

ER 系列工业机器人三维视觉调试手册

RCS2 V1.5.3

修订记录

序号	日期	版本	描述
1	2018.08.17	V1.0	1.初次发布
2	2018.10.10	V1.1	1.增加视觉对机器人的命令
3	2019.01.16	V1.2	1.修改命令中域的属性值
4	2022.03.09	V1.4	1.修改视觉协议介绍 2.修改部分接口描述 3.修改设置负载变量 4.修改视觉配置介绍 5.修改通讯心跳说明 6.修改调试步骤
5	2021.08.04	V1.5.1	1.增加 3.39、3.40、3.41、3.42、3.43、3.44、3.45、3.46、3.47 接口
6	2021.12.03	V1.5.2	1.更改接口 3.17、3.42、3.45，增加 3.48，3.49，3.50，第 2 章增加使用说明描述
7	2022.07.27	V1.5.3	1.更改部分文字描述和删除接口中空 格； 2.修复 3.24、3.26、3.36 和 3.37 接 口实现。

目录

前言	4
第 1 章 功能简介	5
第 2 章 视觉协议介绍	5
第 3 章 视觉命令介绍	6
3.1 修改位置点命令	10
3.2 获取位置点命令	11
3.3 修改变量值	11
3.4 获取变量值	12
3.5 获取机器人当前关节位置	12
3.6 获取机器人当前世界坐标系位置	12
3.7 修改 IO 口值	12
3.8 获取 IO（实际和虚拟）口值	13
3.9 更改机器人当前模式	14
3.10 获取当前机器人模式	14
3.11 修改全局速度	14
3.12 修改程序运行模式	15
3.13 获取机器人运行状态	15
3.14 报警复位	15
3.15 获取报警错误码	15
3.16 程序运行	16
3.17 程序停止	16
3.18 加载程序文件	16
3.19 注销工程	16
3.20 注销程序	17
3.21 判断机器人是否在动作	17
3.21 判断是否有工程加载	17
3.22 设置 PC 指针	17
3.23 伺服使能控制	18
3.24 获取当前伺服使能状态	18
3.25 设置手动的运行坐标系	18
3.26 获取当前点动的坐标系	18
3.27 单轴点动	18
3.28 停止当前点动	19

3.29 获取全局速度值	19
3.30 多轴定点运动(带速度参数无过渡参数)	19
3.31 多轴定点运动(带速度参数和过度参数)	20
3.32 多轴运动停止	20
3.33 示教指定点	21
3.34 寻位运动功能	21
3.35 设置负载变量	22
3.36 选择工具坐标系	22
3.37 选择用户坐标系	22
3.38 设置机器人报错停止	22
3.39 GetCurWPosV3 获得位置带 CFG 值	23
3.40 GetVarV3 获得变量值	23
3.41 SetVarV3 设置变量值	25
3.42 MovePointV3 机器人运动接口	29
3.43 GetToolV3 获得当前工具	31
3.44 GetUserCoordV3 获得当前用户坐标	31
3.45 MoveFinsih 运动任务完成信号	31
3.46 GetSoftLimits 获得当前机器人的轴软限位	32
3.47 SetSoftLimits 获得当前机器人的轴软限位	32
3.48 setMultiIOValue 设置多 IO 接口	32
3.49 GetMultiIOValue 读取多 IO 口值	33
3.50 Clear3dCmds 清空 3d 接口缓存的命令	33
第 4 章 视觉配置介绍	33
第 5 章 三维视觉调试	34
5.1 通信心跳的说明	34
5.2 SocketTest 调试助手	35
5.3 调试步骤	35

前言

概述

本手册适用于控制系统 RCS2 V1.24 版本之后，描述自主控制器视觉功能介绍，包括配置三维视觉接口配置、三维视觉调试方法。

读者对象

本手册仅供受过培训，熟悉各种适用国家标准的“控制、自动化和驱动工程”领域专业人员。

- 系统生产商：对系统进行功能诊断的操作人员。
- 系统集成商：指机床厂家的技术人员。

注意事项

- 在安装和调试这些组件时，操作人员必须严格遵循本文档的说明和解释。
- 相关负责人员必须确保所述产品的应用或使用满足所有安全要求，包括相关法律、法规、准则和标准。
- 尽管本文档经过精心编制，但由于其中所描述的产品仍处于不断更新换代中，我们可能不会在每次更新后都检查文档中所描述的产品性能数据、标准或其它特性总是与实际产品相一致。
- 本文档中难免会出现一些技术或者编辑错误，我们保留随时对文档信息做出修改之权力，恕不另行通知。对于已经变更的产品，如果本文档中的数据、图表以及文字描述没有修改，我们将不再特别加以声明。
- 任何人不得对软、硬件配置进行文本档中规定之外的修改，ESTUN 公司对因此而造成的一切后果不承担任何责任。
- 本文档中出现图示单位在没有特别标注说明时，默认单位为毫米 mm。

安全说明

 警告	受伤的危險 不遵守本标志相关的安全说明将危及个人生命和健康安全。
 注意	对环境和设备有危險 不遵守本标志相关安全说明可能明显危害环境和设备安全。
 说明	说明或提示 该标志表示这些信息能够帮助您更好的理解安全说明。

第 1 章 功能简介

机器人视觉是指使机器人具有视觉感知功能的系统，是机器人系统组成的重要组成部分之一。机器人视觉可以通过视觉传感器获取环境的三维图像，并通过视觉处理器进行分析和解释，进而转换为命令，用来设置机器人的变量、位置点、IO，同时也需要获取机器人的变量、位置点、IO、当前位置等命令，让机器人能够辨识物体，受视觉逻辑控制。

