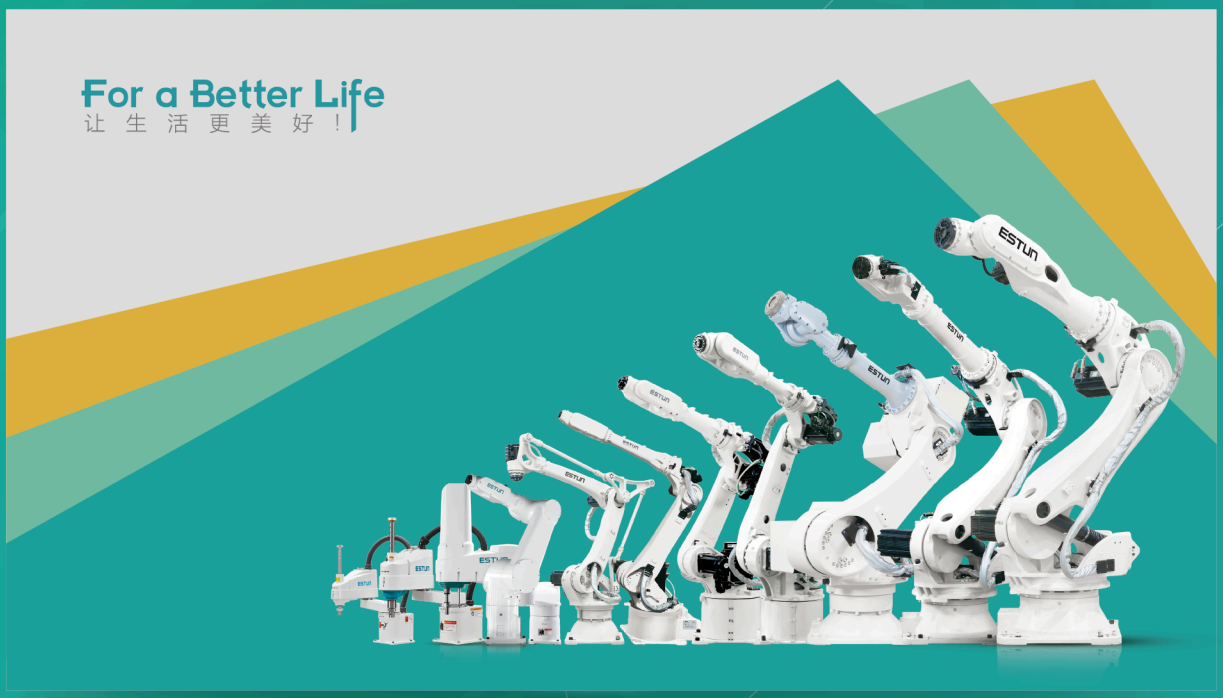
ESTUN
AUTOMATION

工业机器人
INDUSTRIAL
ROBOT




# iER系列

## ERI功能使用手册

iEPM01V30413-01

自动化核心部件
运动控制系统

工业机器人及
智能系统工程

数字化产品
数字化服务

# 申明

## 版权与知识产权

本手册及其包含的所有文本、图像、设计、软件及相关信息（统称“文档材料”）均为南京埃斯顿自动化股份有限公司（简称“埃斯顿公司”）的独家财产，受中华人民共和国及相关国际版权法和条约保护。除非获得埃斯顿公司的明确书面授权，否则严禁对本文档材料进行任何形式的复制、分发、修改、公开展示或反向工程。

## 免责声明

1. 埃斯顿公司致力于确保手册信息的准确性，但不对其完整性、准确性或无误性作任何明示或默示的担保。埃斯顿公司将定期校对本手册的内容，并在之后的版本中作必要的更改，同时保留在不影响功能的情况下进行技术更改的权利，恕不另行通知。
2. 安全操作是用户的首要责任。用户有责任确保其人员已接受充分培训，并严格按照本手册、所有相关安全规程以及国家、地区和行业标准来安装、操作和维护本产品。任何因未遵守操作规程、擅自改装或使用非原装备件所导致的人身伤害、设备损坏或生产损失，埃斯顿公司概不负责。
3. 在任何情况下， 埃斯顿公司均不对因使用或无法使用本产品及手册而引发的任何间接损失、附带损失、利润损失、数据丢失或生产中断等后果性损害承担责任。

## 文档申明

1. 本手册为“Original Instruction”，经制造商埃斯顿公司核实确认。
2. 如需本手册的其他语言翻译版本，可向制造商或其授权代表申请获取。

# 前言

## 资料简介

本手册适用于控制系统RCS3系列，介绍埃斯顿机器人ERI功能使用方法。

## 修订记录

| 修订日期 | 发布版本 | 变更内容 |
| --- | --- | --- |
| 2026.02 | 01 | 新建手册。 |

## 资料获取

本文档不随产品发货，如需获取电子版PDF文件，可以通过以下方式获取： 登录埃斯顿公司官网网站（www.estun.com），“**服务与支持 -> 资料下载**”，搜索关键字并下载。

# 安全使用须知

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

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

## 使用人员的定义

1. 操作人员：
   - 执行机器人电源的开启（ON）与关闭（OFF）操作；
   - 通过示教器和控制柜操作面板启动、暂停或停止已预设的机器人运行程序；
   - 监控机器人日常运行状态，及时反馈异常情况，不涉及程序修改、示教或维修操作；
   - 操作人员不得在安全围栏内进行作业。
2. 编程人员：
   - 进行机器人程序的编写、修改、优化及存储管理；
   - 在指定安全区域内，通过示教器完成机器人的示教编程、轨迹调试等作业；
   - 验证程序运行效果，确保程序符合作业需求，不涉及机器人核心部件维修或结构调整；
   - 编程人员必须接受针对机器人的专业培训。
3. 维修人员：
   - 执行机器人的定期维护、故障排查、部件修理、参数调整及零件更换作业；
   - 为维修作业需要，可在安全区域内进行必要的机器人辅助示教或操作；
   - 记录维护维修数据，反馈设备运行损耗情况，确保机器人性能稳定；
   - 维修人员必须接受针对机器人的专业培训。

使用人员权限划分：本表格界定安全围栏外作业人员的操作权限范围；符号 “O”代表该人员具备对应操作权限，“-” 代表无操作权限。

表 0-1 安全围栏外作业权限一览表

| 作业内容 | 操作人员 | 编程人员 | 维修人员 |
| --- | --- | --- | --- |
| 机器人控制柜电源通断 | O | O | O |

表 0-1 安全围栏外作业权限一览表 (续)

| 作业内容 | 操作人员 | 编程人员 | 维修人员 |
|---|---|---|---|
| 选择自动/远程（A)、手动（T1）、手动100%（T2）操作模式 | - | O | O |
| 示教器选择机器人程序 | - | O | O |
| 外部设备选择机器人程序 | - | O | O |
| 示教器启动机器人程序 | O | O | O |
| 示教器复位报警 | - | O | O |
| 示教器参数设置 | - | O | - |
| 示教器示教操作 | - | O | O |
| 控制柜急停按钮操作 | O | O | O |
| 示教器急停按钮操作 | O | O | O |
| 控制面板维护 | - | O | - |
| 维护期间示教操作 | - | - | O |

所有涉及机器人操作、编程调试及维护维修的作业场景，操作人员、编程人员、维修人员必须严格遵守安全防护要求，全程规范穿戴符合安全标准的防护装备：

- 需选用与作业内容适配的作业服，避免衣物卷入设备或勾挂部件；
- 必须穿戴具备防砸、防滑功能的安全鞋；
- 作业期间需正确佩戴安全帽；
- 需定期检查防护装备完好性，破损、老化的装备需立即更换，若涉及电气维修、高温作业等特殊场景，还需按对应作业规范额外配备绝缘手套、耐热手套等专项防护装备。

**说明**

关于培训的更多信息，请咨询埃斯顿公司工作人员。

## 安全标示

本手册中出现的以下安全标示，均为核心操作与安全指引依据，用户在阅读手册内容时，必须仔细研读标示对应的说明，严格遵守相关要求，规避安全风险。

| 标示符号 | 定义说明 |
|---|---|
|  危险 | • 本标志对应的文本表示存在高度潜在危险。若无法避免相关风险，将直接导致人员死亡或严重伤害，必须严格遵守相关安全要求。 |
|  警告 | • 本标志对应的文本表示存在中度或低度潜在危险。若忽视相关提示，可能导致人员轻微伤害或中等程度伤害，需规范操作规避风险。 |

| 标示符号 | 定义说明 |
| --- | --- |
|  注意 | • 本标志对应的文本表示存在非人身伤害类潜在风险。若忽视相关说明，可能造成设备损坏、数据丢失、设备性能下降或其他不可预知的不良后果。 |
|  说明 | • 本标志对应的文本为正文的附加补充信息，用于强调核心内容、补充操作细节或解释关键概念，帮助用户更准确理解手册内容并高效完成操作。 |

## 使用人员的安全事项

1. 搬运与安装机器人时，必须严格遵循埃斯顿公司规定的标准流程操作。采用非标准方法可能导致机器人倾倒、部件损坏，进而引发安全事故。
2. 机器人安装前必须划定安全防护区域，需在其工作区域周边设置防护栅栏及安全警示牌。
3. 机器人上方严禁放置悬挂物，防止物体掉落砸坏机器人及周边设备。
4. 严禁倚靠控制柜或随意触碰按钮，避免机器人产生意外动作，造成人身伤害或设备损坏
5. 拆分机器人过程中，需警惕设备上松动或脱落的零件，防止砸伤作业人员。
6. 对外部设备进行单独调试时，必须先断开机器人电源，再开展操作。
7. 所有外围设备必须连接符合规范的接地线，确保用电安全。
8. 首次操作机器人时，必须先以低速运行，再逐步提升速度，全程观察并确认设备无异常。
9. 使用示教器时，戴手套可能引发操作失误，必须摘下手套再进行操作。
10. 程序、系统变量等关键信息可存储至存储卡等介质，为防止意外导致数据丢失，建议用户定期备份数据。
11. 严禁擅自搬动机器人各轴，否则可能引发人身伤害或设备损坏。
12. 进行控制柜与机器人、外围设备间的配线及配管时，必须采取防护措施（如将管线、线缆从预埋坑穿过或加装保护盖遮挡），避免被人员踩踏或叉车碾压损坏。
13. 机器人工作期间均可能产生意外动作，对工作范围内人员造成严重伤害或损坏设备。机器人启动前，必须测试安全门、抱闸、安全指示灯等所有安全措施的可靠性，且必须确保工作范围内无其他人员。
14. 通过软件设定的动作范围及负载条件，严禁设定超出产品规格表中的规定值；设置不当可能引发人员伤害或设备损坏。
15. 若必须在机器人工作范围内作业，需遵循以下安全规则：
    - 需将机器人切换至手动模式后再通过示教器上使能，并断开上位机控制等所有自动控制方式；
    - 留意机器人转动关节，防止头发、衣物卷入；同时需警惕机器人及其他附属设备运动可能引发的其他危险（如碰撞、挤压等）；
    - 作业前需测试电机抱闸功能是否正常，防止因机器人异常运动导致人身伤害；

- 需预先规划机器人突然向自身方位运动时的应急避让方案；
- 需预设安全躲避区域，以备紧急情况使用。

**⚠ 危险**

在任何工况下，操作人员、编程人员及维修人员均严禁在机器人本体机械臂下方站立、停留或进行作业。该区域存在极高安全风险：机器人可能因系统故障发生异常运动，或被他人误触使能开关启动，导致人员被碰撞、挤压，造成重伤甚至死亡。所有相关人员必须严格遵守此禁令，确保作业安全。

