JK-Robot

软件用户手册 V2.0

					XT-DDD-SUM	
标记	数量	修改单号		签字	日期	
编制			会签			软件用户手册模板
校对			标检			共 41 页 第 1 页
审核			批准			敬科（深圳）机器人科技有限公司
会签						

修订记录

版本号	修订状态	简要说明修订内容和范围	修订日期	修订人	批准日期

注：修订记录在体系文件发布后换版时使用，修订状态栏填写：A—增加，M—修改，D—删除

目 次

1 范围

1.1 标识

简要说明软件的标识号、名称、缩略名、版本号。

a) 标识号:;

b) 标题:《JK-Robot 软件使用手册》;

c) 缩略名:;

d) 版本号：20190710V1;

1.2 系统概述

本文档为 JK 机器人 SDK 使用说明。该软件使用手册向用户详细描述了开发 JK 机器人的 API 使用方法。结合该手册，用户可以轻松，灵活的开发机器人控制算法。

该 SDK 包使用 C++语言开发，可在 LINUX， Window 系统进行使用。本软件为初次开发，无维护历史，未设置保障机构。

a) 需方：

b) 用户：

c) 开发方：敬科（深圳）科技有限公司

1.3 文档概述

本文档主要用于给出软件用途，使用环境以及使用方法。通过该文档，使用人员应能够掌握该软件的使用方法。

本文档主要包含以下内容：

第一章： 范围

第二章： 软件综述

第三章： 软件入门

第四章： 快速开发历程

2 软件综述：

本文档为 JK5 机器人 SDK 使用说明。开发者使用 TCP/IP 协议与机器人通信，其中机器人控制器为服务器短，上位机为客户端。连接时，请设置机器人控制器端 IP 与上位机控制端 IP 相同。例如设置机器人控制器 IP 为 192.168.0.150 ，上位机控制器设置为 192.168.0.100（相同网段即可，网关默认）。机器人控制器通信接口默

认为 8800。

3 软件入门

3.1 通信协议

1、建立连接：

Client->setup(“192.168.0.150”, 8800);

2、通信协议：

通信包括命令和输入参数。通信协议使用**命令头 + 参数**方式进行。

详细命令通信见下表：

数据类型为下位机将处理的数据类型。但是由于使用 TCP/IP 协议，发送与接收均为字符串。请上位机程序自行相应解析。

布尔型发送时请发送 1 代替 true，0 代替 false。

序号	命令头	输入/输出参数个数	数据类型	含义
1	GETJ	6	Double	获取关节位置
2	GETJV	6	Double	获取关节速度
3	GETP	6	Double	获取末端位置
4	GETM	1	Int	获取机械臂当前模式 0 关节速度模式 1 关节位置模式 2 末端速度模式 3 末端位置模式 4 手柄控制模式 5 视觉伺服模式 6 力控制模式 7 示教再现模式

5	GETPVR	1	Double	获取末端速度比例
6	GETJVR	1	Double	获取关节速度比例
7	GETSF4	1	Bool	获得程序运行状态
8	GETJF	1	Bool	获取关节规划到位标志位状态
9	GETPF	1	Bool	获取笛卡尔空间规划位置到位标志位状态
10	GETCF	1	Bool	获取位置标志位状态（圆弧）
11	GETFTF	1	Bool	获取受力超限标志位状态
12	GETFORCE	6	Double	获取力限制检测阈值
13	GETCFFF	6	Double	获取期望力/力矩
14	GETCIC	1	Bool	获取阻抗控制方式
15	GETKX	6	Double	获取阻抗刚度
16	GETKV	6	Double	获取阻抗刚度（阻尼）
17	GETP7E	3	Double	获取6关节原点到末端旋转点的矢径坐标
18	GETLP	4	Double	获得负载惯性参数。 1质量 2-4质量*质心位置
19	SETJ	6	Double	设置关节位置

20	SETJV	6	Double	设置关节速度
21	SETM	1	Int	设置机械臂当前模式 0 关节速度模式 1 关节位置模式 2 末端速度模式 3 末端位置模式 4 手柄控制模式 5 视觉伺服模式 6 力控制模式 7 示教再现模式
22	SETJVR	1	Double	设置关节速度比例
23	SETSF4	1	Bool	设置运行状态
24	SETP	6	Double	设置末端位置
25	SETPVR	1	Double	设置末端速度比例
26	SETENABLE	1	Bool	机器人上电使能设置
27	SETFM	2	Int	设置坐标系 0 0 基坐标绝对运动 0 1 基坐标系相对运动 1 0 末端坐标系绝对运动 1 1 末端坐标相对运动
28	SETLP	4	Double	设置负载惯性参数。 1 质量 2-4 质量*质心位置

29	SETVWL	6	Double	设置虚拟墙上限和下限
30	SETVWA	1	Double	设置虚拟墙绕 z 轴旋转角
31	SETVWE	1	Bool	设置虚拟墙启用
32	SETJVS	1	Double	设置关节速度限幅
33	SETTOOLDATA	3	Bool	设置 TOOL 输入
34	SETBOXDATA	16	Bool	设置 BOX 输入
35	SETLEVEL	1	Bool	设置电压
36	GETTOOLDATA	3	Bool	获取 TOOL 输出
37	GETBOXDATA	16	Bool	获取 BOX 输出

