

API 文档

May 22, 2026

Contents

1	Aubo SDK	9
2	专题索引	13
2.1	专题	13
3	目录结构	15
3.1	目录	15
4	命名空间索引	17
4.1	命名空间列表	17
5	继承关系索引	19
5.1	类继承关系	19
6	类索引	21
6.1	类列表	21
7	文件索引	23
7.1	文件列表	23
8	专题文档	25
8.1	AuboApi (主入口)	25
8.1.1	详细描述	25
8.1.2	函数说明	26
8.2	AxisInterface (外部轴)	29
8.2.1	详细描述	31
8.2.2	函数说明	31
8.3	GripperInterface (夹爪)	38
8.3.1	详细描述	39
8.3.2	函数说明	39
8.4	Math (数学工具)	45
8.4.1	详细描述	46
8.4.2	函数说明	46
8.5	RegisterControl (寄存器操作)	59
8.5.1	详细描述	62
8.5.2	函数说明	62
8.6	RobotInterface(机器人模块)	105
8.6.1	详细描述	106
8.6.2	函数说明	106
8.6.3	ForceControl(力控模块)	109
8.6.4	IoControl (IO 输入输出控制)	140
8.6.5	MotionControl (运动规划与控制)	212
8.6.6	RobotAlgorithm (机器人算法工具)	299
8.6.7	RobotConfig (机器人配置管理)	327
8.6.8	RobotManage (机器人生命周期管理)	376
8.6.9	RobotState (机器人状态查询)	398
8.7	RuntimeMachine (运行时管理)	443
8.7.1	详细描述	445

8.7.2	函数说明	445
8.8	Serial (串口通信)	463
8.8.1	详细描述	464
8.8.2	函数说明	464
8.9	Socket (socket 网络通信)	469
8.9.1	详细描述	470
8.9.2	函数说明	470
8.10	SyncMove (同步运动控制和轴组管理)	477
8.10.1	详细描述	479
8.10.2	函数说明	479
8.11	SystemInfo (系统信息)	490
8.11.1	详细描述	490
8.11.2	函数说明	491
8.12	Trace (日志与弹窗)	494
8.12.1	详细描述	494
8.12.2	函数说明	494
9	目录说明	497
9.1	include/aubo 目录参考	497
9.2	doc 目录参考	498
9.3	include/aubo/error_stack 目录参考	498
9.4	include 目录参考	499
9.5	include/aubo/robot 目录参考	499
10	命名空间文档	501
10.1	arcs 命名空间参考	501
10.2	arcs::common_interface 命名空间参考	501
10.2.1	类型定义说明	504
10.2.2	枚举类型说明	507
10.2.3	函数说明	513
10.3	arcs::error_stack 命名空间参考	514
10.3.1	枚举类型说明	514
10.3.2	函数说明	514
11	类说明	517
11.1	arcs::common_interface::AuboApi 类参考	517
11.1.1	详细描述	518
11.1.2	构造及析构函数说明	518
11.1.3	类成员变量说明	518
11.2	arcs::common_interface::AuboException 类参考	518
11.2.1	详细描述	519
11.2.2	构造及析构函数说明	519
11.2.3	成员函数说明	520
11.2.4	类成员变量说明	521
11.3	arcs::common_interface::AxisInterface 类参考	521
11.3.1	详细描述	523
11.3.2	构造及析构函数说明	523
11.3.3	类成员变量说明	523
11.4	arcs::common_interface::CircleParameters 结构体参考	523
11.4.1	详细描述	524
11.4.2	类成员变量说明	524
11.5	arcs::common_interface::Enveloping 结构体参考	525
11.5.1	详细描述	526
11.5.2	类成员变量说明	526
11.6	arcs::common_interface::ForceControl 类参考	526
11.6.1	详细描述	528
11.6.2	构造及析构函数说明	528
11.6.3	类成员变量说明	529
11.7	arcs::common_interface::GripperInterface 类参考	529

11.7.1 详细描述	530
11.7.2 构造及析构函数说明	530
11.7.3 成员函数说明	530
11.7.4 类成员变量说明	532
11.8 arcs::common_interface::GripperStatus 结构体参考	532
11.8.1 详细描述	533
11.8.2 类成员变量说明	533
11.9 arcs::common_interface::IoControl 类参考	534
11.9.1 详细描述	538
11.9.2 构造及析构函数说明	538
11.9.3 类成员变量说明	538
11.10 arcs::common_interface::Math 类参考	538
11.10.1 详细描述	540
11.10.2 构造及析构函数说明	540
11.10.3 类成员变量说明	540
11.11 arcs::common_interface::MotionControl 类参考	540
11.11.1 详细描述	545
11.11.2 构造及析构函数说明	545
11.11.3 成员函数说明	545
11.11.4 类成员变量说明	547
11.12 arcs::common_interface::RegisterControl 类参考	547
11.12.1 详细描述	550
11.12.2 构造及析构函数说明	550
11.12.3 成员函数说明	551
11.12.4 类成员变量说明	551
11.13 arcs::common_interface::RobotAlgorithm 类参考	552
11.13.1 详细描述	554
11.13.2 构造及析构函数说明	554
11.13.3 成员函数说明	554
11.13.4 类成员变量说明	555
11.14 arcs::common_interface::RobotConfig 类参考	555
11.14.1 详细描述	558
11.14.2 构造及析构函数说明	558
11.14.3 成员函数说明	559
11.14.4 类成员变量说明	559
11.15 arcs::common_interface::RobotInterface 类参考	559
11.15.1 详细描述	560
11.15.2 构造及析构函数说明	560
11.15.3 类成员变量说明	560
11.16 arcs::common_interface::RobotManage 类参考	560
11.16.1 详细描述	562
11.16.2 构造及析构函数说明	562
11.16.3 类成员变量说明	562
11.17 arcs::common_interface::RobotMsg 结构体参考	562
11.17.1 详细描述	563
11.17.2 类成员变量说明	563
11.18 arcs::common_interface::RobotSafetyParameterRange 结构体参考	564
11.18.1 详细描述	565
11.18.2 构造及析构函数说明	566
11.18.3 类成员变量说明	566
11.19 arcs::common_interface::RobotState 类参考	571
11.19.1 详细描述	574
11.19.2 构造及析构函数说明	574
11.19.3 成员函数说明	574
11.19.4 类成员变量说明	575
11.20 arcs::common_interface::RtdeRecipe 结构体参考	575
11.20.1 详细描述	576
11.20.2 类成员变量说明	576

11.21arcs::common_interface::RuntimeMachine 类参考	576
11.21.1 详细描述	578
11.21.2 构造及析构函数说明	579
11.21.3 成员函数说明	579
11.21.4 类成员变量说明	579
11.22arcs::common_interface::Serial 类参考	579
11.22.1 详细描述	580
11.22.2 构造及析构函数说明	580
11.22.3 类成员变量说明	580
11.23arcs::common_interface::Socket 类参考	580
11.23.1 详细描述	581
11.23.2 构造及析构函数说明	581
11.23.3 类成员变量说明	581
11.24arcs::common_interface::SpiralParameters 结构体参考	582
11.24.1 详细描述	582
11.24.2 类成员变量说明	582
11.25arcs::common_interface::SyncMove 类参考	583
11.25.1 详细描述	584
11.25.2 构造及析构函数说明	585
11.25.3 类成员变量说明	585
11.26arcs::common_interface::SystemInfo 类参考	585
11.26.1 详细描述	585
11.26.2 构造及析构函数说明	585
11.26.3 类成员变量说明	586
11.27arcs::common_interface::Trace 类参考	586
11.27.1 详细描述	586
11.27.2 构造及析构函数说明	586
11.27.3 成员函数说明	586
11.27.4 类成员变量说明	587
11.28arcs::common_interface::TrajConfig 结构体参考	587
11.28.1 详细描述	588
11.28.2 类成员变量说明	588
11.29arcs::common_interface::VibrationRecalibrationParameter 结构体参考	589
11.29.1 详细描述	589
11.29.2 类成员变量说明	589
11.30arcs::common_interface::WObjectData 结构体参考	590
11.30.1 详细描述	591
11.30.2 类成员变量说明	591
12 文件说明	593
12.1 doc/sdk_overview.md 文件参考	593
12.2 include/aubo/aubo_api.h 文件参考	593
12.2.1 详细描述	593
12.3 aubo_api.h	594
12.4 include/aubo/axis_interface.h 文件参考	598
12.5 axis_interface.h	599
12.6 include/aubo/error_stack/error_stack.h 文件参考	607
12.6.1 详细描述	608
12.6.2 宏定义说明	608
12.7 error_stack.h	612
12.8 include/aubo/error_stack/hal_error.h 文件参考	613
12.8.1 详细描述	614
12.8.2 宏定义说明	614
12.9 hal_error.h	616
12.10include/aubo/error_stack/rtm_error.h 文件参考	622
12.10.1 详细描述	623
12.10.2 宏定义说明	623
12.11rtm_error.h	623

12.12include/aubo/error_stack/system_error.h 文件参考	627
12.12.1 详细描述	627
12.12.2 宏定义说明	627
12.13system_error.h	628
12.14include/aubo/gripper_interface.h 文件参考	628
12.14.1 详细描述	629
12.15gripper_interface.h	629
12.16include/aubo/math.h 文件参考	633
12.16.1 详细描述	634
12.17math.h	634
12.18include/aubo/register_control.h 文件参考	646
12.18.1 详细描述	648
12.18.2 枚举类型说明	648
12.19register_control.h	648
12.20include/aubo/robot/force_control.h 文件参考	689
12.20.1 详细描述	690
12.21force_control.h	691
12.22include/aubo/robot/io_control.h 文件参考	721
12.22.1 详细描述	722
12.23io_control.h	723
12.24include/aubo/robot/motion_control.h 文件参考	789
12.24.1 详细描述	790
12.25motion_control.h	790
12.26include/aubo/robot/robot_algorithm.h 文件参考	869
12.26.1 详细描述	870
12.27robot_algorithm.h	870
12.28include/aubo/robot/robot_config.h 文件参考	895
12.28.1 详细描述	896
12.29robot_config.h	896
12.30include/aubo/robot/robot_manage.h 文件参考	940
12.30.1 详细描述	941
12.31robot_manage.h	941
12.32include/aubo/robot/robot_state.h 文件参考	963
12.32.1 详细描述	964
12.33robot_state.h	965
12.34include/aubo/robot_interface.h 文件参考	1005
12.34.1 详细描述	1006
12.35robot_interface.h	1006
12.36include/aubo/runtime_machine.h 文件参考	1011
12.36.1 详细描述	1012
12.37runtime_machine.h	1012
12.38include/aubo/serial.h 文件参考	1031
12.38.1 详细描述	1032
12.39serial.h	1032
12.40include/aubo/socket.h 文件参考	1037
12.40.1 详细描述	1038
12.41socket.h	1038
12.42include/aubo/sync_move.h 文件参考	1046
12.42.1 详细描述	1047
12.43sync_move.h	1047
12.44include/aubo/system_info.h 文件参考	1059
12.44.1 详细描述	1060
12.45system_info.h	1060
12.46include/aubo/trace.h 文件参考	1064
12.46.1 详细描述	1065
12.47trace.h	1065
12.48include/aubo/type_def.h 文件参考	1068
12.48.1 详细描述	1072

12.48.2 宏定义说明	1072
12.49type_def.h	1078
索引	1091

Chapter 1

Aubo SDK

当前 API 文档下载地址: [Aubo SDK API 文档.pdf](#)

概述

Aubo SDK 提供了一套完整的接口来控制和管理机器人系统。SDK 采用分层架构设计,通过[AuboApi \(主入口\)](#)作为主入口, 提供对各个功能模块的访问。

目录

- 快速开始
- SDK 架构
- 核心模块
- 使用示例
- 命名空间
- 错误处理
- 支持的编程语言

快速开始

基本使用流程

```
// 1. 获取主 API 接口
auto rpc_cli = std::make_shared<RpcClient>();

// 2. 获取机器人列表
auto robot_names = rpc_cli->getRobotNames();
auto robot_name = robot_names.front();

// 3. 获取机器人接口
auto robot = rpc_cli->getRobotInterface(robot_name);

// 4. 获取具体功能模块
auto motion = robot->getMotionControl();
auto state = robot->getRobotState();
```

SDK 架构

层次结构

```
AuboApi (主入口)
├── 系统级接口
│   ├── SystemInfo (系统信息)
│   └── RuntimeMachine (运行时管理)
```

```

RegisterControl (寄存器操作)
Math (数学工具)
SyncMove (同步运动控制和轴组管理)
Trace (日志与弹窗)
设备接口
RobotInterface (机器人)
  RobotConfig (机器人配置管理)
  MotionControl (运动规划与控制)
  ForceControl (力控功能)
  IoControl (IO 输入输出控制)
  RobotAlgorithm (机器人算法工具)
  RobotManage (机器人生命周期管理)
  RobotState (机器人状态查询)
AxisInterface (外部轴)
GripperInterface (夹爪)
通信接口
Socket (网络通信)
Serial (串口通信)

```

核心模块

功能	接口	文件	主要功能说明
机器人运动	MotionControl	robot/motion_control.h	路径规划, 关节运动、直线运动、圆弧运动控制, 速度与加速度参数设置, 等效半径配置
机器人配置	RobotConfig	robot/robot_config.h	机器人本体参数管理, 工具坐标系 (TCP) 设置, 用户坐标系设置, 负载及质心参数配置
机器人状态	RobotState	robot/robot_state.h	实时关节角度与末端位姿查询, 机器人运行状态获取, 错误与异常状态查询
机器人管理	RobotManage	robot/robot_manage.h	机器人上电、使能、断电控制, 急停、暂停与恢复操作, 运行模式切换
IO 控制	IoControl	robot/io_control.h	数字 IO 读写, 模拟 IO 读写, IO 参数与模式配置
力控	ForceControl	robot/force_control.h	力/扭矩传感器配置, 力控运动接口, 阻抗控制与柔顺控制
算法工具	RobotAlgorithm	robot/robot_algorithm.h	坐标系变换计算, 正/逆运动学求解, 碰撞检测与空间约束判断
外部轴	AxisInterface	axis_interface.h	外部轴上电与启动管理, 外部轴运动控制, 外部轴安装位姿与标定设置
夹爪	GripperInterface	gripper_interface.h	夹爪设备扫描与添加, 夹爪连接与状态管理, 夹爪开合与动作控制
系统信息	SystemInfo	system_info.h	控制器软件版本查询, 硬件信息获取, 系统整体运行状态
运行时	RuntimeMachine	runtime_machine.h	任务创建与删除, 脚本运行与管理, 断点设置与程序控制
寄存器	RegisterControl	register_control.h	布尔寄存器、整数寄存器、浮点寄存器的读写操作
数学工具	Math	math.h	矩阵运算, 四元数运算, 坐标与姿态变换相关数学工具
网络通信	Socket	socket.h	TCP 客户端通信接口, 网络连接与会话管理
串口通信	Serial	serial.h	串口数据读写, 波特率及通信参数配置

功能	接口	文件	主要功能说明
同步运动	SyncMove	sync_move.h	多设备同步运动控制，同步时序管理，外部轴轴组管理
告警日志	Trace	trace.h	告警信息查询，系统日志记录与导出

使用示例

获取机器人状态

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot = rpc_cli->getRobotInterface("rob1");
auto state = robot->getRobotState();
```

```
// 获取关节角度
auto joint_angles = state->getJointState();
```

执行运动

```
auto motion = robot->getMotionControl();

// 设置运动参数
motion->setEqradius(0.8);

// 执行关节运动
std::vector<double> target_pose = {0, -1.57, 1.57, 0, 0, 0};
motion->moveJoint(target_pose, 0.5, 0.5);
```

控制 IO

```
auto io = robot->getIoControl();

// 设置数字输出
io->setStandardDigitalOutput(0, true);

// 读取数字输入
bool input_state = io->getStandardDigitalInput(0);
```

命名空间

所有接口都定义在以下命名空间中：

```
namespace arcs {
namespace common_interface {
    // 所有接口定义
}
}
```

错误处理

大多数接口方法返回整数错误码，其中 0 表示成功，负数表示错误。某些接口可能抛出 `AuboException` 异常。

支持的编程语言

- C++
- Python
- Lua
- JSON-RPC

相关文件

- [type_def.h](#) - 类型定义
- [global_config.h](#) - 全局配置
- [error_stack/](#) - 错误处理相关

Chapter 2

专题索引

2.1 专题

这里是所有专题及其简介:

AuboApi (主入口)	25
AxisInterface (外部轴)	29
GripperInterface (夹爪)	38
Math (数学工具)	45
RegisterControl (寄存器操作)	59
RobotInterface(机器人模块)	105
ForceControl(力控模块)	109
IoControl (IO 输入输出控制)	140
MotionControl (运动规划与控制)	212
RobotAlgorithm (机器人算法工具)	299
RobotConfig (机器人配置管理)	327
RobotManage (机器人生命周期管理)	376
RobotState (机器人状态查询)	398
RuntimeMachine (运行时管理)	443
Serial (串口通信)	463
Socket (socket 网络通信)	469
SyncMove (同步运动控制和轴组管理)	477
SystemInfo (系统信息)	490
Trace (日志与弹窗)	494

Chapter 3

目录结构

3.1 目录

aubo	497
error_stack	498
error_stack.h	607
hal_error.h	613
rtm_error.h	622
system_error.h	627
robot	499
force_control.h	689
io_control.h	721
motion_control.h	789
robot_algorithm.h	869
robot_config.h	895
robot_manage.h	940
robot_state.h	963
aubo_api.h	593
axis_interface.h	598
gripper_interface.h	628
math.h	633
register_control.h	646
robot_interface.h	1005
runtime_machine.h	1011
serial.h	1031
socket.h	1037
sync_move.h	1046
system_info.h	1059
trace.h	1064
type_def.h	1068
doc	498
error_stack	498
error_stack.h	607
hal_error.h	613
rtm_error.h	622
system_error.h	627
include	499
aubo	497
error_stack	498
error_stack.h	607
hal_error.h	613
rtm_error.h	622
system_error.h	627

robot	499
force_control.h	689
io_control.h	721
motion_control.h	789
robot_algorithm.h	869
robot_config.h	895
robot_manage.h	940
robot_state.h	963
aubo_api.h	593
axis_interface.h	598
gripper_interface.h	628
math.h	633
register_control.h	646
robot_interface.h	1005
runtime_machine.h	1011
serial.h	1031
socket.h	1037
sync_move.h	1046
system_info.h	1059
trace.h	1064
type_def.h	1068
robot	499
force_control.h	689
io_control.h	721
motion_control.h	789
robot_algorithm.h	869
robot_config.h	895
robot_manage.h	940
robot_state.h	963

Chapter 4

命名空间索引

4.1 命名空间列表

这里列出了所有命名空间定义，附带简要说明:

arcs	501
arcs::common_interface	501
arcs::error_stack	514

Chapter 5

继承关系索引

5.1 类继承关系

此继承关系列表按字典顺序粗略的排序:

arcs::common_interface::AuboApi	517
arcs::common_interface::AxisInterface	521
arcs::common_interface::CircleParameters	523
arcs::common_interface::Enveloping	525
std::exception	
arcs::common_interface::AuboException	518
arcs::common_interface::ForceControl	526
arcs::common_interface::GripperInterface	529
arcs::common_interface::GripperStatus	532
arcs::common_interface::IoControl	534
arcs::common_interface::Math	538
arcs::common_interface::MotionControl	540
arcs::common_interface::RegisterControl	547
arcs::common_interface::RobotAlgorithm	552
arcs::common_interface::RobotConfig	555
arcs::common_interface::RobotInterface	559
arcs::common_interface::RobotManage	560
arcs::common_interface::RobotMsg	562
arcs::common_interface::RobotSafetyParameterRange	564
arcs::common_interface::RobotState	571
arcs::common_interface::RtdeRecipe	575
arcs::common_interface::RuntimeMachine	576
arcs::common_interface::Serial	579
arcs::common_interface::Socket	580
arcs::common_interface::SpiralParameters	582
arcs::common_interface::SyncMove	583
arcs::common_interface::SystemInfo	585
arcs::common_interface::Trace	586
arcs::common_interface::TrajConfig	587
arcs::common_interface::VibrationRecalibrationParameter	589
arcs::common_interface::WObjectData	590

Chapter 6

类索引

6.1 类列表

这里列出了所有类、结构、联合以及接口定义等，并附带简要说明:

arcs::common_interface::AuboApi	517
arcs::common_interface::AuboException	
自定义异常类 AuboException	518
arcs::common_interface::AxisInterface	521
arcs::common_interface::CircleParameters	
圆周运动参数定义	523
arcs::common_interface::Enveloping	525
arcs::common_interface::ForceControl	526
arcs::common_interface::GripperInterface	529
arcs::common_interface::GripperStatus	532
arcs::common_interface::IoControl	534
arcs::common_interface::Math	538
arcs::common_interface::MotionControl	540
arcs::common_interface::RegisterControl	547
arcs::common_interface::RobotAlgorithm	552
arcs::common_interface::RobotConfig	555
arcs::common_interface::RobotInterface	559
arcs::common_interface::RobotManage	560
arcs::common_interface::RobotMsg	562
arcs::common_interface::RobotSafetyParameterRange	564
arcs::common_interface::RobotState	571
arcs::common_interface::RtdeRecipe	
RTDE 菜单	575
arcs::common_interface::RuntimeMachine	576
arcs::common_interface::Serial	579
arcs::common_interface::Socket	580
arcs::common_interface::SpiralParameters	582
arcs::common_interface::SyncMove	583
arcs::common_interface::SystemInfo	585
arcs::common_interface::Trace	586
arcs::common_interface::TrajConfig	
用于负载辨识的轨迹配置	587
arcs::common_interface::VibrationRecalibrationParameter	589
arcs::common_interface::WObjectData	590

Chapter 7

文件索引

7.1 文件列表

这里列出了所有文件，并附带简要说明：

include/aubo/ aubo_api.h	
机器人及外部轴等控制API 接口，如获取机器人列表、获取系统信息等等	593
include/aubo/ axis_interface.h	598
include/aubo/ gripper_interface.h	
通用夹爪接口	628
include/aubo/ math.h	
数学方法接口，如欧拉角与四元数转换、位姿的加减运算	633
include/aubo/ register_control.h	
寄存器操作接口，用于三个模块之间的数据交换功能	646
include/aubo/ robot_interface.h	
机器人API 接口	1005
include/aubo/ runtime_machine.h	
Script interpreter runtime interface, allows pausing the script interpreter and setting/removing breakpoints	1011
include/aubo/ serial.h	
串口通信	1031
include/aubo/ socket.h	
Socket 通信	1037
include/aubo/ sync_move.h	
同步运行	1046
include/aubo/ system_info.h	
获取系统信息接口，如接口板的版本号、示教器软件的版本号	1059
include/aubo/ trace.h	
向控制器日志系统注入日志方面的接口	1064
include/aubo/ type_def.h	
数据类型的定义	1068
include/aubo/error_stack/ error_stack.h	
汇总错误码	607
include/aubo/error_stack/ hal_error.h	
定义硬件抽象层的错误码	613
include/aubo/error_stack/ rtm_error.h	
运行时错误码	622
include/aubo/error_stack/ system_error.h	
系统错误码	627
include/aubo/robot/ force_control.h	
689	
include/aubo/robot/ io_control.h	
IO 控制接口	721
include/aubo/robot/ motion_control.h	
运动控制接口	789

include/aubo/robot/ robot_algorithm.h	
机器人算法相关的对外接口	869
include/aubo/robot/ robot_config.h	
获取机器人配置接口, 如获取DH 参数、碰撞等级、安装位姿等等	895
include/aubo/robot/ robot_manage.h	
机器人管理接口, 如上电、启动、拖动示教模式等	940
include/aubo/robot/ robot_state.h	
获取机器人状态接口, 如关节速度、关节角度、固件/硬件版本	963

Chapter 8

专题文档

8.1 AuboApi (主入口)

AuboApi

函数

- [MathPtr arcs::common_interface::AuboApi::getMath \(\)](#)
Math (数学工具) 获取纯数学相关接口
- [SystemInfoPtr arcs::common_interface::AuboApi::getSystemInfo \(\)](#)
SystemInfo (系统信息) 获取系统信息
- [RuntimeMachinePtr arcs::common_interface::AuboApi::getRuntimeMachine \(\)](#)
RuntimeMachine (运行时管理) 获取运行时接口
- [RegisterControlPtr arcs::common_interface::AuboApi::getRegisterControl \(\)](#)
RegisterControl (寄存器操作) 对外寄存器接口
- [std::vector< std::string > arcs::common_interface::AuboApi::getRobotNames \(\)](#)
获取机器人列表
- [RobotInterfacePtr arcs::common_interface::AuboApi::getRobotInterface \(const std::string &name\)](#)
RobotInterface(机器人模块) 根据名字获取 *RobotInterfacePtr* 接口
- [std::vector< std::string > arcs::common_interface::AuboApi::getAxisNames \(\)](#)
获取外部轴列表
- [AxisInterfacePtr arcs::common_interface::AuboApi::getAxisInterface \(const std::string &name\)](#)
AxisInterface (外部轴) 获取外部轴接口
- [SocketPtr arcs::common_interface::AuboApi::getSocket \(\)](#)
Socket (socket 网络通信) 获取 *socket*
- [SerialPtr arcs::common_interface::AuboApi::getSerial \(\)](#)
Serial (串口通信) 获取 *Serial* 串口
- [SyncMovePtr arcs::common_interface::AuboApi::getSyncMove \(const std::string &name\)](#)
SyncMove (同步运动控制和轴组管理) 获取同步运动接口
- [TracePtr arcs::common_interface::AuboApi::getTrace \(const std::string &name\)](#)
Trace (日志与弹窗) 获取告警信息接口
- [GripperInterfacePtr arcs::common_interface::AuboApi::getGripperInterface \(\)](#)
GripperInterface (夹爪) 获取通用夹爪接口

8.1.1 详细描述

AuboApi

8.1.2 函数说明

getAxisInterface()

```
AxisInterfacePtr arcs::common_interface::AuboApi::getAxisInterface (
    const std::string & name)
AxisInterface (外部轴) 获取外部轴接口
```

参数

<i>name</i>	

返回

getAxisNames()

```
std::vector< std::string > arcs::common_interface::AuboApi::getAxisNames ()
获取外部轴列表
```

getGripperInterface()

```
GripperInterfacePtr arcs::common_interface::AuboApi::getGripperInterface ()
GripperInterface (夹爪) 获取通用夹爪接口
```

返回

GripperInterfacePtr 对象的指针

getMath()

```
MathPtr arcs::common_interface::AuboApi::getMath ()
Math (数学工具) 获取纯数学相关接口
```

返回

MathPtr 对象的指针

Python 函数原型

```
getMath(self: pyaubo_sdk.AuboApi) -> pyaubo_sdk.Math
```

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
MathPtr ptr = rpc_cli->getMath();
```

getRegisterControl()

```
RegisterControlPtr arcs::common_interface::AuboApi::getRegisterControl ()
RegisterControl (寄存器操作) 对外寄存器接口
```

返回

RegisterControlPtr 对象的指针

Python 函数原型

```
getRegisterControl(self: pyaubo_sdk.AuboApi) -> pyaubo_sdk.RegisterControl
```

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
RegisterControlPtr ptr = rpc_cli->getRegisterControl();
```

getRobotInterface()

```
RobotInterfacePtr arcs::common_interface::AuboApi::getRobotInterface (
    const std::string & name)
```

RobotInterface(机器人模块) 根据名字获取 RobotInterfacePtr 接口

参数

<i>name</i>	机器人名字

返回

RobotInterfacePtr 对象的指针

Python 函数原型

```
getRobotInterface(self: pyaubo_sdk.AuboApi, arg0: str) -> pyaubo_sdk.RobotInterface
```

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
RobotInterfacePtr ptr = rpc_cli->getRobotInterface(robot_name);
```

getRobotNames()

```
std::vector< std::string > arcs::common_interface::AuboApi::getRobotNames ()
```

获取机器人列表

返回

机器人列表

Python 函数原型

```
getRobotNames(self: pyaubo_sdk.AuboApi) -> List[str]
```

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "getRobotNames", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": ["rob1"]}
```

getRuntimeMachine()

`RuntimeMachinePtr` arcs::common_interface::AuboApi::getRuntimeMachine ()

`RuntimeMachine` (运行时管理) 获取运行时接口

返回

`RuntimeMachinePtr` 对象的指针

Python 函数原型

`getRuntimeMachine(self: pyaubo_sdk.AuboApi) -> pyaubo_sdk.RuntimeMachine`

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
RuntimeMachinePtr ptr = rpc_cli->getRuntimeMachine();
```

getSerial()

`SerialPtr` arcs::common_interface::AuboApi::getSerial ()

`Serial` (串口通信) 获取Serial 串口

返回

`SerialPtr` 对象的指针

Python 函数原型

`getSerial(self: pyaubo_sdk.AuboApi) -> arcs::common_interface::Serial`

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
SerialPtr ptr = rpc_cli->getSerial();
```

getSocket()

`SocketPtr` arcs::common_interface::AuboApi::getSocket ()

`Socket` (socket 网络通信) 获取 socket

返回

`SocketPtr` 对象的指针

Python 函数原型

`getSocket(self: pyaubo_sdk.AuboApi) -> arcs::common_interface::Socket`

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
SocketPtr ptr = rpc_cli->getSocket();
```

getSyncMove()

`SyncMovePtr` arcs::common_interface::AuboApi::getSyncMove (const std::string & name)

`SyncMove` (同步运动控制和轴组管理) 获取同步运动接口

返回

`SyncMovePtr` 对象的指针

getSystemInfo()

`SystemInfoPtr` `arcs::common_interface::AuboApi::getSystemInfo ()`
`SystemInfo` (系统信息) 获取系统信息

返回

`SystemInfoPtr` 对象的指针

Python 函数原型

`getSystemInfo(self: pyaubo_sdk.AuboApi) -> pyaubo_sdk.SystemInfo`

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
SystemInfoPtr ptr = rpc_cli->getSystemInfo();
```

getTrace()

`TracePtr` `arcs::common_interface::AuboApi::getTrace (`
`const std::string & name)`
`Trace` (日志与弹窗) 获取告警信息接口

返回

`TracePtr` 对象的指针

8.2 AxisInterface (外部轴)

外部轴API 接口

函数

- `int arcs::common_interface::AxisInterface::poweronExtAxis ()`
通电
- `int arcs::common_interface::AxisInterface::poweroffExtAxis ()`
断电
- `int arcs::common_interface::AxisInterface::enableExtAxis ()`
使能
- `int arcs::common_interface::AxisInterface::setExtAxisMountingPose (const std::vector< double > &pose)`
设置外部轴的安装位姿 (相对于世界坐标系)
- `int arcs::common_interface::AxisInterface::moveExtJoint (double pos, double v, double a, double duration)`
运动到指定点, 旋转或者平移
- `int arcs::common_interface::AxisInterface::speedExtJoint (double v, double a, double duration)`
制定目标运动速度
- `int arcs::common_interface::AxisInterface::stopExtJoint (double a)`
停止外部轴运动
- `int arcs::common_interface::AxisInterface::getExtAxisType ()`
获取外部轴的类型 0 代表是旋转 1 代表平移
- `int arcs::common_interface::AxisInterface::setExtAxisType (int type)`
设置外部轴的类型
- `AxisModeType arcs::common_interface::AxisInterface::getAxisModeType ()`
获取当前外部轴的状态

- `std::vector< double > arcs::common_interface::AxisInterface::getExtAxisMountingPose ()`
获取外部轴安装位姿
- `std::vector< double > arcs::common_interface::AxisInterface::getExtAxisPose ()`
获取相对于安装坐标系的位姿, 外部轴可能为变位机或者导轨
- `double arcs::common_interface::AxisInterface::getExtAxisPosition ()`
获取外部轴位置
- `double arcs::common_interface::AxisInterface::getExtAxisVelocity ()`
获取外部轴运行速度
- `int arcs::common_interface::AxisInterface::setExtAxisVelocity (double velocity)`
设置外部轴运行速度
- `double arcs::common_interface::AxisInterface::getExtAxisAcceleration ()`
获取外部轴运行加速度
- `int arcs::common_interface::AxisInterface::setExtAxisAcceleration (double acc)`
设置外部轴运行加速度
- `double arcs::common_interface::AxisInterface::getExtAxisCurrent ()`
获取外部轴电流
- `double arcs::common_interface::AxisInterface::getExtAxisTemperature ()`
获取外部轴温度
- `double arcs::common_interface::AxisInterface::getExtAxisBusVoltage ()`
获取外部轴电压
- `double arcs::common_interface::AxisInterface::getExtAxisBusCurrent ()`
获取外部轴电流
- `double arcs::common_interface::AxisInterface::getExtAxisMaxPosition ()`
获取外部轴最大位置 (物理极限)
- `int arcs::common_interface::AxisInterface::setExtAxisMaxPositionLimit (double q)`
设置外部轴最大位置 (软件/算法限制)
- `double arcs::common_interface::AxisInterface::getExtAxisMaxPositionLimit ()`
获取外部轴最大位置 (软件/算法限制)
- `double arcs::common_interface::AxisInterface::getExtMinPosition ()`
获取外部轴最小位置 (物理极限)
- `int arcs::common_interface::AxisInterface::setExtMinPositionLimit (double q)`
设置外部轴最小位置 (软件/算法限制)
- `double arcs::common_interface::AxisInterface::getExtMinPositionLimit ()`
获取外部轴最小位置 (软件/算法限制)
- `double arcs::common_interface::AxisInterface::getExtAxisMaxVelocity ()`
获取外部轴最大速度 (物理极限)
- `int arcs::common_interface::AxisInterface::setExtAxisMaxVelocityLimit (double v)`
设置外部轴最大速度 (软件/算法限制)
- `double arcs::common_interface::AxisInterface::getExtAxisMaxVelocityLimit ()`
获取外部轴最大速度 (软件/算法限制)
- `double arcs::common_interface::AxisInterface::getExtAxisMaxAcceleration ()`
获取外部轴最大加速度 (物理极限)
- `int arcs::common_interface::AxisInterface::setExtAxisMaxAccelerationLimit (double acc)`
设置外部轴最大加速度 (软件/算法限制)
- `double arcs::common_interface::AxisInterface::getExtAxisMaxAccelerationLimit ()`
获取外部轴最大加速度 (软件/算法限制)
- `int arcs::common_interface::AxisInterface::followAnotherAxis (const std::string &target_name, double phase, double err)`
跟踪另一个外部轴的运动 (禁止运动过程中使用)
- `int arcs::common_interface::AxisInterface::stopFollowAnotherAxis ()`
stopFollowAnotherAxis (禁止运动过程中使用)

- int `arcs::common_interface::AxisInterface::getErrorCode ()`
`getErrorCode`(获取外部轴错误码)
- int `arcs::common_interface::AxisInterface::clearAxisError ()`
重置外部轴错误
- int `arcs::common_interface::AxisInterface::setExtAxisZero ()`
将当前外部轴位置设置为零位
- int `arcs::common_interface::AxisInterface::setReductionRatio (double ratio)`
设置外部轴减速比
- double `arcs::common_interface::AxisInterface::getReductionRatio ()`
获取外部轴减速比

8.2.1 详细描述

外部轴API 接口

8.2.2 函数说明

`clearAxisError()`

```
int arcs::common_interface::AxisInterface::clearAxisError ()
重置外部轴错误
```

返回

`enableExtAxis()`

```
int arcs::common_interface::AxisInterface::enableExtAxis ()
使能
```

`followAnotherAxis()`

```
int arcs::common_interface::AxisInterface::followAnotherAxis (
    const std::string & target_name,
    double phase,
    double err)
跟踪另一个外部轴的运动 (禁止运动过程中使用)
```

参数

<i>target_name</i>	目标的外部轴名字
<i>phase</i>	相位差
<i>err</i>	跟踪运行的最大误差

返回

`getAxisModeType()`

```
AxisModeType arcs::common_interface::AxisInterface::getAxisModeType ()
获取当前外部轴的状态
```

返回

当前外部轴的状态

getErrorCode()

```
int arcs::common_interface::AxisInterface::getErrorCode ()  
getErrorCode(获取外部轴错误码)
```

getExtAxisAcceleration()

```
double arcs::common_interface::AxisInterface::getExtAxisAcceleration ()  
获取外部轴运行加速度
```

返回

外部轴运行加速度

getExtAxisBusCurrent()

```
double arcs::common_interface::AxisInterface::getExtAxisBusCurrent ()  
获取外部轴电流
```

返回

外部轴电流

getExtAxisBusVoltage()

```
double arcs::common_interface::AxisInterface::getExtAxisBusVoltage ()  
获取外部轴电压
```

返回

外部轴电压

getExtAxisCurrent()

```
double arcs::common_interface::AxisInterface::getExtAxisCurrent ()  
获取外部轴电流
```

返回

外部轴电流

getExtAxisMaxAcceleration()

```
double arcs::common_interface::AxisInterface::getExtAxisMaxAcceleration ()  
获取外部轴最大加速度 (物理极限)
```

返回

外部轴最大加速度

getExtAxisMaxAccelerationLimit()

```
double arcs::common_interface::AxisInterface::getExtAxisMaxAccelerationLimit ()  
获取外部轴最大加速度 (软件/算法限制)
```

返回

外部轴最大加速度

getExtAxisMaxPosition()

```
double arcs::common_interface::AxisInterface::getExtAxisMaxPosition ()
```

获取外部轴最大位置 (物理极限)

返回

外部轴最大位置

getExtAxisMaxPositionLimit()

```
double arcs::common_interface::AxisInterface::getExtAxisMaxPositionLimit ()
```

获取外部轴最大位置 (软件/算法限制)

返回

外部轴最大位置

getExtAxisMaxVelocity()

```
double arcs::common_interface::AxisInterface::getExtAxisMaxVelocity ()
```

获取外部轴最大速度 (物理极限)

返回

外部轴最大速度

getExtAxisMaxVelocityLimit()

```
double arcs::common_interface::AxisInterface::getExtAxisMaxVelocityLimit ()
```

获取外部轴最大速度 (软件/算法限制)

返回

外部轴最大速度

getExtAxisMountingPose()

```
std::vector< double > arcs::common_interface::AxisInterface::getExtAxisMountingPose ()
```

获取外部轴安装位姿

返回

外部轴安装位姿

getExtAxisPose()

```
std::vector< double > arcs::common_interface::AxisInterface::getExtAxisPose ()
```

获取相对于安装坐标系的位姿，外部轴可能为变位机或者导轨

返回

相对于安装坐标系的位姿

getExtAxisPosition()

```
double arcs::common_interface::AxisInterface::getExtAxisPosition ()
```

获取外部轴位置

返回

外部轴位置

getExtAxisTemperature()

```
double arcs::common_interface::AxisInterface::getExtAxisTemperature ()
```

获取外部轴温度

返回

外部轴温度

getExtAxisType()

```
int arcs::common_interface::AxisInterface::getExtAxisType ()
```

获取外部轴的类型 0 代表是旋转 1 代表平移

getExtAxisVelocity()

```
double arcs::common_interface::AxisInterface::getExtAxisVelocity ()
```

获取外部轴运行速度

返回

外部轴运行速度

getExtMinPosition()

```
double arcs::common_interface::AxisInterface::getExtMinPosition ()
```

获取外部轴最小位置 (物理极限)

返回

外部轴最小位置

getExtMinPositionLimit()

```
double arcs::common_interface::AxisInterface::getExtMinPositionLimit ()
```

获取外部轴最小位置 (软件/算法限制)

返回

外部轴最小位置

getReductionRatio()

```
double arcs::common_interface::AxisInterface::getReductionRatio ()
```

获取外部轴减速比

返回

moveExtJoint()

```
int arcs::common_interface::AxisInterface::moveExtJoint (
    double pos,
    double v,
    double a,
    double duration)
```

运动到指定点, 旋转或者平移

参数

<i>pos</i>	
<i>v</i>	
<i>a</i>	
<i>duration</i>	

返回

poweroffExtAxis()

```
int arcs::common_interface::AxisInterface::poweroffExtAxis ()
```

断电

poweronExtAxis()

```
int arcs::common_interface::AxisInterface::poweronExtAxis ()
```

通电

setExtAxisAcceleration()

```
int arcs::common_interface::AxisInterface::setExtAxisAcceleration (
    double acc)
```

设置外部轴运行加速度

参数

<i>acc</i>	

返回

setExtAxisMaxAccelerationLimit()

```
int arcs::common_interface::AxisInterface::setExtAxisMaxAccelerationLimit (
    double acc)
```

设置外部轴最大加速度 (软件/算法限制)

参数

<i>acc</i>	

返回

setExtAxisMaxPositionLimit()

```
int arcs::common_interface::AxisInterface::setExtAxisMaxPositionLimit (  
    double q)  
设置外部轴最大位置 (软件/算法限制)
```

参数

q	

返回

setExtAxisMaxVelocityLimit()

```
int arcs::common_interface::AxisInterface::setExtAxisMaxVelocityLimit (  
    double v)  
设置外部轴最大速度 (软件/算法限制)
```

参数

v	

返回

setExtAxisMountingPose()

```
int arcs::common_interface::AxisInterface::setExtAxisMountingPose (  
    const std::vector< double > & pose)  
设置外部轴的安装位姿 (相对于世界坐标系)
```

参数

$pose$	

setExtAxisType()

```
int arcs::common_interface::AxisInterface::setExtAxisType (  
    int type)  
设置外部轴的类型
```

参数

$type$	0 代表是旋转 1 代表平移

setExtAxisVelocity()

```
int arcs::common_interface::AxisInterface::setExtAxisVelocity (  
    double velocity)  
设置外部轴运行速度
```

返回

setExtAxisZero()

```
int arcs::common_interface::AxisInterface::setExtAxisZero ()  
将当前外部轴位置设置为零位
```

返回

setExtMinPositionLimit()

```
int arcs::common_interface::AxisInterface::setExtMinPositionLimit (  
    double q)  
设置外部轴最小位置 (软件/算法限制)
```

参数

q	

返回

setReductionRatio()

```
int arcs::common_interface::AxisInterface::setReductionRatio (  
    double ratio)  
设置外部轴减速比
```

返回

speedExtJoint()

```
int arcs::common_interface::AxisInterface::speedExtJoint (  
    double v,  
    double a,  
    double duration)  
制定目标运动速度
```

参数

v	
a	

<i>duration</i>	

返回

stopExtJoint()

```
int arcs::common_interface::AxisInterface::stopExtJoint (
    double a)
    停止外部轴运动
```

参数

<i>a</i>	

返回

stopFollowAnotherAxis()

```
int arcs::common_interface::AxisInterface::stopFollowAnotherAxis ()
    stopFollowAnotherAxis(禁止运动过程中使用)
```

8.3 GripperInterface (夹爪)

通用夹爪API 接口

函数

- `std::vector< std::string > arcs::common_interface::GripperInterface::gripperGetSupportedModels ()`
获取支持的所有夹爪
- `ResultWithErrno2 arcs::common_interface::GripperInterface::gripperScanDevices (const std::string &model, const std::string &device_name)`
扫描该型号下的所有设备
- `std::vector< std::string > arcs::common_interface::GripperInterface::gripperGetNames ()`
获取已添加的夹爪
- `int arcs::common_interface::GripperInterface::gripperAdd (const std::string &name, const std::string &model)`
添加夹爪
- `int arcs::common_interface::GripperInterface::gripperDelete (const std::string &name)`
删除夹爪
- `int arcs::common_interface::GripperInterface::gripperRename (const std::string &name, const std::string &new_name)`
修改夹爪名
- `int arcs::common_interface::GripperInterface::gripperConnect (const std::string &name, const std::string &device_name)`
夹爪连接
- `int arcs::common_interface::GripperInterface::gripperDisconnect (const std::string &name)`

- 夹爪断开连接
- `bool arcs::common_interface::GripperInterface::gripperIsConnected (const std::string &name)`
夹爪是否连接
- `int arcs::common_interface::GripperInterface::gripperSetWorkMode (const std::string &name, int work_mode)`
设置工作模式
- `int arcs::common_interface::GripperInterface::gripperGetWorkMode (const std::string &name)`
获取工作模式
- `int arcs::common_interface::GripperInterface::gripperSetMountPose (const std::string &name, const std::vector< double > &pose, bool enable_collision)`
设置夹爪安装偏移
- `int arcs::common_interface::GripperInterface::gripperEnable (const std::string &name, bool enable)`
使能夹爪
- `bool arcs::common_interface::GripperInterface::gripperIsEnabled (const std::string &name)`
夹爪是否使能
- `int arcs::common_interface::GripperInterface::gripperSetPosition (const std::string &name, const double position)`
设置运动参数
- `int arcs::common_interface::GripperInterface::gripperMove (const std::string &name)`
开始运动
- `int arcs::common_interface::GripperInterface::gripperStop (const std::string &name)`
停止运动
- `std::string arcs::common_interface::GripperInterface::gripperGetHardwareVersion (const std::string &name)`
获取夹爪状态
- `int arcs::common_interface::GripperInterface::gripperResetSlaveId (const std::string &name, const int slave_id)`
重置 *modbus* 从站ID
- `int arcs::common_interface::GripperInterface::gripperGetStatusCode (const std::string &name)`
获取夹爪状态

8.3.1 详细描述

通用夹爪API 接口

8.3.2 函数说明

`gripperAdd()`

```
int arcs::common_interface::GripperInterface::gripperAdd (
    const std::string & name,
    const std::string & model)
```

添加夹爪

参数

<i>name</i>	用户自定义名字, 唯一
<i>model</i>	夹爪品牌及型号

返回

成功返回 0, 失败返回错误码

gripperConnect()

```
int arcs::common_interface::GripperInterface::gripperConnect (  
    const std::string & name,  
    const std::string & device_name)
```

夹爪连接

参数

<i>name</i>	
<i>device_name</i>	设备名

返回

成功返回 0，失败返回错误码

gripperDelete()

```
int arcs::common_interface::GripperInterface::gripperDelete (  
    const std::string & name)
```

删除夹爪

参数

<i>name</i>	用户自定义名字

返回

成功返回 0，失败返回错误码

gripperDisconnect()

```
int arcs::common_interface::GripperInterface::gripperDisconnect (  
    const std::string & name)
```

夹爪断开连接

参数

<i>name</i>	

返回

成功返回 0，失败返回错误码

gripperEnable()

```
int arcs::common_interface::GripperInterface::gripperEnable (  
    const std::string & name,  
    bool enable)
```

使能夹爪

参数

<i>name</i>	

返回

gripperGetHardwareVersion()

```
std::string arcs::common_interface::GripperInterface::gripperGetHardwareVersion (  
    const std::string & name)
```

获取夹爪状态

参数

<i>name</i>	

返回

注解

位置，速度，夹持力，旋转角度，旋转速度，旋转扭矩的单位与Set 接口一致

gripperGetNames()

```
std::vector< std::string > arcs::common_interface::GripperInterface::gripperGetNames ()
```

获取已添加的夹爪

返回

gripperGetStatusCode()

```
int arcs::common_interface::GripperInterface::gripperGetStatusCode (  
    const std::string & name)
```

获取夹爪状态

参数

<i>name</i>	

返回

状态

gripperGetSupportedModels()

```
std::vector< std::string > arcs::common_interface::GripperInterface::gripperGetSupportedModels ()
```

获取支持的所有夹爪

返回

gripperGetWorkMode()

```
int arcs::common_interface::GripperInterface::gripperGetWorkMode (
```

```
    const std::string & name)
```

获取工作模式

参数

<i>name</i>	

返回

返回工作模式

gripperIsConnected()

```
bool arcs::common_interface::GripperInterface::gripperIsConnected (
```

```
    const std::string & name)
```

夹爪是否连接

参数

<i>name</i>	

返回

已连接返回 true，未连接返回 false

gripperIsEnabled()

```
bool arcs::common_interface::GripperInterface::gripperIsEnabled (
```

```
    const std::string & name)
```

夹爪是否使能

参数

<i>name</i>	

返回

gripperMove()

```
int arcs::common_interface::GripperInterface::gripperMove (  
    const std::string & name)
```

开始运动

参数

<i>name</i>	

返回

gripperRename()

```
int arcs::common_interface::GripperInterface::gripperRename (  
    const std::string & name,  
    const std::string & new_name)
```

修改夹爪名

参数

<i>name</i>	
<i>new_name</i>	新名字

返回

成功返回 0，失败返回错误码

gripperResetSlaveId()

```
int arcs::common_interface::GripperInterface::gripperResetSlaveId (  
    const std::string & name,  
    const int slave_id)
```

重置 modbus 从站ID

参数

<i>name</i>	
<i>slave_id</i>	从站Id

返回

gripperScanDevices()

```
ResultWithErrno2 arcs::common_interface::GripperInterface::gripperScanDevices (  
    const std::string & model,  
    const std::string & device_name)
```

扫描该型号下的所有设备

参数

<i>model</i>	夹爪品牌型号
<i>device_name</i>	设备名

返回

成功返回设备列表和 0，失败返回错误码

gripperSetMountPose()

```
int arcs::common_interface::GripperInterface::gripperSetMountPose (  
    const std::string & name,  
    const std::vector< double > & pose,  
    bool enable_collision)  
设置夹爪安装偏移
```

参数

<i>name</i>	
<i>isVisibleed</i>	是否显示
<i>pose</i>	末端（不包括手指）相对法兰的位姿

返回

成功返回 true，失败返回 false

gripperSetPosition()

```
int arcs::common_interface::GripperInterface::gripperSetPosition (  
    const std::string & name,  
    const double position)  
设置运动参数
```

参数

<i>name</i>	
<i>position</i>	位置，单位：m/Pa
<i>velocity_percent</i>	速度，单位：，范围：[0, 1]
<i>force</i>	夹持力，单位：N
<i>angle</i>	旋转角度，单位：rad
<i>r_velocity_percent</i>	旋转速度，单位：，范围：[0, 1]
<i>torque_percent</i>	旋转扭矩，单位：，范围：[0, 1]

返回

gripperSetWorkMode()

```
int arcs::common_interface::GripperInterface::gripperSetWorkMode (
    const std::string & name,
    int work_mode)
    设置工作模式
```

参数

<i>name</i>	
<i>work_mode</i>	工作模式

返回

成功返回 0, 失败返回错误码

gripperStop()

```
int arcs::common_interface::GripperInterface::gripperStop (
    const std::string & name)
    停止运动
```

参数

<i>name</i>	

返回**8.4 Math (数学工具)**

数学工具API 接口

函数

- `std::vector< double > arcs::common_interface::Math::poseAdd` (`const std::vector< double > &p1,`
`const std::vector< double > &p2`)
- `std::vector< double > arcs::common_interface::Math::poseSub` (`const std::vector< double > &p1,`
`const std::vector< double > &p2`)
位姿相减
- `std::vector< double > arcs::common_interface::Math::interpolatePose` (`const std::vector< double`
`> &p1, const std::vector< double > &p2, double alpha`)
计算线性插值
- `std::vector< double > arcs::common_interface::Math::poseTrans` (`const std::vector< double > &pose←`
`_from, const std::vector< double > &pose_to`)
位姿变换
- `std::vector< double > arcs::common_interface::Math::poseTransInv` (`const std::vector< double >`
`&pose_to, const std::vector< double > &pose_from`)
姿态逆变换
- `std::vector< double > arcs::common_interface::Math::poseInverse` (`const std::vector< double >`
`&pose`)

- 获取位姿的逆
- double `arcs::common_interface::Math::poseDistance` (const std::vector< double > &p1, const std::vector< double > &p2)
 - 计算两个位姿的位置距离
- double `arcs::common_interface::Math::poseAngleDistance` (const std::vector< double > &p1, const std::vector< double > &p2)
 - 计算两个位姿的轴角距离
- bool `arcs::common_interface::Math::poseEqual` (const std::vector< double > &p1, const std::vector< double > &p2, double eps=5e-5)
 - 判断两个位姿是否相等
- std::vector< double > `arcs::common_interface::Math::transferRefFrame` (const std::vector< double > &F_b_a_old, const `Vector3d` &V_in_a, int type)
 - 姿态旋转
- std::vector< double > `arcs::common_interface::Math::rpyToQuaternion` (const std::vector< double > &rpy)
 - 欧拉角转四元数
- std::vector< double > `arcs::common_interface::Math::quaternionToRpy` (const std::vector< double > &quat)
 - 四元数转欧拉角
- `ResultWithErrno` `arcs::common_interface::Math::tcpOffsetIdentify` (const std::vector< std::vector< double > > &poses)
 - 四点法标定 TCP 偏移
- `ResultWithErrno` `arcs::common_interface::Math::calibrateCoordinate` (const std::vector< std::vector< double > > &poses, int type)
 - 三点法标定坐标系
- `ResultWithErrno` `arcs::common_interface::Math::calculateCircleFourthPoint` (const std::vector< double > &p1, const std::vector< double > &p2, const std::vector< double > &p3, int mode)
 - 根据圆弧的三个点, 计算出拟合成的圆的另一半圆弧的中间点位置
- std::vector< double > `arcs::common_interface::Math::forceTrans` (const std::vector< double > &pose_a_in_b, const std::vector< double > &force_in_a)
 -
- std::vector< double > `arcs::common_interface::Math::getDeltaPoseBySensorDistance` (const std::vector< double > &distances, double position, double radius, double track_scale)
 -
- std::vector< double > `arcs::common_interface::Math::deltaPoseTrans` (const std::vector< double > &pose_a_in_b, const std::vector< double > &ft_in_a)
 -
- std::vector< double > `arcs::common_interface::Math::deltaPoseAdd` (const std::vector< double > &pose_a_in_b, const std::vector< double > &v_in_b)
 -
- std::vector< double > `arcs::common_interface::Math::changePoseWithXYRef` (const std::vector< double > &pose_tar, const std::vector< double > &pose_ref)
 -
- std::vector< double > `arcs::common_interface::Math::homMatrixToPose` (const std::vector< double > &homMatrix)
 -
- std::vector< double > `arcs::common_interface::Math::poseToHomMatrix` (const std::vector< double > &pose)
 -

8.4.1 详细描述

数学工具API 接口

8.4.2 函数说明

calculateCircleFourthPoint()

```
ResultWithErrno arcs::common_interface::Math::calculateCircleFourthPoint (
    const std::vector< double > & p1,
    const std::vector< double > & p2,
```

```
const std::vector< double > & p3,
int mode)
```

根据圆弧的三个点, 计算出拟合成的圆的另一半圆弧的中间点位置

参数

<i>p1</i>	圆弧的起始点
<i>p2</i>	圆弧的中间点
<i>p3</i>	圆弧的结束点
<i>mode</i>	当 mode 等于 1 的时候, 表示需要对姿态进行圆弧规划; 当 mode 等于 0 的时候, 表示不需要对姿态进行圆弧规划

返回

拟合成的圆的另一半圆弧的中间点位置和计算结果是否有效

Lua 函数原型

```
calculateCircleFourthPoint(p1: table, p2: table, p3: table, mode: int)
```

Lua 示例

```
center_pose, cal_result = calculateCircleFourthPoint({0.5488696249770836,-0.1214996547187204,0.2631931199112321,-3.14159198038469,-3.673205103150083e-06,1.570796326792424}, {0.5488696249770835,-0.1214996547187207,0.3599720701808493,-3.14159198038469,-3.6732051029273e-06,1.570796326792423}, {0.5488696249770836,-0.0389996547187214,0.3599720701808496,-3.141591980384691,-3.673205102557476e-06,1.570796326792422},1)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "Math.calculateCircleFourthPoint", "params": [[0.5488696249770836,-0.1214996547187204,0.2631931199112321,-3.14159198038469,-3.673205103150083e-06,1.570796326792424], [0.5488696249770835,-0.1214996547187207,0.3599720701808493,-3.14159198038469,-3.6732051029273e-06,1.570796326792423], [0.5488696249770836,-0.0389996547187214,0.3599720701808496,-3.141591980384691,-3.673205102557476e-06,1.570796326792422],1], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": [[0.5488696249770837,-0.031860179583911546,0.27033259504604207,-3.1415919803846903,-3.67320510285378e-06,1.570796326792423],1]}
```

calibrateCoordinate()

```
ResultWithErrno arcs::common_interface::Math::calibrateCoordinate (
    const std::vector< std::vector< double > > & poses,
    int type)
```

三点法标定坐标系

参数

<i>poses</i>	三个点的位姿集合

<i>type</i>	类型:
0 - oxy	原点 x 轴正方向 xy 平面 (y 轴正方向)
1 - oxz	原点 x 轴正方向 xz 平面 (z 轴正方向)
2 - oyz	原点 y 轴正方向 yz 平面 (z 轴正方向)
3 - oyx	原点 y 轴正方向 yx 平面 (x 轴正方向)
4 - ozx	原点 z 轴正方向 zx 平面 (x 轴正方向)
5 - ozy	原点 z 轴正方向 zy 平面 (y 轴正方向)

返回

坐标系标定结果和标定结果是否有效

Lua 函数原型

calibrateCoordinate(poses: table, type: int) -> table, number

Lua 示例

```
p1 = {0.55462,0.06219,0.37175,-3.142,0.0,1.580}
p2 = {0.63746,0.11805,0.37175,-3.142,0.0,1.580}
p3 = {0.40441,0.28489,0.37174,-3.142,0.0,1.580}
p = {p1,p2,p3}
coord_pose, cal_result = calibrateCoordinate(p,0)
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "Math.calibrateCoordinate", "params": [[[0.55462,0.06219,0.37175,-3.142,0.0,1.580], [0.63746,0.11805,0.37175,-3.142,0.0,1.580],[0.40441,0.28489,0.37174,-3.142,0.0,1.580]],0], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": [[0.55462,0.06219,0.37175,-3.722688983883945e-05,-1.6940658945086007e-21,0.5932768162455785],0] }
```

changePoseWithXYRef()

```
std::vector< double > arcs::common_interface::Math::changePoseWithXYRef (
    const std::vector< double > & pose_tar,
    const std::vector< double > & pose_ref)
changePoseWithXYRef: 修改 pose_tar 的 x y 轴方向, 尽量与 pose_ref 一致,
```

参数

<i>pose_tar</i>	需要修改的目标位姿
<i>pose_ref</i>	参考位姿

返回

修改后的位姿, 采用 pose_tar 的 xyz 坐标和 z 轴方向

deltaPoseAdd()

```
std::vector< double > arcs::common_interface::Math::deltaPoseAdd (
    const std::vector< double > & pose_a_in_b,
    const std::vector< double > & v_in_b)
```

addDeltaPose: 计算以给定速度变换单位时间后的位姿

参数

<i>pose_a_in_b</i>	当前时刻 a 相对于 b 的位姿
<i>v_in_b</i>	当前时刻 a 坐标系的速度在 b 的描述

返回

pose_in_b, 单位时间后的位姿在 b 的描述

deltaPoseTrans()

```
std::vector< double > arcs::common_interface::Math::deltaPoseTrans (
    const std::vector< double > & pose_a_in_b,
    const std::vector< double > & ft_in_a)
```

changeFTFrame: 变换力和力矩的参考坐标系

参数

<i>pose_a_in_b</i>	a 坐标系在 b 坐标系的位姿
<i>ft_in_a</i>	作用在 a 点的力和力矩在 a 坐标系的描述

返回

ft_in_b, 作用在 b 点的力和力矩在 b 坐标系的描述

forceTrans()

```
std::vector< double > arcs::common_interface::Math::forceTrans (
    const std::vector< double > & pose_a_in_b,
    const std::vector< double > & force_in_a)
```

forceTrans: 变换力和力矩的参考坐标系 $\text{force\_in\_b} = \text{pose\_a\_in\_b} * \text{force\_in\_a}$

参数

<i>pose_a_in_b</i>	a 坐标系在 b 坐标系的位姿
<i>force_in_a</i>	力和力矩在 a 坐标系的描述

返回

force_in_b, 力和力矩在 b 坐标系的描述

getDeltaPoseBySensorDistance()

```
std::vector< double > arcs::common_interface::Math::getDeltaPoseBySensorDistance (
    const std::vector< double > & distances,
    double position,
    double radius,
    double track_scale)
```

通过距离计算工具坐标系下的位姿增量

参数

<i>distances</i>	N 个距离, $N \geq 3$
<i>position</i>	距离参考轨迹的保持高度
<i>radius</i>	传感器中心距离末端 tcp 的等效半径
<i>track_scale</i>	跟踪比例, 设置范围 (0, 1], 1 表示跟踪更快

返回

基于工具坐标系的位姿增量

homMatrixToPose()

```
std::vector< double > arcs::common_interface::Math::homMatrixToPose (
    const std::vector< double > & homMatrix)
```

homMatrixToPose: 由齐次变换矩阵得到位姿

参数

<i>homMatrix</i>	4*4 齐次变换矩阵, 输入元素采用横向排列

返回

对应的位姿

interpolatePose()

```
std::vector< double > arcs::common_interface::Math::interpolatePose (
    const std::vector< double > & p1,
    const std::vector< double > & p2,
    double alpha)
```

计算线性插值

参数

<i>p1</i>	起点的TCP 位姿
<i>p2</i>	终点的TCP 位姿

<i>alpha</i>	系数, 当 $0 < \alpha < 1$, 返回 $p1$ 和 $p2$ 两点直线的之间靠近 $p1$ 端且占总路径比例为 α 的点; 例如当 $\alpha=0.3$, 返回的是靠近 $p1$ 那端, 总路径的百分之 30 的点; 当 $\alpha > 1$, 返回 $p2$; 当 $\alpha < 0$, 返回 $p1$;

返回

插值计算结果

Python 函数原型

```
interpolatePose(self: pyaubo_sdk.Math, arg0: List[float], arg1: List[float], arg2: float) -> List[
float]
```

Lua 函数原型

```
interpolatePose(p1: table, p2: table, alpha: number) -> table
```

Lua 示例

```
pose_interpolate = interpolatePose({0.2, 0.2, 0.4, 0, 0, 0},{0.2, 0.2, 0.6, 0, 0, 0},0.5)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"Math.interpolatePose","params":[[0.2, 0.2, 0.4, 0, 0, 0],[0.2, 0.2, 0.6, 0, 0, 0],0.5],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.2,0.2,0.5,0.0,-0.0,0.0]}
```

poseAdd()

```
std::vector< double > arcs::common_interface::Math::poseAdd (
    const std::vector< double > & p1,
    const std::vector< double > & p2)
```

poseAngleDistance()

```
double arcs::common_interface::Math::poseAngleDistance (
    const std::vector< double > & p1,
    const std::vector< double > & p2)
```

计算两个位姿的轴角距离

参数

<i>p1</i>	位姿 1
<i>p2</i>	位姿 2

返回

轴角距离

Lua 函数原型

```
poseAngleDistance(p1: table, p2: table) -> number
```

Lua 示例

```
pose__angle__distance = poseAngleDistance({0.1, 0.3, 0.1, 0.3142, 0.0, 1.571},{0.2, 0.5, 0.6, 0, -0.172, 0.0})
```

poseDistance()

```
double arcs::common_interface::Math::poseDistance (
    const std::vector< double > & p1,
    const std::vector< double > & p2)
```

计算两个位姿的位置距离

参数

<i>p1</i>	位姿 1
<i>p2</i>	位姿 2

返回

两个位姿的位置距离

Lua 函数原型

```
poseDistance(p1: table, p2: table) -> number
```

Lua 示例

```
pose__distance = poseDistance({0.1, 0.3, 0.1, 0.3142, 0.0, 1.571},{0.2, 0.5, 0.6, 0, -0.172, 0.0})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"Math.poseDistance","params":[[0.1, 0.3, 0.1, 0.3142, 0.0, 1.571],[0.2, 0.5, 0.6, 0, -0.172, 0.0]],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0.5477225575051661}
```

poseEqual()

```
bool arcs::common_interface::Math::poseEqual (
    const std::vector< double > & p1,
    const std::vector< double > & p2,
    double eps = 5e-5)
```

判断两个位姿是否相等

参数

<i>p1</i>	位姿 1

<i>p2</i>	位姿 2
<i>eps</i>	误差

返回

相等返回 true, 反之返回 false

Lua 函数原型

```
poseEqual(p1: table, p2: table, eps: number) -> boolean
```

Lua 示例

```
pose_equal = poseEqual({0.1, 0.3, 0.1, 0.3142, 0.0, 1.571},{0.1, 0.3, 0.1, 0.3142, 0.0, 1.5711},0.01)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"Math.poseEqual","params":[[0.1, 0.3, 0.1, 0.3142, 0.0, 1.571],[0.1, 0.3, 0.1, 0.3142, 0.0, 1.5711],0.01]],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0.0}
```

poseInverse()

```
std::vector< double > arcs::common_interface::Math::poseInverse (
    const std::vector< double > & pose)
```

获取位姿的逆

参数

<i>pose</i>	工具位姿 (空间向量)

返回

工具位姿的逆转换 (空间向量)

Python 函数原型

```
poseInverse(self: pyaubo_sdk.Math, arg0: List[float]) -> List[float]
```

Lua 函数原型

```
poseInverse(pose: table) -> table
```

Lua 示例

```
pose_inverse = poseInverse({0.2, 0.5, 0.1, 1.57, 0, 3.14})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"Math.poseInverse","params":[[0.2, 0.5, 0.1, 1.57, 0, 3.14]],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.19920341988726448,-0.09960155178838484,-0.5003973704832628,1.5699999989900404,-0.0015926530848129354,-3.1415913853161266]}
```

poseRotation()

```
std::vector< double > arcs::common_interface::Math::poseRotation (
    const std::vector< double > & pose,
    const std::vector< double > & rotv)
```

姿态旋转

参数

<i>pose</i>	
<i>rotv</i>	

返回

Python 函数原型

```
poseRotation(self: pyaubo_sdk.Math, arg0: List[float], arg1: List[float]) -> List[float]
```

Lua 函数原型

```
poseRotation(pose: table, rotv: table) -> table
```

poseSub()

```
std::vector< double > arcs::common_interface::Math::poseSub (
    const std::vector< double > & p1,
    const std::vector< double > & p2)
```

位姿相减

两个参数都包含三个位置参数 (x、y、z)，统称为P，以及三个旋转参数 (R_x、R_y、R_z)，统称为R。此函数根据以下方式计算结果 p_3，即给定位姿的相加： $p\_3.P = p\_1.P - p\_2.P$ ， $p\_3.R = p\_1.R * p\_2.R.inverse$

参数

<i>p1</i>	工具位姿 1
<i>p2</i>	工具位姿 2

返回

位姿相减计算结果

Python 函数原型

```
poseSub(self: pyaubo_sdk.Math, arg0: List[float], arg1: List[float]) -> List[float]
```

Lua 函数原型

```
poseSub(p1: table, p2: table) -> table
```

Lua 示例

```
pose_sub = poseSub({0.2, 0.5, 0.1, 1.57, 0, 0},{0.2, 0.5, 0.6, 1.57, 0, 0})
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "Math.poseSub", "params": [[0.2, 0.5, 0.1, 1.57, 0, 0], [0.2, 0.5, 0.6, 1.57, 0, 0]], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, -0.5, 0.0, -0.0, 0.0] }
```

poseToHomMatrix()

```
std::vector< double > arcs::common_interface::Math::poseToHomMatrix (
    const std::vector< double > & pose)
poseToHomMatrix: 位姿变换得到齐次变换矩阵
```

参数

<i>pose</i>	输入的位姿

返回

输出的齐次变换矩阵, 元素横向排列

poseTrans()

```
std::vector< double > arcs::common_interface::Math::poseTrans (
    const std::vector< double > & pose_from,
    const std::vector< double > & pose_from_to)
```

位姿变换

第一个参数 *p_from* 用于转换第二个参数 *p_from_to*, 并返回结果。这意味着结果是从 *p_from* 的坐标系开始, 然后在该坐标系中移动 *p_from_to* 后的位姿。

这个函数可以从两个不同的角度来看。一种是函数将 *p_from_to* 根据 *p_from* 的参数进行转换, 即平移和旋转。另一种是函数被用于获取结果姿态, 先对 *p_from* 进行移动, 然后再对 *p_from_to* 进行移动。如果将姿态视为转换矩阵, 它看起来像是:

$$T_{world \rightarrow to} = T_{world \rightarrow from} * T_{from \rightarrow to}, \quad T_{x \rightarrow to} = T_{x \rightarrow from} * T_{from \rightarrow to}$$

这两个方程描述了姿态变换的基本原理, 根据给定的起始姿态和相对于起始姿态的姿态变化, 可以计算出目标姿态。

举个例子, 已知B 相对于A 的位姿、C 相对于B 的位姿, 求C 相对于A 的位姿。第一个参数是B 相对于A 的位姿, 第二个参数是C 相对于B 的位姿, 返回值是C 相对于A 的位姿。

参数

<i>pose_from</i>	起始位姿 (空间向量)
<i>pose_from</i> ↔ <i>_to</i>	相对于起始位姿的姿态变化 (空间向量)

返回

结果位姿 (空间向量)

Python 函数原型

```
poseTrans(self: pyaubo_sdk.Math, arg0: List[float], arg1: List[float]) -> List[float]
```

Lua 函数原型

```
poseTrans(pose_from: table, pose_from_to: table) -> table
```

Lua 示例

```
pose_trans = poseTrans({0.2, 0.5, 0.1, 1.57, 0, 0},{0.2, 0.5, 0.6, 1.57, 0, 0})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"Math.poseTrans","params":[[0.2, 0.5, 0.1, 1.57, 0, 0],[0.2, 0.5, 0.6, 1.57, 0, 0]],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.4,-0.09960164640373415,0.6004776374923573,3.14,-0.0,0.0]}
```

poseTransInv()

```
std::vector< double > arcs::common_interface::Math::poseTransInv (
    const std::vector< double > & pose_from,
    const std::vector< double > & pose_to_from)
```

姿态逆变换

已知C 相对于A 的位姿、C 相对于B 的位姿，求B 相对于A 的位姿。第一个参数是C 相对于A 的位姿，第二个参数是C 相对于B 的位姿，返回值是B 相对于A 的位姿。

参数

<i>pose_from</i>	起始位姿
<i>pose_to_from</i>	相对于结果位姿的姿态变化

返回

结果位姿

Python 函数原型

```
poseTransInv(self: pyaubo_sdk.Math, arg0: List[float], arg1: List[float]) -> List[float]
```

Lua 函数原型

```
poseTransInv(pose_from: table, pose_to_from: table) -> table
```

Lua 示例

```
pose_trans_inv = poseTransInv({0.4, -0.0996016, 0.600478, 3.14, 0, 0},{0.2, 0.5, 0.6, 1.57, 0, 0})
```


JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "Math.poseTransInv", "params": [[0.4, -0.0996016, 0.600478, 3.14, 0, 0], [0.2, 0.5, 0.6, 1.57, 0, 0]], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": [0.2, 0.5000000464037341, 0.10000036250764266, 1.57, -0.0, 0.0] }
```

quaternionToRpy()

```
std::vector< double > arcs::common_interface::Math::quaternionToRpy (
    const std::vector< double > & quat)
```

四元数转欧拉角

参数

<i>quat</i>	四元数

返回

欧拉角

Python 函数原型

```
quaternionToRpy(self: pyaubo_sdk.Math, arg0: List[float]) -> List[float]
```

Lua 函数原型

```
quaternionToRpy(quat: table) -> table
```

Lua 示例

```
rpy = quaternionToRpy({0.834722, 0.0780426, 0.451893, 0.304864})
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "Math.quaternionToRpy", "params": [[0.834722, 0.0780426, 0.451893, 0.304864]], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": [0.6110000520523781, 0.7849996877683915, 0.960000543982093] }
```

rpyToQuaternion()

```
std::vector< double > arcs::common_interface::Math::rpyToQuaternion (
    const std::vector< double > & rpy)
```

欧拉角转四元数

参数

<i>rpy</i>	欧拉角

返回

四元数

Python 函数原型

```
rpyToQuaternion(self: pyaubo_sdk.Math, arg0: List[float]) -> List[float]
```

Lua 函数原型

```
rpyToQuaternion(rpy: table) -> table
```

Lua 示例

```
quaternion = rpyToQuaternion({0.611, 0.785, 0.960})
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "Math.rpyToQuaternion", "params": [[0.611, 0.785, 0.960]], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": [0.834721517970497, 0.07804256900772265, 0.4518931575790371, 0.3048637712043723]}
```

tcpOffsetIdentify()

```
ResultWithErrno arcs::common_interface::Math::tcpOffsetIdentify (
    const std::vector< std::vector< double > > & poses)
```

四点法标定TCP 偏移

找一个尖点，将机械臂工具末端点绕着尖点示教四个位置，姿态差别要大。设置完毕后即可计算出来结果。

参数

<i>poses</i>	四个点的位姿集合

返回

TCP 标定结果和标定结果是否有效

Python 函数原型

```
tcpOffsetIdentify(self: pyaubo_sdk.Math, arg0: List[List[float]]) -> Tuple[List[float], int]
```

Lua 函数原型

```
tcpOffsetIdentify(poses: table) -> table
```

Lua 示例

```
p1 = {0.48659, 0.10456, 0.24824, -3.135, 0.004, 1.569}
p2 = {0.38610, 0.09096, 0.22432, 2.552, -0.437, 1.008}
p3 = {0.38610, 0.05349, 0.16791, -3.021, -0.981, 0.517}
p4 = {0.49675, 0.06417, 0.21408, -2.438, 0.458, 0.651}
p = {p1, p2, p3, p4}
tcp_result = tcpOffsetIdentify(p)
```

transferRefFrame()

```
std::vector< double > arcs::common_interface::Math::transferRefFrame (
    const std::vector< double > & F_b_a_old,
    const Vector3d & V_in_a,
    int type)
```

参数

<i>F_b_a_old</i>	
<i>V_in_a</i>	
<i>type</i>	

返回**Python 函数原型**

```
transferRefFrame(self: pyaubo_sdk.Math, arg0: List[float], arg1: List[float[3]], arg2: int) -> List[←
float]
```

Lua 函数原型

```
transferRefFrame(F_b_a_old: table, V_in_a: table, type: number) -> table
```

8.5 RegisterControl (寄存器操作)

通用寄存器**函数**

- bool [arcs::common_interface::RegisterControl::getBoolInput](#) (uint32_t address)
从一个输入寄存器中读取布尔值，也可以通过现场总线进行访问。注意，它使用自己的内存空间。
- int [arcs::common_interface::RegisterControl::setBoolInput](#) (uint32_t address, bool value)
- int [arcs::common_interface::RegisterControl::getInt32Input](#) (uint32_t address)
从一个输入寄存器中读取整数值，也可以通过现场总线进行访问。注意，它使用自己的内存空间。
- int [arcs::common_interface::RegisterControl::setInt32Input](#) (uint32_t address, int value)
- float [arcs::common_interface::RegisterControl::getFloatInput](#) (uint32_t address)
Reads the float from one of the input registers, which can also be accessed by a Field bus.
- int [arcs::common_interface::RegisterControl::setFloatInput](#) (uint32_t address, float value)
- double [arcs::common_interface::RegisterControl::getDoubleInput](#) (uint32_t address)
从一个输入寄存器中读取双精度浮点数，也可以通过现场总线进行访问。注意，它使用自己的内存空间。
- int [arcs::common_interface::RegisterControl::setDoubleInput](#) (uint32_t address, double value)
- bool [arcs::common_interface::RegisterControl::getBoolOutput](#) (uint32_t address)
从一个输出寄存器中读取布尔值，也可以通过现场总线进行访问。注意，它使用自己的内存空间。
- int [arcs::common_interface::RegisterControl::setBoolOutput](#) (uint32_t address, bool value)
- int [arcs::common_interface::RegisterControl::getInt32Output](#) (uint32_t address)
从一个输出寄存器中读取整数值，也可以通过现场总线进行访问。注意，它使用自己的内存空间。
- int [arcs::common_interface::RegisterControl::setInt32Output](#) (uint32_t address, int value)
- float [arcs::common_interface::RegisterControl::getFloatOutput](#) (uint32_t address)
从一个输出寄存器中读取浮点数，也可以通过现场总线进行访问。注意，它使用自己的内存空间。

- `int arcs::common_interface::RegisterControl::setFloatOutput (uint32_t address, float value)`
- `double arcs::common_interface::RegisterControl::getDoubleOutput (uint32_t address)`
从一个输出寄存器中读取双精度浮点数。
- `int arcs::common_interface::RegisterControl::setDoubleOutput (uint32_t address, double value)`
- `int16_t arcs::common_interface::RegisterControl::getInt16Register (uint32_t address)`
用于 *Modbus Slave*
- `int arcs::common_interface::RegisterControl::setInt16Register (uint32_t address, int16_t value)`
- `bool arcs::common_interface::RegisterControl::getInt16RegisterBit (uint32_t address, uint8_t bit←_offset)`
获取 *Int16* 寄存器的某个 *bit* 的状态
- `int arcs::common_interface::RegisterControl::setInt16RegisterBit (uint32_t address, uint8_t bit←_offset, bool value)`
设置 *Int16* 寄存器的某个 *bit* 的状态
- `bool arcs::common_interface::RegisterControl::hasNamedVariable (const std::string &key)`
具名变量是否存在
- `std::string arcs::common_interface::RegisterControl::getNamedVariableType (const std::string &key)`
获取具名变量的类型
- `bool arcs::common_interface::RegisterControl::variableUpdated (const std::string &key, uint64_t since)`
具名变量是否更新
- `bool arcs::common_interface::RegisterControl::getBool (const std::string &key, bool default_value)`
获取变量值
- `int arcs::common_interface::RegisterControl::setBool (const std::string &key, bool value)`
设置/更新变量值
- `std::vector< char > arcs::common_interface::RegisterControl::getVecChar (const std::string &key, const std::vector< char > &default_value)`
获取变量值
- `int arcs::common_interface::RegisterControl::setVecChar (const std::string &key, const std::vector< char > &value)`
设置/更新变量值
- `int arcs::common_interface::RegisterControl::getInt32 (const std::string &key, int default_value)`
获取变量值
- `int arcs::common_interface::RegisterControl::setInt32 (const std::string &key, int value)`
设置/更新变量值
- `std::vector< int32_t > arcs::common_interface::RegisterControl::getVecInt32 (const std::string &key, const std::vector< int32_t > &default_value)`
获取变量值
- `int arcs::common_interface::RegisterControl::setVecInt32 (const std::string &key, const std::vector< int32_t > &value)`
设置/更新变量值
- `float arcs::common_interface::RegisterControl::getFloat (const std::string &key, float default_←value)`
获取变量值
- `int arcs::common_interface::RegisterControl::setFloat (const std::string &key, float value)`
设置/更新变量值
- `std::vector< float > arcs::common_interface::RegisterControl::getVecFloat (const std::string &key, const std::vector< float > &default_value)`
获取变量值
- `int arcs::common_interface::RegisterControl::setVecFloat (const std::string &key, const std::vector< float > &value)`
设置/更新变量值

- double `arcs::common_interface::RegisterControl::getDouble` (const std::string &key, double default_value)

获取变量值
- int `arcs::common_interface::RegisterControl::setDouble` (const std::string &key, double value)

设置/更新变量值
- std::vector< double > `arcs::common_interface::RegisterControl::getVecDouble` (const std::string &key, const std::vector< double > &default_value)

获取变量值
- int `arcs::common_interface::RegisterControl::setVecDouble` (const std::string &key, const std::vector< double > &value)

设置/更新变量值
- std::string `arcs::common_interface::RegisterControl::getString` (const std::string &key, const std::string &default_value)

获取变量值
- int `arcs::common_interface::RegisterControl::setString` (const std::string &key, const std::string &value)

设置/更新变量值
- int `arcs::common_interface::RegisterControl::clearNamedVariable` (const std::string &key)

清除变量
- int `arcs::common_interface::RegisterControl::setWatchDog` (const std::string &key, double timeout, int action)

设置看门狗
- int `arcs::common_interface::RegisterControl::getWatchDogAction` (const std::string &key)

获取看门狗动作
- int `arcs::common_interface::RegisterControl::getWatchDogTimeout` (const std::string &key)

获取看门狗超时时间
- int `arcs::common_interface::RegisterControl::modbusAddSignal` (const std::string &device_info, int slave_number, int signal_address, int signal_type, const std::string &signal_name, bool sequential_mode)

添加一个新的 *Modbus* 信号以供控制器监视。不需要返回响应。
- int `arcs::common_interface::RegisterControl::addFixedModbusGroup` (const std::string &signal_name, int count)

为指定 *Modbus* 寄存器信号创建固定读取分组，优化连续地址的批量读取效率
- int `arcs::common_interface::RegisterControl::modbusDeleteSignal` (const std::string &signal_name)

删除指定名称的信号。
- int `arcs::common_interface::RegisterControl::modbusDeleteAllSignals` ()

删除所有 *modbus* 信号
- int `arcs::common_interface::RegisterControl::modbusGetSignalStatus` (const std::string &signal_name)

读取特定信号的当前值。
- std::vector< std::string > `arcs::common_interface::RegisterControl::modbusGetSignalNames` ()

获取所有信号的名字集合
- std::vector< int > `arcs::common_interface::RegisterControl::modbusGetSignalTypes` ()

获取所有信号的类型集合
- std::vector< int > `arcs::common_interface::RegisterControl::modbusGetSignalValues` ()

获取所有信号的数值集合
- std::vector< int > `arcs::common_interface::RegisterControl::modbusGetSignalErrors` ()

获取所有信号的请求是否有错误 (0:无错误, 其他:有错误) 集合
- int `arcs::common_interface::RegisterControl::modbusSendCustomCommand` (const std::string &device_info, int slave_number, int function_code, const std::vector< uint8_t > &data)

将用户指定的命令发送到指定 *IP* 地址上的 *Modbus* 单元。由于不会接收到响应，因此不能用于请求数据。用户负责提供对所提供的功能码有意义的数据。内置函数负责构建 *Modbus* 帧，因此用户不需要关心命令的长度。

- `int arcs::common_interface::RegisterControl::modbusSetDigitalInputAction` (const std::string &robot_name, const std::string &signal_name, `StandardInputAction` action)
将选择的数字输入信号设置为 “default” 或 “freedrive”
- `int arcs::common_interface::RegisterControl::modbusSetOutputRunstate` (const std::string &robot_name, const std::string &signal_name, `StandardOutputRunState` runstate)
设置 *Modbus* 信号输出动作
- `int arcs::common_interface::RegisterControl::modbusSetOutputRunstate1` (const std::string &robot_name, const std::string &signal_name, `StandardOutputRunState` runstate, const std::string &object_name, double threshold)
设置 *Modbus* 信号输出动作, 并支持基于路点/平面/障碍物的条件阈值判断。
- `int arcs::common_interface::RegisterControl::modbusSetOutputSignal` (const std::string &signal_name, uint16_t value)
将指定名称的输出寄存器信号设置为给定的值
- `int arcs::common_interface::RegisterControl::modbusSetOutputSignal1` (const std::string &signal_name, const std::vector< uint16_t > &values)
将从指定名称开始的若干个连续输出寄存器信号设置为给定的值
- `int arcs::common_interface::RegisterControl::modbusSetOutputSignalPulse` (const std::string &signal_name, uint16_t value, double duration)
设置 *modbus* 信号输出脉冲 (仅支持线圈输出类型)
- `int arcs::common_interface::RegisterControl::modbusSetSignalUpdateFrequency` (const std::string &signal_name, int update_frequency)
设置机器人向 *Modbus* 控制器发送请求的频率, 用于读取或写入信号值
- `int arcs::common_interface::RegisterControl::modbusGetSignalIndex` (const std::string &signal_name)
获取指定 *modbus* 信号索引, 从 0 开始, 不能存在则返回 -1
- `int arcs::common_interface::RegisterControl::modbusGetSignalError` (const std::string &signal_name)
获取指定 *modbus* 信号的错误状态
- `int arcs::common_interface::RegisterControl::getModbusDeviceStatus` (const std::string &device_name)
获取指定 *modbus* 设备的连接状态
- `int arcs::common_interface::RegisterControl::addModbusEncoder` (int encoder_id, int range_id, const std::string &signal_name)
将某个 *modbus* 寄存器信号作为编码器
- `int arcs::common_interface::RegisterControl::addInt32RegEncoder` (int encoder_id, int range_id, const std::string &key)
添加 *Int32* 寄存器的虚拟编码器
- `int arcs::common_interface::RegisterControl::deleteVirtualEncoder` (int encoder_id)
删除虚拟编码器

8.5.1 详细描述

通用寄存器

8.5.2 函数说明

addFixedModbusGroup()

```
int arcs::common_interface::RegisterControl::addFixedModbusGroup (
    const std::string & signal_name,
    int count)
```

为指定 *Modbus* 寄存器信号创建固定读取分组, 优化连续地址的批量读取效率

功能: 根据传入的起始信号名定位到对应的 16 位 *Modbus* 寄存器地址, 并基于该起始地址和连续数量创建 “不可扩展的固定分组”, 该分组会被独立管理, 不会与其他普通分组合并或扩展, 适用于需要稳定、原子化读取的连续寄存器地址段; 固定分组创建后, 对该段地址的读取操作会严格按照分组范围执行, 避免因动态分组扩展导致的读取范围变化。

参数

<i>signal_name</i>	起始信号名（作为固定分组的基准信号，需为已通过 modbusAddSignal 创建的 16 位Modbus 寄存器信号，支持保持寄存器/输入寄存器类型，不支持数字量信号）
<i>count</i>	固定分组包含的连续信号（寄存器）数量（大于等于 1）： • 值为 1：仅包含起始信号对应的单个寄存器地址（取消包含该信号的固定分组）； • 值 >1：从起始信号地址开始的连续 count 个寄存器地址，需确保地址连续且不超出设备支持范围；

返回

成功返回 0，失败返回错误码。

注解

1. 该接口仅针对 16 位Modbus 寄存器信号（signal_type 为 2/3）生效；
2. 单个固定分组的最大连续数量受设备限制（默认最大 30 个）；
3. 重复为同一信号创建固定分组会覆盖原有配置（以最后一次的 count 值为准）。

Python 接口原型

```
addFixedModbusGroup(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: int) -> int
```

Lua 接口原型

```
addFixedModbusGroup(signal_name: string, count: number) -> nil
```

Lua 示例

```
– 为起始信号”Modbus_0” 创建包含 10 个连续寄存器的固定分组 addFixedModbusGroup(“↵  
Modbus_0”, 10)
```

JSON-RPC 请求示例

```
{”jsonrpc”:”2.0”,”method”:”RegisterControl.addFixedModbusGroup”,”params”:[”Modbus_0”,10]↵  
,”id”:1}
```

JSON-RPC 响应示例

```
{”id”:1,”jsonrpc”:”2.0”,”result”:0}
```

addInt32RegEncoder()

```
int arcs::common_interface::RegisterControl::addInt32RegEncoder (  
    int encoder_id,  
    int range_id,  
    const std::string & key)
```

添加Int32 寄存器的虚拟编码器

参数

<i>encoder_id</i>	编码器ID
<i>range_id</i>	范围ID
<i>key</i>	变量名

返回

addModbusEncoder()

```
int arcs::common_interface::RegisterControl::addModbusEncoder (  
    int encoder_id,  
    int range_id,  
    const std::string & signal_name)
```

将某个 modbus 寄存器信号作为编码器

参数

<i>encoder_id</i>	不能为 0
<i>signal_name</i>	modbus 信号名字, 必须为寄存器类型

返回

clearNamedVariable()

```
int arcs::common_interface::RegisterControl::clearNamedVariable (  
    const std::string & key)
```

清除变量

参数

<i>key</i>	

返回

Python 函数原型

```
clearNamedVariable(self: pyaubo_sdk.RegisterControl, arg0: str) -> int
```

Lua 函数原型

```
clearNamedVariable(key: string) -> nil
```


Lua 示例

```
clearNamedVariable("custom")
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.clearNamedVariable","params":["custom"],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":1}
```

deleteVirtualEncoder()

```
int arcs::common_interface::RegisterControl::deleteVirtualEncoder (
```

```
    int encoder_id)
```

删除虚拟编码器

参数

<i>encoder</i> ↔	
<i>_id</i>	

返回

getBool()

```
bool arcs::common_interface::RegisterControl::getBool (
```

```
    const std::string & key,
```

```
    bool default_value)
```

获取变量值

参数

<i>key</i>	
<i>default_value</i>	

返回

Python 函数原型

```
getBool(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: bool) -> bool
```

Lua 函数原型

```
getBool(key: string, default_value: boolean) -> boolean
```

Lua 示例

```
Bool_var = getBool("custom",false)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.getBool","params":["custom",false],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":true}
```

getBoolInput()

```
bool arcs::common_interface::RegisterControl::getBoolInput (
    uint32_t address)
```

从一个输入寄存器中读取布尔值，也可以通过现场总线进行访问。注意，它使用自己的内存空间。

参数

<i>address</i>	寄存器的地址 (0:127)

返回

寄存器中保存的布尔值 (true、false)

注解

布尔输入寄存器的较低范围[0:63]保留供FieldBus/PLC 接口使用。较高范围[64:127]无法通过FieldBus/PLC 接口访问，因为它保留供外部RTDE 客户端使用。

Python 函数原型

```
getBoolInput(self: pyaubo_sdk.RegisterControl, arg0: int) -> bool
```

Lua 函数原型

```
getBoolInput(address: number) -> boolean
```

Lua 示例

```
BoolInput_0 = getBoolInput(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.getBoolInput","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":false}
```

getBoolOutput()

```
bool arcs::common_interface::RegisterControl::getBoolOutput (
    uint32_t address)
```

从一个输出寄存器中读取布尔值，也可以通过现场总线进行访问。注意，它使用自己的内存空间。

参数

<i>address</i>	寄存器地址 (0:127)

返回

寄存器中保存的布尔值 (true, false)

注解

布尔输出寄存器的较低范围[0:63]保留供现场总线/PLC 接口使用。较高范围[64:127]不能通过现场总线/PLC 接口访问, 因为它们是为外部RTDE 客户端保留的。

Python 函数原型

```
getBoolOutput(self: pyaubo_sdk.RegisterControl, arg0: int) -> bool
```

Lua 函数原型

```
getBoolOutput(address: number) -> boolean
```

Lua 示例

```
BoolOutput_0 = getBoolOutput(0)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RegisterControl.getBoolOutput", "params": [0], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": false}
```

getDouble()

```
double arcs::common_interface::RegisterControl::getDouble (  
    const std::string & key,  
    double default_value)
```

获取变量值

参数

<i>key</i>	
<i>default_value</i>	

返回**Python 函数原型**

```
getDouble(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: float) -> float
```

Lua 函数原型

```
getDouble(key: string, default_value: number) -> number
```

Lua 示例

```
var_Double = getDouble("custom",0.0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.getDouble","params":["custom",0.0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0.0}
```

getDoubleInput()

```
double arcs::common_interface::RegisterControl::getDoubleInput (
    uint32_t address)
```

从一个输入寄存器中读取双精度浮点数，也可以通过现场总线进行访问。注意，它使用自己的内存空间。

参数

<i>address</i>	寄存器的地址 (0:47)

返回

寄存器中保存的双精度浮点数值

Python 函数原型

```
getDoubleInput(self: pyaubo_sdk.RegisterControl, arg0: int) -> float
```

Lua 函数原型

```
getDoubleInput(address: number) -> number
```

Lua 示例

```
DoubleInput_0 = getDoubleInput(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.getDoubleInput","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0.0}
```

getDoubleOutput()

```
double arcs::common_interface::RegisterControl::getDoubleOutput (
    uint32_t address)
```

从一个输出寄存器中读取双精度浮点数。

参数

<i>address</i>	寄存器地址

返回

寄存器中保存的双精度浮点数值

Python 函数原型

```
getDoubleOutput(self: pyaubo_sdk.RegisterControl, arg0: int) -> float
```

Lua 函数原型

```
getDoubleOutput(address: number) -> number
```

Lua 示例

```
DoubleOutput_0 = getDoubleOutput(0)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RegisterControl.getDoubleOutput", "params": [0], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0.0}
```

getFloat()

```
float arcs::common_interface::RegisterControl::getFloat (  
    const std::string & key,  
    float default_value)
```

获取变量值

参数

<i>key</i>	
<i>default_value</i>	

返回**Python 函数原型**

```
getFloat(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: float) -> float
```

Lua 函数原型

```
getFloat(key: string, default_value: number) -> number
```

Lua 示例

```
var_Float = getFloat("custom",0.0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.getFloat","params":["custom",0.0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":4.400000095367432}
```

getFloatInput()

```
float arcs::common_interface::RegisterControl::getFloatInput (
    uint32_t address)
```

Reads the float from one of the input registers, which can also be accessed by a Field bus.

Note, uses it' s own memory space.