**操作人员的安全注意事项：**

1. 操作机器人前，必须先按下控制柜前门及示教器右上方的急停按钮，检查 “伺服准备” 指示灯是否熄灭，确认机器人电源已完全关闭。
2. 操作期间，严禁非工作人员触碰或操作控制柜。
3. 向机器人安装工具时，必须先关闭（OFF）控制柜及待装工具的电源，并悬挂警示牌。安装期间若擅自接通电源，可能造成触电，或导致机器人意外运动引发伤害。
4. 急停功能：
   - 急停功能独立于机器人所有电气控制，可强制停止机器人全部运动。
   - 触发急停后，驱动机器人上伺服电机运行的动力电源将会断开，伺服电机抱闸，机器人停止运动；
     需先释放急停按钮，再重新启动机器人，设备方可恢复运行。

| 急停按钮示意图 | 特别说明 |
| --- | --- |
|  | 所有急停按钮均需安装在易于触及的位置，确保突发意外时能快速触发，及时停止机器人运行。 |

**⚠ 危险**

操作人员需要注意伺服电机的动力线、连接夹具和其他装置的动力线的高电压危险。

**⚠ 注意**

急停功能仅用于紧急场景（如人身安全受威胁、设备突发失控等）下强制停止机器人，严禁用于日常程序暂停、正常关机等非紧急操作。

**编程人员的安全注意事项：**

机器人示教作业期间，编程人员若需进入机器人动作范围，必须重点关注作业安全。

**⚠ 注意**

自动模式下，使能的接通和断开通过操作示教器上的Mot按钮实现切换；对于正在运行的机器人，需要先按下Stop按键停止运行，之后再按下Mot按钮实现使能断开。

安全使用示教器，必须遵守以下规则：

- 确保紧急按钮始终处于有效状态，避免应急时无法正常操作。
- 示教者进入机器人工作区域时，必须随身携带示教器，避免他人在不知情时操作机器人。
- 禁止将示教器放置于机器人工作范围内，防止机器人运动时碰撞示教器，引发异常动作。

**维修人员的安全注意事项：**

1. 机器人发热部件安全警示：机器人正常运行时，部分关键部件会发热，其中伺服电机、减速机的发热尤为明显，靠近或直接触碰易导致烫伤。若需在部件发热状态下触碰，必须佩戴耐热手套等防护用具，避免直接接触高温部件。

   **⚠ 注意**

   触摸这些高温部件前，需先用手靠近部件表面初步感受温度，判断是否安全，防止直接触摸导致烫伤。

   机器人停机后，需等待高温部件冷却至安全温度，再开展维修工作，避免高温引发人员灼伤。

2. 拆卸部件的安全注意事项：拆卸前必须先确认齿轮、轴承等内部零件完全停止旋转与运动，再打开部件盖子或保护装置；严禁在上述部件旋转时开启保护装置，防止卷入风险。若内部存在活动零件（非固定状态），需使用辅助装置将其固定在原始位置，避免拆卸过程中零件移位、掉落引发安全隐患。
   维修、安装或保养后的首次启动测试，必须严格遵循以下步骤：
   a. 彻底清理机器人本体及其工作区域内的所有工具、材料及临时装置，确保无遗留物；
   b. 完整恢复并确认所有安全防护装置已正确安装且功能正常；
   c. 确认所有人员已撤离至机器人安全工作范围之外，禁止任何人员在测试期间进入作业区域；
   d. 启动测试时，应重点监控本次维修或更换部件的运行状态，实时观察是否存在异响、卡顿等异常，及时停机排查。

**⚠ 注意**

在维修过程中，禁止站立、攀爬或倚靠机器人任何部位。必须使用符合安全标准的登高设备，以防跌落造成严重人身伤害。

3. 气动系统安全注意事项：关闭气源泵后，系统内仍会残留带压气体或液体，此类残留物具有潜在能量。需通过专项措施释放残留能量，避免其对人员或设备造成冲击、喷射等伤害；维修气动元件前，必须确保系统内残留能量已完全释放。

**⚠ 注意**

为防止气动系统能量意外释放引发安全事故，必须安装安全阀，作为系统压力过载时的强制保护装置。

4. 故障诊断阶段需开启机器人电源，但维修机器人时必须完全关闭电源，并切断所有相关电源连接，避免带电作业引发安全风险。
5. 抱闸检测：机器人正常运行时，抱闸易因磨损导致锁止性能下降，需定期开展功能检测以保障关节安全锁止。具体检测步骤如下：
   a. 将机器人各关节调整至承受最大负载的姿态，模拟极端受力场景。
   b. 关闭机器人主电源，使抱闸自动生效。
   c. 在各关节的固定部件与活动部件间做对位标记，确保标记清晰可辨，便于后续观察位移。
   d. 静置规定时长（通常为 10-30 分钟）后，观察各关节的对位标记是否偏移，以此判断抱闸是否存在滑闸、锁止失效情况。
6. 向减速机加注润滑油时，存在高温油液烫伤、压力溅射、油液污染等风险，可能伤及人员或损坏设备。开展加油作业前，必须严格遵守以下安全规范：
   - 加注或排放润滑油时，需佩戴防化手套、护目镜等防护装备。
   - 打开油腔盖前需确认油腔温度降至安全范围（建议≤40℃），操作时保持安全距离（建议≥50cm）；油腔内可能残留压力，需缓慢开启盖子，避免油液溅射引发伤害。
   - 按油量表刻度加注润滑油，严禁超过最大刻度线；加注完成后，需检查油液指示口，确认油位处于标准范围内。

- 严禁将不同型号、不同牌号的润滑油混入同一减速机；更换润滑油型号前，需用专用清洁剂彻底冲洗油腔，确保残余旧油清理干净。
- 排放旧油时，需将油腔内旧油彻底排空（可借助负压设备辅助）；加油后需再次检查油液指示口，双重确认油位符合要求。

**⚠ 注意**

放空减速机旧油前，可先短时间低速运行机器人（建议 5-10 分钟，使油液温度升至 40-60℃），通过加热降低油液黏度，让放油更顺畅，同时减少油腔内壁的旧油残留。需注意控制运行时长，避免油液温度过高（不超过 60℃），防止后续操作时因高温油液引发烫伤。

## 刀具及外围设备安全注意事项

机器人关闭后，其连接的刀具、外围设备可能仍独立运行（未随机器人断电）。若此类设备的电源线、动力线缆出现破损，或人员误触带电部件，易引发触电等人身伤害。因此，操作前必须单独切断刀具及外围设备的电源，并检查线缆完整性，再开展后续作业。

## 机器人手臂紧急救援安全规范

当发生机器人手臂夹伤操作人员的紧急情况时，需立即对夹持手臂进行紧急移除，具体操作及安全要求如下：

1. 紧急移除操作：
   - 小型机器人手臂：可采用手动移除方式，需 2 人及以上协作（1 人固定手臂、1 人协助受困者脱困），严禁单人操作以防手臂晃动加重伤害。
   - 大型机器人手臂：必须使用吊车等专业承重设备，严禁使用小型推车、撬棍等非承重工具，避免设备倾覆或手臂坠落引发二次事故。
2. 释放关节抱闸前，必须先通过专用支架、承重绳索等工具稳固固定机械臂，确保机械臂完全脱离 “悬空承重” 状态，防止释放抱闸后手臂因重力下坠，对受困者造成挤压、砸伤等二次伤害。

**⚠ 警告**

移除过程中若遇机械锁死、部件卡顿等问题，需立即停止操作并联系埃斯顿公司技术人员获取专业指导，严禁擅自拆解关节、抱闸等关键部件，避免破坏设备结构或引发新风险。

# 目录

申明.....2
前言.....3
安全使用须知.....4
图目录.....14
1 产品介绍.....15
1.1 ERI库说明.....15
1.2 通讯机制.....15
1.2.1 IO数据交互方式.....15
1.2.2 TCP/IP通讯交互方式.....16
1.2.3 总线形式数据交互.....18
1.3 ERI实时控制.....18
1.3.1 概况.....18
1.3.2 关节控制.....18
1.3.3 笛卡尔控制.....19
1.4 ERI 纠偏控制.....19
1.4.1 概况.....19
1.4.2 纠偏坐标系.....19
1.5 ERI信号处理.....22
2 安全.....23
2.1 安全提示.....23
3 安装.....24
3.1 系统要求.....24
3.1.1 硬件.....24
3.1.2 软件.....24
3.2 ERI软件包安装.....24
3.3 通过示教器界面激活.....24
4 配置.....26
4.1 C++工程调用SDK库.....26
4.1.1 环境要求.....26
4.1.2 搭建步骤.....26
5 编程.....29
5.1 SDK库接口列表.....29
5.1.1 默认构造函数.....29
5.1.2 设置机器人通信参数.....29
5.1.3 获取机器人连接状态.....29

5.1.4 命令接口列表.....30
5.1.4.1 获取机器人错误信息 .....30
5.1.4.2 获取对机器人控制权.....30
5.1.4.3 释放对机器人的控制权.....30
5.1.4.4 开启实时控制.....30
5.1.4.5 MovL指令.....30
5.1.4.6 MovJ指令.....31
5.1.4.7 设置采样周期.....31
5.1.4.8 设置ERI纠偏参数.....31
5.1.4.9 设置工具.....31
5.1.4.10 设置用户坐标系.....32
5.1.4.11 获取工具参数.....32
5.1.4.12 获取用户坐标系.....32
5.1.4.13 修改工具参数.....32
5.1.4.14 修改用户坐标系.....32
5.1.4.15 获取机器人当前笛卡尔位置.....33
5.1.4.16 获取机器人当前关节轴数据.....33
5.1.4.17 获取当前系统所有的DO状态.....33
5.1.4.18 获取指定输出口的状态.....33
5.1.4.19 设置指定输出口的状态.....33
5.1.5 实时接口列表.....34
5.1.5.1 实时关节运动.....34
5.1.5.2 实时笛卡尔运动 .....34
5.1.5.3 实时偏移运动.....34
5.1.6 状态数据回调接口.....34
5.2 总线命令交互列表.....35
5.2.1 数据格式说明.....35
5.2.2 命令交互说明.....36
5.3 ERI指令概览.....37
5.3.1 SetERIProgID.....37
5.3.2 EnableERIControl.....37
5.3.3 SetERIParam.....37
5.3.4 SetCorrParam.....38
5.3.5 EnableERICorrect.....39
6 示例配置和示例程序.....40
6.1 接口使用样例.....40
6.1.1 获取控制权.....40
6.1.2 MOVJ命令.....40

6.1.3 MOVL命令.....40
6.1.4 实时运动CPOS.....41
6.1.5 实时运动APOS.....41
6.1.6 纠偏点位发送.....41
6.1.7 释放控制权.....42
6.1.8 UDP状态包使用示例.....42
7 典型案例.....43
7.1 客户端连接机器人并实时控制.....43
7.1.1 通过关节实时控制自行规划运动操作机器人到目标位置.....43
7.1.2 通过笛卡尔实时控制控制机器人画圆.....46
7.2 示教程序与外部实时数据交替运行.....48
7.3 通过SIMIO控制机器人纠偏的使用样例.....49
附录 A 错误列表.....52

# 图目录

图 1-1 通过IO数据交互示意图..... 16
图 1-2 通过SDK数据交互示意图..... 17
图 1-3 基于WORLD坐标系的纠偏..... 20
图 1-4 基于TOOL坐标系的纠偏..... 21
图 1-5 基于TTS坐标系的纠偏..... 22
图 3-1 ERI功能激活界面..... 25

