myCobot API

Before using the following function interfaces, import our API library first. Otherwise it is impossible to run successfully. To download and import the library, refer to 9.2 Compiling and Running of a MycobotCpp Case.

1 Preconditions for controlling the robot arm

1.1 MyCobot(string port, int baud=115200)
Function: to instantiate MyCobot
Return value: none
Parameter description: Parameter 1: serial port number ("COM*" on Windows (such as COM30)
Parameter 2: baud rate (default 115200)
Note: If you want to call the following API, you need to instantiate it first

1.2 Open()
Function: to open the serial port
Return value: none
Parameter description: none
Note: for communication with the robot arm, the serial port should be opened first

1.3 Close()
Function: to close the serial port
Return value: none
Parameter description: none
Note: When the program ends, it is better to close the serial port

2 Overall running status of Robot

2.1 PowerOn()
Function: to power up the robot arm
Return value: none
Parameter description: none
Note: After the robot arm is powered up, it cannot be moved manually

2.2 PowerOff()
Function: to cut off teh power for the robot arm
Return value: none
parameter description: none
Note: After the robot arm is powered up, if you want to move it manually, you may use this api to cut off the power of the robot arm

3 MDI program control mode

3.1 SendOneAngle(int jointNo, int angle, int speed)
Function: to send single joint angle
Return value: none
Parameter description: Parameter 1: joint number (1 - 6); Parameter 2: angle (ranging from -170° to 170°); Parameter 3: speed (0-100)

3.2 GetAngles()
Function: to get the angles of all joints
Return value: return an array of int type, int[], length: 6
Description of parameters: none

3.3 SendAngles(int[] angles, int speed)
Function: to send the angles of all joints
Return value: none
Parameter description: Parameter 1: the angles of all joints (ranging from -170° to 170°); Parameter 2: speed (0-100)

3.4 GetCoords()
Function: to get all coordinates
Return value: return an array of int type, int[], length: 6
Description of parameters: none

3.5 SendCoords(int[] coords, int speed, int mode)
Function: to send multi-parameter coordinates
Return value: none
Parameter description: Parameter 1: All coordinates (The values of X, Y and Z range from -300 to 300.00mm; the values of RX, RY and RZ range from -180 to 180); Parameter 2: speed (0-100); Parameter 3: mode (0 - angular, and 1 - linear)
3.6 SendOneCoord(int coord, int value, int speed)
Function: to send single parameter coordinates
Return value: none
Parameter description: Parameter 1: coordinate number (1-6(x, y, z, rx, ry, rz)); Parameter 2: coordinate (The values of X, Y and Z range from -300 to 300.00mm; the values of RX, RY and RZ range from -180 to 180); Parameter 3: speed (0-100)

4 Atom end IO control

4.1 SetDigitalOut(byte pin_number, byte pin_signal)
Function: to set the high and low levels of output io
Return value: none
Parameter description: Parameter 1: pin number (atom output pin number), and Parameter 2: state (0-low level, and 1-high level)

4.2 GetDigitalIn(byte pin_number)
Function: to get the state of input io
Return value: pin state (0-low level, and 1-high level)
Parameter description: pin number (atom input pin number)

4.3 setGripperValue(byte angle, byte speed)
Function: to control the adaptive gripper
Return value: none
Parameter description: Parameter 1: gripper opening and closing angles (ranging from 0 to 100; 0–closed; 100-maximum open angle); Parameter 2: gripper opening and closing speeds (0-100)

4.4 SetElectricGriper(byte open)
Function: to control the electric gripper
Return value: none
Parameter description: gripper switch state (0-off, and 1-on)

4.5 getGripperValue()
Function: to get the angle of the adaptive gripper
Return value: int type, return gripper angle (0 - closed; 100 - maximum open angle)
Parameter description: none

5 Stand M5Stack-basic IO control

5.1 SetBasicOut(byte pin_number, byte pin_signal)
Function: to set the high and low levels of output io
Return value: none
Parameter description: Parameter 1: pin number (atom output pin number), and Parameter 2: state (0-low level, and 1-high level)

5.2 GetBasicIn(byte pin_number)
Function: to get state of input io
Return value: pin state (0-low level, and 1-high level)
Parameter description: pin number (basic input pin number)