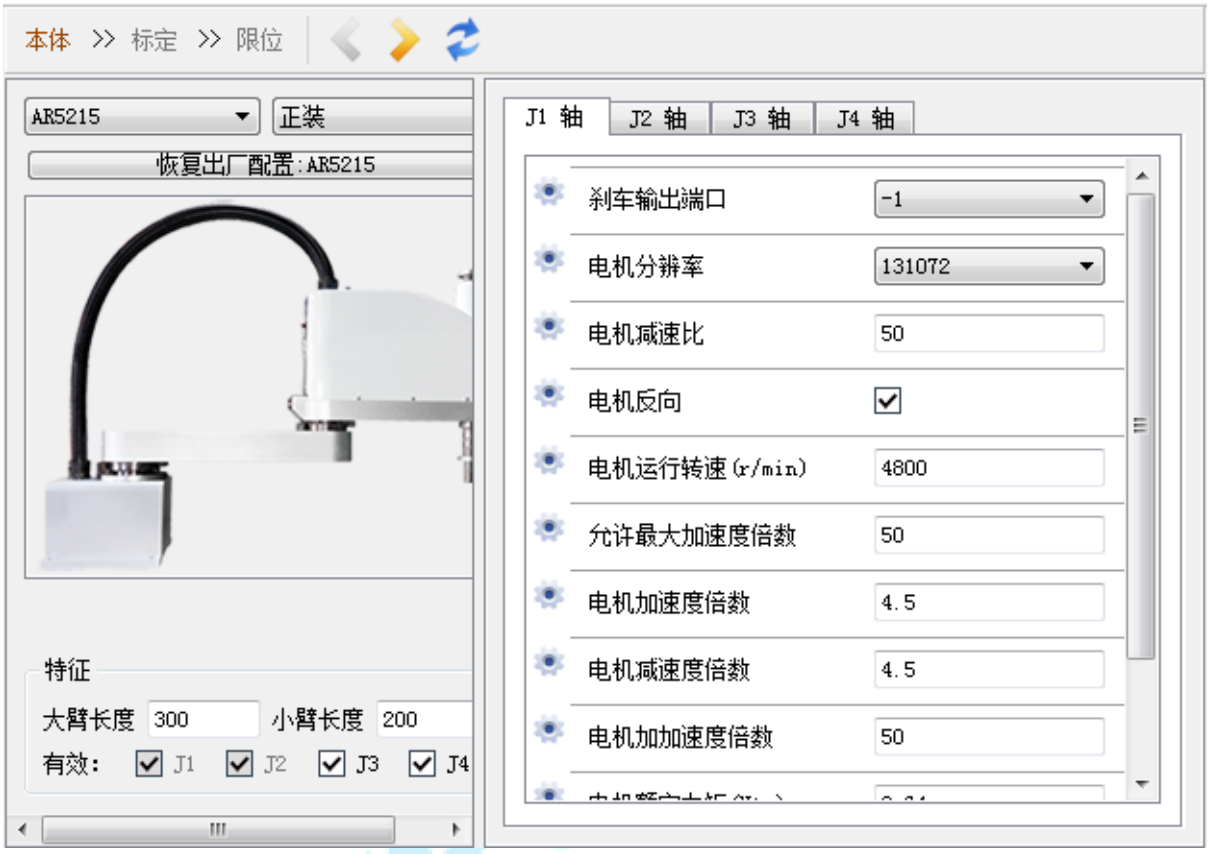


1.4.1 安装配置

安装配置主要为机器人的本体参数设置，它包含：本体、标定、限位三部分内容，一般仅在新机调试时，才会需要修改里面的内容。须获取管理员及以上的权限才能修改

里面的参数。

1.4.1.1 本体



大臂长度：机器人本体大臂实际臂长，单位：mm；

小臂长度：机器人本体小臂实际臂长，单位：mm；

有效：当前轴是否有使用；

J3&J4 复合轴:第三轴和第四轴是否是复合轴;

J3 螺距：机器人第三轴螺距长度，单位：mm;

刹车输出端口：带刹车电机，根据刹车接线，配置相应端口，其中“**-1**”是未配置的意思；

电机分辨率：当前轴电机编码器分辨率；

电机减速比：当前轴机械传动减速比；

电机反向：当前轴电机运动方向是否取反；

电机运行转速(r/min)：当前轴电机每分钟最大转动圈数也就是点到点运动在 100%倍率时，电机的转速，其单位：r/min；设定范围：1000~10000

允许最大加速度倍数：当前轴的电机实际最大加速度与电机最大转速的比值。用于设定

当前轴的电机的最大加速度限定值，如果实际运动中，当前轴电机的加速度超过限定值，会触发报警；设定范围为 10~100；

电机加速度倍数：当前轴的电机实际加速度是电机转速的倍数；设定范围为 1~20；

电机减速度倍数：当前轴的电机实际减速度是电机转速的倍数；设定范围为 1~20；

电机加加速度倍数：加加速度是加速度的导数。加加速度倍数：当前轴的电机实际加加速度是加速度的倍数；设定范围为 10~200；

电机额定力矩：当前轴电机额定力矩，单位是（N*m）；根据电机铭牌上给定的值设定

电机额定电流：当前轴电机额定电流，单位是（A）；根据电机铭牌上给定的值设定：

- **小技巧：**编码器分辨率、减速机减速比以及上下轴螺距需根据实际情况填写；
- **小技巧：**刹车接 Relay1~Relay4 分别对应输出端口 OUT23~OUT26；
- **小技巧：**更改上述参数需点击“同步参数”按钮来完成更新；
- 小技巧：通过左右按键 操作来切换三个界面的安装；
- 小技巧：修改其中的某一个或多个参数，须点击 按钮同步参数生效。

1.4.1.2 标定

标定界面主要包含：清电池报警以及动力学参数辨识。

【清电池报警】

清除与编码器电池相关的错误报警。例如 11042 电池错误报警以及 11090 点击编码器电池低压报警。

【动力学参数辨识】

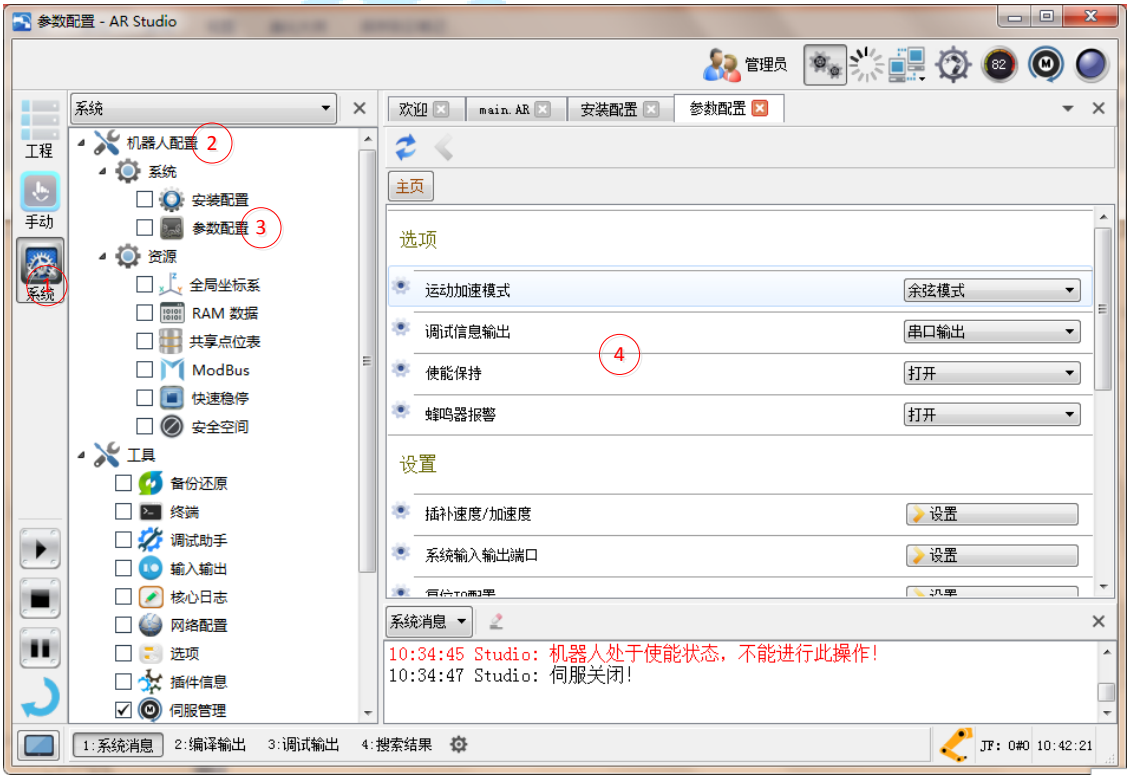
由于机器人动力学参数辨识的精度与目标轨迹直接相关，所以标定时必须首先确定标定轨迹。“激励轨迹”指的是最大可能将机器人动力学特性激发出来的，并确保能对机器人动力学参数进行可靠、精确、快速和充分辨识的轨迹。激励轨迹辨识是指通过“激励轨迹”来获得动力学参数。详情请参见附件“QC410 动力学参数辨识”和“最佳加速度设置”。

1.4.1.3 限位

在此界面设定 J1、J2、J3、J4 轴的正负软限位即对应关节正反转方向的极限角度，单位为度。点击 同步参数完成更新设定。

1.4.2 参数配置

机器人用户参数设置，主要包含通讯方式、速度、加速度设置。选中【机器人配置】折叠菜单中的【参数配置】进入到参数设置【主页】界面：



1.4.2.1 选项

（1） 运动加速模式：直线模式、余弦模式、指数模式；

（2） 调试信息输出：串口输出、文件输出。该参数用于调试人员调试底层程序时输出调试信息的一种方式；

（3） 使能保持：主要是针对 AR 程序运行停止后，使能是否需要保持。

（4） 蜂鸣器报警：当系统有报警的时候，控制器中的蜂鸣器是否有声音提示。默认打开，即有声音提示。

1.4.2.2 设置

➢ **插补速度/加速度**

硬跟随：0 为关闭，1 为打开此功能；主要应用于机器人走笛卡尔空间轨迹过程中需

要保持匀速的场合，例如点胶工艺等。

当机器人走笛卡尔空间轨迹例如直线或圆弧轨迹的时候，如果开启了硬跟随功能，在姿态突然发生较大变化的路径，也可以保持匀速过渡，不会产生明显的速度波动，但是会产生瞬间的加速度冲击，可根据实际需要选择开启与否。

直线速度/加速度

手动直线速度(mm/s)	50
手动直线加速度(mm/s^2)	500
手动直线加加速度(mm/s^3)	6000
自动直线速度(mm/s)	2000
自动直线加速度(mm/s^2)	5000
自动直线加加速度(mm/s^3)	50000
最大直线速度(mm/s)	2000
最大直线加速度(mm/s^2)	5000
最大直线加加速度(mm/s^3)	50000

- 手动直线速度：100%倍率下，在笛卡尔坐标系下手动运动时，机器人末端位置运动处于匀速时的速度；
- 手动直线加速度：100%倍率下，在笛卡尔坐标系下手动运动时，机器人末端位置运动处于加（减）速时的加（减）速度；
- 手动直线加加速度：100%倍率下，在笛卡尔坐标系下手动运动时，机器人末端运动加（减）速度的导数；
- 自动直线速度：100%倍率下，机器人处于程序自动运行状态下，机器人末端位置处于匀速时的速度，主要是指 MovL 指令；
- 自动直线加速度：100%倍率下，机器人处于程序自动运行状态时，机器人末端位置运动处于加（减）速时的加（减）速度；
- 自动直线加加速度：100%倍率下，机器人处于程序自动运行状态时，机器人末端位置运动处于加（减）速时的导数；
- 最大直线速度：针对上面直线速度设置的一个限定值；
- 最大直线加速度：针对上面直线加速度设置的一个限定值；
- 最大直线加加速度：针对上面直线加加速度设置的一个限定值。

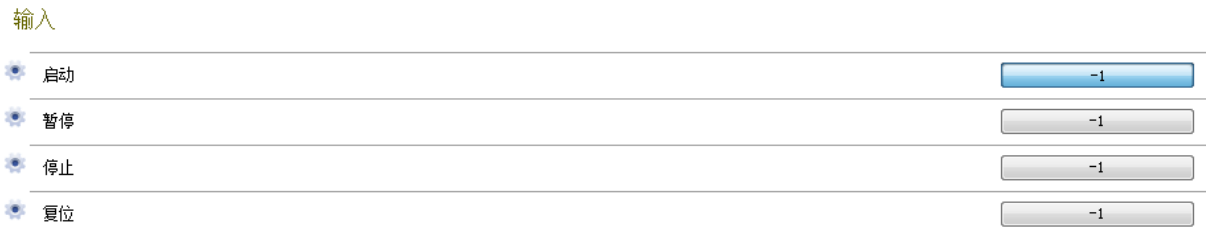
姿态速度/加速度

手动姿态速度(度/s)	100
手动姿态加速度(度/s^2)	1000
手动姿态加加速度(度/s^3)	200000
自动姿态速度(度/s)	360
自动姿态加速度(度/s^2)	3000
自动姿态加加速度(度/s^3)	15000
最大姿态速度(度/s)	500
最大姿态加速度(度/s^2)	3000
最大姿态加加速度(度/s^3)	30000

- 手动姿态速度：100%倍率下，手动在笛卡尔坐标系下运动时，C 轴运动处于匀速时的速度；
- 手动姿态加速度：100%倍率下，手动在笛卡尔坐标系下运动时，C 轴运动处于加（减）速时的加（减）速度；
- 手动姿态加加速度：100%倍率下，手动在笛卡尔坐标系下运动时，C 轴运动处于加（减）速的导数；
- 自动姿态速度：100%倍率下，机器人处于程序自动运行状态时，C 轴运动处于匀速时的速度；
- 自动姿态加速度：100%倍率下，机器人处于程序自动运行状态时，C 轴运动处于加（减）速时的加（减）速度；
- 自动姿态加加速度：100%倍率下，机器人处于程序自动运行状态时，C 轴运动处于加（减）速的导数；
- 最大姿态速度：针对上面设置的姿态速度的限定值；
- 最大姿态加速度：针对上面设置的姿态加速度的限定值；
- 最大姿态加加速度：针对上面设置的姿态加加速度的限定值。

➢ **系统输入输出端口**

- 系统输入信号



启动：外部输入信号启动程序的运行；

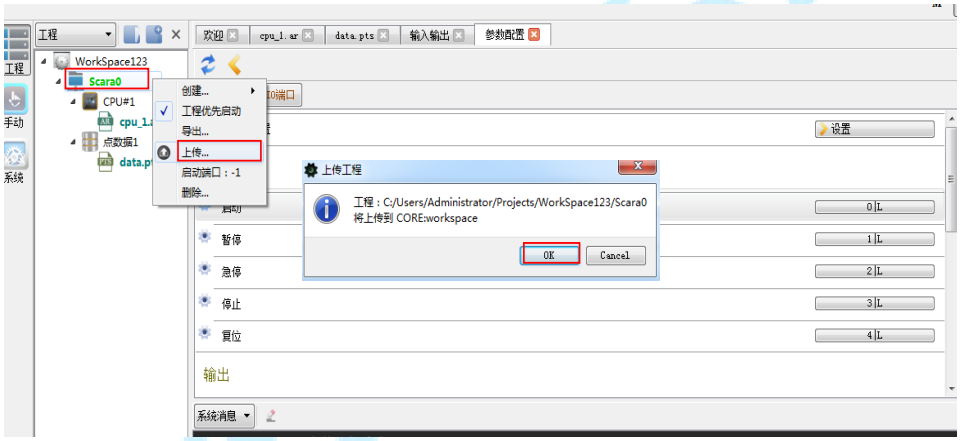
暂停：外部暂停信号暂停程序的运行；

停止：外部停止信号停止程序的运行；

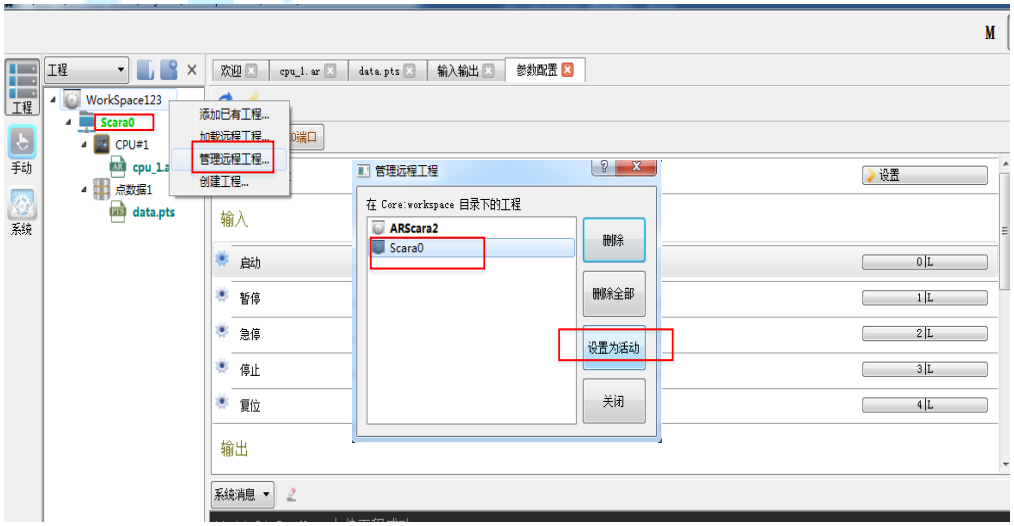
复位：外部复位信号清除系统报警；

以上 4 个外部输入信号的使用步骤大致如下：

1. 首先将当前工程上传至 QC410 控制器默认工作空间中；

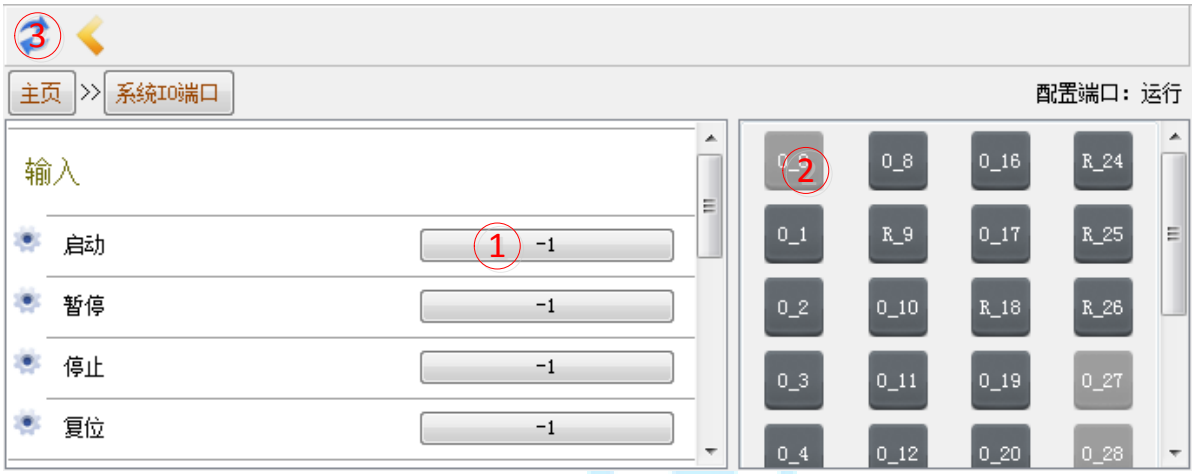


2. 其次设置当前工程为活动状态；



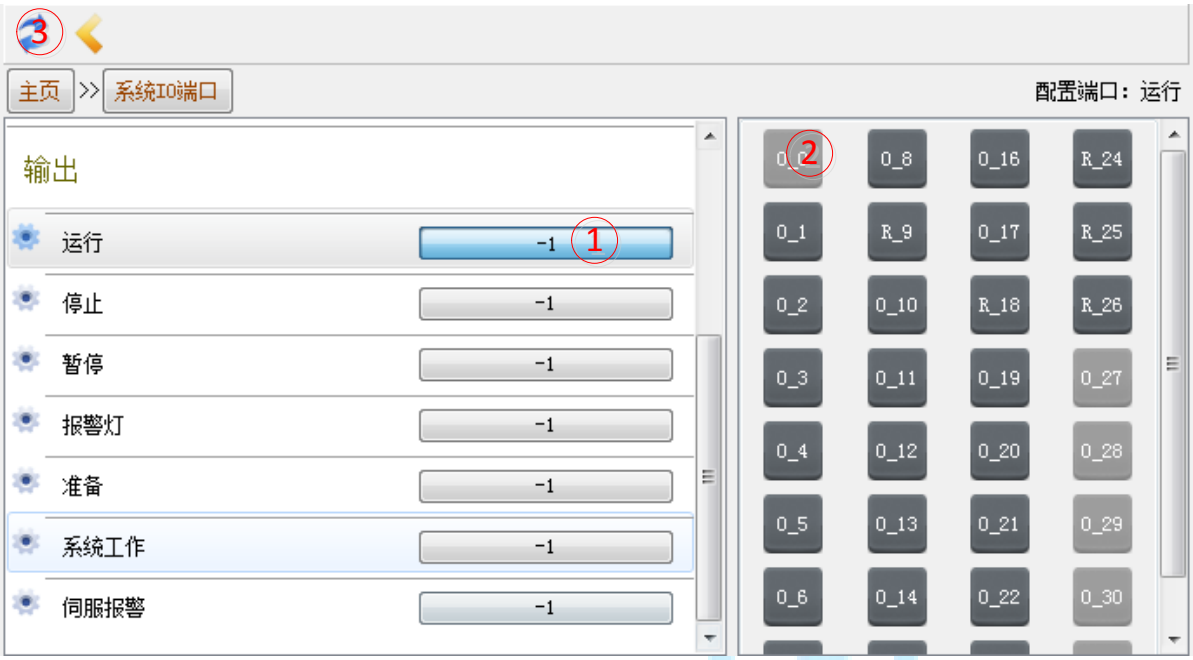
3. 选中需要设置的端口号。QC410 控制器的输入端口即支持高电平输入也支持低电平输入，因此对于这里的系统输入信号来说，有三个选项，这里将以**启动**功

能设置为外部输入 0 号端口为例说明。如果是输入低电平有效：“0|L”，L 是 Low 低电平有效的意思；“0|H”，H 是 High 高电平有效的意思；如果不配置设为“-1”即可。首先选中需要设置的功能如同“①”，然后选择需要设置的端口号“②”，鼠标左键点击端口号进行切换三种选项选择对应的高低电平有效。设置完成后点击“③”进行同步参数。



注：在模拟状态、虚拟 IO 的情况下，默认为输入低电平有效。因此，如果配置为“0|H”，那么该端口常闭的状态下也就是灯灭的情况下是有效的，常开的状态下即灯亮是无效的，如果配置为“0|L”是指端口常开是有效的，常闭是无效的。如果外部接线为输入高电平有效，那么配置为“0|L”是指端口常开的情况是无效的，常闭是有效的，配置为“0|H”是指端口常开是有效的，常闭是无效的。

- 系统输出信号



运行：AR 处于运行状态则对应的输出端口打开

- 停止：AR 程序运行停止则对应的输出端口打开
- 暂停：AR 程序运行暂停则对应的输出端口打开
- 报警灯：报警则对应的输出端口打开
- 准备：开机的时候只要系统不报警，则配置的输出端口打开；如有报警，则配置的输出端口不响应（关闭）；
- 系统工作：只要系统启动，GUI 已创建成功，则系统工作输出端口打开
- 伺服报警：指的是机器人使能的状态，若使能，则输出打开；若非使能，则输出关闭；

注：众为兴的输出是只支持低电平输出的，因此，如果运行配置为“**1|L**”，那么当程序运行时 1 号端口会打开；不运行时 1 号端口会关闭。如果配置为“1|H”，那么当程序运行时 1 号端口呈关闭状态，不运行时 1 号端口打开。

➢ 复位 IO 设置

“保持”是给机器人复位信号时（或按复位按钮）原有的输出信号保持其原有的状态；

“关闭”是给机器人复位信号时（或按复位按钮）后不管原有的输出信号是打开还是关闭这里都会进行关闭。

“打开”则是给机器人复位信号时（或按复位按钮）后不管原有的输出信号是打开还是关闭这里都会进行打开。

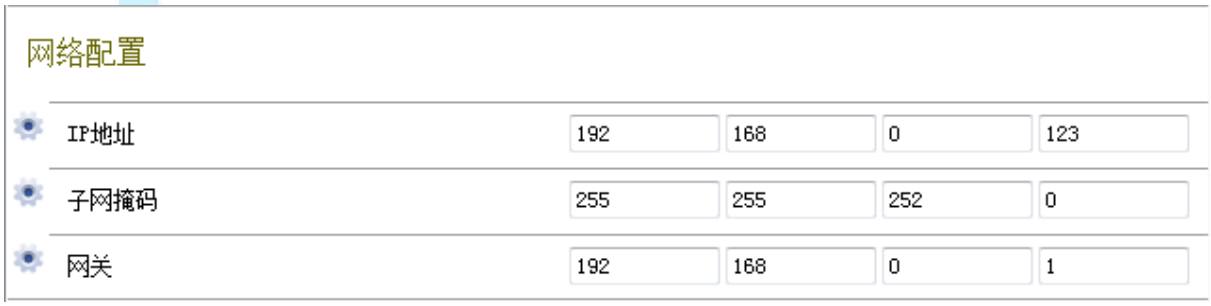
注：选中需要修改的信号，鼠标单击左键切换状态，最后点击同步参数。

➢ **串口/网络配置**

网络配置是指配置控制器系统网络 IP 地址、子网掩码地址、网关以及 MAC 地址。串口配置是指 RS232、RS485 串口参数配置。

注：这里的网络配置与工具菜单下的网络配置是同步，只需设置一个点击同步参数另一个自动更新，设置完成后需要重启一体机才能更新网络参数。

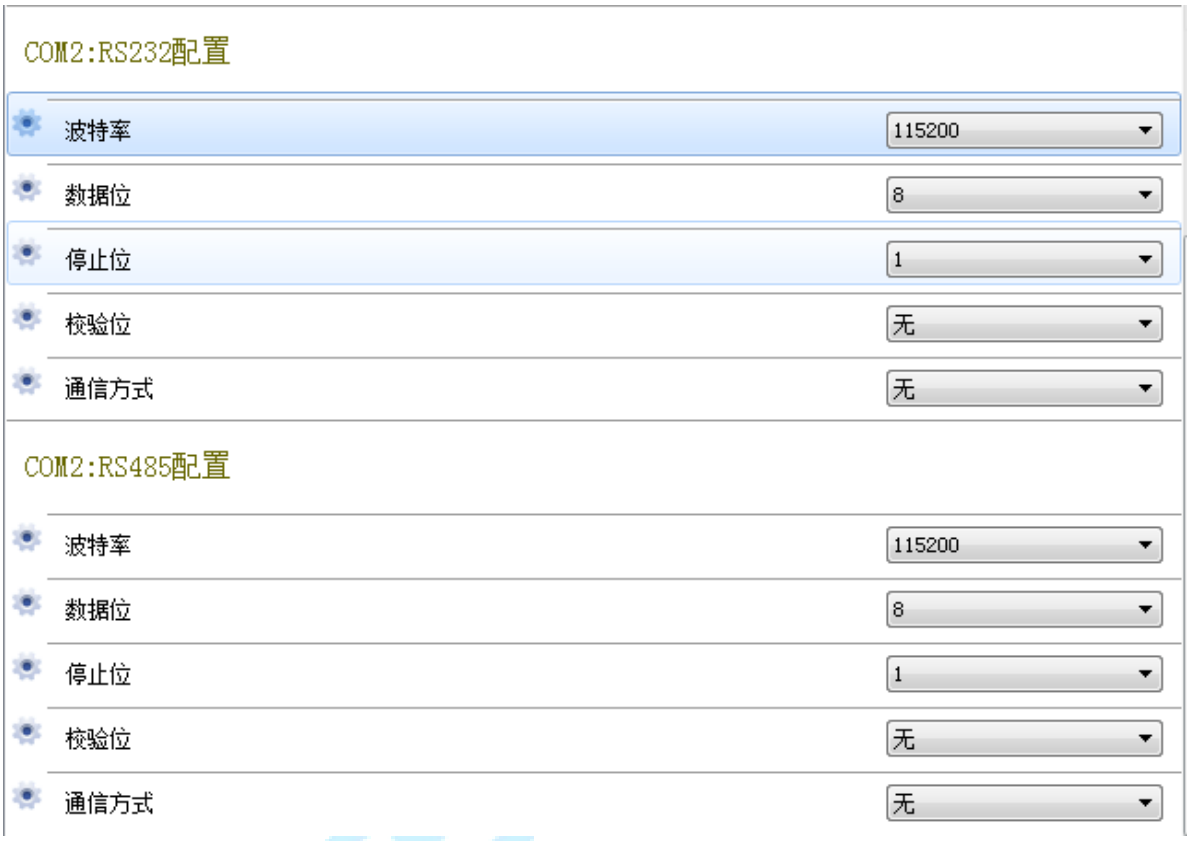
网络配置：



注意：如果子网掩码为 255.255.255.0；那么在同一网段内的 IP 可设置范围为：192.168.0.1-192.168.0.255；如果子网掩码为 255.255.252.0，那么在同一网段内的 IP 可设

置为 192.168.0.1-192.168.0.255、192.168.1.1-192.168.1.255、192.168.2.1-192.168.2.255、192.168.3.1-192.168.3.255。

串口设置：



➢ **最佳加速度设置**

加速度自适应是指根据机器人实际工艺中的负载情况重新计算出动力学模型参数，将这些参数代入动力学模型中在额定电流的 1.5 倍以下计算出最佳的加速度。在出厂前，我司已进行了激励轨迹辨识获取了机器人在空载状态下的动力学模型参数。因此客户或者 FAE 工程人员来说，只需将负载参数进行配置。在运动的过程，会根据所输入的负载参数重新计算实际工艺参数获取最佳加速度。具体操作参考《最佳加速度设置》操作指引。

➢ **样条曲线参数**

样条曲线应用场合为不规则边缘点胶，主要指令为 MSpline(p1,p10)。如果 p1 与 p10 点相同的话，就是一个连续的闭合路径。如果不同，就是指一段曲线。

姿态法向跟随开关：一般默认为关闭，应用场合为点胶工艺，如果针头是歪的，就需要沿着法向进行跟随。

拆分步长：即每两个点之间步长为 1mm 进行拆分。步长越小，插补的点越多。

➢ **远程 IO**

QC410 控制器自带有 34 路输入和 27 路输出，当现场应用这些输入输出数量欠缺的情况下，则可通过远程 IO（IO 模块）的方式扩展 IO 的数量达到应用的要求。一个远程 IO 模块包含有 16 路输入和 16 路输出。QC410 最多可以扩展 2 个 IO 模块（即最多扩展 32 路输入 100 ~ 131 和 32 路输出 100 ~ 131）。远程 IO 模块与 QC410 之间通过 Modbus TCP 通讯，其中 **QC410** 控制器作为**主站**，远程 **IO 模块**为**从站**。

IO 模块默认的 IP 地址为 192.168.0.214，端口为 502.若 IO 模块 IP 地址或端口号修

改，QC410 里的【参数配置】-----【远程 IO】界面中的以下参数须同步修改。

远程 IO 使用步骤如下：

（1） 一根网线连接 QC410 控制器和 IO 模块；

（2） 正确设置 IO 模块 IP 地址和控制器 IP 地址；

（3） 设置 QC410 里的【参数配置】-----【远程 IO】界面的**模块 1** 和**模块 2** 中的 IP 地址和端口；

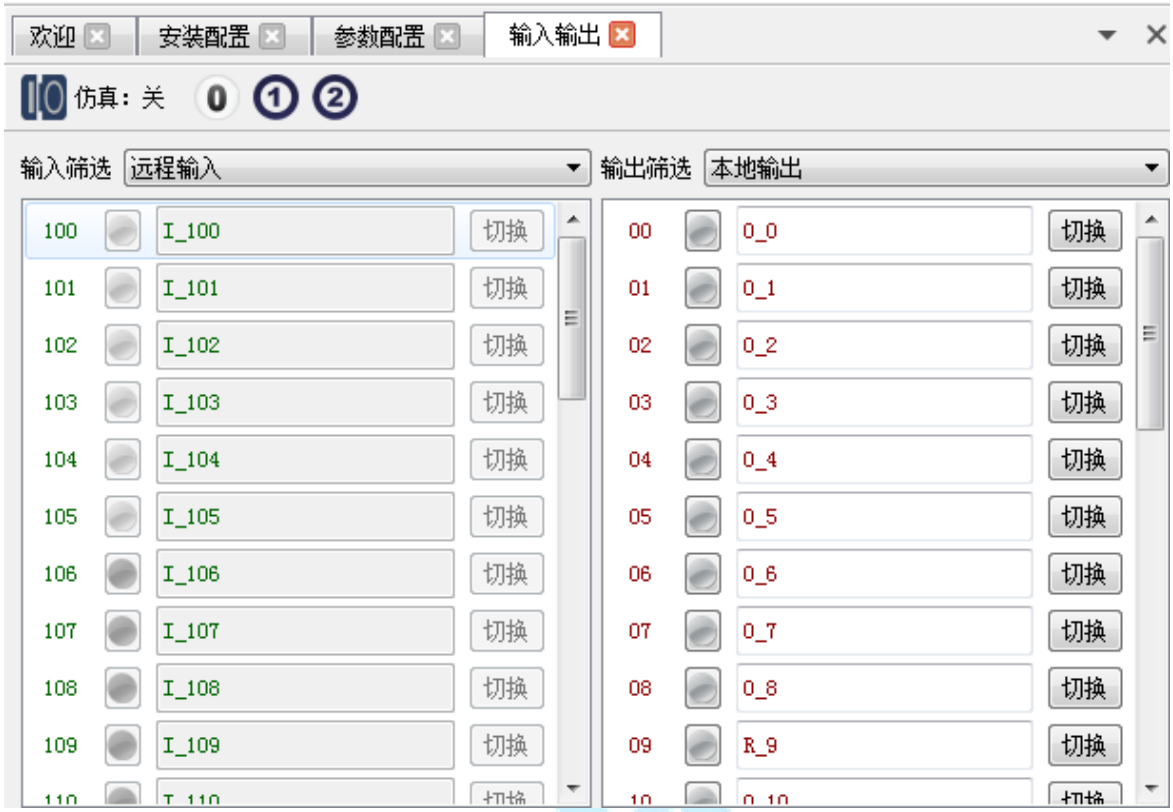
（4） 进入【输入输出】界面，点击 ① 或 ② 建立 QC410 与远程模块 1 或模块 2 的连接，连接失败或成功，都会有相应的弹窗提示；

（5） 连接成功后可用 DI 和 DO 指令操作对应的输入和输出端口，例如：

DO(100,ON) --打开远程输出端口 100，对应的是 IO 远程模块 1 输出 output 0

if DI(100) == ON then --判断输入端口 100（对应的 IO 模块 1 的输入 input 0）

--的状态是否为 ON



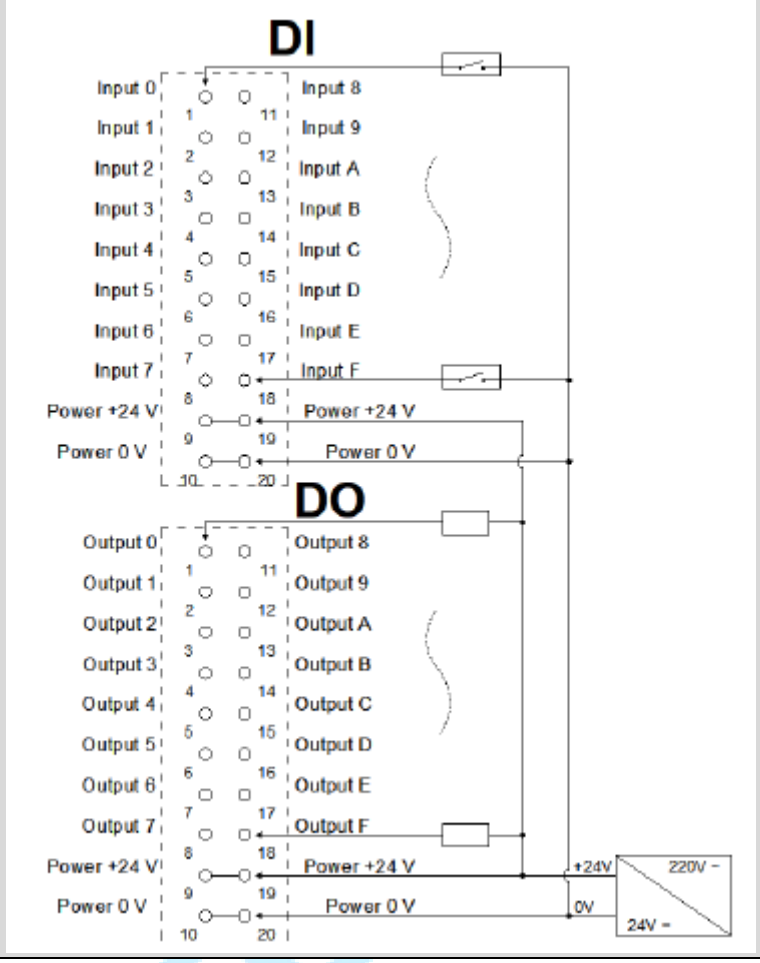
✧ **小技巧：远程IO不支持组合IO（2个或2个以上的IO）的使用，例如DO({100,101},{ON,ON})非法；**

✧ **EM1144A 外部 24V 电源接线示意图如下：**

注意 1：EM1144A 模块须接外部 24V 才能使用；

注意 2：输入低电平输入有效 NPN；

注意 3：输出为低电平（NPN 输出）；



✧ EM1144A 模块的 IP 地址修改操作指引

（该操作需要慎重检查，此操作不可逆）

使用 ModBus 主站软件可以修改模块的 IP，对应 IP 寄存器的 Modbus 地址如下图表格：

功能码	0x10（有些软件显示16）
起始地址	1000
长度	13

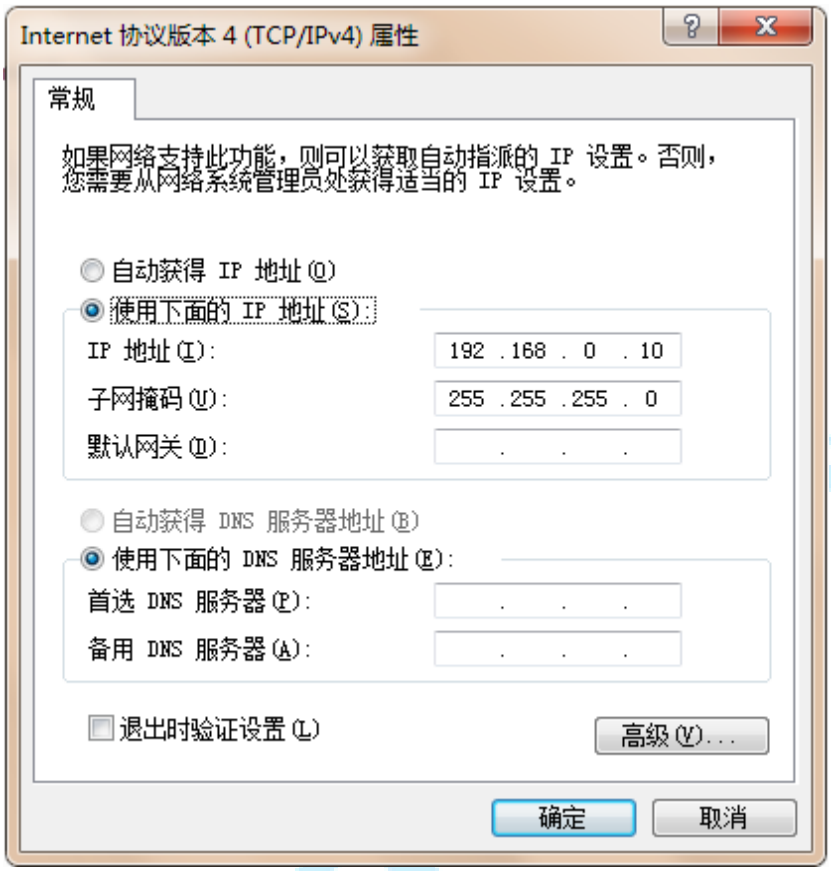
具体操作步骤如下：

（1） 用一跟网线连接 PC 与 EM1144A 模块；

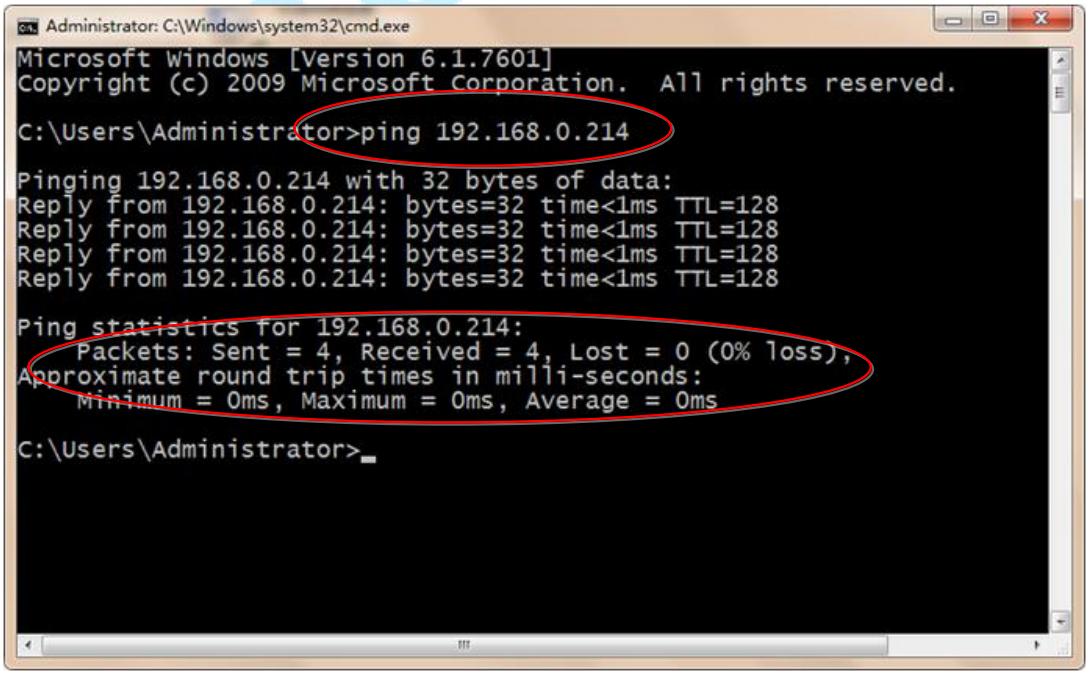
（2） PC 机安装 Modbus Poll

软件；

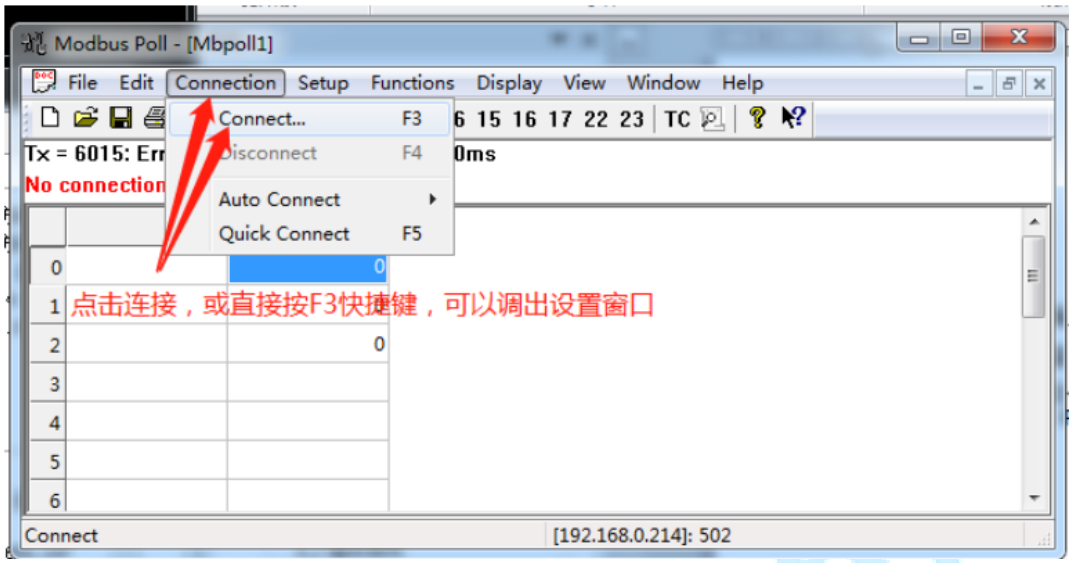
（3） 将 PC 的 IP 地址设置成跟 EM1144A 模块在同一个网段；



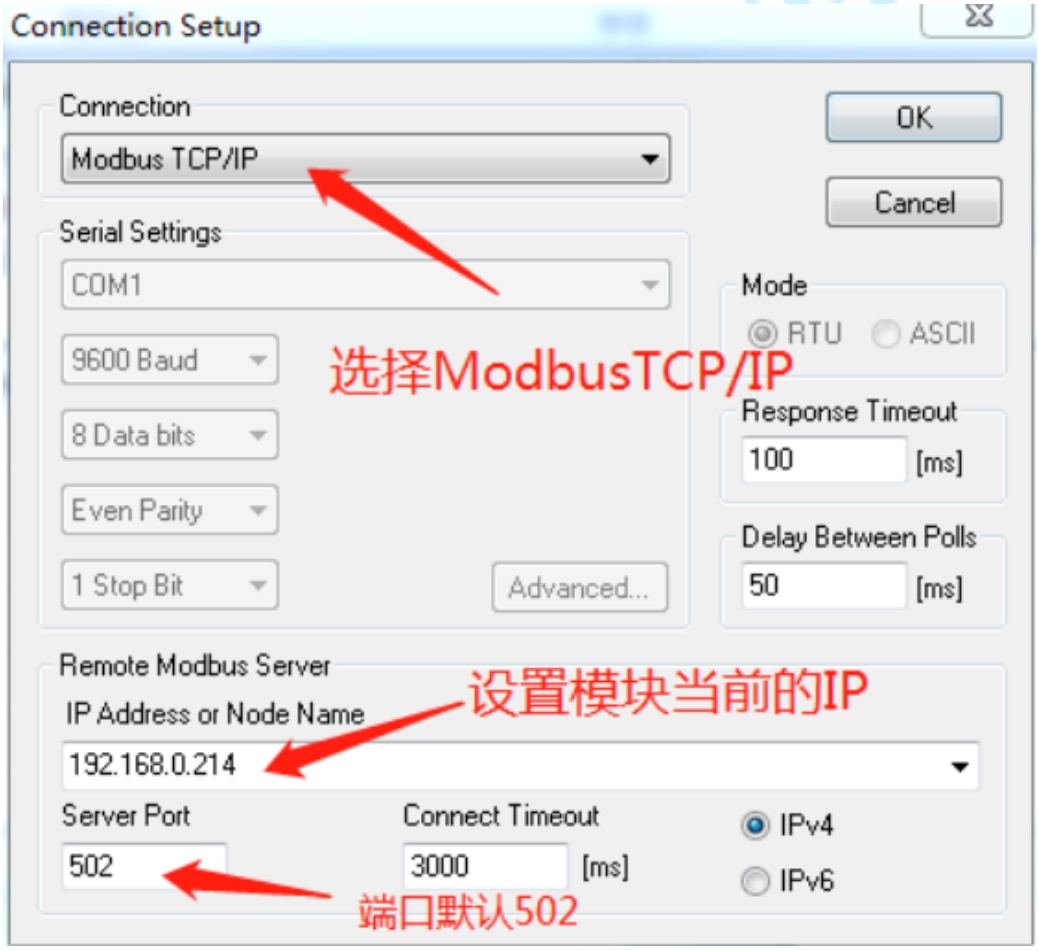
（4） 运行 Windows 的 CMD 命令，测试网络的物理连接是否成功。在命令提示符下，输入： ping 192.168.0.214 观察网络连接情况，无丢包为正常。若出现异常，请检查 IP 地址设置以及网络连接。



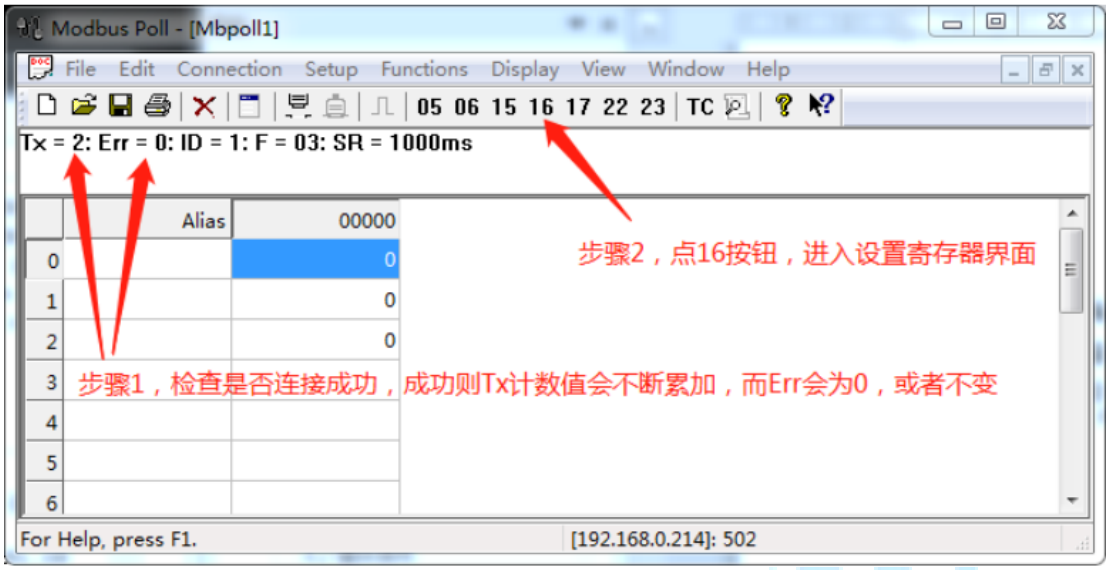
（5） 打开 PC 端的 Modbus Poll 软件，进入设置菜单界面：



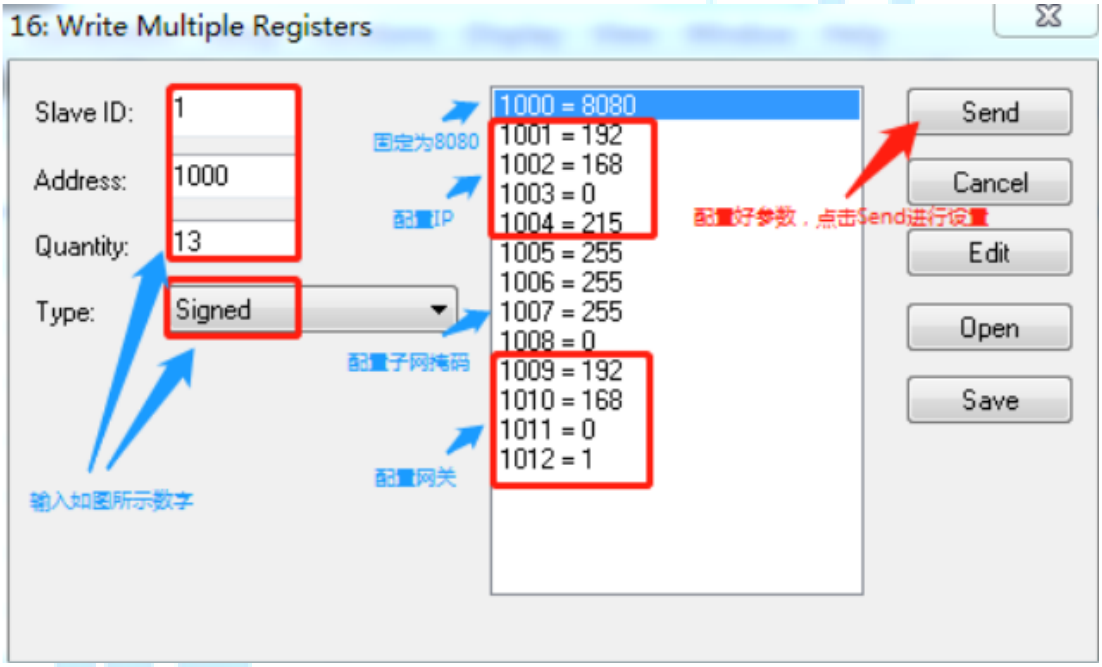
（6） 设置连接信息，点击“OK”会自动连接设备：



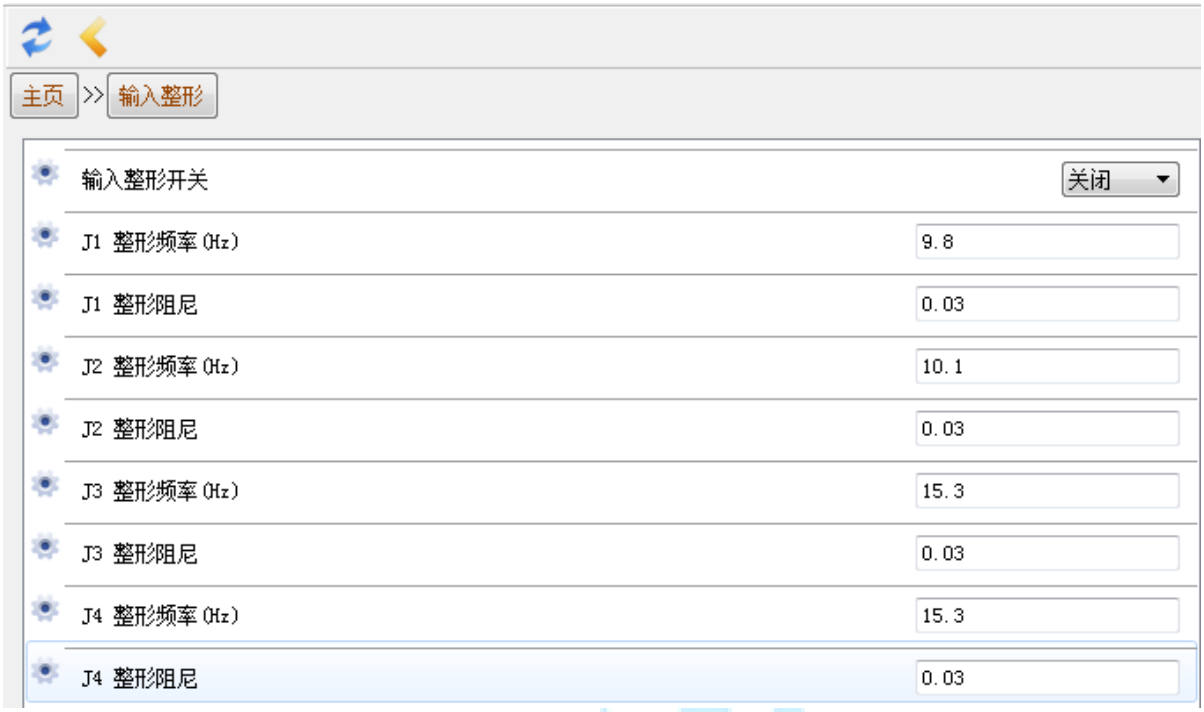
（7） 设置连接信息，点击 Ok 会自动连设备，连接成功会显示如下，并做下一步操作：



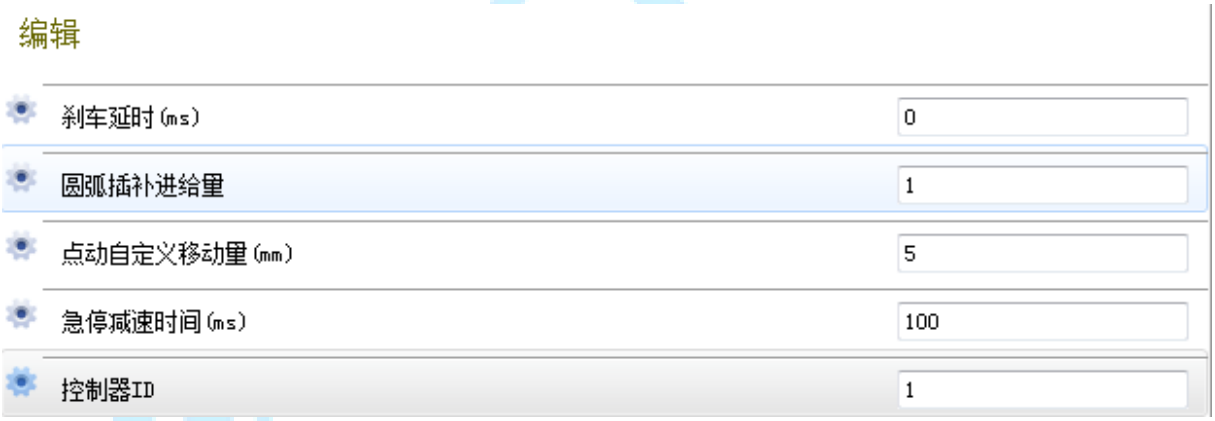
（8） 在下列界面设置配置信息：



➢ 输入整型参数



1.4.2.3 编辑



- **刹车延时:**避免 Z 轴负载情况下下掉的现象，默认为 100ms，设定范围 0~500ms
- **圆弧插补进给量：**是指进行圆弧插补时的单段步长。圆弧拆分精度,默认为 1；
- **点动自定义移动量：**定义单步移动的最大值（默认为 5）；
- **急停减速时间：**按下急停按钮后，减速的时间，默认为 100ms，设定范围：50~100ms；
- **控制器 ID：**在进行 Modbus 通讯时，需配置控制器 ID 号。

1.5 机器人配置：资源

1.5.1 全局坐标系

当不以基座为参考零点，进行位置示教与计算时，用户坐标系可以方便量测工作区间中各点的位置并加以任务安排，更符合人的直观感受。系统总共 20 个用户坐标，用户 0，为系统默认不可更改。用户 1-19 可手动设置，也通过三点示教法（直角）或六点法示教（圆弧）生成。

当机械末端增加夹具，运动轨迹将不以法兰中心为参考，而是夹具末端，工具坐标将使示教、编程更加灵活。系统总共 20 个工具坐标，工具 0，为系统默认不可更改。工具 1- 19 可手动设置，也通过二点法示教法直接生成。

在手动界面，点击“ ”图表，切换到用户/工具标定界面，如下图所示，可以进行用户/工具坐标系的标定。

全局用户坐标系 | 全局工具坐标系

	名称	X	Y	Z	C
#0	qas	500	0	0	0
#1	e	234	120	0	0
#2		0	0	0	0
#3					
#4					
#5					
#6					
#7					
#8					
#9					

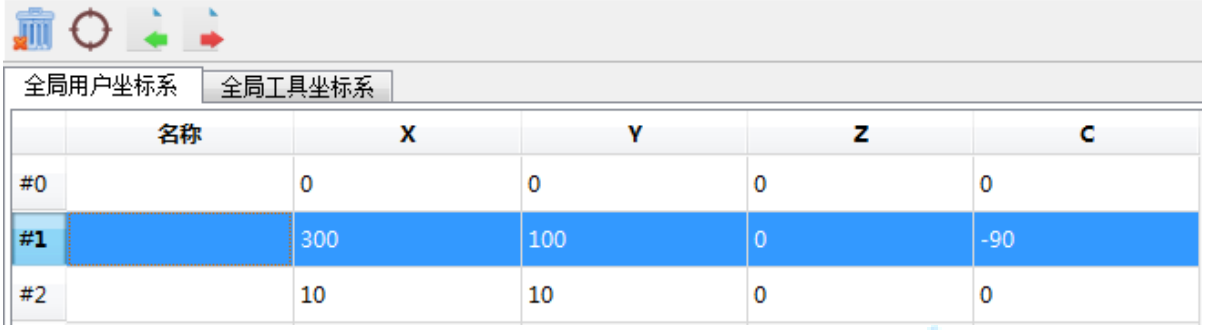
1.5.1.1 用户坐标系标定

➢ **标准用户坐标系标定**

标定步骤如下：

（1） 从 1~19 个用户中选中一个用户号（选中之后用户号所在的行由白色或浅灰

色标记为蓝色）；



	名称	X	Y	Z	C
#0		0	0	0	0
#1		300	100	0	-90
#2		10	10	0	0

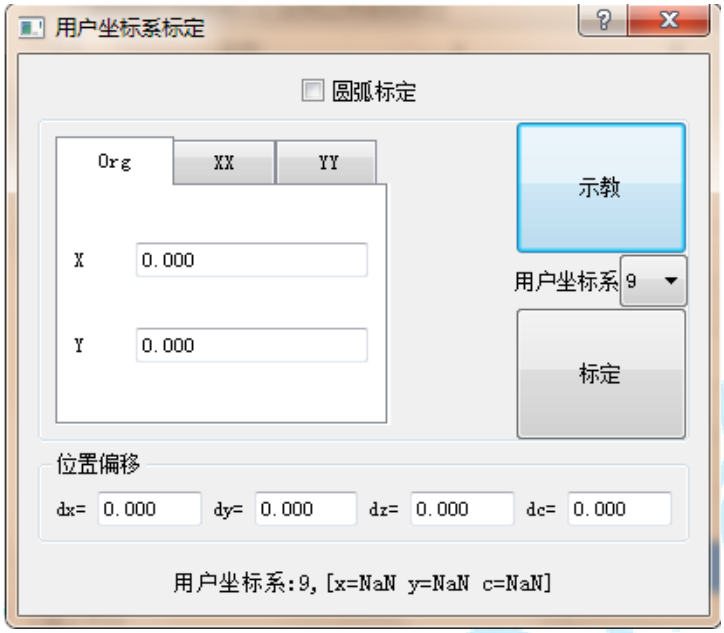
（2） 点击“ ”图表，进入【用户坐标系标定】界面；

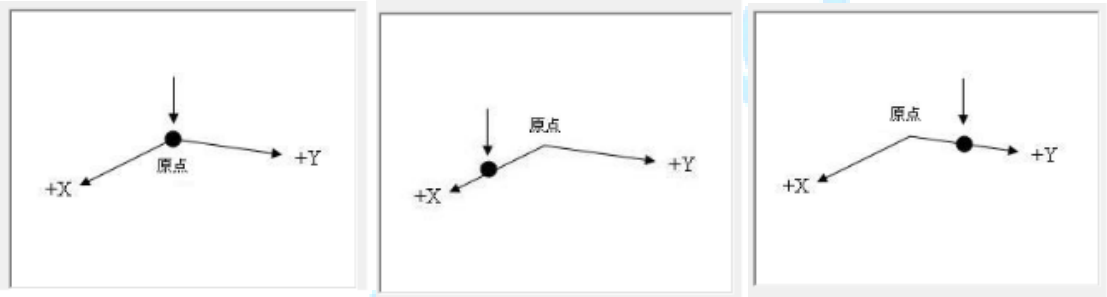
（3） 选择“用户坐标系”**Org** 点，在笛卡尔坐标系下，手动调整工具末端与将要新建的用户坐标系的原点重合，点击示教(当前位置赋值给 Org 点)；

（4） 选择“用户坐标系”**XX** 点，在笛卡尔坐标系下，沿着工件 X 方向移动,移动距离尽可能大点，选择合适的位置，点击示教(当前位置赋值给 XX 点，移动过程一定不能旋转 C 轴，否则计算的值出错)；

（5） 选择“用户坐标系”**YY** 点，在笛卡尔坐标系下，沿着工件的 Y 方向移动，移动距离尽可能大点，选择合适的位置，点击示教(当前位置赋值给 YY 点，移动过程一定不能旋转 C 轴，否则计算的值出错)。

（6） Org、XX、YY 示教完毕后，点击【标定】按钮，生成用户坐标，如要修改用户坐标号，重新选择一个用户号 n（n = 1 至 19），点击【标定】即重新生成)。查看生成的用户坐标参数（x，y，c）

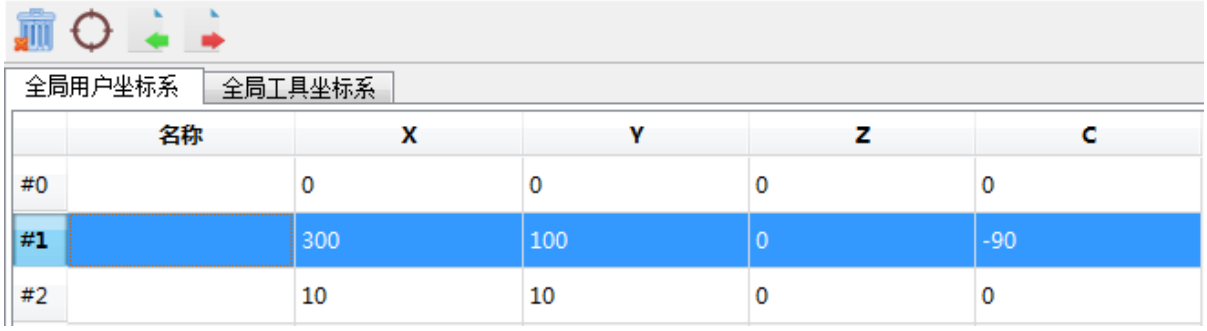




➢ **圆弧用户坐标系**

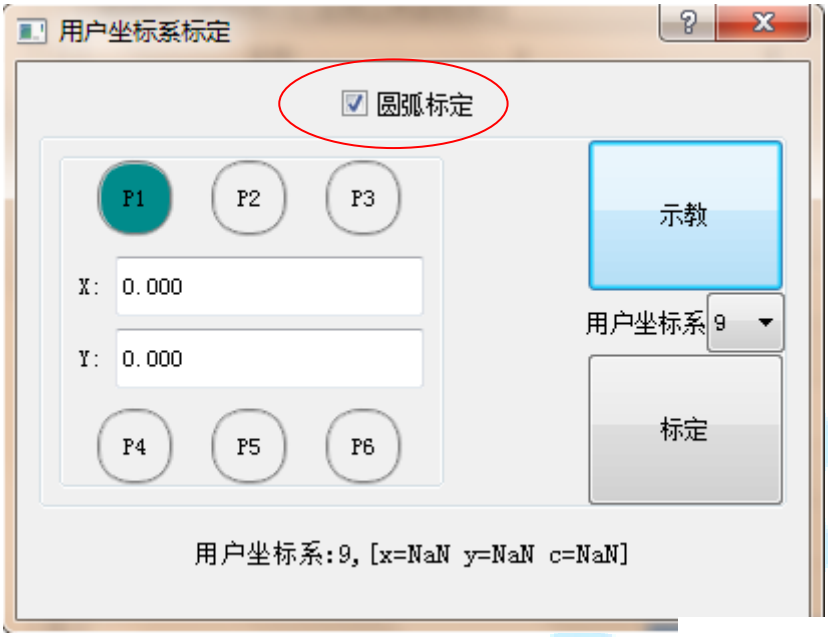
标定步骤如下：

(1) 从 1~19 个用户中选中一个用户号（选中之后用户号所在的行由白色或浅灰色标记为蓝色）；

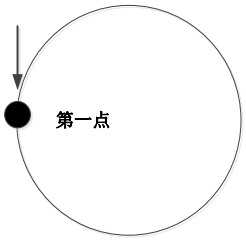


	名称	X	Y	Z	C
#0		0	0	0	0
#1		300	100	0	-90
#2		10	10	0	0

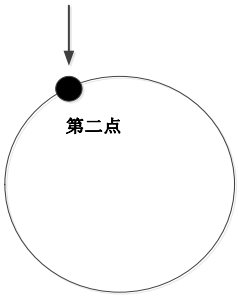
(2) 点击“ ”图表，进入【用户坐标系标定】界面，并勾选“圆弧标定”，点击“标定”进入【圆弧用户坐标标定】界面；



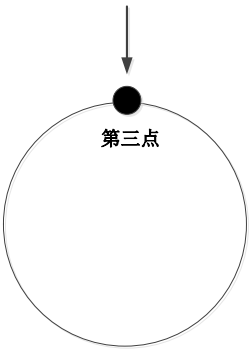
(3) 选中“P1”点，在笛卡尔坐标系下，手动调整机器人末端与圆弧上的第一个点（任意指定）重合，点击“示教”（把当前位置赋值给 P1 点）；



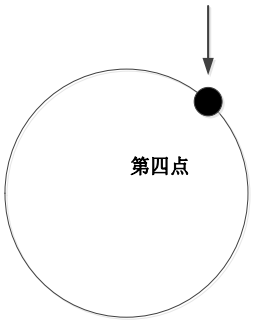
(4) 选中“P2”点，在笛卡尔坐标系下，手动调整机器人末端与圆弧上的第二个点（任意指定，异于第一点）重合，点击“示教”（把当前位置赋值给 P2 点，**移动过程一定不能旋转 C 轴，否则计算的值出错**）；



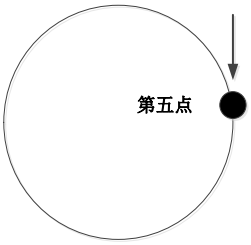
(5) 选中“P3”点，在笛卡尔坐标系下，手动调整机器人末端与圆弧上的第三个点（任意指定，异于第一、二点）重合，点击“示教”（把当前位置赋值给 P3 点，**移动过程一定不能旋转 C 轴，否则计算的值出错**）；



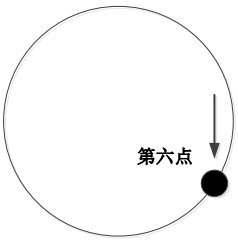
(6) 选中“P4”点，在笛卡尔坐标系下，手动调整机器人末端与圆弧上的第四个点（任意指定，异于第一、二、三点）重合，点击“示教”（把当前位置赋值给 P4 点，**移动过程一定不能旋转 C 轴，否则计算的值出错**）；



(7) 选中“P5”点，在笛卡尔坐标系下，手动调整机器人末端与圆弧上的第五个点（任意指定，异于第一、二、三、四点）重合，点击“示教”（把当前位置赋值给 P5 点，**移动过程一定不能旋转 C 轴，否则计算的值出错**）；



(8) 选中“P6”点，在笛卡尔坐标系下，手动调整机器人末端与圆弧上的第六个点（任意指定，异于第一、二、三、四，五点）重合，点击“示教”（把当前位置赋值给 P6 点，**移动过程一定不能旋转 C 轴，否则计算的值出错**）；



(9) P1、P2、P3、P4、P5、P6 示教完成后，点击“标定”，生成用户坐标，如要修改用户坐标号，选择用户 n（n = 1~19），点击“标定”即重新生成。查看生成的用户坐标参数（X,Y,Z,C）。

✧ 小技巧：坐标偏移指的是用户的原点在 x,y,c 方向上的偏移；

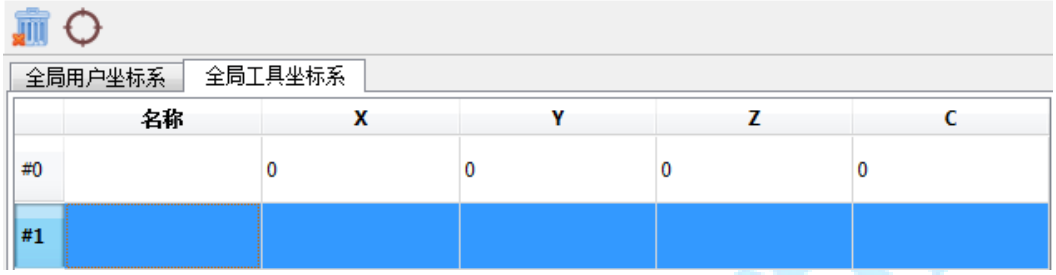
✧ 小技巧：若写入坐标偏移，需点击“标定”重新计算。

✧ 小技巧：新建用户涉及到的点位示教都必须在机器人用户 0 坐标系下获取，即新建用户只能基于用户 0 下完成，不能在用户 0 外的其它用户（1~19）下新建其它新的用户。

1.5.1.2 工具坐标系标定

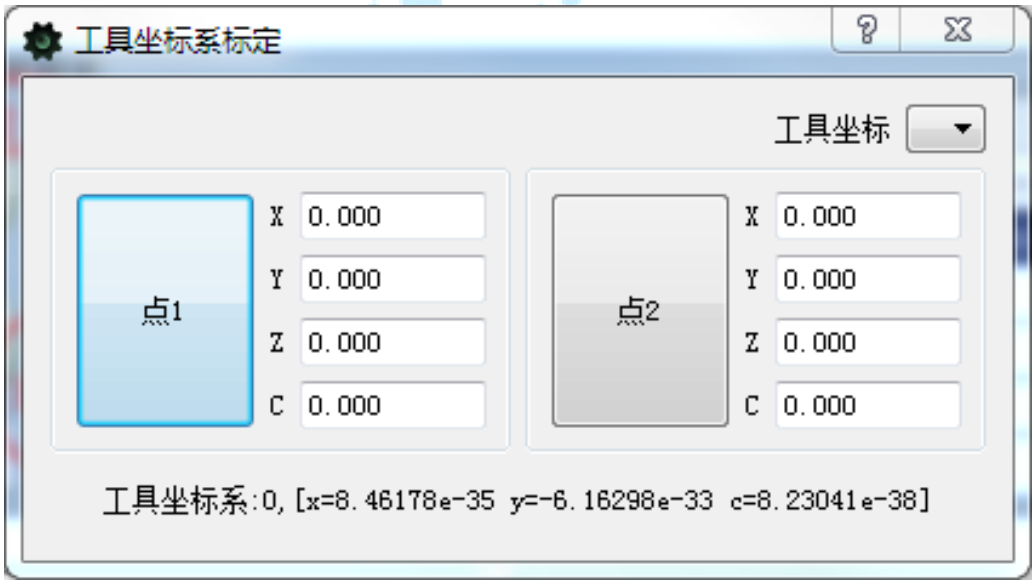
工具坐标系标定采用两点示教法，具体标定步骤如下：

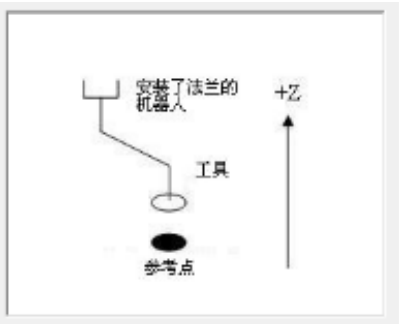
（1）在工具号 1~9 选中一个没有被占用的工具号（若选中，则该行所在的颜色标记为蓝色）；

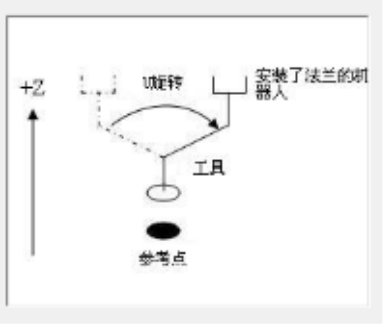


（2）两点示教法：**手动模式下**调整机器人使工具末端与参考点重合，点击“点 1”（则当前位置赋值给点 1）；

（3）**笛卡尔坐标系下旋转 C 轴（大约 60~180 度）后**，移动 X 轴和 Y 轴（不能再旋转 C 轴）再次使工具末端与参考点重合，点击“点 2” (当前位置赋值给点 2。则计算的工具参数（X/Y/C）记录在选中的行。





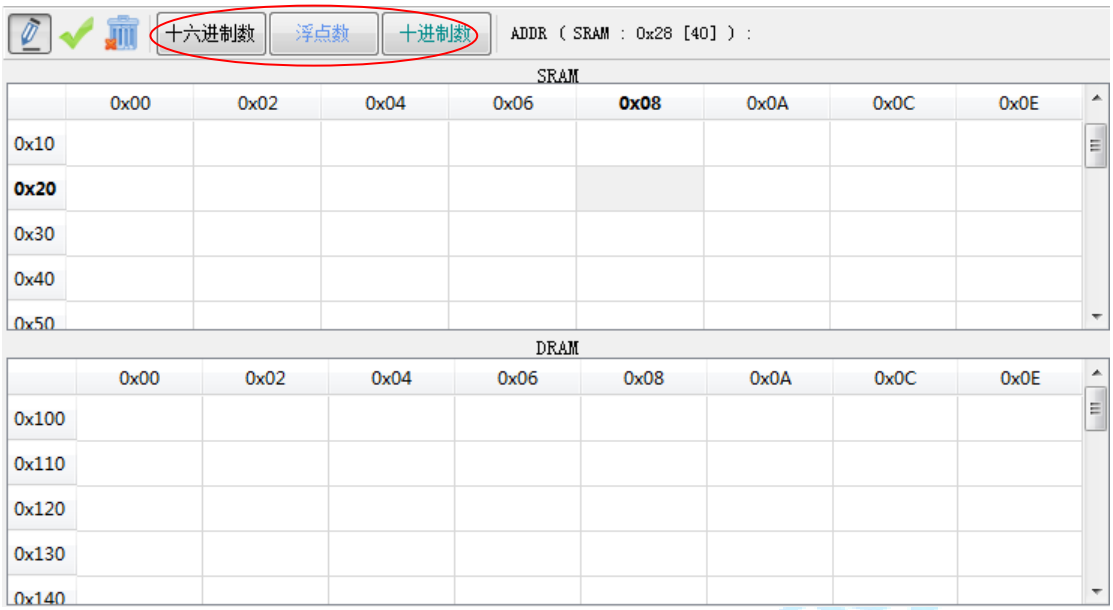


1.5.2 RAM 数据

点击“ RAM 数据 ”，进入外设界面，处理外部 PLC 设备与 QC410 控制器之间关于 Modbus 通讯数据存储的问题。包含两种存储方式：SRAM（staic RAM，静态随机存储器）和 DRAM（Dynamic RAM, 动态随机存储器）。SRAM 是非易失性存储器，在掉电的情况下并不会丢失所存储的数据；DRAM 是一种易失性存储器，在掉电的情况下所存储的数据会丢失。

QC410 控制器与外部设备通过 Modbus 通讯，QC410 控制器作为从站，外部设备作为主站。存储的数据长度 32 位 ，每个数据须占地址的长度为 2。SRAM 和 DRAM 各存储 128 个数据，SRAM 存储地址从 0 到 255，DRAM 存储地址从 256 到 511。外部设备通过 Modbus 通讯写入到控制器的数据，可在全局数据界面显示出来。

可编辑保存清除所有数据数据类型



✧ **小技巧：**手动编辑 RAM 地址中的数据时，编辑按钮须处于可编辑状态；编辑 RAM 地址中数据正确的操作步骤是：1. 编辑按钮切换到可编辑状态；2.往对应地址写入数据；3. 点击保存；4. 切换到非编辑状态，这样地址表中的数据才能刷新生效。

✧ **小技巧：**可编辑状态下才能清除 RAM 所有地址中的数据（全部清除，不能单独清除某一个地址中的数据）；

✧ **小技巧：**读取的数据格式需与 RAM 地址中的数据格式一致，例如：写入到控制器的是浮点数，读取一定要按浮点数读取。

1.5.3 共享点位表

共享点位表共有 500 个（G0000~G0499）公共点位数据，适用于 WorkSpace 中的所有工程。共享点位表中的点位用法与工程里点位表的点法用法稍有不同，共享点位表的点位名字为 g0、g1…..g499. 例如：MovP(g1).

