

目录

Introduction	1.1
1 运动指令	1.2
1.1 点到点, 目标点位为笛卡尔	1.2.1
1.2 点到点, 目标点位为关节	1.2.2
1.3 直线运动	1.2.3
1.4 圆弧运动	1.2.4
1.5 门型运动	1.2.5
1.6 整圆运动	1.2.6
1.7 笛卡尔点位偏移	1.2.7
1.8 关节点位偏移	1.2.8
1.9 以直线运动至笛卡尔偏移位置	1.2.9
1.10 以点到点方式运动至笛卡尔偏移位置	1.2.10
1.11 以点到点方式运动至关节偏移角度	1.2.11
1.12 阻塞执行队列指令	1.2.12
1.13 Go运动并行设置	1.2.13
1.14 Move运动并行设置	1.2.14
1.15 MoveJ运动并行设置	1.2.15
1.16 检测Move执行后轨迹状态	1.2.16
1.17 检测是否运动到笛卡尔点位	1.2.17
2 运动参数设置指令	1.3
2.1 设置Go、MoveJ、GoR、MoveJR运动加速度比例	1.3.1
2.2 设置Move、Jump、Arc3、Circle3、MoveR运动加速度比例	1.3.2
2.3 设置Go、MoveJ、GoR、MoveJR运动速度比例	1.3.3
2.4 设置Move、Jump、Arc3、Circle3、MoveR运动速度比例	1.3.4
2.5 设置Jump门型参数索引	1.3.5
2.6 CP	1.3.6
2.7 设置Jump最大抬升高度	1.3.7
2.8 设置全局速度比例	1.3.8
2.9 设置Move、Jump、Arc3、Circle3、MoveR姿态运动速度比例	1.3.9
2.10 设置Move、Jump、Arc3、Circle3、MoveR姿态运动加速度比例	1.3.10
3 六维传感器运动指令	1.4
3.1 六维传感器回零	1.4.1
3.2 螺旋运动寻找孔位	1.4.2
3.3 旋转运动寻找孔位	1.4.3
3.4 线性插孔运动	1.4.4

4 输入输出指令	1.5
4.1 DI	1.5.1
4.2 DO	1.5.2
4.3 DOExecute(立即)	1.5.3
4.4 ToolDI	1.5.4
4.5 ToolDO	1.5.5
4.6 ToolDOExecute(立即)	1.5.6
4.7 ToolAnalogMode(立即)	1.5.7
4.8 ToolAI	1.5.8
4.9 AI	1.5.9
4.10 AO	1.5.10
4.11 AOExecute(立即)	1.5.11
4.12 WaitDI	1.5.12
5 程序管理指令	1.6
5.1 运动指令等待	1.6.1
5.2 脚本暂停时间	1.6.2
5.3 暂停程序运行	1.6.3
5.4 开始计时	1.6.4
5.5 结束计时	1.6.5
5.6 获取系统时间	1.6.6
5.7 打印	1.6.7
6 获取位姿指令	1.7
6.1 获取笛卡尔坐标	1.7.1
6.2 获取关节坐标	1.7.2
7 TCP	1.8
7.1 创建TCP	1.8.1
7.2 建立TCP连接	1.8.2
7.3 接收TCP数据	1.8.3
7.4 发送TCP数据	1.8.4
7.5 关闭TCP	1.8.5
7.6 读取单点数据	1.8.6
7.7 读取多点数据	1.8.7
8 UDP	1.9
8.1 创建UDP	1.9.1
8.2 接收UDP数据	1.9.2
8.3 发送UDP数据	1.9.3
9 Modbus	1.10
9.1 创建Modbus主站	1.10.1
9.2 断开连接	1.10.2

9.3 读取Modbus从站线圈寄存器地址的值	1.10.3
9.4 设置Modbus从站线圈寄存器地址的值	1.10.4
9.5 读取Modbus从站触点寄存器地址的值	1.10.5
9.6 读取Modbus从站输入寄存器地址的值	1.10.6
9.7 读取Modbus从站保持寄存器地址的值	1.10.7
9.8 设置Modbus从站保持寄存器地址的值	1.10.8
10 末端工具	1.11
10.1 设置末端工具供电状态	1.11.1
10.2 设置末端工具波特率	1.11.2
11 坐标系	1.12
11.1 修改用户坐标系	1.12.1
11.2 计算用户坐标系	1.12.2
11.3 修改工具坐标系	1.12.3
11.4 计算工具坐标系	1.12.4
12 编码器	1.13
12.1 设置编码器当前值	1.13.1
12.2 获取编码器当前位置	1.13.2
13 轨迹复现指令	1.14
13.1 轨迹拟合	1.14.1
13.2 获取轨迹首个点位(轨迹拟合)	1.14.2
13.3 轨迹复现	1.14.3
13.4 获取轨迹首个点位(轨迹复现)	1.14.4
14 负载	1.15
14.1 设置当前负载	1.15.1
14.2 控制负载设置状态	1.15.2
15 码垛	1.16
15.1 实例化矩阵踩盘变量	1.16.1
15.2 设置下次操作踩的序号	1.16.2
15.3 获取当前操作踩的序号	1.16.3
15.4 设置下次操作码垛层的序号	1.16.4
15.5 获取当前操作层序号	1.16.5
15.6 复位码垛	1.16.6
15.7 查询码垛或拆垛是否完成	1.16.7
15.8 释放码垛实例	1.16.8
15.9 机械臂从当前位姿按照配置的码垛路径运到第一踩目标点	1.16.9
15.10 机械臂从当前位姿按照配置的拆垛路径运到安全过渡点	1.16.10
16 传送带跟踪	1.17
16.1 设置传送带编号	1.17.1

16.2 获取传送带上工件状态	1.17.2
16.3 设置传送带用户坐标系下X、Y方向的偏移	1.17.3
16.4 设置时间补偿	1.17.4
16.5 同步指定的传送带	1.17.5
16.6 停止同步传送带	1.17.6
17 其他指令	1.18
17.1 设置安全皮肤开关	1.18.1
17.2 设置安全皮肤避障模式开关状态	1.18.2
17.3 设置碰撞等级	1.18.3

Introduction

Lua语言是CC系列控制柜采用的编程语言，用户可根据实际工艺需求，使用Lua语言在APP器上灵活的编写相应的控制程序。CC系列控制柜还封装了Dobot API指令供用户直接调用，简化了编程。本节描述常用的编程指令供用户参考。

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:26

运动指令

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:24

点到点，目标点位为笛卡尔

- 原型：

```
Go(P," User=1 Tool=2 CP=1 Speed=50 Accel=20 SYNC=1 ")
```

- 描述：从当前位置以点到点方式运动至笛卡尔坐标系下的目标位置
- 必选参数：P，表示目标点，可从“示教点”页面获取，也可自定义点位，但只支持笛卡尔坐标点位
- 可选参数：
 - CP：运动时设置平滑过渡，取值范围：0~100
 - Speed：运动速度比例，取值范围：1~100
 - Accel：运动加速度比例，取值范围：1~100
 - SYNC：同步标识，取值范围：0或1。SYNC = 0表示异步执行，调用后立即返回，但不关注指令执行情况；SYNC = 1表示同步执行，调用后，待指令执行完才返回
- 示例

```
Go(P1)
```

机械臂以默认设置运动至P1点

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间：2021-08-11 16:37:20

点到点，目标点位为关节

- 原型：

```
MoveJ(P,"CP=1 Speed=50 Accel=20 SYNC=1")
```

- 描述：从当前位置以点到点方式运动至目标关节角度
- 必选参数：P，表示目标关节角度，不能通过“示教点”页面导入，需先定义关节点位
- 可选参数：
 - CP：运动时设置平滑过渡，取值范围：0~100
 - Speed：运动速度比例，取值范围：1~100
 - Accel：运动加速度比例，取值范围：1~100
 - SYNC：同步标识，取值范围：0或1。SYNC = 0表示异步执行，调用后立即返回，但不关注指令执行情况；SYNC = 1表示同步执行，调用后，待指令执行完才返回
- 示例

```
local P = {joint={0,-0.0674194,0,0,0,0}}  
MoveJ(P)
```

自定义关节坐标点点位P，机械臂设置运动至P点

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间：2021-08-06 18:06:26

直线运动

- 原型:

```
Move(P," User=1 Tool=2 CP=1 SpeedS=50 AccelS=20 SYNC=1")
```

- 描述:从当前位置以直线方式运动至笛卡尔坐标系下的目标位置
- 必选参数:P, 表示目标点, 可从“示教点”页面获取, 也可自定义点位, 但只支持笛卡尔坐标点位
- 可选参数:
 - CP:运动时设置平滑过渡, 取值范围:0~100
 - SpeedS:运动速度比例, 取值范围:1~100
 - AccelS:运动加速度比例, 取值范围:1~100
 - SYNC:同步标识, 取值范围:0或1。SYNC = 0表示异步执行, 调用后立即返回, 但不关注指令执行情况; SYNC = 1表示同步执行, 调用后, 待指令执行完才返回
- 示例

```
Move(P1)
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-11 16:37:20

圆弧运动

- 原型:

```
Arc3(P1,P2, " User=1 Tool=2 CP=1 SpeedS=50 AccelS=20 SYNC=1")
```

- 描述:从当前位置以圆弧插补方式移动至笛卡尔坐标系下的目标位置

该指令需结合其他运动指令确定圆弧起始点

- 必选参数:

- P1, 表示圆弧中间点, 可从“示教点”页面获取, 也可自定义点位, 但只支持笛卡尔坐标点位
- P2: 表示圆弧结束点, 可从“示教点”页面获取, 也可自定义点位, 但只支持笛卡尔坐标点位

- 可选参数:

- CP: 运动时设置平滑过渡, 取值范围:0~100
- SpeedS: 运动速度比例, 取值范围:1~100
- AccelS: 运动加速度比例, 取值范围:1~100
- SYNC: 同步标识, 取值范围:0或1。SYNC = 0表示异步执行, 调用后立即返回, 但不关注指令执行情况;
SYNC = 1表示同步执行, 调用后, 待指令执行完才返回

- 示例

```
While true do  
  Go(P1)  
  Arc3(P2,P3)  
end
```

机械臂循环从P1点以圆弧的方式经过P2点运动至P3点

门型运动

- 原型:

