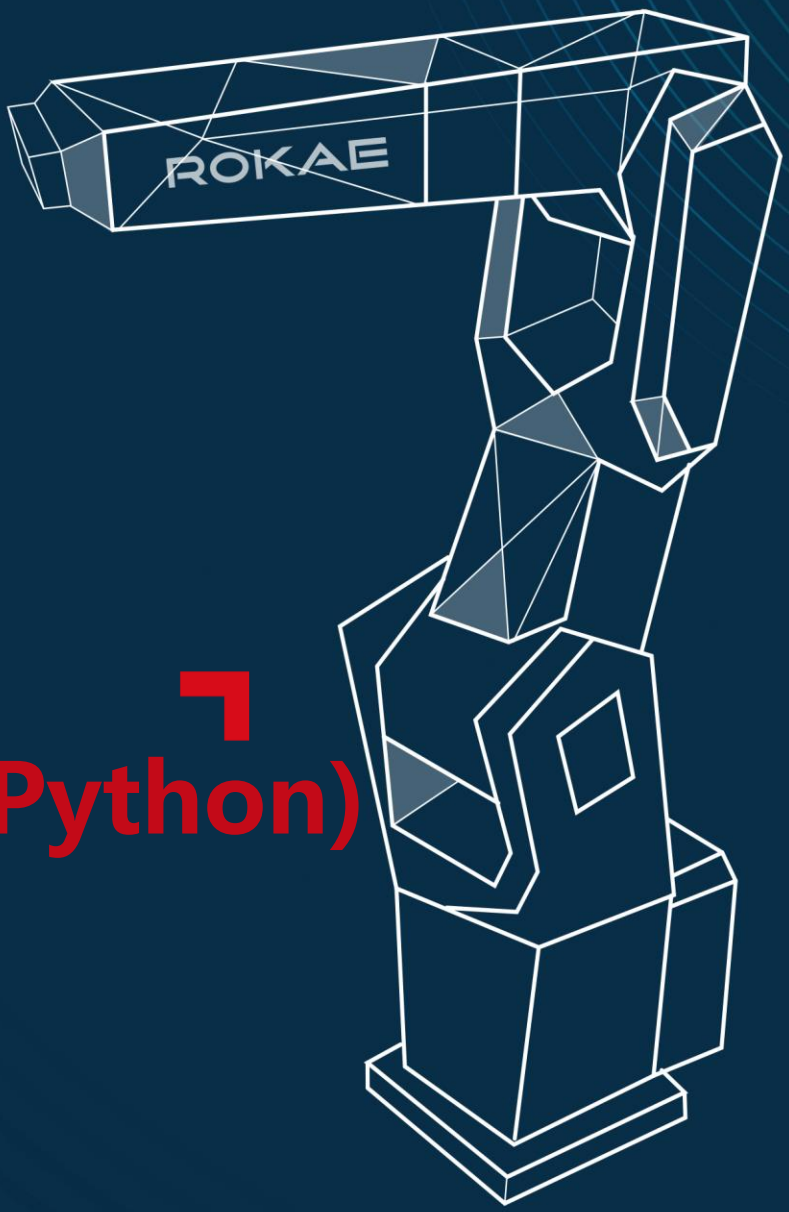
ROKAE 珞石
轻 型 机 器 人 专 家



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：V2.1

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

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 v1.7 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 Python3.8.X
Windows 10	默认 Python3.8.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
- 焊接相关操作
- 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 机器人控制系统使用手册 V2.1》。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
├── doc: 文档和使用手册
├── example: 示例程序
├── readme: 版本变更记录
└── lib: 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(lib 文件路径)）。