# 1 产品介绍

## 1.1 ERI库说明

针对实时位置引导、焊缝跟踪、力控装配等对实时性、自主扩展性要求较高的高级应用场景，埃斯顿（ESTUN）机器人提供一套外部控制接口（以下简称ERI 库），支持用户通过二次开发实现对机器人的精准控制。

ERI具备如下核心特性：

- 通讯协议： 支持通过 TCP/IP、工业总线两种方式与外部设备进行双向数据交互。
- 实时控制：可按照外部系统的轨迹规划逻辑，执行周期性轨迹控制。
- 纠偏控制：支持接入视觉、力觉等传感器数据，通过实时处理传感器反馈信息，动态修正机器人运动轨迹。

## 1.2 通讯机制

机器人控制系统和外部系统之间，可通过以下两种方式实现通讯与数据交互：

- 通过机器人控制系统的SimIO模块完成数据交换。
- 通过SDK 接口直接与机器人控制系统进行交互。

### 1.2.1 IO数据交互方式

机器人控制系统的输入/输出系统数据交互的流程如下：

- 外部系统可向机器人控制系统的SIMDI（模拟数字输入）/SIMAI（模拟量输入） 通道中写入数据和信号。ERI会周期性读取并处理这些数据和信号，再由机器人控制系统将处理完成的数据和结果，通过SIMDO（模拟数字输出）/SIMAO（模拟量输出）通道反馈至外部系统。
- 外部系统可通过工业总线系统（如 EtherNet/IP、Profinet 通讯协议）与机器人控制系统的输入/输出系统进行数据映射，实现高速、稳定的 IO 信号交互。。

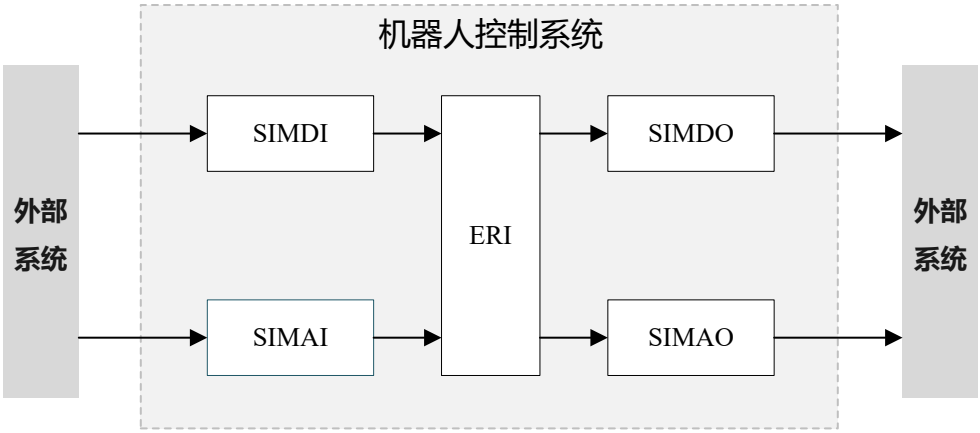



图 1-1 通过IO数据交互示意图

### 1.2.2 TCP/IP通讯交互方式

ERI 库的 TCP/IP 通讯交互基于 SDK 库实现，SDK 内部已封装与机器人系统的实时通讯交互机制，无需用户自行开发，仅需调用封装好的接口即可完成数据交互与指令下发，核心支持以下三类功能：

1. 机器人数据主动上报
   机器人控制系统会按照固定周期，向 SDK 库主动传输核心运行数据，外部系统可直接读取，包含但不限于：位置类数据：笛卡尔坐标位置信息、关节坐标位置信息；
   状态类数据：机器人运行状态（如空闲、运动中、暂停）、系统错误信息 / 报警码。
2. ERI 模式下的实时数据下发
   当机器人控制系统切换至ERI 模式后，外部系统可调用 SDK 中的实时函数接口，向机器人控制系统传输实时控制数据，例如：
   传感器类数据：视觉 / 力觉传感器的实时测量值；
   轨迹类数据：外部系统规划的运动点位信息、轨迹插值数据。
3. ERI 模式下的命令指令下发
   机器人控制系统切换至 ERI 模式后，外部系统可调用 SDK 中的命令函数接口，向机器人发送各类控制命令，例如：
   运动类命令：关节运动指令、笛卡尔空间运动指令；
   系统类命令：清除错误命令、实时控制功能开启 / 关闭命令、急停复位命令。

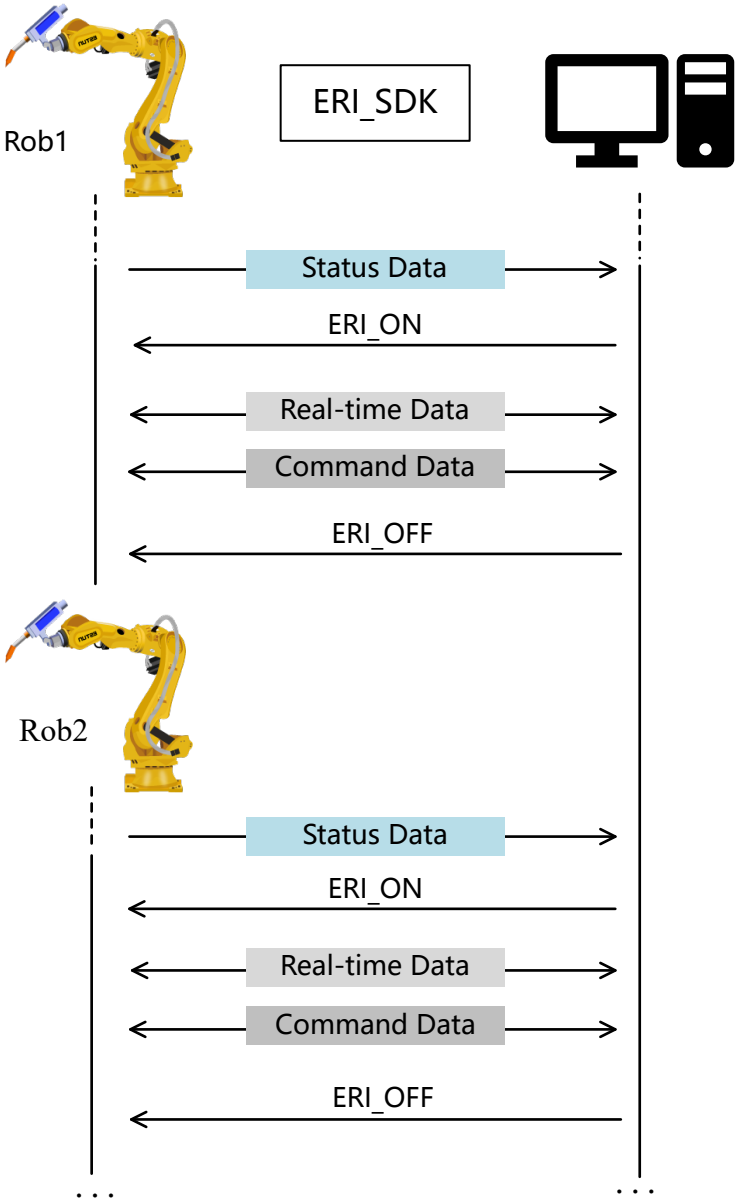
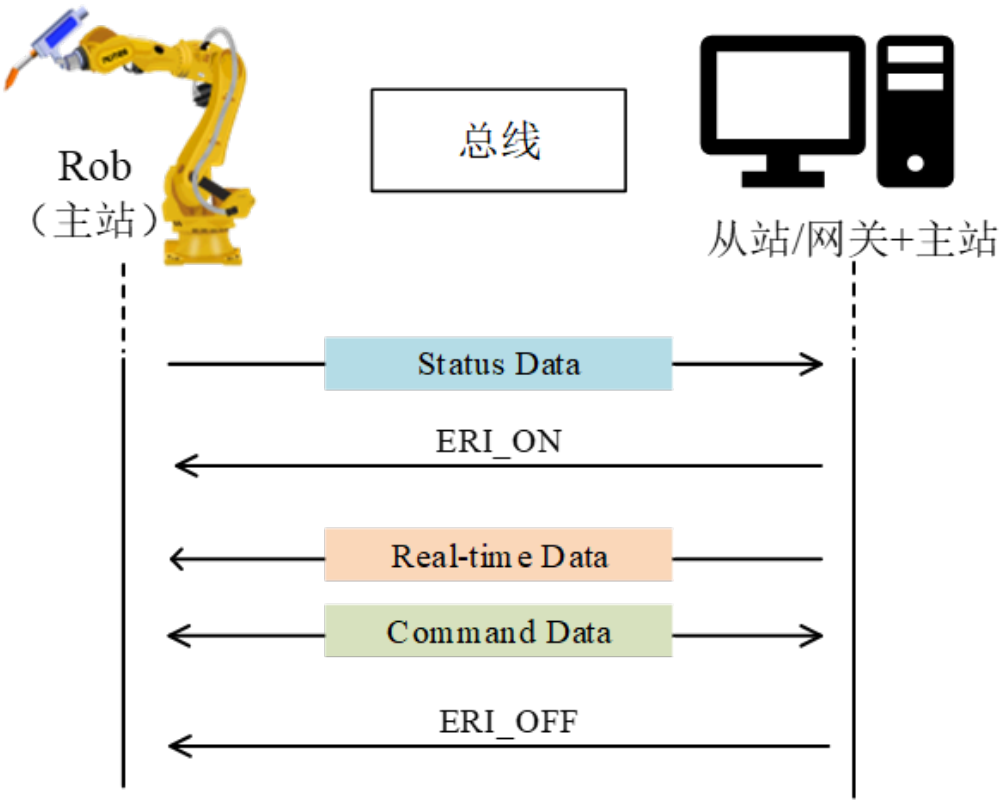



图 1-2 通过SDK数据交互示意图

### 1.2.3 总线形式数据交互




ERI 库支持通过EtherCAT 等工业实时总线实现与外部系统的高速数据交互。机器人作为主站，可以与从站或者通过网关与其他主站通讯。如需网关产品，可联系厂家。

## 1.3 ERI实时控制

### 1.3.1 概况

ERI 实时控制功能支持用户跳过机器人控制器底层的轨迹规划模块，直接通过 ERI 接口向控制器高速写入目标关节位置或笛卡尔位置数据

### 1.3.2 关节控制

关节控制模式下，外部系统可通过 ERI 接口实时向下发规划好的关节位置数据。

机器人各轴将直接以下发的关节角度为目标值进行运动，无需控制器进行轨迹插补计算，可实现多轴的独立或协同精准控制。

### 1.3.3 笛卡尔控制

笛卡尔控制模式下，外部系统可通过 ERI 接口实时向机器人下发末端执行器（TCP）的目标位置数据，数据包含 6 个维度参数：

位置参数：XYZ（空间直角坐标系下的位置坐标）

姿态参数：ABC（采用ZYX 顺序的欧拉角定义末端姿态）

机器人将以下发的 TCP 位置与姿态为目标，自主协调各轴运动，完成末端执行器的空间定位。

## 1.4 ERI 纠偏控制

### 1.4.1 概况

ERI纠偏控制功能可让用户在编写好的示教轨迹基础上，根据工艺需求和外部传感器反馈数据，实时调整机器人运动轨迹，弥补示教轨迹与实际作业需求的偏差，提升作业精度。

### 1.4.2 纠偏坐标系

ERI纠偏控制的纠偏量将在示教轨迹的基础上，基于用户指定的纠偏坐标系进行偏移。偏移量包括dX、dY、dZ（线性偏移）和dA、dB、dC（姿态偏移），其中姿态偏移为基于指定纠偏坐标系，按照XYZ顺序绕固定轴旋转的偏移量。ERI纠偏控制支持多种坐标系，适配不同作业场景，具体如下：