从一个输入寄存器中读取浮点数，也可以通过现场总线进行访问。注意，它使用自己的内存空间。

参数

<i>address</i>	Address of the register (0:47) 寄存器地址 (0:47)

返回

The value held by the register (float) 寄存器中保存的浮点数值

注解

The lower range of the float input registers [0:23] is reserved for FieldBus/PLC interface usage. The upper range [24:47] cannot be accessed by FieldBus/PLC interfaces, since it is reserved for external RTDE clients. 浮点数输入寄存器的较低范围[0:23]保留供现场总线/PLC 接口使用。较高范围[24:47]不能通过现场总线/PLC 接口访问，因为它们是为外部RTDE 客户端保留的。

Python 函数原型

```
getFloatInput(self: pyaubo_sdk.RegisterControl, arg0: int) -> float
```

Lua 函数原型

```
getFloatInput(address: number) -> number
```

Lua 示例

```
FloatInput_0 = getFloatInput(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.getFloatInput","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0.0}
```

getFloatOutput()

```
float arcs::common_interface::RegisterControl::getFloatOutput (
    uint32_t address)
```

参数

<i>address</i>	寄存器地址 (0:47)

返回

寄存器中保存的浮点数值 (float)

注解

浮点数输出寄存器的较低范围[0:23]保留供现场总线/PLC 接口使用。较高范围[24:47]不能通过现场总线/PLC 接口访问, 因为它们是为外部RTDE 客户端保留的。

Python 函数原型

```
getFloatOutput(self: pyaubo_sdk.RegisterControl, arg0: int) -> float
```

Lua 函数原型

```
getFloatOutput(address: number) -> number
```

Lua 示例

```
FloatOutput_0 = getFloatOutput(0)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RegisterControl.getFloatOutput", "params": [0], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 3.3}
```

getInt16Register()

```
int16_t arcs::common_interface::RegisterControl::getInt16Register (  
    uint32_t address)
```

用于 Modbus Slave

参数

<i>address</i>	

返回**Python 函数原型**

```
getInt16Register(self: pyaubo_sdk.RegisterControl, arg0: int) -> int
```

Lua 函数原型

```
getInt16Register(address: number) -> number
```

Lua 示例

```
Int16Register_0 = getInt16Register(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.getInt16Register","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getInt16RegisterBit()

```
bool arcs::common_interface::RegisterControl::getInt16RegisterBit (
    uint32_t address,
    uint8_t bit_offset)
```

获取 Int16 寄存器的某个 bit 的状态

参数

<i>address</i>	Int16 寄存器地址
<i>bit_offset</i>	寄存器中的 bit 偏移, 0 ~ 15

返回

指定 bit 的状态, true 表示 1, false 表示 0 或参数无效

Python 函数原型

```
getInt16RegisterBit(self: pyaubo_sdk.RegisterControl, address: int, bit_offset: int) -> bool
```

Lua 函数原型

```
getInt16RegisterBit(address: number, bit_offset: number) -> boolean
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.getInt16RegisterBit","params":[1, 5],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":true}
```

getInt32()

```
int arcs::common_interface::RegisterControl::getInt32 (
    const std::string & key,
    int default_value)
```

获取变量值

参数

<i>key</i>	
<i>default_value</i>	

返回**Python 函数原型**

```
getInt32(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: int) -> int
```

Lua 函数原型

```
getInt32(key: string, default_value: number) -> number
```

Lua 示例

```
Int32 = getInt32("custom",0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.getInt32","params":["custom",0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":6}
```

getInt32Input()

```
int arcs::common_interface::RegisterControl::getInt32Input (  
    uint32_t address)
```

从一个输入寄存器中读取整数值，也可以通过现场总线进行访问。注意，它使用自己的内存空间。

参数

<i>address</i>	寄存器的地址 (0:47)

返回

寄存器中保存的整数值[-2,147,483,648 : 2,147,483,647]

注解

整数输入寄存器的较低范围[0:23]保留供FieldBus/PLC 接口使用。较高范围[24:47]无法通过FieldBus/PLC 接口访问，因为它保留供外部RTDE 客户端使用。

Python 函数原型

```
getInt32Input(self: pyaubo_sdk.RegisterControl, arg0: int) -> int
```

Lua 函数原型

```
getInt32Input(address: number) -> number
```

Lua 示例

```
Int32Input_0 = getInt32Input(0)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RegisterControl.getInt32Input", "params": [0], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

getInt32Output()

```
int arcs::common_interface::RegisterControl::getInt32Output (
    uint32_t address)
```

从一个输出寄存器中读取整数值，也可以通过现场总线进行访问。注意，它使用自己的内存空间。

参数

<i>address</i>	寄存器地址 (0:47)

返回

寄存器中保存的整数值 (-2,147,483,648 : 2,147,483,647)

注解

整数输出寄存器的较低范围[0:23]保留供现场总线/PLC 接口使用。较高范围[24:47]不能通过现场总线/PLC 接口访问，因为它们是为外部RTDE 客户端保留的。

Python 函数原型

```
getInt32Output(self: pyaubo_sdk.RegisterControl, arg0: int) -> int
```

Lua 函数原型

```
getInt32Output(address: number) -> number
```

Lua 示例

```
Int32Output_0 = getInt32Output(0)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RegisterControl.getInt32Output", "params": [0], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

getModbusDeviceStatus()

```
int arcs::common_interface::RegisterControl::getModbusDeviceStatus (
    const std::string & device_name)
```

参数

<i>device_name</i>	设备名是TCP 格式, "ip:port", 例如: "127.0.0.1:502" 设备名是RTU 格式, "serial_port", 例如: "/dev/ttyUSB0"

返回

0: 表示设备处于连接状态 -1: 表示设备不存在 -2: 表示设备处于断开状态

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.getModbusDeviceStatus","params":["172.16.26.248:502"],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getNamedVariableType()

```
std::string arcs::common_interface::RegisterControl::getNamedVariableType (  
    const std::string & key)
```

获取具名变量的类型

参数

<i>key</i>	

返回

Lua 函数原型

```
getNamedVariableType(key: string) -> string
```

Lua 示例

```
NamedVariableType = getNamedVariableType("custom")
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.getNamedVariableType","params":["custom"],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":"NONE"}
```

getString()

```
std::string arcs::common_interface::RegisterControl::getString (  
    const std::string & key,  
    const std::string & default_value)
```

获取变量值

参数

<i>key</i>	
<i>default_value</i>	

返回

Python 函数原型

```
getString(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: str) -> str
```

Lua 函数原型

```
getString(key: string, default_value: string) -> string
```

Lua 示例

```
var_String = getString("custom", "")
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RegisterControl.getString", "params": ["custom", ""], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": "test"}
```

getVecChar()

```
std::vector< char > arcs::common_interface::RegisterControl::getVecChar (  
    const std::string & key,  
    const std::vector< char > & default_value)
```

获取变量值

参数

<i>key</i>	
<i>default_value</i>	

返回

Python 函数原型

```
getVecChar(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: List[str]) -> List[str]
```

Lua 函数原型

```
getVecChar(key: string, default_value: table) -> table
```

Lua 示例

```
VecChar = getVecChar("custom",{})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.getVecChar","params":["custom",[]],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0,1,0]}
```

getVecDouble()

```
std::vector< double > arcs::common_interface::RegisterControl::getVecDouble (
    const std::string & key,
    const std::vector< double > & default_value)
```

获取变量值

参数

<i>key</i>	
<i>default_value</i>	

返回

Python 函数原型

```
getVecDouble(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: List[float]) -> List[float]
```

Lua 函数原型

```
getVecDouble(key: string, default_value: table) -> table
```

Lua 示例

```
VecDouble = getVecDouble("custom",{})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.getVecDouble","params":["custom",[]],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.1,0.2,0.3]}
```

getVecFloat()

```
std::vector< float > arcs::common_interface::RegisterControl::getVecFloat (
    const std::string & key,
    const std::vector< float > & default_value)
```

获取变量值

参数

<i>key</i>	
<i>default_value</i>	

返回

Python 函数原型

```
getVecFloat(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: List[float]) -> List[float]
```

Lua 函数原型

```
getVecFloat(key: string, default_value: table) -> table
```

Lua 示例

```
VecFloat = getVecFloat("custom",{})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.getVecFloat","params":["custom",[]],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.0,0.10000000149011612,3.299999952316284]}
```

getVecInt32()

```
std::vector< int32_t > arcs::common_interface::RegisterControl::getVecInt32 (  
    const std::string & key,  
    const std::vector< int32_t > & default_value)
```

获取变量值

参数

<i>key</i>	
<i>default_value</i>	

返回

Python 函数原型

```
getVecInt32(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: List[int]) -> List[int]
```

Lua 函数原型

```
getVecInt32(key: string, default_value: table) -> table
```

Lua 示例

```
VecInt32 = getVecInt32("custom",{})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.getVecInt32","params":["custom",[]],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[1,2,3,4]}
```

getWatchDogAction()

```
int arcs::common_interface::RegisterControl::getWatchDogAction (  
    const std::string & key)
```

获取看门狗动作

参数

<i>key</i>	

返回

getWatchDogTimeout()

```
int arcs::common_interface::RegisterControl::getWatchDogTimeout (  
    const std::string & key)
```

获取看门狗超时时间

参数

<i>key</i>	

返回

hasNamedVariable()

```
bool arcs::common_interface::RegisterControl::hasNamedVariable (  
    const std::string & key)
```

具名变量是否存在

参数

<i>key</i>	变量名

返回

Lua 函数原型

```
hasNamedVariable(key: string) -> boolean
```

Lua 示例

```
NamedVariable = hasNamedVariable("custom")
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RegisterControl.hasNamedVariable", "params": ["custom"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": false}
```

modbusAddSignal()

```
int arcs::common_interface::RegisterControl::modbusAddSignal (
    const std::string & device_info,
    int slave_number,
    int signal_address,
    int signal_type,
    const std::string & signal_name,
    bool sequential_mode)
```

添加一个新的Modbus 信号以供控制器监视。不需要返回响应。

参数

<i>device_info</i>	设备信息设备信息是RTU 格式，例如： "serial_port,baud,parity,data_bit,stop_bit" (1)serial_port 参数指定串口的名称，例如， 在Linux 上为"/dev/ttyS0" 或"/dev/ttyUSB0"， 在Windows 上为"\\.\\COM10" (2)baud 参数指定通信的波特率， 例如 9600、19200、57600、115200 等 (3)parity 参数指定奇偶校验方式， N 表示无校验， E 表示偶校验， O 表示奇校验 (4)data_bit 参数指定数据位数， 允许的值为 5、6、7 和 8 (5)stop_bit 参数指定停止位数， 允许的值为 1 和 2

设备信息是TCP 格式，例如： "ip address,port" (1)ip address 参数指定服务器的IP 地址 (2)port 参数指定服务器监听的端口号

参数

<i>slave_number</i>	通常不使用， 设置为 255 即可， 但可以在 0 到 255 之间自由选择
<i>signal_address</i>	指定新信号应该反映的线圈或寄存器的地址。请参考Modbus 单元的配置以获取此信息。
<i>signal_type</i>	指定要添加的信号类型。0 = 数字输入， 1 = 数字输出， 2 = 寄存器输入， 3 = 寄存器输出。
<i>signal_name</i>	唯一标识信号的名词。如果提供的字符串与已添加的信号相等， 则新信号将替换旧信号。字符串的长度不能超过 20 个字符。

<i>sequential_mode</i>	设置为True 会强制Modbus 客户端在发送下一个请求之前等待响应。某些 fieldbus 单元需要此模式。可选参数。

返回

Python 函数原型

```
modbusAddSignal(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: int, arg2: int, arg3: int,
arg4: str, arg5: bool) -> int
```

Lua 函数原型

```
modbusAddSignal(device_info: string, slave_number: number, signal_address: number, signal_
_type: number, signal_name: string, sequential_mode: boolean) -> nil
```

Lua 示例

```
modbusAddSignal("/dev/ttyRobotTool,115200,N,8,1", 1, signal_address: number, 264, "Modbus_
_0", false)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RegisterControl.modbusAddSignal", "params": ["/dev/ttyRobotTool,115200,
N,8,1", 1, 264, 3, "Modbus_0", false], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

modbusDeleteAllSignals()

```
int arcs::common_interface::RegisterControl::modbusDeleteAllSignals ()
删除所有 modbus 信号
```

返回

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RegisterControl.modbusDeleteAllSignals", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

modbusDeleteSignal()

```
int arcs::common_interface::RegisterControl::modbusDeleteSignal (
    const std::string & signal_name)
删除指定名称的信号。
```

参数

<i>signal_name</i>	要删除的信号的名称

返回

Python 函数原型

```
modbusDeleteSignal(self: pyaubo_sdk.RegisterControl, arg0: str) -> int
```

Lua 函数原型

```
modbusDeleteSignal(signal_name: string) -> nil
```

Lua 示例

```
modbusDeleteSignal("Modbus_1")
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RegisterControl.modbusDeleteSignal", "params": ["Modbus_1"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

modbusGetSignalError()

```
int arcs::common_interface::RegisterControl::modbusGetSignalError (  
    const std::string & signal_name)
```

获取指定 modbus 信号的错误状态

参数

<i>signal_name</i>	

返回

返回错误代码 [ModbusErrorNum](#)

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RegisterControl.modbusGetSignalError", "params": ["Modbus_0"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 6}
```

modbusGetSignalErrors()

```
std::vector< int > arcs::common_interface::RegisterControl::modbusGetSignalErrors ()
```

获取所有信号的请求是否有错误 (0:无错误, 其他:有错误) 集合

返回

[ModbusErrorNum](#)

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.modbusGetSignalErrors","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[6,6,6,6]}
```

modbusGetSignalIndex()

```
int arcs::common_interface::RegisterControl::modbusGetSignalIndex (
```

```
    const std::string & signal_name)
```

获取指定 modbus 信号索引, 从 0 开始, 不能存在则返回-1

参数

<i>signal_name</i>	

返回

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.modbusGetSignalIndex","params":["Modbus_0"],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

modbusGetSignalNames()

```
std::vector< std::string > arcs::common_interface::RegisterControl::modbusGetSignalNames ()
```

获取所有信号的名字集合

返回

所有信号的名字集合

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.modbusGetSignalNames","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":["Modbus_0"]}
```

modbusGetSignalStatus()

```
int arcs::common_interface::RegisterControl::modbusGetSignalStatus (
```

```
    const std::string & signal_name)
```

读取特定信号的当前值。

参数

<i>signal_name</i>	要获取值的信号的名称

返回

对于数字信号：1 或 0。对于寄存器信号：表示为整数的寄存器值。如果值为-1，则表示该信号不存在。

Python 函数原型

```
modbusGetSignalStatus(self: pyaubo_sdk.RegisterControl, arg0: str) -> int
```

Lua 函数原型

```
modbusGetSignalStatus(signal_name: string) -> number
```

Lua 示例

```
var0 = modbusGetSignalStatus("Modbus_0")
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.modbusGetSignalStatus","params":["Modbus_0"],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":1}
```

modbusGetSignalTypes()

```
std::vector< int > arcs::common_interface::RegisterControl::modbusGetSignalTypes ()
```

获取所有信号的类型集合

返回

所有信号的类型集合

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.modbusGetSignalTypes","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[1,0,2,3]}
```

modbusGetSignalValues()

```
std::vector< int > arcs::common_interface::RegisterControl::modbusGetSignalValues ()
```

获取所有信号的数值集合

返回

所有信号的数值集合

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.modbusGetSignalValues","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[1,1,88,33]}
```

modbusSendCustomCommand()

```
int arcs::common_interface::RegisterControl::modbusSendCustomCommand (
    const std::string & device_info,
    int slave_number,
    int function_code,
    const std::vector< uint8_t > & data)
```

将用户指定的命令发送到指定IP 地址上的Modbus 单元。由于不会接收到响应，因此不能用于请求数据。用户负责提供对所提供的功能码有意义的数据。内置函数负责构建Modbus 帧，因此用户不需要关心命令的长度。

参数

<i>device_info</i>	设备信息设备信息是RTU 格式，例如： "serial_port,baud,parity,data_bit,stop_bit" (1)serial_port 参数指定串口的名称，例如，在Linux 上为"/dev/ttyS0" 或"/dev/ttyUSB0"，在Windows 上为"\\.\\COM10" (2)baud 参数指定通信的波特率， 例如 9600、19200、57600、115200 等 (3)parity 参数指定奇偶校验方式，N 表示无校验， E 表示偶校验，O 表示奇校验 (4)data_bit 参数指定数据位数，允许的值为 5、6、7 和 8 (5)stop_bit 参数指定停止位数，允许的值为 1 和 2

设备信息是TCP 格式，例如： "ip address,port" (1)ip address 参数指定服务器的IP 地址 (2)port 参数指定服务器监听的端口号

参数

<i>slave_number</i>	指定用于自定义命令的从站号
<i>function_code</i>	指定自定义命令的功能码

Modbus 功能码 MODBUS_FC_READ_COILS 0x01 MODBUS_FC_READ_DISCRETE_INPUTS
0x02 MODBUS_FC_READ_HOLDING_REGISTERS 0x03 MODBUS_FC_READ_INPUT_REGISTERS
0x04 MODBUS_FC_WRITE_SINGLE_COIL 0x05 MODBUS_FC_WRITE_SINGLE_REGISTER
0x06 MODBUS_FC_READ_EXCEPTION_STATUS 0x07 MODBUS_FC_WRITE_MULTIPLE_↵
COILS 0x0F MODBUS_FC_WRITE_MULTIPLE_REGISTERS 0x10 MODBUS_FC_REPORT_↵
SLAVE_ID 0x11 MODBUS_FC_MASK_WRITE_REGISTER 0x16 MODBUS_FC_WRITE_AND_↵
_READ_REGISTERS 0x17

参数

<i>data</i>	必须是有效的字节值 (0-255)

返回

Python 函数原型

```
modbusSendCustomCommand(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: int, arg2: int,  
arg3: List[int]) -> int
```

Lua 函数原型

```
modbusSendCustomCommand(device_info: string, slave_number: number, function_code: number, data: table) -> nil
```

Lua 示例

```
modbusSendCustomCommand("/dev/ttyRobotTool,115200,N,8,1", 1, 10, {1,2,0,2,4,0,0,0}) -> nil
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RegisterControl.modbusSendCustomCommand", "params": ["/dev/ttyRobotTool,115200,N,8,1", 1, 10, [1,2,0,2,4,0,0,0]], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

modbusSetDigitalInputAction()

```
int arcs::common_interface::RegisterControl::modbusSetDigitalInputAction (
    const std::string & robot_name,
    const std::string & signal_name,
    StandardInputAction action)
```

将选择的数字输入信号设置为“default”或“freedrive”

参数

<i>robot_name</i>	连接的机器人名称
<i>signal_name</i>	先前被添加的数字输入信号
<i>action</i>	操作类型。操作可以是“default”或“freedrive”

返回

Python 函数原型

```
modbusSetDigitalInputAction(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: str, arg2: int)
```

Lua 函数原型

```
modbusSetDigitalInputAction(robot_name: string, signal_name: string, action: number) -> nil
```

Lua 示例

```
modbusSetDigitalInputAction("rob1", "Modbus_0", "Handguide")
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RegisterControl.modbusSetDigitalInputAction", "params": ["rob1", "Modbus_0", "Handguide"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

modbusSetOutputRunstate()

```
int arcs::common_interface::RegisterControl::modbusSetOutputRunstate (  
    const std::string & robot_name,  
    const std::string & signal_name,  
    StandardOutputRunState runstate)
```

设置 Modbus 信号输出动作

参数

<i>robot_name</i>	
<i>signal_name</i>	
<i>runstate</i>	

返回

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RegisterControl.modbusSetOutputRunstate", "params": ["rob1", "Modbus←  
_0", "None"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

modbusSetOutputRunstate1()

```
int arcs::common_interface::RegisterControl::modbusSetOutputRunstate1 (  
    const std::string & robot_name,  
    const std::string & signal_name,  
    StandardOutputRunState runstate,  
    const std::string & object_name,  
    double threshold)
```

设置 Modbus 信号输出动作，并支持基于路点/平面/障碍物的条件阈值判断。

详细说明：当需要根据机器人是否到达某个路点或与某个平面/障碍物的距离来触发 Modbus 信号时，可通过本接口指定条件对象及阈值。在条件满足时会将指定的 Modbus 信号设置为对应的运行状态。

参数

<i>robot_name</i>	目标机器人的名称，用于标识要监听/触发条件的机器人实例
<i>signal_name</i>	要设置的 Modbus 信号名称（需已通过 modbusAddSignal 添加）
<i>runstate</i>	输出动作状态
<i>object_name</i>	条件对象的名称（需为系统中已定义的唯一标识）

<i>threshold</i>	判断阈值： • 对于路点：表示角度阈值，单位为弧度 (rad) • 对于平面/障碍物：表示距离阈值，单位为米 (m)

返回

返回操作结果码：

- 0: 操作成功
- 非 0: 操作失败，返回具体错误码（参见系统错误码定义）

Python 接口原型

```
modbusSetOutputRunstate1(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: str, arg2: str,
arg3: str, arg4: float) -> int
```

Lua 接口原型

```
modbusSetOutputRunstate1(robot_name: string, signal_name: string, runstate: string, object↵
_name: string, threshold: number) -> nil
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RegisterControl.modbusSetOutputRunstate1", "params": ["rob1", "Modbus↵
_0", "WaypointArrived", "P1", 3], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

注解

使用说明：

1. 请确保 `object_name` 与 `object_type` 对应且已在系统中定义；
2. `threshold` 的单位依据 `object_type` 不同而不同，务必使用正确单位以避免触发误判；

modbusSetOutputSignal()

```
int arcs::common_interface::RegisterControl::modbusSetOutputSignal (
    const std::string & signal_name,
    uint16_t value)
```

将指定名称的输出寄存器信号设置为给定的值

参数

<i>signal_name</i>	提前被添加的输出寄存器信号
<i>value</i>	必须是有效的整数，范围是 0-65535

返回

Python 函数原型

```
modbusSetOutputSignal(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: int) -> int
```

Lua 函数原型

```
modbusSetOutputSignal(signal_name: string, value: number) -> nil
```

Lua 示例

```
modbusSetOutputSignal("Modbus_0",0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.modbusSetOutputSignal","params":["Modbus_0",0]←
,"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

modbusSetOutputSignal1()

```
int arcs::common_interface::RegisterControl::modbusSetOutputSignal1 (
    const std::string & signal_name,
    const std::vector< uint16_t > & values)
```

将从指定名称开始的若干个连续输出寄存器信号设置为给定的值

参数

<i>signal_name</i>	提前被添加的输出寄存器信号的起始名称
<i>values</i>	整数数组，每个元素范围是 0-65535，对应连续的多个信号

返回

Python 函数原型

```
modbusSetOutputSignal1(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: list[int]) -> int
```

Lua 函数原型

```
modbusSetOutputSignal1(signal_name: string, value: table) -> nil
```

Lua 示例

```
modbusSetOutputSignal1("Modbus_0", {0, 1, 2})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.modbusSetOutputSignal1","params":["Modbus_0",[0,1,2]←
],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

modbusSetOutputSignalPulse()

```
int arcs::common_interface::RegisterControl::modbusSetOutputSignalPulse (  
    const std::string & signal_name,  
    uint16_t value,  
    double duration)
```

设置 modbus 信号输出脉冲 (仅支持线圈输出类型)

参数

<i>signal_name</i>	提前被添加的输出寄存器信号
<i>value</i>	必须是有效的整数, 范围是 0-65535
<i>duration</i>	信号持续时间, 单位为秒

返回

Python 函数原型

```
modbusSetOutputSignalPulse(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: int, arg2 double)  
-> int
```

Lua 函数原型

```
modbusSetOutputSignalPulse(signal_name: string, value: number, duration: number) -> nil
```

Lua 示例

```
modbusSetOutputSignalPulse("Modbus_0",1,0.5)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.modbusSetOutputSignalPulse","params":["Modbus_0",1,0.5],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

modbusSetSignalUpdateFrequency()

```
int arcs::common_interface::RegisterControl::modbusSetSignalUpdateFrequency (  
    const std::string & signal_name,  
    int update_frequency)
```

设置机器人向Modbus 控制器发送请求的频率, 用于读取或写入信号值

参数

<i>signal_name</i>	提前被添加的输出数字信号
<i>update_frequency</i>	更新频率 (以赫兹为单位), 范围是 0-125

返回

Python 函数原型

```
modbusSetSignalUpdateFrequency(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: int) -> int
```

Lua 函数原型

```
modbusSetSignalUpdateFrequency(signal_name: string, update_frequency: number) -> nil
```

Lua 示例

```
modbusSetSignalUpdateFrequency("Modbus_0",1)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.modbusSetSignalUpdateFrequency","params":["Modbus_0",1],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setBool()

```
int arcs::common_interface::RegisterControl::setBool (
    const std::string & key,
    bool value)
```

设置/更新变量值

参数

<i>key</i>	
<i>value</i>	

返回

Python 函数原型

```
setBool(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: bool) -> int
```

Lua 函数原型

```
setBool(key: string, value: boolean) -> nil
```

Lua 示例

```
setBool("custom",true)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.setBool","params":["custom",true],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setBoolInput()

```
int arcs::common_interface::RegisterControl::setBoolInput (
    uint32_t address,
    bool value)
```

参数

<i>address</i>	
<i>value</i>	

返回

注解

只有在实现 RTDE/Modbus Slave/PLC 服务端时使用

Python 函数原型

```
setBoolInput(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: bool) -> int
```

Lua 函数原型

```
setBoolInput(address: number, value: boolean) -> nil
```

Lua 示例

```
setBoolInput(0)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RegisterControl.setBoolInput", "params": [0, true], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setBoolOutput()

```
int arcs::common_interface::RegisterControl::setBoolOutput (
    uint32_t address,
    bool value)
```

参数

<i>address</i>	寄存器地址 (0:127)
<i>value</i>	要设置的布尔值 (true 或 false)

返回

返回 0 表示成功, 其他为错误码

Python 函数原型

```
setBoolOutput(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: bool) -> int
```

Lua 函数原型

```
setBoolOutput(address: number, value: boolean) -> nil
```

Lua 示例

```
setBoolOutput(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.setBoolOutput","params":[0,false],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setDouble()

```
int arcs::common_interface::RegisterControl::setDouble (
    const std::string & key,
    double value)
```

设置/更新变量值

参数

<i>key</i>	
<i>value</i>	

返回

Python 函数原型

```
setDouble(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: float) -> int
```

Lua 函数原型

```
setDouble(key: string, value: number) -> nil
```

Lua 示例

```
setDouble("custom",6.6)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.setDouble","params":["custom",6.6],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setDoubleInput()

```
int arcs::common_interface::RegisterControl::setDoubleInput (
    uint32_t address,
    double value)
```

参数

<i>address</i>	
<i>value</i>	

返回

注解

只有在实现 RTDE/Modbus Slave/PLC 服务端时使用

Python 函数原型

```
setDoubleInput(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: float) -> int
```

Lua 函数原型

```
setDoubleInput(address: number, value: number) -> nil
```

Lua 示例

```
setDoubleInput(0, 3.3)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RegisterControl.setDoubleInput", "params": [0, 6.6], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setDoubleOutput()

```
int arcs::common_interface::RegisterControl::setDoubleOutput (
    uint32_t address,
    double value)
```

参数

<i>address</i>	寄存器地址
<i>value</i>	要设置的双精度浮点数值

返回

返回 0 表示成功，其他为错误码

Python 函数原型

```
setDoubleOutput(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: float) -> int
```

Lua 函数原型

```
setDoubleOutput(address: number, value: number) -> nil
```

Lua 示例

```
setDoubleOutput(0,4.4)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.setDoubleOutput","params":[0,4.4],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setFloat()

```
int arcs::common_interface::RegisterControl::setFloat (
    const std::string & key,
    float value)
```

设置/更新变量值

参数

<i>key</i>	
<i>value</i>	

返回

Python 函数原型

```
setFloat(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: float) -> int
```

Lua 函数原型

```
setFloat(key: string, value: number) -> nil
```

Lua 示例

```
setFloat("custom",4.4)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.setFloat","params":["custom",4.4],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setFloatInput()

```
int arcs::common_interface::RegisterControl::setFloatInput (
```

参数

<i>address</i>	寄存器的地址 (0:47)
<i>value</i>	要设置的浮点数值

返回

返回 0 表示成功, 其他为错误码

注解

只有在实现 RTDE/Modbus Slave/PLC 服务端时使用

Python 函数原型

```
setFloatInput(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: float) -> int
```

Lua 函数原型

```
setFloatInput(address: number, value: number) -> nil
```

Lua 示例

```
setFloatInput(0, 3.3)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RegisterControl.setFloatInput", "params": [0, 3.3], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setFloatOutput()

```
int arcs::common_interface::RegisterControl::setFloatOutput (  
    uint32_t address,  
    float value)
```

参数

<i>address</i>	寄存器地址 (0:47)
<i>value</i>	要设置的浮点数值

返回

返回 0 表示成功, 其他为错误码

Python 函数原型

```
setFloatOutput(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: float) -> int
```


Lua 函数原型

```
setFloatOutput(address: number, value: number) -> nil
```

Lua 示例

```
setFloatOutput(0,5.5)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.setFloatOutput","params":[0,5.5],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setInt16Register()

```
int arcs::common_interface::RegisterControl::setInt16Register (
    uint32_t address,
    int16_t value)
```

参数

<i>address</i>	寄存器地址
<i>value</i>	要设置的值

返回

返回 0 表示成功，其他为错误码

Python 函数原型

```
setInt16Register(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: int) -> int
```

Lua 函数原型

```
setInt16Register(address: number, value: number) -> nil
```

Lua 示例

```
setInt16Register(0,4.4)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.setInt16Register","params":[0,0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setInt16RegisterBit()

```
int arcs::common_interface::RegisterControl::setInt16RegisterBit (
    uint32_t address,
    uint8_t bit_offset,
    bool value)
```

设置 Int16 寄存器的某个 bit 的状态

参数

<i>address</i>	Int16 寄存器地址
<i>bit_offset</i>	寄存器中的 bit 偏移, 0 ~ 15
<i>value</i>	要设置的值, true 表示置 1, false 表示清 0

返回

成功返回 0, 失败返回错误码

Python 函数原型

```
setInt16RegisterBit(self: pyaubo_sdk.RegisterControl, address: int, bit_offset: int, value: bool)
-> int
```

Lua 函数原型

```
setInt16RegisterBit(address: number, bit_offset: number, value: boolean) -> number
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RegisterControl.setInt16RegisterBit", "params": [1, 5, true], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setInt32()

```
int arcs::common_interface::RegisterControl::setInt32 (
    const std::string & key,
    int value)
```

设置/更新变量值

参数

<i>key</i>	
<i>value</i>	

返回**Python 函数原型**

```
setInt32(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: int) -> int
```

Lua 函数原型

```
setInt32(key: string, value: number) -> nil
```

Lua 示例

```
setInt32("custom",6)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.setInt32","params":["custom",6],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setInt32Input()

```
int arcs::common_interface::RegisterControl::setInt32Input (  
    uint32_t address,  
    int value)
```

参数

<i>address</i>	寄存器的地址 (0:47)
<i>value</i>	要设置的整数值

返回

返回 0 表示成功, 其他为错误码

注解

只有在实现 RTDE/Modbus Slave/PLC 服务端时使用

Python 函数原型

```
setInt32Input(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: int) -> int
```

Lua 函数原型

```
setInt32Input(address: number, value: number) -> nil
```

Lua 示例

```
setInt32Input(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.setInt32Input","params":[0,33],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setInt32Output()

```
int arcs::common_interface::RegisterControl::setInt32Output (  
    uint32_t address,  
    int value)
```

参数

<i>address</i>	寄存器地址 (0:47)
<i>value</i>	要设置的整数值

返回

返回 0 表示成功, 其他为错误码

Python 函数原型

```
setInt32Output(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: int) -> int
```

Lua 函数原型

```
setInt32Output(address: number, value: number) -> nil
```

Lua 示例

```
setInt32Output(0, 100)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RegisterControl.setInt32Output", "params": [0, 100], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setString()

```
int arcs::common_interface::RegisterControl::setString (  
    const std::string & key,  
    const std::string & value)  
设置/更新变量值
```

参数

<i>key</i>	
<i>value</i>	

返回**Python 函数原型**

```
setString(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: str) -> int
```

Lua 函数原型

```
setString(key: string, value: string) -> nil
```

Lua 示例

```
setString("custom","test")
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.setString","params":["custom","test"],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setVecChar()

```
int arcs::common_interface::RegisterControl::setVecChar (
    const std::string & key,
    const std::vector< char > & value)
设置/更新变量值
```

参数

<i>key</i>	
<i>value</i>	

返回

Python 函数原型

```
setVecChar(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: List[str]) -> int
```

Lua 函数原型

```
setVecChar(key: string, value: table) -> nil
```

Lua 示例

```
setVecChar("custom",{0,1,0})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.setVecChar","params":["custom",[0,1,0]],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setVecDouble()

```
int arcs::common_interface::RegisterControl::setVecDouble (
    const std::string & key,
    const std::vector< double > & value)
设置/更新变量值
```

参数

<i>key</i>	
<i>value</i>	

返回

Python 函数原型

```
setVecDouble(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: List[float]) -> int
```

Lua 函数原型

```
setVecDouble(key: string, value: table) -> nil
```

Lua 示例

```
setVecDouble("custom",{0.1,0.2,0.3})
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "RegisterControl.setVecDouble", "params": ["custom", [0.1, 0.2, 0.3]], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

setVecFloat()

```
int arcs::common_interface::RegisterControl::setVecFloat (
    const std::string & key,
    const std::vector< float > & value)
```

设置/更新变量值

参数

<i>key</i>	
<i>value</i>	

返回

Python 函数原型

```
setVecFloat(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: List[float]) -> int
```

Lua 函数原型

```
setVecFloat(key: string, value: table) -> nil
```

Lua 示例

```
setVecFloat("custom", {0.0,0.1,3.3})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.setVecFloat","params":["custom",[0.0,0.1,3.3]],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setVecInt32()

```
int arcs::common_interface::RegisterControl::setVecInt32 (
    const std::string & key,
    const std::vector< int32_t > & value)
```

设置/更新变量值

参数

<i>key</i>	
<i>value</i>	

返回

Python 函数原型

```
setVecInt32(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: List[int]) -> int
```

Lua 函数原型

```
setVecInt32(key: string, value: table) -> nil
```

Lua 示例

```
setVecInt32("custom",{1,2,3,4})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.setVecInt32","params":["custom",[1,2,3,4]],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setWatchDog()

```
int arcs::common_interface::RegisterControl::setWatchDog (
    const std::string & key,
    double timeout,
    int action)
```

设置看门狗

看门狗被触发之后控制器会执行对应的动作，并自动删除看门狗

参数

<i>key</i>	
<i>timeout</i>	超时时间, 单位秒 (s), 超时时间最小为 0.1s
<i>action</i>	NONE (0): 无动作 PAUSE(1): 暂停运行时 STOP (2): 停止运行时/停止机器人运动 PROTECTIVE_STOP (3): 触发防护停止

返回

variableUpdated()

```
bool arcs::common_interface::RegisterControl::variableUpdated (  
    const std::string & key,  
    uint64_t since)  
具名变量是否更新
```

参数

<i>key</i>	
<i>since</i>	

返回

Python 函数原型

```
variableUpdated(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: int) -> bool
```

Lua 函数原型

```
variableUpdated(key: string, since: number) -> boolean
```

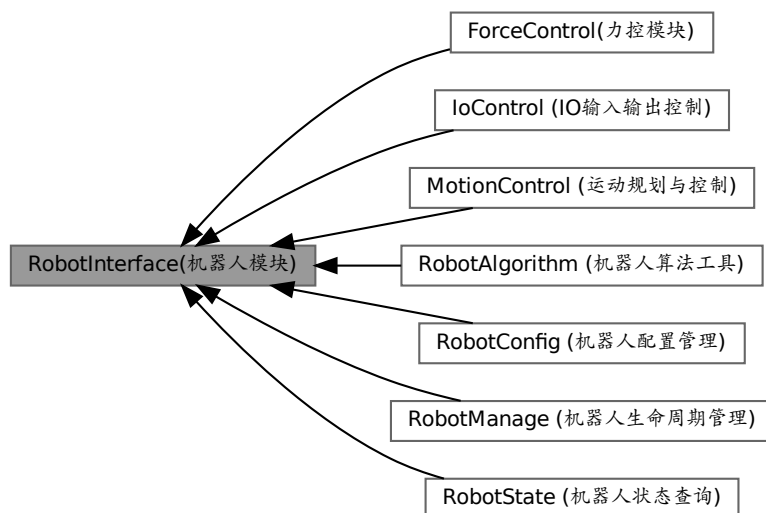

Lua 示例

```
Variable_Updated = variableUpdated("custom" , 0)
```

8.6 RobotInterface(机器人模块)

机器人模块

RobotInterface(机器人模块) 的协作图:



专题

- [ForceControl\(力控模块\)](#)
力控接口抽象类
- [IoControl \(IO 输入输出控制\)](#)
IoControl 类提供了一系列的接口对机器人标配的一些数字、模拟IO 进行配置，输出状态设置、读取
- [MotionControl \(运动规划与控制\)](#)
MotionControl
- [RobotAlgorithm \(机器人算法工具\)](#)
机器人算法相关的对外接口
- [RobotConfig \(机器人配置管理\)](#)
RobotConfig
- [RobotManage \(机器人生命周期管理\)](#)
RobotManage
- [RobotState \(机器人状态查询\)](#)
RobotState

函数

- [RobotConfigPtr arcs::common_interface::RobotInterface::getRobotConfig \(\)](#)
RobotConfig (机器人配置管理) 获取*RobotConfig* 接口
- [MotionControlPtr arcs::common_interface::RobotInterface::getMotionControl \(\)](#)
MotionControl (运动规划与控制) 获取运动规划接口
- [ForceControlPtr arcs::common_interface::RobotInterface::getForceControl \(\)](#)

- ForceControl*(力控模块) 获取力控接口
- `IoControlPtr arcs::common_interface::RobotInterface::getIoControl ()`
IoControl (IO 输入输出控制) 获取IO 控制的接口
- `SyncMovePtr arcs::common_interface::RobotInterface::getSyncMove ()`
SyncMove (同步运动控制和轴组管理) 获取同步运动接口
- `RobotAlgorithmPtr arcs::common_interface::RobotInterface::getRobotAlgorithm ()`
RobotAlgorithm (机器人算法工具) 获取机器人实用算法接口
- `RobotManagePtr arcs::common_interface::RobotInterface::getRobotManage ()`
RobotManage (机器人生命周期管理) 获取机器人管理接口 (上电、启动、停止等)
- `RobotStatePtr arcs::common_interface::RobotInterface::getRobotState ()`
RobotState (机器人状态查询) 获取机器人状态接口
- `TracePtr arcs::common_interface::RobotInterface::getTrace ()`
Trace (日志与弹窗) 获取告警信息接口

8.6.1 详细描述

机器人模块

8.6.2 函数说明

getForceControl()

`ForceControlPtr arcs::common_interface::RobotInterface::getForceControl ()`
ForceControl(力控模块) 获取力控接口

返回

`ForceControlPtr` 对象的指针

Python 函数原型

`getForceControl(self: pyaubo_sdk.RobotInterface) -> arcs::common_interface::ForceControl`

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
ForceControlPtr ptr =
rpc_cli->getRobotInterface(robot_name)->getForceControl();
```

getIoControl()

`IoControlPtr arcs::common_interface::RobotInterface::getIoControl ()`
IoControl (IO 输入输出控制) 获取IO 控制的接口

返回

`IoControlPtr` 对象的指针

Python 函数原型

`getIoControl(self: pyaubo_sdk.RobotInterface) -> arcs::common_interface::IoControl`

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
IoControlPtr ptr =
rpc_cli->getRobotInterface(robot_name)->getIoControl();
```

getMotionControl()

`MotionControlPtr` `arcs::common_interface::RobotInterface::getMotionControl ()`
`MotionControl` (运动规划与控制) 获取运动规划接口

返回

`MotionControlPtr` 对象的指针

Python 函数原型

`getMotionControl(self: pyaubo_sdk.RobotInterface) -> arcs::common_interface::MotionControl`

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
MotionControlPtr ptr =
rpc_cli->getRobotInterface(robot_name)->getMotionControl();
```

getRobotAlgorithm()

`RobotAlgorithmPtr` `arcs::common_interface::RobotInterface::getRobotAlgorithm ()`
`RobotAlgorithm` (机器人算法工具) 获取机器人实用算法接口

返回

`RobotAlgorithmPtr` 对象的指针

Python 函数原型

`getRobotAlgorithm(self: pyaubo_sdk.RobotInterface) -> arcs::common_interface::RobotAlgorithm`

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
RobotAlgorithmPtr ptr =
rpc_cli->getRobotInterface(robot_name)->getRobotAlgorithm();
```

getRobotConfig()

`RobotConfigPtr` `arcs::common_interface::RobotInterface::getRobotConfig ()`
`RobotConfig` (机器人配置管理) 获取RobotConfig 接口

返回

`RobotConfigPtr` 对象的指针

Python 函数原型

`getRobotConfig(self: pyaubo_sdk.RobotInterface) -> arcs::common_interface::RobotConfig`

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
RobotConfigPtr ptr =
rpc_cli->getRobotInterface(robot_name)->getRobotConfig();
```

getRobotManage()

`RobotManagePtr` `arcs::common_interface::RobotInterface::getRobotManage ()`
`RobotManage` (机器人生命周期管理) 获取机器人管理接口 (上电、启动、停止等)

返回

`RobotManagePtr` 对象的指针

Python 函数原型

`getRobotManage(self: pyaubo_sdk.RobotInterface) -> arcs::common_interface::RobotManage`

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
RobotManagePtr ptr =
rpc_cli->getRobotInterface(robot_name)->getRobotManage();
```

getRobotState()

`RobotStatePtr` `arcs::common_interface::RobotInterface::getRobotState ()`
`RobotState` (机器人状态查询) 获取机器人状态接口

返回

`RobotStatePtr` 对象的指针

Python 函数原型

`getRobotState(self: pyaubo_sdk.RobotInterface) -> arcs::common_interface::RobotState`

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
RobotStatePtr ptr =
rpc_cli->getRobotInterface(robot_name)->getRobotState();
```

getSyncMove()

`SyncMovePtr` `arcs::common_interface::RobotInterface::getSyncMove ()`
`SyncMove` (同步运动控制和轴组管理) 获取同步运动接口

返回

`SyncMovePtr` 对象的指针

Python 函数原型

`getSyncMove(self: pyaubo_sdk.RobotInterface) -> arcs::common_interface::SyncMove`

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
SyncMovePtr ptr = rpc_cli->getRobotInterface(robot_name)->getSyncMove();
```

getTrace()

`TracePtr arcs::common_interface::RobotInterface::getTrace ()`
 Trace (日志与弹窗) 获取告警信息接口

返回

TracePtr 对象的指针

Python 函数原型

`getTrace(self: pyaubo_sdk.RobotInterface) -> arcs::common_interface::Trace`

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
TracePtr ptr = rpc_cli->getRobotInterface(robot_name)->getTrace();
```

8.6.3 ForceControl(力控模块)

力控接口抽象类

ForceControl(力控模块) 的协作图:

**函数**

- `int arcs::common_interface::ForceControl::fcEnable ()`
 使能力控。 *fcEnable* 被用于使能力控。在力控被使能的同时, *fcEnable* 用于定义力控的坐标系, 并调整力和力矩的阻尼。如果在 *fcEnable* 中未指定坐标系, 则会创建一个默认的力控制坐标系, 其方向与工作对象坐标系相同。所有力控制监管功能都被 *fcEnable* 激活。
- `int arcs::common_interface::ForceControl::fcDisable ()`
 失能力控。 *fcDisable* 被用于失能力控。在成功失能力控之后, 机器人将回到位置控制模式。
- `bool arcs::common_interface::ForceControl::isFcEnabled ()`
 判断力控是否被使能
- `int arcs::common_interface::ForceControl::setTargetForce (const std::vector< double > &feature, const std::vector< bool > &compliance, const std::vector< double > &wrench, const std::vector< double > &limits, TaskFrameType type=TaskFrameType::FRAME_FORCE)`
 设置力控参考 (目标) 值
- `int arcs::common_interface::ForceControl::setDynamicModel1 (const std::vector< double > &env←_stiff, const std::vector< double > &damp_scale, const std::vector< double > &stiff_scale)`
 设置力控动力学模型
- `DynamicsModel arcs::common_interface::ForceControl::fcCalDynamicModel (const std::vector< double > &env←_stiff, const std::vector< double > &damp_scale, const std::vector< double > &stiff←_scale)`
 计算力控动力学模型
- `int arcs::common_interface::ForceControl::setDynamicModelSearch (const std::vector< double > &damp_scale, const std::vector< double > &stiff_scale)`
 设置力控搜孔场景下的动力学模型

- `int arcs::common_interface::ForceControl::setDynamicModelInsert (const std::vector< double > &damp_scale, const std::vector< double > &stiff_scale)`
设置力控插/拔孔场景下的动力学模型
- `int arcs::common_interface::ForceControl::setDynamicModelContact (const std::vector< double > &env_stiff, const std::vector< double > &damp_scale, const std::vector< double > &stiff_scale)`
设置力控接触场景下的动力学模型
- `int arcs::common_interface::ForceControl::setDynamicModel (const std::vector< double > &m, const std::vector< double > &d, const std::vector< double > &k)`
设置力控动力学模型
- `int arcs::common_interface::ForceControl::fcSetSensorThresholds (const std::vector< double > &thresholds)`
设置力控阈值
- `int arcs::common_interface::ForceControl::fcSetSensorLimits (const std::vector< double > &limits)`
设置力控最大受力限制
- `std::vector< double > arcs::common_interface::ForceControl::getFcSensorThresholds ()`
获取力控阈值
- `std::vector< double > arcs::common_interface::ForceControl::getFcSensorLimits ()`
获取最大力限制
- `DynamicsModel arcs::common_interface::ForceControl::getDynamicModel ()`
获取力控动力学模型
- `int arcs::common_interface::ForceControl::setCondForce (const std::vector< double > &min, const std::vector< double > &max, bool outside, double timeout)`
设置力控终止条件: 力, 当测量的力在设置的范围之内, 力控算法将保持运行, 直到设置的条件不满足, 力控将退出
- `int arcs::common_interface::ForceControl::setCondOrient (const std::vector< double > &frame, double max_angle, double max_rot, bool outside, double timeout)`
设置力控终止条件: 姿态, 当测量的姿态在设置的范围之内, 力控算法将保持运行, 直到设置的条件不满足, 力控将退出.
- `int arcs::common_interface::ForceControl::setCondPlane (const std::vector< double > &plane, double timeout)`
指定力控有效平面, x - y 平面, z 方向有效
- `int arcs::common_interface::ForceControl::setCondCylinder (const std::vector< double > &axis, double radius, bool outside, double timeout)`
指定力控有效圆柱体, 提供中心轴和圆柱半径, 可以指定圆柱内部还是外部
- `int arcs::common_interface::ForceControl::setCondSphere (const std::vector< double > &center, double radius, bool outside, double timeout)`
指定力控有效球体, 提供球心和半径, 可以指定球体内部还是外部
- `int arcs::common_interface::ForceControl::setCondTcpSpeed (const std::vector< double > &min, const std::vector< double > &max, bool outside, double timeout)`
设置 TCP 速度的终止条件。该条件可通过调用 *FCCondWaitWhile* 指令激活, 在指定条件为真时, 程序将等待并保持执行。这样可以使参考力、力矩和运动继续, 直到速度超出指定范围。
- `int arcs::common_interface::ForceControl::setCondDistance (double distance, double timeout)`
力控终止条件-距离
- `int arcs::common_interface::ForceControl::setCondAdvanced (const std::string &type, const std::vector< double > &args, double timeout)`
高级力控终止条件
- `int arcs::common_interface::ForceControl::setCondActive ()`
激活力控终止条件
- `bool arcs::common_interface::ForceControl::isCondFullfiled ()`
力控终止条件是否已经满足
- `int arcs::common_interface::ForceControl::setSupvForce (const std::vector< double > &min, const std::vector< double > &max)`
setSupvForce 用于在力控中设置力监督。监督在通过 *FCAct* 指令激活力控时被激活。

- `int arcs::common_interface::ForceControl::setSupvOrient` (`const std::vector< double > &frame`, `double max_angle`, `double max_rot`, `bool outside`)
`setSupvOrient` 用于设置工具姿态的监督条件。当通过 `FCAct` 指令激活力控时, 监督条件被激活。
- `int arcs::common_interface::ForceControl::setSupvPosBox` (`const std::vector< double > &frame`, `const Box &box`)
`setSupvPosBox` 用于在力控中设置位置监督。监督在通过 `FCAct` 指令激活力控时被激活。位置监督通过为 `TCP` 定义空间体积来设置。一旦激活, 如果 `TCP` 超出该体积, 监督将停止执行。
- `int arcs::common_interface::ForceControl::setSupvPosCylinder` (`const std::vector< double > &frame`, `const Cylinder &cylinder`)
- `int arcs::common_interface::ForceControl::setSupvPosSphere` (`const std::vector< double > &frame`, `const Sphere &sphere`)
- `int arcs::common_interface::ForceControl::setSupvReoriSpeed` (`const std::vector< double > &speed`, `double limit`, `bool outside`, `double timeout`)
`setSupvReoriSpeed` 用于在力控中设置重新定向速度监督。监督在通过 `FCAct` 指令激活力控时被激活。
- `int arcs::common_interface::ForceControl::setSupvTcpSpeed` (`const std::vector< double > &speed`, `double limit`, `bool outside`, `double timeout`)
`setSupvTcpSpeed` 用于在力控中设置 `TCP` 速度监督。监督在通过 `FCAct` 指令激活力控时被激活。 `TCP` 速度监督通过定义工作对象坐标系各方向上的最小和最大速度限制来设置。一旦激活, 如果检测到过高的 `TCP` 速度值, 监督将停止执行。
- `int arcs::common_interface::ForceControl::setLpFilter` (`const std::vector< double > &cutoff_freq`)
 设置低通滤波器
- `int arcs::common_interface::ForceControl::resetLpFilter` ()
 重置低通滤波器
- `int arcs::common_interface::ForceControl::speedChangeEnable` (`double ref_force`)
`speedChangeEnable` 用于激活 `FC SpeedChange` 功能, 并设置期望的参考力和恢复行为。当 `FC SpeedChange` 功能被激活时, 机器人速度会根据测量信号与参考值的接近程度自动降低或提高。
- `int arcs::common_interface::ForceControl::speedChangeDisable` ()
 停用 `FC SpeedChange` 功能。
- `int arcs::common_interface::ForceControl::speedChangeTune` (`int speed_levels`, `double speed_ratio_min`)
`speedChangeTune` 用于将 `FC SpeedChange` 系统参数设置为新值。
- `int arcs::common_interface::ForceControl::setDamping` (`const std::vector< double > &damping`, `double ramp_time`)
`setDamping` 用于在力控坐标系中调整阻尼。可调参数包括扭矩 x 方向到扭矩 z 方向的阻尼 (见第 255 页) 以及力 x 方向到力 z 方向的阻尼 (见第 254 页)。
- `int arcs::common_interface::ForceControl::resetDamping` ()
 重置阻尼参数
- `int arcs::common_interface::ForceControl::softFloatEnable` ()
 启用软浮动功能。
- `int arcs::common_interface::ForceControl::softFloatDisable` ()
 停用软浮动功能。
- `bool arcs::common_interface::ForceControl::isSoftFloatEnabled` ()
 返回是否开启了软浮动
- `int arcs::common_interface::ForceControl::setSoftFloatParams` (`bool joint_space`, `const std::vector< bool > &select`, `const std::vector< double > &stiff_percent`, `const std::vector< double > &stiff_damp_ratio`, `const std::vector< double > &force_threshold`, `const std::vector< double > &force_limit`)
 设置软浮动参数
- `int arcs::common_interface::ForceControl::toolContact` (`const std::vector< bool > &direction`)
 检测工具和外部物体的接触
- `std::vector< double > arcs::common_interface::ForceControl::getActualJointPositionsHistory` (`int steps`)

详细描述

力控接口抽象类

函数说明

fcCalDynamicModel()

```
DynamicsModel arcs::common_interface::ForceControl::fcCalDynamicModel (
    const std::vector< double > & env_stiff,
    const std::vector< double > & damp_scale,
    const std::vector< double > & stiff_scale)
```

计算力控动力学模型

参数

<i>env_stiff</i>	环境刚度，表示为接触轴方向上的工件刚度，取值范围[0, 1]，默认为 0
<i>damp_scale</i>	表征阻尼水平的参数，取值范围[0, 1]，默认为 0.5
<i>stiff_scale</i>	表征软硬程度的参数，取值范围[0, 1]，默认为 0.5

返回

力控动力学模型MDK AUBO_BUSY AUBO_BAD_STATE-AUBO_INVL_ARGUMENT-AUBO←
_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
fcCalDynamicModel(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: List[float], arg2: List←
[float]) -> Tuple[List[float], List[float], List[float]]
```

Lua 函数原型

```
fcCalDynamicModel(env_stiff: table, damp_scale: table, stiff_scale: table) -> table
```

Lua 示例

```
fc_table = fcCalDynamicModel({0,0,0,0,0,0},{0.5,0.5,0.5,0.5,0.5,0.5},{0.5,0.5,0.5,0.5,0.5,0.5})
```

fcDisable()

```
int arcs::common_interface::ForceControl::fcDisable ()
```

失能力控。fcDisable 被用于失能力控。在成功失能力控之后，机器人将回到位置控制模式。

返回

成功返回 0；失败返回错误码 AUBO_BUSY AUBO_BAD_STATE-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
fcDisable(self: pyaubo_sdk.ForceControl) -> int
```

Lua 函数原型

```
fcDisable() -> nil
```

Lua 示例

```
fcDisable()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.ForceControl.fcDisable","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

fcEnable()

```
int arcs::common_interface::ForceControl::fcEnable ()
```

使能力控。fcEnable 被用于使能力控。在力控被使能的同时，fcEnable 用于定义力控的坐标系，并调整力和力矩的阻尼。如果在 fcEnable 中未指定坐标系，则会创建一个默认的力控制坐标系，其方向与工作对象坐标系相同。所有力控制监管功能都被 fcEnable 激活。

返回

成功返回 0；失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
fcEnable(self: pyaubo_sdk.ForceControl) -> int
```

Lua 函数原型

```
fcEnable() -> nil
```

Lua 示例

```
fcEnable()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.ForceControl.fcEnable","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

fcSetSensorLimits()

```
int arcs::common_interface::ForceControl::fcSetSensorLimits (
    const std::vector< double > & limits)
```

设置力控最大受力限制

参数

<i>limits</i>	力限制

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
fcSetSensorLimits(self: pyaubo_sdk.ForceControl, arg0: List[float]) -> int
```

Lua 函数原型

```
fcSetSensorLimits(limits: table) -> nil
```

Lua 示例

```
fcSetSensorLimits({200.0,200.0,200.0,50.0,50.0,50.0})
```

fcSetSensorThresholds()

```
int arcs::common_interface::ForceControl::fcSetSensorThresholds (
    const std::vector< double > & thresholds)
```

设置力控阈值

参数

<i>thresholds</i>	力控國值

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
fcSetSensorThresholds(self: pyaubo_sdk.ForceControl, arg0: List[float]) -> int
```

Lua 函数原型

```
fcSetSensorThresholds(thresholds: table) -> nil
```

Lua 示例

```
fcSetSensorThresholds({1.0,1.0,1.0,0.5,0.5,0.5})
```

\endch

getActualJointPositionsHistory()

```
std::vector< double > arcs::common_interface::ForceControl::getActualJointPositionsHistory (
    int steps)
```

获取历史关节角度

根据给定的周期步数，从关节状态历史中回退指定数量的周期，获取当时的关节角度数据。

参数

<i>steps</i>	需要回退的周期数（单位：控制周期数），值越大表示获取越早的历史数据

返回

std::vector<double> 对应时间点的各关节角度（单位：弧度）

getDynamicModel()

```
DynamicsModel arcs::common_interface::ForceControl::getDynamicModel ()
```

获取力控动力学模型

返回

力控动力学模型

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
getDynamicModel(self: pyaubo_sdk.ForceControl) -> Tuple[List[float], List[float], List[float]]
```

Lua 函数原型

```
getDynamicModel() -> table
```

Lua 示例

```
Fc_Model = getDynamicModel()
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.ForceControl.getDynamicModel", "params": [], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": [[], [20.0, 20.0, 20.0, 5.0, 5.0, 5.0], []] }
```

getFcSensorLimits()

```
std::vector< double > arcs::common_interface::ForceControl::getFcSensorLimits ()  
获取最大力限制
```

返回

力控最大力限制

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getFcSensorLimits(self: pyaubo_sdk.ForceControl) -> list
```

Lua 函数原型

```
getFcSensorLimits() -> table
```

Lua 示例

```
Fc_Limits = getFcSensorLimits()
```

getFcSensorThresholds()

```
std::vector< double > arcs::common_interface::ForceControl::getFcSensorThresholds ()  
获取力控阈值
```

返回

力控最小力阈值

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getFcSensorThresholds(self: pyaubo_sdk.ForceControl) -> list
```

Lua 函数原型

```
getFcSensorThresholds() -> table
```

Lua 函数示例

```
Fc_Thresholds = getFcSensorThresholds()
```

isCondFullfiled()

```
bool arcs::common_interface::ForceControl::isCondFullfiled ()
    力控终止条件是否已经满足
```

返回

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
isCondFullfiled(self: pyaubo_sdk.ForceControl) -> bool
```

Lua 函数原型

```
isCondFullfiled() -> boolean
```

Lua 示例

```
status = isCondFullfiled()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.ForceControl.isCondFullfiled","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":false}
```

isFcEnabled()

```
bool arcs::common_interface::ForceControl::isFcEnabled ()
    判断力控是否被使能
```

返回

使能返回 true, 失能返回 false

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
isFcEnabled(self: pyaubo_sdk.ForceControl) -> bool
```

Lua 函数原型

```
isFcEnabled() -> boolean
```

Lua 示例

```
Fc_status = isFcEnabled()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.ForceControl.isFcEnabled","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":false}
```

isSoftFloatEnabled()

```
bool arcs::common_interface::ForceControl::isSoftFloatEnabled ()  
返回是否开启了软浮动
```

返回

异常

<code>arcs::common_interface::AuboException</code>

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.ForceControl.isSoftFloatEnabled","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":false}
```

resetDamping()

```
int arcs::common_interface::ForceControl::resetDamping ()  
重置阻尼参数
```

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
resetDamping(self: pyaubo_sdk.ForceControl) -> int
```

Lua 函数原型

```
resetDamping() -> nil
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.ForceControl.resetDamping", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

resetLpFilter()

```
int arcs::common_interface::ForceControl::resetLpFilter ()  
重置低通滤波器
```

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
resetLpFilter(self: pyaubo_sdk.ForceControl) -> int
```

Lua 函数原型

```
resetLpFilter() -> nil
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.ForceControl.resetLpFilter", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setCondActive()

```
int arcs::common_interface::ForceControl::setCondActive ()  
激活力控终止条件
```

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
setCondActive(self: pyaubo_sdk.ForceControl) -> int
```

Lua 函数原型

```
setCondActive() -> nil
```

Lua 示例

```
setCondActive()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.ForceControl.setCondActive","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setCondAdvanced()

```
int arcs::common_interface::ForceControl::setCondAdvanced (
    const std::string & type,
    const std::vector< double > & args,
    double timeout)
```

高级力控终止条件

参数

<i>type</i>	类型
<i>args</i>	参数
<i>timeout</i>	超时时间

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Lua 函数原型

```
setCondAdvanced(type: number, args: table, timeout: number)
```

Lua 示例

```
setCondAdvanced(1, {0,1,0,0,0,0},10)
```

setCondCylinder()

```
int arcs::common_interface::ForceControl::setCondCylinder (
    const std::vector< double > & axis,
    double radius,
    bool outside,
    double timeout)
```

指定力控有效圆柱体, 提供中心轴和圆柱半径, 可以指定圆柱内部还是外部

参数

<i>axis</i>	
<i>radius</i>	
<i>outside</i>	
<i>timeout</i>	

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setCondCylinder(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: float, arg2: bool, arg3: float) -> int
```

Lua 函数原型

```
setCondCylinder(axis: table, radius: number, outside: boolean, timeout: number) -> nil
```

Lua 示例

```
setCondCylinder({0,1,0},10.0,true,5,0)
```

setCondDistance()

```
int arcs::common_interface::ForceControl::setCondDistance (
    double distance,
    double timeout)
```

力控终止条件-距离

参数

<i>distance</i>	距离
<i>timeout</i>	超时时间

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Lua 函数原型

```
setCondDistance(distance: number, timeout: number)
```

Lua 示例

```
setCondDistance(0.2, 10.0)
```

setCondForce()

```
int arcs::common_interface::ForceControl::setCondForce (
    const std::vector< double > & min,
    const std::vector< double > & max,
    bool outside,
    double timeout)
```

设置力控终止条件：力，当测量的力在设置的范围之内，力控算法将保持运行，直到设置的条件不满足，力控将退出

The condition is later activated by calling the instruction FCCondWaitWhile, which will wait and hold the program execution while the specified condition is true. This allows the reference force, torque and movement to continue until the force is outside the specified limits.

A force condition is set up by defining minimum and maximum limits for the force in the directions of the force control coordinate system. Once activated with FCCondWaitWhile, the program execution will continue to wait while the measured force is within its specified limits.

It is possible to specify that the condition is fulfilled when the force is outside the specified limits instead. This is done by using the switch argument Outside. The condition on force is specified in the force control coordinate system. This coordinate system is setup by the user in the instruction FCAct.

参数

<i>min</i>	各方向最小的力/力矩
<i>max</i>	各方向最大的力/力矩
<i>outside</i>	false 在设置条件的范围之内有效 true 在设置条件的范围之外有效
<i>timeout</i>	时间限制，单位 s(秒)，从开始力控到达该时间时，不管是否满足力控终止条件，都会终止力控

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setCondForce(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: List[float], arg2: bool, arg3: float) -> int
```

Lua 函数原型

```
setCondForce(min: table, max: table, outside: boolean, timeout: number) -> nil
```

Lua 示例

```
setCondForce({0.1,0.1,0.1,0.1,0.1,0.1},{50,50,50,5,5,5},{true,true,true,true,true,true},10)
```

setCondOrient()

```
int arcs::common_interface::ForceControl::setCondOrient (
    const std::vector< double > & frame,
    double max_angle,
    double max_rot,
    bool outside,
    double timeout)
```

设置力控终止条件：姿态，当测量的姿态在设置的范围之内，力控算法将保持运行，直到设置的条件不满足，力控将退出。

setCondOrient is used to set up an end condition for the tool orientation. The condition is later activated by calling the instruction FCCondWaitWhile, which will wait and hold the program execution while the specified condition is true. This allows the reference force, torque and movement to continue until the orientation is outside the specified limits.

An orientation condition is set up by defining a maximum angle and a maximum rotation from a reference orientation. The reference orientation is either defined by the current z direction of the tool, or by specifying an orientation in relation to the z direction of the work object.

Once activated, the tool orientation must be within the limits (or outside, if the argument Outside is used).

参数

<i>frame</i>	
<i>max_angle</i>	
<i>max_rot</i>	
<i>outside</i>	
<i>timeout</i>	

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setCondOrient(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: float, arg2: float, arg3: bool,
arg4: float) -> int
```

Lua 函数原型

```
setCondOrient(frame: table, max_angle: number, max_rot: number, outside: boolean, timeout:
number) -> nil
```

Lua 示例

```
setCondOrient({0.1, 0.3, 0.1, 0.3142, 0.0, 1.571},30.0,20.0,true,10.0)
```

setCondPlane()

```
int arcs::common_interface::ForceControl::setCondPlane (
    const std::vector< double > & plane,
    double timeout)
```

指定力控有效平面, x-y 平面, z 方向有效

参数

$plane=\{A,B,C,D\}$	平面表示方法 $Ax + By + Cz + D = 0$ 其中, $n = (A, B, C)$ 是平面的法向量, D 是将平面平移到坐标原点所需距离 (所以 $D=0$ 时, 平面过原点)
<i>timeout</i>	

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
setCondPlane(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: float) -> int
```

Lua 函数原型

```
setCondPlane(plane: table, timeout: number) -> nil
```

Lua 示例

```
setCondPlane({0.1, 0.3, 0.1, 0.3},10.0)
```

setCondSphere()

```
int arcs::common_interface::ForceControl::setCondSphere (
    const std::vector< double > & center,
    double radius,
    bool outside,
    double timeout)
```

指定力控有效球体, 提供球心和半径, 可以指定球体内部还是外部

参数

<i>center</i>	
<i>radius</i>	
<i>outside</i>	
<i>timeout</i>	

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setCondSphere(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: float, arg2: bool, arg3: float)
-> int
```

Lua 函数原型

```
setCondSphere(center: table, radius: number, outside: boolean, timeout: number) -> nil
```

Lua 示例

```
setCondSphere({0.2, 0.5, 0.1, 1.57, 0, 0},10.0,true,5,0)
```

setCondTcpSpeed()

```
int arcs::common_interface::ForceControl::setCondTcpSpeed (
    const std::vector< double > & min,
    const std::vector< double > & max,
    bool outside,
    double timeout)
```

设置TCP 速度的终止条件。该条件可通过调用FCCondWaitWhile 指令激活，在指定条件为真时，程序将等待并保持执行。这样可以使参考力、力矩和运动继续，直到速度超出指定范围。

通过定义TCP 在工作对象所有方向上的最小和最大速度限制来设置TCP 速度条件。一旦通过FCCondWaitWhile 激活，程序将在测量速度处于指定范围内时继续等待。

也可以指定当速度超出指定范围时条件成立，通过使用 outside 参数实现。TCP 速度条件在工作对象坐标系中指定。

参数

<i>min</i>	最小速度
<i>max</i>	最大速度
<i>outside</i>	是否在范围外有效
<i>timeout</i>	超时时间

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
setCondTcpSpeed(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: List[float], arg2: bool,
arg3: float) -> int
```

Lua 函数原型

```
setCondTcpSpeed(min: table, max: table, outside: boolean, timeout: number) -> nil
```

Lua 示例

```
setCondTcpSpeed({0.2, 0.2, 0.2, 0.2, 0.2, 0.2},{1.2, 1.2, 1.2, 1.2, 1.2, 1.2},true,5,0)
```

setDamping()

```
int arcs::common_interface::ForceControl::setDamping (
    const std::vector< double > & damping,
    double ramp_time)
```

setDamping 用于在力控坐标系中调整阻尼。可调参数包括扭矩 x 方向到扭矩 z 方向的阻尼（见第 255 页）以及力 x 方向到力 z 方向的阻尼（见第 254 页）。

阻尼可以通过配置文件或FCAct 指令设置。不同之处在于本指令可在力控激活时使用。FCSet↔DampingTune 调整的是FCAct 指令设置的实际值，而不是配置文件中的值。

参数

<i>damping</i>	阻尼参数
<i>ramp_time</i>	斜坡时间

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
setDamping(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: float) -> int
```

Lua 函数原型

```
setDamping(damping: table, ramp_time: number) -> nil
```

setDynamicModel()

```
int arcs::common_interface::ForceControl::setDynamicModel (
    const std::vector< double > & m,
    const std::vector< double > & d,
    const std::vector< double > & k)
```

设置力控动力学模型

参数

m	质量参数
d	阻尼参数
k	刚度参数

参数单位说明:

- 质量:
 - 笛卡尔空间: 单位为 kg, 长度为工作空间维度 (一般为 6), 顺序为 [x, y, z, Rx, Ry, Rz]
 - 关节空间: 单位为 kg·m², 长度为机器人自由度 (一般为 6), 顺序为 [J1, J2, J3, J4, J5, J6]
- 阻尼:
 - 笛卡尔空间: 单位为 N·s/m, 顺序为 [x, y, z, Rx, Ry, Rz]
 - 关节空间: 单位为 N·m·s/rad, 顺序为 [J1, J2, J3, J4, J5, J6]
- 刚度:
 - 笛卡尔空间: 单位为 N/m, 顺序为 [x, y, z, Rx, Ry, Rz]
 - 关节空间: 单位为 N·m/rad, 顺序为 [J1, J2, J3, J4, J5, J6]

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setDynamicModel(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: List[float], arg2: List[←
[float]) -> int
```

Lua 函数原型

```
setDynamicModel(m: table, d: table, k: table) -> nil
```

Lua 示例

```
setDynamicModel({20.0, 20.0, 20.0, 10.0, 10.0, 10.0},{2000, 0, 0, 0, 0, 0},{0, 0, 0, 0, 0, 0})
```

setDynamicModel1()

```
int arcs::common_interface::ForceControl::setDynamicModel1 (
    const std::vector< double > & env_stiff,
    const std::vector< double > & damp_scale,
    const std::vector< double > & stiff_scale)
```

设置力控动力学模型

参数

<i>env_stiff</i>	环境刚度，表示为接触轴方向上的工件刚度，取值范围[0, 1]，默认为 0
<i>damp_scale</i>	表征阻尼水平的参数，取值范围[0, 1]，默认为 0.5
<i>stiff_scale</i>	表征软硬程度的参数，取值范围[0, 1]，默认为 0.5

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
setDynamicModel1(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: List[float], arg2: List[←
[float]]) -> int
```

Lua 函数原型

```
setDynamicModel1(env_stiff: table, damp_scale: table, stiff_scale: table) -> nil
```

Lua 示例

```
setDynamicModel1({0,0,0,0,0,0},{0.5,0.5,0.5,0.5,0.5,0.5},{0.5,0.5,0.5,0.5,0.5,0.5})
```

setDynamicModelContact()

```
int arcs::common_interface::ForceControl::setDynamicModelContact (
    const std::vector< double > & env_stiff,
    const std::vector< double > & damp_scale,
    const std::vector< double > & stiff_scale)
```

设置力控接触场景下的动力学模型

参数

<i>env_stiff</i>	表征环境刚度的参数，取值范围[0, 1]，默认为 0
<i>damp_scale</i>	表征阻尼水平的参数，取值范围[0, 1]，默认为 0.5
<i>stiff_scale</i>	表征软硬程度的参数，取值范围[0, 1]，默认为 0.5

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setDynamicModelContact(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: List[float], arg2:←  
List[float]) -> int
```

Lua 函数原型

```
setDynamicModelContact(env_stiff: table, damp_scale: table, stiff_scale: table) -> nil
```

Lua 示例

```
setDynamicModelContact({0,0,0,0,0,0},{0.5,0.5,0.5,0.5,0.5,0.5},{0.5,0.5,0.5,0.5,0.5,0.5})
```

setDynamicModelInsert()

```
int arcs::common_interface::ForceControl::setDynamicModelInsert (  
    const std::vector< double > & damp_scale,  
    const std::vector< double > & stiff_scale)  
设置力控插/拔孔场景下的动力学模型
```

参数

<code>damp_scale</code>	表征阻尼水平的参数, 取值范围[0, 1], 默认为 0.5
<code>stiff_scale</code>	表征软硬程度的参数, 取值范围[0, 1], 默认为 0.5

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setDynamicModelInsert(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: List[float]) -> int
```

Lua 函数原型

```
setDynamicModelInsert(damp_scale: table, stiff_scale: table) -> nil
```

Lua 示例

```
setDynamicModelInsert({0.5,0.5,0.5,0.5,0.5,0.5},{0.5,0.5,0.5,0.5,0.5,0.5})
```

setDynamicModelSearch()

```
int arcs::common_interface::ForceControl::setDynamicModelSearch (
    const std::vector< double > & damp_scale,
    const std::vector< double > & stiff_scale)
```

设置力控搜孔场景下的动力学模型

参数

<i>damp_scale</i>	表征阻尼水平的参数，取值范围[0, 1]，默认为 0.5
<i>stiff_scale</i>	表征软硬程度的参数，取值范围[0, 1]，默认为 0.5

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setDynamicModelSearch(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: List[float]) -> int
```

Lua 函数原型

```
setDynamicModelSearch(damp_scale: table, stiff_scale: table) -> nil
```

Lua 示例

```
setDynamicModelSearch({0.5,0.5,0.5,0.5,0.5,0.5},{0.5,0.5,0.5,0.5,0.5,0.5})
```

setLpFilter()

```
int arcs::common_interface::ForceControl::setLpFilter (
    const std::vector< double > & cutoff_freq)
```

设置低通滤波器

— force frame filter: 过滤测量到的力/力矩 +++ force loop filter: 力控输出参考速度的滤波器

参数

<i>cutoff_freq</i>	截止频率

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setLpFilter(self: pyaubo_sdk.ForceControl, arg0: List[float]) -> int
```

Lua 函数原型

```
setLpFilter(cutoff_freq: table) -> nil
```

setSoftFloatParams()

```
int arcs::common_interface::ForceControl::setSoftFloatParams (
    bool joint_space,
    const std::vector< bool > & select,
    const std::vector< double > & stiff_percent,
    const std::vector< double > & stiff_damp_ratio,
    const std::vector< double > & force_threshold,
    const std::vector< double > & force_limit)
```

设置软浮动参数

参数

<i>joint_softfloat</i>	是否在关节空间启用软浮动
<i>select</i>	选择哪些自由度启用软浮动
<i>stiff_percent</i>	刚度百分比
<i>stiff_damp_ratio</i>	刚度阻尼比
<i>force_threshold</i>	力阈值
<i>force_limit</i>	力限制

返回

返回 0 表示成功，其他为错误码

setSupvForce()

```
int arcs::common_interface::ForceControl::setSupvForce (
    const std::vector< double > & min,
    const std::vector< double > & max)
```

setSupvForce 用于在力控中设置力监督。监督在通过 FCAct 指令激活力控时被激活。

力监督通过在力控坐标系的各个方向上定义最小和最大力限制来设置。一旦激活，如果力超出允许的范围，监督将停止执行。力监督在力控坐标系中指定。该坐标系由用户通过 FCAct 指令设置。

参数

<i>min</i>	最小力限制
<i>max</i>	最大力限制

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
setSupvForce(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: List[float]) -> int
```

Lua 函数原型

```
setSupvForce(min: table, max: table) -> nil
```

Lua 示例

```
setSupvForce({0.1 ,0.1 ,0.1 ,0.1 ,0.1 ,0.1}, {10.0 ,10.0 ,10.0 ,10.0 ,10.0 ,10.0})
```

setSupvOrient()

```
int arcs::common_interface::ForceControl::setSupvOrient (
    const std::vector< double > & frame,
    double max_angle,
    double max_rot,
    bool outside)
```

setSupvOrient 用于设置工具姿态的监督条件。当通过 FCAct 指令激活力控时，监督条件被激活。

姿态监督通过定义相对于参考姿态的最大角度和最大旋转来设置。参考姿态可以由工具当前的 z 方向定义，也可以通过指定相对于工作对象 z 方向的姿态来定义。

一旦激活，工具姿态必须在限制范围内，否则监督将停止执行。

参数

<i>frame</i>	
<i>max_angle</i>	
<i>max_rot</i>	
<i>outside</i>	

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
setSupvOrient(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: float, arg2: float, arg3: bool)
-> int
```

Lua 函数原型

```
setSupvOrient(frame: table, max_angle: number, max_rot: number, outside: boolean) -> nil
```

setSupvPosBox()

```
int arcs::common_interface::ForceControl::setSupvPosBox (
    const std::vector< double > & frame,
    const Box & box)
```

setSupvPosBox 用于在力控中设置位置监督。监督在通过 FCAct 指令激活力控时被激活。位置监督通过为TCP 定义空间体积来设置。一旦激活，如果TCP 超出该体积，监督将停止执行。

参数

<i>frame</i>	
<i>box</i>	

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setSupvPosBox(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: List[float[6]]) -> int
```

Lua 函数原型

```
setSupvPosBox(frame: table, box: table) -> nil
```

setSupvPosCylinder()

```
int arcs::common_interface::ForceControl::setSupvPosCylinder (
    const std::vector< double > & frame,
    const Cylinder & cylinder)
```

参数

<i>frame</i>	
<i>cylinder</i>	

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
setSupvPosCylinder(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: List[float[5]]) -> int
```

Lua 函数原型

```
setSupvPosCylinder(frame: table, cylinder: table) -> nil
```

setSupvPosSphere()

```
int arcs::common_interface::ForceControl::setSupvPosSphere (
    const std::vector< double > & frame,
    const Sphere & sphere)
```

参数

<i>frame</i>	
<i>sphere</i>	

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
setSupvPosSphere(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: List[float[3]]) -> int
```

Lua 函数原型

```
setSupvPosSphere(frame: table, sphere: table) -> nil
```

setSupvReoriSpeed()

```
int arcs::common_interface::ForceControl::setSupvReoriSpeed (
    const std::vector< double > & speed_limit,
    bool outside,
    double timeout)
```

setSupvReoriSpeed 用于在力控中设置重新定向速度监督。监督在通过 FCAct 指令激活力控时被激活。

重新定向速度监督通过定义工作对象坐标系轴周围重新定向速度的最小和最大限制来设置。一旦激活，如果重新定向速度的值过高，监督将停止执行。

有两种速度监督：FCSupvReoriSpeed 和 FCSupvTCPSpeed，后者在第 199 页的 FCSupvTCPSpeed 部分有描述。可能需要两种监督，因为：

- 机器人轴可以在 TCP 静止时高速旋转。
- 当 TCP 距离旋转轴较远时，轴的微小旋转可能导致 TCP 的高速运动。

参数

<i>speed_limit</i>	速度限制
<i>outside</i>	是否在范围外有效
<i>timeout</i>	超时时间

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setSupvReoriSpeed(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: bool, arg2: float) -> int
```

Lua 函数原型

```
setSupvReoriSpeed(speed_limit: table, outside: boolean, timeout: number) -> nil
```

setSupvTcpSpeed()

```
int arcs::common_interface::ForceControl::setSupvTcpSpeed (
    const std::vector< double > & speed_limit,
    bool outside,
    double timeout)
```

setSupvTcpSpeed 用于在力控中设置TCP 速度监督。监督在通过 FCAct 指令激活力控时被激活。TCP 速度监督通过定义工作对象坐标系各方向上的最小和最大速度限制来设置。一旦激活，如果检测到过高的TCP 速度值，监督将停止执行。

有两种速度监督：FCSupvTCPSpeed 和 FCSupvReoriSpeed，后者在第 197 页的 FCSupvReoriSpeed 部分有描述。

可能需要两种监督，因为：

- 机器人轴可以在TCP 静止时高速旋转。
- 当TCP 距离旋转轴较远时，轴的微小旋转可能导致TCP 的高速运动。

参数

<i>speed_limit</i>	速度限制
<i>outside</i>	是否在范围外有效
<i>timeout</i>	超时时间

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setSupvTcpSpeed(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: bool, arg2: float) -> int
```

Lua 函数原型

```
setSupvTcpSpeed(speed_limit: table, outside: boolean, timeout: number) -> nil
```

setTargetForce()

```
int arcs::common_interface::ForceControl::setTargetForce (
    const std::vector< double > & feature,
    const std::vector< bool > & compliance,
    const std::vector< double > & wrench,
    const std::vector< double > & limits,
    TaskFrameType type = TaskFrameType::FRAME_FORCE)
```

设置力控参考 (目标) 值

参数

<i>feature</i>	参考几何特征, 用于生成力控参考坐标系
<i>compliance</i>	柔性轴 (方向) 选择
<i>wrench</i>	目标力/力矩
<i>limits</i>	速度限制
<i>type</i>	力控参考坐标系类型

使用说明:

1. 基坐标系: `feature = {0,0,0,0,0}` `type = TaskFrameType::NONE`
2. 法兰坐标系: `feature = {0,0,0,0,0}` `type = TaskFrameType::TOOL_FORCE`
3. TCP 坐标系: `feature = tcp_offset` `type = TaskFrameType::TOOL_FORCE`
4. 用户坐标系 (FRAME_FORCE): `type = TaskFrameType::FRAME_FORCE` `feature` 设为用户定义的参考坐标, 例如 `getTcpPose()` 表示以当前 TCP 坐标作为力控坐标系。

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setTargetForce(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: List[bool], arg2: List[float],
arg3: List[float], arg4: arcs::common_interface::TaskFrameType) -> int
```

Lua 函数原型

```
setTargetForce(feature: table, compliance: table, wrench: table, limits: table, type: number) -> nil
```

Lua 示例

```
setTargetForce({0,0,0,0,0,0},{true,false,false,false,false,false},{10,0,0,0,0,0},{0,0,0,0,0,0},4)
```

softFloatDisable()

```
int arcs::common_interface::ForceControl::softFloatDisable ()
```

停用软浮动功能。

返回

成功返回 0；失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.ForceControl.softFloatDisable","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":-1}
```

softFloatEnable()

```
int arcs::common_interface::ForceControl::softFloatEnable ()
```

启用软浮动功能。

返回

成功返回 0；失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.ForceControl.softFloatEnable", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

speedChangeDisable()

```
int arcs::common_interface::ForceControl::speedChangeDisable ()
```

停用 FC SpeedChange 功能。

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
speedChangeDisable(self: pyaubo_sdk.ForceControl) -> int
```

Lua 函数原型

```
speedChangeDisable() -> nil
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.ForceControl.speedChangeDisable", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

speedChangeEnable()

```
int arcs::common_interface::ForceControl::speedChangeEnable (
```

```
double ref_force)
```

speedChangeEnable 用于激活 FC SpeedChange 功能，并设置期望的参考力和恢复行为。当 FC SpeedChange 功能被激活时，机器人速度会根据测量信号与参考值的接近程度自动降低或提高。

参数

<i>ref_force</i>	参考力

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
speedChangeEnable(self: pyaubo_sdk.ForceControl, arg0: float) -> int
```

Lua 函数原型

```
speedChangeEnable(ref_force: number) -> nil
```

speedChangeTune()

```
int arcs::common_interface::ForceControl::speedChangeTune (
    int speed_levels,
    double speed_ratio_min)
```

speedChangeTune 用于将 FC SpeedChange 系统参数设置为新值。

参数

<code>speed_levels</code>	速度级别
<code>speed_ratio_min</code>	最小速度比例

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
speedChangeTune(self: pyaubo_sdk.ForceControl, arg0: int, arg1: float) -> int
```

Lua 函数原型

```
speedChangeTune(speed_levels: number, speed_ratio_min: number) -> nil
```

toolContact()

```
int arcs::common_interface::ForceControl::toolContact (
    const std::vector< bool > & direction)
```

检测工具和外部物体的接触

参数

<i>direction</i>	预期的接触方向, 如果所有的元素为 0, 表示检测所有方向的接触

返回

返回从当前点回退到碰撞开始点的周期步数, 如果返回值为 0, 表示没有接触

8.6.4 IoControl (IO 输入输出控制)

IoControl 类提供了一系列的接口对机器人标配的一些数字、模拟IO 进行配置, 输出状态设置、读取 IoControl (IO 输入输出控制) 的协作图:



函数

- int `arcs::common_interface::IoControl::getStandardDigitalInputNum ()`
获取标准数字输入数量
- int `arcs::common_interface::IoControl::getToolDigitalInputNum ()`
获取工具端数字IO 数量 (包括数字输入和数字输出)
- int `arcs::common_interface::IoControl::getConfigurableDigitalInputNum ()`
获取可配置数字输入数量
- int `arcs::common_interface::IoControl::getStandardDigitalOutputNum ()`
获取标准数字输出数量
- int `arcs::common_interface::IoControl::getToolDigitalOutputNum ()`
获取工具端数字IO 数量 (包括数字输入和数字输出)
- int `arcs::common_interface::IoControl::setToolIoInput (int index, bool input)`
设置指定的工具端数字IO 为输入或输出
- bool `arcs::common_interface::IoControl::isToolIoInput (int index)`
判断指定的工具端数字IO 类型是否为输入
- int `arcs::common_interface::IoControl::getConfigurableDigitalOutputNum ()`
获取可配置数字输出数量
- int `arcs::common_interface::IoControl::getStandardAnalogInputNum ()`
获取标准模拟输入数量
- int `arcs::common_interface::IoControl::getToolAnalogInputNum ()`
获取工具端模拟输入数量
- int `arcs::common_interface::IoControl::getStandardAnalogOutputNum ()`
获取标准模拟输出数量
- int `arcs::common_interface::IoControl::getToolAnalogOutputNum ()`
获取工具端模拟输出数量
- int `arcs::common_interface::IoControl::setDigitalInputActionDefault ()`
设置所有数字输入动作为无触发
- int `arcs::common_interface::IoControl::setStandardDigitalInputAction (int index, StandardInputAction action)`
设置标准数字输入触发动作

- `int arcs::common_interface::IoControl::setToolDigitalInputAction (int index, StandardInputAction action)`
设置工具数字输入触发动作
- `int arcs::common_interface::IoControl::setConfigurableDigitalInputAction (int index, StandardInputAction action)`
设置可配置数字输入触发动作
- `StandardInputAction arcs::common_interface::IoControl::getStandardDigitalInputAction (int index)`
获取标准数字输入触发动作
- `StandardInputAction arcs::common_interface::IoControl::getToolDigitalInputAction (int index)`
获取工具端数字输入触发动作
- `StandardInputAction arcs::common_interface::IoControl::getConfigurableDigitalInputAction (int index)`
获取可配置数字输入的输入触发动作
- `int arcs::common_interface::IoControl::setDigitalOutputRunstateDefault ()`
设置所有数字输出状态选择为无
- `int arcs::common_interface::IoControl::setStandardDigitalOutputRunstate (int index, StandardOutputRunState runstate)`
设置标准数字输出状态选择
- `int arcs::common_interface::IoControl::setToolDigitalOutputRunstate (int index, StandardOutputRunState runstate)`
设置工具端数字输出状态选择
- `int arcs::common_interface::IoControl::setConfigurableDigitalOutputRunstate (int index, StandardOutputRunState runstate)`
设置可配置数字输出状态选择
- `int arcs::common_interface::IoControl::setDigitalOutputRunstate (int io_type, int index, StandardOutputRunState runstate, const std::string &object_name, double threshold)`
设置数字输出的状态选择, 并支持基于路点/平面/障碍物的条件阈值判断。
- `StandardOutputRunState arcs::common_interface::IoControl::getStandardDigitalOutputRunstate (int index)`
获取标准数字输出状态选择
- `StandardOutputRunState arcs::common_interface::IoControl::getToolDigitalOutputRunstate (int index)`
获取工具端数字输出状态选择
- `StandardOutputRunState arcs::common_interface::IoControl::getConfigurableDigitalOutputRunstate (int index)`
获取可配置数字输出状态选择
- `StandardOutputRunState arcs::common_interface::IoControl::getDigitalOutputRunstate (int io_type, int index)`
获取指定类型数字输出状态选择。
- `int arcs::common_interface::IoControl::addCrossConnection (const std::string &id, int operator_type, int result_type, int result_pin, const std::vector< int > &actor_types, const std::vector< int > &actor_pins, const std::vector< bool > &actor_inverts)`
新增一条 *DIO Cross Connection* 逻辑规则。
- `int arcs::common_interface::IoControl::deleteCrossConnection (const std::string &id)`
删除一条 *DIO Cross Connection* 逻辑规则。
- `int arcs::common_interface::IoControl::clearCrossConnections ()`
清空所有 *DIO Cross Connection* 逻辑规则。
- `int arcs::common_interface::IoControl::setStandardAnalogOutputRunstate (int index, StandardOutputRunState runstate)`
设置标准模拟输出状态选择
- `int arcs::common_interface::IoControl::setToolAnalogOutputRunstate (int index, StandardOutputRunState runstate)`

- 设置工具端模拟输出状态选择
- `StandardOutputRunState arcs::common_interface::IoControl::getStandardAnalogOutputRunstate (int index)`
获取标准模拟输出状态选择
- `StandardOutputRunState arcs::common_interface::IoControl::getToolAnalogOutputRunstate (int index)`
获取工具端模拟输出状态选择
- `int arcs::common_interface::IoControl::setDigitalOutputAfterEStopDefault ()`
设置所有数字输出急停后状态为默认 (不做改变)
- `int arcs::common_interface::IoControl::setAnalogOutputAfterEStopDefault ()`
设置所有模拟输出急停后状态为默认 (不做改变)
- `int arcs::common_interface::IoControl::setStandardDigitalOutputAfterEStop (int index, bool value)`
设置标准数字输出急停后的输出值
- `int arcs::common_interface::IoControl::setConfigurableDigitalOutputAfterEStop (int index, bool value)`
设置可配置数字输出急停后的输出值
- `int arcs::common_interface::IoControl::setStandardAnalogOutputAfterEStop (int index, double value)`
设置标准模拟输出急停后的输出值
- `int arcs::common_interface::IoControl::setStandardAnalogInputDomain (int index, int domain)`
设置标准模拟输入的范围
- `int arcs::common_interface::IoControl::setToolAnalogInputDomain (int index, int domain)`
设置工具端模拟输入的范围
- `int arcs::common_interface::IoControl::getStandardAnalogInputDomain (int index)`
获取标准模式输入范围
- `int arcs::common_interface::IoControl::getToolAnalogInputDomain (int index)`
获取工具端模式输入范围
- `int arcs::common_interface::IoControl::setStandardAnalogOutputDomain (int index, int domain)`
设置标准模拟输出的范围
- `int arcs::common_interface::IoControl::setToolAnalogOutputDomain (int index, int domain)`
设置工具端模拟输出范围
- `int arcs::common_interface::IoControl::getStandardAnalogOutputDomain (int index)`
获取标准模拟输出范围
- `int arcs::common_interface::IoControl::getToolAnalogOutputDomain (int index)`
获取工具端模拟输出范围
- `int arcs::common_interface::IoControl::setToolVoltageOutputDomain (int domain)`
设置工具端电源电压值 (单位 V)
- `int arcs::common_interface::IoControl::getToolVoltageOutputDomain ()`
获取工具端电源电压值 (单位 V)
- `int arcs::common_interface::IoControl::setStandardDigitalOutput (int index, bool value)`
设置标准数字输出值
- `int arcs::common_interface::IoControl::addGroupOutput (const std::string &name, int io_type, const std::vector< int > &indexes)`
注册组数字输出
- `int arcs::common_interface::IoControl::deleteGroupOutput (const std::string &name)`
删除组数字输出
- `int arcs::common_interface::IoControl::setGroupOutput (const std::string &name, uint32_t value)`
设置组数字输出值
- `int arcs::common_interface::IoControl::setStandardDigitalOutputPulse (int index, bool value, double duration)`
设置数字输出脉冲
- `int arcs::common_interface::IoControl::setToolDigitalOutput (int index, bool value)`

- 设置工具端数字输出值
 - `int arcs::common_interface::IoControl::setToolDigitalOutputPulse` (int index, bool value, double duration)
- 设置工具端数字输出脉冲
 - `int arcs::common_interface::IoControl::setConfigurableDigitalOutput` (int index, bool value)
- 设置可配置数字输出值
 - `int arcs::common_interface::IoControl::setConfigurableDigitalOutputPulse` (int index, bool value, double duration)
- 设置可配置数字输出脉冲
 - `int arcs::common_interface::IoControl::setStandardAnalogOutput` (int index, double value)
- 设置标准模拟输出值
 - `int arcs::common_interface::IoControl::setToolAnalogOutput` (int index, double value)
- 设置工具端模拟输出值
 - `bool arcs::common_interface::IoControl::getStandardDigitalInput` (int index)
- 获取标准数字输入值
 - `uint32_t arcs::common_interface::IoControl::getStandardDigitalInputs` ()
- 获取所有的标准数字输入值
 - `int arcs::common_interface::IoControl::addGroupInput` (const std::string &name, int io_type, const std::vector< int > &indexes)
- 注册组数字输入
 - `int arcs::common_interface::IoControl::deleteGroupInput` (const std::string &name)
- 删除组数字输入
 - `uint32_t arcs::common_interface::IoControl::getGroupInput` (const std::string &name)
- 获取组数字输入值
 - `bool arcs::common_interface::IoControl::getToolDigitalInput` (int index)
- 获取工具端数字输入值
 - `uint32_t arcs::common_interface::IoControl::getToolDigitalInputs` ()
- 获取所有的工具端数字输入值
 - `bool arcs::common_interface::IoControl::getConfigurableDigitalInput` (int index)
- 获取可配置数字输入值
 - `uint32_t arcs::common_interface::IoControl::getConfigurableDigitalInputs` ()
- 获取所有的可配置数字输入值
 - `bool arcs::common_interface::IoControl::getStandardDigitalOutput` (int index)
- 获取标准数字输出值
 - `uint32_t arcs::common_interface::IoControl::getStandardDigitalOutputs` ()
- 获取所有的标准数字输出值
 - `uint32_t arcs::common_interface::IoControl::getGroupOutput` (const std::string &name)
- 获取组数字输出值
 - `bool arcs::common_interface::IoControl::getToolDigitalOutput` (int index)
- 获取工具端数字输出值
 - `uint32_t arcs::common_interface::IoControl::getToolDigitalOutputs` ()
- 获取所有的工具端数字输出值
 - `bool arcs::common_interface::IoControl::getConfigurableDigitalOutput` (int index)
- 获取可配值数字输出值
 - `uint32_t arcs::common_interface::IoControl::getConfigurableDigitalOutputs` ()
- 获取所有的可配值数字输出值
 - `double arcs::common_interface::IoControl::getStandardAnalogInput` (int index)
- 获取标准模拟输入值
 - `double arcs::common_interface::IoControl::getToolAnalogInput` (int index)
- 获取工具端模拟输入值
 - `double arcs::common_interface::IoControl::getStandardAnalogOutput` (int index)

- 获取标准模拟输出值
- `double arcs::common_interface::IoControl::getToolAnalogOutput (int index)`
获取工具端模拟输出值
- `int arcs::common_interface::IoControl::getStaticLinkInputNum ()`
获取联动输入数量
- `int arcs::common_interface::IoControl::getStaticLinkOutputNum ()`
获取联动输出数量
- `uint32_t arcs::common_interface::IoControl::getStaticLinkInputs ()`
获取所有的联动输入值
- `uint32_t arcs::common_interface::IoControl::getStaticLinkOutputs ()`
获取所有的联动输出值
- `bool arcs::common_interface::IoControl::hasEncoderSensor ()`
机器人是否配置了编码器集成编码器的编号为 0
- `int arcs::common_interface::IoControl::setEncDecoderType (int type, int range_id)`
设置集成编码器的解码方式
- `int arcs::common_interface::IoControl::setEncTickCount (int tick)`
设置集成编码器脉冲数
- `int arcs::common_interface::IoControl::getEncDecoderType ()`
获取编码器的解码方式
- `int arcs::common_interface::IoControl::getEncTickCount ()`
获取脉冲数
- `int arcs::common_interface::IoControl::unwindEncDeltaTickCount (int delta_count)`
防止在计数超出范围时计数错误
- `bool arcs::common_interface::IoControl::getToolButtonStatus ()`
获取末端按钮状态
- `uint32_t arcs::common_interface::IoControl::getHandleIoStatus ()`
获取手柄按键状态
- `int arcs::common_interface::IoControl::getHandleType ()`
获取手柄类型

详细描述

IoControl 类提供了一系列的接口对机器人标配的一些数字、模拟IO 进行配置，输出状态设置、读取

1. 获取各种IO 的数量
2. 配置IO 的输入输出功能
3. 可配置IO 的配置
4. 模拟IO 的输入输出范围设置、读取

标准数字输入输出：控制柜IO 面板上的标准IO
 工具端数字输入输出：通过工具末端航插暴露的数字IO
 可配置输入输出：可以配置为安全IO 或者普通数字IO

函数说明

addCrossConnection()

```
int arcs::common_interface::IoControl::addCrossConnection (
    const std::string & id,
    int operator_type,
    int result_type,
    int result_pin,
    const std::vector< int > & actor_types,
    const std::vector< int > & actor_pins,
    const std::vector< bool > & actor_inverts)
```

新增一条 DIO Cross Connection 逻辑规则。

支持 digital/bool 类 DIO 信号和虚拟 DO 信号。规则添加成功后立即生效；DI 只能作为 actor 被读取，不能作为 result 写入。actor_inverts 表示各 actor 读取原始 bool 值后是否取反参与运算，不改变 IO 的硬件方向。

result_type 和 actor_types 使用同一套数字类型。作为 result 时表示输出侧，作为 actor 时表示输入侧或可读的虚拟 DO：

- 0: 标准数字 IO，result 表示标准 DO，actor 表示标准 DI。
- 1: 工具数字 IO，result 表示工具 DO，actor 表示工具 DI。
- 2: 可配置数字 IO，result 表示可配置 DO，actor 表示可配置 DI。
- 3: 虚拟 DO，不占用真实 DO 口，可作为级联中间结果；作为 result 时写入虚拟 DO，作为 actor 时读取虚拟 DO。

operator_type 由控制器实现解释，建议统一为 0=FWD、1=NOT、2=AND、3=OR。FWD/NOT 必须恰好 1 个 actor，AND/OR 必须为 2-5 个 actor。actor_types、actor_pins、actor_inverts 长度必须一致。

限制：

- Cross Connection 规则总数最大 300。
- 规则 id 最大 32 个字符，且不能重复。
- 虚拟 DO 级联深度最大 20。
- 不允许形成闭环依赖。
- 不允许多个规则写同一个 result，包括真实 DO 和虚拟 DO。
- 虚拟 DO actor 必须由前序规则生成，读取未定义虚拟 DO 应返回失败。
- 虚拟 DO pin/index 只要求非负，不占用真实 DO 口。
- 已被 Cross Connection 写入的真实 DO 不应再配置 runstate；后续手动 set/pulse 该 DO 时应拒绝并提示该 DO 已被对应 Cross Connection 占用。

参数

<i>id</i>	规则唯一 ID
<i>operator_type</i>	逻辑运算类型
<i>result_type</i>	结果信号类型，必须是可写输出类类型或虚拟 DO
<i>result_pin</i>	结果信号 pin/index；虚拟 DO 使用虚拟信号 index
<i>actor_types</i>	actor 信号类型列表，可读取 DI 或虚拟 DO
<i>actor_pins</i>	actor 信号 pin/index 列表；虚拟 DO 使用虚拟信号 index
<i>actor_inverts</i>	actor 极性列表

返回

成功返回 0；失败返回错误码

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
addCrossConnection(self: pyaubo_sdk.IoControl, arg0: str, arg1: int, arg2: int, arg3: int, arg4:
List[int], arg5: List[int], arg6: List[bool]) -> int
```

Lua 函数原型

```
addCrossConnection(id: string, operator_type: number, result_type: number, result_pin: num-
ber, actor_types: table, actor_pins: table, actor_inverts: table) -> number
```

Lua 示例

```
addCrossConnection("start_permit", 2, 0, 0, {0, 0}, {3, 4}, {false, true})
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.IoControl.addCrossConnection", "params": ["start_permit", 2, 0, 0, ↵
[0, 0], [3, 4], [false, true]], "id": 1 }
```

JSON-RPC 虚拟 DO 级联示例