3.2 通用函数介绍

1：获取关节位置

上位机客户端发送：GETJ

服务器端（JK5 机器人）：将发送 str 字符串 GETJ 12.0 34.0 22.0 12.0 10.0 20.0 。GETJ 为数据头，后面 6 个为关节当前角度信息。

程序如下：

上位机客户端	tcp->Send(cmdStr);
	string str = tcp->receive(1000);
	cout << "recStr: " << str << endr;

下位机返回：	if (stricmp(Command, "GETJ") == 0) {
	rtn = pRobotComm->GetJoints();
	}
	int CRobotComm::GetJoints()
	{

	double joint[6];
	char str[1024] = "";
	……
	……
	sprintf(str, "GETJ %lf %lf %lf %lf %lf %lf", joint[0], joint[1], joint[2], joint[3], joint[4], joint[5]);
	return tcpServer.Send(str, strlen(str));
	}

2：获取关节速度

上位机客户端发送：GETJV

服务器端（JK5 机器人）：将发送 str 字符串 GETJV 12.0 34.0 22.0 12.0 10.0 20.0 。GETJV 为数据头，后面 6 个为关节当前角速度信息。

程序如下：

上位机客户端	tcp->Send(cmdStr);
	string str = tcp->receive(1000);
	cout << "recStr: " << str << endr;

下位机返回：	else if (stricmp(Command, "GETJV") == 0) {
	rtn = pRobotComm->GetJointVel();
	}
	int CRobotComm::GetJointVel()
	{
	double JointVel[6];
	char str[1024] = "";
	……
	……
	sprintf(str, "GETJV %lf %lf %lf %lf %lf %lf", JointVel[0], JointVel[1], JointVel[2], JointVel[3], JointVel[4], JointVel[5]);

	return tcpServer.Send(str, strlen(str)); }

3：获取末端位置

上位机客户端发送：GETP

服务器端（JK5 机器人）：将发送 str 字符串 GETP 12.0 34.0 22.0 12.0 10.0 20.0 。GETP 为数据头，后面 6 个数据中前 3 个为末端位置信息，后 3 个为末端姿态角信息。

程序如下：

上位机客户端	tcp->Send(cmdStr);
	string str = tcp->receive(1000);
	cout << "recStr: " << str << endr;

下位机返回：	else if (stricmp(Command, "GETP") == 0) {
	rtn = pRobotComm->GetPos();
	}
	int CRobotComm::GetPos()
	{
	double Pos[6];
	char str[1024] = "";
	……
	……
	sprintf(str, "GETP %lf %lf %lf %lf %lf %lf ", Pos[0], Pos[1], Pos[2], Pos[3], Pos[4], Pos[5]);
	return tcpServer.Send(str, strlen(str));
	}

4：获取模式

上位机客户端发送：GETM

服务器端（JK5 机器人）：将发送 str 字符串 GETM 4 。GETM 为数据头，后面 1 个为机器人当前控制模式信息。

程序如下：

上位机客户端	tcp->Send(cmdStr);
	string str = tcp->receive(1000);
	cout << "recStr: " << str << endr;

下位机返回：	else if (stricmp(Command, "GETM") == 0) {
	rtn = pRobotComm->GetMode();
	}
	int CRobotComm::GetMode()
	{
	int mode = 0;
	char str[1024] = "";
	……
	……
	sprintf(str, "GETM %d", mode);
	return tcpServer.Send(str, strlen(str));
	}

5：获取末端速度比例

上位机客户端发送：GETPVR

服务器端（JK5 机器人）：将发送 str 字符串 GETPVR 0.4 。GETPVR 为数据头，后面 1 个为末端速度比例信息。

程序如下：

上位机客户端	tcp->Send(cmdStr);
	string str = tcp->receive(1000);
	cout << "recStr: " << str << endr;

下位机返回：	else if (stricmp(Command, "GETPVR") == 0) {
	rtn = pRobotComm->GetPosVelRatio();
	}
	int CRobotComm::GetPosVelRatio()
	{

```
        double PosVelRatio;
        char str[1024] = "";
        ……
……
        sprintf(str, "GETJVR %lf", PosVelRatio);
        return tcpServer.Send(str, strlen(str));
}
```

6：获取关节速度比例

上位机客户端发送：GETJVR

服务器端（JK5 机器人）：将发送 str 字符串 GETJVR 0.4 。GETJVR 为数据头，后面 1 个为关节速度比例信息。

程序如下：

上位机客户端

```
tcp->Send(cmdStr);
string str = tcp->receive(1000);
cout << "recStr: " << str << endr;
```

下位机返回：

```
else if (stricmp(Command, "GETJVR") == 0) {
        rtn = pRobotComm->GetJointVelRatio();
}
int CRobotComm::GetJointVelRatio()
{
        double jointVelRatio;
        char str[1024] = "";
        ……
……
        sprintf(str, "GETJVR %lf", jointVelRatio);
        return tcpServer.Send(str, strlen(str));
}
```