第 2 章 视觉协议介绍

机器人作为 TCP 客户端，视觉设备作为 TCP 服务端。

机器人发送数据到服务端设备

机器人端通过指令 `SendMessage` 发送字符串到服务端，设备字符串内容由用户自定义。在机器人运行发送数据指令 `SendMessage` 后，机器人将向服务端设备发送给定的字符串。

机器人指令执行：`SendMessage ("Message",0)`

服务端收到内容：`[Msg: Message]`

服务端设备发送数据到机器人

视觉发送命令字符串到机器人时，机器人会发送应答 `Ack` 给服务端设备

服务端设备发送：`[GetCurJPos();id=1]`

格式：`[ 命令 + ; + id ]`

说明：`[]`是表示一帧完整命令，分号前面是命令，分号后面 `id` 对应命令 `id`，这是一个随机不重复的整数，用于判断错误跳帧的，因为机器人 `Ack` 返回的应答帧`[id = 1; Ok]`,就会有对应的 `id`。

机器人应答：`[id = 1; Ok; XXX]`

格式: [id + ; + (Ok/FAIL) + ; + 数据]

指令使用说明

接口类型

1: 状态、IO、变量等读取接口

2: 状态、IO、变量设置接口

3: 运动接口

接口命令执行方式:

a) 在执行类型 2 和类型 3 的接口命令时:

- 1) 存在命令队列, 接口命令必须进入队列依次执行。
- 2) 接口命令执行完成后, 反馈应答(未执行没有应答)。
- 3) 队列满, 接口命令未能进入队列, 反馈队列满请等待应答。

b) 在执行类型 1 的命令时:

- 1) 立即执行, 立即反馈。

c) 异常情况:

- 1) 运动指令出错或停止, 反馈[RobotStop:XX];其它设置指令基本不受影响, 客户需要通过指令获得错误状态进行错误处理

d) 命令队列指令:

- 1) 命令队列清空

e) 运动指令, 检测安全门信号, 如果有安全门, 反馈[SafeDoorIsOpen:XX], 指令结束。

说明: 报错、停止运动指令无法继续, 不支持暂停、继续。

第 3 章 视觉命令介绍

视觉命令的使用需要配合视觉软件及外配摄像头一起使用, 并需要提前进行视觉的标定。

视觉对机器人的命令包括：

修改位置点	SetPointPos_1
获取位置点	CamGetPoint_s
修改变量值	CamSetVar_1_s
获取变量值	CamReadVar_s
获取机器人当前关节位置	GetCurJPos
获取机器人当前世界坐标系位置	GetCurWPos
修改 IO 口值	SetIOValue
获取 IO（实际和虚拟）口值	IOGetDout IOGetDin IOGetAout IOGetAin IOGetSimDout IOGetSimDin IOGetSimAout IOGetSimAin
更改机器人当前模式	changemode_IFace

获取当前机器人模式	getCurSysMode_IFace
修改全局速度	setGoableSpeed_IFace
修改程序运行模式	setRunMode_IFace
获取机器人运行状态	getRobotRunStatus_IFace
报警复位	resetErrorId_IFace
获取报警错误码	getErrorId_IFace
程序运行	startRun_IFace
程序停止	stopRun_IFace
加载程序文件	loadUserPrjProg_IFace
注销工程	UnloadUserPrj_IFace
注销程序	UnloadUserProg_IFace
判断机器人是否在动作	IsRobotMoving_IFace
判断是否有工程加载	IsProgramLoaded_IFace
设置 PC 指针	SetPc_IFace
伺服使能控制	SetMotServoStatus_IFace
获取当前伺服使能状态	GetServoSts_IFace

设置手动的运行坐标系	SetCoordType_IFace
获取当前点动的坐标系	GerCurCoordType_IFace
单轴点动	JogMotion_IFace
停止当前点动	JogMotionStop_IFace
获取全局速度值	getGoableSpeed_IFace
多轴定点运动(带速度参数无过渡参数)	MoveToSelectPoint_IFace
多轴定点运动(带速度参数和过度参数)	MoveToSelectPointb_IFac
多轴运动停止	StopDestPosMotion_IFace
示教指定点	TeachSelectPoint_IFace
寻位运动功能	MoveWithSearch_IFace
设置负载	SetPayload_IFace
选择工具坐标系	SetTool_IFace
选择用户坐标系	SetCoord_IFace
设置机器人报错停止	SetRTtoErr_IFace
获得机器人位置带 CFG 值	GetCurWPosV3

获得变量接口	GetVarV3
设置变量接口	SetVarV3
机器人运动接口	MovePointV3
获得当前工具	GetToolV3
获得当前用户坐标	GetUserCoordV3
清除运动任务号	ClearJobIdV3
获得轴软限位	GetSoftLimits
设置轴软限位	SetSoftLimits

视觉使用前需要机器人标定一个用户坐标系，此坐标系需与视觉的坐标系一致。

3.1 修改位置点命令

命令格式: SetPointPos_1(string PName, string value, int type, int scope)

PName: 位置点名字

value: 位置点值 (example: 11.0_12.1_13.2_14.3_15.4_16.5)

type: 位置点 5 轴正负 0—5 轴正, 1—5 轴负 (AP05 无效)

scope: 域 0: 表示系统, 1 表示全局, 2 表示工程, 3 表示程序

发送: [SetPointPos_1("P1", "1.1_2.2_3.3_4.4_5.5_6.6", 0, 2); id=3]

接收: 成功[id = 3; Ok] 失败[id = 3; FAIL]

3.2 获取位置点命令

命令格式: CamGetPoint_s(string pointName,int scope)

pointName:位置点名字

scope:域, 0:表示系统, 1 表示全局, 2 表示工程, 3 表示程序

发送: [CamGetPoint_s("P1",1);id=13] (参数 1 是名字)