```
先生成中间信号 VirtualDO[0] = DI[0] AND NOT DI[1]，再输出 DO[0] = VirtualDO[0] OR DI[2]:
{ "jsonrpc": "2.0", "method": "rob1.IoControl.clearCrossConnections", "params": [], "id": 10 } { "jsonrpc": ↵
"2.0", "method": "rob1.IoControl.addCrossConnection", "params": ["v0_di0_and_not_di1", 2, 3, 0, [0, 0] ↵
, [0, 1], [false, true]], "id": 11 } { "jsonrpc": "2.0", "method": "rob1.IoControl.addCrossConnection", "params": ↵
["v0_or_di2_to_do0", 3, 0, 0, [3, 0], [0, 2], [false, false]], "id": 12 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

addGroupInput()

```
int arcs::common_interface::IoControl::addGroupInput (
    const std::string & name,
    int io_type,
    const std::vector< int > & indexes)
```

注册组数字输入

注册一个按名称访问的数字输入组。indexes[0] 对应 bit0，indexes[1] 对应 bit1，以此类推。

参数

<i>name</i>	组名称。

<i>io_type</i>	IO 类型, 0 表示Standard, 1 表示Tool, 2 表示Configurable。
<i>indexes</i>	组成员对应的数字输入索引列表。

返回

成功返回 0; 失败返回错误码

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
addGroupInput(self: pyaubo_sdk.IoControl, arg0: str, arg1: int, arg2: List[int]) -> int
```

Lua 函数原型

```
addGroupInput(name: string, io_type: number, indexes: table) -> nil
```

Lua 示例

```
addGroupInput("gi_state", 0, {0,1,2,3})
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.IoControl.addGroupInput", "params": ["gi_state", 0, [0,1,2,3]], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

addGroupOutput()

```
int arcs::common_interface::IoControl::addGroupOutput (
    const std::string & name,
    int io_type,
    const std::vector< int > & indexes)
```

注册组数字输出

注册一个按名称访问的数字输出组。indexes[0] 对应 bit0, indexes[1] 对应 bit1, 以此类推。

参数

<i>name</i>	组名称。
<i>io_type</i>	IO 类型, 0 表示Standard, 1 表示Tool, 2 表示Configurable。
<i>indexes</i>	组成员对应的数字输出索引列表。

返回

成功返回 0; 失败返回错误码

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
addGroupOutput(self: pyaubo_sdk.IoControl, arg0: str, arg1: int, arg2: List[int]) -> int
```

Lua 函数原型

```
addGroupOutput(name: string, io_type: number, indexes: table) -> nil
```

Lua 示例

```
addGroupOutput("go_state", 0, {4,5,6,7})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.addGroupOutput","params":["go_state",0,[4,5,6,7]],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

clearCrossConnections()

```
int arcs::common_interface::IoControl::clearCrossConnections ()
```

清空所有 DIO Cross Connection 逻辑规则。

该接口只清空规则，不清空或重置真实数字输出值。

返回

成功返回 0；失败返回错误码

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
clearCrossConnections(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
clearCrossConnections() -> number
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.clearCrossConnections","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

deleteCrossConnection()

```
int arcs::common_interface::IoControl::deleteCrossConnection (
```

参数

<i>id</i>	规则唯一 ID

返回

成功返回 0；失败返回错误码

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
deleteCrossConnection(self: pyaubo_sdk.IoControl, arg0: str) -> int
```

Lua 函数原型

```
deleteCrossConnection(id: string) -> number
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.IoControl.deleteCrossConnection", "params": [ "start_permit", "id": 1 ] }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

deleteGroupInput()

```
int arcs::common_interface::IoControl::deleteGroupInput (
    const std::string & name)
删除组数字输入
```

参数

<i>name</i>	组名称。

返回

成功返回 0；失败返回错误码

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
deleteGroupInput(self: pyaubo_sdk.IoControl, arg0: str) -> int
```

Lua 函数原型

```
deleteGroupInput(name: string) -> nil
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.IoControl.deleteGroupInput", "params": ["gi_state"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

deleteGroupOutput()

```
int arcs::common_interface::IoControl::deleteGroupOutput (
    const std::string & name)
```

删除组数字输出

参数

<i>name</i>	组名称。

返回

成功返回 0；失败返回错误码

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
deleteGroupOutput(self: pyaubo_sdk.IoControl, arg0: str) -> int
```

Lua 函数原型

```
deleteGroupOutput(name: string) -> nil
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.IoControl.deleteGroupOutput", "params": ["go_state"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

getConfigurableDigitalInput()

```
bool arcs::common_interface::IoControl::getConfigurableDigitalInput (
```

```
    int index)
```

获取可配置数字输入值

注解

可用于获取安全IO 的输入值

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

高电平返回 true; 低电平返回 false

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getConfigurableDigitalInput(self: pyaubo_sdk.IoControl, arg0: int) -> bool
```

Lua 函数原型

```
getConfigurableDigitalInput(index: number) -> boolean
```

Lua 示例

```
status = getConfigurableDigitalInput(0)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.IoControl.getConfigurableDigitalInput", "params": [0], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": false}
```

getConfigurableDigitalInputAction()

```
StandardInputAction arcs::common_interface::IoControl::getConfigurableDigitalInputAction (
```

```
    int index)
```

获取可配置数字输入的输入触发动作

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

返回输入触发动作

Python 函数原型

```
getConfigurableDigitalInputAction(self: pyaubo_sdk.IoControl, arg0: int) -> arcs::common\_interface::StandardInp
```

Lua 函数原型

```
getConfigurableDigitalInputAction(index: number) -> number
```

Lua 示例

```
num = getConfigurableDigitalInputAction(0)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.IoControl.getConfigurableDigitalInputAction", "params": [0], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": "Default"}
```

getConfigurableDigitalInputNum()

```
int arcs::common_interface::IoControl::getConfigurableDigitalInputNum ()  
获取可配置数字输入数量
```

返回

可配置数字输入数量

异常

arcs::common_interface::AuboException

Python 函数原型

```
getConfigurableDigitalInputNum(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getConfigurableDigitalInputNum\(\) -> number
```

Lua 示例

```
num = getConfigurableDigitalInputNum\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.IoControl.getConfigurableDigitalInputNum", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 16}
```


getConfigurableDigitalInputs()

```
uint32_t arcs::common_interface::IoControl::getConfigurableDigitalInputs ()
```

获取所有的可配置数字输入值

注解

可用于获取安全IO 的输入值

返回

所有的可配置数字输入值

例如, 当返回值是 2863267846 时, 换成 2 进制后是 101010101010101000000000000000110。后 16 位就是所有的输入状态值, 最后一位表示管脚 0 的输入状态值, 倒数第二位表示管脚 1 的输入状态值, 以此类推。

1 表示高电平状态, 0 表示低电平状态

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getConfigurableDigitalInputs(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getConfigurableDigitalInputs() -> number
```

Lua 示例

```
num = getConfigurableDigitalInputs()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getConfigurableDigitalInputs","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getConfigurableDigitalOutput()

```
bool arcs::common_interface::IoControl::getConfigurableDigitalOutput (
    int index)
获取可配值数字输出值
```

注解

可用于获取安全IO 的输出值

参数

<i>index</i>	表示IO 口的管脚, 管脚编号从 0 开始。例如, 0 表示第一个管脚。

返回

高电平返回 true; 低电平返回 false

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getConfigurableDigitalOutput(self: pyaubo_sdk.IoControl, arg0: int) -> bool
```

Lua 函数原型

```
getConfigurableDigitalOutput(index: number) -> boolean
```

Lua 示例

```
status = getConfigurableDigitalOutput(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getConfigurableDigitalOutput","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":true}
```

getConfigurableDigitalOutputNum()

```
int arcs::common_interface::IoControl::getConfigurableDigitalOutputNum ()
获取可配置数字输出数量
```

返回

可配置数字输出数量

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getConfigurableDigitalOutputNum(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getConfigurableDigitalOutputNum() -> number
```

Lua 示例

```
num = getConfigurableDigitalOutputNum()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getConfigurableDigitalOutputNum","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":16}
```

getConfigurableDigitalOutputRunstate()

```
StandardOutputRunState arcs::common_interface::IoControl::getConfigurableDigitalOutputRunstate (
    int index)
```

获取可配置数字输出状态选择

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

输出状态选择

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getConfigurableDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int) -> arcs::common_interface::StandardOutputRunState
```

Lua 函数原型

```
getConfigurableDigitalOutputRunstate(index: number) -> number
```

Lua 示例

```
num = getConfigurableDigitalOutputRunstate(0)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.IoControl.getConfigurableDigitalOutputRunstate", "params": [0], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": "None"}
```

getConfigurableDigitalOutputs()

```
uint32_t arcs::common_interface::IoControl::getConfigurableDigitalOutputs ()
```

获取所有的可配值数字输出值

注解

可用于获取安全IO 的输出值

返回

所有的可配值数字输出

例如，当返回值是 2863267846 时，换成 2 进制后是 101010101010100000000000000110。后 16 位就是所有的输出值，最后一位表示管脚 0 的输出状态值，倒数第二位表示管脚 1 的输出状态值，以此类推。

1 表示高电平状态，0 表示低电平状态。

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getConfigurableDigitalOutputs(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getConfigurableDigitalOutputs() -> number
```

Lua 示例

```
num = getConfigurableDigitalOutputs()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getConfigurableDigitalOutputs","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":1}
```

getDigitalOutputRunstate()

```
StandardOutputRunState arcs::common_interface::IoControl::getDigitalOutputRunstate (
    int io_type,
    int index)
```

获取指定类型数字输出状态选择。

用于查询不同类型的数字输出（标准数字输出、工具端数字输出、可配置数字输出）的当前运行状态选择（例如 WaypointArrived 等），一般跟 setDigitalOutputRunstate 一起使用。

参数

<i>io_type</i>	数字输出类型（0= 标准数字输出、1= 工具数字输出、2= 可配置数字输出）
<i>index</i>	输出口的索引，管脚编号从 0 开始

返回

返回对应的 `StandardOutputRunState` 枚举值

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int, arg1: int) -> arcs::common_interface::StandardO
```

Lua 函数原型

```
getDigitalOutputRunstate(io_type: number, index: number) -> number
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getDigitalOutputRunstate","params":[1,0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":"None"}
```

getEncDecoderType()

```
int arcs::common_interface::IoControl::getEncDecoderType ()
```

获取编码器的解码方式

返回

成功返回 0；失败返回错误码 AUBO_NO_ACCESS AUBO_BUSY AUBO_BAD_STATE - AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getEncDecoderType","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getEncTickCount()

```
int arcs::common_interface::IoControl::getEncTickCount ()
```

获取脉冲数

返回

成功返回 0；失败返回错误码 AUBO_NO_ACCESS AUBO_BUSY AUBO_BAD_STATE - AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getEncTickCount","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getGroupInput()

```
uint32_t arcs::common_interface::IoControl::getGroupInput (  
    const std::string & name)
```

获取组数字输入值

按名称查找组数字输入后，读取对应组成员并按位拼装返回整数值。

参数

<i>name</i>	组名称。

返回

返回组数字输入的组合值。

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getGroupInput(self: pyaubo_sdk.IoControl, arg0: str) -> int
```

Lua 函数原型

```
getGroupInput(name: string) -> number
```

Lua 示例

```
value = getGroupInput("gi_state")
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.IoControl.getGroupInput", "params": ["gi_state"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 13}
```

getGroupOutput()

```
uint32_t arcs::common_interface::IoControl::getGroupOutput (  
    const std::string & name)
```

获取组数字输出值

按名称查找组数字输出后，读取对应组成员并按位拼装返回整数值。

参数

<i>name</i>	组名称。

返回

返回组数字输出的组合值。

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getGroupOutput(self: pyaubo_sdk.IoControl, arg0: str) -> int
```

Lua 函数原型

```
getGroupOutput(name: string) -> number
```

Lua 示例

```
value = getGroupOutput("go_state")
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getGroupOutput","params":["go_state"],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":13}
```

getHandleIoStatus()

```
uint32_t arcs::common_interface::IoControl::getHandleIoStatus ()
```

获取手柄按键状态

注解

获取手柄按键状态

返回

所有的手柄按键输入值

例如, 当返回值是 2863267846 时, 换成 2 进制后是 1010101010101000000000000000110。后 16 位就是所有的输入状态值, 最后一位表示管脚 0 的输入状态值, 倒数第二位表示管脚 1 的输入状态值, 以此类推。

1 表示高电平状态, 0 表示低电平状态

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getHandleIoStatus(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getHandleIoStatus() -> number
```

Lua 示例

```
num = getHandleIoStatus()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getHandleIoStatus","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getHandleType()

```
int arcs::common_interface::IoControl::getHandleType ()  
获取手柄类型
```

返回

```
type
```

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getHandleType() -> int
```

Lua 函数原型

```
getHandleType() -> int
```

Lua 示例

```
int_num = getHandleType()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getHandleType","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getStandardAnalogInput()

```
double arcs::common_interface::IoControl::getStandardAnalogInput (  
    int index)  
获取标准模拟输入值
```


参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

标准模拟输入值

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getStandardAnalogInput(self: pyaubo_sdk.IoControl, arg0: int) -> float
```

Lua 函数原型

```
getStandardAnalogInput(index: number) -> number
```

Lua 示例

```
getStandardAnalogInput(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getStandardAnalogInput","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0.0}
```

getStandardAnalogInputDomain()

```
int arcs::common_interface::IoControl::getStandardAnalogInputDomain (
    int index)
获取标准模式输入范围
```

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

标准模式输入范围

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getStandardAnalogInputDomain(self: pyaubo_sdk.IoControl, arg0: int) -> int
```

Lua 函数原型

```
getStandardAnalogInputDomain(index: number) -> number
```

Lua 示例

```
num = getStandardAnalogInputDomain(0)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardAnalogInputDomain", "params": [0], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

getStandardAnalogInputNum()

```
int arcs::common_interface::IoControl::getStandardAnalogInputNum ()
```

获取标准模拟输入数量

返回

标准模拟输入数量

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getStandardAnalogInputNum(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getStandardAnalogInputNum() -> number
```

Lua 示例

```
num = getStandardAnalogInputNum()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardAnalogInputNum", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 2}
```

getStandardAnalogOutput()

```
double arcs::common_interface::IoControl::getStandardAnalogOutput (
    int index)
    获取标准模拟输出值
```

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

标准模拟输出值

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getStandardAnalogOutput(self: pyaubo_sdk.IoControl, arg0: int) -> float
```

Lua 函数原型

```
getStandardAnalogOutput(index: number) -> number
```

Lua 示例

```
num = getStandardAnalogOutput(0)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardAnalogOutput", "params": [0], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0.0}
```

getStandardAnalogOutputDomain()

```
int arcs::common_interface::IoControl::getStandardAnalogOutputDomain (
    int index)
    获取标准模拟输出范围
```

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

标准模拟输出范围

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getStandardAnalogOutputDomain(self: pyaubo_sdk.IoControl, arg0: int) -> int
```

Lua 函数原型

```
getStandardAnalogOutputDomain(index: number) -> number
```

Lua 示例

```
num = getStandardAnalogOutputDomain(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getStandardAnalogOutputDomain","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getStandardAnalogOutputNum()

```
int arcs::common_interface::IoControl::getStandardAnalogOutputNum ()  
获取标准模拟输出数量
```

返回

标准模拟输出数量

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getStandardAnalogOutputNum(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getStandardAnalogOutputNum() -> number
```

Lua 示例

```
num = getStandardAnalogOutputNum()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getStandardAnalogOutputNum","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":2}
```

getStandardAnalogOutputRunstate()

```
StandardOutputRunState arcs::common_interface::IoControl::getStandardAnalogOutputRunstate (
    int index)
```

获取标准模拟输出状态选择

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

标准模拟输出状态选择

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getStandardAnalogOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int) -> arcs::common_interface::StandardOutputRunState
```

Lua 函数原型

```
getStandardAnalogOutputRunstate(index: number) -> number
```

Lua 示例

```
num = getStandardAnalogOutputRunstate(0)
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.IoControl.getStandardAnalogOutputRunstate", "params": [0], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": "None" }
```

getStandardDigitalInput()

```
bool arcs::common_interface::IoControl::getStandardDigitalInput (
    int index)
```

获取标准数字输入值

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

高电平返回 true; 低电平返回 false

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getStandardDigitalInput(self: pyaubo_sdk.IoControl, arg0: int) -> bool
```

Lua 函数原型

```
getStandardDigitalInput(index: number) -> boolean
```

Lua 示例

```
status = getStandardDigitalInput(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getStandardDigitalInput","params":[0,"id":1]}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":false}
```

getStandardDigitalInputAction()

```
StandardInputAction arcs::common_interface::IoControl::getStandardDigitalInputAction (
    int index)
```

获取标准数字输入触发动作

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

标准数字输入触发动作

Python 函数原型

```
getStandardDigitalInputAction(self: pyaubo_sdk.IoControl, arg0: int) -> arcs::common_interface::StandardInputA
```

Lua 函数原型

```
getStandardDigitalInputAction(index: number) -> number
```

Lua 示例

```
num = getStandardDigitalInputAction(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getStandardDigitalInputAction","params":[0,"id":1]}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":"Default"}
```

getStandardDigitalInputNum()

```
int arcs::common_interface::IoControl::getStandardDigitalInputNum ()
```

获取标准数字输入数量

返回

标准数字输入数量

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getStandardDigitalInputNum(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getStandardDigitalInputNum() -> number
```

Lua 示例

```
num = getStandardDigitalInputNum()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getStandardDigitalInputNum","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":16}
```

getStandardDigitalInputs()

```
uint32_t arcs::common_interface::IoControl::getStandardDigitalInputs ()
```

获取所有的标准数字输入值

返回

所有的标准数字输入值

例如, 当返回值是 2863267846 时, 换成 2 进制后是 101010101010100000000000000110。后 16 位就是所有的标准数字输入状态值, 最后一位表示DI00 的输入状态值, 倒数第二位表示DI01 的输入状态值, 以此类推。

1 表示高电平状态, 0 表示低电平状态

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getStandardDigitalInputs(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getStandardDigitalInputs() -> number
```

Lua 示例

```
num = getStandardDigitalInputs()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getStandardDigitalInputs","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getStandardDigitalOutput()

```
bool arcs::common_interface::IoControl::getStandardDigitalOutput (
    int index)
获取标准数字输出值
```

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

高电平返回 true; 低电平返回 false

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getStandardDigitalOutput(self: pyaubo_sdk.IoControl, arg0: int) -> bool
```

Lua 函数原型

```
getStandardDigitalOutput(index: number) -> boolean
```

Lua 示例

```
status = getStandardDigitalOutput(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getStandardDigitalOutput","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":true}
```


getStandardDigitalOutputNum()

```
int arcs::common_interface::IoControl::getStandardDigitalOutputNum (
    获取标准数字输出数量
```

返回

标准数字输出数量

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getStandardDigitalOutputNum(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getStandardDigitalOutputNum() -> number
```

Lua 示例

```
num = getStandardDigitalOutputNum()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getStandardDigitalOutputNum","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":8}
```

getStandardDigitalOutputRunstate()

```
StandardOutputRunState arcs::common_interface::IoControl::getStandardDigitalOutputRunstate (
    int index)
    获取标准数字输出状态选择
```

参数

<code>index</code>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

输出状态选择

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getStandardDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int) -> arcs::common_interface::StandardOutputRunstate
```

Lua 函数原型

```
getStandardDigitalOutputRunstate(index: number) -> number
```

Lua 示例

```
num = getStandardDigitalOutputRunstate(0)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardDigitalOutputRunstate", "params": [0, "id": 1]}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": "None"}
```

getStandardDigitalOutputs()

```
uint32_t arcs::common_interface::IoControl::getStandardDigitalOutputs ()
```

获取所有的标准数字输出值

返回

所有的标准数字输出值

例如，当返回值是 2863267846 时，换成 2 进制后是 10101010101010100000000000000110。后 16 位就是所有的标准数字输出状态值，最后一位表示DI00 的输出状态值，倒数第二位表示DI01 的输出状态值，以此类推。

1 表示高电平状态，0 表示低电平状态。

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getStandardDigitalOutputs(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getStandardDigitalOutputs() -> number
```

Lua 示例

```
num = getStandardDigitalOutputs()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardDigitalOutputs", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 69}
```

getStaticLinkInputNum()

```
int arcs::common_interface::IoControl::getStaticLinkInputNum ()
```

获取联动输入数量

返回

联动输入数量

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getStaticLinkInputNum(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getStaticLinkInputNum() -> number
```

Lua 示例

```
num = getStaticLinkInputNum()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getStaticLinkInputNum","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":8}
```

getStaticLinkInputs()

```
uint32_t arcs::common_interface::IoControl::getStaticLinkInputs ()
```

获取所有的联动输入值

返回

所有的联动输入值

例如, 当返回值是 2863267846 时, 换成 2 进制后是 101010101010100000000000000110。后 16 位就是所有的联动输入状态值, 最后一位表示管脚 0 的输入状态值, 倒数第二位表示管脚 1 的输入状态值, 以此类推。

1 表示高电平状态, 0 表示低电平状态。

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getStaticLinkInputs(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getStaticLinkInputs() -> number
```

Lua 示例

```
num = getStaticLinkInputs()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.IoControl.getStaticLinkInputs", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

getStaticLinkOutputNum()

```
int arcs::common_interface::IoControl::getStaticLinkOutputNum ()
获取联动输出数量
```

返回

联动输出数量

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getStaticLinkOutputNum(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getStaticLinkOutputNum() -> number
```

Lua 示例

```
num = getStaticLinkOutputNum()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.IoControl.getStaticLinkOutputNum", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

getStaticLinkOutputs()

```
uint32_t arcs::common_interface::IoControl::getStaticLinkOutputs ()
获取所有的联动输出值
```

返回

返回所有的联动输出值

例如，当返回值是 2863267846 时，换成 2 进制后是 101010101010101000000000000000110。后 16 位就是所有的联动输出状态值，最后一位表示管脚 0 的输出状态值，倒数第二位表示管脚 1 的输出状态值，以此类推。

1 表示高电平状态，0 表示低电平状态。

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getStaticLinkOutputs(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getStaticLinkOutputs() -> number
```

Lua 示例

```
num = getStaticLinkOutputs()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getStaticLinkOutputs","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getToolAnalogInput()

```
double arcs::common_interface::IoControl::getToolAnalogInput (
    int index)
获取工具端模拟输入值
```

参数

<code>index</code>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

工具端模拟输入值

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getToolAnalogInput(self: pyaubo_sdk.IoControl, arg0: int) -> float
```

Lua 函数原型

```
getToolAnalogInput(index: number) -> number
```

Lua 示例

```
num = getToolAnalogInput(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getToolAnalogInput","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0.0}
```

getToolAnalogInputDomain()

```
int arcs::common_interface::IoControl::getToolAnalogInputDomain (
    int index)
获取工具端模式输入范围
```

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

工具端模式输入范围

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getToolAnalogInputDomain(self: pyaubo_sdk.IoControl, arg0: int) -> int
```

Lua 函数原型

```
getToolAnalogInputDomain(index: number) -> number
```

Lua 示例

```
num = getToolAnalogInputDomain(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getToolAnalogInputDomain","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":10}
```

getToolAnalogInputNum()

```
int arcs::common_interface::IoControl::getToolAnalogInputNum ()
获取工具端模拟输入数量
```

返回

工具端模拟输入数量

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getToolAnalogInputNum(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getToolAnalogInputNum() -> number
```

Lua 示例

```
num = getToolAnalogInputNum()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getToolAnalogInputNum","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":2}
```

getToolAnalogOutput()

```
double arcs::common_interface::IoControl::getToolAnalogOutput (
    int index)
获取工具端模拟输出值
```

参数

<code>index</code>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

工具端模拟输出值

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getToolAnalogOutput(self: pyaubo_sdk.IoControl, arg0: int) -> float
```

Lua 函数原型

```
getToolAnalogOutput(index: number) -> number
```

Lua 示例

```
num = getToolAnalogOutput(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getToolAnalogOutput","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0.0}
```

getToolAnalogOutputDomain()

```
int arcs::common_interface::IoControl::getToolAnalogOutputDomain (
    int index)
获取工具端模拟输出范围
```

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

工具端模拟输出范围

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getToolAnalogOutputDomain(self: pyaubo_sdk.IoControl, arg0: int) -> int
```

Lua 函数原型

```
getToolAnalogOutputDomain(index: number) -> number
```

Lua 示例

```
num = getToolAnalogOutputDomain(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getToolAnalogOutputDomain","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getToolAnalogOutputNum()

```
int arcs::common_interface::IoControl::getToolAnalogOutputNum ()
获取工具端模拟输出数量
```

返回

工具端模拟输出数量

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getToolAnalogOutputNum(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getToolAnalogOutputNum() -> number
```

Lua 示例

```
num = getToolAnalogOutputNum()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getToolAnalogOutputNum","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getToolAnalogOutputRunstate()

```
StandardOutputRunState arcs::common_interface::IoControl::getToolAnalogOutputRunstate (
    int index)
```

获取工具端模拟输出状态选择

参数

<code>index</code>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

工具端模拟输出状态选择

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getToolAnalogOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int) -> arcs::common_interface::StandardOutputRunState
```

Lua 函数原型

```
getToolAnalogOutputRunstate(index: number) -> number
```

Lua 示例

```
num = getToolAnalogOutputRunstate(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getToolAnalogOutputRunstate","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result": "None"}
```

getToolButtonStatus()

```
bool arcs::common_interface::IoControl::getToolButtonStatus ()  
获取末端按钮状态
```

返回

按下返回 true; 否则返回 false

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
getToolButtonStatus() -> bool
```

Lua 函数原型

```
getToolButtonStatus() -> boolean
```

Lua 示例

```
status = getToolButtonStatus()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getToolButtonStatus","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":false}
```

getToolDigitalInput()

```
bool arcs::common_interface::IoControl::getToolDigitalInput (  
    int index)  
获取工具端数字输入值
```

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

高电平返回 true; 低电平返回 false

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getToolDigitalInput(self: pyaubo_sdk.IoControl, arg0: int) -> bool
```

Lua 函数原型

```
getToolDigitalInput(index: number) -> boolean
```

Lua 示例

```
status = getToolDigitalInput(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getToolDigitalInput","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":false}
```

getToolDigitalInputAction()

```
StandardInputAction arcs::common_interface::IoControl::getToolDigitalInputAction (
    int index)
```

获取工具端数字输入触发动作

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

工具端数字输入触发动作

Python 函数原型

```
getToolDigitalInputAction(self: pyaubo_sdk.IoControl, arg0: int) -> arcs::common\_interface::StandardInputAction
```

Lua 函数原型

```
getToolDigitalInputAction(index: number) -> number
```

Lua 示例

```
getToolDigitalInputAction(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getToolDigitalInputAction","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":"Default"}
```

getToolDigitalInputNum()

```
int arcs::common_interface::IoControl::getToolDigitalInputNum ()  
获取工具端数字IO 数量 (包括数字输入和数字输出)
```

返回

工具端数字IO 数量 (包括数字输入和数字输出)

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getToolDigitalInputNum(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getToolDigitalInputNum() -> number
```

Lua 示例

```
num = getStandardDigitalInputNum()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getToolDigitalInputNum","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":4}
```

getToolDigitalInputs()

```
uint32_t arcs::common_interface::IoControl::getToolDigitalInputs ()  
获取所有的工具端数字输入值
```

返回

返回所有的工具端数字输入值

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getToolDigitalInputs(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getToolDigitalInputs() -> number
```

Lua 示例

```
num = getToolDigitalInputs()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getToolDigitalInputs","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getToolDigitalOutput()

```
bool arcs::common_interface::IoControl::getToolDigitalOutput (
    int index)
获取工具端数字输出值
```

参数

<code>index</code>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

高电平返回 true; 低电平返回 false

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getToolDigitalOutput(self: pyaubo_sdk.IoControl, arg0: int) -> bool
```

Lua 函数原型

```
getToolDigitalOutput(index: number) -> boolean
```

Lua 示例

```
status = getToolDigitalOutput(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getToolDigitalOutput","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":false}
```

getToolDigitalOutputNum()

```
int arcs::common_interface::IoControl::getToolDigitalOutputNum ()  
获取工具端数字IO 数量 (包括数字输入和数字输出)
```

返回

工具端数字IO 数量 (包括数字输入和数字输出)

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getToolDigitalOutputNum(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getToolDigitalOutputNum() -> number
```

Lua 示例

```
num = getToolDigitalOutputNum()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getToolDigitalOutputNum","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":4}
```

getToolDigitalOutputRunstate()

```
StandardOutputRunState arcs::common_interface::IoControl::getToolDigitalOutputRunstate (  
    int index)  
获取工具端数字输出状态选择
```

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

输出状态选择

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getToolDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int) -> arcs::common_interface::StandardOutput
```

Lua 函数原型

```
getToolDigitalOutputRunstate(index: number) -> number
```

Lua 示例

```
num = getToolDigitalOutputRunstate(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getToolDigitalOutputRunstate","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result": "None"}
```

getToolDigitalOutputs()

```
uint32_t arcs::common_interface::IoControl::getToolDigitalOutputs ()  
获取所有的工具端数字输出值
```

返回

所有的工具端数字输出值

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getToolDigitalOutputs(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getToolDigitalOutputs() -> number
```

Lua 示例

```
num = getToolDigitalOutputs()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getToolDigitalOutputs","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":9}
```

getToolVoltageOutputDomain()

```
int arcs::common_interface::IoControl::getToolVoltageOutputDomain ()  
获取工具端电源电压值 (单位 V)
```

返回

工具端电源电压值 (单位 V)

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getToolVoltageOutputDomain(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
getToolVoltageOutputDomain() -> number
```

Lua 示例

```
num = getToolVoltageOutputDomain()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.getToolVoltageOutputDomain","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

hasEncoderSensor()

```
bool arcs::common_interface::IoControl::hasEncoderSensor ()  
机器人是否配置了编码器集成编码器的编号为 0
```

返回

机器人配置编码器返回 true, 反之返回 false

异常

<code>arcs::common_interface::AuboException</code>

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.hasEncoderSensor","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":true}
```

isToolIoInput()

```
bool arcs::common_interface::IoControl::isToolIoInput (
    int index)
```

判断指定的工具端数字IO 类型是否为输入

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。

返回

当指定的IO 为输入时返回 true, 否则为 false

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
isToolIoInput(self: pyaubo_sdk.IoControl, arg0: int) -> bool
```

Lua 函数原型

```
isToolIoInput(index: number) -> boolean
```

Lua 示例

```
status = isToolIoInput(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.isToolIoInput","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":true}
```

setAnalogOutputAfterEStopDefault()

```
int arcs::common_interface::IoControl::setAnalogOutputAfterEStopDefault ()
```

设置所有模拟输出急停后状态为默认（不做改变）

返回

成功返回 0；失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setAnalogOutputAfterEStopDefault(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
setAnalogOutputAfterEStopDefault() -> nil
```

Lua 示例

```
setAnalogOutputAfterEStopDefault()
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.IoControl.setAnalogOutputAfterEStopDefault", "params": [], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

setConfigurableDigitalInputAction()

```
int arcs::common_interface::IoControl::setConfigurableDigitalInputAction (
    int index,
    StandardInputAction action)
```

设置可配置数字输入触发动作

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。
<i>action</i>	触发动作

返回

成功返回 0；失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

注解

需要将可配置输入的安全输入动作设置为 `SafetyInputAction::Unassigned` 时这个函数的配置才会生效

Python 函数原型

```
setConfigurableDigitalInputAction(self: pyaubo_sdk.IoControl, arg0: int, arg1: arcs::common_interface::StandardI
-> int
```

Lua 函数原型

```
setConfigurableDigitalInputAction(index: number, action: number) -> nil
```

Lua 示例

```
setConfigurableDigitalInputAction(0,1)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.setConfigurableDigitalInputAction","params":[0,"Handguide"]↵
,"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setConfigurableDigitalOutput()

```
int arcs::common_interface::IoControl::setConfigurableDigitalOutput (
    int index,
    bool value)
```

设置可配置数字输出值

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。
<i>value</i>	数字输出值

返回

成功返回 0; 失败返回错误码 `AUBO_REQUEST_IGNORE` `AUBO_BUSY` `AUBO_BAD_STATE`
`-AUBO_INVL_ARGUMENT` `-AUBO_BAD_STATE`

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setConfigurableDigitalOutput(self: pyaubo_sdk.IoControl, arg0: int, arg1: bool) -> int
```

Lua 函数原型

```
setConfigurableDigitalOutput(index: number, value: boolean) -> nil
```

Lua 示例

```
setConfigurableDigitalOutput(0,true)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.IoControl.setConfigurableDigitalOutput", "params": [0, true], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setConfigurableDigitalOutputAfterEStop()

```
int arcs::common_interface::IoControl::setConfigurableDigitalOutputAfterEStop (
    int index,
    bool value)
```

设置可配置数字输出急停后的输出值

参数

<i>index</i>	表示IO 口的管脚,
<i>value</i>	输出值

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setConfigurableDigitalOutputAfterEStop(self: pyaubo_sdk.IoControl, arg0: int, arg1: bool) -> int
```

Lua 函数原型

```
setConfigurableDigitalOutputAfterEStop(index: number, value: boolean) -> nil
```

Lua 示例

```
setConfigurableDigitalOutputAfterEStop(0,true)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.setConfigurableDigitalOutputAfterEStop","params":[0,true]←  
,"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setConfigurableDigitalOutputPulse()

```
int arcs::common_interface::IoControl::setConfigurableDigitalOutputPulse (  
    int index,  
    bool value,  
    double duration)
```

设置可配置数字输出脉冲

参数

<i>index</i>	
<i>value</i>	
<i>duration</i>	

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setConfigurableDigitalOutputPulse(self: pyaubo_sdk.IoControl, arg0: int, arg1: bool, arg2: float)  
-> int
```

Lua 函数原型

```
setConfigurableDigitalOutputPulse(index: number, value: boolean, duration: number) -> nil
```

Lua 示例

```
setConfigurableDigitalOutputPulse(0,true,2.3)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.setConfigurableDigitalOutputPulse","params":[0,true,0.←  
5],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setConfigurableDigitalOutputRunstate()

```
int arcs::common_interface::IoControl::setConfigurableDigitalOutputRunstate (
    int index,
    StandardOutputRunState runstate)
```

设置可配置数字输出状态选择

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。
<i>runstate</i>	输出状态选择

返回

成功返回 0；失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setConfigurableDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int, arg1: arcs::common_interface::Stand  
-> int
```

Lua 函数原型

```
setConfigurableDigitalOutputRunstate(index: number, runstate: number) -> nil
```

Lua 示例

```
setConfigurableDigitalOutputRunstate(0,1)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.IoControl.setConfigurableDigitalOutputRunstate", "params": [0, "↵  
None"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setDigitalInputActionDefault()

```
int arcs::common_interface::IoControl::setDigitalInputActionDefault ()
```

设置所有数字输入动作为无触发

注解

当输入动作为无触发时，用户设置数字输入值为高电平，不会触发机器人发生动作

返回

成功返回 0；失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setDigitalInputActionDefault(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
setDigitalInputActionDefault() -> nil
```

Lua 示例

```
setDigitalInputActionDefault()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.setDigitalInputActionDefault","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setDigitalOutputAfterEStopDefault()

```
int arcs::common_interface::IoControl::setDigitalOutputAfterEStopDefault ()
设置所有数字输出急停后状态为默认（不做改变）
```

返回

成功返回 0；失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setDigitalOutputAfterEStopDefault(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
setDigitalOutputAfterEStopDefault() -> nil
```

Lua 示例

```
setDigitalOutputAfterEStopDefault()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.setDigitalOutputAfterEStopDefault","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setDigitalOutputRunstate()

```
int arcs::common_interface::IoControl::setDigitalOutputRunstate (
    int io_type,
    int index,
    StandardOutputRunState runstate,
    const std::string & object_name,
    double threshold)
```

设置数字输出的状态选择，并支持基于路点/平面/障碍物的条件阈值判断。

详细说明：当需要根据机器人是否到达某个路点或与某个平面/障碍物的距离来触发数字输出时，可通过本接口指定条件对象及阈值。在条件满足时会将指定的数字输出设置为对应的运行状态。

参数

<i>io_type</i>	数字输出类型（0= 标准数字输出、1= 工具数字输出、2= 可配置数字输出），用整数表示
<i>index</i>	输出口的索引，管脚编号从 0 开始
<i>runstate</i>	输出动作状态
<i>object_name</i>	条件对象的名称
<i>threshold</i>	判断阈值：对路点为角度阈值，单位为弧度 (rad)，对平面/障碍物为距离阈值，单位为米 (m)

返回

返回操作结果码：0 表示成功，非 0 表示失败

Python 函数原型

```
setDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int, arg1: int, arg2: arcs::common_interface::StandardOutputRunState, arg3: str="", arg4: float=0.0) -> int
```

Lua 函数原型

```
setDigitalOutputRunstate(io_type: number, index: number, runstate: number, object_name:↵
string="", threshold: number=0.0) -> nil
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.setDigitalOutputRunstate","params":[1,0,"WaypointArrived"],"↵
P1",3],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setDigitalOutputRunstateDefault()

```
int arcs::common_interface::IoControl::setDigitalOutputRunstateDefault ()
设置所有数字输出状态选择为无
```

注解

当给输出状态设置为无 (StandardOutputRunState::None) 时，用户可以设置数字输出值。

当给输出设置状态时，用户不可设置数字输出值，控制器会自动设置数字输出值。

例如，当设置DO0 的输出状态为高电平指示正在拖动示教 (StandardOutputRunState::Handguiding) 时，机器人进入拖动示教，DO0 会自动变为高电平。机器人退出拖动示教，DO0 会自动变为低电平。

返回

成功返回 0；失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setDigitalOutputRunstateDefault(self: pyaubo_sdk.IoControl) -> int
```

Lua 函数原型

```
setDigitalOutputRunstateDefault() -> nil
```

Lua 示例

```
setDigitalOutputRunstateDefault()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.setDigitalOutputRunstateDefault","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setEncDecoderType()

```
int arcs::common_interface::IoControl::setEncDecoderType (
    int type,
    int range_id)
设置集成编码器的解码方式
```

参数

<i>type</i>	0-禁用编码器 1-AB 正交 2-AB 正交 +Z 3-AB 差分正交 4-AB 差分正交 +Z 差分
<i>range_id</i>	0 表示 32 位有符号编码器，范围为 [-2147483648, 2147483647] 1 表示 8 位无符号编码器，范围为 [0, 255] 2 表示 16 位无符号编码器，范围为 [0, 65535] 3 表示 24 位无符号编码器，范围为 [0, 16777215] 4 表示 32 位无符号编码器，范围为 [0, 4294967295]

返回

成功返回 0；失败返回错误码 AUBO_NO_ACCESS AUBO_BUSY AUBO_BAD_STATE - AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

setEncTickCount()

```
int arcs::common_interface::IoControl::setEncTickCount (  
    int tick)
```

设置集成编码器脉冲数

参数

<i>tick</i>	脉冲数

返回

成功返回 0；失败返回错误码 AUBO_NO_ACCESS AUBO_BUSY AUBO_BAD_STATE - AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

setGroupOutput()

```
int arcs::common_interface::IoControl::setGroupOutput (  
    const std::string & name,  
    uint32_t value)
```

设置组数字输出值

按名称查找组数字输出后，将 value 按位拆分写入对应组成员。

参数

<i>name</i>	组名称。
<i>value</i>	需要写入组数字输出的整数值。

返回

成功返回 0；失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setGroupOutput(self: pyaubo_sdk.IoControl, arg0: str, arg1: int) -> int
```

Lua 函数原型

```
setGroupOutput(name: string, value: number) -> nil
```

Lua 示例

```
setGroupOutput("go_state", 13)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.IoControl.setGroupOutput", "params": ["go_state", 13], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setStandardAnalogInputDomain()

```
int arcs::common_interface::IoControl::setStandardAnalogInputDomain (
    int index,
    int domain)
```

设置标准模拟输入的范围

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。
<i>domain</i>	输入的范围

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
setStandardAnalogInputDomain(self: pyaubo_sdk.IoControl, arg0: int, arg1: int) -> int
```

Lua 函数原型

```
setStandardAnalogInputDomain(index: number, domain: number) -> nil
```

Lua 示例

```
setStandardAnalogInputDomain(0,1)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.IoControl.setStandardAnalogInputDomain", "params": [0, 8], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setStandardAnalogOutput()

```
int arcs::common_interface::IoControl::setStandardAnalogOutput (  
    int index,  
    double value)  
设置标准模拟输出值
```

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。
<i>value</i>	模拟输出值

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
setStandardAnalogOutput(self: pyaubo_sdk.IoControl, arg0: int, arg1: float) -> int
```

Lua 函数原型

```
setStandardAnalogOutput(index: number, value: number) -> nil
```

Lua 示例

```
setStandardAnalogOutput(0,5.4)
```

setStandardAnalogOutputAfterEStop()

```
int arcs::common_interface::IoControl::setStandardAnalogOutputAfterEStop (  
    int index,  
    double value)  
设置标准模拟输出急停后的输出值
```

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。
<i>value</i>	模拟输出值

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
setStandardAnalogOutputAfterEStop(self: pyaubo_sdk.IoControl, arg0: int, arg1: float) -> int
```

Lua 函数原型

```
setStandardAnalogOutputAfterEStop(index: number, value: number) -> nil
```

Lua 示例

```
setStandardAnalogOutputAfterEStop(0,5.4)
```

setStandardAnalogOutputDomain()

```
int arcs::common_interface::IoControl::setStandardAnalogOutputDomain (
    int index,
    int domain)
```

设置标准模拟输出的范围

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。
<i>domain</i>	输出的范围

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
setStandardAnalogOutputDomain(self: pyaubo_sdk.IoControl, arg0: int, arg1: int) -> int
```

Lua 函数原型

```
setStandardAnalogOutputDomain(index: number, domain: number) -> nil
```

Lua 示例

```
setStandardAnalogOutputDomain(0,1)
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.IoControl.setStandardAnalogOutputDomain", "params": [0, 8], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

setStandardAnalogOutputRunstate()

```
int arcs::common_interface::IoControl::setStandardAnalogOutputRunstate (
    int index,
    StandardOutputRunState runstate)
```

设置标准模拟输出状态选择

注解

当给输出状态设置为无 (StandardOutputRunState::None) 时, 用户可以设置模拟输出值。

当给输出设置状态时, 用户不可设置模拟输出值, 控制器会自动设置模拟输出值。

例如, 当设置AO0的输出状态为高电平指示正在拖动示教 (StandardOutputRunState::Handguiding) 时, 机器人进入拖动示教, AO0 的值会自动变为最大值。机器人退出拖动示教, AO0 的值会自动变为 0。

参数

<i>index</i>	表示IO 口的管脚, 管脚编号从 0 开始。例如, 0 表示第一个管脚。
<i>runstate</i>	输出状态选择

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
setStandardAnalogOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int, arg1: arcs::common_interface::StandardOutputRunState) -> int
```

Lua 函数原型

```
setStandardAnalogOutputRunstate(index: number, runstate: number) -> nil
```

Lua 示例

```
setStandardAnalogOutputRunstate(0, 6)
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.IoControl.setStandardAnalogOutputRunstate", "params": [0, "None"], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

setStandardDigitalInputAction()

```
int arcs::common_interface::IoControl::setStandardDigitalInputAction (
    int index,
    StandardInputAction action)
```

设置标准数字输入触发动作

注解

当给输入设置为无触发动作 (StandardInputAction::Default) 时, 用户设置数字输入值为高电平, 不会触发机器人发生动作。

当给输入设置触发动作时, 用户设置数字输入值为高电平, 会触发机器人执行相应的动作。

例如, 当设置DI0 的触发动作为拖动示教 (StandardInputAction::Handguide) 时, 用户设置DI0 为高电平, 机器人会进入拖动示教。设置DI0 为低电平, 机器人会退出拖动示教。

参数

<i>index</i>	表示IO 口的管脚, 管脚编号从 0 开始。例如, 0 表示第一个管脚。
<i>action</i>	触发动作

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setStandardDigitalInputAction(self: pyaubo_sdk.IoControl, arg0: int, arg1: arcs::common_interface::StandardInputAction) -> int
```

Lua 函数原型

```
setStandardDigitalInputAction(index: number, action: number) -> nil
```

Lua 示例

```
setStandardDigitalInputAction(0,1)
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.IoControl.setStandardDigitalInputAction", "params": [0, "Handguide"], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

setStandardDigitalOutput()

```
int arcs::common_interface::IoControl::setStandardDigitalOutput (
    int index,
    bool value)
```

设置标准数字输出值

参数

<i>index</i>	表示IO 口的管脚,
<i>value</i>	输出值

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setStandardDigitalOutput(self: pyaubo_sdk.IoControl, arg0: int, arg1: bool) -> int
```

Lua 函数原型

```
setStandardDigitalOutput(index: number, value: boolean) -> nil
```

Lua 示例

```
setStandardDigitalOutput(0,true)
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.IoControl.setStandardDigitalOutput", "params": [0,true], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

setStandardDigitalOutputAfterEStop()

```
int arcs::common_interface::IoControl::setStandardDigitalOutputAfterEStop (
    int index,
    bool value)
```

设置标准数字输出急停后的输出值

参数

<i>index</i>	表示IO 口的管脚,
<i>value</i>	输出值

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setStandardDigitalOutputAfterEStop(self: pyaubo_sdk.IoControl, arg0: int, arg1: bool) -> int
```

Lua 函数原型

```
setStandardDigitalOutputAfterEStop(index: number, value: boolean) -> nil
```

Lua 示例

```
setStandardDigitalOutputAfterEStop(0,true)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.setStandardDigitalOutputAfterEStop","params":[0,true]↵  
,"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setStandardDigitalOutputPulse()

```
int arcs::common_interface::IoControl::setStandardDigitalOutputPulse (  
    int index,  
    bool value,  
    double duration)
```

设置数字输出脉冲

参数

<i>index</i>	
<i>value</i>	
<i>duration</i>	

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setStandardDigitalOutputPulse(self: pyaubo_sdk.IoControl, arg0: int, arg1: bool, arg2: float) ->
int
```

Lua 函数原型

```
setStandardDigitalOutputPulse(index: number, value: boolean, duration: number) -> nil
```

Lua 示例

```
setStandardDigitalOutputPulse(0,true,2.5)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.setStandardDigitalOutputPulse","params":[0,true,0.5]←
,"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setStandardDigitalOutputRunstate()

```
int arcs::common_interface::IoControl::setStandardDigitalOutputRunstate (
    int index,
    StandardOutputRunState runstate)
```

设置标准数字输出状态选择

注解

当给输出状态设置为无 (StandardOutputRunState::None) 时，用户可以设置数字输出值。

当给输出设置状态时，用户不可设置数字输出值，控制器会自动设置数字输出值。

例如,当设置DO0 的输出状态为高电平指示正在拖动示教 (StandardOutputRunState::Handguiding) 时，机器人进入拖动示教，DO0 会自动变为高电平。机器人退出拖动示教，DO0 会自动变为低电平。

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。
<i>runstate</i>	输出状态选择

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setStandardDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int, arg1: arcs::common_interface::StandardDigitalOutputRunstate) -> int
```

Lua 函数原型

```
setStandardDigitalOutputRunstate(index: number, runstate: number) -> nil
```

Lua 示例

```
setStandardDigitalOutputRunstate(0,1)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.setStandardDigitalOutputRunstate","params":[0,"PowerOn"],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setToolAnalogInputDomain()

```
int arcs::common_interface::IoControl::setToolAnalogInputDomain (
    int index,
    int domain)
```

设置工具端模拟输入的范围

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。
<i>domain</i>	输入的范围

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setToolAnalogInputDomain(self: pyaubo_sdk.IoControl, arg0: int, arg1: int) -> int
```

Lua 函数原型

```
setToolAnalogInputDomain(index: number, domain: number) -> nil
```

Lua 示例

```
setToolAnalogInputDomain(0,1)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.setToolAnalogInputDomain","params":[0,8],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setToolAnalogOutput()

```
int arcs::common_interface::IoControl::setToolAnalogOutput (
    int index,
    double value)
```

设置工具端模拟输出值

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。
<i>value</i>	模拟输出

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setToolAnalogOutput(self: pyaubo_sdk.IoControl, arg0: int, arg1: float) -> int
```

Lua 函数原型

```
setToolAnalogOutput(index: number, value: number) -> nil
```

Lua 示例

```
setToolAnalogOutput(0,1.2)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.setToolAnalogOutput","params":[0,0.5],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":13}
```

setToolAnalogOutputDomain()

```
int arcs::common_interface::IoControl::setToolAnalogOutputDomain (
    int index,
    int domain)
设置工具端模拟输出范围
```

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。
<i>domain</i>	输出的范围

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
setToolAnalogOutputDomain(self: pyaubo_sdk.IoControl, arg0: int, arg1: int) -> int
```

Lua 函数原型

```
setToolAnalogOutputDomain(index: number, domain: number) -> nil
```

Lua 示例

```
setToolAnalogOutputDomain(0,1)
```

setToolAnalogOutputRunstate()

```
int arcs::common_interface::IoControl::setToolAnalogOutputRunstate (
    int index,
    StandardOutputRunState runstate)
设置工具端模拟输出状态选择
```

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。
<i>runstate</i>	输出状态选择

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
setToolAnalogOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int, arg1: arcs::common_interface::StandardOutputRunstate) -> int
```

Lua 函数原型

```
setToolAnalogOutputRunstate(index: number, runstate: number) -> nil
```

Lua 示例

```
setToolAnalogOutputRunstate(0,1)
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.IoControl.setToolAnalogOutputRunstate", "params": [0, "None"], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

setToolDigitalInputAction()

```
int arcs::common_interface::IoControl::setToolDigitalInputAction (
    int index,
    StandardInputAction action)
设置工具数字输入触发动作
```

注解

当给输入设置为无触发动作 (StandardInputAction::Default) 时，用户设置工具数字输入值为高电平，不会触发机器人发生动作。
当给输入设置触发动作时，用户设置工具数字输入值为高电平，会触发机器人执行相应的动作。
例如,当设置TOOL_IO[0]的类型为输入而且触发动作为拖动示教 (StandardInputAction::Handguide) 时，用户设置TOOL_IO[0]为高电平，机器人会进入拖动示教。设置TOOL_IO[0]为低电平，机器人会退出拖动示教。

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。
<i>action</i>	触发动作

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
setToolDigitalInputAction(self: pyaubo_sdk.IoControl, arg0: int, arg1: arcs::common_interface::StandardInputAct
-> int
```

Lua 函数原型

```
setToolDigitalInputAction(index: number, action: number) -> nil
```

Lua 示例

```
setToolDigitalInputAction(0,1)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.setToolDigitalInputAction","params":[0,"Handguide"]←
,"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setToolDigitalOutput()

```
int arcs::common_interface::IoControl::setToolDigitalOutput (
    int index,
    bool value)
设置工具端数字输出值
```

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。
<i>value</i>	数字输出值

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
setToolDigitalOutput(self: pyaubo_sdk.IoControl, arg0: int, arg1: bool) -> int
```

Lua 函数原型

```
setToolDigitalOutput(index: number, value: boolean) -> nil
```

Lua 示例

```
setToolDigitalOutput(0,true)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.IoControl.setToolDigitalOutput","params":[0,true],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setToolDigitalOutputPulse()

```
int arcs::common_interface::IoControl::setToolDigitalOutputPulse (
    int index,
    bool value,
    double duration)
设置工具端数字输出脉冲
```

参数

<i>index</i>	
<i>value</i>	
<i>duration</i>	

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setToolDigitalOutputPulse(self: pyaubo_sdk.IoControl, arg0: int, arg1: bool, arg2: float) -> int
```

Lua 函数原型

```
setToolDigitalOutputPulse(index: number, value: boolean, duration: number) -> nil
```

Lua 示例

```
setToolDigitalOutputPulse(0,true,2)
```


JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.IoControl.setToolDigitalOutputPulse", "params": [0, true, 0.5], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

setToolDigitalOutputRunstate()

```
int arcs::common_interface::IoControl::setToolDigitalOutputRunstate (
    int index,
    StandardOutputRunState runstate)
```

设置工具端数字输出状态选择

注解

当给输出状态设置为无 (StandardOutputRunState::None) 时, 用户可以设置数字输出值。

当给输出设置状态时, 用户不可设置数字输出值, 控制器会自动设置数字输出值。

例如, 当设置TOOL_IO[0]类型为输出且输出状态为高电平指示正在拖动示教 (StandardOutputRunState::Handguiding) 时, 机器人进入拖动示教, TOOL_IO[0]会自动变为高电平。机器人退出拖动示教, TOOL_IO[0]会自动变为低电平。

参数

<i>index</i>	表示IO 口的管脚, 管脚编号从 0 开始。例如, 0 表示第一个管脚。
<i>runstate</i>	输出状态选择

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setToolDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int, arg1: arcs::common_interface::StandardOutputRunState) -> int
```

Lua 函数原型

```
setToolDigitalOutputRunstate(index: number, runstate: number) -> nil
```

Lua 示例

```
setToolDigitalOutputRunstate(0,1)
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.IoControl.setToolDigitalOutputRunstate", "params": [0, "None"], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

setToolIoInput()

```
int arcs::common_interface::IoControl::setToolIoInput (
    int index,
    bool input)
```

设置指定的工具端数字IO 为输入或输出

工具端数字IO 比较特殊，IO 可以配置为输入或者输出

参数

<i>index</i>	表示IO 口的管脚，管脚编号从 0 开始。例如，0 表示第一个管脚。
<i>input</i>	表示指定IO 是否为输入。input 为 true 时，设置指定IO 为输入，否则为输出

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setToolIoInput(self: pyaubo_sdk.IoControl, arg0: int, arg1: bool) -> int
```

Lua 函数原型

```
setToolIoInput(index: number, input: boolean) -> nil
```

Lua 示例

```
setToolIoInput(0,true)
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.IoControl.setToolIoInput", "params": [0,true], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

setToolVoltageOutputDomain()

```
int arcs::common_interface::IoControl::setToolVoltageOutputDomain (
    int domain)
```

设置工具端电源电压值 (单位 V)

参数

<i>domain</i>	工具端电源电压值, 可选三个档位, 分别为 0、12 和 24。 0 表示 0V, 12 表示 12V, 24 表示 24V。

返回

成功返回 0; 失败返回错误码 AUBO_REQUEST_IGNORE AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setToolVoltageOutputDomain(self: pyaubo_sdk.IoControl, arg0: int) -> int
```

Lua 函数原型

```
setToolVoltageOutputDomain(domain: number) -> nil
```

Lua 示例

```
setToolVoltageOutputDomain(24)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.IoControl.setToolVoltageOutputDomain", "params": [24], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

unwindEncDeltaTickCount()

```
int arcs::common_interface::IoControl::unwindEncDeltaTickCount (
    int delta_count)
```

防止在计数超出范围时计数错误

参数

<i>delta_count</i>	

返回

成功返回 0; 失败返回错误码 AUBO_NO_ACCESS AUBO_BUSY AUBO_BAD_STATE -
AUBO_BAD_STATE

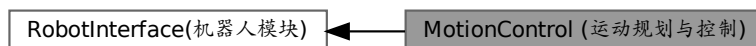
异常

<code>arcs::common_interface::AuboException</code>

8.6.5 MotionControl (运动规划与控制)

MotionControl

MotionControl (运动规划与控制) 的协作图:



函数

- double `arcs::common_interface::MotionControl::getEqradius ()`
获取等效半径, 单位 *m* *moveLine*/*moveCircle* 时, 末端姿态旋转的角度等效到末端位置移动可以通过 *setEqradius* 设置, 默认为 1
- int `arcs::common_interface::MotionControl::setEqradius (double eqradius)`
设置等效半径, 单位 *m* *moveLine*/*moveCircle* 时, 末端姿态旋转的角度等效到末端位置移动, 数值越大, 姿态旋转速度越快
- int `arcs::common_interface::MotionControl::setSpeedFraction (double fraction)`
动态调整机器人运行速度和加速度比例 (0., 1.
- double `arcs::common_interface::MotionControl::getSpeedFraction ()`
获取速度和加速度比例, 默认为 1 可以通过 *setSpeedFraction* 接口设置
- int `arcs::common_interface::MotionControl::speedFractionCritical (bool enable)`
速度比例设置临界区, 使能之后速度比例被强制设定为 1.
- int `arcs::common_interface::MotionControl::speedFractionCritical1 (bool enable)`
速度比例设置临界区, 使能之后速度比例被强制设定为当前生效的比例, 失能之后恢复之前的速度比例
- bool `arcs::common_interface::MotionControl::isSpeedFractionCritical ()`
是否处于速度比例设置临界区
- bool `arcs::common_interface::MotionControl::isBlending ()`
是否处交融区
- int `arcs::common_interface::MotionControl::pathOffsetLimits (double v, double a)`
设置偏移的最大速度和最大加速度仅对 *pathOffsetSet* 中 *type=1* 有效
- int `arcs::common_interface::MotionControl::pathOffsetCoordinate (int ref_coord)`
设置偏移的参考坐标系仅对 *pathOffsetSet* 中 *type=1* 有效
- int `arcs::common_interface::MotionControl::pathOffsetEnable ()`
路径偏移使能
- int `arcs::common_interface::MotionControl::pathOffsetSet (const std::vector< double > &offset, int type=0)`
设置路径偏移
- int `arcs::common_interface::MotionControl::pathOffsetDisable ()`
路径偏移失能
- int `arcs::common_interface::MotionControl::pathOffsetSupv (const std::vector< double > &min, const std::vector< double > &max, int strategy)`
- int `arcs::common_interface::MotionControl::jointOffsetEnable ()`
关节偏移使能

- int `arcs::common_interface::MotionControl::jointOffsetSet` (const std::vector< double > &offset, int type=1)
设置关节偏移
- int `arcs::common_interface::MotionControl::jointOffsetDisable` ()
关节偏移失能
- int `arcs::common_interface::MotionControl::getQueueSize` ()
获取已经入队的指令段 (INST) 数量, 运动指令包括 *moveJoint/moveLine/moveCircle* 等运动指令以及 *setPayload* 等配置指令
- int `arcs::common_interface::MotionControl::getTrajectoryQueueSize` ()
获取已经入队的运动规划插补点数量
- int `arcs::common_interface::MotionControl::getExecId` ()
获取当前正在插补的运动指令段的ID
- double `arcs::common_interface::MotionControl::getDuration` (int id)
获取指定ID 的运动指令段的预期执行时间
- double `arcs::common_interface::MotionControl::getMotionLeftTime` (int id)
获取指定ID 的运动指令段的剩余执行时间
- int `arcs::common_interface::MotionControl::stopMove` (bool quick, bool all_tasks)
StopMove 用于临时停止机器人和外部轴的运动以及相关工艺进程。如果调用 *StartMove* 指令, 则运动和工艺进程将恢复。
- int `arcs::common_interface::MotionControl::startMove` ()
StartMove 用于在以下情况下恢复机器人、外部轴的运动以及相关工艺进程: • 通过 *StopMove* 指令停止后。• 执行 *StorePath ... RestoPath* 序列后。• 发生异步运动错误 (如 *ERR_PATH_STOP*) 或特定工艺错误并在 *ERROR* 处理器中处理后。
- int `arcs::common_interface::MotionControl::storePath` (bool keep_sync)
storePath
- int `arcs::common_interface::MotionControl::clearPath` ()
ClearPath (清除路径) 清除当前运动路径层级 (基础层级或 *StorePath* 层级) 上的所有运动路径。
- int `arcs::common_interface::MotionControl::restoPath` ()
restoPath
- double `arcs::common_interface::MotionControl::getProgress` ()
获取当前运动指令段的执行进度
- int `arcs::common_interface::MotionControl::setWorkObjectHold` (const std::string &module_name, const std::vector< double > &mounting_pose)
当工件安装在另外一台机器人的末端或者外部轴上时, 指定其名字和安装位置
- std::tuple< std::string, std::vector< double > > `arcs::common_interface::MotionControl::getWorkObjectHold` ()
获取工件安装信息
- std::vector< double > `arcs::common_interface::MotionControl::getPauseJointPositions` ()
- int `arcs::common_interface::MotionControl::setResumeStartPoint` (const std::vector< double > &q, int move_type, double blend_radius, const std::vector< double > &qdmax, const std::vector< double > &qddmax, const std::vector< double > &vmax, const std::vector< double > &amax)
设置继续运动参数
- int `arcs::common_interface::MotionControl::getResumeMode` ()
获取继续运动模式
- ARCS_DEPRECATED int `arcs::common_interface::MotionControl::setServoMode` (bool enable)
设置伺服模式使用 *setServoModeSelect* 替代
- ARCS_DEPRECATED bool `arcs::common_interface::MotionControl::isServoModeEnabled` ()
判断伺服模式是否使能使用 *getServoModeSelect* 替代
- int `arcs::common_interface::MotionControl::setServoModeSelect` (int mode)
设置伺服运动模式
- int `arcs::common_interface::MotionControl::getServoModeSelect` ()
获取伺服运动模式

- `int arcs::common_interface::MotionControl::servoJoint` (`const std::vector< double > &q`, `double a`, `double v`, `double t`, `double lookahead_time`, `double gain`)
关节空间伺服
- `int arcs::common_interface::MotionControl::servoCartesian` (`const std::vector< double > &pose`, `double a`, `double v`, `double t`, `double lookahead_time`, `double gain`)
笛卡尔空间伺服
- `int arcs::common_interface::MotionControl::servoJointWithAxes` (`const std::vector< double > &q`, `const std::vector< double > &extq`, `double a`, `double v`, `double t`, `double lookahead_time`, `double gain`)
伺服运动 (带外部轴), 用于执行离线轨迹、透传用户规划轨迹等
- `int arcs::common_interface::MotionControl::servoCartesianWithAxes` (`const std::vector< double > &pose`, `const std::vector< double > &extq`, `double a`, `double v`, `double t`, `double lookahead_time`, `double gain`)
伺服运动 (带外部轴), 用于执行离线轨迹、透传用户规划轨迹等与 *servoJointWithAxes* 区别在于接收笛卡尔空间位姿而不是关节角度 (由软件内部直接做逆解)
- `int arcs::common_interface::MotionControl::trackJoint` (`const std::vector< double > &q`, `double t`, `double smooth_scale`, `double delay_sacle`)
跟踪运动, 用于执行离线轨迹、透传用户规划轨迹等
- `int arcs::common_interface::MotionControl::trackCartesian` (`const std::vector< double > &pose`, `double t`, `double smooth_scale`, `double delay_sacle`)
跟踪运动, 用于执行离线轨迹、透传用户规划轨迹等与 *trackJoint* 区别在于接收笛卡尔空间位姿而不是关节角度 (由软件内部直接做逆解)
- `int arcs::common_interface::MotionControl::followJoint` (`const std::vector< double > &q`)
关节空间跟随
- `int arcs::common_interface::MotionControl::followLine` (`const std::vector< double > &pose`)
笛卡尔空间跟随
- `int arcs::common_interface::MotionControl::speedJoint` (`const std::vector< double > &qd`, `double a`, `double t`)
关节空间速度跟随
- `int arcs::common_interface::MotionControl::resumeSpeedJoint` (`const std::vector< double > &qd`, `double a`, `double t`)
关节空间速度跟随 (机械臂运行工程时发生碰撞, 通过此接口移动到安全位置)
- `int arcs::common_interface::MotionControl::speedLine` (`const std::vector< double > &xd`, `double a`, `double t`)
笛卡尔空间速度跟随
- `int arcs::common_interface::MotionControl::resumeSpeedLine` (`const std::vector< double > &xd`, `double a`, `double t`)
笛卡尔空间速度跟随 (机械臂运行工程时发生碰撞, 通过此接口移动到安全位置)
- `int arcs::common_interface::MotionControl::moveSpline` (`const std::vector< double > &q`, `double a`, `double v`, `double duration`)
在关节空间做样条插值
- `int arcs::common_interface::MotionControl::moveJoint` (`const std::vector< double > &q`, `double a`, `double v`, `double blend_radius`, `double duration`)
添加关节运动
- `int arcs::common_interface::MotionControl::moveJointWithAxisGroup` (`const std::vector< double > &q`, `double a`, `double v`, `double blend_radius`, `double duration`, `const std::string &group_name`, `const std::vector< double > &extq`)
机器人与外部轴同步运动
- `int arcs::common_interface::MotionControl::resumeMoveJoint` (`const std::vector< double > &q`, `double a`, `double v`, `double duration`)
通过关节运动移动到暂停点的位置
- `int arcs::common_interface::MotionControl::moveLine` (`const std::vector< double > &pose`, `double a`, `double v`, `double blend_radius`, `double duration`)

- 添加直线运动
 - int `arcs::common_interface::MotionControl::moveLineWithAxisGroup` (const std::vector< double > &pose, double a, double v, double blend_radius, double duration, const std::string &group_name, const std::vector< double > &extq)
- 直线运动与外部轴同步运动
 - int `arcs::common_interface::MotionControl::moveProcess` (const std::vector< double > &pose, double a, double v, double blend_radius)
- 添加工艺运动
 - int `arcs::common_interface::MotionControl::resumeMoveLine` (const std::vector< double > &pose, double a, double v, double duration)
- 通过直线运动移动到暂停点的位置
 - int `arcs::common_interface::MotionControl::moveCircle` (const std::vector< double > &via_pose, const std::vector< double > &end_pose, double a, double v, double blend_radius, double duration)
- 添加圆弧运动
 - int `arcs::common_interface::MotionControl::moveCircleWithAxisGroup` (const std::vector< double > &via_pose, const std::vector< double > &end_pose, double a, double v, double blend_radius, double duration, const std::string &group_name, const std::vector< double > &extq)
- moveCircle* 与外部轴同步运动
 - int `arcs::common_interface::MotionControl::setCirclePathMode` (int mode)
- 设置圆弧路径模式
 - int `arcs::common_interface::MotionControl::moveCircle2` (const `CircleParameters` &param)
- 高级圆弧或者圆周运动
 - int `arcs::common_interface::MotionControl::pathBufferAlloc` (const std::string &name, int type, int size)
- 新建一个路径点缓存
 - int `arcs::common_interface::MotionControl::pathBufferAppend` (const std::string &name, const std::vector< std::vector< double > > &waypoints)
- 向路径缓存添加路点
 - int `arcs::common_interface::MotionControl::pathBufferEval` (const std::string &name, const std::vector< double > &a, const std::vector< double > &v, double t)
- 计算、优化等耗时操作，传入的参数相同时不会重新计算
 - bool `arcs::common_interface::MotionControl::pathBufferValid` (const std::string &name)
- 指定名字的 *buffer* 是否有效
 - int `arcs::common_interface::MotionControl::pathBufferFree` (const std::string &name)
- 释放路径缓存
 - int `arcs::common_interface::MotionControl::pathBufferFilter` (const std::string &name, int order, double fd, double fs)
- 关节空间路径滤波器
 - std::vector< std::string > `arcs::common_interface::MotionControl::pathBufferList` ()
- 列出所有缓存路径的名字
 - int `arcs::common_interface::MotionControl::movePathBuffer` (const std::string &name)
- 执行缓存的路径
 - int `arcs::common_interface::MotionControl::moveIntersection` (const std::vector< std::vector< double > > &poses, double a, double v, double main_pipe_radius, double sub_pipe_radius, double normal_distance, double normal_alpha)
- 相贯线接口
 - int `arcs::common_interface::MotionControl::stopJoint` (double acc)
- 关节空间停止运动
 - int `arcs::common_interface::MotionControl::resumeStopJoint` (double acc)
- 关节空间停止运动 (机械臂运行工程时发生碰撞，通过 *resumeSpeedJoint* 接口移动到安全位置后需要停止时调用此接口)
 - int `arcs::common_interface::MotionControl::stopLine` (double acc, double acc_rot)
- 停止 *moveLine*/*moveCircle* 等在笛卡尔空间的运动

- int `arcs::common_interface::MotionControl::resumeStopLine` (double acc, double acc_rot)
笛卡尔空间停止运动 (机械臂运行工程时发生碰撞, 通过 `resumeSpeedLine` 接口移动到安全位置后需要停止时调用此接口)
- int `arcs::common_interface::MotionControl::weaveStart` (const std::string &params)
- int `arcs::common_interface::MotionControl::weaveEnd` ()
结束摆动
- int `arcs::common_interface::MotionControl::setFuturePointSamplePeriod` (double sample_time)
设置未来路径上点的采样时间间隔
- std::vector< std::vector< double > > `arcs::common_interface::MotionControl::getFuturePathPointsJoint` ()
获取未来路径上的轨迹点
- int `arcs::common_interface::MotionControl::setConveyorTrackEncoder` (int encoder_id, int tick←_per_meter)
设置传送带编码器参数
- int `arcs::common_interface::MotionControl::conveyorTrackCircle` (int encoder_id, const std::vector< double > &center, bool rotate_tool)
圆形传送带跟随
- int `arcs::common_interface::MotionControl::conveyorTrackLine` (int encoder_id, const std::vector< double > &direction)
线性传送带跟随
- int `arcs::common_interface::MotionControl::conveyorTrackStop` (int encoder_id, double a)
终止传送带跟随
- bool `arcs::common_interface::MotionControl::conveyorTrackSwitch` (int encoder_id)
切换传送带追踪物品如果当前物品正在处于跟踪状态, 则将该物品出队, 不再跟踪, 返回 `true` 如果没有物品正在处于跟踪状态, 返回 `false`
- bool `arcs::common_interface::MotionControl::hasItemOnConveyorToTrack` (int encoder_id)
传送带上是否有物品可以跟踪
- int `arcs::common_interface::MotionControl::conveyorTrackCreatItem` (int encoder_id, int item_id, const std::vector< double > &offset)
增加传送带队列
- int `arcs::common_interface::MotionControl::setConveyorTrackCompensate` (int encoder_id, double comp)
设置传送带跟踪的补偿值
- bool `arcs::common_interface::MotionControl::isConveyorTrackSync` (int encoder_id)
判断传送带与机械臂之间是否达到相对静止
- int `arcs::common_interface::MotionControl::setConveyorTrackLimit` (int encoder_id, double limit)
设置传送带跟踪的最大距离限制
- int `arcs::common_interface::MotionControl::setConveyorTrackStartWindow` (int encoder_id, double window_min, double window_max)
设置传送带跟踪的启动窗口
- int `arcs::common_interface::MotionControl::setConveyorTrackSensorOffset` (int encoder_id, double offset)
设置传送带示教位置到同步开关之间的距离
- int `arcs::common_interface::MotionControl::setConveyorTrackSyncSeparation` (int encoder_id, double distance, double time)
设置传送带同步分离, 用于过滤掉同步开关中不需要的信号
- int `arcs::common_interface::MotionControl::setConveyorTrackVelThreshold` (int encoder_id, double threshold)
设置传送带运行速度阈值, 用于过滤掉编码器突变值导致的位置突变
- int `arcs::common_interface::MotionControl::conveyorTrackClearItems` (int encoder_id)
清空传动带队列中的所有对象
- std::vector< int > `arcs::common_interface::MotionControl::getConveyorTrackQueue` (int encoder←_id)

- 获取传送带队列的编码器值
- `int arcs::common_interface::MotionControl::moveSpiral (const SpiralParameters &param, double blend_radius, double v, double a, double t)`
螺旋线运动
 - `int arcs::common_interface::MotionControl::getLookAheadSize ()`
获取前瞻段数
 - `int arcs::common_interface::MotionControl::setLookAheadSize (int size)`
设置前瞻段数 *1*.
 - `int arcs::common_interface::MotionControl::weaveUpdateParameters (const std::string &params)`
更新摆动过程中的频率和振幅
 - `int arcs::common_interface::MotionControl::enableJointSoftServo (const std::vector< double > &stiffness)`
设置关节电流环刚度系数
 - `int arcs::common_interface::MotionControl::disableJointSoftServo ()`
关闭关节电流环刚度系数
 - `bool arcs::common_interface::MotionControl::isJointSoftServoEnabled ()`
判断关节电流环刚度系数是否使能
 - `int arcs::common_interface::MotionControl::enableVibrationSuppress (const std::vector< double > &omega, const std::vector< double > &zeta, int level)`
打开振动抑制
 - `int arcs::common_interface::MotionControl::disbaleVibrationSuppress ()`
关闭振动抑制
 - `int arcs::common_interface::MotionControl::setTimeOptimalEnable (bool enable)`
设置时间最优算法默认关闭
 - `bool arcs::common_interface::MotionControl::isTimeOptimalEnabled ()`
获取时间最优算法状态: *true* - 开启 *false* - 关闭
 - `bool arcs::common_interface::MotionControl::isSupportedTimeOptimal ()`
获取是否支持时间最优算法: *true* - 支持 *false* - 不支持
 - `int arcs::common_interface::MotionControl::setTcpMaxLinearVelocity (double v)`
设置TCP 最大线速度
 - `double arcs::common_interface::MotionControl::getTcpMaxLinearVelocity ()`
获取TCP 最大线速度
 - `int arcs::common_interface::MotionControl::resetTcpMaxLinearVelocity ()`
重置 TCP 最大线速度
 - `int arcs::common_interface::MotionControl::setEndPath ()`
设置轨迹终止点 (以当前轨迹段为界)
 - `int arcs::common_interface::MotionControl::getWeaveDirection ()`
获取当前摆焊轨迹点方向

详细描述

MotionControl

函数说明

clearPath()

`int arcs::common_interface::MotionControl::clearPath ()`

ClearPath (清除路径) 清除当前运动路径层级 (基础层级或 StorePath 层级) 上的所有运动路径。

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE AUBO_BUSY -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Lua 函数原型

```
clearPath() -> number
```

Lua 示例

```
num = clearPath()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.clearPath","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

conveyorTrackCircle()

```
int arcs::common_interface::MotionControl::conveyorTrackCircle (
    int encoder_id,
    const std::vector< double > & center,
    bool rotate_tool)
```

圆形传送带跟随

注解

暂未实现

参数

<code>encoder↔ _id</code>	0-集成传感器
<code>rotate_tool</code>	

返回**异常**

<code>arcs::common_interface::AuboException</code>

conveyorTrackClearItems()

```
int arcs::common_interface::MotionControl::conveyorTrackClearItems (
    int encoder_id)
```

清空传动带队列中的所有对象

参数

<i>encoder↔ _id</i>	预留

返回

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
conveyorTrackClearItems(self: pyaubo_sdk.MotionControl, arg0: int) -> int
```

Lua 函数原型

```
conveyorTrackClearItems(encoder_id: number) -> int
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.conveyorTrackClearItems", "params": [0], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

conveyorTrackCreatItem()

```
int arcs::common_interface::MotionControl::conveyorTrackCreatItem (  
    int encoder_id,  
    int item_id,  
    const std::vector< double > & offset)
```

增加传送带队列

参数

<i>encoder↔ _id</i>	预留
<i>item_id</i>	物品ID
<i>offset</i>	当前物品点位相对于模板物品点位的偏移值

返回

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
conveyorTrackCreatItem(self: pyaubo_sdk.MotionControl, arg0: int, arg1:int, arg2: List[float]) ->
int
```

Lua 函数原型

```
conveyorTrackCreatItem(encoder_id: number, item_id: number, offset: table) -> number
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.conveyorTrackCreatItem","params":[0,2,{0.0,0.0,0.0,0.0,0.0,0.0,0.0}], "id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0.0}
```

conveyorTrackLine()

```
int arcs::common_interface::MotionControl::conveyorTrackLine (
    int encoder_id,
    const std::vector< double > & direction)
```

线性传送带跟随

参数

<code>encoder_id</code>	预留
<code>direction</code>	传送带相对机器人基坐标系的移动方向

返回

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
conveyorTrackLine(self: pyaubo_sdk.MotionControl) -> int
```

Lua 函数原型

```
conveyorTrackLine -> int
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.MotionControl.conveyorTrackLine", "params": [1, [1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

conveyorTrackStop()

```
int arcs::common_interface::MotionControl::conveyorTrackStop (
    int encoder_id,
    double a)
```

终止传送带跟随

参数

<i>encoder_id</i>	预留
<i>a</i>	预留

返回

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
conveyorTrackStop(self: pyaubo_sdk.MotionControl) -> int
```

Lua 函数原型

```
conveyorTrackStop -> int
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.MotionControl.conveyorTrackStop", "params": [0, 1.0], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

conveyorTrackSwitch()

```
bool arcs::common_interface::MotionControl::conveyorTrackSwitch (
    int encoder_id)
```

切换传送带追踪物品如果当前物品正在处于跟踪状态，则将该物品出队，不再跟踪，返回 true 如果没有物品正在处于跟踪状态，返回 false

返回

异常

<code>arcs::common_interface::AuboException</code>

参数

<code>encoder↔ _id</code>	预留

Python 函数原型

```
conveyorTrackSwitch(self: pyaubo_sdk.MotionControl) -> bool
```

Lua 函数原型

```
conveyorTrackSwitch() -> boolean
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.conveyorTrackSwitch","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":true}
```

disableJointSoftServo()

```
int arcs::common_interface::MotionControl::disableJointSoftServo ()
```

关闭关节电流环刚度系数

返回

返回 0 表示成功，其他为错误码

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
disableJointSoftServo(self: pyaubo_sdk.MotionControl) -> int
```

Lua 函数原型

```
disableJointSoftServo() -> number
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.disableJointSoftServo","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":1}
```

disbaleVibrationSuppress()

```
int arcs::common_interface::MotionControl::disbaleVibrationSuppress ()
```

关闭振动抑制

返回

已使能返回 true, 反之则返回 false

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
disbaleVibrationSuppress(self: pyaubo_sdk.MotionControl) -> int
```

Lua 函数原型

```
disbaleVibrationSuppress() -> number
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.disbaleVibrationSuppress","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":1}
```

enableJointSoftServo()

```
int arcs::common_interface::MotionControl::enableJointSoftServo (
    const std::vector< double > & stiffness)
设置关节电流环刚度系数
```

参数

<code>stiffness</code>	各个关节刚度系数, 百分比。[0 -> 1], 值越大, 表现为越硬。

返回

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
enableJointSoftServo(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
```

Lua 函数原型

```
enableJointSoftServo(stiffness: table) -> int
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.RobotConfig.enableJointSoftServo", "params": [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

enableVibrationSuppress()

```
int arcs::common_interface::MotionControl::enableVibrationSuppress (
    const std::vector< double > & omega,
    const std::vector< double > & zeta,
    int level)
```

打开振动抑制

参数

<i>omega</i>	振动抑制频率, 长度一般为笛卡尔维度或关节自由度. 为 0 时关闭抑制
<i>zeta</i>	振动抑制阻尼比, 长度一般为笛卡尔维度或关节自由度. 为 0 时关闭抑制
<i>level</i>	振动抑制等级 (1~3), 等级越高抑制效果越明显响应速度相应变慢. 等级 0 关闭振动抑制

返回值

0	成功
<i>AUBO_BAD_STATE(1)</i>	当前安全模式处于非 Normal、ReducedMode、Recovery 状态
<i>AUBO_QUEUE_FULL(2)</i>	规划队列已满
<i>AUBO_BUSY(3)</i>	上一条指令正在执行中
<i>-AUBO_BAD_STATE(-1)</i>	可能的原因包括但不限于: 线程已分离、线程被终止、task_id 未找到, 或者当前机器人模式非 Running
<i>-AUBO_TIMEOUT(-4)</i>	调用接口超时

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
enableVibrationSuppress(self: pyaubo_sdk.MotionControl, arg0: list[float], arg1: list[float], arg2: int) -> int
```


Lua 函数原型

```
enableVibrationSuppress(omega: table, zeta: table, level: number) -> nil
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.enableVibrationSuppress", "params": [[0.5, 0.5, 0.5, 0.5, 0.5, 0.5], [0.5, 0.5, 0.5, 0.5, 0.5, 0.5], 2], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

followJoint()

```
int arcs::common_interface::MotionControl::followJoint (
    const std::vector< double > & q)
```

关节空间跟随

注解

暂未实现

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
followJoint(self: pyaubo_sdk.MotionControl, arg0: List[float]) -> int
```

Lua 函数原型

```
followJoint(q: table) -> nil
```

Lua 示例

```
followJoint({0,0,0,0,0,0})
```

followLine()

```
int arcs::common_interface::MotionControl::followLine (
    const std::vector< double > & pose)
```

笛卡尔空间跟随

注解

暂未实现

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
followLine(self: pyaubo_sdk.MotionControl, arg0: List[float]) -> int
```

Lua 函数原型

```
followLine(pose: table) -> nil
```

Lua 示例

```
followLine({0.58712,-0.15775,0.48703,2.76,0.344,1.432})
```

getConveyorTrackQueue()

```
std::vector< int > arcs::common_interface::MotionControl::getConveyorTrackQueue (
    int encoder_id)
```

获取传送带队列的编码器值

参数

<i>encoder_id</i>	预留

返回

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getConveyorTrackQueue(self: pyaubo_sdk.MotionControl, arg0: int) -> List[int]
```

Lua 函数原型

```
getConveyorTrackQueue(encoder_id: number) -> table
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.getConveyorTrackQueue", "params": [0, "id": 1]}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": {[-500,-200,150,-50]}}
```

getDuration()

```
double arcs::common_interface::MotionControl::getDuration (
    int id)
```

获取指定ID 的运动指令段的预期执行时间

参数

<i>id</i>	运动指令段ID

返回

返回预期执行时间

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getDuration(self: pyaubo_sdk.MotionControl, arg0: int) -> float
```

Lua 函数原型

```
getDuration(id: number) -> number
```

Lua 示例

```
num = getDuration(16)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.getDuration", "params": [16], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0.0}
```

getEqradius()

`double arcs::common_interface::MotionControl::getEqradius ()`
 获取等效半径，单位 m moveLine/moveCircle 时，末端姿态旋转的角度等效到末端位置移动可以通过 setEqradius 设置，默认为 1

返回

返回等效半径

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getEqradius(self: pyaubo_sdk.MotionControl) -> float
```

Lua 函数原型

```
getEqradius() -> number
```

Lua 示例

```
num = getEqradius()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.getEqradius","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":1.0}
```

getExecId()

```
int arcs::common_interface::MotionControl::getExecId ()
```

获取当前正在插补的运动指令段的ID

返回

当前正在插补的运动指令段的ID

返回值

-1	表示轨迹队列为空 像 movePathBuffer 运动中的 buffer 或者规划器 (moveJoint 和 moveLine 等) 里的队列都属于轨迹队列

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getExecId(self: pyaubo_sdk.MotionControl) -> int
```

Lua 函数原型

```
getExecId() -> number
```

Lua 示例

```
num = getExecId()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.getExecId","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":-1}
```

getFuturePathPointsJoint()

```
std::vector< std::vector< double > > arcs::common_interface::MotionControl::getFuturePathPointsJoint ()
```

获取未来路径上的轨迹点

返回

路点 (100ms * 10)

异常

<code>arcs::common_interface::AuboException</code>

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.getFuturePathPointsJoint","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[]}
```

getLookAheadSize()

```
int arcs::common_interface::MotionControl::getLookAheadSize ()
```

获取前瞻段数

返回

返回前瞻段数

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getLookAheadSize(self: pyaubo_sdk.MotionControl) -> int
```

Lua 函数原型

```
getLookAheadSize() -> number
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.getLookAheadSize","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":1}
```

getMotionLeftTime()

```
double arcs::common_interface::MotionControl::getMotionLeftTime (
    int id)
```

获取指定ID 的运动指令段的剩余执行时间

参数

<i>id</i>	运动指令段ID

返回

返回剩余执行时间

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getMotionLeftTime(self: pyaubo_sdk.MotionControl, arg0: int) -> float
```

Lua 函数原型

```
getMotionLeftTime(id: number) -> number
```

Lua 示例

```
num = getMotionLeftTime(16)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.getMotionLeftTime", "params": [16], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0.0}
```

getPauseJointPositions()

```
std::vector< double > arcs::common_interface::MotionControl::getPauseJointPositions ()
```

注解

获取暂停点关节位置

常用于运行工程时发生碰撞后继续运动过程中 (先通过 resumeMoveJoint 或 resumeMoveLine 运动到暂停位置, 再恢复工程)

返回

暂停关节位置

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getPauseJointPositions(self: pyaubo_sdk.MotionControl) -> List[float]
```

Lua 函数原型

```
getPauseJointPositions() -> table
```

Lua 示例

```
JointPositions = getPauseJointPositions()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.getPauseJointPositions","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[8.2321e-13,-0.200999,1.33999,0.334999,1.206,-6.39383e-12]}
```

getProgress()

```
double arcs::common_interface::MotionControl::getProgress ()  
获取当前运动指令段的执行进度
```

返回

返回执行进度

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getProgress(self: pyaubo_sdk.MotionControl) -> float
```

Lua 函数原型

```
getProgress() -> number
```

Lua 示例

```
num = getProgress()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.getProgress","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0.0}
```

getQueueSize()

```
int arcs::common_interface::MotionControl::getQueueSize ()
```

获取已经入队的指令段 (INST) 数量, 运动指令包括 moveJoint/moveLine/moveCircle 等运动指令以及 setPayload 等配置指令

指令一般会在接口宏定义里面用 _INST 指示, 比如 moveJoint _INST(MotionControl, 5, moveJoint, q, a, v, blend_radius, duration)

返回

已经入队的指令段数量

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getQueueSize(self: pyaubo_sdk.MotionControl) -> int
```

Lua 函数原型

```
getQueueSize() -> number
```

Lua 示例

```
num = getQueueSize()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.getQueueSize","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getResumeMode()

```
int arcs::common_interface::MotionControl::getResumeMode ()
```

获取继续运动模式

返回

0:继续运动起始点为暂停点 1:继续运动起始点为指定点

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getResumeMode(self: pyaubo_sdk.MotionControl) -> int
```


Lua 函数原型

```
getResumeMode() -> int
```

Lua 示例

```
num = getResumeMode()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.getResumeMode","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":false}
```

getServoModeSelect()

```
int arcs::common_interface::MotionControl::getServoModeSelect ()  
获取伺服运动模式
```

返回

Lua 函数原型

```
getServoModeSelect() -> number
```

Lua 示例

```
num = getServoModeSelect()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.getServoModeSelect","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getSpeedFraction()

```
double arcs::common_interface::MotionControl::getSpeedFraction ()  
获取速度和加速度比例，默认为 1 可以通过 setSpeedFraction 接口设置
```

返回

返回速度和加速度比例

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getSpeedFraction(self: pyaubo_sdk.MotionControl) -> float
```

Lua 函数原型

```
getSpeedFraction() -> number
```

Lua 示例

```
num = getSpeedFraction()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.getSpeedFraction","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":1.0}
```

getTcpMaxLinearVelocity()

```
double arcs::common_interface::MotionControl::getTcpMaxLinearVelocity ()  
获取TCP 最大线速度
```

返回

当前TCP 最大线速度值, 单位为 m/s (米/秒)

异常

<code>arcs::common_interface::AuboException</code>

Lua 函数原型

```
getTcpMaxLinearVelocity() -> number
```

Lua 示例

```
local v = getTcpMaxLinearVelocity() -- 获取TCP 最大线速度
```

Python 函数原型

```
getTcpMaxLinearVelocity() -> float
```

Python 示例

```
v = getTcpMaxLinearVelocity() # 获取TCP 最大线速度
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.getTcpMaxLinearVelocity","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0.5}
```

getTrajectoryQueueSize()

```
int arcs::common_interface::MotionControl::getTrajectoryQueueSize ()
```

获取已经入队的运动规划插补点数量

返回

已经入队的运动规划插补点数量

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getTrajectoryQueueSize(self: pyaubo_sdk.MotionControl) -> int
```

Lua 函数原型

```
getTrajectoryQueueSize() -> number
```

Lua 示例

```
num = getTrajectoryQueueSize()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.getTrajectoryQueueSize","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getWeaveDirection()

```
int arcs::common_interface::MotionControl::getWeaveDirection ()
```

获取当前摆焊轨迹点方向

获取机器人当前执行轨迹点在摆焊（Weave）运动中的方向状态，用于指示该点位于摆焊坐标系 Y 轴的正半轴或负半轴区域。

当机器人正在执行摆焊轨迹时，该接口返回当前摆焊点所属的摆动方向，可用于判断摆焊运动处于正向摆动阶段或反向摆动阶段。

当机器人未执行摆焊轨迹时，该接口返回 0。

该接口为状态查询接口，不会影响当前轨迹执行，也不会改变机器人运动状态。

返回

返回当前摆焊方向，对应 WeaveDirection 枚举值：

- 0 : NONE 当前未执行摆焊轨迹
- 1 : POSITIVE 当前位于摆焊坐标系 Y 轴正半轴区域
- 2 : NEGATIVE 当前位于摆焊坐标系 Y 轴负半轴区域

异常

<code>arcs::common_interface::AuboException</code>	

Lua 函数原型

```
getWeaveDirection() -> number
```

Lua 示例

```
local dir = getWeaveDirection() if dir == 1 then - 正方向摆动 elseif dir == 2 then - 负方向摆动
end
```

Python 函数原型

```
getWeaveDirection() -> int
```

Python 示例

```
dir = getWeaveDirection() if dir == 1:
```

```
    positive direction
```

```
pass elif dir == 2:
```

```
    negative direction
```

```
    pass
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.getWeaveDirection","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":1}
```

getWorkObjectHold()

```
std::tuple< std::string, std::vector< double > > arcs::common_interface::MotionControl::getWorkObjectHold ( )
```

```
    获取工件安装信息
```

注解

```
    暂未实现
```

返回

```
    返回一个包含控制模块名字和安装位姿的元组
```

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getWorkObjectHold(self: pyaubo_sdk.MotionControl) -> Tuple[str, List[float]]
```

Lua 函数原型

```
getWorkObjectHold() -> table
```

Lua 示例

```
Object_table = getWorkObjectHold()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.getWorkObjectHold","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":["",[]]}
```

hasItemOnConveyorToTrack()

```
bool arcs::common_interface::MotionControl::hasItemOnConveyorToTrack (
    int encoder_id)
    传送带上是否有物品可以跟踪
```

参数

<i>encoder</i> ↔ <i>_id</i>	预留

返回

如果队列第一个物品为跟踪状态，返回 true 如果队列第一个物品不为跟踪状态，则对队列中其余物品进行是否在启动窗口内的判断超出启动窗口则出队，直到有物品处于启动窗口内，使其变为跟踪状态返回 true，队列中没有物品在启动窗口内返回 false

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
hasItemOnConveyorToTrack(self: pyaubo_sdk.MotionControl) -> bool
```

Lua 函数原型

```
hasItemOnConveyorToTrack() -> boolean
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.hasItemOnConveyorToTrack","params":[0],"id":↔1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":{"true}}
```

isBlending()

```
bool arcs::common_interface::MotionControl::isBlending ()
```

是否处交融区

返回

处交融区返回 true; 反之返回 false

异常

<i>arcs::common_interface::AuboException</i>	

Lua 函数原型

```
isBlending() -> bool
```

Lua 示例

```
status = isBlending()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.isBlending","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":false}
```

isConveyorTrackSync()

```
bool arcs::common_interface::MotionControl::isConveyorTrackSync (
    int encoder_id)
```

判断传送带与机械臂之间是否达到相对静止

参数

<i>encoder↔ _id</i>	预留

返回

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
isConveyorTrackSync(self: pyaubo_sdk.MotionControl, arg0: int) -> bool
```

Lua 函数原型

```
isConveyorTrackSync(encoder_id: number) -> bool
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.isConveyorTrackSync","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":false}
```

isJointSoftServoEnabled()

```
bool arcs::common_interface::MotionControl::isJointSoftServoEnabled ()
```

判断关节电流环刚度系数是否使能

返回

已使能返回 true, 反之则返回 false

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
isJointSoftServoEnabled(self: pyaubo_sdk.MotionControl) -> bool
```

Lua 函数原型

```
isJointSoftServoEnabled() -> boolean
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.isJointSoftServoEnabled","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":1}
```

isServoModeEnabled()

```
ARCS_DEPRECATED bool arcs::common_interface::MotionControl::isServoModeEnabled ()
```

判断伺服模式是否使能使用 getServoModeSelect 替代

返回

已使能返回 true, 反之则返回 false

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
isServoModeEnabled(self: pyaubo_sdk.MotionControl) -> bool
```

Lua 函数原型

```
isServoModeEnabled() -> boolean
```

Lua 示例

```
Servo_status = isServoModeEnabled()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.isServoModeEnabled","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":false}
```

isSpeedFractionCritical()

```
bool arcs::common_interface::MotionControl::isSpeedFractionCritical ()  
是否处于速度比例设置临界区
```

返回

处于速度比例设置临界区返回 true; 反之返回 false

异常

<code>arcs::common_interface::AuboException</code>

Lua 函数原型

```
isSpeedFractionCritical() -> bool
```

Lua 示例

```
status = isSpeedFractionCritical()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.isSpeedFractionCritical","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":true}
```

isSupportedTimeOptimal()

```
bool arcs::common_interface::MotionControl::isSupportedTimeOptimal ()  
获取是否支持时间最优算法: true - 支持 false - 不支持
```

返回

返回是否支持时间最优算法

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
isTimeOptimalEnabled(self: pyaubo_sdk.MotionControl) -> bool
```

Lua 函数原型

```
isSupportedTimeOptimal() -> boolean
```

Lua 示例

```
num = isSupportedTimeOptimal()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.isSupportedTimeOptimal","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":1.0}
```

isTimeOptimalEnabled()

```
bool arcs::common_interface::MotionControl::isTimeOptimalEnabled ()
```

获取时间最优算法状态: true - 开启 false - 关闭

返回

返回时间最优算法状态

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
isTimeOptimalEnabled(self: pyaubo_sdk.MotionControl) -> float
```

Lua 函数原型

```
isTimeOptimalEnabled() -> number
```

Lua 示例

```
num = isTimeOptimalEnabled()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.isTimeOptimalEnabled","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":1.0}
```

jointOffsetDisable()

```
int arcs::common_interface::MotionControl::jointOffsetDisable ()
```

关节偏移失能

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE AUBO_BUSY -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
jointOffsetDisable(self: pyaubo_sdk.MotionControl) -> int
```

Lua 函数原型

```
jointOffsetDisable() -> nil
```

Lua 示例

```
jointOffsetDisable()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.jointOffsetDisable","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

jointOffsetEnable()

```
int arcs::common_interface::MotionControl::jointOffsetEnable ()
```

关节偏移使能

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE AUBO_BUSY -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
jointOffsetEnable(self: pyaubo_sdk.MotionControl) -> int
```

Lua 函数原型

```
jointOffsetEnable() -> nil
```

Lua 示例

```
jointOffsetEnable()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.jointOffsetEnable","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

jointOffsetSet()

```
int arcs::common_interface::MotionControl::jointOffsetSet (
    const std::vector< double > & offset,
    int type = 1)
设置关节偏移
```

参数

<i>offset</i>	在各关节的位姿偏移
<i>type</i>	

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE AUBO_BUSY -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
jointOffsetSet(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: int) -> int
```

Lua 函数原型

```
jointOffsetSet(offset: table, type: number) -> nil
```

Lua 示例

```
jointOffsetSet({0.1,0,0,0,0},1)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.jointOffsetSet","params":[[0.1,0,0,0,0],1],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

moveCircle()

```
int arcs::common_interface::MotionControl::moveCircle (
    const std::vector< double > & via_pose,
    const std::vector< double > & end_pose,
    double a,
    double v,
    double blend_radius,
    double duration)
```