```
Jump(P," User=1 Tool=2 SpeedS=50 AccelS=20 Start=10 ZLimit=80 End=50 SYNC=1")
```

- 描述:从当前位置以Move运动方式进行门型运动,使机械臂移动至目标位置。Jump需结合其他运动指令一起使用
- 必选参数:P,表示目标点,可从“示教点”页面获取,也可自定义点位,但只支持笛卡尔坐标点位。且P点高度不能高于ZLimit,否则会触发JUMP参数错误报警
- 可选参数:
 - SpeedS:运动速度比例,取值范围:1~100
 - AccelS:运动加速度比例,取值范围:1~100
 - Arch:设置Jump门型参数编号,取值范围:0~9
 - Start:起始点高度
 - ZLimit:最大抬升高度
 - End:结束点下降高度
 - SYNC:同步标识,取值范围:0或1。SYNC = 0表示异步执行,调用后立即返回,但不关注指令执行情况; SYNC = 1表示同步执行,调用后,待指令执行完才返回
- 示例

```
Go(P6)  
Jump(P5,"Start=10 ZLimit=600 End=10")
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-17
12:41:49

整圆运动

- 原型:

```
Circle3(P1,P2, Count, "User=1 Tool=2 CP=1 SpeedS=50 AccelS=20 SYNC=1")
```

- 描述:从当前位置以整圆插补方式移动至笛卡尔坐标系下的目标位置

该指令需结合其他运动指令确定圆弧起始点

- 必选参数:

- P1, 表示整圆中间点, 可从“示教点”页面获取, 也可自定义点位, 但只支持笛卡尔坐标点位
- P2: 表示整圆结束点, 可从“示教点”页面获取, 也可自定义点位, 但只支持笛卡尔坐标点位
- Count: 整圆个数, 取值范围: 1~999

- 可选参数:

- CP: 运动时设置平滑过渡, 取值范围: 0~100
- SpeedS: 运动速度比例, 取值范围: 1~100
- AccelS: 运动加速度比例, 取值范围: 1~100
- SYNC: 同步标识, 取值范围: 0或1。SYNC = 0表示异步执行, 调用后立即返回, 但不关注指令执行情况; SYNC = 1表示同步执行, 调用后, 待指令执行完才返回

- 示例

```
Go(P1)  
Circle3(P2,P3,1)
```

机械臂从P1点以整圆的方式经过P2点运动至P3点

笛卡尔点位偏移

- 原型:

```
RP(P1, {OffsetX, OffsetY, OffsetZ})
```

- 描述:笛卡尔坐标系下增加X、Y、Z方向上的偏移量并返回一个新的笛卡尔坐标点

除MoveJ外, 其他运动指令均支持运行至该点

- 可选参数:

- P1, 表示当前笛卡尔坐标点, 可从“示教点”页面获取, 也可自定义点位, 但只支持笛卡尔坐标点位
- OffsetX, OffsetY, OffsetZ: 笛卡尔坐标系下X轴、Y轴、Z轴方向上的偏移, 单位:毫米

- 返回:笛卡尔坐标点

- 示例

```
P2=RP(P1, {50,10,32})  
Move(P2)或 Move(RP(P1, {50,10,32}))
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:26

关节点位偏移

- 原型:

```
RJ(P1, {Offset1, Offset2, Offset3, Offset4, Offset5, Offset6})
```

- 描述: 关节坐标系下增加每个轴的偏移量, 并返回一个新的关节坐标点, 仅支持MoveJ运动指令运行至该点
- 参数:
 - P1, 表示当前关节点, 不能通过“示教点”页面导入, 需先定义
 - Offset1~Offset6: 关节坐标系下J1轴 ~ J6轴方向上的偏移。单位: 度
- 返回: 关节坐标点
- 示例

```
local P1 = {joint={0,-0.0674194,0,0,0,0}}  
P2=RJ(P1, {60,50,32,30,25,30})  
MoveJ(P2)或 MoveJ(RJ(P1, {60,50,32,30,25,30}))
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:26

以直线运动至笛卡尔偏移位置

- 原型:

```
MoveR({OffsetX, OffsetY, OffsetZ}, " User=1 Tool=2 CP=1 SpeedS=50 AccelS=20 SYNC=1")
```

- 描述: 从当前位置以直线方式运动至笛卡尔坐标系下偏移位置
- 必选参数:
- OffsetX, OffsetY, OffsetZ: 笛卡尔坐标系下X轴、Y轴、Z轴方向上的偏移, 单位: 毫米
- 可选参数:
 - CP: 运动时设置平滑过渡, 取值范围: 0~100
 - SpeedS: 运动速度比例, 取值范围: 1~100
 - AccelS: 运动加速度比例, 取值范围: 1~100
 - SYNC: 同步标识, 取值范围: 0或1。SYNC = 0表示异步执行, 调用后立即返回, 但不关注指令执行情况; SYNC = 1表示同步执行, 调用后, 待指令执行完才返回
- 示例

```
Go(P1)  
MoveR({20,20,20}, "AccelS=100 SpeedS=100 CP=100")
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-11 16:37:20

以点到点方式运动至笛卡尔偏移位置

- 原型:

```
GoR({OffsetX, OffsetY, OffsetZ}, " User=1 Tool=2 CP=1 Speed=50 Accel=20 SYNC=1 ")
```

- 描述: 从当前位置以点到点方式运动至笛卡尔坐标系下的偏移位置
- 必选参数: OffsetX, OffsetY, OffsetZ: 笛卡尔坐标系下X轴、Y轴、Z轴方向上的偏移, 单位: 毫米
- 可选参数:
 - Speed: 运动速度比例, 取值范围: 1~100
 - Accel: 运动加速度比例, 取值范围: 1~100
 - SYNC: 同步标识, 取值范围: 0或1。SYNC = 0表示异步执行, 调用后立即返回, 但不关注指令执行情况; SYNC = 1表示同步执行, 调用后, 待指令执行完才返回

- 示例

```
Go(P1)  
GoR({10,10,10}, "Accel=100 Speed=100 CP=100")
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-11 16:37:20

以点到点方式运动至关节偏移角度

- 原型:

```
MoveJR({Offset1, Offset2, Offset3, Offset4, Offset5, Offset6}, "CP=1 Speed=50 Accel=20 SYNC=1")
```

- 描述: 从当前位置以点到点方式运动至关节偏移角度
- 必选参数: Offset~Offset6: 关节坐标系下J1~J6轴方向上的偏移, 单位: 度
- 可选参数:
 - CP: 运动时设置平滑过渡, 取值范围: 0~100
 - Speed: 运动速度比例, 取值范围: 1~100
 - Accel: 运动加速度比例, 取值范围: 1~100
 - SYNC: 同步标识, 取值范围: 0或1。SYNC = 0表示异步执行, 调用后立即返回, 但不关注指令执行情况; SYNC = 1表示同步执行, 调用后, 待指令执行完才返回
- 示例

```
Go(P1)  
MoveJR({20, 20, 10, 0, 10, 0}, "SYNC=1")
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:24

阻塞执行队列指令

- 原型:

```
Sync()
```

- 描述:阻塞程序执行队列指令, 待所有队列指令执行完才返回
- 参数:无

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:26

Go运动并行设置

- 原型:

```
GoIO(P, { {Mode, Distance, Index, Status}, {Mode, Distance, Index, Status}...}, "ARM=Left User=1 Tool=2 CP=1 Speed=50 Accel=20 SYNC=1")
```

- 描述: 在Go运动模式下运动时并行设置数字输出端口状态

- 必选参数:

- P: 表示目标点, 可从“示教点”页面获取, 也可自定义点位, 但只支持笛卡尔坐标点位
- Mode: 设置Distance模式。0: Distance为距离百分比; 1: Distance为离起始点或目标点的距离
- Distance: 运行指定的距离
 - 若Mode为0, 则Distance表示起始点与目标点之间距离的百分比, 取值范围: 0~100
 - 若Mode为1, 则Distance表示离起始点或目标点的距离
 - 若Distance取值为正, 则表示离起始点的距离
 - 若Distance取值为负, 则表示离目标点的距离
- Index: 数字输出索引, 取值范围: 1~24
- Status: 数字输出状态, 取值范围: 0或1

- 可选参数:

- ARM: 机械臂方向。若机械臂为水平多关节机械臂(SCARA四轴), 则设置为Left或Right。若机械臂为垂直多关节机械臂(六轴), 则该参数无效
- CP: 运动时设置平滑过渡, 取值范围: 0~100
- Speed: 运动速度比例, 取值范围: 1~100
- Accel: 运动加速度比例, 取值范围: 1~100
- SYNC: 同步标识, 取值范围: 0或1。SYNC = 0表示异步执行, 调用后立即返回, 但不关注指令执行情况; SYNC = 1表示同步执行, 调用后, 待指令执行完才返回

- 返回: 无

- 示例

```
GoIO(P1, {0, 10, 2, 1})
```

机械臂以默认设置向P1点运动, 当运动到距离10%的位置时, 将数字输出端口2设置为高电平

Move运动并行设置

- 原型:

```
MoveIO(P, { {Mode, Distance, Index, Status},{Mode, Distance, Index, Status}...}, "ARM=Left User=1 Tool=2 CP=1 Speed=50 Accel=20 SYNC=1")
```

- 描述:在Move运动模式下运动时并行设置数字输出端口状态

- 必选参数:

- o P:表示目标点,可从“示教点”页面获取,也可自定义点位,但只支持笛卡尔坐标点位
- o Mode:设置Distance模式, 0:Distance为距离百分比; 1:Distance为离起始点或目标点的距离
- o Distance:运行指定的距离
 - 若Mode为0,则Distance表示起始点与目标点之间距离的百分比,取值范围:0~100
 - 若Mode为1,则Distance表示离起始点或目标点的距离
 - 若Distance取值为正,则表示离起始点的距离
 - 若Distance取值为负,则表示离目标点的距离
- o Index:数字输出索引,取值范围:1~24
- o Status:数字输出状态,取值范围:0或1

- 可选参数:

- o ARM:机械臂方向。若机械臂为水平多关节机械臂(SCARA四轴),则设置为Left或Right。若机械臂为垂直多关节机械臂(六轴),则该参数无效
- o CP:运动时设置平滑过渡,取值范围:0~100
- o Speed:运动速度比例,取值范围:1~100
- o Accel:运动加速度比例,取值范围:1~100
- o SYNC:同步标识,取值范围:0或1。SYNC = 0表示异步执行,调用后立即返回,但不关注指令执行情况; SYNC = 1表示同步执行,调用后,待指令执行完才返回

- 返回:无

- 示例

```
MoveIO(P1, {0, 10, 2, 1})
```

机械臂以默认设置向P1点运动,当运动到距离10%的位置时,将数字输出端口2设置为高电平

MoveJ运动并行设置

- 原型:

