



# Dobot TCP/IP 远程控制 接口文档



# 目录

前言

1 概述

2 通用指令

2.1 控制相关指令

2.2 设置相关指令

2.3 计算和获取相关指令

2.4 IO相关指令

2.5 Modbus相关指令

2.6 总线寄存器相关指令

2.7 运动相关指令

2.8 轨迹恢复指令

2.9 日志导出指令

2.10 力控指令

3 实时反馈信息

4 通用错误码

5 各状态下允许执行的TCP指令

# 前言

## 目的

本手册介绍了Dobot工业机器人控制柜V4版本TCP/IP二次开发接口及其使用方式，帮助用户了解和开发基于TCP/IP的机器人控制软件。

## 读者对象

本手册适用于：

- 客户
- 销售工程师
- 安装调试工程师
- 技术支持工程师

## 修订记录

时间	版本号	修订记录
2024/12/26	V4.6.0	1. 对应六轴机器人控制器4.6.0版本。 2. 新增RequestControl指令、轨迹恢复指令、日志导出指令、力控指令、运动相关指令RelPointTool、RelPointUser、RelJoint。 3. 新增通用错误码-8，新增各状态下允许执行的TCP指令。 4. 修正StartPath指令格式。 5. 修正DO/DI/ToolDO/ToolDI/ToolAI指令的index索引范围。 6. 修正ServoP、ServoJ指令的运行时间范围。 7. 优化实时反馈信息。
2024/08/15	V4.5.1	修正ServoJ指令示例，补充ServoJ和ServoP的返回值。
2024/03/25	V4.5.1	对应六轴机器人控制器4.5.1版本。
2023/10/19	V4.5.0	对应六轴机器人控制器4.5.0版本，新增CreateTray、GetTrayPoint、ServoJ、ServoP指令，优化部分说明。
2023/07/26	V4.4.0	对应六轴机器人控制器4.4.0版本。
2023/05/12	V4.3.0	对应六轴机器人控制器4.3.0版本。
2023/02/17	V4.2.0	对应六轴机器人控制器4.2.0版本。

# 1 概述

由于基于TCP/IP的通讯具有成本低、可靠性高、实用性强、性能高等特点，许多工业自动化项目对基于TCP/IP协议控制机器人的需求广泛，因此Dobot机器人在TCP/IP协议的基础上，提供了丰富的接口用于与外部设备的交互。

## 端口说明

根据设计，Dobot机器人会开启29999、30004、30005以及30006服务器端口；

- 29999服务器端口：上位机可以通过29999端口直接发送**控制指令**给机器人，或者**主动获取**机器人的某些状态，这些功能被称为Dashboard。
- 30004、30005以及30006服务器端口：30004端口即实时反馈端口，客户端**每8ms**能收到一次机器人实时状态信息。30005端口**每200ms**反馈机器人的信息。30006端口为**可配置**的反馈机器人信息端口(默认为**每50ms**反馈，如需修改，请联系技术支持)。通过实时反馈端口每次收到的数据包有1440个字节，这些字节以标准的格式排列。

## 消息格式

消息命令与消息应答都是 ASCII 码格式(字符串形式)。

上位机**下发消息**格式如下：

```
消息名称(Param1,Param2,Param3.....ParamN)
```

由消息名称和参数组成，参数放在括号内，每一个参数之间以英文逗号“,”相隔，一个完整的消息以右括号结束。

TCP/IP远程控制指令不区分大小写格式，如以下三种写法都会被识别为使能机器人的指令：

- ENABLEROBOT()
- enablerobot()
- eNabLErobOt()

机器人收到命令后，会返回**应答消息**，格式如下：

```
ErrorID,{value,...,valueN},消息名称(Param1,Param2,Param3.....ParamN);
```

- ErrorID 为0时表示命令接收成功，返回非0则代表命令有错误，详见[通用错误码](#)；
- {value,...,valueN} 表示返回值，没有返回值则返回{}；
- 消息名称(Param1,Param2,Param3.....ParamN) 为下发的命令消息。

例如：

下发:

```
MovL(-500,100,200,150,0,90)
```

返回:

```
0,{},MovL(-500,100,200,150,0,90);
```

0表示接收成功，{}表示没有返回值。

下发:

```
Mov(-500,100,200,150,0,90)
```

返回:

```
-10000,{},Mov(-500,100,200,150,0,90);
```

-10000表示命令不存在，{}表示没有返回值。

## 队列指令与立即指令

- 队列指令：系统会等待之前的指令队列执行完毕后再执行这条指令。例如，DO指令之前是一串运动指令，系统会等待机器人运动完毕后再设置DO。
- 立即指令：系统会无视指令队列，在读到这条指令后立刻执行。例如，DOInstant指令之前是一串运动指令，系统不会等待机器人运动完毕，而是在读到这条指令后立刻设置DO。

如无特殊说明，读取输入的指令都是立即指令。

**本文档中的示例均为伪代码，无法直接运行，仅用于说明如何使用接口。**

## 2 通用指令

- 2.1 控制相关指令
- 2.2 设置相关指令
- 2.3 计算和获取相关指令
- 2.4 IO相关指令
- 2.5 Modbus相关指令
- 2.6 总线寄存器相关指令
- 2.7 运动相关指令
- 2.8 轨迹恢复指令
- 2.9 日志导出指令
- 2.10 力控指令

## 2.1 控制相关指令

### 指令列表

指令	功能	指令类型
RequestControl	请求将设备控制模式切换为TCP模式	立即指令
PowerOn	机器人上电	立即指令
EnableRobot	使能机器人	立即指令
DisableRobot	下使能机器人	立即指令
ClearError	清除机器人报警	立即指令
RunScript	运行指定工程	立即指令
Stop	停止运动（或正在运行的工程）	立即指令
Pause	暂停运动（或正在运行的工程）	立即指令
Continue	继续运动（或已暂停的工程）	立即指令
EmergencyStop	紧急停止机器人	立即指令
BrakeControl	控制指定关节的抱闸	立即指令
StartDrag	机器人进入关节拖拽模式	立即指令
StopDrag	机器人退出拖拽模式	立即指令

### RequestControl

#### 原型

```
RequestControl()
```

#### 描述

请求将设备控制模式切换为TCP模式。只有在TCP模式下才可执行其他TCP指令。

仅当机器人处于**未上电**或**下使能**（且非暂停或松抱闸状态）时才可切换TCP模式。

#### 返回

```
ErrorID, {}, RequestControl();
```

## 示例

```
RequestControl()
```

请求切换TCP模式。

## 允许切换TCP模式的场景

控制器状态	是否允许切换TCP模式
未上电	允许
下使能（非暂停状态、非抱闸松开状态）	允许
使能空闲	不允许
拖拽模式	不允许
单次运动中	不允许
运行中	不允许
暂停	不允许
错误（上使能情况下）	不允许
松抱闸	不允许
开启手自动模式	不允许

## PowerOn

### 原型

```
PowerOn()
```

### 描述

机器人上电。机器人上电到完成，需要大概10秒钟的时间，然后再进行使能操作。请勿在机器人开机初始化完成前下发控制信号，否则可能会造成机器人异常动作。

### 返回

```
ErrorID, {}, PowerOn();
```

## 示例

```
PowerOn()
```



控制机器人上电。

## EnableRobot

### 原型

```
EnableRobot(load,centerX,centerY,centerZ,isCheck)
```

### 描述

使能机器人。

### 可选参数

参数名	类型	说明
load	double	设置负载重量，取值范围不能超过各个型号机器人的负载范围。单位：kg。
centerX	double	X方向偏心距离，单位：mm。
centerY	double	Y方向偏心距离，单位：mm。
centerZ	double	Z方向偏心距离，单位：mm。
isCheck	int	是否检查负载。1表示检查，0表示不检查。如果设置为1，则机械臂使能后会检查实际负载是否和设置负载一致，如果不一致会自动下使能。默认值为0。

可携带的参数数量如下：

- 0：不携带参数，表示使能时不设置负载重量和偏心参数。
- 1：携带一个参数，该参数表示负载重量。
- 4：携带四个参数，分别表示负载重量和偏心参数。
- 5：携带五个参数，分别表示负载重量、偏心参数和是否检查负载。

### 返回

```
ErrorID,{},EnableRobot(load,centerX,centerY,centerZ,isCheck);
```

### 示例1

```
EnableRobot()
```

使能机器人，不设置负载重量和偏心参数。

### 示例2

```
EnableRobot(1.5)
```

使能机器人并设置负载重量1.5kg。

### 示例3

```
EnableRobot(1.5,0,0,30.5)
```

使能机器人并设置负载重量1.5kg，Z方向偏心30.5mm，不检查负载。

### 示例4

```
EnableRobot(1.5,0,0,30.5,1)
```

使能机器人并设置负载重量1.5kg，Z方向偏心30.5mm，检查负载。

## DisableRobot

### 原型

```
DisableRobot()
```

### 描述

下使能机器人。

### 返回

```
ErrorID, {}, DisableRobot();
```

### 示例

```
DisableRobot()
```

下使能机器人。

## ClearError

### 原型

```
ClearError()
```

### 描述

清除机器人报警。清除报警后，用户可以根据RobotMode判断机器人是否还处于报警状态。部分报警需要解决报警原因或者重启控制柜后才能清除。

## 返回

```
ErrorID, {}, ClearError();
```

## 示例

```
uint64_t robotMode = parseRobotMode(RobotMode()); // parseRobotMode用于获取RobotMode指令返回的值
, 请自行实现
if(robotMode=9){
    ClearError()
}
```

清除机器人报警。

# RunScript

## 原型

```
RunScript(projectName)
```

## 描述

运行指定工程。如果需要运行工程后立即暂停，则需要在下发RunScript指令后至少间隔1s再下发Pause指令。

## 必选参数

参数名	类型	说明
projectName	string	工程文件的名称。 如果名称包含中文，必须将发送端的编码方式设置为UTF-8，否则会导致中文接收异常。如果名称为纯数字，则需加上双引号。

## 返回

```
ErrorID, {}, RunScript(projectName);
```

## 示例1

```
RunScript(demo)
```

运行名称为demo的脚本工程。

## 示例2

```
RunScript("123")
```

运行名称为123的脚本工程。

## 示例3

```
RunScript("blockly_test")
```

DobotStudio Pro在保存积木编程工程时会自动添加“blockly\_”前缀。例如在DobotStudio Pro中保存了一个名为“test”的积木编程工程，则执行该指令时工程名称需要指定为"blockly\_test"。

# Stop

## 原型

```
Stop()
```

## 描述

停止已下发的运动指令队列或者RunScript指令运行的工程。

## 返回

```
ErrorID, {}, Stop();
```

## 示例

```
Stop()
```

停止点动、脚本运行、关节运动等一系列运动。

# Pause

## 原型

```
Pause()
```

## 描述

暂停已下发的运动指令队列或者RunScript指令运行的工程。

## 返回

```
ErrorID, {}, Pause();
```

## 示例

```
Pause()
```

暂停movj()等一系列运动，使机器人处于暂停状态，点动不可暂停。

# Continue

## 原型

```
Continue()
```

## 描述

继续已暂停的运动指令队列或者RunScript指令运行的工程。

## 返回

```
ErrorID, {}, Continue();
```

## 示例

```
Continue()
```

继续运动，暂停状态下的算法队列指令可继续运动，机器人处于运行状态。

# EmergencyStop

## 原型

```
EmergencyStop(mode)
```

## 描述

紧急停止机器人。急停后机器人会下使能并报警，需要松开急停、清除报警后才能重新使能。

## 必选参数

参数名	类型	说明
mode	int	急停操作模式。1表示按下急停，0表示松开急停。

## 返回

```
ErrorID,{},EmergencyStop(mode);
```

## 示例

```
EmergencyStop(1)
```

紧急停止机器人。

# BrakeControl

## 原型

```
BrakeControl(axisID,value)
```

## 描述

控制指定关节的抱闸。机器人静止时关节会自动抱闸，如果用户需进行关节拖拽操作，可开启抱闸，即在机器人下使能状态，手动扶住关节后，下发开启抱闸的指令。

仅能在机器人下使能时控制关节抱闸，否则ErrorID会返回-1。

## 必选参数

参数名	类型	说明
axisID	int	关节轴序号，1表示J1轴，2表示J2轴，以此类推。 取值范围：[1,6]。

value	int	设置抱闸状态。 0表示抱闸锁死（关节不可移动），1表示松开抱闸（关节可移动）。
-------	-----	--

## 返回

```
ErrorID, {}, BrakeControl(axisID, value);
```

## 示例

```
BrakeControl(1, 1)
```

松开关节1的抱闸。

# StartDrag

## 原型

```
StartDrag()
```

## 描述

机器人进入关节拖拽模式。机器人处于报警状态下时，无法通过该指令进入关节拖拽模式。

## 返回

```
ErrorID, {}, StartDrag();
```

## 示例

```
StartDrag()
```

机器人进入关节拖拽模式。

# StopDrag

## 原型

```
StopDrag()
```

## 描述

机器人退出拖拽模式。关节拖拽和力控拖拽均使用此指令退出。

## 返回

```
ErrorID,{}),StopDrag());
```

## 示例

```
StopDrag()
```

机器人退出拖拽模式。



## 2.2 设置相关指令

### 指令列表

指令	功能	指令类型
SpeedFactor	设置全局速度比例	立即指令
User	设置全局用户坐标系	队列指令
SetUser	修改指定的用户坐标系	立即指令
CalcUser	计算用户坐标系	立即指令
Tool	设置全局工具坐标系	队列指令
SetTool	修改指定的工具坐标系	立即指令
CalcTool	计算工具坐标系	立即指令
SetPayload	设置机械臂末端负载	队列指令
AccJ	设置关节运动方式的加速度比例	立即指令
AccL	设置直线和弧线运动方式的加速度比例	立即指令
VelJ	设置关节运动方式的速度比例	立即指令
VelL	设置直线和弧线运动方式的速度比例	立即指令
CP	设置平滑过渡比例	立即指令
SetCollisionLevel	设置碰撞检测等级	队列指令
SetBackDistance	设置碰撞回退距离	队列指令
SetPostCollisionMode	设置碰撞后处理方式	队列指令
DragSensitivity	设置拖拽灵敏度	立即指令
EnableSafeSkin	开启或关闭安全皮肤功能	队列指令
SetSafeSkin	设置安全皮肤各个部位的灵敏度	队列指令
SetSafeWallEnable	开启或关闭指定的安全墙	队列指令
SetWorkZoneEnable	开启或关闭指定的安全区域	队列指令

#### 说明:

如无特殊说明，TCP指令设置的参数均只在本次TCP/IP控制模式中生效。

# SpeedFactor

## 原型

```
SpeedFactor(ratio)
```

## 描述

设置全局速度比例。

- 机器人点动时实际运动加速度/速度比例 = 控制软件点动设置中的值 x 全局速度比例。

例：控制软件设置的关节速度为 $12^{\circ}/s$ ，全局速率为50%，则实际点动速度为 $12^{\circ}/s \times 50\% = 6^{\circ}/s$

- 机器人再现时实际运动加速度/速度比例 = 运动指令可选参数设置的的比例 x 控制软件再现设置中的值 x 全局速度比例。

例：控制软件设置的坐标系速度为 $2000\text{mm}/s$ ，全局速率为50%，运动指令设置的速率为80%，则实际运动速度为 $2000\text{mm}/s \times 50\% \times 80\% = 800\text{mm}/s$

未设置时沿用进入TCP/IP控制模式前控制软件设置的值。

## 必选参数

参数名	类型	说明
ratio	int	全局运动速度比例，取值范围：[1, 100]。

## 返回

```
ErrorID,{ },SpeedFactor(ratio);
```

## 示例

```
SpeedFactor(80)
```

设置全局运动速度比例为80%。

# User

## 原型

```
User(index)
```

## 描述

设置全局用户坐标系。用户下发运动指令时可选择用户坐标系，如未指定，则会使用全局用户坐标系。

未设置时默认的全局用户坐标系为用户坐标系0。

#### 必选参数

参数名	类型	说明
index	int	选择已标定的用户坐标系索引。需要通过控制软件等方式标定后才可在此处通过索引选择。取值范围：[0,50]。

#### 返回

```
ErrorID,{ResultID},User(index);
```

若ErrorID返回-1，表示设置失败。ResultID为算法队列ID，可用于判断指令执行顺序。

#### 示例

```
User(1)
```

设置用户坐标系1为全局用户坐标系。

## SetUser

#### 原型：

```
SetUser(index,value,type)
```

#### 描述：

修改指定的用户坐标系。

#### 必选参数：

参数名	类型	说明
index	int	选择已标定的用户坐标系索引。需要通过控制软件等方式标定后才可在此处通过索引选择。取值范围：[1,50]。
value	string	修改后的用户坐标系，格式为{x, y, z, rx, ry, rz}。建议使用 <a href="#">CalcUser</a> 指令获取。

### 可选参数：

参数名	类型	说明
type	int	是否使坐标系改动全局生效。 0：该命令修改的坐标系仅在当前工程运行中生效，退出TCP模式后恢复为原来的值。 1：该命令修改的坐标系将会被控制器保存，退出TCP模式后依然保持修改后的值。

### 返回：

```
ErrorID, {}, SetUser(index, table, type);
```

### 示例：

```
SetUser(1, {10, 10, 10, 0, 0, 0})
```

修改用户坐标系1为x=10, y=10, z=10, rx=0, ry=0, rz=0。

## CalcUser

### 原型：

```
CalcUser(index, matrix, offset)
```

### 描述：

计算用户坐标系。

### 必选参数：

参数名	类型	说明
index	int	选择已标定的用户坐标系索引。需要通过控制软件等方式标定后才可在 此处通过索引选择。取值范围：[0,50]。
matrix	int	计算的方向。 1表示左乘，即index指定的坐标系沿基坐标系偏转offset指定的值。 0表示右乘，即index指定的坐标系沿自己偏转offset指定的值。
offset	string	格式为{x, y, z, rx, ry, rz}，表示用户坐标系的偏移值。

### 返回：

```
ErrorID, {x, y, z, rx, ry, rz}, CalcUser(index, matrix, offset);
```

其中{x, y, z, rx, ry, rz}为计算得出的用户坐标系。

### 示例1:

```
newUser = CalcUser(1,1,{10,10,10,10,10,10})
```

计算用户坐标系1左乘{10,10,10,0,0,0}后的值。计算过程可等价于：一个初始位姿与用户坐标系1相同的坐标系，沿基坐标系平移{x=10, y=10, z=10}并旋转{rx=10, ry=10, rz=10}后，得到的新坐标系为newUser。

### 示例2:

```
newUser = CalcUser(1,0,{10,10,10,10,10,10})
```

计算用户坐标系1右乘{10,10,10,0,0,0}后的值。计算过程可等价于：一个初始位姿与用户坐标系1相同的坐标系，沿用户坐标系1平移{x=10, y=10, z=10}并旋转{rx=10, ry=10, rz=10}后，得到的新坐标系为newUser。

## Tool

### 原型

```
Tool(index)
```

### 描述

设置全局工具坐标系。用户下发运动指令时可选择工具坐标系，如未指定，则会使用全局工具坐标系。

未设置时默认的全局工具坐标系为工具坐标系0。

### 必选参数

参数名	类型	说明
index	int	选择已标定的工具坐标系索引。 需要通过控制软件等方式标定后才可在此处通过索引选择。取值范围：[0,50]。

### 返回

```
ErrorID,{ResultID},Tool(index);
```

若ErrorID返回-1，表示设置的工具坐标系索引不存在；ResultID为算法队列ID，可用于判断指令执行顺序。

## 示例

```
Tool(1)
```

设置工具坐标系1为全局工具坐标系。

## SetTool

### 原型：

```
SetTool(index,value,type)
```

### 描述：

修改指定的工具坐标系。

### 必选参数：

参数名	类型	说明
index	int	选择已标定的工具坐标系索引。需要通过控制软件等方式标定后才可在此处通过索引选择。取值范围：[1,50]。
value	string	修改后的工具坐标系，格式为{x, y, z, rx, ry, rz}。表示该坐标系相对默认工具坐标系的偏移量。

### 可选参数：

参数名	类型	说明
type	int	是否使坐标系改动全局生效。 0：该命令修改的坐标系仅在当前工程运行中生效，退出TCP模式后恢复为原来的值。 1：该命令修改的坐标系将会被控制器保存，退出TCP模式后依然保持修改后的值。

### 返回：

```
ErrorID,{},SetTool(index,table,type);
```

### 示例：

```
SetTool(1,{10,10,10,0,0,0})
```

修改工具坐标系1为x=10, y=10, z=10, rx=0, ry=0, rz=0。

# CalcTool

原型:

```
CalcTool(index,matrix,offset)
```

描述:

计算工具坐标系。

必选参数:

