Lua接口说明

①指令目录

②参数说明

③指令说明

④指令返回错误码说明

①指令目录

1 配置

- 1.1 robot ---------- 选择机械手，包括模式
- 1.2 runLuaFile ---------- 运行指定路径的lua脚本文件
- 1.3 sendFile ---------- 通过tcp发送文件
- 1.4 openFreqStatusPort ---------- 请求服务端打开一个端口，按照给定的刷新频率不停的返回机械手状态,IP地址是连接机械手的IP地址
- 1.5 luaSleep ---------- 指定的事件条件成立之前或指定时间内，使程序处于待机状态

2 获取机械手信息

- 2.1 getPulses ---------- 获取机械手各轴马达的当前脉冲
- 2.2 getPos ---------- 获取机械手当前位置
- 2.3 getGlobalPoint ---------- 获取点信息
- 2.4 getRobotState ---------- 获取机械手的当前状态
- 2.5 addPtsFromYml ---------- 从点数据yaml文件获得点信息，添加到全局变量

3 go与ja命令设置

- 3.1 go ---------- 以 PTP(点对点) 动作从当前位置移动到指定位置，包括坐标位置移动，轴关节角度旋转
- 3.2 coord ---------- 产生一个包含坐标和姿态的表(table)
- 3.3 goCoord ---------- 以 PTP 动作从当前位置移动到指定位置
- 3.4 goDelta ---------- 以 PTP 动作从当前位置移动到增量后的指定位置
- 3.5 ja ---------- 产生一个包含各轴姿态角的表(table)
- 3.6 jaDelta ---------- 设定增量角度，在各轴原角度的基础上旋转到增加增量后的角度
- 3.7 jaToFarthest ---------- 在当前位置上，将机械手的轴关节往逆时针/顺时针方向旋转到最大角度
- 3.8 goja ---------- 根据关节角度计算的机器人坐标
- 3.9 setSpeed ---------- 设置go和ja命令的速度

- 3.10 getSpeed ---------- 获取go和ja命令运行时的速度
- 3.11 setAccel ---------- 设置go和ja命令的加速度
- 3.12 getAccel ---------- 获取go和ja命令运行时的加速度

4 move命令设置

- 4.1 move ---------- 机械手以直线插补动作的形式到达指定的位置
- 4.2 moveDelta ---------- 在当前位置到指定增量后的位置之间以直线插补动作移动机械臂。
- 4.3 moveToFarthest ---------- 在当前位置将机械手往x轴或往y轴或往z轴移动到对应轴方向的最远边界点。
- 4.4 setMoveSpeed ---------- 以表达式或数值设置PTP(点对点)动作速度相对于最大动作速度(PTP动作)的比例
- 4.5 getMoveSpeed ---------- 获取PTP(点对点)动作速度相对于最大动作速度(PTP 动作)的比例
- 4.6 setMoveAccel ---------- 设置move命令的加速度
- 4.7 getMoveAccel ---------- 获取move命令运行时的加速度

5 单方向与单旋转轴设置

- 5.1 x_pace ---------- 机械手在X轴上方向以给定的步长移动
- 5.2 y_pace ---------- 机械手在Y轴上方向以给定的步长移动
- 5.3 z_pace ---------- 机械手在Z轴上方向以给定的步长移动
- 5.4 u_pace ---------- 机械手在U轴方向以给定的角度旋转
- 5.5 v_pace ---------- 机械手在V轴方向以给定的角度旋转
- 5.6 w_pace ---------- 机械手在Z轴方向以给定的角度旋转

6 输入输出设置

- 6.1 on ---------- 打开输出位，通过I/O(输入输出)和存储器I/O(输入输出)来使用
- 6.2 off ---------- 关闭输出位，通过I/O(输入输出)和存储器I/O(输入输出)来使用
- 6.3 isOn ---------- 读取输出位的状态

7 急停设置

- 7.1 emgStop ---------- 急停设置，获取急停状态
- 7.2 getEmgStop ---------- 获取当前机械手的运动状态，停止或运动