1. WORLD坐标系
   如图 1-3 基于WORLD坐标系的纠偏所示，基于WORLD坐标系的纠偏，在机器人运行示教轨迹（橘色）的同时，若缓慢增加X方向纠偏量，机器人实际轨迹（紫色）将为示教轨迹和沿WORLD坐标系X轴方向的纠偏量的叠加。

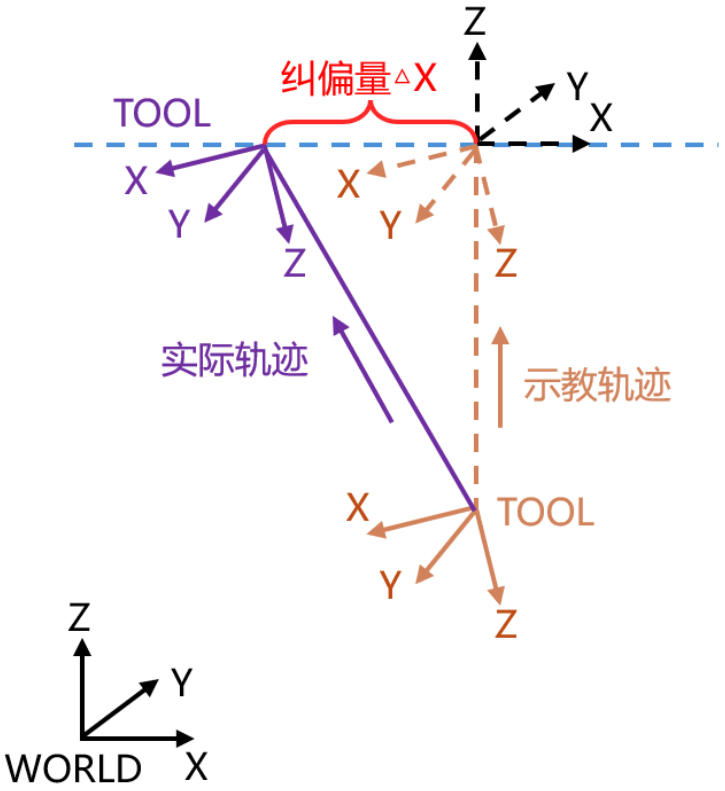



图 1-3 基于WORLD坐标系的纠偏

2. TOOL坐标系

如图 1-4 基于TOOL坐标系的纠偏所示，基于TOOL坐标系的纠偏，在机器人运行示教轨迹（橘色）的同时，若缓慢增加X方向纠偏量，机器人实际轨迹（紫色）将为示教轨迹与沿TOOL坐标系X轴方向的纠偏量的叠加。

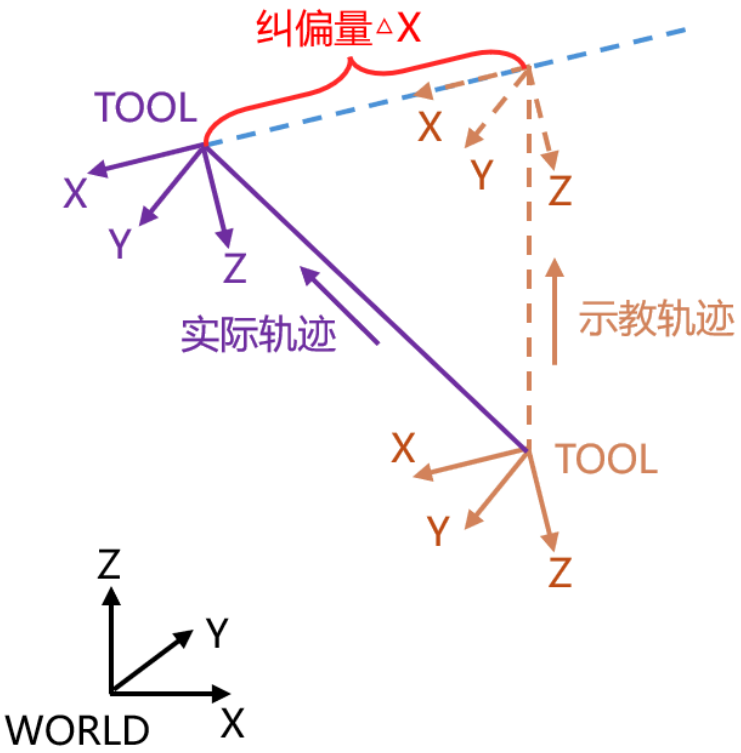



图 1-4 基于TOOL坐标系的纠偏

3. TTS坐标系

TTS（Trajectory Tangent System）坐标系定义如下：坐标系原点与机器人当前TCP位置始终绑定，X轴沿运动轨迹的瞬时切线方向（即路径前进方向），Y轴由路径切线（X轴）与工具坐标系的X轴叉积确定，Z轴由该坐标系的X轴和Y轴叉积确定。

如图 1-5 基于TTS坐标系的纠偏所示，基于TTS坐标系的纠偏，在机器人运行示教轨迹（橘色）的同时，缓慢增加Z方向纠偏量，机器人实际轨迹（紫色）为示教轨迹和沿TTS坐标系Z轴方向纠偏量的叠加。

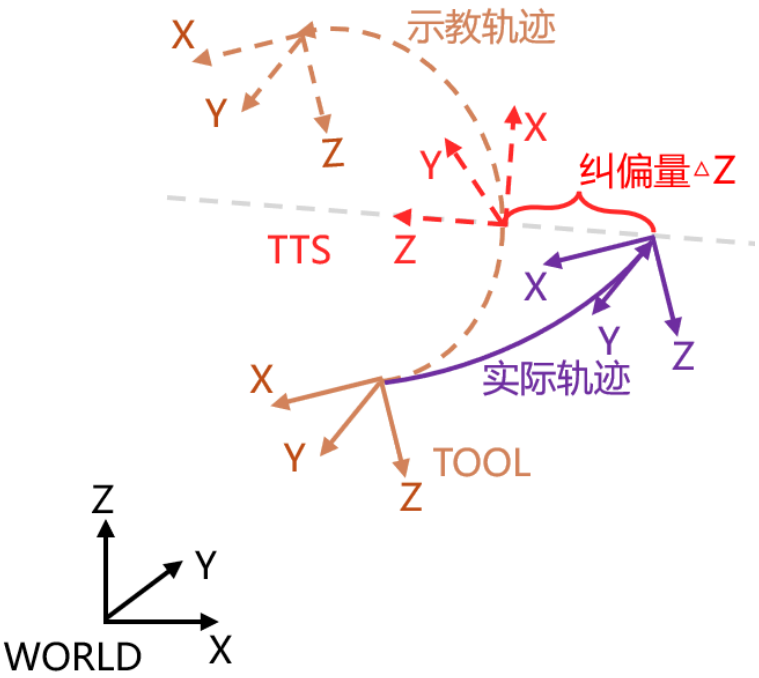



图 1-5 基于TTS坐标系的纠偏

## 1.5 ERI信号处理

用户在使用ERI实时控制与纠偏控制时，可高速获取机器人状态，包括：笛卡尔位置、关节位置、纠偏状态、IO状态、错误信息等。

ERI可接收外部系统高速实时数据（周期默认4ms），同时提供三种滤波模式处理规划数据，若用户运动规划比较光滑，可关闭滤波器以降低延迟。

# 2 安全

## 2.1 安全提示

错误使用ERI时，可能导致人员受伤和财产损失，如：

- 在实时模式或者纠偏模式下，发送了异常位置信息。
- 在硬件存在故障，如布线错误，外部传感器故障等。

在ERI模式下使用实时控制时，系统集成商有责任通过建立合适的安全措施将受伤危险以及财产损失降低到最小程度，例如：根据机器人的安全行程范围在外部系统中设立合理的工作区域。

纠偏模式下，ERI纠偏控制默认纠偏阈值为±6mm/°，用户可通过指令修改该阈值。当外部发来的纠偏量超阈值时，系统会按照设置的纠偏阈值来执行该纠偏量。

# 3 安装

安装ERI功能前，需确保机器人控制系统、外部控制设备（如上位机、PLC）的硬件、软件符合以下要求，否则可能导致ERI功能无法正常激活、运行，或出现通讯异常、控制精度不足等问题。

## 3.1 系统要求

### 3.1.1 硬件

- ERC60F系列机器人控制器；
- 通过SDK库进行数据交换时，带实时操作系统和100Mbit实时网卡的、在全双工模式下受处理器保护的外部系统；
- 埃斯顿机器人。

### 3.1.2 软件

机器人控制系统V3.04及以上的系统软件版本，低于该版本需升级控制系统软件。

## 3.2 ERI软件包安装

联系ESTUN获取ERI软件包，并参考《iER系列机器人操作使用手册》进行软件包安装。

## 3.3 通过示教器界面激活

**关于本任务**

使用ERI软件包，需要对ERI功能进行授权激活，可以通过以下步骤进行激活：




图 3-1 ERI功能激活界面

## 步骤

步骤1　在管理员权限下，依次进入“**通讯 -> ERI**”界面，通过点击界面中**“获取”**按钮获得产品码。将此产品码发送给厂家，由厂家处获取激活码（与产品码一一对应）；（如果进入界面时，激活状态为已激活，则不需要此激活配置过程）。

步骤2　通过由厂家获取的激活码输入到激活码编辑框中，点击**“激活”**按钮，如果输入的激活码正确，则状态变为授权；如果输入的激活码不正确，则状态仍为未授权。

# 4 配置

## 4.1 C++工程调用SDK库

### 4.1.1 环境要求

环境需满足以下要求：

| 配置项 | 最低要求 | 推荐配置 | 备注 |
| --- | --- | --- | --- |
| 系统版本 | Win7 | Win10 | / |
| 内存大小 | 4GB | 8GB | / |
| VisualStudio版本 | 2015 | 2019 | 如有其他VS版本，可联系厂家 |