	点名称	X	Y	Z	C	用户坐标	手系	J1标志	J2标志
G0000	Home	100	100	0.0	0.0	0	LHand	0	0
G0001	fangliao	400	-0.000383...	0	-5.49316e...	0	LHand	0	0
G0002	teach_2	300	99.9999	-20.0001	29.9999	0	LHand	0	0
G0003		367.793	86.755	-23.363	0	0	LHand	0	0
G0004		300	100	-50	0.0	0	LHand	0	0
G0005*									
G0006*									
G0007*									
G0008*									
G0009*									
G0010	new_10	400	-0.000383...	0	-5.49316e...	0	LHand	0	0
G0011		400	-0.000383...	0	-5.49316e...	0	LHand	0	0

✧ **小技巧：Point\teach\new** 这三个指令只能适用于操作共享点位表里的点位数据，具体用法可参考 QC410-AR 语言手册说明书。

1.5.4 Modbus

Modbus 从站地址表共有 21 个功能块，每个功能块对应的是地址的分配，以及地址的权限、地址长度和对应功能描述。此地址仅适于 QC410 作为 Modbus 从站，外围设备作为 Modbus 主站时使用。具体使用可参考 **Modbus 通讯机器人作为从站**说明书。

	address	W/R	length	描述
1	0x000000 / 000000	W&R	256	(32位)断电记忆共享地址表-SRAM
2	0x000100 / 000256	W&R	256	(32位)断电不记忆共享地址表-DRAM
3	0x000200 / 000512	R	4	(16位)基本输入IO字地址
4	0x000204 / 000516	W&R	8	(16位)基本输出IO字地址,虚拟辅助点MO
5	0x00020c / 000524	R	4	(16位)扩展输入IO
6	0x000210 / 000528	W&R	4	(16位)扩展输出IO
7	0x000214 / 000532	W	1	(16位)系统AR程序自动运行控制
8	0x000215 / 000533	R	10	(float)当前笛卡尔坐标位置
9	0x00021f / 000543	R	10	(float)当前关节坐标位置
10	0x000229 / 000553	W&R	1	(16bit)当前系统速率
11	0x00022a / 000554	R	1	(16bit)当前系统状态
12	0x00022b / 000555	W&R	1	(16bit)当前系统模式
20	0x00022c / 000556	W	1	(16bit)示教共享点位表
14	0x00022d / 000557	R	8	(16bit)系统报警号
15	0x000235 / 000565	W&R	1	(16bit)系统使能控制
16	0x000236 / 000566	W	2	(32bit)手动连续/单步模式
17	0x000238 / 000568	W	1	(16bit)手动笛卡尔轴移动
18	0x000239 / 000569	W	1	(16bit)手动关节移动
19	0x00023a / 000570	W	2	(32bit)跟踪到点位表
20	0x00023c / 000572	W&R	5000	(float)共享点位表读写
21	0x0015c4 / 005572	R	1	(16bit)Modbus控制错误码

1.5.5 快速稳停

当现场速度较快的情况下，伺服响应有一定的滞后性，就会影响到机器人到达目标点位的时间。此时可以在运动指令后使用 waitrealpos()，即等待运动到位停稳指令，该指令几乎不影响节拍。若此指令未有明显改变，通常会加入延时，但是加入延时后，会影响节拍，当需要加入 300ms 以上的节拍时，可通过此功能减少将近 200ms 的延时达到同样的稳停效果。

1. 点击【系统
2. 】图标进入机器人系统设置→点击【快速稳停】进入快速稳停调节界面；

分别是【编辑】、【保存】、【辨识】、【加载】功能。

3. 点击【编辑】图标，配置快速稳停的参数“④、⑤”。其中稳停时间是指：点位运动时到达目标点后进入稳定阶段需要的时间一般设置为 80ms~200ms。超调脉冲是指：到达目标点后可以接受的超过的脉冲范围，一般设置为 100~250。因为无法稳定，所以在到位后，脉冲值会有一定波动。我司现采用的都是 131072 的编码器，也就是说编码器一圈的距离可以走过 131072 个脉冲。因此可估计出 100~250 个脉冲大概是走过多少距离。此距离反映在到达目标点后，会超过此超调脉冲值对应的距离再回到目标点。

注：当设置的稳停时间过小时，由于现场的负载、速度不同可能使本功能无法满足设定要求进而导致辨识不成功。

4. 在可编辑状态下，选中某一行的起始点—>选择需要导入的点位表—>选择该点位表中需要作为起始点的点位，其中“0”代表点位表中“P0000”点依次类推。如需更改，直接重新导入新的点位即可，原来的点位会直接被覆盖。如需删除，选中点位“②”，点击【清除点位即可】，点位中前四位代表 X，Y，Z，C，或者 J1，J2，J3，J4，最后一位代表左右手系“0”为左手系，“1”为右手系。

- 每个点位中【Gain1】、【Gain2】、【Gain3】、【Gain4】指的是每个轴辨识完获得的辨识参数，目前只有前两个轴可以辨识，因此辨识完成以后 Gain3 和 Gain4 仍然是 0；
- 【标志】是指是否完成辨识。辨识前“未完成”，辨识完成后更新“完成”；
- 默认每个点位的运动【指令类型】为“MovP”，【参数 1】、【参数 2】、【参数 3】是指如果用拱形指令对应拱形参数。

注：目前只有 MArchP、MovP、MovL 三种运动指令可支持。这里需要手动将需要辨识的点位的【标志】改为“未完成”。

5. 编辑完成之后，再次点击编辑退出可编辑状态，并点击保存图标。此时，需要辨识的点位的起始终止都已确定可以点击辨识开始自动完成辨识。辨识完成后【标志】位更新为“完成”。

6. 在 AR 中需要使用此功能时，需严格按照“指令类型”并且和程序中起始终止点位保持一致，在需要使用快速稳停功能的运动指令中加入标志位“QS=1”。

   例如：MovP(p1,“QS=1”)

注：总的点位顺序无需保持一致，只需起点终点相同。

1.5.6 安全空间

安全空间：机器人末端执行器能在工作空间中活动的最大范围，通俗讲就是给机器人笛卡尔坐标系下的 X、Y、Z 重新设置一个有效的工作区域。

	最小 X	最大 X	最小 Y	最大 Y	最小 Z	最大 Z	开关	内部极性	输出
1									
2									
3									
4									
5									
6									
7									
8									
9									
10									
限位	200	350	-100	100	-50	-10	打开	ON	O_0

- 根据实际工艺需要设定合适的 X/Y/Z 的范围；
- **开关**包括**打开**和**关闭**两种操作，其中“**打开**”指打开工作空间功能，“**关闭**”指禁用工作空间功能；
- **内部极性**分为 **ON** 和 **OFF** 两种，其中 **ON** 表示机器人末端的位置在设定的工作空间，对应的输出端口会打开，**OFF** 表示机器人末端的位置超出设定的工作空间，对应的输出端口才会打开；
- **输出**根据实际的电气接线设定对应的输出端口号；
- **1~10** 号为普通的安全空间设置，当机器人末端的位置超过设定的工作空间，机器人还可以继续运行，只是对应的输出状态会改变；
- **限位**为特定的安全空间设置。

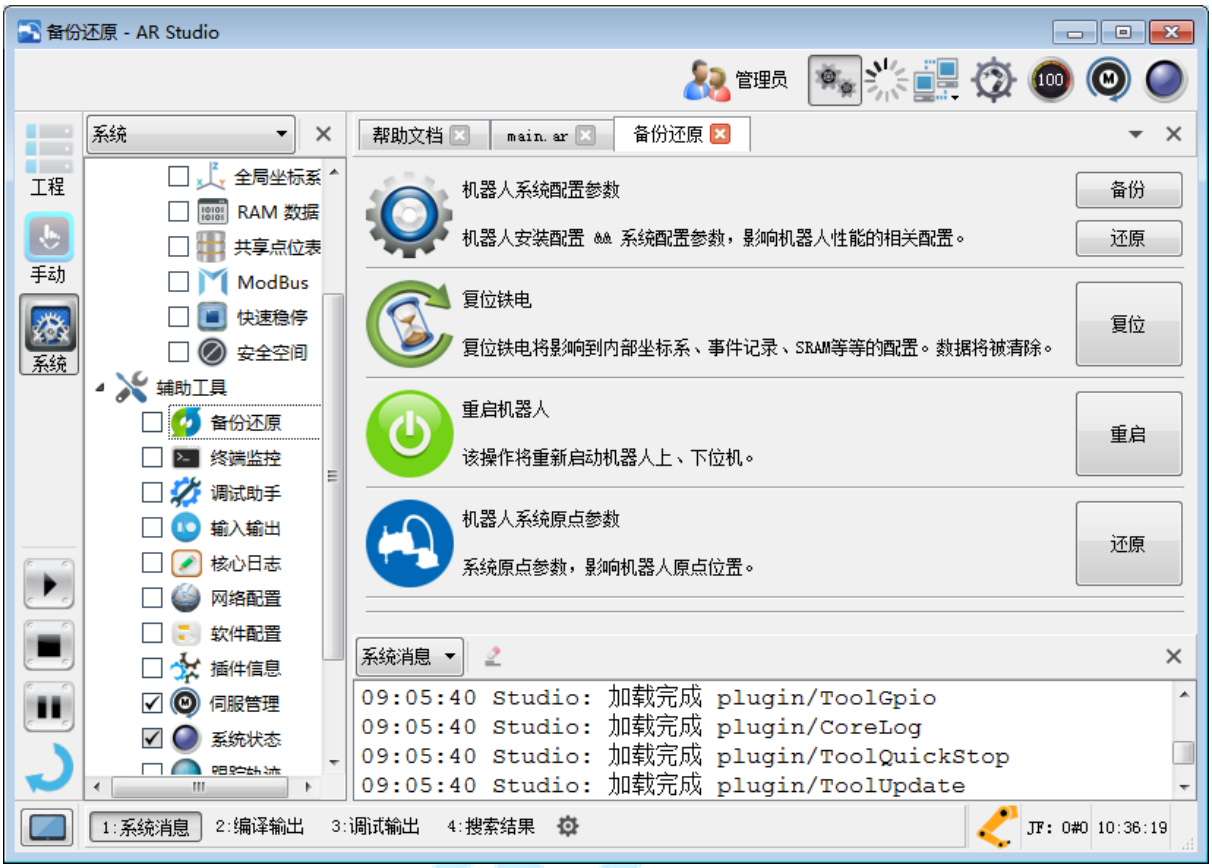
（1） 手动运行：若机器人末端位置启动之前处于**安全空间限位**之内，当机器人末端到达设定的工作空间的边缘位置时，此时手动继续控制机器人移动，机器人会一直停留在边缘位置，不会越过设定的安全空间；若机器人末端位置启动之前处于**安全空间限位**之外，则会有报警“20073：超出工作空间限位”提示，可手动推动机器人到安全空间限位内，再按复位可清除报警；

（2） 自动运行：运动指令中用到的点位处于安全空间限位之外，会给出“20073：超出工作空间限位”报警；

1.6 辅助工具

调试工具是在机器人调试过程中的一些辅助工具，包括通信调试助手、输入输出仿真监控、语言设置等。接下来我们会介绍一些在调试应用过程中要用到的一些小工具和操作技巧。

1.6.1 备份还原

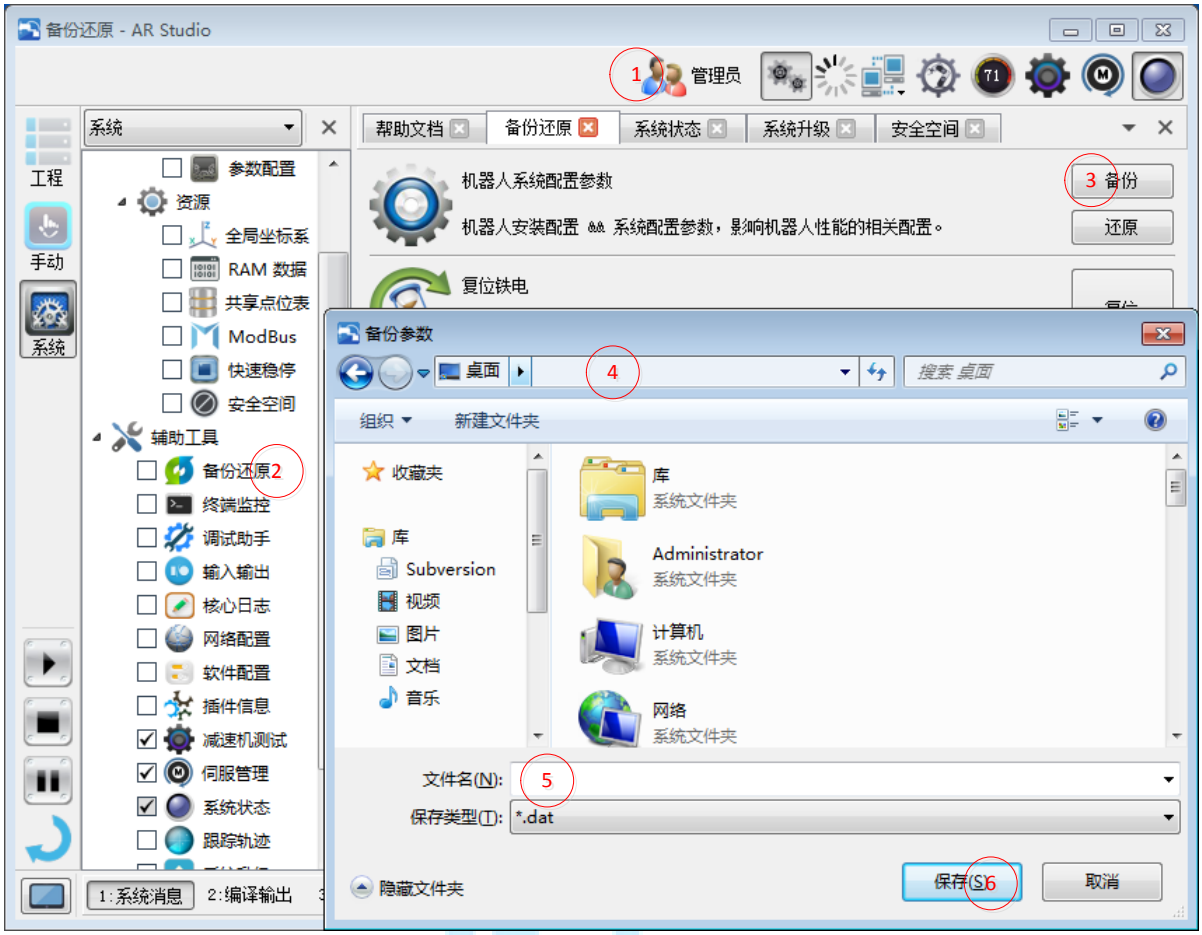


备份还原主要包括四个部分内容：机器人系统配置参数的备份和还原、复位铁电、重启机器人、机器人系统原点参数还原。

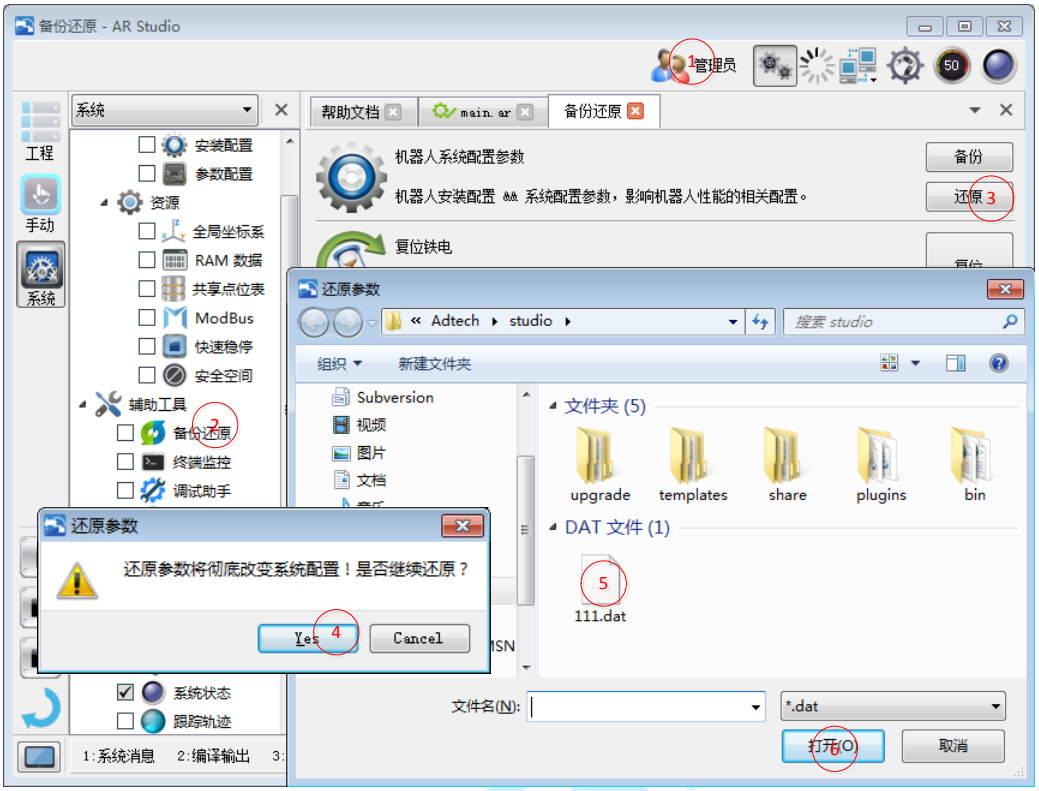
> ✧ **小技巧：**备份还原操作需基于管理员或厂家权限。

1.6.1.1 机器人系统配置参数

（1）**备份：**指的是将一体机里的系统安装配置与参数配置的参数拷入用户的个人电脑上。备份时需要在“5”里输入文件名，文件以**.dat** 结尾。



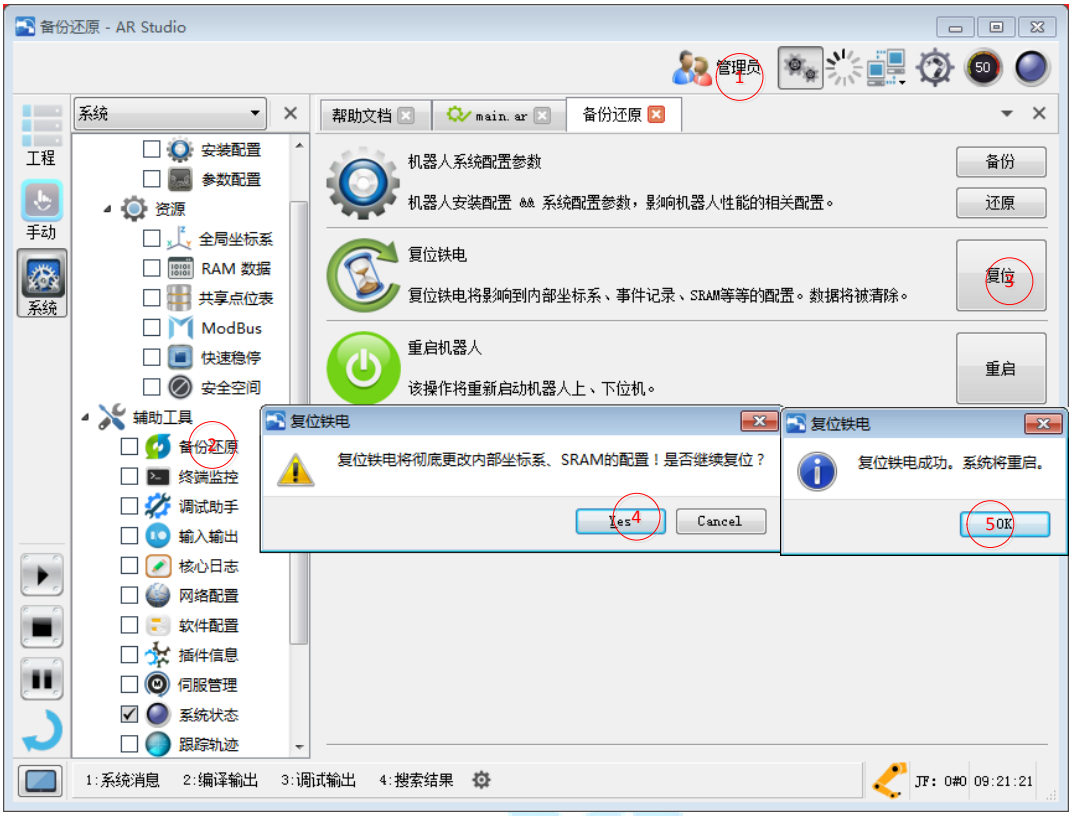
（2）**还原：**是将外部的系统安装配置与参数配置的参数重新拷一体机中。还原时需要选择原先保存至电脑的参数文件，其格式为“.dat”。



注意：当点击【还原】参数按钮无响应时，请检查权限管理是否处于“工人或操作员”模式，若是，请登录到管理员及以上的权限。

1.6.1.2 复位铁电

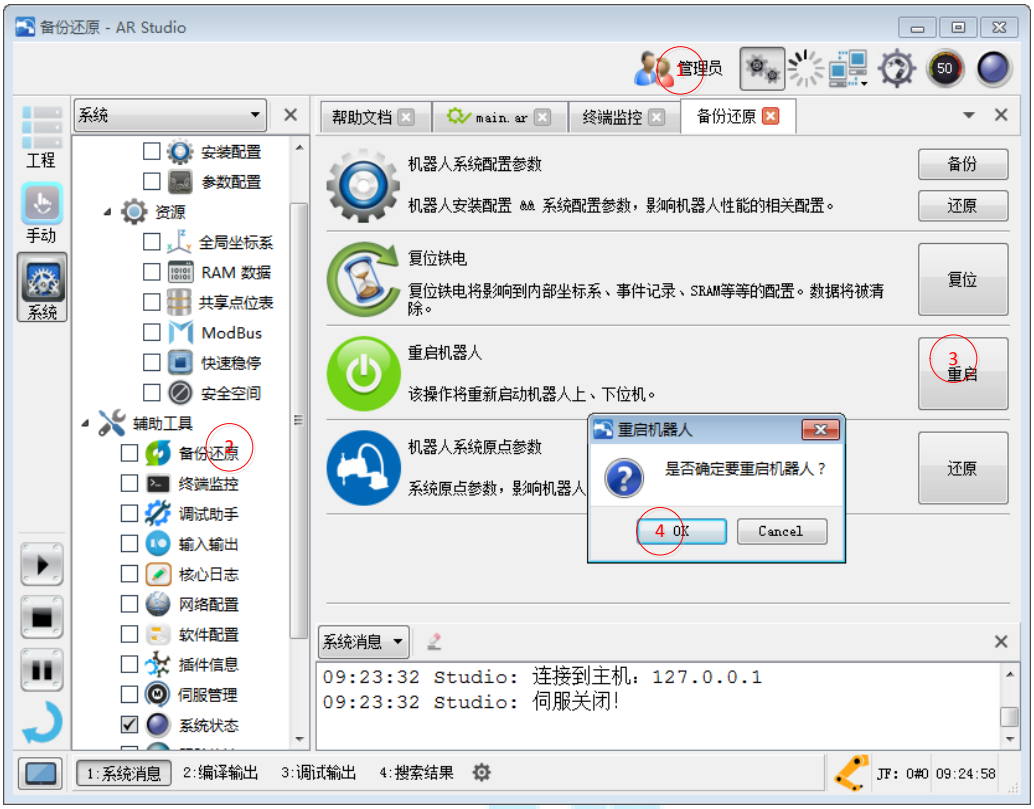
用来清除内部坐标系、事件记录、SRAM 等等配置，清除成功后一体机需要重新上电重启。这里内部坐标系指的是系统菜单下资源列表里的全局坐标系，事件记录指的是系统状态指示灯里的报警记录，SRAM 为静态断电不失变量。



注意：当点击【还原】参数按钮无响应时，请检查权限管理是否处于“工人或操作员”模式，若是，请登录到管理员及以上的权限。

1.6.1.3 重启机器人

软重启的方式将上位机软件和控制器重新自启动。



1.6.1.4 机器人系统原点参数

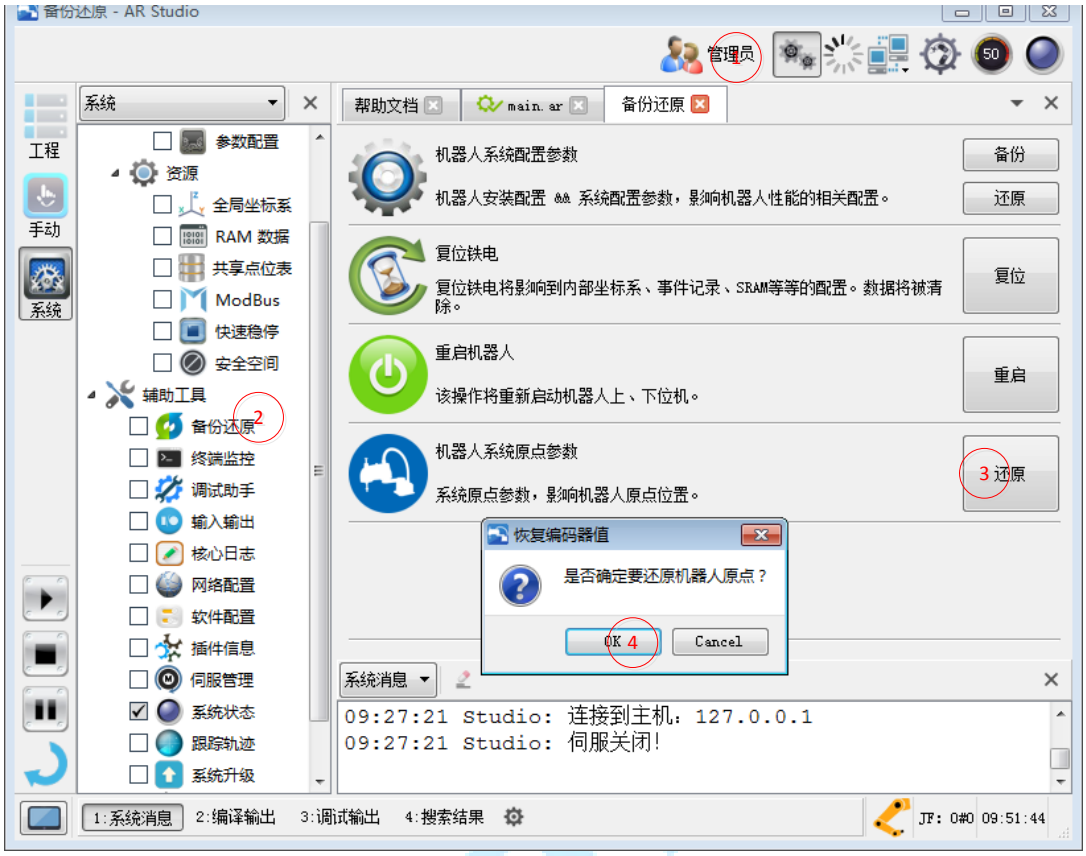
机器人系统原点参数的还原操作适用于机器人原点参数丢失，可通过出厂备份的原点位置或手动输入原点位置来还原机器人原点。具体还原操作如下：

(1) 登录管理员权限；

(2) 鼠标左键双击【辅助工具】中的“备份还原”进入【备份还原】界面；

(3) 点击“机器人系统还原参数”中的“还原”按钮；

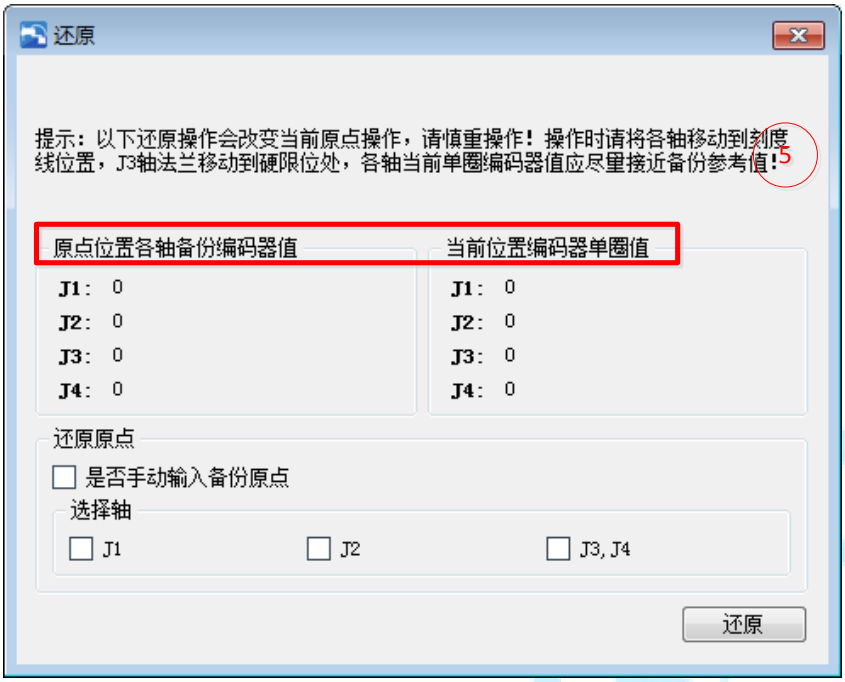
(4) 在弹出的“恢复编码器”对话框中，选择“OK”确认进行原点还原操作；



(5) 还原之前须将机器人的第一轴、第二轴、第四轴移至与刻度线对齐，第三轴向上移动到法兰的硬限位处。

查看机器人“当前位置编码器的单圈值”是否接近“原点位置各轴备份的编码器值”，这两个值的差值应在半圈 65536 个脉冲之内（以 17 位编码器为例），否则，须重复之前的操作再次还原。

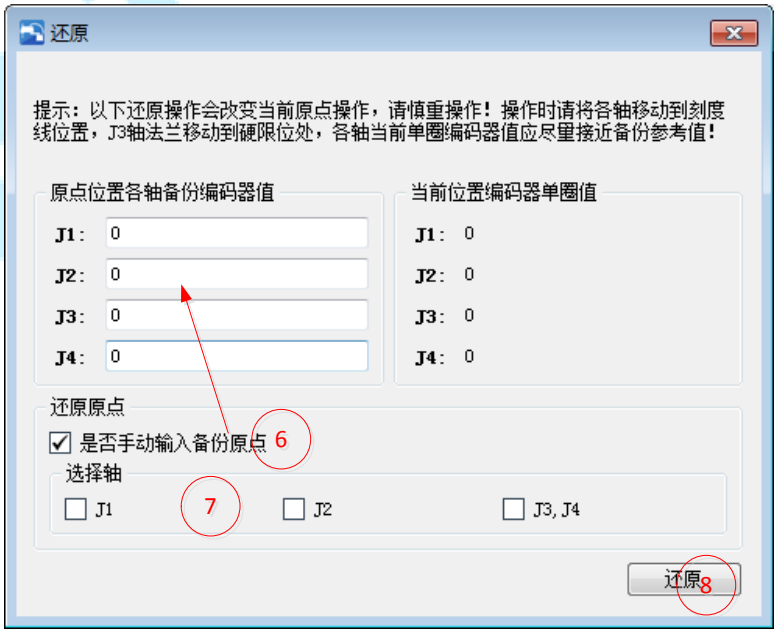
※ 原点位置各轴备份的编码器值可通过机器人本体标签上的原点位置查看；



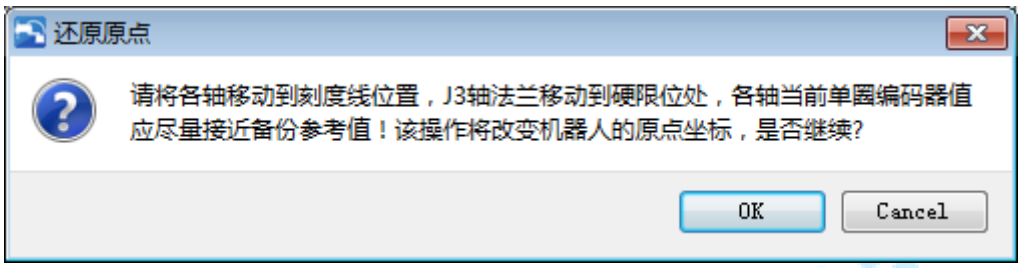
(6) “是否手动输入备份原点”适用于现场更换本体或控制器的操作。若勾选，则须在“原点位置各轴备份编码器值”一栏手动输入机器人本体标签上写的对应四个轴的原点编码器值；若不勾选，默认还原的就是机器人系统参数中备份的原点编码器值。

(7) “选择轴”适用于选择性地还原某一个或几个轴的编码器位置；

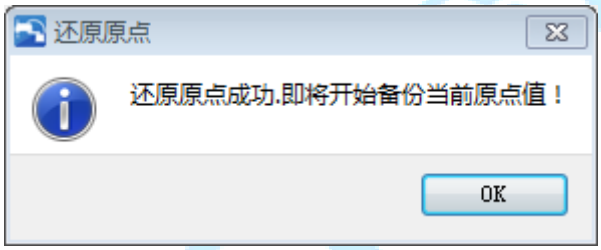
(8) 点击“还原”开始还原；



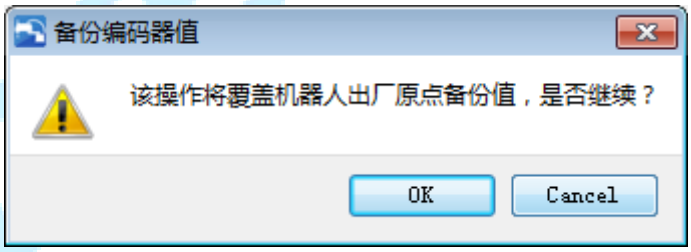
(9) 还原之前，再次给出了“还原原点”确认的提示“请将各轴移动到刻度线位置，J3 轴法兰移动到硬限位处，各轴当前单圈编码器值应尽量接近备份参考值！该操作将改变机器人的原点坐标，是否继续？”若，确认还原原点以及准备工作已就绪，就点击“OK”，则会自动开始还原原点。



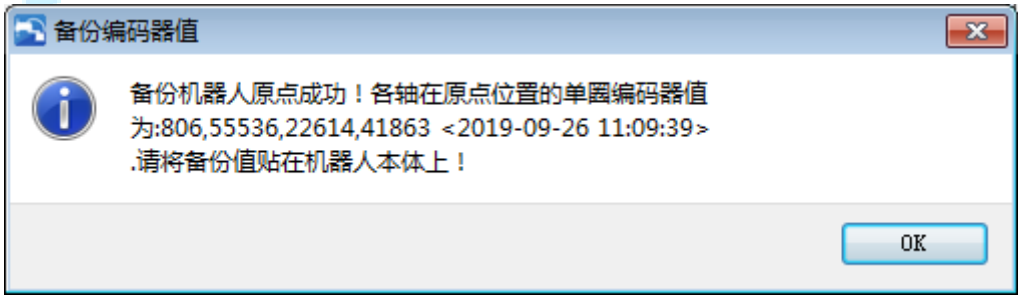
还原成功，会弹出“还原原点成功。即将开始备份当前原点值！”对话框，点击“OK”，开始备份。



备份之前，弹出“改操作将覆盖机器人出厂原点备份值，是否继续？”的提示，若继续点击“OK”即可。



备份原点的信息包括各轴在原点位置的单圈编码器值以及对应的时间日期。备份机器人原点成功，会弹出如下信息的提示：



注意：备份完成后，若最新备份的原点位置较之前的有改动，须最新的原点位置信息贴

在机器人本体上。

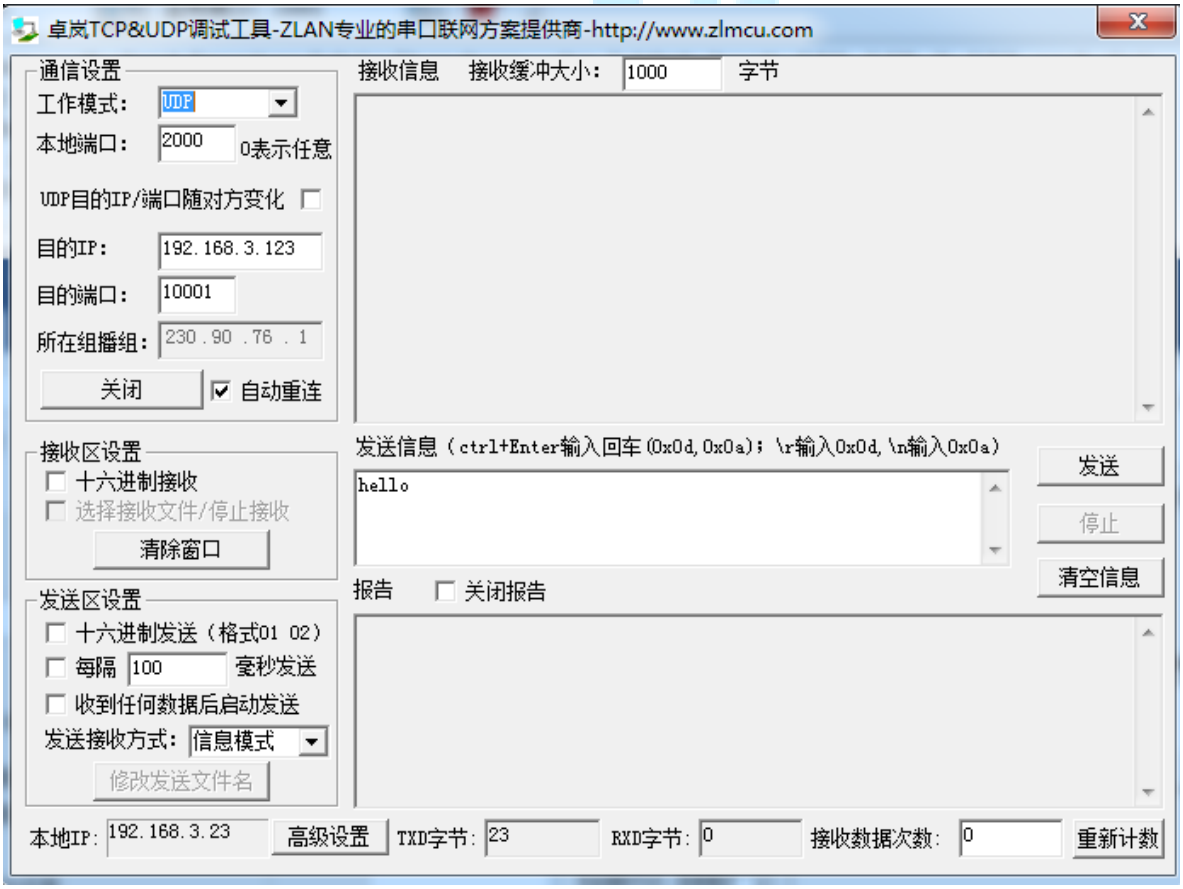
1.6.2 终端监控

终端是用来监视一体机某个端口下常用目录/文件查询、网络通讯任务、系统管理任务、用户和组管理、文件权限管理。使用要求：ARStudio 需连接一体机的情况下。此功能的操作与 Linux 系统下的终端操作几乎无异。具体操作用户可自行参考 Linux 下终端操作。

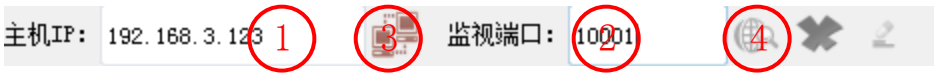
这里以网络调试助手代替外部 PLC 进行说明终端的通讯功能。QC410 支持四种网络通讯方式，UDP 服务器、UDP 客户端、TCP 服务器、TCP 客户端。

第一种. 机器人做 UDP 通讯方式，不管是做服务器还是客户端。

Step1. 配置网络调试助手。其中目的 IP 为一体机 IP，目的端口需要在机器人网络配置中进行配置。

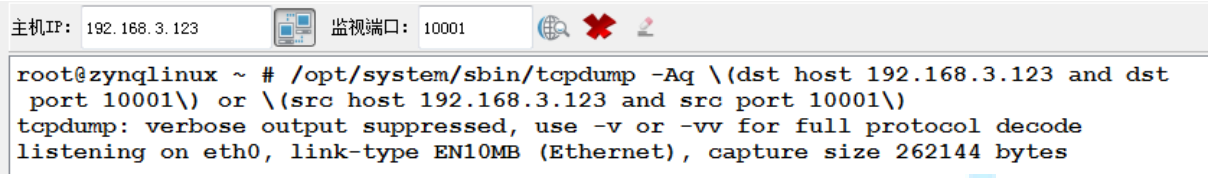


Step3. 打开终端界面，配置主机 IP 和监视端口号。之后点击连接和监视按钮。



Step4. 打开成功后界面如下：

点击完 监视以后，会弹出 tcpdump -Aq 指令的部分用法。



- \(dst host 192.168.3.123 and dst port 10001\)表示从外部发送给目的主机 IP 和端口号为 192.168.3.123、10001 的数据都会被截获并在此界面打印出来。
- \(src host 192.168.3.123 and src port 10001\) 表示从主机 IP 和端口号为 192.168.3.123、10001 发出的数据都会被截获并在此界面打印出来。

因此，如果直接打开监视端口截获数据的话。不管是 udp 还是 tcp 通讯方式，也不管谁作为发送方，这里都会被截获。

例如：Step1 图片当中显示的是由 IP 为 192.168.3.23 和端口号 2000 发出的字符串内容，但是从如下终端打印的信息看无法区分发送内容的起始位置和结束位置。

```
root@zynqlinux ~ # /opt/system/sbin/tcpdump -Aq \(dst host 192.168.3.123 and dst
 port 10001\) or \(src host 192.168.3.123 and src port 10001\)
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
23:23:03.871056 IP 192.168.3.23.2000 > 192.168.3.123.10001: UDP, length 5
E..!|B....6........{..'....>hello............
```

Step5. 因此如果需要具体区分，则需要修改此命令。首先点击 终止当前监视进程。

成功终止后终端会出现如下界面

```
root@zynqlinux ~ # /opt/system/sbin/tcpdump -Aq \(dst host 192.168.3.123 and dst
 port 10001\) or \(src host 192.168.3.123 and src port 10001\)
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
23:23:03.871056 IP 192.168.3.23.2000 > 192.168.3.123.10001: UDP, length 5
E..!|B....6........{..'....>hello............
^C
1 packet captured
1 packet received by filter
0 packets dropped by kernel
root@zynqlinux ~ #
```

Step6. 通过键盘上的“←”，“→”，“↑”，“↓”可以调出 tcpdump –Aq 命令，在此基础上进行修改。例如点击“↑”，会弹出指令历史记录如下

```
root@zynqlinux ~ # /opt/system/sbin/tcpdump -Aq \(dst host 192.168.3.123 and dst
 port 10001\) or \(src host 192.168.3.123 and src port 10001\)
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
23:23:03.871056 IP 192.168.3.23.2000 > 192.168.3.123.10001: UDP, length 5
E..!|B....6........{..'....>hello.............
^C
1 packet captured
1 packet received by filter
0 packets dropped by kernel
root@zynqlinux ~ # /opt/system/sbin/tcpdump -Aq \(dst host 192.168.3.123 and dst
 port 10001\) or \(src host 192.168.3.123 and src port 10001\)
```

现在如果需要只监视 udp 且机器人做为目的地址和端口的需要改为

```
root@zynqlinux ~ # /opt/system/sbin/tcpdump -AqX \(dst host 192.168.3.123 and ds
t port 10001 and udp \)
```

如果需要只监视机器人做为源地址发送方，那么需要做如下修改：

```
root@zynqlinux ~ # /opt/system/sbin/tcpdump -AqX \(src host 192.168.3.123 and sr
c port 10001 and udp \)
```

修改成功后点击键盘“Enter”即可弹出对应信息。

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

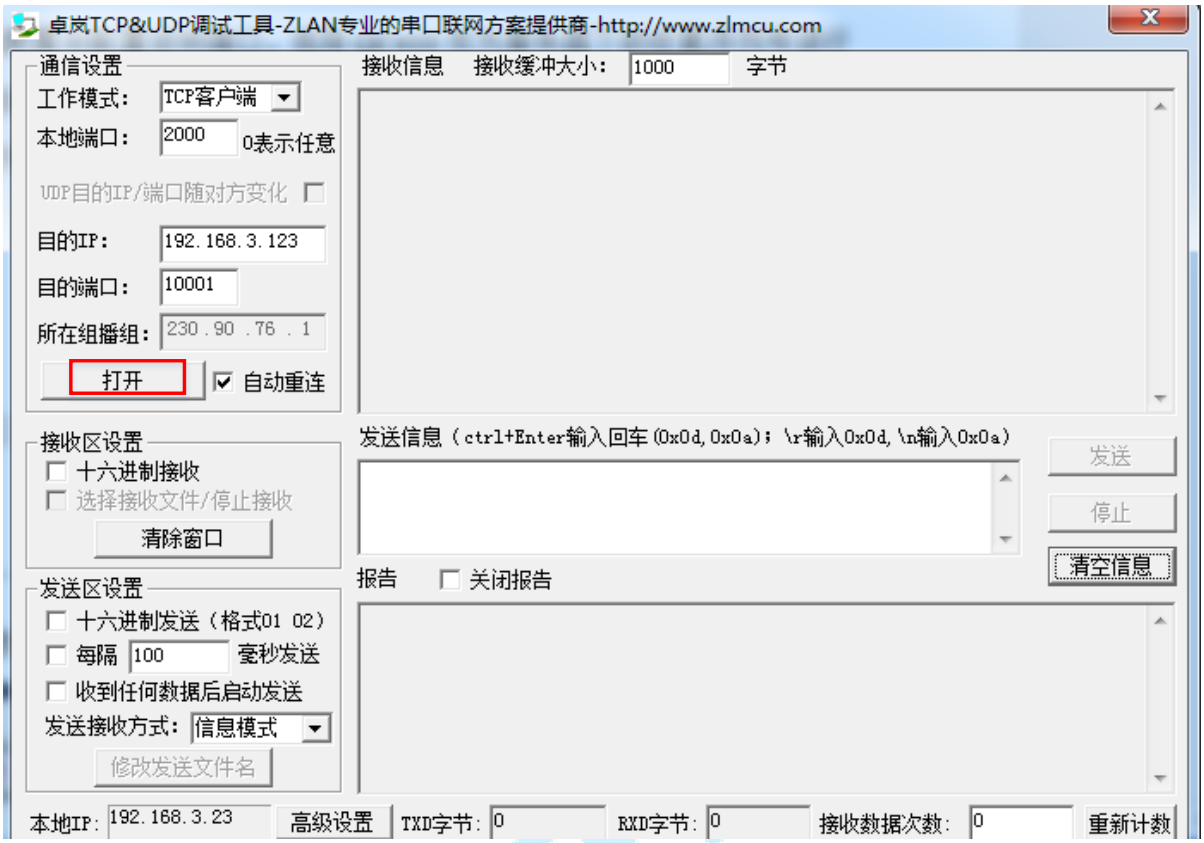
这时调试助手再次发送，发现终端打印结果如下。

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
17:15:11.435100 IP 192.168.3.23.2000 > 192.168.3.123.10001: UDP, length 5
        0x0000:  4500 0021 241c 0000 8011 8ecd c0a8 0317  E..!$...........
        0x0010:  c0a8 037b 07d0 2711 000d 053e 6865 6c6c  ...{..'....>hell
        0x0020:  6f00 0000 0000 0000 0000 0000 0000       o.............
```

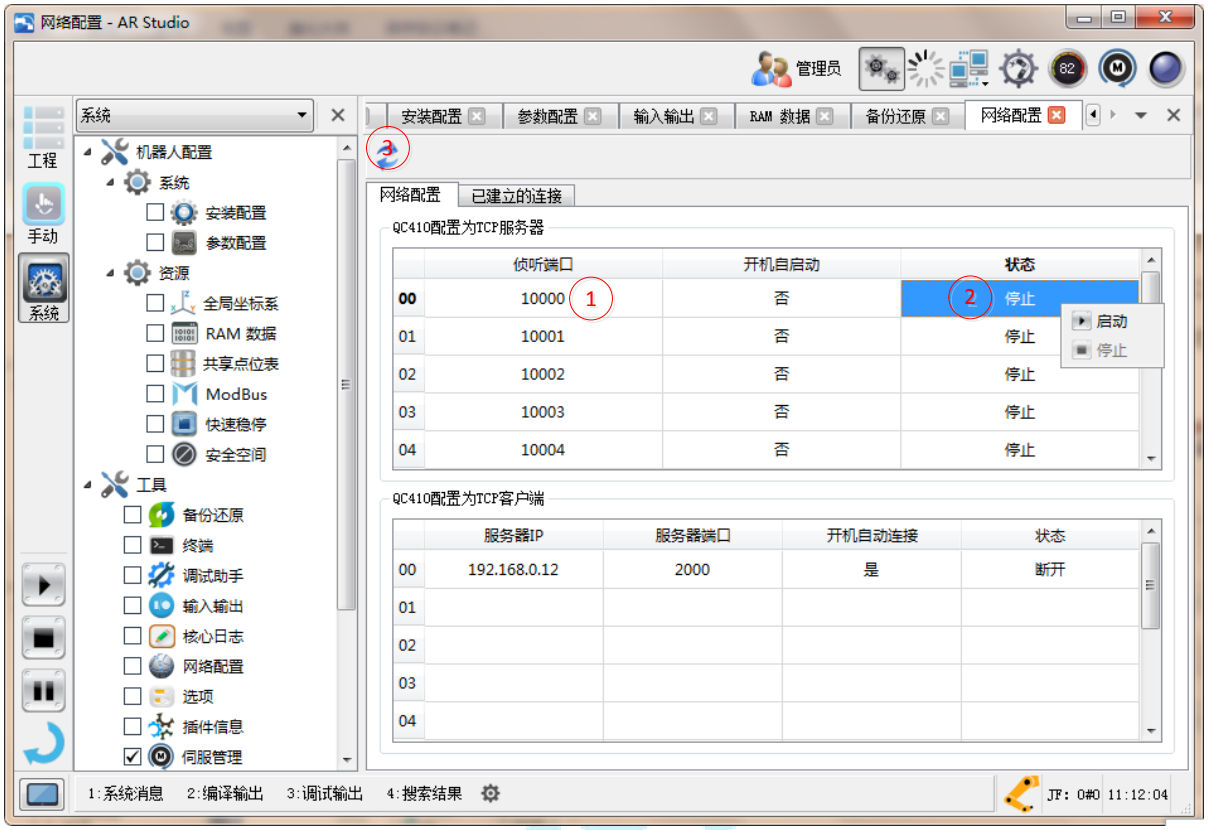
这里一共有三行十六进制的数据作为 UDP 发送的数据报内容，其中红色方框内的内容表示 UDP 传送的数据报的头，主要包含数据的长度，源地址，目标地址，源端口，和目的端口等内容。例如：c0a8，0317 表示的就是源地址 192.168.3.23；07d0 为源端口号 2000。剩下的内容均为调试助手发送的字符串内容，当发送内容不足 18 个字节时，用 0 补齐。例如：这里发送的字符串内容为“hello”，对应 ASCII 码表十六进制为 68 65 6c 6c 6f。

第二种. 机器人做为 TCP 服务器。

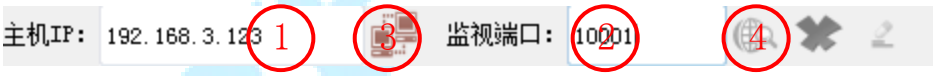
Step1. 调试助手如下配置，配置完以后需要点击【打开】按钮。目的 IP 和端口指的是一体机的 IP 和端口。



Step2. 一体机网络配置中配置侦听端口。选择 QC410 作为服务器下的任意序列号进行配置，例如这里选择序列号 0 进行示范。首先需要配置与调试助手里相同的端口号“10000”；然后，鼠标右击选择“状态”为“启动”；最后，点击同步 按钮保存生效。注意：是否设置“开机自启动”用户自由选择。



Step3. 打开终端界面，配置主机 IP 和监视端口号。之后点击连接和监视按钮。



Step4. 打开成功后界面如下：

再点击终止图标，通过键盘上的“←”，“→”，“↑”，“↓”可以调出 tcpdump –Aq 命令，进行修改为如下指令：

```
root@zynqlinux ~ # /opt/system/sbin/tcpdump -AqX \(dst host 192.168.3.123 and ds
t port 10001 and tcp \)
```

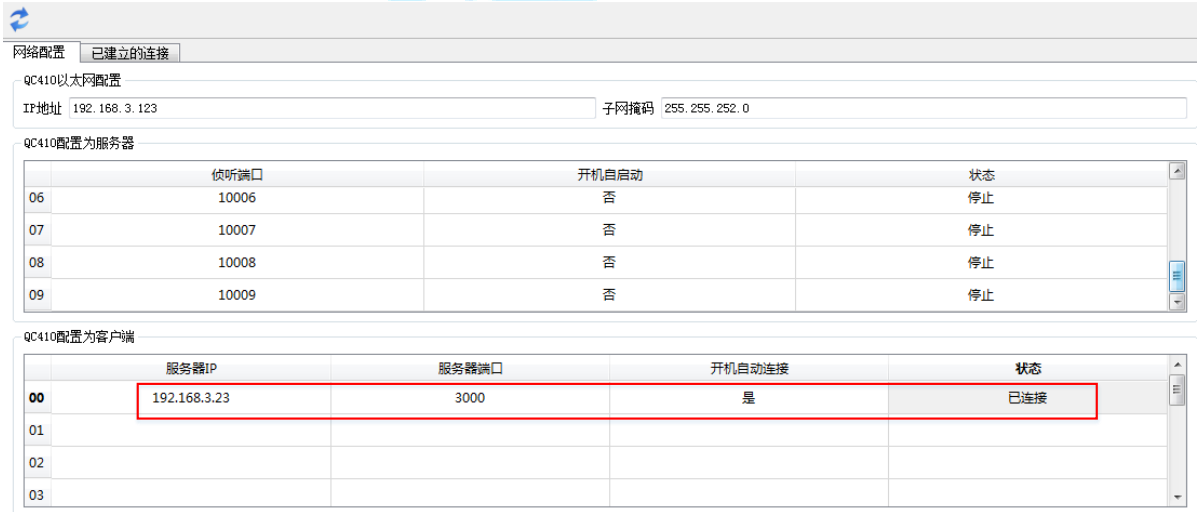
Step5. 全部设置完成后，调试助手就可以发送字符串了。在调试助手点击打开按钮建立连接的时候会自动传输两条空数据内容。如下图中，660561 和 662220 这两条就是在建立连接时发送的空数据。

```
root@zynqlinux ~ # /opt/system/sbin/tcpdump -AqX \(dst host 192.168.3.123 and ds
t port 10001 and tcp \)
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
19:01:23.660561 IP 192.168.3.23.2000 > 192.168.3.123.10001: tcp 0
        0x0000:  4500 0034 6cd9 4000 8006 0608 c0a8 0317  E..4l.@.........
        0x0010:  c0a8 037b 07d0 2711 9a38 d94e 0000 0000  ...{..'..8.N....
        0x0020:  8002 2000 24cb 0000 0204 05b4 0103 0302  ....$...........
        0x0030:  0101 0402                                ....
19:01:23.662220 IP 192.168.3.23.2000 > 192.168.3.123.10001: tcp 0
        0x0000:  4500 0028 6cda 4000 8006 0613 c0a8 0317  E..(l.@.........
        0x0010:  c0a8 037b 07d0 2711 9a38 d94f c35e 504a  ...{..'..8.O.^PJ
        0x0020:  5010 4029 31b6 0000 0000 0000 0000       P.@)1........
19:01:50.140424 IP 192.168.3.23.2000 > 192.168.3.123.10001: tcp 3
        0x0000:  4500 002b 77cd 4000 8006 fb1c c0a8 0317  E..+w.@.........
        0x0010:  c0a8 037b 07d0 2711 9a38 d94f c35e 504a  ...{..'..8.O.^PJ
        0x0020:  5018 4029 cd78 0000 3132 3300 0000       P.@).x..123...
```

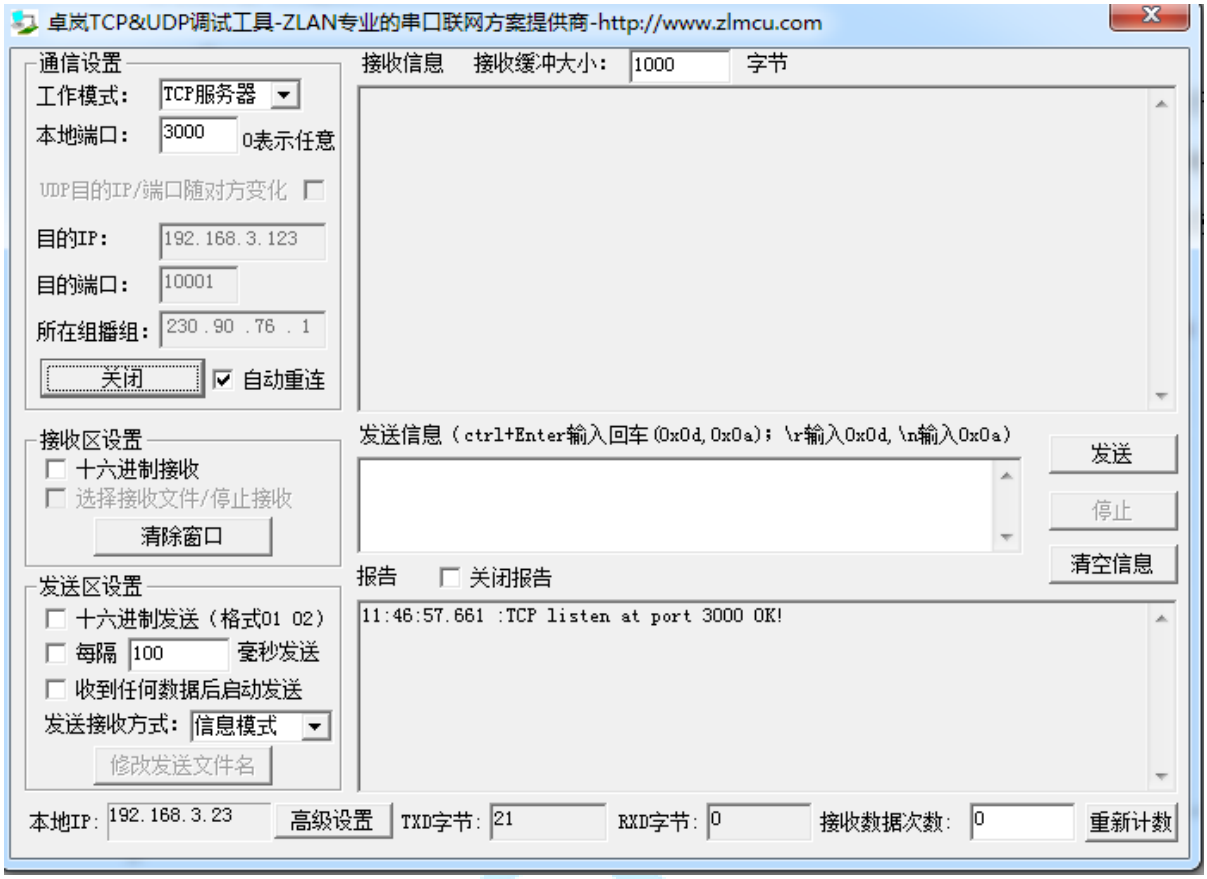
注：从红色框内 3132 的位置起打印调试助手发过来的字符串，至少为 6 个字节的长度，不足的用 0 补齐，超过 6 个字节长度的向后延伸。例如这里调试助手发送的内容为“123”，对应的 ASCII 码为 31 32 33。

第三种. 机器人做为 TCP 客户端，由于是客户端主动和服务器建立联系，因此需要先设置一体机这边的参数。

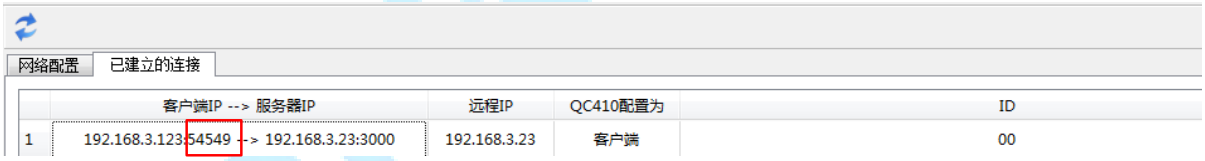
Step1. 打开工具菜单栏下的网络配置，配置成如下界面，配置完成后需要点击同步参数。



Step2. 调试助手如下配置，配置完以后需要点击【打开】按钮。这里只需要配置本地端口号即可。



Step3. 如下图，通过网络配置中确认当前连接一体机的端口号。



Step4. 打开终端界面，配置主机 IP 和监视端口号。之后点击连接和监视按钮。

Step5. 打开成功后界面如下：

再点击终止图标 ✖，通过键盘上的“←”，“→”，“↑”，“↓”可以调出 tcpdump –Aq 命令，进行修改为如下指令：

```
root@zynqlinux ~ # /opt/system/sbin/tcpdump -AqX \(dst host 192.168.3.123 and ds
t port 54549 and tcp \)
```

Step5. 全部设置完成后，调试助手就可以发送字符串了。