添加圆弧运动

参数

<i>via_pose</i>	圆弧运动途中点的位姿
<i>end_pose</i>	圆弧运动结束点的位姿
<i>a</i>	加速度 (如果 <i>via_pose</i> 与上一个路点位置变化小于 1mm, 姿态变化大于 1e-4 rad, 此加速度会被作为角加速度, 单位 rad/s^2. 否则为线加速度, 单位 m/s^2)
<i>v</i>	速度 (如果 <i>via_pose</i> 与上一个路点位置变化小于 1mm, 姿态变化大于 1e-4 rad, 此速度会被作为角速度, 单位 rad / s. 否则为线速度, 单位 m / s)
<i>blend_radius</i>	交融半径, 单位: m
<i>duration</i>	运行时间, 单位: s 通常当给定了速度和加速度, 便能够确定轨迹的运行时间。如果想延长轨迹的运行时间, 便要设置 <i>duration</i> 这个参数。 <i>duration</i> 可以延长轨迹运动的时间, 但是不能缩短轨迹时间。 当 <i>duration</i> = 0 的时候, 表示不指定运行时间, 即没有明确完成运动的时间, 将根据速度与加速度计算运行时间。如果 <i>duration</i> 不等于 0, <i>a</i> 和 <i>v</i> 的值将被忽略。

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE AUBO_BUSY -AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

arcs::common_interface::AuboException

Python 函数原型

```
moveCircle(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: List[float], arg2: float, arg3: float, arg4: float, arg5: float) -> int
```

Lua 函数原型

```
moveCircle(via_pose: table, end_pose: table, a: number, v: number, blend_radius: number, duration: number) -> nil
```

Lua 示例

```
moveCircle({0.440164,-0.00249391,0.398658,2.45651,0,1.5708},{0.440164,0.166256,0.297599,2.45651,0,1.5708},1.2,0.25,0)
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.MotionControl.moveCircle", "params": [[0.440164,-0.00249391,0.398658,2.45651,0,1.5708],[0.440164,0.166256,0.297599,2.45651,0,1.5708],1.2,0.25,0,0], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

moveCircle2()

```
int arcs::common_interface::MotionControl::moveCircle2 (
    const CircleParameters & param)
```

高级圆弧或者圆周运动

参数

<i>param</i>	圆周运动参数

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE AUBO_BUSY -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

arcs::common_interface::AuboException

Python 函数原型

```
moveCircle2(self: pyaubo_sdk.MotionControl, arg0: arcs::common\_interface::CircleParameters) -> int
```

Lua 函数原型

```
moveCircle2(param: table) -> nil
```

Lua 示例

```
moveCircle2({0.440164,-0.00249391,0.398658,2.45651,0,1.5708},{0.440164,0.166256,0.297599,2.45651,0,1.5708},1.2,0.25,0,0,0,0,0)
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.MotionControl.moveCircle2", "params": [{"pose_via": [0.440164,-0.00249391,0.398658,2.45651,0,1.5708], "pose_to": [0.440164,0.166256,0.297599,2.45651,0,1.5708], "a": 1.2, "v": 0.25, "blend_radius": 0, "duration": 0, "helix": 0, "spiral": 0, "direction": 0, "loop_times": 0}], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

moveCircleWithAxisGroup()

```
int arcs::common_interface::MotionControl::moveCircleWithAxisGroup (
    const std::vector< double > & via_pose,
    const std::vector< double > & end_pose,
    double a,
    double v,
    double blend_radius,
    double duration,
    const std::string & group_name,
    const std::vector< double > & extq)
```

moveCircle 与外部轴同步运动

参数

<i>group_name</i>	外部轴组名称
<i>via_pose</i>	圆弧运动途中点的位姿
<i>end_pose</i>	圆弧运动结束点的位姿
<i>a</i>	加速度
<i>v</i>	速度
<i>blend_radius</i>	交融半径
<i>duration</i>	运行时间

返回

moveIntersection()

```
int arcs::common_interface::MotionControl::moveIntersection (
    const std::vector< std::vector< double > > & poses,
    double a,
    double v,
    double main_pipe_radius,
    double sub_pipe_radius,
    double normal_distance,
    double normal_alpha)
```

相贯线接口

参数

<i>pose</i>	由三个示教位姿组成 (首先需要运动到起始点, 起始点的要求:过主圆柱柱体轴心且与子圆柱体轴线平行的平面与子圆柱体在底部的交点) p1:过子圆柱体轴线且与大圆柱体轴线平行的平面, 与小圆柱体在左侧顶部的交点 p2:过子圆柱体轴线且与大圆柱体轴线平行的平面, 与大圆柱体在左侧底部的交点 p3:过子圆柱体轴线且与大圆柱体轴线平行的平面, 与大圆柱体在右侧底部的交点
<i>v</i>	速度
<i>a</i>	加速度
<i>main_pipe_radius</i>	主圆柱体半径
<i>sub_pipe_radius</i>	子圆柱体半径
<i>normal_distance</i>	两圆柱体轴线距离

<i>normal_alpha</i>	两圆柱体轴线的夹角

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
moveIntersection(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: float, arg2: float, arg3: float, arg4: float, arg5: float, arg6: float) -> int
```

Lua 函数原型

```
moveIntersection(poses: table, a: number, v: number, main_pipe_radius: number, sub_pipe_radius: number, normal_distance: number, normal_alpha: number) -> nil
```

moveJoint()

```
int arcs::common_interface::MotionControl::moveJoint (
    const std::vector< double > & q,
    double a,
    double v,
    double blend_radius,
    double duration)
```

添加关节运动

参数

<i>q</i>	关节角, 单位 rad
<i>a</i>	加速度, 单位 rad/s ² , 最大值可通过RobotConfig 类中的接口 getJointMaxAccelerations() 来获取
<i>v</i>	速度, 单位 rad/s, 最大值可通过RobotConfig 类中的接口 getJointMaxSpeeds() 来获取
<i>blend_radius</i>	交融半径, 单位 m
<i>duration</i>	运行时间, 单位 s 通常当给定了速度和加速度, 便能够确定轨迹的运行时间。如果想延长轨迹的运行时间, 便要设置 duration 这个参数。duration 可以延长轨迹运动的时间, 但是不能缩短轨迹时间。 当 duration = 0 的时候, 表示不指定运行时间, 即没有明确完成运动的时间, 将根据速度与加速度计算运行时间。如果 duration 不等于 0, a 和 v 的值将被忽略。

返回值

<i>0</i>	成功
<i>AUBO_BAD_STATE(1)</i>	当前安全模式处于非 Normal、ReducedMode、Recovery 状态
<i>AUBO_QUEUE_FULL(2)</i>	规划队列已满
<i>AUBO_BUSY(3)</i>	上一条指令正在执行中
<i>-AUBO_BAD_STATE(-1)</i>	可能的原因包括但不限于：线程已分离、线程被终止、task_id 未找到，或者当前机器人模式非 Running
<i>-AUBO_TIMEOUT(-4)</i>	调用接口超时
<i>-AUBO_INVL_ARGUMENT(-5)</i>	参数数组 q 的长度小于当前机器臂的自由度
<i>AUBO_REQUEST_IGNORE(13)</i>	参数数组 q 的长度太短且无需在该点停留

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
moveJoint(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: float, arg2: float, arg3: float,
arg4: float) -> int
```

Lua 函数原型

```
moveJoint(q: table, a: number, v: number, blend_radius: number, duration: number) -> nil
```

Lua 示例

```
moveJoint({-2.05177,-0.400292, 1.19625, 0.0285152, 1.57033, -2.28774},0.3,0.3,0,0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.moveJoint","params":[[-2.05177, -0.400292, 1.19625,
0.0285152, 1.57033, -2.28774],0.3,0.3,0,0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

moveJointWithAxisGroup()

```
int arcs::common_interface::MotionControl::moveJointWithAxisGroup (
    const std::vector< double > & q,
    double a,
    double v,
    double blend_radius,
    double duration,
    const std::string & group_name,
    const std::vector< double > & extq)
```

机器人与外部轴同步运动

参数

<i>group_name</i>	
<i>q</i>	
<i>a</i>	
<i>v</i>	
<i>blend_radius</i>	
<i>duration</i>	

返回

moveLine()

```
int arcs::common_interface::MotionControl::moveLine (
    const std::vector< double > & pose,
    double a,
    double v,
    double blend_radius,
    double duration)
```

添加直线运动

参数

<i>pose</i>	目标位姿
<i>a</i>	加速度 (如果位置变化小于 1mm, 姿态变化大于 1e-4 rad, 此加速度会被作为角加速度, 单位 rad/s^2. 否则为线加速度, 单位 m/s^2) 最大值可通过RobotConfig 类中的接口 getTcpMaxAccelerations() 来获取
<i>v</i>	速度 (如果位置变化小于 1mm, 姿态变化大于 1e-4 rad, 此速度会被作为角速度, 单位 rad/s. 否则为线速度, 单位 m/s) 最大值可通过RobotConfig 类中的接口 getTcpMaxSpeeds() 来获取
<i>blend_radius</i>	交融半径, 单位 m, 值介于 0.001 和 1 之间
<i>duration</i>	运行时间, 单位 s 通常当给定了速度和加速度, 便能够确定轨迹的运行时间。如果想延长轨迹的运行时间, 便要设置 duration 这个参数。duration 可以延长轨迹运动的时间, 但是不能缩短轨迹时间。 当 duration = 0 的时候, 表示不指定运行时间, 即没有明确完成运动的时间, 将根据速度与加速度计算运行时间。如果 duration 不等于 0, a 和 v 的值将被忽略。

返回值

0	成功
AUBO_BAD_STATE(1)	当前安全模式处于非 Normal、ReducedMode、Recovery 状态
AUBO_QUEUE_FULL(2)	轨迹队列已满
AUBO_BUSY(3)	上一条指令正在执行中

<code>-AUBO_BAD_STATE(-1)</code>	可能的原因包括但不限于：线程已分离、线程被终止、task_id 未找到，或者当前机器人模式非 Running
<code>-AUBO_TIMEOUT(-4)</code>	调用接口超时
<code>-AUBO_INVL_ARGUMENT(-5)</code>	参数数组 pose 的长度小于当前机器臂的自由度
<code>AUBO_REQUEST_IGNORE(13)</code>	参数数组 pose 路径长度太短且无需在该点停留

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
moveLine(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: float, arg2: float, arg3: float,
arg4: float) -> int
```

Lua 函数原型

```
moveLine(pose: table, a: number, v: number, blend_radius: number, duration: number) -> nil
```

Lua 示例

```
moveLine({0.58815,0.0532,0.62391,2.46,0.479,1.619},1.3,1.0,0)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.moveLine", "params": [[0.58815,0.0532,0.62391,2.46,0.479,1.619],1.2,0.25,0,0], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

moveLineWithAxisGroup()

```
int arcs::common_interface::MotionControl::moveLineWithAxisGroup (
    const std::vector< double > & pose,
    double a,
    double v,
    double blend_radius,
    double duration,
    const std::string & group_name,
    const std::vector< double > & extq)
```

直线运动与外部轴同步运动

参数

<code>group_name</code>	外部轴组名称
<code>pose</code>	目标位姿
<code>a</code>	加速度

<i>v</i>	速度
<i>blend_radius</i>	交融半径
<i>duration</i>	运行时间

返回

movePathBuffer()

```
int arcs::common_interface::MotionControl::movePathBuffer (
    const std::string & name)
    执行缓存的路径
```

参数

<i>name</i>	缓存路径的名字

返回

成功返回 0；失败返回错误码 -AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
movePathBuffer(self: pyaubo_sdk.MotionControl, arg0: str) -> int
```

Lua 函数原型

```
movePathBuffer(name: string) -> nil
```

Lua 示例

```
movePathBuffer("traje_name")
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.movePathBuffer","params":["rec"],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

moveProcess()

```
int arcs::common_interface::MotionControl::moveProcess (
    const std::vector< double > & pose,
    double a,
    double v,
    double blend_radius)
    添加工艺运动
```

参数

<i>pose</i>	
<i>a</i>	
<i>v</i>	
<i>blend_radius</i>	

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE AUBO_BUSY -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Lua 函数原型

```
moveProcess(pose: table, a: number, v: number, blend_radius: number, duration: number) -> nil
```

Lua 示例

```
moveProcess({0.58815,0.0532,0.62391,2.46,0.479,1.619},1.2,0.25,0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.moveProcess","params":[[0.58815,0.0532,0.62391,2.46,0.479,1.619],1.2,0.25,0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

moveSpiral()

```
int arcs::common_interface::MotionControl::moveSpiral (
    const SpiralParameters & param,
    double blend_radius,
    double v,
    double a,
    double t)
```

螺旋线运动

参数

<i>param</i>	封装的参数
<i>blend_radius</i>	
<i>v</i>	
<i>a</i>	

t	

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>	

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.MotionControl.moveSpiral", "params": [ { "spiral": 0.005, "helix": 0.005, "angle": 18.84, "plane": 0, "frame": [ 0, 0, 0, 0, 0, 0 ], 0, 0.25, 1.2, 0 }, "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

moveSpline()

```
int arcs::common_interface::MotionControl::moveSpline (
    const std::vector< double > & q,
    double a,
    double v,
    double duration)
```

在关节空间做样条插值

参数

q	关节角度, 如果传入参数维度为 0, 表示样条运动结束
a	加速度, 单位 rad/s^2 , 最大值可通过RobotConfig 类中的接口 <code>getJointMaxAccelerations()</code> 来获取
v	速度, 单位 rad/s , 最大值可通过RobotConfig 类中的接口 <code>getJointMaxSpeeds()</code> 来获取
$duration$	

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
moveSpline(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: float, arg2: float, arg3: float)
-> int
```

Lua 函数原型

```
moveSpline(q: table, a: number, v: number, duration: number) -> nil
```

Lua 示例

```
moveSpline({-2.05177,-0.400292, 1.19625, 0.0285152, 1.57033, -2.28774},1.3,1.0,0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.moveSpline","params":[[0,0,0,0,0,0],1,1,0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

pathBufferAlloc()

```
int arcs::common_interface::MotionControl::pathBufferAlloc (
    const std::string & name,
    int type,
    int size)
```

新建一个路径点缓存

参数

<i>name</i>	指定路径的名字
<i>type</i>	路径的类型 1: toppra 时间最优路径规划 2: cubic_spline(录制的轨迹) 3: 关节B 样条插值, 最少三个点
<i>size</i>	缓存区大小

返回

新建成功返回 AUBO_OK(0)

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
pathBufferAlloc(self: pyaubo_sdk.MotionControl, arg0: str, arg1: int, arg2: int) -> int
```

Lua 函数原型

```
pathBufferAlloc(name: string, type: number, size: number) -> nil
```

Lua 示例

```
pathBufferAlloc("traje_name",5,100)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.pathBufferAlloc","params":["rec",2,3],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

pathBufferAppend()

```
int arcs::common_interface::MotionControl::pathBufferAppend (
    const std::string & name,
    const std::vector< std::vector< double > > & waypoints)
```

向路径缓存添加路点

参数

<i>name</i>	路径缓存的名字
<i>waypoints</i>	路点

返回

成功返回 0; 失败返回错误码 -AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
pathBufferAppend(self: pyaubo_sdk.MotionControl, arg0: str, arg1: List[List[float]]) -> int
```

Lua 函数原型

```
pathBufferAppend(name: string, waypoints: table) -> nil
```

Lua 示例

```
pathBufferAppend("traje_name",{ {-0.000000,0.000009,-0.000001,0.000002,0.000002,0.000000}, {-0.000001,0.000010,-0.000002,0.000000,0.000003,0.000002}})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.pathBufferAppend","params":["rec",[[-0.000000,0.000009,-0.000001,0.000002,0.000002,0.000000],[-0.000001,0.000010,-0.000002,0.000000,0.000003,0.000002]]],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

pathBufferEval()

```
int arcs::common_interface::MotionControl::pathBufferEval (
    const std::string & name,
    const std::vector< double > & a,
    const std::vector< double > & v,
    double t)
```

计算、优化等耗时操作，传入的参数相同时不会重新计算

参数

<i>name</i>	通过 pathBufferAlloc 新建的路径点缓存的名字
<i>a</i>	关节加速度限制, 需要和机器人自由度大小相同, 单位 rad/s ²
<i>v</i>	关节速度限制, 需要和机器人自由度大小相同, 单位 rad/s
<i>t</i>	时间 pathBufferAlloc 这个接口分配内存的时候要指定类型, 根据 pathBufferAlloc 这个接口的类型: 类型为 1 时, 表示运动持续时间 类型为 2 时, 表示采样时间间隔 类型为 3 时: <i>t</i> 表示运动持续时间 若 <i>t</i> =0, 则由使用软件内部默认的时间 (推荐使用) 若 <i>t</i> !=0, <i>t</i> 需要设置为 路径点数 * 相邻点时间间隔 (points * interval, interval >= 0.01)

返回

成功返回 0; 失败返回错误码 -AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
pathBufferEval(self: pyaubo_sdk.MotionControl, arg0: str, arg1: List[float], arg2: List[float]↵
, arg3: float) -> int
```

Lua 函数原型

```
pathBufferEval(name: string, a: table, v: table, t: number) -> nil
```

Lua 示例

```
pathBufferEval("traje_name",{1,1,1,1,1,1},{1,1,1,1,1,1},0.02)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.pathBufferEval","params":["rec",[1,1,1,1,1,1],[1,1,1,1,1,1]↵
,0.02],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```


pathBufferFilter()

```
int arcs::common_interface::MotionControl::pathBufferFilter (
    const std::string & name,
    int order,
    double fd,
    double fs)
```

关节空间路径滤波器

pathBufferFilter

参数

<i>name</i>	缓存路径的名称
<i>order</i>	滤波器阶数 (一般取 2)
<i>fd</i>	截止频率, 越小越光滑, 但是延迟会大 (一般取 3-20)
<i>fs</i>	离散数据的采样频率 (一般取 20-500)

返回

成功返回 0

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
pathBufferFilter(self: pyaubo_sdk.MotionControl, arg0: str, arg1: int, arg2: float, arg3: float) ->
int
```

Lua 函数原型

```
pathBufferFilter(name: string, order: number, fd: number, fs:number) -> nil
```

Lua 示例

```
pathBufferFilter("traje_name",2,5,125)
```

pathBufferFree()

```
int arcs::common_interface::MotionControl::pathBufferFree (
    const std::string & name)
```

释放路径缓存

参数

<i>name</i>	缓存路径的名字

返回

成功返回 0; 失败返回错误码 -AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
pathBufferFree(self: pyaubo_sdk.MotionControl, arg0: str) -> int
```

Lua 函数原型

```
pathBufferFree(name: string) -> nil
```

Lua 示例

```
pathBufferFree("traje_name")
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.pathBufferFree","params":["rec"],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":-5}
```

pathBufferList()

```
std::vector< std::string > arcs::common_interface::MotionControl::pathBufferList ()
```

列出所有缓存路径的名字

返回

返回所有缓存路径的名字

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
pathBufferList(self: pyaubo_sdk.MotionControl) -> List[str]
```

Lua 函数原型

```
pathBufferList() -> table
```

Lua 示例

```
Buffer_table = pathBufferList()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.pathBufferList","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[]}
```

pathBufferValid()

```
bool arcs::common_interface::MotionControl::pathBufferValid (
    const std::string & name)
```

指定名字的 buffer 是否有效

buffer 需要满足三个条件有效:

- 1、buffer 存在, 已经分配过内存
- 2、传进 buffer 的点要大于等于 buffer 的大小
- 3、要执行一次 pathBufferEval

参数

<i>name</i>	buffer 的名字

返回

有效返回 true; 无效返回 false

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
pathBufferValid(self: pyaubo_sdk.MotionControl, arg0: str) -> bool
```

Lua 函数原型

```
pathBufferValid(name: string) -> boolean
```

Lua 示例

```
buffer_status = pathBufferValid("traje_name")
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.pathBufferValid", "params": ["rec"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": false}
```

pathOffsetCoordinate()

```
int arcs::common_interface::MotionControl::pathOffsetCoordinate (
    int ref_coord)
```

设置偏移的参考坐标系仅对 pathOffsetSet 中 type=1 有效

参数

<i>ref_coord</i>	参考坐标系 0-基坐标系 1-TCP

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
pathOffsetCoordinate(self: pyaubo_sdk.MotionControl, arg0: int) -> float
```

Lua 函数原型

```
pathOffsetCoordinate(ref_coord: number) -> number
```

Lua 示例

```
num = pathOffsetCoordinate(0)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.pathOffsetCoordinate", "params": [0], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

pathOffsetDisable()

```
int arcs::common_interface::MotionControl::pathOffsetDisable ()
```

路径偏移失能

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE AUBO_BUSY -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
pathOffsetDisable(self: pyaubo_sdk.MotionControl) -> int
```

Lua 函数原型

```
pathOffsetDisable() -> nil
```

Lua 示例

```
pathOffsetDisable()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.pathOffsetDisable", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

pathOffsetEnable()

```
int arcs::common_interface::MotionControl::pathOffsetEnable ()
    路径偏移使能
```

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
pathOffsetEnable(self: pyaubo_sdk.MotionControl) -> int
```

Lua 函数原型

```
pathOffsetEnable() -> number
```

Lua 示例

```
num = pathOffsetEnable()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.pathOffsetEnable","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

pathOffsetLimits()

```
int arcs::common_interface::MotionControl::pathOffsetLimits (
    double v,
    double a)
```

设置偏移的最大速度和最大加速度仅对 pathOffsetSet 中 type=1 有效

参数

<i>v</i>	最大速度
<i>a</i>	最大加速度

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
pathOffsetLimits(self: pyaubo_sdk.MotionControl, arg0: float, arg1: float) -> int
```

Lua 函数原型

```
pathOffsetLimits(v: number, a: number) -> nil
```

Lua 示例

```
pathOffsetLimits(1.5,2.5)
```

pathOffsetSet()

```
int arcs::common_interface::MotionControl::pathOffsetSet (
    const std::vector< double > & offset,
    int type = 0)
```

设置路径偏移

参数

<i>offset</i>	在各方向的位姿偏移
<i>type</i>	运动类型 0-位置规划 1-速度规划

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE AUBO_BUSY -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
pathOffsetSet(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: int) -> int
```

Lua 函数原型

```
pathOffsetSet(offset: table, type: number) -> nil
```

Lua 示例

```
pathOffsetSet({ 0, 0, 0.1, 0, 0, 0 }, 0)
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.MotionControl.pathOffsetSet", "params": [[0,0,0.01,0,0,0],0], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

pathOffsetSupv()

```
int arcs::common_interface::MotionControl::pathOffsetSupv (
    const std::vector< double > & min,
    const std::vector< double > & max,
    int strategy)
```

监控轨迹偏移范围

参数

<i>min</i>	沿坐标轴负方向最大偏移量		
<i>max</i>	沿坐标轴正方向最大偏移量		
<i>strategy</i>	达到最大偏移量后监控策略 2-保护停止	0-禁用监控	1-饱和限制，即维持最大姿态

返回

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
pathOffsetSupv(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: List[float], arg2: int) -> int
```

Lua 函数原型

```
pathOffsetSupv(min: table, max: table, strategy: number) -> number
```

Lua 示例

```
num = pathOffsetSupv({0,0,-0.2,0,0,0},{0,0,0.5,0,0,0},0)
```

resetTcpMaxLinearVelocity()

```
int arcs::common_interface::MotionControl::resetTcpMaxLinearVelocity ()
```

重置 TCP 最大线速度

清除通过 setTcpMaxLinearVelocity 接口设置的 TCP 最大线速度覆盖值

返回

成功返回 0；失败返回错误码

异常

<code>arcs::common_interface::AuboException</code>

Lua 函数原型

`resetTcpMaxLinearVelocity()` -> number

Lua 示例

`resetTcpMaxLinearVelocity()` – 重置 TCP 最大线速度为安全配置中的值

Python 函数原型

`resetTcpMaxLinearVelocity()` -> int

Python 示例

`resetTcpMaxLinearVelocity()` – 重置 TCP 最大线速度为安全配置中的值

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.MotionControl.resetTcpMaxLinearVelocity", "params": [], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

restoPath()

```
int arcs::common_interface::MotionControl::restoPath ()
restoPath
```

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE AUBO_BUSY -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Lua 函数原型

`restoPath()` -> number

Lua 示例

num = `restoPath()`

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.MotionControl.restoPath", "params": [], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```


resumeMoveJoint()

```
int arcs::common_interface::MotionControl::resumeMoveJoint (
    const std::vector< double > & q,
    double a,
    double v,
    double duration)
```

通过关节运动移动到暂停点的位置

参数

<i>q</i>	关节角, 单位 rad
<i>a</i>	加速度, 单位 rad/s ²
<i>v</i>	速度, 单位 rad/s
<i>duration</i>	运行时间, 单位 s 通常当给定了速度和加速度, 便能够确定轨迹的运行时间。如果想延长轨迹的运行时间, 便要设置 duration 这个参数。duration 可以延长轨迹运动的时间, 但是不能缩短轨迹时间。当 duration = 0 的时候, 表示不指定运行时间, 即没有明确完成运动的时间, 将根据速度与加速度计算运行时间。如果 duration 不等于 0, a 和 v 的值将被忽略。

返回

成功返回 0; 失败返回错误码 -AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
resumeMoveJoint(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: float, arg2: float, arg3: float) -> int
```

Lua 函数原型

```
resumeMoveJoint(q: table, a: number, v: number, duration: number) -> nil
```

Lua 示例

```
resumeMoveJoint({-2.05177,-0.400292, 1.19625, 0.0285152, 1.57033, -2.28774},1.3,1.0,0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.resumeMoveJoint","params":[[-2.05177, -0.400292, 1.19625, 0.0285152, 1.57033, -2.28774],0.3,0.3,0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":-1}
```

resumeMoveLine()

```
int arcs::common_interface::MotionControl::resumeMoveLine (
    const std::vector< double > & pose,
    double a,
    double v,
    double duration)
```

通过直线运动移动到暂停点的位置

参数

<i>pose</i>	目标位姿
<i>a</i>	加速度, 单位 m/s^2
<i>v</i>	速度, 单位 m/s
<i>duration</i>	运行时间, 单位 s 通常当给定了速度和加速度, 便能够确定轨迹的运行时间。如果想延长轨迹的运行时间, 便要设置 <i>duration</i> 这个参数。duration 可以延长轨迹运动的时间, 但是不能缩短轨迹时间。当 <i>duration</i> = 0 的时候, 表示不指定运行时间, 即没有明确完成运动的时间, 将根据速度与加速度计算运行时间。如果 <i>duration</i> 不等于 0, <i>a</i> 和 <i>v</i> 的值将被忽略。

返回

成功返回 0; 失败返回错误码 -AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
resumeMoveLine(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: float, arg2: float, arg3: float) -> int
```

Lua 函数原型

```
resumeMoveLine(pose: table, a: number, v: number,duration: number) -> nil
```

Lua 示例

```
resumeMoveLine({0.58815,0.0532,0.62391,2.46,0.479,1.619},1.2,0.25,0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.resumeMoveLine","params":[[0.58815,0.0532,0.62391,2.46,0.479,1.619],1.2,0.25,0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

resumeSpeedJoint()

```
int arcs::common_interface::MotionControl::resumeSpeedJoint (
    const std::vector< double > & qd,
    double a,
    double t)
```

关节空间速度跟随 (机械臂运行工程时发生碰撞, 通过此接口移动到安全位置)

当机械臂还没达到目标速度的时候, 给一个新的目标速度, 机械臂会立刻达到新的目标速度

参数

<i>qd</i>	目标关节速度, 单位 rad/s
<i>a</i>	主轴的加速度, 单位 rad/s ²
<i>t</i>	函数返回所需要的时间, 单位 s 如果 $t = 0$, 当达到目标速度的时候, 函数将返回; 反之, 则经过 t 时间后, 函数返回, 不管是否达到目标速度。如果没有达到目标速度, 会减速到零。如果达到了目标速度就是按照目标速度匀速运动。

返回

成功返回 0; 失败返回错误码 AUBO_BUSY-AUBO_INVL_ARGUMENT-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
resumeSpeedJoint(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: float, arg2: float) ->
int
```

Lua 函数原型

```
resumeSpeedJoint(q: table, a: number, t: number) -> nil
```

Lua 示例

```
resumeSpeedJoint({0.2,0,0,0,0,0},1.5,10)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.resumeSpeedJoint","params":[[0.2,0,0,0,0,0],1.5,100]←
,"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":-1}
```

resumeSpeedLine()

```
int arcs::common_interface::MotionControl::resumeSpeedLine (
    const std::vector< double > & xd,
    double a,
    double t)
```

笛卡尔空间速度跟随 (机械臂运行工程时发生碰撞, 通过此接口移动到安全位置)

当机械臂还没达到目标速度的时候, 给一个新的目标速度, 机械臂会立刻达到新的目标速度

参数

<i>xd</i>	工具速度, 单位 m/s
<i>a</i>	工具位置加速度, 单位 m/s ²
<i>t</i>	函数返回所需要的时间, 单位 s 如果 $t = 0$, 当达到目标速度的时候, 函数将返回; 反之, 则经过 t 时间后, 函数返回, 不管是否达到目标速度。如果没有达到目标速度, 会减速到零。如果达到了目标速度就是按照目标速度匀速运动。

返回

成功返回 0; 失败返回错误码 -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
resumeSpeedLine(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: float, arg2: float) -> int
```

Lua 函数原型

```
resumeSpeedLine(pose: table, a: number, t: number) -> nil
```

Lua 示例

```
resumeSpeedLine({0.25,0,0,0,0,0},1.2,100)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.resumeSpeedLine","params":[[0.25,0,0,0,0,0],1.2,100]↵
,"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":-1}
```

resumeStopJoint()

```
int arcs::common_interface::MotionControl::resumeStopJoint (
    double acc)
```

关节空间停止运动 (机械臂运行工程时发生碰撞, 通过 resumeSpeedJoint 接口移动到安全位置后需要停止时调用此接口)

参数

<i>acc</i>	关节加速度, 单位: rad/s ²

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
resumeStopJoint(self: pyaubo_sdk.MotionControl, arg0: float) -> int
```

Lua 函数原型

```
resumeStopJoint(acc: number) -> nil
```

Lua 示例

```
resumeStopJoint(31)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.resumeStopJoint","params":[31],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":-1}
```

resumeStopLine()

```
int arcs::common_interface::MotionControl::resumeStopLine (
    double acc,
    double acc_rot)
```

笛卡尔空间停止运动 (机械臂运行工程时发生碰撞, 通过 resumeSpeedLine 接口移动到安全位置后需要停止时调用此接口)

参数

<code>acc</code>	位置加速度, 单位: m/s^2
<code>acc_rot</code>	姿态加速度, 单位: rad/s^2

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
resumeStopLine(self: pyaubo_sdk.MotionControl, arg0: float, arg1: float) -> int
```

Lua 函数原型

```
resumeStopLine(acc: number, acc_rot: number) -> nil
```

Lua 示例

```
resumeStopLine(10,10)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.resumeStopLine", "params": [10, 10], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": -1}
```

servoCartesian()

```
int arcs::common_interface::MotionControl::servoCartesian (
    const std::vector< double > & pose,
    double a,
    double v,
    double t,
    double lookahead_time,
    double gain)
```

笛卡尔空间伺服

目前可用参数只有 pose 和 t;

参数

<i>pose</i>	位姿, 单位 m,
<i>a</i>	加速度, 单位 m/s ² ,
<i>v</i>	速度, 单位 m/s,
<i>t</i>	运行时间, 单位 s t 值越大, 机器臂运动越慢, 反之, 运动越快; 该参数最优值为连续调用 servoCartesian 接口的间隔时间。
<i>lookahead_time</i>	前瞻时间, 单位 s 指定机器臂开始减速前要运动的时长, 用前瞻时间来平滑轨迹[0.03, 0.2], 当 lookahead_time 小于一个控制周期时, 越小则超调量越大, 该参数最优值为一个控制周期。
<i>gain</i>	比例增益跟踪目标位置的比例增益[100, 200], 用于控制运动的顺滑性和精度, 比例增益越大, 到达目标位置的时间越长, 超调量越小。

返回值

0	成功
<i>AUBO_BAD_STATE(1)</i>	当前安全模式处于非 Normal、ReducedMode、Recovery 状态
<i>AUBO_QUEUE_FULL(2)</i>	规划队列已满
<i>AUBO_BUSY(3)</i>	上一条指令正在执行中

<code>-AUBO_BAD_STATE(-1)</code>	可能的原因包括但不限于：线程已分离、线程被终止、 <code>task_id</code> 未找到，或者当前机器人模式非 <code>Running</code>
<code>-AUBO_TIMEOUT(-4)</code>	调用接口超时
<code>-AUBO_INVL_ARGUMENT(-5)</code>	轨迹位置超限或速度超限
<code>-AUBO_REQUEST_IGNORE(-13)</code>	当前处于非 <code>servo</code> 模式
<code>-AUBO_IK_NO_CONVERGE(-23)</code>	逆解计算不收敛，计算出错；
<code>-AUBO_IK_OUT_OF_RANGE(-24)</code>	逆解计算超出机器人最大限制；
<code>AUBO_IK_CONFIG_DISMATCH(-25)</code>	逆解输入配置存在错误；
<code>-AUBO_IK_JACOBIAN_FAILED(-26)</code>	逆解雅可比矩阵计算失败；
<code>-AUBO_IK_NO_SOLU(-27)</code>	目标点存在解析解，但均不满足选解条件；
<code>-AUBO_IK_UNKOWN_ERROR(-28)</code>	逆解返回未知类型错误；

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
servoCartesian(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: float, arg2: float, arg3: float, arg4: float, arg5: float) -> int
```

Lua 函数原型

```
servoCartesian(pose: table, a: number, v: number, t: number, lookahead_time: number, gain: number) -> nil
```

Lua 示例

```
servoCartesian({0.58712,-0.15775,0.48703,2.76,0.344,1.432},0,0,10,0,0)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.servoCartesian", "params": [[0.58712,-0.15775,0.48703,2.76,0.344,1.432],0,0,10,0,0], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": -13}
```

servoCartesianWithAxes()

```
int arcs::common_interface::MotionControl::servoCartesianWithAxes (
    const std::vector< double > & pose,
    const std::vector< double > & extq,
    double a,
    double v,
    double t,
    double lookahead_time,
    double gain)
```

伺服运动（带外部轴），用于执行离线轨迹、透传用户规划轨迹等与 `servoJointWithAxes` 区别在于接收笛卡尔空间位姿而不是关节角度（由软件内部直接做逆解）

参数

<i>pose</i>	
<i>extq</i>	
<i>t</i>	
<i>smooth_scale</i>	
<i>delay_sacle</i>	

返回

servoJoint()

```
int arcs::common_interface::MotionControl::servoJoint (
    const std::vector< double > & q,
    double a,
    double v,
    double t,
    double lookahead_time,
    double gain)
```

关节空间伺服

目前可用参数只有 q 和 t;

参数

<i>q</i>	关节角, 单位 rad,
<i>a</i>	加速度, 单位 rad/s ² ,
<i>v</i>	速度, 单位 rad/s,
<i>t</i>	运行时间, 单位 s t 值越大, 机器臂运动越慢, 反之, 运动越快; 该参数最优值为连续调用 servoJoint 接口的间隔时间。
<i>lookahead_time</i>	前瞻时间, 单位 s 指定机器臂开始减速前要运动的时长, 用前瞻时间来平滑轨迹[0.03, 0.2], 当 lookahead_time 小于一个控制周期时, 越小则超调量越大, 该参数最优值为一个控制周期。
<i>gain</i>	比例增益跟踪目标位置的比例增益[100, 200], 用于控制运动的顺滑性和精度, 比例增益越大, 到达目标位置的时间越长, 超调量越小。

返回值

<i>0</i>	成功
<i>AUBO_BAD_STATE(1)</i>	当前安全模式处于非 Normal、ReducedMode、Recovery 状态
<i>AUBO_QUEUE_FULL(2)</i>	规划队列已满
<i>AUBO_BUSY(3)</i>	上一条指令正在执行中

<code>-AUBO_BAD_STATE(-1)</code>	可能的原因包括但不限于：线程已分离、线程被终止、task_id 未找到，或者当前机器人模式非 Running
<code>-AUBO_TIMEOUT(-4)</code>	调用接口超时
<code>-AUBO_INVL_ARGUMENT(-5)</code>	轨迹位置超限、速度超限或相邻点角度差超限 ($\pm 15^\circ$)
<code>-AUBO_REQUEST_IGNORE(-13)</code>	当前处于非 servo 模式
<code>-AUBO_POS_BOUND_ERR(-18)</code>	关节位置超出机器人最大限制
<code>-AUBO_TRAJ_SELF_COLLISION(-22)</code>	轨迹存在自碰撞风险
<code>-AUBO_ALGORITHM_PLAN_FAILED(-30)</code>	轨迹规划失败

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
servoJoint(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: float, arg2: float, arg3: float,
arg4: float, arg5: float) -> int
```

Lua 函数原型

```
servoJoint(q: table, a: number, v: number, t: number, lookahead_time: number, gain: number)
-> nil
```

Lua 示例

```
servoJoint({0,0,0,0,0,0},0,0,10,0,0)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.servoJoint", "params": [[0,0,0,0,0,0],0,0,10,0,0], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": -13}
```

servoJointWithAxes()

```
int arcs::common_interface::MotionControl::servoJointWithAxes (
    const std::vector< double > & q,
    const std::vector< double > & extq,
    double a,
    double v,
    double t,
    double lookahead_time,
    double gain)
```

伺服运动（带外部轴），用于执行离线轨迹、透传用户规划轨迹等

参数

q	
$extq$	
t	
$smooth\_scale$	
$delay\_scale$	

返回

setCirclePathMode()

```
int arcs::common_interface::MotionControl::setCirclePathMode (
    int mode)
    设置圆弧路径模式
```

参数

$mode$	模式 0:工具姿态相对于圆弧路径点坐标系保持不变 1:姿态线性变化, 绕着空间定轴转动, 从起始点姿态变化到目标点姿态 2:从起点姿态开始经过中间点姿态, 变化到目标点姿态

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE AUBO_BUSY -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setCirclePathMode(self: pyaubo_sdk.MotionControl, arg0: int) -> int
```

Lua 函数原型

```
setCirclePathMode(mode: number) -> nil
```

Lua 示例

```
setCirclePathMode(1)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.setCirclePathMode","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setConveyorTrackCompensate()

```
int arcs::common_interface::MotionControl::setConveyorTrackCompensate (
    int encoder_id,
    double comp)
    设置传送带跟踪的补偿值
```

参数

<i>encoder_id</i>	预留
<i>comp</i>	传送带补偿值

返回

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setConveyorTrackCompensate(self: pyaubo_sdk.MotionControl, arg0: int, arg1: float) -> int
```

Lua 函数原型

```
setConveyorTrackCompensate(comp: number) -> number
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.setConveyorTrackCompensate","params":[0, 0.1],
"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setConveyorTrackEncoder()

```
int arcs::common_interface::MotionControl::setConveyorTrackEncoder (
    int encoder_id,
    int tick_per_meter)
    设置传送带编码器参数
```

参数

<i>encoder_id</i>	预留
<i>tick_per_meter</i>	线性传送带 ==> 一米的脉冲值 环形传送带 ==> 一圈的脉冲值

返回

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
setConveyorTrackEncoder(self: pyaubo_sdk.MotionControl) -> int
```

Lua 函数原型

```
setConveyorTrackEncoder(encoder_id: bumber,tick_per_meter: number) -> int
```

Lua 示例

```
num = setConveyorTrackEncoder(1,40000)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.setConveyorTrackEncoder","params":[1,40000],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setConveyorTrackLimit()

```
int arcs::common_interface::MotionControl::setConveyorTrackLimit (
    int encoder_id,
    double limit)
```

设置传送带跟踪的最大距离限制

参数

<code>encoder_id</code>	预留
<code>limit</code>	传送带跟踪的最大距离限制, 单位: 米

返回

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
setConveyorTrackLimit(self: pyaubo_sdk.MotionControl, arg0: int, arg1: double) -> int
```

Lua 函数原型

```
setConveyorTrackLimit(encoder_id: number, limit: number) -> int
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.setConveyorTrackLimit", "params": [0, 1.5], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setConveyorTrackSensorOffset()

```
int arcs::common_interface::MotionControl::setConveyorTrackSensorOffset (
    int encoder_id,
    double offset)
```

设置传送带示教位置到同步开关之间的距离

参数

<i>encoder_id</i>	预留
<i>offset</i>	传送带示教位置到同步开关之间的距离, 单位: 米

返回

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
setConveyorTrackSensorOffset(self: pyaubo_sdk.MotionControl, arg0: int, arg1: double) -> int
```

Lua 函数原型

```
setConveyorTrackSensorOffset(encoder_id: number, offset: number) -> int
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.setConveyorTrackSensorOffset", "params": [0, 0.2], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setConveyorTrackStartWindow()

```
int arcs::common_interface::MotionControl::setConveyorTrackStartWindow (
    int encoder_id,
    double window_min,
    double window_max)
```

设置传送带跟踪的启动窗口

参数

<i>encoder_id</i>	预留
<i>min_window</i>	启动窗口的起始位置, 单位: 米
<i>max_window</i>	启动窗口的结束位置, 单位: 米

返回

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
setConveyorTrackStartWindow(self: pyaubo_sdk.MotionControl, arg0: int, arg1: double, arg2:
double) -> int
```

Lua 函数原型

```
setConveyorTrackStartWindow(encoder_id: number, min_window: number, max_window: num-
ber) -> int
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.setConveyorTrackStartWindow","params":[0, 0.4,
2, 1.0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setConveyorTrackSyncSeparation()

```
int arcs::common_interface::MotionControl::setConveyorTrackSyncSeparation (
    int encoder_id,
    double distance,
    double time)
```

设置传送带同步分离, 用于过滤掉同步开关中不需要的信号

参数

<i>encoder↔ _id</i>	预留
<i>distance</i>	从出现一个同步信号后到把一个新的同步信号接受为一个有效对象前走的最短距离, 单位: 米
<i>time</i>	从出现一个同步信号后到把一个新的同步信号接受为一个有效对象前走的最短时间, 单位: 秒

distance 和 time 设置数值大于 0 即为生效当 distance 与 time 同时设置时, 优先生效 distance
返回

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
setConveyorTrackSyncSeparation(self: pyaubo_sdk.MotionControl, arg0: int, arg1: double, arg2: double) -> int
```

Lua 函数原型

```
setConveyorTrackSyncSeparation(encoder_id: number, distance: number, time: number) -> int
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.setConveyorTrackSyncSeparation", "params": [0, 0.05, 0.2], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setConveyorTrackVelThreshold()

```
int arcs::common_interface::MotionControl::setConveyorTrackVelThreshold (
    int encoder_id,
    double threshold)
```

设置传送带运行速度阈值, 用于过滤掉编码器突变值导致的位置突变

参数

<i>encoder↔ _id</i>	预留
<i>threshold</i>	传送带运行时期望容许的最大速度 (默认为 0.2), 单位: 米/秒

返回

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setConveyorTrackVelThreshold(self: pyaubo_sdk.MotionControl, arg0: int, arg1: double) -> int
```

Lua 函数原型

```
setConveyorTrackVelThreshold(encoder_id: number, threshold: number) -> int
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.MotionControl.setConveyorTrackVelThreshold", "params": [0, 0.4, 2], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

setEndPath()

```
int arcs::common_interface::MotionControl::setEndPath ()
```

设置轨迹终止点（以当前轨迹段为界）

显式指定以当前正在执行的轨迹段为边界，终止后续轨迹段的交融（blending）。当前轨迹段将完整执行完毕，但不会与后续轨迹进行速度或位置上的平滑过渡，后续轨迹段将被视为新的独立轨迹起点。

该接口不会触发急停，也不会中断当前轨迹段的执行，仅影响轨迹段之间的交融行为。

返回

成功返回 0；失败返回错误码

异常

<code>arcs::common_interface::AuboException</code>

Lua 函数原型

```
setEndPath() -> number
```

Lua 示例

```
setEndPath() - 以当前轨迹段为界，终止后续轨迹交融
```

Python 函数原型

```
setEndPath() -> int
```

Python 示例

```
setEndPath() # 以当前轨迹段为界，终止后续轨迹交融
```


JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.setEndPath","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setEqradius()

```
int arcs::common_interface::MotionControl::setEqradius (
    double eqradius)
```

设置等效半径, 单位 m moveLine/moveCircle 时, 末端姿态旋转的角度等效到末端位置移动, 数值越大, 姿态旋转速度越快

参数

<i>eqradius</i>	0 表示只规划移动, 姿态旋转跟随移动

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE AUBO_BUSY -AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.setEqradius","params":[0.8],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

Python 函数原型

```
setEqradius(self: pyaubo_sdk.MotionControl, arg0: float) -> int
```

Lua 函数原型

```
setEqradius(eqradius: number) -> number
```

Lua 示例

```
num = setEqradius(1)
```

setFuturePointSamplePeriod()

```
int arcs::common_interface::MotionControl::setFuturePointSamplePeriod (
    double sample_time)
```

设置未来路径上点的采样时间间隔

参数

<i>sample_time</i>	采样时间间隔单位: m/s^2

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

setLookAheadSize()

```
int arcs::common_interface::MotionControl::setLookAheadSize (
    int size)
```

设置前瞻段数 1.

对于有较高速度平稳性要求的任务, 如数控加工, 涂胶, 焊接等匀速需求, 较长的前瞻段数可以提供更优的速度规划, 产生的运动会更加平滑; 2. 对于快速响应的抓取类任务, 更倾向于较短的前瞻段数, 以提高反应速度, 但可能因为进给的路径不够及时导致速度波动很大.

返回

成功返回 0

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setLookAheadSize(self: pyaubo_sdk.MotionControl, arg0: int) -> int
```

Lua 函数原型

```
setLookAheadSize(eqradius: number) -> number
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.setLookAheadSize", "params": [1], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setResumeStartPoint()

```
int arcs::common_interface::MotionControl::setResumeStartPoint (
    const std::vector< double > & q,
    int move_type,
    double blend_radius,
    const std::vector< double > & qdmax,
    const std::vector< double > & qddmax,
    const std::vector< double > & vmax,
    const std::vector< double > & amax)
```

设置继续运动参数

参数

<i>q</i>	继续运动起始位置
<i>move_type</i>	0:关节空间 1: 笛卡尔空间
<i>blend_radius</i>	交融半径
<i>qdmax</i>	关节运动最大速度 (6 维度数据)
<i>qddmax</i>	关节运动最大加速度 (6 维度数据)
<i>vmax</i>	直线运动最大线速度, 角速度 (2 维度数据)
<i>amax</i>	直线运动最大线加速度, 角加速度 (2 维度数据)

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE AUBO_BUSY -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
setResumeStartPoint(self: pyaubo_sdk.MotionControl, arg0:List[float],arg1: int,arg2: float,arg3:↵
List[float], arg4: List[float],arg5:float,arg5:float) -> int
```

Lua 函数原型

```
setResumeStartPoint(q : table, move_type: number,blend_radius: number, qdmax: table, qd-
dmax: table,vmax:number,amax:number) -> nilr
```

Lua 示例

```
setResumeStartPoint({0,0,0,0,0,0},1,1,{0,0,0,0,0,0},{0,0,0,0,0,0},{1,1},{1,1})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.setResumeStartPoint","params":[[0,0,0,0,0,0],1,1,↵
[0,0,0,0,0,0],[0,0,0,0,0,0],1,1],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setServoMode()

```
ARCS_DEPRECATED int arcs::common_interface::MotionControl::setServoMode (
    bool enable)
```

设置伺服模式使用 setServoModeSelect 替代

参数

<i>enable</i>	是否使能

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE AUBO_BUSY -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setServoMode(self: pyaubo_sdk.MotionControl, arg0: bool) -> int
```

Lua 函数原型

```
setServoMode(enable: boolean) -> nil
```

Lua 示例

```
setServoMode(true)
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.MotionControl.setServoMode", "params": [true], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

setServoModeSelect()

```
int arcs::common_interface::MotionControl::setServoModeSelect (
    int mode)
```

设置伺服运动模式

参数

<i>mode</i>	0-退出伺服模式 1-(截断式) 规划伺服模式 2-透传模式 (直接下发) 3-透传模式 (缓存) 4-1ms 透传模式 (缓存) 5-规划伺服模式 6-(截断式) 规划伺服模式, 可以叠加力控 7-规划伺服模式, 可以叠加力控伺服模式 1 添加路点后, 会实时调整目标点和规划路线 (当前的目标点被更新后, 不能保证达到之前设定的目标点) 伺服模式 5 添加路点后, 能保证经过所有目标点

返回

Lua 函数原型

```
setServoModeSelect(0) -> nil
```

Lua 示例

```
setServoModeSelect(0)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.setServoModeSelect","params":[0],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setSpeedFraction()

```
int arcs::common_interface::MotionControl::setSpeedFraction (
    double fraction)
动态调整机器人运行速度和加速度比例 (0., 1.
]
```

参数

<i>fraction</i>	机器人运行速度和加速度比例

返回

成功返回 0; 失败返回错误码 AUBO_INVL_ARGUMENT AUBO_BUSY AUBO_BAD_STATE
-AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
setSpeedFraction(self: pyaubo_sdk.MotionControl, arg0: float) -> int
```

Lua 函数原型

```
setSpeedFraction(fraction: number) -> nil
```

Lua 示例

```
setSpeedFraction(0.5)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.setSpeedFraction","params":[0.8],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setTcpMaxLinearVelocity()

```
int arcs::common_interface::MotionControl::setTcpMaxLinearVelocity (
    double v)
    设置TCP 最大线速度
```

参数

<i>v</i>	TCP 最大线速度值, 单位为 m/s (米/秒)

异常

<i>arcs::common_interface::AuboException</i>	

Lua 函数原型

```
setTcpMaxLinearVelocity(number) -> number
```

Lua 示例

```
setTcpMaxLinearVelocity(0.5) – 设置TCP 最大线速度为 0.5 米/秒
```

Python 函数原型

```
setTcpMaxLinearVelocity(v: float) -> int
```

Python 示例

```
setTcpMaxLinearVelocity(0.5) – 设置TCP 最大线速度为 0.5 米/秒
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.setTcpMaxLinearVelocity", "params": [0.5], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setTimeOptimalEnable()

```
int arcs::common_interface::MotionControl::setTimeOptimalEnable (
    bool enable)
    设置时间最优算法默认关闭
```

参数

<i>enable</i>	

异常

<code>arcs::common_interface::AuboException</code>

Lua 函数原型

```
setTimeOptimalEnable() -> bool
```

Lua 示例

```
setTimeOptimalEnable(true)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.setTimeOptimalEnable","params":[true],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setWorkObjectHold()

```
int arcs::common_interface::MotionControl::setWorkObjectHold (
    const std::string & module_name,
    const std::vector< double > & mounting_pose)
```

当工件安装在另外一台机器人的末端或者外部轴上时，指定其名字和安装位置

注解

暂未实现

参数

<code>module_name</code>	控制模块名字
<code>mounting_pose</code>	抓取的相对位置，如果是机器人，则相对于机器人末端中心点（非TCP 点）

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE AUBO_BUSY -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setWorkObjectHold(self: pyaubo_sdk.MotionControl, arg0: str, arg1: List[float]) -> int
```

Lua 函数原型

```
setWorkObjectHold(module_name: string, mounting_pose: table) -> nil
```

Lua 示例

```
setWorkObjectHold("object",{0.2,0.1,-0.4,3.14,0,-1.57})
```

speedFractionCritical()

```
int arcs::common_interface::MotionControl::speedFractionCritical (
    bool enable)
```

速度比例设置临界区，使能之后速度比例被强制设定为 1。
，失能之后恢复之前的速度比例

参数

<i>enable</i>	

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE AUBO_BUSY -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Lua 函数原型

```
speedFractionCritical\(\) -> int
```

Lua 示例

```
speedFractionCritical(true)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.speedFractionCritical","params":[true],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

speedFractionCritical1()

```
int arcs::common_interface::MotionControl::speedFractionCritical1 (
    bool enable)
```

速度比例设置临界区，使能之后速度比例被强制设定为当前生效的比例，失能之后恢复之前的速度比例

参数

<i>enable</i>	

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE AUBO_BUSY -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Lua 函数原型

```
speedFractionCritical1() -> int
```

Lua 示例

```
speedFractionCritical1(true)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.speedFractionCritical1","params":[true],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

speedJoint()

```
int arcs::common_interface::MotionControl::speedJoint (
    const std::vector< double > & qd,
    double a,
    double t)
```

关节空间速度跟随

当机械臂还没达到目标速度的时候，给一个新的目标速度，机械臂会立刻达到新的目标速度

参数

<i>qd</i>	目标关节速度, 单位 rad/s
<i>a</i>	主轴的加速度, 单位 rad/s ²
<i>t</i>	函数返回所需要的时间, 单位 s 如果 $t = 0$, 当达到目标速度的时候, 函数将返回; 反之, 则经过 t 时间后, 函数返回, 不管是否达到目标速度。 如果没有达到目标速度, 会减速到零。如果达到了目标速度就是按照目标速度匀速运动。

返回值

<i>0</i>	成功
<i>AUBO_BAD_STATE(1)</i>	当前安全模式处于非 Normal、ReducedMode、Recovery 状态
<i>AUBO_BUSY(3)</i>	上一条指令正在执行中
<i>-AUBO_BAD_STATE(-1)</i>	可能的原因包括但不限于: 线程已分离、线程被终止、task_id 未找到, 或者当前机器人模式非 Running
<i>-AUBO_TIMEOUT(-4)</i>	调用接口超时
<i>-AUBO_INVL_ARGUMENT(-5)</i>	参数数组 qd 的长度小于当前机器臂的自由度

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
speedJoint(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: float, arg2: float) -> int
```

Lua 函数原型

```
speedJoint(qd: table, a: number, t: number) -> nil
```

Lua 示例

```
speedJoint({0.2,0,0,0,0,0}, 1.5,10)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.speedJoint","params":[[0.2,0,0,0,0,0],1.5,100],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

speedLine()

```
int arcs::common_interface::MotionControl::speedLine (
    const std::vector< double > & xd,
    double a,
    double t)
```

笛卡尔空间速度跟随

当机械臂还没达到目标速度的时候，给一个新的目标速度，机械臂会立刻达到新的目标速度

参数

xd	工具速度, 单位 m/s
a	工具位置加速度, 单位 m/s^2
t	函数返回所需要的时间, 单位 s 如果 $t = 0$, 当达到目标速度的时候, 函数将返回; 反之, 则经过 t 时间后, 函数返回, 不管是否达到目标速度。如果没有达到目标速度, 会减速到零。如果达到了目标速度就是按照目标速度匀速运动。

返回值

0	成功
<code>AUBO_BAD_STATE(1)</code>	当前安全模式处于非 Normal、ReducedMode、Recovery 状态
<code>AUBO_BUSY(3)</code>	上一条指令正在执行中

-AUBO_BAD_STATE(-1)	可能的原因包括但不限于：线程已分离、线程被终止、task_id 未找到，或者当前机器人模式非 Running
-AUBO_TIMEOUT(-4)	调用接口超时

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
speedLine(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: float, arg2: float) -> int
```

Lua 函数原型

```
speedLine(pose: table, a: number, t: number) -> nil
```

Lua 示例

```
speedLine({0.25,0,0,0,0,0},1.2,100)
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.MotionControl.speedLine", "params": [[0.25,0,0,0,0,0],1.2,100], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

startMove()

```
int arcs::common_interface::MotionControl::startMove ()
```

StartMove 用于在以下情况下恢复机器人、外部轴的运动以及相关工艺进程：

- 通过 StopMove 指令停止后。
- 执行 StorePath ... RestoPath 序列后。
- 发生异步运动错误（如 ERR_PATH_STOP）或特定工艺错误并在 ERROR 处理器中处理后。

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE AUBO_BUSY -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Lua 函数原型

```
startMove() -> number
```

Lua 示例

```
num = startMove()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.startMove", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

stopJoint()

```
int arcs::common_interface::MotionControl::stopJoint (
    double acc)
```

关节空间停止运动

参数

<i>acc</i>	关节加速度, 单位: rad/s^2

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
stopJoint(self: pyaubo_sdk.MotionControl, arg0: float) -> int
```

Lua 函数原型

```
stopJoint(acc: number) -> nil
```

Lua 示例

```
stopJoint(2)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.stopJoint", "params": [31], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

stopLine()

```
int arcs::common_interface::MotionControl::stopLine (
    double acc,
    double acc_rot)
```

停止 moveLine/moveCircle 等在笛卡尔空间的运动

参数

<i>acc</i>	工具加速度, 单位: m/s ²
<i>acc_rot</i>	

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
stopLine(self: pyaubo_sdk.MotionControl, arg0: float, arg1: float) -> int
```

Lua 函数原型

```
stopLine(acc: number, acc_rot: number) -> nil
```

Lua 示例

```
stopLine(10,10)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.stopLine","params":[10,10],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

stopMove()

```
int arcs::common_interface::MotionControl::stopMove (
    bool quick,
    bool all_tasks)
```

StopMove 用于临时停止机器人和外部轴的运动以及相关工艺进程。如果调用 StartMove 指令, 则运动和工艺进程将恢复。

该指令可用于中断处理程序中, 在发生中断时临时停止机器人。

参数

<i>quick</i>	true: 以最快速度在路径上停止机器人。未指定 quick 参数时, 机器人将在路径上停止, 但制动距离较长 (与普通程序停止相同)。

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE AUBO_BUSY -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>	

Lua 函数原型

```
StopMove(quick: bool,all_tasks: bool) -> number
```

Lua 示例

```
num = StopMove(true,true)
```

storePath()

```
int arcs::common_interface::MotionControl::storePath (
    bool keep_sync)
storePath
```

参数

<code>keep_sync</code>	

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE AUBO_BUSY -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>	

Lua 函数原型

```
storePath(keep_sync: bool) -> number
```

Lua 示例

```
num = storePath()
```

trackCartesian()

```
int arcs::common_interface::MotionControl::trackCartesian (
    const std::vector< double > & pose,
    double t,
    double smooth_scale,
    double delay_sacle)
```

跟踪运动, 用于执行离线轨迹、透传用户规划轨迹等与 trackJoint 区别在于接收笛卡尔空间位姿而不是关节角度 (由软件内部直接做逆解)

参数

<i>pose</i>	
<i>t</i>	
<i>smooth_scale</i>	
<i>delay_sacle</i>	

返回**Lua 函数原型**

```
trackCartesian(pose: table,t: number,smooth_scale: number,delay_sacle: number) -> nil
```

Lua 示例

```
trackCartesian({0.58712,-0.15775,0.48703,2.76,0.344,1.432},0.01,0.5,1)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.trackCartesian","params":[[0.58712,-0.15775,0.48703,2.76,0.344,1.432],0.01,0.5,1],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

trackJoint()

```
int arcs::common_interface::MotionControl::trackJoint (
    const std::vector< double > & q,
    double t,
    double smooth_scale,
    double delay_sacle)
```

跟踪运动，用于执行离线轨迹、透传用户规划轨迹等

参数

<i>q</i>	
<i>smooth_scale</i>	
<i>delay_sacle</i>	

返回**Lua 函数原型**

```
trackJoint(q: table,t: number,smooth_scale: number,delay_sacle: number) -> nil
```

Lua 示例

```
trackJoint({0,0,0,0,0,0},0.01,0.5,1)
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.MotionControl.trackJoint", "params": [[0,0,0,0,0,0],0.01,0.5,1], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

weaveEnd()

```
int arcs::common_interface::MotionControl::weaveEnd ()  
结束摆动
```

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.MotionControl.weaveEnd", "params": [], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

weaveStart()

```
int arcs::common_interface::MotionControl::weaveStart (  
    const std::string & params)
```

警告

调用 weaveStart 前, 需提前开启 [RuntimeMachine](#) 规划器 start。

开始摆动: 在 weaveStart 与 weaveEnd 之间的 moveLine/moveCircle/moveProcess 根据 params 执行摆动。

参数

<i>params</i>	Json 字符串。

自 v0.29 起的字段

- "type": <string> ——波形类型: "SINE" | "SPIRAL" | "TRIANGLE" | "TRAPEZOIDAL" (区分大小写)。
- "step": <number> ——相邻摆动采样的弧长步长, 单位 m。
- "amplitude": [<number>,<number>] ——摆弧幅度左右侧, 单位 m。
- "hold_distance": [<number>,<number>] ——在幅度峰值处沿路径的停留距离, 单位 m。
- "hold_time": [<number>,<number>] ——在幅度峰值处的停留时间, 单位 s。
- "angle": <number> ——叠加方向与法平面的夹角, 单位 rad。
- "direction": <integer> ——起始摆动方向: 0= 上方, 1= 下方。
- "movep_uniform_vel": <bool> ——moveProcess 匀速规划, 布尔量。

自 v0.31 起

- 与 v0.29 相同。

自 v0.32 起新增/变更

- "angle": [<number>,<number>] ——运动叠加方向与法平面的夹角, 单位 rad。
- "frequency": <number> ——摆动频率, 单位 Hz (范围[0.1, 5])。
- "ori_weave_range": [<number>,<number>,<number>] ——姿态摆动范围 [x,y,z], 单位 rad; 常与阿基米德螺旋用于力控搜孔。
- "ori_weave_frequency": [<number>,<number>,<number>] ——姿态摆动频率 [x,y,z], 单位 m^{-1} (按路径弧长计; 范围 [0, 1])。
- "adjust_cycle_num": <integer> ——动态调节频率/幅值的过渡段数量。
- "azimuth": <number> ——摆焊波形方位角, 单位 rad。
- "ridge_height": <number> ——中心上升高度, 单位 m。

警告

可使用关系 **frequency = vel / step** 与外部速度 vel 关联 (vel 单位 $m \cdot s^{-1}$)。

hold_time 仅在 ****SINE**** 下可用, 且可左右不对称。

"hold_distance"/"hold_time" 互斥, 仅一个生效

注解

- 数组约定: "amplitude"/"hold_distance"/"hold_time" 为 [left,right]; "ori_weave_*" 为姿态 [x,y,z]。
- 所有角度均为弧度 (rad)。

返回

成功返回 0; 失败返回错误码

- AUBO_BUSY
- AUBO_BAD_STATE
- -AUBO_INVL_ARGUMENT
- -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

示例 v0.29/v0.31

```
params = {
    "type": "SINE",
    "step": 0.005,
    "amplitude": [0.01, 0.01],
    "hold_distance": [0.001, 0.001],
    "hold_time": [0, 0],
    "angle": 0,
    "direction": 0,
    "movep_uniform_vel": false
}
```

示例 v0.32

```
params = {
    "type" : "SINE", // "SINE" "SPIRAL" "TRIANGLE" "SAWTOOTH" "CRESCENT"
    "step" : 0.0, // 可用 frequency = vel / step (vel: m * s^-1)
    "frequency" : 0.0, // [0.1, 5] Hz
    "amplitude" : [0.01, 0.01], // m
    "hold_distance" : [0.001, 0.001], // m
    "hold_time" : [0, 0], // s (仅 SINE)
    "angle" : [0, 0], // rad
    "direction" : 0,
    "ori_weave_range" : [0.001, 0.001, 0.001], // rad
    "ori_weave_frequency" : [0.001, 0.001, 0.001], // m^-1
    "adjust_cycle_num" : 0,
    "azimuth" : 0, // rad
    "ridge_height" : 0 // m
}
```

Python 函数原型

```
weaveStart(self: pyaubo_sdk.MotionControl, arg0: str) -> int
```

Python 示例

```
robot_name = rpc_cli.getRobotNames()[0]
robot_interface = rpc_cli.getRobotInterface(robot_name)
robot_interface.getMotionControl().weaveStart(params)
```

Lua 函数原型

```
weaveStart(params: string) -> nil
```

Lua 示例 (v0.29/0.31)

```
weaveStart("{\"type\":\"SINE\",\"step\":0.005,\"amplitude\":[0.01,0.01],\"hold_distance\":[0.001,0.001],\"hold_time\":[0,0],\"angle\":0,\"direction\":0,\"movep_uniform_vel\":false}")
```

Lua 示例 (v0.32)

```
weaveStart("{\"type\":\"SINE\",\"step\":0.0,\"frequency\":2.0,\"amplitude\":[0.01,0.01],\"hold_distance\":[0.001,0.001],\"hold_time\":[0,0],\"angle\":0,\"direction\":0,\"ori_weave_range\":[0.001,0.001,0.001],\"ori_weave_frequency\":[0.001,0.001,0.001],\"adjust_cycle_num\":0,\"azimuth\":0,\"ridge_height\":0}")
```

JSON-RPC 请求示例 (v0.29/0.31)

```
{
  "jsonrpc": "2.0",
  "method": "rob1.MotionControl.weaveStart",
  "params": [
    {
      "type": "SINE",
      "step": 0.005,
      "amplitude": [0.01, 0.01],
      "hold_distance": [0.001, 0.001],
      "hold_time": [0, 0],
      "angle": 0,
      "direction": 0,
      "movep_uniform_vel": false
    }
  ],
  "id": 1
}
```

JSON-RPC 请求示例 (v0.32)

```
{ "jsonrpc": "2.0", "method": "rob1.MotionControl.weaveStart", "params":
[{"type": "SINE", "step": 0.0, "frequency": 2.0, "amplitude": [0.01, 0.01], "hold_distance":
[0.001, 0.001], "hold_time": [0, 0], "angle": 0, "direction": 0, "ori_weave_range":
[0.001, 0.001, 0.001], "ori_weave_frequency":
[0.001, 0.001, 0.001], "adjust_cycle_num": 0, "azimuth": 0, "ridge_height": 0}], "id": 1}
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

weaveUpdateParameters()

```
int arcs::common_interface::MotionControl::weaveUpdateParameters (
    const std::string & params)
更新摆动过程中的频率和振幅
```

参数

<i>params</i>	Json 字符串用于定义摆动参数 { "frequency": <num>, "amplitude": {<num>, <num>} }

返回

成功返回 0

异常

<i>arcs::common_interface::AuboException</i>

8.6.6 RobotAlgorithm (机器人算法工具)

机器人算法相关的对外接口

RobotAlgorithm (机器人算法工具) 的协作图:



函数

- [ForceSensorCalibResult arcs::common_interface::RobotAlgorithm::calibrateTcpForceSensor](#) (const std::vector< std::vector< double > > &forces, const std::vector< std::vector< double > > &poses)
力传感器标定算法 (三点标定法)
- [ForceSensorCalibResultWithError arcs::common_interface::RobotAlgorithm::calibrateTcpForceSensor2](#) (const std::vector< std::vector< double > > &forces, const std::vector< std::vector< double > > &poses)
力传感器标定算法 (三点标定法)
- [ResultWithErrno arcs::common_interface::RobotAlgorithm::calibrateTcpForceSensor3](#) (const std::vector< std::vector< double > > &forces, const std::vector< std::vector< double > > &poses, const double &mass, const std::vector< double > &cog)

- 力传感器偏置标定算法
- `int arcs::common_interface::RobotAlgorithm::payloadIdentify` (const std::string &data_file_no_↵ payload, const std::string &data_file_with_payload)
基于电流的负载辨识算法接口
- `int arcs::common_interface::RobotAlgorithm::payloadIdentify1` (const std::string &file_name)
新版基于电流的负载辨识算法接口
- `int arcs::common_interface::RobotAlgorithm::payloadCalculateFinished` ()
负载辨识是否计算完成
- `Payload arcs::common_interface::RobotAlgorithm::getPayloadIdentifyResult` ()
获取负载辨识结果
- `bool arcs::common_interface::RobotAlgorithm::frictionModelIdentify` (const std::vector< std::vector< double > > &q, const std::vector< std::vector< double > > &qd, const std::vector< std::vector< double > > &qdd, const std::vector< std::vector< double > > &temp)
关节摩擦力模型辨识算法接口
- `ResultWithErrno arcs::common_interface::RobotAlgorithm::calibWorkpieceCoordinatePara` (const std::vector< std::vector< double > > &q, int type)
工件坐标系标定算法接口 (需要在调用之前正确的设置机器人的TCP 偏移) 输入多组关节角度和标定类型, 输出工件坐标系位姿 (相对于机器人基坐标系)
- `ResultWithErrno arcs::common_interface::RobotAlgorithm::forwardDynamics` (const std::vector< double > &q, const std::vector< double > &torqs)
动力学正解
- `ResultWithErrno arcs::common_interface::RobotAlgorithm::forwardDynamics1` (const std::vector< double > &q, const std::vector< double > &torqs, const std::vector< double > &tcp_offset)
动力学正解, 基于给定的TCP 偏移
- `ResultWithErrno arcs::common_interface::RobotAlgorithm::forwardKinematics` (const std::vector< double > &q)
运动学正解, 基于激活的TCP 偏移 (最近的通过 `setTcpOffset` 设置的参数) 输入关节角度, 输出TCP 位姿
- `ResultWithErrno arcs::common_interface::RobotAlgorithm::forwardKinematics1` (const std::vector< double > &q, const std::vector< double > &tcp_offset)
运动学正解输入关节角度, 输出TCP 位姿
- `ResultWithErrno arcs::common_interface::RobotAlgorithm::forwardToolKinematics` (const std::↵ vector< double > &q)
运动学正解 (忽略 TCP 偏移值)
- `ResultWithErrno1 arcs::common_interface::RobotAlgorithm::forwardKinematicsAll` (const std::↵ vector< double > &q)
运动学正解, 基于激活的TCP 偏移 (最近的通过 `setTcpOffset` 设置的参数) 输入关节角度, 输出各连杆位姿
- `ResultWithErrno arcs::common_interface::RobotAlgorithm::inverseKinematics` (const std::vector< double > &qnear, const std::vector< double > &pose)
运动学逆解输入TCP 位姿和参考关节角度, 输出关节角度
- `ResultWithErrno arcs::common_interface::RobotAlgorithm::inverseKinematics1` (const std::vector< double > &qnear, const std::vector< double > &pose, const std::vector< double > &tcp_offset)
运动学逆解输入TCP 位姿和参考关节角度, 输出关节角度
- `ResultWithErrno arcs::common_interface::RobotAlgorithm::inverseKinematics2` (const std::vector< double > &qnear, const std::vector< double > &pose)
同构型最小关节变化逆解按照 `qnear` 的机械臂构型计算目标TCP 位姿逆解, 并在同构型逆解中选择相对 `qnear` 关节变化最小的一组基于激活的TCP 偏移 (最近的通过 `setTcpOffset` 设置的参数)
- `ResultWithErrno1 arcs::common_interface::RobotAlgorithm::inverseKinematicsAll` (const std::↵ vector< double > &pose)
求出所有的逆解, 基于激活的TCP 偏移
- `ResultWithErrno1 arcs::common_interface::RobotAlgorithm::inverseKinematicsAll1` (const std::↵ vector< double > &pose, const std::vector< double > &tcp_offset)
求出所有的逆解, 基于提供的TCP 偏移

- `ResultWithErrno arcs::common_interface::RobotAlgorithm::inverseToolKinematics` (const std::vector< double > &qnear, const std::vector< double > &pose)
运动学逆解 (忽略 TCP 偏移值)
- `ResultWithErrno1 arcs::common_interface::RobotAlgorithm::inverseToolKinematicsAll` (const std::vector< double > &pose)
运动学逆解 (忽略 TCP 偏移值)
- `ResultWithErrno3 arcs::common_interface::RobotAlgorithm::getRobotConfiguration` (const std::vector< double > &q)
根据输入的关节角计算并返回对应的机械臂构型
- `ResultWithErrno arcs::common_interface::RobotAlgorithm::calcJacobian` (const std::vector< double > &q, bool base_or_end)
计算机机械臂末端的雅克比矩阵
- `std::vector< std::vector< double > > arcs::common_interface::RobotAlgorithm::pathBlend3Points` (int type, const std::vector< double > &q_start, const std::vector< double > &q_via, const std::vector< double > &q_to, double r, double d)
求解交融的轨迹点
- `int arcs::common_interface::RobotAlgorithm::generatePayloadIdentifyTraj` (const std::string &name, const `TrajConfig` &traj_conf)
生成用于负载辨识的激励轨迹此接口内部调用 `pathBufferAppend` 将离线轨迹存入 `buffer` 中, 后续可通过 `movePathBuffer` 运行离线轨迹
- `int arcs::common_interface::RobotAlgorithm::payloadIdentifyTrajGenFinished` ()
负载辨识轨迹是否生成完成
- `std::vector< std::vector< double > > arcs::common_interface::RobotAlgorithm::pathMoveS` (const std::vector< std::vector< double > > &q_s, double d)
求解 `moveS` 的轨迹点
- `ResultWithErrno1 arcs::common_interface::RobotAlgorithm::calibVibrationParams` (const std::vector< std::vector< double > > &q, const std::vector< std::vector< double > > &qd, const std::vector< std::vector< double > > &target_q, const std::vector< std::vector< double > > &target_qd, const std::vector< std::vector< double > > &target_qdd, const std::vector< double > &tool_offset)
振动抑制参数辨识算法接口
- `ResultWithErrno1 arcs::common_interface::RobotAlgorithm::calibVibrationParams1` (const std::string &record_cache_name, const std::vector< double > &tool_offset)
振动抑制参数辨识算法接口 1
- `int arcs::common_interface::RobotAlgorithm::needVibrationRecalib` (const `VibrationRecalibrationParameter` &param1, const `VibrationRecalibrationParameter` &param2, double threshold)
判断是否需要重新辨识振动参数
- `int arcs::common_interface::RobotAlgorithm::addCollisionBox` (const std::string &name, const std::string &link_name, const std::vector< std::vector< double > > &sizes, const std::vector< std::vector< double > > &poses)
添加立方体碰撞对象
- `int arcs::common_interface::RobotAlgorithm::removeCollisionObject` (const std::string &name)
删除指定的碰撞对象
- `int arcs::common_interface::RobotAlgorithm::validatePath` (int type, const std::vector< double > &start, double r1, const std::vector< double > &end, double r2, double d)
验证机器人运动路径从起点到终点的可达性, 不启用外部碰撞检测。
- `int arcs::common_interface::RobotAlgorithm::validatePath1` (int type, const std::vector< double > &start, double r1, const std::vector< double > &end, double r2, double d, bool check_external_collision)
验证机器人运动路径从起点到终点的可达性, 可选择是否启用外部碰撞检测。

详细描述

机器人算法相关的对外接口

函数说明

addCollisionBox()

```
int arcs::common_interface::RobotAlgorithm::addCollisionBox (
    const std::string & name,
    const std::string & link_name,
    const std::vector< std::vector< double > > & sizes,
    const std::vector< std::vector< double > > & poses)
```

添加立方体碰撞对象

该接口用于向当前机器人算法场景中添加一个命名的立方体碰撞对象组。

参数

<i>name</i>	碰撞对象唯一标识
<i>link_name</i>	碰撞对象挂载的参考 link 名称。常用名称包括: world、base_link、shoulder_Link、upperArm_Link、foreArm_Link、wrist1_Link、wrist2_Link、wrist3_Link、end_effector
<i>sizes</i>	立方体尺寸列表。每个元素格式为[length, width, height], 单位: m
<i>poses</i>	碰撞对象位姿列表。每个元素表示相对参考 link 的位姿, 格式为[x, y, z, rx, ry, rz]

返回

0 表示成功, 非 0 表示失败

Lua 函数原型

```
addCollisionBox(name, link_name, sizes, poses) -> integer
```

Lua 示例

- 添加挂载到 world 的环境碰撞体 addCollisionBox("env_box", "world", {{0.4, 0.4, 0.2}}, {{0.6, 0.0, 0.1, 0.0, 0.0, 0.0}})
- 添加挂载到 end_effector 的末端碰撞体 addCollisionBox("tool_box", "end_effector", {{0.1, 0.08, 0.06}}, {{0.0, 0.0, 0.05, 0.0, 0.0, 0.0}})

Python 函数原型

```
addCollisionBox(name: str, link_name: str, sizes: list[list[float]], poses: list[list[float]]) -> int
```

Python 示例

添加挂载到 world 的环境碰撞体

```
robot_algorithm.addCollisionBox( "env_box", "world", [[0.4, 0.4, 0.2]], [[0.6, 0.0, 0.1, 0.0, 0.0, 0.0]] )
```

添加挂载到 end_effector 的末端碰撞体

```
robot_algorithm.addCollisionBox( "tool_box", "end_effector", [[0.1, 0.08, 0.06]], [[0.0, 0.0, 0.05, 0.0, 0.0, 0.0]] )
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.addCollisionBox", "params": ["env_box", "world", [[0.4, 0.4, 0.2]], [[0.6, 0.0, 0.1, 0.0, 0.0, 0.0]]], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

calcJacobian()

```
ResultWithErrno arcs::common_interface::RobotAlgorithm::calcJacobian (
    const std::vector< double > & q,
    bool base_or_end)
```

计算机机械臂末端的雅克比矩阵

参数

q	关节角
$base\_or\_end$	参考坐标系为基坐标系（或者末端坐标系） true: 在 base 下描述 false: 在末端坐标系下描述

返回

雅克比矩阵是否有效返回值的第一个参数为该构型下对应的雅克比矩阵，第二个为逆解错误码此接口的错误码返回值在 0.28.1-rc.21 0.29.0-alpha.25 版本之后做了修改。此前逆解错误时返回 30082，修改后错误码返回列表如下 0 - 成功 -23 - 逆解计算不收敛，计算出错 -24 - 逆解计算超出机器人最大限制 -25 - 逆解输入配置存在错误 -26 - 逆解雅可比矩阵计算失败 -27 - 目标点存在解析解，但均不满足选解条件 -28 - 逆解返回未知类型错误若错误码非 0，则返回值的第一个参数为输入参考关节角 q_{near}

异常

```
arcs::common_interface::AuboException
```

Python 函数原型

```
calJacobian(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float], arg1: bool) -> Tuple[List[float], int]
```

Lua 函数原型

```
calJacobian(q: table, base_or_end: boolean) -> table
```

Lua 示例

```
calJ_result = calJacobian({0.58815,0.0532,0.62391,2.46,0.479,1.619},true)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.calcJacobian", "params": [[0.58815,0.0532,0.62391,2.46,0.479,1.619],true],"id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": [[0.20822779551242535,-0.5409416184208162,0.2019786999613013,0.061264982268770196,-0.026269884327316487, 0.10131708699859962,0.26388933410019777,-0.36074292664199115,0.1346954733416397,0.04085636647597124,-0.07244204452918337,0.0708466286633346, 0.0,0.10401808481666497,-0.12571344758923886,-0.07741290545882097,0.18818543519232858,0.04628646442706299,0.0,0.5548228314607867,-0.5548228314607868,0.5548228314607868,-0.7901273140338193,0.37230961532208007,0.0,-0.8319685244586092,0.8319685244586091,-0.8319685244586091,-0.5269197820578843,-0.8184088260676008,1.0,3.749399456654644e-33,-0.512048180336603e-18,1.0956823467534067e-16,-0.31313634553301894, 0.43771285536682175],0]}
```

calibrateTcpForceSensor()

```
ForceSensorCalibResult arcs::common_interface::RobotAlgorithm::calibrateTcpForceSensor (
    const std::vector< std::vector< double > > & forces,
    const std::vector< std::vector< double > > & poses)
```

力传感器标定算法 (三点标定法)

参数

<i>force</i>	力数据
<i>q</i>	关节角度

返回

标定结果

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
calibrateTcpForceSensor(self: pyaubo_sdk.RobotAlgorithm, arg0: List[List[float]], arg1: List←
[List[float]]) -> Tuple[List[float], List[float], float, List[float]]
```

Lua 函数原型

```
calibrateTcpForceSensor(force: table, q: table) -> table
```

Lua 示例

```
cal_table = calibrateTcpForceSensor({10.0,10.0,10.0,-1.2,-1.2,-1.2}, {3.083,1.227,1.098,0.670,-1.←
870,-0.397})
```

calibrateTcpForceSensor2()

```
ForceSensorCalibResultWithError arcs::common_interface::RobotAlgorithm::calibrateTcpForceSensor2 (
    const std::vector< std::vector< double > > & forces,
    const std::vector< std::vector< double > > & poses)
```

力传感器标定算法 (三点标定法)

参数

<i>forces</i>	
<i>poses</i>	

返回

force_offset, com, mass, angle error

异常

<code>arcs::common_interface::AuboException</code>

calibrateTcpForceSensor3()

```
ResultWithErrno arcs::common_interface::RobotAlgorithm::calibrateTcpForceSensor3 (
    const std::vector< std::vector< double > > & forces,
    const std::vector< std::vector< double > > & poses,
    const double & mass,
    const std::vector< double > & cog)
```

力传感器偏置标定算法

参数

<i>force</i>	力数据
<i>poses</i>	位姿
<i>mass</i>	质量, 单位: kg
<i>cog</i>	重心, 单位: m, 形式为 (CoGx, CoGy, CoGz)

返回

标定结果

异常

<code>arcs::common_interface::AuboException</code>

calibVibrationParams()

```
ResultWithErrno1 arcs::common_interface::RobotAlgorithm::calibVibrationParams (
    const std::vector< std::vector< double > > & q,
    const std::vector< std::vector< double > > & qd,
    const std::vector< std::vector< double > > & target_q,
    const std::vector< std::vector< double > > & target_qd,
    const std::vector< std::vector< double > > & target_qdd,
    const std::vector< double > & tool_offset)
```

振动抑制参数辨识算法接口

参数

<i>q</i>	当前关节角度
<i>qd</i>	当前关节速度
<i>target_q</i>	目标关节角度
<i>target_qd</i>	关节速度
<i>target_qdd</i>	关节加速度

<i>tool_offset</i>	工具TCP 信息
<i>omega</i>	振动频率
<i>zeta</i>	振动阻尼比

返回

振动抑制参数和是否识别成功

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
calibVibrationParams(self: pyaubo_sdk.RobotAlgorithm, arg0: List[List[float]], arg1: List[List[
[float]], arg2: List[List[float]], arg3: List[List[float]], arg4: List[List[float]], arg5: List[float]) ->
list[list[float]],int
```

Lua 函数原型

```
calibVibrationParams(q: table,qd: table, target_q: table, target_qd: table, target_qdd: table,
tool_offset: table, omega: table, zeta: table) -> table,number
```

calibVibrationParams1()

```
ResultWithError1 arcs::common_interface::RobotAlgorithm::calibVibrationParams1 (
    const std::string & record_cache_name,
    const std::vector< double > & tool_offset)
```

振动抑制参数辨识算法接口 1

参数

<i>record_cache_name</i>	目标缓存名称
<i>tool_offset</i>	工具TCP 信息

返回

振动抑制参数和是否识别成功

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
calibVibrationParams(self: pyaubo_sdk.RobotAlgorithm, arg0: string, arg1: List[float]) -> list[
[list[float]],int
```

Lua 函数原型

```
calibVibrationParams(record_cache_name: string, tool_offset: table) -> table,number
```

calibWorkpieceCoordinatePara()

```
ResultWithErrno arcs::common_interface::RobotAlgorithm::calibWorkpieceCoordinatePara (
    const std::vector< std::vector< double > > & q,
    int type)
```

工件坐标系标定算法接口 (需要在调用之前正确的设置机器人的TCP 偏移) 输入多组关节角度和标定类型, 输出工件坐标系位姿 (相对于机器人基坐标系)

参数

q	关节角度
$type$	标定类型

返回

计算结果 (工件坐标系位姿) 以及错误代码

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
calibWorkpieceCoordinatePara(self: pyaubo_sdk.RobotAlgorithm, arg0: List[List[float]], arg1: int)
-> Tuple[List[float], int]
```

Lua 函数原型

```
calibWorkpieceCoordinatePara(q: table, type: number) -> table, number
```

Lua 示例

```
coord_pose,coord_num = calibWorkpieceCoordinatePara({3.083,1.227,1.098,0.670,-1.870,-0.397},1)
```

forwardDynamics()

```
ResultWithErrno arcs::common_interface::RobotAlgorithm::forwardDynamics (
    const std::vector< double > & q,
    const std::vector< double > & torqs)
```

动力学正解

参数

q	关节角
$torqs$	

返回

计算结果以及错误代码

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
forwardDynamics(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float], arg1: List[float]) -> Tuple←
[List[float], int]
```

Lua 函数原型

```
forwardDynamics(q: table, torqs: table) -> table, number
```

Lua 示例

```
Dynamics, fk_result = forwardDynamics({3.083,1.227,1.098,0.670,-1.870,-0.397},{0.0,0.0,0.0,0.0,0.↵
0,0.0})
```

forwardDynamics1()

```
ResultWithErrno arcs::common_interface::RobotAlgorithm::forwardDynamics1 (
    const std::vector< double > & q,
    const std::vector< double > & torqs,
    const std::vector< double > & tcp_offset)
```

动力学正解，基于给定的TCP 偏移

参数

q	关节角
$torqs$	
tcp_offset	TCP 偏移

返回

计算结果以及错误代码，同 forwardDynamics

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
forwardDynamics(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float], arg1: List[float], arg2:↵
List[float]) -> Tuple[List[float], int]
```

Lua 函数原型

```
forwardDynamics1(q: table, torqs: table, tcp_offset: table) -> table, number
```

Lua 示例

```
Dynamics, fk_result = forwardDynamics1({3.083,1.227,1.098,0.670,-1.870,-0.397},{0.0,0.0,0.0,0.↵
0,0.0,0.0},{0.0,0.13201,0.03879,0,0,0})
```

forwardKinematics()

```
ResultWithErrno arcs::common_interface::RobotAlgorithm::forwardKinematics (
    const std::vector< double > & q)
```

运动学正解, 基于激活的TCP 偏移 (最近的通过 setTcpOffset 设置的参数) 输入关节角度, 输出TCP 位姿

参数

q	关节角

返回

TCP 位姿和正解结果是否有效返回值的第一个参数为正解结果, 第二个为正解错误码, 错误码返回值列表如下 0 - 成功 -1 - 机械臂状态不对 (未初始化完成, 可尝试再次调用) -5 - 输入的关节角无效 (维度错误)

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
forwardKinematics(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float]) -> Tuple[List[float], int]
```

Lua 函数原型

```
forwardKinematics(q: table) -> table, number
```

Lua 示例

```
pose, fk_result = forwardKinematics({3.083,1.227,1.098,0.670,-1.870,-0.397})
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.forwardKinematics", "params": [[3.083688522170976, 1.2273215976885394, 1.098072739631141, 0.6705738810610149, -1.870715392248607, -0.39708546603119627], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": [[0.7137448715395925, 0.08416057568819092, 0.6707994191515292, 2.4599818776908724, 0.4789772388601265, 1.6189630435878408], 0] }
```

forwardKinematics1()

```
ResultWithErrno arcs::common_interface::RobotAlgorithm::forwardKinematics1 (
    const std::vector< double > & q,
    const std::vector< double > & tcp_offset)
```

运动学正解输入关节角度, 输出TCP 位姿

参数

q	关节角
tcp_offset	tcp 偏移

返回

TCP 位姿和正解结果是否有效返回值的第一个参数为正解结果，第二个为正解错误码，错误码返回值列表如下 0 - 成功 -1 - 机械臂状态不对 (未初始化完成，可尝试再次调用) -5 - 输入的关节角或 tcp 偏移无效 (维度错误)

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
forwardKinematics(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float], arg1: List[float]) -> Tuple[List[float], int]
```

Lua 函数原型

```
forwardKinematics1(q: table, tcp_offset: table) -> table, number
```

Lua 示例

```
pose, fk_result = forwardKinematics1({3.083,1.227,1.098,0.670,-1.870,-0.397},{0.0,0.13201,0.03879,0,0,0})
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.forwardKinematics1", "params": [[3.083688522170976, 1.2273215976885394, 1.098072739631141, 0.6705738810610149, -1.870715392248607, -0.39708546603119627], [0.0, 0.13201, 0.03879, 0, 0, 0]], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": [[0.7137636726659518, 0.0837705432006433, 0.6710022027216355, 2.459981877690872, 0.4789772388601267, 1.6189630435878408], 0] }
```

自从

0.24.1

forwardKinematicsAll()

```
ResultWithError1 arcs::common_interface::RobotAlgorithm::forwardKinematicsAll (
    const std::vector< double > & q)
```

运动学正解，基于激活的TCP 偏移 (最近的通过 setTcpOffset 设置的参数) 输入关节角度，输出各连杆位姿

参数

q	关节角

返回

各个连杆位姿和正解结果是否有效返回值的第一个参数为正解结果，第二个为正解错误码，错误码返回值列表如下 0 - 成功 -1 - 机械臂状态不对 (未初始化完成，可尝试再次调用) -5 - 输入的关节角无效 (维度错误)

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
forwardKinematicsAll(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float]) -> Tuple[List[List[float]], int]
```

Lua 函数原型

```
forwardKinematicsAll(q: table) -> table, number
```

Lua 示例

```
poses, fk_result = forwardKinematicsAll({3.083,1.227,1.098,0.670,-1.870,-0.397})
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.forwardKinematicsAll", "params": { "q": [-7.32945e-11, -0.261799, 1.74533, 0.436332, 1.5708, -2.14136e-10] }, "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": [[ [0.0010004, 0.0, 0.122, 0.0, 0.0, 3.141592653589793], [0.0010003999910823934, -0.12166795404953117, 0.12205392567412496, 1.5690838552993878, -1.3089969602251492, 0.0016541204887245322], [0.10659809449330016, -0.12166124765791932, 0.5161518260534821, -1.5718540206872664, 0.43633111081725523, -0.002370419930605744], [0.4482255342315581, -0.1226390142692, 0.3568490758201224, -0.000788980482890453, 1.5665204619271216, -1.5719618592940798], [0.5519603828181741, -0.12282266347465487, 0.35639753697117343, 1.5707938201843654, 0.0042721909279596635, 1.569044569428874], [0.5513243939877184, -0.12401224959696328, 0.2615193425222568, -3.1349118101994096, 0.004272190926529066, 1.569044569643007], [0.7494883076798231, -0.025647479212994567, -0.04023458911333461, -3.1349118101994096, 0.004272190926529066, 1.569044569643007], [0] ] ], 0 ] }
```

forwardToolKinematics()

```
ResultWithError arcs::common_interface::RobotAlgorithm::forwardToolKinematics (
    const std::vector< double > & q)
运动学正解 (忽略 TCP 偏移值)
```

参数

q	关节角

返回

法兰盘中心位姿和正解结果是否有效返回值的第一个参数为正解结果，第二个为正解错误码，错误码返回值列表如下 0 - 成功 -1 - 机械臂状态不对 (未初始化完成, 可尝试再次调用) -5 - 输入的关节角无效 (维度错误)

异常

<code>arcs::common_interface::AuboException</code>	

Lua 函数原型

`forwardToolKinematics(q: table) -> table, number`

Lua 示例

```
pose, fk_result = forwardToolKinematics({3.083,1.227,1.098,0.670,-1.870,-0.397})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.forwardToolKinematics","params":[[3.083688522170976,1.2273215976885394,1.098072739631141,0.6705738810610149,-1.870715392248607,-0.39708546603119627],{"id":1}]}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[[0.5881351149440136,0.05323734739426938,0.623922550656701,2.4599818776908724,0.4789772388601265,1.6189630435878408],0]}
```

frictionModelIdentify()

```
bool arcs::common_interface::RobotAlgorithm::frictionModelIdentify (
    const std::vector< std::vector< double > > & q,
    const std::vector< std::vector< double > > & qd,
    const std::vector< std::vector< double > > & qdd,
    const std::vector< std::vector< double > > & temp)
```

关节摩擦力模型辨识算法接口

参数

q	关节角度
qd	关节速度
qdd	关节加速度
$temp$	关节温度

返回

是否辨识成功

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
frictionModelIdentify(self: pyaubo_sdk.RobotAlgorithm, arg0: List[List[float]], arg1: List[List[←
[float]], arg2: List[List[float]], arg3: List[List[float]]) -> bool
```

Lua 函数原型

```
frictionModelIdentify(q: table, qd: table, qdd: table, temp: table) -> boolean
```

Lua 示例

```
Identify_result = frictionModelIdentify({3.083,1.227,1.098,0.670,-1.870,-0.397}, {10.0,10.0,10.0,10.←
0,10.0,10.0},{20.0,20.0,20.0,20.0,20.0,20.0},{30.0,30.0,30.0,30.0,30.0,30.0})
```

generatePayloadIdentifyTraj()

```
int arcs::common_interface::RobotAlgorithm::generatePayloadIdentifyTraj (
    const std::string & name,
    const TrajConfig & traj_conf)
```

生成用于负载辨识的激励轨迹此接口内部调用 pathBufferAppend 将离线轨迹存入 buffer 中，后续可通过 movePathBuffer 运行离线轨迹

参数

<i>name</i>	轨迹名字
<i>traj_conf</i>	各关节轨迹的限制条件 traj_conf.move_axis: 运动的轴由于实际用户现场可能不希望在负载辨识时控制机械臂多关节大幅度运动，故最好选用 traj_conf.move_axis=LoadIdentifyMoveAxis::Joint_4_6; traj_conf.init_joint: 运动初始关节角，为了避免关节 5 接近零位时的奇异问题，应设置 traj_conf.init_joint[4]的绝对值不小于 0.3(rad)，接近 1.57(rad) 为宜。其余关节的关节角可任意设置 traj_conf.lower_joint_bound, traj_conf.upper_joint_bound: 关节角上下限，维度应与 config.move_axis 维度保持一致，推荐设置 upper_joint_bound 为 2，lower_joint_bound 为-2 config.max_velocity, config.max_acceleration: 关节角速度角加速度限制，维度应与 config.move_axis 维度保持一致，出于安全和驱动器跟随性能的考虑，推荐设置 max_velocity=3,max_acceleration=5

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

getPayloadIdentifyResult()

`Payload` arcs::common_interface::RobotAlgorithm::getPayloadIdentifyResult ()
 获取负载辨识结果

返回

异常

<code>arcs::common_interface::AuboException</code>

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.getPayloadIdentifyResult", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": [0.0, [], [], []]}
```

getRobotConfiguration()

`ResultWithErrno3` arcs::common_interface::RobotAlgorithm::getRobotConfiguration (
 const std::vector< double > & q)

根据输入的关节角计算并返回对应的机械臂构型
 机械臂构型由三个维度的状态组合而成，各维度定义如下：

- 肩部方向：LEFT(肩部朝左) / RIGHT(肩部朝右)
- 肘部方向：UP(肘部朝上) / DOWN(肘部朝下)
- 腕部状态：FLIP(腕部翻转) / NOFLIP(腕部不翻转)

三个维度两两组合共形成 8 种基础构型 (L/R + U/D + F/N)，该接口会根据输入的关节角解析出当前对应的构型类型，并返回其枚举值对应的整数形式及错误码（注：接口返回的是组合构型值，如LUF对应 0，而非单独的LEFT/UP/FLIP 枚举值）。

参数

<code>q</code>	输入的关节角数组，6 轴机械臂为 6 个元素，单位：弧度 (rad)

返回

机械臂构型结果及错误码（类型为ResultWithErrno3，即 std::tuple<int, int>）：

- 第一个 int：机械臂构型枚举 (RobotConfiguration) 对应的整数值，取值范围及含义：-1 (NONE) - 无效构型 0 (LUF) - LEFT+UP+FLIP (左肩、肘上、腕翻转) 1 (LUN) - LEFT+UP+NOFLIP (左肩、肘上、腕不翻转) 2 (LDF) - LEFT+DOWN+FLIP (左肩、肘下、腕翻转) 3 (LDN) - LEFT+DOWN+NOFLIP (左肩、肘下、腕不翻转) 4 (RUF) - RIGHT+UP+FLIP (右肩、肘上、腕翻转) 5 (RUN) - RIGHT+UP+NOFLIP (右肩、肘上、腕不翻转) 6 (RDF) - RIGHT+DOWN+FLIP (右肩、肘下、腕翻转) 7 (RDN) - RIGHT+DOWN+NOFLIP (右肩、肘下、腕不翻转)
- 第二个 int：错误码，错误码含义如下：0 - 成功：构型计算完成，返回有效构型值 -1 - 机械臂状态异常：未初始化完成，可尝试重新初始化后调用 -5 - 输入参数无效：关节角数组维度错误（非 6 个元素）或数值超出合理范围

异常

<code>arcs::common_interface::AuboException</code>	输入参数非法（如空数组、元素数量错误）时抛出

Python 函数原型

