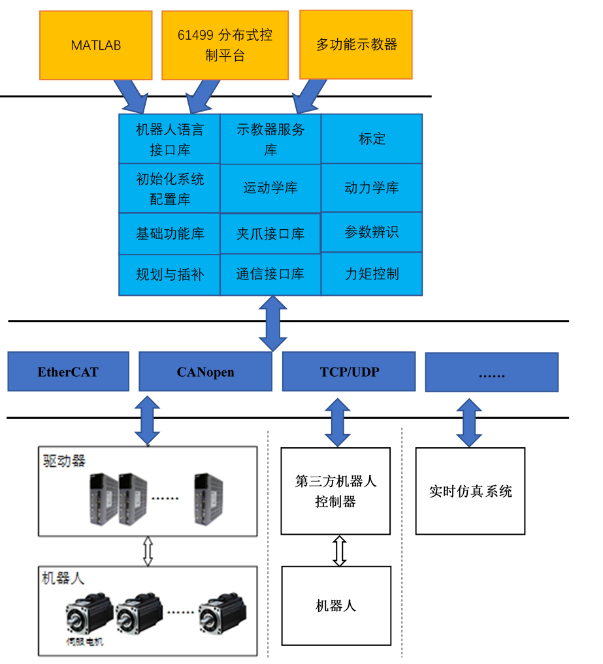
**1、程序架构**



EtherCAT通信服务程序

机器人运动控制程序