7：获取运行状态

上位机客户端发送：GETSF4

服务器端（JK5 机器人）：将发送 str 字符串 GETSF4 1 。GETSF4 为数据头，后面 1 个为当前运行状态信息。

程序如下：

上位机客户端	tcp->Send(cmdStr);
	string str = tcp->receive(1000);
	cout << "recStr: " << str << endr;

下位机返回：	else if (stricmp(Command, "GETSF4") == 0) {
	rtn = pRobotComm->GetSwitchFlag4();
	}
	int CRobotComm::GetSwitchFlag4()
	{
	bool flag;
	char str[1024] = "";
	……
	……
	sprintf(str, "GETSF4 %d", (int)flag);
	return tcpServer.Send(str, strlen(str));
	}

8：获取关节标志位状态

上位机客户端发送：GETJF

服务器端（JK5 机器人）：将发送 str 字符串 GETJF 1 。GETJF 为数据头，后面 1 个为关节规划到位标志位状态信息。

程序如下：

上位机客户端	tcp->Send(cmdStr);
	string str = tcp->receive(1000);
	cout << "recStr: " << str << endr;

下位机返回：	else if (stricmp(Command, "GETJF") == 0) {
	rtn = pRobotComm->GetJointFlag();
	}
	int CRobotComm::GetJointFlag()
	{
	bool ocjm_flag;
	char str[1024] = "";
	……
	……
	sprintf(str, "GETJF %d", ocjm_flag);
	return tcpServer.Send(str, strlen(str));
	}

9：获取位置标志位状态

上位机客户端发送：GETPF

服务器端（JK5 机器人）：将发送 str 字符串 GETPF 1 。GETPF 为数据头，后面 1 个为笛卡尔空间规划位置到位标志位状态信息。

程序如下：

上位机客户端	tcp->Send(cmdStr);
	string str = tcp->receive(1000);
	cout << "recStr: " << str << endr;

下位机返回：	else if (stricmp(Command, "GETPF") == 0) {
	rtn = pRobotComm->GetPosFlag();
	}
	int CRobotComm::GetPosFlag()
	{
	bool ocpm_flag;
	char str[1024] = "";
	……
	……

	sprintf(str, "GETPF %d", ocpm_flag);
	return tcpServer.Send(str, strlen(str));
	}

10：获取位置标志位状态（圆弧）

上位机客户端发送：GETCF

服务器端（JK5 机器人）：将发送 str 字符串 GETCF 1 。GETCF 为数据头，后面 1 个为笛卡尔空间圆弧运动位置到位标志位状态信息。

程序如下：

上位机客户端	tcp->Send(cmdStr);
	string str = tcp->receive(1000);
	cout << "recStr: " << str << endr;

下位机返回：	else if (stricmp(Command, "GETCF") == 0) {
	rtn = pRobotComm->GetCircleFlag();
	}
	int CRobotComm::GetCircleFlag()
	{
	bool occm_flag;
	char str[1024] = "";
	……
	……
	sprintf(str, "GETCF %d", occm_flag);
	return tcpServer.Send(str, strlen(str));
	}

11：获取受力超限标志位状态

上位机客户端发送：GETFTF

服务器端（JK5 机器人）：将发送 str 字符串 GETFTF 0 。GETFTF 为数据头，后面 1 个为笛卡尔空间受力超限标志位状态信息。

程序如下：

上位机客户端	tcp->Send(cmdStr);
	string str = tcp->receive(1000);
	cout << "recStr: " << str << endr;

下位机返回：	else if (stricmp(Command, "GETFTF") == 0) {
	rtn = pRobotComm->GetForceTorchFlag();
	}
	int CRobotComm::GetForceTorchFlag()
	{
	bool f_overflow_flag;
	char str[1024] = "";
	……
	……
	sprintf(str, "GETFTF %d", f_overflow_flag);
	return tcpServer.Send(str, strlen(str));
	}

12：获取力限制检测阈值

上位机客户端发送：GETFORCE

服务器端（JK5 机器人）：将发送 str 字符串 GETFORCE 4.6 4.0 4.5 3.2 2.5 2.0 。GETFORCE 为数据头，后面 6 个为笛卡尔空间力限制检测阈值。

程序如下：

上位机客户端	tcp->Send(cmdStr);
	string str = tcp->receive(1000);
	cout << "recStr: " << str << endr;

下位机返回：	else if (stricmp(Command, "GETFORCE") == 0) {
	rtn = pRobotComm->GetForce();
	}
	int CRobotComm::GetForce()
	{
	double Force[6];

	char str[1024] = "";
	……
	……
	sprintf(str, "GETFORCE %lf %lf %lf %lf %lf %lf", Force[0], Force[1], Force[2], Force[3], Force[4], Force[5]);
	return tcpServer.Send(str, strlen(str));
	}

13：获取期望力/力矩

上位机客户端发送：GETCFFF

服务器端（JK5 机器人）：将发送 str 字符串 GETCFFF 3.6 3.0 3.5 2.2 2.0 1.6 。GETCFFF 为数据头，后面 6 个为笛卡尔空间关节期望力/力矩。

程序如下：

上位机客户端	tcp->Send(cmdStr);
	string str = tcp->receive(1000);
	cout << "recStr: " << str << endr;

