作者： XXX

日期：

版本：

Website: http://www.frtech.fr

文档编号：FR_SZRD_SOFT_2023_0001
法奥意威（苏州）机器人系统有限公司
江苏省苏州市高新区竹园路 209 号

PythonSDK 用户手册

1 概 述

Python 二次开发接口文档

2 免责声明

我们已对本文档描述的内容做测试。但是差错在所难免，无法保证绝对正确并完全满足您的使用需求。本文档的内容可能随时更新，也欢迎您提出改进建议。*文档内容可能随时更新*
如有改动，恕不事先通知

版本记录

版本编号	版本日期	机器人软件版本	说明
V1.0.0	2023.01.19	V3.5.6 及以上	1.创建文档

目录

1 概 述 1
2 免责声明 1
1 简介 7
2 文档须知 7
2.1 单位说明 7
2.2 其他说明 7
3 接口 7
3.1 机器人基础 7
3.1.1 实例化机器人 7
3.1.2 查询 SDK 版本号 7
3.1.3 获取控制器 IP 8
3.1.4 控制机器人进入或退出拖动示教模式 8
3.1.5 查询机器人是否处于拖动模式 9
3.1.6 控制机器人上使能或下使能 9
3.1.7 控制机器人手自动模式切换 9
3.2 机器人运动 10
3.2.1 jog 点动 10
3.2.2 jog 点动减速停止 12
3.2.3 jog 点动立即停止 13
3.2.4 关节空间运动 13
3.2.5 笛卡尔空间直线运动 14
3.2.6 笛卡尔空间圆弧运动 15
3.2.7 笛卡尔空间整圆运动 16
3.2.8 笛卡尔空间螺旋线运动 17
3.2.9 关节空间伺服模式运动 18
3.2.10 笛卡尔空间伺服模式运动 19
3.2.11 笛卡尔空间点到点运动 20
3.2.12 样条运动开始 20
3.2.13 样条运动 PTP 20
3.2.14 样条运动结束 21
3.2.15 新样条运动开始 21
3.2.16 新样条指令点 22
3.2.17 新样条运动结束 22
3.2.18 终止运动 23
3.2.19 点位整体偏移开始 23
3.2.20 点位整体偏移结束 23
3.3 机器人 IO 24
3.3.1 设置控制箱数字量输出 24
3.3.2 设置工具数字量输出 24
3.3.3 设置控制箱模拟量输出 25
3.3.4 设置工具模拟量输出 25

3.3.5 获取控制箱数字量输入 26
3.3.6 获取工具数字量输入 26
3.3.7 等待控制箱数字量输入 26
3.3.8 等待控制箱多路数字量输入 27
3.3.9 等待工具数字量输入 27
3.3.10 获取控制箱模拟量输入 27
3.3.11 获取工具模拟量输入 28
3.3.12 等待控制箱模拟量输入 28
3.3.13 等待工具模拟量输入 28
3.4 机器人常用设置 29
3.4.1 设置全局速度 29
3.4.2 设置系统变量值 29
3.4.3 设置工具坐标系 30
3.4.4 设置工具坐标系列表 30
3.4.5 设置外部工具坐标系 30
3.4.6 设置外部工具坐标系列表 31
3.4.7 设置工件坐标系 31
3.4.8 设置工件坐标系列表 31
3.4.9 设置末端负载重量 32
3.4.10 设置机器人安装方式 32
3.4.11 设置机器人安装角度-自由安装 32
3.4.12 设置末端负载质心坐标 32
3.4.13 等待指定时间 33
3.5 机器人安全设置 33
3.5.1 设置碰撞等级 33
3.5.2 设置碰撞后策略 33
3.5.3 设置正限位 34
3.5.4 设置负限位 34
3.5.5 错误状态清除 34
3.5.6 关节摩擦力补偿开关 35
3.5.7 设置关节摩擦力补偿系数-正装 35
3.5.8 设置关节摩擦力补偿系数-侧装 35
3.5.9 设置关节摩擦力补偿系数-倒装 35
3.5.10 设置关节摩擦力补偿系数-自由安装 36
3.6 机器人状态查询 36
3.6.1 获取机器人安装角度 36
3.6.2 获取系统变量值 36
3.6.3 获取当前关节位置(角度) 37
3.6.4 获取当前关节位置(弧度) 37
3.6.5 获取当前工具位姿 37
3.6.6 获取当前工具坐标系编号 38
3.6.7 获取当前工件坐标系编号 38
3.6.8 获取当前末端法兰位姿 38
3.6.9 逆运动学求解 39
3.6.10 逆运动学求解 39

3.6.11 逆运动学求解 39
3.6.12 正运动学求解 40
3.6.13 获取当前关节转矩 40
3.6.14 获取当前负载的重量 40
3.6.15 获取当前负载的质心 41
3.6.16 获取当前工具坐标系 41
3.6.17 获取当前工件坐标系 41
3.6.18 获取关节软限位角度 42
3.6.19 获取系统时间 42
3.6.20 获取机器人当前关节配置 42
3.6.21 获取默认速度 42
3.6.22 查询机器人运动是否完成 43
3.7 机器人轨迹复现 43
3.7.1 设置轨迹记录参数 43
3.7.2 开始轨迹记录 44
3.7.3 停止轨迹记录 44
3.7.4 删除轨迹记录 45
3.7.5 轨迹预加载 45
3.7.6 轨迹复现 45
3.8 机器人 WebApp 程序使用 46
3.8.1 设置开机自动加载默认的作业程序 46
3.8.2 加载指定的作业程序 46
3.8.3 获取已加载的作业程序名 46
3.8.4 获取当前机器人作业程序的执行行号 47
3.8.5 运行当前加载的作业程序 47
3.8.6 暂停当前运行的作业程序 48
3.8.7 恢复当前暂停的作业程序 48
3.8.8 终止当前运行的作业程序 48
3.8.9 获取机器人作业程序执行状态 48
3.9 机器人外设 48
3.9.1 配置夹爪 48
3.9.2 获取夹爪配置 49
3.9.3 激活夹爪 49
3.9.4 控制夹爪 50
3.9.5 获取夹爪运动状态 50
3.10 机器人力控 50
3.10.1 力传感器配置 50
3.10.2 获取力传感器配置 51
3.10.3 力传感器激活 51
3.10.4 力传感器校零 51
3.10.5 设置力传感器参考坐标系 52
3.10.6 负载重量辨识记录 52
3.10.7 负载重量辨识计算 52
3.10.8 负载质心辨识记录 53
3.10.9 负载质心辨识计算 53

3.10.10 获取参考坐标系下力/扭矩数据 53
3.10.11 获取力传感器原始力/扭矩数据 54
3.10.12 碰撞守护 54
3.10.13 恒力控制 55
3.10.14 螺旋线探索 56
3.10.15 旋转插入 57
3.10.16 直线插入 58
3.10.17 表面定位 59
3.10.18 计算中间平面位置开始 61
3.10.19 计算中间平面位置结束 61
3.10.20 柔顺控制开启 61
3.10.21 柔顺控制关闭 62
4 附录 63

1 简介

本文档为 python 版本的二次开发接口文档。

2 文档须知

2.1 单位说明

机器人位置单位为毫米(mm)，姿态单位为度(°)。

2.2 其他说明

1) 非特别说明的代码示例中都默认机器人已经正常开机使能；
2) 文档中的所有代码示例都默认在机器人的工作空间内没有任何干涉；
3) 实际使用测试时请采用现场机器人的数据使用；

3 接口

3.1 机器人基础

3.1.1 实例化机器人

原型	RPC(ip)
描述	实例化一个机器人对象
参数	ip：机器人的 IP 地址，默认出厂 IP 为“192.168.58.2”
返回值	成功：返回一个机器人对象 失败：创建的对象会被销毁

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
```

3.1.2 查询 SDK 版本号

原型	GetSDKVersion()
描述	查询 SDK 版本号
参数	无
返回值	成功：[0,version] 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
ret = robot.GetSDKVersion()     #查询 SDK 版本号
```

```
if ret[0] == 0:  #0-无故障，返回格式：[errcode,data],errcode-故障码，data-数据
    print("SDK version is:",ret[1])
else:
    print("the errcode is: ", ret[0])
```

3.1.3 获取控制器 IP

原型	GetControllerIP()
描述	查询控制器 IP
参数	无
返回值	成功：[0,IP] 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
ret = robot.GetControllerIP()    #查询控制器 IP
if ret[0] == 0:
    print("controller ip is:",ret[1])
else:
    print("the errcode is: ", ret[0])
```

3.1.4 控制机器人进入或退出拖动示教模式

原型	DragTeachSwitch(state)
描述	控制机器人进入或退出拖动示教模式
参数	state：1-进入拖动示教模式，0-退出拖动示教模式
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
import time
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.Mode(1) #机器人切入手动模式
time.sleep(1)
robot.DragTeachSwitch(1)  #机器人切入拖动示教模式，必须在手动模式下才能切入拖动示教模式
time.sleep(1)
ret = robot.IsInDragTeach()    #查询是否处于拖动示教模式，1-拖动示教模式，0-非拖动示教模式
if ret[0] == 0:
    print("drag state is:",ret[1])
else:
    print("the errcode is: ", ret[0])
time.sleep(3)
```

```
robot.DragTeachSwitch(0)  #机器人切入非拖动示教模式，必须在手动模式下才能切入非拖动示教模式
time.sleep(1)
ret = robot.IsInDragTeach()    #查询是否处于拖动示教模式，1-拖动示教模式，0-非拖动示教模式
if ret[0] == 0:
    print("drag state is:",ret[1])
else:
    print("the errcode is: ", ret[0])
```

3.1.5 查询机器人是否处于拖动模式

原型	IsInDragTeach ()
描述	查询机器人是否处于拖动示教模式
参数	无
返回值	成功：[0,state]，state:0-非拖动示教模式，1-拖动示教模式 失败：[errcode]

代码示例：

见 3.1.4

3.1.6 控制机器人上使能或下使能

原型	RobotEnable(state)
描述	控制机器人上使能或下使能
参数	state：1-上使能，0-下使能
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
import time
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.RobotEnable(0)   #机器人下使能
time.sleep(3)
robot.RobotEnable(1)   #机器人上使能，机器人上电后默认自动上使能
```

3.1.7 控制机器人手自动模式切换

原型	Mode(state)
描述	控制机器人手自动模式切换
参数	state：1-手动模式，0-自动模式
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
import time
# 与机器人控制器建立连接，连接成功返回一个机器人对象
```

```
robot = frrpc.RPC('192.168.58.2')
robot.Mode(0)    #机器人切入自动运行模式
time.sleep(1)
robot.Mode(1)    #机器人切入手动模式
```

3.2 机器人运动

3.2.1 jog 点动

原型	StartJOG(ref,nb,dir,vel,acc,max_dis)
描述	jog 点动
参数	ref：0-关节点动,2-基坐标系点动,4-工具坐标系点动,8-工件坐标系点动； nb：1-1 关节(x 轴),2-2 关节(y 轴),3-3 关节(z 轴),4-4 关节(rx),5-5 关节(ry),6-6 关节(rz); dir：0-负方向，1-正方向; vel：速度百分比，[0~100]; acc：加速度百分比，[0~100]; max_dis：单次点动最大角度/距离，单位° 或 mm
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
import time
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
# 机器人单轴点动
robot.StartJOG(0,1,0,20.0,20.0,30.0)     # 单关节运动,StartJOG 为非阻塞指令，运动状态下接收其他运动指令（包含 StartJOG）会被丢弃
time.sleep(1)
#机器人单轴点动减速停止
# robot.StopJOG(1)
#机器人单轴点动立即停止
robot.ImmStopJOG()
robot.StartJOG(0,2,1,20.0,20.0,30.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(0,3,1,20.0,20.0,30.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(0,4,1,20.0,20.0,30.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(0,5,1,20.0,20.0,30.0)
time.sleep(1)
robot.ImmStopJOG()
```

```
robot.StartJOG(0,6,1,20.0,20.0,30.0)
time.sleep(1)
robot.ImmStopJOG()
# 基坐标
robot.StartJOG(2,1,0,20.0,20.0,100.0)  #基坐标系下点动
time.sleep(1)
#机器人单轴点动减速停止
# robot.StopJOG(2)
# #机器人单轴点动立即停止
robot.ImmStopJOG()
robot.StartJOG(2,1,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(2,2,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(2,3,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(2,4,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(2,5,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(2,6,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
# 工具坐标
robot.StartJOG(4,1,0,20.0,20.0,100.0)  #工具坐标系下点动
time.sleep(1)
#机器人单轴点动减速停止
# robot.StopJOG(5)
# #机器人单轴点动立即停止
robot.ImmStopJOG()
robot.StartJOG(4,1,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(4,2,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(4,3,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(4,4,1,20.0,20.0,100.0)
```