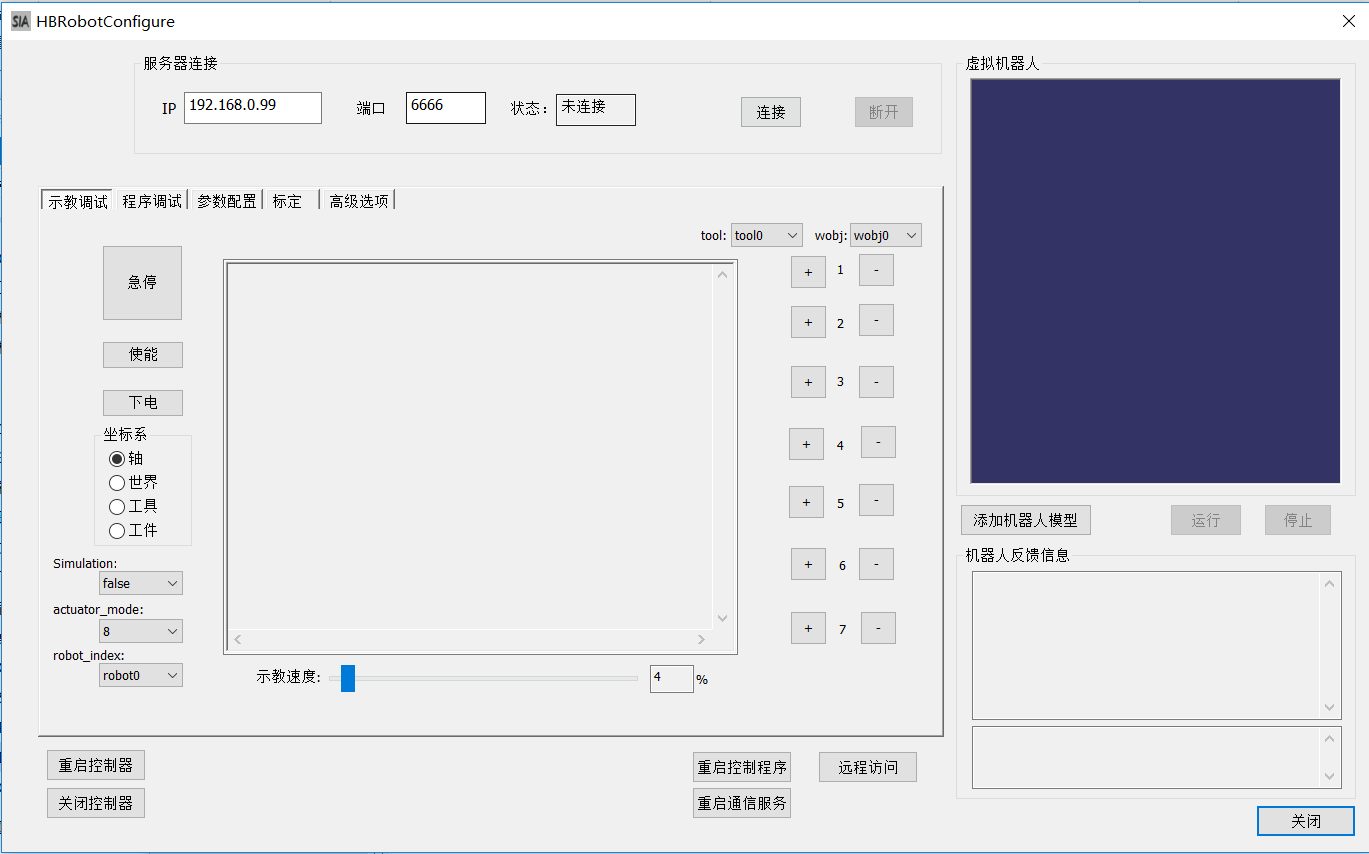
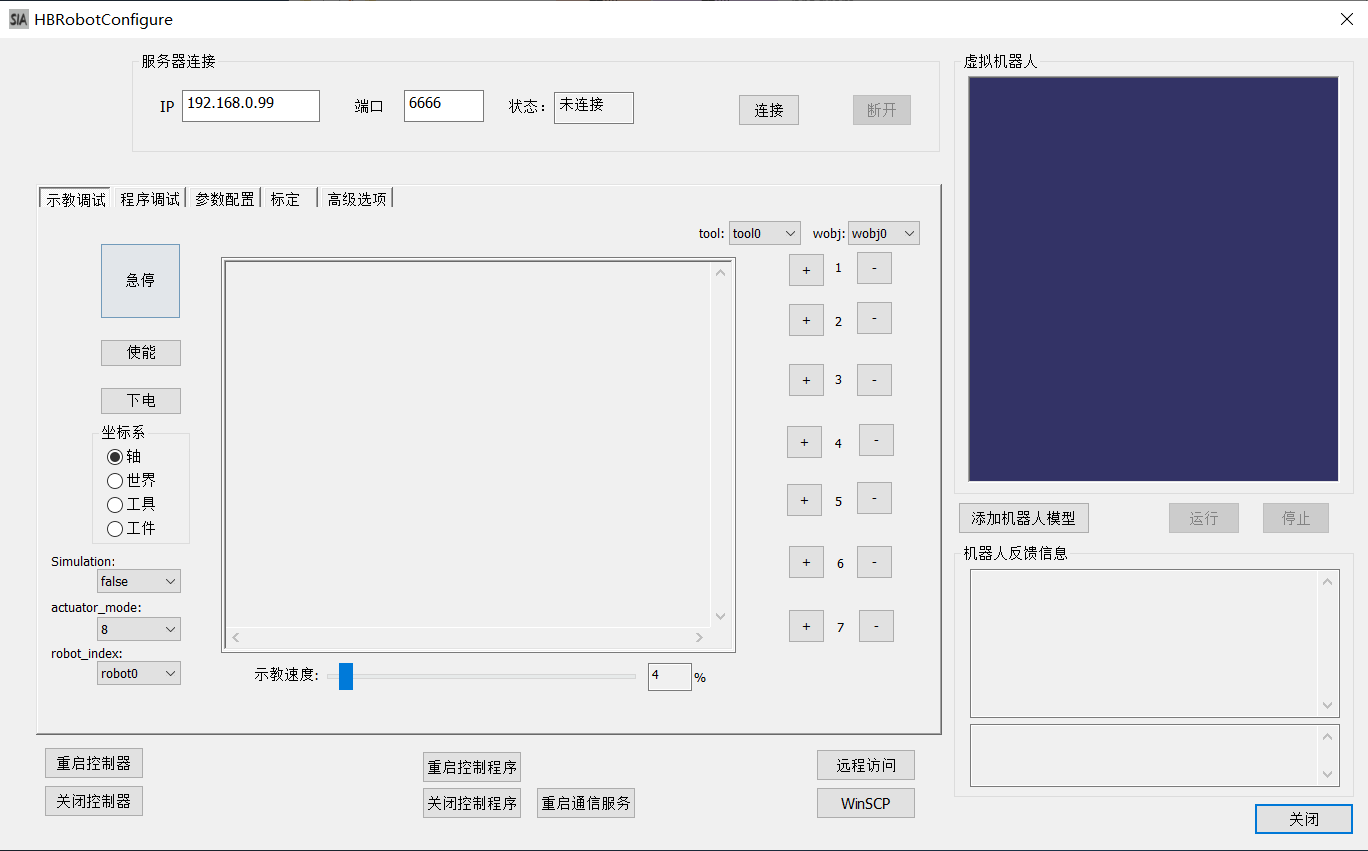
**2、远程访问控制器**

