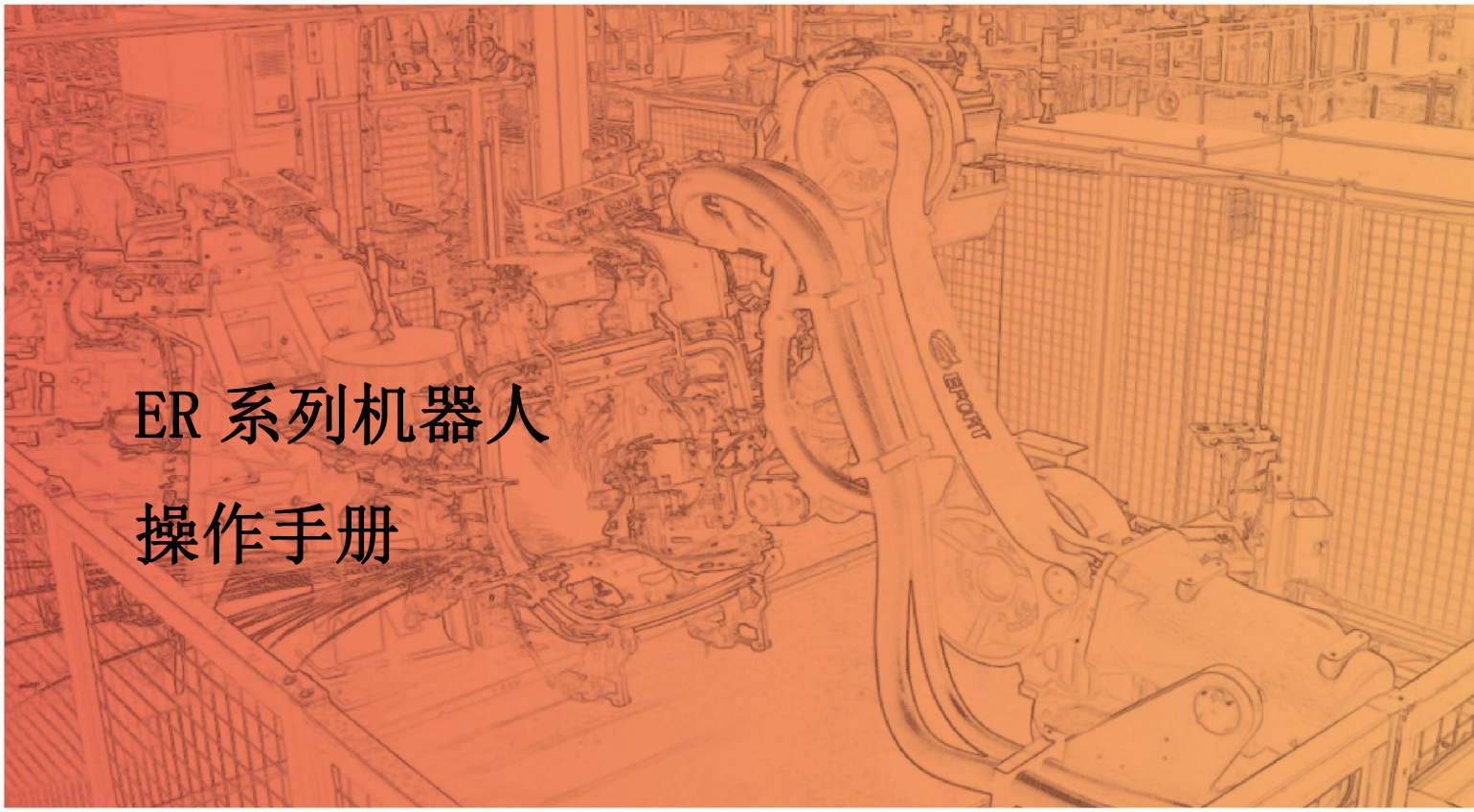
EFORT

智造专家 埃夫特



ER 系列机器人

操作手册

埃夫特智能装备股份有限公司

服务热线（Tel）：400-0528877

声 明

感谢您购买埃夫特机器人产品，为确保已对产品进行正确的设置，请您在使用本产品之前，务必仔细阅读本操作手册。本声明及手册所提及的内容涉及您的人身及财产安全，若不遵循或不按照手册的说明与警告而擅自操作，可能会给您和周围的人带来人身伤害或给埃夫特机器人或周围的其他物品造成财产损失。本声明及手册为截至本批次产品出厂前的最新版本，后续请通过访问 www.efort.com.cn 官方网站以获取更新的信息。

本手册仅作为对产品进行正常操作的指导，在产品使用过程中，埃夫特公司并不对除产品缺陷外的其他原因引发的人身伤害、财产损失承担责任。埃夫特公司郑重建议：参与机器人操作、示教、维护、维修、点检等相关活动的人员，在学习完毕埃夫特公司准备的培训课程前，请勿赋予其对机器人的操作使用权限。

版本号：V 2.8

目 录

概 述

1 关于本手册

本手册介绍了如何使用示教器来操作埃夫特机器人系统。

2 手册使用

本手册应在操作机器人过程中使用。

3 本手册的阅读对象

本手册主要面向：产品开发与测试人员；技术服务人员；操作人员。

4 操作手册阅读指南

本操作手册分为以下几个章节。

章节	标题	内容
1	安 全	安全标准以及安全预防。
2	欢迎使用埃夫特机器人	从硬件方面说明其组成和基本概念。
3	操作界面	操作系统的界面布局、登录。
4	机器人设置	设置机器人的参数以及 IP 等。
5	点动操作	点动操作概念、操作步骤及注意事项。
6	文件管理与编程	文件管理，程序的编辑与调试。
7	坐标系管理	工具坐标系标定及用户坐标系标定。
8	零点恢复	零点恢复概念、操作步骤及零点文件重写。
9	零点标定	文件管理，程序的编辑与调试。
10	碰撞检测	碰撞检测的参数说明、设置步骤、RPL 程序设置碰撞检测参数。
11	安全监控	区域监控和安全位置的设置步骤和激活使用方法。
12	IO 设置	更新 IO 模块、IO 自由配置、远程 IO 模块适配和模拟量 IO 配置操作步骤。
13	附加轴	附加轴轴数配置、参数设置、位置监控、清零及附加轴指令操作。
14	简单码垛	简单码垛系统组成及功能、界面说明及操作说明。
15	监控	介绍机器人监控功能的使用，主要包括位置、IO、驱动器、现场总线部分。
16	固定视觉	固定视觉的 APP 界面介绍、固定视觉的标定及用例。
17	Mot 程序管理	Mot 程序的导入导出、复制剪切粘贴等功能。
18	传送带跟踪	介绍跟踪视觉相关设置及编程操作。
19	冲压	介绍冲压功能界面及各功能使用操作步骤。
20	高级码垛	介绍 EFORT 工业机器人高级码垛文件导入导出、

		高级码垛设置及生产操作。
21	故障处理	控制器和驱动器故障处理，以及程序运行故障处理。
22	负载辨识	需要负载补偿功能以保证碰撞检测功能正常使用
23	弧焊设置及编程	用户通过对弧焊机器人适当设置后，可以按照指令实现弧焊工艺焊接。
24	变位机	变位机系统是机器人系统插补轴之外的辅助轴所组成的附加插补系统，辅助轴系统在与机器人插补轴进行同步插补运动的同时，还能保证机器人末端在辅助轴坐标系统下的精确轨迹。
25	程序预约	通过每个工位上的启动按钮，机器人按照预约的顺序运行各个工位上程序。
26	TCP/IP 通讯	机器人可通过 TCP/IP 和第三方设备进行字符串数据交换和传递，其中机器人即可作为服务器也可作为客户端，可通过示教器界面配置对应的通讯参数以及使用对应的通讯指令进行通讯。
27	工具手对齐	工具手对齐操作手册
28	ALias	ALias 操作手册
29	动力学碰撞检测操作手册	动力学碰撞检测操作手册
30	总线设置	总线相关设置
附录 1	控制器报警及警告	介绍控制器报警的原因和建议操作。
附录 2	驱动器报警及警告	介绍驱动器报警的原因和建议操作。
附录 3	编程指令说明	机器人编程中的指令。

5 操作前提

阅读本手册前，读者应当具备一定的机器人专业知识，或受过相应的机器人操作方面的技术培训。

6 修订历史

版本	日期（年/月/日）	原因（创建/修订）	创建或修订人
V1.1	2019.05.09	创建	谷涛涛
V1.2	2019.06.25	修订	谷涛涛
V1.3	2019.07.19	修订	蒋中慧
V1.4-beta1	2019.12.30	修订，补充	蒋中慧 湛少胜 朱桂林
V1.4-beta2	2020.03.13	修订	朱桂林
V1.4	2020.04.20	修订项目中测试的问题	朱桂林
V2.4	2020.06.06	按新模板修改封面、版本等	朱桂林
V2.5-beta1	2020.06.24	软件更新 V1.9.0_beta6 版本更新说明书	朱桂林、孙建刚、蒋中慧、湛少胜、王连兵
V2.5	2020.07.22	修订 beta1 中测试问题	朱桂林、孙建刚、蒋中慧、湛少胜、魏小敏

V2.7_alpha2	2021.05.12	修订，补充	余晗
V2.7_alpha3	2021.08.21	修订，补充	余晗、朱桂林、孙建刚、魏小敏
V2.8_alpha1	2021.10.31	修订，补充	余晗、朱桂林、张明洪、魏小敏、王连兵、田菲

第 1 章　安全

1.1 安全须知

根据国家和当地的有关法律、法规、条例，在使用包括机器人的工业系统时，安全防范是最基本的关注点。

在使用机器人导致的人身伤害和财产损失的意外中，使用机器人的工厂是负有责任的。因此，除了理解本手册及其相关资料外，必须理解所有有关健康和安全的法规和标准，并请一定遵守。

为了安全，遵守本手册及埃夫特公司其他手册的规定只是最起码的要求。本手册记载的安全相关信息作为一个总则，并没有完全包括机器人应用系统的各方各面。所以，在使用机器人时，应当根据系统及其应用环境的实际情况，采取必要的安全措施，并严格遵守。

操作人员务必认真阅读以下信息，尤其注意本章所列的安全措施部分。

EFORT 工业机器人的用户应负责确保遵守所在国家/地区的适用安全法律和法规，并且用于保护机器人系统操作者的必要安全设备设计合理且安装正确。机器人操作者必须熟悉诸如以下适用文档中描述的工业机器人的操作和处理：

1)《ER 系列机器人安全手册》

2)《ER 系列机器人操作手册》

本手册包含机器人与控制器的产品手册中所含的全部安全说明。机器人系统应设计和制造良好以便在运行、调节和维护期间实现安全进入全部有干预必要的区域。对于有必要在安全保护空间作业的情形，必须保证能安全且充分的进入作业位置。

1.2 安全准则

禁止行为

1. 不要随意改动或拆除工业机器人防护装置和安全装置。

2. 如果发生积涝情况，不要触碰机器人，应先切断所有电源、对场地进行排水。

3. 工业机器人的操作只能由受过充分的培训和指导（包括已经熟读本手册）的专业人员来进行。

4. 务必保证急停设备周围畅通，不可再急停设备前堆放杂物，妨碍紧急情况下设备的使用。

5. 不得对机器人使用不合适的材料、进行不适当的调节和改动。

6. 未经授权人员、或者未接受过机器人使用的培训了解存在的风险的人员不得操作机器人。

7. 以下情况时不得使用机器人：

- **机器人元件暴露。**
- **安全装置被禁用。**
- **保险丝和/或机械设备的全部或者部分被禁用时。**
- **加工材料不符合要求。**
- **同一时间不允许超过一人使用机器。**

8. 严格禁止任何违反上述要求使用机器人的行为，特别是不得随意使用非原装配

	件。 9. 切勿移动安全防护装置，用户有责任确保安全防护装置固定稳当并且有序运行。 10. 只有在维修时才可以移动安全装置，但必须要遵守维修人员的操作程序，在保证机器人安全的情况下进行。
	强制性措施 1. 在启动机器前务必确认没有人在危险区域内。 2. 所有操作人员必须接受专门的工业机器使用和维修培训。 3. 工头要持续监控确保所有程序正常运行，确保安全防护程序应用正确到位。 4. 按照本手册中维护保养中的要求进行维护，保持工业机器人的整洁干净。 5. 要准备合适的工具箱用来归纳清洁工具和维修工具;工作人员必须穿戴所述个人防护设备。 6. 除了这些说明，工作人员还必须遵守现行的健康和安全规范。 7. 机器人出现故障、或疑似损坏、机器不运转或发出异样噪音时应停止机器工作。 8. 一旦贵方发现机器出现火情（无论火情大小），应当立即报警，找专业队伍扑救。 9. 机器的运行状态时控制柜门必须一直关闭不得打开。控制柜钥匙必须由电工保管。 10. 在通电模式下操作时，工作人员不得进入安全防护区域。 11. 在开启自动模式前，所有暂时停用的安全功能必须恢复到正常的工作状态。
	警告 1. 重力和制动装置的释放可能会导致坠落危险。 2. 对安全防护装置进行检查时可能会因安全防护装置无法工作给维修人员保护而造成危险。因此，维修人员必须非常小心，并做好万全的防护措施。

1.3 各工作过程中的安全注意事项

1.3.1 机器人安装和连接的安全

	危险 对于安装连接的所有操作，请严格遵守下列事项，同时参考下列国家/国际标准。机器人遵照工业环境用机器人安全要求（GB11291.1-2011/ISO10218-1:2006）进行安全功能方面的设计。 1. 操作前，请完整阅读和理解所有手册、规格说明和埃夫特公司提供的其他相关文件。另外，完整理解操作、示教、维护等各过程。同时，确认所有的安全措施到位并有效。 2. 运输机器人时，应避免超过指定的高度： ● 只允许具备叉车和起重机操作资格的人，来移动/运输机器人本体、控制柜等等。

	● 在搬运中，决不可靠近或走到提起的机器人本体、控制柜下方。 ● 切勿在搬运中呆在机器人本体、控制柜上面,也决不可触碰或人工支撑它们。 3. 按机器人起吊图示所描述的，将钢丝绳钩住吊环，并在操作前，确认吊环没有松动。 4. 当使用吊带转运控制柜时，请去除示教器及其支架，以免电缆等钩住其他设备。 5. 在搬送机器人前，请移除所有不需要的物体，并清理到安装位置的通道。 6. 如果用叉车搬运，请对控制柜进行固定，防止控制柜倾倒。 7. 由于机器人由精密的元器件组成，请保护机器人免受碰撞、冲击。 8. 当安装地的总电源开启时，切不可连接控制柜的电源电缆。否则将是极端危险并可导致触电。连接输入电源电缆时，请务必确定主电源为关断状态。同时为防止输入电源或断路器被误合上，请在所有的电源单元、断路器上放置清晰的关断标志，表示检查/保养、维修进行中，并用锁锁定或放置夹头夹住主电源开关。 9. 当接线工作完毕时，务必盖上输入电源连接端的盖板。否则将是极端危险的，如果误触到端子可导致触电事故。 10. 请将连接机器人的电机/信号线束放置在电缆槽内，以防止受到损害。另外请采取措施以免它们受压。控制柜与机器人本体之间全部连接完毕之前，请勿连接接入电源。否则则非常危险，可导致触电等事故。

1.3.2 机器人启动前的安全

	危险 机器人开动前的操作，必须严格遵照以下事项，并请参阅相关的国内/国际安全标准。机器人遵照工业环境用机器人安全要求（GB11291.1-2011/ISO10218-1:2006）进行安全功能方面的设计。 1. 操作前，请完整阅读和理解所有手册、规格说明和埃夫特公司提供的其他相关文件。另外，完整理解操作、示教、维护等各过程。同时，确认所有的安全措施到位并有效。 2. 务必把机器人的控制柜、操作面板和所有其他的控制装置安装在安全防护装置(围栏）之外，只有这样才能监视整个机器人的运动范围。 3. 确认在机器人手臂的运动范围内，没有任何人员、包装材料、夹具或其他各类障碍物。 4. 消除固定设备和移动设备之间任何可能夹人的区域。 5. 连接电源电缆前，请确认供电电源的电压、频率、电缆规格等是否符合要求。 6. 确保控制柜和周边设备的正确接地。机器人控制柜的接地线和周边设备的接地线应分开接地，不能连在一起。同时如果外部设备上加电磁开关、接触器等装置时，请在邻近机器人控制柜的电源进线上，安装电源滤波器或相当装置。 7. 在打开机器人的“电源”ON 之前，请确认机器人的安装符合机器人安装的要求。 8. 在操作员操作机器人时，必须配置有一个观察员进行监控，这个观察员也必须完成埃夫特公司对应的培训。 9. 对于应用项目（水、压缩空气、保护气体等)，系统必须配置有监控仪表，以

	便及时自动发现供水供气的不正常情况。 10. 如果在机器人工作过程中会产生大最的废料、金属尘粒、细小粒子等，请在机器人本体、机器人控制柜、周边装置上罩上合适的罩壳。

1.3.3 机器人启动的安全

	危险 要启动机器人，首先连接好电源线，然后将电源开关由 OFF 旋转至 ON。这些操作，请严格遵守如下事项，同时参考相关的国内/国际的标准。 机器人遵照工业环境用机器人安全要求（GB11291.1-2011/ISO10218-1:2006）进行安全功能方面的设计。 开动机器人前，请确认急停止开关工作正常。 1. 操作前.请完整阅读和理解所有手册、规格说明和埃夫特公司提供的其他相关文件。另外，完整理解操作、示教、维护等各过程。同时，确认所有的安全措施到位并有效。 2. 检查所有机器人操作必须的开关、显示以及信号的名称及其功能。 3. 除非机器人电源断开，否则不可进入安全围栏。同时，在开动机器人前确认各安全防护装置功能正常。 4. 如果机器人应用系统中有几个操作人员一起工作，务必让全部操作者及其相关人员都清楚机器人已激活后，才可以启动机器人。 5. 在接通电机电源 ON、开始示教或自动操作前，请再次确认在机器人安全栅栏内和机器人周围没有任何工人员或遗留的障碍物存在。 6. 当启动机器人和从故障状态恢复运行时，在开启控制柜电源后，请把你的手放在紧急停止开关上,以便在出现异常情况时，可以立即切断马达电源。 7. 在激活机器人前，请再次确认下列条件已满足。 ● 确认机器人的安装状态是正确的和稳定的。 ● 确认机器人控制柜的各种连接都是正确的，电源规格（电源电压、频率等）符合要求。 ● 确认各种应用连接（水、压缩空气、保护气体等）是正确的，并和规格型号是一致的。 ● 确认与周边装置的连接是正确的。 ● 请确认在使用软件运动限位外，也已安装了机械限位挡块/或限位开关来限定机器人的运动范围。 ● 当机器人被机械限位挡块停止时，请确认检查了相关零件或已更换了失效的机械限位挡块（如果有必要）。 ● 确认采取了安全措施：已安装了安全围栏或报警装置及联锁信号等安装防护装置。 ● 请确认安全防护装置及联锁的功能正常。 ● 确认环境条件（温度、湿度、光、噪声、灰尘等）都满足要求，或者说没有超过系统和机器人的规格要求。

1.3.4 试车安全

危险

试车时，示教程序、夹具、逻辑控制器等各种要素中可能存在设计错误、示教错误、工作错误。因此，进行试车作业时必须进一步提高安全意识。

试车过程中需要注意以下几点：

1. 首先，确认紧急停止按钮、保持/运行开关等用于停止机器人的按钮、开关、信号的动作是否正常。一旦发生危险情况，若无法停止机器人将无法阻止事故的发生。

2. 机器人试车时，首先将机器人的操作速度设定为低速（5%～10%左右的速度），对示教的动作进行确认。以 2～3 周期左右，反复进行动作的确认，若发现有问题时，应立即停止机器人并进行修正。确保没有问题之后，逐渐提高速度（50%→70%→100%），各以 2～3 周期左右，再次反复作确认动作。

1.3.5 示教过程中的安全

危险

埃夫特公司建议应在安全围栏外完成示教工作。但如果确实需要进入安全栅栏，请严格遵守下面事项，同时参考下面国内/国际安全标准。

机器人遵照工业环境用机器人安全要求（GB11291.1-2011/ISO10218-1:2006）进行安全功能方面的设计。

示教工作前，请确认紧急停止开关功能正常。

1. 操作前，请完整阅读和理解所有手册、规格说明和埃夫特公司提供的其他相关文件。另外，完整理解操作、示教、维护等各过程。同时，确认所有的安全措施到位并有效。

2. 开动机器人前，请确认所有的安全防护装置 (安全围栏）工作正常。

3. 示教工作应由两个人来做—个示教员、一个观察员。观察员同时也承担安全监督的责任；并在示教前，确认“工作启动”等信号情况。

4. 示教员在进入安全围栏前，必须把示教器上的示教开关打到手动位置，以防控制柜模式开关打到自动模式而引发事故。一旦机器人做出任何不正常的运动，立即按下紧急停止开关，并立即从预设的撤退路径退出机器人工作区。

5. 在安全围栏外、可监控整个机器人运动的位置上，请为观察员安装一个急停开关。一旦机器人出现不正确的运动，观察员必须可以非常方便地按下开关来立即停止机器人。另外，如果需在紧急停止后重新启动机器人，请在安全围栏外进行复位和重启手动操作。示教员和观察员必须是经过特别培训的合格人员。

6. 请清楚地标示示教工作正在进行中，以免有人通过控制柜、操作面板、示教器等误操作任何机器人系统装置。

7. 完成示教工作后，在确认示教的运动轨迹和示教数据前，请清除安全围栏内、机器人周围的全部人员和障碍遗留物，确认安全围栏内没有任何人员和障碍遗留物后，请在安全围栏外执行确认工作。这时，机器人的速度应小于等于安全速度（250mm/s），直到运动确认正常。

	8. 如需在紧急停止后重启机器人，请在安全围栏外手动复位和重启。同时确认所有的安全条件，确认机器人周围、安全围栏内没有任何人员和障碍遗留物。 **9.** 示教过程中，请确认机器人的运动范围，禁止接近器人手臂的下方。防止因意外操作产生的危险，特别注意，当机器人手爪中抓有工件时，禁止接近机器人手臂，防止因工件意外掉落而产生的危险。 **10.** 为了安全，在示教或检查模式中，机器人的最大速度被限制在了 **250mm/s** 之内（安全操作速度）。但是，在刚完成示教或出错恢复后，操作员校验示教数据时，请把检查运行的速度设得越低越好。 **11.** 示教过程中，无论示教操作员还是监督员，必须时刻监视机器人有无异常运动、机器人及其周围可能的碰装、挤压点。同时，请确认示教操作员的安全通道，以供在紧急时撤退之用。 **12.** 在机器人的运动示教完毕后，请把机器人的软件限位设定在机器人示教运动范围之外一点点的地方。如何设定软件限位，请参阅埃夫特工业机器人操作手册。

1.3.6 自动运行时的安全

危险

由于示教的程序将高速重现运行，所以请严格遵守如下事项，同时参阅相关国际国内安全标准。

机器人遵照工业环境用机器人安全要求（**GB11291.1-2011/ISO10218-1:2006**）进行安全功能方面的设计。

在自动操作前，请确认所有的开关功能正常。

1. 操作前，请完整阅读和理解埃夫特公司提供的所有手册及其他相关文件。另外，完整理解操作、示教、维护等各过程。同时，确认所有的安全措施到位并有效。

2. 在自动运行中，永远不要进入或部分身体进入安全围栏。同时，请在启动运行机器人前，确认安全围栏内没有任何人员或障碍遗留物。

3. 自动运行中，机器人在等待定时器延时或外部信号输入时，看上去像停止了一样。但这时千万不要靠近机器人，因为当定时器时间到或外部信号输入时，机器人将立即恢复运行。

4. 在自动运行中，这种情况将是极端危险的：如果工件的抓握力不够，在机器人运动中，工件有可能会被甩脱。请务必确认工件已被牢间地抓紧。当工件是通过气动手爪、电磁方法机构等抓握的，请采用失效安全系统，来确保一旦机构的驱动力被突然断开时，工件不被弹出。即使在出错时，工件出的可能性为最小时，也请安装保护栅，如网罩等。

5. 在安全围栏上显示“自动运行中”标志，并且不得进入工作区域。同时，请确认安全通道，以便操作人员在紧急情况下撤出。

6. 如果存故障导致机器人在自动运行中停止，请检查显示的故障信息，按照正确的故障恢复顺序，来恢复和重启机器人。

7. 请在故障恢复顺序后、重新启动机器人前，确认安全的工作条件满足，并且确认在安全防护装置内或机器人周围没有遗留任何人员、夹具、周边装置或障碍物等。

1.3.7 维修时的安全

危险

要进行维修时，请严格遵守下列条款，同时参阅相关国际国内安全标准。

机器人遵照工业环境用机器人安全要求（GB11291.1-2011/ISO10218-1:2006）进行安全功能方面的设计。

在维修前，请确认所有开关功能正常。

1. 操作前，请完整阅读和理解埃夫特公司提供的所有手册及其他相关文件。另外，完整理解操作、示教、维护等各过程。同时，确认所有的安全措施到位并有效。

2. 在进入安全围栏前，请确认所有必须的安全措施都已准备好并且功能良好。

3. 在进入安全围栏前，请切断控制电源一直到总电源。并放置清晰的信号显示关断、维修进行中，并且采用锁定或夹定主电源开关，以免有人误开电源。

4. 维修工作仅限于完成了相应型号机器人的特别培训的人员。

5. 在维修工作前，确认机器人周围具备足够的空间，以免与周边设备干涉。同时将周边装置于固定状态，防止它们出现任何的突然动作。

6. 在进入安全围栏前，请务必关断自动操作功能。如果机器人出现任何的异常运动，应立即按急停开关，并立即从规定的撤离路线撤出。

7. 除操作人员手中示教器的的紧急停止开关之外，请在安全栏外、便于观察全部机器人运动范围的地方，为监察员安装另外一急停开关。一旦在维修中机器人出现异常动作，此开关必须可以让监察员非常容易地按到。在急停后，请从围栏外面来复位并重启机器人。此外，操作者和监察员都必须是完成了特别培训课程的人员。

8. 操作中，操作者和监察员都必须时刻注意观察异常运动、可能的碰撞点及机器人周围。

9. 更换时，请只使用埃夫特提供的零部件。

10. 在拆除任何关节轴的伺服电机前，请用合适的提升装置支撑好机器人手臂。拆除电机，将使该轴的刹车机构失效，如果没有可靠的支撑，手臂将会下坠。请注意，如果按控制柜上的任何轴抱闸释放开关，会出现相同的危险。

11. 当需要更换驱动模块、电源模块，请关断控制电源，并且至少等待 7 分钟。然后，请在确认电源的输出电压为 0V 后，才开始更换工作、拆除连接器等。也请注意，不要触碰任何零件，防止触电或烫伤。

12. 如果供有压缩空气或水时，维修前，请切断供应源、并清除管线内的任何剩余压力。

13. 当机器人扩展附加轴时务必确认附加轴的急停信号要串接到控制柜的急停链路中。

14. 当变更机器人部件时一定要确认该部件和原部件的匹配程度，并仔细核对原理图，防止误接线造成机器人控制柜元器件或者外部元器件损坏。

1.3.8 点检和维护时的安全

危险

为防止系统故障，请严格按照下列的条款进行机器人的清洗、检查、维护或更换部件。同时参阅相关国际国内安全标准。

在检查与维护前，请确认所有的急停开关功能正常。

1. 操作前，请完整阅读和理解埃夫特公司提供的所有手册及其他相关文件。另外，完整理解操作、示教、维护等各过程。同时，确认所有的安全措施到位并有效。

2. 在检查与维护工作前，清除不要的物体，并清理到安装位置的通道。

3. 点检和维护保养工作，只限于完成了本机器人或相同型号机器人特别培训的人员。

4. 进行点检和维护保养工作前，请确认机器人周围足够的空间，以避免与周边设备发生干涉。同时把周边设备设成固定状态，确保它们不会突然运动。

5. 在进入安全围栏前，请按工作需要切断整条线的电源或机器人电源，并请切断电源一直到总电源。并放置清晰的信号显示关断、检查/维修进行中，并且采用锁锁定或夹夹定主电源开关，以免有人误开电源。如果整条线不能停止来，请在目标机器人与任何相邻机器人之间安装临时安全围栏。

6. 当进行联锁信号线路的点检和维护工作时，请无误地关闭所有信号关联设备的电源，以确保安全。在进行此项工作期间，不得进入安全围栏。

7. 在完成点检和维护工作后,请确认安全防护装置（安全栅栏、安全插销、急停止开关等)、周边设备、联锁线路等安全装置的工作正常。

8. 除操作者持存的紧急停止开关之外，请为安全护栏外的监督员安装另一个急停开关，安装位置请选在可以监控全部机器人运动范围的地方。如果在维护/点检中，机器人出现不正常的运动，监督员必须很容易地按到开关。急停后，恢复和重启机器人必须在安全围栏外进行。另外，操作员和监督员必须是完成了特别培训课程的人员。

9. 示教员在进入安全栅栏前，必须把示教器上的示教模式开关打到手动模式，以防控制柜模式开关打到自动模式而引发事故。一旦机器人做出任何不正常的运动，立即按下紧急停止开关，并立即从预设的撤退路径退出机器人工作区。

10. 点检/维护过程中，无论操作员还是监督员，必须时刻监视机器人有无异常运动、机器人及其周围可能的碰撞、挤压等等。同时，请确认操作员的安全通道，以供紧急撤离之用。

11. 如果在点检/维护过程中，不可避免地需要拆除安全围栏，请提供足够的安全措施：

- 把机器人和周边设备停在合适的地方。
- 锁定/标定电源和开关，必须避免任何人误开电源或误把开关打到自动模式。
- 完成点检/维护后，重新装好安全围栏，并确认所有的安全措施、安全功能和原来的一样。

12. 请只使用埃夫特公司认可的零件来替换。并且，在点检/维护中，请一定用示教模式、并以尽可能低的速度运动机器人。

13. 当需要更换驱动模块、电源模块，请关断控制电源，并且至少等待 7 分钟。然后，请在确认电源的输出电压为 0V 后。在确认直流电源输出电正变为 0V 后，再开始更换或拔出连接器等工作。另外，如果机器人刚停止运行，散热片或再生吸收电阻可能还是烫的。因此，小心不要触摸任何热的部件。

	14. 在从转轴上拆除伺服电机前，请用合适的提升装置，牢固支撑住机器人的手臂。拆除转轴外的电机将使该轴的刹车系统失效，手臂将会掉落。另外，按控制面板上的任何刹车释放按钮，也会导致同样的危险。 **15.** 如果在维修前后，机器人必须保持同样的姿态，请在更换部件前，记录机器人的姿态数据。 **16.** 在更换过程开始阶段，当拆除印刷线路板或电缆时，检查并记录他们的位置、连接器编号、安装方式、设置数据等，这样就可以按原样恢复了。连接器在插入完毕后，必须把它的锁紧机构牢靠地锁定。另外永远不要触摸连接器的插针。 **17.** 当应用装置(水、压缩空气、保护气体等）使用时，在进行点检/维护前，请关闭它们的供应源，清除管路中的剩余压力。 **18.** 检修/维护后，请确认全部的安全防护装宜功能正常. **19.** 未经公司许可，不要改变或改装机器人。如果发生未经许可的改装，埃夫特公司将不负任何责任。 **20.** 在机器人手臂和控制柜中，内置有多种数据后备电池。如果使用错误的电池，将会引起燃烧、过热、爆炸、腐蚀、漏液等情况发生。因此必须严格遵照下列要求。 ● 只使用埃夫特公司指定的电池； ● 不可再充电、拆开、变换和加热电池； ● 不可把电池丢弃在水中或火中； ● 表面损坏的电池，其内部可能已经短路，决不能再使用； ● 不可用金属，如电线等，短路电池的正负极。不可将废旧电池丢弃在焚化、填埋、倾倒到地面的垃圾中。丢弃电池时，请把它们用袋子包起来，以免它们接触其他金属，同时请遵照当地的规定规章正确处理。 **21.** 当机器人扩展附加轴时务必确认附加轴的急停信号要串接到控制柜的急停链路中。接入扩展轴后需要对急停链路的安全功能进行测试，确保符合安全控制逻辑。变更与安全相关部件后需对急停链路的安全功能进行测试，确保符合安全控制逻辑。 **22.** 变更机器人部件时一定要确认该部件和原部件的匹配程度，并仔细核对原理图，防止误接线造成机器人控制柜元器件或者外部元器件损坏。

第 2 章　欢迎使用埃夫特机器人

2.1 本章简介

本章将介绍机器人控制系统的两个重要部分：控制柜和示教器。主要从硬件方面说明其组成和基本概念。

2.2 示教器

2.2.1 关于示教器

示教器（如图 2-1）是操作者与机器人交互的设备，使用示教器操作者可以完成控制机器人的所有功能。比如手动控制机器人运动、编程控制机器人运动、设置 IO 交互信号等等。

图 2-1 EFORT 示教器

表 2-1 示教器基本参数

序号	项目	技术参数
1	显示器尺寸	TFT 8-inch LCD
2	显示器分辨率	1024*768
3	是否触摸	是
4	功能按键	急停按钮、模式选择钥匙开关，分别为：自动（Auto）、手动慢速（T1）、手动全速（T2），28 个薄膜按键

5	模式旋钮	三段式模式旋钮
6	外接 USB	一个 USB 2.0 接口
7	电源	DV24V
8	防尘防水等级	IP65
9	工作环境	环境温度-20℃~70℃

2.2.2 功能区与接口

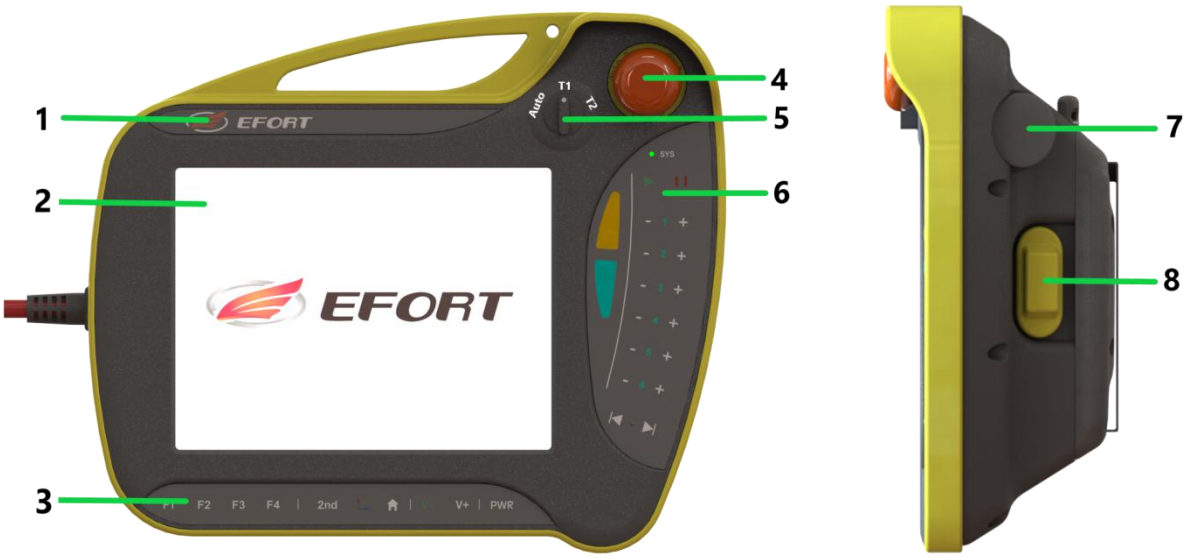


图 2-2 EFORT 示教器

表 2-2 示教器各部分功能

序号	名称	描述
1	薄膜面板 3	公司 LOGO 彩绘
2	液晶显示屏	用于人机交互，操作机器人
3	薄膜面板 2	含有 10 颗按键
4	急停开关	双回路急停开关
5	模式旋钮	三段式模式旋钮
6	薄膜面板 1	含有 18 颗按键和 1 颗红黄绿三色 LED
7	USB 接口	USB2.0，用于导入与导出文件及更新示教器
8	三段手压开关	手动模式下，按下手压开关伺服

示教器功能按键说明如表 2-3 和表 2-4。

表 2-3 示教器上硬件及其功能

序号	名称	序号	名称
1	三色灯	11	轴 4 运动+
2	开始	12	轴 5 运动-
3	暂停	13	轴 5 运动+
4	轴 1 运动-	14	轴 6 运动-
5	轴 1 运动+	15	轴 6 运动+

6	轴 2 运动-	16	单步后退
7	轴 2 运动+	17	单步前进
8	轴 3 运动-	18	热键 1：慢速开关
9	轴 3 运动+	19	热键 2：步进长度开关
10	轴 4 运动-		

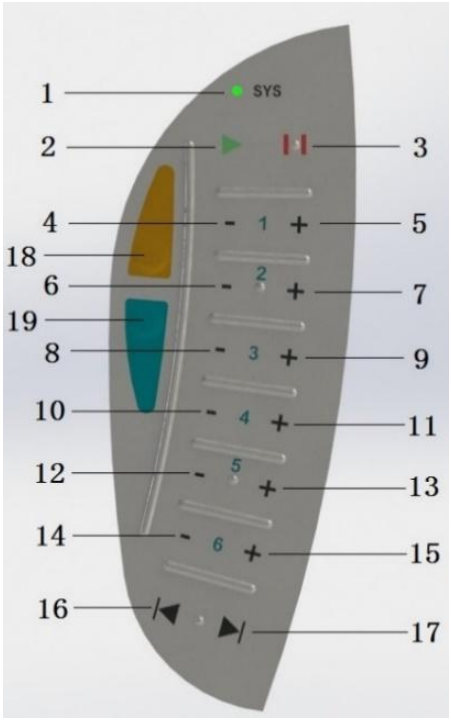


图 2-3 右侧按键

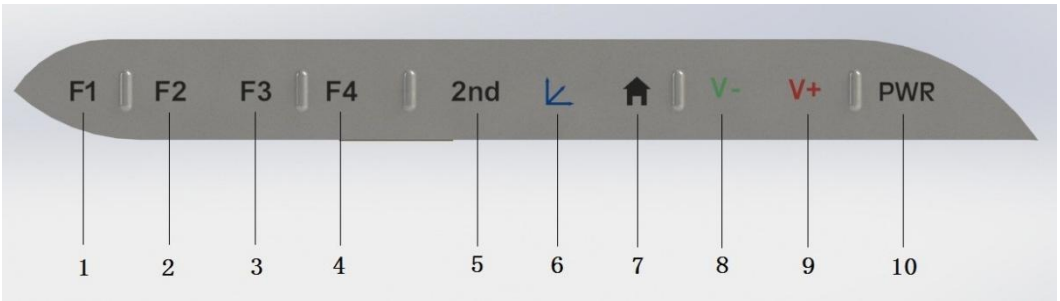


图 2-4 下侧按键

表 2-4 示教器下侧按键及其功能

序号	名称	序号	名称
1	多功能键 F1， 调出/隐藏当前报警内容	6	坐标系切换
2	多功能键 F2， 双击截图	7	回主页
3	多功能键 F3， 程序运行方式（连续、单步进	8	速度-

	入、单步跳过等）		
4	多功能键 F4	9	速度+
5	翻页	10	伺服上电

2.2.3 如何握持示教器

左手握持示教器，点动机器人时，左手指需要按下手压开关，使得机器人处于伺服开的状态。具体方法如下图所示。

图 2-5 示教器握持方法

2.3 更新系统

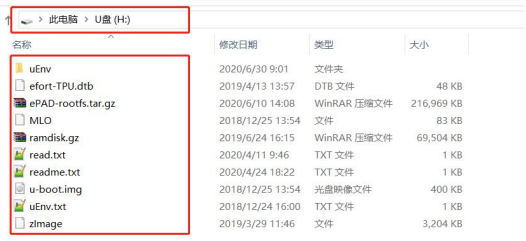
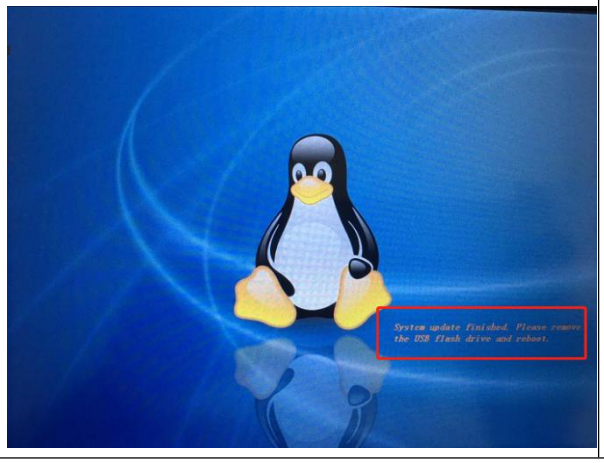
EFORT 工业机器人示教器系统的更新包括两个部分，第一个部分为固件系统升级，第二个部分为示教器 C30 软件系统升级，两个部分的升级相互独立。

2.3.1 固件升级

查看当前示教器的固件版本号并与售后技术人员联系，如果版本号不是已发布的最新版本号，建议升级到最新版固件。

表 2-5 固件升级操作

步骤	图示	说明

1.进入左上角 logo 图进入桌面，点击“**关于**”上方的图标进入 APP 中。	 	查看软件版本信息。其中 R*表示当前机器人控制器硬件版本，R1 表示使用 RP1 控制器，R2 表示使用 RP2 控制器； T1 表示当前示教器为 EFORT 示教器，T2 表示华图示教器； E*表示当前示教器的固件系统版本。 R*T*E*_右侧剩余部分为当前机型的软件系统版本。 示例图片解释：当前 EFORT 示教器使用 RP1 版本的控制器，当前固件系统的版本信息为 E2，当前软件系统的版本信息为 R2T1E_V2.1.0_20210927（编号 1），可以在自定义用户机型名（编号 2），修改好机型名后点击保存重启控制器即可生效（编号 3）。
2.如果发现当前示教器内固件系统不是最新的，且需要更新固件，则按如下步骤操作。		准备一个空的 U 盘，其文件系统格式为 FAT32，将售后技术人员提供的最新版固件文件复制到 U 盘根目录下，固件中包含左图中 10 个文件。
3.升级固件。		将包含固件文件的 U 盘插入 EFORT 示教器上，重启电柜上电源开关等待示教器自动升级固件。 请注意：升级时间较长，请耐心等待，切记不要在升级过程中拔下 U 盘，直到示教器上显示如左图片中的提示，此时示教器系统固件升级完成，

		请拔出 U 盘，再次重启电柜上电源开关等待示教器重新启动进行屏幕校准。

2.3.2 屏幕校准

屏幕校准功能用于如下几种情形下：

1) 示教器固件升级完成并重启后需要进行屏幕校准。

2) 进行精确校准，但是校准功能显示失败，需再次进行一轮屏幕校准。

3) 在示教器界面点击时有偏差或错位，或校准有明显误差显示校准成功，需手动进入屏幕校准功能进行校准。

表 2-6 屏幕校准操作

步骤	图示	说明
1.进入屏幕校准界面。		如果是升级完固件后示教器重启，则自动进入屏幕校准功能界面。 如果是需要手动进入屏幕校准界面，则按如下步骤操作：首先重启示教器，待示教器界面出现滚动条，且滚动条进度（绿色部分）过半时，多次按下示教器底部最左侧**“F1”**按键，可以进入屏幕校准功能界面。
2.进行屏幕校准。		出现屏幕校准界面后，请用示教器自带的触屏笔尽可能点击左图中红圈备注的图标中央位置，总共需校准五个点，每个点点击一次。直至界面上显示“Calibration OK!”表示屏幕校准完成。

3.屏幕校准完成。		完成屏幕校准后，示教器自动重启进入示教器C30软件系统。若校准失败或校准效果不理想，则自动从最开始的位置重新进行校准。 请注意：如果升级完固件后自动进入的平面校准界面，则还需要进行示教器C30软件系统更新。

2.3.3 C30 软件系统升级

示教器的C30软件系统会不定期更新版本，如果您需要更新该系统版本，请联系售后技术人员提供软件版本更新文件，同时需要准备一个空U盘，其文件系统格式为FAT32，将更新文件拷入到U盘的根目录中。示教器固件升级文件和示教器软件系统升级文件不能放在一个U盘内，如果同时升级建议准备两个空U盘。

表 2-7 C30 软件系统升级

步骤	图示	说明
1.准备升级工具。		将售后技术人员提供的软件系统升级文件夹（rtm_update）复制到U盘根目录下，文件夹内包含左图中3个文件。
2.升级软件系统。		将包含更新文件的U盘插入到EFORT示教器上，重启电柜上电源开关，出现左图后等待示教器自动升级软件系统。 请注意：要求系统固件版本为E1及以上。系统版本请查看示教器关于APP界面，如果更新后软件系统不能启动或更新失败，请联系售后技术人员。
3.升级软件系统。		直至示教器界面显示“App update completed!”示教器C30软件系统更新完成，示教器进入系统，拔出U盘，更新完成。

2.4 启动系统

2.4.1 本体检查

1) 检查机器人本体是否固定到位。

2) 检查打包运输时的固定夹具和橡胶垫是否拆除。

2.4.2 系统连接

1) 连接本体到控制柜动力线电缆。

2) 连接本体到控制柜编码器电缆。

3) 连接本体到控制柜抱闸型电缆。

4) 连接示教器到控制柜上。

5) 连接控制柜电源到外部电源。

6) 接通控制装置的电源之前，将地线连接至机构部和控制部。

2.4.3 系统上电

完成上述操作后，使用控制柜上的电源开关启动系统，如果一切正常，从示教器上可以看到系统自动进入登录界面，用户可以根据不同的权限操作机器人。如果有报错提示，请参照故障处理章节处理。

第 3 章　操作界面

3.1 本章简介

本章主要介绍 EFORT 工业机器人示教器的界面布局、登录和设置。

3.2 界面布局

EFORT 工业机器人示教器的界面布局分为状态栏、任务栏和显示区 3 个部分。

图 3-1 界面布局

3.2.1 状态栏

状态栏显示了机器人工作状态，其中程序循环方式、机器人运动坐标系、机器人运行速度可以手动点击图标进行选择。

图 3-2 状态栏

表 3-1 状态栏图标介绍

序号	描述
1	桌面按键，点击图标进入桌面界面。

2	机型显示，双击截图，长按 2s，导出截图功能。
3	状态显示按键，点击进入报警日志界面。状态分为： 正常，图标绿色，机器人正常状态； 错误，图标红色，机器人存在报警； 未连接，图标红色，示教器和控制器未连接。
4	急停信号状态，图标绿色正常；图标红色表示急停被按下。
5	伺服状态，图标白色伺服关；图标绿色伺服开。
6	程序运行模式，图标白色表示 RPL 未运行；图标绿色表示 RPL 运行中。
7	程序循环方式，有以下方式： 连续：程序连续运行； 单步跳过：单步执行一条指令，如果当前指令为调用子程序，子程序直接执行完成； 单步进入：单步执行一条指令，如果当前指令为调用子程序，进入子程序，单步执行子程序的指令； 运动跳过：单步执行运动指令，遇到非运动指令直接执行完成，到下一条运动指令暂停，如果指令为调用子程序，则子程序直接执行完毕，到下一条运动指令暂停； 运动进入：单步执行运动指令，遇到非运动指令直接执行完成，到下一条运动指令暂停，如果指令为调用子程序，进入子程序，子程序中运动指令单步执行。
8	机器人运行模式，分为：自动（Auto）；手动慢速（T1）；手动全速（T2）。
9	机器人运动坐标系，分为：关节；机器人；工具；用户。
10	当前工具坐标系。
11	当前用户坐标系。
12	机器人运行速度。

3.2.2 任务栏

任务栏中显示的是已打开的 App 界面快捷按键。其中，登录、文件、程序和监控是默认一直显示的，其余显示的是在桌面中打开的各 APP。

图 3-3 任务栏

3.2.3 桌面

EFORT 工业机器人的设置和功能 App 都放置在桌面上，点击 App 图标进入相应的 App 界面（因不同型号的机器人功能不同，开放的桌面 App 不同，故界面 App 存在差异，使用方法相同）。

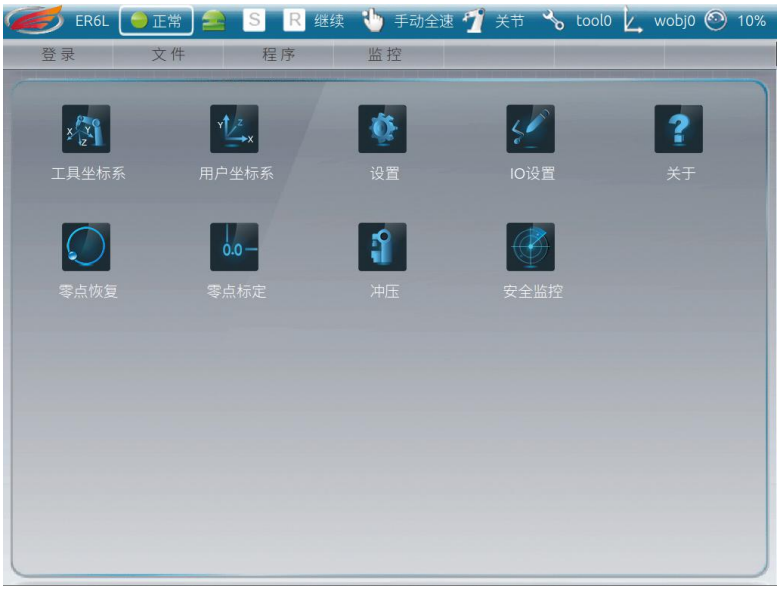


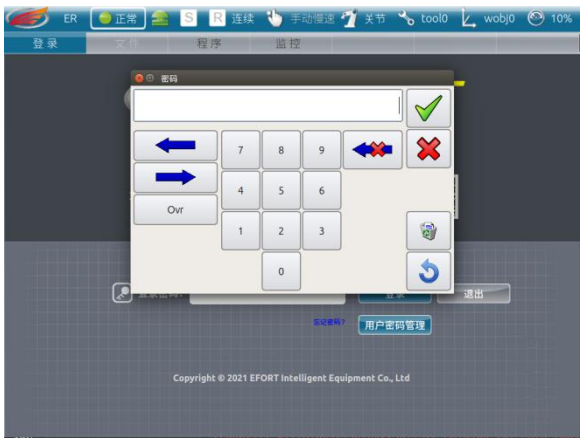
图 3-4 桌面

3.3 登录

EFORT 工业机器人提供操作员、工程师、管理员三个权限等级的账号，默认登录账号为操作者。可以进行账号切换：在密码弹窗中输入密码，点击**“登录”**按钮，即可登录相应账号。

表 3-2 登录操作

步骤	图示	说明
1.进入登录页面，点击显示区的“**输入框**”。		若不在登录页面，点击任务栏中的登录按键进入登录页面。

2．输入正确密码，然后点击“√”确认。		
3.点击“**登录**”，登录成功。		账号已由操作者切换为管理员。 版权信息：版权为埃夫特智能装备股份有限公司所有。

操作权限划分：

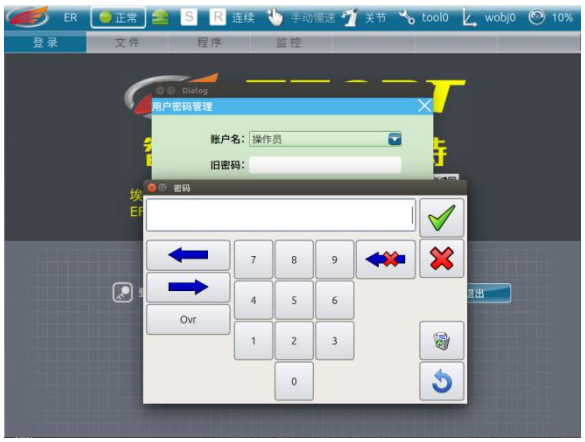
	操作者	工程师	管理员
登录	√	√	√
监控	√	√	√
程序	×	√	√
文件	×	×	√
密码	----	666666	999999

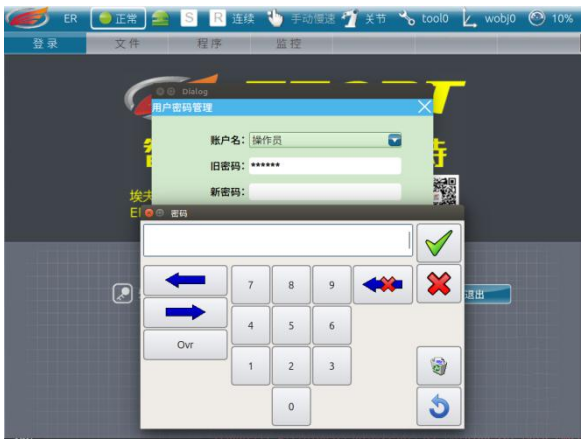
3.4 修改用户密码

Efort 示教器账户的登录的方式为：输入账户密码自动登录对应的账户！因此不允许账户的密码出现重复！Efort 示教器的账户密码要求是：4-6 位的数字。

表 3-3 修改账户密码操作

步骤	图示	说明

1.进入登录页面，点击显示区的“**用户密码管理**”按钮。		若不在登录页面，点击任务栏中的登录按键进入登录页面。
2．点击账户名下拉框，选择要修改密码的账户。		
3.点击“旧密码”输入框，在弹出键盘中输入旧密码。		

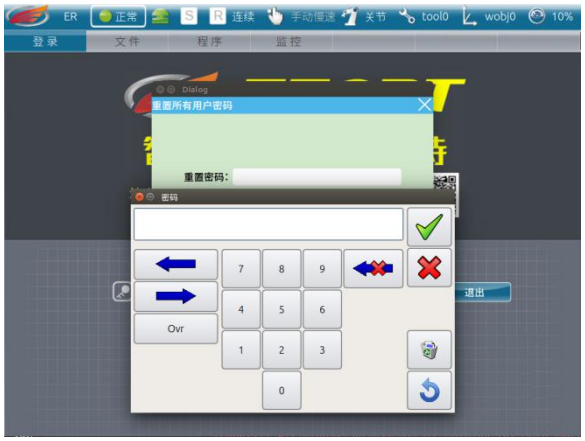
4.点击“新密码”输入框，通过弹出键盘输入新的密码；然后，在“重复新密码”输入框中重复输入新密码。		
5.点击“保存”按钮完成密码修改。		

3.5 忘记密码操作

Efort 示教器允许用户输入统一口令对示教器中的账号密码进行统一的重置。

表 3-4 重置所有账号密码操作

步骤	图示	说明
1.进入登录页面，点击显示区蓝色字体的“忘记密码”字样。		若不在登录页面，点击任务栏中的登录按键进入登录页面。

2．点击重置密码输入框，输入统一重置口令：688165		
3.点击“保存”按钮，完成对当前所用账户密码的重置操作。		

第 4 章　机器人设置

4.1 本章简介

设置 App 界面用于设置系统、轴参数、笛卡尔参数、DH 参数、零位数据、切换 Logo、应用选择、屏幕设置等。其中系统设置包括语言设置和 IP 设置。不同型号机器人功能不同，开放的设置项目可能不同，实际以示教器显示为准。

系统参数中设置修改后，可能导致机器人不能正常工作，所以在需要修改之前，需要进行解锁操作，否则只能查看相关参数。

解锁操作：点击左上角“**解锁**”按钮，如下图所示，然后输入密码后进行相关操作（默认密码：1975），退出 App 或者离开 App 界面 60s 后自动锁定。

图 29-1 设置 app 解锁界面

4.2 系统

4.2.1 语言设置

系统设置上半区为语言设置。语言设置用于切换界面显示语言。目前提供汉语、英语和意大利语三种语言，点击显示的国旗图标切换到对应国家的语言。

注：意大利语设置功能暂未开放。

图 29-2 语言设置界面

4.2.2 IP 设置

控制器上有多个网口，主要功能如下：

ETH1：主要用于 Ethercat 通信

ETH2：用于 Ethercat 通信（当已经使用 ETH1 时，该方式为环网）；也可做以太网通信，比如 ModbusTcp，Ethernet/IP，Tcp/IP， BCC 通信等等

ETH3(A, B)：主要用于以太网通信，比如 ModbusTcp，Ethernet/IP，Tcp/IP， BCC 通信等等

ETH4，ETH5：用于 Profinet 通信。

WAN：路由功能对外接口，可用于配置局域网，广域网等等。

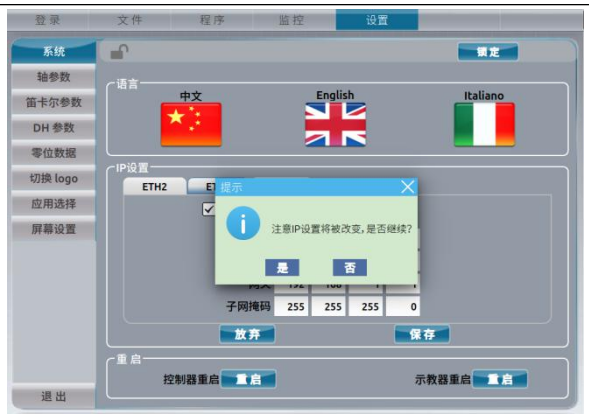
IP 设置分为两个部分，分别为 ETH 网口的设置和 WAN 功能的设置，其他功能暂不支持修改。

4.2.2.1 ETH 网口设置

IP 设置界面用于设置控制器 ETH 网口的 IP 地址、示教器的 IP 地址、子网掩码和网关，对应 ETH2 和 ETH3。

表 29-1 IP 设置操作流程

步骤	图示	说明

1．在系统设置界面，进行 IP 设置。		
2.点击 IP 每段数字所在编辑框后，旁边会弹出数字键盘，输入完成后点击对号。		
3．IP 设置全部完成后确定无误点击“**保存**”。		若输入 IP 后仍想保留原 IP 设置，则点击“**放弃**”恢复原 IP 设置。
4．点击保存后在提示是否继续信息框上点击“**是**”，重启机器人。		IP 地址改变后需要重启机器人才能生效。

4.2.2.2 WAN 口设置

控制器使用 ETH3 口或者 ETH2 口通信时，在一些复杂组网情况下，会出现网络阻塞情况，从而引起控制器与示教器失联或者其他通信报警。在此情况下，需要使用路由器进行网络隔离，机器人控制器自带路由功能，对外端口为 wan 口。

表 29-2 WAN 设置操作流程

步骤	图示	说明
1. 在 WAN 界面设置映射端口。		使用 WAN 口功能，与外部设备通信时，需要通过 LAN 口进行数据转发，可以与 ETH3 口通信，即选择其映射关系。选择后，LAN 口 IP 以及默认网关必须与 ETH3 口的网关一致。
2. 设置 LAN 参数。		LAN 口网关配置根据实际需求配置，不配置可以为 0.0.0.0。若配置网关，网关参数需要根据子网掩码范围确定。
3. 设置 WAN 参数。		不和 LAN 口同一个网段即可。

4. 设置 NAT，完成路由表设置，使外部端口和内部端口进行映射，实现通信链路。		设置 NAT 时，需要先选择协议，然后可以填写名称和端口号，未选择协议则不可以设置。 协 议： tcp 、 udp 、 tcpudp。
5. IP 设置全部完成后确定无误点击“**保存**”。		注意在 WAN 的相关设置发生修改后，需要断电重启才能生效。

如下图所示，为 WAN 口通讯示意图，其映射网口为 ETH3。其 NAT 路由表如表 29-2 所示。

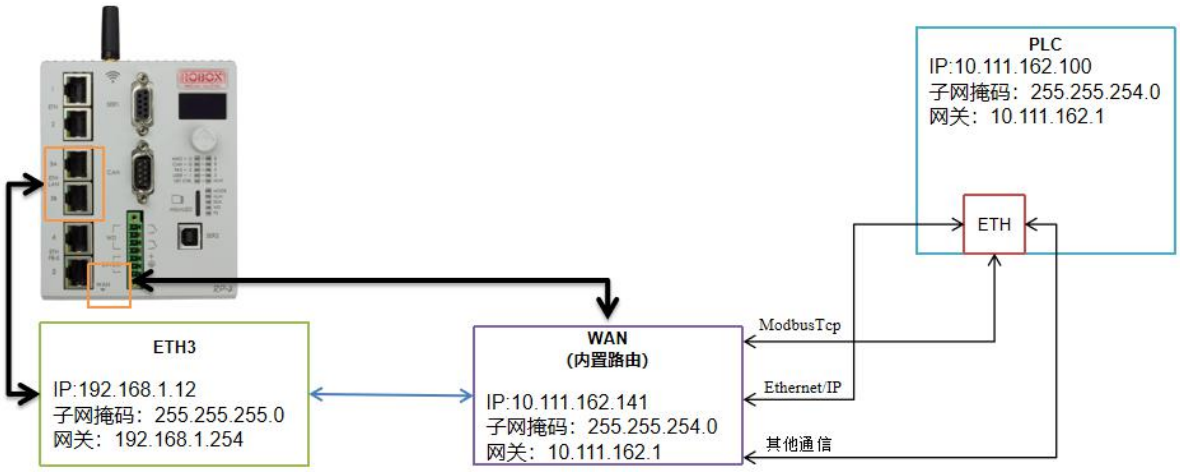


图 29-3 WAN 口通讯示意图

表 29-3 NAT 地址映射示例

ETH3 IP	ETH3 端口	WAN IP	WAN 端口	传输协议	说明
192.168.1.12	8100	10.111.162.141	8100	tcp	BCC
192.168.1.12	502	10.111.162.141	502	tcp	ModbusTcp
192.168.1.12	2222	10.111.162.141	2222	tcpudp	Ethernet/IP 实时通信
192.168.1.12	44818	10.111.162.141	44818	tcpudp	Ethernet/IP 非实时通信

4.3 轴参数和笛卡尔参数

轴参数主要为机器人的各个关节的参数，包括手自动的关节速度、减速比等参数；笛卡尔参数则为机器人在笛卡尔坐标系下运动时使用的参数，包括手自动的速度类参数。

自行修改机器人轴参数或者笛卡尔参数存在风险，建议联系售后技术人员确认后再进行修改，详细操作流程如下:

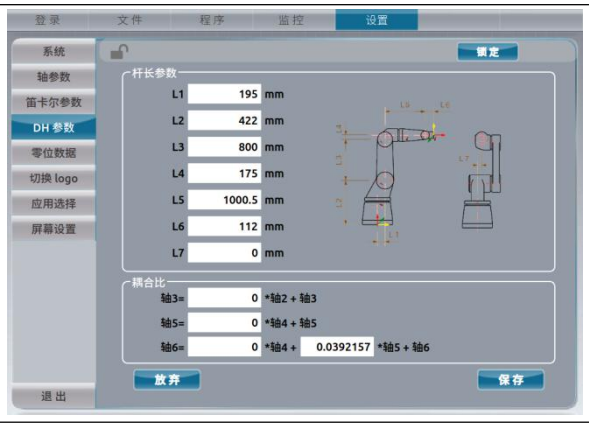
表 29-4 轴参数和笛卡尔参数操作流程

步骤	图示	说明
1.点击左侧的“**轴参数**”或者“**笛卡尔参数**”按钮，然后根据需要，点击编辑框输入，完后点击“**保存**”按钮。	 	若不修改参数，可以点击“**放弃**”按钮。 参数修改后需要重启机器人才能生效。

4.4 DH 参数

可以通过 DH 参数设置界面查看 DH 杆长参数和机器人耦合比参数。设置流程如下：

表 29-5 DH 参数操作流程

步骤	图示	说明
1. 点击左侧的“**DH 参数**”按钮，然后根据需要，点击编辑框输入，完后点击“**保存**”按钮。		若不修改参数，可以点击“**放弃**”按钮。 参数修改后需要重启机器人才能生效。

4.5 零点文件记录

零点文件记录的是机器人出厂时标准的零点位置，只有两种情况允许使用：

（1）机器人出厂时，经过激光标定零点之后需要使用该功能记录文件，此文件将是“零点恢复”

功能的基准，若没有记录此文件，零点恢复功能将不可用；

（2）机器人现场进行机械装拆，导致原始零点丢失，此时经过标准的零点标定操作（例如：激光标定，机器人零点标定中 20 点法）之后，执行零点记录操作，记录新的零点文件。非专业人员请勿操作。具体操作步骤如下：

表 29-6 零点文件重写操作步骤

步骤	图示	说明
1．点击“**零位数据**”按钮，进入界面。		单圈值为各轴当前位置的编码器单圈值数据；编码器值为当前位置的编码器值；零点数据为记录的零点位置。
2．点击**“记录”**按钮，并在弹出的提示框中点击**“是”**进行确认。	 	记录零点后，机器人当前位置将被设置为零点，需要重新校对。零点记录成功后，会提示相应弹窗。零点找回时，可根据零点恢复章节操作。

4.6 切换 Logo

前期准备：在 U 盘中导入需要切换的 Logo 图片，图片格式为 png，暂不支持其它格式的图片。目前在示教器上需要替换 4 处 Logo，分别为开机 Logo（分辨率：1024*768）、状态栏 Logo（分辨率：80*40）、登录界面 Logo（分辨率：650*260）和关于界面 Logo（分辨率：960*370），若替换图片不是标准的分辨率尺寸则会使 Logo 图片出现一定程度变形。

表 29-7 Logo 图片

开机界面 Logo	状态栏 Logo 和登录界面 Logo	关于界面 Logo
		

表 29-8 切换 Logo 操作流程

步骤	图示	说明
1. 进入设置界面，点击“**切换 Logo**”按钮。		在示教器上插入 U 盘，选择需要替换 Logo 的位置，然后点击“**替换**”按钮。 右侧 4 张图片是替换图片的缩略图。
2．在选择需要替换的 Logo 图片，然后点击“**导入**”按钮，然后点击“**Ok**”，确认替换。		
3．在示教器右侧相应的位置会显示替换的 Logo 的缩略图，表示 Logo 替换成功。		替换完开机界面和登录界面的 Logo 后须重启机器人才能在相应的位置更新。

4.7 应用选择

可以通过应用选择设置界面选择应用功能的开启与关闭，包括零点恢复、零点标定、冲压、弧焊等功能和工艺包。其操作流程如下：

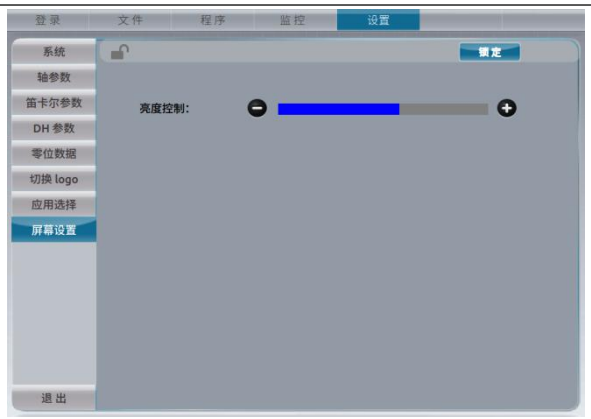
表 29-9 应用选择操作流程

步骤	图示	说明
1．进入设置界面，点击“**应用选择**”按钮。		
2．需用使用的应用勾选完毕后，确定无误后，点击“**保存**”按钮。		若对应用显示修改后仍想保留原应用显示设置，则点击“**放弃**”恢复原设置。 如果需要保存参数，需要注意，确定保存之后，则机器人会重启。

4.8 屏幕设置

屏幕设置目前只开放亮度设置。设置方式如下：

表 29-10 屏幕设置操作步骤

步骤	图片	描述
1.进入“**屏幕设置**”页面，调节亮度。		点击亮度控制条的左右“—”“+”可以调节示教器屏幕亮度。

第 5 章　点动操作

5.1 本章简介

本章主要介绍 EFORT 工业机器人的点动操作、坐标系统、点动操作注意事项和点动操作步骤。

5.2 什么是点动操作

点动操作是通过按压示教器面板右侧的点动按键“-”、“+”使机器人运动，此操作只允许在手动模式下进行。伺服使能后，需设置机器人的坐标系类型和运动速率，再进行点动操作。

点动操作分为连续点动和增量点动两种方式：

1) 连续点动是长按“-”、“+”按键使机器人运动；

2) 增量点动需设置步进长度，之后点按“-”、“+”按键使机器人进行增量式运动。

5.3 坐标系统介绍

坐标系是一种位置指标系统，其作用是确定工业机器人处于空间中的位置及其姿态。机器人根据不同的参考对象，使用以下四种坐标系。

1) 关节坐标系

关节坐标系是设定在工业机器人关节中的坐标系。在关节坐标系中，工业机器人的位置和姿态以各个关节底座侧的原点角度为基准，关节坐标系中的数值即为关节正负方向转动的角度值。

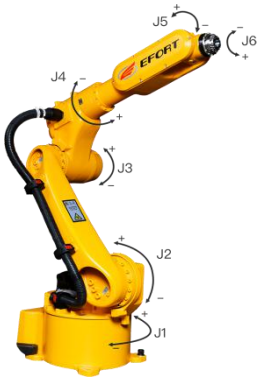


图 4-1 关节坐标系各关节轴运动角度图示

2) 机器人坐标系

机器人坐标系中的工业机器人的位置和姿态，通过从空间上的直角坐标系原点到工具侧的直角坐标系原点（工具中心点）的坐标值 X、Y、Z 和绕着空间上直角坐标系的 Z 轴、Y 轴、X 轴方向旋转的角度 A、B、C 予以定义。

图 4-2 机器人坐标系图示

3) 工具坐标系

工具坐标系,即安装在机器人末端的工具坐标系,原点及方向都是随着末端位置与角度不断变化的,该坐标系实际是由机器人坐标系通过旋转和位移变换得出。

图 4-3 工具坐标系图示

4) 用户坐标系

用户坐标系即用户自定义坐标系,是用户对每个作业空间进行定义的直角坐标系，该坐标系实际是对基础坐标系通过轴向偏转角度变换得出。

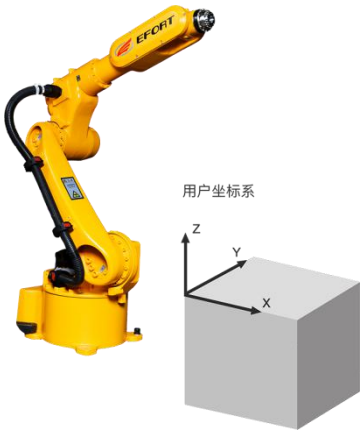


图 4-4 用户坐标系图示

5.4 点动操作注意事项

1) 操作者须站立于机器人运行的工作空间之外；

2) 操作者保持从正面观察机器人，确保发生紧急情况时有安全退路；

3) 确保机器人辐射范围内没有人员；

4) 查看机器人有无报警，如有报警请先清除后运行；

5) 查看机器人机械零位是否与示教器各轴位置相吻合；

6) 上伺服前确认点动全局速度，确认当前所选的坐标系。

5.5 开始点动操作

以管理员身份登陆后，点击菜单栏**“监控”**,然后点击**“位置”**，在跳转出的界面中即可进行以下的点动操作，如图 4-5 所示。

图 4-5 关节坐标系下位置监控界面

点击示教器面板上的坐标系切换按键可进行坐标系类型切换，如图 4-6 所示。切换顺序依次为关节坐标系->机器人坐标系->工具坐标系->用户坐标系，切换结果显示于示教器状态栏位置。

图 4-6 坐标系切换和不同坐标系图标

5.5.1 关节坐标系-点动操作

将坐标系类型设置为关节坐标系，点击示教器面板下方的坐标系按键，直到示教器状态栏中显示**“关节”**状态；或手动点击状态栏按钮，出现下拉框后选择“关节”。

按住手压开关的同时，点击示教器面板右侧的相应**“-”**、**“+”**按键，如图 4-7 所示，即可调节工业机器人相应关节轴的运动角度。

图 4-7 手压开关和点动按钮位置

5.5.2 机器人坐标系-点动操作

图 4-8 机器人坐标系下位置监控界面

如图 4-8 所示，切换坐标系为**“机器人”**，点击示教器面板右侧的相应**“-”**、**“+”**按键，即可沿机器人坐标系的 X、Y、Z、A、B、C 方向移动机器人。其中 X、Y、Z 表示与机器人坐标系 X、Y、Z 方向平行的正负方向，A、B、C 表示绕着机器人坐标系 Z、Y、X 方向正负转动方向。

5.5.3 工具坐标系-点动操作

图 4-9 工具坐标系下位置监控界面

如图 4-9 所示，切换坐标系为**“工具”**，点击示教器面板右侧的相应**“-”**、**“+”**按键，即可调节工具坐标系中相应 X、Y、Z、A、B、C 的坐标值。

5.5.4 用户坐标系-点动操作

图 4-10 用户坐标系下位置监控界面

如图 4-10 所示，切换坐标系为**“用户”**，点击示教器面板右侧的相应**“-”**、**“+”**按键，即可调节用户坐标系中相应 X、Y、Z、A、B、C 的坐标值。

5.5.5 点动-快速运动

将机器人模式开关转动至中间位置（T1），此时状态栏中的图标变更为“手动慢速”。将机器

人模式开关转动至右侧位置（T2），此时状态栏中的图标变更为“手动全速”。

手动全速模式下，通过调速按键“**V+**”和“**V-**”调整全局速度，其速度范围可设置为 0%—100%，相应的，手动低速模式下，其速度范围可设置为 0%—20%。

现选择手动全速模式且全局速度调节为 100%，取消勾选‘慢速’复选框，如图 4-11 所示，在这种设置下执行点动操作，机器人轴运动速度较快，坐标系值会以较快幅度发生变化。

图 4-11 未勾选慢速复选框

5.5.6 点动-慢速运动

选择手动全速模式且全局速度调节为 100%，勾选‘慢速’复选框，如图 4-12 所示，在这种设置下执行点动操作，机器人轴运动速度较慢，坐标系值会以较慢幅度发生变化。

图 4-12 勾选慢速复选框

5.5.7 点动-步进运动

图 4-13 选择步进运动

点击步进长度设置单步运动的步长，上图中设置步长为 15 且坐标系为关节坐标系，每次点按相应关节的“-”、“+”按键，机器人首先运动到相应方向距离最近的步进长度值整数倍位置，然后再按键机器人每次运动一个步进长度距离。比如一轴关节角度目前为 11.96 度，选择步进运动设置步进长度为 15，按一轴“+”按键一次，此时机器人运动至距离 11.96 最近的 15 度，其中 15 度是此时步进长度最近的正整数倍位置，而后每次按一轴“+”按键一次，一轴关节角度增加 15 度。

第 6 章　文件管理与编程

6.1 本章简介

本章主要介绍 EFORT 工业机器人的文件管理和编辑程序。

6.2 文件管理

文件管理器是为方便用户管理项目文件而设计，主体部分展示目录结构，底部为文件操作的功能按钮。支持新建、删除、重命名、复制、粘贴、剪切等功能。

由于程序文件都存储在控制器上，因此更换示教器不会造成程序文件丢失。如果需要在不同机器人之间拷贝程序，请使用 U 盘，示教器上提供了标准 USB 接口。

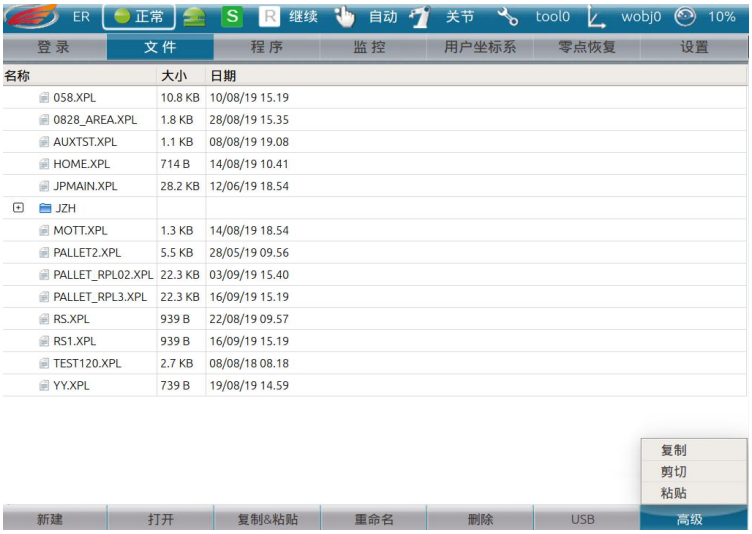


图 5-1 文件管理界面

6.2.1 新建

“新建”包括新建文件和新建文件夹。点击界面底部**“新建”**按钮，在菜单中选择文件或者文件夹。然后用弹出的键盘输入文件或者文件夹的名称。如果需要在指定文件夹下新建文件，需选中该文件夹。

6.2.2 打开

“打开”是将当前选中的文件，从存储卡中加载到控制器内存中。

6.2.3 复制/粘贴

“复制/粘贴”包括了复制或粘贴两个操作。首先复制选中的文件或者文件夹，然后粘贴该文件或者文件夹，粘贴后的文件或者文件夹将自动重命名。

6.2.4 重命名

“重命名”需注意，不可使用已有的名称。在对文件进行重命名时，可不输入文件后缀。

6.2.5 删除

“删除”操作需谨慎，该操作不可逆。

6.2.6 USB

“USB”操作包括从 USB 导入和导出到 USB。

从 USB：在示教盒接口插入 U 盘，点击界面底部**“USB”**按钮，在菜单中选择**“从 USB”**，弹出文件选择窗口，选择需要导入的文件。

到 USB：在示教盒接口插入 U 盘，选择一个文件或者文件夹，点击界面底部**“USB”**按钮，在菜单中选择**“到 USB”**。

6.2.7 高级

点击**“高级”**功能按钮会展开折叠菜单，折叠菜单中包含的功能按钮有**“复制”**、**“剪切”**、**“粘贴”**，该处**“复制”**、**“剪切”**、**“粘贴”**功能能完成不同目录间的文件操作。首先，选中文件，点击“**高级**”按钮中的“**复制**”或“**剪切**”按钮，然后选中目标位置，点击“**高级**”按钮中的“**粘贴**”按钮完成整个粘贴工作。

6.3 编辑程序

在文件管理器界面，新建或者加载一个程序，示教器界面会自动跳转到程序编辑器界面。程序编辑器显示的程序，即当前控制器内存中加载的程序。如下图：

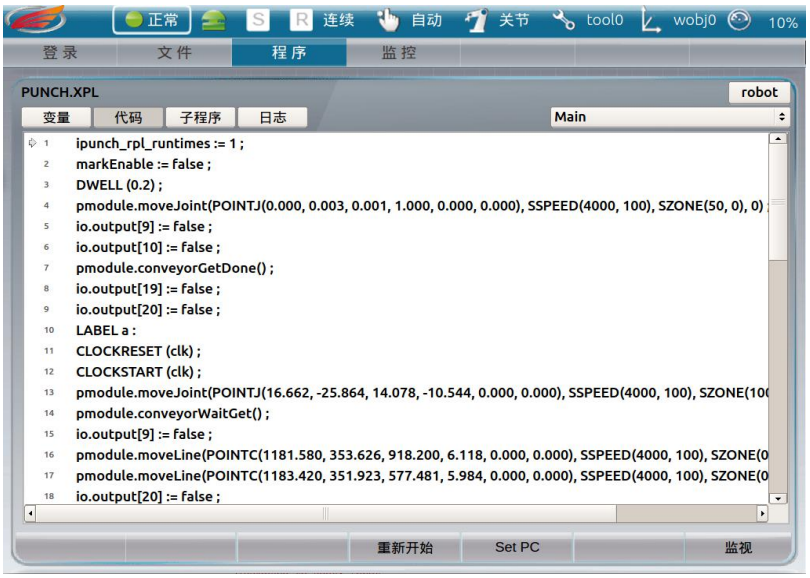


图 5-2 编辑程序界面

编辑一个程序，包括两个方面：

1) 程序指令的操作；

2) 程序变量的操作。

6.3.1 程序中变量的操作

在进行程序变量操作之前，需要了解一些相关信息，例如，变量的数据类型、变量的存储类型、变量的作用域等。

1) 变量的数据类型

变量数据类型包括 TOOL、SPEED、POINTC、ZONE、VECT3、POINTJ、BOOL、DINT、UDINT、TRIGGER、LREAL、STRING、REFSYS、ROBOT、INTR、CLOCK、TRACKING、EPOINTC、EPOINTJ 。

TOOL：工具，运动指令中使用的工具参数，用于移动指令。

SPEED：速度，用于指定移动指令中的目标速度。它包含以 mm / s 表示的切向速度和以度/ s 表示的定向速度。

POINTC：笛卡尔空间位姿，用于表示一个笛卡尔点。数据有 LREAL 类型的 x，y，z，a，b，c 分量。分量 x，y，z 表示位置，分量 a，b，c 表示姿态。分量 a 指向 z 轴方向，b 指向 y 轴方向，c 是指向 x 轴。

ZONE：圆弧过渡，用于指定两个连续移动指令之间的混合参数。它包含以 mm 为单位的线性距离和以度为单位的重定向角距离。

VECT3：三维实向量，用于表示三维向量。数据由(x，y，z) 3 个 LREAL 类型值组成。

POINTJ：关节位置，用于确定机器人关节的位置。数据有 LREAL 类型的 j1，j2，j3，j4，j5，j6 分量。

BOOL：布尔，布尔类型数值，值可以是 true 或 false。初始化后的默认值为 false。

DINT：双精度整数，用于整数值，可以是正数或负数；整数值用 32 位值表示。

UDINT：无符号双精度整数，用于仅为正数的整数值。整数值用 32 位值表示。

TRIGGER：触发，在运动指令中用于触发事件的数据类型。

LREAL：长实数，用于具有双精度的十进制数。该值是 64 位有符号值。

STRING：字符串，用于存储字符串，最多可包含 128 个字符。

REFSYS：参考坐标系，用于定义笛卡尔空间运动的参考坐标系。

ROBOT：机器人轴组名，用于程序中运动指令指定轴组。

INTR：中断处理，用于中断处理。请注意，此类数据必须在程序执行时只设置一次，否则将发出错误。

CLOCK：时间测量时钟，用于时间测量。时间测量以秒表示。无法直接设置或读取此类数据。

TRACKING：跟踪，用于存储跟踪应用程序的数据。

EPOINTC ：笛卡尔空间位姿（附加轴关节位置点），用于表示笛卡尔点和附加轴关节位置。它包含与 POINTC 类型相同的数据以及与附加轴关节位置相关的值 ea1，ea2，ea3，ea4，ea5，ea6 分量。

EPOINTJ：关节位置（附加轴关节位置点），用于定义参考包含附加轴关节位置的机器人关节的位置。它包含与 POINTJ 类型相同的数据以及与附加轴关节位置相关的 ea1，ea2，ea3，ea4，ea5，ea6 分量。

2) 变量的存储类型

变量的存储类型包括变量、常量、保持三种。

变量：可变量，该变量可以在 RPL 程序中赋值，当 RPL 程序重新启动时它的值就会丢失。

常量：常量变量，该变量不能在 RPL 程序中赋值，必须使用初始值来赋值。

保持：持续性变量，当 RPL 程序从内存中卸载时，变量的值将被保留。

3) 变量的作用域

变量作用域和变量的用途有关，变量用途分为程序变量和功能块变量，当变量用途为程序变量，变量的作用域只能是本地，该作用域下的变量只能在定义它的程序或子例程中被看到和使用。外部的程序或子例程无法看到和使用。

当变量用途为功能块时，变量的作用域包括本地、公共、任务和全局，分别如下：

本地：该作用域下的变量可以在所有程序或子程序中看到和使用。使用本地变量，您可以在子例程中设置一个值，稍后您可以从另一个子例程中读取该变量。您不能用相同的名称定义多个模块的局部变量。这些变量不能从其他模块中看到。

公共：它就像本地变量，但是这个变量可以从其他模块中看到。在其他模块中，可以使用模块名来使用这种类型的变量(e.g: moduleName.variableName)。

任务：这就像公共变量。在其他模块中，这种变量可以在不使用模块名之前使用(e.g: variableName)。

全局：这种类型的变量对于系统的所有任务来说都是通用的。在不同的任务之间共享数据是很有用的。如果对相同的全局有不同的定义，则会报告错误。一个全局变量在前面没有模块名。

4) 全局变量

目前 RPL 编辑器中定义了三种数据类型的全局变量供 RPL 程序进行数据逻辑交互，根据存储类型分为两个类别，为程序重新启动值就会丢失的可变量和重新重新启动值会一直保留的持续性变量。

表 5-1 预定义的全局变量

变量名称	长度	数据类型	存储类型
grpl.gBool[]	16	布尔型变量	可变量
grpl.gInt[]	16	整型变量	可变量
grpl.gReal[]	32	浮点型变量	可变量
grpl.pBool[]	8	布尔型变量	持续性变量
grpl.pInt[]	8	整型变量	持续性变量
grpl.pReal[]	8	浮点型变量	持续性变量

5) 添加变量

图 5-3 添加变量界面

如图 5-3，在变量管理界面，点击红色标识处的按钮，弹出添加变量窗口。根据需要选择变量的作用域、数据类型、存储类型。

6) 修改变量

在变量管理界面，选择需要修改的变量，双击弹出变量窗口，根据需要修改变量。

7) 删除变量

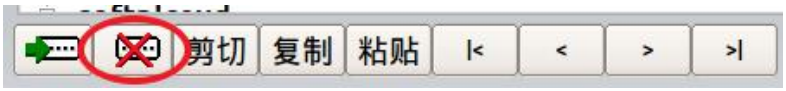


图 5-4 变量操作按钮

在变量管理界面，选择需要删除的变量，然后点击图 5-4 中红圈按钮，可删除变量。

6.3.2 程序指令的操作

1) 指令的类型

目前 RPL 程序语言包括 Common（通用）、Movement、Interrupt、Other（其他）、Trigger 五种指令集。如下图所示：

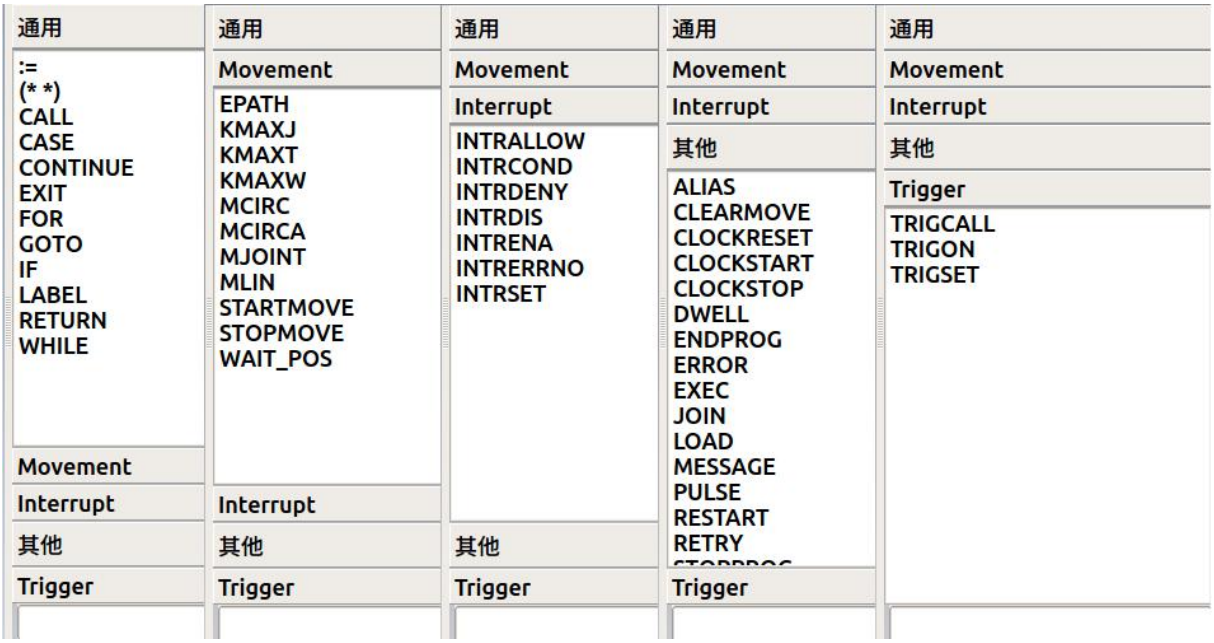


图 5-5 RPL 指令种类

2) 添加指令

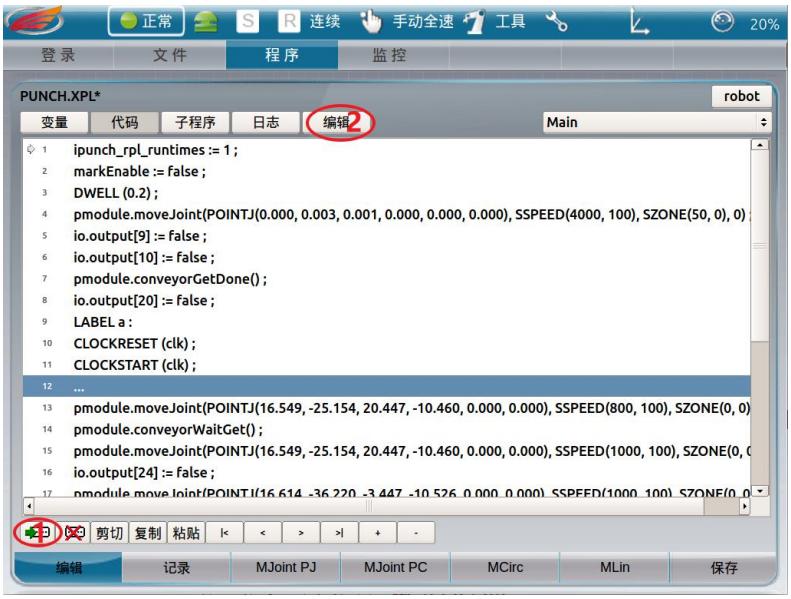


图 5-6 添加指令界面

在程序编辑界面，点击“**添加**”按钮（如图标记 1），将会在当前选中行的上一行插入“**...**”。选中“**...**”，然后点击“**编辑**”按钮（如图标记 2），进入指令编辑界面。

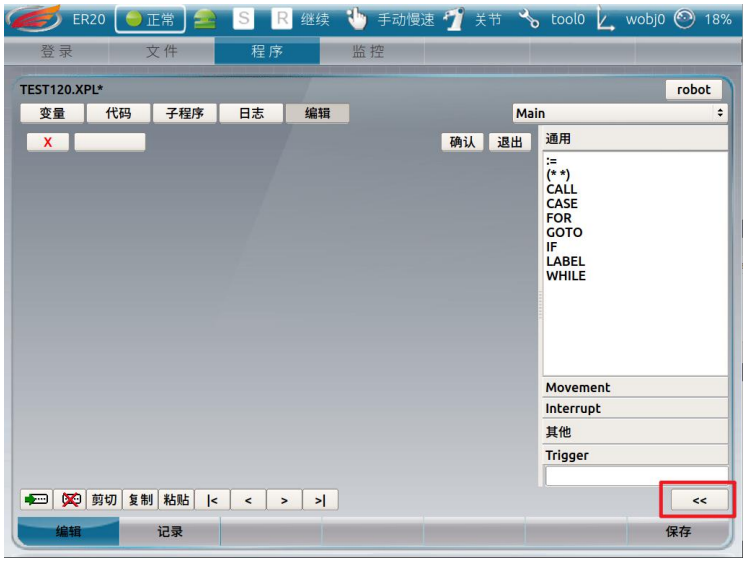


图 5-7 选择指令界面

如图 5-7 所示，右侧是 RPL 语言的指令集，选中需要插入的指令，然后双击或点击上图红色方框**“左移”**按钮。不同的指令需要不同的参数，设置完参数后，点击**“确认”**按钮，即可插入指令。点击**“退出”**可取消指令插入。

3) 修改指令

在程序编辑界面，选中需要修改的指令，点击**“编辑”**按钮，进入指令编辑界面。

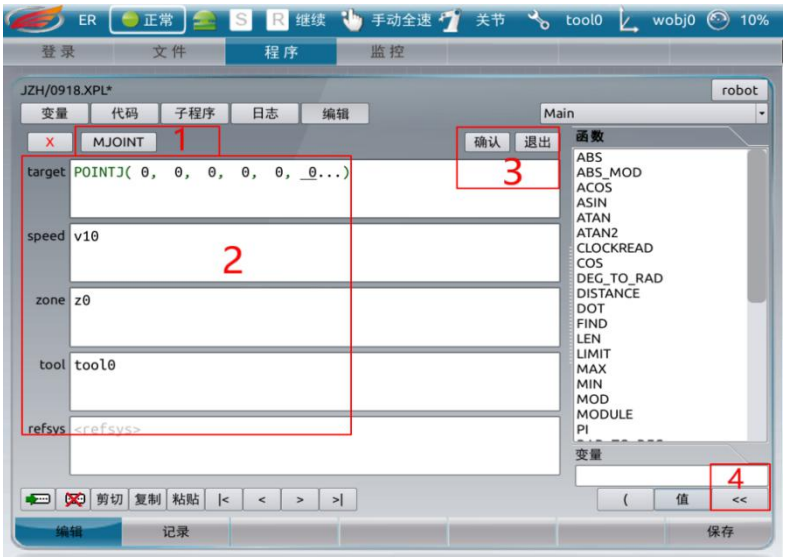


图 5-8 编辑指令及其参数操作界面

标记 1 显示为当前指令名称，可先选中标记 1，在最右侧指令处双击其他指令，或者点击上图右下角标记 4**“左移”**按钮，可以完成指令的修改。

标记 2 区域显示的是当前指令的参数信息，可设置或修改。

标记 3 处，设置完指令和参数后，点击**“确认”**按钮，即可完成本条指令编写。点击**“退出”**按钮可取消本条指令编辑，并退出编辑界面。

4) 删除指令

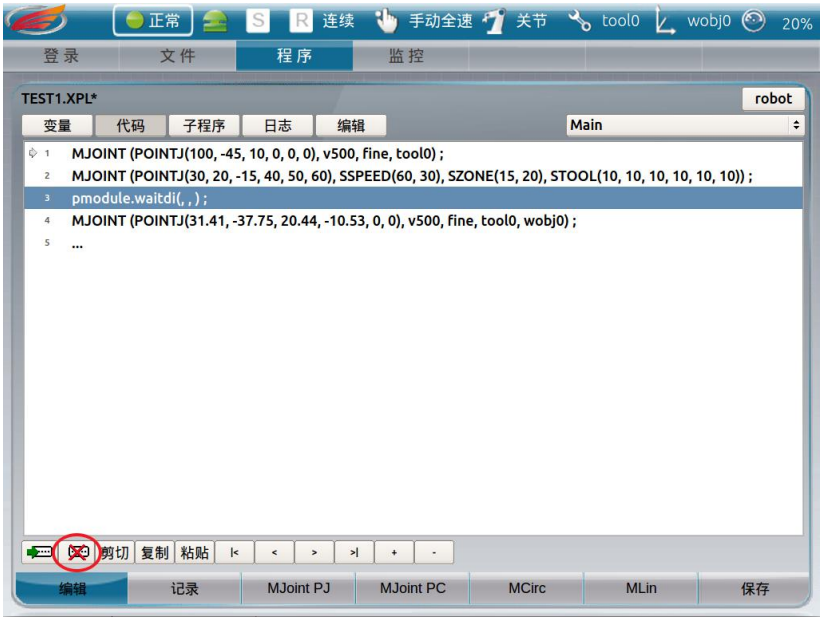


图 5-9 删除指令操作界面

在程序编辑界面，选中需要删除的程序行，然后点击**“删除”**按钮（如图红色标记），即可删除当前选中行，注意此操作不可逆。

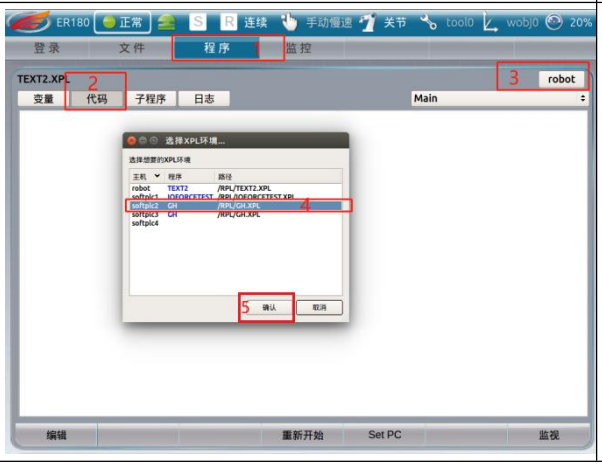
6.4 软 PLC 程序

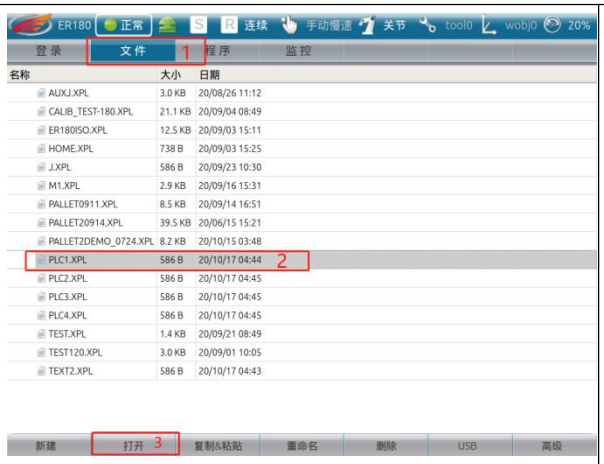
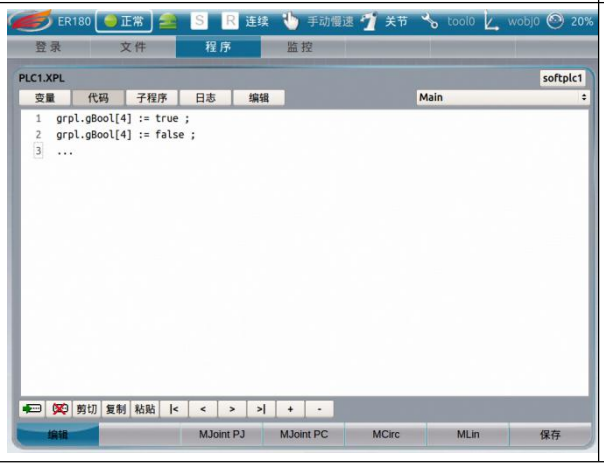
6.4.1 软 PLC 功能简介

软 PLC 功能是内置于控制器的一个虚拟 PLC。可以采用 RPL 指令进行逻辑指令的编写，但是不能处理机器人运动指令。ROBOX 系统一共内置 2 至 4 个软 PLC，软 PLC 程序可以与主程序并行运行，二者可以进行 IO 信号、总线数据、自定义数据的交互。

6.4.2 编辑软 PLC 程序

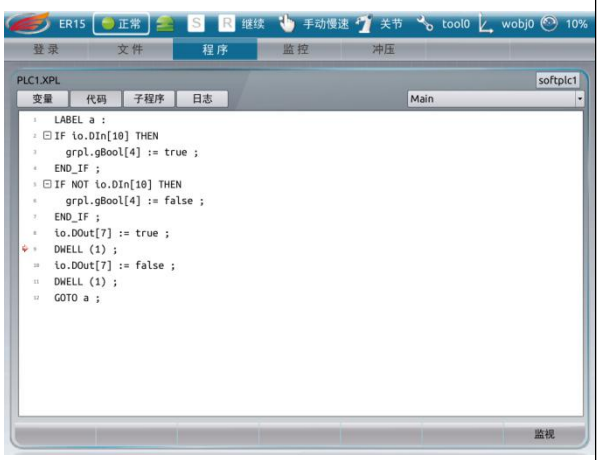
表 5-2 编辑及运行软 PLC 程序步骤

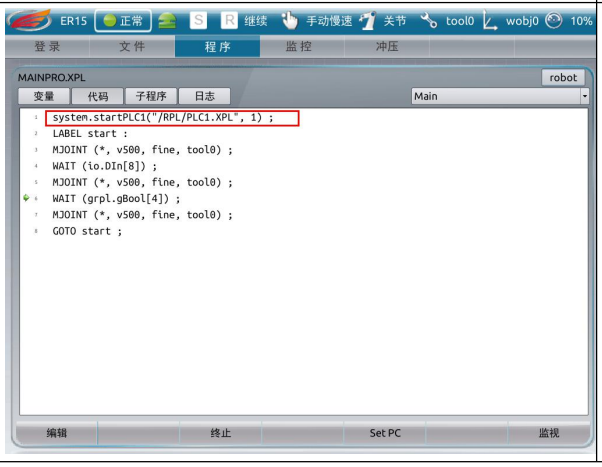
步骤	图示	说明
1.查看当前在使用的软 PLC 程序。		开机登录权限后，进入程序界面，点击界面右上角的按钮，可以查看当前设定的主程序和软 PLC 程序。 如果需要编辑软 PLC 程序，则需要选中相应的 XPL 环境，再点击“确认”按钮。

2.加载程序。		选择好 XPL 环境后，点击进入文件界面，选择加载已存在的 XPL 格式程序文件，或新建一个空白的 XPL 格式程序文件。**请注意**：如果选择的 XPL 环境是“robot”，则加载的程序将作为主程序进行使用，如果选择的环境是“softplc1/2/3/4”，则加载的程序将作为软 PLC 程序使用。
3.编辑程序。		加载或新建程序完成后，可以进行编辑，此时如果是作为软 PLC 程序使用，则只能编辑逻辑语句，而不能使用运动指令。指令和变量的编程使用方式与常规程序相同。

6.4.3 使用软 PLC 程序

表 5-3 指令调用软 PLC 程序步骤

步骤	图示	说明
1.运行 PLC 程序。		完成软 PLC 程序的编辑。 手动运行：如左示意图，选择 XPL 环境为“softplc1/2/3/4”，打开一个编辑好的软 PLC 程序文件，在“代码”界面中通过“Set PC”设置程序当前运行行，自动模式下按下“PWR”或自动模式下按下手压完成上伺服，再按下运行按钮启动程序。 程序调用：选择 XPL 环境为“robot”：并在文件界面打开一个主程序文

		件，在主程序的编辑状态下，插入 CALL 指令，参数为 system.startPLC() 系列指令，编辑完成后启动主程序可实现软 PLC 程序文件的调用。
2．机器人主程序调用软 PLC 程序。		通过指令启动软 PLC 程序，指令调用格式见表格最后详细说明。 **示例主程序说明：** 当主程序运行至第 4 行时等待输入端口信号，继续运行至第 6 行等待信号，结合上步的软 PLC 程序，给 10 号输入端口信号，此时程序继续运行。

主程序中调用软 PLC 程序指令格式：

Call system.startPLC1(“string”，type)

Call system.startPLC2(“string”，type)

Call system.startPLC3(“string”，type)

Call system.startPLC4(“string”，type)

上述四条指令可分别对应指定启动 PLC1~PLC4 任务；

string：PLC 程序的完整路径,可点击程序界面右上角按钮，在弹出界面中进行查看 PLC 程序存储路径，例：/RPL/PLC1.XPL。

type：PLC 运行方式，0 表示主程序运行时 PLC 程序运行，主程序暂停时，PLC 暂停；1 表示无论主程序是否运行或暂停，PLC 程序一直在后台运行。

6.4.4 设置软 PLC 自启动

通过界面设置软 PLC 开机自启动后，下次断电重启后，软 PLC 程序可以自动运行。

步骤	图示	说明
1．按图示标记编号，依次打开示教器桌面，点击**“程序预约”**图标进入主界面。		

2.进入界面后，根据图示，点击“**软PLC**”按钮进入设置界面。	 	备注： 指令启动：表示对应的软PLC需要通过程序编辑器调用system.startPLC函数来启动； 开机自启动：表示设置完成后，程序编辑界面配置的对应的软PLC程序，会在下次开机后自行启动，无需通过调用函数来启动。
3.界面支持两个软PLC，分别是softplc1和softplc2，设置模式有指令启动和开机自启动两种模式		备注; 对应的程序设置需要参考6.4.2节的**编辑 PLC 程序**内容，

4.例如：将 softplc1 设置为开机自启动，点击**“保存”**成功后生效。		备注： 设置为开机自启动时，务必需要保存PLC程序没有任何问题，否则存在危险。
6.完成设置后，点击**“返回”**按钮返回主界面，设置完毕。		

6.5 调试程序

1) 程序指针

程序指针用于显示当前程序运行位置及状态。

表 5-1 指针状态说明

状态	说明
	当前没有任何操作，只指示当前行号。
	表示当前行处于预备状态，可以执行。
	表示当前行处于激活状态，在运行中。
	当前程序行有错误。
	当前程序行有运动。
	表示当前行处于激活状态，且有运动在执行。
	当前行运动有错误。

表 5-2 程序运行状态说明

模式	说明
单步进入	主程序每执行一行结束都将停下。当执行子程序时会进入子程序的界面，并且在子程序中每执行一行都停下。
单步跳过	主程序每执行一行结束都将停下。当执行子程序时会进入子程序的界面，并且一次性运行完子程序内全部指令。
继续	程序开始执行后，一直运行到程序末尾结束执行。
运动进入	主程序每执行至一条运动指令前停下，当执行子程序时会进入子程序的界面，并且一次性运行完子程序内全部指令。
运动跳过	程序开始执行后，一直运行到程序末尾结束执行。

2) 单步运行

在运行程序前，需要将机器人伺服使能（将钥匙开关切换到手动模式，并按下手压开关）。点击“**F3**”切换至“**单步进入**”状态，这里以“单步进入”状态为例。

选择第 11 行，点击“**Set PC**”，将程序指针定位到某一行，这里以第 11 行为例。

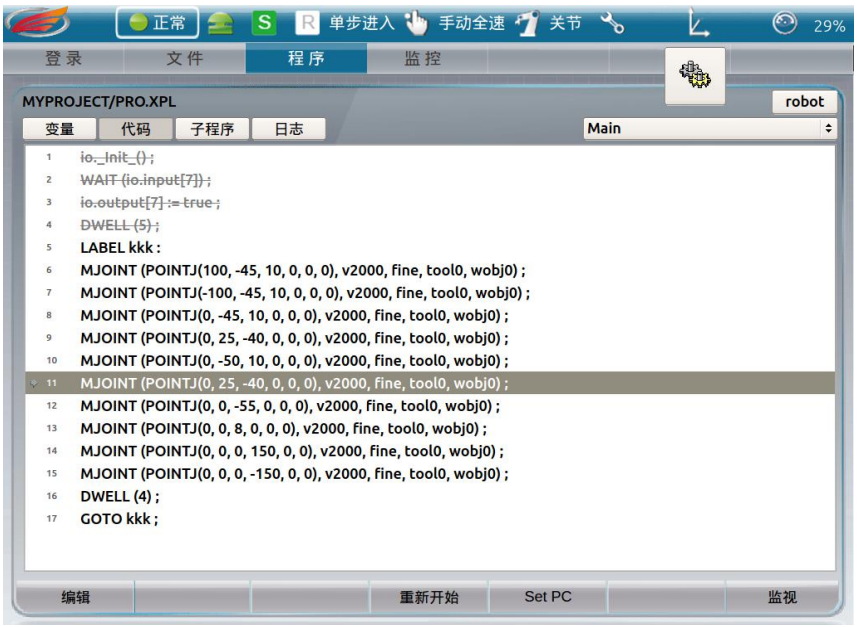


图 5-10 单步运行界面

点击“**开始**”按钮，程序从当前行开始运行。当前行运行完成后，指针将跳转至下一行，行首的程序指针状态由灰色状态变成绿色状态。

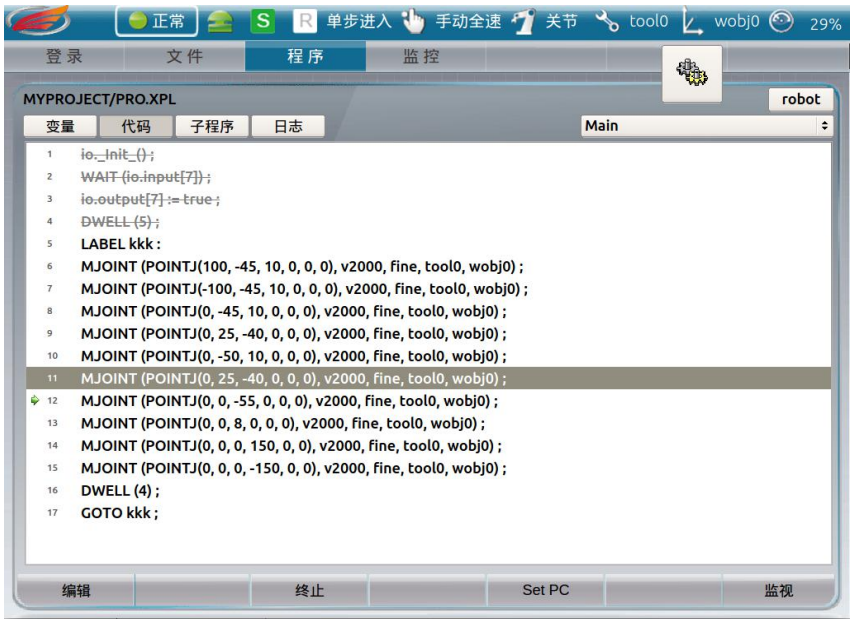


图 5-11 单步进入模式

若选择其他行，再点击“**Set PC**”，指针可以切换到该行。

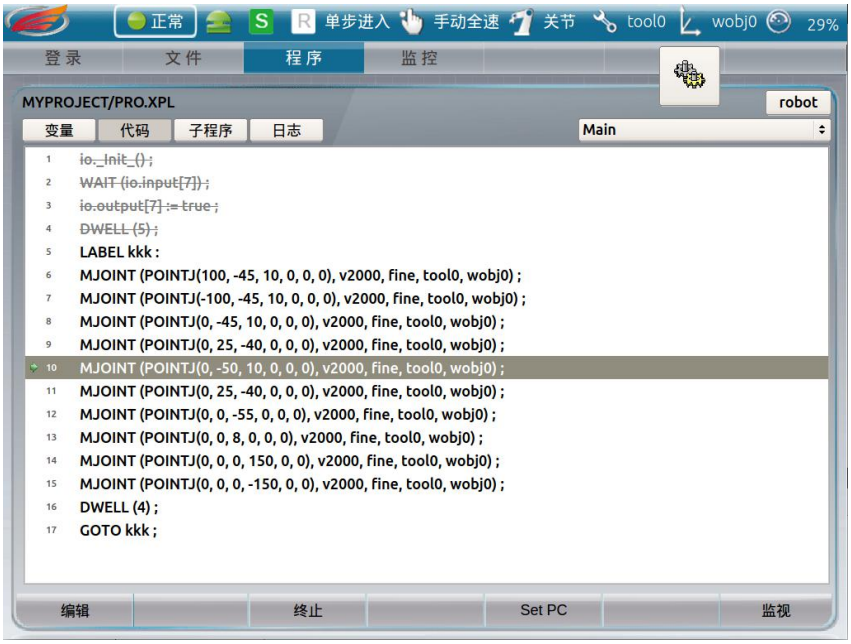


图 5-12 设置 PC 操作

若点击**“终止”**按钮，行首的程序指针由绿色状态变成灰色状态，当前程序被终止。

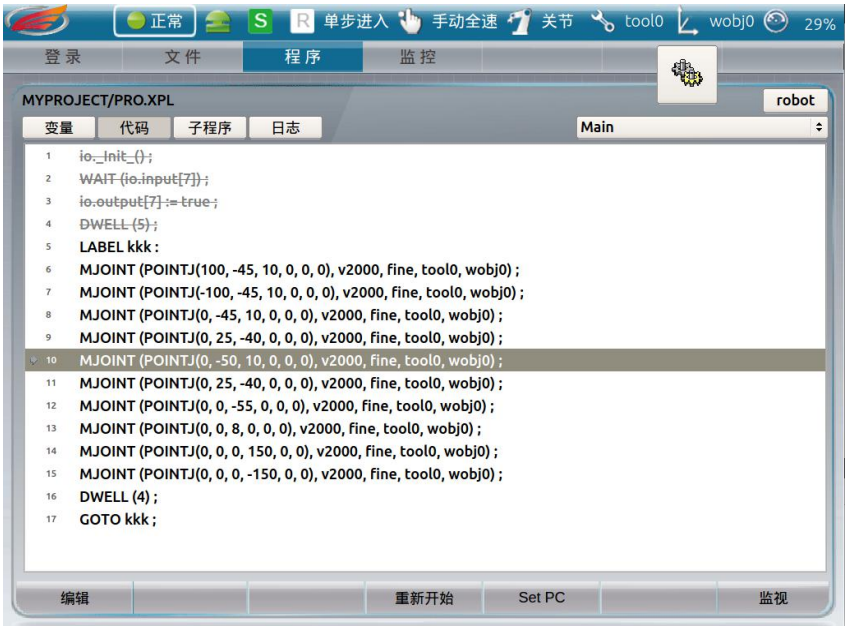


图 5-13 终止程序操作

点击**“重新开始”**按钮，程序指针会返回至第一行。

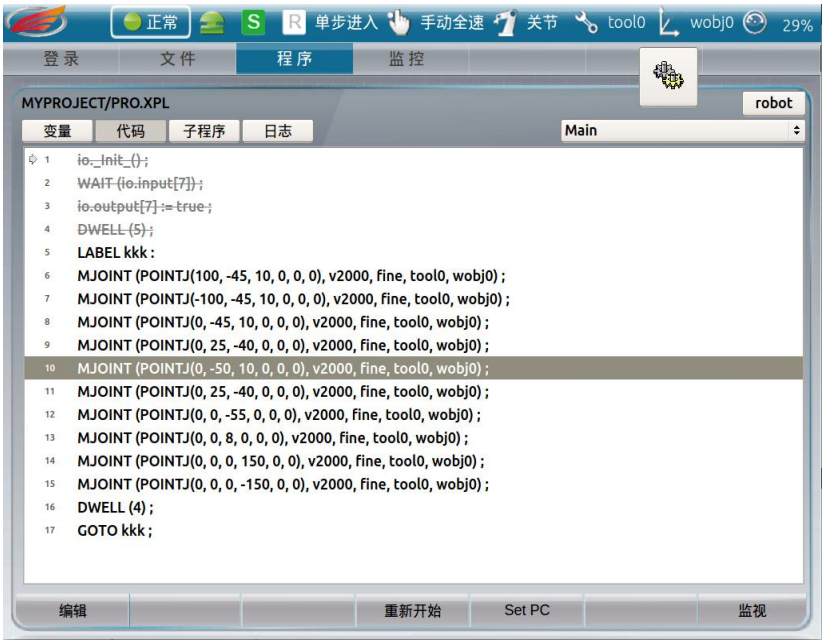


图 5-14 重新开始操作

点击**“监视”**按钮可以查看当前机器人的位置。

程序运行过程中，在程序区域下方的操作按钮中，除了“**监视**”按钮其他均会被隐藏。当程序执行到末尾结束后，程序指针会消失，需要重新设置程序的执行位置。

3) 继续

在运行程序前，需要将机器人伺服使能（将钥匙开关切换到手动模式，并按下手压开关；或将钥匙开关切换到自动模式，并按下示教器上**“PWR”**功能键），该过程与单步运行相似。

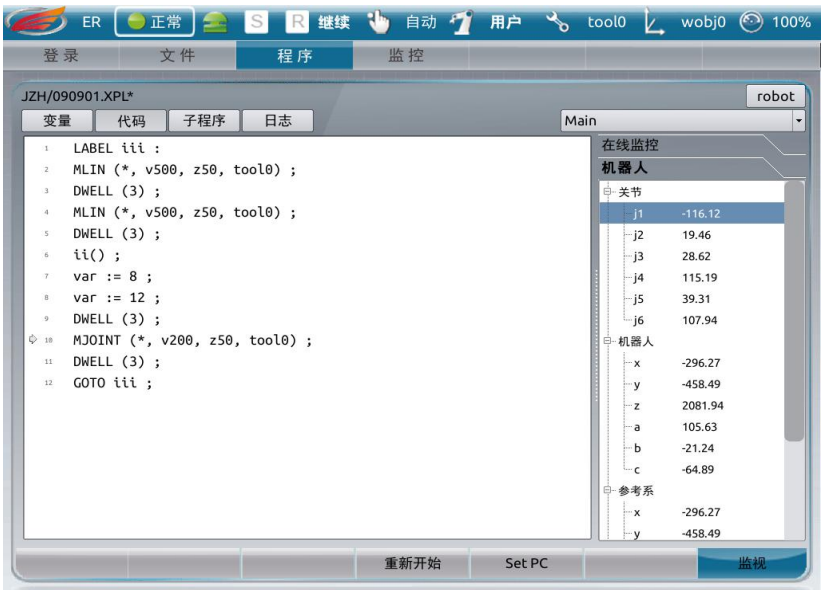


图 5-15 继续模式下运行程序

与单步运行不同之处在于，当程序从某一行开始执行后，直到程序末尾结束。在运行过程中点击**“暂停”**按钮，程序暂停运行；再按下**“开始”**按钮，程序能够继续执行。

4) 运行错误

当编辑的程序文件存在问题，语句存在问题，以及运动的错误都会产生报警。

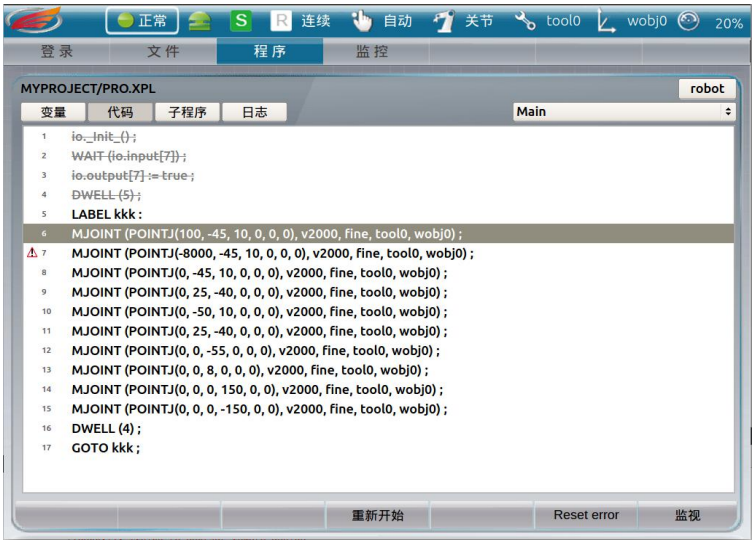


图 5-16 程序运行出错

通过点击**“日志”**（英文“Log”）按钮查看运行日志，可以获取具体的报警信息。

6.6 子程序

6.6.1 子程序的分类

子程序大致可分为 3 类：

1) 无输入无输出参数的子程序；

2) 有输入无输出参数的子程序；

3) 有输入有输出参数的子程序。

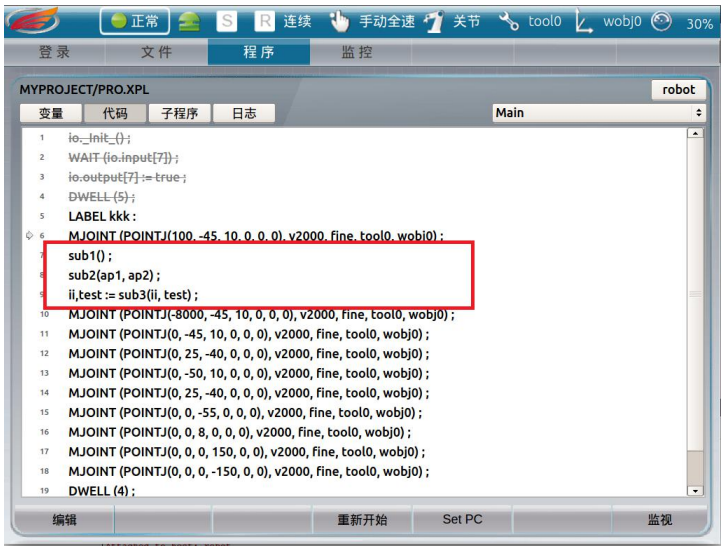


图 5-17 调用子程序操作

如图 5-17 所示；sub1()是无输入无输出参数的子程序；sub2(ap1, ap2)是有输入无输出参数的子程序；ii,test := sub3(ii, test)是有输入有输出参数的子程序。

6.6.2 新建子程序

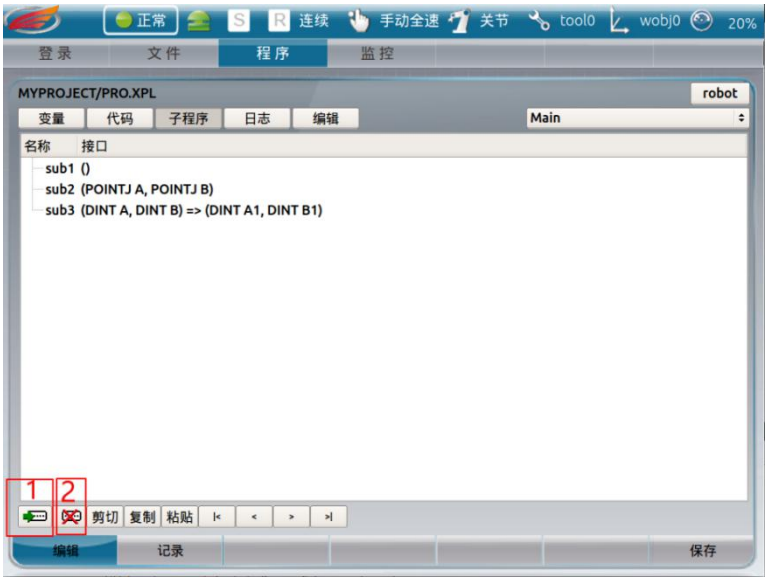


图 5-18 新建或删除子程序

如图，在“**子程序**”标签页面，点击左下角标示 1 的**“新建”**按钮，在弹出的软键盘中输入子程序的名称，即可新建一个子程序。

6.6.3 修改子程序

若右上角的下拉框显示的是 main，“变量”和“代码”即是主程序的变量管理和程序管理。

若右上角显示的是子程序名，“变量”和“代码”即是子程序的变量管理和程序管理。

点击下拉框，选择需要修改的子程序，此时，“变量”和“代码”即是子程序的变量管理和程序管理。子程序的修改请参考主程序的修改。

6.6.4 删除子程序

如图 5-18 所示，点击“子程序”标签，选中需要删除的子程序，然后点击左下角标示 2 的**“删除”**按钮，即可完成对子程序的删除，此操作不可逆。

6.6.5 子程序的调用

[变量列表] := 子程序名 (输入参数) ;

例: ExecuteSquare(A,B,C,D) ;

调用子程序 ExecuteSquare，该子程序有 4 个参数。

“子程序”标签中：

ExecuteSquare (POINTC A, POINTC B, POINTC C, POINTC D) ;

“变量”标签中：

+Input

POINTC A

POINTC B

POINTC C

POINTC D

子程序“代码”标签中：

MJOINT (A)

MLIN (B)

MLIN (C)

MLIN (D)

MLIN (A)

例: A,B,C,D := RotoTranslation(A,B,C,D,RT) ;

调用子程序 RotoTranslation，该子程序有 5 个输入参数。在执行完子程序后，输出 4 个变量。输出的四个变量给调用程序中的变量赋值。在这个例子中，四个点（A、B、C、D）转换成 RT 坐标系下的值。

“子程序” 标签：

RotoTranslation (POINTC A, POINTC B, POINTC C, POINTC D, POINTC RT) => (POINTC A1, POINTC B1, POINTC C1, POINTC D1)

变量（英文“Vars”）标签：

+Input

POINTC A

POINTC B

POINTC C

POINTC D

POINTC RT

+Output

POINTC A1

POINTC B1

POINTC C1

POINTC D1

代码（英文“Code”）标签：

A1 := FROMLOCAL(A, RT) ;

B1 := FROMLOCAL(B, RT) ;

C1 := FROMLOCAL(C, RT) ;

D1 := FROMLOCAL(D, RT) ;

第 7 章　坐标系管理

7.1 本章简介

本章主要介绍 EFORT 工业机器人的工具坐标系标定和用户坐标系标定。

7.2 工具坐标系标定

表 6-1 工具坐标系标定方法说明

	方法说明
标定方法	TCP&默认方向：方向与法兰末端一致。 TCP&Z：工具的 Z 方向需要标定确定。 TCP&Z, X：工具的 Z，X 方向需要标定确定。

7.2.1 工具标定

这里以 TCP&默认方向标定为例，操作步骤如下：

表 6-2 TCP&默认方法工具标定操作流程

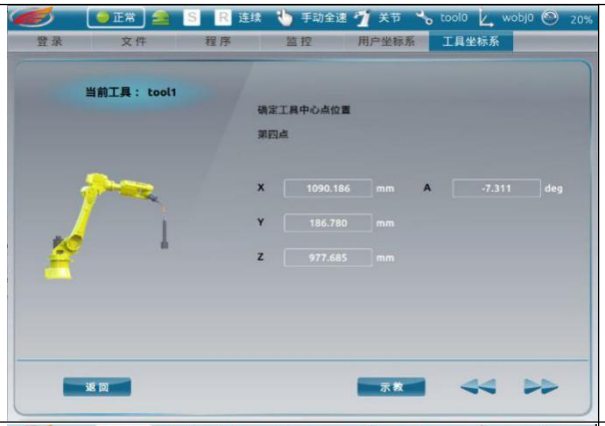
步骤	图示	描述
1.在桌面点击**“工具坐标系”**的图标，计入工具标定设置界面。		所有已定义的工具坐标系名称列表。 手动标定的方法，包括 TCP(默认)，TCP&Z 以及 TCP&Z, X 三种方法。
2.在工具设置界面点击**“标定”**按钮，进入标定界面，显示需要标定的第一点。		移动机器人将工具末端对准参考尖点。 点击**“示教”**按钮，将当前机器人位置记录。 示教完成后，点击右箭头图标标定下一个点。 **注：标定点是机器人以不同的姿态去对准同一个尖点。** 若未标定完成，需要结束标定过程，点击**“返回”**按钮，返回设置界面。

3.标定第二点界面。后续标定 TCP 点位置所需要点的过程与其一致，但是每一个记录点的机器人姿态变化尽量大一些。		改变机器人姿态，移动机器人，以不同方向将工具末端对准参考尖点。 点击**“示教”**按钮，将当前机器人位置记录。 示教完当前位置，点击右箭头图标标定下一个点，点击左箭头图标可查看上一点。
4.当四点标定完成后，会出现**“计算”**按钮。		
5.点击**“计算”**按钮，会进入最终的计算结果显示界面。		点击**“保存”**按钮，将当前计算结果保存到指定的工具中，并返回主界面。 点击**“返回”**按钮，可不保存标定结果，返回设置界面。

若选择的不是 TCP（默认方向），则需要手动标定工具的 Z 方向或者 X 方向，以下以 TCP&Z, X 方法为例。

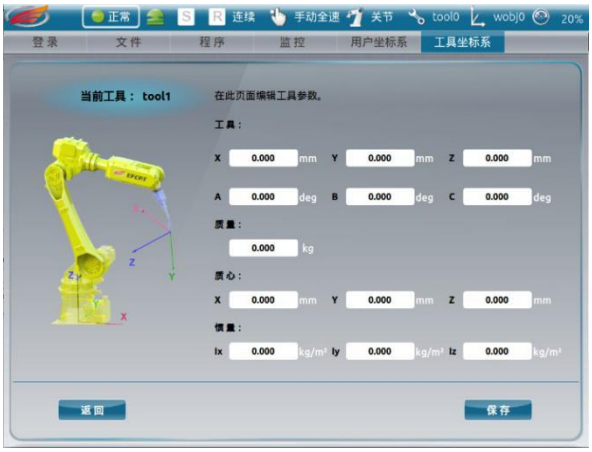
表 6-3 TCP&Z 工具标定操作流程

步骤	图示	描述

1.前四点的标定过程与 TCP（默认方向）方法过程一样。		当标定完成前四点后，**“计算”**按钮不会出现。点击右箭头，进入工具方向的标定。
2.标定工具坐标系的 Z 方向。		保持机器人姿态不变，移动机器人远离参考尖点，该方向作为工具坐标系的 Z 正方向。 点击“**示教**”按钮，将当前机器人位置记录。 示教完当前位置，点击右箭头图标标定下一个点，点击左箭头图标可查看上一点。
3. 标定工具坐标系的 X 方向。		保持机器人姿态不变，移动机器人远离参考尖点（如图所示），该方向作为工具坐标系的 X 正方向。 点击“**示教**”按钮，将当前机器人位置记录。 示教完当前位置，点击左箭头图标可查看上一点，点击“**计算**”按钮显示最终结果。
5.点击**“计算”**按钮，会进入最终的计算结果显示界面。		点击**“保存”**按钮，将当前计算结果保存到指定的工具中，并返回主界面。 点击**“返回”**按钮，可不保存标定结果，返回设置界面。

7.2.2 修改工具

表 6-4 修改工具操作流程

步骤	图示	描述
1.工具标定的设置界面，点击“**修改**”按钮，进入工具的编辑界面。		
2.在工具编辑界面输入参数并保存。		在白色的编辑框中输入工具坐标系的数值。 点击“**保存**”按钮，将当前计算结果保存到指定的工具中。 点击“**返回**”按钮，结束编辑，返回设置界面。
		机器人运行过程中，保存和激活的操作是不允许的，并出现如图提示。

7.3 用户坐标系标定

表 6-5 用户坐标系下标定方法说明

	方法说明
标定方法	用户坐标系只需要三点法进行标定，只是在点的选取上有略微不同。

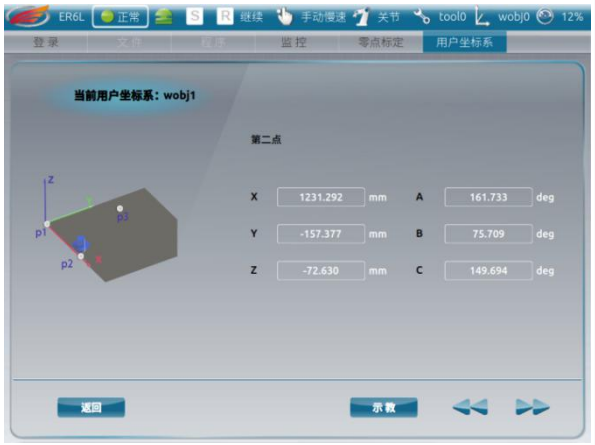
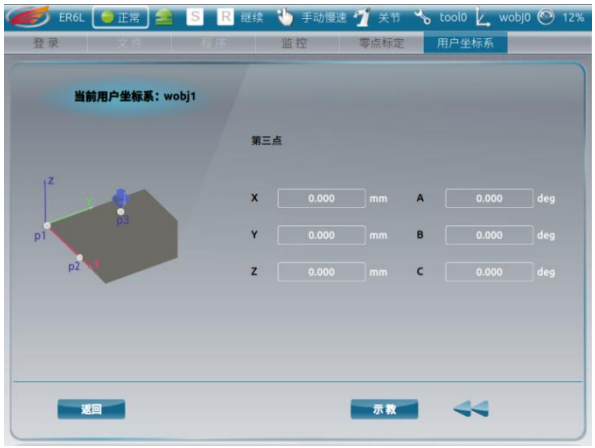
	1) 有原点：标定原点已知。记录三个点位如下： 第一点：坐标系原点； 第二点：X 轴正方向上一点； 第三点：在 Y 轴正方向的 XY 平面内任意一点。 2) 无原点：标定原点未知，通过标定计算可以得到。 第一点：X 轴上任意一点； 第二点：相对第一点，指向 X 轴正方向上一点； 第三点：Y 轴正方向上一点。

7.3.1 用户坐标系标定

这里选择“有原点”的方法标定为例。

表 6-6 用户坐标系标定操作流程

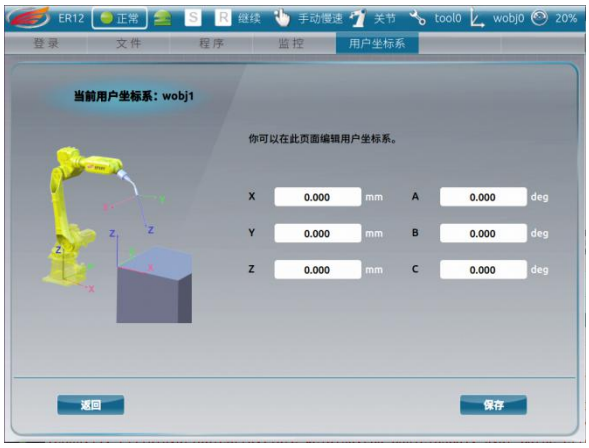
步骤	图示	描述
1. 在桌面点击**“用户坐标系”**的图标，进入用户坐标系标定的设置界面。		所有已定义的用户坐标系名称列表。 手动标定的方法，包括已知原点和未知原点两种方法。 点击**“标定”**按钮，开始进行标定。
2. 在点击**“标定”**按钮后，进入标定界面，右图所示开始标定第一点。		移动机器人至所需用户坐标系的原点位置。 点击**“示教”**按钮，将当前机器人位置记录。 示教完当前位置，示教正确完成后直接就会跳到下一点的示教界面。 若未标定完成，需要结束标定过程，点击**“返回”**按钮。