下位机返回：	else if (stricmp(Command, "GETCFFF") == 0) {
	rtn = pRobotComm->GetCartesianForceFeedForward();
	}
	int CRobotComm::GetCartesianForceFeedForward()
	{
	double CartesianForceFeedForward[6];
	char str[1024] = "";
	……
	sprintf(str, "GETCFFF %lf %lf %lf %lf %lf %lf", CartesianForceFeedForward[0], CartesianForceFeedForward[1], CartesianForceFeedForward[2],

	CartesianForceFeedForward[3],
	CartesianForceFeedForward[4],
	CartesianForceFeedForward[5]);
	return tcpServer.Send(str, strlen(str));
	}

14：获取阻抗控制方式

上位机客户端发送：GETCIC

服务器端（JK5 机器人）：将发送 str 字符串 GETCIC 1 。GETCIC 为数据头，后面 1 个为当前阻抗控制方式信息。

程序如下：

上位机客户端	tcp->Send(cmdStr);
	string str = tcp->receive(1000);
	cout << "recStr: " << str << endr;

下位机返回：	else if (stricmp(Command, "GETCIC") == 0) {
	rtn = pRobotComm->GetCartesianImpedanceControl();
	}
	int CRobotComm::GetCartesianImpedanceControl()
	{
	bool CartesianImpedanceControl;
	char str[1024] = "";
	……
	……
	sprintf(str, "GETCIC %d", CartesianImpedanceControl);
	return tcpServer.Send(str, strlen(str));
	}

15：获取阻抗刚度

上位机客户端发送：GETKX

服务器端（JK5 机器人）：将发送 str 字符串 GETKX 0.6 0.5 0.5 0.3 0.4 0.2 。GETKX 为数据头，后面 6 个为笛卡尔空间阻抗刚度。

程序如下：

上位机客户端	tcp->Send(cmdStr);
	string str = tcp->receive(1000);
	cout << "recStr: " << str << endr;

下位机返回：	else if (stricmp(Command, "GETKX") == 0) {
	rtn = pRobotComm->GetKX();
	}
	int CRobotComm::GetKX()
	{
	double KX[6];
	char str[1024] = "";
	……
	……
	sprintf(str, "GETKX %lf %lf %lf %lf %lf %lf", KX[0], KX[1], KX[2], KX[3], KX[4], KX[5]);
	return tcpServer.Send(str, strlen(str));
	}

16：获取阻抗刚度（阻尼）

上位机客户端发送：GETKV

服务器端（JK5 机器人）：将发送 str 字符串 GETKV 0.6 0.5 0.5 0.3 0.4 0.2。GETKV 为数据头，后面 6 个为笛卡尔空间阻尼。

程序如下：

上位机客户端	tcp->Send(cmdStr);

	string str = tcp->receive(1000); cout << "recStr: " << str << endr;

下位机返回：	else if (stricmp(Command, "GETKV") == 0) {
	rtn = pRobotComm->GetKV();
	}
	int CRobotComm::GetKV()
	{
	double KV[6];
	char str[1024] = "";
	……
	……
	sprintf(str, "GETKV %lf %lf %lf %lf %lf %lf", KV[0], KV[1], KV[2], KV[3], KV[4], KV[5]);
	return tcpServer.Send(str, strlen(str));
	}

17：获取 6 关节原点到末端旋转点

上位机客户端发送：GETP7E

服务器端（JK5 机器人）：将发送 str 字符串 GETP7E 0.2 0.09 0.08 。GETP7E 为数据头，后面 3 个为第 6 关节原点到末端旋转点的矢径坐标。

程序如下：

上位机客户端	tcp->Send(cmdStr);
	string str = tcp->receive(1000);
	cout << "recStr: " << str << endr;

下位机返回：	else if (stricmp(Command, "GETP7E") == 0) {
	rtn = pRobotComm->GetP7E();
	}
	int CRobotComm::GetP7E()
	{
	double p_7_e[3];

	char str[1024] = "";
	……
	……
	sprintf(str, "GETP7E %lf %lf %lf ", p_7_e[0], p_7_e[1], p_7_e[2]);
	return tcpServer.Send(str, strlen(str));
	}

18：获取负载惯性参数

上位机客户端发送：GETLP

服务器端（JK5 机器人）：将发送 str 字符串 GETLP 3.0 0.6 0.27 0.24 。GETLP 为数据头，后面 4 个数据中前 1 个为负载质量信息，后 3 个为负载质量矩信息。

程序如下：

上位机客户端	tcp->Send(cmdStr);
	string str = tcp->receive(1000);
	cout << "recStr: " << str << endr;

下位机返回：	else if (stricmp(Command, "GETLP") == 0) {
	rtn = pRobotComm->GetLoadParas();
	}
	int CRobotComm::GetLoadParas()
	{
	double load_params[4];
	char str[1024] = "";
	……
	……
	sprintf(str, "GETLP %lf %lf %lf %lf", load_params[0], load_params[1], load_params[2], load_params[3]);
	return tcpServer.Send(str, strlen(str));