### 4.1.2 搭建步骤

**步骤**

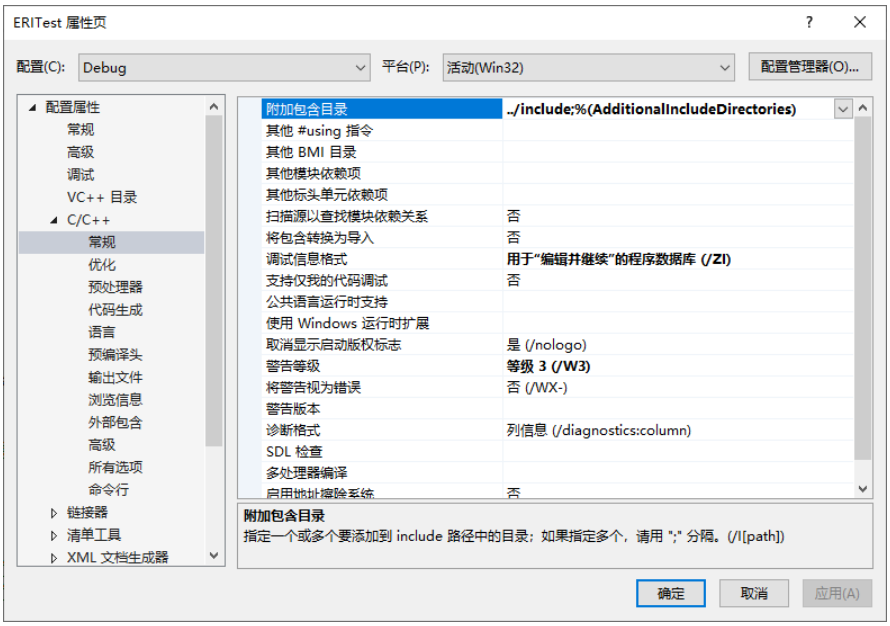
步骤1 新建工程

打开vs2019，点击左上角“**文件 -> 新建 -> 项目**”，选择项目类型为【控制台应用】，如下图所示，为项目命名并选择其存放位置，完成新建工程。

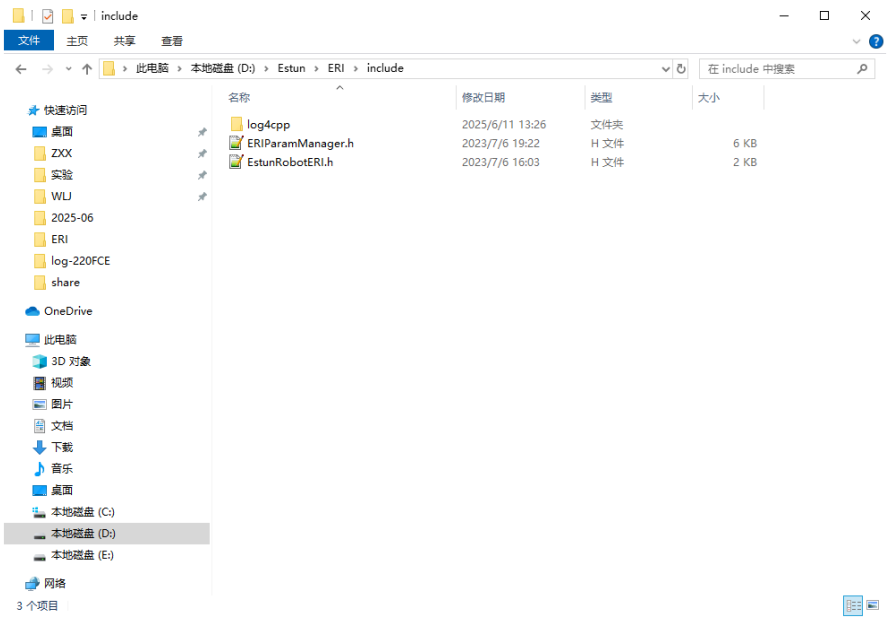



步骤2 配置头文件路径和库路径

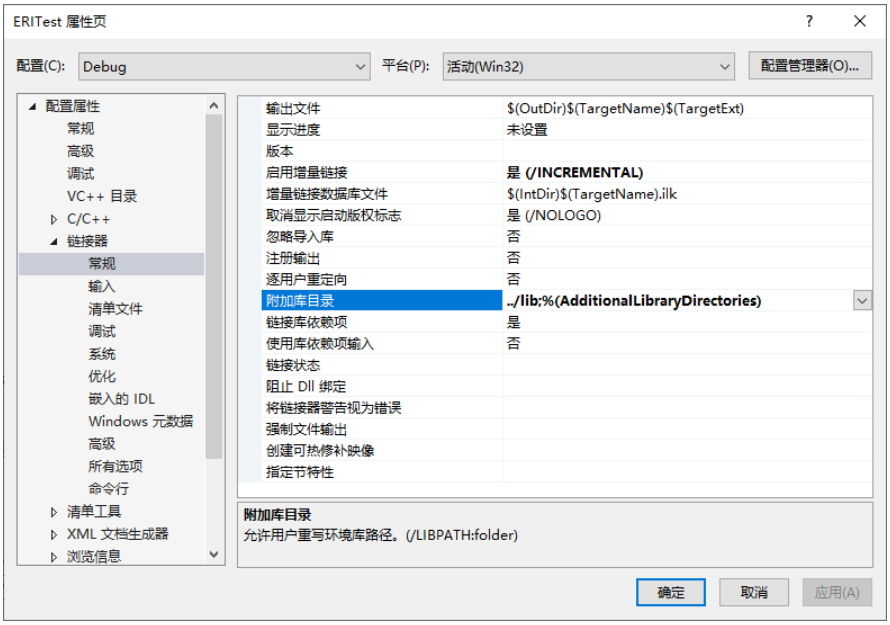
a. 右键工程，点击“**属性**”，点击“**配置属性 -> C/C++ -> 常规**”，在【附加包含目录】中添加ERI库的include目录。



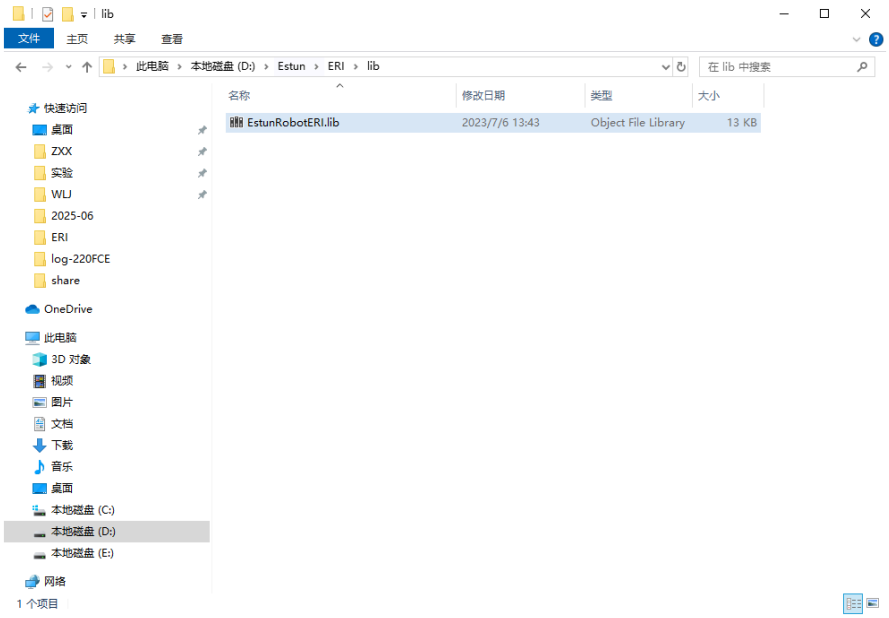

include目录内包含ERI头文件，如下图所示：




b. 右键工程，点击“**属性**”，点击“**配置属性 -> 链接器 -> 常规**”，在【附加库目录】中添加ERI库的lib目录。




lib目录内包含文件如下图所示：



# 5 编程

## 5.1 SDK库接口列表

ERI的SDK库函数分为命令接口、实时接口、状态接口三种方式。

1. 命令接口：命令执行为阻塞式机制，即发送数据后必须等待执行结束后才会返回结果。特别是运动指令，必须等当前运动结束了才能下发新运动指令。
2. 实时接口：实时接口为发送完数据即返回结果，具体的发送频率由用户决定，但需要保证与控制器采样周期的频率保持一致。如外部设备每间隔20ms调用实时接口向控制器发送点位数据，那应该先调用“设置采样周期”函数将采样周期设为20ms。
3. 状态接口：提供UDP状态回调函数，SDK库会每隔2ms调用该函数推送控制器相关的状态数据。也可通过命令接口中的部分函数获取，如通过调用获取笛卡尔位置/关节位置函数获取坐标。

### 5.1.1 默认构造函数

| | |
|---|---|
| 名称 | ERIParamManager::ERIParamManager() |
| 功能 | 类的默认构造接口。 |

### 5.1.2 设置机器人通信参数

| | |
|---|---|
| 名称 | int ERIParamManager::setCommunicationParam(char *robotIp,UINT16 cmdPort,UINT16 servoPort,UINT16 statusPort) |
| 功能 | 设置机器人通信参数，设置完毕客户端会根据设置参数，自动连接机器人。 |
| 返回值 | 返回值为0，设置成功；返回值为1，设置失败。 |
| 参数说明 | * @robotIp 机器人ip地址<br>* @cmdPort 命令通道端口号<br>* @servoPort servo通道端口号<br>* @statusPort 状态包通道端口号<br>* @返回值 0设置成功 1设置失败 |

### 5.1.3 获取机器人连接状态

| | |
|---|---|
| 名称 | void getRobotConnStatus(char &cmdStatus,char &servoStatus,char &udpStatus) |
| 功能 | 获取机器人连接情况 |
| 返回值 | void |

| | |
|---|---|
| 参数说明 | * @ cmdStatus 命令通道连接状态 成功1<br>* @ servoStatus实时通道连接状态 成功1<br>* @ udpStatus 机器人状态反馈连接状态 成功 1 |

### 5.1.4 命令接口列表

此分类下为命令相关接口。

#### 5.1.4.1 获取机器人错误信息

| | |
|---|---|
| 名称 | bool ERIParamManager::getCurErrMsg(int &errId, std::string &errMsg) |
| 功能 | 获取机器人错误信息。 |
| 返回值 | 成功返回true，失败返回false。 |
| 参数说明 | * @errId 错误ID<br>* @errMsg 错误消息内容 |

#### 5.1.4.2 获取对机器人控制权

| | |
|---|---|
| 名称 | bool ERIParamManager::getControlRobot() |
| 功能 | 获取对机器人控制权。 |
| 返回值 | 成功返回true，失败返回false。 |
| 参数说明 | 无 |

#### 5.1.4.3 释放对机器人的控制权

| | |
|---|---|
| 名称 | bool ERIParamManager::releaseControlOverRobot() |
| 功能 | 释放对机器人的控制权。 |
| 返回值 | 成功返回true，失败返回false。 |
| 参数说明 | 无 |

#### 5.1.4.4 开启实时控制

| | |
|---|---|
| 名称 | bool ERIParamManager::startServo(int motionType = 0) |
| 功能 | 开始实时控制。 |
| 返回值 | 成功返回true，失败返回false。 |
| 参数说明 | * @motionType 运动类型，0为关节，1为笛卡尔 |

#### 5.1.4.5 MovL指令

| | |
|---|---|
| 名称 | bool ERIParamManager::movL(double pos[]) |

| | |
|---|---|
| 功能 | 执行MovL运动。 |
| 返回值 | 成功返回true，失败返回false。 |
| 参数说明 | * @pos 点位信息 |

#### 5.1.4.6 MovJ指令

| | |
|---|---|
| 名称 | bool ERIParamManager::movj(double pos[]) |
| 功能 | 执行MovJ运动。 |
| 返回值 | 成功返回true，失败返回false。 |
| 参数说明 | * @pos 点位信息 |

#### 5.1.4.7 设置采样周期

| | |
|---|---|
| 名称 | bool ERIParamManager::setERIMotionParam(UINT16 period,UINT16 filterType) |
| 功能 | 设置采样周期。 |
| 返回值 | 成功返回true，失败返回false。 |
| 参数说明 | * @period 采样周期，单位ms<br>* @filterType 滤波类型， 0-关闭滤波 1-平均滤波值 2-二阶低通滤波 3-椭圆滤波 |

#### 5.1.4.8 设置ERI纠偏参数

| | |
|---|---|
| 名称 | bool ERIParamManager::setCorrectParam(double alarmThreshold[6],UINT16 corrCoord) |
| 功能 | 设置ERI纠偏参数。 |
| 返回值 | 成功返回true，失败返回false。 |
| 参数说明 | * @alarmThreshold 纠偏量报警阈值，单位mm/°<br>* @corrCoord 纠偏坐标系。<br>• 0-世界坐标系<br>• 1-工具坐标系<br>• 2-用户坐标系<br>• 3-TTS坐标系 |