```
MoveJIO(P, { {Mode, Distance, Index, Status},{Mode, Distance, Index, Status}...}, "CP=1 Speed=50 Accel=20 SYNC=1")
```

- 描述:在MoveJ运动模式下运动时并行设置数字输出端口状态

- 必选参数:

- o P:表示目标关节角度, 不能通过“示教点”页面导入, 需先定义关节点位
- o Mode:设置Distance模式。0:Distance为距离百分比;1:Distance为离起始点或目标点的距离
- o Distance:运行指定的距离

若Mode为0, 则Distance表示起始点与目标点之间距离的百分比, 取值范围:0~100

若Mode为1, 则Distance表示离起始点或目标点的距离

若Distance取值为正, 则表示离起始点的距离

若Distance取值为负, 则表示离目标点的距离。

- o Index:数字输出索引, 取值范围:1~24
- o Status:数字输出状态, 取值范围:0或1

- 可选参数:

- o CP:运动时设置平滑过渡, 取值范围:0~100
- o Speed:运动速度比例, 取值范围:1~100
- o Accel:运动加速度比例, 取值范围:1~100
- o SYNC:同步标识, 取值范围:0或1。SYNC = 0表示异步执行, 调用后立即返回, 但不关注指令执行情况; SYNC = 1表示同步执行, 调用后, 待指令执行完才返回

- 返回:无

- 示例

```
MoveJIO (P1, {0, 10, 2, 1})
```

机械臂以默认设置向P1点运动, 当运动到距离10%的位置时, 将数字输出端口2设置为高电平

检测Move执行后轨迹状态

- 原型:

```
CheckMove(P," User=1 Tool=2 CP=1 SpeedS=50 AccelS=20 SYNC=1")
```

- 描述:检测Move指令执行后的轨迹状态
- 必选参数:P, 表示目标点, 可从“示教点”页面获取, 也可自定义点位, 但只支持笛卡尔坐标点位
- 可选参数:
 - CP:运动时设置平滑过渡, 取值范围:0~100
 - SpeedS:运动速度比例, 取值范围:1~100 DOBOT CR系列机器人编程语言
 - AccelS:运动加速度比例, 取值范围:1~100
 - SYNC:同步标识, 取值范围:0或1。SYNC = 0表示异步执行, 调用后立即返回, 但不关注指令执行情况; SYNC = 1表示同步执行, 调用后, 待指令执行完才返回
- 返回:返回运动指令的轨迹状态
 - 0:无错误
 - 16:终点接近肩部奇异
 - 17:终点逆解无解
 - 18:终点逆解限位
 - 22:手势切换错误
 - 26:终点接近腕部奇异
 - 27:终点接近肘部奇异
 - 29:速度参数错误
 - 32:轨迹有肩部奇异点
 - 33:轨迹存在逆解无解点
 - 34:轨迹存在逆解限位点
 - 35:轨迹有腕部奇异点
 - 36:轨迹有轴部奇异点
 - 37:轨迹存在关节跳变点
- 示例

```
local status=CheckMove(P1)
```

当机械臂以默认设置运动至P1点时, 检测轨迹状态

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-11 16:37:20

检测是否运动到笛卡尔点位

- 原型:

```
CheckGo(P," User=1 Tool=2 CP=1 Speed=50 Accel=20 SYNC=1")
```

- 描述:检测是否运动到点位
- 必选参数:P, 表示目标点, 可从“示教点”页面获取, 也可自定义点位, 但只支持笛卡尔坐标点位
- 可选参数:
 - CP:运动时设置平滑过渡, 取值范围:0~100
 - Speed:运动速度比例, 取值范围:1~100
 - Accel:运动加速度比例, 取值范围:1~100
 - SYNC:同步标识, 取值范围:0或1。SYNC = 0表示异步执行, 调用后立即返回, 但不关注指令执行情况; SYNC = 1表示同步执行, 调用后, 待指令执行完才返回
- 返回:返回运动指令的轨迹状态
 - 0:无错误
 - 16:终点接近肩部奇异
 - 17:终点逆解无解
 - 18:终点逆解限位
 - 22:手势切换错误
 - 26:终点接近腕部奇异
 - 27:终点接近肘部奇异
 - 29:速度参数错误
 - 32:轨迹有肩部奇异点
 - 33:轨迹存在逆解无解点
 - 34:轨迹存在逆解限位点
 - 35:轨迹有腕部奇异点
 - 36:轨迹有轴部奇异点
 - 37:轨迹存在关节跳变点
- 示例

```
local status=CheckGo (P1)
```

当机械臂以默认设置运动至P1点时, 检测轨迹状态

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-11 16:37:20

运动参数设置指令

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:24

设置Go、MoveJ、GoR、MoveJR运动加速度比例

- 原型:

```
Accel(R)
```

- 描述: 设置Go、MoveJ、GoR、MoveJR运动加速度比例
- 参数: R: 百分比, 取值范围1~100
- 示例

```
Accel(50)  
Go(P1)
```

表示机械臂以50%的加速度比例运动至P1点

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:26

设置Move、Jump、Arc3、Circle3、MoveR运动加速度比例

- 原型:

```
AccelS(R)
```

- 描述: 设置Move、Jump、Arc3、Circle3、MoveR运动加速度比例
- 参数: R: 百分比, 取值范围1~100
- 示例

```
AccelS(20)  
Move(P1)
```

表示机械臂以20%的加速度比例运动至P1点

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06 18:06:26

设置Go、MoveJ、GoR、MoveJR运动速度比例

- 原型:

```
Speed(R)
```

- 描述: 设置Go、MoveJ、GoR、MoveJR运动速度比例
- 参数: R: 百分比, 取值范围1~100
- 示例

```
Speed(20)  
Go(P1)
```

表示机械臂以20%的速度比例运动至P1点

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:24

设置Move、Jump、Arc3、Circle3、MoveR运动速度比例

- 原型:

```
SpeedS(R)
```

- 描述: 设置Move、Jump、Arc3、Circle3、MoveR运动速度比例
- 参数: R: 百分比, 取值范围1~100
- 示例

```
SpeedS(20)  
Move(P1)
```

表示机械臂以20%的速度比例运动至P1点

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:26

设置Jump门型参数索引

- 原型:

```
Arch(Index)
```

- 描述: 设置Jump门型参数索引(起始点抬升高度、最大抬升高度、结束点下降高度)

Jump门型参数需在“设置再现设置弧形参数”中设置

- 参数: Index: 门型参数索引, 取值范围: 0~9

需在“设置弧形再现参数”

- 示例

```
Arch(1)  
Jump(P1)
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:24

CP

- 原型:

```
CP(R)
```

- 描述: 设置平滑过渡比例, 即从起始点经过中间点到达终点时, 经过中间点是以直角方式过渡还是以曲线方式过渡, 如图 10.1所示

仅对Go、Move、Arc3、Circle3、GoR、MoveR有效

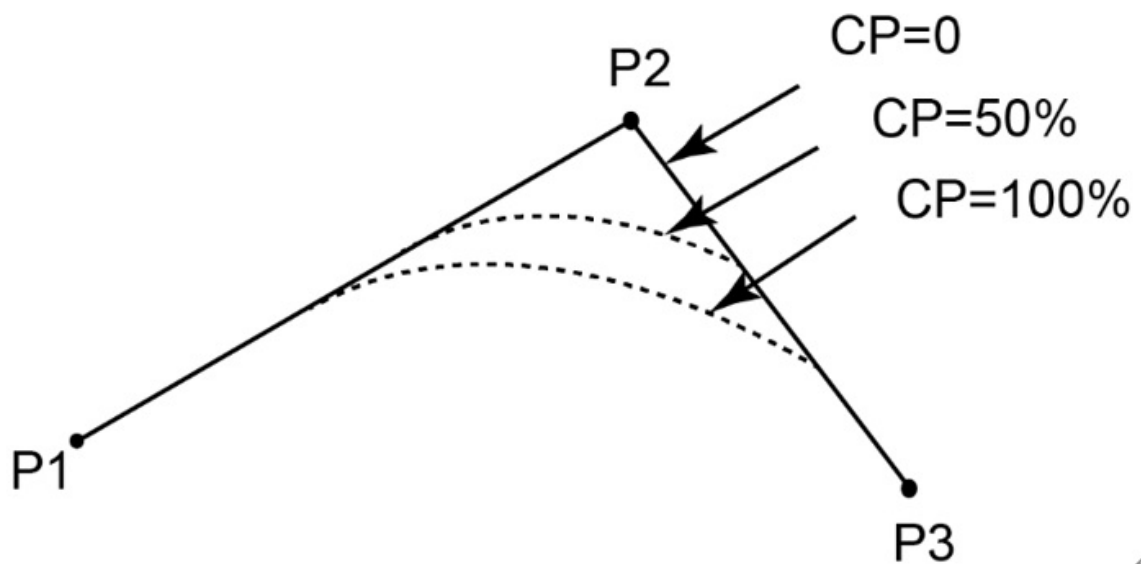
- 参数: R: 平滑过渡比例。取值范围: 0~100

“0”表示运动时不开启平滑过渡

- 示例

```
CP(50)  
Move(P1)  
Move(P2)  
Move(P3)
```

机械臂从P1运动至P3点经过P2点时以50%平滑过渡



设置Jump最大抬升高度

- 原型:

```
LimZ(zValue)
```

- 描述: 设置Jump模式最大抬升高度
- 参数: zValue: 最大抬升高度, 不能超过机械臂Z轴极限位置
- 示例

```
LimZ(80)  
Jump(P," Start=10 Zlimit=LimZ End=50")
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:23

设置全局速度比例

- 原型:

```
SpeedFactor(ratio)
```

- 描述: 设置全局速度比例
- 参数: ratio: 运动速度比例, 取值范围: 0~100, 但不包括0和100
- 返回: 无
- 示例

```
SpeedFactor (20)
```

设置全局速度比例为20%

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:26

设置Move、Jump、Arc3、Circle3、MoveR姿态运动速度比例

- 原型:

```
SpeedR(ratio)
```

- 描述: 设置Move、Jump、Arc3、Circle3、MoveR姿态运动速度比例
- 参数: Ratio: 比例值, 范围是0~100之间, 不包括0和100
- 返回: 无
- 示例

```
SpeedR (20)  
Move(P1)
```

表示机械臂以20%的速度比例运动至P1点

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:26

设置Move、Jump、Arc3、Circle3、MoveR姿态运动加速度比例

- 原型:

```
AccelR(ratio)
```

- 描述: 设置Move、Jump、Arc3、Circle3、MoveR姿态运动加速度比例
- 参数: Ratio: 比例值, 范围是0~100之间, 不包括0和100
- 返回: 无
- 示例