参数名	类型	说明
index	int	选择已标定的工具坐标系索引。需要通过控制软件等方式标定后才可在 此处通过索引选择。取值范围： [0,50]。
matrix	int	计算的方向。 1表示左乘，即index指定的工具坐标系沿法兰坐标系偏转offset指定的 值。 0表示右乘，即index指定的工具坐标系沿自己偏转offset指定的值。
offset	string	格式为{x, y, z, rx, ry, rz}，表示工具坐标系的偏移值。

返回:

```
ErrorID,{x,y,z,rx,ry,rz},CalcTool(index,matrix,offset);
```

其中{x, y, z, rx, ry, rz}为计算得出的工具坐标系。

示例1:

```
CalcTool(1,1,{10,10,10,0,0,0})
```

计算工具坐标系1左乘{10,10,10,0,0,0}后的值。计算过程可等价于：一个初始位姿与工具坐标系1  
相同的坐标系，沿法兰坐标系平移{x=10, y=10, z=10}并旋转{rx=10, ry=10, rz=10}后，得到的  
新坐标系为newTool。

示例2:

```
CalcTool(1,0,{10,10,10,0,0,0})
```

计算工具坐标系1右乘{10,10,10,0,0,0}后的值。计算过程可等价于：一个初始位姿与工具坐标系1  
相同的坐标系，沿工具坐标系1平移{x=10, y=10, z=10}并旋转{rx=10, ry=10, rz=10}后，得到的  
新坐标系为newTool。

# SetPayload

## 原型

```
SetPayload(load,x,y,z)
```

```
SetPayload(name)
```

## 描述

设置机器人末端负载，支持两种设置方式。

**方式一：**直接设置负载参数

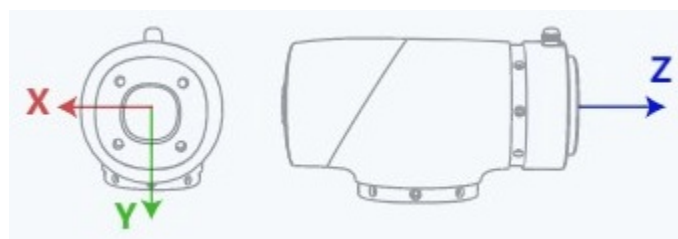
### 必选参数1

参数名	类型	说明
load	double	设置负载重量，取值范围不能超过各个型号机器人的负载范围。 单位：kg。

### 可选参数1

参数名	类型	说明
x	double	末端负载X轴偏心坐标，单位：mm。
y	double	末端负载Y轴偏心坐标，单位：mm。
z	double	末端负载Z轴偏心坐标，单位：mm。

需同时设置或不设置这三个参数。偏心坐标为负载（含治具）的质心在默认工具坐标系下的坐标，参考下图。

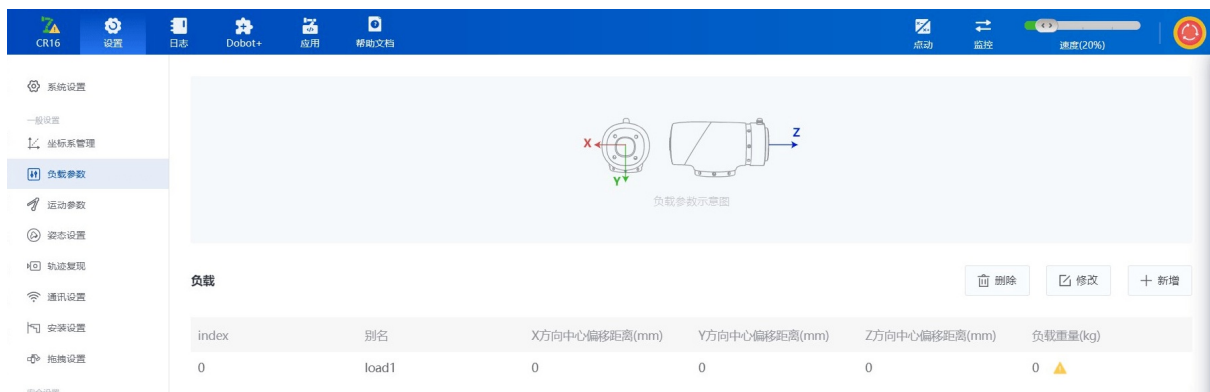


**方式二：**通过控制软件保存的预设负载参数组设置

### 必选参数2

参数名	类型	说明
name	string	控制软件保存的预设负载参数组的名称。





## 返回

```
ErrorID,{ResultID},SetPayload(load,x,y,z);
```

```
ErrorID,{ResultID},SetPayload(name);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

## 示例1

```
SetPayload(3,10,10,10)
```

设置末端负载重量为3kg，偏心坐标为{10,10,10}。

## 示例2

```
SetPayload("Load1")
```

加载名称为“Load1”的预设负载参数组。

# AccJ

## 原型

```
AccJ(R)
```

## 描述

设置关节运动方式的加速度比例。

未设置时默认值为100。

### 必选参数

参数名	类型	说明
R	int	关节加速度比例。取值范围：[1,100]

### 返回

```
ErrorID,{},AccJ(R);
```

### 示例

```
AccJ(50)
```

设置关节运动方式的加速度比例为50%。

## AccL

### 原型

```
AccL(R)
```

### 描述

设置直线和弧线运动方式的加速度比例。

未设置时默认值为100。

### 必选参数

参数名	类型	说明
R	int	加速度比例。取值范围：[1,100]

### 返回

```
ErrorID,{},AccL(R);
```

### 示例

```
AccL(50)
```

设置直线和弧线运动方式的加速度比例为50%。

## VelJ

## 原型

```
VelJ(R)
```

## 描述

设置关节运动方式的速度比例。

未设置时默认值为100。

## 必选参数

参数名	类型	说明
R	int	速度比例。取值范围：[1,100]

## 返回

```
ErrorID, {}, VelJ(R);
```

## 示例

```
VelJ(50)
```

设置关节运动方式的速度比例为50%。

# VelL

## 原型

```
VelL(R)
```

## 描述

设置直线和弧线运动方式的速度比例。

未设置时默认值为100。

## 必选参数

参数名	类型	说明
R	int	速度比例。取值范围：[1,100]

## 返回

```
ErrorID, {}, VelL(R);
```

## 示例

```
VelL(50)
```

设置直线和弧线运动方式的速度比例为50%。

## CP

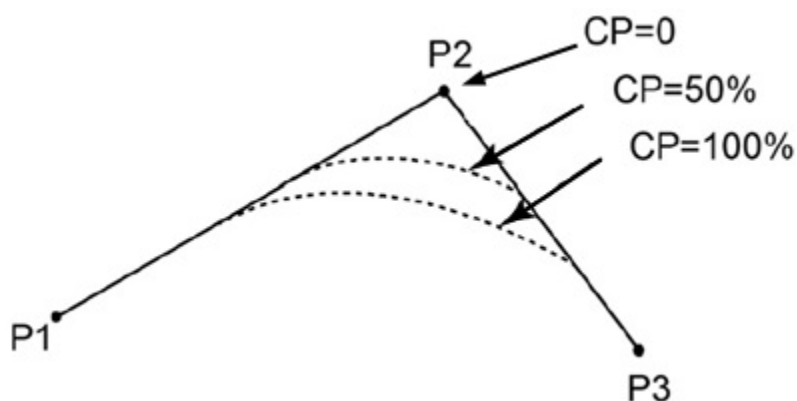
### 原型

```
CP(R)
```

### 描述

设置平滑过渡比例，即机器人连续运动经过多个点时，经过中间点是以直角方式过渡还是以曲线方式过渡。

未设置时默认值为0。



### 必选参数

参数名	类型	说明
R	int	平滑过渡比例。取值范围：[0, 100]

### 返回

```
ErrorID, {}, CP(R);
```

## 示例

```
CP(50)
```

设置平滑过渡比例为50%。

## SetCollisionLevel

### 原型

```
SetCollisionLevel(level)
```

### 描述

设置碰撞检测等级。

未设置时沿用进入TCP/IP控制模式前控制软件设置的值。

### 必选参数

参数名	类型	说明
level	int	碰撞检测等级，取值范围：[0,5]。 0表示关闭碰撞检测，1~5数字越大灵敏度越高。

### 返回

```
ErrorID,{ResultID},SetCollisionLevel(level);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

### 示例

```
SetCollisionLevel(1)
```

设置碰撞检测等级为1。

## SetBackDistance

### 原型：

```
SetBackDistance(distance)
```

### 描述：

设置机器人检测到碰撞后原路回退的距离。

未设置时沿用进入TCP/IP控制模式前控制软件设置的值。

### 必选参数：

参数名	类型	说明
distance	double	碰撞回退的距离，取值范围：[0,50]，单位：mm。

### 返回

```
ErrorID,{ResultID},SetBackDistance(distance)
```

ResultID为算法队列ID，可用于判断指令执行顺序。

### 示例：

```
SetBackDistance(20)
```

设置碰撞回退距离为20mm。

## SetPostCollisionMode

### 原型：

```
SetPostCollisionMode(mode)
```

### 描述：

设置机器人检测到碰撞后进入的状态。

未设置时沿用进入TCP/IP控制模式前控制软件设置的值。

### 必选参数：

参数名	类型	说明
mode	int	碰撞后处理方式。 0表示检测到碰撞后进入停止状态，1表示检测到碰撞后进入暂停状态。

### 返回

```
ErrorID,{ResultID},SetPostCollisionMode(mode)
```

ResultID为算法队列ID，可用于判断指令执行顺序。

### 示例：

```
SetPostCollisionMode(0)
```

设置机器人检测到碰撞后进入停止状态。

## DragSensitivity

### 原型

```
DragSensitivity(index,value)
```

### 描述

设置拖拽灵敏度。

未设置时沿用进入TCP/IP控制模式前控制软件设置的值。

### 必选参数

参数名	类型	说明
index	int	轴序号，取值范围：[0,6]。 0表示所有轴设置为相同的灵敏度。 1~6分别表示设置J1~J6轴的灵敏度。
value	int	拖拽灵敏度，值越小，拖拽时的阻力越大。取值范围：[1, 90]。

### 返回

```
ErrorID, {}, DragSensitivity(index,value);
```

### 示例

```
DragSensitivity(0,50)
```

设置所有轴的拖拽灵敏度为50。

## EnableSafeSkin

### 原型

```
EnableSafeSkin(status)
```

### 描述

开启或关闭安全皮肤功能。仅对安装了安全皮肤的机器人有效。

### 必选参数

参数名	类型	说明
status	int	电子皮肤功能开关，0表示关闭，1表示开启。

### 返回

```
ErrorID,{ResultID},EnableSafeSkin(status);
```

ResultID为算法队列ID，可用于判断指令执行顺序。若返回ErrorID为-1，可能是当前无电子皮肤。

### 示例

```
EnableSafeSkin(1)
```

开启电子皮肤功能。

## SetSafeSkin

### 原型

```
SetSafeSkin(part,status)
```

### 描述

设置安全皮肤各个部位的灵敏度。仅对安装了安全皮肤的机器人有效。

未设置时沿用进入TCP/IP控制模式前控制软件设置的值。

### 必选参数

参数名	类型	说明
part	int	要设置的部位，3表示arm（小臂安全皮肤），4~6分别表示J4~J6关节。
status	int	灵敏度，0表示关闭，1表示low，2表示middle，3表示high。

### 返回

```
ErrorID,{ResultID},SetSafeSkin(part,status);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

### 示例



```
SetSafeSkin(3,1)
```

设置小臂的电子皮肤为低灵敏度。

## SetSafeWallEnable

**原型：**

```
SetSafeWallEnable(index,value)
```

**描述：**

开启或关闭指定的安全墙。

**必选参数：**

参数名	类型	说明
index	int	要设置的安全墙索引，需要先在控制软件中添加对应的安全墙。 取值范围：[1,8]。
value	int	安全墙开关，0表示关闭，1表示开启。

**返回**

```
ErrorID,{ResultID},SetSafeWallEnable(index,value);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

**示例：**

```
SetSafeWallEnable(1,1)
```

开启索引为1的安全墙。

## SetWorkZoneEnable

**原型：**

```
SetWorkZoneEnable(index,value)
```

**描述：**

开启或关闭指定的安全区域。

### 必选参数：

参数名	类型	说明
index	int	要设置的安全区域索引，需要先在控制软件中添加对应的安全区域。 取值范围：[1,6]。
value	int	安全区域开关，0表示关闭，1表示开启。

### 返回

```
ErrorID,{ResultID},SetWorkZoneEnable(index,value);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

### 示例：

```
SetWorkZoneEnable(1,1)
```

开启索引为1的安全区域。

## 2.3 计算和获取相关指令

### 指令列表

指令	功能	指令类型
RobotMode	获取机器人当前状态	立即指令
PositiveKin	进行正解运算	立即指令
InverseKin	进行逆解运算	立即指令
GetAngle	获取机械臂当前位姿的关节坐标	立即指令
GetPose	获取机械臂当前位姿在指定的坐标系下的笛卡尔坐标	立即指令
GetErrorID	获取机器人当前报错的错误码	立即指令
CreateTray	创建托盘	立即指令
GetTrayPoint	获取托盘点	立即指令

### RobotMode

#### 原型

```
RobotMode()
```

#### 描述

获取机器人当前状态。

#### 返回

```
ErrorID,{Value},RobotMode();
```

Value取值范围如下：

取值	定义	说明
1	ROBOT_MODE_INIT	初始化状态
2	ROBOT_MODE_BRAKE_OPEN	有任意关节的抱闸松开
3	ROBOT_MODE_POWEROFF	机械臂下电状态
4	ROBOT_MODE_DISABLED	未使能（无抱闸松开）

5	ROBOT_MODE_ENABLE	使能且空闲
6	ROBOT_MODE_BACKDRIVE	拖拽模式（关节拖拽或力控拖拽）
7	ROBOT_MODE_RUNNING	运行状态(工程，TCP队列运动等)
8	ROBOT_MODE_SINGLE_MOVE	单次运动状态（点动、RunTo等）
9	ROBOT_MODE_ERROR	有未清除的报警。此状态优先级最高。 无论机械臂处于什么状态，有报警时都返回9
10	ROBOT_MODE_PAUSE	暂停状态
11	ROBOT_MODE_COLLISION	碰撞检测触发状态

### 示例

```
RobotMode()
```

获取机器人当前状态。

## PositiveKin

### 原型

```
PositiveKin(J1,J2,J3,J4,J5,J6,user,tool)
```

### 描述

进行正解运算：给定机器人各关节角度，计算机器人末端在给定的笛卡尔坐标系中的坐标值。

### 必选参数

参数名	类型	说明
J1	double	J1轴位置，单位：度。
J2	double	J2轴位置，单位：度。
J3	double	J3轴位置，单位：度。
J4	double	J4轴位置，单位：度。
J5	double	J5轴位置，单位：度。
J6	double	J6轴位置，单位：度。

## 可选参数

参数名	类型	说明
user	string	格式为"user=index", index为已标定的用户坐标系索引。不指定时使用全局用户坐标系。取值范围: [0,50]。
tool	string	格式为"tool=index", index为已标定的工具坐标系索引。不指定时使用全局工具坐标系。取值范围: [0,50]。

## 返回

```
ErrorID,{x,y,z,a,b,c},PositiveKin(J1,J2,J3,J4,J5,J6,user,tool);
```

{x,y,z,a,b,c}为点位的笛卡尔坐标值。

## 示例

```
PositiveKin(0,0,-90,0,90,0,user=1,tool=1)
```

关节坐标为{0,0,-90,0,90,0}，计算机器人末端在用户坐标系1和工具坐标系1下的笛卡尔坐标。

# InverseKin

## 原型

```
InverseKin(X,Y,Z,Rx,Ry,Rz,useJointNear, jointNear,user,tool)
```

## 描述

进行逆解运算：给定机器人末端在给定的笛卡尔坐标系中的坐标值，计算机器人各关节角度。

由于笛卡尔坐标仅定义了TCP的空间坐标与倾斜角，所以机器人可以通过多种不同的姿态到达同一个位姿，意味着一个位姿变量可以对应多个关节变量。为得出唯一的解，系统需要一个指定的关节坐标，选择最接近该关节坐标的解作为逆解结果。

## 必选参数

参数名	类型	说明
X	double	X轴位置，单位：mm。
Y	double	Y轴位置，单位：mm。
Z	double	Z轴位置，单位：mm。
Rx	double	Rx轴位置，单位：度。
Ry	double	Ry轴位置，单位：度。

Rz	double	Rz轴位置，单位：度。
----	--------	-------------

### 可选参数

参数名	类型	说明
useJointNear	string	格式为 "useJointNear=value"，用于设置JointNear参数是否有效。 "useJointNear=0"或不携带表示JointNear参数无效，系统根据机械臂当前关节角度就近选解。 "useJointNear=1"表示根据JointNear就近选解。 仅携带该参数而不携带JointNear时，该参数无效。
jointNear	string	格式为"jointNear={j1,j2,j3,j4,j5,j6}"，用于就近选解的关节坐标。
user	string	格式为"user=index"，index为已标定的用户坐标系索引。 不指定时使用全局用户坐标系。取值范围：[0,50]。
tool	string	格式为"tool=index"，index为已标定的工具坐标系索引。 不指定时使用全局工具坐标系。取值范围：[0,50]。

### 返回

```
ErrorID,{J1,J2,J3,J4,J5,J6},InverseKin(X,Y,Z,Rx,Ry,Rz,useJointNear,jointNear,user,tool);
```

{J1,J2,J3,J4,J5,J6}为点位的关节坐标值。

### 示例

```
InverseKin(473.000000,-141.000000,469.000000,-180.000000,0.000,-90.000)
```

机器人末端在全局用户坐标系和全局关节坐标系下的笛卡尔坐标为{473,-141,469,-180,0,-90}，计算关节坐标，选择机器人当前关节角度的最近解。

## GetAngle

### 原型

```
GetAngle()
```

### 描述

获取机器人当前位姿的关节坐标。

### 返回

```
ErrorID,{J1,J2,J3,J4,J5,J6},GetAngle();
```

{J1, J2, J3, J4, J5, J6}表示机器人当前位姿的关节坐标。

### 示例

```
GetAngle()
```

获取机器人当前位姿的关节坐标。

## GetPose

### 原型

```
GetPose(user,tool)
```

### 描述

获取机器人当前位姿在指定的坐标系下的笛卡尔坐标。

### 可选参数

参数名	类型	说明
user	string	格式为"user=index", index为已标定的用户坐标系索引。 取值范围: [0,50]。
tool	string	格式为"tool=index", index为已标定的工具坐标系索引。 取值范围: [0,50]。

必须同时传或同时不传, 不传时默认为全局用户和工具坐标系。

### 返回

```
ErrorID, {X,Y,Z,Rx,Ry,Rz}, GetPose(user,tool);
```

{X,Y,Z,Rx,Ry,Rz}表示机器人当前位姿的笛卡尔坐标。

### 示例

```
GetPose(user=1,tool=1)
```

获取机器人当前位姿在用户坐标系1和工具坐标系1下的笛卡尔坐标。

## GetErrorID

### 原型

```
GetErrorID()
```

## 描述

获取机器人当前报错的错误码。

## 返回

```
ErrorID, {[id,...,id], [id], [id], [id], [id], [id], [id]}, GetErrorID();
```

- [id,...,id]为控制器以及算法报警信息，无报警时返回[]，有多个报警时以英文逗号“,”相隔。
- 后面六个[id]分别表示机器人六个伺服的报警信息，无报警时返回[]。

## 示例

