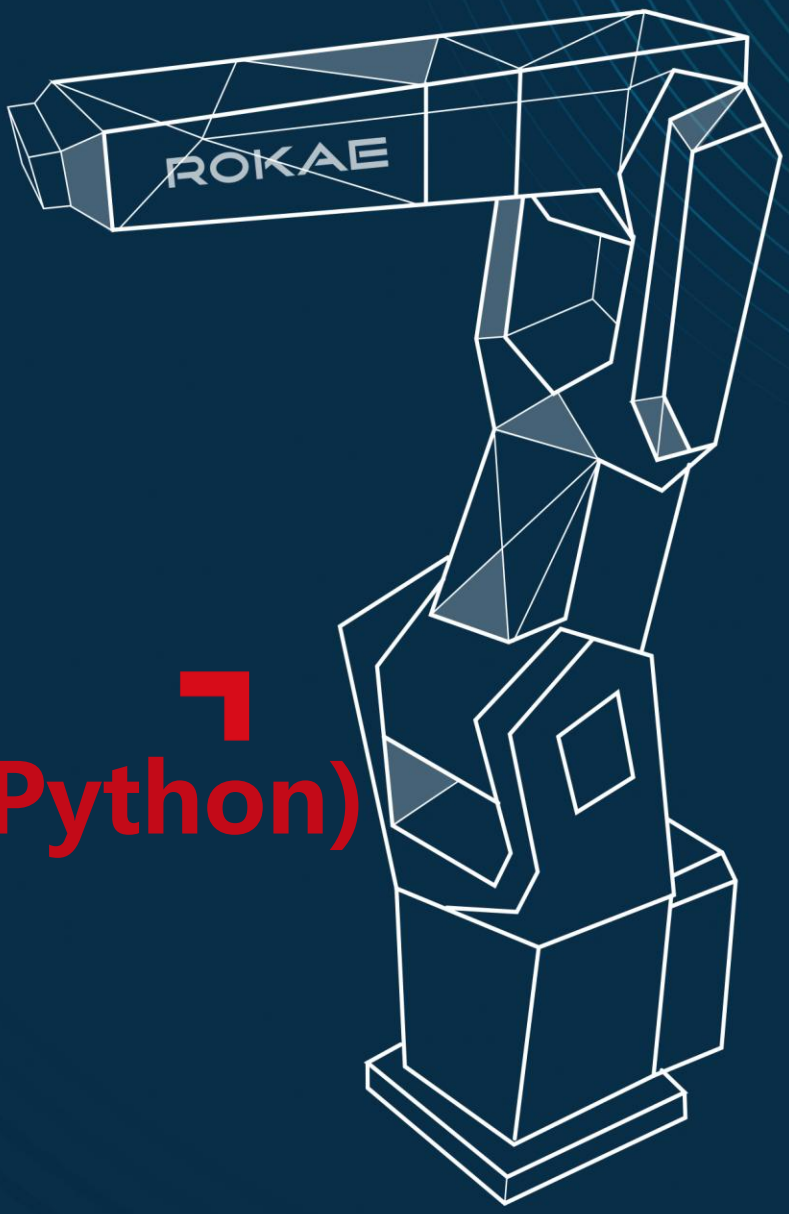
ROKAE 珞石

轻 型 机 器 人 专 家



xCore SDK (Python)

使用手册

让智造更高效

xCore SDK(Python)
使用手册

[类别]

[备注]

控制系统版本：V3.0.1

文档版本：A

本手册中记载的内容如有变更，恕不事先通告。本公司对手册中可能出现的错误均不承担任何责任。

本公司对因使用本手册及其中所述产品而引起的意外或间接伤害均不承担任何责任，敬请谅解。

本公司不可能预见所有的危险和后果，因此本手册不能警告用户所有可能的危险。

如您发现本手册的内容有误或需要改进抑或补充之处，请不吝指正。

本手册的原始语言为中文，所有其他语言版本均翻译自中文版本。

珞石（北京）智能科技有限公司

中国.北京

目录

1 手册概述

1.1 关于本手册

感谢您购买本公司的机器人系统。
本手册记载了正确使用机器人的以下说明：

- 机器人二次开发接口 SDK（Python）的使用。

使用该机器人系统前，请仔细阅读本手册与其他相关手册。
阅读之后，请妥善保管，以便随时取阅。

1.2 手册对象

本手册面向：

- 机器人应用开发工程师。

请务必保证以上人员具备基础的机器人操作、Python 编程等所需的知识，并已接受本公司的相关培训。

1.3 如何阅读产品手册

本手册包含单独的安全章节，必须在阅读安全章节后，才能进行操作作业。

1.4 版本记录

版本编号	日期	说明
V1.6	2022.10	Rokae SDK 初版，适配 xCore 版本 v1.6.1；
V1.7	2023.02	Rokae SDK 正式版本，适配 xCore 版本 v1.7；
V2.0	2023.05	添加部分运动接口
V2.1	2023.11	xCore SDK(v0.1.8)：适配焊接相关接口
V2.2	2024.03	xCore SDK(v0.1.11)：完善焊接相关接口
V2.2	2024.08	xCore SDK(v0.4.1)：新增力控相关接口，移除焊接相关接口
V3.0	2024.12	xCore SDK(v0.5.0)：新增导轨相关接口

2 概述

xCore SDK 编程接口库是珞石机器人提供给客户用于二次开发的软件产品，通过编程接口库，客户可以对配套了 xCore 系统的机器人进行一系列控制和操作，包括实时和非实时的运动控制，机器人通信相关的读写操作，查询及运行 RL 工程，等等。该使用说明书主要介绍编程接口库的使用方法，以及各接口函数的功能。用户可编写自己的应用程序，集成到外部软硬件模块中。

2.1 兼容性

2.1.1 控制器版本和机器人型号

- 控制器版本：xCore v3.0.1 及以后。
- 机器人型号：支持控制所有机型，根据协作和工业机器人支持的功能不同，可调用的接口有所差别。

2.1.2 操作系统及语言要求

操作系统	语言
Ubuntu 18.04/20.04/22.04	默认 Python3.10.X
Windows 10	默认 Python3.12.X

2.2 非实时控制

xCore SDK 提供对机器人的非实时控制，主要通过给机器人发送运动指令，使用控制器内部的轨迹规划，完成路径规划和运动执行。非实时模式提供的操作有：

- 轴空间运动（MoveAbsJ）
- 笛卡尔空间运动（MoveL，MoveJ，MoveC）
- 机器人通信：数字量和模拟量 I/O
- RL 工程的查询
- 其他操作：清除报警，查询控制器日志等等

3 使用指南

本章介绍如何配置并运行一个 xCore SDK Python 程序。
其他语言版本（C++、Java）请参考分手册。

3.1.1 硬件设置

关于机器人本体和控制柜等硬件的设置，请参考《xCore 控制系统使用手册 V3.0.1》。除网络配置外，使用 xCore SDK 无需其他额外的硬件设置。

3.1.2 网络配置

xCore SDK 通过以太网（TCP/IP）连接机器人。通过有线或无线连接皆可，使用户 PC 和机器人连接同一局域网。如果只使用非实时控制，对于网络性能要求不高，可以通过无线连接。
使用实时控制的话推荐通过有线直连到机器人。机器人配置有 2 个网口，一个是外网口，一个是直连网口。直连网口默认静态 IP 地址是 192.168.0.160。连接机器人有两种方式：

- 连接方式 1：机器人与用户 PC 采用网线直连的方式连接。如果用户工控机与机器人不处于同一个网段，需要配置用户 PC 的 IP 使其与机器人静态 IP 地址处于同一个网段，例如 192.168.0.22。
- 连接方式 2：机器人外网口连接路由器或者交换机，用户 PC 也连接路由器或者交换机，两者处于同一局域网。

注：推荐使用方式 1 进行连接，连接方式 2 网络通信质量差时可能会造成机器人运动不稳定现象。

3.1.3 机器人功能设置

无需通过 RobotAssist 进行任何设置，用户可直接用 xCore SDK 控制机器人。

3.1.4 xCore SDK 工程包说明

```
librokae
├── example: 示例程序
├── CHANGELOG.md: 版本变更记录
└── Release: 各操作系统库文件
```

3.1.5 运行设置

1. 根据系统配置将对应的软件包下载到本地；
2. 创建项目后配置路径，将软件包下的 lib 路径加入到工作路径中（sys.path.append(库路径或者文件夹路径)）。

4 接口说明

本章列出各版本 xCore SDK 所支持的接口和功能简述。不同开发语言的版本对接口的功能定义基本一致，但是参数、返回值和调用方法会有区别。本章节介绍主要针对 Python 版本的开发接口，其他语言开发接口请见分手册。

4.1 API 支持

下表是各语言版本接口支持情况概览。

模块	API 功能	C++	Python	Android
rokae::Robot	基本操作	全部支持	全部支持	全部支持
	非实时运动	全部支持	全部支持	全部支持
	Jog 机器人	全部支持	全部支持	不支持
	通信	全部支持	全部支持	部分支持
	RL 工程	全部支持	全部支持	不支持
	协作相关	全部支持	部分支持	全部支持
	焊接相关	部分支持	不支持	不支持
rokae::Model	运动学计算	全部支持	全部支持	不支持
rokae::RtMotionControl	实时模式	全部支持	不支持	不支持
rokae::Planner	上位机路径规划	全部支持	不支持	不支持
rokae::xMateModel	运动学和动力学计算	全部支持（仅 Linux）	不支持	不支持

4.2 Python: 实例化 Robot 类

根据机器人构型和轴数不同，Python 版本的 SDK 提供了下列几个可供实例化的 Robot 类，初始化时会检查所选构型是否和连接的机器人匹配：

类名	适用机型
xMateRobot	协作 6 轴
xMateErProRobot	协作 7 轴
xMateCr5Robot	协作 CR5 轴
StandardRobot	工业 6 轴
PCB4Robot	工业 4 轴
PCB3Robot	工业 3 轴

4.3 机器人基本操作及信息查询

简述	接口	参数	返回
连接机器人	connectToRobot()		
	connectToRobot(remoteIP, localIP)	remoteIP - 机器人 IP 地址 localIP - 本机地址。实时 模式下收发交互数据用	
断开连接	disconnectFromRobot()		
查询机器人基本信息	robotInfo()		控制器版本，机型，轴数
查询上电状态	powerState()		on/off/Estop/Gstop
机器人上下电	setPowerState(state)	state - on/off	
查询当前操作模式	operateMode()		auto/manual
切换手自动模式	setOperateMode(mode)	mode - auto/manual	
查询机器人运行状态	operationState()		idle/jog/RLprogram/moving 等状态
获取当前末端/法兰位姿	posture(ct)	ct – 坐标系类型	[X, Y, Z, Rx, Ry, Rz]
获取当前末端/法兰位姿	cartPosture(ct)	ct – 坐标系类型	[X, Y, Z, Rx, Ry, Rz] 及轴配

			置参数
获取当前关节角度	jointPos()		各轴角度 rad
获取当前关节速度	jointVel()		各轴速率 rad/s
获取关节力矩	jointTorque()		各轴力矩 Nm
查询基坐标系	baseFrame()		[X, Y, Z, A, B, C]
查询当前工具工件组	toolset()		末端坐标系，参考坐标系，负载信息
设置工具工件组	setToolset(toolset)	toolset – 工具工件组信息	
通过 HMI 标定工具设置坐标系	setToolset(toolName, wobjName)	toolName – 工具名称 wobjName - 工件名称	
计算逆解	calcIk(posture)	posture – 末端相对于外部 参考坐标系位姿	关节角度
计算逆解	calcIk(posture,tool_set)	posture – 末端相对于外部 参考坐标系位姿 tool_set - 工具坐标	关节角度
计算正解	calcFk(joints)	joints – 关节角度	末端相对于外部参考坐标系 位姿
计算正解	calcFk(joints, tool_set)	joints – 关节角度 tool_set - 工具坐标	末端位姿
清除伺服报警	clearServoAlarm()		
查询控制器日志	queryControllerLog(count, level)	count - 查询个数 level - 日志等级	控制器日志列表
设置碰撞检测相关参数, 打开碰撞检测功能	enableCollisionDetection (sensitivity, behaviour, fallback)	sensitivity – 灵敏度 behaviour – 碰撞后行为 fallback – 回退距离/柔顺 度	
关闭碰撞检测功能	disableCollisionDetection()		
坐标系标定	calibrateFrame(type, points, is_held, base_aux)	type – 坐标系类型 points – 标定轴角度列表 is_held – 手持/外部工具工 件 base_aux – 基坐标系标定 辅助点	标定结果：坐标系和偏差
获取当前软限位数值	getSoftLimit(limits)	limits - 各轴软限位	已打开/已关闭
设置软限位	setSoftLimit(enable, limits)	enable – 打开/关闭 limits -各轴软限位	
恢复状态	recoverState(item)	item - 恢复选项 1：急停恢复	
设置导轨参数	setRailParameter(name, value)	name – 参数名	

		value – 参数值	
读取导轨参数	getRailParameter(name, value)	name – 参数名 value – 参数值	
配置 NTP	configNtp(server_ip)	server_ip - NTP 服务端 IP	
手动同步一次 NTP 时间	syncTimeWithServer()		
查询 SDK 版本号	sdkVersion()		版本号