```
/opt/system/sbin/tcpdump -Aq \(dst host 192.168.3.123 and dst port 54549\) or \(s
src port 54549\)
^C
2 packets captured
4 packets received by filter
0 packets dropped by kernel
root@zynqlinux ~ # /opt/system/sbin/tcpdump -AqX \(dst host 192.168.3.123 and ds
t port 54549 and tcp \)
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
02:49:43.096371 IP 192.168.3.23.3000 > 192.168.3.123.54549: tcp 3
        0x0000:  4500 0037 4065 4000 8006 3279 c0a8 0317  E..7@e@...2y....
        0x0010:  c0a8 037b 0bb8 d515 8455 8eeb e7b5 ee57  ...{.....U.....W
        0x0020:  8018 0104 e575 0000 0101 080a 000d 747a  .....u........tz
        0x0030:  000b 6775 3131 31                        ..
```

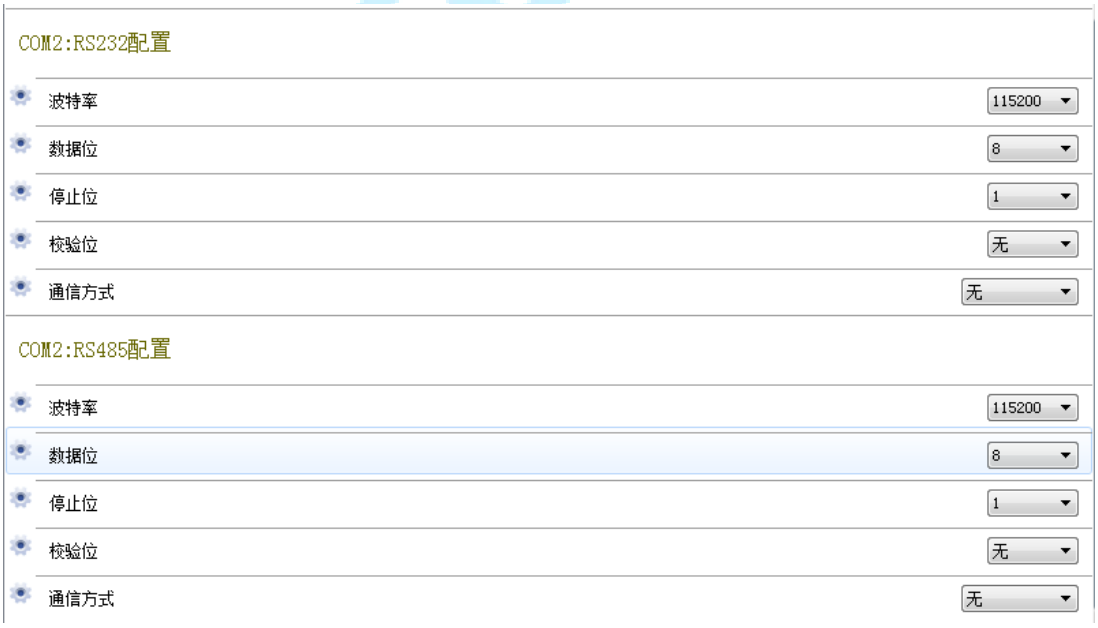
注：从红色框内 3131 的位置起打印调试助手发过来的字符串，例如这里调试助手发送的内容为“111”，对应的 ASCII 码为 31 31 31。

1.6.3 调试助手

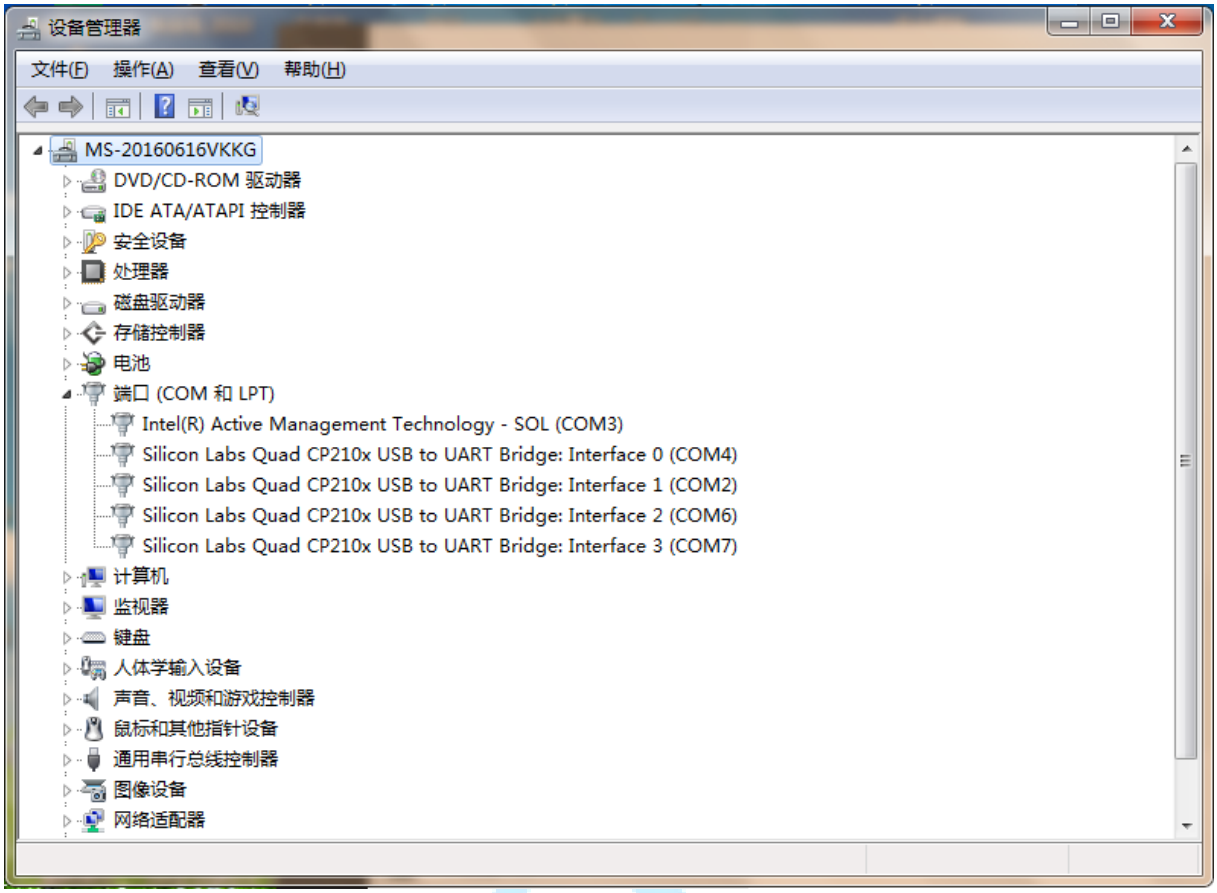
调试助手工具界面可以用来监控和模拟机器人与外部设备进行串口或网络通讯数据的接收和发送。

第一种. RS232/RS485 串口通信

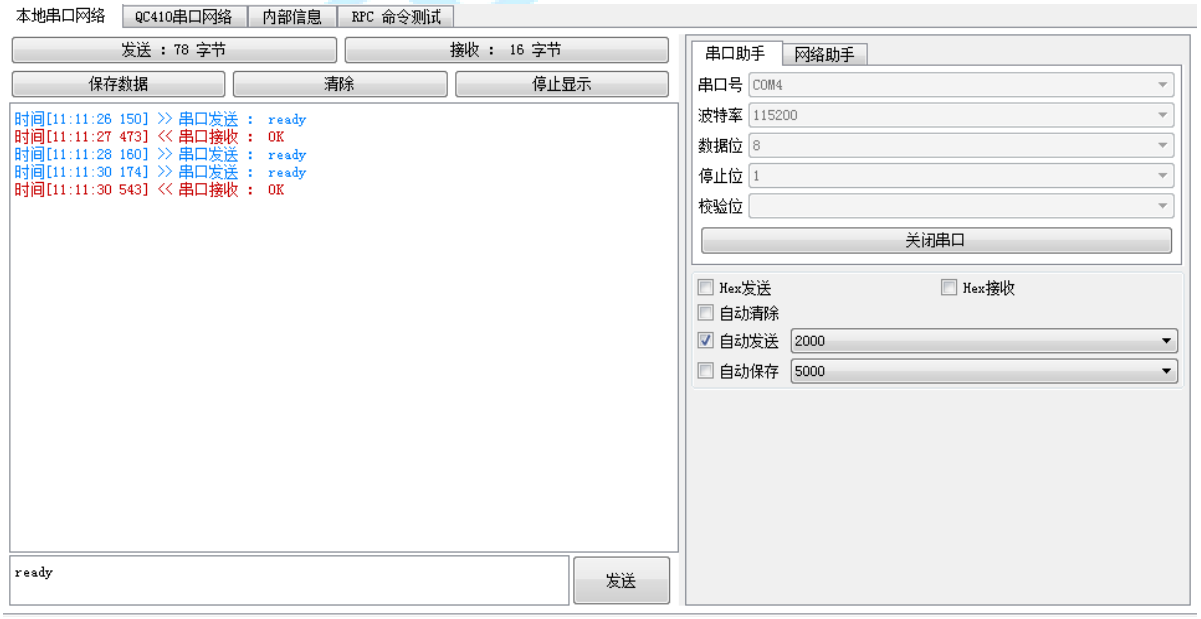
Step1. 首先需要将参数设置里的【COM2:RS232/RS485 配置】里的参数修改。



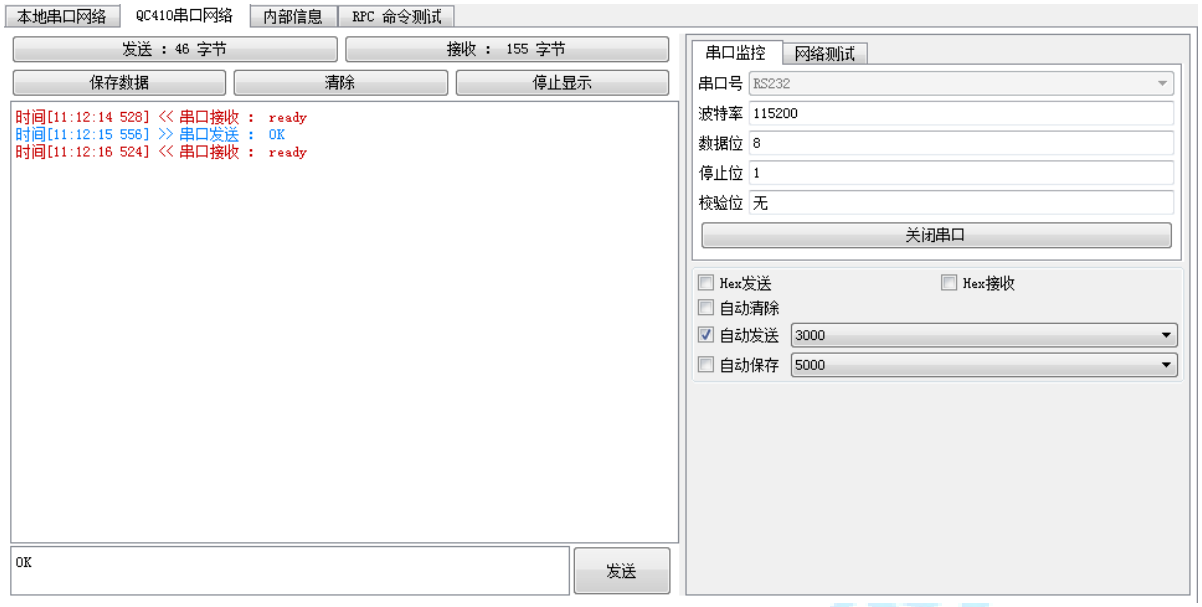
Step2. 查看设备管理器中的 Interface0 对应的端口，例如 COM4;



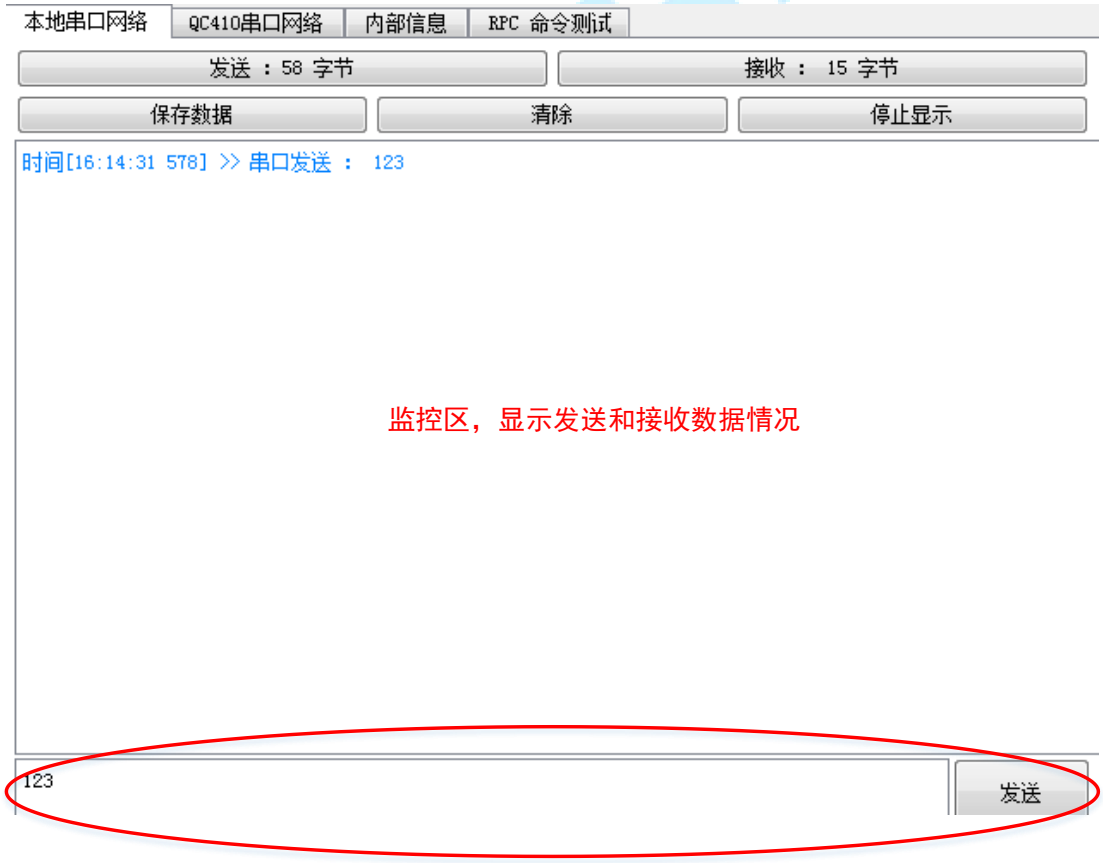
Step3. 打开【调试助手】设置本地串口网络，选择 COM4，设置波特率、数据位、停止位、校验位需要与 Step1 里设置的一致。设置完选择【打开串口】。



Step4. 设置 QC410 串口网络，选择【打开串口】。



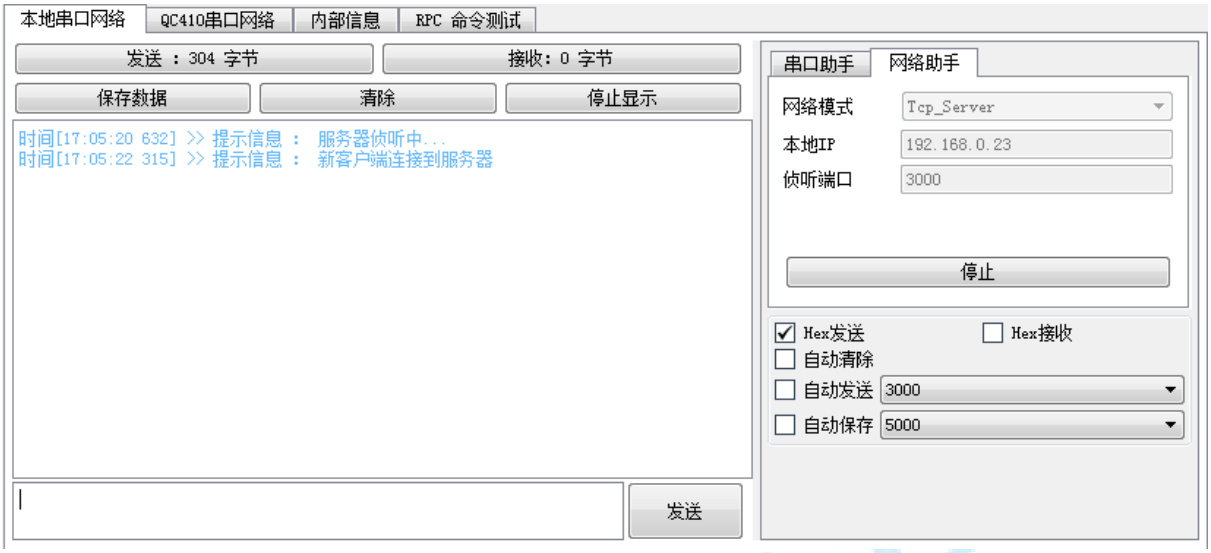
Step5. 以上全部设置完成后，就可以在发送区发送内容，另一端进行接收了。下图是以本地串口网络发送给 QC410 串口网络为例。同样的也可以从 QC410 发送给本体串口网络。



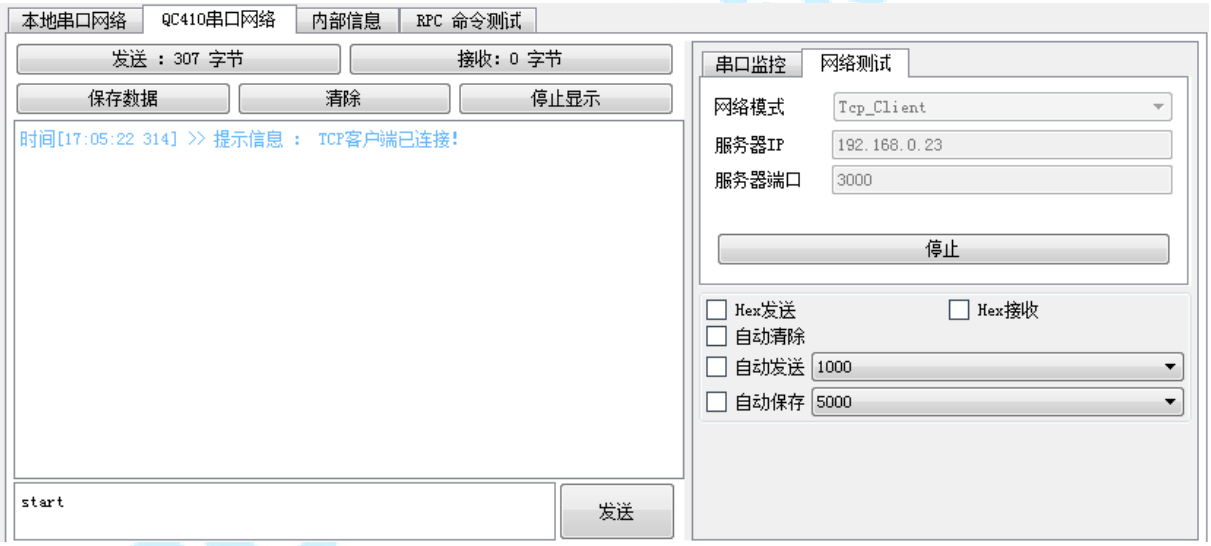
第二种. 网络通讯（一体机 IP 为 192.168.0.123；电脑 IP 为 192.168.0.23）

Step1. 打开【调试助手】设置本地串口网络下的网络测试，参数设置如下，设置完

选择【启动】。



Step2. 打开 QC410 串口网络下的【网络测试】，如下修改参数，设置完选择【启动】。



Step3 以上全部设置完成后，就可以在发送区发送内容，另一端进行接收了。这里和串口通讯发送内容方式一致。这里不再赘述。

1.6.4 输入输出

输入输出工具用于模拟仿真、监控机器人系统对应的 IO 状态。众为兴机器人 QC410 系统包含有 34 路输入，27 路输出，其中：

- **I_0~I_27（绿色标识）**分别对应外接输入板上的0~27输入；
- **I_28~I_33（蓝色标识）**分别对应重载连接器的16芯IO中的28~33输入；

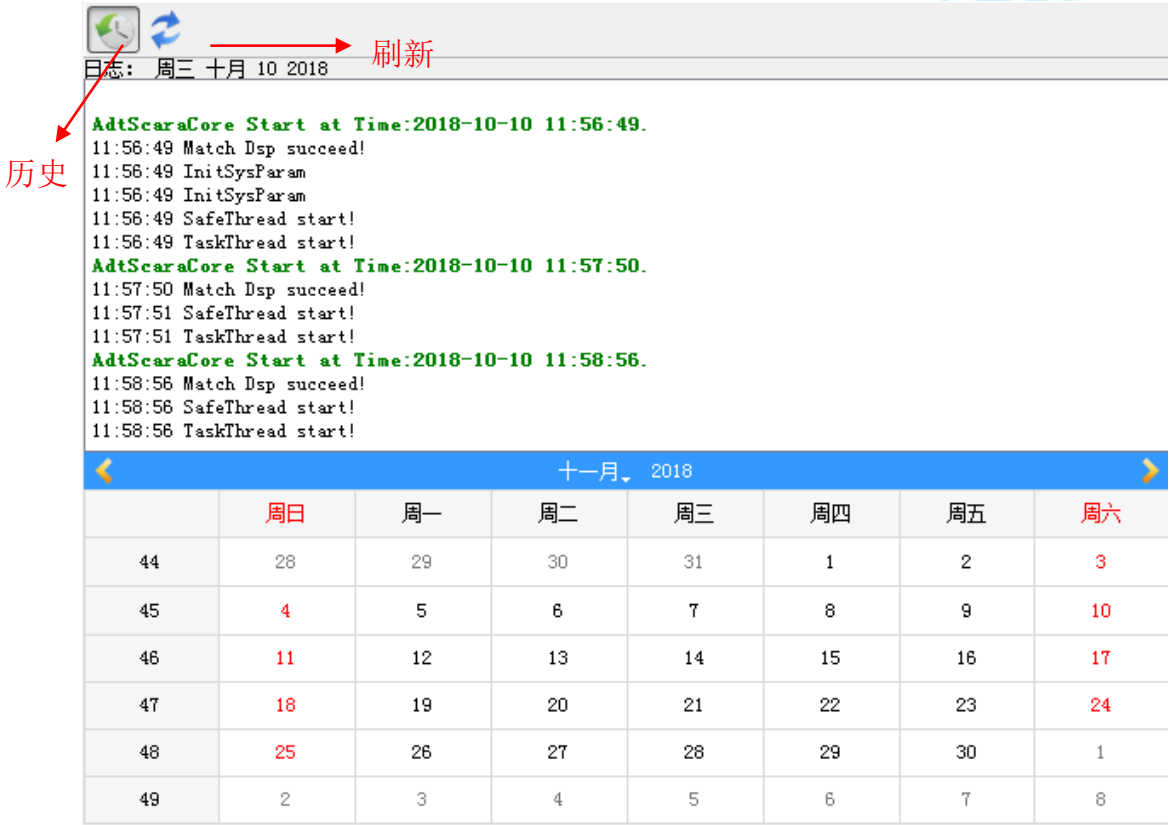
- **O_0~O_17（红色标识）**分别对应外接输出板上的0~17号输出；
- **R_9,R_18(红色标识)**分别对应两路继电器(SVST_A,SVST_B)和(EMSST_A,EMSST_B)；
- **O_19~O_22（紫色标识）**分别对应重载连接器的16芯中的19~22输出；
- **R_23~R_26（紫色标识）**分别对应重载连接器的16芯中的四路继电器(23~26)

> ✧ **小技巧：**I_0、I_1….以及 O_0、O_1…这些备注名字只是为了辅助用户记忆理解这些输入输出信号所代表的含义，在 DI 或 DO 或 WDO 或 WDI 这些指令中不能直接使用这些备注的名字，必须使用输入输出序列号 0、1、2、3…..
>
> ✧ **小技巧：**输入计数应用场景：当 QC410 与外部设备有信号的交互，但无法确定外部设备是否把输入信号给到机器人，可以通过给输入信号前后对应输入端口一侧的输入计数是否累加 1。若累加 1，则信号输入成功，否则失败。

> ✧ **小技巧**：当虚拟运行 AR 程序（没有连接实际的 IO 信号），AR 程序里面涉及到有输入信号等指令时，可点击 IO 打开 IO 仿真进行输入信号的模拟输入。

1.6.5 核心日志

在此界面可以查看一些核心日志的查看和追踪，例如：系统参数是否初始化、任务线程是否开启、安全任务是否开启、报警信息等。



1.6.6 网络配置

网络配置主要是用来配置一体机的 IP，子网掩码以及分别为服务器和客户端时的网络参数。

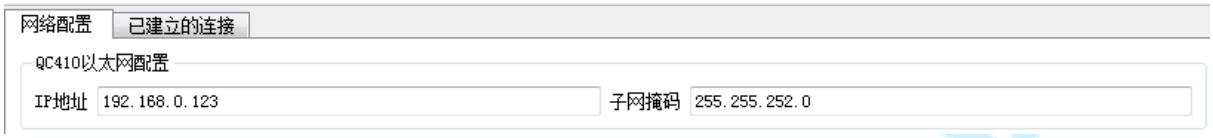
注：网络配置是针对机器人采用 TCP 通讯方式时，进行参数配置的界面。

- **IP 与子网掩码**

出厂时一体机的 IP 可能会被默认设置为 192.168.0.123，子网掩码为 255.255.252.0。但是如果在同一局域网下时，IP 地址都一样的情况下，可能会连接到别的一体机上，因此建议对此 IP 和子网掩码做修改。IP 可修改后两位，例如，这里修改后的为

192.168.3.123。

注意：如果子网掩码为 255.255.255.0；那么在同一网段内的 IP 可设置范围为：192.168.0.1-192.168.0.255；如果子网掩码为 255.255.252.0，那么在同一网段内的 IP 可设置为 192.168.0.1-192.168.0.255、192.168.1.1-192.168.1.255、192.168.2.1-192.168.2.255、192.168.3.1-192.168.3.255。

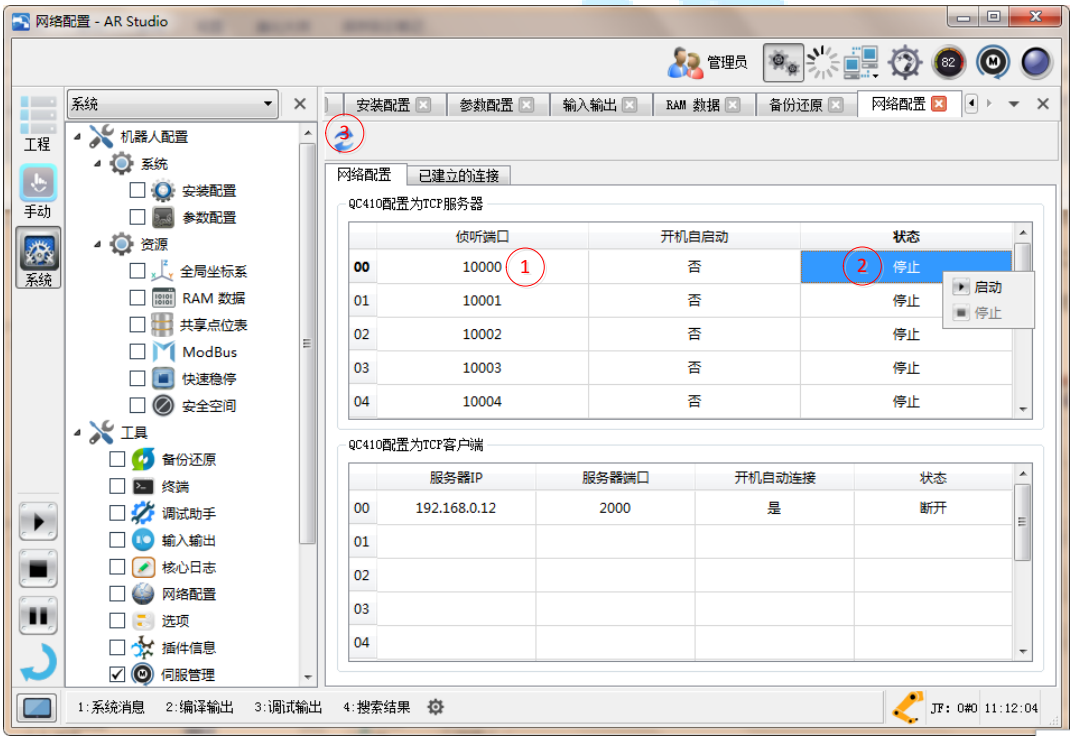


- QC410 配置为服务器

  a) 配置侦听端口（外部设备，例如 PLC 需要知道的机器人的端口）；

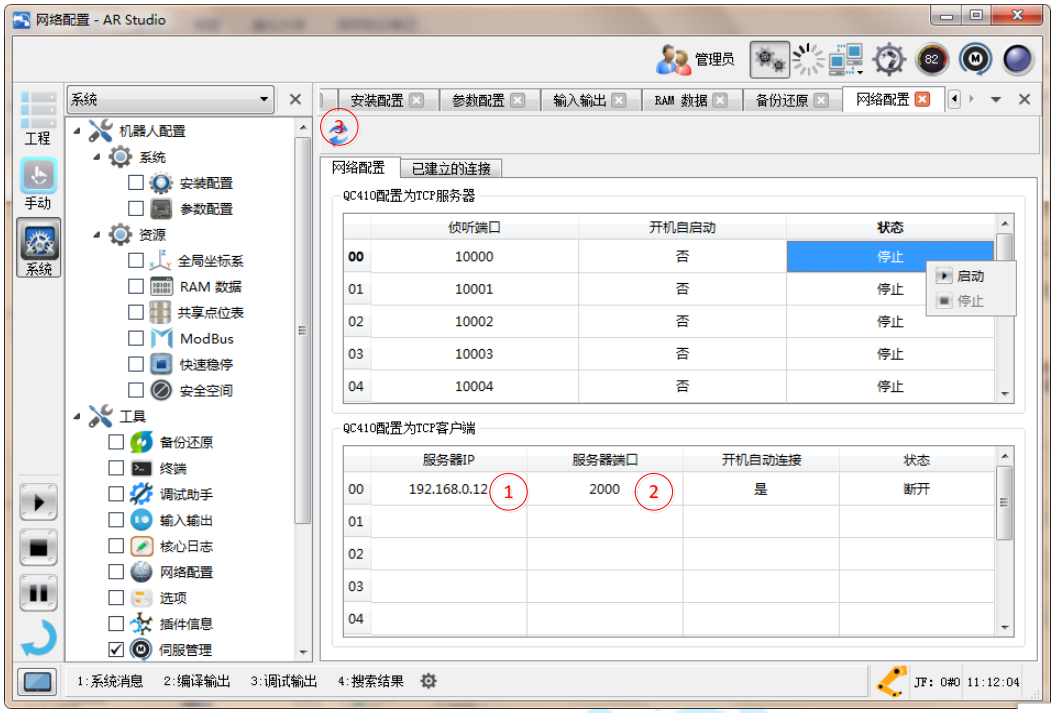
  b) 对应的“状态”设置为“启动”；

  c) 点击同步按钮 保存生效；



- QC410 配置为客户端

  QC410（机器人）作为客户端须主动和服务器建立联系联系，这里需要配置服务器的 IP 以及端口号。是否设置“开机自动连接”用户可根据需求自由设定。当客户端与服务器连接成功后，“状态”会显示为“已连接”。



1.6.7 软件配置

【软件配置】工具界面主要的功能是可以切换机器人系统显示的语言，目前只包含中英文两种语言，切换语言后须点击“ ”应用按钮生效，并重启控制器。辅助功能有:系统主题界面设置、输出和编辑器字体设置等。

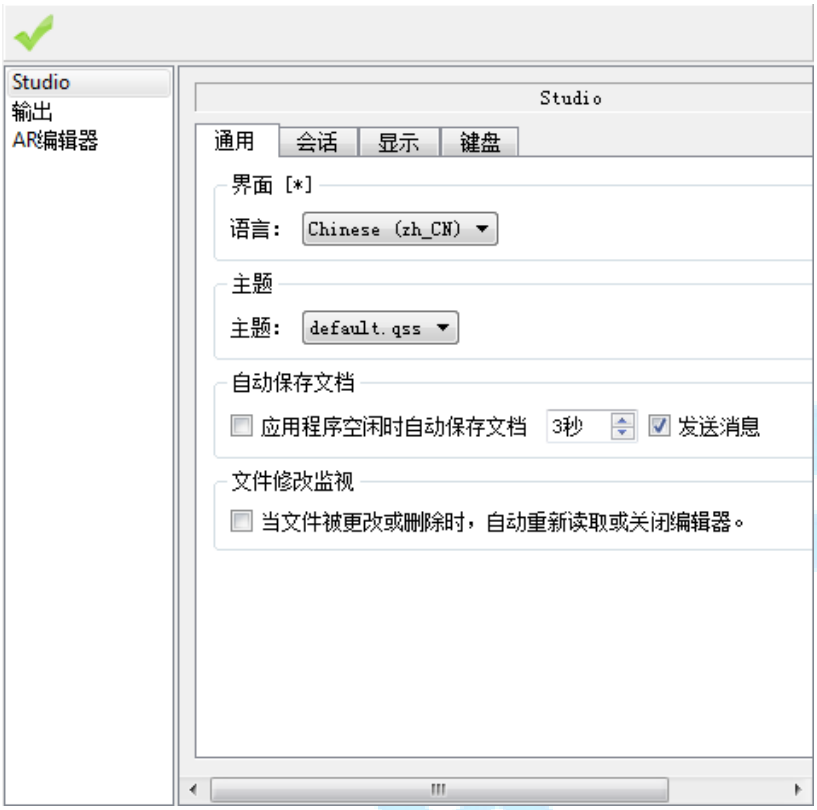


图 4.5.4 选项

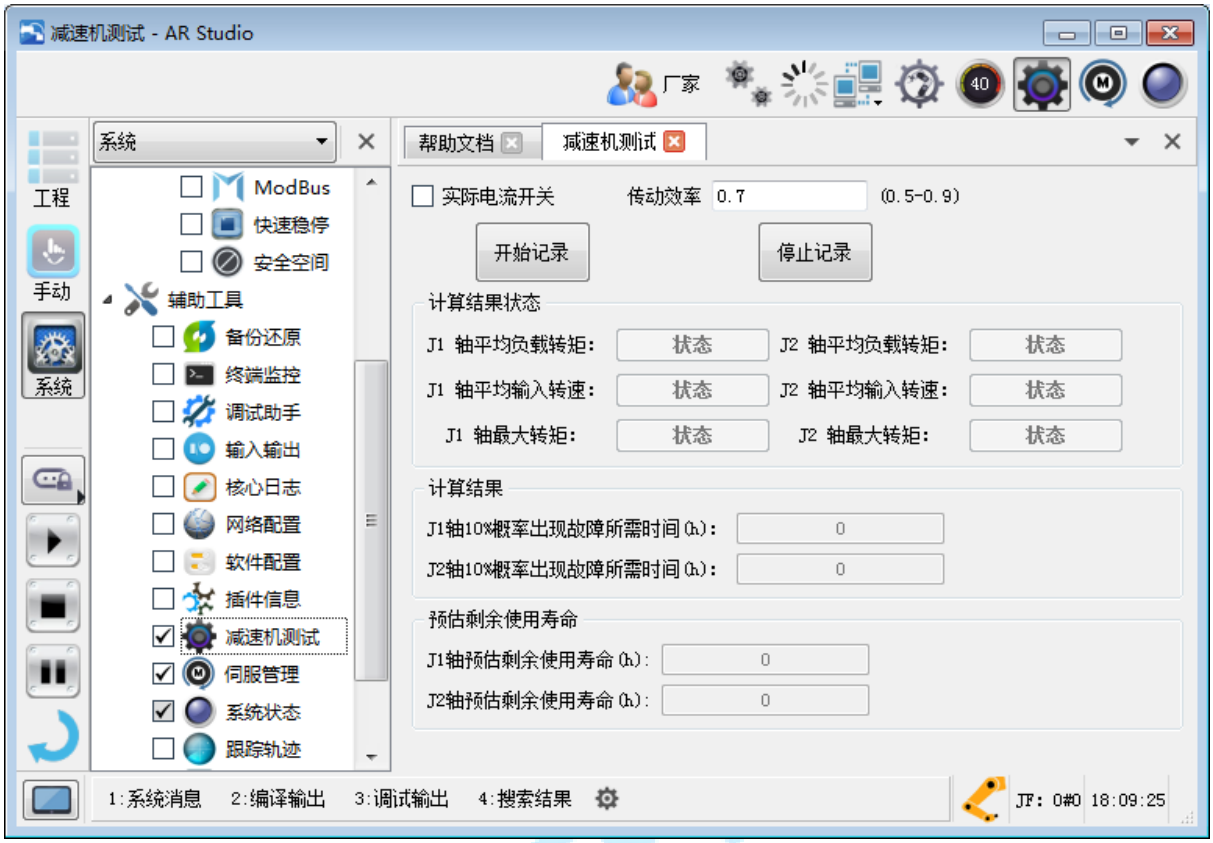
1.6.8 插件信息

插件信息主要给出 ARStudio 软件上的一些插件信息。可方便查看插件功能是否升级或是否存在问题。

名称	可用	详细信息	作者	版本	文件名
AdtRpc	☑	AdtRpc	gex	V1.00	adtrpc.dll
AR Builder	☑	Core AR Builder	gex	V1.00	arbuild.dll
AR Debug	☑	Core Debug Manager	stl,gex	V1.00	ardebug.dll
AR Editor	☑	Core AR Editor	adt	V1.00	areditor.dll
Find	☑	Core Find/Replace	adt	V1.00	arfind.dll
Backup&Restore	☑	Backup & Restore	gex	V1.00	backuprestore.dll
cfgcoord	☑	Cfg Coord	gex	V1.00	cfgcoord.dll
GlData Editor	☑	RAM Data	gex	V1.00	cfgram.dll
RobotProject	☑	Robot Project	gex	V1.00	robotproject.dll
ConveyorNode	☑	Conveyor Node	luoqi	V1.00	conveyornode.dll
CpuNode	☑	Cpu Node	gex	V1.00	cpunode.dll
DataNode	☑	Data Node	gex	V1.00	datanode.dll
GdbDebugger	☑	Core Gdb Debugger	stl	V1.00	gdbdebugger.dll
SysCfgParam	☑	System Config Param	gex	V1.00	syscfgparam.dll
MovePos	☑	Robot MovePanel	gex	V1.00	movepos.dll
NetworkNode	☑	Network Node	luoqi	V1.00	networknode.dll
PalletNode	☑	Pallet Node	gex	V1.00	palletnode.dll
SafeSpace	☑	Safespace Tool	gex	V1.00	safespace.dll
ServoState	☑	ServoState	stl,gex	V1.00	servostate.dll
ServoTunning	☑	Servo Tunning	timyear	V1.00	servoTunning.dll
SpeedRate	☑	SpeedRate	gex	V1.00	speedrate.dll
Track3d	☑	Track3d	gex	V1.00	track3d.dll
Standard User Ui	☑	Standard Ui	gex	V1.00	standarduserui.dll

1.6.9 减速机测试

【减速机测试】工具可用于预测 J1 轴（第一轴）和 J2 轴（第二轴）减速机的使用寿命。此操作须登录到厂家权限。



1.6.10 系统状态

系统状态指示灯用来查看机器人当前处于的状态，总共包含有四种状态，分别用四种颜色表示：

：机器人处于停止状态（蓝色）；

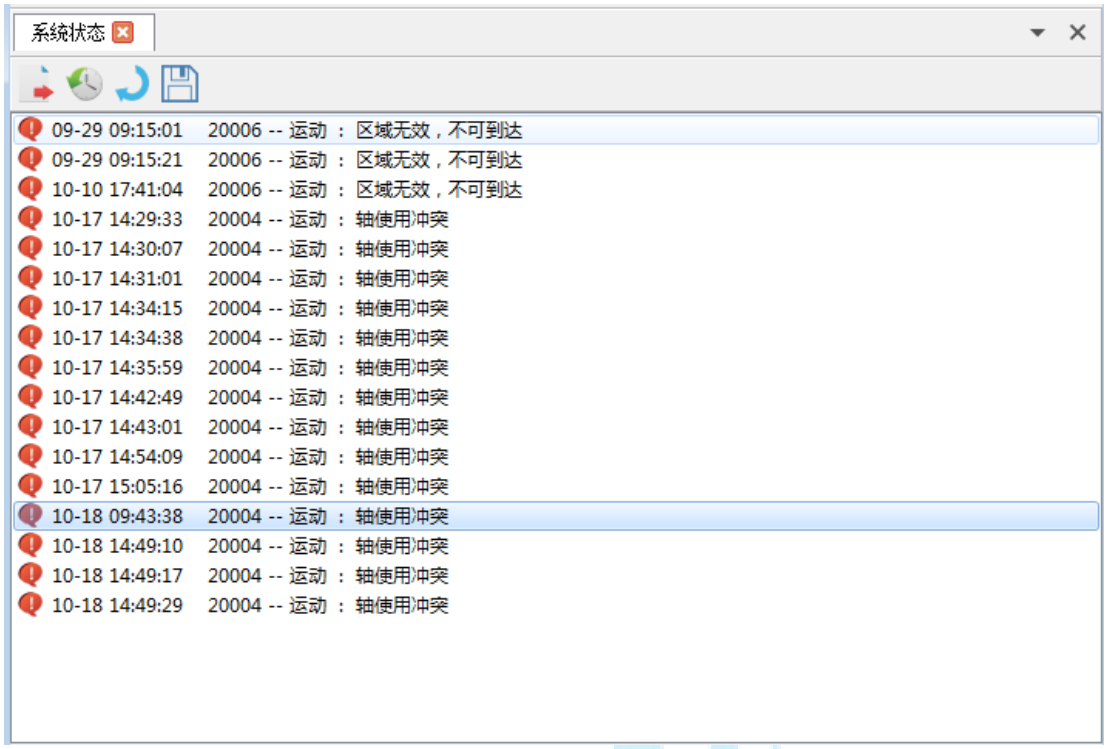
：机器人手动运行状态（深绿色）；

：机器人处于暂停状态（黄色）；

：机器人处于运行状态（绿色）；

：机器人系统有报警（红色）。

进入系统状态后可查看报警内容如下，其中 可以导出系统报警内容到指定路径， 可查看指定日期的报警历史记录， 为手动复位按钮，也可直接鼠标右键双击 进行复位， 为导出报警信息文件，可将报警前 100ms 四个轴的某些伺服参数（例如，参考位置/实际位置、参考速度/实际速度）导出用于报警分析。



1.6.11 跟踪轨迹

轨迹跟踪界面主要是对运行的程序进行轨迹仿真，在加工运行中，我们在此界面可以很直观的看到末端的运行轨迹情况，一目了然，方便实用。

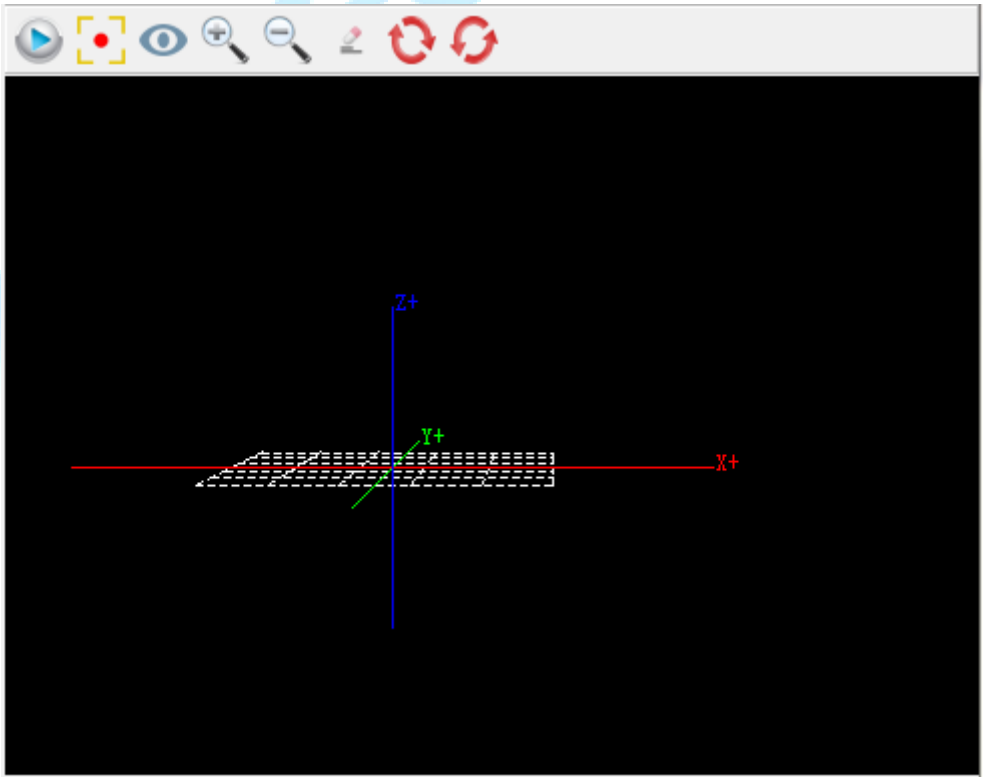


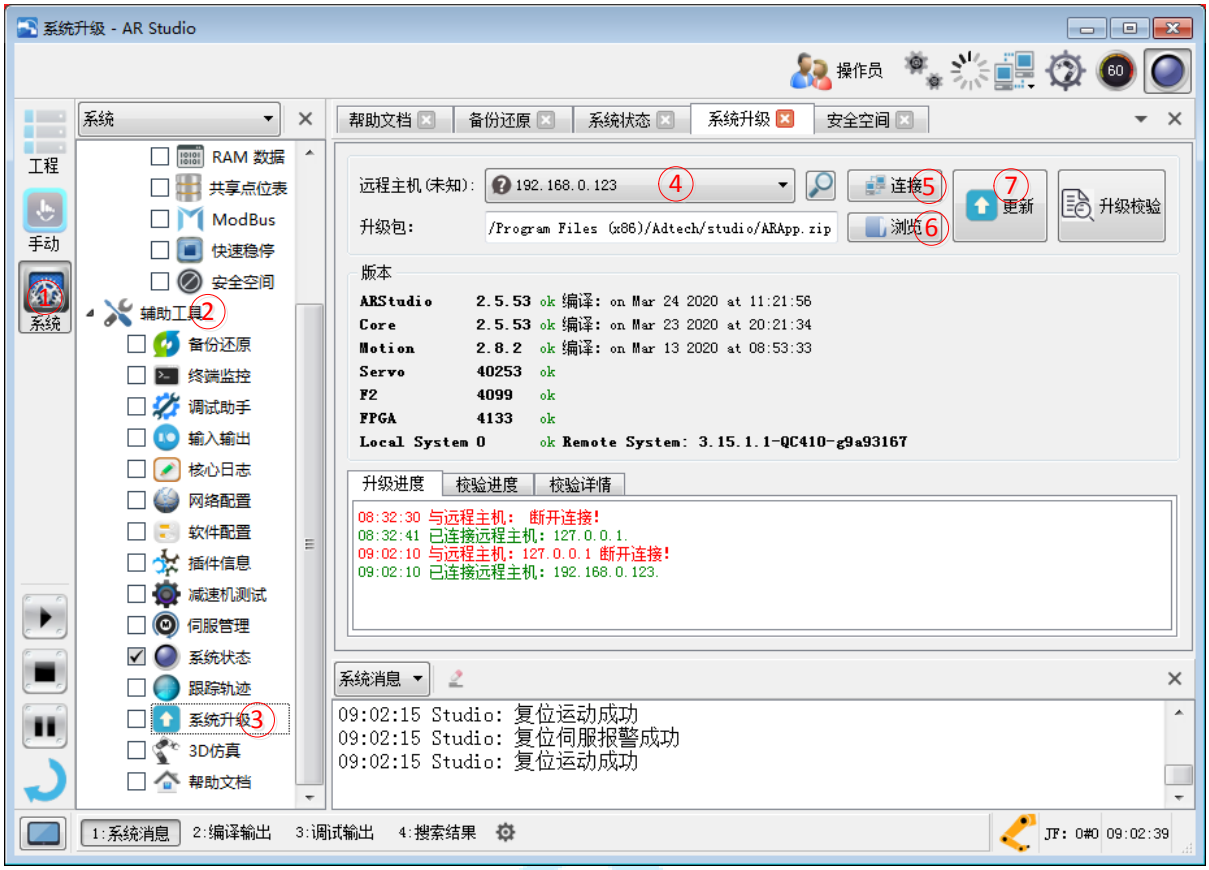
图 4.5.6 轨迹监控界面

表 4.5.6 轨迹跟踪界面图标说明

图标	名称	功能
	启动跟踪	启动轨迹跟踪功能
	画点\线切换	轨迹以点或线的方式呈现
	视角	切换观察轨迹的视角
	放大	放大轨迹呈现的大小
	缩小	缩小轨迹呈现的大小
	清除	清除轨迹
	顺时针旋转	轨迹绕选中的轴顺时针旋转
	逆时针旋转	轨迹绕选中的轴逆时针旋转

1.6.12 系统升级

【系统升级】功能用来升级系统应用程序，从而完善系统的功能以及性能。升级步骤大致如下图序号标识所示：



（1） 点击“系统”进入系统界面；

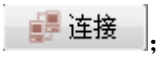
（2） 展开“辅助工具”；

（3） 鼠标左键双击“系统升级”选项，弹出【系统升级】界面；

（4） 在“远程主机（未知）”一栏扫描 IP 地址：

- 升级 QC410 控制器系统：这里的 IP 地址就是 QC410 控制器的 IP 地址（当开机后 ARStudio 与 QC410 网络连接成功时，这里的 IP 会自动设置成 QC410 的 IP），默认为：192.168.0.123；
- 升级 RPB07 示教器系统：这里的 IP 地址就是 RPB07 示教器的 IP 地址（一根网线连接 PC 机与 QC410 控制器，RPB07 示教器插入 QC410 的 TP 接口），默认为 192.168.2.110；

（5） 点击“远程主机”右侧的“连接”按钮，若连接成功，连接图标显示为蓝色

；若连接失败（需检查 IP 设置是否正确或者 PC 与 QC410 之间的网线是否断开），此图标为暗红色 

；

（6） 升级包默认的路径是 ARStudio 的安装路径下的/studio/ARApp.zip 中。若安

装包 ARApp.zip 在其它路径下，可通过“浏览”找到安装包的位置；

（7） 点击“更新”，则系统更新会自动完成，整个过程大约需要 3min 左右。升级的进度可在“升级进度”显示栏中显示。升级完成后会自动重启控制器。

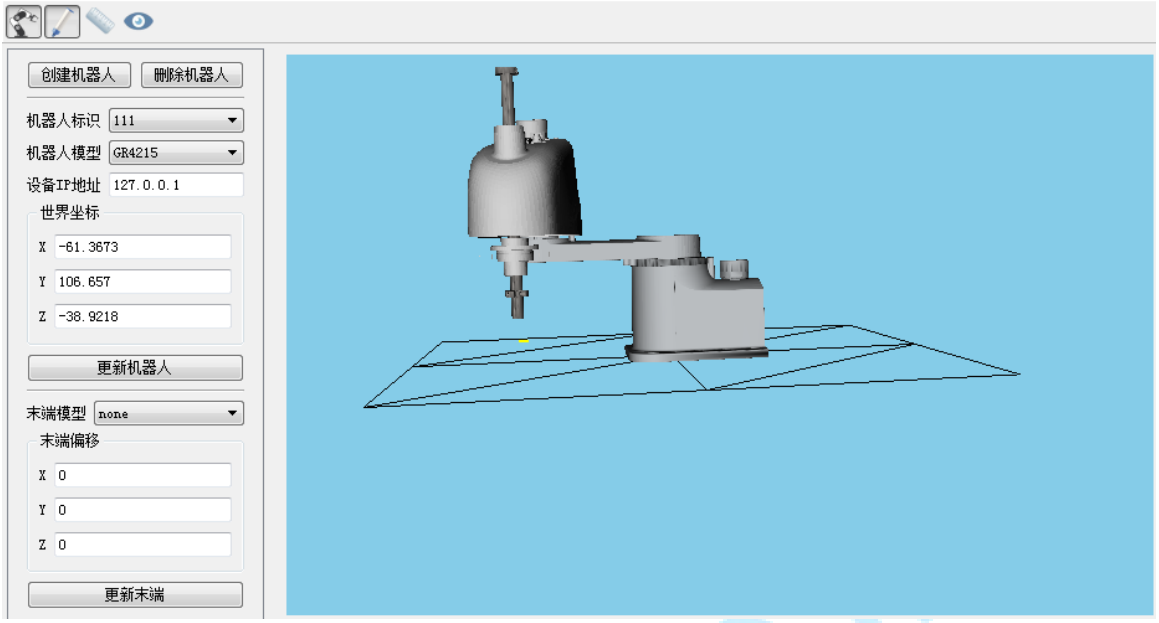
升级进度

```
06:51:19 Update servo.bin ...


********** enter ************
stop cpu1.
fileLen = 152056
********
BinSize:0x251f8
format complete, starting program...
program 0 sector Load:0x108000 Run:0x108000 Size:0x12520 Word
program 1 sector Load:0x147ffa Run:0x147ffa Size:0x2 Word
         100%
program complete

06:51:21 Update done, Reboot System ...

06:51:21 Update Progress normally finished.