	}

19：设置关节角位置

上位机客户端发送字符串：SETJ 10.0 10.0 10.0 10.0 10.0 10.0 ， 设置机器人每个关节角为 10 度，

服务器端（JK5 机器人）：接收后会控制每个关节移动，到位成功后，将发送字符串：SETJ ： OK

上位机客户端	Client->send(“SETJ 10 10 10 10 10 10”) string str = Client->receive(1000); cout << "recStr: "<< str<< endr;

下位机返回：	if (stricmp(Command, "SETJ") == 0) {
	rtn =
	pRobotComm->SetJoints(Parameters);
	}
	int CRobotComm::SetJoints(const char
	Parameters[MAX_PARAMETER_LEN][256])
	{
	double jointPos[6];
	for (int i = 0; i < 6; i++) {
	jointPos[i] = atof(Parameters[i]);
	}
	……
	char str[100] = "SETJ OK";
	return tcpServer.Send(str, strlen(str));
	}

20：设置关节速度

上位机客户端发送字符串：SETJV 0.4 0.4 0.4 0.4 0.4 0.4 ， 设置机器人每个期望关节速度为 0.4 弧度，

服务器端（JK5 机器人）：接收后会设置每个关节期望速度，设置成功后，将发送字符串： SETJV ： OK

上位机客户端	Client->send(“SETJV 0.4 0.4 0.4 0.4 0.4 0.4”) string str = Client->receive(1000); cout << "recStr: "<< str<< endr;

下位机返回：	if (stricmp(Command, "SETJV") == 0) {
	rtn =
	pRobotComm->SetJointsVel(Parameters);
	}
	int CRobotComm::SetJointsVel(const char
	Parameters[MAX_PARAMETER_LEN][256])
	{
	double jointVel[6];
	for (int i = 0; i < 6; i++) {
	jointVel[i] = atof(Parameters[i]);
	}
	……
	……
	char str[100] = "SETJV OK";
	return tcpServer.Send(str, strlen(str));
	}

21：设置机械臂当前模式

0 关节速度模式

1 关节位置模式

2 末端速度模式

3 末端位置模式

4 手柄控制模式

5 视觉伺服模式

6 力控制模式

7 示教再现模式

上位机客户端发送字符串：SETM 0， 设置机器人的工作模式为关节速度模式

服务器端（JK5 机器人）：接收后会设置机器人的工作模式，设置成功后，将发送字符串： SETM : OK

上位机客户端	Client->send(“SETM 0”) string str = Client->receive(1000); cout << "recStr: "<< str<< endr;

下位机返回：	if (stricmp(Command, "SETM") == 0){
	if (nPar == 1) {
	rtn =
	pRobotComm->SetMode(Parameters);
	}
	}
	int CRobotComm::SetMode(const char
	Parameters[MAX_PARAMETER_LEN][256])
	{
	int mode = 0;
	char str[1024] = "";
	mode = atoi(Parameters[0]);
	……
	……
	sprintf(str, "SETM OK");
	return tcpServer.Send(str, strlen(str));
	}

22：设置关节速度比例

上位机客户端发送字符串：SETJVR 0.5， 设置机器人的关节速度比例为 0.5

服务器端（JK5 机器人）：接收后会设置机器人的关节速度比例，设置成功后，将发送字符串： SETJVR : OK

上位机客户端	Client->send(“SETJVR 0.5”) string str = Client->receive(1000); cout << "recStr: "<< str<< endr;

下位机返回：	if (stricmp(Command, "SETJVR") == 0) {
	rtn = pRobotComm->SetJointVelRatio(Parameters);
	}
	int CRobotComm::SetJointVelRatio(const char Parameters[MAX_PARAMETER_LEN][256])
	{
	double jointVelRatio;
	jointVelRatio = atof(Parameters[0]);
	……
	……
	char str[100] = "SETJVR OK";
	return tcpServer.Send(str, strlen(str));
	}

23：设置运行状态

上位机客户端发送字符串：SETSF4 0， 设置机器人的运行状态

服务器端（JK5 机器人）：接收后会设置机器人的关节速度比例，设置成功后，将发送字符串： SETJVR : OK

上位机客户端	Client->send(“SETSF4 0”) string str = Client->receive(1000); cout << "recStr: "<< str<< endr;
下位机返回：	if (stricmp(Command, "SETSF4") == 0) { rtn = pRobotComm->SetSwitchFlag4(Parameters);

	pDlg->SetSwitchFlag4(rtn);
	
	}
	int CRobotComm::SetSwitchFlag4(const char Parameters[MAX_PARAMETER_LEN][256])
	{
	long int flag = -1;
	char str[1024] = "";
	flag = atoi(Parameters[0]);
	
	sprintf(str, "SETSF4 OK");
	tcpServer.Send(str, strlen(str));
	return flag;
	}

24：设置末端位置

上位机客户端发送字符串：SETP 10.0 10.0 10.0 10.0 10.0 10.0， 设置机器人的各个绝对关节位置为 10.0 10.0 10.0 10.0 10.0 10.0，单位为角度

服务器端（JK5 机器人）：接收后会设置机器人的绝对关节位置，设置成功后，将发送字符串： SETP : OK

上位机客户端	Client->send(“SETP 10.0 10.0 10.0 10.0 10.0 10.0”) string str = Client->receive(1000); cout << "recStr: "<< str<< endr;