#### 5.1.4.9 设置工具

| | |
|---|---|
| 名称 | bool ERIParamManager::setTool(const UINT16 toolId) |
| 功能 | 设置工具。详细含义见《ER系列机器人操作手册》10.3系统数据类型TOOL。 |
| 返回值 | 成功返回true，失败返回false。 |
| 参数说明 | * @toolId 工具Id |

#### 5.1.4.10 设置用户坐标系

| | |
|---|---|
| 名称 | bool ERIParamManager::setUser(const UINT16 userId) |
| 功能 | 设置用户坐标系。详细含义见《ER系列机器人操作手册》10.3系统数据类型USER-COORD。 |
| 返回值 | 成功返回true，失败返回false。 |
| 参数说明 | * @userId 用户坐标系Id |

#### 5.1.4.11 获取工具参数

| | |
|---|---|
| 名称 | bool ERIParamManager::getTool(int toolId,double toolData[6]) |
| 功能 | 获取工具参数。详细含义见《ER系列机器人操作手册》10.3系统数据类型TOOL。 |
| 返回值 | 成功返回true，失败返回false。 |
| 参数说明 | * @toolId 工具Id<br>* @toolData 对应的工具坐标参数，分别对应 x,y,z,a,b,c |

#### 5.1.4.12 获取用户坐标系

| | |
|---|---|
| 名称 | bool ERIParamManager::getUser(int userId,double userData[6]) |
| 功能 | 获取用户坐标系。详细含义见《ER系列机器人操作手册》10.3系统数据类型USER-COORD）。 |
| 返回值 | 成功返回true，失败返回false。 |
| 参数说明 | * @userId 用户坐标系Id<br>* @userData 对应的用户坐标系参数 |

#### 5.1.4.13 修改工具参数

| | |
|---|---|
| 名称 | bool ERIParamManager::modifyTool(int toolId,double toolData[6]) |
| 功能 | 修改工具参数。详细含义见《ER系列机器人操作手册》10.3系统数据类型TOOL。 |
| 返回值 | 成功返回true，失败返回false。 |
| 参数说明 | * @toolId 要修改的工具Id<br>* @toolData 对应的工具坐标参数，分别对应 x,y,z,a,b,c |

#### 5.1.4.14 修改用户坐标系

| | |
|---|---|
| 名称 | bool ERIParamManager:: modifyUser(int userId,double userData[6]) |
| 功能 | 获取用户坐标系。详细含义见《ER系列机器人操作手册》10.3系统数据类型USER-COORD。 |
| 返回值 | 成功返回true，失败返回false。 |

| | |
|---|---|
| 参数说明 | * @userId 要修改的用户坐标系Id<br>* @userData 对应的用户坐标系参数 |

#### 5.1.4.15 获取机器人当前笛卡尔位置

| | |
|---|---|
| 名称 | bool getWorldCpos(double pos[16]) |
| 功能 | 获取机器人世界坐标系下当前位置 |
| 返回值 | 成功返回true，失败返回false。 |
| 参数说明 | * @pos 输出型参数，存放机器人当前位置 |

#### 5.1.4.16 获取机器人当前关节轴数据

| | |
|---|---|
| 名称 | bool getJointValue(double pos[16]) |
| 功能 | 获取机器人当前各关节轴值 |
| 返回值 | 成功返回true，失败返回false。 |
| 参数说明 | * @@pos 输出型参数，存放机器人当前位置 |

#### 5.1.4.17 获取当前系统所有的DO状态

| | |
|---|---|
| 名称 | UINT64 getCurrentDO() |
| 功能 | 以十进制格式返回当前系统64个DO的值 |
| 参数说明 | @val 以十进制格式显示当前系统所有的DO值 |

#### 5.1.4.18 获取指定输出口的状态

| | |
|---|---|
| 名称 | UINT32 getCurrentDO(const UINT32 port) |
| 功能 | 获取指定输出口的输出状态，仅能获取单个DO状态 |
| 返回值 | @val 端口值 |
| 参数说明 | @port 端口号，返回值为0或1 |

#### 5.1.4.19 设置指定输出口的状态

| | |
|---|---|
| 名称 | bool setDO(UINT32 port,UINT32 val) |
| 功能 | 设置指定输出口的输出状态 |
| 返回值 | 成功返回true，失败返回false。 |
| 参数说明 | @port 待设置的端口号<br>@val 待设置的端口值，参数可设为0或1 |

### 5.1.5 实时接口列表

此分类下为实时相关接口。

#### 5.1.5.1 实时关节运动

| | |
|---|---|
| 名称 | int ERIParamManager::servoToAPOS(UINT64& timeStamp,double *pos,bool IsEndPoint = false) |
| 功能 | 执行实时关节运动。 |
| 返回值 | 返回值为-1，执行失败；其他，执行成功 |
| 参数说明 | * @timeStamp 时间戳<br>* @pos 点位信息<br>* @IsEndPoint 是否结束点 |

#### 5.1.5.2 实时笛卡尔运动

| | |
|---|---|
| 名称 | int ERIParamManager::servoToCPOS(UINT64& timeStamp,double *pos,bocol IsEndPoint = false) |
| 功能 | 执行实时笛卡尔运动。 |
| 返回值 | 返回值为-1，执行失败；其他，执行成功 |
| 参数说明 | * @timeStamp 时间戳<br>* @pos 点位信息 xyzabc（姿态为基于欧拉角表示）<br>* @IsEndPoint 是否结束点 |

#### 5.1.5.3 实时偏移运动

| | |
|---|---|
| 名称 | int ERIParamManager::servoToOFFSET(UINT64& timeStamp, double *pos,bool IsEndPoint = false) |
| 功能 | 执行实时偏移运动。 |
| 返回值 | 返回值为-1，执行失败；其他，执行成功 |
| 参数说明 | * @timeStamp 时间戳<br>* @pos 点位信息：xyzabc（姿态为基于固定坐标系表示）<br>* @IsEndPoint 是否结束点 |

### 5.1.6 状态数据回调接口

| | |
|---|---|
| 名称 | void setRobotStatusCallback(P_RobotStatusCallback pfunc) |
| 功能 | UDP状态回调函数，通过该设置回调函数方法，设置需要回调的函数信息。 |
| 返回值 | 无 |

| 参数说明 | * @pfunc 函数指针，指向需要的状态包回调函数 |
| --- | --- |
| 补充说明 | typedef void (*P_RobotStatusCallback)(bool isRoborError, char isStartCorrect, bool isEndPoint)<br>/**<br>* @isRobotError 机器人端是否有错误<br>* @isStartCorrect 是否开始纠偏<br>* @ isRobotError 是否是结束点<br>**/ |

## 5.2 总线命令交互列表

### 5.2.1 数据格式说明

埃斯顿机器人的ERI实时接口可支持总线的通讯方式，具体的数据定义如下表格所示：

| Estun Ecat master send/ User Ecat master receive | | Bytes accumulation | | User Ecat master send/Estun Ecat master receive | |
| --- | --- | --- | --- | --- | --- |
| Data type | Note | 0 | 0 | Data type | Note |
| Int32 (4 bytes) | 初始指令计数 | 4 | 1 | Int32 | 初始指令计数 |
| float (4 bytes) | J1轴实际轴角 | 8 | 33 | float | J1轴目标轴角 |
| float (4 bytes) | J2轴实际轴角 | 12 | 65 | float | J2轴目标轴角 |
| float (4 bytes) | J3轴实际轴角 | 16 | 97 | float | J3轴目标轴角 |
| float (4 bytes) | J4轴实际轴角 | 20 | 129 | float | J4轴目标轴角 |
| float (4 bytes) | J5轴实际轴角 | 24 | 161 | float | J5轴目标轴角 |
| float (4 bytes) | J6轴实际轴角 | 28 | 193 | float | J6轴目标轴角 |
| float (4 bytes) | J1位置差分求速度 | 32 | 225 | Int32 | 结束指令计数 |
| float (4 bytes) | J2位置差分求速度 | 36 | 257 | Int32 | Control word (User发给Estun) $IN[1] - $IN[32] |
| float (4 bytes) | J3位置差分求速度 | 40 | 289 | Uint8 | 异常标志位位（1正常；0异常，包含断网掉电） |
| float (4 bytes) | J4位置差分求速度 | 44 | 297 | Int32 | 预留 |
| float (4 bytes) | J5位置差分求速度 | 48 | 329 | Int32 | 预留 |
| float (4 bytes) | J6位置差分求速度 | 52 | 361 | Int32 | 预留 |
| float (4 bytes) | J1扭矩百分比 | 56 | 393 | Int32 | 预留 |
| float (5 bytes) | J2扭矩百分比 | 60 | 425 | Int32 | 预留 |
| float (6 bytes) | J3扭矩百分比 | 64 | 457 | Int32 | 预留 |
| float (7 bytes) | J4扭矩百分比 | 68 | 489 | Int32 | 预留 |
| float (8 bytes) | J5扭矩百分比 | 72 | 521 | Int32 | 预留 |

| Estun Ecat master send/ User Ecat master receive | | Bytes accumulation | | User Ecat master send/Estun Ecat master receive | |
|---|---|---|---|---|---|
| float (9 bytes) | J6扭矩百分比 | 76 | 553 | Int32 | 预留 |
| Int32 (4 bytes) | 结束指令计数 | 80 | 585 | Int32 | 预留 |
| Int32 (4 bytes) | Status word (Estun发给User) $OUT[201] - $OUT[232] | 84 | 617 | Int32 | 预留 |
| Int32 (4 bytes) | 错误报警 $OUT[233] - $OUT[264] | 88 | 649 | Int32 | 预留 |
| Int32 (4 bytes) | 预留 | 92 | 681 | Int32 | 预留 |
| Int32 (4 bytes) | 预留 | 96 | 713 | Int32 | 预留 |
| Int32 (4 bytes) | 预留 | 100 | 745 | Int32 | 预留 |
| Int32 (4 bytes) | 预留 | 104 | 777 | Int32 | 预留 |
| Int32 (4 bytes) | 预留 | 108 | 809 | Int32 | 预留 |
| Int32 (4 bytes) | 预留 | 112 | 841 | Int32 | 预留 |
| Int32 (4 bytes) | 预留 | 116 | 873 | Int32 | 预留 |
| Int32 (4 bytes) | 预留 | 120 | 905 | Int32 | 预留 |
| Int32 (4 bytes) | 预留 | 124 | 937 | Int32 | |
| Int32 (4 bytes) | 预留 | 128 | 969 | Int32 | |

### 5.2.2 命令交互说明

对Control word作说明：

| 命令编号 | 含义 | 说明 |
|---|---|---|
| 1 | 获取控制权 | 切换机器人的控制权限，变更为外部控制。 |
| 2 | 开启实时控制 | 开启机器人的实时控制模式，开启后机器人会周期性从总线上获取目标位置，并控制机器人运动。 |
| 3 | 清楚报警 | 可清除机器人当前的报警信息 |
| 4 | 上使能 | 控制机器人上励磁 |
| 5 | 释放控制权下使能 | 切换机器人的控制权限，变更为示教器控制,并控制机器人下励磁 |

**⚠ 注意**

机器人系统每次成功响应命令时，会将同样的命令号存放在Status word，切换失败则返回0。User端得到答复后，需同步将Control word的命令数据复位成0。