3. 标定第二点以及第三点时，其操作与标定第一点过程相同。注意标定的三点不能再一条直线上，且两点间距离至少大于 10mm。		示教完当前位置，点击右箭头图标标定下一个点，点击左箭头图标可查看上一点。
4.标定完第三点后，**“计算”**按钮会出现。”		点击**“计算”**按钮后，界面会跳转至标定经过界面。
5.标定结果界面。		点击**“保存”**按钮，将当前计算结果保存到指定的用户坐标系中。 点击**“激活”**按钮，将当前的用户坐标系设为已激活的用户坐标系。 点击**“返回”**按钮，可返回设置界面。

7.3.2 修改用户坐标系

表 6-7 修改用户坐标系操作流程

步骤	图示	描述

1. 在桌面点击“用户坐标系”的图标，进入用户坐标系标定的设置界面。		选择需要输入的用户坐标系名称。 点击**“修改”**按钮进入修改界面。
2.在用户坐标系编辑界面输入参数并保存。		在白色的编辑框中输入工具坐标系的数值。 点击**“保存”**按钮，将当前计算结果保存到指定的工具中。 点击**“返回”**按钮，结束编辑，返回设置界面。
		在机器人运行过程中，保存和激活的操作是不允许的，并出现如图提示。

第 8 章　零点恢复

8.1 本章简介

本章主要介绍 EFORT 工业机器人零点恢复功能和零点恢复的操作步骤。

8.2 零点恢复简介

零点恢复功能是指当机器人由于编码器电池停止供电或拆卸电机等非正常操作引起机器人零点丢失后，快速找回正常零点值的功能。

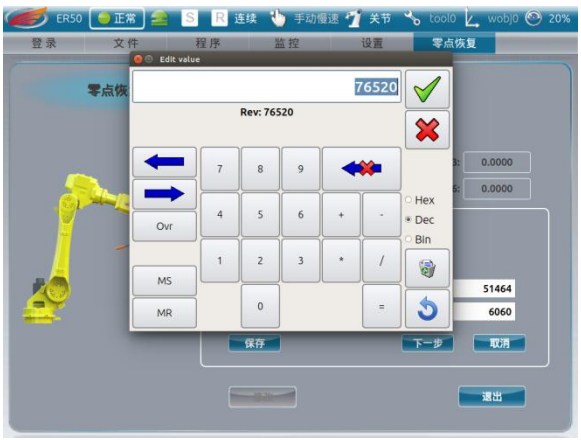
零点文件记录步骤见 DH 参数设置中零点文件记录的相关内容。

注：在使用该功能之前，需要先手动大致对齐机械零刻线，本功能是对手动对零刻线的一个修正功能，并不能直接在任意角度恢复零点。

8.3 零点恢复操作步骤

表 7-1 零点恢复操作步骤

步骤	图示	说明
1．在桌面点击**“零点恢复”**图标，进入零点恢复功能主页面。	 	计算结果是指：机器人当前位置与控制器记录的原始零点位置的差值。 点击“修改”按钮，可以进入零点文件的编码器单圈值修改界面，操作方式同第 3 步。

2．点击**“开始”**按钮，启用零点恢复功能，确认零点数据。		检测各轴的编码器单圈值是否与标准文件记录数据相同（标准文件数据出厂自带）。 如果需要修改，可以点击**“修改”**按钮进行修改。 不修改，直接跳过第3步。
3．修改编码器单圈值。	 	在编辑框中直接输入修改后的数据。然后点击**“保存”**按钮。 注意保存编码器单圈值后，零点文件中的编码器多圈值数据会清除掉。但这不影响零点恢复功能使用。

4．点击**“下一步”**按钮，进入计算页面。		点击**“计算”**按钮，开始计算结果，计算成功会有状态反馈，LED 被点亮，同时显示计算结果。
3．点击**“下一步”**按钮，进入恢复页面。		点击**“恢复”**按钮，机器人将按照计算结果自动运动。运动完成将会反馈状态，LED 灯被点亮。 操作此步骤需要将运行模式打到自动模式，且机器人处于上伺服的状态。
4.观察机械零标刻度线。		此时观察各轴机械刻线是否对齐，若出现机械刻线明显偏移得更远的情况，请手动将该轴重新对准，重复前 3 项步骤，直到所有轴的机械刻线均对齐为止。
5．点击**“下一步”**按钮，进入重置页面。		点击**“重置零位”**按钮，弹出提示框点击**“是”**，机器人将自动重置所有零点位置，重置成功将会有状态反馈，LED 灯被点亮。
6. 点击**“确认”**按钮，返回零点恢复		

页面。		

第 9 章　零点标定

9.1 本章简介

本章主要介绍标定点记录、零点计算以及零点程序生成、记录新零点的操作步骤。

9.2 零点标定说明

机器人在出厂时，会有一个唯一的零点位置，并存在机器人控制器中。如果机器人在使用过程中，出现零点丢失，通过该功能可重新标定机器人零点位置。

适用范围：更换电机、更换驱动器、零位不准、意外碰撞导致机器人零点丢失的情况下，可通过零点标定功能找回机器人零点位置。以 6 轴机器人为例，该功能只能标定出 J2、J3、J4、J5 轴零点位置。

标定方法说明：标定前检查机械各轴是否正常运动，检查传动链是否有松动，准备顶尖工装一套，分别安装到机器人末端法兰和机器人外固定装置上。安装工装时，必须保证尖点偏离 6 轴轴线。标定过程中，记录点的时候要求机器人姿态差异尽量大，最终计算的结果会比较理想。

9.3 标定点记录

表 8-1 标定点记录步骤

步骤	图片	描述
1.打开零点标定功能。		打开示教器桌面，点击零点标定功能按钮进入下图主界面。
2.点击“开始”按钮，进入点位记录界面。		

3.进行 20 个点位数据的记录。	 	需要绕着二个固定的尖点进行标定，每个尖点需要 10 种不同姿态，共需要记录 20 次。 “**点数**”加一后，点动机器人姿态变化后再对准尖点，点击一次“**记录**”按钮，依次完成两个尖点数据的记录。 点击“**点位信息**”按钮可以切换到表格模式查看所有的点位信息，再次点击，则正常切换回去。 **注意：相邻尖点之间距离应大于 300 毫米。**
4.保存 20 个点位数据，生成零点标定文件。		当 20 个点记录完成后会自动出现“**保存**”按钮，点击“**保存**”按钮进行尖点坐标的保存，系统会自动将点位数据写入到标定文件 CalibData.xml 中。

9.4 文件导出

表 8-2 标定文件导出步骤

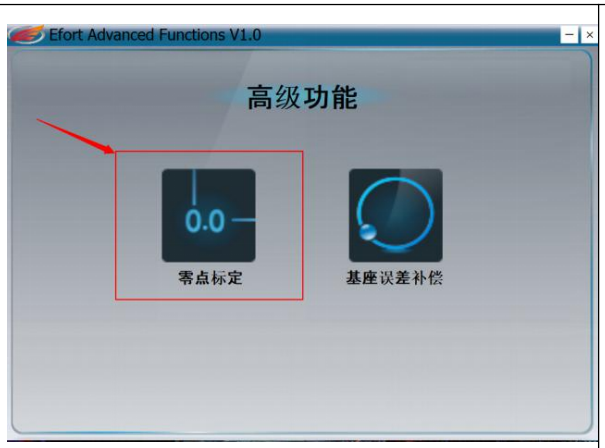
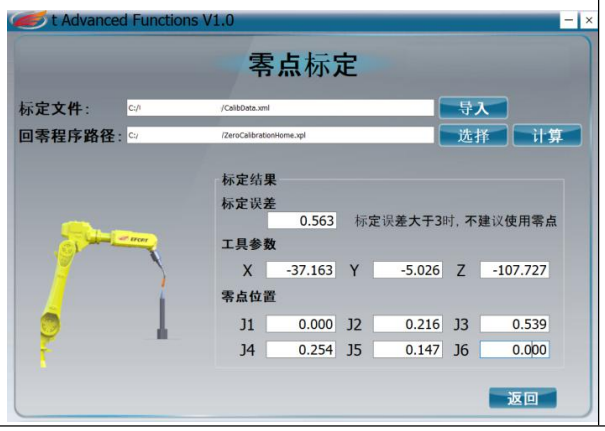
步骤	图片	描述

1. 导出点位数据用于零点计算。	 	插入 U 盘，点击**“导出”**按钮，弹出导出窗口后点击“**导出**”按钮。

9.5 零点计算及零点程序生成

表 8-3 零点计算操作步骤

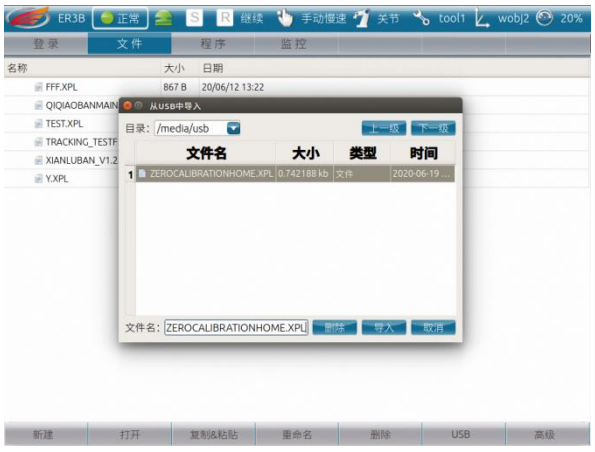
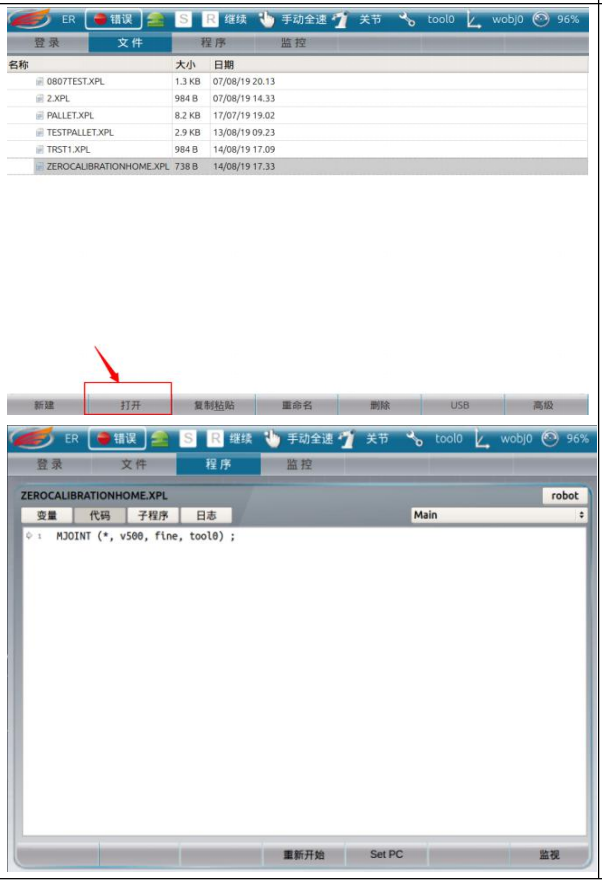
步骤	图片	描述
1.打开零点标定应用程序。		在电脑中双击打开零点计算应用程序

2.选择零点标定。		
3.零点标定计算。	 	点击“**导入**”按钮，选择 8.4 章节中导出到 U 盘的零点标定文件。 点击“**选择**”按钮，选择计算生成的回零程序文件存放路径。 点击“**计算**”按钮，计算生成标定结果和回零程序文件。 **注意：标定误差大于 3 时，不建议使用零点。**

9.6 记录新零点

表 8-4 记录新零点步骤

步骤	图片	描述

1.将 USB 中回零程序拷贝至示教器中。		进入文件界面，点击“**USB**”，选择“**从 USB**”。 选择零点计算生成的回零程序，点击“**导入**”。
2. 运行回零程序。		选择生成的回零程序，点击“**打开**”。 运行回零程序。

3.检查当前坐标位置。		点击“**监控**”按钮，选择“**位置**”，检查当前关节坐标是否为零点标定计算出来的坐标。相同说明位置正确，否则重新运行。
4.将当前位置设置成为零点坐标。	 	点击“**监控**”按钮，选择“**驱动器**”。 在重置零位列点击“**轴 x 清零**”按钮，将各轴清零。 点击“**监控**”按钮，选择“**位置**”，查看各关节坐标值，为 0 说明零点标定成功。

第 10 章　碰撞检测

10.1 本章简介

碰撞检测功能用于检测机器人发生碰撞事件。通过理论预测参数与实际参数比较，当超过一定阈值范围则认为发生了碰撞。目前，检测到机器人发生碰撞后，会第一时间发送停止命令，使机器人停下来，避免后续的碰撞。本章节主要介绍两种碰撞检测功能。

1. 基于动力学的碰撞检测，通过建立动力学模型计算各关节的理论预测力矩，与实际力矩比较，当超过一定阈值范围则认为发生了碰撞。

2. 基于最大力矩的碰撞检测，通过循环多次运行作业程序采集各轴正常运行最大力矩值保存，当实际作业中各轴力矩超过设定的阈值则认为发生了碰撞。

10.2 基于动力学的碰撞检测

10.2.1 注意事项

1.确认动力学模型是否与机器人安装方式（正装、倒装、侧装）匹配，出厂默认为正装的动力学模型。若机器人不是正装方式，请联系埃夫特技术支持人员获取非正装的动力学模型参数。

2.使用碰撞检测功能应该检查机器人末端负载参数是否配置正确，以及是否被正确激活。在使用动力学碰撞检测功能时需要配合负载辨识功能一起使用，根据实际情况在负载辨识界面对负载参数进行参数设置，否则会带来误报警，具体操作可查看负载辨识章节。

3.设置合适的碰撞检测灵敏度，值越小越灵敏，范围是 1-300%。

4.当碰撞检测开关打开后，碰撞检测功能才会生效，否则无效。另外，当需要单独对各轴灵敏度进行设置时，可以进入“高级”设置界面进行各轴基本值参数调整。当需要追求更优的灵敏度参数以及碰撞检测效果时，可以进入“高级”设置界面进行参数整定操作，特别需要注意的是，整定的效果与整定时的速度和工艺程序有关，仅在对应的速度下才具有最优的效果，一般整定的周期为 3-5 个循环。

5.阈值自整定是需保证机器人所运动程序过程中无碰撞及与外界力交互操作产生，否则将导致更新出的阈值上限偏大，检测灵敏程度降低。

6.阈值自整定过程将自动关闭所有模式下的碰撞检测功能，更新后需要手动恢复更新前的开启状态。

7.阈值更新过程的机器人运动程序轨迹及速度加速度负载等信息，需要尽可能接近实际机器人应用时的状态。

8.在应用阈值自整定结果前，建议停止当前所运行的程序。

10.2.2 参数说明

表 9-1 界面参数介绍

步骤	图示
1.打开示教器桌面，点击“碰撞检测”功能图标进入碰撞检测功能界面。	
2.碰撞检测界面	

编号 1：安装方式的设置和对应动力学模型名称的显示。

编号 2：碰撞检测的灵敏度值（1%~300%），用于设置碰撞灵敏值，其默认值为 100%。

编号 3：碰撞检测的使能开关，绿色表示开启，灰色表示关闭。

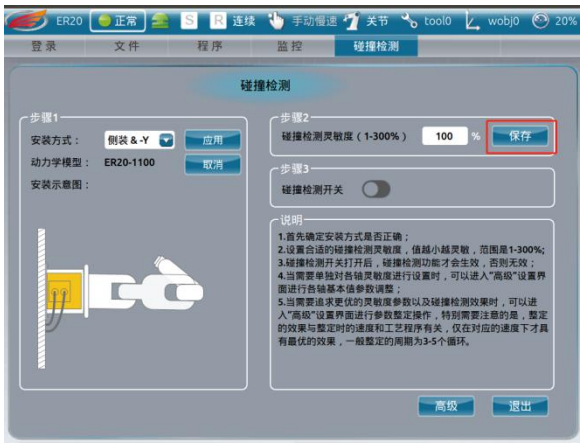
编号 4：高级功能，用来设置各轴的碰撞检测基本值以及阈值自整定等高级操作。

编号 5：点击“退出”按钮，退出碰撞检测功能界面，返回到桌面。

10.2.3 设置基本操作步骤

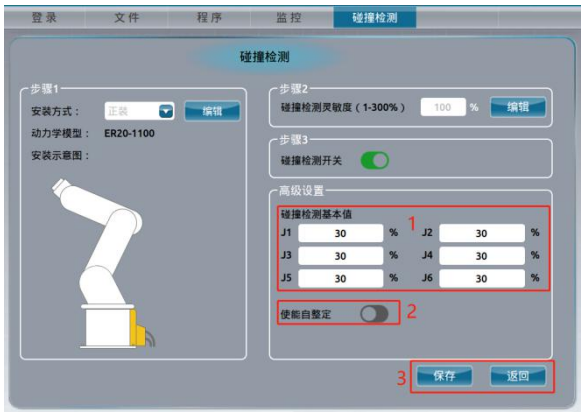
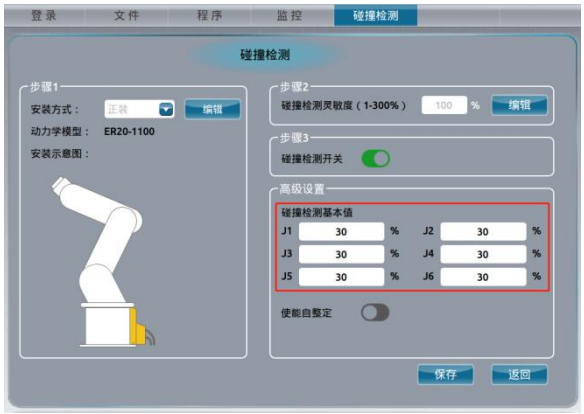
表 9-2 动力学碰撞检测操作步骤

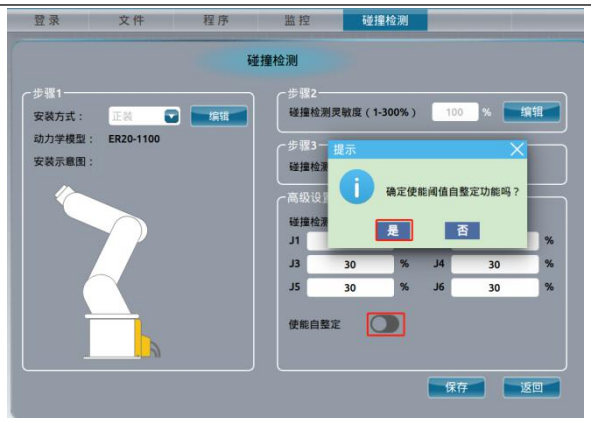
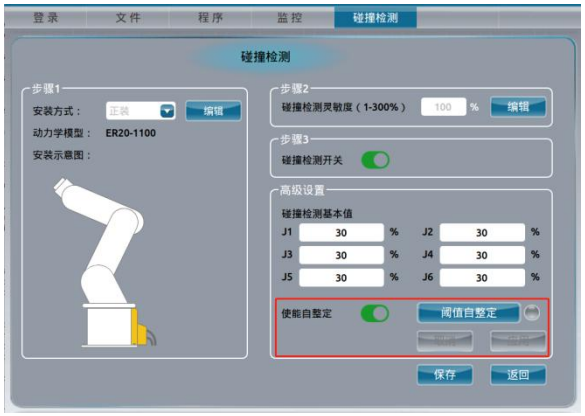
步骤	图示	说明
1.进入桌面，点击“碰撞检测”App图标，进入碰撞检测主界面。	  	1）确认机器人的安装方式，出厂默认的动力学模型为机器人正装方式下的模型。若客户现场为非正装方式，则需联系埃夫特技术支持人员获取机器人非正装的动力学模型。 2）如果现场不是默认的正装方式，在获得非正装的动力学模型的后，则点击“编辑”按钮，选择对应的安装方式，然后点击“应用”按钮进行保存。该参数需要重启后才能生效。 3）点击“取消”，则退出安装方式设置。 备注： 1、不同的安装方式都有不同安装示意图，客户可以根据示意图选择和实际现场一致的安装方式。 2、关于侧装的四个特殊方向的定义，如侧装&+X对应0°安装，即表示重力方向和世界坐标系的X正方向一致；侧装&-Y对应90°安装，即表示重力方向和世界坐标系的Y的负方向一致，即其它以此类推。

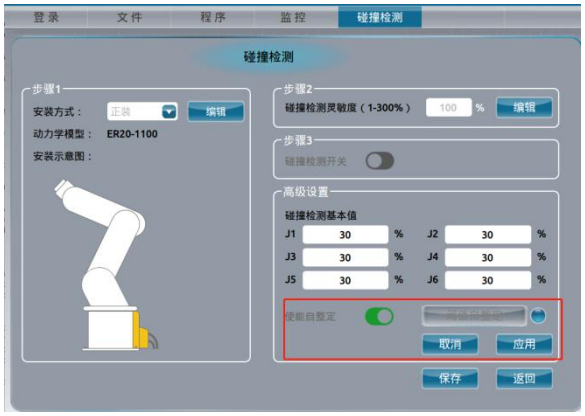
2.设置碰撞灵敏度	 	1）点击“编辑”按钮后才可以对碰撞灵敏度进行修改。 2）修改完成后，点击“保存”后才能生效。
3.打开碰撞检测开关	 	1）点击开关按钮后，弹出提示框，点击“是”打开碰撞检测功能，状态显示绿色，点击“否”取消该操作。 2）当需要关闭时，再次点击该按钮，弹出提示，点击“是”关闭碰撞检测功能，状态显示灰色，点击“否”取消该操作。

10.2.4 高级设置操作步骤

表 9-3 动力学碰撞检测操作步骤

步骤	图示	说明
1.进入桌面，点击“碰撞检测”App图标，进入碰撞检测主界面。点击“高级”	 	1）进入主界面后，点击“高级”即可进行高级设置界面。 2）主要分三部分：第一个部分是各轴的独立的灵敏度修改，值越小越灵敏，范围是 1-100%；第二部分是使能自整定阈值功能选择；第三部分是参数保存和返回主界面
2、设置碰撞基本值		1）进入高级主界面后，当需要单独对各轴灵敏度进行设置时，可在碰撞检测基本值中进行修改设置。修改完成后点击“保存”，根据提示，点击“是”保存数据，点击“否”则返回主界面。

		
3、设置自整定阈值功能	 	1）进入高级主界面后，当需要根据当前工艺速度下的运行程序阈值自整定设置时，可点击使能自整定开关。根据弹出提示，点击“是”进行打开该功能。

3、进行自整定操作	 	1）关闭碰撞检测开关后，才可点击“阈值自整定”按钮，否则会有弹出提示，用户根据弹框提示，可强制关闭碰撞检测开关。 2）点击“阈值自整定”按钮后，状态指示灯显示蓝色，表示正在整定参数。，当工艺程序循环运行 3-5 次后，可直接点击“应用”按钮直接使得整定的参数生效。点击“取消”则不使用本次整定的参数值。 3）当仅仅只使用了整定参数功能没有修改各轴的碰撞检测基本值，可以不进行保存操作，可直接点击“返回”按钮，返回主界面。 备注： 当需要追求更优的灵敏度参数以及碰撞检测效果时，可以进入“高级”设置界面进行参数整定操作，特别需要注意的是，整定的效果与整定时的速度和工艺程序有关，仅在对应的速度下才具有最优的效果，一般整定的周期为 3-5 个循环。周期数越大整定效果也越好。

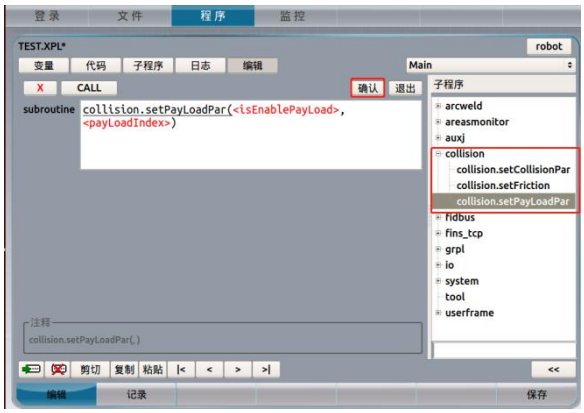
10.2.5 RPL 程序设置碰撞检测参数

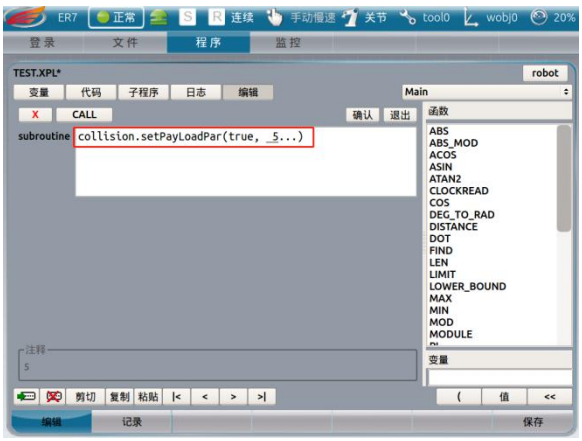
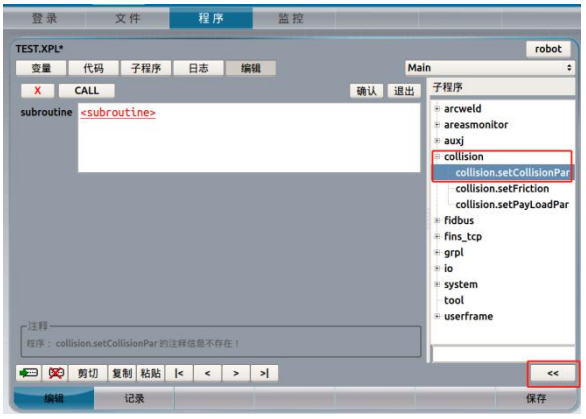
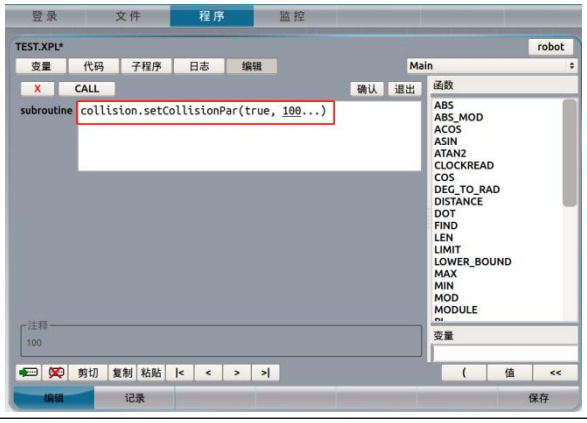
表 9-4 RPL 程序指令介绍

指令	名称	功能
Collision.SetCollisionPar(enableCollision,collisionSensitivity)	碰撞检测设置参数指令	实现碰撞检测功能的开启和碰撞检测的灵敏度设置
collision.setFriction（startUpdating，	摩擦更新指令函数	指令中第 1 个输入参数是用来开启

updateInertia）		或关闭摩擦更新的（开启为 true），第 2 个输入参数是用来确定更新摩擦时是否同步更新关节惯量（同步更新惯量为 true）
collision.setPayLoadPar(isEnablePayLoad,payLoadIndex)	碰撞检测负载参数设置函数	其中第 1 个参数激活当前负载，第 2 个参数为当前负载号。

表 9-5 RPL 程序碰撞检测参数操作步骤

步骤	图示	说明
1.新建或打开 RPL 程序		
2．插入 call 指令，在子程序中调用负载辨识指令：collision.setPayLoadPar（isEnablePayLoad,payLoadIndex）	 	其中第 1 个参数激活当前负载，第 2 个参数为当前负载号。 本例中是执行完该指令后表示激活 5 号负载参数

		
3．插入 call 指令，找到碰撞检测参数设置指令：collision.SetCollisionPar(isEableCollisionDetect,collisionSensitive)	  	其中第 1 个参数为 true 时，启动碰撞检测，为 false 时，不启用；第 2 个参数值设置范围为 1~200%。 备注： 1、该指令参数 100 表示设置的灵敏度是 100% 2、总的灵敏度=界面设置值*指令设置值*各关节独立设置灵敏值。

4．保存程序测试运行即可。		备注： 本例仅仅是指令调用的说明，用户可以根据实际工艺情况，多次调用碰撞检测指令，对其进行激活和关闭操作。

10.2.6 碰撞误报警原因及解决方案

表 9-6　碰撞误报警原因及其解决方案

异常情况：	解决措施：
1. 出厂未进行摩擦更新，本体实际摩擦参数与配置摩擦参数相差较大	进行摩擦更新，修正摩擦参数
2. 负载信息设置不正确	进行负载辨识，或根据负载模型评估负载参数
3. 更换负载后未进行充分阈值整定或者默认阈值余量设置过小	调大检测阈值，或进行阈值整定
4. 阈值设置过小	调大检测阈值
5. 速度倍率突变的速度调节	速度逐步调节，或调大检测阈值速度调整好后，恢复阈值
6. 机器人安装方式设置错误	设置正确的安装方式，或关闭碰撞检测
7. 机械零位偏差较大	恢复零位，或调大检测阈值或关闭碰撞检测
8. 负载质量或者惯量非常大	调大检测阈值
9. 奇异点附近轴高速回转动作	关闭碰撞检测或调大检测阈值
10. 应用过程中机器人或工具与外部设备有过多的接触，有交互力产生	按 30%为步长，逐步提高检测阈值，直到不再出现误报警

10.3 基于最大力矩的碰撞检测

10.3.1 设置步骤

表 9-7 基于最大力矩的碰撞检测操作步骤

步骤	图片	描述
1、进入桌面，点击“碰撞检测”功能，进入碰撞检测主界面。		根据机型自动加载不同的碰撞检测模型进行设置，这里可以查看原设置的各个参数。
2 点击屏幕右上方碰撞检测使能右侧开关。		开关绿色表示碰撞检测处于使能状态，灰色表示使能关闭。开启碰撞检测则打开使能开关。
3、点击“开始测试”按钮，测试各轴最大力矩。		点击**“开始测试”**按钮后，按钮右侧指示灯处于点亮状态，表示正在测试，按钮变为**“完成测试”**。 运行机器人生产程序 5~10 遍，各轴在运行过程中的最大扭矩会采集保存。

4、点击“完成测试”按钮，结束各轴最大力矩测试。		点击**“完成测试”**按钮后，按钮右侧指示灯处于灰色状态，表示完成测试，按钮变为**“开始测试”**。 采集到的测试最大力矩值会显示在界面左侧部分。点击**“重置力矩”**按钮可将所有最大力矩值清零。
5、设置各轴力矩灵敏度。		可对各轴力矩百分比编辑框点击弹出键盘进行设置，设置范围为 1-100； 设置完成后点击**“保存”**按钮，将测试和配置的数据进行写文件保存，并发送至控制器中。 开启碰撞检测使能后进行作业会检测各轴实际力矩，当检测到当前实际运行力矩大于等于设置的最大力矩*(2-灵敏度/100)后认为发生碰撞，机器人停止并报警。

第 11 章　安全监控

11.1 本章简介

本章主要介绍区域监控设置、激活、区域违反报警后恢复、编程实现区域监控功能、IO 控制激活使用以及安全位置激活使用。

11.2 功能简介

区域监控是以机器人 TCP（TCP，工具中心点）为参考点，限制 TCP 点（或者 TCP 设定区域）在指定区域内工作或者禁止进入指定区域，也可以设置信号区域和多台机器人共享区域的功能。此功能可以有效的保护机器人或者现场设备。当 TCP 将要离开允许工作区域或进入禁止区域时，提前给出报警，停止机器人运动。

11.3 区域监控

11.3.1 区域监控设置

表 10-1 区域监控设置操作步骤

步骤	图示	说明
1.打开安全监控有两大部分，区域监控和安全位置，首先说明区域监控功能。	 	打开示教器桌面，点击安全监控功能图标。 进入安全监控界面，勾选区域监控上方**“激活”**复选框，点击区域监控图标。 进入区域监控主界面。
主界面功能介绍		

IO 控制选择框:勾选后，通过 IO 信号控制区域的监视和控制功能，在使用 IO 控制的时候，程序中指令不起作用，App 禁止修改区域监视和控制。

区域名称：目前支持设置八个区域，八个区域可单独或者多个一起工作。

区域类型：显示当前区域设置的类型，包括：

工作区：TCP 只能在区域内运动；
禁止区：TCP 不能进入此区域；
共享区：TCP 在区域内发出信号，区域外取消信号。

状态：显示机器人 TCP 端和区域之间的关系。

图标	说明
	不在监视
	在工作区内，且不违反
	在工作区外，且违反
	在禁止区外，且不违反
	在禁止区内，且违反

注：共享区只显示在区域内还是区域外。不存在违反情况。

设置：进行监控区域和 Tcp 区域设置界面。

监控：包括监控开关按钮。按钮绿色表示监控功能打开。监控功能打开，则只可以看到机器人 Tcp 与机器人的状态，但是机器人违反后不报警，不停止。

控制：包括控制开关按钮。按钮绿色表示控制功能打开。控制功能打开，则机器人违反后会立即报警且停止运动。如果需要打开控制功能，必须先打开监控功能。

当区域为共享区时，控制开关决定是否进入被占用的共享区域。（当共享区被占用，打开控制开关，若机器人仍然向共享区运动则报警停止）

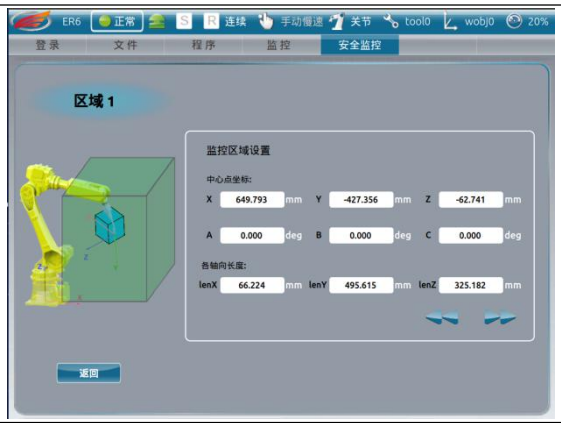
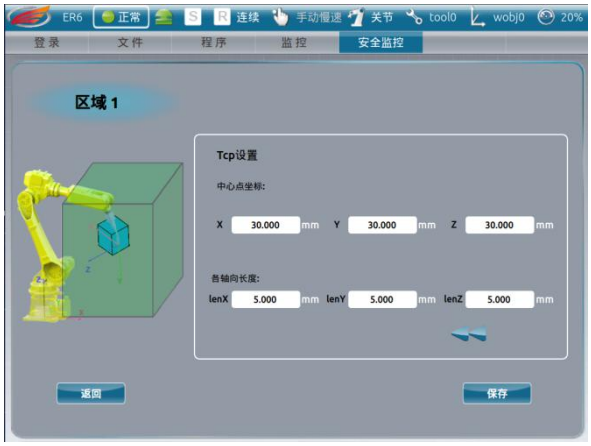
区域违反指示灯：绿色表示不违反，红色表示违反；

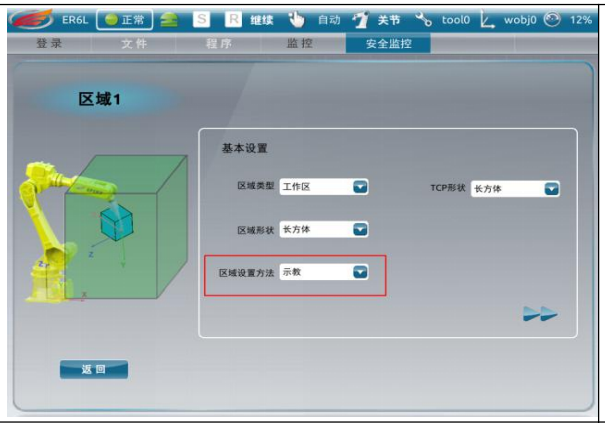
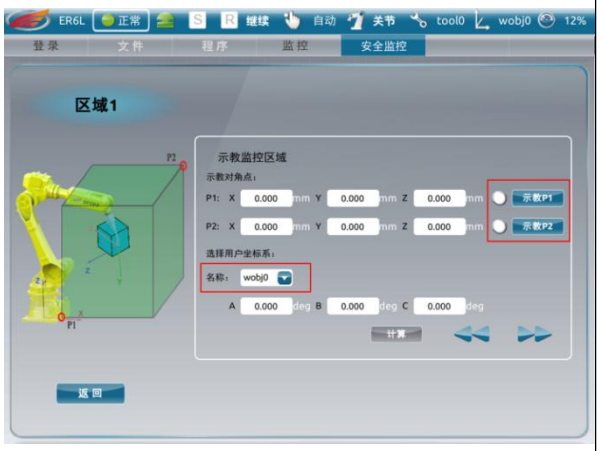
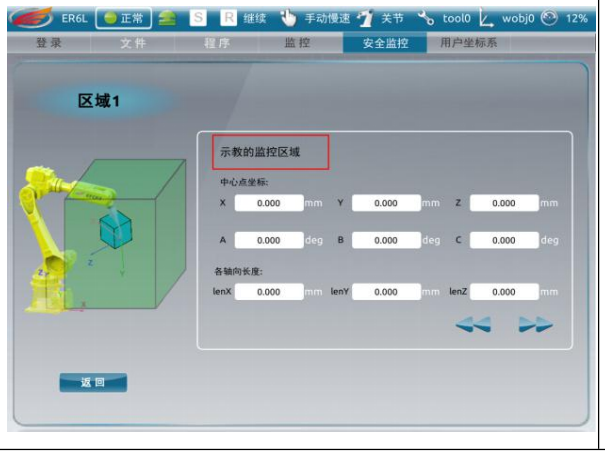
多个工作区时候，不在任何一个工作区，指示灯为红色；

多个禁止区，只要单个区域违反，指示灯为红色；

禁止区和工作区同时存在，进入了某一个禁止区，指示灯为红色，不在任何一个工作区，指示灯为红色。

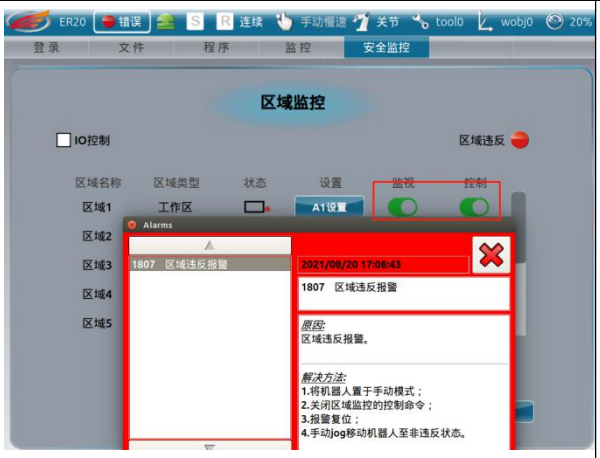
2.设置区域监控数据。		在界面中选择需要设置区域，点击对应的设置按钮，进入区域设置。以设置区域 1 为例。

3.基本信息设置。区域设置方法有两种，编辑和示教两种方法，首先说明编辑方法。		区域类型选择**工作区、禁止区、共享区**。 区域形状目前支持长方体。 Tcp 形状可以设置为长方体和球体。 **注：当选择共享区时，需要设置占用输入输出信号。设置内容可以在：桌面->IO 设置 APP->功能 IO 配置中完成。**
4.监控区域设置		设置监控区域的**中心点**位置及区域姿态。 设置长方体边长。 设置完点击“**向右箭头**”图标进入下一页。 **注：监控区域中心点坐标是相对于机器人基坐标进行设置的。**
5.TCP 设置		设置 TCP 区域的**中心点**位置及 TCP 区域姿态。 TCP 形状为长方体设置边长，球体则设置半径。 设置完成点击保存，将返回首页。 **注：TCP 区域是指左图中用长方体或球体将机器人末端工具包络起来的一个蓝色区域。TCP 中心点坐标是相对于机器人法兰末端进行设置的。**

6.说明区域设置的示教方法。		示教方法的基本设置与编辑方法相同，将区域设置方法的下拉框选择为**“示教”**。 点击右下角**“向右箭头”**图标。
7.示教监控区域。	 	将确保周围环境安全的情况下，将机器人末端法兰中心移动到待监控区域的一个顶点进行示教。 再移动到该点的对角点进行示教。 在选择用户坐标系下拉框中选择设定好的坐标系。（到用户坐标系 APP 中标定一个坐标系，要求标定的 X 轴、Y 轴与监控区域底面的长、宽边平行）。 完成两个对角点的示教和用户坐标系的选择后，点击**“计算”**按钮。 完成监控区域位置信息的计算。
8.示教的监控区域显示与编辑。		上一步点击计算按钮后在左图显示计算的结果。 示教的监控区域中心点坐标与各轴向长度可点击修改设置。 点击右下角**“向右箭头”**按钮可切换到 Tcp 设置界面，此界面设置与编辑方法相同。

11.3.2 区域违反报警后恢复

表 10-2 区域违反报警后恢复操作步骤

步骤	图示	说明
1.确定违反区域。		区域违反指示灯变红色、说明出现区域违反报警，打开控制使能后，此时上伺服将会弹出抱紧弹框，可清除弹框，再次上伺服仍会弹出。
2.取消控制功能，清除报警。		1. 关闭控制使能。 2. 清除报警弹框，此时再上伺服不会弹出报警。 **3.务必在手动模式下运动机器人，将机器人移动到工作区内或者禁止区外，才能使区域违反状态等变为绿色。**

11.3.3 IO 控制激活使用

表 10-3 IO 控制激活使用操作步骤

步骤	图示	说明
1.激活 IO 控制功能。		勾选后，通过 IO 信号控制区域的监视和控制功能，在使用 IO 控制的时候，程序中指令不起作用，App 禁止修改区域监视和控制。 IO 口的配置与使用见 IO 设置中功能 IO 相关内容。

2.激活 IO 控制功能后，通过 IO 设置中配置的相关 IO 口控制区域监控监视与控制按钮的开关，并通过区域监控界面观察该区域机器人的状态。		通过 IO 口给予监视和控制输入信号，可以打开相应的监视和控制开关。 监视打开后可以看见该区域机器人的状态。 当该区域为共享区时，若该区域已经被另一台机器人占用，则我们会在相应的占用输入 IO 口收到另一台机器人占用输出的信号，当前机器人不可进入共享区，等待占用输入信号关闭才可进入。 若当前共享区无其他机器人，则当前机器人可进入该共享区，机器人进入该共享区后，会在相应的占用输出 IO 口输出信号，告知其他机器人该区域已被占用，不可进入。（**注意：该占用输出信号不可强制，此强制权限已禁用**）。 **机器人进出共享区与外界有信号交互，机器人进出工作区与禁止区无信号交互。且当监视未使能而给予控制信号使其使能时，此时该功能不生效，会出现提示语告知。**

11.4 安全位置

11.4.1 安全位置激活使用

表 10-5 安全位置激活使用操作步骤

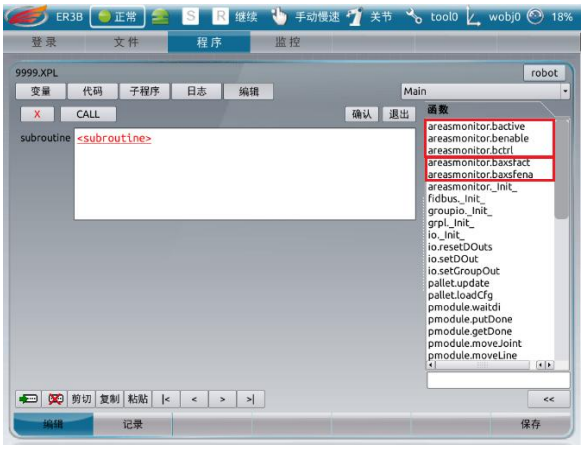
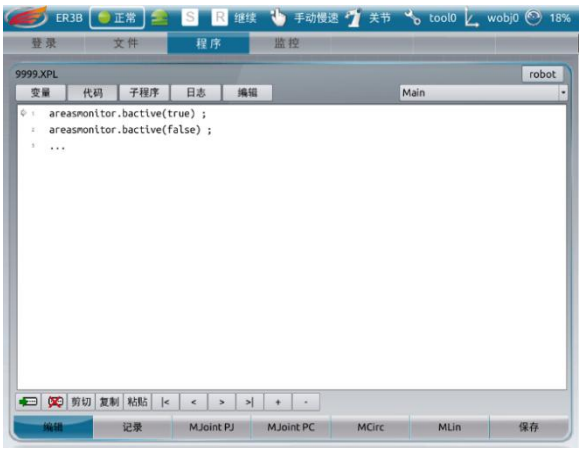
步骤	图示	说明

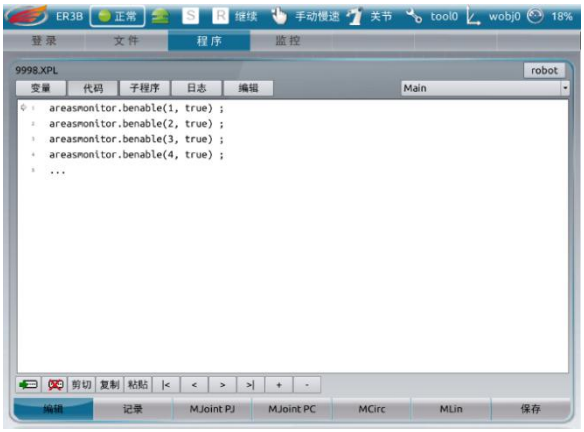
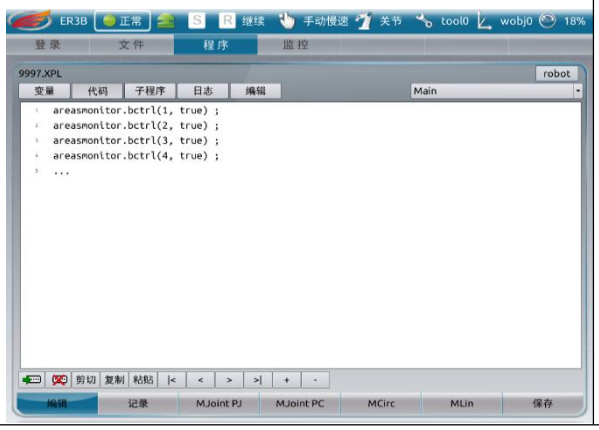
1.激活安全位置功能，进入安全位置界面。	 	打开示教器桌面，点击安全监控功能图标。 进入安全监控界面，勾选安全位置上方**“激活”**复选框，点击安全位置图标。 进入安全位置界面。
2.配置安全位置信息。	 	进入 IO 配置 APP 内选择 IO 自由配置，在输出界面找到八个安全位置，为其配置端口。 打开使能列需要配置安全位置的开关。 点击位置列的一个**“黑边圆形”**按钮。 点击**“编辑”**按钮，可以直接对**各关节坐标**数值进行设置，或将机器人移动至安全位置再点击**“示教”**按钮，并设置**允许误差值**，再点击**“保存”**按钮。 **说明：**机器人各关节坐标都在安全位置关节坐标值加减允许误差值范围内则状态灯显示绿色并在指定端口输出信号，否则

		显示为红色无信号输出。（**注意：八个安全位置的输出信号不可被强制，此强制权限已被禁用**）

11.5 安全监控编程操作

表 10-4 编程实现安全监控功能操作步骤

步骤	图示	说明
1.添加安全监控指令。		五条不同指令代码完成不同功能的打开和关闭。 注：areasmonitor_Init_是系统自带初始化函数，编程中无需使用。
2.通过编程添加不同指令实现不同使能开关功能。		“areasmonitor.bactive()”功能是区域监控功能使能开关，参数为“true”打开，“false”时关闭。 “areasmonitor.benable()”和 areasmonitor.bctrl()可以控制区域监控中监视和控制的开关，其中第一个参数为 1-8，选择八个区域中的某个区域，第二个参数为“true”时打开，为“false”时关闭。 “areasmonitor.bactive()”功能是安全位置功能使能开关，参数为“true”

	 	打开，“false”时关闭。 “areasmonitor.baxsfena()”可以安全位置的开关，其中第一个参数为1-8，选择八个安全位置中的一个，第二个参数为“true”时打开，为“false”时关闭。

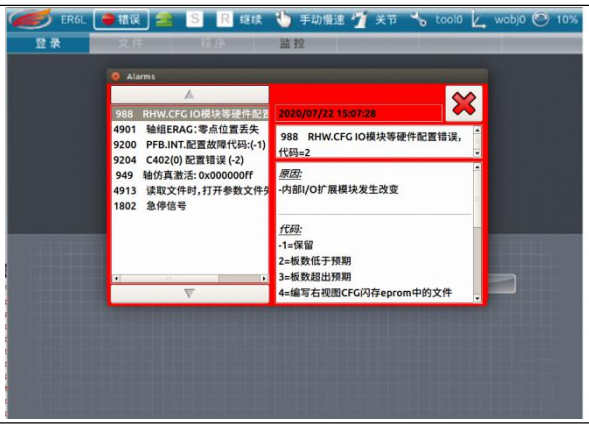
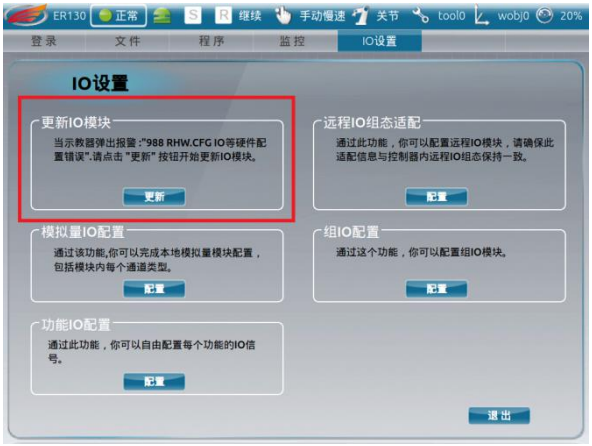
第 12 章　IO 设置

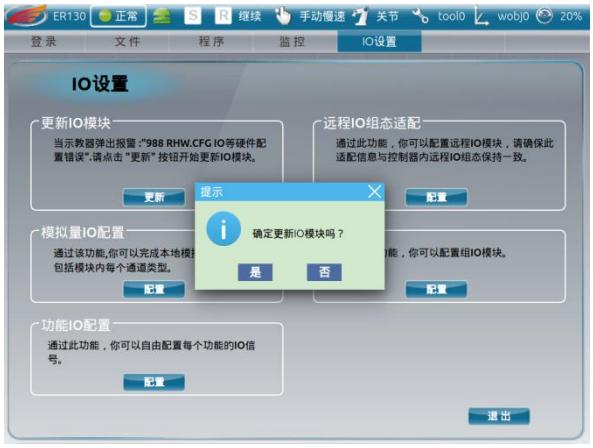
12.1 本章简介

本章主要介绍更新 IO 模块、远程 IO 组态适配、模拟量 IO 配置、组 IO 配置、功能 IO 配置的操作步骤，相关 IO 的监测需要到监控-IO 部分查看。

12.2 更新 IO 模块

表 11-1 更新 IO 模块操作步骤

步骤	图示	说明
1.硬件实际 IO 数量与预设 IO 数量不匹配，示教器会弹出报警。		″988　RHW.CFG IO 模块等硬件配置错误″出现该报警，将报警框隐藏，然后通过更新 IO 模块来清除报警。
2. 进入 IO 设置 APP，选择更新 IO 模块功能，点击“**更新**”按钮。	 	

3.确定更新 IO 模块后，点击“**是**”按钮，重启机器人。		机器人重启过程中，示教器界面不可操作。待控制器完全启动后，示教器可正常操作。

12.3 远程 IO 配置

目前埃夫特支持埃夫特自主 IO 模块和汇川的远程 IO 模块。

12.3.1 埃夫特远程 IO 模块

表 11-2 埃夫特远程 IO 配置操作步骤

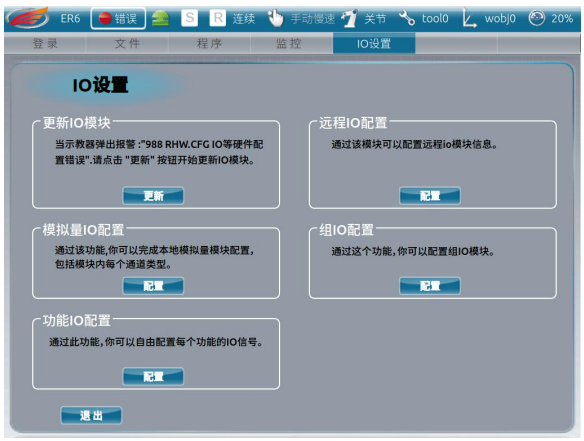
步骤	图示	说明
1.打开远程 IO 配置界面。		打开示教器桌面，点击“**IO 设置**”图标。 选择“远程 IO 组态适配”，点击“**配置**”按钮进入配置界面。
2.选择埃夫特 IO 界面。		选择埃夫特选项，进入埃夫特远程 IO 设置的界面。 注意：应用埃夫特远程 IO 需要先到总线设置的 EtherCat 设置中将对应的远程 IO 模块开启，方可正常使用。

3.进行远程 IO 模块配置。		总线设置的 EtherCat 设置中可以看到 IO 模块信息，在此地方设置模块数量不能超过总线中配置的模块。 使能：模块的使能信号，不使能则不可以使用。 地址：总线设置的 EtherCat 设置保存一致。 DI/DO/AI/AO 数量：根据总线设置的 EtherCat 设置的模块对应设置通道数量。 注：目前埃夫特 IO 有两种：数字量和模拟量。数字量 16DI 16DO，模拟量 4AI 4AO。
4.设置模拟量通道类型。		当模拟量输入或者输出大于 0 时，可以点击“**通道设置**”按钮，配置输入和输出的通道类型和分辨率，（**分辨率值建议使用默认，如果满足不了再修改**）点击确定后，点击保存按钮。

12.3.2 汇川远程 IO 模块

表 11-3 汇川远程 IO 配置操作步骤

步骤	图示	说明

1.打开远程 IO 配置界面。		打开示教器桌面，点击**“IO 设置”**图标。 选择“远程 IO 组态适配”，点击**“配置”**按钮进入配置界面。
2.选择汇川 IO 界面。		选择汇川选项，进入适配汇川远程 IO 的界面。
3.进行远程 IO 模块配置。		首先点击密码输入框，输入**“1975”**，再点击“**进入**”按钮。错误输入密码无法进行远程适配。 设置配置**模块数量**，每一个设置的模块都需要完成配置。 选择**模块类型**，包括数字信号和模拟信号的输入和输出。 设置各**模块地址**，注意不同模块占用地址长度，不可设置已被占用的地址。 当选择模块为 AM600-4AD 或 AM600-4DA 模块时，需要选择四个**通道的类型**。

		点击**“保存”**按钮，将远程 IO 配置信息保存。

远程 IO 配置说明：

模块数量：每一个模块对应的内容都需要设置，否则无法保存。

类型：共有六种类型，数字信号的输入与输出，各自包括 16 位和 32 位。模拟信号的输入和输出。

地址：地址范围为 300-500，不同模块占用地址长度不同，且不能配置已被占用地址。

AM600-0016XXX(16DO)：数字信号 16 位输出，占用 1 位地址

AM600-0032XXX(32DO): 数字信号 32 位输出，占用 2 位地址

AM600-0016END(16DI): 数字信号 16 位输入，占用 1 位地址

AM600-0032END(32DI)：数字信号 32 位输入，占用 2 位地址

AM600-4AD(4AI): 模拟信号输入，占用 4 位地址

AM600-4DA(4AO)：模拟信号输出，占用 4 位地址

通道：当选择模块为 AD 或 DA 时，需要对通道值的类型进行选择，AD 模块的通道类型有 7 种选择，DA 模块的通道类型有 6 种选择。

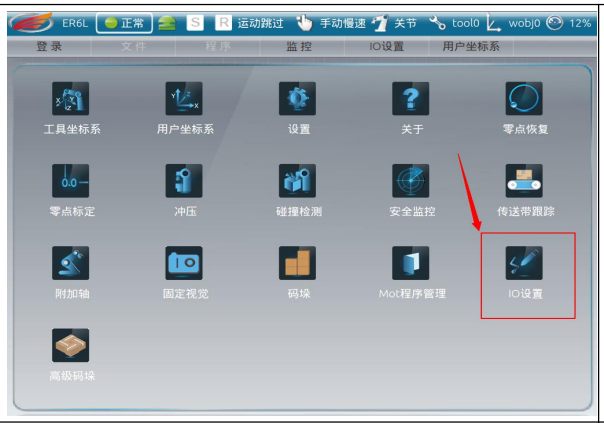
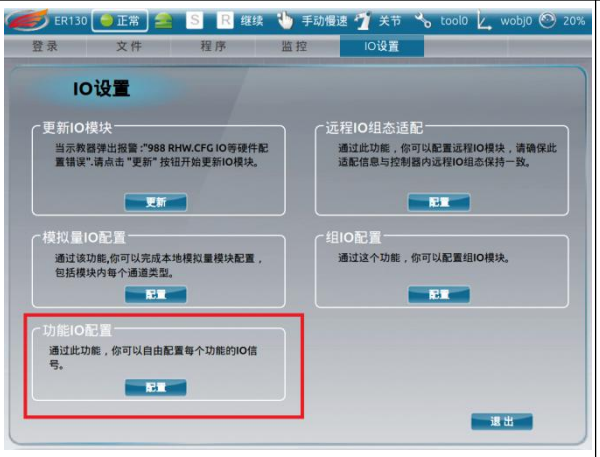
使能：远程 IO 配置信息是否生效的开关，勾上“使能”开关配置信息才能生效。

12.4 功能 IO 配置

功能 IO 配置模块主要包括七个功能：通用功能、安全监控、附加轴、冲压、高级码垛、弧焊、程序预约；每个功能目前包括输入 IO 和输出 IO。通过选择具体的功能，用户可以自由配置信号的地址，有效值等信息。

表 11-4 功能 IO 配置操作步骤

步骤	图示	说明

1.进入 IO 设置界面。		打开示教器桌面，点击“**IO 设置**”。
2.进入功能 IO 配置界面。		选择“IO 自由配置”功能。 点击“**配置**”按钮进入配置界面。
3.功能 IO 界面。		目前包括通用、安全监控、附加轴、冲压、高级码垛、弧焊、程序预约七个功能选项。 点击功能按钮即可进入相应功能的 IO 自由配置界面。 此处以通用功能为例进行说明。
4.进入通用功能的 IO 配置界面。		点击“**通用功能**”按钮进入通用功能的 IO 配置界面。 此前配置的输入输出 IO 配置信息，可以在这里查看，包括信号地址，信号有效值类型，输入信号的滤波时间，输入输出信号的当前状态。

		
5.进行 IO 配置。	 	点击“**编辑**”按钮，将启用编辑功能，编辑模式下不能实时刷新显示 IO 状态。 编辑完成后点击“**保存**”按钮，保存设置的 IO 配置信息。 点击“**退出**”按钮，返回配置主页面。

IO 自由配置说明：

输入界面：

1、描述-输入：

通用功能

序号	描述	说明	检测信号	操作模式
1	伺服	控制机器人伺服开关	脉冲信号	自动模式有效
2	启动	程序从当前行开始运行	脉冲信号	自动模式有效
3	暂停	程序暂停运行	脉冲信号	自动模式有效

4	报警复位	清除当前报警信息	脉冲信号	自动/手动有效
5	急停信号 1/2	控制机器人紧急停止开关	高低电平	自动/手动有效
6	外部报警信号	外部设备发送的报警信号	高低电平	自动/手动有效
7	全局速度+/-5	控制机器人速度加减 5	脉冲信号	自动/手动有效
8	重新开始	程序指针会返回至第一行	脉冲信号	自动模式有效
9	加载程序	加载设定程序，程序 1 到程序 4 信号名按顺序组成数字为名称设定的程序。	脉冲信号	自动模式有效
10	程序设置二进制位 1/2/3/4	程序 4 到 1 信号的状态按顺序组成四位二进制数，程序 4 在最高位，程序 1 在最低位。例如程序 4 到 1 的状态为 0、1、0、1，则组成的二进制数为 0101，对应十进制数为 5，则加载程序文件名为 5；	高低电平	自动模式有效
11	远程伺服确认	通过给信号代替手动按伺服确认按钮的作用。	脉冲信号	自动模式有效

安全监控

序号	描述	说明	检测信号	操作模式
1	区域监控使能	控制区域监控使能开关	高低电平	自动/手动有效
2	A1 监视激活 A2 监视激活 A3 监视激活 A4 监视激活 A5 监视激活 A6 监视激活 A7 监视激活 A8 监视激活	控制区域 1 到 8 的监视开关	高低电平	自动/手动有效
3	A1 控制使能 A2 控制使能 A3 控制使能 A4 控制使能 A5 控制使能 A6 控制使能 A7 控制使能 A8 控制使能	控制区域 1 到 8 的控制开关	高低电平	自动/手动有效
4	A1 占用输入 A2 占用输入 A3 占用输入 A4 占用输入 A5 占用输入	区域 1 到 8 的占用输入信号，**当共享区外机器人接收到占用输入信号，此时机器人立即停止等待，直至占用输入信号消失，机器人继续运动。**	高低电平	自动/手动有效

	A6 占用输入 A7 占用输入 A8 占用输入			

附加轴

序号	描述	说明	检测信号	操作模式
1	附加轴 1/2/3/4 步进信号 1（+）	发送给机器人的运动方向信号，附加轴 1/2/3/4 按正方向运行。例如给附加轴 1 步进信号 1（+），则附加轴 1 按正方向运动。	高低电平	手动有效
2	附加轴 1/2/3/4 步进信号 2（-）	发送给机器人的运动方向信号，附加轴 1/2/3/4 按负方向运行。例如给附加轴 1 步进信号 2（-），则附加轴 1 按负方向运动。	高低电平	手动有效

冲压

序号	描述	说明	检测信号	操作模式
1	工具 1 有件	检测工具 1 是否有料	高低电平	自动模式有效
2	工具 2 有件	检测工具 2 是否有料	高低电平	自动模式有效
3	取料允许	允许机器人取料	高低电平	自动模式有效
4	放料允许	允许机器人放料	高低电平	自动模式有效
5	前站上死点	前一台冲床上死点（最高点）	高低电平	自动/手动有效
6	本站上死点	本台机器人上死点（最高点）	高低电平	自动/手动有效
7	冲床急停	冲床急停信号	高低电平	自动模式有效
8	双张检测	双张料片检测	高低电平	自动模式有效
9	冲床单次模式	冲床冲一次检测	高低电平	自动模式有效
10	等待码垛	等待允许机器人码垛	高低电平	自动模式有效
11	等待拆垛	等待允许机器人拆 垛	高低电平	自动模式有效
12	传送带有料	传送带夹具有料检测	高低电平	自动模式有效
13	垛盘 1 有料	拆垛盘有料检测	高低电平	自动模式有效
14	垛盘 2 有料	拆垛盘有料检测	高低电平	自动模式有效
15	垛盘 3 有料	拆垛盘有料检测	高低电平	自动模式有效
16	寻料有料	机器人寻到料信号检测	高低电平	自动模式有效
17	寻料确认 1	开始寻料确认信号	高低电平	自动模式有效
18	寻料确认 2	开始寻料确认信号	高低电平	自动模式有效
19	寻料确认 3	开始寻料确认信号	高低电平	自动模式有效
20	传送带夹具有料	传送带夹具有料	高低电平	自动模式有效

高级码垛

序号	描述	说明	检测信号	操作模式
1	一/二/三/四号垛位来	一/二/三/四号垛位上对	高低电平	自动/手动有效

	料信号	应传送带上有待码放工件到达信号		
2	一/二/三/四号垛位准备信号	一/二/三/四号垛位准备码放工件信号	高低电平	自动/手动有效
3	一/二/三/四号夹持区松开反馈	一/二/三/四号夹持区松开反馈给机器人的信号	高低电平	自动/手动有效
4	一/二/三/四号夹持区闭合反馈	一/二/三/四号夹持区闭合反馈给机器人的信号	高低电平	自动/手动有效

弧焊

序号	描述	说明	检测信号	操作模式
1	起弧成功	起弧是否成功	高低电平	自动/手动有效
2	寻位成功	在寻位应用中，接收焊机反馈的寻位成功信号。	脉冲信号	自动/手动有效
3	焊机准备	焊机是否准备好	高低电平	自动/手动有效
4	碰撞检测	有无碰撞，通常焊枪防碰撞器为常闭触点，故在配置时候，通常有效值为 0。	高低电平	自动/手动有效

弧焊（模拟量）

序号	描述	说明	操作模式
1	焊机反馈电流	焊机实时反馈电流数据	自动/手动有效
2	焊机反馈电压	焊机实时反馈电压数据	自动/手动有效

程序预约

序号	描述	说明	检测信号	操作模式
1-4	程序序号设置位 1-4	当程序预约的模式为单独，四个信号分别对应程序 1/2/3/4。此时当前程序为非预约且不在运行中，接收到此脉冲信号，程序会预约上，无需确认。 当程序预约的模式为二进制，程序 4 到 1 信号的状态按顺序组成四位二进制数,程序 4 在最高位，程序 1 在最低位。例如程序 4 到 1 的状态为 0、1、0、1，则组成的二进制数为 0101,对应十进制数为 5，则预约 5 号程序；	单独时候脉冲信号；二进制时候高低电平	自动模式有效
5	确认程序预约	当程序预约的模式为二进制时有效，先用程序号确定预约程序号，然后输入此信号，用以确定预约。	脉冲信号	自动模式有效

6	取消程序预约	当程序状态为预约中，先选择程序号，单独时候，需要输入单独的 IO 并保持。然后输入此信号，用以取消当前已经预约的信号。	脉冲信号	自动模式有效
7	启动/停止程序预约	当程序预约处于停止状态，输入此信号，可以开始程序预约运行；当程序预约处于启动状态，输入此信号，可以停止程序预约运行。注意，停止程序预约，机器人依旧会将当前正在运行的程序运行完成。之后已经预约的不再执行。	脉冲信号	自动模式有效

2、地址值：信号需要配置的实际 io 端口号，若未配置则显示-1，其中系统占用的 IO 需要参考电气手册且不可进行配置。远程模块地址值是跟随本地实际最大地址值之后的。比如本地一共 16 输入口，则远程模块第一个端口地址为 17。

3、有效值：0 或 1，如果检测脉冲信号，则 0 表示检测到下降沿有信号，1 表示检测到上升沿有信号；如果检测高低电平，则 0 表示检测到低电平有信号，1 表示检测到高电平有信号。

4、滤波时间：为消除干扰信号，设置一个较小非负数时间，单位为秒。

输出界面：

1、描述-输出：

通用功能

序号	描述	说明	输出信号	操作模式
1	伺服状态	当前机器人的伺服状态	高低电平	自动/手动有效
2	运行状态	当前程序是否正在运行状态	高低电平	自动模式有效
4	报警状态	当前是否存在报警	高低电平	自动/手动有效
5	急停状态	机器人紧急停止开关状态	高低电平	自动/手动有效
6	程序加载完成状态	程序加载完成，发出此信号；程序开始运行，信号复位。	高低电平	自动/手动有效

安全监控

序号	描述	说明	输出信号	操作模式
1	A1 占用输出 A2 占用输出 A3 占用输出 A4 占用输出 A5 占用输出 A6 占用输出 A7 占用输出 A8 占用输出	区域 1 到 8 的占用输出信号，机器人进入共享区域内发送占用输出信号，退出共享区后取消发送该信号。	高低电平	自动/手动有效
2	安全位置	安全监控功能中定义的八个位置	高低电平	自动/手动有效

	1/2/3/4/5/6/7/8	状态信号，当机器人到达位置则输出对应信号。		

冲压

序号	输出-描述	说明	输出信号	操作模式
1	工具1动作状态	工具打开关闭动作	高低电平	自动模式有效
2	工具2动作状态	工具打开关闭动作	高低电平	自动模式有效
3	允许取料	允许下台机器人取料	高低电平	自动模式有效
4	允许放料	允许前一台机器人放料	高低电平	自动模式有效
5	冲床动作	冲床冲压动作	高低电平	自动模式有效
6	冲床急停	使机器人急停	高低电平	自动/手动有效
7	吸盘 1 破真空	吸盘吹气动作	高低电平	自动模式有效
8	吸盘 2 破真空	吸盘吹气动作	高低电平	自动模式有效
9	码垛完成	码垛完成后给予完成信号	高低电平	自动模式有效
10	拆垛完成	拆垛完成后给予完成信号	高低电平	自动模式有效
11	定位装置动作	定位夹具打开关闭动作	高低电平	自动模式有效
12	传送带夹具动作	传送带夹具打开关闭动作	高低电平	自动模式有效
13	传送带动作	传送带开始结束动作	高低电平	自动模式有效

高级码垛

序号	描述	说明	输出信号	操作模式
1	一/二/三/四号垛位满垛信号	垛盘码放满垛后输出信号	高低电平	自动/手动有效
2	一/二/三/四号夹持区打开	输出用于控制各夹持区将产品释放	高低电平	自动/手动有效
3	一/二/三/四号夹持区关闭	输出用于控制各夹持区将产品抓取	高低电平	自动/手动有效

弧焊

序号	描述	说明	检测信号	操作模式
1	起弧	起弧信号	高低电平	自动/手动有效
2	送丝	送丝信号	高低电平	自动/手动有效
3	退丝	退丝信号	高低电平	自动/手动有效
4	检气	打开焊接保护气	高低电平	自动/手动有效
5	机器人故障	机器人发生故障输出信号	高低电平	自动/手动有效
6	寻位开始	在接触寻位应用中，发生给焊机开始寻位的信号。	高低电平	自动/手动有效

弧焊（模拟量）

序号	描述	说明	操作模式
1	给定电流	当焊接协议为模拟量时，发送给焊机的电流数据	自动/手动有效
2	给定电压	当焊接协议为模拟量时，发送给焊机的电压数据	自动/手动有效

程序预约

序号	描述	说明	检测信号	操作模式
1	程序预约启动状态	启动程序预约，发出此信号；停止程序预约，信号复位。	高低电平	自动模式有效
2-16	程序状态 1-15	反馈程序预约的状态，当程序未预约，则无信号；当程序预约中，则输出常量信号；当程序运行中，则输出方波信号。	/	自动模式有效

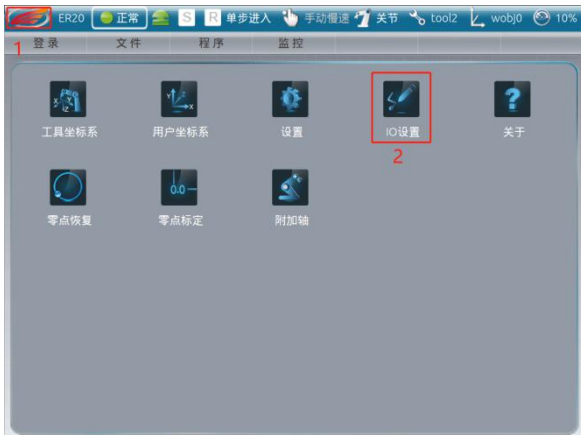
2、地址：信号需要配置的实际 io 端口号，若未配置则显示-1，其中系统占用的 IO 需要参考电气手册且不可配置。远程模块地址值是跟随本地实际最大地址值之后的。比如本地一共 16 输入口，则远程模块第一个端口地址为 17。

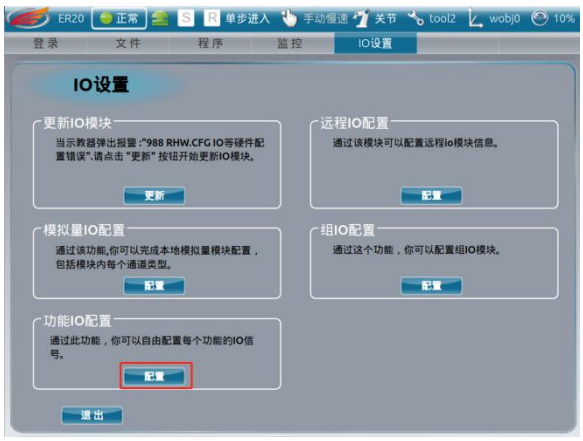
3、有效值：0 或 1，如果检测脉冲信号，则 0 表示检测到下降沿有信号，1 表示检测到上升沿有信号；如果检测高低电平，则 0 表示检测到低电平有信号，1 表示检测到高电平有信号。

12.4.1 可编程 F 按键

通过可编程 F 按键，可以绑定对应的 IO 口进行控制，一般应用于手动模式下快速控制外围气缸和气爪之类的工具。目前仅支持 F4 按键，其他按键已被系统占用。

表 1-1 设置 F 键对应的 IO 控制信号

步骤	图示	说明
1．按图示标记编号，依次打开示教器桌面，点击**“IO 设置”**图标进入主界面。		

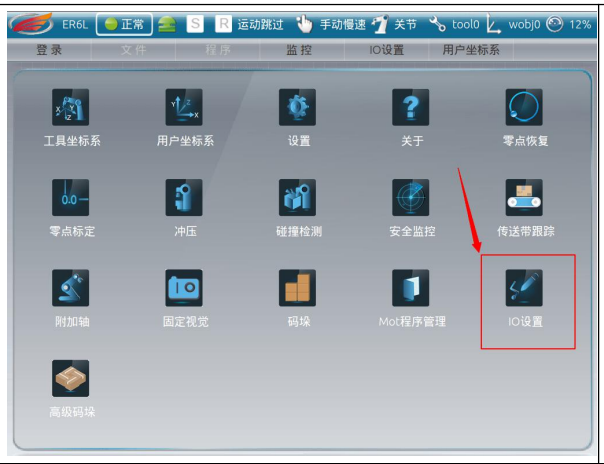
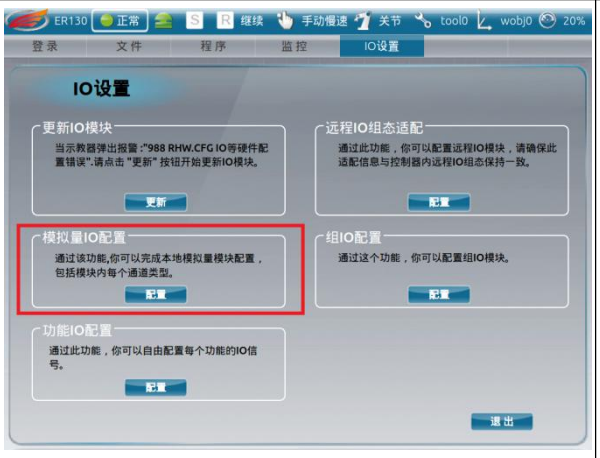
2.根据图示，选择功能IO配置，点击**“配置”**进入界面。		
3.根据图示，点击**“可编程按键”**进入界面。	 	

4.进入主界面后，点击**“编辑”**，可以对参数进行设置，如选择**“输出”**类型。设置完成后点击保存即可。		参数含义： 1、类型：可选无、输入、输出类型，如图选择输出，表示F按键对应是IO输出功能； 2、按键按下：支持切换、置为1、置为0三种模式，如图选择切换，则表示没按一次F4键，信号在TRUE和FALSE之间进行切换控制； 3、允许自动模式：不支持修改，仅手动模式有效； 4、IO地址：即表示F4按键对应控制类型绑定的的IO地址，如图表示F4可以控制输出IO地址为7的信号。
5.如果是选择输入类型，界面如图，修改IO地址后点击**“保存”**，即可完成F键对应输入地址的强制控制。		说明： 1、选择输入类型时，设置完对应IO地址保存后，按下F4键可对相应IO进行强制。可在监控IO界面查看到强制状态。不会对实际物理IO口进行强制点亮IO口。
6.完成设置后，点击**“退出”**按钮返回主界面，设置完毕。		

12.5 模拟量 IO 配置

表 11-5 模拟量 IO 配置操作步骤

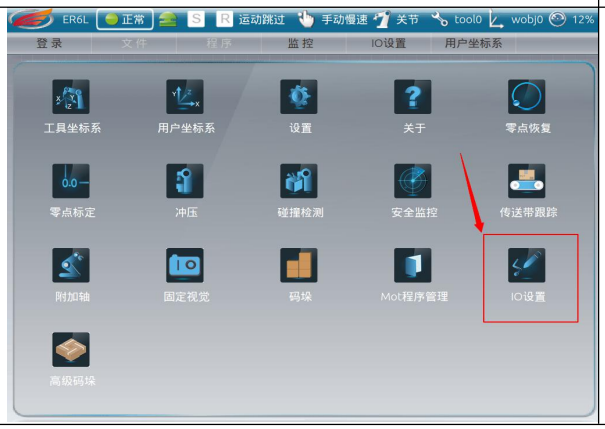
步骤	图示	说明

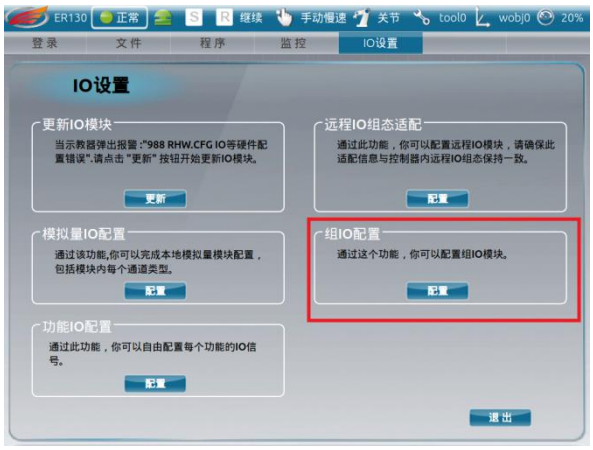
1.进入 IO 设置界面。		打开示教器桌面，点击“IO **设置**”图标。
2.进入模拟量 IO 配置界面。	 	选择**“模拟量 IO 配置”**功能。 点击**“配置”**按钮进入配置界面。 若没有配置模拟量 IO 硬件模块，则有弹窗提示且无法进入模拟量配置功能。 存在模拟量 IO 硬件模块时，则可以进入模拟量 IO 界面，显示 IO 模块类型和上次配置的通道类型。

3.进入模拟量 IO 配置界面，进行通道参数配置。	 	此前配置的模拟量 IO 信息，可以在这里查看。 点击“**编辑**”按钮，将启用编辑功能。 编辑完成后点击“**保存**”按钮，保存设置的模拟量通道信息，点击“**放弃**”按钮则不保存。 点击“**退出**”按钮，返回配置主页面。

12.6 组 IO 配置

表 11-6 组 IO 配置操作步骤

步骤	图示	说明
1. 进入 IO 设置界面。		打开示教器桌面，点击“**IO 设置**”。

2. 进入连续地址的组 IO 输入输出配置界面。		选择组 IO 配置功能。 点击**“组 IO”**按钮进入配置界面。
3. 进入组 IO 配置界面，进行每组 IO 的连续地址。		红色框 1 显示当前表格为输入或输出表格，可点击向左向右箭头完成切换。 红色框 2 显示组编号，最多支持配置 16 组。 红色框 3 确定连续地址的起始地址。 红色框 4 确定连续地址的终止地址。 红色框 5 显示一组 IO 信号的状态转换成的十进制值。
		点击**“编辑”**按钮后，按钮变为保存，表格变为可编辑状态，用户可对每组的起始结束地址进行编辑。 编辑完成后点击**“保存”**按钮，使得表格处于不可编辑状态，并将生成的配置文件发送至控制器中保存。 输入界面的信号值列显示十进制数，对应连续地址 IO 信号状态组成二进制值转换成的十进制值。 输出界面的信号值列可设置输出值，在弹出键

		盘上设置一个十进制值，转换为二进制值控制相应的组 IO 输出。

第 13 章　附加轴配置

13.1 本章简介

本章主要介绍附加轴轴数配置、参数设置、位置监控、附加轴指令的使用。

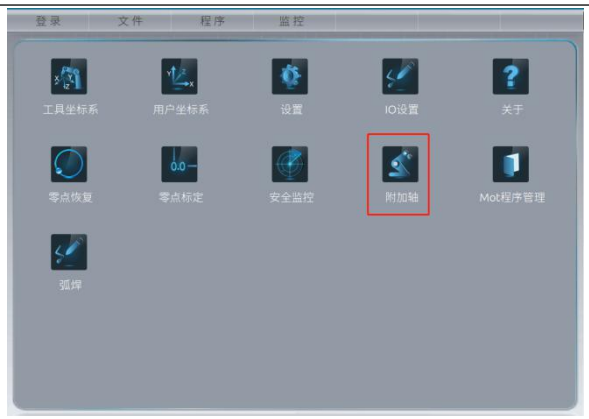
13.2 附加轴功能简介

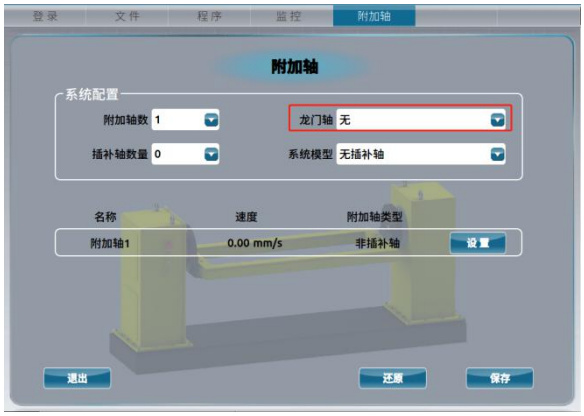
附加轴功能主要是在标准的机器人基础上，增加 1 到 4 个附加轴，来配合机器人完成客户现场复杂工作。机器人如果存在龙门双驱同步附加轴（以下简称龙门轴），则最多支持 3 个附加轴。附加轴可分为插补轴和非插补轴，当所有附加轴都不为插补轴时候，非插补轴又可以分为同步轴和异步轴。

13.3 附加轴配置

13.3.1 附加轴轴数配置

表 12-1 附加轴轴数配置操作步骤

步骤	图示	说明
1. 打开附加轴功能。		打开示教器桌面，点击**“附加轴”**功能图标进入主界面。
2. 配置附加轴轴数。		选择附加轴数后，下方会对应出现需要配置的附加轴个数。 最多支持配置 4 个附加轴。如果需要配置龙门轴，则最多配置 3 个附加轴，因为龙门轴占用两个物理驱动器位置。

3. 配置是否存在龙门轴。	 	龙门轴在软件中是指两个电机同步运动，控制龙门架移动运行。使用龙门轴需要两个电机型号参数应保持相同。在本系统中龙门轴的设置及运动使用等同一个附加轴，且龙门轴必须为所有附加轴中的最后一个轴。 如果存在龙门轴，注意两个电机的运动方向： 1. 主从轴电机相向运动即为两个电机转动方向相反，一般电机如果安装在导轨相对两侧，或者相对放置选择此选项。 2. 主从轴电机同向运动即为电机转动方向相同，一般电机安装在导轨同一边，或者同向放置选择此选项。 存在龙门轴需要配置主从轴最大误差，范围：0.01-100mm。
4. 配置插补轴数量。		插补轴数量不同，对应的系统模型存在差别，插补轴数量修改，相应出现当前轴数可配置的系统模型。具体看第 5 步。 插补轴数量不超过附加轴总轴数，且不超过 3。

1. 配置插补轴的系统模型。		选择**系统模型**，确定各附加轴类型。 插补轴数量为 1 时，系统模型分为：X、Y、Z 直线插补和 X、Y、Z 旋转插补。 插补轴数量为 2 时，系统模型分为：X-Y、Y-X、X-Z、Z-X、Y-Z、Z-Y 直线插补。 插补轴数量为 3 时，系统模型分为：X-Y-Z、X-Z-Y、Y-X-Z、Y-Z-X、Z-X-Y、Z-Y-X 直线插补。 附加轴作为插补轴，各个轴的运动方向必须和机器人在机器人坐标系下的运动方向相同。
6. 设置单个附加轴参数。		点击**“设置”**按钮进入附加轴设置参数界面。 如果要退出 APP 回到桌面，则点击下方**“退出”**按钮。 若确定配置完成后使其生效则点击下方**“保存”**按钮修改配置文件保存参数，点击按钮后等待控制器和示教器重启完成后生效。若不想要保存此次修改设置则点击**“还原”**按钮恢复到上次保存的数据。

7. 附加轴信息监控。		图中左侧红框是用来表示当前PLC信号选定的附加轴。绿色图标即为当前轴（只有不存在插补轴，且异步轴功能打开，选择PLC信号控制为**“是”**的时候才有效）。 图中右侧红框是附加轴运动的实时速度。

13.3.2 附加轴参数设置

表 12-2 附加轴参数设置操作步骤

步骤	图示	说明
1. 进入附加轴配置界面。		点击**“设置”**按钮进入附加轴设置限制参数界面。 如果要退出APP回到桌面，则点击下方**“退出”**按钮。
2. 设置附加轴参数。		在此界面进行附加轴参数设置。（以附加轴1为例）。 除了基础参数外，作为非插补轴使用且异步轴功能开关打开后，在底部需要配置异步轴参数。 设置完成后，点击**“返回”**按钮。 设置过程中不想保存当前附加轴设置的参数，可点击**“放弃修改”**按钮将当前轴所有参数还原为上次保存的参数。 异步轴PLC信号控制分为否和是： 否：附加轴作为异步

		轴的时候，各个附加轴由自己的附加轴信号进行控制。 是：附加轴作为异步轴的时候，各个附加轴由附加轴 1 的两个信号进行控制，由 PLC 信号决定当前控制哪一个附加轴。

附加轴参数说明：

异步轴功能：选择异步轴开关，开表示附加轴作为同步轴或异步轴使用，两者可以在程序中切换使用。关表示附加轴只能作为同步轴使用。若附加轴类型为插补轴，则异步轴使能开关强制为关闭状态，无法打开，无需配置异步轴参数，无需配置轴类型参数，在系统设置主界面已确定。无需配置路径规划，插补轴路径规划参数默认为正常路径，且无法修改。

方向：改变减速机输出转动方向，即改变旋转轴或者直线轴的运动方向。1 表示正方向，-1 表示负方向。

正限位/负限位：可根据实际情况填写附加轴的机械限位值，要求正限位必须大于负限位。

减速比：减速机输出端转一圈，电机需要转多少圈。如减速机转 1 圈，电机转 50 圈，减速比为 50。此数值可以为小数，但是不能为负数。

位置转换：表示减速机输出端转一圈，机械结构运行的行程，比如减速机转 1 圈，机械结构转 360°，则位置转换的值为 360。

编码器分辨率：电机的编码器分辨率，填写分辨率数值。比如 17 位的编码器分辨率就是 131072。

手动速度：手动情况下，用示教器按键移动附加轴的最大速度。直线轴范围为 0.001-250mm/s，旋转轴范围 0.001-50°/s，需要注意，如果附加轴参数中的最大速度比手动速度小，则手动模式下，示教器按键移动附加轴的速度为最大速度参数中设置的值。

最大速度：轴机械机构最大速度值，可根据选型的电机最大速度来确认该值，一般情况下，最大速度≤（电机最大转速*位置转换）/（60*减速比）。

最大加/减速度：轴运动的加速度和减速度值，一般可按照最大速度参数的 2-4 倍数值设置，一般大型龙门或者行走轴，建议 2 倍即可。

最大加加速度：轴运动的加加速度值，一般可按照最大速度参数的 20-100 倍数值设置，一般大型龙门或者行走轴，建议 20 倍即可。

轴类型：确定附加轴系统模型后轴属于插补轴或非插补轴已确定，非插补轴可自由设置轴作为直线轴还是旋转轴，插补轴不能自由选择，龙门轴必须为直线轴。

路径规划：分为正常路径和最快路径。非插补的旋转轴可以配置附加轴运动路径规划方式为正常路径或最快路径，当选为最快路径时，正负限位自动设为：0-360°。而直线轴只能配置为正常路径，手动配置为最快路径会有弹窗提醒并强制为正常路径。

1) 正常路径，附加轴在设置的正负限位中间运动，设置多少度，运行到多少度。

2) 最快路径，设置最快路径，需要将附加轴负限位和正限位设置为 0° 和 360° 。当为异步轴的时候，机器人运行到 360° 后，轴位置又从 0° 开始。当为同步轴时候，机器人会以最快的路径运行到目标位置，比如机器人当前位置在 90° ，运行到 300° ，机器人会以 90° ~0° ~300° 方式运行，如果运行到 250° ，会以 90° ~180° ~250° 方式运行。

异步轴速度：附加轴作为异步轴时的运行速度，值是最大速度的百分比，可输入 1-100 整数。

异步轴运动方式：异步轴运动方式分为单步和连续。

单步模式：接收到对应的单步信号，机器人运行设置的角度，正在运动的轴不再接收此类信号。

连续模式：单个轴分为正负两个方向运动，接收到信号，附加轴往信号对应的方向运动，取消信号，附加轴立即停止运动。

注：详细信号配置见 IO 设置章节的功能 IO。

步进角度 1/步进角度 2：表示附加轴作为异步轴时，接收到一次信号后，运行的行程。运动分正负方向运动，数值为负数表示反方向运动。

PLC 信号控制：是选项表示通过 PLC 信号接收到一个附加轴索引，则该索引对应的附加轴使用附加轴 1 的 IO 信号进行控制；否选项表示不通过 PLC 信号复用 IO 端口，各轴独立使用各自 IO 端口。

注：附加轴步进信号设置可以在：IO 设置的 IO 自由配置中完成。详细设置方法见 IO 设置章节的功能 IO 配置。

13.4 附加轴位置监控及清零

表 12-3 附加轴位置监控及清零操作步骤

步骤	图示	说明
1.单关节运动和附加轴位置监控。		在附加轴参数配置完成后。点击任务栏**“监控”**中**“位置”**，进入位置监控界面，按示教器下方的**“2nd”**按键或者点击图标选择**“附加轴”**快捷菜单键，当状态栏坐标系信息显示为**“附加轴”**，则可通过示教器右侧的轴运动按键控制附加轴运动，1 为控制附加轴 1 运动按键，依次类推。 如果只是查看附加轴当前位置，在状态栏坐标系信息显示的不是附加轴时可以通过界面中下拉框

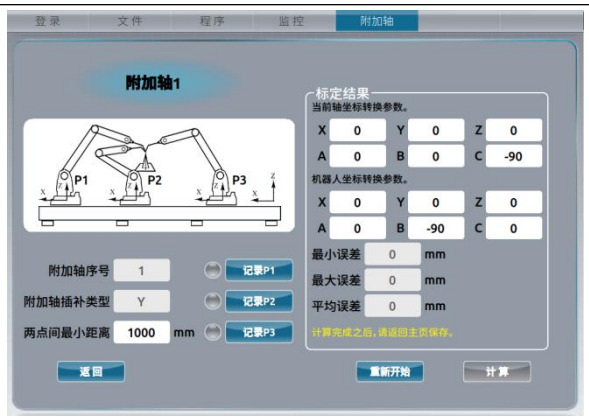
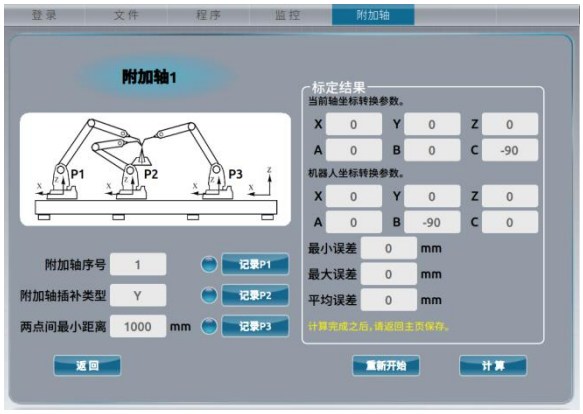
		选择附加轴，但这时轴运动按键对附加轴运动无效。 **注意：自动模式下，按示教器下方的“2nd”键无效。**
2.附加轴清零。		点击任务栏**“监控”**中**“驱动器”**，按示教器下方的**“2nd”**按键或者点击图标选择**“附加轴”**快捷菜单键，切换附加轴和本体轴清零界面。当状态栏坐标系信息显示为**“附加轴”**，进行附加轴清零，操作和正常轴相同。 龙门轴显示包括主从轴，清零和编码器重置只需要操作主轴即可，操作和正常轴相同。

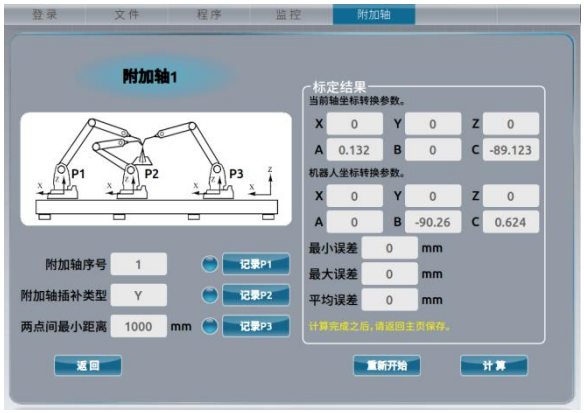
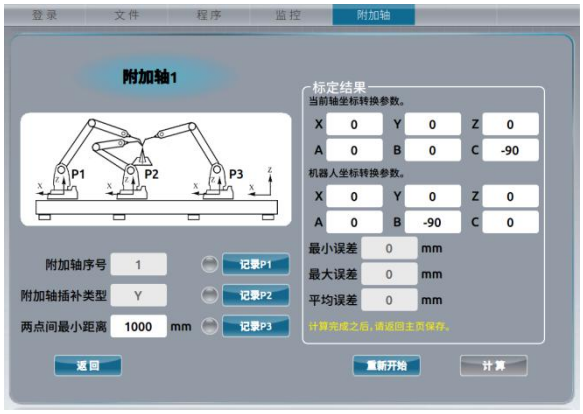
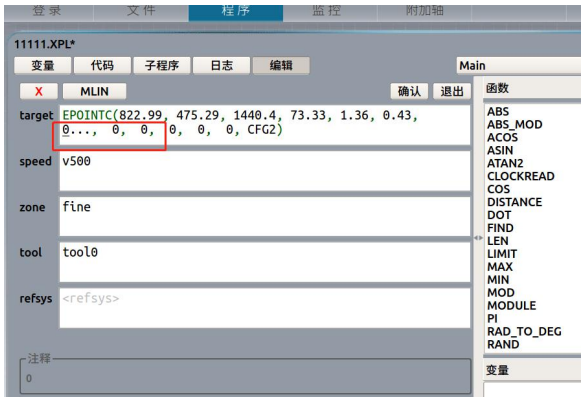
13.5 附加轴标定

当一个附加轴类型为直线插补轴时候，可以对这个附加轴进行标定，主要目的是为了补偿安装的误差，提高附加轴参与插补的精度。附加轴标定的步骤如下所示。

表 1-4 附加轴标定步骤

步骤	图示	说明
1.配置带插补轴的附加轴。		按附加轴配置章节中说明正确配置附加轴参数后，可以看到带插补的附加轴后面“标定”按钮，点击后可以进入标定界面。 附加轴标定前需要先正确保存附加轴参数。如果参数未保存，要求先保存才能进行附加轴标定。 **附加轴标定必须按照附加轴的轴序号进行标定，即按照附加轴 1、2、3、4 的顺序进行标定。标定一个附加轴时候，其他**

		**附加轴不能移动。**非插补轴无需标定。 附加轴标定完成之后，如果修改系统模型之后，附加轴标定参数会丢失，需要重新标定。 如果龙门轴的左右边出现误差，导致龙门左右和之前标定的时候存在偏差。那么需要将龙门插补轴重新标定。
2.设置两点最小距离。		进入到附加轴标定界面，需要先填写两点间最小距离，表示示教三个点的附加轴位置距离，一般越大约准确。 如果实际示教位置时候，两个位置的附加轴位置小于该距离，则计算时候会出现相应报错。
3. 示教标定点。		将附加轴上机器人移动到一端，然后将机器人工具末端对准一个固定点，点击“**记录 P1**”，示教第一点。 移动机器人，使机器人 TCP 在位置 2 和位置 3 对准固定点。分别记录 P2 和 P3 点。 注意：使用附加轴标定，工具坐标系要求准确。否则影响附加轴标定精度。

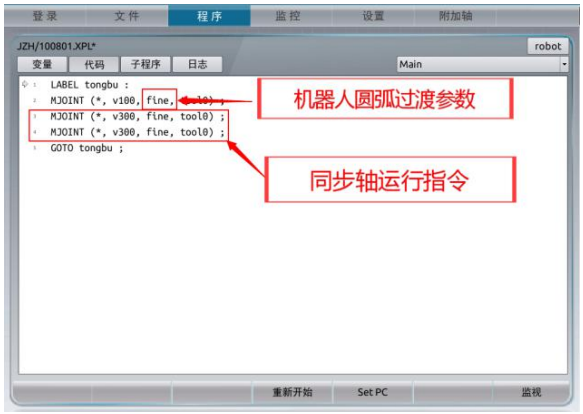
4. 计算结果。		三个点记录完成后，点击右下角“**计算**”按钮，则计算结果会显示在界面上。 计算完成之后，需要返回主页进行数据保存。
5. 标定结果直接填写。		在标定结果中，附加轴转换参数和机器人坐标转换参数是可以直接修改的。修改完可以直接回主页保存即可。
6. 附加轴标定精度验证。		附加轴标定完成后，新建个程序，在程序建立几个相同位置点，机器人 TCP 对准一个固定尖点。 手动修改附加轴的坐标（修改后的位置点，机器人和附加轴能正常到达），注意机器人 xyz 的位置坐标不要修改。 运行程序，程序运行过程中，TCP 点偏离固定点越小，精度越高。

13.6 附加轴指令

13.6.1 同步非插补轴指令

表 12-4 同步非插补轴指令操作步骤

步骤	图示	说明

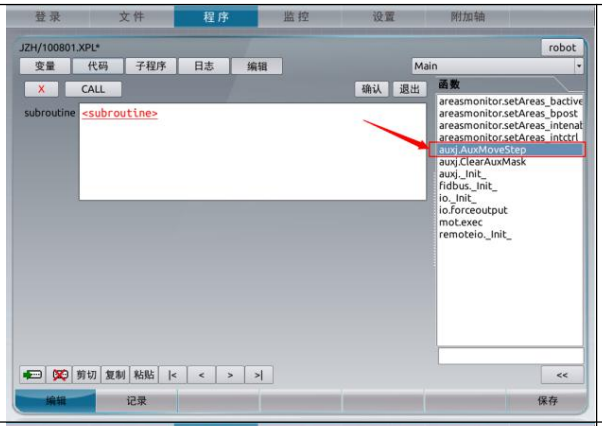
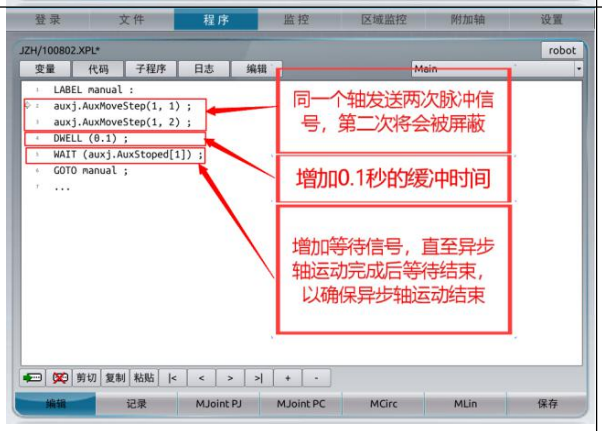
1.通过指令控制同步非插补轴的运动。		在附加轴参数配置完成后，附加轴功能已经开启，通过特定的 RPL 语句完成对附加轴运动的控制。 **注意：j1-j6 代表机器人轴，ea1-ea6 代表附加轴。**
2.同步非插补轴的指令使用方法。	 	附加轴的类型属于同步非插补轴，不参与机器人运动轨迹的插补，在编程时，建议将**圆弧过渡参数（zone）**设置为 **fine**（即运动到目标位置后再执行下一指令）。 同步轴关节运行指令为 **MJOINT（EPOINTJ)**，直线运行指令为 **MLIN(EPOINTC)**。 **注意：同步轴关节运行指令不可使用 MJOINT（EPOINTC)，否则将发生报错。**

13.6.2 异步轴指令

异步轴的运行主要分为两个模式，自动模式下编程运动和手动模式下 IO 信号控制运动。

表 12-5 异步轴指令操作步骤

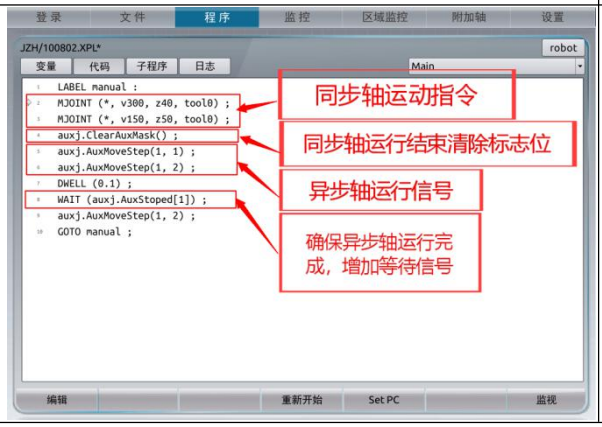
步骤	图示	说明

1.自动模式添加附加轴异步运动指令。		在相关程序文件中新建一行程序，选择**“CALL”**指令。 在**”CALL”**指令里面选择相关附加轴函数。
2. 编程控制异步轴运动。		**auxj.AuxMoveStep（AuxNum，Step）**指令中，**AuxNum** 参数是附加轴序号，参数值为 1-4，表示选择运动的附加轴，**Step** 参数是附加轴 APP 中设置的步进角度值，参数为 1 或者 2，表示一次完成步进角度为步进角度 1 或者步进角度 2。 **例：auxj.AuxMoveStep(1,2)**表示**附加轴** 1，一次运行完成**步进角度** 2 表示的角度。 对同一个轴的连续两次脉冲信号，第二次将被屏蔽，为保证异步轴运动已经完成，增加等待信号。 为确保异步轴运动完成，在发送异步轴运动脉冲信号后紧接着等待信号，如果异步轴未运动完成则 **WAIT 语句**一直运行等待。增加 **DWELL 语句**，0.1 秒的缓冲时间确保异步轴开始运动后再执行 **WAIT 语句**，否则异步轴还未开始运动，则 **WAIT 语句**执行结束。
3.手动模式下，IO 信号控制异步轴运		在手压信号存在的情况下，通过给特定的输入

动。		口脉冲信号，来控制选定的异步轴按照设定的步进角度进行运行。 在异步轴运行期间，屏蔽所有的控制异步轴运动的脉冲信号。 **注意：IO 配置方法见 IO 设置的功能 IO 章节。**

13.6.3 同步异步在线切换

表 12-6 同步异步指令在线切换操作步骤

步骤	图片	描述
1. 同步非插补轴和异步轴在线切换，且同步非插补轴的优先级高于异步轴。		

同步和异步在线切换说明：

同步轴运行期间，一切异步轴请求将不被响应；在异步轴运行期间，如果有同步非插补轴请求，将中断异步轴运行指令。

在同步插补轴切换成异步轴时，如果圆滑过渡指令为 fine（即没有圆滑过渡）或者后面紧跟着的指令为 MJOINT(POINTJ/POINTC)，则不需要执行 auxj.ClearAuxMask()指令，如果圆弧过渡指令为 zone 类型（z50 等等）且其后跟随异步轴运动信号：auxj.AuxMoveStep()指令，则需要在同步轴运动结束后执行标志位清除指令：auxj.ClearAuxMask()。

在异步轴换成同步插补轴切时,为保证异步轴运行已经结束，需执行指令 WAIT（auxj.AuxStoped[auxNum])指令让程序一直在此行等待，直至指定异步轴运动完成继续执行下一行指令，否则异步轴的运行有可能被打断，其中 auxNum 参数表示附加轴序号，可设置为 1-4，表示附加轴 1 到附加轴 4。

例如：当附加轴 1 在运动过程中，auxj.AuxStoped[1]的值为 false，WAIT（auxj.AuxStoped[1]) 使得程序一直在这一行等待，当附加轴 1 运动结束后，auxj.AuxStoped[1]的值为 true，WAIT（auxj.AuxStoped[1])指令执行完成，继续执行下一行指令。

第 14 章　简单码垛

14.1 本章简介

本章主要介绍简单码垛系统组成及功能、界面说明及操作说明。

14.2 系统组成及功能说明

本码垛功能为简单码垛，码垛工件形状一致，摆放方向相同，行列分布均匀。简单码垛操作流程分为三大部分：码盘坐标系的建立、码垛基本信息设置和码垛执行程序编写。

垛盘坐标系的建立。该步骤为码垛操作流程，建立以垛盘为基准的用户坐标系。

码垛基本信息的设置。该步骤主要设置堆垛物品矩阵的行、列、高信息以及码垛中机器人的路径信息。系统会将保存的码垛信息直接生成用户可调用的机器人动作指令，方便用户调用。

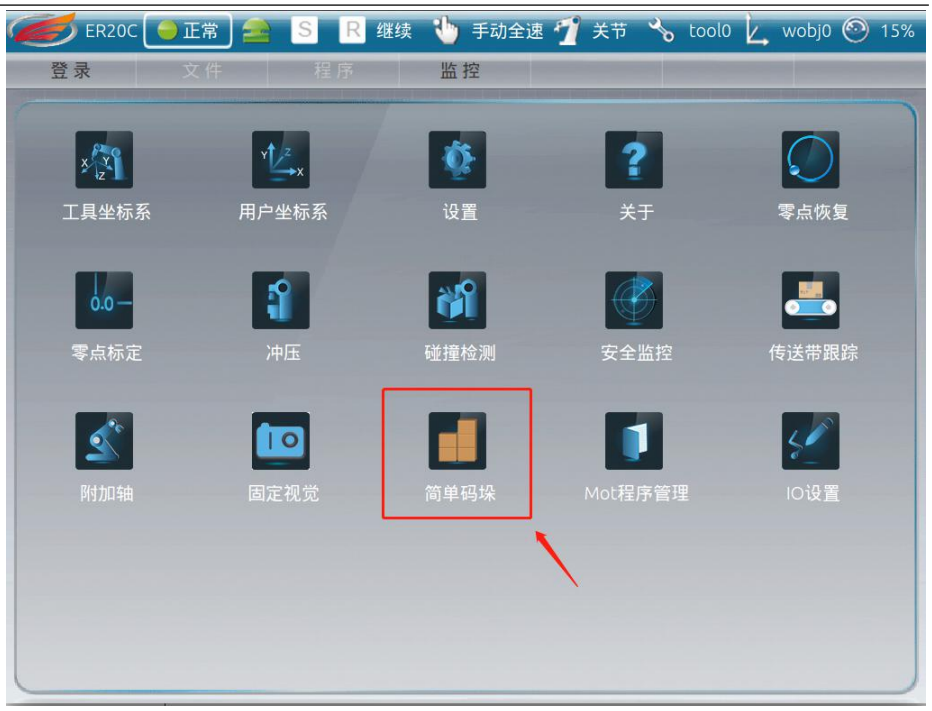
码垛执行程序的编写。该部分需要用户自身需求，将码垛动作指令嵌入工艺程序中，有出厂默认配置程序模板。

14.3 用户坐标系的声明和标定

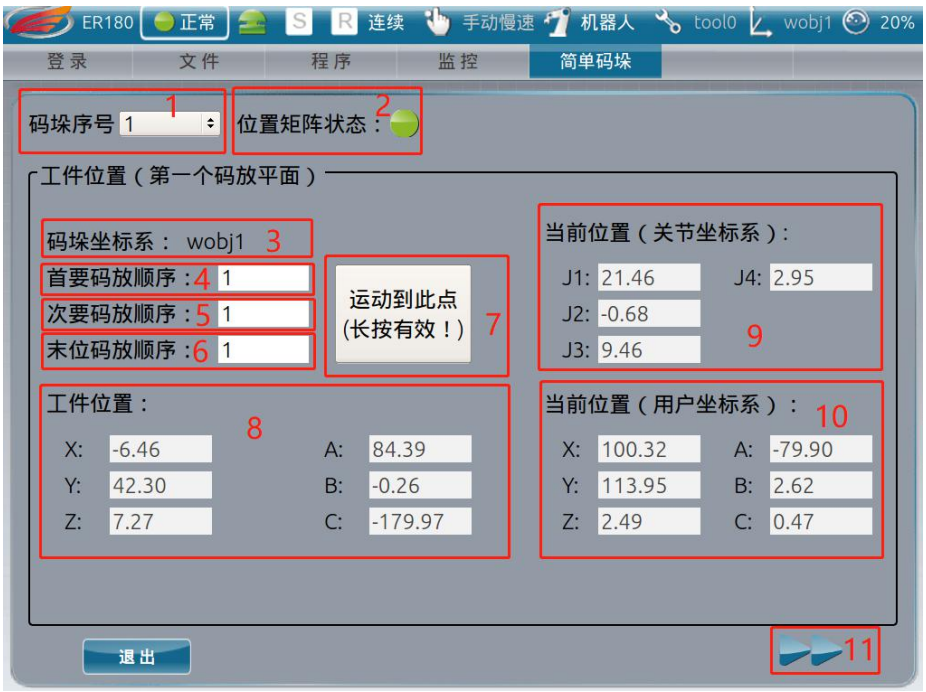
如若使用新的用户坐标系，需在用户程序中声明“**REFSYS**”用户参考坐标系变量，并使用桌面“**用户坐标系**”app 对新建的用户坐标系进行标定；若使用未标定的已声明的坐标系，请先行标定该坐标系。

14.4 码垛基本信息设置

表 14-1 码垛基本信息设置操作步骤

步骤	图示
1. 打开示教器桌面，点击“简单码垛”功能图标进入“步骤 2”中主界面。	

2.配置码垛基本信息。



1) 码垛序号：堆/垛的序号，码垛程序最大支持 8 个堆/垛。

2) 位置矩阵状态：当位置矩阵所有信息完备，且当前无修改的情况下，会显示绿色；当位置矩阵信息不全，或对当前位置矩阵信息做了修改而未保存时，会显示红色。

3) 码垛坐标系：显示本堆/垛的位置信息所基于的坐标系名称。

4) 首要码放顺序：此处“码放顺序”为基础设置中选择的码放作业方向顺序关系。比如选择的 X-Y-Z 表示先按 X 方向逐个码放，再向 Y 方向移动一个码放间距后按 X 方向继续码放，直至 X-Y 平面码放完成，再向 Z 方向移动一个码放间距继续码放直至完成。则“首要码放顺序”值表示 X 方向的第几个产品的索引值。

5) 次要码放顺序：设置“码放顺序”选项的第二顺序方向的产品索引。

6) 末位码放顺序：设置“码放顺序”选项的第三顺序方向的产品索引。

7) 运动到此点：上伺服状态下，长按此按钮，机器人将向当前码垛点的位置做直线运动，当松开按钮时，机器人会立刻停止运动！如果要将机器人运动到当前码垛点位置，请长按该按钮直到机器人运动结束。

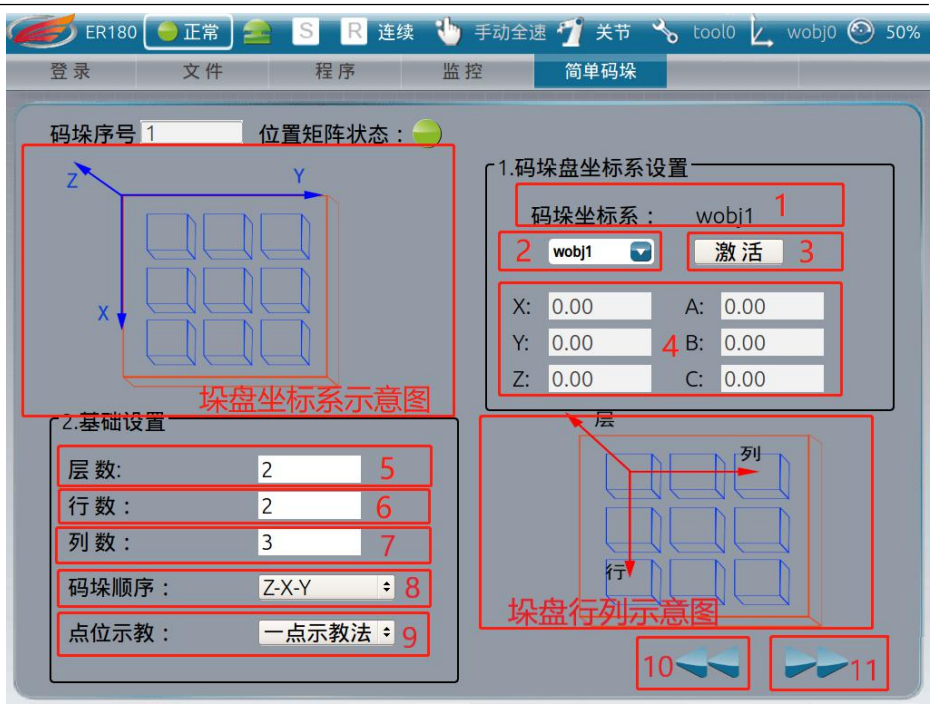
8) 工件位置：显示当前工件的点的坐标，及放置时的姿态。

9) 当前位置（关节坐标系）：显示机器人在关节坐标系下的位置信息。

10) 当前位置（用户坐标系）：显示在机器人在当前用户坐标系下的位置信息。

11) 下一页：点击“向右箭头”按钮，对码垛信息进行修改，将进入码垛信息修改第一页。

3.点击编辑按钮，将进入码垛信息修改第一页。



1) 码垛盘坐标系：显示码垛盘使用的坐标系。说明：该坐标系的 z 轴正方向必须为垛层增加方向相同（通常为竖直向上），且为右手坐标系。

2) 下拉框：显示系统中所有的用户坐标系，能够生效的用户坐标系仅为系统默认的“wobj0-wobj5”6 个系统坐标系和声明为 module 变量的用户坐标系。**着重说明**：垛盘的用户坐标系必须是右手坐标系，且 **z 轴正方向必须为垛层增加方向相同（通常为竖直向上）！**

3) 激活按钮：会将机器人的当前坐标系设置为当前选择的坐标系。

4) 为码垛盘坐标系的位置与姿态信息。

5) 层数：堆/垛的层数。

6) 行数：堆/垛工件矩阵的行数。

7) 列数：堆/垛工件矩阵的列数。

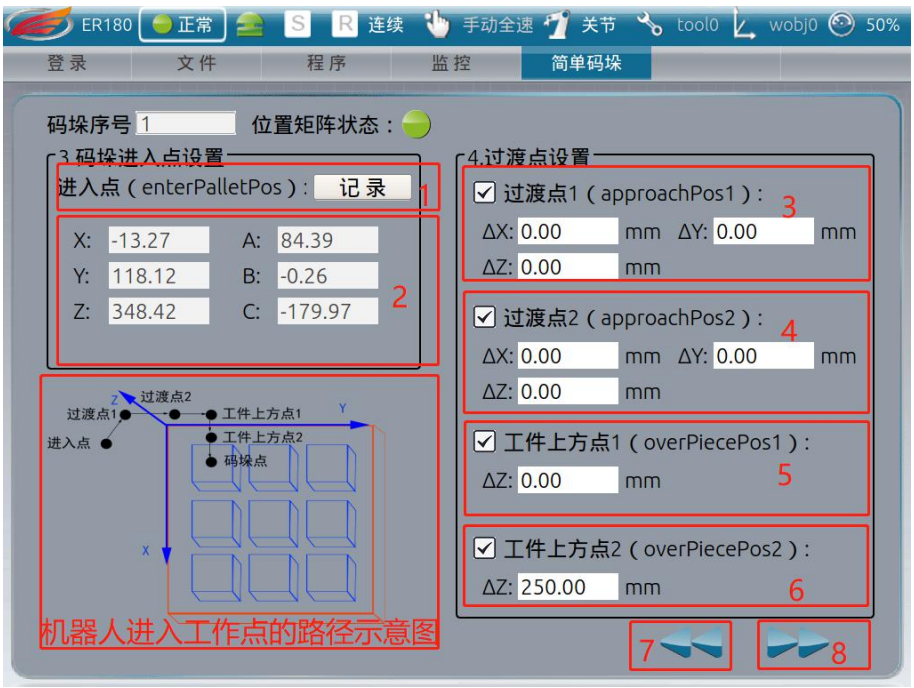
8) 码垛顺序：设置码放过程中码放方向先后顺序。

9) 点位示教：设置点位示教的方法，包括一点示教法和四点示教法。

10) 上一页按钮:返回上一页面，即码垛首页。

11) 下一页按钮：进入码垛信息设置的下一个页面。

4.点击下一步按钮，将进入码垛信息修改第二页。



1) 记录按钮：将当前机器人的位置信息，记录为码垛进入点的坐标信息。

2) X、Y、Z、A、B、C：显示进入点的坐标和工具姿态信息。

3) 过渡点 1(approachPos1)：进入点到工件上方点间的第一停顿点。

4) 过渡点 2(approachPos2)：进入点到工件上方点间的第二停顿点。

5) 工件上方点 1(overPiecePos1)：进入点到工件点间的第一个位于工件正上方的点。

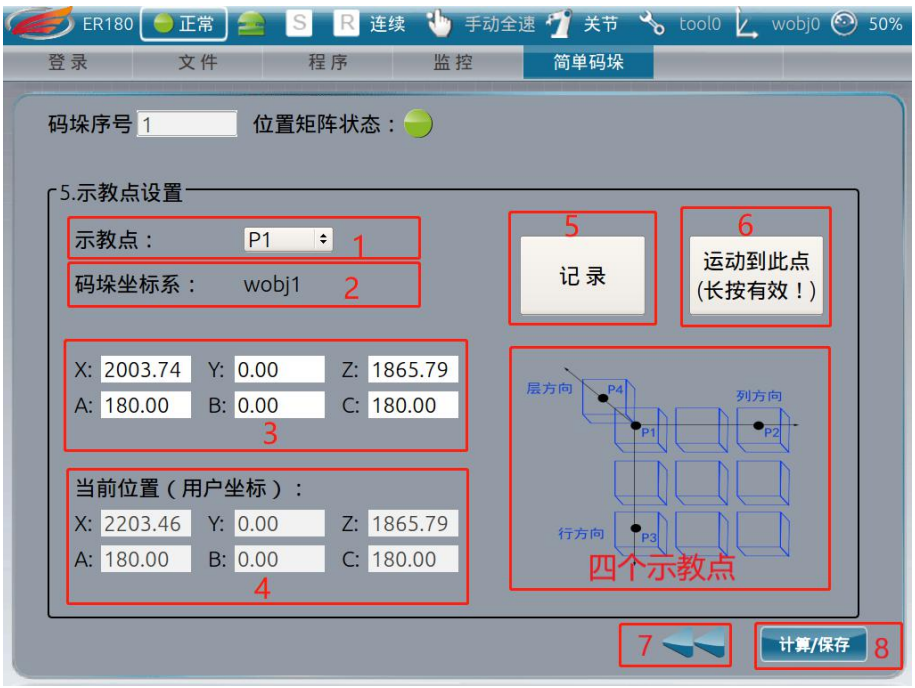
6) 工件上方点 2(overPiecePos2)：进入点到工件点间的第二个位于工件正上方的点。

7) 上一页按钮：返回到码垛信息修改第一个界面。

8) 下一页按钮：进入码垛信息修改第三个界面。

9) 注意：码垛中放置每个工件时，最多允许添加两个过渡点和两个工件上方点。过渡点为从进入点到工件上方的过渡点，可以有 X、Y、Z 三个坐标轴上的偏移量（该偏移量是以码垛盘用户坐标系为准）；工件上方点为码放点正上方的点，与码放点的位置只允许有 Z 轴方向的偏移。

5.点击下一步按钮，将进入码垛信息修改第三页。此页面存在四点示教法和一点示教法界面的区别，先对四点法示教进行说明。



1) 四点示教法：码垛程序需要示教四个点来计算码垛盘上工件矩阵的每个工件的位置及垛层的高度，四个需要示教的点分别为：

第一个点 P1：码垛起始位置，工件序号为 1 的位置。

第二个点 P2：第一行最后一个，列方向结束点位置，姿态与 P1 相同。如果只有一列，P2 与 P1 重合，此时不用记录。

第三个点 P3：最后一行第一个，行方向结束点位置，姿态与 P1 相同，如果只有一行，P3 与 P1 重合，此时不用记录。

根据前三个点，可以算出码垛盘上所有工件基于码垛盘坐标系的码垛点位置矩阵。

第四个点 P4：P1 沿着摆放高度方向一个点，姿态与 P1 相同，用来标定摆放上升高度方向。

2) 码垛坐标系：显示码垛过程所采用的用户坐标系。

3) X、Y、Z、A、B、C：为示教点的具体位置和姿态信息，可以直接输入。

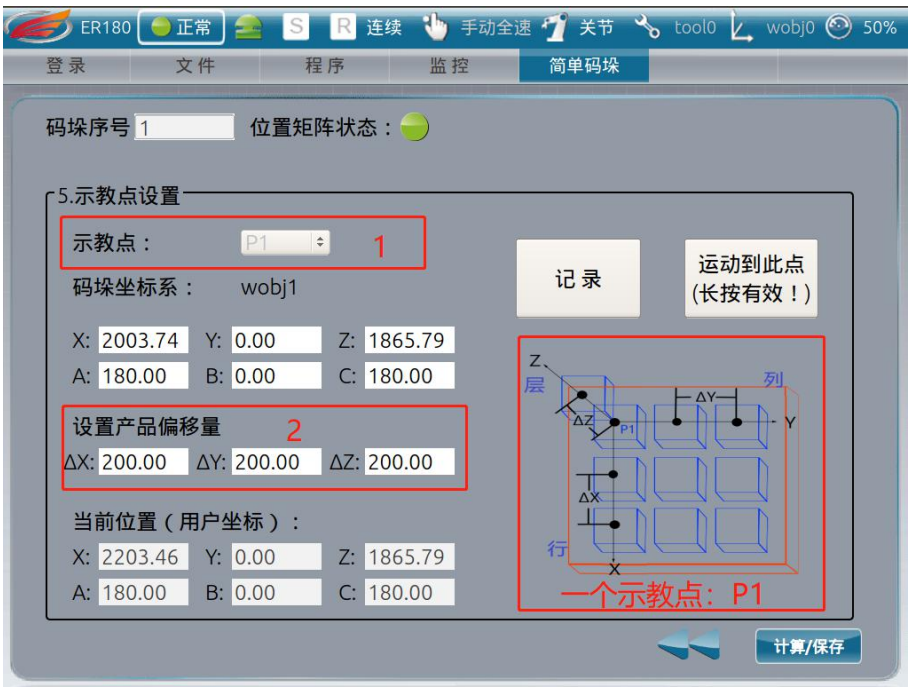
4) 当前位置：显示机器人的当前位置。

5) 记录按钮：记录当前机器人的位置作为示教点的坐标。

6) 运动到此点按钮：点击运动到上面记录的点，**该功能按钮只在“计算并保存”当前垛盘信息后才可使用**。

7) 上一页按钮：返回到码垛信息修改第二页。

8) 计算/保存 按钮：将计算码垛中所有点的坐标信息，并保存进系统文件中。

6.一点法示教说明。	

1) 一点示教法：该方法只需要示教一个点，并设置间距值，其他内容与四点示教法相同。

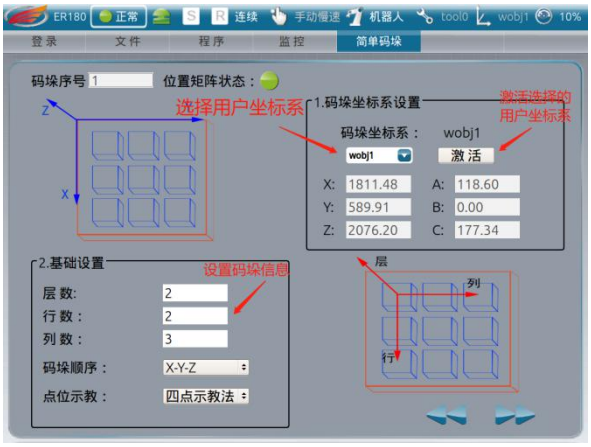
P1 点：码垛起始位置，工件序号为 1 的位置。

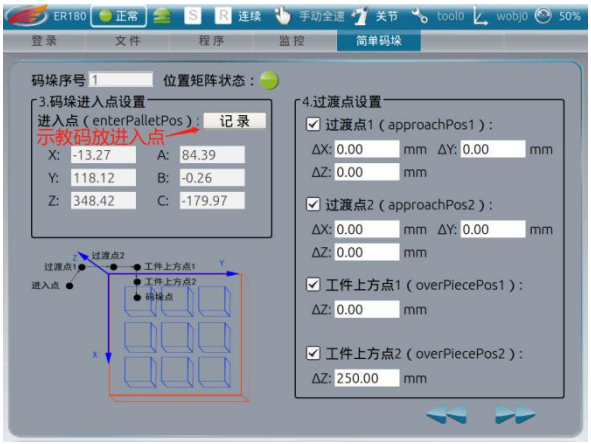
2) 设置产品偏移量：设置 X、Y、Z 方向相邻产品之间的间距。

14.5 操作说明

14.5.1 垛盘信息设置

表 14-2 码垛盘信息设置操作步骤

步骤	图示	说明
1.打开码垛功能。在简单码垛主界面点击向右翻页按钮进入第一个编辑界面。		码垛首页点击“**编辑**”按钮，进入编辑第一页； 编辑第一页“1.码垛盘坐标系设置”中，点击下拉框，选择码垛盘坐标系，并点击“**激活**”。 在“2.基础设置”中，输入码垛盘中工件的排列矩阵的相关信息，完毕后点击“**下一步**”。 注：垛盘的用户坐标系必须是右手坐标系，且 z 轴正方向必须为垛层增加方向相同（通常为竖向

		上）！
2.点击下一步按钮，将进入垛盘信息设置界面。		先将机器人移动至码垛盘中第一层第一个工件的码垛进入点位置，点击“码垛进入点设置”中的“**记录**”按钮； 根据路径需要设置“**过渡点设置**”中的生效的点，这些点的位置是相对于工件点的偏移来确定的；设置完毕后点击“**下一步**”。
3.点击下一步按钮，将进入码垛信息设置第三页		示教点 P1 为码垛第一层的第一工件放置点，位于工件矩阵第一层的第一行第一列，将机器人移动至该工件点后，点击“**记录**”进行示教； 选取下一示教点 P2，P2 为第一行最后一个工件放置点，将机器人移动至该工件点后，点击“**记录**”进行示教； 选取下一示教点 P3，P3 为第一列最后一个工件放置点，将机器人移动至该工件点后，点击“**记录**”进行示教； 选取下一示教点 P4，P4 为第二层中位于 P1 正上方的工件放置点，将机器人移动至该工件点后，

		点击“**记录**”进行示教； 点击“**计算/保存**”按钮完成码垛盘信息设置。

14.5.2 RPL 程序调用码垛功能

14.5.2.1 常用码垛指令介绍

常用函数：

1) pallet.loadCfg()函数，用来更新码垛参数设置，当用户使用简单码垛工艺包在线修改了堆垛信息后，该函数将修改后的信息更新到运行程序中，建议在码垛程序开始时调用。

说明：该函数为控制器版本 1.9.0 新增；控制器版本低于 1.9.0 时，如使用简单码垛 App 修改并计算了简单码垛信息，需重启机器人，才能让新修改数据生效。

2) pallet.update(DINT PalletID, DINT Type, DINT PieceID，DINT SequenceID)函数，用于更新当前工件点的位置信息，参数说明如下：

DINT PalletID：垛盘的序号。

DINT Type：值为 0 代表码垛，值为 1 代表取垛。

DINT PieceID：工件点的序号；码垛模式下，从 1 开始，逐次增加；拆垛模式下，从最大工件序号开始，逐次减少。

DINT SequenceID：码放的方向；按 Z 方向码放值为 1（对应码放顺序 X-Y-Z 和 Y-X-Z），按 Y 方向码放为 2（对应码放顺序 X-Z-Y 和 Z-X-Y），按 X 方向码放为 3（对应码放顺序 Z-Y-X 和 Y-Z-X）.