```
AccelR (20)  
Move(P1)
```

表示机械臂以20%的加速度比例运动至P1点

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:26

六维传感器运动指令

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06
18:06:26

以下为例

六维传感器回零

- 原型:

```
SixForceHome()
```

- 描述:对六维传感器进行回零操作
- 参数:无
- 示例

```
SixForceHome()
```

运行该指令回零六维传感器

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-11 16:37:20

螺旋运动寻找孔位

- 原型:

```
Spiral(P, User, Tool, Direction, SpeedC, Force, Insertion, Perturn, PeckMode, MaxValue)
```

- 描述:机械臂在当前位置和指定位置之间做螺旋运动寻找孔位,该指定点要求离孔位距离较近,是孔位探索的起始点

- 参数:

- o P:指定的点位
- o Direction: 选定的插孔方向(0:正向;1:反向)
- o SpeedC: 插孔的速度(mm/s)
- o Force: 螺旋阈值(N)
- o Insertion: 插孔阈值(N)
- o Perturn: 螺旋半径(mm)
- o PeckMode: 点接触模式(ON/OFF)
- o MaxValue: 最大螺旋半径(mm)

- 示例

```
Spiral(P1,"User=1 Tool=2 Dirction=0 SpeedC=5 Force =10 Insertion=3 Perturn=0.7 PeckMode=OFF MaxValue =5")
```

在当前位置和P1之间做螺旋运动寻找孔位。当插孔方向上的阻力大于Force阈值时,机器人做螺旋运动进行孔位探索,当插孔方向上的阻力小于Insertion阈值时,机器人沿着插孔方向运动,进行插孔工作。

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-11 16:37:20

旋转运动寻找孔位

- 原型:

```
Rotation (P, User, Tool, Direction, SpeedC, Force, RotationSpeed, MaxTorque, PeckMode, MaxValue)
```

- 描述:机械臂在当前位置和指定位置之间做旋转运动寻找孔位,该指定点要求离孔位距离较近,是孔位探索的起始点

- 参数:

- o P:指定的点位
- o User:表示用户坐标系,取值范围:0~9
- o Tool:表示工具坐标系,取值范围:0~9
- o Direction:选定的插孔方向(0:正向;1:反向)
- o SpeedC:插孔的速度(mm/s)
- o Force:旋转阈值(N)
- o RotationSpeed:旋转速度(°/s)
- o MaxTorque:最大力矩(Nm)
- o PeckMode:点接触模式(ON/OFF)
- o MaxValue:最大螺旋半径(mm)

- 示例

```
Rotation (P1, "User=1 Tool=2 Dirction=0 SpeedC =5 Force =10 RotationSpeed=5 MaxTorque=1 PeckMode=OFF MaxValue =45")
```

在当前位置和P1之间做旋转运动进行插孔。当插孔方向上的阻力大于Force阈值时,机器人做旋转运动进行孔位探索,当插孔方向上的阻力小于Force阈值时,机器人沿着插孔方向运动,进行插孔工作。

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06 18:06:23

线性插孔运动

- 原型:

```
Linear (User, Tool, Direction, SpeedC, Force, MaxValue)
```

- 描述:机械臂在孔位方向做线性插孔运动

- 参数:

- User:表示用户坐标系,取值范围:0~9
- Tool:表示工具坐标系,取值范围:0~9
- Direction:选定的插孔方向(0:正向;1:反向)
- SpeedC:插孔的速度(mm/s)
- Force:螺旋阈值(N)
- MaxValue:最大螺旋半径(mm)

- 示例

```
Linear("User=1 Tool=2 Dircction=0 SpeedC =5 Force=10 MaxValue=45")
```

在当前孔位做线性插孔运动,当插入方向的阻力大于Force阈值,则认为插入完成。

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06 18:06:26

输入输出指令

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:23

DI

- 原型:

```
DI(index)
```

- 描述: 读取输入端口状态

- 参数:

- index: 数字输入索引, 取值范围: 1~32

- 返回:

- 当设置了索引时, DI(index)返回对应的index的电平 (ON/OFF)
- 当没有参数时, 返回所有index的电平并以table形式存储

假如local di=DI(), table存储形式为{num = 24 value = {0x55, 0xAA, 0x52}},则可通过di.num, di.value[n]等来获取对应index的值

- 示例

```
if (DI(1))==ON then  
  Move(P1)  
end
```

数字输入端口1为高电平时机械臂以直线运动方式运动至P1点

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06

18:06:24

DO(队列)

- 原型:

```
DO(index,ON | OFF)
```

- 描述:设置数字输出端口状态(队列指令)
- 参数:
 - index:数字输出索引,取值范围:1~24
 - ON/OFF:数字输出端口状态, ON:高电平;OFF:低电平
- 示例

```
DO(1,ON)
```

将数字输出端口1设置为高电平

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:26

DOExecute(立即)

- 原型:

```
DOExecute(index,ON | OFF)
```

- 描述:设置数字输出端口状态(立即指令)
- 参数:
 - index:数字输出索引,取值范围:1~16
 - ON/OFF:数字输出端口状态, ON:高电平;OFF:低电平
- 示例

```
DOExecute(1,OFF)
```

将数字输出端口1设置为低电平

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06 18:06:24

ToolDI

- 原型:

```
ToolDI(index)
```

- 描述: 读取末端输入端口状态
- 参数:
 - index: 数字输入索引, 取值范围: 1或2
- 返回: 返回对应的index的电平, 类型为number, 返回值为0或1, 分别代表低电平和高电平

假如local di=DI(), table存储形式为{num = 24 value = {0x55, 0xAA, 0x52}},则可通过di.num, di.value[n]等来获取对应index的值

- 示例

```
if (ToolDI (1))==1 then  
Move(P1)  
end
```

末端数字输入端口1为高电平时机械臂以直线运动方式运动至P1点

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:24

ToolDO

- 原型:

```
ToolDO(index, ON | OFF)
```

- 描述: 设置末端数字输出端口状态(算法指令)

- 参数:

- index: 末端数字输出索引, 取值范围: 1或2
- ON/OFF: 末端数字输出端口状态, ON: 高电平; OFF: 低电平

- 返回: 无

- 示例

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

将末端数字输出端口1设置为低电平

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:26

ToolDOExecute(立即)

- 原型:

```
ToolDOExecute(index, ON | OFF)
```

- 描述: 设置末端数字输出端口状态(立即指令)
- 参数:
 - index: 末端数字输出索引, 取值范围: 1或2
 - ON/OFF: 末端数字输出端口状态, ON: 高电平; OFF: 低电平
- 返回: 无
- 示例

```
ToolDOExecute (1,OFF)
```

将末端数字输出端口1设置为低电平

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:26

ToolAnalogMode(立即)

- 原型:

```
ToolAnalogMode(mode)
```

- 描述: 设置末端模拟输入端口的模式(立即指令)

- 参数:

- mode: 模拟输入端口的模式
 - 00: 默认, 485模式
 - 10: 电流采集模式
 - 11: 0~3.3V 电压输入模式
 - 12: 0~10V 电压输入模式

- 返回: 无

- 示例

```
ToolAnalogMode(11)
```

设置末端模拟输入端口的模式为电压输入模式

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-17 12:41:49

ToolAI

- 原型:

```
ToolAI(index)
```

- 描述: 读取末端模拟输入端口电压

- 参数:

- index: 数字输入索引, 取值范围: 1或2 注意: 使用之前需要先设置模式, 详见ToolAnalogMode; 另外, 模拟输入端口和末端485通讯端口存在复用

- 返回: 返回对应index的电压值

- 示例

```
Voltage = ToolAI (1)
```

获得机械臂控制柜上的输入端口1的电压值

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:24

AI

- 原型:

```
AI(index)
```

- 描述: 读取控制柜模拟输入端口的电压值(立即指令)
- 参数:
 - index: 控制柜模拟输入索引, 取值范围: 1或2
- 返回: 返回对应index的电压值
- 示例

```
Voltage = AI(1)
```

获得机械臂控制柜上的模拟输入端口1的电压值

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:24

AO

- 原型:

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

- 描述:设置控制柜模拟输出端口的电压值

- 参数:

- index:控制柜模拟输出索引, 取值范围:1或2
- value:对应index的电压值, 取值范围0~10, 类型:number

- 返回:无

- 示例

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

设置控制柜模拟输出端口1的电压值为2V

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06 18:06:26

AOExecute (立即)

- 原型:

```
AOExecute(index,value)
```

- 描述: 设置控制柜模拟输出端口的电压值(立即指令)

- 参数:

- index: 控制柜模拟输出索引, 取值范围: 1或2
- value: 对应index的电压值, 取值范围0~10, 类型: number

- 返回:

- index: 控制柜模拟输出索引, 取值范围: 1或2
- value: 对应index的电压值, 取值范围0~10, 类型: number

- 示例

```
AOExecute (1,2)
```

设置控制柜模拟输出端口1的电压值为2V

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:26

WaitDI

- 原型:

```
WaitDI(index, ON | OFF, period)
```

- 描述: 读取数字输入端口状态, 若读取到的状态与指定状态一致, 程序才往下执行, 否则, 间隔指定周期读取数字输入端口状态

- 参数:

- index: 数字输入索引, 取值范围: 1~32
- ON | OFF: 数字输入端口状态, ON: 高电平; OFF: 低电平
- period: 读取周期, 默认50ms

- 返回: 无

- 示例

```
WaitDI(1, ON)  
Move(P1)
```

每50ms读取数字输入端口1的状态, 如果数字输入端口1为高电平时, 程序才往下执行, 机械臂以直线运动方式运动至P1点

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-12 14:06:35

程序管理指令

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:24

运动指令等待

- 原型:

```
Wait(time)
```

- 描述: 运动暂停时间
- 参数:
 - time: 延时时间。单位: 毫秒
- 示例

```
Go(P1)  
Wait(1000)
```

机械臂执行至P1点后延时1000毫秒

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:24

脚本暂停时间

- 原型:

```
Sleep(time)
```

- 描述:通用脚本暂停时间
- 参数:
 - time:延时时间。单位:毫秒
- 示例




```
while true do
  Speed(100)
  Go(P1)
  sleep(3)
  Speed(100)
  Accel(40)
  Go(P2)
  sleep(3)
end
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:26