举例说明：User需要进行清楚机器人报警操作，先将Control word值设为3，机器人系统接收到数据后会进行清除当前报警信息，成功清除后会将Status word值设为3，User端接收到数据后会同步将Control word值复位成0。

## 5.3 ERI指令概览

ERI的指令需搭配ERI的外部接口一起使用，可设置实时纠偏参数与切换机器人的控制权。

ERI指令列表：

| | |
|---|---|
| 控制指令 | SetERIProgID |
| | EnableERIControl |
| | SetERIParam |
| | SetCorrParam |
| | EnableERICorrect |

### 5.3.1 SetERIProgID

该指令用于设置ERI执行外部程序ID。通常搭配EnableERIControl指令一起使用。当系统的控制权切换到外部控制时，外部系统使用设置的程序编号或者工艺来控制机器人运动。

SetERIProgID ID

·ID：外部程序ID

示例：

```
SetERIProgID(1) // 设置外部程序
```

### 5.3.2 EnableERIControl

该指令用于将机器人控制权切换到ERI。远程模式下执行该指令后，可将机器人的执行权限切换给ERI控制，只有在该模式下，外部设备才可控制机器人进行实时运动。

示例：

```
EnableERIControl() // 将机器人的执行权限切换到ERI控制，该命令会阻
塞执行，待执行权限切换回机器人控制后才会执行结束。
```

### 5.3.3 SetERIParam

该指令用于设置采样周期和滤波器类型。

SetERIParam period filterType

- period：采样周期 单位 ms
- filterType：滤波器类型，系统内置四种滤波器，可对外部发来的实时位置按照设定的滤波类型进行滤波处理。

| | |
|---|---|
| DisEnable | 关闭滤波器 |
| Average | 平均滤波器 |
| LowPass | 二阶低通滤波 |
| Ellipse | 椭圆滤波 |

示例：

```
SetERIParam(8, ” Average” ) // 系统将按照8ms来运行点位数据，并利用
平均滤波器对数据进行滤波处理。
```

### 5.3.4 SetCorrParam

该指令用于设置报警阈值、纠偏坐标系等。

SetCorrParam XCThreshold YCThreshold ZCThreshold ACThreshold BCThreshold CC-Threshold corrCoord

- XCThreshold：相对纠偏坐标系X轴方向最大纠偏量。
- YCThreshold：相对纠偏坐标系Y轴方向最大纠偏量。
- ZCThreshold：相对纠偏坐标系Z轴方向最大纠偏量。
- ACThreshold：相对纠偏坐标系A轴方向最大纠偏量。
- BCThreshold：相对纠偏坐标系B轴方向最大纠偏量
- CCThreshold：相对纠偏坐标系C轴方向最大纠偏量。
- corrCoord：纠偏数据所参考的坐标系类型，
  - World：世界坐标系
  - Tool：工具坐标系
  - TTS：TTS坐标系

示例：

```
SetCorrParam(5,5,5,5,5,5,0) //
将XYZABC的最大纠偏量设为5mm，同时设置计算纠偏数据所基于的坐标系类型为世界坐标系。
```

### 5.3.5 EnableERICorrect

该指令用于开启/关闭ERI的实时纠偏功能，通常搭配外部实时接口使用。开启后系统会根据发来的纠偏数据调整机器人的运行轨迹。

EnableERICorrect EnableSts

- EnableSts：使能状态

示例：

```
EnableERICorrect (ON) // 开启纠偏功能
```

# 6 示例配置和示例程序

## 6.1 接口使用样例

本节主要通过基本样例进一步介绍ERI库主要接口的使用。

### 6.1.1 获取控制权

```
ERIParamManager eriManager;
if (eriManager.getControlRobot())
{
 printf("获取控制权成功!\n");
}
else
{
 printf("获取控制权失败!\n");
}
```

### 6.1.2 MOVJ命令

```
ERIParamManager eriManager;
double targetJoint[6];
targetJoint[0] = -1.262151;
targetJoint[1] = 23.006887;
targetJoint[2] = 32.440697;
targetJoint[3] = 3.420175;
targetJoint[4] = 48.415222;
targetJoint[5] = 14.849081;
if (eriManager.movj(targetJoint))
{
 printf("movj success \n");
}
else
{
 printf("movj failed \n");
}
```

### 6.1.3 MOVL命令

```
ERIParamManager eriManager;
double targetJoint[6];
targetJoint[0] = 855.84;
targetJoint[1] = -10.379000000000019;
targetJoint[2] = 561.518;
targetJoint[3] = -154.89800000000014;
targetJoint[4] = 25.520999999999958;
targetJoint[5] = -162.25800000000004;
if (eriManager.movL(targetJoint))
{
 printf("MovL success \n");
}
else
{
```

```
 printf("MovL failed \n");
}
```

### 6.1.4 实时运动CPOS

```
if (eriManager.startServo(1))
{
g  eriSDKLog->info("start servo cpos success");
 while(true)
{
  bool isEndPoint = false;
  double jointData[6] = {0};
  UINT64 currentTimeStamp = 0;
  if (-1 == eriManager.servoToCPOS(currentTimeStamp,jointData,isEndPoint))
  {
   g_eriSDKLog->error("servoToCpos error stop!");
   return;
  }

}
}
else
{
 g_eriSDKLog->debug("start servo cpos failed");
}
```

### 6.1.5 实时运动APOS

```
if (eriManager.startServo(0))
{
 printf("start servo movj success \n");
 while(true)
{
bool isEndPoint = false;
 double jointData[6] = {0};
 UINT64 currentTimeStamp = 0;
 if (-1 == eriManager.servoToAPOS(currentTimeStamp,jointData,isEndPoint))
 {
  g_eriSDKLog->error("servoToCpos error stop!");
  return;
 }

}

}
else
{
 printf("start servo movj failed \n");
}
```

### 6.1.6 纠偏点位发送

```
While(true)
{
bool isEndPoint= false;
double jointData[6] = {0.0};
```

```
UINT64 currentTimeStamp = 0;
if (-1 == eriManager.servoToOFFSET(currentTimeStamp,jointData,isEndPoint))
{
 g_eriSDKLog->error("servoToOFFSET error stop!");
 return;

}
}
```

### 6.1.7 释放控制权

```
ERIParamManager eriManager;

if (eriManager.releaseControlOverRobot())
{
 printf("释放控制权成功!\n");
}
else
{
 printf("释放控制权失败!\n");
}
```

### 6.1.8 UDP状态包使用示例

- **定义回调函数**

```
//机器人状态信息
struct RobotStatus
{
 bool isRobotError; //机器人端是否有错误
 char isStartCorrect;//是否开始纠偏
 bool isEndPoint;//机器人处理完毕最后一个点位
 double worldCpos[6];//当前机器人世界坐标系下位置
 double jointValue[6];//当前机器人各关节轴值
};
void RobotStatusCallback(RobotStatus robotStatus)
{
}
```

- **注册回调函数**

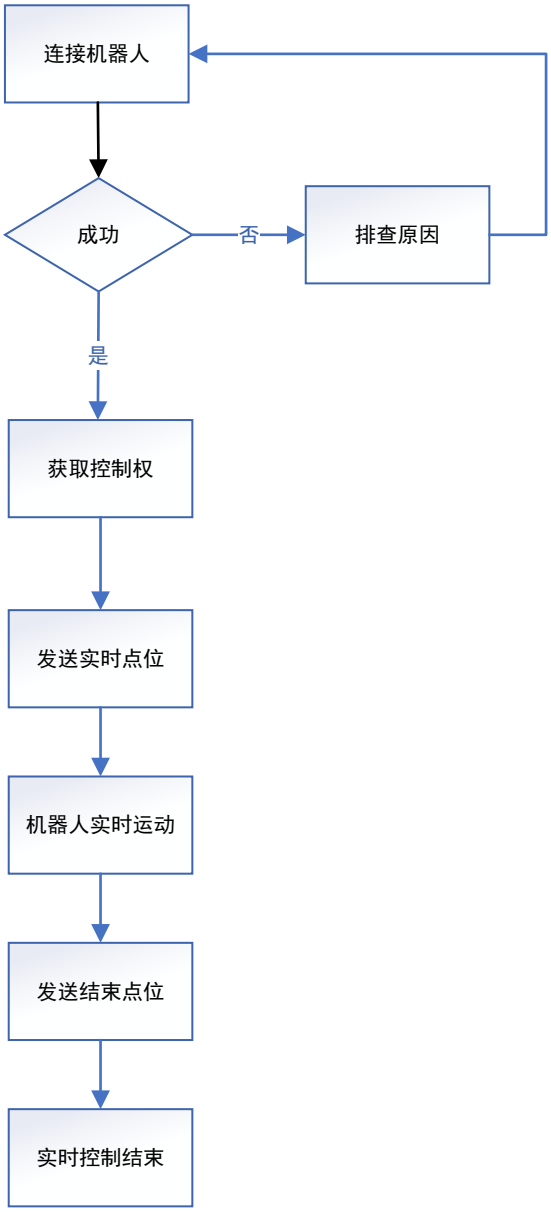
```
ERIParamManager eriManager;
eriManager.setRobotStatusCallback(RobotStatusCallback);
在该回调函数中可周期性接收，机器人有无错误(robotStatus.isRobotError)、
纠偏是否开始(robotStatus.isStartCorrect)、实时运动是否结束(robotStatus.isEndPoint)
```