方法一：

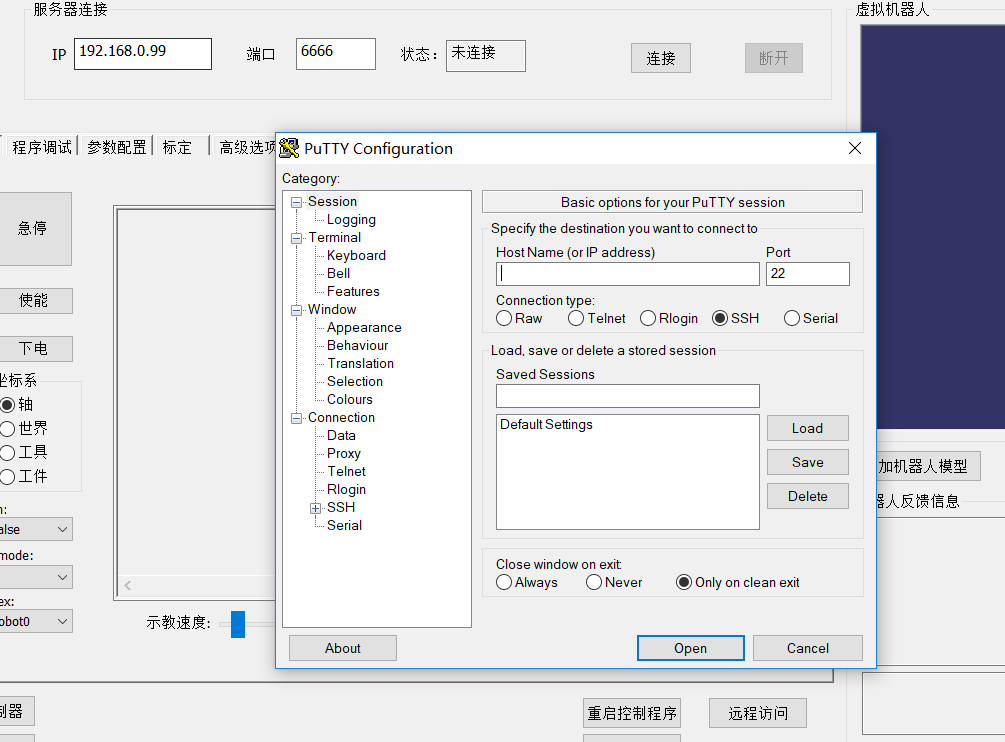
Ubuntu系统内, 命令行输入ssh [root@192.168.0.99](mailto:root@192.168.0.99), 按提示输入密码root

方法二：

Windows系统内打开机器人桌面控制程序HBRobotConfigure.exe



单击“远程访问”弹出如下界面(无需连接操作)