特别备注： 在程序使用任何码垛工艺包相关变量前，请先行调用 pallet.update()函数，否则，码垛工艺包相关变量不会被赋值。

常用变量：

1) pallet.enterPalletPos：当前工件点的进入点

2) pallet.approachPos1、pallet.approachPos2：当前工件点的中间过渡点

3) pallet.overPiecePos1、pallet.overPiecePos2：当前工件点的上方点

4) pallet.piecePos：当前工件点

5) pallet.maxPieces：当前垛盘中的工件点的数量

6) pallet.maxPallets：有效垛盘的数量

14.5.2.2 程序示例

1) 变量声明

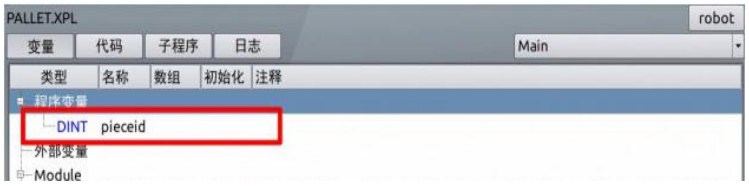


图 14-1 DINT 变量声明

2) 主函数体

```
(* 更新垛盘配置信息 *)
pallet.loadCfg() ;
pieceid := 1 ;
(* 循环码垛 *)
WHILE pallet.maxPieces >= pieceid DO
  (* 运动到抓取工件点 *)
  MJOINT (*, v500, fine, tool0) ;
  (* 通知机器人已准备好抓取 *)
  io.DOut[10] := true ;
  (* 等待抓取指令 *)
  WAIT (io.DIn[10]) ;
  (* 抓取工件 *)
  io.DOut[9] := true ;
  DWELL (0.5) ;
  (* 重置机器人抓取等待状态 *)
  io.DOut[10] := false ;
  (* 运动到抓取缓冲点 *)
  MJOINT (*, v500, fine, tool0) ;
  (* 运动到码垛进入点 *)
  MJOINT (pallet.enterPalletPos, v500, fine, tool0, wobj1) ;
  (* 更新码垛位置信息 *)
  pallet.update(1, 0, pieceid, 1) ;
  (* 运动到码垛接近点1 *)
  MJOINT (pallet.approachPos1, v500, fine, tool0, wobj1) ;
  (* 运动到码垛接近点2 *)
  MJOINT (pallet.approachPos2, v500, fine, tool0, wobj1) ;
  (* 运动到放置工件的正上方点1 *)
  MJOINT (pallet.overPiecePos1, v500, fine, tool0, wobj1) ;
  (* 运动到放置工件的正上方点2 *)
  MJOINT (pallet.overPiecePos2, v500, fine, tool0, wobj1) ;
  (* 运动到放置工件点 *)
  MJOINT (pallet.piecePos, v50, fine, tool0, wobj1) ;
  (* 松开工件 *)
  DWELL (0.3) ;
  io.DOut[9] := false ;
  (* 离开,运动到放置工件的正上方点2 *)
  MJOINT (pallet.overPiecePos2, v500, fine, tool0, wobj1) ;
  (* 离开,运动到放置工件的正上方点1 *)
  MJOINT (pallet.overPiecePos1, v500, fine, tool0, wobj1) ;
  (* 运动到码垛接近点2 *)
  MJOINT (pallet.approachPos2, v500, fine, tool0, wobj1) ;
  (* 运动到码垛接近点1 *)
  MJOINT (pallet.approachPos1, v500, fine, tool0, wobj1) ;
  (* 更新工件序号 *)
  pieceid := pieceid + 1 ;
END_WHILE ;
```

图 14-2 码垛程序示例

第 15 章　监控

15.1 本章简介

本章主要介绍机器人监控功能的使用，主要包括位置、IO、驱动器、现场总线部分。

15.2 位置

表 14-1 位置数据查看步骤

步骤	图片	描述
1.打开位置监控界面。		点击状态栏下“**监控**”按钮。 点击下拉菜单中**“位置”**。
2.查看各坐标系下的机器人位置。	 	界面根据当前设置的坐标系，显示两组机器人坐标。左边一列可以选择关节坐标系或者附加轴（当机器人没有附加轴时，附加轴选项不可选择），右边一列可以选择机器人坐标系或者用户坐标系。 同时，坐标系显示会根据当前机器人运行坐标系自动切换。比如机器人切换当前坐标系为用户，则监控坐标系自动变成用户坐标系；如果此时需要查看机器人坐标系，通过选择框可以选择机器人坐标系，查看机器人坐标系下的值。

3.切换工具/用户坐标。		左边的工具/用户坐标系选择框，可以选择不同的工具/用户坐标系，点击设置，则将选择的坐标系激活。 比如选择 tool2 和 wobj1，分别点击**“设置”**按钮，则修改当前坐标系为 tool2 和 wobj1。
4.快速复位	 	点击“快速复位”按钮展开快速复位界面。 回零位：按下“回零位”按钮使机器人自动移至零位，移动过程中松开按钮机器人将停止移动（自动模式下需上伺服使用，手动模式下需按下手压使用） 移至自定义位置：按下“移至自定义位置”按钮使机器人自动移至自定义的位置。操作同“回零位”按钮。 示教位置：将机器人移至自定义的安全位置，点击“示教位置”按钮，此时当前自定义位置将显示当前记录的位置。 点击快速复位标题栏区域关闭快速复位界面。

注：位置监控中慢速、步进功能见点动操作章节。

15.3 IO 监控

通过监控 IO 可实现对数字量 IO 和模拟量 IO 的值监控、手动模式下的强制操作及 IO 信息注释。

15.3.1 数字 IO 监控

在数字量 IO 监控界面能够查看当前数字量 IO 的状态；能对非系统占用的 IO 在手动模式下进行强制操作，切换到自动模式会复位强制状态；能查看已被占用 IO 的信息，能对未被系统、功能占用

的 IO 进行自由注释。

表 14-2 数字量 IO 监控

步骤	图片	描述
1. 打开 I0 监控界面。		1）点击“监控”。 2）点击“IO”。
2. 查看数字量 I0 信号的输入输出状态。		红色框 1 中，  表示状态为 true；红色框 2 中，  表示状态为 false。
3.可以对非系统占用的数字量IO进行强制操作。		红色框 1 中，进行强制操作，单击  变为  ，表示当前状态由 true，强制变为 false。 红色框 2 中，进行强制操作，单击  变为  ，表示当前的状态由 false 强制变为 true。

4.系统占用的数字量 IO 无法进行强制操作。		
5.查看系统占用的数字量 IO 信息。		被系统占用的数字量 IO 地址是固定的，不能被自由配置，将会在描述栏显示：“系统：xxx”信息。
6.功能 IO 信息显示。		在“IO 设置”APP 的“功能 IO 配置”中能够对“通用功能”、“安全监控”、“附加轴”、“冲压”和“高级码垛”等信号进行自由配置，详细步骤见第 11 章 11.4 节功能 IO 配置，保存成功后，即可在 IO 监控中查看配置信息。

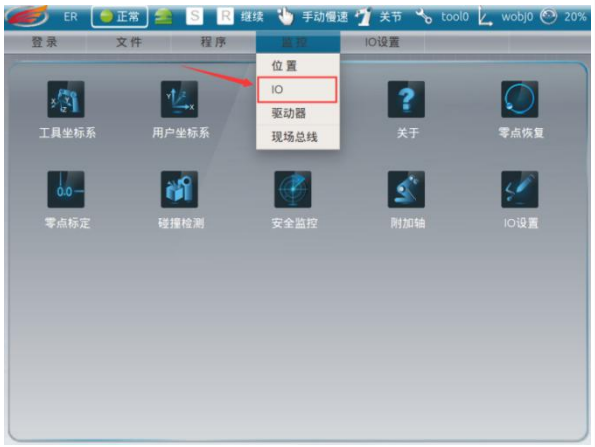
7.用户 IO 查看、编辑及修改注释信息。	 	1）点击“编辑”按钮。 2）点击要编辑的描述栏，系统和功能 IO 信息不能被修改，只能修改、编辑用户 IO 信息。 3）点击“保存”按钮。
8.用户一键释放 IO 强制状态的功能。		点击“复位所有强制 IO”可以将强制操作的状态恢复到操作之前。

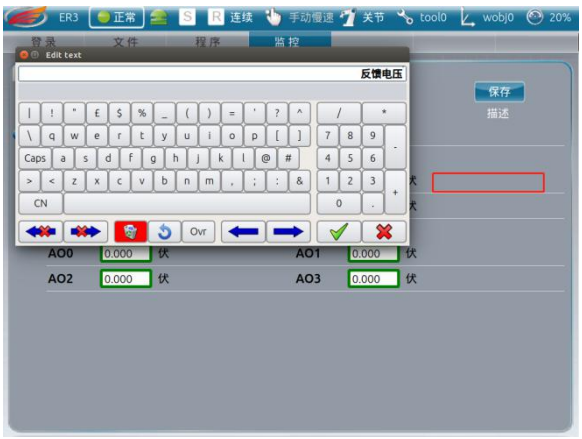
15.3.2 模拟量 IO 监控

在模拟量 IO 监控界面能够查看模拟量模块的类型（本地或远程）、模块的数量、通道数量、通道类型（电流或电压）及通道数值；在手动模式下可对输出通道的数值进行强制操作，切换到自动模式时，会自动复位；能对通道进行注释信息。

表 14-2 模拟量 IO 监控

步骤	图片	描述

1. 打开 IO 监控界面。		1）点击“监控”。 2）点击“IO”。
2. 查看模拟量 IO 按钮，进入模拟量监控界面。		编号 1：进入模拟量监控界面的按钮； 编号 2：编辑和保存通道注释的按钮； 编号 3：显示或强制通道数值 编号 4：单位；电流通道为毫安，电压通道为伏；具体通道类型可在“IO 设置”的“远程 IO 配置”中进行设置； 编号 5：显示和设置通道注释。
3. 输出通道值强制和恢复		步骤 1：点击要强制的编辑框，会有弹框提示，点击“是”； 步骤 2：在键盘中输入要强制的值，需在限制的范围内；确认后，编辑框由 0.000 变为了 3.000，表示强制成功； 步骤 3：再次点击编辑框，在弹框提示中点击“是”，或直接切换到自动模式，会取消强制值。 （注：**只有手动模式下对**

		**输出通道才能强制操作**）
4. 通道注释	 	步骤 1：点击“编辑”按钮； 步骤 2：点击要注释的编辑框； 步骤 3：在键盘中输入要注释的信息； 步骤 4：点击“保存”按钮，会将注释信息保存到控制器中。

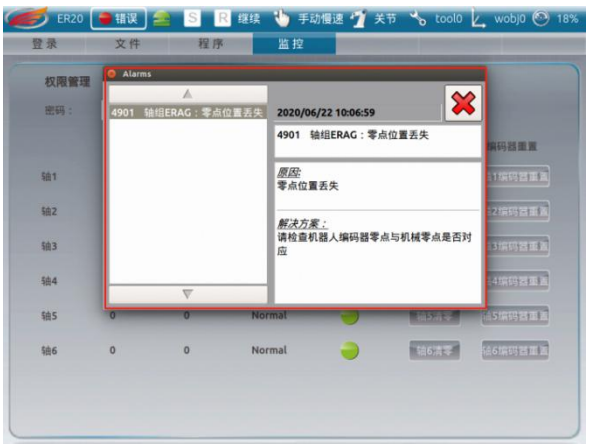
15.4 驱动器

在驱动器监控界面能够查看各轴的状态字、报警代码、描述、零位状态，能够进行重置零位、编码器重置等操作；目前 ER3B-C30 机型，无软抱闸操作。

15.4.1 重置零位

表 14-3 重置零位

步骤	图片	描述
1.点击任务栏“监控”。		
2.点击“驱动器”，进入驱动器监控界面。		红色框 1 内为权限管理，如果要操作驱动器监控界面，则需输入“1975”密码，进入后才能进行操作。 红色框 2、3、4、5 为各轴的状态、报警代码及描述。 红框 6 为各轴重置零点操作。

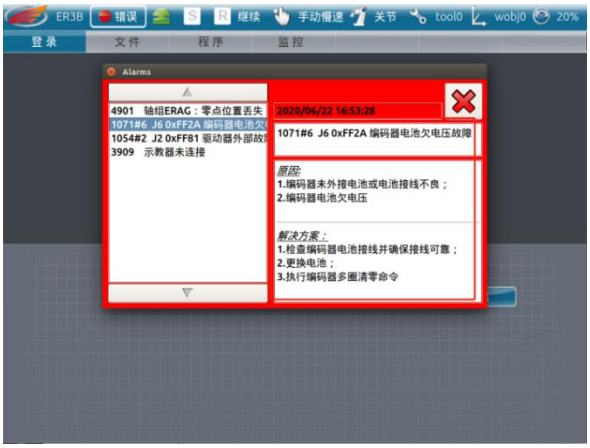
3.当报零点位置丢失警告，需要进行零点重置操作。		零位丢失除了报警告外，零点丢失的轴“零位状态”为  。
4.将零点丢失的某轴移动到机械零位，然后输入权限密码，点击“轴 X 清零”		
5.零点重置成功		零点重置成功后，零点状态由  变为  。

15.4.2 编码器重置

当遇到编码器电池欠电压报警或警告时需要进行编码器重置操作。

表 14-4 编码器重置

步骤	图片	描述

1.界面上有编码器电池欠电压报告或警告		在驱动器界面，输入权限密码，进入操作界面。
2.排查完故障后进行编码器重置操作		编码器重置成功后，会导致零位丢失，故还需将机器人移动到机械零位，进行零位重置。

15.4.3 软抱闸操作

表 14-5 软抱闸操作

步骤	图片	描述
1. 抱闸操作		在权限管理中“进入”才能进行抱闸操作；红框 1☑表示激活抱闸，才能对各轴的抱闸进行操作；红框 2 表示各轴的抱闸操作及状态，

		☑表示打开抱闸， 表示打开抱闸成功。

15.5 现场总线

15.5.1 现场总线数据监控

现场总线监控主要是用来查看总线通讯中的数据，同时可以设置数据的输出值。

目前主要包括 ModbusTCP、Profibus-DP(仅支持 Robox RP1 控制器)、Profinet(仅支持 Robox RP2 Pro 控制器)、FinsTCP、EtherCAT 的通讯协议数据监控，其中 FinsTCP 和 EtherCAT 依据需求选择是否在界面中显示。ModbusTCP 分为可读可写部分和只读只写部分，只读只 写部分的输入输出各自包括 64 个 bool、6 个 int、24 个 float 类型的数据；可读可写部分包 括 64 个 bool、64 个 int、16 个 float；Profibus-DP、Profinet、FinsTCP、EtherCAT 的输入输出各自包括 64 个 bool、6 个 int、24 个 float 类型的数据，Ethernet/IP 功能分为数据接收端与发送端，两端各留有 64 个 BOOL, 6 个 int, 24 个 float 数据接口。

15.5.2 总线数据查看

表 14-6 现场总线数据查看步骤

步骤	图片	描述
1.打开现场总线监控界面。		点击状态栏下“监控”按钮。 点击下拉菜单中“现场总线”。

2.选中需要查看的变量。		在界面中选择需要查看的通讯协议的数据。这里以Modbus_ro_wo为例。 选择“名称”下面加（减）号可以展开（收起）数据列表。
3.查看变量数据		展开数据中可以查看以下数据： 第 1 列是数据名称； 第 2 列数据类型； 第 3 列是数据的值； 第 4 列是对应的示教器程序中的变量名称。

15.5.3 设置总线数据输出

步骤	图片	描述
1.打开数据监控列表的输出部分		打开数据监控列表的输出部分。 点击需要输出的变量的“值”这一列，则弹出输出框。以一个 Float 变量为例。

2.设置输出数据的值		设置输出的数据。 点击“√”则数据输出；点击“×”取消输出。
3.输出数据		数据输出如图所示。

15.5.4 Modbus-tcp 功能

15.5.4.1 综述

Modbus Tcp 的 IP 地址即为控制器设置地址，端口号固定为 502。功能分为两个部分：可读可写部分和只读只写部分。其中只读只写部分又分为机器人为接收端和发送端。

15.5.4.2 只读只写变量

只读只写部分按协议内容分为系统协议和用户协议。

系统协议内容，这部分由开发人员进行配置及维护，主要是读取及设置机器人状态及运动参数等。用户只能通过固定的地址对固定的变量进行读写。

用户层协议内容，该部分有终端用户自行配置及维护，该部分工程只有在使用机器人语言编程中可以使用，不支持在工艺包中使用。

接收端和发送端各留有 64 个 BOOL, 6 个 int, 24 个 float 数据接口，终端用户可以通过示教器编写程序读写主站 PLC 的数据。具体的地址与变量映射关系如表 2-1 和表 2-2 所示。

表 14-7 ModbusTcp 接收端协议

变量分类	物理地址	单位	字节数	子单位	备注
系统变量	40101	I16	2	BOOL	Bit0: 上/下伺服（脉冲）
					Bit1: 运行（脉冲）
					Bit2: 停止（脉冲）
					Bit3: 清除报警（脉冲）
					Bit4: 加载程序（脉冲）
					Bit5: 重新开始（程序回到第一行）（脉冲）
					Bit6: Plc 报警（高电平）
					Bit7: 伺服准备确认（脉冲）
					Bit8: 机器人位置类型
					Bit9：程序预约添加确认（脉冲）
					Bit10：程序预约删除确认（脉冲）
					Bit11：预约程序启动（脉冲）
					Bit12：伺服使能（脉冲）
					Bit13：取消伺服使能（脉冲）
	40102	I16	2	BOOL	系统预留 BOOL 变量，用户不可用
	40103	I16	2	I16	设置机器人速度
	40104	I16	2	I16	加载目标程序号，例：首先设置目标程序号为 2，然后给 40101 的 Bit4 高电平触发信号，完成程序加载（在程序运行过程中不可加载）
	40105	I16	2	I16	附加轴轴号选择（1：七轴 2：八轴 3：九轴 4：十轴）
	40106	I16	2	I16	附加轴速度设定
	40107~40110	I16	2 * 4	I16	系统预留 Int 变量，用户不可用
	40111~40134	FLOAT	4 * 12	FLOAT	系统预留 Float 变量，用户不可用

用户变量	40135	I16	2	BOOL	Bit0~Bit15 对应 fidbus.mtcp_ro_b[0] ~ fidbus.mtcp_ro_b[15]
	40136	I16	2	BOOL	Bit0~Bit15 对应 fidbus.mtcp_ro_b[16] ~ fidbus.mtcp_ro_b[31]
	40137	I16	2	BOOL	Bit0~Bit15 对应 fidbus.mtcp_ro_b[32] ~ fidbus.mtcp_ro_b[47]
	40138	I16	2	BOOL	Bit0~Bit15 对应 fidbus.mtcp_ro_b[48] ~ fidbus.mtcp_ro_b[63]
	40139	I16	2	I16	对应 fidbus.mtcp_ro_i[0]
	40140	I16	2	I16	对应 fidbus.mtcp_ro_i[1]
	40141	I16	2	I16	对应 fidbus.mtcp_ro_i[2]
	40142	I16	2	I16	对应 fidbus.mtcp_ro_i[3]
	40143	I16	2	I16	对应 fidbus.mtcp_ro_i[4]
	40144	I16	2	I16	对应 fidbus.mtcp_ro_i[5]
	40145-40146	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[0]
	40147-40148	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[1]
	40149-40150	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[2]
	40151-40152	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[3]
	40153-40154	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[4]
	40155-40156	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[5]
	40157-40158	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[6]
	40159-40160	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[7]
	40161-40162	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[8]
	40163-40164	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[9]
	40165-40166	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[10]
	40167-40168	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[11]
	40169-40170	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[12]
	40171-40172	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[13]
	40173-40174	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[14]
	40175-40176	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[15]
	40177-40178	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[16]
	40179-40180	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[17]
	40181-40182	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[18]
	40183-40184	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[19]
	40185-40186	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[20]
	40187-40188	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[21]
	40189-40190	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[22]
	40191-40192	FLOAT	4	FLOAT	对应 fidbus.mtcp_ro_r[23]

表 14-8　ModbusTcp 发送端协议

变量分类	物理地址	单位	字节数	子单位	备注
系统变量	40001	I16	2	BOOL	Bit0: 手动状态
					Bit1: 自动状态
					Bit2: 远程状态
					Bit3: 伺服状态
					Bit4: 报警状态
					Bit5: 急停状态
					Bit6: 程序运行状态
					Bit7: 安全位置 1 状态
					Bit8: 安全位置 2 状态
					Bit9: 安全位置 3 状态
					Bit10: 安全位置 4 状态
					Bit11: 程序加载状态
					Bit12: 伺服准备状态
					Bit13: 程序预约激活状态
					Bit14: 程序复位状态（程序重新开始）
	40002	I16	2	BOOL	Bit0: 安全位置 5 状态
					Bit1: 安全位置 6 状态
					Bit2: 安全位置 7 状态
					Bit3: 安全位置 8 状态
	40003	I16	2	I16	机器人运行速度(全局)
	40004	I16	2	I16	报警代码 1
	40005	I16	2	I16	报警代码 2
	40006	I16	2	I16	程序号（用于反馈加载目标程序是否完成，例：在 Plc 端加载程序号为 2 的程序，如果加载完成，则程序号反馈为 2，否则为其他值）
	40007	I16	2	I16	预约程序预约状态
	40008	I16	2	I16	预约程序运行状态
	40009~40010	I16	2 * 2	I16	系统预留 Int 变量，用户不可用
	40011~40022	FLOAT	2 * 6	FLOAT	J1~J6 关节角度值
	40023~40030	FLOAT	2 * 4	FLOAT	J7~J10(附加轴 1-4)关节角度值
	40031~40033	FLOAT		FLOAT	系统预留 Float 变量，用户不可用
用户变量	40035	I16	2	BOOL	Bit0~Bit15 对应 fidbus.mtcp_wo_b[0] ~ fidbus.mtcp_wo_b[15]
	40036	I16	2	BOOL	Bit0~Bit15 对应 fidbus.mtcp_wo_b[16] ~ fidbus.mtcp_wo_b[31]
	40037	I16	2	BOOL	Bit0~Bit15 对应 fidbus.mtcp_wo_b[32] ~ fidbus.mtcp_wo_b[47]
	40038	I16	2	BOOL	Bit0~Bit15 对应 fidbus.mtcp_wo_b[48] ~ fidbus.mtcp_wo_b[63]

	40039	I16	2	I16	对应 fidbus.mtcp_wo_i[0]
	40040	I16	2	I16	对应 fidbus.mtcp_wo_i[1]
	40041	I16	2	I16	对应 fidbus.mtcp_wo_i[2]
	40042	I16	2	I16	对应 fidbus.mtcp_wo_i[3]
	40043	I16	2	I16	对应 fidbus.mtcp_wo_i[4]
	40044	I16	2	I16	对应 fidbus.mtcp_wo_i[5]
	40045-40046	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[0]
	40047-40048	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[1]
	40049-40050	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[2]
	40051-40052	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[3]
	40053-40054	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[4]
	40055-40056	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[5]
	40057-40058	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[6]
	40059-40060	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[7]
	40061-40062	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[8]
	40063-40064	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[9]
	40065-40066	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[10]
	40067-40068	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[11]
	40069-40070	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[12]
	40071-40072	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[13]
	40073-40074	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[14]
	40075-40076	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[15]
	40077-40078	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[16]
	40079-40080	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[17]
	40081-40082	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[18]
	40083-40084	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[19]
	40085-40086	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[20]
	40087-40088	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[21]
	40089-40090	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[22]
	40091-40092	FLOAT	4	FLOAT	对应 fidbus.mtcp_wo_r[23]

15.5.4.3 可读可写变量

具体的地址与变量映射关系如表 2-3 所示。

表 14-9 ModbusTcp 可读可写部分协议

变量分类	物理地址	单位	字节数	子单位	备注
用户变量	40301	I16	2	BOOL	Bit0~Bit15 对应 fidbus.mtcp_rw_b[0] ~ fidbus.mtcp_rw_b[15]
	40302	I16	2	BOOL	Bit0~Bit15 对应 fidbus.mtcp_rw_b[16] ~ fidbus.mtcp_rw_b[31]
	40303	I16	2	BOOL	Bit0~Bit15 对应 fidbus.mtcp_rw_b[32] ~

					fidbus.mtcp_rw_b[47]
	40304	I16	2	BOOL	Bit0~Bit15 对应 fidbus.mtcp_rw_b[48] ~ fidbus.mtcp_rw_b[63]
	40305	I16	2	I16	对应 fidbus.mtcp_rw_i[0]
	40306	I16	2	I16	对应 fidbus.mtcp_rw_i[1]
	...	...	...	...	...
	40367	I16	2	I16	对应 fidbus.mtcp_rw_i[62]
	40368	I16	2	I16	对应 fidbus.mtcp_rw_i[63]
	40369 - 40370	FLOAT	4	FLOAT	对应 fidbus.mtcp_rw_r[0]
	40371 - 40372	FLOAT	4	FLOAT	对应 fidbus.mtcp_rw_r[1]
	...	...	...	...	...
	40397- 40398	FLOAT	4	FLOAT	对应 fidbus.mtcp_rw_r[14]
	40399 -40400	FLOAT	4	FLOAT	对应 fidbus.mtcp_rw_r[15]

15.5.5 Ethercat 功能

15.5.5.1 综述

目前机器人仅支持 EtherCAT 协议，用户可通过 EtherCAT 协议转为 CCLink 等其他协议，EtherCAT 功能分为数据接收端与发送端，两端各留有 64 个 BOOL, 6 个 int, 24 个 float 数据 接口，终端用户可以通过示教器编写程序读写主站 PLC 的数据。具体的地址与变量映射关系如表 3-1 和表 3-2 所示。

注：ANYBUS_ECAT 是通用的网关协议转换功能，硬件需安装 ANYBUS 网关。

15.5.5.2 PLC 到机器人

PLC 到机器人，PLC 作为发送端，机器人作接收端。接收数据主要包括系统变量和 TPU 变量。其中 Bus_Get[0]-Bus_Get[191]为数据接收的 192 BYTE。

表 14-10 EtherCAT 接收数据协议

变量分类	物理地址	单位	字节数	子单位	备注
系统变量	Bus_Get[0]-[1]	I16	2	BOOL	Bit0: 上/下伺服（脉冲）
					Bit1: 运行程序（脉冲）
					Bit2: 暂停程序（脉冲）
					Bit3: 清除报警（脉冲）
					Bit4: 加载程序（脉冲）
					Bit5: 重新开始（程序回到第一行）（脉冲）
					Bit6: Plc 报警（高电平）
					Bit7：伺服准备确认（脉冲）
					Bit8：机器人位置类型
					Bit9：程序预约添加确认
					Bit10：程序预约删除确认
					Bit11：预约程序启动（脉冲）
					Bit12：伺服使能（脉冲）
					Bit13：取消伺服使能（脉冲）
	Bus_Get[2]-[3]	I16	2	BOOL	系统预留 BOOL 变量，用户不可以使用
	Bus_Get[4]-[5]	I16	2	I16	运行速度（全局）
	Bus_Get[6]-[7]	I16	2	I16	加载目标程序号，例：首先设置目标程序号为 2，然后给 Bus_Get[0]-[1]的 Bit4 高电平触发信号，完成程序加载（在程序运行过程中不可加载）。
	Bus_Get[8]-[9]	I16	2	I16	附加轴轴号选择（1：七轴 2：八轴 3：九轴 4：十轴）
	Bus_Get[10]-[11]	I16	2	I16	附加轴速度设定
	Bus_Get[12]-[19]	I16	8	I16	系统预留 I16 类型变量，用户不可以使用
用户变量	Bus_Get[20]-[21]	I16	2	BOOL	Bit0~Bit15 对应 fidbus.ethcat_ro_b[0]-[15]

	Bus_Get[22]-[23]	I16	2	BOOL	Bit0~Bit15 对应 fidbus.ethcat_ro_b[16]-[31]
	Bus_Get[24]-[25]	I16	2	BOOL	Bit0~Bit15 对应 fidbus.ethcat_ro_b[32]-[47]
	Bus_Get[26]-[27]	I16	2	BOOL	Bit0~Bit15 对应 fidbus.ethcat_ro_b[48]-[63]
	Bus_Get[28]-[29]	I16	2	I16	对应 fidbus.ethcat_ro_i[0]
	Bus_Get[30]-[31]	I16	2	I16	对应 fidbus.ethcat_ro_i[1]
	Bus_Get[32]-[33]	I16	2	I16	对应 fidbus.ethcat_ro_i[2]
	Bus_Get[34]-[35]	I16	2	I16	对应 fidbus.ethcat_ro_i[3]
	Bus_Get[36]-[37]	I16	2	I16	对应 fidbus.ethcat_ro_i[4]
	Bus_Get[38]-[39]	I16	2	I16	对应 fidbus.ethcat_ro_i[5]
系统变量	Bus_Get[40]-[87]	FLOAT	4 * 12	FLOAT	系统预留 FLOAT 类型变量，用户不可以使用
用户变量	Bus_Get[88]-[91]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[0]
	Bus_Get[92]-[95]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[1]
	Bus_Get[96]-[99]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[2]
	Bus_Get[100]-[103]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[3]
	Bus_Get[104]-[107]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[4]
	Bus_Get[108]-[111]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[5]
	Bus_Get[112]-[115]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[6]
	Bus_Get[116]-[119]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[7]
	Bus_Get[120]-[123]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[8]
	Bus_Get[124]-[127]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[9]
	Bus_Get[128]-[131]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[10]
	Bus_Get[132]-[135]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[11]
	Bus_Get[136]-[139]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[12]
	Bus_Get[140]-[143]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[13]
	Bus_Get[144]-[147]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[14]
	Bus_Get[148]-[151]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[15]
	Bus_Get[152]-[155]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[16]
	Bus_Get[156]-[159]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[17]
	Bus_Get[160]-[163]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[18]
	Bus_Get[164]-[167]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[19]
	Bus_Get[168]-[171]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[20]
	Bus_Get[172]-[175]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[21]
	Bus_Get[176]-[179]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[22]
	Bus_Get[180]-[183]	FLOAT	4	FLOAT	对应 fidbus.ethcat_ro_r[23]
	Bus_Get[184]-[191]	/	/	/	未定义

15.5.5.3 机器人到 PLC

机器人到 PLC，机器人作为发送端，PLC 作为接收端。发送数据主要包括系统变量和 TPU 变量。其中 Bus_Set[0] - Bus_Set[191]为数据发送的 192 BYTE。

表 14-11 EtherCAT 发送数据协议

变量分类	物理地址	单位	字节数	子单位	备注
系统变量	Bus_Set[0]-[1]	I16	2	BOOL	Bit0: 手动状态
					Bit1: 自动状态
					Bit2: 远程状态
					Bit3: 伺服状态
					Bit4: 报警状态
					Bit5: 急停状态
					Bit6: 程序运行状态
					Bit7: 安全位置 1 状态
					Bit8: 安全位置 2 状态
					Bit9: 安全位置 3 状态
					Bit10: 安全位置 4 状态
					Bit11: 加载程序状态
					Bit12: 伺服确认状态
					Bit13: 程序预约激活状态
					Bit14: 程序复位状态（程序重新开始）
	Bus_Set[2]-[3]	I16	2	BOOL	Bit0: 安全位置 5 状态
					Bit1: 安全位置 6 状态
					Bit2: 安全位置 7 状态
					Bit3: 安全位置 8 状态
	Bus_Set[4]-[5]	I16	2	I16	运行速度（全局）
	Bus_Set[6]-[7]	I16	2	I16	报警代码 1
	Bus_Set[8]-[9]	I16	2	I16	报警代码 2
	Bus_Set[10]-[11]	I16	2	I16	程序号（用于反馈加载目标程序是否完成，例：在 Plc 端加载程序号为 2 的程序，如果加载完成，则程序号反馈为 2，否则为其他值）
	Bus_Set[12]-[13]	I16	2	I16	预约程序预约状态
	Bus_Set[14]-[15]	I16	2	I16	预约程序运行状态
	Bus_Set[16]-[19]	I16	4	I16	系统预留 I16 类型变量，用户不可用
用户变量	Bus_Set[20]-[21]	I16	2	BOOL	Bit0~Bit15 对应 fidbus.ethcat_wo_b[0]-[15]
	Bus_Set[22]-[23]	I16	2	BOOL	Bit0~Bit15 对应 fidbus.ethcat_wo_b[16]-[31]
	Bus_Set[24]-[25]	I16	2	BOOL	Bit0~Bit15 对应 fidbus.ethcat_wo_b[32]-[47]

	Bus_Set[26]-[27]	I16	2	BOOL	Bit0~Bit15 对应 fidbus.ethcat_wo_b[48]-[63]
	Bus_Set[28]-[29]	I16	2	I16	对应 fidbus.ethcat_wo_i[0]
	Bus_Set[30]-[31]	I16	2	I16	对应 fidbus.ethcat_wo_i[1]
	Bus_Set[32]-[33]	I16	2	I16	对应 fidbus.ethcat_wo_i[2]
	Bus_Set[34]-[35]	I16	2	I16	对应 fidbus.ethcat_wo_i[3]
	Bus_Set[36]-[37]	I16	2	I16	对应 fidbus.ethcat_wo_i[4]
	Bus_Set[38]-[39]	I16	2	I16	对应 fidbus.ethcat_wo_i[5]
系统变量	Bus_Set[40]-[63]	FLOAT	4 * 6	FLOAT	J1~J6 关节角度值/笛卡尔空间位姿
	Bus_Set[64]-[79]	FLOAT	4 * 4	FLOAT	J7~J10(附加轴 1-4)关节角度值
	Bus_Set[80]-[87]	FLOAT	4 * 2	FLOAT	系统预留 FLOAT 类型变量,用户不可用
用户变量	Bus_Set[88]-[91]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[0]
	Bus_Set[92]-[95]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[1]
	Bus_Set[96]-[99]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[2]
	Bus_Set[100]-[103]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[3]
	Bus_Set[104]-[107]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[4]
	Bus_Set[108]-[111]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[5]
	Bus_Set[112]-[115]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[6]
	Bus_Set[116]-[119]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[7]
	Bus_Set[120]-[123]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[8]
	Bus_Set[124]-[127]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[9]
	Bus_Set[128]-[131]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[10]
	Bus_Set[132]-[135]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[11]
	Bus_Set[136]-[139]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[12]
	Bus_Set[140]-[143]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[13]
	Bus_Set[144]-[147]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[14]
	Bus_Set[148]-[151]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[15]
	Bus_Set[152]-[155]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[16]
	Bus_Set[156]-[159]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[17]
	Bus_Set[160]-[163]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[18]
	Bus_Set[164]-[167]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[19]
	Bus_Set[168]-[171]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[20]
	Bus_Get[172]-[175]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[21]
	Bus_Get[176]-[179]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[22]
	Bus_Get[180]-[183]	FLOAT	4	FLOAT	对应 fidbus.ethcat_wo_r[23]
	Bus_Get[184]-[191]	/	/	/	未定义

15.5.6 FinsTcp 功能

15.5.6.1 综述

FinsTCP 功能使用 FinsTCP 通讯协议进行数据的传递，分为数据接收端与发送端，两端各留有 64 个 BOOL, 6 个 int, 24 个 float 数据接口，IO 内存地址（DM 区）用户可进行自定义配置，终端用户可以通过示教器编写程序读写主站 PLC 的数据。具体的地址与变量映射关系如表 4-1 和表 4-2 所示。

15.5.6.2 PLC 到机器人

PLC 到机器人，PLC 作为发送端，机器人作接收端。接收数据主要包括系统变量和 TPU 变量。其中 ReadDmAddress + 0 ~ ReadDmAddress + 94 为数据接收的 96 WORD。
(ReadDmAddress 对应示教器界面上现场总线监控 FinsTCP_RO_WO 只读部分 Dm 全称)

表 14-12 FinsTCP 接收数据协议

变量分类	物理地址	单位	字数	子单位	备注
系统变量	ReadDmAddress + 0	I16	1	BOOL	Bit0: 上/下伺服（脉冲）
					Bit1: 运行程序（脉冲）
					Bit2: 暂停程序（脉冲）
					Bit3: 清除报警（脉冲）
					Bit4: 加载程序（脉冲）
					Bit5: 重新开始（程序回到第一行）（脉冲）
					Bit6: Plc 报警（高电平）
					Bit7：伺服准备确认（脉冲）
	ReadDmAddress + 1	I16	1	BOOL	系统预留 BOOL 变量，用户不可以使用
	ReadDmAddress + 2	I16	1	I16	运行速度（全局）
	ReadDmAddress + 3	I16	1	I16	加载目标程序号，例：首先设置目标程序号为 2，然后给 ReadDmAddress + 0 的 Bit4 高电平触发信号，完成程序加载（在程序运行过程中不可加载）。
	ReadDmAddress + 4 ~ReadDmAddress + 9	I16	1 * 6	I16	系统预留 I16 类型变量，用户不可以使用
	ReadDmAddress + 10 ~ReadDmAddress + 32	FLOAT	2 * 12	FLOAT	系统预留 FLOAT 类型变量，用户不可以使用
用户变量	ReadDmAddress + 34	I16	1	BOOL	Bit0~Bit15 对应 fins_tcp_ro_b[0]-[15]
	ReadDmAddress + 35	I16	1	BOOL	Bit0~Bit15 对应 fins_tcp_ro_b[16]-[31]
	ReadDmAddress + 36	I16	1	BOOL	Bit0~Bit15 对应 fins_tcp_ro_b[32]-[47]
	ReadDmAddress + 37	I16	1	BOOL	Bit0~Bit15 对应 fins_tcp_ro_b[48]-[63]
	ReadDmAddress + 38	I16	1	I16	对应 fins_tcp_ro_i[0]
	ReadDmAddress + 39	I16	1	I16	对应 fins_tcp_ro_i[1]
	ReadDmAddress + 40	I16	1	I16	对应 fins_tcp_ro_i[2]

	ReadDmAddress + 41	I16	1	I16	对应 fins_tcp_ro_i[3]
	ReadDmAddress + 42	I16	1	I16	对应 fins_tcp_ro_i[4]
	ReadDmAddress + 43	I16	1	I16	对应 fins_tcp_ro_i[5]
	ReadDmAddress + 44	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[0]
	ReadDmAddress + 46	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[1]
	ReadDmAddress + 48	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[2]
	ReadDmAddress + 50	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[3]
	ReadDmAddress + 52	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[4]
	ReadDmAddress + 54	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[5]
	ReadDmAddress + 56	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[6]
	ReadDmAddress + 58	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[7]
	ReadDmAddress + 60	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[8]
	ReadDmAddress + 62	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[9]
	ReadDmAddress + 64	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[10]
	ReadDmAddress + 66	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[11]
	ReadDmAddress + 68	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[12]
	ReadDmAddress + 70	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[13]
	ReadDmAddress + 72	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[14]
	ReadDmAddress + 74	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[15]
	ReadDmAddress + 76	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[16]
	ReadDmAddress + 78	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[17]
	ReadDmAddress + 80	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[18]

	ReadDmAddress + 82	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[19]
	ReadDmAddress + 84	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[20]
	ReadDmAddress + 86	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[21]
	ReadDmAddress + 88	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[22]
	ReadDmAddress + 90	FLOAT	2	FLOAT	对应 fins_tcp_ro_r[23]
系统变量	ReadDmAddress + 92~ReadDmAddress + 94	/	/	/	未定义

15.5.6.3 机器人到 PLC

机器人到 PLC，机器人作为发送端，PLC 作为接收端。发送数据主要包括系统变量和 TPU 变量。其中 WriteDmAddress + 0 ~ WriteDmAddress + 94 为数据接收的 96 WORD (WriteDmAddress 对应示教器界面上现场总线监控 FinsTCP_RO_WO 只写部分 Dm 全称)。

表 14-13　FinsTCP 发送数据协议

变量分类	物理地址	单位	字数	子单位	备注
系统变量	WriteDmAddress + 0	I16	1	BOOL	Bit0: 手动状态
					Bit1: 自动状态
					Bit2: 远程状态
					Bit3: 伺服状态
					Bit4: 报警状态
					Bit5: 急停状态
					Bit6: 程序运行状态
					Bit7: 安全位置 1 状态
					Bit8: 安全位置 2 状态
					Bit9: 安全位置 3 状态
					Bit10: 安全位置 4 状态
	WriteDmAddress + 1	I16	1	BOOL	系统预留 BOOL 类型变量，用户不可用
	WriteDmAddress + 2	I16	1	I16	运行速度（全局）
	WriteDmAddress + 3	I16	1	I16	报警代码 1
	WriteDmAddress + 4	I16	1	I16	报警代码 2
	WriteDmAddress + 5	I16	1	I16	程序号（用于反馈加载目标程序是否完成，例：在 Plc 端加载程序号为 2 的程序，如果加载完成，则程序号反馈为 2，否则为其他值）
	WriteDmAddress + 6 ~WriteDmAddress + 9	I16	1 * 4	I16	系统预留 I16 类型变量，用户不可用
	WriteDmAddress + 10	FLOAT	2 * 6	FLOAT	J1~J6 关节角度值/笛卡尔空间位姿

	~WriteDmAddress + 20				
	WriteDmAddress + 22 ~WriteDmAddress + 28	FLOAT	2 * 4	FLOAT	J7~J10(附加轴 1-4)关节角度值
	WriteDmAddress + 30 ~WriteDmAddress + 32	FLOAT	2 * 2	FLOAT	系统预留 FLOAT 类型变量,用户不可用
用户变量	WriteDmAddress + 34	I16	1	BOOL	Bit0~Bit15 对应 fins_tcp_wo_b[0]-[15]
	WriteDmAddress + 35	I16	1	BOOL	Bit0~Bit15 对应 fins_tcp_wo_b[16]-[31]
	WriteDmAddress + 36	I16	1	BOOL	Bit0~Bit15 对应 fins_tcp_wo_b[32]-[47]
	WriteDmAddress + 37	I16	1	BOOL	Bit0~Bit15 对应 fins_tcp_wo_b[48]-[63]
	WriteDmAddress + 38	I16	1	I16	对应 fins_tcp_wo_i[0]
	WriteDmAddress + 39	I16	1	I16	对应 fins_tcp_wo_i[1]
	WriteDmAddress + 40	I16	1	I16	对应 fins_tcp_wo_i[2]
	WriteDmAddress + 41	I16	1	I16	对应 fins_tcp_wo_i[3]
	WriteDmAddress + 42	I16	1	I16	对应 fins_tcp_wo_i[4]
	WriteDmAddress + 43	I16	1	I16	对应 fins_tcp_wo_i[5]
用户变量	WriteDmAddress + 44	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[0]
	WriteDmAddress + 46	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[1]
	WriteDmAddress + 48	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[2]
	WriteDmAddress + 50	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[3]
	WriteDmAddress + 52	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[4]
	WriteDmAddress + 54	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[5]
	WriteDmAddress + 56	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[6]
	WriteDmAddress + 58	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[7]
	WriteDmAddress + 60	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[8]
	WriteDmAddress + 62	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[9]
	WriteDmAddress + 64	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[10]
	WriteDmAddress + 66	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[11]
	WriteDmAddress + 68	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[12]
	WriteDmAddress + 70	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[13]
	WriteDmAddress + 72	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[14]
	WriteDmAddress + 74	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[15]
	WriteDmAddress + 76	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[16]
	WriteDmAddress + 78	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[17]
	WriteDmAddress + 80	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[18]
	WriteDmAddress + 82	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[19]
	WriteDmAddress + 84	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[20]
	WriteDmAddress + 86	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[21]
	WriteDmAddress + 88	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[22]
	WriteDmAddress + 90	FLOAT	2	FLOAT	对应 fins_tcp_wo_r[23]
系统变量	WriteDmAddress +	/	/	/	未定义

	92~WriteDmAddress + 94				

15.5.7 Ethernet/IP 功能

15.5.7.1 综述

Ethernet/IP 功能支持机器人做从站（slave），其 IP 地址与控制器一致，控制器默认站号为 0，端口号默认为 2222，数据的存储模式为小端模式 。Ethernet/IP 功能分为数据接收端与发送端，两端各留有 64 个 BOOL, 6 个 int, 24 个 float 数据 接口，终端用户可以通过示教器编写程序读写主站 PLC 的数据。具体的地址与变量映射关系如表 5-1 和表 5-2 所示。

15.5.7.2 PLC 到机器人

PLC 到机器人，PLC 作为发送端，机器人作接收端。接收数据主要包括系统变量和 TPU 变量。其中 Bus_Get[0] - Bus_Get[191]为数据接收为 192 BYTE。

变量分类	物理地址	单位	字数	子单位	备注
系统变量	Bus_Get[0]	I16	1	BOOL	Bit0：上/下伺服（脉冲）
					Bit1：运行程序（脉冲）
					Bit2：暂停程序（脉冲）
					Bit3：清除报警（脉冲）
					Bit4：加载程序（脉冲）
					Bit5：重新开始（程序回到第一行）（脉冲）
					Bit6: Plc 报警（高电平）
					Bit7：伺服准备确认（脉冲）
					Bit8：机器人位置类型
					Bit9：程序预约添加确认
					Bit10：程序预约删除确认
					Bit11：预约程序启动（脉冲）
					Bit12：伺服使能（脉冲）
					Bit13：取消伺服使能（脉冲）
	Bus_Get[1]	I16	1	BOOL	系统预留 BOOL 变量，**用户不可以使用**
	Bus_Get[2]	I16	1	I16	运行速度（全局）
	Bus_Get[3]	I16	1	I16	加载目标程序号，例：首先设置目标程序号为 2，然后给 Bus_Get[0]的 Bit4 高电平触发信号，完成程序加载（在程序运行过程中不可加载）。
	Bus_Get[4]	I16	1	I16	附加轴轴号选择（1：七轴 2：八轴 3：九轴 4：十轴）
	Bus_Get[5]	I16	1	I16	附加轴速度设定
	Bus_Get[6]-[9]	I16	1 * 4	I16	系统预留 I16 类型变量，用户不可以使用
	Bus_Get[10]-[33]	FLOAT	2 * 12	FLOAT	系统预留 FLOAT 类型变量，用户不可以使用
用户变量	Bus_Get[34]	I16	1	BOOL	Bit0~Bit15 对应 fidbus.eip_ro_b[0]-[15]
	Bus_Get[35]	I16	1	BOOL	Bit0~Bit15 对应 fidbus.eip_ro_b[16]-[31]
	Bus_Get[36]	I16	1	BOOL	Bit0~Bit15 对应 fidbus.eip_ro_b[32]-[47]

	Bus_Get[37]	I16	1	BOOL	Bit0~Bit15 对应 fidbus.eip_ro_b[48]-[63]
	Bus_Get[38]	I16	1	I16	对应 fidbus.eip_ro_i[0]
	Bus_Get[39]	I16	1	I16	对应 fidbus.eip_ro_i[1]
	Bus_Get[40]	I16	1	I16	对应 fidbus.eip_ro_i[2]
	Bus_Get[41]	I16	1	I16	对应 fidbus.eip_ro_i[3]
	Bus_Get[42]	I16	1	I16	对应 fidbus.eip_ro_i[4]
	Bus_Get[43]	I16	1	I16	对应 fidbus.eip_ro_i[5]
	Bus_Get[44]-[45]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[0]
	Bus_Get[46]-[47]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[1]
	Bus_Get[48]-[49]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[2]
	Bus_Get[50]-[51]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[3]
	Bus_Get[52]-[53]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[4]
	Bus_Get[54]-[55]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[5]
	Bus_Get[56]-[57]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[6]
	Bus_Get[58]-[59]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[7]
	Bus_Get[60]-[61]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[8]
	Bus_Get[62]-[63]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[9]
	Bus_Get[64]-[65]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[10]
	Bus_Get[66]-[67]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[11]
	Bus_Get[68]-[69]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[12]
	Bus_Get[70]-[71]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[13]
	Bus_Get[72]-[73]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[14]
	Bus_Get[74]-[75]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[15]
	Bus_Get[76]-[77]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[16]
	Bus_Get[78]-[79]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[17]
	Bus_Get[80]-[81]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[18]
	Bus_Get[82]-[83]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[19]
	Bus_Get[84]-[85]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[20]
	Bus_Get[86]-[87]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[21]
	Bus_Get[88]-[89]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[22]
	Bus_Get[90]-[91]	FLOAT	2	FLOAT	对应 fidbus.eip_ro_r[23]

表 14-14　Ethernet/IP 接收数据协议

15.5.7.3 机器人到 PLC

机器人到 PLC，机器人作为发送端，PLC 作为接收端。发送数据主要包括系统变量和 TPU 变量。其中 Bus_Set[0] - Bus_Set[191]为数据发送为 192 BYTE。

表 14-15　Ethernet/IP 发送数据协议

变量分类	物理地址	单位	字数	子单位	备注
系统变量	Bus_Set[0]	I16	1	BOOL	Bit0: 手动状态

					Bit1: 自动状态
					Bit2: 远程状态
					Bit3: 伺服状态
					Bit4: 报警状态
					Bit5: 急停状态
					Bit6: 程序运行状态
					Bit7: 安全位置 1 状态
					Bit8: 安全位置 2 状态
					Bit9: 安全位置 3 状态
					Bit10: 安全位置 4 状态
					Bit11: 加载程序状态
					Bit12: 伺服确认状态
					Bit13: 程序预约激活状态
					Bit14: 程序复位状态（程序重新开始）
	Bus_Set[1]	I16	1	BOOL	Bit0: 安全位置 5 状态
					Bit1: 安全位置 6 状态
					Bit2: 安全位置 7 状态
					Bit3: 安全位置 8 状态
	Bus_Set[2]	I16	1	I16	运行速度（全局）
	Bus_Set[3]	I16	1	I16	报警代码
	Bus_Set[4]	I16	1	I16	报警代码
	Bus_Set[5]	I16	1	I16	程序号（用于反馈加载目标程序是否完成，例：在 Plc 端加载程序号为 2 的程序，如果加载完成，则程序号反馈为 2，否则为其他值）
	Bus_Set[6]	I16	1	I16	预约程序预约状态
	Bus_Set[7]	I16	1	I16	预约程序运行状态
	Bus_Set[8]-Bus_Set[9]	I16	1* 2	I16	系统预留 I16 类型变量，用户不可用
	Bus_Set[10]-[21]	FLOAT	2 * 6	FLOAT	J1~J6 关节角度值/笛卡尔空间位姿
	Bus_Set[23]-[31]	FLOAT	2 * 4	FLOAT	J7~J10(附加轴 1-4)关节角度值
	Bus_Set[32]-[33]	FLOAT	2	FLOAT	系统预留 FLOAT 类型变量,用户不可用
用户变量	Bus_Set[34]	I16	1	BOOL	Bit0~Bit15 对应 fidbus.eip_wo_b[0]-[15]
	Bus_Set[35]	I16	1	BOOL	Bit0~Bit15 对应 fidbus.eip_wo_b[16]-[31]
	Bus_Set[36]	I16	1	BOOL	Bit0~Bit15 对应 fidbus.eip_wo_b[32]-[47]
	Bus_Set[37]	I16	1	BOOL	Bit0~Bit15 对应 fidbus.eip_wo_b[48]-[63]
	Bus_Set[38]	I16	1	I16	对应 fidbus.eip_wo_i[0]
	Bus_Set[39]	I16	1	I16	对应 fidbus.eip_wo_i[1]
	Bus_Set[40]	I16	1	I16	对应 fidbus.eip_wo_i[2]
	Bus_Set[41]	I16	1	I16	对应 fidbus.eip_wo_i[3]

	Bus_Set[42]	I16	1	I16	对应 fidbus.eip_wo_i[4]
	Bus_Set[43]	I16	1	I16	对应 fidbus.eip_wo_i[5]
用户变量	Bus_Set[44]-[45]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[0]
	Bus_Set[46]-[47]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[1]
	Bus_Set[48]-[49]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[2]
	Bus_Set[50]-[51]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[3]
	Bus_Set[52]-[53]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[4]
	Bus_Set[54]-[55]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[5]
	Bus_Set[56]-[57]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[6]
	Bus_Set[58]-[59]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[7]
	Bus_Set[60]-[61]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[8]
	Bus_Set[62]-[63]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[9]
	Bus_Set[64]-[65]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[10]
	Bus_Set[66]-[67]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[11]
	Bus_Set[68]-[69]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[12]
	Bus_Set[70]-[71]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[13]
	Bus_Set[72]-[73]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[14]
	Bus_Set[74]-[75]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[15]
	Bus_Set[76]-[77]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[16]
	Bus_Set[78]-[79]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[17]
	Bus_Set[80]-[81]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[18]
	Bus_Set[82]-[83]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[19]
	Bus_Set[84]-[85]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[20]
	Bus_Get[86]-[87]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[21]
	Bus_Get[88]-[89]	FLOAT	2	FLOAT	对应 fidbus.eip_wo_r[22]
	Bus_Get[90]-[91]	FLOAT	4	FLOAT	对应 fidbus.eip_wo_r[23]

15.5.8 * Profibus_DP/Profinet 功能

15.5.8.1 综述

Profibus_DP/Profinet 功能支持机器人做从站（slave），其地址为 2，包括 6 个 input block 和 6 个 output block，每个 block 包含 32byte，数据类型为 unsigned int（2 byte）或者是 float（4 byte）。数据的存储模式为大端模式，所以当将 unsigned int 拆分成 byte 类型时，注意数据的高低位转换 。

注：Profibus_DP 功能仅支持 Robox RP1 控制器，Profinet 功能仅支持 Robox RP2 Pro 控制器！Robox RP2 Eco 的 TPU 界面中无该显示内容。

15.5.8.2 到机器人

PLC 到机器人，PLC 作为发送端，机器人作接收端。接收数据主要包括系统变量和 TPU 变量。其中 Bus_Get[0]-Bus_Get[191]为数据接收的 192 BYTE。

表 14-16　Profibus/Profinet 接收数据协议

变量分类	物理地址	单位	字节数	子单位	备注
系统变量	Bus_Get[0]-[1]	I16	2	BOOL	Bit0：上/下伺服（脉冲）
					Bit1：运行程序（脉冲）
					Bit2：暂停程序（脉冲）
					Bit3：清除报警（脉冲）
					Bit4：加载程序（脉冲）
					Bit5：重新开始（程序回到第一行）（脉冲）
					Bit6: Plc 报警（高电平）
					Bit7：伺服准备确认（脉冲）
					Bit8：机器人位置类型
					Bit9：程序预约添加确认
					Bit10：程序预约删除确认
					Bit11：预约程序启动（脉冲）
					Bit12：伺服使能（脉冲）
					Bit13：取消伺服使能（脉冲）
	Bus_Get[2]-[3]	I16	2	BOOl	系统预留 BOOL 变量，用户不可以使用
	Bus_Get[4]-[5]	I16	2	I16	运行速度（全局）
	Bus_Get[6]-[7]	I16	2	I16	加载目标程序号，例：首先设置目标程序号为 2，然后给 Bus_Get[0]-[1]的 Bit4 高电平触发信号，完成程序加载（在程序运行过程中不可加载）。
	Bus_Get[8]-[9]	I16	2	I16	附加轴轴号选择（1：七轴 2：八轴 3：九轴 4：十轴）
	Bus_Get[10]-[11]	I16	2	I16	附加轴速度设定
	Bus_Get[12]-[19]	I16	2 * 4	I16	系统预留 I16 类型变量，用户不可以使用
用户变量	Bus_Get[20]-[21]	I16	2	BOOL	Bit0~Bit15 对应

					fidbus.pfb_ro_b[0]-[15]/fidbus.pfn_ro_b[0]-[15]
	Bus_Get[22]-[23]	I16	2	BOOL	Bit0~Bit15 对应 fidbus.pfb_ro_b[16]-[31]/fidbus.pfn_ro_b[16]-[31]
	Bus_Get[24]-[25]	I16	2	BOOL	Bit0~Bit15 对应 fidbus.pfb_ro_b[32]-[47]/fidbus.pfn_ro_b[32]-[47]
	Bus_Get[26]-[27]	I16	2	BOOL	Bit0~Bit15 对应 fidbus.pfb_ro_b[48]-[63]/fidbus.pfn_ro_b[48]-[63]
	Bus_Get[28]-[29]	I16	2	I16	对应 fidbus.pfb_ro_i[0]/fidbus.pfn_ro_i[0]
	Bus_Get[30]-[31]	I16	2	I16	对应 fidbus.pfb_ro_i[1]/fidbus.pfn_ro_i[1]
	Bus_Get[32]-[33]	I16	2	I16	对应 fidbus.pfb_ro_i[2]/fidbus.pfn_ro_i[2]
	Bus_Get[34]-[35]	I16	2	I16	对应 fidbus.pfb_ro_i[3]/fidbus.pfn_ro_i[3]
	Bus_Get[36]-[37]	I16	2	I16	对应 fidbus.pfb_ro_i[4]/fidbus.pfn_ro_i[4]
	Bus_Get[38]-[39]	I16	2	I16	对应 fidbus.pfb_ro_i[5]/fidbus.pfn_ro_i[5]
系统变量	Bus_Get[40]-[87]	FLOAT	4 * 12	FLOAT	系统预留 FLOAT 类型变量，用户不可以使用
用户变量	Bus_Get[88]-[91]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[0]/fidbus.pfn_ro_r[0]
	Bus_Get[92]-[95]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[1]/fidbus.pfn_ro_r[1]
	Bus_Get[96]-[99]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[2]/fidbus.pfn_ro_r[2]
	Bus_Get[100]-[103]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[3]/fidbus.pfn_ro_r[3]
	Bus_Get[104]-[107]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[4]/fidbus.pfn_ro_r[4]
	Bus_Get[108]-[111]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[5]/fidbus.pfn_ro_r[5]
	Bus_Get[112]-[115]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[6]/fidbus.pfn_ro_r[6]
	Bus_Get[116]-[119]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[7]/fidbus.pfn_ro_r[7]
	Bus_Get[120]-[123]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[8]/fidbus.pfn_ro_r[8]
	Bus_Get[124]-[127]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[9]/fidbus.pfn_ro_r[9]
	Bus_Get[128]-[131]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[10]/fidbus.pfn_ro_r[10]
	Bus_Get[132]-[135]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[11]/fidbus.pfn_ro_r[11]
	Bus_Get[136]-[139]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[12]/fidbus.pfn_ro_r[12]
	Bus_Get[140]-[143]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[13]/fidbus.pfn_ro_r[13]
	Bus_Get[144]-[147]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[14]/fidbus.pfn_ro_r[14]
	Bus_Get[148]-[151]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[15]/fidbus.pfn_ro_r[15]

	Bus_Get[152]-[155]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[16]/fidbus.pfn_ro_r[16]
	Bus_Get[156]-[159]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[17]/fidbus.pfn_ro_r[17]
	Bus_Get[160]-[163]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[18]/fidbus.pfn_ro_r[18]
	Bus_Get[164]-[167]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[19]/fidbus.pfn_ro_r[19]
	Bus_Get[168]-[171]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[20]/fidbus.pfn_ro_r[20]
	Bus_Get[172]-[175]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[21]/fidbus.pfn_ro_r[21]
	Bus_Get[176]-[179]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[22]/fidbus.pfn_ro_r[22]
	Bus_Get[180]-[183]	FLOAT	4	FLOAT	对应 fidbus.pfb_ro_r[23]/fidbus.pfn_ro_r[23]
	Bus_Get[180]-[183]	/	/	/	未定义

15.5.8.3 机器人到 PLC

机器人到 PLC，机器人作为发送端，PLC 作为接收端。发送数据主要包括系统变量和 TPU 变量。其中 Bus_Set[0]-Bus_Set[191]为数据发送的 192 BYTE。

表 14-17　Profibus/Profinet 发送数据协议

变量分类	物理地址	单位	字节数	子单位	备注
系统变量	Bus_Set[0]-[1]	I16	2	BOOL	Bit0：手动状态
					Bit1：自动状态
					Bit2：远程状态
					Bit3：伺服状态
					Bit4：报警状态
					Bit5：急停状态
					Bit6：程序运行状态
					Bit7：安全位置 1 状态
					Bit8：安全位置 2 状态
					Bit9：安全位置 3 状态
					Bit10：安全位置 4 状态
					Bit11：加载程序状态
					Bit12：伺服确认状态
					Bit13：程序预约激活状态
					Bit14：程序复位状态（程序重新开始）
	Bus_Set[2]-[3]	I16	2	BOOL	Bit0：安全位置 5 状态
					Bit1：安全位置 6 状态

					Bit2: 安全位置 7 状态
					Bit3: 安全位置 8 状态
	Bus_Set[4]-[5]	I16	2	I16	运行速度（全局）
	Bus_Set[6]-[7]	I16	2	I16	报警代码 1
	Bus_Set[8]-[9]	I16	2	I16	报警代码 2
	Bus_Set[10]-[11]	I16	2	I16	程序号(用于反馈加载目标程序是否完成，例：在 Plc 端加载程序号为 2 的程序，如果加载完成，则程序号反馈为 2，否则为其他值）
	Bus_Set[12]-[13]	I16	2	I16	预约程序预约状态
	Bus_Set[14]-[15]	I16	2	I16	预约程序运行状态
	Bus_Set[16]-[19]	I16	4	I16	系统预留 I16 类型变量，用户不可用
用户变量	Bus_Set[20]-[21]	I16	2	BOOL	Bit0~Bit15 对应 fidbus.pfb_wo_b[0]-[15]/fidbus.pfn_wo_b[0]-[15]
	Bus_Set[22]-[23]	I16	2	BOOL	Bit0~Bit15 对应 fidbus.pfb_wo_b[16]-[31]/fidbus.pfn_wo_b[16]-[31]
	Bus_Set[24]-[25]	I16	2	BOOL	Bit0~Bit15 对应 fidbus.pfb_wo_b[32]-[47]/fidbus.pfn_wo_b[32]-[47]
	Bus_Set[26]-[27]	I16	2	BOOL	Bit0~Bit15 对应 fidbus.pfb_wo_b[48]-[63]/fidbus.pfn_wo_b[48]-[63]
	Bus_Set[28]-[29]	I16	2	I16	对应 fidbus.pfb_wo_i[0]/fidbus.pfn_wo_i[0]
	Bus_Set[30]-[31]	I16	2	I16	对应 fidbus.pfb_wo_i[1]/fidbus.pfn_wo_i[1]
	Bus_Set[32]-[33]	I16	2	I16	对应 fidbus.pfb_wo_i[2]/fidbus.pfn_wo_i[2]
	Bus_Set[34]-[35]	I16	2	I16	对应 fidbus.pfb_wo_i[3]/fidbus.pfn_wo_i[3]
	Bus_Set[36]-[37]	I16	2	I16	对应 fidbus.pfb_wo_i[4]/fidbus.pfn_wo_i[4]
	Bus_Set[38]-[39]	I16	2	I16	对应 fidbus.pfb_wo_i[5]/fidbus.pfn_wo_i[5]
系统变量	Bus_Set[40]-[63]	FLOAT	4 * 6	FLOAT	J1~J6 关节角度值/笛卡尔空间位姿
	Bus_Set[64]-[79]	FLOAT	4 * 4	FLOAT	J7~J10(附加轴 1-4)关节角度值
	Bus_Set[80]-[87]	FLOAT	4 * 2	FLOAT	系统预留 FLOAT 类型变量,用户不可用
用户变量	Bus_Set[88]-[91]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[0]/fidbus.pfn_wo_r[0]
	Bus_Set[92]-[95]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[1]/fidbus.pfn_wo_r[1]
	Bus_Set[96]-[99]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[2]/fidbus.pfn_wo_r[2]
	Bus_Set[100]-[103]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[3]/fidbus.pfn_wo_r[3]
	Bus_Set[104]-[107]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[4]/fidbus.pfn_wo_r[4]
	Bus_Set[108]-[111]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[5]/fidbus.pfn_wo_r[5]
	Bus_Set[112]-[115]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[6]/fidbus.pfn_wo_r[6]
	Bus_Set[116]-[119]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[7]/fidbus.pfn_wo_r[7]

	Bus_Set[120]-[123]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[8]/fidbus.pfn_wo_r[8]
	Bus_Set[124]-[127]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[9]/fidbus.pfn_wo_r[9]
	Bus_Set[128]-[131]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[10]/fidbus.pfn_wo_r[10]
	Bus_Set[132]-[135]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[11]/fidbus.pfn_wo_r[11]
	Bus_Set[136]-[139]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[12]/fidbus.pfn_wo_r[12]
	Bus_Set[140]-[143]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[13]/fidbus.pfn_wo_r[13]
	Bus_Set[144]-[147]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[14]/fidbus.pfn_wo_r[14]
	Bus_Set[148]-[151]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[15]/fidbus.pfn_wo_r[15]
	Bus_Set[152]-[155]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[16]/fidbus.pfn_wo_r[16]
	Bus_Set[156]-[159]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[17]/fidbus.pfn_wo_r[17]
	Bus_Set[160]-[163]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[18]/fidbus.pfn_wo_r[18]
	Bus_Set[164]-[167]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[19]/fidbus.pfn_wo_r[19]
	Bus_Set[168]-[171]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[20]/fidbus.pfn_wo_r[20]
	Bus_Get[172]-[175]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[21]/fidbus.pfn_wo_r[21]
	Bus_Get[176]-[179]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[22]/fidbus.pfn_wo_r[22]
	Bus_Get[180]-[183]	FLOAT	4	FLOAT	对应 fidbus.pfb_wo_r[23]/fidbus.pfn_wo_r[23]
	Bus_Get[184]-[191]	/	/	/	未定义

第 16 章　固定视觉

16.1 本章简介

本章主要介绍固定视觉的 APP 界面、固定视觉的标定及使用示例。

16.2 固定视觉功能介绍

16.2.1 功能简介

视觉功能是指机器人与视觉系统通过 TCP/IP 协议进行通讯，视觉系统作为服务器，机器人作为客户端，视觉系统将获取的基于视觉系统坐标下物体的位置信息转化成机器人坐标下的位置，从而实现机器人按指定轨迹运动。

固定视觉是指相机安装在固定台架上，拍摄的物体在固定的工作台面上。

16.2.2 TCP/IP 通讯协议及数据格式

在使用过程中，视觉系统（相机）需要将图像处理后的工件信息通过机器人提供的固定通讯格式传输给机器人，机器人根据接收到的数据进行取放动作。因此，机器人通讯格式主要包括三个部分：

物体坐标参数：X、Y、A

物体属性参数：ATTR

物体 ID 编码：ID

物体坐标参数是指物体在相机视野范围内的位置及旋转角度，该位置为相机/像素坐标系（单位 mm 或 px）下的位置。

物体属性参数是指根据物体不同属性（例如：形状、颜色等）给出物体的对应属性值，以数字：0、1、2、3……来表示。

物体 ID 编码是指为了方便管理给每一个物体制定的唯一编码。

属性参数与 ID 编码用户可根据实际情况选择是否使用以及具体的使用方式，如不需要应用，在相机通讯格式设置时将其默认为 0 即可。

具体通讯格式如下：

Image\r\n

[X:xxx.xxx;Y:xxx.xxx;A:xxx.xxx;ATTR:xxx;ID:xxx]\r\n

……

[X:xxx.xxx;Y:xxx.xxx;A:xxx.xxx;ATTR:xxx;ID:xxx]\r\n

Done\r\n

上述格式中，“Image”表示数据头，即一组图像数据下发开始的标志。“Done”表示数据尾，即一组图像数据下发完成（注意“Image”和“Done”区分大小写）。“\r\n”为回车换行符。

“[X:xxx.xxx;Y:xxx.xxx;A:xxx.xxx;ATTR:xxx;ID:xxx]”表示相机下发的一个物体的数据，其中包含了物体坐标参数 XYA，物体属性参数 ATTR 和物体 ID，每个数据用“数据名:数据;”的格式表示，每个物体的数据均以“[”开始，以“]”结束。**当相机没有拍到物体或者识别物体失败时发送字符串“Error”**。

注：固定视觉一次只能传输 1 组数据。

16.3 固定视觉 APP 界面介绍

16.3.1 固定视觉主界面

点击左上角进入桌面，点击桌面上的“**固定视觉**”按钮，进入固定视觉 APP 主界面，如下图 15-1 所示：

图 15-1 固定视觉 APP 主界面

图中红框内信息说明如下：

1) 固定视觉开关：用来是否开启控制器中的固定视觉功能；

2) TCP/IP 连接状态：表示当前机器人与相机通讯的连接状态，“**灰色**”表示当前处于断开状态，“**绿色**”表示当前处于连接状态；

3) 固定视觉坐标系：用于显示手眼标定成功后的标定结果；

4) 在机器人坐标系下工件的位置：在相机标定模式或手眼标定模式下用于显示拍照后工件在机器人坐标系下的位置；

5) 在相机坐标系下工件的位置：在相机上进行手眼标定后用于显示拍照后工件在相机坐标系下的位置；

6) 工件属性和 ID：用于显示工件的信息；

7) 像素分辨率、相机触发指令、拍照间隔和拍照按钮：像素分辨率，即为拍照得出照片的分辨

率，单位为毫米/像素；相机触发指令，当相机为指令触发模式时，输入相机内设置的拍照指令（该指令只能为 int 型）；拍照间隔用于设定测试时拍照的时间间隔（即：当触发相机拍照后，机器人获取相机数据的最长时间，该值设置不能低于 600 毫秒）；拍照按钮用于触发相机拍照；

8) 设置、标定和退出按钮：设置按钮用于从主页面切换到 TCP/IP 的设置界面；标定按钮用于从主页面切换到手眼标定页面；退出按钮用于退出固定视觉 APP。

16.3.2 视觉设置界面

点击固定视觉主界面的“**设置**”按钮，进入设置界面，设置界面如图 15-2 所示：

图 15-2 固定视觉设置界面

图中红框内信息说明如下：

相机连接设置：

1) 相机品牌：目前相机品牌可选通用相机、康耐视相机和麦克玛视相机；当选择康耐视相机时，需要输入相机账号和密码；

2) 相机登录账号：当选择康耐视相机时，需要输入相机上设置好的登录账号；

3) 登录密码：当选择康耐视相机时，需要输入相机上设置的登录密码；

4) 相机 IP 地址：需要输入相机上设置的 IP 地址；

5) 相机端口：需要输入相机的端口号；

6) 数据格式：设置数据格式，目前只有一种数据格式；

7) 相机触发方式：有指令触发和 IO 触发两种方式；当选择指令触发时，相机触发 IO 呈灰色，不可设置；

8) 拍照时间间隔：用来设置拍照的时间间隔（与主界面相同）；

9) 相机触发 IO：当相机触发方式选择为 IO 触发时，可设置触发 IO 的地址；

10) 标定方式：有相机标定和手眼标定两种方式；相机标定是在相机上完成的；

11) 相机触发指令：在相机设置好命令触发方式后可在此处进行设置（与主界面相同）；

12) 相机数据获取指令：此次设置只针对康耐视相机，当登录康耐视相机后，此设置才生效；

13) 保存、连接、返回按钮：当所有的设置完成后，须点击“**保存**”按钮，会将设置信息保存到控制器中；点击“**连接**”后，机器人会跟相机进行连接，连接成功后，界面右上角的状态会变成绿色；返回到主界面时须点击“**返回**”按钮。

16.3.3 手眼标定界面

在设置界面将标定方式选为“**手眼标定**”，点击“**保存**”按钮，然后返回到主界面，点击“**标定**”按钮，进入标定界面，如图 15-3 所示：

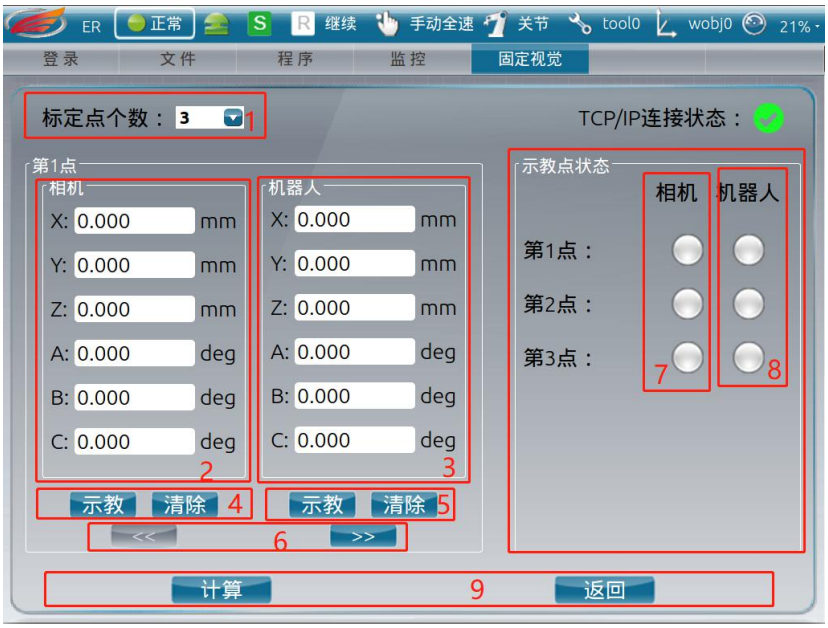


图 15-3 固定视觉标定界面

图中红框内信息说明如下：

1) 标定点个数：目前可选标定点的个数为：3~6；

2) 相机：当点击“**示教**”按钮时，显示工件在相机坐标系下的值；

3) 机器人：当点击“**示教**”按钮时，显示机器人当前的坐标值；

4) 示教、清除：工件在相机下的操作，当点击“**示教**”按钮时，记录工件在相机坐标系下的值；当点击“**清除**”按钮时，将当前显示的值清零；

5) 示教、清除：工件在机器人下的操作，其作用同上；

6) 当前点的切换：当按下“<<”，可以切换到上一点；当按下“>>”，可以切换到下一点；

7) 相机：相机坐标系下，示教点的示教状态；“**灰色**”表示点未示教；“**黄色**”表示点已示教；

8) 机器人：机器人坐标系下，示教点的示教状态；“**灰色**”表示点未示教；“**黄色**”表示点已示教；

9) 计算、返回：当所有点示教完成后点击“**计算**”按钮，其计算结果会在主界面的固定视觉坐

标系中显示；点击“**返回**”，返回主界面。

16.3.4 像素分辨率标定界面

点击“**标定**”按钮，会弹出是否使用像素分辨率标定提示框，如下图 15-4 所示，点击“**是**”，进入像素分辨率标定界面，如图 15-5 所示：

图 15-4 像素分辨率提示框

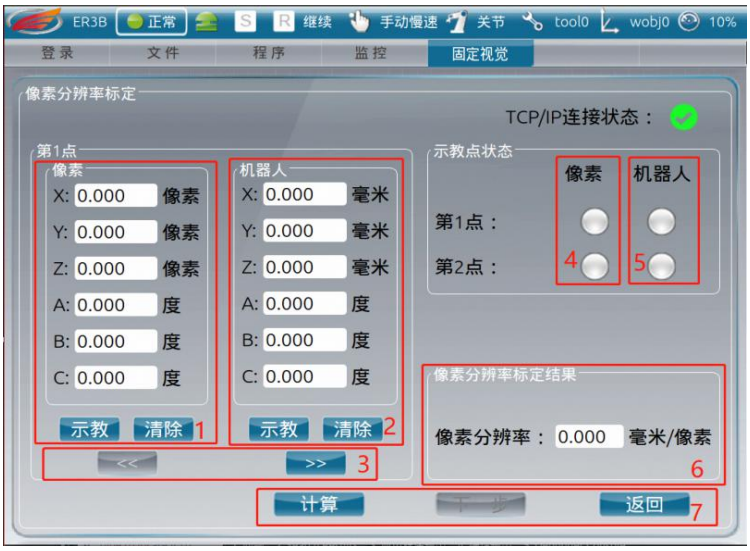


图 15-5 像素分辨率标定界面

图中红框内信息说明如下：

1) 像素：工件在相机坐标系下像素值；

2) 机器人：工件在机器人坐标系的坐标值；

3) 上下点切换：切换到上一点和下一点；

4) 像素示教状态：像素标定是否成功状态；

5) 机器人示教状态：机器人坐标值标定是否成功状态；

6) 像素分辨率标定结果：标定后计算得到的像素分辨率标定结果；

7) 计算、下一步、返回按钮：计算标定的结果；进入到传送带标定界面；返回的跟踪视觉主界面。

16.4 固定视觉的标定及用例

固定视觉的标定方式：相机标定和手眼标定。

16.4.1 相机标定

16.4.1.1 设置界面参数设定及相机标定

进入设置界面，选择使用的相机品牌，完成相关设置，将标定方式选为相机标定，点击“**保存**”，然后点击“**连接**”按钮与相机进行连接，连接上后返回到主界面，如图 15-6 所示。相机的标定在相机软件上完成，具体标定操作流程请根据相机提供的标定流程进行。在相机上标定完成后，进行拍照测试。

16.4.1.2 相机标定测试

在相机标定的主界面中，点击“**拍照**”按钮，工件在机器人坐标系下的值会刷新。

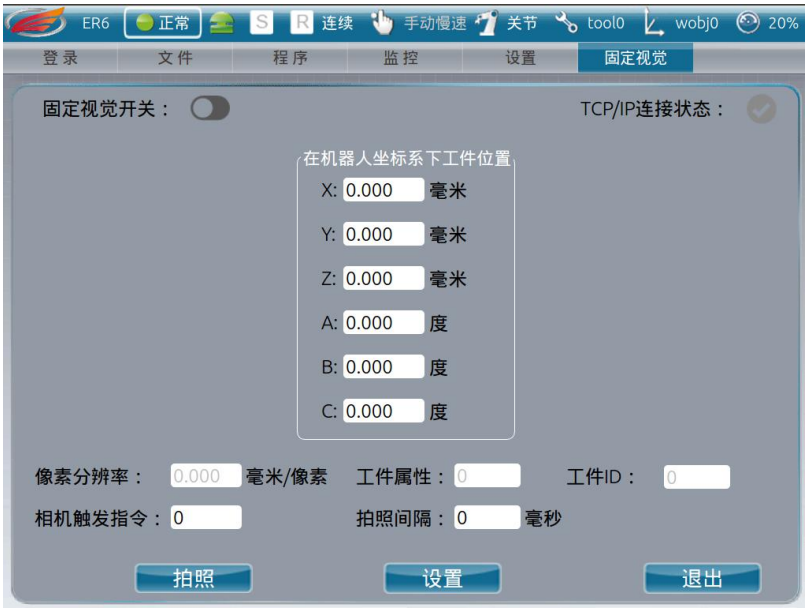


图 15-6 相机标定的主界面

固定视觉 RPL 指令说明：

表 15-1 固定视觉 RPL 指令

指令	名称	功能
Vision.connectCam(int p)	相机通讯连接指令	调用该命令，相机可自动连接服务器通讯。
Vision.closeCam()	相机通讯断开指令	调用该命令，相机可自动断开服务器通讯。
Vision._Init_()	视觉功能初始化命令	该命令视觉相关功能的初始化，在正常使用中不需要调用，该函数会在程序开始自动运行
Vision.getData()	相机拍照命令	调用该命令，触发相机拍照动作并返回相应数据
Vision.setTrigCmd(int p)	设置相机触发指令	相机在指令触发的模式下，该命令可设置相机触发的指令
Vision.trigCam()	触发相机拍照命令	相机在指令触发的模式下，该命令可触发拍照

变量	名称
Vision.x real	工件位置：X 方向坐标
Vision.y real	工件位置：Y 方向坐标
Vision.z real	工件位置：Z 方向坐标
Vision.a real	工件姿态：绕 Z 轴角度
Vision.b real	工件姿态：绕 Y 轴角度
Vision.c real	工件姿态：绕 X 轴角度
Vision.attr int	工件属性
Vision.id int	工件 ID

固定视觉 RPL 程序用例（相机标定模式）：

```
1   LABEL a :
2   ret1 := vision.connectCam(5) ;
3   IF ret1 = 1 THEN
4     MJOINT (*, v500, fine, tool0) ;
5     vision.setTrigCmd("1") ;
6     hasObj := vision.getData() ;
7     IF hasObj THEN
8       hight := 400 ;
9       point1pick := POINTC(vision.x, vision.y, hight, vision.a, 180, 0) ;
10      point1 := POINTC(vision.x, vision.y, hight + 50, vision.a, 180, 0) ;
11      MJOINT (point1, v500, fine, tool0) ;
12      MJOINT (point1pick, v500, fine, tool0) ;
13      DWELL (5) ;
14      MJOINT (point1, v500, fine, tool0) ;
15    END_IF ;
16  END_IF ;
17  GOTO a ;
```

Line1:循环开始；

Line2:连接相机通讯；（在固定视觉设置界面手动连接效果与此命令相同，该步骤是在程序运行后自动连接相机，减少手动操作步骤）

Line3:ret1 等于 1 表示相机连接成功，则执行 Line4-15；

Line4:机器人运动到工位 1；

Line5:设置相机触发指令为 1；（当相机设置为 IO 触发时，不需要该步骤）

Line6:触发相机拍照，当成功获取数据时变量 hasObj=true；

Line7:如果成功获取相机数据，则执行 Line8-14；

Line8:定义工件高度位置补偿值 hight；

Line9:定义工位 2（抓取点）；

Line10:定义工位 3（抓取点上方位置）；

Line11:机器人运动到工位 3；

Line12:机器人向下运动到工位 2；

Line13:等待抓取工件；

Line14:机器人向上运动到工位 3；

Line17:循环结束；

注意：因为在相机端做手眼标定，所以 vision.x,vision.y,vision.a 三个工件数据结果可以直接应用，vision.z,vision.b,vision.c，需要根据现场实际情况进行数值上的补偿。

16.4.2 手眼标定

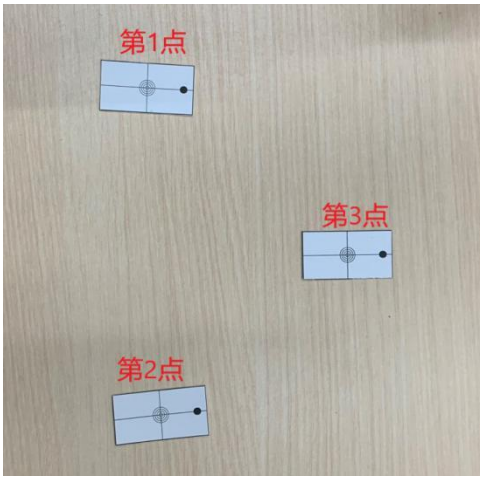
16.4.2.1 设置界面参数设定及手眼标定

相机安装要求：相机镜头平面需要尽量与平台的平面平行，相机坐标系的 Z 轴方向需要竖直向上；目前这个版本固定视觉只支持抓取点与相机识别点相同的情况。

表 15-2 手眼标定操作步骤

步骤	图示	说明
1.点击“设置”，进入参数设置界面。		完成参数设置，点击“**保存**”按钮，保存好数据后，点击“连接”，连上相机后返回到主界面。
2. 点击“**标定**”按钮，弹出弹框提示。		如果使用像素分辨率标定则点击“**是**”，进入像素分辨率标定界面，若不使用，则选择“**否**”，进入传送带标定界面。

		
3.若进入了像素分辨率标定界面，则进行像素分辨率标定操作。	 	需标定传送带上相机拍照范围内的 2 个点，标定的这 2 个点的位置尽量在相机拍照范围的对角边上。 首先进行第 1 个点标定，确保机器人移动到相机视觉范围外，将工件放在相机拍照范围内，点击“**示教**”按钮，在相机像素坐标值显示工件的位置值，并且在示教状态中第 1 点像素由“**灰色**”变为“**黄色**”，表示第一点像素示教成功，再进行第 1 点机器人位置示教，将机器人末端工具移动到工件表面上方，点击“**示教**”按钮，传送带标定在机器人坐标值会更新，并且在机器人示教状态中第 1 点由“灰**色**”变为 “**黄色**”，表示第一点机器人坐标值示教成功。 进行第 2 点标定，点击**“>>”**按钮，切换到第 2 点，进行第 2 点像素坐标值和机器人坐标值示教，步骤同第 1 点；两点示教完成。

4.点击“计算”按钮。		计算成功后会弹出成功提示框，然后点击“**是**”，将更新像素分辨率标定的结果，“**下一步**”按钮由灰色变为蓝色，点击“**下一步**”,进入手眼标定界面。
5.在手眼标定标定界面进行标定操作。		以 3 点标定为例，进行机器人手眼标定。在标定前，选择所需的工具坐标系，默认的工具坐标系为 tool0。
	 	在标定时，将工件放在如左侧图 1 所示第一点的位置，点击相机下的“**示教**”按钮，此时会相机下的坐标值显示框中显示工件在相机坐标系下的坐标值，然后将机器人移动到工件的位置，点击机器人下的“**示教**”按钮，在机器人下的坐标值显示框中显示当前工件在机器人坐标系下的位置值，如果示教成功，则在右侧示教点的状态中相应的状态灯会变成橙色。根据如左侧图 1 所示的 3 点位置，依次示教完后，如左侧图 2 所示，点击“**计算**”按钮。标定完后点击“**返回**”按钮，进入主界面。

标定结果显示在固定视觉坐标系中，如左侧图所示，点击“**拍照**”按钮，会刷新工件在相机坐标系和机器人坐标系下的位置值。可以根据现场需求设置相机拍照时间间隔（**注意：拍照的时间间隔不能小于 600 毫秒**）。

16.4.2.2 测试标定结果和测试程序

编写一个测试程序，如下：

```
1   LABEL a :
2   MJOINT (*, v500, fine, tool1) ;
3   vision.setTrigCmd(1) ;
4   hasObj := vision.getData() ;
5   IF hasObj THEN
6     point1pick := POINTC(vision.x, vision.y, vision.z + 5, -180 - vision.a, 180, 0) ;
7     point1 := POINTC(vision.x, vision.y, vision.z + 50, -180 - vision.a, 180, 0) ;
8     MJOINT (point1, v500, fine, tool1) ;
9     MJOINT (point1pick, v500, fine, tool1) ;
10    DWELL (5) ;
11    MJOINT (point1, v500, fine, tool1) ;
12  END_IF ;
13  GOTO a ;
```

图 15.7 测试程序界面

Line1:循环开始；

Line2:机器人运动到工位 1；

Line3:设置相机触发指令为 1；（当相机设置为 IO 触发时，不需要该步骤）

Line4:触发相机拍照，当成功获取数据时变量 hasObj=true；

Line5:如果成功获取相机数据，则执行 Line6-11；

Line6:定义工位 2（抓取点）；

Line7:定义工位 3（抓取点上方位置）；

Line8:机器人运动到工位 3；

Line9:机器人向下运动到工位 2；

Line10:等待抓取工件；

Line11:机器人向上运动到工位 3；

Line13:循环结束；

注意：因为在机器人端做手眼标定，所以 vision.x,vision.y,vision.z 三个工件位置数据结果可以直接应用，vision.a,vision.b,vision.c，需要根据现场实际情况进行数值上的补偿。

根据测试时机器人的抓取点来判断标定的准确性。

第 17 章　Mot 程序管理

17.1 本章简介

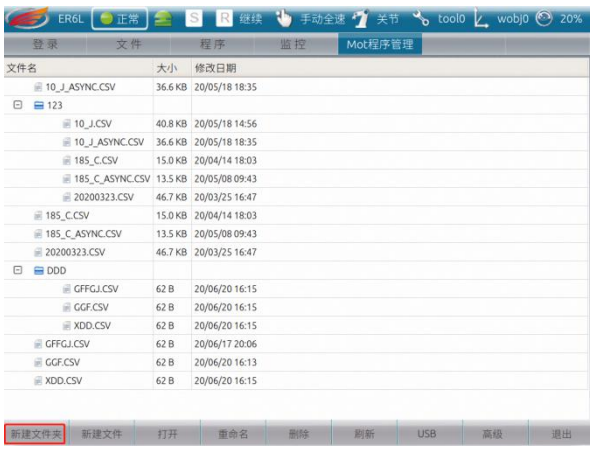
本章主要介绍 Mot 程序的编辑管理功能和 Mot 程序在程序中调用方法。此功能可代替 RPL 编程功能，记录大量的轨迹点位数据用于优化运动轨迹，一般需要配合无关节动力臂等外部设备采集数据后使用。

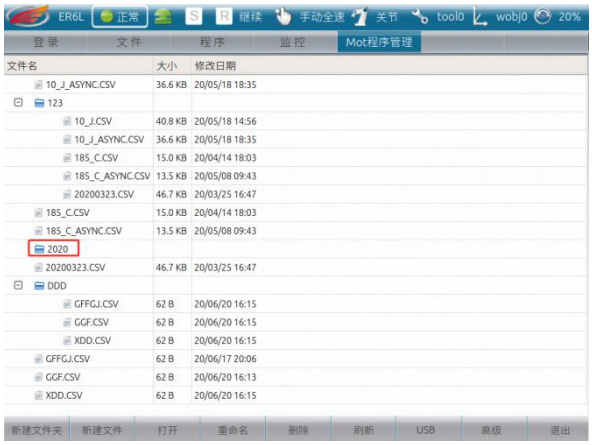
17.2 Mot 程序管理

Mot 程序管理目前可以对轨迹运动点数据程序文件进行管理和编辑操作，功能包括新建、重命名、删除、USB 的导入和导出、复制、剪切和粘贴，以及对 Mot 程序进行编辑和手动运行操作。

17.2.1 新建文件夹

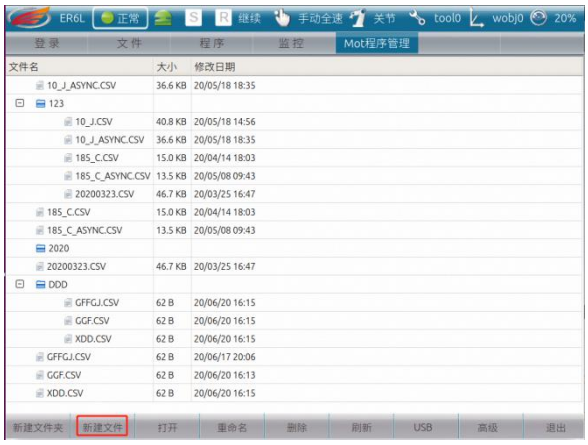
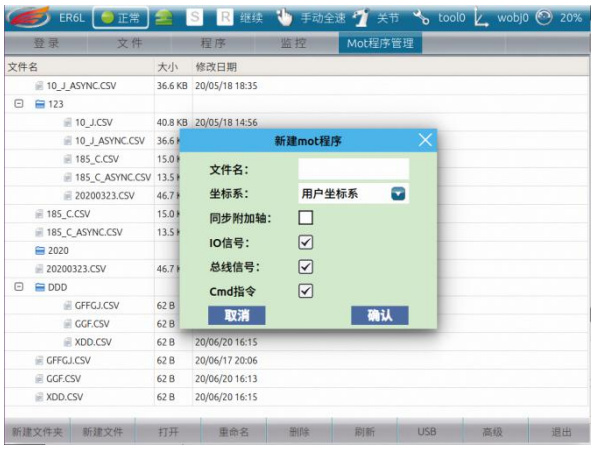
表 16-1 新建文件夹操作步骤

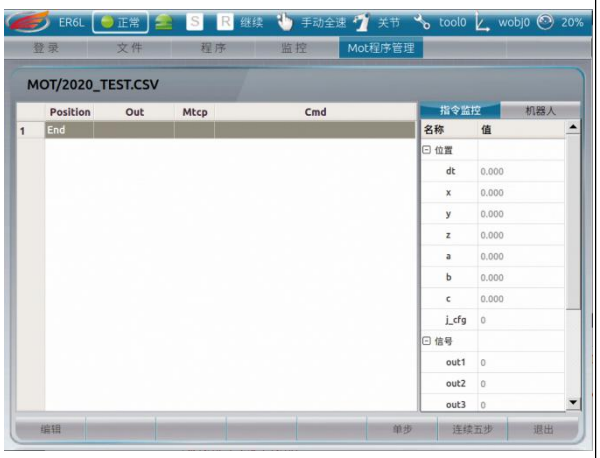
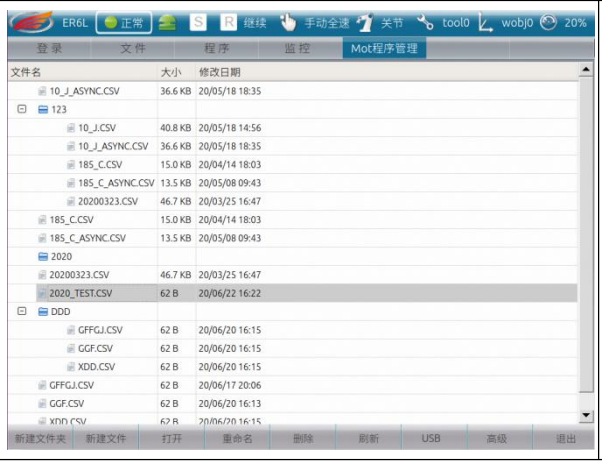
步骤	图示	说明
1.打开 Mot 程序管理。		打开示教器桌面，点击 Mot 程序管理图标进入 APP 界面。
2.新建文件夹。		点击“**新建文件夹**”按钮。
3.为新建文件夹命名。		在弹出命名窗口后输出新建文件夹的名称。 点击绿色“√”按钮，创建新文件夹。

17.2.2 新建文件

表 16-2 新建文件操作步骤

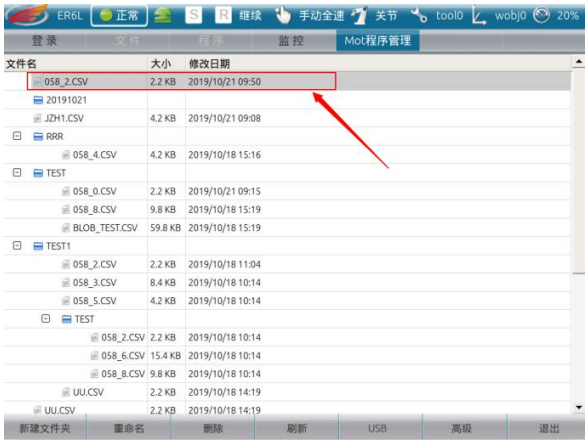
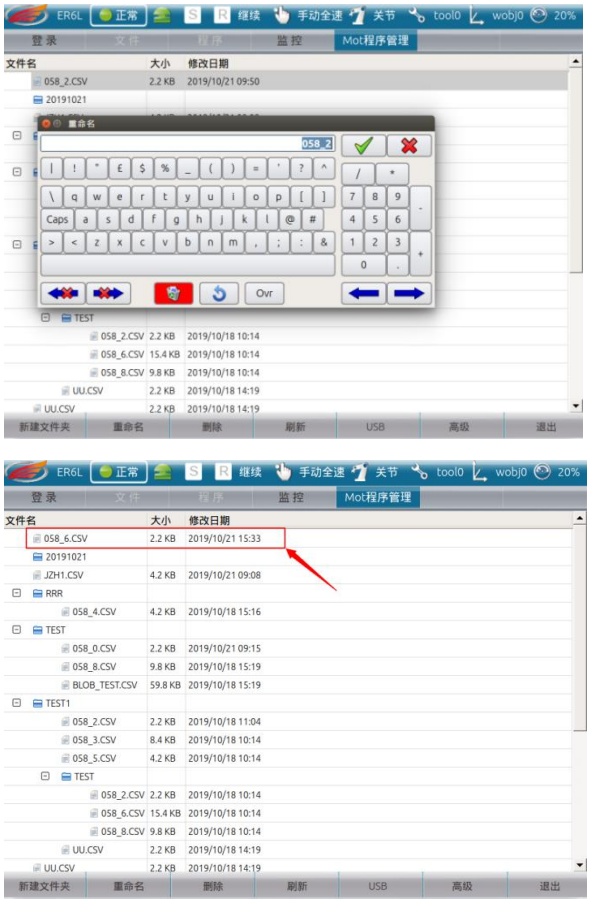
步骤	图示	说明
1.选择新建文件位置，点击**“新建文件”**按钮。		选中文件夹名称或者该文件夹目录下的文件，则新建的文件在该目录下。 未选中或者选中根目录下文件，新建文件在根目录中。
2.设置新建文件内容。		文件名：新建文件的名称。 坐标系：可选择用户坐标系和关节坐标系。表示 Mot 程序中只能记录该坐标系下的数据。 同步附加轴：选择是否需要记录同步附加轴位置。勾选则记录两个同步附加轴位置（实际小于 2 个附加轴时，无实际附加轴的轴数据一直为 0）。 IO 信号：选择程序中是否需要改变 IO 状态的信号。

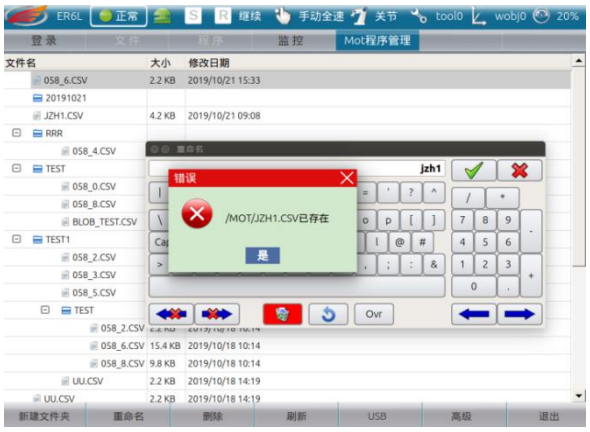
		总线信号：选择程序中是否需要设置 modbus 可读可写变量中 bool 信号的状态。 Cmd 指令：选择程序中是否需要设置 Cmd 指令。Cmd 中主要可以设置 IO 和 modbus_RW 的 bool 信号输出脉冲，以及控制附加轴进行异步运动。
3.新建文件后进入到编辑界面。		
4.退出可查看新建的文件。		

17.2.3 重命名

表 16-3 重命名操作步骤

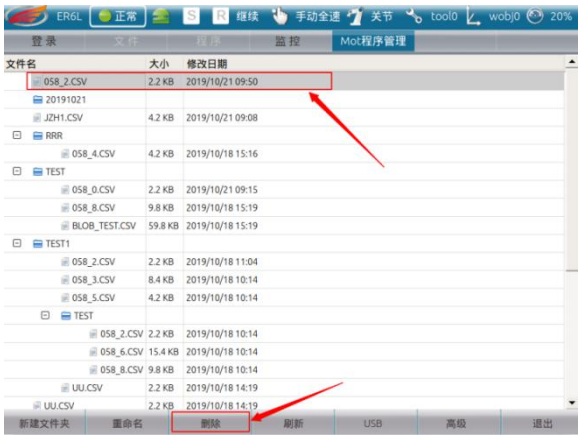
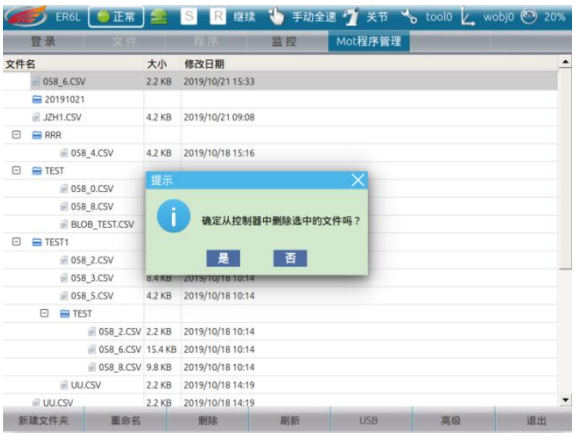
步骤	图示	说明

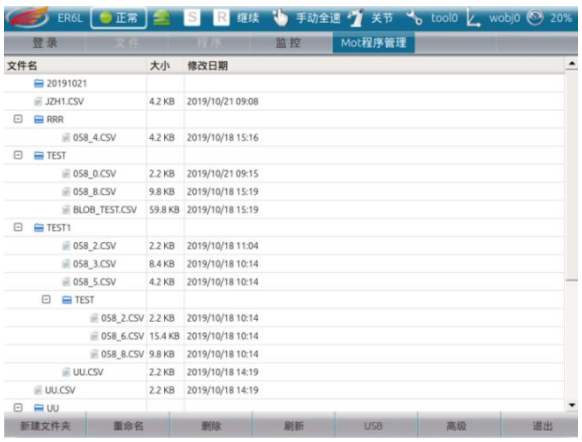
1.选择需要重命名的文件或文件夹。此次以文件为例，文件夹的操作步骤相同。		选择需要进行重命名的文件在文件名上点击，点击后文件背景会变为灰色。 再点击**“重命名”**按钮。
2.编辑输入新文件名。		在弹出的弹窗中编辑新的文件名（此文件名在相同目录下未被使用）。 然后点击绿色**“√”**按钮，修改文件名称。 查看文件名已被修改。

3.若输入的新文件名在相同目录下已被使用。		若在弹窗键盘中输入的新文件名已存在于相同目录下，则提示该文件已存在，文件名不作修改。

17.2.4 删除

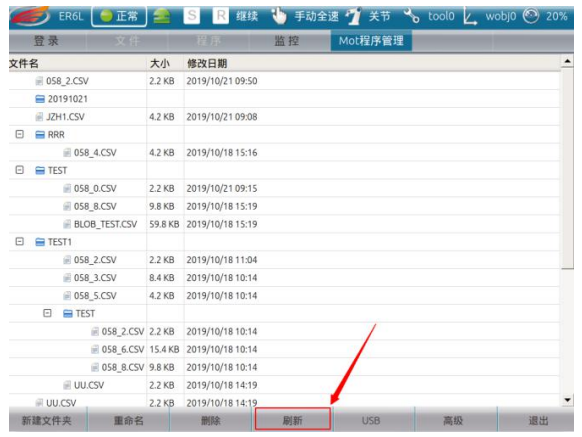
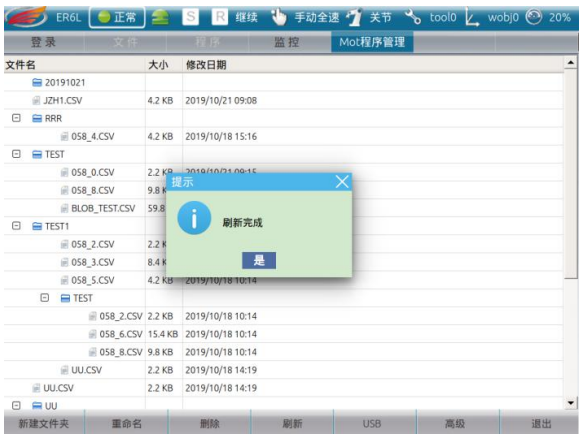
表 16-4 删除操作步骤

步骤	图示	说明
1.选择需要删除的文件或文件夹。此次以文件为例，文件夹的操作步骤相同。		选择需要删除的文件，在文件名上点击，点击后文件背景会变为灰色。 再点击**“删除”**按钮。
2.删除前请认真确认，以免删错。		如果确定需要删除，则点击**“是”**按钮；否则点击**“否”**按钮，或者将弹窗关闭。

17.2.5 刷新

表 16-5 刷新操作步骤

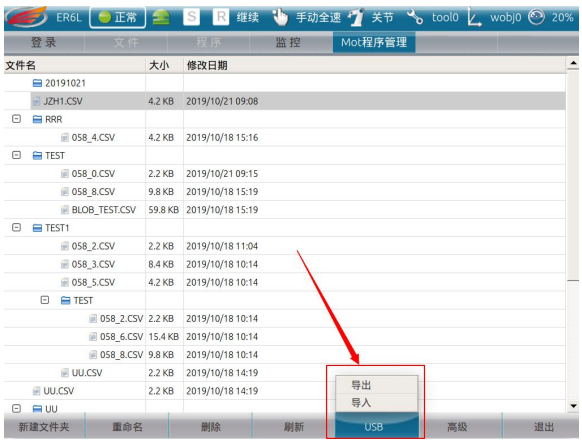
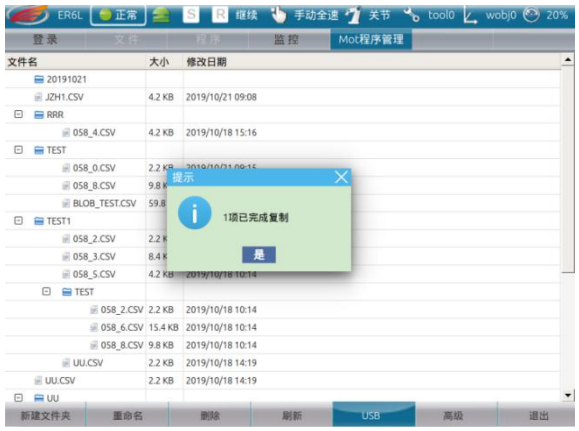
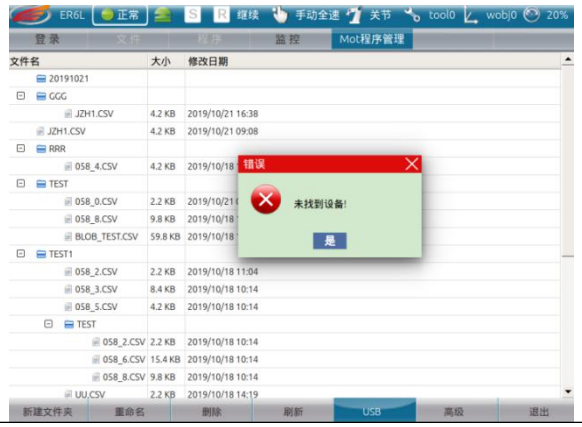
步骤	图示	说明
1.使用网线等外部文件传输后，刷新示教器 Mot 程序界面显示。	 	点击“**刷新**”按钮。 弹窗提示“**刷新完成**”。

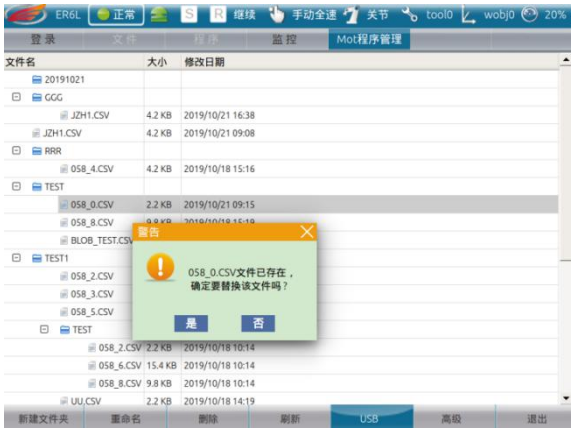
17.2.6 USB

17.2.6.1 导出

表 16-6 USB 导出操作步骤

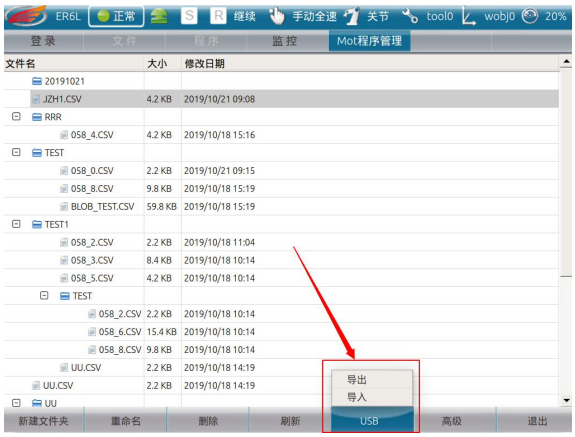
步骤	图示	说明

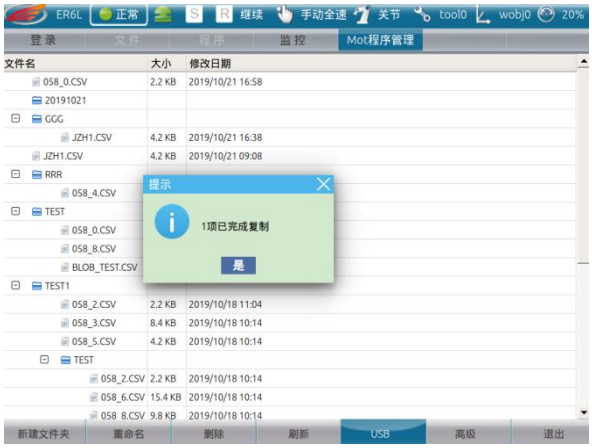
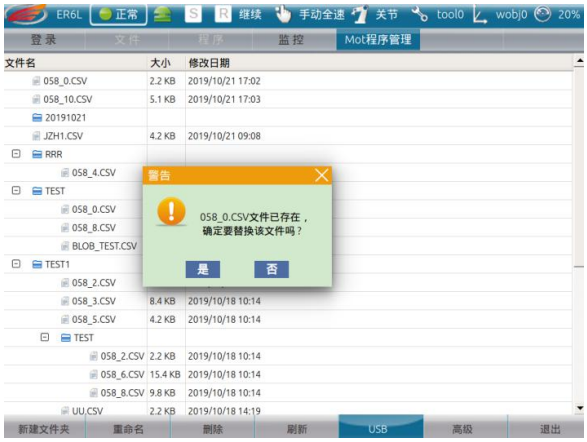
1.USB 导出功能是用于一次性将控制器内若干 Mot 程序文件或文件夹导出到 U 盘中。此处以文件为例，文件夹的操作与文件相同。	   	插入一个可用 U 盘。 选中需要导出的一个或多个文件（选中的文件背景会变灰色）。 点击“**USB**”按钮，再选择“**导出**”功能。 导出成功后则有如下图弹窗提示。 **注：若未正确插入 U 盘或者 U 盘无法被识别则会弹窗提示“未找到设备！”。**

2.若被导出的文件或文件夹已存在于U盘中，则可以选择是否替换。		点击**“是”**则替换U盘中该文件。 点击**“否”**则取消本次导出功能，弹窗提示**“0项已完成复制”**。 **注：导出的文件或文件夹都会被默认复制到U盘的一级目录中。**

17.2.6.2 导入

表 16-7 USB 导入操作步骤

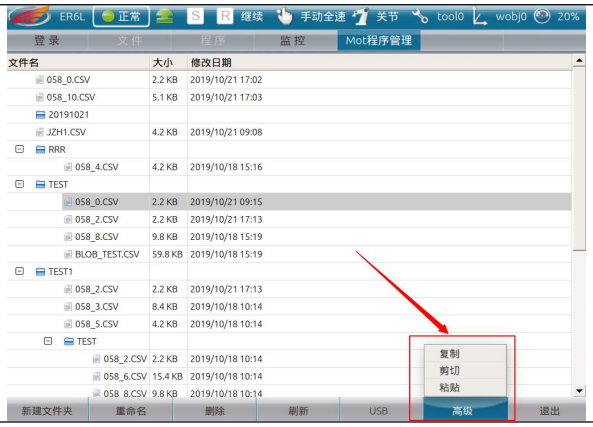
步骤	图示	说明
1.USB 导入功能是用于一次性将 U 盘内选中的 Mot 程序文件或文件夹导入到控制器中。此处以文件为例，文件夹的操作与文件相同。		插入一个可用 U 盘，点击**“USB”**按钮，再选择**“导入”**功能。
2.选中文件，并导入。		选中需要导入的文件（选中的文件背景会变蓝色）。 点击蓝色**“导入”**按钮。 导入成功后则有如下图弹窗提示。

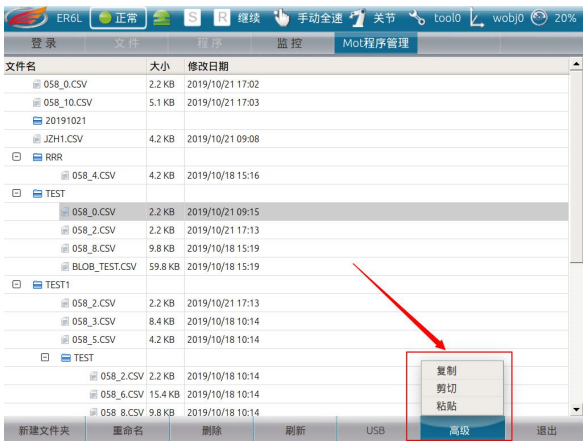
		
2.若被导入的文件或文件夹已存在于控制器中，则可以选择是否替换。		点击**“是”**则替换控制器中该程序文件。 点击**“否”**则取消本次导入功能。 **注：1.若导入的是文件夹，且在控制器中已存在同名文件夹，则 U 盘和控制器中的同名文件夹合并，并将控制器文件夹中的同名文件进行替换。** **2.文件夹的导入只导入两层文件夹中内容，若有第三层文件夹则不作处理。**

17.2.7 高级

17.2.7.1 复制&剪切

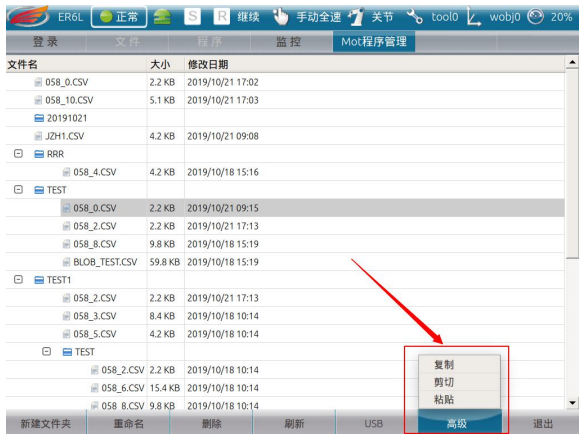
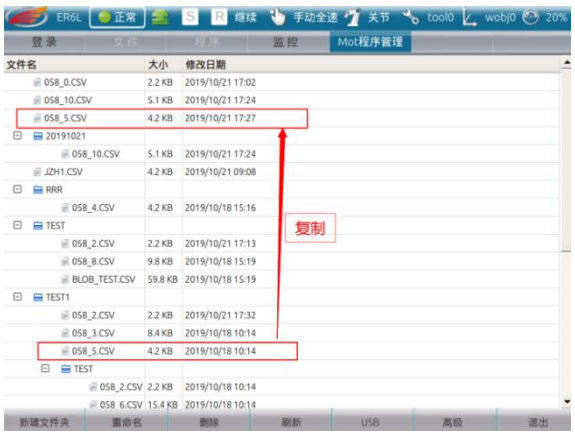
表 16-8 复制&剪切操作步骤

步骤	图示	说明
1.复制功能是对控制器内文件或文件夹进行一次拷贝，原文件或文件夹不变。		选择需要复制的文件，在文件名上点击，点击后文件背景会变为灰色。 点击**“高级”**按钮，再点击弹出的**“复制”**按钮。

2.剪切功能是对控制器内文件或文件夹进行一次拷贝，然后将原文件或文件夹删除。		选择需要复制的文件，在文件名上点击，点击后文件背景会变为灰色。 点击**“高级”**按钮，再点击弹出的**“剪切”**按钮。

17.2.7.2 粘贴

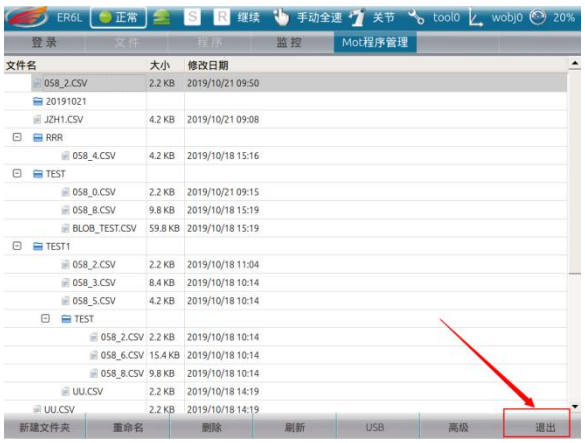
表 16-9 粘贴操作步骤

步骤	图示	说明
1.粘贴功能是在进行复制或剪切功能后，将选中的文件或文件夹放置到指定的路径中去。	 	点击一级目录下的文件或不选择目录（默认一级目录），点击**“高级”**按钮，再点击弹出的**“粘贴”**按钮。 复制或剪切的文件或文件夹则会被粘贴至指定路径下。 **注：粘贴只可以粘贴至一级目录（默认目录）或二级文件夹目录下，若指定至三级文件夹目录则不作处理。**

17.2.8 退出

表 16-10 退出操作步骤

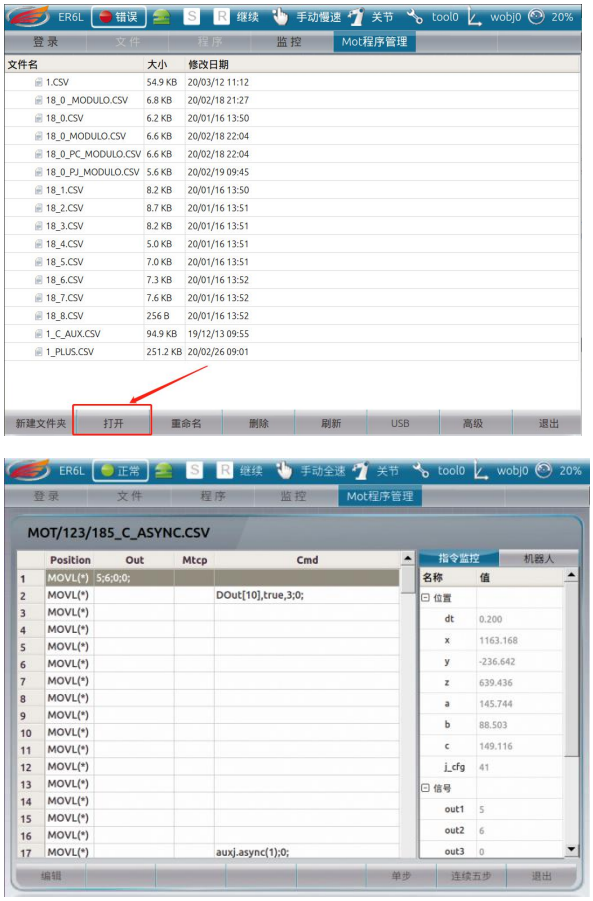
步骤	图示	说明

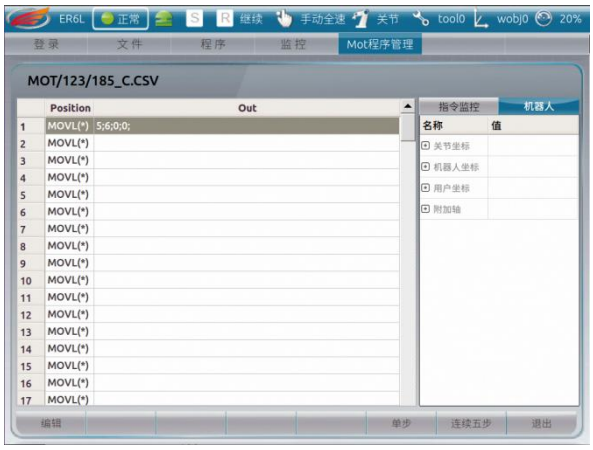
1.退出功能是完成 Mot 程序管理 APP 到示教器桌面切换功能的按钮。		点击**“退出”**按钮，则可以从 Mot 程序管理界面切换到示教器桌面。

17.3 Mot 程序编辑和运行

17.3.1 程序编辑

表 16-11 程序编辑操作步骤

步骤	图示	说明
1. 打开或新建一个 Mot 程序。	 	选择想要编辑的 Mot 程序，点击**“打开”**按钮，则可以查看程序内容。 新建 Mot 程序则根据实际应用选择指令信息。



界面右侧可以看到当前指令的详细信息以及监控机器实时位置。

1）指令监控

分为位置信息和信号信息，分别如下：

位置：

dt：等时插补的点位之间的时间间隔。

xyzabc：机器人用户坐标系下的位置。

j_cfg：关节姿态，示教后会根据位置自动计算，请不要随意修改。

j1-j6：机器人关节坐标系下的位置。

flags：关节标志位，关节坐标系固定为 1。

aux1-aux2：同步附加轴 1-附加轴 2 的关节位置。

注：当记录数据为用户坐标系时，位置信息由 dt、xyzabc、j_cfg 和 aux1-aux2 组成，当记录数据为关节坐标系时，位置信息由 dt、j1-j6、flags 和 aux1-aux2 组成。这两种情况下，若无同步附加轴，可以选择不记录 aux1-aux2。

信号：

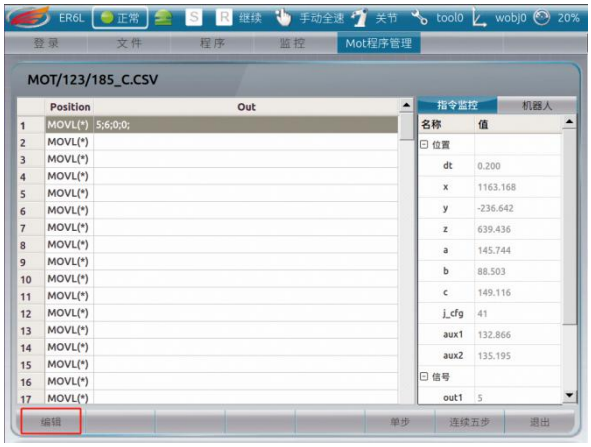
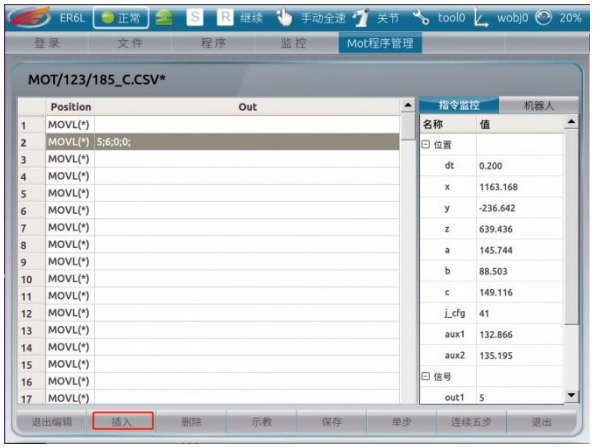
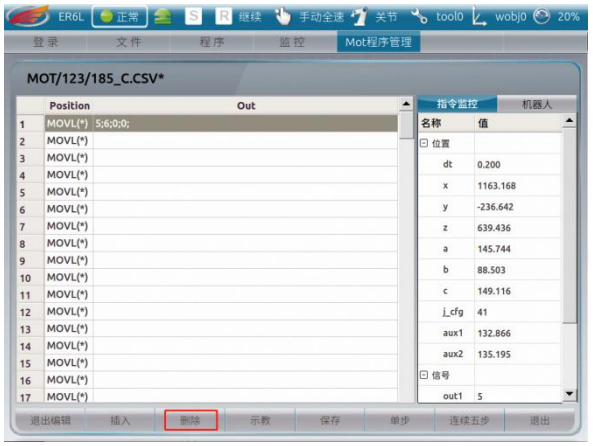
out1-out4：IO 输出端口输出，最多可设置 4 个 io 口，正数表示对应口输出高电平，负数表示对应口输出低电平。

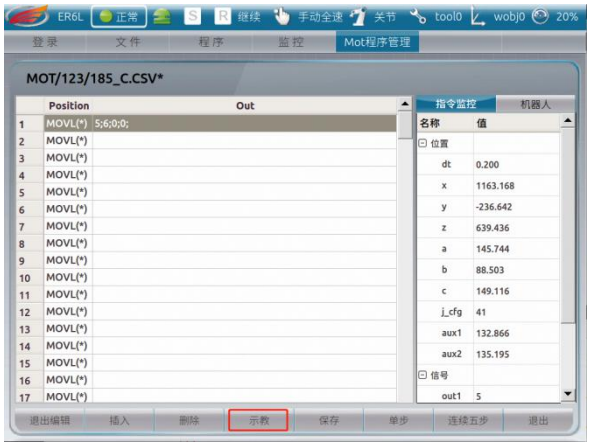
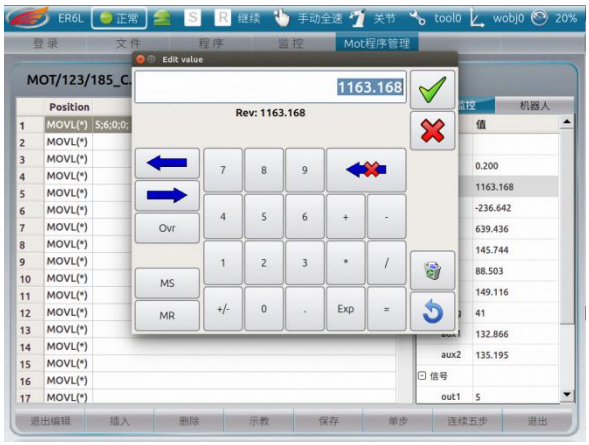
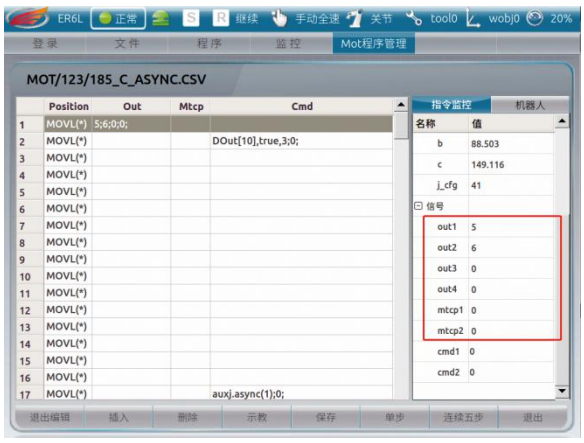
mtcp1-mtcp2：总线 modbus_rw_b 的端口输出，最多可设置 2 个总线口，正数表示对应口输出高电平，负数表示对应口输出低电平。

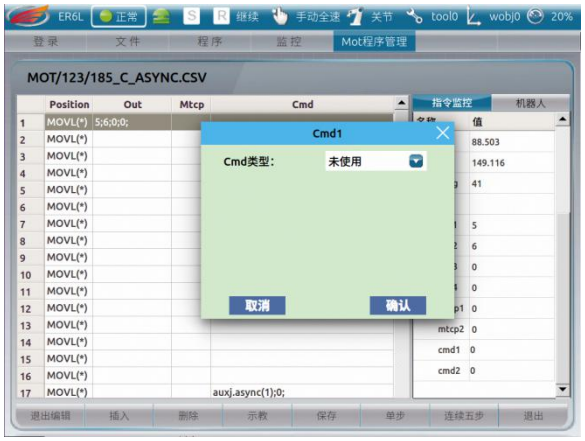
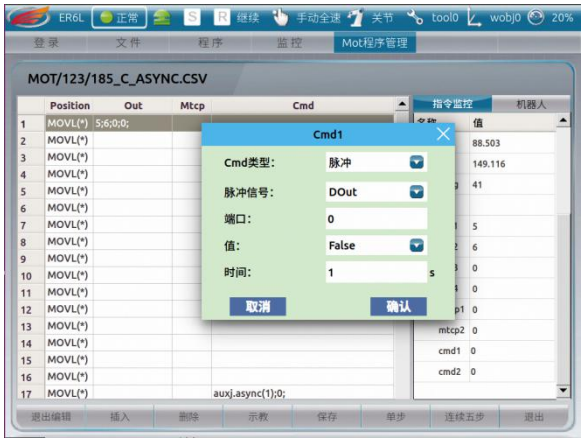
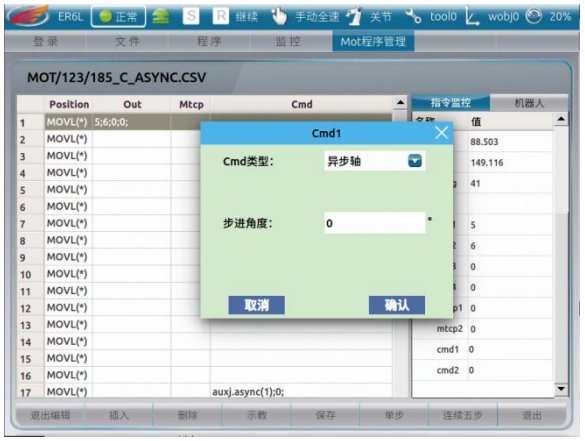
cmd1-cmd2：设置 io 或者总线 modbus_rw_b 的端口输出脉冲，或者控制异步轴运动，最多可设置 2 个端口。

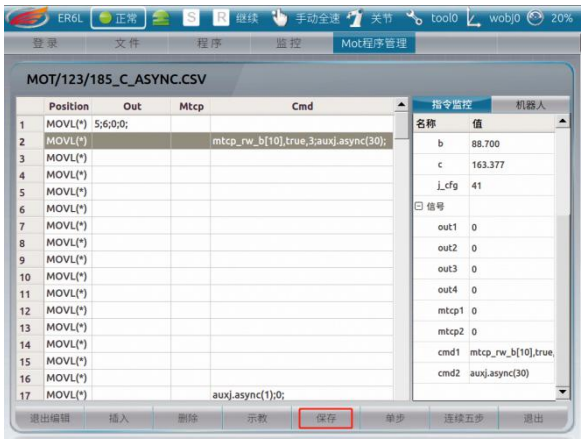
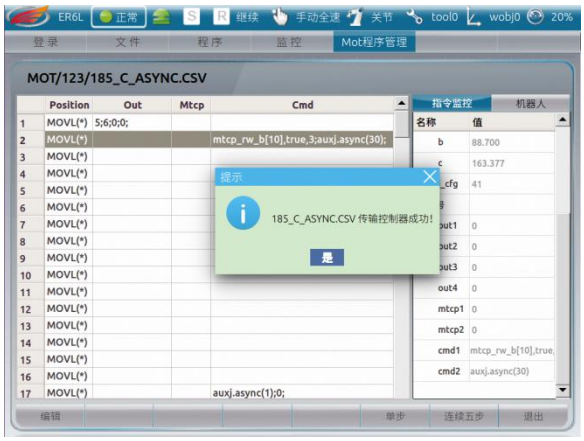
2）机器人信息

机器人信息用来查看机器人当前位置信息，可以查看关节坐标系、机器人坐标系和用户坐标系下的机器人位置。同时可以查看附加轴当前位置。

2. 进入编辑状态。		点击左下角“**编辑**”按钮，可以进入编辑状态。
3.插入程序。		点击“**插入**”按钮，在当前选中行上方插入一个点位。点位的位置为当前机器人所在位置。
4.删除程序。		点击“**删除**”按钮，会将当前选中行删除。

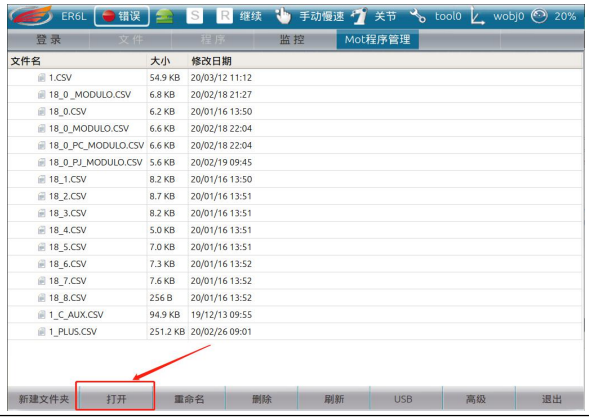
5.示教程序。		点击“**示教**”按钮，会将当前选中行位置修改为当前机器人位置信息。
6.手动编辑。		先选中需要修改的行； 在右边窗口，点击需要修改的数据，会弹出如图的键盘； 在键盘中输入修改后的值，点击“√”即可。
7.设置 IO/总线输出。		输入 io 地址则将对应地址的 io 口置为 True，输入负的 io 地址，则将 io 口置为 False。 如图第三行 out1 和 out2 输入 13、14，则运行这一行，13、14 号 io 会置为 True；在第七行 out1、out2 输入-13、-14，则运行这一行，13、14 号 io 会置为 False。此处打开和关闭 io 口可以不在同一个 out 通道，即-13、-14 在 out3 或者 out4 中，效果相同。 总线输出和 io 相同。

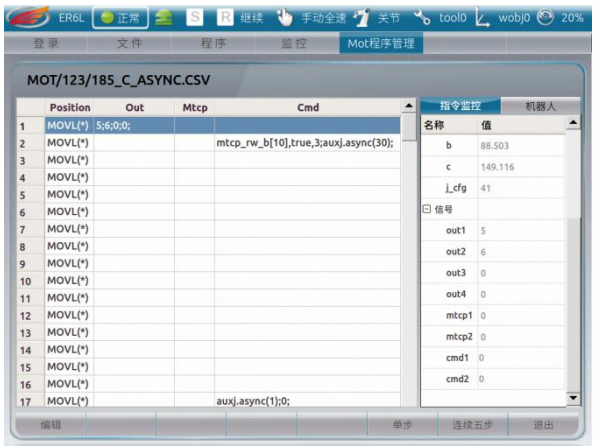
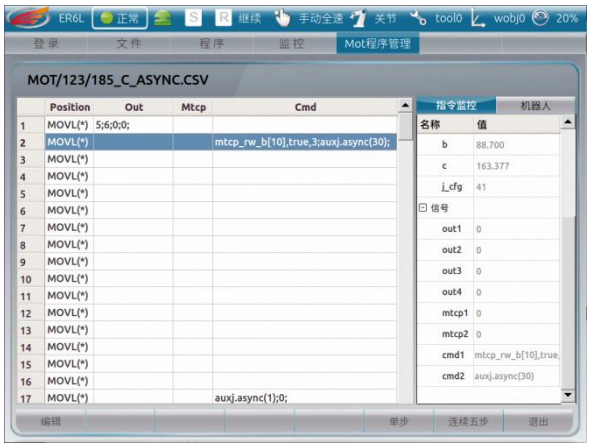
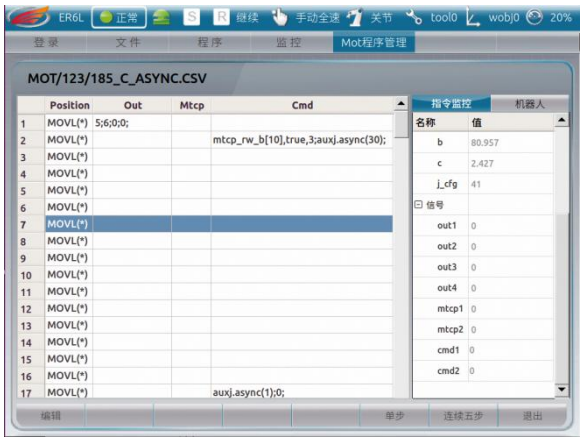
8.设置 cmd 信号	  	Cmd 类型：分为未使用、脉冲、异步轴。三种 cmd 的格式为： 未使用：0。 脉冲：DOut[port], value, time 或 mtcp_rw_b [port], value, time。 附加轴：auxj.async (angle)。 **1)脉冲参数说明** 脉冲信号：可选择 io 和 modbus 可读可写 bool 变量。 端口(port)：输出端口号。 值(value)：输出的值，true 或 false。 时间(time)：输出脉冲的持续时间。 **2)附加轴参数说明** 步进角度(angle)：附加轴异步运动的角度值。附加轴轴数由总线信号确定，如果程序中使用同步轴，则异步轴指令不生效。

9.保存程序。	 	点击“**保存**”按钮,会将文件保存并下发控制器，成功后有如图提示。

17.3.2 程序运行

表 16-12 程序运行操作步骤

步骤	图示	说明
1. 打开 Mot 程序。		选址想要编辑的 Mot 程序，点击**“打开”**按钮，则可以查看程序内容。

		
2. 单步运行。		打到自动模式并上伺服，或者手动模式下，按住手压。 点击“**单步**”按钮。机器人运动到该点。
3.连续 5 步		打到自动模式并上伺服，或者手动模式下，按住手压。 点击“**连续五步**”按钮。机器人按程序连续运行 5 个位置点。

17.4 Mot 程序使用

Mot 程序作为子程序通过 RPL 程序 Module 指令调用运行。

17.4.1 指令

17.4.1.1 mot.exec()

表 16-13 Mot.exec()说明

名称	DINT mot.exec(STRING filename,LREAL execfr)
filename	Mot 程序路径

execfr	设置运行速度，范围 0.5~2
Result	返回 Mot 程序运行状态：0 是 Mot 禁止；1 是 Mot 空闲；2 是运行；3 是丢弃；4 是错误

例子：

Mot_status = Mot.exec(“/Mot/1_c.csv”,1)

程序调用示例：

图 16-1 程序调用界面

第 18 章　传送带跟踪

18.1 光电跟踪功能介绍

通过 5 点标定，计算传送带分辨率、传送带固定坐标系；当光电传感器检测到物体时，即知物体在固定传送带坐标系下的位置，根据记录的编码器值即可将用户坐标系动态的建立在物体上，实现机器人跟踪抓取功能。

18.1.1 硬件准备工作

在进行使用跟踪视觉功能之前，应做好以下准备：

1）若控制器类型为 RP1：

a. 编码器类型：ABZ 三相增量式编码器。电压需选择 5V 的，由 RP1 控制器提供电源。

b. 传送带上编码器线与控制器（RP1）硬件接口 ENC 相匹配。编码器接口连接方式如图 17-1。

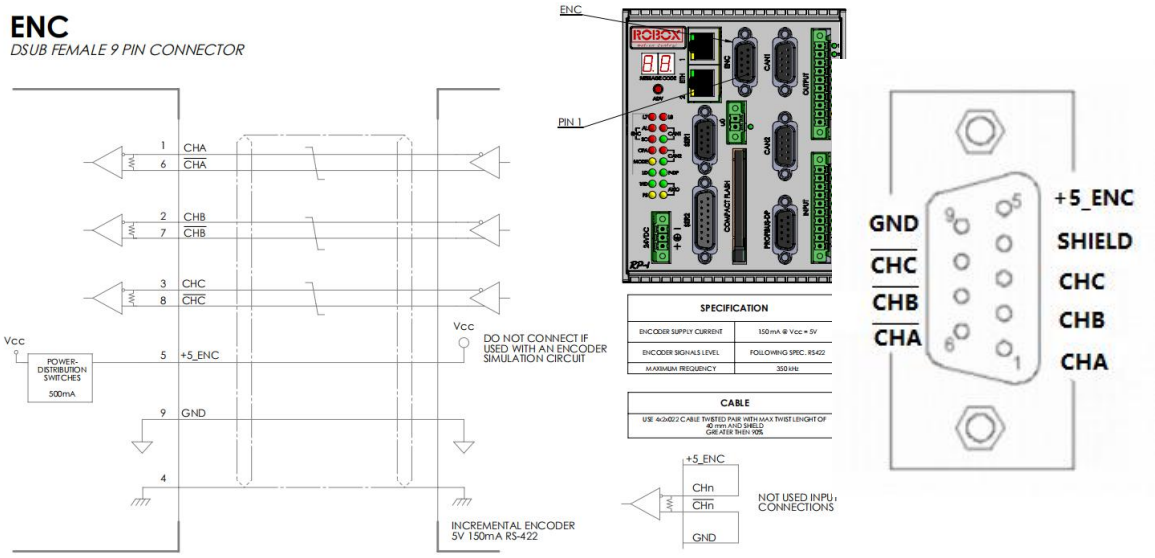


图 17-1 编码器连接图示例

2）若控制器类型为 RP2：

a. 编码器类型：ABZ 三相增量式编码器。电源由外部提供，根据需求选择合适的电压。

b. 转接模块：选择转接模块时需满足支持 EthCat 通讯协议和增量式编码器接口（推荐品牌：AM600-2HCE）。

c. 转接模块与 RP2 的通讯连接：转接模块的网口可以连接 RP2 未被占用的 EthCat 的网口。

d. 联系埃夫特技术支持人员获取与转接模块相匹配的控制器工程。

3）反光式光电传感器：输出信号类型为光电开关信号，安装在传送带一端；目前软件上只支持光无遮挡时给控制器输入高电平，有遮挡时，给控制器输入低电平，以下降沿作为有料触发信号，工作电压为 24V，具体接线如图 17-2 所示。