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	不支持	不支持
	通信	全部支持	部分支持	部分支持
	RL 工程	全部支持	不支持	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	全部支持	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
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()		
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
获取法兰当前位姿	flangePos()		[X, Y, Z, A, B, C]
获取工具当前位姿	tcpPos()		[X, Y, Z, A, B, C]
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	setToolName(toolName, wobjName)	toolName – 工具名称 wobjName - 工件名称	
计算逆解	calcIk(posture)	posture – 法兰末端位姿	关节角度
计算正解	calcFk(joints)	joints – 关节角度	法兰末端位姿
清除伺服报警	clearServoAlarm()		
查询 SDK 版本号	sdkVersion()		版本号
查看当前点位运行位置	getPointPos()		当前运动点位序号
路径点的可达性/奇异性/相邻点校验	pathCheck(start_point, start_point_joint, end_point)	start_point – 起点笛卡尔位姿 start_point_joint – 起点关节轴角 end_point – 终点的笛卡尔位姿	终点的计算关节轴角
力传感器一键标定	calibForceSensor()		

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
重置运动缓存	moveReset()		
机器人开始运动	moveStart()		
停止机器人运动	stop()		
暂停机器人运动	pause()		
添加运动命令缓存	executeCommand()	values 一条或多条相同类型 MoveL/MoveJ/MoveAbsJ/MoveC 指令	
添加运动命令缓存	executeDiffCommand()	values 一条或多条不同类型 MoveL/MoveJ/MoveAbsJ/MoveC 指令	
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	
调整速度指令	adjustSpeedOnline(per)	per – 速度百分比	
查询运动指令执行错误码	lastErrorCode()		错误码

4.5 通信相关

简述	接口	参数	返回值
查询 DI 信号值	getDI(board, channel)	board - IO 板序号 channel - 信号端口号	on | off
查询 DO 信号值	getDO(board, channel)	board - IO 板序号 channel - 信号端口号	on | off
设置 DO 信号值	setDO(board, channel, state)	board - IO 板序号 channel - 信号端口号	

		state - 信号值	

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)	space – 拖动空间 type – 拖动类型	
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

4.8 焊接相关

包括摆动开关及设置摆动所需要的参数。

简述	接口	参数	返回值
开始摆动	weaveOn()		
关闭摆动	weaveOff()		
设置焊接摆动所需参数	setWeaveParams(amplitude, frequency, left_dwell_time, right_dwell_time, mid_dwell_time)	amplitude – 幅值 frequency – 频率 left_dwell_time – 左侧停留时间 right_dwell_time – 右侧停留时间 mid_dwell_time – 中部停留时间	

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)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。