4.4 运动控制

非实时模式运动控制相关接口。

简述	接口	参数	返回
设置运动控制模式	setMotionControlMode(mode)	mode - NRT/RT/RL 工程	
重置运动缓存	moveReset()		
机器人开始/继续运动	moveStart()		
停止机器人运动	stop()		
暂停机器人运动	pause()		
添加运动指令	moveAppend(command, id)	command - 一条或多条 MoveL/MoveJ/MoveAbsJ/ MoveC/MoveCF/MoveSP 指令 id – 指令 ID, 用于执行信息反馈	
设置默认运动速度	setDefaultSpeed(speed)	speed - 末端最大线速度	
设置默认转弯区	setDefaultZone(zone)	zone - 转弯区半径	
设置是否使用 conf	setDefaultConfOpt(forced)	forced – 是/否使用	
设置最大缓存指令个数	setMaxCacheSize(number)	number – 个数	
开始 Jog 机器人	startJog(space, rate, step, index, direction)	space - 参考坐标系 rate - 速率 step - 步长 index - XYZABC/J1-7 direction - 方向	
调整速度指令	adjustSpeedOnline(per)	per – 速度百分比	
设置接收事件的回调函数	setEventWatcher(eventType, callback)	eventType – 事件类型 callback – 处理事件的回调 函数	
执行运动指令	executeCommand(command)	command - 一条或多条 MoveL/MoveJ/MoveAbsJ/ MoveC/MoveCF/MoveSP 指令	
读取当前加速度	getAcceleration(acc, jerk)	acc – 加速度 jerk – 加加速度	
设置运动加速度	adjustAcceleration(acc,	acc – 加速度	

	jerk)	jerk – 加加速度	
打开奇异规避功能	setAvoidSingularity(method, enable, threshold)	method – 奇异规避方式 enable – 打开/关闭 threshold – 阈值参数	
查询是否打开规避奇异功能	getAvoidSingularity(method)	method – 奇异规避方式	已打开/已关闭
检验笛卡尔轨迹是否可达，直线轨迹	checkPath(start, start_joint, target)	start - 起始点 start_joint - 起始轴角 target - 目标点	计算出的目标轴角，仅当无错误码时有效
校验多个直线轨迹	checkPath(start_joint, points, target_joint_calculated)	start_joint -起始轴角，单位[弧度] points - 笛卡尔点位，至少需要 2 个点，第一个点是起始点 target_joint_calculated - 若校验通过。返回计算出的目标轴角	若校验失败，返回 points 中出错目标点的下标。其它情况返回 0
检验笛卡尔轨迹是否可达，包括圆弧，全圆	checkPath(start, start_joint, aux, target, angle, rot_type)	start - 起始点 start_joint - 起始轴角 aux - 辅助点 target - 目标点 angle - 全圆执行角度，不等于零时代表校验全圆轨迹 rot_type - 全圆旋转类型	计算出的目标轴角，仅当无错误码时有效

4.5 通信相关

简述	接口	参数	返回值
查询 DI 信号值	getDI(board, port)	board - IO 板序号 port - 信号端口号	on | off
设置 DI 信号值	setDI(board, port, state)	board - IO 板序号 port - 信号端口号 state - 信号值	
查询 DO 信号值	getDO(board, port)	board - IO 板序号 port - 信号端口号	on | off
设置 DO 信号值	setDO(board, port, state)	board - IO 板序号 port - 信号端口号 state - 信号值	
查询 AI 信号值	getAI(board, port)	board - IO 板序号 port - 信号端口号	信号值
设置 AO 信号	setAO(board, port, value)	board - IO 板序号 port - 信号端口号 value - 信号值	
设置输入仿真模式	setSimulationMode(state)	state – 打开/关闭	
读取寄存器值	readRegister(name, index,	name - 寄存器名称	

	value)	index - 寄存器数组索引 value - 读取的数值	
写入寄存器值	writeRegister(name, index, value)	name - 寄存器名称 index - 寄存器数组索引 value - 写入的数值	
设置 xPanel 对外供电模式	setxPanelVout(opt)	opt – 模式	
获取末端按键状态	getKeypadState()		末端按键的状态
使用 CR 和 SR 末端的 485 通信功能	setxPanelRS485(opt, if_rs485)	opt - 对外供电模式 if_rs485 - 接口工作模式，是否打开末端 485 通信	
通过 xPanel 末端读写 modbus 寄存器	XPRWModbusRTUReg(slave_addr, fun_cmd, reg_addr, data_type, num, data_array, if_crc_reverse)	slave_addr - 设备地址 fun_cmd - 功能码 reg_addr - 寄存器地址 data_type - 支持的数据类型 num - 一次连续操作寄存器的个数 data_array - 发送或接收数据的数组 if_crc_reverse - 是否改变 CRC 校验高低位	
通过 xPanel 末端读写 modbus 线圈或离散输入	XPRWModbusRTUCoil(slave_addr, fun_cmd, coil_addr, num, data_array, if_crc_reverse)	slave_addr 设备地址 fun_cmd 功能码 coil_addr 线圈或离散输入寄存器地址 num 一次连续读写线圈离散输入的个数 data_array 发送或接收数据的数组 if_crc_reverse 是否改变 CRC 校验高低位	
通过 xPanel 末端直接传输 RTU 协议裸数据	XPRS485SendData(send_byte, rev_byte, send_data, rev_data)	send_byte 发送字节长度 rev_byte 接收字节长度 send_data 发送字节数据 rev_data 接收字节数据	

4.6 RL 工程

控制器中需要有已创建好的 RL 工程，支持查询工程信息和运行。

简述	接口	参数	返回值
查询 RL 工程列表	projectInfo()		工程名称和任务名
加载工程	loadProject(name, tasks)	name - 工程名称	

		tasks – 任务列表	
pp-to-main	ppToMain()		
开始运行工程	runProject()		
暂停运行工程	pauseProject()		
设置运行速率和循环模式	setProjectRunningOpt(rate, loop)	rate – 运行速率 loop – 循环/单次	
查询工具信息	toolsInfo()		工具名称，位姿，负载等信息
查询工件信息	wobjsInfo()		工件名称，位姿，负载等信息

4.7 协作相关

包括拖动示教和路径录制相关功能。

简述	接口	参数	返回值
打开拖动	enableDrag(space, type, enable_drag_button)	space – 拖动空间 type – 拖动类型 enable_drag_button - 打开拖动功能之后可以直接拖动机器人，不需要按住末端按键	
关闭拖动	disableDrag()		
开始记录拖动路径	startRecordPath(duration)	duration – 记录时间	
停止记录拖动路径	stopRecordPath()		
保存拖动路径	saveRecordPath(name, saveAs)	name – 拖动路径保存名称 saveAs – 如名称存在的命名	
取消记录拖动路径	cancelRecordPath()		
回放路径	replayPath(name, rate)	name – 回放路径名称 rate – 回放速率	
删除保存的拖动路径	removePath(name)	name – 希望删除的路径名称	
查询已保存的拖动路径	queryPathLists()		已保存的路径名称列表
力传感器标定	calibrateForceSensor(all_axes, axis_index)	all_axes – 标定所有轴 axis_index – 单轴标定下标	

4.8 力控指令

简述	接口	参数	返回值
获取当前力矩信息	getEndTorque(ref_type, joint, external, cart_torque, cart_force)	ref_type -力矩相对的参考系 joint - 各轴测量 external -各轴外部力 cart_torque - 笛卡尔空间力矩 cart_force - 笛卡尔空间力	
力控初始化	fcInit(frame_type)	frame_type - 力控坐标系	
开始力控	fcStart()		
停止力控	fcStop()		

设置阻抗控制类型	setControlType(type)	type - 阻抗类型	
设置力控模块使用的负载	setLoad(load)	load - 负载	
设置关节阻抗刚度	setJointStiffness(stiffness)	stiffness - 刚度	
设置笛卡尔阻抗刚度	setCartesianStiffness(stiffness)	stiffness - 刚度	
设置笛卡尔零空间阻抗刚度	setCartesianNullspaceStiffness(stiffness)	stiffness - 刚度	
设置关节期望力矩	setJointDesiredTorque(torque)	torque - 力矩值	
设置笛卡尔期望力/力矩	setCartesianDesiredForce(value)	value -期望力/力矩	
设置绕单轴旋转的正弦搜索运 动	setSineOverlay(line_dir, amplify, frequency, phase, bias)	line_dir - 参考轴 amplify -幅值 frequency - 频率 phase - 相位 bias - 偏置	
设置平面内的莉萨如搜索运动	setLissajousOverlay (int plane, double amplify_one, double frequency_one, double amplify_two, double frequency_two, double phase_diff, error_code &ec)	plane - 参考平面 amplify_one - 一方向幅值 frequency_one - 一方向频率 amplify_two - 二方向幅值 frequency_two - 二方向频率 phase_diff 相位偏差	
开启搜索运动	startOverlay()		
停止搜索运动	stopOverlay()		
暂停搜索运动	pauseOverlay()		
重新开启暂停的搜索运动	restartOverlay()		
设置与接触力有关的终止条件	setForceCondition (range, isInside, timeout)	range - 力限制 isInside - 超出/符合限制条件 时停止等待 timeout - 超时时间	
设置与接触力矩有关的终止条件	setTorqueCondition (range, isInside, timeout)	range - 力矩限制 isInside - 超出/符合限制条件 时停止等待 timeout - 超时时间	
设置与接触位置有关的终止条件	setPoseBoxCondition(supervising_frame, box, isInside, timeout)	supervising_frame - 长方体所在的参考坐标系 box - 长方体	

		isInside - 超出/符合限制条件 时停止等待 timeout - 超时时间	
激活设置的终止条件并等待	waitCondition()		
启动/关闭力控模块保护监控	fcMonitor(enable)	enable - 打开|关闭	
设置力控模式下的轴最大速度	setJointMaxVel(velocity)	velocity – 轴速度	

4.9 错误码和异常

非实时接口的调用结果通过错误码反馈，每个接口都会传入一个错误码 ec，可通过 message(ec)，或者 ec[“message”]来获取错误码对应的信息。
实时模式下发送运动指令、周期调度、读取状态数据等接口在调用过程中会抛出异常。

数值	错误信息	原因及处理方法
0	操作成功完成	无
-1	发生错误	可能由于未识别的错误码，请反馈技术支持人员
-3	机器人急停按钮被按下，请先恢复	恢复急停状态
-16	该操作不允许在当前模式下执行(手动/自动)，请切换到另一个模式	切换手动、自动模式
-17	该操作不允许在当前上下电状态下执行	切换上下电状态
-18	该操作不允许在机器人当前运行状态下执行	机器人非空闲，可能处于拖动/实时模式控制/辨识中等
-19	该操作不允许在当前控制模式(位置/力控)下执行	切换位置控制/力控
-20	机器人运动中	停止机器人运动
-32	计算逆解错误	传入正确位姿
-33	逆解是奇异点	规避奇异点
-34	逆解超出机器人软限位	检查软限位设置时候合适，传入限位内位姿
-35	目标点超出运动范围	传入可达点位
-36	目标点超出运动范围	传入可达点位
-37	单步距离过大，无法计算逆解	传入可达点位
-38	配置数据 cfx 错误逆解无解	检查 conf data
-39	输入数据错误，无法计算逆解	传入可达点位
-40	逆解参考点是奇异点	传入可达点位
-41	算法失效，无法计算逆解	可能由于控制器计算问题，请反馈技术支持人员
-257	通信协议解析失败	请反馈技术支持人员
-258	未找到匹配的数据名称	可能由于控制器和 SDK 版本不适配
-259	未找到匹配的指令名称	可能由于控制器和 SDK 版本不适配
-260	数据值错误，可能是通信协议不匹配	可能由于控制器和 SDK 版本不适配
-272	未找到要查询的数据名称	可能由于控制器和 SDK 版本不适配
-273	数据不可读	可能由于控制器和 SDK 版本不适配
-288	数据不可写	可能由于控制器和 SDK 版本不适配
-289	设置数据失败	设置的数据不合理
-290	设置数据失败	设置的数据不合理
-291	监控数据失败	不会发生