```
getRobotConfiguration(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float]) -> Tuple[int, int]
```

Lua 函数原型

```
getRobotConfiguration(q: table) -> number, number
```

Lua 示例

```
- 输入 6 轴关节角（单位：rad），获取构型及错误码 local joint_angles = {3.083,1.227,1.098,0.670,-1.870,-0.397} local config_val, err_code = getRobotConfiguration(joint_angles) if err_code == 0 then print(" 机械臂构型值：", config_val) - 示例输出：4（对应RUF，右肩、肘上、腕翻转） else print(" 获取构型失败，错误码：", err_code) end
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.getRobotConfiguration","params":[[3.083688522170976,1.2273215976885394,1.098072739631141,0.6705738810610149,-1.870715392248607,-0.39708546603119627]],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[4,0]}
```

inverseKinematics()

```
ResultWithErrno arcs::common_interface::RobotAlgorithm::inverseKinematics (
    const std::vector< double > & qnear,
    const std::vector< double > & pose)
```

运动学逆解输入TCP 位姿和参考关节角度，输出关节角度

参数

<i>qnear</i>	参考关节角
<i>pose</i>	TCP 位姿

返回

关节角和逆解结果是否有效返回值的第一个参数为逆解结果，第二个为逆解错误码，错误码返回列表如下 0 - 成功 -1 - 机械臂状态不对（未初始化完成，可尝试再次调用） -5 - 输入的参考关节角或TCP 位姿无效（维度错误） -23 - 逆解计算不收敛，计算出错 -24 - 逆解计算超出机器人最大限制 -25 - 逆解输入配置存在错误 -26 - 逆解雅可比矩阵计算失败 -27 - 目标点存在解析解，但均不满足选解条件 -28 - 逆解返回未知类型错误若错误码非 0，则返回值的第一个参数为输入参考关节角 *qnear*

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
inverseKinematics(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float], arg1: List[float]) -> Tuple[
List[float], int]
```

Lua 函数原型

```
inverseKinematics(qnear: table, pose: table) -> table, int
```

Lua 示例

```
joint,ik_result = inverseKinematics({0,0,0,0,0,0},{0.81665,-0.20419,0.43873,-3.135,0.004,1.569})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.inverseKinematics","params":[[0,0,0,0,0,0],[0.71374,0.08417,0.6708,2.46,0.479,1.619]],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[[3.083688522170976,1.2273215976885394,1.098072739631141,0.6705738810610149,-1.870715392248607,-0.39708546603119627],0]}
```

inverseKinematics1()

```
ResultWithErrno arcs::common_interface::RobotAlgorithm::inverseKinematics1 (
    const std::vector< double > & qnear,
    const std::vector< double > & pose,
    const std::vector< double > & tcp_offset)
```

运动学逆解输入TCP 位姿和参考关节角度，输出关节角度

参数

<i>qnear</i>	参考关节角
<i>pose</i>	TCP 位姿
<i>tcp_offset</i>	TCP 偏移

返回

关节角和逆解结果是否有效，同 inverseKinematics

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
inverseKinematics1(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float], arg1: List[float], arg2: List[float]) -> Tuple[List[float], int]
```

Lua 函数原型

```
inverseKinematics1(qnear: table, pose: table, tcp_offset: table) -> table, int
```

Lua 示例

```
joint,ik_result = inverseKinematics1({0,0,0,0,0,0},{0.81665,-0.20419,0.43873,-3.135,0.004,1.569},{0.04,-0.035,0.1,0,0,0})
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.inverseKinematics1", "params": [[0,0,0,0,0,0],[0.71374,0.08417,0.6708,2.46,0.479,1.619],[0.0, 0.13201,0.03879,0,0,0]], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": [[3.084454549595208,1.2278265883747776,1.0986586440159576,0.6708221281915528,1.8712459848518375,-0.3965111476861782],0]}
```

inverseKinematics2()

```
ResultWithErrno arcs::common_interface::RobotAlgorithm::inverseKinematics2 (
    const std::vector< double > & qnear,
    const std::vector< double > & pose)
```

同构型最小关节变化逆解按照 qnear 的机械臂构型计算目标 TCP 位姿逆解，并在同构型逆解中选择相对 qnear 关节变化最小的一组基于激活的 TCP 偏移 (最近的通过 setTcpOffset 设置的参数)

参数

qnear	参考关节角
pose	TCP 位姿，基于激活的 TCP 偏移 (最近的通过 setTcpOffset 设置的参数)

返回

关节角和逆解结果是否有效返回值的第一个参数为逆解结果，第二个为逆解错误码，错误码返回列表如下 0 - 成功 -1 - 机械臂状态不对 (未初始化完成，可尝试再次调用) -5 - 输入的参考关节角或TCP 位姿无效 (维度错误) -23 - 逆解计算不收敛，计算出错 -24 - 逆解计算超出机器人最大限制 -25 - 逆解输入配置存在错误 -26 - 逆解雅可比矩阵计算失败 -27 - 目标点存在解析解，但均不满足同构型最小关节变化选解条件 -28 - 逆解返回未知类型错误若错误码非 0，则返回值的第一个参数为输入参考关节角 qnear

异常

```
arcs::common_interface::AuboException
```

Python 函数原型

```
inverseKinematics2(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float], arg1: List[float]) -> Tuple[List[float], int]
```

Lua 函数原型

```
inverseKinematics2(qnear: table, pose: table) -> table, int
```

Lua 示例

```
joint,ik_result = inverseKinematics2({0,0,0,0,0,0},{0.81665,-0.20419,0.43873,-3.135,0.004,1.569})
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.inverseKinematics2", "params": [[0,0,0,0,0,0],[0.71374,0.08417,0.6708,2.46,0.479,1.619]], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": [[3.083688522170976,1.2273215976885394,1.098072739631141,0.6705738810610149,-1.870715392248607,-0.39708546603119627],0] }
```

inverseKinematicsAll()

```
ResultWithError1 arcs::common_interface::RobotAlgorithm::inverseKinematicsAll (
    const std::vector< double > & pose)
```

求出所有的逆解, 基于激活的 TCP 偏移

参数

<i>pose</i>	TCP 位姿

返回

关节角和逆解结果是否有效返回的错误码同 `inverseKinematics`

异常

<code>arcs::common_interface::AuboException</code>	

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.inverseKinematicsAll", "params": [[0.71374,0.08417,0.6708,2.46,0.479,1.619]], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": [[[3.083688522170976,1.2273215976885394,1.098072739631141,0.6705738810610149,-1.870715392248607,-0.39708546603119627], [3.081056801097411,0.17985038037652645,-1.0991717292664145,-0.4806460200109001,-1.869182975312333,-0.402066016835411], [0.4090095277807992,-0.1623365054641728,1.081775890307679,0.26993250263224805,0.9738255833642309,0.000572556627720845], [0.4116449425067969,-1.1931664523907126,-1.0822709833775688,-0.8665964106161371,0.9732141569888207,0.006484919654891586],0] ] }
```

inverseKinematicsAll1()

```
ResultWithErrno1 arcs::common_interface::RobotAlgorithm::inverseKinematicsAll1 (
    const std::vector< double > & pose,
    const std::vector< double > & tcp_offset)
```

求出所有的逆解, 基于提供的 TCP 偏移

参数

<i>pose</i>	TCP 位姿
<i>tcp_offset</i>	TCP 偏移

返回

关节角和逆解结果是否有效, 同 inverseKinematicsAll 返回的错误码同 inverseKinematics

异常

<i>arcs::common_interface::AuboException</i>

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.inverseKinematicsAll1", "params": [[0.71374, 0.08417, 0.6708, 2.46, 0.479, 1.619], [0.0, 0.13201, 0.03879, 0, 0, 0]], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": [[[3.084454549595208, 1.2278265883747776, 1.0986586440159576, 0.6708221281915528, -1.8712459848518375, -0.3965111476861782], [3.0818224058231602, 0.17980369843203092, -1.0997576631122077, -0.48102131527371267, -1.8697135490338517, -0.40149459722060593], [0.40972960018231047, -0.16226026285489026, 1.0823403816496, 0.2700204411869427, 0.9734251963887868, 0.0012903686498106507], [0.41236549588802296, -1.193621392918341, -1.0828346680836718, -0.8671097369314354, 0.972815367289568, 0.007206851371073478]], [0]] }
```

inverseToolKinematics()

```
ResultWithErrno arcs::common_interface::RobotAlgorithm::inverseToolKinematics (
    const std::vector< double > & qnear,
    const std::vector< double > & pose)
```

运动学逆解 (忽略 TCP 偏移值)

参数

<i>qnear</i>	参考关节角
<i>pose</i>	法兰盘中心的位姿

返回

关节角和逆解结果是否有效返回值的第一个参数为逆解结果, 第二个为逆解错误码, 错误码返回列表如下 0 - 成功 -1 - 机械臂状态不对 (未初始化完成, 可尝试再次调用) -5 - 输入的参考关节角或位姿无效 (维度错误)

异常

<code>arcs::common_interface::AuboException</code>

Lua 函数原型

```
inverseToolKinematics(qnear: table, pose: table) -> table, int
```

Lua 示例

```
joint, ik_result = inverseToolKinematics({0,0,0,0,0,0},{0.58815,0.0532,0.62391,2.46,0.479,1.619})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.inverseToolKinematics","params":[[0,0,0,0,0,0]↵
,[0.58815,0.0532,0.62391,2.46,0.479,1.619]],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[[3.083609363838651,1.22736129158332,1.098095443698268,0.6705395395487186,↵
1.8706605026855632,-0.39714507002376465],0]}
```

inverseToolKinematicsAll()

```
ResultWithErrno1 arcs::common_interface::RobotAlgorithm::inverseToolKinematicsAll (
    const std::vector< double > & pose)
```

运动学逆解 (忽略 TCP 偏移值)

参数

<i>qnear</i>	参考关节角
<i>pose</i>	法兰盘中心的位姿

返回

关节角和逆解结果是否有效

异常

<code>arcs::common_interface::AuboException</code>

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.inverseToolKinematicsAll","params":[[0.58815,0.↵
0532,0.62391,2.46,0.479,1.619]],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[[[3.083609363838651,1.22736129158332,1.098095443698268,0.6705395395487186,↵
1.8706605026855632,-0.39714507002376465], [3.0809781797426523,0.17987122696706134,-1.0991932793263717,↵
0.4807053707530958,-1.8691282890274434,-0.40212516672751814], [0.40892195618737215,-0.16235398607358653,1.↵
081812753177426,0.27003586475871766,0.9738744130114284,0.00048462518316674287], [0.41155633414333076,↵
1.1932173012004512,-1.082306542045813,-0.8665312056504818,0.9732632365861417,0.0063958311601771175]↵
],0]}
```


needVibrationRecalib()

```
int arcs::common_interface::RobotAlgorithm::needVibrationRecalib (
    const VibrationRecalibrationParameter & param1,
    const VibrationRecalibrationParameter & param2,
    double threshold)
```

判断是否需要重新辨识振动参数

参数

<i>param1</i>	参考参数（如上一次辨识结果）
<i>param2</i>	当前参数（如新测量结果）
<i>threshold</i>	变化阈值（0~1），超过则需重新辨识

返回

>0 需要重新辨识，=0 不需要，<0 出错

Lua 函数原型

```
needVibrationRecalib(param1: table, param2: table, threshold: number) -> number
```

JSON-RPC 示例

```
{ "jsonrpc": "2.0", "method": "VibrationController.needVibrationRecalib", "params": [param1_obj, param2_obj, 0.1], "id": 1 }
```

pathBlend3Points()

```
std::vector< std::vector< double > > arcs::common_interface::RobotAlgorithm::pathBlend3Points (
    int type,
    const std::vector< double > & q_start,
    const std::vector< double > & q_via,
    const std::vector< double > & q_to,
    double r,
    double d)
```

求解交融的轨迹点

参数

<i>type</i>	0-movej 和 movej 1-movej 和 movel 2-movel 和 movej 3-movel 和 movel
<i>q_start</i>	交融前路径的起点
<i>q_via</i>	交融点
<i>q_to</i>	交融后路径的终点
<i>r</i>	在 <i>q_via</i> 处的交融半径
<i>d</i>	采样距离

返回

q_via 处的交融段笛卡尔空间离散轨迹点 (x,y,z) 集合

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
pathBlend3Points(self: pyaubo_sdk.RobotAlgorithm, arg0: int, arg1: List[float], arg2: List[float],
arg3: List[float], arg4: float, arg5: float) -> List[List[float]]
```

Lua 函数原型

```
pathBlend3Points(type: number, q_start: table, q_via: table, q_to: table, r: number, d: number)
-> table, number
```

Lua 示例

```
q_via , num = pathBlend3Points(1,{0.58815,0.0532,0.62391,2.46,0.479,1.619},{0.0,-0.2618,1.7453,0.↵
4364,1.5711,0.0}, {0.3234,-0.5405,1.5403,0.5881,1.2962,0.7435},0.25,0.02)
```

pathMoveS()

```
std::vector< std::vector< double > > arcs::common_interface::RobotAlgorithm::pathMoveS (
    const std::vector< std::vector< double > > & qs,
    double d)
```

求解 moveS 的轨迹点

pathMoveS

参数

<i>qs</i>	样条轨迹生成点集合
<i>d</i>	采样距离

返回**异常**

<code>arcs::common_interface::AuboException</code>

payloadCalculateFinished()

```
int arcs::common_interface::RobotAlgorithm::payloadCalculateFinished ()
负载辨识是否计算完成
```

返回

完成返回 0; 正在进行中返回 1; 计算失败返回 <0;

异常

<code>arcs::common_interface::AuboException</code>

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.payloadCalculateFinished", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

payloadIdentify()

```
int arcs::common_interface::RobotAlgorithm::payloadIdentify (
    const std::string & data_file_no_payload,
    const std::string & data_file_with_payload)
```

基于电流的负载辨识算法接口

需要采集空载时运行激励轨迹的位置、速度、电流以及带负载时运行激励轨迹的位置、速度、电流

参数

<code>data_file_no_payload</code>	空载时运行激励轨迹各关节数据的文件路径（.csv 格式），共 18 列，依次为 6 个关节位置、6 个关节速度、6 个关节电流
<code>data_file_with_payload</code>	带负载运行激励轨迹各关节数据的文件路径（.csv 格式），共 18 列，依次为 6 个关节位置、6 个关节速度、6 个关节电流

返回

辨识的结果

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
payloadIdentify(self: pyaubo_sdk.RobotAlgorithm, arg0: List[List[float]], arg1: List[List[float]]) ->
Tuple[List[float], List[float], float, List[float]]
```

Lua 函数原型

```
payloadIdentify(data_with_payload: table, data_with_payload: table) -> table
```

payloadIdentify1()

```
int arcs::common_interface::RobotAlgorithm::payloadIdentify1 (
    const std::string & file_name)
```

新版基于电流的负载辨识算法接口

需要采集带载时运行最少三个点的位置、速度、加速度、电流、温度、末端传感器数据、底座数据

参数

<i>data</i>	带负载的各关节数据的文件路径（.csv 格式），共 42 列，末端传感器数据、底座数据默认为 0

返回

辨识的结果

payloadIdentifyTrajGenFinished()

```
int arcs::common_interface::RobotAlgorithm::payloadIdentifyTrajGenFinished ()
```

负载辨识轨迹是否生成完成

返回

完成返回 0; 正在进行中返回 1; 计算失败返回 <0;

异常

<i>arcs::common_interface::AuboException</i>	

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.payloadIdentifyTrajGenFinished", "params": [], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

removeCollisionObject()

```
int arcs::common_interface::RobotAlgorithm::removeCollisionObject (
    const std::string & name)
```

删除指定的碰撞对象

参数

<i>name</i>	需要删除的碰撞对象名称

返回

0 表示成功，非 0 表示失败

Lua 函数原型

```
removeCollisionObject(name) -> integer
```

Lua 示例

```
removeCollisionObject("tool_box")
```

Python 函数原型

```
removeCollisionObject(name: str) -> int
```

Python 示例

```
robot_algorithm.removeCollisionObject("tool_box")
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.removeCollisionObject", "params": ["tool_box"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

validatePath()

```
int arcs::common_interface::RobotAlgorithm::validatePath (
    int type,
    const std::vector< double > & start,
    double r1,
    const std::vector< double > & end,
    double r2,
    double d)
```

验证机器人运动路径从起点到终点的可达性，不启用外部碰撞检测。
该接口保留用于兼容旧版 SDK、Lua、JSON-RPC 和 ROS 服务调用。

参数

<i>type</i>	路径类型标识，用于指定起点和终点的数据类型：0 - 起点：关节角，终点：关节角 1 - 起点：位姿，终点：位姿
<i>start</i>	路径起点数据
<i>r1</i>	起点交融半径，单位：m（米）
<i>end</i>	路径终点数据
<i>r2</i>	终点交融半径，单位：m（米）
<i>d</i>	路径采样间隔，单位：m（米），间隔越小验证精度越高，但耗时越长

返回

路径可达性结果码

validatePath1()

```
int arcs::common_interface::RobotAlgorithm::validatePath1 (
    int type,
    const std::vector< double > & start,
    double r1,
    const std::vector< double > & end,
    double r2,
    double d,
    bool check_external_collision)
```

验证机器人运动路径从起点到终点的可达性，可选择是否启用外部碰撞检测。

该接口通过采样的方式验证指定路径是否存在超限、自碰撞、外部碰撞、奇异等不可达情况，支持关节角 <-> 关节角、位姿 <-> 位姿两种路径类型验证。

参数

<i>type</i>	路径类型标识, 用于指定起点和终点的数据类型: 0 - 起点: 关节角, 终点: 关节角 1 - 起点: 位姿, 终点: 位姿
<i>start</i>	路径起点数据
<i>r1</i>	起点交融半径, 单位: m (米)
<i>end</i>	路径终点数据
<i>r2</i>	终点交融半径, 单位: m (米)
<i>d</i>	路径采样间隔, 单位: m (米), 间隔越小验证精度越高, 但耗时越长
<i>check_external_collision</i>	是否启用外部碰撞检测: false - 保持原有路径可达性与自碰撞校验 true - 在原有校验基础上增加与外部对象的碰撞检测

返回

路径可达性结果码: 0 - 可达: 路径无异常, 可正常运动 -18 - 路径中存在关节超限/笛卡尔空间超限 -21 - 轨迹生成失败 -22 - 路径中机器人本体发生自碰撞 -24 - 路径中经过机器人奇异位形 -27 - 目标点有解但超出关节限位

异常

<code>arcs::common_interface::AuboException</code>

Lua 函数原型

```
validatePath1(type, start, r1, end, r2, d, check_external_collision) -> number
```

Lua 示例

```
local result = validatePath1(0, start_joint, 0.01, end_joint, 0.01, 0.05, true)
```

Python 函数原型

```
validatePath1(type: int, start: list[float], r1: float, end: list[float], r2: float, d: float, check_external_collision: bool) -> int
```

Python 示例

```
result = validatePath1(1, start_joint, 0.01, end_pose, 0.01, 0.05, True)
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.validatePath1", "params": [0, [0.0, 0.0, 1.57, 0.0, 1.57, 0.0], 0.01, [0.52, 0.0, 1.57, 0.0, 1.57, 0.0], 0.01, 0.05, true], "id": 1 }
```

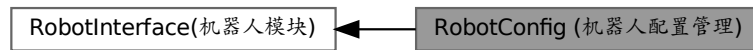
JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

8.6.7 RobotConfig (机器人配置管理)

RobotConfig

RobotConfig (机器人配置管理) 的协作图:



函数

- `std::string arcs::common_interface::RobotConfig::getName ()`
获取机器人的名字
- `int arcs::common_interface::RobotConfig::getDof ()`
获取机器人的自由度 (从硬件抽象层读取)
- `double arcs::common_interface::RobotConfig::getCycletime ()`
获取机器人的伺服控制周期 (从硬件抽象层读取)
- `int arcs::common_interface::RobotConfig::setSlowDownFraction (int level, double fraction)`
预设缓速模式下的速度缩减比例
- `double arcs::common_interface::RobotConfig::getSlowDownFraction (int level)`
获取预设的缓速模式下的速度缩减比例
- `double arcs::common_interface::RobotConfig::getDefaultToolAcc ()`
获取默认的工具端加速度, 单位 m/s^2
- `double arcs::common_interface::RobotConfig::getDefaultToolSpeed ()`
获取默认的工具端速度, 单位 m/s
- `double arcs::common_interface::RobotConfig::getDefaultJointAcc ()`
获取默认关节加速度, 单位 rad/s^2
- `double arcs::common_interface::RobotConfig::getDefaultJointSpeed ()`
获取默认关节速度, 单位 rad/s
- `std::string arcs::common_interface::RobotConfig::getRobotType ()`
获取机器人类型代码
- `std::string arcs::common_interface::RobotConfig::getRobotSubType ()`
获取机器人子类型代码
- `std::string arcs::common_interface::RobotConfig::getControlBoxType ()`
获取控制柜类型代码
- `int arcs::common_interface::RobotConfig::setMountingPose (const std::vector< double > &pose)`
设置安装位姿 (机器人的基坐标系相对于世界坐标系) $world \rightarrow base$
- `std::vector< double > arcs::common_interface::RobotConfig::getMountingPose ()`
获取安装位姿 (机器人的基坐标系相对于世界坐标系)
- `int arcs::common_interface::RobotConfig::attachRobotBaseTo (const std::string &frame)`
将机器人绑定到一个坐标系, 如果这个坐标系是运动的, 那机器人也会跟着运动应用于地轨或者龙门这个函数调用的时候 $frame$ 和 $ROBOTBASE$ 的相对关系就固定了
- `int arcs::common_interface::RobotConfig::setWorkObjectData (const WObjectData &wobj)`
设置工件数据, 编程点位都是基于工件坐标系
- `int arcs::common_interface::RobotConfig::setCollisionLevel (int level)`
设置碰撞灵敏度等级数值越大越灵敏
- `int arcs::common_interface::RobotConfig::getCollisionLevel ()`
获取碰撞灵敏度等级
- `int arcs::common_interface::RobotConfig::setCollisionStopType (int type)`

- 设置碰撞停止类型
- int `arcs::common_interface::RobotConfig::getCollisionStopType ()`
获取碰撞停止类型
- int `arcs::common_interface::RobotConfig::setHomePosition (const std::vector< double > &positions)`
设置机器人的原点位置
- std::vector< double > `arcs::common_interface::RobotConfig::getHomePosition ()`
获取机器人的原点位置
- int `arcs::common_interface::RobotConfig::setFreedriveDamp (const std::vector< double > &damp)`
设置拖动阻尼
- std::vector< double > `arcs::common_interface::RobotConfig::getFreedriveDamp ()`
获取拖动阻尼
- int `arcs::common_interface::RobotConfig::setHandguidDamp (const std::vector< double > &damp)`
设置混合拖动阻尼
- std::vector< double > `arcs::common_interface::RobotConfig::getHandguidDamp ()`
获取混合拖动阻尼
- std::unordered_map< std::string, std::vector< double > > `arcs::common_interface::RobotConfig::getKinematicsParameters (bool real)`
获取机器人DH 参数 α a d θ β
- std::unordered_map< std::string, std::vector< double > > `arcs::common_interface::RobotConfig::getKinematicsCompensate (double ref_temperature)`
获取指定温度下的DH 参数补偿值: α a d θ β
- int `arcs::common_interface::RobotConfig::setKinematicsCompensate (const std::unordered_map< std::string, std::vector< double > > &param)`
设置标准 DH 补偿到机器人
- int `arcs::common_interface::RobotConfig::setPersistentParameters (const std::string &param)`
设置需要保存到接口板底座的参数
- int `arcs::common_interface::RobotConfig::setHardwareCustomParameters (const std::string &param)`
设置硬件抽象层自定义参数目的是为了做不同硬件之间的兼容
- std::string `arcs::common_interface::RobotConfig::getHardwareCustomParameters (const std::string &param)`
获取硬件抽象层自定义参数
- int `arcs::common_interface::RobotConfig::setRobotZero ()`
设置机器人关节零位
- std::vector< std::string > `arcs::common_interface::RobotConfig::getTcpForceSensorNames ()`
获取可用的末端力矩传感器的名字
- int `arcs::common_interface::RobotConfig::selectTcpForceSensor (const std::string &name)`
设置末端力矩传感器如果存在内置的末端力矩传感器，默认将使用内置的力矩传感器
- int `arcs::common_interface::RobotConfig::setTcpForceSensorPose (const std::vector< double > &sensor_pose)`
设置传感器安装位姿
- std::vector< double > `arcs::common_interface::RobotConfig::getTcpForceSensorPose ()`
获取传感器安装位姿
- bool `arcs::common_interface::RobotConfig::hasTcpForceSensor ()`
是否安装了末端力矩传感器
- int `arcs::common_interface::RobotConfig::setTcpForceOffset (const std::vector< double > &force_offset)`
设置末端力矩偏移
- std::vector< double > `arcs::common_interface::RobotConfig::getTcpForceOffset ()`
获取末端力矩偏移
- std::vector< std::string > `arcs::common_interface::RobotConfig::getBaseForceSensorNames ()`
获取可用的底座力矩传感器的名字

- `int arcs::common_interface::RobotConfig::selectBaseForceSensor (const std::string &name)`
设置底座力矩传感器如果存在内置的底座力矩传感器，默认将使用内置的力矩传感器
- `bool arcs::common_interface::RobotConfig::hasBaseForceSensor ()`
是否安装了底座力矩传感器
- `int arcs::common_interface::RobotConfig::setBaseForceOffset (const std::vector< double > &force←_offset)`
设置底座力矩偏移
- `std::vector< double > arcs::common_interface::RobotConfig::getBaseForceOffset ()`
获取底座力矩偏移
- `uint32_t arcs::common_interface::RobotConfig::getSafetyParametersChecksum ()`
获取安全参数校验码 *CRC32*
- `int arcs::common_interface::RobotConfig::confirmSafetyParameters (const RobotSafetyParameterRange &parameters)`
发起确认安全配置参数请求：将安全配置参数写入到安全接口板 *flash* 或文件
- `uint32_t arcs::common_interface::RobotConfig::calcSafetyParametersChecksum (const RobotSafetyParameterRange &parameters)`
计算安全参数的 *CRC32* 校验值
- `std::vector< double > arcs::common_interface::RobotConfig::getJointMaxPositions ()`
获取关节最大位置（物理极限）
- `std::vector< double > arcs::common_interface::RobotConfig::getJointMinPositions ()`
获取关节最小位置（物理极限）
- `std::vector< double > arcs::common_interface::RobotConfig::getJointMaxSpeeds ()`
获取关节最大速度（物理极限）
- `std::vector< double > arcs::common_interface::RobotConfig::getJointMaxAccelerations ()`
获取关节最大加速度（物理极限）
- `std::vector< double > arcs::common_interface::RobotConfig::getTcpMaxSpeeds ()`
获取 *TCP* 最大速度（物理极限）
- `std::vector< double > arcs::common_interface::RobotConfig::getTcpMaxAccelerations ()`
获取 *TCP* 最大加速度（物理极限）
- `int arcs::common_interface::RobotConfig::setGravity (const std::vector< double > &gravity)`
设置机器人安装姿态
- `std::vector< double > arcs::common_interface::RobotConfig::getGravity ()`
获取机器人的安装姿态
- `int arcs::common_interface::RobotConfig::setTcpOffset (const std::vector< double > &offset)`
设置当前的 *TCP* 偏移
- `std::vector< double > arcs::common_interface::RobotConfig::getTcpOffset ()`
获取当前的 *TCP* 偏移
- `int arcs::common_interface::RobotConfig::setToolInertial (double m, const std::vector< double > &com, const std::vector< double > &inertial)`
设置工具端质量、质心及惯量
- `int arcs::common_interface::RobotConfig::setPayload (double m, const std::vector< double > &cog, const std::vector< double > &aom, const std::vector< double > &inertia)`
设置有效负载
- `Payload arcs::common_interface::RobotConfig::getPayload ()`
获取有效负载
- `bool arcs::common_interface::RobotConfig::toolSpaceInRange (const std::vector< double > &pose)`
末端位姿是否在安全范围之内
- `int arcs::common_interface::RobotConfig::firmwareUpdate (const std::string &fw)`
发起固件升级请求，控制器软件将进入固件升级模式
- `std::tuple< std::string, double > arcs::common_interface::RobotConfig::getFirmwareUpdateProcess ()`

获取当前的固件升级的进程

- `std::vector< double > arcs::common_interface::RobotConfig::getLimitJointMaxPositions ()`
获取关节最大位置 (当前正在使用的限制值)
- `std::vector< double > arcs::common_interface::RobotConfig::getLimitJointMinPositions ()`
获取关节最小位置 (当前正在使用的限制值)
- `std::vector< double > arcs::common_interface::RobotConfig::getLimitJointMaxSpeeds ()`
获取关节最大速度 (当前正在使用的限制值)
- `std::vector< double > arcs::common_interface::RobotConfig::getLimitJointMaxAccelerations ()`
获取关节最大加速度 (当前正在使用的限制值)
- `double arcs::common_interface::RobotConfig::getLimitTcpMaxSpeed ()`
获取TCP 最大速度 (当前正在使用的限制值)
- `SafeguedStopType arcs::common_interface::RobotConfig::getSafeguardStopType ()`
获取当前安全停止的类型
- `int arcs::common_interface::RobotConfig::getSafeguardStopSource ()`
按位获取完整的安全停止触发源
- `int arcs::common_interface::RobotConfig::getRobotEmergencyStopSource ()`
按位获取完整的机器人紧急停止触发源
- `std::string arcs::common_interface::RobotConfig::getSelectedTcpForceSensorName ()`
获取工具端力矩传感器的名字
- `int arcs::common_interface::RobotConfig::attachWeldingGun (double m, const std::vector< double > &cog, const std::vector< double > &inertia)`
添加焊枪动力学参数
- `int arcs::common_interface::RobotConfig::setCollisionThreshold (const std::vector< double > &threshold)`
设置碰撞阈值
- `std::vector< double > arcs::common_interface::RobotConfig::getCollisionThreshold ()`
获取碰撞阈值
- `int arcs::common_interface::RobotConfig::enableEndCollisionCheck (bool enable)`
使能末端碰撞检测
- `bool arcs::common_interface::RobotConfig::isEndCollisionCheckEnabled ()`
获取末端碰撞检测是否使能

详细描述

RobotConfig

函数说明

attachRobotBaseTo()

```
int arcs::common_interface::RobotConfig::attachRobotBaseTo (
    const std::string & frame)
```

将机器人绑定到一个坐标系, 如果这个坐标系是运动的, 那机器人也会跟着运动应用于地轨或者龙门
这个函数调用的时候 frame 和 ROBOTBASE 的相对关系就固定了

attachWeldingGun()

```
int arcs::common_interface::RobotConfig::attachWeldingGun (
    double m,
    const std::vector< double > & cog,
    const std::vector< double > & inertia)
```

添加焊枪动力学参数

参数

<i>m</i>	质量, 单位: kg
<i>cog</i>	质心, 单位: m, 形式为 (CoGx, CoGy, CoGz)
<i>inertia</i>	惯性张量, 单位: kg*m ² , 形式为 (Ixx, Iyy, Izz, Ixy, Ixz, Iyz) 或 (Ixx, Ixy, Ixz, Iyx, Iyy, Iyz, Izx, Izy, Izz)

返回

成功返回 0; 失败返回错误码

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
attachWeldingGun(self: pyaubo_sdk.RobotConfig, arg0: float, arg1: List[float], arg2: List[float])
-> int
```

Lua 函数原型

```
attachWeldingGun(m: number, cog: table, inertia: table) -> nil
```

Lua 示例

```
attachWeldingGun(3, {0,0,0}, {0,0,0,0,0,0,0,0,0})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.attachWeldingGun","params":[3,[0,0,0],[0,0,0,0,0,0,0,0,0]←
],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

calcSafetyParametersChecksum()

```
uint32_t arcs::common_interface::RobotConfig::calcSafetyParametersChecksum (
    const RobotSafetyParameterRange & parameters)
计算安全参数的 CRC32 校验值
```

返回

crc32

异常

<i>arcs::common_interface::AuboException</i>

confirmSafetyParameters()

```
int arcs::common_interface::RobotConfig::confirmSafetyParameters (
    const RobotSafetyParameterRange & parameters)
发起确认安全配置参数请求. 将安全配置参数写入到安全接口板 flash 或文件
```

参数

<i>parameters</i>	安全配置参数

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
confirmSafetyParameters(self: pyaubo_sdk.RobotConfig, arg0: arcs::common_interface::RobotSafetyParameterRan  
-> int
```

Lua 函数原型**enableEndCollisionCheck()**

```
int arcs::common_interface::RobotConfig::enableEndCollisionCheck (  
    bool enable)  
使能末端碰撞检测
```

参数

<i>enable</i>	使能碰撞阈值

返回

成功返回 0; 失败返回错误码

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
enableEndCollisionCheck(self: pyaubo_sdk.RobotConfig, arg0: bool) -> int
```

Lua 函数原型

```
enableEndCollisionCheck(threshold: table) -> nil
```

Lua 示例

```
enableEndCollisionCheck(true)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.enableEndCollisionCheck","params":[true],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

firmwareUpdate()

```
int arcs::common_interface::RobotConfig::firmwareUpdate (
```

```
    const std::string & fw)
```

发起固件升级请求，控制器软件将进入固件升级模式

参数

<i>fw</i>	固件升级路径。该路径的格式为: 固件安装包路径 # 升级节点列表。其中固件安装包路径和升级节点列表以井号 (#) 分隔，升级节点以逗号 (,) 分隔。如果节点名称后带有!，则表示强制 (不带版本校验) 升级该节点；反之，则表示带校验版本本地升级节点，即在升级该节点前，会先判断当前版本和目标版本是否相同，如果相同就不升级该节点。 可以根据实际需求灵活设置需要升级的节点。 例如， /tmp/firmware_update-1.0.42-rc.5+2347b0d.firm#master_mcu!,slace_mcu!,base!,tool!,joint1!,joint2!,joint3!,joint4!,joint5!,joint6! 表示强制升级接口板主板、接口板从板、基座、工具和 6 个关节 (joint1 至 joint6)。 all 表示所有的节点，例如 /tmp/firm_XXX.firm#all 表示带校验版本本地升级全部节点， /tmp/firm_XXX.firm#all! 表示强制升级全部节点

返回

指令下发成功返回 0; 失败返回错误码。

-AUBO_BAD_STATE: 运行时 (RuntimeMachine) 的当前状态不是Stopped, 固件升级请求被拒绝。AUBO_BAD_STATE 的值是 1。

-AUBO_TIMEOUT: 超时。AUBO_TIMEOUT 的值是 4。

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
firmwareUpdate(self: pyaubo_sdk.RobotConfig, arg0: str) -> int
```

Lua 函数原型

```
firmwareUpdate(fw: string) -> nil
```

Lua 示例

```
firmwareUpdate("/tmp/firmware_update-1.0.42-rc.12+3e33eac.firm#master_mcu")
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.firmwareUpdate","params":["/tmp/firmware_update-1.0.42-rc.12+3e33eac.firm#master_mcu"],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getBaseForceOffset()

```
std::vector< double > arcs::common_interface::RobotConfig::getBaseForceOffset ()  
获取底座力矩偏移
```

返回

返回底座力矩偏移

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
getBaseForceOffset(self: pyaubo_sdk.RobotConfig) -> List[float]
```

Lua 函数原型

```
getBaseForceOffset\(\) -> table
```

Lua 示例

```
BaseForceOffset = getBaseForceOffset\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getBaseForceOffset","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[]}
```

getBaseForceSensorNames()

```
std::vector< std::string > arcs::common_interface::RobotConfig::getBaseForceSensorNames ()  
获取可用的底座力矩传感器的名字
```

返回

返回可用的底座力矩传感器的名字

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getBaseForceSensorNames(self: pyaubo_sdk.RobotConfig) -> List[str]
```

Lua 函数原型

```
getBaseForceSensorNames() -> table
```

Lua 示例

```
BaseForceSensorNames = getBaseForceSensorNames()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getBaseForceSensorNames","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[]}
```

getCollisionLevel()

```
int arcs::common_interface::RobotConfig::getCollisionLevel ()
```

获取碰撞灵敏度等级

返回

碰撞灵敏度等级

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getCollisionLevel(self: pyaubo_sdk.RobotConfig) -> int
```

Lua 函数原型

```
getCollisionLevel() -> number
```

Lua 示例

```
CollisionLevel = getCollisionLevel()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getCollisionLevel","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":6}
```

getCollisionStopType()

```
int arcs::common_interface::RobotConfig::getCollisionStopType ()
```

获取碰撞停止类型

返回

返回碰撞停止类型
 0: 碰撞后浮动, 即碰撞之后进入拖动示教模式
 1: 碰撞后静止
 2: 碰撞后抱闸

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
getCollisionStopType(self: pyaubo_sdk.RobotConfig) -> int
```

Lua 函数原型

```
getCollisionStopType() -> number
```

Lua 示例

```
CollisionStopType = getCollisionStopType()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getCollisionStopType","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":1}
```

getCollisionThreshold()

```
std::vector< double > arcs::common_interface::RobotConfig::getCollisionThreshold ()
```

获取碰撞阈值

返回

碰撞阈值

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
getCollisionThreshold(self: pyaubo_sdk.RobotConfig) -> List[float]
```


Lua 函数原型

`getCollisionThreshold()` -> table

Lua 示例

`CollisionThreshold = getCollisionThreshold()`

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.MotionControl.getCollisionThreshold", "params": [99.0, 114.0, 103.0, 56.0, 51.0, 60.0], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 1 }
```

getControlBoxType()

`std::string arcs::common_interface::RobotConfig::getControlBoxType ()`
获取控制柜类型代码

返回

控制柜类型代码

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

`getControlBoxType(self: pyaubo_sdk.RobotConfig) -> str`

Lua 函数原型

`getControlBoxType()` -> string

Lua 示例

`ControlBoxType = getControlBoxType()`

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.RobotConfig.getControlBoxType", "params": [], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": "cb_ISStation" }
```

getCycletime()

`double arcs::common_interface::RobotConfig::getCycletime ()`
获取机器人的伺服控制周期 (从硬件抽象层读取)

返回

机器人的伺服控制周期

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getCycletime(self: pyaubo_sdk.RobotConfig) -> float
```

Lua 函数原型

```
getCycletime() -> number
```

Lua 示例

```
robot_Cycletime = getCycletime()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotConfig.getCycletime", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0.005}
```

getDefaultJointAcc()

```
double arcs::common_interface::RobotConfig::getDefaultJointAcc ()
```

获取默认关节加速度，单位 rad/s²

返回

默认关节加速度

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getDefaultJointAcc(self: pyaubo_sdk.RobotConfig) -> float
```

Lua 函数原型

```
getDefaultJointAcc() -> number
```

Lua 示例

```
DefaultJointAcc = getDefaultJointAcc()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotConfig.getDefaultJointAcc", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0.0}
```

getDefaultJointSpeed()

double arcs::common_interface::RobotConfig::getDefaultJointSpeed ()
 获取默认关节速度, 单位 rad/s

返回

默认关节速度

异常

arcs::common_interface::AuboException

Python 函数原型

getDefaultJointSpeed(self: pyaubo_sdk.RobotConfig) -> float

Lua 函数原型

getDefaultJointSpeed() -> number

Lua 示例

DefaultJointSpeed = getDefaultJointSpeed()

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.RobotConfig.getDefaultJointSpeed", "params": [], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0.0 }
```

getDefaultToolAcc()

double arcs::common_interface::RobotConfig::getDefaultToolAcc ()
 获取默认的工具端加速度, 单位 m/s^2

返回

默认的工具端加速度

异常

arcs::common_interface::AuboException

Python 函数原型

getDefaultToolAcc(self: pyaubo_sdk.RobotConfig) -> float

Lua 函数原型

getDefaultToolAcc() -> number

Lua 示例

```
DefaultToolAcc = getDefaultToolAcc\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotConfig.getDefaultToolAcc", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0.0}
```

getDefaultToolSpeed()

```
double arcs::common_interface::RobotConfig::getDefaultToolSpeed ()
```

获取默认的工具端速度, 单位 m/s

返回

默认的工具端速度

异常

arcs::common_interface::AuboException	

Python 函数原型

```
getDefaultToolSpeed(self: pyaubo_sdk.RobotConfig) -> float
```

Lua 函数原型

```
getDefaultToolSpeed\(\) -> number
```

Lua 示例

```
DefaultToolSpeed = getDefaultToolSpeed\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotConfig.getDefaultToolSpeed", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0.0}
```

getDof()

```
int arcs::common_interface::RobotConfig::getDof ()
```

获取机器人的自由度 (从硬件抽象层读取)

返回

返回机器人的自由度

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getDof(self: pyaubo_sdk.RobotConfig) -> int
```

Lua 函数原型

```
getDof() -> number
```

Lua 示例

```
robot_dof = getDof()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getDof","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":6}
```

getFirmwareUpdateProcess()

```
std::tuple< std::string, double > arcs::common_interface::RobotConfig::getFirmwareUpdateProcess ()
```

获取当前的固件升级的进程

返回

当前的固件升级进程。
第一个元素表示步骤名称。如果是 failed, 则表示固件升级失败
第二个元素表示升级的进度 (0~1), 完成之后, 返回 ("", 1)

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getFirmwareUpdateProcess(self: pyaubo_sdk.RobotConfig) -> Tuple[str, float]
```

Lua 函数原型

```
getFirmwareUpdateProcess() -> table
```

Lua 示例

```
step,progress = getFirmwareUpdateProcess()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getFirmwareUpdateProcess","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":["",0.0]}
```

getFreedriveDamp()

```
std::vector< double > arcs::common_interface::RobotConfig::getFreedriveDamp ()
```

获取拖动阻尼

返回

拖动阻尼

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
getFreedriveDamp(self: pyaubo_sdk.RobotConfig) -> List[float]
```

Lua 函数原型

```
getFreedriveDamp() -> table
```

Lua 示例

```
FreedriveDamp = getFreedriveDamp()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getFreedriveDamp","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.5,0.5,0.5,0.5,0.5,0.5]}
```

getGravity()

```
std::vector< double > arcs::common_interface::RobotConfig::getGravity ()
```

获取机器人的安装姿态
如果机器人底座安装了姿态传感器，则从传感器读取数据，否则按照用户设置

返回

返回安装姿态

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
getGravity(self: pyaubo_sdk.RobotConfig) -> List[float]
```

Lua 函数原型

```
getGravity() -> table
```

Lua 示例

```
Gravity = getGravity\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getGravity","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.0,0.0,-9.87654321]}
```

getHandguidDamp()

```
std::vector< double > arcs::common_interface::RobotConfig::getHandguidDamp ()  
获取混合拖动阻尼
```

返回

拖动阻尼

异常

arcs::common_interface::AuboException

Python 函数原型

```
getHandguidDamp(self: pyaubo_sdk.RobotConfig) -> List[float]
```

Lua 函数原型

```
getHandguidDamp\(\) -> table
```

Lua 示例

```
HandguidDamp = getHandguidDamp\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getHandguidDamp","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.5,0.5,0.5,0.5,0.5,0.5]}
```

getHardwareCustomParameters()

```
std::string arcs::common_interface::RobotConfig::getHardwareCustomParameters (  
    const std::string & param)  
获取硬件抽象层自定义参数
```

参数

return 返回硬件抽象层自定义的参数

异常

<code>arcs::common_interface::AuboException</code>	

getHomePosition()

`std::vector< double > arcs::common_interface::RobotConfig::getHomePosition ()`
 获取机器人的原点位置

返回

异常

<code>arcs::common_interface::AuboException</code>	

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getHomePosition","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.0,-0.2617993877991494,1.74532925199433,0.4363323129985824,1.570796326794897,0.0]}
```

getJointMaxAccelerations()

`std::vector< double > arcs::common_interface::RobotConfig::getJointMaxAccelerations ()`
 获取关节最大加速度（物理极限）

返回

返回关节最大加速度

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getJointMaxAccelerations(self: pyaubo_sdk.RobotConfig) -> List[float]
```


Lua 函数原型

```
getJointMaxAccelerations() -> table
```

Lua 示例

```
JointMaxAccelerations = getJointMaxAccelerations()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getJointMaxAccelerations","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[31.104877758314785,31.104877758314785,31.104877758314785,20.73625684294463,20.73625684294463,20.73625684294463]}
```

getJointMaxPositions()

```
std::vector< double > arcs::common_interface::RobotConfig::getJointMaxPositions ()  
获取关节最大位置 (物理极限)
```

返回

返回关节最大位置

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getJointMaxPositions(self: pyaubo_sdk.RobotConfig) -> List[float]
```

Lua 函数原型

```
getJointMaxPositions() -> table
```

Lua 示例

```
JointMaxPositions = getJointMaxPositions()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getJointMaxPositions","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[6.283185307179586,6.283185307179586,6.283185307179586,6.283185307179586,6.283185307179586,6.283185307179586]}
```

getJointMaxSpeeds()

```
std::vector< double > arcs::common_interface::RobotConfig::getJointMaxSpeeds ()  
获取关节最大速度 (物理极限)
```

返回

返回关节最大速度

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getJointMaxSpeeds(self: pyaubo_sdk.RobotConfig) -> List[float]
```

Lua 函数原型

```
getJointMaxSpeeds() -> table
```

Lua 示例

```
JointMaxSpeeds = getJointMaxSpeeds()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getJointMaxSpeeds","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[3.892084231947355,3.892084231947355,3.892084231947355,3.1066860685499065,3.1066860685499065,3.1066860685499065]}
```

getJointMinPositions()

```
std::vector< double > arcs::common_interface::RobotConfig::getJointMinPositions ()
```

获取关节最小位置（物理极限）

返回

返回关节最小位置

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getJointMinPositions(self: pyaubo_sdk.RobotConfig) -> List[float]
```

Lua 函数原型

```
getJointMinPositions() -> table
```

Lua 示例

```
JointMinPositions = getJointMinPositions()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getJointMinPositions","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[-6.283185307179586,-6.283185307179586,-6.283185307179586,-6.283185307179586,-6.283185307179586,-6.283185307179586]}
```

getKinematicsCompensate()

```
std::unordered_map< std::string, std::vector< double > > arcs::common_interface::RobotConfig::getKinematicsCompensate (
    double ref_temperature)
    获取指定温度下的DH 参数补偿值:alpha a d theta beta
```

参数

<i>ref_temperature</i>	参考温度 °C, 默认 20°C

返回

返回DH 参数补偿值

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getKinematicsCompensate(self: pyaubo_sdk.RobotConfig, arg0: float) -> Dict[str, List[float]]
```

Lua 函数原型

```
getKinematicsCompensate(ref_temperature: number) -> table
```

Lua 示例

```
KinematicsCompensate = getKinematicsCompensate(20)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotConfig.getKinematicsCompensate", "params": [20], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": {"a": [0.0, -0.0001255999959539622, 0.0006348000024445355, -0.0002398000069661066, 0.00018950000230688602, 3.7000001611886546e-05], "alpha": [0.0, 0.000678930024150759, 0.002438219962641597, 0.0049148499965667725, 0.00048695001169107854, 0.004321440123021603], "beta": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0], "d": [0.0, 4.769999941345304e-05, -4.769999941345304e-05, 4.769999941345304e-05, 0.0003789000038523227, 0.0], "theta": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}}
```

getKinematicsParam()

```
std::unordered_map< std::string, std::vector< double > > arcs::common_interface::RobotConfig::getKinematicsParam (
    bool real)
    获取机器人DH 参数 alpha a d theta beta
```

参数

<i>real</i>	读取真实参数 (理论值 + 补偿值) 或者理论参数

返回

返回机器人DH 参数

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getKinematicsParam(self: pyaubo_sdk.RobotConfig, arg0: bool) -> Dict[str, List[float]]
```

Lua 函数原型

```
getKinematicsParam(real: boolean) -> table
```

Lua 示例

```
KinematicsParam = getKinematicsParam(true)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotConfig.getKinematicsParam", "params": [true], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": {"a": [0.0, -0.0001255999959539622, 0.4086348000024445, 0.3757601999930339, -0.00018950000230688602, 3.7000001611886546e-05], "alpha": [0.0, -1.5701173967707458, 3.1440308735524347, 3.141592653589793, -1.5703093767832055, 1.5751177669179182], "beta": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0], "d": [0.122, 0.12154769999941345, -4.769999941345304e-05, 4.769999941345304e-05, 0.10287890000385232, 0.116], "theta": [3.141592653589793, -1.5707963267948966, 0.0, -1.5707963267948966, 0.0, 0.0]}}
```

getLimitJointMaxAccelerations()

```
std::vector< double > arcs::common_interface::RobotConfig::getLimitJointMaxAccelerations ()
```

获取关节最大加速度（当前正在使用的限制值）

返回

返回关节最大加速度（当前正在使用的限制值）

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getLimitJointMaxAccelerations(self: pyaubo_sdk.RobotConfig) -> List[float]
```

Lua 函数原型

```
getLimitJointMaxAccelerations() -> table
```

Lua 示例

```
LimitJointMaxAccelerations = getLimitJointMaxAccelerations()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotConfig.getLimitJointMaxAccelerations", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": [31.104877758314785, 31.104877758314785, 31.104877758314785, 20.73625684294463, 20.73625684294463, 20.73625684294463]}
```

getLimitJointMaxPositions()

```
std::vector< double > arcs::common_interface::RobotConfig::getLimitJointMaxPositions ()  
获取关节最大位置（当前正在使用的限制值）
```

返回

返回关节最大位置（当前正在使用的限制值）

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getLimitJointMaxPositions(self: pyaubo_sdk.RobotConfig) -> List[float]
```

Lua 函数原型

```
getLimitJointMaxPositions() -> table
```

Lua 示例

```
LimitJointMaxPositions = getLimitJointMaxPositions()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotConfig.getLimitJointMaxPositions", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": [6.2831854820251465, 6.2831854820251465, 6.2831854820251465, 6.2831854820251465, 6.2831854820251465, 6.2831854820251465]}
```

getLimitJointMaxSpeeds()

```
std::vector< double > arcs::common_interface::RobotConfig::getLimitJointMaxSpeeds ()  
获取关节最大速度（当前正在使用的限制值）
```

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getLimitJointMaxSpeeds(self: pyaubo_sdk.RobotConfig) -> List[float]
```

Lua 函数原型

```
getLimitJointMaxSpeeds() -> table
```

Lua 示例

```
LimitJointMaxSpeeds = getLimitJointMaxSpeeds()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getLimitJointMaxSpeeds","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[3.8920841217041016,3.8920841217041016,3.8920841217041016,3.1066861152648926,1066861152648926,3.1066861152648926]}
```

getLimitJointMinPositions()

```
std::vector< double > arcs::common_interface::RobotConfig::getLimitJointMinPositions ()
```

获取关节最小位置（当前正在使用的限制值）

返回

返回关节最小位置（当前正在使用的限制值）

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getLimitJointMinPositions(self: pyaubo_sdk.RobotConfig) -> List[float]
```

Lua 函数原型

```
getLimitJointMinPositions() -> table
```

Lua 示例

```
LimitJointMinPositions = getLimitJointMinPositions()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getLimitJointMinPositions","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[-6.2831854820251465,-6.2831854820251465,-6.2831854820251465,-6.2831854820251465,-6.2831854820251465,-6.2831854820251465]}
```

getLimitTcpMaxSpeed()

`double arcs::common_interface::RobotConfig::getLimitTcpMaxSpeed ()`
 获取TCP 最大速度（当前正在使用的限制值）

返回

返回TCP 最大速度（当前正在使用的限制值）

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

`getLimitTcpMaxSpeed(self: pyaubo_sdk.RobotConfig) -> List[float]`

Lua 函数原型

`getLimitTcpMaxSpeed\(\) -> number`

Lua 示例

`LimitTcpMaxSpeed = getLimitTcpMaxSpeed\(\)`

JSON-RPC 请求示例

`{"jsonrpc":"2.0","method":"rob1.RobotConfig.getLimitTcpMaxSpeed","params":[],"id":1}`

JSON-RPC 响应示例

`{"id":1,"jsonrpc":"2.0","result":2.0}`

getMountingPose()

`std::vector< double > arcs::common_interface::RobotConfig::getMountingPose ()`
 获取安装位姿（机器人的基坐标系相对于世界坐标系）

返回

安装位姿

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

`getMountingPose(self: pyaubo_sdk.RobotConfig) -> List[float]`

Lua 函数原型

`getMountingPose\(\) -> table`

Lua 示例

```
MountingPose = getMountingPose\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getMountingPose","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.0,0.0,0.0,0.0,0.0,0.0]}
```

getName()

```
std::string arcs::common_interface::RobotConfig::getName ()  
获取机器人的名字
```

返回

返回机器人的名字

异常

arcs::common_interface::AuboException

Python 函数原型

```
getName(self: pyaubo_sdk.RobotConfig) -> str
```

Lua 函数原型

```
getName\(\) -> string
```

Lua 示例

```
robot_name = getName\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getName","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":"rob1"}
```

getPayload()

```
Payload arcs::common_interface::RobotConfig::getPayload ()  
获取有效负载
```

返回

有效负载. 第一个元素表示质量, 单位: kg; 第二个元素表示重心, 单位: m, 形式为 (CoGx, CoGy, CoGz); 第三个元素表示力矩轴的方向, 单位: rad, 形式为 (rx, ry, rz); 第四个元素表示惯量, 单位: kg*m², 形式为 (Ixx, Iyy, Izz, Ixy, Ixz, Iyz)

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getPayload(self: pyaubo_sdk.RobotConfig) -> Tuple[float, List[float], List[float], List[float]]
```

Lua 函数原型

```
getPayload() -> number, table, table, table
```

Lua 示例

```
m, cog, aom, inertia = getPayload()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getPayload","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[3.0,[0.0,0.0,0.0],[0.0,0.0,0.0],[0.0,0.0,0.0,0.0,0.0,0.0]]}
```

getRobotEmergencyStopSource()

```
int arcs::common_interface::RobotConfig::getRobotEmergencyStopSource ()
```

按位获取完整的机器人紧急停止触发源

返回

返回所有机器人紧急停止触发源

紧急停止的原因: 控制柜紧急停止按钮触发 - 1<<0 示教器紧急停止按钮触发 - 1<<1 手柄紧急停止按钮触发 - 1<<2 固定IO 紧急停止触发 - 1<<3

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getRobotEmergencyStopSource","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getRobotSubType()

```
std::string arcs::common_interface::RobotConfig::getRobotSubType ()
```

获取机器人子类型代码

返回

机器人子类型代码

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getRobotSubType(self: pyaubo_sdk.RobotConfig) -> str
```

Lua 函数原型

```
getRobotSubType() -> string
```

Lua 示例

```
RobotSubType = getRobotSubType()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotConfig.getRobotSubType", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": "B0"}
```

getRobotType()

```
std::string arcs::common_interface::RobotConfig::getRobotType ()
```

获取机器人类型代码

返回

机器人类型代码

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getRobotType(self: pyaubo_sdk.RobotConfig) -> str
```

Lua 函数原型

```
getRobotType() -> string
```

Lua 示例

```
RobotType = getRobotType()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotConfig.getRobotType", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": "aubo_i5H"}
```

getSafeguardStopSource()

`int arcs::common_interface::RobotConfig::getSafeguardStopSource ()`
按位获取完整的安全停止触发源

返回

返回所有安全停止触发源

安全停止的原因: 手动模式下可配置安全IO 触发的安全停止 - 1<<0 自动模式下可配置安全IO 触发的安全停止 - 1<<1 控制柜SI 输入触发的安全停止 - 1<<2 示教器三态开关触发的安全停止 - 1<<3 自动切手动触发的安全停止 - 1<<4

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotConfig.getSafeguardStopSource", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

getSafeguardStopType()

`SafeguardStopType arcs::common_interface::RobotConfig::getSafeguardStopType ()`
获取当前安全停止的类型

返回

返回当前安全停止的类型

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotConfig.getSafeguardStopType", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": "None"}
```

getSafetyParametersChecksum()

`uint32_t arcs::common_interface::RobotConfig::getSafetyParametersChecksum ()`
获取安全参数校验码 CRC32

返回

返回安全参数校验码

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getSafetyParametersChecksum(self: pyaubo_sdk.RobotConfig) -> int
```

Lua 函数原型

```
getSafetyParametersChecksum() -> number
```

Lua 示例

```
SafetyParametersChecksum = getSafetyParametersChecksum\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotConfig.getSafetyParametersChecksum", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 2033397241}
```

getSelectedTcpForceSensorName()

```
std::string arcs::common_interface::RobotConfig::getSelectedTcpForceSensorName ()
```

获取工具端力矩传感器的名字

返回

当前工具端力矩传感器名字

异常

arcs::common_interface::AuboException

Python 函数原型

```
getSelectedTcpForceSensorName(self: pyaubo_sdk.RobotConfig) -> str
```

Lua 函数原型

```
getSelectedTcpForceSensorName\(\) -> str
```

Lua 示例

```
TcpForceSensorName = getSelectedTcpForceSensorName\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotConfig.getSelectedTcpForceSensorName", "params": [], "id": 1}
```

getSlowDownFraction()

```
double arcs::common_interface::RobotConfig::getSlowDownFraction (
    int level)
```

获取预设的缓速模式下的速度缩减比例

参数

<i>level</i>	缓速等级 1, 2

返回

返回预设的缓速模式下的速度缩减比例

异常

<code>arcs::common_interface::AuboException</code>

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getSlowDownFraction","params":[1],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0.5}
```

getTcpForceOffset()

```
std::vector< double > arcs::common_interface::RobotConfig::getTcpForceOffset ()
```

获取末端力矩偏移

返回

返回末端力矩偏移

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getTcpForceOffset(self: pyaubo_sdk.RobotConfig) -> List[float]
```

Lua 函数原型

```
getTcpForceOffset() -> table
```

Lua 示例

```
TcpForceOffset = getTcpForceOffset()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getTcpForceOffset","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.0,0.0,0.0,0.0,0.0,0.0]}
```

getTcpForceSensorNames()

```
std::vector< std::string > arcs::common_interface::RobotConfig::getTcpForceSensorNames ()
```

获取可用的末端力矩传感器的名字

返回

返回可用的末端力矩传感器的名字

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getTcpForceSensorNames(self: pyaubo_sdk.RobotConfig) -> List[str]
```

Lua 函数原型

```
getTcpForceSensorNames() -> table
```

Lua 示例

```
TcpForceSensorNames = getTcpForceSensorNames()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getTcpForceSensorNames","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[]}
```

getTcpForceSensorPose()

```
std::vector< double > arcs::common_interface::RobotConfig::getTcpForceSensorPose ()  
获取传感器安装位姿
```

返回

传感器安装位姿

异常

<code>arcs::common_interface::AuboException</code>	

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getTcpForceSensorPose","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.0,0.0,0.0,0.0,0.0,0.0]}
```

getTcpMaxAccelerations()

```
std::vector< double > arcs::common_interface::RobotConfig::getTcpMaxAccelerations ()  
获取TCP 最大加速度（物理极限）
```

返回

返回TCP 最大加速度

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getTcpMaxAccelerations(self: pyaubo_sdk.RobotConfig) -> List[float]
```

Lua 函数原型

```
getTcpMaxAccelerations() -> table
```

Lua 示例

```
TcpMaxAccelerations = getTcpMaxAccelerations()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getTcpMaxAccelerations","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[10.0,10.0]}
```

getTcpMaxSpeeds()

```
std::vector< double > arcs::common_interface::RobotConfig::getTcpMaxSpeeds ()
```

获取TCP 最大速度（物理极限）

返回

返回TCP 最大速度

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getTcpMaxSpeeds(self: pyaubo_sdk.RobotConfig) -> List[float]
```

Lua 函数原型

```
getTcpMaxSpeeds() -> table
```

Lua 示例

```
TcpMaxSpeeds = getTcpMaxSpeeds()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getTcpMaxSpeeds","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[2.0,5.0]}
```

getTcpOffset()

```
std::vector< double > arcs::common_interface::RobotConfig::getTcpOffset ()
```

获取当前的TCP 偏移

TCP 偏移表示形式为 (x,y,z,rx,ry,rz)。其中 x、y、z 是工具中心点 (TCP) 在基坐标系下相对于法兰盘中心的位置偏移, 单位是 m。rx、ry、rz 是工具中心点 (TCP) 在基坐标系下相对于法兰盘中心的姿态偏移, 是ZYX 欧拉角, 单位是 rad。

返回

当前的TCP 偏移, 形式为 (x,y,z,rx,ry,rz)

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getTcpOffset(self: pyaubo_sdk.RobotConfig) -> List[float]
```

Lua 函数原型

```
getTcpOffset() -> table
```

Lua 示例

```
TcpOffset = getTcpOffset()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.getTcpOffset","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.0,0.0,0.0,0.0,0.0,0.0]}
```

hasBaseForceSensor()

```
bool arcs::common_interface::RobotConfig::hasBaseForceSensor ()
```

是否安装了底座力矩传感器

返回

安装返回 true; 没有安装返回 false

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
hasBaseForceSensor(self: pyaubo_sdk.RobotConfig) -> bool
```


Lua 函数原型

`hasBaseForceSensor()` -> boolean

Lua 示例

`hasBaseForceSensor = hasBaseForceSensor()`

JSON-RPC 请求示例

`{"jsonrpc":"2.0","method":"rob1.RobotConfig.hasBaseForceSensor","params":[],"id":1}`

JSON-RPC 响应示例

`{"id":1,"jsonrpc":"2.0","result":false}`

hasTcpForceSensor()

`bool arcs::common_interface::RobotConfig::hasTcpForceSensor ()`
是否安装了末端力矩传感器

返回

安装返回 true; 没有安装返回 false

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

`hasTcpForceSensor(self: pyaubo_sdk.RobotConfig) -> bool`

Lua 函数原型

`hasTcpForceSensor()` -> boolean

Lua 示例

`hasTcpForceSensor = hasTcpForceSensor()`

JSON-RPC 请求示例

`{"jsonrpc":"2.0","method":"rob1.RobotConfig.hasTcpForceSensor","params":[],"id":1}`

JSON-RPC 响应示例

`{"id":1,"jsonrpc":"2.0","result":true}`

isEndCollisionCheckEnabled()

`bool arcs::common_interface::RobotConfig::isEndCollisionCheckEnabled ()`
获取末端碰撞检测是否使能

返回

是否使能末端碰撞检测

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
isEndCollisionCheckEnabled(self: pyaubo_sdk.RobotConfig) -> List[float]
```

Lua 函数原型

```
isEndCollisionCheckEnabled() -> table
```

Lua 示例

```
IsEndCollisionCheckEnabled = isEndCollisionCheckEnabled()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.MotionControl.isEndCollisionCheckEnabled","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":true}
```

selectBaseForceSensor()

```
int arcs::common_interface::RobotConfig::selectBaseForceSensor (
    const std::string & name)
```

设置底座力矩传感器如果存在内置的底座力矩传感器，默认将使用内置的力矩传感器

参数

<code>name</code>	底座力矩传感器的名字

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
selectBaseForceSensor(self: pyaubo_sdk.RobotConfig, arg0: str) -> int
```

Lua 函数原型

```
selectBaseForceSensor(name: string) -> nil
```

selectTcpForceSensor()

```
int arcs::common_interface::RobotConfig::selectTcpForceSensor (
    const std::string & name)
```

设置末端力矩传感器如果存在内置的末端力矩传感器，默认将使用内置的力矩传感器

参数

<i>name</i>	末端力矩传感器的名字

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
selectTcpForceSensor(self: pyaubo_sdk.RobotConfig, arg0: str) -> int
```

Lua 函数原型

```
selectTcpForceSensor(name: string) -> nil
```

setBaseForceOffset()

```
int arcs::common_interface::RobotConfig::setBaseForceOffset (
    const std::vector< double > & force_offset)
设置底座力矩偏移
```

参数

<i>force_offset</i>	底座力矩偏移

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setBaseForceOffset(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
```

Lua 函数原型

```
setBaseForceOffset(force_offset: table) -> nil
```

Lua 示例

```
setBaseForceOffset({0.0,0.0,0.0,0.0,0.0,0.0})
```

setCollisionLevel()

```
int arcs::common_interface::RobotConfig::setCollisionLevel (
    int level)
```

设置碰撞灵敏度等级数值越大越灵敏

参数

<i>level</i>	碰撞灵敏度等级 0: 关闭碰撞检测功能 1~9: 碰撞灵敏等级

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setCollisionLevel(self: pyaubo_sdk.RobotConfig, arg0: int) -> int
```

Lua 函数原型

```
setCollisionLevel(level: number) -> nil
```

Lua 示例

```
setCollisionLevel(6)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.setCollisionLevel","params":[6],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setCollisionStopType()

```
int arcs::common_interface::RobotConfig::setCollisionStopType (
    int type)
```

设置碰撞停止类型

参数

<i>type</i>	类型 0: 碰撞后浮动, 即碰撞之后进入拖动示教模式 1: 碰撞后静止 2: 碰撞后抱闸

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
setCollisionStopType(self: pyaubo_sdk.RobotConfig, arg0: int) -> int
```

Lua 函数原型

```
setCollisionStopType(type: number) -> nil
```

Lua 示例

```
setCollisionStopType(1)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotConfig.setCollisionStopType", "params": [1], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setCollisionThreshold()

```
int arcs::common_interface::RobotConfig::setCollisionThreshold (
    const std::vector< double > & threshold)
设置碰撞阈值
```

参数

<code>threshold</code>	碰撞阈值

返回

成功返回 0; 失败返回错误码

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
setCollisionThreshold(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
```

Lua 函数原型

```
setCollisionThreshold(threshold: table) -> nil
```

Lua 示例

```
setCollisionThreshold({99.0, 114.0, 103.0, 56.0, 51.0, 60.0, 54.6111111, 66.22222222, 59.66666667,
28.94444444, 27.05555556, 30.11111111, 19.1, 28.0, 25.0, 7.3, 7.9, 6.2})
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotConfig.setCollisionThreshold", "params": [[99.0, 114.0, 103.0, 56.0, 51.0, 60.0, 54.6111111, 66.22222222, 59.66666667, 28.94444444, 27.05555556, 30.11111111, 19.1, 28.0, 25.0, 7.3, 7.9, 6.2]], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setFreedriveDamp()

```
int arcs::common_interface::RobotConfig::setFreedriveDamp (
    const std::vector< double > & damp)
设置拖动阻尼
```

参数

<i>damp</i>	阻尼

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setFreedriveDamp(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
```

Lua 函数原型

```
setFreedriveDamp(damp: table) -> nil
```

Lua 示例

```
setFreedriveDamp({0.5,0.5,0.5,0.5,0.5,0.5})
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotConfig.setFreedriveDamp", "params": [[0.5,0.5,0.5,0.5,0.5,0.5]], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setGravity()

```
int arcs::common_interface::RobotConfig::setGravity (
    const std::vector< double > & gravity)
    设置机器人安装姿态
```

参数

<i>gravity</i>	安装姿态

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setGravity(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
```

Lua 函数原型

```
setGravity(gravity: table) -> nil
```

Lua 示例

```
setGravity({0.0,0.0,-9.87654321})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.setGravity","params":[[0.0,0.0,-9.87654321]],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setHandguidDamp()

```
int arcs::common_interface::RobotConfig::setHandguidDamp (
    const std::vector< double > & damp)
    设置混合拖动阻尼
```

参数

<i>damp</i>	阻尼

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
setHandguidDamp(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
```

Lua 函数原型

```
setHandguidDamp(damp: table) -> number
```

Lua 示例

```
setHandguidDamp({0.5,0.5,0.5,0.5,0.5,0.5})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.setHandguidDamp","params":[[0.5,0.5,0.5,0.5,0.5,0.5],{"id":1}]}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setHardwareCustomParameters()

```
int arcs::common_interface::RobotConfig::setHardwareCustomParameters (
    const std::string & param)
```

设置硬件抽象层自定义参数目的是为了做不同硬件之间的兼容

参数

<code>param</code>	自定义参数

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>	

setHomePosition()

```
int arcs::common_interface::RobotConfig::setHomePosition (
    const std::vector< double > & positions)
```

设置机器人的原点位置

参数

<i>positions</i>	关节角度

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.RobotConfig.setHomePosition", "params": [[0.0, -0.2617993877991494, 1.74532925199433, 0.4363323129985824, 1.570796326794897, 0.0]], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

setKinematicsCompensate()

```
int arcs::common_interface::RobotConfig::setKinematicsCompensate (
    const std::unordered_map< std::string, std::vector< double > > & param)
设置标准 DH 补偿到机器人
```

参数

<i>param</i>	

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

setMountingPose()

```
int arcs::common_interface::RobotConfig::setMountingPose (
    const std::vector< double > & pose)
设置安装位姿 (机器人的基坐标系相对于世界坐标系) world->base
一般在多机器人系统中使用, 默认为 [0,0,0,0,0,0]
```

参数

<i>pose</i>	安装位姿

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setMountingPose(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
```

Lua 函数原型

```
setMountingPose(pose: table) -> nil
```

Lua 示例

```
setMountingPose({0.0,0.0,0.0,0.0,0.0,0.0})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.setMountingPose","params":[[0.0,0.0,0.0,0.0,0.0,0.0]↵
],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setPayload()

```
int arcs::common_interface::RobotConfig::setPayload (
    double m,
    const std::vector< double > & cog,
    const std::vector< double > & aom,
    const std::vector< double > & inertia)
```

设置有效负载

参数

<i>m</i>	质量, 单位: kg
<i>cog</i>	重心, 单位: m, 形式为 (CoGx, CoGy, CoGz)
<i>aom</i>	力矩轴的方向, 单位: rad, 形式为 (rx, ry, rz)
<i>inertia</i>	惯量, 单位: kg*m^2, 形式为 (Ixx, Iyy, Izz, Ixy, Ixz, Iyz) 或 (Ixx, Ixy, Ixz, Iyx, Iyy, Iyz, Izx, Izy, Izz)

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setPayload(self: pyaubo_sdk.RobotConfig, arg0: float, arg1: List[float], arg2: List[float], arg3:
List[float]) -> int
```

Lua 函数原型

```
setPayload(m: number, cog: table, aom: table, inertia: table) -> nil
```

Lua 示例

```
setPayload(3, {0,0,0}, {0,0,0}, {0,0,0,0,0,0,0,0,0})
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotConfig.setPayload","params":[3,[0,0,0],[0,0,0],[0,0,0,0,0,0,0,0,0]↵
],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setPersistentParameters()

```
int arcs::common_interface::RobotConfig::setPersistentParameters (
    const std::string & param)
设置需要保存到接口板底座的参数
```

参数

<code>param</code>	补偿数据

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setPersistentParameters(self: pyaubo_sdk.RobotConfig, arg0: str) -> int
```

Lua 函数原型

```
setPersistentParameters(param: string) -> nil
```

setRobotZero()

```
int arcs::common_interface::RobotConfig::setRobotZero ()
```

设置机器人关节零位

返回

成功返回 0；失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setRobotZero(self: pyaubo_sdk.RobotConfig) -> int
```

Lua 函数原型

```
setRobotZero() -> nil
```

Lua 示例

```
setRobotZero()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotConfig.setRobotZero", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setSlowDownFraction()

```
int arcs::common_interface::RobotConfig::setSlowDownFraction (
```

```
    int level,
```

```
    double fraction)
```

预设缓速模式下的速度缩减比例

参数

<i>level</i>	缓速等级 1, 2
<i>fraction</i>	

返回

成功返回 0；失败返回错误码 AUBO_BUSY AUBO_BAD_STATE AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.RobotConfig.setSlowDownFraction", "params": [1, 0.8], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

setTcpForceOffset()

```
int arcs::common_interface::RobotConfig::setTcpForceOffset (
    const std::vector< double > & force_offset)
    设置末端力矩偏移
```

参数

<code>force_offset</code>	末端力矩偏移

返回

成功返回 0; 失败返回错误码

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setTcpForceOffset(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
```

Lua 函数原型

```
setTcpForceOffset(force_offset: table) -> nil
```

Lua 示例

```
setTcpForceOffset({0.0,0.0,0.0,0.0,0.0,0.0,0.0})
```

setTcpForceSensorPose()

```
int arcs::common_interface::RobotConfig::setTcpForceSensorPose (
    const std::vector< double > & sensor_pose)
    设置传感器安装位姿
```

参数

<i>sensor_pose</i>	传感器安装位姿

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

setTcpOffset()

```
int arcs::common_interface::RobotConfig::setTcpOffset (
    const std::vector< double > & offset)
```

设置当前的TCP 偏移

TCP 偏移表示形式为 (x,y,z,rx,ry,rz)。其中 x、y、z 是工具中心点 (TCP) 在基坐标系下相对于法兰盘中心的位置偏移, 单位是 m。rx、ry、rz 是工具中心点 (TCP) 在基坐标系下相对于法兰盘中心的姿态偏移, 是ZYX 欧拉角, 单位是 rad。

参数

<i>offset</i>	当前的TCP 偏移, 形式为 (x,y,z,rx,ry,rz)

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setTcpOffset(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
```

Lua 函数原型

```
setTcpOffset(offset: table) -> nil
```

Lua 示例

```
setTcpOffset({0.0,0.0,0.0,0.0,0.0,0.0})
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.RobotConfig.setTcpOffset", "params": [[0.0,0.0,0.0,0.0,0.0,0.0]], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": 0 }
```

setToolInertial()

```
int arcs::common_interface::RobotConfig::setToolInertial (
    double m,
    const std::vector< double > & com,
    const std::vector< double > & inertial)
```

设置工具端质量、质心及惯量

参数

<i>m</i>	工具端质量
<i>com</i>	质心
<i>inertial</i>	惯量

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
setToolInertial(self: pyaubo_sdk.RobotConfig, arg0: float, arg1: List[float], arg2: List[float]) -> int
```

Lua 函数原型

```
setToolInertial(m: number, com: table, inertial: table) -> nil
```

Lua 示例

```
setToolInertial(3, {0,0,0}, {0,0,0}, {0,0,0,0,0,0,0,0,0})
```

setWorkObjectData()

```
int arcs::common_interface::RobotConfig::setWorkObjectData (
    const WObjectData & wobj)
```

设置工件数据，编程点位都是基于工件坐标系

toolSpaceInRange()

```
bool arcs::common_interface::RobotConfig::toolSpaceInRange (
    const std::vector< double > & pose)
```

末端位姿是否在安全范围之内

参数

<i>pose</i>	末端位姿

返回

在安全范围内返回 true; 反之返回 false

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
toolSpaceInRange(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> bool
```

Lua 函数原型

```
toolSpaceInRange(pose: table) -> boolean
```

Lua 示例

```
SpaceInRange = toolSpaceInRange({0.58712,-0.15775, 0.48703, 2.76, 0.344, 1.432})
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.RobotConfig.toolSpaceInRange", "params": [[0.58712, -0.15775, 0.48703, 2.76, 0.344, 1.432]], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": true }
```

8.6.8 RobotManage (机器人生命周期管理)[**RobotManage**](#)

RobotManage (机器人生命周期管理) 的协作图:

**函数**

- int [*arcs::common_interface::RobotManage::poweron*](#) ()
发起机器人上电请求
- int [*arcs::common_interface::RobotManage::startup*](#) ()
发起机器人启动请求
- int [*arcs::common_interface::RobotManage::releaseRobotBrake*](#) ()

- 发起机器人松开刹车请求
- `int arcs::common_interface::RobotManage::lockRobotBrake ()`
发起机器人刹车请求
- `int arcs::common_interface::RobotManage::poweroff ()`
发起机器人断电请求
- `int arcs::common_interface::RobotManage::backdrive (bool enable)`
发起机器人反向驱动请求
- `int arcs::common_interface::RobotManage::freedrive (bool enable)`
发起机器人自由驱动请求接口在软件版本 0.31.x 后已废弃, 使用 *handguideMode* 接口替换 *handguideMode* (*{1,1,1,1,1}*, *{0,0,0,0,0}*)
- `int arcs::common_interface::RobotManage::setHandguideParams (const std::vector< int > &freeAxes, const std::vector< double > &feature)`
设置拖动示教参数
- `std::vector< int > arcs::common_interface::RobotManage::getHandguideFreeAxes ()`
获取拖动轴
- `std::vector< double > arcs::common_interface::RobotManage::getHandguideFeature ()`
获取拖动参考坐标系
- `int arcs::common_interface::RobotManage::handguideMode (const std::vector< int > &freeAxes, const std::vector< double > &feature)`
高阶拖动示教
- `int arcs::common_interface::RobotManage::exitHandguideMode ()`
退出拖动示教
- `int arcs::common_interface::RobotManage::getHandguideStatus ()`
获取拖动示教器的状态 (是否处于奇异空间)
- `int arcs::common_interface::RobotManage::getHandguideTrigger ()`
获取拖动示教器触发源
- `bool arcs::common_interface::RobotManage::isHandguideEnabled ()`
获取拖动示教使能状态
- `int arcs::common_interface::RobotManage::setSim (bool enable)`
发起机器人进入/退出仿真模式请求
- `int arcs::common_interface::RobotManage::setOperationalMode (OperationalModeType mode)`
设置机器人操作模式
- `OperationalModeType arcs::common_interface::RobotManage::getOperationalMode ()`
获取机器人操作模式
- `RobotControlModeType arcs::common_interface::RobotManage::getRobotControlMode ()`
获取控制模式
- `bool arcs::common_interface::RobotManage::isFreedriveEnabled ()`
是否使能了拖动示教模式
- `bool arcs::common_interface::RobotManage::isBackdriveEnabled ()`
是否使能了反向驱动模式
- `bool arcs::common_interface::RobotManage::isSimulationEnabled ()`
是否使能了仿真模式
- `int arcs::common_interface::RobotManage::setUnlockProtectiveStop ()`
清除防护停机, 包括碰撞停机
- `int arcs::common_interface::RobotManage::restartInterfaceBoard ()`
重置安全接口板, 一般在机器人断电之后需要重置时调用, 比如机器人急停、故障等之后
- `int arcs::common_interface::RobotManage::recordCacheFree (const std::string &name)`
释放并清空指定内存缓存的记录数据
- `int arcs::common_interface::RobotManage::startRecordCache (const std::string &name)`
开始实时轨迹的内存缓存记录 (不落盘)
- `int arcs::common_interface::RobotManage::stopRecordCache ()`

- 停止当前实时轨迹内存缓存记录
- int `arcs::common_interface::RobotManage::pauseRecordCache` (bool pause)
暂停/恢复当前实时轨迹内存缓存记录
- int `arcs::common_interface::RobotManage::getRecordCache` (const std::string &name, size_t frames=0)
获取指定内存缓存的记录数据
- int `arcs::common_interface::RobotManage::startRecord` (const std::string &file_name)
开始实时轨迹的记录
- int `arcs::common_interface::RobotManage::stopRecord` ()
停止实时记录
- int `arcs::common_interface::RobotManage::pauseRecord` (bool pause)
暂停实时记录
- int `arcs::common_interface::RobotManage::setLinkModeEnable` (bool enable)
发起机器人进入/退出联动模式请求, 只有操作模式为自动或者无时, 才能使能联动模式
- bool `arcs::common_interface::RobotManage::isLinkModeEnabled` ()
是否使能了联动模式, 联动模式下用户可以通过外部IO 控制机器人 (用户可以对IO 的功能进行配置)
- int `arcs::common_interface::RobotManage::generateDiagnoseFile` (const std::string &reason)
手动触发生成诊断文件

详细描述

`RobotManage`

函数说明

`backdrive()`

```
int arcs::common_interface::RobotManage::backdrive (
    bool enable)
发起机器人反向驱动请求
```

参数

<code>enable</code>	

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
backdrive(self: pyaubo_sdk.RobotManage, arg0: bool) -> int
```

Lua 函数原型

```
backdrive(enable: boolean) -> number
```

Lua 示例

```
num = backdrive(false)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotManage.backdrive","params":[false],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
rpc_cli->getRobotInterface(robot_name)->getRobotManage()->backdrive(true);
```

exitHandguideMode()

```
int arcs::common_interface::RobotManage::exitHandguideMode ()
```

退出拖动示教

注解

需 0.31.x 及以后软件版本

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>	

freedrive()

```
int arcs::common_interface::RobotManage::freedrive (
    bool enable)
```

发起机器人自由驱动请求接口在软件版本 0.31.x 后已废弃,使用 handguideMode 接口替换 handguide↔ Mode({1,1,1,1,1}, {0,0,0,0,0,0})

参数

<code>enable</code>	

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
freedrive(self: pyaubo_sdk.RobotManage, arg0: bool) -> int
```

Lua 函数原型

```
freedrive(enable: boolean) -> number
```

Lua 示例

```
num = freedrive(false)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotManage.freedrive","params":[true,"id":1]}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
rpc_cli->getRobotInterface(robot_name)->getRobotManage()->freedrive(true);
```

generateDiagnoseFile()

```
int arcs::common_interface::RobotManage::generateDiagnoseFile (
    const std::string & reason)
```

手动触发生成诊断文件

返回

指令下发成功返回 0; 失败返回错误码。

-AUBO_BAD_STATE: 运行时 (RuntimeMachine) 的当前状态不是Stopped, 固件升级请求被拒绝。

AUBO_BAD_STATE 的值是 1。

-AUBO_TIMEOUT: 超时。AUBO_TIMEOUT 的值是 4。

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
generateDiagnoseFile(self: pyaubo_sdk.RobotManage, arg0: str) -> int
```

Lua 函数原型

```
generateDiagnoseFile(reason: string) -> nil
```

Lua 示例

```
generateDiagnoseFile("reason")
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotManage.generateDiagnoseFile","params":["reason"],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":false}
```

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
bool isEnabled =
rpc_cli->getRobotInterface(robot_name)->getRobotManage()->generateDiagnoseFile("reason");
```

getHandguideFeature()

```
std::vector< double > arcs::common_interface::RobotManage::getHandguideFeature ()
获取拖动参考坐标系
```

getHandguideFreeAxes()

```
std::vector< int > arcs::common_interface::RobotManage::getHandguideFreeAxes ()
获取拖动轴
```

getHandguideStatus()

```
int arcs::common_interface::RobotManage::getHandguideStatus ()
获取拖动示教器的状态（是否处于奇异空间）
```

注解

暂未实现

返回

- 0 - 正常操作.
- 1 - 接近奇异空间.
- 2 - 极其接近奇异点, 将产生较大的拖动阻尼.

异常

<code>arcs::common_interface::AuboException</code>

getHandguideTrigger()

```
int arcs::common_interface::RobotManage::getHandguideTrigger ()
获取拖动示教器触发源
```

注解

暂未实现

返回

异常

<code>arcs::common_interface::AuboException</code>

getOperationalMode()

`OperationalModeType` arcs::common_interface::RobotManage::getOperationalMode ()
 获取机器人操作模式

返回

机器人操作模式

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

`getOperationalMode(self: pyaubo_sdk.RobotManage) -> arcs::common_interface::OperationalModeType`

Lua 函数原型

`getOperationalMode() -> number`

Lua 示例

```
num = getOperationalMode()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotManage.getOperationalMode", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": "Manual"}
```

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
OperationalModeType mode =
rpc_cli->getRobotInterface(robot_name)->getRobotManage()->getOperationalMode();
```

getRecordCache()

`int` arcs::common_interface::RobotManage::getRecordCache (
 const std::string & name,
 size_t frames = 0)

获取指定内存缓存的记录数据

参数

<i>name</i>	缓存名称
<i>frames</i>	获取帧数上限： - frames=0: 获取全部已记录帧（1 帧 =1 周期） - frames>0: 最多获取最近 frames 帧

返回

成功返回获取到的帧数 (≥ 0)；失败返回错误码 (< 0)

异常

<i>arcs::common_interface::AuboException</i>

Lua 函数原型

```
getRecordCache(name: string, frames: number = 0) -> number
```

Lua 示例

```
n = getRecordCache("rec") -- 获取全部, 返回帧数
n = getRecordCache("rec", 2000) -- 获取最近 2000 帧, 返回帧数
```

getRobotControlMode()

```
RobotControlModeType arcs::common_interface::RobotManage::getRobotControlMode ()
```

获取控制模式

返回

控制模式

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
getRobotControlMode(self: pyaubo_sdk.RobotManage) -> arcs::common\_interface::RobotControlModeType
```

Lua 函数原型

```
getRobotControlMode\(\) -> number
```

Lua 示例

```
num = getRobotControlMode\(\)
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.RobotManage.getRobotControlMode", "params": [], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": "Position" }
```

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
RobotControlModeType mode =
rpc_cli->getRobotInterface(robot_name)->getRobotManage()->getRobotControlMode();
```

handguideMode()

```
int arcs::common_interface::RobotManage::handguideMode (
    const std::vector< int > & freeAxes,
    const std::vector< double > & feature)
```

高阶拖动示教

参数

<i>freeAxes</i>	可以拖动的轴 0-不能拖动 1-可以拖动
<i>feature</i>	如果维度为 0, 代表基于 TCP 坐标系拖动

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

isBackdriveEnabled()

```
bool arcs::common_interface::RobotManage::isBackdriveEnabled ()
```

是否使能了反向驱动模式

返回

使能返回 true; 反之返回 false

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
isBackdriveEnabled(self: pyaubo_sdk.RobotManage) -> bool
```


Lua 函数原型

`isBackdriveEnabled()` -> boolean

Lua 示例

`BackdriveEnabled = isBackdriveEnabled()`

JSON-RPC 请求示例

`{"jsonrpc":"2.0","method":"rob1.RobotManage.isBackdriveEnabled","params":[],"id":1}`

JSON-RPC 响应示例

`{"id":1,"jsonrpc":"2.0","result":false}`

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
bool isEnabled =
rpc_cli->getRobotInterface(robot_name)->getRobotManage()->isBackdriveEnabled();
```

isFreedriveEnabled()

`bool arcs::common_interface::RobotManage::isFreedriveEnabled ()`

是否使能了拖动示教模式

返回

使能返回 true; 反之返回 false

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

`isFreedriveEnabled(self: pyaubo_sdk.RobotManage) -> bool`

Lua 函数原型

`isFreedriveEnabled()` -> boolean

Lua 示例

`FreedriveEnabled = isFreedriveEnabled()`

JSON-RPC 请求示例

`{"jsonrpc":"2.0","method":"rob1.RobotManage.isFreedriveEnabled","params":[],"id":1}`

JSON-RPC 响应示例

`{"id":1,"jsonrpc":"2.0","result":false}`

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
bool isEnabled =
rpc_cli->getRobotInterface(robot_name)->getRobotManage()->isFreedriveEnabled();
```

isHandguideEnabled()

```
bool arcs::common_interface::RobotManage::isHandguideEnabled ()
```

获取拖动示教使能状态

返回

使能返回 true; 失能返回 false

异常

<code>arcs::common_interface::AuboException</code>

Lua 函数原型

```
isHandguideEnabled() -> boolean
```

Lua 示例

```
Handguide = isHandguideEnabled()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotManage.isHandguideEnabled","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":false}
```

isLinkModeEnabled()

```
bool arcs::common_interface::RobotManage::isLinkModeEnabled ()
```

是否使能了联动模式，联动模式下用户可以通过外部IO 控制机器人（用户可以对IO 的功能进行配置）

返回

使能返回 true; 反之返回 false

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
isLinkModeEnabled(self: pyaubo_sdk.RobotManage) -> bool
```

Lua 函数原型

```
isLinkModeEnabled() -> boolean
```

Lua 示例

```
LinkModeEnabled = isLinkModeEnabled()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotManage.isLinkModeEnabled","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":false}
```

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
bool isEnabled =
rpc_cli->getRobotInterface(robot_name)->getRobotManage()->isLinkModeEnabled();
```

isSimulationEnabled()

```
bool arcs::common_interface::RobotManage::isSimulationEnabled ()
是否使能了仿真模式
```

返回

使能返回 true; 反之返回 false

异常

<i>arcs::common_interface::AuboException</i>

Python 函数原型

```
isSimulationEnabled(self: pyaubo_sdk.RobotManage) -> bool
```

Lua 函数原型

```
isSimulationEnabled() -> boolean
```

Lua 示例

```
SimulationEnabled = isSimulationEnabled()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotManage.isSimulationEnabled","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":true}
```

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
bool isEnabled =
rpc_cli->getRobotInterface(robot_name)->getRobotManage()->isSimulationEnabled();
```

lockRobotBrake()

```
int arcs::common_interface::RobotManage::lockRobotBrake ()
发起机器人刹车请求
```

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
lockRobotBrake(self: pyaubo_sdk.RobotManage) -> int
```

Lua 函数原型

```
lockRobotBrake() -> number
```

Lua 示例

```
num = lockRobotBrake()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotManage.lockRobotBrake","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
rpc_cli->getRobotInterface(robot_name)->getRobotManage()->lockRobotBrake();
```

pauseRecord()

```
int arcs::common_interface::RobotManage::pauseRecord (
    bool pause)
```

暂停实时记录

参数

<code>pause</code>

返回

成功返回 0；失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotManage.pauseRecord","params":[true],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

pauseRecordCache()

```
int arcs::common_interface::RobotManage::pauseRecordCache (
    bool pause)
    暂停/恢复当前实时轨迹内存缓存记录
```

参数

<i>pause</i>	true: 暂停缓存记录; false: 恢复缓存记录

返回

成功返回 0; 失败返回错误码 (<0)

异常

<i>arcs::common_interface::AuboException</i>	

Lua 函数原型

```
pauseRecordCache(pause: boolean) -> nil
```

Lua 示例

```
pauseRecordCache(true) – 暂停 pauseRecordCache(false) – 恢复
```

poweroff()

```
int arcs::common_interface::RobotManage::poweroff ()
    发起机器人断电请求
```

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
poweroff(self: pyaubo_sdk.RobotManage) -> int
```

Lua 函数原型

```
poweroff\(\) -> number
```

Lua 示例

```
num = poweroff\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotManage.poweroff", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
rpc_cli->getRobotInterface(robot_name)->getRobotManage()->poweroff();
```

poweron()

```
int arcs::common_interface::RobotManage::poweron ()
```

发起机器人上电请求

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
poweron(self: pyaubo_sdk.RobotManage) -> int
```

Lua 函数原型

```
poweron() -> number
```

Lua 示例

```
num = poweron()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotManage.poweron", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

C++ 实例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
rpc_cli->getRobotInterface(robot_name)->getRobotManage()->poweron();
```

recordCacheFree()

```
int arcs::common_interface::RobotManage::recordCacheFree (
```

```
    const std::string & name)
```

释放并清空指定内存缓存的记录数据

参数

<i>name</i>	缓存名称

返回

成功返回 0；失败返回错误码 (<0)

异常

<i>arcs::common_interface::AuboException</i>	

Lua 函数原型

```
recordCacheFree(name: string) -> nil
```

Lua 示例

```
recordCacheFree("rec")
```

releaseRobotBrake()

```
int arcs::common_interface::RobotManage::releaseRobotBrake ()
```

发起机器人松开刹车请求

返回

成功返回 0；失败返回错误码 AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
releaseRobotBrake(self: pyaubo_sdk.RobotManage) -> int
```

Lua 函数原型

```
releaseRobotBrake() -> number
```

Lua 示例

```
num = releaseRobotBrake()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotManage.releaseRobotBrake","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
rpc_cli->getRobotInterface(robot_name)->getRobotManage()->releaseRobotBrake();
```

restartInterfaceBoard()

```
int arcs::common_interface::RobotManage::restartInterfaceBoard ()
```

重置安全接口板，一般在机器人断电之后需要重置时调用，比如机器人急停、故障等之后

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
restartInterfaceBoard(self: pyaubo_sdk.RobotManage) -> int
```

Lua 函数原型

```
restartInterfaceBoard() -> number
```

Lua 示例

```
num = restartInterfaceBoard()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotManage.restartInterfaceBoard","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
rpc_cli->getRobotInterface(robot_name)->getRobotManage()->restartInterfaceBoard();
```

setHandguideParams()

```
int arcs::common_interface::RobotManage::setHandguideParams (
```

```
    const std::vector< int > & freeAxes,
```

```
    const std::vector< double > & feature)
```

设置拖动示教参数

参数

<i>freeAxes</i>	可以拖动的轴 0-不能拖动 1-可以拖动
<i>feature</i>	如果维度为 0，代表基于 TCP 坐标系拖动

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

setLinkModeEnable()

```
int arcs::common_interface::RobotManage::setLinkModeEnable (
    bool enable)
```

发起机器人进入/退出联动模式请求, 只有操作模式为自动或者无时, 才能使能联动模式

参数

<code>enable</code>

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_REQUEST_IGNORE -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setLinkModeEnable(self: pyaubo_sdk.RobotManage, arg0: bool) -> int
```

Lua 函数原型

```
setLinkModeEnable(enable: boolean) -> number
```

Lua 示例

```
num = setLinkModeEnable(true)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotManage.setLinkModeEnable","params":[true],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
rpc_cli->getRobotInterface(robot_name)->getRobotManage()->setLinkModeEnable(true);
```

setOperationalMode()

```
int arcs::common_interface::RobotManage::setOperationalMode (
    OperationalModeType mode)
```

设置机器人操作模式

参数

<i>mode</i>	操作模式

返回

成功返回 0; 失败返回错误码 -AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
setOperationalMode(self: pyaubo_sdk.RobotManage, arg0: arcs::common_interface::OperationalModeType)
-> int
```

Lua 函数原型

```
setOperationalMode(mode: number) -> number
```

Lua 示例

```
num = setOperationalMode("Manual")
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotManage.setOperationalMode","params":["Manual"],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
rpc_cli->getRobotInterface(robot_name)->getRobotManage()->setOperationalMode(OperationalModeType::Automatic);
```

setSim()

```
int arcs::common_interface::RobotManage::setSim (
    bool enable)
```

发起机器人进入/退出仿真模式请求

参数

<i>enable</i>	

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setSim(self: pyaubo_sdk.RobotManage, arg0: bool) -> int
```

Lua 函数原型

```
setSim(enable: boolean) -> number
```

Lua 示例

```
num = setSim(true)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotManage.setSim", "params": [true], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
rpc_cli->getRobotInterface(robot_name)->getRobotManage()->setSim(true);
```

setUnlockProtectiveStop()

```
int arcs::common_interface::RobotManage::setUnlockProtectiveStop ()
清除防护停机，包括碰撞停机
```

返回

成功返回 0；失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
setUnlockProtectiveStop(self: pyaubo_sdk.RobotManage) -> int
```

Lua 函数原型

```
setUnlockProtectiveStop() -> number
```

Lua 示例

```
num = setUnlockProtectiveStop()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotManage.setUnlockProtectiveStop", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
rpc_cli->getRobotInterface(robot_name)->getRobotManage()->setUnlockProtectiveStop();
```

startRecord()

```
int arcs::common_interface::RobotManage::startRecord (
    const std::string & file_name)
```

开始实时轨迹的记录

参数

<i>file_name</i>	

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_INVL_ARGUMENT
-AUBO_BAD_STATE

异常

<i>arcs::common_interface::AuboException</i>	

Lua 函数原型

```
startRecord(file_name: string) -> nil
```

Lua 示例

```
startRecord("traje.csv")
```

startRecordCache()

```
int arcs::common_interface::RobotManage::startRecordCache (
    const std::string & name)
```

开始实时轨迹的内存缓存记录（不落盘）

参数

<i>name</i>	缓存名称（用于索引内存中的记录数据）。空字符串将返回参数错误。

返回

成功返回 0; 失败返回错误码（<0）

异常

<code>arcs::common_interface::AuboException</code>

Lua 函数原型

```
startRecordCache(name: string) -> nil
```

Lua 示例

```
startRecordCache("rec")
```

startup()

```
int arcs::common_interface::RobotManage::startup ()
发起机器人启动请求
```

返回

成功返回 0; 失败返回错误码 AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
startup(self: pyaubo_sdk.RobotManage) -> int
```

Lua 函数原型

```
startup() -> number
```

Lua 示例

```
num = startup()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotManage.startup", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

C++ 示例