接收:

成功:

1、APOS:[id = 13; Ok; %0.3f,%0.3f,%0.3f,%0.3f,%0.3f,%0.3f] (值分别是 A1, A2, A3, A4, A5, A6)

2、CPOS:[id = 13;
Ok; %0.3f,%0.3f,%0.3f,%0.3f,%0.3f,%0.3f,%d,%d,%d,%d,%d,%d] (值分别是 X, Y, Z, A, B, C, mode, cf1, cf2, cf3, cf4, cf5, cf6)

失败: [id = 13; FAIL], mode 是 5 轴正负, 0—5 轴正, 1—5 轴负

3.3 修改变量值

命令格式: CamSetVar_1_s(string varName,string dDataIn,int type,int scope)

varName:是变量名字

dDataIn:是值,以字符串的形式给的值

type:是类型:1 表示 int 型; 2 表示是 real 型

scope:域, 0:表示系统, 1 表示全局, 2 表示工程, 3 表示程序

发送: [CamSetVar_1_s("Result","1.0",1,2);id=112]

接收: 成功[id = 112; Ok] 失败[id = 112; FAIL]

3.4 获取变量值

命令格式: `CamReadVar_s(string varName,int type,int scope)`

varName:是变量名字,

type:是类型 1 表示 int 型, 2 表示是 real 型

scope:域, 0:表示系统, 1 表示全局, 2 表示工程, 3 表示程序

发送: `[CamReadVar_s("Result", 2, 2);id=122]`

接收:成功[id = 122; 0k; 1.1](最后一个值是值) 失败[id = 122; FAIL]

3.5 获取机器人当前关节位置

命令格式: `GetCurJPos()`

发送:`[GetCurJPos();id=1]`

接收成功:[id = 1; 0k; %0.3f %0.3f %0.3f %0.3f %0.3f %0.3f]其中的值分别对应 (J1, J2, J3, J4, J5, J6)

接收失败:[id = 1; FAIL]

3.6 获取机器人当前世界坐标系位置

命令格式: `GetCurWPos()`

发送`[GetCurWPos();id=2]`

接收成功 : [id = 2; 0k; %0.3f %0.3f %0.3f %0.3f %0.3f %0.3f %d]其中的值分别对应(x, y, z, a, b, c, mot)

接收失败: [id = 2; FAIL]

3.7 修改 IO 口值

命令格式: `SetIOValue(int ioIndex, int ioType, double ioValue)`

ioIndex:端口号

ioType:端口类型 1- 数字输出(Dout) 2- 模拟输出(Aout)

11-虚拟数字输入(SimDI) 12-虚拟数字输出 (SimDout)

13-虚拟模拟量输入 (SimAin) 14-虚拟模拟量输出 (SimAout)

ioValue:端口值 (模拟量可以为整数、浮点数, 正负都可, 数字量只能为 0/1)

发送:[SetIOValue(11,1,1);id=4]

接收:成功[id = 4; 0k] 失败[id = 4; FAIL]

3.8 获取 IO (实际和虚拟) 口值

命令: IOGetDout(int ioIndex)

发送: [IOGetDout(1);id=5] (括号中是参数 ioIndex)

接收: 成功[id = 5; 0k; 1](最后一位是端口值) 失败[id = 5; FAIL]

命令: IOGetDin(int ioIndex)

发送: [IOGetDin(2);id=6] (括号中是参数 ioIndex)

接收: [id = 6; 0k; 1] (最后一位是端口值) 失败[id = 6; FAIL]

命令: IOGetAout(int ioIndex)

发送: [IOGetAout(1);id=7] (括号中是参数 ioIndex)

接收: [id = 7; 0k; 1.1] (最后一位是端口值) 失败[id = 7; FAIL]

命令: IOGetAin(int ioIndex)

发送: [IOGetAin(1);id=8] (括号中是参数 ioIndex)

接收: [id = 8; 0k; 1.1] (最后一位是端口值) 失败[id = 8; FAIL]

命令: IOGetSimDout(int ioIndex)

发送: [IOGetSimDout(1);id=5] (括号中是参数 ioIndex)

接收：成功[id = 5; 0k; 1] (最后一位是端口值) 失败[id = 5; FAIL]

命令：IOGetSimDin(int ioIndex)

发送：[IOGetSimDin(2);id=6] (括号中是参数 ioIndex)

接收：[id = 6; 0k; 1] (最后一位是端口值) 失败[id = 6; FAIL]

命令：IOGetSimAout(int ioIndex)

发送：[IOGetSimAout(1);id=7] (括号中是参数 ioIndex)

接收：[id = 7; 0k; 1.1] (最后一位是端口值) 失败[id = 7; FAIL]

命令：IOGetSimAin(int ioIndex)

发送：[IOGetSimAin(1);id=8] (括号中是参数 ioIndex)

接收：[id = 8; 0k; 1.1] (最后一位是端口值) 失败[id = 8; FAIL]

3.9 更改机器人当前模式

命令：changemode_IFace(int modeType)

modeType：模式 (0 代表手动 1 代表自动 2 代表远程)

发送：[changemode_IFace(0); id = 9] (括号中是参数 modeType)

接收：成功[id = 9; 0k] 失败[id = 9; FALL]

3.10 获取当前机器人模式

命令：getCurSysMode_IFace()

发送：[getCurSysMode_IFace(); id = 10]

接收：成功[id = 10; 0k; 1] (0-手动 1-自动 2-远程) 失败[id = 10; FALL]

3.11 修改全局速度

命令：setGoableSpeed_IFace(int goableSpeed)

发送: [setGoableSpeed_IFace(20); id = 11]

接收: 成功[id = 11; Ok] 失败[id = 11; FALL]