---

②参数说明

- x,y,z表示机械手X,Y,Z坐标，单位是毫米(mm)
- u,v,w表示机械手绕z,y,x轴的旋转角,单位角度(degree)
- j1,j2,j3,j4,j5,j6表示第1-6个关节的旋转角，单位角度(degree)
- 可选参数： L表示左手模式，R表示右手模式，A表示是高手模式，B表示低手模式，I表示可被打断,无参

数表示默认手模式

- **注意：**

go 与 move 的差异:
move 和 go 都是使机器人机械臂动作的命令。两者之间的最大不同是 go 是进行 PTP(点对点) 动作，而 move 是在直线轨道上移动机械臂。在重视到达目标点时的机械臂的姿势时，使用 go 命令；而更重视控制动作中的机械臂的轨迹时，使用 move 命令。在 go 命令中，在将机械臂停止在动作的目标坐标之前，必须减速。

go 与 jump 的差异:
jump 和 go 都是以 PTP(点对点) 动作移动机械手的命令。但是，jump 拥有一个 go 所没有的功能。jump 首先将机械手的夹具末端抬起到LimZ值(Z坐标值初始值)，然后水平移动机械臂，在达到目标坐标的上空时开始下降动作。这种移动的优点是可以切实地避开障碍物，更重要的是通过吸附和配置动作可以提高作业的循环时间。

向 go 发出适当的速度和加减速度指示:
go 命令的动作速度和加减速度的设置可以通过 setSpeed 和 setAccel 命令实施。setSpeed 和 setAccel 命令与 go 命令相同，均可对 PTP(点对点) 动作进行设置，此点是至关重要的。

---

③指令说明

> 1 配置

---

robot

- 函数: robot
- 描述: 选择机械手，包括模式（"RR"或"KENT"）选择，轴数选择4轴或6轴
- 输入: robot(轴数,"品牌");
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
robot(4,"RR");//选择4轴，RR品牌
robot(4,"KENT");//选择4轴，KENT品牌
robot(6,"RR");//选择6轴，RR品牌
robot(6,"KENT");//选择6轴，KENT品牌
```

---

runLuaFile

- 函数: runLuaFile
- 描述: 运行指定路径的lua脚本文件
- 输入: runLuaFile(LuaFilePath);例如LuaFilePath=”../test.lua”
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
runLuaFile(”../test.lua”);//运行上一目录下的test.lua脚本文件
```

sendFile

- 函数: sendFile
- 描述: 通过tcp发送文件
- 输入: sendFile(”文件名称”, 文件大小)；
- 返回: Err_InputArgs --> 错误码返回值，请看返回错误码说明
- 例子:

```
--abc.yml文件大小是1000字节
FileName=”abc.yml”
sendFile(FileName,1000) --发送文件abc.yml
... --继续发送文件内容
```

---

openFreqStatusPort

- 函数: openFreqStatusPort
- 描述: 请求服务端打开一个端口，按照给定的刷新频率不停的返回机械手状态,IP地址是连接机械手的IP地址
- 输入: openFreqStatusPort(端口号,刷新时间)；刷新时间毫秒(ms)
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
--假设连接机械手的IP地址是192.168.0.1
openFreqStatusPort(5656,100)；//打开端口号，100ms刷新一次
客户端可以这个IP地址192.168.0.1,端口号5656,连接到服务器,按照每隔100ms接收机械手返回的状态
```

---

luaSleep

- 函数: luaSleep
- 描述: 指定的事件条件成立之前或指定时间内，使程序处于待机状态。设置延时时间，单位是秒(second)
- 输入: luaSleep(time);time的单位是秒(second)
- 返回: Err_InputArgs --> 错误码返回值，请看返回错误码说明
- 例子:

```
luaSleep(2.1);//等待2.1s
```

> 2 获取机械手信息

---

getPulses

- 函数: getPulses
- 描述: 获取机械手各轴马达的当前脉冲
- 输入: getPulses();
- 返回: 脉冲数组
- 例子:

```
getPulses();//返回当前的脉冲数组 [182044,182049,272,86,0,0]，分别对应马达1-6
```

---

getPos

- 函数: getPos
- 描述: 获取机械手当前位置,返回(x,y,z,u,v,w)信息
- 输入: getPos();
- 返回: 坐标和姿态数据的数组
- 例子:

```
getPos();//返回当前坐标(101.12,201.265,30.2,60.12,23.4,90.2658)
```

---

getGlobalPoint

- 函数: getGlobalPoint
- 描述: 获取点信息，例如P0是全局点变量，P0点信息如下：

```
Id: 0
Elbow: A
Hand: L
IsTeach: 1
Descripton: FDDI1_HIGH
Data: [136.86347127, -470.45043116,LUA_TUSERDATA 286.44892721, -89.21789268,
-54.75090832, -88.34883695]
```

通过getGlobalPoint(P0)获取P0信息。

- 输入: getGlobalPoint(P0);
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
getGlobalPoint(P0);//等同于调用go(P0)；go(P0, “I”)
```

---

getRobotState

- 函数: getRobotState
- 描述: 获取机械手的当前状态
- 输入: getRobotState();
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
getRobotState();

机械手状态格式如下:
err: 0
ErrMsg:
CmdName: getRobotState
Pos_mm: [418.399,-0.00199016,629.92,-1.56615,-89.9941,-178.434]
Joint_deg:
[-0.000272388,-0.000299133,-0.00895182,-0.00258502,0.0151611,0.00274658]
Pulse: [-12,-11,-264,-48,276,36]
Output: [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
Input:
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
Speed: [3,45,3]
Accel: [1,45,1]
HandMode: 0
Mov_Elbow: 0
targetOK: 1
```

---

addPtsFromYml

- 函数: addPtsFromYml
- 描述: 从点数据yaml文件获得点信息，添加到全局变量
- 输入: addPtsFromYml();
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
比如点数据文件如下:
- Id: 101
Elbow: A
Hand: R
IsTeach: 0
Description: "0"
Data: [0,1,2,3,4,5]
- Id: 102
Elbow: A
Hand: L
```

```
IsTeach: 1
Description: "test"
Data: [380.4,-0.000663152,580.9,0,-90,180]
- Id: 103
Elbow: B
Hand: L
IsTeach: 0
Description: "test point"
Data: [0,0,0,0,0,0]

    发送文件到服务端的命令: "sendFile('MyPoint.pts')"; 成功后服务端会存在一个同样名称的副本"MyPoint.pts"
    将"MyPoint.pts"标志了已示教的点添加到全局变量, 例如上述例子的Id=102被示教"IsTeach: 1",执行完该步骤后lua会添加一个P102的全局变量; 命令是: 'addPtsFromYml("MyPoint.pts")'
    然后即可使用"go(P102)"或"go(P102, 'I')",控制机械手按照指定的手模式"LA"到达位置: [380.4,-0.000663152,580.9,0,-90,180].
```

3 go与ja命令设置

---

go

- 函数: go
- 描述: go 用于以 PTP(点对点)动作将机器人机械臂的所有关节同时移动。机械手以给定的姿态到达指定位置(坐标或关节角度)
- 输入:
  go(x,y,z,u,v,w,'可选参数');//以具体坐标值指定移动目标。(x,y,z)表示设置的坐标，(u,v,w)表示设置的姿态角，可选参数有L,R,A,B,I,含义可见上面的参数说明。
  go(P0);//以特定点编号进行指定，如P0是一个点编号，则设置机械手去到P0点信息的坐标和姿态角
  go(ja(j1,j2,j3,j4,j5,j6),"可选参数"); //设置各关节姿态角
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
go有两种模式:
1. 笛卡尔坐标系
1.1
6轴机械手:
    go(10.0,20.5,30.8,35.23,40.1,-20.9);//非打断模式，默认手模式，机械手到达(10.0,20.5,30.8)的XYZ坐标，姿态角到达(35.23,40.1,-20.9)。