```
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(4,5,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(4,6,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
# 工件坐标
robot.StartJOG(8,1,0,20.0,20.0,100.0)  #工件坐标系下点动
time.sleep(1)
#机器人单轴点动减速停止
# robot.StopJOG(9)
# #机器人单轴点动立即停止
robot.ImmStopJOG()
robot.StartJOG(8,1,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(8,2,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(8,3,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(8,4,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(8,5,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(8,6,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
```

3.2.2 jog 点动减速停止

原型	StopJOG (ref)
描述	jog 点动减速停止
参数	ref：1-关节点动停止,3-基坐标系点动停止,5-工具坐标系点动停止,9-工件坐标系点动停止；
返回值	成功：[0] 失败：[errcode]

代码示例：

见 3.2.1

3.2.3 jog 点动立即停止

原型	ImmStopJOG ()
描述	jog 点动立即停止
参数	无
返回值	成功：[0] 失败：[errcode]

代码示例：

见 3.2.1

3.2.4 关节空间运动

原型	MoveJ(joint_pos,desc_pos,tool,user,vel,acc,ovl,exaxis_pos,blendT,offset_flag,offset_pos)
描述	关节空间运动
参数	joint_pos:目标关节位置，单位[°]； desc_pos:目标笛卡尔位姿，单位[mm][°]； tool:工具号，[0~14]； user:工件号，[0~14]； vel:速度百分比，[0~100]； acc:加速度百分比，[0~100]，暂不开放； ovl:速度缩放因子，[0~100]； exaxis_pos:外部轴 1 位置~外部轴 4 位置； blendT:[-1.0]-运动到位(阻塞)，[0~500]-平滑时间(非阻塞)，单位[ms]； offset_flag:[0]-不偏移，[1]-工件/基坐标系下偏移，[2]-工具坐标系下偏移； offset_pos:位姿偏移量，单位[mm][°]。
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
import time
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
J1=[-168.847,-93.977,-93.118,-80.262,88.985,11.831]
P1=[-558.082,27.343,208.135,-177.205,-0.450,89.288]
eP1=[0.000,0.000,0.000,0.000]
dP1=[1.000,1.000,1.000,1.000,1.000,1.000]
J2=[168.968,-93.977,-93.118,-80.262,88.986,11.831]
P2=[-506.436,236.053,208.133,-177.206,-0.450,67.102]
eP2=[0.000,0.000,0.000,0.000]
dP2=[1.000,1.000,1.000,1.000,1.000,1.000]
robot.MoveJ(J1,P1,1,0,100.0,180.0,100.0,eP1,-1.0,0,dP1)      #关节空间运动 PTP,工具号 1，实际测试根据现场数据及工具号使用
robot.MoveJ(J2,P2,1,0,100.0,180.0,100.0,eP2,-1.0,0,dP2)
time.sleep(2)
```

```
j1 = robot.GetInverseKin(0,P1,-1)          #只有笛卡尔空间坐标的情况下，可用逆运动学接口求解关节位置
print(j1)
j1 = [j1[1],j1[2],j1[3],j1[4],j1[5],j1[6]]
robot.MoveJ(j1,P1,1,0,100.0,180.0,100.0,eP1,-1.0,0,dP1)
j2 = robot.GetInverseKin(0,P2,-1)
print(j2)
j2 = [j2[1],j2[2],j2[3],j2[4],j2[5],j2[6]]
robot.MoveJ(j2,P2,1,0,100.0,180.0,100.0,eP2,-1.0,0,dP2)
time.sleep(2)
p1 = robot.GetForwardKin(J1)          #只有关节位置的情况下，可用正运动学接口求解笛卡尔空间坐标
print(p1)
p1 = [p1[1],p1[2],p1[3],p1[4],p1[5],p1[6]]
robot.MoveJ(J1,p1,1,0,100.0,180.0,100.0,eP1,-1.0,0,dP1)
p2 = robot.GetForwardKin(J2)
print(p2)
p2 = [p2[1],p2[2],p2[3],p2[4],p2[5],p2[6]]
robot.MoveJ(J2,p2,1,0,100.0,180.0,100.0,eP2,-1.0,0,dP2)
```

3.2.5 笛卡尔空间直线运动

原型	MoveL(joint_pos,desc_pos ,tool,user,vel,acc,ovl,exaxis_pos ,blendR,search, offset_flag,offset_pos)
描述	笛卡尔空间直线运动
参数	joint_pos:目标关节位置，单位[°]； desc_pos:目标笛卡尔位姿，单位[mm][°]； tool:工具号，[0~14]； user:工件号，[0~14]； vel:速度百分比，[0~100]； acc:加速度百分比，[0~100]，暂不开放； ovl:速度缩放因子，[0~100]； exaxis_pos:外部轴 1 位置~外部轴 4 位置； blendR:[-1.0]-运动到位(阻塞)，[0~1000]-平滑半径(非阻塞)，单位[mm]； search:[0]-不焊丝寻位，[1]-焊丝寻位； offset_flag:[0]-不偏移，[1]-工件/基坐标系下偏移，[2]-工具坐标系下偏移； offset_pos:偏移量，单位[mm][°]。
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
J1=[95.442,-101.149,-98.699,-68.347,90.580,-47.174]
P1=[75.414,568.526,338.135,-178.348,-0.930,52.611]
```

```
eP1=[0.000,0.000,0.000,0.000]
dP1=[10.000,10.000,10.000,0.000,0.000,0.000]
J2=[123.709,-121.190,-82.838,-63.499,90.471,-47.174]
P2=[-273.856,643.260,259.235,-177.972,-1.494,80.866]
eP2=[0.000,0.000,0.000,0.000]
dP2=[0.000,0.000,0.000,0.000,0.000,0.000]
J3=[167.066,-95.700,-123.494,-42.493,90.466,-47.174]
P3=[-423.044,229.703,241.080,-173.990,-5.772,123.971]
eP3=[0.000,0.000,0.000,0.000]
dP3=[0.000,0.000,0.000,0.000,0.000,0.000]
robot.MoveL(J1,P1,0,0,100.0,180.0,100.0,-1.0,eP1,0,1 ,dP1)    #笛卡尔空间直线运动
robot.MoveL(J2,P2,0,0,100.0,180.0,100.0,-1.0,eP2,0,0,dP2)
robot.MoveL(J3,P3,0,0,100.0,180.0,100.0,-1.0,eP3,0,0,dP3)
```

3.2.6 笛卡尔空间圆弧运动

原型	MoveC(joint_pos_p,desc_pos_p,ptool,puser,pvel,pacc,exaxis_pos_p,poffset_flag,offset_pos_p,joint_pos_t ,desc_pos_t ,ttool,tuser,tvel,tacc,exaxis_pos_t ,toffset_flag,offset_pos_t ,ovl,blendR)
描述	笛卡尔空间圆弧运动
参数	joint_pos_p:路径点关节位置，单位[°]； desc_pos_p:路径点笛卡尔位姿，单位[mm][°]； ptool:工具号，[0~14]； puser:工件号，[0~14]； pvel:速度百分比，[0~100]； pacc:加速度百分比，[0~100]，暂不开放； exaxis_pos_p:外部轴 1 位置~外部轴 4 位置； poffset_flag:[0]-不偏移，[1]-工件/基坐标系下偏移，[2]-工具坐标系下偏移； offset_pos_p:偏移量，单位[mm][°]； joint_pos_t:目标点关节位置，单位[°]； desc_pos_t:目标点笛卡尔位姿，单位[mm][°]； ttool:工具号，[0~14]； tuser:工件号，[0~14]； tvel:速度百分比，[0~100]； tacc:加速度百分比，[0~100]，暂不开放； exaxis_pos_t:外部轴 1 位置~外部轴 4 位置； toffset_flag:[0]-不偏移，[1]-工件/基坐标系下偏移，[2]-工具坐标系下偏移； offset_pos_t:偏移量，单位[mm][°]。 ovl:速度缩放因子，[0~100]； blendR:[-1.0]-运动到位(阻塞)，[0~1000]-平滑半径(非阻塞)，单位[mm]。
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
```

```
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
J1=[121.381,-97.108,-123.768,-45.824,89.877,-47.296]
P1=[-127.772,459.534,221.274,-177.850,-2.507,78.627]
eP1=[0.000,0.000,0.000,0.000]
dP1=[10.000,10.000,10.000,10.000,10.000,10.000]
J2=[138.884,-114.522,-103.933,-49.694,90.688,-47.291]
P2=[-360.468,485.600,196.363,-178.239,-0.893,96.172]
eP2=[0.000,0.000,0.000,0.000]
dP2=[10.000,10.000,10.000,10.000,10.000,10.000]
pa2=[0.0,0.0,100.0,180.0]
J3=[159.164,-96.105,-128.653,-41.170,90.704,-47.290]
P3=[-360.303,274.911,203.968,-176.720,-2.514,116.407]
eP3=[0.000,0.000,0.000,0.000]
dP3=[10.000,10.000,10.000,10.000,10.000,10.000]
pa3=[0.0,0.0,100.0,180.0]
dP=[10.000,10.000,10.000,10.000,10.000,10.000]
robot.MoveJ(J1,P1,0,0,100.0,180.0,100.0,eP1,-1.0,0,dP1)          #关节空间运动 PTP
robot.MoveC(J2,P2,pa2,eP2,0,dP2,J3,P3,pa3,eP3,0,dP3,100.0,-1.0)      #笛卡尔空间圆弧运动
```

3.2.7 笛卡尔空间整圆运动

原型	Circle(joint_pos_p,desc_pos_p ,ptool,puser,pvel,pacc, exaxis_pos_p, joint_pos_t ,desc_p_t, ttool,tuser,tsvel,tacc,exaxis_pos_t ,ovl,offset_flag, offset_pos)
描述	笛卡尔空间整圆运动
参数	joint_pos_p:路径点关节位置，单位[°]； desc_pos_p:路径点笛卡尔位姿，单位[mm][°]； ptool:工具号，[0~14]； puser:工件号，[0~14]； pvel:速度百分比，[0~100]； pacc:加速度百分比，[0~100]，暂不开放； exaxis_pos_p:外部轴 1 位置~外部轴 4 位置； joint_pos_t:目标点关节位置，单位[°]； desc_pos_t:目标点笛卡尔位姿，单位[mm][°]； ttool:工具号，[0~14]； tuser:工件号，[0~14]； tvel:速度百分比，[0~100]； tacc:加速度百分比，[0~100]，暂不开放； exaxis_pos_t:外部轴 1 位置~外部轴 4 位置； ovl:速度缩放因子，范围[0~100%]； offset_flag:[0]-不偏移，[1]-工件/基坐标系下偏移，[2]-工具坐标系下偏移； offset_pos:偏移量，单位[mm][°]。
返回值	成功：[0]