下位机返回：	if (stricmp(Command, "SETP") == 0) {
	rtn = pRobotComm->SetPos(Parameters);
	}
	int CRobotComm::SetPos(const char Parameters[MAX_PARAMETER_LEN][256]) //注意此处设置为

	绝对值
	{
	double Pos[6];
	for (int i = 0; i < 6; i++) {
	Pos[i] = atof(Parameters[i]);
	}
	……
	……
	char str[100] = "SETP OK";
	return tcpServer.Send(str, strlen(str));
	}

25：设置末端速度比例

上位机客户端发送字符串：SETPVR 0.5， 设置机器人的末端速度比例 0.5

服务器端（JK5 机器人）：接收后会设置机器人的末端速度比例，设置成功后，将发送字符串： SETPVR : OK

上位机客户端	Client->send(“SETPVR 0.5”) string str = Client->receive(1000); cout << "recStr: "<< str<< endr;

下位机返回：	if (stricmp(Command, "SETPVR") == 0) {
	rtn =
	pRobotComm->SetPosVelRatio(Parameters);
	}
	int CRobotComm::SetPosVelRatio(const char
	Parameters[MAX_PARAMETER_LEN][256])
	{
	double PosVelRatio;
	PosVelRatio = atof(Parameters[0]);
	……
	……

	char str[100] = "SETPVR OK";
	return tcpServer.Send(str, strlen(str));
	}

26：机器人能否使能获取

上位机客户端发送字符串：GETPREENABLE ， 查询机器人能否上电使能

服务器端（JK 机器人）：接收后会使机器人上电使能，设置成功后，将发送字符串：SETENABLE : OK

上位机客户端	Client->send(“GETPREENABLE”) string str = Client->receive(1000); cout << "recStr: "<< str<< endr;

下位机返回：	if (stricmp(Command, " GETPREENABLE") == 0) {
	……
	……
	sprintf(str, " GETPREENABLE 1");
	tcpServer.Send(str, strlen(str));
	return flag;
	}

26：机器人使能设置

上位机客户端发送字符串：SETENABLE 1， 使机器人上电使能

服务器端（JK5 机器人）：接收后会使机器人上电使能，设置成功后，将发送字符串：SETENABLE : OK

上位机客户端	Client->send(“SETENABLE 1”) string str = Client->receive(1000); cout << "recStr: "<< str<< endr;

下位机返回：	if (stricmp(Command, "SETENABLE") == 0) {
	rtn =
	pRobotComm->SetEnable(Parameters);
	pDlg->SetEnable(rtn);

	}
	int CRobotComm::SetEnable(const char
	Parameters[MAX_PARAMETER_LEN][256])
	{
	int flag = -1;
	char str[1024] = "";
	……
	……
	sprintf(str, "SETENABLE OK");
	tcpServer.Send(str, strlen(str));
	return flag;
	}

27：设置坐标系

0 0 基坐标绝对运动

0 1 基坐标系相对运动

1 0 末端坐标系绝对运动

1 1 末端坐标相对运动

上位机客户端发送字符串：SETFM 0 1， 设置机器人基坐标系相对坐标

服务器端（JK5 机器人）：接收后设置机器人坐标系，设置成功后，将发送字符串：SETFM : OK

上位机客户端	Client->send(“SETFM 0 1”) string str = Client->receive(1000); cout << "recStr: "<< str<< endr;

下位机返回：	if (stricmp(Command, "SETFM") == 0) {
	rtn = pRobotComm->SetFrameMotion(Parameters);
	}
	int CRobotComm::SetFrameMotion(const char
	Parameters[MAX_PARAMETER_LEN][256])

	{
	int frame = 0;
	int motion = 0;
	frame = atoi(Parameters[0]);
	motion = atoi(Parameters[1]);
	……
	……
	char str[100] = "SETFM OK";
	return tcpServer.Send(str, strlen(str));
	}

28：设置负载惯性参数

1 质量，2-4 质量*质心位置

上位机客户端发送字符串：SETLP 10.0 15.0 15.0 15.0 ， 设置机器人的负载惯性参数

服务器端（JK5 机器人）：接收后会设置机器人的负载惯性参数，设置成功后，将发送字符串： SETLP : OK

上位机客户端	Client->send(“SETLP 10.0 15.0 15.0 15.0”) string str = Client->receive(1000); cout << "recStr: "<< str<< endr;

下位机返回：	if (stricmp(Command, "SETLP") == 0) {
	rtn =
	pRobotComm->SetLoadParas(Parameters);
	}
	int CRobotComm::SetLoadParas(const char
	Parameters[MAX_PARAMETER_LEN][256])
	{
	double load_params[4];
	for (int i = 0; i < 4; i++) {
	load_params[i] = atof(Parameters[i]);

	}
	……
	……
	char str[100] = "SETLP OK";
	return tcpServer.Send(str, strlen(str));
	}