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：信息查询，拖动，机器人运动等

```python
from robot import *
from convert_tools import *
import time


def waitRobot(robot):
    running = True
    while running:
        time.sleep(0.1)
        ec = {}
        st = robot.operationState(ec)
        if st == rokae.OperationState.idle.value or st == rokae.OperationState.unknown.value:
            running = False


def main():
    ip = "192.168.66.128"
    ec = {}
    robot = xMateRobot(ip)

    with robot:
        # 连接机器人
        robot.connectToRobot(ec)
        # 设置机器人上下电状态-上电
        robot.setPowerState(True, ec)
        # 查询机器人状态
        power = robot.powerState(ec)
        print("当前上下电状态为:", power)
        time.sleep(2)
        # 设置机器人上下电状态-下电
        robot.setPowerState(False, ec)
        power = robot.powerState(ec)
        print("当前上下电状态为:", power)

        ############ 2. 查询信息 ######################
        # 获取机器人的基本信息
        info = robot.robotInfo(ec)
        print("机器人轴数:", info["joint_num"], "机型:", info["type"], "控制器版本:", info["version"])
```

```
39.     # 获取 SDK 版本
40.     print("SDK 版本:", robot.sdkVersion(ec))
41.     # 获取机器人的上下电状态
42.     power = robot.powerState(ec)
43.     print("当前上下电状态为:", power)
44.     # 获取机器人的操作模式
45.     mode = robot.operateMode(ec)
46.     print("当前机器人的操作模式为:", mode)
47.     # 获取机器人运行状态
48.     state = robot.operationState(ec)
49.     print("当前机器人的运行状态为:", state)
50.
51.     ######## 3. 获取机器人当前位姿，轴角度，基坐标系等信息 ##########
52.     # 获取关节位置
53.     joint_pos = robot.jointPos(ec)
54.     print("当前关节位置:", joint_pos)
55.     # 获取关节速度
56.     joint_vel = robot.jointVel(ec)
57.     print("当前关节速度:", joint_vel)
58.     # 获取关节力矩
59.     joint_torque = robot.jointTorque(ec)
60.     print("当前关节力矩:", joint_torque)
61.     # 获取法兰位姿
62.     posture = robot.flangePos(ec)
63.     print("当前法兰位姿:", posture)
64.     # 获取基坐标系-----原 model()类
65.     base = robot.baseFrame(ec)
66.     print("当前基坐标系:", base)
67.     # 获取当前的工具坐标系
68.     toolset = robot.toolset(ec)
69.     print("当前的工具坐标系为:", toolset)
70.     # 设置新的坐标系
71.     coor_new = {'end': {'rot': [1, -1, 1], 'trans': [0.0, 0.0, 0.0]}, 'load': {'cog': [0.0, 0.0, 0.0],
72.                                                                                 'inertia': [0.0, 0.0, 0.0], 'mass': 0.0},
73.                 'ref': {'rot': [0.0, -0.0, 0.0], 'trans': [0.0, 0.0, 0.0]}}
74.     robot.setToolset(coor_new, ec)
75.     # 获取当前的工具坐标系
76.     toolset = robot.toolset(ec)
77.     print("修改后的工具坐标系为:", toolset)
78.     zero = zeroToolset()
79.     robot.setToolset(zero, ec)
80.
81.     ################## 4. 计算正解和逆解 ######################
82.     # 计算正解-> 输入一个与当前机型轴数相同的 List，返回一个当前位姿的 list
```

```
        point = [10, 20, 30, 40, 50, 10]
        point = degree2rad(point)
        print(point)
        fk = robot.calcFK(point, ec)
        print("计算正解为:", fk)

        # 计算逆解->输入一个位姿，返回一个轴角的list
        pos = [0.06323804204745481, -0.4863339081678629, 0.4460326936303843, 3.1228803161697467, -0.09853314560134144,
            0.12170264387228817]
        ik = robot.calcIK(pos, ec)
        # ik = rad2degree(ik)
        print("计算逆解为:", ik)

        ################### 5. 查询DO 和DI ####################
        # 查询端口1_0 的DO 值
        do = robot.getDO(1, 0, ec)
        print(message(ec))
        print("DO1_0 当前的信号值为:", do)
        # 查询端口1_0 的DI 值
        di = robot.getDI(0, 0, ec)
        print("DI1_0 当前的信号值为:", di)
        # 将DO1_0 的值设为false
        robot.setDO(0, 0, False, ec)
        # 查询端口1_0 的DO 值
        do = robot.getDO(0, 0, ec)
        print("DO0_0 修改后信号值为:", do)
        robot.setDO(0, 0, True, ec)

        ################## 6. 断开连接再重连 ####################
        # 机器人断开连接
        robot.disconnectFromRobot(ec)
        time.sleep(2)
        # 机器人再次连接
        robot.connectToRobot(ec)

        ################ 7. 打开和关闭拖动 ######################
        ## 机器人下电，因机器人拖动模式自动上电
        # robot.setPowerState(False, ec)
        ## 将机器人操作模式设为手动
        # robot.setOperateMode(rokae.OperateMode.manual, ec)
        ## 开启拖动
        # robot.enableDrag(rokae.DragParameter.Space.cartesianSpace.value, rokae.DragParameter.
    Type.freely.value, ec)
```

```
125.        # print("机器人状态:", robot.operationState(ec))
126.        # time.sleep(2)
127.        # # 关闭拖动
128.        # robot.disableDrag(ec)
129.        # print("机器人状态:", robot.operationState(ec))
130.        # print("非 Drag 模式下的上下电模式为： ", robot.powerState(ec))
131.        # time.sleep(2)
132.
133.        ############ 8. 查询工件/工具信息 ###############
134.        # 查询所有工具的信息
135.        tool = robot.toolsInfo(ec)
136.        print(tool)
137.        for name in tool.keys():
138.            print(name, "质量:", tool[str(name)]["load"]["mass"])
139.        # 查询所有工件的信息
140.        wobj = robot.wobjsInfo(ec)
141.        print("查询工件名信息为:")
142.        for name in wobj.keys():
143.            print(name)
144.
145.        ################ 9. 运动指令 ###################################
146.        robot.setOperateMode(rokae.OperateMode.automatic, ec)
147.        robot.setPowerState(True, ec)
148.        robot.moveReset(ec)
149.        # robot.setDefaultZone(100, ec)
150.        # robot.setDefaultSpeed(1000, ec)
151.        # p0 = robot.flangePos(ec)
152.        # print(p0)
153.
154.        ######## 10. Move L 点位测试/ NB4 运动指令 ######################
155.        p1 = MoveLCommand(
156.        [0.35650000000000004, 0.0, 0.578780145523736, 3.141592653589793, 0.523598775598299,
        3.141592653589793], 500, 50)
157.        p2 = MoveLCommand(
158.        [0.35650000000000004 + 0.1, 0.0, 0.578780145523736, 3.141592653589793, 0.523598775598
        299, 3.141592653589793],
159.        100, 50)
160.        p3 = MoveLCommand([0.35650000000000004 + 0.1, 0.0 + 0.1, 0.578780145523736, 3.141592
        653589793, 0.523598775598299,
161.                           3.141592653589793], 100, 50)
162.        p4 = MoveLCommand([0.35650000000000004, 0.0 + 0.1, 0.578780145523736, 3.1415926535
        89793, 0.523598775598299,
163.                           3.141592653589793], 100, 50)
164.        # 单条 append 测试转弯区
```

```
165.         robot. executeCommand ([p1], ec)
166.         robot. executeCommand ([p2], ec)
167.         robot. executeCommand ([p3], ec)
168.         robot. executeCommand ([p4], ec)
169.         robot. executeCommand ([p1], ec)
170.         robot.moveStart(ec)
171.
172.         time.sleep(1)
173.         robot.setPowerState(False, ec)
174.         robot.disconnectFromRobot(ec)
175.
176. if __name__ == '__main__':
177.     main()
```