```
auto rpc_cli = std::make_shared<RpcClient>();
auto robot_name = rpc_cli->getRobotNames().front();
rpc_cli->getRobotInterface(robot_name)->getRobotManage()->startup();
```

stopRecord()

```
int arcs::common_interface::RobotManage::stopRecord ()
停止实时记录
```

返回

成功返回 0; 失败返回错误码 AUBO_BUSY AUBO_BAD_STATE -AUBO_BAD_STATE

异常

<code>arcs::common_interface::AuboException</code>

Lua 函数原型

```
stopRecord() -> nil
```

Lua 示例

```
stopRecord()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotManage.stopRecord","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

stopRecordCache()

```
int arcs::common_interface::RobotManage::stopRecordCache ()
停止当前实时轨迹内存缓存记录
```

返回

成功返回 0；失败返回错误码 (<0)

异常

<code>arcs::common_interface::AuboException</code>

Lua 函数原型

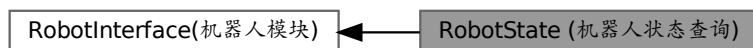
```
stopRecordCache() -> nil
```

Lua 示例

```
stopRecordCache()
```

8.6.9 RobotState (机器人状态查询)**RobotState**

RobotState (机器人状态查询) 的协作图:



函数

- `RobotModeType arcs::common_interface::RobotState::getRobotModeType ()`
获取机器人的模式状态
- `SafetyModeType arcs::common_interface::RobotState::getSafetyModeType ()`
获取安全模式
- `bool arcs::common_interface::RobotState::isPowerOn ()`
获取机器人通电状态
- `bool arcs::common_interface::RobotState::isSteady ()`
机器人是否已经停止下来
- `bool arcs::common_interface::RobotState::isCollisionOccurred ()`
机器人是否发生了碰撞
- `bool arcs::common_interface::RobotState::isWithinSafetyLimits ()`
机器人是否已经在安全限制之内
- `std::vector< double > arcs::common_interface::RobotState::getTcpPose ()`
获取当前的TCP位姿, 其TCP偏移可以通过 `getActualTcpOffset` 获取
- `std::vector< double > arcs::common_interface::RobotState::getActualTcpOffset ()`
获取当前的TCP偏移, 也就是 `getTcpPose` 返回的 `pose` 用到的TCP偏移
- `std::vector< double > arcs::common_interface::RobotState::getTargetTcpPose ()`
获取下一个目标路点注意与 `getTcpTargetPose` 的区别, 此处定义存在歧义, 命名需要优化
- `std::vector< double > arcs::common_interface::RobotState::getToolPose ()`
获取工具端的位姿 (不带TCP偏移)
- `std::vector< double > arcs::common_interface::RobotState::getTcpSpeed ()`
获取TCP速度
- `std::vector< double > arcs::common_interface::RobotState::getTcpForce ()`
获取TCP的力/力矩
- `std::vector< double > arcs::common_interface::RobotState::getElbowPosistion ()`
获取肘部的位置
- `std::vector< double > arcs::common_interface::RobotState::getElbowVelocity ()`
获取肘部速度
- `std::vector< double > arcs::common_interface::RobotState::getBaseForce ()`
获取基座力/力矩
- `std::vector< double > arcs::common_interface::RobotState::getTcpTargetPose ()`
获取上一次发送的TCP目标位姿
- `std::vector< double > arcs::common_interface::RobotState::getTcpTargetSpeed ()`
获取TCP目标速度
- `std::vector< double > arcs::common_interface::RobotState::getTcpTargetForce ()`
获取TCP目标力/力矩
- `std::vector< JointStateType > arcs::common_interface::RobotState::getJointState ()`
获取机械臂关节标志
- `std::vector< JointServoModeType > arcs::common_interface::RobotState::getJointServoMode ()`
获取关节的伺服状态
- `std::vector< double > arcs::common_interface::RobotState::getJointPositions ()`
获取机械臂关节角度
- `std::vector< double > arcs::common_interface::RobotState::getJointPositionsHistory (int steps)`
获取机械臂历史关节角度
- `std::vector< double > arcs::common_interface::RobotState::getJointSpeeds ()`
获取机械臂关节速度
- `std::vector< double > arcs::common_interface::RobotState::getJointAccelerations ()`
获取机械臂关节加速度
- `std::vector< double > arcs::common_interface::RobotState::getJointTorqueSensors ()`

- 获取机械臂关节力矩
- `std::vector< double > arcs::common_interface::RobotState::getJointContactTorques ()`
获取机械臂关节接触力矩 (外力矩)
- `std::vector< double > arcs::common_interface::RobotState::getJointGravityTorques ()`
获取机械臂关节重力矩
- `std::vector< double > arcs::common_interface::RobotState::getTcpForceSensors ()`
获取 *TCP* 力传感器读数
- `std::vector< double > arcs::common_interface::RobotState::getJointCurrents ()`
获取机械臂关节电流
- `std::vector< double > arcs::common_interface::RobotState::getJointVoltages ()`
获取机械臂关节电压
- `std::vector< double > arcs::common_interface::RobotState::getJointTemperatures ()`
获取机械臂关节温度
- `std::vector< std::string > arcs::common_interface::RobotState::getJointUniqueIds ()`
获取关节全球唯一 *ID*
- `std::vector< int > arcs::common_interface::RobotState::getJointFirmwareVersions ()`
获取关节固件版本
- `std::vector< int > arcs::common_interface::RobotState::getJointHardwareVersions ()`
获取关节硬件版本
- `std::string arcs::common_interface::RobotState::getMasterBoardUniqueId ()`
获取 *MasterBoard* 全球唯一 *ID*
- `int arcs::common_interface::RobotState::getMasterBoardFirmwareVersion ()`
获取 *MasterBoard* 固件版本
- `int arcs::common_interface::RobotState::getMasterBoardHardwareVersion ()`
获取 *MasterBoard* 硬件版本
- `std::string arcs::common_interface::RobotState::getSlaveBoardUniqueId ()`
获取 *SlaveBoard* 全球唯一 *ID*
- `int arcs::common_interface::RobotState::getSlaveBoardFirmwareVersion ()`
获取 *SlaveBoard* 固件版本
- `int arcs::common_interface::RobotState::getSlaveBoardHardwareVersion ()`
获取 *SlaveBoard* 硬件版本
- `std::string arcs::common_interface::RobotState::getHandleUniqueId ()`
获取 *Handle* 全球唯一 *ID*
- `int arcs::common_interface::RobotState::getHandleFirmwareVersion ()`
获取 *Handle* 固件版本
- `int arcs::common_interface::RobotState::getHandleHardwareVersion ()`
获取 *Handle* 硬件版本
- `HandleStateType arcs::common_interface::RobotState::getHandleState ()`
获取 *Handle* 状态
- `HandleModeType arcs::common_interface::RobotState::getHandleMode ()`
获取 *Handle* 类型
- `std::string arcs::common_interface::RobotState::getPowerlossBoardUniqueId ()`
获取 *PowerlossBoard* 全球唯一 *ID*
- `int arcs::common_interface::RobotState::getPowerlossBoardFirmwareVersion ()`
获取 *PowerlossBoard* 固件版本
- `int arcs::common_interface::RobotState::getPowerlossBoardHardwareVersion ()`
获取 *PowerlossBoard* 硬件版本
- `std::string arcs::common_interface::RobotState::getToolUniqueId ()`
获取工具端全球唯一 *ID*
- `int arcs::common_interface::RobotState::getToolFirmwareVersion ()`
获取工具端固件版本

- `int arcs::common_interface::RobotState::getToolHardwareVersion ()`
获取工具端硬件版本
- `int arcs::common_interface::RobotState::getToolCommMode ()`
获取末端通信模式
- `std::string arcs::common_interface::RobotState::getPedestalUniqueId ()`
获取底座全球唯一ID
- `int arcs::common_interface::RobotState::getPedestalFirmwareVersion ()`
获取底座固件版本
- `int arcs::common_interface::RobotState::getPedestalHardwareVersion ()`
获取底座硬件版本
- `std::vector< double > arcs::common_interface::RobotState::getJointTargetPositions ()`
获取机械臂关节目标位置角度
- `std::vector< double > arcs::common_interface::RobotState::getJointTargetSpeeds ()`
获取机械臂关节目标速度
- `std::vector< double > arcs::common_interface::RobotState::getJointTargetAccelerations ()`
获取机械臂关节目标加速度
- `std::vector< double > arcs::common_interface::RobotState::getJointTargetTorques ()`
获取机械臂关节目标力矩
- `std::vector< double > arcs::common_interface::RobotState::getJointTargetCurrents ()`
获取机械臂关节目标电流
- `bool arcs::common_interface::RobotState::isTeachPendantEnabled ()`
获取示教器是否已启用指示教器使能按钮是否处于按下状态
- `bool arcs::common_interface::RobotState::isToolFlangeEnabled ()`
获取机械臂末端是否已启用
- `double arcs::common_interface::RobotState::getControlBoxTemperature ()`
获取控制柜温度
- `double arcs::common_interface::RobotState::getControlBoxHumidity ()`
获取控制柜湿度
- `double arcs::common_interface::RobotState::getMainVoltage ()`
获取母线电压
- `double arcs::common_interface::RobotState::getMainCurrent ()`
获取母线电流
- `double arcs::common_interface::RobotState::getRobotVoltage ()`
获取机器人电压
- `double arcs::common_interface::RobotState::getRobotCurrent ()`
获取机器人电流
- `double arcs::common_interface::RobotState::getPedestalBrakeResTemp ()`
获取底座制动电阻温度
- `int arcs::common_interface::RobotState::getSlowDownLevel ()`
获取机器人缓速等级
- `bool arcs::common_interface::RobotState::getTcpForceSensorStatus (const std::string &name)`
获取末端力传感器通信状态

详细描述

RobotState

函数说明

getActualTcpOffset()

`std::vector< double > arcs::common_interface::RobotState::getActualTcpOffset ()`
 获取当前的 TCP 偏移, 也就是 getTcpPose 返回的 pose 用到的 TCP 偏移

返回

当前的 TCP 偏移

Lua 函数原型

`getActualTcpOffset()` -> table

Lua 示例

ActualTcpOffset = `getActualTcpOffset()`

getBaseForce()

`std::vector< double > arcs::common_interface::RobotState::getBaseForce ()`
 获取基座力/力矩

返回

基座力/力矩

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

`getBaseForce(self: pyaubo_sdk.RobotState)` -> List[float]

Lua 函数原型

`getBaseForce()` -> table

Lua 示例

BaseForce = `getBaseForce()`

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotState.getBaseForce", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
```

getControlBoxHumidity()

`double arcs::common_interface::RobotState::getControlBoxHumidity ()`
 获取控制柜湿度

返回

控制柜湿度

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getControlBoxHumidity(self: pyaubo_sdk.RobotState) -> float
```

Lua 函数原型

```
getControlBoxHumidity() -> number
```

Lua 示例

```
ControlBoxHumidity = getControlBoxHumidity()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getControlBoxHumidity","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":20.0}
```

getControlBoxTemperature()

```
double arcs::common_interface::RobotState::getControlBoxTemperature ()
```

获取控制柜温度

返回

控制柜温度

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getControlBoxTemperature(self: pyaubo_sdk.RobotState) -> float
```

Lua 函数原型

```
getControlBoxTemperature() -> number
```

Lua 示例

```
ControlBoxTemperature = getControlBoxTemperature()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getControlBoxTemperature","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":25.0}
```

getElbowPosistion()

```
std::vector< double > arcs::common_interface::RobotState::getElbowPosistion ()
```

获取肘部的位置

返回

肘部的位置

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getElbowPosistion(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getElbowPosistion() -> table
```

Lua 示例

```
ElbowPosistion = getElbowPosistion()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getElbowPosistion","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.07355755887512408,-0.1325,0.43200874126125227,-1.5707963267948968,0.433006344376404,0.0]}
```

getElbowVelocity()

```
std::vector< double > arcs::common_interface::RobotState::getElbowVelocity ()
```

获取肘部速度

返回

肘部速度

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getElbowVelocity(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getElbowVelocity() -> table
```

Lua 示例

```
ElbowVelocity = getElbowVelocity()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getElbowVelocity","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.0,0.0,0.0,0.0,0.0,0.0]}
```

getHandleFirmwareVersion()

```
int arcs::common_interface::RobotState::getHandleFirmwareVersion ()  
获取Handle 固件版本
```

返回

Handle 固件版本

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getHandleFirmwareVersion(self: pyaubo_sdk.RobotState) -> int
```

Lua 函数原型

```
getHandleFirmwareVersion() -> number
```

Lua 示例

```
HandleFirmwareVersion = getHandleFirmwareVersion()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getHandleFirmwareVersion","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getHandleHardwareVersion()

```
int arcs::common_interface::RobotState::getHandleHardwareVersion ()  
获取Handle 硬件版本
```

返回

Handle 硬件版本

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getHandleHardwareVersion(self: pyaubo_sdk.RobotState) -> int
```

Lua 函数原型

```
getHandleHardwareVersion() -> number
```

Lua 示例

```
HandleHardwareVersion = getHandleHardwareVersion()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotState.getHandleHardwareVersion", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 6030098}
```

getHandleMode()

```
HandleModeType arcs::common_interface::RobotState::getHandleMode ()
```

获取Handle 类型

返回

Handle 类型

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getHandleMode(self: pyaubo_sdk.RobotState) -> int
```

Lua 函数原型

```
getHandleMode() -> number
```

Lua 示例

```
HandleMode = getHandleMode()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotState.getHandleMode", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 6030098}
```

getHandleState()

`HandleStateType` `arcs::common_interface::RobotState::getHandleState ()`

获取Handle 状态

返回

Handle 状态

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

`getHandleState(self: pyaubo_sdk.RobotState) -> int`

Lua 函数原型

`getHandleState() -> number`

Lua 示例

`HandleState = getHandleState()`

JSON-RPC 请求示例

`{"jsonrpc":"2.0","method":"rob1.RobotState.getHandleState","params":[],"id":1}`

JSON-RPC 响应示例

`{"id":1,"jsonrpc":"2.0","result":6030098}`

getHandleUniqueId()

`std::string` `arcs::common_interface::RobotState::getHandleUniqueId ()`

获取Handle 全球唯一ID

返回

Handle 全球唯一ID

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

`getHandleUniqueId(self: pyaubo_sdk.RobotState) -> str`

Lua 函数原型

`getHandleUniqueId() -> string`

Lua 示例

```
HandleUniqueId = getHandleUniqueId\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getHandleUniqueId","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":"73657263008000000000000000000000"}
```

getJointAccelerations()

```
std::vector< double > arcs::common_interface::RobotState::getJointAccelerations ()
```

获取机械臂关节加速度

返回

机械臂关节加速度

异常

arcs::common_interface::AuboException

Python 函数原型

```
getJointAccelerations(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getJointAccelerations\(\) -> table
```

Lua 示例

```
JointAccelerations = getJointAccelerations\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getJointAccelerations","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.0,0.0,0.0,0.0,0.0,0.0,0.0]}
```

getJointContactTorques()

```
std::vector< double > arcs::common_interface::RobotState::getJointContactTorques ()
```

获取机械臂关节接触力矩（外力矩）

返回

机械臂关节接触力矩

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getJointContactTorques(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getJointContactTorques() -> table
```

Lua 示例

```
JointContactTorques = getJointContactTorques()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getJointContactTorques","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.0,0.0,0.0,0.0,0.0,0.0]}
```

getJointCurrents()

```
std::vector< double > arcs::common_interface::RobotState::getJointCurrents ()
```

获取机械臂关节电流

返回

机械臂关节电流

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getJointCurrents(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getJointCurrents() -> table
```

Lua 示例

```
JointCurrents = getJointCurrents()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getJointCurrents","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.0,1.25885009765625,-1.5289306640625,0.71868896484375,0.1007080078125,0.3021240234375]}
```

getJointFirmwareVersions()

```
std::vector< int > arcs::common_interface::RobotState::getJointFirmwareVersions ()
```

获取关节固件版本

返回

关节固件版本

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getJointFirmwareVersions(self: pyaubo_sdk.RobotState) -> List[int]
```

Lua 函数原型

```
getJointFirmwareVersions() -> table
```

Lua 示例

```
JointFirmwareVersions = getJointFirmwareVersions()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotState.getJointFirmwareVersions", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": [1000010, 1000010, 1000010, 1000010, 1000010, 1000010]}
```

getJointGravityTorques()

```
std::vector< double > arcs::common_interface::RobotState::getJointGravityTorques ()
```

获取机械臂关节重力矩

返回

机械臂关节重力矩，包含 6 个关节的力矩值（单位：Nm）

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getJointGravityTorques(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getJointGravityTorques() -> table
```

Lua 示例

```
local jointGravityTorques = getJointGravityTorques\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getJointGravityTorques","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[1.23, -2.34, 3.45, -4.56, 0.78, -0.12]}
```

getJointHardwareVersions()

```
std::vector< int > arcs::common_interface::RobotState::getJointHardwareVersions ()
```

获取关节硬件版本

返回

关节硬件版本

异常

arcs::common_interface::AuboException

Python 函数原型

```
getJointHardwareVersions(self: pyaubo_sdk.RobotState) -> List[int]
```

Lua 函数原型

```
getJointHardwareVersions\(\) -> table
```

Lua 示例

```
JointHardwareVersions = getJointHardwareVersions\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getJointHardwareVersions","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[1000000,1000000,1004000,1004000,1004000,1004000]}
```

getJointPositions()

```
std::vector< double > arcs::common_interface::RobotState::getJointPositions ()
```

获取机械臂关节角度

返回

机械臂关节角度

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getJointPositions(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getJointPositions() -> table
```

Lua 示例

```
JointPositions = getJointPositions()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotState.getJointPositions", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": [0.0, -0.26199241371495835, 1.7418102574563423, 0.4330197667082982, 1.5707963267948966, 0.0]}
```

getJointPositionsHistory()

```
std::vector< double > arcs::common_interface::RobotState::getJointPositionsHistory (
    int steps)
```

获取机械臂历史关节角度

参数

<code>steps</code>	步数

返回

机械臂历史关节角度

getJointServoMode()

```
std::vector< JointServoModeType > arcs::common_interface::RobotState::getJointServoMode ()
```

获取关节的伺服状态

返回

关节的伺服状态

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getJointServoMode(self: pyaubo_sdk.RobotState) -> List[arcs::common_interface::JointServoModeType] ←
```

Lua 函数原型

```
getJointServoMode() -> table
```

Lua 示例

```
JointServoMode = getJointServoMode()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getJointServoMode","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":["Position","Position","Position","Position","Position","Position","Position"]} ←
```

getJointSpeeds()

```
std::vector< double > arcs::common_interface::RobotState::getJointSpeeds ()  
获取机械臂关节速度
```

返回

机械臂关节速度

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getJointSpeeds(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getJointSpeeds() -> table
```

Lua 示例

```
JointSpeeds = getJointSpeeds()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getJointSpeeds","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.0,0.0,0.0,0.0,0.0,0.0,0.0]}
```

getJointState()

```
std::vector< JointStateType > arcs::common_interface::RobotState::getJointState ()  
获取机械臂关节标志
```

返回

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getJointState(self: pyaubo_sdk.RobotState) -> List[arcs::common_interface::JointStateType]
```

Lua 函数原型

```
getJointState() -> table
```

Lua 示例

```
JointState = getJointState()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getJointState","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":["Running","Running","Running","Running","Running","Running"]}
```

getJointTargetAccelerations()

```
std::vector< double > arcs::common_interface::RobotState::getJointTargetAccelerations ()
```

获取机械臂关节目标加速度

返回

机械臂关节目标加速度

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getJointTargetAccelerations(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getJointTargetAccelerations() -> table
```

Lua 示例

```
JointTargetAccelerations = getJointTargetAccelerations()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getJointTargetAccelerations","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.0,-0.6737932929246071,-12.610253240108449,0.0,0.0,0.0]}
```

getJointTargetCurrents()

```
std::vector< double > arcs::common_interface::RobotState::getJointTargetCurrents ()
```

获取机械臂关节目标电流

返回

机械臂关节目标电流

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getJointTargetCurrents(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getJointTargetCurrents() -> table
```

Lua 示例

```
JointTargetCurrents = getJointTargetCurrents()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getJointTargetCurrents","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.0,0.0,0.0,0.0,0.0,0.0]}
```

getJointTargetPositions()

```
std::vector< double > arcs::common_interface::RobotState::getJointTargetPositions ()
```

获取机械臂关节目标位置角度

返回

机械臂关节目标位置角度

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getJointTargetPositions(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getJointTargetPositions() -> table
```

Lua 示例

```
JointTargetPositions = getJointTargetPositions\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotState.getJointTargetPositions", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": [0.0, -0.2619944355631239, 1.7418124015308052, 0.4330219266665035, 1.5707963267948966, 0.0]}
```

getJointTargetSpeeds()

```
std::vector< double > arcs::common_interface::RobotState::getJointTargetSpeeds ()
```

获取机械臂关节目标速度

返回

机械臂关节目标速度

异常

arcs::common_interface::AuboException

Python 函数原型

```
getJointTargetSpeeds(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getJointTargetSpeeds\(\) -> table
```

Lua 示例

```
JointTargetSpeeds = getJointTargetSpeeds\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotState.getJointTargetSpeeds", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.00024227101509399773, 0.0016521760307419697, 0.0026521060731088397, 0.0, 0.0]}
```

getJointTargetTorques()

```
std::vector< double > arcs::common_interface::RobotState::getJointTargetTorques ()
```

获取机械臂关节目标力矩

返回

机械臂关节目标力矩

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getJointTargetTorques(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getJointTargetTorques() -> table
```

Lua 示例

```
JointTargetTorques = getJointTargetTorques()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getJointTargetTorques","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.0,0.0,0.0,0.0,0.0,0.0]}
```

getJointTemperatures()

```
std::vector< double > arcs::common_interface::RobotState::getJointTemperatures ()
```

获取机械臂关节温度

返回

机械臂关节温度

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getJointTemperatures(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getJointTemperatures() -> table
```

Lua 示例

```
JointTemperatures = getJointTemperatures()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getJointTemperatures","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[38.0,38.0,38.0,39.0,38.0,39.0]}
```

getJointTorqueSensors()

`std::vector< double > arcs::common_interface::RobotState::getJointTorqueSensors ()`
 获取机械臂关节力矩

返回

机械臂关节力矩

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

`getJointTorqueSensors(self: pyaubo_sdk.RobotState) -> List[float]`

Lua 函数原型

`getJointTorqueSensors() -> table`

Lua 示例

`JointTorqueSensors = getJointTorqueSensors()`

JSON-RPC 请求示例

`{"jsonrpc":"2.0","method":"rob1.RobotState.getJointTorqueSensors","params":[],"id":1}`

JSON-RPC 响应示例

`{"id":1,"jsonrpc":"2.0","result":[0.0,6275.367736816406,-7704.2816162109375,3586.9766235351563,503.0364990234375,1506.0882568359375]}`

getJointUniqueIds()

`std::vector< std::string > arcs::common_interface::RobotState::getJointUniqueIds ()`
 获取关节全球唯一ID

返回

关节全球唯一ID

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

`getJointUniqueIds(self: pyaubo_sdk.RobotState) -> List[str]`

Lua 函数原型

`getJointUniqueIds() -> table`

Lua 示例

```
JointUniqueIds = getJointUniqueIds\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getJointUniqueIds","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":["00800020ffffff31405153","00800020ffffff3e3f5153","00800020ffffff414b5153","00800020ffffff454d5153","00800020ffffff494f5153","00800020ffffff4d5153"]}
```

getJointVoltages()

```
std::vector< double > arcs::common_interface::RobotState::getJointVoltages ()  
获取机械臂关节电压
```

返回

机械臂关节电压

异常

arcs::common_interface::AuboException

Python 函数原型

```
getJointVoltages(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getJointVoltages\(\) -> table
```

Lua 示例

```
JointVoltages = getJointVoltages\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getJointVoltages","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[2.0,2.5,3.0,2.0,2.5,2.0]}
```

getMainCurrent()

```
double arcs::common_interface::RobotState::getMainCurrent ()  
获取母线电流
```

返回

母线电流

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getMainCurrent(self: pyaubo_sdk.RobotState) -> float
```

Lua 函数原型

```
getMainCurrent() -> number
```

Lua 示例

```
MainCurrent = getMainCurrent()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getMainCurrent","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0.3204345703125}
```

getMainVoltage()

```
double arcs::common_interface::RobotState::getMainVoltage ()
```

获取母线电压

返回

母线电压

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getMainVoltage(self: pyaubo_sdk.RobotState) -> float
```

Lua 函数原型

```
getMainVoltage() -> number
```

Lua 示例

```
MainVoltage = getMainVoltage()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getMainVoltage","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":52.75}
```

getMasterBoardFirmwareVersion()

```
int arcs::common_interface::RobotState::getMasterBoardFirmwareVersion ()
```

获取MasterBoard 固件版本

返回

MasterBoard 固件版本

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getMasterBoardFirmwareVersion(self: pyaubo_sdk.RobotState) -> int
```

Lua 函数原型

```
getMasterBoardFirmwareVersion() -> number
```

Lua 示例

```
MasterBoardFirmwareVersion = getMasterBoardFirmwareVersion()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getMasterBoardFirmwareVersion","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":1000004}
```

getMasterBoardHardwareVersion()

```
int arcs::common_interface::RobotState::getMasterBoardHardwareVersion ()
```

获取MasterBoard 硬件版本

返回

MasterBoard 硬件版本

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getMasterBoardHardwareVersion(self: pyaubo_sdk.RobotState) -> int
```

Lua 函数原型

```
getMasterBoardHardwareVersion() -> number
```

Lua 示例

```
MasterBoardHardwareVersion = getMasterBoardHardwareVersion()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotState.getMasterBoardHardwareVersion", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 1000000}
```

getMasterBoardUniqueId()

```
std::string arcs::common_interface::RobotState::getMasterBoardUniqueId ()  
获取MasterBoard 全球唯一ID
```

返回

MasterBoard 全球唯一ID

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getMasterBoardUniqueId(self: pyaubo_sdk.RobotState) -> str
```

Lua 函数原型

```
getMasterBoardUniqueId() -> string
```

Lua 示例

```
MasterBoardUniqueId = getMasterBoardUniqueId()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotState.getMasterBoardUniqueId", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": "001e0044510f343037323637"}
```

getPedestalBrakeResTemp()

```
double arcs::common_interface::RobotState::getPedestalBrakeResTemp ()  
获取底座制动电阻温度
```

返回

底座制动电阻温度

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getPedestalBrakeResTemp(self: pyaubo_sdk.RobotState) -> float
```

Lua 函数原型

```
getPedestalBrakeResTemp() -> number
```

Lua 示例

```
PedestalBrakeResTemp = getPedestalBrakeResTemp()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getPedestalBrakeResTemp","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":23.1111}
```

getPedestalFirmwareVersion()

```
int arcs::common_interface::RobotState::getPedestalFirmwareVersion ()
```

获取底座固件版本

返回

底座固件版本

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getPedestalFirmwareVersion(self: pyaubo_sdk.RobotState) -> int
```

Lua 函数原型

```
getPedestalFirmwareVersion() -> number
```

Lua 示例

```
PedestalFirmwareVersion = getPedestalFirmwareVersion()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getPedestalFirmwareVersion","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":1000004}
```

getPedestalHardwareVersion()

```
int arcs::common_interface::RobotState::getPedestalHardwareVersion ()
```

获取底座硬件版本

返回

底座硬件版本

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getPedestalHardwareVersion(self: pyaubo_sdk.RobotState) -> int
```

Lua 函数原型

```
getPedestalHardwareVersion() -> number
```

Lua 示例

```
PedestalHardwareVersion = getPedestalHardwareVersion()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getPedestalHardwareVersion","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":1007000}
```

getPedestalUniqueId()

```
std::string arcs::common_interface::RobotState::getPedestalUniqueId ()
```

获取底座全球唯一ID

返回

底座全球唯一ID

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getPedestalUniqueId(self: pyaubo_sdk.RobotState) -> str
```

Lua 函数原型

```
getPedestalUniqueId() -> string
```


Lua 示例

```
PedestalUniqueId = getPedestalUniqueId\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getPedestalUniqueId","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":"205257533543065248544339"}
```

getPowerlossBoardFirmwareVersion()

```
int arcs::common_interface::RobotState::getPowerlossBoardFirmwareVersion ()  
获取PowerlossBoard 固件版本
```

返回

PowerlossBoard 固件版本

异常

arcs::common_interface::AuboException

Python 函数原型

```
getPowerlossBoardFirmwareVersion(self: pyaubo_sdk.RobotState) -> int
```

Lua 函数原型

```
getPowerlossBoardFirmwareVersion\(\) -> number
```

Lua 示例

```
PowerlossBoardFirmwareVersion = getPowerlossBoardFirmwareVersion\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getPowerlossBoardFirmwareVersion","params":[],"id":←  
1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getPowerlossBoardHardwareVersion()

```
int arcs::common_interface::RobotState::getPowerlossBoardHardwareVersion ()  
获取PowerlossBoard 硬件版本
```

返回

PowerlossBoard 硬件版本

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getPowerlossBoardHardwareVersion(self: pyaubo_sdk.RobotState) -> int
```

Lua 函数原型

```
getPowerlossBoardHardwareVersion() -> number
```

Lua 示例

```
PowerlossBoardHardwareVersion = getPowerlossBoardHardwareVersion()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getPowerlossBoardHardwareVersion","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":6030098}
```

getPowerlossBoardUniqueId()

```
std::string arcs::common_interface::RobotState::getPowerlossBoardUniqueId ()
获取PowerlossBoard 全球唯一ID
```

返回

```
PowerlossBoard 全球唯一ID
```

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getPowerlossBoardUniqueId(self: pyaubo_sdk.RobotState) -> str
```

Lua 函数原型

```
getPowerlossBoardUniqueId() -> string
```

Lua 示例

```
PowerlossBoardUniqueId = getPowerlossBoardUniqueId()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getPowerlossBoardUniqueId","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":"73657263008000000000000000000000"}
```

getRobotCurrent()

```
double arcs::common_interface::RobotState::getRobotCurrent ()
```

获取机器人电流

返回

机器人电流

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getRobotCurrent(self: pyaubo_sdk.RobotState) -> float
```

Lua 函数原型

```
getRobotCurrent() -> number
```

Lua 示例

```
RobotCurrent = getRobotCurrent()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getRobotCurrent","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0.3204345703125}
```

getRobotModeType()

```
RobotModeType arcs::common_interface::RobotState::getRobotModeType ()
```

获取机器人的模式状态

返回

机器人的模式状态

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getRobotModeType(self: pyaubo_sdk.RobotState) -> arcs::common_interface::RobotModeType
```

Lua 函数原型

```
getRobotModeType() -> number
```

Lua 示例

```
RobotModeType = getRobotModeType()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotState.getRobotModeType", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": "Running"}
```

getRobotVoltage()

```
double arcs::common_interface::RobotState::getRobotVoltage ()  
获取机器人电压
```

返回

机器人电压

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getRobotVoltage(self: pyaubo_sdk.RobotState) -> float
```

Lua 函数原型

```
getRobotVoltage() -> number
```

Lua 示例

```
RobotVoltage = getRobotVoltage()
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotState.getRobotVoltage", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 52.75}
```

getSafetyModeType()

```
SafetyModeType arcs::common_interface::RobotState::getSafetyModeType ()  
获取安全模式
```

返回

安全模式

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getSafetyModeType(self: pyaubo_sdk.RobotState) -> arcs::common_interface::SafetyModeType
```

Lua 函数原型

```
getSafetyModeType() -> number
```

Lua 示例

```
SafetyModeType = getSafetyModeType()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getSafetyModeType","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":"Normal"}
```

getSlaveBoardFirmwareVersion()

```
int arcs::common_interface::RobotState::getSlaveBoardFirmwareVersion ()
```

获取SlaveBoard 固件版本

返回

SlaveBoard 固件版本

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getSlaveBoardFirmwareVersion(self: pyaubo_sdk.RobotState) -> int
```

Lua 函数原型

```
getSlaveBoardFirmwareVersion() -> number
```

Lua 示例

```
SlaveBoardFirmwareVersion = getSlaveBoardFirmwareVersion()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getSlaveBoardFirmwareVersion","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getSlaveBoardHardwareVersion()

```
int arcs::common_interface::RobotState::getSlaveBoardHardwareVersion ()
```

获取SlaveBoard 硬件版本

返回

SlaveBoard 硬件版本

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getSlaveBoardHardwareVersion(self: pyaubo_sdk.RobotState) -> int
```

Lua 函数原型

```
getSlaveBoardHardwareVersion() -> number
```

Lua 示例

```
SlaveBoardHardwareVersion = getSlaveBoardHardwareVersion()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getSlaveBoardHardwareVersion","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":6030098}
```

getSlaveBoardUniqueId()

```
std::string arcs::common_interface::RobotState::getSlaveBoardUniqueId ()
```

获取SlaveBoard 全球唯一ID

返回

SlaveBoard 全球唯一ID

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getSlaveBoardUniqueId(self: pyaubo_sdk.RobotState) -> str
```

Lua 函数原型

```
getSlaveBoardUniqueId() -> string
```

Lua 示例

```
SlaveBoardUniqueId = getSlaveBoardUniqueId()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getSlaveBoardUniqueId","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":"73657263008000000000000000000000"}
```

getSlowDownLevel()

```
int arcs::common_interface::RobotState::getSlowDownLevel ()  
获取机器人缓速等级
```

返回

机器人缓速等级

异常

<code>arcs::common_interface::AuboException</code>

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getSlowDownLevel","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getTargetTcpPose()

```
std::vector< double > arcs::common_interface::RobotState::getTargetTcpPose ()
```

获取下一个目标路点注意与 getTcpTargetPose 的区别，此处定义存在歧义，命名需要优化
位姿表示形式为 (x,y,z,rx,ry,rz)。其中 x、y、z 是工具中心点（TCP）在基坐标系下的目标位置，单位是 m。rx、ry、rz 是工具中心点（TCP）在基坐标系下的目标姿态，是 ZYX 欧拉角，单位是 rad。

返回

当前目标位姿，形式为 (x,y,z,rx,ry,rz)

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getTargetTcpPose(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

`getTargetTcpPose()` -> table

Lua 示例

TargetTcpPose = `getTargetTcpPose()`

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.RobotState.getTargetTcpPose", "params": [], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": [0.4173932217619493, -0.13250000000000012, 0.43296496133045825, 3.141577313781914, 0.0, 1.5707963267948963] }
```

getTcpForce()

`std::vector< double > arcs::common_interface::RobotState::getTcpForce ()`
获取TCP 的力/力矩

返回

TCP 的力/力矩

Python 函数原型

`getTcpForce(self: pyaubo_sdk.RobotState) -> List[float]`

Lua 函数原型

`getTcpForce()` -> table

Lua 示例

TcpForce = `getTcpForce()`

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.RobotState.getTcpForce", "params": [], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0] }
```

getTcpForceSensors()

`std::vector< double > arcs::common_interface::RobotState::getTcpForceSensors ()`
获取TCP 力传感器读数

返回

TCP 力传感器读数

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getTcpForceSensors(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getTcpForceSensors() -> table
```

Lua 示例

```
TcpForceSensors = getTcpForceSensors()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getTcpForceSensors","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.0,0.0,0.0,0.0,0.0,0.0]}
```

getTcpForceSensorStatus()

```
bool arcs::common_interface::RobotState::getTcpForceSensorStatus (
    const std::string & name)
获取末端力传感器通信状态
```

参数

<code>name</code>	力传感器名称, 与 selectTcpForceSensor 的参数一致

返回

通信正常返回 true; 反之返回 false

异常

<code>arcs::common_interface::AuboException</code>

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getTcpForceSensorStatus","params":["tool.KWR75↵
A"],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

getTcpPose()

```
std::vector< double > arcs::common_interface::RobotState::getTcpPose ()
```

获取当前的TCP 位姿，其 TCP 偏移可以通过 getActualTcpOffset 获取

位姿表示形式为 (x,y,z,rx,ry,rz)。其中 x、y、z 是工具中心点（TCP）在基坐标系下的位置，单位是 m。rx、ry、rz 是工具中心点（TCP）在基坐标系下的姿态，是ZYX 欧拉角，单位是 rad。

返回

TCP 的位姿, 形式为 (x,y,z,rx,ry,rz)

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getTcpPose(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getTcpPose() -> table
```

Lua 示例

```
TcpPose = getTcpPose()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getTcpPose","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.41777839846910425,-0.132500000000000012,0.20928451364415995,3.1415792312578987,0.0,1.5707963267948963]}
```

getTcpSpeed()

```
std::vector< double > arcs::common_interface::RobotState::getTcpSpeed ()
```

获取TCP 速度

返回

TCP 速度

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getTcpSpeed(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

`getTcpSpeed()` -> table

Lua 示例

`TcpSpeed = getTcpSpeed()`

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.RobotState.getTcpSpeed", "params": [], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0] }
```

getTcpTargetForce()

`std::vector< double > arcs::common_interface::RobotState::getTcpTargetForce ()`

获取TCP 目标力/力矩

返回

TCP 目标力/力矩

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

`getTcpTargetForce(self: pyaubo_sdk.RobotState) -> List[float]`

Lua 函数原型

`getTcpTargetForce()` -> table

Lua 示例

`TcpTargetForce = getTcpTargetForce()`

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "rob1.RobotState.getTcpTargetForce", "params": [], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0] }
```

getTcpTargetPose()

`std::vector< double > arcs::common_interface::RobotState::getTcpTargetPose ()`

获取上一次发送的TCP 目标位姿

返回

TCP 目标位姿

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getTcpTargetPose(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getTcpTargetPose() -> table
```

Lua 示例

```
TcpTargetPose = getTcpTargetPose()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getTcpTargetPose","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.41777829240862013,-0.13250000000000012,0.2092832117232601,3.1415812372223217,0.0,1.5707963267948963]}
```

getTcpTargetSpeed()

```
std::vector< double > arcs::common_interface::RobotState::getTcpTargetSpeed ()
```

获取TCP 目标速度

返回

TCP 目标速度

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getTcpTargetSpeed(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getTcpTargetSpeed() -> table
```

Lua 示例

```
TcpTargetSpeed = getTcpTargetSpeed()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getTcpTargetSpeed","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.0,0.0,0.0,0.0,0.0,0.0]}
```

getToolCommMode()

```
int arcs::common_interface::RobotState::getToolCommMode ()
```

获取末端通信模式

返回

末端通信模式 0: 表示无串口 1: 表示只有串口 2: 表示带力传感器和串口

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getToolCommMode(self: pyaubo_sdk.RobotState) -> int
```

Lua 函数原型

```
getToolCommMode() -> number
```

Lua 示例

```
ToolCommMode = getToolCommMode()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getToolCommMode","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":1}
```

getToolFirmwareVersion()

```
int arcs::common_interface::RobotState::getToolFirmwareVersion ()
```

获取工具端固件版本

返回

工具端固件版本

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getToolFirmwareVersion(self: pyaubo_sdk.RobotState) -> int
```

Lua 函数原型

```
getToolFirmwareVersion() -> number
```

Lua 示例

```
ToolFirmwareVersion = getToolFirmwareVersion\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotState.getToolFirmwareVersion", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 1001003}
```

getToolHardwareVersion()

```
int arcs::common_interface::RobotState::getToolHardwareVersion ()  
获取工具端硬件版本
```

返回

工具端硬件版本

异常

arcs::common_interface::AuboException

Python 函数原型

```
getToolHardwareVersion(self: pyaubo_sdk.RobotState) -> int
```

Lua 函数原型

```
getToolHardwareVersion\(\) -> number
```

Lua 示例

```
ToolHardwareVersion = getToolHardwareVersion\(\)
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.RobotState.getToolHardwareVersion", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 1000000}
```

getToolPose()

```
std::vector< double > arcs::common_interface::RobotState::getToolPose ()  
获取工具端的位姿（不带TCP 偏移）
```

位姿表示形式为 (x,y,z,rx,ry,rz)。其中 x、y、z 是法兰盘中心在基坐标系下的目标位置，单位是 m。rx、ry、rz 是法兰盘中心在基坐标系下的目标姿态，是ZYX 欧拉角，单位是 rad。

返回

工具端的位姿，形式为 (x,y,z,rx,ry,rz)

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getToolPose(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getToolPose() -> table
```

Lua 示例

```
ToolPose = getToolPose()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getToolPose","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.41777820858878617,-0.132500000000000012,0.20928410288421018,3.141579231257899,0.0,1.5707963267948963]}
```

getToolUniqueId()

```
std::string arcs::common_interface::RobotState::getToolUniqueId ()
```

获取工具端全球唯一ID

返回

工具端全球唯一ID

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
getToolUniqueId(self: pyaubo_sdk.RobotState) -> str
```

Lua 函数原型

```
getToolUniqueId() -> string
```

Lua 示例

```
ToolUniqueId = getToolUniqueId()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getToolUniqueId","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":"397d4e5331541252314d3042"}
```

isCollisionOccurred()

```
bool arcs::common_interface::RobotState::isCollisionOccurred ()  
机器人是否发生了碰撞
```

返回

发生碰撞返回 true; 反之返回 false

异常

<code>arcs::common_interface::AuboException</code>

Lua 函数原型

```
isCollisionOccurred() -> boolean
```

Lua 示例

```
CollisionOccurred = isCollisionOccurred()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.isCollisionOccurred","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":false}
```

isPowerOn()

```
bool arcs::common_interface::RobotState::isPowerOn ()  
获取机器人通电状态
```

返回

机器人通电返回 true; 反之返回 false

异常

<code>arcs::common_interface::AuboException</code>

Lua 函数原型

```
isPowerOn() -> boolean
```

Lua 示例

```
PowerOn = isPowerOn()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.isPowerOn","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":true}
```


isSteady()

`bool arcs::common_interface::RobotState::isSteady ()`
 机器人是否已经停止下来

返回

停止返回 true; 反之返回 false

异常

<code><i>arcs::common_interface::AuboException</i></code>

Python 函数原型

`isSteady(self: pyaubo_sdk.RobotState) -> bool`

Lua 函数原型

`isSteady() -> boolean`

Lua 示例

`Steady = isSteady()`

JSON-RPC 请求示例

`{"jsonrpc":"2.0","method":"rob1.RobotState.isSteady","params":[],"id":1}`

JSON-RPC 响应示例

`{"id":1,"jsonrpc":"2.0","result":true}`

isTeachPendantEnabled()

`bool arcs::common_interface::RobotState::isTeachPendantEnabled ()`
 获取示教器是否已启用指示教器使能按钮是否处于按下状态

返回

按下示教器使能按钮返回 false; 反之返回 true

异常

<code><i>arcs::common_interface::AuboException</i></code>

Python 函数原型

`isTeachPendantEnabled(self: pyaubo_sdk.RobotState) -> bool`

Lua 函数原型

`isTeachPendantEnabled() -> boolean`

Lua 示例

```
TeachPendantEnabled = isTeachPendantEnabled()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.isTeachPendantEnabled","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":true}
```

isToolFlangeEnabled()

```
bool arcs::common_interface::RobotState::isToolFlangeEnabled ()  
获取机械臂末端是否已启用
```

返回

无末端返回 false; 有末端返回 true

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
isToolFlangeEnabled(self: pyaubo_sdk.RobotState) -> bool
```

Lua 函数原型

```
isToolFlangeEnabled() -> boolean
```

Lua 示例

```
toolEnabled = isToolFlangeEnabled()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.isToolFlangeEnabled","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":true}
```

isWithinSafetyLimits()

```
bool arcs::common_interface::RobotState::isWithinSafetyLimits ()  
机器人是否已经在安全限制之内
```

返回

在安全限制之内返回 true; 反之返回 false

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
isWithinSafetyLimits(self: pyaubo_sdk.RobotState) -> bool
```

Lua 函数原型

```
isWithinSafetyLimits() -> boolean
```

Lua 示例

```
WithinSafetyLimits = isWithinSafetyLimits()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.isWithinSafetyLimits","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":true}
```

8.7 RuntimeMachine (运行时管理)

The `RuntimeMachine` class

函数

- int `arcs::common_interface::RuntimeMachine::newTask` (bool daemon=false)
返回 `task_id`
- int `arcs::common_interface::RuntimeMachine::deleteTask` (int tid)
删除 `task`, 会终止正在执行的运动
- int `arcs::common_interface::RuntimeMachine::detachTask` (int tid)
等待 `task` 自然结束
- bool `arcs::common_interface::RuntimeMachine::isTaskAlive` (int tid)
判断任务是否存活
- int `arcs::common_interface::RuntimeMachine::getTaskQueueSize` (int tid)
获取任务中缓存的指令的数量
- int `arcs::common_interface::RuntimeMachine::switchTask` (int tid)
切换当前线程, 切换之后接下来的指令将被插入切换后的线程中
- int `arcs::common_interface::RuntimeMachine::setLabel` (int lineno, const std::string &comment)
标记记下来的指令的行号和注释
- ARCS_DEPRECATED int `arcs::common_interface::RuntimeMachine::setPlanContext` (int tid, int lineno, const std::string &comment)
向 `aubo_control` 日志中添加注释使用 `setLabel` 替换
- int `arcs::common_interface::RuntimeMachine::nop` ()
空操作
- std::tuple< std::string, std::string > `arcs::common_interface::RuntimeMachine::getExecutionStatus` ()
获取耗时的接口 (`INST`) 执行状态, 如 `setPersistentParameters`
- int `arcs::common_interface::RuntimeMachine::gotoLine` (int lineno)

- 跳转到指定行号
- `std::tuple< int, int, std::string > arcs::common_interface::RuntimeMachine::getPlanContext (int tid=-1)`
获取当前运行上下文
- `std::tuple< int, int, std::string > arcs::common_interface::RuntimeMachine::getAdvancePlanContext (int tid=-1)`
获取提前运行规划器的上下文信息
- `int arcs::common_interface::RuntimeMachine::getAdvancePtr (int tid=-1)`
获取 *AdvanceRun* 的程序指针
- `int arcs::common_interface::RuntimeMachine::getMainPtr (int tid=-1)`
获取机器人运动的程序指针
- `int arcs::common_interface::RuntimeMachine::getInterpPtr (int tid)`
获取最近解释过的指令指针
- `int arcs::common_interface::RuntimeMachine::loadProgram (const std::string &program)`
加载本地工程文件 *Lua* 脚本, 只需要给出文件名字, 不需要后缀, 需要从 `${ARCS_WS}/program` 目录中查找
- `int arcs::common_interface::RuntimeMachine::preloadProgram (int index, const std::string &program)`
预加载工程文件
- `std::string arcs::common_interface::RuntimeMachine::getPreloadProgram (int index)`
获取预加载工程文件名字, 如果没有加载或者超出索引范围则返回空字符串
- `int arcs::common_interface::RuntimeMachine::clearPreloadPrograms ()`
清除所有已预加载的工程文件调用此方法将释放所有通过 *preloadProgram* 预加载的工程索引及其关联的工程名称
- `int arcs::common_interface::RuntimeMachine::runProgram ()`
运行已经加载的工程文件
- `int arcs::common_interface::RuntimeMachine::start ()`
开始运行时
- `int arcs::common_interface::RuntimeMachine::stop ()`
停止运行时即脚本运行, 无法停止运行时状态为 *Stopped* 时的机器人运动
- `int arcs::common_interface::RuntimeMachine::abort ()`
终止机器人运行.
- `int arcs::common_interface::RuntimeMachine::pause ()`
暂停解释器
- `int arcs::common_interface::RuntimeMachine::step ()`
单步运行
- `int arcs::common_interface::RuntimeMachine::resume ()`
恢复解释器
- `int arcs::common_interface::RuntimeMachine::arbitraryResume ()`
恢复解释器 (不检查当前点和暂停点距离)
- `int arcs::common_interface::RuntimeMachine::setResumeWait (bool wait)`
恢复解释器之前等待恢复前之前的序列完成
- `int arcs::common_interface::RuntimeMachine::enterCritical (double timeout)`
进入临界区, *abort* 命令会被推迟执行, 避免临界区内的指令被打断
- `int arcs::common_interface::RuntimeMachine::exitCritical ()`
退出临界区
- `ARCS_DEPRECATED RuntimeState arcs::common_interface::RuntimeMachine::getStatus ()`
获取规划器的状态
- `int arcs::common_interface::RuntimeMachine::setBreakPoint (int lineno)`
设置断点
- `int arcs::common_interface::RuntimeMachine::removeBreakPoint (int lineno)`

- 移除断点
- `int arcs::common_interface::RuntimeMachine::clearBreakPoints ()`
清除所有断点
- `int arcs::common_interface::RuntimeMachine::timerStart (const std::string &name)`
定时器开始
- `int arcs::common_interface::RuntimeMachine::timerStop (const std::string &name)`
定时器结束
- `int arcs::common_interface::RuntimeMachine::timerReset (const std::string &name)`
定时器重置
- `int arcs::common_interface::RuntimeMachine::timerDelete (const std::string &name)`
定时器删除
- `double arcs::common_interface::RuntimeMachine::getTimer (const std::string &name)`
获取定时器数值
- `int arcs::common_interface::RuntimeMachine::triggBegin (double distance, double delay)`
开始配置触发
- `int arcs::common_interface::RuntimeMachine::triggEnd ()`
终止配置触发
- `int arcs::common_interface::RuntimeMachine::triggInterrupt (double distance, double delay)`
返回自动分配的中断号
- `std::vector< int > arcs::common_interface::RuntimeMachine::getTriggInterrupts ()`
获取所有的中断号列表

8.7.1 详细描述

The `RuntimeMachine` class

8.7.2 函数说明

abort()

```
int arcs::common_interface::RuntimeMachine::abort ()
```

终止机器人运行.

如果只是考虑停止运行时, 可以调用 `RuntimeMachine::stop` 接口

如果脚本运行时处于 `Running` 状态, 则终止运行时; 如果运行时处于 `Stopped` 且机器人正在移动, 则停止机器人移动; 如果此时力控开启了, 则机器人停止力控

返回

Python 函数原型

```
abort(self: pyaubo_sdk.RuntimeMachine) -> int
```

Lua 函数原型

```
abort() -> number
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RuntimeMachine.abort","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

arbitraryResume()

```
int arcs::common_interface::RuntimeMachine::arbitraryResume ()  
恢复解释器 (不检查当前点和暂停点距离)
```

返回

Python 函数原型

```
arbitraryResume(self: pyaubo_sdk.RuntimeMachine) -> int
```

Lua 函数原型

```
arbitraryResume() -> number
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RuntimeMachine.arbitraryResume","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

clearBreakPoints()

```
int arcs::common_interface::RuntimeMachine::clearBreakPoints ()  
清除所有断点
```

返回

Python 函数原型

```
clearBreakPoints(self: pyaubo_sdk.RuntimeMachine) -> int
```

Lua 函数原型

```
clearBreakPoints() -> number
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RuntimeMachine.clearBreakPoints","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

clearPreloadPrograms()

```
int arcs::common_interface::RuntimeMachine::clearPreloadPrograms ()  
清除所有已预加载的工程文件调用此方法将释放所有通过 preloadProgram 预加载的工程索引及其关  
联的工程名称
```

返回

成功返回 0;

deleteTask()

```
int arcs::common_interface::RuntimeMachine::deleteTask (  
    int tid)
```

删除 task, 会终止正在执行的运动

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.deleteTask", "params": [26], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

detachTask()

```
int arcs::common_interface::RuntimeMachine::detachTask (  
    int tid)
```

等待 task 自然结束

参数

<i>tid</i>	

返回

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.detachTask", "params": [26], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

enterCritical()

```
int arcs::common_interface::RuntimeMachine::enterCritical (  
    double timeout)
```

进入临界区, abort 命令会被推迟执行, 避免临界区内的指令被打断

参数

<i>timeout</i>	单位秒, 范围 0~5 秒, 表示 abort 命令最大推迟时间, 超过这个时间自动退出临界区并 abort

返回

Python 函数原型

```
enterCritical(self: pyaubo_sdk.RuntimeMachine, arg0: double) -> int
```

Lua 函数原型

```
enterCritical(timeout: number) -> number
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.enterCritical", "params": [5.0], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

exitCritical()

```
int arcs::common_interface::RuntimeMachine::exitCritical ()  
退出临界区
```

参数


```
return
```

Python 函数原型

```
exitCritical(self: pyaubo_sdk.RuntimeMachine) -> int
```

Lua 函数原型

```
exitCritical\(\) -> number
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.exitCritical", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

getAdvancePlanContext()

```
std::tuple< int, int, std::string > arcs::common_interface::RuntimeMachine::getAdvancePlanContext (  
    int tid = -1)  
获取提前运行规划器的上下文信息
```

参数

<i>tid</i>	任务编号如果指定 (不是-1)，返回对应任务运行规划器的上下文信息；如果不指定 (是-1)，返回正在运行的线程运行规划器的上下文信息

返回

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "RuntimeMachine.getAdvancePlanContext", "params": [-1], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": [-1, -1, ""] }
```

getAdvancePtr()

```
int arcs::common_interface::RuntimeMachine::getAdvancePtr (
    int tid = -1)
```

获取AdvanceRun 的程序指针

返回

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "RuntimeMachine.getAdvancePtr", "params": [-1], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": -1 }
```

getExecutionStatus()

```
std::tuple< std::string, std::string > arcs::common_interface::RuntimeMachine::getExecutionStatus ()
```

获取耗时的接口 (INST) 执行状态, 如 setPersistentParameters

返回

指令名字, 执行状态执行状态: EXECUTING/FINISHED

Python 函数原型

```
getExecutionStatus(self: pyaubo_sdk.RuntimeMachine) -> Tuple[str, str, int]
```

Lua 函数原型

```
getExecutionStatus() -> string, string, number
```

JSON-RPC 请求示例

```
{ "jsonrpc": "2.0", "method": "RuntimeMachine.getExecutionStatus", "params": [], "id": 1 }
```

JSON-RPC 响应示例

```
{ "id": 1, "jsonrpc": "2.0", "result": ["confirmSafetyParameters", "FINISHED"] }
```

getInterpPtr()

```
int arcs::common_interface::RuntimeMachine::getInterpPtr (
    int tid)
```

获取最近解释过的指令指针

参数

<i>tid</i>	

返回

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.getInterpPtr", "params": [26], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": -1}
```

getMainPtr()

```
int arcs::common_interface::RuntimeMachine::getMainPtr (  
    int tid = -1)  
获取机器人运动的程序指针
```

参数

<i>tid</i>	任务编号如果指定 (不是-1), 返回对应任务的程序指针; 如果不指定 (是-1), 返回正在运行线程的程序指针

返回

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.getMainPtr", "params": [-1], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": -1}
```

getPlanContext()

```
std::tuple< int, int, std::string > arcs::common_interface::RuntimeMachine::getPlanContext (  
    int tid = -1)  
获取当前运行上下文
```

参数

<i>tid</i>	任务编号如果指定 (不是-1), 返回对应任务的运行上下文; 如果不指定 (是-1), 返回正在运行的线程的运行上下文

返回

Python 函数原型

```
getPlanContext(self: pyaubo_sdk.RuntimeMachine) -> Tuple[int, int, str]
```

Lua 函数原型

```
getPlanContext() -> number
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.getPlanContext", "params": [-1], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": [-1, 0, ""]}
```

getPreloadProgram()

```
std::string arcs::common_interface::RuntimeMachine::getPreloadProgram (
    int index)
```

获取预加载工程文件名字，如果没有加载或者超出索引范围则返回空字符串

参数

<i>index</i>	0~99 工程索引号

返回

工程文件名字

getStatus()

```
ARCS_DEPRECATED RuntimeState arcs::common_interface::RuntimeMachine::getStatus ()
```

获取规划器的状态

返回

Python 函数原型

```
getStatus(self: pyaubo_sdk.RuntimeMachine) -> arcs::common_interface::RuntimeState
```

Lua 函数原型

```
getStatus() -> number
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.getStatus", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": "Running"}
```

getTaskQueueSize()

```
int arcs::common_interface::RuntimeMachine::getTaskQueueSize (
```

```
    int tid)
```

获取任务中缓存的指令的数量

参数

<i>tid</i>	

返回

getTimer()

```
double arcs::common_interface::RuntimeMachine::getTimer (
```

```
    const std::string & name)
```

获取定时器数值

参数

<i>name</i>	

返回

Python 函数原型

```
getTimer(self: pyaubo_sdk.RuntimeMachine, arg0: str) -> float
```

Lua 函数原型

```
getTimer(name: string) -> number
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RuntimeMachine.getTimer","params":["timer"],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":25.409769612}
```

getTriggInterrupts()

```
std::vector< int > arcs::common_interface::RuntimeMachine::getTriggInterrupts ()
```

获取所有的中断号列表

返回

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RuntimeMachine.getTriggInterrupts","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[]}
```

gotoLine()

```
int arcs::common_interface::RuntimeMachine::gotoLine (
```

```
    int lineno)
```

跳转到指定行号

参数

<i>lineno</i>	

返回

Python 函数原型

```
gotoLine(self: pyaubo_sdk.RuntimeMachine, arg0: int) -> int
```

Lua 函数原型

```
gotoLine(lineno: number) -> number
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RuntimeMachine.gotoLine","params":[10],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

isTaskAlive()

```
bool arcs::common_interface::RuntimeMachine::isTaskAlive (
```

```
    int tid)
```

判断任务是否存活

参数

<i>tid</i>	

返回

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RuntimeMachine.isTaskAlive","params":[26],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":true}
```

loadProgram()

```
int arcs::common_interface::RuntimeMachine::loadProgram (
```

```
    const std::string & program)
```

加载本地工程文件 Lua 脚本，只需要给出文件名字，不需要后缀，需要从 \${ARCS_WS}/program 目录中查找

参数

<i>program</i>	

返回

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.loadProgram", "params": ["demo"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

newTask()

```
int arcs::common_interface::RuntimeMachine::newTask (  
    bool daemon = false)
```

返回 task_id

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.newTask", "params": [false], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 26}
```

nop()

```
int arcs::common_interface::RuntimeMachine::nop ()
```

空操作

返回

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.nop", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

pause()

```
int arcs::common_interface::RuntimeMachine::pause ()
```

暂停解释器

返回

Python 函数原型

```
pause(self: pyaubo_sdk.RuntimeMachine) -> int
```

Lua 函数原型

```
pause() -> number
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.pause", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

preloadProgram()

```
int arcs::common_interface::RuntimeMachine::preloadProgram (  
    int index,  
    const std::string & program)
```

预加载工程文件

参数

<i>index</i>	0~99 工程索引号
<i>program</i>	工程名字

返回

removeBreakPoint()

```
int arcs::common_interface::RuntimeMachine::removeBreakPoint (  
    int lineno)
```

移除断点

参数

<i>lineno</i>	

返回

Python 函数原型

```
removeBreakPoint(self: pyaubo_sdk.RuntimeMachine, arg0: int) -> int
```

Lua 函数原型

```
removeBreakPoint(lineno: number) -> number
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RuntimeMachine.removeBreakPoint","params":[15],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

resume()

```
int arcs::common_interface::RuntimeMachine::resume ()
恢复解释器
```

返回

Python 函数原型

```
resume(self: pyaubo_sdk.RuntimeMachine) -> int
```

Lua 函数原型

```
resume() -> number
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RuntimeMachine.resume","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

runProgram()

```
int arcs::common_interface::RuntimeMachine::runProgram ()
运行已经加载的工程文件
```

返回

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RuntimeMachine.runProgram","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

setBreakPoint()

```
int arcs::common_interface::RuntimeMachine::setBreakPoint (
    int lineno)
设置断点
```


参数

<i>lineno</i>	

返回

Python 函数原型

```
setBreakPoint(self: pyaubo_sdk.RuntimeMachine, arg0: int) -> int
```

Lua 函数原型

```
setBreakPoint(lineno: number) -> number
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.setBreakPoint", "params": [15], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setLabel()

```
int arcs::common_interface::RuntimeMachine::setLabel (  
    int lineno,  
    const std::string & comment)
```

标记记下来的指令的行号和注释

参数

<i>lineno</i>	
<i>comment</i>	

返回

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.setLabel", "params": [5, "moveJoint"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setPlanContext()

```
ARCS_DEPRECATED int arcs::common_interface::RuntimeMachine::setPlanContext (  
    int tid,  
    int lineno,  
    const std::string & comment)
```

向 aubo_control 日志中添加注释使用 setLabel 替换

参数

<i>tid</i>	指令的线程ID
<i>lineno</i>	行号
<i>comment</i>	注释

返回

Python 函数原型

```
setPlanContext(self: pyaubo_sdk.RuntimeMachine, arg0: int, arg1: int, arg2: str) -> int
```

Lua 函数原型

```
setPlanContext(tid: number, lineno: number, comment: string) -> number
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.setPlanContext", "params": [26, 3, "moveJoint"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

setResumeWait()

```
int arcs::common_interface::RuntimeMachine::setResumeWait (  
    bool wait)
```

恢复解释器之前等待恢复前之前的序列完成

参数

<i>wait</i>	

返回

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.setResumeWait", "params": [true], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

start()

```
int arcs::common_interface::RuntimeMachine::start ()  
    开始运行时
```

返回

Python 函数原型

```
start(self: pyaubo_sdk.RuntimeMachine) -> int
```

Lua 函数原型

```
start() -> number
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.start", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

step()

```
int arcs::common_interface::RuntimeMachine::step ()  
    单步运行
```

返回

Python 函数原型

```
step(self: pyaubo_sdk.RuntimeMachine) -> int
```

Lua 函数原型

```
step() -> number
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.step", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

stop()

```
int arcs::common_interface::RuntimeMachine::stop ()  
    停止运行时即脚本运行，无法停止运行时状态为 Stopped 时的机器人运动  
    如果考虑停止机器人所有运动，可以调用 RuntimeMachine::abort 接口
```

返回

Python 函数原型

```
stop(self: pyaubo_sdk.RuntimeMachine) -> int
```

Lua 函数原型

```
stop() -> number
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RuntimeMachine.stop","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

switchTask()

```
int arcs::common_interface::RuntimeMachine::switchTask (
    int tid)
```

切换当前线程，切换之后接下来的指令将被插入切换后的线程中

参数

<i>tid</i>	

返回

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RuntimeMachine.switchTask","params":[26],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

timerDelete()

```
int arcs::common_interface::RuntimeMachine::timerDelete (
    const std::string & name)
```

定时器删除

参数

<i>name</i>	

返回

Python 函数原型

```
timerDelete(self: pyaubo_sdk.RuntimeMachine, arg0: str) -> int
```

Lua 函数原型

```
timerDelete(name: string) -> nil
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.timerDelete", "params": ["timer"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

timerReset()

```
int arcs::common_interface::RuntimeMachine::timerReset (  
    const std::string & name)  
    定时器重置
```

参数

<i>name</i>	

返回

Python 函数原型

```
timerReset(self: pyaubo_sdk.RuntimeMachine, arg0: str) -> int
```

Lua 函数原型

```
timerReset(name: string) -> nil
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.timerReset", "params": ["timer"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

timerStart()

```
int arcs::common_interface::RuntimeMachine::timerStart (  
    const std::string & name)  
    定时器开始
```

参数

<i>name</i>	

返回

Python 函数原型

```
timerStart(self: pyaubo_sdk.RuntimeMachine, arg0: str) -> int
```

Lua 函数原型

```
timerStart(name: string) -> nil
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.timerStart", "params": ["timer"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

timerStop()

```
int arcs::common_interface::RuntimeMachine::timerStop (  
    const std::string & name)
```

定时器结束

参数

<i>name</i>	

返回

Python 函数原型

```
timerStop(self: pyaubo_sdk.RuntimeMachine, arg0: str) -> int
```

Lua 函数原型

```
timerStop(name: string) -> nil
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.timerStop", "params": ["timer"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

triggBegin()

```
int arcs::common_interface::RuntimeMachine::triggBegin (  
    double distance,
```

参数

<i>distance</i>	
<i>delay</i>	

返回

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.triggBegin", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

triggEnd()

```
int arcs::common_interface::RuntimeMachine::triggEnd ()  
终止配置触发
```

返回

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "RuntimeMachine.triggEnd", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

triggInterrupt()

```
int arcs::common_interface::RuntimeMachine::triggInterrupt (  
    double distance,  
    double delay)  
返回自动分配的中断号
```

参数

<i>distance</i>	
<i>delay</i>	
<i>intnum</i>	

返回

8.8 Serial (串口通信)

串口通信

函数

- `int arcs::common_interface::Serial::serialOpen` (const std::string &device, int baud, float stop_bits, int even, const std::string &serial_name="serial_0")
打开TCP/IP 以太网通信串口
- `int arcs::common_interface::Serial::serialClose` (const std::string &serial_name="serial_0")
关闭TCP/IP 串口通信关闭与服务器的串口连接。
- `int arcs::common_interface::Serial::serialReadByte` (const std::string &variable, const std::string &serial_name="serial_0")
从串口读取指定数量的字节。字节为网络字节序。一次最多可读取 30 个值。
- `int arcs::common_interface::Serial::serialReadByteList` (int number, const std::string &variable, const std::string &serial_name="serial_0")
从串口读取指定数量的字节。字节为网络字节序。一次最多可读取 30 个值。返回读取到的数字列表 (`int` 列表, 长度 = `number+1`)。
- `int arcs::common_interface::Serial::serialReadString` (const std::string &variable, const std::string &serial_name="serial_0", const std::string &prefix="", const std::string &suffix="", bool interpret_escape=false)
从串口读取所有数据, 并将数据作为字符串返回。字节为网络字节序。
- `int arcs::common_interface::Serial::serialSendByte` (char value, const std::string &serial_name="serial_0")
发送一个字节到服务器通过串口发送字节
。不期望有响应。可用于发送特殊的ASCII 字符; 10 为换行符, 2 为文本开始, 3 为文本结束。
- `int arcs::common_interface::Serial::serialSendInt` (int value, const std::string &serial_name="serial_0")
发送一个整数 (`int32_t`) 到服务器通过串口发送整数
。以网络字节序发送。不期望有响应。
- `int arcs::common_interface::Serial::serialSendLine` (const std::string &str, const std::string &serial_name="serial_0")
发送带有换行符的字符串到服务器以ASCII 编码通过串口发送字符串 `<str>`, 并在末尾添加换行符。不期望有响应。
- `int arcs::common_interface::Serial::serialSendString` (const std::string &str, const std::string &serial_name="serial_0")
发送字符串到服务器以ASCII 编码通过串口发送字符串 `<str>`。不期望有响应。
- `int arcs::common_interface::Serial::serialSendAllString` (bool is_check, const std::vector< char > &str, const std::string &serial_name="serial_0")

8.8.1 详细描述

串口通信

8.8.2 函数说明

serialClose()

```
int arcs::common_interface::Serial::serialClose (
    const std::string & serial_name = "serial_0")
```

关闭TCP/IP 串口通信关闭与服务器的串口连接。

参数

<code>serial_name</code>	串口名称

返回

返回值

Python 函数原型

```
serialClose(self: pyaubo_sdk.Serial, arg0: str) -> int
```

Lua 函数原型

```
serialClose(serial_name: string) -> nil
```

serialOpen()

```
int arcs::common_interface::Serial::serialOpen (  
    const std::string & device,  
    int baud,  
    float stop_bits,  
    int even,  
    const std::string & serial_name = "serial_0")
```

打开TCP/IP 以太网通信串口

参数

<i>device</i>	设备名
<i>baud</i>	波特率
<i>stop_bits</i>	停止位
<i>even</i>	校验位
<i>serial_name</i>	串口名称

返回

返回值

Python 函数原型

```
serialOpen(self: pyaubo_sdk.Serial, arg0: str, arg1: int, arg2: float, arg3: int, arg4: str) -> int
```

Lua 函数原型

```
serialOpen(device: string, baud: number, stop_bits: number, even: number, serial_name: string)  
-> nil
```

serialReadByte()

```
int arcs::common_interface::Serial::serialReadByte (  
    const std::string & variable,  
    const std::string & serial_name = "serial_0")
```

从串口读取指定数量的字节。字节为网络字节序。一次最多可读取 30 个值。

参数

<i>variable</i>	变量
<i>serial_name</i>	串口名称

返回

返回值

Python 函数原型

```
serialReadByte(self: pyaubo_sdk.Serial, arg0: str, arg1: str) -> int
```

Lua 函数原型

```
serialReadByte(variable: string, serial_name: string) -> number
```

serialReadByteList()

```
int arcs::common_interface::Serial::serialReadByteList (
    int number,
    const std::string & variable,
    const std::string & serial_name = "serial_0")
```

从串口读取指定数量的字节。字节为网络字节序。一次最多可读取 30 个值。返回读取到的数字列表 (int 列表, 长度 = number+1)。

参数

<i>number</i>	读取的字节数
<i>variable</i>	变量
<i>serial_name</i>	串口名称

返回

返回值

Python 函数原型

```
serialReadByteList(self: pyaubo_sdk.Serial, arg0: int, arg1: str, arg2: str) -> int
```

Lua 函数原型

```
serialReadByteList(number: number, variable: string, serial_name: string) -> number
```

serialReadString()

```
int arcs::common_interface::Serial::serialReadString (
    const std::string & variable,
    const std::string & serial_name = "serial_0",
    const std::string & prefix = "",
    const std::string & suffix = "",
    bool interpret_escape = false)
```

从串口读取所有数据，并将数据作为字符串返回。字节为网络字节序。

可选参数“prefix”和“suffix”用于指定从串口提取的内容。“prefix”指定提取子串（消息）的起始位置。直到“prefix”结束的数据会被忽略并从串口移除。“suffix”指定提取子串（消息）的结束位置。串口中“suffix”之后的剩余数据会被保留。例如，如果串口服务器发送字符串“noise>hello<”，控制器可以通过设置 prefix=”>”和 suffix=”<”来接收“hello”。通过使用“prefix”和“suffix”，还可以一次向控制器发送多条字符串，因为“suffix”定义了消息的结束位置。例如发送“>hello<>world<”

参数

<i>variable</i>	变量
<i>serial_name</i>	串口名称
<i>prefix</i>	前缀
<i>suffix</i>	后缀
<i>interpret_escape</i>	是否解释转义字符

返回

返回值

Python 函数原型

```
serialReadString(self: pyaubo_sdk.Serial, arg0: str, arg1: str, arg2: str, arg3: str, arg4: bool) -> int
```

Lua 函数原型

```
serialReadString(variable: string, serial_name: string, prefix: string, suffix: string, interpret_escape: boolean) -> number
```

serialSendAllString()

```
int arcs::common_interface::Serial::serialSendAllString (
    bool is_check,
    const std::vector< char > & str,
    const std::string & serial_name = "serial_0")
```

参数

<i>is_check</i>	是否校验
<i>str</i>	字符串数组
<i>serial_name</i>	串口名称

返回

返回值

Python 函数原型

```
serialSendAllString(self: pyaubo_sdk.Serial, arg0: bool, arg1: List[str], arg2: str) -> int
```

Lua 函数原型

```
serialSendAllString(is_check: boolean, str: table, serial_name: string) -> nil
```

serialSendByte()

```
int arcs::common_interface::Serial::serialSendByte (
    char value,
    const std::string & serial_name = "serial_0")
```

发送一个字节到服务器通过串口发送字节

。不期望有响应。可用于发送特殊的ASCII 字符；10 为换行符，2 为文本开始，3 为文本结束。

参数

<i>value</i>	字节值
<i>serial_name</i>	串口名称

返回

返回值

Python 函数原型

```
serialSendByte(self: pyaubo_sdk.Serial, arg0: str, arg1: str) -> int
```

Lua 函数原型

```
serialSendByte(value: string, serial_name: string) -> nil
```

serialSendInt()

```
int arcs::common_interface::Serial::serialSendInt (  
    int value,  
    const std::string & serial_name = "serial_0")  
发送一个整数（int32_t）到服务器通过串口发送整数  
。以网络字节序发送。不期望有响应。
```

参数

<i>value</i>	整数值
<i>serial_name</i>	串口名称

返回

返回值

Python 函数原型

```
serialSendInt(self: pyaubo_sdk.Serial, arg0: int, arg1: str) -> int
```

Lua 函数原型

```
serialSendInt(value: number, serial_name: string) -> nil
```

serialSendLine()

```
int arcs::common_interface::Serial::serialSendLine (  
    const std::string & str,  
    const std::string & serial_name = "serial_0")  
发送带有换行符的字符串到服务器以ASCII 编码通过串口发送字符串 <str>，并在末尾添加换行符。  
不期望有响应。
```

参数

<i>str</i>	字符串
<i>serial_name</i>	串口名称

返回

返回值

Python 函数原型

```
serialSendLine(self: pyaubo_sdk.Serial, arg0: str, arg1: str) -> int
```

Lua 函数原型

```
serialSendLine(str: string, serial_name: string) -> nil
```

serialSendString()

```
int arcs::common_interface::Serial::serialSendString (
    const std::string & str,
    const std::string & serial_name = "serial_0")
```

发送字符串到服务器以ASCII 编码通过串口发送字符串 <str>。不期望有响应。

参数

<i>str</i>	字符串
<i>serial_name</i>	串口名称

返回

返回值

Python 函数原型

```
serialSendString(self: pyaubo_sdk.Serial, arg0: str, arg1: str) -> int
```

Lua 函数原型

```
serialSendString(str: string, serial_name: string) -> nil
```

8.9 Socket (socket 网络通信)

串口通信

函数

- int [arcs::common_interface::Socket::socketOpen](#) (const std::string &address, int port, const std::string &socket_name="socket_0")
打开TCP/IP 以太网通信 *socket*
- int [arcs::common_interface::Socket::socketClose](#) (const std::string &socket_name="socket_0")
关闭TCP/IP *socket* 通信关闭与服务器的 *socket* 连接。

- `int arcs::common_interface::Socket::socketReadAsciiFloat` (int number, const std::string &variable, const std::string &socket_name="socket_0")
从 *socket* 读取指定数量的 *ASCII* 格式浮点数。一次最多可读取 30 个值。读取到的数字列表（浮点数列表，长度 = *number*+1）
- `int arcs::common_interface::Socket::socketReadBinaryInteger` (int number, const std::string &variable, const std::string &socket_name="socket_0")
从 *socket* 读取指定数量的 32 位整数。字节为网络字节序。一次最多可读取 30 个值。读取到的数字列表（整数列表，长度 = *number*+1）
- `int arcs::common_interface::Socket::socketReadByteList` (int number, const std::string &variable, const std::string &socket_name="socket_0")
从 *socket* 读取指定数量的字节。字节为网络字节序。一次最多可读取 30 个值。读取到的数字列表（整数列表，长度 = *number*+1）
- `int arcs::common_interface::Socket::socketReadString` (const std::string &variable, const std::string &socket_name="socket_0", const std::string &prefix="", const std::string &suffix="", bool interpret_escape=false)
从 *socket* 读取所有数据并将其作为字符串返回。字节为网络字节序。
- `int arcs::common_interface::Socket::socketReadAllString` (const std::string &variable, const std::string &socket_name="socket_0")
从 *socket* 读取所有数据并将其作为 *char* 向量返回。
- `int arcs::common_interface::Socket::socketSendByte` (char value, const std::string &socket_name="socket_0")
发送一个字节到服务器通过 *socket* 发送字节 <value>，不期望响应。可用于发送特殊 *ASCII* 字符；10 为换行符，2 为文本开始，3 为文本结束。
- `int arcs::common_interface::Socket::socketSendInt` (int value, const std::string &socket_name="socket_0")
发送一个 *int* (*int32_t*) 到服务器通过 *socket* 发送 *int*，以网络字节序发送。不期望响应。
- `int arcs::common_interface::Socket::socketSendLine` (const std::string &str, const std::string &socket_name="socket_0")
发送带有换行符的字符串到服务器。
- `int arcs::common_interface::Socket::socketSendString` (const std::string &str, const std::string &socket_name="socket_0")
发送字符串到服务器通过 *socket* 以 *ASCII* 编码发送字符串 <str>，不期望响应。
- `int arcs::common_interface::Socket::socketSendAllString` (bool is_check, const std::vector< char > &str, const std::string &socket_name="socket_0")
发送给定 *char* 向量中的所有数据到服务器。
- `bool arcs::common_interface::Socket::socketHasConnected` (const std::string &socket_name="socket_0")
检测 *socket* 连接是否成功

8.9.1 详细描述

串口通信

8.9.2 函数说明

socketClose()

```
int arcs::common_interface::Socket::socketClose (
    const std::string & socket_name = "socket_0")
```

关闭TCP/IP socket 通信关闭与服务器的 socket 连接。
指令

参数

<i>socket_name</i>	套接字名称

返回

返回值

Python 函数原型

```
socketClose(self: pyaubo_sdk.Socket, arg0: str) -> int
```

Lua 函数原型

```
socketClose(socket_name: string) -> nil
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "Socket.socketClose", "params": ["socket_0"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

socketHasConnected()

```
bool arcs::common_interface::Socket::socketHasConnected (  
    const std::string & socket_name = "socket_0")  
检测 socket 连接是否成功
```

socketOpen()

```
int arcs::common_interface::Socket::socketOpen (  
    const std::string & address,  
    int port,  
    const std::string & socket_name = "socket_0")  
打开TCP/IP 以太网通信 socket  
指令
```

参数

<i>address</i>	地址
<i>port</i>	端口
<i>socket_name</i>	套接字名称

返回

返回值

Python 函数原型

```
socketOpen(self: pyaubo_sdk.Socket, arg0: str, arg1: int, arg2: str) -> int
```

Lua 函数原型

```
socketOpen(address: string, port: number, socket_name: string) -> nil
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "Socket.socketOpen", "params": ["172.16.26.248", 8000, "socket_0"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

socketReadAllString()

```
int arcs::common_interface::Socket::socketReadAllString (
    const std::string & variable,
    const std::string & socket_name = "socket_0")
```

从 socket 读取所有数据并将其作为 char 向量返回。

指令 std::vector<char>

参数

<i>variable</i>	变量名
<i>socket_name</i>	套接字名称

返回

返回值

Python 函数原型

```
socketReadAllString(self: pyaubo_sdk.Socket, arg0: str, arg1: str) -> int
```

Lua 函数原型

```
socketReadAllString(variable: string, socket_name: string) -> number
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "Socket.socketReadAllString", "params": ["camera", "socket_0"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

socketReadAsciiFloat()

```
int arcs::common_interface::Socket::socketReadAsciiFloat (
    int number,
    const std::string & variable,
    const std::string & socket_name = "socket_0")
```

从 socket 读取指定数量的 ASCII 格式浮点数。一次最多可读取 30 个值。读取到的数字列表（浮点数列表，长度 = number+1）

结果将存储在名为 reg_key 的寄存器中。使用 getFloatVec 获取数据

参数

<i>number</i>	数量
<i>variable</i>	变量名
<i>socket_name</i>	套接字名称

返回

返回值

Python 函数原型

```
socketReadAsciiFloat(self: pyaubo_sdk.Socket, arg0: int, arg1: str, arg2: str) -> int
```

Lua 函数原型

```
socketReadAsciiFloat(number: number, variable: string, socket_name: string) -> number
```

socketReadBinaryInteger()

```
int arcs::common_interface::Socket::socketReadBinaryInteger (
    int number,
    const std::string & variable,
    const std::string & socket_name = "socket_0")
```

从 socket 读取指定数量的 32 位整数。字节为网络字节序。一次最多可读取 30 个值。读取到的数字列表（整数列表，长度 =number+1）

指令

```
std::vector<int>
```

参数

<i>number</i>	数量
<i>variable</i>	变量名
<i>socket_name</i>	套接字名称

返回

返回值

Python 函数原型

```
socketReadBinaryInteger(self: pyaubo_sdk.Socket, arg0: int, arg1: str, arg2: str) -> int
```

Lua 函数原型

```
socketReadBinaryInteger(number: number, variable: string, socket_name: string) -> number
```

socketReadByteList()

```
int arcs::common_interface::Socket::socketReadByteList (
    int number,
    const std::string & variable,
    const std::string & socket_name = "socket_0")
```

从 socket 读取指定数量的字节。字节为网络字节序。一次最多可读取 30 个值。读取到的数字列表（整数列表，长度 =number+1）

指令

```
std::vector<char>
```

参数

<i>number</i>	数量
<i>variable</i>	变量名
<i>socket_name</i>	套接字名称

返回

返回值

Python 函数原型

```
socketReadByteList(self: pyaubo_sdk.Socket, arg0: int, arg1: str, arg2: str) -> int
```

Lua 函数原型

```
socketReadByteList(number: number, variable: string, socket_name: string) -> number
```

socketReadString()

```
int arcs::common_interface::Socket::socketReadString (
    const std::string & variable,
    const std::string & socket_name = "socket_0",
    const std::string & prefix = "",
    const std::string & suffix = "",
    bool interpret_escape = false)
```

从 socket 读取所有数据并将其作为字符串返回。字节为网络字节序。

可选参数“prefix”和“suffix”可用于指定从 socket 中提取的内容。“prefix”指定提取子字符串（消息）的起始位置。直到“prefix”结尾的数据将被忽略并从 socket 中移除。“suffix”指定提取子字符串（消息）的结束位置。“suffix”之后的任何剩余数据将保留在 socket 中。例如，如果 socket 服务器发送字符串“noise>hello<”，控制器可以通过调用此脚本函数并设置 prefix=”>”和 suffix=”<”来接收“hello”。通过使用“prefix”和“suffix”，还可以一次向控制器发送多条字符串，因为“suffix”定义了消息的结束位置。例如发送“>hello<>world<”

指令

```
std::string
```

参数

<i>variable</i>	变量名
<i>socket_name</i>	套接字名称
<i>prefix</i>	前缀
<i>suffix</i>	后缀
<i>interpret_escape</i>	是否解释转义字符

返回

返回值

Python 函数原型

```
socketReadString(self: pyaubo_sdk.Socket, arg0: str, arg1: str, arg2: str, arg3: str, arg4: bool) -> int
```

Lua 函数原型

```
socketReadString(variable: string, socket_name: string, prefix: string, suffix: string, interpret_↵
escape: boolean) -> number
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"Socket.socketReadString","params":["camera","socket_0","","",false]↵
,"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

socketSendAllString()

```
int arcs::common_interface::Socket::socketSendAllString (
    bool is_check,
    const std::vector< char > & str,
    const std::string & socket_name = "socket_0")
```

发送给定 char 向量中的所有数据到服务器。

参数

<i>is_check</i>	是否检查发送状态
<i>str</i>	要发送的数据, char 向量
<i>socket_name</i>	套接字名称

返回

状态码

Python 函数原型

```
socketSendAllString(self: pyaubo_sdk.Socket, arg0: bool, arg1: List[str], arg2: str) -> int
```

Lua 函数原型

```
socketSendAllString(is_check: boolean, str: table, socket_name: string) -> nil
```

socketSendByte()

```
int arcs::common_interface::Socket::socketSendByte (
    char value,
    const std::string & socket_name = "socket_0")
```

发送一个字节到服务器通过 socket 发送字节 <value>, 不期望响应。可用于发送特殊ASCII 字符; 10 为换行符, 2 为文本开始, 3 为文本结束。

指令

参数

<i>value</i>	字节值
<i>socket_name</i>	套接字名称

返回

返回值

Python 函数原型

```
socketSendByte(self: pyaubo_sdk.Socket, arg0: str, arg1: str) -> int
```

Lua 函数原型

```
socketSendByte(value: string, socket_name: string) -> nil
```

socketSendInt()

```
int arcs::common_interface::Socket::socketSendInt (  
    int value,  
    const std::string & socket_name = "socket_0")  
发送一个 int (int32_t) 到服务器通过 socket 发送 int  
, 以网络字节序发送。不期望响应。  
指令
```

参数

<i>value</i>	整数值
<i>socket_name</i>	套接字名称

返回

返回值

Python 函数原型

```
socketSendInt(self: pyaubo_sdk.Socket, arg0: int, arg1: str) -> int
```

Lua 函数原型

```
socketSendInt(value: number, socket_name: string) -> nil
```

socketSendLine()

```
int arcs::common_interface::Socket::socketSendLine (  
    const std::string & str,  
    const std::string & socket_name = "socket_0")  
发送带有换行符的字符串到服务器。  
通过 socket 以 ASCII 编码发送字符串 <str>, 不期望响应。  
指令
```

参数

<i>str</i>	字符串
<i>socket_name</i>	套接字名称

返回

返回值

Python 函数原型

```
socketSendLine(self: pyaubo_sdk.Socket, arg0: str, arg1: str) -> int
```

Lua 函数原型

```
socketSendLine(str: string, socket_name: string) -> nil
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "Socket.socketSendLine", "params": ["abcd", "socket_0"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

socketSendString()

```
int arcs::common_interface::Socket::socketSendString (
    const std::string & str,
    const std::string & socket_name = "socket_0")
```

发送字符串到服务器通过 socket 以ASCII 编码发送字符串 <str>, 不期望响应。
指令

参数

<i>str</i>	字符串
<i>socket_name</i>	套接字名称

返回

返回值

Python 函数原型

```
socketSendString(self: pyaubo_sdk.Socket, arg0: str, arg1: str) -> int
```

Lua 函数原型

```
socketSendString(str: string, socket_name: string) -> nil
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "Socket.socketSendString", "params": ["abcd", "socket_0"], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": 0}
```

8.10 SyncMove (同步运动控制和轴组管理)

同步运动控制和轴组管理API 接口

函数

- int `arcs::common_interface::SyncMove::syncMoveOn` (const std::string &syncident, const TaskSet &taskset)
syncMoveOn 用于启动同步运动模式。
- bool `arcs::common_interface::SyncMove::syncMoveSegment` (int id)
 设置同步路径段的ID 在同步运动模式下, 所有同时执行的移动指令必须全部编程为圆角区 (*corner zones*) 或全部为停止点 (*stop points*)。这意味着具有相同ID 的移动指令要么全部带有圆角区, 要么全部带有停止点。如果在各自的程序任务中同步执行的移动指令中, 一个带有圆角区而另一个带有停止点, 则会发生错误。同步执行的移动指令可以有不同大小的圆角区 (例如, 一个使用 *z10*, 另一个使用 *z50*)。
- int `arcs::common_interface::SyncMove::syncMoveOff` (const std::string &syncident)
syncMoveOff 用于结束同步运动模式。
- int `arcs::common_interface::SyncMove::syncMoveUndo` ()
syncMoveUndo 用于关闭同步运动, 即使不是所有其他任务程序都执行了 *syncMoveUndo* 指令。
- int `arcs::common_interface::SyncMove::waitSyncTasks` (const std::string &syncident, const TaskSet &taskset)
waitSyncTasks 用于在程序中的特定点同步多个任务程序。
- bool `arcs::common_interface::SyncMove::isSyncMoveOn` ()
isSyncMoveOn 用于判断机械单元组是否处于同步运动模式。
- int `arcs::common_interface::SyncMove::syncMoveSuspend` ()
 暂停同步运动模式。
- int `arcs::common_interface::SyncMove::syncMoveResume` ()
 恢复同步运动模式。
- int `arcs::common_interface::SyncMove::frameAdd` (const std::string &name, const std::vector< double > &pose, const std::string &ref_name)
 添加一个名为 *name* 的坐标系, 其初始位姿为 *pose*, 位姿以 *ref_frame* 坐标系表达。此命令仅向世界模型添加一个坐标系, 并不会将其附加到 *ref_frame* 坐标系。如需将新添加的坐标系附加到 *ref_frame*, 请使用 *frameAttach()*。
- int `arcs::common_interface::SyncMove::frameAttach` (const std::string &child, const std::string &parent)
 将子坐标系附加到父世界模型对象。附加时会设置父子之间的相对变换, 使得子坐标系在世界中不会移动。
- int `arcs::common_interface::SyncMove::frameDeleteAll` ()
 删除所有已添加到世界模型的坐标系。
- int `arcs::common_interface::SyncMove::frameDelete` (const std::string &name)
 删除指定名称的坐标系。
- int `arcs::common_interface::SyncMove::frameMove` (const std::string &name, const std::vector< double > &pose, const std::string &ref_name)
 更改名为 *name* 的坐标系的位置, 将其移动到由 *pose* 指定的新位置, *pose* 以 *ref_name* 坐标系表达。
- std::vector< double > `arcs::common_interface::SyncMove::frameGetPose` (const std::string &name, const std::string &rel_frame, const std::string &ref_frame)
 获取名为 *name* 的坐标系相对于 *rel_frame* 坐标系的位姿, 并以 *ref_frame* 坐标系表达。如果未提供 *ref_frame*, 则返回 *name* 坐标系相对于 *rel_frame* 坐标系的位姿, 并以 *rel_frame* 坐标系表达。
- std::vector< double > `arcs::common_interface::SyncMove::frameConvertPose` (const std::vector< double > &pose, const std::string &from_frame, const std::string &to_frame)
 将位姿从 *from_frame* 坐标系转换到 *to_frame* 坐标系。
- bool `arcs::common_interface::SyncMove::frameExist` (const std::string &name)
 查询指定名称的坐标系是否存在。
- std::string `arcs::common_interface::SyncMove::frameGetParent` (const std::string &name)
 获取名为 *name* 的坐标系在世界模型中的父坐标系名称。
- std::vector< std::string > `arcs::common_interface::SyncMove::frameGetChildren` (const std::string &name)
 返回指定父对象的直接子对象坐标系名称列表。父子关系由世界模型的附加关系定义。如果用于 *Motion↔Plus*, 子对象也可以是轴组或轴。

- `int arcs::common_interface::SyncMove::axisGroupAdd` (`const std::string &name`, `const std::vector< double > &pose`, `const std::string &ref_frame`)
向世界模型添加一个新的轴组, 名称为 *name*。轴组基座放置在 *ref_frame* 坐标系下的 *pose* 位置。
- `int arcs::common_interface::SyncMove::axisGroupDelete` (`const std::string &name`)
删除具有给定名称的轴组。
- `int arcs::common_interface::SyncMove::axisGroupAddAxis` (`const std::string &group_name`, `const std::string &name`, `const std::string &parent`, `const std::vector< double > &pose`)
向名为 *group_name* 的轴组添加一个名为 *name* 的外部轴。该轴在 *parent* 坐标系下的 *pose* 位置附加, *pose* 表示轴位置为 0 时的位姿。轴的类型、最大速度、最大加速度、位置限制和索引分别由 *type*、*v_limit*、*a_limit*、*q_limits* 和 *axis_index* 定义。*pose* 参数通常通过外部轴调试标定流程获得。如果该轴组正被其他函数控制, 或附加关系形成闭环, 则操作会失败。
- `int arcs::common_interface::SyncMove::axisGroupUpdateAxis` (`const std::string &name`, `const std::vector< double > &pose`)
更新指定名称的轴的相关属性。*pose* 参数通常通过外部轴调试标定流程获得。如果该轴所属的轴组正被其他命令控制, 则操作会失败。如果该轴组中任何已附加的轴处于激活和使能状态, 则操作会失败。
- `int arcs::common_interface::SyncMove::axisGroupGetAxisIndex` (`const std::string &name`)
返回指定轴名称在 *RTDE* 目标位置 and 实际位置数组中的索引。
- `std::string arcs::common_interface::SyncMove::axisGroupGetAxisName` (`int index`)
返回指定轴索引对应的轴名称。
- `std::vector< double > arcs::common_interface::SyncMove::axisGroupGetTargetPositions` (`const std::string &group_name`)
返回指定轴组的当前目标位置。如果未指定 *group_name*, 则返回所有外部轴的目标位置。
- `std::vector< double > arcs::common_interface::SyncMove::axisGroupGetActualPositions` (`const std::string &group_name`)
返回指定轴组的当前实际位置。如果未指定 *group_name*, 则返回所有外部轴的实际位置。
- `int arcs::common_interface::SyncMove::axisGroupOffsetPositions` (`const std::string &group_name`, `const std::vector< double > &offset`)
通过给定的 *offset*, 将轴组 *group_name* 的目标位置 and 实际位置整体偏移。
- `int arcs::common_interface::SyncMove::axisGroupMoveJoint` (`const std::string &group_name`, `const std::vector< double > &q`, `double a`, `double v`)
以梯形速度曲线将名为 *group_name* 的轴组移动到新的位置 *q*。参数 *a* 指定本次运动的最大加速度占各轴加速度极限的百分比。参数 *v* 指定本次运动的最大速度占各轴速度极限的百分比。
- `int arcs::common_interface::SyncMove::axisGroupSpeedJoint` (`const std::string &group_name`, `const std::vector< double > &qd`, `double a`, `double t`)
以指定的加速度因子 *a*, 将名为 *group_name* 的轴组加速到目标速度 *qd*。该函数会运行 *t* 秒。

8.10.1 详细描述

同步运动控制和轴组管理API 接口

8.10.2 函数说明

axisGroupAdd()

```
int arcs::common_interface::SyncMove::axisGroupAdd (
    const std::string & name,
    const std::vector< double > & pose,
    const std::string & ref_frame)
```

向世界模型添加一个新的轴组, 名称为 *name*。轴组基座放置在 *ref_frame* 坐标系下的 *pose* 位置。轴组只能附加到世界坐标系。

每个轴组的基座都附加有一个坐标系, 可以通过轴组名称作为参数传递给其他世界模型函数。世界模型最多可添加 6 个轴组。

参数

<i>name</i>	要添加的轴组名称，不能为空字符串。世界模型对象（如坐标系、轴组、轴等）的名称必须唯一。
<i>pose</i>	轴组基座在参考坐标系下的位姿。
<i>ref_frame</i>	(可选): <i>pose</i> 所在的参考坐标系名称，可以是任何带有坐标系的世界模型实体（如坐标系、轴组、轴等）。默认值“base”表示机器人基座坐标系。

返回

axisGroupAddAxis()

```
int arcs::common_interface::SyncMove::axisGroupAddAxis (
    const std::string & group_name,
    const std::string & name,
    const std::string & parent,
    const std::vector< double > & pose)
```

向名为 *group_name* 的轴组添加一个名为 *name* 的外部轴。该轴在 *parent* 坐标系下的 *pose* 位置附加，*pose* 表示轴位置为 0 时的位姿。轴的类型、最大速度、最大加速度、位置限制和索引分别由 *type*、*v_limit*、*a_limit*、*q_limits* 和 *axis_index* 定义。*pose* 参数通常通过外部轴调试标定流程获得。如果该轴组正被其他函数控制，或附加关系形成闭环，则操作会失败。

参数

<i>group_name</i>	要添加轴的轴组名称，需已通过 <i>axis_group_add()</i> 创建且存在。
<i>name</i>	新轴的名称，不能为空且需唯一。
<i>parent</i>	父轴名称，若为空或与 <i>group_name</i> 相同，则附加到轴组基座。父轴需已存在于该轴组。
<i>pose</i>	轴在父坐标系下的零位姿。 <i>type</i> 为 0（旋转轴）时， <i>z</i> 轴为旋转轴； <i>type</i> 为 1（直线轴）时， <i>z</i> 轴为移动方向。
<i>type</i>	轴类型，0 表示旋转轴，1 表示直线轴。
<i>v_limit</i>	最大速度。
<i>a_limit</i>	最大加速度。
<i>q_limits</i>	位置限制。
<i>axis_index</i>	轴索引。

axisGroupDelete()

```
int arcs::common_interface::SyncMove::axisGroupDelete (
    const std::string & name)
```

删除具有给定名称的轴组。

所有附加的轴也会被禁用（如果处于活动状态）并删除。

如果该轴组正被其他函数控制，则操作会失败。

参数

<i>name</i>	要删除的轴组名称。该名称的轴组必须存在。

返回

axisGroupGetActualPositions()

```
std::vector< double > arcs::common_interface::SyncMove::axisGroupGetActualPositions (  
    const std::string & group_name)
```

返回指定轴组的当前实际位置。如果未指定 `group_name`，则返回所有外部轴的实际位置。如果外部轴总线被禁用，则该函数会失败。

参数

<i>group_name</i>	(可选): (string) 要查询的轴组名称。该名称的轴组必须真实存在。

返回

Double[]: 涉及轴的实际位置，顺序为其外部轴索引顺序。

axisGroupGetAxisIndex()

```
int arcs::common_interface::SyncMove::axisGroupGetAxisIndex (  
    const std::string & name)
```