Host Name内输入192.168.0.99，单击Open后弹出终端，输入用户名root和密码root

**3、启动EtherCAT服务程序（注：驱动器通电）**

方法一：

EtherCAT服务程序为自己程序，在驱动器状态正常通电情况下启动控制器，EtherCAT服务程序会自启动。

方法二：

远程终端内，进入/opt/ECMworkspace\_64文件夹，终端输入./runECM.sh 即可。

方法三：

打开桌面控制程序，单击下方“重启通信服务”即可。

注：驱动出现异常，去要重启控制柜，之后重启通信服务程序。

**4、启动机器人运动控制程序（EtherCAT服务程序已启动，否则进入仿真模式）**

方法一：

远程终端内，进入/robot文件夹，终端输入./RobotMain 即可。

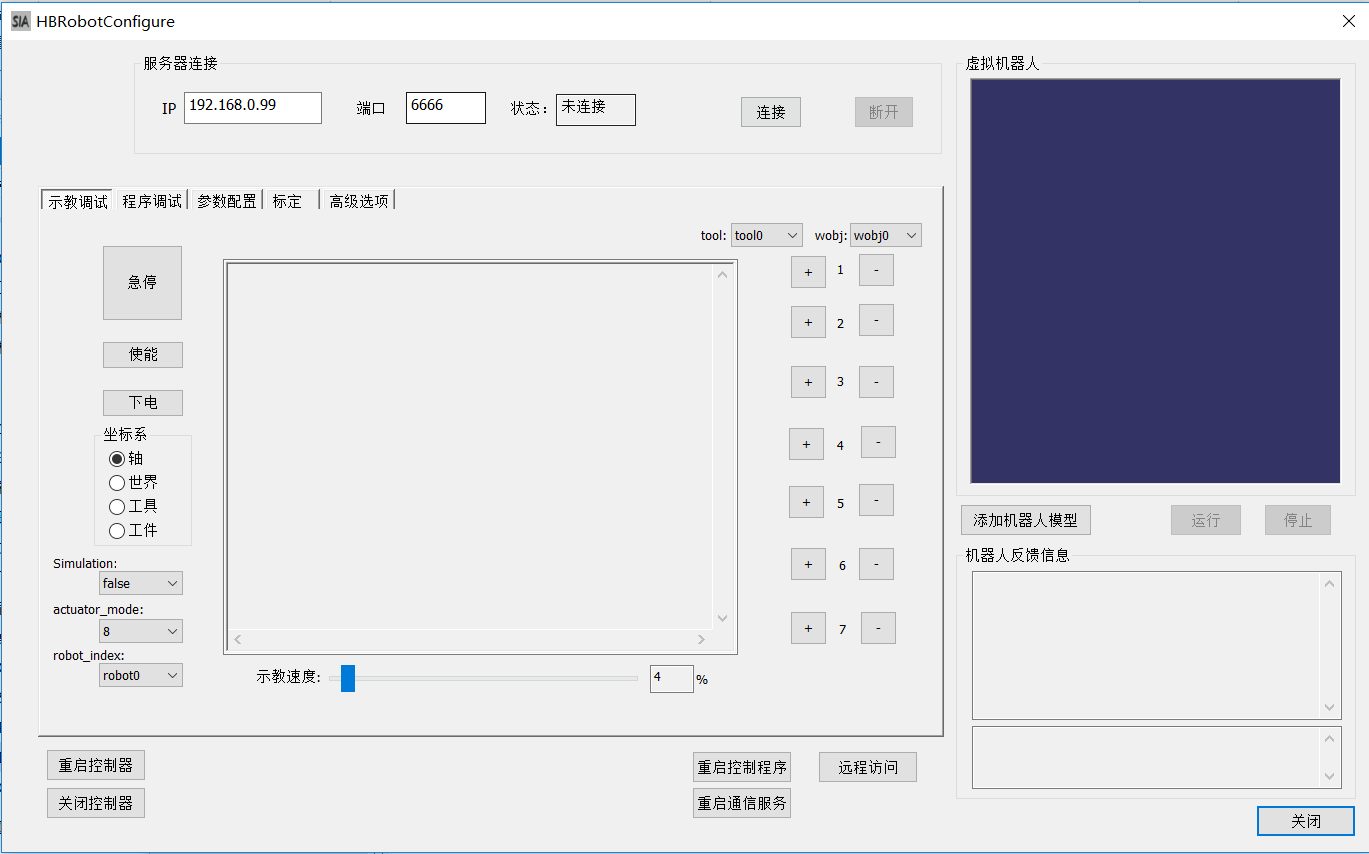
方法二：

打开桌面控制程序，单击下方“重启控制程序”即可。

注：重启通信服务程序后需重启机器人运动控制程序。

**5、桌面控制程序的使用**

*关于示教：*



a.单击“连接”与控制器建立连接（机器人运动控制程序已启动）。

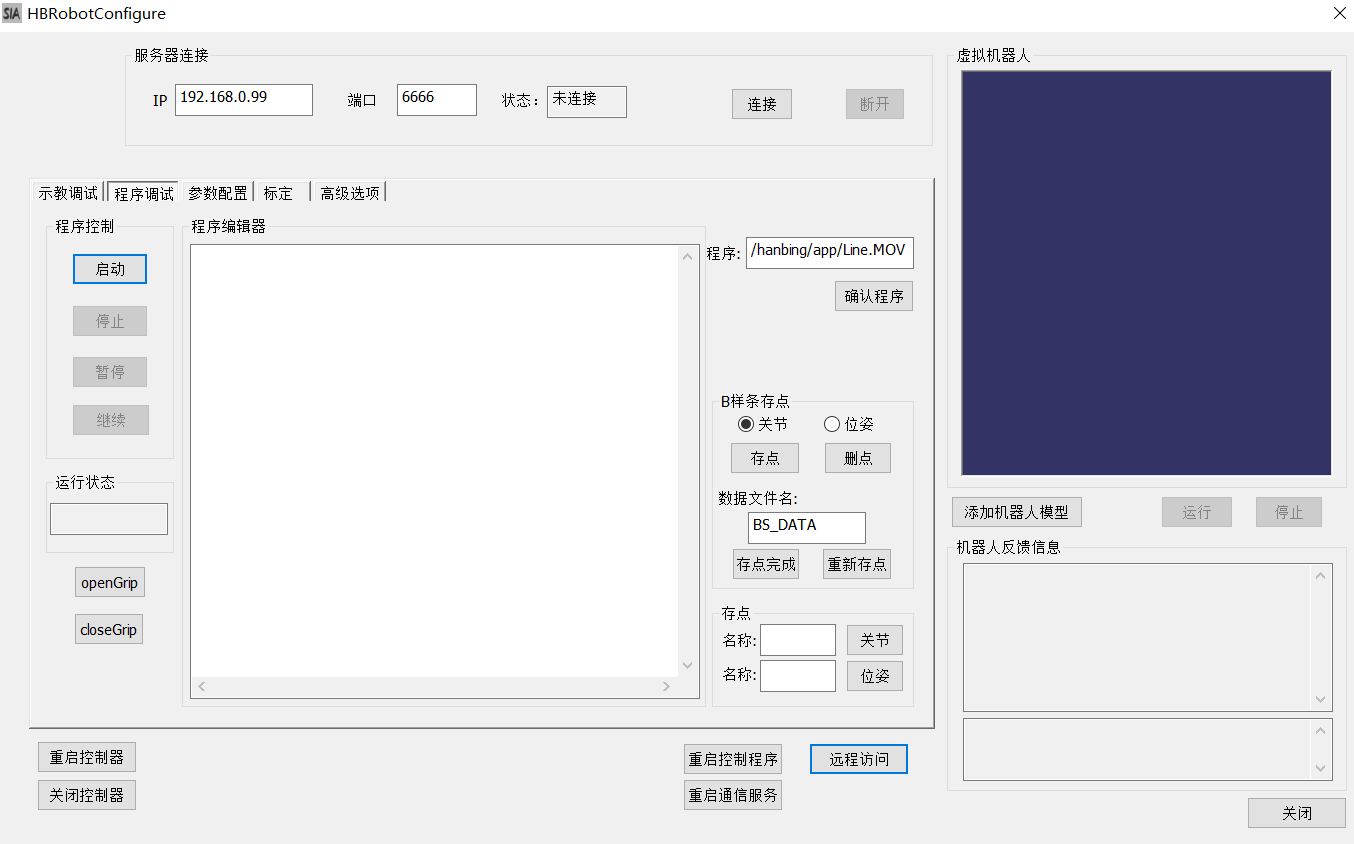
b.单击“添加机器人模型”，并单击“运行”，显示实际或仿真机器人状态。

c.左侧“使能”和“下电”，控制机器人的使能状态，使能后才能控制机器人运动。

d.“坐标系”控制机器人运动的坐标系，选择“轴”时，右侧数字对应的是机器人关节的运动控制按键。选择“世界”时，为笛卡尔空间运动，参考系为基坐标系，右侧数据对应的是x,y,z,Rx,Ry,Rz。选择“工具”时参考坐标系为工具坐标系，即tool的选项，默认为法兰坐标系。选择“工件”时参考系为工件坐标系，即wobj的选项，默认为基坐标系。

e.左侧“Simulation”为ture时为仿真模式，false为真机模式。

*关于脚本执行：*



程序内输入脚本名称：如/hanbing/app/Lin0.MOV，单击左侧“启动”即可运行脚本。单击“停止”结束脚本运行。