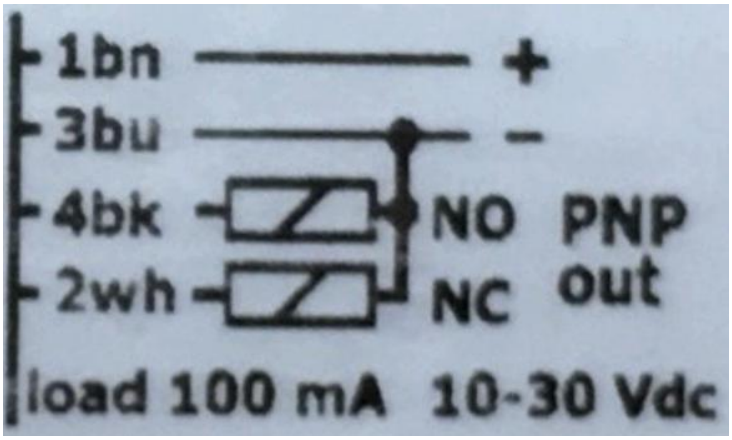


图 17-2 光电传感器连接图示例

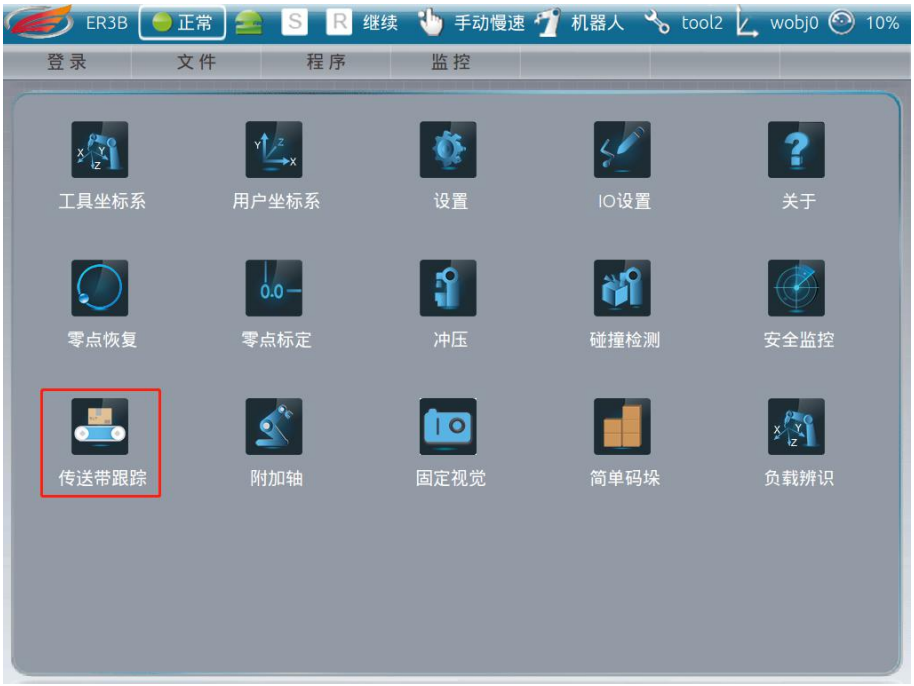
4）传送带坐标系的确定：传送带运动编码器值增加的方向为传送带固定坐标系+X 方向；垂直于传送带平面向上规定为传送带固定坐标系+Z 方向；根据右手坐标系确定 Y 轴方向。

18.1.2 光电跟踪界面介绍

18.1.2.1 光电跟踪主界面

表 17-1 光电跟踪主界面

步骤	图示
1. 打开示教器桌面，点击“传送带跟踪”功能图标进入传送带跟踪功能界面。	

	
2.点击“光电跟踪”，进入光电跟踪主界面。	

3.光电跟踪主界面	

1）光电跟踪功能开关：打开状态，启用光电跟踪功能；关闭状态，停用光电跟踪功能。

2）在固定传送带坐标系下工件位置：点击“获取数据”按钮后，用来显示队列中当前工件在固定传送带坐标系的位置值。

3）光电 IO 端口：可以根据 IO 端口配置光电信号的地址。

4）获取数据：当点击“获取数据”按钮后，获取当前工件在固定传送带坐标系的位置。

5）清除当前工件：用来清除队列中当前工件。

6）清除所有工件：用来清除队列中所有的工件。

7）传送带标定按钮：点击“传送带标定”按钮，会进入传送带标定界面。

8）返回：点击“返回”按钮，返回到传送带跟踪功能界面。

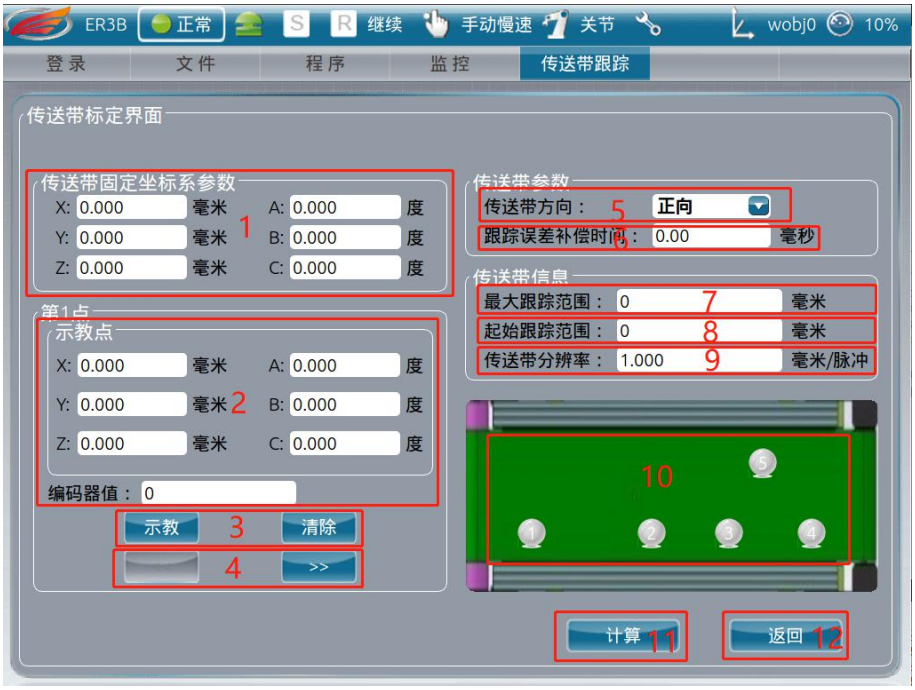
9）光电触发方式：选择“上升沿”触发还是“下降沿”触发。

10）传送速度：传送带的速度，传送带速度不可更改。

18.1.2.2 传送带标定界面介绍

表 17-2 传送带标定界面

步骤	图示

1.传送带标定界面	

1）传送带固定坐标系参数：5 点标定计算得出的传送带固定坐标系原点在当前工具坐标系下的值。

2）示教值：点击“示教”后，用来显示工件分别在工具坐标系下的值和传送带上编码器的值。（注意：1 号点只显示编码器值）

3）示教/清除：点击“示教”按钮，记录工件在当前工具坐标系下的值和编码器值；点击“清除”按钮，清除当前点工件在当前工具坐标系下的值和编码器值。

4）点切换：用于切换当前点。

5）传送带方向：设置传送带的正反向，编码器值增减的方向为正向，编码器值减小的方向为反向。

6）跟踪误差补偿时间：跟随误差补偿时间。

7）最大跟踪范围：标定完成后计算出地最大跟踪范围值。

8）起始跟踪范围：标定完成后计算出地起始跟踪范围值。

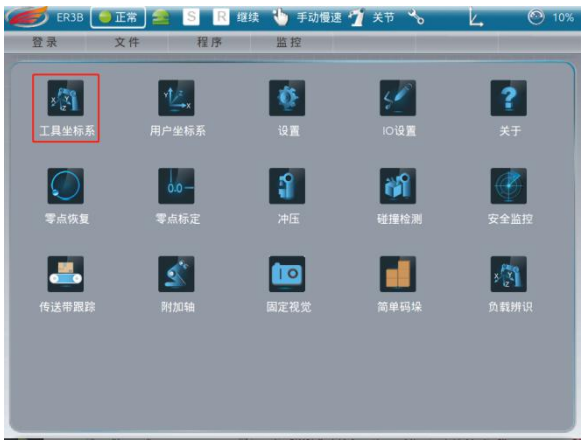
9）传送带分辨率：标定完成后计算出地传送带分辨率值。

10）传送带上标定点的大概位置：共标定 5 个点，其中第 1 点为工件刚好挡住光电传感器光线的位置，第 2 点为跟踪起始点，第 3 点为坐标系 x 方向的第二个点，第 4 点为最大跟踪终点，第 5 点与第 2、3 点一起确定传送带固定坐标系参数。其中 2,3 为 x 方向，5 为 y 方向，2 点为坐标系原点。

18.1.3 光电跟踪功能标定

表 17-3 光电跟踪功能标定

步骤	图示	说明

1. 进行工具坐标系标定		具体见第 6 章 6.2 节。下面以 tool0 为例进行光电跟踪的传送带标定。
2.进入光电跟踪主界面。		1）打开光电功能开关； 2）配置光电 IO 端口，确保该地址没有被占用。
3.点击“传送带标定”按钮，进入传送带标定界面		1）设置传送带方向； 2）示教第 1 点。

4.示教第 2~5 点		1）完成第 2~5 点示教
5.点击“计算”按钮		1）计算成功后，显示传送带固定坐标系参数及传送带信息。
6.点击“返回”按钮，进入光电跟踪功能主界面		1）点击“获取数据”，会获取队列当前工件在固定坐标系下的值。

18.1.4 Module 指令及 RPL 程序用例

18.1.4.1 Module 指令说明

表 17-4 光电跟踪功能指令说明

指令	名称	功能
track.init()	跟踪视觉功能初始化命令	初始化跟踪视觉相关功能
track.getWobj(n)	获取队列数据命令	调用该命令，获取跟踪队列中的物体位置数据，并将跟踪物体坐标系 wobj_cvy 建立

		在该物体上；n 为 DINT 型，表示外部输入 io 端口号，用来控制机器人是否等待获取数据。例如 track.getWobj(7)，为 7 号输入 IO，IO 为 true 时，程序中断等待，直接运行下一步，当 IO 为 false 时，程序在 getobjReady：=track.getWobj（7）位置等待，直到获取数据，进行跟踪抓取
track.dropObj()	删除当前队列数据命令	调用该命令，可以删除队列中当前的一组数据
track.dropAllObj()	清空队列数据命令	调用该命令，将删除队列中所有数据
坐标系	名称	功能
wobj_cvy	跟踪物体坐标系	建立在传送带上的移动物体的坐标系
wobj_cvy_fixed	传送带固定坐标系	建立在传送带上的固定坐标系

18.1.4.2 RPL 程序用例

光电跟踪程序运行时，track.getWobj()会将用户坐标系 wobj_cvy 直接建立在传送带动态的物体上，即 wobj_cvy 坐标系的原点建立在物体上。因此在 wobj_cvy 下跟踪和抓取物体时，可直接将目标点定位原点位置。

为了完整的演示光电跟踪工作流程，本用例分为 main 函数和 skipMove 函数两个部分：

1.Main（）为机器人跟踪抓取传送带上动态物体的工序流程；

2.skipMove（）为机器人跟踪抓取失败后的处理动作；

程序中共用到 5 个位置点，具体数据如下：

表 17-5 位置点信息

位置 1	准备位置	POINTJ（-41，87，35，0，0，0）
位置 2	等待位置	PONITC（20，0，50，0，0，0）
位置 3	跟踪位置	PONITC（0，0，50，0，0，0）
位置 4	抓取位置	PONITC（0，0，0，0，0，0）
位置 5	放置位置	POINTJ（-41，114，35，0，0，0）

用户应用时，需要根据实际工作流程编写相应程序。

表 17-6 光电跟踪程序用例

步骤	图示

Main 函数

```
1  IF firstRun = false THEN
2      firstRun := true ;
3      track.init() ;
4      cvyAxisGroup := ROBOT("linearbelt") ;
5      INTRDIS (int1) ;
6      INTRSET (int1, skipMove(toolflag)) ;
7      INTRERRNO (int1) ;
8      track.dropAllObj() ;
9      speed := v3000 ;
10 END_IF ;
11 wobj_cvy_fixed := GETWOBJ(false, cvyAxisGroup) ;
12 (* Go to ready position *)
13 MJOINT (*, speed, fine, tool0) ;
14 LABEL loop :
15 cnt := cnt + 1 ;
16 MESSAGE ("Piece number = %1", cnt) ;
17 (* Wait for a valid piece *)
18 getObjReady := false ;
19 IF NOT getObjReady THEN
20     MLIN (*, speed, fine, tool0, wobj_cvy_fixed) ;
21     getObjReady := track.getWobj(7) ;
22 END_IF ;
23 INTRENA (int1) ;
24 MESSAGE ("wobj = %1", wobj_cvy) ;
25 (* Move over piece *)
26 MLIN (*, speed, fine, tool0, wobj_cvy) ;
27 (* Go to grap piece *)
28 MLIN (*, speed, fine, tool0, wobj_cvy) ;
29 (* Grap the piece *)
30 DWELL (0.5) ;
31 (* Move back over piece *)
32 MLIN (*, speed, fine, tool0, wobj_cvy) ;
33 INTRDIS () ;
34 track.dropObj() ;
35 (* Go to deposit piece *)
36 MJOINT (*, speed, fine, tool0) ;
37 GOTO loop ;
```

Line1：判断是否为该程序的首次运行，首次运行标志信号 firstRun，当 firstRun=false 时，代表首次运行，执行 IF 语句中的 Line2-9 程序；当 firstRun=true 时，说明不是首次运行，跳过 IF 语

句，直接执行 Line11

Line2：将 firstRun 信号置位 1，代表首次执行完毕

Line3：初始化跟踪视觉相应功能

Line4：获取系统中名字为“conveyor”的轴组的数据

Line5：关闭中断程序 int1 的使能

Line6：设置中断变量，将一个 intr 变量 int1 与一个子函数 skipMove()关联，当变量 int1 被触发时，程序自动跳入子函数 skipMove()中，执行子程序内的语句

Line7：设置 int1 的触发条件：当_errno_!=0 时，启动中断程序 int1

Line8：清空队列中的所有数据

Line9：定义速度变量 speed

Line10：IF 语句结束

Line11：定义传送带固定坐标系

Line13：将机器人运动到准备位置 1

Line14：开启循环 Loop：

Line15：抓取的物体数量记录

Line16：在日志中打印抓取的物体数量信息

Line18：获取数据状态复位，获取数据的状态信号 getObjReady

Line19：判断是否获取数据成功，当状态信号 getObjReady=false 时，说明当前未获取到数据，执行 IF 语句中的 Line20-21；当 getObjReady=true 时，跳过 IF 语句，直接执行 Line23

Line20：将机器人移动到传送带上方位置 2，准备抓取物体

Line21：获取队列中的物体数据

Line22：IF 语句结束

Line23：开启中断程序 int1 的使能

Line24：在日志中打印出建立在动态物体上的坐标系参数信息

Line26：机器人跟随物体运动到动态物体上方位置 3

Line28：机器人向下运动抓取物体位置 4

Line30：延时 0.5s，这里模拟机器人抓取物体的时间

Line31：机器人向上运动回物体上方位置 3

Line33：关闭中断程序 int1 的使能

Line34：删除队列中当前数据（因为当前数据已完成抓取，将该数据删除，获取下一组数据）

Line36：机器人运动到物体放置位 5

Line37：GOTO loop 循环终止点，从 line14 开始下一次循环

步骤	图示
skipMove()函数	1 track.dropObj() ; 2 STOPMOVE () ; 3 CLEARMOVE ; 4 STARTMOVE () ; 5 cntSkipped := cntSkipped + 1 ; 6 (* Go to start position *) 7 MJOINT (*, v1000, fine, tool0, wobj_cvy_fixed) ; 8 MESSAGE ("Piece skipped %1", cntSkipped) ; 9 RESTART ;

Line1：删除队列中当前数据

Line2：机器人停止当前动作

Line3：清除未完成的动作

Line4：开始接下来的运动

Line5：记录跟踪失败的物体数量

Line7：机器人移动到传送带上方位置 2，准备抓取下一个物体

Line8：在日志中打印跟踪失败的物体数量

Line9：程序重新开始

18.2 2D 视觉跟踪功能介绍

跟踪（Tracking）是指机器人工具末端 TCP（Tool Center Point，不是指网络 TCP /IP 协议）跟随一个运动的物体。一个典型的例子是跟随传送带上的一个物体。该功能通过相机拍照，获取传送带上物体的位姿（X、Y、A 等值），机器人末端跟随该物体运动，并抓取该物体。

18.2.1 机器人与视觉系统上位机准备工作

在进行使用跟踪视觉功能之前，应做好以下准备：

1）传送带编码器根据控制器的类型进行相关准备，详细见 17.1.1 节 1）、2）处内容。

2）相机安装：传送带的 Z 轴正方向垂直于传送带平面向外，相机的 Z 轴正方向与传送带的 Z 轴正方向相同，相机端 X 轴正方向与传送带 X 轴正方向（传送带的 X 轴正方向：传送带的运动前进的方向）的夹角为锐角。

3）视觉相机处于 IO 触发模式（硬件触发）。

4）机器人与相机端 TCP/IP 网络通讯协议（见 17.2.2 节）。

18.2.2 TCP/IP 通讯协议及数据格式

在使用过程中，视觉系统（相机）需要将图像处理后的工件信息通过机器人提供的固定通讯格

式传输给机器人，机器人根据接收到的数据进行取放动作。因此，机器人通讯格式主要包括三个部分：

1. 物体坐标参数：X、Y、A

2. 物体属性参数：ATTR

3. 物体 ID 编码：ID

1) 物体坐标参数是指物体在相机视野范围内的位置及旋转角度，该位置为相机/像素坐标系（单位 mm 或 px）下的位置。

2) 物体属性参数是指根据物体不同属性（例如：形状、颜色等）给出物体的对应属性值，以数字：0、1、2、3……来表示。

3) 物体 ID 编码是指为了方便管理给每一个物体制定的唯一编码。

属性参数与 ID 编码用户可根据实际情况选择是否使用以及具体的使用方式，如不需要应用，在相机通讯格式设置时将其默认为 0 即可。

具体通讯格式如下：

Image\r\n

[X:xxx.xxx;Y:xxx.xxx;A:xxx.xxx;ATTR:xxx;ID:xxx]\r\n

……

[X:xxx.xxx;Y:xxx.xxx;A:xxx.xxx;ATTR:xxx;ID:xxx]\r\n

Done\r\n

上述格式中，“Image”表示数据头，即一组图像数据下发开始的标志。“Done”表示数据尾，即一组图像数据下发完成（注意“Image”和“Done”区分大小写）。“\r\n”为回车换行符。“[X:xxx.xxx;Y:xxx.xxx;A:xxx.xxx;ATTR:xxx;ID:xxx]”表示相机下发的一个物体的数据，其中包含了物体坐标参数 XYA，物体属性参数 ATTR 和物体 ID，每个数据用“数据名:数据;”的格式表示，每个物体的数据均以“[”开始，以“]”结束。**当相机没有拍到物体或者识别物体失败时发送字符串“Error”**。

注：跟踪视觉一次最多支持传输 10 组数据，固定视觉一次只能传输 1 组数据。

18.3 跟踪视觉 APP 界面介绍

18.3.1 跟踪视觉主界面

点击左上角埃夫特图标进入桌面，点击桌面上的“**传送带跟踪**”按钮，进入传送带跟踪 APP 主界面，如图 17-3 所示：

图 17-3 跟踪视觉主界面

图中红框内信息说明如下：

1) 跟踪视觉开关：用来是否开启控制器中的跟踪视觉功能。

2) TCP/IP 连接状态：表示当前机器人与相机通讯的连接状态，“**灰色**”表示当前处于断开状态，“**绿色**”表示当前处于连接状态。

3) 在相机坐标系下工件位置：工件在相机坐标系下的位置值。

4) 在固定传送带坐标系下工件位置：工件在标定的固定传送带坐标系下的位置值。

5) 传送带速度、循环拍照开关、循环拍照间隔、相机触发指令：显示实时传送带速度；确认是否打开循环拍照开关；设置循环拍照的时间间隔；设置相机触发指令。

6) 获取数据、清除当前工件、清除所有工件：获取当前工件位置值；将当前工件从队列中删除；清除队列中所有的工件。

7) TCP/IP 设置：相机参数设置。

8) 传送带标定：进入传送带或像素分辨率标定界面。

9) 抓取标定：进行偏置抓取点标定。

10) 退出：退出跟踪视觉 APP 界面，返回到主界面。

18.3.2 跟踪视觉设置界面

点击跟踪视觉主界面的“**设置**”按钮，进入设置界面，设置界面如图 17-4 所示：

图 17-4 TCP/IP 设置界面

图中红框内信息说明如下：

1. 相机连接设置：

1) 相机品牌：目前相机品牌可选通用相机、康耐视相机和麦克玛视相机；当选择康耐视相机时，需要输入相机账号和密码。

2) 相机登录账号：当选择康耐视相机时，需要输入相机上设置好的登录账号。

3) 登录密码：当选择康耐视相机时，需要输入相机上设置的登录密码。

4) 连接类型：连接相机的方式有手动连接和开机自动连接两种。

5) 相机 IP 地址：需要输入相机上设置的 IP 地址。

6) 相机端口：需要输入相机的端口号。

7) 数据格式：设置数据格式，目前只有一种数据格式。

8) 连接超时：当选择自动连接方式时，设置连接超时报警时间。

2. 拍照触发设置：

9) 相机触发方式：有指令触发和 IO 触发两种方式；当选择指令触发时，相机触发 IO 呈灰色，不可设置。

10) 拍照时间间隔：用来设置拍照的时间间隔（与主界面相同）。

11) 相机触发 IO：当相机触发方式选择为 IO 触发时，可设置触发 IO 的地址，跟踪视觉需要选择 IO 触发模式。

12) 相机触发指令：在相机设置好命令触发方式后可在此处进行设置（与主界面相同）。

13) 相机数据获取指令：此次设置只针对康耐视相机，当登录康耐视相机后，此设置才生效。

14) 保存、连接、返回按钮：当所有的设置完成后，须点击“**保存**”按钮，会将设置信息保存到控制器中；点击“**连接**”后，机器人会跟相机进行连接，连接成功后，界面右上角

的状态会变成绿色；返回到主界面时须点击“返回”按钮。

18.3.3 像素分辨率标定界面

点击“**传送带标定**”按钮，会弹出是否使用像素分辨率标定提示框，如下图 17-5 所示，点击“**是**”，进入像素分辨率标定界面，如下图 17-6 所示：

图 17-5 像素分辨率提示框

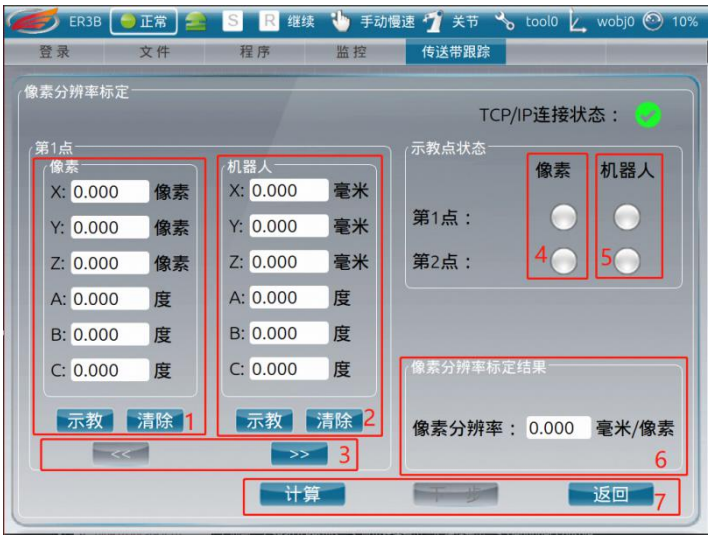


图 17-6 像素分辨率标定界面

图中红框内信息说明如下：

1) 像素：工件在相机坐标系下像素值。

2) 机器人：工件在机器人坐标系的坐标值。

3) 上下点切换：切换到上一点和下一点。

4) 像素示教状态：像素标定是否成功状态。

5) 机器人示教状态：机器人坐标值标定是否成功状态。

6) 像素分辨率标定结果：标定后计算得到的像素分辨率标定结果。

7) 计算、下一步、返回按钮：计算标定的结果；进入到传送带标定界面；返回的跟踪视觉主界面。

18.3.4 传送带标定界面

点击“**传送带标定**”按钮，点击弹出是否使用像素分辨率标定提示框上的“**否**”，或当像素分辨率标定计算成功后，点击“**下一步**”按钮，进入传送带标定界面，如图 17-7 所示：

图 17-7 传送带标定界面

图中红框内信息说明如下：

1) 传送带固定坐标系参数：显示标定后的计算结果；

2) 示教点：显示坐标值及编码器值；

3) 上下点切换：切换上一点与下一点；

4) 传送带参数：设置传送带方向及跟踪误差补偿时间，显示编码器计数值；

5) 传送带信息：显示标定后计算的相关结果；

6) 标定状态：标定点的示教是否成功状态；

7) 计算、返回按钮：计算标定结果；返回到跟踪视觉主界面。

18.3.5 抓取标定界面

在主界面点击“**抓取标定**”按钮，进入抓取标定界面，如图 17-8 所示：

图 17-8 抓取标定界面

图中红框内信息说明如下：

1) 示教点：显示标定点坐标值及编码器值；示教按钮；清除示教按钮；

2) 上下点切换：切换上一点及下一点；

3) 标定状态：示教点标定是否成功的状态；

4) 计算、返回按钮：计算标定结果；返回到跟踪视觉主界面。

5) 抓取标定结果：显示当前抓取标定的结果。

6) 编辑/保存按钮，点击可对抓取标定结果直接进行编辑保存操作。

18.4 跟踪视觉的标定

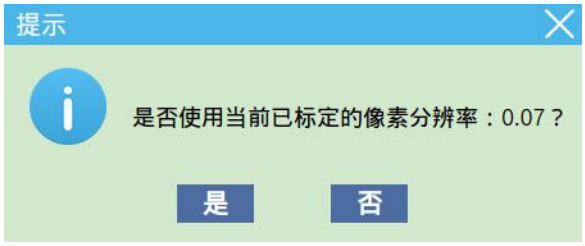
18.4.1 TCP/IP 设置

表 17-7 参数设置及相机连接连接

步骤	图示	说明
1. 点击“**TCP/IP 设置**”按钮，进入设置界面。		选择使用的相机品牌，完成相关设置，相机触发方式选为 IO 触发，根据硬件 IO 接线输入相机触发 IO 口地址，点击“**保存**”，然后点击“**连接**”按钮与相机进行连接，连接上后返回到主界面。

18.4.2 像素分辨率标定

表 17-8 像素分辨率标定步骤

步骤	图示	说明
1. 点击“**传送带标定**”按钮，弹出弹框提示。	 	如果使用像素分辨率标定则点击“**是**”，若当前存在已标定的像素分辨率，则弹出是否使用当前像素分辨率的弹出框，选择“是”则直接使用当前的像素分辨率，进入传送带标定界面，选择“否”则重新标定，进入像素分辨率标定界面； 若不使用，则选择“**否**”，进入传送带标定界面。
2.若进入了像素分辨率标定界面，则进行像素分辨率标定操作。	 	需标定传送带上相机拍照范围内的 2 个点，标定的这 2 个点的位置尽量在相机拍照范围的对角边上。 首先进行第 1 点标定，确保机器人移动到相机视觉范围外，将工件放在相机拍照范围内，点击“**示教**”按钮，在相机像素坐标值显示工件的位置值，并且在示教状态中第 1 点像素由“**灰色**”变为“**黄色**”，表示第一点像素示教成功，再进行第 1 点机器人位置示教，将机器人末端工具移动到工件表面上方，点击“**示教**”按钮，传送带标定在机器人坐标值会更新，并且在机器人示教状态中第 1 点由“灰**色**”变为 “**黄色**”，表示第一点机器人坐标值

		示教成功。 进行第 2 点标定，点击“>>”按钮，切换到第 2 点，进行第 2 点像素坐标值和机器人坐标值示教，步骤同第 1 点；两点示教完成。
3.点击“计算”按钮		计算成功后会弹出成功提示框，然后点击“**是**”，将更新像素分辨率标定的结果，“**下一步**”按钮由灰色变为蓝色，点击“**下一步**”，进入传送带标定界面。

18.4.3 传送带标定

表 17-8 传送带标定操作

步骤	图示	说明
1. 在传送带标定界面进行传送带标定操作。	 	进入传送带标定界面后，传送带标定需要标定 6 个点，第 1、2 点为相机标定点，第 3、4、6 为机器人标定点，确定传送带的方向。 首先进行第 1、2 点标定，将工件放置在相机拍照范围内的第 1 个点，然后点击“**示教**”，示教器成功后，状态由“**灰色**”变为 “**黄色**”。 点击“>>”，保证工件与传送带相对位置不变，开动传送带，将工件运送到要标定的第 2 点位置，点击“**示教**”。

2.进行 3~6 点标定操作		点击“>>”，进行第 3 点标定，开动传送带运行至第 3 点位置，将机器人末端工具移动到工件表面上方，点击“**示教**”，示教成功后，进行第 4 点标定；开动传送带，运动标定位置 4，将机器人移动工件表面上方，点击“**示教**”，示教成功后，进行第 5 点标定；开动传送带，运动标定位置 4，将机器人移动工件表面上方，点击“**示教**”，示教成功后，进行第 6 点标定；将工件放到传送带 Y 方向的某个位置，点击“**示教**”。
3.所有点示教完成后，点击“计算”按钮		计算成功后，点击提示框上的“**是**”，计算结果更新在传送带固定坐标系参数和传送带信息中。

上述第 1、2 点需要记录的是相机/像素坐标值，因此标定时此两点需要在相机拍照范围内，第 3、4、5、6 点需要记录的是机器人坐标系的下的值。第 3 点为机器人抓取范围的起始点，第 5 点为机器人抓取范围的终止点。即 3-5 点之间为机器人的最大跟踪范围。第 4 点是判断工件是否可以开始跟踪的位置，根据实际情况调整，一般会设置在第 3、5 点中间的位置。

18.4.4 抓取点标定

实际抓取点的位置可能并不是相机拍照特征点坐标的位置，增加了偏置抓取点标定功能。因抓取点标定记录传送带位置需要实际值，因此应在传送带标定之后，设置正确的传送带分辨率之后进行。

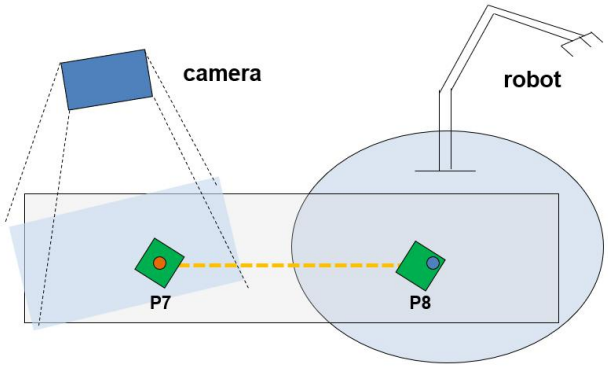


图 17-9 抓取点标定示意

操作步骤：

1) 将物体放置在相机视野范围内 P7 点处，记录物体的相机像素坐标位置和编码器值，并通过已求出的像素分辨率，将像素坐标位置转化为相机坐标系下位置。黄色圆圈代表物体相机拍照坐标点位置。

2) 移动传送带，物体到达机器人工作空间内 P8 点处，将机器人工具末端移动到想要抓取的偏置位置，例如蓝色圆圈代表实际的偏置抓取位置。记录此时机器人坐标系下的位置。

3) 点击“**计算**”，计算成功后抓取点标定生效。

18.5 Module 指令及 RPL 程序用例

18.5.1 Module 指令说明

表 17-9 Module 指令说明

指令	名称	功能
track.init()	跟踪视觉功能初始化命令	初始化跟踪视觉相关功能
track.getWobj()	获取队列数据命令	调用该命令，获取跟踪队列中的物体位置数据，并将跟踪物体坐标系 wobj_cvy 建立在该物体上
track.setTrigCmd(intp)	设置相机触发指令	相机在指令触发的模式下，该命令可设置相机触发的指令
track.dropObj()	删除当前队列数据命令	调用该命令，可以删除队列中当前的一组数据
track.dropAllObj()	清空队列数据命令	调用该命令，将删除队列中所有数据
坐标系	**名称**	**功能**
wobj_cvy	跟踪物体坐标系	建立在传送带上的移动物体的坐标系
wobj_cvy_fixed	传送带固定坐标系	建立在传送带上的固定坐标系
参数	**名称**	**功能**
ivt_objid	工件 ID	相机识别的物体的 ID（在外部变量中建立）

18.5.2 跟踪视觉程序用例

18.5.2.1 RPL 程序设计思路

跟踪视觉程序运行时，track.getWobj()会将用户坐标系 wobj_cvy 直接建立在传送带动态的物体

上，即 wobj_cvy 坐标系的原点建立在物体上。因此在 wobj_cvy 下跟踪和抓取物体时，可直接将目标点定位原点位置。

为了完整的演示跟踪视觉工作流程，本用例分为 main 函数和 skipMove 函数两个部分：

1. skipMove（）为机器人跟踪抓取失败后的处理动作；

2. Main（）为机器人跟踪抓取传送带上动态物体的工序流程；

程序中共用到 5 个位置点，具体数据如下：

表 17-10 位置点信息

位置 1	准备位置	POINTJ（-41，87，35，0，0，0）
位置 2	等待位置	PONITC（20，0，50，0，0，0）
位置 3	跟踪位置	PONITC（0，0，50，0，0，0）
位置 4	抓取位置	PONITC（0，0，0，0，0，0）
位置 5	放置位置	POINTJ（-41，114，35，0，0，0）

用户应用时，需要根据实际工作流程编写相应程序。

1) Main（）函数

```
IF firstRun = false THEN
    firstRun := true ;
    track.init() ;
    cvyAxisGroup := ROBOT("linearbelt") ;
    INTRSET (int1, skipMove()) ;
    INTRERRNO (int1) ;
    INTRDIS (int1) ;
    track.dropAllObj() ;
    speed := v3000 ;
END_IF ;
wobj_cvy_fixed := GETWOBJ(false, cvyAxisGroup) ;
(* Go to ready position *)
MJOINT (*, speed, fine, tool1) ;   位置1：准备位置
LABEL loop :
cnt := cnt + 1 ;
MESSAGE ("Piece number = %1", cnt) ;
(* Wait for a valid piece *)
getObjReady := false ;
```

```
19 IF NOT getObjReady THEN
20    MLIN (*, speed, fine, tool1, wobj_cvy_fixed) ;  位置2：等待位置
21    getObjReady := track.getWobj() ;
22  END_IF ;
23  INTRENA (int1) ;
24  MESSAGE ("wobj = %1", wobj_cvy) ;
25  (* Move over piece *)
26  MLIN (*, speed, fine, tool1, wobj_cvy) ;  位置3：跟踪位置
27  (* Go to grap piece *)
28  MLIN (*, speed, fine, tool1, wobj_cvy) ;  位置4：抓取位置
29  (* Grap the piece *)
30  DWELL (0.5) ;
31  INTRDIS () ;
32  (* Move back over piece *)
33  MLIN (*, speed, fine, tool1, wobj_cvy) ;  位置3：跟踪位置
34  track.dropObj() ;
35  (* Go to deposit piece *)
36  MJOINT (*, speed, fine, tool1) ;  位置5：放置位置
37  GOTO loop ;
```

图 17-10 Main（）函数

Line1：判断是否为该程序的首次运行，首次运行标志信号 firstRun，当 firstRun=false 时，代表首次运行，执行 IF 语句中的 Line2-9 程序；当 firstRun=true 时，说明不是首次运行，跳过 IF 语句，直接执行 Line11；

Line2：将 firstRun 信号置位 1，代表首次执行完毕；

Line3：初始化跟踪视觉相应功能；

Line4：获取系统中名字为“conveyor”的轴组的数据；

Line5：设置中断变量，将一个 intr 变量 int1 与一个子函数 skipMove()关联，当变量 int1 被触发时，程序自动跳入子函数 skipMove()中，执行子程序内的语句；

Line6：设置 int1 的触发条件：当_errno_!=0 时，启动中断程序 int1；

Line7：关闭中断程序 int1 的使能；

Line8：清空队列中的所有数据；

Line9：定义速度变量 speed；

Line10：IF 语句结束；

Line11：定义传送带固定坐标系；

Line13：将机器人运动到准备位置 1；

Line14：开启循环 Loop；

Line15：抓取的物体数量记录；

Line16：在日志中打印抓取的物体数量信息；

Line18：获取数据状态复位，获取数据的状态信号 getObjReady；

Line19：判断是否获取数据成功，当状态信号 getObjReady=false 时，说明当前未获取到数据，执行 IF 语句中的 Line20-21；当 getObjReady=true 时，跳过 IF 语句，直接执行 Line23；

Line20：将机器人移动到传送带上方位置 2，准备抓取物体；

Line21：获取队列中的物体数据；

Line22：IF 语句结束；

Line23：开启中断程序 int1 的使能；

Line24：在日志中打印出建立在动态物体上的坐标系参数信息；

Line26：机器人跟随物体运动到动态物体上方位置 3；

Line28：机器人向下运动抓取物体位置 4；

Line30：延时 0.5s，这里模拟机器人抓取物体的时间；

Line31：关闭中断程序 int1 的使能；

Line33：机器人向上运动回物体上方位置 3；

Line34：删除队列中当前数据（因为当前数据已完成抓取，将该数据删除，获取下一组数据）；

Line36：机器人运动到物体放置位 5；

Line37：GOTO loop 循环终止点，从 line14 开始下一次循环；

2) skipMove()函数

```
1  track.dropObj() ;
2  STOPMOVE () ;
3  CLEARMOVE ;
4  STARTMOVE () ;
5  cntSkipped := cntSkipped + 1 ;
6  (* Go to start position *)
7  MJOINT (*, v3000, fine, tool1, wobj_cvy_fixed) ;  位置2：等待位置
8  MESSAGE ("Piece skipped %1", cntSkipped) ;
9  RESTART ;
```

图 17-11 skipMove()函数

Line1：删除队列中当前数据；

Line2：机器人停止当前动作；

Line3：清除未完成的动作；

Line4：开始接下来的运动；

Line5：记录跟踪失败的物体数量；

Line7：机器人移动到传送带上方位置 2，准备抓取下一个物体；

Line8：在日志中打印跟踪失败的物体数量；

Line9：程序重新开始；

第 19 章　冲压

19.1 功能简介

该功能包是主要面向冲床加工而开发的一个软件包。它采用面向对象的建模方式，将机器人冲压工作单元中的工作站和夹具动作模块化。

该软件从用户角度出发，具有简单直观的图形界面，可以帮助冲压生产用户快速上手，用户无需了解机器人编程，在配置好机械和电气之后，熟练用户可以在 5 分钟内快速完成冲压工艺设置，然后系统自动为用户生成机器人程序。它提供了动态图形来实时显示机器人的位置、I/O 信号的值以及生产信息，用户可以在生产时随时了解到所需要的信息，安全方面提供了单步测试，空跑测试以及自动逃离功能，以保证机器人和用户设备的安全。

该软件包可以处理以下情况:

1) 用户的机器人冲压单元中有几个设备（进料设备，压机和出料设备，总数有上限），设备随时变动。

2) 客户的冲压机可以是让机器人只放不取，既放又取。

3) 机器人末端配置了多个夹具。

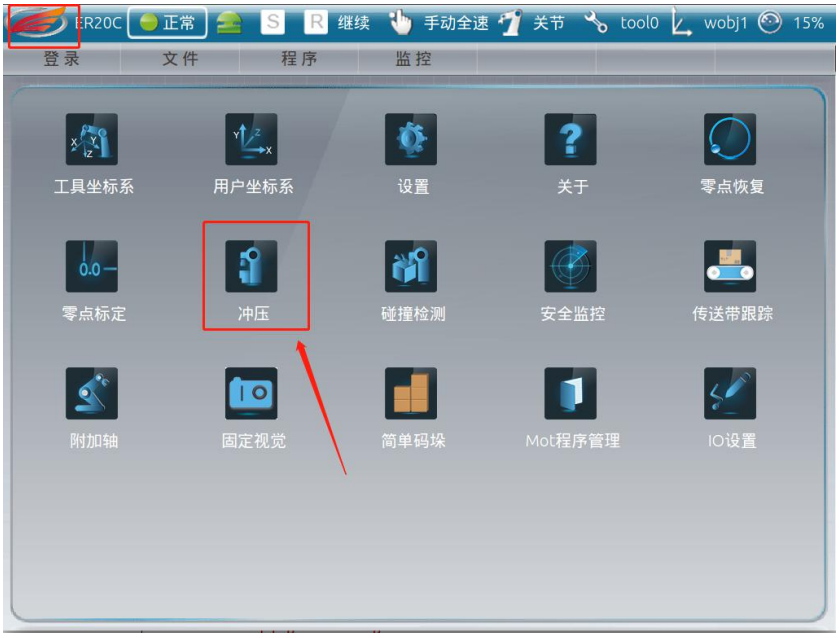
4) 同一个夹具有多个动作，比如打开，闭合，半开等。

19.2 设置界面

设置界面主要用于完成冲压单元的工艺设置。包括单元中的工作站的个数、布局，夹具的动作，每个工作站的机器人路径的编辑，机器人运动位置的示教、速度及动作，及单步走和空跑测试，生产时需要监控的信号等参数的设置。此外还有日产量和月产量的清空，以及设置信息保存导入等功能。

表 18-1 冲压设置操作步骤

步骤	图示

1. 首先，在登录界面点击“登录”，输入“999999”进入管理员模式，如图所示。	
2.点击“Efort”徽标  ,进入桌面，如图所示。	

3.在冲压工艺包主界面可以看到三个图标，“**文件**”、“**设置**”、“**生产**”。“**文件**”图标主要是提供新建冲压文件或者运用已有的冲压文件。“**设置**”图标主要是设置具体的冲压流程，灰色不可点击需要输入密码。“**生产**”图标主要是运行已有的冲压流程。	
4.新建一个文件，如图所示，点击“**文件**”图标后，点击“新建”，输入文件名后点击“**是**”，则可以成功建立一个新的冲压文件。选中新建的冲压文件，点击“**加载**”则可以加载这个文件。	

5.点击密码框，输入密码“333333”，“**设置**”可点击，点击“**设置**”图标，进入冲压设置界面，如图所示。在界面右上角显示机器人当前位置，点击“坐标系”右侧的按钮,切换“**关节**”或者“**笛卡尔**”，改变不同坐标系。点击菜单  可以进行不同的设置。

6.点击  中的“**设备类型**”，则可以选择当前的机器为“**首台机**”、“**中间机**”、“**末端机**”。此外，还可以设置机器人的末端工具，包括工具类型，工具设置等。根据使用需求用户可以灵活的选择“**多台机**”和“**单台机**”，“**首台机**”单独运用时，适用于单机工作，“**多台机**”用于多机连线冲压功能。这里以单台机为例。

7.设置单元，点击菜单  中的设置单元进入页面，该页面用于设置机器人冲压工作单元中的工作站个数及其在窗口中的摆放位置。更重要的，是设置运行时机器人在各工作站之间的行走顺序。Robot 单元是默认单元。

1) 选择左侧列表中的“工作站类型”，再点击按钮  ，选中的“工作站类型”添加到右侧的工作站单元的列表中。“工作站单元”列表第一项固定为“Robot”（机器人）。故当选择列表的第一项时，  按钮变成禁止状态；当选择列表的第二项时，  按钮变成禁止状态。

2) 选择右侧列表中的“工作站单元”，再点击按钮  ,选中的“工作站单元”在列表中上移和下移。

3) 点击按钮  ，删除“工作站单元”列表中的最后一项。若只剩下“Robot”，则该按钮被禁止。

4) 点击“关闭”按钮，关闭当前界面，并返回会冲压设置界面，被添加的工作站单元将在设置界面和生产界面中显示。

5) 设置界面可以根据选中的工作单元进行任意摆放。如果选中“自动摆放”，关闭“设置单元”时界面工作单元将根据列表中的次序，以固定位置进行保存；若没有选中，则根据当前设置界面上各工作单元的实际位置进行保存。

6) 若不点击“逆时针”前的复选框，则工作站单元按默认方向（顺时针）摆放；若选择“逆时针”，则以机器人为中心按逆时针摆放各工作单元。

注：机器人在各工作站之间的行走顺序由各工作站图标上部的数字标出，机器人将按照由低到高的顺序依次经过各站。

“工作站类型”中，目前可选的类型是：Input, Punch1, Punch2， Output，Conveyor，等。“Input”为进料设备，通常是传送带；“Punch1”代表只允许机器人放料的冲压机，机器人不

负责从压机取出的压机类型；“Punch2”代表机器人将工件放入压机，等待压机压完之后再取出的压机类型；“Output”是出料设备，通常是成品框； Conveyor 是带有信号检测的传送带。根据客户需求，后续将持续更新其他工作站类型。

除了在“设置单元”页面中设置单元中各个工作站自动摆放之外，在设置窗口主界面上还可以手动拖动各个工作站的位置，以使得窗口中的显示与实际位置一致，便于观察运行情况。

注意：这里的取料延时设置对 detect tool 动作有效，放料延时设置对 Close Tool 动作有效。

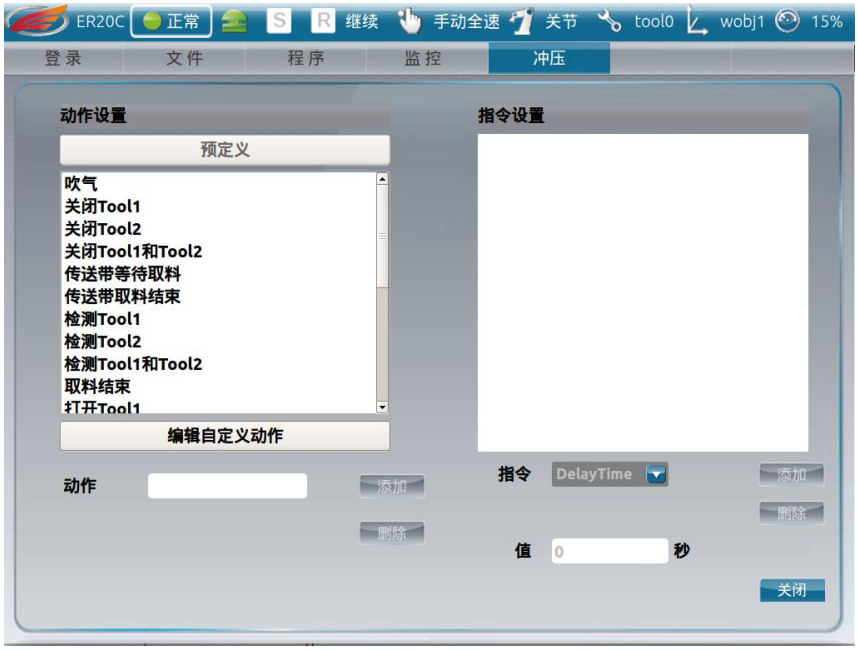
例如：设置点 3 动作，打开工具 1 并延时，可对机器人和点 3 做如下设置

8.设置动作，该页面用于工具夹具的动作，无论单元中有几个工具，每个工具有几个动作，都可以在该页面分别设置。

注：工具的每个动作包含三种最基本的操作：触发输出信号，等待输入信号和延时，这三个指令有不同的参数，对于触发输出信号和等待输入信号，必须要指明信号的具体位置在哪个 IO 模块上以及 IO 引脚地址。

点击菜单

中的“**设置动**



作”，进入“设置动作”界面，如图所示。	

在编辑自定义动作下面（如下图），若左侧动作列表为空，则所有的“**添加**”“**删除**”按钮皆为禁止状态。若左侧动作列表未选中任何一项，则除动作列表下方的“**添加**”按钮，其他按钮一定是禁止状态。若动作名称输入框为空，则动作列表下方的“**添加**”按钮变为禁止状态。

点击动作列表下方的输入框，在弹出的键盘下输入动作名称。输入完成后，动作列表“**添加**”按钮使能。动作输入框为空时，动作列表“**添加**”按钮被禁止。

系统已为客户预先设定了一系列常用工具动作指令（如上步骤 8 图），包括：打开工具，关闭工具，检测工具，传送带等待取料，传送带取料结束等。

1) 点击动作列表“**添加**”按钮，输入框中的动作名称作为项添加到动作列表中，且输入框清空。

2) 点击动作列表“**删除**”按钮，若动作列表未选中任何一项，则删除列表最后一项；选中某一项，则删除选中项。

3) 手动模式并按下手压开关，或者自动模式下点击“**Mot**”按键，此时点击动作列表下的“测试”按钮。指令列表中的命令将会依次执行，且执行过程中状态栏的程序运行状态为 P 指令表中若有 WaitDI 指令，且等待延时数值不为零，则在测试过程中信号等待超时后，会弹出错误提示消息框“**信号等待超时**”。

4) 选中某个动作，点击指令类型下拉框，并选择指令“Send”，出现子指令下拉框和地址下拉框，以及数值的下拉框。

5) 点击子指令下拉框，选择“PLC”，下拉列表中有“PLC”、“DO”、“PfbBus”、“ModBus”四项。

点击地址栏，选择填写端口。

点击数值的下拉框，选择“1”，下拉列表中的“0”和“1”，表示 IO 的输出。

在未选择指令列表中的任何一项时，点击指令列表下的“添加”按钮，“SendPLC_00=0”该项被添加列表的末端。

6) 在选择指令列表中的某一项时，点击指令列表下的“**添加**”按钮，“SendPLC_00=0”该项被添加到当前行的下一行。

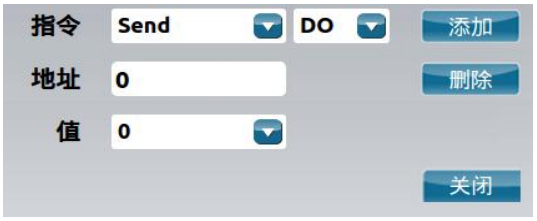
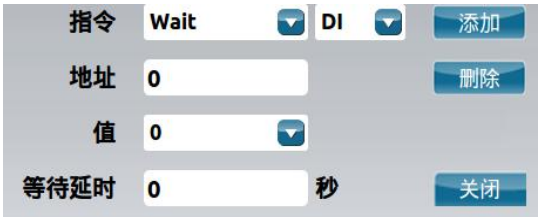
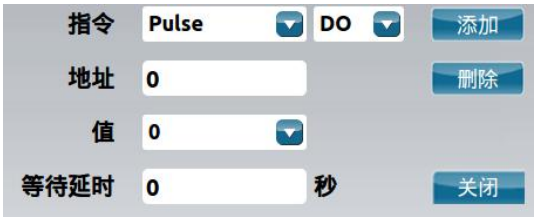
7) 在选择指令列表中的某一项,点击指令列表下的“删除”按钮，该项被删除。

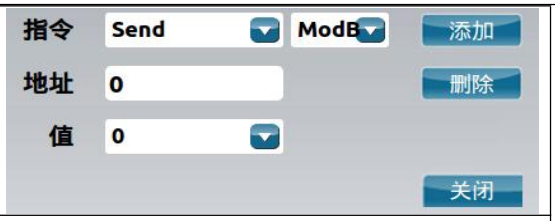
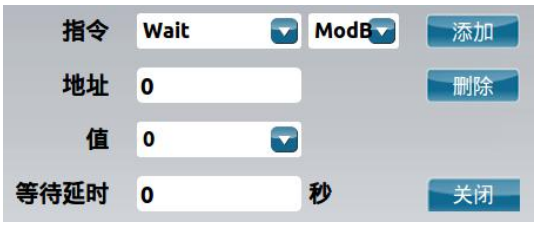
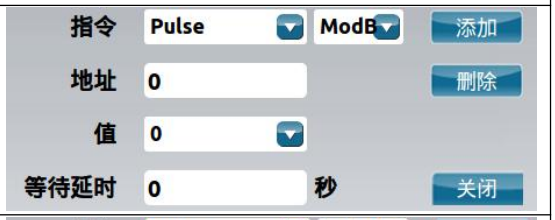
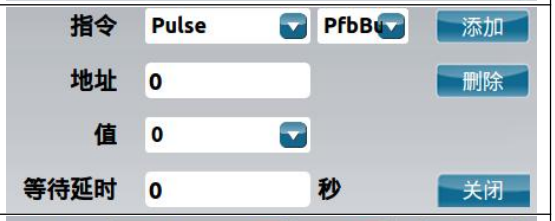
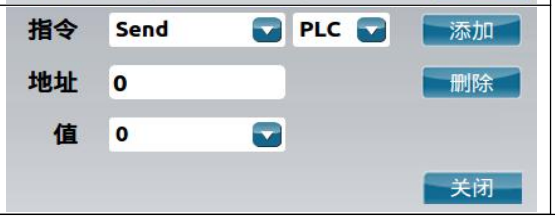
8) 点击指令类型下拉框，并选择指令“**Wait**”，出现子指令下拉框和地址下拉框，数值的下拉框,以及延时等待的输入框。

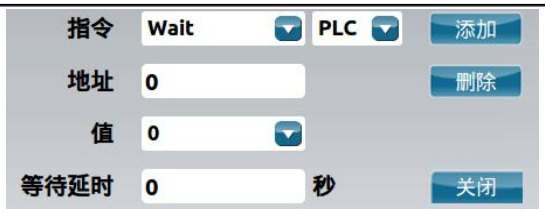
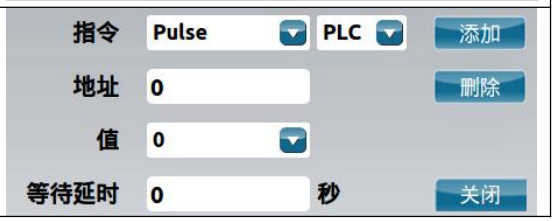
9) 点击指令类型下拉框，并选择指令“DelayTime”，出现数值输入框。

10) 点击“**关闭**”按钮，关闭当前页面，并返回冲压设置界面。

自定义指令列表及相应地址范围如下表：

信号类型	指令	地址范围	含义	实例
延时	Delaytime	-	等待延时	
硬件 IO	SendDo	1-已有 IO 模块	Do 输出	
	WaitDi	1-已有 IO 模块	等待 Di 输入	
	PulseDo	1-已有 IO 模块	脉冲 Do 输出	

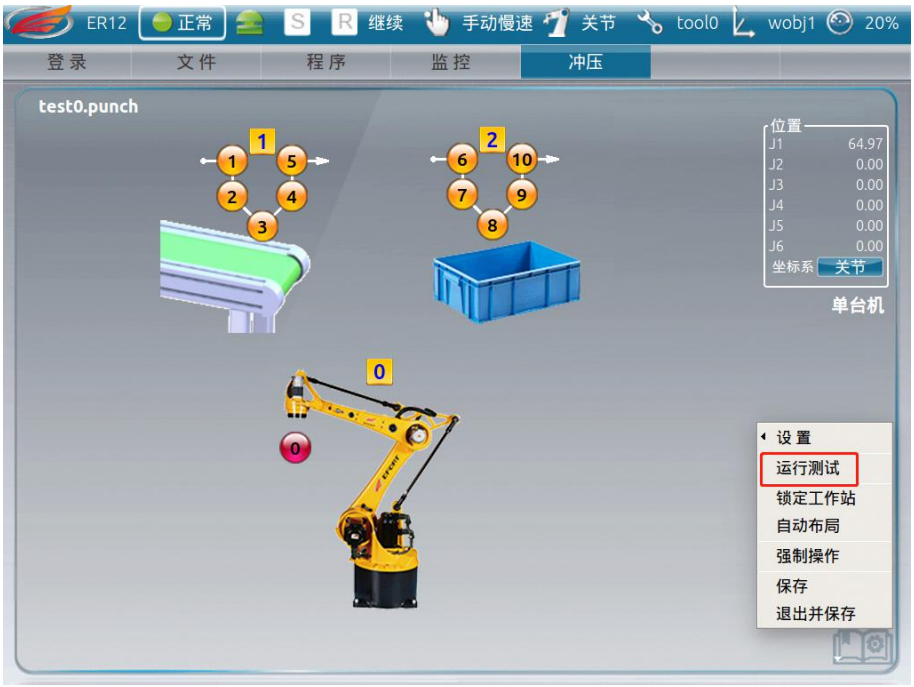
	总线（bool）	SendModBus	0-64	总线输出	
		SendPfbbus	0-64	总线输出	
		WaitModBus	0-64	等待总线输入	
		WaitPfbbus	0-64	等待总线输入	
		PulseModBus	0-64	脉冲总线输出	
		PulsePfbBus	0-64	脉冲总线输出	
	软 PLC（bool）	SendPLC	0-19	PLC 输出	

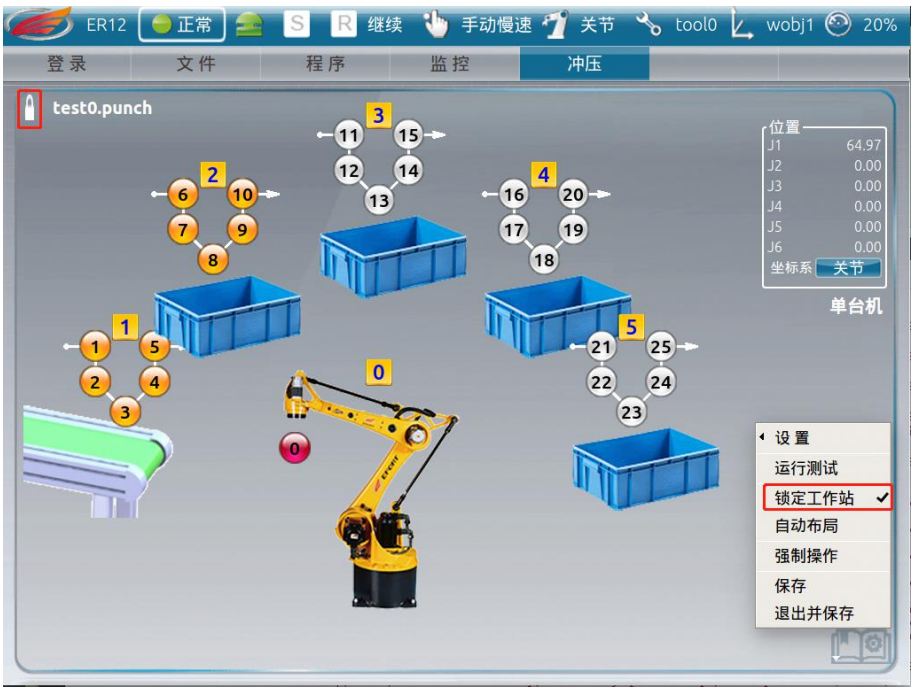
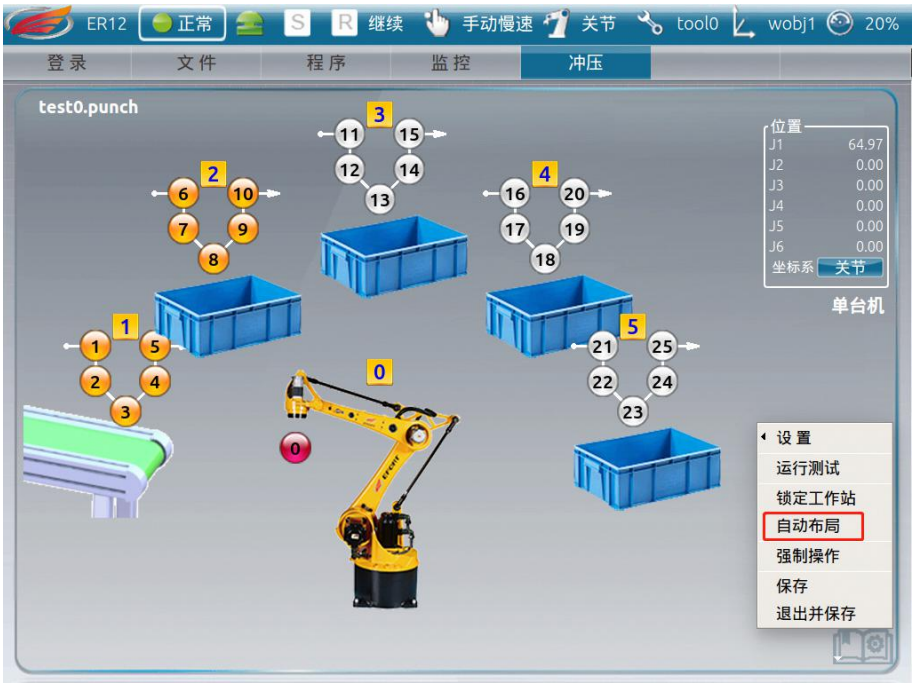
	WaitPLC	0-19	等待PLC输入	
	PulsePLC	0-19	脉冲PLC输出	

点击“**关闭**”按钮，关闭当前页面，并返回冲压设置界面。

9.运行测试，在“**手动低速**”“**手动全速**”模式下按下手压开关，或者“**自动模式**”下按下“**Mot**”键，确保伺服状态为S，此时点击菜单

中的“**运行测试**”。机器人将运行所有点（不包括设置的动作），只运行一次。在运行过程中，状态栏中程序运行状态为P。在运行过程中，当前运行至某一点，该点会有红色标记。



10.锁定工作站，点击菜单  中的“**锁定工作站**”命令，锁定工作站中的各机器人位置。	
11.自动布局，点击菜单  中的“**自动布局**”命令，令工作站中各机器人位置自动排列，恢复到出厂设置的排列位置。	

12.强制操作，点击菜单  中的“**强制操作**”命令，弹出强制操作操作复选框。点击“**工具 1 关闭**”（Tool1 close），强制关闭工具 1；点击“**工具 1 打开**”（Tool1 open），强制打开工具 1。

13.添加点，每个工作站内的机器人路径都已经默认配置了 5 个点或 9 个点。如果需要添加在工作站之间的过渡点，那么点击工作站中的黄色圆形标号  （已设置好位置），或灰色圆形标号  （未设置位置），弹出添加点的操作框,如图所示。

1) 点击“**往前添加**”按钮，顺着轨迹箭头反方向的后段添加一个点。

2) 点击“**往后添加**”按钮，顺着轨迹箭头方向的前段添加一个点。

3) 点击空白处，或其他控件时，该操作框消失。

14.设置点，点击路径上的某一点，弹出“**点属性**”界面，如图所示。

1) 点击“**直线运动**”（“Linear Move”）或者“**关节运动**”（“Joint Move”）的单选框，位置区域的参数变为“XYZABC”或者“J1-J6”。

2) 点击转弯半径，输入半径参数，更改机器人运动时转弯半径。

3) 勾选“**碰撞检测**”组框，将自动生成碰撞检测指令。true/false 下拉框，打开或关闭程序碰撞检测功能，数值 100 为程序碰撞检测系数，推荐值 100，可修改。

4) 点击动作区域的“**编辑**”下拉框，选择列表中“**动作库**”（“Action Library”）中的动作，选择其中一项，点击，将动作添加到动作列表（“Action List”）中，点击空白处，完成添加动作操作。

注意：对于 open tool 和 close tool 两个动作，必须成对使用。若只使用其中一个指令，将会影响机器人工具下次执行动作。例如：点 1 动作 open tool，点 2 动作 close tool。

5) 拖动“**速度**”滑块，更改当前点的速度，数值也会随着拖动而变化。

6) 点击“**示教**”按钮，将机器人当前的位置记录下来（记录位置的形式根据运动类型），若该点未示教，示教后则点由灰色变成黄色。

7) 点击“**复制**”下拉框，可以选择之前的点位置作为该点的位置（该点必须被示教过）。

8) 点击“**删除**”按钮（该点时通过添加点操作而来的时候，该按钮才使能），该点从当前单元的路径上被删除。

9) 点击“”按钮，根据站点次序，递退到后一点属性供用户去设置信息，同理点击“”按钮，根据站点次序，递进到前一点属性供用户去设置信息。

10) 点击“**单步**”按钮（该点必须已经示教，否则将处于禁止状态），在“**手动低速**”“**手动全速**”模式下按下手压开关，或者“**自动模式**”下按下“**Mot**”键，确保伺服状态为，机器人将运行到示教位置。

15.偏移，点击路径某一点，弹出“**点属性**”界面，见步骤 16，点击“**偏移示教**”，进入基于参考点的示教，如图所示。

在“**点偏移**”界面中，参考下拉菜单中选择需要参考的点。在“**点属性**”下部左侧参考点位置中，显示的是所选中的参考点的位置，右侧“**偏移距离**”可以设置偏移距离或角度。

注意：如果在参考点的点属性设置中选择的是“运动类型”中的“关节运动”，则在偏移距离中偏移的是角度值，所修改的是基于参考点关节的角度转动值；如果在参考点的点属性设置中选择的是“运动类型”中的“直线运动”，则在偏移距离中偏移的是距离值以及姿态值，即修改的是基于原参考点的位置与姿态。

注意：

1) 在基准点位置修改之后，偏移点位置也随之改变。如图所示，其中偏移点 4 位置随着参考点 1 的位置变动而变动。

a 参考点“1”未修改时基于参考点 1 的偏移点 4 的位置

b 参考点“1”修改后基于参考点 1 的偏移点“4”的位置

2) **在示教完偏移点之后，若在参考点的点属性设置中，修改“运动类型”，如将“直线运动”换成“关节运动”则所有基于该参考点的偏移点都将失去偏移关系，需重新进行示教。**

16.关闭，点击菜单中的“**退出并保存**”，弹出提示“**保存改变并传输文件？**”消息框，如图所示。

1) 若点击“**是**”，则保存更改,并弹出消息框“**请将机器人移动至安全区域再进行生产！**”，再点击“是”可退出当前页面。若有点未示教，则在退出之前，会弹出相应的警告消息框，点击“是”不可以退出当前界面。

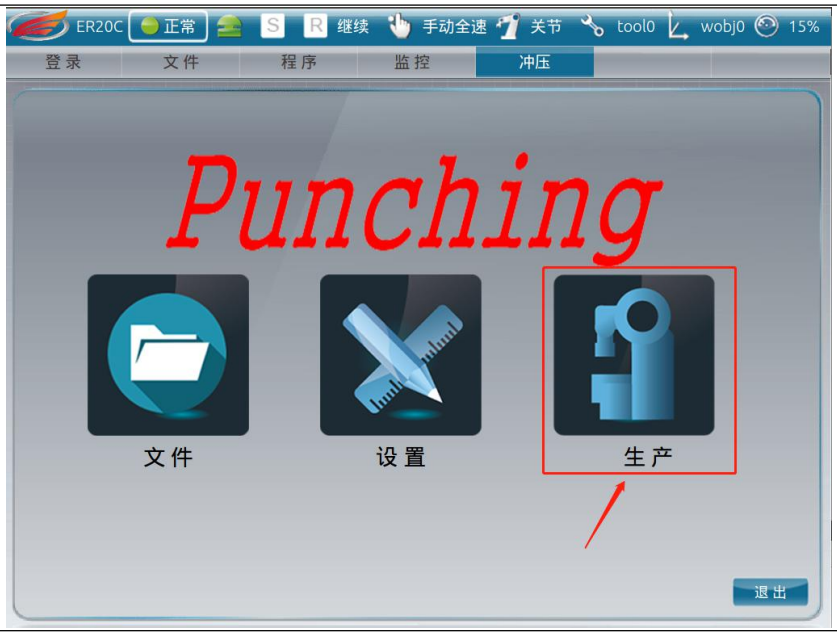
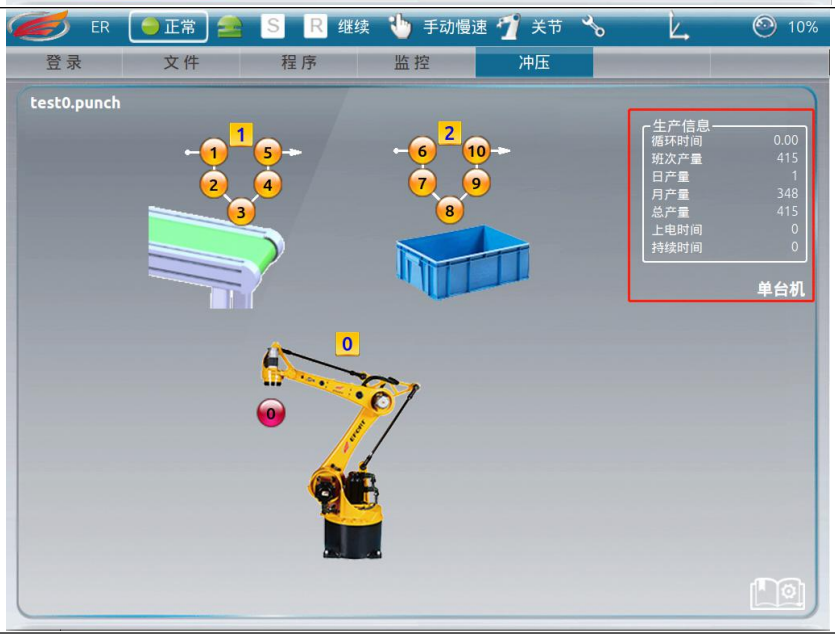
2) 若点击“**否**”，则不保存更改，退出当前界面，返回冲压功能初始界面。若有点未示教，则在退出之前，会弹出相应的警告消息框，点击“是”依然可以退出当前界面。

3) 若点击“**取消**”，消息框消失，无任何其他事件发生。

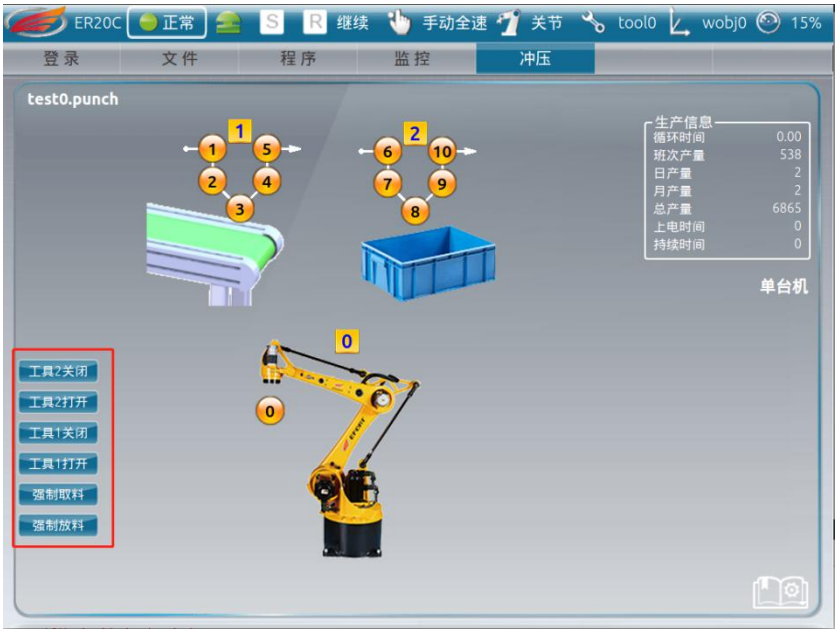
19.3 生产界面

生产界面主要用做生产时的信息实时监控，包括生产信息显示、从头执行、预约次数、重置、导出数据、强制操作等。

表 18-2 冲压生产操作步骤

步骤	图示
1. 在关闭设置界面后，返回冲压功能界面，再点击“**生产**”图标，进入生产界面。	
2.生产监控，右侧的生产信息中，上电时间表示抱闸打开的持续时间，一旦抱闸关闭该时间清零；持续运行时间，机器人持续运行此程序的时间，一旦机器人停止运行程序，该时间也会清零。 界面中的点以及工作单元序号图标均不可编辑。	

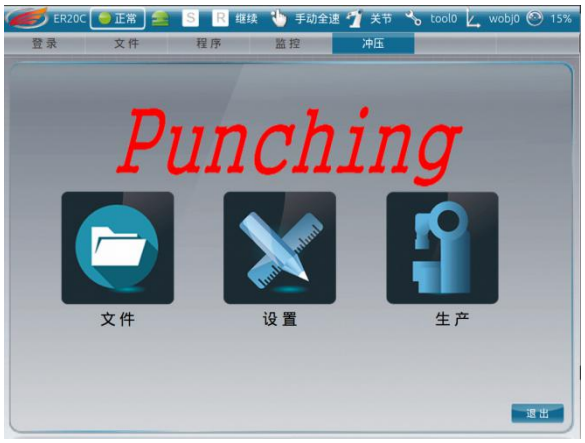
3.重置日产量，点击菜单中的“**重置**” 命令下的“**重置日产量**”，弹出“**确定重置当日产量吗？**”的消息框，如图所示。点击“**是**”，则在生产界面中显示的日产量数值将被清零；点击“**否**”，消息框消失，日产量数值将不被清零。	
4.重置月产量，点击菜单中的“**重置**”命令下的“**重置月产量**”，弹出“**确定重置当月产量吗？**”的消息框，如图所示。点击“**是**”，则在生产界面中显示的月产量数值将被清零；点击“**否**”，消息框消失，月产量数值将不被清零。	

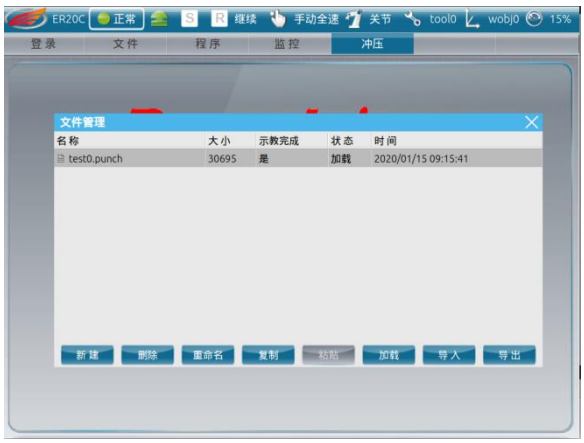
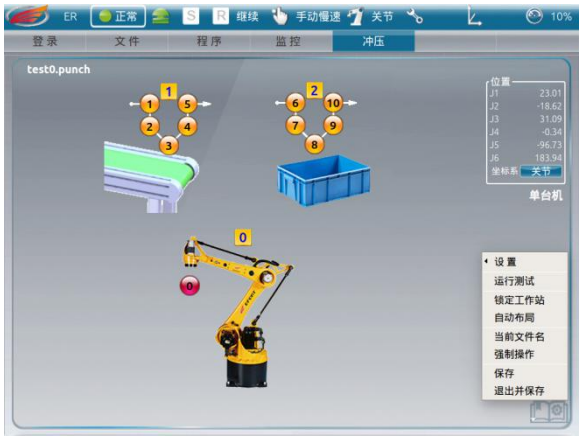
5.重置总产量，点击菜单中的“**重置**”命令下的“**重置总产量**”，弹出“**确定重置总产量吗？**”的消息框，如图所示。点击“**是**”,则在生产界面中显示的总产量数值将被清零；点击“**否**”，消息框消失，总产量数值将不被清零。	
6.强制操作，点击菜单中的“**强制操作**”命令，弹出强制操作操作复选框。点击“**工具 1 关闭**”（Tool1 close），强制关闭工具 1；点击“**工具 1 打开**”（Tool1 open），强制打开工具 1；点击“**强制取料**”，强制机器人从上台冲压机或传送带上取料；点击“**强制放料**”，强制机器人将物料放入放料框或下台冲压机中。	

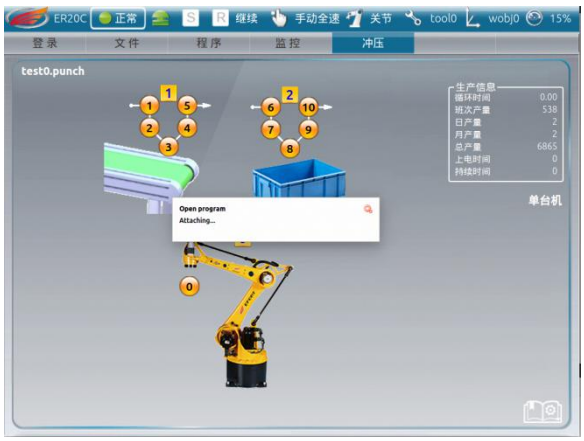
7.运行生产，点击“**Start**”键，开始运行 punch 程序。 若按下时，没有将钥匙开关拨至自动模式，且未按下“**Mot**”键将伺服状态未变为 S，会出现以下消息框，如图所示。若当前已是自动模式，且已按下“**Mot**”键将伺服状态未变为 S，此时再按下“**Start**”键，上面的消息框消失，且冲压程序开始运行。 点击消息框中的“**确定**”,消息框也会消失。	

19.4 冲压软件开机操作

表 18-3 冲压开机操作步骤

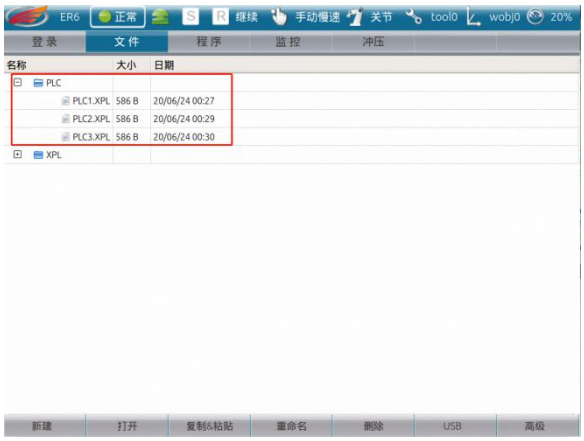
步骤	图示	说明
1.先进入桌面，点击“**冲压**”图标,进入冲压功能界面，如图所示。		

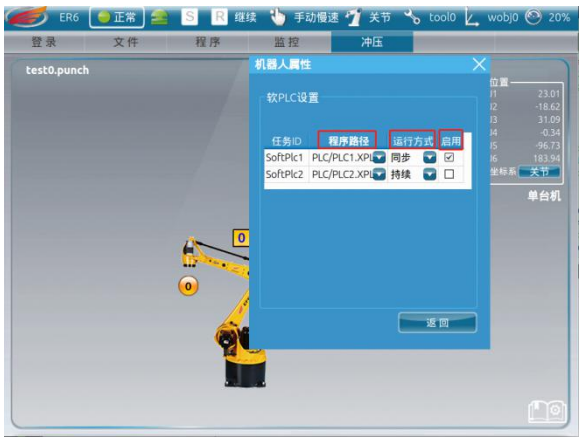
2.点击“**文件**”图标，进入文件管理界面，选中需要运行文件，点击“**加载**”（“Load”）按钮，然后点击右上角叉号退出“**文件管理**”。		1) 新建文件 点击“**文件**”图标，进入文件管理界面，点击下部按钮“**新建**”，新建文件，输入文件名，点击“Yes”，完成文件创建操作。 2) 删除文件 点击“**文件**”图标，进入文件管理界面，点击下部按钮“**删除**”，弹出“**确定删除文件吗**”对话框，点击“**是**”，完成删除操作。 3) 导入文件 插入 U 盘，点击下部按钮“**导入**”，弹出文件对话框，如下图所示，选择需要导入文件，点击“**打开**”，弹出确认对话框，点击“**是**”，完成导入操作。 4) 导出操作 插入 U 盘，选中文件，点击下部按钮“**导出**”，弹出“**确认要导出选中文件吗**”对话框，点击“**是**”，完成导出操作。
3.点击“**设置**”图标，进入步骤 2 所选定冲压文件的设置界面。然后点击界面右下角菜单按钮，选择最后一项退出并保存设置，如图所示，选择保存设置，再确定进入机器人安全区域。		

4.最后进入生产界面，如图所示，按 18.3 章节操作步骤内容运行生产操作。		

19.5 在机器人站属性中设置软 PLC

表 18-4 冲压中软 PLC 设置操作步骤

步骤	图示	说明
1.在任务栏的文件中新建 PLC 文件夹，在 PLC 文件内新建软 PLC 程序，可以新建多个。		新建的文件夹名必须为“PLC”，具体软 PLC 程序名可以自己定义。
2.进入机器人站属性，点击“软 PLC 设置”		

3.进入软 PLC 设置界面，进行 PLC 设置。		1. 选择软 PLC 的存放路径。 2. 选择运行方式：“同步”为 plc 程序与主程序运行同步，主程序运行，plc 程序运行，主程序暂停，plc 程序暂停；“持续”为只要在主程序运行生成的 plc 指令后，plc 程序会一直运行，不受主程序运行状态的影响。 3. 启用：☑ 表示启用该任务 ID，在自动生成程序过程中生成 plc 指令；☐ 表示不启用该任务 ID，在自动生成程序过程中不生成 plc 指令。

19.6 多机连线

在冲压客户现场，大多数工艺场合需要多台压机、机器人、输送设备等组成生产线，为了解决现场人员 PLC 编程困难问题，此功能包增加了多机连线设置运行功能，无需 PLC 编程，只需客户按以下内容设置连线即可。

19.6.1 设置

19.6.1.1 机器人工作模式设置

点击设置，选择设置单元进入机器人单机、多机选择，如图 18-1 所示

图 18-1 设置单元

19.6.1.1.1 机器人工作模式首台机动作设置

在设置界面，点击界面下方机器人站的“**0**”按键，进入机器人属性设置界面，在“**设备类型**”下拉菜单中选择“**首台机**”（First Machine）命令，点击空白处返回，机器人工作模式将变成首台机状态。如图 18-2 所示。

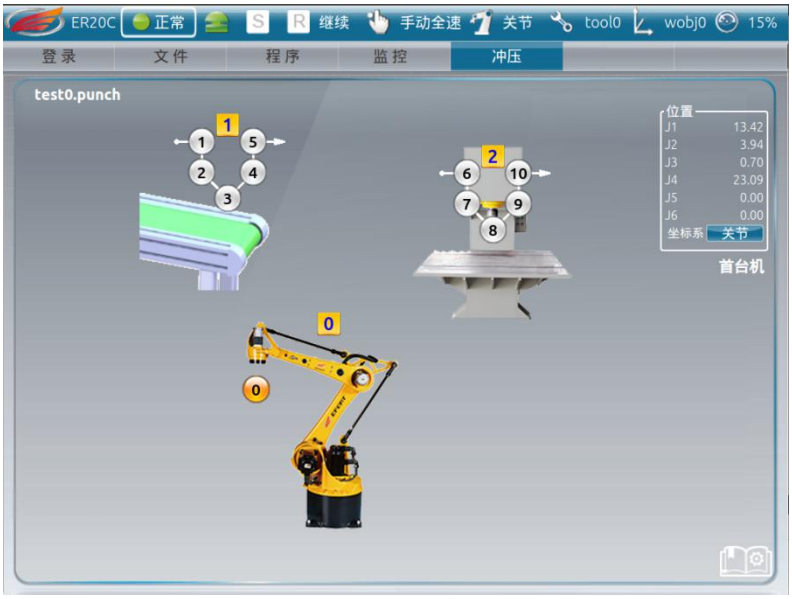


图 18-2 首台机界面

进入设置动作界面，可以查看预定义动作，在机器人工作类型不同情况下，其执行的逻辑不同。

图 18-3 预定义动作查看

1) 关闭工具：表示首台机强制关闭气嘴或其他末端工具。

2) 检测工具：表示首台机检测工具工作状态。

3) 吹起：表示首台机末端工具进行吹气操作。

4) 等待取料：表示首台机等待取料信号。

5) 等待放料：表示首台机等待放料信号。

6) 取料结束：表示首台机取料结束。

7) 放料结束：表示首台机放料结束，发送压机启动信号，并检测压机工作状态，压机工作完成后，发送信号给下一台机器人。

注意：这里的取料延时设置对检测动作有效，放料延时设置对关闭工具动作有效。

例如：设置点 3 动作，打开工具 1 并延时，可对机器人和点 3 做如下设置

图 18-4 点 3 设置

19.6.1.1.2 机器人工作模式中间机设置

在设置界面，点击界面下方机器人站的“**0**”按键，进入机器人属性设置界面，在“**设备类型**”下拉菜单中选择“**中间机**”（Mid Machine）命令，点击空白处返回，机器人工作模式将变成中间机状态，如图 18-5 所示。

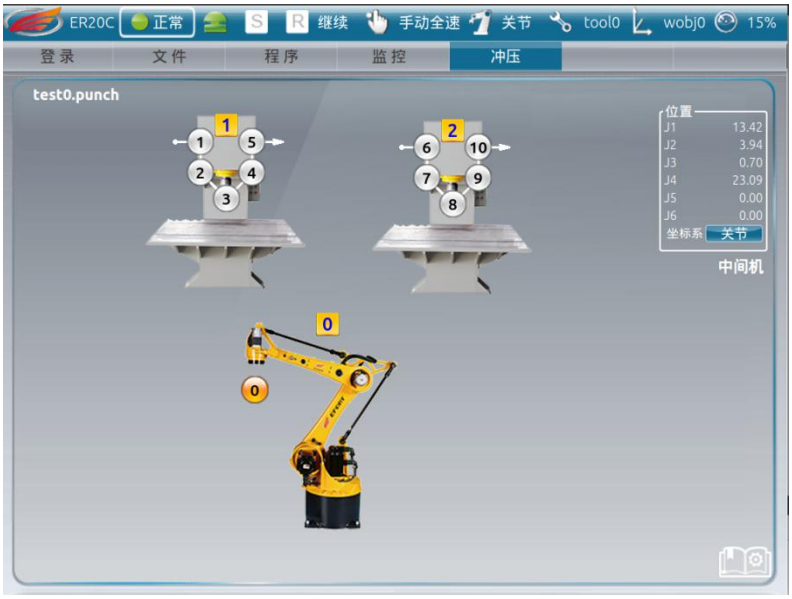


图 18-5 中间机界面

进入设置动作界面，动作设置名称有四个固定动作名称，分别：等到取料，取料结束，等待放料，放料结束。在机器人工作类型不同情况下，其下发指令内容都不同，中间机需下发的指令内容（四条指令）如下：

1) 等待取料：表示中间机等待取料信号。

2) 取料结束：表示中间机取料结束，发送信号给上一台机器人。

3) 等待放料：表示中间机等待放料信号。

4) 放料结束：表示中间机放料结束，发送压机启动信号，并检测压机工作状态，压机工作完成后，发送信号给下一台机器人。

19.6.1.1.3 机器人工作模式末端机设置

如果机器人工作模式选择为末端机，保存后返回设置主界面，机器人工作类型改变为末端机如图 18-6 所示。

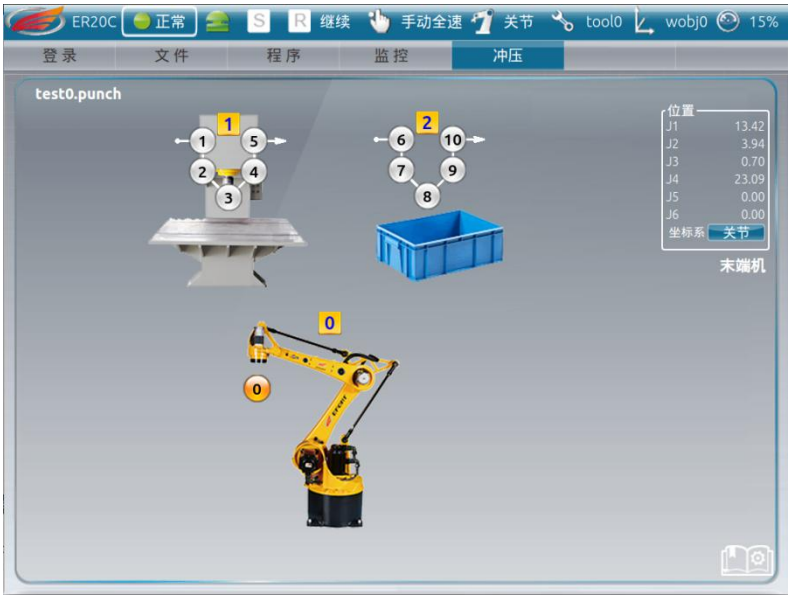


图 18-6 末端机界面

进入设置动作界面，动作设置名称有四个固定动作名称，分别：等待取料，等待放料，取料结束，放料结束。在机器人工作类型不同情况下，其下发指令内容都不同。

1) 等待取料：表示末端机等待取料信号。

2) 等待放料：表示末端机等待发送放料信号。

3) 放料结束：表示末端机取料结束，发送信号给上一台机器人。

19.6.1.2 机器人模式取料工具设置

图 18-7 取料工具设置

如图 18-9 所示，进入设置界面后，在“**单台机**”下点击机器人站的“**0**”按键可以看到机器人属性，可以在“**工具 1 类型**”和“**工具 2 类型**”右侧的下拉菜单里选择所要使用的工具。

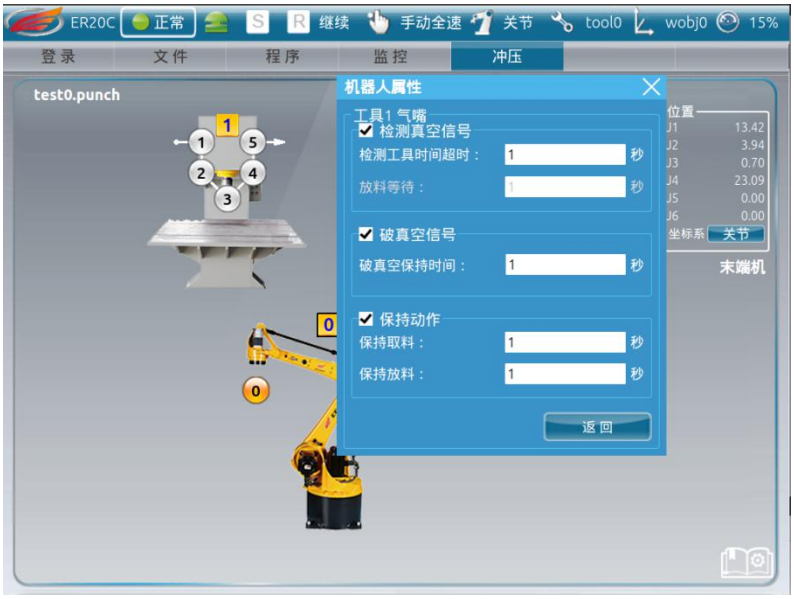


图 18-8 工具属性设置

如图 18-10 所示，这里以工具为“**气嘴**”为例，点击“**工具 1 设置**”可以对“**气嘴**”的真空压力信号、破真空信号、延时进行设置。

1) 真空压力信号：机器人在取料的时候会等待真空压力信号，如果这个压力信号超过设置时间没有收到，则触发工具报警。同理，在放料时，该信号应在设置时间内消失。

2) 破真空信号：放料时，破真空。

3) 延时设置：取料完成或放料完成后，延时设置时间后再运动。

19.6.1.3 机器人模式是否启动冲压设置

图 18-9 冲压启动设置

这里以末端机为例，点击“**1**”按键进入图 18-11，在“**设置**”图标的左边可以通过看是否打钩来判断是否启动冲压设置。

19.6.1.4 机器人压机设置

图 18-10 压机设置

在如图 18-12 所示，可以对压机“**冲压脉冲**”、“**压机单次模式**”等参数操作进行设置。

常见错误操作：

如果此时将冲压脉冲设置 1s，最小间隔设置 2s，最大间隔设置 5s，点击退出并保存时会出现如下图提示框，提示“冲压机的时间参数错误。冲压时间需介于最小冲压时间与最大冲压时间之间。请检查”。

图 18-11 弹框信息

原因在于压机动作设置为 1s，机器人通过检测压机信号，发现压机动作时间未在允许范围之内，产生提示。

解决方案：将冲压脉冲时间设置在最小与最大允许范围之内。

19.6.2 多机连线基本操作

上伺服，点击示教器上的“Start”按钮，运行程序。

若处于测试阶段则点击 force get，通过点 1：当抓取工具为磁吸和气嘴；点 2：工具打开动作；点 3：检测工具打开。

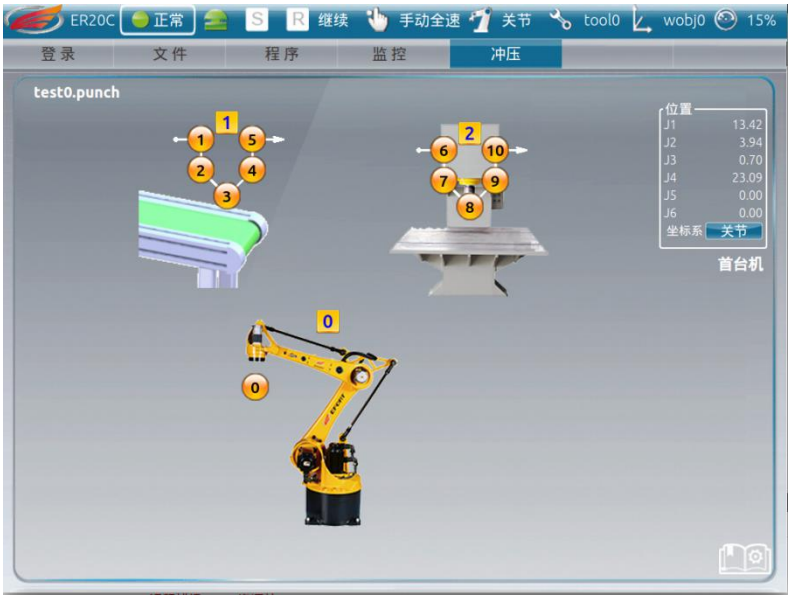


图 18-12 冲压主界面

具体设置如图 18-15 所示：

图 18-13 工具抓取操作设置

根据工具 1 设置的属性，机器人运动到点 3 后，工具 1 延时设定取料延时 4 秒，再检测真空信号，若 3 秒内没接收到信号，则会出现如图 18-16 报错。

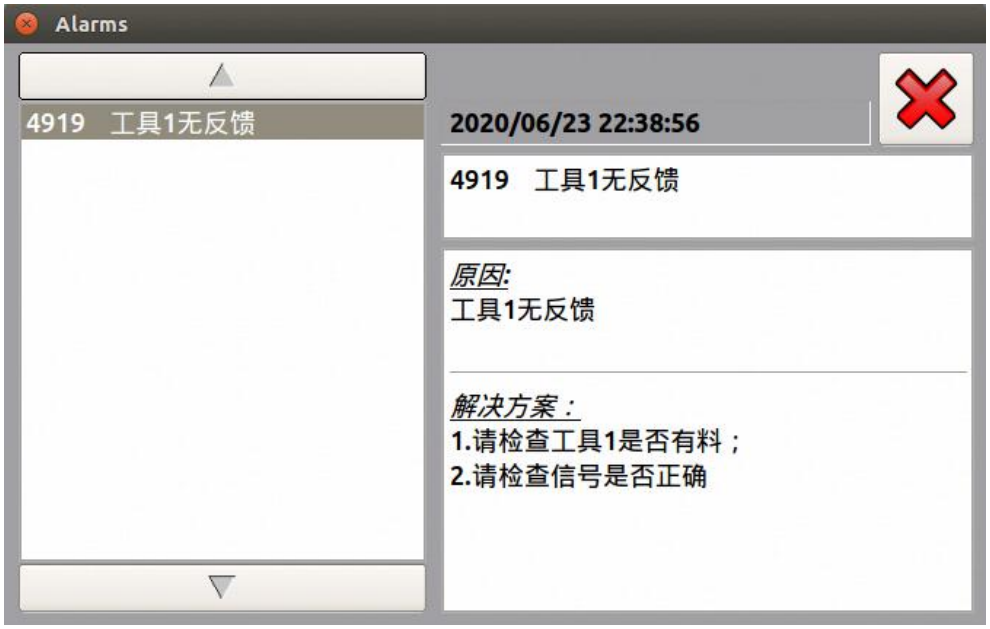


图 18-14 报警信息

清除报警，通过强制真空信号关联的工具 1 打开和工具 2 打开高电平,点击示教器"Start"按钮继续运行。工具 1 延时 4 秒后，因为工具 1 打开已强制，即完成检测工具打开动作。

运行点 4、5 至点 6，检测压机信号，若测试阶段则点击强制放料，强制机器人放物料。

运行至点 7-8,在第 8 点时，放下物料，继续运行至 9-10 点。

在 8 点时，若出现图 18-14 报警。在清除报警后，通过强制真空信号关联的信号为低电平,点击示教器"Start"按钮继续运行。工具 1 延时 6 秒后，因为真空信号已强制，然后破真空 5 秒，即完成检测工具打开动作。

注意：

当机器人运行至 8 点过程中，若压机发生突然落下、压机未及时返回至最高点等压机未在正常上停滞点情况或手动强制冲床动作信号时，会出现如图 18-17 报警“冲压机上死点丢失”。

图 18-15 报警信息

当抓取工具为气爪时，点击 Force get，通过 1-2 点，至点 3 时，气爪打开，抓取物料，继续运行 4-5，至点 6-7-8-9-10，同上。

19.7 码垛

在冲压完成后，本控制系统解决市面上已有工业机器人的界面设计缺陷，将码垛与冲压放在一起，用户可以在完成冲压后，在操作界面上选择将物料放入物料框中或者将物料按一定顺序堆叠起来。

19.7.1 设置界面

如 18 章冲压的前五节的内容，在文件中选择需要加载的程序，点击“**加载**”，进入“**设置**”界面。点击“**菜单**”中的“**设置**”—“**设置单元**”命令，如图 18-16 所示，点击“Pallet”，选择向右箭头，将“**Pallet**”放入右边方框中，退出。如图 18-17 所示，为码垛设置界面。从 0 点到 5 点，对机器人从原点到冲压上下料过程进行轨迹规划，步骤同 18 章冲压的前五节的内容。点击码垛上方“**2**”按键，对码垛进行相关参数设置。

图 18-16 加载码垛单元

图 18-17 码垛示意图

19.7.2 码垛参数设置

如图 18-18 所示，为码垛参数相关设置。

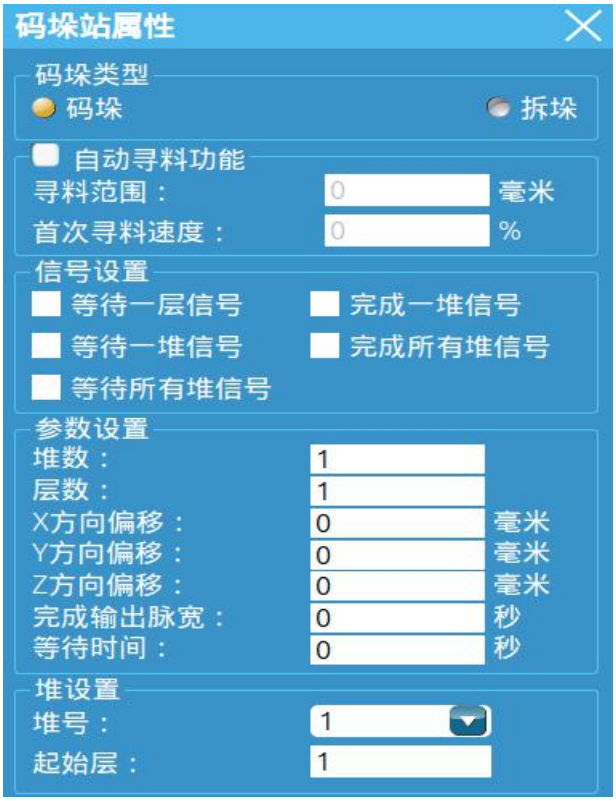


图 18-18 码垛参数设置

堆数：对需要的堆数进行设置，如果需要码成三 垛，则设置为 3。

层数：每个码垛堆所有的层数，如果需要每堆码成 3 层，则设置为 3。

X 方向偏移：每码一层后，下一次码垛在 x 方向较上次偏移的数值。 （注意：单位是毫米）

Y 方向偏移：每码一层后，下一次码垛在 y 方向较上次偏移的数值。

Z 方向偏移：每码一层后，下一次码垛在 z 方向较上次偏移的数值，一般设置为物料的高度，以避免机器人末端抓手碰到所码的垛。

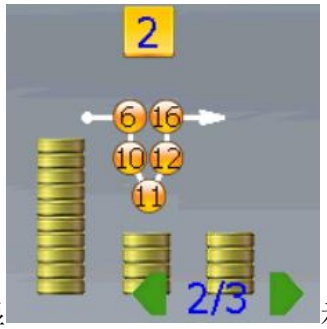
码垛完成输出脉宽：在码垛动作完成后，向外输出高电平，表示已完成动作。

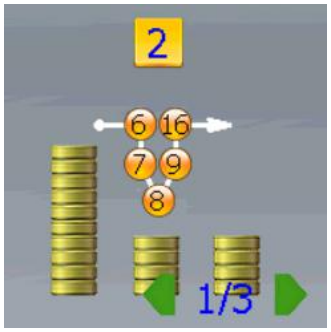
堆号：对当前堆号进行设置，如果堆号选择 2，则代表对第二个码垛堆进行设置。（注意：该堆号设置只是针对于下面的起始层而言，对上方的“参数设置”没有作用）

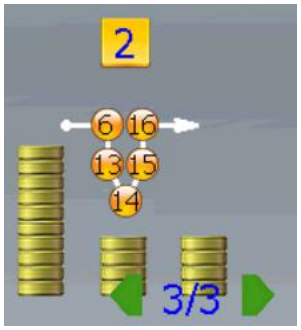
起始层：从第几层开始码垛。如果设置为 2，则是从第二层开始码，第一层跳过。

19.7.2.1 码垛点的示教

在“设置”界面，如果在参数设置中设置为 3 堆物料则在码垛图片下方界面中会显示 1/3 图样，表示需要码 3 堆，当前是第一堆。如图 18-19 显示，在第一堆显示图片上显示 6-7-8-9-16，在第二堆显示图片上显示 6-10-11-12-16，在第三堆显示图片上显示 6-13-14-15-16。







a b c

图 18-19 码垛点示意图

其中 6 点为放料等待点；7 点为放料上方点，表示进入码垛区域，区别于 8 点放料点的另外一个点。10、13 同 7 点；8 点为放料点，表示准备放料的点，为实际放料的点位置。11、14 同 8 点；9 点为放料上方点，12、15 同 9 点；16 点为放料结束点。

拆 垛原理同码垛原理，不同点主要集中于：码垛是将物料码上去，而拆 垛是将物料从码垛堆中搬下来。故在执行拆 垛指令时，机器人在搬完一层后，向下偏移，搬运下一层的物料。

19.7.3 码垛运动路径

在设置好相关参数后，对每一堆的路径进行示教（方法同冲压点的示教），退出设置界面，进入生产界面。将开关转到自动模式，按下“**Mot**”键，点击“**start**”，机器人开始运动。

机器人将物料从压机上拿取，默认从第一层开始码垛（如果希望可以跳过第一层，则可以在“**设**

置”中“**堆设置**”中对“**起始层**”进行设置），每完成一层的码垛，下一次码垛将在 z 方向自动向上加 z 轴偏移量（注：若在“**设置**”界面的码垛站属性中对 X、Y 偏移量进行参数调整，则在运行过程中如同 z 方向偏移一样，在完成一层码垛后，下次码垛时，会在所调整的 X、Y 方向上自动产生偏移量。）。通俗来说，若设置 x=3，y=2，z=1，则码垛时每完成一次动作，执行下一次动作时，机器人抓手会自动在 x 方向偏移 3 个单位、在 y 方向偏移 2 个单位、在 z 方向即垂直于码垛平面的方向偏移 1 个单位，在完成第一堆的码垛后，转到第二堆，在码垛图标下部会显示 2/3 表示已进入第二堆的码垛程序，开始码垛，以此类推，完成整个码垛操作。

19.8 拆垛

拆垛与码垛相反，是将垛盘上的一垛垛料（每层只能有一个工件）拆放到冲床或平台上。该功能包含两种工作模式：其一、按照垛盘和垛料设置堆数、层数和层高；其二、在拆垛时通过设置堆数和层高，无需设置层数，使用自动寻料功能来进行拆垛。

19.8.1 设置固定层数拆垛

19.8.1.1 设置界面

新建文件，然后加载该文件，进入设置界面，然后点击“设置”->“设置单元”，添加“Pallet”（码垛站），及其它需要的站，点击“退出”，如图所示：

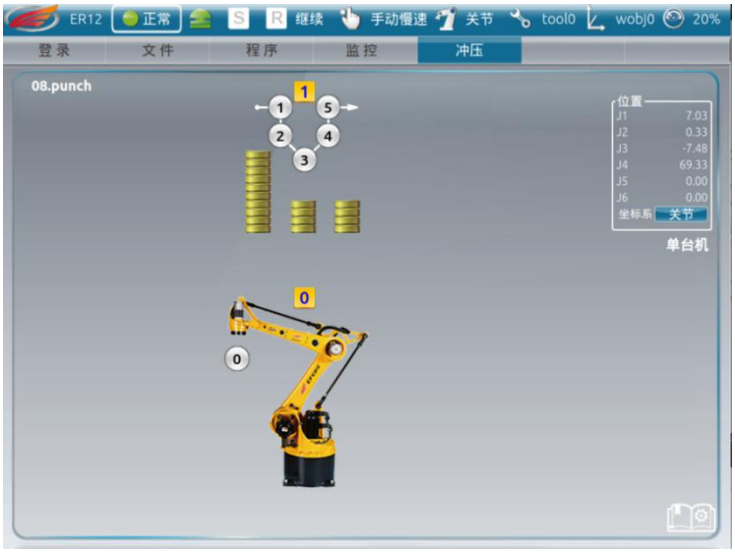


图 18-20 设置界面

19.8.1.2 参数设置

点击站属性按钮，弹出参数设置界面，如图所示：

图 18-21 码垛站属性设置

设置红框内的相关参数：

一、码垛类型选择拆垛；

二、信号设置：分为输入信号和输出信号；

输入信号：

1.等待一堆信号，即判断每一堆的输入信号；

2.等待所有堆信号，即开始拆垛时只判断一次堆信号，；

（注意：等待信号只能选一种）

输出信号：

1.完成一堆信号，即每拆完一堆时输出一个信号；

2.完成所有堆信号：即拆完所有堆后输出一个信号；

（注意：完成信号只能选一种）

三、参数设置

其值设置与码垛参数类似，具体参数含义见码垛参数介绍。

四、堆设置

值设置与码垛参数类似，具体参数含义见码垛参数介绍。

19.8.1.3 示教点

该示教过程与码垛相似，可参照码垛点的示教过程。

19.8.1.4 添加预定义动作

与其相关的预定义动作有：“等待拆垛信号”和“拆垛完成信号”；其对应的 IO 信号地址可以在 IO 设置的冲压中进行自由配置。将“等待拆垛信号”预定义动作添加在拆垛进入点，将“拆垛完成信号”添加在拆垛退出点。

19.8.1.5 生成程序

所有的设置在设置界面完成后，点击保存，然后进入生产界面，生成程序，并无报错后，即可进行调试生产。

19.8.2 自动寻料功能拆垛

19.8.2.1 设置界面

点击相应站序号按钮，弹出站属性参数设置界面，如下图所示：

图 18-22 拆垛自动寻料参数设置界面

设置相应红框内的参数：

一、自动寻料功能参数设置

1.自动寻料勾选框：当勾选时启用自动寻料功能；

2.寻料范围：参数值为正时，机器人沿 Z 轴正向寻料；参数值为负时，相反；

3.首次寻料速度：其值设置范围为 0~100%，根据实际需要设置适当的值。

二、参数设置

只需设置堆数（注意：在拆垛时需判断每堆是否有料时，目前只能设置 1~3 堆料，如需增加堆数，则请说明）及 Z 方向偏移值（一般为物料厚度）。

19.8.2.2 示教点

码垛站的第一点和最后一点为拆垛进入点和退出点，第二点和倒数第二点为拆垛上方点（寻料起始点）和拆垛完成上方点，中间点为拆垛抓取点。如需在拆垛过程中增加过渡点，则需在第一点和第二点之间增加，第二点和中间点不能增加过渡点。以设置 2 堆为例进行说明，在拆垛进入点示教拆垛第 1 点，一般拆垛进入点与退出点是同一个点，则第 8 点可以设置为偏移与第 1 点，如果第 1、8 点位置不同，则根据需要示教相应的点；下一步示教 2、4 点，为拆垛上方点，根据物料高度，示教合适的高度即可；下一步示教中间点，由于首次寻料时物料点的位置未知，由控制器中获取寻料点位置，则中间点（当前为第 3 点）设置为偏移于第 2 点即可。此外，在示教完点后设置每点的速度和转弯半径参数。第一堆完成后，点击下一堆按钮，进行相关点的示教，过程与第一堆示教过

程类似。

19.8.2.3 添加预定义动作

与自动寻料功能相关的预定义动作只有“等待拆垛寻料”;与其相关的 IO 信号有 5 个输入（Input）IO 信号：

1. 垛盘 1 有料；2.垛盘 2 有料；3.寻料确认；4.寻料有料；5.工具有件或真空检测，具体地址可以在 IO 设置的冲压中进行自由配置。具体 IO 信号状态可通过监控 IO 中查看。

将“等待拆垛寻料”预定义动作添加在拆垛进入点。

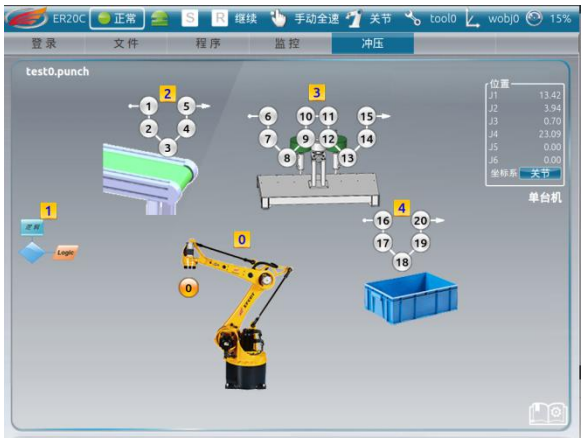
19.8.2.4 生成程序

所有的设置在设置界面完成后，点击保存，然后进入生产界面，生成程序，并无报错后，即可进行调试生产。

19.9 传送带-打磨/抛光工艺

此功能主要为客户解决打磨/抛光工艺。根据工艺要求，本例中工具在各个关键点的动作顺序为：点 1 等待取料，点 3 工具 1 取料，点 5 工具 1 取料完成；逻辑判断：进入工作站 3 或 4，执行取料、放料动作；逻辑动作完成，进入工作站 5，工件放入物料框。机器人末端设置多个工具，在相应位置执行对应动作，具体操作流程如下：

表 18-5 冲压打磨/抛光工艺操作步骤

步骤	图示	说明
1.添加工作站并示教关键点。		添加 robot（机器人）、logic（逻辑控制）、conveyor（传送带）、polisher（打磨/抛光机）、output（物料框）等工作站单元，如图所示。刚添加的工作站，由于关键点没有示教，因此关键代表的颜色为灰色。示教每个工作站中机器人末端位置，使得个关键点的颜色为黄色。

2.设置工作站属性。每个工作站上方方框内的数字  表示工作站在该工程中的顺序。点击工作站数字，即可进入工作站属性设置。

a.点击机器人（robot）上方的数字 0，进入机器人属性设置，如图所示。由于工具在后续已自定义形式设置，这里工具类型选择 None,设备类型选择单台机。

b.点击传送带（conveyor）上方的数字 1，进入传送带属性设置，如图所示。

c.点击逻辑站（logic）上方的数字 2，进入逻辑设置，如图所示。逻辑站主要采用 IF——ELSE——ENDIF 结构。将内部/外部输入信号作为判断条件，机器人根据逻辑对多个工作站工作。

d. 点击打磨/抛光站（polisher）进入站属性设置，如图所示。None——屏蔽属性设置；shield polisher——屏蔽工作站；condition——工作站条件设置。选择 condition 选项，可分别激活 Wait（等待输入信号）和 IF（逻辑控制）。

注：逻辑站与抛光站均有逻辑判断，逻辑站包含抛光站，具体的结构如下：

IF （ condition 1 ） //IF_ELSE-END_IF 结构，由逻辑站控制

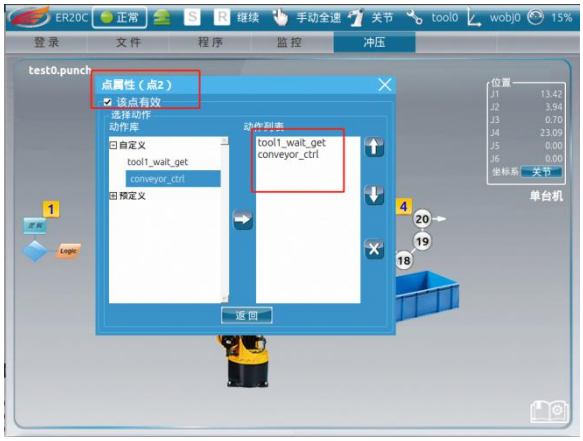
IF （ condition 2 ） //IF-END_IF 结构，由打磨/抛光站控制

执行 1;

END_IF

ELSE

		**IF（condition 3）** **执行 2;** **END_IF** **END_IF**
3.关键动作设置。关键动作设置。	 	根据工艺要求，机器人末端在传送带上方需要等待传送带信号，因此，在关键点 1 设置动作“传送带等待有料”（等待获取工件，具体地址可以在 IO 设置的冲压功能中进行地址配置），在关键点 5 设置动作“传送带取料结束”。

4.关键点工具动作自定义设置。	 	在冲压功能包里，点击“**设置-动作设置**”进入动作设置界面。这里包含了系统预先设定的动作，也可以自定义动作。如图所示，自定义动作 tool1_wait_get 包含的指令有信号输出、延时等。 对点 2 添加相应的动作，本例中添加的动作有 tool1_wait_get 和 conveyor_ctrl，分别表示工具 1 等待获取和传送带停止运动。 其余各关键点处需要根据具体工艺要求，添加系统预设动作或自定义动作，以完成全部工艺过程。 设置完以上步骤，经运行测试后确保各关键点示教正确并机器人动作无误，即可进入生产模式循环运行。 **注：I/O 口信号设置具有灵活性，需要根据现场情况做具体分配，本例中 I/O 分配不做具体规定。**

第 20 章　高级码垛

20.1 本章简介

本章主要介绍 EFORT 工业机器人高级码垛文件导入导出、高级码垛设置及生产操作。

机器人高级码垛是机器人根据特定的要求从输送线依序抓取产品完成垛位的产品单层摆放，而后依层完成产品码放作业，并能够完成不同垛位的码放数量的统计。本工艺包设计针对单输送线单工位和多输送线多工位类型作业，且码垛过程支持单抓单放和单抓多放。

机器人作为中间搬运设备，负责分别从产品输送线上抓取产品并从托盘库抓取托盘，然后进行按照特定的码垛模式进行堆垛作业，这样由机器人、托盘和产品输送线构成了机器人码垛单元。

高级码垛过程是指机器人完成单垛的整个流程，对于单工位的机器人，仅有一个码垛流程；对于多工位的机器人，则有多个码垛流程。码垛流程中需要设置取料的入口位置（常规是输送带）和放料的出口位置(常规是托盘)。码垛入口、垛盘信息、码垛路径、码垛层数、码垛样式组成码垛流程。注：由于码垛方式和支持码垛的作业类型较丰富，与简单码垛有很大区别，所以称为高级码垛。在后续说明过程中，高级码垛过程简称码垛。

20.2 文件导入导出

20.2.1 文件导入

文件导入是从外部设备中导入码垛所需的配置文件。

表 19-1 文件导入操作步骤

步骤	图片	描述
1. 打开高级码垛 APP，点击“**文件**”按钮点击“**导入**”。		

		
2.判断有无外部设备。		若无外部设备则弹出左边图示的提示框。 若有外部设备则进入导入界面。
3.导入文件夹操作。		如左图所示，选择要导入的文件夹点击“**导入**”，若示教器中有与“**pallet2**”文件夹重名的文件，则会提醒是否合并文件夹并替换文件夹中同名文件，点击是替换掉导入成功。

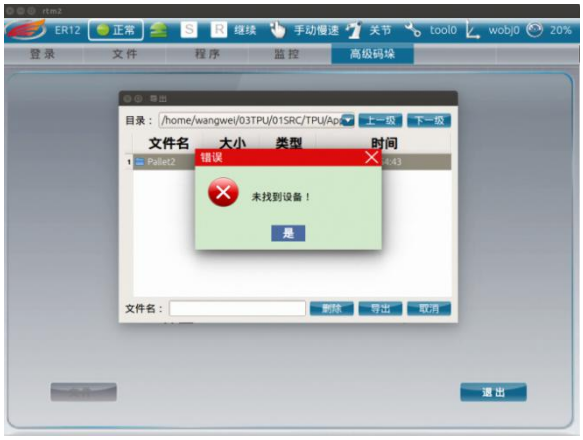
	 	

20.2.2 文件导出

文件导出是将高级码垛设置好的 xml 文件所在文件夹导出到外部设备中。

表 19-2 文件导出操作步骤

步骤	图片	描述
1. 打开高级码垛 APP，点击“**文件**”按钮点击“**导出**”。		

		
2.判断有无外部设备。		点击“**导出**”按钮，若外部无设备则会弹出提示框进行提醒。
3.导出文件夹操作。		如左图所示，选择要导出的文件夹点击“**导出**”，若外部设备中有与“**pallet2**”文件夹重名的文件，则会提醒是否融合且替换，点击是替换掉导入成功。

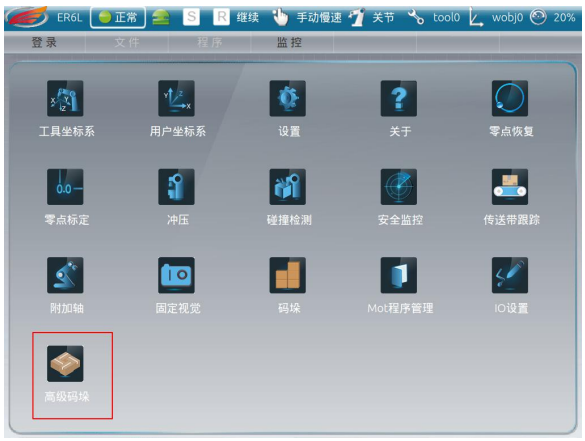
20.3 高级码垛设置

高级码垛的设置功能包括四个模块，分别是基本信息设置、标定设置、工艺流信息设置和虚拟运行，下面对四个模块的设置进行说明。

20.3.1 基本信息设置

20.3.1.1 基本参数设置

表 19-3 基本参数设置操作步骤

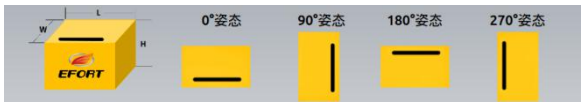
步骤	图片	描述
1.打开高级码垛APP。		打开示教器桌面，点击高级码垛功能图标。 进入高级码垛界面，点击设置图标。 点击基本信息设置按钮进入基本信息设置界面。

	 	
2.设置码垛基本信息数据。		左侧为码垛基本信息设置的流程图。 右侧为基本信息设置。具体设置通过下拉框确定参数值，参数信息如下： 夹持区数量：夹持区是指通过 IO 控制的单个末端抓取工具，当前支持最多 4 个夹持区。 传送带数量：码垛系统中使用的来料设备，当前支持最多 2 个传送带； 码放产品种类数：码放作业中被码放的产品种类个数，当前支持最多 2 种产品尺寸可供配置； 码放垛位数量：码放作业中需要码放的 垛盘

		位置，当前同时支持最多4个垛位同时码放； 工艺流数量：码放作业中支持的码放路径个数，每个码放路径对应一个工艺流程，当前支持最多4个工艺流。 设置完毕后点击右下角“**向右箭头**”按钮进入第一个设置模块，点击“**返回**”按钮回到主界面。

20.3.1.2 尺寸参数设置

表 19-4 尺寸参数设置操作步骤

步骤	图片	描述
1.码垛的参数设置模块包括两个部分：托盘与隔板参数部分，产品参数部分。	  	先设置托盘的长宽高参数。 设置隔板的长宽高参数。 在托盘&隔板参数设置界面点击右下角**“向右箭头”**按钮，切换到产品参数设置界面。设置若干个产品的长宽高参数。 中间图片中立方体上顶面黑色线条位置与产品标签所在平面一致，并根据标签所在平面约定了产品的四个姿态角度，详见右侧四幅投影图片，可以用于实际码放时设置标签所在平面。 产品参数设置完毕后可以点击右下角**“向左箭头”**按钮，返回到托盘与隔板参数设置界面，也可以点击右下角“向右箭头”按钮，切换到虚拟手爪的设置界面。

20.3.1.3 虚拟手爪设置

虚拟手爪是指实际手爪在作业中对应的实际手爪的映射，如在实际作业过程中存在单抓单放和单抓多放的作业，且这两种作业会夹叉使用，此时需要设置虚拟手爪为两个，其中一个用于单抓单放，另外一个用于单抓多放。

夹持区是指机器人末端工具码放过程中可单独信号控制的实际单个手爪区域。

夹持带是在虚拟手爪下中实际取放作业的夹持区域和夹持信号对应的逻辑映射。

注意：虚拟手爪当前支持最多设置 4 种；夹持带数量等于机器人一次最多可以抓取产品的数量，且一个夹持带支持设置多个夹持区。单抓单放作业则虚拟手爪内设置一个夹持带，单抓多放作业则虚拟手爪内设置多个夹持带。

表 19-5 虚拟手爪设置操作步骤

步骤	图片	描述
1. 设置虚拟手爪，并保存。	 	点击方框 1 处虚拟手爪数量编辑框，在弹窗中设置码垛作业需要的虚拟手爪种类。 在方框 2 处虚拟手爪下拉框中选择设置的虚拟手爪编号。 方框 3 处内容显示当前在设置的虚拟手爪名称。 方框 4 处设置当前虚拟手爪包含夹持带的数量。 方框 5 处设置当前虚拟手爪每个夹持带中夹持区的起始编号。 点击**“保存”**按钮，保存设置的基本信息数据。

20.3.2 标定设置

手爪坐标系（码垛手爪是指机器人末端工具，如夹持器、真空吸盘或者海绵吸具）即机器人的工具坐标系，用于显示机器人末端的实际位姿。对于常规作业单台机器人末端只夹持一个手爪，因此只有一个手爪坐标系。但是对于复合手爪或者通过快换盘切换的多手爪情况，此时有几个手爪则有几个手爪坐标系。

注意：标定时应注意时应保持机器人所有手爪都抓取了产品的情况下进行标定，且保证在机器

人坐标系下标定。

20.3.2.1 标定手爪

表 19-6 标定手爪操作步骤

步骤	图片	描述
1.记录一个位置坐标。		机器人抓取产品后，将其末端移动到托盘的一个顶点。 点击图中**“记录”**按钮。 参考位置的 X1, Y1, Z1 将会显示此时机器人坐标值，并且此时旋转 180 度位置坐标的**“记录”**按钮变为可点击状态。
2.将机器人末端旋转 180° 后，确保抓取的产品处于相同位置并记录。		点击图示中**“记录”**按钮。 参考位置的 X2, Y2, Z2 将会显示此时机器人坐标值，并且此时修正值的**“计算并激活”**按钮变为可点击状态。
3.设置手爪的高度。		点击手爪高度编辑框，在弹窗中设置工具手爪高度。

4.计算手爪标定的修正值。		选择工具坐标系中标定好的工具坐标系。 点击“计算并激活”按钮后，修正值中 X，Y，Z 的值会完成计算并显示，同时第二个**“记录”**按钮与**“手爪高度”**编辑框，**“计算并激活”**按钮变为不可点击状态。 点击**“向右”**箭头按钮可以切换到托盘标定界面。

20.3.2.2 标定托盘

托盘坐标系是指机器人码放产品的托架坐标系，即托盘坐标系是用户坐标系，码放产品的位姿信息需要在垛位坐标系建立。一般存在单机器人多垛位的情况，即机器人同时需要多个垛位进行码垛作业(即托盘坐标系个数大于 1)，则需要建立多个垛位坐标系。

表 19-7 标定托盘操作步骤

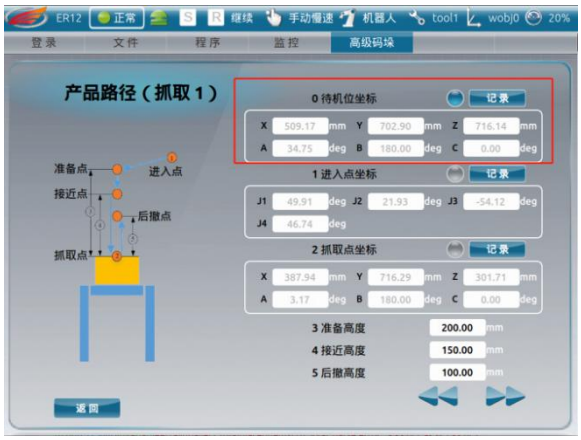
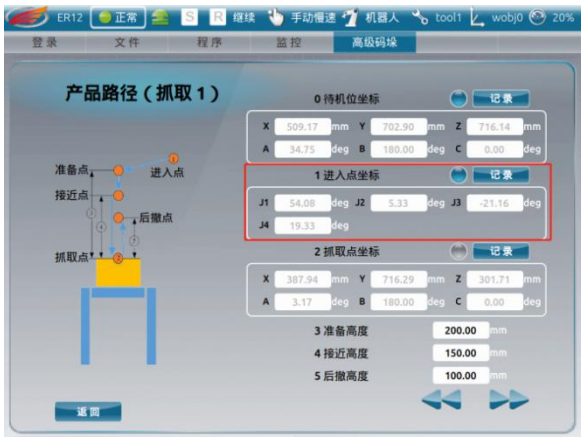
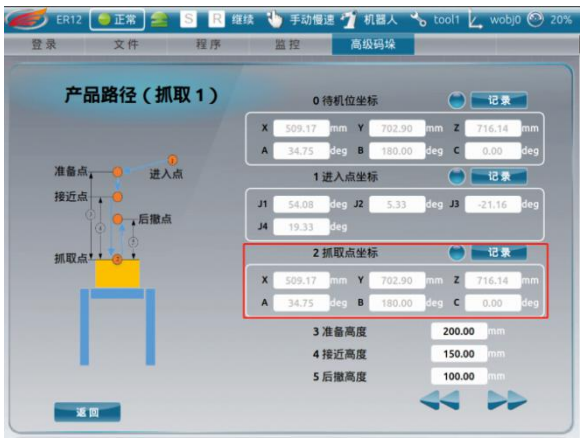
步骤	图片	描述
1.标定托盘个数根据前面基本信息设置的托盘个数为准，设置几个垛位那么就需要对几个托盘进行标定，以托盘 1 为例进行说明，其他托盘同理进行设置。	 	在托盘上建立一个坐标系。机器人抓取产品，选择托盘上三个顶点，进行记录，首先记录原点位置。 选取三个顶点，机器人抓取产品后，将其移动到托盘的一个顶点（将该顶点位置视为原点，沿该顶点两边为 X,Y 轴建立坐标系）。 点击**“记录”**按钮，此时按钮左侧的指示灯会被点亮，表示已记录；原点位置的 X1,Y1,Z1 值会被显示；同时 X 轴方向的**“记录”**按钮会从不可点击状态变为可点击状态。

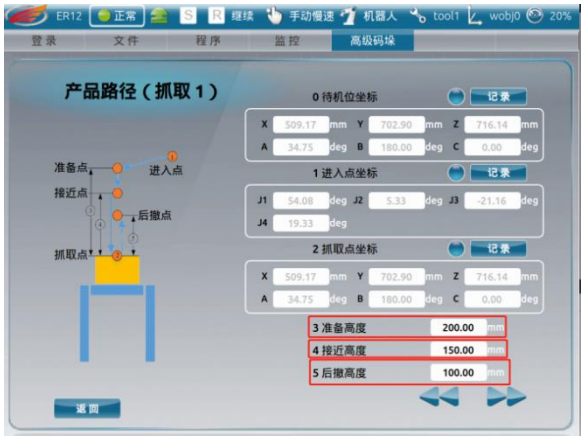
2.将抓取的产品移动至坐标系的 X 轴上，并示教。		机器人抓取产品状态，将末端工具平移到托盘上定义的一个 x 轴上相邻顶点。 点击**“记录”**按钮，此时按钮左侧的指示灯会被点亮，表示已记录；原点位置的 X2,Y2,Z2 值会被显示；同时 Y 轴方向的**“记录”**按钮会从不可点击状态变为可点击状态。
3.将抓取的产品移动至坐标系的 Y 轴上，并示教。		机器人抓取产品状态，将末端工具平移到托盘上定义的一个 y 轴上相邻顶点。 点击**“记录”**按钮，此时按钮左侧的指示灯会被点亮，表示已记录；原点位置的 X3,Y3,Z3 值会被显示；同时标定结果的**“计算并保存”**按钮会从不可点击状态变为可点击状态。
4.计算标定结果。		选择在前面标定好的工具坐标系。 点击**“计算并保存”**按钮后，修正值中坐标值会完成计算并显示，此时界面上的按钮回到初始状态。 **注意：如果选择多个垛位的话则多个托盘的坐标系名字不能相同，否则会提示该坐标系已被使用。**

20.3.2.3 标定抓取路径

表 19-8 标定抓取路径操作步骤

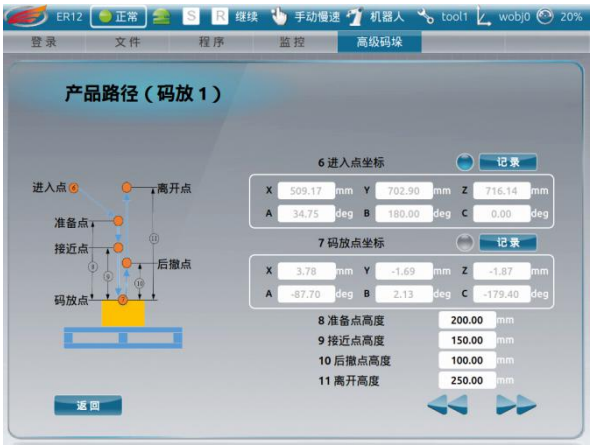
步骤	图片	描述

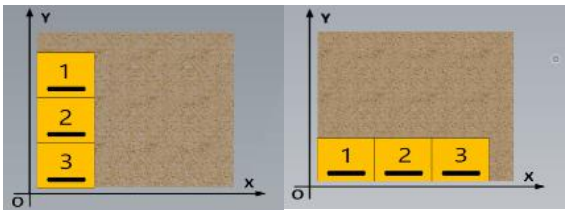
1.抓取路径同样也可以有多个，它的数量是根据用户在前面设置的传送带的个数来决定的，首先记录待机位坐标。		将机器人运动到一个自己想要的待机位置，点击**“记录”**按钮。 此时**“记录”**按钮旁边的指示灯会点亮，待机位的点的信息会显示。
2.记录抓取进入点位置。		如界面左侧图示，将机器人末端夹爪移动至一个进入点，点击**“记录”**按钮。 此时**“记录”**按钮旁边的指示灯会点亮，进入点坐标数据会显示。 **注意：为防止机器人达到限位，此时记录的点的位置是关节坐标的值。**
3.记录抓取点坐标。		将机器人末端夹爪移动至要抓取的位置。点击**“记录”**按钮。 此时**“记录”**按钮旁边的指示灯会点亮，抓取点坐标数据会显示。

4.设置抓取路径上的高度。		抓取点向上运动一个准备高度即为准备点，点击准备高度编辑框设置一个高度值。 抓取点向上运动一个接近高度即为接近点，点击接近高度编辑框设置一个高度值。 抓取后离开时，抓取点向上运动一个后撤高度即为后撤点，点击后撤高度编辑框设置一个高度值。

20.3.2.4 标定码放路径

表 19-9 标定码放路径操作步骤

步骤	图片	描述
1.存在几个垛位，则有几个码放的路径设置。以一号垛位的码放路径设置为例进行说明，二号垛位的设置相同。 对于码放路径的设置，首先记录进入点位置。	 	如界面左侧图示，将机器人末端夹爪移动至一个进入点，点击**“记录”**按钮。 此时**“记录”**按钮旁边的指示灯会点亮，进入点坐标数据会显示。