-292	数据已被监控，不支持重复监控	不会发生
-320	指令执行未完成	不会发生
-337	录制的路径过短，数据点不足，请重新录制	增长录制时间重新录制
-338	录制的路径中存在超过关节软限位的情况。请重新录制	在软限位内拖动
-339	录制的路径中存在关节速度过快的情况，请重新录制	放慢拖动动作
-340	未从机器人静止状态开始录制路径	机器人静止时再开始录制
-341	停止录制路径时机器人未停止运动	机器人静止时再停止录制
-342	机器人处于下电状态，保存路径失败	机器人上电
-352	缓冲区不存在有效的路径，请录制	先录制路径再保存
-353	保存路径到磁盘失败	重新录制拖动轨迹，或重启机器人
-513	切换手动/自动操作模式失败	- 停止机器人运动 - 恢复急停 - 关闭拖动
-514	上下电失败	- 恢复急停 - 清除伺服报警
-515	打开/关闭拖动失败，请检查机器人是否处于下电	机器人手动模式下电时打开拖动
-10005	标定中，或标定时重置负载失败	- 等待标定完成 - 正确设置负载
-10030	未打开仿真模式或信号不存在	打开仿真模式后再设置 DI/AI
-10040	开始拖动失败,正确设置负载并标定力矩传感器	正确地设置工具负载质量和质心；设置完成后执行力矩传感器标定
-10051	软限位超出机械限位	设置机械限位内的软限位
-10065	同步 NTP 时间失败	- 安装好 NTP 功能 - 确认服务端 NTP 功能可用，确认 IP 地址正确
-10101	IP 地址非法	检查是否符合 IPv4 地址格式
-10079	力控模块处于错误状态	触发了力控保护，请检查力控模式下机器人状态是否正常，并设置合理的力控保护参数
-10141	负载质量超过了机器人额定负载	使用并设置额定负载范围内的负载
-14010	已绑定为系统 IO	- 取消绑定改信号为系统 IO - 使用其它信号
-14501	DO 信号不存在或为系统输出	检查 DO 信号是否已创建
-17001	读取寄存器错误	- 检查寄存器是否创建 - 是否超数组下标 - 用匹配的数据类型读取
-17002	写寄存器错误	- 检查寄存器是否创建 - 是否超数组下标 - 或用匹配的数据类型写入
-17320	执行过 RL 程序，需要重置运动缓存后再开始运动	调用 moveReset 重置
-17407	开启导轨需关闭安全区域	关闭安全区域功能
-28672	实时模式网络错误	- 本机 IP 地址不能为 localhost 或机器人地址 存在端口占用情况，请检查 RCI 设置-端口
-28688	切换运动控制模式失败	- 停止机器人运动 重启控制器
-28689	该频率不支持	状态数据发送频率支持 1kHZ, 500Hz, 250Hz, 125Hz
-28705	已经开始运动	停止运动后再开始
-28706	无法执行该操作，可能由于机器人未处于空闲状态	停止机器人运动

-28707	起始位置为奇异点	运动机器人到非奇异点
-28708	参数错误	重新设置数据
-28709	机器人处于碰撞停止	上电恢复碰撞状态
-28710	机器人处于急停状态	恢复急停
-28711	请求被拒绝	阻抗控制时多由于力控模型偏差，重新标定力矩传感器并设置正确的负载
-41400	尚有力控指令未完成	等待前序力控指令执行完毕
-41419	TCP 长度超限，力控初始化失败	末端工具长度限制为 0.3m
-41420	未进行力控初始化；或未正确设置负载；或力控模型偏差较大	- 标定零点、力矩传感器； - 设置正确的负载质量和质心； - 检查基坐标系是否正确设置； - 开启拖动时机器人没有收到外力
-41421	停止力控失败，当前不处于力控运行状态	机器人不处于力控状态
-41422	重新开启力控失败	先暂停或停止力控，再重新开启
-41425	非阻抗控制模式，或搜索运动在运行中	- 保证当前处于阻抗模式下，且搜索运动处于停止状态 - 请检查参数设置
-41426	非阻抗控制模式，或搜索运动在运行中	- 保证当前处于阻抗模式下，且搜索运动处于停止状态 - 请检查参数设置
-41427	当前不处于笛卡尔阻抗控制模式或未设置搜索运动	检查控制模式指令，设置搜索运动参数后再尝试
-41428	当前不处于笛卡尔阻抗控制模式或未开始搜索运动	请保证搜索运动已开启且处于笛卡尔阻抗运行状态
-41429	当前未处于笛卡尔阻抗模式或搜索运动未暂停	暂停当前的力控任务并检查控制模式，切换到笛卡尔阻抗控制模式后再尝试
-41430	不支持的阻抗控制类型或力控坐标系	停止当前的力控任务并重新参数初始化
-41431	当前未处于关节阻抗控制模式或未初始化	运行力控前设置轴空间阻抗控制模式
-41432	当前未处于笛卡尔阻抗控制模式或未初始化	运行力控前设置笛卡尔阻抗控制模式
-41433	当前未处于笛卡尔阻抗控制模式或未初始化	检查当前的阻抗控制模式后再进行尝试。确保力值输入正确，且设定笛卡尔阻抗控制模式
-41434	关节期望力超过限制	检查当前的阻抗控制模式后再进行尝试。确保期望力值输入正确，且设定轴空间阻抗控制模式
-41435	笛卡尔空间期望力超过限制	检查当前的阻抗控制模式后再进行尝试。确保期望力值输入正确，且设定笛卡尔阻抗控制模式
-41442	机器人不满足软限位要求，开启力控失败	请确认机器人开启阻抗时机器人在软限位内
-41444	阻抗刚度设置失败，数值不合理或当前状态不能设置刚度	请重新设置阻抗刚度数值在合理范围内
-41448	力控过程中下电，初始化失败	重新上电并执行力控初始化
-41449	力控指令尚未执行完毕，初始化失败	力控指令发送太频繁，请增加退出力控到重新开启之间的时间
-41457	执行过力控停止,需要重新开始	执行力控初始化
-41459	处于力控模式，不允许执行的操作	停止力控后再回放路径
-50000	两条轨迹夹角过小或长度过短，转弯区无法生成	- 增大前后两条轨迹夹角; - 增大前后两条轨迹长度; - 增大转弯区数值
-50001	控制器状态错误，无法生成轨迹	调用 moveReset()重置
-50002	目标点超出运动范围，或为奇异点	- 检查目标点位置; - 用关节方式移动机器人 - 检查 confdata 配置
-50003	相邻两个目标点过近	检查相邻两条运动指令的目标点是否是同一个点

-50004	无法生成圆弧,起始点和目标点距离过近	调整圆弧起点或终点点位之间的距离
-50005	无法生成圆弧，起始点和辅助点距离过近	调整圆弧起点或辅助点点位之间的距离
-50006	无法生成圆弧，辅助点和目标点距离过近	调整圆弧终点或辅助点点位之间的距离
-50007	无法生成圆弧，起始点/辅助点/目标点距离过近	调整圆弧起点、辅助点、终点点位间之间的距离
-50008	无法生成圆弧，点位在一条直线	调整圆弧起点、辅助点、终点点位之间的距离
-50009	无法生成圆弧，半径过小	调整圆弧起点、辅助点、终点点位之间的距离
-50010	无法生成圆弧	调整圆弧起点、辅助点、终点点位之间的距离或方位
-50019	生成轨迹失败	重新调整目标点的位置、姿态和臂角(仅 7 轴机器人需要考虑臂角)
-50021	指定 conf 参数下目标点无解，请检查数值或不设置 confData	- 调用 setDefaultConfOpt(false) 取消 confdata - 重新示教点位，传入正确的 confdata
-50027	轨迹短于最小转弯区半径，自动拼接路径	- 该轨迹需要前后都衔接转弯区，但是轨迹长度小于最小转弯区半径的两倍; - 该轨迹设定衔接一条转弯区，但是轨迹长度小于最小转弯区半径。 该功能可以使得运动更平滑，如想关闭该功能，可将最小转弯区半径设为 0
-50033	机器人处于锁轴状态，目标点锁轴角度发生了偏离，请调整目标点或关闭锁轴状态	调整轨迹目标点或者关闭锁轴指令，锁轴状态下四轴角度要求为 0 度或 180 度或-180 度
-50034	锁轴状态下或 5 轴机型无法到达目标点	6 轴机型在锁轴状态下或 5 轴机型无法到达目标点,请更改点位或关闭锁轴
-50101	轴角度超出运动范围，尝试取消软限位后恢复各轴到允许的范围内	- 取消软限位 - 手动将机器人各轴移动到正常的工作范围内
-50102	存在穿越奇异点的轨迹	- 请规避奇异点 - 重新示教点位,更换目标点; - 将笛卡尔空间运动指令改为关节空间运动指令
-50103	笛卡尔路径终点不符合给定的 ConfData	- 调用 setDefaultConfOpt(false)不使用 confdata - 改为 MoveJ 或 MoveAbsJ - 更改目标点 confdata
-50104	关节力矩超限	- 检查负载数值是否符合实际负载情况 - 检查机器人摩擦力系数、电机过载系数、传动过载系数等参数 - 尝试更换指令形式，例如将笛卡尔空间指令更换为关节空间指令
-50112	位姿有误计算逆解失败	- 重新示教位姿 - 如果当前机型为三轴或者四轴机器人,请检查输入位姿与当前机型特征是否相符 - 如果使用了 setAvoidSingularity(lock4axis)，请检查目标是是否符合锁轴要求
-50113	改变的姿态超过设定的阈值	- 请规避奇异点 - 重新设置奇异规避姿态改变的阈值 - 重新示教点位,更换目标点 - 使用其他方式的奇异规避
-50114	前瞻中轴运动超出运动范围，提前停止运动	- 取消软限位 - 手动将机器人各轴移动到正常的工作范围内
-50115	轨迹前瞻过程中遇到奇异点	- 请规避奇异点 - 重新示教点位,更换目标点 - 更换 Jog 方式，尝试轴空间 Jog - 将笛卡尔空间运动指令改为关节空间运动指令
-50118	改变姿态后求解的终点与所需要终点不一致	- 请规避奇异点

		- 重新示教点位,更换目标点 - 更换奇异规避方式
-50120	奇异规避下搜索路径终点角度失败	- 请规避奇异点 - 重新示教点位,更换目标点 - 更换奇异规避方式 - 如果 Conf 开启，可以关闭 Conf
-50121	奇异规避下搜索路径终点角度失败	- 请规避奇异点 - 重新示教点位,更换目标点 - 更换奇异规避方式
-50204	规划过程中采样点不足，请重新运行	工控机状态不稳定，重新运行或尝试重启
-50205	相邻位置指令相差过大	- 重新示教轨迹目标点位 - 尝试更改指令形式，例如 MoveL 改为 MoveJ
-50208	内部轨迹错误	- 请调用 moveReset()重置运动指令缓存 - 增大或减小转弯区
-50401	检测到碰撞	- 请检查机器人运行环境，确认人员、设备安全后，重新上电，再恢复运行 - 检查当前工具设置是否与实际一致，设定的工具质量质心是否合理
-50501	工具工件坐标系设置失败，可能由于工具或工件不存在	设置已创建的工具工件，且分别为手持和外部
-50512	轴数不匹配	轴空间 Jog 的下标参数不能超出轴数+外部轴数
-50513	速度设置无效，机器人无法运动	传入 0.01~1 范围内的速度参数
-50514	步长设置无效，机器人无法运动	传入大于 0 的步长参数
-50515	参考坐标系设置无效，机器人无法运动	传入支持的坐标系类型参数
-50516	运动轴设置无效，机器人无法运动	按照说明传入 index 参数
-50518	运动失败，目标点可能是当前点	目标点位可能是当前点
-50519	生成轨迹失败，目标点位可能超出机器人工作范围	请在机器人正常工作范围内，重新示教点位
-50525	该机型不支持奇异规避模式运动，或机器人锁轴失败,4 轴角度不为 0 不运行运动，请调整 4 轴角度	调整四轴角度至 0 或者正负 180 度
-60005	RL 工程或指定任务不存在	运行已创建的工程和任务
-60014	控制器当前状态不允许开始运动	确保机器人上电并处于空闲中
-60200	负载信息错误，设置失败	- 负载信息错误,请检查负载重量是否超过额定负载，质心不超过 0.3m - 力控不支持外部工具或手持工件
-60511	路径不存在	使用已保存的回放路径
-60611	正在生成诊断数据不能运动,请等待	等待 10 秒后再次尝试启动
-60702	当前四轴角度不为 0，不允许切换到四轴固定模式，或机型不支持	请检查第四轴当前角度是否为 0°或 180°
-60704	不满足牺牲姿态奇异规避的开启条件	请检查当前状态是否满足牺牲姿态奇异规避开启状态
-60706	不满足轴空间插补奇异规避的开启条件	请检查当前状态是否满足轴空间插补奇异规避开启状态
255	未初始化连接机器人	一般不会发生，正常构造 Robot 类实例即可
256	网络连接错误	- 工控机未开机 - 机器人地址错误，或未处于同一局域网 - 实例类型错误或 SDK 未授权，连接被拒绝
257	无法解析机器人消息，可能由于 SDK 版本与控制器版本不匹配	可能由于控制器和 SDK 版本不适配
258	参数错误,数值超出范围	按照函数说明检查参数数值范围