    go(0.01,-202.2,300.221,20,-40.23,90.41234, 'I');//打断模式，默认手模式，机械手到达(0.01,-202.2,300.221)的XYZ坐标，姿态角到达(20,-40.23,90.41234)。

    go(20,433.2,103.3,10,-20,-60, 'R');//非打断模式，右手模式，机械手到达(20,433.2,103.3)的XYZ坐标，姿态角到达(10,-20,-60)。

    go(0,500.323,414.23,-123.24,-50,-69.4, 'IRA');//打断模式，右高手模式，机械手到
```

达(0,500.323,414.23)的XYZ坐标，姿态角到达(-123.24,-50,-69.4)。

1.2
4轴机械手：
go(10.0,20.5,30.8,35.23);//非打断模式，默认手模式，机械手到达(10.0,20.5,30.8)的XYZ坐标，姿态角到达(35.23)。

go(0.01,-202.2,300.221,20,'I');//打断模式，默认手模式，机械手到达(0.01,-202.2,300.221)的XYZ坐标，姿态角到达(20)。

go(20,433.2,103.3,10,'R');//非打断模式，右手模式，机械手到达(20,433.2,103.3)的XYZ坐标，姿态角到达(10)。

go(0,500.323,414.23,-123.24,'IRA');//打断模式，右高手模式，机械手到达(0,500.323,414.23)的XYZ坐标，姿态角到达(-123.24)。

1.3
调用点信息：
go(P0);//以点编号表示位置信息，如P0是一个点编号，表示机械手去到P0点信息的坐标和姿态角。

P0点信息的产生是通过getGlobalPoint()获取，即getGlobalPoint(P0)将获取的点信息给P0。
P0是全局点变量，例如P0点信息如下：
Id: 0
Elbow: A
Hand: L
IsTeach: 1
描述: FDDI1_HIGH
Data: [136.86347127, -470.45043116,LUA_TUSERDATA 286.44892721, -89.21789268, -54.75090832, -88.34883695]

go(P0);//表示以'LA'手模式去到坐标位置[136.86347127, -470.45043116, 286.44892721, -89.21789268, -54.75090832, -88.34883695]
等同执行goCoord(136.86347127, -470.45043116, 286.44892721, -89.21789268, -54.75090832, -88.34883695,'LA')

2. 关节模式：
go(ja(j1,j2,j3,j4,j5,j6),”可选参数”);

go(ja(90,100,120,-20,-50,10)，”I”);//打断模式,关节j1-j6旋转设定的角度，分别旋转(90,100,120,-20,-50,10)角度

go(ja(90,0,0,0,0,0)，””);//非打断模式,关节j1-j6旋转设定的角度，分别旋转(90,0,0,0,0,0)角度

coord

- 函数: coord
- 描述: 产生一个包含坐标和姿态的table
- 输入: coord(x,y,z,u,v,w);
- 返回: 一个表(table)
- 例子:

```
Point=cood(518.4, 0, 629.89, 0, -90, -180);//产生一个表(table)，名字叫Point
go(Point);//传入一个表(table)给go命令
go(Point,"I");//机械手到达(518.4, 0, 629.89)位置，姿态角旋转到(0, -90, -180),打断模式