# 7 典型案例

## 7.1 客户端连接机器人并实时控制

在进行该操作前，请确保机器人系统一切正常，无任何错误，机器人可正常点动。

整体流程：




### 7.1.1 通过关节实时控制自行规划运动操作机器人到目标位置

示例代码如下：

```
// ================================================================================
// 示例名称：ERI 实时控制插补示例
// 功能：连接机器人 → 获取控制权 → 设置运动参数 → MoveJ 到起点 → 启动实时控制
//     → 多维五次多项式轨迹插补 → 实时发送目标位姿 → 结束运动 → 释放控制权
// ================================================================================

#include "stdafx.h"
#include <iostream>
#include "ERIParamManager.h"

#ifndef M_PI
#define M_PI 3.14159265358979323846
#endif

// ------------------------ ERI 控制器实例 ------------------------
ERIParamManager eriManager;

// ------------------------ 五次多项式插值函数 ------------------------
double compute5thPoly(double p0, double p1, double T, double t) {
    double s = t / T;
    double s3 = s * s * s;
    double s4 = s3 * s;
    double s5 = s4 * s;
    return p0 + (p1 - p0) * (10.0 * s3 - 15.0 * s4 + 6.0 * s5);
}

// ------------------------ 实时插补运动主函数 ------------------------
void processRealtimeData() {
    // ===== 控制与滤波参数 =====
    const int sampleTime = 4;   // 控制周期（ms）
    const int filterType = 0;   // 0 = 无滤波

    // ===== 起点与终点位姿 =====
    double startPos[6] = { 0, 0, 0, 0, 0, 0 };        // 初始位姿（可替换为实际读取值）
    double goalPos[6] = { 20, 30, 40, 50, 60, 70 };   // 目标位姿
    double target[6];                                 // 实时目标位姿
    UINT64 timeStamp = 0;                             // 时间戳

    // ===== 获取控制权 =====
    if (!eriManager.getControlRobot()) {
        std::cerr << "获取机器人控制权失败!" << std::endl;
        return;
    }
    std::cout << "成功获取机器人控制权!" << std::endl;

    // ===== 设置实时控制参数 =====
    if (!eriManager.setERIMotionParam(sampleTime, filterType)) {
        std::cerr << "设置ERI运动参数失败!" << std::endl;
        return;
    }
    std::cout << "已设置实时控制参数。" << std::endl;

    // ===== MoveJ 到起始位置 =====
    if (!eriManager.movJ(startPos)) {
        std::cerr << "MoveJ 到起始点失败!" << std::endl;
        return;
    }
    std::cout << "MoveJ 成功！等待稳定..." << std::endl;
    Sleep(1000);

    // ===== 启动实时控制（APOS模式） =====
    if (!eriManager.startServo(0)) {
        std::cerr << "启动实时控制失败!" << std::endl;
        return;
```

```
    }
    std::cout << "已启动实时控制。" << std::endl;

    // ===== 初始化插补时间参数 =====
    const double T = 2000.0;  // 插补总时间（ms）
    double t = 0.0;           // 当前时间（ms）
    std::copy(startPos, startPos + 6, target); // 初始目标位姿赋值

    // ===== 实时插补主循环（五次多项式）=====
    while (t <= T) {
        for (int i = 0; i < 6; ++i) {
            target[i] = compute5thPoly(startPos[i], goalPos[i], T, t);
        }

        if (eriManager.servoToAPOS(timeStamp, target, false) == -1) {
            std::cerr << "实时目标发送失败!" << std::endl;
            break;
        }

        timeStamp += sampleTime;
        t += sampleTime;
    }

    // ===== 发送最终点，标记为终点 =====
    eriManager.servoToAPOS(timeStamp, target, true);
    Sleep(1000);

    // ===== 释放机器人控制权 =====
    if (eriManager.releaseControlOverRobot())
        std::cout << "控制权释放成功。" << std::endl;
    else
        std::cout << "释放控制权失败。" << std::endl;
}

// ------------------------ 主程序入口 ------------------------
int main() {
    // 设置机器人通信参数（根据实际修改）
    std::string robotIp = "172.31.16.204";
    uint16_t cmdPort = 61210;
    uint16_t servoPort = 61211;
    uint16_t statusPort = 61212;

    // 配置通信
    if (!eriManager.setCommunicationParam(robotIp.c_str(), cmdPort, servoPort, statusPort)) {
        std::cout << "机器人通信参数设置失败!" << std::endl;
        return -1;
    }
    Sleep(50); // 等待通信稳定

    // 用户输入控制界面
    char input = '0';
    while (true) {
        std::cout << "输入指令: (3:开始插补, q:退出): ";
        std::cin >> input;

        if (input == 'q') break;
        if (input == '3') processRealtimeData();
    }

    return 0;
}
```

## 7.1.2 通过笛卡尔实时控制控制机器人画圆

示例代码如下：

```
// ============================================================================
// 示例名称：ERI 实时圆轨迹控制
// 功能：控制机器人 TCP 沿一个平面圆轨迹运动，使用五次多项式进行平滑插值
// 步骤：连接 → 获取控制权 → 参数设置 → MoveJ → 获取起点 → 实时圆轨迹控制 → 终点 → 释放控制权
// ============================================================================

#include "stdafx.h"
#include <iostream>
#include "ERIParamManager.h"

#ifndef M_PI
#define M_PI 3.14159265358979323846
#endif

// ------------------------ ERI 控制器实例 ------------------------
ERIParamManager eriManager;

// ------------------------ 插值函数（五次多项式） ------------------------
double computeInterpolatedTheta(double t, double T, double theta0, double theta1) {
    double s = t / T;
    return theta0 + (theta1 - theta0) * (10 * pow(s, 3) - 15 * pow(s, 4) + 6 * pow(s, 5));
}

// ------------------------ 实时圆轨迹执行函数 ------------------------
void processRealtimeData() {
    // ===== 控制与滤波参数 =====
    const int sampleTime = 4;    // 控制周期（ms）
    const int filterType = 0;    // 滤波器类型（0 = 无滤波）

    // ===== 初始位姿（MoveJ 目标） =====
    double startPos[6] = { 1.224, -0.999, 19.75, 0, 71.249, 1.209 };  // 自定义起点
    double startTCP[6];    // 实际TCP起点（从控制器读取）
    double target[6];      // 每周期的实时目标位置
    UINT64 timeStamp = 0;  // 实时控制时间戳

    // ===== 圆轨迹参数 =====
    const double r = 50.0;         // 圆半径（mm）
    double centerX, centerY, centerZ; // 圆心坐标
    const double theta_0 = 0.0;
    const double theta_1 = 2 * M_PI; // 绕一整圈

    // ===== 插值参数 =====
    const double T = 1000.0 * sampleTime;  // 总时长（ms）
    const double dt = 1.0 * sampleTime;    // 每周期步进（ms）
    double t = 0.0;          // 当前时间

    // ===== 获取机器人控制权 =====
    if (!eriManager.getControlRobot()) {
        std::cerr << "获取机器人控制权失败!" << std::endl;
        return;
    }
    std::cout << "已获取控制权。" << std::endl;

    // ===== 设置实时控制参数 =====
    if (!eriManager.setERIMotionParam(sampleTime, filterType)) {
        std::cerr << "设置实时控制参数失败!" << std::endl;
        return;
    }
```

```
// ===== MoveJ 到初始位置 =====
if (!eriManager.movJ(startPos)) {
    std::cerr << "MoveJ 失败！" << std::endl;
    return;
}
std::cout << "MoveJ 成功。" << std::endl;
Sleep(1000); // 等待稳定

// ===== 获取当前 TCP 位置作为轨迹起点 =====
if (!eriManager.getWorldCpos(startTCP)) {
    std::cerr << "获取当前TCP位置失败!" << std::endl;
    return;
}

// 打印当前 TCP 位置
std::cout << "当前TCP位置：";
for (int i = 0; i < 6; ++i) {
    std::cout << startTCP[i] << " ";
    target[i] = startTCP[i]; // 初始化目标位姿
}
std::cout << std::endl;

// ===== 计算圆心位置（假设圆心在 X 方向左移 r）=====
centerX = startTCP[0] - r;
centerY = startTCP[1];
centerZ = startTCP[2]; // 保持Z不变

// ===== 启动实时控制（CPOS模式）=====
if (!eriManager.startServo(1)) {
    std::cerr << "开启实时控制失败!" << std::endl;
    return;
}
std::cout << "实时控制已开启。" << std::endl;

// ===== 主控制循环：发送轨迹点 =====
while (t <= T) {
    // 插值计算当前角度
    double theta = computeInterpolatedTheta(t, T, theta_0, theta_1);

    // 计算当前圆周点（X-Y 平面圆）
    target[0] = centerX + r * std::cos(theta); // X
    target[1] = centerY + r * std::sin(theta); // Y
    target[2] = centerZ;                       // Z 保持不变

    // 发送实时目标位姿
    if (eriManager.servoToCPOS(timeStamp, target, false) == -1) {
        std::cerr << "实时控制发送失败!" << std::endl;
        break;
    }

    // 更新时间戳与时间变量
    timeStamp += sampleTime;
    t += dt;
}

// ===== 最后一点作为终点 =====
eriManager.servoToCPOS(timeStamp, target, true);
Sleep(1000);

// ===== 释放机器人控制权 =====
if (eriManager.releaseControlOverRobot())
    std::cout << "控制权释放成功。" << std::endl;
else
```

```
        std::cerr << "释放控制权失败。" << std::endl;
}

// ------------------------ 主程序入口 ------------------------
int main() {
    // 设置机器人连接参数
    std::string robotIp = "172.31.16.204";
    uint16_t cmdPort = 61210;
    uint16_t servoPort = 61211;
    uint16_t statusPort = 61212;

    if (!eriManager.setCommunicationParam(robotIp.c_str(), cmdPort, servoPort, statusPort)) {
        std::cerr << "机器人通信参数设置失败!" << std::endl;
        return -1;
    }
    Sleep(50); // 等待通信稳定

    // 用户交互命令行界面
    char input = '0';
    while (true) {
        std::cout << "输入指令: (3: 开始画圆, q: 退出): ";
        std::cin >> input;

        if (input == 'q') break;
        if (input == '3') processRealtimeData();
    }

    return 0;
}
```

## 7.2 示教程序与外部实时数据交替运行

在进行该操作前，请确保示教器处于再现模式，且示教器程序可正常运行。

整体流程：




示例代码如下：

```
Start:
MovJ(G:P0)
MovL(G:P1)
MovL(G:P2)
SetERIProgID(1) //设置执行外部程序ID
EnableERIControl() //控制权切换到外部,程序卡在此处，直到外部程序执行完毕,控制权收回
End;
```

## 7.3 通过SIMIO控制机器人纠偏的使用样例

ERI和外部应用通过虚拟IO接口交互数据，使用到的虚拟接口有SIMAI(100-120)，SIM-DO（100-120）。

其中系统默认使用的端口为：

表 7-1

| 端口 | 说明 |
| --- | --- |
| SIMDO100 | 新数据标志位，1表示有新数据，0表示数据已被处理 |
| SIMAI100 | 基于纠偏坐标系x轴纠偏量 |
| SIMAI101 | 基于纠偏坐标系y轴纠偏量 |
| SIMAI102 | 基于纠偏坐标系z轴纠偏量 |
| SIMAI103 | 基于纠偏坐标系x轴旋转的固定角纠偏量 a |
| SIMAI104 | 基于纠偏坐标系y轴旋转的固定角纠偏量 b |
| SIMAI105 | 基于纠偏坐标系z轴旋转的固定角纠偏量 c |

- **在示教器端编写纠偏指令**

  先在示教器端编写如下程序指令：

```
Start:
EnableERICorrect(“ON”) //开启纠偏后面运动段将会根据纠偏数据进行纠偏
MovJ(L:P0)
MovJ(L;P1)
EnableERICorrect(“OFF”)//关闭纠偏后续运动不会再执行纠偏
Movj(L:P2)
Goto Start
End;
```

- **在客户应用程序中设置SIMIO**

  客户应用程序通过EIP /PN总线通讯与机器人进行数据交互，周期性在SIMIO对应的映射表中写入纠偏数据，实时控制机器人的纠偏运动。

```
 while(true)
{
Int do100Val = getSimDO(100);
If(0 == do100Val) //可发送纠偏数据
{
Int x,y,z,a,b,c;
setSimAI(100,x);
setSimAI(101,y);
setSimAI(102,z);
setSimAI(103,a);
setSimAI(104,b);
setSimAI(105,c);
setSimDO(100,1);//通知ERI服务，纠偏数据已发送完毕
}
```

```
Else
{
 Sleep(1);//等待1毫秒，具体根据实际情况可调整
}
}
```

- **运行示教器程序与客户应用程序**
  示教程序运行到纠偏指令开启纠偏后，机器人走P0与P1时会同步叠加客户应用程序写入的偏移位置，实现实时纠偏运动。

# 附录 A　错误列表

| 错误号 | 错误信息 | 处理方法 |
| --- | --- | --- |
| **88817** | 实时运动中不允许释放控制权 | 等待实时运动结束 |
| **88818** | 未获取控制权不允许movj运动 | 先获取控制前 |
| **88819** | 实时运动中不允许movj运动 | 等待实时运动结束 |
| **88820** | 机器人状态获取失败 | 未注册状态回调函数 |
| **88821** | startServo失败 | 获取控制权后再执行 |

For a Better Life

让 生 活 更 美 好 !


**埃斯顿自动化股份有限公司**

ESTUN AUTOMATION CO., LTD.

南京市江宁经济开发区吉印大道1888号

025-52785866 www.estun.com

025-52785576 info@estun.com