存点框内输入关节和位置名称，如：j1 和p1，单击“关节”和“位置”记录示教点，在脚本编写时使用。

运行脚本可自行创建和编写，后缀名称.MOV。放入至/hanbing/app下即可在界面中启动。

语法规则参加lua相关文档。

机器人控制接口函数参加《机器人封装接口v1.7》，确定udp即可介绍，udp的使用同tcp。

例子：

robot.sleep(0.5)

dy1=robot.SocketCreate("192.168.0.99",6661,"HMI")

print(dy1)

robot.sleep(1)

dy2=robot.ClientCreate("192.168.0.168",8080,"camera")

print(dy2)

robot.sleep(1)

robot.MoveA(j1,v1)

robot.sleep(1)

hmi\_count=0

pos\_offset=100

grip\_offset=20

fx\_offset=100

put\_offset=130

data={}

while(1) do

hmi=robot.SocketRecvIntArray("HMI")--array num 5

hmi\_num=#hmi

print(hmi\_num)

hmi\_num=math.modf(hmi\_num/4)

print(hmi\_num)

hmi\_count=0

while(hmi\_count<hmi\_num) do

robot.Grip\_position(0)

robot.sleep(1)

nn=hmi\_count\*4

data[1]=hmi[1+nn]

data[2]=hmi[2+nn]

data[3]=hmi[3+nn]

data[4]=hmi[4+nn]

print(hmi[1],hmi[4])

robot.SocketSendIntArray(data,4,"camera")

pos=robot.SocketRecvDoubleArray("camera")