	失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
J1=[121.381,-97.108,-123.768,-45.824,89.877,-47.296]
P1=[-127.772,459.534,221.274,-177.850,-2.507,78.627]
eP1=[0.000,0.000,0.000,0.000]
dP1=[10.000,10.000,10.000,10.000,10.000,10.000]
J2=[138.884,-114.522,-103.933,-49.694,90.688,-47.291]
P2=[-360.468,485.600,196.363,-178.239,-0.893,96.172]
eP2=[0.000,0.000,0.000,0.000]
dP2=[10.000,10.000,10.000,10.000,10.000,10.000]
pa2=[0.0,0.0,100.0,180.0]
J3=[159.164,-96.105,-128.653,-41.170,90.704,-47.290]
P3=[-360.303,274.911,203.968,-176.720,-2.514,116.407]
eP3=[0.000,0.000,0.000,0.000]
dP3=[10.000,10.000,10.000,10.000,10.000,10.000]
pa3=[0.0,0.0,100.0,180.0]
dP=[10.000,10.000,10.000,10.000,10.000,10.000]
robot.MoveJ(J1,P1,0,0,100.0,180.0,100.0,eP1,-1.0,0,dP1)      #关节空间运动 PTP
robot.Circle(J2,P2,pa2,eP2,J3,P3,pa3,eP3,100.0,0,dP)      #笛卡尔空间整圆运动
```

3.2.8 笛卡尔空间螺旋线运动

原型	NewSpiral(joint_pos,desc_pos,tool,user,vel,acc,exaxis_pos,ovl,offset_flag,offset_pos,param)
描述	笛卡尔空间螺旋线运动
参数	joint_pos:目标关节位置，单位[°]； desc_pos:目标笛卡尔位姿，单位[mm][°]； tool:工具号，[0~14]； user:工件号，[0~14]； vel:速度百分比，[0~100]； acc:加速度百分比，[0~100]，暂不开放； exaxis_pos:外部轴 1 位置~外部轴 4 位置； ovl:速度缩放因子，[0~100]； offset_flag:[0]-不偏移，[1]-工件/基坐标系下偏移，[2]-工具坐标系下偏移； offset_pos:位姿偏移量，单位[mm][°]。 param:[circle_num,circle_angle,rad_init,rad_add,rotaxis_add,rot_direction] circle_num:螺旋圈数，circle_angle:螺旋倾角，rad_init:螺旋初始半径 rad_add:半径增量，rotaxis_add:转轴方向增量，rot_direction:旋转方向，0-顺时针，1-逆时针；
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc

# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')


J1=[127.888,-101.535,-94.860,17.836,96.931,-61.325]
eP1=[0.000,0.000,0.000,0.000]
dP1=[50.0,0.0,0.0,-30.0,0.0,0.0]
J2=[127.888,-101.535,-94.860,17.836,96.931,-61.325]
eP2=[0.000,0.000,0.000,0.000]
dP2=[50.0,0.0,0.0,-5.0,0.0,0.0]
Pa = [5.0,5.0,50.0,10.0,10.0,0.0]
P1 = robot.GetForwardKin(J1)        #只有关节位置的情况下，可用正运动学接口求解笛卡尔空间坐标
print(P1)
P1 = [P1[1],P1[2],P1[3],P1[4],P1[5],P1[6]]
robot.MoveJ(J1,P1,0,0,100.0,180.0,100.0,eP1,0.0,2,dP1)
P2 = robot.GetForwardKin(J2)        #只有关节位置的情况下，可用正运动学接口求解笛卡尔空间坐标
print(P2)
P2 = [P2[1],P2[2],P2[3],P2[4],P2[5],P2[6]]
robot.NewSpiral(J2,P2,0,0,100.0,180.0,eP2,100.0,2,dP2,Pa)    #螺旋线运动
```

3.2.9 关节空间伺服模式运动

原型	ServoJ(joint_pos,acc,vel,cmdT,filterT,gain)
描述	关节空间伺服模式运动
参数	joint_pos:目标关节位置，单位[°]； acc:加速度，范围[0~100]，暂不开放，默认为 0； vel:速度，范围[0~100]，暂不开放，默认为 0； cmdT:指令周期，单位[s]，[0.001~0.016]； filterT:滤波时间，单位[s]，暂不开放； gain:目标位置的比例放大器，暂不开放。
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
import time
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
joint_pos = robot.GetActualJointPosDegree(0)
print(joint_pos)
joint_pos =
[joint_pos[1],joint_pos[2],joint_pos[3],joint_pos[4],joint_pos[5],joint_pos[6
]]
```

```
acc = 0.0
vel = 0.0
t = 0.008
lookahead_time = 0.0
P = 0.0
count = 100
while(count):
    robot.ServoJ(joint_pos, acc, vel, t, lookahead_time, P)
    joint_pos[0] = joint_pos[0] + 0.1
    count = count - 1
    time.sleep(0.008)
```

3.2.10 笛卡尔空间伺服模式运动

原型	ServoCart(mode,desc_pos,pos_gain,acc,vel,cmdT,filterT,gain)
描述	笛卡尔空间伺服模式运动
参数	mode:[0]-绝对运动(基坐标系)，[1]-增量运动（基坐标系），[2]-增量运动（工具坐标系）； desc_pos:目标笛卡尔位置/目标笛卡尔位置增量； pos_gain:位姿增量比例系数，仅在增量运动下生效，范围[0~1]； acc:加速度百分比，[0~100]，暂不开放，默认为 0； vel:速度百分比，[0~100]，暂不开放，默认为 0； cmdT:指令周期，单位[s]，建议范围[0.001~0.016]； filterT:滤波时间，单位[s]，暂不开放； gain:目标位置的比例放大器，暂不开放。
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
import time
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
mode = 2  #工具坐标系增量运动
n_pos = [0.0,0.0,0.5,0.0,0.0,0.0]    #笛卡尔空间位姿增量
gain = [0.0,0.0,1.0,0.0,0.0,0.0]
acc = 0.0
vel = 0.0
t = 0.008
lookahead_time = 0.0
P = 0.0
count = 100
while(count):
    robot.ServoCart(mode, n_pos, gain, acc, vel, t, lookahead_time, P)
    count = count - 1
    time.sleep(0.008)
```

3.2.11 笛卡尔空间点到点运动

原型	MoveCart(desc_pos,tool,user,vel,acc,ovl,blendT,config)
描述	笛卡尔空间点到点运动
参数	desc_pos:目标笛卡尔位置； tool:工具号，[0~14]； user:工件号，[0~14]； vel:速度百分比，[0~100]； acc:加速度百分比，[0~100]，暂不开放； ovl:速度缩放因子，[0~100]； blendT:[-1.0]-运动到位(阻塞)，[0~500]-平滑时间(非阻塞)，单位[ms]； config:关节配置，[-1]-参考当前关节位置求解，[0~7]-依据关节配置求解
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
import time
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
P1=[75.414,568.526,338.135,-178.348,-0.930,52.611]
P2=[-273.856,643.260,259.235,-177.972,-1.494,80.866]
P3=[-423.044,229.703,241.080,-173.990,-5.772,123.971]
robot.MoveCart(P1,0,0,100.0,100.0,100.0,-1.0,-1)        #笛卡尔空间点到点运动
robot.MoveCart(P2,0,0,100.0,100.0,100.0,-1.0,-1)
robot.MoveCart(P3,0,0,100.0,100.0,100.0,0.0,-1)
time.sleep(1)
robot.StopMotion()     #停止运动
```

3.2.12 样条运动开始

原型	SplineStart()
描述	样条运动开始
参数	无
返回值	成功：[0] 失败：[errcode]

代码示例：

见 3.2.13

3.2.13 样条运动 PTP

原型	SplinePTP(joint_pos,desc_pos,tool,user,vel,acc,ovl)
描述	样条运动 PTP
参数	joint_pos:目标关节位置，单位[°]； desc_pos:目标笛卡尔位姿，单位[mm][°]； tool:工具号，[0~14]； user:工件号，[0~14]； vel:速度百分比，[0~100]；

	acc:加速度百分比，[0~100]，暂不开放； ovl:速度缩放因子，[0~100]；
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
J1 = [114.578,-117.798,-97.745,-54.436,90.053,-45.216]
P1 = [-140.418,619.351,198.369,-179.948,0.023,69.793]
eP1 = [0.000,0.000,0.000,0.000]
dP1 = [0.000,0.000,0.000,0.000,0.000,0.000]
J2 = [115.401,-105.206,-117.959,-49.727,90.054,-45.222]
P2 = [-95.586,504.143,186.880,178.001,2.091,70.585]
J3 = [135.609,-103.249,-120.211,-49.715,90.058,-45.219]
P3 = [-252.429,428.903,188.492,177.804,2.294,90.782]
J4 = [154.766,-87.036,-135.672,-49.045,90.739,-45.223]
P4 = [-277.255,272.958,205.452,179.289,1.765,109.966]
robot.MoveJ(J1,P1,0,0,100.0,180.0,100.0,eP1,-1.0,0,dP1)
robot.SplineStart()     #样条运动开始
robot.SplinePTP(J1,P1,0,0,100.0,180.0,100.0)     #样条 PTP 运动
robot.SplinePTP(J2,P2,0,0,100.0,180.0,100.0)
robot.SplinePTP(J3,P3,0,0,100.0,180.0,100.0)
robot.SplinePTP(J4,P4,0,0,100.0,180.0,100.0)
robot.SplineEnd()      #样条运动结束
```

3.2.14 样条运动结束

原型	SplineEnd()
描述	样条运动结束
参数	无
返回值	成功：[0] 失败：[errcode]

代码示例：

见 3.2.13

3.2.15 新样条运动开始

原型	NewSplineStart(type)
描述	新样条运动开始
参数	type:0-圆弧过渡，1-给定点位路径点
返回值	成功：[0] 失败：[errcode]

代码示例：

见 3.2.16

3.2.16 新样条指令点

原型	NewSplinePoint(joint_pos,desc_pos,tool,user,vel,acc,ovl,blendR,lastFlag)
描述	新样条指令点
参数	joint_pos:目标关节位置，单位[°]； desc_pos:目标笛卡尔位姿，单位[mm][°]； tool:工具号，[0~14]； user:工件号，[0~14]； vel:速度百分比，[0~100]； acc:加速度百分比，[0~100]，暂不开放； ovl:速度缩放因子，[0~100]； blendR: [0~1000]-平滑半径，单位[mm]； lastFlag:是否为最后一个点，0-否，1-是；
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
J1 = [114.578,-117.798,-97.745,-54.436,90.053,-45.216]
P1 = [-140.418,619.351,198.369,-179.948,0.023,69.793]
eP1 = [0.000,0.000,0.000,0.000]
dP1 = [0.000,0.000,0.000,0.000,0.000,0.000]
J2 = [115.401,-105.206,-117.959,-49.727,90.054,-45.222]
P2 = [-95.586,504.143,186.880,178.001,2.091,70.585]
J3 = [135.609,-103.249,-120.211,-49.715,90.058,-45.219]
P3 = [-252.429,428.903,188.492,177.804,2.294,90.782]
J4 = [154.766,-87.036,-135.672,-49.045,90.739,-45.223]
P4 = [-277.255,272.958,205.452,179.289,1.765,109.966]
robot.MoveJ(J1,P1,0,0,100.0,180.0,100.0,eP1,-1.0,0,dP1)
robot.NewSplineStart(1)     #样条运动开始
robot.NewSplinePoint(J1,P1,0,0,50.0,50.0,50.0,0.0,0)     #样条控制点
robot.NewSplinePoint(J2,P2,0,0,50.0,50.0,50.0,0.0,0)
robot.NewSplinePoint(J3,P3,0,0,50.0,50.0,50.0,0.0,0)
robot.NewSplinePoint(J4,P4,0,0,50.0,50.0,50.0,0.0,1)
robot.NewSplineEnd()
```

3.2.17 新样条运动结束

原型	NewSplineEnd()
描述	新样条运动结束
参数	无
返回值	成功：[0] 失败：[errcode]

代码示例：

见 3.2.16

3.2.18 终止运动

原型	StopMotion ()
描述	终止运动，使用终止运动需运动指令为非阻塞状态。
参数	无
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
import time
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
P1=[75.414,568.526,338.135,-178.348,-0.930,52.611]
P2=[-273.856,643.260,259.235,-177.972,-1.494,80.866]
P3=[-423.044,229.703,241.080,-173.990,-5.772,123.971]
robot.MoveCart(P1,0,0,100.0,100.0,100.0,-1.0,-1)          #关节空间点到点运动
robot.MoveCart(P2,0,0,100.0,100.0,100.0,-1.0,-1)
robot.MoveCart(P3,0,0,100.0,100.0,100.0,0.0,-1)    #此条运动指令为非阻塞状态
time.sleep(1)
robot.StopMotion()     #停止运动
```

3.2.19 点位整体偏移开始

原型	PointsOffsetEnable(flag,offset_pos)
描述	点位整体偏移开始
参数	flag:0-基坐标或工件坐标系下偏移， 2-工具坐标系下偏移； offset_pos:偏移量，单位[mm][°]。
返回值	成功：[0] 失败：[errcode]

3.2.20 点位整体偏移结束

原型	PointsOffsetDisable()
描述	点位整体偏移结束
参数	无
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
import time
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
#机器人点位整体偏移
J1=[-168.847,-93.977,-93.118,-80.262,88.985,11.831]
P1=[-558.082,27.343,208.135,-177.205,-0.450,89.288]
eP1=[0.000,0.000,0.000,0.000]
```