14:49:29 与远程主机：192.168.0.123 断开连接！
```

> ✧ **小技巧：**程序更新前，一定要参考 1.4 章节建立 PC 机与 QC410 之间的连接或联系厂家；
>
> ✧ **小技巧：**点击“更新”会弹出“确定更新升级远程程序？”，点击“Yes”即可；

1.6.13 3D 仿真

虚拟机器人是可以实现三维空间内机器人的运动，可以通过鼠标切换视角、放大缩小、拖动机器人的位置。 分别表示“配置机器人、显示路径、最小距离、恢复视图”。



- 配置机器人

- 显示路径：显示机器人末端运动轨迹。
- 最小距离：夹具末端中心与机器人大臂、基座之间的最小距离；
- 恢复视图：恢复默认参数对应的视图。

1.6.14 帮助文档

上位机软件欢迎界面，可以进行快速的查阅一些帮助文档、创建工作空间、打开工作空间等快捷操作。

1.7 报警错误处理

QC410 驱控一体机在使用中因各种安全保护，有可能会出现报警现象，每一个报警都有相应的报警代码及相关故障内容。

在实际使用中，若出现报警，为避免造成不必要的损伤以及可能引起的安全隐患，我们应立即停止运行。对照故障相应的代码进行逐一排查，直到故障完全排查，才可继续运行。

1.7.1 伺服报警

错误 ID		
11003	：	编码器线未连接
	分析	伺服调试软件中编码器类型选择不对或电机编码器线连接错误。
	处理	● 检查调试软件中编码器类型选择是否正确。 ● 参考电气接线中的接线实例，检查编码器线接线。
11007	：	电机堵转
	分析	此报警可能的原因有：伺服上位机中堵转保护条件设置过于严格；电机带有刹车，刹车未打开；负载大，电机功率选型偏小；机械卡死。
	处理	连接伺服上位机调试软件，将堵转保护条件放大。如继续报警，检

		查机械结构上是否有卡住，如果机械上无异常，可能是电机功率不匹配。
11008	：	母线电压太高
	分析	母线电压不稳定
	处理	工厂昼夜用电负荷有差异，一般晚上母线电压会有所上升，检查伺服保护参数中的母线电压设置是否正确。
11009	：	母线电压太低
	分析	母线电压不稳定
	处理	机器人带载或高速运行，电流升高，母线电压会有所下降，可能报警，此时检查伺服保护参数中“允许的最小母线电压”，将其修改为 180V。如仍报警，请联系厂家。
11013	：	A 相电流过高
11014	：	B 相电流过高
11015	：	C 相电流过高
	分析	三相电流超过保护范围
	处理	● 一体机断电，量电机三相电阻是否平衡； ● 电机正常情况下，检查电机动力线接线是否有误。 ● 动力线接线无误的情况下，降低机械速度或减小负载，看报警是否有还存在，如不报警，表示机械负载太重，或运行速度超过电机最大转速。
11016	：	电机电流过高
	分析	电机实际电流超过保护范围
	处理	● 一体机断电，量电机三相电阻是否平衡； ● 检查动力线接线是否正确。
11018	：	IPM 模块温度太高
	分析	IPM 模块温度高于 65° 报警
	处理	出现该报警，预示着 IPM 模块温度过高，导致出现该报警的原因主要有以下 2 个方面： ● 电机功率选取太大，超过驱控一体机总功率 3KW 时，机器人高

		速运行，会出现模块温度过高现象； ● 刹车电阻打开时序异常导致温度过高，此时需联系厂家处理。
11019	：	速度过高
	分析	电机速度超过伺服保护参数中的“最大允许速度”
	处理	● 检查伺服保护参数中的最大允许速度是否合理（一般设置为电机最高转速）； ● 参数设置正常的情况下，出现该报警预示着电机即将飞车或已飞车。可能为动力线接触不良引起，检查接线。
11020	：	位置偏差超限
	分析	位置偏差超过保护中的限制值
	处理	● 检查伺服保护参数中的位置偏差限制值是否过小（应设定为实际位置偏差的 5-10 倍以上）。 ● 位置环增益设置不当，此时在保证机械不抖的情况下，适当增大位置环增益。 ● 加减速设置不合理，降低加减速。
11021	：	速度偏差超限
	分析	速度偏差超过保护中的限制值
	处理	● 检查动力线接线是否正确。 ● 检查伺服保护参数中的速度偏差限制值是否过小（应设定为实际速度偏差的 5-10 倍以上）。 ● 速度环参数设置不当，在保证机械无异响的情况下，适当增大速度环增益。 ● 检查动力线屏蔽线是否连接可靠，检查带有刹车的电机，是否接有抱闸滤波板。
11023	：	母线电压太高，超过了绝对限制值
	分析	母线电压超过伺服保护参数中的最大允许的绝对母线电压
	处理	● 检查保护参数中的“最大允许的绝对母线电压”设置是否合理，默认情况下为 380V； ● 监测一体机上电（机械不运行情况下），母线电压是否为正常值，一般情况下上电母线电压约为 320V；如此值偏高，可适当放大最大允许的绝对母线电压；

		● 如遇硬件有问题，请联系厂家。
11027	：	IPM 模块报警
	分析	IPM 模块报警被触发
	处理	● 一体机断电，量电机三相电阻是否平衡； ● 电机正常情况下，上电，轻微推动轴，看是否有卡顿现象，如有，模块损坏或动力线接触不良，检查动力线，检查模块； ● 都正常情况下，检查是否是外部因素（干扰、示教点位不正确，导致电机飞车或撞机）引起报警。
11028	：	选择的编码器类型不支持
	分析	编码器类型不正确
	处理	检查伺服上位机软件中选择的编码器类型是否正确，与厂家确认是否支持此类型的编码器。
11035	：	驱动器供电电源断开
	分析	供电电源检测异常
	处理	● 一体机上电之后，听机柜内的继电器是否有吸合声； ● 检查 220V 供电是否有波动或异常。 **注：**可接受 200V 至 240V 的电压。
11036	：	电机参考电流过高
	分析	电机给定电流超过峰值电流限制值
	处理	● 检查动力线接线是否正确。 ● 如果降低速度情况下，电机正常运行。说明负载太重或选择的电机不匹配，需降低速度运行或更换更大功率电机。
11037	：	电机 1.2 倍过载
11038	：	电机 1.5 倍过载
11039	：	电机 2 倍过载
11040	：	电机 2.5 倍过载
11041	：	电机 3 倍过载
	分析	● 电机 1.2 倍过载：电机电流超过持续电流限制的 1.2 倍且持续时间超过 10min；

		● 电机 1.5 倍过载：电机电流超过持续电流限制的 1.5 倍且持续时间超过 1min； ● 电机 2 倍过载：电机电流超过持续电流限制的 2 倍且持续时间超过 20s； ● 电机 2.5 倍过载：电机电流超过持续电流限制的 2.5 倍且持续时间超过 5s； ● 电机 3 倍过载：电机电流超过持续电流限制的 3 倍且持续时间超过 1s；
	处理	● 如果降低速度情况下，电机正常运行。说明负载太重或选择的电机不匹配，加减速设置不当，需降低速度运行或更换更大功率电机。
11042	：	电池错误报警
	分析	电机多圈信息错误报警
	处理	● 安装绝对式编码器用电池，机子第一次启动时，需清除电池错误报警（手持盒上有清除电池错误报警界面） ● 非第一次安装使用出现该报警，需特别注意该轴零点已丢失（可能是电池接线异常或者电池没电，未及时更换电池，导致多圈数据丢失，需清除报警并重新清零）。
11043	：	编码器数据 CRC 校验错误
	分析	编码器数据异常
	处理	检查编码器线是否接错，检查编码器屏蔽线是否漏接。
11051	：	电流采样信号异常
	分析	驱控一体内部电流采样信号异常
	处理	断电检查控制器内部软排线，如无法清除报警请联系厂家。
11057	：	脉冲输入速度过大
	分析	控制器下发速度超过保护范围
	处理	● 检查伺服保护参数中的最大允许速度是否设置合理； ● 检查控制器下发的脉冲是否异常。
11058	：	FPGA 与 DSP 之间总线异常

	分析	FPGA 与 DSP 之间数据总线或地址总线异常
	处理	请联系厂家
11059	：	脉冲输入加速度过大
	分析	控制器所发脉冲过大
	处理	检查手持盒里设置的加减速是否在合理范围内，电机最高速度为 5000r/min，加速时间最短为 10ms，对应 1ms 控制器所发脉冲应控制在 1000P 以内
11060	：	动力线 U 相缺相
11061	：	动力线 V 相缺相
11062	：	动力线 W 相缺相
	分析	动力线接触不良；
	处理	检查动力线接线
11063	：	控制器发脉冲校验错误
	分析	伺服与控制器间收发数据异常
	处理	请联系厂家
11064	：	电源信号异常
	分析	驱控一体上电之后电源无法完成充电过程
	处理	检查控制器内部 J11 端子是否松动，若无法清除报警请联系厂家。
11065	：	NTC 热敏电阻未连接
	分析	驱控一体驱动板上热敏电阻异常
	处理	● 检查热敏电阻是否出现未连接或接触不良现象； ● 检查热敏电阻是否出现损坏或短路现象。
11066	：	刹车电阻温度异常
	分析	刹车电阻温度高于 70° 报警
	处理	出现该报警，与 IPM 模块温度过高报警类似（一体机中 NTC 电阻贴在散热器端，散热器底部放置 IPM 模块，散热器侧面固定刹车电阻），导致出现该报警的原因主要有以下 2 个方面：

		● 电机功率选取太大，超过驱控一体机总功率 3KW 时，机器人高速运行，会出现刹车电阻温度异常现象； ● 刹车电阻打开时序异常导致温度过高，此时需联系厂家处理；
19999	：	编码器通信错误
	分析	编码器异常
	处理	检查编码器线接线是否正确，编码器屏蔽线接线是否可靠。
19998	：	电机编码器 BISS 协议通信错误
	分析	编码器异常
	处理	● 检查编码器接线是否正确，编码器屏蔽线接线是否可靠。 ● 此类型编码器，电池低压，报警未清除的情况下，也会报此警。需更换电池。更换电池之后，通过命令方式清除报警。需重新标定零点位置。
11090	：	电机编码器电池电量低
	分析	编码器侧电池异常
	处理	检查电池是否低压，如有低压，更换电池。更换电池要控制器上电的时候更换，这样可以避免电池完全拔掉的时候，导致零点位置丢失。如电池无低压，检查电池接线是否松动。

1.7.2 运动 DSP 报警

错误 ID		
20001	：	参数错误
	分析	AR 指令传入参数出现错误（例如只允许输入正值，实际输入的是负值，或者漏传参数，多传参数情况
	处理	检查 AR 指令传入参数是否存在问题。如果仍然报警，请联系厂家！
20003	：	运动冲突
	分析	底层运动产生冲突
	处理	请联系厂家!
20004	：	轴使用冲突

	分析	在前一段轨迹的状态还没有结束，就开始走下一段轨迹，造成轴使用冲突
	处理	请联系厂家！
20005	：	轴不存在
	分析	当前将要规划运动的轴不存在！
	处理	检查编程参数是否正确，如仍然报警，请联系厂家。
20006	：	区域无效，不可到达
	分析	运动目标位置不在机器人的合理工作范围内，也可能是拱形运动的 z 最大高度超出限位，也可能是一些位置是属于奇异点位置，在走直线的时候会失败从而报该错误。
	处理	● 确定目标位置是否机器人行程之外的，或者在奇异点位置的。 ● 如果位置数据是从其他地方导入的，需要检查是否相同臂长的机器手臂。
20007	：	伺服校验错误
	分析	与伺服的通讯出现问题，系统无法正常工作！
	处理	请联系厂家！
20008	：	急停
	分析	急停按钮被按下，或者急停端子没有正确接线，或者急停 IO 配置错误
	处理	检查急停按键是否被按下，检查急停端子是否正确连接。
20009	：	奇异点区域不可做轨迹运动
	分析	奇异点是指机械本体存在干涉的运动区域。这个警告一般是运动之前检查当前点的位置是否合法而产生的，通常情况下是由于要**走直线运动，并且当前点处于机械手臂的零界点或干涉点**。
	处理	将运动指令改为不要直线运行，改走点位运动或拱形移动。 将当前点手动移出零界点（手臂全部在一条线上，代表零界点），然后再执行运动指令。
20010	：	手系不同不可做轨迹运动

	分析	机器人在进行笛卡尔运动（直线或圆弧等）过程中，目标点的手系与当前点的手系不一致。
	处理	修改当前手系，或目标手系，必须保证在同一个手系内做运动。
20011	：	钻孔攻丝运动冲突
	分析	切换到钻孔攻丝运动时，还存在其他运动
	处理	请联系厂家！
20013	：	插补队列满，请等待
20014	：	事件队列满，请等待
	分析	插补队列或事件队列已满。
	处理	请联系厂家！
20015	：	动态运动时间超限
	分析	机器人在跟随过程中，实际运动的时间比设定的最大工艺时间大。
	处理	缩短机器人跟随过程中的运动时间或设大最大工艺时间（工艺时间：即进入跟随后开始，到跟随结束的总时。
20016	：	未定义指令
	分析	Motion(运动 DSP)接收到未定义的指令
	处理	请联系厂家！
20017	：	伺服报警
	分析	伺服产生了报警，请注意同时出现的伺服详细报警。这个报警跟伺服报警是一起产生的
	处理	查看机器人的报警状态，根据具体的伺服报警进行处理。
20019	：	运动中止
	分析	自锁保护报警，需要配合其它同一时间的报警信息来看。
	处理	根据其它报警信息来判断原因。
20021	：	J1 轴软限位
20022	：	J2 轴软限位
20023	：	J3 轴软限位

20024	：	J4 轴软限位
	分析	J1、J2、J3、J4 轴的实际位置超出了机器人本身设定的正负限位。
	处理	先找到机器人出现软限位的原因，然后再手动拖动机器人对应报警轴到软限位以内。
20026	：	编码器通讯错误
	分析	M5，M6 两路外部编码器出现通讯异常。
	处理	确保外部编码器可以正常使用，检查是否接线牢靠和屏蔽是否良好以及 5v 电压是否稳定。
20027	：	BISS 协议出错
	分析	使用 SSI 模式的外部编码器（M5 或 M6）出现 BISS 协议出错。
	处理	● 检查编码器接线是否正确，编码器屏蔽线接线是否可靠； ● 检查所用外接编码器是否是磁编码器； ● 检查 5v 电压是否稳定。此类型编码器，电池低压，报警未清除的情况下，也会报此警。需更换电池，更换电池之后，通过短接方式可清除报警。
20030	：	指令冲突
	分析	在严重报警情况下仍然下发指令，造成指令冲突。
	处理	请联系厂家！
20031	：	动态坐标系不可做关节运动
	分析	在动态跟随运动中，进入同步之后调用非 MovL 的指令，造成报警。
	处理	动态跟随的同步过程中，只允许直线插补运动（MovL），不允许 ptp 运动。
20032	：	J1 轴加速度超限
20033	：	J2 轴加速度超限
20034	：	J3 轴加速度超限
20035	：	J4 轴加速度超限
	分析	在走直线插补运动中，J1、J2、J3、J4 轴加速度超出设定值。
	处理	找到插补运动过程中，加速度超限的原因，检查相关运动参数设定

		是否合理，检查当前姿态是否接近奇异点，导致某一关节运动超限。
20036	：	J1 轴速度超限
20037	：	J2 轴速度超限
20038	：	J3 轴速度超限
20039	：	J4 轴速度超限
	分析	在走直线插补运动中，J1、J2、J3、J4 轴速度超出设定值。
	处理	找到插补运动过程中，速度超限的原因，检查相关运动参数设定是否合理，检查当前姿态是否接近奇异点，导致某一关节运动超限。
20042	：	运动使能冲突
	分析	上使能时候，上一运动的状态未清除。
	处理	请联系厂家！
20044	：	使能运动冲突
	分析	上使能时候，发现有其它轴已经上了使能。
	处理	请联系厂家！
20045	：	倍率异常错误
	分析	倍率出现大于 100%或小于 0 的情况。
	处理	请联系厂家！
20047	：	碰撞报警
	分析	在启用碰撞功能后，发生了外力与机器人碰撞。
	处理	检查机器人是否遇到障碍，发生了碰撞。
20048	：	激励轨迹运行出错
	分析	激励轨迹运行出错。
	处理	请联系厂家！
20050	：	J1 轴单周期脉冲增量异常
20051	：	J2 轴单周期脉冲增量异常
20052	：	J3 轴单周期脉冲增量异常

20053	：	J4 轴单周期脉冲增量异常
	分析	机器人下发脉冲发生异常，可能电机加速度设置过大。
	处理	检查电机加速度是否设置过大，若正常请记录操作内容，保留现场及时联系厂家！
20054	：	J1 单周期脉冲输入异常
20055	：	J2 单周期脉冲输入异常
20056	：	J3 单周期脉冲输入异常
20057	：	J4 单周期脉冲输入异常
	分析	机器人下发脉冲发生异常，可能电机速度设置过大
	处理	检查电机速度是否设置过大，若正常请记录操作内容，保留现场及时联系厂家！
20064	：	急停！双回路可能有一路失效
	分析	急停！双回路可能有一路失效。
	处理	检查急停端子是否正确连接，需连接两路。
20065	：	伺服未准备好，5 秒后再试！
	分析	伺服未准备好。
	处理	伺服未准备好，5 秒后再试！
20066	：	使能状态禁止参数修改
	分析	使能状态禁止参数修改。
	处理	不要在使能情况下修改参数！
20067	：	使能失败，电机被伺服占用，需伺服释放控制
	分析	电机被伺服占用，需伺服释放控制。
	处理	可能单轴上了使能，需要通过伺服软件单独操作各轴的使能，确保各轴下掉使能。
20068	：	电机上/下使能超时失败
	分析	电机上/下使能超时失败。
	处理	伺服异常，检查手动上/下使能是否正常。

20069	：	电机使能状态异常改变，可能被伺服软件控制
	分析	电机使能状态异常改变，可能被伺服软件控制。
	处理	检查是否有打开伺服软件来操控伺服电机。
20070	：	与伺服数据同步异常
	分析	与伺服数据同步异常。
	处理	请联系厂家！
20071	：	飞拍过程错误
	分析	飞拍过程错误，可能原因：机器人没有经过飞拍点的设定区域内；位置锁存异常；高速输入 IO 端口选择错误；飞拍点不可到达；没有示教飞拍点。
	处理	请检查是否：1.是否示教了飞拍点；2.机器人实际运动轨迹是否经过了飞拍点的设定区域内；3.指令设定高速 IO 端口等参数是否正确；4.飞拍点机器人是否可到达。
20073	：	超出工作空间限位
	分析	机器人当前的笛卡尔坐标超出了**工作空间：限位**中设置的工作范围
	处理	1. 查看工作空间：限位中设置的工作范围；2. 手动推动机器人使机器人末端处于工作空间：限位设定的范围内；3. 按“复位”即可清除该报警

1.7.3 系统报警

错误 ID		
81016	：	FRAM 初始化失败
	分析	铁电初始化失败
	处理	请联系厂家！

81017	：	FRAM 校验失败
	分析	铁电校验失败
	处理	请联系厂家！
81018	：	AR Error 指令报警
	分析	当前运行的 AR 程序中有 Error 指令被执行
	处理	检查 AR 程序
81019	：	软件版本依赖出错
	分析	检测到 QC410 控制器的某些模块的软件版本不匹配
	处理	出现此报警，需检查升级包的一致性或联系厂家索要成套的升级包 ARApp.zip 重新升级

1.8 附录：修改记录

修改时间	修改内容
2020.3.9	1. 更新 1.7.2 章节的运动 DSP 报警； 2. 更新 1.7.3 章节的系统报警；
2020.3.23	1. 更新 1.5.6 章节安全空间； 2. 新增 20073 报警；

第二章电气连接篇

2.1 系统介绍与功能概述

2.1.1 系统组成

驱控一体机系统主要包括以下三个部分：

1）QC410 驱控一体机

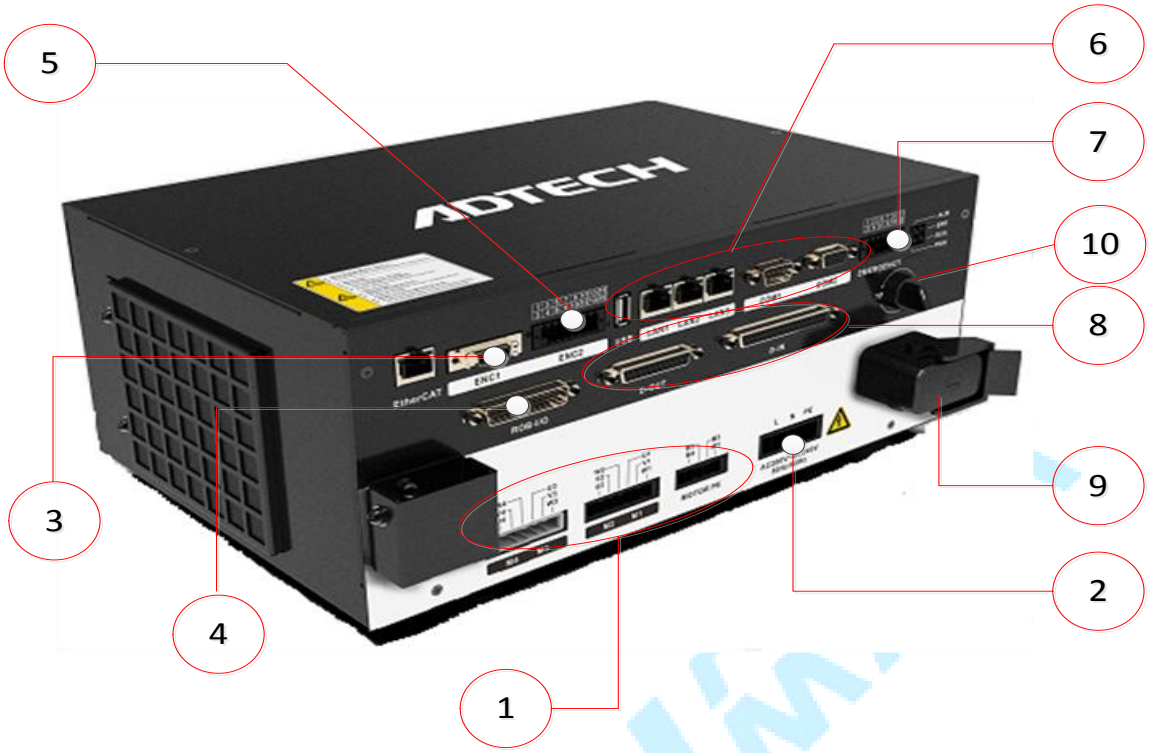
2）RPB07 手持示教器（选配）

3）连接线缆

2.1.2 驱控一体机说明

如图所示为 QC410 驱控一体机侧面示意图。驱控一体机集成多 CPU(A9+DSP+FPGA)控制模块、高性能伺服驱动模块、I/O 模块、显示模块、通讯模块为一体。

接口处主要包括：电机动力线、编码器线、IO 线、电源线等。



① ： 动力线以及地线接口

② ： 电源接口

③ ： 编码器线接口

④ ： 本体IO接口

⑤ ： 外部编码器接口

⑥ ： 通讯接口（TCP/232/USB）

⑦ ： 急停接口

⑧ ： IO板接口

⑨ ： 手持示教器接口（TP）

⑩ ： TP与PC切换旋钮

图 1：QC410 接口介绍

2.1.3 示教器说明

如图所示为手持示教器正面示意图。

图 2： RPB07 正面示意图

手持示教器背面示意图：

图 3：RPB07 背面示意图

示教器操作界面显示：

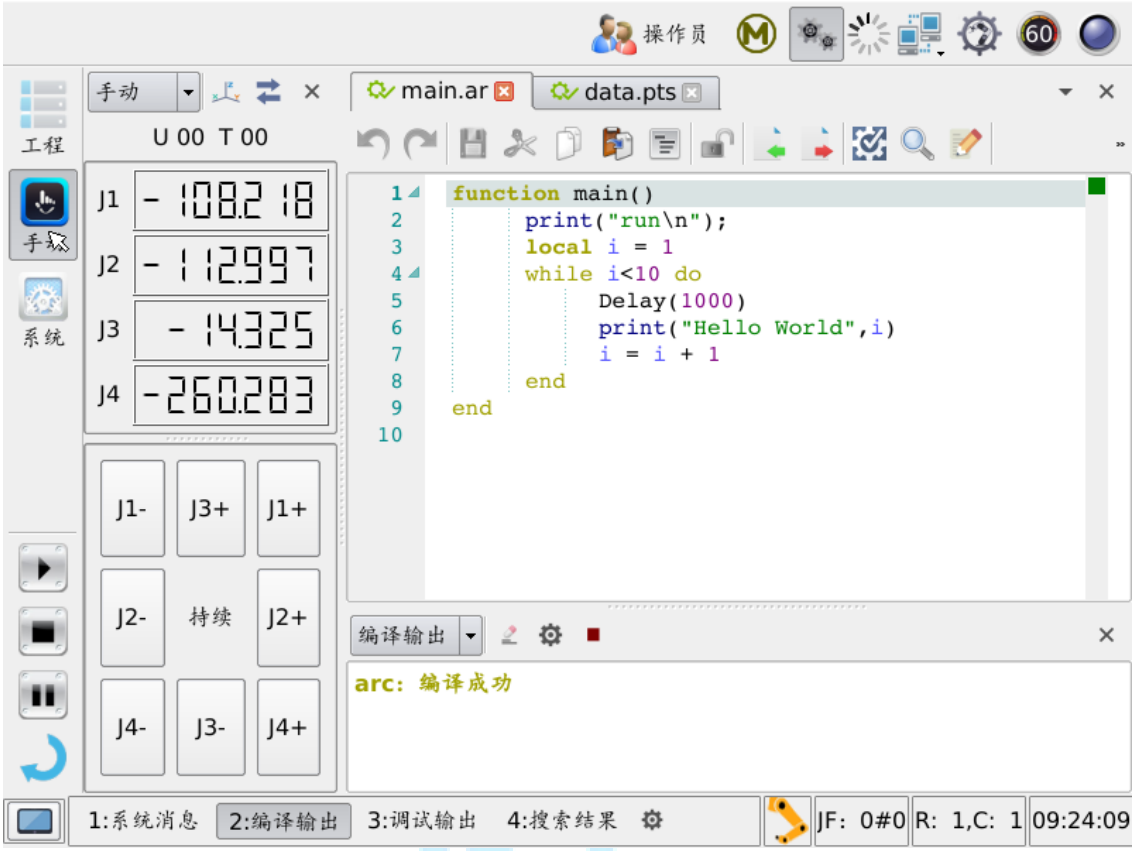


图 4：操作界面显示

示教器显示界面如上图所示，客户可根据实际需求，对机器人参数进行设置，示教器具体的设置与使用方式详见“示教器操作手册”。

2.1.4 功能概述

2.1.4.1 技术参数

表 1-1 驱控一体机技术参数

型号			ADT-QC410
手持示教器	显示屏		彩色显示屏，分辨率 800*600
	编程语言		AR 语言
	示教方式		手动示教、拖拽示教
	尺寸/重量		355mm*248mm*70mm/2KG
QC410 驱控一体机	驱控功能	控制轴数	4 轴
		支持电机类型	支持全数字 AC 伺服（山洋、松下、多摩川、三协、BISS 协议、众为兴等多种品牌电机）
		位置检测方式	检测编码器（增量式/绝对式）
		功率	单轴功率低于 1KW，四轴合计功率低于 3KW
	外部输	标准 IO	34 路（带光耦隔离）

	入		
	外部输出	标准 IO	27 路（6 路继电器、21 路 NPN 开路集电极）
	运动控制功能		直线插补、圆弧插补、连续轨迹、轨迹跟随、平滑以及各类加减速等
	坐标系		关节坐标系、世界坐标系、工具坐标系、用户坐标系
	坐标显示方式		笛卡尔坐标、关节坐标
	外部通讯		RS-232：2CH（9 针） RS-485：1CH（9 针） Ethernet：3CH（100Mbps/10Mbps） EtherCAT：1CH（100Mbps/10Mbps） USB2.0：1CH
	尺寸/重量		长 345mm * 宽（含接线端子）234mm * 高 150mm/7.6KG
	电源		单相 AC200V~240V 以内，50Hz/60Hz

2.1.4.2 产品配置

表 1-2 产品配置

物品名称	说明	数量
QC410	主机控制器	1 台
电源插头	主机供电插头	1 个
本体连接线	主机与电机、编码器、本体 IO 端子连接线	3 根
输入接线	37 针主机输入端口接线	1 根
输入板	I/O 输入板	1 块
输出接线	25 针主机输出端口接线	1 根
输出板	I/O 输出板	1 块
RPB07	手持示教器（选配）	1 台
急停端子	外部急停信号端口	1 个

2.1.4.3 基本规格

表 1-3 基本规格

气候条件		
温度	工作	-5℃～50℃
	贮存运输	-30℃～70℃
相对湿度	工作	20％～90％（℃）
	贮存运输	10％～95％（℃）
大气压力		86Kpa～106Kpa

2.1.4.4 系统功能

ADT-QC410 四轴 SCARA 机器人控制器，硬件部分：创新的采用运动控制系统与伺服驱动系统集成设计技术，达到伺服驱动、控制之间真正的全闭环控制。软

件部分，采用先进的 3D 空间圆弧、NURBS 插补算法技术、伺服震动抑制技术、加工速度自动优化技术，兼容机器人 G 代码、PLC（IEC61131-3）编程的二次开发环境，满足 SCARA 机器人各类应用场合。

2.2 外部接线

2.2.1 系统构成图

使用驱控一体机前，用户需完成系统接线，接线分布如下图：

图 2-1 系统构成图

四轴水平关节机器人平面图如下：

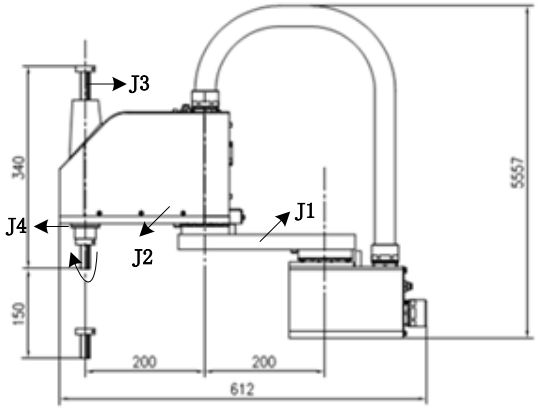


图 2-2 四轴水平机器人平面图

工业机器人各轴定义如下：**大臂为 J1 轴，小臂为 J2 轴，上下轴为 J3 轴，旋转轴为 J4 轴**。

2.2.2 产品各部位说明

QC410 驱控一体机端子面板有多个外部接线端口，丝印对照如下图：

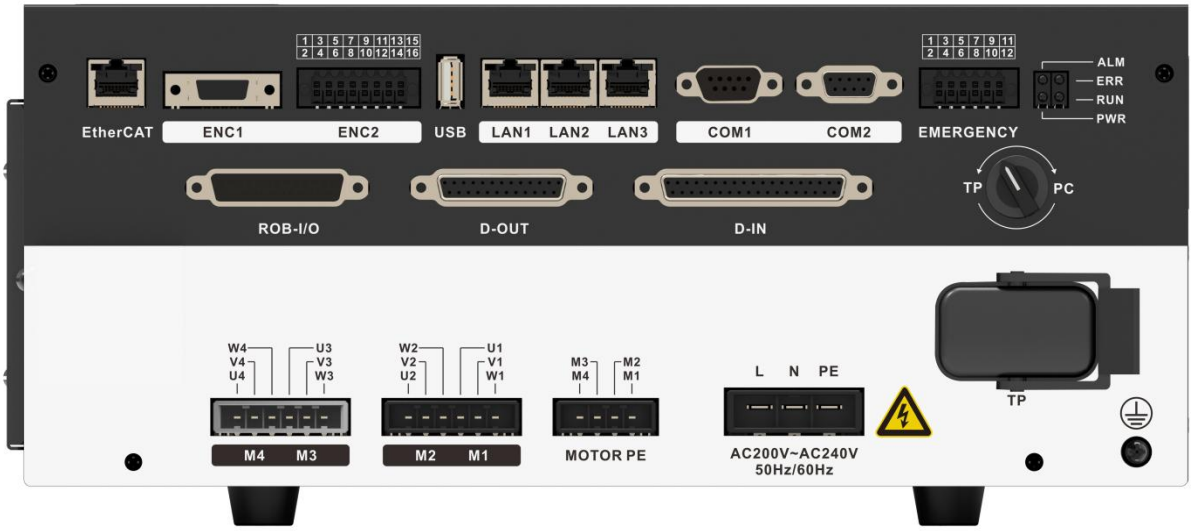


图 2-2 QC410 驱控一体端子说明图

各端口名称及功能介绍如下表所示：

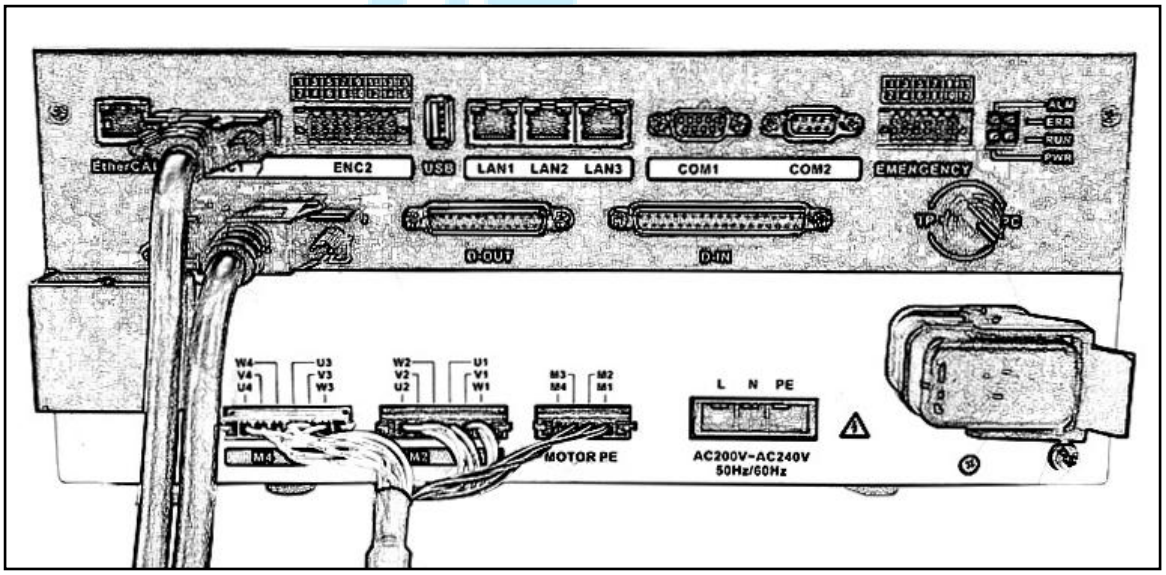
表 2-1 驱控一体机端口列表

端口标识	名称	功能介绍
EtherCAT	EtherCAT 总线端口	IO 扩展、外部轴扩展
ENC1	机器人 1-4 轴编码器端口	机器人本体编码器端口
ENC2	外部扩展编码器端口	外部扩展编码器端口
USB	USB2.0 接口	USB 通讯
COM1	RS232 接口	伺服调试端口
COM2	RS232 接口	RS232/RS485 通讯端口
LAN1	以太网接口	网络通讯
LAN2	以太网接口	网络通讯
LAN3	以太网接口	网络通讯
EMERGENCY	急停端子	外部急停
ALM	LED 指示灯	显示伺服报警状态
ERR	LED 指示灯	显示系统报警状态
RUN	LED 指示灯	显示系统运行状态
PWR	LED 指示灯	显示电源工作状态
ROB I/O	机器人本体 IO 端子	机器人本体 IO 端子，到末端夹具
D-OUT	数字输出端口	数字信号输出，到 25 脚 DO 端子板
D-IN	数字输入端口	数字信号输入，到 37 脚 DI 端子板
TP/PC	模式切换钥匙	启用示教器/屏蔽示教器

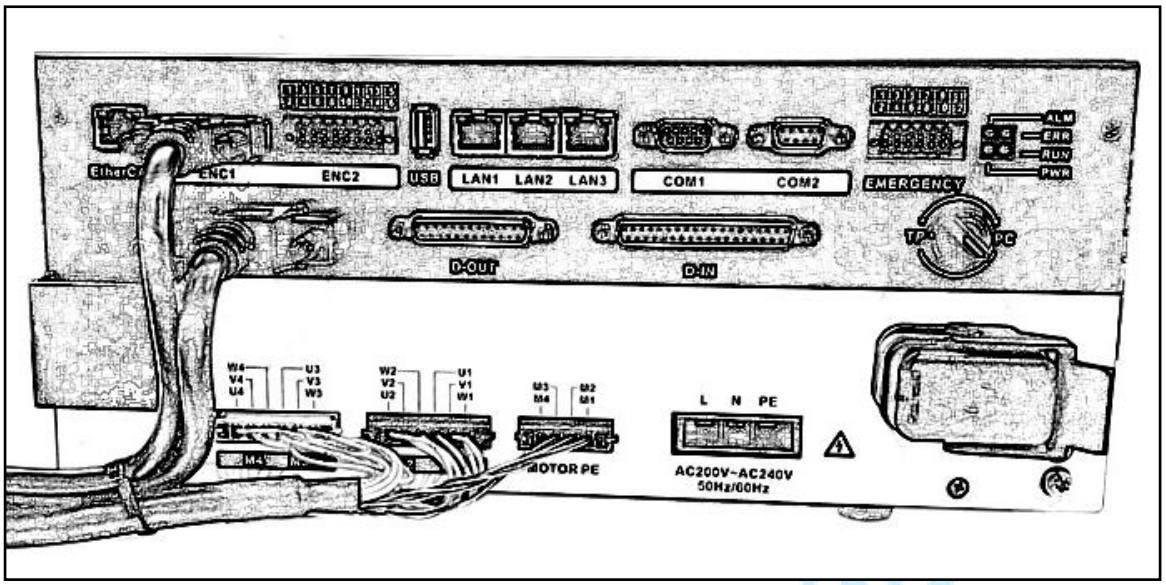
M4 M3	4 轴 3 轴电机动力线端口	4 轴 3 轴电机动力线端口
M2 M1	2 轴 1 轴电机动力线端口	2 轴 1 轴电机动力线端口
MOTOR PE	电机接地端子	电机接地端子，保护地 PE
PE N L	电源端子	电源供电端子 （AC200V~AC240V）
TP	示教器接线端子	左侧为 LAN 通讯线、右侧为急停回路
TP/PC	RPB07 示教器/PC 操作切换旋钮	TP：基于 RPB07 示教器操作机器人 PC：基于 PC（ARStudio）操作机器人

2.2.3 QC410 接线示意图

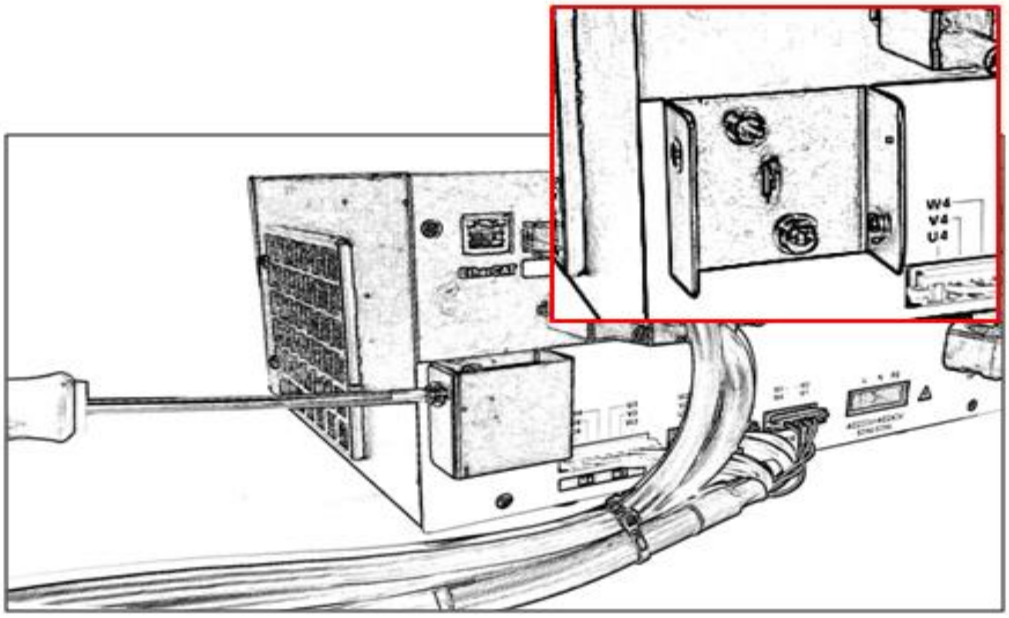
（1）将编码器线、动力线、本体 IO 线插入到对应的 QC410 控制器的接口：



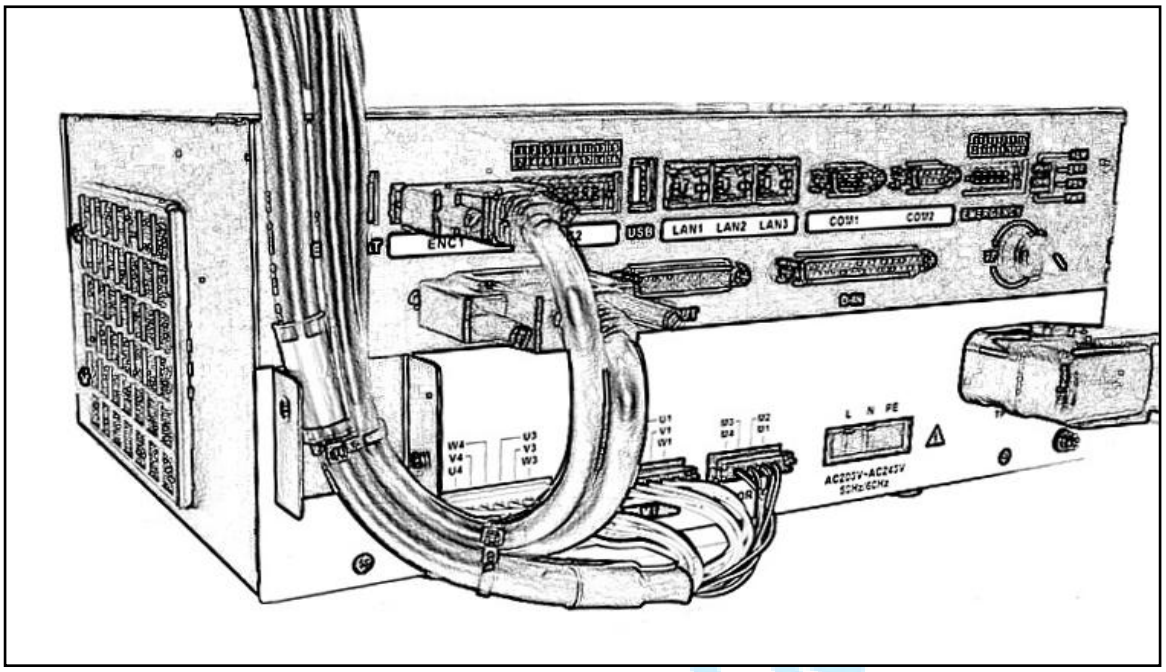
（2）用配置包里附带的扎线带按图示将三股线扎在一起：



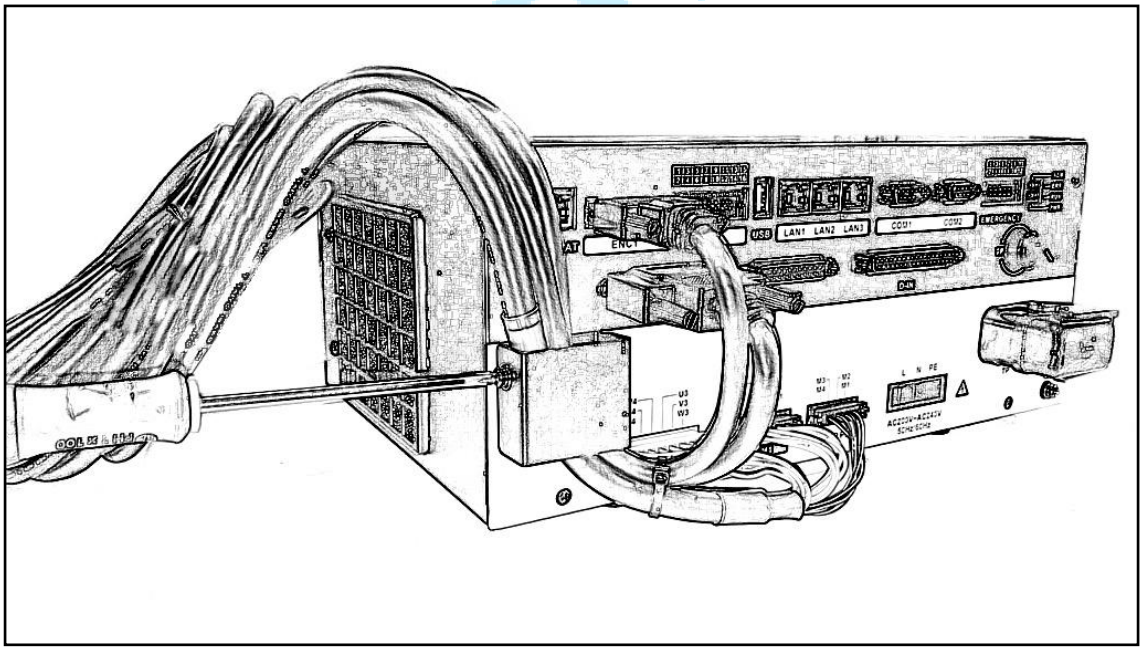
（3）拆下接线板上壳的螺钉：



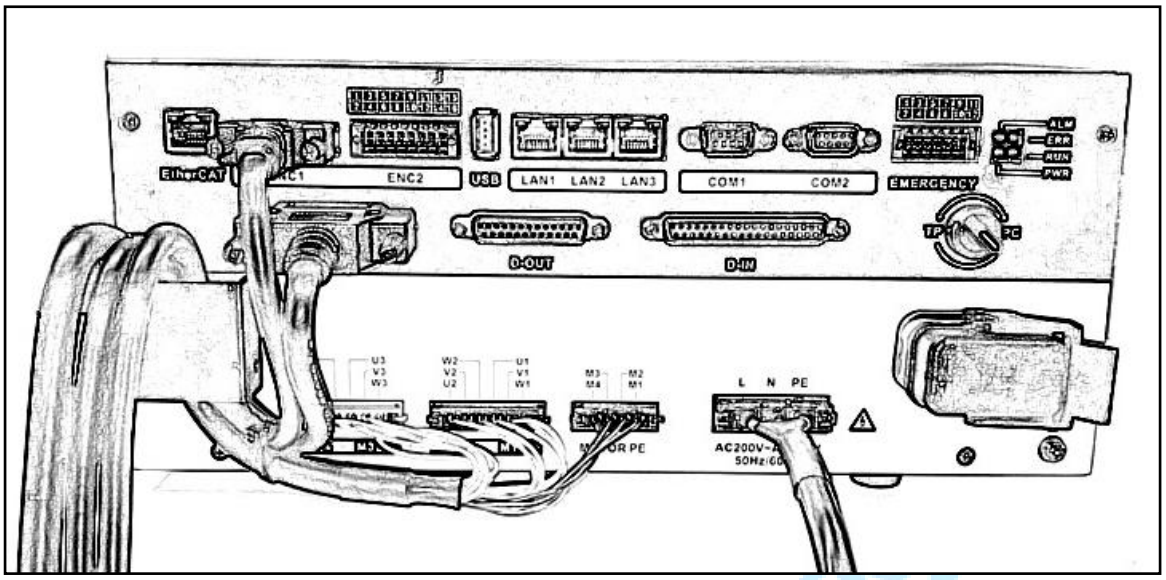
（4）再用扎线带将线绑在接线板内：



（5）盖上接线板上壳并锁上螺丝：



（6）接线完成示意图：


EtherCAT
ENC2
USB
LAN1 LAN2 LAN3
COM1 COM2
EMERGENCY
ALM
ERR
RUN
PWR
TP
PC
D-OUT
D-IN
L N PE

2.3 电源线以及外部急停端子接线

2.3.1 电源线的连接

2.3.1.1 AC220V 主电源

驱控一体机电源线端子（AC　IN）连接示意图如下所示：

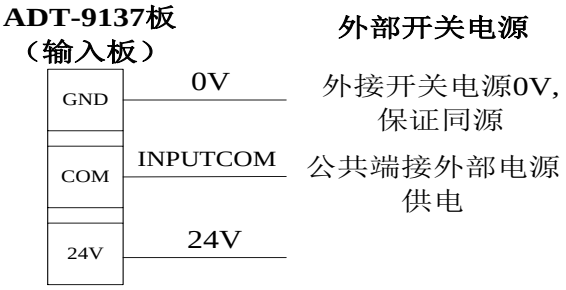
图 3-1 电源端子示意图

接线端子规格：3 位，单层，母头，脚间距 10.16mm，黑色，两端带卡扣，总高 29mm；电源线线缆规格为：3 芯，200mm，2.5mm2 ， L（棕色）N（蓝色）PE（黄绿）。

2.3.1.2 DC24 输入输出信号电源

IO 板内部已提供 DC24V 电源，供输入输出用，开机后 ADT9137 输入板上的 GND 引脚与 24V 引脚之间应有 DC24V；

IO 板也可使用外部提供 DC24V 电源，接线示意图如下所示：



注意：

1、IO 板输入输出共用同一个电源，即输入板的 24V 与输出板的 24V 是同一个 24V；

2、输入输出板上有多个 24V、COM、GND 端口，同名端子内部是导通的，其目的仅为便于接线；

3、输出板输出端口不宜直接驱动电磁阀，建议加继电器转接。

4、QC100 控制器内部提供的是 24V 直流输出电压，其额定输出功率为 60W，额定输出电流为 2.5A。

5、QC100 控制器内部提供的 24V 电源主要用于控制器内部的风扇以及每个轴的刹车供电，客户现场使用的输入输出须优先考虑使用外部 24V 电源供电；当外部使用的输入输出负载小于 2A 时，可考虑使用 QC410 控制器内部提供的 24V 电源。

2.3.2 外部急停接线

2.3.2.1 急停端子引脚定义

急停端子丝印图

急停端子丝印编号：

1	3	5	7	9	11
2	4	6	8	10	12

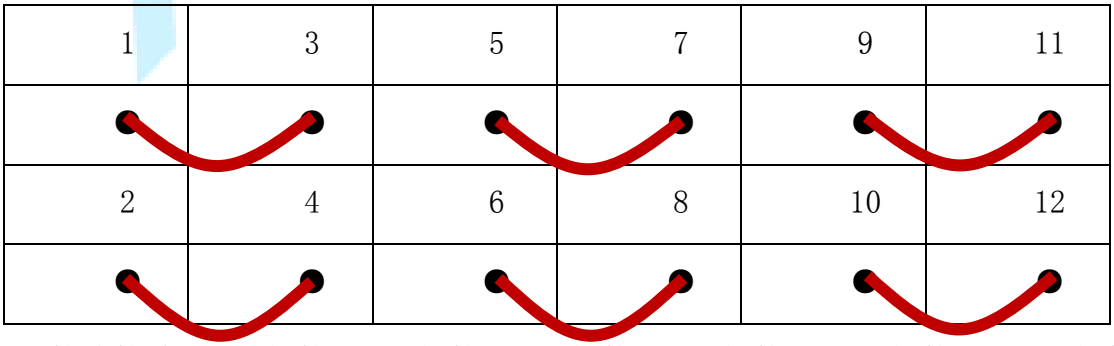
急停端子实物图

急停端子引脚定义：

VCC24V	ESTOP11	ESTOP12	ESTOP-RDY1	ESTOP-COM1	GND
VCC24V	ESTOP21	ESTOP22	ESTOP-RDY2	ESTOP-COM2	GND

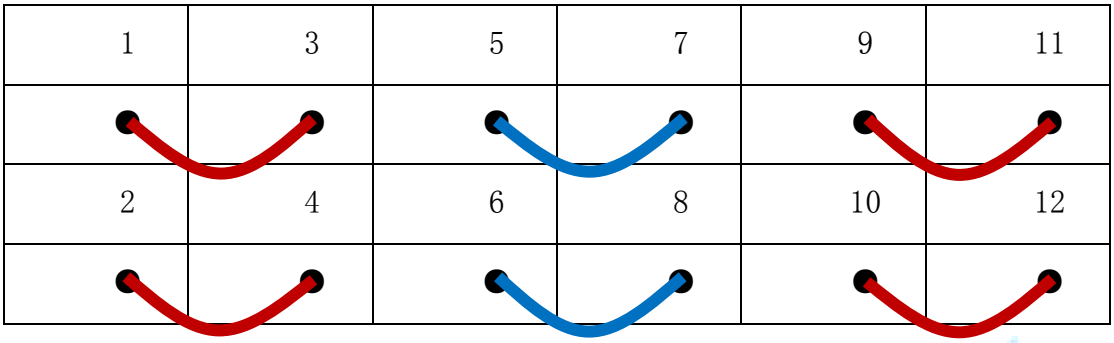
2.3.2.2 急停端子接线

1） 不接外部急停按钮：不通过外部急停按钮控制机器人的急停



接线接法：1-3 短接，2-4 短接，5-7 短接，6-8 短接，9-11 短接，10-12 短接；

2） 接外部急停按钮：通过外部按钮控制机器人的急停

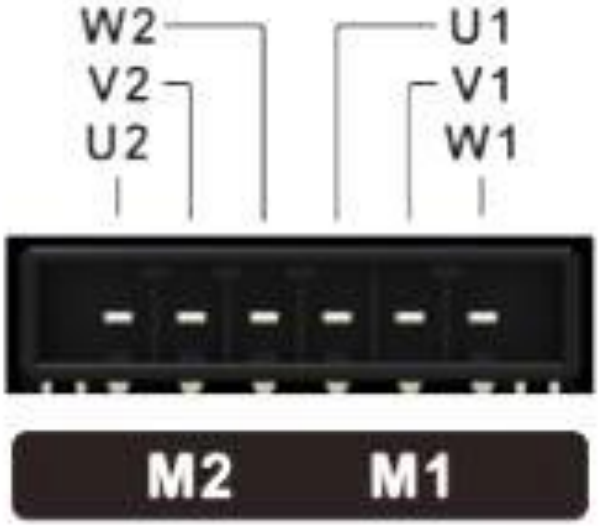


接线接法：1-3 短接，2-4 短接，9-11 短接，10-12 短接；5-7、6-8 之间接外部急停引线；以上 6 组可任选一路或多路，其它短接。（注意：急停可以接机械开关常闭触点或者继电器常闭触点，不可以接有源信号触点。PLC 输出信号做急停时需要加继电器转换，急停不可以与其它信号点串联或并联使用。

2.4 机器人本体与 QC410 控制器连线

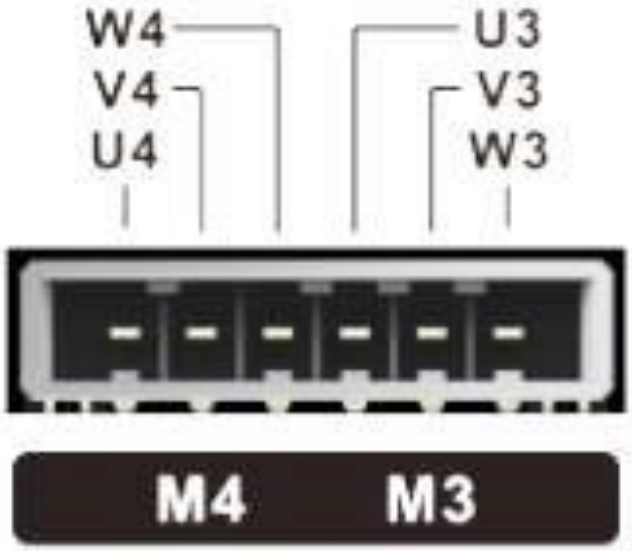
2.4.1 电机动力线接口

2.4.1.1 M1、M2 动力线接口



引脚	定义	功能
1	U2	M2 电机 U 相供电
2	V2	M2 电机 V 相供电
3	W2	M2 电机 W 相供电
4	U1	M1 电机 U 相供电
5	V1	M1 电机 V 相供电
6	W1	M1 电机 W 相供电

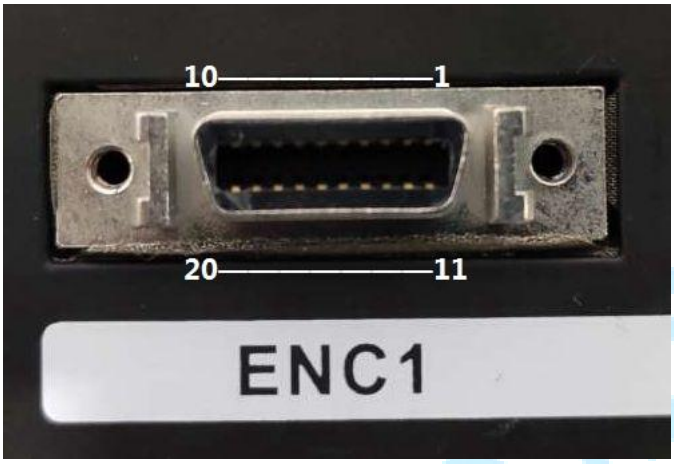
2.4.1.2 M3、M4 动力线接口



引脚	定义	功能
1	U4	M4 电机 U 相供电
2	V4	M4 电机 V 相供电
3	W4	M4 电机 W 相供电
4	U3	M3 电机 U 相供电
5	V3	M3 电机 V 相供电
6	W3	M3 电机 W 相供电

2.4.2 电机编码器线

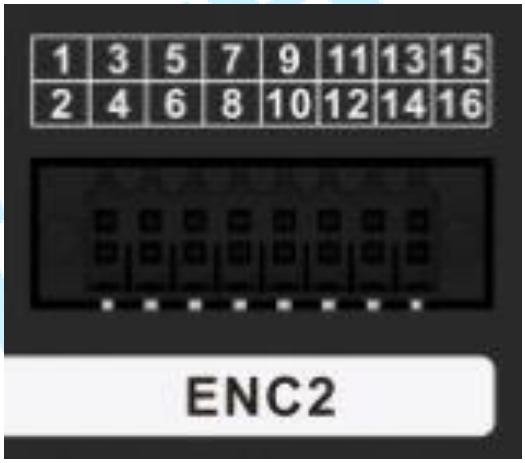
2.4.2.1 M1、M2、M3、M4 编码器接口 ENC1



引脚	定义	功能
1	VDD5V3	M1 编码器 5.3V 电源
2	GND_ENC	M1 编码器信号地
3	M1+	M1 编码器信号+
4	M1-	M1 编码器信号-
5	/	/
6	/	/
7	VDD5V3	M2 编码器 5.3V 电源
8	GND_ENC	M2 编码器信号地
9	M2+	M2 编码器信号+
10	M2-	M2 编码器信号-
11	VDD5V3	M3 编码器 5.3V 电源

12	GND_ENC	M3 编码器信号地
13	M3+	M3 编码器信号+
14	M3-	M3 编码器信号-
15	/	/
16	/	/
17	VDD5V3	M4 编码器 5.3V 电源
18	GND_ENC	M4 编码器信号地
19	M4+	M4 编码器信号+
20	M4-	M4 编码器信号-

2.4.2.2 M5、M6 编码器接口（扩展接口 ENC2）



引脚	定义	功能
1	VDD5V3	M5 编码器 5.3V 电源
2	VDD5V3	M6 编码器 5.3V 电源
3	GND_ENC	M5 编码器信号地

4	GND_ENC	M6 编码器信号地
5	M51+	M5 编码器信号 1+
6	M61+	M6 编码器信号 1+
7	M51-	M5 编码器信号 1-
8	M61-	M6 编码器信号 1-
9	M52+	M5 编码器信号 2+
10	M62+	M6 编码器信号 2+
11	M52-	M5 编码器信号 2-
12	M62-	M6 编码器信号 2-
13	/	/
14	/	/
15	PE	接编码器线缆的屏蔽层
16	PE	接编码器线缆的屏蔽层

支持的常见编码器品牌有多摩川、松下、尼康，增量式或绝对式，如选用其它品牌先咨询供应商。

当使用 ABZ 增量型编码器时，需要选用差分输出类型，A+、A- 、B+、B- 分别对应

接 1+、 1- 、2+、 2-，编码器 Z+ 、Z-可悬空不接。

当使用通讯协议类型的编码器时 SD+ 、SD- 分别对应接 1+ 、1-，ENC1 其它脚悬空。

特别注意不要接错电源脚，防止损坏编码器。

2.4.3 本体 IO

2.4.3.1 ROB-I/O 端子引脚定义（控制器侧）

QC410 控制器有一部分 IO 通过 ROB-I/O 端子引入到机器人本体，用于第一轴(大

机型）、第三轴刹车控制，以及方便安装在机器人丝杆末端夹具、电磁阀等设备的接线。

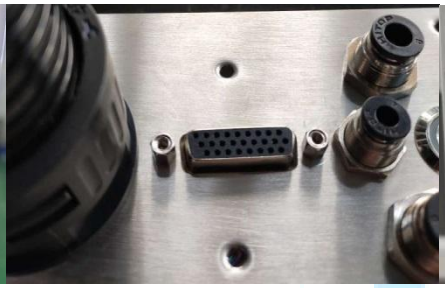
引脚	定义	功能
1	IN28	机器人本体端输入点 DI_28
2	IN29	机器人本体端输入点 DI_29
3	IN30	机器人本体端输入点 DI_30
4	IN31	机器人本体端输入点 DI_31
5	IN32	机器人本体端输入点 DI_32
6	IN33	机器人本体端输入点 DI_33
9	OUT19	机器人本体端输出点 DO_19
10	OUT20	机器人本体端输出点 DO_20
11	OUT21	机器人本体端输出点 DO_21
12	OUT22	机器人本体端输出点 DO_22
13	Relay1	机器人本体端继电器输出点 OUT_23
14	Relay2	机器人本体端继电器输出点 OUT_24
15	Relay3	机器人本体端继电器输出点 OUT_25
16	Relay4	机器人本体端继电器输出点 OUT_26
17~20	EXT24V	24V 电源

22~23	GND	24V 地
7、8、21、24、25	暂未使用,用户自定义	

2.4.3.2 本体 IO 端子引脚定义（本体侧）

此接口为方便用户末端夹具接线用，端子位于机器人 J2 臂（如下图），内部已接好线经过弯管引到控制器 ROB-I/O 接口。

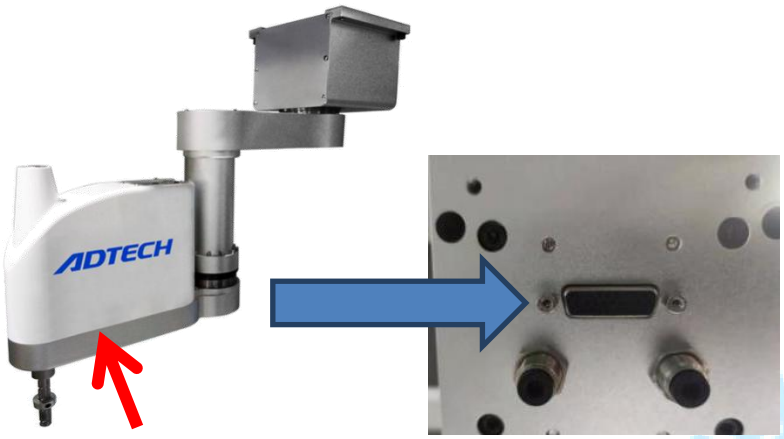
AR 系列（小臂后方）



GR 系列（小臂正上方）

FR 系列（小臂正下方）

➢ AR 系列及 FR 系列（FR8320 机型除外）

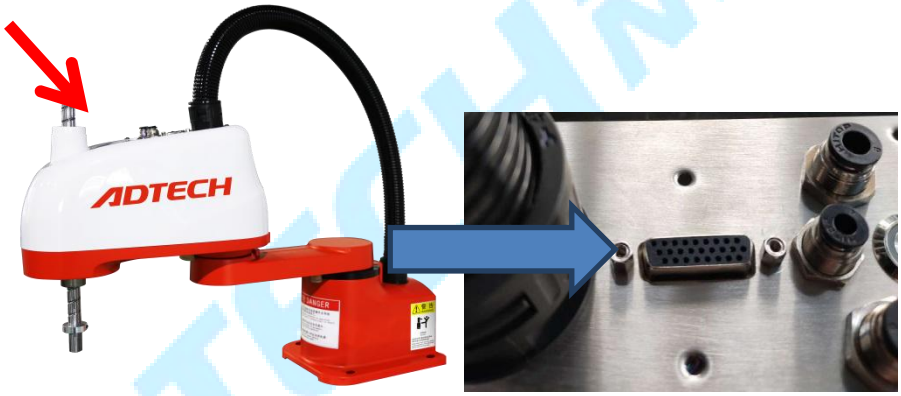


引脚	定义	功能
1	IN28	机器人本体端输入点 DI_28
2	IN29	机器人本体端输入点 DI_29
3	IN30	机器人本体端输入点 DI_30
4	IN31	机器人本体端输入点 DI_31
5	IN32	机器人本体端输入点 DI_32
6	IN33	机器人本体端输入点 DI_33
7	Relay1	机器人本体端继电器输出点 OUT_23
8	Relay2	机器人本体端继电器输出点 OUT_24
9	OUT19	机器人本体端输出点 DO_19
10	OUT20	机器人本体端输出点 DO_20
11	OUT21	机器人本体端输出点 DO_21
12	OUT22	机器人本体端输出点 DO_22
13	Relay3	机器人本体端继电器输出点 OUT_25
14	Relay4	机器人本体端继电器输出点 OUT_26

15	EXT24V	24V 电源
16	GND	24V 地
17、18、19、20、21、22、23、24、25、26	暂未使用,用户自定义	

说明：Relay1、Relay2、Relay3、Relay4（DO_23、DO_24、DO_25、DO_26）设计为刹车端口，Relay1 用作 J3 轴刹车输出对应 DO_23,已经被系统占用, Relay2 用作 J1 轴刹车只在 AR61030、AR81030、AR101030 等大型机上使用, Relay3、Relay4 为空闲可用户自由配置。

➢ GR 系列及 FR8320 机型



引脚	定义	功能
1	IN28	机器人本体端输入点 DI_28
2	IN29	机器人本体端输入点 DI_29
3	IN30	机器人本体端输入点 DI_30
4	IN31	机器人本体端输入点 DI_31
5	IN32	机器人本体端输入点 DI_32
6	IN33	机器人本体端输入点 DI_33

8	Relay2	机器人本体端继电器输出点 OUT_24
9	OUT19	机器人本体端输出点 DO_19
10	OUT20	机器人本体端输出点 DO_20
11	OUT21	机器人本体端输出点 DO_21
12	OUT22	机器人本体端输出点 DO_22
13	Relay3	机器人本体端继电器输出点 OUT_25
14	Relay4	机器人本体端继电器输出点 OUT_26
15	EXT24V	24V 电源
16	GND	24V 地
7、17、18、19、20、21、22、23、24、25、26	暂未使用,用户自定义	

说明：Relay1、Relay2、Relay3、Relay4（DO_23、DO_24、DO_25、DO_26）设计为刹车端口，Relay1 用作 J3 轴刹车输出对应 DO_23,已经被系统占用, Relay2 用作 J1 轴刹车只在 AR61030、AR81030、AR101030 等重型机上使用, Relay3、Relay4 为空闲可用户自由配置。

2.5 数字输入输出接口

2.5.1 数字输入接口

QC410 中包含一个扩展接线板，该板由外部 24V 电源独立供电，具有 34 路普通输入和 27 路普通输出（包含 6 路继电器输出）。输入有效电平可以通过 COM 端来选择是高电平还是低电平有效；输出为开漏输出,低电平有效，不可选择。

输入板中单个输入能承受的最大电流为 60mA,正常使用电流为 5~10mA.

2.5.1.1 输入接口定义及引脚功能

数字输入接口 D-IN 采用一个 37 针 DB 头，为方便用户接线标配一条 37 芯延长线和导轨安装式的接线板 ADT-9137。用户使用时 37 芯延长线一端接控制器 D-IN 端口，另一端接 ADT-9137 端子板。数字输入端口引脚排列如图所示：

图 5-1 数字输入端口

各个输入端口定义及引脚功能如下表所示：

丝印号	端口定义	功能介绍	丝印号	端口定义	功能介绍
0-25	IN0-IN25	普通输入端口 0-25	GND	0V/GND	内部 24V 电源地
26-27	IN26-IN27	高速输入端口 26、27	24V	+24V/P24V	内部 24V 电源正
COM	INPUTC	输入信号公共端（低电平有效时			

	OM	接 24V，高电平有效时接 GND）			

说明：高速输入也可以作为普通输入使用；丝印相同的脚位内部是导通的可任取一路接线

2.5.1.2 数字输入简要电路图

数字输入简要内部电路：

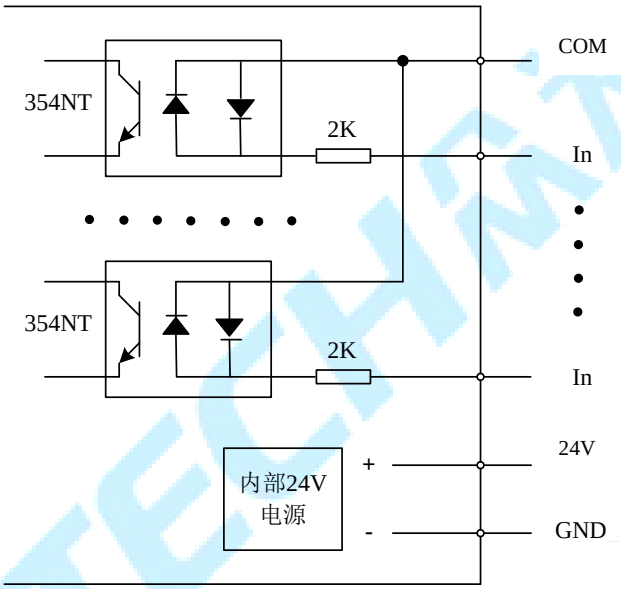
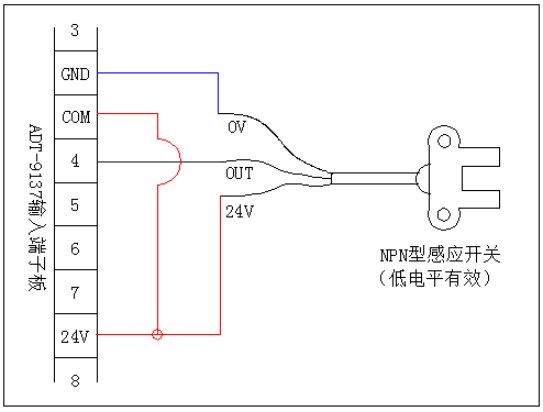
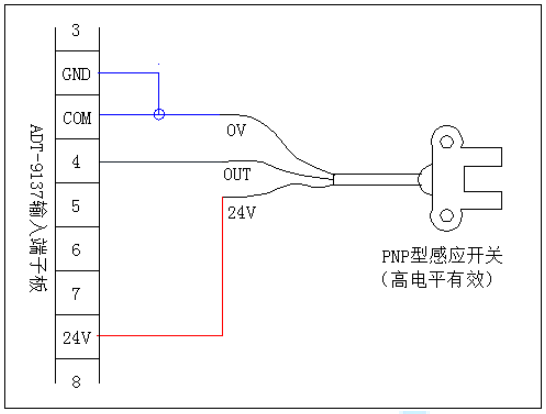


图 5-2 数字输入内部电路

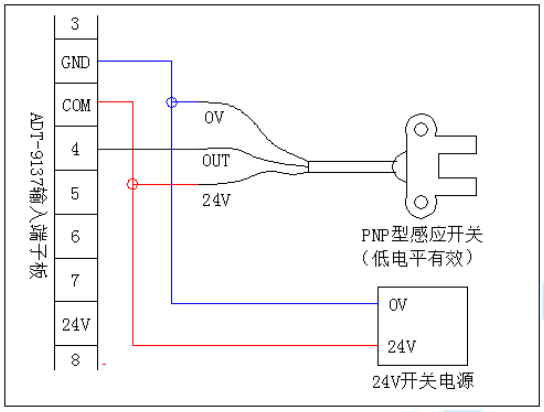
与感应开关接线实例如下，与 PLC 接线时相同，注意同一时刻只支持一种有效电平。输入板的 24V 可以由 QC410 自提供也可以由外部 24 电源提供，两种接线方式分别如下：

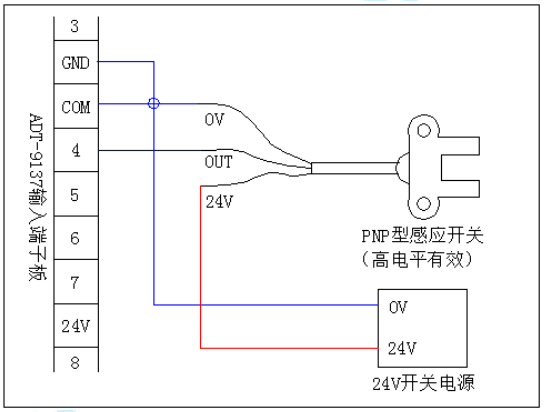
(1) 当使用机器人内部 24V 电源时：





(2) 当使用外部 24V 开关电源时：





2.5.2 数字输出接口

2.5.2.1 输出接口定义及引脚功能

数字输出接口 D-OUT 采用一个 25 针 DB 头，为方便用户接线标配一条 25 芯延长线和导轨安装式接线板 ADT-9125。用户使用时 25 芯延长线一端接控制器 D-OUT 端口，另一端接 ADT-9125 端子板。数字输出端口引脚排列如图所示：

图 5-5 数字输出端口

丝印标注 24V 的引脚是系统内部 24V 电源正，与输入板的 24V 为同一组电源，可供给外部使用。输出板上没有 0V 端子,接线时接到输入的 0V 端子。

系统输出信号包括两路继电器接点输出，其中“9”和“9”为一路继电器，对应控制程序 OUT_9；“18”和“18”为另一路继电器对应控制程序 OUT_18。

输出板中的单个输出端口能够输出的最大电流为 1A，正常使用电流在 50mA 以内（当输出点数较多的情况下请控制在 30mA 以内）。

输出端口定义及引脚功能如下表所示：

丝印号	端口定义	功能介绍	丝印号	端口定义	功能介绍
0～8	OUT0～OUT8	普通输出端口 0～8	9、9	OUT9	继电器常开输出点

10-17	OUT10～OUT17	普通输出端口 10～17	18、18	OUT18	继电器常开输出点
24V	24V	内部 24V 正			

2.5.2.2 数字输出简要电路图

1.数字输出内部电路：

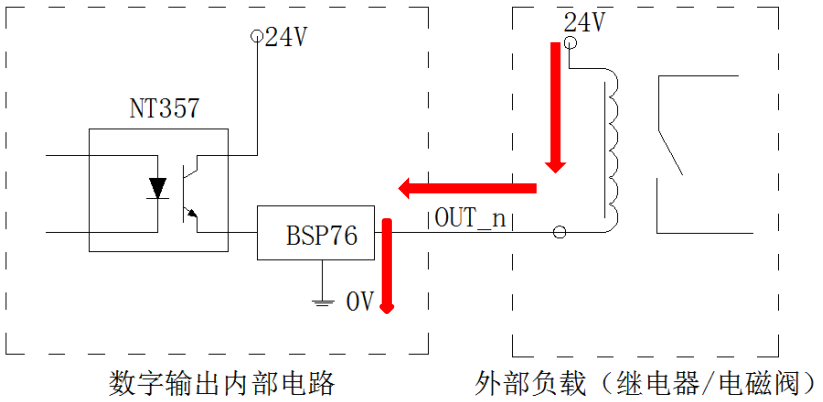


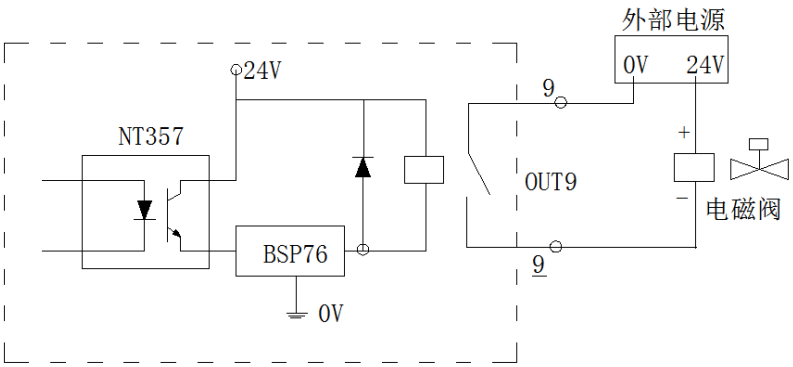
图 5-5 数字输出内部电路

上图给出了数字量输出信号内部简要电路，以及输出信号与继电器负载的接线示意，红色箭头表示负载电流回路，线圈 24V 正极可以直接接到 ADT-9125 输出端子板的 24V；当负载为多路电磁阀或继电器时，建议 24V 正极从外部 24V 开关电源取电，同时外部电源的 0V 与机器人输入板的 0V 接通。

数字输出与 PLC 输入信号接线时，要求 PLC 输入为低电平有效，机器人输出接 PLC 输入，另外机器人输入板的 0V 要与 PLC 的 0V 接到一起。

2.继电器输出内部电路：

继电器输出属于隔离输出，适用于负载电流较大、不同电压等级、不同有效电平等场合。



继电器输出内部电路

图 5-6 继电器内部电路图

2.6 通信端口

2.6.1 串口 COM1 端口接线

为便于驱控一体中伺服程序的升级与维护，QC410 与 PC 机通过 COM1 端口通信，COM1 端口采用标准的 DB9 公头端子，采用 RS-232 接口，无需电平转接板。

端口定义如图所示：

图 6-1 COM1 端口示意图

各个引脚功能与说明如表所示：

引脚	定义	功能
1	DSP_RX	伺服调试串口，RS232 RX 引脚
2	TX0	系统调试串口，RS232 TX 引脚
3	RX0	系统调试串口，RS232 RX 引脚
4	/	/
5	GND	系统信号地
6	DSP_TX	伺服调试串口，RS232 TX 引脚
7	/	/
8	/	/

9	/	/

表 6-1 COM1 端子信号说明

QC410 驱控一体伺服调试时，通过 COM1 与 PC 机相连，连接示意图如下：

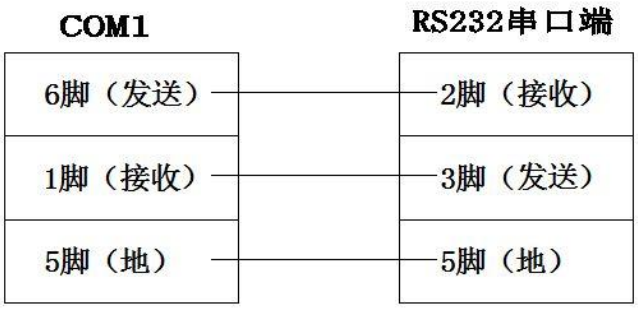


图 6-2 COM1 端口与 RS232 串口端连接示意图

QC410 驱控一体系统调试（如 BIOS 操作时），通过 COM1 与 PC 机相连，连接示意图如下：

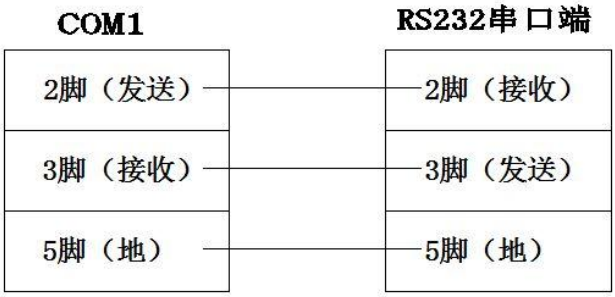


图 6-3 COM1 端口与 RS232 串口端连接示意图

2.6.2 串口 COM2 端口接线

为便于驱控一体中控制器的维护，QC410 驱控一体上提供 COM2 通信端口，COM2 端口采用标准的 DB9 母头端子，无需电平转接板。

端口定义如图所示：

图 6-4 COM2 端口示意图

各个引脚功能与说明如表所示：

引脚	定义	功能
1	485_A	RS485-A 引脚
2	TXD_232	RS232 TX 引脚
3	RXD_232	RS232 RX 引脚
4	/	/
5	GND_232	隔离 RS232 信号地
6	485_B	RS485-B 引脚
7	GND_485	隔离 RS485 信号地
8	/	/
9	/	/

表 6-2 COM2 端子信号说明

QC410 驱控一体与外围设备进行 RS232 串口通讯时，通过 COM2 中（2、3）脚进行发送接收数据，连接示意图如下：

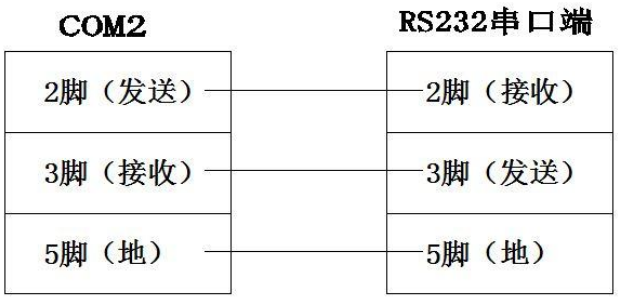


图 6-5 COM2 端口与 RS232 串口端连接示意图

QC410 驱控一体与外围设备进行 RS485 串口通讯时，通过 COM2 中（1、6）脚进行发送接收数据，连接示意图如下：

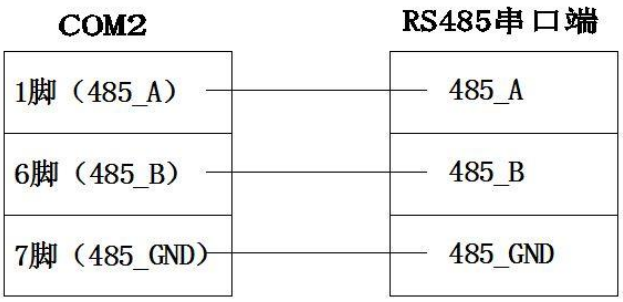
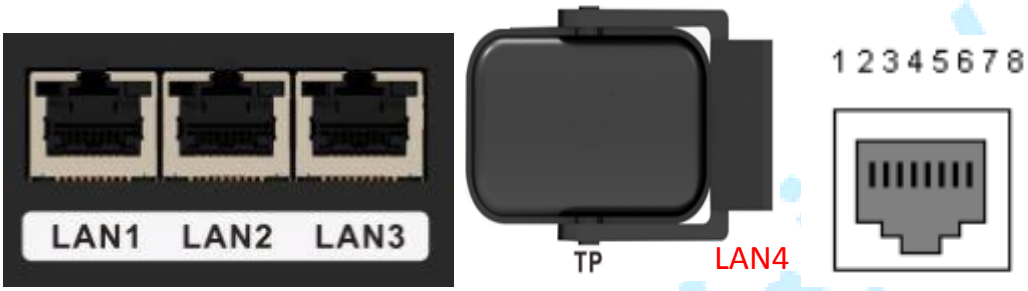


图 6-6 COM2 端口与 RS485 连接示意图

2.6.3 LAN 端口

2.6.3.1 EtherNet 网络接口 LAN1、LAN2、LAN3、LAN4

QC410 驱控一体机提供以太网接口 LAN1、LAN2、LAN3、LAN4，支持 TCP/UDP 协议，最大通信速率 100M，其 IP 地址相同，出厂默认为 192.168.0.123；其中 LAN1、LAN2、LAN3 为用户接口， LAN4 为示教器专用网口内置于 TP 接口。



用户网络接口示教器专用网口脚位定义

图 6-7 LAN 网口

各个引脚功能与说明如表所示：

引脚编号	信号名称	说明
1	TX+	发送数据+
2	TX-	发送数据-
3	RX+	接收数据+
4	N/C	双向数据
5	N/C	双向数据
6	RX-	接收数据-
7	N/C	双向数据
8	N/C	双向数据

表 6-3 LAN 网口端子信号说明

2.6.3.2 EtherCAT 接口

QC410 驱控一体网络接口 EtherCAT，支持 EtherCAT 协议，最大通信速率 100M。

EtherCAT 接口定义如下：

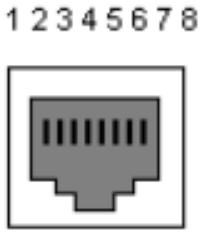


图 6-8 EtherCAT 网口

各个引脚功能与说明如表所示：

引脚编号	信号名称	说明
1	TX+	发送数据+
2	TX-	发送数据-
3	RX+	接收数据+
4	N/C	双向数据
5	N/C	双向数据
6	RX-	接收数据-
7	N/C	双向数据
8	N/C	双向数据

表 6-4 EtherCAT 端子信号说明

2.6.3 USB 接口

QC410 驱控一体提供标准的 USB 通信接口，用于 U 盘复制数据，端口内部引脚排列如下：

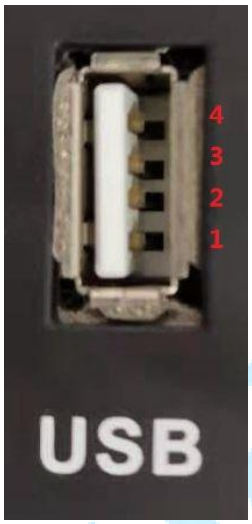


图 6-9 USB 端子

各个引脚功能与说明如表所示：

引脚编号	信号名称	说明
1	VBUS	USB 电源端（+5V）
2	D-	信号负
3	D+	信号正
4	GND	地线

表 6-5 USB 端子信号说明

2.7 产品安装尺寸图

2.7.1 产品安装尺寸图

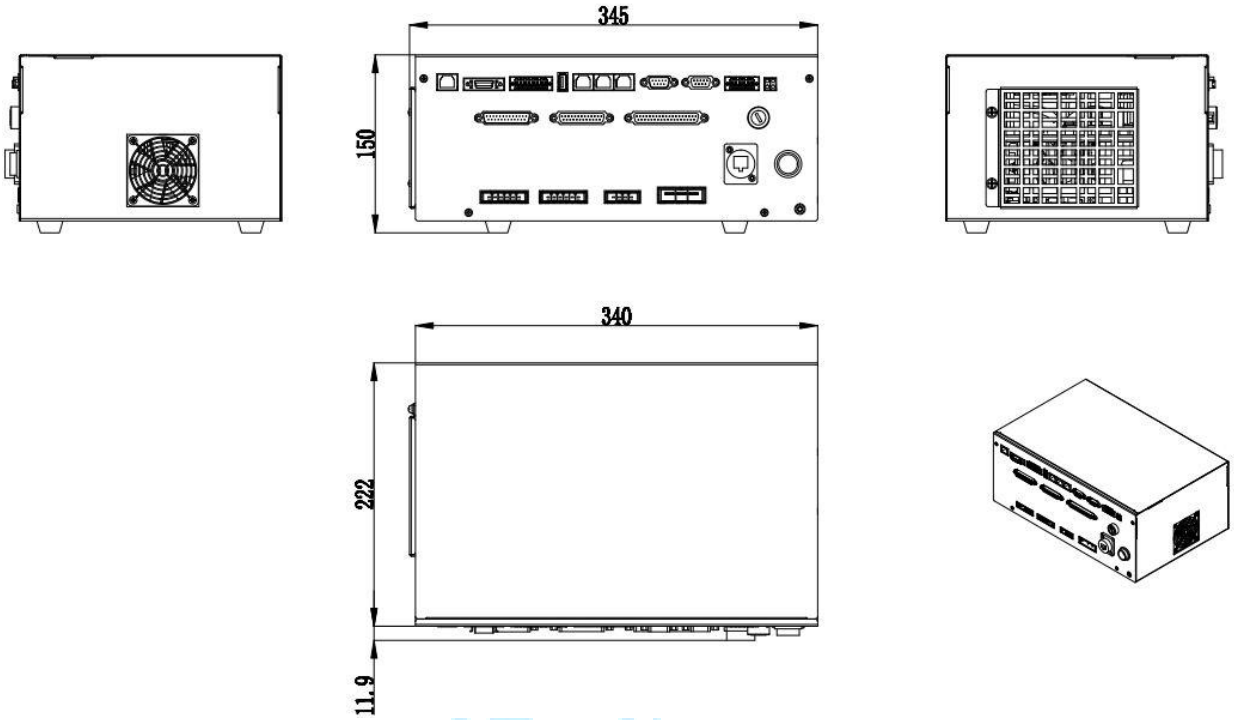


图 7-1 驱控一体外形尺寸图

对机械手臂电机动力线、编码器线、IO 线连接时，必须切断用户所有的主电源之后，方能进行操作。

为避免电缆线过度折弯，控制器接线端子面板位置需要预留 120mm 以上安装空间，即上图 7-1 所示宽度方向 222mm 基础上增加线缆折弯空间。

设备布局时，如需要将机器人控制器与本体之间的线缆穿过设备台面或隔层，建议开孔尺寸：圆孔直径 70mm,或方孔 70mm×50mm。

2.8 附录：修改记录

修改时间	修改内容
2020.3.9	1. 增加了 QC400 控制器内部 24V 电源的规格：额定功率、额定电流；

	2. 增加了输入板中每个输入端口能承载的最大电流以及正常使用电流； 3. 增加了输出板中每个输出端口能驱动的最大电流以及正常使用电流；
2020.3.27	1. 修改 2.6.3.1 章节网口 LAN4 描述
2020.5.14	1. 新增本体侧 ROB-I/O 端子引脚定义； 2. 完善 AR、GR、FR 系列本体 IO（26pin）端子引脚定义；

第三章 AR 语言手册篇

3.1 AR 语言总览

指令类别	指令符号	指令说明
算术运算符	+	加法运算
	-	减法运算
	*	乘法运算
	/	除法运算
	//	取整除法，即得到不大于结果的最大整数值
	%	取余除法
	^	指数运算
	-	取负运算
	~	异或运算
	&	与运算
	|	或运算
	~	取反运算
	<<	左移运算
	>>	右移运算
关系运算符	==	等于
	~=	不等于
	<=	小于等于
	>=	大于等于
	<	小于
	>	大于
逻辑运算符	or	逻辑或
	not	逻辑非
	and	逻辑与
	false	假（注：nil 也为 false）

	true	真
一般符号	#	求 table 数组长度
	=	赋值操作
	--	单行注释
	--[[	多行注释开始行
	--]]	多行注释结束行
	( )	用于函数定义、调用，表达式运算
	{ }	用于定义 table 数组
	[ ]	table 数组元素操作符
	::	定义 goto 指令的跳转位置标签
	;	语句结束符，可以省略不写
	,	用于函数参数定义、调用，多变量赋值，table 数组定义等
	.	用于 table 数组元素访问
	..	字符串连接符
	…	定义函数可变参数项
过程控制指令	if then else elseif end	if 条件分支指令
	while do end	while 循环控制指令
	for do end	for 循环控制指令
	repeat until	repeat 循环控制指令
	goto	goto 无条件跳转指令
	function end	用户函数定义指令
一般关键字	break	跳出 for、while、repeat 循环
	local	定义局部变量
	nil	变量为空值
	return	函数调用返回
机器人轴定义	AX	笛卡尔坐标系 X 轴号
	AY	笛卡尔坐标系 Y 轴号
	AZ	笛卡尔坐标系 Z 轴号

	AC	笛卡尔坐标系 C 轴号
	J1	J1 关节
	J2	J2 关节
	J3	J3 关节
	J4	J4 关节
字符串操作	string.find	函数用于在一个给定的目标字符串中搜索一个模式，返回该模式所在的位置
	string.match	用于在一个字符串中搜索一种模式，返回的是目标字符串中与模式相匹配的那部分字符串
	string.sub	截取字符串 s 的从第 i 个字符到第 j 个字符之间的字符串
	string.gsub	将目标字符串中所有出现模式的地方替换为目标字符串
	string.char,string.byte	用于转换字符和对应的数字
	string.format	用来对字符串格式化，与 C 语言中的 printf()函数类似
	string.len	计算字符串的长度
	tonumber	将数字组成的字符串 string 变量转化成数字 number 类型
	tostring	将布尔类型和数值类型转换为字符串类型
机器人全局变量	ON	打开状态
	OFF	关闭状态
	p0~p2999	机器人默认点名称
运动指令	MovL	直线方式运行到笛卡尔坐标系绝对位置的指令
	MovLR	直线方式运行到笛卡尔坐标系相对位置的指令
	MovP	点到点方式运行到笛卡尔坐标系绝对位置的指令
	MovPR	点到点方式运行到笛卡尔坐标系相对位置的指令
	MovJ	控制各个关节移动到目标绝对角度的指令
	MArchP	点到点方式控制机器人进行拱形移动的指令

	MArc	从当前位置圆弧插补到笛卡尔坐标系绝对位置的指令
	MCircle	笛卡尔坐标系下的整圆插补指令
	MSpline	样条曲线插补运动指令
	waitpos	等待运动脉冲发送完成指令（运动 DSP 发送脉冲完成指令）
	waitrealpos	机器人运动到位（停稳）指令（伺服响应脉冲到位指令）
	MAccept	线程获取与运动相关的使用权：运动指令、设置用户、设置工具
	MRelease	线程释放与运动相关的使用权：运动指令、设置用户、设置工具
运动参数设置指令	AccJ	设置加速度比例指令，影响 MovJ、MovJR、MovP、MovPR、MArchP 指令的加速时间
	DecJ	设置减速度比例指令，影响 MovJ、MovJR、MovP、MovPR、MArchP 指令的减速时间
	SpdJ	设置速度比例指令，影响 MovJ、MovJR、MovP、MovPR、MArchP 指令的运行速度
	JerkJ	设置加加速度比例指令，影响 MovJ、MovJR、MovP、MovPR、MArchP 指令的运行速度
	AccL	设置插补运动指令加速度，影响 MovL、MovLR、MArc、MCircle 指令的加速时间。单位 mm/
	DecL	设置插补运动指令减速度，影响 MovL、MovLR、MArc、MCircle 指令的减速时间。单位 mm/
	SpdL	设置插补运动指令速度，影响 MovL、MovLR、MArc、MCircle 指令的运行速度。单位 mm/s
	JerkL	设置插补运动指令加加速度，影响 MovL、MovLR、MArc、MCircle 指令的加速时间。单位 mm/
	AccC	设置插补姿态运动的加速度，影响 MovL、MovLR、MArc、MCircle 指令的加速时间。单位度/
	SpdC	设置插补姿态运动指令速度，影响 MovL、MovLR、MArc、MCircle 指令

		的运行速度。单位度/s
	JerkC	设置插补姿态运动指令加加速度，影响 MovL、MovLR、MArc、MCircle 指令的加速度变化时间。单位度/
程序管理指令	Delay	延时指令。单位：毫秒
	Exit	程序运行终止
	Pause	暂停程序运行
一般指令	X	创建指定 X 轴笛卡尔坐标绝对位置点的指令
	Y	创建指定 Y 轴笛卡尔坐标绝对位置点的指令
	Z	创建指定 Z 轴笛卡尔坐标绝对位置点的指令
	C	创建指定 C 轴笛卡尔坐标绝对位置点的指令
	XYZC	创建指定 XYZC 轴笛卡尔坐标绝对位置点的指令
输入/输出指令	DI	读取输入端口状态
	DO	输出端口打开或关闭操作
	WDI	读取输入端口状态,直到等待某一信号有效，则继续执行后续的程序
	WDO	读取输出端口状态,直到等待某一信号有效，则继续执行后续的程序
	MO	辅助输出打开或关闭操作
	WMO	读取辅助输出端口状态,直到等待某一信号有效，则继续执行后续的程序
坐标系指令	SetU	设置机器人当前用户坐标系
	WrU	修改用户坐标系的数据
	RdU	读取目标用户里的值（X\Y\Z\C）
	CacU	两点法新建用户坐标系
	U2U	用户（0~9）之间坐标相互转换
	SetT	设置机器人当前工具坐标系
	WrT	修改工具坐标系的数据
	RdT	读取目标工具里的值（X\Y\Z\C）
	V2Tool	根据视觉坐标计算工具坐标

	CacT	两点法新建工具坐标系
	Getpos	获取当前位置笛卡尔坐标以及关节坐标
	encoderget	获取编码器的脉冲值
	Targetok	判断目标位置是否为机器人可到达点位
	SetI	设置当前最佳加速度负载惯量使用组别（0~4）
码垛指令	Set_Plt	设置码垛数指令
	Get_Plt	取码垛数据点指令
	GetPLTPos	获取码盘是否码满、码盘当前码垛的个数以及码垛点位置信息（适用于配置码垛）
	ResetPLT	重置码垛个数
	InitPlts	加载配置码垛参数
	Set_ArcPlt	圆形码垛设置
	Get_ArcPlt	获取圆形码垛当前位置
伺服管理指令	MotOn	伺服所有轴上电使能
	MotOff	关闭伺服所有轴使能
	DragMode	AR 程序实现拖拽功能
串口通信指令	SendCom	往 RS232 串口发送数据
	RecCom	从 RS232 串口接收数据
	ClsCom	关闭串口
	InitCom	初始化串口
	ClrCom	清除 RS232/RS485 串口缓存
RAM 地址读写指令	publicread	读取全局表（RAM 数据表）数据值
	publicwrite	写入数据值到全局表（RAM 数据表）
网络通信指令	InitNet	网络通讯初始化
	SendNet	机器人发送字符串数据给外部设备（例如 PLC、视觉）
	CCDtrigger	触发视觉拍照
	CCDrecv	接收外部设备（例如视觉）返回的坐标

		数据
	ReadNet	接收外部设备（例如 PLC、视觉）发送的字符串类型数据
传送带指令	CnvQueInit	传送带跟随参数初始化
	CnvQueAdd	将坐标数据存入缓存队列待跟随
	CnvQueGet	监控物体在传送带上的位置
	CnvQueDel	删除位置抓取队列中的首元素
	MissQueAdd	超出抓取区未抓取的物体位置信息存入到丢失队列
	CnvSynCatch	执行同步跟随
	SynOver	结束同步跟随
	CnvFastCatch	执行快速抓取，没有同步过程
	LockExcoder	传送带软触发单次记录编码器值并下发到队列
调试指令	print	打印输出用户调试数据
	Error	终止程序运行，并输出错误信息
共享点位操作指令	Point	调用点位表中的点位
	new	将要更新的点位写入到共享点位表里
	teach	将当前点位示教到共享点位表里
系统指令	syswork	设置系统工作状态
	sysstate	读取系统状态
	sysrate	设置全局速度倍率
	systime	获取系统时钟
Modbus 主站读写指令	ReadRegW	从从站指定的 16 位字地址中读取整型数据
	ReadRegDW	从从站指定的 32 位双字地址中读取数据
	WriteRegW	往从站指定的 16 位字地址中写入整型数据
	WriteRegDW	往从站指定的 32 位双字地址中写入数据
	MsetIport	设置从站的 IP 地址和端口号
文件操作指令	io.open	打开文件

	io.type	检测一个文件句柄是否可用
	file:write	写数据到文件中
	file:read	按指定的格式读取一个文件
	file:seek	文件定位，设置和获取当前文件位置
	file:lines	逐行获取文件中的内容
	file:close	关闭文件
队列操作指令	qexist	判断队列是否存在
	qcreate	创建队列
	qpush	往队列中写入数据
	qpop	删除队列中的首元素
	qfront	取队列中的首元素
	qpopfront	取队列中的首元素然后从队列中删除
	qempty	判断队列是否为空
	qsize	队列大小
	qdestroy	删除队列
其它指令	dofile	同一个工程中不同 AR 程序的加载

3.1.1 算术运算符

指令符号	指令说明
+	加法运算
-	减法运算
*	乘法运算
/	除法运算
//	取整除法，即得到不大于结果的最大整数值
%	取余除法
^	指数运算
-	取负运算
~	异或运算
&	与运算
|	或运算
~	取反运算
<<	左移运算
>>	右移运算

算术运算符提供了各种实数的运算功能，整型数据的位运算功能等。

“+”加法运算还可以实现两个点位数据的加法运算。

举例说明:

```
local point1=p1
local point2=p2
```

注：自定义的数据点位不能直接相加。能直接相加的数据点位，必须是机器人系统定义的数据点位（p0~p2999）。

3.1.2 关系运算符

指令符号	指令说明
==	等于
~=	不等于
<=	小于等于
>=	大于等于
<	小于
>	大于

关系运算符用在流程控制语句的条件判断中。

举例说明：

```
local    a =10
local    b = 20
if a == b then
……
end
if a < b then
……
end
if a < 30 then
……
end
```

== 等号可以比较字符串类型变量。

```
local    a = “ROBOT”
if    a == “ROBOT”    then
……
end
```

3.1.3 逻辑运算符

指令符号	指令说明
or	逻辑或
not	逻辑非
and	逻辑与
false	假（注：nil 也为假）
true	真

逻辑运算符用在流程控制语句的条件判断中。

举例说明：

```
if    5 or 6 then      --真
……
end

local    a = 30
local    b = true
if a and b then      --真
……
end

local a = 0
local b = 1
if a and b then      --真
…..
end

local a = 0
local b = nil
if a and b then      --假
……
end
```

◆ 注意：AR 语言中，只有 nil 和 false 值为假，其它值都为真，包括 0 也为真。

3.1.4 一般符号

指令符号	指令说明
#	求 table 数组中元素的个数或计算字符串的长度
=	赋值操作
--	单行注释
--[[	多行注释开始行
--]]	多行注释结束行

( )	用于函数定义、调用，表达式运算
{ }	用于定义 table 数组
[ ]	table 数组元素操作符
::	定义 goto 指令的跳转位置标签
;	语句结束符，可以省略不写
,	用于函数参数定义、调用，多变量赋值，table 数组定义等
.	用于 table 数组元素访问
..	字符串连接符
…	定义函数可变参数项

举例说明：

```
1.  print(#("123,456"))          --返回值为 7（字符串）
    print(#{123,456})            --返回值为 2（数组）
2.  local a,b,c = 1,2,3          --用,符号写多变量赋值语句
3.  p.x                          --用.符号访问数组元素
4.  local a={10,20,30}           --用{}符号定义数组
5.  a[1]                         --用[]符号访问数组元素
6.  ::lab::                      --用::符号定义 goto 指令跳转位置标签
```

◆ 注意：table 类型数组变量下标是从 1 开始，如 a[1]的值为 10。

3.1.5 一般关键字

指令符号	指令说明
break	跳出 for、while、repeat 循环
local	定义局部变量
global.*	同一个工程中，多个 CPU 之间变量的共用
nil	变量为空值
return	函数调用返回

3.1.5.1 break

break 语句用来退出当前循环（for、repeat、while）。在循环外部不可以使用。

```
local i = 1
while true do
    if a[i]=v then
        break
```

```
        end
        Delay(10)
        i = i + 1
    end
```

3.1.5.2 local

local 符号用来声明局部变量

举例说明：

```
local   a                                --声明局部变量 a ，值为 nil
local   b={10,10}                        --声明 table 数组
local   b={{10,20},{30,40}}              --声明 table 二维数组
local   a="ROBOT"--声明字符串变量 a
```

- 介绍局部变量之前，首先来介绍一下 lua 语言中，代码块内全局变量是如何定义的；全局变量不需要声明，给一个变量赋值后即创建了这个全局变量，访问一个没有初始化的全局变量也不会出错，只不过得到的结果是：nil

```
print(b)    -->nil
b = 10
print(b)    -->10
--如果想删除一个全局变量，只需要将变量赋值为 nil;这样变量 b 就好像从来没被使用过一样，换句话说，当且仅当一个变量不等于 nil 时，这个变量存在。
b =nil
print(b)    -->nil
```

- 使用 local 创建一个局部变量，与全局变量不同，局部变量只在被声明的那个代码块内有效。代码块：指一个控制结构内，一个函数体，或者一个 chunk（变量被声明的那个文件或者文本串）。

```
x = 10
local i = 1                -- local to the chunk
while i<=x do
    local x = i*2      -- local to the "while" body
    print(x)           --> 2, 4, 6, 8, ...
    i = i + 1
end
if i > 20 then
    local x        -- local to the "then" body
    x = 20
    print(x +2)        -->22(the local one)
else
    print(x)           --> 10 (the global one)
end
print(x)               --> 10 (the global one)
```

3.1.5.3 global.*

根据实际应用工艺需求，一个工程须包含多个线程。全局变量 global* 可以实现多线程的共享通信，即全局变量（global.*）可解决多个线程之间共用同一个变量的问题。

使用说明：

- **使用时请先赋初值，避免开机时已被定义或者在使用过后清除赋值；并只在一个线程内赋值；由于多线程有优先级的问题,在使用全局变量时要在一个线程中初始化完成标志，其它线程等到初始化完成在使用全局变量; [请查阅例 1]**
- **在各个 CPU 中，公共变量定义为 global.*的形式，例如：global.var, global.a,global.b;**
- **全局变量里可嵌套一级数组，例如： global.a={1,2}则 global.a[1]=1；例如：global.a ={x=100,y=200}**

global.a ={x=300,y=100} **-->global.a.x=300,global.a.y=100**

global.a={x=400} **-->global.a.x=400,global.a.y=nil**

- 全局变量不能有二级 **table** ，例如：**global.v = {p1}** ，**global.v = {{11},22}** 会出错；
- 运动指令可以直接调用，例如： **global.v = p1** **MovP(global.v)** 正确；**global.v = {x=300,y=10,z=-10}** **MovP(global.v)** 正确；
- 全局变量最大支持 **200** 个,例如 **global.v1 ... global.v200**，但且当全局变量元素 **v** 为 **table** 时，最多允许 **20** 个 **table** 变量；
- 当全局变量赋值为 **table** 类型后不允许再修改为其它类型例如：先赋值 **global.v = p1** 后再赋值 **global.v = 10** 会出错，而其它类型可以任意使用如 **global.v = 10** 为 **numble** 类型在赋值为字符串类型 **global.v = "test"** 是允许的；
- 全局变量在上电期间是保持的，只在开机时初始化一次；全局变量不等运动状态是否结束；

  注：针对多维数组，可对整体数组赋值，不可对单个数组元素赋值。
- 全局变量仅限于一个工程中多个线程之间变量的共用，单个 **CPU** 中全局变量的使用可参考 **AR** 编程手册中的 **local** 指令的使用；

```
--开始初始化全局变量
global.a = nil
global.b = 0
global.c = "string"
global.d = false
global.e = p0 --点类型数据
--[[
由于多线程有优先级的问题,在使用全局变量时要在一个线程中初始化完成标志，
其它线程等到初始化完成在使用全局变量
--]]
global.init = true    --初始化完成

function main()
    print("run\n");
    print(global.init)
    --当前 CPU 打印全局变量
    print(global.a)
    Delay(1000)
    print(global.b)
    Delay(1000)
    print(global.c)
    Delay(1000)
    print(global.d)
    Delay(1000)
    print(global.e.x,global.e.y,global.e.z,global.e.c,global.e.h)
   Delay(1000)

    while true do
        global.b = global.b + 1    --修改全局变量 b
        Delay(1000)
    end
end
```

线程 1

```
function main()
    print("run\n");
    repeat
        Delay(5)
    until global.init == true    --等待初始化完成
    global.init = nil            --使用后释放
    print("全局变量初始化完成")
    while true do
        print(global.a)
        Delay(1000)
        print(global.b)
        Delay(1000)
        if global.b > 10 then
            global.b = 0
            print("global.b 大于 10 清 0")
        end
        print(global.c)
        Delay(1000)
        print(global.d)
        Delay(1000)
        print(global.e.x,global.e.y,global.e.z,global.e.c,global.e.h)
        Delay(1000)
    end
end
```

线程 2

3.1.6 基本函数库

函数名	描述	示例	结果
math.pi	圆周率	math.pi	3.1415926

math.abs	取绝对值	math.abs(-2012)	2012
math.ceil	向上取整	math.ceil(9.1)	10
math.floor	向下取整	math.floor(9.9)	9
math.max	取参数最大值	math.max(2,4,6,8)	8
math.min	取参数最小值	math.min(2,4,6,8)	2
math.pow	计算 x 的 y 次幂	math.pow(2,16)	65536
math.sqrt	开平方	math.sqrt(65536)	256
math.fmod	取模	math.fmod(14,5)	4
math.modf	取整数和小数部分	math.modf(20.12)	20　0.12
math.randomseed	设随机数种子	math.randomseed(os.time())	
math.random	随机坐标系	math.random(5,90)	5~90
math.rad	角度转弧度	math.rad(180)	3.1415926
math.deg	弧度转角度	math.deg(math.pi)	180
math.exp	e 的 x 次方	math.exp(4)	54.5981
math.log	计算 x 的自然对数	math.log(54.5981)	4
math.log10	计算 10 为底，x 的对数	math.log10(1000)	3
math.frexp	将参数拆成 x*（2^y）的形式	math.frexp(160)	0.6258
math.ldexp	计算 x*(2^y)	math.ldexp(0.625,8)	160
math.sin	正弦	math.sin(math.rad(30))	0.5
math.cos	余弦	math.cos(math.rad(60))	0.5
math.tan	正切	math.tan(math.rad(45))	1
math.asin	反正弦	math.deg(math.asin(0.5))	30
math.acos	反余弦	math.deg(math.acos(0.5))	60

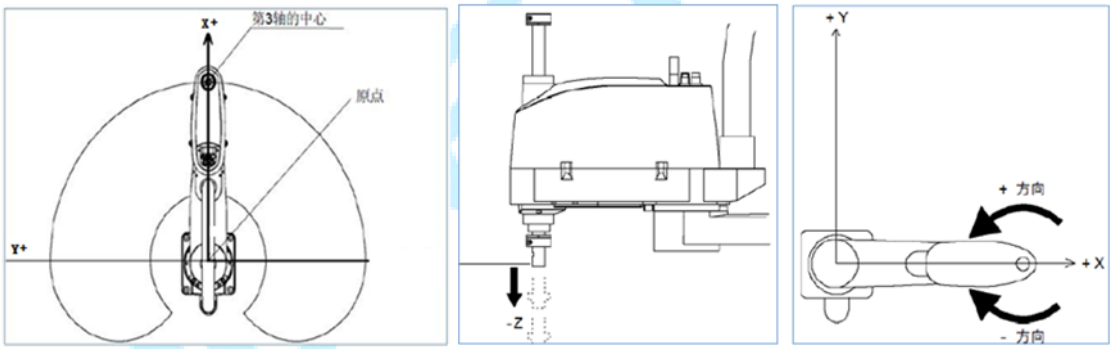
math.atan	反正切	math.deg(math.atan(1))	45

3.1.7 机器人轴定义

指令符号	指令说明
AX	笛卡尔坐标系 X 轴号
AY	笛卡尔坐标系 Y 轴号
AZ	笛卡尔坐标系 Z 轴号
AC	笛卡尔坐标系 C 轴号
J1	J1 关节
J2	J2 关节
J3	J3 关节
J4	J4 关节

轴号是全局变量，作为运动指令的一些参数使用，用户也不能重新定义。

◆ 笛卡尔坐标 X/Y/Z/C 轴方向定义示意图：

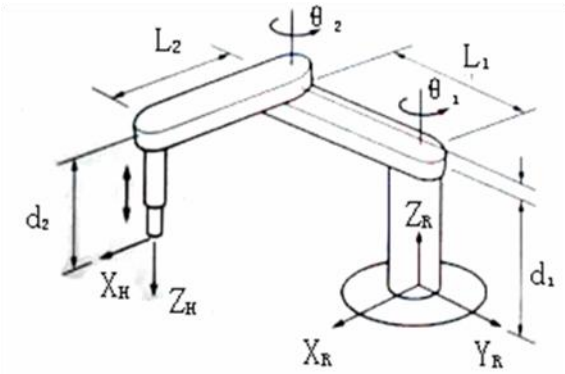


笛卡尔-XY(俯视图)　笛卡尔-Z(侧视图)　笛卡尔-C(俯视图)

◆ 关节坐标 J1/J2/J3/J4 轴方向定义示意图：

3个旋转关节：J1，J2，J4；(俯视)逆时针旋转为正方向，顺时针负方向。

1个移动关节：J3；(侧视) 上为正，下为负。



3.1.8 机器人全局变量

指令符号	指令说明
ON	打开状态
OFF	关闭状态
p0~p2999	机器人默认点名称

全局变量为系统定义好的变量，有具体的意义，用户不能再重新定义或赋值，否则可能运行出错。

点位变量 p0~p2999 为 table 类型的点变量，定义如下：

```
local    p={x=VALUE1,y=VALUE2,z=VALUE3,c=VALUE4,h=VALUE5}
```

程序中，用户可以以 p.x p.y 的方式访问点的数据值。

```
local   a = p1                    -- a 为 p1 的引用
a.x = 10                          --那么 p1.x 的值也变为 10
locala = #p1                     --拷贝 p1 的值到 a
a.x=10                            --p1.x 保持原来的值
```

3.1.9 过程控制指令

指令符号	指令说明
if then else elseif end	if 条件分支指令
while do end	while 循环控制指令
for do end	for 循环控制指令
repeat until	repeat 循环控制指令
goto	goto 无条件跳转指令
functionend	用户函数定义指令

3.1.9.1 if 条件分支指令

使用说明: if then else elseif end

语法说明：

Case1	Case2	Case3

if conditionsthen
then-part
end
	if conditions then
	then-part
	else
	else-part
	end
		if conditionsthen
		then-part
		elseif
		conditionsthen
		elseif-part
		else
		else-part
		end

Conditions 表示控制语句的条件，如果为 true 则条件成立，part 为执行的代码部分。

Conditions 条件可以是常量，变量，表达式，函数调用等，程序只判断最终结果为 false 或 true 去选择执行程序。

3.1.9.2 while 循环指令

使用说明: while do end

语法说明:

```
while condition do
    statements
end
```

conditions 表示控制条件，如果为 true 则条件成立，statements 为执行的代码部分，如果为 false 则条件不成立，不执行 statements 代码部分。

```
a = 15
while a>10 do          --条件判断，如果为 true，则继续执行 a = a-1
a = a-1
end
```

3.1.9.3 for 循环指令

使用说明: for….. do…..end

语法说明:

```
for var=exp1,exp2,exp3 do
    loop-part
end
```

for 将用 exp3 作为 step（步进值）从 exp1（初始值）到 exp2（终止值），执行 loop-part（循环体）。其中 exp3 可以省略，默认 step=1

exp 是一个表达式，可以是数值常量、变量或函数调用的返回值、表达式运算。

有几点需要注意：

1. 三个表达式只会被计算一次，并且是在循环开始前。

```
for i=1,f(x) do        --调用 f(x)函数一次，函数返回值作为循环的终止值
    print(i)
end
fori=10,1,-1 do
    print(i)
end
```

2. 控制变量 var 是局部变量自动被声明,并且只在循环内有效.

```
for i=1,10do
    print(i)
end
max =i                          --错误引用 i，i 是局部变量
```

如果需要保留控制变量的值，需要在循环中将其保存

```
local found = nil
for i=1,a.n do
    if a[i] == value then
        found= i
        break
    end
end
print(found)
```

3. 循环过程中不要改变控制变量的值，那样做的结果是不可预知的。如果要退出循环，使用 break 语句。

3.1.9.4 repeat–until 循环指令

使用说明: repeat-until

语法说明:

```
repeat
    statements
untilconditions
```

while 语句大致相同，只是循环部分 statements 的执行先后不一样，repeat-until 的循环体先执行在判断循环条件，而 while 语句是先判断循环条件。

3.1.9.5 goto 无条件跳转指令

使用说明 goto

语法说明 goto lab_name

其中 lab_name 用户定义的文件行跳转位置名称，字符串类型，如果没有定义会出错。

```
local a = 0
::lab::                          --定义跳转位置名称 lab
MovL(p0)
MovL(p1)
a = a + 1
print("跳转次数:",a)
goto lab                         --跳转到示例 2  循环执行
```

◆ 注意：goto 指令不能从一个函数跳转到另外一个函数，lab_name 跳转名称不能重复定义。

3.1.9.6 函数指令

使用说明 function …… end

语法说明 function func_name (arguments-list)

```
functionfunc_name(arguments-list)
    statements-list        --函数体部分
end
```

func_name 是函数名字，根据 AR 语言命名规则进行定义，且函数名不能重复，否则会出错。

arguments-list 为函数的参数列表，列表中的参数可以是任何数据类型的变量，当有多个参数时，通过”,”分隔。函数还可以没有参数，在定义的时候参数列表为空即可，但是括号不能省略。

statements-list 为函数体部分，函数体就是实现该函数功能的具体程序细节，函数体结束部分可以有返回值，也可以没有返回值，如果有返回值则通过**关键字 return** 说明。

例如：定义一个加法功能函数:

```
functionadd (a,b)
    return a+b
end
add(1,2)   --函数相加并返回结果 3
```

返回多个值，求相加与相乘函数

```
functionaddmul(a,b)
    return a+b,a*b
end
addmul(2,5)                    --函数相加相乘，返回 7   10
```

注意 1：

function main 主函数的使用方法，当 AR 程序中包含有 function main 主函数是，子函数既可以放在 function main 函数的前面定义也可以放在后面定义，例如：

```
functionaddmul(a,b)
    return a+b,a*b
end
function main()
    while true do
        addmul(2,5)            --函数相加相乘，返回 7   10
    end
end
```

```
function main()
    while true do
        addmul(2,5)            --函数相加相乘，返回 7   10
    end
end
functionaddmul(a,b)
    return a+b,a*b
end
```

注意 2：

当 AR 程序中没有主函数时，子函数一定要放在调用前定义，否则程序出错，例如：

```
functionaddmul(a,b)
    return a+b,a*b
end

while true do
    print("开始运行")
addmul(2,5)
end
```

运行正常

```
while true do
    print("开始运行")
addmul(2,5)
end
functionaddmul(a,b)
```

运行异常

3.1.10 字符串操作

指令符号	指令说明
string.find	函数用于在一个给定的目标字符串中搜索一个模式，返回该模式所在的位置
string.match	用于在一个字符串中搜索一种模式，返回的是目标字符串中与模式相匹配的那部分字符串
string.sub	函数截取字符串 s 的从第 i 个字符到第 j 个字符之间的串
string.gsub	将目标字符串中所有出现模式的地方替换为目标字符串
string.char,string.byte	用于转换字符和对应的数字之间值

string.format	用来对字符串格式化，与 C 语言中的 printf()函数类似
string.len	用来计算字符串的长度
tonumber	将数字组成的字符串 string 变量转化成数字 number 类型
tostring	将布尔类型和数值类型转换为字符串类型

1. **string.find()**函数用于在一个给定的目标字符串中搜索一个模式。最简单的模式就是一个单词，它只会匹配与自己完全相同的拷贝。当 find 找到一个模式后，它会返回两个值:匹配到的起始索引和结尾索引；如果没有找到任何匹配，它就返回 nil。示例代码：

```
local str = “Hello World”
local i,j = string.find(str, “Hello”)     --返回 Hello 在 str 中的起始位置和终止位置
print(i,j)     --> 1    5
```

2. **string.match()**与 string.find()非常相似，它也是用于在一个字符串中搜索一种模式。区别在于，string.match 返回的是目标字符串中与模式相匹配的那部分字符串，并不是该模式所在的位置。示例代码：

```
local str = “Hello12345World”
local subStr = string.match(str, “%d+”)
print(subStr)     --> 1 2 3 4 5

local i,j = string.find(str, “%d+”)
subStr = string.sub(str,i,j)
print(subStr)     --> 1 2 3 4 5
```

3. **string.sub(s,i,j)**函数截取字符串 s 的从第 i 个字符到第 j 个字符之间的串. Lua 中，字符串的第一个字符索引从 1 开始。你也可以使用负索引，负索引从字符串的结尾向前计数：-1 指向最后一个字符，-2 指向倒数第二个，以此类推。示例代码：

```
s = “[in brackets]”
b = string.sub(s,2,-2)
print(b)     --> in brackets
```

4. **string.gsub(s, pattern,reps)**有三个参数，给定字符串，匹配模式和替代字符串。作用就是将所有符合匹配模式的地方都替换成替代字符串。并返回替换后的字符串，以及替换次数。示例代码：

```
s = string.gsub("Lua is cute", "cute", "great")
print(s) --> Lua is great
```

5. **string.char() &&string.byte()** 用于转换字符和对应的数字之间值。示例代码：

```
i = 97
print(string.char(i, i+1, i+2)) --> abc
print(string.byte("abc"))          --> 97
print(string.byte("abc"), -2)     -->97    -2
```

6. **string.format()**的功能就是格式化一个字符串。第一个参数为字符串格式，后面的参数可以任意多个，用于填充第一个参数中的格式控制符，最后返回完整的格式化后的字符串。示例代码:

```
str = string.format("%0.2f",34.2344)
print(str)      --> 34.23
str = string.format("0x%08X",348)
print(str)      --> 0x0000015C
```

7. **string.len()**函数的功能是计算字符串的长度。

```
local str = "Hello world"
n = string.len(str)
print("n=",n)      --> 11
```

8. **tonumber()** 函数的功能是将数字组成的字符串变量转化成数字类型，如果要转换的 string 不能被转为 number，它返回 nil。示例代码：

```
1.  Str1 = "12"
A = tonumber(Str1)
      print(A)          --12
2.  Str2="123,456"
    B=tonumber(Str2)
```

```
    print(B)        --nil
3.  Str3 = “123,a”
    C=tonumber(Str3)
    print(C)        --nil
```

9. **tostring()** 函数的功能是将布尔类型和数值类型转换成字符串类型,如果要转换的数据不能被转为 string，它返回 nil。示例代码：

注：在打印界面无法显示字符串。

```
1.  local bVar = false
    print(tostring(bVar)) --  输出"false"

2.  local num1 = 10
    local num2 = 10.0
    local num3 = 10.03
    print(tostring(num1)) --输出"10"
    print(tostring(num2)) --输出"10.0"
    print(tostring(num3)) --输出"10.03"
```

3.1.11 运动指令

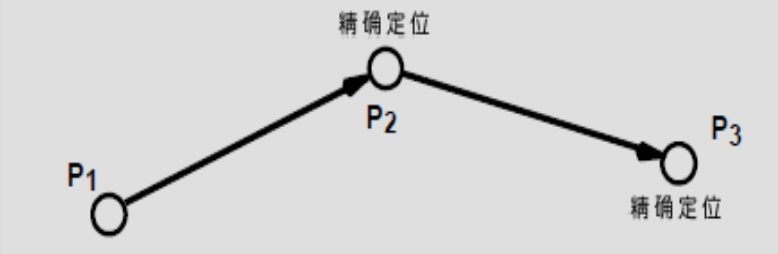
指令符号	指令说明
MovL	直线方式运行到笛卡尔坐标系绝对位置的指令
MovLR	直线方式运行到笛卡尔坐标系相对位置的指令
MovP	点到点方式运行到笛卡尔坐标系绝对位置的指令
MovPR	点到点方式运行到笛卡尔坐标系相对位置的指令
MovJ	控制各个关节移动到目标角度的指令
MArchP	点到点方式控制机器人进行拱形移动的指令
MArc	从当前位置圆弧插补到笛卡尔坐标系绝对位置的指令
MCircle	笛卡尔坐标系下的整圆插补指令
MSpline	样条曲线插补运动指令
waitpos	等待运动脉冲发送完成指令（运动 DSP 发送脉冲完成

	指令）
waitealpos	机器人运动到位（稳停）指令（伺服响应脉冲到位指令）
MAccept	线程获取与运动相关的使用权：运动指令、设置用户、设置工具
MRelease	线程释放与运动相关的使用权：运动指令、设置用户、设置工具

MovL

使用说明	以直线的方式运动到笛卡尔坐标系下的目标绝对位置
语法说明	STA = MovL(A, “CP=20Acc=20 Dec=20 Spd=100 AccC=20 SpdC=20 Stop=0 I=0 In=10 ON/ OFF”)
参数说明	该指令一共两个参数，第一个参数 A 为目标点，第二个参数为可选参数、省略时系统默认全局状态值 **输入参数说明**

A	笛卡尔坐标目标位置。可以是点的名称 p1~p9999，也可以是点的索引号 1~9999。可选参数，可以指定运动到目标位置的各项参数
CP	可选参数，说明运动到目标点时是否平滑过渡，范围为 0~100
Acc	可选参数，指定运动到目标位置的加速度，单位为 mm/s^2
Dec	可选参数，指定运动到目标位置的减速度，单位为 mm/s^2
Spd	可选参数，指定运动到目标位置的速度，单位为 mm/s
AccC	可选参数，指定运动到目标位置的姿态加速度，单位度/s^2
SpdC	可选参数，指定运动到目标位置的姿态速度，单位

		度/s
	Stop	可选参数，用来描述当第三轴的电流超过设定阈值或输入信号被触发时，机器人停止的两种方式（默认是减速停 Stop=0）： Stop = 0 减速停（默认） Stop = 1 不减速直接停（不减速直接停仅适用于机器人在超低速运行时使用，否则会出现速度偏差超限等伺服报警）
	I	可选参数，第三轴电流设定阈值，单位 mA
	In	可选参数，输入检测信号端口，此参数与 ON or OFF 搭配使用
	ON/OFF	ON：打开；OFF：关闭
	返回值说明	
	STA	0：指令运行正常
		1：输入信号被触发或第三轴电流超过设定的阈值
示意图		
举例说明	1. MovL(p1) --机器人以直线的方式从当前位置运动到 p1 目标点 2. MovL(10) --机器人以直线的方式从当前位置运动到 p10 目标点 3. MovL(10, "Acc=100Spd=1000") --机器人以直线的方式从当前位置运动到目标点 p10，其中加速度为 100mm/s2，速度为 1000mm/s 4. MovL(p20, "CP=20") --机器人以直线的方式运动到 p20 位置，其中目标位置 p20 以平滑度 20 过渡 5. MovL(p20, "In=10 ON") --机器人以直线运动到 p20 目标位置，在运动过程中如果检测输入信号 In10 打开则当前指令运行减速停止。 6. Sta = MovL(p20, "Stop = 0 In=10 ON") --机器人以直线运动	

到 p20 目标位置，在运动过程中如果检测输入信号 In10 打开则当前指令运行减速停止。

```
if    sta == 0 then
      print("未检测到输入信号 10，运动正常")
elseif sta == 1 then
      print("检测到输入信号 10")
      Error("报警退出")
 end
```

7. MovL(p20, "Stop = 1 In=10 ON") --机器人以直线运动到 p20 目标位置，在运动过程中如果检测输入信号 In10 打开则当前指令运行不减速直接停止。
8. MovL(p3, "I = 500") --机器人以直线的方式运动到 p3 目标位置，在运动过程中如果检测到第三轴的电流超过 500mA，当前运行指令减速停止。
9. MovL(p3, "Stop = 0 I = 500") --机器人以直线的方式运动到 p3 目标位置，在运动过程中如果检测到第三轴的电流超过 500mA，当前运行指令减速停止。

```
if    sta == 0 then
      print("未检测第三轴电流超过设定阈值，运动正常")
elseif sta == 1 then
      print("检测第三轴电流超过设定阈值")
end
```

10. MovL(p3, "Stop = 1 I = 500") --机器人以直线的方式运动到 p3 目标位置，在运动过程中如果检测到第三轴的电流超过 500mA，当前运行指令不减速直接停止。
11. MovL(p10+Z(-10)) --机器人以直线的方式从当前位置运动到 p10 点下方 10mm 的位置
12. MovL(p11+Z(10)) --机器人以直线的方式从当前位置运动到 p10 点上方 10mm 的位置
13. local pos = {x=300,y=100,z=-50,c=60} --用户定义 pos 点变量

    MovL(pos+X(-10)) --机器人以直线的方式从当前位置运动到相对于 pos 点 X 负方向 10mm 的位置，运动的手系与当前机器人的手系一致
14. MovL(p12+Y(10)) --机器人以直线的方式从当前位置运动到相对于 p12 点 Y 正方向 10mm 的位置

注意：

- 可选参数编程时可以根据需要是否设置，其中 Acc、Spd 省略时，系统默认全局值，当设置 In 选项触发停止时，参数 ON/OFF 才会有意义，且只能二选一设置信号有效停止或无效停止。
- 当使用 MovL 走直线偏移（例如，例 7、8、9、10）时，一定要用正号（+）连接轴实现偏移，偏移的正负取决于轴括号内的数字的正负。
- 大写 X 、Y、Z、C 分别表示 X 轴、Y 轴、Z 轴、C 轴方向的轴偏移

MovLR	
使用说明	以直线的方式运动到笛卡尔坐标系下的目标相对位置
语法说明	STA = MovLR(A,B,“CP=20 Acc=20 Dec=20 Spd=100 AccC=20 SpdC=20 Stop= 0 I=0 In=10 ON/ OFF ")
参数说明	该指令一共三个参数，第一个为笛卡尔坐标轴号，第二个参数为移动的相对距离，第三个参数为可选参数、省略时系统默认全局状态值 **输入参数说明**

A	AX,AY,AZ,AC 各笛卡尔坐标轴号
B	各轴移动的相对距离
CP	可选参数，说明运动到目标点时是否平滑过渡，范围为 0~100
Acc	可选参数，指定运动到目标位置的直线加速度，单位为 mm/s^2
Dec	可选参数，指定运动到目标位置的直线减速度，单位为 mm/s^2
Spd	可选参数，指定运动到目标位置的直线速度，单位 mm/s
AccC	可选参数，指定运动到目标位置的姿态加速度，单位度/s^2
SpdC	可选参数，指定运动到目标位置的姿态速度，单位度/s

	Stop	可选参数，用来描述当第三轴的电流超过设定阈值或输入信号被触发时，机器人停止的两种方式（默认是减速停 Stop=0）： Stop = 0　减速停（默认） Stop = 1　不减速直接停（不减速直接停仅适用于机器人在超低速运行时使用，否则会出现速度偏差超限等伺服报警）
	I	可选参数，第三轴电流设定阈值，单位 mA
	In	可选参数，输入检测信号端口，此参数与 ON or OFF 搭配使用
	ON/OFF	ON：打开；OFF：关闭
	返回值说明	
	STA	0：指令运动正常
		1：输入信号被触发或第三轴电流超过设定的阈值
举例说明	1. MovLR(AX,10)　--从当前位置往 X 轴正方向移动 10mm 的距离 2. MovLR(AC,10)　--从当前位置往 C 轴正方向移动 10 度的角度 3. MovLR(AY,10)　--从当前位置往 Y 轴正方向移动 10mm 的距离 4. MovLR(AZ,-10)　--从当前位置往 Z 轴负方向移动 10mm 的距离	

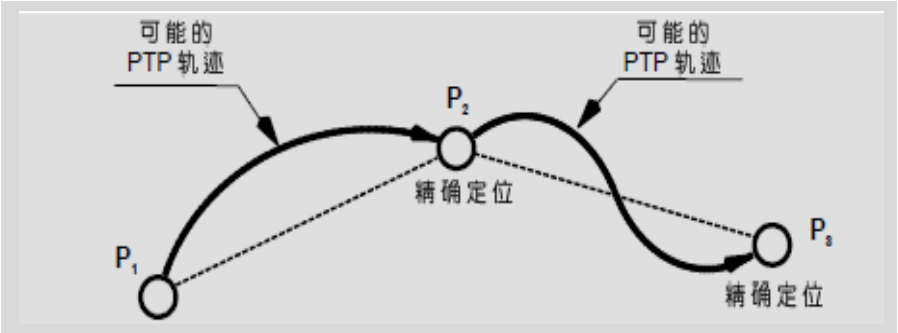
➢ 输入检测和电流检测功能的使用可参考 MovL 指令的例子 5、6、7、8、9、10

MovP	
使用说明	点到点方式移动到笛卡尔坐标系下的目标绝对位置
语法说明	STA = MovP(A,"CP=20 Acc=20Dec=20 Spd=100 Stop=0 I=0 In=10 ON/ OFF”)
参数说明	该指令一共两个参数，第一个参数 A 为目标点，第二个参数为可选参数，省略时系统默认全局状态值 **输入参数说明**

A	笛卡尔坐标目标位置。可以是点的名称 p1~p9999，也可以是点的索引号 1~9999
CP	可选参数，说明运动到目标点时是否平滑过渡，范围 0~100
Acc	可选参数，指定运动到目标位置的加速度比例，范围 1~100
Dec	可选参数，指定运动到目标位置的减速度比例，范围 1~100
Spd	可选参数，指定运动到目标位置的速度比例，范围 1~100
Stop	可选参数，用来描述当第三轴的电流超过设定阈值或输入信号被触发时，机器人停止的两种方式（默认是减速停 Stop=0）： Stop = 0　减速停（默认） Stop = 1　不减速直接停（不减速直接停仅适用于机器人在超低速运行时使用，否则会出现速度偏差超限等伺服报警）
I	可选参数，第三轴电流设定阈值，单位 mA
In	可选参数，输入检测信号端口，此参数与 ON or OFF 搭配使用
ON/OFF	ON：打开；OFF：关闭

返回值说明

STA	0: 指令运动正常
	1：输入信号被触发或第三轴电流超过设定的阈值

示意图	
举例说明	1. MovP(p1) --机器人以点到点的方式从当前位置运动到 p1 目标点 MovP(10) --机器人以点到点方式从当前位置运动到 p10 目标点 2. MovP(10, "Acc=50Spd=50") --机器人以点到点方式从当前位置运动到 p10 目标点，其中加速度为 50%的倍率，速度为 50%的倍率 3. MovP(p20, "CP=20") --机器人以点到点方式运动到 p20 位置，其中目标位置 p20 以平滑度 20 过渡 4. local pos={x=200,y=10,z=-10,c=30,h=0} --用户定义 p 点变量 MovP(pos) --机器人从当前位置以左手系运动到 pos 点目标位置,若 pos 点中的 h 省略不写，则运动的手系默认与当前手系一致 5. MovP(p10+Z(-10)) --机器人以点到点的方式从当前位置运动到 p10 点下方 10mm 的位置 6. MovP(p11+Z(10)) --机器人以点到点的方式从当前位置运动到 p11 点上方 10mm 的位置 7. local pos = {x=300,y=100,z=-50,c=60} --用户定义 pos 点变量 MovP(pos+X(-10)) --机器人以点到点的方式从当前位置运动到相对于 pos 点 X 负方向 10mm 的位置 8. MovP(p12+Y(10)) --机器人以点到点的方式从当前位置运动到相对于 p12 点 Y 正方向 10mm 的位置

注：

- 当使用 MovP 走点位偏移（例如，例 5、6、7、8）时，一定要用正号（+）连接轴实现偏移，偏移的正负取决于轴括号内的数字的正负。
- 大写 X 、Y、Z、C 分别表示 X 轴、Y 轴、Z 轴、C 轴方向的轴偏移
- 输入检测和电流检测功能的使用可参考 MovL 指令的例子 5、6、7、8、9、10

MovPR

使用说明	点到点方式移动到笛卡尔坐标系下的目标相对位置
语法说明	STA = MovPR(A,B," CP=20 Acc=20 Dec=20 Spd=100 Stop=0 I=0 In=10 ON/ OFF ")
参数说明	**输入参数说明**

A	AX,AY,AZ,AC 各笛卡尔坐标轴号
B	各轴移动的相对距离
CP	可选参数，说明运动到目标点时是否平滑过渡，范围 0~100
Acc	可选参数，指定运动到目标位置的加速度比例，范围 1~100
Dec	可选参数，指定运动到目标位置的减速度比例，范围 1~100
Spd	可选参数，指定运动到目标位置的速度比例，范围 1~100
Stop	可选参数，用来描述当第三轴的电流超过设定阈值或输入信号被触发时，机器人停止的两种方式（默认是减速停 Stop=0）： Stop = 0 减速停（默认） Stop = 1 不减速直接停（不减速直接停仅适用于机器人在超低速运行时使用，否则会出现速度偏差超限等伺服报警）
I	可选参数，第三轴电流设定阈值，单位 mA
In	可选参数，输入检测信号端口，此参数与 ON or OFF 搭配使用
ON/OFF	ON：打开；OFF：关闭

返回值说明

STA	0: 指令运动正常
	1：输入信号被触发或第三轴电流超过设定的阈值

举例说明	1. MovPR(AX,10) --从当前位置以点到点的方式往 X 轴正方向移动 10mm 的距离 2. MovPR(AC,10) --从当前位置以点到点的方式往 C 轴正方向移动 10 度的角度

➢ 输入检测和电流检测功能的使用可参考 MovL 指令的例子 5、6、7、8、9、10

MovJ		
使用说明	点到点的方式移动机器人各个关节到指定的角度绝对位置	
语法说明	1. MovJ (A,B," CP=20 Acc=20 Dec=20 Spd=20") 2. MovJ(A,"CP=20 Acc=20 Dec=20 Spd=20")	
参数说明	**用法 1：下表各指令参数说明**	
	A	J1,J2,J3,J4 对应机器人各个关节号
	B	各关节移动的目标角度
	CP	可选参数，说明运动到目标点时是否平滑过渡，范围 0~100
	Acc	可选参数，指定运动到目标位置的加速度比例，范围 1~100
	Dec	可选参数，指定运动到目标位置的减速度比例，范围 1~100
	Spd	可选参数，指定运动到目标位置的速度比例，范围 1~100
	用法 2：下表各指令参数说明	
	A	每个轴的关节目标位置（数组变量）
	CP	可选参数，说明运动到目标点时是否平滑过渡，范围 0~100
	Acc	可选参数，指定运动到目标位置的加速度比例，范围 1~100
	Dec	可选参数，指定运动到目标位置的减速度比例，范围 1~100

	Spd	可选参数，指定运动到目标位置的速度比例，范围 1~100

举例说明	1. MovJ (J1,10) --机器人第一关节移动到 10 度的位置 2. MovJ (J3,-10) --机器人第三关节移动到-10 毫米的位置 3. JointAngle = {x=30,y=40,z=-20,c=30} –目标关节位置 J1=30，J2=40，J3=-20，J4=30 MovJ(JointAngle, “Acc=20 Dec=20 Spd=20”) –关节运动到目标位置点

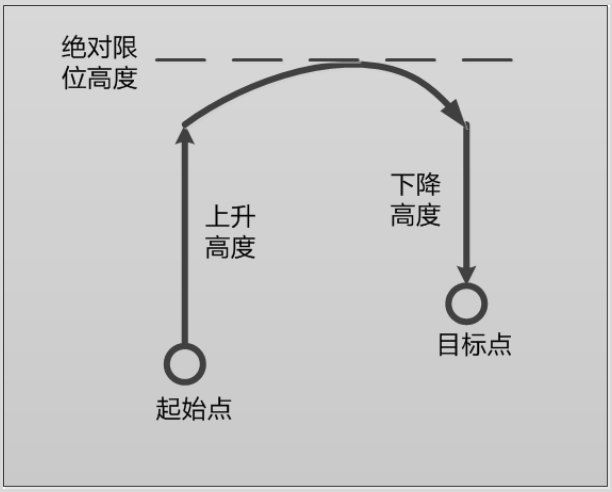
◆ **注意：关节运动时 J3 为毫米单位，其它关节单位为角度。**

MArchP

使用说明	机器人以点到点的方式做拱形运行
语法说明	1. STA = MArchP(A,B,C,D, "CP=20 Acc=20 Dec=20 Spd=100 ZsAcc=20 ZsSpd=80 ZsDec=20 ZeAcc=20 ZeSpd=80 ZeDec=20 Stop=0 I=0 In=10 ON/ OFF”) 2. STA = MArchP(A,B, "CP=20 Acc=20 Dec=20 Spd=100 ZsAcc=20 ZsSpd=80 ZsDec=20 ZeAcc=20 ZeSpd=80 ZeDec=20 Stop=0 I=0 In=10 ON/ OFF")
参数说明	**用法 1：输入参数说明**

A	笛卡尔坐标目标位置。可以是点的名称 p1~p9999，也可以是点的索引号 1~9999
B	Z 轴最高限位绝对位置，单位为毫米
C	Z 轴上升的高度，单位为毫米
D	Z 轴下降的高度，单位为毫米
CP	可选参数，说明运动到目标点时是否平滑过渡，范围 0~100
Acc	可选参数，指定拱形运动水平段的加速度比例，范围 1~100
Dec	可选参数，指定拱形运动水平段的减速度比例，范围 1~100
Spd	可选参数，指定拱形运动水平段的速度比例，范围 1~100

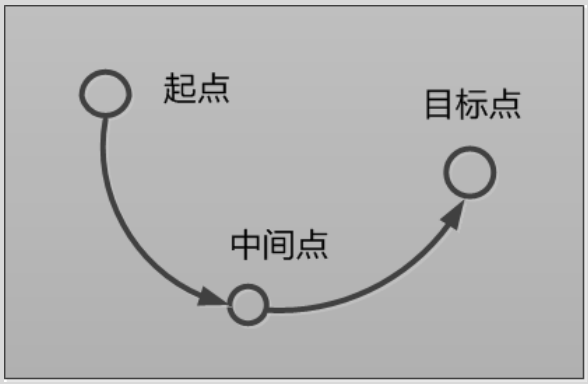
ZsAcc	可选参数，指定拱形运动上升段的加速度比例，范围 1~100
ZsDec	可选参数，指定拱形运动上升段的减速度比例，范围 1~100
ZsSpd	可选参数，指定拱形运动上升段的速度比例，范围 1~100
ZeAcc	可选参数，指定拱形运动下降段的加速度比例，范围 1~100
ZeDec	可选参数，指定拱形运动下降段的减速度比例，范围 1~100
ZeSpd	可选参数，指定拱形运动下降段的速度比例，范围 1~100
Stop	可选参数，用来描述当第三轴的电流超过设定阈值或输入信号被触发时，机器人停止的两种方式（默认是减速停 Stop=0）： Stop = 0 减速停（默认） Stop = 1 不减速直接停（不减速直接停仅适用于机器人在超低速运行时使用，否则会出现速度偏差超限等伺服报警）
I	可选参数，第三轴电流设定阈值，单位 mA
In	可选参数，输入检测信号端口，此参数与 ON or OFF 搭配使用
ON/OFF	ON：打开；OFF：关闭

示意图		
	用法 2：输入参数说明	
	A	笛卡尔坐标目标位置。可以是点的名称 p1~p9999，也可以是点的索引号 1~9999
	B	Z 轴最高限位（绝对位置），也是 Z 轴要实际走到的高度，单位为毫米
	CP	可选参数，说明运动到目标点时是否平滑过渡，范围 0~100
	Acc	可选参数，指定拱形运动的加速度比例，范围 1~100
	Dec	可选参数，指定拱形运动的减速度比例，范围 1~100
	Spd	可选参数，指定拱形运动的速度比例，范围 1~100
	ZsAcc	可选参数，指定拱形运动上升段的加速度比例，范围 1~100
	ZsSpd	可选参数，指定拱形运动上升段的减速度比例，范围 1~100
	ZsDec	可选参数，指定拱形运动上升段的速度比例，范围 1~100
	ZeAcc	可选参数，指定拱形运动下降段的加速度比例，范围 1~100
	ZeSpd	可选参数，指定拱形运动下降段的减速度比例，范围 1~100

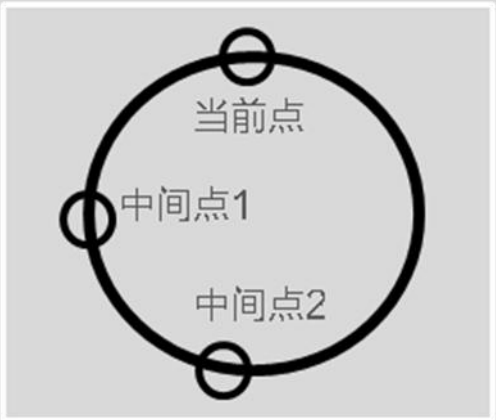
	ZeDec	可选参数，指定拱形运动下降段的速度比例，范围 1~100
	Stop	可选参数，用来描述当第三轴的电流超过设定阈值或输入信号被触发时，机器人停止的两种方式（默认是减速停 Stop=0）： Stop = 0 减速停（默认） Stop = 1 不减速直接停（不减速直接停仅适用于机器人在超低速运行时使用，否则会出现速度偏差超限等伺服报警）
	I	可选参数，第三轴电流设定阈值，单位 mA
	In	可选参数，输入检测信号端口，此参数与 ON or OFF 搭配使用
	ON/OFF	ON：打开；OFF：关闭
	返回值说明	
	STA	0: 指令运动正常
		1：输入信号被触发或第三轴电流超过设定的阈值
举例说明	1. MArchP(p1,0,10,5) --机器人从当前位置 Z 轴上升 10mm 距离，然后以点到点的运动方式移动到距离目标位置正上方 5mm 处，Z 轴下降 5mm 到达目标位置（拱形的高度不能超过 Z 轴的最高限 0mm （绝对位置）） 2. MArchP(p1,0,10,5, “Acc=50 Spd=100”) --机器人以 50%的加速度，100%的速度做点到点方式拱形运动 3. MArchP(p2,-10) --机器人从当前位置 Z 轴上升到-10mm 位置，然后以点到点的运动方式移动到（x=p2.x,y=p2.y,z=-10,c=p2.c）位置，最后 Z 轴下降(-10-p2.z)到达目标位置 p2	

➢ 输入检测和电流检测功能的使用可参考 MovL 指令的例子 5、6、7、8、9、10

MArc	
使用说明	笛卡尔坐标系下的圆弧运动
语法说明	MArc(A,B,"CP=20Acc=20 Dec=20 Spd=100 Angle = 360")
参数说明	**下表各指令参数说明**

	A	笛卡尔坐标圆弧经过点。可以是点的名称 p1~p9999，也可以是点的索引 1~9999
	B	笛卡尔坐标圆弧终点。可以是点的名称 p1~p9999，也可以是点的索引 1~9999
	CP	可选参数，说明运动到目标点时是否平滑过渡，范围 0~100
	Acc	可选参数，指定运动到目标位置的加速度，单位为 mm/s^2
	Dec	可选参数，指定运动到目标位置的减速度，单位为 mm/s^2
	Spd	可选参数，指定运动到目标位置的速度，单位为 mm/s
	Angle	可选参数，指定圆弧的角度，范围 1~360
示意图		
举例说明	1. MArc(p1,p2) --机器人从当前位置以圆弧的方式经过 p1 点运动到 p2 目标位置 2. MArc(1,2) --机器人从当前位置以圆弧的方式经过 p1 点运动到 p2 目标位置 3. MArc(1,2,"Acc=100 Spd=1000") --机器人从当前位置以圆弧的方式经过 p1 点运动到 p2 目标点，其中加速度为 100mm/s2，速度为 1000mm/s 4. MArc(1,2,"CP=20") --机器人圆弧运动到 p2 点时平滑过渡	

MCircle

使用说明	机器人在笛卡尔坐标系下的圆周运动	
语法说明	MCircle(A,B," CP=20 Acc=20 Dec=20 Spd=100")	
参数说明	**下表各指令参数说明**	
	A	笛卡尔坐标圆周经过点 1。可以是点的名称 p1~p9999，也可以是点的索引 1~9999
	B	笛卡尔坐标圆周经过点 2。可以是点的名称 p1~p9999，也可以是点的索引 1~9999
	CP	可选参数，说明运动到目标点时是否平滑过渡，范围 0~100
	Acc	可选参数，指定运动到目标位置的加速度，单位为 mm/s^2
	Dec	可选参数，指定运动到目标位置的减速度，单位为 mm/s^2
	Spd	可选参数，指定运动到目标位置的速度，单位为 mm/s
示意图		
举例说明	1. MCircle(p1,p2) --机器人从当前位置经过 p1 点和 p2 点的圆周运动 2. MCircle(1,2) --机器人从当前位置经过 p1 点和 p2 点的圆周运动。 3. MCircle(1,2,"Acc=100 Spd=1000") --机器人从当前位置经过 p1 点和 p2 点的圆周运动，其中加速度为 100mm/s2，速度为 1000mm/s。 4. MCircle(1,2,"CP=20") --机器人从当前位置经过 p1 点和 p2 点的圆周平滑运动。	

MSpline

使用说明	样条曲线插补运动指令
语法说明	MSpline(A,B, “CP=20Acc=20Dec=20 Spd=100AccC=20 SpdC=20”)
参数说明	**下表各指令参数说明**

A	样条曲线起始点（点位表中示教）
B	样条曲线结束点（点位表中示教）
Acc	可选参数，指定运动到目标位置的加速度，单位 mm/s^2
Dec	可选参数，指定运动到目标位置的减速度，单位 mm/s^2
Spd	可选参数，指定运动到目标位置的速度，单位 mm/s
AccC	可选参数，指定运动到目标位置的姿态加速度，单位度/s^2
SpdC	可选参数，指定运动到目标位置的姿态速度，单位度/s

举例说明	1. MSpline(p1,p10) --p1~p10 点共 10 个点形成的样条曲线轨迹 2. MSpline(p10,p150) --p10~p150 共 141 个点形成的样条曲线轨迹

注：

- 起始点、结束点、以及之间的点位都需要在点位表（DATA.PTS）中示教（必须连续，中间的点位数据不能为空，否则出错）；
- 从起始点到结束点，点位坐标（数据）个数不能少于 3 个，也不能超过 300 个；
- 样条曲线插补运动指令，须结合示教器参数里的**样条曲线参数（默认拆分步长）；**
- 该指令会自动判断起始点与结束点坐标是否为同一个点，若为同一个点，则曲线封闭，否则曲线不封闭。

waitpos

使用说明	等待运动脉冲发送完成指令（运动 DSP 发送脉冲完成指令）
语法说明	waitpos()
参数说明	无
举例说明	1. MovP(p1) MovP(p2) waitpos() --等待运动 DSP 将运动到 P2 点的脉冲全部发送给伺服 DSP a =1 --若无 waitpos()指令， a 的值会提前打印 print(“a”)

waitrealpos

使用说明	机器人运动到位（停稳）指令（伺服响应脉冲到位指令）
语法说明	waitrealpos()
参数说明	无
举例说明	1. MovP(p1) waitrealpos() --等待机器人到达 p1 点（伺服响应有一定的滞后性） print(“已到达 p1 点”)

MAccept

使用说明	线程获取与运动相关的使用权：运动指令、设置用户、设置工具	
语法说明	1. MAccept() 2. MAccept(A)	
参数说明	A	● 线程获取与运动相关的使用权等待的时间（单位：ms），超过了设定的时间系统会报警：运动请求错误，当前运动没有结束 ● 输入为空时线程会一直等待获取到与运动相关的使用权

MRelease		
使用说明	线程释放与运动相关的使用权：运动指令、设置用户、设置工具	
语法说明	MRelease()	
参数说明	无	
举例说明	--线程 1：AR 程序 function main() while true do MAccept() MovP(1) MovP(2) MRelease() Dleay(500) end end	--线程 2：AR 程序 function main() while true do Delay(500) MAccept() MovP(3) MovP(4) MRelease() end end

注 1：MAccept 指令与 MRelease 指令是配套使用；

注 2：哪个线程先执行运动就会优先获取运动使用权，其它线程若要使用必须等正在拥有运动使用的线程释放该使用权。

3.1.12 运动参数设置指令

指令符号	指令说明
AccJ	设置加速度比例指令，影响 MovJ、MovP、MovPR、MArchP 指令的加速时间
DecJ	设置减速度比例指令，影响 MovJ、MovP、MovPR、MArchP 指令的减速时间
SpdJ	设置速度比例指令，影响 MovJ、MovP、MovPR、MArchP 指令的运行速度
JerkJ	设置加加速度比例指令，影响 MovJ、MovP、MovPR、MArchP 指令的运行速度

AccL	设置插补运动指令加速度，影响 MovL、MovLR、MArc、MCircle 指令的加速时间。单位 mm/
DecL	设置插补运动指令减速度，影响 MovL、MovLR、MArc、MCircle 指令的减速时间。单位 mm/
SpdL	设置插补运动指令速度，影响 MovL、MovLR、MArc、MCircle 指令的运行速度。单位 mm/s
JerkL	设置插补运动指令加加速度，影响 MovL、MovLR、MArc、MCircle 指令的加速时间。单位 mm/
AccC	设置插补姿态运动的加速度，影响 MovL、MovLR、MArc、MCircle 指令的加速时间。单位度/
SpdC	设置插补姿态运动指令速度，影响 MovL、MovLR、MArc、MCircle 指令的运行速度。单位度/s
JerkC	设置插补姿态运动指令加加速度，影响 MovL、MovLR、MArc、MCircle 指令的加速度变化时间。单位度/

AccJ

使用说明	设置点到点运动方式的加速度比例	
语法说明	AccJ(A)	
参数说明	A	百分比，范围 1~100
举例说明	1. AccJ(50) --设置点到点 50%的加速度比例 MovP(p2) --机器人以 50%的加速度比例运动到 p2 目标位置	

◆ 注意：设置后机器人一直保持该全局加速度比例为点到点运动的默认加速度比例值，直到下一次更新。

DecJ

使用说明	设置点到点运动方式的减速度比例
语法说明	DecJ(A)

参数说明	A	百分比，范围 1~100
举例说明	1. DecJ(50) --设置点到点 50%的减速度比例 MovP(p2) --机器人以 50%的减速度运动到 p2 目标位置	

SpdJ

使用说明	设置点到点运动方式的速度比例
语法说明	SpdJ(A)
参数说明	A 百分比，范围 1~100
举例说明	1. SpdJ(50) --设置点到点 50%的速度比例 MovP(p2) --机器人以 50%的速度比例运动到 p2 目标位置

JerkJ

使用说明	设置点到点运动方式的加加速度的百分比
语法说明	JerkJ(A)
参数说明	A 百分比，范围 1~100
举例说明	1. JerkJ(50) --设置点到点 50%的加加速度比例 MovP(p2) --机器人以 50%的加加速度运动到 p2 目标位置

AccL

使用说明	设置直线运动方式的加速度
语法说明	AccL(A)
参数说明	A 实际加速度，单位 **mm/**，范围 **100~100000**
举例说明	1. AccL(500) --设置直线运动的加速度为 500 mm/ MovL(p2) --机器人以 500 mm/的加速度运动到 p2 目标位置

DecL		
使用说明	设置直线运动方式的减速度	
语法说明	DecL(A)	
参数说明	A	实际减速度，单位 **mm/**，范围 **100~100000**
举例说明	1. DecL(500) --设置直线运动的减速度为 500 mm/ MovL(p2) --机器人以 500 mm/的减速度运动到 p2 目标位置	

SpdL		
使用说明	设置插补运动方式的速度	
语法说明	SpdL(A)	
参数说明	A	实际插补速度，单位 **mm/s**,范围 **10~10000**
举例说明	1. SpdL(500) --设置直线运动的速度为 500mm/s MovL(p2) --机器人以 500mm/s 的速度运动到 p2 目标位置	

JerkL		
使用说明	设置插补运动的加加速度	
语法说明	JerkL(A)	
参数说明	A	实际加加速度，单位 **mm/**，范围为 **1~999999**
举例说明	1. JerkL(50000) --设置直线运动的加加速度为 50000 mm/ MovL(p2) --机器人以 50000 mm/的加加速度运动到 p2 目标位置	

AccC		
使用说明	设置插补运动的姿态加速度	
语法说明	AccC(A)	
参数说明	A	实际姿态加速度，单位**度/**，范围 **100~100000**

举例说明	1. AccC (500) --设置插补运动的姿态加速度为 500 度/ MovL(p2) --机器人以 500 度/的姿态加速度运动到 p2 目标位置

SpdC		
使用说明	设置插补运动的姿态速度	
语法说明	SpdC(A)	
参数说明	A	实际加速度，单位**度/s**，范围 **10~1000**
举例说明	1. SpdC (500) --设置插补运动的姿态速度为 500 度/ s MovL(p2) --机器人以 500 度/ s 的插补姿态速度运动到 p2 目标位置	

JerkC		
使用说明	设置插补运动的姿态加加速度	
语法说明	JerkC(A)	
参数说明	A	实际姿态加加速度，单位**度/**，范围 **1~999999**
举例说明	1. JerkC (500) --设置插补运动的姿态加加速度为 500 度/ MovL(p2) --机器人以 500 度/的姿态加加速度运动到 p2 目标位置	

3.1.13 程序管理指令

指令符号	指令说明
Delay	延时指令。单位：毫秒
Exit	程序运行终止
Pause	暂停程序运行

Delay	
使用说明	延时指令
语法说明	Delay(A)

参数说明	A	延时时间，单位为毫秒
举例说明	1. Delay(1000) --程序延时 1000 毫秒 2. Delay(1) --机器人延时 1 毫秒 3. local time = 1000 Delay(time) --参数为变量	

Exit	
使用说明	退出程序执行指令
语法说明	Exit()
参数说明	该指令无参数
举例说明	Exit() --执行该指令后程序程序停止执行 MovL(1) --该指令不会被执行

Pause	
使用说明	暂停程序执行指令
语法说明	Pause()
参数说明	该指令无参数
举例说明	Pause() --执行该指令后程序暂停执行,按**启动键**后程序继续运行 MovL(p1)

◆ 注意：**暂停后，按启动键后程序会继续从暂停行继续运行。**

3.1.14 一般指令

指令符号	指令说明
X	创建指定 X 轴笛卡尔坐标绝对位置点的指令
Y	创建指定 Y 轴笛卡尔坐标绝对位置点的指令
Z	创建指定 Z 轴笛卡尔坐标绝对位置点的指令
C	创建指定 C 轴笛卡尔坐标绝对位置点的指令

XYZC	创建指定 XYZC 轴笛卡尔坐标绝对位置点的指令

X/Y/Z/C		
使用说明	创建指定某个轴笛卡尔坐标位置的点	
语法说明	X(A) Y(A) Z(A) C(A)	
参数说明	A	各个轴笛卡尔坐标位置，C 轴为角度，其它轴为 mm
举例说明	1. MovL(p10 + X(20)) --机器人运动到相对于 p10 点往 X 轴正向偏移 20 毫米的位置 2. MovL(p11 + Y(-20)) --机器人运动到相对于 p11 点往 Y 轴负向偏移 20 毫米的位置 3. MovL(p12 + Z(-10)) --机器人运动到相对于 p12 点往 Z 轴负向偏移 10 毫米的位置 4. MovL(p13 +C(30)) --机器人运动到相对于 p13 点往 C 轴正向偏(逆时针)移 30 度的位置	

◆ 注意：该指令是用来生成一个点数据，不会产生运动，一般与点数据运算后作为参数传递给运动指令。

XYZC		
使用说明	创建指定各个轴笛卡尔坐标位置的点	
语法说明	XYZC(A,B,C,D)	
参数说明	**下表各指令参数说明**	
	A	指定 X 轴笛卡尔坐标位置，单位 mm
	B	指定 Y 轴笛卡尔坐标位置，单位 mm
	C	指定 Z 轴笛卡尔坐标位置，单位 mm
	D	指定 C 轴笛卡尔坐标位置，单位度
举例说明	MovL(p10 + XYZC(10,20,-5,30)) --机器人运动到相对于 p10 点往 X 轴正向偏移 10 毫米,Y 轴正向偏移 20 毫米、Z 轴负向偏移 5 毫米、C 轴正向偏移 30 度的位置移动	

3.1.15 输入/输出指令

指令符号	指令说明
DI	读取输入端口状态
DO	输出端口打开或关闭操作
WDI	读取输入端口状态,直到等待某一信号有效，则继续执行后续的程序
WDO	读取输出端口状态,直到等待某一信号有效，则继续执行后续的程序
MO	辅助输出打开或关闭操作
WMO	读取辅助输出端口状态,直到等待某一信号有效，则继续执行后续的程序

DI	
使用说明	读取输入端口状态
语法说明	**形式 1：**DI(-1)或 DI(-2) **形式 2：**DI(A)
参数说明	**形式 1：**DI(-1)的返回值是一个 32 位的二进制数转化的十进制数值，从低位到高位分别代表输入端口（0~31）的状态值；DI(-2)的返回值也是一个 32 位的二进制数转化的十进制数值，低两位分别代表输入端口（32~33）的状态值。其中 0 代表关闭，1 代表打开。 **形式 2：**返回值　ON(OFF) 或十进制的值 **下表各指令参数说明** A　读取的输入端口号，范围 0~33
举例说明	1. local input = DI(-1)　　--获取输入端口（0~31）的状态 if ((input>>0)&0x0001)==1 then　--判断输入端口 0 是否打开，若为 1，则输入端口 0 打开 MovP(p1)

elseif ((input>>9)&0x0001)==1 then--判断输入端口 9 是否打开，若为 1，则输入端口 1 打开

MovP(p2)

end

2. if DI(10) == ON then --输入端口 10 有效时运动到 p1 点

MovP(p1)

end

3. value = DI({1,2,3}) --读取输入端口 1,2,3 的状态，若返回值 value 为 7（111）则表示输入端口 1,2,3 打开；若 value 为 6（110），则 1,2 打开，3 关闭；若 value 为 5（101），则 1,3 打开，2 关闭….依次类推…

- ◆ 注意：DI 指令只是读取输入端口的状态，状态有效或无效都不会死等待。
- ◆ 注意：若同时读取某几个输入端口的状态，输入端口一定要按照从小到大或从大到小的顺序排序。例如 DI({2,3,4}),DI({4,7,9}),DI({8,7,3,1}) 正确，DI({3,2,4}),DI({7,4,9}), DI({7,8,1,3})错误

DO	
使用说明	输出端口打开或关闭操作
语法说明	用法 1：DO(A,B) 用法 2：DO(A,B,"**Time** = C") 用法 3：DO(A,B, “**F**”) 用法 4：DO(A,B, “**Time** = C **H**”) 用法 5：DO(A,B , “**Time** = C **F**”) 用法 6：DO(A,B, “**POS**=D”)
参数说明	返回值 ON 或 OFF **下表各指令参数说明**

A	读写的输出端口号，范围 0~26，可以是单个端口，也可以是多个端口
B	输出端口的状态值
C	可选参数，时间（ms）
D	位置百分比

举例说明

1. DO(10,ON) --打开 10 号输出端口为 ON

   --DO({1,2,3,4},{ON,ON,ON,ON}) --打开输出 1,2,3,4 为 ON

   --DO({5,6,7,8},{ON,OFF,ON,OFF}) --打开输出 5,7 为 ON，关闭输出 6,8 为 OFF

2. DO(10,ON,“**Time**=1000”) --打开 10 号输出端口为 ON,并保持 1 秒钟，1 秒之后输出 10 切换为 OFF

   --DO({1,2,3},5) --5 对应的二进制为 101，则表示打开输出端口 1,3 为 ON，关闭输出端口 2 为 OFF

3. MovP(p1)

   DO(1,ON, “**F**”) --机器人从当前位置运动到 p1 点，到达 p1 点后打开输出端口 1，然后运动到 p2 点，运动连续

   MovP(p2)

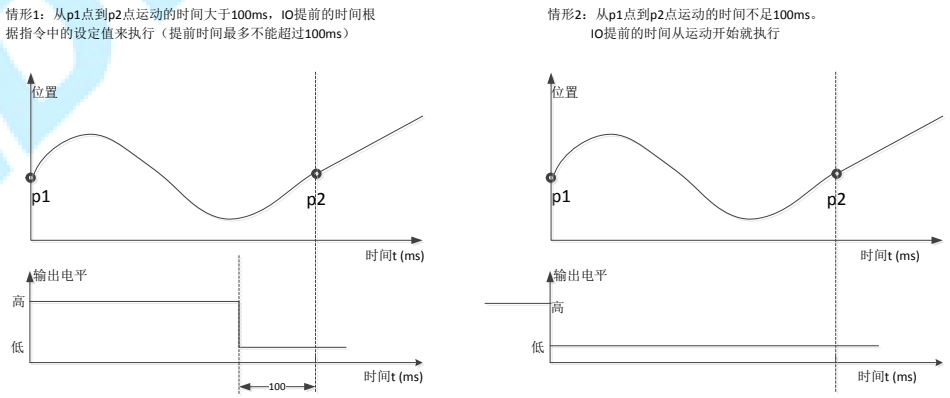
4. MovP(p1)

   MovP(p2)

   DO(1,ON, “**Time**=100 H”)

   情形 1: 机器人从 p1 点到 p2 点运动的时间大于 100ms, 如果 **Time** 设定的值小于 100ms,则 IO 输出提前动作的时间取决于 **Time** 设定的值;若 **Time** 设定的值大于 100ms,则 IO 输出提前动作的时间就是 100ms。

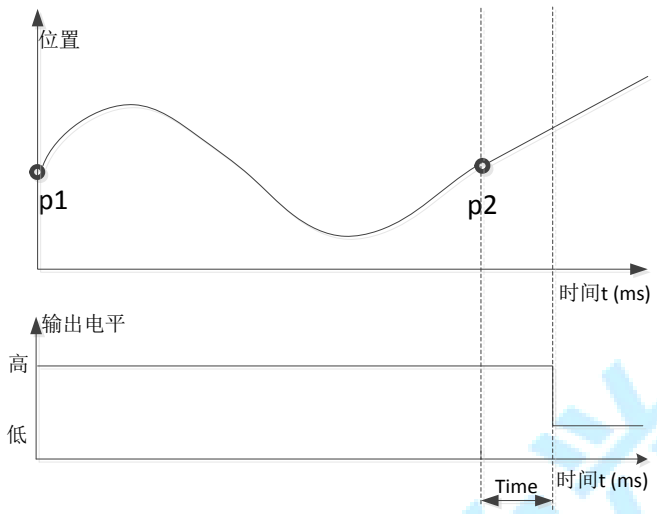
   情形 2: 机器人从 p1 点到 p2 点运动的时间小于 100ms, IO 输出提前动作的时间从运动一开始（即 p1 点）就执行。



5. MovP(p1)

   MovP(p2)

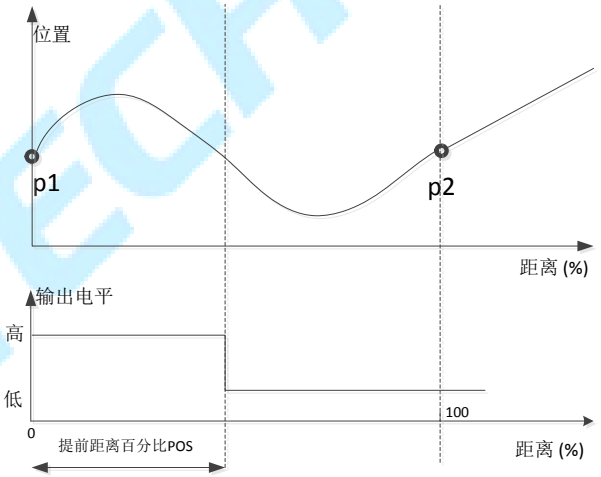
   DO(1,ON, “**Time**=50 F”) --从 p1 点运动到 p2 的点，在到达 p2 点后的 50ms 打开输出端口 1



6. MovP(p1)

   MovP(p2)

   DO(1,ON, “**POS**=50”) --机器人在 p1 点与 p2 点之间距离的 50%时打开输出端口 1



- 用法 2 中的可选参数 Time 为输出端口保持时间，超过保持时间后会将该当前的输出端口状态切换为相反状态；
- 若同时打开或关闭某几个输出端口，输出端口一定要按照从小到大或从大到小的顺序排序；
- 缓存 IO 功能不适用于组合 IO 的情况；
- 缓存 IO 功能用法 3、4、5 适用于点到点运动（MovP）、直线运动（MovL）以及拱形运动(MArchP)，但用法 6 适用于点到点运动(MovP)和直线运动(MovL)；
- **Time**、**POS**、**F**、**H** 为 DO 指令中的关键字，不能修改；
- 运动过程中启用缓存 IO 功能，当前的运动不会终止，运动连续。
- DO(A,B, “H”)效果与 DO(A,B)相同。

WDI	
使用说明	读取输入端口状态,直到等待某一或多个信号有效,则继续执行后续的程序
语法说明	WDI(A,B) WDI(A,B, “Time=1000”)
参数说明	返回值　1 (ON)或 0 (OFF) **下表各指令参数说明** A　读取的输入端口号 B　输入端口的状态值，ON 或 OFF Time　可选参数，等待时间，单位为毫秒
举例说明	1. local flag = WDI(10,ON)　--直到等到输入端口 10 状态为 ON 再执行 MovP 指令 print(flag)　--返回值为 1 MovP(p1) 2. local flag = WDI(10,OFF)　--直到等到输入端口 10 状态为 OFF 再执行 MovP 指令 print(flag)　--返回值为 0 MovP(p1) 3. local flag = WDI(10,ON, “Time=2000”) --等待输入端口 10 状态为 ON，等待时间 2 秒，若 2 秒后输入端口 10 状态仍为 OFF，则继续执行 MovP 指令 if flag == 1 then　在 2 秒内输入端口 10 状态检测为 ON MovP(p1) elseif flag == 0 then　--在 2 秒内输入端口 10 状态一直检测为 OFF MovP(p2) end 4. local flag = WDI({1,2,3},{ON,ON,ON})　--直到同时等到输入端口 1,2,3 同时为 ON 再执行 MovP 指令 print(flag)　--返回结果为 7（对应二进制的 111） MovP(p1)

5. local flag = WDI({1,2,3},{ON,ON,ON}, “Time=2000”) --等待输入端口 1,2,3 状态为 ON，等待时间为 2 秒，若 2 秒之后输入端口仍 OFF ，仍继续执行 MovP 指令

   if flag ==7 then --输入端口 1、2、3 在 2 秒内状态检测到为 ON

   MovP(p1)

   elseif flag ==0 then

   MovP(p2) --输入端口 1、2、3 在 2 秒内状态一直检测为 OFF

   end

◆ 注意：若同时等待某几个输入端口的状态，输入端口一定要按照从小到大或从大到小的顺序排列。

WDO	
使用说明	读取输出端口状态,等待某一信号有效,则继续执行后续的程序
语法说明	WDO(A,B) WDO(A,B, “Time=1000”)
参数说明	返回值 ON 或 OFF **下表各指令参数说明** A：读取的输出端口号 B：输出端口的状态值，ON 或 OFF Time：可选参数，等待时间，单位为毫秒
举例说明	1. WDO(10,ON) --直到等到输入端口 10 状态为 ON 再执行 MovP 指令 MovP(p2) 2. WDO(10,ON, “Time=2000”) --等待输出端口 10 状态为 ON，等待时间为 2 秒，若 2 秒之后输出端口 10 状态仍为 OFF，则继续执行 MovP MovP(p2)

MO

使用说明	辅助输出端口打开或关闭操作
语法说明	用法 1：MO(A,B) 用法 2：MO(A,B, “**Time** = C”) 用法 3：MO(A,B, “**F**”) 用法 4：MO(A,B, “**Time** = C **H**”) 用法 5：MO(A,B , “**Time** = C **F**”) 用法 6：MO(A,B, “**POS**=D”)
参数说明	返回值 ON 或 OFF **下表各指令参数说明**

A	读写的辅助输出端口号，范围 0~31，可以是单个端口，也可以是多个端口
B	输出端口的状态值
C	可选参数，时间（ms）
D	位置百分比

举例说明

1. MO(10,ON) --打开 10 号辅助输出端口为 ON
   --MO({1,2,3,4},{ON,ON,ON,ON}) --打开辅助输出 1,2,3,4 为 ON
   --MO({5,6,7,8},{ON,OFF,ON,OFF}) --打开辅助输出 5,7 为 ON，关闭输出 6,8 为 OFF
2. MO(10,ON,“**Time**=1000”) --打开 10 号辅助输出端口为 ON,并保持 1 秒钟，1 秒之后辅助输出 10 切换为 OFF
   --MO({9,10},{ON,ON},“**Time**=2000”) --打开辅助输出 9,10,并保持 2 秒钟，2 秒之后辅助输出 9,10 切换为 OFF
   --MO({9,10},{0,0},“**Time**=2000”) --打开辅助输出 9,10 为 OFF,并保持 2 秒钟，2 秒之后辅助输出 9,10 切换为 ON
   --MO({1,2,3},5) --5 对应的二进制为 101，则表示打开辅助输出端口 1,3 为 ON，关闭辅助输出端口 2 为 OFF
3. MovP(p1)
   MO(1,ON, “**F**”) --机器人从当前位置运动到 p1 点，到达 p1 点

后打开辅助输出端口 1，然后运动到 p2 点，运动连续

MovP(p2)

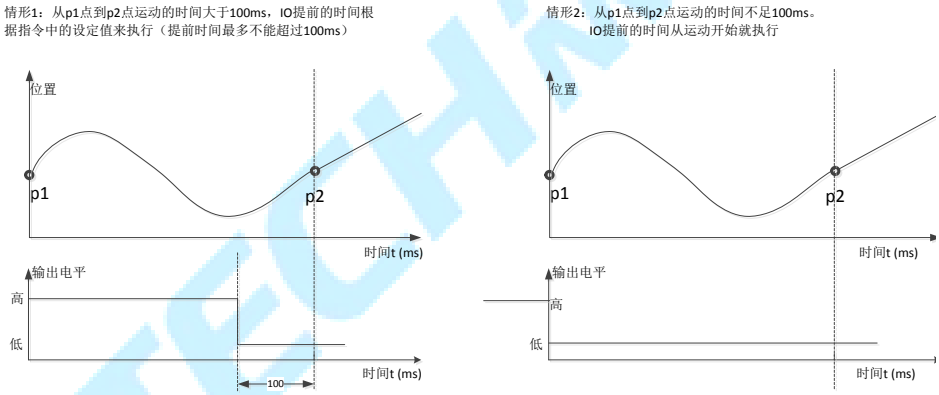
4. MovP(p1)

MovP(p2)

MO(1,ON, “**Time**=100　H”)

情形 1: 机器人从 p1 点到 p2 点运动的时间大于 100ms, 如果 **Time** 设定的值小于 100ms,则辅助 IO 输出提前动作的时间取决于 **Time** 设定的值；若 **Time** 设定的值大于 100ms,则辅助 IO 输出提前动作的时间就是 100ms。

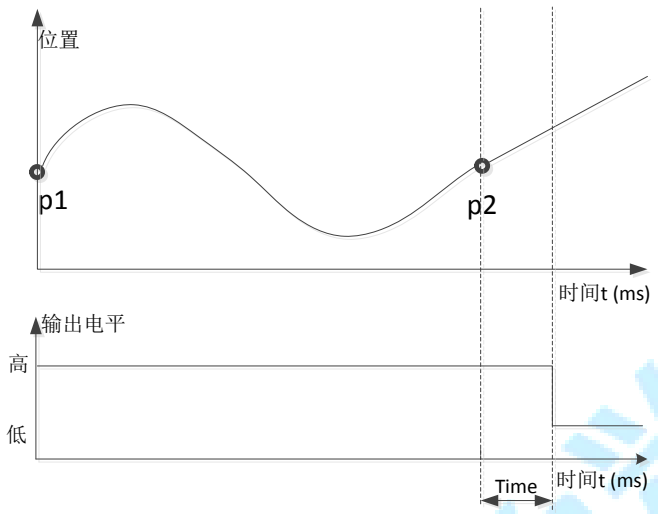
情形 2: 机器人从 p1 点到 p2 点运动的时间小于 100ms, IO 输出提前动作的时间从运动一开始（即 p1 点）就执行。



5. MovP(p1)

MovP(p2)

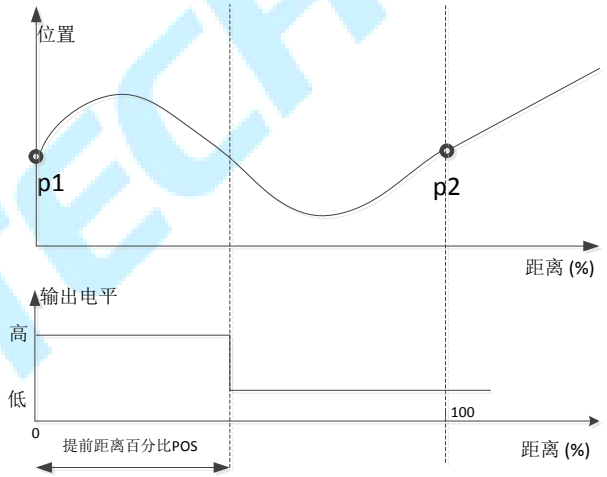
MO(1,ON, “**Time**=50　F”)　--从 p1 点运动到 p2 的点，在到达 p2 点后的 50ms 打开输出端口 1



6. MovP(p1)

   MovP(p2)

   MO(1,ON, “**POS**=50”) --机器人在 p1 点与 p2 点之间距离的 50%时打开辅助输出端口 1



- 用法 2 中的可选参数 Time 为辅助输出端口保持时间，超过保持时间后会将该当前的辅助输出端口状态切换为相反状态；
- 若同时打开或关闭某几个辅助输出端口，输出端口一定要按照从小到大或从大到小的顺序排序；
- 缓存 IO 功能不适用于组合 IO 的情况；
- 缓存 IO 功能适用于点到点运动（MovP）、直线运动（MovL）以及拱形运动（MArchP）；
- **Time**、**POS**、**F**、**H** 为 MO 指令中的关键字，不能修改；
- 运动过程中启用缓存 IO 功能，当前的运动不会终止，运动连续。
- MO(A,B, “H”)效果与 MO(A,B)相同。

WMO	
使用说明	读取辅助输出端口状态,等待某一信号有效,则继续执行后续的程序
语法说明	WMO(A,B) WMO(A,B, “Time=1000”)
参数说明	返回值 ON 或 OFF **下表各指令参数说明** A：读取的输出端口号 B：输出端口的状态值，ON 或 OFF Time：可选参数，等待时间，单位为毫秒
举例说明	1. WMO(10,ON) --直到等到输入端口 10 状态为 ON 再执行 MovP 指令 MovP(p2) 2. WMO(10,ON, “Time=2000”) --等待辅助输出端口 10 状态为 ON，等待时间为 2 秒，若 2 秒之后辅助输出端口 10 状态仍为 OFF，则继续执行 MovP MovP(p2)

3.1.16 坐标系指令

指令符号	指令说明
SetU	设置机器人当前用户坐标系
WrU	修改用户坐标系的数据
RdU	读取目标用户里的值（X\Y\Z\C）
CacU	两点法新建用户坐标系
U2U	用户（0~19）之间坐标相互转换
SetT	设置机器人当前工具坐标系
WrT	修改工具坐标系的数据
RdT	读取目标工具里的值（X\Y\Z\C）
V2Tool	根据视觉坐标计算工具坐标

CacT	两点法新建工具坐标系
Getpos	获取当前位置笛卡尔坐标以及关节坐标
encoderget	获取编码器的脉冲值
Targetok	判断目标位置是否为机器人可到达点位
SetI	设置当前最佳加速度负载惯量使用组别（0~4）

SetU

使用说明	设置当前用户坐标系	
语法说明	SetU(A)	
参数说明	A	设置的用户坐标系号，范围 0~19
	B	用户坐标偏移量
举例说明	1. SetU(1) --选择用户坐标系 1 MovL(p1) --运动到用户坐标系 1 中的 p1 位置	

◆ 注意：坐标系设置后立即有效并保持到下一次更新。

WrU

使用说明	修改用户坐标系的数据	
语法说明	**形式 1：** WrU(A,B,C) **形式 2：** WrU(A,B)	
参数说明	**下表各指令参数说明(形式 1)**	
	A	要修改的用户坐标系号，范围为 1~19
	B	修改的轴号，分别为 AX、AY、AZ、AC
	C	修改的值，X/Y/Z 轴单位为毫米,C 轴单位为角度
	下表各指令参数说明(形式 2)	
	A	要修改的第几组用户坐标系号，范围为 1~19
	B	Table 类型，包含用户坐标系的原点坐标，以及该用户相对于用户 0 旋转的角度
举例说明	1. WrU(1,AX,200) --修改第一组用户坐标系的 X 轴为 200 毫米	

2. WrU(1,AC,100) --修改第一组用户坐标系的 C 轴为 100 度
3. WrU(2,{x=200,y=100,z=0,c=90}) --修改用户 2 的原点为(200,100,0),用户 2 相对于用户 0 的旋转角度为 90 度
4. Pos = { x=300,y=100,z=0,c=-90}
   WrU(3,Pos)

RdU

使用说明	读取目标用户里的值（X\Y\Z\C）	
语法说明	PosU = RdU(A)	
参数说明	A	全局用户坐标系列表里的序号，范围 0~19
返回值	PosU	table 类型数据,目标用户对应的 X\Y\Z\C 组成的数组
举例说明	1. local PosU = {} PosU = RdU(1) --读取用户 1 里对应的 X\Y\Z\C 值 print(PosU.x, PosU.y, PosU.z, PosU.c)	

CacU

使用说明	新建用户坐标系	
语法说明	CacU (pos1,pos2)	
参数说明	**下表各指令参数说明**	
	pos1	用户坐标系的原点
	pos2	用户坐标系 X 方向上的一点
举例说明	1. local pos1 = {x=300,y=100,z=0,c=0} --用户坐标系的原点 local pos2 = {x=300,y=120,z=0,c=0} --用户坐标系 X 轴方向上的一点 WrU(1,CacU(pos1,pos2)) --将新建的用户坐标指定为用户坐标系 1	

U2U

使用说明	用户（0~19）之间坐标相互转换
语法说明	pos = U2U (A,B,C)
参数说明	**下表各指令参数说明**

A	被转换的用户号（0~19）
B	目标用户号（0~19）
C	被转换用户号下的坐标
pos	目标用户号下的坐标

举例说明

1. local pos = U2U(0,2,p1) --将用户 0 下的 p1 点转换到用户 2 下的坐标

   SetU(2)

   MovP(pos)

2. local pos1= {x= 100,y=40,z=-10,c=30}

   local pos = U2U(1,3,pos1) –把用户 1 中的 pos1 点的坐标转换到用户 3 下的坐标

   SetU(3)

   MovP(pos)

SetT

使用说明	设置当前工具坐标系
语法说明	SetT(A)
参数说明	A　设置的工具坐标系号，范围 0~19

举例说明

1. SetT (1)　　--选择工具坐标系 1

   MovL(p1)　　--运动到工具坐标系 1 中的 p1 位置

WrT

使用说明	修改工具坐标系的数据
语法说明	**形式 1：** WrT (A,B,C) **形式 2：** WrT(A,B)
参数说明	**下表各指令参数说明(形式 1)**

	A	要修改的工具坐标系号，范围为 1~19
	B	修改的轴号，分别为 AX、AY、AZ、AC
	C	修改的值，单位为毫米,C 轴为角度
	下表各指令参数说明(形式 2)	
	A	要修改的工具坐标系号，范围为 1~19
	B	Table 类型，包含要修改的工具末端相对于机器人末端在 X,Y,Z,C 轴方向的偏移量
举例说明	1. WrT(1,AX,20) --修改工具坐标系 1 的 X 轴偏移量为 20 毫米 2. WrT(1,AY,10) --修改工具坐标系 1 的 Y 轴偏移量为 10 毫米 3. WrT(1,{x= 10,y=20,z=0,c=30}) --修改工具 1 相对于机器人末端在 X、Y、Z、C 方向上的偏移量分别为 10mm,20mm,0mm,30 度。 4. Offset = { x=10,y=10,z=0,c=-30} WrT(2,Offset)	

RdT		
使用说明	读取目标工具里的值（X\Y\Z\C）	
语法说明	PosT= RdT(A)	
参数说明	A	全局工具坐标系列表里的序号，范围 0~19
返回值	PosT	table 类型数据,目标工具对应的 X\Y\Z\C 组成的数组
举例说明	1. local PosT = {} PosT = RdT(1) --读取工具 1 里对应的 X\Y\Z\C 值 print(PosT.x, PosT.y, PosT.z, PosT.c)	

V2Tool	
使用说明	根据视觉坐标计算工具坐标
语法说明	V2Tool(A,B)

参数说明	A	视觉输出的笛卡尔坐标值
	B	工具坐标系号，范围 1~19
举例说明	1. local vis = {x=300, y=10,z=0,c=0} --视觉坐标 V2Tool(vis,3) --将计算出来的工具写入到工具坐标系 3 SetT(3) --设置当前工具坐标系为 3 号坐标系	

◆ 注意：视觉坐标和当前的机器人坐标必须在同一坐标系下。

CacT		
使用说明	两点法新建工具坐标系	
语法说明	CacT(pos1,pos2)	
参数说明	**下表各指令参数说明**	
	pos1	机器人末端笛卡尔坐标
	pos2	夹具末端笛卡尔坐标
举例说明	1. pos1=Getpos () pos2 = {x=300,y=100,z=0,C=30} WrT(1,CacT(pos1,pos2))	

注：

（1） 夹具末端的笛卡尔坐标通过视觉定位获取；

（2） 夹具末端笛卡尔坐标和机器人末端笛卡尔坐标必须处于同一坐标系；

Getpos		
使用说明	获取当前笛卡尔坐标以及关节坐标	
语法说明	**形式 1：**local pos1,pos2 = Getpos() **形式 2：**local pos = Getpos()	
参数说明	无参数	
返回值	pos1	机器人当前的笛卡尔坐标
	pos2	机器人当前的关节坐标
举例说明	1. local pos1,pos2=Getpos()--获取当前笛卡尔坐标及关节坐标 print(“pos1=”,pos1.x,pos1.y,pos1.z,pos1.c,pos1.h)	

	--pos1.x,pos1.y,pos1.z,pos1.c,pos1.h 分别为笛卡 x/y/z/c/h 的值 print(“pos2=”,pos2.x,pos2.y,pos2.z,pos2.c,pos2.h) --pos2.x,pos2.y,pos2.z,pos2.c,pos2.h 为关节 J1/J2/J3/J4 以及手系的值 2. local pos = Getpos() --获取当前笛卡尔坐标 print(“pos=”,pos.x,pos.y,pos.z,pos.c,pos.h) -- pos.x,pos.y,pos.z,pos.c,pos.h 分别为笛卡尔 x/y/z/c/h 的值

encoderget		
使用说明	获取编码器的脉冲值	
语法说明	Pulse = encoderget(A)	
参数说明	A	编码器号，范围 1~6
返回值	Pulse	返回对应编码器的脉冲值
举例说明	1. Pulse1=encoderget(1) --获取 J1 轴编码器 M1 的脉冲值； 2. Pulse2=encoderget(2) --获取 J2 轴编码器 M2 的脉冲值； 3. Pulse3=encoderget(3) --获取 J3 轴编码器 M3 的脉冲值； 4. Pulse4=encoderget(4) --获取 J4 轴编码器 M4 的脉冲值； 5. Pulse5=encoderget(5) --获取外接编码器 M5 的脉冲值； 6. Pulse6=encoderget(6) --获取外接编码器 M6 的脉冲值；	

Targetok		
使用说明	判断目标位置是否为机器人可到达点位	
语法说明	sta,joint = Targetok(A,“B”) --判断目标位置是否能够按照指定路径到达 sta,joint = Targetok(A) --判断目标位置能否按照当前手系到达	
参数说明	A	目标点位，table 类型数据
	B	可选参数，left/right/short/now
		注：short 主要是针对吊装机器人，不管什么路径只
		到达，都会返回 sta=0，并且返回最优路径的关节坐

返回值	sta	是否为可到达标志位，sta=0 为可到达，sta=1 为不可
	joint	如果可到达，输出对应路径下的关节坐标

举例说明

1. sta,joint = Targetok(p1) --判断 p1 点能否按照当前手系到达
   --sta,joint=Targetok(p1,“left”) --判断 p1 点能否以左手系到达
   --sta,joint=Targetok(p1,“right”)--判断 p1 点能否以右手系到达
   -- sta,joint = Targetok(p1,“short”) --判断 p1 点能否到达，并
   --返回最优路径的关节坐标
   --sta,joint=Targetok(p1,“now”)--判断 p1 点能否以当前手系到达