```
GetErrorID()
```

获取机器人当前报错的错误码。

# CreateTray

## 原型：

```
CreateTray(Trayname, {Count}, {P1,P2}) -- 一维托盘  
CreateTray(Trayname, {row,col}, {P1,P2,P3,P4}) -- 二维托盘  
CreateTray(Trayname, {row,col,layer}, {P1,P2,P3,P4,P5,P6,P7,P8}) -- 三维托盘
```

## 描述：

创建托盘，支持创建一维、二维和三维的托盘。最多可创建20个托盘，创建同名的托盘时会覆盖已有的托盘，不会增加托盘数量。

## 必选参数：

参数名	类型	说明
Trayname	string	托盘名称，最长32字节的字符串，不允许为纯数字或者纯空格。

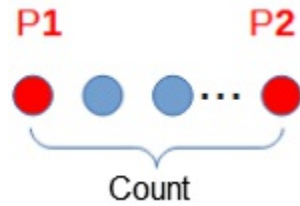
后两个参数为table变量，根据要创建的托盘维度不同，table内的值的数量不同，下文分别进行介绍。

- 创建一维托盘：一维托盘是在一条直线上等距分布的一组点。

参数名	类型	说明
{Count}	table	Count表示点位数量，取值范围：[2, 50]，输入非整数会自动向下取整。

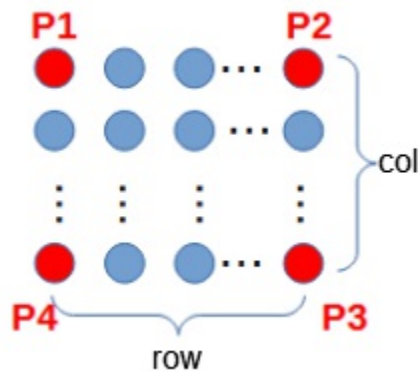


{P1,P2}	table	P1和P2分别为一维托盘的2个端点，每个点的格式都为pose = {x,y,z,rx,ry,rz}。
---------	-------	--



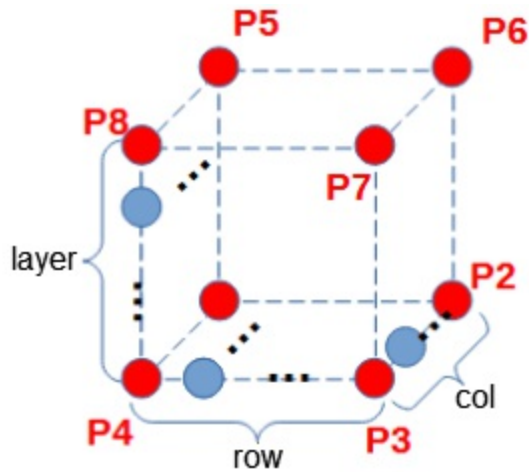
- 创建二维托盘：二维托盘是在一个平面上阵列分布的一组点。

参数名	类型	说明
{row,col}	table	row表示行方向（P1到P2方向）上点位的数量，col表示列方向（P1到P4方向）上点位的数量，取值范围都与一维托盘的Count相同。
{P1,P2,P3,P4}	table	P1、P2、P3、P4分别为二维托盘的4个顶点，每个点的格式都格式为pose = {x,y,z,rx,ry,rz}。



- 创建三维托盘：三维托盘是在空间上立体分布的一组点，可视为竖向排布的多个二维托盘。

参数名	类型	说明
{row,col,layer}	table	row表示行方向（P1到P2方向）上点位的数量，col表示列方向（P1到P4方向）上点位的数量，layer表示层数（P1到P5方向）。
{P1,P2,P3,P4,P5,P6,P7,P8}	table	P1~P8分别为三维托盘的8个顶点，每个点的格式都格式为pose = {x,y,z,rx,ry,rz}。



返回

```
ErrorID, {}, CreateTray( ... );
```

示例：

```
-- 创建名称为t1的5个点的一维托盘。
CreateTray(t1, {5}, {pose = {x1,y1,z1,rx1,ry1,rz1},pose = {x2,y2,z2,rx2,ry2,rz2}})
-- 创建名称为t2的4x5的二维托盘,下方示例中P1到P4均为pose = {x,y,z,rx,ry,rz}格式的点位。
CreateTray(t2, {4,5}, {P1,P2,P3,P4})
-- 创建名称为t3的4x5x6的三维托盘,下方示例中P1到P8均为pose = {x,y,z,rx,ry,rz}格式的点位。
CreateTray(t2, {4,5,6}, {P1,P2,P3,P4,P5,P6,P7,P8})
```

## GetTrayPoint

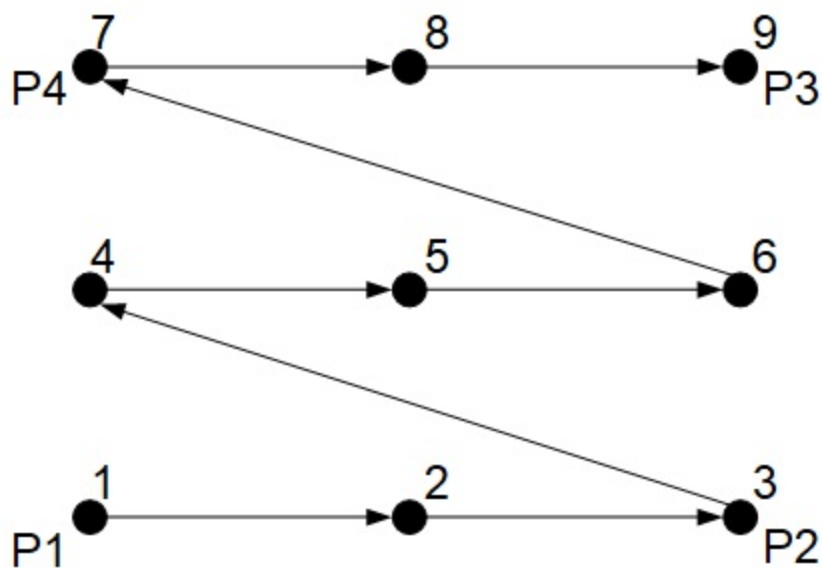
原型：

```
GetTrayPoint(Trayname,index)
```

描述：

获取指定托盘指定序号的点位。点位序号和创建托盘时传入的点位顺序有关。

- 一维托盘：P1点序号为1，P2点序号与点位数量相同，以此类推。
- 二维托盘：下图以3x3的托盘为例说明示教点与点位序号的关系。



- 三维托盘：参考二维托盘，第二层的第一个点的序号为第一层最后一个点的序号加一，以此类推。

#### 必选参数：

参数名	类型	说明
Trayname	string	已创建的托盘名称，最长32字节的字符串。
index	int	要获取的点位的序号。

#### 返回：

```
ErrorID,{isErr,x,y,z,rx,ry,rz},GetTrayPoint(Trayname,index);
```

isErr表示获取点位的结果，0表示获取成功，-1表示获取失败。

x,y,z,rx,ry,rz为获取到的点位坐标。

#### 示例：

```
-- 获取名称为t1的托盘的序号为3的点位。
GetTrayPoint(t1,3)
```

# 2.4 IO相关指令

## 指令列表

指令	功能	指令类型
DO	设置数字输出端口状态	队列指令
DOInstant	设置数字输出端口状态	立即指令
GetDO	获取数字输出端口状态	立即指令
DOGroup	设置多个数字输出端口状态	队列指令
GetDOGroup	获取多个数字输出端口状态	立即指令
ToolDO	设置末端数字输出端口状态	队列指令
ToolDOInstant	设置末端数字输出端口状态	立即指令
GetToolDO	获取末端数字输出端口状态	立即指令
AO	设置模拟输出端口的值	队列指令
AOInstant	设置模拟输出端口的值	立即指令
GetAO	获取模拟输出端口的值	立即指令
DI	获取DI端口的状态	立即指令
DIGroup	获取多个DI端口的状态	立即指令
ToolDI	获取末端DI端口的状态	立即指令
AI	获取AI端口的值	立即指令
ToolAI	获取末端AI端口的值	立即指令
SetTool485	设置末端485通信格式	立即指令
SetToolPower	设置末端工具供电状态	立即指令
SetToolMode	设置末端复用端子的模式	立即指令

## DO

### 原型

```
DO(index,status,time)
```

### 描述

设置数字输出端口状态。

必选参数

参数名	类型	说明
index	int	DO端子的编号。取值范围：[1,MAX]或[100,1000]。MAX代表当前控制柜的DO范围，不同控制柜的DO资源数量不一样。当取值范围为[100,1000]时，需要有拓展IO模块的硬件支持。
status	int	DO端子的状态，1：打开；0：关闭。

可选参数

参数名	类型	说明
time	int	持续输出时间。取值范围：[25, 60000]，单位：ms 如果设置了该参数，系统会在指定时间后对DO自动取反。取反为异步动作，不会阻塞指令队列，系统执行了DO输出后就会执行下一条指令。

返回

```
ErrorID,{ResultID},DO(index,status,time);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

示例

```
DO(1,1,2000)
```

设置DO\_1为打开状态，2秒后自动取反（关闭）。

DOInstant

原型

```
DOInstant(index,status)
```

描述

设置数字输出端口状态。

必选参数

参数名	类型	说明
		DO端子的编号。取值范围：[1,MAX]或[100,1000]。MAX代表

index	int	当前控制柜的DO范围，不同控制柜的DO资源数量不一样。当取值范围为[100,1000]时，需要有拓展IO模块的硬件支持。
status	int	DO端子的状态，1：打开；0：关闭。

## 返回

```
ErrorID, {}, DOInstant(index, status);
```

## 示例

```
DOInstant(1, 1)
```

无视指令队列，立即设置DO\_1为打开状态。

# GetDO

## 原型

```
GetDO(index)
```

## 描述

获取数字输出端口状态。

## 必选参数

参数名	类型	说明
index	int	DO端子的编号。取值范围：[1,MAX]或[100,1000]。MAX代表当前控制柜的DO范围，不同控制柜的DO资源数量不一样。当取值范围为[100,1000]时，需要有拓展IO模块的硬件支持。

## 返回

```
ErrorID, {value}, GetDO(index);
```

value表示DO端子的状态，0为关闭，1为打开。

## 示例

```
GetDO(1)
```

获取DO\_1的开关状态。

# DOGroup

## 原型

```
DOGroup(index1,value1,index2,value2,...,indexN,valueN)
```

## 描述

设置多个数字输出端口状态，最大支持64个。

## 必选参数

参数名	类型	说明
index1	int	第一个DO端子的编号。取值范围：[1,MAX]或[100,1000]。MAX代表当前控制柜的DO范围，不同控制柜的DO资源数量不一样。当取值范围为[100,1000]时，需要有拓展IO模块的硬件支持。
value1	int	第一个DO端子的状态，1：打开；0：关闭。
...	...	...
indexN	int	第N个DO端子的编号。取值范围：[1,MAX]或[100,1000]。MAX代表当前控制柜的DO范围，不同控制柜的DO资源数量不一样。当取值范围为[100,1000]时，需要有拓展IO模块的硬件支持。
valueN	int	第N个DO端子的状态，1：打开；0：关闭。

## 返回

```
ErrorID,{ResultID},DOGroup(index1,value1,index2,value2,...,indexN,valueN);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

## 示例

```
DOGroup(4,1,6,0,2,1,7,0)
```

设置DO\_4为打开，DO\_6为关闭，DO\_2为打开，DO\_7为关闭。

# GetDOGroup

## 原型

```
GetDOGroup(index1,index2,...,indexN)
```

## 描述

获取多个数字输出端口状态。

## 必选参数

参数名	类型	说明
index	int	DO端子的编号。取值范围：[1,MAX]或[100,1000]。MAX代表当前控制柜的DO范围，不同控制柜的DO资源数量不一样。当取值范围为[100,1000]时，需要有拓展IO模块的硬件支持。

## 返回

```
ErrorID,{value1,value2,...,valueN},GetDOGroup(index1,index2,...,indexN);
```

{value1,value2,...,valueN}分别表示DO\_1到DO\_N的状态，0为关闭，1为打开。

## 示例

```
GetDOGroup(1,2)
```

获取DO\_1和DO\_2的状态。

# ToolDO

## 原型

```
ToolDO(index,status)
```

## 描述

设置末端数字输出端口状态。

## 必选参数

参数名	类型	说明
index	int	末端DO端子的编号，取值范围：[1,MAX]。MAX代表当前末端的DO范围，不同末端的DO资源数量不一样。
status	int	末端DO端子的状态，1：打开；0：关闭。

## 返回

```
ErrorID,{ResultID},ToolDO(index,status);
```



ResultID为算法队列ID，可用于判断指令执行顺序。

示例

```
ToolDO(1,1)
```

设置末端DO\_1为打开状态。

ToolDOInstant

原型

```
ToolDOInstant(index,status)
```

描述

设置末端数字输出端口状态。

必选参数

参数名	类型	说明
index	int	末端DO端子的编号，取值范围：[1,MAX]。 MAX代表当前末端的DO范围，不同末端的DO资源数量不一样。
status	int	末端DO端子的状态，1：打开；0：关闭。

返回

```
ErrorID,{},ToolDOInstant(index,status);
```

示例

```
ToolDOInstant(1,1)
```

无视指令队列，立即设置末端DO\_1为打开状态。

GetToolDO

原型

```
GetToolDO(index)
```

描述

获取末端数字输出端口状态。

必选参数

参数名	类型	说明
index	int	末端DO端子的编号，取值范围：[1,MAX]。 MAX代表当前末端的DO范围，不同末端的DO资源数量不一样。

返回

```
ErrorID,{value},GetToolDO(index);
```

value表示末端DO端子的状态，0为关闭，1为打开。

示例

```
GetToolDO(1)
```

获取末端DO\_1的状态。

AO

原型

```
AO(index,value)
```

描述

设置模拟输出端口的值。

必选参数

参数名	类型	说明
index	int	AO端子的编号，取值范围：1/2。
value	double	AO端子的输出值，电压取值范围：[0,10]，单位：V；电流取值范围：[4,20]，单位：mA。

返回

```
ErrorID,{ResultID},AO(index,value);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

示例

```
AO(1,2)
```

设置AO\_1的值为2。

## AOInstant

### 原型

```
AOInstant(index,value)
```

### 描述

设置模拟输出端口的值。

### 必选参数

参数名	类型	说明
index	int	AO端子的编号，取值范围：1/2。
value	double	AO端子的输出值，电压取值范围：[0,10]，单位：V；电流取值范围：[4,20]，单位：mA。

### 返回

```
ErrorID,{},AOInstant(index,value);
```

### 示例

```
AOInstant(1,2)
```

无视指令队列，立即设置AO\_1的值为2。

## GetAO

### 原型

```
GetAO(index)
```

### 描述

获取模拟量输出端口的值。

### 必选参数

参数名	类型	说明
index	int	AO端子的编号，取值范围：1/2。

## 返回

```
ErrorID,{value},GetAO(index);
```

value表示AO端子的值。

## 示例

```
GetAO(1)
```

获取AO\_1的值。

# DI

## 原型

```
DI(index)
```

## 描述

获取数字量输入端口的状态。

## 必选参数

参数名	类型	说明
index	int	DI端子的编号。取值范围：[1,MAX]或[100,1000]。MAX代表当前控制柜的DI范围，不同控制柜的DI资源数量不一样。当取值范围为[100,1000]时，需要有拓展IO模块的硬件支持。

## 返回

```
ErrorID,{value},DI(index);
```

value表示DI端子的状态，0为关闭，1为打开。

## 示例

```
DI(1)
```

获取DI\_1的状态。

# DIGroup

## 原型

```
DIGroup(index1,index2,...,indexN)
```

## 描述

获取多个DI端口的状态，最大支持64个。

## 必选参数

参数名	类型	说明
index	int	DI端子的编号。取值范围：[1,MAX]或[100,1000]。MAX代表当前控制柜的DI范围，不同控制柜的DI资源数量不一样。当取值范围为[100,1000]时，需要有拓展IO模块的硬件支持。

## 返回

```
ErrorID,{value1,value2,...,valueN},DIGroup(index1,index2,...,indexN);
```

{value1,value2,...,valueN}表示返回当前index1到indexN端子的状态，0为关闭，1为打开。

## 示例

```
DIGroup(4,6,2,7)
```

获取DI\_4，DI\_6，DI\_2，DI\_7端子的状态。

# ToolDI

## 原型

```
ToolDI(index)
```

## 描述

获取末端数字量输入端口的状态。

## 必选参数

参数名	类型	说明
index	int	末端DI端子的编号，取值范围：[1,MAX]。MAX代表当前控制柜的DI范围，不同控制柜的DI资源数量不一样。

## 返回

```
ErrorID,{value},ToolDI(index);
```

value表示末端DI端子的状态，0为关闭，1为打开。

## 示例

```
ToolDI(1)
```

获取末端DI\_1的状态。

# AI

## 原型

```
AI(index)
```

## 描述

获取模拟量输入端口的值。

## 必选参数

参数名	类型	说明
index	int	AI端子的编号，取值范围：1/2。

## 返回

```
ErrorID,{value},AI(index);
```

value表示AI端子的输入值。

## 示例

```
AI(1)
```

获取AI\_1的输入值。

# ToolAI

## 原型

```
ToolAI(index)
```

## 描述

获取末端模拟量输入端口的值。使用前需要通过[SetToolMode](#)将端子设置为模拟输入模式。

## 必选参数

参数名	类型	说明
index	int	末端AI端子的编号，取值范围：[1,MAX]。 MAX代表当前控制柜的AI范围，不同控制柜的AI资源数量不一样。

## 返回

```
ErrorID,{value},ToolAI(index);
```

value表示末端AI端子的输入值。

## 示例

```
ToolAI(1)
```

获取末端AI\_1的输入值。

# SetTool485

## 原型：

```
SetTool485(baud,parity,stopbit,identify)
```

## 描述：

设置末端工具的RS485接口对应的数据格式。

## 必选参数

参数名	类型	说明
baud	int	RS485接口的波特率。

## 可选参数

参数名	类型	说明
parity	string	是否有奇偶校验位。"O"表示奇校验，"E"表示偶校验，"N"表示无奇偶校验位。默认值为"N"。
stopbit	int	停止位长度。取值范围：1，2。默认值为1。

identify	int	当机械臂为多航插机型时，用于指定设置的航插。1：航插1；2：航插2
----------	-----	-----------------------------------

## 返回

```
ErrorID,{},SetTool485(baud,parity,stopbit,identify);
```

## 示例：

```
SetTool485(115200,"N",1)
```

将末端工具的RS485接口对应的波特率设置为115200Hz，无奇偶校验位，停止位长度为1。

# SetToolPower

## 原型：

```
SetToolPower(status)
```

## 描述：

设置末端工具供电状态，一般用于重启末端电源，例如对末端夹爪重新上电初始化。如需连续调用该接口，建议至少间隔4ms以上。

### 说明：

Magician E6机器人不支持该指令，调用无效果。

## 必选参数

参数名	类型	说明
status	int	末端工具供电状态，0：关闭电源；1：打开电源。

## 返回

```
ErrorID,{},SetToolPower(status);
```

## 示例：

```
SetToolPower(0)
```

关闭末端电源。



# SetToolMode

原型:

```
SetToolMode(mode,type,identify)
```

描述:

机器人末端AI接口与485接口复用端子时，可通过此接口设置末端复用端子的模式。默认模式为485模式。

 **说明:**  
不支持末端模式切换的机器人调用此接口无效果。

必选参数

参数名	类型	说明
mode	int	复用端子的模式，1：485模式，2：模拟输入模式。

可选参数

参数名	类型	说明
type	int	当mode为1时，该参数无效。 当mode为2时，可设置模拟输入的模式（见type参数含义的说明）。个位表示AI1的模式，十位表示AI2的模式。十位为0时可仅输入个位。
identify	int	当机器人为多航插机型时，用于指定设置的航插。1：航插1；2：航插2。不填默认为航插1。

- type参数含义：
  - 0: 0~10V电压输入模式
  - 1: 电流采集模式
  - 2: 0~5V电压输入模式
- 例子：
  - 0: AI1与AI2均为0~10V电压输入模式
  - 1: AI2是0~10V电压输入模式，AI1是电流采集模式
  - 11: AI2和AI1都是电流采集模式
  - 12: AI2是电流采集模式，AI1是0~5V电压输入模式
  - 20: AI2是0~5V电压输入模式，AI1是0~10V电压输入模式

返回

```
ErrorID,{}),SetToolMode(mode,type,identify);
```

### 示例:

```
SetToolMode(2,0)
```

设置末端复用端子为模拟输入，两路都是0~10V电压输入模式。

## 2.5 Modbus相关指令

### 指令列表

指令	功能	指令类型
ModbusCreate	创建Modbus主站	立即指令
ModbusRTUCreate	创建基于RS485接口的Modbus主站	立即指令
ModbusClose	和Modbus从站断开连接	立即指令
GetInBits	读取触点寄存器	立即指令
GetInRegs	读取输入寄存器	立即指令
GetCoils	读取线圈寄存器	立即指令
SetCoils	写入线圈寄存器	立即指令
GetHoldRegs	读取保持寄存器	立即指令
SetHoldRegs	写入保持寄存器	立即指令

Modbus函数用于建立Modbus主站与从站进行通讯，寄存器地址的取值范围与定义请参考对应从站的Modbus寄存器地址定义说明。

各类寄存器对应的Modbus功能码遵循标准Modbus协议：

寄存器类型	读取寄存器	写单个寄存器	写多个寄存器
线圈寄存器	01	05	0F
触点寄存器	02	-	-
输入寄存器	04	-	-
保持寄存器	03	06	10

