JAKA

SDK & Force Control Function

Quick Start

Before using any force control function through SDK, please be sure to refer to the *JAKA ROBOTICS User Manual for Force Control Product* to correctly build the software and hardware environment, understand the detailed information of the product functions and strictly follow the usage precautions.

Contents below only briefly mentions the API interfaces involved in common force control functions. Please refer to JAKA SDK manual for detailed and complete descriptions.

This guide is applicable to controller from versions 1.7.0.38 to versions 1.7.2.

Introduction to API related to force control function

The basic APIs involved in building force control applications through SDK include:

General:
errno_t get_robot_status (RobotStatus * status); Get robot status monitoring data
errno_t set_torque_sensor_mode (int sensor_mode); Turn on or off the end force sensor
errno_t set_ft_ctrl_frame (const int ftFrame); Set the force control coordinate system (the factory default setting is the tool coordinate system)
errno_t set_compliant_type (int sensor_compensation , int compliance_type); Set the force control type and zero calibration (initialization) options
errno_t disable_force_control (); Closing force smooth control
errno_t set_torque_sensor_filter (const float torque_sensor_filter); Set the end force sensor low-pass filter cutoff frequency (factory default setting is 0, filter is not enabled)

Constant compliant force control mode:
errno_t set_admit_ctrl_config (int axis, int opt, int ftUser , int ftConstant , int ftNnormal Track, int ftReboundFK); Set constant compliant force control parameters

Speed compliant force control mode (cannot be used with servo mode, no longer supported in versions 1.7.2 and above):

errno_t set_vel_compliant_ctrl (const VelCom * vel_cfg); Set the speed compliance control parameters

errno_t set_compliance_condition (const FTxyz * ft); Set the speed compliance control condition

Tool drag teaching mode (cannot be used with any motion mode):

errno_t enable_admittance_ctrl (const int enable_flag); tool drag on and off

The above force control functions are the same with the content in JAKA Zu App. Please be sure to refer to the *JAKA ROBOTICS User Manual for Force Control Product* for better understanding of the detailed meanings of these functions and precautions for use.

It should be noted that the constant force compliance mode must be used in conjunction with motion instructions to be effective, including:

General Movement Instructions:

errno_t joint_move (const JointValue * joint_pos , MoveMode move_mode,BOOL is_block , double speed); robot joint movement

errno_t linear_move (const CartesianPose * end_pos , MoveMode move_mode , BOOL is_block , double speed); robot linear movement

Servo motion mode:

errno_t servo_move_enable (BOOL enable); Enable the robot servo position control mode

errno_t servo_j (const JointValue * joint_pos , MoveMode move_mode); robot joint space servo mode movement

errno_t servo_p (const CartesianPose * cartesian_pose , MoveMode move_mode); robot Cartesian space servo mode movement

For detailed parameter descriptions of the above APIs, please refer to JAKA SDK Manual.

In addition, SDK also supports:

errno_t set_torsenosr_brand (int sensor_brand); Set the force sensor model

errno_t set_torque_sensor_soft_limit (const FTxyz torque_sensor_soft_limit); Setting the soft limits of the force sensor

errno_t set_torq_sensor_tool_payload (const PayLoad * payload); Set the sensor end load

errno_t start_torq_sensor_payload_identify (const JointValue * joint_pos); Start identifying the tool end load

These APIs have the same functions as JAKA Zu App. For usage, users can refer to *JAKA ROBOTICS User Manual for Force Control Product*. In general, it is recommended to use the app for these configurations rather than SDK, unless the sensor limit or load information needs to be automatically updated.

Note:
Before using any force control function, the sensor model and load information must be correctly configured, whether through the app or through SDK. The sensor model and load information will be permanently saved after being set once, unless the robot system is set again or reset.

When using SDK to set the sensor model, there must be a 2-second interval between the set_torsenosr_brand command and the set_torque_sensor_mode command to turn on the sensor to ensure that the model has been set successfully. Otherwise, the sensor cannot be turned on successfully.

Build a constant compliant force control example with servo motion

Before you begin, please correctly configure the sensor hardware and software environment, including correctly setting the sensor model and payload.