暂停程序运行

- 原型:

```
Pause()
```

- 描述: 程序运行暂停指令 当程序运行至该指令后, 程序暂停运行, 此时程序运行页面中图标由  暂停 变为  恢复。若需继续运行, 点击  恢复即可

- 参数: 无

- 示例

```
while true
do
Go(P1)
Go(P2)
Pause()
Go(P3)
Go(P4)
end
```

机械臂执行至P2点后暂停运行

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:26

开始计时

- 原型:

```
ResetElapsedTime()
```

- 描述:待此指令前所有指令执行完成后开始计时,需结合ElapsedTime()一起使用 如计算脚本中某段代码执行完成所耗时间
- 参数:无
- 返回:无
- 示例

```
Go(P2, " Speed=100 Accel=100")
ResetElapsedTime()
for i=1,10 do
Jump(P1, " Speed=100 Accel=100 Start=0 End=0 ZLimit=185")
Jump(P2, " Speed=100 Accel=100 Start=0 End=0 ZLimit=185")
end
print (ElapsedTime())
Sleep(1000)
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:26

结束计时

- 原型:

```
ElapsedTime()
```

- 描述:结束计时, 返回时间差。需结合ResetElapsedTime()一起使用
- 参数:无
- 返回:返回时间差, 单位:毫秒
- 示例

```
Go(P2, " Speed=100 Accel=100")
ResetElapsedTime()
for i=1,10 do
  Jump(P1, " Speed=100 Accel=100 Start=0 End=0 ZLimit=185")
  Jump(P2, " Speed=100 Accel=100 Start=0 End=0 ZLimit=185")
end
print (ElapsedTime())
Sleep(1000)
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06 18:06:26

获取系统时间

- 原型:

```
Systime()
```

- 描述: 获取当前时间
- 参数: 无
- 返回: 返回当前时间
- 示例

```
Go(P2, " Speed=100 Accel=100")
local time1=Systime()
for i=1,10 do
  Jump(P1, " Speed=100 Accel=100 Start=0 End=0 ZLimit=185")
  Jump(P2, " Speed=100 Accel=100 Start=0 End=0 ZLimit=185")
end
local time2=Systime()
local time = time2 - time1
Sleep(1000)
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:23

打印

- 原型:

```
print(value)
```

- 描述: 将调试信息等打印出来
- 参数: value: 待打印的数据, 目前支持的数据类型: table、number、string和bool
- 返回: 无
- 示例

```
print("hello world")
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:24

获取位姿指令

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:26

获取笛卡尔坐标

- 原型:

```
GetPose()
```

- 描述: 获取笛卡尔坐标系下机械臂的实时位姿。如果设置了用户坐标系或工具坐标系, 则获取的位姿为当前坐标系下的位姿
- 参数: 无
- 返回: 机械臂当前位置的笛卡尔坐标值
- 示例

```
local currentPose = GetPose()
-- 获取当前位置
local liftPose = {coordinate = {currentPose.coordinate[1], currentPose.coordinate[2], currentPose.coordinate[3], currentPose.coordinate[4]}, tool = currentPose.tool, user = currentPose.user}
-- 原地抬升
Go(liftPose, "Speed=100 Accel=100")
Go(P1)
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:26

获取关节坐标

- 原型:

```
GetAngle()
```

- 描述: 获取关节坐标系下机械臂的实时位姿
- 参数: 无
- 返回: 机械臂当前位置的关节坐标值
- 示例

```
local armPose
local joint = GetAngle()
-- 获取当前位姿
local liftPose = { joint = {joint.joint[1], joint.joint[2], joint.joint[3], joint.joint[4]}, tool = 0, user = 0}
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:26

TCP指令

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:26

创建TCP

- 原型:

```
err, socket = TCPCreate(isServer, IP, port)
```

- 描述: 创建TCP网络, 仅支持单连接

- 参数:

- isServer: 是否创建服务器。0: 表示创建客户端; 1: 表示创建服务端
- IP: 服务端IP地址, 需与客户端IP地址在同一网段, 且不冲突
- port: 服务端端口

机械臂作为服务端时, “port”不能设置为502或8080, 否则会与Modbus默认端口或流水线跟踪中使用的端口冲突, 导致创建TCP网络失败

- 返回:

- err: 0: 创建TCP网络成功 1: 创建TCP网络失败
- socket: 创建的socket对象

- 示例1: TCP服务端

```
local ip="192.168.5.1"           //机械臂的IP地址作为服务端的IP地址
local port=6001                  //服务端端口
local err=0
local socket=0
err, socket = TCPCreate(true, ip, port)
```

- 示例2: TCP客户端

```
local ip="192.168.5.25"          //外部设备如相机的IP地址作为服务端的IP地址
local port=6001                  //服务端端口
local err=0
local socket=0
err, socket = TCPCreate(false, ip, port)
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06
18:06:26

建立TCP连接

- 原型:

```
TCPStart(socket, timeout)
```

- 描述: TCP连接功能
- 参数:
 - socket: socket对象
 - timeout: 等待超时时间。单位: 秒。如果为0, 则一直等待连接。如果不为0, 则超过设定的时间后退出连接
- 返回:
 - 0: TCP网络连接成功
 - 1: 输入参数错误
 - 2: socket对象不存在
 - 3: 设置超时时间错误
 - 4: 若机械臂作为客户端, 则说明连接错误; 若机械臂作为服务端, 则说明接收错误
- 示例1: TCP服务端

```
local ip="192.168.5.1"                //机械臂的IP地址作为服务端的IP地址
local port=6001                       //服务端端口
local err=0
local socket=0
err, socket = TCPCreate(true, ip, port)
if err == 0 then
    err = TCPStart(socket, 0)
```

- 示例2: TCP客户端

```
local ip="192.168.5.25"               //外部设备如相机的IP地址作为服务端的IP地址
local port=6001                       //服务端端口
local err=0
local socket=0
err, socket = TCPCreate(false, ip, port)
if err == 0 then
    err = TCPStart(socket, 0)
```

接收TCP数据

- 原型:

```
err, Recbuf = TCPRead(socket, timeout, type)
```

- 描述:

- 机械臂作为客户端接收来自服务端的数据
- 机械臂作为服务端时接收来自客户端的数据

- 参数:

- socket: socket对象
- timeout: 接收超时时间, 单位: 秒。若timeout为0或者不设置, 则该指令为阻塞式读取, 即接收完数据后程序才往下执行; 若不为0, 则超过设定的timeout后, 程序继续往下执行, 即不考虑数据是否接收完, 继续往下执行
- type: 缓存类型。若设置为空, 则RecBuf缓存格式为table形式, 如果设置为“string”, 则RecBuf缓存字符串形式

- 返回:

- err: 0: 接收数据成功 1: 接收数据失败
- Recbuf: 接收数据缓存区

- 示例1: TCP服务端

```
local ip="192.168.5.1"                //机械臂的IP地址作为服务端的IP地址
local port=6001                      //服务端端口
local err=0
local socket=0
err, socket = TCPCreate(true, ip, port)
if err == 0 then
err = TCPStart(socket, 0)
if err == 0 then
local RecBuf
while true do
TCPWrite(socket, "tcp server test")    //服务端发送数据至客户端
err, RecBuf = TCPRead(socket,0,"string") //服务端接收客户端发送的数据
```

- 示例2: TCP客户端

```
local ip="192.168.5.25"              //外部设备如相机的IP地址作为服务端的IP地址
local port=6001                      //服务端端口
local err=0
local socket=0
err, socket = TCPCreate(false, ip, port)
if err == 0 then
err = TCPStart(socket, 0)
if err == 0 then
local RecBuf
while true do
TCPWrite(socket, "tcp client test")    //客户端发送数据至服务端
TCPWrite(socket, {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07})
err, RecBuf = TCPRead(socket, 0)      //客户端接收服务端发送的数据
```


发送TCP数据

- 原型:

```
TCPWrite(socket, buf, timeout)
```

- 描述:

- 机械臂作为客户端时发送数据给服务端
- 机械臂作为服务端时发送数据给客户端

- 参数:

- socket: socket对象
- buf: 发送的数据
- timeout: 超时时间, 单位: 秒。若timeout为0或者不设置, 则该指令为阻塞式读取, 即发送完数据后程序才往下执行; 若不为0, 则超过设定的timeout后, 程序继续往下执行, 即不考虑数据是否接收完, 继续往下执行

- 返回:

- 0: 发送数据成功
- 1: 发送数据失败

- 示例1: TCP服务端

```
local ip="192.168.5.1"                //机械臂的IP地址作为服务端的IP地址
local port=6001                      //服务端端口
local err=0
local socket=0
err, socket = TCPCreate(true, ip, port)
if err == 0 then
    err = TCPStart(socket, 0)
if err == 0 then
    local RecBuf
    while true do
        TCPWrite(socket, "tcp server test")    //服务端发送数据至客户端
    end
end
end
end
```

- 示例2: TCP客户端

```
local ip="192.168.5.25"              //外部设备如相机的IP地址作为服务端的IP地址
local port=6001                      //服务端端口
local err=0
local socket=0
err, socket = TCPCreate(false, ip, port)
if err == 0 then
    err = TCPStart(socket, 0)
if err == 0 then
    local RecBuf
    while true do
        TCPWrite(socket, "tcp client test")    //客户端发送数据至服务端
        TCPWrite(socket, {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07})
    end
end
end
end
```

12:41:49

关闭TCP

- 原型:

```
TCPDestroy(socket)
```

- 描述:关闭TCP功能

- 参数:

- socket:socket对象

- 返回:

- 0:关闭TCP成功
- 1:关闭TCP失败

- 示例1:TCP服务端

```
local ip="192.168.5.1"                // IP address of the robot as a server
local port=6001                      // Server port
local err=0
local socket=0
err, socket = TCPCreate(true, ip, port)
if err == 0 then
err = TCPStart(socket, 0)
if err == 0 then
local RecBuf
while true do
TCPWrite(socket, "tcp server test")    // Server sends data to client
err, RecBuf = TCPRead(socket,0,"string") // Server receives the data from client
if err == 0 then
Go(P1)                                //Start to run motion commands after the server receives data
Go(P2)
print(buf)
else
print("Read error ".. err)
break
end
end
else
print("Create failed ".. err)
end
TCPDestroy(socket)
else
print("Create failed ".. err)
end
end
```

- 示例2:TCP客户端

```
local ip="192.168.5.25"              // External equipment such as a camera is set as the server
local port=6001                      // Server port
local err=0
local socket=0
err, socket = TCPCreate(false, ip, port)
if err == 0 then
err = TCPStart(socket, 0)
if err == 0 then
local RecBuf
```

```

while true do
  TCPWrite(socket, "tcp client test")           // Client sends data to server
  TCPWrite(socket, {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07})
  err, RecBuf = TCPRead(socket, 0)             // Client receives data from server
  if err == 0 then
    Go(P1)                                     // Start to run motion commands after the client receives the data
    Go(P2)
    print(buf)
  else
    print("Read error ".. err)
    break
  end
end
else
  print("Create failed ".. err)
end
TCPDestroy(socket)
else
  print("Create failed ".. err)
end
end

```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06
18:06:24

读取单点数据

- 原型:

```
err, Recbuf = TCPReadVision(socket, timeout, type)
```

- 描述: 读取单个点数据

- 参数:

- socket: socket对象
- timeout: 接收超时时间, 单位: 秒。若timeout为0或者不设置, 则该指令为阻塞式读取; 若不为0, 则超过设定的timeout后, 程序继续往下执行
- type: 缓存类型。若设置为空, 则RecBuf缓存格式为table形式, 如果设置为“string”, 则RecBuf缓存字符串形式

- 返回:

- err: 0: 接收数据成功 1: 接收数据失败
- Recbuf: 接收数据缓存区

- 示例

```
local err, RecBuf = TCPReadVision (socket, timeout, "string");
```

其中, RecBuf内容为{ coordinate={x, y, z, Rx, Ry, Rz}}

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-17 12:41:49

读取多点数据

- 原型:

```
err, Recbuf = TCPReadMultiVision(socket, timeout, type)
```

- 描述: 读取多个点数据

- 参数:

- socket: socket对象
- timeout: 接收超时时间, 单位: 秒。若timeout为0或者不设置, 则该指令为阻塞式读取; 若不为0, 则超过设定的timeout后, 程序继续往下执行
- type: 缓存类型。若设置为空, 则RecBuf缓存格式为table形式, 如果设置为“string”, 则RecBuf缓存字符串形式

- 返回:

- err: 0: 接收数据成功 1: 接收数据失败
- Recbuf: 接收数据缓存区

- 示例

```
local err, RecBuf = TCPReadMultiVision (socket, timeout, "string")
```

其中, RecBuf内容为{ coordinate0={x, y, z, Rx, Ry, Rz}, coordinate1={x, y, z, Rx, Ry, Rz}}

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-17 13:09:19

UDP指令

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:26

创建UDP

- 原型:

```
err, socket = UDPCreate(isServer, IP, port)
```

- 描述: 创建UDP网络, 仅支持单连接

- 参数:

- o isServer: 是否创建服务器。0: 表示创建客户端; 1: 表示创建服务端
- o IP: 服务端IP地址, 需与客户端IP地址在同一网段, 且不冲突
- o port: 服务端端口 机械臂作为服务端时, “port”不能设置为502或8080, 否则会与Modbus默认端口或流水线跟踪中使用的端口冲突, 导致创建UDP网络失败

- 返回:

- o err: 0: 创建UDP网络成功 1: 创建UDP网络失败
- o socket: 创建的socket对象

- 示例1: UDP服务端

```
local ip="192.168.5.1"                //机械臂的IP地址作为服务端的IP地址
local port=6201                       //服务端端口
local err=0
local socket=0
err, socket = UDPCreate(true, ip, port)
```

- 示例2: UDP客户端

```
local ip="192.168.1.25"               //外部设备的IP地址作为服务端的IP地址
local port=6200                       //服务端端口
local err=0
local socket=0
err, socket = UDPCreate(false, ip, port)
```

接收UDP数据

- 原型:

```
err, Recbuf = UDPRead(socket, timeout, type)
```

- 描述:

- 机械臂作为客户端接收来自服务端的数据
- 机械臂作为服务端时接收来自客户端的数据

- 参数:

- socket: socket对象
- timeout: 超时时间, 单位: 秒。若timeout为0或者不设置, 则该指令为阻塞式读取, 即接收完数据后程序才往下执行; 若不为0, 则超过设定的timeout后, 程序继续往下执行, 即不考虑数据是否接收完, 继续往下执行
- type: 缓存类型。若设置为空, 则RecBuf缓存格式为table形式, 如果设置为“string”, 则RecBuf缓存字符串形式

- 返回:

- err: 0: 接收数据成功 1: 接收数据失败
- Recbuf: 接收数据缓存区

- 示例1: UDP服务端

```
local ip="192.168.5.1"                //机械臂的IP地址作为服务端的IP地址
local port=6201                       //服务端端口
local err=0
local socket=0
err, socket = UDPCreate(true, ip, port)
if err == 0 then
    local RecBuf
    while true do
        UDPWrite(socket, "udp server test")    //服务端发送数据至客户端
        err, RecBuf = UDPRead(socket, 0)        //服务端接收客户端发送的数据
    end
end
```

- 示例2: UDP客户端

```
local ip="192.168.1.25"                //外部设备的IP地址作为服务端的IP地址
local port=6200                       //服务端端口
local err=0
local socket=0
err, socket = UDPCreate(false, ip, port)
if err == 0 then
    local buf
    while true do
        UDPWrite(socket, "udp client test")    //客户端发送数据至服务端
        UDPWrite(socket, {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07})
        err, RecBuf = UDPRead(socket, 0)        //客户端接收服务端发送的数据
    end
end
```


发送UDP数据

- 原型:

```
UDPWrite(socket, buf, timeout)
```

- 描述:

- 机械臂作为客户端时发送数据给服务端
- 机械臂作为服务端时发送数据给客户端

- 参数:

- socket: socket对象
- buf: 发送的数据
- timeout: 超时时间, 单位: 秒。若timeout为0或者不设置, 则该指令为阻塞式读取, 即发送完数据后程序才往下执行; 若不为0, 则超过设定的timeout后, 程序继续往下执行, 即不考虑数据是否发送完, 继续往下执行

- 返回:

- 0: 发送数据成功
- 1: 发送数据失败

- 示例1: UDP服务端

```
local ip="192.168.5.1"                //机械臂的IP地址作为服务端的IP地址
local port=6201                      //服务端端口
local err=0
local socket=0
err, socket = UDPCreate(true, ip, port)
if err == 0 then
    local RecBuf
    while true do
        UDPWrite(socket, "udp server test")    //服务端发送数据至客户端
```

- 示例2: UDP客户端

```
local ip="192.168.1.25"              //外部设备的IP地址作为服务端的IP地址
local port=6200                      //服务端端口
local err=0
local socket=0
err, socket = UDPCreate(false, ip, port)
if err == 0 then
    local buf
    while true do
        UDPWrite(socket, "udp client test")    //客户端发送数据至服务端
        UDPWrite(socket, {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07})
```

Modbus指令

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:26

创建Modbus主站

- 原型:

```
ModbusCreate()
```

- 描述: 创建Modbus主站, 并和从站建立连接

- 参数:

- IP: 从站IP地址
- port: 从站端口
- slave_id: 从站ID

- 返回:

- err:
 - 0: 创建Modbus主站成功
 - 1: 创建Modbus主站失败
- id: 返回的从站设备号, 最多支持5个设备。取值范围: 0~4

注意: 当ip, port, slave_id为空, 或者ip为“127.0.0.1”或“0.0.0.1”的时候, 就连接本机的Modbus从站; 例如用户输入如下ModbusCreate接口任意一个指令, 则表示连接本机的Modbus从站;

- ModbusCreate()
- ModbusCreate(“127.0.0.1”)
- ModbusCreate(“0.0.0.1”)
- ModbusCreate(“127.0.0.1”,xxx,xxx) //xxx任意值
- ModbusCreate(“0.0.0.1”,xxx,xxx) //xxx任意值

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-17 12:41:49

9.2 断开连接

- 原型:

```
ModbusClose()
```

- 描述: 和Modbus从站断开连接
- 参数:
 - id: 返回的从站设备号, 最多支持5个设备。取值范围: 0~4
- 返回:
 - 0: 关闭Modbus成功
 - 1: 关闭Modbus失败
- 示例

```
err, id = ModbusCreate(ip, port, slave_id)
if err == 0 then
    coils = {0, 1, 1, 1, 0}
    SetCoils(id, 1024, #coils, coils)
    ModbusClose(id)
else
    print("Create failed:", err)
end
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-12 19:56:57

读取Modbus从站线圈寄存器地址的值

- 原型:

```
GetCoils(id, addr, count)
```

- 描述: 读取Modbus从站线圈寄存器地址的值
- 参数:
 - id: 从站设备号, 最多支持5个设备。取值范围: 0~4
 - addr: 线圈寄存器起始地址。取值范围: 0~4095
 - count: 连续读取线圈寄存器的地址个数。取值范围: 0~ 4096 - addr
- 返回: 线圈寄存器地址的值, 存储在table中。table中第一个值对应线圈寄存器起始地址的值, 返回的数据类型: 位
- 示例

```
从地址0开始连续读取5个线圈的值
Coils = GetCoils(id,0,5)
返回结果:
Coils={1,0,0,0,0}
结合下表, 表示机械臂处于启动状态
```

线圈寄存器地址 (PLC为例)	线圈寄存器地址 (CR系列机器人)	数据类型	说明
00001	0	位	启动
00002	1	位	暂停
00003	2	位	继续
00004	3	位	停止
00005	4	位	急停
00006	5	位	清除报警
00007~0999	6~998	位	系统预留
01001~04096	999~4095	位	用户自定义

设置Modbus从站线圈寄存器地址的值

- 原型:

```
SetCoils(id, addr, count, table)
```

- 描述: 设置Modbus从站线圈寄存器地址的值 对于Modbus从站线圈寄存器地址0~5, 该指令不可用
- 参数:
 - id: 从站设备号, 最多支持5个设备。取值范围: 0~4
 - addr: 线圈寄存器起始地址。取值范围: 6~4095
 - count: 连续写入线圈寄存器的地址个数。取值范围: 0~ 4096 - addr
- table: 线圈寄存器地址的值。数据类型: 位
- 返回: 无
- 示例

```
local Coils = {0,1,1,1,0}  
SetCoils(id, 1024, #coils, Coils)
```

从地址1024开始, 连续写5个线圈

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-11 16:37:20

读取Modbus从站触点寄存器地址的值

- 原型:

```
GetInBits(id, addr, count)
```

- 描述: 读取Modbus从站触点寄存器地址的值
- 参数:
 - id: 从站设备号, 最多支持5个设备。取值范围: 0~4
- addr: 触点寄存器起始地址地址。取值范围: 0 ~ 4095
 - count: 连续读取触点寄存器的地址个数。取值范围: 0 ~ 4096-addr
- 返回: 触点寄存器地址的值, 存储在table中。table中第一个值对应线圈寄存器起始地址的值 返回的数据类型: 位
- 示例

```
从地址0开始连续读取5个地址的值
inBits = GetInBits(id,0,5)
返回结果:
inBits = {0,0,0,1,0}
结合下表, 表示机械臂处于运行状态
```

