MyCobot API

When 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 8.2 Compiling and running of MycobotCpp.

1 Instantiate MyCobot

1.1 **I()**
Function: Instantiate MyCobot
return value: MyCobot type, single instance of myCobot object
parameter description: none
Note: When calling the following API, it is not necessary to do separate instantiation, and just call this API

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 call this API

2.3 **SetFreeMoveMode(bool free_move = true)**
Function: to set free movement mode
Return value: none
parameter description: enable or disable free movement; true – open free movement, and false – close free automatic
Note: After free movement is enabled, you may move the robot arm manually; and at the same time the light on the atom will turn yellow, and it will turn green when it is turned off

2.4 **IsFreeMoveMode()**
Function: to check whether the current free movement mode
Return value: bool type, true - free movement is enabled, false - free movement is not enabled
parameter description: none

2.5 **IsControllerConnected()**
Function: to check whether the system is normal
return value: none
parameter description: bool type, if false is returned, the robot arm cannot be controlled

3 MDI program control mode

3.1 **IsInPosition(const Coords& coords, bool is_linear = true)**
Function: to check whether the robot arm reaches a specified point (angle or coordinate)
Return value: bool type; if false is returned , it means failing to reach the specified point; and if true is returned, it means having reached the specified point
Parameter description: Parameter 1: all angles or coordinates; Parameter 2: 0 or 1 (Coordinate is 1 (true), and angle is 0 (false))

3.2 **IsMoving()**
Function: to detect whether the robot arm is moving
Return value: bool type, true - moving, false - not moving
Parameter description: none

3.3 **WriteAngle(Joint joint, double value, int speed = DefaultSpeed)**
Function: to send single joint angle
Return value: none
parameter description: Parameter 1: joint number (1-6), Parameter 2: angle (-170°- 170°), and Parameter 3: speed (0-100), and the default is 30

3.4 **GetAngles()**
Function: to get the angles of all joints
Return value: Angles type
Parameter description: none

3.5 **WriteAngles(const Angles& angles, int speed = DefaultSpeed)**
Function: to send the angles of all joints
Return value: none
parameter description: Parameter 1: all angles (std::array, and ranging from -170° to 170°), and Parameter 2: speed (0-100) with a default of 30

3.6 **WriteCoord(Axis axis, double value, int speed = DefaultSpeed)**
Function: to send single parameter coordinate
Return value: none
parameter description: Parameter 1: coordinate number (Axis enumeration type, int: 1-6 (X-RZ)), Parameter 2: coordinate (X, Y and Z range form -300 to 300.00mm; RX, RY and RZ range from -180 to 180), and Parameter 3: speed (0-100), and the default is 30

3.7 **GetCoords()**
Function: to get all coordinates
Return value: Coords type
Parameter description: none

3.8 **WriteCoords(const Coords& coords, int speed = DefaultSpeed)**
Function: to send all coordinates
Return value: none
Parameter description: Parameter 1: coordinate (X, Y and Z range form -300 to 300.00mm; RX, RY and RZ range from -180 to 180), and Parameter 2: speed (0-100), the default is 30

3.9 **StopRobot()**
Function: to stop the movement of the robot arm. When the robot arm is moving, you may stop it by calling this API
Return value: none
Parameter description: none

4 Running Auxiliary Information

4.1 **GetSpeed()**
Function: to get the speed of the robot arm
Return value: int type, the movement speed of the robot arm (0-100)
Parameter description: none

4.2 **SetSpeed(int percentage)**
Function: to set the movement speed of the robot arm
Return value: none
Parameter description: the movement speed of the robot arm (0-100)

4.3 **GetJointMin(Joint joint)**
Function: to read the minimum angle of a joint
Return value: double type, the minimum angle (the minimum angle that the joint can reach)
Parameter description: joint number (1-6)

4.4 **GetJointMax(Joint joint)**
Function: to read the maximum angle of a joint
Return value: double type, the maximum angle (the maximum angle that the joint can reach)
Parameter description: joint number (1-6)

4.5 **SleepSecond(unsigned time)**
Function: to wait
Return value: none
Parameter description: time unit is second

5 JOG mode and operation

5.1 **JogCoord(Axis axis, int direction, int speed = DefaultSpeed)**
Function: to make the robot arm move in the direction of a coordinate axis
Return value: none
Parameter description: Parameter 1: coordinate number (1-6, xyz rx ry rz), Parameter 2: direction (1 - positive direction, and 0 - negative direction), and Parameter 3: speed (Default is 30, ranging from 0 to 100). Notice: This API will make the robot arm move in the forward and reverse directions of the coordinate axis, and it will stop moving after reaching a limited position or when JogStop is called during the movement

5.2 **JogAngle(Joint joint, int direction, int speed = DefaultSpeed)**
Function: to make a joint move until Jogstop or reaching a limited position
Return value: none
Parameter description: Parameter 1: joint number (1-6), Parameter 2: direction (1 - positive direction, and 0 - negative direction), and Parameter 3: speed (Default 30, ranging from 0 to 100). Notice: This API will make the robot arm move in the forward and reverse directions, and it will stop moving after reaching the limited position or when JogStop is called during the movement

5.3 **JogCoordAbsolute(Axis axis, double value, int speed = DefaultSpeed)**
Function: to make a coordinate to move to a specified coordinate
Return value: none
Parameter description: Parameter 1: coordinate number (1-6, x y z rx ry rz), Parameter 2: coordinate (X, Y and Z range form -300 to 300.00mm; and RX, RY and RZ range from -180 to 180), and Parameter 3: speed (with a default of 30 and ranging from 0 to 100)

5.4 **JogAngleAbsolute(Joint joint, double value, int speed = DefaultSpeed)**
Function: to make a joint move to a specified angle
Return value: none
Parameter description: Parameter 1: joint number (1-6), Parameter 2: angle (ranging from -170 to 170), and Parameter 3: speed (with a default of 30 and ranging from 0 to 100)

5.5 **JogCoordIncrement(Axis axis, double increment, int speed = DefaultSpeed)**
Function: to make a coordinate move by a set coordinate increment
Return value: none
Parameter description: Parameter 1: coordinate number (1-6, xyz rx ry rz), Parameter 2: coordinate increment value, and Parameter 3: speed (with a default of 30, and ranging from 0 to 100). Notice: The robot arm performs discrete motion; for example, the current coordinate of X-axis is 100, and the incremental value is 50. After the movement, the coordinate of X-axis will be 150

5.6 **JogAngleIncrement(Joint joint, double increment, int speed = DefaultSpeed)**
Function: to make a joint move by a set angle increment
Return value: none
Parameter description: Parameter 1: joint number (1-6), Parameter 2: joint increment value, and Parameter 3: speed (with a default of 30, and ranging from 0 to 100). Notice: The robot arm performs discrete motion; for example, the current coordinate of the joint 1 is -100, and the incremental value is 50. After the movement, the coordinate of the joint 1 will be 50

6 Atom end IO control

6.1 **SetDigitalOut(int pin_number, int 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)

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

6.3 **SetGriper(int open)**
Function: to control the adaptive gripper
Return value: none
Parameter description: gripper switch state (0-off, and 1-on)

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

7 Stand M5Stack-basic IO control

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

7.2 **GetBasicIn(int 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)