```
dP1=[10.000,10.000,10.000,0.000,0.000,0.000]
J2=[168.968,-93.977,-93.118,-80.262,88.986,11.831]
P2=[-506.436,236.053,208.133,-177.206,-0.450,67.102]
eP2=[0.000,0.000,0.000,0.000]
dP2=[0.000,0.000,0.000,0.000,0.000,0.000]
robot.MoveJ(J1,P1,1,0,100.0,180.0,100.0,eP1,-1.0,0,dP1)
robot.MoveJ(J2,P2,1,0,100.0,180.0,100.0,eP2,-1.0,0,dP2)
time.sleep(2)
flag = 0
offset = [100.0,5.0,6.0,0.0,0.0,0.0]    #位姿偏移量
robot.PointsOffsetEnable(flag, offset)    #整体偏移开始
robot.MoveJ(J1,P1,1,0,100.0,180.0,100.0,eP1,-1.0,0,dP1)
robot.MoveJ(J2,P2,1,0,100.0,180.0,100.0,eP2,-1.0,0,dP2)
robot.PointsOffsetDisable()  #整体偏移结束
```

3.3 机器人 IO

3.3.1 设置控制箱数字量输出

原型	SetDO(id,status,smooth,block)
描述	设置控制箱数字量输出
参数	id:io 编号，范围[0~15]； status:0-关，1-开； smooth:0-不平滑，1-平滑； block:0-阻塞，1-非阻塞。
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
for i in range(0,16):
    robot.SetDO(i,1,0,0)    #打开控制箱 DO
robot.WaitMs(1000)
for i in range(0,16):
    robot.SetDO(i,0,0,0)    #关闭控制箱 DO
robot.WaitMs(1000)
```

3.3.2 设置工具数字量输出

原型	SetToolDO(id,status,smooth,block)
描述	设置工具数字量输出
参数	id:io 编号，范围[0~1]； status:0-关，1-开； smooth:0-不平滑，1-平滑； block:0-阻塞，1-非阻塞。

返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
for i in range(0,2):
    robot.SetToolDO(i,1,0,0)     #打开工具 DO
robot.WaitMs(1000)
for i in range(0,2):
    robot.SetToolDO(i,0,0,0)     #关闭工具 DO
```

3.3.3 设置控制箱模拟量输出

原型	SetAO(id,value,block)
描述	设置控制箱模拟量输出
参数	id:io 编号，范围[0~1]； value:电流或电压值百分比，范围[0~100%]对应电流值[0~20mA]或电压[0~10V]； block:[0]-阻塞，[1]-非阻塞。
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.SetAO(0,0.0,0)     # 设置控制箱模拟量输出
robot.WaitMs(1000)
robot.SetAO(1,100.0,0)
```

3.3.4 设置工具模拟量输出

原型	SetToolAO(id,value,block)
描述	设置工具模拟量输出
参数	id:io 编号，范围[0]； value:电流或电压值百分比，范围[0~100%]对应电流值[0~20mA]或电压[0~10V]； block:[0]-阻塞，[1]-非阻塞。
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.SetToolAO(0,100.0,0)    # 设置工具模拟量输出
robot.WaitMs(1000)
```

```
robot.SetToolAO(0,0.0,0)
```

3.3.5 获取控制箱数字量输入

原型	GetDI(id,block)
描述	获取控制箱数字量输入
参数	id:io 编号，范围[0~15]； block:0-阻塞，1-非阻塞。
返回值	成功：[0,di],di: 0-低电平，1-高电平 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
di = robot.GetDI(0,0)    # 获取控制箱数字量输入
print(di)
```

3.3.6 获取工具数字量输入

原型	GetToolDI(id,block)
描述	获取工具数字量输入
参数	id:io 编号，范围[0~1]； block:0-阻塞，1-非阻塞。
返回值	成功：[0,di],di: 0-低电平，1-高电平 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
tool_di = robot.GetToolDI(1,0)    # 获取工具数字量输入
print(tool_di)
```

3.3.7 等待控制箱数字量输入

原型	WaitDI(id,status,maxtime,opt)
描述	等待控制箱数字量输入
参数	id:io 编号，范围[0~15]； status:0-关，1-开； maxtime:最大等待时间，单位[ms]； opt:超时后策略，0-程序停止并提示超时，1-忽略超时提示程序继续执行，2-一直等待；
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
```

```
robot.WaitDI(0,1,0,2)     # 一直等待控制箱数字量输入
```

3.3.8 等待控制箱多路数字量输入

原型	WaitMultiDI(mode,id,status,maxtime,opt)
描述	获取控制箱多路数字量输入
参数	mode:[0]-多路与，[1]-多路或； id:io 编号，bit0~bit7 对应 DI0~DI7，bit8~bit15 对应 CI0~CI7； status(uint16_t):bit0~bit7 对应 DI0~DI7 状态，bit8~bit15 对应 CI0~CI7 状态，位的状态[0]-关，[1]-开； maxtime:最大等待时间，单位[ms]； opt:超时后策略，0-程序停止并提示超时，1-忽略超时提示程序继续执行，2-一直等待；
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.WaitMultiDI(1,3,3,10000,2)    #   一直等待控制箱多路数字量输入
```

3.3.9 等待工具数字量输入

原型	WaitToolDI(id,status,maxtime,opt)
描述	等待末端数字量输入
参数	id:io 编号，范围[0~1]； status:0-关，1-开； maxtime:最大等待时间，单位[ms]； opt:超时后策略，0-程序停止并提示超时，1-忽略超时提示程序继续执行，2-一直等待；
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.WaitToolDI(1,1,0,2)     #   一直等待工具数字量输入
```

3.3.10 获取控制箱模拟量输入

原型	GetAI(id,block)
描述	获取控制箱模拟量输入
参数	id:io 编号，范围[0~1]； block:0-阻塞，1-非阻塞。
返回值	成功：[0,value], value:输入电流或电压值百分比，范围[0~100]对应电流值[0~20mA]或电压[0~10V]； 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
ai = robot.GetAI(0,1)    #   获取控制箱模拟量输入
print(ai)
```

3.3.11 获取工具模拟量输入

原型	GetToolAI(id,block)
描述	获取末端模拟量输入
参数	id:io 编号，范围[0]； block:0-阻塞，1-非阻塞。
返回值	成功：[0,value],value:输入电流或电压值百分比，范围[0~100]对应电流值[0~20mA]或电压[0~10V]； 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
tool_ai = robot.GetToolAI(0,1)     #    获取工具模拟量输入
print(tool_ai)
```

3.3.12 等待控制箱模拟量输入

原型	WaitAI(id,sign,value,maxtime,opt)
描述	等待控制箱模拟量输入
参数	id:io 编号，范围[0~1]； sign:0-大于，1-小于 value:输入电流或电压值百分比，范围[0~100]对应电流值[0~20mA]或电压[0~10V]； maxtime:最大等待时间，单位[ms]； opt:超时后策略，0-程序停止并提示超时，1-忽略超时提示程序继续执行，2-一直等待；
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.WaitAI(0,0,50,0,2)    #   一直等待控制箱模拟量输入
```

3.3.13 等待工具模拟量输入

原型	WaitToolAI(id,sign,value,maxtime,opt)
描述	等待末端模拟量输入
参数	id:io 编号，范围[0]； sign:0-大于，1-小于

	value:输入电流或电压值百分比，范围[0~100]对应电流值[0~20mA]或电压[0~10V]； maxtime:最大等待时间，单位[ms]； opt:超时后策略，0-程序停止并提示超时，1-忽略超时提示程序继续执行，2-一直等待。
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.WaitToolAI(0,0,50,0,2)    #   一直等待工具模拟量输入
```

3.4 机器人常用设置

3.4.1 设置全局速度

原型	SetSpeed (vel)
描述	设置全局速度
参数	vel:速度百分比，范围[0~100]
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.SetSpeed(20)    # 设置全局速度，手动模式与自动模式独立设置
```

3.4.2 设置系统变量值

原型	SetSysVarValue (id,value)
描述	设置系统变量
参数	id：变量编号，范围[1~20]; value：变量值
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
for i in range(1,21):
    robot.SetSysVarValue(i,i+0.5)     #   设置系统变量值
robot.WaitMs(1000)
for i in range(1,21):
    sys_var = robot.GetSysVarValue(i)  #   查询系统变量值
    print(sys_var)
```

3.4.3 设置工具坐标系

原型	SetToolCoord(id,t_coord ,type,install)
描述	设置工具坐标系
参数	id:坐标系编号，范围[0~14]； t_coord:工具中心点相对末端法兰中心位姿，单位[mm][°]； type:0-工具坐标系，1-传感器坐标系； install:安装位置，0-机器人末端，1-机器人外部
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
t_coord = [1.0,2.0,3.0,4.0,5.0,6.0]
robot.SetToolCoord(10,t_coord,0,0)  #  设置工具坐标系
```

3.4.4 设置工具坐标系列表

原型	SetToolList(id,t_coord ,type,install)
描述	设置工具坐标系列表
参数	id:坐标系编号，范围[0~14]； t_coord:工具中心点相对末端法兰中心位姿，单位[mm][°]； type:0-工具坐标系，1-传感器坐标系； install:安装位置，0-机器人末端，1-机器人外部
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
t_coord = [1.0,2.0,3.0,4.0,5.0,6.0]
robot.SetToolList(10,t_coord,0,0)  #  设置工具坐标系
```

3.4.5 设置外部工具坐标系

原型	SetExToolCoord(id,etcp ,etool)
描述	设置外部工具坐标系
参数	id:坐标系编号，范围[0~14]； etcp:外部工具坐标系，单位[mm][°]； etool:末端工具坐标系，单位[mm][°]；
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
```

```
robot = frrpc.RPC('192.168.58.2')
etcp = [1.0,2.0,3.0,4.0,5.0,6.0]
etool = [21.0,22.0,23.0,24.0,25.0,26.0]
robot.SetExToolCoord(10,etcp,etool)
```

3.4.6 设置外部工具坐标系列表

原型	SetExToolList(id,etcp ,etool)
描述	设置外部工具坐标系列表
参数	id:坐标系编号，范围[0~14]； etcp:外部工具坐标系，单位[mm][°]； etool:末端工具坐标系，单位[mm][°]；
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
etcp = [1.0,2.0,3.0,4.0,5.0,6.0]
etool = [21.0,22.0,23.0,24.0,25.0,26.0]
robot.SetExToolList(10,etcp,etool)
```

3.4.7 设置工件坐标系

原型	SetWObjCoord(id,w_coord)
描述	设置工件坐标系
参数	id:坐标系编号，范围[0~14]； w_coord:坐标系相对位姿，单位[mm][°]；
返回值	成功：[0,] 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
w_coord = [11.0,12.0,13.0,14.0,15.0,16.0]
robot.SetWObjCoord(11,w_coord)
```

3.4.8 设置工件坐标系列表

原型	SetWObjList(id,w_coord)
描述	设置工件坐标系列表
参数	id:坐标系编号，范围[0~14]； w_coord:坐标系相对位姿，单位[mm][°]；
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
```

```
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
w_coord = [11.0,12.0,13.0,14.0,15.0,16.0]
robot.SetWObjList(11,w_coord)
```

3.4.9 设置末端负载重量

原型	SetLoadWeight(weight)
描述	设置末端负载重量
参数	weight:单位[kg]
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.SetLoadWeight(3.0)    # 设置负载重量
```

3.4.10 设置机器人安装方式

原型	SetRobotInstallPos(method)
描述	设置机器人安装方式
参数	method:0-平装，1-侧装，2-挂装
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.SetRobotInstallPos(0)     #    设置机器人安装方式
```

3.4.11 设置机器人安装角度-自由安装

原型	SetRobotInstallAngle(yangle,zangle)
描述	设置机器人安装角度-自由安装
参数	yangle-倾斜角，zangle-旋转角
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.SetRobotInstallAngle(0.0,0.0)     #    设置机器人安装角度
```

3.4.12 设置末端负载质心坐标

原型	SetLoadCoord(x,y,z)
描述	设置末端负载质心坐标

参数	x,y,z:单位[mm]
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.SetLoadCoord(3.0,4.0,5.0)     #    设置负载质心坐标
```