线圈寄存器地址(PLC为例)	线圈寄存器地址(CR系列机器人)	数据类型	说明
00001	0	位	启动
00002	1	位	暂停
00003	2	位	继续
00004	3	位	停止
00005	4	位	急停
00006	5	位	清除报警
00007~0999	6~998	位	系统预留
01001~04096	999~4095	位	用户自定义

读取Modbus从站输入寄存器地址的值

- 原型:

```
GetInRegs(id, addr, count, type)
```

- 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
- 参数:
 - id:从站设备号, 最多支持5个设备。取值范围:0~4
 - addr:输入寄存器起始地址。取值范围:0 ~ 4095
 - count:读取输入寄存器的地址个数。取值范围:0 ~4096-addr
 - type:数据类型
 - 如果为空, 默认读取16位无符号整数(2个字节, 占用1个寄存器)
 - “U16”:读取16位无符号整数(2个字节, 占用1个寄存器)
 - “U32”:读取32位无符号整数(4个字节, 占用2个寄存器)
 - “F32”:读取32位单浮点数(4个字节, 占用2个寄存器)
 - “F64”:读取64位双精度浮点数(8个字节, 占用4个寄存器)
- 返回:输入寄存器地址的值, 存储在table中。table中第一个值对应线圈寄存器起始地址的值
- 示例1

```
data = GetInRegs(id,2048,1)
```

从地址2048开始读取一个16位无符号整数

- 示例2

```
data = GetInRegs(id, 2048, 1, "U32")
```

从地址2048开始读取一个32位无符号整数

读取Modbus从站保持寄存器地址的值

- 原型:

```
GetHoldRegs(id, addr, count, type)
```

- 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
- 参数:
 - id:从站设备号, 最多支持5个设备。取值范围:0~4
 - addr:保持寄存器的起始地址。取值范围:0 ~ 4095
 - count:读取保持寄存器的地址个数。取值范围:0 ~4096-addr
 - type:数据类型
 - 如果为空, 默认读取16位无符号整数(2个字节, 占用1个寄存器)
 - “U16”:读取16位无符号整数(2个字节, 占用1个寄存器)
 - “U32”:读取32位无符号整数(4个字节, 占用2个寄存器)
 - “F32”:读取32位单精度浮点数(4个字节, 占用2个寄存器)
 - “F64”:读取64位双精度浮点数(8个字节, 占用4个寄存器)
- 返回:保持寄存器地址的值, 存储在table中。table中第一个值对应线圈寄存器起始地址的值
- 示例1

```
data = GetHoldRegs(id,2048,1)
```

从地址2048开始读取一个16位无符号整数

- 示例2

```
data = GetInRegs(id, 2048, 1, "U32")
```

从地址2048开始读取一个32位无符号整数

设置Modbus从站保持寄存器地址的值

- 原型:

```
SetHoldRegs(id, addr, count, table, type)
```

- 描述: 设置保持寄存器地址的值

- 参数:

- o id: 从站设备号, 最多支持5个设备。取值范围: 0~4
- o addr: 保持寄存器的起始地址。取值范围: 0 ~ 4095
- o count: 写入保持寄存器的地址个数。取值范围: 0 ~ 4096-addr
- o table: 保持寄存器地址的值
- o type: 数据类型
 - 如果为空, 默认写入16位无符号整数(2个字节, 占用1个寄存器)
 - “U16”: 写入16位无符号整数(2个字节, 占用1个寄存器)
 - “U32”: 写入32位无符号整数(4个字节, 占用2个寄存器)
 - “F32”: 写入32位单精度浮点数(4个字节, 占用2个寄存器)
 - “F64”: 写入64位双精度浮点数(8个字节, 占用4个寄存器)

- 返回: 无

- 示例1

```
local data = {6000}  
SetHoldRegs(id, 2048, #data, data, "U16")
```

从地址2048开始, 写入一个16位无符号整数

- 示例2

```
local data = {95.32105}  
SetHoldRegs(id, 2048, #data, data, "F64")
```

从地址2048开始, 写入一个64位双精度浮点数

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-11 16:37:20

末端工具指令

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:26

设置末端工具供电状态

- 原型:

```
SetToolPower(status)
```

- 描述: 设置末端工具供电状态
- 参数: Status: 末端工具供电状态, 0: 关闭电源; 1: 打开电源
- 示例

```
SetToolPower(0)
```

关闭末端工具的电源

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:26

设置末端工具波特率

- 原型:

```
SetToolBaudRate(baud)
```

- 描述: 设置末端工具的RS485接口对应的波特率
- 参数: baud: RS485接口的波特率
- 返回: 无
- 示例

```
SetToolBaudRate(115200)
```

将末端工具的RS485接口对应的波特率设置为115200Hz

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:24

坐标系

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:23

修改用户坐标系

- 原型:

```
SetUser(index, table)
```

- 描述: 修改用户坐标系
- 参数: 坐标系序号, 取值范围: 0~9, 其中, 0 为默认的坐标系; table: 坐标转换矩阵, 通常为 CalcUser() 的返回值
- 返回: 无
- 示例

```
Go(P1)
local u = {}
u = CalcUser(2, 0, {10, 10, 0, 0, 0, 0}) -- 用户坐标系2左乘{x, y, z, rx, ry, rz}, 当第二个参数为1时则右乘
SetUser(2, u) -- 修改用户坐标系2
Go(P1, "User=2") -- 使用修改后的用户坐标系2
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:26

计算用户坐标系

- 原型:

```
CalcUser(index,matrix_direction,table)
```

- 描述:计算用户坐标系

- 参数:

- Index:坐标系序号, 取值范围:0~9, 其中, 0为默认的坐标系
- matrix_direction:计算的方向
 - 0:index对应的用户坐标系左乘{x,y,z,rx,ry,rz}
 - 1:index对应的用户坐标系右乘{x,y,z,rx,ry,rz}
- Table:{x,y,z,rx,ry,rz}坐标转换矩阵

- 返回:计算后的坐标转换矩阵

- 示例

```
Go(P1)
local u = {}
u = CalcUser(2,0,{10,10,0,0,0,0}) --用户坐标系2左乘{x,y,z,rx,ry,rz},当第二个参数为1时则右乘
    SetUser(2, u)    --修改用户坐标系2
Go(P1,"User=2") --使用修改后的用户坐标系2
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:24

修改工具坐标系

- 原型:

```
SetTool(index1,table)
```

- 描述:修改工具坐标系
- 参数:
 - index1:坐标系序号, 取值范围:0~9, 其中, 0为默认的坐标系
 - Table:坐标转换矩阵, 通常为CalcTool()的返回值
- 返回:无
- 示例

```
Go(P1)
local u = {}
u = CalcTool(2,0,{10,10,0,0,0,0}) --工具坐标系2左乘{x,y,z,rx,ry,rz},当第二个参数为1时则右乘
SetTool(2, u) --修改工具坐标系2
Go(P1,"Tool=2") --使用修改后的工具坐标系2
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06 18:06:23

计算工具坐标系

- 原型:

```
CalcTool(index,matrix_direction,table)
```

- 描述: 计算工具坐标系
- 参数:
 - Index: 坐标系序号, 取值范围: 0~10, 其中, 0 为默认的坐标系
 - matrix_direction: 计算的方向。
 - 0: index对应的工具坐标系左乘{x,y,z,rx,ry,rz}
 - 1: index对应的工具坐标系右乘{x,y,z,rx,ry,rz}
 - Table: {x,y,z,rx,ry,rz}坐标转换矩阵
- 返回: 计算后的坐标转换矩阵
- 示例

```
Go(P1)
local u = {}
u = CalcTool(2,0,{10,10,0,0,0,0}) --工具坐标系2左乘{x,y,z,rx,ry,rz},当第二个参数为1时则右乘
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06 18:06:23

编码器

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:26

设置编码器当前值

- 原型:

```
SetABZPPC(value)
```

- 描述: 设置编码器的当前值
- 参数: value: 编码器的位置, 当前编码器的计数器为32位
- 返回: 无
- 示例

```
SetABZPPC(3000)
```

设置编码器的当前值为3000

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:24

获取编码器当前位置

- 原型:

```
GetABZ()
```

- 描述: 获取编码器当前位置
- 参数: 无
- 返回: 编码器当前位置
- 示例

```
local value=GetABZ()  
printf(value)
```

获取编码器当前位置并打印

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:24

轨迹复现指令

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:26

轨迹拟合

- 原型:

```
StartTrace(string, Option)
```

- 描述: 轨迹拟合

- 参数:

- String: 轨迹文件名(含后缀)
- option: 轨迹复现设置项
 - CP: 运动时设置平滑过渡, 取值范围: 0~100
 - SpeedS: 运动速度比例, 取值范围: 1~100
 - AccelS: 运动加速度比例, 取值范围: 1~100
 - SYNC: 同步标识, 取值范围: 0或1。SYNC = 0表示异步执行, 调用后立即返回, 但不关注指令执行情况; SYNC = 1表示同步执行, 调用后, 待指令执行完才返回

- 返回: 无

- 示例

```
local string="demo.json"  
StartTrace(string, "CP=10 SpeedS=50 AccelS=20 SYNC=1")
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-13 16:52:48

获取轨迹首个点位(轨迹拟合)

- 原型:

```
GetTraceStartPose(string)
```

- 描述: 获取轨迹中首个点位
- 参数: string: 轨迹文件名(含后缀)
- 返回: 点位坐标值
- 示例

```
local P1 = GetTraceStartPose(string)
```

获取轨迹中首个点位, 并赋值给P1

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-13 16:55:45

轨迹复现

- 原型:

```
StartPath(string, option)
```

- 描述: 轨迹复现

- 参数:

- String: 轨迹文件名(含后缀)
- option: 轨迹复现设置项
 - Multi: 速度倍数, 范围: 0.01~2.0
 - isConst: isConst=1时, 匀速复现, 轨迹中的停顿及死区会被移除
 - isCart: isCart取值为0或1(现在默认使用0)
 - 0: 表示运行关节点位
 - 1: 表示运行笛卡尔点位
 - isRev: isRev=1时, 轨迹反转
 - isRel: isRel=1时, 轨迹以当前点为起点, 并做整体轨迹偏移

- 返回: 无

- 示例

```
local string="demo.json"  
StartPath(string, "Multi=1 isConst=1 isCart=1 isRev=1 isRel=1")
```

以1倍速度, 按笛卡尔路径匀速复现, 同时轨迹反转, 整体轨迹偏移

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-13 16:54:12

获取轨迹首个点位(轨迹复现)

- 原型:

```
GetPathStartPose(string)
```

- 描述: 获取轨迹中首个点位
- 参数: string: 轨迹文件名(含后缀)
- 返回: 点位坐标值
- 示例