print(pos[1],pos[2],pos[3],pos[4],pos[5])

if (pos[4]>90) then

pos[4]=90

end

zz=pos[4]+pos\_offset

robot.MoveL(robot.Offs(p2,pos[2],pos[3],zz,0,0,pos[5]),v1,0,tool10,wobj10)

robot.sleep(0.5)

zz0=pos[4]-grip\_offset

robot.MoveL(robot.Offs(p2,pos[2],pos[3],zz0,0,0,pos[5]),v1,0,tool10,wobj10)

robot.sleep(0.5)

robot.Grip\_position(1)

robot.sleep(1)

zz=pos[4]+pos\_offset

robot.MoveL(robot.Offs(p2,pos[2],pos[3],zz,0,0,pos[5]),v1,0,tool10,wobj10)

robot.sleep(0.5)

robot.MoveA(j3,v1)

zz1=-put\_offset

robot.MoveL(robot.Offs(p3,pos[2],pos[3],0,0,0,0),v1,0,tool10,wobj10)

robot.sleep(0.5)

robot.MoveL(robot.Offs(p3,pos[2],pos[3],zz1,0,0,0),v1,0,tool10,wobj10)

robot.sleep(0.5)

robot.Grip\_position(0)

robot.sleep(0.5)

robot.MoveL(robot.Offs(p3,pos[2],pos[3],0,0,0,0),v1,0,tool10,wobj10)

robot.MoveA(j1,v1)

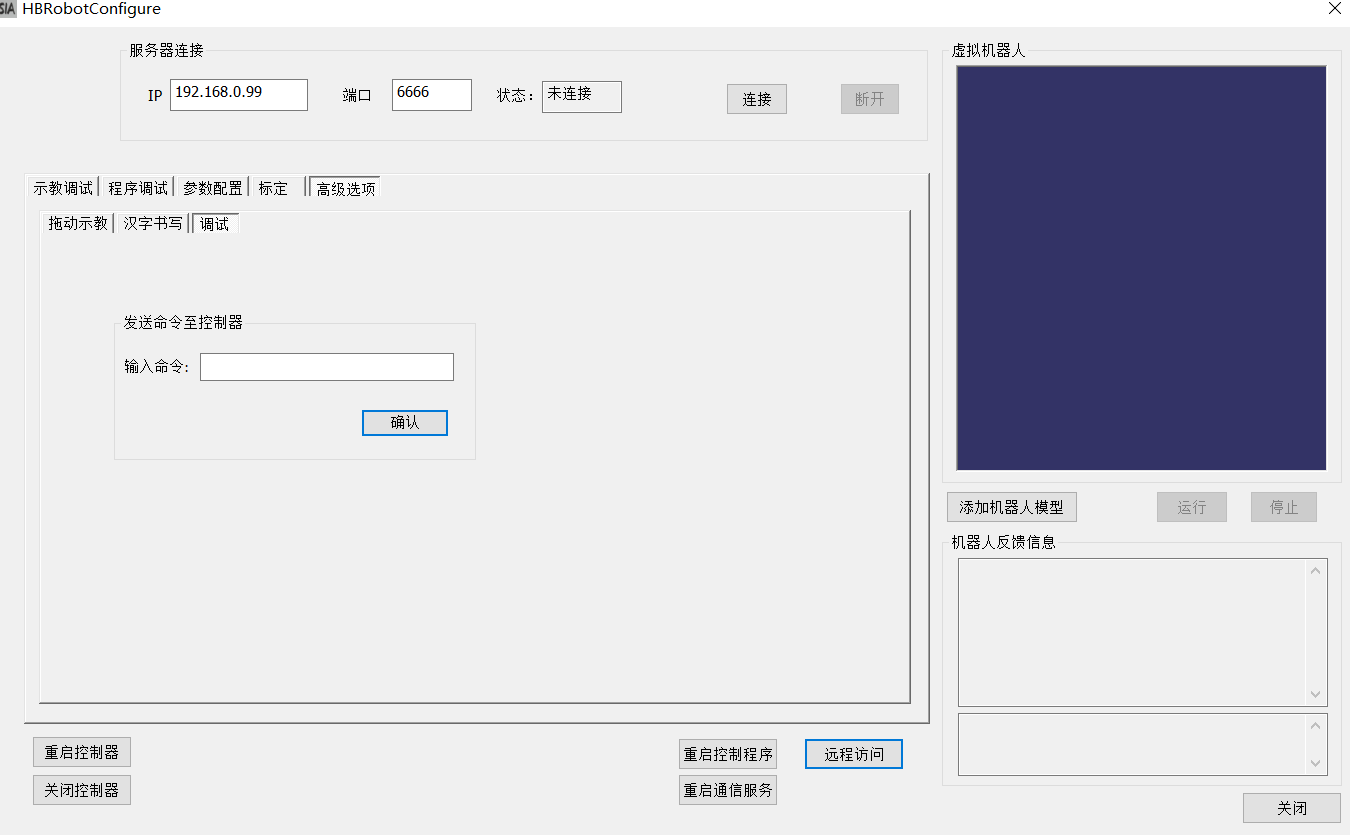
hmi\_count=hmi\_count+1;