2.记录码放点坐标。码放点为码放过程实际码放的第一个产品码放位置。	 	示教码放点需满足两点要求： 所有末端夹爪工具上均抓满产品，且所有产品为约定的 0° 姿态。 确保示教点位时所有夹持带为 Y 轴负方向或 X 轴正方向夹持带序号递增姿态。 将机器人末端夹爪按要求移动至初始码放位置。点击**“记录”**按钮。 此时**“记录”**按钮旁边的指示灯会点亮，码放点坐标数据会显示。 **注意：此时记录的码放点是相对于用户坐标系下的码放点。**
3.设置码放路径上的高度。		码放点向上运动一个高度即为准备点，点击编辑框设置一个高度值。 码放点向上运动一个高度即为接近点，点击编辑框设置一个高度值。 码放后离开时，码放点向上运动一个高度即为后撤点，点击编辑设置高度值。 码放点向上运动一个高度即为离开点，点击编辑框设置一个高度值。

20.3.3 工艺流信息设置

20.3.3.1 选择工艺流和对应参数

表 19-10 工艺流设置操作步骤

步骤	图片	描述
1.选择当前设置的工艺流编号。		在**当前工艺流编号**右边的下拉框中选择当前设置的是哪一个工艺流。
2.设置当前工艺流对应的参数。		选择当前工艺流对应的输送线编号，从而确定抓取路径。 选择当前工艺流对应的产品编号，确定使用哪种产品。 选择当前工艺流对应的垛位编号，确定码放路径和垛盘。 选择当前工艺流的工作方式，是码垛还是拆垛。 注：拆垛和码垛设置工艺流程步骤相同，后面以码垛为例进行说明。

20.3.3.2 设置平面样式

表 19-11 平面样式设置操作步骤

步骤	图片	描述
1.选择设置的样式编号。		在名称下拉框中选择当前要设置的样式编号，目前支持设置 8 种样式，名称分别为样式 1 至样式 8。

		
2.选择对应的平面样式。		在样式下拉框中选择当前码放的样式，目前支持 3 种码放样式，分别为行列样式，针轮样式，组样式。
3.根据选择的平面样式设置对应的参数。		若选择的样式为行列样式或针轮样式，则需点击行数和列数的编辑框设置行数与列数； 若选择的样式为组样式，则需先点击组数编辑框设置组样式的组数，并点击行数和列数的编辑框设置每个组的行数和列数。设置完一组行和列之后，选择前面设置的产品后自动加载产品尺寸信息，以及该产品的旋转角度。

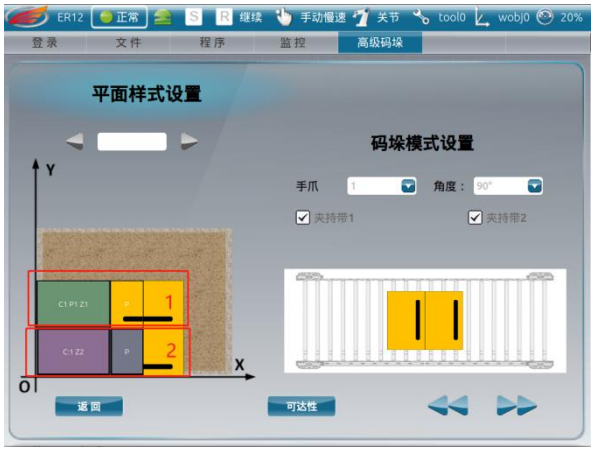
	 	
4.选择产品的旋转角度。		点击角度下拉框，选择产品旋转的角度，则在左边的托盘上产品根据角度进行旋转。 产品可旋转的角度一共有 0°，90°，180°，270° 四个角度。

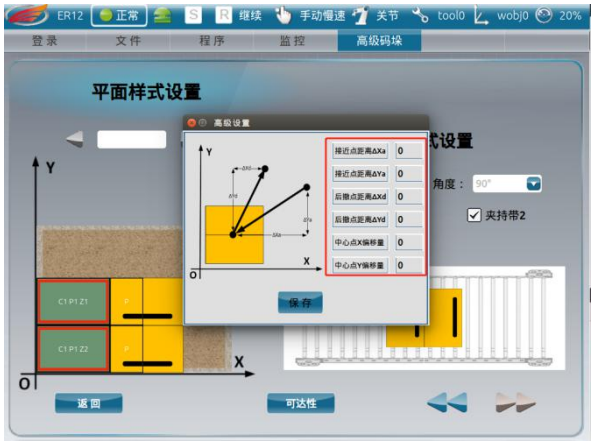
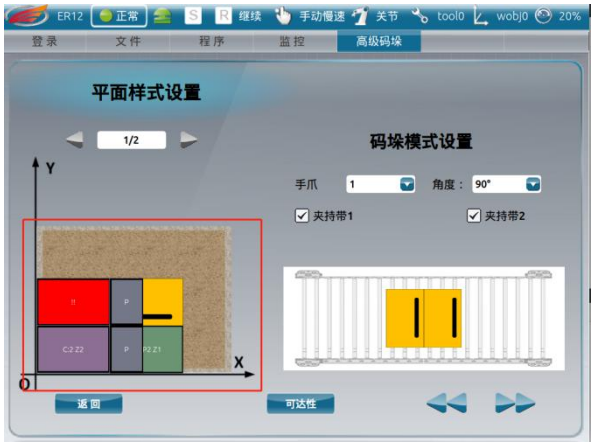
		
5.设置 X,Y 轴边距。		点击 X,Y 编辑框进行编辑码放时距离托盘 X,Y 轴的边距。
6.超出托盘提醒。		设置完成后点击下一步，若设置的产品到了托盘的外面，则会弹出产品码放已超出托盘界限的提醒。

20.3.3.3 设置码垛样式

表 19-12 码垛样式设置操作步骤

步骤	图片	描述

1.选择虚拟手爪和传送带上的产品角度。	 	从手爪下拉框中选择虚拟手爪界面设置的虚拟手爪，则会显示对应虚拟手爪的夹持带。 根据传送带的产品摆放角度，从角度下拉框中选择角度，目前支持选择 4 种角度，分别为 0°，90°，180°，270°。
2.设置具体码垛步骤信息，以行列样式为例。		根据每次抓取的产品数量来勾选夹持带个数，若需一次抓取 n 个，则按顺序勾选 n 个夹持带。 每次放完之后的都会弹出一个高级设置的弹出框，在框内可以根据图示设置用户作业中需要的偏移值。 产品未码放状态背景颜色为黄色，在图片左侧基础图上点击产品，产品背景变为紫色表示选中状态，在弹出灰色背景的 P 按钮上点击，表示在选中的产品位置进行码放，产品变为绿色。依次将左侧所有黄色背景产品点击变为绿色。点击顺序即为实际作业中产品码放顺序。

	  	若码放过的位置用户继续码放的话产品产品颜色会变成红色，提醒用户此处已经有产品了。

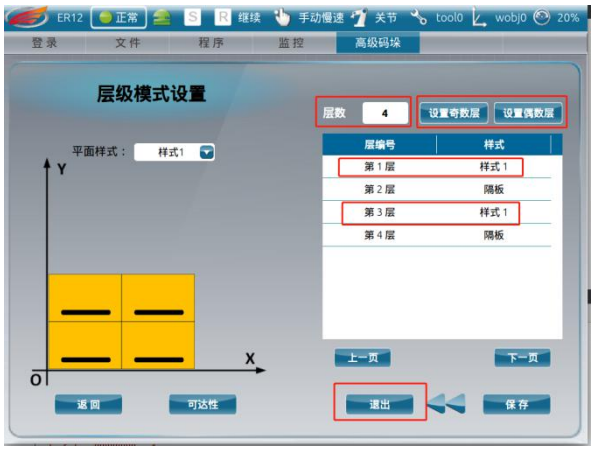
3.码放结束后点击左上角的步骤信息可以查看每一次的码放记录。	 	
4.可达性检测		点击**“可达性”**按钮进行检测当前平面样式的点位信息是否可达。 根据可达性检测系统给出的弹框可以预先检测码放点位是否可达。

20.3.3.4 码垛层级模式设置

表 19-13 码垛样式设置操作步骤

步骤	图片	描述

1.每个工艺流都有自己对应的垛位，在垛位上对应着每层的样式，设置层级模式如右图及步骤所示。	 	进入层级模式界面会读取上次保存的层级模式设置数据，或未曾设置状态。 若需要修改当前垛位的层级模式信息，可直接在原设置基础上进行编辑。 在平面样式下拉框中选择隔板或各平面样式，下方会显示对应样式，右方表格中显示无内容。 点击右侧上方的**“添加层”**按钮，将选中的平面样式添加到最上层，点击**“删除层”**按钮可以删除当前最上层码垛内容。 注意：目前最多支持设置 20 层，最少为 0 层。表格左边一栏表示第几层右边一栏表示对应的样式。
2.点击“快捷设置”，用户进入快捷设置界面。		进入快捷设置界面先在左上角层数上选择托盘层数。 点击**“设置奇数层”**或**“设置偶数层”**会把左边选择的样式加到对应奇数层或偶数层中。 点击**“保存”**按钮，将会把用户配置的信息保存。 注意：层级模式设置的次数是根据用户在前面基本设置中设置的垛位数量决定的。

		
3.可达性检测。		点击**“可达性”**按钮。 此时检测的是垛位上最上面一层的点位信息。 根据可达性检测系统给出的弹框可以预先检测最上层的码放点位是否可达。

20.3.4 虚拟运行

表 19-14 虚拟运行操作步骤

步骤	图片	描述
1.进入虚拟运行功能。		打开示教器桌面，点击高级码垛功能图标。 进入高级码垛界面，点击设置图标。 点击**“虚拟运行”**按钮进入虚拟运行界面。

2.开始虚拟运行。		设置完成后点击**“开始虚拟运行”**按钮进入虚拟运行状态。 进入高级码垛界面，点击设置图标。 点击**“虚拟运行”**按钮进入虚拟运行界面。
3.结束虚拟运行。		RPL 程序运行结束后或想结束虚拟运行状态，点击**“结束虚拟运行”**按钮，再重启机器人。 可点击**“返回”**按钮返回设置主界面。

20.4 运行码垛程序

用户在完成所有码垛设置之后，需先切换到生产界面更新配置，再初始化功能，码垛则运行 PALLET2DEMO.RPL 程序，调用码垛功能。拆垛则运行 DEPALLET2DEMO.RPL 程序，调用拆垛功能。

20.4.1 程序常用

20.4.1.1 常用程序变量

1) DINT pkret：“pallet2.updatePick”函数返回值；
2) DINT plert：“pallet2.updatePlace”函数返回值；
3) DINT cycleid：当前层对应的循环 id 编号；
4) DINT layerid：当前层编号；
5) DINT placeid：码放次数编号，为第几次码放；
6) DINT flowid：工艺流编号；
7) DINT cfgret：更新配置文件函数的返回值；
8) DINT wobj：存储托盘的用户坐标系；

9) DINT no：for 循环变量；

10) DINT run：程序变量；

20.4.1.2 常用函数

1) initFlow(flowid);

初始化工艺流，将对应编号的 flow 置为当前工艺流并初始化当前工艺流中的数据。

2) updateFlow(flowid);

更新工艺流，将对应编号的 flow 置为当前工艺流，并获取当前工艺流对应的数据。

3) updatePick(layerid,cycleid);

更新当前层的当前循环对应的抓取路径的点位信息值。

4) PickGippersOpen();

机器人抓取产品时打开手爪。

5) PickGippersOpen();

机器人抓取产品时闭合手爪。

6) WaitPickGrippersOpened();

等待机器人抓取产品时手爪打开。

7) WaitPickGrippersClosed();

等待机器人抓取产品手爪闭合。

8) PlaceGrippersOped();

机器人码放产品时闭合手爪

9) WaitPlaceGrippersOpeded();

等待机器人码放产品时手爪打开。

10) Ioprocess();

信号处理过程，等待码垛准备信号和来料信号，当码垛准备信号为 true 则将对应的码垛信号置为 false。

20.4.2 程序示例

```
LABEL layer_label :
LABEL cycle_label :
LABEL wait :
pallet2.ioprocess() ;
IF pallet2.ffull[1] THEN
  layerid,cycleid := pallet2.initFlow(1) ;
END_IF ;
IF pallet2.ffull[2] THEN
  layerid,cycleid := pallet2.initFlow(2) ;
END_IF ;
run := false ;
FOR no := 1 TO 2 BY 1 DO
  IF pallet2.incordy[no] AND pallet2.prdy[no] AND NOT pallet2.ffull[no] THEN
    run := true ;
  END_IF ;
END_FOR ;
IF NOT run THEN
  GOTO wait ;
END_IF ;
IF pallet2.prdy[1] AND pallet2.incordy[1] THEN
  layerid,cycleid := pallet2.updateFlow(1) ;
  flowid := 1 ;
  wobj := wobj1 ;
END_IF ;
IF pallet2.prdy[2] AND pallet2.incordy[2] THEN
  layerid,cycleid := pallet2.updateFlow(2) ;
  flowid := 2 ;
  wobj := wobj2 ;
END_IF ;
pkret := pallet2.updatePick(layerid, cycleid) ;
MJOINT (pallet2.pickEntryPos, v1000, fine, tool1, wobj0) ;
MLIN (pallet2.pickReadyPos, v500, fine, tool1, wobj0) ;
MLIN (pallet2.pickApproachPos, v1000, fine, tool1, wobj0) ;
pallet2.PickGrippersOpen() ;
MLIN (pallet2.pickPos, v1000, fine, tool1, wobj0) ;
pallet2.PickGrippersClose() ;
MLIN (pallet2.pickDepartPos, v500, fine, tool1, wobj0) ;
MLIN (pallet2.placeEntryPos, v500, fine, tool1, wobj0) ;
placeid := 1 ;
```

图 19-1 程序示例（1）

```
40  LABEL place_label :
41  plret := pallet2.updatePlace(placeid) ;
42  MLIN (pallet2.placeReadyPos, v500, fine, tool1, wobj) ;
43  MLIN (pallet2.placeApproachPos, v1000, fine, tool1, wobj) ;
44  MLIN (pallet2.placePos, v1000, fine, tool1, wobj) ;
45  pallet2.PlaceGrippersOpen() ;
46  MLIN (pallet2.placeDepartPos, v500, fine, tool1, wobj) ;
47  MLIN (pallet2.placeLeavePos, v500, fine, tool1, wobj) ;
48  placeid := placeid + 1 ;
49  IF placeid <= pallet2.maxCyclePlcNums THEN
50    GOTO place_label ;
51  END_IF ;
52  MJOINT (pallet2.pickEntryPos, v1000, fine, tool1, wobj0) ;
53  LABEL break1 :
54  pallet2.ioprocess() ;
55  IF pallet2.palletBreak[flowid] = 1 THEN
56    WAIT (pallet2.palletBreak[flowid] <> 1) ;
57  END_IF ;
58  cycleid := cycleid + 1 ;
59  IF cycleid <= pallet2.maxLayerCycles THEN
60    pallet2.updateReg(layerid, cycleid) ;
61    GOTO cycle_label ;
62  END_IF ;
63  LABEL break2 :
64  pallet2.ioprocess() ;
65  IF pallet2.palletBreak[flowid] = 2 THEN
66    WAIT (pallet2.palletBreak[flowid] <> 2) ;
67  END_IF ;
68  layerid := layerid + 1 ;
69  IF layerid <= pallet2.maxLayers THEN
70    pallet2.updateReg(layerid, cycleid) ;
71    GOTO layer_label ;
72  END_IF ;
73  LABEL break3 :
74  pallet2.ioprocess() ;
75  IF pallet2.palletBreak[flowid] = 3 THEN
76    WAIT (pallet2.palletBreak[flowid] <> 3) ;
77  END_IF ;
78  GOTO layer_label ;
```

图 19-2 程序示例（2）

Line1:码垛层循环开始；

Line2:码垛每层码放次数循环开始；

Line3:循环等待；

Line4:io 信号检测；包括（来料信号、垛盘准备信号、满垛信号）

Line5:如果垛位 1 满垛执行 Line6；

Line6:初始化工艺流 1 的信息；包括（层 id、每层码放次数 id）

Line7:结束 Line5 判断；

Line8:如果垛位 2 满垛执行 Line9；

Line9:初始化工艺流 2 信息；包括（层 id、每层码放次数 id）

Line10:结束 Line8 判断；

Line11:将 false 赋值给 run 变量；

Line12:从 1 到 2 循环 no 变量，执行 Line13-Line15；

Line13:如果来料信号、垛盘准备信号、满垛信号满足要求，执行 Line14；

Line14:将 true 赋值给 run 变量；

Line15:结束 Line14 判断；

Line16:结束 Line12 循环；

Line17:如果 run 变量为 false，执行 Line18；

Line18:结束 Line3 等待循环；

Line19:结束 Line17 判断；

Line20:如果垛位 1 准备信号和来料信号满足要求，执行 Line21-Line23；

Line21:更新工艺流 1 的信息；包括（层 id、每层码放次数 id）

Line22:将 1 赋值给 flowid 变量；

Line23:将当 wobj1 赋值给 wobj；（用户可根据实际标定的托盘坐标系来编辑）

Line24:结束 Line20 判断；

Line25:如果垛位 2 准备信号和来料信号满足要求，执行 Line26-Line28；

Line26:更新工艺流 2 的信息；包括（层 id、每层码放次数 id）

Line27:将 2 赋值给 flowid 变量；

Line28:将当 wobj2 赋值给 wobj；（用户可根据实际标定的托盘坐标系来编辑）

Line29:结束 Line25 判断；

Line30:通过当前的层 id 和层循环 id 更新抓取点信息；

Line31:机器人运动到抓取进入点；

Line32:机器人运动到抓取准备点；

Line33:机器人运动到抓取接近点；

Line34:机器人打开手爪；

Line35:机器人运动到抓取点；

Line36:机器人闭合手爪；

Line37:机器人运动到抓取离开点；

Line38:机器人运动到码放进入点；

Line39:将 1 赋值给 placeid 变量；

Line40:码放循环开始；

Line41:通过当前码放 id 更新码放点位信息；

Line42:机器人运动到码放准备点；

Line43:机器人运动到码放接近点；

Line44:机器人运动到码放点；

Line45:机器人打开手爪；

Line46:机器人运动到码放离开点；

Line47:机器人运动到码放结束点；

Line48:将当前码放 id 自增 1；

Line49:如果当前循环的码放动作次数全部完成执行 Line50；

Line50:结束 Line40 的码放循环；

Line51:结束 Line49 判断；

Line52:机器人运动到抓取进入点；

Line53:中断 1 循环开始；

Line54:io 信号检测；包括（来料信号、垛盘准备信号、满垛信号）；

Line55:如果当前工艺流接收到中断信号 1，执行 Line56；

Line56:执行中断操作 1，在当前循环码放结束之后中断；

Line57:结束 Line55 判断；

Line58:当前层的循环 id 自增 1；

Line59:如果当前循环动作次数全部完成执行 Line60-Line61；

Line60:更新当前层数循环数到寄存器中；

Line61:结束 Line2 循环；

Line62:结束 Line59 判断；

Line63:中断 2 循环开始；

Line64:io 信号检测；包括（来料信号、垛盘准备信号、满垛信号）；

Line65:如果当前工艺流接收到中断信号 2，执行 Line66；

Line66:执行中断操作 2，在当前层码放结束之后中断；

Line67:结束 Line65 判断；

Line68:当前层 id 自增 1；

Line69:如果当前层动作次数全部完成执行 Line70；

Line70:更新当前层数循环数到寄存器中；

Line71:结束 Line1 循环；

Line72:结束 Line69 判断；

Line73:中断 3 循环开始；

Line74:io 信号检测；包括（来料信号、垛盘准备信号、满垛信号）；

Line75:如果当前工艺流接收到中断信号 3，执行 Line76；

Line76:执行中断操作 3，在当前垛码放结束之后中断；

Line77:结束 Line75 判断；

Line78:结束 Line1 循环；

20.5 生产

20.5.1 码垛生产状态

表 19-15 高级码生产状态操作步骤

步骤	图片	描述
1.进入高级码垛生产界面。	 	打开示教器桌面，点击**高级码垛**功能图标。 进入高级码垛界面，点击**生产**图标。 进入码垛生产状态界面。

		
2.使用生产状态界面上的快捷键，若存在两个垛位则可以分别使用两个工艺流的快捷键，以工艺流一为例进行说明。		

① 点击**“更新配置”**按钮控制器将会重新读取用户设置的信息更新 RPL 里的变量。
② 点击**“初始化”**按钮将会初始化当前工艺流的信息。
③ 点击**“中断”**按钮中断当前正在进行的码垛作业。
④ 点击**“继续”**按钮来继续完成中断前的作业。
⑤ 长按**“待机位置”**按钮将机器人移动到预定义的位置。
⑥ 长按快捷键将机器人位置移动到设置界面定义的位置。
⑦ 点击**“码垛信息”**按钮进入**码垛信息**界面，进行监控码垛的一些信息。
⑧ 点击**“码垛补偿”**按钮进入**位置补偿**界面，进行码垛过程中的一些层高补偿或平面样式组补偿。
⑨ 点击**“退出”**按钮将退出生产界面。
⑩ 点击**“掉包使能”**按钮打开使能，可以选择为使能一：掉包报警但不停止，使能二：掉包报警立即停止。

20.5.2 码垛生产信息

表 19-13 高级码生产信息操作步骤

步骤	图片	描述

1.进入码垛信息界面。		在生产转台界面点击**“码垛信息”**按钮进入码垛信息界面。
2.码垛信息界面主要是向用户展示码垛过程中的一些信息状态。		

① 展示当前垛位，来料准备、码垛准备、以及满垛信号，绿色代表信号为 true，灰色代表 false。
② 左边灰色框为当前码垛层数，右边蓝色框为一共需码放层数。
③ 左边灰色框为当前单层循环数，右边蓝色框为该层一共需循环数。
④ 左边灰色框为当前垛的码放次数，右边蓝色框为一共需码放次数。
⑤ 左边灰色框为当前垛码放产品数量/当前层码放产品数量，右边蓝色框为当前垛一共需码放产品数量/当前层一共需码放产品数量。
⑥ 开机已码放垛数。
⑦ 码垛节拍统计，用来统计码完一垛所用的时间，单位包/时。
⑧ 点击**“编辑”**、**“重置”**、**“写入”**按钮可以进行编辑并保存当前码垛层数和当前码放单层循环数。
⑨ 点击**“返回”**按钮回到生产状态界面。

20.5.3 码垛补偿

码垛补偿模块是用户在码垛生产过程中遇到码放错误的情况，可以在码垛补偿模块对层高、X、Y 以及角度进行补偿。

表 19-14 高级码垛补偿操作步骤

步骤	图片	描述

1.进入码垛补偿界面。		在生产状态界面点击“**码垛补偿**”按钮进入码垛信息界面。
2.码垛补偿界面。		

① 展示当前工艺流的对应层的平面样式。

② 点击“**工艺流**”下拉框选择要补偿的工艺流。

③ 点击“**层**”下拉框选择要补偿的层。

④ 点击“**层高补偿**”编辑框对层高进行编辑。

⑤ 对每一层对应的每一组进行补偿，默认行列模式和针轮模式都是一组，组模式根据用户设置的组数进行实际补偿。

⑥ 点击“**退出**”按钮退出补偿界面回到生产状态界面。

⑦ 点击“**保存**”按钮将上面设置的补偿值保存到 xml 中。

第 21 章　故障处理

21.1 本章简介

本章主要介绍 EFORT 工业机器人控制器故障处理、驱动器故障处理及程序运行故障处理。

21.2 控制器故障处理

点击状态栏的标注 1 的图标按钮，可以查看系统的事件，包括操作信息、报警信息等等。

图 20-1 系统登录界面

21.2.1 查看事件日志

图 20-2 事件日志界面

1. 事件日志显示区域。显示事件的相关代码、产生日期以及内容。

2. 事件说明区域。显示指定事件产生的原因以及给出的解决方法。

3. 筛选区域。通过勾选不同的事件类型，显示区域显示不同的事件。例如，只勾选报警的选项，信息显示区值显示记录的所有报警。

4. 操作区。包括导出日志，清空，运行监视和导出黑屏。

1) 通过点击相应的事件行，可以在“详细信息”区域显示指定事件产生的原因以及给出的解决方法。

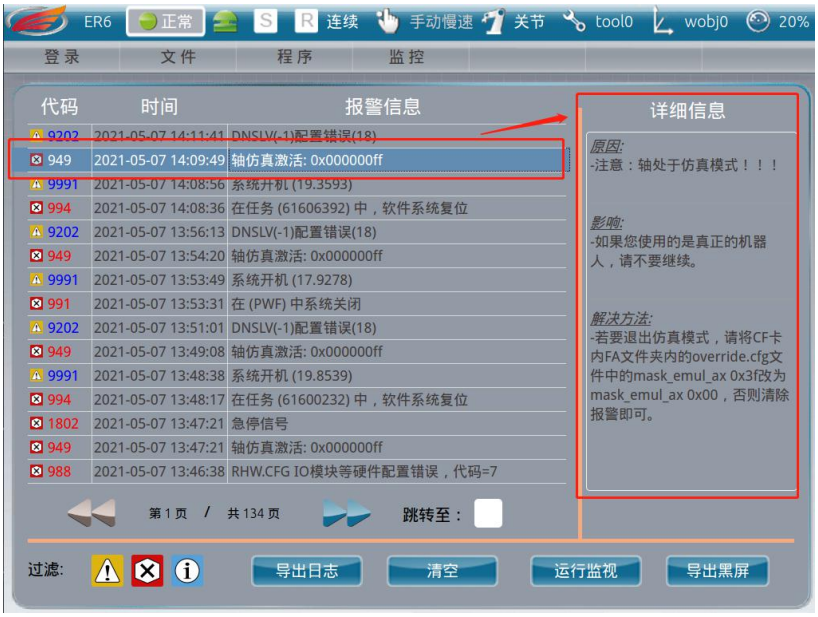


图 20-3 事件日志详情界面

2) 通过点击“**导出日志**”按钮，可将当前所有日志保存至 U 盘中。

3) 通过点击“**清空**”按钮，可将当前所有日志清空。

图 20-4 清空事件日志操作

4) 通过点击“**运行监视**”按钮，切换相应界面，可以查看系统时间、总计伺服开时间、总计上电时间、总计报警时间。

图 20-5 运行监控界面

21.2.2 控制器的故障处理

控制器的故障处理方法可按照事件说明区域给出解决办法进行处理，或参考附录 1。

21.3 驱动器故障处理

在任务栏的“**监控**”菜单下点击“**驱动器**”按钮，进入到驱动器监控界面。这里显示了各轴的驱动的状态，是否有报警以及报警的描述。常见的驱动器报警请查看附录 2。

图 20-6 驱动器监控界面

21.4 程序运行故障处理

程序运行报警在程序的日志界面中可以查看。

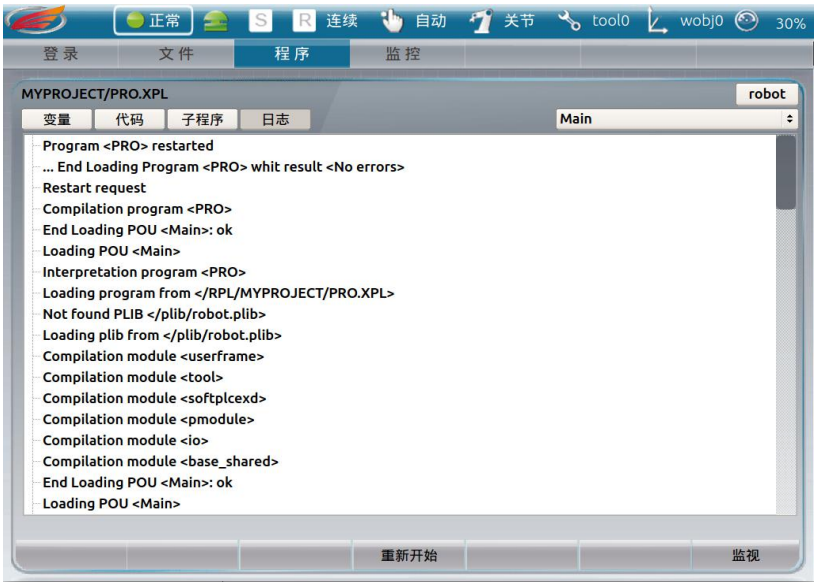


图 20-7 程序运行日志界面

第 22 章　负载辨识

22.1 本章简介

当机器人末端加入负载时对机器人拖动示教功能和碰撞检测等动力学功能产生影响，所以需要负载补偿功能以保证碰撞检测功能正常使用。负载表示加载到机器人末端的任何物体。

负载手动设置和负载辨识最终实现的功能都是使机器人在有负载之后，能够正常使用动力学相关功能。

（1）一般情况下，建议客户根据 3D 数模图里面的数据使用负载手动设置的功能，直接输入参数即可。

（2）负载辨识通过机器人带动负载开展一组预定义的运动实现负载参数的识别，但这组预先定义的运动很可能与周边环境干涉，而且会花一些时间，因此建议优先使用负载手动设置的方式。

22.2 负载辨识界面介绍

22.2.1 负载辨识主界面介绍

表 21-1 负载辨识主界面

步骤	图示
1. 负载辨识主界面。在该界面能够切换负载号，查看、修改负载信息，激活、辨识负载。	

编号 1：当前加载的负载信息及状态：可以激活或取消激活当前的负载；显示当前负载、负载名及激活状态。

编号 2：用来切换负载号（负载 0~负载 9，共 10 组负载），同步更新相应的负载信息。

编号 3：将切换后的负载加载为当前负载，并激活。

编号 4：当点击“修改”按钮后，能够修改负载信息；修改完成后，点击“保存”按钮。

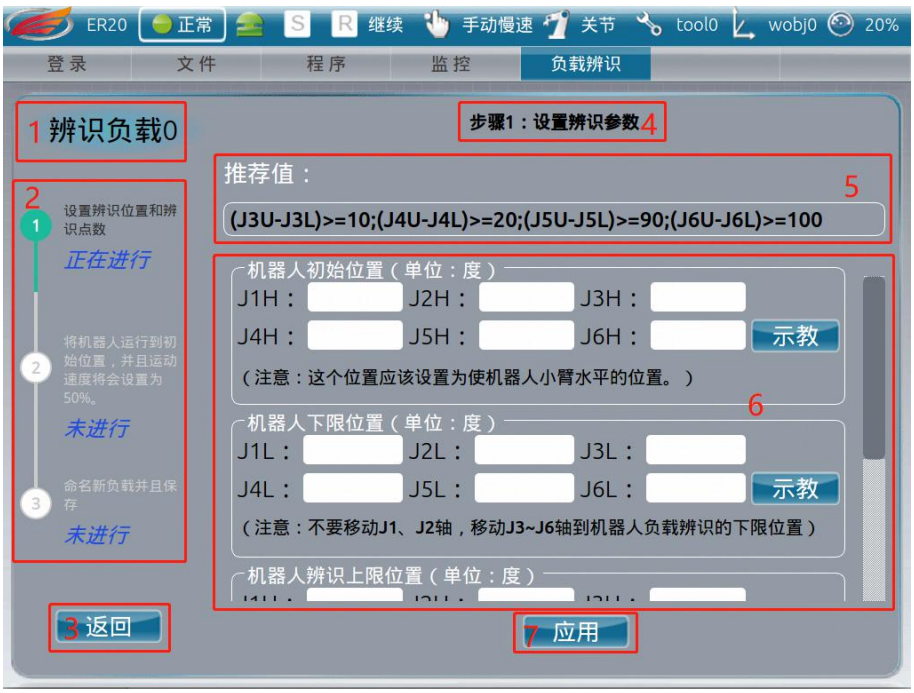
编号 5：点击“辨识”按钮后，进入负载辨识界面。

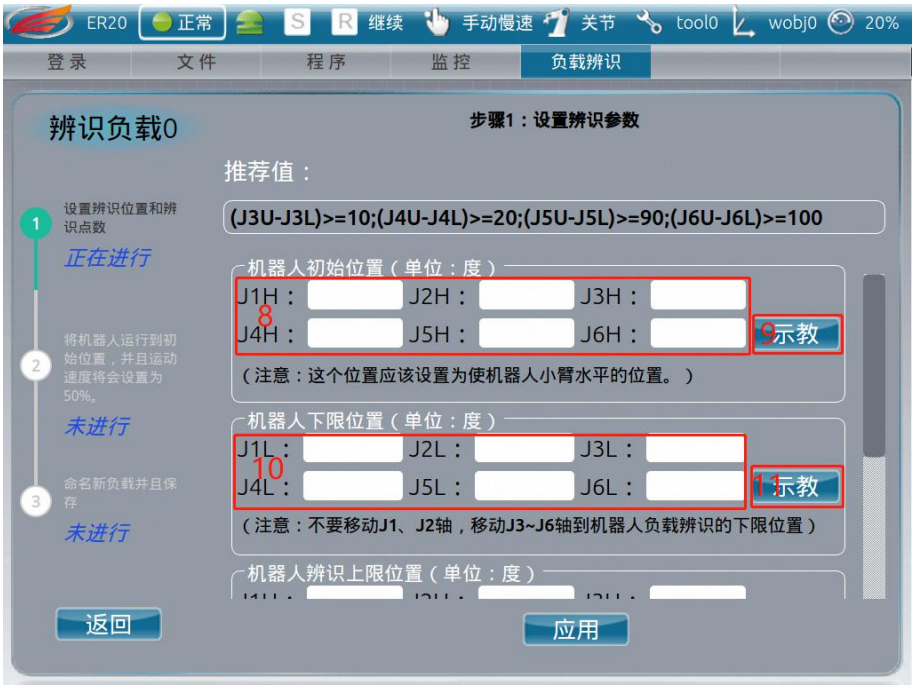
编号 6：显示和编辑负载信息。

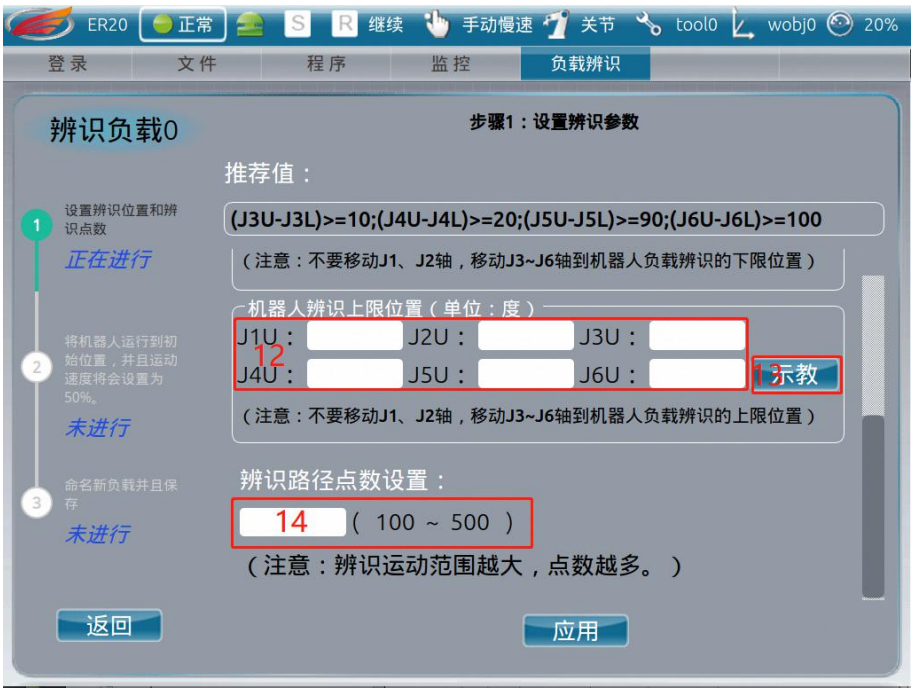
编号 7：点击“退出”按钮后，退出负载辨识界面，返回到主界面。

22.2.2 辨识负载界面介绍

表 21-2 辨识负载界面

步骤	图示
1. 设置辨识参数界面。	





编号 1：辨识的当前负载号。

编号 2：用来显示辨识的进度。

编号 3：点击“返回”按钮，退出辨识过程，进入到负载辨识的主界面。

编号 4：显示辨识的步骤。

编号 5：设置辨识过程中机器人上限位置与下限位置的差值推荐值。

编号 6：设置机器人辨识的初始位置、下限位置、上限位置、辨识路径点数。

编号 7：点击“应用”按钮，将设置的参数保存到控制器中。

编号 8：显示机器人辨识的初始位置值，J3~J6 可以手动修改。

编号 9：点击“示教”按钮，会将机器人当前位置记录为机器人辨识的初始位置。

编号 10：显示机器人辨识的下限位置，J3~J6 可以手动修改。

编号 11：点击“示教”按钮，会将机器人当前位置记录为机器人辨识的下限位置。

编号 12：显示机器人辨识的上限位置，J3~J6 可以手动修改。

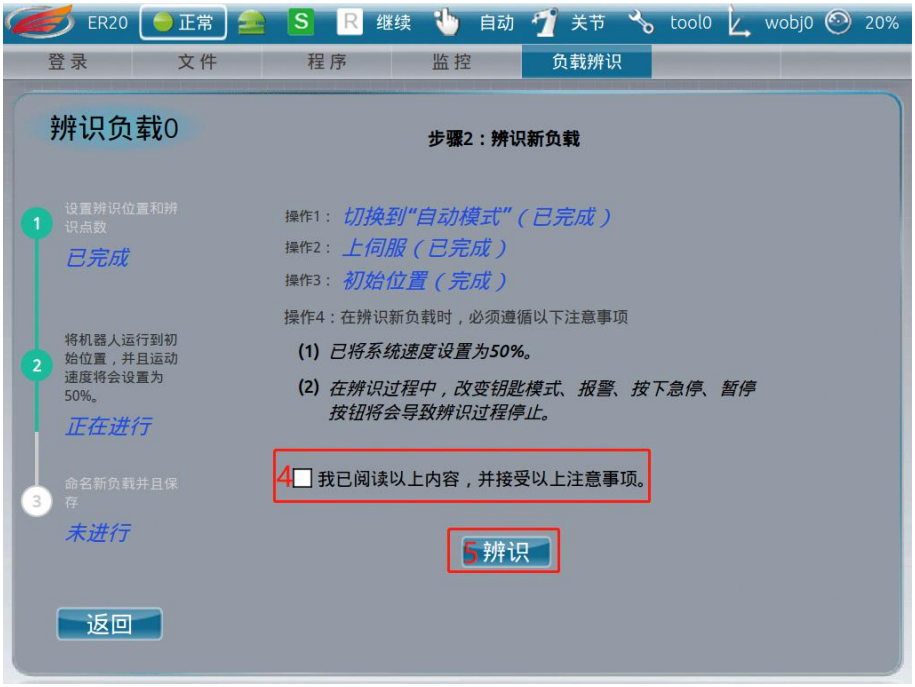
编号 13：点击“示教”按钮，会将机器人当前位置记录为机器人辨识的上限位置。

编号 14：设置辨识路径点数。

2.步骤 2：辨识新负载界面


ER20
正常
继续
自动
关节
tool0
wobj0
20%
登录
文件
程序
监控
负载辨识
辨识负载0
步骤2：辨识新负载
1
设置辨识位置和辨识点数
已完成
操作1：
切换到“自动模式”（已完成）
操作2：按示教器上“PWR”按键来完成上伺服
PWR
2
2
将机器人运行到初始位置，并且运动速度将会设置为50%。
正在进行
3
命名新负载并且保存
未进行
返回


ER20
正常
继续
自动
关节
tool0
wobj0
20%
登录
文件
程序
监控
负载辨识
辨识负载0
步骤2：辨识新负载
1
设置辨识位置和辨识点数
已完成
操作1：
切换到“自动模式”（已完成）
操作2：
上伺服（已完成）
操作3：机器人姿态返回到初始位置
2
将机器人运行到初始位置，并且运动速度将会设置为50%。
正在进行
回初始位置
3
3
命名新负载并且保存
未进行
返回



编号 1：将钥匙模式切换到自动模式后，将自动跳转至下一步。

编号 2：按下示教器上的“PWR”键后，自动跳转至下一步。

编号 3：点击“回初始位置”按钮，机器人将回到初始位置，状态灯由  变为  。

编号 4：勾选后表示接受注意事项。

编号 5：点击“辨识”按钮，将进行辨识负载，进度条辨识辨识的进度。

3. 命名新负载并保存

编号 1：对辨识的新负载命名。

编号 2：负载质量，可修改。

编号 3：负载质心，可修改。

编号 4：负载惯性张量，可修改。

22.3 设置负载操作流程

22.3.1 手动设置负载信息

表 21-3 手动设置负载界面操作流程

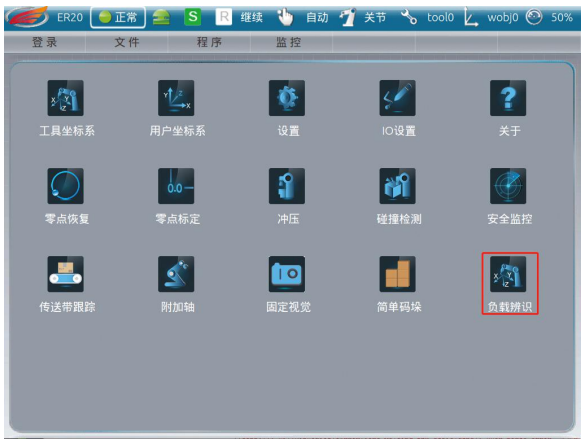
步骤	图片	描述
1.进入负载辨识主界面		1）在示教器主界面点击“负载辨识”，进入负载辨识主界面。

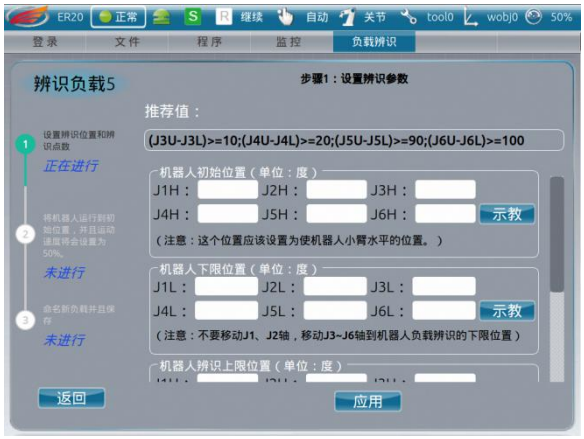
2.手动设置负载	  	1）选择想要设置的负载号； 2）点击“修改”按钮； 3）将 3D 数模计算结果写入到负载信息中，注意单位。 4）点击“保存”按钮，即可将修改的信息保存。 5）若要将修改的负载激活，则点击“激活”按钮，即可置为当前，并激活。

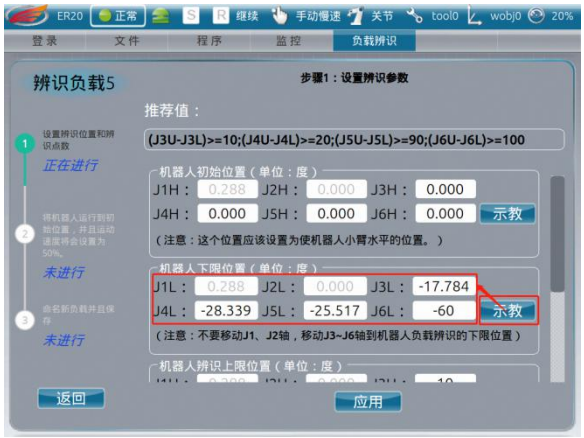
22.3.2 辨识负载信息流程

表 21-3 辨识负载信息操作流程

步骤	图片	描述

1.进入负载辨识主界面	 	
2.辨识负载		1）选择要辨识的负载号。 2）点击“辨识”按钮，若当前负载处于激活状态，会有弹框提示，点击“是”，进入辨识负载界面。 3）将机器人手动运行至辨识开始的初始位置；点击“示教”按钮，会记录下当前机器人的关节位置。 4）将机器人手动移动到辨识的下限位；点击“示教”，会记录下当前机器人的关节位置。 5）将机器人手动移动到辨识的上限位；点击“示教”，会记录下当前机器





人的关节位置。

6）根据辨识的范围设置辨识的路径点数。

7）点击“应用”按钮，若辨识的上下限位置值没有按照推荐的值示教，则会有以下 2 种情况；a、如果某轴的上下限关节差值小于 0，则会有报警提示，此时需要重新示教上下限位置的值；b、如果某轴的上下限的关节差值小于推荐值，会有警告提示，若机器人位置已经受到外部环境的限制，则点击“否”，进入到辨识的下一步，若不受外部环境限制，则重新设置辨识的上下限位置，否则会影响辨识的结果准确性。

8）进入到辨识新负载界面，将示教器上的钥匙模式切换到自动模式，会自动跳转到下一步。

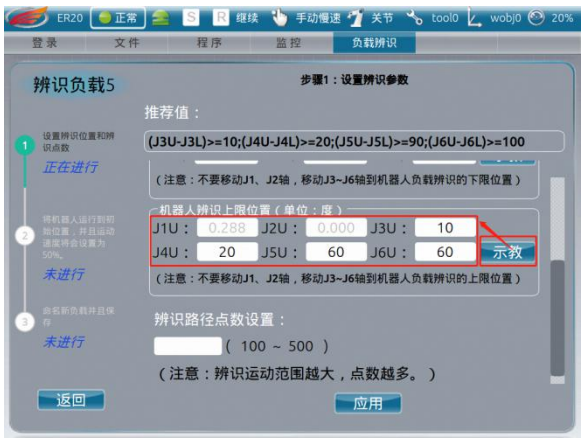
9）点击示教器上的“PWR”按钮，上伺服。

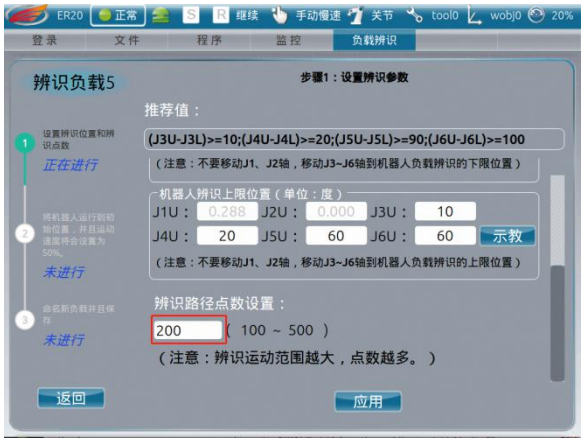
10）点击“回初始位置”按钮，机器人回到设置的辨识开始的初始位置后，自动跳转至下一步。

11）阅读注意事项后，勾选复选框。

12）点击“辨识”按钮，出现进度条，表示辨识的进度，等辨识结束后，会自动跳转至下一页。

13）进入步骤 3 界面后，命名新负载名；若对辨识的结果不太满意，可以修改辨识结果。





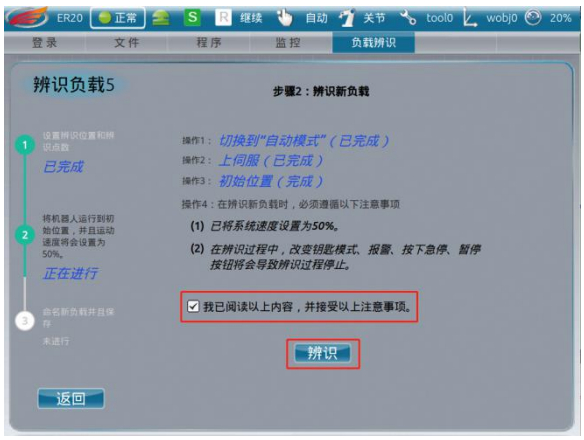
14）点击“保存”按钮。保存成功后会有弹框提示，点击“是”，界面会自动跳转至负载辨识主界面。

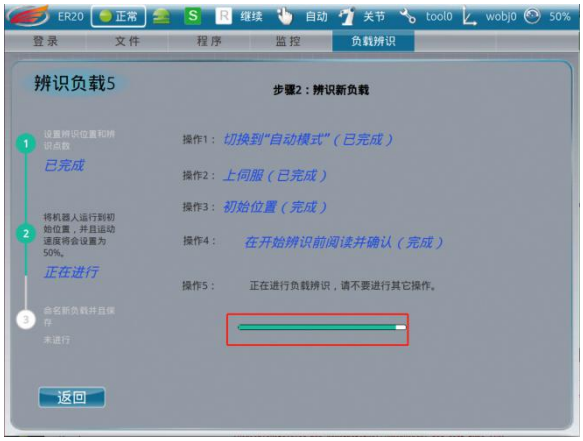
15）若要激活当前辨识的负载，则点击“激活”按钮即可。


ER20
正常
继续
手动慢速
关节
tool0
wobj0
20%
登录
文件
程序
监控
负载辨识
辨识负载5
步骤2：辨识新负载
设置辨识位置和辨识点数
已完成
操作1：切换示教器到“自动模式”（如下所示）
AUTO
T1
T2
将机器人运行到初始位置，并且运动速度将会设置为50%。
正在进行
命名新负载并且保存
未进行
返回


ER20
正常
继续
自动
关节
tool0
wobj0
20%
登录
文件
程序
监控
负载辨识
辨识负载5
步骤2：辨识新负载
设置辨识位置和辨识点数
已完成
操作1：切换到“自动模式”（已完成）
操作2：按示教器上“PWR”按键来完成上伺服
PWR
将机器人运行到初始位置，并且运动速度将会设置为50%。
正在进行
命名新负载并且保存
未进行
返回


ER20
正常
继续
自动
关节
tool0
wobj0
20%
登录
文件
程序
监控
负载辨识
辨识负载5
步骤2：辨识新负载
设置辨识位置和辨识点数
已完成
操作1：切换到“自动模式”（已完成）
操作2：上伺服（已完成）
操作3：机器人姿态返回到初始位置
回初始位置
将机器人运行到初始位置，并且运动速度将会设置为50%。
正在进行
命名新负载并且保存
未进行
返回


ER20
正常
继续
自动
关节
tool0
wobj0
20%
登录
文件
程序
监控
负载辨识
辨识负载5
步骤2：辨识新负载
设置辨识位置和辨识点数
已完成
操作1： 切换到"自动模式"（已完成）
操作2： 上伺服（已完成）
操作3： 初始位置（完成）
操作4：在辨识新负载时，必须遵循以下注意事项
(1) 已将系统速度设置为50%。
(2) 在辨识过程中，改变钥匙模式、报警、按下急停、暂停按钮将会导致辨识过程停止。
将机器人运行到初始位置，并且运动速度将会设置为50%。
正在进行
命名新负载并且保存
未进行
我已阅读以上内容，并接受以上注意事项。
辨识
返回


ER20
正常
继续
自动
关节
tool0
wobj0
50%
登录
文件
程序
监控
负载辨识
辨识负载5
步骤2：辨识新负载
设置辨识位置和辨识点数
已完成
操作1： 切换到"自动模式"（已完成）
操作2： 上伺服（已完成）
操作3： 初始位置（完成）
操作4： 在开始辨识前阅读并确认（完成）
操作5： 正在进行负载辨识，请不要进行其它操作。
将机器人运行到初始位置，并且运动速度将会设置为50%。
正在进行
命名新负载并且保存
未进行
返回


ER20
正常
继续
自动
关节
tool0
wobj0
50%
登录
文件
程序
监控
负载辨识
辨识负载5
步骤3：命名新负载并且保存
设置辨识位置和辨识点数
已完成
负载辨识：
负载名： 18kg负载
质量： 18.32 KG
质心（单位：mm）
Lx: 33.4731
Ly: 16.6365
Lz: -110.7659
惯性张量（单位：Kg*mm^2）
Ixx: 0.0000
Ixy: 0.0000
Ixz: 0.0000
Iyy: 0.0000
Iyz: 0.0000
Izz: 0.0000
将机器人运行到初始位置，并且运动速度将会设置为50%。
已完成
命名新负载并且保存
正在进行
返回
保存

22.4 Module 指令及 RPL 程序用例

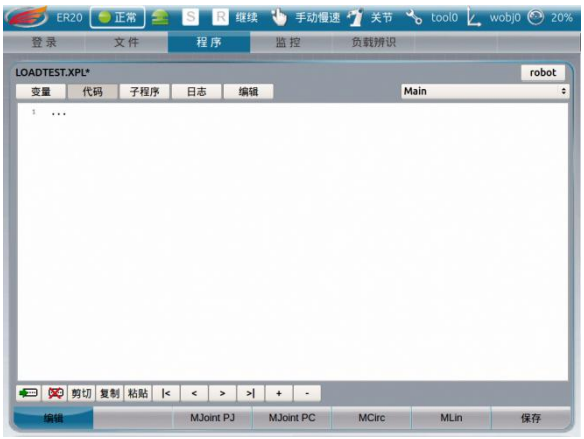
22.4.1 Module 指令介绍

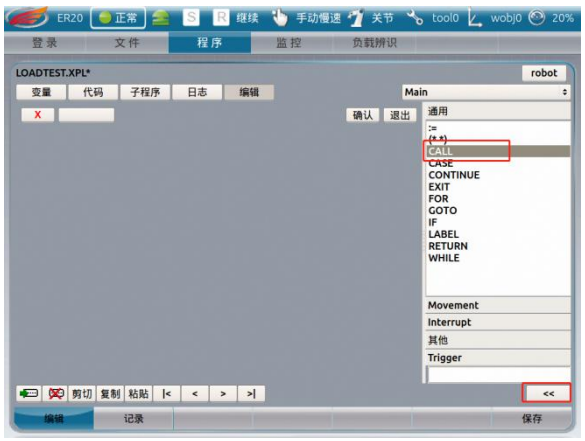
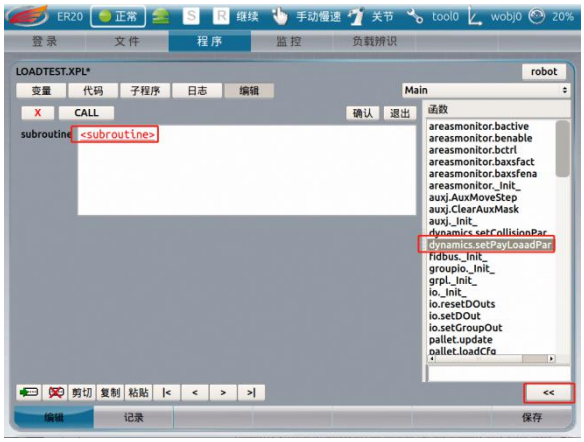
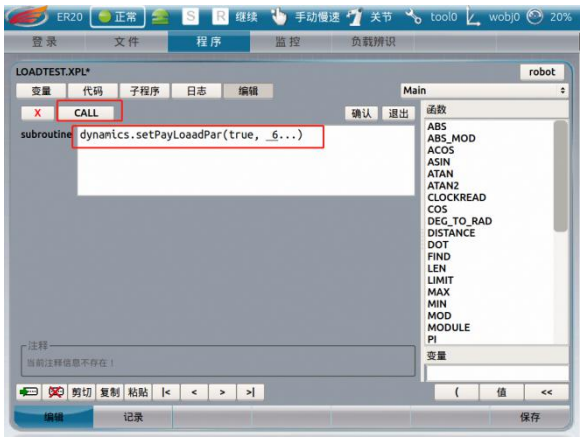
表 21-4 负载指令介绍

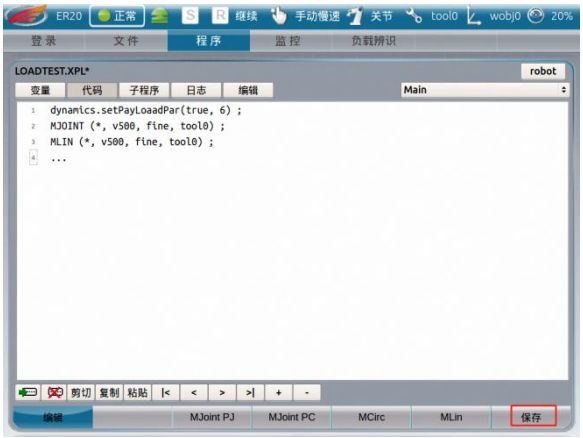
指令	名称	功能
dynamics.setPayLoadPar(Bool isEnablePayLoad， DINT payLoadIndex)	设置负载参数指令	实现负载的激活关闭，以及当前负载号的切换设置

22.4.2 RPL 程序用例

表 21-5 负载程序用例

步骤	图片	描述
1.新建或打开 RPL 程序		

2.插入 call 指令		
3.找到碰撞检测指令，并设置参数如： Dynamics.setPayLoadPar(true,6)	 	RPL 程序中只有加入了此条指令并运行过后所切换的负载参数才会被加载，以及负载的激活状态才能被更新

4.保存程序测试运行即可		

22.4.3 注意事项

1.所设定的负载参数均在机器人法兰坐标下进行描述的。

2.在通过负载指令切换或激活关闭负载时需提前通过指令关闭程序碰撞检测或者调高碰撞检测的全局阈值。

3.无论是工具还是夹持着工件的工具均可被定义为一个负载，需要设置不同的负载参数。

第 23 章　弧焊设置及编程

23.1 功能包简介

弧焊工艺软件主要是指：用户通过对弧焊机器人适当设置后，可以按照指令实现弧焊工艺焊接。利用该软件，用户可以快速设置焊接参数及配置相关设备，以便快速实现机器人焊接。机器人弧焊工艺软件要求主要有以下几点：

1. 人机交互方便，参数设置简单。

2. 可以保存不同焊接工艺，以便实现快速换产。

3. 可以匹配不同厂家的焊接设备。

23.2 弧焊设置

23.2.1 弧焊主界面

表 3-1 弧焊主界面

步骤	图片	描述
1.在示教器桌面上找到弧焊功能并打开。		**“焊接功能开关”**关闭，弧焊功能包不启用，不能进行弧焊相关设置。 如果需要使用弧焊功能，需要先开启**“焊接功能开关”**。
2.进入子功能界面和退出。		在弧焊主界面包括各子功能的按钮，点击对应按钮可以进入相应的子功能界面。（根据版本不同，其上子功能按钮可能不相同。） 点击右下角**“退出”**按钮，可以退出弧焊功能app。

23.2.2 设备设置

表 3-2 设备设置界面介绍

步骤	图片	描述

1.配置“设备设置”参数。

在弧焊主界面点击“**设备设置**”按钮。

配置相关参数，配置完成后，点击“**保存**”，保存设置的参数。

注意：选择模拟量焊机的时候，需要配置在“特性曲线”中设置的电流电压特性曲线。

序号	标题	描述	备注
1	焊接协议	选择焊机类型，目前支持模拟量、麦格米特、奥太、时代四种类型的焊机。	
2	电流文件号	配置焊接电流文件参数，即机器人模拟量输出电流与焊机电流的对应关系。	仅当选择模拟量焊机需要配置，范围：0-19
3	电压文件号	配置焊接电压文件参数，即机器人模拟量输出电压与焊机电压的对应关系。	仅当选择模拟量焊机需要配置，范围：0-19
4	空运行	空运行开关，空运行打开时，运行程序时跳过引弧，熄弧，摆弧。	
5	焊接设备异常	检测焊机异常报警信号的开关。	
6	焊接开关	焊接功能使能开关，只有将焊接开关打开，才能正常使用焊接功能。	
7	碰撞检测开关	检测碰撞报警信号开关。	
8	悬浮窗口开关	示教器界面悬浮窗开关。	
9	空运行笛卡尔速度	机器人空运行时候的笛卡尔速度。单位mm/s。	范围：1-1000
10	手动进退丝速度	手动进退丝的速度。	暂未开放
11	手动进退丝时间	手动进退丝的时间。	暂未开放，目前悬浮窗按钮按住进/退丝，松开停止。

23.2.3 特性曲线

表 3-3 特性曲线界面介绍

步骤	图片	描述
1.配置“特性曲线”参数。仅当焊接协议选择模拟量的时候，才需要配置此参数。	 	在焊接主界面点击“**特性曲线**”按钮。 按顺序配置相关参数： 1. 选择特性曲线类型。 2. 填写特性文件号。 3. 填写使用节点数量。 4. 在如图表格的模拟量列填写机器人模拟量模块实际输出电压值，然后点击生效栏的按钮，这时候，焊机面板上电流/电压值发生改变，将特性曲线类型的对应值手动填写到实际值一列中。 当焊机工作模式为一元化时候，实际值中填写弧长修正值。 配置完所有需要使用的节点后，点击“**保存**”，保存设置的参数。

序号	标题	描述	备注
1	特性曲线类型标题栏	选择当前要配置的特性曲线类型：电流特性和电压特性	蓝色表示当前的类型
2	特性文件号	电流/电压特性文件号。各自对应 20 组参数。	范围：0-19
3	使用节点数	选择有多少点来拟合特性曲线，最少两个点。在节点后填写的数据不参与曲线的拟合。	范围：2-16，点位越多，曲线准确度越高。
4	模拟量	机器人模拟量模块输出值。单位：V。	建议填写的时候根据使用节点数平均分布输出值。范围：0-10
5	实际值	焊机实际电流电压值。当选择电流特性文件，此列代表焊机电流值，单位：A；当	

		选择电压特性文件，此列代表焊机电压值，单位：V。	
6	生效	生效按钮，点击后输出当前行的模拟量值	绿色表示当前生效值。

23.2.4 焊接设置

表 3-4 焊接设置界面介绍

步骤	图片	描述
1.配置“焊接设置”中的参数信息。		在弧焊主界面点击“**焊接设置**”按钮。 根据实际生产需要，配置相关参数。配置完成后，点击“**保存**”，保存设置的参数。

分类	标题	描述	备注
起弧设置	提前送气时间	在起弧前，提前送气时间，单位：s。	范围：0-10
	起弧检测时间	起弧检测时间：例如输入 2s，如果超过 2s 没有引弧成功信号，则引弧失败。	范围：0-10
	起弧确认时间	起弧持续时间：例如输入 0.2s，则只有当起弧成功信号持续 0.2s，才确认起弧成功。	范围：0-10
	起弧次数	重复起弧次数。	范围：1-10
	再起弧退丝时间	再次起弧前，焊丝后退时间：例如输入 1s，则在再次起弧前，焊丝后退 1s 后，再次开始起弧。	范围：0-10
收弧设置	滞后关气时间	在熄弧后，保护滞后关闭的时间。单位：s。	范围：0-10
	收弧检测时间	在收弧之后，检测收弧成功信号的时间，当时设置时间内未检测到收弧信号，则报错。设置为 0，则不检测信号。单位：s。	范围：0-10
	收弧退丝时间	在熄弧后，焊丝回退时间。单位：s。	范围：0-10

再启动设置	再启动使能	使能开时，在焊接过程中断弧或者暂停焊接，按示教器开始按键，可以再次起弧焊接。如果中间手动移开机器人，机器人会再次回到断弧点进行起弧焊接。使能关时，无此功能。	断弧后，焊接信号复位后，则不能进行再启动。
断弧设置	断弧检测	断弧检测功能开关：“开”：当发生断弧时，检测出断弧并根据设置进行后续处理；“关”：当发生断弧时，依然正常完成机器人示教程序。	
	断弧检测时间	断弧信号持续时间：例如输入 0.5s，只有当断弧持续 0.5s，才输出断弧信号。	范围：0-10
刮擦设置	刮擦启动	刮擦启动开关（刮擦启动开启，引弧失败后也会开始运动，在运动过程中若引弧成功，则返回指定距离再继续正常焊接）。	
	刮擦距离	设置刮擦启动后刮擦引弧前进的最大距离。单位：mm。	范围：0.01-100
	刮擦速度	设置刮擦启动后刮擦引弧前进的速度。单位：mm/s。	范围：0.01-100

23.2.5 焊接参数

表 3-5 焊接参数界面介绍

步骤	图片	描述
1.配置“焊接参数”中的参数信息。		在弧焊主界面点击“**焊接参数**”按钮。 配置相关参数，配置完成后，点击“**保存**”，保存设置的参数。

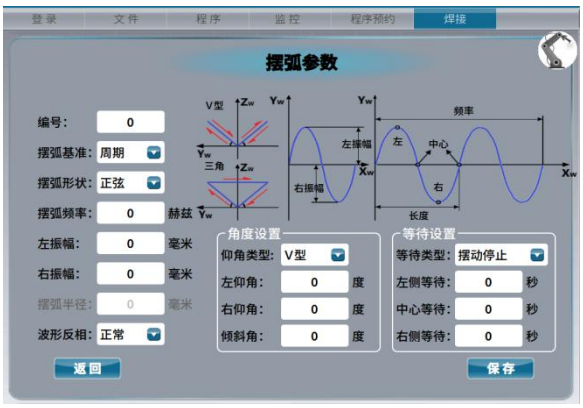
分类	标题	描述	备注
/	文件号	保存焊接参数的文件编号，共可以保存 100 组参数。	范围：0-99
/	注释	客户可以添加注释信息，方便辨识。	
焊接参数	电流模式	电流模式分为电流和送丝速度两个选项，电流：焊接过程中以焊接电流为准；送丝速度：焊接过程中以送丝速度为准。	目前只开放电流值设置方式。
	电压模式	电压模式分为一元化和分别两个选项，一元化：设置焊接电流，焊接电压值焊机自动匹配，可以百分比方式进行上下调节；分别：	

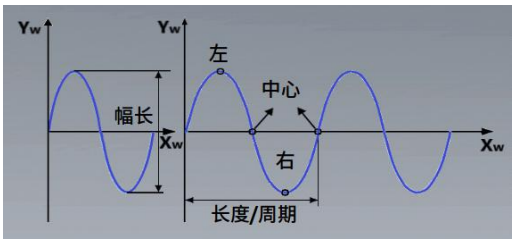
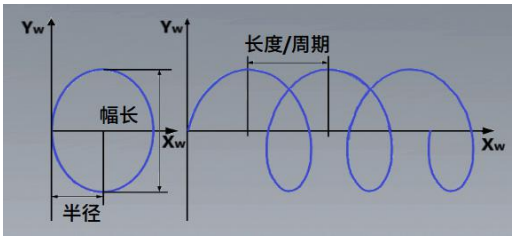
		焊接电流和焊接电压单独给定，互不影响。	
	工作模式	选择焊机的起弧工作模式，不太焊机的工作模式略有不同，请根据焊接工艺需求选择。	
	焊接电流	焊接时，焊机输出电流/送丝速度，电流模式为电流时，单位：A；电流模式为送丝速度时，单位：m/min。	
	焊接电压	焊接时，焊机输出电压强度/电压值，电压模式分别时，单位：V；电压模式一元化时，单位：根据焊机可能为%或 V。	
	焊接速度	正常焊接时，焊枪末端沿焊接方向的速度，单位：mm/s。	范围：0.01-100
起弧参数	工作模式	选择焊机的起弧工作模式，不太焊机的工作模式略有不同，请根据焊接工艺需求选择。	模拟量不支持单独选择，需要和焊接参数的工作模式保持一致。
	起弧电流	起弧时，焊机输出电流/送丝速度，电流模式为电流时，单位：A；电流模式为送丝速度时，单位：m/min。	
	起弧电压	起弧时，焊机输出电压强度/电压值，电压模式分别时，单位：V；电压模式一元化时，单位：根据焊机可能为%或 V。。	
	起弧时间	引弧成功后， 焊机的电流电压由起弧电流电压渐变到焊接电流电压的时间。	
收弧参数	工作模式	选择焊机的起弧工作模式，不太焊机的工作模式略有不同，请根据焊接工艺需求选择。	模拟量不支持单独选择，需要和焊接参数的工作模式保持一致。
	收弧电流	收弧时，焊机输出电流/送丝速度，电流模式为电流时，单位：A；电流模式为送丝速度时，单位：m/min。	
	收弧电压	收弧时，焊机输出电压强度/电压值，电压模式分别时，单位：V；电压模式一元化时，单位：根据焊机可能为%或 V。。	
	收弧时间	熄弧持续的时间。单位：s。	范围：0-10
	防粘丝电流	在收弧结束之后，给出一个防粘丝的电流电压，用来防止焊丝粘接到工件上。设置的防粘丝的电流。	
	防粘丝电压	在收弧结束之后，给出一个防粘丝的电流电压，用来防止焊丝粘接到工件上。设置的防	

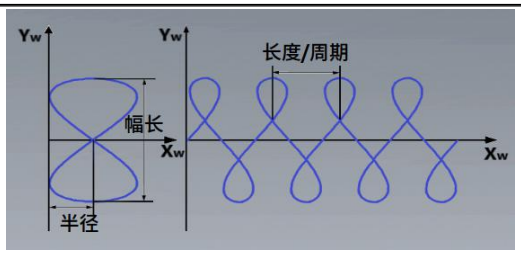
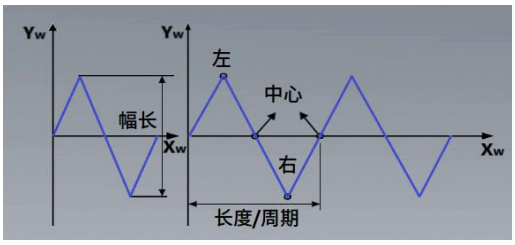
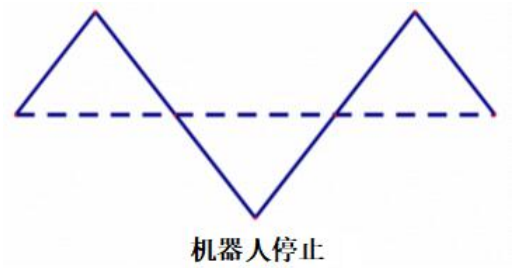
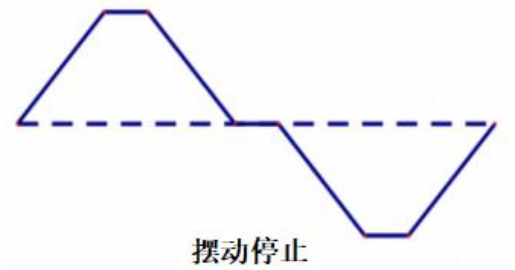
		粘丝的电流。	
	防粘丝时间	设置防粘丝电流电压的持续时间。	

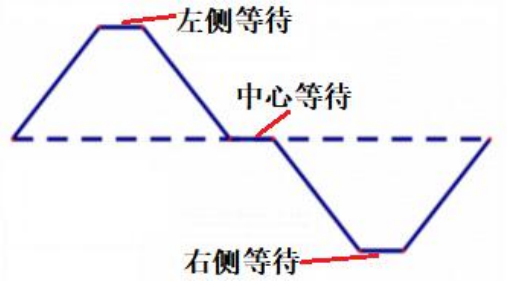
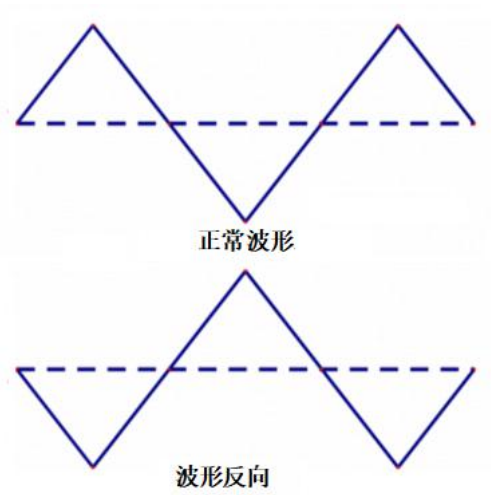
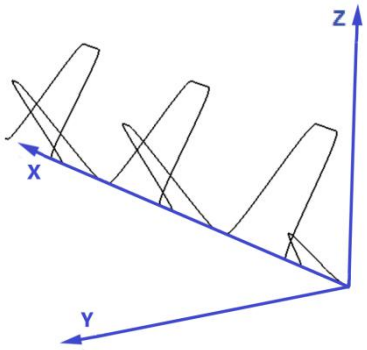
23.2.6 摆弧参数

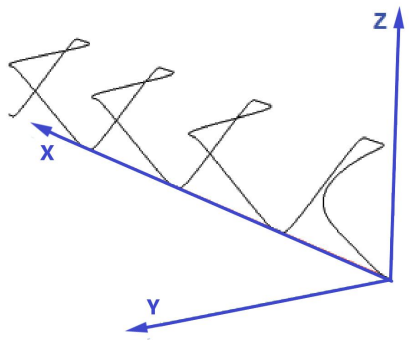
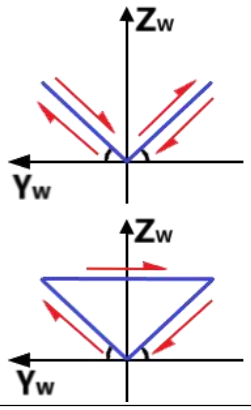
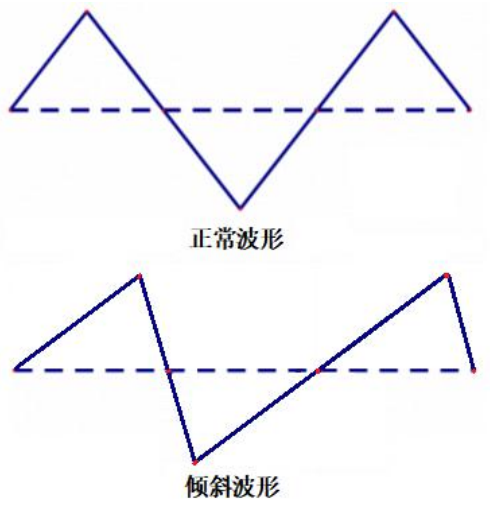
表 3-6 摆弧参数界面介绍

步骤	图片	描述
1.配置“摆弧参数”中的参数信息。		在弧焊主界面点击**“摆弧参数”**按钮。 配置相关参数，配置完成后，点击“**保存**”，保存设置的参数。

序号	标题	描述	备注
1	编号	摆弧文件编号，最多可以同时保存 100 组摆弧文件。	范围：0-99
2	摆弧基准	摆弧设置的基准，分为长度和周期，长度表示机器人周期运行的长度距离进行设置，周期表示机器人摆弧按周期运行频率进行设置。	目前只支持周期。
3	摆弧形状	摆弧类型是摆弧的轨迹，分以下四种： 2. 正弦波  3. 圆弧波  4. 8 字型	切换后可在示教器上查看对应波形形状。

		 5. 三角 	
4	摆弧长度/频率	机器人摆动一个周期的情况下，机器人运行的长度/时间。长度时，单位为 mm，范围：1-100。频率时，单位为 hz，范围：0.01-5。	焊接频率和左右振幅之和直接需要满足两者之积小于 50。不允许高频大振幅运动。
5	左/右振幅	指摆焊时，焊缝左边/右边的最大距离。单位：mm，范围：0.1-100。 焊缝左右区分：人面对焊缝前进方向，左手边为左边，右手边为右边。	
6	摆弧半径	设置圆弧波和 8 字形摆动时候，要设置摆弧半径。单位：mm	范围：0.1-100
7	等待类型	分为机器人停止和摆动停止，选择摆动停止，只有摆动停止，前进方向的运动不停止。如果选择机器人停止，则表示在停止时间处，摆动和前进运动都停止。  	
8	左侧/中心/右侧等	左侧/中心/右侧等待指的是在每个周期的	

	待	左侧、中心、右侧处摆弧停止的时间，当摆弧基准选择长度且等待类型选择摆动停止单位为 mm。其他情况单位是 s。此参数只有摆弧形状为正弦和三角有效，其中正弦不支持中间等待。  	
9	波形反向	当前波形按焊缝方向进行轴对称翻转。  	
10	仰角类型	仰角类型分为 V 型和三角，仅当摆弧形状为三角时候有效。 V 型：类似于将三角波形沿着示教的焊缝位置折叠。沿焊缝反向视图成 V 型.  三角：从焊缝位置摆动到左侧波峰点，然后摆动到右侧波峰点，再返回示教的焊缝位置。沿焊缝反向视图成三角形。	

			
11	左/右仰角	摆动的左/右平面与焊枪工具Z轴方向垂直平面的夹角。 	范围：-90-90； 当类型为 V 时候，角度设置成 0 度，机器人在单平面摆动。 当类型为三角时候，两个角度之和不能为 0、180、-180。
12	倾斜角	摆动的方向与前进方向的垂直方向的角度。即摆动方向上发生倾斜。 	范围：-90-90；当角度设置成 90 或者-90，机器人会变成前后摆动。

23.2.7 电弧跟踪

表 3-7 电弧跟踪界面介绍

步骤	图片	描述

1.配置“电弧跟踪”中的参数信息。

在弧焊主界面点击**“电弧跟踪”**按钮。

配置相关参数，配置完成后，点击“**保存**”，保存设置的参数。

序号	标题	描述	备注
1	编号	电弧跟踪文件编号，最多可以同时保存 20 组文件。	范围：0-19
2	信号源	使用电弧跟踪时候，采样焊接电流的信号来源。分为焊机反馈信号和利用跟踪采集信号，跟踪器数据可以设置十组。选择跟踪器，可以设置对应跟踪器的参数。	目前跟踪器只支持时代电弧跟踪器。
3	无效采样计数	在采集电弧数据前，为保证采集数据的准确，用于电弧稳定的周期数。	范围：3-10，建议 3。
4	采样延迟时间	焊接采样与接受到焊接反馈电流值直接的时间差，单位：s。麦格米特和奥太焊接时间约为 0.15s。	范围：0-10。
5	左右跟踪使能	左右跟踪功能的开关。关闭则不会跟踪。	
6	左右基准偏差	左右两个方向的基准偏差，即在焊接一开始就加上基准偏差。正数表示左边，负数表示右边，单位：mm。	范围：-100-100。
7	左右单次最大补偿量	左右方向上一次补偿的最大值，计算出偏移超过设定值，会以最大值跟踪。下一周期继续计算跟踪。单位：mm。	范围：0-10，建议 1-2。
8	左右累计最大补偿量	左右方向上所有补偿量最大值，即左右单方向最大偏移量的值，超过设置值，则不会跟踪补偿。单位：mm。	范围：0-200。
9	左右开始跟踪计数	左右方向上开始跟踪的周期数，建议在无效采样计数的基础上加 2-3 个周期。	范围：4-20，建议 5。
10	上下跟踪使能	上下跟踪功能的开关。关闭则不会跟踪。	
11	上下基准偏差	上下两个方向的基准偏差，即在焊接一开始就加上基准偏差。正数表示上方，负数表示下方，单位：mm。	范围：-100-100。

12	上下单次最大补偿量	上下方向上一次补偿的最大值，计算出偏移超过设定值，会以最大值跟踪。下一周期继续计算跟踪。单位：mm。	范围：0-10，建议 1-2。
13	上下累计最大补偿量	上下方向上所有补偿量最大值，即左右单方向最大偏移量的值，超过设置值，则不会跟踪补偿。单位：mm。	范围：0-200。
14	上下开始跟踪计数	上下方向上开始跟踪的周期数，建议在无效采样计数的基础上加 2-3 个周期。	范围：4-20，建议 5。

2. 设置跟踪器参数。

如果需要使用电弧跟踪器，则在信号源中选择跟踪器编号（目前跟踪器只支持时代电弧跟踪器）。

点击右方设置按钮，进入电弧跟踪器设置界面。

注意设置的为对应跟踪器编号的参数。

序号	标题	描述	备注
1	IP 地址	跟踪控制器的 IP 地址。注意机器人接口的 IP 地址和跟踪器地址处于同一个网段。	此参数为跟踪器公用参数，在任一个跟踪器参数中修改，所以参数中都会修改。
2	端口	跟踪控制器的端口号。	
3	频率	采样频率，建议值：250。	
4	跟踪器程序编号	范围为 0~10：0 代表机器人下发跟踪控制器的参数，1 代表调用跟踪控制器本地存储的程序编号为 1 的参数，以此类推。	
5	左右偏置	调整范围为-50~+50，数值 0 对应左右对称的坡口情况，其他数值对应坡口不对称情况。	
6	左右静态误差灵敏度	范围（0~20000），代表消除左右位置误差的能力，数值越小消除位置误差的能力越小，越大会造成焊缝震荡甚至直接跑出坡口区域。	
7	左右动态误差灵敏度	范围（0~20000），代表左右跟踪能力。数值越小，跟踪能力弱，可能越来越偏离。过大，容易超调震荡。	
8	焊丝干伸长	干伸长度，范围为-50~+50。	
9	上下跟踪灵敏度	范围（0~100），代表高度的跟踪能力。数值越小，跟踪能力弱，可能越来越偏离。过大，容易在高度调整上超调震荡。一般数值固定设置为 3。	
10	采样电流下限	电弧跟踪器采样电流窗口下限，范围	

		（0~90），如果设置为 60，认为采样电流低于实际平均电流的 60%时为无效数据。	
11	采样电流上限	电弧跟踪器采样电流窗口上限，范围（110~300），如果设置为 200，认为采样电流高于实际平均电流的 2 倍时为无效数据。	

23.2.8 接触寻位

表 3-7 电弧跟踪界面介绍

步骤	图片	描述
1.配置“**接触寻位**”中的参数信息。		在弧焊主界面点击**“接触寻位”**按钮。 配置相关参数，配置完成后，点击“**保存**”，保存设置的参数。

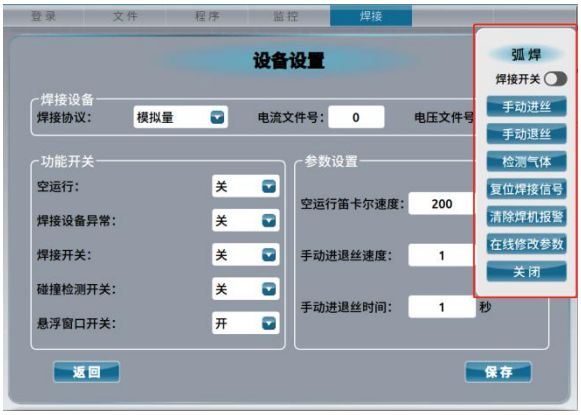
序号	标题	描述	备注
1	编号	接触寻位文件编号，最多可以同时保存 20 组文件。	范围：0-19
2	注释	客户可以添加注释信息，方便辨识。	
3	基准旗标	程序中工艺号对应的相关寻位点基准位置数据写入开关。开启时，寻位到的位置作为基准数据保存；关闭后，基准数据被保护，不再写入。 使用时，在第一次示教好的程序，基准旗标打开，记录基准位置。然后关闭基准旗标，后续程序运行与基准旗标对比，进行偏移。	如果基准旗标没关，对偏移工件进行了寻位，那么偏移工件上的寻位数据会保存，覆盖之前准确数据，则后续偏移都会失败，需要重新编写轨迹部分程序。
4	搜寻信号类型	接收搜寻到焊机反馈的寻位成功信号，分为总线和 IO，如果使用 IO 信号，需要配合 IO 功能配置使用。	数字量通讯焊机协议中具备此信号，可以选择使用。
5	信号沿	机器人检测寻位成功信号的信号沿类型。分为上升沿和下降沿。	总线一般都选择上升沿。
6	搜寻距离	从寻位开始点往工具方向的寻位距离，超过这个距离，系统将报警。	范围：0-200。
7	搜寻速度	从寻位开始点往工件移动寻位的速度，寻位速度越小，精度越高。	范围：0.01-200，建议设置 5mm/s

8	停止方式	机器人接收到寻位信号的停止方式。	
9	自动返回	自动返回开关。设置为开，寻位接触到工件后，机器人将参考自动返回距离和速度，沿之前运动路径返回。设置为关，接触到工件后暂停，按 start 继续。	建议选择开。
10	返回距离	设定自动返回距离，当这个距离超过寻位开始点时，机器人运动到寻位开始点就结束，不在运行。	范围：0-200。
11	返回速度	设定寻位时，寻到工件后的返回速度。	范围：0.01-200。
12	超偏差范围	计算结果的工件偏移量限制。当计算结果里面的偏差范围超过设置值，机器人报错。	范围：0-500。

23.2.9 悬浮窗口

要使用弧焊的悬浮窗口需要在弧焊 app 的设备设置中，将悬浮窗开关打开。不使用，则可以将其关闭。

表 3-8 弧焊悬浮窗口使用

步骤	图片	描述
1.打开悬浮窗功能开关。		在设备设置中悬浮窗口开关配置为**“开”**，然后点击**“保存”**按钮保存参数。 保存成功后，界面中会出现一个小机器人图标按钮，按钮可在界面上随意拖动。 注：当图标为彩色时候，焊接开关打开。当图标为灰色时候，焊接开关关闭。
2.弧焊悬浮窗口功能。		点击步骤 1 中的按钮，可以切换出弧焊悬浮窗口。

序号	标题	描述	备注

1	焊接开关	可以在这里快速切换焊接开关，焊接功能使能开关，只有将焊接开关打开，才能正常使用焊接功能。	在切换设备设置中的焊接开关，会保存相应文件，机器人如果重启，则以文件中为准。当前实际生效的焊接开关，可以在窗口中监控。
2	手动进丝	按下按钮，焊丝前进，松开停止。	进丝速度以焊机设置为准。
3	手动退丝	按下按钮，焊丝退后，松开停止。	退丝速度以焊机设置为准。
4	检测气体	按下按钮，打开焊接保护气。松开按钮，关闭焊接保护气。	
5	复位焊接信号	按下按钮，复位焊接信号，比如焊接或者摆弧中断，这时候需要将焊接信号复位。否则会影响弧焊程序再次启动。	
6	清除焊机报警	按下按钮，清除焊机的报警。	
7	在线修改参数	点击后弹窗在线修改参数窗口。	
8	关闭	隐藏当前窗口，显示悬浮按钮。	

3.在线修改参数		在线修改是指在焊接过程中，修改焊接参数，现在支持修改电流和电压参数。 先设置调节的步长，然后点击“+”或“-”可以在当前焊接参数上进行设置步长的调整。参考值中显示的是经过调整后的焊接参数。 当工作模式为一元化时，电压最小步长为 1，调节的是焊接电压的百分比；当工作模式为分别时，电压最小步长为 0.1，调节的是焊接电压的值。

23.3 指令说明

弧焊指令，由标准的 RPL 语句组成，并且做了封装。使用时，通过 call 指令，调用弧焊指令即可。常用弧焊指令的内容及使用方法如下：

表 3-9 弧焊指令说明

序号	函数名	说明
1	ArcOn	开始起弧。fileNum：所使用的起弧参数文件号(0-99)
2	ArcOff	收弧指令
3	GradOn	设置电流/电压渐变开始。type：渐变模式，0-电压渐变，1-电流渐变；stratValue：渐变的起始值；endValue：渐变的终点值
4	GradOff	关闭电流电压渐变功能
5	SetWeldingPar	设置焊接参数。current：设置电流，voltage：设置电压
6	SetArcSpeed	设置焊接速度。inWeldVelocty：焊接速度，单位：mm/s。该函数必须配合运动指令的变量 arcweld.speed
7	IntermittentOn	间断焊开始。type：0 为点焊，1 为无摆弧间断焊，2 为带摆弧间断焊。t_11：点焊输入时间或间断焊输入距离(>0)。12：空走距离(>0)。fileNum：焊接文件号（0-99）。weaveFile：摆弧文件号（0-99）
8	IntermittentOff	间断焊结束
9	FeedOnWire	送丝指令。time：进丝时间
10	FeedBackWire	退丝指令。time：退丝时间
11	DetectGas	检气指令。time：检气时间
12	WeaveOn	摆弧开始。fileNum：所使用的摆弧参数文件号（0-99）
13	WeaveOff	摆弧结束
14	ArcTrackOn	打开电弧跟踪。fileNum：所使用的电弧跟踪参数文件号（0-19）
15	ArcTrackOff	关闭电弧跟踪
16	ResetVar	焊接信号复位，通常该指令放在第一行
17	DWell	等待时间。time：等待时间，单位：s
18	OpenLaser	连接并打开激光器
19	CloseLaser	关闭激光器并断开连接
20	LaserSearchOn	开启激光寻位。LaserCalibNum：激光标定文件号（0-19）。LaserSearchNum：激光寻位文件号（0-19）
21	LaserSearchOff	关闭激光寻位，并计算位置。LaserSearchSaveIndex：寻位点序号(1-5)
22	LaserTrackOn	开启激光跟踪。LaserCalibNum：激光标定文件号（0-19）。LaserTrackParNum：激光跟踪文件号（0-19）
23	LaserTrackOff	关闭激光跟踪

24	calSearchFrame	寻位坐标系计算
25	PC_Initialize	功能初始化
26	TouchSearchStart	调用寻位工艺号（0-19）
27	TouchSearch	接触寻位开始。toolC：工具坐标系。refsysC：用户坐标系。dir：设置寻位方向（±X，±Y，±Z）。searchingId 寻位点序号（0-6），单次寻位按顺序填写
28	TouchSearchEnd	接触寻位结束
29	CalculateOffset	计算寻位偏移，searchmode：焊接模式 0 为角焊接，1 为内外径。searchtype：焊接类型分为 1D,2D,3D,2D+,3D+。offsetid：存储偏移量序号（0~39）
30	OffsetStart	寻位偏移开始。offsetid：存储偏移量序号（0~39）
31	OffsetEnd	寻位偏移结束
32	PreArcOn	提前开始起弧，使用此指令，前一句移动指令的圆滑过渡不能为 fine。fileNum：所使用的起弧参数文件号(0-99)

23.4 功能使用介绍

23.4.1 基本功能

以完整的焊接程序说明焊接指令的具体使用方法。假设需要焊接一条直线，机器人的运动路径如图 3-2 所示。

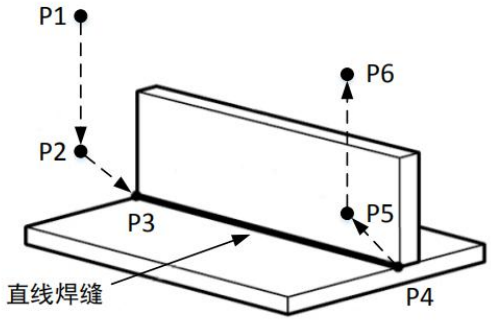


图 3-2 机器人路径示意图

表 3-10 程序举例说明

序号	程序	说明
1	arcweld.ResetVar();	变量初始化
2	MJOINT(P1, v500, fine, tool0, wobj0);	焊机安全点P1，通常先以关节运动的形式运动到该点。
3	MLIN(P2, v200, fine, tool0, wobj0);	焊接起点上方P2。
4	MLIN(P3, v200, fine, tool0, wobj0);	焊接起点P3。

5	WAIT_POS();	保证机器人运动到P3点
6	arcweld.WeaveOn(1);	打开摆弧功能，并且以1号摆弧参数设置摆弧。
7	arcweld.GradOn(1, 100, 150);	打开渐变功能，电流渐变模式，渐变值从100A渐变到150A。
8	arcweld.ArcOn(1);	开始起弧，使用1号起弧参数。
9	MLIN(P4, arcweld.Speed, fine, tool0, wobj0);	运行焊缝路径，焊缝终点P4，焊接速度以arcweld.Speed变量表示。
10	WAIT_POS();	保证机器人运动到P4点
11	arcweld.ArcOff();	收弧。
12	arcweld.GradOff();	关闭渐变功能。
13	arcweld.WeaveOff();	关闭摆弧功能。
14	MLIN(P5, v200, fine, tool0, wobj0);	运动到焊缝终点上方点P5。
15	MLIN(P6, v200, fine, tool0, wobj0);	运动到焊机安全点P6。

注意事项：

1) 起弧与收弧指令需要成对使用；摆弧打开与摆弧关闭指令需要成对使用；电流电压渐变的打开与关闭指令需要成对使用。

2) 在执行焊接功能的指令之前，需要用 WAIT_POS()指令，保证机器人准确运动到预定的点位，否则，若运动指令当中设置了过渡半径 zone 后，机器人还没有运动到指定点，将提前触发该指令之后的指令。

23.4.2 间断焊

间断焊指令需要单独成对使用，并且间断焊开始指令中输入相应参数。

对于一条焊缝，焊接中需要不断起弧、收弧。该功能采用的方式为反复读取模板作业，将焊缝分为若干段，每段包含一个焊接距离和空程距离，循环调用模板作业进行不停的起弧、收弧操作。若将间断焊的焊接距离设置为零，并且设置焊接时间，该功能也就成为点焊。

图 3-5 点焊

图 3-6 间断焊

具体间断焊的使用方法如下：

表 3-11 间断焊举例说明

1	arcweld.ResetVar();	变量初始化
2	MJOINT(P1, v500, fine, tool0, wobj0);	焊机安全点P1，通常先以关节运动的形式运动到该点。
3	MLIN(P2, v200, fine, tool0, wobj0);	起弧点位置
4	arcweld.IntermittentOn(0, 1, 3, 1);	开始间断焊功能。

		间断焊类型：点焊；点焊时间1秒，空程距离3mm，使用1号焊接参数。
5	MLIN(P3, arcweld.Speed, fine, tool0, wobj0);	运行焊缝路径，收弧点，焊接速度以arcweld.Speed变量表示。
6	WAIT_POS();	保证机器人运动到P4点
7	arcweld.IntermittentOff();	关闭间断焊功能。
8	MLIN(P4, v200, fine, tool0, wobj0);	运动到焊缝终点上方点P4。

注意：间断焊功能必须单独使用，无需使用 ArcOn 指令，不能与摆动同时使用。

23.4.3 电弧跟踪编程

电弧跟踪在使用过程中存在一定限制：

1) 电弧跟踪的电流需要在 200A 以上。

2) 电弧跟踪必须配合摆弧功能一起使用。

3) 电弧跟踪使用必须是角焊缝或者带坡口的焊缝，两个母材平面角度需要在 30 度-120 度，过大过小都可能造成跟踪不准确。

4) 跟踪过程中，母材平面之间的角度变化不超过 10 度。

电弧跟踪在使用前，先配置好相关参数，然后编程即可使用，具体间断焊的使用方法如下：

表 3-12 电弧跟踪举例说明

序号	程序	说明
1	arcweld.ResetVar();	变量初始化
2	MJOINT(P1, v500, fine, tool0, wobj0);	焊机安全点P1，通常先以关节运动的形式运动到该点。
3	MLIN(P2, v200, fine, tool0, wobj0);	焊接起点上方P2。
4	MLIN(P3, v200, fine, tool0, wobj0);	焊接起点P3。
5	arcweld.WeaveOn(1);	打开摆弧功能，并且以1号摆弧参数设置摆弧。
6	arcweld.SeamTrackOn(1);	打开电弧跟踪功能，并且以1号参数设置跟踪。
7	arcweld.ArcOn(1);	开始起弧，使用1号起弧参数。
8	MLIN(P4, arcweld.Speed, fine, tool0, wobj0);	运行焊缝路径，焊缝终点P4，焊接速度以arcweld.Speed变量表示。
9	arcweld.ArcOff();	收弧。
10	arcweld.SeamTrackOn();	关闭电弧跟踪。
11	arcweld.WeaveOff();	关闭摆弧功能。
12	MLIN(P5, v200, fine, tool0, wobj0);	运动到焊缝终点上方点P5。
13	MLIN(P6, v200, fine, tool0, wobj0);	运动到焊机安全点P6。

23.4.4 接触寻位使用方法

23.4.4.1 接触寻位原理介绍

焊丝接触寻位的原理：首次安装工件后，打开旗标，机器人按照寻位程序进行寻位，当焊丝碰

到工件后，焊机或寻位装置发给机器人信号，机器人停止运动，记录下基础位置，后面更换同种产品后，关闭旗标，进行寻位，条件满足后，记录下当前位置；根据当前位置和基准位置计算出偏差值，实现焊接轨迹基于基准轨迹的偏移。

23.4.4.2 焊丝接触寻位应用场景

寻位需满足以下条件：

1. 工件一致性高，只有安装位置的偏差；

2. 示教准确的基准焊接轨迹；

3. 提前获取工件放置的偏差方向，用于确定使用哪种寻位；

4. 工件导电性良好；

5. 目前暂支持角焊缝 1D、2D、3D、2D+、3D+寻位方式。

6. 焊机需支持寻位功能或有外部寻位装置。

23.4.4.3 接触寻位方式使用及示例

目前接触接触寻位暂支持角焊缝寻位，包含 1D、2D、3D、2D+、3D+。

1.1D 寻位

应用场景：工件安放位置较基准位置只有一个方向上的位置偏移。

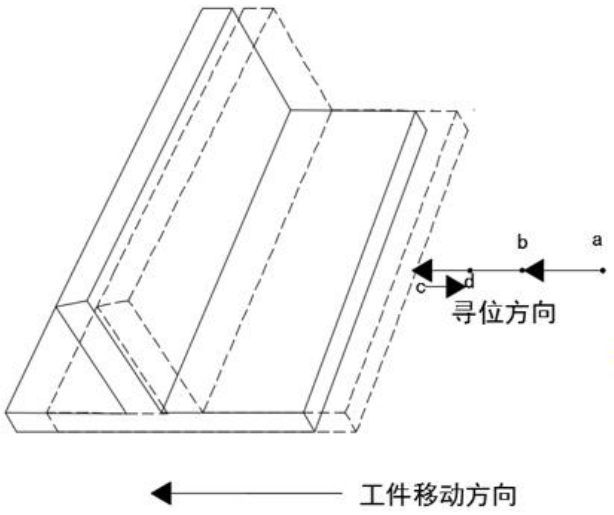


图 3-7 1D 寻位

操作流程：操作流程：在一个方向确定 1 个寻位点（P0）；编写寻位程序是寻位点严格按照这个顺序进行。

表 3-14 1D 程序举例说明

序号	程序	说明
1	MJOINT (*, v50perc, fine, tool1) ;	机器人起始位置
2	TouchSearchStart(0) ;	开启寻位，工艺号0（可设置0~19）
3	MLIN (*, v500, fine, tool1) ;	P0寻位起始点
4	TouchSearch(tool1, wobj0, "-Z", 0) ;	P0寻位点，沿-Z方向寻位
5	MLIN (*, v500, fine, tool1) ;	过渡点

6	TouchSearchEnd() ;	寻位结束
7	CalculateOffset(0, "1D", 0) ;	计算偏移量； 参数1：默认设置为0（角焊缝）； 参数2：寻位方式设置为3D+； 参数3：偏置保存的索引（0~39）
8	OffsetStart(0) ;	偏置开始；
9	MLIN (*, v500, fine, tool1) ;	焊接准备点
10	MLIN (*, v500, fine, tool1) ;	焊接起始点
11	MLIN (*, v500, fine, tool1) ;	焊接结束点
12	MLIN (*, v500, fine, tool1) ;	焊接离开点
13	OffsetEnd() ;	偏置结束

2.2D 寻位

应用场景：工件安放位置较基准位置仅在两个方向上的位置偏移。

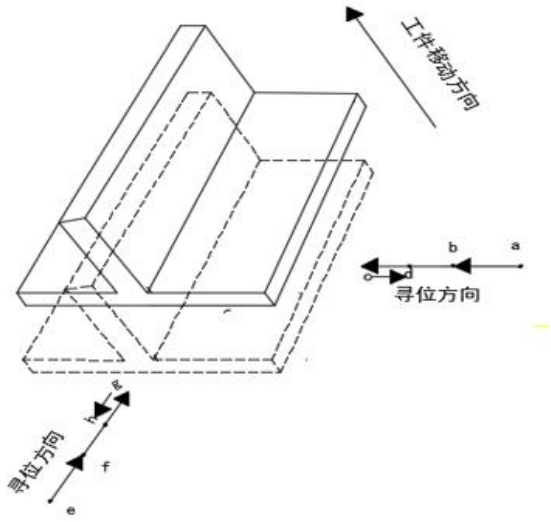


图 3-8 2D 寻位

操作流程：操作流程：在一个方向确定 1 个寻位点（P0）；在另一个方向确定 1 个寻位点（P1）；编写寻位程序是寻位点严格按照这个顺序进行。

表 3-15 2D 举例说明

序号	程序	说明
1	MJOINT (*, v50perc, fine, tool1) ;	机器人起始位置
2	TouchSearchStart(0) ;	开启寻位，工艺号0（可设置0~19）
3	MLIN (*, v500, fine, tool1) ;	P0寻位起始点
4	TouchSearch(tool1, wobj0, "-Z", 0) ;	P0寻位点，沿-Z方向寻位
5	MLIN (*, v500, fine, tool1) ;	过渡点（根据实际需要添加）
6	MLIN (*, v500, fine, tool1) ;	P1寻位起始点
7	TouchSearch(tool1, wobj0, "+X", 1) ;	P1寻位点，沿+X方向寻位
8	MLIN (*, v500, fine, tool1) ;	过渡点（根据实际需要添加）

9	TouchSearchEnd() ;	寻位结束
10	CalculateOffset(0, "2D", 0) ;	计算偏移量； 参数1：默认设置为0（角焊缝）； 参数2：寻位方式设置为3D+； 参数3：偏置保存的索引（0~39）
11	OffsetStart(0) ;	偏置开始；
12	MLIN (*, v500, fine, tool1) ;	焊接准备点
13	MLIN (*, v500, fine, tool1) ;	焊接起始点
14	MLIN (*, v500, fine, tool1) ;	焊接结束点
15	MLIN (*, v500, fine, tool1) ;	焊接离开点
16	OffsetEnd() ;	偏置结束

3.3D 寻位

应用场景：工件安放位置较基准位置仅在三个方向上位置偏移。

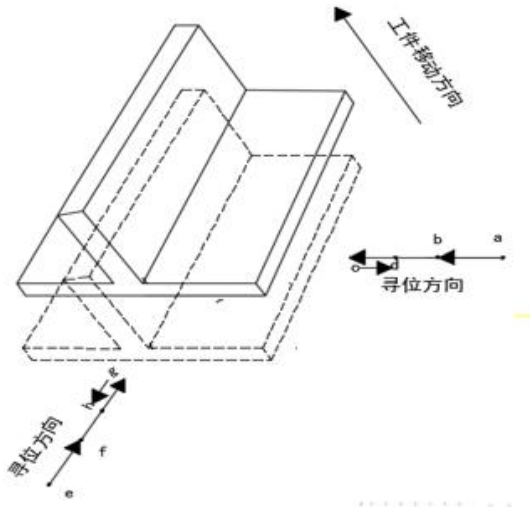


图 3-9 3D 寻位

操作流程：操作流程：在一个方向确定 1 个寻位点（P0）；在另一个方向确定 1 个寻位点（P1）；在剩余一个方向确定 1 个寻位点（P2）；编写寻位程序是寻位点严格按照这个顺序进行。

表 3-16 3D 程序举例说明

序号	程序	说明
1	MJOINT (*, v50perc, fine, tool1) ;	机器人起始位置
2	TouchSearchStart(0) ;	开启寻位，工艺号0（可设置0~19）
3	MLIN (*, v500, fine, tool1) ;	P0寻位起始点
4	TouchSearch(tool1, wobj0, "-Z", 0) ;	P0寻位点，沿-Z方向寻位
5	MLIN (*, v500, fine, tool1) ;	过渡点（根据实际需要添加）
6	MLIN (*, v500, fine, tool1) ;	P1寻位起始点
7	TouchSearch(tool1, wobj0, "+X", 1) ;	P1寻位点，沿+X方向寻位
8	MLIN (*, v500, fine, tool1) ;	过渡点（根据实际需要添加）

9	MLIN (*, v500, fine, tool1) ;	P2寻位起始点
10	TouchSearch(tool1, wobj0, "-Y", 2) ;	P2寻位点，沿-Y方向寻位
11	MLIN (*, v500, fine, tool1) ;	过渡点
12	TouchSearchEnd() ;	寻位结束
13	CalculateOffset(0, "3D", 0) ;	计算偏移量； 参数1：默认设置为0（角焊缝）； 参数2：寻位方式设置为3D+； 参数3：偏置保存的索引（0~39）
14	OffsetStart(0) ;	偏置开始；
15	MLIN (*, v500, fine, tool1) ;	焊接准备点
16	MLIN (*, v500, fine, tool1) ;	焊接起始点
17	MLIN (*, v500, fine, tool1) ;	焊接结束点
18	MLIN (*, v500, fine, tool1) ;	焊接离开点
19	OffsetEnd() ;	偏置结束

4.2D+寻位

应用场景：绕工件上的 X、Y、Z 任意 1 个轴（或机器人坐标系）旋转和任意 2 个方向的移动。

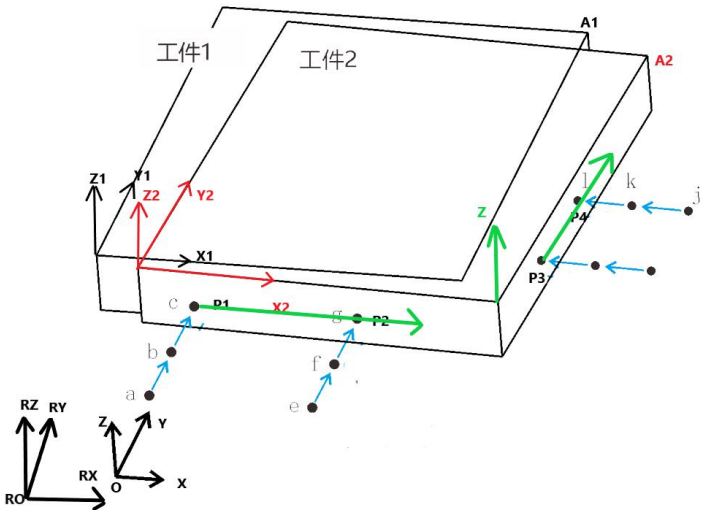


图 3-10 2D+寻位

操作流程：在一个方向确定 2 个点（P0、P1）确定一条线；在另一个方向确定 2 个点（P2、P3）确定另一条线；编写寻位程序是寻位点严格按照这个顺序进行。

表 3-17 2D+举例说明

序号	程序	说明
1	MJOINT (*, v50perc, fine, tool1) ;	机器人起始位置
2	TouchSearchStart(0) ;	开启寻位，工艺号0（可设置0~19）
3	MLIN (*, v500, fine, tool1) ;	P0寻位起始点
4	TouchSearch(tool1, wobj0, "+X", 0) ;	P0寻位点，沿+X方向寻位
5	MLIN (*, v500, fine, tool1) ;	P1寻位起始点

6	TouchSearch(tool1, wobj0, "+X", 1) ;	P1寻位点，沿+X方向寻位
7	MLIN (*, v500, fine, tool1) ;	过渡点（根据实际需要添加）
8	MLIN (*, v500, fine, tool1) ;	过渡点（根据实际需要添加）
9	MLIN (*, v500, fine, tool1) ;	P2寻位起始点
10	TouchSearch(tool1, wobj0, "-Y", 2) ;	P2寻位点，沿-Y方向寻位
11	MLIN (*, v500, fine, tool1) ;	P3寻位起始点
12	TouchSearch(tool1, wobj0, "-Y", 3) ;	P3寻位点，沿-Y方向寻位
13	MLIN (*, v500, fine, tool1) ;	过渡点
14	TouchSearchEnd() ;	寻位结束
15	CalculateOffset(0, "2D+", 0) ;	计算偏移量； 参数1：默认设置为0（角焊缝）； 参数2：寻位方式设置为3D+； 参数3：偏置保存的索引（0~39）
16	OffsetStart(0) ;	偏置开始；
17	MLIN (*, v500, fine, tool1) ;	焊接准备点
18	MLIN (*, v500, fine, tool1) ;	焊接起始点
19	MLIN (*, v500, fine, tool1) ;	焊接结束点
20	MLIN (*, v500, fine, tool1) ;	焊接离开点
21	OffsetEnd() ;	偏置结束

5.3D+寻位

应用场景：绕工件上的 X、Y、Z 任意旋转（或机器人坐标系）和 3 方向的移动。

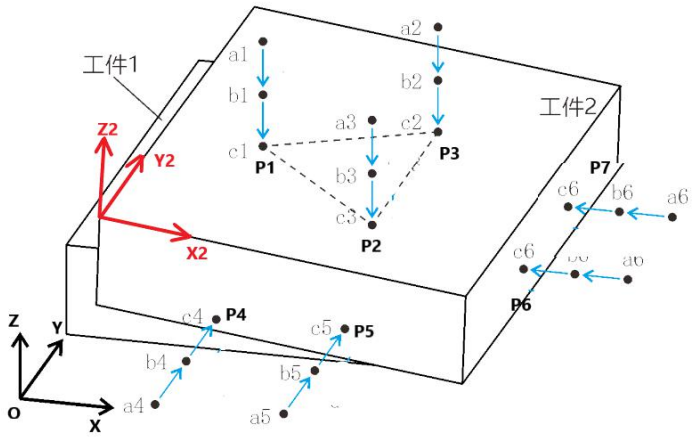


图 3-11 3D+寻位

操作流程：在工件平面的 1 个方向找 3 个不共线的点（P1、P2、P3）确认一个面；在另一个方

向确定 2 个点（P4、P5）确定一条线；在最后一个方向确定 2 个点（P6、P7）确定另一条线；编写寻位程序是寻位点严格按照这个顺序进行。

举例说明 3D+寻位程序，如下表

表 3-18 3D+举例说明

序号	程序	说明
1	MJOINT (*, v50perc, fine, tool1) ;	机器人起始位置
2	TouchSearchStart(0) ;	开启寻位，工艺号0（可设置0~19）
3	MLIN (*, v500, fine, tool1) ;	P0寻位起始点
4	TouchSearch(tool1, wobj0, "-Z", 0) ;	P0寻位点，沿-Z方向寻位
5	MLIN (*, v500, fine, tool1) ;	P1寻位起始点
6	TouchSearch(tool1, wobj0, "-Z", 1) ;	P1寻位点，沿-Z方向寻位
7	MLIN (*, v500, fine, tool1) ;	P2寻位起始点
8	TouchSearch(tool1, wobj0, "-Z", 2) ;	P2寻位点，沿-Z方向寻位
9	MLIN (*, v500, fine, tool1) ;	过渡点（根据实际需要添加）
10	MLIN (*, v500, fine, tool1) ;	P3寻位起始点
11	TouchSearch(tool1, wobj0, "+X", 3) ;	P3寻位点，沿+X方向寻位
12	MLIN (*, v500, fine, tool1) ;	P4寻位起始点
13	TouchSearch(tool1, wobj0, "+X", 4) ;	P4寻位点，沿+X方向寻位
14	MLIN (*, v500, fine, tool1) ;	过渡点（根据实际需要添加）
15	MLIN (*, v500, fine, tool1) ;	过渡点（根据实际需要添加）
16	MLIN (*, v500, fine, tool1) ;	P5寻位起始点
17	TouchSearch(tool1, wobj0, "-Y", 5) ;	P5寻位点，沿-Y方向寻位
18	MLIN (*, v500, fine, tool1) ;	P6寻位起始点
19	TouchSearch(tool1, wobj0, "-Y", 6) ;	P6寻位点，沿-Y方向寻位
20	MLIN (*, v500, fine, tool1) ;	过渡点
21	TouchSearchEnd() ;	寻位结束
22	CalculateOffset(0, "3D+", 0) ;	计算偏移量； 参数1：默认设置为0（角焊缝）； 参数2：寻位方式设置为3D+； 参数3：偏置保存的索引（0~39）
23	OffsetStart(0) ;	偏置开始；
24	MLIN (*, v500, fine, tool1) ;	焊接准备点
25	MLIN (*, v500, fine, tool1) ;	焊接起始点
26	MLIN (*, v500, fine, tool1) ;	焊接结束点
27	MLIN (*, v500, fine, tool1) ;	焊接离开点
28	OffsetEnd() ;	偏置结束

第 24 章　变位机

24.1 本章简介

变位机系统是机器人系统插补轴之外的辅助轴所组成的附加插补系统，辅助轴系统在与机器人插补轴进行同步插补运动的同时，还能保证机器人末端在辅助轴坐标系统下的精确轨迹。本章主要介绍附加轴的标定和使用。**在使用变位机功能前，需要先配置附加轴信息，要求附加轴配置为同步非插补轴，**详细配置方法见**“附加轴”**章节。

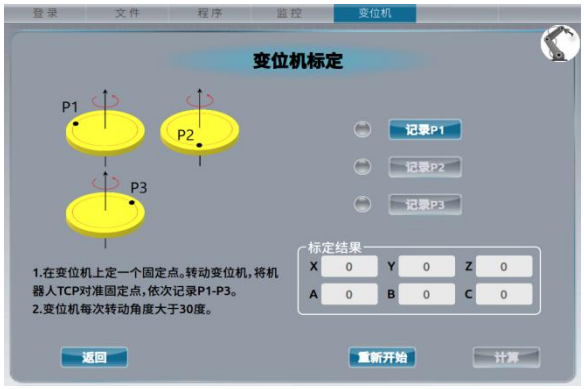
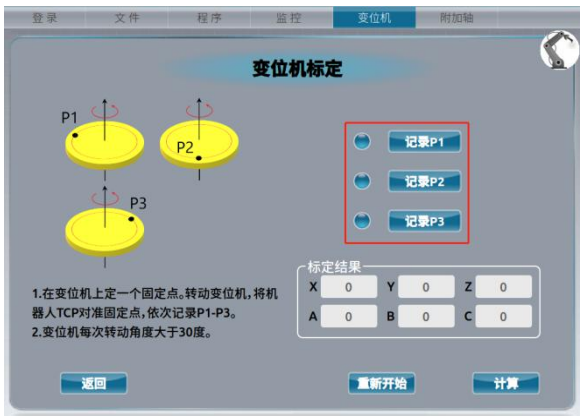
24.2 变位机标定

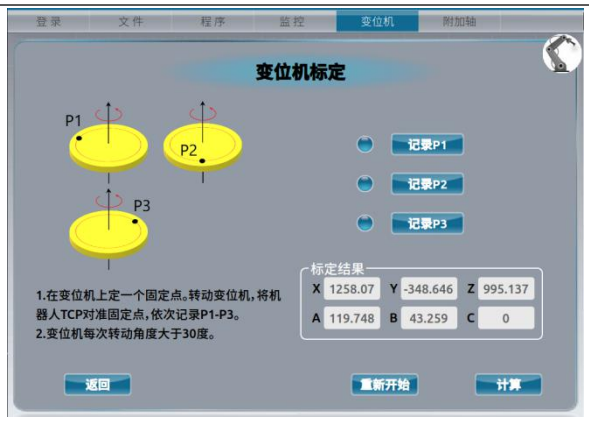
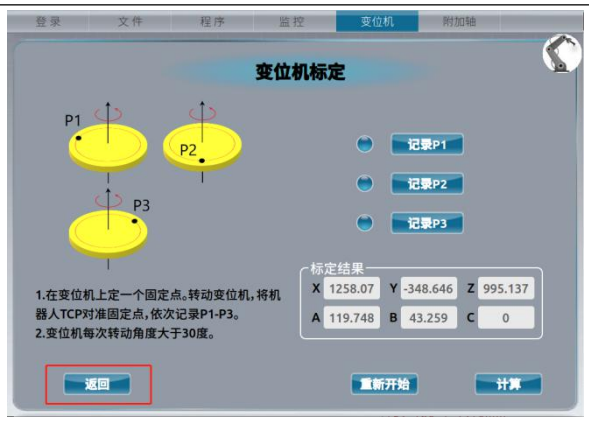
变位机系统在使用前必须先进行标定，标定分为两个部分：一是变位机轴的标定；二是随动的用户坐标系的标定，用户坐标在实际使用过程中也可以不进行标定，但是要保证数据全为 0,则默认用户坐标系在变位机中心位置。

24.2.1 变位机轴的标定

表 23-1 变位机轴标定步骤

步骤	图片	描述
1. 进入变位机的功能包。		打开示教器桌面，点击**“变位机”**功能图标进入主界面。
2. 打开变位机功能开关。		如果变位机功能开关处于关闭状态，需要将**“变位机功能开关”**打开。

3. 选择标定的变位机序号和变位机类型。		变位机序号：分为1-10，最多可以存储 10 组变位机数据。 变位机类型分为：无，即为不使用；旋转轴。 附加轴序号：选择对应的附加轴编号。目前分为附加轴 1 到附加轴 4，选择对应附加轴前，必须在附加轴 app 中配置附加轴参数。
4. 进入标定界面。		点击界面上**“标定”**按钮，即可进入变位机标定界面。
5. 记录点标定点位置。	 	在变位机上选择一个固定点，固定点会随变位机旋转而旋转。 将机器人的工具末端 TCP 点移动到固定点位置，点击**“记录 P1”**按钮，记录为 P1 点。 移开机器人，然后旋转变位机一定的角度，建议旋转角度大于 30 度，使得固定点旋转到 P2 位置，将机器人工具末端 TCP 移动到 P2 点，点击**“记录 P2”**按钮，记录为 P2 点。 同记录 P2 点的方式，记录 P3 点。 如果需要重新记录一个点，则再次点击记录按钮，即可覆盖原来记录的

		点位。 注意：记录 P1-P3 点过程中，必须同一方向旋转变位机。
6. 计算结果。		点击**“计算”**按钮，标定结果会显示在标定结果栏。如果不满意，可以点击**“重新开始”**按钮，然后重新开始标定。
7. 返回主界面。	 	点击**“返回”**按钮，返回主界面，标定结果会临时存储，并且在主界面显示。 如果需要保存，点击**“保存”**按钮，也可等用户坐标系标定完成后，一起保存。

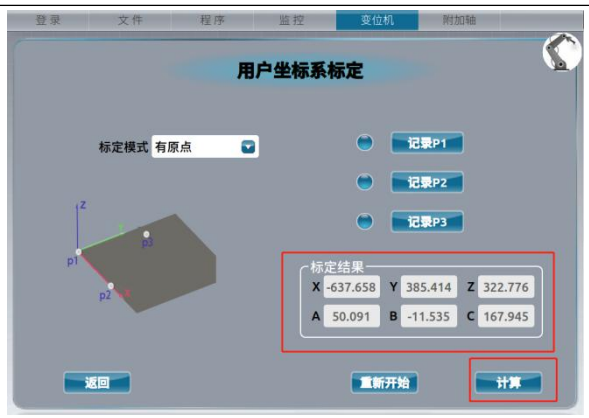
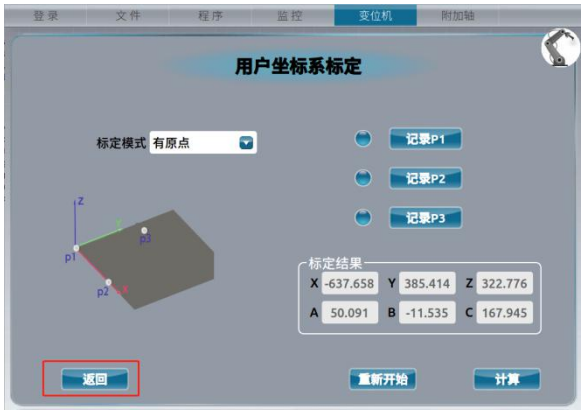
8. 双轴变位机标定。		双轴变位机的标定只需要配置好变位机类型和附加轴序号，然后点击对应“**标定**”按钮。 标定方法单轴变位机标定方法相同，注意在标定变位机轴 2 时候，变位机轴 1 对应的附加轴不能发生运动。

24.2.2 用户坐标系的标定

变位机上用户坐标系标定和正常的用户坐标系标定方法相同。参考步骤如下：

表 23-2 用户坐标系标定步骤

步骤	图片	描述
1. 在标定完变位机后，再标定用户坐标系。		点击用户坐标系标定栏的“**标定**”按钮，进入用户坐标系标定界面。
2. 选择标定模式。		标定模式分为： 有原点：标定 P1 点为原点，P2 为 x 轴正方向上一点，P3 为 xy 平面上一点，且在 y 轴正向平面上。

3. 依次记录点位信息。		**在示教三个点的过程中，变位机必须保持同一位置。**
4. 计算结果		点击**“计算”**按钮，标定结果会显示在标定结果栏。如果不满意，可以点击**“重新开始”**按钮，然后重新开始标定。
5. 返回主界面。		点击**“返回”**按钮，返回主界面，标定结果会临时存储，并且在主界面显示。
6. 保存数据。		点击**“保存”**按钮，将保存变位机标定数据和用户坐标系数据。 如果不保存数据，可以点击**“还原”**按钮，数据会恢复成上一次保存的数据。

24.2.3 标定数据手动修改

表 23-3 标定数据手动修改步骤

步骤	图片	描述
1. 进入手动编辑模式。		点击**“编辑”**按钮，进入手动修改模式，
2. 修改数据。		编辑情况下，可以修改变位机和用户坐标系的数据。未使用的变位机数据不可以修改。
3. 修改数据后结束编辑并保存。		修改完需要修改的数据后，点击“**结束编辑**”按钮，机器人会提示相应对话框，点击“**是**”，则将修改后的数据进行保存；点击“**否**”，则退出编辑模式，不保存修改数据。

24.3 变位机编程使用

24.3.1 变位机激活

变位机在编程使用之前，需要进行相应的激活操作。其步骤如下：

表 23-4 变位机激活步骤

步骤	图片	描述
1. 在变位机应用中激活。		在变位机主界面中，先选择需要激活的变位机序号，然后点击**“激活”**按钮。
2. 激活确认。		在变位机序号旁边可以查看当前已经激活的变位机序号，如果变位机序号为 0，则表示当前无激活的变位机。 变位机激活之后，对应的用户坐标系为“wobj_dyn0”。当此用户坐标系处于激活状态，运动变位机轴时，机器人会随动，以保证机器人相对变位机上用户坐标系位置不发生变化。
3. 状态栏快速切换用户坐标系。		点击状态栏上用户坐标系显示名称，会出现相应的用户坐标系，可以点击相应用户坐标系进行切换。需要注意以下几点: 1. 开机第一次激活变位机的用户坐标系，必须到变位机应用中激活，切换成其他用户坐标系之后，方可用状态栏激活的方式。 2. 当变位机激活后，切换到其他用户坐标系，比如 wobj1，这时变位机还是处于激活状态，但机器人不会与变位机随动。

		3. 状态栏激活变位机的用户坐标系，只能激活当前已激活的变位机上标定的坐标系，不能切换变位机序号。

24.3.2 变位机编程使用

带变位机编程指令和正常运动指令相同，采用 Epointc 变量记录点位信息，如果机器人要执行相对变位机上的精准轨迹，需要先将变位机激活，同时需要激活 wobj_dyn0 的用户坐标系，记录当前坐标系下的位置点。

激活变位机函数指令为：userframedyn.activeWobjDyn（int positionerID），此函数需要用 Call（）指令调用。参数 int positionerID 为变位机 ID。注意输入值超出变位机序号范围（1-10）。在示教带变位机轨迹的程序之前需要加上此指令，用于选择程序中运行的变位机序号，如图 5-1 所示。

如果机器人执行的轨迹与变位机系统无关，即变位机系统与机器人系统各自独立运动，只需要切换当前用户坐标系不为 wobj_dyn0 即可。

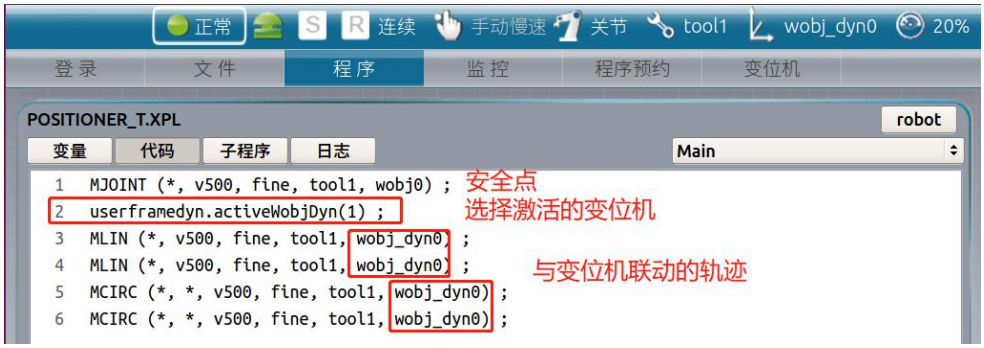


图 23-1 带变位机插补的精准轨迹运动编程

在程序中可以实现变位机序号的切换，或者两个程序使用不同的变位机。使用变位机激活函数指令切换变位机序号。在示教轨迹之前需要在变位机 app 中激活对应变位机，然后记录相应位置点。如图 5-2 所示。

图 23-2 激活变位机序号切换

第 25 章　程序预约

25.1 本章简介

程序预约是指用户通过外部 IO 端口或者总线，目前支持 Modbus-TCP、Profibus-DP 和 Profinet（需要转接模块）总线。通过每个工位上的启动按钮，机器人按照预约的顺序运行各个工位上程序。本章主要介绍埃夫特 ER 系列机器人程序预约的配置及使用。

25.2 程序预约配置及状态查看

程序预约功能配置使用之前，需要先新建出需要预约的程序，如图 6-1 所示，本文中以 TEST1、TEST2、TEST3、TEST4 为例。

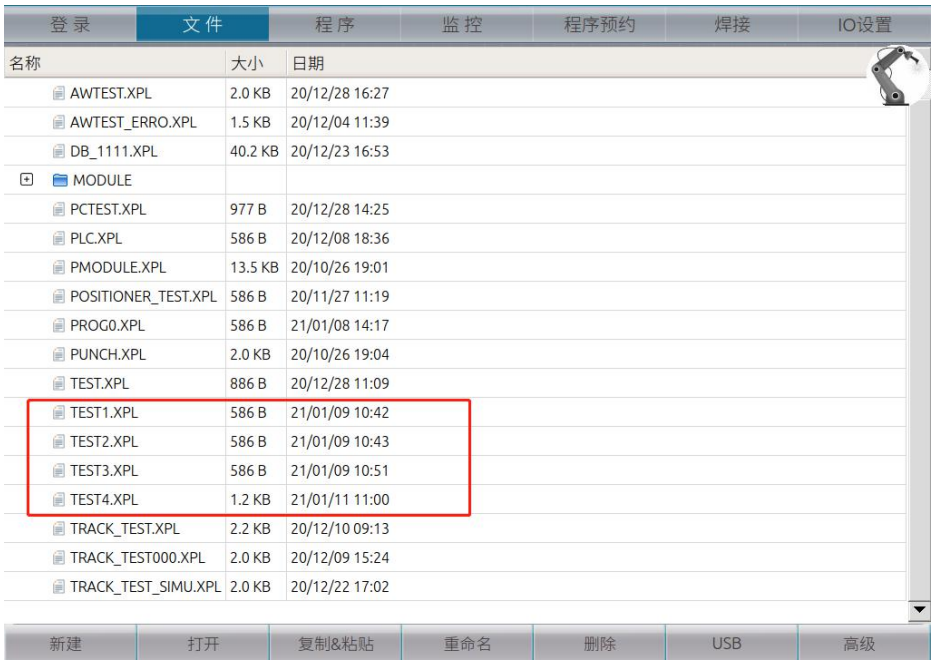


图 24-1 程序预约的程序

表 24-1 程序预约配置及状态查看步骤

步骤	图片	描述
1. 进入程序预约的功能包。		打开示教器桌面，点击“**程序预约**”功能图标进入主界面。

2. 打开程序预约功能开关。		如果程序预约的功能开关处于关闭状态，需要将“**功能开关**”打开。
3. 进入设置界面配置程序预约信息。		
4. 选择模式和协议。	 	模式分为：单独和二进制。单独模式通过 4 个输入端口分别可以关联 4 个示教文件。二进制模式通过 4 个输入端口的二进制组合最多可以关联 15 个示教文件。详细信息见程序预约 IO 口配置中相关说明。 协议分为：IO、Modbus-TCP、Profibus-DP 和 Profinet。协议中使用的信号相同，详细信息见程序预约 IO 口配置中相关说明。

5. 配置预约程序。		在名称栏选择预约的程序即可，名称栏会自动更新程序文件列表中所有的文件。 循环次数为接收到一次预约信号，程序运行的次数，最少为 1 次。
6. 保存参数。		点击**“保存”**按钮，将设置程序保存到文件中。 如果想恢复成原来文件中的数据，点击**“还原”**按钮。
7. 开始运行。	 	配置好参数后，选择机器人模式为自动，打开伺服。 然后回到程序预约首页，点击**“开始”**按钮，这时候指示灯会编程绿色。再次点击，会停止程序预约，注意，停止程序预约，机器人依旧会将当前正在运行的程序运行完成。之后已经预约的不再执行。 然后可以通过界面、IO 或总线，进行预约操作，一旦预约成功，机器人会自动加载程序运行。

8. 返回首界面查看状态。

首界面上方可以查看配置好的模式和协议。

首界面下方可以查看各个预约程序的状态，以及可以在示教器进行预约和取消。

名称栏显示各编号对应的程序名称，如果设置程序后，再将程序删除，则会在名称后面显示**“（无文件）”**。

状态：分为已预约、未预约、运行中。

排序：按预约顺序显示数字，0 无效，从 1 开始往后依次排序，表示其运行的顺序。

总数：表示程序运行总次数。如果在程序预约设置中循环次数为 3，那么预约一次，总数会加 3。如果预约的程序发生改变，总数会清零。

预约：点击**“预约”**按钮，则预约当前按钮对应程序，状态为运行中或者已预约，按钮无效。

取消预约：点击**“取消”**按钮，则取消预约当前按钮对应程序，未预约，按钮无效，如果是正在运行的程序，则机器人立即停止，并清除当前预约状态。

取消全部预约：点击**“取消全部预约”**按钮，将状态为已预约的程序取消预约。正在运行的程序正常运行。

9. 清除总数。		点击**“清除总数”**按钮，则总数列中计数都清零。

25.3 程序预约的信号配置及使用

25.3.1 IO 信号配置

程序预约如果使用 IO 控制，则需要在 IO 功能中配置相应的 IO 口，如果使用总线控制，则无需配置，根据地址表直接使用即可，相应地址表可以在《埃夫特 ER 系列机器人操作手册》的现场总线章节查找。下面以使用 IO 作为控制方式为例，介绍其中几个主要的信号。详细信息参考《埃夫特 ER 系列机器人操作手册》的功能 IO 章节。

在示教器的 IO 设置中，进入**功能 IO 配置的程序预约**配置中，根据需要配置相应 IO 口。

表 24-2 程序预约输入信号

序号	描述	说明	检测信号	操作模式
1-4	预约程序位 1/2/3/4	当程序预约的模式为单独，四个信号分别对应程序 1/2/3/4。此时当前程序为非预约且不在运行中，接收到此脉冲信号，程序会预约上，无需确认。 当程序预约的模式为二进制，程序 4 到 1 信号的状态按顺序组成四位二进制数，程序 4 在最高位，程序 1 在最低位。例如程序 4 到 1 的状态为 0、1、0、1，则组成的二进制数为 0101，对应十进制数为 5，则预约 5 号程序；	单独时候脉冲信号；二进制时候高低电平	自动有效
5	确定预约程序	当程序预约的模式为二进制时有效，先用程序号确定预约程序号，然后输入此信号，用以确定预约。	脉冲信号	自动有效
6	取消预约程序	当程序状态为预约中，先选择程序号，单独时候，需要输入单独的 I0 并保持。然后输入此信号，用以取消当前已经预约的信号。	脉冲信号	自动有效

7	启动/停止程序预约	当程序预约处于停止状态，输入此信号，可以开始程序预约运行；当程序预约处于启动状态，输入此信号，可以停止程序预约运行。注意，停止程序预约，机器人依旧会将当前正在运行的程序运行完成。之后已经预约的不再执行。	脉冲信号	自动有效

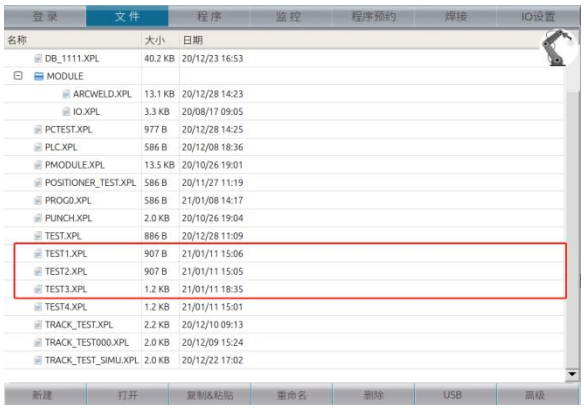
表 24-3 程序输出信号

序号	描述	说明	输出信号	操作模式
1	程序预约启动状态	启动程序预约，发出此信号；停止程序预约，信号复位。	高低电平	自动有效
2-16	程序 1-15 状态	反馈程序预约的状态，当程序未预约，则无信号；当程序预约中，则输出常量信号；当程序运行中，则输出方波信号。	/	自动有效

25.3.2 使用示例

25.3.2.1 单独示例

表 24-4 单独模式使用示例

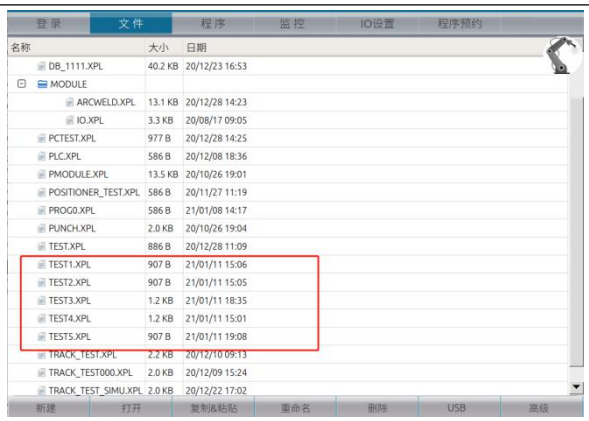
步骤	图片	描述
1. 新建程序并编程。		以 TEST1、TEST2、TEST3 为例。
2. 程序预约配置。		在程序预约中配置好相关参数并保存，不使用的则不需要选择。