```
if sta == 0 then
    print(“p1 点为机器人可到达点位”)
    print(“joint=”,joint.x,joint.y.joint.z,joint.c)
     --返回对应路径下的关节坐标
elseif sta== 1 then
    print(“p1 点位为机器人不可到达点位，请检查”)
    Pause()     --暂停
end
```

2. pos = {x=300,y=10,z=-20,c=30,h=1}
   sta = Targetok(pos) --判断 pos 点是否为机器人可到达点位
   --sta,joint= (pos,”left”) --判断 pos 点能否以左手系到达
   --sta,joint=Targetok(pos,”right”)--判断 pos 点能否以右手系到达
   -- sta,joint = Targetok(pos,”short”) --判断 pos 点能否到达，返回最优路径
   --sta,joint=Targetok(p1,”now”)--判断 p1 点能否以当前手系到达

```
if sta == 0 then
    print(“pos 点为机器人可到达点位”)
    print(“joint=”,joint.x,joint.y.joint.z,joint.c)
elseif sta== 1 then
    print(“pos 点位为机器人不可到达点位，请检查”)
    Pause()   --暂停
```

```
    end
3.  local pos = { }
local n,data,err=CCDrecv("CAM1")--接收视觉 CAM1 数据
     if err == 0 and n ~=nil then
for i=1,ndo        --n 决定循环的次数
    print(i,data[i][1],data[i][2],data[i][3]) --打印每组数据
    if data[i][1]~=0 or data[i][2]~=0then--判断数据不同时为零
        pos.x=data[i][1]--数据 data[i][1]赋值给 pos.x
        pos.y=data[i][2]--数据 data[i][2]赋值给 pos.y
        pos.z=0
        pos.c=data[i][3]--数据 data[i][3]赋值给 pos.c
                pos.h = 0
                sta = Targetok(pos)
-- sta,joint = Targetok(pos, "left")     --判断 pos 点能否以左手系到达
 -- sta,joint = Targetok(pos, "right")     --判断 pos 点能否以右手系到达
 -- sta,joint = Targetok(pos, "short")     --判断 pos 点能否以最优路径到达
 -- sta,joint = Targetok(pos, "now")    --判断 pos 点能否以当前手系到达
                if sta == 0 then
                    print("pos 点为机器人可到达点位")
                    print("joint=",joint.x,joint.y.joint.z,joint.c)
                    MovP(pos)--点到点方式运动到点 pos
    elseif sta== 1 then
    print("pos 点位为不可到达点位，请检查")
    Pause()   --暂停运动
    end
```

```
end
        end
    elseif err==0 and n == nil then
        print("数据格式配置错误")
    elseif err~=0 then
        print("网线断开或 IP 配置错误或网络超时")
    end
```

注：判断目标点是否为机器人可到达点位时，一定要指定目标点的手系，因为对于四轴机器人来说有些点位只有一个手系可以到达。

SetI		
使用说明	设置当前最佳加速度负载惯量使用组别	
语法说明	SetI(A)	
参数说明	A	最佳加速度负载惯量使用组别号，范围 0~4
举例说明	SetI (1) --负载惯量组别号 1，也就是调用最佳加速度参数中的辨识出的负载转动惯量 MovL(p1)	

注：此指令的具体使用可参考《最佳加速度设置》文档。

3.1.17 码垛指令

3.1.17.1 矩形编程码垛

指令符号	指令说明
Set_Plt	设置码垛数指令（用于编程码垛）
Get_Plt	获取码垛点位指令(编程码垛、配置码垛均可)

三点码垛 Set_Plt	
使用说明	设置码垛参数
语法说明	单层码垛 Set_Plt(NO,3,org,px,py,nx,ny)

多层码垛　Set_Plt(NO,3,org,px,py,pz,nx,ny,nz)

注：多层码垛中，当 nz=1 时，pz 点默认无效为单层码垛。

参数说明　**下表各指令参数说明**

单层码垛　Set_Plt(NO,3,org,px,py,nx,ny)

NO	码垛号（number 类型数据或者字符型数据）
org	码垛原点位置
px	码垛行方向的最后一个点
py	码垛列方向的最后一个点
nx	码垛的行数
ny	码垛的列数

多层码垛　Set_Plt(NO,3,org,px,py,pz,nx,ny,nz)

NO	码垛号（number 类型数据或者字符型数据）
org	码垛原点位置
px	码垛行方向的最后一个点
py	码垛列方向的最后一个点
pz	码垛层方向的最后一个点
nx	码垛的行数
ny	码垛的列数
nz	码垛的层数

举例说明

1. Set_Plt(1,3,p1,p2,p3,5,11)　--设置 XY 轴码垛
2. Set_Plt(1,3,p1,p2,p3,p4,11,5,3)　--设置 XYZ 轴码垛

◆ 注意：码垛设置指令只要在程序开始执行一次，指令自动根据参数的个数判断是 XY 轴码垛或 XYZ 轴码垛。指令里的“3”指的是三点码垛。

四点码垛 Set_Plt

使用说明　设置码垛参数

语法说明　单层码垛　Set_Plt(NO,4,org,px,py,pxy,nx,ny)

多层码垛　Set_Plt(NO,4,org,px,py,pxy,pz,nx,ny,nz)

注：多层码垛中，当 nz=1 时，pz 点默认无效为单层码垛。

参数说明

下表各指令参数说明

单层码垛 Set_Plt(NO,4,org,px,py,pxy,nx,ny)

NO	码垛号（number 类型数据或者字符型数据）
org	码垛原点位置
px	码垛行方向的最后一个点
py	码垛列方向的最后一个点
pxy	码垛第一层对角线的点位
nx	码垛的行数
ny	码垛的列数

多层码垛 Set_Plt(NO,4,org,px,py,pxy,pz,nx,ny,nz)

NO	码垛号（number 类型数据或者字符型数据）
org	码垛原点位置
px	码垛行方向的最后一个点
py	码垛列方向的最后一个点
pxy	码垛第一层对角线的点位
pz	码垛层方向的最后一个点
nx	码垛的行数
ny	码垛的列数
nz	码垛的层数

举例说明

1. Set_Plt(1,4,p1,p2,p4,p5,5,4) --单层码垛
2. Set_Plt(1,4,p1,p2,p3,p4,p5,5,4,3) --多层码垛

◆ 注意：设置码垛指令里的“4”指的是四点码垛。

Get_Plt

使用说明 获得码垛数据点

语法说明

按码垛序号获取单层/多层码垛 pos=Get_Plt(NO,num)

按行列获取单层码垛 pos=Get_Plt(NO,i,j)

按行列层获取多层码垛 pos=Get_Plt(NO,i,j,k)

参数说明

返回值返回码垛中的各个点位数据

下表各指令参数说明

用法 1：按码垛序号获取单层/多层码垛 pos=Get_Plt(NO,num)

NO	码垛号（number 类型数据或者字符型数据）
num	码垛序号

1. Set_Plt(1,4,p1,p2,p3,p4,3,4) --4 点单层码垛
   pos=Get_Plt(1,5) --获取第五个码垛点
2. pos=Get_Plt(“Pallet”,5)
   --获取配置码垛里名为 Pallet 码垛的第 5 个点

注：□ 按序号取码垛点时，是先走行再走列，最后走层。

用法 2：按行列获取单层码垛 pos=Get_Plt(NO,i,j)

NO	码垛号（number 型或者字符型数据）
i	码垛行数，从 1 开始
j	码垛列数，从 1 开始

举例说明

1. Set_Plt(1,4,p1,p2,p3,p4,3,4) --4 点单层码垛
   pos=Get_Plt(1,2,3) --获取第 2 行第 3 列码垛点
2. pos=Get_Plt(“Pallet”,2,3)
   --获取配置码垛里名为 Pallet 码垛的第 2 行第 3 列点

用法 3：按行列层获取多层码垛 pos=Get_Plt(NO,i,j,k)

NO	码垛号（number 型或者字符型数据）
i	码垛行数，从 1 开始
j	码垛列号，从 1 开始
K	码垛层号，从 1 开始

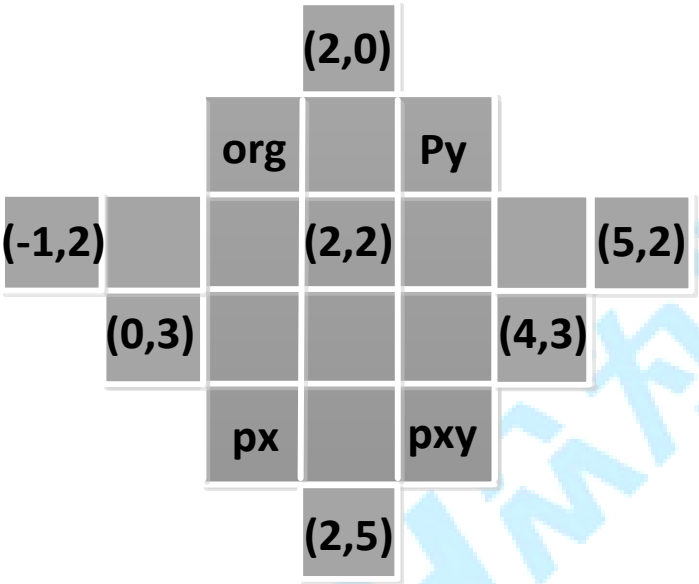
举例说明

1. Set_Plt(1,4,p1,p2,p3,p4,p5,3,4,2) -- 4 点多层码垛
   pos=Get_Plt(1,2,3,1) --获取第 1 层第 2 行第 3 列码垛点
2. pos=Get_Plt(“Pallet”,2,3,1)
   --获取配置码垛里名为 Pallet 码垛第 1 层第 2 行第 3 列码垛点

注：刚才介绍的 3 种获取码垛点的用法均是只能获取示教区域内的点，若不在示教区域内则会提示获取码垛参数超出设定范围。针对实际应用中当码盘部分

边角点不可示教时，码盘需要分割成多个码垛的情况，增加了用法 4 与用法 5。

例如如下码盘：



用法 4：按行列获取单层码垛示教区以外的点

pos=Get_Plt(NO,i,j,”outside”)

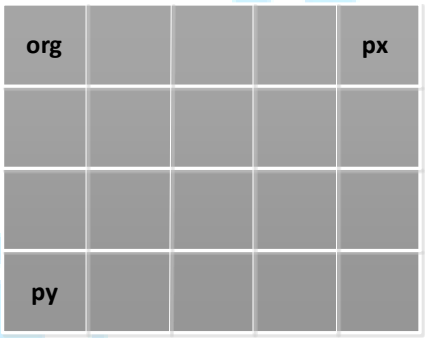
	NO	码垛号（number 型或者字符型数据）
	i	码垛行数，从 1 开始
	j	码垛列号，从 1 开始
	outside	**获取示教区域外点的关键字，不可更改**
举例说明	1. Set_Plt(1,4,org,px,py,pxy,4,3) -- 4 点单层码垛 pos=Get_Plt(1,2,0,”outside”) --获取第 2 行第 0 列(示教码垛区域列负方向第一个)码垛点 pos=Get_Plt(1,-1,2,”outside”) 2. pos=Get_Plt(“Pallet”,2,5,”outside”) --获取配置码垛里名为 Pallet 码垛区域外第 2 行第 5 列码垛点	

用法 5：按行列层获取多层码垛示教区以外的点

pos=Get_Plt(NO,i,j,k,”outside”)

	NO	码垛号（number 型或者字符型数据）
	i	码垛行数，从 1 开始
	j	码垛列号，从 1 开始
	K	码垛层数，从 1 开始
	outside	**获取示教区域外点的关键字，不可更改**
举例说明	1. Set_Plt(1,4,org,px,py,pxy,pz,4,3,2) -- 4 点两层码垛 pos=Get_Plt(1,2,0,3,”outside”) --获取第 2 行第 0 列(示教码垛区域列负方向第一个) 第 3 层码垛点 pos=Get_Plt(1,-1,2,0,”outside”) 2. pos=Get_Plt(“Pallet”,2,5,0,”outside”) --获取配置码垛里名为 Pallet 码垛的区域外第 2 行第 5 列第 0 层(示教码垛区域层负方向第一个)码垛点	

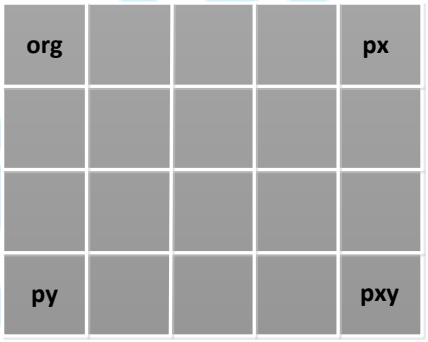
案例 1. 三点法矩形编程码垛




```
local posready = p1              --待机点
local place = p2                --放料点
local org = p3                  --码垛原点
local px = p4                   --码垛行方向最后一个
local py = p5                   --码垛列方向最后一个
local pz = p7                   --码垛最后一层对角的点位
local nx,ny,nz = 5,4,3          --码垛的行数，列数，层数
local pos = {}
Set_Plt(1,3,org,px,py,pz,nx,ny,nz)        --设置单层码垛
```

```
MArchP(posready,0,10,10)                    --运动到待机点
while true do
   for k = 1,nz do
       for j = 1,ny do
           for i = 1,nx do
       pos = Get_Plt(1,i,j,k)              --获取码垛点位
       print(pos.x,pos.y,pos.z,pos.c)
       MArchP(pos,0,10,10)            --运动到码垛位置
      MArchP(place,0,10,10)         --运动到放料点
end
       end
   end
   Delay(20)
end                                        --主程序结束
```

案例 2.四点法矩形编程码垛




```
local posready = p1                        --待机点
local place = p2                           --放料点
local org = p3                             --码垛原点
local px = p4--码垛行方向最后一个
local py = p5--码垛列方向最后一个
local pxy = p6--码垛原点对角线位置
local nx,ny = 5,4
```

```
local pos = {}
Set_Plt(1,4,org, px,py,pxy,nx,ny)      --设置单层码垛
while true do
    for i =1, nx*ny do
        pos = Get_Plt (1,i)                    --获取码垛点位
        print(pos.x,pos.y,pos.z,pos.c)
        MArchP(pos,0,10,10)
    MArchP(place,0,10,10)
    end
     Delay(20)
end
```

- 注意：当使用编程设置矩形码垛时 Get_Plt 需与 Set_Plt 组合使用，但若用界面配置

  码垛时 GetPLTPos 须与 InitPlts 组合使用(后续章节 1.17.3“配置码垛”中详述)；
- 编程码垛是通过 AR 程序实现的，码垛中涉及到的参数（例如：码垛名、行数、列数等）都是在 AR 程序中设定的，故称作编程码垛；
- 编程码垛须添加码垛库(Get_Plt 和 Set_Plt 这两个指令是在 pallet.lib 库里面封装定义的）；
- 四点法码垛相对于三点法码垛须多示教一个码垛点位，但是精度高于三点法；故在某些应用中出现码盘对角线位置附近的某些点出现偏差时须考虑用四点法码垛来替代三点法码垛。

3.1.17.2 圆弧编程码垛

指令符号	指令说明
Set_ArcPlt	圆弧码垛设置
Get_ArcPlt	获取圆弧码垛当前位置

Set_ArcPlt	
使用说明	设置圆形码垛参数
语法说明	单层整圆　Set_ArcPlt(NO,po,pst,qty,cor) 单层圆弧 Set_ArcPlt(NO,po,pst,ped,qty,cor) 多层圆弧　Set_ArcPlt(NO,po,pst,ped,pz,qty,la,cor)

参数说明

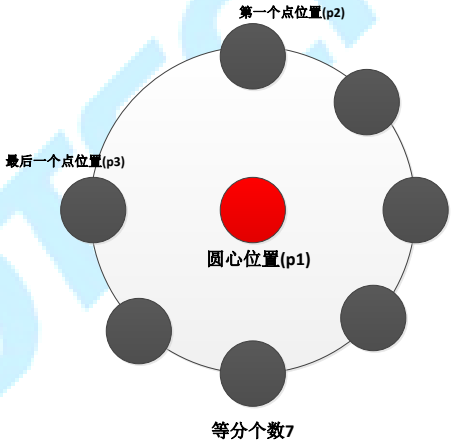
下表各指令参数说明

单层整圆 Set_ArcPlt(NO,po,pst,qty,cor)	
NO	码垛号（number 型或者字符型数据）
po	码垛原点，即圆弧码垛的圆心位置
pst	码垛起点位置
qty	法方向码垛个数
cor	圆弧方向(0 为逆时针，1 为顺时针)
单层圆弧(包括整圆)Set_ArcPlt(NO,po,pst,ped,qty,cor)	
NO	码垛号（number 型或者字符型数据）
po	圆弧的圆心位置
pst	码垛的起点
ped	码垛的终点
qty	法方向等分个数
cor	圆弧方向(0 为逆时针，1 为顺时针)
多层圆弧 Set_ArcPlt(NO,po,pst,ped,pz,qty,la,cor)	
NO	码垛号（number 型或者字符型数据）
po	圆弧第一层的圆心位置
pst	码垛第一层起点位置
ped	码垛第一层终点位置
pz	Z 方向最后一层起点位置
qty	法向方向码垛等分个数（第一层起点与终点之间沿着圆周方向的间隔数）
la	空间码垛层数
cor	圆弧方向(0 为逆时针，1 为顺时针)

举例说明

1. Set_ArcPlt(1,p1,p2,7,0) --设置 XY 轴码垛，如下图所示
2. Set_ArcPlt(1,p1,p2,p3,7,0) --设置 XY 轴圆弧码垛，如下图所示
3. Set_ArcPlt(1,p1,p2, p3, p4,7,3,1) --设置空间（多层）码垛

◆ 注意：圆形码垛设置指令只要在程序开始执行一次，指令自动根据参数的个数判断是 XY 轴码垛或 XYZ 轴码垛。

Get_ArcPlt	
使用说明	获取圆形码垛上每个点的位置
语法说明	pos=Get_ArcPlt(NO,num) pos= Get_ArcPlt(NO,i,k)
参数说明	**下表各指令参数说明** **用法 1：**以码垛序号获取码垛 pos=Get_ArcPlt(NO,num) NO：码垛号（number 型或字符型数据） num：从第一个位起，码垛序号 **用法 2：**以第几排第几层获取码垛号 pos= Get_ArcPlt(NO,i,k) NO：码垛号（number 型或字符型数据） i：从第一个位置点起的第几个位置 k：第几层
举例说明	 SetU(1) --以圆形码盘上的六个点建立的圆形用户坐标系 1 --若圆弧码垛的圆心未知，则一定先以圆形码盘建立一个圆形用户坐标--系（六点法）,并在该坐标系下示教圆弧的圆心、起点、终点；若圆心--位置已知，则不必建立圆弧坐标系，直接在默认同一坐标系下示教圆--弧的圆心、起点和终点即可。 local posready = p10 --待机点（用户 1 下示教） local place = p1 --放料点（用户 1 下示教）

```
local Arc_org={x=0,y=0,z=0,c=0} --码垛圆心位置(用户 1 原点)
local Arc_start = p2     --码垛第一层起点位置（用户 1 下示教）
local Arc_end = p3       --码垛第一层终点位置（用户 1 下示教）
local Arc_layer = p4 --码垛最后一层终点位置（用户 1 下示教）
local qty = 7    --起点(p2)和终点(p3)之间(顺时针)的等分数 7
local layer = 6 --圆形码垛层数（p2 点与 p4 点之间的间隔数）
local cor = 0       --圆弧码垛的方向（逆时针）
Set_ArcPlt(1,Arc_org,Arc_start,Arc_end,Arc_layer,qty,layer,cor)
--设置多层圆形码垛
MArchP(posready,0,10,10)
while true do
    i = 1
    j = 1
    for j =1,layer do
fori=1,qty do
        pos=Get_ArcPlt(1,i,j,0) --读取码垛点位数据（逆时针）
            MArchP(pos,0,10,10)          --运动到码垛位置
            MArchP(place,0,10,10)        --运动到放料点
        end
    end                          --主程序结束
    Delay(20)
end
```

- 注意：当使用编程设置圆弧码垛时 Get_ArcPlt 指令与 Set_ArcPlt 指令是组合使用，但若使用界面配置圆弧码垛时 Get_ArcPlt 需与 InitPlts 组合使用；
- 配置码垛须添加码垛库（Get_ArcPlt 与 Set_ArcPlt 这两个指令是在 **pallet.lib** 库里面封装定义的）。
- 若圆弧码垛的圆心未知，则一定先以圆形码盘建立一个圆形用户坐标系（六点法）,并在该坐标系下示教圆弧的圆心、起点、终点；若圆心位置已知，则不必建立圆弧坐标系，直接在默认同一坐标系下示教圆弧的圆心、起点和终点即可。

3.1.17.3 配置码垛

指令符号	指令说明
GetPLTPos	获取码盘是否码满、码盘当前码垛的个数以及码垛点位置信息（适用于配置码垛）
ResetPLT	重置码垛个数（适用于配置码垛）
InitPlts	加载配置码垛

GetPLTPos		
使用说明	获取码盘是否码满、码盘当前码垛的个数以及码垛点位置信息	
语法说明	flag,num,pos=GetPLTPos(“PltName”)	
参数说明	无返回值	
	输入参数说明	
	PltName	码垛名（默认 PLT0~PLT9）
	flag	码盘是否码满的标志位
		0 表示没有码满
		1 表示码满
		2 表示最后一个为不码垛点,码满的情况
	num	当前码垛的个数
	pos	当前码垛点的位置

注：使用该语句，须在码垛配置中设置关于码垛的一些参数，包括码垛名、码垛顺序、参考坐标系等。

ResetPLT	
使用说明	重置码垛个数
语法说明	ResetPLT(“PltName”,count)
参数说明	**返回值**
	无

	输入参数说明	
	PltName	码垛名（默认 PLT0~PLT9）
	count	码垛个数，可任意设置个数，码垛个数从 1 开始

InitPlts		
使用说明	加载配置界面	
语法说明	InitPlts(“PltName”)	
参数说明	**返回值**	
	无	
	输入参数说明	
	PltName	配置码垛名，默认为 pallet
举例说明	local posReady = p1 --待机点&安全点 local quliao = p2 --取料点 local flag,num local pos ={} InitPlts() --加载默认配置文件 MArchP(posReady,5,1,1) --待机点&安全点 while true do MArchP(quliao,0,10,10) --运动到取料点 DO(1,ON) flag,num,pos =GetPLTPos(“PLT0”) --获取码垛点 if flag == 0 then --未码完 MArchP(pos,0,10,10) --运动到码垛点 DO(1,OFF) elseif flag ==1 then --码满 MArchP(pos,0,10,10) --运动到码垛点 DO(1,OFF) ResetPLT(“PLT0”,1) --重置码垛	

```
        elseif flag==2 then --码盘的最后一个点为不码垛点，码完的返回值
            ResetPLT("PLT0",1)   --重置码垛
end
        Delay(20)
end
```

- 注意：GetPLTPos 指令、ResetPLT 指令以及 InitPlts 指令是组合配套使用的；
- 在使用 GetPLTPos 指令前必须先要使用 InitPlts 加载配置的文件；
- 若使用配置码垛，除可以利用 GetPltPos 获取配置码垛点位外，还可以利用 Get_Plt 或者 Get_ArcPlt，由码垛形状决定。当使用 Get_Plt 或 Get_ArcPlt 获取指定码垛点时，遇到不码垛点返回空值；
- 配置码垛是通过在**码垛配置界面**设定码垛工艺用到的参数（例如：码垛名、行数、列数、间隔数等），然后结合 AR 程序实现码垛工艺，故称作配置码垛；
- 配置码垛须添加码垛库（GetPLTPos 与 ResetPLT 这两个指令是在 pallet.lib 库里面封装定义的）。

3.1.18 伺服管理指令

指令符号	指令说明
MotOn	伺服所有轴打开使能
MotOff	伺服所有轴关闭使能
DragMode	AR 程序中实现拖拽功能

MotOn	
使用说明	伺服所有轴使能打开
语法说明	MotOn()
参数说明	无参数
举例说明	MotOn()　　--所有轴使能打开

MotOff	
使用说明	伺服所有轴使能关闭

语法说明	MotOff()	
参数说明	无参数	
举例说明	MotOff()	--所有轴使能关闭

DragMode	
使用说明	AR 程序实现拖拽功能
语法说明	DragMode()
参数说明	无参数
举例说明	MovP(p1) MotOff() --先断开断使能 Delay(10) --延时 10ms DragMode() --再切换到拖拽模式

注：AR 程序中实现自动模式下拖拽模式的切换，一定要在断使能的情况下使用。

3.1.19 串口通信指令

指令符号	指令说明
RecCom	从 RS232/RS485 串口接收数据
SendCom	往 RS232/RS485 串口发送数据
ClsCom	关闭 RS232/RS485 串口
InitCom	初始化 RS232/RS485 串口参数
ClrCom	清除 RS232/RS485 串口缓存

RecCom		
使用说明	无协议 RS232/RS485 串口通讯接收数据	
语法说明	Err,RecBuf=RecCom(**A, "Time**=B")	
参数说明	**指令参数说明**	
	A	**如果是 RS232，串口通讯端口号 A 为 1，如果是 RS485，串口通讯端口号 A 为 2。**

	B	接收数据超时时间，单位为毫秒	
	返回值		
	RecBuf	接收数据缓冲区	
	Err	0	表示接收成功
		非 0	表示接收超时
举例说明	local RecBuf --定义接收缓冲区 local Err --定义接收错误号 Err,RecBuf = RecCom(1,"Time=5000") --串口 1 接收数据，接收超时时间为 5 秒 print(RecBuf.buff) --RecBuf 缓冲区中接收到上位机发送的数据		

注：

- 若通过 RecCom 指令接收到的数据为字符串类型，则数据存储在缓冲区 RecBuf 的 buff 变量中；调用方法为：**RecBuf.buff**
- 若通过 RecCom 指令接收到的数据为十六进制，则数据存储在数组 RecBuf 变量中；调用方法为：**RecBuf[1] 、RecBuf[2]…..**

SendCom		
使用说明	无协议 RS232/RS485 串口通讯发送数据	
语法说明	SendCom(A, “B”)	
参数说明	**指令参数说明**	
	A	**如果是 RS232，串口通讯端口号 A 为 1，如果是 RS485，串口通讯端口号 A 为 2。**
	B	发送的数据，可以是ASCII码类型，也可以使TABLE十六进制类型数据，指令自动判断参数类型进行发送
举例说明	1. local SendBuf={0x01,0x05,0x00,0x1A,0xff,0x00} SendCom(1,SendBuf) --串口 1 以十六进制方式发送数据 2. SendCom(1,“ROBOT”) --发送字符串 3. pos =Getpos() SendCom(1,string.format(“%f,%f,%f,%f;”,pos.x,pos.y.pos.z,pos.c)) --将数值变量 pos.x,pos.y,pos.z,pos.c 以字符串形式发送	

SendCom(1, “abc,123,34,56”)　--发送字符串

ClsCom		
使用说明	无协议 RS232/RS485 串口通讯关闭串口指令	
语法说明	ClsCom(A)	
参数说明	A	**如果是 RS232，串口通讯端口号 A 为 1，如果是 RS485，串口通讯端口号 A 为 2。**
举例说明	ClsCom(1)　　--关闭 RS232 串口	

InitCom		
使用说明	初始化 RS232/RS485 串口参数	
语法说明	err = InitCom(A)	
参数说明	A	**--如果是 RS232，串口通讯端口号 A 为 1，如果是 RS485，串口通讯端口号 A 为 2。**

举例说明	err = InitCom(A)　　--判断初始化串口是否成功，如果返回 0 则成功，非 0 则不成功。
	print(“err=”, err)
	if err ==0 then
	print(“初始化串口成功”)
	else
	print(“初始化串口失败”)
	end	

ClrCom	
使用说明	清除 RS232/RS485 串口缓存
语法说明	ClrCom(A)

参数说明	A	**--如果是 RS232，串口通讯端口号 A 为 1，如果是 RS485，串口通讯端口号 A 为 2。**

3.1.20 RAM 地址读写指令

指令符号	指令说明
publicread	读取全局表（RAM 数据表）数据值
publicwrite	写入数据值到全局表（RAM 数据表）

publicread		
使用说明	读取全局表（RAM 数据表）数据值	
语法说明	用法 1：C= publicread(A,“B”) 用法 2：C = publicread(A,len,“B”)	
参数说明	**指令参数说明**	
	A	全局数据表中的地址
	B	可选参数，读取数据的方式，整型（可省略）/浮点型/十六进制
	len	连续读取地址的个数
	返回值	
	C	读取全局表中地址对应的数据值
举例说明	1. local a = publicread(0x100,"float") --AR 以浮点型方式读地址 0x100 的值 2. local b = publicread(0x102) --AR 以整型方式读地址 0x102 的值 3. local c = publicread(0x100,3) --从 0x100 开始连续读取 3 个地址（0x100,0x102,0x104）的数值(整型),返回的值存储在数组 c 中， c[1]为地址 0x100 里面的值，依次类推 4. local c = publicread(0x100,3, “float”) --从 0x100 开始连续读取 3 个地址（0x100,0x102,0x104）的数值(浮点型),返回的值存储在数组 c 中， c[1]为地址 0x100 里面的值，依次类推	

publicwrite	
使用说明	写入数据值到全局表（RAM 数据表）
语法说明	用法 1：publicwrite(A,B,"D") 用法 2：publicwrite(A,C, “D”)
参数说明	返回值无返回值 **指令参数说明** A：全局表数据值的地址 B：写入到对应地址的数据 C：写入到连续地址中的数组 D：写入数据的方式，整型（可省略）/浮点型/十六进制
举例说明	1. publicwrite(0x102,10.5,"float") --浮点型方式写入到地址 0x100 的值 2. publicwrite(0x102,5) --以整型方式写入到地址 0x102 的值 3. publicwrite(0x100,{100,200,300}) –将数组{100,200,300} (整型)分别写入到以 0x100 为起始地址的连续 3 个地址中（0x100,0x102,0x104）） 4. publicwrite(0x100,{10.1,200.1,300.1}, “float”) – 将数组{10.1,20.1,30.1} (浮点型)分别写入到以 0x100 为起始地址的连续 3 个地址中（0x100,0x102,0x104））

3.1.21 网络通讯指令

指令符号	指令说明
InitNet	网络通讯初始化
SendNet	机器人发送字符串数据给外部设备（例如 PLC、视觉）
CCDtrigger	触发视觉拍照
CCDrecv	接收外部设备（例如视觉）返回的坐标数据
ReadNet	接收外部设备（例如 PLC、视觉）发送的字符串类型数据

InitNet		
使用说明	网络通讯初始化	
语法说明	sta =InitNet(“CamName”)	
参数说明	CamName	视觉名称或传送带名称(字符串类型)，在 UI 配置时生成； 网络名称默认为 NET0、NET1… 视觉名称默认为 CAM0、CAM1… 传送带名称默认为 CON0、CON1…
返回值	sta	网络连接状态
		sta = 0：连接成功
		sta = 1：连接失败或无连接对象
举例说明	1. InitNet(“CAM0”) --视觉网络通讯初始化 2. InitNet(“CON0”) --传送带网络通讯初始化 3. InitNet(“NET0”) --网络通讯初始化	

SendNet		
使用说明	机器人发送字符串数据给外部设备（例如 PLC，视觉）	
语法说明	sta = SendNet(“Name”,Buff)	
参数说明	Name	网络名称、视觉名称或传送带名称(字符串类型)，在 UI 配置时生成； 网络名称默认为 NET0、NET1… 视觉名称默认为 CAM0、CAM1… 传送带名称默认为 CON0、CON1…
	Buff	发送的字符串
返回值	sta	数据发送状态
		sta=0: 数据发送成功
		sta=1: 当前网络断开或发送数据超时

举例说明	1. SendNet(“CAM2”, “123”) 2. SendNet(“CAM1”, “[0]”) 3. SendNet(“CAM0”, “0,0,0,0;”)

CCDtrigger

使用说明	触发视觉拍照	
语法说明	sta = CCDtrigger(“Name”)	
参数说明	Name	视觉名称或传送带名称(字符串类型)，在 UI 配置时生成； 视觉名称默认为 CAM0、CAM1… 传送带名称默认为 CON0、CON1…
返回值	sta	触发拍照状态
		sta=0: 触发拍照成功
		sta=1: 触发拍照失败
举例说明	1. CCDtrigger(“CAM0”)	

CCDrecv

使用说明	接收外部设备（例如视觉）返回的坐标数据	
语法说明	n,data,err,recbuf =CCDrecv(“Name”)	
参数说明	Name	网络名称、视觉名称或传送带名称(字符串类型)，在 UI 配置时生成； 网络名称默认为 NET0、NET1… 视觉名称默认为 CAM0、CAM1… 传送带名称默认为 CON0、CON1…
返回值	n	接收到外部设备发送数据的组数（原始字符串数据 recbuf 按照指定数据格式解析）
	data	接收到外部设备发送的每组数据(table 类型数据)组成的二维数组，可直接用于点位运动

	err	错误号： 0 为正常返回，非 0 为异常返回
	recbuf	接收到外部设备发送的字符串数据

举例说明

```
local pos = {}
Initnet("CAM0")                    --网络初始化
SendNet("CAM0","0,0,0,0;")      --触发视觉拍照
--CCDtrigger("CAM0")                --触发视觉拍照
local n,data,err,recbuf=CCDrecv("CAM0")        --接收视觉 CAM0
的数据
if err == 0 and n ~= nil then
for i=1,ndo                        --n 决定循环的次数
    print(i,data[i][1],data[i][2],data[i][3])     --打印每组数据
    if data[i][1]~=0 or data[i][2]~=0then     --判断数据不同时为零
        pos.x=data[i][1]       --数据 data[i][1]赋值给 pos.x
        pos.y=data[i][2]       --数据 data[i][2]赋值给 pos.y
        pos.z=0
        pos.c=data[i][3]       --数据 data[i][3]赋值给 pos.c
            MovP(pos)          --点到点方式运动到点 pos
        end
    end
elseif err==0 and n == nil then
    print("数据格式配置错误")
elseif err == nil then
    print("网线断开或 IP 配置错误或网络超时")
end
```

ReadNet

使用说明	接收外部设备（例如 PLC、视觉）发送的字符串类型数据
语法说明	RecBuf,err=ReadNet("Name")

参数说明	CamName	网络名称、视觉名称或传送带名称(字符串类型)， 在 UI 配置时生成； 网络名称默认为 NET0、NET1… 视觉名称默认为 CAM0、CAM1… 传送带名称默认为 CON0、CON1…
返回值	RecBuf	接收外部设备（例如 PLC，视觉）返回的数据
	err	错误号： 0 为正常返回，非 0 为异常返回

举例说明	local RecBuf,err = ReadNet(“NET0”)
	if err == 0 then
	　if RecBuf == “OK” then
	　　MovP(p1)
	　elseif RecBuf == “NG” then
	　　MovP(p2)
	　end
	end	

3.1.22 传送带指令

指令符号	指令说明
CnvQueInit	传送带跟踪参数初始化
CnvQueAdd	发送数据给控制器，将坐标数据存入缓存队列待跟随
CnvQueGet	监控物体在传送带上的位置，是否进入抓取区
CnvQueDel	删除位置抓取队列中的首元素
MissQueAdd	超出抓取区未抓取的物体位置信息存入到丢失队列
CnvSynCatch	执行同步跟随
SynOver	结束同步跟随
CnvFastCatch	执行快速抓取（没有同步过程）
LockExcoder	传送带软触发单次记录编码器值并下发到队列

CnvQueInit

使用说明	传送带跟踪参数初始化	
语法说明	CnvQueInit(CnvName)	
参数说明	CnvName	传送带名称(字符串类型)，在 UI 配置时生成
返回值	无返回值	
举例说明	1. CnvQueInit (“CON0”) --初始化传送带 CON0 的参数	

CnvQueAdd		
使用说明	发送数据给控制器，将坐标数据存入缓存队列待跟随	
语法说明	CnvQueAdd(CnvName,QueNo,pos)	
参数说明	CnvName	传送带名称(字符串类型)，在 UI 配置时生成
	QueNo	抓取队列号（数字类型）
	pos	存入抓取队列的位置坐标（table 类型）

举例说明	local n,flag
	local data
	local pos={x=0,y=0,z=0,c=0,no=0}
	local putpos={x=50,y=-10,z=-60,c=90,no=0} --最好要给模版号赋值
	InitNet("CON0") --步骤 1：初始化网络 IP 以及端口
	InitNet ("CON1")--步骤 1：初始化网络 IP 以及端口
	CnvQueInit("CON0") --传送带跟踪参数初始化
	CnvQueInit("CON1") --传送带跟踪参数初始化
	while 1 do
	------视觉传送带,取料----------
	n,data=CCDrecv("CON1")
	if data then
	for i=1,n do
	if data[i][1]~=0 or data[i][2]~=0 then
	print("cnv2:",data[i][1],data[i][2],data[i][3])	

```
                --pos.pulse=tonumber(data[i].AA)
                    pos.x=data[i].x
                    pos.y=data[i].y
                pos.z=-50
                    pos.c=0
                    pos.no=tonumber(data[i].NO)
                CnvQueAdd("CON1",2,pos)
                flag=1;
            end
        end
    end
    -----无视觉传送带,放料----------------
    if flag==1 then
        flag=0
        Delay(100)
        CnvQueAdd("CON0",1,putpos)
    end
end
```

CnvQueGet		
使用说明	监控已入队列物体的实时位置，进入抓取区域时数据	
语法说明	flag,data=CnvQueGet(CnvName,QueNo)	
参数说明	CnvName	传送带名称(字符串类型)，在 UI 配置时生成
	QueNo	抓取队列号（数字类型）
返回值	flag	flag=0，物体超出抓取区
		flag=1，物体进入抓取区并返回数据
		flag = nil，物体还未进入抓取区或队列无位置数据
	data	当物体进入抓取区域，按先入先出原则获取位置数

		据

举例说明	while 1 do
	Delay(2)
	flag,data=CnvQueGet("CON0",1) --1 号传送带,按照先入先出的原则获取 1 号队列内进入抓取区的位置数据，超出抓取区域的位置数据重新存入丢失队列，返回 flag=0
	if flag==0 then --超出抓取区
	MissQueAdd(data) --存入下发从机队列
	
	elseifflag==1 then
	break
	end
	flag,data=CnvQueGet("CON1",2) --2 号传送带
	if flag==0 then
	MissQueAdd(data) --存入下发从机队列
	elseifflag==1 then
	break
	end
	end	

CnvQueDel		
使用说明	删除已入队列的位置数据中的首元素，也就是删除队列里的第一个数据	
语法说明	CnvQueDel(CnvName,QueNo)	
参数说明	CnvName	传送带名称(字符串类型)，在 UI 配置时生成
	QueNo	抓取队列号（数字类型）
举例说明	CnvQueDel("CON0",1)	

MissQueAdd		
使用说明	超出抓取区未抓取的物体位置信息存入到丢失队列	
语法说明	MissQueAdd(data)	
参数说明	data	超出抓取区未抓取的工件位置信息

CnvSynCatch		
使用说明	执行同步跟随	
语法说明	**形式 1：** sta = CnvSynCatch(pos,limit,up,down) **形式 2：** sta = CnvSynCatch(pos)	
参数说明	pos	进入抓取区的工件位置信息
	limit	抓取工件时机器人 Z 轴的限位高度
	up	抓取工件时机器人 Z 轴从等待位置上升的高度
	down	抓取工件时机器人 Z 轴相对于抓取位置上方下降的高度
	sta	sta = 2: 进入同步
		sta = 3: 错误退出

SynOver	
使用说明	结束跟随（待完成同步中工艺后，可执行此命令进行退出同步）
语法说明	SynOver()
参数说明	无参数
返回值	无返回值

CnvFastCatch	
使用说明	执行快抓跟随
语法说明	**形式 1：** sta = CnvFastCatch(pos,limit,up,dowm) **形式 2：** sta = CnvFastCatch(pos) --点动运动

参数说明	pos	进入抓取区的位置信息
	limit	抓取工件时机器人 Z 轴的限位高度
	up	抓取工件时机器人 Z 轴从等待位置上升的高度
	down	抓取工件时机器人 Z 轴相对于抓取位置上方下降的高度
返回值	sta	sta =2: 可以抓取
		sta = 3: 错误退出

LockExcoder	
使用说明	传送带软触发单次记录编码器值并下发到队列
语法说明	**LockExcoder(1)**--记录 1 号传送带的编码器值 **LockExcoder(2)** --记录 2 号传送带的编码器值
参数说明	1 --传送带编号
	2--传送带编号
返回值	无返回值

注：传送带这一章节的指令不能单独使用，须组合才能完成传送带工艺；

此章节只是对这几个指令做简单的介绍说明，具体应用请参考动态跟随应用 AR 程序或咨询厂家。

3.1.23 调试指令

指令符号	指令说明
print	打印输出用户调试数据
Error	终止程序运行，并输出错误信息

print	
使用说明	从 RS232 串口输出调试数据
语法说明	print(...)

参数说明	可变参数，参数个数和类型可以任意
举例说明	1. print(12) --从串口输出数据 12 2. print (“Robot”) --从串口输出字符串数据 Robot 3. local a = 10 4. print ("Robot",a) --从串口输出字符串数据 Robot 和数字 10

Error	
使用说明	终止程序运行，并输出错误信息
语法说明	Error(A)
参数说明	用于输出用户定义的错误信息，字符串类型
举例说明	Error(“程序运行错误”) --执行后在界面输出报警信息并终止程序运行

3.1.24 共享点位表操作指令

指令符号	指令说明
Point	调用点位表中的点位
new	将要更新的点位写入到共享点位表里
teach	将当前点位示教到共享点位表里

Point		
使用说明	调用点位表中的点位	
语法说明	用法 1：B= Point (A) 用法 2：B = Point(A,“g”)	
参数说明	**用法 1：调用当前工程中点位表里的数据**	
	A	点位表中的点位数据序号
	用法 2：调用共享点位表里的数据	
	A	共享点位表中的点位数据序号（0~499）
返回值	B	将共享点位表里的点位赋值给 B

举例说明	1. local pos={} pos=Point(10) --调用当前工程中点位表里 p10 点赋值给 pos print(pos.x,pos.y,pos.z,pos.c) 2. local pos={} pos=Point(10,“g”) --调用共享点位数据表中 g10 点赋值给 pos print(pos.x,pos.y,pos.z,pos.c)

注 1：Point 指令使用场景：工程下有多个线程，主线程外的其它线程若要使用共享点位表里的点位须使用 Point 指令；

注 2：“g”是关键字不能修改。

new		
使用说明	将要更新的点位写入到共享点位表里	
语法说明	new(A,B,C,D,E,F,G,“g”)	
参数说明	A	共享点位表中的点位序号，范围 0~499
	B	新点位中的 x 坐标值
	C	新点位中的 y 坐标值
	D	新点位中的 z 坐标值
	E	新点位中的 c 坐标值
	F	新点位中的手系 h
	G	可选参数（可填、可不填）。新点位的名字（string 类型）
返回值	无	
举例说明	1. print(pos.x,pos.y,pos.z,pos.c) new(5,300,100,-10,30,0, “g”) --将点位（300,100,-10,30）写入到共享点位表中的 g5 点； 2. new(1,300,100,-10,30,0, “fangliao”, “g”) -- 将点位（300,100,-10,30）写入到共享点位表中的 g1 点, g1 点注释为 fangliao 点； 3. local pos = getcart() new(10,pos.x,pos.y.pos.z,pos.c,pos.h, “g”) --将获取的当前点	

	位写入到共享点位表中的 g10 点； MovP(g10) --运动到共享点位表中的 g10 点

注 1：A 的值只能为 0~499 中的数字，不能写成 g1~g499；

注 2：“g”是关键字不能修改。

teach		
使用说明	将当前点位示教到共享点位表里	
语法说明	teach (A, “g”)	
参数说明	A	共享点位表 DATA 中的点位数据，范围 0~499
举例说明	teach(10, “g”) --将当前点位写入到共享点位表中的 g10 点；	

注 1：A 的值只能为 0~499 中的数字，不能写成 g1~g499；

注 2：“g”是关键字不能修改。

3.1.25 系统指令

指令符号	指令说明
syswork	设置系统工作状态
sysstate	读取各个轴的电流值或获取系统状态
sysrate	设置全局速度速率
systime	获取系统时钟

syswork		
使用说明	设置系统工作状态	
语法说明	syswork(A)	
参数说明	无返回值	
	输入参数说明	
	A	1 或 RUN 启动程序运行
		2 或 PAUSE 暂停程序运行

		3 或 STOP　　停止程序运行
		4 或 RESET　　程序复位
举例说明	sywork(2)　　--程序暂停 --sywork(PAUSE)　　--程序暂停 Delay(100)　　--延时 100ms sywork(1)　　--启动程序 --sywork(RUN)　　--启动程序	

注：此指令（非阻塞）仅限于后台程序使用，若在非后台程序中使用会报错“后台线程特有指令”。

sysstate		
使用说明	读取各轴的电流值或获取系统状态	
语法说明	state = sysstate(n)	

n	state	
空	获取机器人机器人报警信息	0：无报警
		1：伺服报警
		2：DSP 报警
		4：操作报警
		8：系统报警
0	获取机器人的使能状态信息	0：断使能状态
		1：使能状态
1	获取第一轴的电流值，单位毫安	
2	获取第二轴的电流值，单位毫安	
3	获取第三轴的电流值，单位毫安	
4	获取第四轴的电流值，单位毫安	
5	获取机器人的运行状态	0：空闲
		1：暂停
		2：运行

举例说明	1. local state1 = sysstate(1) --获取第一轴的电流值 local state2 = sysstate(2) --获取第二轴的电流值 local state3 = sysstate(3) --获取第三轴的电流值 local state4 = sysatate(4) --获取第四轴的电流值 2. local state = sysstate(5) if state == 0 then print(“机器人空闲状态”) elseif state==1then print(“机器人处于暂停状态”) elseif state==2 then print(“机器人处于运行状态”) end

sysrate

使用说明	获取或设置全局速度倍率	
语法说明	Rate = sysrate(A)	
参数说明	A	Rate
	空	获取全局速度倍率
	1~100	设置全局速度倍率
举例说明	1. sysrate() --获取当前的全局速度倍率 2. sysrate(50) --设置全局速度倍率为 50%	

注：当设置全局速度倍率时，设置的倍率值必须是 1~100 之间的整数值。

systime

使用说明	获取系统时钟	
语法说明	time = systime()	
参数说明	无输入	
	返回值	**time**
	time	系统时钟值，单位：**毫秒**

举例说明	local time1=systime() --获取当前的系统时钟 MovP(p1) MovP(p2) local time2 = systime()-time1 --获取程序运行的时间 print(time2)

3.1.26 modbus 主站读写指令

指令符号	指令说明
ReadRegW	从从站指定的 16 位字地址中读取整型数据
ReadRegDW	从从站指定的 32 位双字地址中读取数据
WriteRegW	往从站指定的 16 位字地址中写入整型数据
WriteRegDW	往从站指定的 32 位双字地址中写入数据
MsetIport	设置从站的 IP 地址和端口号

ReadRegW		
使用说明	从从站指定的 16 位字地址中读取整型数据	
语法说明	Value = ReadRegW ({A,B},C,D)	
参数说明	A	数据通讯协议，A=1 代表 232 通讯（Modbus RTU）；A=2 代表 485 通讯（Modbus RTU）；A=3 代表网络通讯（Modbus TCP/IP）
	B	从站站号
	C	寄存器地址
	D	读取变量的个数（可选），默认为 1
返回值	Value	返回读取寄存器对应地址中变量的值
举例说明	1. local plc={1,1} --Modbus 232 通讯，从站站号为 1 ReadRegW(plc,250) --读从站 250 这个 16 位整型地址的值 local a=ReadRegW(plc,250,20)--从从站地址 250 开始连续读 --20 个 16 位整型变量	

```
for i=1,20 do
 print(a[i])
      Delay(10)
  end
```

2. local plc = {3,1} --Modbus TCP/IP 通讯，从站站号为 1