A typical servo motion constant compliant force control application is built as follows:

```
#include " JAKAZuRobot.h "
#include <iostream>
#include " Windows.h "
int main( )
{
    // Instantiate the robot arm control class
    JAKAZuRobot demo;
    // Log in to the robot and replace 10.5.5.100 with the IP address of your controller
    std:: cout << demo.login_in ( "10.5.5.100" ) << std:: endl ;
    // Power on the robot
    demo.power_on ( ) ;
    // Enable on the robot
    demo.enable_robot ( ) ;

    // Turn on the sensor. Before turning it on for the first time, it is recommended to check whether the sensor can work normally in the app
     demo.set_torque_sensor_mode ( 1 ) ;
     // Set the constant compliant force parameters. Here, only the z-direction compliance control is turned on, the damping is 50, and the constant force is 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);

 // The robot enters servo motion mode
demo.servo_move_enable (1);
Sleep(100);
 // Turn on constant force control. You must turn on constant force control after entering the servo motion mode .
 //Note: For controllers with 1.7.1.37FC and 1.7.2 or higher versions, if initialization is required, you must first send parameters 1, 0 and then wait for 1 second before sending 0, 1. You cannot send 1, 1 at the same time.
demo.set_compliant_type (1, 1);

// The robot starts from the current position and moves in the positive x direction at a speed of 1 2.5 mm /s for 4 seconds
// At the same time , because the constant compliant force control has been turned on, the contact force in the z direction is set to 5N
CartesianPose cart;
RobotStatus status; // Robot status detection structure, including force information
for ( int i = 0; i < 500; i + +) {
    // Real-time monitoring and printing of z-direction force information
    demo.get_robot_status (&status );
    std:: cout << "Force at z direction is: " << status.torq_sensor_monitor_data.actTorque [2] << std:: endl ;
    // Send motion instructions to the robot to move 0.1 mm in the positive direction of the x-axis every 8 milliseconds
    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);
}

// The robot exits servo motion mode
demo.servo_move_enable (0) ;
// Turn off constant compliant force control. You must exit the servo motion mode before turning off constant compliant force control
// The set_compliant_type command and disable_force_control must be used together to safely disable force control
demo.set_compliant_ type (0, 0) ;
demo.disable _force_control () ;
// Shut down the robot and log out
demo.power_off ( ) ;
demo.disable_robot ( ) ;
demo.login_out ( );
```

```
    return 0;
}
```

Warning:
When constant compliant force control is used with general motion mode, if the motion command selects non-blocking mode, constant compliant force control must be turned off after all motion commands are completed or terminated to avoid uncontrolled movement of the robot.

When using constant compliant force control with general motion mode, if you want to turn off force control, you must use set_compliant_type (1, 0) or set_compliant_type (0, 0) to set compliant type must be set to 0 and the disable_force_control () command must be called continuously to shut down the robot safely. Otherwise, the robot may move uncontrollably.

Note:
In servo motion mode, the robot controller does not perform motion planning. The robot will continuously execute motion point instructions at a cycle of 8ms. The user needs to plan the trajectory and speed by himself and continuously send position instructions to the robot. For safety reasons, when using the force control function, the distance between two adjacent position instructions is recommended to be within 3 mm and 0.3 °.

Constant compliant force control must be turned on after the servo motion mode is turned on and must be exited after the servo motion mode is exited.

When using constant compliant force control with general motion mode, the usage is the same as the app. Please refer to *JAKA ROBOTICS User Manual for Force Control Product.*

Example with linear motion

Before you begin, please correctly configure the sensor hardware and software environment, including correctly setting the sensor model and payload.
A typical application of constant compliant force control with linear motion is built as follows:

```
#include " JAKAZuRobot.h "
#include <iostream>
#include " Windows.h "
int main( )
{
    // Instantiate the robot arm control class
    JAKAZuRobot demo;
    // Log in to the robot and replace 10.5.5.100 with the IP address of your controller
    std:: cout << demo.login_in ( "10.5.5.100" ) << std:: endl ;
    // Power on the robot
    demo.power_on ( ) ;
    // Enable on the robot
    demo.enable_robot ( ) ;
```

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

    // Turn on the sensor. Before turning it on for the first time, it is recommended to check whether the sensor can work normally in the app
     demo.set_torque_sensor_mode ( 1 ) ;
     // Set the constant force compliance parameters. Here, only the z-direction compliance control is turned on, the damping is 50, and the constant force is 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) ;

     //Turn on constant force smooth control. You must turn on constant force smooth control after entering the servo motion mode .
     //Note: For controllers with 1.7.1.37FC and 1.7.2 or higher versions, if initialization is required, you must first send parameters 1, 0 and then wait for 1 second before sending 0, 1. You cannot send 1, 1 at the same time.
    demo.set_compliant_type (1, 1) ;
    //Note: force control will not take effect immediately after set_compliant_type is enabled. It will not take effect until the first motion command is received. After that, even if the motion command is executed, as long as force control is not actively turned off, compliant control will always be in effect. state
    demo.linear_move (&pos, ABS, TRUE, 30); //If you want to trigger force control without the robot actually moving, you can send a motion command with the target position being the current robot position
    //The robot starts from the current position and moves 150mm in the positive y direction at a speed of 30 mm/s
    //At the same time, because the constant compliant force control has been turned on, the contact force in the z direction is controlled to 5N
    Sleep(2000); //Before actively closing the force control, the force control is still effective even during the waiting period
    //Turn off constant compliant force control. You must exit the servo motion mode before turning off constant compliant force control
    //The set_compliant_type command and disable_force_control must be used together to safely disable force control
    demo.set_compliant_ type (0, 0);
    demo.disable _force_control ();
    // Shut down the robot and log out
    demo.power_off ();
    demo.disable_robot ();
    demo.login_out ();
    return 0;
}
```