节卡 SDK 力控功能快速入门指南

在通过 SDK 使用任何力控功能前，请务必参阅《节卡力控产品使用手册》进行正确的软、硬件环境搭建，了解产品功能的详细信息并严格遵循使用注意事项。此外，此文档需要结合节卡 SDK 手册进行阅读，本文仅对常用力控功能涉及的 API 接口进行简略提及，节卡 SDK 手册则列出了所有支持的 API 接口以及它们的使用说明。

本指南适用于 1.7.0.38 及以上、1.7.2 以下的控制器版本。

力控功能相关 API 简介

通过二次开发搭建力控应用涉及的基本 API 包括:

通用：
errno_t get_robot_status(RobotStatus* status); 获取机械臂状态监测数据
errno_t set_torque_sensor_mode(int sensor_mode); 开启或关闭末端力传感器
errno_t set_ft_ctrl_frame(const int ftFrame); 设置力控坐标系（出厂默认设置为工具坐标系）
errno_t set_compliant_type(int sensor_compensation, int compliance_type); 设置力控类型和校零（初始化）选项
errno_t disable_force_control(); 关闭力柔顺控制
errno_t set_torque_sensor_filter(const float torque_sensor_filter); 设置末端力传感器低通滤波器截止频率（出厂默认设置为 0，不开启滤波）

恒力柔顺模式：
errno_t set_admit_ctrl_config(int axis, int opt, int ftUser, int ftConstant, int ftNnormal Track, int ftReboundFK); 设置恒力柔顺控制参数

速度柔顺模式（不可配合伺服模式使用，1.7.2 及以上版本不再支持）：
errno_t set_vel_compliant_ctrl(const VelCom* vel_cfg); 设置速度柔顺控制参数
errno_t set_compliance_condition(const FTxyz* ft); 设置速度柔顺控制力条件

工具拖拽示教模式（不可配合任何运动模式使用）：
errno_t enable_admittance_ctrl(const int enable_flag); 工具拖拽开启和关闭
上述力控功能与 JAKA Zu APP 中的同名功能是一致的，使用前请务必参阅力控用户手册，了解这些功能的详细含义以及使用注意事项。

需要注意的是，恒力柔顺模式必须搭配运动指令一同使用才会生效，包括:
一般运动指令：
errno_t joint_move(const JointValue* joint_pos, MoveMode move_mode,BOOL is_block, double speed); 机械臂关节运动

errno_t linear_move(const CartesianPose* end_pos, MoveMode move_mode, BOOL is_block, doub le speed); 机械臂末端直线运动

伺服运动模式：

errno_t servo_move_enable(BOOL enable); 机械臂伺服位置控制模式使能

errno_t servo_j(const JointValue* joint_pos, MoveMode move_mode); 机械臂关节空间伺服模式运动

errno_t servo_p(const CartesianPose* cartesian_pose, MoveMode move_mode); 机械臂笛卡尔空间伺服模式运动

上述各 API 的详细参数说明请参阅节卡二次开发手册。此外，二次开发还支持诸如：

errno_t set_torsenosr_brand(int sensor_brand); 设置力传感器型号

errno_t set_torque_sensor_soft_limit(const FTxyz torque_sensor_soft_limit); 设置力传感器的软限位

errno_t set_torq_sensor_tool_payload(const PayLoad* payload); 设置传感器末端负载

errno_t start_torq_sensor_payload_identify(const JointValue* joint_pos); 开始辨识工具末端负载

等 API，这些 API 的作用与 JAKA Zu APP 上的同名功能相同，使用方式可以参考力控产品用户手册。在一般情况下，建议使用 APP 进行这些配置，而不是使用二次开发 API，除非传感器限位或负载等信息是需要自动更新的。

注意：在使用任何力控功能前，传感器型号和负载信息都必须正确配置，无论是通过 APP 还是通过二次开发 API 的方式。传感器型号和负载信息在设置一次后将永久保存，除非再次设置或重置机器人系统。

注意：使用二次开发 API 设置传感器型号时，set_torsenosr_brand 指令与通过 set_torque_sensor_mode 指令开启传感器之间必须间隔 2 秒以确保型号已设置成功，否则无法成功开启传感器。