3.12 修改程序运行模式

命令: setRunMode_IFace(int modeType)

modeType 1-单步 2-连续

发送: [setRunMode_IFace(1); id = 12]

接收: 成功[id = 12; Ok] 失败[id = 12; FALL]

3.13 获取机器人运行状态

命令: getRobotRunStatus_IFace()

发送: [getRobotRunStatus_IFace(); id = 13]

接收: 成功[id = 13; Ok; 3] (PROG_INIT = 0, PROG_RUNNING = 1, PROG_PAUSED = 2, PROG_STOPPED = 3, PROG_ERROR = 4) 失败[id = 13; FALL]

3.14 报警复位

命令: resetErrorId_IFace()

发送: [resetErrorId_IFace(); id = 14]

接收: 成功[id = 14; Ok] 失败[id = 14; FALL]

3.15 获取报警错误码

命令: getErrorId_IFace()

发送: [getErrorId_IFace(); id = 15]

接收: 成功[id = 15; Ok; 207] (此处 207 是错误码) 失败[id = 15; FALL]

3.16 程序运行

命令: `startRun_IFace()`

发送: `[startRun_IFace(); id = 16]`

接收: 成功[id = 16; Ok] 失败[id = 16; FALL]

注意: 只对示教器程序有效, 对 3d 接口的运动指令无效。3d 接口指令一旦停止, 需要指令重发一次, 不能用 StartRun 把暂停的 3d 接口运动恢复。

3.17 程序停止

命令: `stopRun_IFace()`

发送: `[stopRun_IFace(); id = 17]`

接收: 成功[id = 17; Ok] 失败[id = 17; FALL]

说明: stop 指令停止并清空命令队列

3.18 加载程序文件

命令: `loadUserPrjProg_IFace(std::string prjName, std::string proName)`

prjName: 工程名

proName: 程序文件名

发送: `[loadUserPrjProg_IFace("estun", "main"); id = 18]`

接收: 成功[id = 18; Ok] 失败[id = 18; FALL]

3.19 注销工程

命令: `UnloadUserPrj_IFace(std::string prjName)`

prjName: 工程名

发送: `[UnloadUserPrj_IFace("estun"); id = 19]`

接收：成功[id = 19; Ok] 失败[id = 19; FALL]

3.20 注销程序

命令：UnloadUserProg_IFace(std::string prjName, std::string progName)

prjName 工程名

progName 程序文件名

发送：[UnloadUserProg_IFace("estun", "main"); id = 20]

接收：成功[id = 20; Ok] 失败[id = 20; FALL]

3.21 判断机器人是否在动作

命令：IsRobotMoving_IFace()

发送：[IsRobotMoving_IFace(); id = 21]

接收：成功[id = 21; Ok; 0] (0-stop 1-moving) 失败[id = 21; FALL]

3.21 判断是否有程序加载

命令：IsProgramLoaded_IFace()

发送：[IsProgramLoaded_IFace(); id = 21]

接收：[id = 21; Ok]有程序加载；[id = 21; FALL]没有程序加载。

3.22 设置 PC 指针

命令：SetPc_IFace(int index)

index：程序行数

发送：[SetPc_IFace(6); id = 22]

接收：成功[id = 22; Ok] 失败[id = 22; FALL]

3.23 伺服使能控制

命令: SetMotServoStatus_IFace(int sts_value)

sts_value: 0 代表关掉使能 1 代表打开使能

发送: [SetMotServoStatus_IFace(1); id = 23]

接收: 成功[id = 23; Ok] 失败[id = 23; FALL]

3.24 获取当前伺服使能状态

命令: GetMotServoStatus_IFace()

发送: [GetServoSts_IFace(); id = 24]

接收: 成功[id = 24; Ok; 1] (0 代表未上使能 1 代表上使能) 失败[id = 24; FALL]

3.25 设置手动的运行坐标系

命令: SetCoordType_IFace(int type)

type: 0-关节坐标系; 1-世界坐标系; 2-工具坐标系; 3-用户坐标系。

发送: [SetCoordType_IFace(0); id = 25]

接收: 成功[id = 25; Ok] 失败[id = 25; FALL]

3.26 获取当前点动的坐标系

命令: GetCoordType_IFace()

发送: [GerCurCoordType_IFace(); id = 26]

接收: 成功[id = 26; Ok; 1] (0 代表关节坐标系; 1 代表世界坐标系; 2 代表工具坐标系; 3 代表用户坐标系) 失败[id = 26; FALL]

3.27 单轴点动

命令: JogMotion_IFace(int axis_id, int move_dir)

axis_id: 控制的轴号或坐标方向

move_dir: 轴运动的方向

发送: [JogMotion_IFace(1,1); id = 27]

接收: 成功[id = 27; Ok] 失败[id = 27; FALL]

3.28 停止当前点动

命令: JogMotionStop_IFace()

发送: [JogMotionStop_IFace(); id = 28]

接收: 成功[id = 28; Ok] 失败[id = 28; FALL]

3.29 获取全局速度值

命令: getGoableSpeed_IFace()

发送: [getGoableSpeed_IFace(); id = 29]

接收: 成功[id = 29; Ok; 20] (当前全局速度是 20) 失败[id = 29; FALL]

3.30 多轴定点运动(带速度参数无过渡参数)

命令: MoveToSelectPoint_IFace(int move_type, std::string point_pos, std::string param)

move_type: 运动类型, 1 代表 MoveJ , 2 代表 MoveL (当第三个参数中的高度不为零时, 有一个抬起动作, 运动类型是 movj 的就是轴抬起运动, 运动类型是 movl 的就是直线抬起)

point_pos: 各轴的运动坐标, 以下划线为分隔。(使用时请注意区分把 movej 的坐标与 movel 的坐标) 对于 scara 而言, 仅需输入四个坐标。

param: (高度_速度) 高度参数是机器人抬起特定高度, 移动到目标点的上方或下方, 再运动到目标点, 默认存在过渡, 类型是绝对过渡, 参数是 50, 实际速度是全局速度百分比与设定速度值的乘积。