### ModbusCreate

#### 原型

```
ModbusCreate(ip,port,slave_id,isRTU)
```

#### 描述

创建Modbus主站，并和从站建立连接。最多支持同时连接5个设备。

### 必选参数

参数名	类型	说明
ip	string	从站IP地址。
port	int	从站端口。
slave_id	int	从站ID。

### 可选参数

参数名	类型	说明
isRTU	int	如果不携带或为0，建立modbusTCP通信。如果为1，建立modbusRTU通信。

#### 注意：

此参数决定了连接建立后传输数据使用的协议格式，并不影响连接结果。因此，如果创建主站时该参数设置错误，依然可以创建成功，但后续通讯时会导致异常。

### 返回

```
ErrorID,{index},ModbusCreate(ip,port,slave_id,isRTU);
```

- ErrorID为0表示创建成功，-1表示创建失败，其余错误码请参考通用错误码
- index为返回的主站索引，后续调用其他Modbus指令时使用

### 示例

```
ModbusCreate("127.0.0.1",60000,1,0)
```

建立modbusTCP通信主站，连接本机的Modbus从站，端口为60000，从站ID为1。

## ModbusRTUCreate

### 原型：

```
ModbusRTUCreate(slave_id,baud,parity,data_bit,stop_bit)
```

### 描述：

创建基于RS485接口的Modbus主站，并和从站建立连接。最多支持同时连接5个设备。

### 必选参数

参数名	类型	说明
slave_id	int	从站ID。
baud	int	RS485接口的波特率。

#### 可选参数

参数名	类型	说明
parity	string	是否有奇偶校验位。"O"表示奇校验，"E"表示偶校验，"N"表示无奇偶校验位。默认值为"E"。
data_bit	int	数据位长度。默认值为8。
stop_bit	int	停止位长度。默认值为1。

#### 返回：

```
ErrorID,{index},ModbusRTUCreate(slave_id,baud,parity,data_bit,stop_bit);
```

- ErrorID为0表示创建成功，-1表示创建失败，其余错误码请参考通用错误码
- index为返回的主站索引，后续调用其他Modbus指令时使用

#### 示例：

```
ModbusRTUCreate(1,115200)
```

创建Modbus主站并与RS485接口连接的从站建立连接，从站ID为1，波特率为115200。

## ModbusClose

#### 原型

```
ModbusClose(index)
```

#### 描述

和Modbus从站断开连接，释放主站。

#### 必选参数

参数名	类型	说明
index	int	创建主站时返回的主站索引。

#### 返回

```
ErrorID,{},ModbusClose(index);
```

## 示例

```
ModbusClose(0)
```

释放索引为0的Modbus主站。

## GetInBits

### 原型

```
GetInBits(index,addr,count)
```

### 描述

读取Modbus从站触点寄存器（离散输入）地址的值。

### 必选参数

参数名	类型	说明
index	int	创建主站时返回的主站索引。
addr	int	触点寄存器起始地址。
count	int	连续读取触点寄存器的值的数量。取值范围：[1, 16]。

### 返回

```
ErrorID,{value1,value2,...,valuen},GetInBits(index,addr,count);
```

{value1,value2,...,valuen}为读取的值，数量与count相同。

### 示例

```
GetInBits(0,3000,5)
```

从地址为3000的触点寄存器开始读取5个值。

## GetInRegs

### 原型

```
GetInRegs(index,addr,count,valType)
```

## 描述

按照指定的数据类型，读取Modbus从站输入寄存器地址的值。

## 必选参数

参数名	类型	说明
index	int	创建主站时返回的主站索引。
addr	int	输入寄存器起始地址。
count	int	连续读取输入寄存器的值的数量。取值范围：[1, 4]。

## 可选参数

参数名	类型	说明
valType	string	读取的数据格式： U16：16位无符号整数（2个字节，占用1个寄存器）； U32：32位无符号整数（4个字节，占用2个寄存器）； F32：32位单精度浮点数（4个字节，占用2个寄存器）； F64：64位双精度浮点数（8个字节，占用4个寄存器）； 默认值为U16。

## 返回

```
ErrorID,{value1,value2,...,valuen},GetInRegs(index,addr,count,valType);
```

{value1,value2,...,valuen}为读取的值，数量与count相同。

## 示例

```
GetInRegs(0,4000,3)
```

从地址为4000的输入寄存器开始读取3个值，值类型为U16。

# GetCoils

## 原型

```
GetCoils(index,addr,count)
```

## 描述

读取Modbus从站线圈寄存器地址的值。

## 必选参数

参数名	类型	说明
index	int	创建主站时返回的主站索引。
addr	int	线圈寄存器起始地址。
count	int	连续读取线圈寄存器的值的数量。取值范围：[1, 16]。

## 返回

```
ErrorID,{value1,value2,...,valuen},GetCoils(index,addr,count);
```

{value1,value2,...,valuen}为读取的值，数量与count相同。

## 示例

```
GetCoils(0,1000,3)
```

从地址为1000的线圈寄存器开始读取3个值。

# SetCoils

## 原型

```
SetCoils(index,addr,count,valTab)
```

## 描述

将指定的值写入线圈寄存器指定的地址。

## 必选参数

参数名	类型	说明
index	int	创建主站时返回的主站索引。
addr	int	线圈寄存器起始地址。
count	int	连续写入线圈寄存器的值的数量。取值范围：[1, 16]。
valTab	string	要写入的值，数量与count相同。

## 返回

```
ErrorID,{},SetCoils(index,addr,count,valTab);
```

## 示例



```
SetCoils(0,1000,3,{1,0,1})
```

从地址为1000的线圈寄存器开始连续写入3个值，分别为1，0，1。

## GetHoldRegs

### 原型

```
GetHoldRegs(index,addr,count,valType)
```

### 描述

按照指定的数据类型，读取Modbus从站保持寄存器地址的值。

### 必选参数

参数名	类型	说明
index	int	创建主站时返回的主站索引，最多支持5个设备。取值范围：[0,4]。
addr	int	保持寄存器起始地址。
count	int	连续读取保持寄存器的值的数量。

### 可选参数

参数名	类型	说明
valType	string	读取的数据类型： U16：16位无符号整数（2个字节，占用1个寄存器）； U32：32位无符号整数（4个字节，占用2个寄存器）； F32：32位单精度浮点数（4个字节，占用2个寄存器）； F64：64位双精度浮点数（8个字节，占用4个寄存器）； 默认值为U16。

### 返回

```
ErrorID,{value1,value2,...,valuen},GetHoldRegs(index,addr,count,valType);
```

{value1,value2,...,valuen}为读取的值，数量与count相同。

### 示例

```
GetHoldRegs(0,3095,1)
```

从地址为3095的保持寄存器开始读取1个值，值类型为U16。

# SetHoldRegs

## 原型

```
SetHoldRegs(index, addr, count, valTab, valType)
```

## 描述

将指定的值以指定的数据类型写入Modbus从站保持寄存器指定的地址。

## 必选参数

参数名	类型	说明
index	int	创建主站时返回的主站索引，最多支持5个设备。取值范围：[0,4]。
addr	int	保持寄存器起始地址。
count	int	连续写入保持寄存器的值的数量。取值范围：[1, 4]
valTab	string	要写入的值，数量与count相同。

## 可选参数

参数名	类型	说明
valType	string	写入的数据类型： U16：16位无符号整数（2个字节，占用1个寄存器）； U32：32位无符号整数（4个字节，占用2个寄存器）； F32：32位单精度浮点数（4个字节，占用2个寄存器）； F64：64位双精度浮点数（8个字节，占用4个寄存器）； 默认值为U16。

## 返回

```
ErrorID, {}, SetHoldRegs(index, addr, count, valTab, valType);
```

## 示例

```
SetHoldRegs(0, 3095, 2, {6000, 300}, U16)
```

从地址为3095的保持寄存器开始写入两个U16类型的值，分别为6000和300。

## 2.6 总线寄存器相关指令

### 指令列表

总线寄存器指令用于读写Profinet或Ethernet/IP总线寄存器。

指令	功能	指令类型
GetInputBool	获取输入寄存器指定地址的bool值	立即指令
GetInputInt	获取输入寄存器指定地址的int值	立即指令
GetInputFloat	获取输入寄存器指定地址的float值	立即指令
GetOutputBool	获取输出寄存器指定地址的bool值	立即指令
GetOutputInt	获取输出寄存器指定地址的int值	立即指令
GetOutputFloat	获取输出寄存器指定地址的float值	立即指令
SetOutputBool	设置输出寄存器指定地址的bool值	立即指令
SetOutputInt	设置输出寄存器指定地址的int值	立即指令
SetOutputFloat	设置输出寄存器指定地址的float值	立即指令

### GetInputBool

原型：

```
GetInputBool(address)
```

描述：

获取输入寄存器指定地址的bool类型的数值。

必选参数

参数名	类型	说明
address	int	寄存器地址，取值范围：[0,63]。

返回

```
ErrorID,{value},GetInputBool(address);
```

value表示指定的寄存器地址的值，为0或1。

**示例：**

```
GetInputBool(0)
```

读取输入寄存器地址位0的布尔值。

## GetInputInt

**原型：**

```
GetInputInt(address)
```

**描述：**

获取输入寄存器指定地址的int类型的数值。

**必选参数**

参数名	类型	说明
address	int	寄存器地址，取值范围：[0,23]。

**返回**

```
ErrorID,{value},GetInputInt(address);
```

value表示指定的寄存器地址的值，为整型数（int32）。

**示例：**

```
GetInputInt(1)
```

读取输入寄存器地址位1的int值。

## GetInputFloat

**原型：**

```
GetInputFloat(address)
```

**描述：**

获取输入寄存器指定地址的float类型的数值。

**必选参数**

参数名	类型	说明
address	int	寄存器地址，取值范围：[0,23]。

## 返回

```
ErrorID,{value},GetInputFloat(address);
```

value表示指定的寄存器地址的值，为单精度浮点数（float）

## 示例：

```
GetInputFloat(2)
```

读取输入寄存器地址位2的float值。

# GetOutputBool

## 原型：

```
GetOutputBool(address)
```

## 描述：

获取输出寄存器指定地址的bool类型的数值。

## 必选参数

参数名	类型	说明
address	int	寄存器地址，取值范围：[0,63]。

## 返回

```
ErrorID,{value},GetOutputBool(address);
```

value表示指定的寄存器地址的值，为0或1。

## 示例：

```
GetOutputBool(0)
```

获取输出寄存器地址位0的布尔值。

# GetOutputInt

### 原型:

```
GetOutputInt(address)
```

### 描述:

获取输出寄存器指定地址的int类型的数值。

### 必选参数

参数名	类型	说明
address	int	寄存器地址，取值范围：[0,23]。

### 返回

```
ErrorID,{value},GetOutputInt(address);
```

value表示指定的寄存器地址的值，为整型数（int32）。

### 示例:

```
GetOutputInt(1)
```

读取输出寄存器地址位1的int值。

## GetOutputFloat

### 原型:

```
GetOutputFloat(address)
```

### 描述:

获取输出寄存器指定地址的float类型的数值。

### 必选参数

参数名	类型	说明
address	int	寄存器地址，取值范围：[0,23]。

### 返回

```
ErrorID,{value},GetOutputFloat(address);
```

value表示指定的寄存器地址的值，为单精度浮点数（float）。

**示例：**

```
GetOutputFloat(2)
```

读取输出寄存器地址位2的float值。

## SetOutputBool

**原型：**

```
SetOutputBool(address,value)
```

**描述：**

设置输出寄存器指定地址的bool类型的数值。

**必选参数**

参数名	类型	说明
address	int	寄存器地址，取值范围：[0,63]。
value	int	要设置的值，支持0或1。

**返回**

```
ErrorID,{},SetOutputBool(address, value);
```

**示例：**

```
SetOutputBool(0,0)
```

设置输出寄存器0的值为假。

## SetOutputInt

**原型：**

```
SetOutputInt(address,value)
```

**描述：**

设置输出寄存器指定地址的int类型的数值。

### 必选参数

参数名	类型	说明
address	int	寄存器地址，取值范围：[0,23]。
value	int	要设置的值，支持带符号的32位整型数。

### 返回

```
ErrorID,{},SetOutputInt(address,value);
```

### 示例：

```
SetOutputInt(1,123)
```

设置输出寄存器地址位1的值为123。

## SetOutputFloat

### 原型：

```
SetOutputFloat(address,value)
```

### 描述：

设置输出寄存器指定地址的float类型的数值。

### 必选参数

参数名	类型	说明
address	int	寄存器地址，取值范围：[0,23]。
value	float	要设置的值，支持单精度浮点数。

### 返回

```
ErrorID,{},SetOutputFloat(address,value);
```

### 示例：

```
SetOutputFloat(2,12.3)
```

设置输出寄存器地址位2的float值为12.3。



## 2.7 运动相关指令

### 参数格式

运动指令中**点位参数**和**可选参数**均为string类型，格式为“key=value”，例如“joint = {10, 10, 10, 0, 0, 0}”，“user=1”。为方便用户理解参数，下文参数表中此类参数的类型列均表示value的类型。

### 运动方式

机器人支持的运动方式可分为下述几类。

#### 关节运动

机器人根据当前各关节角度和目标点各关节角度的差值规划各个关节的运动，使各个关节同时完成运动。关节运动不约束TCP（Tool Center Point）的运动轨迹，一般情况下该轨迹非直线。



关节运动不受奇异位置限制（奇异点位置详见机器人对应的硬件手册），因此如果对运动轨迹没有要求，或目标点位在奇异位置附近，建议使用关节运动。

#### 直线运动

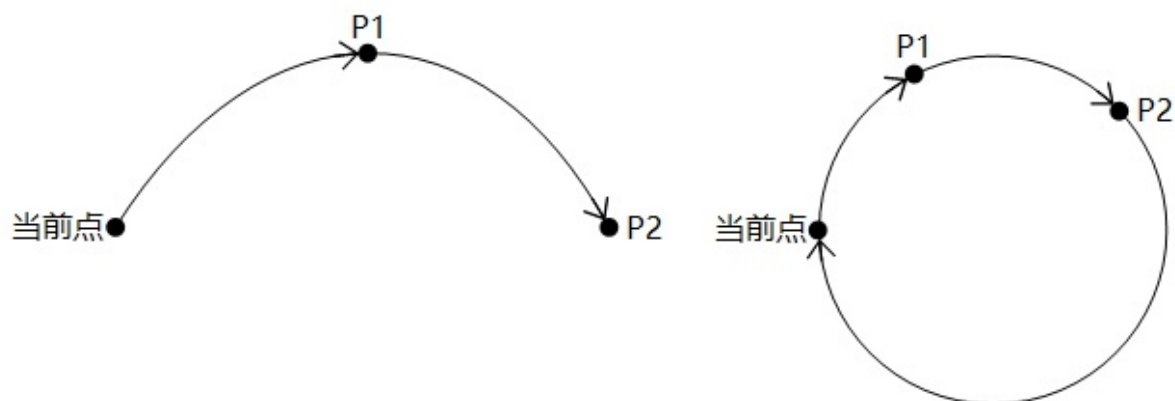
机器人根据当前位姿和目标点的位姿规划运动轨迹，使TCP运动轨迹为直线，且末端姿态在运动过程中匀速变化。



当运动轨迹会经过奇异位置时，下发直线运动指令给机器人会产生报错，建议重新规划点位或在奇异位置附近采用关节运动。

#### 弧线运动

机器人通过当前位置，P1，P2三个不共线的点确定一个圆弧或整圆。运动过程中的机器人末端姿态通过当前点和P2点的姿态插补算出，P1点的姿态不参与运算（即运动过程中机器人到达P1点时的姿态可能与示教姿态不同）。



当运动轨迹会经过奇异位置时，下发弧线运动指令给机器人会产生报错，建议重新规划点位或在奇异位置附近采用关节运动。

## 点位参数

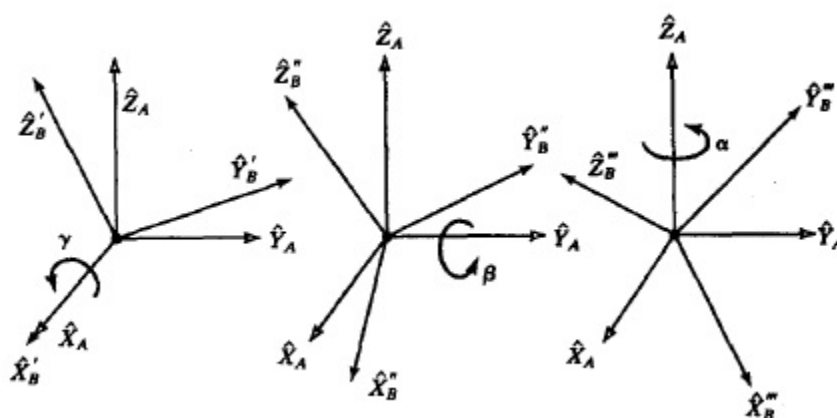
如无特殊说明，运动指令中所有点位参数（P）都支持两种表达方式：

- 关节变量：使用各个机器人各个关节的角度（j1~j6）表示目标点位。作为目标点时会通过正解变换为位姿变量再使用。

```
joint = {j1, j2, j3, j4, j5, j6}
```

- 位姿变量：使用笛卡尔坐标（x, y, z）表示目标点位在用户坐标系中的空间位置，使用欧拉角（rx, ry, rz）表示TCP（Tool Center Point）到达该点时工具坐标系相对于用户坐标系的旋转角度。

越疆机器人计算欧拉角时的旋转顺序为X->Y->Z，每个轴都是绕固定轴（用户坐标系）旋转，如下图所示（rx=γ, ry=β, rz=α）。



确定了旋转顺序后，就可以将旋转矩阵（其中cα为cosα，sα为sinα的简写，以此类推）

$${}^A_B R_{XYZ}(\gamma, \beta, \alpha) = R_Z(\alpha) R_Y(\beta) R_X(\gamma)$$

$$= \begin{bmatrix} c\alpha & -s\alpha & 0 \\ s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\beta & 0 & s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\gamma & -s\gamma \\ 0 & s\gamma & c\gamma \end{bmatrix}$$

推导为方程

$${}^A_B R_{XYZ}(\gamma, \beta, \alpha) = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix}$$

通过该方程计算机器人末端的姿态。

```
pose = {x, y, z, rx, ry, rz}
```

## 坐标系参数

笛卡尔坐标系相关的运动指令，可选参数的user和tool用于指定目标点的用户和工具坐标系：

当前仅支持通过索引序号指定，需要先在控制软件中添加对应坐标系。

如果不携带user和tool参数，则使用全局用户和工具坐标系，详见[设置相关指令](#)中的user和tool指令说明（未调用指令设置时的默认坐标系均为0）。

## 运动参数

### 相对速率

可选参数中的a和v用于指定机器人执行该运动指令时的加速度和速度比例。

```
机器人实际运动速度 = 最大速度 x 全局速率 x 指令速率
机器人实际运动加速度 = 最大加速度 x 指令速率
```

其中最大速度/加速度受[再现参数](#)的控制，可在DobotStudio Pro的运动参数页面查看与修改。



全局速率可通过DobotStudio Pro（上图右上角）或SpeedFactor指令设置。

指令速率为运动指令可选参数携带的比例，未通过可选参数指定运动加速度/速度比例时，默认使用运动参数中设置的值（详见VelJ, AccJ, VelL, AccL指令，未调用指令设置时的默认值均为100）。

例：

```
AccJ(50) -- 设置关节运动默认加速度为50%
VelJ(60) -- 设置关节运动默认速度为60%
AccL(70) -- 设置直线运动默认加速度为70%
VelL(80) -- 设置直线运动默认速度为80%

-- 全局速率为20%；

MovJ(P1) -- 以（关节加速度最大值 x 50%）的加速度和（关节速度最大值 x 20% x 60%）的速度关节运动至P1
MovJ(P2,{a = 30, v = 80}) -- 以（关节加速度最大值 x 30%）的加速度和（关节速度最大值 x 20% x 80%）关节运动至P1

MovL(P1) -- 以（笛卡尔加速度最大值 x 70%）的加速度和直线速度（笛卡尔速度最大值 x 20% x 80%）的速度运动至P1
MovL(P1,{a = 40, v = 90}) -- 以（笛卡尔加速度最大值 x 40%）的加速度和（笛卡尔速度最大值 x 20% x 90%）的速度直线运动至P1
```

## 绝对速度

直线和弧线运动指令可选参数中的speed用于指定机械臂执行该运动指令时的绝对速度。

绝对速度不受全局速率影响，但受再现参数中的最大速度限制（如果机器人进入了缩减模式，则受缩减后的最大速度限制），即speed参数设置的目标速度如果大于再现参数中的最大速度，则以最大速度为准。

例：

```
MovL(P1,{speed = 1000}) -- 以1000的绝对速率直线移动至P1
```

MovL设置了speed为1000，小于再现参数中的最大速度2000，则机器人会以1000mm/s为目标速度进行运动，该目标速度与此时的全局速率无关。但如果机器人处于缩减模式（假设缩减率为10%），则最大速度变为200，小于1000，此时机器人会以200mm/s为目标速度进行运动。

speed参数和v参数互斥，若同时存在以speed为准。

## 平滑过渡参数

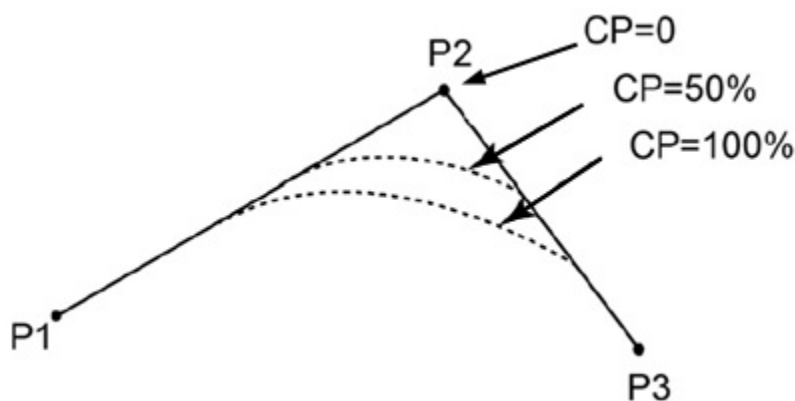
机器人连续运动经过多个点时，可以通过平滑过渡的方式经过中间点，避免机器人拐弯过于生硬。如果用户指定的几个路径点基于不同的工具坐标系，则无法平滑过渡。