等同于
go(cood(518.4, 0, 629.89, 0, -90, -180));//机械手到达(518.4, 0, 629.89)位置，姿态角旋转到(0, -90, -180),非打断模式
go(cood(518.4, 0, 629.89, 0, -90, -180),"I");//机械手到达(518.4, 0, 629.89)位置，姿态角旋转到(0, -90, -180),打断模式
```

---

goCoord

- 函数: goCoord
- 描述: 机械手以点对点的方式到达指定坐标
- 输入: goCoord(x,y,z,u,v,w,'可选参数'),可选参数:L表示左手模式，R表示右手模式，A表示是高手模式，B表示低手模式，I表示可被打断,无参数
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
goCoord(10.0,20.5,30.8,35.23,40.1,-20.9);//非打断模式，默认手模式，机械手到达(10.0,20.5,30.8)，姿态角(35.23,40.1,-20.9)。
goCoord(0.01,-202.2,300.221,20,-40.23,90.41234,'IR');//打断模式，右手模式，机械手到达(0.01,-202.2,300.221)，姿态角(20,-40.23,90.41234)。
goCoord(0,500.323,414.23,-123.24,-50,-69.4,"IRA");//打断模式，右高手模式，机械手到达(0,500.323,414.23)，姿态角(-123.24,-50,-69.4)。
```

---

goDelta

- 函数: goDelta
- 描述: 给定坐标增量，在上一坐标上增加给定的增量，以点对点的运动方式到达增量后的坐标。
- 输入:
  goDelta(detlax,detlay,detlaz,detlau,deltav,detlaw);
  detlax,detlay,detlaz,分别表示各xyz轴的坐标增量，单位是毫米(mm)；

detlau,deltav,detlaw，分别表示各uvw轴的角度增量，单位是角度(degree)；

- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
goDetla(100,200,300,10.3,0,-20.4);//在原坐标的基础上，xyz轴各增加(100,200,300);在原姿态的基础上，uvw轴各增加(10.3,0,-20.4)角度。
```

---

ja

- 函数: ja
- 描述: 产生一个包含各轴姿态角的table
- 输入: ja(j1,j2,j3,j4,j5,j6);
- 返回: 一个表(table)
- 例子:

```
J=ja(90,-90,0, 0, -90, -180);//产生一个table，名字叫Point
go(J);//传入一个table给go函数，//各轴依次转动(90,-90,0, 0, -90, -180)姿态角

等同于
go(ja(90,-90,0, 0, -90, -180));//各轴依次转动(90,-90,0, 0, -90, -180)姿态角
```

---

jaDelta

- 函数: jaDelta
- 描述: 设定增量角度，在各轴原角度的基础上旋转到增加增量后的角度。
- 输入: jaDelta(10,20,30,-40,-50,-60);
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
jaDelta(10,20,30,-40,-50,-60);//设定各轴的增量角度是(10,20,30,-40,-50,-60),在原姿态角的基础上旋转到增加增量后的角度
```

---

jaToFarthest

- 函数: jaToFarthest
- 描述: 在当前坐标点上，根据传入的参数将机械手的关节往正/负方向旋转到最大角度。
- 输入: jaToFarthest(index,+/-1)；index范围是0-5，0-5分别表示各j1-j6各关节编号；+1表示正方向，-1表示负方向
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
jaToFarthest(0,1);//表示j1关节往正方向旋转到最大角度。
JaToFarthest(5,-1);//表示j6关节往负方向旋转到最大角度。
```

---

goja

- 函数: goja
- 描述: 机械臂的关节以给定的角度旋转
- 输入: goja(j1,j2,j3,j4,j5,j6，'可选参数')，可选参数是'I'，表示打断模式
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
goja(90,0,0,0,0,0); //默认手模式，关节j1-j6旋转设定的角度，分别旋转(90,0,0,0,0,0)角
度
goja(23.43,-90,20,3.3,23.456,0,'I');//可打断模式，关节j1-j6旋转设定的角度，分别旋转
(23.43,-90,20,3.3,23.456,0)角度
```

---

setSpeed

- 函数: setSpeed
- 描述: 设置go和ja命令的速度
- 输入: setSpeed(n);n属于0-100的浮点数，表示设置的速度相当于最大速度的百分比
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
setSpeed(40.43);//设置机械手当执行go或ja命令时的速度，相当于最大运行速度的40.43%的速度
```

---

getSpeed

- 函数: getSpeed
- 描述: 获取go和ja命令运行时的速度
- 输入: getSpeed();
- 返回: 当前速度相对于最大速度的百分比，范围是0-100的浮点数或整数
- 例子:

```
getSpeed();//当执行go或ja命令时,调用getSpeed()会返回机械手的当前速度
```

---

setAccel

- 函数: setAccel
- 描述: 设置go和ja命令的加速度
- 输入: setAccel(n);n属于0-100的浮点数，表示设置加速度相当于最大加速度的百分比
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
setAccel(80.89);//设置机械手执行go或ja命令时的速度，相当于最大运行速度的80.89%的加速度
```

---

getAccel

- 函数: getAccel
- 描述: 获取go和ja命令运行时的加速度
- 输入: getAccel();
- 返回: 当前加速度相对于最大加速度的百分比，范围是0-100的浮点数或整数
- 例子:

```
getAccel();//当执行go或ja命令时,调用getAccel()会返回机械手的当前加速度
```

> 4 move命令设置

---

move

- 函数: move
- 描述: 机械手以直线插补动作的形式到达指定的位置，即机械手在当前位置到指定位置之间以直线的方式移动。
- 输入: move(x,y,z,u,v,w，'可选参数');可选参数只有I，I表示可被打断
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
move(10.345,20.2323,30.5342,90,-45,45);//默认手模式，以直线的方式从当前坐标移动到坐标(10.345,20.2323,30.5342)，到达姿态角(90,-45,45)
move(10.2344,20.9685,30.3245,90.09234,-45.134,45.6547,'I');//打断模式，以直线的方式从当前坐标移动到坐标(10.2344,20.9685,30.3245)，到达姿态角(90.09234,-45.134,45.6547)

注意:
不能利用 move 进行的操作:
进行动作之前，不能确认动作范围。这样的话，即使目标坐标位置在容许动作范围之内，而如果到达此处的轨迹通过容许动作范围以外位置，则可能会导致机械臂突然停止，并造成伺服冲击，导致发生故障，这很危险。为了防止发生这种情况，高速执行 move 之前，请先以低速确认动作范围。
也就是说，即使目标坐标在机械臂动作范围之内，从物理角度来讲，如果通过 move 动作到达此处的轨迹超出机械臂容许动作范围，机械臂则动不了。
```

moveDelta

- 函数: moveDelta
- 描述: 给定坐标增量，在上一坐标上增加给定的增量，以直线运动方式到达增量后的坐标。
- 输入:
  moveDelta(detlax,detlay,detlaz,detlau,deltav,detlaw);
  detlax,detlay,detlaz,单位是毫米(mm);
  detlau,deltav,detlaw，单位是角度(degree);
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
goDetla(100,200,300,90,0,-90);//在原坐标的基础上，xyz轴各增加(100,200,300);在原姿态的基础上，uvw轴各增加(90,0,-90)角度。
```

---

moveToFarthest

- 函数: moveToFarthest
- 描述:在当前坐标点上，根据传入的参数将机械手往x轴或y轴或z轴移动到对应轴方向的最远边界点。
- 输入: (index, +/-1);index范围是0-5，0-2分别表示各xyz轴，3-5分别表示各uvw轴；+1表示正方向，-1表示负方向
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
index: [0,2] 底层调用move到最远点; [3,5]底层调用的是go指令
moveToFarthest(0,1);//表示往当前坐标点的x轴正方向移动到最远边界点，底层调用move命令
moveToFarthest(2,-1);//表示往当前坐标点的z轴负方向移动到最远边界点，底层调用move命令
moveToFarthest(4,1);//表示往当前坐标点的V轴正方向选转到最大正方向角度，底层调用go命令
moveToFarthest(5,-1);//表示往当前坐标点的w轴正方向移动到最大负方向角度，底层调用go命令
```

---

setMoveSpeed

- 函数: setMoveSpeed
- 描述: 以表达式或数值指定相对于最大动作速度（ PTP 动作）的比例（ 1 - 100 的整数或浮点数，单位： %）。
- 输入: setMoveSpeed(speed);speed表示百分比，范围是0-100，
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
setMoveSpeed(50.0);//以50.0%的百分比设置运行速度
```