发送: [MoveToSelectPoint_IFace(1, "0_0_0_0", "0_50"); id = 30]

接收:

成功[FeedMovFinish: 30]或[ActMovFinish: 30]或[RobotStop: 30]或
[SafeDoorIsOpen:30]

失败 [id = 30; FAIL]

3.31 多轴定点运动(带速度参数和过度参数)

命令: MoveToSelectPointb_IFace(int move_type, std::string
point_pos, std::string Param)

move_type: 运动类型, 1 代表 MoveJ 2 代表 MoveL(当第三个参数中的高度不为零时, 有一个抬起动作, 运动类型是 movj 的就是轴抬起运动, 运动类型是 movl 的就是直线抬起)

point_pos: 各轴的运动坐标, 以下划线为分隔。(使用时请注意区分把 movej 的坐标与 movel 的坐标) 对于 scara 而言, 仅需输入四个坐标。

param: (高度_速度_过度值) 没有单位纯数字, 高度参数是机器人抬起特定高度, 移动到目标点的上方或下方, 再运动到目标点, 过渡是绝对过渡, 过渡值是参数的过渡值, 实际速度是全局速度百分比与设定速度值的乘积。

发送: [MoveToSelectPointb_IFace(1, "0_0_0_0", "0_50_100"); id = 31]

接收:

成功[FeedMovFinish: 31]或[ActMovFinish: 31]或[RobotStop: 31]或
[SafeDoorIsOpen: 31]

失败 [id = 31; FAIL]

3.32 多轴运动停止

命令: StopDestPosMotion_IFace()

发送: [StopDestPosMotion_IFace(); id = 32]

接收: 成功[id = 32; Ok] 失败[id = 32; FALL]

3.33 示教指定点

命令: TeachSelectPoint_IFace(std::string PName, int scope_type)

PName: 位置变量名

scope_type: 变量作用域

发送: [TeachSelectPoint_IFace("P10", 2); id = 46]

接收: 成功[id = 46; Ok] 失败[id = 46; FALL]

3.34 寻位运动功能

命令: MoveWithSearch_IFace(int move_type, std::string point_pos, std::string Param)

move_type: 运动类型, 1 代表 MoveJ 2 代表 MoveL

point_pos: 各轴的运动坐标, 以下划线为分隔。(使用时请注意区分把 movej 的坐标与 movel 的坐标) 对于 scara 而言, 仅需输入四个坐标。

param: (速度值_过度值_触发类型_触发编号_触发值) 没有单位纯数字,, 过渡是绝对过渡, 过渡值是参数的过渡值, 实际速度是全局速度百分比与设定速度值的乘积。

触发类型: 1: TorqTrig 力矩触发 2: Inputing 外部 IO 触发

触发编号: 是 IO 的端口号

触发值: 对于力矩触发, 是电机额定扭矩的千分比; 对于外部 IO 触发, 是 IO 的高低电平

发送:

[MoveWithSearch_IFace(1, "0_0_0_0_90_0_0", "10_20_2_10_1"); id=34]

接收:

成功[FeedMovFinish: 34]或[ActMovFinish: 34]或[RobotStop: 34]或
[SafeDoorIsOpen: 34]

失败 [id = 34; FAIL]

3.35 设置负载变量

命令: SetPayload_IFace(int scope, std::string payloadName)

scope: 域, 0:表示系统, 1 表示全局, 2 表示工程, 3 表示程序

payloadName: 负载变量名

发送: [SetPayload_IFace(1, "PAYLOAD0");id=35]

接收: 成功[id = 35;0k] 失败[id = 35;FAIL]

3.36 选择工具坐标系

命令: SetTool_IFace(int scope, std::string name)

scope: 0:表示系统, 1 表示全局, 2 表示工程, 3 表示程序

name: 工具坐标系名

发送: [SetTool_IFace(1, "TOOL0");id=36]

接收: 成功[id= 36;0k] 失败[id=36;FAIL]

3.37 选择用户坐标系

命令: SetCoord_IFace(int scope, std::string name)

scope: 0:表示系统, 1 表示全局, 2 表示工程, 3 表示程序

name: 用户坐标系名

发送: [SetCoord_IFace(0, "USERCOORD0");id=37]

接收: 成功[id = 37;0k] 失败[id = 37;FAIL]

3.38 设置机器人报错停止

命令: SetRTtoErr_IFace(std::string strvalue , int errNum)

strvalue: 自定义的报警信息

errNum: 自定义的报警号

发送: [SetRTtoErr_IFace("testerror100", 9999); id=38]

接收: 成功[id = 38;Ok] 失败[id = 38;FAIL]

3.39 GetCurWPosV3 获得位置带 CFG 值

发送: [GetCurWPosV3(); id=3]

反馈: [Id = 3; Ok; 0 0 0 0 0 0 2304.000 0.0000 2025.500 0.0000 90.000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000]

说明: Ok 之后的分别代表机器人世界坐标 mod cf1 cf2 cf3 cf4 cf5 cf6 x y z a b c 的值(具体意义参看机器人手册), 中间是空格隔开

3.40 GetVarV3 获得变量值

1: 获得整型数据

发送: [GetVarV3(1, "INT0", 1); id=3]

反馈: [Id = 3; Ok; 45], [Id = 3; Fail]