可选参数中的cp或r用于指定当前运动指令到下一条运动指令之间的平滑过渡比例（cp）或者平滑过渡半径（r），两者互斥，若同时存在以r为准。

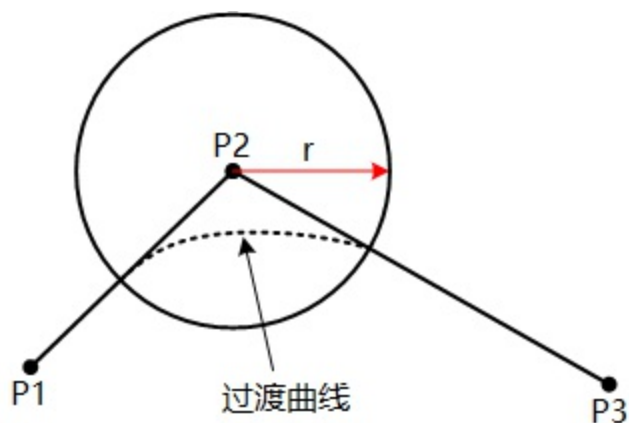
### 说明：

关节运动相关命令不支持设置平滑过渡半径（r），详见各指令的可选参数。

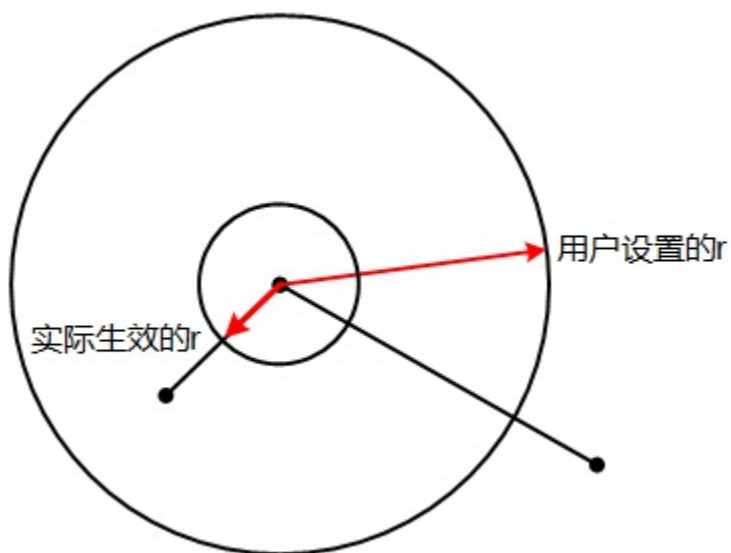
设置平滑过渡比例时，系统会自动计算过渡曲线的弧度，CP值越大曲线越平滑，如下图所示。CP过渡曲线会受运动速度/加速度影响，即使点位和CP值都相同，运动速度/加速度不同时的过渡曲线弧度也会不同。



设置平滑过渡半径时，系统会以过渡点为圆心，根据指定半径计算过渡曲线。R过渡曲线不受运动速度/加速度影响，只由点位和过渡半径决定。



如果用户设置的过渡半径过大（超过起始点/终点与过渡点之间的距离），则系统会自动使用起始点/终点与过渡点之间较短距离的一半作为过渡半径计算过渡曲线。



未通过可选参数指定平缓过渡比例或半径时，默认使用运动参数中设置的平滑过渡比例（详见CP指令，未调用指令设置时的默认值为0）。

#### **i 说明：**

平滑过渡会导致机器人运动不经过中间点，因此设置了平滑过渡时，两条运动指令之间IO信号输出或功能设置（例如开关安全皮肤）指令会在过渡过程中执行。

如果希望能够在机器人准确抵达中间点时执行指令，请将前一条指令的平滑过渡参数设置为0。

## 指令列表

指令	功能	指令类型
MovJ	关节运动	队列指令
MovL	直线运动	队列指令
MovLIO	直线运动并输出DO	队列指令
MovJIO	关节运动并输出DO	队列指令
Arc	圆弧插补运动	队列指令
Circle	整圆插补运动	队列指令
ServoJ	基于关节空间的动态跟随命令	队列指令
ServoP	基于笛卡尔空间的动态跟随命令	队列指令
MoveJog	点动机械臂	立即指令
RunTo	运动至指定点位	立即指令
GetStartPose	获取指定轨迹的第一个点位	立即指令
StartPath	复现录制的运动轨迹	队列指令
RelMovJTool	沿工具坐标系进行相对关节运动	队列指令
RelMovLTool	沿工具坐标系进行相对直线运动	队列指令
RelMovJUser	沿用户坐标系进行相对关节运动	队列指令
RelMovLUser	沿用户坐标系进行相对直线运动	队列指令
RelJointMovJ	沿关节坐标系进行相对关节运动	队列指令
RelPointTool	沿工具坐标系笛卡尔点偏移	立即指令
RelPointUser	沿用户坐标系笛卡尔点偏移	立即指令
RelJoint	关节点位偏移	立即指令
GetCurrentCommandID	获取当前执行指令的算法队列ID	立即指令

## MovJ

### 原型

```
MovJ(P, user, tool, a, v, cp)
```

### 描述

从当前位置以关节运动方式运动至目标点。

### 必选参数

参数名	类型	说明
p	string	目标点，支持关节变量或位姿变量。 格式为"joint = {j1, j2, j3, j4, j5, j6}"或"pose = {x, y, z, rx, ry, rz}"。

### 可选参数

参数名	类型	说明
user	string	格式为"user=index"，index为已标定的用户坐标系索引。取值范围：[0,50]。
tool	string	格式为"tool=index"，index为已标定的工具坐标系索引。取值范围：[0,50]。
a	string	格式为"a=value"。value表示执行该条指令时的机器人运动加速度比例。取值范围：[1,100]。
v	string	格式为"v=value"。value表示执行该条指令时的机器人运动速度比例。取值范围：[1,100]。
cp	string	格式为"cp=value"。value表示平滑过渡比例。取值范围：[0,100]。

### 返回

```
ErrorID,{ResultID},MovJ(P,user,tool,a,v,cp);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

### 示例

```
MovJ(pose={-500,100,200,150,0,90},user=1, tool=0, a=20, v=50, cp=100)
```

机器人从当前位置以50%速度，20%加速度，100%平滑过渡比例通过关节运动方式运动至笛卡尔坐标点{-500,100,200,150,0,90}（用户坐标系1，工具坐标系0）。

## MovL

### 原型

```
MovL(P,user,tool,a,v|speed,cp|r)
```



## 描述

从当前位置以直线运动方式运动至目标点。

## 必选参数

参数名	类型	说明
P	string	目标点，支持关节变量或位姿变量。 格式为"joint = {j1, j2, j3, j4, j5, j6}"或"pose = {x, y, z, rx, ry, rz}"。

## 可选参数

参数名	类型	说明
user	string	格式为"user=index"，index为已标定的用户坐标系索引。取值范围：[0,50]。
tool	string	格式为"tool=index"，index为已标定的工具坐标系索引。取值范围：[0,50]。
a	string	格式为"a=value"。value表示执行该条指令时的机器人运动加速度比例。取值范围：[1,100]。
v	string	格式为"v=value"。value表示执行该条指令时的机器人运动速度比例，与speed互斥。取值范围：[1,100]。
speed	string	格式为"speed=value"。value表示执行该条指令时的机器人运动目标速度，与v互斥，若同时存在以speed为准。取值范围：[1, 最大运动速度]，单位：mm/s。
cp	string	格式为"cp=value"。value表示平滑过渡比例，与r互斥。取值范围：[0,100]。
r	string	格式为"r=value"。value表示平滑过渡半径，与cp互斥，若同时存在以r为准。单位：mm。

## 返回

```
ErrorID,{ResultID},MovL(P,user,tool,a,v|speed,cp|r);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

## 示例

```
MovL(pose={-500,100,200,150,0,90},v=60)
```

机器人从当前位置以60%的速度通过直线运动方式运动至笛卡尔坐标点{-500,100,200,150,0,90}。

# MovLIO

## 原型

```
MovLIO(P,{Mode,Distance,Index,Status},...,{Mode,Distance,Index,Status},user,tool,a,v|speed,cp|  
r)
```

## 描述

从当前位置以直线运动方式运动至目标点，运动时并行设置数字输出端口状态。

## 必选参数

参数名	类型	说明
P	string	目标点，支持关节变量或位姿变量。 格式为"joint = {j1, j2, j3, j4, j5, j6}"或"pose = {x, y, z, rx, ry, rz}"。

{Mode,Distance,Index,Status}为并行数字输出参数，用于设置当机器人运动到指定距离或百分比时，触发指定DO。可设置多组(至少需设置一组)，参数具体含义如下：

参数名	类型	说明
Mode	int	触发模式。0表示距离百分比，1表示距离数值。
Distance	int	指定距离。 Distance为正数时，表示离起点的距离； Distance为负数时，表示离目标点的距离； Mode为0时，Distance表示起始点与目标点之间距离的百分比；取值范围：[0,100]； Mode为1时，Distance表示距离的值。单位：mm。
Index	int	DO端子的编号。取值范围：[1,24]或[100,1000]。当取值范围为[100,1000]时，需要有拓展IO模块的硬件支持。不同机型，取值范围有所差异。
Status	int	要设置的DO状态，1：打开；0：关闭。

## 可选参数

参数名	类型	说明
user	string	格式为"user=index"，index为已标定的用户坐标系索引。取值范围：[0,50]。
tool	string	格式为"tool=index"，index为已标定的工具坐标系索引。取值范围：[0,50]。
a	string	格式为"a=value"。value表示执行该条指令时的机器人运动加速度比例。取值范围：[1,100]。

v	string	格式为“v=value”。value表示执行该条指令时的机器人运动速度比例，与speed互斥。取值范围：[1,100]。
speed	string	格式为“speed=value”。value表示执行该条指令时的机器人运动目标速度，与v互斥，若同时存在以speed为准。取值范围：[1, 最大运动速度]，单位：mm/s。
cp	string	格式为“cp=value”。value表示平滑过渡比例，与r互斥。取值范围：[0,100]。
r	string	格式为“r=value”。value表示平滑过渡半径，与cp互斥，若同时存在以r为准。单位：mm。

## 返回

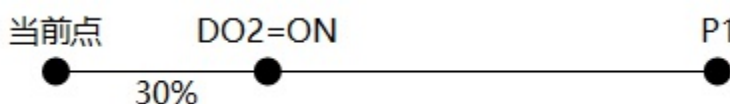
```
ErrorID,{ResultID},MovLIO(P,{Mode,Distance,Index,Status},...,{Mode,Distance,Index,Status},user,tool,a,v|speed,cp|r);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

## 示例1

```
MovLIO(pose={-500,100,200,150,0,90},{0, 30, 2, 1})
```

机器人从当前位置通过直线运动方式运动至笛卡尔坐标点{-500,100,200,150,0,90}，当运动到距离起点30%的位置时，将DO2设置为打开。



## 示例2

```
MovLIO(pose={-500,100,200,150,0,90},{1, -15, 3, 0})
```

机器人从当前位置通过直线运动方式运动至笛卡尔坐标点{-500,100,200,150,0,90}，当运动到距离终点15mm的位置时，将DO3设置为关闭。



# MovJIO

## 原型

```
MovJIO(P,{Mode,Distance,Index,Status},...,{Mode,Distance,Index,Status},user,tool,a,v,cp)
```

## 描述

从当前位置以关节运动方式运动至目标点，运动时并行设置数字输出端口状态。

## 必选参数

参数名	类型	说明
P	string	目标点，支持关节变量或位姿变量。 格式为"joint = {j1, j2, j3, j4, j5, j6}"或"pose = {x, y, z, rx, ry, rz}"。

{Mode,Distance,Index,Status}为并行数字输出参数，用于设置当机器人运动到指定距离或百分比时，触发指定DO。可设置多组，参数具体含义如下：

参数名	类型	说明
Mode	int	触发模式。0表示距离百分比，1表示距离数值。系统会将各关节角合成一个角度向量，并计算终点和起点的角度差作为运动的总距离。
Distance	int	指定距离。 Distance为正数时，表示离起点的距离； Distance为负数时，表示离目标点的距离； Mode为0时，Distance表示起始点与目标点之间距离的百分比；取值范围：[0,100]； Mode为1时，Distance表示距离的角度。单位：°
Index	int	DO端子的编号。取值范围：[1,24]或[100,1000]。当取值范围为[100,1000]时，需要有拓展IO模块的硬件支持。不同机型，取值范围有所差异。
Status	int	要设置的DO状态，1：打开；0：关闭。

## 可选参数

参数名	类型	说明
user	string	格式为"user=index"，index为已标定的用户坐标系索引。取值范围：[0,50]。
tool	string	格式为"tool=index"，index为已标定的工具坐标系索引。取值范围：[0,50]。
a	int	格式为"a=value"。value表示执行该条指令时的机器人运动加速度比例。取值范围：[1,100]。
v	int	格式为"v=value"。value表示执行该条指令时的机器人运动速度比例。取值范围：[1,100]。
		格式为"cp=value"。value表示平滑过渡比例。取值范围：

cp	int	[0,100]。
----	-----	----------

## 返回

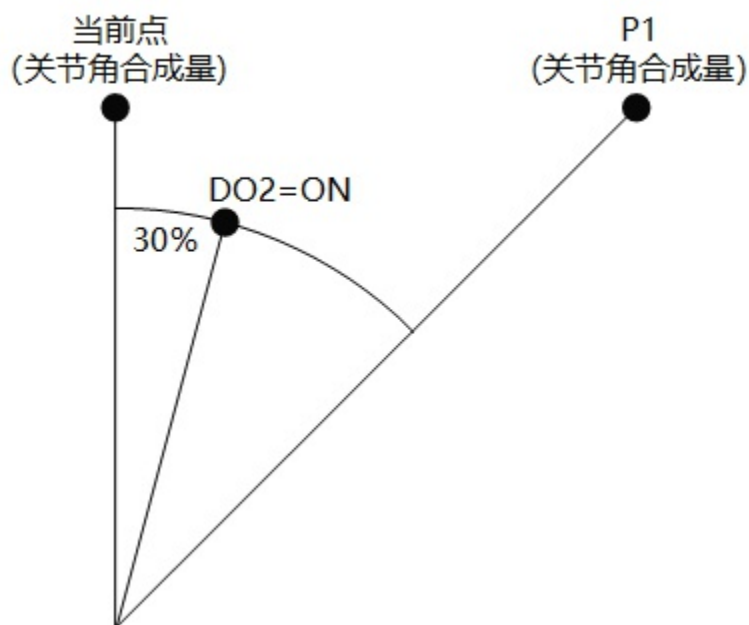
```
ErrorID,{ResultID},MovJIO(P,{Mode,Distance,Index,Status},...,{Mode,Distance,Index,Status},user,tool,a,v,cp);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

## 示例1

```
MovJIO(pose={-500,100,200,150,0,90},{0, 30, 2, 1})
```

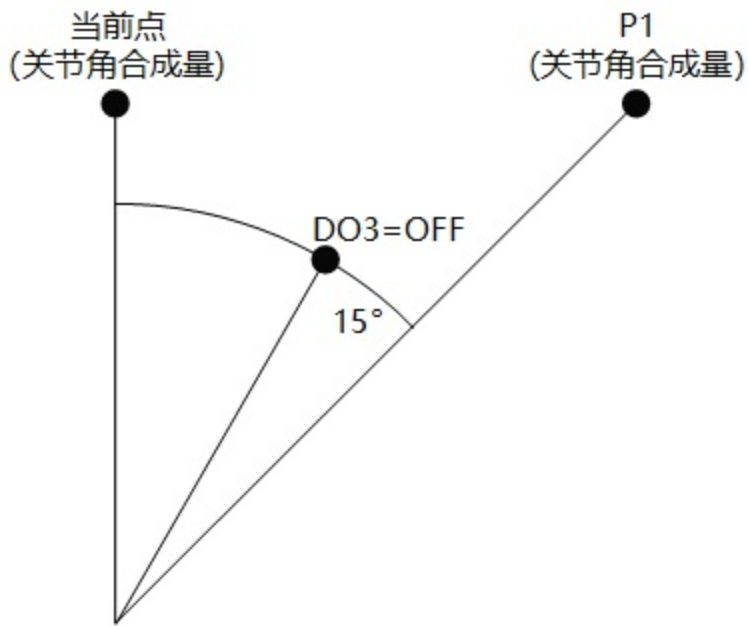
机器人从当前位置通过关节运动方式运动至笛卡尔坐标点{-500,100,200,150,0,90}，当运动到距离起点30%的位置时，将DO2设置为打开。



## 示例2

```
MovJIO(pose={-500,100,200,150,0,90},{1, -15, 3, 0})
```

机器人从当前位置通过关节运动方式运动至笛卡尔坐标点{-500,100,200,150,0,90}，当运动到距离终点还有15°的位置时，将DO3设置为关闭。



## Arc

### 原型

```
Arc(P1,P2,user,tool,a,v|speed,cp|r,ori_mode)
```

### 描述

从当前位置以圆弧插补方式运动至目标点。

需要通过当前位置，圆弧中间点，运动目标点三个点确定一个圆弧，因此当前位置不能在P1和P2确定的直线上。

### 必选参数

参数名	类型	说明
P1	string	圆弧中间点，支持关节变量或位姿变量。 格式为"joint = {j1, j2, j3, j4, j5, j6}"或"pose = {x, y, z, rx, ry, rz}"。
P2	string	运动目标点，支持关节变量或位姿变量。 格式为"joint = {j1, j2, j3, j4, j5, j6}"或"pose = {x, y, z, rx, ry, rz}"。

### 可选参数

参数名	类型	说明
user	string	格式为"user=index", index为已标定的用户坐标系索引。取值范围：[0,50]。
tool	string	格式为"tool=index", index为已标定的工具坐标系索引。取值范围：[0,50]。

a	string	格式为“a=value”。value表示执行该条指令时的机械臂运动加速度比例。取值范围：[1,100]。
v	string	格式为“v=value”。value表示执行该条指令时的机械臂运动速度比例。取值范围：[1,100]。
speed	string	格式为“speed=value”。value表示执行该条指令时的机械臂运动目标速度，与v互斥，若同时存在以speed为准。取值范围：[1, 最大运动速度]，单位：mm/s。
cp	string	格式为“cp=value”。value表示平滑过渡比例，与r互斥。取值范围：[0,100]。
r	string	格式为“r=value”。value表示平滑过渡半径，与cp互斥，若同时存在以r为准。单位：mm。
ori_mode	int	0表示起始姿态按Slerp插值，目标点姿态可达； 1表示起始姿态按圆弧Z轴旋转，目标点姿态不可达。

## 返回

```
ErrorID,{ResultID},Arc(P1,P2,user,tool,a,v|speed,cp|r,ori_mode);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

## 示例

```
Arc(pose={-350,-200,200,150,0,90},pose={-300,-250,200,150,0,90})
```

机器人从当前位置通过圆弧运动方式经由笛卡尔坐标点{-350,-200,200,150,0,90}运动至笛卡尔坐标点{-300,-250,200,150,0,90}。

# Circle

## 原型

```
Circle(P1,P2,count,user,tool,a,v|speed,cp|r)
```

## 描述

从当前位置进行整圆插补运动，运动指定圈数后重新回到当前位置。

需要通过当前位置，P1，P2三个点确定一个整圆，因此当前位置不能在P1和P2确定的直线上，且三个点确定的整圆不能超出机器人的运动范围。

## 必选参数

参数名	类型	说明
-----	----	----

P1	string	整圆中间点，支持关节变量或位姿变量。 格式为"joint = {j1, j2, j3, j4, j5, j6}"或"pose = {x, y, z, rx, ry, rz}"。
P2	string	整圆结束点点，支持关节变量或位姿变量。 格式为"joint = {j1, j2, j3, j4, j5, j6}"或"pose = {x, y, z, rx, ry, rz}"。
count	int	进行整圆运动的圈数，取值范围：[1,999]。

### 可选参数

参数名	类型	说明
user	string	格式为"user=index"，index为已标定的用户坐标系索引。取值范围：[0,50]。
tool	string	格式为"tool=index"，index为已标定的工具坐标系索引。取值范围：[0,50]。
a	string	格式为"a=value"。value表示执行该条指令时的机器人运动加速度比例。取值范围：[1,100]。
v	string	格式为"v=value"。value表示执行该条指令时的机器人运动速度比例，与speed互斥。取值范围：[1,100]。
speed	string	格式为"speed=value"。value表示执行该条指令时的机器人运动目标速度，与v互斥，若同时存在以speed为准。取值范围：[1, 最大运动速度]，单位：mm/s。
cp	string	格式为"cp=value"。value表示平滑过渡比例，与r互斥。取值范围：[0,100]。
r	string	格式为"r=value"。value表示平滑过渡半径，与cp互斥，若同时存在以r为准。单位：mm。

### 返回