3. 配置输入 IO 信号。		如图所示，配置三个程序对应的 IO 口，未使用的可以不配置。 配置启动/停止程序预约信号。 其他如远程伺服确认、急停、暂停等，根据实际需求配置。
4. 配置输出 IO 信号。		如图所示，配置程序预约启动状态信号。
5. 将机器人打到自动状态并上伺服。		
6. 开始程序预约		可以通过示教器的程序预约界面“开始”按钮来开始程序预约。 也可以通过 12 号 IO 口输入信号，开始程序预约功能。 开始之后，输出 9 会输出信号。

7. 通过 IO 预约程序运行。		注意单独模式无需程序预约确认。

25.3.2.2 二进制示例

表 24-5 二进制模式使用示例

步骤	图片	描述
1. 新建程序并编程。		以 TEST1、TEST2、TEST3、TEST4、TEST5 为例。
2. 程序预约配置。		在程序预约中配置好相关参数并保存。

3. 配置输入 IO 信号。		如图所示，配置四个程序对应的 IO 口。 配置确认程序预约信号。 配置启动/停止程序预约信号。 其他如远程伺服确认、急停、暂停等，根据实际需求配置。
4. 配置输出 IO 信号。		如图所示，配置程序预约启动状态信号。
5. 将机器人打到自动状态并上伺服。		
6. 开始程序预约		可以通过示教器的程序预约界面“开始”按钮来开始程序预约。 也可以通过 14 号 IO 口输入信号，开始程序预约功能。 开始之后，输出 9 会输出信号。

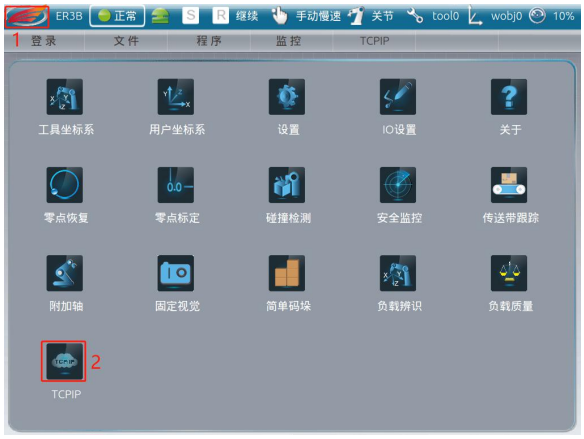
7. 通过 IO 预约程序运行。		二进制预约需要通过四个程序位的 IO 口确定程序，然后确认预约。 本例中比如预约 3 号程序，则需要在 IO 口的 9、10 号 IO 口输入信号并保持，然后在 13 口输入个脉冲信号，3 号程序才成功预约。

第 26 章　TCP/IP 通讯

机器人可通过 TCP/IP 和第三方设备进行字符串数据交换和传递，其中机器人即可作为服务器也可作为客户端，可通过示教器界面配置对应的通讯参数以及使用对应的通讯指令进行通讯。控制器里的 TCPIP 通讯功能可以同时开通 4 个 Socket，即客户端可以设置 4 路 Socket，服务器也可以设置 4 路 Socket。网线一端接入第三方设备通讯网口，一端接入机器人控制器的通讯网口，该网口默认地址:192.168.1.12，同时第三方设备通讯网口 IP 地址需与机器人控制器网段保持一致。

26.1 TCP/IP 客户端通讯设置

表 25-1 设置客户端通讯参数以及通讯操作步骤

步骤	图示	说明
1．登录管理员权限后，按图示标记编号，依次打开示教器桌面，点击**“TCPIP”**图标进入主界面。	 	默认情况下，不能对通讯参数进行保存，获得管理员权限后才可进行保存参数。

2．获得管理员权限后，在通讯参数设置栏菜单下可以设置协议类型、套接字号、服务器 IP、端口号、超时时间、首字符、结束符设置参数。填写完后，点击**“保存”**按钮，完成参数保存。

编号 1：协议类型，分为客户端和服务器，选择客户端表示机器人做客户端。

编号 2：套接字号，即表示当前设置的协议类型使用对应套接字号。修改套接字号有两种方式，一种是通过编号 2 中的下拉框进行选择对应的套接字号；另一种是通过点击编号 10 和编号 11 来分别递减和递增对应的套接字号，编号 12 对应的数字就是对应的套接字号。

编号 3：服务器 IP 地址，填写第三方设备的 IP 地址，要和机器人控制器网口在同一网段。

编号 4：填写服务器端口号。

编号 5：设置超时时间，接收信息时如果时间超出这个值就会认为接收失败。

编号 6：首字符，默认为空，用户可在接受和发送的有效数据前自定义首字符和结束符，若首字符或结束符不为空，则在

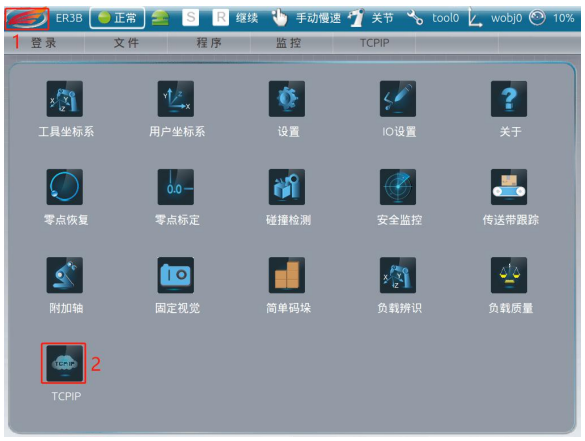
		接收数据时，控制器会自动去除首尾字符，只存储有效数据；在发送数据时，则会自动加上首尾字符。 编号 7：结束符参数设置，同编号 6，同时可通过勾选编号 8 和编号 9 在结束符后面添加回车换行符。 编号 8：回车符，勾选即可在结束符后面添加回车符。 编号 9：换行符，勾选即可在结束符后面添加换行符。 编号 10：套接字号递减按钮； 编号 11：套接字号递增按钮。 编号 12：套接字号显示。 编号 13：连接状态显示，  表示未连接状态  表示已连接上。 编号 14：每次修改或设置新的通路参数，必须点击**“保存”**按钮，才会刷新设置参数。 编号 15：**“退出”**按钮，点击可退出 app 界面。

3.设置完参数，点击**“保存”**按钮后，会弹出提示窗口。点击**“是”**进行参数保存，点击**“否”**取消参数保存。点击**“是”**进行保存参数后，会弹出对应的提示窗口。如果保存成功，会提示“保存成功”点击“是”，即完成了通讯参数的设置。		

26.2 TCP/IP 服务器通讯设置

表 25-2 设置服务器通讯参数以及通讯操作步骤

步骤	图示	说明
1．登录管理员权限后，按图示标记编号，依次打开示教器桌面，点击**“TCPIP”**图标进入主界面。		默认情况下，不能对通讯参数进行保存，获得管理员权限后才可进行保存参数。

<table>
<tr>
<td></td>
<td> 


</td>
<td></td>
</tr>
<tr>
<td>2．获得管理员权限后，在通讯参数设置栏菜单下可以设置协议类型、套接字号、端口号、超时时间、首字符、结束符、监听设置参数。填写完后，点击“保存”按钮，完成参数保存。</td>
<td> 
</td>
<td>编号 1：协议类型，分为客户端和服务器，选择服务器表示机器人做服务器。

编号 2：套接字号，即表示当前设置的协议类型使用对应套接字号。修改套接字号有两种方式，一种是通过编号 2 中的下拉框进行选择对应的套接字号；另一种是通过点击编号 10 和编号 11 来分别递减和递增对应的套接字号，编号 12 对应的数字就是对应的套接字号。

编号 3：服务器 IP 地址，控制器作为服务器时，则可以不填写该参数。

编号 4：填写服务器端口号，控制器做服务器</td>
</tr>
</table>

		时，表示本地端口号。 编号 5：设置超时时间，接收信息时如果时间超出这个值就会认为接收失败。 编号 6：首字符，默认为空，用户可在接受和发送的有效数据前自定义首字符和结束符，若首字符或结束符不为空，则在接收数据时，控制器会自动去除首尾字符，只存储有效数据；在发送数据时，则会自动加上首尾字符。 编号 7：结束符参数设置，同编号 6，同时可通过勾选编号 8 和编号 9 在结束符后面添加回车换行符。 编号 8：回车符，勾选即可在结束符后面添加回车符。 编号 9：换行符，勾选即可在结束符后面添加换行符。 编号 10：套接字号递减按钮； 编号 11：套接字号递增按钮。 编号 12：套接字号显示。 编号 13：连接状态显示，  表示未连接状态  表示已连接上。 编号 14：每次修改或设置新的通路参数，必须点击**“保存”**按钮，才会

		刷新设置参数。 编号 15：**“退出”**按钮，点击可退出 app 界面。 编号 16：手动设置停止和开始监听按钮。 编号 17：开机自启动监听功能。当勾选后表示保存参数后下次开机默认自动打开监听功能。
3.设置完参数，点击**“保存”**按钮后，会弹出提示窗口。点击**“是”**进行参数保存，点击**“否”**取消参数保存。点击**“是”**进行保存参数后，会弹出对应的提示窗口。如果保存成功，会提示“保存成功”点击“是”，即完成了通讯参数的设置。		

26.3 通讯指令说明

表 25-3 TCP/IP 指令说明

指令	名称	功能
tcpip.sockopen(id)	客户端通道开启函数	打开 tcp/ip 客户端通道： 输入：id 需要开启的套接字号。 输出：返回值=1 开启成功；

		返回值=-1　开启失败； 返回值=-2　重复开启。
tcpip.sockclose(id)	客户端通道关闭函数	关闭 tcp/ip 客户端通道： 输入：id 需要关闭的套接字号。 输出：返回值=1 关闭成功。
tcpip.socksend(id，str，type)	发送信息函数	向外发送字符串信息： 输入：id 需要发送信息的套接字号； str 需要发送的字符串信息； type =true 客户端；=false 服务器。 输出：返回值=1 发送成功； 返回值!=1 发送失败。
tcpip.sockrecv(id，type)	接收信息函数	接收外界字符串信息： 输入：id 需要接收信息的套接字号； type =true 客户端；=false 服务器。 输出：str　接收的字符串； 返回值=1 接收成功； 返回值=-1 接收失败； 返回值=-2 接收时间超时； 返回值=-3 信息格式错误。
tcpip.getrobotdata()	获取机器人状态	获取机器人当前状态，并体现为字符串的形式： 输入：无。 输出：str1 机器人状态； str2 报警信息； str3 笛卡尔坐标； str4 关节坐标。
tcpip.client_con[i]	客户端连接状态	得到对应套接字号的客户端连接状态，i 表示客户端的第几个套接字。
tcpip.server_con[i]	服务器连接状态	得到对应套接字号的服务器连接状态，i 表示服务器的第几个套接字。

机器人状态的格式：

首字符	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	尾字符
@	伺服状态	急停状态	报警状态	RPL 程序状态	手自动模式	速度百分比	&

机器人报警信息的格式：

首字符	Byte1	Byte2	尾字符
@	报警号	报警信息	&

机器人笛卡尔坐标的格式：

首字符	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	尾字符
@	X	Y	Z	A	B	C	&

机器人关节坐标的格式：

首字符	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7-10	尾字符
@	J1	J2	J3	J4	J5	J6	Aux1-4	&

备注：以上字节之间是已头号“，”隔开。

26.4 字符串指令说明

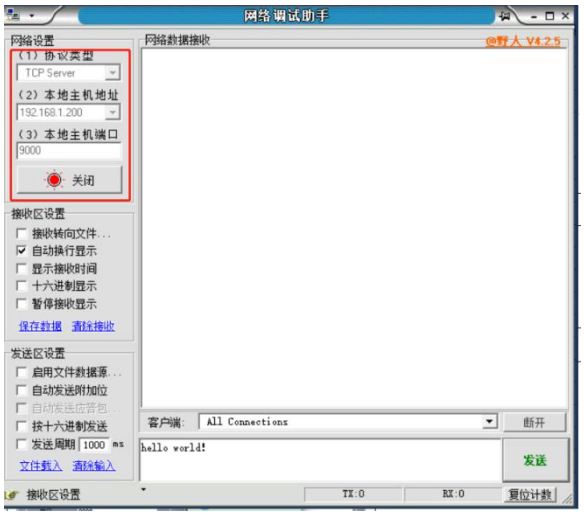
表 25-3 字符串指令说明

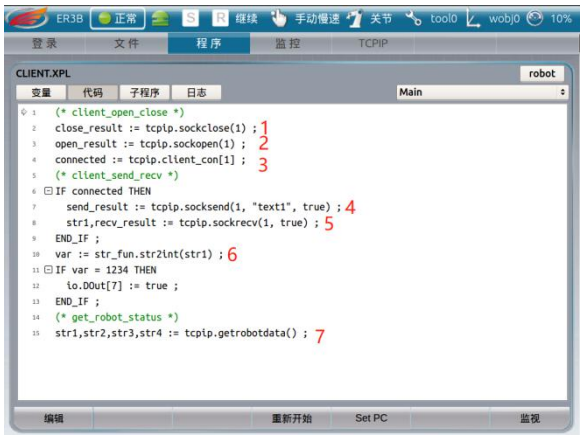
指令	名称	功能
str_fun.str2int()	字符串转整型	将字符串转换成整型。
str_fun.str2real()	字符串转实数	将字符串转换成实数。
str_fun.int2str()	整型转字符串	将整型数转换成字符串。
str_fun.real2str()	实数转字符串	将实数转换成字符串。
str_fun.strlen()	字符串长度	得到字符串长度。
str_fun.strcmp(str1，str2)	字符串对比	str1=str2 返回值=0，str1>str2 返回值>0，str1<str2 返回值<0。

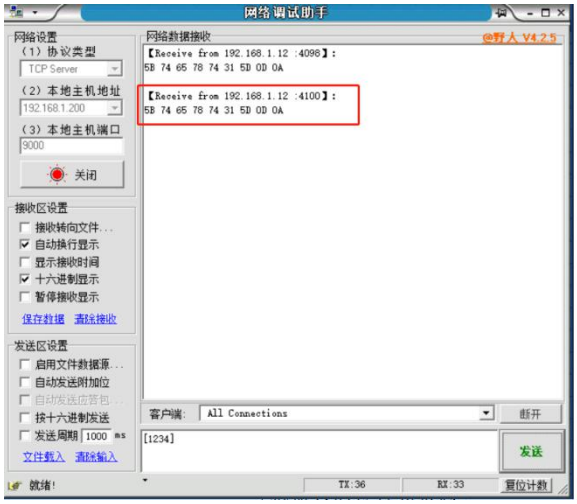
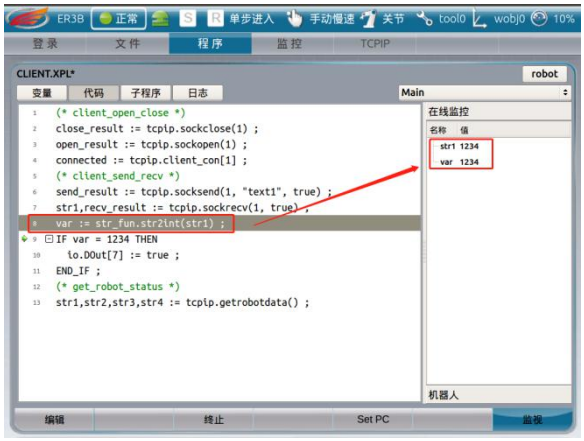
str_fun.strsplit(str1，delim，N)	字符串分割提取	把 str1 按照 delim 分隔符进行拆分，输出拆分开后的第 N 个字符串，N 从 1 开始。
str_fun.strleft(str1，N)	字符串取左	在字符串 str1 中从最左边开始数 N 个字符的字符串进行输出，N 从 1 开始。
str_fun.strright(str1，N)	字符串取右	在字符串 str1 中从最右边开始数 N 个字符的字符串进行输出，N 从 1 开始。
str_fun.strmid(str1，pos，N)	字符串取中	在字符串 str1 中取出从第 pos 个字符起始的长度为 N 的字符串。pos 从 1 开始。
str_fun.getbit(int，N)	整型取位	取整型 int 的第 N 位进行输出。
str_fun.strconcat(str1，str2，delim)	字符串拼接	输出拼接后的字符 str1+delim+str2。

26.5 TCP/IP 客户端程序

本例程使用网络调试助手做服务器进行模拟发送，用控制器作为客户端进行接收数据。

步骤	图示	说明
1.服务器应先开启，本例程用调试助手模拟，设置好端口号，先打开服务器监听。		备注： 本例程仅作为指令的使用简易说明，具体程序搭建需客户根据实际项目进行扩展或编写。

2.进入 TCPIP 设置界面设置参数。示例中对首字符和结束符都进行设置测试。		
3. 编辑示教程序。		编号 1：先关闭套接字 1 通讯，参数 1 表示套接字 1。 编号 2：打开套接字 1。 编号 3：获得客户端套接字 1 的连接状态。 编号 4：客户端套接字 1 给服务器发送字符，字符由首字符+发送字符+结束符组成。 编号 5:客户端套接字 1 接收服务器发送的字符，发送数据格式也要和配置的格式一致，需要添加对应的首字符和结束符。 编号 6：将接收到的字符 1234 转换成整

	接收到[text]以及回车换行符的十六进制数据  服务器发的 1234 字符串转为整型数值 1234  	型数 1234。 编号 7：获得当前机器人的位置和状态信息。

26.6 TCP/IP 服务器程序

本例程使用网络调试助手做客户端进行模拟发送，用控制器作为服务器进行接收数据。

步骤	图示	说明
1.进入 TCPIP 设置界面设置参数，并打开监听。本例中发送和接收数据中不设置首字符和结束符。同时打开下次系统启动后自动打开监听功能，因此服务器打开监听功能需要手动进行打开。		备注： 本例程仅作为指令的使用简易说明，具体程序搭建需客户根据实际项目进行扩展或编写。

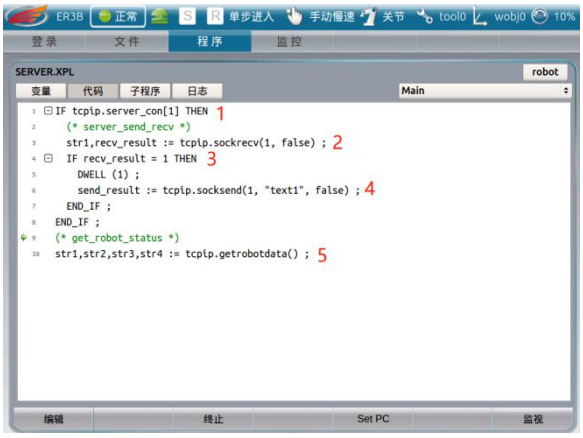
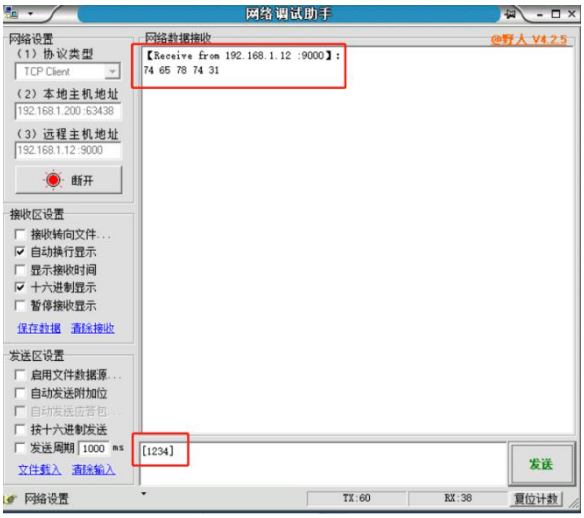
2.点击“开始监听”后，监听按钮显示为停止监听，服务器监听功能已打开，客户端可以进行连接。

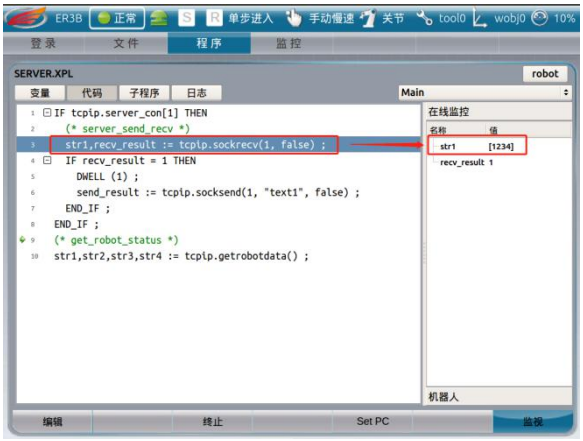
连接正常后的状态

备注：

1.服务器的监听开关需要在设置界面进行操作。

2.当客户端断开连接后，服务器的连接状态不会立刻显示为未连接状态，服务器会进行尝试重连，一旦尝试重连后仍然连接失败，状态才会变成未连接状态。

3.编辑示教程序。	 服务器发送 text1 给客户端的十六进制数据  客户端发送的[1234]字符串给服务器	编号 1：判断服务器套接字 1 是否连接成功。 编号 2：接收客户端发送的数据。 编号 3：判断是否成功接收数据。 编号 4：服务器发送字符给客户端，字符由首字符+发送字符+结束符组成。 编号 5：获得当前机器人的位置和状态信息。

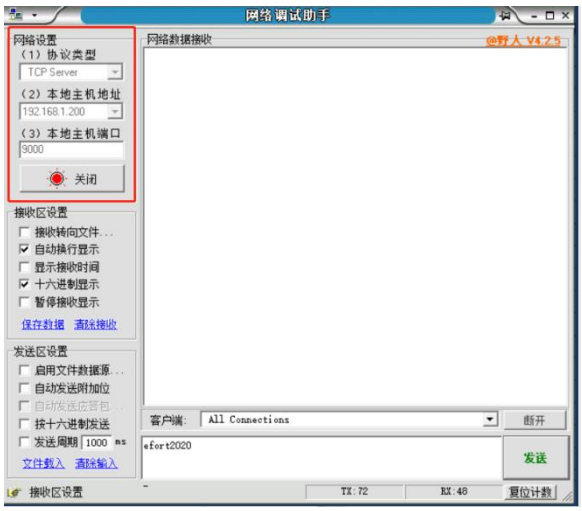

ER3B
正常
单步进入
手动慢速
关节
tool0
wobj0
10%
登录
文件
程序
监控
SERVER.XPL
robot
变量
代码
子程序
日志
Main
1 IF tcpip.server_con[1] THEN
2 (* server_send_recv *)
3 str1,recv_result := tcpip.sockrecv(1, false) ;
4 IF recv_result = 1 THEN
5 DWELL (1) ;
6 send_result := tcpip.socksend(1, "text1", false) ;
7 END_IF ;
8 END_IF ;
9 (* get_robot_status *)
10 str1,str2,str3,str4 := tcpip.getrobotdata() ;
在线监控
名称
值
str1
[1234]
recv_result 1
机器人
编辑
终止
Set PC
监视

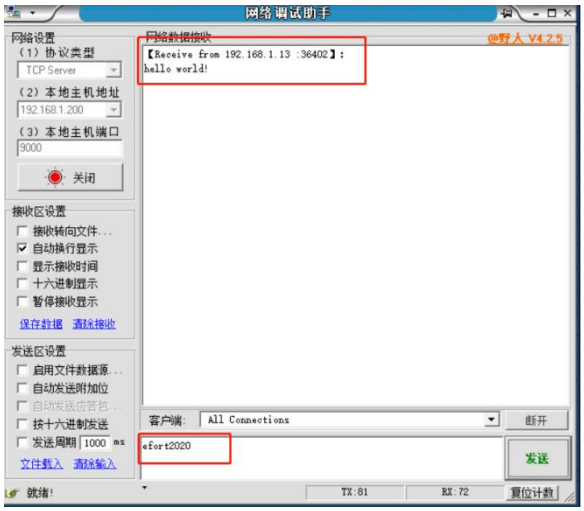
26.7 TPU 网络调试助手

TPU 网络调试助手主要用于第三方设备和示教器进行通讯测试网络是否通畅以及直观快捷的测试以及查看显示第三方设备发送的具体数据。

26.7.1 客户端模式

本例程使用网络调试助手做服务器进行模拟接收，用示教器作为客户端进行发送数据。

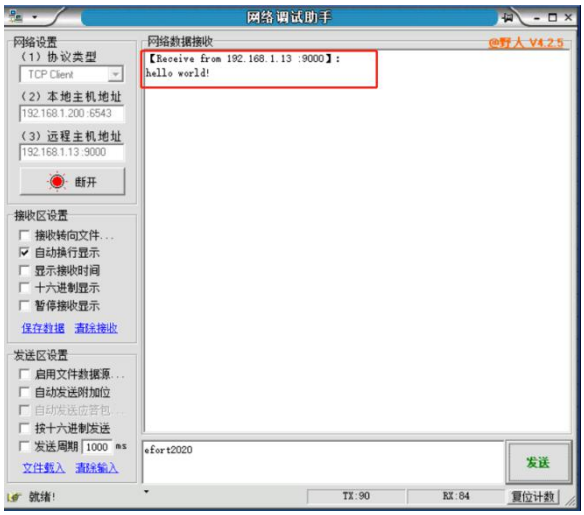
步骤	图示	说明
1.服务器应先开启，本例程用调试助手模拟，设置好端口号，先打开服务器监听。		
2.进入 TCPIP 设置界面，设置协议类型、服务器 IP 以及端口号，点击**“连接”**按钮，连接成功后显示在线。然后，点击**“发送”**按钮，发送数据给服务器。		编号 1：客户端打开和关闭按钮。 编号 2：清空接收区按钮。 编号 3：发送按钮。 编号 4：清空发送区按钮。 编号 5：接收到数据显示为十六进制数据。 编号 6：发送的数据转换为十六进制进行发送。
3.调试助手中显示接收到的数据，点击调试助手中**“发送”**按钮，发送数据给客户端示教器。		

	显示调试助手接收到的数据  显示示教器上客户端接收到的数据  	
4.点击**“关闭”**按钮，即可关闭客户端。		备注： 若切换页面后再次进入 TCPIP 界面，默认会自动断开客户端连接。

26.7.2 服务器模式

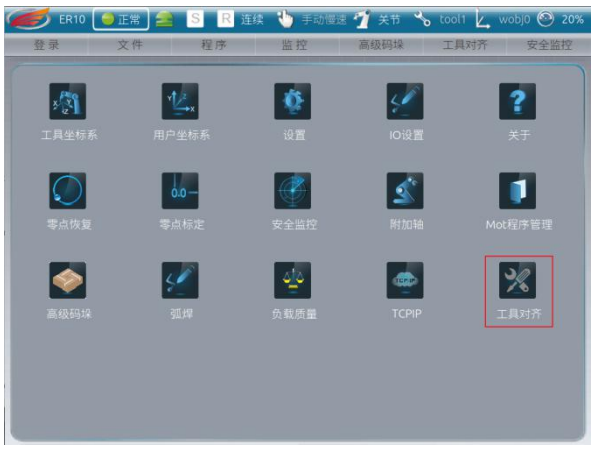
本例程使用网络调试助手做客户端进行模拟发送，用示教器作为服务器进行接收数据。

步骤	图示	说明
1.进入 TCPIP 设置界面，设置协议类型为服务器、以及端口号，服务器 IP 可不填写，点击“打开”按钮打开监听功能，连接成功后会在接收区显示连接成功。		
3.点击调试助手中**“发送”**按钮，发送数据给服务器示教器。	 	

4.点击示教器上的“**发送**”按钮发送数据给调试助手。	 	
5.点击示教器上的“**关闭**”按钮关闭服务器监听功能。		备注： 若切换页面后再次进入 TCPIP 界面，默认会自动关闭之前服务器监听。

第 27 章　工具手对齐

在所有测试之前需要连接控制器。

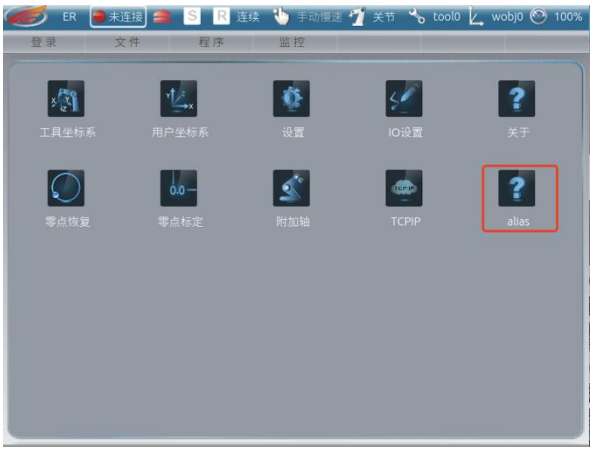
步骤	图片	描述
1. 连接好控制后，打开示教器进入桌面，选择“**工具对齐**”APP。		
2. 进入 APP 后，设置“**对齐信息**”，设置参考坐标和工具坐标，选择“**对齐方式**”仅垂直方向对齐或者所有方向对齐；点击“**启动**”按钮进入对齐页面。		① 选择对齐信息，设置对齐坐标。 ② 选择对齐方式，选择垂直对齐或者所有方向对齐。
3. 点击“Go”按钮，进行初始点和目标点计算。		① 计算当前设置的初始位置和对齐位置。

		
4. 按手压，并长安“+”按钮，直到，机器人远动到平齐位置。		① 按住手压，并按住“+”按钮，直到反馈目标位置到达。
5. 按手压，并长安“-”按钮，直到，机器人远动到初始位置。		② 按住手压，并按住“-”按钮，直到反馈到达初始位置。

6.按**“退出”**按钮，退出APP；按**“返回”**按钮返回首页。		

第 28 章　ALias

在所有测试之前需要连接控制器。

步骤	图片	描述
2. 连接好控制后，打开示教器进入桌面，选择“ALias”APP。		
2.进入APP后，设置，显示ALias变量信息		③ 点击 IO 进入 IO 变量主界面。 ④ 点击 Fidbus 进入 Fidbus 变量主界面。 ⑤ 点击退出当前 app
3.点击IO进入IO变量主界面。		① 计算当前设置的初点击Din进入Din变量主界面。 ② 点击 Dout 进入 Dout 变量主界面。 ③ 点击“添加”按钮，在当前页面添加一条 ALias 指令。 ④ 点击“删除”按钮，在当前页面选中的ALias指令

4. 点击 Fidbus 进入 Fidbus 变量主界面。	  	① 计算当前设置的初点击 Mtcp 进入 Mtcp 变量主界面。 ② 点击 Ethcat 进入 Ethcat 变量主界面。 ③ 点击“添加”按钮，在当前页面添加一条 ALias 指令。 ④ 点击“删除”按钮，在当前页面选中的 ALias 指令。 ⑤ 点击“写入”按钮，写入当前页面所有配置的指令信息。

第 29 章　动力学碰撞检测操作手册

29.1　步骤一：确认本体的安装方式

进入设置 APP 的“安装方式”界面进行确认，如果没有该界面，默认都是正装模式，如更改本体安装方式且需用动力学碰撞检测功能时，请第一时间联系技术服务人员进行指导操作。

29.2 步骤二：设置负载信息

确认主界面菜单有负载辨识 APP。

如界面上没有显示，需进入设置 APP 中的“应用选择”，输入密码 1975 后进行系统，并勾选负载辨识应用。

进入负载辨识 APP,手动设置负载信息。

1）选择想要设置的负载号（支持负载 0-9，共十组，可自行选择，本例选择负载 1；

2）点击“修改”按钮；

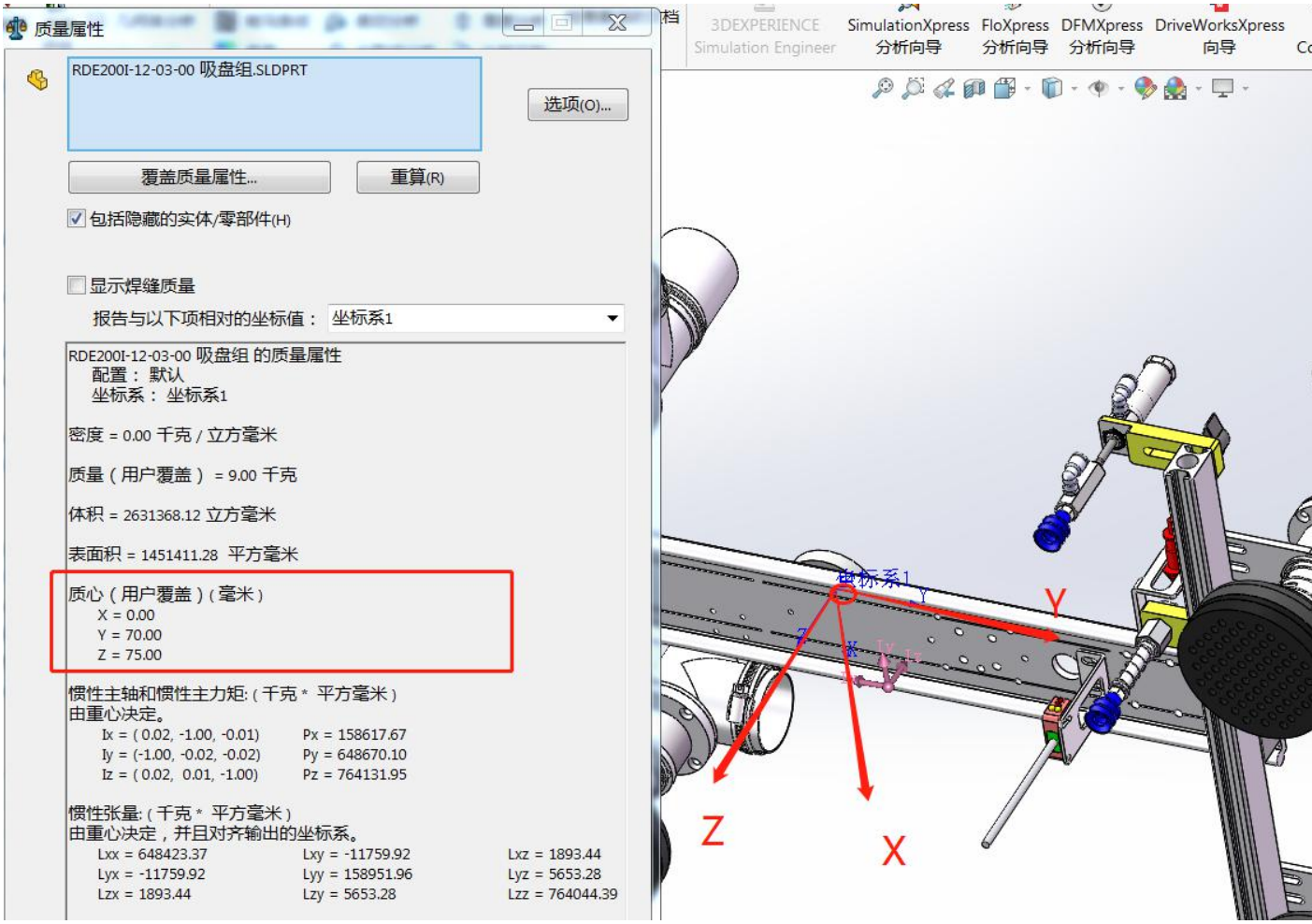
3）将 3D 数模计算结果写入到负载信息中，注意单位。

4）点击“保存”按钮，即可将修改的信息保存。

5）若要将修改的负载激活，则点击“激活”按钮，即可置为当前，并激活。


登录
文件
程序
监控
负载辨识
负载辨识功能
激活当前负载
当前负载：
负载0
负载名：
负载：
负载1
激活
修改
辨识
(Lx,Ly,Lz)
注意：坐标系的原点在末端法兰盘的中心
负载信息：
负载名：
质量：
0.0000
Kg
质心（单位：mm）
Lx:
0.0000
Ly:
0.0000
Lz:
0.0000
惯性张量（单位：Kg*mm^2）
Ixx:
0.0000
Ixy:
0.0000
Ixz:
0.0000
Iyy:
0.0000
Iyz:
0.0000
Izz:
0.0000
退出

模型数据：



夹具空载模型质量 9kg，另外根据客户提供的工件 PCB 板质量 3kg，因此总质量填 12kg 即可。（正常情况下，为了得到更加准确的动力学碰撞检测效果，应该设置两组不同负载参数，空载和带载，并根据实际模型质量和重心位置填写，并在程序中根据实际情况通过 RPL 指令激活不同的负载参数，但是一般情况，为了简化操作，可以设置一组带载的负载参数）。

激活设置的负载参数

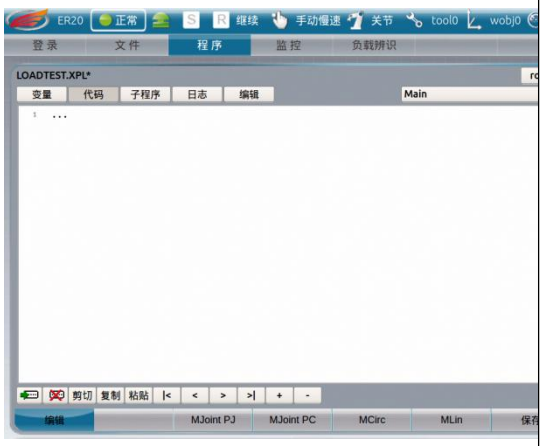
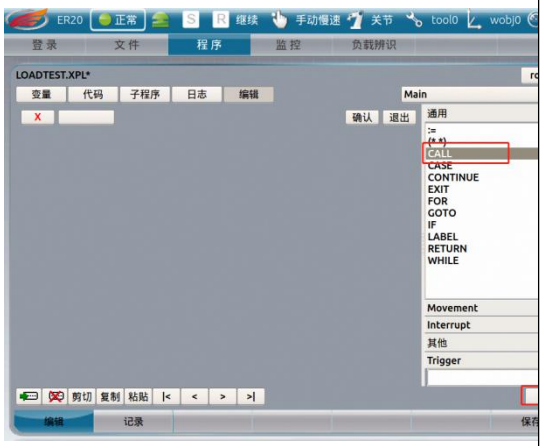
两种方式可以激活：1：界面直接激活法；2、RPL 指令激活法

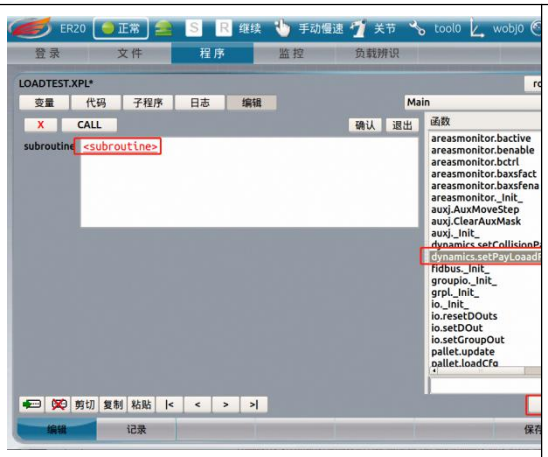
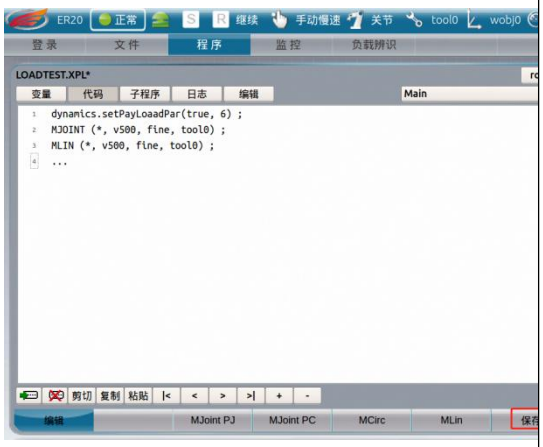
1：界面直接激活法

2、RPL 指令激活法

指令	名称	功能
dynamics.setPayLoadPar(Bool isEnablePayLoad， DINT payLoadIndex)	设置负载参数指令	实现负载的激活关闭，以及当前负载号的切换设置

负载程序用例

步骤	图片	描述
1.新建或打开 RPL 程序		
2.插入 call 指令		

3.找到碰撞检测指令，并设置参数如： Dynamics.setPayLoadPar(true,6)	 	RPL 程序中只有加入了此条指令并运行过后所切换的负载参数才会被加载，以及负载的激活状态才能被更新 注意：该指令必须用在碰撞检测指令前。
4.保存程序测试运行即可	 	

注意事项

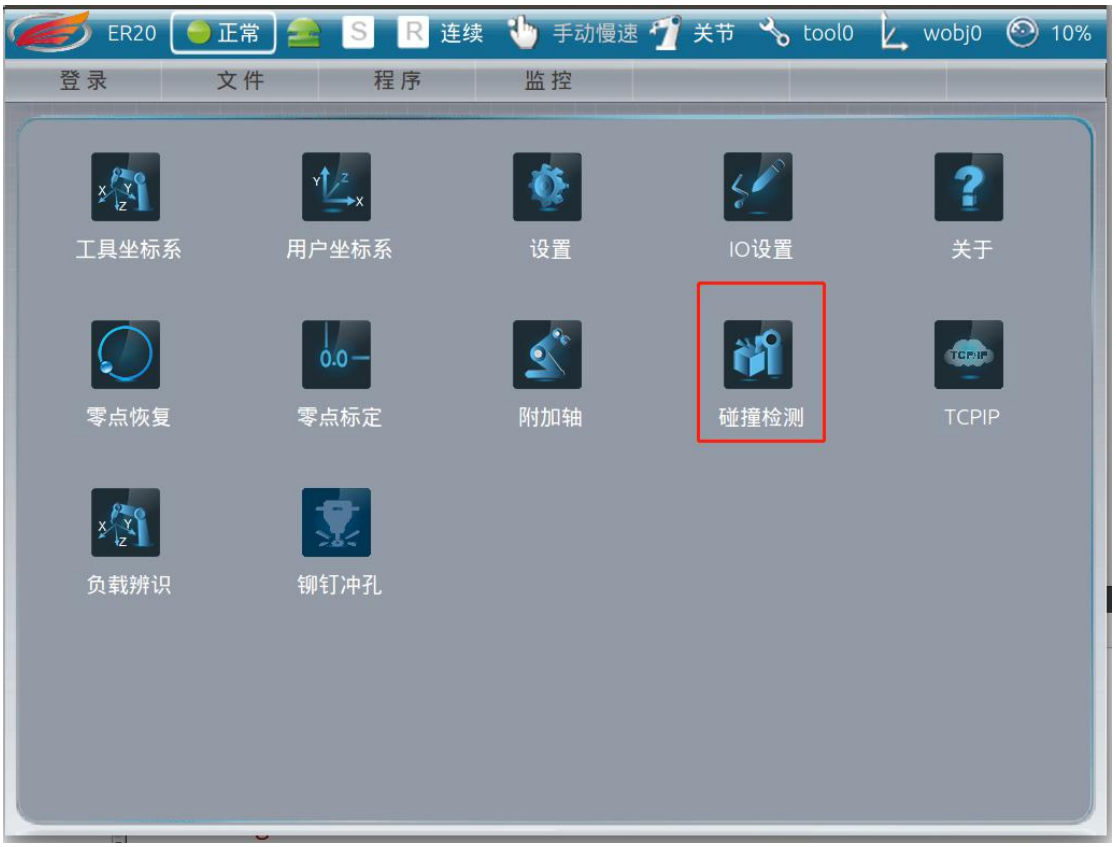
1.所设定的负载参数均在机器人法兰坐标下进行描述的。

2.在通过负载指令切换或激活关闭负载时需提前通过指令关闭程序碰撞检测或者调高碰撞检测的全局阈值。

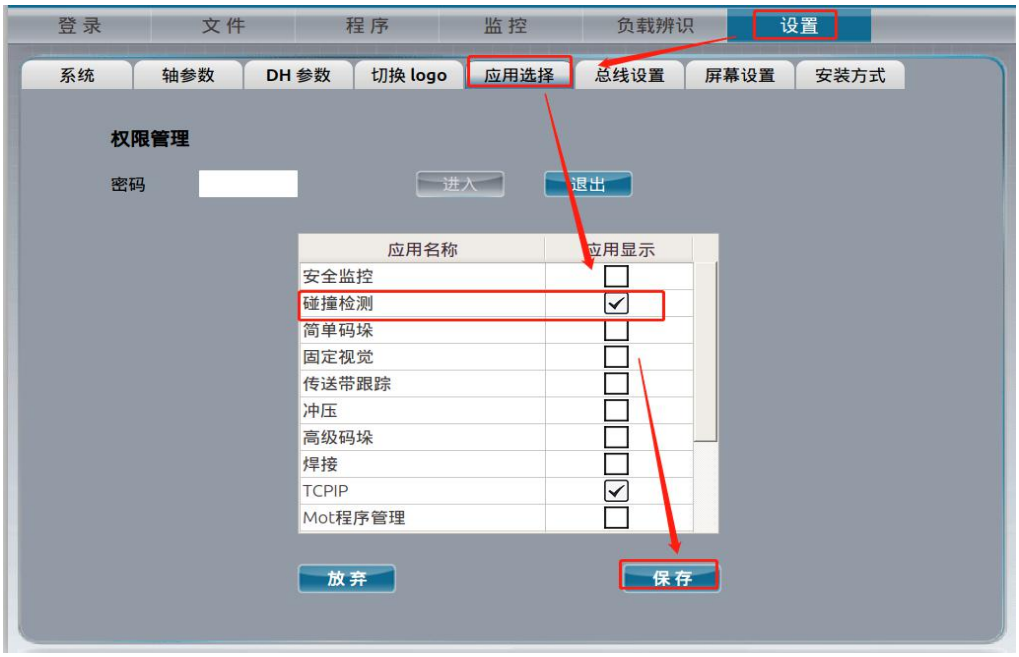
3.无论是工具还是夹持着工件的工具均可被定义为一个负载，需要设置不同的负载参数

29.3 步骤三：打开碰撞检测功能并设置碰撞检测的灵敏度

1、确认主界面菜单有碰撞检测 APP。



如界面上没有显示，需进入设置 APP 中的“应用选择”，输入密码 1975 后进行系统，并勾选碰撞检测应用。



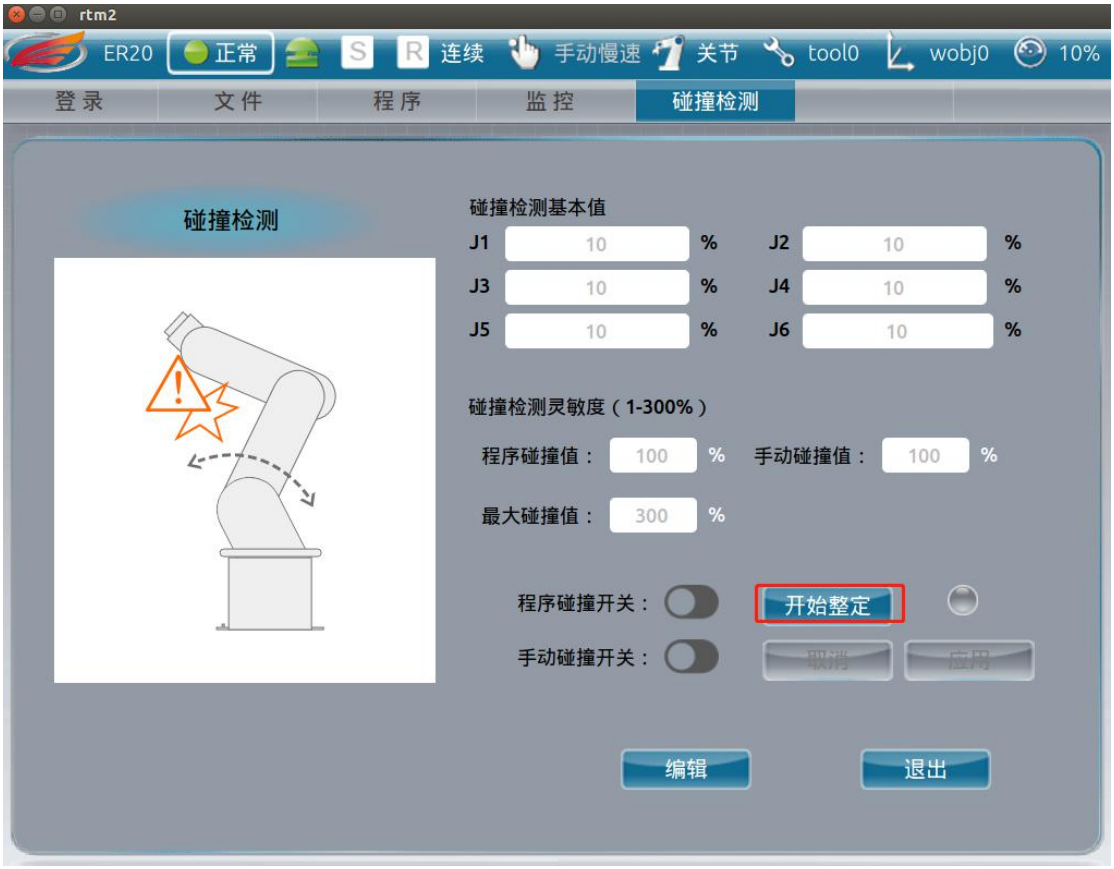
1、进入 app 后先进行整定参数。

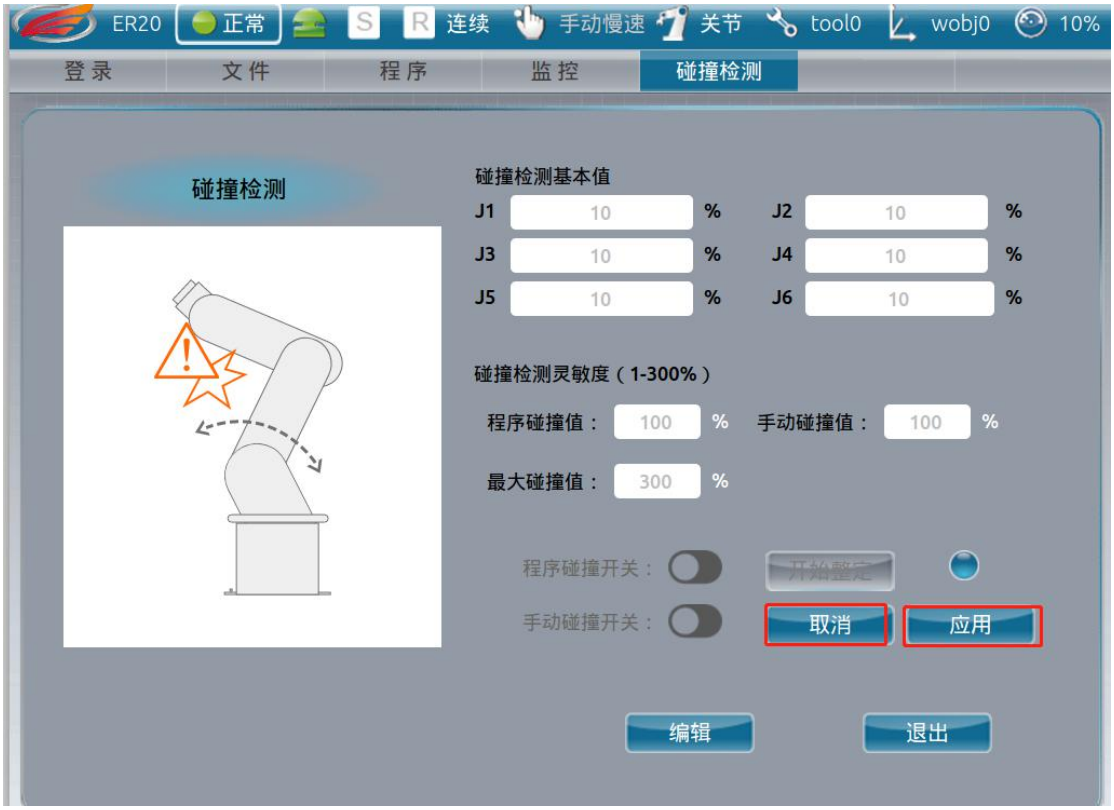
编写要生产的 RPL 程序。

在自动模式下运行 RPL 程序。

点击“开始整定”按钮，在运行程序过程中建议可模拟实际生成，运行一段时间后，可按下暂定按钮，然后再按启动按钮恢复机器人的运行。当运行生产程序 2 至 3 遍后，暂停机器人。

点击“应用”按钮，会将采集到的数据保存到控制器中；若不想使用整定结果，则点击“取消”按钮。



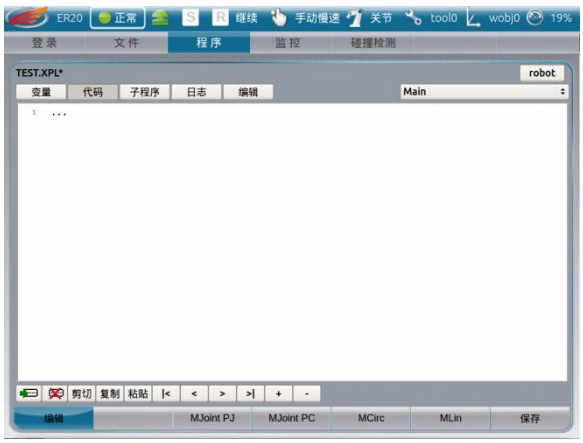


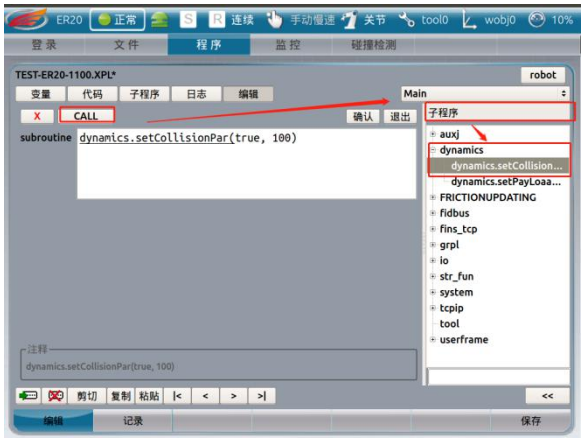
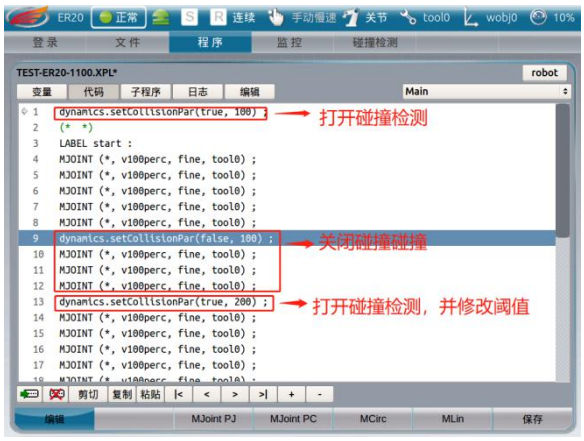
2、整定完成后，具有两种方式打开碰撞检测功能：1）、界面打开程序碰撞检测开关；2)、RPL 指令打开碰撞检测开关。

1）界面打开程序碰撞检测法。当打开碰撞开关并正常运行程序时，出现碰撞报警，需要手动调大对应轴的碰撞检测基本值。每次按 5%的递增值进行调整。直到不报警。

2）RPL 指令打开碰撞检测开关。

指令	名称	功能
dynamics.SetCollisionPar(Bool enableCollision， DINT collisionSensitivity)	碰撞检测设置参数指令	实现碰撞检测功能的开启和碰撞检测的灵敏度设置

步骤	图示	说明
1.新建或打开 RPL 程序		

2．插入 call 指令，找到碰撞检测参数设置指令：dynamics.SetCollisionPar(__,__)		其中第 1 个参数为 true 时，启动碰撞检测，为 false 时，不启用；第 2 个参数值设置范围为 1~200，默认设置为 100。
3．根据实际情况可灵活的在特定行打开和关闭碰撞检测开关。特别是当程序中存在某一段不是关键点位的的程序行经常碰撞检测误报警，可以先关闭碰撞检测功能，待运行完这一段代码后，可通过指令再次打开碰撞检测功能。		1）在运行 RPL 程序时，最终各关节碰撞检测灵敏度=界面设置的程序碰撞灵敏度 • 程序指令设定参数 • 各运动轴设定独立碰撞检测基本值； 2）当最终各关节的碰撞检测灵敏度值大于界面设置的最大碰撞值时，以界面设置的最大碰撞值为准。

29.4 注意事项

1.确认动力学模型是否与机器人安装方式（正装、倒装、侧装）匹配，出厂默认为正装的动力学模型。若机器人不是正装方式，请联系埃夫特技术支持人员获取非正装的动力学模型参数。

2.使用碰撞检测功能应该检查机器人末端负载参数是否配置正确，以及是否被正确激活。

3.阈值更新是需保证机器人所运动程序过程中无碰撞及与外界力交互操作产生，否则将导致更新出的阈值上限偏大，检测灵敏程度降低。

4.阈值更新过程将自动关闭所有模式下的碰撞检测功能，更新后需要手动恢复更新前的开启状态。

5.阈值更新过程的机器人运动程序轨迹及速度加速度负载等信息，需要尽可能接近实际机器人应用时的状态。

6.通过界面修改了各关节的运动速度最大上限并重启后需要重新更新阈值。

7.在应用阈值更新结果前，建议停止当前所运行的程序。

第 30 章　总线设置

30.1 本章简介

本章主要是对机器人支持的一些总线协议进行相关设置。支持的协议根据机器人的不同可能存在差异，实际支持设置项目以机器人为准。

30.2 总线设置

总线设置主要设置机器人机器人开放的总线类型，根据机器人不同，可能部分协议不支持选择。其设置流程如下。

表 30-1 配置总线操作流程

步骤	图片	描述
1.打开“**总线设置**”功能。		打开示教器桌面，点击“**总线设置**”图标进入 APP 界面。
2.根据需要，选择开启的总线类型，不使用的建议关闭。		修改完成后，点击“**保存**”按钮，保存成功后，重启才能生效。

30.3 Mes 监控配置

表 30-2 MES 监控配置操作流程

步骤	图片	描述

1.打开“**总线设置**”功能。		进入“**总线设置**”页面，选择 Mes。
2.停止服务器监听。		服务器开机自动打开，若需要修改配置，须停止监听。
3.Mes 设置		设置监听端口、模式、协议以及广播时间间隔，点击“**保存**”按钮，使配置生效。 控制器 IP 地址修改须在设置 app 中修改。

监听模式：

广播：服务器主动发送数据，支持机器人和附加轴协议。
问询：客户端发送问询指令，服务器应答机器人状态，只支持机器人协议。

监听内容：

机器人：机器人状态信息。
附加轴：只有附加轴位置信息。

间隔时间：

只在广播模式下有效，发送数据的间隔时间。

MES 的协议数据以内存的方式进行存储，其协议分为两部分：机器人和附加轴部分。

30.3.1 机器人协议内容

问询模式，Mes 系统主发机器人协议。具体协议内容如下：

Mes 系统主发机器人					
序号	意义	变量	类型	字节	备注
1	报文开始标记	Packet_Start	string	16	MessageHead
2	数据长度	Packet_Length	short	2	38
3	数据命令	Packet_Orders	short	2	1001
4	数据心跳	Packet_Heartbeat	short	2	
5	报文结束标记	Packet_End	string	16	MessageTail

其中：数据心跳需要本次和上次发送不一样的数值。

机器人回发 Mes 系统					
序号	意义	变量	类型	字节	备注
1	报文开始标记	Packet_Start	string	16	MessageHead
2	数据长度	Packet_Length	short	2	788
3	数据命令	Packet_Orders	short	2	1002
4	数据心跳	Packet_Heartbeat	short	2	
5	报警状态	bErrorStatus	bool	1	1：有报警；0：无报警
6	急停状态	bHstopStatus	bool	1	1：无急停；0：有急停
7	权限状态	bAuthorityStatus	bool	1	没有外部 PLC 模式
8	伺服状态	bServoStatus	bool	1	1：有使能；0：未使能
9	轴运动状态	bAxisMoveStatus	bool	1	1：有运动；0：未运动
10	程序运行状态	bProgMoveStatus	bool	1	1：有运行；0：未运行
11	程序加载状态	bProgLoadStatus	bool	1	1：有加载；0：无加载
12	程序暂停状态	bProgHoldStatus	bool	1	1：有暂停；0：无暂停
13	模式状态	nModeStatus	short	2	0:手动慢速；1:手动全速；2:自动
14	速度状态	nSpeedStatus	short	2	百分比（单位）
15	IoDOut 状态	bIoDOut[0..31]	bool	1*32	数字量输出： 0-7 本地数字量输出 1-8； 8-31 远程数字量输 0-23；
16	IoDIn 状态	bIoDIn[0..31]	bool	1*32	数字量输入： 0-7 本地数字量输入 1-8； 8-31 远程数字量输入 0-23
17	IoIOut 状态	nIoIOut[0..31]	int	4*32	预留
18	IoIIn 状态	nIoIIn[0..31]	int	4*32	预留
19	加载工程名	strProjectName	string	32	预留
20	加载程序名	strProgramName	string	32	
21	错误信息	strErrorText	string	128	
22	各轴角度	dbAxisPos[0..6]	float	4*7	度（单位）

23	姿态位姿	dbCartPos[0..5]	float	4*6	毫米|度（单位）
24	各轴速度	dbAxisVel[0..6]	float	4*7	度/秒（单位）
25	各轴加速度	dbAxisAcc[0..6]	float	4*7	度/秒^2（单位）
26	各轴加加速度	dbAxisJer[0..6]	float	4*7	度/秒^3（单位）
27	各轴力矩	dbAxisTor[0..6]	float	4*7	额定力矩百分比（单位）
28	各轴反向计数	nAxisDirCnt[0..6]	unsigned int	4*7	次（单位）统计
29	各轴工作总时长	nAxisTime[0..6]	unsigned int	4*7	秒（单位）统计
30	设备开机总时长	nDeviceTime	unsigned int	4	秒（单位）统计
31	报文结束标记	Packet_End	string	16	MessageTail

其中：数据心跳需要发送接收到的数据心跳的数值。

30.3.2 附加轴协议内容

客户端连上服务器后，服务器主动以指定间隔时间发送数据。

Mes 系统主发机器人			
序号	**附加轴数量**	**数据类型**	**备注**
1	0	string	Current robot has no auxiliary axes\r\n
2	1	string	EA1:xx.xx;\r\n
3	2	string	EA1:xx.xx;EA2:xx.xx;\r\n

30.4 PFB/PFN 设置

PFB/PFN 即为 Profibus-DP 和 Profinet 的简写，这里可以设置 PFB/PFN 的从站地址，部分机器人可能不支持此功能，以实际使用机器人为准。

表 30-3 PFB/PFN 设置从站地址

步骤	图片	描述
1.打开“**总线设置**”功能。		进入“**总线设置**”页面，选择 PFB/PFN。

2.设置 PFB/PFN 的从站地址和报警开关，点击**“保存”**按钮保存设置。		服务器开机自动打开，若需要修改配置，须停止监听。 检测连接报警开，则检测 PFB/PFN 通讯协议的通断，如果通讯断开，示教器弹出报警。选项关，则不检测通讯的通断。

30.5 FinsTcp 设置

表 30-4 FinsTcp 设置

步骤	图片	描述
1.打开**“总线设置”**功能。		进入**“总线设置”**页面，选择 FinsTcp。
2.点击启用 FinsTcp 复选按钮，开启 FinsTcp 通讯功能。		按钮左侧连接状态灯亮则连接成功，反之未连接。 点击启用 FinsTcp 按钮后该按钮将禁用至 FinsTcp 通讯功能被完全开启/关闭才使能。

2.设置服务器 IP，服务器端口，D 区读写地址，点击**“保存”**按钮保存设置。		本地网口与客户端地址从配置文件中读取。

30.6 EtherCat 设置

表 30-5 EtherCat 设置

步骤	图片	描述
1.打开**“总线设置”**功能。		进入**“总线设置”**页面，选择 EtherCat。
2.点击使能 ECAT 从站按钮选中或取消选中以此开启或关闭 ECAT 通讯功能，点击**“保存”**按钮保存设置。		ECAT 从站地址为仅可读状态。

30.7 EtherNet 设置

表 30-5 EtherCat 设置

步骤	图片	描述

1.打开“**总线设置**”功能。		进入“**总线设置**”页面，选择 EtherNet。
2.选择对应本地网口，修改 EtherNet 的网口。		选择后自动读取对应网口设置的 IP 和子网掩码。 点击“保存”按钮保存参数后，重启即可。 注：当机器人开机时出现“9204：DNSLV(0)配置错误”报警的时候，则为系统 IP 和 EtherNet 的 ip 不对应，则需要修改 EtherNet 的 ip 即可。

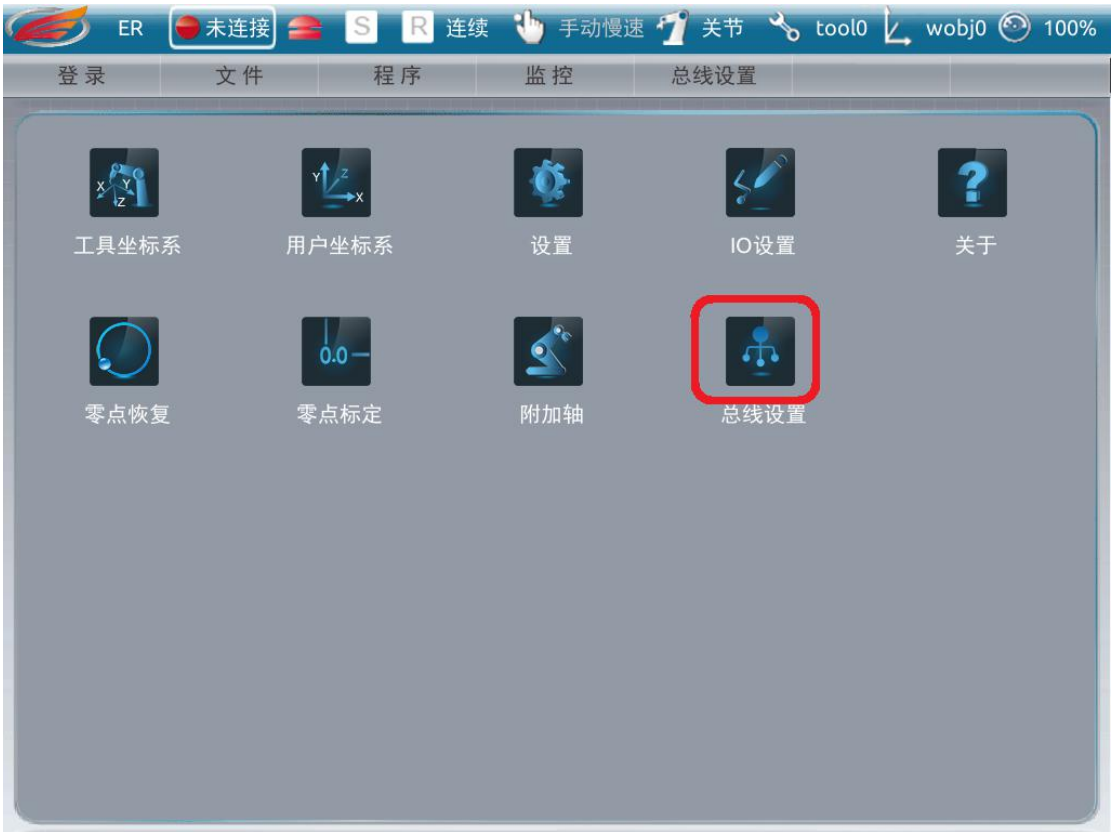
30.8 心跳检测设置

30.8.1 心跳检测功能介绍：

1. 点击设置--

2. 应用选择--勾选“总线设置”--保存

3. 点击“总线设置”--



4. 点击“心跳检测”--勾选相应的协议的开关来开启心跳检测--并设置最小检测时间（超过这个时间

没收到心跳会报警）--最后点击保存。

30.8.2 界面操作介绍

表 30-5 心跳检测设置

步骤	图片	描述
1.打开“**总线设置**”功能。		进入“**总线设置**”页面，选择心跳检测。 开关：负责心跳功能的打开和关闭。 时间：在您设置的时间内机器人保持心跳检测开启的时间。

2.打开对应四种类型协议的开关按钮和输入需要检测的时间，点击**“保存”**按钮保存设置。		通过检测心跳值有没有改变来判断通讯的连接状态是否断开，连接状态的显示呈灰色时表示未连接，红色表示连接失败，绿色表示连接成功。 通过机器人和 PLC 之间的相互发送信号，双向检测，保证故障的时候双方都可以停下来。

30.8.3 心跳检测协议介绍

①Modbus 协议

机器人会检测heartbeat RX 变量的数值变化，使用PLC或其他嵌入式设备对指定协议的heartbeat RX进行累加操作，机器人会根据规定时间内 heartbeat RX 有无变化来判断心跳是否丢失。

40105	R(3005)	R(3405)	I16	3	aux number	/
40106	R(3006)	R(3406)		4	set aux speed	
40107	R(3007)	R(3407)		5	reserved 5	
40108	R(3008)	R(3408)		6	reserved 6	
40109	R(3009)	R(3409)		7	heartbeat RX	
40110	R(3010)	R(3410)		8	reserved 8	
40111	RR(2001)	RR(2401)		1	reserved 1	
40113	RR(2002)	RR(2402)		2	reserved 2	
40115	RR(2003)	RR(2403)		3	reserved 3	
40117	RR(2004)	RR(2404)		4	reserved 4	

机器人的 heartbeat TX 会不停的累加，使用 PLC 或其他嵌入式设备检测相应协议的 heartbeat TX，可以判断机器人心跳信号是否丢失。

R(3455)		I16	3	alarm code 2	
R(3456)			4	program number	
R(3457)			5	appointment program reserved	
R(3458)			6	appointment program running	
R(3459)			7	heartbeat TX	
R(3460)			8	reserved 8	
RR(2451)			1	position J1/X	
RR(2452)			2	position J2/Y	

当开启了心跳检测后，如果 RX 心跳丢失，机器人会弹出报警，并关闭伺服电机。

值得注意的是，如果开启心跳检测功能的话，PLC 端也务必对机器人发送的心跳进行检测，保证链路收发都有检测机制，以此来保证通讯安全。

②Ethercat 协议

机器人会检测”心跳检测接收“变量的数值变化，使用 PLC 或其他嵌入式设备对指定协议的 heartbeat RX 进行累加操作，机器人会根据规定时间内 heartbeat RX 有无变化来判断心跳是否丢失。

R(3506)	Y1050	11-12	Bus_Get[10]-[11]	INT		INT	附加轴速度设定	OB INT
R(3507)	Y1060	13-14	Bus_Get[12]-[13]	INT		INT	系统预留 INT 5	
R(3508)	Y1070	15-16	Bus_Get[14]-[15]	INT		INT	系统预留 INT 6	
R(3509)	Y1080	17-18	Bus_Get[16]-[17]	INT		INT	心跳检测接收	
R(3510)	Y1090	19-20	Bus_Get[18]-[19]	INT		INT	系统预留 INT 8	
R(3511)	Y10A0	21-22	Bus_Get[20]-[21]	INT		INT	TPU预留 BOOL 0	

机器人的”心跳检测发送“会不停的累加，使用 PLC 或其他嵌入式设备检测相应协议的 heartbeat TX，可以判断机器人心跳信号是否丢失。

R(3604)	X1030	7-8	Bus_Set[6]-[7]	INT		INT	报警代码	OB INT
R(3605)	X1040	9-10	Bus_Set[8]-[9]	INT		INT	报警代码	
R(3606)	X1050	11-12	Bus_Set[10]-[11]	INT		INT	程序号	
R(3607)	X1060	13-14	Bus_Set[12]-[13]	INT		INT	预约程序预约状态	
R(3608)	X1070	15-16	Bus_Set[14]-[15]	INT		INT	预约程序运行状态	
R(3609)	X1080	17-18	Bus_Set[16]-[17]	INT		INT	心跳检测发送	
R(3610)	X1090	19-20	Bus_Set[18]-[19]	INT		INT	系统预留 INT 8	
R(3611)	X10A0	21-22	Bus_Set[20]-[21]	INT		INT	TPU预留 BOOL 0	TPU BOOL
R(3612)	X10B0	23-24	Bus_Set[22]-[23]	INT		INT	TPU预留 BOOL 1	

当开启了心跳检测后，如果 RX 心跳丢失，机器人会弹出报警，并关闭伺服电机。

值得注意的是，如果开启心跳检测功能的话，PLC 端也务必对机器人发送的心跳进行检测，保证链路收发都有检测机制，以此来保证通讯安全。

③profinet 协议

机器人会检测 heartbeat RX 变量的数值变化，使用 PLC 或其他嵌入式设备对指定协议的 heartbeat RX 进行累加操作，机器人会根据规定时间内 heartbeat RX 有无变化来判断心跳是否丢失。

40105	R(3005)	R(3405)	I16	3	aux number	/
40106	R(3006)	R(3406)		4	set aux speed	
40107	R(3007)	R(3407)		5	reserved 5	
40108	R(3008)	R(3408)		6	reserved 6	
40109	R(3009)	R(3409)		7	heartbeat RX	
40110	R(3010)	R(3410)		8	reserved 8	
40111	RR(2001)	RR(2401)		1	reserved 1	
40113	RR(2002)	RR(2402)		2	reserved 2	
40115	RR(2003)	RR(2403)		3	reserved 3	
40117	RR(2004)	RR(2404)		4	reserved 4	

机器人的 heartbeat TX 会不停的累加，使用 PLC 或其他嵌入式设备检测相应协议的 heartbeat TX，可以判断机器人心跳信号是否丢失。

R(3455)		I16	3	alarm code 2	
R(3456)			4	program number	
R(3457)			5	appointment program reserved	
R(3458)			6	appointment program running	
R(3459)			7	heartbeat TX	
R(3460)			8	reserved 8	
RR(2451)			1	position J1/X	
RR(2452)			2	position J2/Y	

当开启了心跳检测后，如果 RX 心跳丢失，机器人会弹出报警，并关闭伺服电机。

值得注意的是，如果开启心跳检测功能的话，PLC 端也务必对机器人发送的心跳进行检测，保证链路收发都有检测机制，以此来保证通讯安全。

④Ethernet/IP 协议

机器人会检测 heartbeat RX 变量的数值变化，使用 PLC 或其他嵌入式设备对指定协议的 heartbeat RX 进行累加操作，机器人会根据规定时间内 heartbeat RX 有无变化来判断心跳是否丢失。

R(3706)	1	/	aux axis async move speed
R(3707)	1	/	reserved
R(3708)	1	/	reserved
R(3709)	1	/	heartbeat RX
R(3710)	1	/	reserved
RR(2501)	2	/	reserved
RR(2502)	2	/	reserved

机器人的 heartbeat TX 会不停的累加，使用 PLC 或其他嵌入式设备检测相应协议的 heartbeat TX，可以判断机器人心跳信号是否丢失。

R(3755)	1	/	alarm code 2
R(3756)	1	/	program number
R(3757)	1	/	appointment program reserved
R(3758)	1	/	appointment program running
R(3759)	1	/	heartbeat TX
R(3760)	1	/	reserved 8
RR(2551)	2	/	position J1/X
RR(2552)	2	/	position J2/Y
RR(2553)	2	/	position J3/Z

当开启了心跳检测后，如果 RX 心跳丢失，机器人会弹出报警，并关闭伺服电机。

值得注意的是，如果开启心跳检测功能的话，PLC 端也务必对机器人发送的心跳进行检测，保证链路收发都有检测机制，以此来保证通讯安全。

附录 1　控制器报警及警告

1.1 控制器报警

1.1.1 系统报警(1-999)

1.1.1.1 Alarm 1

故障描述：可保持用户寄存器值异常

故障原因：可保持用户寄存器读取值与上次关机前不一致

处理建议：如果在控制器通电时发生警报，请持续按压控制器上的 ADV 红色按钮，并重启。

1.1.1.2 Alarm 2

故障描述：可保持参数寄存器值异常

故障原因：可保持参数寄存器读取值与上次关机前不一致

处理建议：如果在控制器通电时发生警报，请持续按压控制器上的 ADV 红色按钮，并重启。

1.1.1.3 Alarm 3

故障描述：系统报警信息异常

故障原因：系统报警信息与上次关机前不一致

处理建议：如果在控制器通电时发生警报，请持续按压控制器上的 ADV 红色按钮，并重启。

1.1.1.4 Alarm 10

故障描述：编码器连接异常

故障原因：编码器故障

处理建议：检查编码器和 Robox 控制器之间的接线。

1.1.1.5 Alarm 11

故障描述：跟随误差异常

故障原因：目标位置和实际位置之间的差超过了最大允许值

处理建议：

1. 调整驱动器 PID 参数或者最大允许误差；
2. 检查机器人目标位置与当前位置是否合理(距离太远)；
3. 降低机器人运动参数(速度，加速度，加加速度)；
4. 检查控制器和驱动器的接线、驱动器和伺服电机的接线。

1.1.1.6 Alarm 14

故障描述：到达负限位

故障原因：目标位置超出设置的最小限位

处理建议：重新设定目标位置

1.1.1.7 Alarm 15

故障描述：到达正限位

故障原因：目标位置超出设置的最大限位

处理建议：重新设定目标位置

1.1.1.8 Alarm 21

故障描述：编码器连接异常

故障原因：可能是线路连接存在问题或总线通讯出错

处理建议：

1. 检查编码器和控制器之间的接线；

2. 检查总线通讯。

1.1.1.9 Alarm 82

故障描述：总线通讯中断

故障原因：EtherCAT 通信中断

处理建议：检查总线通讯

1.1.1.10 Alarm 83

故障描述：丢失节点，总线通讯中断

故障原因：EtherCAT 通信诊断

处理建议：检查总线通讯

1.1.1.11 Alarm 90

故障描述：系统同步周期异常

故障原因：可能是控制器故障或被干扰

处理建议：

1. 重启控制器；
2. 检查外部 PLC 等设备是否有干扰；
3. 联系厂家。

1.1.1.12 Alarm 804

故障描述：task 加载失败

故障原因：可能是控制器故障

处理建议：

1. 重启控制器；
2. 联系厂家。

1.1.1.13 Alarm 807

故障描述：OB 加载失败

故障原因：可能是控制器故障

处理建议：

1. 重启控制器；
2. 联系厂家。

1.1.1.14 Alarm 949

故障描述：轴仿真激活: 0x%08X

故障原因：进入了轴仿真模式

处理建议：若要退出仿真模式，请将 CF 卡内 FA 文件夹内的 override.cfg 文件中的 mask_emul_ax 0x3f 改为 mask_emul_ax 0x00，否则清除报警即可。

1.1.1.15 Alarm 984

故障描述：RTE.CFG 文件中编码器定义非法

故障原因：编码器配置参数错误

处理建议：检查工程配置或 RTE.CFG 文件中的编码器定义

1.1.1.16 Alarm 985

故障描述：RTE.CFG 文件错误

故障原因：RTE.CFG 文件中的一般性错误

处理建议：通过 REPORT 指令或者相关操作检查该错误

1.1.1.17 Alarm 988

故障描述：不好的 RHW.CFG 代码=%d。请使用状态报告了解详情

故障原因：当前控制器的硬件配置，与 RHW.CFG 文件中的配置信息不符

处理建议：

1. 检查 IO 模块等硬件设备是否连接正常；
2. 前往设置->IO 自由配置页面，自点击更新自动匹配硬件信息。

1.1.1.18 Alarm 990

故障描述：系统锁定

故障原因：该步指令导致系统阻塞或控制器故障

处理建议：

1. 检查程序
2. 重启控制器；
3. 检查外部 PLC 等设备是否有干扰；
4. 联系厂家。

1.1.1.19 Alarm 999

故障描述：LICENSE 文件异常

故障原因：编码器配置参数错误

处理建议：请使用读卡器检查 CF 卡 KEY 文件夹内的 LICENSE 文件是否正常

1.2 用户报警及警告

1.2.1 MajorAlarms (1800-1999)

1.2.1.1 Alarm 1800

故障描述：关节运动错误(代码=%d)

故障原因：关节运动错误

处理建议：

代码=1 ：无效轴组。检查控制器轴组设置。
代码=2 ：无效目标点。检查目标点是否合理。
代码=3 ：无效的速度和加速度参数。请重新设置相关参数。
代码=4 ：目标点超出关节运动范围。请重新设置目标点位置。
代码=5 ：运动队列已满。重试或检查控制器。
代码=6 ：运动轴组未执行(3)。重试或检查控制器。

1.2.1.2 Alarm 1801

故障描述：直线运动错误(代码=%d)

故障原因：直线运动错误

处理建议：

1.代码=-1：无效轴组。检查控制器轴组设置；
2.代码=-2：无效目标点。检查目标点是否合理；
3.代码=-3：无效的速度和加速度参数。请重新设置相关参数；
4.代码=-4：目标点超出关节运动范围。请重新设置目标点位置；
5.代码=-5：运动队列已满。重试或检查控制器；
6.代码=-6：运动轴组未执行(3)。重试或检查控制器；

7.代码=-7：保留；

8.代码=-8：开始点接近奇异点。

1.2.1.3 Alarm 1802

故障描述：急停信号

故障原因：急停按钮被按下

处理建议：请检查示教器和电柜上的急停按钮是否被松开

1.2.1.4 Alarm 1803

故障描述：配置错误

故障原因：配置内容出错

处理建议：检查配置的内容重新配置

1.2.1.5 Alarm 1804

故障描述：IO 模块配置错误，请前往设置界面修改，code_h=%u code_l=%u

故障原因：IO 模块配置错误

处理建议：在 IO 模块配置界面重新配置

1.2.1.6 Alarm 1805

故障描述：控制柜过热报警

故障原因：控制柜过热

处理建议：

检查控制柜内风扇、温控开关是否正常工作；

温控开关值设置是否正确。

1.2.1.7 Alarm 1806

故障描述：请开启电柜上的伺服确认按钮

故障原因：电柜上伺服按钮没有开启

处理建议：打开电柜查看伺服确认按钮，并开启电柜上的伺服确认按钮。

1.2.1.8 Alarm 1807

故障描述：区域违反报警

故障原因：机器人违反了用户定义的监控区域

处理建议：

1. 将机器人置于手动模式；
2. 关闭区域监控的控制命令；
3. 报警开关复位；
4. 手动 jog 移动机器人至非违反状态。

1.2.1.9 Alarm 1808

故障描述：轴 %d:轴发生碰撞

故障原因：轴发生碰撞

处理建议：将机器人移动至安全位置

1.2.1.10 Alarm 1809

故障描述：附加轴%d 驱动器报警%x

故障原因：附加轴驱动器错误

处理建议：

1. 请查阅驱动器报警手册中索引对应处理办法；
2. 请检查附加轴传动装置。

1.2.1.11 Alarm 1810

故障描述：附加轴%d 目标位置超过最大限位

故障原因：单步运行的目标位置超过限位或者附加轴没有进行清零

处理建议：重新设定目标位置或对附加轴进行清零操作

1.2.1.12 Alarm 1811

故障描述：外部 PLC 报警

故障原因：外部 PLC 报警

处理建议：检查外部 PLC 状态。

1.2.1.13 Alarm 1812

故障描述：安全门报警

故障原因：安全门报警

处理建议：

1.检查安全门是否被打开并确认工作区域内是否有工作人员；

2.检查安全门信号是否正常。

1.2.1.14 Alarm 1813

故障描述：请检查运动模式是否设置正确

故障原因：请检查运动模式是否设置正确

处理建议：检查附加轴 APP 中，运动模式是否设置正确。

1.2.1.15 Alarm 1814

故障描述：非法状态重启控制器

故障原因：非法状态重启控制器

处理建议：检查机器人当前伺服状态。

1.2.1.16 Alarm 1815

故障描述：mot 程序：非法状态单步运行

故障原因：mot 程序：非法状态单步运行。

处理建议：检查机器人当前伺服状态。

1.2.1.17 Alarm 1816

故障描述：mot 程序：单步运行错误

故障原因：mot 程序：单步运行错误

处理建议：检查当前位置点、工具坐标系、用户坐标系设置是否正确。

1.2.1.18 Alarm 1817

故障描述：mot 程序：打开程序文件错误！

故障原因：mot 程序：打开程序文件错误！

处理建议：检查文件路径是否正确。

1.2.1.19 Alarm 1818

故障描述：mot 程序：读取程序文件错误！

故障原因：mot 程序：读取程序文件错误！

处理建议：检查文件路径是否正确。

1.2.1.20 Alarm 1819

故障描述：动力学：初始化失败

故障原因：动力学辨识文件或负载辨识文件错误。

处理建议：检查动力学辨识文件及负载辨识文件是否正确。

1.2.1.21 Alarm 1820

故障描述：安全板：外部急停信号

故障原因：外部急停按钮被按下

处理建议：检查外部急停按钮是否被松开。

1.2.1.22 Alarm 1821

故障描述：安全板：安全门报警

故障原因：安全门报警

处理建议：

1.检查安全门是否被打开并确认工作区域内是否有工作人员；

2.检查安全门信号是否正常。

1.2.1.23 Alarm 1822

故障描述：安全板：安全光栅报警

故障原因：安全光栅报警

处理建议：

1.检查安全光栅是否被触发并确认工作区域内是否有工作人员；

2.检查安全光栅信号是否正常。

1.2.1.24 Alarm 1823

故障描述：安全板急停信号错误

故障原因：安全板急停信号错误。

处理建议：

1.请检查急停按钮按下后是否复位；

2.按下电柜前面板复位按钮，观察安全板 EMG 与 EMGO 灯是否亮起；

3.检查安全板 X4 连接器与电柜前面板急停按钮接线是否断线

或接触不良；

4.检查安全板 X5 连接器与示教器急停接线是否断线或接触不良

5.更换安全板；

6.重启电柜。

1.2.1.25 Alarm 1824

故障描述：安全板外部急停错误

故障原因：安全板外部急停错误

处理建议：

1.检查外部设备急停被按下后是否复位；

2.按下电柜前面板复位按钮，观察安全板 EEMG 灯是否亮起；

3.检查安全板 X3 连接器与电柜前面板外部急停端子之间的接线是否断开或者接触不良；

4.更换安全板；

5.重启电柜。

1.2.1.26 Alarm 1825

故障描述：安全板手压信号错误

故障原因：安全板手压信号错误

处理建议：

1.按下电柜前板复位按钮后再次按压手压按钮，观察安全板 ENA 灯是否亮起；

2.检查示教器与安全板 X9 连接器之间手压信号接线是否断线或者接触不良；

3.更换安全板；

4.重启电柜。

1.2.1.27 Alarm 1826

故障描述：安全板 STOP1 错误

故障原因：安全板 STOP1 错误

处理建议：

1.按下电柜前面板复位按钮，观察安全板 STOP1 灯是否亮起；

2.检查安全板 X7 连接器与电柜前面板外部急停端子间接线是否断线或接触不良；

3.更换安全板；

4.重启电柜。

1.2.1.28 Alarm 1827

故障描述：安全板 STOP2 错误

故障原因：安全板 STOP2 错误

处理建议：

1.按下电柜前面板复位按钮，观察安全板 STOP2 灯是否亮起；

2.检查安全板 X8 连接器与电柜前面板外部急停端子间接线是否断线或接触不良；

3.更换安全板；

4.重启电柜。

1.2.1.29 Alarm 1828

故障描述：奇异点处机器人停止运行

故障原因：

1. 机器人程序经过奇异点。

2. 点动机器人时，接近奇异点。

处理建议：

1. 请重新示教程序，避开机器人奇异点。

2. 避开奇异点，或者切换到关节空间下运动机器人。

1.2.1.30 Alarm 1829

故障描述：附加轴：龙门轴序号错误

故障原因：龙门轴序号错误

处理建议：重新配置龙门附加轴。

1.2.1.31 Alarm 1830

故障描述：附加轴：龙门轴从动轴跟随误差过大

故障原因：龙门轴从动轴与主动轴的位置误差过大。

处理建议：

1. 重新设置跟随误差阈值。

2. 检查龙门附加轴零点是否丢失。

3. 检查龙门附加轴机械故障。

1.2.1.32 Alarm 1831

故障描述：码垛 ：码垛机器人的 2/3 轴耦合角度超出设置范围

故障原因：指令程序中 2/3 轴角度设置错误

处理建议：

1. 终止程序，清除报警。

2.点动机器人，运动至限位范围内。

3. 然后修改 RPL 程序中的 2 轴或者 3 轴的值，使其在运动范围内。

1.2.1.33 Alarm 1850

故障描述：弧焊：焊机异常

故障原因：焊机异常。

处理建议：

查看焊机报警，解决焊机异常后再开始焊接。

代码 0：总线通讯时，焊机与机器人未连接；模拟量通讯时，焊机异常。

代码 1~100：总线通讯时，焊机面板有错误显示。

1.2.1.34 Alarm 1851

故障描述：弧焊：冷却系统异常

故障原因：焊机冷却系统异常

处理建议：解决冷却系统异常后，再开始焊接。

1.2.1.35 Alarm 1852

故障描述：弧焊：保护气异常。

故障原因：保护气异常。

处理建议：解决保护气异常后，再开始焊接。

1.2.1.36 Alarm 1853

故障描述：弧焊：焊枪发生碰撞。

故障原因：焊枪发生碰撞。

处理建议：

1.依次打开“弧焊 APP-设备设置-碰撞检测开关”，关闭焊枪碰撞检测信号，并清除报警；

2.将焊枪复位；

3.重新开启焊枪碰撞检测信号。

1.2.1.37 Alarm 1854

故障描述：弧焊：工具末端速度超过设定值。

故障原因：当前机器人末端的运动速度，超过设定值。

处理建议：

1.检查机器人是否发生飞车；

2.设置合理的工具末端速度最大值。

1.2.1.38 Alarm 1855

故障描述：弧焊：激光传感器连接错误。

故障原因：

1.连接超时；

2.标定时未连接传感器。

处理建议：

1.检查网络，重新连接；

2.连接正常后再进行标定。

1.2.1.39 Alarm 1856

故障描述：弧焊：激光传感器功能模式执行错误。

故障原因：

1.激光打开失败；

2.激光关闭失败；

3.激光器寻位模式打开失败；

4.激光器寻位模式关闭失败；

5.激光器跟踪模式打开失败；

6.激光器跟踪模式关闭失败。

处理建议：

1.检查激光器连接状态，尝试重新连接后执行；

2.检查激光器是否损坏。

1.2.1.40 Alarm 1857

故障描述：弧焊：激光寻位失败。

故障原因：

1.寻位超时；

2.超出搜寻距离。

处理建议：

1.合理设置搜寻时间；

2.合理设置搜寻距离。

1.2.2 MinorAlarms (3900-3999)

1.2.2.1 Alarm 3900

故障描述：冲压机急停

故障原因：可能是冲压机接受到急停输入信号

处理建议：

1. 冲压机急停输入;
2. 检查信号否正确。

1.2.2.2 Alarm 3901

故障描述：冲压机上死点丢失

故障原因：可能是冲压机未处于最高点

处理建议：

1. 冲压机上死点丢失信号输入。
2. 检查信号是否正确。

1.2.2.3 Alarm 3902

故障描述：非单张料片

故障原因：料片不是单张状态

处理建议：

1. 检测到非单张料片信号;
2. 检查信号是否正确。

1.2.2.4 Alarm 3903

故障描述：冲压机脉宽小于最短时间

故障原因：冲压机脉宽小于最短时间

处理建议：冲压机属性设置：重新设置脉宽时间。

1.2.2.5 Alarm 3904

故障描述：冲压机脉宽大于最短时间

故障原因：冲压机脉宽大于最短时间

处理建议：冲压机属性设置，重新设置脉宽时间。

1.2.2.6 Alarm 3905

故障描述：冲压机非单次模式

故障原因：冲压机处于非单次模式下

处理建议：

1. 检测到冲压机非单次模式信号输入;
2. 检查冲压机属性设置是否合理;
3. 检查信号是否正确。

1.2.2.7 Alarm 3906

故障描述：等待超时

故障原因：可能是冲压机接受到急停输入信号

处理建议：

1. 输入信号等待超时;
2. 检查自定义动作时间设置是否合理。

1.2.2.8 Alarm 3907

故障描述：手动模式下区域被占用

故障原因：手动模式下监控区域内存在机器人

处理建议：区域内机器人移出后，报警自动复位。

1.2.2.9 Alarm 3908

故障描述：运行程序时轴%d 接近限位

故障原因：运行程序时当前轴接近限位。

处理建议：检查轴是否到达限位。

1.2.2.10 Alarm 3909

故障描述：示教器未连接

故障原因：示教器未连接

处理建议：检查示教器是否连接好。

1.2.2.11 Alarm 3910

故障描述：轴%d 发生碰撞

故障原因：轴发生碰撞

处理建议：检查轴是否发生碰撞。

1.2.2.12 Alarm 3911

故障描述：码件掉落

故障原因：码件掉落

处理建议：将掉落的码件放回夹爪上。

1.2.2.13 Alarm 3912

故障描述：冲压：机器人同时进入工作空间

故障原因：机器人同时进入工作空间

处理建议：从头执行程序。

1.2.2.14 Alarm 3920

故障描述：弧焊：焊丝粘丝。

故障原因：粘丝检测开，机器人检测到焊丝粘丝。

处理建议：手动处理粘丝。若需要自动解除粘丝，需要开启粘丝解除功能。

1.2.2.15 Alarm 3921

故障描述：弧焊：焊丝粘丝解除失败。

故障原因：粘丝自动解除开，但粘丝后，机器人解除粘丝失败。

处理建议：手动处理粘丝。设置合理的粘丝解除电流电压。

1.2.2.16 Alarm 3922

故障描述：弧焊：焊接过程断弧。

故障原因：断弧检测开，焊接过程中，发生电弧中断。

处理建议：断弧重启或重新焊接。

1.2.2.17 Alarm 3923

故障描述：弧焊：焊接起弧失败。

故障原因：开始起弧时，起弧失败。

处理建议：将焊件表面处理干净后再执行起弧命令。

1.2.2.18 Alarm 3924

故障描述：弧焊：焊接收弧失败。

故障原因：开始收弧时，收弧失败。

处理建议：设定合理的收弧参数。

1.2.2.19 Alarm 3925

故障描述：弧焊：刮擦起弧失败。

故障原因：刮擦起弧功能开，机器人在起弧点起弧失败，机器人边运动边起弧仍然失败。

处理建议：

1. 将焊件表面处理干净后再起弧；
2. 设置合理的起弧参数。

1.2.2.20 Alarm 3926

故障描述：弧焊：寻位失败，未寻到工件。

故障原因：

1. 寻位距离设置不合理；
2. 焊件表面有油或漆等覆盖物。

处理建议：

1. 设置合理的寻位距离；
2. 清理焊件表面覆盖物。

1.2.2.21 Alarm 3927

故障描述：弧焊：启用电弧跟踪之前请先开启摆弧功能。

故障原因：启用电弧跟踪之前摆弧功能未开启。

处理建议：启用电弧跟踪之前请先开启摆弧功能。

1.2.2.22 Alarm 3928

故障描述：弧焊：参数设置非法，请检查，代码：%d

故障原因：参数设置超出了规定范围。

1. 代码 0：预留；
2. 代码 1：起弧、收弧指令文件号非法；
3. 代码 2：预留；
4. 代码 3：预留；
5. 代码 4：预留；
6. 代码 5：摆弧指令文件号非法；
7. 代码 6：电弧跟踪指令文件号非法。

处理建议：请在规定范围内设置参数。

1.2.2.23 Alarm 3929

故障描述：弧焊：指令使用错误，代码：%d。

故障原因：

1. 代码 0：焊接功能未开启。
2. 代码 1：（1）.起弧失败后未重新执行该指令；（2）.非断弧情况下，机器人产生故障；（3）.非断弧情况下，手动暂停了机器人。
3. 代码 2：执行摆弧功能时检测到间断焊已打开；
4. 代码 3：执行间断焊功能时检测到摆弧已打开。

处理建议：

1. 点击“弧焊”APP，打开焊接功能开关。
2. 将弧焊信号复位，重新执行程序。
3. 摆弧与间断焊功能请勿同时使用。

1.2.2.24 Alarm 3930

故障描述：弧焊：当前机器人运动非笛卡尔运动

故障原因：弧焊焊接过程中，机器人非笛卡尔运动。

处理建议：弧焊过程中选择笛卡尔空间的运动方式，如直线运动，圆弧运动。

1.2.2.25 Alarm 3931

故障描述：pallet2 码垛模式中产品抓取信息有错误

故障原因：高级码垛模式中产品抓取信息有错误。

处理建议：

1. 检查模式文件的产品抓取信息是否正确。
2. 如果初次运行出现此报警，请到码垛生产界面，初始化该工艺流。

1.2.3 Warnings（4900-4999）

1.2.3.1 Warning 4900

故障描述：轴组%s: 非法轴组模式

故障原因：非法轴组模式。

处理建议：

1.2.3.2 Warning 4901

故障描述：轴组%s: 零点位置丢失

故障原因：零点位置丢失报警

处理建议：手动进行零点标定，然后清除警告。

1.2.3.3 Warning 4902

故障描述：XPL API 错误，请在程序日志中查找原因

故障原因：XPL 文件 API 出现错误

处理建议：请查看程序日志，根据日志中的内容排查问题。

1.2.3.4 Warning 4903

故障描述：XPL 程序错误，请在程序日志中查找原因。

故障原因：XPL 程序中存在错误

处理建议：请查看程序日志，根据日志中的内容排查问题。

1.2.3.5 Warning 4904

故障描述：XPL 程序错误，程序运行时远程加载程序。

故障原因：加载当前程序错误: 其他程序正在运行

处理建议：请停止当前运行的程序，然后再进行远程加载程序。

1.2.3.6 Warning 4905

故障描述：XPL 程序错误，远程加载程序不存在

故障原因：远程加载的程序不存在

处理建议：请确保加载的程序存在后重新加载

1.2.3.7 Warning 4906

故障描述：碰撞检测，读取参数文件失败。

故障原因：读取参数文件失败

处理建议：检查/application/collisiondetect/collisionpardata.xml 是否存在或者文件内容是否正常，若不存在则重新配置参数生成文件。

1.2.3.8 Warning 4907

故障描述：碰撞检测，读取配置文件失败。

故障原因：读取配置文件失败

处理建议：检查卡中目录 application/collisiondetect/目录下是否有此机器人配置文件或综合文件手否正确

1.2.3.9 Warning 4908

故障描述：附加轴%d 正在运行，运动指令无效。

故障原因：附加轴运行期间接受到运动指令

处理建议：请在附加轴运行结束后，再发运动指令。

1.2.3.10 Warning 4909

故障描述：读取附加轴参数文件失败

故障原因：读取附加轴参数文件失败提示报警

处理建议：检查卡中目录/application/AuxAxis/auxParData.txt 文件是否存在，不存在则重新配置参数生成文件。

1.2.3.11 Warning 4910

故障描述：写附加轴参数文件失败

故障原因：写附加轴参数文件失败提示报警

处理建议：检查卡中目录/application/AuxAxis/auxParData.txt 文件是否存在以及是否保存成功，不存在则重新配置参数生成文件。

1.2.3.12 Warning 4911

故障描述：在上电情况下修改配置参数

故障原因：在上电情况下修改配置参数提示报警

处理建议：请在手动掉伺服的情况下，点击保存按键。

1.2.3.13 Warning 4912

故障描述：目标位置超过最大限位

故障原因：目标位置超过最大限位提示报警

处理建议：单步运行的目标位置设置不超过限位的值或者对附加轴进行清零

1.2.3.14 Warning 4913

故障描述：读取文件时，打开参数文件失败

故障原因：读取文件时，打开参数文件失败提示报警

处理建议：检查卡中目录/application/AuxAxis/auxParData.txt 文件是否存在，不存在则重新配置参数生成文件。

1.2.3.15 Warning 4914

故障描述：读取附加轴文件错误

故障原因：读取附加轴文件出错报警

处理建议：检查文件中数据项是否为 13 个

1.2.3.16 Warning 4915

故障描述：写文件时，打开参数文件失败

故障原因：写文件时，打开参数文件失败报警

处理建议：检查卡中目录/application/AuxAxis/auxParData.txt 文件是否存在，若不存在则重新生成参数文件。

1.2.3.17 Warning 4916

故障描述：自动模式下区域被占用

故障原因：自动模式下，监控区域内存在机器人。

处理建议：将区域内机器人移出，自动模式下运行机器人。

1.2.3.18 Warning 4917

故障描述：等待取料超时

故障原因：等待取料超时提示报警

处理建议：

1. 请检查取料点是否有料;
2. 请检查信号是否正确。

1.2.3.19 Warning 4918

故障描述：等待放料超时

故障原因：等待放料超时

处理建议：

1. 请检查放料点是否有料;
2. 请检查信号是否正确。

1.2.3.20 Warning 4919

故障描述：工具 1 无反馈

故障原因：工具 1 无反馈提示报警

处理建议：

1. 请检查工具 1 是否有料;
2. 请检查信号是否正确。

1.2.3.21 Warning 4920

故障描述：工具 2 无反馈

故障原因：工具 2 无反馈提示报警

处理建议：

1. 请检查工具 2 是否有料;
2. 请检查信号是否正确。

1.2.3.22 Warning 4921

故障描述：机器人运行时切换模式是非法的

故障原因：机器人在运行过程中被切换了运行模式

处理建议：直接清除报警，不要在机器人运行过程中切换运行模式。

1.2.3.23 Warning 4922

故障描述：读取外部 IO 文件失败

故障原因：读取外部 IO 文件失败提示报警

处理建议：检查卡中目录 application/externalio/externaliopardata.xml 文件是否存在，若不存在则重新配置生成文件。

1.2.3.24 Warning 4923

故障描述：外部 IO 模块组态错误

故障原因：外部 IO 模块组态错误提示错误

处理建议：检查模块数量和顺序是否和 IO 配置中的组态一致

1.2.3.25 Warning 4924

故障描述：外部 IO 设备故障

故障原因：外部 IO 设备发生故障

处理建议：检查外部 IO 设备是否正常

1.2.3.26 Warning 4925

故障描述：外部 IO 模块故障

故障原因：外部 IO 模块出现故障

处理建议：检查外部 IO 模块是否正常

1.2.3.27 Warning 4926

故障描述：外部 IO ECT 计数器错误

故障原因：外部 IO ECT 计数器发生错误

处理建议：检查外部 IO 模块通讯是否正常

1.2.3.28 Warning 4927

故障描述：热插示教器后有错误的钥匙模式

故障原因：示教器热插拔前后模式不同。

处理建议：确保热插拔示教器前后，示教器模式相同。

1.2.3.29 Warning 4928

故障描述：pallet2 码垛配置信息错误或者文件不存在

故障原因：高级码垛配置信息错误或者文件不存在。

处理建议：检查配置文件是否存在或者信息是否正确。

1.2.3.30 Warning 4929

故障描述：pallet2 码垛模式信息错误或者文件不存在

故障原因：高级码垛模式信息错误或者文件不存在。

处理建议：检查模式文件是否存在或者信息是否正确。

1.2.3.31 Warning 4930

故障描述：码件掉落

故障原因：码件掉落

处理建议：将掉落的码件放回夹爪上。

1.2.3.32 Warning 4931

故障描述：pallet2 码垛模式中产品码放信息有错误

故障原因：高级码垛模式中产品码放信息有错误。

处理建议：检查模式文件的产品放置信息是否正确。

1.2.3.33 Warning 4932

故障描述：读取模拟量 I0 配置文件失败

故障原因：读取模拟量 I0 配置文件失败

处理建议：检查配置文件是否存在。

1.2.3.34 Warning 4933

故障描述：轴%d 多圈值丢失，请在零点位置重置多圈值

故障原因：当前轴多圈值丢失，请在零点位置重置多圈值。

处理建议：请勿重启机器人

1. 将机器人回零；
2. 重置多圈值；
3. 清除报警；
4. 重置零点。

1.2.3.35 Warning 4934

故障描述：附加轴%d:请确认远程示教信号是否打开

故障原因：没有附加轴远程确认信号。

处理建议：检查 IO 自由配置界面中附加轴远程示教信号是否存在。

1.2.3.36 Warning 4935

故障描述：附加轴%d:请检查运动模式是否设置正确

故障原因：附加轴运动模式设置错误。

处理建议：检查附加轴 APP 中，运动模式是否设置正确。

1.2.3.37 Warning 4936

故障描述：简单码垛：更新码垛位置错误

故障原因：

1. 码垛序号输入错误；
2. 码垛模式输入错误；
3. 工件编号选择错误；
4. 输入垛盘未标定。

处理建议：

1. 请确认当前输入码垛序号正确（0-8）；
2. 请确认码垛模式输入值符合要求（0/1）；
3. 请确定正确的工件编号；
4. 请选择已标定的垛盘，或标定当前选择垛盘。

1.2.3.38 Warning 4937

故障描述：动力学：读取动力学辨识文件路径错误

故障原因：动力学辨识文件路径错误。

处理建议：查看动力学辨识文件路径是否正确。

1.2.3.39 Warning 4938

故障描述：动力学：读取动力学辨识参数错误

故障原因：动力学辨识参数错误。

处理建议：查看动力学辨识文件参数是否正确。

1.2.3.40 Warning 4939

故障描述：动力学：读取负载辨识辨识文件路径错误

故障原因：负载辨识辨识文件路径错误。

处理建议：查看负载辨识文件路径是否正确。

1.2.3.41 Warning 4940

故障描述：动力学：读取负载辨识辨识参数错误

故障原因：负载辨识辨识参数错误。

处理建议：查看负载辨识文件参数是否正确。

1.2.3.42 Warning 4941

故障描述：动力学：辨识过程中发生异常

故障原因：辨识过程中发生异常。

处理建议：

1. 辨识过程中是否出现异常报警；
2. 检查 RPL 程序是否终止或者正在运行。

1.2.3.43 Warning 4942

故障描述：动力学：请在自动上伺服模式下开启运行轨迹

故障原因：机器人模式或者伺服状态错误。

处理建议：检查机器人模式及伺服状态。

1.2.3.44 Warning 4943

故障描述：动力学：生成负载辨识轨迹错误

故障原因：负载辨识参数错误。

处理建议：检查负载辨识参数是否正确。

1.2.3.45 Warning 4944

故障描述：跟踪视觉：相机自动连接超时

故障原因：跟踪视觉：相机自动连接超时。

处理建议：

1. 检查相机是否正确开启；
2. 检查相机硬件连接是否正常；
3. 检查超时时间设置是否合理；
4. 尝试手动连接。

1.2.3.46 Warning 4945

故障描述：附加轴：在 PLC 控制模式下，设置附加轴运行轴数错误

故障原因：设置附加轴运行轴数错误。

处理建议：在 PLC 控制运动轴数时，请检查设置运行的附加轴是否正确，其中七轴为 1，依次类推。

1.2.3.47 Warning 4946

故障描述：附加轴：在 PLC 控制模式下，在改变附加轴运行轴数之前请将附加轴运行信号按钮关闭

故障原因：在改变附加轴运行轴数之前请将附加轴运行信号按钮关闭。

处理建议：在 PLC 控制运动轴数时，请检查附加轴运行按钮是否被按下。

1.2.3.48 Warning 4947

故障描述：附加轴：在 PLC 控制模式下，伺服关闭后，请将附加轴运行信号按钮关闭

故障原因：伺服关闭后，请将附加轴运行信号按钮关闭。

处理建议：在 PLC 控制运动轴数时，伺服关闭后，请检查附加轴运行按钮是否被按下。

1.2.3.49 Warning 4948

故障描述：远程 IO：不能同时打开汇川 IO 和 EFORT IO

故障原因：远程 IO：不能同时打开汇川 IO 和 EFORT IO。

处理建议：

1. 关闭汇川 IO 或者 EFORT IO。
2. 点击保存按键生成新的配置文件。

1.2.3.50 Warning 4949

故障描述：EFORT IO 板：ARM 初始化异常，设备需要重新启动

故障原因：EFORT IO 板：ARM 初始化异常。

处理建议：ARM 初始化异常，设备需要重新启动。

1.2.3.51 Warning 4950

故障描述：弧焊：尝试重新起弧中。

故障原因：设置起弧次数大于 2 后，当首次起弧失败，机器人将再次起弧。

处理建议：等待机器人自动处理。

1.2.3.52 Warning 4951

故障描述：弧焊：断弧后，尝试回退并重新起弧。

故障原因：断弧重启开，断弧后，机器人回退一定距离再重新起弧。

处理建议：等待机器人自动处理。

1.2.3.53 Warning 4952

故障描述：弧焊：加载配置文件失败，代码：%d。

故障原因：控制器中配置文件缺失，导致配置文件读取失败。

处理建议：在机器人示教器界面重新配置参数文件并保存。

1. 代码 0：加载焊机设置文件失败；
2. 代码 1：加载焊接参数文件失败；
3. 代码 2：加载设备设置文件失败；
4. 代码 3：加载电流特性文件失败；
5. 代码 4：加载电压特性文件失败；
6. 代码 5：加载摆弧文件失败；
7. 代码 6：加载电弧跟踪文件失败。

1.2.3.54 Warning 4953

故障描述：弧焊：将机器人移动到故障点附近后再开始故障启动。

故障原因：故障启动前，机器人末端与故障点位置距离超过设定距离。

处理建议：

1. 将机器人移动到故障点附近；
2. 设置合理的距离范围。

1.2.3.55 Warning 4954

故障描述：弧焊：尝试通过刮擦方式起弧。

故障原因：刮擦启动开，正在进行刮擦启动。

处理建议：等待机器人自动处理。

1.2.3.56 Warning 4955

故障描述：弧焊：请将机器人的模式开关切换至自动模式并使能伺服。

故障原因：PC 端控制机器人运动，模式非自动或伺服未使能。

处理建议：将机器人运动模式切换到自动并伺服使能。

1.2.3.57 Warning 4956

故障描述：弧焊：协议文件与硬件不匹配，请在装置设置中重新配置

故障原因：开机时检测到协议文件与硬件不匹配。

处理建议：在装置设置中重新配置协议并保存重启机器人。

1.2.3.58 Warning 4957

故障描述：弧焊：手动起弧运行失败

故障原因：手动起弧运行失败

处理建议：手动起弧运行失败，尝试检查以下原因。

原因 1：机器人未处于暂停状态；

原因 2：机器人伺服未使能；

原因 3：未执行过起弧指令，当前运动非焊缝路径。

1.2.3.59 Warning 4961

故障描述：EFORT IO 板：系统时钟初始化异常，设备需要重新启动

故障原因：EFORT IO 板：系统时钟初始化异常。

处理建议：设备需要重新启动。

1.2.3.60 Warning 4962

故障描述：EFORT IO 板：系统内部总线初始化异常，设备需要重新启动

故障原因：EFORT IO 板：系统内部总线初始化异常。

处理建议：设备需要重新启动。

1.2.3.61 Warning 4963

故障描述：EFORT IO 板：DI 运行异常，设备需要重新初始化或者重新启动；

故障原因：EFORT IO 板：DI 运行异常。

处理建议：设备需要重新初始化或者重新启动。

1.2.3.62 Warning 4964

故障描述：EFORT IO 板：DO 运行异常，设备需要重新初始化或者重新启动；

故障原因：EFORT IO 板：DO 运行异常。

处理建议：设备需要重新初始化或者重新启动。

1.2.3.63 Warning 4965

故障描述：EFORT IO 板：模拟输出运行异常，设备需要重新初始化或者重新启动；

故障原因：EFORT IO 板：模拟输出运行异常。

处理建议：设备需要重新初始化或者重新启动。

1.2.3.64 Warning 4966

故障描述：EFORT IO 板：模拟输入运行异常，设备需要重新初始化或者重新启动；

故障原因：EFORT IO 板：模拟输入运行异常。

处理建议：设备需要重新初始化或者重新启动。

1.2.3.65 Warning 4967

故障描述：拖动示教：加载拖动配置文件错误

故障原因：拖动示教：加载拖动配置文件错误。

处理建议：检查拖动配置文件是否存在或内容是否有误。

1.2.3.66 Warning 4968

故障描述：拖动示教：加载驱动参数文件错误

故障原因：拖动示教：加载驱动参数文件错误

处理建议：检查驱动参数文件是否存在或内容是否有误。

1.2.3.67 Warning 4969

故障描述：拖动示教：加载末端工具配置文件错误

故障原因：拖动示教：加载末端工具配置文件错误。

处理建议：检查末端设置文件是否存在或内容是否有误。

1.2.3.68 Warning 4970

故障描述：拖动示教：错误 ID:%d：无法开启拖动功能

故障原因：拖动示教：无法开启拖动功能。

处理建议：

1. 确保机器人已上伺服并处于自动模式的执行模式；
2. 检查驱动器的力矩控制状态是否正确；
3. 请先停止机器人再开启拖动示教；
4. 检查动力学模型是否有误，负载设置是否正确；
5. 检查各关节当前位置是否在拖动示教的允许范围内或 2，3 轴位置距离零点是否过近（小于 20 度）；
6. 检查拖动示教配置文件或驱动器力矩控制文件是否有误。

1.2.3.69 Warning 4971

故障描述：拖动示教：拖动过程太危险，关节超出限位或限速

故障原因：拖动示教：拖动过程太危险，关节超出限位或限速。

处理建议：检查拖动过程中的力是否过大或检查拖动配置文件参数设置是否有误。

1.2.3.70 Warning 4978

故障描述：未识别 IO 板

故障原因：控制器未识别扩展 IO 模块

处理建议：检查扩展 IO 模块是否为机器人适配类型，若不是请更换。

1.2.3.71 Warning 4979

故障描述：奇异点处机器人停止运行

故障原因：

1. 机器人程序经过奇异点。
2. 点动机器人时，接近奇异点。

处理建议：

1. 请重新示教程序，避开机器人奇异点。
2. 避开奇异点，或者切换到关节空间下运动机器人。

1.2.3.72 Warning 4980

故障描述：折弯：工件厚度不合格。

故障原因：折弯：工件厚度不合格。

处理建议：请检查工件数量，并重新放置。

1.2.3.73 Warning 4981

故障描述：折弯：折弯刀速度不合法。

故障原因：折弯：折弯刀速度不合法。

处理建议：请重新配置折弯刀速度。

1.2.3.74 Warning 4982

故障描述：动力学：错误 ID:%d：更新摩擦参数失败

故障原因：

1. 摩擦更新时程序并未一直在运行
2. 摩擦更新时不能安装负载
3. 摩擦更新时倍率不得低于 70%
4. 关节实际最大速度超过允许范围
5. 速度未达到最大速度的 65%，无法求解
6. 更新动力学配置文件失败

处理建议：

1. 摩擦参数更新时保持机器人程序运行
2. 拆下负载，并取消负载设定；
3. 增大倍率至 70%以上；
4. 降低倍率至 95%；
5. 增大倍率且增大各轴的运动范围
6. 检查动力学配置文件是否存在及格式是否错误

1.2.3.75 Warning 4983

故障描述：动力学：更新摩擦参数成功，请重启机器人

故障原因：重启机器人控制器

处理建议：重启机器人控制器

1.2.3.76 Warning 4984

故障描述：工具平齐：方向夹角计算超限，请调整工具姿态

故障原因：工具坐标系与参考坐标之间的夹角过大

处理建议：调整工具姿态，使工具坐标系与参考坐标之间的夹角尽可能减少

1.2.3.77 Warning 4985

故障描述：工具平齐：JC 转换失败

故障原因：逆解失败

处理建议：调整工具姿态，使工具坐标系与参考坐标之间的夹角尽可能减少

1.2.3.78 Warning 4986

故障描述：工具平齐：关节角计算超限，请调整工具姿态

故障原因：当前位置与目标位置之间误差过大

处理建议：调整工具姿态，使工具坐标系与参考坐标之间的夹角尽可能减少

1.2.3.79 Warning 4987

故障描述：工具平齐：目标位置不可达

故障原因：目标位置不可达

处理建议：调整工具姿态，使工具坐标系与参考坐标之间的夹角尽可能减少

1.2.3.80 Warning 4988

故障描述：上伺服前没有进行伺服确认操作

故障原因：/

处理建议：

1. 按下电柜面板的伺服确认按钮；
2. 清除示教器关于模式切换的弹窗报警

1.2.3.81 Warning 4989

故障描述：驱动器：关节滑移超限

故障原因：关节滑移超过限值

处理建议：

1. 检查原始绝对值编码器数据是否存在；
2. 记录当前绝对值编码器位置时，是否走到原始位置；
3. 关节滑移超过 5°；

1.2.3.82 Warning 4990

故障描述：轴发生碰撞。

故障原因：轴发生碰撞。

处理建议：检查轴是否发生碰撞。

1.2.3.83 Warning 4991

故障描述：冲压：拆垛失败，代码：%d

故障原因：拆垛失败。

代码 0：尝试拆垛次数大于设定的堆垛数。

代码 1：等待拆垛确认按钮信号示教超时。

处理建议：

1、检查堆垛信号是否存在。

2、检查拆垛确认按钮信号是否正确。

1.2.3.84 Warning 4992

故障描述：零点丢失

故障原因：清零操作造成与其耦合的轴零点丢失

处理建议：1. 对耦合轴进行清零操作

1.2.3.85 Warning 4993

故障描述：安全监控：更新安全监控参数文件失败

故障原因：

1. 更新安全监控（区域监控）参数文件失败提示报警

2. 更新安全监控（安全位置）参数文件失败提示报警

处理建议：

1. 检查卡中目录/Application/areasmonitor/area.xml 文件是否存在以及是否保存成功，不存在则重新配置参数生成文件

2.检查卡中目录/Application/areasmonitor/axsf.xml 文件是否存在以及是否保存成功，不存在则重新配置参数生成文件

附录 2　驱动器报警及警告

2.1　清能驱动 (Alarm:1000-1199，Warning:4050-4099)

2.1.1 报警信息

2.1.1.1 Alarm 1002

故障描述：编码器内部通信异常

故障原因：

1. .编码器发生故障；
2. 电机编码器接线异常（比如断线，未采用屏蔽双绞线，与电机动力线耦合在一起）；
3. 驱动器地线未可靠连接 ；
4. 驱动器周围存在强干扰源。

处理建议：

1. 检查电机编码器接线并确保接线规范正确；
2. 编码器线缆，电机动力增加磁环；
3. 可靠的连接驱动器地线；
4. 更换电机编码器；
5. 移除驱动器周围强干扰源，或者驱动器与周围强干扰源独自供电；
6. 驱动器动力输入电源增加进线滤波器。

2.1.1.2 Alarm 1003

故障描述：驱动器短路

故障原因：

1. 驱动器 U、V、W 输出存在短接现象；
2. 驱动器受干扰导致 DI 信号异常，此为误报现象（地线未接好或电流环调节器参数设置不适合，导致电流振荡引发干扰）；
3. 驱动器损坏（比如 IGBT 短路，电流检测电路异常）。

处理建议：

1. 排查驱动器 U、V、W 接线（比如断开电机动力线缆后再观察驱动器是否仍报短路故障，须在电机抱闸断开的前提下进行以保证机械安全）；
2. 用万用表检查驱动器 IGBT，确认是否短路；
3. 规范布线，尤其是地线；
4. 调节电流环参数；
5. 更换驱动器。

2.1.1.3 Alarm 1004

故障描述：漏电

故障原因：

1. 驱动器 U、V、W 输出对地存在短路现象；
2. 驱动器损坏（比如电流检测电路异常）。

处理建议：

1. 排查驱动器 U、V、W 接线；
2. 更换驱动器。

2.1.1.4 Alarm 1005

故障描述：驱动器 UV 短路

故障原因：驱动器损坏

处理建议：维修或更换驱动器

2.1.1.5 Alarm 1006

故障描述：驱动器 VW 短路

故障原因：驱动器损坏

处理建议：维修或更换驱动器

2.1.1.6 Alarm 1007

故障描述：驱动器 WU 短路

故障原因：驱动器损坏

处理建议：维修或更换驱动器

2.1.1.7 Alarm 1008

故障描述：AD 采样电路异常

故障原因：驱动器损坏

处理建议：维修或更换驱动器

2.1.1.8 Alarm 1009

故障描述：EEPROM 异常

故障原因：参数 CRC 校验错误

处理建议：检查驱动器硬件

2.1.1.9 Alarm 1010

故障描述：栈空间溢出

故障原因：驱动器固件运行错误

处理建议：维修或更换驱动器

2.1.1.10 Alarm 1011

故障描述：参数未初始化

故障原因：驱动器损坏

处理建议：维修或更换驱动器

2.1.1.11 Alarm 1012

故障描述：编码器数据异常

故障原因：

1. 编码器接线错误；
2. 编码器线缆损坏；
3. 编码器 AD 参数正确；
4. 编码器损坏；
5. 编码器参数设置错误。

处理建议：

1. 检查编码器接线
2. 检查编码器线缆；
3. 重新校准编码器 AD 参数；
4. 更换编码器；
5. 检查编码器参数。

2.1.1.12 Alarm 1013

故障描述：转子定位错误

故障原因：

1. 电机转子位置补偿角设置错误；
2. 驱动器损坏。

处理建议：

1. .重新检测电机转子位置补偿角；
2. 维修或更换驱动器。

2.1.1.13 Alarm 1014

故障描述：驱动器内部故障

故障原因：

1. 驱动器硬件异常；
2. 参数设置异常。

处理建议：

1. 更换或维修驱动器；
2. 检查参数设置。

2.1.1.14 Alarm 1015

故障描述：电机抱闸输出异常

故障原因：

1. 电机抱闸接线错误；
2. 驱动器损坏。

处理建议：

1. 检查电机抱闸接线；
2. 维修或更换驱动器。

2.1.1.15 Alarm 1016

故障描述：充电继电器异常

故障原因：驱动器硬件异常

处理建议：更换或维修驱动器

2.1.1.16 Alarm 1017

故障描述：能耗制动接线错误

故障原因：

1. 制动电阻接线不正确；
2. 驱动器硬件异常。

处理建议：

1. 检查制动电阻接线；
2. 更换或维修驱动器。

2.1.1.17 Alarm 1018

故障描述：AD 校正系数异常

故障原因：AD 校正系数设置错误

处理建议：重新设置 AD 校正系数

2.1.1.18 Alarm 1019

故障描述：可编程器件固件匹配错误

故障原因：驱动器损坏

处理建议：维修或更换驱动器

2.1.1.19 Alarm 1020

故障描述：控制板参数与功率板匹配错误

故障原因：驱动器损坏

处理建议：维修或更换驱动器

2.1.1.20 Alarm 1021

故障描述：电机接线相序错误

故障原因：电机相序接线错误

处理建议：检查电机接线相序

2.1.1.21 Alarm 1022

故障描述：系统初始化失败

故障原因：

1. 伺服参数设置错误；
2. 电机编码器接线错误或编码器损坏；
3. 驱动器损坏。

处理建议：

1. 检查伺服参数；
2. 检查电机编码器接线和编码器；
3. 维修或更换驱动器。

2.1.1.22 Alarm 1023

故障描述：编码器内部故障

故障原因：

1. 编码器接线错误；
2. 编码器线缆损坏；
3. 编码器损坏；
4. 编码器参数设置错误。

处理建议：

1. 检查编码器接线；
2. 检查编码器线缆；
3. 更换编码器；
4. 检查编码器参数。

2.1.1.23 Alarm 1024

故障描述：编码器类型变更

故障原因：编码器类型发生变化

处理建议：重启驱动器或软复位

2.1.1.24 Alarm 1025

故障描述：驱动器过流 U

故障原因：

1. 驱动器 U 相输出短路；
2. 电机负载过大
3. 电机绝缘不良；
4. 驱动器损坏。

处理建议：

1. 检查 U 相接线；
2. 降低电机负载；
3. 测量电机绝缘，必要时维修更换；
4. 维修或更换驱动器。

2.1.1.25 Alarm 1026

故障描述：驱动器过流 V

故障原因：

1. 驱动器 V 相输出短路；
2. 电机负载过大；
3. 电机绝缘不良；
4. 驱动器损坏。

处理建议：

1. 检查 V 相接线；
2. 降低电机负载；
3. 测量电机绝缘，必要时维修更换；
4. 维修或更换驱动器。

2.1.1.26 Alarm 1027

故障描述：驱动器过流 W

故障原因：

1. 驱动器 W 相输出短路；
2. 电机负载过大；
3. 电机绝缘不良；
4. 驱动器损坏。

处理建议：

1. 检查 W 相接线；
2. 降低电机负载；
3. 测量电机绝缘，必要时维修更换；
4. 维修或更换驱动器。

2.1.1.27 Alarm 1028

故障描述：直流母线过压

故障原因：

1. 动力电源电压过高；
2. 制动电阻功率过小，阻值过高；
3. 基本电源模块负载过大；
4. 驱动器故障。

处理建议：

1. 检查动力电源电压；
2. 加大制动电阻功率，适当降低阻值；
3. 增大电源模块容量或降低负载；
4. 维修或更换驱动器。

2.1.1.28 Alarm 1029

故障描述：24V 控制电源欠压

故障原因：24V 控制电源电压过低

处理建议：检查控制电源电压

2.1.1.29 Alarm 1030

故障描述：看门狗溢出

故障原因：内部堆栈溢出

处理建议：

1. 重新上电；
2. 维修或更换驱动器。

2.1.1.30 Alarm 1031

故障描述：驱动器持续过载

故障原因：

1. 机械卡阻；
2. 驱动器负载过大；
3. 电机故障；
4. 驱动器故障。

处理建议：

1. 检查机械传动部分，改善机械传动性能；
2. 检查电机负载，或加大电机驱动模块容量；
3. 维修或更换电机；
4. 维修或更换驱动器。

2.1.1.31 Alarm 1032

故障描述：编码器接线错误

故障原因：

1. 编码器接线错误；
2. 编码器线缆损坏。

处理建议：

1. 检查编码器接线；
2. 检查编码器线缆；
3. 更换编码器。

2.1.1.32 Alarm 1033

故障描述：CPU 过载

故障原因：

1. 控制指令超过 CPU 负载能力；
2. 驱动器损坏。

处理建议：

1. 降低控制指令操作频率；
2. 更换或维修驱动器。

2.1.1.33 Alarm 1034

故障描述：电机动力线断开

故障原因：

1. 驱动器 U、V、W 输出存在断线或接线不良等现象；
2. 电机阻抗过大；
3. 驱动器内部电流采样电路异常。

处理建议：

1. 检查电机 U、V、W 接线并确保接线可靠；
2. 更换电机（或关闭驱动器输出缺相检测功能）；
3. 更换驱动器。

2.1.1.34 Alarm 1035

故障描述：编码器操作异常故障

故障原因：

1. 编码器接线错误；
2. 编码器线缆损坏；
3. 编码器损坏；
4. 编码器参数设置错误。

处理建议：

1. 检查编码器接线；
2. 检查编码器线缆；
3. 更换编码器；
4. 检查编码器参数。

2.1.1.35 Alarm 1036

故障描述：驱动器瞬时过载

故障原因：

1. 输出侧短路；
2. 因干扰误动作；
3. 控制参数不合理；
4. 驱动器损坏。

处理建议：

1. 检查输出侧电缆接线；
2. 接线可靠接地；
3. 重新调整控制参数；
4. 维修或更换驱动器。

2.1.1.36 Alarm 1037

故障描述：编码器外部通信发送异常

故障原因：

1. 编码器接线错误；
2. 编码器线缆损坏；
3. 编码器损坏；
4. 编码器参数设置错误。

处理建议：

1. 检查编码器接线；
2. 检查编码器线缆；
3. 更换编码器；
4. 检查编码器参数。

2.1.1.37 Alarm 1038

故障描述：编码器外部通信接收异常

故障原因：

1. 编码器接线错误；

2. 编码器线缆损坏；
3. 编码器损坏；
4. 编码器参数设置错误。

处理建议：

1. 检查编码器接线；
2. 检查编码器线缆；
3. 更换编码器；
4. 检查编码器参数。

2.1.1.38 Alarm 1039

故障描述：驱动器硬件过流

故障原因：

1. 机械卡阻；
2. 电机负载过大；
3. 电机参数或控制参数设置不正确；
4. 电机故障；
5.驱动器故障。

处理建议：

1. 检查机械传动部分，包括电机抱闸，改善机械传动性能；
2. 检查电机负载，或加大电机容量；
3. 检查电机参数和控制参数设置；
4. 维修或更换电机；
5.维修或更换驱动器。

2.1.1.39 Alarm 1040

故障描述：输入缺相故障

故障原因：

1. 输入电源缺相；
2. 参数 0x202C 选择三相输入动力电源，实际接入单相输入动力电源；
3.驱动器输入缺相检测电路损坏。

处理建议：

1. 检查电源电路，主电路在接通状态下某一相电压过低或使用了单相电源；
2.按照实际接入电源设置参数 0x202C；
3.维修或更换驱动器。

2.1.1.40 Alarm 1041

故障描述：直流母线欠压

故障原因：

1. 动力电源输入电压过低；
2. 直流母线接触不良；
3. 驱动器输出侧线缆绝缘不良；
4. 驱动器损坏。

处理建议：

1. 检查动力电源电路；
2. 检查直流母线；
3. 检查驱动器输出侧线缆；

4. 维修或更换驱动器。

2.1.1.41 Alarm 1042

故障描述：逆变功率模块过热

故障原因：

1. 驱动器散热不良；
2. 环境温度过热；
3. 逆变负载过大；
4. 驱动器输出线缆绝缘不良；
5. 驱动器损坏。

处理建议：

1. 检查驱动器散热系统，确认散热孔畅通，散热风扇运行正常；或增加外部散热措施；
2. 保持环境温度正常；
3. 更换更大容量的逆变器；
4. 检查输出线缆，必要时更换；
5. 维修或更换驱动器。