说明: 参数 1: 类型参数 1—整型 (INT) ; 2—浮点型 (REAL) ; 3—APOS 类型;
4—CPOS 类型; 5—字符串类型 (string) ; 6—数组类型; 8—TOOL 变量;
9—USERCOORD 变量

参数 2: 变量名

参数 3: 变量域

Ok 之后的为获取到的值

scope: 0:表示系统, 1 表示全局, 2 表示工程, 3 表示程序

2: 获得浮点型数据 REAL

发送: [GetVarV3(2,"RealTimeX",1);id=3]

反馈: [Id = 3; Ok; 954.550]

说明: 同上

3: 获得 APOS 数据类型

发送: [GetVarV3(3,"AP0",3);id=3]

反馈: [Id = 3; Ok; 3.000 11.960 58.983 0.0000 19.056 183.000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000], [Id = 3; Fail]

说明: 参数同上

Ok 之后的值为 a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 a12 a13 a14 a15 a16

4: 获得 CPOS 数据类型

发送: [GetVarV3(4,"CP0",3);id=3]

反馈: [Id = 3; Ok; 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.000 1300.000
-68.139 -300.000 0.0000 0.0000 180.000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000]

说明: 参数同上

Ok 之后的分别代表机器人世界坐标 mod cf1 cf2 cf3 cf4 cf5 cf6 x y z a b c 的值(具体意义参看机器人手册), 中间是空格隔开

5: 获得 String 类型

发送: [GetVarV3(5,"STRING0",1);id=3]

反馈: [Id = 3; Ok; vvffg], [Id = 3; Fail]

说明: 同上

6: 获得数组类型变量

发送: [GetVarV3(6,"bbb",1);id=3]

反馈: [Id = 3; Ok; 95.500 38.500 83.500 23.400 83.500 94.500], [Id = 3; Fail]

说明: Ok 后是 array 的值

8: 获得 TOOL 变量

发送: [GetVarV3(8,"TOOL0",1);id=1]

反馈: [id = 1; Ok; 1.000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000], [Id = 1;
Fail]

说明: Ok 后是 tool 的值: id x y z a b c M Mx My Mz lxx lyy lzz lxy lxz lyz

9: 获得 Usercoord 变量

发送: [GetVarV3(9,"USERCOORD",1);id=1]

反馈: [id = 1; Ok; 1.000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000], [Id = 1;
Fail]

说明: Ok 后是 UserCoord 的值: id x y z a b c

3.41 SetVarV3 设置变量值

1: 设置整型数据

发送: [SetVarV3(1,"INT0","65",1);id=3]

反馈: [Id = 3; Ok], [Id = 3; Fail]

说明:

参数 1: 类型参数 1--整型 (INT) ; 2—浮点型 (REAL) ; 3—APOS 类型;
4—CPOS 类型 (不含 cfg 值, 使用当前 cfg 值) ; 5—字符串类型 (string) ; 6—
数组类型; 7-- CPOS 类型 (含 cfg 值) ; 8—Tool 变量类型 (不带 payload 参数) ;
9—UserCoord 变量类型; 10—Tool 变量类型 (带 payload 参数)

参数 2: 变量名

参数 3: 变量值, string 类型

参数 4: 变量域

Ok 之后的为获取到的值

scope: 0:表示系统, 1 表示全局, 2 表示工程, 3 表示程序

2: 设置浮点型数据 REAL

发送: [SetVarV3(2,"RealTimeX","234.44",1);id=3]

反馈: [Id = 3; Ok], [Id = 3; Fail]

说明: 同上

3: 设置 APOS 数据类型

发送:

[SetVarV3(3,"AP0","13.000_111.960_158.983_10.0200_191.056_18.000",3);id=3]

反馈: [Id = 3; Ok], [Id = 3; Fail]

说明:

参数 1、2 同上

参数 3: 1_a2_a3_a4_a5_a6_a7_a8_a9_a10_a11_a12_a13_a14_a15_a16, 值个数最大 16 个, 少的参数默认补 0, 必须是浮点数;

4: 设置 CPOS 数据类型

发送:

[SetVarV3(4,"CP0","1_13.000_111.960_158.983_10.0200_191.056_18.000",3);id=3]

反馈: [Id = 3; Ok], [Id = 3; Fail]

说明:

参数同上

参数 3: mod_x_y_z_a_b_c_a7_a8_a9_a10_a11_a12_a13_a14_a15_a16 的值个数最大 16 个, 最少 7 个即 (mod_x_y_z_a_b_c), 少的参数默认补 0, 必须是浮点数。

5: 设置 String 类型

发送: [SetVarV3(5,"STRING0","cvdsdf",1);id=3]

反馈: [Id = 3; Ok], [Id = 3; Fail]

说明: 同上

6: 设置数组类型变量

发送: [SetVarV3(6,"bbb","1.1_2.2_3.3_4.4_5.5",1);id=3]

反馈: [Id = 3; Ok], [Id = 3; Fail]

说明:

参数 1、2 同上

参数 3: 必须是浮点数

7: 设置 CPOS 含有 CFG 值

发送: [SetVarV3

(7,"CP0","1_2_3_4_5_6_7_13.000_111.960_158.983_10.0200_191.056_18.000",3);id=3]

反馈: [Id = 3; Ok], [Id = 3; Fail]

说明:

参数 1、2 同上

参数 3: mod_cf1_cf2_cf3_cf4_cf5_cf6_x_y_z_a_b_c_a7_a8_a9_a10_a11_a12_a13_a14_a15_a16 的值个数最大 23 个, 最少 13 个即 (mod_cf1_cf2_cf3_cf4_cf5_cf6_x_y_z_a_b_c), 少的参数默认补 0, 必须是浮点数。