```
ErrorID,{ResultID},Circle(P1,P2,count,user,tool,a,v|speed,cp|r);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

### 示例

```
Circle(pose={-350,-200,200,150,0,90},pose={-300,-250,200,150,0,90},1)
```

机器人从当前位置经由笛卡尔坐标点{-350,-200,200,150,0,90}和{-300,-250,200,150,0,90}整圆运动一圈。

## ServoJ



## 原型

```
ServoJ(J1,J2,J3,J4,J5,J6,t,aheadtime,gain)
```

## 描述

基于关节空间的动态跟随命令，一般用于在线控制的寸动功能，通过循环调用实现动态跟随。调用频率建议设置为33Hz，即循环调用的间隔时间为30ms。



### 注意：

- 该指令不受全局速率影响，但受速度限制约束。
- t值设置过小时，机器人执行指令时会因为速度限制无法满足指定的t。
- 调用该指令前建议对运行点位进行速度规划，按照固定时间间隔t下发速度规划后的点位，保证机器人能平稳跟踪目标点位。

## 必选参数

参数名	类型	说明
J1,J2,J3,J4,J5,J6	double	点J1,J2,J3,J4,J5,J6轴位置，单位：度。

## 可选参数

参数名	类型	说明
t	float	格式为“t=value”。value表示该点位的运行时间，单位：s，取值范围:[0.004,3600.0]，默认值0.1。
aheadtime	float	格式为“aheadtime=value”。value表示提前量，作用类似于PID控制中的D项。标量，无单位，取值范围：[20.0,100.0]，默认值50。
gain	float	格式为“gain=value”。value表示目标位置的比例增益，作用类似于PID控制中的P项。标量，无单位，取值范围：[200.0,1000.0]，默认值500。

aheadtime和gain参数共同决定机器人运动的响应时间和轨迹平滑度，较小的aheadtime值或较大的gain值能使机器人快速响应，但可能造成不稳定和抖动。

## 返回

```
ErrorID,{ResultID},ServoJ(J1,J2,J3,J4,J5,J6,t,aheadtime,gain);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

## 示例

```
ServoJ(0,0,-90,0,90,0,t=0.1,aheadtime=50,gain=500)
// 间隔30ms循环调用，每次第三个参数加1
ServoJ(0,0,-89,0,90,0,t=0.1,aheadtime=50,gain=500)
```

J3轴进行步伐为1度的寸动。

## ServoP

### 原型

```
ServoP(X,Y,Z,Rx,Ry,Rz,t,aheadtime,gain)
```

### 描述

基于笛卡尔空间的动态跟随命令，一般用于在线控制的寸动功能，通过循环调用实现动态跟随。调用频率建议设置为33Hz，即循环调用的间隔时间为30ms。



#### 注意：

- 该指令不受全局速率影响，但受速度限制约束。
- t值设置过小时，机器人执行指令时会因为速度限制无法满足指定的t。
- 调用该指令前建议对运行点位进行速度规划，按照固定时间间隔t下发速度规划后的点位，保证机器人能平稳跟踪目标点位。

</div>

### 必选参数

参数名	类型	说明
X,Y,Z,Rx,Ry,Rz	double	目标点位位姿变量。X,Y,Z单位：毫米，Rx,Ry,Rz单位：度。参考坐标系为全局用户和工具坐标系，详见 <a href="#">设置相关指令</a> 中的User和Tool指令说明（默认值均为0）。

### 可选参数

参数名	类型	说明
t	float	格式为“t=value”。value表示该点位的运行时间，单位：s，取值范围：[0.004,3600.0]，默认值0.1。
aheadtime	float	格式为“aheadtime=value”。value表示提前量，作用类似于PID控制中的D项。标量，无单位，取值范围：[20.0,100.0]，默认值50。
gain	float	格式为“gain=value”。value表示目标位置的比例增益，作用类似于PID控制中的P项。标量，无单位，取值范围：[200.0,1000.0]，默认值500。

aheadtime和gain参数共同决定机器人运动的响应时间和轨迹平滑度，较小的aheadtime值或较大的gain值能使机器人快速响应，但可能造成不稳定和抖动。

### 返回

```
ErrorID,{ResultID},ServoP(X,Y,Z,Rx,Ry,Rz,t,aheadtime,gain);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

### 示例

```
ServoP(-500,100,200,150,0,90)  
// 间隔30ms循环调用，每次第一个参数加1  
ServoP(-499,100,200,150,0,90)
```

沿X轴进行步伐为1mm的寸动。

## MoveJog

### 原型

```
MoveJog(axisID,coordtype,user,tool)
```

### 描述

点动或停止点动机器人。下发命令后机器人会沿指定轴持续点动，需要再下发MoveJog()停止机器人运动。另外，机器人点动时下发携带任意非指定string的MoveJog(string)也会使机器人停止运动。

该指令为立即指令，支持在工程暂停时调用。

### 必选参数

参数名	类型	说明
axisID	string	点动运动轴， <b>请注意大小写</b> 。不携带或携带错误的参数表示停止点动机器人 J1+ 表示关节1正方向运动， J1- 表示关节1负方向运动 J2+ 表示关节2正方向运动， J2- 表示关节2负方向运动 J3+ 表示关节3正方向运动， J3- 表示关节3负方向运动 J4+ 表示关节4正方向运动， J4- 表示关节4负方向运动 J5+ 表示关节5正方向运动， J5- 表示关节5负方向运动 J6+ 表示关节6正方向运动， J6- 表示关节6负方向运动 X+ 表示X轴正方向运动， X- 表示X轴负方向运动 Y+ 表示Y轴正方向运动， Y- 表示Y轴负方向运动 Z+ 表示Z轴正方向运动， Z- 表示Z轴负方向运动 Rx+ 表示Rx轴正方向运动， Rx- 表示Rx轴负方向运动 Ry+ 表示Ry轴正方向运动， Ry- 表示Ry轴负方向运动

		Rz+ 表示Rz轴正方向运动，Rz- 表示Rz轴负方向运动
--	--	-------------------------------

## 可选参数

参数名	类型	说明
coordtype	string	格式为“coordtype=value”。value表示指定运动轴所属的坐标系。0表示关节点动，1表示用户坐标系，2表示工具坐标系。默认值为上次成功调用时的设置值。 当axisID为关节轴时，coordtype只能取值0（忽略用户携带的该参数）。 当axisID为笛卡尔坐标轴时，coordtype只能取值1或2，取值为0会返回错误码-6。
user	string	格式为"user=index"，index为已标定的用户坐标系索引。取值范围：[0,50]。
tool	string	格式为"tool=index"，index为已标定的工具坐标系索引。取值范围：[0,50]。

## 返回

```
ErrorID, {}, MoveJog(axisID, coordtype, user, tool);
```

## 示例1

```
MoveJog(J2-)
// 停止点动
MoveJog()
```

沿J2轴负方向点动，然后停止点动。

## 示例2

```
MoveJog(X+, coordtype=1, user=1)
// 停止点动
MoveJog()
```

沿用户坐标系1的X轴正方向点动，然后停止点动。

## 示例3

```
MoveJog(J2-, coordtype=1, user=1)
// 停止点动
MoveJog()
```

沿J2轴负方向点动，然后停止点动。axisID指定关节时，可选参数无效。

## RunTo

### 原型

```
RunTo(P,moveType,user,tool,a,v)
```

### 描述

从当前位置运动至目标点。

**该指令为立即指令，支持在工程暂停时调用。**

### 必选参数

参数名	类型	说明
P	string	目标点，支持关节变量或位姿变量。 格式为"joint = {j1, j2, j3, j4, j5, j6}"或"pose = {x, y, z, rx, ry, rz}"。

### 可选参数

参数名	类型	说明
moveType	string	格式为“moveType=value”。value表示运动类型，0表示关节运动，1表示直线运动，默认值为0。
user	string	格式为"user=index"，index为已标定的用户坐标系索引。取值范围：[0,50]。
tool	string	格式为"tool=index"，index为已标定的工具坐标系索引。取值范围：[0,50]。
a	string	格式为“a=value”。value表示执行该条指令时的机械臂运动加速度比例。取值范围：[1,100]。
v	string	格式为“v=value”。value表示执行该条指令时的机械臂运动速度比例。取值范围：[1,100]。

### 返回

```
ErrorID, {}, RunTo(P,moveType,user,tool,a,v);
```

### 示例1

```
RunTo(joint = {0, 0, 90, 0, 90, 90}, moveType = 0, a = 20, v = 50)
```

机器人从当前位置以50%速度，20%加速度通过关节运动方式运动至关节坐标{0, 0, 90, 0, 90, 90}。

## 示例2

```
RunTo(pose= {-500,100,200,150,0,90}, moveType = 1, user = 1, tool = 0, a = 20, v = 50)
```

机器人从当前位置以50%速度，20%加速度通过直线运动方式运动至笛卡尔坐标点{-500,100,200,150,0,90}（用户坐标系1，工具坐标系0）。

## GetStartPose

### 原型

```
GetStartPose(traceName)
```

### 描述

获取指定轨迹的第一个点位。

### 必选参数

参数名	类型	说明
traceName	string	轨迹文件名（含后缀）； 轨迹文件存放在/dobot/userdata/project/process/trajectory/； 如果名称包含中文，必须将发送端的编码方式设置为UTF-8，否则会导致中文接收异常。

### 返回

```
ErrorID,{pointtype,{j1,j2,j3,j4,j5,j6},user,tool,{x,y,z,rx,ry,rz}},GetStartPose(traceName);
```

其中pointtype表示返回点位的类型，0表示示教点，1表示关节变量，2表示位姿变量。根据点位类型不同，携带的点位数据也有所不同，示例如下：

```
ErrorID,{0,{j1,j2,j3,j4,j5,j6},user,tool,{x,y,z,rx,ry,rz}},GetStartPose(traceName); // 示教点  
ErrorID,{1,{j1,j2,j3,j4,j5,j6}},GetStartPose(traceName); // 关节变量  
ErrorID,{2,{x,y,z,rx,ry,rz}},GetStartPose(traceName); // 位姿变量
```

### 示例

```
GetStartPose(recv_string.csv)
```

获取recv\_string.csv中记录的第一个点位。

## StartPath

### 原型

```
StartPath(traceName,isConst,multi,sample,freq,user,tool)
```

### 描述

根据指定的轨迹文件中的记录点位进行运动，复现录制的运动轨迹。

下发轨迹复现指令成功后，用户可以通过RobotMode指令查询机器人运行状态，

ROBOT\_MODE\_RUNNING表示机器人在轨迹复现运行中，变成ROBOT\_MODE\_IDLE表示轨迹复现运行完成，ROBOT\_MODE\_ERROR表示报警。

### 必选参数

参数名	类型	说明
traceName	string	轨迹文件名（含后缀）； 轨迹文件存放在/dobot/userdata/project/process/trajectory/； 如果名称包含中文，必须将发送端的编码方式设置为UTF-8，否则会导致中文接收异常。

### 可选参数

参数名	类型	说明
isConst	string	格式为“isConst=value”。value表示是否匀速复现，默认值为0。 isConst=1表示匀速复现，机械臂会按照全局速率匀速复现轨迹。 isConst=0表示按照轨迹录制时的原速复现，并可以使用multi参数等比缩放运动速度，此时机械臂的运动速度不受全局速率的影响。
multi	string	格式为“multi=value”。value表示复现时的速度倍数，仅当isConst=0时有效。 取值范围：[0.25, 2]，默认值为1。
sample	string	格式为“sample=value”。value表示轨迹点位采样间隔，即生成轨迹文件时相邻两个点位的采样时间差。取值范围：[8,1000]，单位ms，默认值为50ms（控制器录制轨迹文件时的采样间隔）。
freq	string	格式为“freq=value”。value表示滤波系数，该参数的值越小，复现的轨迹曲线越平滑，但相对原轨迹的变形越严重。请根据原轨迹的平滑程度设置合适的滤波系数。取值范围：(0,1]，当取值为1时，表示关闭滤波；默认值为0.2。
user	string	格式为"user=index"，index为轨迹点位对应的用户坐标系索引，不指

user	string	定时使用轨迹文件中记录的用户坐标系索引。取值范围：[0,50]。
tool	string	格式为"tool=index", index为轨迹点位对应的工具坐标系索引，不指定时使用轨迹文件中记录的坐标系索引。取值范围：[0,50]。

## 返回

```
ErrorID,{},StartPath(traceName,isConst,multi,sample,freq,user,tool);
```

## 示例

```
StartPath(recv_string.csv,isConst=0,multi=1,sample=20,freq=1,user=0,tool=0)
```

按原速复现recv\_string.csv中记录的轨迹。轨迹点位采样间隔为20ms，滤波系数为1（完全还原录制的轨迹），用户和工具坐标系均为0。

# RelMovJTool

## 原型

```
RelMovJTool(offsetX,offsetY,offsetZ,offsetRx,offsetRy,offsetRz,user,tool,a,v,cp)
```

## 描述

沿工具坐标系进行相对运动，末端运动方式为关节运动。

## 必选参数

参数名	类型	说明
offsetX	double	X轴方向偏移量，单位：mm。
offsetY	double	Y轴方向偏移量，单位：mm。
offsetZ	double	Z轴方向偏移量，单位：mm。
offsetRx	double	Rx轴方向偏移量，单位：度。
offsetRy	double	Ry轴方向偏移量，单位：度。
offsetRz	double	Rz轴方向偏移量，单位：度。

## 可选参数

参数名	类型	说明
user	string	格式为"user=index", index为已标定的用户坐标系索引。取值范围：[0,50]。



tool	string	格式为"tool=index"，index为已标定的工具坐标系索引。取值范围：[0,50]。
a	string	格式为"a=value"。value表示执行该条指令时的机器人运动加速度比例。取值范围：[1,100]。
v	string	格式为"v=value"。value表示执行该条指令时的机器人运动速度比例。取值范围：[1,100]。
cp	string	格式为"cp=value"。value表示平滑过渡比例。取值范围：[0,100]。

### 返回

```
ErrorID,{ResultID},RelMovJTool(offsetX,offsetY,offsetZ,offsetRx,offsetRy,offsetRz,user,tool,a,v,cp);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

### 示例

```
RelMovJTool(10,10,10,0,0,0)
```

机器人沿工具坐标系进行相对关节运动，在X、Y、Z轴上各偏移10mm。

## RelMovLTool

### 原型

```
RelMovLTool(offsetX,offsetY,offsetZ,offsetRx,offsetRy,offsetRz,user,tool,a,v|speed,cp|r)
```

### 描述

沿工具坐标系进行相对运动，末端运动方式为直线运动。

### 必选参数

参数名	类型	说明
offsetX	double	X轴方向偏移量，单位：mm。
offsetY	double	Y轴方向偏移量，单位：mm。
offsetZ	double	Z轴方向偏移量，单位：mm。
offsetRx	double	Rx轴方向偏移量，单位：度。
offsetRy	double	Ry轴方向偏移量，单位：度。
offsetRz	double	Rz轴方向偏移量，单位：度。

## 可选参数

参数名	类型	说明
user	string	格式为"user=index", index为已标定的用户坐标系索引。取值范围：[0,50]。
tool	string	格式为"tool=index", index为已标定的工具坐标系索引。取值范围：[0,50]。
a	string	格式为"a=value"。value表示执行该条指令时的机器人运动加速度比例。取值范围：[1,100]。
v	string	格式为"v=value"。value表示执行该条指令时的机器人运动速度比例，与speed互斥。取值范围：[1,100]。
speed	string	格式为"speed=value"。value表示执行该条指令时的机器人运动目标速度，与v互斥，若同时存在以speed为准。取值范围：[1, 最大运动速度]，单位：mm/s。
cp	string	格式为"cp=value"。value表示平滑过渡比例，与r互斥。取值范围：[0,100]。
r	string	格式为"r=value"。value表示平滑过渡半径，与cp互斥，若同时存在以r为准。单位：mm。

## 返回

```
ErrorID,{ResultID},RelMovLTool(offsetX,offsetY,offsetZ,offsetRx,offsetRy,offsetRz,user,tool,a,v|speed,cp|r);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

## 示例

```
RelMovLTool(10,10,10,0,0,0)
```

机器人沿工具坐标系进行相对直线运动，在X、Y、Z轴上各偏移10mm。

## RelMovJUser

### 原型

```
RelMovJUser(offsetX,offsetY,offsetZ,offsetRx,offsetRy,offsetRz,user,tool,a,v,cp)
```

### 描述

沿用户坐标系进行相对运动，末端运动方式为关节运动。

## 必选参数

参数名	类型	说明
offsetX	double	X轴方向偏移量，单位：mm。
offsetY	double	Y轴方向偏移量，单位：mm。
offsetZ	double	Z轴方向偏移量，单位：mm。
offsetRx	double	Rx轴偏移量，单位：度。
offsetRy	double	Ry轴偏移量，单位：度。
offsetRz	double	Rz轴偏移量，单位：度。

## 可选参数

参数名	类型	说明
user	string	格式为"user=index"，index为已标定的用户坐标系索引。取值范围：[0,50]。
tool	string	格式为"tool=index"，index为已标定的工具坐标系索引。取值范围：[0,50]。
a	string	格式为"a=value"。value表示执行该条指令时的机器人运动加速度比例。取值范围：[1,100]。
v	string	格式为"v=value"。value表示执行该条指令时的机器人运动速度比例。取值范围：[1,100]。
cp	string	格式为"cp=value"。value表示平滑过渡比例。取值范围：[0,100]。

## 返回

```
ErrorID,{ResultID},RelMovJUser(offsetX,offsetY,offsetZ,offsetRx,offsetRy,offsetRz,user,tool,a,v,cp);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

## 示例

```
RelMovJUser(10,10,10,0,0,0)
```

机器人沿用户坐标系进行相对关节运动，在X、Y、Z轴上各偏移10mm。

# RelMovLUser

## 原型

```
RelMovLUser(offsetX,offsetY,offsetZ,offsetRx,offsetRy,offsetRz,user,tool,a,v|speed,cp|r)
```

## 描述

沿用户坐标系进行相对运动，末端运动方式为直线运动。

## 必选参数

参数名	类型	说明
offsetX	double	X轴方向偏移量，单位：mm。
offsetY	double	Y轴方向偏移量，单位：mm。
offsetZ	double	Z轴方向偏移量，单位：mm。
offsetRx	double	Rx轴偏移量，单位：度。
offsetRy	double	Ry轴偏移量，单位：度。
offsetRz	double	Rz轴偏移量，单位：度。

## 可选参数

参数名	类型	说明
user	string	格式为"user=index"，index为已标定的用户坐标系索引。取值范围：[0,50]。
tool	string	格式为"tool=index"，index为已标定的工具坐标系索引。取值范围：[0,50]。
a	string	格式为"a=value"。value表示执行该条指令时的机器人运动加速度比例。取值范围：[1,100]。
v	string	格式为"v=value"。value表示执行该条指令时的机器人运动速度比例，与speed互斥。取值范围：[1,100]。
speed	string	格式为"speed=value"。value表示执行该条指令时的机器人运动目标速度，与v互斥，若同时存在以speed为准。取值范围：[1,最大运动速度]，单位：mm/s。
cp	string	格式为"cp=value"。value表示平滑过渡比例，与r互斥。取值范围：[0,100]。
r	string	格式为"r=value"。value表示平滑过渡半径，与cp互斥，若同时存在以r为准。单位：mm。

## 返回

```
ErrorID,{ResultID},RelMovLUser(offsetX,offsetY,offsetZ,offsetRx,offsetRy,offsetRz,user,tool,a,v|speed,cp|r);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

示例

```
RelMovLUser(10,10,10,0,0,0)
```

机器人沿用户坐标系进行相对直线运动，在X、Y、Z轴上各偏移10mm。

RelJointMovJ

原型

```
RelJointMovJ(offset1,offset2,offset3,offset4,offset5,offset6,user,tool,a,v,cp)
```

描述

沿关节坐标系进行相对运动，末端运动方式为关节运动。

必选参数

参数名	类型	说明
offset1	double	J1轴偏移量，单位：度。
offset2	double	J2轴偏移量，单位：度。
offset3	double	J3轴偏移量，单位：度。
offset4	double	J4轴偏移量，单位：度。
offset5	double	J5轴偏移量，单位：度。
offset6	double	J6轴偏移量，单位：度。