259	参数错误,参数类型或个数错误	按照函数说明检查参数类型或数组长度
260	不是合法的变换矩阵	检查传入参数是否符合其次变换矩阵要求
261	数组元素个数与机器人轴数不符	传入和机器人轴数一致的数组长度
262	运动控制模式错误,请切换到正确的模式	根据实际情况调用 setMotionControlMode 切换模式
263	超时前未收到机器人回复，可能由于网络通信问题	检查网络连接状态，或反馈技术支持人员
264	重复操作	碰撞检测已打开，需要先关闭
265	通过 UDP 端口接收数据失败，请检查网络及防火墙配置	- 检查本机地址设置是否正确 - 检查防火墙设置，是否允许 UDP 连接
266	客户端校验失败，请检查控制器版本，授权状态和机器人型号	- 按照手册或 README 将控制器升级到匹配版本 - 创建类型匹配的机器人实例 - 联系技术支持人员授权 SDK 功能
272	事件未监听	先调用 setEventWatcher 开始监听数据
273	点位距离过近，坐标系标定失败	参考《xCore 控制系统使用手册》工具工件标定方法，重新标定
512	IP 地址或端口设置错误，或端口被占用	- 本机 IP 地址不能为 localhost 或机器人地址 - 存在端口占用情况，请检查 RCI 设置-端口
513	设置了不支持的字段，或总长度超出限制	支持的字段见 RtSupportedFields，总长度限制为 1024 字节
768	没有可执行的运动指令	先调用 moveAppend 下发运动指令，再开始运动
769	机器人停止中或当前状态无法暂停	可能由于调用 stop()前刚刚手动模式下电，或者按下急停，控制器正在响应停止中，请等待

5 注意事项与问题排查

目前如果使用 xCore SDK 控制机器人，并不会限制通过 RobotAssist 的控制。机器人的一些状态，通过 xCore SDK 更改后也会体现在 Robot Assist 界面上；一些工程运行，运动控制则是分离的。大致总结如下：

会同步更新的组件	● 底部状态栏：手自动，机器人状态，上下电； ● 状态监控窗口； ● 日志上报；
双方可控，即通过 RobotAssist 修改会生效的组件	● RL 工程运行速率和循环/单次模式； ● 非实时模式运动的停止；
不会同步更新的组件，包括但不限于	● 下发的运动指令无法通过点击开始按钮让机器人开始执行； ● 加载的 RL 工程，以及前瞻指针、运动指针等，不会同步显示； ● 所有机器人设置界面显示的功能打开状态和设定值等；

建议的控制方式是单一控制源，避免混淆。对于运动指令，推荐在每次使用 SDK 下发指令之前调用 moveReset() 接口来重置运动缓存。

6 使用示例

本章展示一些 Python 非实时控制示例程序，更多示例请见软件包中 examples。

6.1 非实时接口

6.1.1 示例一：基本操作

```
import xCoreSDK_python
from log import print_log,print_separator

def base_op(robot,ec):
    print_separator("base_op",length=110)
    disconnect(robot,ec)
    connect(robot,ec)
    get_sdk_version(robot)
    get_powerstate(robot,ec)
    set_powerstate(robot,ec)
    get_operatemode(robot,ec)
    set_operatemode(robot,ec)
    get_robotinfo(robot,ec)
    get_operationState(robot,ec)
    get_posture(robot,ec)
    get_cart_posture(robot,ec)
    get_joint_pos(robot,ec)
    get_joint_vel(robot,ec)
    get_joint_torque(robot,ec)
    get_baseframe(robot,ec)
    set_baseframe(robot,ec)
    get_toolset(robot,ec)
    set_toolset(robot,ec)
    set_toolset_by_name(robot,ec)
    clear_servo_alarm(robot,ec)
    calcFk(robot,ec)
    calcIk(robot,ec)
    get_soft_limit(robot,ec)
    set_soft_limit(robot,ec)
    restore_soft_limit(robot,ec)

def disconnect(robot,ec):
    '''断开连接机器人'''
    print_separator("disconnect",length=80)
    robot.disconnectFromRobot(ec)
    print_log("disconnectFromRobot",ec)

def connect(robot,ec):
```

```
    '''连接机器人'''
    robot.connectToRobot(ec)

def get_sdk_version(robot):
    '''获取 sdk 版本'''
    print_separator("get_sdk_version",length=80)
    sdk_version = robot.sdkVersion()
    print(f"sdkVersion={sdk_version}")

def get_powerstate(robot,ec):
    '''获取上电状态'''
    print_separator("get_powerstate",length=80)
    power_state =   robot.powerState(ec)
    print_log("powerState",ec,f"powerState={power_state}")

def set_powerstate(robot,ec):
    '''设置机器人上下电状态，true：上电，false：下电'''
    print_separator("set_powerstate",length=80)
    robot.setPowerState(True,ec)
    print_log("setPowerState",ec)

def get_operatemode(robot,ec):
    '''获取操作模式'''
    print_separator("get_operatemode",length=80)
    operate_mode = robot.operateMode(ec)
    print_log("operateMode",ec,f"operateMode={operate_mode}")

def set_operatemode(robot,ec):
    '''设置操作模式'''
    print_separator("set_operatemode",length=80)
    robot.setOperateMode(xCoreSDK_python.OperateMode.automatic,ec)
    print_log("setOperateMode",ec)

def get_robotinfo(robot,ec):
    '''获取机器人信息'''
    print_separator("get_robotinfo",length=80)
    robot_info = robot.robotInfo(ec)

print_log("robotInfo",ec,f"{robot_info.id,robot_info.version,robot_info.type,robot_info.joint_num}")

def get_operationState(robot,ec):
    '''获取操作状态'''
    print_separator("get_operationState",length=80)
    operation_state = robot.operationState(ec)
```

```
        print_log("operationState",ec,f"operation_state={operation_state}")

def get_posture(robot,ec):
    '''获取当前位姿'''
    print_separator("get_posture",length=80)
    pos = robot.posture(xCoreSDK_python.CoordinateType.endInRef, ec)
    print_log("posture",ec,', '.join(map(str, pos)))

def get_cart_posture(robot,ec):
    '''获取当前笛卡尔坐标信息'''
    print_separator("get_cart_posture",length=80)
    cart_posture = robot.cartPosture(xCoreSDK_python.CoordinateType.endInRef, ec)
    print_log("cartPosture",ec)
    print(f"elbow,{cart_posture.elbow}")
    print(f"hasElbow,{cart_posture.hasElbow}")
    print(f"confData,f{','.join(map(str,cart_posture.confData))}")
    print(f"external size,{len(cart_posture.external)}")
    print(f"trans,{','.join(map(str,cart_posture.trans))}")
    print(f"rpy,{','.join(map(str,cart_posture.rpy))}")
    print(f"pos,{','.join(map(str,cart_posture.pos))}")

def get_joint_pos(robot,ec):
    '''获取关节位置'''
    print_separator("get_joint_pos",length=80)
    joint_pos = robot.jointPos(ec)
    print_log("jointPos",ec,','.join(map(str,joint_pos)))

def get_joint_vel(robot,ec):
    '''获取关节速度'''
    print_separator("get_joint_vel",length=80)
    joint_vel = robot.jointVel(ec)
    print_log("jointVel",ec,','.join(map(str,joint_vel)))

def get_joint_torque(robot,ec):
    '''获取当前关节力矩'''
    print_separator("get_joint_torque",length=80)
    joint_torque = robot.jointTorque(ec)
    print_log("jointTorque",ec,','.join(map(str,joint_torque)))

def get_baseframe(robot,ec):
    '''查询基坐标系'''
    print_separator("get_baseframe",length=80)
    baseframe = robot.baseFrame(ec)
    print_log("baseFrame",ec,','.join(map(str,baseframe)))
```

```
126.
127.  def set_baseframe(robot,ec):
128.      '''设置基坐标系'''
129.      print_separator("set_baseframe",length=80)
130.      frame= xCoreSDK_python.Frame()
131.      robot.setBaseFrame(frame,ec)
132.      print_log("setBaseFrame",ec)
133.
134.  def get_toolset(robot,ec):
135.      '''查询当前工具工件组'''
136.      print_separator("get_toolset",length=80)
137.      toolset = robot.toolset(ec)
138.      print_log("toolset",ec)
139.      print(f"""
140.              load mass: {toolset.load.mass}
141.              load cog: {', '.join(map(str, toolset.load.cog))}
142.              load inertia:{','.join(map(str,toolset.load.inertia))}
143.              end trans:{','.join(map(str,toolset.end.trans))}
144.              end rpy:{','.join(map(str,toolset.end.rpy))}
145.              end pos:{','.join(map(str,toolset.end.pos))}
146.              ref trans:{','.join(map(str,toolset.ref.trans))}
147.              ref rpy:{','.join(map(str,toolset.ref.rpy))}
148.              ref pos:{','.join(map(str,toolset.ref.pos))}
149.                 """)
150.
151.  def set_toolset(robot,ec):
152.      '''设置工具工件组'''
153.      print_separator("set_toolset",length=80)
154.      toolset = xCoreSDK_python.Toolset()
155.      # 设置 toolset 参数
156.      robot.setToolset(toolset, ec)
157.      print_log("setToolset",ec)
158.
159.  def set_toolset_by_name(robot,ec):
160.      '''通过已加载的工程中的工具工件名设置坐标系'''
161.      print_separator("set_toolset_by_name",length=80)
162.      # 调用已设置好的工具工件名
163.      toolset = robot.setToolset("tool0", "wobj0", ec)
164.      print_log("setToolset",ec)
165.      print(f"""
166.              load mass: {toolset.load.mass}
167.              load cog: {', '.join(map(str, toolset.load.cog))}
168.              load inertia:{','.join(map(str,toolset.load.inertia))}
169.              end trans:{','.join(map(str,toolset.end.trans))}
```

```
                end rpy:{','.join(map(str,toolset.end.rpy))}
                end pos:{','.join(map(str,toolset.end.pos))}
                ref trans:{','.join(map(str,toolset.ref.trans))}
                ref rpy:{','.join(map(str,toolset.ref.rpy))}
                ref pos:{','.join(map(str,toolset.ref.pos))}
                """)

def clear_servo_alarm(robot,ec):
    '''清除伺服报警'''
    print_separator("clear_servo_alarm",length=80)
    robot.clearServoAlarm(ec)
    print_log("clearServoAlarm",ec)

def calcFk(robot,ec):
    '''计算正解，关节角度->笛卡尔坐标'''
    print_separator("calcFk",length=80)
    start_angle = [0, 0.557737,-1.5184888, 0,-1.3036738, 0] #  单位弧度
    robot_model = robot.model()
    cart_pose = robot_model.calcFk(start_angle, ec)
    print_log("calcFk",ec)
    print(f"elbow,{cart_pose.elbow}")
    print(f"hasElbow,{cart_pose.hasElbow}")
    print(f"confData,f{','.join(map(str,cart_pose.confData))}")
    print(f"external size,{len(cart_pose.external)}")
    print(f"trans,{','.join(map(str,cart_pose.trans))}")
    print(f"rpy,{','.join(map(str,cart_pose.rpy))}")
    print(f"pos,{','.join(map(str,cart_pose.pos))}")

def calcIk(robot,ec):
    '''计算逆解，笛卡尔坐标 -> 关节角度'''
    print_separator("calcIk",length=80)
    cart_pos = xCoreSDK_python.CartesianPosition([0.614711,0.136,0.416211,-1.57,0,-1.57])
#s4 拖拽位姿
    robot_model = robot.model()
    joint_pos = robot_model.calcIk(cart_pos,ec)
    print_log("calcIk",ec,','.join(map(str,joint_pos)))

def get_soft_limit(robot,ec):
    '''读取当前软限位设置'''
    print_separator("get_soft_limit",length=80)
    soft_limits = xCoreSDK_python.PyTypeVectorArrayDouble2()
    robot.getSoftLimit(soft_limits, ec)
    limits_content = soft_limits.content()
    print_log("soft_limit",ec,limits_content)
```

```
213.
214. def set_soft_limit(robot,ec):
215.     '''设置软限位'''
216.     print_separator("set_soft_limit",length=80)
217.     # 设置软限位需要在下电和手动模式
218.     robot.setPowerState(False, ec)
219.     print_log("setPowerState",ec)
220.     robot.setOperateMode(xCoreSDK_python.OperateMode.manual, ec)
221.     print_log("setOperateMode",ec)
222.     robot.setSoftLimit(False, ec) # 关闭软限位
223.     print_log("setSoftLimit",ec)
224.
225.     # 打开并且设置软限位值
226.     soft_limits = [[-2.0543261909900767, 2.0543261909900767], [-1.356194490192345,
    1.356194490192345], [-1.96705972839036, 1.443460952792061], [-2.0543261909900767,
    2.0543261909900767], [-2.0543261909900767, 2.0543261909900767], [-2.0543261909900767,
    2.0543261909900767]]
227.     robot.setSoftLimit(True, ec,soft_limits)
228.     print_log("setSoftLimit",ec)
229.
230. def restore_soft_limit(robot,ec):
231.     '''还原软限位'''
232.     print_separator("restore_soft_limit",length=80)
233.     # 设置软限位需要在下电和手动模式
234.     robot.setPowerState(False, ec)
235.     print_log("setPowerState",ec)
236.     robot.setOperateMode(xCoreSDK_python.OperateMode.manual, ec)
237.     print_log("setOperateMode",ec)
238.     robot.setSoftLimit(False, ec) # 关闭软限位
239.     print_log("setSoftLimit",ec)
240.
241.     # 打开并且设置软限位值
242.     sr4_default_limit = [[-6.283185307179586, 6.283185307179586], [-2.356194490192345,
    2.356194490192345], [-2.96705972839036, 2.443460952792061], [-6.283185307179586,
    6.283185307179586], [-6.283185307179586, 6.283185307179586], [-6.283185307179586,
    6.283185307179586]] # sr4 默认软限位
243.     soft_limits = sr4_default_limit
244.     robot.setSoftLimit(True, ec,soft_limits)
245.     print_log("setSoftLimit",ec)
246.
247.
248. if __name__ == "__main__":
249.     try:
250.         # 连接机器人
```

```
            # 不同的机器人对应不同的类型
            ip = "10.0.40.129"
            robot = xCoreSDK_python.xMateRobot(ip)
            ec = {}
            base_op(robot,ec)
        except Exception as e:
            print(f"An error occurred: {e}")
```