```
SlaveIp = “192.168.0.100”      --从站 IP 地址
SlavePort = 502
MsetIport(SlaveIp,SlavePort)   --设置从站参数
local a = ReadRegW(plc,2,6) --从从站地址 2 开始连续读 6 个
                            --16 位地址的数据赋值给数组变量 a 中
 for i=1,6 do
      print(a[i])
      Delay(10)
 end
 local b = ReadRegW(plc,4)    --读从站地址 4 的数据赋值给
变量 b
print(b)
```

ReadRegDW		
使用说明	从从站指定的 32 位双字地址中读取数据	
语法说明 1	Value=ReadRegDW ({A,B},C,D,E)	
参数说明	A	数据通讯协议，A=1 代表 232 通讯（Modbus RTU）；A=2 代表 485 通讯（Modbus RTU）；A=3 代表网络通讯（Modbus TCP/IP）
	B	从站站号
	C	寄存器地址
	D	读取变量的个数（可选），默认为 1
	E	读取从站地址中数据的类型（可选），整型(int)和浮

		点型(float)，默认为整型
返回值	Value	返回读取寄存器对应地址中变量的值

语法说明 2	ReadRegDW ({A,B},C,D,E,F)	
参数说明	A	数据通讯协议，A=1 代表 232 通讯（Modbus RTU）；A=2 代表 485 通讯（Modbus RTU）；A=3 代表网络通讯（Modbus TCP/IP）
	B	从站站号
	C	从站被访问的首地址
	D	机器人 RAM 表中的地址
	E	读取变量的个数（可选），默认为 1
	F	读取从站地址中数据的类型（可选），整型(int)和浮点型(float)，默认为整型

语法说明 3	ReadRegDW ({A,B},C,D,E, “ F”,G)	
参数说明	A	数据通讯协议，A=1 代表 232 通讯（Modbus RTU）；A=2 代表 485 通讯（Modbus RTU）；A=3 代表网络通讯（Modbus TCP/IP）
	B	从站站号
	C	从站被访问的首地址
	D	机器人共享点位表的点序号
	E	共享点位的个数
	F	写入到共享点位表中点位的手系，“lg”(左手系), “rg”(右手系)
	G	读取从站地址中数据的类型（可选），整型(int)和浮点型(float)，默认为整型

举例说明

1. local plc={1,1}　　--Modbus 232 通讯，从站站号为 1
   print(ReadRegDW(plc,200))　--读从站 200 这个 32 位整型地址的值
   print(ReadRegDW(plc,210,"float"))--读从站 210 这个 32 位浮点型地址的值
   local a=ReadRegDW(plc,250,20)　--从从站地址 250 开始连续
   --读 20 个 32 位整型变量
   for i=1,20 do
   　　print(a[i])
   end
   local b=ReadRegDW(plc,350,20,"float")　--从从站地址 350
   --开始连续读 20 个 32 位浮点型变量
   for i=1,20 do
   　　print(b[i])
   end
2. local plc = {3,1}　　--Modbus TCP/IP 通讯，从站站号为 1
   SlaveIp = “192.168.0.100”　　--从站 IP 地址
   SlavePort =502
   MsetIport(SlaveIp,SlavePort)　--设置从站参数
   ReadRegDW(plc,250,0x100,10) --从从站地址 250 开始连续读
   --10 个 32 位整型地址的值分别写入到机器人 RAM 数据表的
   --0x100、0x102……0x112 这 10 个地址中
   ReadRegDW(plc,250,0x10,10,"float") --从从站地址 250 开始连
   --续读 10 个 32 位浮点型地址的值分别写入到机器人 RAM 数
   --据表的 0x010、0x012 ……0x022 这 10 个地址中
3. local plc={2,1}　--Modbus 485 通讯，从站站号为 1
   --ReadRegDW(plc,80,23,1,"lg") --从从站地址 80 开始连续读地
   --址（4 个 32 位整型地址）的值写入到机器人共享点位表的
   --g23 点位中，手系为左手系

	ReadRegDW(plc,100,24,3,"rg","float")--从从站地址 100 开始连续读地址（12 个 32 位浮点型地址）的值写入到机器人共享点位表的 g24、g25、g26 这三个点位中，手系为右手系

WriteRegW		
使用说明	往从站指定的 16 位字地址中写入整型数据	
语法说明	Value =WriteRegW ({A,B},C,D)	
参数说明	A	数据通讯协议，A=1 代表 232 通讯（Modbus RTU）；A=2 代表 485 通讯（Modbus RTU）；A=3 代表网络通讯（Modbus TCP/IP）
	B	从站站号
	C	从站被写入的首地址
	D	写入从站地址中的数据（可以是单个整型数据，也可以是整型数组）
返回值	Value	返回写入到寄存器对应地址中变量的值
举例说明	1. local plc={1,1} --PLC 通信参数使用 1 为 UART,站号为 1 WriteRegW(plc,250,1) --写 PLC 寄存器 250 16 位字地址 WriteRegW(plc,250,{10,20,30}) --连续写 PLC 寄存器 250 16 位字地址	

WriteRegDW		
使用说明	往从站指定的 32 位双字地址中写入数据	
语法说明 1	WriteRegDW ({A,B},C,D,E)	
参数说明	A	数据通讯协议，A=1 代表 232 通讯（Modbus RTU）；A=2 代表 485 通讯（Modbus RTU）；A=3 代表网络通讯（Modbus TCP/IP）
	B	从站站号

	C	从站被写入的首地址
	D	写入到寄存器的数据（可以是单个数据，也可以是数组）
	E	写入从站地址中数据的类型（可选），整型(int)和浮点型(float)，默认为整型
语法说明 2	WriteRegDW ({A,B},C,D,E,F)	
参数说明	A	数据通讯协议，A=1 代表 232 通讯（Modbus RTU）；A=2 代表 485 通讯（Modbus RTU）；A=3 代表网络通讯（Modbus TCP/IP）
	B	从站站号
	C	从站被写入的首地址
	D	机器人 RAM 数据表中的地址
	E	连续访问地址的个数（可选），默认为 1
	F	写入从站地址中数据的类型（可选），整型(int)和浮点型(float)，默认为整型
举例说明	1. local plc={1,1} -- Modbus 232 通讯，从站站号为 1 WriteRegDW(plc,100,12) --将数据 12 写入到从站地址 100 中 WriteRegDW(plc,110,10, “int”) --将数据 10 写入到从站地址 110 中 WriteRegDW(plc,120,12.34,”float”) --将浮点型数据 12.34 写入到从站地址 120 中 WriteRegDW(plc,60,{246,-974,84,-24817,5146}) --将数组中的 5 个整型数据写入到从站地址以 60 开始的 5 个 --32 位整型地址中 WriteRegDW(plc,80,{24.6,-97.4,8.4,-248.17,51.46},“float”) --将数组中的 5 个浮点型数据写入到从站地址以 80 开始的 5-- 个 32 位浮点型地址中 2. local plc = {3,1} --Modbus TCP/IP 通讯，从站站号为 1 SlaveIp = “192.168.0.100” --从站 IP 地址	

SlavePort = 502

MsetIport(SlaveIp,SlavePort)　--设置从站参数

WriteRegDW(plc,250,0x100,10) --将机器人 RAM 地址表中从 --0x100 开始连续 10 个地址中的整型数据获取后分别写入到 --从站地址以 250 开始的 10 个的 32 位整型地址中

WriteRegDW(plc,20,0x10,5,"float") --将机器人 RAM 地址表中从 0x10 开始连续 5 个地址中的浮点型数据获取后分别写入到从站地址以 20 开始的 5 个的 32 位整型地址中

MsetIport		
使用说明	设置从站的 IP 地址和端口号	
语法说明	MsetIport(A,B)	
参数说明	A	从站的 IP 地址（字符串类型）
	B	从站的端口号（number 类型）
举例说明	MsetIport(“192.168.0.100”,8080)　--从站的 IP 地址为 192.168.0.100，端口号为 8080	

注意 1：此指令仅在 Modbus TCP/IP 网络通讯，机器人作为主站，外部设备（例如，PLC）作为从站时使用；机器人使用主站读写指令前，一定要用此指令来设置从站的 IP 地址和端口号，否则无法读写从站地址；

注意 2：Modbus 232/485 不使用此指令，须在参数中将通信方式设置为“Modbus 主站”才能使用 Modbus 232/485 主站功能。

3.1.27 文件操作指令

指令符号	指令说明
io.open	打开文件
io.type	检测一个文件句柄是否可用
file:write	写数据到文件中
file:read	按指定的格式读取一个文件
file:seek	文件定位，设置和获取当前文件位置
file:lines	逐行获取文件中的内容
file:close	关闭文件

io.open		
使用说明	打开文件	
语法说明	file =io.open(path,mode)	
参数说明	path	**文件路径** 相对路径---不写盘符，文件位置在当前工程目录下，例如当前工程目录为 d:\\projects\\scara,则文件位置为 d:\\projects\\scara\\test.txt 绝对路径---指定文件的完整路径，例如 d:\\test.txt
		打开方式

	mode	"r" ---读模式 "w" ---写模式，原数据会被覆盖 “a” ---附加模式,所有之前的数据将被保存，只允许在文件尾进行添加 "rb" 或“wb” ---以二进制方式读写文件
返回值	file	打开成功，则返回文件句柄（指针地址）,大于 0 的一个值； 打开失败，则返回 nil，以及对应的错误信息和错误代码

io.type		
使用说明	检测一个文件句柄是否可用	
语法说明	flag = io.type(obj)	
参数说明	obj	被检测的对象
返回值	flag	标志位 file ---一个打开的文件句柄
		closed file ---一个已关闭的文件句柄
		nil ---被检测的 obj 不是一个文件句柄

file:write		
使用说明	写文件，写数据到文件中	
语法说明	size = file:write(data)	
参数说明	file	通过 io.open 函数获取的文件句柄
	data	写入到文件中的数据
返回值	size	写入数据的大小,大于 0 成功,0 为失败

file:read

使用说明	按指定的格式读取一个文件	
语法说明	data = file:read (way)	
参数说明	file	通过 io.open 函数获取的文件句柄
	way	读取文件的格式 “*n” ---读取纯数字的文件 “*a” ---从当前位置读取整个文件，若为文件尾，则返回空的字符串； “*l” ---读取下一行的内容，若为文件尾，则返回 nil; number:大于 0 的整数；
返回值	data	读取的数据 读取成功，则按设定的读取格式返回的一个字符串或数字； 读取失败，则返回 nil； 若没有指定格式将默认按行方式进行读取

file:seek		
使用说明	文件定位，设置和获取当前文件位置	
语法说明	flag = file:seek(whence,offset)	
参数说明	file	通过 io.open 函数获取的文件句柄
	whence	"set"：定位到文件头开始 "cur"：定位到文件当前位置开始（默认） "end"：定位到文件尾开始
	offset	偏移位置，默认为 0
返回值	flag	标志位 定位成功，返回最终的文件位置（按字节）； 定位失败，则返回 nil 加错误信息
举例说明	1. file:seek() --不带参数，则定位到当前位置 2. file:seek(“set”) --定位到文件头 3. file:seek(“end”) --定位到文件尾，返回文件的大小	

4. file:seek(“set”,2) --定位到文件头开始的两个字节
5. file:seek(“cur”,3) --定位到文件当前位置开始的两个字节

注 1：whence 的默认值为“cur”, offset 的默认值为 0；

file:lines		
使用说明	逐行获取文件中的内容	
语法说明	data= file:lines()	
参数说明	file	通过 io.open 函数获取的文件句柄
返回值	data	通过一个迭代函数，每次调用获取文件中的一行内容
举例说明	file = io.open(“test.txt”) for line in file:lines() do print(line) end	

file:close		
使用说明	关闭文件	
语法说明	flag = file:close()	
参数说明	file	通过 io.open 函数获取的文件句柄
返回值	flag	标志位 关闭成功，返回 true
举例说明	functionmain() **----------情形 1:文件只写：被写入文件中的原数据被清除、读文件失败** local file1 = io.open(“../11.txt”, “w”) --写方式打开文件名为相对路径在当前工程目录下 file1:write("write test111!") --往文件里写数据 file1:close() --关闭文件 **----------情形 2:文件只读：写文件会失败**	

local file2=io.open("../22.txt","r") –读方式打开，文件名为相对路径在当前工程目录下

local data=file2:read(“*all”)　　--读取文件中的所有数据

file2:close()　　--关闭文件

----------情形 3:追加方式读写文件—被写入文件中的原数据保留

local file3=io.open("../4.txt","a")　　--追加方式写文件

file3:write("\r\nwrite test!")　　--再次写文件

file3:close()　　--关闭文件

end

注意 1：当文件句柄被垃圾收集后，文件将自动关闭。句柄将变为一个不可预知的值;

注意 2：每次读写文件前要用 io.open 打开文件，读写文件完毕后用 file:close 关闭文件;

注意 3：文件读写不能同时进行，即只能以读或者写的方式打开一个文件，文件操作完后必须关闭文件，再次打开文件，才能切换另外一种模式打开文件。

3.1.28 队列操作指令

指令符号	指令说明
qexist	判断队列是否存在
qcreate	创建队列
qpush	往队列中写入数据
qpop	删除队列中的首元素
qfront	取队列中的首元素
qpopfront	取队列中的首元素然后从队列中删除
qempty	判断队列是否为空
qsize	队列大小
qdestroy	删除队列

qexist		
使用说明	判断队列是否存在	
语法说明	Flag = qexist(ID)	
参数说明	ID	队列的名字，类型：整形数字
返回值	Flag	Flag 为 true，队列存在；Flag 为 false，队列不存在

qcreate		
使用说明	创建队列	
语法说明	qcreate(ID)	
参数说明	ID	队列的名字，类型：整形数字
返回值	无	

qpush		
使用说明	往队列里写入数据	
语法说明	qpush(ID,data)	
参数说明	ID	队列的名字，类型：整形数字
	data	写入到队列中的数据，数据类型可以是数字、字符串、数组等，或者组合
返回值	无	

qpop		
使用说明	删除队列中的首元素	
语法说明	qpop(ID)	
参数说明	ID	队列的名字，类型：整形数字
返回值	无	

qfront		
使用说明	取队列中的首元素	
语法说明	Data=qfront(ID)	
参数说明	ID	队列的名字，类型：整形数字
返回值	Data	队列中第一个元素的值

qpopfront		
使用说明	取队列中的首元素然后从队列中删除	
语法说明	Data=qpopfront(ID)	
参数说明	ID	队列的名字，类型：整形数字
返回值	Data	队列中第一个元素的值

qempty		
使用说明	判断队列是否为空	
语法说明	Flag =qempty(ID)	
参数说明	ID	队列的名字，类型：整形数字
返回值	Flag	Flag 为 1，队列为空；Flag 不为 1，队列中有数据

qsize		
使用说明	队列的大小	
语法说明	Len=qsize(ID)	
参数说明	ID	队列的名字，类型：整型数字
返回值	Len	队列的长度

qdestroy	
使用说明	删除队列

语法说明	qdestroy(ID)	
参数说明	ID	队列的名字，类型：整型数字
返回值	无	
举例	flag1 = qexist(1) --判断 1 号队列是否存在 if flag1 == truethen print("OK") elseif flag1 ==false then --若不存在创建 1 号队列 qcreate(1) end qpush(1,{x=300,y=100,z=0,c=30,h=0,no=0,pulse=0,coder=0,type=0,cnvno=0,nameno=0}) --将数组存入到 1 号队列 len = qsize(1) --求取队列的长度（元素的个数） print(len) --3 while qempty(1)~=1 do --判断 1 号队列是否为空，非 1 表示不为空 data = qfront(1) --取 1 号队列中的首元素 print(data) --打印 qpop(1) --删除 1 号队列中的首元素 end qdestroy(1) --删除 1 号队列	

注：队列操作指令暂时仅用于传送带跟随应用。

3.1.29 其它指令

指令符号	指令说明
dofile	同一个工程中不同 AR 程序的加载

dofile	
使用说明	同一个工程中不同 AR 程序的加载
语法说明	dofile(“../ProjectName/ARFileName”)

参数说明	..	固定格式
	ProjectName	工程文件的名字
	ARFileName	AR 程序文件的名字
返回值	无	

举例	--一个线程中有多个 AR 程序的时候，可以在优先加载的那个 AR-程序中用 dofile 指令调用其它的 AR 程序的子函数（A=1，A=2）；
	--或一个线程中有多个 AR 程序的时候，可以在优先加载的那个 AR 程序中用 dofile 指令调用其它的 AR 程序（A=3，A=4）；
	--可参考 ARStudio 上位机软件中的案例程序 DoFile 工程
	function main()
	while **true** do
	A = publicread(0x100)
	if **A** == 1 then
	dofile("../DoFile/SubFunction1.AR")
	--加载 Subfunction1.AR 文件
	process1()
	--调用 SubFunction1.AR 中的 process1()函数
	elseif **A** == 2 then
	dofile("../DoFile/SubFunction2.AR")
	--加载 Subfunction2.AR 文件
	process2()
	--调用 SubFunction2.AR 中的 process2()函数
	elseif **A** == 3 then
	dofile("../DoFile/SubFunction3.AR")
	--加载 Subfunction3.AR 文件
	elseif **A** == 4 then
	dofile("../DoFile/SubFunction4.AR")
	--加载 Subfunction4.AR 文件	