3.4.13 等待指定时间

原型	WaitMs(t_ms)
描述	等待指定时间
参数	t_ms:单位[ms]。
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.WaitMs(1000)     #   等待 1000ms
```

3.5 机器人安全设置

3.5.1 设置碰撞等级

原型	SetAnticollision (mode,level,config)
描述	设置碰撞等级
参数	mode:0-等级，1-百分比； level=[j1,j2,j3,j4,j5,j6]:碰撞阈值； config:0-不更新配置文件，1-更新配置文件
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
level = [1.0,2.0,3.0,4.0,5.0,6.0]
robot.SetAnticollision(0,level,1)      #   设置碰撞等级
level = [50.0,20.0,30.0,40.0,50.0,60.0]
robot.SetAnticollision(1,level,1)      #   设置碰撞百分比
```

3.5.2 设置碰撞后策略

原型	SetCollisionStrategy (strategy)
描述	设置碰撞后策略
参数	strategy：0-报错暂停，1-继续运行

返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.SetCollisionStrategy(1)     # 设置碰撞后策略，1-继续运行
```

3.5.3 设置正限位

原型	SetLimitPositive(p_limit)
描述	设置正限位
参数	p_limit=[j1,j2,j3,j4,j5,j6]，六个关节位置
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
p_limit = [170.0,80.0,150.0,80.0,170.0,160.0]
robot.SetLimitPositive(p_limit)    #  设置正限位
```

3.5.4 设置负限位

原型	SetLimitPositive(n_limit)
描述	设置负限位
参数	n_limit=[j1,j2,j3,j4,j5,j6]，六个关节位置
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
n_limit = [-170.0,-260.0,-150.0,-260.0,-170.0,-160.0]
robot.SetLimitNegative(n_limit)    # 设置负限位
```

3.5.5 错误状态清除

原型	ResetAllError()
描述	错误状态清除，只能清除可复位的错误
参数	无
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
```

```
robot.ResetAllError()     #  错误状态清除
```

3.5.6 关节摩擦力补偿开关

原型	FrictionCompensationOnOff (state)
描述	关节摩擦力补偿开关
参数	state：0-关，1-开
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.FrictionCompensationOnOff(1)    #  关节摩擦力补偿开
```

3.5.7 设置关节摩擦力补偿系数-正装

原型	SetFrictionValue_level (coeff)
描述	设置关节摩擦力补偿系数-正装
参数	coeff=[j1,j2,j3,j4,j5,j6]，六个关节补偿系数
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.FrictionCompensationOnOff(1)    #  关节摩擦力补偿开
lcoeff = [0.9,0.9,0.9,0.9,0.9,0.9]
robot.SetFrictionValue_level(lcoeff)    #  设置关节摩擦力补偿系数
```

3.5.8 设置关节摩擦力补偿系数-侧装

原型	SetFrictionValue_wall (coeff)
描述	设置关节摩擦力补偿系数-正装
参数	coeff=[j1,j2,j3,j4,j5,j6]，六个关节补偿系数
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.FrictionCompensationOnOff(1)    #  关节摩擦力补偿开
wcoeff = [0.4,0.4,0.4,0.4,0.4,0.4]
robot.SetFrictionValue_wall(wcoeff)  #  设置关节摩擦力补偿系数
```

3.5.9 设置关节摩擦力补偿系数-倒装

原型	SetFrictionValue_ceiling (coeff)

描述	设置关节摩擦力补偿系数-正装
参数	coeff=[j1,j2,j3,j4,j5,j6]，六个关节补偿系数
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.FrictionCompensationOnOff(1)    #  关节摩擦力补偿开
ccoeff = [0.6,0.6,0.6,0.6,0.6,0.6]
robot.SetFrictionValue_ceiling(ccoeff)  #  设置关节摩擦力补偿系数
```

3.5.10 设置关节摩擦力补偿系数-自由安装

原型	SetFrictionValue_freedom (coeff)
描述	设置关节摩擦力补偿系数-正装
参数	coeff=[j1,j2,j3,j4,j5,j6]，六个关节补偿系数
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.FrictionCompensationOnOff(1)    #  关节摩擦力补偿开
fcoeff = [0.5,0.5,0.5,0.5,0.5,0.5]
robot.SetFrictionValue_freedom(fcoeff)    #  设置关节摩擦力补偿系数
```

3.6 机器人状态查询

3.6.1 获取机器人安装角度

原型	GetRobotInstallAngle ()
描述	获取机器人安装角度
参数	无
返回值	成功：[0,yangle,zangle],yangle-倾斜角，zangle-旋转角 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
ret = robot.GetRobotInstallAngle()  # 获取机器人安装角度
print(ret)
```

3.6.2 获取系统变量值

原型	GetSysVarValue (id)
描述	获取系统变量值

参数	id：系统变量编号，范围[1~20]
返回值	成功：[0,var_value] 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
for i in range(1,21):
    robot.SetSysVarValue(i,i+0.5)    #  设置系统变量值
robot.WaitMs(1000)
for i in range(1,21):
    sys_var = robot.GetSysVarValue(i)  #  查询系统变量值
    print(sys_var)
```

3.6.3 获取当前关节位置(角度)

原型	GetActualJointPosDegree(flag)
描述	获取关节当前位置(角度)
参数	flag：0-阻塞，1-非阻塞
返回值	成功：[0,joint_pos],joint_pos=[j1,j2,j3,j4,j5,j6] 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
ret = robot.GetActualJointPosDegree(0)  # 获取机器人当前关节位置
print(ret)
```

3.6.4 获取当前关节位置(弧度)

原型	GetActualJointPosRadian(flag)
描述	获取关节当前位置(弧度)
参数	flag：0-阻塞，1-非阻塞
返回值	成功：[0,joint_pos],joint_pos=[j1,j2,j3,j4,j5,j6] 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
ret = robot.GetActualJointPosRadian(0)  # 获取机器人当前关节位置
print(ret)
```

3.6.5 获取当前工具位姿

原型	GetActualTCPPose(flag)
描述	获取当前工具位姿
参数	flag：0-阻塞，1-非阻塞
返回值	成功：[0,tcp_pose],tcp_pose=[x,y,z,rx,ry,rz]

	失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
ret = robot.GetActualTCPPose(0)  # 获取机器人当前工具位姿
print(ret)
```

3.6.6 获取当前工具坐标系编号

原型	GetActualTCPNum(flag)
描述	获取当前工具坐标系编号
参数	flag：0-阻塞，1-非阻塞
返回值	成功：[0,tool_id] 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
ret = robot.GetActualTCPNum(0)  # 获取机器人当前工具坐标系编号
print(ret)
```

3.6.7 获取当前工件坐标系编号

原型	GetActualWObjNum(flag)
描述	获取当前工件坐标系编号
参数	flag：0-阻塞，1-非阻塞
返回值	成功：[0,wobj_id] 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
ret = robot.GetActualWObjNum(0)  # 获取机器人当前工件坐标系编号
print(ret)
```

3.6.8 获取当前末端法兰位姿

原型	GetActualToolFlangePose (flag)
描述	获取当前末端法兰位姿
参数	flag：0-阻塞，1-非阻塞
返回值	成功：[0,flange_pose],flange_pose=[x,y,z,rx,ry,rz] 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
```

```
ret = robot.GetActualToolFlangePose(0)  # 获取机器人当前末端法兰位姿
print(ret)
```

3.6.9 逆运动学求解

原型	GetInverseKin(type,desc_pos ,config)
描述	逆运动学，笛卡尔位姿求解关节位置
参数	type:0-绝对位姿(基坐标系)，1-相对位姿（基坐标系），2-相对位姿（工具坐标系） desc_pose:[x,y,z,rx,ry,rz],工具位姿，单位[mm][°] config:关节配置，[-1]-参考当前关节位置求解，[0~7]-依据关节配置求解
返回值	成功：[0,joint_pos],joint_pos=[j1,j2,j3,j4,j5,j6] 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
P1=[75.414,568.526,338.135,-178.348,-0.930,52.611]
ret = robot.GetInverseKin(0,P1,-1)
print(ret)
```

3.6.10 逆运动学求解

原型	GetInverseKinRef(desc_pos,joint_pos_ref)
描述	逆运动学，工具位姿求解关节位置，参考指定关节位置求解
参数	desc_pos：[x,y,z,rx,ry,rz]工具位姿，单位[mm][°] joint_pos_ref：[j1,j2,j3,j4,j5,j6]，关节参考位置，单位[°]
返回值	成功：[0,joint_pos],joint_pos=[j1,j2,j3,j4,j5,j6] 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
P1=[75.414,568.526,338.135,-178.348,-0.930,52.611]
J1=[95.442,-101.149,-98.699,-68.347,90.580,-47.174]
ret = robot.GetInverseKinRef(0,P1,J1)
print(ret)
```

3.6.11 逆运动学求解

原型	GetInverseKinHasSolution(desc_pos,joint_pos_ref)
描述	逆运动学，工具位姿求解关节位置 是否有解
参数	desc_pos：[x,y,z,rx,ry,rz]工具位姿，单位[mm][°] joint_pos_ref：[j1,j2,j3,j4,j5,j6]，关节参考位置，单位[°]
返回值	成功：[0,result],“True”-有解，“False”-无解 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
P1=[75.414,568.526,338.135,-178.348,-0.930,52.611]
J1=[95.442,-101.149,-98.699,-68.347,90.580,-47.174]
ret = robot.GetInverseKinHasSolution(0,P1,J1)
print(ret)
```

3.6.12 正运动学求解

原型	GetForwardKin(joint_pos)
描述	正运动学，关节位置求解工具位姿
参数	joint_pos:[j1,j2,j3,j4,j5,j6]:关节位置，单位[°]
返回值	成功：[0,desc_pos],desc_pos=[x,y,z,rx,ry,rz]:工具位姿，单位[mm][°] 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
J1=[95.442,-101.149,-98.699,-68.347,90.580,-47.174]
ret = robot.GetForwardKin(J1)
print(ret)
```

3.6.13 获取当前关节转矩

原型	GetJointTorques (flag)
描述	获取当前关节转矩
参数	flag：0-阻塞，1-非阻塞
返回值	成功：[0,torques],torques=[j1,j2,j3,j4,j5,j6] 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
ret = robot.GetJointTorques(0)  # 获取机器人当前关节转矩
print(ret)
```

3.6.14 获取当前负载的重量

原型	GetTargetPayload (flag)
描述	获取当前负载的质量
参数	flag：0-阻塞，1-非阻塞
返回值	成功：[0,weight],单位[kg] 失败：[errcode,]:

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
```

```
robot = frrpc.RPC('192.168.58.2')
ret = robot.GetTargetPayload(0)  # 获取机器人当前负载重量
print(ret)
```

3.6.15 获取当前负载的质心

原型	GetTargetPayloadCog(flag)
描述	获取当前负载的质心
参数	flag：0-阻塞，1-非阻塞
返回值	成功：[0,cog], cog=[x,y,z]:质心坐标，单位[mm] 失败：[errcode,]:

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
ret = robot.GetTargetPayloadCog(0)  # 获取机器人当前负载质心
print(ret)
```

3.6.16 获取当前工具坐标系

原型	GetTCPOffset (flag)
描述	获取当前工具坐标系
参数	flag：0-阻塞，1-非阻塞
返回值	成功：[0,tcp_offset], tcp_offset=[x,y,z,rx,ry,rz]:相对位姿，单位[mm][°] 失败：[errcode,]:

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
ret = robot.GetTCPOffset(0)  # 获取机器人当前工具坐标系
print(ret)
```

3.6.17 获取当前工件坐标系

原型	GetWObjOffset (flag)
描述	获取当前工件坐标系
参数	flag：0-阻塞，1-非阻塞
返回值	成功：[0,wobj_offset], wobj _offset=[x,y,z,rx,ry,rz]:相对位姿，单位[mm][°] 失败：[errcode,]:

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
ret = robot.GetWObjOffset(0)  # 获取机器人当前工件坐标系
print(ret)
```

3.6.18 获取关节软限位角度

原型	GetJointSoftLimitDeg(flag)
描述	获取关节软限位角度
参数	flag：0-阻塞，1-非阻塞
返回值	成功：[0, j1min,j1max,j2min,j2max,j3min,j3max,j4min,j4max,j5min,j5max, j6min,j6max] :轴 1~轴 6 关节负限位与正限位，单位[mm] 失败：[errcode,]:

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
ret = robot.GetJointSoftLimitDeg(0)  # 获取机器人关节软限位角度
print(ret)
```

3.6.19 获取系统时间

原型	GetSystemClock()
描述	获取系统时间
参数	无
返回值	成功：[0,t_ms]:单位[ms] 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
ret = robot.GetSystemClock()  # 获取控制器系统时间
print(ret)
```