7 反馈与勘误

文档中若出现不准确的描述或者错误，恳请读者指正批评。如果您在阅读过程中发现任何问题或者有想提出的意见，可以发送邮件到 xuqiu@rokae.com,我们的同事会尽量一一回复。

8 附录 A – Python API

8.1 枚举类型

8.1.1 机器人工作状态 OperationState

Idle = 0	机器人静止
Jog = 1	Jog 机器人中
rtControlling = 2	实时模式控制中
Drag = 3	拖动已开启
rlProgram = 4	RL 工程运行中
Demo = 5	Demo 演示中
dynamicIdentify = 6	动力学辨识中
frictionIdentify = 7	摩擦力辨识中
loadIdentify = 8	负载辨识中
Moving = 9	机器人运动中
Jogging = 10	jog 运动中
Unknown = -1	未知

8.1.2 机型类别 WorkType

industrial	工业机器人
collaborative	协作机器人

8.1.3 机器人操作模式 OperateMode

manual	手动
automatic	自动
unknown	未知(发生异常)

8.1.4 机器人上下电及急停状态 PowerState

On = 0	上电
Off = 1	下电
Estop = 2	急停被按下
Gstop = 3	安全门打开
Unknown = -1	未知(发生异常)

8.1.5 位姿坐标系类型 CoordinateType

flangeInBase	法兰相对于基坐标系
endInRef	末端相对于外部坐标系

8.1.6 运动控制模式 MotionControlMode

Idle	空闲
NrtCommand	非实时模式执行运动指令
NrtRLTask	非实时模式运行 RL 工程
RtCommand	实时模式控制（暂不支持）

8.1.7 机器人停止运动等级 StopLevel

注：目前仅支持 stop2

stop0	快速停止机器人运动后断电
stop1	规划停止机器人运动后断电，停在原始路径上
stop2	规划停止机器人运动后不断电，停在原始路径上
suppleStop	柔顺停止，仅适用于协作机型

8.1.8 机器人拖动模式参数 DragParameter

DragParameterSpace.jointSpace = 0	轴空间拖动
DragParameterSpace.cartesianSpace = 1	笛卡尔空间拖动
DragParameterType.translationOnly = 0	仅平移
DragParameterType.rotationOnly = 1	仅旋转
DragParameterType.freely = 2	自由拖拽

8.1.9 坐标系类型 FrameType

world	世界坐标系
base	基坐标系
flange	法兰坐标系
tool	工具坐标系
wobj	工件坐标系
path	路径坐标系，力控任务坐标系需要跟踪轨迹变化的过程
rail	导轨基坐标系

8.1.10 Jog 选项 - 坐标系 JogOptSpace

world	世界坐标系
flange	法兰坐标系
baseFrame	基坐标系
toolFrame	工具坐标系
wobjFrame	工件坐标系
jointSpace	轴空间
singularityAvoidMode	奇异规避模式，适用于工业六轴，xMateCR 和 xMateSR 六轴机型
baseParallelMode	平行基座模式，适用于工业六轴，xMateCR 和 xMateSR 六轴机型

8.1.11 奇异规避方式 AvoidSingularityMethod

lockAxis4	四轴锁定
wrist	牺牲姿态
jointWay	轴空间短轨迹插补

8.1.12 MoveCF 全圆姿态旋转类型 MoveCFCommandRotType

constPose	不变姿态
rotAxis	动轴旋转
fixedAxis	定轴旋转

8.1.13 xPanel 配置：对外供电模式 xPanelOptVout

off	不输出
reserve	保留
supply12v	输出 12V

supply24v	输出 24V

8.1.14 事件类型 Event

moveExecution	非实时运动指令执行信息
safety	安全 (是否碰撞)
rlExecution	RL 执行状态

8.2 数据结构

8.2.1 机器人基本信息 Info

string id	机器人 uid，可用于区分连接的机器人
string version	控制器版本
string type	机器人机型名称
int joint_num	轴数

8.2.2 坐标系 Frame

list[float] trans	平移量, [X, Y, Z]，单位:米
list[float] rpy	XYZ 欧拉角, [A, B, C]，单位:弧度
list[float] pos	行优先变换矩阵

8.2.3 笛卡尔点位 CartesianPosition

list[float] trans	平移量, [x, y, z]，单位:米
list[float] rpy	XYZ 欧拉角, [A, B, C]，单位:弧度
double elbow	臂角，适用于 7 轴机器人，单位：弧度
bool hasElbow	是否有臂角
list[int] confData	轴配置数据，元素个数应和机器人轴数一致
list[float] external	外部关节角度，单位:弧度

8.2.4 笛卡尔点位偏移量 CartesianPositionOffset

CartesianPositionOffsetType type	类型: Offs/RelTool
Frame frame	偏移坐标

8.2.5 关节点位 JointPosition

list[float] joints	关节角度值，单位:弧度
list[float] external	外部关节角度值，单位:弧度

8.2.6 关节扭矩，不包含重力和摩擦力 Torque

list[float] tau	期望关节扭矩，单位: Nm

8.2.7 负载信息 Load

double mass	负载质量，单位:千克
List[float] cog	质心 [x, y, z]，单位:米
List[float] inertia	惯量 [ix, iy, iz]，单位:千克·平方米

8.2.8 工具工件组信息 Toolset

根据一对工具工件的坐标、负载、机器人手持设置计算得出。

Load load	机器人末端手持负载
Frame end	机器人末端坐标系相对法兰坐标系转换
Frame ref	机器人参考坐标系相对世界坐标系转换

8.2.9 坐标系标定结果 FrameCalibrationResult

Frame frame	标定结果
List[float] errors	样本点与 TCP 标定值的偏差，依次为最小值,平均值,最大值，单位 m

8.2.10 RL 工程信息 RLProjectInfo

string name	工程名称
List[string] taskList	任务名称列表

8.2.11 工具/工件信息 WorkToolInfo

string name	名称
string alias	别名，暂未使用
bool robotHeld	是否机器人手持
Frame pos	位姿。工件的坐标系已相对其用户坐标系变换
Load load	负载

8.2.12 运动指令 MoveAbsJ MoveAbsJCommand

JointPosition target	目标关节点位
int speed	末端线速度，单位 mm/s，关节速度根据末端线速度大小划分几个区间，详见 setDefaultSpeed()
double jointSpeed	关节速度百分比
int zone	转弯区大小
string customInfo	自定义信息，可在运动信息反馈中返回出来

8.2.13 运动指令 MoveJ MoveJCommand

CartesianPosition target	目标笛卡尔点位
int speed	末端线速度，单位 mm/s，关节速度根据末端线速度大小划分几个区间，详见 setDefaultSpeed()
double jointSpeed	关节速度百分比
int zone	转弯区大小
CartesianPosition::Offset offset	偏移量
string customInfo	自定义信息，可在运动信息反馈中返回出来

8.2.14 运动指令 MoveL MoveLCommand

CartesianPosition target	目标笛卡尔点位
int speed	速率
double rotSpeed	空间旋转速度
int zone	转弯区大小
CartesianPosition ::Offset offset	偏移量
string customInfo	自定义信息，可在运动信息反馈中返回出来

8.2.15 运动指令 MoveC MoveCCommand

CartesianPosition target	目标笛卡尔点位

CartesianPosition aux	辅助点位
int speed	速率
double rotSpeed	空间旋转速度
int zone	转弯区大小
CartesianPositionOffset targetOffset	目标点偏移量
CartesianPositionOffset auxOffset	辅助点偏移量
string customInfo	自定义信息，可在运动信息反馈中返回出来

8.2.16 运动指令 MoveCF MoveCFCommand

CartesianPosition target	目标笛卡尔点位
CartesianPosition aux	辅助点位
int speed	速率
double rotSpeed	空间旋转速度
int zone	转弯区大小
double angle	全圆执行角度，单位：弧度
CartesianPosition ::Offset targetOffset	目标点偏移量
CartesianPosition ::Offset auxOffset	辅助点偏移量
RotType rotType	全圆姿态旋转模式
string customInfo	自定义信息，可在运动信息反馈中返回出来

8.2.17 运动指令 MoveSP MoveSPCommand

CartesianPosition target	终点笛卡尔点位
CartesianPositionOffset targetOffset	偏移选项
double radius	初始半径，单位：米
double radius_step	每旋转单位角度，半径的变化，单位：米/弧度
double angle	合计旋转角度，单位：弧度
bool direction	旋转方向, true - 顺时针 | false - 逆时针
string customInfo	自定义信息，可在运动信息反馈中返回出来

8.2.18 运动停留指令 MoveWait MoveWaitCommand

duration_	停留时长，最小有效时长 1ms

8.2.19 控制器日志信息 LogInfo

Int id	日志 ID 号
string timestamp	日期及时间
string content	日志内容
string repair	修复办法

8.2.20 末端按键状态 KeyPadState

bool key1_state	CR1 号
bool key2_state	CR2 号
bool key3_state	CR3 号
bool key4_state	CR4 号
bool key5_state	CR5 号
bool key6_state	CR6 号
bool key7_state	CR7 号

8.3 方法

8.3.1 机器人基本操作及信息查询

connectToRobot() [1/2]
def connectToRobot(self, ec: dict) 建立与机器人的连接。机器人地址和端口号为创建 Robot 实例时传入的
参数 [out] ec： 错误码

connectToRobot() [2/2]
def connectToRobot(self, remoteIP: str, localIP: str = '') 连接到机器人
参数 remoteIP 机器人 IP 地址 localIP 本机地址。实时模式下收发交互数据用，可不设置；PCB3/4 轴机型不支持

disconnectFromRobot()
def disconnectFromRobot(self, ec: dict) 断开与机器人连接。断开前会停止机器人运动，请注意安全
参数 [out] ec： 错误码

robotInfo()
def robotInfo(self, ec: dict) -> Info 查询机器人基本信息
参数 [out] ec ：错误码 **返回** 机器人基本信息：控制器版本，机型，轴数

powerState()
def powerState(self, ec: dict) -> PowerState 机器人上下电以及急停状态
参数 [out] ec ：错误码 **返回** 0：on-上电 | 1：off-下电 |2： estop-急停 | 3：gstop-安全门打开

setPowerState()
def setPowerState(self, on: bool, ec: dict) 机器人上下电。注：只有无外接使能开关或示教器的机器人才能手动模式上电。
参数 [in] on： true-上电 | false-下电 [out] ec ：错误码

operateMode()
def operateMode(self, ec: dict) -> OperateMode 查询机器人当前操作模式
参数 [out] ec： 错误码 **返回** 0：mannual-手动 | 1：automatic-自动

setOperateMode()
def setOperateMode(self, mode: OperateMode, ec: dict) 切换手自动模式
参数 [in] mode： manual：手动 | automatic：自动 [out] ec： 错误码

operationState()
def operationState(self, ec: dict) -> OperationState 查询机器人当前运行状态 (空闲,运动中，拖动开启等)
参数 [out] ec： 错误码