29：设置虚拟墙上限和下限

上位机客户端发送字符串：SETVWL -0.5 0.5 -0.5 0.5 0 0.5 ， 在基坐标系下的设置机器人的虚拟墙立方体

服务器端（JK5 机器人）：接收后会设置机器人的虚拟墙，成功后，将发送字符串：SETVWA : OK

上位机客户端	Client->send(“SETVWL -0.5 0.5 -0.5 0.5 0 0.5”) string str = Client->receive(1000); cout << "recStr: "<< str<< endr;

下位机返回：	if (stricmp(Command, "GETVWL") == 0) {
	rtn = pRobotComm->GetVirtualWallLimit();
	}
	int CRobotComm::SetVirtualWallLimit(const char Parameters[MAX_PARAMETER_LEN][256])
	{
	double VirtualWallLimit[6];
	for (int i = 0; i < 6; i++) {
	VirtualWallLimit[i] = atof(Parameters[i]);
	}
	……
	……
	char str[100] = "SETVWL OK";

	return tcpServer.Send(str, strlen(str)); }

30：设置虚拟墙绕 z 轴旋转角

上位机客户端发送字符串：SETVWA 20 ， 在基坐标系下的设置机器人的虚拟墙立方体后，再使其绕 z 轴旋转 20 度

服务器端（JK5 机器人）：接收后会设置机器人的虚拟墙绕 z 轴的旋转角，成功后，将发送字符串： SETVWA : OK

上位机客户端	Client->send(“SETVWA 20”) string str = Client->receive(1000); cout << "recStr: "<< str<< endr;

下位机返回：	if (stricmp(Command, "SETVWE") == 0) {
	rtn = pRobotComm->SetVirtualWallEnable(Parameters);
	}
	int CRobotComm::GetVirtualWallAngle()
	{
	double VirtualWallAngle = 0;
	char str[1024] = "";
	……
	……
	sprintf(str, "GETVWA %d", VirtualWallAngle);
	return tcpServer.Send(str, strlen(str));
	}

31：设置虚拟墙启用

上位机客户端发送字符串：SETVWE 1 ， 在启用机器人的虚拟墙

服务器端（JK5 机器人）：接收后会启用机器人的虚拟墙，成功后，将发送字符串：SETVWE : OK

上位机客户端	Client->send(“SETVWE 1”)

	string str = Client->receive(1000); cout << "recStr: "<< str<< endr;

下位机返回：	if (stricmp(Command, "SETVWE") == 0) {
	rtn =
	pRobotComm->SetVirtualWallEnable(Parameters);
	}
	int CRobotComm::SetVirtualWallEnable(const char
	Parameters[MAX_PARAMETER_LEN][256])
	{
	int VirtualWallEnable = 0;
	VirtualWallEnable = atoi(Parameters[0]);
	……
	……
	char str[100] = "SETVWE OK";
	return tcpServer.Send(str, strlen(str));
	}

32：设置关节速度限幅

上位机客户端发送字符串：SETJVS 1 ， 在启用机器人关节速度限幅

服务器端（JK5 机器人）：接收后会启用机器人关节速度限幅，成功后，将发送字符串： SETJVS : OK

上位机客户端	Client->send(“SETJVS 1”) string str = Client->receive(1000); cout << "recStr: "<< str<< endr;

下位机返回：	if (stricmp(Command, "SETJVS") == 0) {
	rtn =
	pRobotComm->SetJoyVelScale(Parameters);
	}
	int CRobotComm::SetJoyVelScale(const char

	Parameters[MAX_PARAMETER_LEN][256])
	{
	double JoyVelScale = 0;
	char str[1024] = "";
	JoyVelScale = atof(Parameters[0]);
	……
	……
	sprintf(str, "SETJVS OK");
	return tcpServer.Send(str, strlen(str));
	}

33：设置 TOOL 输入

上位机客户端发送字符串：SETTOOLDATA 1 1 1 ，将 tool 的 3 个输入量设为 true
服务器端（JK5 机器人）：接收后会设置 tool 的 3 个输入量，设置成功后，将发送字符串： SETTOOLDATA OK

上位机客户端	Client->send(“SETTOOLDATA 1 1 1”) string str = Client->receive(1000); cout << "recStr: "<< str<< endr;

下位机返回：	else if (stricmp(Command, "SETTOOLDATA") == 0) {
	rtn = pRobotComm->SettoolDatain(Parameters);
	}
	int CRobotComm::SettoolDatain(const char Parameters[MAX_PARAMETER_LEN][256])
	{
	int tooldata[3];
	for (int i = 0; i < 3; i++)
	{
	tooldata[i] = atoi(Parameters[i]);
	if (tooldata[i] == 0)

	{
	……
	}
	else ……
	}
	char str[100] = "SETTOOLDATA OK";
	return tcpServer.Send(str, strlen(str));
	}

34：设置 BOX 输入

上位机客户端发送字符串：SETBOXDATA 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1，将 box 的 16 个输入量设为 true

服务器端（JK5 机器人）：接收后会设置 box 的 16 个输入量，设置成功后，将发送字符串： SETBOXDATA OK

上位机客户端	Client->send(“SETBOXDATA 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1”) string str = Client->receive(1000); cout << "recStr: "<< str<< endr;