3.6.20 获取机器人当前关节配置

原型	GetRobotCurJointsConfig ()
描述	获取机器人当前关节配置
参数	无
返回值	成功：[0,config]:范围[0~7] 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
ret = robot.GetRobotCurJointsConfig()  # 获取机器人当前关节配置
print(ret)
```

3.6.21 获取默认速度

原型	GetDefaultTransVel()
描述	获取默认速度
参数	无

返回值	成功：[0,vel]:单位[mm/s] 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
ret = robot.GetDefaultTransVel()  # 获取机器人当前速度
print(ret)
```

3.6.22 查询机器人运动是否完成

原型	GetRobotMotionDone()
描述	查询机器人运动是否完成
参数	无
返回值	成功：[0,state],state:0-未完成，1-完成 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
ret = robot.GetRobotMotionDone()     #查询机器人运动完成状态
if ret[0] == 0:
    print(ret[1])
else:
    print("the errcode is: ", ret[0])
```

3.7 机器人轨迹复现

3.7.1 设置轨迹记录参数

原型	SetTPDParam (type,name,period_ms,di_choose,do_choose)
描述	设置轨迹记录参数
参数	type：数据类型，1-关节位置； name：轨迹名； period_ms：采样周期，固定值，2ms 或 4ms 或 8ms； di_choose：DI 选择,bit0~bit7 对应控制箱 DI0~DI7，bit8~bit9 对应末端 DI0~DI1，0-不选择，1-选择 do_choose：DO 选择,bit0~bit7 对应控制箱 DO0~DO7，bit8~bit9 对应末端 DO0~DO1，0-不选择，1-选择
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
type = 1  # 数据类型，1-关节位置
```

```
name = 'tpd2023'  # 轨迹名
period = 4  #采样周期，2ms 或 4ms 或 8ms
di_choose = 0 # di 输入配置
do_choose = 0 # do 输出配置
robot.SetTPDParam(type, name, period, di_choose, do_choose)     #配置 TPD 参数
```

3.7.2 开始轨迹记录

原型	SetTPDStart(type,name,period_ms,di_choose,do_choose)
描述	开始轨迹记录
参数	type：数据类型，1-关节位置； name：轨迹名； period_ms：采样周期，固定值，2ms 或 4ms 或 8ms； di_choose：DI 选择,bit0~bit7 对应控制箱 DI0~DI7，bit8~bit9 对应末端 DI0~DI1，0-不选择，1-选择 do_choose：DO 选择,bit0~bit7 对应控制箱 DO0~DO7，bit8~bit9 对应末端 DO0~DO1，0-不选择，1-选择
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
import time
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
type = 1  # 数据类型，1-关节位置
name = 'tpd2023'  # 轨迹名
period = 4  #采样周期，2ms 或 4ms 或 8ms
di_choose = 0 # di 输入配置
do_choose = 0 # do 输出配置
robot.SetTPDParam(type, name, period, di_choose, do_choose)     #配置 TPD 参数
robot.Mode(1)  # 机器人切入手动模式
time.sleep(1)
robot.DragTeachSwitch(1)  #机器人切入拖动示教模式
robot.SetTPDStart(type, name, period, di_choose, do_choose)    # 开始记录示教轨迹
time.sleep(30)
robot.SetWebTPDStop()  # 停止记录示教轨迹
robot.DragTeachSwitch(0)  #机器人切入非拖动示教模式
```

3.7.3 停止轨迹记录

原型	SetWebTPDStop ()
描述	停止轨迹记录
参数	无
返回值	成功：[0] 失败：[errcode]

代码示例：

见 3.7.2

3.7.4 删除轨迹记录

原型	SetTPDDelete (name)
描述	删除轨迹记录
参数	name:轨迹名
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.SetTPDDelete('tpd2023')    # 删除 TPD 轨迹
```

3.7.5 轨迹预加载

原型	LoadTPD (name)
描述	轨迹预加载
参数	name:轨迹名
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
P1=[-378.9,-340.3,107.2,179.4,-1.3,125.0]
name = 'tpd2023'    #轨迹名
blend = 1    #是否平滑，1-平滑，0-不平滑
ovl = 100.0    #速度缩放
robot.LoadTPD(name)  #轨迹预加载
robot.MoveCart(P1,1,0,100.0,100.0,100.0,-1.0,-1)        #运动到起始点
robot.MoveTPD(name, blend, ovl)  #轨迹复现
```

3.7.6 轨迹复现

原型	MoveTPD (name,blend,ovl)
描述	轨迹复现
参数	name：轨迹名 blend：是否平滑，0-不平滑，1-平滑 ovl：速度缩放因子，范围[0~100]
返回值	成功：[0] 失败：[errcode]

代码示例：

见 3.7.5

3.8 机器人 WebApp 程序使用

3.8.1 设置开机自动加载默认的作业程序

原型	LoadDefaultProgConfig (flag,program_name)
描述	设置开机自动加载默认的作业程序
参数	flag：1-开机自动加载默认程序，0-不自动加载默认程序 program_name：作业程序名及路径，如“/fruser/movej.lua”，其中“/fruser/”为固定路径
返回值	成功：[0]； 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.LoadDefaultProgConfig(1, "/fruser/splineptp.lua")  # 设置开机自动加载默认的作业程序
```

3.8.2 加载指定的作业程序

原型	ProgramLoad (program_name)
描述	加载指定的作业程序
参数	program_name：作业程序名及路径，如“/fruser/movej.lua”，其中“/fruser/”为固定路径
返回值	成功：[0]； 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
#机器人 webapp 程序使用接口
robot.Mode(0)    #机器人切入自动运行模式
robot.ProgramLoad('/fruser/testPTP.lua')    #加载要执行的机器人程序，testPTP.lua 程序需要先在 webapp 上编写好
```

3.8.3 获取已加载的作业程序名

原型	GetLoadedProgram ()
描述	获取已加载的作业程序名
参数	无
返回值	成功：[0,program_name]； 失败：[errcode,]

代码示例：

```
import frrpc
import time
import _thread
def print_program_state(name,rb):
```

```
    while(1):
        pstate = robot.GetProgramState()     #查询程序运行状态,1-程序停止或无程序运行，2-程序运行中，3-程序暂停
        linenum = robot.GetCurrentLine()     #查询当前作业程序执行的行号
        name = robot.GetLoadedProgram()      #查询已加载的作业程序名
        print("the robot program state is:",pstate[1])
        print("the robot program line number is:",linenum[1])
        print("the robot program name is:",name[1])
        time.sleep(1)
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
#机器人 webapp 程序使用接口
robot.Mode(0)    #机器人切入自动运行模式
robot.ProgramLoad('/fruser/testPTP.lua')    #加载要执行的机器人程序，testPTP.lua 程序需要先在 webapp 上编写好
robot.ProgramRun()      #执行机器人程序
_thread.start_new_thread(print_program_state,("print_state",robot))
time.sleep(5)          #休息 10s
robot.ProgramPause()    #暂停正在执行的机器人程序
time.sleep(5)
robot.ProgramResume()  #恢复暂停执行的机器人程序
time.sleep(5)
robot.ProgramStop()     #停止正在执行的机器人程序
time.sleep(2)
```

3.8.4 获取当前机器人作业程序的执行行号

原型	GetCurrentLine ()
描述	获取当前机器人作业程序的执行行号
参数	无
返回值	成功：[0,line_num]； 失败：[errcode,]

代码示例：

见 3.8.3

3.8.5 运行当前加载的作业程序

原型	ProgramRun ()
描述	运行当前加载的作业程序
参数	无
返回值	成功：[0]； 失败：[errcode]

代码示例：

见 3.8.3

3.8.6 暂停当前运行的作业程序

原型	ProgramPause ()
描述	暂停当前运行的作业程序
参数	无
返回值	成功：[0]； 失败：[errcode]

代码示例：

见 3.8.3

3.8.7 恢复当前暂停的作业程序

原型	ProgramResume ()
描述	恢复当前暂停的作业程序
参数	无
返回值	成功：[0]； 失败：[errcode]

代码示例：

见 3.8.3

3.8.8 终止当前运行的作业程序

原型	ProgramStop ()
描述	终止当前运行的作业程序
参数	无
返回值	成功：[0]； 失败：[errcode]

代码示例：

见 3.8.3

3.8.9 获取机器人作业程序执行状态

原型	GetProgramState ()
描述	获取机器人作业程序执行状态
参数	无
返回值	成功：[0,state],state:1-程序停止或无程序运行，2-程序运行中，3-程序暂停 失败：[errcode,]

代码示例：

见 3.8.3

3.9 机器人外设

3.9.1 配置夹爪

原型	SetGripperConfig(company,device,softversion,bus)
描述	配置夹爪
参数	comany：夹爪厂商，1-Robotiq，2-慧灵，3-天机，4-大寰，5-知行；

	device：设备号,Robotiq(0-2F-85 系列)，慧灵(0-NK 系列,1-Z-EFG-100)；天机(0-TEG-110)，大寰(0-PGI-140)，知行(0-CTPM2F20) softvesion：软件版本号，暂不使用，默认为 0； bus：设备挂载末端总线位置，暂不使用，默认为 0；
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
import time
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.SetGripperConfig(4,0,0,1)  # 配置夹爪
time.sleep(1)
config = robot.GetGripperConfig()  # 获取夹爪配置
print(config)
robot.ActGripper(1,0)   # 夹爪复位
time.sleep(1)
robot.ActGripper(1,1)   # 夹爪激活
time.sleep(2)
robot.MoveGripper(1,100,48,46,30000,0)   # 夹爪运动
time.sleep(3)
robot.MoveGripper(1,0,50,0,30000,0)
ret = robot.GetGripperMotionDone()    # 查询夹爪运动完成状态
print(ret)
```

3.9.2 获取夹爪配置

原型	GetGripperConfig()
描述	获取夹爪配置
参数	无
返回值	成功：[0, company,device,softversion,bus],company:夹爪厂商 失败：[errcode,]

代码示例：

见 3.9.1

3.9.3 激活夹爪

原型	ActGripper (index,action)
描述	激活夹爪
参数	index:夹爪编号； action:0-复位，1-激活
返回值	成功：[0] 失败：[errcode]

代码示例：

见 3.9.1

3.9.4 控制夹爪

原型	MoveGripper (index,pos,speed,force,maxtime,block)
描述	控制夹爪
参数	index:夹爪编号； pos:位置百分比，范围[0~100]，； speed:速度百分比，范围[0~100]; force:力矩百分比，范围[0~100]； maxtime:最大等待时间，范围[0~30000]，单位[ms]； block:0-阻塞，1-非阻塞。
返回值	成功：[0] 失败：[errcode]

代码示例：

见 3.9.1

3.9.5 获取夹爪运动状态

原型	GetGripperMotionDone()
描述	获取夹爪运动状态
参数	无
返回值	成功：[0,status], status:0-运动未完成，1-运动完成 失败：[errcode,]

代码示例：

见 3.9.1

3.10 机器人力控

3.10.1 力传感器配置

原型	FT_SetConfig(company,device,softversion,bus)
描述	力传感器配置
参数	comany：传感器厂商，17-坤维科技，19-航天十一院，20-ATI 传感器，21-中科米点，22-伟航敏芯； device：设备号，坤维(0-KWR75B)，航天十一院(0-MCS6A-200-4)，ATI(0-AXIA80-M8)，中科米点(0-MST2010)， 伟航敏芯(0-WHC6L-YB-10A)； softvesion：软件版本号，暂不使用，默认为 0； bus：设备挂载末端总线位置，暂不使用，默认为 0；
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
company = 17    #传感器厂商，17-坤维科技
device = 0      #传感器设备号
softversion = 0 #软件版本号
bus = 1         #末端总线位置
```

```
robot.FT_SetConfig(company, device, softversion, bus)    #配置力传感器
config = robot.FT_GetConfig() #获取力传感器配置信息，厂商编号下发比反馈大 1
print(config)
```

3.10.2 获取力传感器配置

原型	FT_GetConfig()
描述	获取力传感器配置
参数	无
返回值	成功：[0, company,device,softversion,bus],company:传感器厂商 失败：[errcode,]

代码示例：

见 3.10.1

3.10.3 力传感器激活

原型	FT_Activate(state)
描述	力传感器激活
参数	state：0-复位，1-激活
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.FT_Activate(0)  #传感器复位
time.sleep(1)
robot.FT_Activate(1)  #传感器激活
time.sleep(1)
```