返回	运行状态枚举类

posture()	
def posture(self, ct: CoordinateType, ec: dict) -> List[float] 获取机器人法兰或末端的当前位姿	
参数	[in] ct 坐标系类型 1) flangeInBase: 法兰相对于基坐标系; 2) endInRef: 末端相对于外部参考坐标系。例如,当设置了手持工具及外部工件后，该坐标系类型返回的是工具相对于工件坐标系的坐标。 [out] ec 错误码
返回	[X, Y, Z, Rx, Ry, Rz]，其中平移量单位为米旋转量单位为弧度

cartPosture()	
def cartPosture(self, ct: CoordinateType, ec: dict) -> CartesianPosition 获取机器人法兰或末端的当前位姿	
参数	[in] ct 坐标系类型 [out] ec 错误码
返回	当前笛卡尔位置

jointPos()	
def jointPos(self, ec: dict) -> List[float] 机器人当前轴角度，机器人本体+外部轴，单位：弧度，外部轴导轨单位米	
参数	[out] ec：错误码
返回	轴角度值

jointVel()	
def jointVel(self, ec: dict) -> List[float] 机器人当前关节速度，机器人本体+外部轴，单位：弧度/秒,外部轴单位米/秒	
参数	[out] ec： 错误码
返回	关节速度

jointTorque()	
def jointTorque(self, ec: dict) -> List[float] 关节力传感器数值，单位: Nm	
参数	[out] ec： 错误码
返回	力矩值

baseFrame()	
def baseFrame(self, ec: dict) -> List[float] 用户定义的基坐标系，相对于世界坐标系	
参数	[out] ec ：错误码
返回	数组, [X, Y, Z, A, B, C]，其中平移量单位为米旋转量单位为弧度

toolset()	
def toolset(self, ec: dict) -> Toolset 查询当前工具工件组信息	
注解	此工具工件组仅为 SDK 运动控制使用，不与 RL 工程相关.
参数	[out] ec： 错误码
返回	返回当前工具组信息

setToolset() [1/2]	
def setToolset(self, toolset: Toolset, ec: dict) 设置工具工件组信息	
注解	此工具工件组仅为 SDK 运动控制使用，不与 RL 工程相关. 除此接口外，如果通过 RobotAssist 更改默认工具工件(右上角的选项)，该工具工件组也会相应更改.
参数	[in] toolset：工具工件组信息 [out] ec： 错误码

setToolset() [2/2]	
def setToolset(self, toolName: str, wobjName: str, ec: dict) -> Toolset 通过工程中的工具工件，或全局工具工件名称，来进行工具工件坐标系设置。	
注解	设置前提: 已加载一个 RL 工程，且创建了工具和工件。否则，只能设置为默认的工具工件，即"tool0"和"wobj0"。一组工具工件无法同时为手持或外部；如果有冲突，以工具的位置为准，例如工具工件同时为手持，不会返回错误，但是工件的坐标系变成了外部
参数	[in] toolName：工具名称 [in] wobjName: 工件名称 [out] ec： 错误码
返回	设置后的工具工件组信息。当发生错误设置失败时，返回 Toolset 类型初始化默认值

calcIk() [1/2]	
def calcIk(self, posture : CartesianPosition, ec: dict) -> list[float] 根据位姿计算逆解	
参数	[in] posture： 法兰末端位姿,相对于基坐标系 [out] ec ：错误码
返回	轴角度数组（单位:弧度）

calcIk() [2/2]	
def calcIk(self, posture : CartesianPosition, toolset: Toolset, ec: dict) -> list[float] 根据位姿计算逆解	
参数	[in] posture： 法兰末端位姿 [in] toolset : 工具工件组信息 [out] ec ：错误码
返回	轴角度数组（单位:弧度）

calcFk() [1/2]	
def calcFk(self, joints: list[float], ec: dict) -> CartesianPosition 根据轴角度计算正解	
参数	[in] joints： 轴角度，单位：弧度 [out] ec：错误码
返回	机器人末端位姿，相对于外部参考坐标系

calcFk() [2/2]	
def calcFk(self, joints: list[float],toolset: Toolset, ec: dict) -> CartesianPosition 根据轴角度计算正解	
参数	[in] joints： 轴角度，单位：弧度 [in] toolset : 工具工件组信息 [out] ec：错误码
返回	机器人末端位姿，相对于外部参考坐标系

calibrateFrame ()	
def calibrateFrame(self, type: FrameType, points: list[float], is_held: bool, ec: dict, base_aux: list[float]) -> FrameCalibrationResult 坐标系标定 (N 点标定)	
注解	各坐标系类型支持的标定方法及注意事项： 1) 工具坐标系: 三点/四点/六点标定法 2) 工件坐标系: 三点标定。标定结果不会相对用户坐标系做变换，即，若为外部工件，返回的结果是相对于基坐标系的。 3) 基坐标系: 六点标定。标定前请确保动力学约束和前馈已关闭。 若标定成功(无错误码)，控制器会自动保存标定结果，重启控制器后生效。
参数	[in] points 轴角度列表，列表长度为 N。例如，使用三点法标定工具坐标系，应传入 3 组轴角度。轴角度的单位是弧度。 [in] is_heldtrue - 机器人手持 | false - 外部。仅影响工具/工件的标定 [out] ec 错误码 [in] base_aux 基坐标系标定时用到的辅助点，单位[米]
返回	标定结果，当错误码没有被置位时，标定结果有效

clearServoAlarm()	
def clearServoAlarm(self, ec: dict) 清除伺服报警	
参数	[out] ec：错误码，当有伺服报警且清除失败的情况下错误码置为-1

enableCollisionDetection()	
def enableCollisionDetection(self, sensitivity: list[float], behaviour: StopLevel, fallback_compliance: float, ec: dict) 设置碰撞检测相关参数，打开碰撞检测功能	
参数	[in] sensitivity 碰撞检测灵敏度，范围 0.01-2.0 [in] behaviour 碰撞后机器人行为，支持 stop1(安全停止, stop0 和 stop1 处理方式相同)和 stop2(触发暂停）, suppleStop(柔顺停止) [in] fallback_compliance 1) 碰撞后行为是安全停止或触发暂停时，该参数含义是碰撞后回退距离，单位: 米 2) 碰撞后行为是柔顺停止时，该参数含义是柔顺度，范围 [0.0, 1.0] [out] ec 错误码

disableCollisionDetection()	
def disableCollisionDetection(self, ec: dict) 关闭碰撞检测功能	
参数	[out] ec 错误码

getSoftLimit()	
def getSoftLimit(self, limits: PyTypeVectorArrayDouble2, ec: dict) -> bool 获取当前软限位数值	
参数	[out] limits各轴软限位 [下限，上限]，单位: 弧度 [out] ec 错误码
返回	true - 已打开 | false - 已关闭

setSoftLimit()	
def setSoftLimit(self, enable: bool, ec: dict, limits: list[float]) 设置软限位。软限位设定要求： 1) 打开软限位时，机械臂应下电且处于手动模式; 2) 软限位不能超过机械硬限位 3) 机械臂当前各轴角度应在设定的限位范围内	
参数	[in] enable true - 打开 | false - 关闭。 [out] ec 错误码 [in] limits各轴[下限，上限]，单位：弧度。 1) 当 limits 为默认值时，视为仅打开软限位不修改数值; 不为默认值时，先修改软限位再打开 2) 关闭软限位时不会修改限位数值

queryControllerLog()	
def queryControllerLog(self, count: int, level: set[LogInfoLevel], ec: dict) -> list[LogInfo] 查询控制器最新的日志	
参数	[in] count查询个数，上限是 10 条 [in] level 指定日志等级，空集合代表不指定 [out] ec 错误码
返回	日志信息

recoverState()	
def recoverState(self, item: int, ec: dict) 根据选项恢复机器人状态	
参数	[in] item 恢复选项，1：急停恢复 [out] ec 错误码

sdkVersion()	
def sdkVersion() -> str 查询 RokaeSDK 版本	
返回	版本号

setRailParameter()	
def setRailParameter(name, value, ec) 设置导轨参数	
参数	[in] name 参数名，见 value 说明 [in] value 参数 | 参数名 | 数据类型 开关 | enable | bool 基坐标系 | baseFrame | Frame 导轨名称 | name | str 编码器分辨率 | encoderResolution | int 减速比 | reductionRatio | float 电机最大转速(rpm) | motorSpeed | int 软限位(m), [下限,上限] | softLimit | list[float] 运动范围(m), [下限,上限] | range | list[float] 最大速度(m/s) | maxSpeed | float 最大加速度（m/s^2) | maxAcc | float 最大加加速度(m/s^3) | maxJerk | float [out] ec 错误码

getRailParameter()	
def getRailParameter(name, value, ec) 设置导轨参数	
模板参数	参数类型
参数	[in] name 参数名，见 setRailParameter() [out] value 参数数值，见 setRailParameter() [out] ec 错误码，参数名不存在或数据类型不匹配返回错误码

syncTimeWithServer()	
def syncTimeWithServer(ec) 手动同步一次时间，远端 IP 是通过 configNtp 配置的。耗时几秒钟，阻塞等待同步完成，接口预设的超时时间是 12 秒	
参数	[out] ec 错误码，参数名不存在或数据类型不匹配返回错误码

8.3.2 运动控制（非实时模式）

setMotionControlMode()	
def setMotionControlMode(self, mode: MotionControlMode, ec: dict) 设置运动控制模式	
注解	在调用各运动控制接口之前，须设置对应的控制模式。
参数	[in] mode 模式 [out] ec 错误码

moveReset()	
def moveReset(self, ec: dict) 运动重置，清空已发送的运动指令，清除执行信息	
注解	Robot 类在初始化时会调用一次运动重置。RL 程序和 SDK 运动指令切换控制，需要先运动重置。
参数	[out] ec 错误码

pause()	
def pause(self, ec: dict) 停止机器人运动	
注解	同 stop()
参数	[out] ec： 错误码

stop()	
def stop(self, ec: dict) 停止机器人运动	
注解	目前支持 stop2 停止类型，规划停止不断电，参见 StopLevel。 调用此接口后，已经下发的运动指令会被清除，不再执行。
参数	[out] ec： 错误码

moveStart()	
def moveStart(self, ec: dict) 开始/继续运动	
参数	[out] ec 错误码

moveAppend() [1/2]	
def moveAppend(self, cmds: list[Command], cmdID: PyString, ec: dict) 添加单条或多条运动指令，添加后调用 moveStart()开始运动	
模板参数	Command 运动指令类：MoveJCommand | MoveAbsJCommand | MoveLCommand | MoveCCommand | MoveCFCommand | MoveSPCommand
参数	[in] cmds 指令列表，允许的个数为 1-100，须为同类型的指令 [out] cmdID 本条指令的 ID，可用于查询指令执行信息 [out] ec 错误码，仅反馈指令发送前的错误，包括：1) 网络连接问题；2) 指令个数不符；

moveAppend() [2/2]	
def moveAppend(self, cmd: Command, cmdID: PyString, ec: dict) 添加单条运动指令，添加后调用 moveStart()开始运动	
模板参数	Command 运动指令类：MoveJCommand | MoveAbsJCommand | MoveLCommand | MoveCCommand | MoveCFCommand | MoveSPCommand
参数	[in] cmd 运动指令 [out] cmdID 本条指令的 ID，可用于查询指令执行信息 [out] ec 错误码，仅反馈指令发送前的错误，包括：1) 网络连接问题；2) 指令个数不符；

executeCommand()	
def executeCommand(self, cmds: list[Command], ec: dict) 执行单条或多条运动指令，调用后机器人立刻开始运动	
模板参数	Command 运动指令类：MoveJCommand | MoveAbsJCommand | MoveLCommand | MoveCCommand | MoveCFCommand | MoveSPCommand;
参数	[in] cmds 指令列表，允许的个数为 1-1000 [out] ec 错误码，仅反馈执行前的错误，包括：1) 网络连接问题；2) 指令个数不符；3) 机器人当前状态下无法运动，例如没有上电

setDefaultSpeed()	
def setDefaultSpeed(self, speed: int, ec: dict) 设定默认运动速度，初始值为 100	
注解	该数值表示末端最大线速度(单位 mm/s)，自动计算对应关节速度
参数	[in] speed：该接口不对参数进行范围限制。末端线速度的实际有效范围分别是 5-4000(协作), 5-7000(工业)。 关节速度百分比划分为 5 个的范围: < 100 : 10% ； 100 ~ 200 : 30% ； 200 ~ 500 : 50% ； 500 ~ 800 : 80% ； > 800 : 100% [out] ec：错误码

setDefaultZone()	
def setDefaultZone(self, zone: int, ec: dict) 设定默认转弯区	
注解	该数值表示运动最大转弯区半径(单位:mm)，自动计算转弯百分比. 若不设置，则为 0 (fine，无转弯区)
参数	[in] zone：该接口不对参数进行范围限制。转弯区半径大小实际有效范围是 0-200。 转弯百分比划分 4 个范围: < 1 : 0 (fine) ； 1 ~ 20 : 10% ； 20 ~ 60 : 30% ； > 60 : 100% [out] ec：错误码

setDefaultConfOpt()	
def setDefaultConfOpt(self, forced: bool, ec: dict) 设置是否使用轴配置数据(confData)计算逆解。初始值为 false	
参数	[in] forced true -使用运动指令的 confData 计算笛卡尔点位逆解，如计算失败则返回错误; false - 不使用，逆解时会选取机械臂当前轴角度的最近解 [out] ec 错误码

setMaxCacheSize()	
def setMaxCacheSize(self, number: int, ec: dict) 设置最大缓存指令个数，指发送到控制器待规划的路径点个数，允许的范围[1,300]，初始值为 30。	
注解	如果轨迹多为短轨迹，可以调大这个数值，避免因指令发送不及时导致机器人停止运动(停止后如果有未执行的指令，可 moveStart()继续;
参数	[in] number 个数 [out] ec 错误码

adjustSpeedOnline()	
def adjustSpeedOnline(self, scale: float, ec: dict) 动态调整机器人运动速率，非实时模式时生效。	
参数	[in] scale 运动指令的速度的比例，范围 0.01 - 1。当设置 scale 为 1 时，机器人将以路径原本速度运动。 [out] ec 错误码

getAcceleration()
def getAcceleration(self, acc: float, jerk: float, ec: dict) 读取当前加/减速度和加加速度
参数 [out] acc 系统预设加速度的百分比 [out] jerk 系统预设的加加速度的百分比 [out] ec 错误码