返回指定轴名称在 RTDE 目标位置和实际位置数组中的索引。

参数

<i>axis_name</i>	(string) 要查询的轴名称。该名称的轴必须存在。

返回

integer: 该轴在 RTDE 目标位置和实际位置数组中的索引。

axisGroupGetAxisName()

```
std::string arcs::common_interface::SyncMove::axisGroupGetAxisName (  
    int index)
```

返回指定轴索引对应的轴名称。

参数

<i>axis_index</i>	(整数) 要查询的轴索引。该索引的轴必须存在。

返回

字符串: 轴的名称。

axisGroupGetTargetPositions()

```
std::vector< double > arcs::common_interface::SyncMove::axisGroupGetTargetPositions (
    const std::string & group_name)
```

返回指定轴组的当前目标位置。如果未指定 group_name, 则返回所有外部轴的目标位置。
如果外部轴总线被禁用, 则该函数会失败。

参数

<i>group_name</i>	(可选): (string) 要查询的轴组名称。该名称的轴组必须真实存在。

返回

Double[]: 涉及轴的目标位置, 顺序为其外部轴索引顺序。

axisGroupMoveJoint()

```
int arcs::common_interface::SyncMove::axisGroupMoveJoint (
    const std::string & group_name,
    const std::vector< double > & q,
    double a,
    double v)
```

以梯形速度曲线将名为 group_name 的轴组移动到新的位置 q。参数 a 指定本次运动的最大加速度占各轴加速度极限的百分比。参数 v 指定本次运动的最大速度占各轴速度极限的百分比。

实际的加速度和速度由最受限制的轴决定, 以确保所有轴在加速、匀速和减速阶段同时完成。

参数

<i>group_name</i>	(string) 要移动的轴组名称。该名称的轴组必须存在。
<i>q</i>	(float[]) 目标位置, 旋转轴为弧度, 直线轴为米。如果目标超出位置极限, 则会被限制在最近的极限值。涉及的轴按其索引递增排序。q 的大小必须与该轴组包含的轴数量一致。
<i>a</i>	(float) 本次运动的最大加速度因子, 取值范围 (0,1], 表示占加速度极限的百分比。
<i>v</i>	(float) 本次运动的最大速度因子, 取值范围 (0,1], 表示占速度极限的百分比。

返回值: 无

axisGroupOffsetPositions()

```
int arcs::common_interface::SyncMove::axisGroupOffsetPositions (
    const std::string & group_name,
    const std::vector< double > & offset)
```

通过给定的 offset, 将轴组 group_name 的目标位置 and 实际位置整体偏移。

这是一个仅在控制器内部进行的软件偏移, 不会影响外部轴驱动器。该偏移也会应用于通过 RTDE 发布的任何目标和实际位置流。

参数

<i>group_name</i>	(string) 要应用偏移的轴组名称。该名称的轴组必须存在。
<i>offset</i>	(float[]) 目标和实际位置需要整体偏移的量。offset 的大小必须与该轴组所包含的轴数量一致。

返回

axisGroupSpeedJoint()

```
int arcs::common_interface::SyncMove::axisGroupSpeedJoint (
    const std::string & group_name,
    const std::vector< double > & qd,
    double a,
    double t)
```

以指定的加速度因子 a，将名为 group_name 的轴组加速到目标速度 qd。该函数会运行 t 秒。

参数

<i>group_name</i>	(string) 要控制的外部轴组名称，必须已存在。
<i>qd</i>	(float[]) 轴组各轴的目标速度。如果目标速度超过速度极限，则会被限制在极限值。涉及的轴按其索引递增排序。qd 的大小必须与该轴组包含的轴数量一致。
<i>a</i>	(float) 本次运动的最大加速度因子，取值范围 (0,1]，表示占加速度极限的百分比。
<i>t</i>	(可选): (float) 函数运行的持续时间（秒）。若 $t < 0$ ，则函数将在目标速度达到时返回；若 $t \geq 0$ ，则函数将在该持续时间后返回，无论实际速度是否达到目标值。

axisGroupUpdateAxis()

```
int arcs::common_interface::SyncMove::axisGroupUpdateAxis (
    const std::string & name,
    const std::vector< double > & pose)
```

更新指定名称的轴的相关属性。pose 参数通常通过外部轴调试标定流程获得。如果该轴所属的轴组正被其他命令控制，则操作会失败。如果该轴组中任何已附加的轴处于激活和使能状态，则操作会失败。

参数

<i>name</i>	要更新的轴的名称，需已存在。
<i>pose</i>	(可选): 轴在父轴（或轴组）坐标系下的零位姿。即轴位置为 0 时的位姿。

frameAdd()

```
int arcs::common_interface::SyncMove::frameAdd (
    const std::string & name,
    const std::vector< double > & pose,
    const std::string & ref_name)
```

添加一个名为 name 的坐标系，其初始位姿为 pose，位姿以 ref_frame 坐标系表达。此命令仅向世界模型添加一个坐标系，并不会将其附加到 ref_frame 坐标系。如需将新添加的坐标系附加到 ref_frame，请使用 [frameAttach\(\)](#)。

参数

<i>name</i>	要添加的坐标系名称。名称不能与任何已存在的世界模型对象（坐标系、轴或轴组）重复，否则会抛出异常。
<i>pose</i>	新对象的初始位姿。
<i>ref_frame</i>	位姿所表达的参考坐标系对象名称。若未指定，默认使用机器人“base”坐标系。

返回

frameAttach()

```
int arcs::common_interface::SyncMove::frameAttach (
    const std::string & child,
    const std::string & parent)
```

将子坐标系附加到父世界模型对象。附加时会设置父子之间的相对变换，使得子坐标系在世界中不会移动。

子坐标系不能为“world”、“flange”、“tcp”，也不能与父坐标系同名。

如果子或父不是已存在的坐标系，或导致形成闭环，则操作会失败。

如果用于 MotionPlus，parent 参数可以是外部轴或轴组的名称。

参数

<i>child</i>	要附加的子坐标系名称，不能为“world”、“flange”或“tcp”。
<i>parent</i>	子坐标系将要附加到的父对象名称。

返回

frameConvertPose()

```
std::vector< double > arcs::common_interface::SyncMove::frameConvertPose (
    const std::vector< double > & pose,
    const std::string & from_frame,
    const std::string & to_frame)
```

将位姿从 from_frame 坐标系转换到 to_frame 坐标系。

如果任一坐标系参数不是已存在的坐标系，则操作会失败。

如果用于 MotionPlus，所有三个参数也可以是外部轴或轴组的名称。

参数

<i>pose</i>	要转换的位姿
<i>from_frame</i>	原始坐标系的参考坐标系名称
<i>to_frame</i>	新坐标系的参考坐标系名称

返回

以 to_frame 坐标系表达的 pose 值。

frameDelete()

```
int arcs::common_interface::SyncMove::frameDelete (  
    const std::string & name)
```

删除指定名称的坐标系。

“world”、“base”、“flange”和“tcp”坐标系不能被删除。

任何附加到被删除坐标系的坐标系将会被附加到“world”坐标系，并设置新的偏移，使得被分离的坐标系在世界中不会移动。

如果指定的坐标系不存在，则操作会失败。

参数

<i>name</i>	要删除的坐标系名称

返回

frameDeleteAll()

```
int arcs::common_interface::SyncMove::frameDeleteAll ()
```

删除所有已添加到世界模型的坐标系。

“world”、“base”、“flange”和“tcp”坐标系不能被删除。

任何附加到被删除坐标系的坐标系将会被附加到“world”坐标系，并设置新的偏移，使得被分离的坐标系在世界中不会移动。

返回

frameExist()

```
bool arcs::common_interface::SyncMove::frameExist (  
    const std::string & name)
```

查询指定名称的坐标系是否存在。

参数

<i>name</i>	要查询的坐标系名称。

返回

如果存在该名称的坐标系则返回 true，否则返回 false。

frameGetChildren()

```
std::vector< std::string > arcs::common_interface::SyncMove::frameGetChildren (  
    const std::string & name)
```

返回指定父对象的直接子对象坐标系名称列表。父子关系由世界模型的附加关系定义。如果用于 MotionPlus，子对象也可以是轴组或轴。

参数

<i>name</i>	父对象的名称。

返回

直接子对象坐标系名称列表

frameGetParent()

```
std::string arcs::common_interface::SyncMove::frameGetParent (  
    const std::string & name)
```

获取名为 *name* 的坐标系在世界模型中的父坐标系名称。

如果该坐标系没有附加到其他坐标系，则其父坐标系为“world”。

参数

<i>name</i>	要查询的坐标系名称

返回

父坐标系的名称，字符串类型

frameGetPose()

```
std::vector< double > arcs::common_interface::SyncMove::frameGetPose (  
    const std::string & name,  
    const std::string & rel_frame,  
    const std::string & ref_frame)
```

获取名为 *name* 的坐标系相对于 *rel_frame* 坐标系的位姿，并以 *ref_frame* 坐标系表达。如果未提供 *ref_frame*，则返回 *name* 坐标系相对于 *rel_frame* 坐标系的位姿，并以 *rel_frame* 坐标系表达。

如果任一参数不是已存在的坐标系，则操作会失败。

如果用于 MotionPlus，所有三个参数也可以是外部轴或轴组的名称。

参数

<i>name</i>	要查询的坐标系名称。
<i>rel_frame</i>	“相对坐标系”，用于计算相对位姿的坐标系。
<i>ref_frame</i>	“参考坐标系”，用于表达结果相对位姿的坐标系。如果未提供，则默认为 <i>rel_frame</i> 。

返回

以 *ref_frame* 坐标系表达的 *name* 坐标系的位姿。

frameMove()

```
int arcs::common_interface::SyncMove::frameMove (
    const std::string & name,
    const std::vector< double > & pose,
    const std::string & ref_name)
```

更改名为 `name` 的坐标系的位置，将其移动到由 `pose` 指定的新位置，`pose` 以 `ref_name` 坐标系表达。如果 `name` 为“world”、“flange”、“tcp”，或该坐标系不存在，则操作会失败。注意：如需移动“tcp”坐标系，请使用 `set_tcp()` 命令。

如果用于 MotionPlus，`ref_name` 参数可以是外部轴或轴组的名称。

参数

<i>name</i>	要移动的坐标系名称
<i>pose</i>	新的位置
<i>ref_name</i>	<code>pose</code> 所表达的参考坐标系，默认值为机器人的“base”坐标系。

返回

isSyncMoveOn()

```
bool arcs::common_interface::SyncMove::isSyncMoveOn ()
```

`isSyncMoveOn` 用于判断机械单元组是否处于同步运动模式。

不控制任何机械单元的任务可以通过该函数判断参数“使用机械单元组”中定义的机械单元是否处于同步运动模式。

返回

Python 函数原型

```
isSyncMoveOn(self: pyaubo_sdk.SyncMove) -> bool
```

Lua 函数原型

```
isSyncMoveOn() -> boolean
```

syncMoveOff()

```
int arcs::common_interface::SyncMove::syncMoveOff (
    const std::string & syncident)
```

`syncMoveOff` 用于结束同步运动模式。

`syncMoveOff` 指令会等待其他任务程序。当所有任务程序都到达 `syncMoveOff` 时，它们将继续以非同步模式执行。在 `syncMoveOff` 指令之前必须编程一个停止点。

参数

<i>syncident</i>	

返回

Python 函数原型

```
syncMoveOff(self: pyaubo_sdk.SyncMove, arg0: str) -> int
```

Lua 函数原型

```
syncMoveOff(syncident: string) -> nil @endcoe
```

syncMoveOn()

```
int arcs::common_interface::SyncMove::syncMoveOn (
    const std::string & syncident,
    const TaskSet & taskset)
```

syncMoveOn 用于启动同步运动模式。

syncMoveOn 指令会等待其他任务程序。当所有任务程序都到达 syncMoveOn 时, 它们将继续以同步运动模式执行。不同任务程序中的移动指令将同时执行, 直到执行 syncMoveOff 指令为止。在 syncMoveOn 指令之前必须编程一个停止点。

参数

<i>syncident</i>	
<i>taskset</i>	

返回

Python 函数原型

```
syncMoveOn(self: pyaubo_sdk.SyncMove, arg0: str, arg1: Set[str]) -> int
```

Lua 函数原型

```
syncMoveOn(syncident: string, taskset: table) -> nil @endcoe
```

syncMoveResume()

```
int arcs::common_interface::SyncMove::syncMoveResume ()
```

恢复同步运动模式。

返回

Python 函数原型

```
syncMoveResume(self: pyaubo_sdk.SyncMove) -> int
```

Lua 函数原型

```
syncMoveResume() -> nil
```


syncMoveSegment()

```
bool arcs::common_interface::SyncMove::syncMoveSegment (
    int id)
```

设置同步路径段的ID 在同步运动模式下，所有同时执行的移动指令必须全部编程为圆角区（corner zones）或全部为停止点（stop points）。这意味着具有相同ID 的移动指令要么全部带有圆角区，要么全部带有停止点。如果在各自的程序任务中同步执行的移动指令中，一个带有圆角区而另一个带有停止点，则会发生错误。同步执行的移动指令可以有不同大小的圆角区（例如，一个使用 z10，另一个使用 z50）。

参数

<i>id</i>	

返回

Python 函数原型

```
syncMoveSegment(self: pyaubo_sdk.SyncMove, arg0: int) -> bool
```

Lua 函数原型

```
syncMoveSegment(id: number) -> boolean @endcode
```

syncMoveSuspend()

```
int arcs::common_interface::SyncMove::syncMoveSuspend ()
```

暂停同步运动模式。

返回

Python 函数原型

```
syncMoveSuspend(self: pyaubo_sdk.SyncMove) -> int
```

Lua 函数原型

```
syncMoveSuspend() -> nil
```

syncMoveUndo()

```
int arcs::common_interface::SyncMove::syncMoveUndo ()
```

syncMoveUndo 用于关闭同步运动，即使不是所有其他任务程序都执行了 syncMoveUndo 指令。
syncMoveUndo 主要用于 UNDO 处理程序。当程序指针从过程移动时，syncMoveUndo 用于关闭同步。

返回

Python 函数原型

```
syncMoveUndo(self: pyaubo_sdk.SyncMove) -> int
```

Lua 函数原型

```
syncMoveUndo() -> nil @endcode
```

waitSyncTasks()

```
int arcs::common_interface::SyncMove::waitSyncTasks (
    const std::string & syncident,
    const TaskSet & taskset)
```

waitSyncTasks 用于在程序中的特定点同步多个任务程序。

waitSyncTasks 指令会等待其他任务程序。当所有任务程序都到达 waitSyncTasks 指令时，它们将继续执行。

参数

<i>syncident</i>	
<i>taskset</i>	

返回**Python 函数原型**

```
waitSyncTasks(self: pyaubo_sdk.SyncMove, arg0: str, arg1: Set[str]) -> int
```

Lua 函数原型

```
waitSyncTasks(syncident: string, taskset: table) -> nil @endcode
```

8.11 SystemInfo (系统信息)**系统信息****函数**

- int [arcs::common_interface::SystemInfo::getControlSoftwareVersionCode \(\)](#)
获取控制器软件版本号
- std::string [arcs::common_interface::SystemInfo::getControlSoftwareFullVersion \(\)](#)
获取完整控制器软件版本号
- int [arcs::common_interface::SystemInfo::getInterfaceVersionCode \(\)](#)
获取接口版本号
- std::string [arcs::common_interface::SystemInfo::getControlSoftwareBuildDate \(\)](#)
获取控制器软件构建时间
- std::string [arcs::common_interface::SystemInfo::getControlSoftwareVersionHash \(\)](#)
获取控制器软件 *git* 版本
- uint64_t [arcs::common_interface::SystemInfo::getControlSystemTime \(\)](#)
获取系统时间 (软件启动时间 *ns* 纳秒)

8.11.1 详细描述**系统信息**

8.11.2 函数说明

getControlSoftwareBuildDate()

std::string arcs::common_interface::SystemInfo::getControlSoftwareBuildDate ()
获取控制器软件构建时间

返回

返回控制器软件构建时间

Python 函数原型

```
getControlSoftwareBuildDate(self: pyaubo_sdk.SystemInfo) -> str
```

Lua 函数原型

```
getControlSoftwareBuildDate() -> string
```

C++ 示例

```
std::string build_date =  
rpc_cli->getSystemInfo()->getControlSoftwareBuildDate();
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "SystemInfo.getControlSoftwareBuildDate", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": "2024-3-5 07:03:20"}
```

getControlSoftwareFullVersion()

std::string arcs::common_interface::SystemInfo::getControlSoftwareFullVersion ()
获取完整控制器软件版本号

返回

返回完整控制器软件版本号

Python 函数原型

```
getControlSoftwareFullVersion(self: pyaubo_sdk.SystemInfo) -> str
```

Lua 函数原型

```
getControlSoftwareFullVersion() -> string
```

C++ 示例

```
std::string control_version =  
rpc_cli->getSystemInfo()->getControlSoftwareFullVersion();
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "SystemInfo.getControlSoftwareFullVersion", "params": [], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": "0.31.0-alpha.16+20alc76"}
```

getControlSoftwareVersionCode()

```
int arcs::common_interface::SystemInfo::getControlSoftwareVersionCode ()
```

获取控制器软件版本号

返回

返回控制器软件版本号

Python 函数原型

```
getControlSoftwareVersionCode(self: pyaubo_sdk.SystemInfo) -> int
```

Lua 函数原型

```
getControlSoftwareVersionCode() -> number
```

C++ 示例

```
int control_version =
rpc_cli->getSystemInfo()->getControlSoftwareVersionCode();
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"SystemInfo.getControlSoftwareVersionCode","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":28003}
```

getControlSoftwareVersionHash()

```
std::string arcs::common_interface::SystemInfo::getControlSoftwareVersionHash ()
```

获取控制器软件 git 版本

返回

返回控制器软件 git 版本

Python 函数原型

```
getControlSoftwareVersionHash(self: pyaubo_sdk.SystemInfo) -> str
```

Lua 函数原型

```
getControlSoftwareVersionHash() -> string
```

C++ 示例

```
std::string git_version =
rpc_cli->getSystemInfo()->getControlSoftwareVersionHash();
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"SystemInfo.getControlSoftwareVersionHash","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":"fa4f64a"}
```

getControlSystemTime()

```
uint64_t arcs::common_interface::SystemInfo::getControlSystemTime ()  
获取系统时间 (软件启动时间 ns 纳秒)
```

返回

返回系统时间 (软件启动时间 ns 纳秒)

Python 函数原型

```
getControlSystemTime(self: pyaubo_sdk.SystemInfo) -> int
```

Lua 函数原型

```
getControlSystemTime() -> number
```

C++ 示例

```
std::string system_time =  
rpc_cli->getSystemInfo()->getControlSystemTime();
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"SystemInfo.getControlSystemTime","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":9287799079682}
```

getInterfaceVersionCode()

```
int arcs::common_interface::SystemInfo::getInterfaceVersionCode ()  
获取接口版本号
```

返回

返回接口版本号

Python 函数原型

```
getInterfaceVersionCode(self: pyaubo_sdk.SystemInfo) -> int
```

Lua 函数原型

```
getInterfaceVersionCode() -> number
```

C++ 示例

```
int interface_version =  
rpc_cli->getSystemInfo()->getInterfaceVersionCode();
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"SystemInfo.getInterfaceVersionCode","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":22003}
```

8.12 Trace (日志与弹窗)

提供给控制器扩展程序的日志记录系统

函数

- `int arcs::common_interface::Trace::alarm (TraceLevel level, int code, const std::vector< std::string > &args={})`
向 `aubo_control` 日志注入告警信息 `code` 定义参考 [error_stack](#)
- `int arcs::common_interface::Trace::textmsg (const std::string &msg)`
打印文本信息到日志中
- `int arcs::common_interface::Trace::popup (TraceLevel level, const std::string &title, const std::string &msg, int mode)`
向连接的 `RTDE` 客户端发送弹窗请求
- `RobotMsgVector arcs::common_interface::Trace::peek (size_t num, uint64_t last_time=0)`
`peek` 最新的 `AlarmInfo`(上次一获取之后)

8.12.1 详细描述

提供给控制器扩展程序的日志记录系统

8.12.2 函数说明

alarm()

```
int arcs::common_interface::Trace::alarm (
    TraceLevel level,
    int code,
    const std::vector< std::string > & args = {})
向 aubo_control 日志注入告警信息 code 定义参考 error\_stack
```

Python 函数原型 @par Lua 函数原型 @par JSON-RPC 请求示例 @par JSON-RPC 响应示例

peek()

```
RobotMsgVector arcs::common_interface::Trace::peek (
    size_t num,
    uint64_t last_time = 0)
peek 最新的 AlarmInfo(上次一获取之后)
last_time 设置为 0 时, 可以获取到所有的AlarmInfo
```

参数

<i>num</i>	
<i>last_time</i>	

返回

Python 函数原型

```
peek(self: pyaubo_sdk.Trace, arg0: int, arg1: int) -> List[arcs::common_interface::RobotMsg]
```

Lua 函数原型

```
peek(num: number, last_time: number) -> table
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.Trace.peek","params":[1,0],"id":1}
```

JSON-RPC 响应示例

```
{ "id":1,"jsonrpc":"2.0","result":[{"args":["RobotModeType.Running"], "code":30045,"level":"INFO","source":←
"rob1","timestamp":5102883064300}]} }
```

popup()

```
int arcs::common_interface::Trace::popup (
    TraceLevel level,
    const std::string & title,
    const std::string & msg,
    int mode)
```

向连接的 RTDE 客户端发送弹窗请求

参数

<i>level</i>	
<i>title</i>	
<i>msg</i>	
<i>mode</i>	模式 0: 普通模式 1: 阻塞模式 2: 输入模式 bool 3: 输入模式 int 4: 输入模式 double 5: 输入模式 string

返回

Python 函数原型

```
popup(self: pyaubo_sdk.Trace, arg0: arcs::common_interface::TraceLevel, arg1: str, arg2: str,
arg3: int) -> int
```

Lua 函数原型

```
popup(level: number, title: string, msg: string, mode: number) -> nil
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.Trace.popup","params":["","Error","Trajectory planning failed!",1]←
,"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

textmsg()

```
int arcs::common_interface::Trace::textmsg (  
    const std::string & msg)  
打印文本信息到日志中
```

参数

<i>msg</i>	文本信息

返回**Python 函数原型**

```
textmsg(self: pyaubo_sdk.Trace, arg0: str) -> int
```

Lua 函数原型

```
textmsg(msg: string) -> nil
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.Trace.textmsg","params":["test"],"id":1}
```

JSON-RPC 响应示例

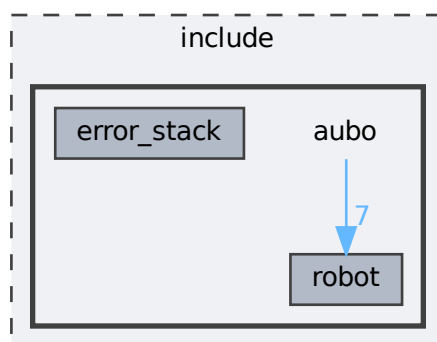
```
{"id":1,"jsonrpc":"2.0","result":0}
```


Chapter 9

目录说明

9.1 include/aubo 目录参考

aubo 的目录依赖关系图



目录

- 目录 [error_stack](#)
- 目录 [robot](#)

文件

- 文件 [aubo_api.h](#)
机器人及外部轴等控制API接口，如获取机器人列表、获取系统信息等等
- 文件 [axis_interface.h](#)
- 文件 [gripper_interface.h](#)
通用夹爪接口
- 文件 [math.h](#)
数学方法接口，如欧拉角与四元数转换、位姿的加减运算
- 文件 [register_control.h](#)
寄存器操作接口，用于三个模块之间的数据交换功能
- 文件 [robot_interface.h](#)
机器人API接口

- 文件 [runtime__machine.h](#)

Script interpreter runtime interface, allows pausing the script interpreter and setting/removing break-points.

- 文件 [serial.h](#)

串口通信

- 文件 [socket.h](#)

socket 通信

- 文件 [sync__move.h](#)

同步运行

- 文件 [system__info.h](#)

获取系统信息接口，如接口板的版本号、示教器软件的版本号

- 文件 [trace.h](#)

向控制器日志系统注入日志方面的接口

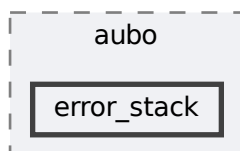
- 文件 [type__def.h](#)

数据类型的定义

9.2 doc 目录参考

9.3 include/aubo/error_stack 目录参考

error_stack 的目录依赖关系图



文件

- 文件 [error_stack.h](#)

汇总错误码

- 文件 [hal_error.h](#)

定义硬件抽象层的错误码

- 文件 [rtm_error.h](#)

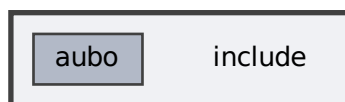
运行时错误码

- 文件 [system_error.h](#)

系统错误码

9.4 include 目录参考

include 的目录依赖关系图

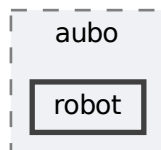


目录

- 目录 [aubo](#)

9.5 include/aubo/robot 目录参考

robot 的目录依赖关系图



文件

- 文件 [force_control.h](#)
- 文件 [io_control.h](#)
IO 控制接口
- 文件 [motion_control.h](#)
运动控制接口
- 文件 [robot_algorithm.h](#)
机器人算法相关的对外接口
- 文件 [robot_config.h](#)
获取机器人配置接口，如获取DH 参数、碰撞等级、安装位姿等等
- 文件 [robot_manage.h](#)
机器人管理接口，如上电、启动、拖动示教模式等
- 文件 [robot_state.h](#)
获取机器人状态接口，如关节速度、关节角度、固件/硬件版本

Chapter 10

命名空间文档

10.1 arcs 命名空间参考

命名空间

- namespace [common_interface](#)
- namespace [error_stack](#)

10.2 arcs::common_interface 命名空间参考

类

- class [AuboApi](#)
- class [AxisInterface](#)
- class [GripperInterface](#)
- class [Math](#)
- class [RegisterControl](#)
- class [ForceControl](#)
- class [IoControl](#)
- class [MotionControl](#)
- class [RobotAlgorithm](#)
- class [RobotConfig](#)
- class [RobotManage](#)
- class [RobotState](#)
- class [RobotInterface](#)
- class [RuntimeMachine](#)
- class [Serial](#)
- class [Socket](#)
- class [SyncMove](#)
- class [SystemInfo](#)
- class [Trace](#)
- struct [RobotSafetyParameterRange](#)
- struct [WObjectData](#)
- struct [VibrationRecalibrationParameter](#)
- struct [CircleParameters](#)
 - 圆周运动参数定义
- struct [SpiralParameters](#)
- struct [Enveloping](#)
- struct [TrajConfig](#)
 - 用于负载辨识的轨迹配置
- struct [RobotMsg](#)
- struct [GripperStatus](#)

- struct [RtdeRecipe](#)
RTDE 菜单
- class [AuboException](#)
自定义异常类 *AuboException*

类型定义

- using [AuboApiPtr](#) = std::shared_ptr<[AuboApi](#)>
- using [AxisInterfacePtr](#) = std::shared_ptr<[AxisInterface](#)>
- using [GripperInterfacePtr](#) = std::shared_ptr<[GripperInterface](#)>
- using [MathPtr](#) = std::shared_ptr<[Math](#)>
- using [RegisterControlPtr](#) = std::shared_ptr<[RegisterControl](#)>
- using [ForceControlPtr](#) = std::shared_ptr<[ForceControl](#)>
- using [IoControlPtr](#) = std::shared_ptr<[IoControl](#)>
- using [MotionControlPtr](#) = std::shared_ptr<[MotionControl](#)>
- using [RobotAlgorithmPtr](#) = std::shared_ptr<[RobotAlgorithm](#)>
- using [RobotConfigPtr](#) = std::shared_ptr<[RobotConfig](#)>
- using [RobotManagePtr](#) = std::shared_ptr<[RobotManage](#)>
- using [RobotStatePtr](#) = std::shared_ptr<[RobotState](#)>
- using [RobotInterfacePtr](#) = std::shared_ptr<[RobotInterface](#)>
- using [RuntimeMachinePtr](#) = std::shared_ptr<[RuntimeMachine](#)>
- using [SerialPtr](#) = std::shared_ptr<[Serial](#)>
- using [SocketPtr](#) = std::shared_ptr<[Socket](#)>
- typedef std::unordered_set< std::string > [TaskSet](#)
- using [SyncMovePtr](#) = std::shared_ptr<[SyncMove](#)>
- using [SystemInfoPtr](#) = std::shared_ptr<[SystemInfo](#)>
- using [TracePtr](#) = std::shared_ptr<[Trace](#)>
- using [Vector3d](#) = std::array<double, 3>
- using [Vector4d](#) = std::array<double, 4>
- using [Vector3f](#) = std::array<float, 3>
- using [Vector4f](#) = std::array<float, 4>
- using [Vector6f](#) = std::array<float, 6>
- using [ResultWithErrno](#) = std::tuple<std::vector<double>, int>
- using [ResultWithErrno1](#) = std::tuple<std::vector<std::vector<double>>, int>
- using [ResultWithErrno2](#) = std::tuple<std::vector<std::string>, int>
- using [ResultWithErrno3](#) = std::tuple<int, int>
- using [Payload](#)
- using [ForceSensorCalibResult](#)
- using [ForceSensorCalibResultWithError](#)
- using [DynamicsModel](#)
- using [Box](#) = std::vector<double>
- using [Cylinder](#) = std::vector<double>
- using [Sphere](#) = std::vector<double>
- using [RobotMsgVector](#) = std::vector<[RobotMsg](#)>
- using [GripperStatusVector](#) = std::vector<[GripperStatus](#)>

枚举

- enum [PathBufferType](#) {
 [PathBuffer_TOPPRA](#) = 1, [PathBuffer_CubicSpline](#), [PathBuffer_JointSpline](#) = 3, [PathBuffer_JointSplineC](#)
 = 4,
 [PathBuffer_JointBSpline](#) = 5, [PathBuffer_JointBSplineC](#) = 6 }
 pathBuffer 类型
- enum [AuboErrorCodes](#) : int { [ENUM_AuboErrorCodes_DECLARES](#) }
- enum class [RuntimeState](#) : int { [ENUM_RuntimeState_DECLARES](#) }

- enum class [RobotModeType](#) : int { [ENUM_RobotModeType_DECLARES](#) }
- enum class [AxisModeType](#) : int { [ENUM_AxisModeType_DECLARES](#) }
- enum class [HandleStateType](#) : int { [ENUM_HandleStateType_DECLARES](#) }
- enum class [HandleModeType](#) : int { [ENUM_HandleModeType_DECLARES](#) }
- enum class [SafetyModeType](#) : int { [ENUM_SafetyModeType_DECLARES](#) }
- 安全状态:
 - enum class [OperationalModeType](#) : int { [ENUM_OperationalModeType_DECLARES](#) }
 - 操作模式
 - enum class [RobotControlModeType](#) : int { [ENUM_RobotControlModeType_DECLARES](#) }
 - 机器人控制模式
 - enum class [JointServoModeType](#) : int { [ENUM_JointServoModeType_DECLARES](#) }
 - 关节伺服模式
 - enum class [JointStateType](#) : int { [ENUM_JointStateType_DECLARES](#) }
 - 关节状态
 - enum class [StandardOutputRunState](#) : int { [ENUM_StandardOutputRunState_DECLARES](#) }
 - 标准输出运行状态
 - enum class [StandardInputAction](#) : int { [ENUM_StandardInputAction_DECLARES](#) }
 - The StandardInputAction enum*
 - enum class [SafetyInputAction](#) : int { [ENUM_SafetyInputAction_DECLARES](#) }
 - enum class [SafetyOutputRunState](#) : int { [ENUM_SafetyOutputRunState_DECLARES](#) }
 - enum [TaskFrameType](#) { [ENUM_TaskFrameType_DECLARES](#) }
 - enum [EnvelopingShape](#) : int { [ENUM_EnvelopingShape_DECLARES](#) }
 - enum [PayloadIdentifyMoveAxis](#) : int { [ENUM_PayloadIdentifyMoveAxis_DECLARES](#) }
 - enum [TraceLevel](#) { [ENUM_TraceLevel_DECLARES](#) }
 - enum [SafeguardedStopType](#) : int { [ENUM_SafeguardedStopType_DECLARES](#) }
 - enum [RobotEmergencyStopType](#) : int { [ENUM_RobotEmergencyStopType_DECLARES](#) }
 - enum class [ForceControlState](#) { [Stopped](#) , [Starting](#) , [Stropping](#) , [Running](#) }
 - enum class [RefFrameType](#) { [None](#) , [Tool](#) , [Path](#) , [Base](#) }
 - enum [error_type](#) {
[parse_error](#) = -32700 , [invalid_request](#) = -32600 , [method_not_found](#) = -32601 , [invalid_params](#)
 = -32602 ,
[internal_error](#) = -32603 , [server_error](#) = -32500 , [invalid](#) = -32400 }
 异常类型
 - enum [ExceptionCode](#) {
[EC_DISCONNECTED](#) = -1 , [EC_NOT_LOGINED](#) = -2 , [EC_INVAL_SOCKET](#) = -3 , [EC_REQUEST_BUSY](#)
 = -4 ,
[EC_SEND_FAILED](#) = -5 , [EC_RECV_TIMEOUT](#) = -6 , [EC_RECV_ERROR](#) = -7 , [EC_PARSE_ERROR](#)
 = -8 ,
[EC_INVALID_REQUEST](#) = -9 , [EC_METHOD_NOT_FOUND](#) = -10 , [EC_INVALID_PARAMS](#)
 = -11 , [EC_INTERNAL_ERROR](#) = -12 ,
[EC_SERVER_ERROR](#) = -13 , [EC_INVALID](#) = -14 }
 异常码

函数

- std::ostream & [operator<<](#) (std::ostream &os, const [RobotSafetyParameterRange](#) &vd)
- std::ostream & [operator<<](#) (std::ostream &os, [WObjectData](#) p)
- std::ostream & [operator<<](#) (std::ostream &os, const [VibrationRecalibrationParameter](#) p)
- std::ostream & [operator<<](#) (std::ostream &os, [CircleParameters](#) p)
- std::ostream & [operator<<](#) (std::ostream &os, [SpiralParameters](#) p)
- std::ostream & [operator<<](#) (std::ostream &os, [Enveloping](#) p)
- std::ostream & [operator<<](#) (std::ostream &os, [TrajConfig](#) p)
- const char * [returnValue2Str](#) (int retval)

10.2.1 类型定义说明

AuboApiPtr

using `arcs::common_interface::AuboApiPtr` = `std::shared_ptr<AuboApi>`
在文件 `aubo_api.h` 第 390 行定义.

AxisInterfacePtr

using `arcs::common_interface::AxisInterfacePtr` = `std::shared_ptr<AxisInterface>`
在文件 `axis_interface.h` 第 639 行定义.

Box

using `arcs::common_interface::Box` = `std::vector<double>`
在文件 `type_def.h` 第 841 行定义.

Cylinder

using `arcs::common_interface::Cylinder` = `std::vector<double>`
在文件 `type_def.h` 第 848 行定义.

DynamicsModel

using `arcs::common_interface::DynamicsModel`
初始值:

`std::tuple<std::vector<double>, std::vector<double>, std::vector<double>>`
在文件 `type_def.h` 第 832 行定义.

ForceControlPtr

using `arcs::common_interface::ForceControlPtr` = `std::shared_ptr<ForceControl>`
在文件 `force_control.h` 第 2618 行定义.

ForceSensorCalibResult

using `arcs::common_interface::ForceSensorCalibResult`
初始值:

`std::tuple<std::vector<double>, std::vector<double>, double,`
 `std::vector<double>>`
在文件 `type_def.h` 第 822 行定义.

ForceSensorCalibResultWithError

using `arcs::common_interface::ForceSensorCalibResultWithError`
初始值:

`std::tuple<std::vector<double>, std::vector<double>, double,`
 `std::vector<double>, double>`
在文件 `type_def.h` 第 827 行定义.

GripperInterfacePtr

using `arcs::common_interface::GripperInterfacePtr` = `std::shared_ptr<GripperInterface>`
在文件 `gripper_interface.h` 第 287 行定义.

GripperStatusVector

using `arcs::common_interface::GripperStatusVector` = `std::vector<GripperStatus>`
在文件 `type_def.h` 第 891 行定义.

IoControlPtr

using `arcs::common_interface::IoControlPtr` = `std::shared_ptr<IoControl>`
在文件 `io_control.h` 第 5774 行定义.

MathPtr

using `arcs::common_interface::MathPtr` = `std::shared_ptr<Math>`
在文件 `math.h` 第 1073 行定义.

MotionControlPtr

using `arcs::common_interface::MotionControlPtr` = `std::shared_ptr<MotionControl>`
在文件 `motion_control.h` 第 6779 行定义.

Payload

using `arcs::common_interface::Payload`
初始值:
`std::tuple<double, std::vector<double>, std::vector<double>, std::vector<double>>`
在文件 `type_def.h` 第 818 行定义.

RegisterControlPtr

using `arcs::common_interface::RegisterControlPtr` = `std::shared_ptr<RegisterControl>`
在文件 `register_control.h` 第 3548 行定义.

ResultWithErrno

using `arcs::common_interface::ResultWithErrno` = `std::tuple<std::vector<double>, int>`
在文件 `type_def.h` 第 812 行定义.

ResultWithErrno1

using `arcs::common_interface::ResultWithErrno1` = `std::tuple<std::vector<std::vector<double>>, int>`
在文件 `type_def.h` 第 813 行定义.

ResultWithErrno2

using `arcs::common_interface::ResultWithErrno2` = `std::tuple<std::vector<std::string>, int>`
在文件 `type_def.h` 第 814 行定义.

ResultWithErrno3

using `arcs::common_interface::ResultWithErrno3` = `std::tuple<int, int>`
在文件 `type_def.h` 第 815 行定义.

RobotAlgorithmPtr

using `arcs::common_interface::RobotAlgorithmPtr` = `std::shared_ptr<RobotAlgorithm>`
在文件 `robot_algorithm.h` 第 1986 行定义.

RobotConfigPtr

using `arcs::common_interface::RobotConfigPtr` = `std::shared_ptr<RobotConfig>`
在文件 `robot_config.h` 第 3808 行定义.

RobotInterfacePtr

using `arcs::common_interface::RobotInterfacePtr` = `std::shared_ptr<RobotInterface>`
在文件 `robot_interface.h` 第 391 行定义.

RobotManagePtr

```
using arcs::common_interface::RobotManagePtr = std::shared_ptr<RobotManage>
```

在文件 `robot_manage.h` 第 1917 行定义.

RobotMsgVector

```
using arcs::common_interface::RobotMsgVector = std::vector<RobotMsg>
```

在文件 `type_def.h` 第 872 行定义.

RobotStatePtr

```
using arcs::common_interface::RobotStatePtr = std::shared_ptr<RobotState>
```

在文件 `robot_state.h` 第 3510 行定义.

RuntimeMachinePtr

```
using arcs::common_interface::RuntimeMachinePtr = std::shared_ptr<RuntimeMachine>
```

在文件 `runtime_machine.h` 第 1620 行定义.

SerialPtr

```
using arcs::common_interface::SerialPtr = std::shared_ptr<Serial>
```

在文件 `serial.h` 第 407 行定义.

SocketPtr

```
using arcs::common_interface::SocketPtr = std::shared_ptr<Socket>
```

在文件 `socket.h` 第 642 行定义.

Sphere

```
using arcs::common_interface::Sphere = std::vector<double>
```

在文件 `type_def.h` 第 853 行定义.

SyncMovePtr

```
using arcs::common_interface::SyncMovePtr = std::shared_ptr<SyncMove>
```

在文件 `sync_move.h` 第 1011 行定义.

SystemInfoPtr

```
using arcs::common_interface::SystemInfoPtr = std::shared_ptr<SystemInfo>
```

在文件 `system_info.h` 第 349 行定义.

TaskSet

```
typedef std::unordered_set<std::string> arcs::common_interface::TaskSet
```

在文件 `sync_move.h` 第 36 行定义.

TracePtr

```
using arcs::common_interface::TracePtr = std::shared_ptr<Trace>
```

在文件 `trace.h` 第 232 行定义.

Vector3d

```
using arcs::common_interface::Vector3d = std::array<double, 3>
```

在文件 `type_def.h` 第 31 行定义.

Vector3f

using `arcs::common_interface::Vector3f` = `std::array<float, 3>`
 在文件 `type_def.h` 第 33 行定义.

Vector4d

using `arcs::common_interface::Vector4d` = `std::array<double, 4>`
 在文件 `type_def.h` 第 32 行定义.

Vector4f

using `arcs::common_interface::Vector4f` = `std::array<float, 4>`
 在文件 `type_def.h` 第 34 行定义.

Vector6f

using `arcs::common_interface::Vector6f` = `std::array<float, 6>`
 在文件 `type_def.h` 第 35 行定义.

10.2.2 枚举类型说明**AuboErrorCodes**

enum `arcs::common_interface::AuboErrorCodes` : int

枚举值

ENUM_AuboErrorCodes_DECLARES	

在文件 `type_def.h` 第 562 行定义.

AxisModeType

enum class `arcs::common_interface::AxisModeType` : int [strong]

枚举值

ENUM_AxisModeType_DECLARES	

在文件 `type_def.h` 第 577 行定义.

EnvelopingShape

enum `arcs::common_interface::EnvelopingShape` : int

枚举值

ENUM_EnvelopingShape_DECLARES	

在文件 `type_def.h` 第 664 行定义.

error_type

enum `arcs::common_interface::error_type`
 异常类型

枚举值

parse_error	解析错误
invalid_request	无效请求
method_not_found	方法未找到
invalid_params	无效参数
internal_error	内部错误
server_error	服务器错误
invalid	无效

在文件 `type_def.h` 第 906 行定义.

ExceptionCode

```
enum arcs::common_interface::ExceptionCode
异常码
```

枚举值

EC_DISCONNECTED	断开连接
EC_NOT_LOGINED	未登录
EC_INVALID_SOCKET	无效套接字
EC_REQUEST_BUSY	请求繁忙
EC_SEND_FAILED	发送失败
EC_RECV_TIMEOUT	接收超时
EC_RECV_ERROR	接收错误
EC_PARSE_ERROR	解析错误
EC_INVALID_REQUEST	无效请求
EC_METHOD_NOT_FOUND	方法未找到
EC_INVALID_PARAMS	无效参数
EC_INTERNAL_ERROR	内部错误
EC_SERVER_ERROR	服务器错误
EC_INVALID	无效

在文件 `type_def.h` 第 918 行定义.

ForceControlState

```
enum class arcs::common_interface::ForceControlState [strong]
```

枚举值

Stopped	
Starting	
Stropping	
Running	

在文件 [type_def.h](#) 第 729 行定义.

HandleModeType

```
enum class arcs::common_interface::HandleModeType : int [strong]
```

枚举值

ENUM_HandleModeType_DECLARES	

在文件 [type_def.h](#) 第 587 行定义.

HandleStateType

```
enum class arcs::common_interface::HandleStateType : int [strong]
```

枚举值

ENUM_HandleStateType_DECLARES	

在文件 [type_def.h](#) 第 582 行定义.

JointServoModeType

```
enum class arcs::common_interface::JointServoModeType : int [strong]
```

关节伺服模式

枚举值

ENUM_JointServoModeType_DECLARES	

在文件 [type_def.h](#) 第 620 行定义.

JointStateType

```
enum class arcs::common_interface::JointStateType : int [strong]
```

关节状态

枚举值

ENUM_JointStateType_DECLARES	

在文件 [type_def.h](#) 第 628 行定义.

OperationalModeType

```
enum class arcs::common_interface::OperationalModeType : int [strong]
```

操作模式

枚举值

ENUM_OperationalModeType_DECLARES	

在文件 `type_def.h` 第 604 行定义.

PathBufferType

```
enum arcs::common_interface::PathBufferType
```

pathBuffer 类型

机器人运动被编程为位姿到位姿的运动，即从当前位置移动到新位置。两点之间的路径由机器人自动计算。

枚举值

PathBuffer_TOPPRA	1: toppra 时间最优路径规划
PathBuffer_CubicSpline	2: cubic_spline(录制的轨迹)
PathBuffer_JointSpline	3: 关节B 样条插值，最少三个点废弃，建议用 5 替代，现在实际是关节空间 CUBIC_SPLINE
PathBuffer_JointSplineC	4:关节B 样条插值，最少三个点，但是传入的是笛卡尔空间位姿 5: 关节B 样条插值，最少三个点
PathBuffer_JointBSpline	
PathBuffer_JointBSplineC	6:关节B 样条插值，最少三个点，但是传入的是笛卡尔空间位姿

在文件 `motion_control.h` 第 34 行定义.

PayloadIdentifyMoveAxis

```
enum arcs::common_interface::PayloadIdentifyMoveAxis : int
```

枚举值

ENUM_PayloadIdentifyMoveAxis_DECLARES	

在文件 `type_def.h` 第 669 行定义.

RefFrameType

```
enum class arcs::common_interface::RefFrameType [strong]
```

枚举值

None	
Tool	工具坐标系
Path	轨迹坐标系
Base	基坐标系

在文件 `type_def.h` 第 737 行定义.

RobotControlModeType

```
enum class arcs::common_interface::RobotControlModeType : int [strong]
```

机器人控制模式

枚举值

ENUM_RobotControlModeType_DECLARES	

在文件 [type_def.h](#) 第 612 行定义.

RobotEmergencyStopType

```
enum arcs::common_interface::RobotEmergencyStopType : int
```

枚举值

ENUM_RobotEmergencyStopType_DECLARES	

在文件 [type_def.h](#) 第 684 行定义.

RobotModeType

```
enum class arcs::common_interface::RobotModeType : int [strong]
```

枚举值

ENUM_RobotModeType_DECLARES	

在文件 [type_def.h](#) 第 572 行定义.

RuntimeState

```
enum class arcs::common_interface::RuntimeState : int [strong]
```

枚举值

ENUM_RuntimeState_DECLARES	

在文件 [type_def.h](#) 第 567 行定义.

SafeguedStopType

```
enum arcs::common_interface::SafeguedStopType : int
```

枚举值

ENUM_SafeguedStopType_DECLARES	

在文件 [type_def.h](#) 第 679 行定义.

SafetyInputAction

```
enum class arcs::common_interface::SafetyInputAction : int [strong]
```

枚举值

ENUM_SafetyInputAction_DECLARES	

在文件 [type_def.h](#) 第 649 行定义.

SafetyModeType

```
enum class arcs::common_interface::SafetyModeType : int [strong]
```

安全状态:

枚举值

ENUM_SafetyModeType_DECLARES	

在文件 [type_def.h](#) 第 596 行定义.

SafetyOutputRunState

```
enum class arcs::common_interface::SafetyOutputRunState : int [strong]
```

枚举值

ENUM_SafetyOutputRunState_DECLARES	

在文件 [type_def.h](#) 第 654 行定义.

StandardInputAction

```
enum class arcs::common_interface::StandardInputAction : int [strong]
```

The [StandardInputAction](#) enum

枚举值

ENUM_StandardInputAction_DECLARES	

在文件 [type_def.h](#) 第 644 行定义.

StandardOutputRunState

```
enum class arcs::common_interface::StandardOutputRunState : int [strong]
```

标准输出运行状态

枚举值

ENUM_StandardOutputRunState_DECLARES	

在文件 [type_def.h](#) 第 636 行定义.

TaskFrameType

```
enum arcs::common_interface::TaskFrameType
```


枚举值

ENUM_TaskFrameType_DECLARES	

在文件 [type_def.h](#) 第 659 行定义.

TraceLevel

```
enum arcs::common_interface::TraceLevel
```

枚举值

ENUM_TraceLevel_DECLARES	

在文件 [type_def.h](#) 第 674 行定义.

10.2.3 函数说明

operator<<() [1/7]

```
std::ostream & arcs::common_interface::operator<< (
    std::ostream & os,
    CircleParameters p) [inline]
```

在文件 [type_def.h](#) 第 760 行定义.

operator<<() [2/7]

```
std::ostream & arcs::common_interface::operator<< (
    std::ostream & os,
    const RobotSafetyParameterRange & vd) [inline]
```

在文件 [type_def.h](#) 第 180 行定义.

operator<<() [3/7]

```
std::ostream & arcs::common_interface::operator<< (
    std::ostream & os,
    const VibrationRecalibrationParameter p) [inline]
```

在文件 [type_def.h](#) 第 223 行定义.

operator<<() [4/7]

```
std::ostream & arcs::common_interface::operator<< (
    std::ostream & os,
    Enveloping p) [inline]
```

在文件 [type_def.h](#) 第 788 行定义.

operator<<() [5/7]

```
std::ostream & arcs::common_interface::operator<< (
    std::ostream & os,
    SpiralParameters p) [inline]
```

在文件 [type_def.h](#) 第 774 行定义.

operator<<() [6/7]

```
std::ostream & arcs::common_interface::operator<< (
    std::ostream & os,
    TrajConfig p) [inline]
```

在文件 [type_def.h](#) 第 806 行定义.

operator<<() [7/7]

```
std::ostream & arcs::common_interface::operator<< (
    std::ostream & os,
    WObjectData p) [inline]
```

在文件 [type_def.h](#) 第 208 行定义.

returnValue2Str()

```
const char * arcs::common_interface::returnValue2Str (
    int retval) [inline]
```

在文件 [type_def.h](#) 第 972 行定义.

引用了 [ENUM_AuboErrorCodes_DECLARES](#).

10.3 arcs::error_stack 命名空间参考

枚举

- enum [ErrorCodes](#) { [ARCS_ERROR_CODES](#) }

函数

- constexpr int [ARCS_ABI_EXPORT codeCompose](#) (int aa, int bb, int cccc)
- constexpr int [ARCS_ABI_EXPORT mod](#) (int x)
- int [str2ErrorCode](#) (const char *err_code_name)
- const char * [errorCode2Str](#) (int err_code)
- std::ostream & [dump](#) (std::ostream &os)

10.3.1 枚举类型说明

ErrorCodes

```
enum arcs::error_stack::ErrorCodes
```

枚举值

ARCS_ERROR_CODES	

在文件 [error_stack.h](#) 第 56 行定义.

10.3.2 函数说明

codeCompose()

```
int ARCS_ABI_EXPORT arcs::error_stack::codeCompose (
    int aa,
    int bb,
    int cccc) [constexpr]
```

在文件 [error_stack.h](#) 第 27 行定义.

dump()

```
std::ostream & arcs::error_stack::dump (
```

```
    std::ostream & os) [inline]
```

在文件 [error_stack.h](#) 第 103 行定义.

引用了 [ARCS_ERROR_CODES](#).

errorCode2Str()

```
const char * arcs::error_stack::errorCode2Str (
```

```
    int err_code) [inline]
```

在文件 [error_stack.h](#) 第 73 行定义.

引用了 [ARCS_ERROR_CODES](#).

mod()

```
int ARCS_ABI_EXPORT arcs::error_stack::mod (
```

```
    int x) [constexpr]
```

在文件 [error_stack.h](#) 第 32 行定义.

str2ErrorCode()

```
int arcs::error_stack::str2ErrorCode (
```

```
    const char * err_code_name) [inline]
```

在文件 [error_stack.h](#) 第 63 行定义.

Chapter 11

类说明

11.1 arcs::common_interface::AuboApi 类参考

```
#include <aubo_api.h>
```

Public 成员函数

- [AuboApi](#) ()
- virtual [~AuboApi](#) ()
- [MathPtr](#) [getMath](#) ()
Math (数学工具) 获取纯数学相关接口
- [SystemInfoPtr](#) [getSystemInfo](#) ()
SystemInfo (系统信息) 获取系统信息
- [RuntimeMachinePtr](#) [getRuntimeMachine](#) ()
RuntimeMachine (运行时管理) 获取运行时接口
- [RegisterControlPtr](#) [getRegisterControl](#) ()
RegisterControl (寄存器操作) 对外寄存器接口
- [std::vector< std::string >](#) [getRobotNames](#) ()
获取机器人列表
- [RobotInterfacePtr](#) [getRobotInterface](#) (const [std::string](#) &name)
RobotInterface (机器人模块) 根据名字获取 *RobotInterfacePtr* 接口
- [std::vector< std::string >](#) [getAxisNames](#) ()
获取外部轴列表
- [AxisInterfacePtr](#) [getAxisInterface](#) (const [std::string](#) &name)
AxisInterface (外部轴) 获取外部轴接口
- [SocketPtr](#) [getSocket](#) ()
Socket (*socket* 网络通信) 获取 *socket*
- [SerialPtr](#) [getSerial](#) ()
Serial (串口通信) 获取 *Serial* 串口
- [SyncMovePtr](#) [getSyncMove](#) (const [std::string](#) &name)
SyncMove (同步运动控制和轴组管理) 获取同步运动接口
- [TracePtr](#) [getTrace](#) (const [std::string](#) &name)
Trace (日志与弹窗) 获取告警信息接口
- [GripperInterfacePtr](#) [getGripperInterface](#) ()
GripperInterface (夹爪) 获取通用夹爪接口

Protected 属性

- void * [d_](#) { nullptr }

11.1.1 详细描述

在文件 [aubo_api.h](#) 第 27 行定义.

11.1.2 构造及析构函数说明

AuboApi()

```
arcs::common_interface::AuboApi::AuboApi ()
```

~AuboApi()

```
virtual arcs::common_interface::AuboApi::~~AuboApi () [virtual]
```

11.1.3 类成员变量说明

d_

```
void* arcs::common_interface::AuboApi::d_ { nullptr } [protected]
```

在文件 [aubo_api.h](#) 第 388 行定义.

该类的文档由以下文件生成:

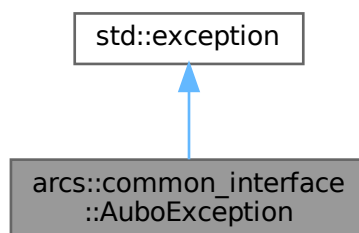
- [include/aubo/aubo_api.h](#)

11.2 arcs::common_interface::AuboException 类参考

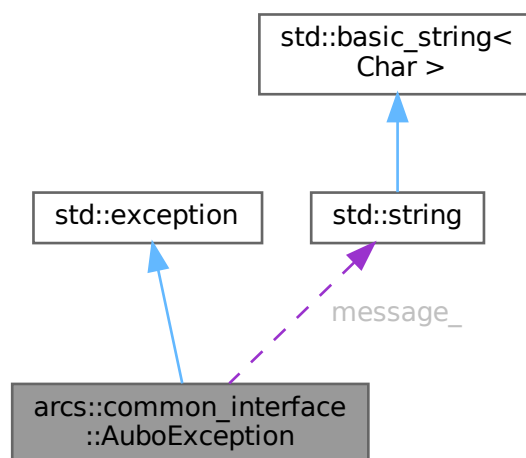
自定义异常类 [AuboException](#)

```
#include <type_def.h>
```

类 arcs::common_interface::AuboException 继承关系图:



arcs::common_interface::AuboException 的协作图:



Public 成员函数

- [AuboException](#) (int `code`, const std::string &prefix, const std::string &message) noexcept
- [AuboException](#) (int `code`, const std::string &message) noexcept
- [error_type](#) type () const
- int `code` () const
- const char * [what](#) () const noexcept override

Private 属性

- int `code__`
异常码
- std::string `message__`
异常消息 *Exception message*

11.2.1 详细描述

自定义异常类 [AuboException](#)
在文件 [type_def.h](#) 第 938 行定义.

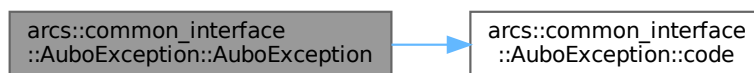
11.2.2 构造及析构函数说明

AuboException() [1/2]

```
arcs::common_interface::AuboException::AuboException (
    int code,
    const std::string & prefix,
    const std::string & message) [inline], [noexcept]
```

在文件 [type_def.h](#) 第 941 行定义.
引用了 [code\(\)](#).

函数调用图:



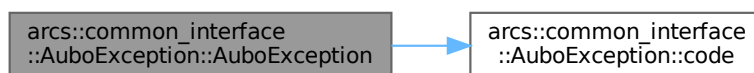
AboException() [2/2]

```
arcs::common_interface::AboException::AboException (
    int code,
    const std::string & message) [inline], [noexcept]
```

在文件 [type_def.h](#) 第 947 行定义.

引用了 [code\(\)](#).

函数调用图:



11.2.3 成员函数说明

code()

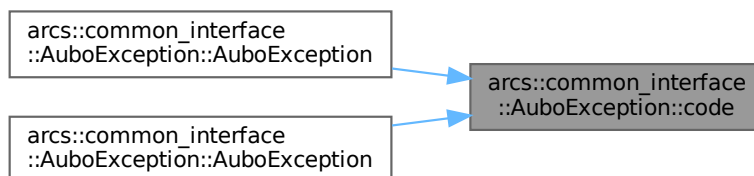
```
int arcs::common_interface::AboException::code () const [inline]
```

在文件 [type_def.h](#) 第 964 行定义.

引用了 [code_](#).

被这些函数引用 [AboException\(\)](#), 以及 [AboException\(\)](#).

这是这个函数的调用关系图:



type()

```
error_type arcs::common_interface::AboException::type () const [inline]
```

在文件 [type_def.h](#) 第 952 行定义.

引用了 [code_](#), [arcs::common_interface::invalid](#), [arcs::common_interface::parse_error](#), 以及 [arcs::common_interface::](#)

what()

const char * arcs::common_interface::AuboException::what () const [inline], [override], [noexcept]
 在文件 [type_def.h](#) 第 965 行定义.
 引用了 [message_](#).

11.2.4 类成员变量说明**code_**

int arcs::common_interface::AuboException::code_ [private]
 异常码
 在文件 [type_def.h](#) 第 968 行定义.
 被这些函数引用 [code\(\)](#) , 以及 [type\(\)](#).

message_

std::string arcs::common_interface::AuboException::message_ [private]
 异常消息 Exception message
 在文件 [type_def.h](#) 第 969 行定义.
 被这些函数引用 [what\(\)](#).
 该类的文档由以下文件生成:

- [include/aubo/type_def.h](#)

11.3 arcs::common_interface::AxisInterface 类参考

```
#include <axis_interface.h>
```

Public 成员函数

- [AxisInterface](#) ()
- virtual [~AxisInterface](#) ()
- int [poweronExtAxis](#) ()
通电
- int [poweroffExtAxis](#) ()
断电
- int [enableExtAxis](#) ()
使能
- int [setExtAxisMountingPose](#) (const std::vector< double > &pose)
设置外部轴的安装位姿 (相对于世界坐标系)
- int [moveExtJoint](#) (double pos, double v, double a, double duration)
运动到指定点, 旋转或者平移
- int [speedExtJoint](#) (double v, double a, double duration)
制定目标运动速度
- int [stopExtJoint](#) (double a)
停止外部轴运动
- int [getExtAxisType](#) ()
获取外部轴的类型 0 代表是旋转 1 代表平移
- int [setExtAxisType](#) (int type)
设置外部轴的类型
- [AxisModeType](#) [getAxisModeType](#) ()
获取当前外部轴的状态
- std::vector< double > [getExtAxisMountingPose](#) ()
获取外部轴安装位姿
- std::vector< double > [getExtAxisPose](#) ()

- 获取相对于安装坐标系的位姿，外部轴可能为变位机或者导轨
- double `getExtAxisPosition` ()
获取外部轴位置
 - double `getExtAxisVelocity` ()
获取外部轴运行速度
 - int `setExtAxisVelocity` (double velocity)
设置外部轴运行速度
 - double `getExtAxisAcceleration` ()
获取外部轴运行加速度
 - int `setExtAxisAcceleration` (double acc)
设置外部轴运行加速度
 - double `getExtAxisCurrent` ()
获取外部轴电流
 - double `getExtAxisTemperature` ()
获取外部轴温度
 - double `getExtAxisBusVoltage` ()
获取外部轴电压
 - double `getExtAxisBusCurrent` ()
获取外部轴电流
 - double `getExtAxisMaxPosition` ()
获取外部轴最大位置 (物理极限)
 - int `setExtAxisMaxPositionLimit` (double q)
设置外部轴最大位置 (软件/算法限制)
 - double `getExtAxisMaxPositionLimit` ()
获取外部轴最大位置 (软件/算法限制)
 - double `getExtMinPosition` ()
获取外部轴最小位置 (物理极限)
 - int `setExtMinPositionLimit` (double q)
设置外部轴最小位置 (软件/算法限制)
 - double `getExtMinPositionLimit` ()
获取外部轴最小位置 (软件/算法限制)
 - double `getExtAxisMaxVelocity` ()
获取外部轴最大速度 (物理极限)
 - int `setExtAxisMaxVelocityLimit` (double v)
设置外部轴最大速度 (软件/算法限制)
 - double `getExtAxisMaxVelocityLimit` ()
获取外部轴最大速度 (软件/算法限制)
 - double `getExtAxisMaxAcceleration` ()
获取外部轴最大加速度 (物理极限)
 - int `setExtAxisMaxAccelerationLimit` (double acc)
设置外部轴最大加速度 (软件/算法限制)
 - double `getExtAxisMaxAccelerationLimit` ()
获取外部轴最大加速度 (软件/算法限制)
 - int `followAnotherAxis` (const std::string &target_name, double phase, double err)
跟踪另一个外部轴的运动 (禁止运动过程中使用)
 - int `stopFollowAnotherAxis` ()
stopFollowAnotherAxis(禁止运动过程中使用)
 - int `getErrorCode` ()
getErrorCode(获取外部轴错误码)
 - int `clearAxisError` ()
重置外部轴错误

- int `setExtAxisZero` ()
将当前外部轴位置设置为零位
- int `setReductionRatio` (double ratio)
设置外部轴减速比
- double `getReductionRatio` ()
获取外部轴减速比

Protected 属性

- void * `d_`

11.3.1 详细描述

在文件 `axis_interface.h` 第 18 行定义.

11.3.2 构造及析构函数说明

AxisInterface()

```
arcs::common_interface::AxisInterface::AxisInterface ()
```

~AxisInterface()

```
virtual arcs::common_interface::AxisInterface::~~AxisInterface () [virtual]
```

11.3.3 类成员变量说明

d_

```
void* arcs::common_interface::AxisInterface::d_ [protected]
```

在文件 `axis_interface.h` 第 637 行定义.

该类的文档由以下文件生成:

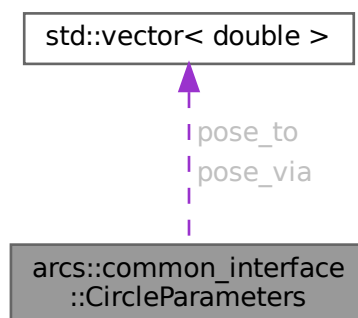
- include/aubo/`axis_interface.h`

11.4 arcs::common_interface::CircleParameters 结构体参考

圆周运动参数定义

```
#include <type_def.h>
```

arcs::common_interface::CircleParameters 的协作图:



Public 属性

- `std::vector< double > pose_via`
圆周运动途中点的位姿
- `std::vector< double > pose_to`
圆周运动结束点的位姿
- `double a`
加速度, 单位: m/s^2
- `double v`
速度, 单位: m/s
- `double blend_radius`
交融半径, 单位: m
- `double duration`
运行时间, 单位: s
- `double helix`
- `double spiral`
- `double direction`
- `int loop_times`
暂不支持

11.4.1 详细描述

圆周运动参数定义

在文件 `type_def.h` 第 746 行定义.

11.4.2 类成员变量说明

a

```
double arcs::common_interface::CircleParameters::a
```

加速度, 单位: m/s^2

在文件 `type_def.h` 第 750 行定义.

blend_radius

```
double arcs::common_interface::CircleParameters::blend_radius
```

交融半径, 单位: m

在文件 `type_def.h` 第 752 行定义.

direction

```
double arcs::common_interface::CircleParameters::direction
```

在文件 `type_def.h` 第 756 行定义.

duration

```
double arcs::common_interface::CircleParameters::duration
```

运行时间, 单位: s

在文件 `type_def.h` 第 753 行定义.

helix

```
double arcs::common_interface::CircleParameters::helix
```

在文件 `type_def.h` 第 754 行定义.

loop_times

```
int arcs::common_interface::CircleParameters::loop_times
```

暂不支持

在文件 `type_def.h` 第 757 行定义.

pose_to

```
std::vector<double> arcs::common_interface::CircleParameters::pose_to
```

圆周运动结束点的位姿

在文件 [type_def.h](#) 第 749 行定义.

pose_via

```
std::vector<double> arcs::common_interface::CircleParameters::pose_via
```

圆周运动途中点的位姿

在文件 [type_def.h](#) 第 748 行定义.

spiral

```
double arcs::common_interface::CircleParameters::spiral
```

在文件 [type_def.h](#) 第 755 行定义.

v

```
double arcs::common_interface::CircleParameters::v
```

速度, 单位: m/s

在文件 [type_def.h](#) 第 751 行定义.

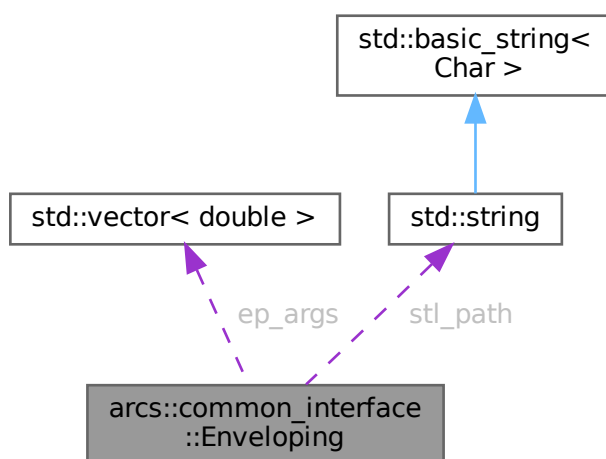
该结构体的文档由以下文件生成:

- [include/aubo/type_def.h](#)

11.5 arcs::common_interface::Enveloping 结构体参考

```
#include <type_def.h>
```

arcs::common_interface::Enveloping 的协作图:

**Public 属性**

- [EnvelopingShape shape](#)
- `std::vector<double>` [ep_args](#)
- `std::string` [stl_path](#)

11.5.1 详细描述

在文件 `type_def.h` 第 779 行定义.

11.5.2 类成员变量说明

ep_args

`std::vector<double> arcs::common_interface::Enveloping::ep_args`
在文件 `type_def.h` 第 782 行定义.

shape

`EnvelopingShape arcs::common_interface::Enveloping::shape`
在文件 `type_def.h` 第 781 行定义.

stl_path

`std::string arcs::common_interface::Enveloping::stl_path`
在文件 `type_def.h` 第 785 行定义.
该结构体的文档由以下文件生成:

- `include/aubo/type_def.h`

11.6 arcs::common_interface::ForceControl 类参考

```
#include <force_control.h>
```

Public 成员函数

- `ForceControl ()`
- `virtual ~ForceControl ()`
- `int fcEnable ()`
使能力控。`fcEnable` 被用于使能力控。在力控被使能的同时，`fcEnable` 用于定义力控的坐标系，并调整力和力矩的阻尼。如果在 `fcEnable` 中未指定坐标系，则会创建一个默认的力控制坐标系，其方向与工作对象坐标系相同。所有力控制监管功能都被 `fcEnable` 激活。
- `int fcDisable ()`
失能力控。`fcDisable` 被用于失能力控。在成功失能力控之后，机器人将回到位置控制模式。
- `bool isFcEnabled ()`
判断力控是否被使能
- `int setTargetForce (const std::vector< double > &feature, const std::vector< bool > &compliance, const std::vector< double > &wrench, const std::vector< double > &limits, TaskFrameType type=TaskFrameType::FRAME_FORCE)`
设置力控参考 (目标) 值
- `int setDynamicModel1 (const std::vector< double > &env_stiff, const std::vector< double > &damp←_scale, const std::vector< double > &stiff_scale)`
设置力控动力学模型
- `DynamicsModel fcCalDynamicModel (const std::vector< double > &env_stiff, const std::vector< double > &damp_scale, const std::vector< double > &stiff_scale)`
计算力控动力学模型
- `int setDynamicModelSearch (const std::vector< double > &damp_scale, const std::vector< double > &stiff_scale)`
设置力控搜孔场景下的动力学模型
- `int setDynamicModelInsert (const std::vector< double > &damp_scale, const std::vector< double > &stiff_scale)`
设置力控插/拔孔场景下的动力学模型
- `int setDynamicModelContact (const std::vector< double > &env_stiff, const std::vector< double > &damp_scale, const std::vector< double > &stiff_scale)`

- 设置力控接触场景下的动力学模型

 - int [setDynamicModel](#) (const std::vector< double > &m, const std::vector< double > &d, const std::vector< double > &k)

设置力控动力学模型
- int [fcSetSensorThresholds](#) (const std::vector< double > &thresholds)

设置力控阈值
- int [fcSetSensorLimits](#) (const std::vector< double > &limits)

设置力控最大受力限制
- std::vector< double > [getFcSensorThresholds](#) ()

获取力控阈值
- std::vector< double > [getFcSensorLimits](#) ()

获取最大力限制
- [DynamicsModel](#) [getDynamicModel](#) ()

获取力控动力学模型
- int [setCondForce](#) (const std::vector< double > &min, const std::vector< double > &max, bool outside, double timeout)

设置力控终止条件: 力, 当测量的力在设置的范围之内, 力控算法将保持运行, 直到设置的条件不满足, 力控将退出
- int [setCondOrient](#) (const std::vector< double > &frame, double max_angle, double max_rot, bool outside, double timeout)

设置力控终止条件: 姿态, 当测量的姿态在设置的范围之内, 力控算法将保持运行, 直到设置的条件不满足, 力控将退出.
- int [setCondPlane](#) (const std::vector< double > &plane, double timeout)

指定力控有效平面, x - y 平面, z 方向有效
- int [setCondCylinder](#) (const std::vector< double > &axis, double radius, bool outside, double timeout)

指定力控有效圆柱体, 提供中心轴和圆柱半径, 可以指定圆柱内部还是外部
- int [setCondSphere](#) (const std::vector< double > &center, double radius, bool outside, double timeout)

指定力控有效球体, 提供球心和半径, 可以指定球体内部还是外部
- int [setCondTcpSpeed](#) (const std::vector< double > &min, const std::vector< double > &max, bool outside, double timeout)

设置 TCP 速度的终止条件。该条件可通过调用 *FCCondWaitWhile* 指令激活, 在指定条件为真时, 程序将等待并保持执行。这样可以使参考力、力矩和运动继续, 直到速度超出指定范围。
- int [setCondDistance](#) (double distance, double timeout)

力控终止条件-距离
- int [setCondAdvanced](#) (const std::string &type, const std::vector< double > &args, double timeout)

高级力控终止条件
- int [setCondActive](#) ()

激活力控终止条件
- bool [isCondFullfiled](#) ()

力控终止条件是否已经满足
- int [setSupvForce](#) (const std::vector< double > &min, const std::vector< double > &max)

setSupvForce 用于在力控中设置力监督。监督在通过 *FCAct* 指令激活力控时被激活。
- int [setSupvOrient](#) (const std::vector< double > &frame, double max_angle, double max_rot, bool outside)

setSupvOrient 用于设置工具姿态的监督条件。当通过 *FCAct* 指令激活力控时, 监督条件被激活。
- int [setSupvPosBox](#) (const std::vector< double > &frame, const [Box](#) &box)

setSupvPosBox 用于在力控中设置位置监督。监督在通过 *FCAct* 指令激活力控时被激活。位置监督通过为 TCP 定义空间体积来设置。一旦激活, 如果 TCP 超出该体积, 监督将停止执行。
- int [setSupvPosCylinder](#) (const std::vector< double > &frame, const [Cylinder](#) &cylinder)
- int [setSupvPosSphere](#) (const std::vector< double > &frame, const [Sphere](#) &sphere)
- int [setSupvReoriSpeed](#) (const std::vector< double > &speed_limit, bool outside, double timeout)

- setSupvReoriSpeed* 用于在力控中设置重新定向速度监督。监督在通过 *FCAct* 指令激活力控时被激活。
- int *setSupvTcpSpeed* (const std::vector< double > &speed_limit, bool outside, double timeout)
setSupvTcpSpeed 用于在力控中设置 *TCP* 速度监督。监督在通过 *FCAct* 指令激活力控时被激活。*TCP* 速度监督通过定义工作对象坐标系各方向上的最小和最大速度限制来设置。一旦激活, 如果检测到过高的 *TCP* 速度值, 监督将停止执行。
- int *setLpFilter* (const std::vector< double > &cutoff_freq)
 设置低通滤波器
- int *resetLpFilter* ()
 重置低通滤波器
- int *speedChangeEnable* (double ref_force)
speedChangeEnable 用于激活 *FC SpeedChange* 功能, 并设置期望的参考力和恢复行为。当 *FC SpeedChange* 功能被激活时, 机器人速度会根据测量信号与参考值的接近程度自动降低或提高。
- int *speedChangeDisable* ()
 停用 *FC SpeedChange* 功能。
- int *speedChangeTune* (int speed_levels, double speed_ratio_min)
speedChangeTune 用于将 *FC SpeedChange* 系统参数设置为新值。
- int *setDamping* (const std::vector< double > &damping, double ramp_time)
setDamping 用于在力控坐标系中调整阻尼。可调参数包括扭矩 *x* 方向到扭矩 *z* 方向的阻尼 (见第 255 页) 以及力 *x* 方向到力 *z* 方向的阻尼 (见第 254 页)。
- int *resetDamping* ()
 重置阻尼参数
- int *softFloatEnable* ()
 启用软浮动功能。
- int *softFloatDisable* ()
 停用软浮动功能。
- bool *isSoftFloatEnabled* ()
 返回是否开启了软浮动
- int *setSoftFloatParams* (bool joint_space, const std::vector< bool > &select, const std::vector< double > &stiff_percent, const std::vector< double > &stiff_damp_ratio, const std::vector< double > &force_threshold, const std::vector< double > &force_limit)
 设置软浮动参数
- int *toolContact* (const std::vector< bool > &direction)
 检测工具和外部物体的接触
- std::vector< double > *getActualJointPositionsHistory* (int steps)

Protected 属性

- void * *d_* { nullptr }

11.6.1 详细描述

在文件 *force_control.h* 第 94 行定义.

11.6.2 构造及析构函数说明

ForceControl()

```
arcs::common_interface::ForceControl::ForceControl ()
```

~ForceControl()

```
virtual arcs::common_interface::ForceControl::~~ForceControl () [virtual]
```


11.6.3 类成员变量说明

d_

```
void* arcs::common_interface::ForceControl::d_ { nullptr } [protected]
```

在文件 [force_control.h](#) 第 2616 行定义。

该类的文档由以下文件生成:

- [include/aubo/robot/force_control.h](#)

11.7 arcs::common_interface::GripperInterface 类参考

```
#include <gripper_interface.h>
```

Public 成员函数

- [GripperInterface \(\)](#)
- virtual [~GripperInterface \(\)](#)
- [std::vector< std::string > gripperGetSupportedModels \(\)](#)
获取支持的所有夹爪
- [ResultWithErrno2 gripperScanDevices \(const std::string &model, const std::string &device_name\)](#)
扫描该型号下的所有设备
- [std::vector< std::string > gripperGetNames \(\)](#)
获取已添加的夹爪
- [int gripperAdd \(const std::string &name, const std::string &model\)](#)
添加夹爪
- [int gripperDelete \(const std::string &name\)](#)
删除夹爪
- [int gripperRename \(const std::string &name, const std::string &new_name\)](#)
修改夹爪名
- [int gripperConnect \(const std::string &name, const std::string &device_name\)](#)
夹爪连接
- [int gripperDisconnect \(const std::string &name\)](#)
夹爪断开连接
- [bool gripperIsConnected \(const std::string &name\)](#)
夹爪是否连接
- [int gripperSetWorkMode \(const std::string &name, int work_mode\)](#)
设置工作模式
- [int gripperGetWorkMode \(const std::string &name\)](#)
获取工作模式
- [int gripperSetMountPose \(const std::string &name, const std::vector< double > &pose, bool enable_collision\)](#)
设置夹爪安装偏移
- [std::vector< double > gripperGetMountPose \(const std::string &name\)](#)
获取夹爪安装偏移
- [int gripperEnable \(const std::string &name, bool enable\)](#)
使能夹爪
- [bool gripperIsEnabled \(const std::string &name\)](#)
夹爪是否使能
- [int gripperSetPosition \(const std::string &name, const double position\)](#)
设置运动参数
- [int gripperSetVelocity \(const std::string &name, const double velocity_percent\)](#)
- [int gripperSetForce \(const std::string &name, const double force\)](#)
- [int gripperSetAngle \(const std::string &name, const double angle\)](#)

- int `gripperSetRVelocity` (const std::string &name, const double r_velocity_percent)
- int `gripperSetTorque` (const std::string &name, const double torque_percent)
- int `gripperMove` (const std::string &name)
开始运动
- int `gripperStop` (const std::string &name)
停止运动
- std::string `gripperGetHardwareVersion` (const std::string &name)
获取夹爪状态
- std::string `gripperGetSoftwareVersion` (const std::string &name)
- double `gripperGetPosition` (const std::string &name)
- double `gripperGetVelocity` (const std::string &name)
- double `gripperGetForce` (const std::string &name)
- double `gripperGetAngle` (const std::string &name)
- double `gripperGetRVelocity` (const std::string &name)
- double `gripperGetTorque` (const std::string &name)
- bool `gripperGetObjectDetection` (const std::string &name)
- bool `gripperGetMotionState` (const std::string &name)
- double `gripperGetVoltage` (const std::string &name)
- double `gripperGetTemperature` (const std::string &name)
- int `gripperResetSlaveId` (const std::string &name, const int slave_id)
重置 *modbus* 从站ID
- int `gripperGetStatusCode` (const std::string &name)
获取夹爪状态

Protected 属性

- void * `d_`

11.7.1 详细描述

在文件 `gripper_interface.h` 第 17 行定义.

11.7.2 构造及析构函数说明

GripperInterface()

```
arcs::common_interface::GripperInterface::GripperInterface ()
```

~GripperInterface()

```
virtual arcs::common_interface::GripperInterface::~~GripperInterface () [virtual]
```

11.7.3 成员函数说明

gripperGetAngle()

```
double arcs::common_interface::GripperInterface::gripperGetAngle (
    const std::string & name)
```

gripperGetForce()

```
double arcs::common_interface::GripperInterface::gripperGetForce (
    const std::string & name)
```

gripperGetMotionState()

```
bool arcs::common_interface::GripperInterface::gripperGetMotionState (
    const std::string & name)
```

gripperGetMountPose()

```
std::vector< double > arcs::common_interface::GripperInterface::gripperGetMountPose (
    const std::string & name)
    获取夹爪安装偏移
```

参数

<i>name</i>	

返回

返回夹爪安装偏移

gripperGetObjectDetection()

```
bool arcs::common_interface::GripperInterface::gripperGetObjectDetection (
    const std::string & name)
```

gripperGetPosition()

```
double arcs::common_interface::GripperInterface::gripperGetPosition (
    const std::string & name)
```

gripperGetRVelocity()

```
double arcs::common_interface::GripperInterface::gripperGetRVelocity (
    const std::string & name)
```

gripperGetSoftwareVersion()

```
std::string arcs::common_interface::GripperInterface::gripperGetSoftwareVersion (
    const std::string & name)
```

gripperGetTemperature()

```
double arcs::common_interface::GripperInterface::gripperGetTemperature (
    const std::string & name)
```

gripperGetTorque()

```
double arcs::common_interface::GripperInterface::gripperGetTorque (
    const std::string & name)
```

gripperGetVelocity()

```
double arcs::common_interface::GripperInterface::gripperGetVelocity (
    const std::string & name)
```

gripperGetVoltage()

```
double arcs::common_interface::GripperInterface::gripperGetVoltage (
    const std::string & name)
```

gripperSetAngle()

```
int arcs::common_interface::GripperInterface::gripperSetAngle (
    const std::string & name,
    const double angle)
```

gripperSetForce()

```
int arcs::common_interface::GripperInterface::gripperSetForce (
    const std::string & name,
    const double force)
```

gripperSetRVelocity()

```
int arcs::common_interface::GripperInterface::gripperSetRVelocity (
    const std::string & name,
    const double r_velocity_percent)
```

gripperSetTorque()

```
int arcs::common_interface::GripperInterface::gripperSetTorque (
    const std::string & name,
    const double torque_percent)
```

gripperSetVelocity()

```
int arcs::common_interface::GripperInterface::gripperSetVelocity (
    const std::string & name,
    const double velocity_percent)
```

11.7.4 类成员变量说明**d_**

```
void* arcs::common_interface::GripperInterface::d_ [protected]
```

在文件 [gripper_interface.h](#) 第 285 行定义。

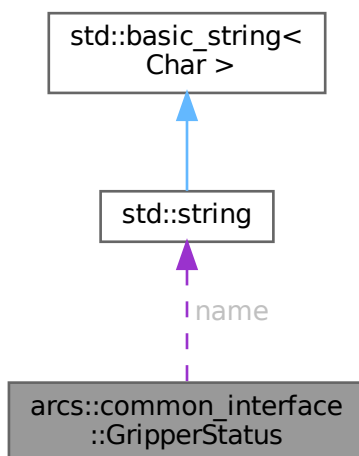
该类的文档由以下文件生成:

- [include/aubo/gripper_interface.h](#)

11.8 arcs::common_interface::GripperStatus 结构体参考

```
#include <type_def.h>
```

arcs::common_interface::GripperStatus 的协作图:



Public 属性

- std::string name
- bool is_connected
- bool is_enabled
- double position
- double velocity
- double force
- double angle
- double r_velocity
- double torque
- bool object_detection
- bool motion_state
- double voltage
- double temperature
- int status_code

11.8.1 详细描述

在文件 [type_def.h](#) 第 874 行定义.

11.8.2 类成员变量说明

angle

double arcs::common_interface::GripperStatus::angle
在文件 [type_def.h](#) 第 882 行定义.

force

double arcs::common_interface::GripperStatus::force
在文件 [type_def.h](#) 第 881 行定义.

is_connected

bool arcs::common_interface::GripperStatus::is_connected
在文件 [type_def.h](#) 第 877 行定义.

is_enabled

bool arcs::common_interface::GripperStatus::is_enabled
在文件 [type_def.h](#) 第 878 行定义.

motion_state

bool arcs::common_interface::GripperStatus::motion_state
在文件 [type_def.h](#) 第 886 行定义.

name

std::string arcs::common_interface::GripperStatus::name
在文件 [type_def.h](#) 第 876 行定义.

object_detection

bool arcs::common_interface::GripperStatus::object_detection
在文件 [type_def.h](#) 第 885 行定义.

position

double arcs::common_interface::GripperStatus::position
在文件 [type_def.h](#) 第 879 行定义.

r_velocity

double arcs::common_interface::GripperStatus::r_velocity
在文件 [type_def.h](#) 第 883 行定义.

status_code

int arcs::common_interface::GripperStatus::status_code
在文件 [type_def.h](#) 第 889 行定义.

temperature

double arcs::common_interface::GripperStatus::temperature
在文件 [type_def.h](#) 第 888 行定义.

torque

double arcs::common_interface::GripperStatus::torque
在文件 [type_def.h](#) 第 884 行定义.

velocity

double arcs::common_interface::GripperStatus::velocity
在文件 [type_def.h](#) 第 880 行定义.

voltage

double arcs::common_interface::GripperStatus::voltage
在文件 [type_def.h](#) 第 887 行定义.
该结构体的文档由以下文件生成:

- [include/aubo/type_def.h](#)

11.9 arcs::common_interface::IoControl 类参考

```
#include <io_control.h>
```

Public 成员函数

- [IoControl](#) ()
- virtual [~IoControl](#) ()
- int [getStandardDigitalInputNum](#) ()
获取标准数字输入数量
- int [getToolDigitalInputNum](#) ()
获取工具端数字IO 数量 (包括数字输入和数字输出)
- int [getConfigurableDigitalInputNum](#) ()
获取可配置数字输入数量
- int [getStandardDigitalOutputNum](#) ()
获取标准数字输出数量
- int [getToolDigitalOutputNum](#) ()
获取工具端数字IO 数量 (包括数字输入和数字输出)
- int [setToolIoInput](#) (int index, bool input)
设置指定的工具端数字IO 为输入或输出
- bool [isToolIoInput](#) (int index)

- 判断指定的工具端数字IO 类型是否为输入
- int [getConfigurableDigitalOutputNum](#) ()
获取可配置数字输出数量
- int [getStandardAnalogInputNum](#) ()
获取标准模拟输入数量
- int [getToolAnalogInputNum](#) ()
获取工具端模拟输入数量
- int [getStandardAnalogOutputNum](#) ()
获取标准模拟输出数量
- int [getToolAnalogOutputNum](#) ()
获取工具端模拟输出数量
- int [setDigitalInputActionDefault](#) ()
设置所有数字输入动作为无触发
- int [setStandardDigitalInputAction](#) (int index, [StandardInputAction](#) action)
设置标准数字输入触发动作
- int [setToolDigitalInputAction](#) (int index, [StandardInputAction](#) action)
设置工具数字输入触发动作
- int [setConfigurableDigitalInputAction](#) (int index, [StandardInputAction](#) action)
设置可配置数字输入触发动作
- [StandardInputAction](#) [getStandardDigitalInputAction](#) (int index)
获取标准数字输入触发动作
- [StandardInputAction](#) [getToolDigitalInputAction](#) (int index)
获取工具端数字输入触发动作
- [StandardInputAction](#) [getConfigurableDigitalInputAction](#) (int index)
获取可配置数字输入的输入触发动作
- int [setDigitalOutputRunstateDefault](#) ()
设置所有数字输出状态选择为无
- int [setStandardDigitalOutputRunstate](#) (int index, [StandardOutputRunState](#) runstate)
设置标准数字输出状态选择
- int [setToolDigitalOutputRunstate](#) (int index, [StandardOutputRunState](#) runstate)
设置工具端数字输出状态选择
- int [setConfigurableDigitalOutputRunstate](#) (int index, [StandardOutputRunState](#) runstate)
设置可配置数字输出状态选择
- int [setDigitalOutputRunstate](#) (int io_type, int index, [StandardOutputRunState](#) runstate, const std::string &object_name, double threshold)
设置数字输出的状态选择, 并支持基于路点/平面/障碍物的条件阈值判断。
- [StandardOutputRunState](#) [getStandardDigitalOutputRunstate](#) (int index)
获取标准数字输出状态选择
- [StandardOutputRunState](#) [getToolDigitalOutputRunstate](#) (int index)
获取工具端数字输出状态选择
- [StandardOutputRunState](#) [getConfigurableDigitalOutputRunstate](#) (int index)
获取可配置数字输出状态选择
- [StandardOutputRunState](#) [getDigitalOutputRunstate](#) (int io_type, int index)
获取指定类型数字输出状态选择。
- int [addCrossConnection](#) (const std::string &id, int operator_type, int result_type, int result_pin, const std::vector< int > &actor_types, const std::vector< int > &actor_pins, const std::vector< bool > &actor_inverts)
新增一条 *DIO Cross Connection* 逻辑规则。
- int [deleteCrossConnection](#) (const std::string &id)
删除一条 *DIO Cross Connection* 逻辑规则。
- int [clearCrossConnections](#) ()

- 清空所有 *DIO Cross Connection* 逻辑规则。
- `int setStandardAnalogOutputRunstate (int index, StandardOutputRunState runstate)`
设置标准模拟输出状态选择
 - `int setToolAnalogOutputRunstate (int index, StandardOutputRunState runstate)`
设置工具端模拟输出状态选择
 - `StandardOutputRunState getStandardAnalogOutputRunstate (int index)`
获取标准模拟输出状态选择
 - `StandardOutputRunState getToolAnalogOutputRunstate (int index)`
获取工具端模拟输出状态选择
 - `int setDigitalOutputAfterEStopDefault ()`
设置所有数字输出急停后状态为默认 (不做改变)
 - `int setAnalogOutputAfterEStopDefault ()`
设置所有模拟输出急停后状态为默认 (不做改变)
 - `int setStandardDigitalOutputAfterEStop (int index, bool value)`
设置标准数字输出急停后的输出值
 - `int setConfigurableDigitalOutputAfterEStop (int index, bool value)`
设置可配置数字输出急停后的输出值
 - `int setStandardAnalogOutputAfterEStop (int index, double value)`
设置标准模拟输出急停后的输出值
 - `int setStandardAnalogInputDomain (int index, int domain)`
设置标准模拟输入的范围
 - `int setToolAnalogInputDomain (int index, int domain)`
设置工具端模拟输入的范围
 - `int getStandardAnalogInputDomain (int index)`
获取标准模式输入范围
 - `int getToolAnalogInputDomain (int index)`
获取工具端模式输入范围
 - `int setStandardAnalogOutputDomain (int index, int domain)`
设置标准模拟输出的范围
 - `int setToolAnalogOutputDomain (int index, int domain)`
设置工具端模拟输出范围
 - `int getStandardAnalogOutputDomain (int index)`
获取标准模拟输出范围
 - `int getToolAnalogOutputDomain (int index)`
获取工具端模拟输出范围
 - `int setToolVoltageOutputDomain (int domain)`
设置工具端电源电压值 (单位 V)
 - `int getToolVoltageOutputDomain ()`
获取工具端电源电压值 (单位 V)
 - `int setStandardDigitalOutput (int index, bool value)`
设置标准数字输出值
 - `int addGroupOutput (const std::string &name, int io_type, const std::vector< int > &indexes)`
注册组数字输出
 - `int deleteGroupOutput (const std::string &name)`
删除组数字输出
 - `int setGroupOutput (const std::string &name, uint32_t value)`
设置组数字输出值
 - `int setStandardDigitalOutputPulse (int index, bool value, double duration)`
设置数字输出脉冲
 - `int setToolDigitalOutput (int index, bool value)`
设置工具端数字输出值

- int [setToolDigitalOutputPulse](#) (int index, bool value, double duration)
设置工具端数字输出脉冲
- int [setConfigurableDigitalOutput](#) (int index, bool value)
设置可配置数字输出值
- int [setConfigurableDigitalOutputPulse](#) (int index, bool value, double duration)
设置可配置数字输出脉冲
- int [setStandardAnalogOutput](#) (int index, double value)
设置标准模拟输出值
- int [setToolAnalogOutput](#) (int index, double value)
设置工具端模拟输出值
- bool [getStandardDigitalInput](#) (int index)
获取标准数字输入值
- uint32_t [getStandardDigitalInputs](#) ()
获取所有的标准数字输入值
- int [addGroupInput](#) (const std::string &name, int io_type, const std::vector< int > &indexes)
注册组数字输入
- int [deleteGroupInput](#) (const std::string &name)
删除组数字输入
- uint32_t [getGroupInput](#) (const std::string &name)
获取组数字输入值
- bool [getToolDigitalInput](#) (int index)
获取工具端数字输入值
- uint32_t [getToolDigitalInputs](#) ()
获取所有的工具端数字输入值
- bool [getConfigurableDigitalInput](#) (int index)
获取可配置数字输入值
- uint32_t [getConfigurableDigitalInputs](#) ()
获取所有的可配置数字输入值
- bool [getStandardDigitalOutput](#) (int index)
获取标准数字输出值
- uint32_t [getStandardDigitalOutputs](#) ()
获取所有的标准数字输出值
- uint32_t [getGroupOutput](#) (const std::string &name)
获取组数字输出值
- bool [getToolDigitalOutput](#) (int index)
获取工具端数字输出值
- uint32_t [getToolDigitalOutputs](#) ()
获取所有的工具端数字输出值
- bool [getConfigurableDigitalOutput](#) (int index)
获取可配置数字输出值
- uint32_t [getConfigurableDigitalOutputs](#) ()
获取所有的可配置数字输出值
- double [getStandardAnalogInput](#) (int index)
获取标准模拟输入值
- double [getToolAnalogInput](#) (int index)
获取工具端模拟输入值
- double [getStandardAnalogOutput](#) (int index)
获取标准模拟输出值
- double [getToolAnalogOutput](#) (int index)
获取工具端模拟输出值
- int [getStaticLinkInputNum](#) ()

- 获取联动输入数量
- `int getStaticLinkOutputNum ()`
获取联动输出数量
- `uint32_t getStaticLinkInputs ()`
获取所有的联动输入值
- `uint32_t getStaticLinkOutputs ()`
获取所有的联动输出值
- `bool hasEncoderSensor ()`
机器人是否配置了编码器集成编码器的编号为 0
- `int setEncDecoderType (int type, int range_id)`
设置集成编码器的解码方式
- `int setEncTickCount (int tick)`
设置集成编码器脉冲数
- `int getEncDecoderType ()`
获取编码器的解码方式
- `int getEncTickCount ()`
获取脉冲数
- `int unwindEncDeltaTickCount (int delta_count)`
防止在计数超出范围时计数错误
- `bool getToolButtonStatus ()`
获取末端按钮状态
- `uint32_t getHandleIoStatus ()`
获取手柄按键状态
- `int getHandleType ()`
获取手柄类型

Protected 属性

- `void * d_`

11.9.1 详细描述

在文件 `io_control.h` 第 47 行定义.

11.9.2 构造及析构造函数说明

IoControl()

```
arcs::common_interface::IoControl::IoControl ()
```

~IoControl()

```
virtual arcs::common_interface::IoControl::~~IoControl () [virtual]
```

11.9.3 类成员变量说明

d_

```
void* arcs::common_interface::IoControl::d_ [protected]
```

在文件 `io_control.h` 第 5772 行定义.

该类的文档由以下文件生成:

- `include/aubo/robot/io_control.h`

11.10 arcs::common_interface::Math 类参考

```
#include <math.h>
```

Public 成员函数

- [Math](#) ()
- virtual [~Math](#) ()
- `std::vector< double > poseAdd` (const `std::vector< double >` &p1, const `std::vector< double >` &p2)
- `std::vector< double > poseSub` (const `std::vector< double >` &p1, const `std::vector< double >` &p2)
位姿相减
- `std::vector< double > interpolatePose` (const `std::vector< double >` &p1, const `std::vector< double >` &p2, double alpha)
计算线性插值
- `std::vector< double > poseTrans` (const `std::vector< double >` &pose_from, const `std::vector< double >` &pose_from_to)
- `std::vector< double > poseTransInv` (const `std::vector< double >` &pose_from, const `std::vector< double >` &pose_to_from)
姿态变换
姿态逆变换
- `std::vector< double > poseInverse` (const `std::vector< double >` &pose)
获取位姿的逆
- `double poseDistance` (const `std::vector< double >` &p1, const `std::vector< double >` &p2)
计算两个位姿的位置距离
- `double poseAngleDistance` (const `std::vector< double >` &p1, const `std::vector< double >` &p2)
计算两个位姿的轴角距离
- `bool poseEqual` (const `std::vector< double >` &p1, const `std::vector< double >` &p2, double eps=5e-5)
判断两个位姿是否相等
- `std::vector< double > transferRefFrame` (const `std::vector< double >` &F_b_a_old, const [Vector3d](#) &V_in_a, int type)
- `std::vector< double > poseRotation` (const `std::vector< double >` &pose, const `std::vector< double >` &rotrv)
姿态旋转
- `std::vector< double > rpyToQuaternion` (const `std::vector< double >` &rpy)
欧拉角转四元数
- `std::vector< double > quaternionToRpy` (const `std::vector< double >` &quat)
四元数转欧拉角
- [ResultWithErrno tcpOffsetIdentify](#) (const `std::vector< std::vector< double > >` &poses)
四点法标定 TCP 偏移
- [ResultWithErrno calibrateCoordinate](#) (const `std::vector< std::vector< double > >` &poses, int type)
三点法标定坐标系
- [ResultWithErrno calculateCircleFourthPoint](#) (const `std::vector< double >` &p1, const `std::vector< double >` &p2, const `std::vector< double >` &p3, int mode)
根据圆弧的三个点, 计算出拟合成的圆的另一半圆弧的中间点位置
- `std::vector< double > forceTrans` (const `std::vector< double >` &pose_a_in_b, const `std::vector< double >` &force_in_a)
- `std::vector< double > getDeltaPoseBySensorDistance` (const `std::vector< double >` &distances, double position, double radius, double track_scale)
- `std::vector< double > deltaPoseTrans` (const `std::vector< double >` &pose_a_in_b, const `std::vector< double >` &ft_in_a)
- `std::vector< double > deltaPoseAdd` (const `std::vector< double >` &pose_a_in_b, const `std::vector< double >` &v_in_b)
- `std::vector< double > changePoseWithXYRef` (const `std::vector< double >` &pose_tar, const `std::vector< double >` &pose_ref)
- `std::vector< double > homMatrixToPose` (const `std::vector< double >` &homMatrix)
- `std::vector< double > poseToHomMatrix` (const `std::vector< double >` &pose)

Protected 属性

- void * [d_](#)

11.10.1 详细描述

在文件 [math.h](#) 第 21 行定义.

11.10.2 构造及析构函数说明

Math()