3.10.4 力传感器校零

原型	FT_SetZero(state)
描述	力传感器校零
参数	state：0-去除零点，1-零点矫正
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.FT_SetZero(0)    #传感器去除零点
time.sleep(1)
robot.FT_SetZero(1)    #传感器零点矫正,注意此时末端不能安装工具，只有力传感器
time.sleep(1)
```

3.10.5 设置力传感器参考坐标系

原型	FT_SetRCS(ref)
描述	设置力传感器参考坐标系
参数	ref：0-工具坐标系，1-基坐标系
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
robot.FT_SetRCS(0)     #设置参考坐标系为工具坐标系，0-工具坐标系，1-基坐标系
time.sleep(1)
```

3.10.6 负载重量辨识记录

原型	FT_PdIdenRecord(tool_id)
描述	负载重量辨识记录
参数	tool_id：传感器坐标系编号，范围[0~14]
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
import time
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
#负载辨识，此时末端安装要辨识的工具，工具安装在力传感器下方,末端竖直向下
robot.FT_SetRCS(0)     #设置参考坐标系为工具坐标系，0-工具坐标系，1-基坐标系
time.sleep(1)
tool_id = 10  #传感器坐标系编号
tool_coord = [0.0,0.0,35.0,0.0,0.0,0.0]    # 传感器相对末端法兰位姿
tool_type = 1  # 0-工具，1-传感器
tool_install = 0 # 0-安装末端，1-机器人外部
robot.SetToolCoord(tool_id,tool_coord,tool_type,tool_install)      #设置传感器坐
标系，传感器相对末端法兰位姿
time.sleep(1)
robot.FT_PdIdenRecord(tool_id)    #记录辨识数据
time.sleep(1)
weight = robot.FT_PdIdenCompute()  #计算负载重量，单位 kg
print(weight)
```

3.10.7 负载重量辨识计算

原型	FT_PdIdenCompute()
描述	负载重量辨识计算
参数	无
返回值	成功：[0,weight] ,weight-负载重量，单位 kg

	失败：[errcode,]

代码示例：

见 3.10.6

3.10.8 负载质心辨识记录

原型	FT_PdCogIdenRecord(tool_id)
描述	负载质心辨识记录
参数	tool_id：传感器坐标系编号，范围[0~14]
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
import time
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
#负载质心辨识，机器人需要示教三个不同的姿态，然后记录辨识数据，最后计算负载质心
P1=[-160.619,-586.138,384.988,-170.166,-44.782,169.295]
robot.MoveCart(P1,9,0,100.0,100.0,100.0,-1.0,-1)          #关节空间点到点运动
time.sleep(1)
robot.FT_PdCogIdenRecord(tool_id,1)                                  #记录辨识数据
time.sleep(1)
P2=[-87.615,-606.209,556.119,-102.495,10.118,178.985]
robot.MoveCart(P2,9,0,100.0,100.0,100.0,-1.0,-1)
time.sleep(1)
robot.FT_PdCogIdenRecord(tool_id,2)
time.sleep(1)
P3=[41.479,-557.243,484.407,-125.174,46.995,-132.165]
robot.MoveCart(P3,9,0,100.0,100.0,100.0,-1.0,-1)
time.sleep(1)
robot.FT_PdCogIdenRecord(tool_id,3)
time.sleep(1)
cog = robot.FT_PdCogIdenCompute()    # 计算辨识的负载质心
print(cog)
```

3.10.9 负载质心辨识计算

原型	FT_PdCogIdenCompute ()
描述	负载质心辨识计算
参数	无
返回值	成功：[0,cog],cog=[cogx,cogy,cogz]，负载质心，单位 mm 失败：[errcode,]

代码示例：

见 3.10.8

3.10.10获取参考坐标系下力/扭矩数据

原型	FT_GetForceTorqueRCS()

描述	获取参考坐标系下力/扭矩数据
参数	无
返回值	成功：[0,data] ,data=[fx,fy,fz,mx,my,mz] 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
rcs = robot.FT_GetForceTorqueRCS()  #查询传感器坐标系下数据
print(rcs)
```

3.10.11获取力传感器原始力/扭矩数据

原型	FT_GetForceTorqueOrigin()
描述	获取力传感器原始力/扭矩数据
参数	无
返回值	成功：[0,data] ,data=[fx,fy,fz,mx,my,mz] 失败：[errcode,]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
origin = robot.FT_GetForceTorqueOrigin()   #查询传感器原始数据
print(origin)
```

3.10.12碰撞守护

原型	FT_Guard(flag,sensor_num,select,force_torque,max_threshold,min_threshold)
描述	碰撞守护
参数	flag：0-关闭碰撞守护，1-开启碰撞守护； sensor_num：力传感器编号； select：六个自由度是否检测碰撞[fx,fy,fz,mx,my,mz]，0-不生效，1-生效； force_torque：碰撞检测力/力矩，单位 N 或 Nm； max_threshold：最大阈值； min_threshold：最小阈值； 力/力矩检测范围:(force_torque-min_threshold,force_torque+max_threshold)
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
actFlag = 1   #开启标志，0-关闭碰撞守护，1-开启碰撞守护
sensor_num = 1  #力传感器编号
is_select = [1,1,1,1,1,1]  #六个自由度选择[fx,fy,fz,mx,my,mz]，0-不生效，1-生效
```

```
force_torque = [0.0,0.0,0.0,0.0,0.0,0.0]  #碰撞检测力和力矩，检测范围
（force_torque-min_threshold,force_torque+max_threshold）
max_threshold = [10.0,10.0,10.0,10.0,10.0,10.0]  #最大阈值
min_threshold = [5.0,5.0,5.0,5.0,5.0,5.0]   #最小阈值
P1=[-160.619,-586.138,384.988,-170.166,-44.782,169.295]
P2=[-87.615,-606.209,556.119,-102.495,10.118,178.985]
P3=[41.479,-557.243,484.407,-125.174,46.995,-132.165]
robot.FT_Guard(actFlag, sensor_num, is_select, force_torque, max_threshold,
min_threshold)    #开启碰撞守护
robot.MoveCart(P1,9,0,100.0,100.0,100.0,-1.0,-1)          #关节空间点到点运动
robot.MoveCart(P2,9,0,100.0,100.0,100.0,-1.0,-1)
robot.MoveCart(P3,9,0,100.0,100.0,100.0,-1.0,-1)
actFlag = 0
robot.FT_Guard(actFlag, sensor_num, is_select, force_torque, max_threshold,
min_threshold)    #关闭碰撞守护
```

3.10.13恒力控制

原型	FT_Control(flag,sensor_num,select,force_torque,gain,adj_sign,ILC_sign, max_dis,max_ang)
描述	恒力控制
参数	flag：恒力控制开启标志，0-关，1-开； sensor_num：力传感器编号； select：六个自由度是否检测 [fx,fy,fz,mx,my,mz]，0-不生效，1-生效； force_torque：检测力/力矩，单位 N 或 Nm； gain：[f_p,f_i,f_d,m_p,m_i,m_d],力 PID 参数，力矩 PID 参数； adj_sign：自适应启停状态，0-关闭，1-开启； ILC_sign: ILC 控制启停状态，0-停止，1-训练，2-实操； max_dis：最大调整距离； max_ang：最大调整角度；
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
status = 1  #恒力控制开启标志，0-关，1-开
sensor_num = 1 #力传感器编号
is_select = [0,0,1,0,0,0]  #六个自由度选择[fx,fy,fz,mx,my,mz]，0-不生效，1-生效
force_torque = [0.0,0.0,-10.0,0.0,0.0,0.0]  #碰撞检测力和力矩，检测范围
（force_torque-min_threshold,force_torque+max_threshold）
gain = [0.0005,0.0,0.0,0.0,0.0,0.0]  #最大阈值
adj_sign = 0  #自适应启停状态，0-关闭，1-开启
ILC_sign = 0  #ILC 控制启停状态，0-停止，1-训练，2-实操
max_dis = 100.0  #最大调整距离
max_ang = 0.0  #最大调整角度
```

```
J1=[-68.987,-96.414,-111.45,-61.105,92.884,11.089]
P1=[62.795,-511.979,291.697,-179.545,3.027,-170.039]
eP1=[0.000,0.000,0.000,0.000]
dP1=[0.000,0.000,0.000,0.000,0.000,0.000]
J2=[-107.596,-109.154,-104.735,-56.176,90.739,11.091]
P2=[-294.768,-503.708,233.158,179.799,0.713,151.309]
eP2=[0.000,0.000,0.000,0.000]
dP2=[0.000,0.000,0.000,0.000,0.000,0.000]
robot.MoveJ(J1,P1,9,0,100.0,180.0,100.0,eP1,-1.0,0,dP1)      #关节空间运动 PTP,工具号 9，实际测试根据现场数据及工具号使用
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)    #恒力控制
robot.MoveL(J2,P2,9,0,100.0,180.0,20.0,-1.0,eP2,0,0,dP2)    #笛卡尔空间直线运动
status = 0
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
```

3.10.14螺旋线探索

原型	FT_SpiralSearch(rcs,dr,fFinsih,t,vmax)
描述	螺旋线探索
参数	rcs：参考坐标系，0-工具坐标系，1-基坐标系 dr：每圈半径进给量，单位 mm； fFinish：力或力矩阈值(0~100)，单位 N 或 Nm； t：最大探索时间，单位 ms； vmax：线速度最大值，单位 mm/s；
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
#恒力参数
status = 1  #恒力控制开启标志，0-关，1-开
sensor_num = 1 #力传感器编号
is_select = [0,0,1,0,0,0]  #六个自由度选择[fx,fy,fz,mx,my,mz]，0-不生效，1-生效
force_torque = [0.0,0.0,-10.0,0.0,0.0,0.0]  #碰撞检测力和力矩，检测范围（force_torque-min_threshold,force_torque+max_threshold）
gain = [0.0001,0.0,0.0,0.0,0.0,0.0]  #最大阈值
adj_sign = 0  #自适应启停状态，0-关闭，1-开启
ILC_sign = 0  #ILC 控制启停状态，0-停止，1-训练，2-实操
max_dis = 100.0  #最大调整距离
max_ang = 5.0  #最大调整角度
#螺旋线探索参数
rcs = 0  #参考坐标系，0-工具坐标系，1-基坐标系
dr = 0.7  #每圈半径进给量，单位 mm
```

```
fFinish = 1.0 #力或力矩阈值（0~100），单位 N 或 Nm
t = 60000.0 #最大探索时间，单位 ms
vmax = 3.0 #线速度最大值，单位 mm/s
is_select = [0,0,1,1,1,0]  #六个自由度选择[fx,fy,fz,mx,my,mz]，0-不生效，1-生效
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_s
ign,max_dis,max_ang)
robot.FT_SpiralSearch(rcs,dr,fFinish,t,vmax)
status = 0
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_s
ign,max_dis,max_ang)
```

3.10.15旋转插入

原型	FT_RotInsertion(rcs,angVelRot,forceInsertion,angleMax,orn,angAccmax,rotorn)
描述	旋转插入
参数	rcs：参考坐标系，0-工具坐标系，1-基坐标系； angVelRot：旋转角速度，单位°/s； forceInsertion：力或力矩阈值(0~100)，单位 N 或 Nm； angleMax：最大旋转角度，单位°； orn：力的方向，1-fz,2-mz； angAccmax：最大旋转加速度，单位°/s^2，暂不使用； rotorn：旋转方向，1-顺时针，2-逆时针；
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
#恒力参数
status = 1  #恒力控制开启标志，0-关，1-开
sensor_num = 1 #力传感器编号
is_select = [0,0,1,0,0,0]  #六个自由度选择[fx,fy,fz,mx,my,mz]，0-不生效，1-生效
force_torque = [0.0,0.0,-10.0,0.0,0.0,0.0]  #碰撞检测力和力矩，检测范围
（force_torque-min_threshold,force_torque+max_threshold）
gain = [0.0001,0.0,0.0,0.0,0.0,0.0]  #最大阈值
adj_sign = 0  #自适应启停状态，0-关闭，1-开启
ILC_sign = 0  #ILC 控制启停状态，0-停止，1-训练，2-实操
max_dis = 100.0  #最大调整距离
max_ang = 5.0  #最大调整角度
#旋转插入参数
rcs = 0  #参考坐标系，0-工具坐标系，1-基坐标系
angVelRot = 2.0  #旋转角速度，单位°/s
forceInsertion = 1.0 #力或力矩阈值（0~100），单位 N 或 Nm
angleMax= 45 #最大旋转角度，单位°
orn = 1 #力的方向，1-fz,2-mz
angAccmax = 0.0 #最大旋转角加速度，单位°/s^2,暂不使用
```

```
rotorn = 1 #旋转方向，1-顺时针，2-逆时针
s_select = [0,0,1,1,1,0]  #六个自由度选择[fx,fy,fz,mx,my,mz]，0-不生效，1-生效
force_torque = [0.0,0.0,-10.0,0.0,0.0,0.0]  #碰撞检测力和力矩，检测范围（force_torque-min_threshold,force_torque+max_threshold）
gain = [0.0001,0.0,0.0,0.0,0.0,0.0]  #最大阈值
status = 1
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
robot.FT_RotInsertion(rcs,angVelRot,forceInsertion,angleMax,orn,angAccmax,rotorn)
status = 0
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
```