getMoveSpeed

- 函数: getMoveSpeed
- 描述: 获取速度数值指定相对于最大动作速度（ PTP 动作）的比例（ 1 - 100 的整数，单 位： %）。
- 输入: getMoveSpeed();
- 返回: RetVal --> 错误码返回值，请看返回错误码说明 ; speed --> 获取的当前move指令的速度
- 例子:

```
GetSpeed=getMoveSpeed();//GetSpeed得到当前速度相对于相对于最大动作速度（ PTP 动作）的比例
```

---

setMoveAccel

- 函数: setMoveAccel
- 描述: 设置move命令的加速度
- 输入: setMoveAccel(n);n属于0-100的浮点数，表示速度相当于最大的速度的百分比
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
setMoveAccel(40.89);//当执行move命令时，调用setMoveAccel(40.89)，则会设置当前的加速度为最大运行加速度的40.89%
```

---

getMoveAccel

- 函数: getMoveAccel
- 描述: 获取move命令运行时的加速度
- 输入: getMoveAccel();
- 返回: 当前加速度相对于最大加速度的百分比，范围是0-100的浮点数
- 例子:

```
getMoveAccel();//当执行move命令时，调用getMoveAccel()，则会获取当前加速度相当于最大运行加速度的百分比
```

> 5 单方向与单旋转轴设置

---

x_pace

- 函数: x_pace
- 描述: 机械手在X轴上方向以给定的步长移动
- 输入: x_pace(DeltaX);单位毫米(mm)，在X轴上以DeltaX mm为步长移动

- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
x_pace(10);//在X轴上以10 mm为步长移动
```

---

y_pace

- 函数: y_pace
- 描述: 机械手在Y轴上方向以给定的步长移动
- 输入: y_pace(DeltaY);单位毫米(mm)，在Y轴上以DeltaY mm为步长移动
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
y_pace(10);//在Y轴上以10 mm为步长移动
```

---

z_pace

- 函数: z_pace
- 描述: 机械手在Z轴上方向以给定的步长移动
- 输入: z_pace(DeltaZ);单位毫米(mm)，在Z轴上以DeltaZ mm为步长移动
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
z_pace(10);//在Z轴上以10 mm为步长移动
```

---

u_pace

- 函数: u_pace
- 描述: 机械手在U轴方向以给定的步长旋转
- 输入: u_pace(DeltaU);单位角度(degree)，在U轴上以DeltaU度为步长旋转
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
u_pace(0.1);//在U轴上以0.1度为步长旋转，单位°
```

---

v_pace

- 函数: v_pace

- 描述: 机械手在V轴方向以给定的步长旋转
- 输入: v_pace(DeltaV);单位角度(degree)，在V轴上以DeltaV度为步长旋转
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
v_pace(0.1);//在V轴上以0.1度为步长旋转，单位°
```

---

w_pace

- 函数: w_pace
- 描述: 机械手在W轴方向以给定的步长旋转
- 输入: w_pace(DeltaW);单位角度(degree)，在W轴上以DeltaW度为步长旋转
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
w_pace(0.1);//在W轴上以0.1度为步长旋转，单位°
```

> 6 输入输出设置

---

on

- 函数: on
- 描述: 以整数值指定要设为 ON 的 I/O 输出位。I/O范围是0-31
- 输入: on(1,2,3,...);//1,2,3,...表示IO编号
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
on(2,4);//设置第2,4IO编号为ON状态
on(1,2,3,4,5,6,...)；//表示设置1,2,3,4,5,6，...等IO的ON状态
```