8: 设置 TOOL 变量（不带负载参数）

发送: [SetVarV3(8,"TOOL0","1_2_3_4_5_6_",1);id=1]

反馈: [Id = 3; Ok], [Id = 3; Fail]

说明:

参数 1、2 同上

参数 3: x_y_z_a_b_c 的值个数是 6 个

9: 设置 USERCOORD 变量

发送: [SetVarV3(9,"USERCOORD","1_2_3_4_5_6",1);id=1]

反馈: [Id = 1; Ok], [Id = 1; Fail]

说明: 参数 1、2 同上

参数 3: x_y_z_a_b_c 的值个数是 6 个

10: 设置 TOOL 变量（带负载参数）

发送:

[SetVarV3(8,"TOOL0","1_2_3_4_5_6_7_8_9_10_11_12_13_14_15_16",1);id=1]

反馈: [Id = 3; Ok], [Id = 3; Fail]

说明:

参数 1、2 同上

参数 3: x_y_z_a_b_c_M_Mx_My_Mz_lxx_lyy_lzz_lxy_lxz_lyz 的值个数是 16 个

11: 设置 TOOL 变量（不带负载参数）写文件

发送: [SetVarV3(11,"TOOL0","1_2_3_4_5_6_",1);id=1]

反馈: [Id = 3; Ok], [Id = 3; Fail]

说明:

参数 1、2 同上

参数 3: x_y_z_a_b_c 的值个数是 6 个

12: 设置 USERCOORD 变量, 写文件

发送: [SetVarV3(12,"USERCOORD","1_2_3_4_5_6",1);id=1]

反馈: [Id = 1; Ok], [Id = 1; Fail]

说明: 参数 1、2 同上

参数 3: x_y_z_a_b_c 的值个数是 6 个

13: 设置 TOOL 变量 (带负载参数), 写文件

发送:

[SetVarV3(13,"TOOL0","1_2_3_4_5_6_7_8_9_10_11_12_13_14_15_16",1);id=1]

反馈: [Id = 3; Ok], [Id = 3; Fail]

说明:

参数 1、2 同上

参数 3: x_y_z_a_b_c_M_Mx_My_Mz_lxx_lyy_lzz_lxy_lxz_lyz 的值个数是 16 个

3.42 MovePointV3 机器人运动接口

1: 关节运动关节位置

发送: [MovePointV3(1,"0.0_0.0_0.0_0.0_90.0_0.0","", "0_100_100");id =3]

反馈:

成功[FeedMovFinish: 34]或[ActMovFinish: 34]或[RobotStop: 34]或
[SafeDoorIsOpen: 34]

失败 [id = 34; FAIL]

说明：

参数 1：是运动类型，1—关节运动关节位置；2--坐标系运动坐标系位置；3--关节运动坐标系位置

参数 2：坐标值，必须是浮点数

参数 3：cfg 值，没有即为空”， 坐标系位置中使用。示

例：“mode_cf1_cf2_cf3_cf4_cf5_cf6”， 如：“1_0_0_0_0_0”， 都为 0 即为空”。（针对 scara 存在在 cf4 为 99 的情况，表示 4 轴最短旋转路径）

参数 4：param：（高度_速度_过度参数_到位精度(可选参数)）

高度参数：在关节运动关节位置即类型 1 下，高度参数无效；高度参数是机器人抬起特定高度，移动到目标点的上方或下方，再运动到目标点，高度参数为 0 时，则无抬起动作，直接运动到目标点。

速度参数：是全局速度百分比与设定速度值的乘积。

过渡参数，类型是绝对过渡。

到位精度(定制功能针对高精度要求，联系 estun)：0 或者没有设置该参数→给定到位，非零→实际到位(该值为到位精度例如 0.05 表示到位精度是 0.05)，过渡值不为 0 时，到位精度参数无效，默认给定到位。

2：坐标系运动坐标系位置

发送：[MovePointV3(2,"2100.0_-100.0_1700.0_0.0_0.0_0.0","","10_100_100");id=3]

反馈：

成功[FeedMovFinish: 3]或[ActMovFinish: 3]或[RobotStop: 3]或
[SafeDoorIsOpen: 3]

失败 [id = 3; FAIL]

说明：参数同上

3: 关节运动坐标系位置

发送: [MovePointV3(3,"2100.0_100.0_1700.0_0.0_0.0_0.0","", "10_100_100");id=3]

反馈: 成功[FeedMovFinish: 3]或[ActMovFinish: 3]或[RobotStop: 3]或[SafeDoorIsOpen: 3]

失败 [id = 3; FAIL]

说明: 参数同上

3.43 GetToolV3 获得当前工具

发送: [GetToolV3();ID=3]

反馈: [id = 3; Ok; TOOL0, 1], [Id = 3; Fail]

说明: Ok 后面的是当前的工具变量 TOOL0, 和工具变量的域 (1-全局) 工具是系统变量, 只能是全局变量

3.44 GetUserCoordV3 获得当前用户坐标

发送: [GetUserCoordV3();ID=3]

反馈: [id = 3; Ok; USERCOORD, 1], [Id = 3; Fail]

说明: Ok 后面的是当前的用户坐标系变量 USERCOORD, 和用户坐标系变量的域 (1-全局) 用户坐标系是系统变量, 只能是全局变量

3.45 MoveFinsih 运动任务完成信号

执行流程:

1: MovePointV3 执行, 下发运动指令。

2: 机器人执行运动指令, 机器人运动直到结束, 发送[FeedMovFinish:XX](指给定到位)或者[ActMovFinish:XX](指实际到位)。

注意：如果运动被停止或者暂停，表示该运动没有完成是不会发送 [FeedMovFinish:XX]/[ActMovFinish:XX]信号的。只有运动真正完成，才发送 [MoveFinsih:XX]信号