可选参数

参数名	类型	说明
user	string	格式为"user=index"，index为已标定的用户坐标系索引。取值范围：[0,50]。
tool	string	格式为"tool=index"，index为已标定的工具坐标系索引。取值范围：[0,50]。
a	string	格式为"a=value"。value表示执行该条指令时的机器人运动加速度比例。取值范围：[1,100]。
v	string	格式为"v=value"。value表示执行该条指令时的机器人运动速度比例。取值范围：[1,100]。
		格式为"cp=value"。value表示平滑过渡比例。取值范围：

		[0,100]。
--	--	----------

### 返回

```
ErrorID,{ResultID},RelJointMovJ(offset1,offset2,offset3,offset4,offset5,offset6,user,tool,a,v,
cp);
```

ResultID为算法队列ID，可用于判断指令执行顺序。

### 示例

```
RelJointMovJ(10,10,10,0,0,0)
```

机器人J1，J2，J3轴分别偏移10度。

## RelPointTool

### 原型

```
RelPointTool(p, {offsetX,offsetY,offsetZ,offsetRx,offsetRy,offsetRz})
```

### 描述

沿工具坐标系笛卡尔点偏移。

### 必选参数

参数名	类型	说明
p	string	格式为"joint = {j1, j2, j3, j4, j5, j6}" 或"pose = {x, y, z, rx, ry, rz}"。表示偏移的起始点位。
{offsetX,offsetY,offsetZ,offsetRx,offsetRy,offsetRz}	double	在笛卡尔坐标系下沿X轴、Y轴、Z轴、Rx轴、Ry轴、Rz轴方向上的偏移量。

### 返回

```
ErrorID,{X,Y,Z,Rx,Ry,Rz},RelPointTool(p, {offsetX,offsetY,offsetZ,offsetRx,offsetRy,offsetRz})
;
```

{X,Y,Z,Rx,Ry,Rz}表示笛卡尔坐标值。

## RelPointUser

### 原型

```
RelPointUser(p, {offsetX,offsetY,offsetZ,offsetRx,offsetRy,offsetRz})
```

## 描述

沿用户坐标系笛卡尔点偏移。

## 必选参数

参数名	类型	说明
p	string	格式为"joint = {j1, j2, j3, j4, j5, j6}" 或 "pose = {x, y, z, rx, ry, rz}"。表示偏移的起始点位。
{offsetX,offsetY,offsetZ,offsetRx,offsetRy,offsetRz}	double	在笛卡尔坐标系下沿X轴、Y轴、Z轴、Rx轴、Ry轴、Rz轴方向上的偏移量。

## 返回

```
ErrorID,{X,Y,Z,Rx,Ry,Rz},RelPointUser(p, {offsetX,offsetY,offsetZ,offsetRx,offsetRy,offsetRz})  
;
```

{X,Y,Z,Rx,Ry,Rz}表示笛卡尔坐标值。

# RelJoint

## 原型

```
RelJoint(J1,J2,J3,J4,J5,J6,{offset1,offset2,offset3,offset4,offset5,offset6})
```

## 描述

关节点位偏移。

## 必选参数

参数名	类型	说明
J1	double	点J1轴位置，单位：度。
J2	double	点J2轴位置，单位：度。
J3	double	点J3轴位置，单位：度。
J4	double	点J4轴位置，单位：度。
J5	double	点J5轴位置，单位：度。
J6	double	点J6轴位置，单位：度。
{offset1,offset2,offset3,offset4,offset5,offset6}	double	关节1/2/3/4/5/6的偏移值，单位：度。

offset4,offset5,offset6}	double	关节1/2/3/4/5/6的偏移值，单位：度。
--------------------------	--------	-------------------------

## 返回

```
ErrorID,{J1,J2,J3,J4,J5,J6},RelJoint(J1,J2,J3,J4,J5,J6,{offset1,offset2,offset3,offset4,offset5,offset6});
```

{J1,J2,J3,J4,J5,J6}表示关节值。

# GetCurrentCommandID

## 原型

```
GetCurrentCommandID()
```

## 描述

获取当前执行指令的算法队列ID，可以用于判断当前机器人执行到了哪一条指令。

下列的指令下发成功后会立刻返回，代表指令已被接受，实际上指令会进入算法队列，在后台按顺序排队执行，下发时返回的ResultID就是该指令在算法队列中的ID。

```
User(), Tool(), SetPayload(), DO(), ToolDO(), AO(), SetCollisionLevel(), DOGroup(), SetSafeWallEnable(), SetBackDistance(), SetPostCollisionMode(), SetUser(), SetTool(), MovJ(), MovL(), MovLIO(), MovJIO(), Arc(), Circle(), StartPath(), RelMovJTool(), RelMovLTool(), RelMovJUser(), RelMovLUser(), RelJointMovJ(), EnableSafeSkin(), SetSafeSkin()
```

机器人当前实际执行到了哪条指令，以及指令是否执行完毕，需要结合算法指令ID和机器人状态判断，参考本指令的示例。

## 返回

```
ErrorID,{ResultID},GetCurrentCommandID();
```

ResultID为当前执行指令的算法队列ID。

## 示例

```
MovJ(P1)
uint64_t p2Id = parseResultId(MovJ(P2)); // parseResultId用于获取指令返回的ResultID，请自行实现

while(true) {
    uint64_t currentId = parseResultId (GetCurrentCommndID()); // 获取当前执行指令的ResultID
    bool isStop = parseResultId (RobotMode()) == 5; // RobotMode为5表示使能且空闲，即运动指令已执行完毕
    if (currentId == p2Id && isStop ) { // currentId等于p2Id，且运动指令执行完毕。
```



```
}  
Sleep(1);  
}
```

上述示例结合算法队列ID和机器人状态，判断机器人已运动到P2点，然后退出循环。

## 2.8 轨迹恢复指令

### 功能概述

**暂停状态支持点动**功能开启后，工程暂停时，用户可下发MoveJog、RunTo和进出拖拽模式的指令，改变机器人的位姿。继续工程前，用户可通过轨迹恢复指令将机器人恢复至暂停时的点位，避免继续工程后机器人动作异常。

### 指令列表

指令	功能	指令类型
SetResumeOffset	设置轨迹恢复的回退距离	立即指令
PathRecovery	开始轨迹恢复	立即指令
PathRecoveryStop	轨迹恢复过程中停止机器人	立即指令
PathRecoveryStatus	查询轨迹恢复状态	立即指令

### SetResumeOffset

#### 原型

```
SetResumeOffset(distance)
```

#### 描述

**该指令仅用于焊接工艺。**设置轨迹恢复的目标点位相对暂停时的点位沿焊缝回退的距离。



#### 说明：

- 该指令仅在焊接过程中（即WeldArcSpeed生效时）生效。
- 该指令需要在工程暂停前下发才会生效。

#### 必选参数

参数名	类型	说明
distance	double	回退距离，单位：mm。

#### 返回

```
ErrorID, {}, SetResumeOffset(distance);
```

## PathRecovery

### 原型

```
PathRecovery()
```

### 描述

开始轨迹恢复：工程暂停后，控制机器人回到暂停时的位姿。

#### 说明：

- 该指令仅控制机器人回到暂停时的位姿，如需继续工程需要再下发Continue指令。
- 该指令为异步接口，下发后立刻返回，机器人是否已返回了暂停时的位姿需要通过PathRecoveryStatus指令判断。

### 返回

```
ErrorID, {}, PathRecovery();
```

## PathRecoveryStop

### 原型

```
PathRecoveryStop()
```

### 描述

轨迹恢复的过程中停止机器人。

### 返回

```
ErrorID, {}, PathRecoveryStop();
```

## PathRecoveryStatus

### 原型

```
PathRecoveryStatus()
```

### 描述

查询轨迹恢复的状态。

## 返回

```
ErrorID,{status},PathRecoveryStatus());
```

其中status表示轨迹恢复的状态：

- 0：已回到暂停时的位姿。
- 1：未回到暂停时的位姿，与暂停时的位姿偏差较小。
- 2：未回到暂停时的位姿，与暂停时的位姿偏差较大。

## 指令示例

```
SetResumeOffset(10); //设置焊接回退距离为10mm

MovL(P1); //按全局速度直线运动至P1点
WeldArcSpeed(10); //设置焊接速度为10mm/s
WeldArcSpeedStart(); //开启焊接速度开关
MovL(P2); //按设置的焊接速度直线运动至P2点
WeldArcSpeedEnd(); //关闭焊接速度开关

//通过实时反馈信息或RobotMode轮询获取到工程暂停状态后
RunTo(P); //将暂停状态的机器人运动至安全点，进行人工处理
PathRecovery(); //机器人返回偏移后（沿焊缝回退10mm）的暂停点
PathRecoveryStop(); //轨迹恢复过程中发现异常，停止机器人。
RunTo(P); //再将机器人运动至安全点，进行人工处理
PathRecovery(); //机器人返回偏移后的暂停点
if(PathRecoveryStatus()==0)
{
    //机器人已回到偏移后的暂停点
    Continue(); //继续运行工程
}
```

# 2.9 日志导出指令

## 功能概述

该组指令用于导出机器人日志和查看导出状态。

## 指令列表

指令	功能	指令类型
LogExportUSB	将机器人日志导出至U盘	立即指令
GetExportStatus	获取日志导出状态	立即指令


## LogExportUSB

### 原型

```
LogExportUSB(range)
```

### 描述

将机器人日志导出至插在机器人控制柜USB接口的U盘根目录。

 说明：

- 导出日志时建议只插入一个U盘，避免导出失败。
- 如果U盘包含多个分区，日志会导出至第一个分区。部分存储设备（例如用作启动盘的U盘）第一个分区为隐藏分区，会导致在Windows中无法直接查看到导出的日志。
- 请勿在导出过程终拔出U盘，否则可能导致文件损坏，必须格式化U盘才可再次导出。

### 必选参数

参数名	类型	说明
range	int	导出范围。 0：导出logs/all 和logs/user文件夹的内容。 1：导出logs文件夹所有内容。

### 返回

```
ErrorID,{},LogExportUSB(range);
```

该指令下发会立刻返回，请通过[GetExportStatus](#)获取日志导出状态。如果在导出过程中下发该指令，会返回-1，表示指令执行失败。

### 示例

```
LogExportUSB(0)
```

导出logs/all和logs/user文件夹的内容至U盘。

## GetExportStatus

### 原型

```
GetExportStatus()
```

### 描述

获取日志导出的状态。

### 返回

```
ErrorID,{status},GetExportStatus();
```

其中status表示日志导出状态。

- 0：未开始导出
- 1：导出中
- 2：导出完成
- 3：导出失败，找不到U盘
- 4：导出失败，U盘空间不足
- 5：导出失败，导出过程中U盘被拔出

导出完成和导出失败的状态会保持到下次用户使用导出功能。

## 2.10 力控指令

### 功能概述

越疆支持选配六维力传感器，并通过力控插件实现力控功能的开关及设置。力控拖拽是指基于末端受力分析的拖拽示教功能，即用户施加一个力在末端六维力传感器上，机器人顺应力的方向进行运行，运动速度和力的大小在一定范围内成正比。实际应用中，用户还可约束机器人的运动方向，使其只能沿一个或几个方向运动。

### 指令列表

指令	功能	指令类型
EnableFTSensor	开启/关闭力传感器	立即指令
SixForceHome	力传感器回零	立即指令
GetForce	获取力传感器数值	立即指令
ForceDriveMode	进入力控拖拽模式	立即指令
ForceDriveSpeed	设置力控拖拽速度	立即指令
StopDrag	退出拖拽模式	立即指令
FCForceMode	以用户指定的参数开启力控	队列指令
FCSetDeviation	设置力控模式下的位移和姿态偏差	立即指令
FCSetForceLimit	设置最大力限制	立即指令
FCSetMass	设置力控模式下各方向的惯性系数	立即指令
FCSetStiffness	设置力控模式下各方向的弹性系数	立即指令
FCSetDamping	设置力控模式下各方向的阻尼系数	立即指令
FCOff	退出力控模式	队列指令
FCSetForceSpeedLimit	设置各方向的力控调节速度	立即指令
FCSetForce	实时调整恒力设置	立即指令

### EnableFTSensor

#### 原型

```
EnableFTSensor(status)
```

## 描述

开启/关闭力传感器。

## 必选参数

参数名	类型	说明
status	int	力传感器开关，1表示开启，0表示关闭。

## 返回

```
ErrorID, {}, EnableFTSensor(status);
```

## 示例

```
EnableFTSensor(1)
```

打开力传感器。

# SixForceHome

## 原型

```
SixForceHome()
```

## 描述

将力传感器当前数值置0，即以传感器当前受力状态作为零点。

## 返回

```
ErrorID, {}, SixForceHome();
```

## 示例

```
SixForceHome()
```

将力传感器当前数值置0。

# GetForce

## 原型

```
GetForce(tool)
```



## 描述

获取力传感器当前数值。

## 可选参数

参数名	类型	说明
tool	int	用于指定获取数值时参考的工具坐标系，取值范围：[0,50]。 不指定时使用 <a href="#">全局工具坐标系</a> 。

## 返回

```
ErrorID,{Fx,Fy,Fz,Mx,My,Mz},GetForce(tool);
```

Fx、Fy、Fz为参考坐标系下各个方向的力值，Mx、My、Mz为扭矩值。

## 示例

```
GetForce(1)
```

获取力传感器当前受力在工具坐标系1下的数值。

# ForceDriveMode

## 原型

```
ForceDriveMode({x,y,z,rx,ry,rz},user)
```

## 描述

指定可拖拽的方向并进入力控拖拽模式。

## 必选参数

参数名	类型	说明
{x,y,z,rx,ry,rz}	string	用于指定可拖拽的方向。 0代表该方向不能拖拽，1代表该方向可以拖拽。 例： <ul style="list-style-type: none"> <li>{1,1,1,1,1,1}表示机械臂可在各轴方向上自由拖动</li> <li>{1,1,1,0,0,0}表示机械臂仅可在XYZ轴方向上拖动</li> <li>{0,0,0,1,1,1}表示机械臂仅可在RxRyRz轴方向上旋转。</li> </ul>

#### 可选参数

参数名	类型	说明
user	int	用于指定拖拽时参考的用户坐标系，取值范围：[0,50]。 不指定时表示不参考用户坐标系，参考 <a href="#">全局工具坐标系</a> 。

#### 返回

```
ErrorID, {}, ForceDriveMode({x,y,z,rx,ry,rz}, user);
```

#### 示例1

```
ForceDriveMode({1,1,1,1,1,1}, 1)
```

进入力控拖拽模式，可在用户坐标系1各轴方向上自由拖动。

#### 示例2

```
ForceDriveMode({1,1,1,0,0,0})
```

进入力控拖拽模式，可在全局工具坐标系XYZ轴方向上拖动。

## ForceDriveSpeed

#### 原型

```
ForceDriveSpeed(speed)
```

#### 描述

设置力控拖拽速度比例。

#### 必选参数

参数名	类型	说明
speed	int	力控拖拽速度比例，取值范围：[1,100]。

## 返回

```
ErrorID, {}, ForceDriveSpeed(speed);
```

## 示例

```
ForceDriveSpeed(10)
```

设置力控拖拽的速度比例为10。

# StopDrag

## 原型

```
StopDrag()
```

## 描述

机器人退出拖拽模式。关节拖拽和力控拖拽均使用此指令退出。

## 返回

```
ErrorID, {}, StopDrag();
```

## 示例

```
StopDrag()
```

机器人退出拖拽模式。

# FCForceMode

## 原型

```
FCForceMode({x,y,z,rx,ry,rz},{fx,fy,fz,frx,fry,frz},reference,user,tool)
```

## 描述

以用户指定的配置参数开启力控。

## 必选参数

参数名	类型	说明
-----	----	----

{x,y,z,rx,ry,rz}	string	开启/关闭笛卡尔空间某个方向的力控调节。 0表示关闭该方向的力控。 1表示开启该方向的力控。
{fx,fy,fz,frx,fry,frz}	string	目标力：是工具末端与作用对象之间接触力的目标值，是一种模拟力，可以由用户自行设定；目标力方向分别对应笛卡尔空间的{x,y,z,rx,ry,rz}方向。 位移方向的目标力范围[-200,200]，单位N；姿态方向的目标力范围[-12,12]，单位N/m。 目标力为0时处于柔顺模式，柔顺模式与力控拖动类似。 如果某个方向未开启力控调节，则该方向的目标力也不会生效。

### 可选参数

参数名	类型	说明
reference	string	格式为“reference=value”。value表示参考坐标系，默认参考工具坐标系。 reference=0表示参考工具坐标系，即沿工具坐标系进行力控调节。 reference=1表示参考用户坐标系，即沿用户坐标系进行力控调节。
user	string	格式为"user=index"，index为已标定的用户坐标系索引。取值范围：[0,50]。
tool	string	格式为"tool=index"，index为已标定的工具坐标系索引。取值范围：[0,50]。

### 返回

```
ErrorID,{ResultID},FCForceMode({x,y,z,rx,ry,rz},{fx,fy,fz,frx,fry,frz},reference,user,tool);
```

### 示例

```
FCForceMode({1,1,1,1,1,1},{100,100,100,10,10,10},reference=1,user=1)
```

参考已标定的用户坐标系1进行所有方向的力控调节，位移方向的目标力为100N，姿态方向的目标力为10N/m。

## FCSetDeviation

### 原型

```
FCSetDeviation({x,y,z,rx,ry,rz}, controltype)
```

### 描述

设置力控模式下的位移和姿态偏差，若力控过程中恒力偏移了较大的距离，机器人会进行相应处理。

**必选参数**

参数名	类型	说明
{x,y,z,rx,ry,rz}	string	x、y、z代表力控模式下的位移偏差，单位为mm。取值范围：(0,1000]，默认值100mm。 rx、ry、rz代表力控模式下的姿态偏差，单位为度。取值范围：(0,360]，默认值36度。

**可选参数**

参数名	类型	说明
controltype	int	表示力控过程中超过规定阈值时，机械臂的处理方式。 0：超过阈值时，机械臂报警（默认值）。 1：超过阈值时，机械臂停止搜寻而在原有轨迹上继续运动。

**返回**

```
ErrorID, {}, FCSetDeviation({x,y,z,rx,ry,rz}, controltype);
```

**示例**

```
FCSetDeviation({200,200,200,36,36,36})
```

设置力控模式下x、y、z方向的位移偏差为200mm，rx、ry、rz方向的姿态偏差为36°。  
TCP模式退出后，参数恢复默认值。

# FCSetForceLimit

**原型**

```
FCSetForceLimit(x,y,z,rx,ry,rz)
```

**描述**

设置各方向的最大力限制（该设置对所有方向均生效，包含未启用力控的方向）。

**必选参数**

参数名	类型	说明
x	double	x方向的力限制，取值范围：(0,500]，默认值500。

y	double	y方向的力限制，取值范围：(0,500]，默认值500。
z	double	z方向的力限制，取值范围：(0,500]，默认值500。
rx	double	rx方向的力限制，取值范围：(0,50]，默认值50。
ry	double	ry方向的力限制，取值范围：(0,50]，默认值50。
rz	double	rz方向的力限制，取值范围：(0,50]，默认值50。

## 返回

```
ErrorID,{},FCSetForceLimit(x,y,z,rx,ry,rz);
```

## 示例

```
FCSetForceLimit(500,500,500,50,50,50)
```

FCSetForceLimit未调用时，x、y、z方向的最大力限制默认为500；rx、ry、rz方向的最大力限制默认为50。

TCP模式退出后，参数恢复默认值。

# FCSetMass

## 原型

```
FCSetMass(x,y,z,rx,ry,rz)
```

## 描述

设置力控模式下各方向的惯性系数。

## 必选参数

参数名	类型	说明
x	double	x方向的惯性系数，取值范围：(0,10000]，默认值20。
y	double	y方向的惯性系数，取值范围：(0,10000]，默认值20。
z	double	z方向的惯性系数，取值范围：(0,10000]，默认值20。
rx	double	rx方向的惯性系数，取值范围：(0,10000]，默认值20。
ry	double	ry方向的惯性系数，取值范围：(0,10000]，默认值20。
rz	double	rz方向的惯性系数，取值范围：(0,10000]，默认值20。

## 返回

```
ErrorID, {}, FCSetMass(x, y, z, rx, ry, rz);
```

## 示例

```
FCSetMass(20, 20, 20, 20, 20, 20)
```

FCSetMass 未调用时，各方向的惯性系数默认为20。  
TCP模式退出后，参数恢复默认值。

## FCSetStiffness

### 原型

```
FCSetStiffness(x, y, z, rx, ry, rz)
```

### 描述

设置力控模式下各方向的弹性系数。

### 必选参数

参数名	类型	说明
x	double	x方向的弹性系数，取值范围：[0,10000]，默认值30。
y	double	y方向的弹性系数，取值范围：[0,10000]，默认值30。
z	double	z方向的弹性系数，取值范围：[0,10000]，默认值30。
rx	double	rx方向的弹性系数，取值范围：[0,10000]，默认值30。
ry	double	ry方向的弹性系数，取值范围：[0,10000]，默认值30。
rz	double	rz方向的弹性系数，取值范围：[0,10000]，默认值30。