adjustAcceleration()
def adjustAcceleration(self, acc: float, jerk: float, ec: dict) 调节运动加/减速度和加加速度。如果在机器人运动中调用，当前正在执行的指令不生效，下一条指令生效
参数 [in] acc 系统预设加速度的百分比，范围[0.2, 1.5]，超出范围不会报错，自动改为上限或下限值 [in] jerk 系统预设的加加速度的百分比，范围[0.1, 2]，超出范围不会报错，自动改为上限或下限值 [out] ec 错误码

setEventWatcher()
def setEventWatcher(self, eventType: Event, callback: typing.Callable[[dict], None], ec: dict) 设置接收事件的回调函数
参数 [in] eventType 事件类型 [in] callback 处理事件的回调函数。说明: 1) 对于 Event::moveExecution，回调函数在同一个线程执行，请避免函数中有执行时间较长的操作; 2) Event::safety 则每次独立线程回调，没有执行时间的限制 [out] ec 错误码

queryEventInfo()
queryEventInfo(self, eventType: Event, ec: dict) -> dict 查询事件信息。与 setEventWatcher()回调时的提供的信息相同，区别是这个接口是主动查询的方式
参数 [in] eventType 事件类型 [out] ec 错误码 **返回** 事件信息

startJog()
def startJog(self, space: JogOptSpace, rate: float, step: float, index: int, direction: bool, ec: dict) 开始 jog 机器人，需要切换到手动操作模式。
注解 调用此接口并且机器人开始运动后，无论机器人是否已经自行停止，都必须调用 stop()来结束 jog 操作，否则机器人会一直处于 jog 的运行状态。 **参数** [in] space jog 参考坐标系。 1) 工具/工件坐标系使用原则同 setToolset(); 2) 工业六轴机型和 xMateCR/SR 六轴机型支持两种奇异规避方式 Jog：Space::singularityAvoidMode, Space::baseParallelMode 3) CR5 轴机型支持平行基座模式 Jog：Space::baseParallelMode [in] rate 速率，范围 0.01 - 1 [in] step 步长。单位: 笛卡尔空间-毫米 | 轴空间-度。步长大于 0 即可，不设置上限，如果机器人机器人无法继续 jog 会自行停止运动。 [in] index 根据不同的 space，该参数含义如下： 1) 世界坐标系,基坐标系,法兰坐标系,工具工件坐标系: 0~5 分别对应 X, Y, Z, Rx, Ry, Rz 2) 轴空间: 关节序号，从 0 开始计数 3) 奇异规避模式,平行基座模式: a) 6 轴机型：0~5 分别对应 X, Y, Z, J4(4 轴), Ry, J6(6 轴);

b) 5 轴机型：0~4 分别对应 X, Y, Z, Ry, J5(5 轴)

[in] direction 根据不同的 space 和 index，该参数含义如下：

1) 奇异规避模式 J4: true - ±180° | false - 0°;

2) 平行基座模式 J4 & Ry: true - ±180° | false - 0°

3) 其它，true - 正向 | false - 负向

[out] ec 错误码

setAvoidSingularity()

def setAvoidSingularity(self, method: AvoidSingularityMethod, enable: bool, threshold: float, ec: dict)

打开/关闭奇异点规避功能。只适用于部分机型:

1) 四轴锁定: 支持工业六轴，xMateCR 和 xMateSR 六轴机型；
2) 牺牲姿态: 支持所有六轴机型；
3) 轴空间插补: 支持工业六轴机型

参数

[in] method 奇异规避方式

[in] enable true - 打开功能 | false - 关闭。对于四轴锁定方式，打开之前要确保 4 轴处于零位。

[in] limit 不同的规避方式，该参数含义分别为:

1) 牺牲姿态: 允许的姿态误差，范围 (0, PI*2]，单位弧度

2) 轴空间插补: 规避半径，范围[0.005, 10]，单位米

3) 四轴锁定: 无参数

[out] ec 错误码

getAvoidSingularity()

def getAvoidSingularity(self, method: AvoidSingularityMethod, ec: dict) -> bool

查询是否处于规避奇异点的状态

参数

[in] method 奇异规避的方式

[out] ec 错误码

返回

true - 已打开 | false – 已关闭

checkPath() [1/3]

checkPath(self, start: CartesianPosition, start_joint: list[float], target: CartesianPosition, ec: dict) -> list[float]

检验笛卡尔轨迹是否可达，直线轨迹。支持导轨，返回的目标轴角为轴数+外部轴数

参数

[in] start 起始点

[in] start_joint 起始轴角 [弧度]

[in] target 目标点

[out] ec 错误码

返回

计算出的目标轴角，仅当无错误码时有效

checkPath() [2/3]

def checkPath(self, start_joint: list[float], points: list[CartesianPosition], target_joint_calculated: PyTypeVectorDouble, ec: dict) -> int

校验多个直线轨迹

[in] start_joint 起始轴角 [弧度]

[in] points 笛卡尔点位，至少需要 2 个点，第一个点是起始点

[in] target_joint_calculated 若校验通过。返回计算出的目标轴角

[out] ec 错误码

返回

若校验失败，返回 points 中出错目标点的下标。其它情况返回 0

checkPath() [3/3]

def checkPath(self, start: CartesianPosition, start_joint: list[float], aux: CartesianPosition, target: CartesianPosition, ec: dict, angle: float, rot_type: MoveCFCommandRotType) -> list[float]

检验笛卡尔轨迹是否可达，包括圆弧，全圆。支持导轨，返回的目标轴角为轴数+外部轴数

参数	[in] start 起始点
	[in] start_joint 起始轴角 [弧度]
	[in] aux 辅助点
	[in] target 目标点
	[out] ec 错误码
	[in] angle 全圆执行角度，不等于零时代表校验全圆轨迹
	[in] rot_type 全圆旋转类型
返回	计算出的目标轴角，仅当无错误码时有效

8.3.3 通信相关

getDI()
def getDI(self, board: int, port: int, ec: dict) -> bool 查询数字量输入信号值
参数 [in] board：IO 板序号 [in] channel：信号端口号 [out] ec：错误码 **返回** True -开 | False -关 | None-该接口不存在

getDO()
def getDO(self, board: int, port: int, ec: dict) -> bool 查询数字输出量信号值
参数 [in] board：IO 板序号 [in] channel：信号端口号 [out] ec：错误码 **返回** True -开 | False -关 | None-该接口不存在

setDO()
def setDO(self, board: int, port: int, state: bool, ec: dict) 设置数字量输出信号值
参数 [in] board：IO 板序号 [in] port： 信号端口号 [in] state：True-开 | False-关 [out] ec：错误码

getAI()
def getAI(self, board: int, port: int, ec: dict) -> float 读取模拟量输入信号值
参数 [in] boardIO 板序号 [in] port 信号端口号 [out] ec 错误码 **返回** 信号值

setAO()
def setAO(self, board: int, port: int, value: float, ec: dict) 设置模拟量输出信号

参数	[in] boardIO 板序号 [in] port 信号端口号 [in] value 输出值 [out] ec 错误码

readRegister()
def readRegister(self, name: str, index: int, value: T, ec: dict) 读取寄存器值。可读取单个寄存器，寄存器数组，或按索引读取寄存器数组。 如果要读取整个寄存器数组，value 传入对应类型的 vector，index 值被忽略。
模板参数 T 读取数值类型 **参数** [in] name 寄存器名称 [in] index 按索引读取寄存器数组中元素，从 0 开始。 下列两种情况会报错：1) 索引超出数组长度; 2) 寄存器不是数组但 index 大于 0 [out] value 寄存器数值，允许的类型有 bool/int/float [out] ec 错误码

writeRegister()
def writeRegister(self, name: str, index: int, value: T, ec: dict) 写寄存器值。可写入单个寄存器，或按索引写入寄存器数组中某一元素。
模板参数 T 写入数值类型 **参数** [in] name 寄存器名称 [in] index 数组索引，从 0 开始。 下列两种情况会报错：1) 索引超出数组长度; 2) 寄存器不是数组但 index 大于 0 [in] value 写入的数值 [out] ec 错误码

setxPanelVout()
def setxPanelVout(self, opt: int, ec: dict) 设置 xPanel 对外供电模式。注：仅部分机型支持 xPanel 功能，不支持的机型会返回错误码
参数 [in] opt 模式 [out] ec 错误码

setxPanelRS485()
def setxPanelRS485(self, opt: int, if_rs485: bool, ec: dict) 使用 CR 和 SR 末端的 485 通信功能，需要修改末端的参数配置，可通过此接口进行参数配置
参数 [in] opt 对外供电模式，0：不输出，1：保留，2：12v，3：24v [in] if_rs485 接口工作模式，是否打开末端 485 通信 t [out] ec 错误码

XPRWModbusRTUReg()
def XPRWModbusRTUReg(self, slave_addr: int, fun_cmd: int, reg_addr: int, data_type: str, num: int, data_array: list[int], if_crc_reverse: bool, ec: dict) 通过 xPanel 末端读写 modbus 寄存器
参数 in] slave_addr 设备地址 0-65535 [in] fun_cmd 功能码 0x03 0x04 0x06 0x10 [in] reg_addr 寄存器地址 0-65535 [in] data_type 支持的数据类型 int32、int16、uint32、uint16

	in] num 一次连续操作寄存器的个数 0-3，类型为 int16/uint16 时，最大为 3；类型为 int32/uint32、float 时，最大为 1，功能码为 0x06 时，此参数无效 [in/out] data_array 发送或接收数据的数组，非 const，功能码为 0x06 时，只使用此数组的数据[0],此时 num 的值无效，除了 0x06 功能码，大小需要与 num 匹配 in] if_crc_reverse 是否改变 CRC 校验高低位，默认 false，少数厂家末端工具需要反转 [out] ec 错误码

XPRWModbusRTUCoil()	
def XPRWModbusRTUCoil(self, slave_addr: int, fun_cmd: int, coil_addr: int, num: int, data_array: list[bool], if_crc_reverse: bool, ec: dict) 通过 xPanel 末端读写 modbus 线圈或离散输入	
参数	in] slave_addr 设备地址 0-65535 [in] fun_cmd 功能码 0x01 0x02 0x05 0x0F [in] coil_addr 线圈或离散输入寄存器地址 0-65535 [in] num 一次连续读写线圈离散输入的个数（0-48），功能码 0x05 时，此值无效 [in/out] data_array 发送或接收数据的数组，非 const，功能码为 0x05 时，只使用此数组的数据[0],此时 num 的值无效，除了 0x05 功能码，大小需要与 num 匹配 [in] if_crc_reverse 是否改变 CRC 校验高低位，默认 false，少数厂家末端工具需要反转 [out] ec 错误码

XPRS485SendData()	
def XPRWModbusRTUReg(self, slave_addr: int, fun_cmd: int, reg_addr: int, data_type: str, num: int, data_array: list[int], if_crc_reverse: bool, ec: dict) 通过 xPanel 末端直接传输 RTU 协议裸数据	
参数	in] send_byte 发送字节长度 0-16 [in] rev_byte 接收字节长度 0-16 [in] send_data 发送字节数据 数组长度需要和 send_byte 参数一致 [out] rev_data 接收字节数据 数组长度需要和 rev_byte 参数一致 [out] ec 错误码

setSimulationMode()	
def setSimulationMode(self, state: bool, ec: dict) 设置输入仿真模式	
参数	[in] state true - 打开 | false - 关闭 [out] ec 错误码

getKeypadState()	
def getKeypadState(self, ec: dict) -> KeyPadState 获取末端按键状态，不支持的机型会返回错误码	
参数	[out] ec 错误码
返回	末端按键的状态。末端按键编号见《xCore 机器人控制系统使用手册》末端把手的图示。

8.3.4 RL 工程

projectsInfo()	
def projectsInfo(self, ec: dict) -> list[RLProjectInfo] 查询工控机中 RL 工程名称及任务	
参数	[out] ec 错误码

返回	工程信息列表，若没有创建工程则返回空列表

loadProject()	
def loadProject(self, name: str, tasks: list[str], ec: dict) 加载工程	
参数	[in] name 工程名称 [in] tasks 要运行的任务。该参数必须指定，不能为空，否则无法执行工程。 [out] ec 错误码

ppToMain()	
def ppToMain(self, ec: dict) 程序指针跳转到 main。调用后，等待控制器解析完工程后返回，阻塞时间视工程大小而定，超时时间设定为 10 秒。	
参数	[out] ec 错误码。错误码能提供的信息有限，不能反馈如 RL 语法错误、变量不存在等错误。可通过 queryControllerLog()查询错误日志。

runProject()	
def runProject(self, ec: dict) 开始运行当前加载的工程	
参数	[out] ec 错误码

pauseProject()	
def pauseProject(self, ec: dict) 暂停运行工程	
参数	[out] ec 错误码

setProjectRunningOpt()	
def setProjectRunningOpt(self, rate: float, loop: bool, ec: dict) 更改工程的运行速度和循环模式	
参数	[in] rate 运行速率，范围 0.01 - 1 [in] loop true - 循环执行 | false - 单次执行 [out] ec 错误码

toolsInfo()	
def toolsInfo(self, ec: dict) -> list[WorkToolInfo] 查询当前加载工程的工具信息	
参数	[out] ec：错误码
返回	工具信息列表，若未加载任何工程或没有创建工具，则返回默认工具 tool0 的信息

wobjsInfo()	
def wobjsInfo(self, ec: dict) -> list[WorkToolInfo] 查询当前加载工程的工件信息	
参数	[out] ec：错误码
返回	工件信息列表，若未加载任何工程或没有创建工件，则返回空 list。