搭建配合伺服运动的恒力柔顺控制应用例程

在开始前，请正确配置传感器硬件和软件环境，包括正确设置传感器型号和负载。
一个典型的伺服运动恒力柔顺控制应用的搭建如下:

```
#include "JAKAZuRobot.h"
#include <iostream>
#include "Windows.h"
int main()
{
    //实例化机械臂控制类
    JAKAZuRobot demo;
    //登陆到机器人，需要将10.5.5.100替换为自己控制器的IP
    std::cout << demo.login_in("10.5.5.100") << std::endl;
    //机器人上电
    demo.power_on();
    //机器人上使能
    demo.enable_robot();

    //开启传感器，初次开启前建议在APP中检查传感器是否可以正常工作
     demo.set_torque_sensor_mode(1);
    //设置恒力柔顺参数，此处仅开启z方向柔顺力控，设置阻尼为50，恒力为5N
    demo.set_admit_ctrl_config(2, 1, 50, 5, 0, 0);
    demo.set_admit_ctrl_config(0, 0, 0, 0, 0, 0);
    demo.set_admit_ctrl_config(1, 0, 0, 0, 0, 0);
    demo.set_admit_ctrl_config(3, 0, 0, 0, 0, 0);
    demo.set_admit_ctrl_config(4, 0, 0, 0, 0, 0);
    demo.set_admit_ctrl_config(5, 0, 0, 0, 0, 0);

    //机器人进入伺服运动模式
    demo.servo_move_enable(1);
    Sleep(100);
    //开启恒力柔顺控制，必须在进入伺服运动模式后再开启恒力柔顺控制。
    //注意: 1.7.1.37FC，1.7.2及以上版本控制器，如果需要初始化，必须先下发参数1，0然后等待1秒再下发0，1。不可同时下发1，1
    demo.set_compliant_type(1, 1);

    //机器人从当前位置开始，向x正方向以12.5mm/s的速度运动4秒
    //同时因为已经开启了恒力柔顺控制，将控制z方向的接触力为5N
    CartesianPose cart;
    RobotStatus status; //机器人状态检测结构体，包括力信息
    for (int i = 0; i < 500; i++) {
        //实时监控并打印z方向力信息
```

```
        demo.get_robot_status(&status);
        std::cout << "Force at z direction is: " <<
    status.torq_sensor_monitor_data.actTorque[2] << std::endl;
        //向机器人发送运动指令，每8毫秒向x轴正方向移动0.1mm
        cart.tran.x = 0.1; cart.tran.y = 0; cart.tran.z = 0;
        cart.rpy.rx = 0; cart.rpy.ry = 0; cart.rpy.rz = 0;
        demo.servo_p(&cart, INCR, 1);
    }

    //机器人退出伺服运动模式
    demo.servo_move_enable(0);
    //关闭恒力柔顺控制，必须在退出伺服运动模式后再关闭恒力柔顺控制
    //set_compliant_type指令与disable_force_control必须同时使用以安全关闭力控
    demo.set_compliant_type(0, 0);
    demo.disable_force_control();
    //关闭机器人并退出登录
    demo.power_off();
    demo.disable_robot();
    demo.login_out();
    return 0;
}
```

警告：恒力柔顺控制配合一般运动模式使用时，如果运动指令选择了不阻塞模式，恒力柔顺控制必须在运动指令全部完成或全部中止后再关闭，以免造成机器人不受控运动。

警告：恒力柔顺控制配合一般运动模式使用时，如需关闭力控，必须使用 set_compliant_type(1, 0)或 set_compliant_type(0, 0)将 compliant type 设置为 0，并且继续调用 disable_force_control()指令才能安全关闭，否则可能造成机器人不受控运动。

注意：伺服运动模式下，机器人控制器不进行运动规划，机器人将以 8ms 的周期连续执行运动点位指令，用户需自行对轨迹和速度进行规划，并连续向机器人发送位置指令，安全起见，使用力控功能时，相邻的两条位置指令距离建议在 3mm，0.3° 以内。

注意：恒力柔顺控制必须在伺服运动模式开启之后开启，也必须在伺服运动模式退出后再退出。

注意：恒力柔顺控制配合一般运动模式使用时，使用方式与 APP 相同，请参阅力控使用手册

搭建配合直线运动的恒力柔顺控制应用例程

在开始前，请正确配置传感器硬件和软件环境，包括正确设置传感器型号和负载。
一个典型的配合直线运动的恒力柔顺控制应用的搭建如下:

```
#include "JAKAZuRobot.h"
#include <iostream>
#include "Windows.h"
int main()
{
    //实例化机械臂控制类
    JAKAZuRobot demo;
    //登陆到机器人，需要将10.5.5.100替换为自己控制器的IP
    std::cout << demo.login_in("10.5.5.100") << std::endl;
    //机器人上电
    demo.power_on();
    //机器人上使能
    demo.enable_robot();

    CartesianPose pos = { 0, 100, 0, 0, 0, 0 };

    //开启传感器，初次开启前建议在APP中检查传感器是否可以正常工作
     demo.set_torque_sensor_mode(1);
    //设置恒力柔顺参数，此处仅开启z方向柔顺力控，设置阻尼为50，恒力为5N
    demo.set_admit_ctrl_config(2, 1, 50, 5, 0, 0);
    demo.set_admit_ctrl_config(0, 0, 0, 0, 0, 0);
    demo.set_admit_ctrl_config(1, 0, 0, 0, 0, 0);
    demo.set_admit_ctrl_config(3, 0, 0, 0, 0, 0);
    demo.set_admit_ctrl_config(4, 0, 0, 0, 0, 0);
    demo.set_admit_ctrl_config(5, 0, 0, 0, 0, 0);

    //开启恒力柔顺控制，必须在进入伺服运动模式后再开启恒力柔顺控制。
    //注意: 1.7.1.37FC，1.7.2及以上版本控制器，如果需要初始化，必须先下发参数1，0然后等待1秒再下发0，1。不可同时下发1，1
    demo.set_compliant_type(1, 1);
    //注意：力控不会在set_compliant_type后立即生效，开启后直到接收到第一条运动指令才会被触发生效，此后即使运动指令执行结束，只要没有主动关闭力控，柔顺控制将一直处于生效状态
    demo.linear_move(&pos, ABS, TRUE, 30); //如果希望在机器人不实际运动的情况下触发力控，可以发送目标位置为当前机器人位置的运动指令
    //机器人从当前位置开始，向y正方向以30mm/s的速度运动150mm
    //同时因为已经开启了恒力柔顺控制，将控制z方向的接触力为5N
    Sleep(2000); //没有主动关闭力控前，即使是等待期间力控仍然生效
    //关闭恒力柔顺控制，必须在退出伺服运动模式后再关闭恒力柔顺控制
```

```
    //set_compliant_type指令与disable_force_control必须同时使用以安全关闭力控
    demo.set_compliant_type(0, 0);
    demo.disable_force_control();
    //关闭机器人并退出登录
    demo.power_off();
    demo.disable_robot();
    demo.login_out();
    return 0;
}
```