### 返回

```
ErrorID, {}, FCSetStiffness(x, y, z, rx, ry, rz);
```

## 示例

```
FCSetStiffness(30, 30, 30, 30, 30, 30)
```

FCSetStiffness 未调用时，各方向的默认弹性系数为30。  
TCP模式退出后，参数恢复默认值。

# FCSetDamping

## 原型

```
FCSetDamping(x,y,z,rx,ry,rz)
```

## 描述

设置力控模式下各方向的阻尼系数。

## 必选参数

参数名	类型	说明
x	double	x方向的阻尼系数，取值范围：[0,1000]，默认值50。
y	double	y方向的阻尼系数，取值范围：[0,1000]，默认值50。
z	double	z方向的阻尼系数，取值范围：[0,1000]，默认值50。
rx	double	rx方向的阻尼系数，取值范围：[0,1000]，默认值50。
ry	double	ry方向的阻尼系数，取值范围：[0,1000]，默认值50。
rz	double	rz方向的阻尼系数，取值范围：[0,1000]，默认值50。

## 返回

```
ErrorID,{},FCSetDamping(x,y,z,rx,ry,rz);
```

## 示例

```
FCSetDamping(50,50,50,50,50,50)
```

FCSetDamping未调用时，各方向的默认阻尼系数为50。

TCP模式退出后，参数恢复默认值。

# FCOff

## 原型

```
FCOff()
```

## 描述

退出力控模式，与FCForceMode配合使用，两者之间的运动指令都会进行力的柔顺控制。

## 返回



```
ErrorID,{ResultID},FCOff();
```

## 示例

```
FCOff()
```

关闭力控。

## FCSetForceSpeedLimit

### 原型

```
FCSetForceSpeedLimit(x,y,z,rx,ry,rz)
```

### 描述

设置各方向的力控调节速度。力控速度上限较小时，力控调节速度较慢，适合低速平缓的接触面。力控速度上限较大时，力控调节速度快，适合高速力控应用。需要根据具体的应用场景进行调整。

### 必选参数

参数名	类型	说明
x	double	x方向的力控调节速度，默认值20mm/s。 CRA机型取值范围：(0,安全限制TCP速度值]。其他机型取值范围：(0,300]。
y	double	y方向的力控调节速度，默认值20mm/s。 CRA机型取值范围：(0,安全限制TCP速度值]。其他机型取值范围：(0,300]。
z	double	z方向的力控调节速度，默认值20mm/s。 CRA机型取值范围：(0,安全限制TCP速度值]。其他机型取值范围：(0,300]。
rx	double	rx方向的力控调节速度，默认值20°/s。 CRA机型取值范围：(0, (4安全限制TCP速度值 x0.001/ 3.14 x180) ]。其他机型取值范围：(0,90]。
ry	double	ry方向的力控调节速度，默认值20°/s。 CRA机型取值范围：(0, (4安全限制TCP速度值 x0.001/ 3.14 x180) ]。其他机型取值范围：(0,90]。
rz	double	rz方向的力控调节速度，默认值20°/s。 CRA机型取值范围：(0, (4安全限制TCP速度值 x0.001/ 3.14 x180) ]。其他机型取值范围：(0,90]。

### 返回

```
ErrorID,{},FCSetForceSpeedLimit(x,y,z,rx,ry,rz);
```

## 示例

```
FCSetForceSpeedLimit(20,20,20,20,20,20)
```

FCSetForceSpeedLimit未调用时，各方向的力控调节速度默认为20mm/s。  
TCP模式退出后，参数恢复默认值。

# FCSetForce

## 原型

```
FCSetForce(x,y,z,rx,ry,rz)
```

## 描述

实时调整各方向的恒力设置。

## 必选参数

参数名	类型	说明
x	double	x方向的恒力值。取值范围：[-200,200]，单位N。
y	double	y方向的恒力值。取值范围：[-200,200]，单位N。
z	double	z方向的恒力值。取值范围：[-200,200]，单位N。
rx	double	rx方向的恒力值。取值范围：[-12,12]，单位N/m。
ry	double	ry方向的恒力值。取值范围：[-12,12]，单位N/m。
rz	double	rz方向的恒力值。取值范围：[-12,12]，单位N/m。

## 返回

```
ErrorID,{},FCSetForce(x,y,z,rx,ry,rz);
```

## 示例

```
FCSetForce(50,50,50,10,10,10)
```

x、y、z方向的恒力设置为50N。rx、ry、rz方向的恒力设置为10N/m。

### 3 实时反馈信息

控制器通过30004、30005以及30006端口实时反馈机器人状态信息。

- 30004端口即实时反馈端口，客户端**每8ms**能收到一次机器人实时状态信息。
- 30005端口**每200ms**反馈机器人的信息。
- 30006端口为**可配置**的反馈机器人信息端口(默认为**每1000ms**反馈，如需修改，请联系技术支持)。

通过实时反馈端口每次收到的数据包有1440个字节，这些字节以标准的格式排列，如下表所示。

实时反馈的数据以小端（低位优先）的方式存储，即一个值用多个字节存储时，数据的低位存储在靠前的字节中。

例如，某个数据值为1234，转换为二进制为0000 0100 1101 0010，通过两个字节传递，第一个字节为1101 0010（二进制值的低8位），第二个字节为0000 0100（二进制值的高8位）。

含义	数据类型	值的数目	字节大小	字节位置值	描述
MessageSize	unsigned short	1	2	0000~0001	消息字节总长度
N/A	N/A	N/A	6	0002~0007	保留位
DigitalInputs	uint64	1	8	0008~0015	当前数字输入端子状态 详见 <a href="#">DI/DO说明</a>
DigitalOutputs	uint64	1	8	0016~0023	当前数字输出端子状态 详见 <a href="#">DI/DO说明</a>
RobotMode	uint64	1	8	0024~0031	机器人模式 详见 <a href="#">RobotMode指令说明</a>
TimeStamp	uint64	1	8	0032~0039	Unix时间戳 (单位ms)
RunTime	uint64	1	8	0040~0047	机器人开机运行时间 (单位ms)
TestValue	uint64	1	8	0048~0055	内存结构测试标准值 0x0123 4567 89AB CDEF
N/A	N/A	N/A	8	0056~0063	保留位
SpeedScaling	double	1	8	0064~0071	速度比例
N/A	N/A	N/A	16	0072~0087	保留位
VRobot	double	1	8	0088~0095	机器人电压
IRobot	double	1	8	0096~0103	机器人电流
ProgramState	double	1	8	0104~0111	脚本运行状态
SafetyIOIn	char	2	2	0112~0113	安全IO输入状态

SafetyIOOut	char	2	2	0114~0115	安全IO输出状态
N/A	N/A	N/A	76	0116~0191	保留位
QTarget	double	6	48	0192~0239	目标关节位置
QDTarget	double	6	48	0240~0287	目标关节速度
QDDTarget	double	6	48	0288~0335	目标关节加速度
ITarget	double	6	48	0336~0383	目标关节电流
MTarget	double	6	48	0384~0431	目标关节扭矩
QActual	double	6	48	0432~0479	实际关节位置
QDActual	double	6	48	0480~0527	实际关节速度
IActual	double	6	48	0528~0575	实际关节电流
ActualTCPForce	double	6	48	0576~0623	TCP各轴受力值 (通过六维力数据原始值计算)
ToolVectorActual	double	6	48	0624~0671	TCP笛卡尔实际坐标值
TCPSpeedActual	double	6	48	0672~0719	TCP笛卡尔实际速度值
TCPForce	double	6	48	0720~0767	TCP力值 (通过关节电流计算)
ToolVectorTarget	double	6	48	0768~0815	TCP笛卡尔目标坐标值
TCPSpeedTarget	double	6	48	0816~0863	TCP笛卡尔目标速度值
MotorTemperatures	double	6	48	0864~0911	关节温度
JointModes	double	6	48	0912~0959	关节控制模式。 8: 位置模式 10: 力矩模式
VActual	double	6	48	0960~1007	关节电压
HandType	char	4	4	1008~1011	手系 (备用参数)
User	char	1	1	1012	用户坐标系
Tool	char	1	1	1013	工具坐标系
RunQueuedCmd	char	1	1	1014	算法队列运行标志
PauseCmdFlag	char	1	1	1015	算法队列暂停标志
VelocityRatio	char	1	1	1016	关节速度比例 (0~100)
AccelerationRatio	char	1	1	1017	关节加速度比例 (0~100)
N/A	N/A	N/A	1	1018	保留位
XYZVelocityRatio	char	1	1	1019	笛卡尔位置速度比例 (0~100)
RVelocityRatio	char	1	1	1020	笛卡尔姿态速度比例 (0~100)

XYZAccelerationRatio	char	1	1	1021	笛卡尔位置加速度比例 (0~100)
RAccelerationRatio	char	1	1	1022	笛卡尔姿态加速度比例 (0~100)
N/A	N/A	N/A	2	1023~1024	保留位
BrakeStatus	char	1	1	1025	机器人抱闸状态 详见 <a href="#">BrakeStatus说明</a>
EnableStatus	char	1	1	1026	机器人使能状态
DragStatus	char	1	1	1027	机器人拖拽状态。 0: 不在拖拽状态, 1: 关节拖拽状态, 2: 力控拖拽状态
RunningStatus	char	1	1	1028	机器人运动状态
ErrorStatus	char	1	1	1029	机器人报警状态
JogStatusCR	char	1	1	1030	机器人点动状态
CRRobotType	char	1	1	1031	机器人型号 详见 <a href="#">RobotType说明</a>
DragButtonSignal	char	1	1	1032	末端按钮拖拽信号
EnableButtonSignal	char	1	1	1033	末端按钮使能信号
RecordButtonSignal	char	1	1	1034	末端按钮录制信号
ReappearButtonSignal	char	1	1	1035	末端按钮复现信号
JawButtonSignal	char	1	1	1036	末端按钮夹爪控制信号
SixForceOnline	char	1	1	1037	六维力传感器在线状态。 0: 离线 1: 在线 2: 异常
CollisionState	char	1	1	1038	碰撞状态
ArmApproachState	char	1	1	1039	小臂安全皮肤接近暂停
J4ApproachState	char	1	1	1040	J4安全皮肤接近暂停
J5ApproachState	char	1	1	1041	J5安全皮肤接近暂停
J6ApproachState	char	1	1	1042	J6安全皮肤接近暂停
N/A	N/A	N/A	61	1043~1103	保留位
VibrationDisZ	double	1	8	1104~1111	加速度计测量Z轴抖动位移
CurrentCommandId	uint64	1	8	1112~1119	当前运动队列id
MAActual[6]	double	6	48	1120~1167	六个关节的实际扭矩
Load	double	1	8	1168~1175	末端负载重量 (单位kg)
CenterX	double	1	8	1176~1183	末端负载X方向偏心距离 (单位mm)

CenterY	double	1	8	1184~1191	末端负载Y方向偏心距离 (单位mm)
CenterZ	double	1	8	1192~1199	末端负载Z方向偏心距离 (单位mm)
User[6]	double	6	48	1200~1247	用户坐标系坐标值
Tool[6]	double	6	48	1248~1295	工具坐标系坐标值
N/A	N/A	N/A	8	1296~1303	保留位
SixForceValue[6]	double	6	48	1304~1351	当前六维力数据原始值
TargetQuaternion[4]	double	4	32	1352~1383	[qw,qx,qy,qz] 目标四元数
ActualQuaternion[4]	double	4	32	1384~1415	[qw,qx,qy,qz] 实际四元数
AutoManualMode	char	1	2	1416~1417	手动/自动模式
ExportStatus	unsigned short	1	2	1418~1419	U盘导出状态
SafetyState	char	1	1	1420	1420 安全状态 1420:0 急停状态 (低有效) 1420:1 防护性停止状态 (低有效) 1420:2 缩减模式状态 (低有效) 1420:3 非停止状态 (低有效) 1420:4 运动中状态 (低有效) 1420:5 系统急停状态 (低有效) 1420:6 用户急停状态 (低有效) 1420:7 安全原点输出状态 (低有效, 不在安全原点时有效)
SafeState N/A	char	1	1	1421	安全状态保留位
N/A	N/A	N/A	18	1422~1439	保留位
TOTAL			1440		1440byte package

## 运动参数反馈值说明

如果在工程中单独设置了运动参数（速度、加速度等），相关反馈值并不会立刻更新，而是要等到机器人执行下一条运动指令时才会更新。

## DI/DO说明

DI/DO各占8个字节，每个字节有8位（二进制），最大可表示DI/DO各64个端口的状态。每个字节从低到高每一位表示一个端子的状态，1表示对应端子为ON，0表示对应端子为OFF或者无对应端子。

例如DI的第一个字节为0x01，二进制表示为00000001，从低到高分别表示D1\_1 ~ DI\_8的状态，即DI\_1为ON，其余7个DI都为OFF；

第二个字节为0x02，二进制表示为00000010，从低到高分别表示D1\_9 ~ DI\_16的状态，即DI\_10为ON，其余7个DI都为OFF；

后续字节以此类推，根据控制柜不同，IO端子的数量也不同，超过IO端子数量的二进制位会全部填充为0。

### BrakeStatus说明

该字节按位表达各个关节的抱闸状态，对应位为1表示该关节的抱闸已松开。位数与关节的对应关系如下表：

位数	7	6	5	4	3	2	1	0
含义	保留位	保留位	关节1	关节2	关节3	关节4	关节5	关节6

示例：

- 0x01 (00000001)：关节6抱闸松开
- 0x02 (00000010)：关节5抱闸松开
- 0x03 (00000011)：关节5和关节6抱闸松开
- 0x04 (00000100)：关节4抱闸松开

### RobotType说明

取值	代表机型
3	CR3
5	CR5
7	CR7
10	CR10
12	CR12
16	CR16
101	Nova 2
103	Nova 5
113	CR3A
115	CR5A
117	CR7A
120	CR10A
122	CR12A
126	CR16A
130	CR20A
150	Magician E6

## 4 通用错误码

错误码	描述	备注
0	无错误	命令下发成功
-1	命令执行失败	已收到命令，但执行失败了
-2	机器人处于报警状态	机器人报警状态下无法执行指令，需要清除报警后重新下发指令。
-3	机器人处于急停状态	机器人急停状态下无法执行指令，需要松开急停并清除报警后重新下发指令。
-4	机械臂处于下电状态	机械臂下电状态下无法执行指令，需要先给机械臂上电。
-5	机械臂处于脚本运行状态	机械臂处于脚本运行状态下拒绝执行部分指令，需要先暂停/停止脚本。
-6	MoveJog指令运动轴与运动类型不匹配	修改coordtype参数值，详见MoveJog指令说明。
-7	机械臂处于脚本暂停状态	机械臂处于脚本暂停状态下拒绝执行部分指令，需要先停止脚本。
-8	机械臂认证过期	机械臂处于不可用状态，需联系FAE处理。
...	...	...
-10000	命令错误	下发的命令不存在
-20000	参数数量错误	下发命令中的参数数量错误
-30001	当必选参数中有带名称的参数时，表示任意带名称的必选参数数据类型错误。 否则表示不带名称的第一个参数的参数数据类型错误	-3000X表示必选参数数据类型错误。 有带名称的必选参数时，表示带名称的必选参数类型错误，如joint="a" 否则最后一位1表示下发第1个必选参数的参数类型错误
-30002	不带名称的第二个必选参数的参数类型错误	-3000X表示必选参数类型错误。最后一位2表示下发第2个必选参数的参数类型错误
...	...	...
-40001	当必选参数中有带名称的参数时，表示任意带名称的必选参数范围错误。 第一个参数的参数范围错误	-4000X表示必选参数范围错误。 有带名称的必选参数时，表示带名称的必选参数范围错误，如joint={999,999,999,999,999,999} 否则最后一位1表示下发第1个必选参数的参数范围错误
	不带名称的第二个必选参数的参	-4000X表示必选参数范围错误。最后一位2



	数范围错误	表示下发第2个必选参数的参数范围错误
...	...	...
-50001	当可选参数中有带名称的参数时，表示任意带名称的可选参数数据类型错误。 否则表示不带名称的第一个可选参数的参数数据类型错误	-5000X表示可选参数数据类型错误。 有带名称的可选参数时，表示带名称的可选参数数据类型错误，如user="ss"。 否则最后一位1表示下发第1个可选参数的参数数据类型错误
-50002	不带名称的第二个可选参数的参数数据类型错误	-5000X表示可选参数数据类型错误。最后一位2表示下发第2个参数的参数数据类型错误
...	...	...
-60001	当可选参数中有带名称的参数时，表示任意带名称的可选参数范围错误。 否则表示不带名称的第一个可选参数的参数范围错误	-6000X表示可选参数范围错误。 有带名称的可选参数时，表示带名称的可选参数错误，如a=200。 最后一位1表示下发第1个可选参数的参数范围错误
-60002	不带名称的第二个可选参数的参数范围错误	-60000表示可选参数范围错误。最后一位2表示下发第2个可选参数的参数范围错误
...	...	...

**说明：**带名称的参数是指参数格式为"key=value"的参数。系统会从前往后检查参数，如果有多个参数错误，会报第一个检查到的错误的错误码。

### 报错示例1

```
// 原型: MovJ(P,user,tool,a,v,cp)
MovJ(joint="a",user=1, tool=0, a=20, v=50, cp=100)
```

上述示例中带名称的必选参数joint的数据类型错误，报-30001错误。

### 报错示例2

```
// 原型: DO(index,status,time)
DO(1,"2")
```

上述示例中不带名称的第二个必选参数数据类型错误，报-30002错误。

### 报错示例3

```
// 原型: MovJ(P,user,tool,a,v,cp)
MovJ(pose={-500,100,200,150,0,90},user="ss", tool=0, a=20, v=50, cp=100)
```

上述示例中带名称的可选参数user的数据类型错误，报-50001错误。

#### 报错示例4

```
// 原型: EnableRobot(load,centerX,centerY,centerZ)
EnableRobot(1.5,"a",0,30.5)
```

上述示例中不带名称的第二个可选参数数据类型错误，报-50002错误。

#### 报错示例5

```
// 原型: SetUser(index,table,type)
SetUser(1,{0,0,100,0,0,0}123,1)
```

系统检查参数类型前会先检查参数数量，而指令参数中的 `}` 到下一个 `,` 之间如果有其他字符，会导致参数被错误分解。例如 `{0,0,100,0,0,0}123` 会被分解为 `{0,0,100,0,0,0}` 和 `123` 两个参数，该指令报 -20000 (参数数量错误) 而不是 -30002 (必选参数2 类型错误)。

## 5 各状态下允许执行的TCP指令

RequestControl()的运行状态不在此处说明，具体请查看[RequestControl\(\)](#)指令。

### 错误状态

机器人处于报错状态（含急停）时允许执行的TCP指令：

- ClearError()
- GetErrorID()
- EmergencyStop()
- Stop()
- RobotMode()
- LogExportUSB()
- GetExportStatus()

### 下电状态

控制柜已开机，机器人未上电的状态。

除**错误状态**允许执行的指令外，还支持下述指令：

- PowerOn()

### 脚本运行状态

脚本工程运行中的状态，允许执行下述指令：

- SpeedFactor()
- RobotMode()
- DoInstant()
- ToolDoInstant()
- AOInstant()
- Stop()
- Pause()
- Continue()
- GetStartPose()
- PositiveKin()
- InverseSolution()
- GetAngle()
- GetPose()
- EmergencyStop()

- ModbusCreate()
- ModbusClose()
- GetInBits()
- GetInRegs()
- GetCoils()
- SetCoils()
- GetHoldRegs()
- SetHoldRegs()
- GetErrorID()
- DI()
- ToolDI()
- AI()
- ToolAI()
- DIGroup()
- GetDO()
- GetAO()
- GetDOGroup()
- SetTool485()
- SetToolPower()
- SetToolMode()
- CalcUser()
- CalcTool()
- GetInputBool()
- GetInputInt()
- GetInputFloat()
- GetOutputBool()
- GetOutputInt()
- GetOutputFloat()
- SetOutputBool()
- SetOutputInt()
- SetOutputFloat()
- GetCurrentCommandId()
- LogExportUSB()
- GetExportStatus()
- ClearError()
- GetForce()

## 脚本暂停状态

脚本工程暂停中的状态。

除**脚本运行状态**允许执行的指令外，还支持下述指令：

- EnableRobot()
- DisableRobot()
- RunTo()
- PathRecovery()
- StartDrag()
- StopDrag()
- SixForceHome()
- EnableFTSensor()
- ForceDriveMode()
- ForceDriveSpeed()

## 其他状态

无限制，指令均可发送。