end

robot.SocketSendInt(0,"HMI")

end

*关于《HIT\_data.txt》数据的执行*



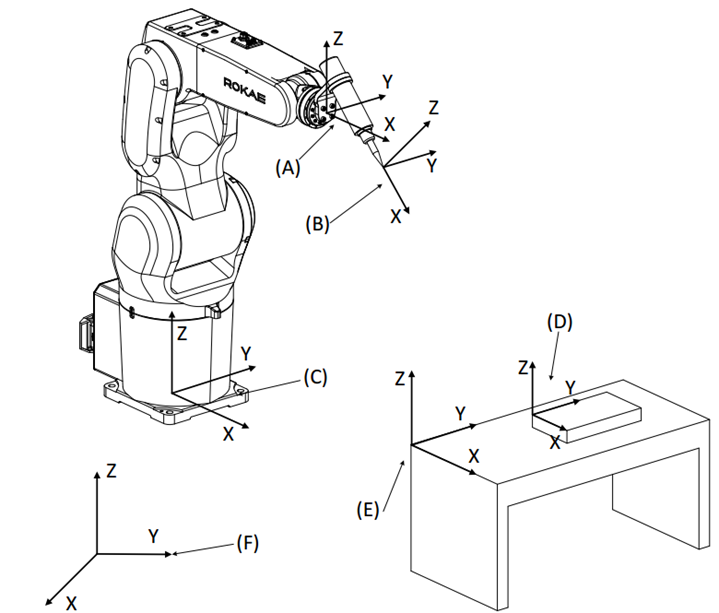
将准备好的数据HIT\_data.txt，拷贝至目标机的/robot目录下。

输入命令：A 0 100设置末端运动速度为100mm/s。默认为（50mm/s）,单击“确认”设置命令。

输入命令：hit\_on 开始执行HIT\_data.txt，内的输入，运动的开始阶段为直线运动到HIT\_data.txt内的第一个点。

注：使之hit\_on前调整好机器人运行的姿态，hit\_off为终止运动（该命令目前没有减速停，慎用）。

**6、坐标系建立**



A:法兰坐标系

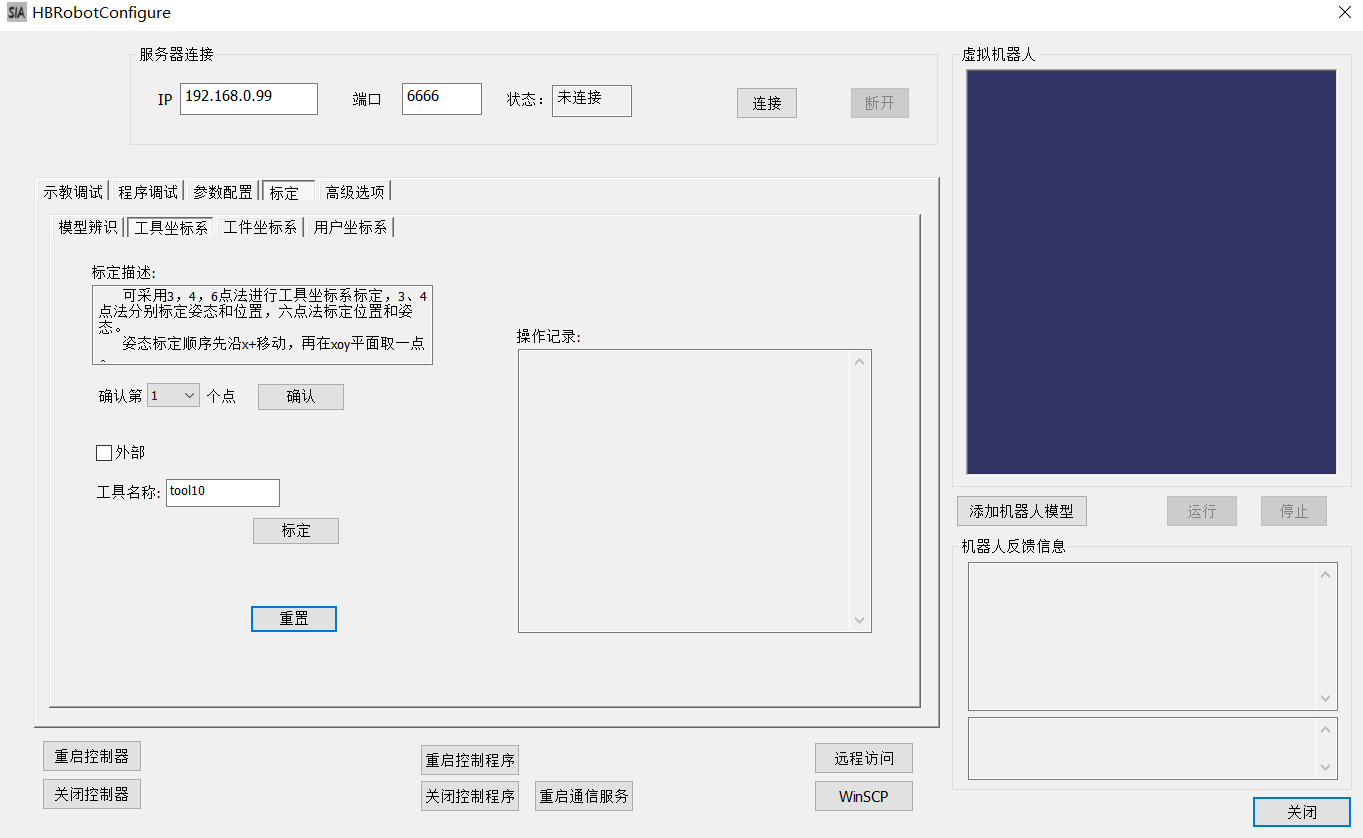
B:工具坐标系（相对A描述）

C:基坐标系

D:工件坐标系（相对E描述）

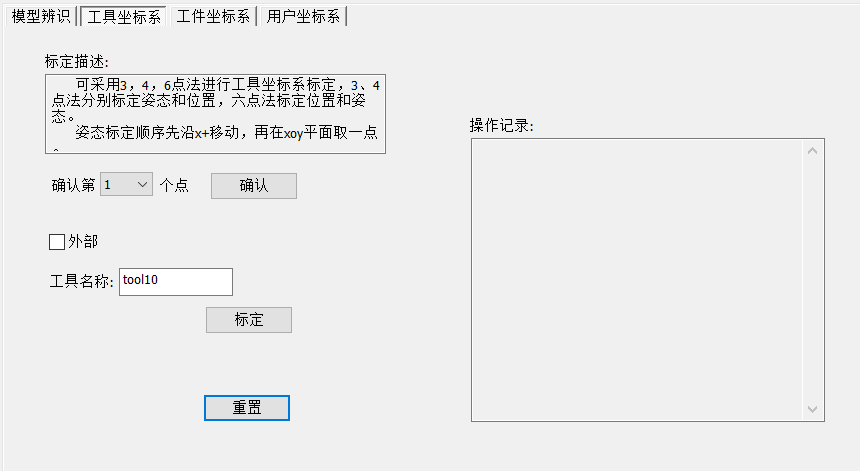
E:用户坐标系（相对C描述）

F:世界坐标系