---

off

- 函数: off
- 描述: 以整数值指定要设为 OFF 的 I/O 输出位。I/O范围是0-31
- 输入: off(1,2,3,...);//1,2,3,...表示IO编号
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
off(2,4);//设置第2,第4个IO为ON状态
off(1,2,3,4,5,6,...)；//表示设置1,2,3,4,5,6，...等IO的OFF状态
```

---

isOn

- 函数: isOn
- 描述: 查询I/O状态,状态是ON或OFF，I/O编号范围是0-31，只能输入单个I/O查询
- 输入: isOn(n)；n的范围是0-31
- 返回: RetVal --> 错误码返回值，请看返回错误码说明； State --> I/O状态
- 例子:

```
isOn(2);//查询第2个I/O的状态，返回ON或OFF
```

> 7 急停设置

---

emgStop

- 函数: emgStop
- 描述: 急停设置，1-->急停，0-->取消急停，emgStop()获取当前状态
- 输入:
  emgStop(0);
  emgStop(1);
  emgStop();
- 返回: RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
emgStop(1);//急停
emgStop(0);//取消急停
emgStop();//获取当前状态
```

---

getEmgStop

- 函数: getEmgStop
- 描述: 获取当前机械手的运动状态，停止还是运动
- 输入: getEmgStop();
- 返回: 0表示正在运动，1表示停止运动，RetVal --> 错误码返回值，请看返回错误码说明
- 例子:

```
getEmgStop();//返回0或1；0表示正在运动，1表示停止运动
```

---

④指令返回错误码说明:

- -# Err_MoveThreadBusy -27: There is a moving command running in the background thread.(后台线程有一个移动的命令在运行)
- -# Err_ScriptFileRunning -26: There is a lua script file running in the background thread.(lua脚本文件正在后台线程运行)
- -# Err_SetThreadAttr -25: Faile to set the thread attribution.(设置线程属性失败)
- -# Err_CreateThread -24: Failed to create the background thread.(创建后台线程失败)
- -# Err_AccessSemFlag -23: Failed to access the semaphore flag.(获取信号量标志失败)
- -# Err_OverRouteShm -22: Route shared memory is writting.(Route-shm共享内存正在写入)
- -# Err_RegexPattern -21: the Regex Pattern is wrong.(正则表达式模式错误)
- -# Err_YamlParser -18: Failed to parser the yaml file, or the yaml file not exist.(解析yaml文件失败，或yaml文件不存在)
- -# Err_RouteShm -17: Failed to read/write the Route Shared Memory.(读写Route-shm共享内存失败)
- -# Err_Power -16: The power flag of robot is not desirable.(机器的电源标志不合法)
- -# Err_RobotInit -15: The LogicRobot object is not initialized.(LogicRobot对象未初始化)
- -# Err_IONum -14: The number of IO is invalid.(IO编号无效)
- -# Err_InputArgs -13: The input function parameters is invalid.(输入函数参数无效)
- -# Err_ParamNotInit -12: Parameter not be initialized.(参数初始化失败)
- -# Err_FoundFile -11: Failed to find the file.(未能找到文件)
- -# Err_UnequalZero -10: the value != 0. But the value expects 0.(当前实际值不是0，但期望值是0)
- -# Err_EqualZero -9 : the value == 0. But the value does not expect 0.(当前实际值是0，但期望值不是0)
- -# Err_CureSteps -8 : The calculated cure-steps less than 4.(计算的固化步骤小于4)
- -# Err_MemMalloc -7 : Failed to call the malloc() function.(调用malloc()函数失败)
- -# Err_Ik -6 : Failed to solve the ik.(解算ik失败)
- -# Err_OutCoordRange -5 : The calculated coordinate (mm) value is out of the valid range.(计算的坐标值超出有效范围)
- -# Err_OutJointsRange -4 : The calculated robot joint angle is out of the valid range.(计算的机械臂关节角度超出有效范围)
- -# Err_Trajectory -3 : Failed to calculate the robot trajectory.(计算机器轨迹失败)
- -# Err_RWShm -2 : Failed to Read/Write Shared Memory.(读写共享内存失败)
- -# Err_Error -1 : General Error.(错误)
- -# OK 0 : OK.(正常)