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
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 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

8.1.6 机器人拖动模式参数 DragParameter

Space.jointSpace = 0	轴空间拖动
Space.cartesianSpace = 1	笛卡尔空间拖动
Type.translationOnly = 0	仅平移

Type.rotationOnly = 1	仅旋转
Type.freely = 2	自由拖拽

8.2 数据结构

8.2.1 运动指令 MoveAbsJ MoveAbsJCommand

target	目标关节点位
speed	速率
zone	转弯区大小
offset_type	偏移类型
offset	沿坐标系偏移大小

8.2.2 运动指令 MoveJ MoveJCommand

target	目标笛卡尔点位
speed	速率
zone	转弯区大小
offset_type	偏移类型
offset	沿坐标系偏移大小

8.2.3 运动指令 MoveL MoveLCommand

target	目标笛卡尔点位
speed	速率
zone	转弯区大小
offset_type	偏移类型
offset	沿坐标系偏移大小

8.2.4 运动指令 MoveC MoveCCommand

target	目标笛卡尔点位
aux	辅助笛卡尔点位
speed	速率
zone	转弯区大小
offset_type	偏移类型
offset	沿坐标系偏移大小

8.3 方法

8.3.1 机器人基本操作及信息查询

connectToRobot()	
def connectToRobot(ec)	
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

disconnectFromRobot()
def disconnectFromRobot(ec)
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(ec)	
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(ec)	
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(on, ec)	
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(ec)	
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(mode, ec)	
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(ec)	
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

flangePos()	
def flangePos(ec)	
机器人法兰相对于基坐标系位姿	
参数	[out] ec： 错误码
返回	数组列表： [X, Y, Z, A, B, C]，其中平移量单位为米旋转量单位为弧度

tcpPos()
def tcpPos(ec)