3.10.16直线插入

原型	FT_LinInsertion(rcs,force_goal,lin_v,lin_a,disMax,linorn)
描述	直线插入
参数	rcs：参考坐标系，0-工具坐标系，1-基坐标系； force_goal：力或力矩阈值(0~100)，单位 N 或 Nm； lin_v：直线速度，单位 mm/s； lin_a：直线加速度，单位 mm/s^2，暂不使用； disMax：最大插入距离，单位 mm； linorn：插入方向，1-正方向，0-负方向；
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
#恒力参数
status = 1  #恒力控制开启标志，0-关，1-开
sensor_num = 1 #力传感器编号
is_select = [0,0,1,0,0,0]  #六个自由度选择[fx,fy,fz,mx,my,mz]，0-不生效，1-生效
force_torque = [0.0,0.0,-10.0,0.0,0.0,0.0]  #碰撞检测力和力矩，检测范围（force_torque-min_threshold,force_torque+max_threshold）
gain = [0.0001,0.0,0.0,0.0,0.0,0.0]  #最大阈值
adj_sign = 0  #自适应启停状态，0-关闭，1-开启
ILC_sign = 0  #ILC 控制启停状态，0-停止，1-训练，2-实操
max_dis = 100.0  #最大调整距离
max_ang = 5.0  #最大调整角度
#直线插入参数
rcs = 0  #参考坐标系，0-工具坐标系，1-基坐标系
force_goal = 20.0  #力或力矩阈值（0~100），单位 N 或 Nm
lin_v = 0.0 #直线速度，单位 mm/s
lin_a = 0.0 #直线加速度，单位 mm/s^2,暂不使用
```

```
disMax = 100.0 #最大插入距离，单位 mm
linorn = 1 #插入方向，1-正方向，2-负方向
is_select = [1,1,1,0,0,0]  #六个自由度选择[fx,fy,fz,mx,my,mz]，0-不生效，1-生效
gain = [0.00005,0.0,0.0,0.0,0.0,0.0]  #最大阈值
force_torque = [0.0,0.0,-30.0,0.0,0.0,0.0]  #碰撞检测力和力矩，检测范围（force_torque-min_threshold,force_torque+max_threshold）
status = 1
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
robot.FT_LinInsertion(rcs,force_goal,lin_v,lin_a,disMax,linorn)
status = 0
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
```

3.10.17表面定位

原型	FT_FindSurface (rcs,dir,axis,lin_v,lin_a,disMax,force_goal)
描述	表面定位
参数	rcs： 参考坐标系，0-工具坐标系，1-基坐标系； dir：移动方向，1-正方向，2-负方向； axis：移动轴，1-x，2-y，3-z； lin_v：探索直线速度，单位 mm/s； lin_a：探索直线加速度，单位 mm/s^2； disMax：最大探索距离，单位 mm force_goal：动作终止力阈值，单位 N；
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
#恒力控制
status = 1  #恒力控制开启标志，0-关，1-开
sensor_num = 1 #力传感器编号
is_select = [1,0,0,0,0,0]  #六个自由度选择[fx,fy,fz,mx,my,mz]，0-不生效，1-生效
force_torque = [-2.0,0.0,0.0,0.0,0.0,0.0]  #碰撞检测力和力矩，检测范围（force_torque-min_threshold,force_torque+max_threshold）
gain = [0.0002,0.0,0.0,0.0,0.0,0.0]  #最大阈值
adj_sign = 0  #自适应启停状态，0-关闭，1-开启
ILC_sign = 0  #ILC 控制启停状态，0-停止，1-训练，2-实操
max_dis = 15.0  #最大调整距离
max_ang = 0.0  #最大调整角度
#表面定位参数
rcs = 0 #参考坐标系，0-工具坐标系，1-基坐标系
direction = 1 #移动方向，1-正方向，2-负方向
axis = 1 #移动轴，1-X,2-Y,3-Z
```

```
lin_v = 3.0  #探索直线速度，单位 mm/s
lin_a = 0.0  #探索直线加速度，单位 mm/s^2
disMax = 50.0 #最大探索距离，单位 mm
force_goal = 2.0 #动作终止力阈值，单位 N
P1=[-230.959,-364.017,226.179,-179.004,0.002,89.999]
robot.MoveCart(P1,9,0,100.0,100.0,100.0,-1.0,-1)        #关节空间点到点运动
#x 方向寻找中心
#第 1 个表面
robot.FT_CalCenterStart()
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_s
ign,max_dis,max_ang)
robot.FT_FindSurface(rcs,direction,axis,lin_v,lin_a,disMax,force_goal)
status = 0
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_s
ign,max_dis,max_ang)
robot.MoveCart(P1,9,0,100.0,100.0,100.0,-1.0,-1)        #关节空间点到点运动
robot.WaitMs(1000)
#第 2 个表面
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_s
ign,max_dis,max_ang)
direction = 2 #移动方向，1-正方向，2-负方向
robot.FT_FindSurface(rcs,direction,axis,lin_v,lin_a,disMax,force_goal)
status = 0
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_s
ign,max_dis,max_ang)
#计算 x 方向中心位置
xcenter= robot.FT_CalCenterEnd()
print(xcenter)
xcenter = [xcenter[1],xcenter[2],xcenter[3],xcenter[4],xcenter[5],xcenter[6]]
robot.MoveCart(xcenter,9,0,60.0,50.0,50.0,0.0,-1)
#y 方向寻找中心
#第 1 个表面
robot.FT_CalCenterStart()
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_s
ign,max_dis,max_ang)
direction = 1 #移动方向，1-正方向，2-负方向
axis = 2 #移动轴，1-X,2-Y,3-Z
disMax = 150.0 #最大探索距离，单位 mm
lin_v = 6.0  #探索直线速度，单位 mm/s
robot.FT_FindSurface(rcs,direction,axis,lin_v,lin_a,disMax,force_goal)
status = 0
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_s
ign,max_dis,max_ang)
robot.MoveCart(P1,9,0,100.0,100.0,100.0,-1.0,-1)        #关节空间点到点运动
robot.WaitMs(1000)
```

```
#第 2 个表面
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_s
ign,max_dis,max_ang)
direction = 2 #移动方向，1-正方向，2-负方向
robot.FT_FindSurface(rcs,direction,axis,lin_v,lin_a,disMax,force_goal)
status = 0
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_s
ign,max_dis,max_ang)
#计算 y 方向中心位置
ycenter=robot.FT_CalCenterEnd()
print(ycenter)
ycenter = [ycenter[1],ycenter[2],ycenter[3],ycenter[4],ycenter[5],ycenter[6]]
robot.MoveCart(ycenter,9,0,60.0,50.0,50.0,-1.0,-1)
```

3.10.18计算中间平面位置开始

原型	FT_CalCenterStart()
描述	计算中间平面位置开始
参数	无
返回值	成功：[0] 失败：[errcode]

代码示例：

见 3.10.17

3.10.19计算中间平面位置结束

原型	FT_CalCenterEnd()
描述	计算中间平面位置结束
参数	无
返回值	成功：[0,pos] ,pos=[x,y,z,rx,ry,rz] 失败：[errcode,]

代码示例：

见 3.10.17

3.10.20柔顺控制开启

原型	FT_ComplianceStart(p,force)
描述	柔顺控制开启
参数	p: 位置调节系数或柔顺系数 force：柔顺开启力阈值，单位 N
返回值	成功：[0] 失败：[errcode]

代码示例：

```
import frrpc
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = frrpc.RPC('192.168.58.2')
J1=[-105.3,-68.0,-127.9,-75.5,90.8,77.8]
```

```
P1=[-208.9,-274.5,334.6,178.8,-1.3,86.7]
eP1=[0.000,0.000,0.000,0.000]
dP1=[0.000,0.000,0.000,0.000,0.000,0.000]
J2=[-105.3,-97.9,-101.5,-70.3,90.8,77.8]
P2=[-264.8,-480.5,341.8,179.2,0.3,86.7]
eP2=[0.000,0.000,0.000,0.000]
dP2=[0.000,0.000,0.000,0.000,0.000,0.000]
#恒力控制参数
status = 1  #恒力控制开启标志，0-关，1-开
sensor_num = 1 #力传感器编号
is_select = [1,1,1,0,0,0]  #六个自由度选择[fx,fy,fz,mx,my,mz]，0-不生效，1-生效
force_torque = [-10.0,-10.0,-10.0,0.0,0.0,0.0]  #碰撞检测力和力矩，检测范围（force_torque-min_threshold,force_torque+max_threshold）
gain = [0.0005,0.0,0.0,0.0,0.0,0.0]  #最大阈值
adj_sign = 0  #自适应启停状态，0-关闭，1-开启
ILC_sign = 0  #ILC 控制启停状态，0-停止，1-训练，2-实操
max_dis = 1000.0  #最大调整距离
max_ang = 0.0  #最大调整角度
#柔顺控制
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
p = 0.00005  #位置调节系数或柔顺系数
force = 30.0 #柔顺开启力阈值，单位 N
robot.FT_ComplianceStart(p,force)
count = 15  #循环次数
while(count):
    robot.MoveL(J1,P1,9,0,100.0,180.0,100.0,-1.0,eP1,0,1,dP1)    #笛卡尔空间直线运动
    robot.MoveL(J2,P2,9,0,100.0,180.0,100.0,-1.0,eP2,0,0,dP2)
    count = count - 1
robot.FT_ComplianceStop()
status = 0
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
```

3.10.21柔顺控制关闭

原型	FT_ComplianceStop()
描述	柔顺控制关闭
参数	无
返回值	成功：[0] 失败：[errcode]

代码示例：

见 3.10.20

4 附录

表 1 接口返回值错误码对照表

errcode	描述	处理方式
-1	其他错误	联系售后工程师查看控制器日志
0	调用成功	
3	接口参数个数不一致	检查接口参数个数
4	接口参数值异常	检查参数值类型或范围
8	轨迹文件打开失败	检查 TPD 轨迹文件是否存在或轨迹名是否正确
14	接口执行失败	检查 web 界面是否报故障或状态反馈是否报故障
18	机器人程序正在运行，请先停止	先停止程序，再进行其他操作
25	数据异常，计算失败	重新标定或辨识
28	逆运动学计算结果异常	检查位姿是否合理
29	ServoJ 关节超限	检查关节数据是否在合理范围
30	不可复位故障，请断电重启控制箱	请断电重启控制箱
34	工件号错误	请检查工件号是否合理
36	文件名过长	请缩减文件名长度
38	奇异位姿，计算失败	请更换位姿
64	未加入指令队列	联系售后工程师查看控制器日志
66	整圆/螺旋线指令中间点 1 错误	检查中间点 1 数据是否正确
67	整圆/螺旋线指令中间点 2 错误	检查中间点 2 数据是否正确
68	整圆/螺旋线指令中间点 3 错误	检查中间点 3 数据是否正确
69	圆弧指令中间点错误	检查中间点数据是否正确
70	圆弧指令目标点错误	检查目标点数据是否正确
73	夹爪运动报错	检查夹爪通信状态是否正常
74	直线指令点错误	检查点位数据是否正确
75	通道错误	检查 IO 编号是否在范围内
76	等待超时	检查 IO 信号是否输入或接线是否正确
82	TPD 指令点错误	重新记录示教轨迹
83	TPD 指令工具与当前工具不符	更改为 TPD 示教时所用的工具坐标系
94	样条指令点错误	检查点位数据是否正确
108	螺旋线指令起点错误	检查起点数据是否正确
112	给定位姿无法到达	检查目标位姿是否合理