```
local P1 = GetPathStartPose(string)
```

获取轨迹中首个点位, 并赋值给P1

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-13 16:54:52

负载

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:23

设置当前负载

- 原型:

```
LoadSet(weight,inertia)
```

- 描述: 设置当前的负载
- 参数:
 - weight: 负载重量 kg
 - inertia: 负载惯量 kgm^2
- 返回: 无
- 示例

```
LoadSet(3,0.4)
```

设置机械臂的负载重量为3kg, 惯量为0.4 kgm^2

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:26

控制负载设置状态

- 原型:

```
LoadSwitch(status)
```

- 描述:控制负载设置状态

- 参数:

- status:

- 0:关闭负载参数设置
 - 1:开启负载参数设置, 开启负载设置会提高碰撞灵敏度

- 返回:无

- 示例

```
LoadSwitch(1)
```

开启负载参数设置

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:26

码垛

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:26

实例化矩阵跺盘变量

- 原型:

```
Pallet = MatrixPallet (index, "IsUnstack= true Userframe= 1")
```

- 描述:实例化矩阵跺盘变量
- 参数:
 - index:创建的矩阵跺型编号
- 可选参数:
 - IsUnstack:跺模式。取值范围:true或false。true:拆跺模式;false:码垛模式。若不设置,默认为码垛模式
 - Userframe:使用的用户坐标系序号。若不设置,默认为0
- 返回:实例化的跺盘变量
- 示例

```
myPallet = MatrixPallet(0, "IsUnstack=true Userframe=8")
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-09 17:59:00

设置下次操作踩的序号

- 原型:

```
SetPartIndex (Pallet, index)
```

- 描述: 设置下次操作踩的序号
- 参数:
 - Pallet: 实例化的踩盘变量
 - index: 下次操作踩的序号。起始值: 0
- 返回: 无
- 示例

```
local myPallet = MatrixPallet(0, "IsUnstack=true Userframe=8")  
SetPartIndex(myPallet, 1)
```

表示下次操作踩的序号为2

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-17 12:41:49

获取当前操作踩的序号

- 原型:

```
GetPartIndex (Pallet)
```

- 描述: 获取当前操作踩序号
- 参数:
 - Pallet: 实例化的踩盘变量
- 返回: 当前操作踩的序号
- 示例

```
local index=GetPartIndex(myPallet)
```

假设返回值为1, 表示当前操作踩的序号为2

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:26

设置下次操作码垛层的序号

- 原型:

```
SetLayerIndex (Pallet, index)
```

- 描述: 设置下次操作码垛层的序号
- 参数:
 - Pallet: 实例化的垛盘变量
 - index: 下次操作码垛层的序号。起始值: 0
- 返回: 无
- 示例

```
local myPallet = MatrixPallet(0, "IsUnstack=true Userframe=8")  
SetLayerIndex(myPallet, 1)
```

表示下次操作的码垛层为第二层

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-17 12:41:49

获取当前操作层序号

- 原型:

```
GetLayerIndex (Pallet)
```

- 描述: 获取当前操作层序号
- 参数:
 - Pallet: 实例化的跺盘变量
- 返回: 当前操作层的序号
- 示例

```
local index=GetLayerIndex(myPallet)
```

假设返回值为1, 表示当前操作码垛层为第二层

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:24

复位码垛

- 原型:

```
Reset (Pallet)
```

- 描述: 复位, 恢复初始化状态
- 参数:
 - Pallet: 实例化的跺盘变量
- 返回: 无
- 示例

```
local myPallet = MatrixPallet(0, "IsUnstack=true Userframe=8")  
Reset(myPallet)
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-17 12:41:49

查询码垛或拆垛是否完成

- 原型:

```
IsDone (Pallet)
```

- 描述: 查询码垛或拆垛是否完成
- 参数:
 - Pallet: 实例化的垛盘变量
- 返回:
 - true: 完成
 - false: 未完成
- 示例

```
Result = IsDone(myPallet)
If (result == true)
...
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:26

释放码垛实例

- 原型:

```
Release (Pallet)
```

- 描述: 释放码垛实例
- 参数:
 - Pallet: 实例化的跺盘变量
- 返回: 无
- 示例

```
Release(myPallet)
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:23

机械臂从当前位姿按照配置的码垛路径运到第一垛目标点

- 原型:

```
MoveIn (Pallet, "velAB=20 velBC=30 accAB=20 accBC=10 CP=20 SYNC=1")
```

- 描述: 表示机械臂从当前位置按照配置的码垛路径运动至第一垛目标点
- 必选参数:
 - Pallet: 实例化的垛盘变量
- 可选参数:
 - velAB: 安全过渡点与准备点之间的运动速度比例, 取值范围: 1~100
 - velBC: 准备点与第一垛目标点之间的运动速度比例, 取值范围: 1~100
 - accAB: 安全过渡点与准备点之间的运动加速度比例, 取值范围: 1~100
 - accBC: 安全过渡点与准备点之间的运动加速度比例, 取值范围: 1~100
 - CP: 运动时设置平滑过渡, 取值范围: 0~100
 - SYNC: 同步标识, 取值范围: 0或1。SYNC = 0表示异步执行, 调用后立即返回, 但不关注指令执行情况; SYNC = 1表示同步执行, 调用后, 待指令执行完才返回
- 返回: 无
- 示例

```
MoveIn(myPallet, "velAB=90 velBC=50")
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:23

机械臂从当前位姿按照配置的拆垛路径运到安全过渡点

- 原型:

```
MoveOut (Pallet, "velAB=20 velBC=30 accAB=20 accBC=10 CP=20 SYNC=1")
```

- 描述: 表示机械臂从当前位置按照配置的拆垛路径运动至安全过渡点
- 必选参数:
 - Pallet: 实例化的垛盘变量
- 可选参数:
 - velAB: 安全过渡点与准备点之间的运动速度比例, 取值范围: 1~100
 - velBC: 准备点与第一垛目标点之间的运动速度比例, 取值范围: 1~100
 - accAB: 安全过渡点与准备点之间的运动加速度比例, 取值范围: 1~100
 - accBC: 安全过渡点与准备点之间的运动加速度比例, 取值范围: 1~100
 - CP: 运动时设置平滑过渡, 取值范围: 0~100
 - SYNC: 同步标识, 取值范围: 0或1。SYNC = 0表示异步执行, 调用后立即返回, 但不关注指令执行情况; SYNC = 1表示同步执行, 调用后, 待指令执行完才返回
- 返回: 无
- 示例

```
MoveOut(myPallet, "velAB=90 velBC=50")
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:26

传送带跟踪

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:26

设置传送带编号

- 原型:

```
CnvVison(CnvID)
```

- 描述: 设置传送带编号, 建立跟踪队列

- 参数:

- CnvID: 传送带编号

- 返回:

- 0: 无错误
 - 有错误

- 示例

```
CnvVison(1)
```

把编号为1的传送带信息(分辨率、起始位置、方向和边界等)下发至机器人控制系统

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:26

获取传送带上工件状态

- 原型:

```
GetCnvObject(CnvID, ObjID)
```

- 描述:从指定的传送带上获取指定工件的数据, 检测是否进入抓取区域

- 参数:

- CnvID:传送带编号
- ObjID:工件编号

- 返回:

- 工件状态:是否检测到工件。取值范围:true 或 false
- 工件分类属性
- 工件坐标(x,y,r)

- 示例

```
P111 = {0,0,0}
while true do
  flag,typeObject,P111 = GetCnvObject(0,0)
  if flag == true then
    break
  end
  Sleep(20)
end
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:26

设置传送带用户坐标系下X、Y方向的偏移

- 原型:

```
SetCnvPointOffset(xOffset,yOffset)
```

- 描述: 设置传送带用户坐标系下X、Y方向的偏移

- 参数:

- xOffset: X轴方向偏移
- yOffset: Y轴方向偏移 单位: 毫米

- 返回:

- 0: 无错误
- 1: 有错误

- 示例

```
SetCnvPointOffset(10,10)
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-11 16:37:20

设置时间补偿

- 原型:

```
SetCnvTimeCompensation(time)
```

- 描述: 设置时间补偿, 该指令用于补偿拍照触发带来的时间延时导致传送带运动方向的抓取位置偏移

- 参数:

- time: 时间偏差, 单位: 毫秒

- 返回:

- 0: 无错误
- 1: 有错误

- 示例

```
SetCnvTimeCompensation(30)
```

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-11 16:37:20

同步指定的传送带

- 原型:

```
SyncCnv (CnvID)
```

- 描述: 同步指定的传送带 SyncCnv (CnvID)与StopSyncCnv (CnvID)之间的运动指令仅支持Move指令
- 参数:
 - CnvID: 传送带编号
- 返回:
 - 0: 无错误
 - 1: 有错误

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:26

停止同步传送带

- 原型:

```
StopSyncCnv (CnvID)
```

- 描述: 停止同步传送带, 运行完该条指令后才会继续下发执行该指令后的其他指令
- 参数:
 - CnvID: 传送带编号
- 返回:
 - 0: 无错误
 - 1: 有错误

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:26

其他指令

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook修订时间: 2021-08-06
18:06:24

设置安全皮肤开关

- 原型:

```
SetSafeSkin (status)
```

- 描述: 设置安全皮肤开关状态
- 参数:
 - Status: 安全皮肤开关状态, 0: 关闭安全皮肤; 1: 开启安全皮肤
- 返回: 无
- 示例

```
SetSafeSkin (1)
```

开启安全皮肤功能

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-12 14:06:35

设置安全皮肤避障模式开关状态

- 原型:

```
SetObstacleAvoid(status)
```

- 描述: 设置安全皮肤避障模式开关状态。如果Dobot+内安全皮肤关闭, 则本指令不起作用; 本指令只在脚本内起作用, 如果退出了脚本, 安全皮肤会回到默认接近暂停模式。
- 参数:
 - Status: 安全皮肤避障模式开关状态
 - 0: 关闭安全皮肤避障模式;
 - 1: 开启安全皮肤避障模式
- 返回: 无
- 示例

```
SetObstacleAvoid(1)
```

开启安全皮肤避障模式

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-12 19:52:03

设置碰撞等级

- 原型:

```
SetCollisionLevel(level)
```

- 描述: 设置碰撞等级
- 参数:
 - level: 碰撞等级
 - 0: 关闭碰撞检测;
 - 1~5: 等级越高越灵敏
- 返回: 无
- 示例

```
SetCollisionLevel(2)
```

设置碰撞等级为2级

Copyright © 深圳市越疆科技有限公司 2021 all right reserved, powered by Gitbook 修订时间: 2021-08-06 18:06:24