机器人工具相对于基坐标系位姿	
参数	[out] ec： 错误码
返回	数组列表： [X, Y, Z, A, B, C]，其中平移量单位为米旋转量单位为弧度

jointPos()	
def jointPos(ec) 机器人当前轴角度，单位: 弧度	
参数	[out] ec：错误码
返回	轴角度值/None（存在错误）

jointVel()	
def jointVel(ec) 机器人当前关节速度，单位：弧度/秒	
参数	[out] ec： 错误码
返回	关节速度/None（存在错误）

jointTorque()	
def jointTorque(ec) 关节力传感器数值，单位: Nm	
参数	[out] ec： 错误码
返回	力矩值/None（存在错误）

baseFrame()	
def baseFrame(ec) 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/None（存在错误）

toolset()	
def toolset(ec) 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()	
def setToolset(toolset, ec) 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（例如：{'end': {'rot': [0.0, 0.0, 0.0], 'trans': [0.0, 0.0, 0.0]}, 'load': {'cog': [0.0, 0.0, 0.0], 'inertia': [0.0, 0.0, 0.0], 'mass': 0.0}, 'ref': {'rot': [0.0, 0.0, 0.0], 'trans': [0.0, 0.0, 0.0]}}） [out] ec： 错误码

setToolName()
def setToolName(toolName, wobjName, ec)

通过工程中的工具工件名称，来进行工具工件坐标系设置。	
注解	注意：如果工程未加载或工具/工件名称不存在，则会引发异常
参数	[in] toolName：工具名称 [in] wobjName: 工件名称 [out] ec： 错误码

calcIk()	
def calcIk (posture, ec) 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）/None（计算有误）

calcFk()	
def calcFk(joints, ec) 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	法兰末端位姿（相对于基坐标系）/None（计算有误）

clearServoAlarm()	
def clearServoAlarm(ec) 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

sdkVersion()	
def sdkVersion(ec) 查询 RokaeSDK 版本	
参数	[out] ec：错误码
返回	版本号

getPointPos()	
def getPointPos(ec) 查询目前的点位序号（仅在一次下发多个点位时使用）	
参数	[out] ec：错误码
返回	当前的运动点位序号。

calibForceSensor()	
def calibForceSensor(ec, index=None) 该指令用于力传感器一键标定/单轴标定	
参数	[in] index：轴数索引(轴数索引号从 0 计数)。默认为 None,为力传感器一键标定 [out] ec：错误码

pathCheck()	
def pathCheck(start_point, start_point_joint, end_point, ec) 该指令用于路径点的可达性/奇异性/相邻点校验	
参数	[in] start_point：校验初始点（MoveLCommand 类型）

	[in] start_point_joint：校验初始点关节角列表 [in] end_point：校验结束点(MoveLCommand/MoveCCommand 类型) [out] ec：错误码
返回	点位可达，返回终点的校验轴角列表；点位不可达，返回 None

8.3.2 运动控制（非实时模式）

moveReset()	
def moveReset(ec) 重置运动缓存	
注解	清空已发送的运动指令，默认速度重置为 100(v100)，默认转弯区重置为 0(fine)．每次程序开始运行并第一次执行运动指令之前，必须调用该函数来重置运动缓存，否则控制器可能会报错
参数	[out] ec：错误码

pause()	
def pasue(ec) 停止机器人运动	
注解	机器人暂停，继续运动需再次调用 moveStart()接口。
参数	[out] ec： 错误码

stop()	
def stop(ec) 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

executeCommand()	
def executeCommand(values, ec) 添加单条或多条运动指令，调用后将运动指令添加至缓存队列。	
注解	须为同类型的指令；运动指令异步执行，如发生执行错误，需通过 lastErrorCode()获取错误码。
模板参数	Command 运动指令类: MoveJCommand | MoveAbsJCommand | MoveLCommand | MoveCCommand
参数	[in] values 指令列表，允许的个数为 1-1000 [out] ec： 错误码

executeDiffCommand()	
def executeDiffCommand(values, ec) 添加单条或多条运动指令，调用后将运动指令添加至缓存队列。	
注解	下发多种不同类型的指令；运动指令异步执行，如发生执行错误，需通过 lastErrorCode()获取错误码。
模板参数	Command 运动指令类: MoveJCommand | MoveAbsJCommand | MoveLCommand | MoveCCommand
参数	[in] values 指令列表，允许的个数为 1-1000 [out] ec： 错误码