3.46 GetSoftLimits 获得当前机器人的轴软限位

发送：[GetSoftLimits(6);id=3]

反馈：[id = 3; 0k; -10 20]、 [Id = 3; Fail]

说明： 参数 1：轴号 (> 0)

0k 后面的是当前轴的关节负软限位和关节正软限位，单位° 度。

3.47 SetSoftLimits 获得当前机器人的轴软限位

发送：[SetSoftLimits (6,-10,20);id=3]

反馈：[id = 3; 0k]、 [Id = 3; Fail]

说明： 参数 1：轴号 (> 0)

参数 2：关节负软限位，单位° 度

参数 3：关节正软限位，单位° 度

3.48 setMultiIOValue 设置多 IO 接口

命令格式：setMultiIOValue_Face ("lostrValue",loStartIndex,lotype,loNum)

参数 1：lostrValue：各个 IO 的端口值。注意这里的端口值必须为浮点数

参数 2：loStartIndex：起始的 IO 端口号

参数 3：lotype：IO 类型(14 代表虚拟 Aout,13 代表虚拟 Ain,12 代表虚拟 Dout,11 代表虚拟 Din, 2 代表实际 Aout, 1 代表实际 Dout)

参数 4：loNum：IO 数量

send:

```
[setMultiIOValue_Face("1.1_2.2_3.3_4.4_5.5_6.6_7.7_8.8_9.9_10.0",1,14,10);id=11]
```

ack: 成功[id = 11; Ok]

2、失败[id =11;Fail]

3.49 GetMultiIOValue 读取多 IO 口值

命令格式: getMultiIOValue_Face(loStartIndex,lotype,loNum)

参数 1: loStartIndex: 起始的 IO 端口号

参数 2: lotype: IO 类型(14 代表虚拟 Aout,13 代表虚拟 Ain,12 代表虚拟 Dout,11 代表虚拟 Din, 2 代表实际 Aout, 1 代表实际 Dout, 0 代表实际 Din)

参数 3: loNum: IO 数量

send:

```
[getMultiIOValue_Face (1,14,10);id=12]
```

ack: 1、成功[id = 12; Ok;1.1,2.2,3.3,4.4,5.5,6.6,7.7,8.8,9.9,10.0]

1、失败[id =12; Fail]

3.50 Clear3dCmds 清空 3d 接口缓存的命令

命令: Clear3dCmds()

发送: [Clear3dCmds(); id = 19]

接收: 成功[id = 19; Ok]

第 4 章 视觉配置介绍

视觉配置说明:

视觉配置位于 通用设置->视觉设置中, 配置界面如图 4-1 所示:

图 4-1 视觉设置界面



相机设备类型：点击选择 2D 或 3D，这里选择 3D。

摄像头 IP：该属性值为相机的 IP 地址，默认连接的相机 IP 为 192.168.6.230。上电之后，机器人控制器就会作为客户端连接该 IP 地址的服务器。

摄像头端口号：该属性值是作为服务器的相机的端口号，默认端口号为 5000。

视觉类型：点击选择静态或动态，这里选择静态。

第 5 章 三维视觉调试

该部分介绍作为视觉客户端的机器人的视觉功能调试。SocketTest 作为调试服务器。

5.1 通信心跳的说明

外部相机作为服务器，当与机器人连接上了之后，机器人在没有发送数据的情况下需要每隔 18 秒发送一次”(16 进制 0x20，空格符)字符给服务器心跳。

如果机器人发送数据失败，会断开当前连接，重新连接服务器。

5.2 SocketTest 调试助手

SocketTestDlg 是卓岚公司开发的，综合的 TCP、UDP 协议的调试软件。自主控制器机器人视觉功能是选择的是 TCP 协议。该调试软件界面如图 5-1 下：

图 5-1 视觉配置界面



5.3 调试步骤

- 1、将本地电脑连通机器人。
- 2、检查机器人连接的 IP 和端口号，将本地电脑 IP 改为机器人连接的 IP，启动 SocketTestDlg.exe，设置工作模式为 TCP 服务器，本地端口为机器人设置的端口。
- 3、机器人每隔 18 秒需要发送一个 “ ” (16 进制 0x20,空格符) 字符，给助手心跳。
- 4、然后在 socketTest 中发送上面的各种命令，就会收到对应的应答。

例：获取当前关节坐标

卓岚TCP&UDP调试工具-ZLAN专业的串口联网方案提供商-http://www.zlmcu.com

通信设置

工作模式: TCP服务器

本地端口: 5000 0表示任意

UDP目的IP/端口随对方变化 ☐

目的IP: 192.168.12.234

目的端口: 4567

所在组播组: 230.90.76.1

关闭

接收区设置

☐ 十六进制接收

☐ 选择接收文件/停止接收

清除窗口

发送区设置

☐ 十六进制发送 (格式01 02)

☐ 每隔 500 毫秒发送

发送接收方式: 信息模式

修改发送文件名

接收信息 接收缓冲大小: 2000 字节

[id = 1; 0k; 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000]

发送信息 (ctrl+Enter输入回车(0x0d,0x0a); \r输入0x0d, \n输入0x0a)

[GetCurJPos();id=1]

发送

停止

清空信息

报告 ☐ 关闭报告

316.295936(s) :TCP rcv from socket 192.168.12.98:53620

298.239340(s) :TCP rcv from socket 192.168.12.98:53620

298.236222(s) :TCP socket 192.168.12.98:53620 Send OK!

291.837575(s) :The NO. 0 TCP socket 192.168.12.98:53620 is accepted!

本地IP: 192.168.74.1

高级设置

计数和校验和 TXD: 19 1228 RXD: 56 798

重新计数