下位机返回：	else if (stricmp(Command, "SETBOXDATA") == 0) {
	rtn = pRobotComm->SetboxDatain(Parameters);
	}
	int CRobotComm:: SetboxDatain(const char Parameters[MAX_PARAMETER_LEN][256])
	{
	int boxdata[16];
	for (int i = 0; i < 16; i++)
	{
	boxdata[i] = atoi(Parameters[i]);
	if (boxdata[i] == 0)

	{
	……
	}
	else ……
	}
	char str[100] = " SETBOXDATA OK";
	return tcpServer.Send(str, strlen(str));
	}

35：设置电压

上位机客户端发送字符串：SETLEVEL 1，将电压设置为 24V

服务器端（JK5 机器人）：接收后会设置电压值，设置成功后，将发送字符串：SETLEVEL OK

上位机客户端	Client->send(“SETLEVEL 1”) string str = Client->receive(1000); cout << "recStr: "<< str<< endr;

下位机返回：	else if (stricmp(Command, "SETLEVEL") == 0) {
	rtn = pRobotComm->Setlevel(Parameters);
	}
	int CRobotComm::Setlevel(const char Parameters[MAX_PARAMETER_LEN][256])
	{
	int Vlevel;
	……
	if (Vlevel == 0)
	{
	……
	}
	else ……
	char str[100] = "SETLEVEL OK";

	return tcpServer.Send(str, strlen(str)); }

36：获取 TOOL 输出

上位机客户端发送：GETTOOLDATA

服务器端（JK5 机器人）：将发送 str 字符串 GETTOOLDATA 0 1 0 。GETTOOLDATA 为数据头，后面 3 个为 TOOL 输出数据。

上位机客户端	tcp->Send(cmdStr);
	string str = tcp->receive(1000);
	cout << "recStr: " << str << endr;

下位机返回：	else if (stricmp(Command, "GETTOOLDATA") == 0) {
	rtn = pRobotComm->GettoolDataout();
	}
	int CRobotComm::GettoolDataout()
	{
	int tooldata_out[3];
	bool TorF;
	char str[1024] = "";
	for (int i = 0; i<3; i++)
	{
	……
	if (TorF == true)
	{
	……
	}
	else ……
	}
	sprintf(str, "GETTOOLDATA %d %d %d", tooldata_out[0], tooldata_out[1], tooldata_out[2]);
	return tcpServer.Send(str, strlen(str));
	}

37：获取 BOX 输出

上位机客户端发送：GETBOXDATA

服务器端（JK5 机器人）：将发送 str 字符串 GETBOXDATA 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0。GETBOXDATA 为数据头，后面 16 个为 BOX 输出数据。

上位机客户端	tcp->Send(cmdStr);
	string str = tcp->receive(1000);
	cout << "recStr: " << str << endr;

下位机返回：	else if (stricmp(Command, "GETBOXDATA") == 0) {
	rtn = pRobotComm->GetboxDataout();
	}
	int CRobotComm::GetboxDataout()
	{
	int boxdata_out[16];
	bool TorF;
	char str[1024] = "";
	for (int i = 0; i<16; i++)
	{
	……
	if (TorF == true)
	{
	……
	}
	else ……
	}
	sprintf(str, "GETBOXDATA %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d", boxdata_out[0], boxdata_out[1],boxdata_out[2],boxdata_out[3], boxdata_out[4], boxdata_out[5],boxdata_out[6], boxdata_out[7],boxdata_out[8],boxdata_out[9], boxdata_out[10],boxdata_out[11], boxdata_out[12],

	boxdata_out[13], boxdata_out[14], boxdata_out[15]);
	return tcpServer.Send(str, strlen(str));
	}

4 快速开发历程：

4.1 基于关节的机器人移动

```
Client->setup(“192.168.0.150”, 8800);  //建立连接

If(0 == Client->send(“SETSF4 0”) )  return 0;    //停止运行机器人
If(0 == Client->send(“SETENABLE 1”) ) return 0;  //机器人使能
If(0 == Client->send(“SETM 2”)) return 0;   // 设置运行模式为 关节位置模式
Client->send(“SETSI  0.6”)              //设置安全系数为 0.6
Client->send(“SETJVR 0.3”)              //设置关节速度 0.3 rad/s
Client->send(“GETJVS 0.4”)              // 设置关节速度限幅 0.4rad/s
Client->send(“SETJ 10 10 10 10 10 10”)   //设置机器人运行到关节绝对位置 注意，请根据机器人当前角度进行设置。

If(0 == Client->send(“SETSF4 1”) )  return 0;    //运行机器人
```

SDK 包提供基础运行功能，便于客户从底层开发。同时本公司能够提供更加简单的运行 SDK 包。如上述功能可运行函数为

MovAbsJoint (Vector6d joints, float Jvl); // joints 关节角度 Jvl 运行速度

其他功能函数，例如：

（1）相对角度运行：

MovRelJoint (Vector6d joints, float Jvl); // joints 关节角度 Jvl 运行速度

（2）笛卡尔空间，基于末端坐标系的相对运行

movRelTool(Vector6d& pos, double Pvr); // pos 相对位置 Pvr 运行速度

方便用户的快速开发要求。