moveStart()	
def moveStart (ec) 执行单条或多条运动指令，调用后机器人立刻开始运动	
注解	一般用于 executeCommand ()指令后，对添加的运动指令缓存进行执行。或执行完 pause()指令后，对机器人进行继续运动指令。
参数	[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(speed, ec) 设定默认运动速度	
注解	该数值表示末端最大线速度(单位 mm/s)，自动计算对应末端旋转速度及轴速度. 若不设置，则为 v100。
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(zone，ec) 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

lastErrorCode()	
def lastErrorCode(ec) 获取最新的错误码，目前为运动指令的执行结果	
返回	[out] ec：错误码，可调用 message(ec)获取详细信息

adjustSpeedOnline()	
def lastErrorCode(scale, ec) 在线调整机器人的运行速度	
返回	[in] scale: 速度调整百分比，调节范围 0.01-1。 [out] ec：错误码，可调用 message(ec)获取详细信息

8.3.3 通信相关

getDI()	
def getDI(board, channel, ec) 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(board, channel, ec) 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(board, channel, state, ec) 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

8.3.4 RL 工程

toolsInfo()
def toolsInfo(ec) 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(ec) 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(space, type, ec) 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(ec) 关闭拖动
参数 [out] ec：错误码

startRecordPath()
def startRecordPath (duration, ec) 开始拖动路径记录
参数 [in] duration：记录时长（秒） [out] ec：错误码

stopRecordPath()
def stopRecordPath(ec) 用于停止路径录制。（如需停止需多开线程调用，开始录制接口会在当前进程中阻塞直至指定录制时间结束）
参数 [out] ec：错误码

cancelRecordPath()
def cancelRecordPath(ec) 用于取消记录的回放路径
参数 [out] ec：错误码

saveRecordPath()
def saveRecordPath(name, ec, saveAs="") 用于保存拖动路径
参数 [in] name：保存路径名称 [out] ec：错误码 [in] saveAs：如保存路径名称存在，则路径名称命名

replayPath()
def replayPath(name, rate, ec) 用于回放路径
参数 [in] name：路径名称 [in] rate：回放速率 [out] ec：错误码

removePath()
def removePath(name, ec, removeAll=False) 用于删除保存的路径
参数 [in] name：希望删除的路径名称 [out] ec：错误码 [in] ec：是否删除所有已保存路径

queryPathLists()
def queryPathLists(ec) 用于查询已保存的路径列表
参数 [out] ec：错误码 **返回** 已存在的路径名称列表

8.3.6 焊接相关

weaveOn()

def weaveOn(ec) 焊接专用接口。开始焊接摆动接口	
参数	[out] ec：错误码

weaveOff()	
def weaveOff(ec) 焊接专用接口。结束焊接摆动接口	
参数	[in] name：希望删除的路径名称

setWeaveParameters()	
def setWeaveParameters (amplitude, frequency, left_dwell_time, right_dwell_time, mid_dwell_time, ec) 焊接专用接口。设置 Z 型摆动所需要的参数。目前仅支持 Z 型摆动且机器人停留的摆动方式。停留时间设置仅支持两侧停留。取值范围如下： 幅值：0.0001~0.05m 频率：0.5~2hz 停留时间：0~2s 使用该接口创建的点位速度不得超过 1000mm/s	
参数	[in] amplitude：单侧幅值（单位：m） [in] frequency：频率（hz） [in] amplitude：左侧停留时间（s） [in] amplitude：右侧停留时间（s） [in] amplitude：中部停留时间（s） [out] ec：错误码

8.3.7 其他

degree2rad()	
def degree2rad(degree) 数组度转弧度	
参数	[in] degree：度
返回	弧度

rad2degree()	
def rad2degree(rad) 数组弧度转度	
参数	[in] rad：弧度
返回	度

zeroToolset()	
def zeroToolset(): 初始化工具工件组	
返回	初始化的工具工件组

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