2.1.1.42 Alarm 1043

故障描述：逆变功率模块过冷

故障原因：驱动器损坏

处理建议：维修或更换驱动器

2.1.1.43 Alarm 1044

故障描述：能耗制动过载

故障原因：

1.制动回路容量不足；

2.驱动器损坏。

处理建议：

1. 降低启停频率；延长加/减速时间常数；减小负载惯量；加大驱动器和电机容量；
2. 维修或更换驱动器。

2.1.1.44 Alarm 1045

故障描述：电机持续过载

故障原因：

1. 机械卡阻；
2. 超过电机额定转矩运行时间过长。

处理建议：

1. 检查机械传动部分，查看是否有堵转现象；
2. 检查负载，降低加减速度，或更换更大容量的驱动器和电机。

2.1.1.45 Alarm 1046

故障描述：能耗制动电阻过热

故障原因：

1. 环境温度过高；
2. 启动停止频繁；
3. 制动电阻容量不足。

处理建议：

1. 增加外部散热措施；

2. 延长加减速时间；
3. 更换更大功率的制动电阻。

2.1.1.46 Alarm 1047

故障描述：整流功率模块过热

故障原因：

1. 驱动器散热不良；
2. 环境温度过高；
3. 驱动器损坏。

处理建议：

1. 检查驱动器散热系统，确认散热孔畅通，散热风扇运行正常；
2. 保持环境温度正常；
3. 维修或更换驱动器。

2.1.1.47 Alarm 1048

故障描述：电机 U 相瞬时过载

故障原因：

1. 电机加速度过大；
2. 控制参数设置不当；
3. 电机故障；
4. 驱动器损坏。

处理建议：

1. 适当降低电机加减速度；
2. 优化电机控制参数 ；
3. 维修或更换电机；
4. 维修或更换驱动器。

2.1.1.48 Alarm 1049

故障描述：电机 V 相瞬时过载

故障原因：

5. 电机加速度过大；
6. 控制参数设置不当；
7. 电机故障；
8. 驱动器损坏。

处理建议：

5. 适当降低电机加减速度；
6. 优化电机控制参数 ；
7. 维修或更换电机；
8. 维修或更换驱动器。

2.1.1.49 Alarm 1050

故障描述：电机 W 相瞬时过载

故障原因：

9. 电机加速度过大；
10. 控制参数设置不当；
11. 电机故障；
12. 驱动器损坏。

处理建议：

9. 适当降低电机加减速度；
10. 优化电机控制参数 ；
11. 维修或更换电机；
12. 维修或更换驱动器。

2.1.1.50 Alarm 1051

故障描述：硬件 STO1 触发

故障原因：外部急停输入

处理建议：查找外围故障

2.1.1.51 Alarm 1052

故障描述：硬件 STO2 触发

故障原因：外部急停输入

处理建议：查找外围故障

2.1.1.52 Alarm 1053

故障描述：STO 配线异常

故障原因：STO 配线错误

处理建议：检查 STO 配线

2.1.1.53 Alarm 1054

故障描述：驱动器外部故障

故障原因：当前轴之外的其它轴故障

处理建议：检查其它轴，确定并排除故障。

2.1.1.54 Alarm 1055

故障描述：参数数据异常

故障原因：

1. 参数范围超限；
2. 位置单位设置错误。

处理建议：

1. 检查参数设置是否超出设定的参数范围；
2. 检查位置单位设置。

2.1.1.55 Alarm 1056

故障描述：位置跟随误差过大

故障原因：

1. 编码器接线错误或连接器接触不良；
2. 控制参数不合适；
3. 外部负载波动或干扰过大。

处理建议：

1. 检查编码器接线；
2. 重新调整控制参数；
3. 增加抗干扰措施。

2.1.1.56 Alarm 1057

故障描述：位置控制溢出

故障原因：反馈位置或位置指令超过 32 位有符号数

处理建议：编码器清零后软复位或重启驱动器

2.1.1.57 Alarm 1058

故障描述：速度跟随误差过大

故障原因：

1. 编码器接线错误或连接器接触不良；
2. 控制参数不合适；
3. 外部负载波动或干扰过大。

处理建议：

1. 检查编码器接线；
2. 重新调整控制参数；
3. 增加抗干扰措施。

2.1.1.58 Alarm 1059

故障描述：控制周期参数设置错误

故障原因：EtherCAT 通讯周期小于伺服控制周期

处理建议：调整 EtherCAT 通讯周期或伺服控制周期，使通讯周期大于伺服控制周期。

2.1.1.59 Alarm 1060

故障描述：EtherCAT 过程数据错误

故障原因：

1. 位置目标值和位置实际值差值超过参数 0x6065 设定阈值；
2. 目标轨迹加速度超过参数 0x60C5 设定阈值；
3. 当前机器人位置接近极限位置。

处理建议：

1. 检查实际位置反馈是否有异常；
2. 检查位置指令轨迹，降低加速度或增大 0x60C5 阈值；
3. 重新调整机器人示教点。

2.1.1.60 Alarm 1061

故障描述：写 EEPROM 失败

故障原因：驱动器损坏

处理建议：维修或更换驱动器

2.1.1.61 Alarm 1062

故障描述：寻原点过程指令非法

故障原因：驱动器损坏

处理建议：维修或更换驱动器

2.1.1.62 Alarm 1063

故障描述：EtherCAT 总线指令非法

故障原因：EtherCAT 通讯未完成（未进入 OP 状态）即发送了伺服使能命令

处理建议：故障复位

2.1.1.63 Alarm 1064

故障描述：DriveStarter 通讯异常

故障原因：

1. 串口通讯线受到干扰；
2. 串口通讯线未可靠接地或接触不良。

处理建议：

1. 检查 R485 线缆是否连接正常；

2. 检查 RS485 转换器是否损坏。

2.1.1.64 Alarm 1065

故障描述：EtherCAT 总线通讯异常

故障原因：

1. EtherCAT 网线断开；
2. 以太帧丢失超过参数“EtherCAT 通讯超时检测设定。

处理建议：

1. 检查总线接线；
2. 检查线缆接头是否压接正。

2.1.1.65 Alarm 1066

故障描述：位置硬超限

故障原因：驱动器外接的位置限

处理建议：检查位置指令规划范

2.1.1.66 Alarm 1067

故障描述：正向软限位

故障原因：位置反馈值超过（正向软限位值+定位完成阈值）

处理建议：

1. 如果不需要正向软限位功能，可通过参数 0x2000 禁止；
2. 检查位置指令规划范围。

2.1.1.67 Alarm 1068

故障描述：负向软限位

故障原因：位置反馈值超过（正向软限位值+定位完成阈值）

处理建议：

1. 如果不需要正向软限位功能，可通过参数 0x2000 禁止；
2. 检查位置指令规划范围。

2.1.1.68 Alarm 1069

故障描述：上电位置偏差过大

故障原因：

1. 驱动器掉电后，电机机位置发生了偏移；
2. 对于带电池的电机编码器，未外接电池或电池欠电压。

处理建议：

1. 对于带电池的电机编码器，确保已接入电池且电池电压正常；
2. 检查机械位置是否改变，确认机械零点无异常后可清除。

2.1.1.69 Alarm 1070

故障描述：非法更改伺服参数

故障原因：修改伺服参数超过了限制值

处理建议：在伺服参数可修改范围内修改参数值

2.1.1.70 Alarm 1071

故障描述：编码器电池欠电压故障

故障原因：

1. 编码器未外接电池或电池接线不良；
2. 编码器电池欠电压。

处理建议：

1. 检查编码器电池接线并确保接线可靠；
2. 更换电池；
3. 执行编码器多圈清零命令。

2.1.1.71 Alarm 1072

故障描述：电机超速

故障原因：

1. 反馈速度超过预设定速度，误差超过设定阈值；
2. 编码器异常。

处理建议：

1. 优化电机参数和控制参数；
2. 检查编码器设置和编码器接线。

2.1.1.72 Alarm 1073

故障描述：电压限幅位置跟随误差过大

故障原因：

1. 电机负载变化过快，变化范围过大；
2. 驱动器损坏。

处理建议：

1. 降低电机负载变化率；
2. 维修或更换驱动器。

2.1.1.73 Alarm 1074

故障描述：编码器超速故障

故障原因：

1. 反馈速度超过编码器最大允许转速；
2. 编码器参数或电机控制参数设置不当；
3. 编码器异常。

处理建议：

1. 适当降低电机运行速度；
2. 检查编码器参数和电机控制参数设置；
3. 检查编码器和编码器接线。

2.1.1.74 Alarm 1075

故障描述：电源单元模块通讯异常

故障原因：

1. 电源单元模块与电机模块之间的控制线缆连接不良；
2. 电机抱闸线或外部 24V（STO）对地短路造成电源模块与电机模块之间控制端口烧坏。

处理建议：

1. 检查电源单元模块与电机模块之间的控制线缆连接并确保接线可靠；
2. 检查电机抱闸线或外部 24V（STO）对地是否有短路；
3. 更换驱动器。

2.1.1.75 Alarm 1076

故障描述：上电位置控制溢出

故障原因：

1. 对于带电池的电机编码器，未外接电池或电池欠电压；

2. 驱动器断电之前的控制模式为无限位置控制模式，或速度模式，或转矩模式，且位置已超出允许的范围。

处理建议：

1. 驱动器执行编码器多圈清零命令后重新上电；
2. 无限位置控制模式下，若不想使用该功能，可通过修改伺服参数“位置控制开关”，禁止上电位置控制溢出检测功能。

2.1.1.76 Alarm 1077

故障描述：EtherCAT 总线同步异常

故障原因：

1. 伺服参数 0x20D3 设置不合理；
2. EtherCAT 主站同步模式配置错误。

处理建议：

1. 正确设置伺服参数 0x20D3；
2. EtherCAT 主站正确配置同步模式。

2.1.1.77 Alarm 1078

故障描述：位置规划运行错误

故障原因：位置规划参数设置不合理，比如位置目标值，规划目标减速度。

处理建议：正确设置位置规划参数

2.1.1.78 Alarm 1079

故障描述：驱动器抱闸电路异常

故障原因：

1. 驱动器抱闸输出短路；
2. 驱动器抱闸输出电流过大导致过热；
3. 驱动器抱闸输出断路；
4. 驱动器内部检测电路异常。

处理建议：

1. 检查驱动器抱闸输出接线并确保接线正确可靠；
2. 更换驱动器。

2.1.1.79 Alarm 1080

故障描述：多轴同步异常

故障原因：驱动器内部电路异常

处理建议：更换驱动器

2.1.1.80 Alarm 1081

故障描述：电机制动控制参数设定错误

故障原因：参数设定错误

处理建议：重新设定电机制动控制参数

2.1.1.81 Alarm 1082

故障描述：EtherCAT PDO 配置错误

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.82 Alarm 1083

故障描述：驱动器旋变电路异常

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.83 Alarm 1084

故障描述：控制模式设定错误

故障原因：

1.判断依据：伺服使能时，驱动器检测不到支持设定的控制模式时报错；

2.可能原因:伺服使能时，控制器设定了驱动器不支持的控制模式（各产品支持的控制模式，详见对象字典 0x6502）。

处理建议：伺服使能前，控制器先设定正确的控制模式。

2.1.1.84 Alarm 1085

故障描述：第二编码器接线错误

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.85 Alarm 1086

故障描述：第二编码器电池欠电压故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.86 Alarm 1087

故障描述：电机堵转故障

故障原因：

1.机器人关节出现卡死现象；

2.驱动器使能状态抱闸无 24V 输出；

3.外部抱闸线接触不良或短路；

4.电机抱闸出现损坏。

处理建议：

1.排查机器人关节是否出现卡死现象；

2.检查驱动器使能状态抱闸有无 24V 输出，没有则更换驱动器；

3.检查外部抱闸线有无接触不良或短路；

4.检查电机抱闸是否出现损坏。

2.1.1.87 Alarm 1088

故障描述：转矩监测故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.88 Alarm 1089

故障描述：上电编码器多圈计数溢出

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.89 Alarm 1090

故障描述：编码器多圈计数溢出

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.90 Alarm 1091

故障描述：版本信息初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.91 Alarm 1092

故障描述：EEPROM 信息初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.92 Alarm 1093

故障描述：DriveStarter 栈地址初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.93 Alarm 1094

故障描述：伺服信息初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.94 Alarm 1095

故障描述：历史故障记录初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.95 Alarm 1096

故障描述：伺服参数上电初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.96 Alarm 1097

故障描述：伺服参数版本初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.97 Alarm 1098

故障描述：伺服参数读取初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.98 Alarm 1099

故障描述：重上电有效参数初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.99 Alarm 1100

故障描述：主回路类型初始化故障

故障原因：

1. 驱动器参数内电机驱动模块型号与实际驱动器型号不一致；
2. 驱动器故障。

处理建议：

1. 驱动器参数内电机驱动模块型号与实际驱动器型号不一致，需手动更改；
2. 驱动器损坏，需要更换驱动器。

2.1.1.100 Alarm 1101

故障描述：对象字典链表初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.101 Alarm 1102

故障描述：EtherCAT 主功能初始故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.102 Alarm 1103

故障描述：EtherCAT 中断初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.103 Alarm 1104

故障描述：EtherCAT 主站选择初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.104 Alarm 1105

故障描述：ethercat 通信参数错误

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.105 Alarm 1106

故障描述： I/O 操作初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.106 Alarm 1107

故障描述：系统编码器初始化故障

故障原因：

1. 电机编码器线没有连上驱动器；
2. 编码器接线接触不良；
3. 电机侧编码器不正常；
4. 驱动器故障。

处理建议：

1. 检查电机编码器线有无连上驱动器；
2. 检查编码器接线有无接触不良；
3. 排查电机侧编码器是否正常；
4. 排查驱动器是否正常。

2.1.1.107 Alarm 1108

故障描述：上电位置偏差初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.108 Alarm 1109

故障描述：电子铭牌初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.109 Alarm 1110

故障描述：调节器参数初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.110 Alarm 1111

故障描述：电机参数初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.111 Alarm 1112

故障描述：电机参数转换初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.112 Alarm 1113

故障描述：AD 校正系数初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.113 Alarm 1114

故障描述：电流 AD 转换系数初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.114 Alarm 1115

故障描述：固件版本匹配初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.115 Alarm 1116

故障描述：AD 校正系数检查故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.116 Alarm 1117

故障描述：驱动器额定电流初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.117 Alarm 1118

故障描述：驱动器最大电流初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.118 Alarm 1119

故障描述：驱动器动力电源初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.119 Alarm 1120

故障描述：电压 AD 转换系数初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.120 Alarm 1121

故障描述：驱动器故障保护阈值初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.121 Alarm 1122

故障描述：位置调节器限幅初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.122 Alarm 1123

故障描述：速度调节器输入限幅初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.123 Alarm 1124

故障描述：速度调节器输出限幅初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.124 Alarm 1125

故障描述：电流调节器输出限幅初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.125 Alarm 1126

故障描述：控制周期初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.126 Alarm 1127

故障描述：电机瞬时过载参数初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.127 Alarm 1128

故障描述：制动电阻过载参数初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.128 Alarm 1129

故障描述：驱动器瞬时过载参数初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.129 Alarm 1130

故障描述：驱动器持续过载参数初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.130 Alarm 1131

故障描述：滤波器初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.131 Alarm 1132

故障描述：伺服参数内部转换初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.132 Alarm 1133

故障描述：系统中断配置初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.133 Alarm 1134

故障描述：系统配置结束初始化故障

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.1.134 Alarm 1135

故障描述：编码器初始化失败

故障原因：

1. 电机编码器线没有连上驱动器；
2. 编码器接线接触不良；
3. 电机侧编码器不正常；
4. 驱动器故障。

处理建议：

1. 检查电机编码器线有无连上驱动器；
2. 检查编码器接线有无接触不良；
3. 排查电机侧编码器是否正常；
4. 排查驱动器是否正常。

2.1.1.135 Alarm 1199

故障描述：无记录报警

故障原因：没有找到报警记录

处理建议：查阅驱动器报警手册

2.1.2 警告信息

2.1.2.1 Warning 4050

故障描述：EEPROM 版本变更

故障原因：变更了 EEPROM 版本

处理建议：重新启动驱动器或软复位

2.1.2.2 Warning 4051

故障描述：电机过载告警

故障原因：

1. 电机负载率超过了设定阈值，缺省值为 80%；
2. 电机负载过大。

处理建议：

1. 检查机械，是否有润滑不良或卡堵现象；

2. 更换功率更大的电机。

2.1.2.3 Warning 4052

故障描述：能耗制动过载告警

故障原因：能耗制动电阻功率过小

处理建议：更换更大功率的能耗制动电阻

2.1.2.4 Warning 4053

故障描述：欠压转速限制告警

故障原因：由于驱动器输入电源电压过低而导致电机转速被限制

处理建议：检查输入电源电压

2.1.2.5 Warning 4054

故障描述：直流母线欠压告警

故障原因：直流母线电压过低

处理建议：检查直流母线电压

2.1.2.6 Warning 4055

故障描述：历史故障记录异常告警

故障原因：

1. 历史故障记录异常；
2. 驱动器损坏。

处理建议：

1. 重新启动驱动器或软复位；
2. 维修或更换驱动器。

2.1.2.7 Warning 4056

故障描述：不支持设定控制模式

故障原因：驱动器控制模式设定超过允许范围

处理建议：重新设定参数 0x6060

2.1.2.8 Warning 4057

故障描述：更改了重上电有效参数

故障原因：外部 IO ECT 计数器发生错误

处理建议：重启驱动器或软复位

2.1.2.9 Warning 4058

故障描述：CPU 过载告警

故障原因：驱动器内部故障

处理建议：更换或维修驱动器

2.1.2.10 Warning 4059

故障描述：编码器电池欠电压告警

故障原因：检测到编码器电池电压过低

处理建议：更换编码器电池

2.1.2.11 Warning 4060

故障描述：驱动器内部告警

故障原因：驱动器未经过出厂测试

处理建议：更换驱动器

2.1.2.12 Warning 4061

故障描述：机械原定未标定

故障原因：

1. 发生了编码器电池欠电压故障，且伺服参数 0x2009.Byte3 设定为"检出编码器电池低电 压故障并提示原点未标定；
2. 发生了上电位置偏差过大故障，且用户判定机械原点已丢失；
3. 电机带单圈绝对式编码器，且驱动器未成功执行寻原点命令。

处理建议：驱动器执行回原点操作

2.1.2.13 Warning 4062

故障描述：驱动器未准备好

故障原因：驱动器内部故障

处理建议：维修或更换驱动器

2.1.2.14 Warning 4063

故障描述：编码器外部通信接收告警

故障原因：

1. 电机编码器接线异常（比如断线，未采用屏蔽双绞线，与电机动力线耦合在一起）；
2. 驱动器地线未可靠连接；
3. 驱动器周围存在强干扰源。

处理建议：

1. 检查电机编码器接线并确保接线规范正确；
2. 编码器线缆，电机动力线缆增加磁环；
3. 可靠的连接驱动器地线；
4. 移除驱动器周围强干扰源，或者驱动器与周围强干扰源分开供电；
5. 驱动器动力输入电源增加进线滤波器。

2.1.2.15 Warning 4064

故障描述：编码器外部通信发送告警

故障原因：

1. 电机编码器接线异常（比如断线，未采用屏蔽双绞线，与电机动力线耦合在一起）；
2. 驱动器地线未可靠连接；
3. 驱动器周围存在强干扰源。

处理建议：

1. 检查电机编码器接线并确保接线规范正确；
2. 编码器线缆，电机动力线缆增加磁环；
3. 可靠的连接驱动器地线；
4. 移除驱动器周围强干扰源，或者驱动器与周围强干扰源独自供电；
5. 驱动器动力输入电源增加进线滤波器。

2.1.2.16 Warning 4065

故障描述：编码器内部通信告警

故障原因：

1. 编码器发生故障；
2. 电机编码器接线异常（比如断线，未采用屏蔽双绞线，与电机动力线耦合在一起）；
3. 驱动器地线未可靠连接；
4. 驱动器周围存在强干扰源。

处理建议：

1. 检查电机编码器接线并确保接线规范正确；

2. 编码器线缆，电机动力线缆增加磁环；
3. 可靠的连接驱动器地线；
4. 更换电机编码器；
5. 移除驱动器周围强干扰源，或者驱动器与周围强干扰源独自供电；
6. 驱动器动力输入电源增加进线滤波器。

2.1.2.17 Warning 4066

故障描述：软限位告警

故障原因：位置实际值或者位置目标值超出了伺服参数 0x2004 和伺服参数 0x2005 设定阈值

处理建议：

1. 适当增大伺服参数 0x2004 和伺服参数 0x2005 设定值；
2. 将电机运行到伺服参数 0x2004 和伺服参数 0x2005 规定的范围内；
3. 减小位置目标设定值，确保其位于伺服参数 0x2004 和伺服参数 0x2005 规定的范围内；
4. 若不想使用该功能，可通过伺服参数“位置控制开关”，禁止软限位检测功能。

2.1.2.18 Warning 4067

故障描述：AD 校正系数无效告警

故障原因：驱动器尚未进行 AD 校正

处理建议：重置驱动器 AD 校正系数

2.1.2.19 Warning 4068

故障描述：位置规划参数异常告警

故障原因：位置规划参数设置不合理

处理建议：正确设置位置规划参数

2.1.2.20 Warning 4069

故障描述：轴%d 0xFF43 上电位置偏差过大告警

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.2.21 Warning 4070

故障描述：轴%d 0xFF44 第二编码器电池欠电压告警

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.2.22 Warning 4071

故障描述：轴%d 0xFF45 转矩监测告警

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.2.23 Warning 4072

故障描述：轴%d 0xFF46 AD 采样电路异常告警

故障原因：驱动器故障。

处理建议：更换驱动器。

2.1.2.24 Warning 4099

故障描述：无记录报警

故障原因：没有找到报警记录

处理建议：查阅驱动器报警手册

2.2 禾川驱动（Alarm:1200-1399，Warning:4100-4199）

2.2.1 报警信息

2.2.1.1 Alarm 1201

故障描述：系统参数异常

故障原因：

1. 控制电源电压瞬时下降；
2. 升级驱动器软件之后，部分参数的范围有改动，导致之前存储的范围有改动，导致之前存储的参数超出上下限。

处理建议：

1. 确保电源电压在规格范围内，恢复出厂参数（P20.06 设置为 1）；
2. 如果升级了软件，请先恢复出厂参数。

2.2.1.2 Alarm 1202

故障描述：产品型号选择故障

故障原因：

1. 编码器连接线损坏或链接松动；
2. 无效的电机型号或驱动器型号。

处理建议：

1. 检查编码器接线是否正常，确保接线牢固；
2. 更换成有效的电机型号或驱动器型号。

2.2.1.3 Alarm 1203

故障描述：参数存储中故障

故障原因：

1. 参数读写过于频繁；
2. 参数存储设备故障；
3. 控制电源不稳定；
4. 驱动器故障。

处理建议：

1. 上位装置用通信修改参数并写入 EEPROM 操作过于频繁。请检查通信程序是否存在频繁修改参数并写入 EEPROM 的指令；
2. 检查控制电接线，同时确保控制电源电压在规格范围内。

2.2.1.4 Alarm 1204

故障描述：FPGA 故障

故障原因：软件版本异常

处理建议：查看软件版本是否匹配

2.2.1.5 Alarm 1205

故障描述：hcfa 错误 5 j%d 产品匹配故障

故障原因：

1. 编码器连接线损坏或连接松动；
2. 使用不支持的外部接口如编码器等；
3. 电机型号与驱动器型号功率不匹配；
4. 不存在的产品型号编码。

处理建议：

1. 检查编码器接线是否良好；
2. 更换不匹配的产品；
3. 选择正确的编码器类型或更换其他类型的驱动器；例如设置的电机型号的功率等级大于驱动器的功率等级，或者设置的电机型号的功率等级比驱动器的功率等级差了两级以上会报出这个故障。

2.2.1.6 Alarm 1206

故障描述：hcfa 错误 6 j%d 程序异常

故障原因：

1. 系统参数异常；
2. 驱动器内部故障。

处理建议：EEPROM 故障，恢复出厂参数（P20.06 设置为 1)，重上电。

2.2.1.7 Alarm 1207

故障描述：hcfa 错误 7 j%d 编码器初始化失败

故障原因：上电时检测到编码器信号异常

处理建议：检查编码器接线，或更换编码器线缆。

2.2.1.8 Alarm 1208

故障描述：hcfa 错误 8 j%d 对地短路检测故障

故障原因：

1. UVW 接线错误；
2. 电机损坏；
3. 驱动器故障。

处理建议：

1. 检测线缆 UVW 是否与地短路，如果是则更换线缆；
2. 检测电机线电阻以及对地电阻是否正常，如异常更换电机。

2.2.1.9 Alarm 1209

故障描述：hcfa 错误 9 j%d 过流故障 A

故障原因：

1. 指令输入与接通伺服同步或指令输入过快；
2. 外接制动电阻过小或短路；
3. 电机电缆接触不良；
4. 电机电缆接地；
5. 电机 UVW 电缆短路；
6. 电机烧坏；
7. 软件检测出功率晶体管过电流。

处理建议：

1. 检查指令输入时序，伺服接通“rdy”后输入指令；
2. 测量制动电阻阻值是否满足规格，按说明书要求重新选择合理制动电阻；
3. 检查线缆连接器是否松脱，确保连接器紧固；
4. 检查电机 UVW 线与电机接地线之间的绝缘电阻绝缘不良时更换电机；
5. 检查电机电缆连接 UVW 是否短路，正确连接电机电缆；
6. 检查电机各线缆间电阻阻值是否相同，不同则更换电机；
7. 减小负载。提升驱动器、电和上容量，延长加减速时间。

2.2.1.10 Alarm 1210

故障描述：hcfa 错误 10 j%d 过流故障

故障原因：

1. 指令输入与接通伺服同步或指令输入过快；
2. 外接制动电阻过小或短路:
3. 电机电缆接触不良；
4. 电机电缆接地；
5. 电机 UVW 电缆短路；
6. 电机烧坏；
7 .软件检测出功率晶体管过电流。

处理建议：

1. 检查指令输入时序，伺服接通“rdy”后输入指令；
2. 测量制动电阻阻值是否满足规格，按说明书要求重新选择合理制动电阻；
3. 检查线缆连接器是否松脱，确保连接器紧固；
4. 检查电机 UVW 线与电机接地线之间的绝缘电阻绝缘不良时更换电机；
5. 检查电机电缆连接 UVW 是否短路，正确连接电机电缆；
6. 检查电机各线缆间电阻阻值是否相同，不同则更换电机；
7. 减小负载。提升驱动器、电和上容量，延长加减速时间。

2.2.1.11 Alarm 1212

故障描述：hcfa 错误 12 j%d 增量光电编码器 Z 断线或者绝对值编码器圈数异常

故障原因：增量式编码器:

1. Z 信号接收异常，Z 信号线接线不良或编码器故障导致 Z 信号丢失，绝对式编码器；
2. 绝对式编码器电池供电不足；
3. 参数 P06.47=1（设置为绝对式系统)，未进行编码器初始化操作；
4. 在驱动器断电期间，编码器电机端接线有拔插。

处理建议：

1. 手动旋转电机轴，如果依然报故障，则检查编码器接线，重新接线或更换电缆，或更换编码器，重新上电；
2. 需要确定电池是否正常，若电池电压不足，请更换电池；
3. 将 P20.06 =7 初始化圈数，重新上电。

2.2.1.12 Alarm 1213

故障描述：hcfa 错误 13 j%d 编码器通信异常

故障原因：

1. 通信式编码器断线；
2. 编码器未接地；
3. 通信校验异常。

处理建议：

1. 检查编码器接线，或者更换编码器线缆；
2. 检查编码器是否接地良好。

2.2.1.13 Alarm 1214

故障描述：hcfa 错误 14 j%d 编码器数据异常

故障原因：

1. 串行编码器断线或接触不良；

2. 串行编码器存储数据读写异常。

处理建议：

1. 检查接线；
2. 或者更换编码器线缆。

2.2.1.14 Alarm 1215

故障描述：hcfa 错误 15 j%d 编码器电池电压过低异常

故障原因：编码器电池电压低于 P06.48 设定的阀值，并且 P06.47 的十位设置为 1。

处理建议：更换编码器电池

2.2.1.15 Alarm 1216

故障描述：hcfa 错误 16 j%d 速度偏差过大

故障原因：速度指令和实际测得的速度的绝对差值超过 P06.45 设定的阀值

处理建议：

1. 将 P06.45 的设定值提高间延长；
2. 或者调节增益提高系统的响应；
3、将速度偏差过大阂值功能置为于扮，即 Pos.45=n；将内部位置指令的加减速时。

2.2.1.16 Alarm 1217

故障描述：hcfa 错误 17 j%d 转矩饱和超时

故障原因：转转矩长时间处于饱和状态，持续时间超过 P06.46 设定的阀值。

处理建议：

1. 提高参数 P06.46 设定时长；
2. 检查 UVW 是否断线。

2.2.1.17 Alarm 1218

故障描述：hcfa 错误 18 j%d 控制电欠压

故障原因：控制电输入接线不良，或输入电源故障。

处理建议：

1.检查输入电源及接线；
2.更换驱动器。

2.2.1.18 Alarm 1219

故障描述：hcfa 错误 19 j%d 飞车故障

故障原因：由于接线等错误，导致控制回路发散，导致电机飞车失速。

处理建议：

1. 检查 UVW 以及编码器接线；
2. 检查驱动器、电机，如有必要请更换，并联系厂家检测。

2.2.1.19 Alarm 1220

故障描述：hcfa 错误 20 j%d 过电压

故障原因：

1. 电源电压超过允许范围，AC280V；
2. 制动电阻断线，制动电阻不匹配，导致无法吸收再生能量；
3. 负载惯量超出允许范围；
4. 驱动器损坏。

处理建议：

1. 输入正确的电压范围；
2. 检查是否已连接外置电阻。测量外置电阻的阻值是否已经断开，确保接线正确，如

果是电阻已烧毁，则建议更换功率更大的外置电阻(可联系厂家获取相关建议)；

3. 延长加减速时间，或者根据负载惯量重新选择合适的驱动器和电机。

2.2.1.20 Alarm 1221

故障描述：hcfa 错误 21 j%d 欠电压

故障原因：

1. 电源电压下降；
2. 发生瞬时停电；
3. 欠压保护阂值(P06.36)设置偏高；
4. 驱动器损坏 (注:这个故障默认不存储记录，可通过 P07.19 设定是否存储)。

处理建议：

1. 提升电源电压容量，确保电源电压稳定；
2. 确认电源电压正常的情况下，检查欠压保护阂值 C P06.36)设置是否偏高。

2.2.1.21 Alarm 1222

故障描述：hcfa 错误 22 j%d 电流采样故障

故障原因：驱动器内部电流采样故障

处理建议：更换伺服驱动器

2.2.1.22 Alarm 1223

故障描述：hcfa 错误 23 j%d AI 采样电压过大

故障原因：

1. AI 接线错误；
2. 外部输入电压偏高。

处理建议：正确连接 AI 输入，将输入电压设定在士 10V 以内。

2.2.1.23 Alarm 1224

故障描述：hcfa 错误 24 j%d 过速

故障原因：

1. 速度指令超过了最高转速设定值；
2. UVW 相序错误；
3. 速度响应严重超调；
4. 驱动器故障。

处理建议：

1. 降低速度指令；
2. 检查 UVW 相序是否正确；
3. 调整速度环增益，减少超调；
4. 更换驱动器。

2.2.1.24 Alarm 1225

故障描述：hcfa 错误 25 j%d 电角度辨识失败

故障原因：

1. 负载或惯量太大；
2. 编码器接线有误。

处理建议：

1. 减小负载或加大电流环增；
2. 更换编码器线缆。

2.2.1.25 Alarm 1226

故障描述：hcfa 错误 26 j%d 惯量辨识失败故障

故障原因：

1. 负载或惯量太大，电机不能按照规定的曲线运行；
2. 辨识过程中出现其他故障导致辨识终止。

处理建议：

1. 减小负载或加大电流环增益；
2. 保证辨识过程正常。

2.2.1.26 Alarm 1227

故障描述：hcfa 错误 27 j%d DI 端子参数设置故障

故障原因：

1. 不同的物理 DI 端子重复分配了同一 DI 功能；
2. 物理 DI 端子与通信控制的 DI 功能同时存在分配。

处理建议：

1. P04.01~P04.09 中有同一功能配置到多个物理 DI 端子的情况；
2. P04.01~P04.09 中分配的功能，与 P09.05 ~P09.08 中相应的二进制位同时启用，请参考 P09.05 ~ P09.08 的使用方法；重新分配 DI 功能配置到多个物理 DI 端子的的情况。

2.2.1.27 Alarm 1228

故障描述：hcfa 错误 28 j%d DO 端子参数设置故障

故障原因：不同的 DO 重复分配了同一输出

处理建议：重新设定电机制动控制参数

2.2.1.28 Alarm 1240

故障描述：hcfa 错误 40 j%d 伺服 ON 指令无效故障

故障原因：执行了让电机通电的辅助功能后，仍然从上位机输入了伺服 0N 命令。

处理建议：改变不当的操作方式

2.2.1.29 Alarm 1242

故障描述：hcfa 错误 42 j%d 分频脉冲输出过速

故障原因：超过了硬件允许的脉冲输出上限

处理建议：更改分频输出设置功能码，使得在伺服工作的整个速度范围内，分频输出脉冲频率不会超限。

2.2.1.30 Alarm 1243

故障描述：hcfa 错误 43 j%d 位置偏差过大故障

故障原因：

1. 伺服电机的 UVW 接线；
2. 伺服驱动器增益较低；
3. 位置指令脉冲的频率较高；
4. 位置指令加速过大；
5. 位置偏差超出位置偏差过故障值(P00.19)设置的值过小；
6. 伺服驱动器/电机故障。

处理建议：

1. 确认电机主电路电缆的接线，重新接线；
2. 确认伺服驱动器增益是否过低，提高增益；
3. 尝试降低指令频率后再运行降低位置指令频率、指令加速度或调整电子齿轮比；

4. 降低指令加速度后再运行加入位置指令加减速时间参数等平滑功能；

5. 确认位置偏差故障值(P00.19)是否合适，正确设定(P00.19 值；

6. 后台查验运行图形，若有输入没反馈请更换伺服驱动器。

2.2.1.31 Alarm 1244

故障描述：hcfa 错误 44 j%d 主回路输入缺相

故障原因：

1. 三相输入线缆接触不良；

2. 缺相故障，即在主电源 ON 状态下，R\S\T 相的某一相电压过低的状态持续了 1 秒以上。

处理建议：

1. 检查三相电源输入的线缆是否连接稳固(注意安全，不要带电操作)；

2. 测量三相电源各相的电压，确保输入电源三相平衡或者确保输入电源电压符合规格。

2.2.1.32 Alarm 1245

故障描述：hcfa 错误 45 j%d 驱动器输出缺相

故障原因：

1. 电机 UVW 接线不良；

2. 电机损坏，出现断路。

处理建议：

1. 检查 UVW 接线；

2. 更换伺服电机。

2.2.1.33 Alarm 1246

故障描述：hcfa 错误 46 j%d 驱动器过载

故障原因：带载运行超过驱动器反时限曲线，原因如下:

1. 电机 UVW 线或编码器线不良或者连接松动；

2. 电机堵转或者被外力驱动，如机械卡死、碰撞，重力或别的外力拖动，或者机械制动器(抱闸)没有打开就运行；

3. 多台驱动器配线时，误将别的同一台电机 UVW 线和编码器线连接到不同的驱动器上；

4. 负载过大，驱动器或电机选型偏小；

5. 可能缺相或相序接错；

6. 驱动器或电机损坏。

处理建议：

1. 确认电机 UVW 线和编码器接线是否存在问题；

2. 确认电机没有堵转或被外力驱动，确认机械制动器（抱闸）已经打开；

3. 确认多台驱动器和电机没有出现交叉配线，即没有出现一台电机 UVW 线和编码器线连接到不同的驱动器上；

4. 延长加减速时间，重新选择合适的驱动器或电机；

5. 检查电机输出的 UVW 是否接错，是否对地短路；

6. 更换驱动器或者电机。

2.2.1.34 Alarm 1247

故障描述：hcfa 错误 47 j%d 电机过载

故障原因：带载运行超过驱动器反时限曲线，原因如下:

1. 电机 UVW 线或编码器线不良或者连接松动；
2. 电机堵转或者被外力驱动，如机械卡死、碰撞，重力或别的外力拖动，或者机械制动器(抱闸)没有打开就运行；
3. 多台驱动器配线时，误将别的同一台电机 UVW 线和编码器线连接到不同的驱动器上；
4. 负载过大，驱动器或电机选型偏小；
5. 可能缺相或相序接错；
6. 驱动器或电机损坏。

处理建议：

1. 确认电机 UVW 线和编码器接线是否存在问题；
2. 确认电机没有堵转或被外力驱动，确认机械制动器（抱闸）已经打开；
3. 确认多台驱动器和电机没有出现交叉配线，即没有出现一台电机 UVW 线和编码器线连接到不同的驱动器上；
4. 延长加减速时间，重新选择合适的驱动器或电机；
5. 检查电机输出的 UVW 是否接错，是否对地短路；
6. 更换驱动器或者电机。

2.2.1.35 Alarm 1248

故障描述：hcfa 错误 48 j%d 电子齿轮设定错误

故障原因：电子齿轮比超过规格范围[编码器分辨率//l0000000，编码器分辨率/2.5]

处理建议：设置正确的齿轮范围

2.2.1.36 Alarm 1249

故障描述：hcfa 错误 49 j%d 散热器过热

故障原因：

1. 风扇损坏；
2. 环境温度过高；
3. 过载后通过关闭电源对过载故障复位，并持续多次；
4. 伺服驱动器的安装方向、与其它伺服驱动器的间隔不合理；
5. 伺服驱动器故障；
6. 驱动器或电机损坏。

处理建议：

1. 运行时风扇是否运转，更换风扇或驱动器；
2. 测量环境温度改善伺服驱动器的冷却条件，降低环境温度；
3. 查看故障记录，是否有报过载故障，变更故障复位方法，过载后等待 30s 后再复位。驱动器、电机选用功率过小，提高驱动器、电机容量，加大加减速时间，降低负载；
4. 确认伺服驱动器的设置状态，根据伺服驱动器的安装标准进行安装；
5. 断电 5 分钟后重启是否依然报故障，重启后如果仍报故障请更换伺服驱动器。

2.2.1.37 Alarm 1250

故障描述：hcfa 错误 50 j%d 脉冲输入异常

故障原因：

1. 输入频率大于脉冲输入最大频率设定值；
2. 输入脉冲受到干扰。

处理建议：

1. 更改最大允许频率，参数 P06.38；

2. 后台软件查看指令是否异常，检查线路接地情况，确保线路可靠接地，信号采用双绞屏蔽线，输入线和动力线分开布线。

2.2.1.38 Alarm 1251

故障描述：hcfa 错误 51 j%d 全闭环位置偏差过大

故障原因：

1. 外部编码器异常；
2. 相关设置过于保守。

处理建议：

1. 确认外部编码器线连接是否正确，更换外部编码器；
2. 全闭环偏差过大，保护功能设置有误确认相关参数的设置重新设置相关参数。

2.2.1.39 Alarm 1254

故障描述：hcfa 错误 54 j%d 用户强制故障

故障原因：通过 DI 功能 32 (FORCE ERR)强制进入故障状态

处理建议：正常的 DI 功能输入，配置了 DI 功能 32 且输入有效。断开输入即可解除故障。

2.2.1.40 Alarm 1255

故障描述：hcfa 错误 55 j%d 绝对位置复位故障

故障原因：绝对位置编码器绝对位置复位故障

处理建议：联系厂家获取技术支持

2.2.1.41 Alarm 1256

故障描述：hcfa 错误 56 j%d 主电源断电

故障原因：停电或主电源线路异常(注:这个故障默认不存储记录，可通过 P07.19 设定是否存储)

处理建议：检查输入主电源是否有瞬间掉电提升电源电压容量

2.2.1.42 Alarm 1260

故障描述：hcfa 错误 60 j%d 写入定制版程序之后第一次启动

故障原因：在已经有标准程序的驱动器下载入定制版程序之后第一次启动

处理建议：恢复出厂值，以便载入定制参数。

2.2.1.43 Alarm 1265

故障描述：hcfa 错误 65 j%d CAN 总线关闭

故障原因：CAN 总线断开或者接收或发送异常

处理建议：检查接线，重新连接。

2.2.1.44 Alarm 1266

故障描述：hcfa 错误 66 j%d 异常的 NMT 命令

故障原因：伺服 ON 时收到 NMT 丁停止命令或复位命令

处理建议：NMT 下节点复位，不要在伺服 ON 时停止或复位 CAN 节点。

2.2.1.45 Alarm 1267

故障描述：hcfa 错误 67 j%d CAN 总线故障

故障原因：CAN 总线断开或者接收或发送异常

处理建议：检查接线，重新连接。

2.2.1.46 Alarm 1271

故障描述：hcfa 错误 71 j%d 节点保护或者心跳超时

故障原因：节点保护和心跳监控到达设定的间没有收到相应的应答

处理建议：检查节点是否在线，NMT 下节点复位。

2.2.1.47 Alarm 1272

故障描述：hcfa 错误 72 j%d 同步失效

故障原因：CANOpen IP 模式下与上位机同步失效

处理建议：NMT 下节点复位，或者 6040 发送故障复位命令。

2.2.1.48 Alarm 1273

故障描述：hcfa 错误 73 j%d CANOpen 轨迹缓冲区下溢

故障原因：CANOpen IP 或 CSP 模式时，同步时钟丢失 2 次以上。

处理建议：检查通信线路是否有干扰，确认上位机正常运行。NMT 下节点复位，或者 6040 发送故障复位命令。

2.2.1.49 Alarm 1274

故障描述：hcfa 错误 74 j%d CANOpen 轨迹缓冲区上溢

故障原因：CANOpen IP 或 CSP 模式时，同步时钟过快，或者实际的时钟频率与配置值不一致。

处理建议：检查通信线路是否有干扰，确认上位机正常运行，确认时钟频率与配置值一致。NMT 下节点复位，或者 6040 发送故障复位命令。

2.2.1.50 Alarm 1275

故障描述：hcfa 错误 75 j%d 从站初始化失败

故障原因：EtherCAT 从站初始化失败

处理建议：请检查从站配置参数

2.2.1.51 Alarm 1276

故障描述：hcfa 错误 76 j%d 同步失败

故障原因：EtherCAT 同步失败

处理建议：NMT 节点复位，或者 6040 发送故障复位命令。

2.2.1.52 Alarm 1277

故障描述：hcfa 错误 77 j%d EtherCAT 通讯中断

故障原因：通讯连续丢失最大次数超过设定值

处理建议：6040 发送故障复位命令

2.2.1.53 Alarm 1278

故障描述：hcfa 错误 78 j%d 指令给定异常

故障原因：指令速度值超过 6080h 设置值

处理建议：NMT 节点复位，或者 6040 发送故障复位命令。

2.2.1.54 Alarm 1279

故障描述：hcfa 错误 79 j%d 使能时无控制模式

故障原因：伺服使能，6060h 为不支持的控制模式。

处理建议：NMT 节点复位，或者 6040 发送故障复位命令。

2.2.1.55 Alarm 1399

故障描述：无记录报警

故障原因：驱动器无记录报警

处理建议：查阅驱动器报警手册

2.2.2 警告信息

2.2.2.1 Warning 4180

故障描述：欠电压警告

故障原因：母线电压较低时输出的警告状态

处理建议：

1. 检查输入主电源是否正常；
2. 调低欠压检测点参数 P06.36。

2.2.2.2 Warning 4181

故障描述：驱动器过载警告

故障原因：带载运行超过驱动器反时限曲线，原因如下:

1. 电机 UVW 线或编码器线不良或者连接松动；
2. 电机堵转或者被外力驱动，如机械卡死、碰撞，重力或别的外力拖动，或者机械制动器(抱闸)没有打开就运行；
3. 多台驱动器配线时，误将别的同一台电机 UVW 线和编码器线连接到不同的驱动器上；
4. 负载过大，驱动器或电机选型偏小；
5. 可能缺相或相序接错；
6. 驱动器或电机损坏。

处理建议：

1. 确认电机 UVW 线和编码器接线是否存在问题；
2. 确认电机没有堵转或被外力驱动，确认机械制动器（抱闸）已经打开；
3. 确认多台驱动器和电机没有出现交叉配线，即没有出现一台电机 UVW 线和编码器线连接到不同的驱动器上；
4. 延长加减速时间，重新选择合适的驱动器或电机；
5. 检查电机输出的 UVW 是否接错，是否对地短路；
6. 更换驱动器或者电机。

2.2.2.3 Warning 4182

故障描述：电机过载警告

故障原因：带载运行超过驱动器反时限曲线，原因如下:

1. 电机 UVW 线或编码器线不良或者连接松动；
2. 电机堵转或者被外力驱动，如机械卡死、碰撞，重力或别的外力拖动，或者机械制动器(抱闸)没有打开就运行；
3. 多台驱动器配线时，误将别的同一台电机 UVW 线和编码器线连接到不同的驱动器上；
4. 负载过大，驱动器或电机选型偏小；
5. 可能缺相或相序接错；
6. 驱动器或电机损坏。

处理建议：

1. 确认电机 UVW 线和编码器接线是否存在问题；
2. 确认电机没有堵转或被外力驱动，确认机械制动器（抱闸）已经打开；
3. 确认多台驱动器和电机没有出现交叉配线，即没有出现一台电机 UVW 线和编码器线连接到不同的驱动器上；
4. 延长加减速时间，重新选择合适的驱动器或电机；
5. 检查电机输出的 UVW 是否接错，是否对地短路；
6. 更换驱动器或者电机。

2.2.2.4 Warning 4183

故障描述：需要重新接通电源的参数变更

故障原因：变更了需要重新接通电源方可生效的参数

处理建议：重新上电

2.2.2.5 Warning 4184

故障描述：伺服未准备好

故障原因：伺服未准备好时伺服 ON

处理建议：检测到伺服 READY 时再给使能

2.2.2.6 Warning 4185

故障描述：写 E2PROM 频繁操作警告

故障原因：程序非正常频繁操作 E2PROM

处理建议：减少 EEPROM 写入操作频率，可以改用不存储 EEPROM 的通信写指令。

2.2.2.7 Warning 4186

故障描述：正向超程警告提示

故障原因：

1. Pot 和 Not 同时有效,一般在工作台上不会同时出现的；
2. 伺服轴在某方向上出现超程状态，可自动解除。

处理建议：正向限位开关被触发，检查运行模式，给负向指令离开正向限位，会自动清除警告(安全防范，超程时禁止人工转动电机)。

2.2.2.8 Warning 4187

故障描述：负向超程警告提示

故障原因：

1. Pot 和 Not 同时有效,一般在工作台上不会同时出现的；
2. 伺服轴在某方向上出现超程状态，可自动解除。

处理建议：负向限位开关被触发，检查运行 模式，给正向指令离开负向限 位，会自动清除警告(安全防范，超程时禁止人工转动电机)。

2.2.2.9 Warning 4188

故障描述：位置指令过速

故障原因：

1. 电子齿轮比设置过大；
2. 脉冲频率过高。

处理建议：

1. 减少设定的电子齿轮比；
2. 减少输入脉冲频率。

2.2.2.10 Warning 4190

故障描述：绝对值编码器角度初始化警告

故障原因：编码器角度重新初始化时偏离过大(大于 7.2 度电角度)警告

处理建议：更换电机

2.2.2.11 Warning 4193

故障描述：能耗制动过载

故障原因：能耗制动功率过载

1. 制动电阻接线错误或接触不良；
2. 使用内置电阻的情况有可能出现默认短接线脱落情况；

3. 制动电阻容量不足；
4. 制动电阻阻值过大导致长时间制动；
5. 输入电压超过规定；
6. 制动电阻阻值、容量、或发热时间常数设置错误；
7. 伺服驱动器故障。

处理建议：

1. 检查制动电阻接线是否正常；
2. 检查内置电阻接线是否正常；
3. 增大制动电阻容量；
4. 减少制动电阻阻值；
5. 减少输入的电压值；
6. 按规格设定合适的参数；
7. 更换伺服驱动器。

2.2.2.12 Warning 4194

故障描述：外接再生泄放电阻过小

故障原因：

1. 外接再生泄放电阻小于驱动器要求的最小值；
2. 参数设置错误。

处理建议：

1. 按规格配置外接再生泄放电阻的功率；
2. 查看参数 P00.21～P00.24 参数是否正确。

2.2.2.13 Warning 4195

故障描述：紧急停止

故障原因：触发了紧急停止

处理建议：正常的 DI 功能输入，配置了 DI 功能 30 且输入有效。断开输入即可解除警告。

2.2.2.14 Warning 4196

故障描述：原点回归错误

故障原因：

1. 搜索原点的时间超过了 P08_95 的设定值；
2. P08.90 参数设置为 3、4 或 5，且碰到限位；
3. 不以限位为原点时，两次碰到限位。

处理建议：

1. 加大 P08.95 设定值；
2. 回原点搜索速度过快导致，减小回原点搜索的速度 P08.92，P08.93。

2.2.2.15 Warning 4197

故障描述：编码器电池欠压

故障原因：编码器电池电压低于 P06.48 设定的阀值

处理建议：检查更换编码器电池

2.2.2.16 Warning 4199

故障描述：无记录报警

故障原因：没有报警的记录

处理建议：

附录 3　编程指令说明

3.1 编程限制说明

1. 变量名最多可达 63 个字符(8 位长度)
2. 变量注释最多 63 个字符(8 位长度)
3. 子程序名最多可以是 31 个字符(8 位长度)
4. 子程序可以有多达 99 个变量
5. 模块名最多可以是 31 个字符(8 位长度)
6. 一个模块最多可以有 99 个变量
7. 一个模块最多可以有 99 个 extern 变量声明
8. 任务可以同时处理多达 8 个 PULSE 脉冲指令
9. 任务可以处理多达 16 个中断
10. 表达式最多可以 127 个字符(8 位长度)

3.2 数据类型

3.2.1 变量声明的数据类型

3.2.1.1 BOOL 布尔

此类数据用于表示逻辑值。**BOOL** 值可以是 **true** 或 **false**。初始化后的默认值为 **false**。

示例:

```
BOOL firstRun
…
IF firstRun = false THEN
firstRun:= true ;
…
END_IF ;
```

当 firstRun 为 **false** 时，执行后，被置为 **true**。

3.2.1.2 DINT 双精度整数

此类数据用于整数值，可以是正数或负数；整数值用 32 位值表示。

DINT 值可以在-2147483648 和 2147483647 之间。

初始化后的默认值为 **0**。

示例:

```
DINT counter
DINT diff
…
counter := counter + 1 ;
…
```

3.2.1.3 UDINT 无符号双精度整数

此类数据用于仅为正数的整数值。整数值用 32 位值表示。

UDINT 值可以在 0 到 4294967295 之间。

初始化后的默认值为 0。

示例:

```
DINT counter
…
counter := counter + 1 ;
```

3.2.1.4 LREAL 长实数

此类数据用于具有双精度的十进制数。该值是 64 位有符号值。

LREAL 值可以在-1.7976931348623158e + 308 和 1.7976931348623158e + 308 之间。

初始化后的默认值为 **0.0**。

示例:

```
LREAL width
…
width := 88.506 ;
...
```

3.2.1.5 VECT3 三维实向量

这种类型的数据用于表示三维向量。

数据由(**x，y，z**) 3 个 **LREAL** 类型值组成。

初始化后的默认值是一个向量，其中所有分量 **x，y，z** 都设置为 **0.0**。

示例:

```
VECT3 pose
…
pose := VECT3( 100, 200, 300) ;
```

3.2.1.6 POINTC 笛卡尔空间位姿

这种类型的数据用于表示一个笛卡尔点。数据具有 **LREAL** 类型的 **x，y，z，a，b，c** 分量。分量 **x，y，z** 表示位置，分量 **a，b，c** 表示姿态。分量 **a** 指向 **z** 轴方向，**b** 指向 **y** 轴方向，**c** 是指向 **x** 轴。需使用由 6 个 **LREAL** 值组合的数据才可以使用 **POINTC** 函数。**POINTC** 可用于笛卡尔运动和关节运动。初始化后的默认值是无效点（在移动指令中直接使用会发出错误），且所有分量 **x，y，z，a，b，c** 都设置为 **0.0**。

示例:

```
POINTC target
…
target := POINTC( 100, 200, 300, 10, 20, 30) ;
MLIN (target, v250, fine, tool0, wobj0) ;
```

3.2.1.7 ROBOT 机器人

此类数据用于与系统中定义的轴组进行交互。例如，可以设置轴组的参考系。要设置此类型的数据，必须使用函数 **ROBOT**，并使用字符串型轴组名作为其参数。

初始化后的默认值是无效的 **ROBOT**。

示例:

```
ROBOT conveyor
REFSYS cvywobj
POINTC cvyOrigin
…
conveyor := ROBOT(“conveyor”) ;
cvywobj:= SETROBOTWOBJ(conveyor, “”, cvyOrigin) ;
…
```

在这个例子中，设置了 conveyor 的参考系。进行此设置是为了在程序中指定 conveyor

的参考系，并在例如跟踪应用中使用。

3.2.1.8 REFSYS 参考坐标系

这种类型的数据用于定义笛卡尔空间运动的参考坐标系。如果在运动指令中使用，则位置将是基于特定的坐标系。参考系可以是固定的或可移动的。使用 **REFSYS** 函数需要父参考系和六个 **LREAL** 型数值。初始化后的默认值是一个所有值都为 **0.0** 的 **REFSYS**，与世界参考系相同。

示例:

```
REFSYS wobj
POINTC p0
…
wobj:= REFSYS( wobj0,100, 200, 50, 0, 0, 0) ;
p0 := POINTC(0, 0, 0, 0, 0, 0) ;
MLIN (p0, v500, fine, tool0, wobj) ;
```

在此示例中，机器人将移动到参考系统 wobj 的原点。

3.2.1.9 POINTJ 关节位置

这种类型的数据用于确定机器人的关节位置，由 **LREAL** 型的 j1，j2，j3，j4，j5，j6 组成。需要 6 个 **LREAL 型**值并使用 **POINTJ** 函数设置此类数据。**POINTJ** 类型用于 **MJOINT** 指令中，将机器人移动到关节位置定义的特定位置。**POINTJ** 类型不能用作笛卡尔空间运动的目标。

初始化后的默认值是无效点（在移动指令中直接使用会发出错误），j1，j2，j3，j4，j5，j6 都被设置为 **0.0**。

示例:

```
POINTJ startpos
…
startpos:= POINTJ( 0, 0, 0, 0, -90, 0) ;
MJOINT(startpos, v500, fine, tool0) ;
```

3.2.1.10 TRIGGER 触发

TRIGGER 数据类型用于存储与移动指令相关事件中涉及到的数据。通过指令 **TRIGSET** 可以将数据存储在 **TRIGGER** 型变量中，指令 **TRIGON** 可以在执行运动指令之前激活触发器。初始化后的默认值是无效触发器。

示例:

```
TRIGGER trig
…
Trig := TRIGSET when Distance is 20 do (io.output[5] := true) ;
…
TRIGON (trig) ;
```

MLIN (target, v500, fine, tool1, wobj1) ;

在此示例中，当机器人在距离目标点 20 mm 时，io.output [5]被设置为 **true**。

3.2.1.11 STRING 字符串

此类数据用于存储字符串。**STRING** 类型最多可包含 128 个字符。初始化后的默认值为空字符串。

示例:

STRING str

…

str := “Hello world” ;

MESSAGE (“%1”, str) ;

此示例在用户日志中显示包含内容为“Hello world”的字符串 str。

3.2.1.12 SPEED 速度

SPEED 数据类型用于指定移动指令中的目标速度。包括切向速度（单位 mm/s）和姿态角速度（degree/s）。要将数据存储在 **SPEED** 型的变量中，需要使用 **SSPEED** 函数。

初始化后的默认值为所有参数值均为 **0.0**。

示例:

SPEED vel

…

vel := SSPEED(250, 5) ;

MLIN (target, vel, fine, tool0) ;

在此示例中，机器人将以 250 mm / s 切向速度和 5 degree/ s 的姿态角速度移动。

示例:

SPEED vel

…

vel := v250 ;

…

MLIN (target1, vel, fine, tool0) ;

MLIN (target2, vel, fine, tool0) ;

MLIN (target3, vel, fine, tool0) ;

此示例显示如何使用变量 vel 中定义的预定义速度为一组运动定义目标速度。

3.2.1.13 ZONE 过渡混合

ZONE 数据类型是用于指定两个连续移动指令之间的混合参数。它包含以 mm 为单位的线性距离和以 degree 为单位的重整定姿态角距离。要设置此类型的数据，需要使用 **SZONE** 函数。初始化后的默认值为所有参数值均为 **0.0**。

示例:

```
ZONE blend
…
blend := SZONE(100, 5) ;
…
MLIN (target1, v250, blend, tool0) ;
MLIN (target2, v250, fine, tool0) ;
```

在此示例中，机器人在距离 target1 100 mm 时开始向 target2 移动。

3.2.1.14 TOOL 工具

此类数据用于描述工具参数。**TOOL** 数据类型用于移动指令。要设置此类数据，需要使用 **STOOL** 函数。初始化后的默认值是 **TOOL**，初始化后的默认值为所有参数值均为 **0.0**。。

示例:

```
TOOL gun
…
gun := STOOL(0, 0, 100, 0, 0, 0) ;
…
MLIN (target, v250, fine, tool0) ;
MLIN (target, v250, fine, gun) ;
```

此示例可用于检验使用相同目标点和两个不同工具的情况下，机器人位置的不同。

3.2.1.15 INTR 中断处理

这种类型的数据用于中断处理。要将此类型的变量与特定中断程序绑定，必须使用 **INTRSET** 指令。可以使用 **INTRCOND** 指令或 **INTRERRNO** 指令指定导致中断的条件。请注意，此类数据必须在程序执行时只设置一次，否则将发出错误。有关其他信息，请参阅以 **INTR** 开头的指令。初始化后的默认值是无效的 **INTR**。

示例:

```
INTR intr1
BOOL firstRun
...
(* set interrupt only once *)
IF NOT firstRun THEN
firstRun := true ;
INTRSET (intr1, intHnd()) ;
INTRCOND (intr1, io.input[8]) ;
END_IF ;
...
```

此示例显示如何绑定中断以及如何设置触发该中断的条件。

3.2.1.16 CLOCK 时间测量时钟

CLOCK 数据类型用于时间测量。时间测量以秒表示。无法直接设置或读取此类数据。

可以使用 **CLOCKRESET** 指令重置时间测量。必须使用指令 **CLOCKSTART** 开始测量，使用 **CLOCKSTOP** 来停止测量。必须使用 **CLOCKREAD** 函数来读取返回的 **LREAL** 型值。初始化后的默认值是 **0.0** 秒的测量值。

示例:

```
CLOCK clk1
…
CLOCKRESET (clk1) ;
CLOCKSTART (clk1) ;
…
CLOCKSTOP (clk1) ;
MESSAGE (“Time elapsed %1”, CLOCKREAD(clk1)) ;
```

此示例显示如何测量执行指令期间所经过的时间。

3.2.1.17 TRACKING 跟踪

TRACKING 数据类型用于存储跟踪应用程序的数据。设置此类型数据必须使用 **TRACKING** 函数。有关此参数的说明，请参阅跟踪应用的说明文档。初始化后的默认值是无效的 **TRACKING**。

示例:

```
TRACKING cvy
REFSYS wobj
REFSYS rsPhoto
…
cvy := TRACKING(1000, 700, 100, 1000, Linear, 0.1) ;
…
wobj := GETTRKWOBJ(rsPhoto, cvy) ;
…
MLIN (target, v500, fine, tool1, wobj) ;
…
```

此示例显示如何为跟踪应用程序设置跟踪参数。

3.2.1.18 EPOINTC (笛卡尔空间位姿,附加轴关节位置点)

这种类型的数据用于表示笛卡尔点和附加轴关节位置。它包含与 **POINTC** 类型相同的数据以及与附加轴关节位置相关的值 **ea1**，**ea2**，**ea3**，**ea4**，**ea5**，**ea6**。**EPOINTC** 可用于笛卡尔空间和关节空间运动。可通过不同的函数设置这种类型的数据，其中一个函

数是 **EPOINTC**。初始化后的默认值是无效点（如果在运动指令中直接使用则发出错误）并且所有值都设置为 **0.0**。

示例:

```
EPOINTC ep0
…
ep0 := EPOINTC( 100, 200, 300, 10, 20, 30, 100, 0, 0, 0, 0,0);
…
MLIN (ep0, v500, fine, tool0) ;
```

3.2.1.19 EPOINTJ（关节位置,附加轴关节位置点)

这种类型的数据用于定义参考包含附加轴关节位置的机器人关节的位置。它包含与 **POINTJ** 类型相同的数据以及与附加轴关节位置相关的 **ea1**，**ea2**，**ea3**，**ea4**，**ea5**，**ea6**。**EPOINTJ** 用于 **MJOINT** 指令中，将机器人移动到关节位置定义的指定位置。**EPOINTJ** 不能用作笛卡尔运动的目标。可通过不同的函数设置此类数据，其中一个是函数 **EPOINTJ**。初始化后的默认值是无效点（如果在运动指令中直接使用则发出错误）并且所有值都设置为 **0.0**。

示例:

```
EPOINTJ ep0
…
ep0 := EPOINTJ( 0, 0, 0, 0, -90, 0, 10, 0, 0, 0, 0, 0) ;
…
MJOINT (ep0, v500, fine, tool0) ;
```

3.2.1.20 TIMER 计时器

TIMER（计时器）数据类型用于等待一定时间。计时器的的时间单位是秒。该中数据类型无法被直接设置和读取。

启动定时器必须使用指令 TIMERSTART。

要停止定时器必须使用指令 TIMERSTOP。

为了读取剩余时间，必须使用返回 LREAL 值的 TIMERR 函数。

读取定时器是否过期必须是返回 BOOL 值的用户 TIMERQ 函数。

计时器变量进过初始化后，是一个保留 0.0 秒剩余时间， 并且未过期的计时器。

示例:

```
TIMER tmr1 ;
…
TIMERSTART (tmr1) ;
…
MESSAGE (“Time remain %1”, TIMERR(tmr1)) ;
```

此示例显示如何测量执行指令期间所经过的时间。

3.2.2 机器环境派生的外部变量的数据类型

3.2.2.1 POINT_L

这种类型的数据用于访问定义在系统加载的库中的点。该数据可用于运动指令中，作为目标点。

示例:

```
POINT_L p0 ; …
MLIN (p0, v500, fine, tool0) ;
```

在这个例子中，机器人将移动到定义在系统加载的库中的 p0 点。

注意：从 XPL 2.1.9 版本开始不再支持该类型。

3.2.2.2 PATN_L

这种类型的数据用于访问定义在系统加载的库中的轨迹。该数据可用于指令 EPATH 中。

示例:

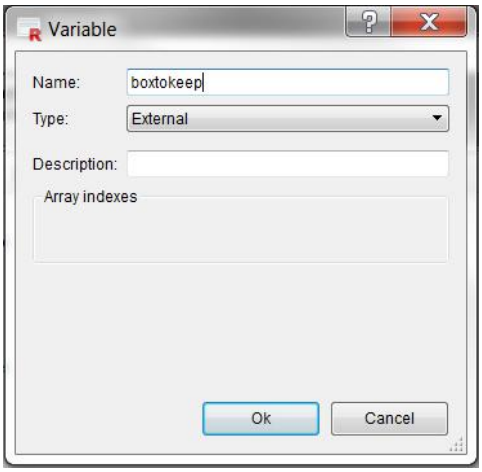
```
PATH_L path0 ;
 …
EPATH (path0, v500, fine, tool0) ;
```

在这个例子中，机器人将执行定义在系统加载的库中的轨迹 path0。

3.3 变量

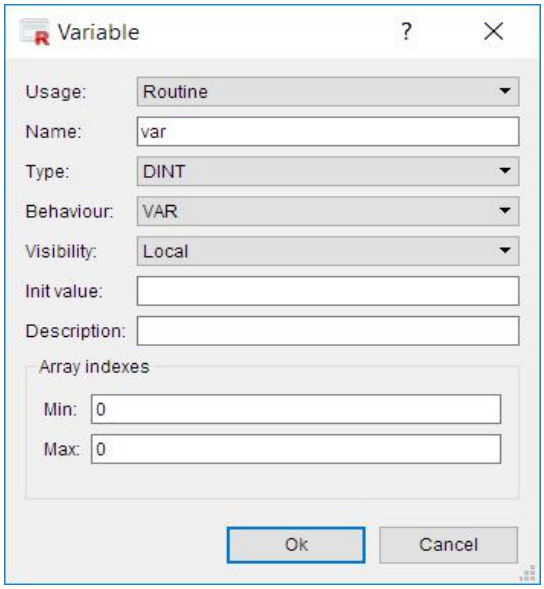
变量能够（在变量标签栏中）被 RPL 程序或其子程序来定义。变量的类型可以是任意的 RPL 数据类型。一个变量可以定义一个数组，其中要求数组的最大索引值要大于最小索引值。

External 外部变量是设备程序中定义，可以为主程序和子程序使用（External 可认为是全局变量）。变量的名称和类型需要在机器人程序中进行定义。为了添加已定义的外部变量类型，必须知道此变量的名称。外部变量的类型不能选择且只能依赖于设备程序中定义。此类型的变量可以用于设备程序和机器人 RPL 程序的数据交互。



变量的定义包括以下三方面：

- 类型
- 行为
- 可见域



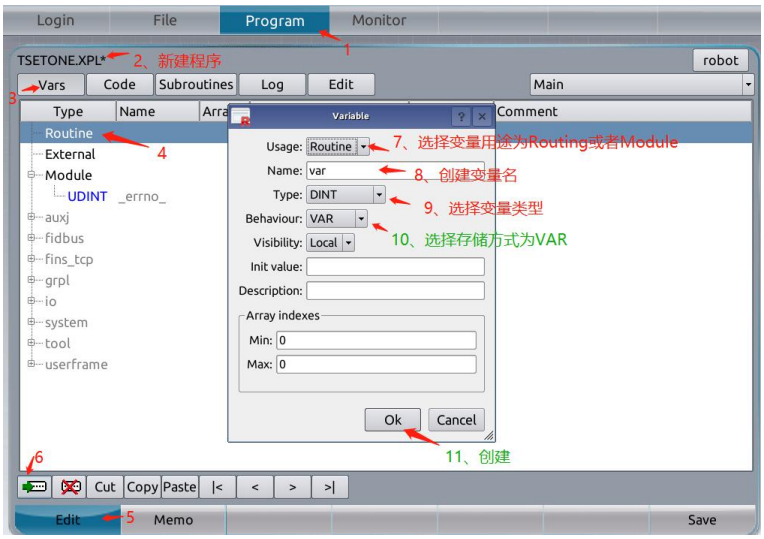
3.3.1 变量行为

3.3.1.1 VAR

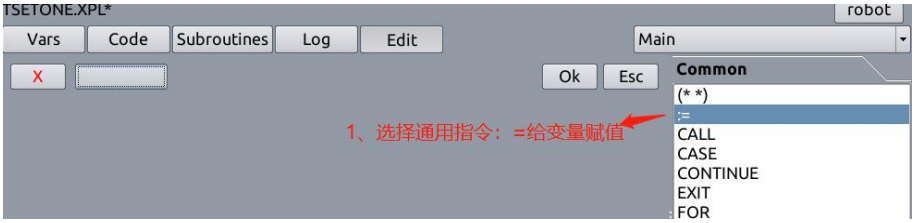
这是变量的默认行为。变量可以在程序中设置，并且当程序重新启动时，其值将丢失。

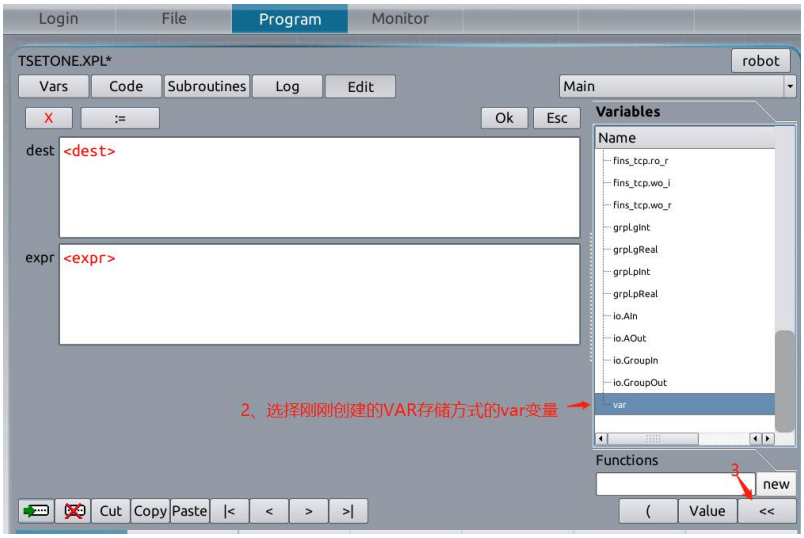
示例:

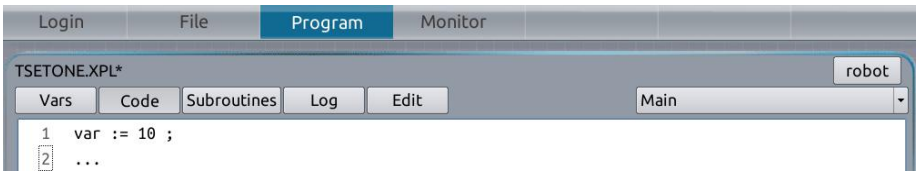
创建变量:



使用变量:





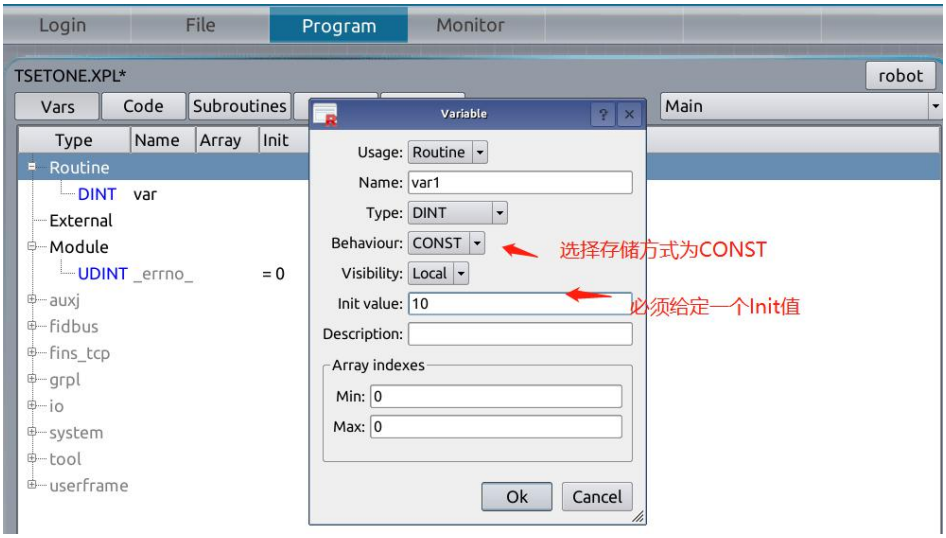


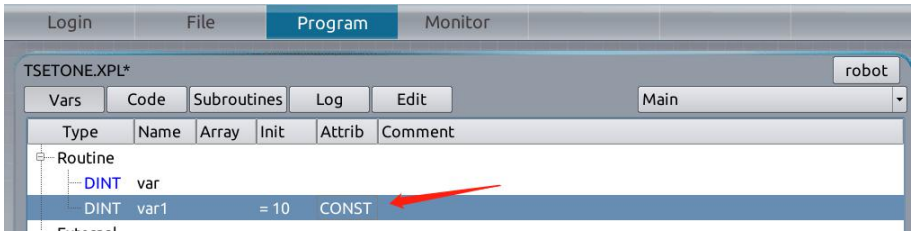
3.3.1.2 CONST

具有此行为的变量不能在程序中进行更改，并且必须设置为 Init 值。

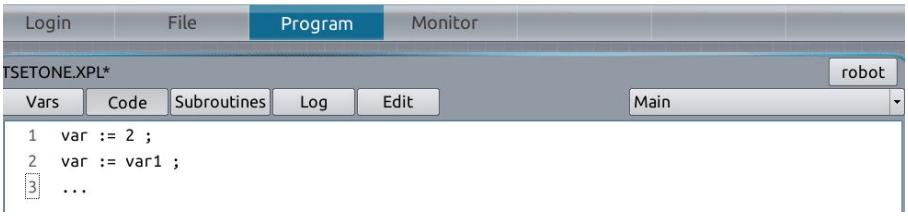
示例:

创建变量:





使用变量:

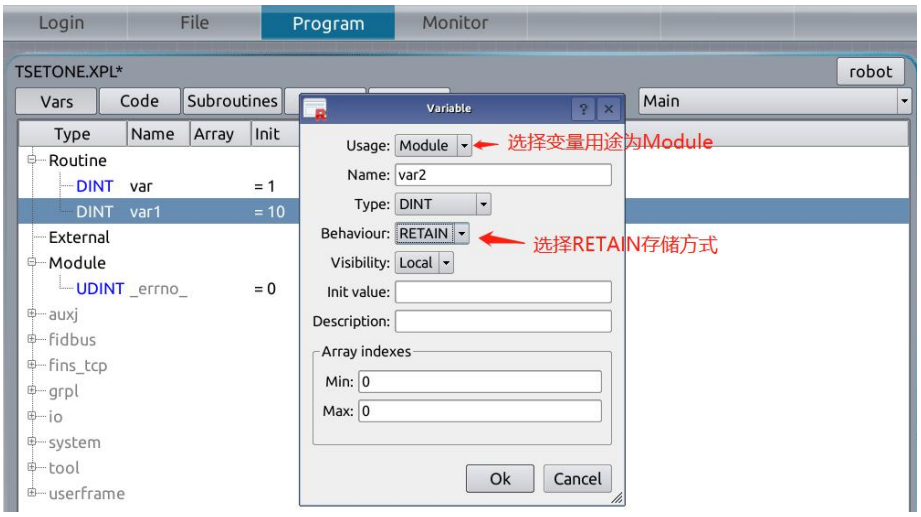


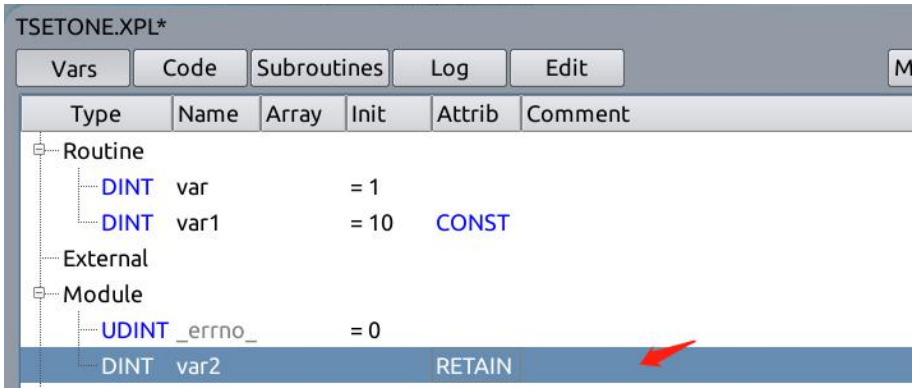
3.3.1.3 RETAIN

当程序重启时变量值被保留下来，当程序从内存下卸载时，此值被保存；对于 **LOCAL** 和 **TASK** 变量，此值将会保存在程序中。变量的此种行为只在 **Module** 域中使用。

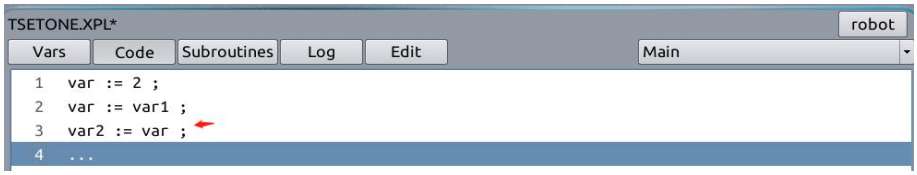
示例:

创建变量:





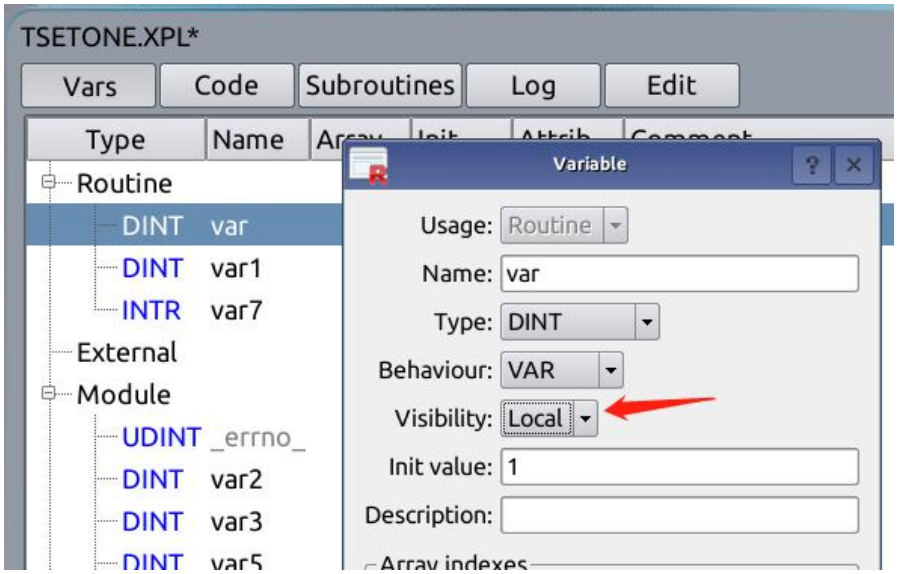
使用变量:



3.3.2 变量的可见域

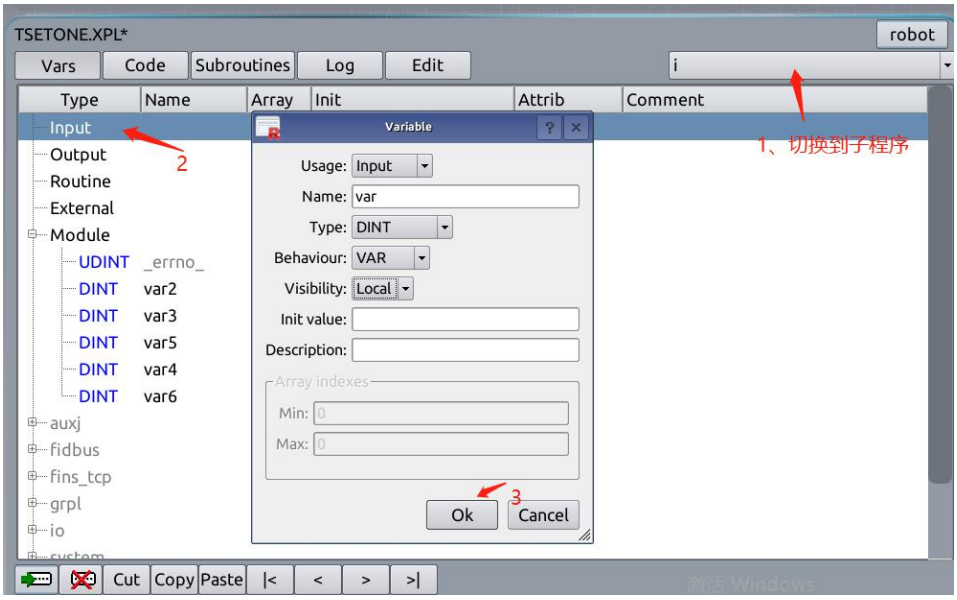
3.3.2.1 Routine's Local

此类型的变量只能在其定义的程序和子程序中可见和使用，不能被其他程序和子程序可见和使用。可以在多个子程序中定义同一个变量名的变量，子程序中定义的变量只能被自己使用。



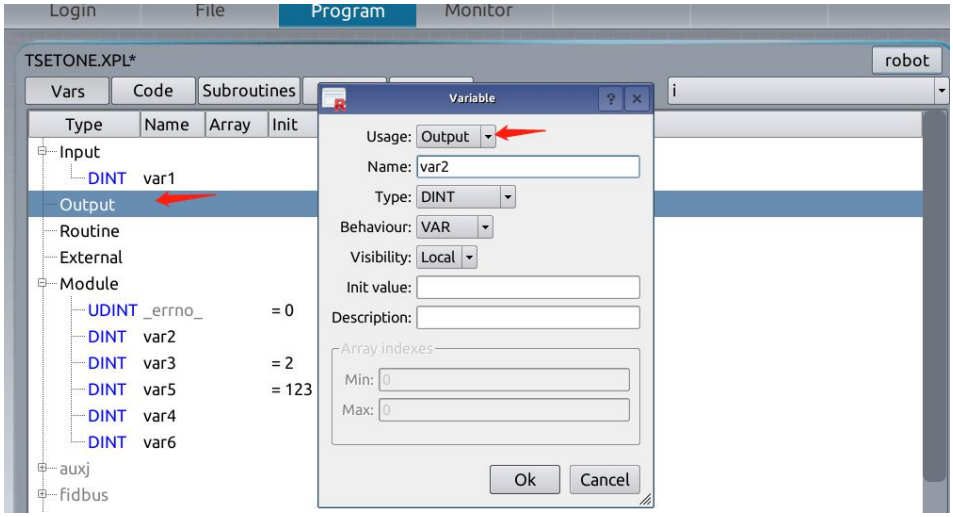
3.3.2.2 Routine's Input

该输入变量用于定义子程序的输入参数。输入变量的使用类似局部变量但是其初始值来源于调用的程序。输入变量定义顺序与调用指令传递参数顺序一致。主程序中不能够使用此类变量。



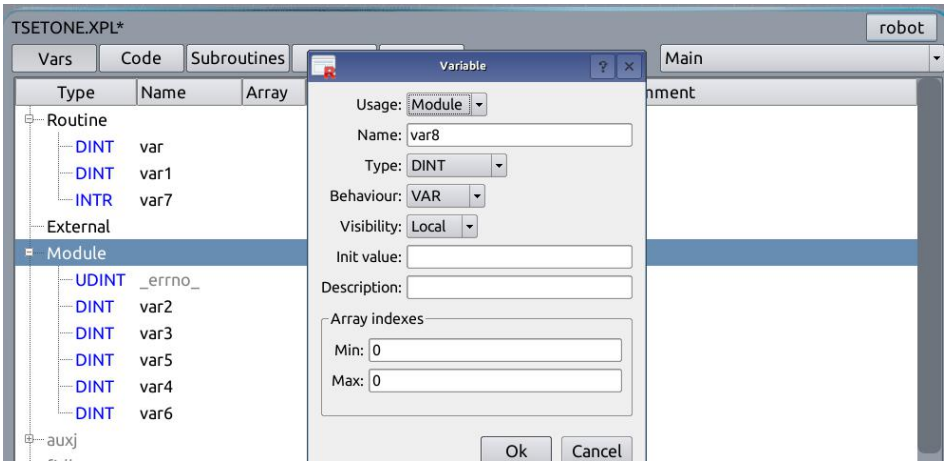
3.3.2.3 Routine's Output

该输出变量用于定义子程序的输出参数。输入变量的使用类似局部变量但是其初始值来源于调用的程序。输出变量定义顺序应与调用指令设置的参数顺序一致。主程序中不能够使用此类变量。



3.3.2.4 Module's Local

此类变量可以被所有程序和子程序使用。**Module** 局部变量可以在一个子程序中设置，然后通过另外一个子程序读取。不能使用相同的名字定义多个 **Module** 局部变量。这些变量对于其他 **Module** 不可见。

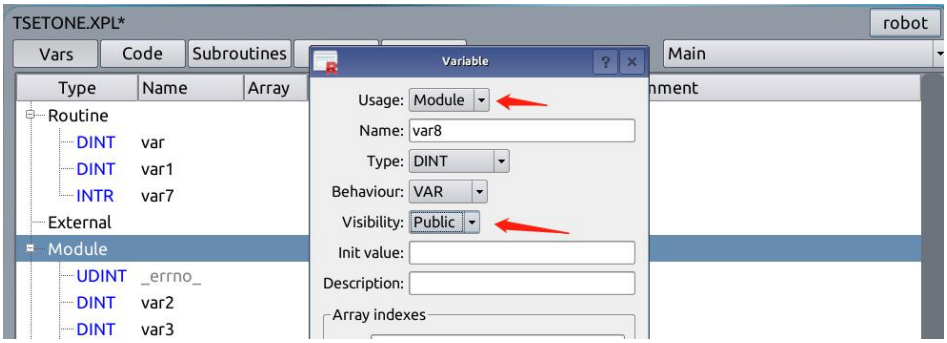


3.3.2.5 Module's Public

同 **Module** 局部相似，但是这个变量可以从其他 **Module** 中看到。在其他模块中，可以在模块名称前面使用这种变量（例如 moduleName.variableName）。

```
localVar: = Utility.moduleVar
```

LocalVar 变量通过 Utility Module 中变量 moduleVar 进行设置。

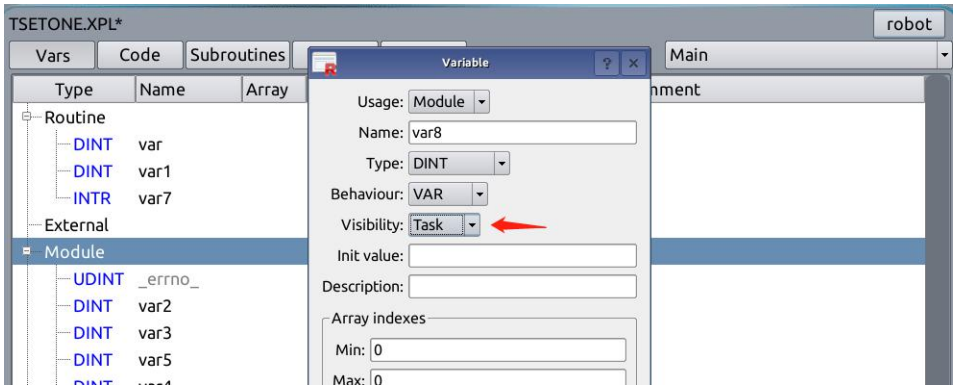


3.3.2.6 Task

类似 **Module** 公共变量，在其他 **Module** 中，不需要使用 **Module** 名字即可直接访问。

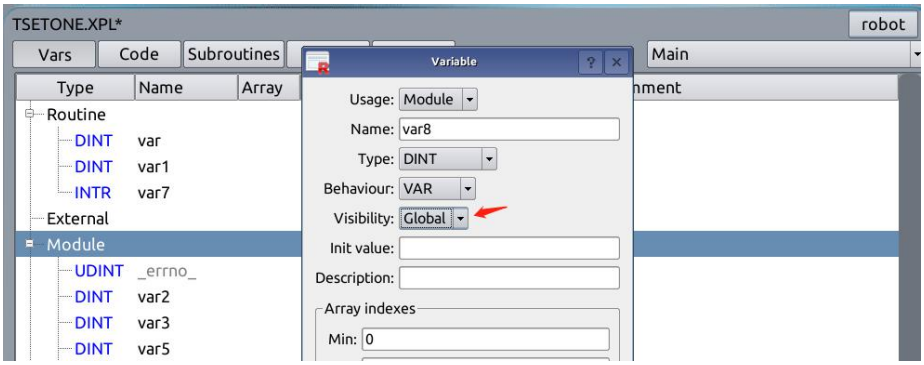
```
localVar := moduleVar
```

使用 moduleVar 设置 localVar，其中 moduleVar 的 module 没有指定。



3.3.2.7 Global

这种变量对于所有系统中的 TASK 都是通用的。在不同的 TASK 之间共享数据很有用。如果对同一 **GLOBAL** 有不同的定义，则会报告错误。



3.3.3 变量初始值

在新建变量时，可以为一些类型的变量设置初始值。

例如可以为**非数组**的 POINTC 类型变量设置初始值。

示例:

```
POINTC p0 = POINTC(100, 200, 300, 0, 90, 0)
DINT a = 5
```

对于基本数据类型，也可以为数组变量设置初始值。参见下面的例子。

示例:

```
DINT varArray[0..2] = [1, 2, 3]
```

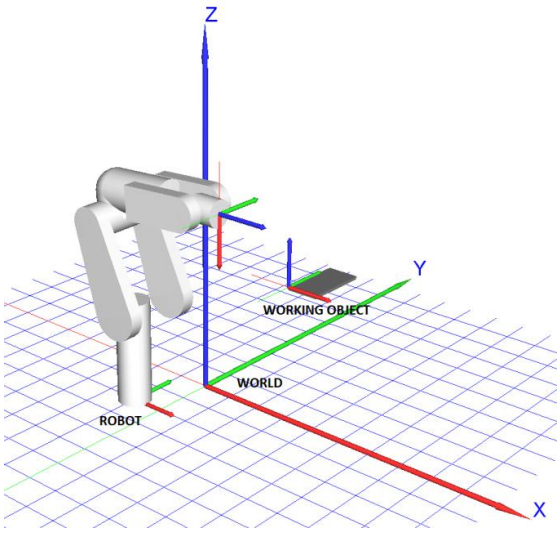
如果初始值小于数组元素，则其余值设置为默认值。如果初始值大于数组元素，则不使用超出的值。

示例:

```
DINT varArrayA[0..2] = [1, 2]
DINT varArrayB[0..2] = [4, 5, 6, 7]
```

varArrayA 中元素的值将设为 1,2,0 而 varArrayB 中元素的值将设为 4,5,6。

3.3.4 参考坐标系



参考坐标系用于简化程序改变。如果机器人的位置和/或工件的位置发生改变，则可以通过仅改变参考坐标系的值来重复使用原程序。

3.3.4.1 世界坐标系

世界参考坐标系是一个固定坐标系，被系统中使用的所有其他参考系来引用。

3.3.4.2 机器人坐标系

机器人坐标系识别机器人在系统中的位置。通常，机器人坐标系设置与世界坐标系相同。

3.3.4.3 工作对象坐标系

工作对象坐标系确定工件的位置。

3.3.5 机器人轴配置

cfg 枚举类型中，从 CFG1 到 CFG8 的用于从八种可能的机器人配置中选择一种。中立配置定义为 CFG0。

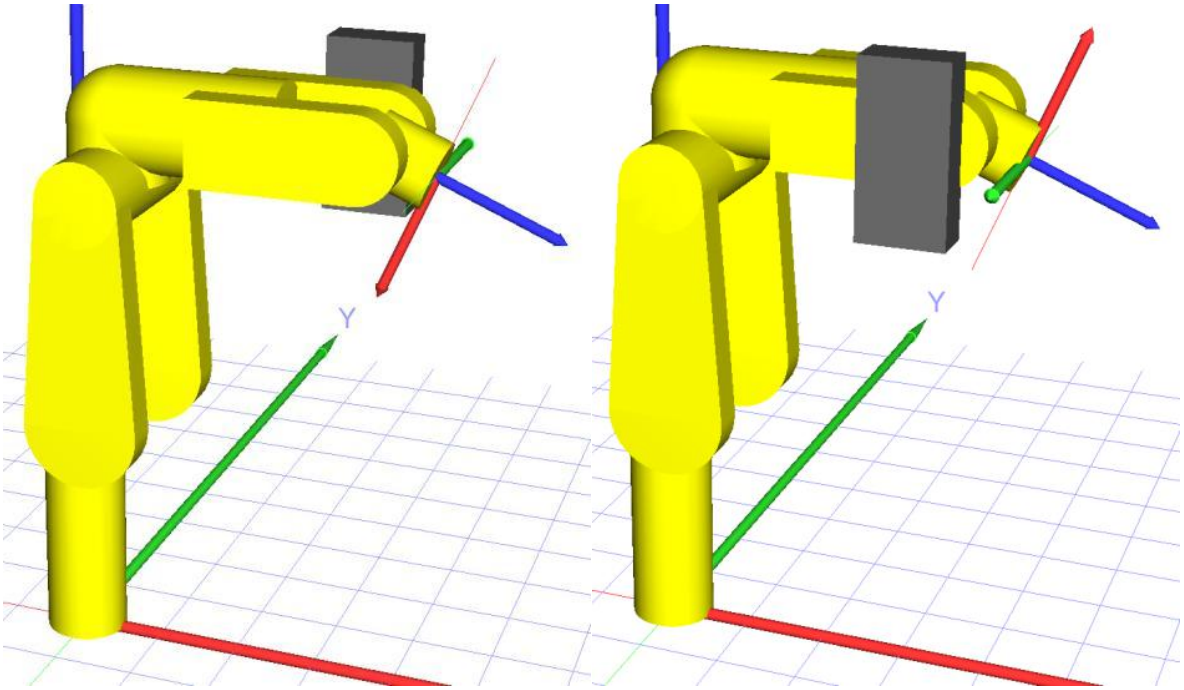
下表描述了机器人相对于三个奇点的位置。

CFG	腕关节中心相对于 1 轴位置	手腕中心相对于小臂位置	5 轴的角度
CFG1	前面	前面	负(<0)
CFG2	前面	前面	正(>0)
CFG3	前面	后面	负(<0)
CFG4	前面	后面	正(>0)

CFG5	后面	前面	负(<0)
CFG6	后面	前面	正(>0)
CFG7	后面	后面	负(<0)
CFG8	后面	后面	正(>0)

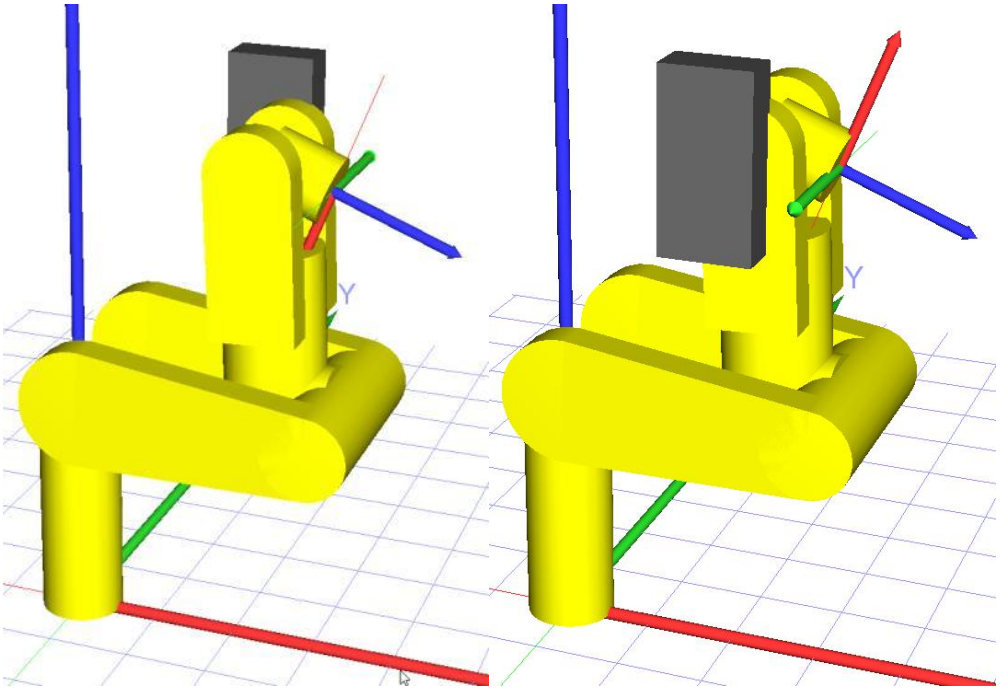
下面的图片给出了一个例子，说明如何通过使用 8 种不同的配置来实现相同的工具位置和方向。

下图为机器人轴参数(cfg)设置为 CFG1 和 CFG2 的示例。注意轴 5 角的不同符号。



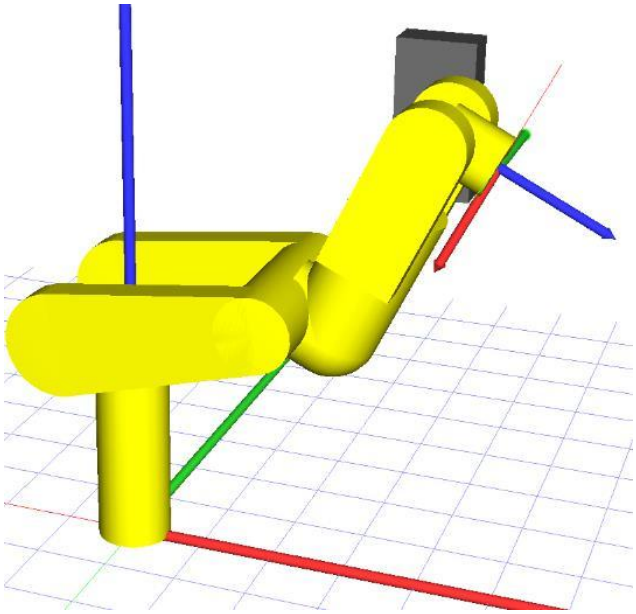
CFG1　　CFG2

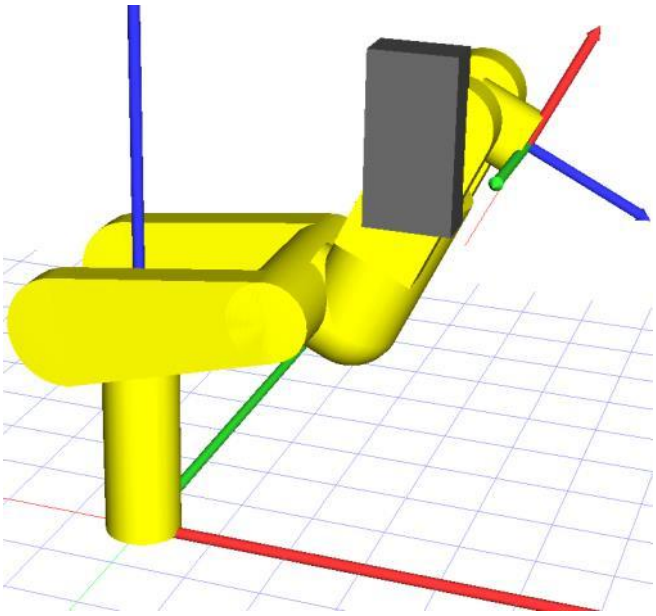
下图为机器人轴参数(cfg)设置为 CFG3 和 CFG4 的示例。注意轴 5 角的不同符号。



CFG3　　CFG4

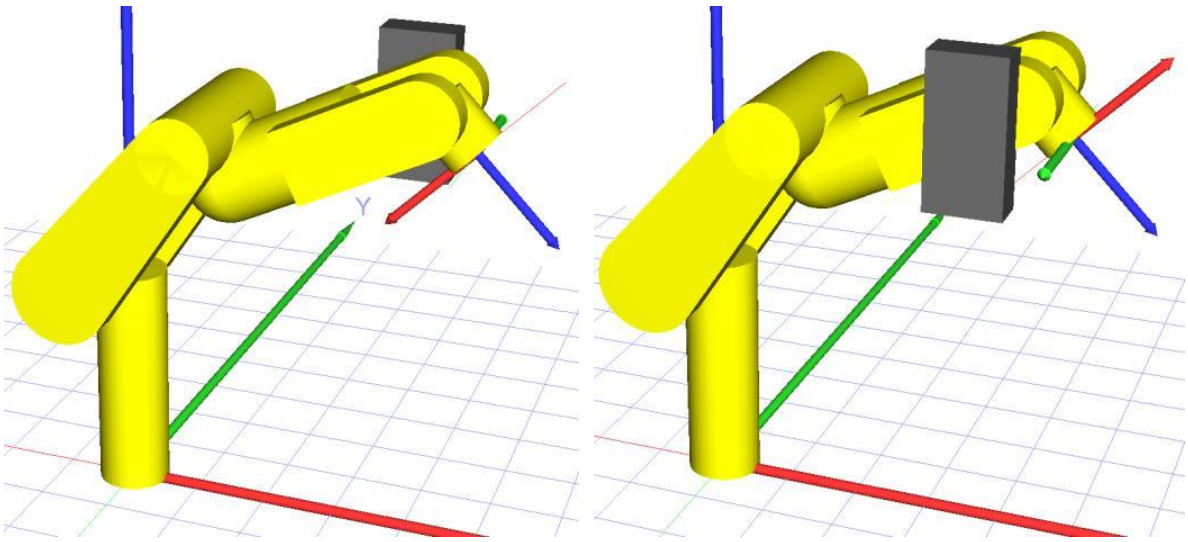
下图为机器人轴参数(cfg)设置为 CFG5 和 CFG6 的示例。注意轴 5 角的不同符号。





CFG5 CFG6

下图为机器人轴参数(cfg)设置为 CFG7 和 CFG8 的示例。注意轴 5 角的不同符号。



CFG7 CFG8

3.3.6 预定义变量

通常你可以在系统中找到一些预定义的模块。这些模块有些会提供一些预定义的变量，以使您能更好的使用系统功能。

3.3.6.1 system 模块（基础共享模块）

这个模块中定义了一下变量:

- REFSYS 类型的 wobj0。这是世界参考系统。
- TOOL 类型的 tool0。如果机器人上没有安装工具，可以机器人的工具使用这个。
- 不同的预定义 SPEED 类型的速度变量，例如 v250, v500。
- ZONE 型的变量 fine。如果一个移动必须在一个精确的点终止，这个设置可以使用，避免移动路径之间的混合。
- 不同的预定义 ZONE 类型的圆滑过渡变量，例如 z50, z100。

3.3.6.2 tool 模块

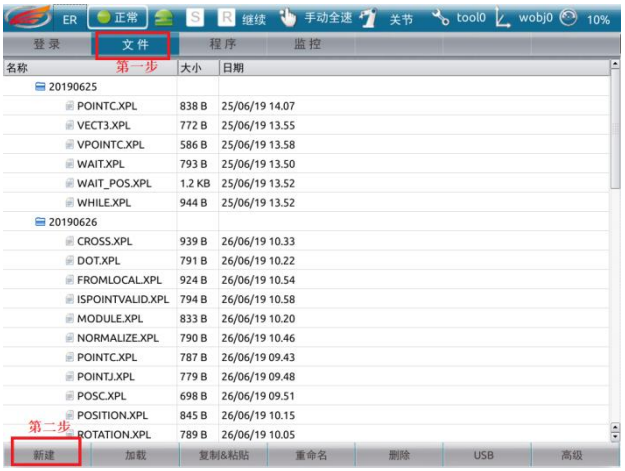
该模块用于存储 TOOL 设置。

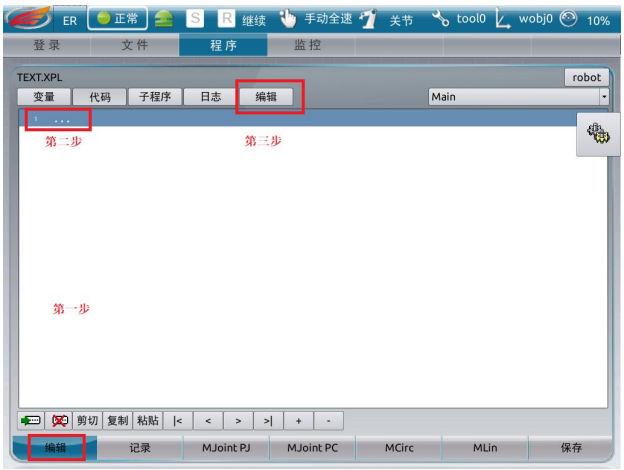
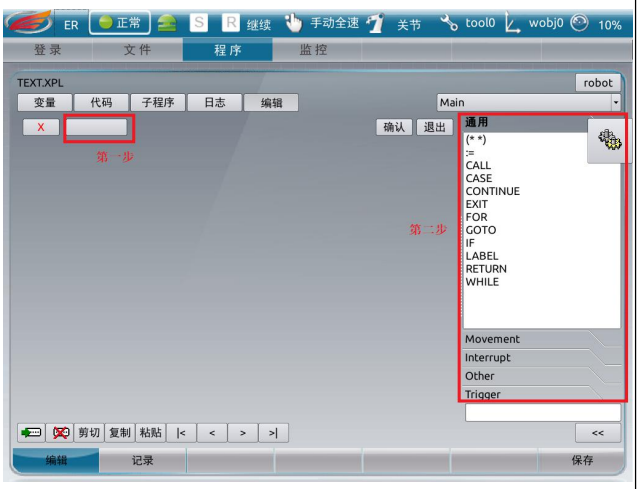
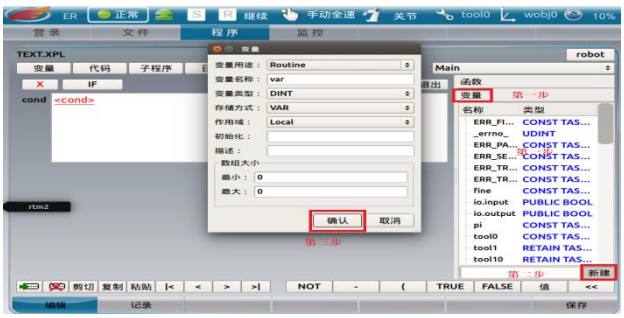
3.3.6.3 userframe 模块

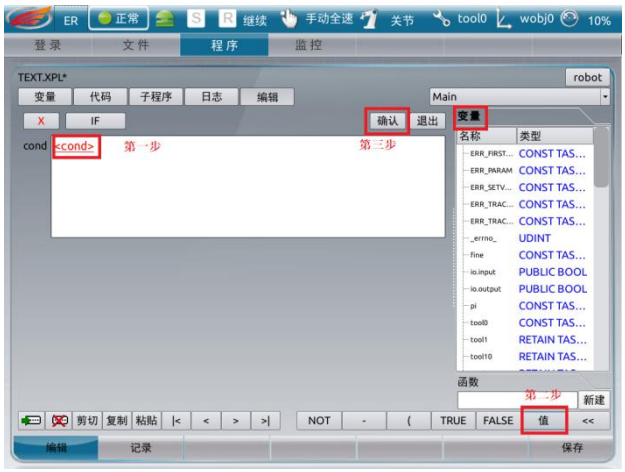
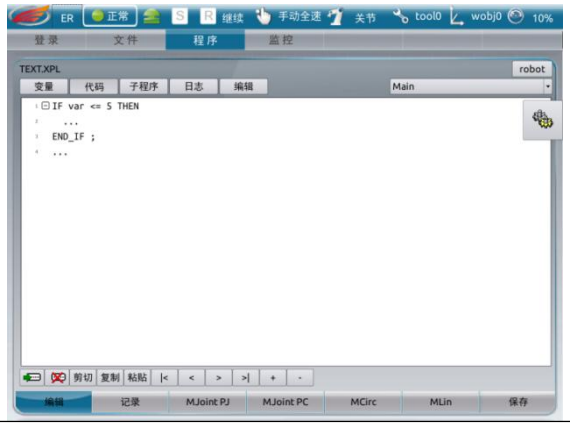
该模块用于存储 REFSYS 设置。

3.4 RPL 指令

3.4.1 添加指令操作

步骤	图片	描述
1.登录。		a. 击标题栏“登录”后输入密码(999999)。 b. 击“登录”。
2.创建文件夹或文件。		c. 界面中选择文件选项。 d. 击新建，新建文件夹或文件。 e. 定文件夹或文件名称。 f. 需加载文件，选中文件后，点击下方加载即可。

3．编写程序准备。		g. 击“编辑”进入编辑功能。 h. 中需要编辑的行。 i. 击上方编辑。
4.选择函数或方法以及创建变量。	 	j. 择白框后即可在右侧选择函数或方法。 k. 中变量点击“新建”即可创建各个类型的变量，最后点击“确认”即可。

5.程序确认。	 	l. 中需要填写的参数（这里的函数选择为 IF）。 m. 变量范围内选择刚刚创建的变量。 n. 变量进行相应的设置和应用。 o. 击“确认”，创建出完整代码。（下图）。

3.4.2 指令详解

3.4.2.1 (* *) 注释指令

表示在示教器界面的注释。使用此指令，可以在代码中输入一个注释。

示例:

```
Started := true;   //* This is a comment *)
...
WAIT(door_opened);   //（* wait door's input *)
```

3.4.2.2 := 赋值指令

通过此指令，可以为变量赋值。

格式：变量 := 表达式;

示例:

```
i := 123 ;
temp := SIN(0.392) ;
```

在这个例子中，变量 i 被设置为 123，temp 的值为 sin(0.392)。

注意：此指令不能用于将一个数组直接赋值给另外一个数组。如果想要把一个数组的值复制到另外一个数组，必须针对数组中的每个元素使用此指令。

3.4.2.3 ALIAS 变量别名

将一个变量绑定另一个变量。

格式：ALIAS(***变量，参考变量***);

输入：变量、参考变量；

输出：参考变量随变量改变而改变；

示例：

示例变量：

```
BOOL grimper;
BOOL out_1;
```

示例程序：

```
ALIAS(grimper, out_1);        //将 out_1 绑定到 grimper
grimper := true;              //将 grimper 赋为 true
IF out_1 THEN;                //判断：当 out_1 为 true 进入判断
MESSAGE("out_1 = grimper = true"); //日志显示消息文本
END_IF;                       //结束判断
```

在此条指令后，可以将 grimper 变量当作 out_1 使用。该例中 out_1 和 grimper 都会是 true,所以程序中会进入判断并在系统日志文件中显示“out_1 = grimper = true”。

3.4.2.4 CALL 调用

用于执行用户定义的子程序

格式：CALL [***指定的变量列表***]:= 子程序名称（***输入参数***）；

示例：

```
CALL ExecuteSquare(A, B, C, D);
```

调用 ExecuteSquare 子程序,并赋值给四个输入变量。将程序中常出现的一些指令集中到一起并以子程序的形式实现，并让主程序调用，这样可以简化主程序。

子程序：

```
ExecuteSquare (POINTC A, POINTC B, POINTC C, POINTC D) ;
```

变量：

```
Input
POINTC A
POINTC B
POINTC C
```

POINTC D

示例程序:

```
MJOINT (A, v100, fine, tool1);
MLIN (B, v100, fine, tool1);
MLIN (C, v100, fine, tool1);
MLIN (D, v100, fine, tool1);
MLIN (A, v100, fine, tool1);
CALL val，inRange := acquireValue(val, min, max) ;
```

该例调用子程序 acquireValue 传递三个变量，在执行调用子程序后设置两个变量值。

子程序:

```
acquireValue(LEARL val,LEARL min,LEARL max)   => (LEARL outval, BOOL
inRange)
```

变量:

Input:

```
LREAL val
LEARL min
LEARL max
Output
LREAL outval
BOOL inRange
```

示例程序:

```
Outval := LIMIT(val, min, max);
IF outval<>val THEN
inRange := false;
    ELSE
    InRange := true;
END_IF
```

注意：不能够使用数组进行参数传递和赋值传递。

3.4.2.5 CASE 分支判断

根据表达式的值，运行多个语句序列中的一个。

格式:

依赖于表达式的值而后执行 **CASE** 语句中一个 case 程序块。

```
CASE 变量表达式 OF
        表达式_1
            … 语句序列…
...
        表达式 n ..
```

```
        … 语句序列 …
    ELSE
        … 语句序列 …
END_CASE ;
```

与 CASE 表达式匹配的 CASE 指令将会被执行; ELSE 指令必须在放置在 CASE 块中的最后, 如果 CASE 中没有匹配的值，ELSE 段将会被执行。

示例程序:

```
CASE i OF
    1:
        MLIN (P1, v100, fine, tool1) ;
    2 .. NumPoints – 1:
        CALL MoveToPoint (i) ;
    99, var, var + 1:
        MJOINT (HomePos, v100, fine, tool1) ;
        i := 0 ;
    ELSE
        i := 99 ;
END_CASE ;
```

3.4.2.6 CLEARMOVE 终止待处理运动

使用此指令将中止所有待处理的移动。 在调用此指令之前，机器人必须处于保持或静止 状态。如果在执行 CLEARMOVE 时机器人处于运动状态，则会发生错误。该指令通常在中断程序中使用。在运动指令后直接使用该指令是无效的。

格式：CLEARMOVE;

输入:

输出：将机器人现有运动规划清除。

示例:

示例变量名:

示例程序:

Main:

```
MJOINT (POINTC(400, 0, 0, 0, 180, 0), v100, fine, tool1) ;
INTRSET (intr_test,sub1());
INTRCOND (intr_test，io.DIn[10],true);
MJOINT (POINTC(400, 0, 100, 0, 180, 0), v100, fine, tool1) ;
MJOINT (POINTC(400, 0, 200, 0, 180, 0), v100, fine, tool1) ;
```

Sub1:

```
STOPMOVE (1);  //在 1 秒内使机器人暂停
CLEARMOVE;  //清除之前机器人正在进行的运动规划
```

STARTMOVE(); //映射上方 STOPMOVE(1)，重新开始程序

MJOINT (POINTC(400, 100, 0, 0, 180, 0), v100, fine, tool1) ;

MJOINT (POINTC(400, 200, 0, 0, 180, 0), v100, fine, tool1) ;

在本例中，利用中断指令使得机器人在运行 MJOINT (POINTC(400, 0, 200, 0, 180, 0), v100, fine, tool1) 时，进入子程序 sub1 中，在 1s 内暂停当前正在运行的程序并放弃后续规划，并以当前暂停点为起始点开始运行。

3.4.2.7 CLOCKRESET 重置时间测量

当在序中用同一个 CLOCK 数据类型来进行多次计时，为使相互之间不发生干涉，就需要对该 CLOCK 类型进行重置。

格式：CLOCKRESET(***变量名***);

输入：**CLOCK** 类型数据；

输出：**CLOCK** 类型数据 ＝0；

示例：

示例变量：**CLOCK** clk1；

示例程序：

CLOCKSTART (clk1);　　//开始计时

MJOINT (*, v500, fine, tool0);　　//机器人关节运动

MJOINT (*, v500, fine, tool0);　　//机器人关节运动

CLOCKSTOP (clk1);　　//计时停止

MESSAGE ("Time %1", CLOCKREAD(clk1)); //显示测量的机器人运动时间

CLOCKRESET (clk1);　　//时间重置

CLOCKSTART (clk1);　　//开始计时

MLIN (*, v500, fine, tool0);　　//机器人直线运动

CLOCKSTOP (clk1);　　//计时结束

MESSAGE ("Time2 %1", CLOCKREAD(clk1));//显示测量的机器人运动时间

上例中展示了如何只使用一个时间测量变量 clk1 进行两次测量。

3.4.2.8 CLOCKSTART 开启时间测量

当我想要开始计算某一过程经历的时间时，就需要通过 CLOCKSTART 函数来开启这一时间计时测量过程。

格式：CLOCKSTART(***变量名***)；

输入：CLOCK 类型数据；

输出：CLOCK 类型数据数值开始增加；

示例：

示例变量：

```
CLOCK clk1；
LREAL partial；
LREAL total；
```

示例程序：

```
CLOCKRESET (clk1)；          //重置时间，此时时间计时为 0
CLOCKSTART (clk1)；          //开始计时，时间开始增加
MJOINT (*, v500, fine, tool0);       //机器人关节运动
MJOINT (*, v500, fine, tool0);       //机器人关节运动
CLOCKSTOP (clk1) ;           //计时停止
partial := CLOCKREAD (clk1);     //partial 即两次关节运动过程计量的时间
CLOCKSTART (clk1);           //开始计时继续
MLIN (*, v500, fine, tool0);       //机器人直线运动
CLOCKSTOP (clk1);          //计时结束
total := CLOCKREAD (clk1);     //total 即该所有运动过程计量的总时间
```

此例子展示了如何测量部分时间和总时间。

3.4.2.9 CLOCKSTOP 停止时间测量

当我开始对某一运动过程进行计时的时候，在该运动结束后，需要停止计时就要用到 CLOCKSTOP 函数。

格式：CLOCKSTOP(*变量名*)；

输入：CLOCK 类型数据；

输出：CLOCK 类型数据数值停止增加；

示例：

示例变量：

```
CLOCK clk1；
LREAL time；
```

示例程序：

```
CLOCKRESET (clk1)；          //重置时间，此时时间计时为 0
CLOCKSTART (clk1)；          //开始计时，时间开始增加
MJOINT (*, v500, fine, tool0);       //机器人关节运动
MJOINT (*, v500, fine, tool0);       //机器人关节运动
CLOCKSTOP (clk1) ;           //计时停止
time := CLOCKREAD (clk1);       //time 即该运动过程计量的时间
```

此例中，展示了计算运动执行时间。

3.4.2.10 CMARC 圆周队列运动

该指令仅适用于 XPL 3.0 及以上版本。

设置机器人的末端（TPC）点按照终点是目标点 target，同时经过中间点 intermediate 的圆弧轨迹进行运动。

CMARC 指令必须成对使用！当程序执行过程中，检测到未成对匹配的 CMARC 指令时，会有错误产生。

在连续模式下，机器人依照第一条 CMARC 指令保存中间点 intermediate 点位信息，第二条 CMARC 指令保存目标点 target 的点位信息的规则进行运动。机器人经过中间点 intermediate 时的姿态由 WRISTMODE 指令设置决定。

在单步模式下，CMARC 指令执行一条终点为当前指令设置的终点的圆弧轨迹。

中间点 intermediate 和目标点 target 距离过近，会导致程序报错!

只有当圆滑过渡参数不同于 fine 时，开启运动的指令才会跟随运动队列结束，否则它将在运动完成时结束。

注意：如果终点与起始点一致，旋转感觉是不可预测的!

警告：因为机器人的实际位置是用来描述圆周运动的，所以当机器人正处于程序设定的位置时，直接从这个指令开始执行程序是很危险的。如果机器人 TCP 位置处于中间点 intermediate 和目标点 target 之间时，执行第一条 CMARC 指令，则执行的圆周运动将与程序设定的方向相反。

如果机器人姿态需要改变，旋转轴(a, b 和 c)将跟随主轴(x, y 和 z)同步进行插值(将开始运动时同步开始，且到达目标位置时同步结束)。如果省略 refsys 参数，则使用世界坐标系统进行运动。如果目标包含辅助轴位置，则辅助轴与机器人轴线同步移动。

格式：CMARC(***目标点，速度，区域，工具,[参考坐标系]***);

警告：只有在上一段移动结束且第二段移动开始之前时，才可以在两个连续的移动之间更改工具。否则，将报告一个错误。

警告：如果圆滑过渡参数 zone 不为 fine，运动会在 CMLIN 队列结束后停止。

示例程序:

```
…
CMLIN (a, v100, fine, tool1) ;
CMARC (b, v100, fine, tool1) ;
CMARC (c, v100, fine, tool1) ;
…
```

在这个例子中，直线移动到点 a 后，从点 a 开始圆弧运动，经过点 b，到达最后点 c。

示例程序:

```
…
CMARC (a, v100, fine, tool1) ;
CMLIN (b, v100, fine, tool1) ;
```

在本例中，当执行 CMARC 指令时报告一个错误，因为有一个未配对的 CMARC 指令。

3.4.2.11 CMLIN 笛卡尔直线队列运动

开始执行 TCP 从最后一个位置到目标位置的线性（笛卡尔）移动。只有当圆弧过渡参数 zone 不为 fine 时，CMLIN 指令会被压入到前瞻队列中进行前瞻处理，进入队列运动，同时指令指针会被顺序向下执行。若 zone 为 fine 时指令指针和运动指针将同时进行。

如果需要改变方向，则将方向轴（a、b 和 c）插入到主轴（x、y 和 z）（将同时开始移动并到达目标位置）。如果参考坐标系参数省略，则默认使用世界坐标系进行移动。

如果目标具有辅助轴的位置，则辅助轴会与机器人的轴同步移动。

警告：只有在上一段移动结束且第二段移动开始之前时，才可以在两个连续的移动之间更改工具。否则，将报告一个错误。

警告：如果圆弧过渡参数 zone 不为 fine，运动会在 CMLIN 队列结束后停止。

格式：CMLIN(***目标点，速度，区域，工具,[参考坐标系]***);

输入：目标点，速度，圆弧过渡，工具，[参考系]

输出：机器人将沿 CMLIN 队列，进行队列运动。

示例:

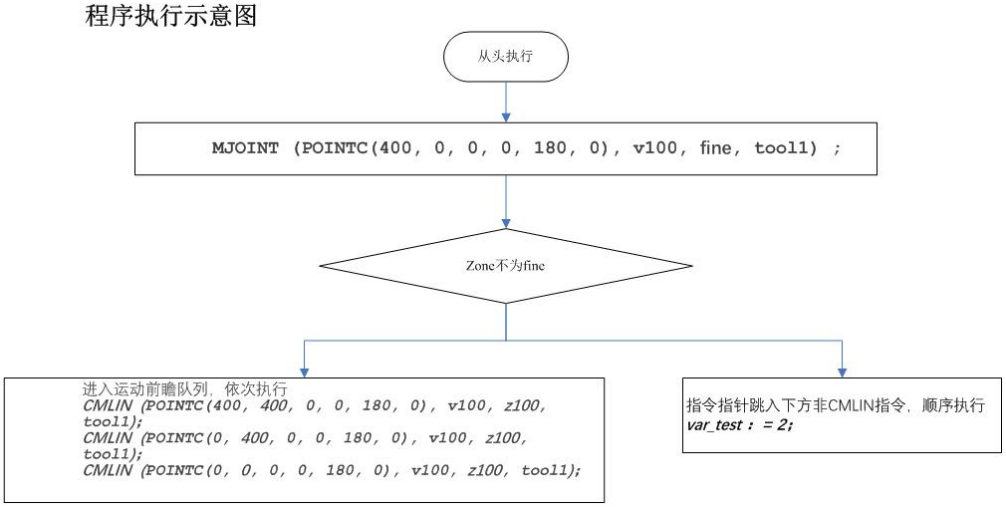
示例变量名: **int var_test = 0;**

示例程序:

```
MJOINT (POINTC(400, 0, 0, 0, 180, 0), v100, fine, tool1) ;
CMLIN (POINTC(400, 400, 0, 0, 180, 0), v100, z100, tool1);//当运行至该行时，下方 CMLIN 进入前瞻队列，同时指令指针指向 var_test，执行 var_test 赋值指令
CMLIN (POINTC(0, 400, 0, 0, 180, 0), v100, z100, tool1);
CMLIN (POINTC(0, 0, 0, 0, 180, 0), v100, z100, tool1);
var_test ： = 2;
```

注意：当运行至 CMLIN 后，若 CMLIN 中圆弧过渡 zone 不为 fine，则机器人将会使 CMLIN 压入前瞻队列执行 CMLIN 运动，同时指令指针将顺序执行下方非运动指令直至停留在最近运动指令处。

程序执行示意图



3.4.2.12 CMCIRC 笛卡尔圆弧队列运动

开始执行 TCP 从最后一个位置，经过中间点到目标位置的圆弧（笛卡尔）运动。只有当圆弧过渡参数 zone 不为 fine 时，CMCIRC 指令会被压入到前瞻队列中进行前瞻处理，进入队列运动，同时指令指针会被顺序向下执行。若 zone 为 fine 时指令指针和运动指针将同时进行。

注意：若终点与起始点为同一点，旋转方向是不可预测的!

警告：如果起始点位置远离程序编程位置，那么机器人开始执行运动指令具有风险性。 如果机器人 TCP 位置在中间点和目标点之间，并且启动 CMCIRC 指令，则机器人执行与编程方向相反的圆周运动。

警告：只有当前一个运动在第二个运动开始之前结束时，才有可能在两个连续运动之间改变工具。否则报告错误。有关更多细节，请参见 CMLIN 指令中的示例。

警告：如果圆弧过渡 zone 参数不为 fine，则当进入运动队列时，结束当前指令指针，并跳入 CMCIRC 指令外继续执行。此时，机器人将继续运行 CMCIRC 队列，直至队列结束。

如果需要改变方向，则插补方向轴（a、b 和 c）到主轴（x、y 和 z）（将同时开始移动并到达目标位置）。如果参考坐标系参数省略，则默认使用世界坐标系进行移动。

如果目标具有辅助轴的位置，则辅助轴会与机器人的轴同步移动。

警告：只有在上一段移动结束且第二段移动开始之前时，才可以在两个连续的移动之间更改工具。否则，将报告一个错误。

警告：如果圆弧过渡参数 zone 不为 fine，运动会在 CMCIRC 队列结束后停止。

格式：CMCIRC(***中间点，目标点，速度，区域，工具，[参考坐标系]***);

输入：圆弧中间点，目标点，速度，圆弧过渡，工具，[参考系]

输出：机器人将沿 CMCIRC 队列，进行队列运动。

示例:

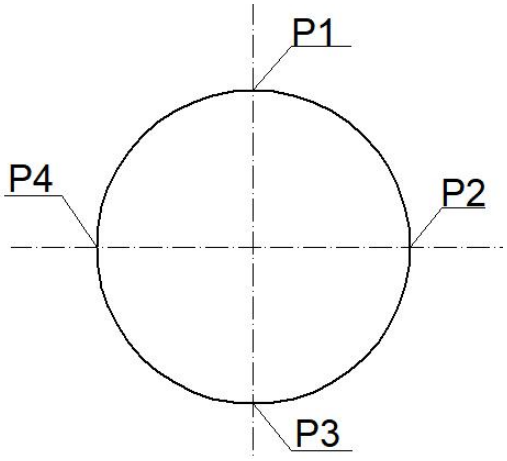
示例变量名:

```
P1 := POINTC(0, 400, 0, 0, 180, 0)
P2 := POINTC(400, 0, 0, 0, 180, 0)
P3 := POINTC(0, -400, 0, 0, 180, 0)
P4 := POINTC(-400, 0, 0, 0, 180, 0)
var_test := 0
```

示例程序:

```
MIIN (P1, v100, fine, tool1) ;
CMCIRC (P2, P3, v100, z100, tool1);
CMCIRC (P4, P1, v100, z100, tool1);
var_test ： = 2;
```

注意：当运行至 CMCIRC 后，若 CMCIRC 中圆弧过渡 zone 不为 fine，则机器人将会使 CMLIN 压入前瞻队列执行 CMCIRC 运动，同时指令指针将顺序执行下方非运动指令直至停留在最近运动指令处。



3.4.2.13 CONTINUE 继续

中断目前的 WHILE 或者 FOR 循环，并强制重新评估目前的循环条件。

示例:

```
WHILE (i <= 10) DO
  i := i + 1 ;
  IF (MOD(i, 2) = 1) THEN
        CONTINUE ;
    END_IF;
  MLIN(POINTC(200, -500, 200, 0, 180,0), v100, fine, tool1) ;
  MLIN(POINTC(200, -900, 200, 0, 180,0), v100, fine, tool1) ;
END_WHILE;
```

此例中如果变量 i 是奇数，后面的两条 MLIN 指令将会被跳过，循环条件重新评估。

3.4.2.14 DWELL 时间等待

在执行后面的指令前等待指定的参数时间（单位为秒）。

格式: DWELL(*值*)

示例程序:

```
MJOINT (POINTC(500, 500, 0, 0, 0, 0), v100, fine, tool1) ;
DWELL (1.5);
MJOINT (POINTC(800, 800, 200, 0, 0, 0), v100, fine, tool1) ;
```

执行第一条运动指令后，机器人将会在执行第二条指令前等待 1.5 秒。

3.4.2.15 ENDPROG 终止程序

终止当前程序的执行

格式: *ENDPROG;*

示例程序:

```
MJOINT (*, v500, fine, tool0);        //运行轨迹
ENDPROGR;          //程序停止
DWELL(1.000);          //等待 1 秒
```

MJOINT (*, v500, fine, tool0);　　//运行轨迹

程序中 ENDPROGR 指令将当前运行的程序停止，程序指针消失。假如需要再次运行程序，重新激活指针。

3.4.2.16　EPATH

启动机器人执行指定的路径。

格式：***EPATH (PATH_L 路径，速度，圆滑过渡，工具, [参考坐标系])；***

输入：PATH_L 路径, SPEED 速度, ZONE 圆滑过渡,TOOL 工具, [REFSYS 参考坐标系]；

在输入此指令之前，必须添加一个外部变量来访问要执行的路径。如果省略 refsys 参数，则使用世界坐标系进行运动。在执行所选路径之前，机器人将通过关节运动移动到轨迹的第一个点。

示例程序:

示例变量:

External

trajectory：PATH_L；

示例程序:

EPATH (trajectory, v100, fine, tool1) ;

此例中名为 trajectory 的路径会被执行（必须确认机器人加载的库中有名为 trajectory 的 PATH_L 类型变量）。

3.4.2.17　ERROR 错误代码指令

设置 expr 值为系统变量_error_的错误代码。然后错误消息将会在系统中用户日志文件输出显示，程序的执行将会被中断，错误处理代码将会执行。默认的错误处理是停止目前程序的执行。

格式：ERROR(***expr,[expr1],[expr2]***);

输入：错误判断字符串；

输出：程序中断并输出字符串；

示例程序:

示例变量:

DINT value；

示例程序:

```
MJOINT (*, v500, fine, tool0);     //机器人关节运动
IF value < 10 THEN；          //判断：当“value”的值小于 10 进入判断
ERROR(1,“value is less than”,10);//程序中断，错误代码执行
END_IF;                    //结束判断
```

MLINE (*, v500, fine, tool0); //机器人直线运动

此例中，定义的 value 值为 0，程序执行的程序将会被停止,最后的直线运动不会执行，并且在系统的用户日志文件中将会显示“ERROR CODE 1: value is less than 10”。

3.4.2.18 EXEC 执行加载子程序

执行加载指令 LOAD 加载的模块程序。

格式：*EXEC (返回值，程序名，输入参数)*;

输入：返回值，程序名（模块程序名.Main 或者 模块程序名.模块子程序名），

输出：

示例程序:

例 1:

LOAD (“WH.XPL”) ; //将 WH.XPL 程序加载到当前程序内存中

EXEC “WH.Main” ; //执行 WH 的主程序

将 WH.XPL 程序作为模块加载到当前程序中，EXCE 指令运行 WH 的 Main 程序

例 2:

LOAD (“WH.XPL”) ; //将 WH.XPL 程序加载到当前程序内存中

EXEC “WH.jj” ; //执行 WH.XPL 的子程序 jj

将 WH.XPL 程序作为模块加载到当前程序中，EXCE 运行 WH.XPL 的子程序 jj

3.4.2.19 EXIT 结束循环

终止 WHILE 或者 FOR 循环，也可以用于主程序的停止执行。

示例:

```
WHILE i <= 10 DO
  i := i + 1 ;
IF (i = 4) THEN
        EXIT;
END_IF;
  MLIN (POINTC(200, -500, 200, 0, 180, 0), v100, fine, tool1) ;
  MLIN (POINTC(200, -900, 200, 0, 180, 0), v100, fine, tool1) ;
END_WHILE ;
```

在此例中如果 i 值等于 4 时，正在执行的 while 循环将会被中断。

3.4.2.20 FMJOINT 笛卡尔圆弧队列运动

开始执行从最后一个位置到目标位置的关节（点对点）运动。 目标位置可以是 POINTC，POINTJ，EPOINTC，EPOINTJ 类型。所有轴将开始移动并同时到达其目标位置。TCP 运动将是单轴运动的组合。如果省略 refsys 参数，则默认使用世界坐标系进行移动。仅当 zone 参数与精细区域不同时，指令才在移动队列中结束，否则在移动完成时结束。

如果目标具有辅助轴的位置，则辅助轴会与机器人的轴同步移动。

警告：如果起始点位置远离程序编程位置，那么机器人开始执行运动指令具有风险性。 如果机器人 TCP 位置在中间点和目标点之间，并且启动 CMCIRC 指令，则机器人执行与编程方向相反的圆周运动。

警告: 如果圆弧过渡 zone 参数不为 fine，则当进入运动队列时，结束当前指令指针，并跳入 FMJOINT 指令外继续执行。此时，机器人将继续运行 FMJOINT 队列，直至队列结束。
警告：只有在上一段移动结束且第二段移动开始之前时，才可以在两个连续的移动之间更改工具。否则，将报告一个错误。

格式：FMJOINT(***目标点，速度，区域，工具，[参考坐标系]***);

输入：目标点，速度，圆弧过渡，工具，[参考系]

输出：机器人将沿 FMJOINT 队列，进行队列运动。

示例程序:

示例变量名:

```
Int var_test := 0
```

示例程序:

```
MJOINT (POINTC(400, 0, 0, 0, 180, 0), v100, fine, tool1) ;
FMJOINT (POINTC(400, 400, 0, 0, 180, 0), v100, z100, tool1)；//当运行至该行时，下方
FMJOINT 进入前瞻队列，同时指令指针指向 var_test，执行 var_test 赋值指令
FMJOINT (POINTC(0, 400, 0, 0, 180, 0), v100, z100, tool1)；
FMJOINT (POINTC(0, 0, 0, 0, 180, 0), v100, z100, tool1)；
var_test := 2；
```

注意：当运行至 FMJOINT 后，若 FMJOINT 中圆弧过渡 zone 不为 fine，则机器人将会使 FMJOINT 压入前瞻队列执行 FMJOINT 运动，同时指令指针将顺序执行下方非运动指令直至停留在最近运动指令处。

3.4.2.21 FMLIN 笛卡尔直线队列运动

开始执行 TCP 从最后一个位置到目标位置的线性（笛卡尔）移动。只有当圆弧过渡参数 zone 不为 fine 时，FMLIN 指令会被压入到前瞻队列中进行前瞻处理，进入队列运动，同时指令指针会被顺序向下执行。若 zone 为 fine 时指令指针和运动指针将同时进行。
如果需要改变方向，则插补方向轴（a、b 和 c）到主轴（x、y 和 z）（将同时开始移动并到达目标位置）。如果参考坐标系参数省略，则默认使用世界坐标系进行移动。
如果目标具有辅助轴的位置，则辅助轴会与机器人的轴同步移动。
警告：只有在上一段移动结束且第二段移动开始之前时，才可以在两个连续的移动之间更改工具。否则，将报告一个错误。
警告：如果圆弧过渡参数 zone 不为 fine，运动会在 FMLIN 队列结束后停止。

格式：FMLIN(***目标点，速度，区域，工具, [参考坐标系]***);

输入：目标点，速度，圆弧过渡，工具，[参考系]

输出：机器人将沿 FMLIN 队列，进行队列运动。

示例程序:

示例变量名:

int var_test = 0；

示例程序:

MJOINT (POINTC(400, 0, 0, 0, 180, 0), v100, fine, tool1) ;

FMLIN (POINTC(400, 400, 0, 0, 180, 0), v100, z100, tool1)；//当运行至该行时，下方 CMLIN 进入前瞻队列，同时指令指针指向 var_test，执行 var_test 赋值指令

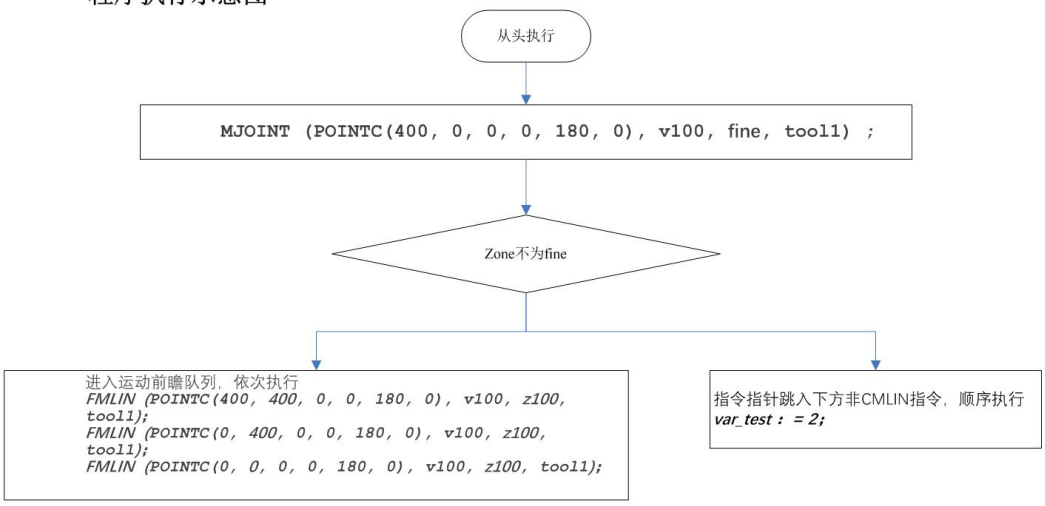
FMLIN (POINTC(0, 400, 0, 0, 180, 0), v100, z100, tool1)；

FMLIN (POINTC(0, 0, 0, 0, 180, 0), v100, z100, tool1)；

var_test ： = 2；

注意：当运行至 FMLIN 后，若 FMLIN 中圆弧过渡 zone 不为 fine，则机器人将会使 FMLIN 压入前瞻队列执行 FMLIN 运动，同时指令指针将顺序执行下方非运动指令直至停留在最近运动指令处。

程序执行示意图



3.4.2.22 FMCIRC 笛卡尔圆弧队列运动

开始执行 TCP 从最后一个位置，经过中间点到目标位置的圆弧（笛卡尔）运动。只有当圆弧过渡参数 zone 不为 fine 时，FMCIRC 指令会被压入到前瞻队列中进行前瞻处理，进入队列运动，同时指令指针会被顺序向下执行。若 zone 为 fine 时指令指针和运动指针将同时进行。

注意：若终点与起始点为同一点，旋转方向是不可预测的!