选择“标定”选卡，分别进入“工具坐标系”、“工件坐标系”、“用户坐标系”进行标定。

工具坐标系标定：



可采用4点法标定工具位置，姿态与法兰姿态一致。利用3点法标定工具姿态，但需要以标定位置的工具上进行标定。采用6点法同时进行位置及姿态标定。

4点法标定工具位置：要求工具尖端位置一致，机器人摆出四种构型（姿态差距尽量大些）分别确认4个点。如果工具未安装在机器人上需要勾选“外部”。输入名字最后单击“标定”即可。

3点法标定工具姿态：分别位原点、X+上一点、XOY平面上一点，注意工具姿态标定采集的3个点相对基座的姿态保持一致。

6点法标定工具位置及姿态：前4点按照4点法标定，后2个点按照3点法标定，分别位X+上一点，XOY平面上一点。

用户坐标系标定：



标定用户坐标系需要确认坐标系上的3个点，分别是坐标系原点、X+上一点、XOY平面上一点。分别“确认”，最后单击“标定”即可。确认点距离较大时有利于标定的精度。

工件坐标系标定：



工件坐标系标定方法与工具坐标系方法一致。由于工件坐标系是相对用户坐标系描述的因此标定时需要输入所依赖的用户坐标系。非必要情况下可选择默认用户坐标系（与基坐标系一致）。