```
                end
                Delay(10)
            end
        end
```

3.2 修改记录

3.2.1 QC410 与 QC400 部分指令对比

指令类别	QC400	**QC410**	对比说明
运动参数设置指令	无	**JerkJ**	
		JerkL	
		AccC	
		SpdC	
		JerkC	
矩形码垛指令	SetPlt	**Set_Plt**	QC410 将 QC400 的三点码垛和四点码垛指令合并成一个指令，通过在参数中区分
	SET_PLT		
	GetPlt	**Get_Plt**	
	GET_PLT		
圆形码垛指令	SetArcPlt	**Set_ArcPlt**	用法基本保持不变，只是细分了整圆与圆弧码垛
	GetArcPlt	**Get_ArcPlt**	
配置码垛指令	GetPltPos	**GetPltPos**	用法基本保持不变
	ResetPlt	**ResetPlt**	
	无	**InitPlts**	QC410 新增
通信指令	sysnetclr	无	QC410 去掉了在 AR 程序中输入对方 IP 和 Port 的函数。如要用此功能需要，须在当前工程下添加网络配置对应的 IP 地址和 Port，然后调用 SendNet 和 ReadNet 函数即可实现
	CloseNet		
	OpenNet		
	ConnectNet		
	sysnetget		
	sysnetcatch		
	RecvNet		
	sysnetsend		
	writeNet		
视觉指令	InitTCPnet	**InitNet**	QC410 修改
	CCDsent	**SendNet**	用法基本一样，但是名字细分为网络名称、视觉名称或传送带名称 1. 网络名称默认为
	CCDtrigger	**CCDtrigger**	
	CCDrecv	**CCDrecv**	
	CCDGet	无	

	CCDoffset	无	COM0、COM1… 2. 视觉名称默认为 CAM0、CAM1… 3. 传送带名称默认为 CON0、CON1…
	无	**ReadNet**	
	GetDynCCDPos	无	QC410 此函数合并到 CCDrecv
传送带指令	QC410 的传送带指令与 QC400 几乎全部改变（SysOver 指令除外）		
坐标系指令	getcart	**Getpos**	获取当前的笛卡尔坐标和关节坐标，功能完全一样，只是 QC410 与 QC400 所使用的名字不一样
	targetok/getjoint	**Targetok**	QC410 保留了 QC400 的 targetok 的功能，也继承了 getjoint 的功能，同时也新增了 short 最短路径功能。
系统指令	syswork	**syswork**	QC410 保留了 QC400 的功能。 对于 QC400，输入参数为 1,2,3,4 分别代表运行、暂停、停止、复位； 对于 QC400，输入参数为 RUN、PAUSE、STOP、RESET 分别代表运行、暂停、停止、复位

文件操作指令	fopen fread fwrite fseek fgets fsize feof fclose	**io.open** **io.type** **file:read** **file:write** **file:seek** **file:lines** **file:close**	关于文件操作的指令，QC410 与 QC400 完全不一样

3.2.2 QC410 指令更新记录

修改时间	修改内容
2019.9.26	新增 SetI 指令用于最佳加速度功能
2019.12.23	新增 Modbus 主站读写指令
2020.3.10	新增 global.*使用例程
2020.3.19	1. CCDrecv 新增第四个返回参数； 2. 将之前的视觉指令修改为网络通讯指令； 3. 串口指令与 RAM 读写指令分开；
2020.4.30	1. 新增 dofile 指令

第四章伺服管理调试篇

进行 QC410 驱控一体机操作之前，必须先对伺服部分进行试运行调试。如果存在已调试好的伺服参数文件，可将伺服参数下载到驱动器，直接试运行；如果没有伺服参数，需严格按照流程对伺服参数进行调试。

伺服调试应严格遵循以下顺序：

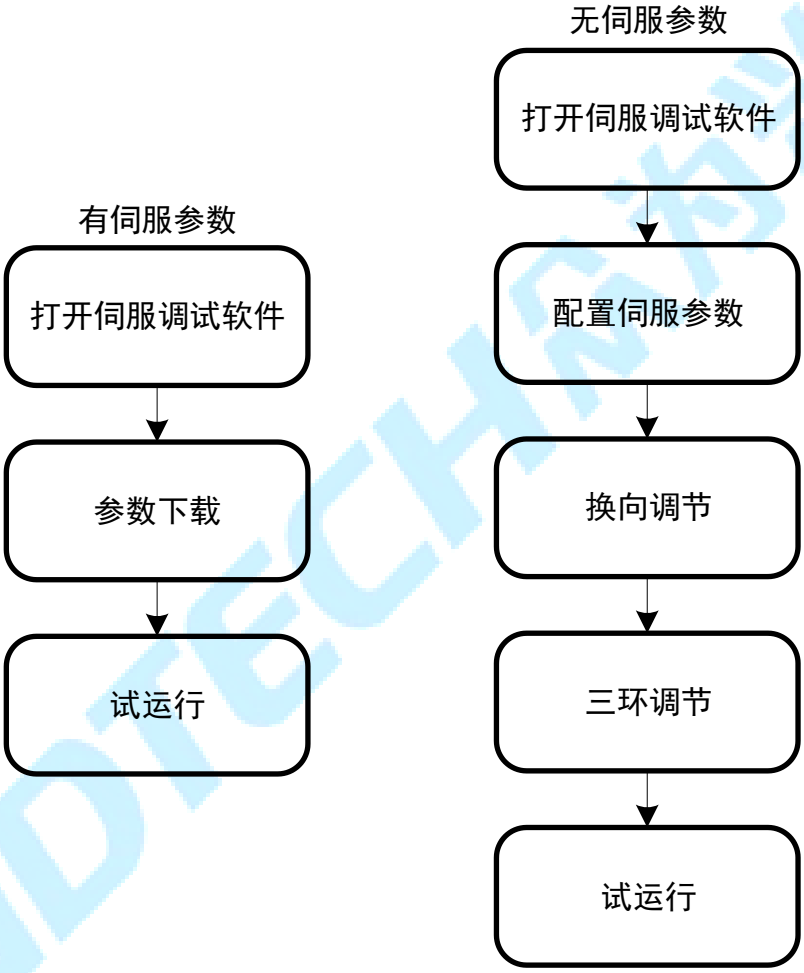


图 1 伺服调试基本顺序

4.1 伺服调试软件启动

4.1.1 伺服调试前检查

为确保安全、有效地进行调试，调试前应对以下几项进行检查与确认，若发现有不良状态，请与本公司联系。

（1）伺服电机的状态

- 检查电机固定部件，确保各线路连接紧固；
- 检查电机编码器线及电源连接线，确保接线正确且连接紧固。

（2）驱控一体机的状态

- 检查 QC410 驱控一体机上的外接电源，确保供电正常；
- 确保 QC410 驱控一体机及伺服电机的编码器线缆与动力线缆接线无误。

（3）检查 ARStudio 上位机软件状态

- 检查 ARStudio 的软件版本状态，确保版本号无误
- 确保 ARStudio 与 QC410 驱控一体机正确连接无误

软件安装环境：

购买驱控一体机后，我们会为客户提供伺服调试软件包。安装前需确认软件安装环境：

- Windows 7、Windows10　32 位；
- Windows 7、Windows10　64 位；
- Windows　XP。

显示器分辨率：1280 x 768 以上

4.1.2 调试软件启动步骤

为正常使用调试软件，务必遵循以下步骤完成调试软件的启动。

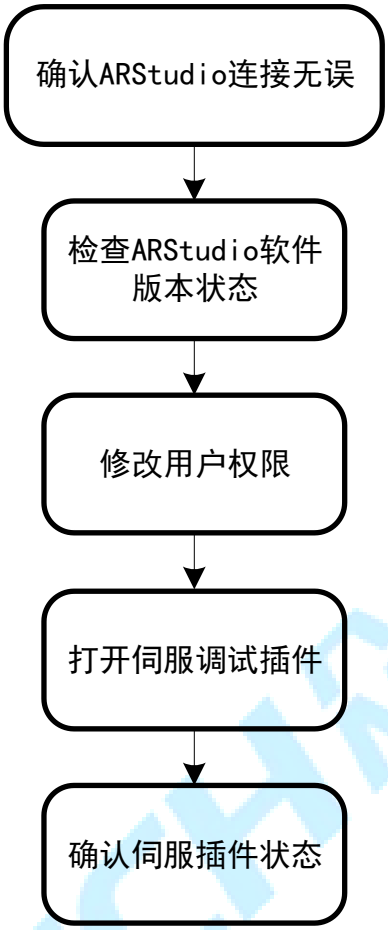


图 1-1 调试插件启动流程图

【确认 ARStudio 连接无误】

ARStudio 连接与 QC410 驱控一体机正确连接时，软件右上角图标显示为 ；ARStudio 未与 QC410 驱控一体机连接时，软件右上角图标显示为 ；ARStudio 与 QC410 驱控一体机正在连接时，软件右上角图标显示为 ；

（详情请查阅 ARStudio 软件的说明文档）

【检查 ARStudio 软件版本状态】

当 ARStudio 软件版本正确时，在系统-工具-更新的插件中，连接远程主机后，版本信息栏将显示绿色的 ok 字样；否则，会出现红色的 err 字样。如图 1-2 所示：

（详情请查阅 ARStudio 软件的说明文档）

(a) 版本状态正常　　(b)版本状态错误

图 1-2 ARStudio 软件版本状态

【修改用户权限】

修改当前的用户权限为**厂家**。点击 ARStudio 软件右上角的用户权限管理控制按钮，然后点选的**厂家**按钮，输入相应的登录密码并点击登录，完成用户权限切换。操作流程如图 1-3 所示：

（详情请查阅 ARStudio 操作说明书）

图 1-3 ARStudio 用户权限修改流程

【打开伺服调试软件】

（1） 通过 ARStudio 右侧的菜单栏系统-辅助工具-插件信息，确认 ServoTunning 可用的复选框已经勾上；若无勾选，则将其勾选，然后重新启动 ARStudio 上位机软件。

系统升级 | 系统状态 | 伺服管理 | 插件信息

名称	可用	详细信息	作者	版本	文件名
GlData Editor	☑	RAM 数据	gex	V1.00	cfgram.dll
RobotProject	☑	工程管理	gex	V1.00	robotproject.dll
VisionNode	☑	视觉模块	luoqi	V1.00	visionnode.dll
ConveyorNode	☑	传送带模块	luoqi	V1.00	conveyornode.dll
CpuNode	☑	线程模块	gex	V1.00	cpunode.dll
DataNode	☑	点位表模块	gex	V1.00	datanode.dll
GdbDebugger	☑	GDB调试器	stl	V1.00	gdbdebugger.dll
SysCfgParam	☑	参数设置	gex	V1.00	syscfgparam.dll
MovePos	☑	手动面板	gex	V1.00	movepos.dll
NetworkNode	☑	网络模块	luoqi	V1.00	networknode.dll
PalletNode	☑	码垛模块	gex	V1.00	palletnode.dll
ReducerTest	☑	减速机测试	luoqi	V1.00	reducertest.dll
SafeSpace	☑	安全空间	gex	V1.00	safespace.dll
ServoState	☑	伺服状态	stl,gex	V1.00	servostate.dll
ServoTunning	☑	伺服管理	timyear	V1.00	servoTunning.dll
SpeedRate	☑	倍率设置	gex	V1.00	speedrate.dll
Track3d	☑	轨迹跟踪	gex	V1.00	track3d.dll
Standard User...	☑	标准界面	gex	V1.00	standarduserui.dll
SysCfgRobot	☑	安装设置	gex	V1.00	syscfgrobot.dll
System Mana...	☑	系统管理	gex	V1.00	sysmanager.dll
SystemState	☑	系统状态	gex	V1.00	systemstate.dll
Input T9	☑	T9键盘	gex	V1.00	t9keyboard.dll
Net Config	☑	网络工具	gex	V1.00	tcpipconfig.dll

（2） 【伺服管理】插件只能在 ARStudio 上位机软件连接 QC410 控制器时才能使用，在虚拟机器人模式下，操作“伺服管理”插件没有意义。

（3） 通过 ARStudio 右侧的菜单栏系统-辅助工具-伺服管理打开启动伺服调试软件（详情请查阅 ARStudio 操作说明书）

（4） 若 ARStudio.exe 同目录下，伺服指令表 ControllerMessagesList.xml 文件[或运动规划指令表 MotionControllerMessagesList.xml 文件]不存在，伺服调试软件将会报错，如图 1-4；在错误确认后，将弹出**建立内建指令列表属性窗口**（如图 1-5），可选择三种处理方式：

建立新的指令表文件：本插件将从伺服端获取底层的伺服指令表信息；然后在 ARStudio.exe 同目录下建立新的伺服指令表 ControllerMessagesList.xml[或 MotionControllerMessagesList.xml]文件；最后使用该文件进行伺服指令属性的构建。

使用内置指令表文件：本插件将使用软件内部内置的 ControllerMessagesList.xml[或 MotionControllerMessagesList.xml]文件构建伺服指令属性。

【ControllerMessagesList.xml】：伺服指令表文件

【MotionControllerMessagesList.xml】：运动规划指令表文件

取消并退出：本插件将停止构建过程，取消打开插件。

图 1-4 指令表 ControllerMessagesList.xml 文件不存在错误弹框

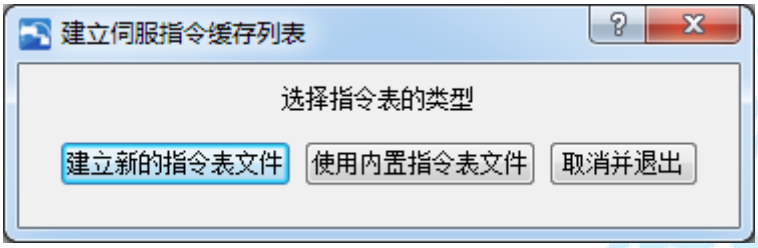


图 1-5 建立内建指令属性列表弹框

【确认伺服插件状态】

当伺服插件正确启动时，伺服插件初始界面如图 1-6 所示：

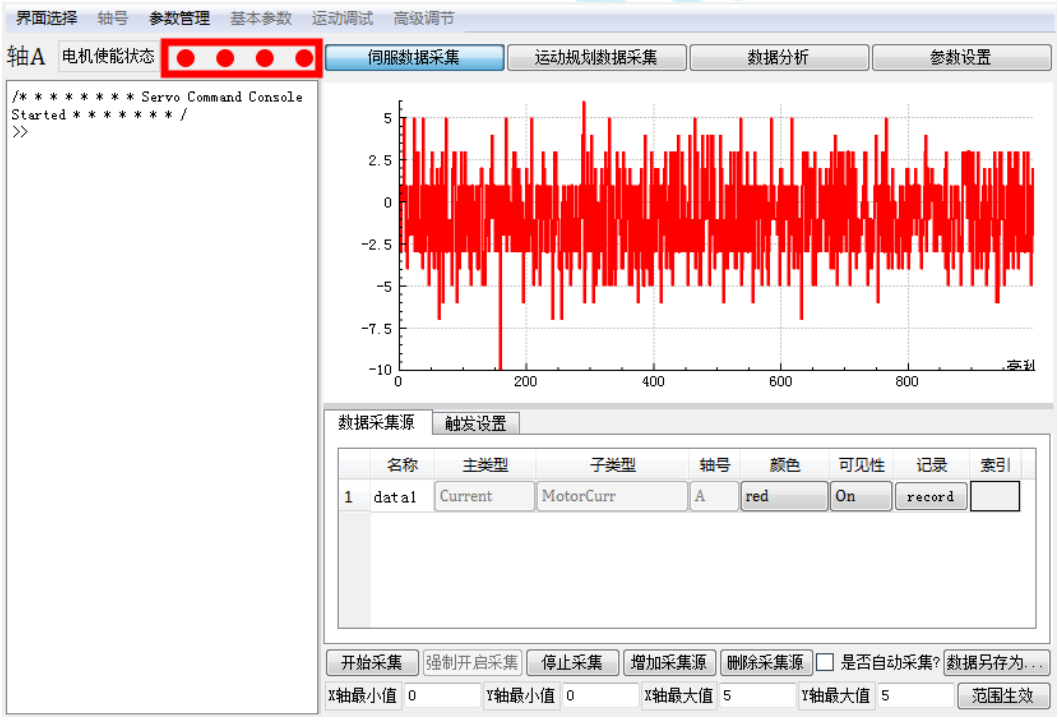


图 1-6 伺服调试插件开始界面

当“伺服管理”插件**未与 QC410 制器连接**或**与 QC410 控制器连接断开**时，在图 1-6 界面左上方黑色框的圆点会变成暗黄色；显示红色，则代表对应的电机轴使能断开；显示绿色，则表示对应的电机轴已经使能；

4.1.3 调试软件介绍

QC410 驱控一体伺服调试插件运行主界面如下图所示：

图 1-7 伺服调试软件主界面

简要说明：

① 机器人电机轴状态：

当“伺服管理”插件**未与 QC410 控制器连接**或**与 QC410 控制器连接断开**时，在图 1-6 界面左上方黑色框的圆点会变成暗黄色；显示红色，则代表对应的电机轴使能断开；显示绿色，则表示对应的电机轴已经使能；

使用鼠标左键双击圆点，即可控制机器人电机轴上使能/去使能（注意：该功能尽量不与 ARStudio 的使能开关一同使用）。

② 伺服管理界面主菜单栏

菜单栏主要控制伺服调试软件的窗口显示以及切换，轴号切换，以及基本的文件处理控制。

其主要包括：界面选择，轴号，参数管理，基本参数，运动调试，高级调节；

界面选择：

用于控制基本界面的显示与否，包括：参数设置，终端，状态、示波器以及示波器数据项显示表；界面的显示控制，只要勾选相应的选项，界面就会显示相应的窗口，如

图 1-8 所示：

图 1-8 界面选择

图 1-8 显示为所有的窗口均显示于界面上。其中当“参数设置”界面设置为不可见时：“基本参数”，“运动调试”以及“高级调节”菜单按钮为禁用状态；

当“参数设置”界面以及“状态”界面设置为不可见时：“轴号”，“基本参数”，“运动调试”以及“高级调节”菜单按钮为禁用状态；

轴号：

用于设置及切换伺服轴号，以切换显示不同轴号的电机的伺服参数及伺服状态；

一般在设置参数之前，需先进行轴选择。选择哪一轴，对应设置或者调试该轴的参数。轴号选择如图 1-9 所示：

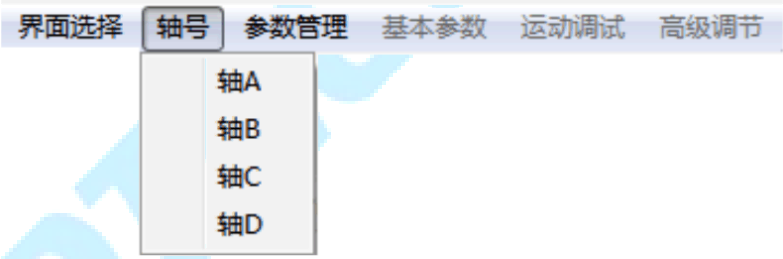


图 1-9 轴号选择

选择相应轴后，“伺服调试提示标签”将会更新显示当前调试轴的轴号。

参数管理：

其负责伺服参数信息的读取以及写入功能。其选项有:“从 Flash 中读取伺服参数”，“保存现伺服参数到 Flash 中”，“写入伺服参数文件”和“保存伺服参数”。

基本参数：

其负责电机的基本参数设置。其选项有：“电机基本参数”，“电机保护参数”以

及“换向调节”。

运动调试：

其主要负责电机运动性能方面的调试设置。其选项有：“点到点”和“PID 调试”。

高级调节：

其主要负责一些高级的运动性能方面的调试设置。其选项有：“输入整形”。

③ 伺服调试提示标签

该提示栏主要显示当前调试的机器人电机轴轴号，参数设置窗口界面的基本说明。

④ 参数设置窗口

该窗口用于显示及设置 QC410 伺服控制器的伺服参数，包括：电机参数，电机保护参数，点到点运动调试，PID 调试，输入整形以及换向调节；

⑤ 伺服状态信息窗口

该窗口用于显示 QC410 伺服的状态信息，主要分为普通信息和报警信息。

普通信息主要以文本形式的参数为主，显示当前电机轴的位置，速度，使能等的状态；而报警信息则以指示灯为主，显示当前电机的警报发生的状况。

⑥ 伺服终端窗口

该窗口为伺服或运动规划的终端窗口，可在该窗口中直接输入伺服指令，控制和查询伺服或运动规划的信息。其中使用运动规划指令时，为了与伺服指令进行区分，需要在指令前加入 m/M 字母作为指令的开头。

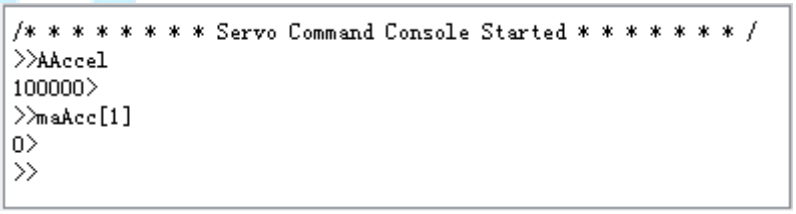


图 1-10 终端界面

⑦ 伺服示波器窗口

该窗口主要用于伺服调试过程中，伺服采集数据的显示以及数据的后续分析。

4.1.4 重要提示

（1）伺服调试插件的每一个窗口界面都有一个最小分辨率大小，当分辨率等于最小分辨率大小是，伺服插件的窗口将无法进行进一步的压缩。可能会导致某些窗口无法拖拽，拉伸。

解决方法 1：

在插件切换选项卡上，双击“伺服管理”选项卡，使伺服调试插件界面以对话窗口形式显示。然后即可对其它窗口进行拉伸，拖拽而不受“伺服管理”界面大小的影响。

解决方法 2：

在插件切换选项卡上，关闭“伺服管理”插件界面，直接释放窗限制。

解决方法 3：

在界面选择中，调整需要显示的界面窗口，减少界面占用。

4.2 参数管理

参数管理的选项如图 2-1：

图 2-1 参数管理操作选项

4.2.1 从 Flash 中读取伺服参数

点击该操作选项后，伺服控制器将会把 Flash 中的伺服参数写入到 RAM 中，然后将对各个电机轴进行计算滤波器操作。

4.2.2 保存现伺服参数到 Flash 中

点击该操作选项后，伺服控制器将会把当前 RAM 中的数据保存并写入到 Flash 中。需要注意，伺服调试界面上显示的伺服参数并不一定为控制器 RAM 中的伺服参数。只有执行完“刷新参数”或“提交参数”（在滤波器中为“计算滤波器”）之后显示的参数，才为 RAM 中的参数；

4.2.3 写入伺服参数文件

点击该操作选项后，界面将弹出确认窗口，点选“确定”后，选择需要载入的伺服参数文件（.par、.txt 格式）即可；

完成以上操作后，伺服控制器将提取文件中的伺服参数，并执行以下操作：

a. 读取参数文件并将参数写入控制器的 RAM 中；
b. 将 RAM 中的参数保存到 Flash 中；
c. 对各个电机轴进行计算滤波器操作；
d. 重启 QC410 以及 ARStudio；

4.2.4 保存伺服参数

点击该操作选项后，设置文件保存路径。在设定好保存路径之后，伺服控制器将获取当前 RAM 中的伺服参数，并将这些参数以文本的形式保存到用户设定的路径中。

4.3 基本参数设置

基本参数设置主要包括：电机基本参数设置、保护参数设置以及换向调节设置。其

选项如图 3-1。

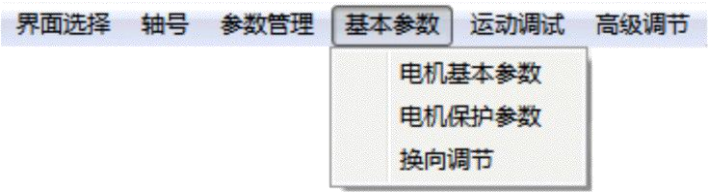


图 3-1 基本参数选项

4.3.1 电机基本参数

电机基本参数主要包括：电机类型，极对数，编码器类型，分辨率等信息。

图 3-2 电机基本参数设置

旋转电机类型对应极对数如下表：

表 3-1 常用电机类型及极对数

电机类型	极对数
三洋 17 位绝对式编码器电机	5
松下电机	4
多摩川电机	4
众为兴电机	4

根据对应电机及编码器信息，选择主编码器类型。

5—尼康绝对式 17 位（山洋）；

6—松下增量式 20 位；

7—松下绝对式 17 位；

10—多摩川省线增量式；

12—多摩川增量式 17 位；

13—多摩川绝对式 17 位；

分辨率：电机旋转一圈，对应编码器输出的脉冲数。分辨率计算公式为：2^N，N 为编码器位数。

表 3-2 常用电机编码器位数及分辨率

编码器位数	分辨率
2500 线	10000
17 位	131072
20 位	1048576

4.3.2 电机保护参数

调试中，由于误操作，会造成电机失控。为降低危险性，需设置相关保护条件。具体配置如下：

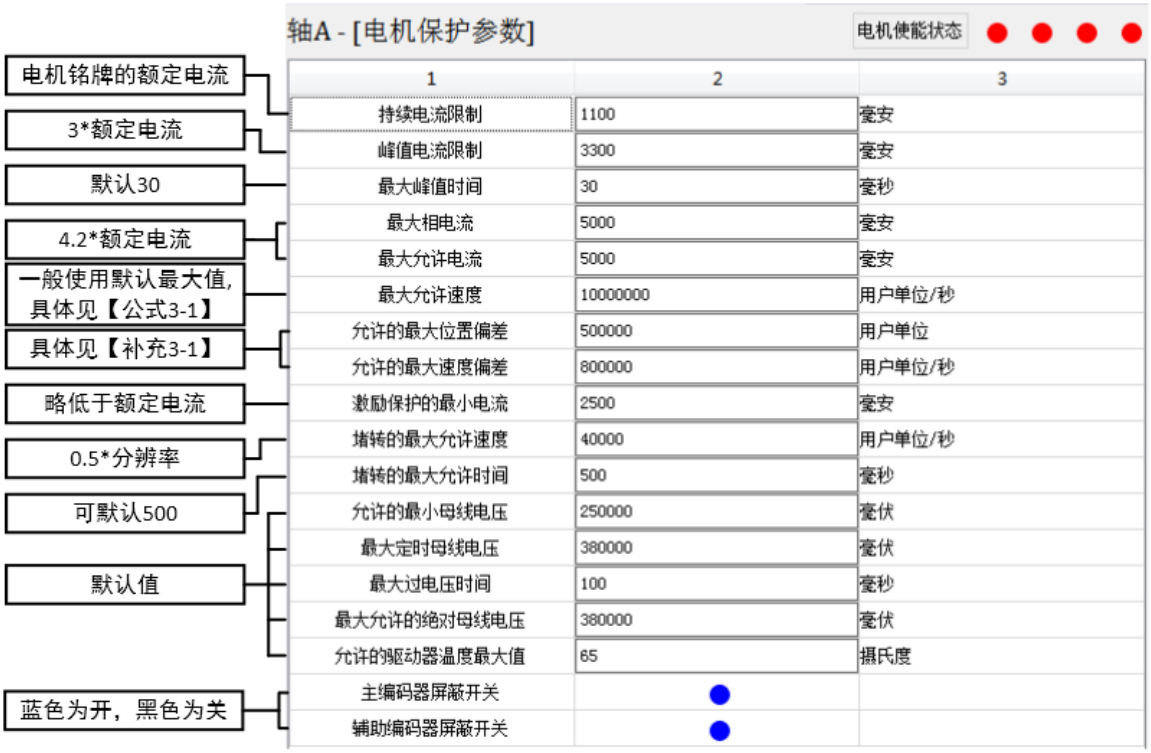


图 3-3 电机保护参数设置

【公式 3-1】

速度单位为 user-units/sec。具体换算关系式为：n（r/min）/60*2^N，N 为编码器位

数。假设电机最大允许速度为 3000r/min，编码器为 17 位。则上位机上对应设置值为：

n（r/min）/60*2^N =（3000/60）*131072=6553600user-units/sec

【补充 3-1】

偏差限制：允许的最大位置偏差、允许的最大速度偏差——初调时为防止跑飞不宜设置过大，调试完成之后设定为实际位置偏差及实际速度偏差的 2-5 倍。初始时设置为 131072，调试中可根据检测的实际偏差值进一步调整。

Motor stuck（电机堵转）：最小激活保护电流——依据实际使用设定，一般设定略小于额定电流；电机堵转最大速度/堵转时间——依据实际使用设定。根据保护设置可知，当电机转速低于“电机堵转的最大速度”，电流超过“激活保护的最小电流”且持续时间大于“堵转允许的最大时间”时，电机堵转报警。

4.3.3 换向调节

QC410 驱控一体机支持多种类型的电机及编码器。针对不同类型编码器，有几种换向模式。

表 3-4 换向模式及特点

换向模式	使用对象	执行特点
0-跳到零相位	所有的编码器	参数设置完成之后，点击“执行换向”，电机运动。
1-微动模式		参数设置完成之后，点击“执行换向”，电机微动。
2-绝对式编码器预置角度	绝对式编码器	使用“0-Jump to zero phase”换向模式，找对电机零位角之后，选用此项，保存零位角；
3-增量式编码器（带霍尔）	带霍尔脉冲输出的增量式编码器	

注意：一般情况下，保存零位角时，增量式编码器选择换向模式 1- 微动模式；绝对式编码器选择换向模式 2- 绝对式编码器预置角度。

点击换向测试方式如下所示：

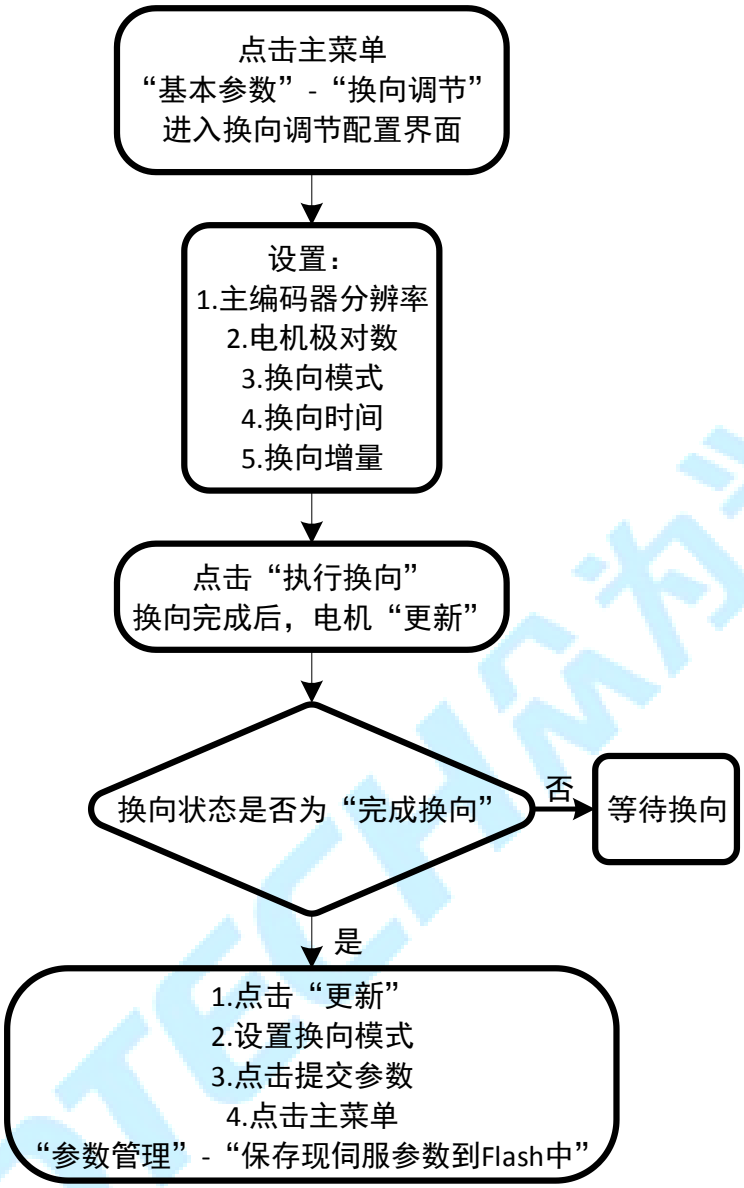


图 3-5 换向测试流程图

假设电机编码器类型为 17 位绝对式编码器，电机换向执行方法如下：主编码器分辨率(131072)——电机极对数(4)——换向模式(0-跳转到 0 相位)——换向时间(1500ms)——换向增量(4)——执行换向（执行时，换向状态显示 Not Completed，电机使能且运行一定角度，几秒之后显示 Completed 以及电机关闭）——刷新（出现一个在 0-131072 之间的数值）。此时将换向模式改为（2-绝对式编码器预置角度）——应用修改——保存到 Flash 中。

具体换相配置如图换向测试设置界面（a）、（b）、（c）所示，其中序号为操作的步骤：

图 3-6 换向测试设置界面(a)

图 3-7 换向测试设置界面(b)

图 3-8 换向测试设置界面(c)

注意：第一次找偏移角时，若电机未转动，需手动旋转电机轴之后再一次找偏移角。为避免电机零位错误对测试效果带来不良影响，应多次找电机零位，确保所找角度的正确性。

基于 QC410 驱控一体机所支持的电机编码器类型，几种电机偏移角测试结果如下。

表 3-9 常见电机类型及其换向角

电机类型	换向角
多摩川 17 位增量式、绝对式	16384
松下 2500 线	9596
松下 20 位增量式	193400
松下 17 位绝对式	71582
格特 17 位增量式、绝对式	82778
三洋 17 位绝对式	13107

已知电机偏移角情况下，用户可省去查找电机偏移角步骤，直接将角度设置到换向调节模式中，

换向调节设置方式如下：

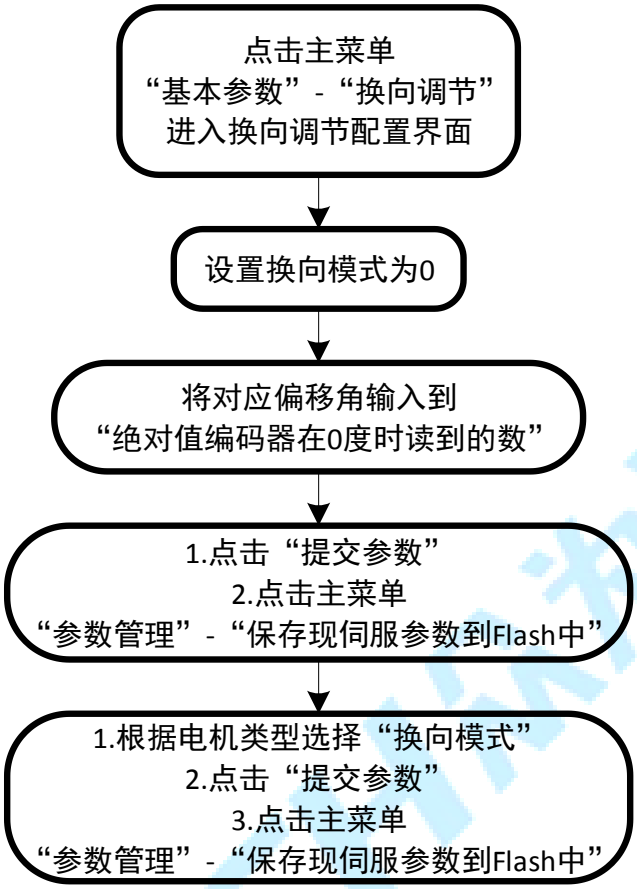


图 3-10 已知偏移角-换向角设置流程图

换向角设置界面图如下：

图 3-11 已知偏移角-换向测试设置界面(a)

图 3-12 已知偏移角-换向测试设置界面(b)

图 5-4 已知偏移角-换向测试设置界面(c)

注意：图“已知偏移角-换向测试设置界面(c)”中，①和②步骤应该按具体应用场景的要求进行设置。

图 3-13 已知偏移角-换向测试设置界面(d)

至此，已知偏移角情况下的换向角设置完成。

4.4 示波器

示波器主要包含两个主要页面，分别为：伺服数据采集，运动规划数据采集，数据分析和参数设置页面。

其中伺服数据采集，运动规划数据采集，数据分析页面的整体分布大同小异，主要分为“页面选择”，“图形显示”和“数据交互”三个部分，如图 4-1 为数据采集页面的整体分布图。

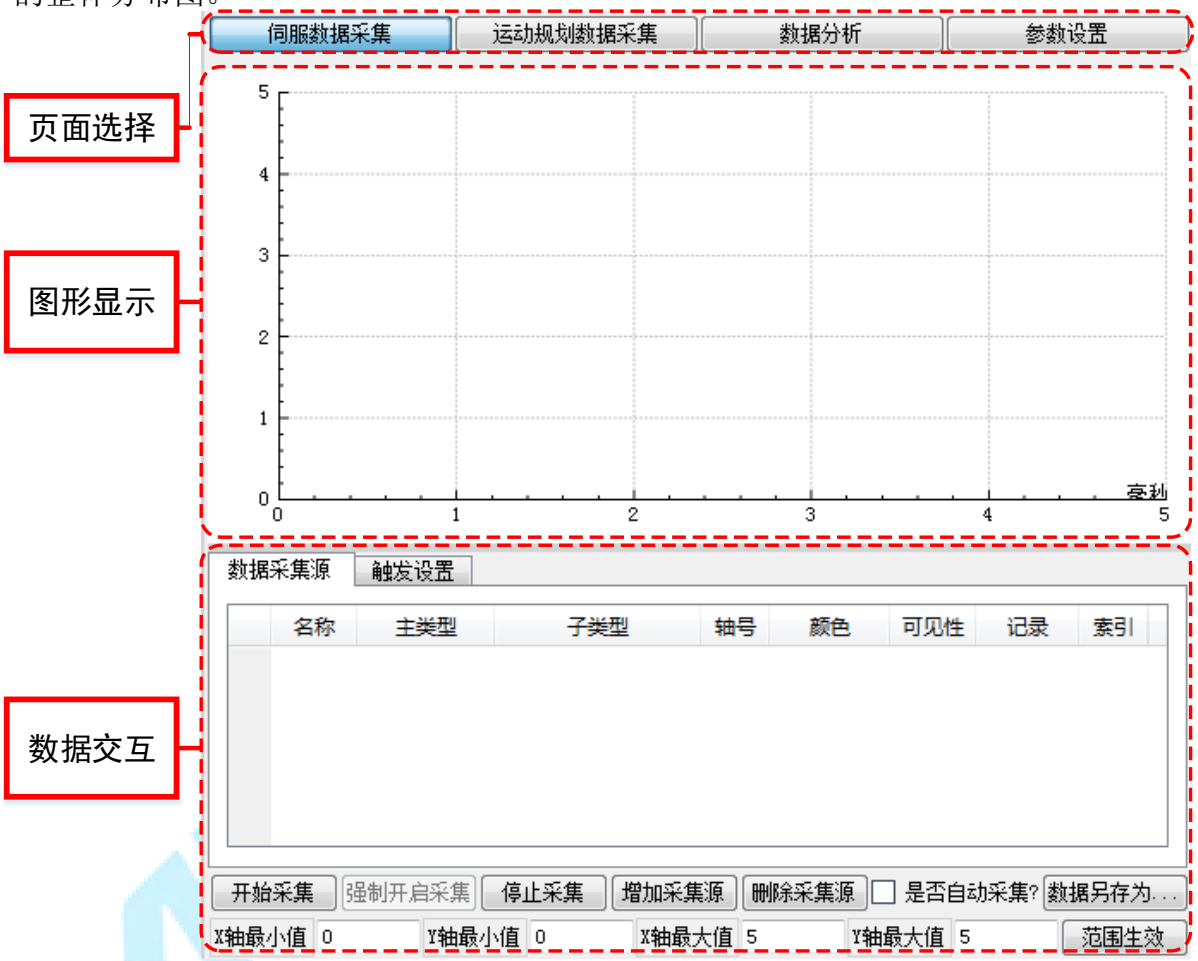


图 4-1 数据采集及分析页面整体图

伺服数据采集，运动规划数据采集，数据分析页面的整体分布大致相同，主要不同在于数据交互页面的具体操作上。伺服数据采集页面主要负责伺服数据采集的相关操作以及图形的预览显示；运动规划数据采集页面主要负责运动规划数据采集的相关操作以及图形的预览显示；而数据分析页面主要负责数据的保存，数据基础运算，性能分析以及分析结果的显示。

4.4.1 数据采集操作

“伺服数据采集”操作与“运动规划数据采集”操作一致，其不同点在于数据采集的采集源参数不同。伺服数据采集页面主要负责伺服信息的采集，而运动规划采集页面则负责运动规划信息的采集，其余的操作均一致。

4.4.1.1 数据采集源设置

（1）增加并设置数据采集源

在“数据采集”页面的“数据采集源”选项卡中，点击“增加采集源”，即可增加数据采集源的个数。增加数据采集源后，即可通过界面修改每个数据采集源的属性，设置采集源的相关属性，如图 4-3 所示。

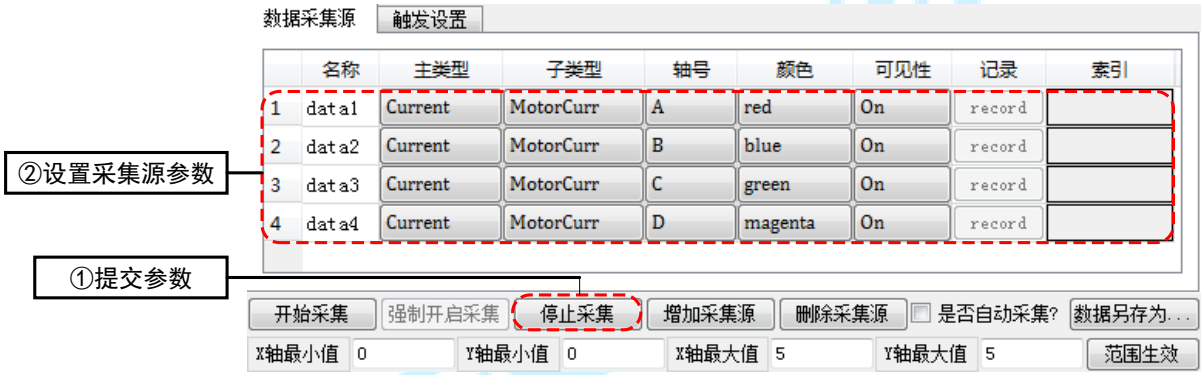


图 4-3 增加数据采集源

其中，数据采集源的个数最多为 4 个（伺服底层限制），多于 4 个将不能再增加数据采集源；数据采集源的属性主要有：“名称”、“主类型”、“子类型”、“轴号”、“颜色”、“可见性”、“记录”和“索引”。

【名称】用以标识每一个采集到的数据，主要作用在于“数据分析”页面中，对每个数据进行区分。在“数据分析”页面中，每个名称对应一个数据，一个数据对象也只能拥有一个名称。

【主类型】用以快速筛选想要采集数据的数据类型，为“子类型”的选择服务。

【子类型】用以指定需要采集的数据源。

【轴号】用以指定需要采集的数据源的轴号。“伺服数据采集”时是以轴号来区分是哪个轴的数据。

【颜色】控制数据在“图形显示”窗口中，线条的颜色。

【可见性】可以设置图形在“图形显示”窗口中，是否可见。主要用于多个数据重叠时，

提高显示的辨识性。

【记录】在数据完成采集并显示出图形时，“记录”按钮可用。“记录”用于把当前指定要保留分析的数据传递到“数据分析”页面中，加以后续分析；当且仅当该项数据采集完成后，此按钮才为可用状态。

【索引】用以指定数据源类型的索引号。“运动规划数据采集”是以该索引区分是哪个轴的数据。

注意：当数据源为“已采集数据”时，该项数据源的“主类型”、“子类型”和“轴号”为禁用状态。因为该对象已存有数据，而数据源的这些项目标识了数据本身的属性，为防止信息混淆，禁用“已采集数据”的数据源以上项目的修改。

“已采集数据”：即该项数据源已经经过采集，且保有相应的数据；

“未采集数据”：即该项数据源未经过采集，暂无相应数据；

（2）删除数据采集源

“删除采集源”按钮对“未采集数据”以及“已采集数据”的数据源均可进行删除操作（当“已采集数据”被删除后，相应的图形信息也一并删除）；

1.直接点击“删除数据源”

直接点击“删除采集源”按钮，将默认删除当前数据采集源中的最后一项采集源。

2.点击指定数据采集源前面的序号，再点击“删除采集源”，即删除该指定的数据采集源项。

（3）数据另存为

“数据另存为…”按钮仅对“已采集数据”的数据源有效。点击该按钮并设置好文件保存路径后，程序将会保存当前“已采集数据”的数据源信息到一个.csv 文件中。

在保存“.csv”文件同时，示波器默认会再同路径下另存一个同文件名的示波器“.png”文件，该文件为当前示波器图形的图形 PNG 文件。**注意，保存的文件名保持默认，不要修改，**默认是以 **STdata-年-月-日-时-分-秒.csv** 命名的文件。

（4）是否自动采集

“是否自动采集？”选项勾选后，点击“开始采集”按钮。在数据采集完成后，程序会延时一小段时间，延时过后，程序将再次自动运行“开始采集”操作，不断循环，直到“停止采集”按钮被按下。

“是否自动采集？”选项未勾选，则采集过程只将会进行一次。

（5）开始采集

当采集源设置，触发设置完成后，点击“开始采集按钮”。程序将启动数据采集的各项操作，并在采集完成后以图形的形式显示采集得到的数据。

（6）强制开始采集

在触发设置中“触发类型”不为[无]，且触发条件未满足的采集情况下，“强制开始采集”按钮有效。即当采集过程中，触发条件未满足时，点击该按钮，即可强制进行数据采集操作，而无需满足触发条件。

（7）停止采集

点击“停止采集”按钮可以停止自动采集功能或者停止当前正在进行的采集过程。

（8）图形显示范围设置

“X 轴最小值”、“Y 轴最小值”、“X 轴最大值”、“Y 轴最大值”这四个参数用于设置示波器图形显示的范围，这几个值设置好后，点击“范围生效”按钮即可生效。使用此功能须勾选结合【参数配置】中的“图形显示范围设置栏”：

4.4.1.2 触发设置

点击“触发设置”选项卡，进入触发设置界面。如图 4-4，为触发设置界面。

图 4-4 触发设置界面

（1）采集时长

该项设置采集数据的采集时间长度。

（2）采集间隔

该项设置采集数据的每个采样点之间的时间间隔。

（3）触发源主类型，触发源子类型，触发源轴号以及触发源索引

该项主要设置触发源的类型，用以确定触发采集操作所需要监测的数据。

（4）触发掩码

未知。

（5）触发类型

设置开始采集数据的触发条件。该项设置触发需要进行的逻辑判断，一旦触发源的数据满足该逻辑判断的条件，则立即进行数据采集。

（6）触发值

用于进行触发判断的比较值。

（7）触发位置%

当该项不为 0 时，在触发条件未满足的情况下，伺服将滚动缓存数据采集源的数据，缓存数据的大小为数据采集总数据量的“触发位置”百分比（即缓存大小=总数据量*触发位置%）。在满足触发条件后，伺服将不再滚动缓存，而是保留缓存区数据并继续采集数据直到采集数据量达到设定的大小为止。

该功可以方便地查看触发前后数据的变化情况，并加以分析。

4.4.1.3 注意事项

（1）采集个数限制

伺服单次总的数据采集个数不能超过 8000 个，采集数据源个数不能超过 4 个。

运动规划单次总的数据采集个数不能超过 8000 个，采集数据源个数不能超过 4 个。

例：当伺服采集源个数为 4 个时，每个数据源的最大采集平分之后为 2000 个；当采集源个数为 2 个时，每个数据源的最大采集平分之后为 4000 个。

（2）多个采集源的采集时间一致性。

采集过程中，采集数据源的数据采集行为同时发生，采集时长以及采集间隔一致。

（3）采集时间间隔限制

采集时间间隔不小于时间 T =（数据源个数）/（采样频率）ms。

4.4.2 数据分析操作

4.4.2.1 添加数据对象

数据对象添加主要有两种途径：

方法【1】

在“已采集数据”的数据源选项中，点击“record”按钮。对应的“已采集数据”源的数据就会建立一个新的副本到“数据分析”的“数据”中，在表单中添加新的一项。由于是对原数据项的副本，因此用户对其进行操作并不会对数据采集源的原数据产生影响，而其数据项的数据以及其它基本属性与原数据项一致。

方法【2】

在“数据分析”页面的“数据”选项卡中，点击“读数据文件信息”按钮，读取成功后，文件中的数据将加入到“数据”选项卡中的数据列表中。用户可以在点击“读数据文件信息”按钮后，查看文本预览窗口中数据文件的基本信息，了解文件中数据的情况，再决定是否插入数据。如图 4-5 所示。

图 4-5 读入分析数据文件

其中，数据项的个数原则上无上限个数（伺服底层限制），但数据项的名称不能重复；数据项的属性主要有：“名称”、“伺服指令码”、“颜色”、“可见性”、“信息”。

【名称】用以标识每一个数据，每一个数据项只能有唯一一个名称。

【伺服指令码】用以标识该数据项的数据源，在伺服指令表中对应的伺服指令。

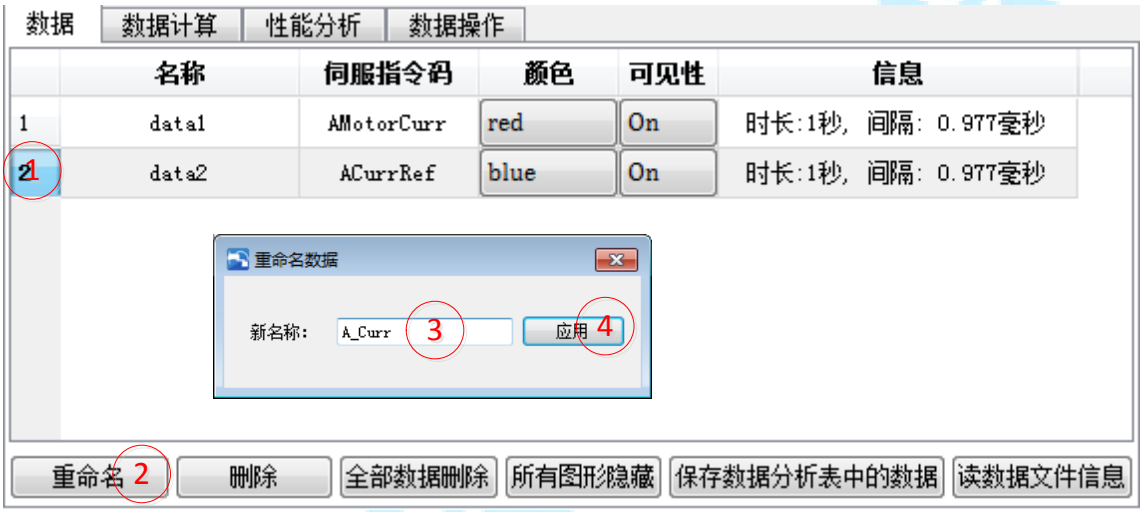
【颜色】控制数据在“图形显示”窗口中线条的颜色。

【可见性】可以设置图形在“图形显示”窗口中，是否可见。主要用于多个数据重叠时，提高显示的辨识性。

【信息】用以简要地显示该数据项的采集信息。

4.4.2.2 重命名数据

修改数据项名称的方法为：点击指定数据项前面的序号，再点击“重命名”，即弹出窗口中键入新的名称，然后点击“应用”即可。



4.4.2.3 删除数据

“数据分析”-“数据”中“删除”按钮的具体操作与“数据采集”-“数据采集源”中的“删除采集源”按钮的操作逻辑一致。

➢ 直接点击“删除”

直接点击“删除”按钮，将默认删除当前“数据”列表中的最后一项数据。

➢ 点击指定数据项前面的序号，再点击“删除”，即删除该指定的数据采集源项。

4.4.2.4 全部数据删除

点击“全部数据删除”按钮将数据列表中的所有数据源信息删除；

4.4.2.5 所有图形可见/所有图形隐藏

此按钮用于可见或隐藏示波器显示框中的图形曲线；

4.4.4.6 保存数据分析表中的数据

此操作与数据采集操作中的“数据另存为”操作一致。

4.4.2.7 读数据文件信息

在“数据分析”页面的“数据”选项卡中，点击“读数据文件信息”按钮，读取成功后，点击“把数据插入到分析表中”，再次成功后，文件中的数据将加入到“数据”选项卡中的数据列表中。用户可以在点击“读数据文件信息”按钮后，查看文本预览窗口中数据文件的基本信息，了解文件中数据的情况，再决定是否插入数据。如图 4-5 所示。

4.4.2.8 数据处理之数据计算

在“数据分析”页面的“数据计算”选项卡中，可以通过点选“数据”和“计算操作”，对数据进行简单的计算操作。

其中每个项之间的含义如下：

【1】“数据 1”或“数据 2”：用以指定参与计算的数据

第一个下拉窗口用以指定数据项，每一个选项的名称对应“数据分析”页面的“数据”列表中的数据项。第二个下拉窗口，用以指定参与计算的数据是“X”还是“Y”轴的数据信息。

【2】数据间计算操作：用以设定数据之间的计算方式，计算操作的双方均为数据。

【3】常数计算操作：用以设定指定数据与常数之间的计算方式，计算的一方为数据项，另一方为常数。

【4】数据保存名称：用以指定计算完成之后，数据项的名称。

【5】执行计算并保存数据：执行计算操作并将计算结果以数据项的形式，保存到“数据分析”页面的“数据”列表中，其名称为“数据保存名称”中的文本；

数据计算的操作流程如图 4-6 所示。

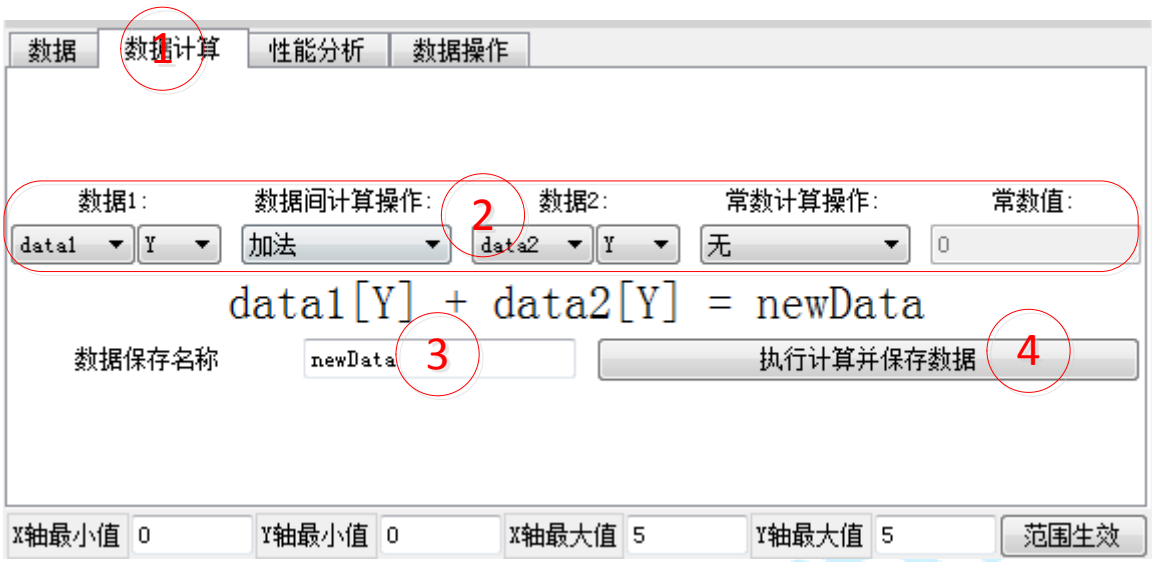


图 4-6 数据计算操作流程图

4.4.2.9 数据处理之性能分析

在“数据分析”页面的“性能分析”选项卡中，可以分析数据的一些性能指标。其中，各个控件的含义如下：

【1】分析类型：指定数据分析的分析类型，主要有“统计”性能和“阶跃”性能分析。

【2】数据名称：指定被分析的数据。

【3】计算范围：指定数据分析过程中，纳入分析计算的数据的范围，主要有三种范围选项:“完整数据”、“X 标线切分的数据”和“X 轴显示范围的数据”。

“完整数据”：把整个数据项的所有数据纳入计算范围内。

“X 标线切分的数据”：把两条 X 标线之间的数据纳入计算范围内。

“X 轴显示范围的数据”：把“图形显示”范围中，X 轴显示范围内的数据纳入计算范围内。

【4】执行计算：执行当前设置的计算选项，并把计算的结果显示到下方的文本框中。

【5】清除结果：清除文本框中的显示结果。

性能分析的操作流程如图 4-7 所示。

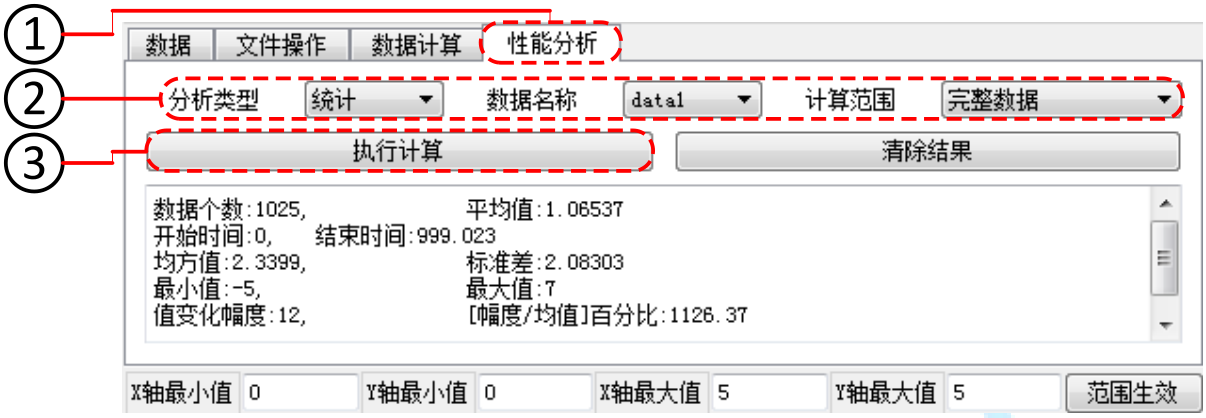


图 4-7 性能分析操作流程图

4.4.2.10 数据处理之数据操作

在“数据分析”页面的“数据操作”选项卡中，可以对分析数据进行一些计算操作。其中，各个控件的含义如下：

【1】操作类型：指定数据操作的操作类型，主要有“数据偏移”和“快速傅里叶变换”。

【2】数据名称：指定被分析的数据。

【3】执行计算并保存数据：执行当前设置的计算选项，并把计算的结果保存到“数据”页面中，其结果的名字为“数据保存名称”中的名字。

数据操作的操作流程如图 4-8 所示。

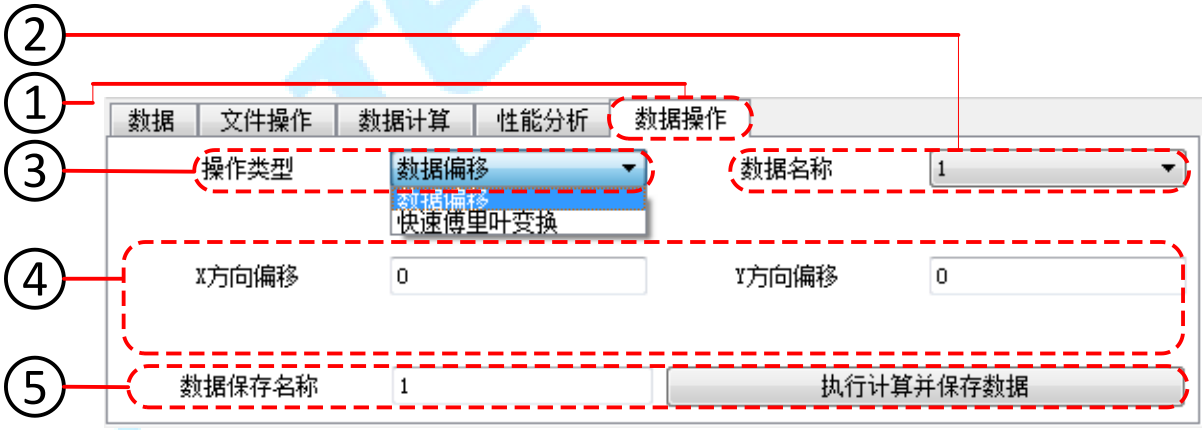


图 4-8 数据操作界面操作流程图

其中，④部分的参数内容随“操作类型”的不同而发生改变

4.4.2.11 注意事项

（1）由于“数据分析”和“数据操作”中的数据项是以属性“名称”来标识每一个数据，当新数据项的名称已存在时，将无法插入新数据项。此时若要加入该新的数据项时，

在不改变新数据项的名称的前提下，只能：1.删除同名称的数据项；2.重命名同名称的数据项；3 在新的数据项的名称中加入程序定义的后缀。

（2）“读数据文件信息”功能中，被读取的文件应符合格式要求(**STdata-年-月-日-时-分-秒.csv 和 AlertData-年-月-日-时-分-秒.csv 两种命名格式的文件**)，否则将产生不能读取文件的报错或程序异常错误。

（3）数据计算中，数据间操作必须确定数据的个数以及数据项的采集时长，采集时间间隔一致，否者会因为数据计算不匹配而产生报错。

4.4.3 参数设置页面

参数设置页面主要设置一些较为特殊的上位机操作参数，主要为一些上位机界面开关参数设置以及界面参数值设置，这些参数值在每一次上位机重新启动时会自动重置。

图 4-9 参数设置界面

（1）保存图片文件

选择是否在保存示波器数据时，同时保存示波器当前的图形文件。(默认为功能开启)

（2）同步数据采集

选择是否“伺服数据采集”和“运动规划数据采集”同时进行“开始采集”操作。在选上该项时，点击“伺服数据采集”或“运动规划数据采集”页面-“数据采集源”-“开始采集”按钮，该点击即视同同时点击两个界面的“开始采集”按钮，进行同步数据采集。(默认为功能关闭)

注意事项：

① 当两个页面中(“伺服数据采集”和“运动规划数据采集”页面)其中一个页面的数据采集设置参数，参数设置错误时，即会报错并不会进行数据采集操作。用户可以通过错误提示框中的内容，判断错误发生的位置。如图 4-10 所示，报警信息会提示

错误发生的位置。

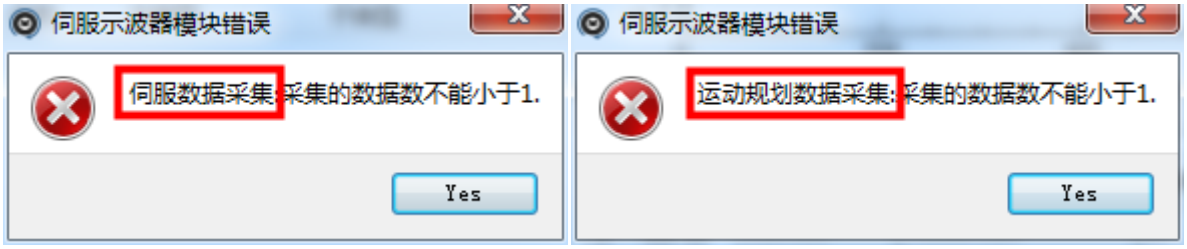


图 4-10 示波器数据采集参数设置错误-“伺服采集”与“运动规划采集”

② 当两个页面中的触发条件不一致时，不能保证两边的数据时同一时间采集的到的数据，因为采集开始时刻未必一致。.

③ 设置启用该功能后，示波器默认先接收“伺服采集信息”，然后接收“运动规划采集”的信息，因此总的数据回传等待时长将会是两者的叠加。

（3）页面信息自动更新刷新

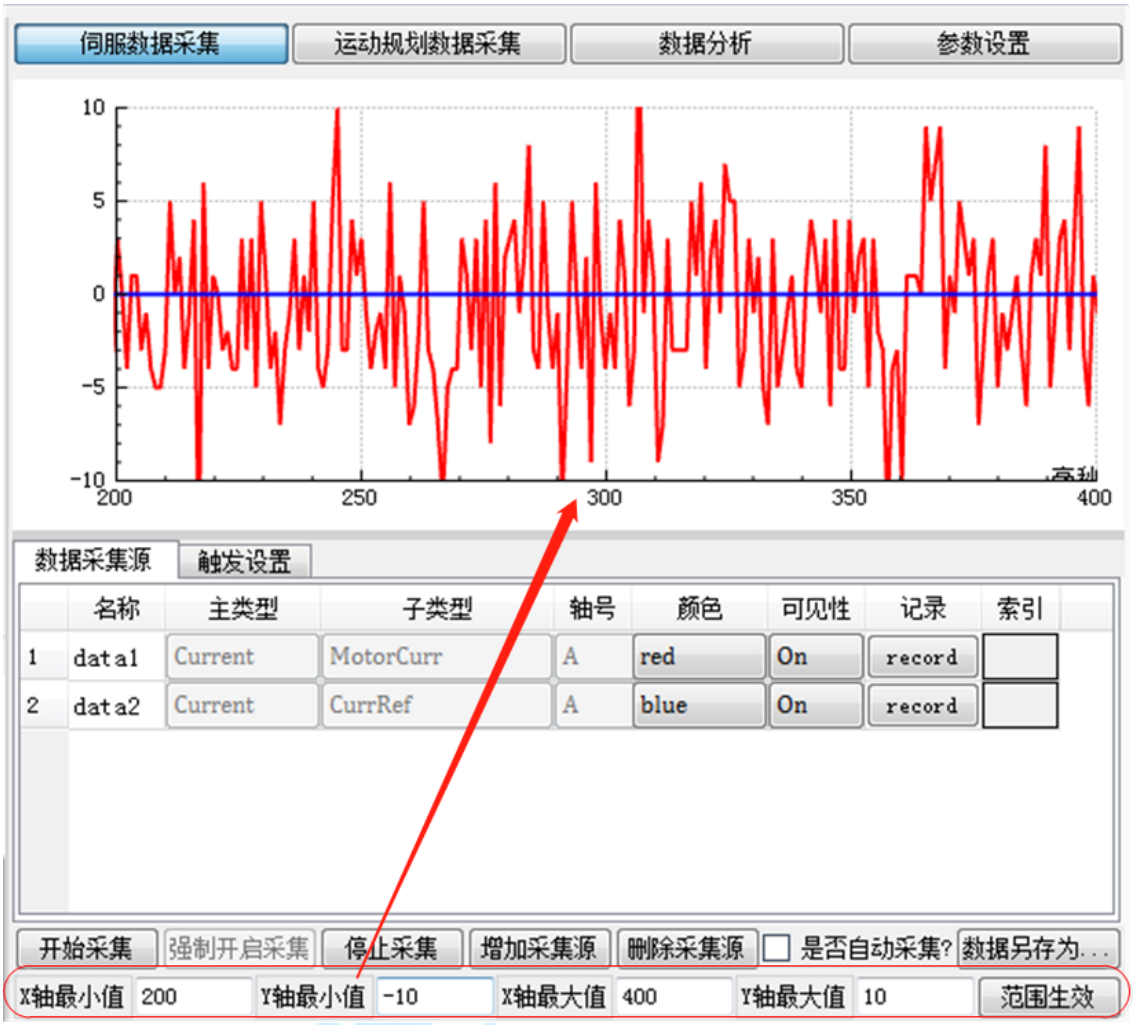
该功能关闭后，一些页面的自动更新刷新将会停止，例如：电机使能状态指示灯的状态，伺服状态信息窗口等等。该功能主要用于程序调试时，禁止过多的打印信息刷新，以便程序调试。(默认为功能开启)

（4）图片保存宽度（像素）和图片保存高度（像素）

此处控制数据采集信息另存为时，图片输出的分辨率。默认分辨率为 1280×720。

（5）图形显示范围设置栏

用于设置示波器界面图形的有效显示区域。此功能默认是不开启的。



4.4.4 图形显示操作

双击图形显示区域，可以用于隐藏或可见“示波器数据项显示表”

4.4.4.1 图形显示范围设置

示波器的图形显示的范围，有多种的控制设定方法，下面将逐一介绍：

【1】精确的范围设定：

在示波器的“数据采集”和“数据分析”页面的最下方，有一个设置 X 轴最小值、X 轴最大值、Y 轴最小值、Y 轴最大值的设置栏，如图 4-11 所示。

图 4-11 图形显示范围精确设置

用户可以通过更改该栏的参数，然后点击“范围生效”按钮来精确地设定图形的显示范围。该设置栏不会因选项卡的切换而切换，但其在“数据采集”和“数据分析”页面中，分别相对应地控制各自页面的图形的显示范围。

【2】至【5】下的操作，为在“图形显示”界面上的操作，因此以下任何鼠标的事件均发生在“图形显示”区域内。

以下图 4-12 为右击菜单栏中的显示范围控制选项。

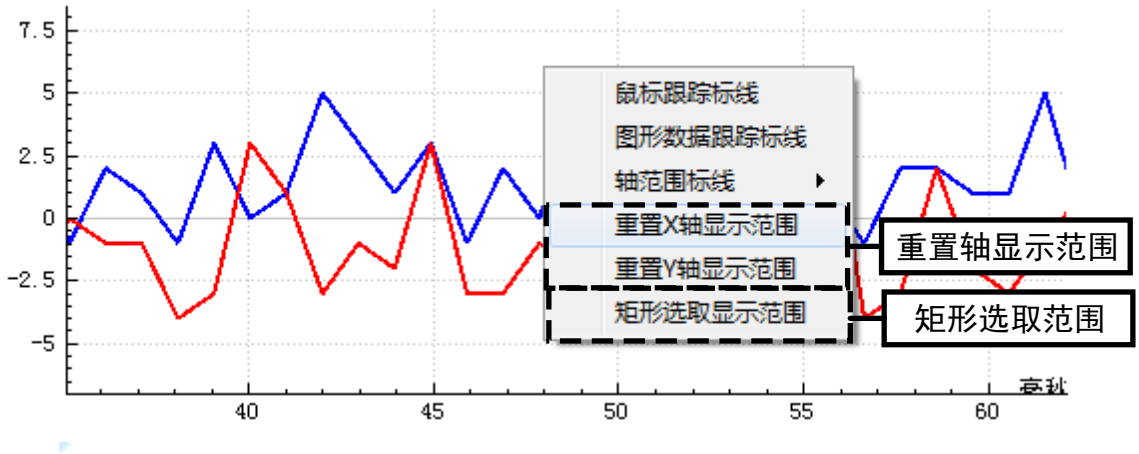


图 4-12 右击菜单显示范围控制选项

【2】鼠标右击菜单栏的“重置 X 轴显示范围”或“重置 Y 轴显示范围”

执行“重置 X 轴显示范围”操作选项，图形的 X 轴显示范围将更变为“所有可见数据的 X 轴范围” (在图形中可见的数据占有 X 轴的涵盖范围)

执行“重置 Y 轴显示范围”操作选项，图形的 Y 轴显示范围将更变为“所有可见

数据的 Y 轴范围” (在图形中可见的数据占有 Y 轴的涵盖范围)

【3】鼠标右击菜单栏的“矩形选取显示范围”

执行该操作选项，然后使用鼠标左键拉伸（鼠标左键下压后移动鼠标）出一个矩形范围。拉伸完毕后，图形的显示范围为所拉伸的范围。即此时 X 轴显示范围为拉伸矩形的 X 轴最小值到拉伸矩形的 X 轴最大值；Y 轴同理。

【4】鼠标左键拖动（鼠标左键下压后移动鼠标）

执行该操作，图形的显示范围会随着光标的移动而发生平移。当鼠标左键松开后，平移的操作状态取消。

该平移有三种状态：

① 当鼠标左键下压，光标位于“图形显示”区域上，且不靠近 X 轴和 Y 轴的轴线及轴刻度附近时：图形随着光标在 X 方向上和 Y 方向上同时发生平移。

② 当鼠标左键下压，光标位于“图形显示”区域上，且光标靠近 X 轴轴线或轴刻度附近时：图形只随光标在 X 方向上发生平移（不理会光标在 Y 方向上的移动）。

③ 当鼠标左键下压，光标位于“图形显示”区域上，且光标靠近 Y 轴轴线或轴刻度附近时：图形只随光标在 Y 方向上发生平移（不理会光标在 X 方向上的移动）。

【5】鼠标滚轮缩放显示范围

执行该操作，图形的显示范围会随着鼠标滚轮的滚动而产生相应的范围缩放。

该平移有三种状态：

① 当鼠标滚轮滚动时，光标位于“图形显示”区域上，且不靠近 X 轴和 Y 轴的轴线及轴刻度附近时：图形的缩放范围会以光标当前指示的点的位置为中心，同时产生 X 方向和 Y 方向上范围的缩放。

② 当鼠标滚轮滚动时，光标位于“图形显示”区域上，且光标靠近 X 轴轴线或轴刻度附近时：图形的缩放范围会以光标当前指示的点的位置为中心，只产生 X 方向上范围的缩放。

③ 当鼠标滚轮滚动时，光标位于“图形显示”区域上，且光标靠近 Y 轴轴线或轴刻度附近时：图形的缩放范围会以光标当前指示的点的位置为中心，只产生 Y 方向上范围的缩放。

4.4.4.2 鼠标标线的显示

在“图形显示”界面上鼠标右击，勾选选择“鼠标跟踪标线”。此时，以光标指示的点为中心点，会显示横竖两条标线，该标线会随着光标在“图形显示”区域内跟踪光标的移动，这两条标线为“鼠标跟踪标线”，如图 4-13。

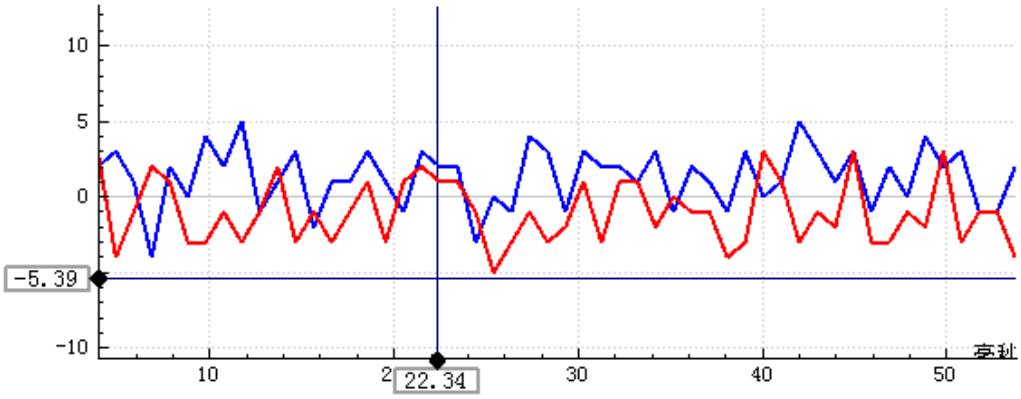


图 4-13 鼠标跟踪标线

4.4.4.3 图形数据跟踪标线的显示

在“图形显示”界面上鼠标右击，勾选点击“图形数据跟踪标线”。此时，若“图形显示”区域内有可视数据图形时，在数据图形上会显示一个数据值标签，这个数据值标签依附在数据图形曲线的一点上。

该点是在数据项中的 X 值与光标的 X 轴坐标值最接近且小于等于光标的 X 轴坐标值的数据点；而标签上的值为该点数据对应的 Y 值。图 4-14 为图形数据跟踪标线的效果图。

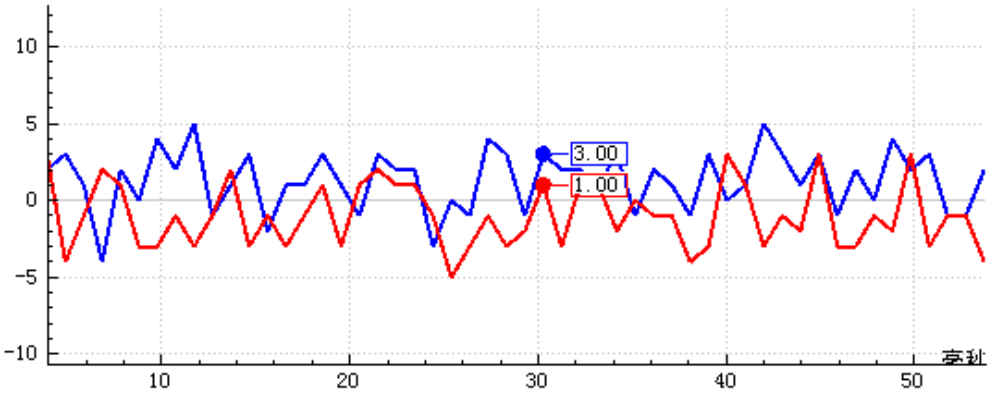


图 4-14 图形数据跟踪标线

4.4.4.4 X 和 Y 标线的设置及显示

在“图形显示”界面上鼠标右击，选择“轴范围标线”，即显示轴范围标线选择菜单，如图 4-15 所示。

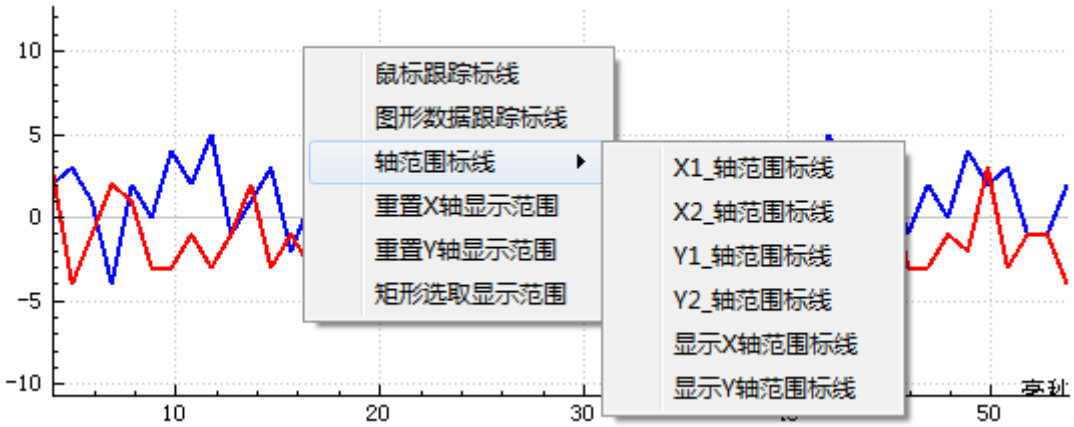


图 4-15 轴范围标线菜单

轴范围标线共有四条，分别为“X1 轴范围标线”、“X2 轴范围标线”、“Y1 轴范围标线”、“Y2 轴范围标线”。四条标线互不干扰，其位置各自独立。

轴范围标线可应用于“数据分析”-“性能分析”的计算范围设置中，也可用于用户直接在图形上分析观察数据。

（1）更改轴范围标线位置

只需要点击对应的轴范围标线选项，“图形显示”区域即会出现对应的成对的轴范围标线，且关闭相反的轴范围标线（例：点击 X1_轴范围标线时，会显示“X1 轴范围标线”和“X2 轴范围标线”，且关闭“Y1 轴范围标线”和“Y2 轴范围标线”的显示）。此时，对应的标线会跟踪鼠标的位置进行位置跟踪并显示标线位置值；只要鼠标进行任意点击操作，标线就会停止跟踪，并保持最后跟踪的位置。如图 4-16 为 X 轴范围标线的跟踪示意图。

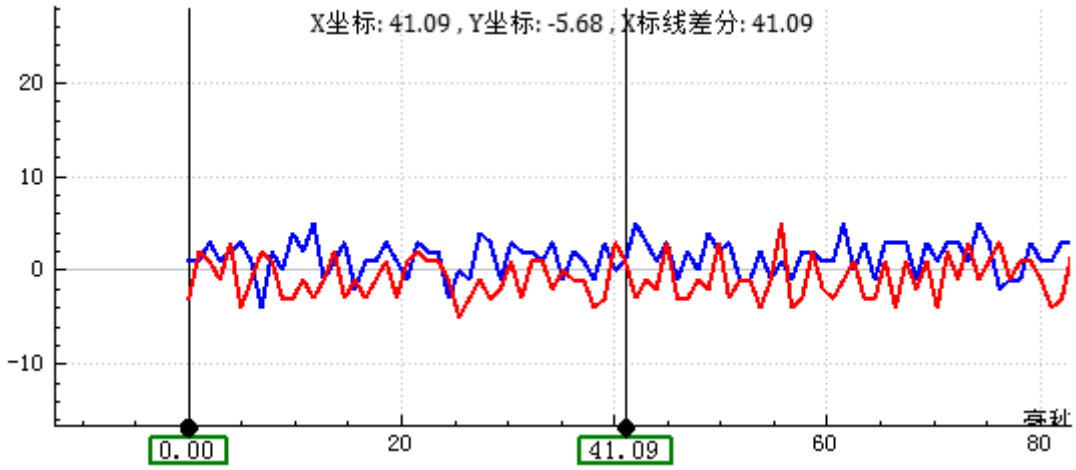


图 4-16 X 轴范围标线的跟踪示意图

（2）显示数据范围标线

只需要点击相应轴的“显示轴范围标线”，即可显示对应轴的轴范围标线。

当 X 轴范围标线被显示时，“图形显示”区域的上方会显示当前光标的位置，以及当前 X 轴标线之间的差分绝对值。

4.5 运动调试

运动调试主要为“点到点”和“PID 调试”。

在进行基本参数设置完成之后，需要用上位机软件监测伺服编码器端接线或类型选择是否有误。具体测试方法为：

1）选择主菜单栏中“运动调试”—“点到点”模式。

2）选择需要检测的当前轴，记下状态栏中该轴的当前位置（假设为 P1）。

3）手动旋转当前轴对应的机器人手臂一定角度，记下该轴此时位置（假设为 P2）。

4）判断两次位置脉冲是否有变动（P1 与 P2 的差值不为 0）。

5）位置脉冲有变动，初步判断编码器端无误，按顺序执行下一步；若位置脉冲没有变动，表明检测不到编码器信号，编码器接线错误或者编码器类型选择不对。需再次检查接线并检验，直至脉冲有变动，才能进行后续调试，其余三轴机械臂编码器线检测方法相同。

注意：

带刹车的电机不能使用此种检验方式。

4.5.1 点到点

完成基本参数设置与调试之后，电机试运行。由于工业机器人每个机械手臂行程范围有一定限制，在此运用点到点运动对调试效果进行验证。

图 4-2 点到点运行配置流程图

点击“运动调试”，进入“点到点”调试界面，选择“点到点”标签，设置如下：

点到点运行设置界面

Acceleration（加速度）：单位为 user-units/sec^2，可将其设置为速度的 5-10 倍。

Deceleration（减速度）：单位为 user-units/sec^2，可将其设置为速度的 5-10 倍。

Speed（速度）：单位为 user-units/sec。假设 speed=1000000 user-units/sec，电机编码器位数为 17 位。转换到电机常用单位 r/min 的关系式为：1000000/131072*60=460r/min。

Repeated wait（往复延时）：表示的是从一个循环进入下一个循环等待的时间。

Target 1（位置 1）：单位为 user-units。

Target 2（位置 2）：单位为 user-units。

对初次调试的设备，试运行时，速度、加速度，位置都应当设置到较小值。设置完成之后，点击“运动到位置 1”或“运动到位置 2”，电机开始试运行，运行过程中如果未出现报警，可增大设置。有报警则应排除报警在继续调试。

利用示波器，观测实际运行过程中，电机电流、实际电流波形跟随情况，对电流环参数进行优化。速度环、位置环优化方式一致。（示波器将在后续章节中介绍）

4.5.2 PID 调试

电机零位找对之后，即可调整驱动器控制参数。在此之前，应先了解伺服控制系统从内到外包括：电流环、速度环、位置环。为精确控制伺服电机运行，需依次调节三环参数。

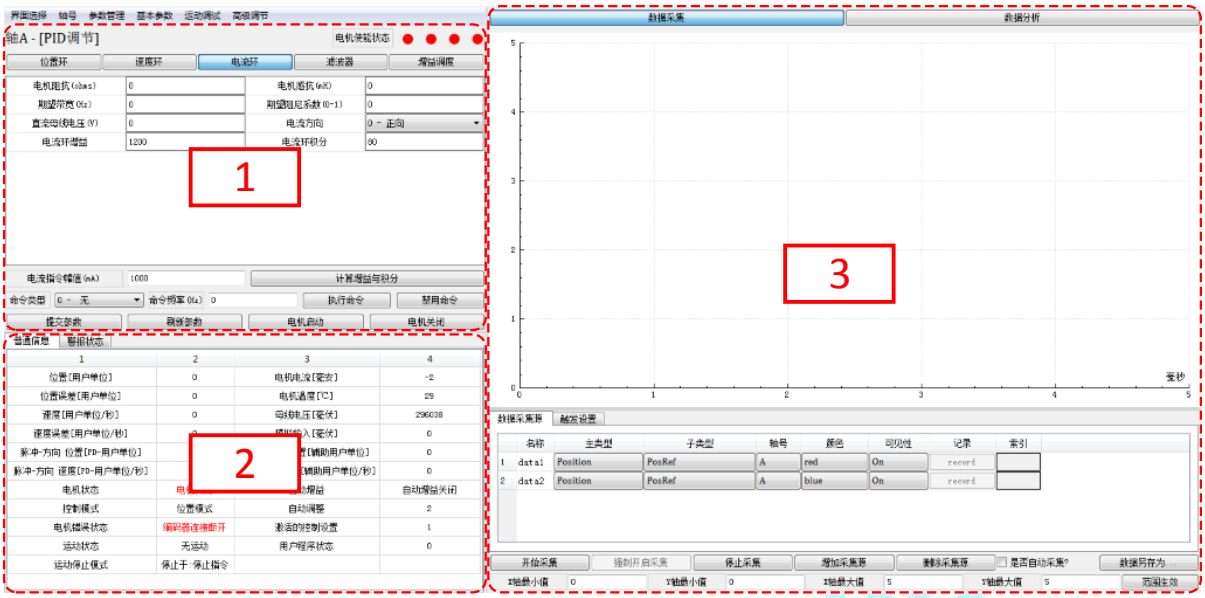


图 5-1 PID 三环调试界面

如上图所示，PID 调节窗口主要包括三个部分：[1]三环调节窗口（设置电流命令、电流环增益、积分）；[2]电机运行状态窗口（显示电机当前运行状态）；[3]示波器窗口（观测电机实际电流跟随情况）。

4.5.2.1 电流环调谐

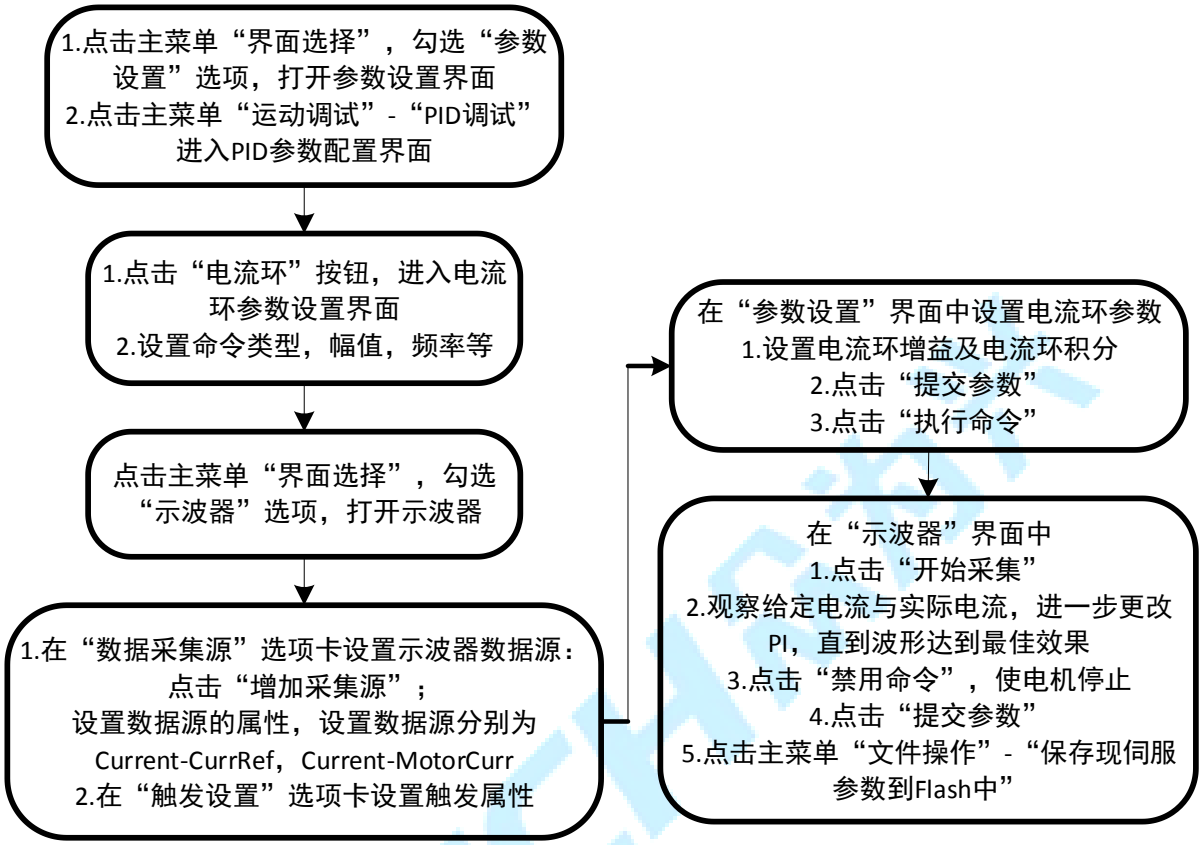


图 5-2 电流环调试流程图

在图 5-1 的[1]中点击“电流环”，进入电流环调试界面：

根据电流环调试流程图 5-2，电流环具体参数配置与调试步骤如下：

（1）电流命令设置

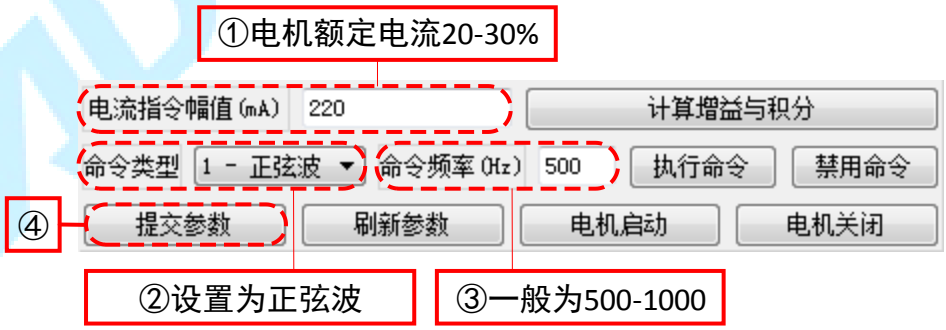


图 5-3 电流指令设置界面

设置完成之后“提交参数”，修改完成；

（2）示波器窗口设置

首先，在示波器上方，电机数据采集按钮，进入数据采集的示波器界面，如图 5-4

所示；

然后，在示波器窗口下方选项卡选择“数据采集源”，进入数据采集源设置窗口。点击增加采集源以加入测量观测数据，在电流环中，主类型为均为“Current”，子类型分别为“MotorCurr”和“CurrRef”；并选择好相应的轴号、颜色，如图 5-5 所示。

最后，在示波器窗口下方选项卡选择“触发设置”，进入触发设置窗口。根据实际需求设置触发参数，默认电流环调试设置触发类型为“0-[无]”，默认采样时长为 1s，默认采样间隔为 1ms（采样时长和采样间隔可以按照实际的需要自行调整，本次测试时长为 0.122s，间隔为 0.122ms）。如图 5-6 所示。

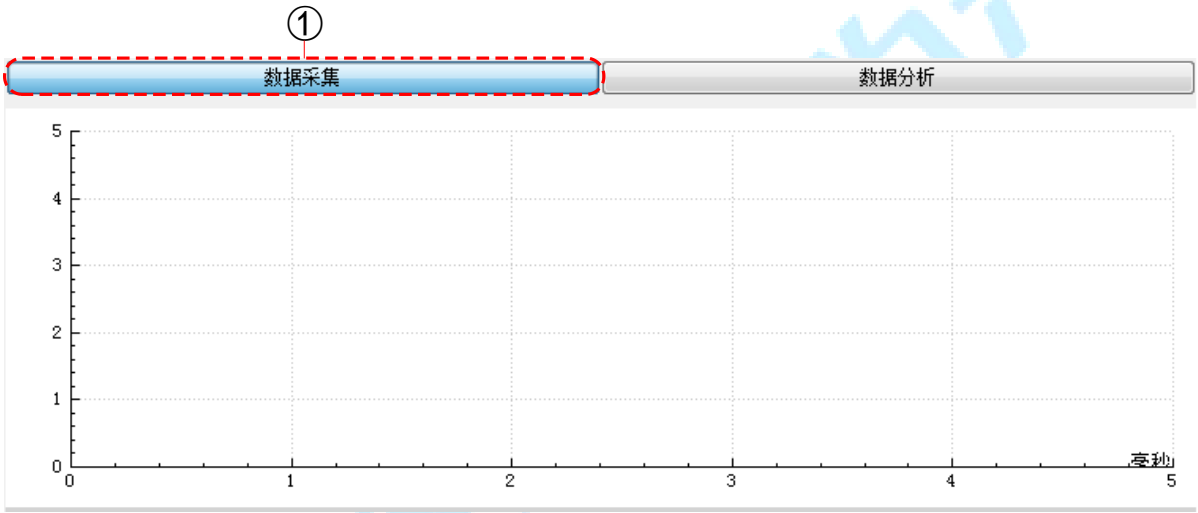


图 5-4 电流环示波器设置图(a)

图 5-5 电流环示波器设置图(b)

【注意】 QC410 采样点数小于 8000 点，示波器窗口最多可记录四个变量。

（3）电流环调节

示波器设置完成之后，回到“PID 调试”—“电流环”窗口，设置电流环增益及积分。首先不宜设置过高，从小到大设置增益及积分，然后点击“提交参数”和“电机启动”，再点击“执行命令”，如图 5-7 所示。

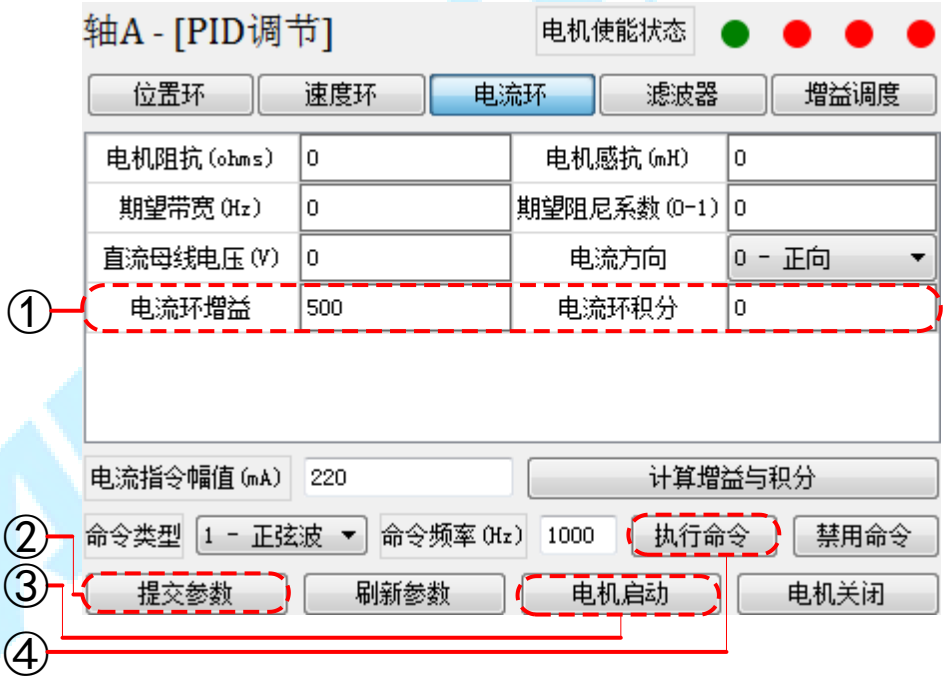


图 5-7 电流环调试界面(a)

然后点击示波器右下角的开始采集，如图 5-8 所示：

图 5-8 电流环调试界面(b)

一开始调整电流环增益，先把命令频率调节成目标命令频率的 1 倍，然后不断增大调节电流环增益，观看示波器波形，如图 5-9 所示。

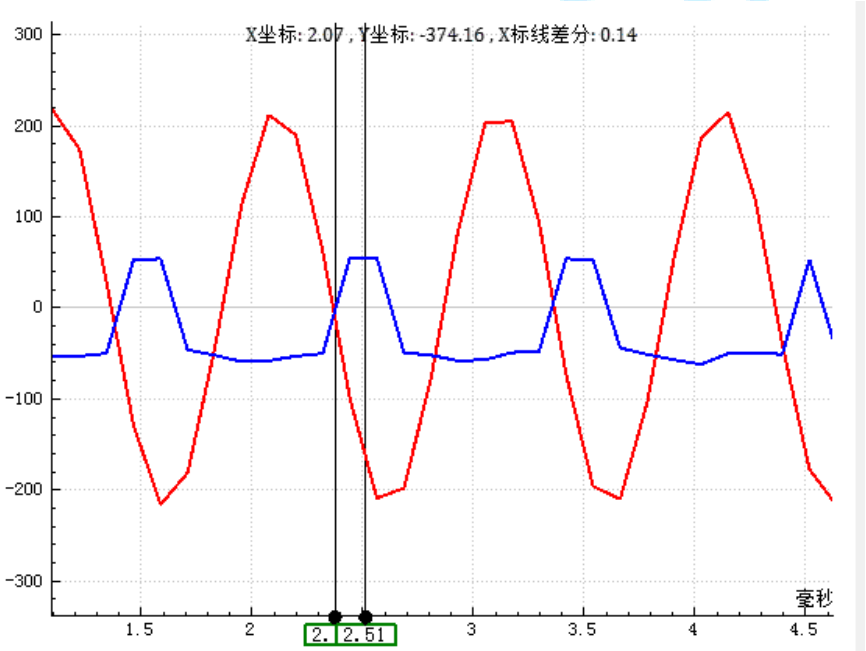


图 5-9 电流环增益调节波形(a)

如图，红色线为“指令电流”值，蓝色线为实际“电机电流值”。调整电流环增益，使其两个之间的相位差接近 1/4 个周期(90°)，则电流环增益调整完成，如图 5-10 所示。

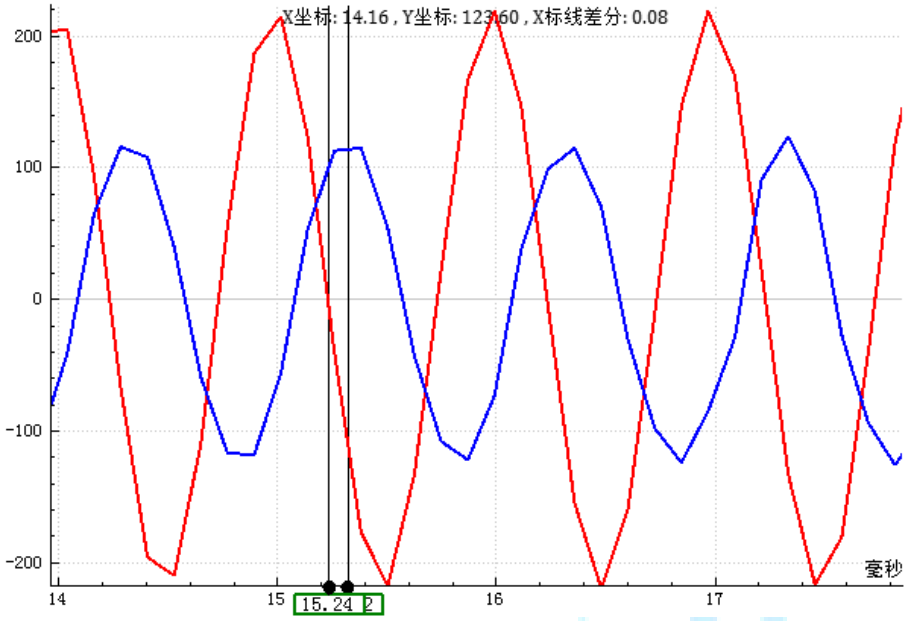


图 5-10 电流环增益调节波形(b)

然后修改命令频率为原目标频率，调整电流环积分，使波形的幅值与命令的幅值接近。至此，电流环调节完成。如图 5-11 所示。

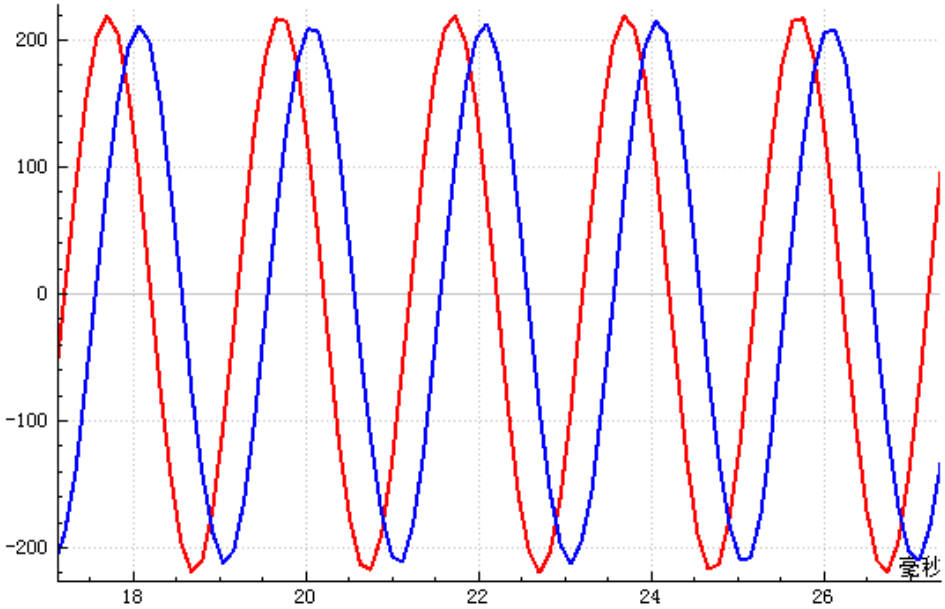


图 5-11 电流环增益调节波形(c)

实际使用中，需对调试好的电流环性能进行分析，以便不影响到后序速度环和位置环的调节。（待完善）

注意：电流环调试完成之后，将参数保存到 Flash。

4.5.2.2 速度环调谐

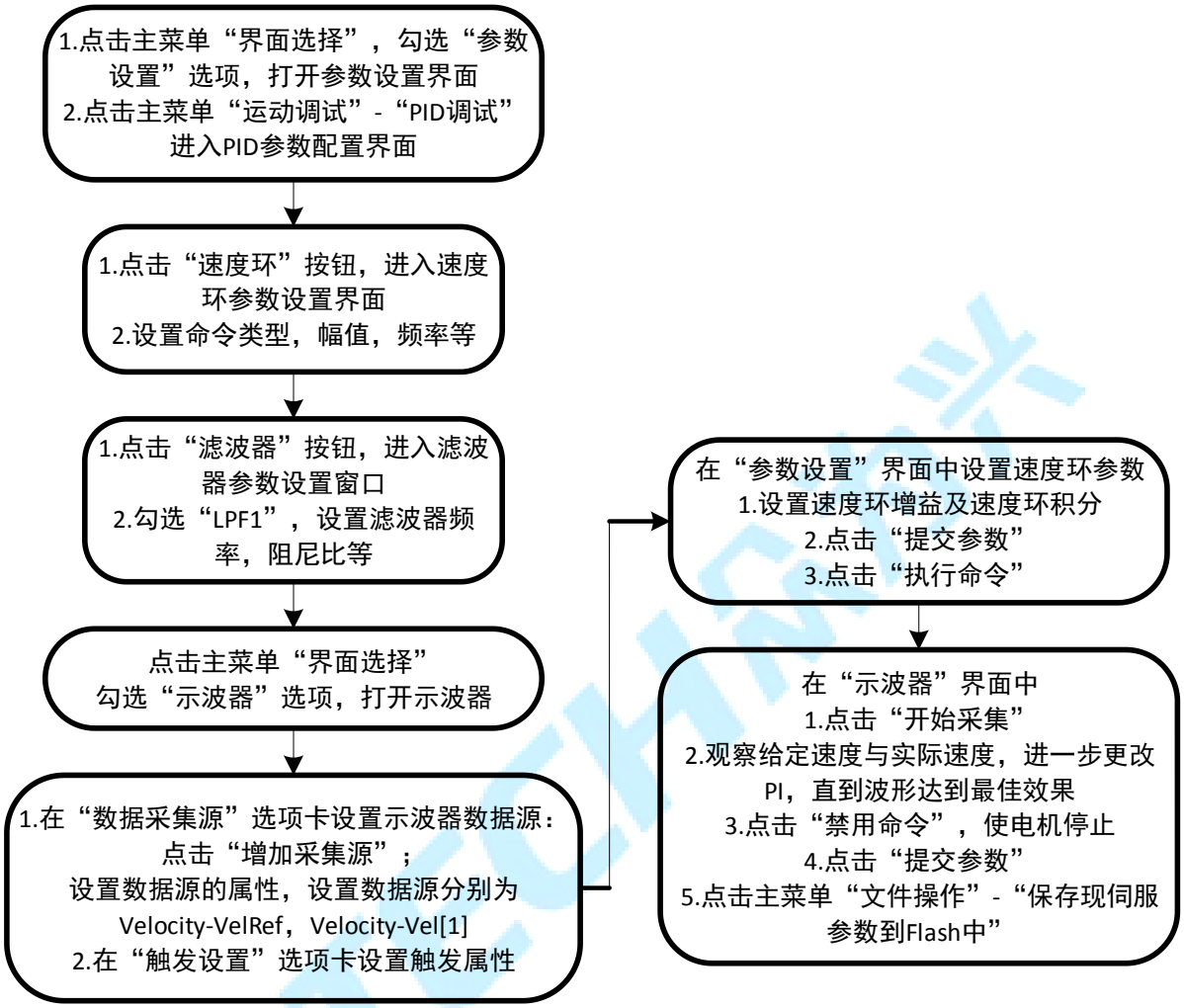


图 5-14 速度环调试流程图

点击“三环调节”，进入“速度环”调节界面，根据速度环流程图，速度环具体参数配置与调试步骤如下：

（1）速度命令窗口设置

点击“速度环”窗口中的“速度命令”标签，设置如下：

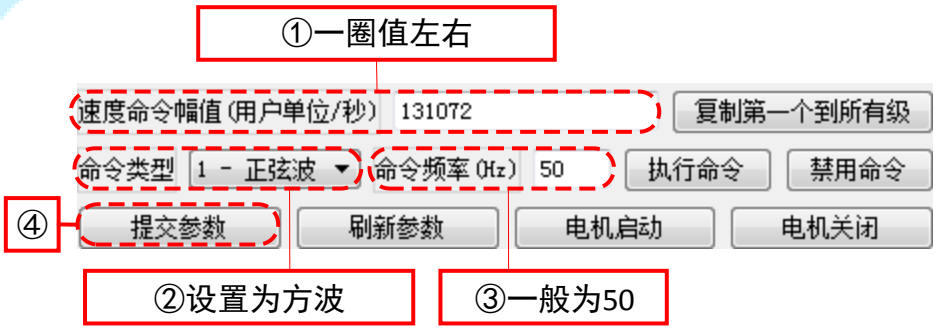


图 5-15 速度指令设置界面

设置完成之后，点击“应用修改”，修改完成；

（2）示波器窗口设置

在示波器上方，电机数据采集按钮，进入数据采集的示波器界面；在示波器窗口下方选项卡选择“数据采集源”，进入数据采集源设置窗口。点击增加采集源以加入测量观测数据，在电流环中，主类型为均为“Velocity”，子类型分别为“Vel”和“VelRef”；并选择好相应的轴号、颜色并设置好“Vel”的索引，如图 5-5 所示。

最后，在示波器窗口下方选项卡选择“触发设置”，进入触发设置窗口。根据实际需求设置触发参数，默认电流环调试设置触发类型为“0-[无]”，默认采样时长为 1s，默认采样间隔为 1ms（采样时长和采样间隔可以按照实际的需要自行调整，本次测试时长为 0.122s，间隔为 0.122ms）。如图 5-16 所示。

图 5-16 速度环示波器设置图(a)

图 5-17 电流环示波器设置图(b)

（3）滤波器设置

点击“速度环”中的“滤波器”标签，进入速度环调节界面，配置方式如下：

图 5-18 速度环滤波器设置界面

对编码器反馈的速度信号及电流给定信号滤波，截止频率依据示波器中速度环给定与反馈波形适当增减。

（4）速度环调节

滤波器设置完成之后，点击“速度环调节”标签，进入速度环调节界面，设置速度环增益及积分。首先不宜设置过高。从小到大设置增益及积分，如下界面：

图 5-19 速度环调试界面

设置完成之后，开始速度环增益与积分调节，调节方法与调试电流环方法一致，此处不在累述。记录得到的速度环波形图如图 5-20 所示：

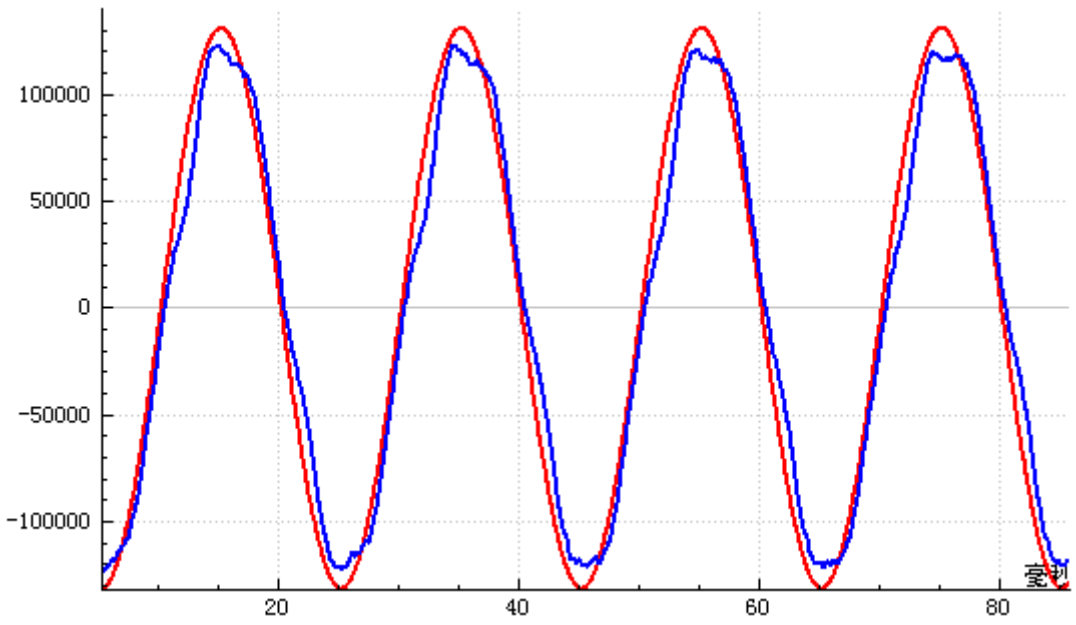


图 5-20 速度环波形图对比（1）

不加滤波的情况下，得到的实际速度波形干扰较大。增加滤波之后，对比滤波前后速度环波形图，滤波之后的波形比滤波之前的效果好。此时基本完成速度环调节。带负载的情况下，应当适当增大速度环增益。得到合适的速度环参数之后将其保存至 Flash。

4.5.2.3 位置环调谐

位置环调谐方式与电流环、速度环基本一致。

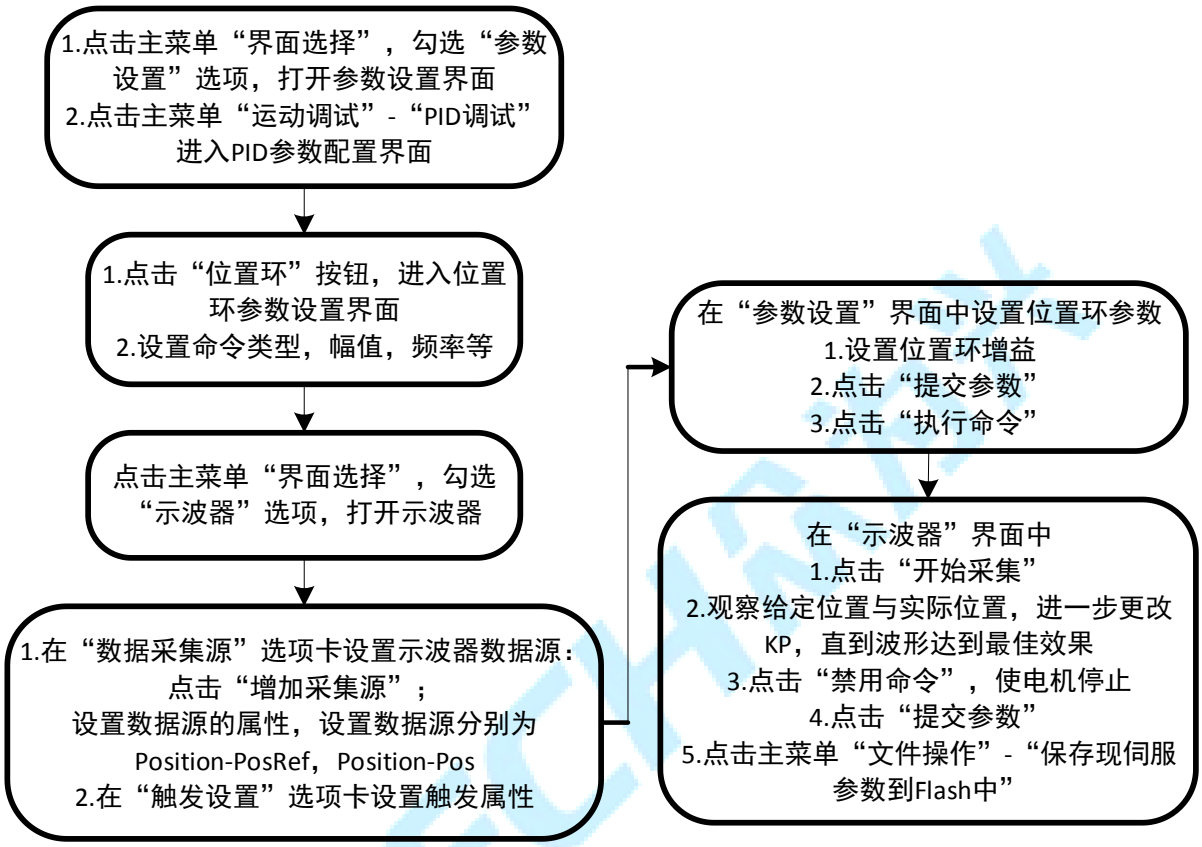


图 5-22 位置环调试流程图

点击“三环调节”，进入“位置环”调节界面，根据位置环流程图，位置环参数配置与调试步骤遵循如下顺序：

（1）位置命令窗口设置

点击“位置环”窗口中的“位置命令”标签，设置如下：

图 5-23 位置指令窗口界面

（2）示波器窗口设置

在示波器上方，电机数据采集按钮，进入数据采集的示波器界面；在示波器窗口下方选项卡选择“数据采集源”，进入数据采集源设置窗口。点击增加采集源以加入测量观测数据，在电流环中，主类型为均为“Position”，子类型分别为“Pos”和“PosRef”；并选择好相应的轴号、颜色，如图 5-5 所示。

最后，在示波器窗口下方选项卡选择“触发设置”，进入触发设置窗口。根据实际需求设置触发参数，默认电流环调试设置触发类型为“0-[无]”，默认采样时长为 1s，默认采样间隔为 1ms。如图 5-6 所示。

图 5-24 位置环示波器设置图(a)

图 5-25 位置环示波器设置图(b)

（3）位置环调节

位置环只有增益 P，调节时，在不超调的情况下，尽量增大 P 值，以增加系统的响应性。但实际增益设定，还需根据应用场合进行设置。

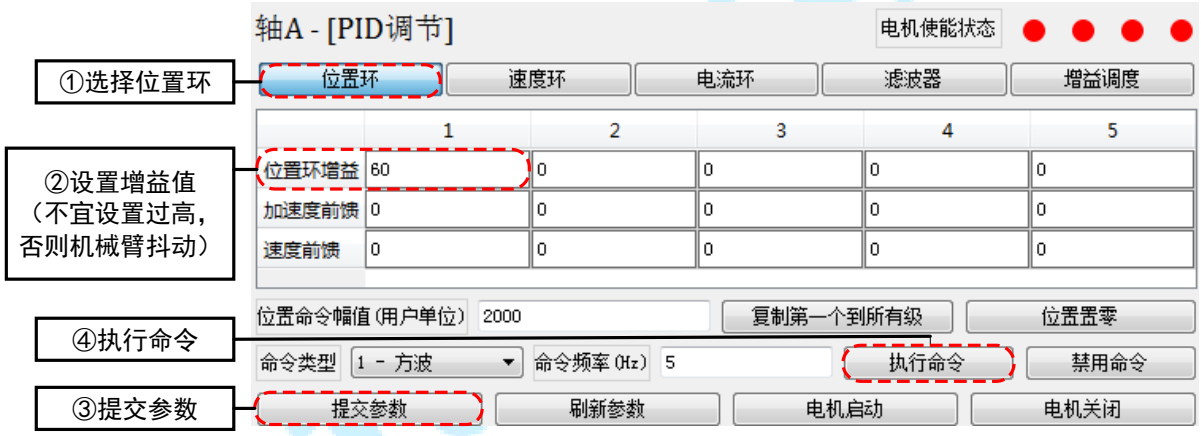


图 5-26 位置环调试界面

如图 5-26 所示，当选择波形为方波时，应适当减少命令幅值，默认为 2000（经验值）。本次图 5-27 为正弦波调试曲线，该选用正弦波来调节 PID，其幅值默认电机半圈脉冲值.

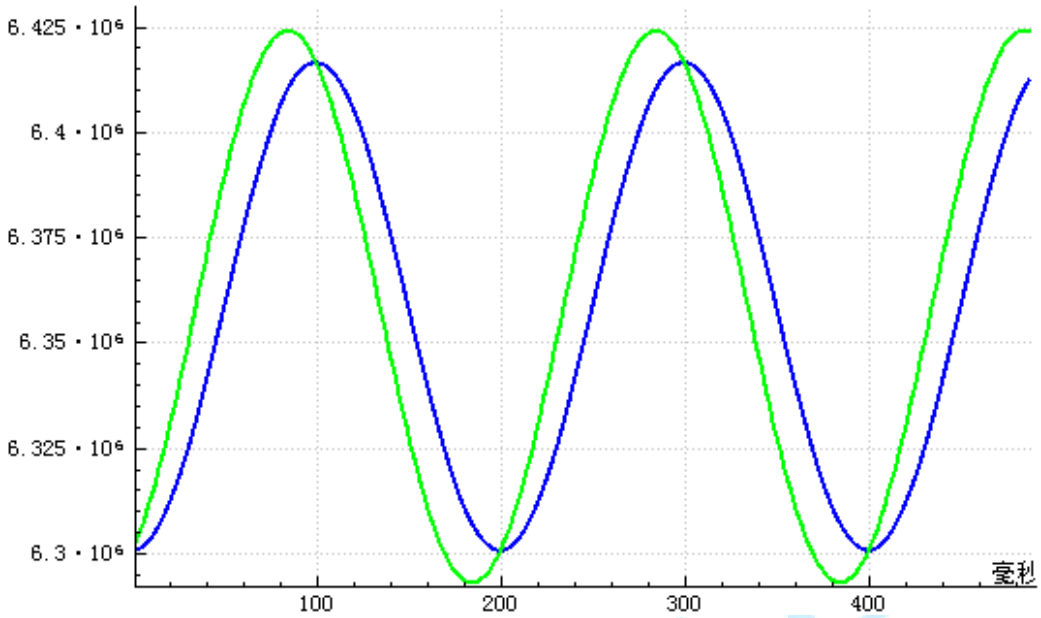


图 5-27 位置环波形图

实际应用中，位置环增益不宜设置过高，尤其是工业机器人应用等场合，过高的位置环增益会引起机械臂抖动。

4.6 高级调试

4.6.1 输入整形

输入整形功能为较高级功能，需要充分了解伺服底层运行的情况下，才可设置使用输入整形功能。

输入整形的参数设置流程如图 6-1 所示。图 6-2 为输入整形的设置界面。

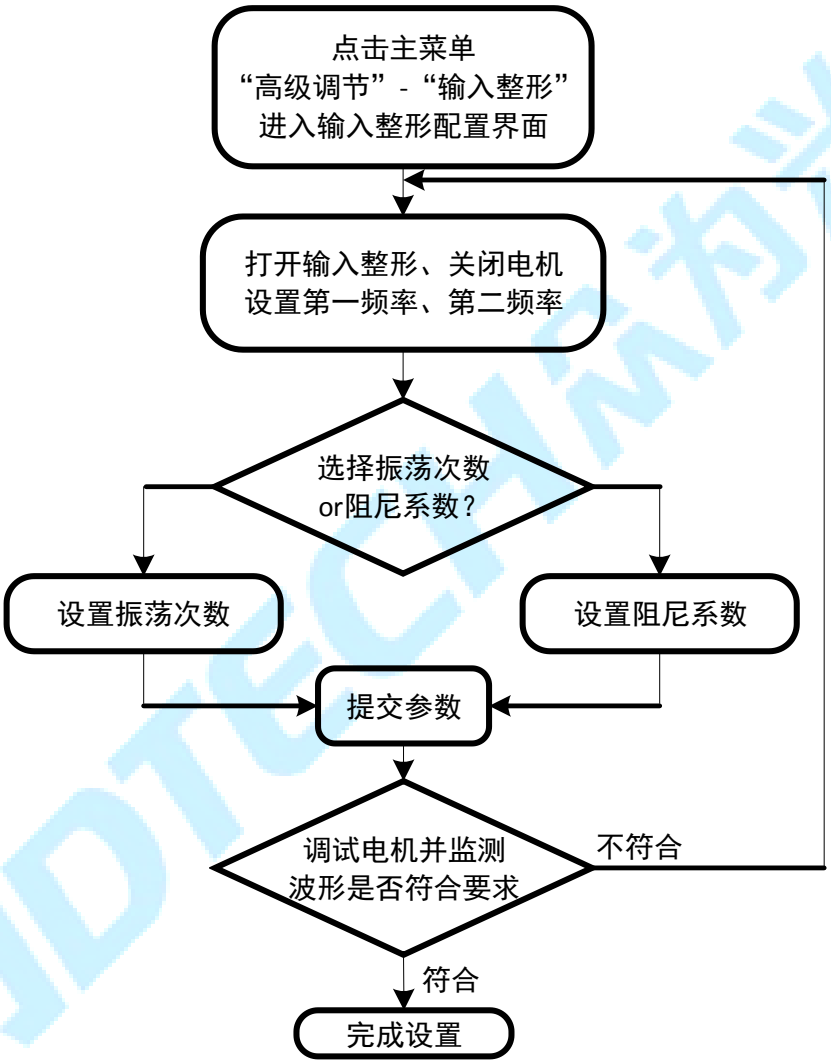


图 6-1 输入整形设置流程图

其中，第一频率和第二频率需要依照实际采集数据的振荡情况设置；而振荡次数的设置实际上是对应内建的阻尼映射表，不同的振荡次数对应不同的阻尼系数，实际上是调节阻尼系数。图 6-2 为输入整形的设置界面图。

图 6-2 输入整形设置界面图

4.7 伺服错误报警与处理

4.7.1 伺服信息显示

调试时，和使用中难免会遇到报警的状况，发生报警时，用户须排除报警，才能完成后续调试与使用。上位机调试软件中的状态栏可直观的看出当前轴的报警内容。

通过伺服调试插件可观测到当前的报警状态，便于用户查看。对应轴号的报警信息会在状态栏显示。

普通信息 警报状态

1	2	3	4
位置[用户单位]	0	电机电流[毫安]	2
位置误差[用户单位]	0	电机温度[℃]	29
速度[用户单位/秒]	0	母线电压[毫伏]	298001
速度误差[用户单位/秒]	0	模拟输入[毫伏]	0
脉冲-方向 位置[PD-用户单位]	0	辅助编码器位置[辅助用户单位]	0
脉冲-方向 速度[PD-用户单位/秒]	0	辅助编码器速度[辅助用户单位/秒]	0
电机状态	电机关闭	自动增益	自动增益关闭
控制模式	位置模式	自动调整	2
电机错误状态	编码器连接断开	激活的控制设置	1
运动状态	无运动	用户程序状态	0
运动停止模式	停止于:停止指令		

图 7-1 普通状态信息

普通信息 | 警报状态

1	2	3	4	5	6	7	8	9	10
换向	●	急停	●	欠压	●	过压	●	绝对-过压	●
反向限位	●	正向限位	●	低位置	●	高位置	●	动态刹车	●
电流饱和	●	电压饱和	●	速度饱和	●	再生制动	●	无再生制动	●
母线电压报警	●	电流报警	●	温度报警	●	饱和报警	●	其他报警	●
用户模式	●	计算滤波器	●	算滤波器失	●	上电自动执行	●	用户程序错误	●

图 7-2 报警状态信息

4.7.2 报警信息导出

在控制器发生报警时，控制器会自动将报警前 100ms 的相关报警信息保存为文件，保存在控制器内部的文件系统中。在进行机器人调试时，可以用软件 AR_Studio 将报警信息文件导出回传至上位机，操作流程如图 7-3 所示。

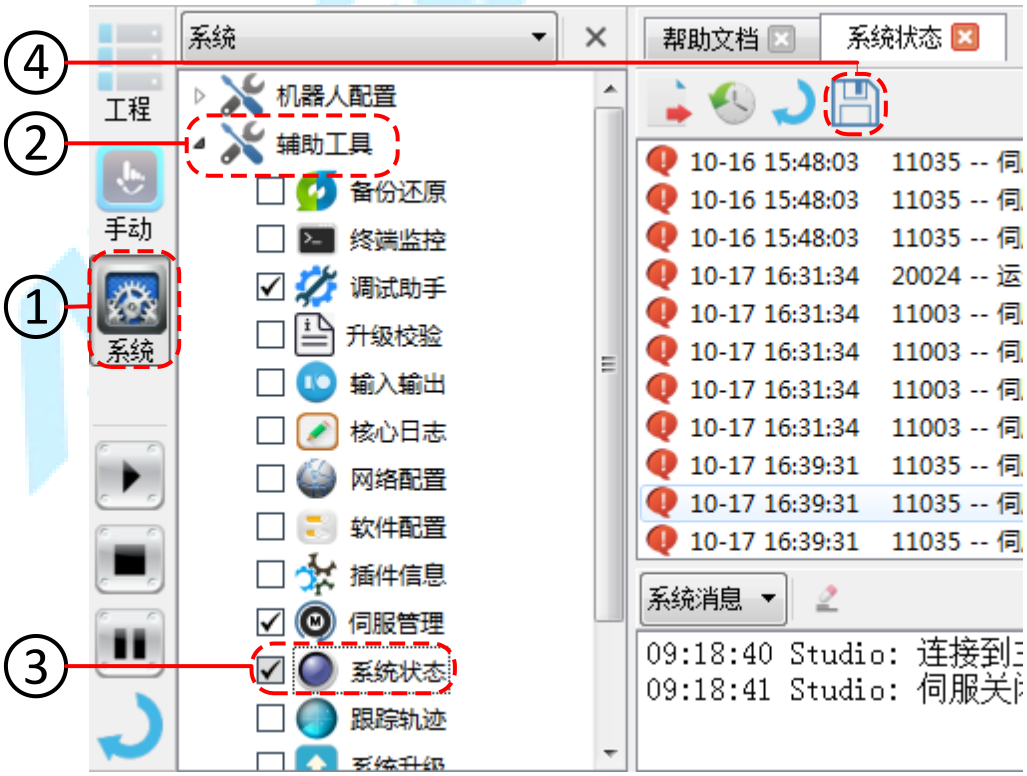


图 7-3 报警信息回传流程图

报警信息导出完成后，将弹出保存路径提示框，提示文件保存在上位机中的路径，如图 7-4。而导出的文件名以日期为区分，例如 AlertData_2019-09-18_15-57-35.csv，表示该文件是 2019 年 9 月 18 日 15 时 57 分 35 秒时，发生报警时记录的文件。该日期信息为控制器内部时钟的时间信息。

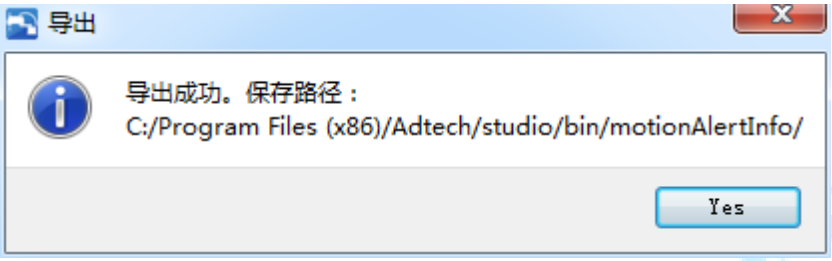


图 7-4 报警导出路径提示窗口

4.7.3 报警信息图形显示

导出报警信息文件后，可以使用示波器以图形化方式查看报警信息。可以点击“示波器”窗口-“数据分析”页面-“文件操作”-“读取报警数据文件”按钮，导入报警信息文件。导入后，报警信息文件中的数据即显示在“数据分析”页面的图形显示窗口中。

图 7-5 报警信息数据图形化显示

4.7.4 报警处理办法

常见的伺服报警及处理办法如下：

1）位置偏差超限

报警原因	处理方式
位置偏差限制值过小	进入“保护参数”，增大允许的最大位置偏差
位置环增益设置不当	重新调节位置环增益
加减速度过大	降低加/减速度
QC410 与电机的 UVW 相接线不正确	检查电机与 QC410 之间动力线连接
QC410 内部走线或电路接线不良	更换 QC410 内部线路

2）速度偏差超限

报警原因	处理方式
速度偏差限制值过小	进入“保护参数”，增大允许的最大速度偏差
速度环增益设置不当	重新调整速度环参数（主要调整增益）
QC410 与电机的 UVW 相接线不正确	检查动力线接线是否有误
QC410 内部编码器线路损坏	更换 QC410 内部线路

3）电机电流过高

报警原因	处理方式
电机电流限制值设置过小	进入“保护参数”，适当增大电机电流限制值
电机电角度未找对	重新执行换向，查找电机电角度
QC410 与电机的 UVW 相接线不正确	检查动力线接线是否有误
加减速度过大	降低加/减速度
负载大，电机功率小，带不动	更换大功率电机（应考虑 QC410 最大功率）

4）主编码器未连接

报警原因	处理方式
电机编码器线连接错误	检查电机编码器与 QC410 编码器线端接线
编码器类型选择不对	进入“电机参数”，重新合理设置编码器类型

5）IPM 模块报警

报警原因	处理方式
QC410 驱控一体机内部编码器线损坏	更换内部编码器线
QC410 内部驱动功率模块过热	停机降温
QC410 内部驱动板异常	检查驱动板内部模块
伺服参数设置不正确	检查伺服参数设置

6）电机堵转

报警原因	处理方式
电机带有刹车，刹车未打开	打开电机刹车
电机堵转保护设置不恰当	进入“保护参数”，更正电机堵转保护条件设置
负载大，电机功率小，带不动	更换大功率电机(应考虑 QC410 最大功率)

7）母线电压过低/过高

报警原因	处理方式
QC410 电源接线或内部供电异常	检查内部接线
QC410 内部驱动板异常	检查驱动板模块

8）换相不成功

报警原因	处理方式
换相模式设置不对	选择正确的换相模式
编码器设置与电机实际编码器类型不符	进入“保护参数”，更正电机编码器类型

	设置
电机极对数设置与实际电机极对数不符	更正电机极对数
换向时间设置不正确	更改换向时间

9）电机使能状态下通讯连接错误

报警原因	处理方式
换向调节设置不正确	在换向调节中，根据编码器类型选择正确的换向模式（增量式编码器选择 1-微动模式；绝对式编码器选择 2-绝对式编码器预置角度）。

10）电池低压报警

报警原因	处理方式
编码器侧电池电压过低	更换电池

11）计算滤波器失败

报警原因	处理方式
滤波器修改之后未执行计算或计算不成功	关闭电机使能，重新执行计算滤波器

12）温度警告

报警原因	处理方式
驱控一体机内环境保护温度设置不正确	进入“保护参数”，更正“允许的驱动器温度最大值”
驱控一体机工作环境温度过高	切断电源降温

13）过载报警

报警原因	处理方式
保护参数中的“持续电流设置不正确”	进入“保护参数”，设置“持续电流”为该轴电机的额定电流

负载大，电机功率小	更换大功率电机(应考虑 QC410 最大功率)

14）相电流过高

报警原因	处理方式
保护参数设置不正确	进入“保护参数”，修改相电流限制
QC410 与电机的 UVW 相接线不正确	检查动力线接线是否有误或松动

注意：若无法解决报警问题，请立即咨询我司.