警告：如果起始点位置远离程序编程位置，那么机器人开始执行运动指令具有风险性。 如果机器人 TCP 位置在中间点和目标点之间，并且启动 FMCIRC 指令，则机器人执行与编程方向相反的圆周运动。

警告：只有当前一个运动在第二个运动开始之前结束时，才有可能在两个连续运动之间改变工具。否则报告错误。有关更多细节，请参见 CMLIN 指令中的示例。

警告：如果圆弧过渡 zone 参数不为 fine，则当进入运动队列时，结束当前指令指针，并跳入 FMCIRC 指令外继续执行。此时，机器人将继续运行 FMCIRC 队列，直至队列结束。

如果需要改变方向，则插补方向轴（a、b 和 c）到主轴（x、y 和 z）（将同时开始移动并到达目标位置）。如果参考坐标系参数省略，则默认使用世界坐标系进行移动。

如果目标具有辅助轴的位置，则辅助轴会与机器人的轴同步移动。

警告：只有在上一段移动结束且第二段移动开始之前时，才可以在两个连续的移动之间更改工具。否则，将报告一个错误。

警告：如果圆弧过渡参数 zone 不为 fine，运动会在 FMCIRC 队列结束后停止。

格式：FMCIRC(***中间点，目标点，速度，区域，工具，[参考坐标系]***);

输入：圆弧中间点，目标点，速度，圆弧过渡，工具，[参考系]

输出：机器人将沿 FMCIRC 队列，进行队列运动。

示例程序:

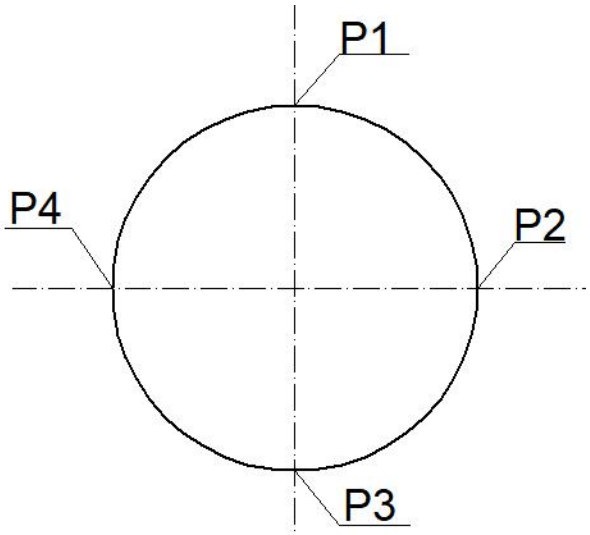
示例变量名:

```
P1 := POINTC(0, 400, 0, 0, 180, 0)
P2 := POINTC(400, 0, 0, 0, 180, 0)
P3 := POINTC(0, -400, 0, 0, 180, 0)
P4 := POINTC(-400, 0, 0, 0, 180, 0)
var_test := 0
```

示例程序:

```
MIIN (P1, v100, fine, tool1) ;
FMCIRC (P2, P3, v100, z100, tool1);
FMCIRC (P4, P1, v100, z100, tool1);
var_test ： = 2;
```

注意：当运行至 FMCIRC 后，若 FMCIRC 中圆弧过渡 zone 不为 fine，则机器人将会使运动信息压入前瞻队列执行 FMCIRC 运动，同时指令指针将顺序执行下方非运动指令直至停留在最近运动指令处。



3.4.2.23 FOR 循环体

循环执行某个代码块几次。首先需要设置使用初始值表达式来初始化变量 variable，在代码块执行每次执行后，变量 variable 将会以增量表达式的值更新；如果更新后的变量 variable 值在初始值和终止值之间指定范围内，代码块将会再次执行，否则执行 FOR 循环体外的后续指令。若增量表达式不进行任何指定，默认变量 variable 的增量为 1。FOR 循环体可以使用 EXIT 和 CONTINUE 指令。

格式：FOR variable := 初始值表达式 TO 终止值表达式 BY [增量表达式] DO

...

END_FOR ;

示例程序:

FOR i := 1 TO 4 BY 1 DO

MLIN(POINTC(200, -500, 200, 0, 180, 0), v100, fine, tool1) ;

MLIN(POINTC(200, -900, 200, 0, 180, 0), v100, fine, tool1) ;

END_FOR ;

使用变量 i 执行四次循环，第一次变量 i 的值为 1，每执行一次循环，i 的值加 1.最后一次执行循环时，i 值为 4。执行此次循环后，i 值为 5。当在程序中键入 FOR 指令时，END_FOR 将会自动添加。

3.4.2.24 GOTO 跳转指令

跳到某一个特定的标签，并执行标签后的程序指令。

格式：**GOTO(标签名称)**

示例程序:

LABEL Start

MJOINT (POINTC(500, 500, 0, 0, 180, 0), v100, fine, tool1) ;

MJOINT (POINTC(800, 800, 200, 0, 180, 0), v100, fine, tool1) ;

GOTO Start;

此例中，在 GOTO (Start)后，程序跳到指令 LABEL (Start)处继续执行。

3.4.2.25 IF THEN ELSE 条件语句

IF 条件为真，则执行 IF 后的指令语句；否则执行 ELSE 后的指令语句。

当键入一个 IF 指令时，同时将会自动添加 END_IF。如果你想要插入 ELSE 语句，必须点击 END_IF 语句，然后在示教器显示的编辑栏添加 ELSE。

格式：IF 条件 THEN

...

ELSE

...

END_IF ;

示例程序:

IF j := 1 THEN

sum := sum + 1 ;

ELSE

sum := sum + 2 ;

END_IF ;

此例中如果变量 j 等于 1，变量 sum 加 1，否则 sum 加 2。

3.4.2.26 INTRCOND 中断触发条件

用于设置关联中断 INTR 变量的触发中断的条件，变量表达式必须是布尔类型。在设置与中断相关的条件后，中断被使能，即当中断触发条件从假转化为真时，中断被触发。

格式：**INTRCOND (INTR 变量，布尔型表达式);**

输入：添加中断变量 INTR

输出：将 INTR 变量与 trapRoutine()子程序相关联，然后设定 INTR 变量触发的条件为 io.input[8]<>false

示例程序:

示例变量名:

INTR trap //声明 trap 变量

示例子程序名:

trapRoutine() //创建 trapRoutine（）子程序

示例程序:

INTRSET(trap, trapRoutine()); //将 trap 变量与 trapRoutine（）子程序关联

INTRCOND(trap, io.input[8]<>false); //设定 trap 变量触发的条件

在此例中当 io.input[8]为 true 时，中断程序 trapRoutine 将会被执行。

注意：在使用该指令前，必须使用 INTRSET 指令设置此指令 INTR 变量，否则报告错误。

3.4.2.27 INTRALLOW 中断允许

启用先前被 INTRDENY 指令禁用的特定中断指令。

格式：**INTRALLOW(INTR 变量);**

输入：添加中断变量 INTR

输出：将 INTR 变量与 trapRoutine()子程序相关联，通过 INTRDENY 禁用，最后通过 INTRALLOW 启用

示例:

示例变量名:

INTR trap //声明 trap 变量

示例子程序名:

trapRoutine() //创建 trapRoutine（）子程序

示例程序:

INTRSET(trap, trapRoutine()); //将 trap 变量与 trapRoutine（）子程序关联

INTCOND(trap, io.input[8]<>false); //设定 trap 变量触发的条件

INTRDENY(trap); //禁用 trap 变量关联的 trapRoutine（）中断指令

(*Interrupt trap must be avoid in following code*)

(*Now is possible to enable interrupt trap*)

INTRALLOW(trap); //启动以上被 INTRDENY 指令禁用的 trapRoutine（）中断指令

此例展示了如何在一个指定的程序段中屏蔽一个中断，并用过 INTRALLOW 指令启动中断。

注意：在使用该指令前，INTR 变量必须要与一个中断关联，否则报告错误。

3.4.2.28 INTRDEL 中断删除

删除指定中断的指令。在此指令之后，中断将不再存在。如果中断只是暂时禁用，则应使用 INTRDENY 或 INTRDIS 指令。

格式：**INTRDEL(INTR 变量);**

输入：待删除中断变量 INTR

示例:

示例变量名:

INTR　trap //声明 trap 变量

示例程序:

…

INTRSET (trap, trapRoutine) ;

INTRCOND (trap, io.input[8]<>false) ;

…

INTRDEL (trap) ;

(* 中断 trap 将不再存在 *)

此例展示了如何删除中断。

3.4.2.29 INTRDENY 中断否决

用于禁用特定中断触发的指令。此指令之后该指定的任何触发条件都将失效。

格式: **INTRDENY(INTR 变量);**

输入: 添加两个中断变量 INTR,

输出: 将 INTR 声明的两个变量分别与 trapRoutine()、trap2Routine()子程序相关联，然后设定中断出发的条件，最后使用 INTRDENY 对其中一个中断指令禁用

示例:

示例变量名:

INTR trap //声明 trap 变量

INTR trap2 //声明 trap2 变量

示例子程序名:

trapRoutine() //创建 trapRoutine（）子程序

Trap2Routine() //创建 trap2Routine（）子程序

示例程序:

INTRSET(trap,trapRoutine()); //将 trap 变量与 trapRoutine（）中断指令关联

INTRSET(trap2,trap2Routine()); //将 trap2 变量与 trap2Routine（）中断指令关联

INTRCOND(trap,io.input[8]<>false); //设定 trap 变量触发的条件

INTRCOND(trap2,io.input[9]<>false); //设定 trap2 变量触发的条件

INTRDENY(trap); //禁用 trap 变量关联的 input[8]<>false 中断指令

(*Interrupt trap will be avoid in following code*)

此例展示了如何在屏蔽一个 trap 中断并且中断 trap2 还是有效的。

注意：在使用该指令前，INTR 变量必须要与一个中断关联，否则报告错误。

3.4.2.30 INTRDIS 中断禁用

禁用所有中断或某个中断的指令。如果省略了指令后的可选参数，所有中断将会被禁用。当中断被禁用了，中断条件仍会被检测；如果满足了中断条件，则一旦中断被重新启用，就会调用该中断。

格式：**INTRDIS([INTR 变量]);**

输入：添加中断变量 INTR

输出：将 INTR 变量与 trapRoutine()子程序相关联，通过 INTRDIS 禁用，再通过 INTRENA 启用

示例：

示例变量名：

INTR trap //声明 trap 变量

示例子程序名：

trapRoutine() //创建 trapRoutine（）子程序

示例程序：

```
INTRSET(trap,trapRoutine()); //将 trap 变量与 trapRoutine（）子程序关联
INTRCOND(trap,io.input[8]<>false); //设定 trap 变量触发的条件
INTRDIS(); //禁用以上所有中断或中断指令
(*Interrupt will be disabled in following code*)
INTRENA(); //启用以上所有中断或中断指令
(*Interrupt will be reenabled in following code*)
```

上例展示了如何禁用中断然后重新启用中断。

注意：如果没用省略指令后的可选参数，则 INTR 变量必须要与一个中断关联，否则报告错误。

3.4.2.31 INTRENA 中断触发

启用所有中断或某个中断的指令。如果省略了指令后的可选参数，所有中断将会被启用。

格式：INTRENA([INTR 变量]);

输入：添加中断变量 INTR

输出：将 INTR 变量与 trapRoutine()子程序相关联，通过 INTRDIS 禁用所有中断或中断指令，再通过 INTRENA 启用所有中断或中断指令

示例：

示例变量名：

INTR trap //声明 trap 变量

示例子程序名：

trapRoutine() //创建 trapRoutine（）子程序

示例程序：

```
INTRSET(trap,trapRoutine()); //将 trap 变量与 trapRoutine（）子程序关联
INTRCOND(trap,io.input[8]<>false); //设定 trap 变量触发的条件
INTRDIS(); //禁用以上所有中断或中断指令
(*Interrupt will be disabled in following code*)
INTRENA(); //启用以上所有中断或中断指令
(*Interrupt will be reenabled in following code*)
```

上例展示了如何禁用中断然后重新启用中断。

注意：如果没用省略指令后的可选参数，则 INTR 变量必须要与一个中断关联，否则报告错误。

3.4.2.32 INTRERRNO 错误编号触发中断

当系统变量_errno_被设置为特定值（如果未省略可选整型变量）或设置为不等于零的值时，该指令用于设置触发与 INTR 变量连接的中断的条件。变量_errno_被赋值时可触发中断。可以通过指令 ERROR 或发生执行错误来设置变量_errno_。

格式：INTRERRNO (INTR 变量, [整型值]) ;

输入：添加中断变量 INTR

输出：将 INTR 变量与 errnoHnd()子程序相关联，设定 2 为触发 trapErr 的系统变量值，再通过 ERROR（2）设置系统变量为 2 并触发 trapErr 关联的子程序

示例:

示例变量名:

INTR trapErr //声明 trapErr 变量

示例子程序名:

errnoHnd() //创建 errnoHnd()子程序

示例程序:

INTRSET (trapErr,errnoHnd()); //将 trapErr 变量与 errnoHnd()中断子程序关联

INTRERRNO(trapErr，2); //设置 2 为系统变量值表达式，当系统变量等于预设值后，将触发 trapErr 变量关联的中断。

ERROR（2）； //设置系统变量为 2，此时便触发以上的 errnoHnd()子程序

该例中当系统变量 ERROR 被赋值时，子程序 errnoHnd()将被调用。

注意：在使用该指令前，INTR 变量必须要与一个中断关联，否则报告错误。

3.4.2.33 INTRSET 中断设置

将一个 INTR 变量与子程序连接的指令。无法设置已由先前的 INTRSET 指令设置的 INTR 变量，否则将会报错。请注意，该指令只可关联一个子程序，不要循环且重复定义一个中断（参见 INTR 变量类型）。

格式：INTRSET(INTR 变量,子程序);

输入：添加两个中断变量 INTR

输出：将 INTR 声明的两个变量分别与 inputHnd()、ErrnoHnd()子程序相关联

示例:

示例变量名:

INTR trapInput //声明 trapInput 变量

INTR trapErr //声明 trapErr 变量

示例子程序名:

inputHnd() //创建 inputHnd()子程序

ErrnoHnd() //创建 ErrnoHnd()子程序

示例程序:

INTRSET(trapInput,inputHnd()); //将 trapInput 变量与 inputHnd()中断子程序关联

INTRSET(trapErr,ErrnoHnd()); //将 trapErr 变量与 errnoHnd()中断子程序关联

INTRCOND(trapInput, io.input[5]); //设定 trapInput 变量触发的条件

INTRERRNO(trapErr，2); //设置 2 为系统变量值表达式，当系统变量等于预设值后，将触发 trapErr 变量关联的中断

ERROR（2）;//设置系统变量为 2，此时便触发以上的 errnoHnd()子程序

上面的示例展示了如何设置两个中断。

该例中当系统变量 ERROR 被赋值时，子程序 errnoHnd()将被调用。

注意：在使用该指令前，INTR 变量必须要与一个中断关联，否则报告错误。

3.4.2.34 INTRTIMER 中断频率设置

用于设置与 INTR 变量连接的定时中断的指令。设置条件后，相关中断也被启用。 如果可选参数单设置为 true，则中断只发生一次。

格式：INTRTIMER(***INTR 变量,time***);

输入：添加中断变量 INTR

输出：将 INTR 声明的变量与 trapRoutine()子程序相关联

示例:

示例变量名:

INTR trap //声明 trap 变量

示例子程序名:

trapRoutine() //创建 trapRoutine（）子程序

示例程序:

INTR trap //声明 trap 变量

INTRSET (trap, trapRoutine()) ； //将 trap 变量与 trapRoutine()中断子程序关联

INTRTIMER (trap, 2) ; //设定 time 值为 2，此时每两秒便会调用一次 trap 变量关联的 trapRoutine()子程序

上面的示例展示了如何设置中断触发的频率

注意：在使用该指令前，INTR 变量必须要与一个中断关联，否则报告错误。

3.4.2.35 KMAXJ 关节运动最大参数设置

设置关节运动参数的最大值。此指令是一个模态设置。

格式：KMAXJ([***速度***] , [***加速度***] , [***减速度***] , [***加加速度***]);

输入：设定 KMAXJ（30），设定后续笛卡尔空间关节运动速度为最大速度的 30%

输出：使用 MJOINT 指令运动机器人

示例:

示例程序 1:

系统速度为 100%

关节 2 最大速度为 150 度/秒

MJOINT(POINTJ(66.57,-23.86,-42.71,0,-0.15,0),V100perc,fine，tool0)；

KMAXJ(30);//设定后续关节运动的速度为最大速度的 30%

MJOINT(POINTJ(66.57,60,-42.71,0,-0.15,0),V100perc,fine，tool0)；

经过上面的设置，关节 2 移动的最大速度被改变为：VJ_2 = 30%*150*100%*100% = 45。

示例程序 2:

系统速度为 20%

关节 2 最大速度为 150 度/秒

MJOINT(POINTJ(66.57,-23.86,-42.71,0,-0.15,0),V100perc,fine，tool0)；

KMAXJ(30);//设定后续关节运动的速度为最大速度的 30%

MJOINT(POINTJ(66.57,60,-42.71,0,-0.15,0),V100perc,fine，tool0)；

经过上面的设置，关节 2 移动的最大速度被改变为：VJ_2 = 30%*150*100%*20% = 9。

注意：关节空间的运动参数以百分比的形式设置，参考机器人配置的最大关节速度 MAX_SPE_J，最大关节加速度 MAX_ACC_J 和最大关节加加速度 MAX_JER_J。所有参数是可选的。

3.4.2.36 KMAXT 笛卡尔空间运动最大参数

用于设置笛卡尔空间运动参数的最大值。此指令是一个模态设置。每个参数的单位与机器人内部的配置一致，正常情况下为 mm/s。所有参数都是可选的。其中速度参数是所有笛卡尔空间运动的限制条件。如果在运动指令中设置的运动值大于 **KMAXT**，则实际的运动值将会降低至 **KMAXT** 中设置值。

格式： KMAXT([速度] , [加速度] , [减速度] , [加加速度]);

输入：设定 KMAXT（40，15），设定后续笛卡尔空间运动速度为 40mm/s，加速度为 15mm/s2

输出：使用 MLIN 指令运动机器人

示例:

示例程序 1:

系统速度：100%

MLIN(POINTC(1057.91,0,1374.93,-180,69.01-180),V500,fine，tool0)；

KMAXT(40);//设定后续笛卡尔空间运动速度为 40mm/s

MLIN(POINTC(1483.09,0,1374.93,-180,69.01,-180),V30,fine，tool0)；

此例中，通过 KMAXT 指令设置后续笛卡尔空间末端线速度为 30。

示例程序 2:

系统速度：100%

MLIN(POINTC(1057.91,0,1374.93,-180,69.01-180),V500,fine，tool0)；

KMAXT(40，15);//设定后续笛卡尔空间运动速度为 40mm/s，加速度为 15mm/s2

MLIN(POINTC(1483.09,0,1374.93,-180,69.01,-180),V1000,fine，tool0)；

此例中，通过 KMAXT 指令设置后续笛卡尔空间末端线速度为 40，加速度为 15mm/s2。

示例程序 3:

系统速度：100%

MLIN(POINTC(1057.91,0,1374.93,-180,69.01-180),V500,fine，tool0)；

KMAXT(200);//设定后续笛卡尔空间运动速度为 200mm/s

MLIN(POINTC(1483.09,0,1374.93,-180,69.01,-180),V1500,fine，tool0)；

此例中，通过 KMAXT 指令设置后续笛卡尔空间末端线速度为 200。
注意：通过 KMAXT 指令设置后续笛卡尔空间运动速度为 50mm/s，加速度为 15mm/s2，减速度和加加速度数值与机器人设置一致。
每个参数不能超过为机器中的轴组配置的参数。如果设置了过大的值，则该值将被设置为机器中配置的最大值。 切向速度 TAN_SPE，切向加速度 TAN_ACC 和切向加加速度 TAN_JER 的值默认将会通过 RPE 进行计算，其值为机器人主轴笛卡尔运动参数的最小值（MAX_SPE_C [1…3], MAX_ACC_C[1…3], MAX_JER_C[1…3]）。

3.4.2.37 KMAXW 笛卡尔腕部运动最大参数

用于设置腕部轴在笛卡尔空间运动的最大参数。这是一个模态设置。

参数是依赖于机器人的配置，正常速度单位为 degree/s。所有参数都是可选的。其中速度参数是所有笛卡尔空间运动的限制条件。如果在运动指令中设置的运动值大于 **KMAXW**，则实际的运动值将会降低至 **KMAXW** 中设置值。

格式：KMAXW([***速度***] , [***加速度***] , [***减速度***] , [***加加速度***]);

输入：设定 KMAXW（20，15），设定后续笛卡尔空间运动速度为 50 degree/s，加速度为 15 degree/s^2

输出：使用 MLIN 指令运动机器人

示例:

示例程序 1:

```
系统速度：100%
MLIN(POINTC(1057.91,0,1374.93,90,69.01-180),V500,fine，tool0);
KMAXW(20,15);//设定后续笛卡尔空间运动速度为 20 degree/s，加速度为 15 degree/s2
MLIN(POINTC(1057.91,0,1374.93,0,69.01,-180),V1500,fine，tool0);
```

此例中，通过 KMAXW 指令设置后续笛卡尔空间运动速度为 20,加速度为 15 degree/s2 。减速度和加加速度数值与机器人设置一致。

示例程序 2:

```
系统速度：100%
MLIN(POINTC(1057.91,0,1374.93,90,69.01-180),V500,fine，tool0);
KMAXW(200);//设定后续笛卡尔空间运动速度为 200 degree/s
MLIN(POINTC(1057.91,0,1374.93,0,69.01,-180),V100,fine，tool0);
```

此例中，通过 KMAXW 指令设置后续笛卡尔空间运动速度为 200 。
注意：如果在 v100 中的重定向速度低于 50，则得到的速度为 v100 内的值。每个参数都不允许超过机器人配置轴组设置参数。切向速度 W_SPE，切向加速度 W_ACC 和切向加加速度 W_JER 的值默认将会通过 RPE 进行计算，其值为机器人腕部轴笛卡尔运动参数的最小值（MAX_SPE_C [4…6], MAX_ACC_C[4…6], MAX_JER_C[4…6]）。

3.4.2.38 KRES 重置笛卡尔运动参数

用于重置所有笛卡儿移动相关的工作参数到他们的默认值。这是一个模式设置。

格式：KRES

示例:

```
KMAXW (50, 15) ;
```

MLIN (POINTC(600, 600, 0, 0, 0, 0), v100, fine, tool1) ;

KRES ;

MLIN (POINTC(300, 400, 0, 0, 0, 0), v100, fine, tool1)

在这个例子中，第一个动作是用修改过的笛卡儿运动参数做出的。在此之后的第二次移动是使用默认的笛卡儿运动参数做出的，因为第二次移动之前执行了 KRES 指令。

3.4.2.39 LABEL 标签指令

在程序代码中输入一个标签名称，用于 GOTO 指令跳转到代码的特殊部分。

格式：LABEL(*标签名称*)

示例:

LABEL loop;

MLIN(pstart,v500,fine，tool1,wobj1);

...

GOTO loop;

3.4.2.40 LOAD 程序加载

用于在程序执行期间，将 xpl 程序作为模块加载到当前程序内存中。通过 EXEC 指令运行已加载的 Main 程序和子程序

格式：*LOAD (文件名);*

输入：设.XPL 格式的文件名称（编辑的文件名为需要加载的目标程序全称）

示例:

LOAD (“WH.XPL”); //将 WH.XPL 程序加载到当前程序内存中 （程序文件已创建 WH.XPL 程序）

EXEC “WH.jj” ; //执行 WH.XPL 的子程序 jj （在 WH 的主程序下已创建名为 jj 的子程序)

EXEC “WH.Main” ; //执行 WH 的主程序（执行主程序用（XXX.Main）格式）

将 WH.XPL 程序作为模块加载到当前程序中，EXCE 运行 WH.XPL 的子程序 jj 和主程序

3.4.2.41 MCIRC 圆弧运动

开始执行 TCP 从最后一个位置到目标位置的圆弧（笛卡尔）移动，经过一个中间点。

如果需要改变姿态，方向轴（A、B 和 C）将被插补并且与位置点保持同时插补（开始移动并同时到达目标位置）。

注意：如果圆弧运动的起始点和终点是同一个点，圆弧运动将是不可知的。

格式：MCIRC(*中间点,目标点,速度,区域,工具,[参考系]*)

示例:

MLIN (a, v100, fine, tool1);

MCIRC (b, c, v100, fine, tool1);

该例中圆弧运动开始于点 a，中间经过 b 点，最后到达 c 点。

3.4.2.42 MCIRCA 圆周运动

以指定的圆弧角度执行 TCP 圆周（笛卡尔）运动。圆周由三个点定义：起始点、中间点、终点。机器人根据三点确定圆，根据角度值确定机器人圆弧运动方向及角度。

机器人是以指定角度结束运动

正向运动指机器人运动方向为：起始点—中间点—目标点的圆周运动（此为圆周运动正方向）；逆向运动指机器人运动方向为：起始点—目标点—中间点的圆周运动（从起始点逆方向运动）。

警告：因为起始点是用来描述从这个指令开始执行圆周运动，如果开始时，TCP 点远离程序中的位置，可能会发生危险。

警告：若需要在 MCIRCA 中切换工具坐标系，需要确定在前一个动作结束且第二个动作开始之前，才有可能在两个连续动作之间更换工具，否则将发生错误报警（机器人移动时工具已修改）。在此状态下，前一个指令中圆弧过渡需设置为 fine。有关更多细节，请参考 MLIN 指令中的示例。

在运动过程中，圆弧运动的方向为初始角度设置的方向。此操作模式与 MCRIC 指令中使用的方向不同（MCIRC 中圆弧运动方向为：起始点—中间点—目标点），如果角度参数较大，则有可能到达一个或多个机器人轴的限位。

从 XPL2.0.0 版本之后，圆弧角度的绝对值范围为：0.0~1E200。

如果省略 refsys 参数，则使用世界坐标系进行移动

格式：MCIRCA(***中间点，目标点，角度，速度，圆弧过渡，工具，[参考系]***);

输入：中间点，目标点，机器人运行角度，速度，圆弧过渡，工具，[参考系]

输出：机器人末端 TCP 点根据指定角度进行圆周运动

示例:

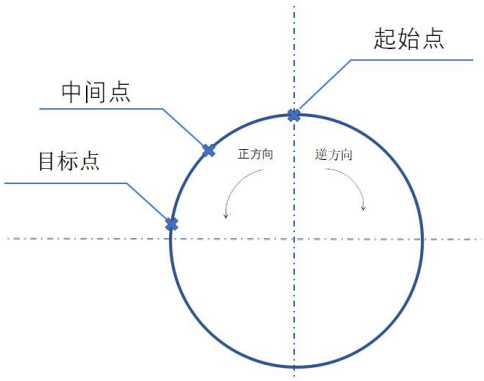
示例变量名:

示例程序:

MLIN(*, V1000, fine，tool0)；//机器人起始点

MCIRCA(*,*, -90, V1000, fine，tool0)；//机器人圆周运动，运动方向为：为逆向运动。

在本程序中机器人将沿下图逆方向运动 90°。



3.4.2.43 MESSAGE 消息指令

在系统和用户的日志文件中打印显示消息。在文本表达式中使用%1 和%2 作为表达式 1 和表达式 2 的形式参数。

格式：MESSAGE(*文本表达式，表达式 1，表达式 2*)；

输入：文本数据；

输出：文本数据；

示例：

示例变量：

POINTC _pos [0..3] = 420,0,660,0,90,0；

LREAL time；

示例程序：

MESSAGE(“Start”)；　　//日志文件中显示字符串 Start

MLIN(_pos[0], v200, fine, tool0)；　　//向数组_pos[0]做直线运动

MESSAGE (“pos %1”, _pos[0]);　　//日志中显示数组_pos[0]的值

MESSAGE(“First %1 Second %2”,10,20); //在日志文件中显示表达式 1 和表达式 2

此例中，在日志文件中显示的第一个字符串的为“Start”，第二个为“pos{“x”:420,”y”:0,”z”:660,”a”:0,”b”:90,”c”:0}”，第三个为“First 10 Second 20”。

3.4.2.44　MJOINT 关节运动

开始关节运动，从上个点到达此目标点。目标点的位置可以是 POINTC 或者 POINTJ。所有轴开始运动并同时到达目标点。TCP 运动是机器人运动各轴的组合。如果参考坐标系未指定，默认在世界坐标系下运动。

格式：MJOINT(*目标点,速度,区域,工具,[参考坐标系]*);

示例：

MJOINT (HomePos, v100, fine, tool1);

[速度]参数是用笛卡尔单位表示的，但是在此运动中，此参数用于计算关节速度的百分比。

百分比的计算如下：关节速度的百分比 = 设定切向速度/ [TAN_SPE]。

示例：

设定切向速度 = 100mm/s

TAN_SPE = 2000mm/s

关节速度的百分比=5%

默认情况下，参数 TAN_SPE 的值为由 RPE 计算的主轴笛卡尔速度参数最小值（MAX_SPE_C [1 ... 3]）。

3.4.2.45　MLIN 直线运动

从上一个位姿点直线运动到目标位姿点。如果在运动过程中要求改变姿态，姿态轴 A，B，C 将会被插补，并且与位置点保持同时插补。如果参考坐标系未指定，默认在世界坐标系下运动。

格式：MLIN(*目标点,速度,区域,工具,[参考坐标系]*);

示例：

MLIN (target1, v100, fine, tool1);

该例中按照直线移动到目标点。

3.4.2.46 PULSE 脉冲函数

为一个布尔型变量设置一个预先定义的时间量的特定值。时间用秒表示。经过一段时间后，布尔变量设置为相反的值。通过将可选参数 [前进] 设置为 true，可以不等待地运行下一条指令。

格式：PULSE(***布尔型变量,设定值,时间,[前进]***);

示例:

```
BOOL gun
...
PULSE (gun, true, 2, true) ;
MLIN (p1, v500, z20, tool1);
```

变量 gun 将会维持 2 秒的 true 状态。此变量设置为 true 后，因为 [前进] 参数为 true，后续指令可以连续地执行。

3.4.2.47 RESTART 程序从头开始执行

从主子程序的第一条指令重新开始执行程序的指令。在执行该指令后，变量的值不会进行初始化。

格式：RESTART;

示例:

```
BOOL firstRun
...
IF NOT firstRun THEN
FirstRun := true;
MJOINT(p0,v500,fine,tool0);
...
END_IF
...
MLIN(p1,v500,z20,tool1);
...
IF pieceLost THEN
RESTART;
END_IF;
...
```

此示例显示如果在工作期间丢失片段，如何跳过部分代码。如果 pieceLost 的状态是 true，程序是重新运行而不重新初始化变量，所以只有在第一次代码运行时执行 MJOINT 指令。

3.4.2.48 RETURN 返回

中断 RETURN 所在的程序或子程序的执行。

示例:

```
_IF input < 0 THEN
    RETURN ;
output := input + 1 ;
```

此例中如果变量 input 小于 0，RETRUN 所在的程序段将会被中断。

3.4.2.49 RETRY 恢复执行

重新执行被中断的指令，恢复被中断程序的执行。此指令只能在中断程序中使用。

格式：RETRY;

示例：

```
...
STOPMOVE ();
CLEARMOVE ;
RETRY ;
```

在这个例子中，在停止和清除未执行完的移动队列之后，程序返回来重新执行被中断的指令。

3.4.2.50 SAVE 保存

将模块从内存保存到文件。name 参数选择要保存的模块。如果省略了可选参数文件，模块将保存到原始位置。程序执行等待模块保存到文件结束。

格式：SAVE(***name,[file]***);

输入：模块；

输出：保存到原始位置或指定位置；

示例：

示例变量：

```
tool;
Savedframe.xpl;
```

示例程序：

```
SAVE("tool");            //保存模块"tool"
SAVE("userframe", "savedframe.xpl"); //保存模块"userframe"
```

此例中，模块"tool"将保存到原始位置，在日志文件中可以查看到"保存模块到文件</xpl/tool.xpl>"；模块"userframe"将保存到文件"savedframe.xpl"中,在日志文件中可以查看到"保存模块到文件</rpl/savedframe.xpl>"。

3.4.2.51 SETOFRAME 设置组件集

REFSYS 类型变量的同一组件集，执行指令 SETOFRAME 后，变量 wobj 将更新为新值。

格式：SETOFRAME(***wobj,oframe***);

输入：wobj, oframe；

输出：新的 wobj 值；

示例：

示例变量：

```
POINTC offset = (100,100,10,0,0,0);
```

POINTC input;

示例程序:

```
input := POINTC(50, 100, 50, 0, 0, 0); //为 POINTC 变量 input 赋值
MLIN(input, v500, fine, tool0, wobj0); //运动到 input 变量位置，此时 wobj0 默
        认为 0
SETOFRAME(wobj0, offset); //个更新 wobj0，更新变量为 offset
MLIN(input, v500, fine, tool0, wobj0); //运动到 input 变量位置，此时 wobj0 为
        更新后的值
MESSAGE("Offset: %1", wobj0); //文本中输出 wobj0 的内容
```

此例中，会在系统日志中显示：Offset:
{"uframe":{"x":0,"y":0,"z":0,"a":0,"b":0,"c":0},"oframe":{"x":100,"y":100,"z":10,"a":0,"b":0,"c":0},"movable":false,"basename":""}。

3.4.2.52 SLINB 寻位指令

自 XPL 2.1.15 版本加入。

SLINB 寻位指令可以用于寻找在从上一个 TCP 点直线运动到目标点（target）的路线中的一个点位。指令在执行期间，会监控传入布尔类型变量（boolvar）的指定变化，并会将该指定变化发生时的机器人姿态信息存储进指定的 POINTC 类型变量（varpos）中。

boolvar 参数为必须设置的 BOOL 类型参数，指令将会监控该参数值的变化。

varpos 参数为必须设置的 POINTC 类型参数，指令将会把机器人的姿态信息存储进该变量。

edge 参数为必需设置的枚举类型参数，该参数指定 boolvar 参数值得变化类型。

- Rise：上升沿触发；
- Fall：下降沿触发；
- Low：值为 false；
- High：值为 true

如果可选参数 checkinit 被设置为 true，则在运动开始之前会确认 BOOL 类型变量 boolvar 的

值。如果运动开始时，boolvar 已经为 edge 设置的值，则指令将会直接报错，并结束执行。

当可选参数 checkinit 被设置为 true 时，如果 edge 为 Rise，机器人在执行寻位指令前，boolvar 已经是 true，则指令将会直接报错，并结束执行。

当可选参数 checkinit 被设置为 true 时，如果 edge 为 Fall，机器人在执行寻位指令前，boolvar 已经是 false，则指令将会直接报错，并结束执行。

当搜索完成后，可以使用可选参数 stopmode 停止移动的类型。

- Stop：运动结束时，将正常减速，并保持减速时 TCP 点的运动轨迹依然为指令运动时的直线。
- Abort：运动结束时，机器人以快速减速到停止，该过程中不会保持 TCP 点运动轨迹与指令运动时的直线一致。

如果需要改变角度，则旋转角度(a, b 和 c)将被插入到主轴(x, y 和 z)中(将同步开始移动，并同时到达目标位置)。

如果省略 refsys 参数，则使用世界坐标系统进行运动。

enerr 参数设置为 true 时，寻位到结束未触发停止信号，在指令完成时会触发报警。

当寻位完成时，指令结束。

如果寻位在机器人进入有圆滑过渡参数决定的圆滑过渡区域前没有结束，则会有指令执行错误发出。

如果目标点位具有附加轴的位置，则附加轴将会和机器人的轴同步运动。

指令格式：

SLINB (POINTC point, SPEED speed, ZONE zone, TOOL tool, [REFSYS refsys], PONTC varpos, BOOL boolvar, ENUM edge, [ENUM stopmode], [BOOL enerr], [BOOL checkinit]) ;

警告：只有当前一个运动在第二个运动开始之前结束时，才有可能在两个连续运动之间更换工具，否则将报错。

示例：

示例变量名：

POINTC target；

POINTC foundpos;

示例程序代码：

…

Target　:= POINTC(24.58,-430.81,390.26,77.49,35.54,170.67);

…

SLINB (target, v100, fine, tool1, wobj0, foundpos, io.Din[2], Rise, Stop) ;

当 io.Din[2]从 false 变为 true 表示运动结束，此时的机器人位姿存储在变量 foundpos 中。

3.4.2.53 STOPMOVE 暂停运动

暂停机器人正在执行的运动，参数以秒为单位设置最大运动保持时间。如果没有设置保持时间，会使用系统设置的时间（通常为 0.5 秒）。暂停的运动可以通过 ***STARTMOVE*** 指令恢复运动或者使用 ***CLEARMOVE*** 指令将原有运动规划取消。

格式：STOPMOVE(***最大运动保持时间***);

输入：最大运动保持时间

输出：机器人在最大运动保持时间内暂停。

示例:

示例变量名: **INTR intr_test**

示例程序:

Main:

```
MJOINT (POINTC(400, 0, 0, 0, 180, 0), v100, fine, tool1) ;
INTRSET (intr_test,sub1());
INTRCOND (intr_test，io.DIn[10],true);
MJOINT (POINTC(400, 0, 100, 0, 180, 0), v100, fine, tool1) ;
MJOINT (POINTC(400, 0, 200, 0, 180, 0), v100, fine, tool1) ;
```

Sub1:

```
STOPMOVE (1);  //在 1 秒内使机器人暂停
CLEARMOVE;  //清除之前机器人正在进行的运动规划
STARTMOVE(); //映射上方 STOPMOVE(1)，重新开始程序
MJOINT (POINTC(400, 100, 0, 0, 180, 0), v100, fine, tool1) ;
MJOINT (POINTC(400, 200, 0, 0, 180, 0), v100, fine, tool1) ;
```

在本例中，利用中断指令使得机器人在运行 MJOINT (POINTC(400, 0, 200, 0, 180, 0), v100, fine, tool1) 时，进入子程序 sub1 中，在 1s 内暂停当前正在运行的程序并清除后续规划，以当前暂停点为起始点开始运行。

注意：此最大运行时间要求机器人在该时间内停止运行。若时间设置较长，机器人在设置时间之内已到达终点则默认该条 STOPMOVE()指令运行完毕，自动进入下条 CLEARMOVE 指令。若 STOPMOVE()前条运动指令含有圆滑过渡，该指令圆滑过渡将默认为 fine。

3.4.2.54 STARTMOVE 恢复已暂停运动

恢复之前被暂停的运动，恢复运动时间参数以秒为单位。如果没有设置恢复时间，会使用系统设置的时间（通常为 0.5 秒）。该指令通常在中断程序中使用。在运动指令后直接使用该指令是无效的。

格式：STARTMOVE(***恢复时间***);

输入：重新开始运动时间

输出：取消机器人暂停状态，开始运行。

示例:

示例变量名:

示例程序:

```
STARTMOVE(1); //在 1s 内，重新开始程序
```

3.4.2.55 STOPPROG 停止程序

停止当前执行的程序

格式：*STOPPROG;*

示例程序:

```
 MJOINT (*, v500, fine, tool0);    //运行轨迹
STOPPROG;      //程序暂停
DWELL(1.000);      //等待 1 秒
MJOINT (*, v500, fine, tool0);    //运行轨迹
```

程序中 STOPPROG 指令将当前运行程序暂停，程序指针依然存在，按下开始执行程序按钮（播放按钮），程序继续运行

3.4.2.56 TIMESTART 开始计时

基于一个 timer 数据类型的变量启动一个计时器。时间用秒表示。只有当程序正在运行时，计时器才能被停止。如果计时器已经激活，则用新的时间值设置计时器。

格式：*TIMERSTART (variable, time);*

示例:

示例变量名:

```
TIMER tmr1;
```

示例程序:

```
TIMERSTART (tmr1, 5.2) ;
...
WAIT (TIMERQ(tmr1)) ;
```

这个例子展示了如何启动一个计时器并一直等待直到过期。

3.4.2.57 TIMERSTOP 停止计时

停止指定 TIMER 类型的计时器变量的计时。

格式：*TIMERSTART (variable, time);*

示例:

示例变量名:

TIMER tmr1;

示例程序:

```
TIMERSTART (tmr1, 5.2) ;
...
TIMERSTOP (tmr1);
...
TIMERSTART (tmr1, TIMERR(tmr1));
...
WAIT (TIMERQ(tmr1)) ;
```

这个例子展示了如何启动和停止一个计时器。它还解释了如何继续计时器计数并等待直到过期。

3.4.2.58 TRIGCALL 设定触发

设置触发器类型的变量。当达到触发条件时，调用特定的程序。子程序不能有输入或输出参数。

格式: TRIGCALL ***触发器名,类型，参考值，子程序（）***；

触发的类型可以是:

1) Distance 距离：与目标点之间的距离。

2) TimeToEnd 时间：到达目标点的时间。

触发变量的数值可以是（与触发变量类型有关）

如果是距离触发，单位为毫米。

如果是时间触发，单位为秒。

示例:

```
TRIGCALL trig2, Distance, 150, OpenGrimper(1) ;
...
TRIGON(trig2);
MLIN(p84,v500,fine,tool1,wobj4);
```

上述指令含义是当 trig2 变量被触发时子程序 OpenGrimper 将会被执行，触发器 trig2 将在距离 p84 目标点 150mm 时被触发，所以当机器人在距离目标点 150mm 时，子程序 OpenGrimper 将会被执行。

3.4.2.59 TRIGON 激活触发

在下一个运动中激活触发变量。使用 TRIGON 指令最多可以设置 4 个事件。如果多次调用 TRIGON，所有的触发变量将会在运动前的最后一个 TRIGON 指令激活。调用运动指令后，将没有触发变量被设置，因此必须使用多个 TRIGON 指令在下一个运动中来激活触发变量。

注意：激活变量只能为 MLIN 和 MCIRC 运动指令使用。其他运动指令 MJOINT 或者 EPATH 不能用于处理事件。

格式: TRIGON (***触发器 1, [触发器 2], [触发器 3], [触发器 4]***) ;

示例:

```
trig1 := TRIGSET when Distance is 50 do (grimper = 1) ;
TRIGON (trig1);
MLIN(pa) ;
trig2 := TRIGSET when TimeToEnd is 1 do (airgun = 1) ;
```

TRIGON (trig2);

MLIN(pb) ;

此例中，变量 grimper 在距离 pa 点 50mm 处被设置为 1。然后在到达 pb 点 1 秒前，变量 airgun 将会被设置为 1。

对于运动到 点 pa 处，只能使用触发事件 trig1。对于到达 pb 点，只能激活事件 trig2。

3.4.2.60 TRIGSET 设置触发

设置触发器类型的变量。当触发触发器的条件时，将使用表达式的值设置指定的变量。执行 **TRIGSET** 指令时计算表达式的值。

格式： ***trigger_var*** := ***TRIGSET*** when ***type*** ***isref_value*** do (***var_to_set*** := ***expr_to_set***)

触发的类型可以是:

1) Distance 距离：与目标点之间的距离。

2) TimeToEnd 时间：到达目标点的时间。

触发变量的数值可以是（与触发变量类型有关）

如果是距离触发，单位为毫米。

如果是时间触发，单位为秒。

var_to_set:必须是数值变量。

expr_to_set:当事件被触发时，表达式的值将会被设定。

示例程序:

TRIGGER trigl

...

trig1 := TRIGSET when Distance is 100 do (glue := 1) ;

...

TRIGON(trig1)

MLIN(p12,v100,fine,tool1,wobj0);

上例中机器人在距离目标点 P12 100mm 时，触发变量 trig1 被触发，变量 glue 的值将会被设置为 1。

3.4.2.61 TRYNEXT 恢复执行下一个

恢复被中断程序的执行，该程序将执行被中断指令后面的指令。

此指令只能在中断程序中使用

格式: TRYNEXT;

示例:

...

STOPMOVE ();

CLEARMOVE ;

TRYNEXT;

在这个例子中，在停止和清除未执行的移动之后，程序返回执行被中断指令后面的指令。

3.4.2.62 WAIT 事件等待

等待条件的执行，如果设置了超时，则可以选择中断等待。如果等待正确终止，则

可选参数的结果将为 0；如果超时，则不等于 0。[***超时***] 和 [***结果***] 参数是可选的。

格式：WAIT (***条件, [超时], [结果]***);

示例:

```
BOOL boxtokeep
...
WAIT (boxtokeep);
MLIN (pick,v100,fine,tool1);
```

此例中在 boxtokeep 为 **true**，**MLIN** 运动将会被执行。

3.4.2.63 UNLOAD 卸载

从先前加载指令 **LOAD** 的存储器中卸载模块。 如果是可选参数 [***保存***] 被定义为 **true**，加载模块后模块被更改将被保存在卸载之前。

格式：UNLOAD (***文件名，保存***);

示例:

```
LOAD (“parprog.xpl) ;
…
UNLOAD (“parprog.xpl”, true) ;
```

在此示例中，从“parprog.xpl”加载的模块将从内存中卸载。如果是模块加载后更改模块将被保存。

3.4.2.64 WAIT_POS 位置等待

从机器人拟定圆弧过渡位置开始计算等待机器人到达目标点（机器人将忽略设定的圆弧过渡数值，并沿圆弧过渡为 0 的轨迹进行运动）。如果 *WAIT_POS* 指令中设置了超时并且形参变量 var 有指派，则当机器人在超时范围之内未到达目标点，则变量将被置为非零（当机器人无报警时，则会以该时刻机器人位置进行圆弧过渡进行运行）。否则，当机器人在超时范围内到达目标点，机器人形参变量将被置为 0。所有变量均为可选。

格式：WAIT_POS(***[超时]，[变量]***);

输入：超时时间，指令结果变量

输出：机器人根据是否在超时时间内到达运动指令的指定位置。

示例:

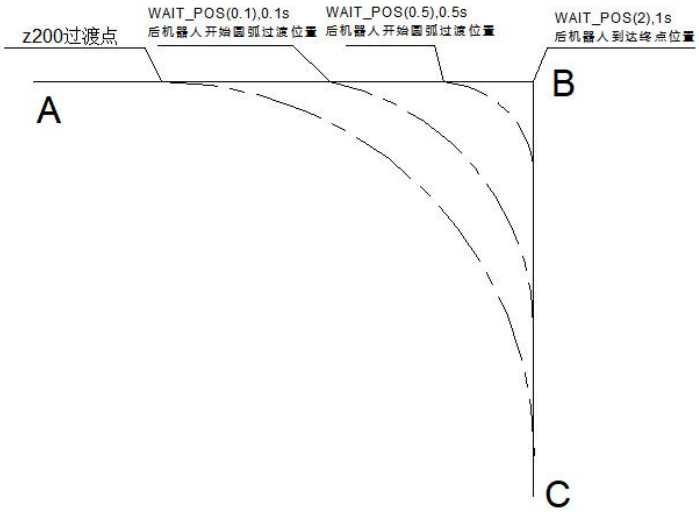
示例变量名: **int var_test;**

示例程序:

```
MJOINT (POINTC(400, 0, 0, 0, 180, 0), v100, z200, tool1) ;
MLIN (POINTC(800, 600, 0, 0, 180, 0), v100, z200, tool1) ;//AB 线段运动
```

WAIT_POS (1，var_test) ; //当机器人运行上行 MLIN 指令时，并运行至 z200 圆弧过渡拟过渡处，机器人将继续以圆弧过渡 fine 进行直线运动。1s 后若未运行至 POINTC(800, 600, 0, 0, 180, 0)点，机器人将以当前位置进行圆弧过渡进行运动，且 var_test 被置为非零数。反之，机器人将运行至 POINTC(800, 600, 0, 0, 180, 0)点，并继续运行下点，var_test 被置为 0。

MLIN (POINTC(800, 400, 0, 0, 180, 0), v100, z200, tool1) ; //BC 线段运动



3.4.2.65 WAIT_ZONE 圆弧等待

从机器人拟定圆弧过渡位置开始计算等待机器人到达 *WAIT_ZONE* 指派圆弧过渡位置（机器人将忽略设定的圆弧过渡数值，并沿圆弧过渡为 0 的轨迹进行运动）。如果 *WAIT_ZONE* 指令中设置了超时并且形参变量 var 有指派，则当机器人在超时范围之内未到达指派圆弧过渡位，则变量将被置为非零（当机器人无报警时，则会以该时刻机器人位置进行圆弧过渡进行运行）。否则，当机器人在超时范围内到达目标过渡点，机器人形参变量将被置为 0。所有变量均为可选。

格式: WAIT_ZONE(***zone, [超时]，[变量]***);

输入: 圆弧过渡等待位，超时时间，指令结果变量

输出: 机器人根据是否在超时时间内到达指定圆弧过渡位置，输出结果变量。

示例:

示例变量名: **int var_test;**

示例程序:

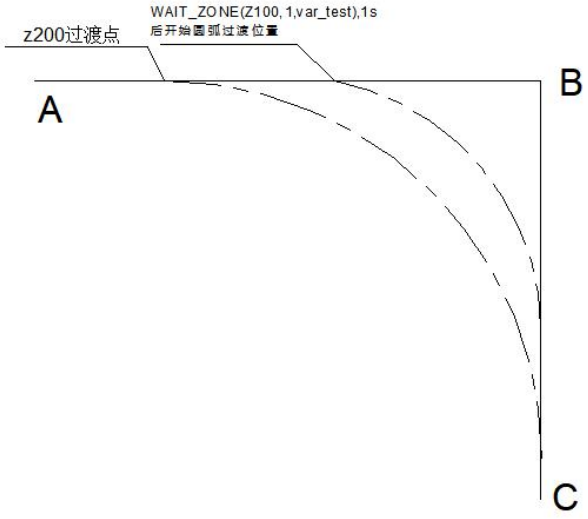
MJOINT (POINTC(400, 0, 0, 0, 180, 0), v100, z200, tool1) ;

MLIN (POINTC(800, 600, 0, 0, 180, 0), v100, z200, tool1) ;//AB 线段运动

WAIT_ZONE(Z100, 1，var_test) ; //当机器人运行上行 MLIN 指令并运行至 z200 圆弧过渡拟过渡处，机器人将继续以圆弧过渡 fine 进行直线运动。1s 后若未运行至 POINTC(800, 600, 0, 0, 180, 0)点，机器人将以当前位置进行圆弧过渡进行运动，且 var_test 被置为非零数。

反之，机器人将运行至 POINTC(800, 600, 0, 0, 180, 0)点，并继续运行下点，var_test 被置为 0。

MLIN (POINTC(800, 400, 0, 0, 180, 0), v100, z200, tool1) ; //BC 线段运动



3.4.2.66 WHILE 循环

如果 while 条件为真，执行一个代码块。

格式: WHILE condition DO

...

END_WHILE ;

示例:

```
WHILE i <= 10 DO
    i := i + 1 ;
    MLIN (POINTC(200, -500, 200, 0, 180, 0),v100, fine, tool1) ;
    MLIN (POINTC(200, -900, 200, 0, 180, 0),v100, fine, tool1) ;
END_WHILE;
```

本例中，当变量 i 的值大于 10 时，while 循环将停止。当键入 **WHILE** 指令时，**END_WHILE** 会自动添加。

3.4.2.67 WRISTMODE 手腕模式

设置用于后续动作的手腕插补模式。

格式: WRISTMODE(type) ;

机器人手腕模式变量 type，可以为一下几种:

● EulerAngles

这是默认模式，工具重定向过程中使用欧拉角进行插值。

● ObjectFrame

首先通过首末姿态的 Z 轴分量确定一个平面，然后姿态 Z 轴分量在此平面内从起始点到终止点进行线性插补，与此同时，通过首末姿态的 X 和 Y 分量分别定义为绕姿态 Z 轴分量的旋转，并其旋转进行线性插补

● PathFrame

此腕部模式与用户坐标系模式类似，但是起始点和终止点的姿态是基于路径方向定义的。直线运动中，路径坐标系模式与用户坐标系模式等同；而圆弧运动中，首先将首末姿态转化为相对圆弧平面的姿态，然后进行插补。

● AuxPointOri

在直线运动中，等同于用户坐标系模式。在圆弧运动，此模式下的姿态规划将使用四元数的三次样条方法对起始点姿态、中间点姿态，和终止点姿态进行姿态插补。

● WristJoint

初始点和终止点的位置逆解为关节轴的位置，其中腕关节轴（J4，J5，J6）采用关节线性插补，而后其他轴（J1，J2，J3）的插补保证使 TCP 位置运动目标点路径为运动指令中的笛卡尔路径（直线或圆弧）。此模式可以用来规避腕部奇异点，但是路径运动中的姿态是未在笛卡尔空间中进行定义。

所有后续动作将遵循所选的手腕模式。所选的手腕模式一直有效直到系统重新启动。

示例:

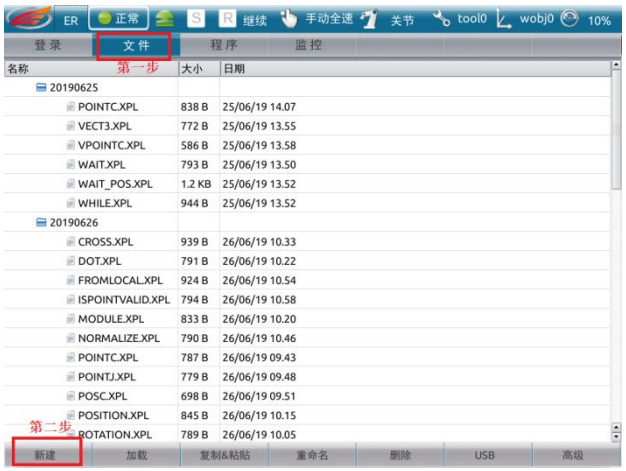
```
MLIN (p11, v100, fine, tool1, wobj0) ;
…
WRISTMODE (ObjectFrame) ;
MLIN (p12, v100, fine, tool1, wobj0) ;
```

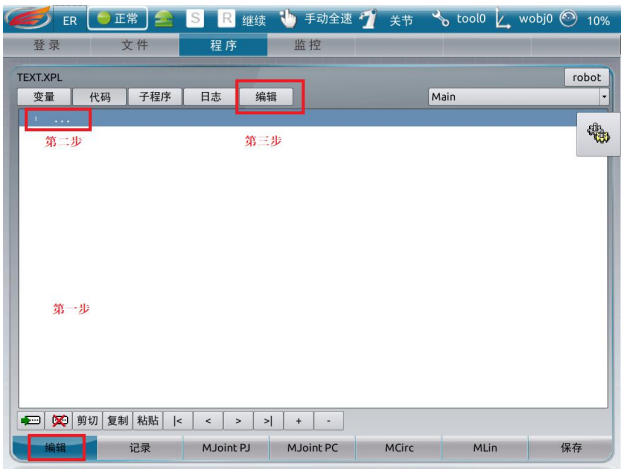
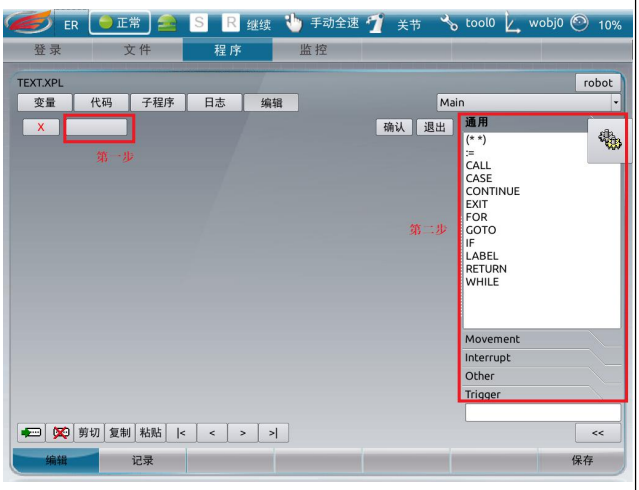
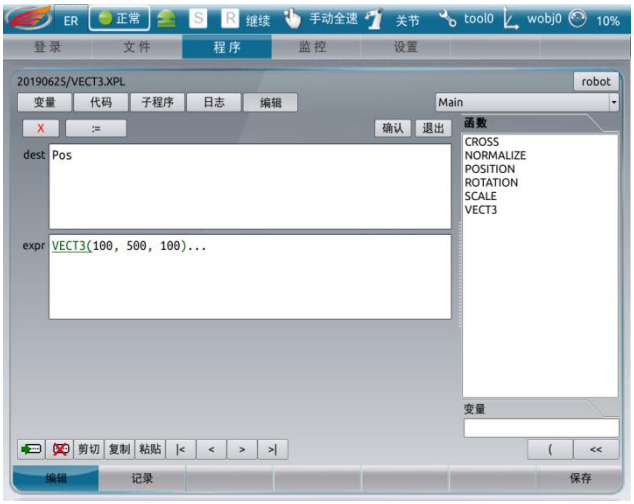
本例中，移动到目标 p11 使用实际选择的手腕模式，默认使用欧拉角插值。

移动到目标 p12 使用 ObjectFrame 手腕插补。

3.5 函数

3.5.1 函数添加步骤

步骤	图片	描述
1.登录。		p. 击标题栏“登录”后输入密码(999999)。 q. 击“登录”。
2.创建文件夹或文件。		r. 界面中选择文件选项。 s. 击新建，新建文件夹或文件。 t. 定文件夹或文件名称。 u. 需加载文件，选中文件后，点击下方加载即可。

3．编写程序准备。		v. 击“编辑”进入编辑功能。 w. 中需要编辑的行。 x. 击上方编辑。
4. 选择赋值函数 :=		y. 择赋值函数后添加一个变量和其对应的值 z. 中变量点击“新建”即可创建各个类型的变量，最后点击“确认”即可。
5.程序确认。		aa. 中需要填写的参数。 ab. 变量范围内选择刚刚创建的变量。 ac. 变量进行相应的设置和应用。 ad. 击“确认”，创建出坐

		标系。（下图）。
		ae. 该函数中可继续添加子函数，如 DOT 函数，继续设置他对应的参数和值即可完成设置。

说明：其他函数添加方法与该表类似，具体方法根据指令的不同而稍有不同。

3.5.2 表达式函数

3.5.2.1 ABS 绝对值

返回值的绝对值。

格式：ABS(value);

输入：常量；

输出：输入值的绝对值；

示例：

示例变量：

```
DINT input1 = -5;
LREAL input2 = -0.5;
```

示例程序：

```
_IF input1 < 0 THEN;                    //判断：当 input1<0 进入判断
        input1 := ABS(input1);          //将 input1 的绝对值赋给自己
input2 := ABS(input2);                  //将 input2 的绝对值赋给自己
END_IF;                                 //结束判断
MESSAGE("First %1 Second %2",input1,input2);//文本中输出 input1 和 input2
```

此例中因为 input1 初始化中小于 0，所以会进入判断进行绝对值化，并将结果赋给自己，所以最后在系统日志文件中会显示“First 5 Second 0.5”。

3.5.2.2 ASIN 反正弦

数学上反三角函数中的反正弦 arcsinx，表示一个正弦值为 x 的角，该角的范围在[-π/2，π/2]区间内。

格式：ASIN(***value***);

输入：[-1，1]之间的值；

输出：输入值的反正弦值，区间在[-π/2，π/2]；

示例：

示例变量：

```
LREAL lSin = 1;
```

示例程序：

```
_IF lSin THEN;          //判断：当 lSin > 0 进入判断
        lSin := ASIN(lSin) / PI();   //将 lSin 的反正弦值除以π赋给自己
END_IF;                 //结束判断
MESSAGE("arcsin %1",lSin);  //文本中输出 lSin 的值
```

此例中因为 lSin 初始化中大于 0，所以会进入判断，最后在系统日志文件中会显示"arcsin 0.5"。

3.5.2.3 ACOS 反余弦

数学上反三角函数中的反余弦 arccosx，表示一个余弦值为 x 的角，该角的范围在[0，π]区间内。

格式：ACOS(***value***);

输入：[-1，1]之间的值；

输出：输入值的反余弦值，区间在[0， π]；

示例：

示例变量：

```
LREAL lCos = -1;
```

示例程序：

```
_IF lCos < 0 THEN;          //判断：当 lCos < 0 进入判断
        lCos := ACOS(lCos) / PI();   //将 lCos 的反余弦值除以π赋给自己
END_IF;                 //结束判断
MESSAGE("arccos %1",lCos);  //文本中输出 lCos 的值
```

此例中因为 lCos 初始化中小于 0，所以会进入判断，最后在系统日志文件中会显示"arccos 1"。

3.5.2.4 ATAN2 反正切

数学上反三角函数中的反正切 arctanx，表示一个正切值为 x 的角，该角的范围在（-π/2，π/2）区间内。定义域 R，值域（-π/2，π/2）。

格式：ATAN2(***y，x***);

输入：[-1，1]之间的值；

输出：输入值的反余弦值，区间在[0， π]；

示例：

示例变量:

DINT val ;

DINT valB ;

示例程序:

val := ATANS(SQRT(2)，SQRT(2)) / PI()； //2 的平方根相除的反正切值除以π赋给 val

valB := ATANS(SQRT(3)，1) / PI() * 180； //3 的平方根除以 1 的反正切值除以π再乘以 180 赋给自己

变量 val 将为 0.25；变量 valB 将为 60。

3.5.2.5 COS 余弦

余弦三角函数。

格式: COS(***radiant***);

输入: 定义域为整个实数集;

输出: 值域[-1，1];

示例:

示例变量:

DINT valA ;

DINT valB ;

示例程序:

valA := COS(PI())； //π的余弦赋值给 valA

valB := COS(0)； //0 的余弦赋值给 valB

变量 valA 将为-1；变量 valB 将为 1。

3.5.2.6 LIMIT 限制

在最大值和最小值范围内，指定值小于最小值则返回最小值，指定值在范围内则返回指定值，指定值大于最大值则返回最大值。

格式: LIMIT(***value，minimum value，maximum value***);

输入: 指定值，最小值，最大值;

输出: 限定值;

示例:

示例变量:

DINT valA ;

DINT valB ;

DINT valC ;

示例程序:

valA := LIMIT(45, 10 ,100)； //指定值 45 处于 10-100 范围哪个位置返回给 valA

valB := LIMIT(800, 10 ,100)； //指定值 800 处于 10-100 范围哪个位置返回给 valB

valC := LIMIT(-20, 10 ,100)； //指定值-20 处于 10-100 范围哪个位置返回给 valC

变量 valA 将为 45；变量 valB 将为 100；变量 valC 将为 10。

3.5.2.7 MAX 最大值

对 a 和 b 进行比较，返回其中的最大值。

格式：MAX(***value a，value b***);

输入：数值 a，数值 b；

输出：a 和 b 中的最大值；

示例：

示例变量：

DINT valA ；

DINT valB ；

DINT valC ；

示例程序：

valA := MAX(3 , 7)； //比较 3 和 7 的大小并将最大值返回给 valA

valB := MAX(10 , -2)； //比较 10 和-2 的大小并将最大值返回给 valB

valC := MAX(valA, valB)； //比较 valA 和 valB 的大小并将最大值返回给 valC

变量 valA 将为 7；变量 valB 将为 10；变量 valC 将为 10。

3.5.2.8 MIN 最小值

对 a 和 b 进行比较，返回其中的最小值。

格式：MIX(***value a，value b***);

输入：数值 a，数值 b；

输出：a 和 b 中的最小值；

示例：

示例变量：

DINT valA ；

DINT valB ；

DINT valC ；

示例程序：

valA := MIX(3 , 7)； //比较 3 和 7 的大小并将最小值返回给 valA

valB := MIX(10 , -2)； //比较 10 和-2 的大小并将最小值返回给 valB

valC := MIX(valA, valB)； //比较 valA 和 valB 的大小并将最小值返回给 valC

变量 valA 将为 3；变量 valB 将为-2;变量 valC 将为-2。

3.5.2.9 MOD 取余数

值除以除数取余值。

格式：MOD(***value，divisor***);

输入：数值 a，数值 b；

输出：a÷b 的余值；

示例：

示例变量：

```
DINT valA ；
```

示例程序：

```
valA := MOD(10 , 3)；  //10 除以 3 将余值返回给 valA
```

变量 valA 将为 1。

3.5.2.10 PI 圆周率 π

返回圆周率π（3.1415926.....）。

格式：PI();

输入：半径 r;

输出：周长 len;

示例：

示例变量：

```
LREAL valA ；
DINT r = 10；
LREAL Len ；
```

示例程序：

```
valA := PI()；    //将圆周率π赋值给 valA
Len := 2 * valA * r；  //计算周长 Len
```

变量 valA 将为 3.1415926……，变量 Len 将为 62.831853……。

3.5.2.11 RAND(最小值，最大值)

返回最小值和最大值之间的随机数。

格式：RAND(minimum value, maximum value);

输入：最小值，最大值;

输出：随机值（介于最小值与最大值之间）；

示例：

示例变量：

```
DINT min = 0；
DINT max = 100；
LREAL random;
```

示例程序：

```
random := RAND(min, max)；    //将 min 到 max 之间的随机值赋给 random
```

```
_IF random < 50 THEN;      //判断：当 random < 50 进入判断
        MESSAGE("The random < 50");//文本中输出"The random < 50"
END_IF;                    //结束判断
```

变量 random 将为 0~100 中间的任意值，如 10.1，当变量小于 50 则进入判断，最后在系统日志文件中会显示"The random < 50"，当变量大于或等于 50 则不进入判断。

3.5.2.12 RANGE(值，最小值，最大值)

用于检验值是否在最小值最大值之间，如果在该范围内，将返回 true。

格式：RANGE(value, minimum value, maximum value);

输入：指定值，最小值，最大值;

输出：TRUE 或 FALSE;

示例：

示例变量：

```
DINT min = 0;
DINT max = 100;
LREAL random;
LREAL value;
```

示例程序：

```
random := RAND(min, max);     //将 min 到 max 之间的随机值赋给 random
value := RANGE(random, 0, 50);  //检验 random 是否在[0,50]之间
_IF value THEN;        //判断：当 value 为真进入判断
        MESSAGE("The random <= 50");//文本中输出"The random < 50"
RETURN;                //中断程序执行
END_IF;                //结束判断
MESSAGE("The random > 50");  //文本中输出"The random > 50"
```

变量 random 将为 0~100 中间的任意值，如 10.1，当变量小于或等于 50 则 value 为真进入判断，最后在系统日志文件中会显示"The random <= 50"；当变量大于 50 则不进入判断，最后在系统日志文件中会显示"The random > 50"。

3.5.2.13 ROUND(值) 近似值

输入一个值，返回与该值最接近值的整数，若输入值是整数，则返回自身。

格式：ROUND(value);

输入：指定值;

输出：输入值的近似整数值;

示例：

示例变量：

```
DINT ValueA;
DINT ValueB;
DINT ValueC;
LREAL value;
```

示例程序：

```
value := 23;          //将变量 value 赋值为 23
ValueA := ROUND(value); //将变量 value 值的近似整数值赋值给变量 ValueA
value := 5.23;        //将变量 value 赋值为 5.23
ValueB := ROUND(value); //将变量 value 值的近似整数值赋值给变量 ValueB
value := -0.55;       //将变量 value 赋值为-0.55
ValueC := ROUND(value); //将变量 value 值的近似整数值赋值给变量 ValueC
```

变量 ValueA 的结果为 23，变量 ValueB 的结果为 5，变量 ValueC 的结果为-1。

3.5.2.14 SIGN(值) 正负值判断

对输入值进行判断，如果输入值是正值，则返回+1，如何输入值为负值，则返回-1。

格式：SIGN(value);

输入：指定值;

输出：输入值的正负;

示例：

示例变量：

```
LREAL absValueA;
LREAL absValueB;
```

示例程序：

```
absValueA := SIGN(-24.5) * -24.5；//将-24.5 的符号再乘以-24.5，将值赋给 absValueA
absValueB := SIGN(5.1) * 5.1;  //将 5.1 的符号再乘以 5.1，将值赋给 absValueB 变量
absValueA 的结果为 24.5，变量 absValueB 的结果为 5.1。
```

3.5.2.15 SIN(弧度) 正弦值

正弦三角函数。

格式：SIN(radiant);

输入：全体实数;

输出：[-1，1];

示例：

示例变量：

```
LREAL ValueA;
LREAL ValueB;
```

示例程序：

```
ValueA := SIN(PI() / 2)；//将π/2 的正弦值赋值给 ValueA
ValueB := SIN(0);   //将 0 的正弦值赋值给 ValueB
```

变量 ValueA 的结果为 1，变量 ValueB 的结果为 0。

3.5.2.16 SQRT(值) 平方根

将输入的值转换成该值的平方根输出。

格式：SQRT(value);

输入：正实数;

输出：实数;

示例：

示例变量：

LREAL ValueA;

LREAL ValueB;

示例程序：

ValueA := SQRT(16)；　//将 16 的平方根赋值给 ValueA

ValueB := S　　QRT(1.21)；//将 1.21 的平方根赋值给 ValueB

变量 ValueA 的结果为 4，变量 ValueB 的结果为 1.1。

3.5.2.17　TAN(弧度) 正切值

正切三角函数。

格式：TAN(radiant);

输入：{x|x≠(π/2)+kπ,k∈Z};

输出：实数;

示例：

示例变量：

LREAL ValueA;

LREAL ValueB;

示例程序：

ValueA := TAN(PI() / 4)；//将π/4 的正切值赋值给 ValueA

ValueB := TAN(0);　//将 0 的正切值赋值给 ValueB

变量 ValueA 的结果为 1，变量 ValueB 的结果为 0。

3.5.2.18　TFB 时间锚点

返回从引导开始经过的时间（以秒为单位）。也可用于时间测量。

格式：TFB();

示例：

示例变量：

LREAL TimeA;

LREAL TimeB;

LREAL Time;

示例程序：

TimeA := TFB()；　　//记录当前时间锚点到 TimeB

MJOINT (*, v500, fine, tool0); //机器人关节运动

MJOINT (*, v500, fine, tool0); //机器人关节运动

TimeB := TFB();　//记录当前时间锚点到 TimeB

Time := TimeB - TimeA;　//TimeA 到 TimeB 的时间差的值赋值给 Time

变量 Time 的结果为 TimeA 到 TimeB 的差值。

3.5.2.19　TRUNC 求整

返回值的整数部分。

格式：TRUNC(value);

输入：实数;

输出：整实数;

示例:

示例变量:

```
DINT ValueA;
DINT ValueB;
DINT ValueC;
```

示例程序:

```
ValueA := TRUNC(4.5);      //将 4.5 的整数值部分赋值给 ValueA
ValueB := TRUNC(-1.11);    //将-1.11 的整数值部分赋值给 ValueB
ValueC := TRUNC(TAN(60));  //将正切函数 tan(60°)的值赋值给 Value
```

变量 ValueA 的结果为 4,ValueB 的结果为-1，ValueC 的结果为 0。

3.5.2.20 TRACKING 跟踪

TRACKING 数据类型用于存储跟踪应用程序的数据。设置此类数据必须使用 TRACKING 函数。有关此参数的说明，请参阅跟踪应用的说明文档。初始化后的默认值是无效的 TRACKING。

格式：TRACKING(maxspace, startspace, maxspe, maxacc, type, par1);

输入：最大行程，起始速度，最大速度，最大加速度，类型，标准;

示例:

示例变量:

```
TRACKING cvy;
POINTC target;
REFSYS wobj;
REFSYS rsPhoto;
```

示例程序:

```
target := POINTC(600, 0, 500, 0, 180, 0);      //给 pointc 变量 target 赋值
cvy := TRACKING(1000, 700, 100, 1000, Linear, 0.1);  //为 tracking 指令添加应用信息
wobj := GETTRKWOBJ(rsPhoto, cvy);      //设置特定参考系的跟踪参数并返回可移动参考系
MLIN(target, v500, fine, tool1, wobj);      //机器人运动
```

此例显示如何为跟踪应用程序设置跟踪参数。

3.5.3 位姿向量和空间点变量函数

3.5.3.1 VECT3 三维向量

表示三维向量的数据类型（三维向量即空间上有大小和方向的量）。数据类型为 LREAL

格式：VECT3(X，Y，Z);

示例:

```
pos_wh :=VECT3(-80，240，500 );  //给变量 pos_wh 赋值
```

变量 pos_wh 的值为（-80，240，500）

3.5.3.2 VPOINTC 六维向量

返回由两个矢量组成的点：位置（x，y，z）和旋转（rotX，rotY，rotZ），注意 POINTC 数据类型中 a 对应 rotZ，b 对应 rotY，c 对应 rotX

格式：VPOINTC(pos，rot，cfg);

示例:

```
POINTC    pos_ss                    //定义一个 POINTC 类型的变量
pos_ss:=VPOINTC(VECT3(30，12，6)，VECT3(0，8，-90));    //给变量 pos_rj 赋值
```

数据 POINTC 是由 lreal 类型 x,y,z,a,b,c 组成，则变量 pos_ss 的值为（30，12，6，-90，8，0）

3.5.3.3 VEPOINTC 多维向量

返回由两个矢量位置（x，y，z），旋转（rotX，rotY，rotZ）和存储外部附加轴关节位置组成的点

格式：***VEPOINTC(pos，rot，ea，cfg)***;

输入：pos（VECT3 类型），rot（VECT3 类型），ea（POINTJ 类型），cfg（可选参数 cfg 用于定义特定的轴配置(CFG0 相当于省略 CFG 参数)）

示例:

示例变量:

```
VECT3          pos_yj       //定义一个 vect3 类型的变量
VECT3          rot_jh       //定义一个 vect3 类型的变量
POINTJ         eaux_jj      //定义一个 pointj 类型的变量
VEPOINTC       get_point    //定义一个 vepointc 类型的变量
```

示例程序:

```
pos_yj:=VECT3(100，200，300);          //给变量 pos_yj 赋值
rot_jh :=VECT3(0，180，-90);           //给变量 pos_jh 赋值
eaux_jj :=POINTJ(1，2，3，4，5，6);        //给变量 eaux_jj 赋值
get_point :=VEPOINTC(pos_yj，rot_jh，eaux_jj);    //给变量 get_point 赋值
```

变量 get_point 的值为(100，200，300，-90，180，0，1，2，3，4，5，6)

3.5.3.4 POINTC 笛卡尔空间位姿点

返回一个由六个实数组成的笛卡尔空间点，其中 a 是关于 z 轴的旋转，b 是关于 y 轴的旋转，c 是关于 x 轴的旋转

格式：POINTC(x，y，z，a，b，c);

输入：输入类型为 LREAL 的变量

示例:

示例变量:

```
POINTC    pos_jh    //定义一个 POINTC 类型的变量
```

示例程序:

```
 pos_jh = POINTC(10.1，20，-30.6，12，14，15);    //给变量 pos_jh 赋值
MLIN (pos_jh , v250, fine, tool0, wobj0) ;        //直线方式运行到 pos_jh 点
```

定义一个 pointc 类型的点，运行到定义的点

3.5.3.5 POINTCOVR 覆盖笛卡尔空间位姿点量

覆盖原有的笛卡尔点的数值，返回新的值

格式：POINTCOVR(pt，x，y，z，a，b，c，cfg);

输入：pt（POINTC 类型），x，y，z，a，b，c（LREAL 类型）

示例:

示例变量:

POINTC pos_jh //定义一个 POINTC 类型的变量

POINTC pos_new //定义一个 POINTC 类型的变量

示例程序:

pos_jh = POINTC(1，2，3，4，5，6); //给变量 pos_jh 赋值

pos_new= POINTCOVR(pos_jh，100，，，，，，); //给变量 pos_new 赋值

pos_new 的值为(100，2，3，4，5，6)

3.5.3.6 POINTJ 关节位置

数据用于定义机器人的关节位置

格式：POINTJ(j1，j2，j3，j4，j5，j6);

输入：各个轴位置 j1,j2,j3,j4,j5,j6(LREAL 类型)

示例:

示例变量:

POINTJ pos_nl //定义一个 POINTJ 类型的变量

示例程序:

pos_nl = POINTJ(30，40，-90，90，90，0); //给变量 pos_nl 赋值

MJOINT (pos_nl , v250, fine, tool0, wobj0) ; //关节方式运行到 pos_nl 点

定义一个 pointc 类型的点，运行到定义的点

3.5.3.7 POINTJOVR 覆盖关节位置

覆盖原有的关节点的数值，返回新的值

格式：POINTJOVR(pt，j1，j2，j3，j4，j5，j6);

输入：pt（POINTJ 类型），各个轴位置 j1,j2,j3,j4,j5,j6(LREAL 类型)

示例:

示例变量:

POINTJ pos_nl //定义一个 POINTJ 类型的变量

POINTJ pos_new //定义一个 POINTJ 类型的变量

示例程序:

pos_nl = POINTJ(1，2，3，4，5，6); //给变量 pos_nl 赋值

pos_new = POINTJOVR(pos_nl ，，，，1，2，3); //给变量 pos_nl 赋值

变量 pos_nl 的值为(1，2，3，1，2，3)

3.5.3.8 EPOINTC 笛卡尔空间位姿和附加轴关节位置点组合

数据用于返回由 12 个实数值组成的扩展点

格式：EPOINTC(x，y，z，a，b，c，ea1，ea2，ea3，ea4，ea5，ea6，cfg);

输入：各个轴位置 j1,j2,j3,j4,j5,j6(LREAL 类型)，外部附加轴关节位置 ea1,ea2,ea3,,ea4,ea5,ea6（LREAL

类型）

示例:

示例变量:

EPOINTJ　pos_fc　//定义一个 POINTC 类型的变量

示例程序:

pos_fc =EPOINTC(30，40，-90，90，90，0，2，5，8，3，6，9);　//给变量 pos_fc 赋值

程序给变量 pos_fc 赋值

3.5.3.9　EPOINTCOVR 覆盖位姿和附加轴关节位置点组合

覆盖原有数据

格式：EPOINTCOVER(pt，x，y，z，a，b，c，ea1，ea2，ea3，ea4，ea5，ea6，cfgx);

输入：pt，各个轴位置 j1,j2,j3,j4,j5,j6(LREAL 类型)，外部附加轴关节位置 ea1,ea2,ea3,,ea4,ea5,ea6（LREAL 类型）

示例:

示例变量:

EPOINTC　pos_fc　　//定义一个 EPOINTC 类型的变量

EPOINTC　pos_new　　//定义一个 EPOINTC 类型的变量

示例程序:

pos_fc =EPOINTC(1，2，3，4，5，6，7，8，9，10，11，12);　//给变量 pos_fc 赋值

pos_new =EPOINTCOVR(pos_fc，，，，1，2，3，1，2，3，1，2，3);　//给变量 pos_fc 赋值

变量 pos_new 的值(1，2，3，1，2，3，1，2，3，1，2，3)

3.5.3.10　EPOINTJ 关节位置点和附加轴关节位置点组合

数据用于返回由 12 个实数值组成的扩展点

格式：EPOINTJ(j1，j2，j3，j4，j5，j6，ea1，ea2，ea3，ea4，ea5，ea6);

输入：各个轴位置 j1,j2,j3,j4,j5,j6（LREAL 类型)，外部附加轴关节位置 ea1,ea2,ea3,,ea4,ea5,ea6（LREAL 类型）

示例:

示例变量:

EPOINTJ　pos_wz　　//定义一个 POINTC 类型的变量

示例程序:

pos_wz = EPOINTJ(30，40，-90，90，90，0，7，8，9，4，5，6);　//给变量 pos_wz 赋值

程序给变量 pos_wz 赋值

3.5.3.11　EPOINTJOVR 覆盖关节角度和附加轴关节位置

覆盖原有数据

格式：EPOINTJOVR(pt，j1，j2，j3，j4，j5，j6，ea1，ea2，ea3，ea4，ea5，ea6);

输入：pt（EPOINTJ 类型），各个轴位置 j1,j2,j3,j4,j5,j6（LREAL 类型)，外部附加轴关节位置 ea1,ea2,ea3,,ea4,ea5,ea6（LREAL 类型）

示例:

示例变量:

EPOINTJ pos_fc //定义一个 EPOINTJ 类型的变量

EPOINTJ pos_new //定义一个 EPOINTJ 类型的变量

示例程序:

pos_fc =EPOINTJ(1，2，3，4，5，6，0，0，0，0，0，0); //给变量 pos_fc 赋值

pos_new =EPOINTJOVR(pos_fc，，，，，，，1，2，3，4，5，6); //给变量 pos_new 赋值

变量 pos_new 的值(1，2，3，4，5，6，1，2，3，4，5，6)

3.5.3.12 AJEPOINTC (笛卡尔空间位姿,附加轴关节位置点)

返回由笛卡尔点和存储外部附加轴关节位置的点组成的扩展点。

示例:

pos_cart := POINTC(100, 200, 300, 0, 180, 0) ;

eaux := POINTJ(1, 2, 3, 4, 5, 6) ;

Point := AJEPOINTC(pos_cart, eaux) ;

3.5.3.13 AJEPOINTJ (笛卡尔空间位姿,附加轴关节位置点)

返回由关节点和存储外部辅助轴关节位置的点组成的扩展点。

示例:

pos_joint := POINTJ(10, 20, 30, 1, 2, 3) ;

eaux := POINTJ(4, 5, 6, 7, 8, 9) ;

Point := AJEPOINTJ(pos_joint, eaux) ;

3.5.3.14 POSC 相对参考坐标系下的位姿

返回机器人当前实际的工具中心位置，返回数据类型为 POINTC

格式：POSC(ref);

输入：wobj 类型的工件坐标系

示例:

示例变量:

POINTC pos_ww //定义一个 pointc 类型的变量

示例程序:

pos_ww :=POSC(wobj0); //给变量 pos_ww 赋值

变量 pos_ww 获取当前坐标系下机器人的实际位姿

3.5.3.15 POSJ 获取关节位置

返回机器人当前实际的关节位置，返回数据类型为 POINTJ

格式：POSJ(参考坐标系);

输入：wobj 类型的工件坐标系

示例:

示例变量:

POINTJ pos_jj //定义一个 pointJ 类型的变量

示例程序:

pos_jj :=POSJ( );　　//给变量 pos_jk 赋值

变量 pos_jj 获取当前坐标系下机器人的关节位置

3.5.3.16　POSITION 提取笛卡尔空间点位置

从笛卡尔点提取位置向量（x，y，z）

格式：POSITION(pt);

输入：pointc 类型的位姿数据

示例:

示例变量:

POINTC　pos_ww　　//定义一个 pointc 类型的变量

VECT3　get_xyz　　//定义一个 vect3 类型的变量

示例程序:

pos_ww :=POSC(wobj0);　　//变量 pos_ww 获取当前坐标系下机器人的实际位姿

get_xyz:=POSITON(pos_ww);　//获取 pos_ww 的 xyz 的位置数据

变量 get_xyz 获取变量 pos_ww 中的 xyz 的数据

3.5.3.17　ROTATION 提取笛卡尔空间点位姿

从笛卡尔点提取位置旋转矢量（rotX，rotY，rotZ）

格式：ROTATION(pt);

输入：pointc 类型的位姿数据

示例:

示例变量:

POINTC　pos_ww　　//定义一个 pointc 类型的变量

VECT3　get_xyz　　//定义一个 vect3 类型的变量

示例程序:

pos_ww :=POSC(wobj0);　　//变量 pos_ww 获取当前坐标系下机器人的实际位姿

get_xyz :=ROTATION(pos_ww);　//获取 pos_ww 的 xyz 的位姿数据

变量 get_xyz 获取变量 pos_ww 中的 a,b,c 的数据(注意 POINTC 数据类型中 a 对应 rotZ, b 对应 rotY, c 对应 rotX)

3.5.3.18　MODULE 向量求模

返回向量的长度值

格式：MODULE(vect);

输入：vect 类型的向量数据

示例:

示例变量:

VECT3　v_a　　//定义一个 vect3 类型的变量

LREAL　get_len　　//定义一个 lreal 类型的变量

示例程序:

v_a := VECT3(100，100，0);　　//对变量 v_a 进行赋值

get_len :=MODULE(v_a);　　//返回向量 v_a 的长度

返回 get_len 的值为 141.421

3.5.3.19 DOT 向量点乘

两个向量点乘得到的是对应元素乘积的和，是一个标量，没有方向

格式：DOT(vect，vect);

输入：vect 类型的向量数据

示例:

示例变量:

```
VECT3   v_a       //定义一个 vect3 类型的变量
VECT3   v_b       //定义一个 vect3 类型的变量
VECT3   get_v     //定义一个 vect3 类型的变量
```

示例程序:

```
 v_a := VECT3(100，100，0);     //对变量 v_a 进行赋值
v_b :=VECT3(-100，0，0);     //对变量 v_b 进行赋值
get_v := DOT(v_a，v_b);   //对变量 v_a 和 v_b 进行向量点乘
```

由点乘公式 V1(x1, y1, z1)·V2(x2, y2, z2) = x1*x2 + y1*y2 + z1*z2 得到 get_v 的值

3.5.3.20 CROSS 向量叉乘

返回两个向量的矢量积

格式：CROSS(vect，vect);

输入：vect 类型的向量数据

示例:

示例变量:

```
VECT3   v_a          //定义一个 vect3 类型的变量
VECT3   v_b          //定义一个 vect3 类型的变量
VECT3   get_cr       //定义一个 vect3 类型的变量
```

示例程序:

```
v_a := VECT3(100，100，0);     //对变量 v_a 进行赋值
v_b :=VECT3(-100，0，0);     //对变量 v_b 进行赋值
get_v := CROSS(v_a，v_b);     //对变量 v_a 和 v_b 进行向量叉乘
```

由叉乘公式，设 a=(X1,Y1,Z1)，b=(X2,Y2,Z2)，则 a×b=（Y1*Z2-Y2*Z1，Z1*X2-Z2*X1，X1*Y2-X2*Y1)

3.5.3.21 NORMALIZE 向量归一化

处理各分量的模长，即按比例缩短，且单位长度方向不变

格式：NORMALIZE(vect);

输入：vect 类型的向量数据

示例:

示例变量:

```
VECT3   v_a          //定义一个 vect3 类型的变量
VECT3   get_v        //定义一个 vect3 类型的变量
```

示例程序:

```
! Main program
v_a := VECT3(100，100，0);     //对变量 v_a 进行赋值
```

get_v := NORMALIZE(v_a);　　//对变量 v_a 进行向量归一

向量归一化后 get_v 的值为（0.707，0.707，0）

3.5.3.22　SCALE 标量积

返回标量积，即数*向量

格式：SCALE(k，vect);

输入：LREAL 类型，vect 类型的向量数据

示例:

示例变量:

VECT3　v_a　　//定义一个 vect3 类型的变量

VECT3　get_v　　//定义一个 vect3 类型的变量

示例程序:

v_a := VECT3(10，20，30);　　//对变量 v_a 进行赋值

get_v := SCALE(3，v_a);　　// 将向量各数值乘以 3

返回 get_v 的标量积为（30，60，90）

3.5.3.23　TOPOS 将局部坐标系下点转化至世界坐标系

返回从 refsys 坐标系的一个点转化至世界坐标系的位姿信息。

格式：TOPOS(POS，ref);

输入：POINTC 类型 POS 位置，ref 参考坐标系

示例:

示例变量:

wobj := REFSYS(wobj0, 100, 200, 50, 0, 0, 0) ;

pointA := POINTC(200, 300, 100, 0, 0, 0) ;

pointB := TOPOS(point A, wobj) ;

变量 pointB 的值将为(300, 500, 150, 0, 0, 0).

3.5.3.24　TOLOCALPOS 将世界坐标系下点转化至局部坐标系

返回从世界坐标系的一个点转化至局部坐标系的位姿信息。

格式：TOLOCALPOS (POS，ref);

输入：POINTC 类型 POS 位置，ref 参考坐标系

示例:

示例变量:

wobj := REFSYS(wobj0, 100, 200, 50, 0, 0, 0) ;

pointA := POINTC(300, 500, 150, 0, 0, 0) ;

pointB := TOLOCALPOS(pointA, wobj) ;

变量 pointB 的值将为(200, 300, 100, 0, 0, 0).

3.5.3.25　CALCPOSJ　转化为关节点格式

将参考坐标系下的笛卡尔点转化为关节点,返回值类型 POINTJ

格式：CALCPOSJ(point，refsys);

输入：POINTC 类型 POS 位置，ref 参考坐标系

示例:

示例变量:

```
POINTC    pos_cc              //定义一个 pointc 类型的变量
POINTJ    pos_jj              //定义一个 POINTC 类型的变量
```

示例程序:

```
pos_cc :=Posc(wobj0) ;     //变量点 pos_cc 获取当前坐标下机器人位姿信息
pos_jj :=CALCPOSJ(pos_cc , wobj0);  //将点 pos_cc 转化为关节点,赋值给 pos_jj
```

将笛卡尔点 pos_cc 转化为关节形式的点

3.5.3.26 CALCPOSC 转化为笛卡尔点格式

将输入的关节点位转换为指定参考坐标系下，指定工具末端的笛卡尔点位。如果省略参考系参数，则使用世界参考系，返回类型时 POINTC。

格式：POINTC := CALCPOSC(POINTJ point，TOOL tool， REFSYS refsys);

输入：POINTJ 类型 point 为待转换的关节位置，TOOL tool 用于指定工具末端，REFSYS refsys 是目标点位所处的参考坐标系

示例:

示例变量:

```
POINTC    pos_cc              //定义一个 POINTC 类型的变量
POINTJ    pos_jj              //定义一个 POINTJ 类型的变量
```

示例程序:

```
pos_jj := POSJ() ;     //变量点 pos_jj 获取当前坐标下机器人位姿信息
pos_cc := CALCPOSC(pos_jj , tool0，wobj0) ;  //将点 pos_jj 转化为关节点,赋值给 pos_cc
```

将关节点 pos_jj 转化为笛卡尔形式的点

3.5.3.27 ISPOINTVALID 判断某个点是否有效

返回一个布尔值，判断某个点是否为有效点

格式：ISPOINTVALID(point);

输入：POINTC/POINTJ 类型 POS 位置

示例:

示例变量:

```
POINTC     pos_cr    //定义一个 POINTC 类型的变量
BOOL       pos_bool  //定义一个 BOOL 类型的变量
```

示例程序:

```
pos_cr :=Posc(wobj0) ;        //变量点 pos_cr 获取当前机器人位姿信息
pos_jj :=CALCPOSJ(pos_cc , wobj0);   //将点 pos_cc 转化为关节点,赋值给 pos_jj
pos_bool :=ISPOINTVALID(pos_jj   );    //判断 pos_cr 点是否有效,返回 bool 值
IF   ISPOINTVALID (pos_bool )   THEN   /*判断 pos_jj 点是否是有效点，有效点则运行到有效点，
MJOINT(pos_jj , v100, fine, tool0) ;    //无效则弹出对话框“点不正确”*/
ELSE
MESSAGE (“Point is not OK”) ;
END_IF ;
```

判断点 pos_jj 是否是有效点,并返回 bool 值，返回值为 true 时运行到 pos_jj 的点，返回值为 flase 弹出对话框。

3.5.3.28 OFFSET 笛卡尔空间点沿 X/Y/Z 方向偏移

将笛卡尔空间的点沿 X/Y/Z 方向分别添加偏移量，单位 mm

格式：OFFSET(pt，X，Y，Z);

输入：pt（POINTC 类型位置），X,Y,Z（x/y/z 方向偏移的距离）

示例:

示例变量:

```
POINTC   pos_cr          //定义一个 POINTC 类型的变量
POINTC   pos_new         //定义一个 POINTC 类型的变量
```

示例程序:

```
pos_cr :=Posc(wobj0) ;    //变量点 pos_cc 获取当前坐标系下机器人位姿信息
pos_new := OFFSET(pos_cr，10，20，-30 );  //对点 pos_cc 点进行偏移,赋值给 pos_new
```

点 pos_cc 沿着 x 方向偏移 10mm，沿着 y 方向偏移 20mm，沿着 z 方向偏移-30mm

3.5.3.29 DISTANCE 空间中两个笛卡尔点之间的距离

计算笛卡尔空间两点之间的距离，该值始终为正值，单位 mm

格式：DISTANCE(pointA，pointB);

输入：pointA（POINTC 类型位置），pointB（POINTC 类型位置）

示例:

示例变量:

```
POINTC   pos_ww          //定义一个 POINTC 类型的变量
POINTC   pos_jj          //定义一个 POINTC 类型的变量
UDINT    dist_ww_jj      //定义一个 UDINT 类型的变量
```

示例程序:

```
pos_ww := POINTC(100，200，-300，12，13，14);   //变量 pos_ww 获取一个 pointc 类型的点
pos_jj := POINTC(200，200，-300，12，13，14);    //变量 pos_jj 获取一个 pointc 类型的点
dist_ww_jj := DISTANCE(pos_ww，pos_jj);    //计算 pos_ww 和 pos_jj 两点之间的距离
```

计算点 pos_ww 和点 pos_jj 之间的空间距离，并存储在变量 dist_ww_jj 中

3.5.3.30 OFFSETOOL 笛卡尔空间点沿 X/Y/Z/A/B/C 方向偏移

沿笛卡尔空间点 X/Y/Z/A/B/C 方向分别添加偏移量

格式：OFFSETTOOL(pt，x，y，z，a，b，c);

输入：pt（POINTC 类型位置），X,Y,Z（x/y/z 方向偏移的距离，单位为 mm），a,b,c（a/b/c 方向旋转的角度，单位为 deg）

示例:

示例变量:

```
POINTC   pos_cr          //定义一个 POINTC 类型的变量
POINTC   pos_new         //定义一个 POINTC 类型的变量
```

示例程序:

```
pos_cr :=POSC(wobj0);     //当前工具（示例选择 tool1)下机器人位姿信息
```

MJOINT (pos_cr, v100, fine, tool1) ;　　//运行到 tool1 坐标系下的 pos_cr 点位

pos_new := OFFSETTOOL(pos_cr，0，10，0，45，0，0); //对点 pos_cr 进行偏移和旋转，赋值给 pos_new

MJOINT (pos_new, v100, fine, tool1) ;　　//运行到 pos_new 点

点 pos_cr 沿着 y 偏移 10mm，绕着 a 方向旋转 45 度

3.5.3.31　CHECKTARGET 点位检验

检查指定点位对本机器人而言是否可用和可到达。

格式：CHECKTARGET (point, tool, reference system);

输入：point（待检查机器人的位姿），tool（到达该位姿时使用的工具）, reference system（运动过程中使用的坐标系）

输出：BOOL 类型变量，point 点是否可达。

示例:

tool := STOOL(0, 0, 100, 0, 45, 0) ;

point := POINTC(300, 0, 400, 180, -45, 180) ;

reachable := CHECKTARGET(point, tool0, wobj0) ;

pointJ := POINTJ(180, 0, 0, 0, 0, 0);

reachableJ := CHECKTARGET(pointJ, tool0, wobj0);

在例子中，以上 reachable 为真，因为点在机器人的工作极限内，reachableJ 为假，因为点在机器人的关节极限外。

3.5.3.32　FRAME 建立新工件坐标系

通过使用 uframe、oframe、movable、basename 创建一个新工件坐标系。对于参数的释义可参考 REFSYS 中数据的解释

格式：FRAME(uframe, oframe, movable, basename);

输入：uframe（工件坐标系），oframe（新工件坐标系），moveable（建立的新工件坐标系是否跟随 uframe 随动），basename（module 名）

输出：创建一个新的工件坐标系。

示例:

示例变量:

POINTC oframe

POINTC uframes

示例程序:

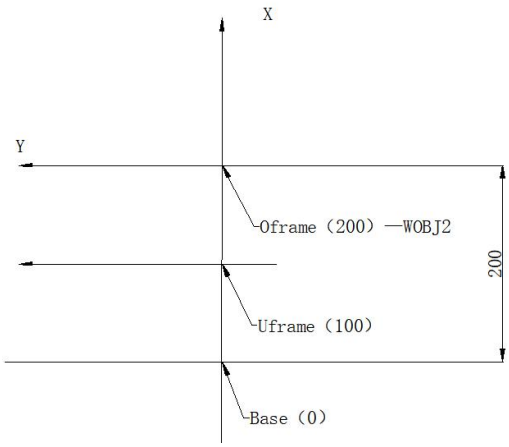
Uframe := pointc(1200,0,0884,-180,0,180);//将该点定义为 Uframe 坐标系

Mjont(uframe,v500,fine,tool0,wobj0);

Oframe := pointc(1200,0,0884,-180,0,180);//将该点定义为 Oframe 坐标系

Wobj2 := frame(uframe,oframe,false,);　将基于 Uframe 坐标系下 Oframe 的坐标系赋给 Wobj2

MLIN (Uframe，v500，fine，tool0，Wobj2)；//机器人将基于 Wobj2 运行至 Uframe 点



3.5.3.33 GETUFRAME 获取 UFRAME 坐标系

返回参考坐标系下的 uframe 坐标系部分。 有关 uframe 的说明，请参见 REFSYS 数据说明

格式：GETUFRAME(wobj);

输入：工件坐标系 wobj

输出：返回 wobj 中的 uframe 坐标系。

示例:

示例变量:

POINTC oframe

POINTC uframes

POINTC uframesTemp

示例程序:

```
Uframe := pointc(1200,0,0884,-180,0,180);//将该点定义为 Uframe 坐标系
Mjont(uframe,v500,fine,tool0,wobj0);
Oframe := pointc(1200,0,0884,-180,0,180);//将该点定义为 Oframe 坐标系
Wobj2 := frame(uframe,oframe,false, );  将基于 Uframe 坐标系下 Oframe 的坐标系赋给 Wobj2
MLIN (Uframe，v500，fine，tool0，Wobj2)；//机器人将基于 Wobj2 运行至 Uframe 点
uframesTemp:= GETUFRAME(Wobj2)；//将构成 Wobj2 中的 Uframe 坐标系赋给 uframesTemp，此时 uframesTemp 等同于 Uframe
```

3.5.3.34 GETOFRAME 获取 OFRAME 坐标系

返回参考坐标系下的 uframe 坐标系部分。 有关 oframe 的说明，请参见 REFSYS 数据说明。

格式：GETOFRAME(wobj);

输入：工件坐标系 wobj

输出：返回 wobj 中的 oframe 坐标系。

示例:

示例变量:

POINTC oframe

POINTC uframes

POINTC OframesTemp

示例程序:

Uframe := pointc(1200,0,0884,-180,0,180);//将该点定义为 Uframe 坐标系

Mjont(uframe,v500,fine,tool0,wobj0);

Oframe := pointc(1200,0,0884,-180,0,180);//将该点定义为 Oframe 坐标系

Wobj2 := frame(uframe,oframe,false,); 将基于 Uframe 坐标系下 Oframe 的坐标系赋给 Wobj2

MLIN (Uframe，v500，fine，tool0，Wobj2)；//机器人将基于 Wobj2 运行至 Uframe 点

OframesTemp:= GETOFRAME(Wobj2)；//将构成 Wobj2 中的 Oframe 坐标系赋给 OframesTemp，此时 OframesTemp 等同于 Oframe

3.5.3.35 FRAMEDISPL 获取相对差坐标系

生成一个可用于将位置从点 pref 转换为点 pnew 的参考系统。pref 和 pnew 都必须被引用到同一参考系统。如果 pref 和 pnew 未参考世界参考系统，则可以使用可选的参数 refsys。使用返回的参考系统，可以转换新位置的执行。其新坐标系相对于原坐标系沿 pref—>pnew 的偏移。

格式：FRAMEDISPL(pref,pnew,refsys);

输入：点位 1，点位 2，参考坐标系

输出：返回标定的坐标系。

示例:

示例变量:

POINTC p1

POINTC p2

示例程序:

P1 := pointc(1261,0,1425,180,8.1,180);//

P2 := pointc(1564,200,1425,180,8.1,180);//

MLIN(P1,v500,fine,tool0,wobj0);//

MLIN(P2,v500,fine,tool0,wobj0);//

Wobj1 := FRAMEDISPL(P1,P2);基于 P1 与 P2 点建立新的坐标系 wobj1

MLIN(P2,v500,fine,tool0,wobj0);//机器人基于新坐标系 wobj1 运行至原基于 wobj0 的位置。

3.5.3.36 REFSYS 创建新参考坐标系

通过父坐标系以及 6 个 LREAL 型的数据创建一个新的坐标系

格式：REFSYS(parent reference,x,y,z,a,b,c);

输入：父坐标系，x,y,z,a,b,c 方向的偏移

输出：返回一个新坐标系。

注意：从 xpl 版本 2.0.0 起，父参考坐标系必须被省略，否则运行会有报错。

Wobj := REFSYS(，100，0，0，0，0，0);

注意：从 xpl 版本 2.1.3 起，为了与旧版本程序相兼容，父参考坐标系被引入，父参考系被用来设置 Uframe。

Wobj := REFSYS(wobj2，100，0，0，0，0，0);

示例:

示例变量:

POINTC oframe

POINTC uframes

POINTC OframesTemp

示例程序:

Uframe := pointc(1200,0,0884,-180,0,180);//将该点定义为 Uframe 坐标系

Mjont(uframe,v500,fine,tool0,wobj0);

Oframe := pointc(1200,0,0884,-180,0,180);//将该点定义为 Oframe 坐标系

Wobj2 := frame(uframe,oframe,false,); 将基于 Uframe 坐标系下 Oframe 的坐标系赋给 Wobj2

MLIN (Uframe，v500，fine，tool0，Wobj2)；//机器人将基于 Wobj2 运行至 Uframe 点

OframesTemp:= GETOFRAME(Wobj2)；//将构成 Wobj2 中的 Oframe 坐标系赋给 OframesTemp，此时 OframesTemp 等同于 Oframe

3.5.3.37 VREFSYS 创建新参考坐标系

通过父坐标系以及 2 个向量创建一个新的坐标系，其中第一个代表位置，第二个代表旋转向量

格式：REFSYS(parent reference,pos,rot);

输入：父坐标系，x,y,z,a,b,c 方向的旋转偏移

输出：返回一个新坐标系。

Wobj := VREFSYS(wobj1,pos,rot);

示例:

示例变量:

VECT3 vectpos

VECT3 vectrot

POINTC pc1

示例程序:

Pc1 := pointc(1200,0,1884,0,0,0);//

MLIN(Pc1,v500,fine,tool0,wobj0);

Vectpos := VECT3(10,10,10);此为两个坐标系中关于位置的偏移量

Vectrot := VECT3(0,0,90);此为两个坐标系中关于姿态的旋转量（绕 Z 轴旋转）

Wobj1 := VREFSYS(wobj0,vectpos,vectrot);基于 vectpos&vectrot 确定两个坐标系之间的转换关系

MLIN(Pc1,v500,fine,tool0,wobj1);

此时机器人姿态将为：在原坐标系下的位置（1300，10，1894）下，绕 Z 轴旋转 90°

3.5.3.38 REFSYS3P 三点创建新参考坐标系

通过父坐标系以及三点创建一个新的坐标系，其中第一个代表新坐标系下的原点位置，第二个代表 X 方向，第三个代表 Y 方向

格式：REFSYS(parent reference,origin pos,x,y);

输入：父坐标系，新坐标系下原点坐标，x 方向的位移值，y 方向的位移值

输出：返回一个新坐标系。

Wobj := REFSYS3P(wobj0,pos,X,Y);

示例:

示例变量:

POINTC P0

POINTC P1

POINTC P2

示例程序:

Pc1 := pointc(1200,0,1884,0,0,0);//

MLIN(Pc1,v500,fine,tool0,wobj0);

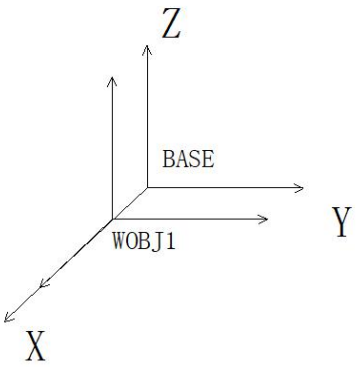
P0 := POINTC(100,0,0,0,0,0);此为新坐标系原点位置

P1 := POINTC(200,0,0,0,0,0);此为新坐标系 X 方向位移

P2 := POINTC(0,100,0,0,0,0);此为新坐标系 Y 方向位移

Wobj1 := REFSYS3P(wobj0,P0,P1,P2);基于原点和两个轴的方向确定新坐标系,如图所示

MLIN(POINTC(1200,0,1884,0,0,0),v500,fine,tool0,wobj1);



3.5.3.39 CWOBJ 当前坐标系

返回当前激活的坐标系

输入：参考坐标系

输出：返回当前激活的坐标系。

Wobj1 := CWOBJ();

示例:

示例变量:

POINTC P0

POINTC P1

POINTC P2

示例程序:

Pc1 := pointc(1200,0,1884,0,0,0);//

P0 := POINTC(100,0,0,0,0,0);此为新坐标系原点位置

P1 := POINTC(200,0,0,0,0,0);此为新坐标系 X 方向位移

P2 := POINTC(0,100,0,0,0,0);此为新坐标系 Y 方向位移

Wobj1 := REFSYS3P(wobj0,P0,P1,P2);//基于原点和两个轴的方向确定新坐标系

MLIN(Pc1,v500,fine,tool0,wobj0);

Wobj1 := CWOBJ();

MLIN(Pc1,v500,fine,tool0,wobj1);//此时，机器人不运动，因为 WOBJ1 等于 WOBJ0

3.5.3.40 RWOBJ 当前坐标系

返回当前激活的坐标系

输入: 参考坐标系

输出: 返回当前激活的坐标系。

Wobj1 := RWOBJ();

示例:

示例变量:

POINTC P0

POINTC P1

POINTC P2

示例程序:

Pc1 := pointc(1200,0,1884,0,0,0);//

P0 := POINTC(100,0,0,0,0,0);此为新坐标系原点位置

P1 := POINTC(200,0,0,0,0,0);此为新坐标系 X 方向位移

P2 := POINTC(0,100,0,0,0,0);此为新坐标系 Y 方向位移

Wobj1 := REFSYS3P(wobj0,P0,P1,P2);//基于原点和两个轴的方向确定新坐标系,如图所示

MLIN(Pc1,v500,fine,tool0,wobj1);

Wobj1 := RWOBJ();

MLIN(Pc1,v500,fine,tool0,wobj1);//此时，机器人沿 X 方向，向后移动 100mm

3.5.3.41 GETWOBJ（基/夹具标志，机器人） 获取机器人标识坐标系

返回特定机器人的基/夹具参考系。如果标志为 **FALSE** 则返回机器人参考系，否则返回夹具参考系。基参考系是固定参考系。夹具参考系可以是移动的参考系。例如，传送机具有固定的参考系描述传送机在系统中的具体位置。

传送机的夹具参考系则是描述与输送带一起移动的参考系。

输入: 基/夹具标志，机器人

输出: 返回机器人标识的坐标系。

示例:

示例变量:

ROBOT cvyAxisGroup

POINTC P0

POINTC P1

POINTC P2

示例程序:

Pc1 := pointc(1200,0,1884,0,0,0);//

P0 := POINTC(100,0,0,0,0,0);此为新坐标系原点位置

P1 := POINTC(200,0,0,0,0,0);此为新坐标系 X 方向位移

P2 := POINTC(0,100,0,0,0,0);此为新坐标系 Y 方向位移

Wobj1 := REFSYS3P(wobj0,P0,P1,P2);//基于原点和两个轴的方向确定新坐标系,如图所示

MLIN(Pc1,v500,fine,tool0,wobj1);

cvyAxisGroup := ROBOT("linearbelt");//获取系统中名字为"conveyor"的轴组的数据

Wobj1 := GETWOBJ(false,cvyAxisGroup);//获取基于传送带的固定坐标系

MLIN(Pc1,v500,fine,tool0,wobj1);此时机器人将沿新坐标系 WOBJ1 运动

3.5.3.42 GETTRKWOBJ（参考系统，跟踪数据） 返回跟踪坐标系

返回特定参考系的跟踪参数并返回可移动参考系。

输入: 参考坐标系，跟踪数据变量

输出: 返回机器人跟踪的坐标系。

示例:

示例变量:

ROBOT cvyAxisGroup

Tracking cvyTrackdata

POINTC P0

POINTC P1

POINTC P2

示例程序:

Pc1 := pointc(1200,0,1884,0,0,0);//

P0 := POINTC(100,0,0,0,0,0);此为新坐标系原点位置

P1 := POINTC(200,0,0,0,0,0);此为新坐标系 X 方向位移

P2 := POINTC(0,100,0,0,0,0);此为新坐标系 Y 方向位移

Wobj1 := REFSYS3P(wobj0,P0,P1,P2);//基于原点和两个轴的方向确定新坐标系,如图所示

MLIN(Pc1,v500,fine,tool0,wobj1);

cvyAxisGroup := ROBOT("linearbelt");//获取系统中名字为"conveyor"的轴组的数据

Wobj1 := GETWOBJ(true,cvyAxisGroup);//获取基于夹具坐标系

Wobj2 := GETTRKWOBJ(Wobj1,cvyTrackdata);//基于传送带上的工件建立坐标系，用以跟踪工件

3.5.3.43 SETROBOTWOBJ（robot，参考系统，跟踪数据） 设置机器人基坐标系

设置并返回特定机器人的基础 refsys。

输入: ROBOT，参考坐标系，跟踪数据变量

输出: 返回机器人新设置的坐标系。

示例:

示例变量:

ROBOT　AxisGroup

POINTC Pc1

POINTC P0

示例程序:

P0 := pointc(1200,0,1884,0,0,0);//

Pc1 := pointc(0,0,200,0,0,0);//

MLIN(P0，v500，fine，tool0)；

AxisGroup := ROBOT(“erag”);//获取系统中名字为“erag”的轴组的数据

Wobj1 := SETROBOTWOBJ(AxisGroup,“WOBJ0”,Pc1);//获取基于 Pc1 的坐标系

MLIN(P0，v500，fine，tool0，wobj1)；//此时机器人将沿 Z 方向，向上移动 200（原因：新建立的 wobj1 较 wobj0 上移 200mm）

3.5.4　设置复杂数据的函数

3.5.4.1　CTOOL 当前激活坐标系

返回当前活动工具的值。该功能可用于设置 TOOL 的数据类型。

格式：CTOOL();

输出：TOOL 类型的数据，值为当前激活的坐标系

示例:

tool := CTOOL()

在这个例子中，我们用当前的工具参数设置工具。

3.5.4.2　ROBOT 机器人函数

此类数据用于与系统中定义的轴组进行交互。例如，可以设置轴组的参考系。要设置此类型的数据，必须使用函数 ROBOT，并使用字符串型轴组名作为其参数。

初始化后的默认值是无效的 ROBOT。

格式：ROBOT(***anxesgroup name***);

示例:

示例变量:

ROBOT conveyor;

REFSYS cyvwobj;

POINTC cyv;

示例程序:

conveyor := ROBOT(“conveyor”)；　　//将字符串轴组名赋值给定义的轴

cyvOrigin := POINTC(550, 0, 380, 180, 0, 180);　　//为变量赋值笛卡尔坐标数据

cyvwobj := SETROBOTWOBJ(conveyor, wobj0, cyv);//设置机器人坐标系并返回特定机器人基础 refsys

此例中，设置了 conveyor 参考系。进行此设置是为了在程序中指定 conveyor 的参考系，并在例如跟踪应用中使用。

3.5.4.3 TRACKING 跟踪

TRACKING 数据类型用于存储跟踪应用程序的数据。设置此类数据必须使用 TRACKING 函数。有关此参数的说明，请参阅跟踪应用的说明文档。初始化后的默认值是无效的 TRACKING。

格式：TRACKING(***maxspace, startspace, maxspe, maxacc, type, par1***);

输入：最大行程，起始速度，最大速度，最大加速度，类型，标准；

输出：null；

示例：

示例变量：

```
TRACKING cvy;
POINTC target;
REFSYS wobj;
REFSYS rsPhoto;
```

示例程序：

```
target := POINTC(600, 0, 500, 0, 180, 0);   //给 pointc 变量 target 赋值
cvy := TRACKING(1000, 700, 100, 1000, Linear, 0.1);  //为 tracking 指令添加应用信息
wobj := GETTRKWOBJ(rsPhoto, cvy);  //设置特定参考系的跟踪参数并返回可移动参考系
MLIN(target, v500, fine, tool1, wobj);      //机器人运动
```

此例显示如何为跟踪应用程序设置跟踪参数。

3.5.4.4 SSPEED 速度设置函数

此函数可用于设置 SPEED 数据类型。切向速度以 mm/s 表示，姿态速度以 degree/s 表示。

格式：SSPEED(tangential speed,orientation speed);

输入：切向速度，定向速度;

输出：速度设置;

示例：

示例变量：

```
DINT tanSpeed;
DINT oriSpeed;
SPEED spe;
```

示例程序：

```
tanSpeed := 500;          //将 500 赋值给 tanSpeed 变量
oriSpeed := 5;          //将 5 赋值给 oriSpeed 变量
spe := SSPEED(tanSpeed, oriSpeed);  //设置切向速度和定向速度
MLIN (*, spe, fine, tool0);      //机器人运动
```

此例中，我们将 spe 设定为切向速度为 500mm/s，姿态速度为 5degree/s。

3.5.4.5 SSPEED_PERC 速度数据类型设置

此功能可用于设置速度数据类型。

切向速度和定向速度以最大切向值的百分比表示。

格式：SSPEED_PERC(percentage tangential speed,percentage orientation speed);

输入：切向速度百分比，定向速度百分比;

输出：速度设置;

示例:

示例变量:

```
DINT pertan = 10;
DINT perori = 50;
SPEED spe1;
SPEED spe2;
```

示例程序:

```
spe1 := SSPEED_PERC(pertan, perori);  //设置切向速度和定向速度
MLIN (*, spe1, fine, tool0);        //机器人运动
pertan := 100;              //将 100 赋值给 pertan 变量
perori := 10;               //将 10 赋值给 perori 变量
spe2 := SSPEED_PERC(pertan, perori);  //设置切向速度和定向速度
MLIN (*, spe2, fine, tool0);        //机器人运动
```

此例中，最大切向速度为 2000mm/s，最大姿态速度为 360degree/s，我们将 spe1 设定为最大切向速度的 10%即为 200mm/s, 最大姿态速度的 50%即为 180degree/s; spe2 设定为切向速度 2000mm/s, 姿态速度 36degree/s。

3.5.4.6 SZONE ZONE 设置函数

此功能可用于设置 ZONE 数据类型。

线性距离以 mm 表示，重整定姿态角距离以 degree 表示。

格式：SZONE(linear distance,reorientation angular distance);

输入：线性距离，重新定姿态角距离;

输出：新的 zone 类型;

示例:

示例变量:

```
ZONE zone1;
POINTC pos1;
POINTC pos2;
```

示例程序:

```
zone1 := SZONE(100, 5);        //设置 ZONE 类型数据
pos1 := POINTC(600, 0, 300, 0, 180, 0);  //pos1 位置点赋值
pos2 := POINTC(500, 0, 200, 0, 180, 0);  //pos2 位置点赋值
MLIN (pos1, v250, zone1, tool0);      //机器人运动到 pos1 点
MLIN (pos2, v250, fine, tool0);      //机器人运动到 pos2 点
```

此例中，机器人在距离 pos1 点 100 mm 时开始向 pos2 点移动。

3.5.4.7 STOOL 工具坐标系设置函数

此函数用来设置工具坐标系 TOOL 的数据类型，强制修改当前程序中应用的 TOOL 数据。

格式：STOOL(x, y, z, a, b, c);

输入：工具坐标系数值;

输出：新工具坐标系;

示例:

示例变量:

```
TOOL t1;
POINTC Pos;
```

示例程序:

```
t1 := STOOL(0, 0, 0, 0, 0, 0);      //设置 TOOL 类型数据
Pos := POINTC(600, 0, 300, 0, 180, 0);    //Pos 位置点赋值
MLIN (Pos, v250, fine, t1);      //机器人运动到 Pos 点
t1 := STOOL(100, 0, 100, 0, 0, 0);    //设置 TOOL 类型数据
MLIN (Pos, v250, fine, t1);      //机器人运动到 Pos 点
```

此例中，检验在使用相同目标点和两个不同的工具坐标的情况下，机器人位置的不同 。

3.5.5 字符串函数

3.5.5.1 BOOL_TO_STRING (BOOL val) 布尔类型转字符串

转换 BOOL 为字符串。

示例:

```
BOOL bool_val
STRING strTrue
STRING strFalse
…
bool_val := true ;
strTrue := BOOL_TO_STRING(bool_val) ;
bool_val:= false ;
strFalse := BOOL_TO_STRING(bool_val) ;
```

执行后 strTrue 将为“true”，strFalse 将为“false”。

3.5.5.2 DINT_TO_STRING (DINT val) 双精度整型转字符串

转换 DINT 为字符串。

示例:

```
DINT dint_val
…
dint_val := -120 ;
strVal := DINT_TO_STRING(dint_val) ;
```

执行后 strVal 为”-120”。

3.5.5.3 LREAL_TO_STRING (LREAL val) 长实数型转字符串

转换 LREAL 为字符串。

示例:

```
LREAL lreal_val
…
lreal_val := 3.14 ;
strVal := LREAL_TO_STRING(lreal_val) ;
```

执行后 strVal 为”3.140000”。

3.5.5.4 UDINT_TO_STRING (UDINT val) 无符号双精度整型转字符串

转换 UDINT 为字符串。

示例:

```
UDINT udint_val
…
udint_val := 456 ;
strVal := UDINT_TO_STRING(udint_val) ;
```

执行后 strVal 为”456”。

3.5.5.5 LEN (STRING str) 返回字符长度

返回字符串的长度。

示例:

```
strA := “value” ;
n := LEN(strA) ;
```

执行后 n 赋值为 5。

3.5.5.6 LEFT (STRING str, pos) 返回左侧位字符串

返回字符串 **str** 最左边 **pos** 位的字符组成的字符串。

示例:

```
strA := “Hello world” ;
strB := LEFT(strA, 5) ;
```

执行后的字符串 strB 将为“Hello”。

3.5.5.7 RIGHT (STRING str, pos) 返回右侧位字符串

返回字符串 **str** 最右边 **pos** 位的字符组成的字符串。

示例:

```
strA := “Hello world” ;
strB := RIGHT(strA, 5) ;
```

执行后的字符串 strB 将为“world”。

3.5.5.8 MID (STRING str, len, pos) 选取字符串

返回字符串 **str** 第 **len** 位到 **pos** 位的字符组成的字符串。

示例:

strA := “Time is over” ;

strB := MID(strA, 2, 6) ;

执行后的字符串 strB 将为“is”。

3.5.5.9 CONCAT (STRING str1, STRING str2) 组合字符串

返回一个由字符串 **str1** 和字符串 **str2** 组成的字符串。

示例:

strA := “air” ;

strB := “plane” ;

strC := CONCAT(strA, strB) ;

执行后的字符串 strC 将是“airplane”。

3.5.5.10 INSERT (STRING str1, STRING str2, pos) 插入字符串

返回在字符串 **str1** 第 **pos** 位插入字符串 **str2** 组成的字符串。

示例:

strA := “I go to office” ;

strB := “post ” ;

strC := INSERT(strA, strB, 8) ;

执行后的字符串 strC 将是”I go to post office”。

3.5.5.11 FIND (STRING str1, STRING str2) 字符串位置

返回字符串 **str2** 在字符串 **str1** 出现的位置。

示例:

strA := “policeman” ;

strB := “man” ;

n := FIND(strA, strB) ;

执行后 n 将是 7。

3.5.5.12 DELETE (STRING str, len, pos) 删除字符串

返回通过从位置 pos 开始从字符串 str 中删除 len 个字符而获得的字符串。

示例:

strA := “policeman” ;

strB := DELETE(strA, 3, 7) ;

执行后字符串 strB 的值将会是 “police”.

3.5.5.13 REPLACE (STRING str1, STRING str2, len, pos) 替换字符串

返回一个字符串，该字符串是将字符串 str1 的位置 pos 处的 len 个字符替换为字符串 str2 的结果。

示例:

strA := “grandmother” ;

strB := “fa” ;

strC := REPLACE(strA, strB, 2, 6) ;

执行后的字符串 strC 将是“granfather”。

3.5.5.14 STRING_TO_LREAL (STRING str) 字符串型转长实数型

转换 string 为 LREAL。

示例:

```
STRING strVal
LREAL lreal_val
…
strVal := “1.5” ;
lreal_val := STRING_TO_LREAL(strVal) ;
```

执行后 lreal_val 为 1.5。

3.5.5.15 STRING_TO_UDINT (STRING str) 字符串型转无符号双精度整型

转换 string 为 UDINT。

示例:

```
STRING strVal
UDINT udint_val
…
strVal := “124” ;
udint_val := STRING_TO_UDINT(strVal) ;
```

执行后 udint_val 为 124。

3.5.5.16 STRING_TO_DINT (STRING str) 字符串型转双精度整型

转换 string 为 DINT。

示例:

```
STRING strVal
DINT dint_val
…
strVal := “-78” ;
dint_val := STRING_TO_DINT(strVal) ;
```

执行后 dint_val 为-78。

3.5.5.17 STRING_TO_BOOL (STRING str) 字符串型转布尔型

转换 string 为 BOOl。

示例:

```
STRING strVal
BOOL bool_val
…
strVal := “true” ;
bool_val := STRING_TO_BOOL(strVal) ;
```

执行后 bool_val 为 true。

3.5.6 其他函数

3.5.6.1 RANDOM () 返回随机正整数

返回 0 到 32767 之间的随机正整数值。

示例:

```
val := RANDOM() ;
```

变量 val 可以具有 0 到 32767 之间的值，例如 val 是 27236。

3.5.6.2 R_AND (UDINT a, UDINT b) 二进制与

返回两个值的二进制与。

示例:

```
binA := 7 ;
binB := 5 ;
binC := R_AND(binA, binB) ;
```

变量 binC 为 5。

3.5.6.3 R_OR (UDINT a, UDINT b) 二进制或

返回两个值的二进制的或。

示例:

```
binA := 7 ;
binB := 5 ;
binC := R_OR(binA, binB) ;
```

变量 binC 为 7。

3.5.6.4 R_XOR (UDINT a, UDINT b) 二进制异或

返回两个整数值的二进制异或。

示例:

```
binA := 7 ;
binB := 5 ;
binC := R_XOR(binA, binB) ;
```

变量 binC 为 2。

3.5.6.5 R_NOT (UDINT) 二进制反转

返回整数值的二进制反转。

示例:

```
binA := 13 ;
binB := 255 ;
binC := R_AND(R_NOT(binA), binB) ;
```

变量 binC 为 242。

3.5.6.6 ABS_MOD (值，除数) 取正余数

返回值为被除数除以除数后的余数，如果值为负，则将除数添加到结果中以使其为正。

示例:

```
valA := ABS_MOD(42, 5) ;
```

valB := ABS_MOD(-42, 5) ;

变量 valA 为 2，valB 为 3。

3.5.6.7 CLOCKREAD (时间变量) 时间读取

返回 CLOCK 变量中包含的已用时间。 时间以秒表示。

示例:

```
CLOCKRESET(clk) ;
CLOCKSTART(clk) ;
…
CLOCKSTOP(clk) ;
MESSAGE (“Cycle time is %1”, CLOCKREAD(clk)) ;
```

在上面的示例中，显示的消息可能为“Cycle time is 3.234”。

3.5.6.8 RUNASPROGRAM() 判断当前是否在主程序

返回一个布尔值，它表示代码是否在主程序内。如果代码在主程序中，则返回 true，否则则返回 false。

格式： PROGRAM NAME ： = RUNASPROGRAM（）

输入：设定 test：=RUNASPROGRAM（）

输出：MESSAGE（“Run from main program”）

示例:

示例变量名:

```
Bool test //声明 test 变量
```

示例程序:

```
test:= RUNASPROGRAM();//判断当前执行的语句是否在 main 主程序中，如果是则返回
true，否则返回 false
IF test THEN
MESSAGE(“Run from main program”);
END_IF ;
```

此例中，通过 RUNASPROGRAM（）函数，输出“Run from main program”

注意：如果当前执行的语句不在 main 主程序中，则不会继续往下执行输出“Run from main program”

3.5.6.9 ISMODLOADED 判断是否加载特定模块

返回一个布尔值，它指示是否加载了具有特定名称的模块

格式： ISMODLOADEDMAXT(name[特定模块名称]);

输入：设定 ISMODLOADED（“user”）

输出：MESSAGE（“User module is not loaded“”）

示例:

示例程序:

```
IF NOT ISMODLOADED(“user”) THEN
MESSAGE(“User module is not loaded”);
END_IF ;
```

此例中，通过 ISMODLOADED（）函数判断是否有 User 名称的特定模块，如果没有则输出 MESSAGE（“User module is not loaded”）。

注意：如果存在 User 名称的特定模块，则不输出 MESSAGE，并返回布尔值。

3.5.6.10 LOWER_BOUND() 给定数组下限

返回一个表示数组变量第 n 维下限的整数值。参数 ndim，选择第 n 个维度。

格式： LOWER_BOUND (array variable[数组变量] , ndim[维数]);

输入：设定 FOR 循环 array[1]-array[2]

输出：MESSAGE 输出 array 数组中的所有值

示例：

示例变量名：

```
DINT i //声明 i 变量
DINT array //声明 array 数组
```

示例程序：

```
FOR i:= LOWER_BOUND(array, 1) TO UPPER_BOUND(array, 2) DO
MESSAGE("Array value %1", array[i]);
END_FOR ;
```

此例中，通过 LOWER_BOUND()、UPPER_BOUND()函数设定 FOR 循环的上下限，并通过 MESSAGE 在日志中打印数组变量的值

注意：定义的整形数组需要给定初始化的值：0

3.5.6.11 UPPER_BOUND() 给定数组上限

返回一个表示数组变量 n 维的上限。参数 ndim，选择第 n 个维度。

格式： UPPER_BOUND *(array variable[数组变量] , ndim[维数]);*

输入：设定 FOR 循环 array[1]-array[5]

输出：MESSAGE 输出 array 数组中的所有值

示例：

示例变量名：

```
DINT i //声明 i 变量
DINT array //声明 array 数组
```

示例程序：

```
FOR i:= LOWER_BOUND(array, 1) TO UPPER_BOUND(array, 5) DO
array[i] := 0;
END_FOR ;
```

此例中，通过 LOWER_BOUND()、UPPER_BOUND()函数设定 FOR 循环的上下限，并通过 MESSAGE 在日志中打印数组变量的值

注意：定义的整形数组需要给定初始化的值：0

3.5.6.12 TIMERQ 计时器过期查询

返回一个 BOOL 值，该值指示计时器是否过期。

格式：TIMERQ (TIMER variable);

输入：TIMER 类型的变量

输出：BOOL 类型的数据，计时器 variable 是否过期

示例:

TIMERSTART(tmr1) ;

…

WAIT (TIMERQ(tmr1);

这个例子展示了如何等待计时器过期。

3.5.6.13 TIMERR 计时器过期查询

返回指定 TIMER 变量中剩余时间。时间以秒为单位。

格式: TIMERR(TIMER variable);

输入: TIMER 类型的变量

输出: LReal 类型的数据，计时器 variable 的剩余时间

示例:

TIMERSTART(tmr1) ;

…

MESSAGE ("time remain is %1", TIMERR(tmr1));

在上面的例子中，一个可能的消息可以是"剩余时间是 1.234"。

3.6 指令长度限制表

V2.0.0 以下

序号	指令名	长度限制说明
1	(* *)	参数允许输入长度<154
2	:=	指令总长字长<255
3	ALIAS	指令总长字长<255
4	CALL	指令总长字长<255
5	CASE	指令总长字长<255
6	CLEARMOVE	指令总长字长<255
7	CLOCKRESET	指令总长字长<255
8	CLOCKSTART	指令总长字长<255
9	CLOCKSTOP	指令总长字长<255
10	CMARC	指令总长字长<255
11	CMLIN	指令总长字长<255
12	CMCIRC	指令总长字长<255
13	CMSPLINE	指令总长字长<255
14	CONTINUE	指令总长字长<255
15	DWELL	指令总长字长<255
16	ENDPROG	指令总长字长<255

17	EPATH	指令总长字长<255
18	ERROR	指令总长字长<255
19	EXEC	指令总长字长<255
20	EXIT	指令总长字长<255
21	FMJOINT	指令总长字长<255
22	FMLIN	指令总长字长<255
23	FMCIRC	指令总长字长<255
24	FMCIRCA	指令总长字长<255
25	FOR	指令总长字长<255
26	GOTO	指令总长字长<255
27	IF	指令总长字长<255
28	INTRCOND	指令总长字长<255
29	INTRALLOW	指令总长字长<255
30	INTRDENY	指令总长字长<255
31	INTRDIS	指令总长字长<255
32	INTRENA	指令总长字长<255
33	INTRERRNO	指令总长字长<255
34	INTRSET	指令总长字长<255
35	INTRTIMER	指令总长字长<255
36	KMAXT	指令总长字长<255
37	KMAXJ	指令总长字长<255
38	KMAXW	指令总长字长<255
39	LABEL	指令总长字长<255
40	LOAD	指令总长字长<255
41	MCIRC	指令总长字长<255
42	MCIRCA	指令总长字长<255
43	MESSAGE	指令总长字长<255
44	MJOINT	指令总长字长<255
45	MLIN	指令总长字长<255
46	PULSE	指令总长字长<255
47	RESTART	指令总长字长<255
48	RETURN	指令总长字长<255
49	SAVE	指令总长字长<255
50	SETOFRAME	指令总长字长<255
51	SLINB	指令总长字长<255
52	STARTMOVE	指令总长字长<255
53	STOPPROG	指令总长字长<255
54	STOPMOVE	指令总长字长<255
55	TRIGCALL	指令总长字长<255
56	TRIGON	指令总长字长<255
57	TRIGSET	指令总长字长<255

58	WAIT	指令总长字长<255
59	UNLOAD	指令总长字长<255
60	WAIT_POS	指令总长字长<255
61	WAIT_ZONE	指令总长字长<255
62	WHILE	指令总长字长<255
63	WRISTMODE	指令总长字长<255

V2.0.0 及以上

序号	指令名	长度限制说明
1	:=	<233(根据使用有不同的限制)
2	(**)	参数允许输入长度<154
3	case	参数允许输入长度<184
4	LABEL	参数允许输入长度<127
5	GOTO	参数允许输入长度<127
6	IF	判断表达式<101;bool and 变量表达式 95（变量名称长度之和<84）
7	DWELL	参数允许输入长度<128
8	WAIT	参数允许输入长度<127
9	STARTMOVE	参数允许输入长度<128
10	STOPMOVE	参数允许输入长度<128
11	WAIT_POS	参数允许输入长度 128
12	FOR	from 值<310， to 值 310,by 值<310（整体 1067）
13	WHILE	条件表达式<128
14	EXEC	条件表达式<126
15	UNLOAD	条件表达式<126
16	LOAD	条件表达式<126
17	PULSE	条件表达式<126
18	ERROR	条件表达式<128
19	MESSAGE	text 参数<126，两个 expr 表达式长度<128
20	TRIGCALL	单个参数<128
21	TRIGSET	单个参数<128
22	INTRCOND	单个参数<128
23	INTRERRNO	单个参数<128
24	INTRSET	受限于变量名的最大长度
25	ALIAS	受限于变量名的最大长度
26	TRIGON	受限于变量名的最大长度
27	INTRALLOW	受限于变量名的最大长度
28	INTRDENY	受限于变量名的最大长度
29	INTRDIS	受限于变量名的最大长度

30	INTRENA	受限于变量名的最大长度
38	CLOCKREST	受限于变量名的最大长度
39	CLOCKSTART	受限于变量名的最大长度
40	CLOCKSTOP	受限于变量名的最大长度
31	MCIRC	第一个点位<126;第二个<124
32	MJOINT	点位<126
33	MLIN	点位<127
34	KMAXJ	单个参数<128
35	KMAXT	单个参数<129
36	KMAXW	单个参数<130
37	MCIRCA	单个参数<127
41	CLEARMOVE	指令本身长度：13
42	CONTINUE	指令本身长度：14
43	EXIT	指令本身长度：15
44	RETURN	指令本身长度：16
45	RESTART	指令本身长度：17
46	ENDPROG	指令本身长度：18

服务热线：**400-0528877**

本产品的额定功率、规格、外部尺寸等如需改良而进行变更，恕不另行通告。技术数据和插图仅作为供货参考，保留更改权利。

埃夫特智能装备股份有限公司
EFORT INTELLIGENT EQUIPMENT CO.,LTD
中国安徽省芜湖市鸠江经济开发区万春东路 96 号
No 96,Wanchun Road,Jiujiang Economic Development Zone,
Wuhu, Anhui,China
网址:http://www.efort.com.cn