8.3.5 协作相关

enableDrag()
def enableDrag(self, space: int, type: int, ec: dict) 打开拖动
参数 [in] space：拖动空间. 轴空间拖动仅支持自由拖拽类型 [in] type：拖动类型 [out] ec：错误码

disableDrag()
def disableDrag(self, ec: dict) 关闭拖动
参数 [out] ec：错误码

startRecordPath()
def startRecordPath(self, duration: int, ec: dict) 开始录制路径
参数 [in] duration 路径的时长，单位:秒，范围 1~1800.此时长只做范围检查用，到时后控制器不会停止录制，需要调用 stopRecordPath()来停止 [out] ec 错误码

stopRecordPath()
def stopRecordPath(self, ec: dict) 停止录制路径，若录制成功(无错误码)则路径数据保存在缓存中
参数 [out] ec 错误码

cancelRecordPath()
def cancelRecordPath(self, ec: dict) 取消录制，缓存的路径数据将被删除
参数 [out] ec 错误码

saveRecordPath()
def saveRecordPath(self, name: str, ec: dict, saveAs: str = '') 保存录制好的路径
参数 [in] name 路径名称 [out] ec 错误码 [in] saveAs 重命名，可选参数。 如果已录制好一条路径但没有保存，则用该名字保存路径。如果没有未保存的路径，则将已保存的名为"name"的路径重命名为"saveAs"

replayPath()
def replayPath(self, name: str, rate: float, ec: dict) 运动指令-路径回放 和其它运动指令类似，调用 replayPath 之后，需调用 moveStart 才会开始运动。
参数 [in] name 要回放的路径名称 [in] rate 回放速率，应小于 3.0, 1 为路径原始速率。注意当速率大于 1 时，可能产生驱动器无法跟随错误 [out] ec 错误码

removePath()

def removePath(self, name: str, ec: dict, removeAll: bool = False) 删除已保存的路径	
参数	[in] name 要删除的路径名称 [out] ec 错误码。若路径不存在，错误码不会被置位 [in] removeAll 是否删除所有路径，可选参数，默认为否

queryPathLists()	
def queryPathLists(self, ec: dict) -> list[str] 查询已保存的所有路径名称	
参数	[out] ec 错误码
返回	名称列表，若没有路径则返回空列表

8.3.6 力控指令

forceControl()	
def forceControl(self) 力控指令类	
返回	ForceControl_T

getEndTorque()	
def getEndTorque(self, ref_type: FrameType, joint_torque_measured: PyTypeVectorDouble, external_torque_measured: PyTypeVectorDouble, cart_torque: PyTypeVectorDouble, cart_force: PyTypeVectorDouble, ec: dict) 获取当前力矩信息	
参数	[in] ref_type 力矩相对的参考系： 1) FrameType::world - 末端相对世界坐标系的力矩信息; 2) FrameType::flange - 末端相对于法兰盘的力矩信息; 3) FrameType::tool - 末端相对于 TCP 点的力矩信息 [out] joint_torque_measured 各轴测量力 [out] external_torque_measured 各轴外部力 [out] cart_torque 笛卡尔空间力矩 [X, Y, Z]，单位 Nm [out] cart_force 笛卡尔空间力 [X, Y, Z]，单位 N [out] ec 错误码

fcInit()	
def fcInit(self, frame_type: FrameType, ec: dict) 力控初始化	
参数	[in] frame_type 力控坐标系，支持 world/wobj/tool/base/flange。工具工件坐标系使用 setToolset()设置的坐标系 [out] ec 错误码

fcStart()	
def fcStart(self, ec: dict) 开始力控，fcInit()之后调用。 如需在力控模式下执行运动指令，fcStart()之后可执行。 注意，如果在 fcStart()之前通过 moveAppend()下发了运动指令但未开始运动，fcStart 之后就会执行这些运动指令。	
参数	[out] ec 错误码

fcStop()	
def fcStop(self, ec: dict) 停止力控	
参数	[out] ec 错误码

setControlType()
def setControlType(self, type: int, ec: dict) 设置阻抗控制类型
参数 [in] type 0 - 关节阻抗 | 1 - 笛卡尔阻抗 [out] ec 错误码

setLoad()
def setLoad(self, load: Load, ec: dict) 设置力控模块使用的负载信息，fcStart()之后可调用。
参数 [in] load 负载 [out] ec 错误码

setJointStiffness()
def setJointStiffness(self, stiffness: list[float], ec: dict) 设置关节阻抗刚度。fcInit()之后调用生效 各机型的最大刚度不同，请参考《xCore 控制系统手册》 SetJntCtrlStiffVec 指令的说明
参数 [in] stiffness 各轴刚度 [out] ec 错误码

setCartesianStiffness()
def setCartesianStiffness(self, stiffness: list[float], ec: dict) 设置笛卡尔阻抗刚度。fcInit()之后调用生效 各机型的最大刚度不同，请参考《xCore 控制系统手册》 SetCartCtrlStiffVec 指令的说明
参数 [in] stiffness 依次为：X Y Z 方向阻抗力刚度[N/m], X Y Z 方向阻抗力矩刚度[Nm/rad] [out] ec 错误码

setCartesianNullspaceStiffness()
def setCartesianNullspaceStiffness(self, stiffness: float, ec: dict) 设置笛卡尔零空间阻抗刚度。fcInit()之后调用生效
参数 [in] stiffness 范围[0,4]，大于 4 会默认设置为 4，单位 Nm/rad [out] ec 错误码

setJointDesiredTorque()
def setJointDesiredTorque(self, torque: list[float], ec: dict) 设置关节期望力矩。fcStart()之后可调用
参数 [in] torque 力矩值，范围[-30,30]，单位 Nm [out] ec 错误码

setCartesianDesiredForce()
def setCartesianDesiredForce(self, value: list[float], ec: dict) 设置笛卡尔期望力/力矩。fcStart()之后可调用
参数 [in] value 依次为: X Y Z 方向笛卡尔期望力，范围[-60,60]，单位 N; X Y Z 方向笛卡尔期望力矩，范围[-10,10]，单位 Nm [out] ec 错误码

setSineOverlay()
def setSineOverlay(self, line_dir: int, amplify: float, frequency: float, phase: float, bias: float, ec: dict) 设置绕单轴旋转的正弦搜索运动。 设置阻抗控制类型为笛卡尔阻抗(即 setControlType(1))之后, startOverlay()之前调用生效。 各机型的搜索运动幅值上限和搜索运动频率上限不同，请参考《xCore 控制系统手册》 SetSineOverlay 指令的说明。
参数 [in] line_dir 搜索运动参考轴: 0 - X | 1 - Y | 2 - Z

	[in] amplify 搜索运动幅值，单位 Nm [in] frequency 搜索运动频率，单位 Hz [in] phase 搜索运动相位，范围[0, PI]，单位弧度 [in] bias 搜索运动偏置，范围[0, 10]，单位 Nm [out] ec 错误码

setLissajousOverlay()	
def setLissajousOverlay(self, plane: int, amplify_one: float, frequency_one: float, amplify_two: float, frequency_two: float, phase_diff: float, ec: dict) 设置平面内的莉萨如搜索运动 设置阻抗控制类型为笛卡尔阻抗(即 setControlType(1))之后, startOverlay()之前调用生效。	
参数	[in] plane 搜索运动参考平面: 0 - XY | 1 - XZ | 2 - YZ [in] amplify_one 搜索运动一方向幅值，范围[0, 20]，单位 Nm [in] frequency_one 搜索运动一方向频率，范围[0, 5]，单位 Hz [in] amplify_two 搜索运动二方向幅值，范围[0, 20]单位 Nm [in] frequency_two 搜索运动二方向频率，范围[0, 5]，单位 Hz [in] phase_diff 搜索运动两个方向相位偏差，范围[0, PI]，单位弧度 [out] ec 错误码

startOverlay()	
def startOverlay(self, ec: dict) 开启搜索运动。fcStart()之后调用生效 搜索运动为前序设置的 setSineOverlay()或 setLissajousOverlay()的叠加	
参数	[out] ec 错误码

stopOverlay()	
def stopOverlay(self, ec: dict) 停止搜索运动	
参数	[out] ec 错误码

pauseOverlay()	
def pauseOverlay(self, ec: dict) 暂停搜索运动。startOverlay()之后调用生效	
参数	[out] ec 错误码

restartOverlay()	
def restartOverlay(self, ec: dict) 重新开启暂停的搜索运动。pauseOverlay()之后调用生效。	
参数	[out] ec 错误码

setForceCondition()	
def setForceCondition(self, range: list[float], isInside: bool, timeout: float, ec: dict) 设置与接触力有关的终止条件	
参数	[in] range 各方向上的力限制 { X_min, X_max, Y_min, Y_max, Z_min, Z_max }，单位 N。 设置下限时，负值表示负方向上的最大值；设置上限时，负值表示负方向上的最小值。 [in] isInside true - 超出限制条件时停止等待; false - 符合限制条件时停止等待 [in] timeout 超时时间，范围[1, 600]，单位秒 [out] ec 错误码

setTorqueCondition()
def setTorqueCondition(self, range: list[float], isInside: bool, timeout: float, ec: dict)

设置与接触力矩有关的终止条件	
参数	[in] range 各方向上的力矩限制 { X_min, X_max, Y_min, Y_max, Z_min, Z_max }，单位 Nm。 设置下限时，负值表示负方向上的最大值；设置上限时，负值表示负方向上的最小值。 [in] isInside true - 超出限制条件时停止等待; false - 符合限制条件时停止等待 [in] timeout 超时时间，范围[1, 600]，单位秒 [out] ec 错误码

setPoseBoxCondition()	
def setPoseBoxCondition(self, supervising_frame: Frame, box: list[float], isInside: bool, timeout: float, ec: dict) 设置与接触位置有关的终止条件	
参数	[in] supervising_frame 长方体所在的参考坐标系，相对于外部工件坐标系。 外部工件坐标系是通过 setToolset()设置的 (Toolset.ref) [in] box 定义一个长方体 { X_start, X_end, Y_start, Y_end, Z_start, Z_end }，单位米 [in] isInside true - 超出限制条件时停止等待; false - 符合限制条件时停止等待 [in] timeout 超时时间，范围[1, 600]，单位秒 [out] ec 错误码

waitCondition()	
def waitCondition(self, ec: dict) 激活前序设置的终止条件并等待，直到满足这些条件或者超时	
参数	[out] ec 错误码

fcMonitor()	
def fcMonitor(self, enable: bool, ec: dict) 启动/关闭力控模块保护监控。 设置监控参数后，不立即生效，调用 fcMonitor(true)后开始生效，并且一直保持，直到调用 fcMotion(false)后结束。 结束后保护阈值恢复成默认值，即仍然会有保护效果，关闭监控后不再是用户设置的参数。	
参数	[in] enable true - 打开 | false - 关闭 [out] ec 错误码
参见	setCartesianMaxVel() setJointMaxVel() setJointMaxMomentum() setJointMaxEnergy()

setJointMaxVel()	
def setJointMaxVel(self, velocity: list[float], ec: dict) 设置力控模式下的轴最大速度	
参数	[in] velocity 轴速度 [rad/s]，范围 >=0 [out] ec 错误码

setJointMaxMomentum()	
def setJointMaxMomentum(self, momentum: list[float], ec: dict) 设置力控模式下轴最大动量。 计算方式：F*t，可以理解为冲量，F 为力矩传感器读数，t 为控制周期，如果超过 30 个周期都超过动量阈值则触发保护	
参数	[in] momentum 动量 [N·s]，范围 >=0 [out] ec 错误码

setJointMaxEnergy()	
def setJointMaxEnergy(self, energy: list[float], ec: dict) 设置力控模式下轴最大动能。 计算方式：F*v，可以理解为功率，F 为力矩传感器读数，v 为关节速度，如果超过 30 个周期都超过动能阈值则触发保护	

参数	[in] energy 动能 [N·rad/s]，范围 >=0 [out] ec 错误码

setCartesianMaxVel()
def setCartesianMaxVel(self, velocity: list[float], ec: dict) 设置力控模式下，机械臂末端相对基坐标系的最大速度
参数 [in] velocity 依次为：X Y Z [m/s], A B C [rad/s], 范围 >=0 [out] ec 错误码

ROKAE 珞石

轻 型 机 器 人 专 家

400-010-8700

北 京 总 部：北京市海淀区农科院西路6号海青大厦A座7层
山东分公司：济宁市邹城市中心店镇机电产业园恒丰路888号
苏州分公司：苏州工业园区星湖街328号创意产业园1-A1F
深圳分公司：深圳市宝安区中粮福安机器人智造产业园10栋1楼