```
arcs::common_interface::Math::Math ()
```

~Math()

```
virtual arcs::common_interface::Math::~Math () [virtual]
```

11.10.3 类成员变量说明

[d_](#)

```
void* arcs::common_interface::Math::d_ [protected]
```

在文件 [math.h](#) 第 1071 行定义.

该类的文档由以下文件生成:

- include/aubo/[math.h](#)

11.11 arcs::common_interface::MotionControl 类参考

```
#include <motion_control.h>
```

Public 成员函数

- [MotionControl](#) ()
- virtual [~MotionControl](#) ()
- double [getEqradius](#) ()

获取等效半径, 单位 *m* [moveLine](#)/[moveCircle](#) 时, 末端姿态旋转的角度等效到末端位置移动可以通过 [setEqradius](#) 设置, 默认为 1
- int [setEqradius](#) (double eqradius)

设置等效半径, 单位 *m* [moveLine](#)/[moveCircle](#) 时, 末端姿态旋转的角度等效到末端位置移动, 数值越大, 姿态旋转速度越快
- int [setSpeedFraction](#) (double fraction)

动态调整机器人运行速度和加速度比例 (0., 1.
- double [getSpeedFraction](#) ()

获取速度和加速度比例, 默认为 1 可以通过 [setSpeedFraction](#) 接口设置
- int [speedFractionCritical](#) (bool enable)

速度比例设置临界区, 使能之后速度比例被强制设定为 1.
- int [speedFractionCritical1](#) (bool enable)

速度比例设置临界区, 使能之后速度比例被强制设定为当前生效的比例, 失能之后恢复之前的速度比例
- bool [isSpeedFractionCritical](#) ()

是否处于速度比例设置临界区
- bool [isBlending](#) ()

是否处交融区
- int [pathOffsetLimits](#) (double v, double a)

设置偏移的最大速度和最大加速度仅对 [pathOffsetSet](#) 中 *type=1* 有效
- int [pathOffsetCoordinate](#) (int ref_coord)

设置偏移的参考坐标系仅对 [pathOffsetSet](#) 中 *type=1* 有效

- int [pathOffsetEnable](#) ()
路径偏移使能
- int [pathOffsetSet](#) (const std::vector< double > &offset, int type=0)
设置路径偏移
- int [pathOffsetDisable](#) ()
路径偏移失能
- int [pathOffsetSupv](#) (const std::vector< double > &min, const std::vector< double > &max, int strategy)
- int [jointOffsetEnable](#) ()
关节偏移使能
- int [jointOffsetSet](#) (const std::vector< double > &offset, int type=1)
设置关节偏移
- int [jointOffsetDisable](#) ()
关节偏移失能
- int [getQueueSize](#) ()
获取已经入队的指令段 (*INST*) 数量, 运动指令包括 *moveJoint*/*moveLine*/*moveCircle* 等运动指令以及 *setPayload* 等配置指令
- int [getTrajectoryQueueSize](#) ()
获取已经入队的运动规划插补点数量
- int [getExecId](#) ()
获取当前正在插补的运动指令段的*ID*
- double [getDuration](#) (int id)
获取指定*ID* 的运动指令段的预期执行时间
- double [getMotionLeftTime](#) (int id)
获取指定*ID* 的运动指令段的剩余执行时间
- int [stopMove](#) (bool quick, bool all_tasks)
StopMove 用于临时停止机器人和外部轴的运动以及相关工艺进程。如果调用 *StartMove* 指令, 则运动和工艺进程将恢复。
- int [startMove](#) ()
StartMove 用于在以下情况下恢复机器人、外部轴的运动以及相关工艺进程: • 通过 *StopMove* 指令停止后。• 执行 *StorePath ... RestoPath* 序列后。• 发生异步运动错误 (如 *ERR_PATH_STOP*) 或特定工艺错误并在 *ERROR* 处理器中处理后。
- int [storePath](#) (bool keep_sync)
storePath
- int [clearPath](#) ()
ClearPath (清除路径) 清除当前运动路径层级 (基础层级或 *StorePath* 层级) 上的所有运动路径。
- int [restoPath](#) ()
restoPath
- double [getProgress](#) ()
获取当前运动指令段的执行进度
- int [setWorkObjectHold](#) (const std::string &module_name, const std::vector< double > &mounting_pose)
当工件安装在另外一台机器人的末端或者外部轴上时, 指定其名字和安装位置
- std::tuple< std::string, std::vector< double > > [getWorkObjectHold](#) ()
获取工件安装信息
- std::vector< double > [getPauseJointPositions](#) ()
- int [setResumeStartPoint](#) (const std::vector< double > &q, int move_type, double blend_radius, const std::vector< double > &qdmax, const std::vector< double > &qddmax, const std::vector< double > &vmax, const std::vector< double > &amax)
设置继续运动参数
- int [getResumeMode](#) ()
获取继续运动模式

- ARCS_DEPRECATED int `setServoMode` (bool enable)
设置伺服模式使用 `setServoModeSelect` 替代
- ARCS_DEPRECATED bool `isServoModeEnabled` ()
判断伺服模式是否使能使用 `getServoModeSelect` 替代
- int `setServoModeSelect` (int mode)
设置伺服运动模式
- int `getServoModeSelect` ()
获取伺服运动模式
- int `servoJoint` (const std::vector< double > &q, double a, double v, double t, double lookahead←_time, double gain)
关节空间伺服
- int `servoCartesian` (const std::vector< double > &pose, double a, double v, double t, double lookahead←_time, double gain)
笛卡尔空间伺服
- int `servoJointWithAxes` (const std::vector< double > &q, const std::vector< double > &extq, double a, double v, double t, double lookahead←_time, double gain)
伺服运动 (带外部轴), 用于执行离线轨迹、透传用户规划轨迹等
- int `servoJointWithAxisGroup` (const std::vector< double > &q, double a, double v, double t, double lookahead←_time, double gain, const std::string &group_name, const std::vector< double > &extq)
- int `servoCartesianWithAxes` (const std::vector< double > &pose, const std::vector< double > &extq, double a, double v, double t, double lookahead←_time, double gain)
伺服运动 (带外部轴), 用于执行离线轨迹、透传用户规划轨迹等与 `servoJointWithAxes` 区别在于接收笛卡尔空间位姿而不是关节角度 (由软件内部直接做逆解)
- int `servoCartesianWithAxisGroup` (const std::vector< double > &pose, double a, double v, double t, double lookahead←_time, double gain, const std::string &group_name, const std::vector< double > &extq)
- int `trackJoint` (const std::vector< double > &q, double t, double smooth_scale, double delay←_sacle)
跟踪运动, 用于执行离线轨迹、透传用户规划轨迹等
- int `trackCartesian` (const std::vector< double > &pose, double t, double smooth_scale, double delay←_sacle)
跟踪运动, 用于执行离线轨迹、透传用户规划轨迹等与 `trackJoint` 区别在于接收笛卡尔空间位姿而不是关节角度 (由软件内部直接做逆解)
- int `followJoint` (const std::vector< double > &q)
关节空间跟随
- int `followLine` (const std::vector< double > &pose)
笛卡尔空间跟随
- int `speedJoint` (const std::vector< double > &qd, double a, double t)
关节空间速度跟随
- int `resumeSpeedJoint` (const std::vector< double > &qd, double a, double t)
关节空间速度跟随 (机械臂运行工程时发生碰撞, 通过此接口移动到安全位置)
- int `speedLine` (const std::vector< double > &xd, double a, double t)
笛卡尔空间速度跟随
- int `resumeSpeedLine` (const std::vector< double > &xd, double a, double t)
笛卡尔空间速度跟随 (机械臂运行工程时发生碰撞, 通过此接口移动到安全位置)
- int `moveSpline` (const std::vector< double > &q, double a, double v, double duration)
在关节空间做样条插值
- int `moveJoint` (const std::vector< double > &q, double a, double v, double blend←_radius, double duration)
添加关节运动
- int `moveJointWithAxisGroup` (const std::vector< double > &q, double a, double v, double blend←_radius, double duration, const std::string &group_name, const std::vector< double > &extq)
机器人与外部轴同步运动

- int [resumeMoveJoint](#) (const std::vector< double > &q, double a, double v, double duration)
通过关节运动移动到暂停点的位置
- int [moveLine](#) (const std::vector< double > &pose, double a, double v, double blend_radius, double duration)
添加直线运动
- int [moveLineWithAxisGroup](#) (const std::vector< double > &pose, double a, double v, double blend_radius, double duration, const std::string &group_name, const std::vector< double > &extq)
直线运动与外部轴同步运动
- int [moveProcess](#) (const std::vector< double > &pose, double a, double v, double blend_radius)
添加工艺运动
- int [resumeMoveLine](#) (const std::vector< double > &pose, double a, double v, double duration)
通过直线运动移动到暂停点的位置
- int [moveCircle](#) (const std::vector< double > &via_pose, const std::vector< double > &end_pose, double a, double v, double blend_radius, double duration)
添加圆弧运动
- int [moveCircleWithAxisGroup](#) (const std::vector< double > &via_pose, const std::vector< double > &end_pose, double a, double v, double blend_radius, double duration, const std::string &group_name, const std::vector< double > &extq)
moveCircle 与外部轴同步运动
- int [setCirclePathMode](#) (int mode)
设置圆弧路径模式
- int [moveCircle2](#) (const [CircleParameters](#) &param)
高级圆弧或者圆周运动
- int [pathBufferAlloc](#) (const std::string &name, int type, int size)
新建一个路径点缓存
- int [pathBufferAppend](#) (const std::string &name, const std::vector< std::vector< double > > &way-points)
向路径缓存添加路点
- int [pathBufferEval](#) (const std::string &name, const std::vector< double > &a, const std::vector< double > &v, double t)
计算、优化等耗时操作，传入的参数相同时不会重新计算
- bool [pathBufferValid](#) (const std::string &name)
指定名字的 *buffer* 是否有效
- int [pathBufferFree](#) (const std::string &name)
释放路径缓存
- int [pathBufferFilter](#) (const std::string &name, int order, double fd, double fs)
关节空间路径滤波器
- std::vector< std::string > [pathBufferList](#) ()
列出所有缓存路径的名字
- int [movePathBuffer](#) (const std::string &name)
执行缓存的路径
- int [moveIntersection](#) (const std::vector< std::vector< double > > &poses, double a, double v, double main_pipe_radius, double sub_pipe_radius, double normal_distance, double normal_alpha)
相贯线接口
- int [stopJoint](#) (double acc)
关节空间停止运动
- int [resumeStopJoint](#) (double acc)
关节空间停止运动 (机械臂运行工程时发生碰撞，通过 *resumeSpeedJoint* 接口移动到安全位置后需要停止时调用此接口)
- int [stopLine](#) (double acc, double acc_rot)
停止 *moveLine*/*moveCircle* 等在笛卡尔空间的运动
- int [resumeStopLine](#) (double acc, double acc_rot)

- 笛卡尔空间停止运动 (机械臂运行工程时发生碰撞, 通过 *resumeSpeedLine* 接口移动到安全位置后需要停止时调用此接口)
- int *weaveStart* (const std::string &params)
 - int *weaveEnd* ()
结束摆动
 - int *setFuturePointSamplePeriod* (double sample_time)
设置未来路径上点的采样时间间隔
 - std::vector< std::vector< double > > *getFuturePathPointsJoint* ()
获取未来路径上的轨迹点
 - int *setConveyorTrackEncoder* (int encoder_id, int tick_per_meter)
设置传送带编码器参数
 - int *conveyorTrackCircle* (int encoder_id, const std::vector< double > &center, bool rotate_tool)
圆形传送带跟随
 - int *conveyorTrackLine* (int encoder_id, const std::vector< double > &direction)
线性传送带跟随
 - int *conveyorTrackStop* (int encoder_id, double a)
终止传送带跟随
 - bool *conveyorTrackSwitch* (int encoder_id)
切换传送带追踪物品如果当前物品正在处于跟踪状态, 则将该物品出队, 不再跟踪, 返回 *true* 如果没有物品正在处于跟踪状态, 返回 *false*
 - bool *hasItemOnConveyorToTrack* (int encoder_id)
传送带上是否有物品可以跟踪
 - int *conveyorTrackCreatItem* (int encoder_id, int item_id, const std::vector< double > &offset)
增加传送带队列
 - int *setConveyorTrackCompensate* (int encoder_id, double comp)
设置传送带跟踪的补偿值
 - bool *isConveyorTrackSync* (int encoder_id)
判断传送带与机械臂之间是否达到相对静止
 - int *setConveyorTrackLimit* (int encoder_id, double limit)
设置传送带跟踪的最大距离限制
 - int *setConveyorTrackStartWindow* (int encoder_id, double window_min, double window_max)
设置传送带跟踪的启动窗口
 - int *setConveyorTrackSensorOffset* (int encoder_id, double offset)
设置传送带示教位置到同步开关之间的距离
 - int *setConveyorTrackSyncSeparation* (int encoder_id, double distance, double time)
设置传送带同步分离, 用于过滤掉同步开关中不需要的信号
 - int *setConveyorTrackVelThreshold* (int encoder_id, double threshold)
设置传送带运行速度阈值, 用于过滤掉编码器突变值导致的位置突变
 - bool *isConveyorTrackExceed* (int encoder_id)
传送带上工件的移动距离是否超过最大限值
 - int *conveyorTrackClearItems* (int encoder_id)
清空传动带队列中的所有对象
 - std::vector< int > *getConveyorTrackQueue* (int encoder_id)
获取传送带队列的编码器值
 - int *getConveyorTrackNextItem* (int encoder_id)
获取下一个跟踪的传送带物品的 *id*
 - int *moveSpiral* (const *SpiralParameters* &param, double blend_radius, double v, double a, double t)
螺旋线运动
 - int *getLookAheadSize* ()
获取前瞻段数

- int [setLookAheadSize](#) (int size)
设置前瞻段数 1.
- int [weaveUpdateParameters](#) (const std::string &params)
更新摆动过程中的频率和振幅
- int [enableJointSoftServo](#) (const std::vector< double > &stiffness)
设置关节电流环刚度系数
- int [disableJointSoftServo](#) ()
关闭关节电流环刚度系数
- bool [isJointSoftServoEnabled](#) ()
判断关节电流环刚度系数是否使能
- int [enableVibrationSuppress](#) (const std::vector< double > &omega, const std::vector< double > &zeta, int level)
打开振动抑制
- int [disbaleVibrationSuppress](#) ()
关闭振动抑制
- int [setTimeOptimalEnable](#) (bool enable)
设置时间最优算法默认关闭
- bool [isTimeOptimalEnabled](#) ()
获取时间最优算法状态: *true* - 开启 *false* - 关闭
- bool [isSupportedTimeOptimal](#) ()
获取是否支持时间最优算法: *true* - 支持 *false* - 不支持
- int [setTcpMaxLinearVelocity](#) (double v)
设置 TCP 最大线速度
- double [getTcpMaxLinearVelocity](#) ()
获取 TCP 最大线速度
- int [resetTcpMaxLinearVelocity](#) ()
重置 TCP 最大线速度
- int [setEndPath](#) ()
设置轨迹终止点 (以当前轨迹段为界)
- int [getWeaveDirection](#) ()
获取当前摆焊轨迹点方向

Protected 属性

- void * [d__](#)

11.11.1 详细描述

在文件 [motion_control.h](#) 第 69 行定义.

11.11.2 构造及析构函数说明

MotionControl()

```
arcs::common_interface::MotionControl::MotionControl ()
```

~MotionControl()

```
virtual arcs::common_interface::MotionControl::~~MotionControl () [virtual]
```

11.11.3 成员函数说明

getConveyorTrackNextItem()

```
int arcs::common_interface::MotionControl::getConveyorTrackNextItem (
```

```
int encoder_id)
```

获取下一个跟踪的传送带物品的 id

参数

<i>encoder↔ _id</i>	预留

返回

返回物品 id, 没有 next item 返回 -1

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
getConveyorTrackNextItem(self: pyaubo_sdk.MotionControl, arg0: int) -> int
```

Lua 函数原型

```
getConveyorTrackNextItem(encoder_id: number) -> int
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.getConveyorTrackNextItem", "params": [0], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": {10}}
```

isConveyorTrackExceed()

```
bool arcs::common_interface::MotionControl::isConveyorTrackExceed (  
    int encoder_id)
```

传送带上工件的移动距离是否超过最大限值

参数

<i>encoder↔ _id</i>	预留

返回

true : 移动距离超过了最大限值 false : 移动距离没有超过了最大限值

异常

<i>arcs::common_interface::AuboException</i>	

Python 函数原型

```
isConveyorTrackExceed(self: pyaubo_sdk.MotionControl, arg0: int) -> bool
```

Lua 函数原型

```
isConveyorTrackExceed(encoder_id: number) -> bool
```

JSON-RPC 请求示例

```
{"jsonrpc": "2.0", "method": "rob1.MotionControl.isConveyorTrackExceed", "params": [0], "id": 1}
```

JSON-RPC 响应示例

```
{"id": 1, "jsonrpc": "2.0", "result": false}
```

servoCartesianWithAxisGroup()

```
int arcs::common_interface::MotionControl::servoCartesianWithAxisGroup (
    const std::vector< double > & pose,
    double a,
    double v,
    double t,
    double lookahead_time,
    double gain,
    const std::string & group_name,
    const std::vector< double > & extq)
```

servoJointWithAxisGroup()

```
int arcs::common_interface::MotionControl::servoJointWithAxisGroup (
    const std::vector< double > & q,
    double a,
    double v,
    double t,
    double lookahead_time,
    double gain,
    const std::string & group_name,
    const std::vector< double > & extq)
```

11.11.4 类成员变量说明

d_

```
void* arcs::common_interface::MotionControl::d_ [protected]
```

在文件 [motion_control.h](#) 第 6777 行定义.

该类的文档由以下文件生成:

- include/aubo/robot/[motion_control.h](#)

11.12 arcs::common_interface::RegisterControl 类参考

```
#include <register_control.h>
```

Public 成员函数

- `RegisterControl ()`
- `virtual ~RegisterControl ()`
- `bool getBoolInput (uint32_t address)`
 从一个输入寄存器中读取布尔值，也可以通过现场总线进行访问。注意，它使用自己的内存空间。
- `int setBoolInput (uint32_t address, bool value)`
- `int getInt32Input (uint32_t address)`
 从一个输入寄存器中读取整数值，也可以通过现场总线进行访问。注意，它使用自己的内存空间。
- `int setInt32Input (uint32_t address, int value)`
- `float getFloatInput (uint32_t address)`
Reads the float from one of the input registers, which can also be accessed by a Field bus.
- `int setFloatInput (uint32_t address, float value)`
- `double getDoubleInput (uint32_t address)`
 从一个输入寄存器中读取双精度浮点数，也可以通过现场总线进行访问。注意，它使用自己的内存空间。
- `int setDoubleInput (uint32_t address, double value)`
- `bool getBoolOutput (uint32_t address)`
 从一个输出寄存器中读取布尔值，也可以通过现场总线进行访问。注意，它使用自己的内存空间。
- `int setBoolOutput (uint32_t address, bool value)`
- `int getInt32Output (uint32_t address)`
 从一个输出寄存器中读取整数值，也可以通过现场总线进行访问。注意，它使用自己的内存空间。
- `int setInt32Output (uint32_t address, int value)`
- `float getFloatOutput (uint32_t address)`
 从一个输出寄存器中读取浮点数，也可以通过现场总线进行访问。注意，它使用自己的内存空间。
- `int setFloatOutput (uint32_t address, float value)`
- `double getDoubleOutput (uint32_t address)`
 从一个输出寄存器中读取双精度浮点数。
- `int setDoubleOutput (uint32_t address, double value)`
- `int16_t getInt16Register (uint32_t address)`
 用于 *Modbus Slave*
- `int setInt16Register (uint32_t address, int16_t value)`
- `bool getInt16RegisterBit (uint32_t address, uint8_t bit_offset)`
 获取 *Int16* 寄存器的某个 *bit* 的状态
- `int setInt16RegisterBit (uint32_t address, uint8_t bit_offset, bool value)`
 设置 *Int16* 寄存器的某个 *bit* 的状态
- `bool hasNamedVariable (const std::string &key)`
 具名变量是否存在
- `std::string getNamedVariableType (const std::string &key)`
 获取具名变量的类型
- `bool variableUpdated (const std::string &key, uint64_t since)`
 具名变量是否更新
- `bool getBool (const std::string &key, bool default_value)`
 获取变量值
- `int setBool (const std::string &key, bool value)`
 设置/更新变量值
- `std::vector< char > getVecChar (const std::string &key, const std::vector< char > &default_value)`
 获取变量值
- `int setVecChar (const std::string &key, const std::vector< char > &value)`
 设置/更新变量值
- `int getInt32 (const std::string &key, int default_value)`
 获取变量值
- `int setInt32 (const std::string &key, int value)`

- 设置/更新变量值
- `std::vector< int32_t > getVecInt32` (const std::string &key, const std::vector< int32_t > &default_value)
- 获取变量值
- `int setVecInt32` (const std::string &key, const std::vector< int32_t > &value)
- 设置/更新变量值
- `float getFloat` (const std::string &key, float default_value)
- 获取变量值
- `int setFloat` (const std::string &key, float value)
- 设置/更新变量值
- `std::vector< float > getVecFloat` (const std::string &key, const std::vector< float > &default_value)
- 获取变量值
- `int setVecFloat` (const std::string &key, const std::vector< float > &value)
- 设置/更新变量值
- `double getDouble` (const std::string &key, double default_value)
- 获取变量值
- `int setDouble` (const std::string &key, double value)
- 设置/更新变量值
- `std::vector< double > getVecDouble` (const std::string &key, const std::vector< double > &default_value)
- 获取变量值
- `int setVecDouble` (const std::string &key, const std::vector< double > &value)
- 设置/更新变量值
- `std::string getString` (const std::string &key, const std::string &default_value)
- 获取变量值
- `int setString` (const std::string &key, const std::string &value)
- 设置/更新变量值
- `int clearNamedVariable` (const std::string &key)
- 清除变量
- `int setWatchDog` (const std::string &key, double timeout, int action)
- 设置看门狗
- `int getWatchDogAction` (const std::string &key)
- 获取看门狗动作
- `int getWatchDogTimeout` (const std::string &key)
- 获取看门狗超时时间
- `int modbusAddSignal` (const std::string &device_info, int slave_number, int signal_address, int signal_type, const std::string &signal_name, bool sequential_mode)
- 添加一个新的 *Modbus* 信号以供控制器监视。不需要返回响应。
- `int addFixedModbusGroup` (const std::string &signal_name, int count)
- 为指定 *Modbus* 寄存器信号创建固定读取分组，优化连续地址的批量读取效率
- `int modbusDeleteSignal` (const std::string &signal_name)
- 删除指定名称的信号。
- `int modbusDeleteAllSignals` ()
- 删除所有 *modbus* 信号
- `int modbusGetSignalStatus` (const std::string &signal_name)
- 读取特定信号的当前值。
- `std::vector< std::string > modbusGetSignalNames` ()
- 获取所有信号的名字集合
- `std::vector< int > modbusGetSignalTypes` ()
- 获取所有信号的类型集合
- `std::vector< int > modbusGetSignalValues` ()

- 获取所有信号的数值集合

 - `std::vector< int > modbusGetSignalErrors ()`
 获取所有信号的请求是否有错误 (0:无错误, 其他:有错误) 集合
- `int modbusSendCustomCommand (const std::string &device_info, int slave_number, int function_code, const std::vector< uint8_t > &data)`
 将用户指定的命令发送到指定IP 地址上的Modbus 单元。由于不会接收到响应, 因此不能用于请求数据。用户负责提供对所提供的功能码有意义的数据。内置函数负责构建Modbus 帧, 因此用户不需要关心命令的长度。
- `int modbusSetDigitalInputAction (const std::string &robot_name, const std::string &signal_name, StandardInputAction action)`
 将选择的数字输入信号设置为 “default” 或 “freedrive”
- `int modbusSetOutputRunstate (const std::string &robot_name, const std::string &signal_name, StandardOutputRunState runstate)`
 设置 Modbus 信号输出动作
- `int modbusSetOutputRunstate1 (const std::string &robot_name, const std::string &signal_name, StandardOutputRunState runstate, const std::string &object_name, double threshold)`
 设置 Modbus 信号输出动作, 并支持基于路点/平面/障碍物的条件阈值判断。
- `int modbusSetOutputSignal (const std::string &signal_name, uint16_t value)`
 将指定名称的输出寄存器信号设置为给定的值
- `int modbusSetOutputSignal1 (const std::string &signal_name, const std::vector< uint16_t > &values)`
 将从指定名称开始的若干个连续输出寄存器信号设置为给定的值
- `int modbusSetOutputSignalWithTimeout (const std::string &signal_name, uint16_t value, double timeout)`
 将指定名称的输出寄存器信号设置为给定的值, 并且带超时判断
- `int modbusSetOutputSignalPulse (const std::string &signal_name, uint16_t value, double duration)`
 设置 modbus 信号输出脉冲 (仅支持线圈输出类型)
- `int modbusSetSignalUpdateFrequency (const std::string &signal_name, int update_frequency)`
 设置机器人向Modbus 控制器发送请求的频率, 用于读取或写入信号值
- `int modbusGetSignalIndex (const std::string &signal_name)`
 获取指定 modbus 信号索引, 从 0 开始, 不存在则返回-1
- `int modbusGetSignalError (const std::string &signal_name)`
 获取指定 modbus 信号的错误状态
- `int getModbusDeviceStatus (const std::string &device_name)`
 获取指定 modbus 设备的连接状态
- `int addModbusEncoder (int encoder_id, int range_id, const std::string &signal_name)`
 将某个 modbus 寄存器信号作为编码器
- `int addInt32RegEncoder (int encoder_id, int range_id, const std::string &key)`
 添加Int32 寄存器的虚拟编码器
- `int deleteVirtualEncoder (int encoder_id)`
 删除虚拟编码器

Protected 属性

- `void * d_`

11.12.1 详细描述

在文件 `register_control.h` 第 64 行定义.

11.12.2 构造及析构函数说明

RegisterControl()

```
arcs::common_interface::RegisterControl::RegisterControl ()
```

~RegisterControl()

```
virtual arcs::common_interface::RegisterControl::~RegisterControl () [virtual]
```

11.12.3 成员函数说明

modbusSetOutputSignalWithTimeout()

```
int arcs::common_interface::RegisterControl::modbusSetOutputSignalWithTimeout (
    const std::string & signal_name,
    uint16_t value,
    double timeout)
```

将指定名称的输出寄存器信号设置为给定的值，并且带超时判断

参数

<i>signal_name</i>	提前被添加的输出寄存器信号
<i>value</i>	必须是有效的整数，范围是 0-65535
<i>timeout</i>	超时时间，单位秒

返回

Python 函数原型

```
modbusSetOutputSignalWithTimeout(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: int, arg2:↵
double) -> int
```

Lua 函数原型

```
modbusSetOutputSignalWithTimeout(signal_name: string, value: number, timeout: number) ->
nil
```

Lua 示例

```
modbusSetOutputSignalWithTimeout("Modbus_0",0,0.5)
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"RegisterControl.modbusSetOutputSignalWithTimeout","params":["Modbus↵
_0",0,0.5],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":0}
```

11.12.4 类成员变量说明

d_

```
void* arcs::common_interface::RegisterControl::d_ [protected]
```

在文件 [register_control.h](#) 第 3546 行定义.

该类的文档由以下文件生成:

- include/aubo/[register_control.h](#)

11.13 arcs::common_interface::RobotAlgorithm 类参考

```
#include <robot_algorithm.h>
```

Public 成员函数

- [RobotAlgorithm \(\)](#)
- [virtual ~RobotAlgorithm \(\)](#)
- [ForceSensorCalibResult calibrateTcpForceSensor](#) (const std::vector< std::vector< double > > &forces, const std::vector< std::vector< double > > &poses)
力传感器标定算法 (三点标定法)
- [ForceSensorCalibResultWithError calibrateTcpForceSensor2](#) (const std::vector< std::vector< double > > &forces, const std::vector< std::vector< double > > &poses)
力传感器标定算法 (三点标定法)
- [ResultWithErrno calibrateTcpForceSensor3](#) (const std::vector< std::vector< double > > &forces, const std::vector< std::vector< double > > &poses, const double &mass, const std::vector< double > &cog)
力传感器偏置标定算法
- [int payloadIdentify](#) (const std::string &data_file_no_payload, const std::string &data_file_with_payload)
基于电流的负载辨识算法接口
- [int payloadIdentify1](#) (const std::string &file_name)
新版基于电流的负载辨识算法接口
- [int payloadCalculateFinished \(\)](#)
负载辨识是否计算完成
- [Payload getPayloadIdentifyResult \(\)](#)
获取负载辨识结果
- [bool frictionModelIdentify](#) (const std::vector< std::vector< double > > &q, const std::vector< std::vector< double > > &qd, const std::vector< std::vector< double > > &qdd, const std::vector< std::vector< double > > &temp)
关节摩擦力模型辨识算法接口
- [ResultWithErrno calibWorkpieceCoordinatePara](#) (const std::vector< std::vector< double > > &q, int type)
工件坐标系标定算法接口 (需要在调用之前正确的设置机器人的TCP 偏移) 输入多组关节角度和标定类型, 输出工件坐标系位姿 (相对于机器人基坐标系)
- [ResultWithErrno forwardDynamics](#) (const std::vector< double > &q, const std::vector< double > &torqs)
动力学正解
- [ResultWithErrno forwardDynamics1](#) (const std::vector< double > &q, const std::vector< double > &torqs, const std::vector< double > &tcp_offset)
动力学正解, 基于给定的TCP 偏移
- [ResultWithErrno forwardKinematics](#) (const std::vector< double > &q)
运动学正解, 基于激活的TCP 偏移 (最近的通过 *setTcpOffset* 设置的参数) 输入关节角度, 输出TCP 位姿
- [ResultWithErrno forwardKinematics1](#) (const std::vector< double > &q, const std::vector< double > &tcp_offset)
运动学正解输入关节角度, 输出TCP 位姿
- [ResultWithErrno forwardToolKinematics](#) (const std::vector< double > &q)
运动学正解 (忽略 TCP 偏移值)
- [ResultWithErrno1 forwardKinematicsAll](#) (const std::vector< double > &q)
运动学正解, 基于激活的TCP 偏移 (最近的通过 *setTcpOffset* 设置的参数) 输入关节角度, 输出各连杆位姿
- [ResultWithErrno inverseKinematics](#) (const std::vector< double > &qnear, const std::vector< double > &pose)
运动学逆解输入TCP 位姿和参考关节角度, 输出关节角度

- [ResultWithErrno inverseKinematics1](#) (const std::vector< double > &qnear, const std::vector< double > &pose, const std::vector< double > &tcp_offset)
运动学逆解输入 TCP 位姿和参考关节角度, 输出关节角度
- [ResultWithErrno inverseKinematics2](#) (const std::vector< double > &qnear, const std::vector< double > &pose)
同构型最小关节变化逆解按照 qnear 的机械臂构型计算目标 TCP 位姿逆解, 并在同构型逆解中选择相对 qnear 关节变化最小的一组基于激活的 TCP 偏移 (最近的通过 setTcpOffset 设置的参数)
- [ResultWithErrno1 inverseKinematicsAll](#) (const std::vector< double > &pose)
求出所有的逆解, 基于激活的 TCP 偏移
- [ResultWithErrno1 inverseKinematicsAll1](#) (const std::vector< double > &pose, const std::vector< double > &tcp_offset)
求出所有的逆解, 基于提供的 TCP 偏移
- [ResultWithErrno inverseToolKinematics](#) (const std::vector< double > &qnear, const std::vector< double > &pose)
运动学逆解 (忽略 TCP 偏移值)
- [ResultWithErrno1 inverseToolKinematicsAll](#) (const std::vector< double > &pose)
运动学逆解 (忽略 TCP 偏移值)
- [ResultWithErrno3 getRobotConfiguration](#) (const std::vector< double > &q)
根据输入的关节角计算并返回对应的机械臂构型
- std::vector< std::vector< double > > [pathMovej](#) (const std::vector< double > &q1, double r1, const std::vector< double > &q2, double r2, double d)
求解 movej 之间的轨迹点
- [ResultWithErrno calcJacobian](#) (const std::vector< double > &q, bool base_or_end)
计算机机械臂末端的雅克比矩阵
- std::vector< std::vector< double > > [pathBlend3Points](#) (int type, const std::vector< double > &q_start, const std::vector< double > &q_via, const std::vector< double > &q_to, double r, double d)
求解交融的轨迹点
- int [generatePayloadIdentifyTraj](#) (const std::string &name, const [TrajConfig](#) &traj_conf)
生成用于负载辨识的激励轨迹此接口内部调用 pathBufferAppend 将离线轨迹存入 buffer 中, 后续可通过 movePathBuffer 运行离线轨迹
- int [payloadIdentifyTrajGenFinished](#) ()
负载辨识轨迹是否生成完成
- std::vector< std::vector< double > > [pathMoveS](#) (const std::vector< std::vector< double > > &q, double d)
求解 moveS 的轨迹点
- [ResultWithErrno1 calibVibrationParams](#) (const std::vector< std::vector< double > > &q, const std::vector< std::vector< double > > &qd, const std::vector< std::vector< double > > &target_q, const std::vector< std::vector< double > > &target_qd, const std::vector< std::vector< double > > &target_qdd, const std::vector< double > &tool_offset)
振动抑制参数辨识算法接口
- [ResultWithErrno1 calibVibrationParams1](#) (const std::string &record_cache_name, const std::vector< double > &tool_offset)
振动抑制参数辨识算法接口 1
- int [needVibrationRecalib](#) (const [VibrationRecalibrationParameter](#) &param1, const [VibrationRecalibrationParameter](#) &param2, double threshold)
判断是否需要重新辨识振动参数
- int [addCollisionBox](#) (const std::string &name, const std::string &link_name, const std::vector< std::vector< double > > &sizes, const std::vector< std::vector< double > > &poses)
添加立方体碰撞对象
- int [removeCollisionObject](#) (const std::string &name)
删除指定的碰撞对象

- int `validatePath` (int type, const std::vector< double > &start, double r1, const std::vector< double > &end, double r2, double d)
验证机器人运动路径从起点到终点的可达性, 不启用外部碰撞检测。
- int `validatePath1` (int type, const std::vector< double > &start, double r1, const std::vector< double > &end, double r2, double d, bool check_external_collision)
验证机器人运动路径从起点到终点的可达性, 可选择是否启用外部碰撞检测。

Protected 属性

- void * `d_`

11.13.1 详细描述

在文件 `robot_algorithm.h` 第 30 行定义.

11.13.2 构造及析构函数说明

RobotAlgorithm()

```
arcs::common_interface::RobotAlgorithm::RobotAlgorithm ()
```

~RobotAlgorithm()

```
virtual arcs::common_interface::RobotAlgorithm::~~RobotAlgorithm () [virtual]
```

11.13.3 成员函数说明

pathMovej()

```
std::vector< std::vector< double > > arcs::common_interface::RobotAlgorithm::pathMovej (
    const std::vector< double > & q1,
    double r1,
    const std::vector< double > & q2,
    double r2,
    double d)
```

求解 movej 之间的轨迹点

参数

<i>q1</i>	movej 的起点
<i>r1</i>	在 q1 处的交融半径
<i>q2</i>	movej 的终点
<i>r2</i>	在 q2 处的交融半径
<i>d</i>	采样距离

返回

q1~q2 之间笛卡尔空间离散轨迹点 (x,y,z,rx,ry,rz) 集合

异常

<code>arcs::common_interface::AuboException</code>

Python 函数原型

```
pathMovej(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float], arg1: float, arg2: List[float],
, arg3: float, arg4: float) -> List[List[float]]
```

Lua 函数原型

```
pathMovej(q1: table, r1: number, q2: table, r2: number, d: number) -> table, number
```

Lua 示例

```
path , num = pathMovej({0.0,-0.2618,1.7453,0.4364,1.5711,0.0},0.25,{0.3234,-0.5405,1.5403,0.5881,1.↵
2962,0.7435},0.03,0.2)
```

11.13.4 类成员变量说明

d_

```
void* arcs::common_interface::RobotAlgorithm::d_ [protected]
```

在文件 [robot_algorithm.h](#) 第 1984 行定义.

该类的文档由以下文件生成:

- [include/aubo/robot/robot_algorithm.h](#)

11.14 arcs::common_interface::RobotConfig 类参考

```
#include <robot_config.h>
```

Public 成员函数

- [RobotConfig](#) ()
- virtual [~RobotConfig](#) ()
- std::string [getName](#) ()
获取机器人的名字
- int [getDof](#) ()
获取机器人的自由度 (从硬件抽象层读取)
- double [getCycletime](#) ()
获取机器人的伺服控制周期 (从硬件抽象层读取)
- int [setSlowDownFraction](#) (int level, double fraction)
预设缓速模式下的速度缩减比例
- double [getSlowDownFraction](#) (int level)
获取预设的缓速模式下的速度缩减比例
- double [getDefaultToolAcc](#) ()
获取默认的工具端加速度, 单位 m/s^2
- double [getDefaultToolSpeed](#) ()
获取默认的工具端速度, 单位 m/s
- double [getDefaultJointAcc](#) ()
获取默认关节的加速度, 单位 rad/s^2
- double [getDefaultJointSpeed](#) ()
获取默认关节的速度, 单位 rad/s
- std::string [getRobotType](#) ()
获取机器人类型代码
- std::string [getRobotSubType](#) ()
获取机器人子类型代码
- std::string [getControlBoxType](#) ()

- 获取控制柜类型代码
- int `setMountingPose` (const std::vector< double > &pose)
设置安装位姿 (机器人的基坐标系相对于世界坐标系) *world->base*
- std::vector< double > `getMountingPose` ()
获取安装位姿 (机器人的基坐标系相对于世界坐标系)
- int `attachRobotBaseTo` (const std::string &frame)
将机器人绑定到一个坐标系, 如果这个坐标系是运动的, 那机器人也会跟着运动应用于地轨或者龙门这个函数调用的时候 *frame* 和 *ROBOTBASE* 的相对关系就固定了
- std::string `getRobotBaseParent` ()
- int `setWorkObjectData` (const `WObjectData` &wobj)
设置工件数据, 编程点位都是基于工件坐标系
- int `setCollisionLevel` (int level)
设置碰撞灵敏度等级数值越大越灵敏
- int `getCollisionLevel` ()
获取碰撞灵敏度等级
- int `setCollisionStopType` (int type)
设置碰撞停止类型
- int `getCollisionStopType` ()
获取碰撞停止类型
- int `setHomePosition` (const std::vector< double > &positions)
设置机器人的原点位置
- std::vector< double > `getHomePosition` ()
获取机器人的原点位置
- int `setFreedriveDamp` (const std::vector< double > &damp)
设置拖动阻尼
- std::vector< double > `getFreedriveDamp` ()
获取拖动阻尼
- int `setHandguidDamp` (const std::vector< double > &damp)
设置混合拖动阻尼
- std::vector< double > `getHandguidDamp` ()
获取混合拖动阻尼
- std::unordered_map< std::string, std::vector< double > > `getKinematicsParam` (bool real)
获取机器人 *DH* 参数 *alpha a d theta beta*
- std::unordered_map< std::string, std::vector< double > > `getKinematicsCompensate` (double ref←_temperature)
获取指定温度下的 *DH* 参数补偿值: *alpha a d theta beta*
- int `setKinematicsCompensate` (const std::unordered_map< std::string, std::vector< double > > &param)
设置标准 *DH* 补偿到机器人
- int `setPersistentParameters` (const std::string &param)
设置需要保存到接口板底座的参数
- int `setHardwareCustomParameters` (const std::string &param)
设置硬件抽象层自定义参数目的是为了做不同硬件之间的兼容
- std::string `getHardwareCustomParameters` (const std::string &param)
获取硬件抽象层自定义参数
- int `setRobotZero` ()
设置机器人关节零位
- std::vector< std::string > `getTcpForceSensorNames` ()
获取可用的末端力矩传感器的名字
- int `selectTcpForceSensor` (const std::string &name)
设置末端力矩传感器如果存在内置的末端力矩传感器, 默认将使用内置的力矩传感器

- int [setTcpForceSensorPose](#) (const std::vector< double > &sensor_pose)
设置传感器安装位姿
- std::vector< double > [getTcpForceSensorPose](#) ()
获取传感器安装位姿
- bool [hasTcpForceSensor](#) ()
是否安装了末端力矩传感器
- int [setTcpForceOffset](#) (const std::vector< double > &force_offset)
设置末端力矩偏移
- std::vector< double > [getTcpForceOffset](#) ()
获取末端力矩偏移
- std::vector< std::string > [getBaseForceSensorNames](#) ()
获取可用的底座力矩传感器的名字
- int [selectBaseForceSensor](#) (const std::string &name)
设置底座力矩传感器如果存在内置的底座力矩传感器，默认将使用内置的力矩传感器
- bool [hasBaseForceSensor](#) ()
是否安装了底座力矩传感器
- int [setBaseForceOffset](#) (const std::vector< double > &force_offset)
设置底座力矩偏移
- std::vector< double > [getBaseForceOffset](#) ()
获取底座力矩偏移
- uint32_t [getSafetyParametersChecksum](#) ()
获取安全参数校验码 *CRC32*
- int [confirmSafetyParameters](#) (const [RobotSafetyParameterRange](#) &parameters)
发起确认安全配置参数请求：将安全配置参数写入到安全接口板 *flash* 或文件
- uint32_t [calcSafetyParametersChecksum](#) (const [RobotSafetyParameterRange](#) &parameters)
计算安全参数的 *CRC32* 校验值
- std::vector< double > [getJointMaxPositions](#) ()
获取关节最大位置（物理极限）
- std::vector< double > [getJointMinPositions](#) ()
获取关节最小位置（物理极限）
- std::vector< double > [getJointMaxSpeeds](#) ()
获取关节最大速度（物理极限）
- std::vector< double > [getJointMaxAccelerations](#) ()
获取关节最大加速度（物理极限）
- std::vector< double > [getTcpMaxSpeeds](#) ()
获取 *TCP* 最大速度（物理极限）
- std::vector< double > [getTcpMaxAccelerations](#) ()
获取 *TCP* 最大加速度（物理极限）
- int [setGravity](#) (const std::vector< double > &gravity)
设置机器人安装姿态
- std::vector< double > [getGravity](#) ()
获取机器人的安装姿态
- int [setTcpOffset](#) (const std::vector< double > &offset)
设置当前的 *TCP* 偏移
- std::vector< double > [getTcpOffset](#) ()
获取当前的 *TCP* 偏移
- int [setToolInertial](#) (double m, const std::vector< double > &com, const std::vector< double > &inertial)
设置工具端质量、质心及惯量
- int [setPayload](#) (double m, const std::vector< double > &cog, const std::vector< double > &aom, const std::vector< double > &inertia)

- 设置有效负载
- `Payload getPayload ()`
获取有效负载
- `bool toolSpaceInRange (const std::vector< double > &pose)`
末端位姿是否在安全范围之内
- `int firmwareUpdate (const std::string &fw)`
发起固件升级请求, 控制器软件将进入固件升级模式
- `std::tuple< std::string, double > getFirmwareUpdateProcess ()`
获取当前的固件升级的进程
- `std::vector< double > getLimitJointMaxPositions ()`
获取关节最大位置 (当前正在使用的限制值)
- `std::vector< double > getLimitJointMinPositions ()`
获取关节最小位置 (当前正在使用的限制值)
- `std::vector< double > getLimitJointMaxSpeeds ()`
获取关节最大速度 (当前正在使用的限制值)
- `std::vector< double > getLimitJointMaxAccelerations ()`
获取关节最大加速度 (当前正在使用的限制值)
- `double getLimitTcpMaxSpeed ()`
获取 TCP 最大速度 (当前正在使用的限制值)
- `SafeguedStopType getSafeguardStopType ()`
获取当前安全停止的类型
- `int getSafeguardStopSource ()`
按位获取完整的安全停止触发源
- `int getRobotEmergencyStopSource ()`
按位获取完整的机器人紧急停止触发源
- `std::string getSelectedTcpForceSensorName ()`
获取工具端力矩传感器的名字
- `int attachWeldingGun (double m, const std::vector< double > &cog, const std::vector< double > &inertia)`
添加焊枪动力学参数
- `int setCollisionThreshold (const std::vector< double > &threshold)`
设置碰撞阈值
- `std::vector< double > getCollisionThreshold ()`
获取碰撞阈值
- `int enableAxisGroup (const std::string &group_name)`
- `int disableAxisGroup ()`
- `int enableEndCollisionCheck (bool enable)`
使能末端碰撞检测
- `bool isEndCollisionCheckEnabled ()`
获取末端碰撞检测是否使能

Protected 属性

- `void * d_`

11.14.1 详细描述

在文件 `robot_config.h` 第 21 行定义.

11.14.2 构造及析构函数说明

RobotConfig()

`arcs::common_interface::RobotConfig::RobotConfig ()`

~RobotConfig()

```
virtual arcs::common_interface::RobotConfig::~~RobotConfig () [virtual]
```

11.14.3 成员函数说明

disableAxisGroup()

```
int arcs::common_interface::RobotConfig::disableAxisGroup ()
```

enableAxisGroup()

```
int arcs::common_interface::RobotConfig::enableAxisGroup (
    const std::string & group_name)
```

getRobotBaseParent()

```
std::string arcs::common_interface::RobotConfig::getRobotBaseParent ()
```

11.14.4 类成员变量说明

d_

```
void* arcs::common_interface::RobotConfig::d_ [protected]
```

在文件 [robot_config.h](#) 第 3806 行定义.

该类的文档由以下文件生成:

- [include/aubo/robot/robot_config.h](#)

11.15 arcs::common_interface::RobotInterface 类参考

```
#include <robot_interface.h>
```

Public 成员函数

- [RobotInterface \(\)](#)
- virtual [~RobotInterface \(\)](#)
- [RobotConfigPtr getRobotConfig \(\)](#)
RobotConfig (机器人配置管理) 获取*RobotConfig* 接口
- [MotionControlPtr getMotionControl \(\)](#)
MotionControl (运动规划与控制) 获取运动规划接口
- [ForceControlPtr getForceControl \(\)](#)
ForceControl (力控模块) 获取力控接口
- [IoControlPtr getIoControl \(\)](#)
IoControl (IO 输入输出控制) 获取IO 控制的接口
- [SyncMovePtr getSyncMove \(\)](#)
SyncMove (同步运动控制和轴组管理) 获取同步运动接口
- [RobotAlgorithmPtr getRobotAlgorithm \(\)](#)
RobotAlgorithm (机器人算法工具) 获取机器人实用算法接口
- [RobotManagePtr getRobotManage \(\)](#)
RobotManage (机器人生命周期管理) 获取机器人管理接口 (上电、启动、停止等)
- [RobotStatePtr getRobotState \(\)](#)
RobotState (机器人状态查询) 获取机器人状态接口
- [TracePtr getTrace \(\)](#)
Trace (日志与弹窗) 获取告警信息接口

Protected 属性

- void * [d_](#)

11.15.1 详细描述

在文件 [robot_interface.h](#) 第 25 行定义.

11.15.2 构造及析构造函数说明

RobotInterface()

```
arcs::common_interface::RobotInterface::RobotInterface ()
```

~RobotInterface()

```
virtual arcs::common_interface::RobotInterface::~~RobotInterface () [virtual]
```

11.15.3 类成员变量说明

d_

```
void* arcs::common_interface::RobotInterface::d_ [protected]
```

在文件 [robot_interface.h](#) 第 389 行定义.

该类的文档由以下文件生成:

- include/aubo/[robot_interface.h](#)

11.16 arcs::common_interface::RobotManage 类参考

```
#include <robot_manage.h>
```

Public 成员函数

- [RobotManage \(\)](#)
- virtual [~RobotManage \(\)](#)
- int [poweron \(\)](#)
发起机器人上电请求
- int [startup \(\)](#)
发起机器人启动请求
- int [releaseRobotBrake \(\)](#)
发起机器人松开刹车请求
- int [lockRobotBrake \(\)](#)
发起机器人刹车请求
- int [poweroff \(\)](#)
发起机器人断电请求
- int [backdrive](#) (bool enable)
发起机器人反向驱动请求
- int [freedrive](#) (bool enable)
发起机器人自由驱动请求接口在软件版本 0.31.x 后已废弃, 使用 [handguideMode](#) 接口替换 [handguideMode\({1,1,1,1,1}, {0,0,0,0,0,0}\)](#)
- int [setHandguideParams](#) (const std::vector< int > &freeAxes, const std::vector< double > &feature)
设置拖动示教参数
- std::vector< int > [getHandguideFreeAxes \(\)](#)
获取拖动轴
- std::vector< double > [getHandguideFeature \(\)](#)
获取拖动参考坐标系
- int [handguideMode](#) (const std::vector< int > &freeAxes, const std::vector< double > &feature)
高阶拖动示教
- int [exitHandguideMode \(\)](#)

- 退出拖动示教
- int [getHandguideStatus](#) ()
获取拖动示教器的状态 (是否处于奇异空间)
- int [getHandguideTrigger](#) ()
获取拖动示教器触发源
- bool [isHandguideEnabled](#) ()
获取拖动示教使能状态
- int [setSim](#) (bool enable)
发起机器人进入/退出仿真模式请求
- int [setOperationalMode](#) (OperationalModeType mode)
设置机器人操作模式
- OperationalModeType [getOperationalMode](#) ()
获取机器人操作模式
- RobotControlModeType [getRobotControlMode](#) ()
获取控制模式
- bool [isFreedriveEnabled](#) ()
是否使能了拖动示教模式
- bool [isBackdriveEnabled](#) ()
是否使能了反向驱动模式
- bool [isSimulationEnabled](#) ()
是否使能了仿真模式
- int [setUnlockProtectiveStop](#) ()
清除防护停机, 包括碰撞停机
- int [restartInterfaceBoard](#) ()
重置安全接口板, 一般在机器人断电之后需要重置时调用, 比如机器人急停、故障等之后
- int [recordCacheFree](#) (const std::string &name)
释放并清空指定内存缓存的记录数据
- int [startRecordCache](#) (const std::string &name)
开始实时轨迹的内存缓存记录 (不落盘)
- int [stopRecordCache](#) ()
停止当前实时轨迹内存缓存记录
- int [pauseRecordCache](#) (bool pause)
暂停/恢复当前实时轨迹内存缓存记录
- int [getRecordCache](#) (const std::string &name, size_t frames=0)
获取指定内存缓存的记录数据
- int [startRecord](#) (const std::string &file_name)
开始实时轨迹的记录
- int [stopRecord](#) ()
停止实时记录
- int [pauseRecord](#) (bool pause)
暂停实时记录
- int [setLinkModeEnable](#) (bool enable)
发起机器人进入/退出联动模式请求, 只有操作模式为自动或者无时, 才能使能联动模式
- bool [isLinkModeEnabled](#) ()
是否使能了联动模式, 联动模式下用户可以通过外部IO 控制机器人 (用户可以对IO 的功能进行配置)
- int [generateDiagnoseFile](#) (const std::string &reason)
手动触发生成诊断文件

Protected 属性

- void * [d__](#)

11.16.1 详细描述

在文件 [robot_manage.h](#) 第 21 行定义.

11.16.2 构造及析构函数说明

RobotManage()

```
arcs::common_interface::RobotManage::RobotManage ()
```

~RobotManage()

```
virtual arcs::common_interface::RobotManage::~~RobotManage () [virtual]
```

11.16.3 类成员变量说明

d_

```
void* arcs::common_interface::RobotManage::d_ [protected]
```

在文件 [robot_manage.h](#) 第 1915 行定义.

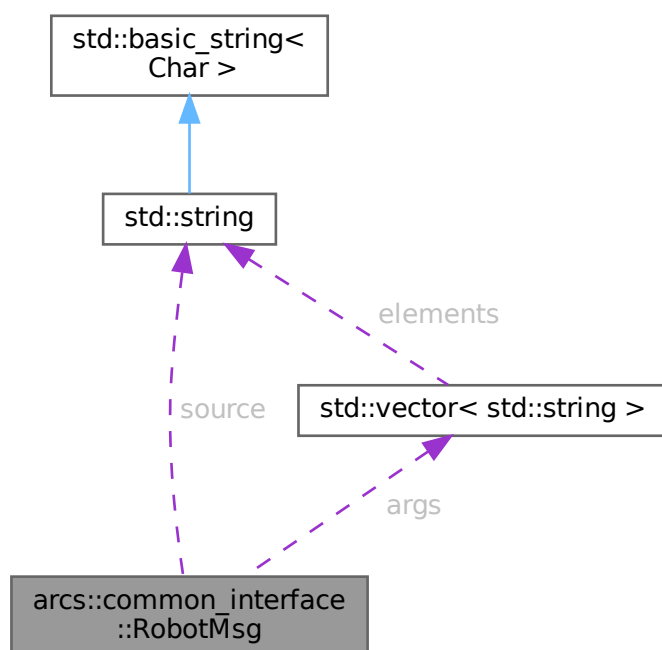
该类的文档由以下文件生成:

- [include/aubo/robot/robot_manage.h](#)

11.17 arcs::common_interface::RobotMsg 结构体参考

```
#include <type_def.h>
```

arcs::common_interface::RobotMsg 的协作图:



Public 属性

- uint64_t [timestamp](#)
时间戳，即系统时间
- [TraceLevel](#) [level](#)
日志等级
- int [code](#)
错误码
- std::string [source](#)
发送消息的机器人别名 *alias* 可在 `/root/arcs_ws/config/aubo_control.conf` 配置文件中查到机器人的 *alias*
- std::vector< std::string > [args](#)
机器人参数

11.17.1 详细描述

在文件 [type_def.h](#) 第 855 行定义.

11.17.2 类成员变量说明**args**

```
std::vector<std::string> arcs::common_interface::RobotMsg::args
```

机器人参数
在文件 [type_def.h](#) 第 870 行定义.

code

```
int arcs::common_interface::RobotMsg::code
```

错误码
在文件 [type_def.h](#) 第 860 行定义.

level

```
TraceLevel arcs::common_interface::RobotMsg::level
```

日志等级
在文件 [type_def.h](#) 第 859 行定义.

source

```
std::string arcs::common_interface::RobotMsg::source
```

发送消息的机器人别名 *alias* 可在 `/root/arcs_ws/config/aubo_control.conf` 配置文件中查到机器人的 *alias*
在文件 [type_def.h](#) 第 862 行定义.

timestamp

```
uint64_t arcs::common_interface::RobotMsg::timestamp
```

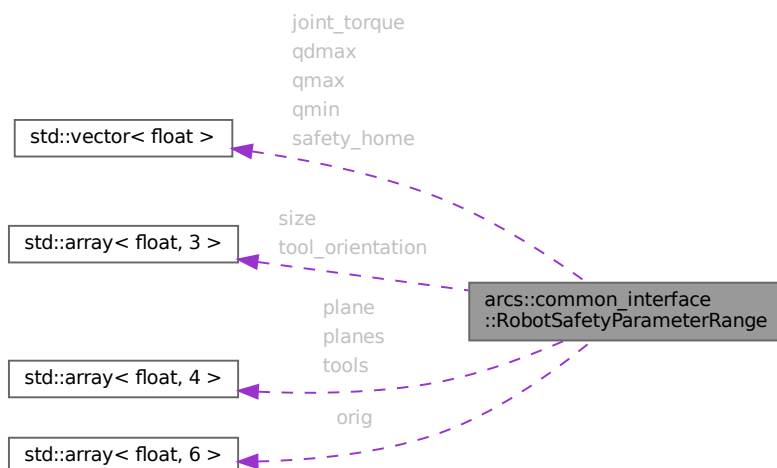
时间戳，即系统时间
在文件 [type_def.h](#) 第 857 行定义.
该结构体的文档由以下文件生成:

- `include/aubo/type_def.h`

11.18 arcs::common_interface::RobotSafetyParameterRange 结构体参考

#include <type_def.h>

arcs::common_interface::RobotSafetyParameterRange 的协作图:



Public 成员函数

- [RobotSafetyParameterRange \(\)](#)

Public 属性

- uint32_t [crc32](#) { 0 }
- struct {
 - float [power](#)
关节力矩与关节角速度的乘积之和
 - float [momentum](#)
机器人动量限制
 - float [stop_time](#)
停机时间 *ms*
 - float [stop_distance](#)
停机距离 *m*
 - float [reduced_entry_time](#)
进入缩减模式的最大时间
 - float [reduced_entry_distance](#)
进入缩减模式的最大距离 (可由安全平面触发)
 - float [tcp_speed](#)
 - float [elbow_speed](#)
 - float [tcp_force](#)
 - float [elbow_force](#)
 - std::vector< float > [qmin](#)
 - std::vector< float > [qmax](#)
 - std::vector< float > [qdmx](#)
 - std::vector< float > [joint_torque](#)
 - Vector3f [tool_orientation](#)
 - float [tool_deviation](#)
 - Vector4f [planes](#) [SAFETY_PLANES_NUM]

```
int restrict_elbow [SAFETY_PLANES_NUM]
    x,y,z,displacement
} params [SAFETY_PARAM_SELECT_NUM]
```

最多可以保存 2 套参数, 默认使用第 0 套参数

- struct {
 - Vector4f plane
 - int restrict_elbow
 - x,y,z,displacement
- } trigger_planes [SAFETY_PLANES_NUM]

8 个触发平面

- struct {
 - Vector6f orig
 - 立方块的原点 (x,y,z,rx,ry,rz)
 - Vector3f size
 - 立方块的尺寸 (x,y,z)
 - int restrict_elbow
- } cubic [SAFETY_CUBIC_NUM]

10 个安全空间

- Vector4f tools [TOOL_CONFIGURATION_NUM]
 - 3 个工具
- float tool_inclination
 - x,y,z,radius
- float tool_azimuth { 0. }
 - 方位角
- std::vector< float > safety_home
- uint32_t safety_input_emergency_stop
 - 可配置IO 的输入输出安全功能配置
- uint32_t safety_input_safeguard_stop
- uint32_t safety_input_safeguard_reset
- uint32_t safety_input_auto_safeguard_stop
- uint32_t safety_input_auto_safeguard_reset
- uint32_t safety_input_three_position_switch
- uint32_t safety_input_operational_mode
- uint32_t safety_input_reduced_mode
- uint32_t safety_input_handguide
- uint32_t safety_output_emergency_stop
- uint32_t safety_output_not_emergency_stop
- uint32_t safety_output_robot_moving
- uint32_t safety_output_robot_steady
- uint32_t safety_output_reduced_mode
- uint32_t safety_output_not_reduced_mode
- uint32_t safety_output_safe_home
- uint32_t safety_output_robot_not_stopping
- uint32_t safety_output_safetyguard_stop
- int tp_3pe_for_handguide
 - 是否将示教器三档位开关作为拖动功能开关
- int allow_manual_high_speed
 - 手动模式下允许高速运行 /~english Allow high-speed operation in manual mode

11.18.1 详细描述

在文件 `type_def.h` 第 37 行定义.

11.18.2 构造及析构函数说明

RobotSafetyParameterRange()

`arcs::common_interface::RobotSafetyParameterRange::RobotSafetyParameterRange () [inline]`

在文件 `type_def.h` 第 39 行定义.

引用了 `allow_manual_high_speed`, `cubic`, `joint_torque`, `params`, `qddmax`, `qmax`, `qmin`, `SAFETY_CUBIC_NUM`, `safety_home`, `safety_input_auto_safeguard_reset`, `safety_input_auto_safeguard_stop`, `safety_input_emergency_stop`, `safety_input_handguide`, `safety_input_operational_mode`, `safety_input_reduced_mode`, `safety_input_safeguard_reset`, `safety_input_safeguard_stop`, `safety_input_three_position_switch`, `safety_output_emergency_stop`, `safety_output_not_emergency_stop`, `safety_output_not_reduced_mode`, `safety_output_reduced_mode`, `safety_output_robot_moving`, `safety_output_robot_not_stopping`, `safety_output_robot_steady`, `safety_output_safeguard_stop`, `SAFETY_PARAM_SELECT_NUM`, `SAFETY_PLANES_NUM`, `tool_azimuth`, `TOOL_CONFIGURATION_NUM`, `tool_inclination`, `tools`, `tp_3pe_for_handguide`, 以及 `trigger_planes`.

11.18.3 类成员变量说明

`allow_manual_high_speed`

`int arcs::common_interface::RobotSafetyParameterRange::allow_manual_high_speed`

手动模式下允许高速运行 /~english Allow high-speed operation in manual mode

在文件 `type_def.h` 第 177 行定义.

被这些函数引用 `RobotSafetyParameterRange()`.

`crc32`

`uint32_t arcs::common_interface::RobotSafetyParameterRange::crc32 { 0 }`

在文件 `type_def.h` 第 105 行定义.

[struct]

`struct { ... } arcs::common_interface::RobotSafetyParameterRange::cubic[SAFETY_CUBIC_NUM]`

10 个安全空间

被这些函数引用 `RobotSafetyParameterRange()`.

`elbow_force`

`float arcs::common_interface::RobotSafetyParameterRange::elbow_force`

在文件 `type_def.h` 第 120 行定义.

`elbow_speed`

`float arcs::common_interface::RobotSafetyParameterRange::elbow_speed`

在文件 `type_def.h` 第 118 行定义.

`joint_torque`

`std::vector<float> arcs::common_interface::RobotSafetyParameterRange::joint_torque`

在文件 `type_def.h` 第 124 行定义.

被这些函数引用 `RobotSafetyParameterRange()`.

`momentum`

`float arcs::common_interface::RobotSafetyParameterRange::momentum`

机器人动量限制

在文件 `type_def.h` 第 112 行定义.

`orig`

`Vector6f arcs::common_interface::RobotSafetyParameterRange::orig`

立方块的原点 (x,y,z,rx,ry,rz)

在文件 `type_def.h` 第 140 行定义.

[struct]

```
struct { ... } arcs::common_interface::RobotSafetyParameterRange::params[SAFETY_PARAM_SELECT_NUM]
```

最多可以保存 2 套参数, 默认使用第 0 套参数
被这些函数引用 [RobotSafetyParameterRange\(\)](#).

plane

```
Vector4f arcs::common_interface::RobotSafetyParameterRange::plane
```

在文件 [type_def.h](#) 第 134 行定义.

planes

```
Vector4f arcs::common_interface::RobotSafetyParameterRange::planes[SAFETY_PLANES_NUM]
```

在文件 [type_def.h](#) 第 127 行定义.

power

```
float arcs::common_interface::RobotSafetyParameterRange::power
```

关节力矩与关节角速度的乘积之和
在文件 [type_def.h](#) 第 111 行定义.

qdmx

```
std::vector<float> arcs::common_interface::RobotSafetyParameterRange::qdmx
```

在文件 [type_def.h](#) 第 123 行定义.
被这些函数引用 [RobotSafetyParameterRange\(\)](#).

qmax

```
std::vector<float> arcs::common_interface::RobotSafetyParameterRange::qmax
```

在文件 [type_def.h](#) 第 122 行定义.
被这些函数引用 [RobotSafetyParameterRange\(\)](#).

qmin

```
std::vector<float> arcs::common_interface::RobotSafetyParameterRange::qmin
```

在文件 [type_def.h](#) 第 121 行定义.
被这些函数引用 [RobotSafetyParameterRange\(\)](#).

reduced_entry_distance

```
float arcs::common_interface::RobotSafetyParameterRange::reduced_entry_distance
```

进入缩减模式的最大距离 (可由安全平面触发)
在文件 [type_def.h](#) 第 116 行定义.

reduced_entry_time

```
float arcs::common_interface::RobotSafetyParameterRange::reduced_entry_time
```

进入缩减模式的最大时间
在文件 [type_def.h](#) 第 115 行定义.

restrict_elbow

```
int arcs::common_interface::RobotSafetyParameterRange::restrict_elbow
```

x,y,z,displacement
在文件 [type_def.h](#) 第 128 行定义.

safety_home

`std::vector<float> arcs::common_interface::RobotSafetyParameterRange::safety_home`

在文件 `type_def.h` 第 152 行定义.

被这些函数引用 `RobotSafetyParameterRange()`.

safety_input_auto_safeguard_reset

`uint32_t arcs::common_interface::RobotSafetyParameterRange::safety_input_auto_safeguard_reset`

在文件 `type_def.h` 第 160 行定义.

被这些函数引用 `RobotSafetyParameterRange()`.

safety_input_auto_safeguard_stop

`uint32_t arcs::common_interface::RobotSafetyParameterRange::safety_input_auto_safeguard_stop`

在文件 `type_def.h` 第 159 行定义.

被这些函数引用 `RobotSafetyParameterRange()`.

safety_input_emergency_stop

`uint32_t arcs::common_interface::RobotSafetyParameterRange::safety_input_emergency_stop`

可配置IO 的输入输出安全功能配置

在文件 `type_def.h` 第 156 行定义.

被这些函数引用 `RobotSafetyParameterRange()`.

safety_input_handguide

`uint32_t arcs::common_interface::RobotSafetyParameterRange::safety_input_handguide`

在文件 `type_def.h` 第 164 行定义.

被这些函数引用 `RobotSafetyParameterRange()`.

safety_input_operational_mode

`uint32_t arcs::common_interface::RobotSafetyParameterRange::safety_input_operational_mode`

在文件 `type_def.h` 第 162 行定义.

被这些函数引用 `RobotSafetyParameterRange()`.

safety_input_reduced_mode

`uint32_t arcs::common_interface::RobotSafetyParameterRange::safety_input_reduced_mode`

在文件 `type_def.h` 第 163 行定义.

被这些函数引用 `RobotSafetyParameterRange()`.

safety_input_safeguard_reset

`uint32_t arcs::common_interface::RobotSafetyParameterRange::safety_input_safeguard_reset`

在文件 `type_def.h` 第 158 行定义.

被这些函数引用 `RobotSafetyParameterRange()`.

safety_input_safeguard_stop

`uint32_t arcs::common_interface::RobotSafetyParameterRange::safety_input_safeguard_stop`

在文件 `type_def.h` 第 157 行定义.

被这些函数引用 `RobotSafetyParameterRange()`.

safety_input_three_position_switch

`uint32_t arcs::common_interface::RobotSafetyParameterRange::safety_input_three_position_switch`

在文件 `type_def.h` 第 161 行定义.

被这些函数引用 `RobotSafetyParameterRange()`.

safety_output_emergency_stop

uint32_t arcs::common_interface::RobotSafetyParameterRange::safety_output_emergency_stop
在文件 [type_def.h](#) 第 166 行定义.
被这些函数引用 [RobotSafetyParameterRange\(\)](#).

safety_output_not_emergency_stop

uint32_t arcs::common_interface::RobotSafetyParameterRange::safety_output_not_emergency_stop
在文件 [type_def.h](#) 第 167 行定义.
被这些函数引用 [RobotSafetyParameterRange\(\)](#).

safety_output_not_reduced_mode

uint32_t arcs::common_interface::RobotSafetyParameterRange::safety_output_not_reduced_mode
在文件 [type_def.h](#) 第 171 行定义.
被这些函数引用 [RobotSafetyParameterRange\(\)](#).

safety_output_reduced_mode

uint32_t arcs::common_interface::RobotSafetyParameterRange::safety_output_reduced_mode
在文件 [type_def.h](#) 第 170 行定义.
被这些函数引用 [RobotSafetyParameterRange\(\)](#).

safety_output_robot_moving

uint32_t arcs::common_interface::RobotSafetyParameterRange::safety_output_robot_moving
在文件 [type_def.h](#) 第 168 行定义.
被这些函数引用 [RobotSafetyParameterRange\(\)](#).

safety_output_robot_not_stopping

uint32_t arcs::common_interface::RobotSafetyParameterRange::safety_output_robot_not_stopping
在文件 [type_def.h](#) 第 173 行定义.
被这些函数引用 [RobotSafetyParameterRange\(\)](#).

safety_output_robot_steady

uint32_t arcs::common_interface::RobotSafetyParameterRange::safety_output_robot_steady
在文件 [type_def.h](#) 第 169 行定义.
被这些函数引用 [RobotSafetyParameterRange\(\)](#).

safety_output_safe_home

uint32_t arcs::common_interface::RobotSafetyParameterRange::safety_output_safe_home
在文件 [type_def.h](#) 第 172 行定义.
被这些函数引用 [RobotSafetyParameterRange\(\)](#).

safety_output_safeguard_stop

uint32_t arcs::common_interface::RobotSafetyParameterRange::safety_output_safeguard_stop
在文件 [type_def.h](#) 第 174 行定义.
被这些函数引用 [RobotSafetyParameterRange\(\)](#).

size

[Vector3f](#) arcs::common_interface::RobotSafetyParameterRange::size
立方块的尺寸 (x,y,z)
在文件 [type_def.h](#) 第 141 行定义.

stop_distance

`float arcs::common_interface::RobotSafetyParameterRange::stop_distance`
 停机距离 m
 在文件 [type_def.h](#) 第 114 行定义.

stop_time

`float arcs::common_interface::RobotSafetyParameterRange::stop_time`
 停机时间 ms
 在文件 [type_def.h](#) 第 113 行定义.

tcp_force

`float arcs::common_interface::RobotSafetyParameterRange::tcp_force`
 在文件 [type_def.h](#) 第 119 行定义.

tcp_speed

`float arcs::common_interface::RobotSafetyParameterRange::tcp_speed`
 在文件 [type_def.h](#) 第 117 行定义.

tool_azimuth

`float arcs::common_interface::RobotSafetyParameterRange::tool_azimuth { 0. }`
 方位角
 在文件 [type_def.h](#) 第 151 行定义.
 被这些函数引用 [RobotSafetyParameterRange\(\)](#).

tool_deviation

`float arcs::common_interface::RobotSafetyParameterRange::tool_deviation`
 在文件 [type_def.h](#) 第 126 行定义.

tool_inclination

`float arcs::common_interface::RobotSafetyParameterRange::tool_inclination`
初始值:

```
{
    0.
}
```

`x,y,z,radius`
 倾角
 在文件 [type_def.h](#) 第 148 行定义.
 被这些函数引用 [RobotSafetyParameterRange\(\)](#).

tool_orientation

`Vector3f arcs::common_interface::RobotSafetyParameterRange::tool_orientation`
 在文件 [type_def.h](#) 第 125 行定义.

tools

`Vector4f arcs::common_interface::RobotSafetyParameterRange::tools[TOOL_CONFIGURATION_NUM]`
 3 个工具
 在文件 [type_def.h](#) 第 146 行定义.
 被这些函数引用 [RobotSafetyParameterRange\(\)](#).

tp_3pe_for_handguide

`int arcs::common_interface::RobotSafetyParameterRange::tp_3pe_for_handguide`
 是否将示教器三档位开关作为拖动功能开关
 在文件 `type_def.h` 第 176 行定义.
 被这些函数引用 `RobotSafetyParameterRange()`.

[struct]

`struct { ... } arcs::common_interface::RobotSafetyParameterRange::trigger_planes[SAFETY_PLANES_NUM]`
 8 个触发平面
 被这些函数引用 `RobotSafetyParameterRange()`.
 该结构体的文档由以下文件生成:

- `include/aubo/type_def.h`

11.19 arcs::common_interface::RobotState 类参考

```
#include <robot_state.h>
```

Public 成员函数

- `RobotState ()`
- `virtual ~RobotState ()`
- `RobotModeType getRobotModeType ()`
 获取机器人的模式状态
- `SafetyModeType getSafetyModeType ()`
 获取安全模式
- `bool isPowerOn ()`
 获取机器人通电状态
- `bool isSteady ()`
 机器人是否已经停止下来
- `bool isCollisionOccurred ()`
 机器人是否发生了碰撞
- `bool isWithinSafetyLimits ()`
 机器人是否已经在安全限制之内
- `std::vector< double > getTcpPose ()`
 获取当前的 TCP 位姿, 其 TCP 偏移可以通过 `getActualTcpOffset` 获取
- `std::vector< double > getActualTcpOffset ()`
 获取当前的 TCP 偏移, 也就是 `getTcpPose` 返回的 `pose` 用到的 TCP 偏移
- `std::vector< double > getTargetTcpPose ()`
 获取下一个目标路点注意与 `getTcpTargetPose` 的区别, 此处定义存在歧义, 命名需要优化
- `std::vector< double > getToolPose ()`
 获取工具端的位姿 (不带 TCP 偏移)
- `std::vector< double > getTcpSpeed ()`
 获取 TCP 速度
- `std::vector< double > getTcpForce ()`
 获取 TCP 的力/力矩
- `std::vector< double > getElbowPosistion ()`
 获取肘部的位置
- `std::vector< double > getElbowVelocity ()`
 获取肘部速度
- `std::vector< double > getBaseForce ()`
 获取基座力/力矩

- `std::vector< double > getTcpTargetPose ()`
获取上一次发送的 *TCP* 目标位姿
- `std::vector< double > getTcpTargetSpeed ()`
获取 *TCP* 目标速度
- `std::vector< double > getTcpTargetForce ()`
获取 *TCP* 目标力/力矩
- `std::vector< JointStateType > getJointState ()`
获取机械臂关节标志
- `std::vector< JointServoModeType > getJointServoMode ()`
获取关节的伺服状态
- `std::vector< double > getJointPositions ()`
获取机械臂关节角度
- `std::vector< double > getJointPositionsHistory (int steps)`
获取机械臂历史关节角度
- `std::vector< double > getJointSpeeds ()`
获取机械臂关节速度
- `std::vector< double > getJointAccelerations ()`
获取机械臂关节加速度
- `std::vector< double > getJointTorqueSensors ()`
获取机械臂关节力矩
- `std::vector< double > getJointContactTorques ()`
获取机械臂关节接触力矩 (外力矩)
- `std::vector< double > getJointGravityTorques ()`
获取机械臂关节重力矩
- `std::vector< double > getBaseForceSensor ()`
获取底座力传感器读数
- `std::vector< double > getTcpForceSensors ()`
获取 *TCP* 力传感器读数
- `std::vector< double > getJointCurrents ()`
获取机械臂关节电流
- `std::vector< double > getJointVoltages ()`
获取机械臂关节电压
- `std::vector< double > getJointTemperatures ()`
获取机械臂关节温度
- `std::vector< std::string > getJointUniqueIds ()`
获取关节全球唯一 *ID*
- `std::vector< int > getJointFirmwareVersions ()`
获取关节固件版本
- `std::vector< int > getJointHardwareVersions ()`
获取关节硬件版本
- `std::string getMasterBoardUniqueId ()`
获取 *MasterBoard* 全球唯一 *ID*
- `int getMasterBoardFirmwareVersion ()`
获取 *MasterBoard* 固件版本
- `int getMasterBoardHardwareVersion ()`
获取 *MasterBoard* 硬件版本
- `std::string getSlaveBoardUniqueId ()`
获取 *SlaveBoard* 全球唯一 *ID*
- `int getSlaveBoardFirmwareVersion ()`
获取 *SlaveBoard* 固件版本
- `int getSlaveBoardHardwareVersion ()`

- 获取*SlaveBoard* 硬件版本
- std::string [getHandleUniqueId](#) ()
 - 获取*Handle* 全球唯一*ID*
- int [getHandleFirmwareVersion](#) ()
 - 获取*Handle* 固件版本
- int [getHandleHardwareVersion](#) ()
 - 获取*Handle* 硬件版本
- [HandleStateType](#) [getHandleState](#) ()
 - 获取*Handle* 状态
- [HandleModeType](#) [getHandleMode](#) ()
 - 获取*Handle* 类型
- std::string [getPowerlossBoardUniqueId](#) ()
 - 获取*PowerlossBoard* 全球唯一*ID*
- int [getPowerlossBoardFirmwareVersion](#) ()
 - 获取*PowerlossBoard* 固件版本
- int [getPowerlossBoardHardwareVersion](#) ()
 - 获取*PowerlossBoard* 硬件版本
- std::string [getToolUniqueId](#) ()
 - 获取工具端全球唯一*ID*
- int [getToolFirmwareVersion](#) ()
 - 获取工具端固件版本
- int [getToolHardwareVersion](#) ()
 - 获取工具端硬件版本
- int [getToolCommMode](#) ()
 - 获取末端通信模式
- std::string [getPedestalUniqueId](#) ()
 - 获取底座全球唯一*ID*
- int [getPedestalFirmwareVersion](#) ()
 - 获取底座固件版本
- int [getPedestalHardwareVersion](#) ()
 - 获取底座硬件版本
- std::vector< double > [getJointTargetPositions](#) ()
 - 获取机械臂关节目标位置角度
- std::vector< double > [getJointTargetSpeeds](#) ()
 - 获取机械臂关节目标速度
- std::vector< double > [getJointTargetAccelerations](#) ()
 - 获取机械臂关节目标加速度
- std::vector< double > [getJointTargetTorques](#) ()
 - 获取机械臂关节目标力矩
- std::vector< double > [getJointTargetCurrents](#) ()
 - 获取机械臂关节目标电流
- bool [isTeachPendantEnabled](#) ()
 - 获取示教器是否已启用指示教器使能按钮是否处于按下状态
- bool [isToolFlangeEnabled](#) ()
 - 获取机械臂末端是否已启用
- double [getControlBoxTemperature](#) ()
 - 获取控制柜温度
- double [getControlBoxHumidity](#) ()
 - 获取控制柜湿度
- double [getMainVoltage](#) ()
 - 获取母线电压

- double `getMainCurrent ()`
获取母线电流
- double `getRobotVoltage ()`
获取机器人电压
- double `getRobotCurrent ()`
获取机器人电流
- double `getPedestalBrakeResTemp ()`
获取底座制动电阻温度
- int `getSlowDownLevel ()`
获取机器人缓速等级
- bool `getTcpForceSensorStatus (const std::string &name)`
获取末端力传感器通信状态

Protected 属性

- void * `d_`

11.19.1 详细描述

在文件 `robot_state.h` 第 20 行定义.

11.19.2 构造及析构函数说明

RobotState()

```
arcs::common_interface::RobotState::RobotState ()
```

~RobotState()

```
virtual arcs::common_interface::RobotState::~~RobotState () [virtual]
```

11.19.3 成员函数说明

getBaseForceSensor()

```
std::vector< double > arcs::common_interface::RobotState::getBaseForceSensor ()
```

获取底座力传感器读数

返回

底座力传感器读数

异常

<code>arcs::common_interface::AuboException</code>	

Python 函数原型

```
getBaseForceSensor(self: pyaubo_sdk.RobotState) -> List[float]
```

Lua 函数原型

```
getBaseForceSensor() -> table
```

Lua 示例

```
BaseForceSensor = getBaseForceSensor()
```

JSON-RPC 请求示例

```
{"jsonrpc":"2.0","method":"rob1.RobotState.getBaseForceSensor","params":[],"id":1}
```

JSON-RPC 响应示例

```
{"id":1,"jsonrpc":"2.0","result":[0.0,0.0,0.0,0.0,0.0,0.0]}
```

11.19.4 类成员变量说明

d__

```
void* arcs::common_interface::RobotState::d_ [protected]
```

在文件 [robot_state.h](#) 第 3508 行定义.

该类的文档由以下文件生成:

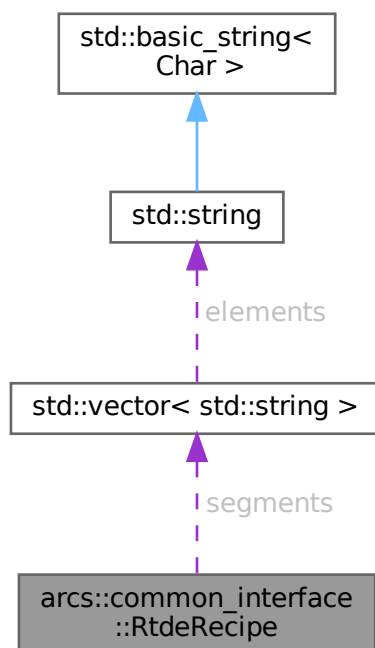
- [include/aubo/robot/robot_state.h](#)

11.20 arcs::common_interface::RtdeRecipe 结构体参考

RTDE 菜单

```
#include <type_def.h>
```

arcs::common_interface::RtdeRecipe 的协作图:



Public 属性

- bool `to_server`
输入/输出
- int `chanel`
通道
- double `frequency`
更新频率
- int `trigger`
触发方式 (该功能暂未实现): 0 - 周期; 1 - 变化
- `std::vector< std::string >` `segments`
字段列表

11.20.1 详细描述

RTDE 菜单

在文件 `type_def.h` 第 894 行定义.

11.20.2 类成员变量说明

chanel

```
int arcs::common_interface::RtdeRecipe::chanel
```

通道

在文件 `type_def.h` 第 897 行定义.

frequency

```
double arcs::common_interface::RtdeRecipe::frequency
```

更新频率

在文件 `type_def.h` 第 898 行定义.

segments

```
std::vector<std::string> arcs::common_interface::RtdeRecipe::segments
```

字段列表

在文件 `type_def.h` 第 902 行定义.

to_server

```
bool arcs::common_interface::RtdeRecipe::to_server
```

输入/输出

在文件 `type_def.h` 第 896 行定义.

trigger

```
int arcs::common_interface::RtdeRecipe::trigger
```

触发方式 (该功能暂未实现): 0 - 周期; 1 - 变化

在文件 `type_def.h` 第 899 行定义.

该结构体的文档由以下文件生成:

- `include/aubo/type_def.h`

11.21 arcs::common_interface::RuntimeMachine 类参考

```
#include <runtime_machine.h>
```


Public 成员函数

- [RuntimeMachine](#) ()
- virtual [~RuntimeMachine](#) ()
- int [newTask](#) (bool daemon=false)
返回 *task_id*
- int [deleteTask](#) (int tid)
删除 *task*, 会终止正在执行的运动
- int [detachTask](#) (int tid)
等待 *task* 自然结束
- bool [isTaskAlive](#) (int tid)
判断任务是否存活
- int [getTaskQueueSize](#) (int tid)
获取任务中缓存的指令的数量
- int [switchTask](#) (int tid)
切换当前线程, 切换之后接下来的指令将被插入切换后的线程中
- int [setLabel](#) (int lineno, const std::string &comment)
标记记下来的指令的行号和注释
- ARCS_DEPRECATED int [setPlanContext](#) (int tid, int lineno, const std::string &comment)
向 *aubo_control* 日志中添加注释使用 *setLabel* 替换
- int [nop](#) ()
空操作
- std::tuple< std::string, std::string > [getExecutionStatus](#) ()
获取耗时的接口 (*INST*) 执行状态, 如 *setPersistentParameters*
- std::tuple< std::string, std::string, int > [getExecutionStatus1](#) ()
- int [gotoLine](#) (int lineno)
跳转到指定行号
- std::tuple< int, int, std::string > [getPlanContext](#) (int tid=-1)
获取当前运行上下文
- std::tuple< int, int, std::string > [getAdvancePlanContext](#) (int tid=-1)
获取提前运行规划器的上下文信息
- int [getAdvancePtr](#) (int tid=-1)
获取 *AdvanceRun* 的程序指针
- int [getMainPtr](#) (int tid=-1)
获取机器人运动的程序指针
- int [getInterpPtr](#) (int tid)
获取最近解释过的指令指针
- int [loadProgram](#) (const std::string &program)
加载本地工程文件 *Lua* 脚本, 只需要给出文件名字, 不需要后缀, 需要从 *\${ARCS_WS}/program* 目录中查找
- int [preloadProgram](#) (int index, const std::string &program)
预加载工程文件
- std::string [getPreloadProgram](#) (int index)
获取预加载工程文件名字, 如果没有加载或者超出索引范围则返回空字符串
- int [clearPreloadPrograms](#) ()
清除所有已预加载的工程文件调用此方法将释放所有通过 *preloadProgram* 预加载的工程索引及其关联的工程名称
- int [runProgram](#) ()
运行已经加载的工程文件
- int [start](#) ()
开始运行时
- int [stop](#) ()

- 停止运行时即脚本运行, 无法停止运行时状态为 *Stopped* 时的机器人运动
- int `abort` ()
终止机器人运行.
 - int `pause` ()
暂停解释器
 - int `step` ()
单步运行
 - int `resume` ()
恢复解释器
 - int `arbitraryResume` ()
恢复解释器 (不检查当前点和暂停点距离)
 - int `setResumeWait` (bool wait)
恢复解释器之前等待恢复前之前的序列完成
 - int `enterCritical` (double timeout)
进入临界区, `abort` 命令会被推迟执行, 避免临界区内的指令被打断
 - int `exitCritical` ()
退出临界区
 - ARCS_DEPRECATED `RuntimeState getStatus` ()
获取规划器的状态
 - `RuntimeState getRuntimeState` ()
 - int `setBreakPoint` (int lineno)
设置断点
 - int `removeBreakPoint` (int lineno)
移除断点
 - int `clearBreakPoints` ()
清除所有断点
 - int `timerStart` (const std::string &name)
定时器开始
 - int `timerStop` (const std::string &name)
定时器结束
 - int `timerReset` (const std::string &name)
定时器重置
 - int `timerDelete` (const std::string &name)
定时器删除
 - double `getTimer` (const std::string &name)
获取定时器数值
 - int `triggBegin` (double distance, double delay)
开始配置触发
 - int `triggEnd` ()
终止配置触发
 - int `triggInterrupt` (double distance, double delay)
返回自动分配的中断号
 - std::vector< int > `getTriggInterrupts` ()
获取所有的中断号列表

Protected 属性

- void * `d__`

11.21.1 详细描述

在文件 `runtime_machine.h` 第 19 行定义.

11.21.2 构造及析构函数说明

RuntimeMachine()

```
arcs::common_interface::RuntimeMachine::RuntimeMachine ()
```

~RuntimeMachine()

```
virtual arcs::common_interface::RuntimeMachine::~~RuntimeMachine () [virtual]
```

11.21.3 成员函数说明

getExecutionStatus1()

```
std::tuple< std::string, std::string, int > arcs::common_interface::RuntimeMachine::getExecutionStatus1 ()
```

getRuntimeState()

```
RuntimeState arcs::common_interface::RuntimeMachine::getRuntimeState ()
```

11.21.4 类成员变量说明

d_

```
void* arcs::common_interface::RuntimeMachine::d_ [protected]
```

在文件 [runtime_machine.h](#) 第 1617 行定义。

该类的文档由以下文件生成:

- [include/aubo/runtime_machine.h](#)

11.22 arcs::common_interface::Serial 类参考

```
#include <serial.h>
```

Public 成员函数

- [Serial \(\)](#)
- [virtual ~Serial \(\)](#)
- [int serialOpen](#) (const std::string &device, int baud, float stop_bits, int even, const std::string &serial_name="serial_0")
打开 TCP/IP 以太网通信串口
- [int serialClose](#) (const std::string &serial_name="serial_0")
关闭 TCP/IP 串口通信关闭与服务器的串口连接。
- [int serialReadByte](#) (const std::string &variable, const std::string &serial_name="serial_0")
从串口读取指定数量的字节。字节为网络字节序。一次最多可读取 30 个值。
- [int serialReadByteList](#) (int number, const std::string &variable, const std::string &serial_name="serial_0")
从串口读取指定数量的字节。字节为网络字节序。一次最多可读取 30 个值。返回读取到的数字列表 (*int* 列表, 长度 = *number+1*)。
- [int serialReadString](#) (const std::string &variable, const std::string &serial_name="serial_0", const std::string &prefix="", const std::string &suffix="", bool interpret_escape=false)
从串口读取所有数据, 并将数据作为字符串返回。字节为网络字节序。
- [int serialSendByte](#) (char value, const std::string &serial_name="serial_0")
发送一个字节到服务器通过串口发送字节
。不期望有响应。可用于发送特殊的 ASCII 字符; 10 为换行符, 2 为文本开始, 3 为文本结束。
- [int serialSendInt](#) (int value, const std::string &serial_name="serial_0")
发送一个整数 (*int32_t*) 到服务器通过串口发送整数
。以网络字节序发送。不期望有响应。

- int `serialSendLine` (const std::string &str, const std::string &serial_name="serial_0")
发送带有换行符的字符串到服务器以 *ASCII* 编码通过串口发送字符串 <str>, 并在末尾添加换行符。不期望有响应。
- int `serialSendString` (const std::string &str, const std::string &serial_name="serial_0")
发送字符串到服务器以 *ASCII* 编码通过串口发送字符串 <str>。不期望有响应。
- int `serialSendAllString` (bool is_check, const std::vector< char > &str, const std::string &serial_name="serial_0")

Protected 属性

- void * `d_`

11.22.1 详细描述

在文件 `serial.h` 第 21 行定义.

11.22.2 构造及析构函数说明

Serial()

```
arcs::common_interface::Serial::Serial ()
```

~Serial()

```
virtual arcs::common_interface::Serial::~~Serial () [virtual]
```

11.22.3 类成员变量说明

d_

```
void* arcs::common_interface::Serial::d_ [protected]
```

在文件 `serial.h` 第 405 行定义.

该类的文档由以下文件生成:

- include/aubo/`serial.h`

11.23 arcs::common_interface::Socket 类参考

```
#include <socket.h>
```

Public 成员函数

- `Socket` ()
- virtual `~Socket` ()
- int `socketOpen` (const std::string &address, int port, const std::string &socket_name="socket_0")
打开 *TCP/IP* 以太网通信 *socket*
- int `socketClose` (const std::string &socket_name="socket_0")
关闭 *TCP/IP socket* 通信关闭与服务器的 *socket* 连接。
- int `socketReadAsciiFloat` (int number, const std::string &variable, const std::string &socket_name="socket_0")
从 *socket* 读取指定数量的 *ASCII* 格式浮点数。一次最多可读取 30 个值。读取到的数字列表 (浮点数列表, 长度 = *number+1*)
- int `socketReadBinaryInteger` (int number, const std::string &variable, const std::string &socket_name="socket_0")
从 *socket* 读取指定数量的 32 位整数。字节为网络字节序。一次最多可读取 30 个值。读取到的数字列表 (整数列表, 长度 = *number+1*)
- int `socketReadByteList` (int number, const std::string &variable, const std::string &socket_name="socket_0")

从 *socket* 读取指定数量的字节。字节为网络字节序。一次最多可读取 30 个值。读取到的数字列表（整数列表，长度 $=number+1$ ）

- int [socketReadString](#) (const std::string &variable, const std::string &socket_name="socket_0", const std::string &prefix="", const std::string &suffix="", bool interpret_escape=false)

从 *socket* 读取所有数据并将其作为字符串返回。字节为网络字节序。

- int [socketReadAllString](#) (const std::string &variable, const std::string &socket_name="socket_0")

从 *socket* 读取所有数据并将其作为 *char* 向量返回。

- int [socketSendByte](#) (char value, const std::string &socket_name="socket_0")

发送一个字节到服务器通过 *socket* 发送字节 $\langle value \rangle$ ，不期望响应。可用于发送特殊 *ASCII* 字符；10 为换行符，2 为文本开始，3 为文本结束。

- int [socketSendInt](#) (int value, const std::string &socket_name="socket_0")

发送一个 *int* (*int32_t*) 到服务器通过 *socket* 发送 *int*，以网络字节序发送。不期望响应。

- int [socketSendLine](#) (const std::string &str, const std::string &socket_name="socket_0")

发送带有换行符的字符串到服务器。

- int [socketSendString](#) (const std::string &str, const std::string &socket_name="socket_0")

发送字符串到服务器通过 *socket* 以 *ASCII* 编码发送字符串 $\langle str \rangle$ ，不期望响应。

- int [socketSendAllString](#) (bool is_check, const std::vector< char > &str, const std::string &socket_name="socket_0")

发送给定 *char* 向量中的所有数据到服务器。

- bool [socketHasConnected](#) (const std::string &socket_name="socket_0")

检测 *socket* 连接是否成功

Protected 属性

- void * [d_](#)

11.23.1 详细描述

在文件 [socket.h](#) 第 22 行定义.

11.23.2 构造及析构函数说明

Socket()

```
arcs::common_interface::Socket::Socket ()
```

~Socket()

```
virtual arcs::common_interface::Socket::~~Socket () [virtual]
```

11.23.3 类成员变量说明

d_

```
void* arcs::common_interface::Socket::d_ [protected]
```

在文件 [socket.h](#) 第 640 行定义.

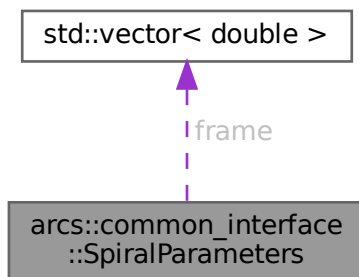
该类的文档由以下文件生成:

- include/aubo/[socket.h](#)

11.24 arcs::common_interface::SpiralParameters 结构体参考

```
#include <type_def.h>
```

arcs::common_interface::SpiralParameters 的协作图:



Public 属性

- `std::vector< double >` `frame`
参考点, 螺旋线的中心点和参考坐标系
- `int` `plane`
参考平面选择 0-XY 1-YZ 2-ZX
- `double` `angle`
转动的角度, 如果为正数, 机器人逆时针旋转
- `double` `spiral`
正数外扩
- `double` `helix`
正数上升

11.24.1 详细描述

在文件 `type_def.h` 第 765 行定义.

11.24.2 类成员变量说明

angle

```
double arcs::common_interface::SpiralParameters::angle
```

转动的角度, 如果为正数, 机器人逆时针旋转
在文件 `type_def.h` 第 769 行定义.

frame

```
std::vector<double> arcs::common_interface::SpiralParameters::frame
```

参考点, 螺旋线的中心点和参考坐标系
在文件 `type_def.h` 第 767 行定义.

helix

```
double arcs::common_interface::SpiralParameters::helix
```

正数上升
在文件 `type_def.h` 第 771 行定义.

plane

```
int arcs::common_interface::SpiralParameters::plane
```

参考平面选择 0-XY 1-YZ 2-ZX

在文件 `type_def.h` 第 768 行定义.

spiral

```
double arcs::common_interface::SpiralParameters::spiral
```

正数外扩

在文件 `type_def.h` 第 770 行定义.

该结构体的文档由以下文件生成:

- `include/aubo/type_def.h`

11.25 arcs::common_interface::SyncMove 类参考

```
#include <sync_move.h>
```

Public 成员函数

- `SyncMove ()`
- `virtual ~SyncMove ()`
- `int syncMoveOn (const std::string &syncident, const TaskSet &taskset)`
syncMoveOn 用于启动同步运动模式。
- `bool syncMoveSegment (int id)`
设置同步路径段的ID 在同步运动模式下, 所有同时执行的移动指令必须全部编程为圆角区 (*corner zones*) 或全部为停止点 (*stop points*)。这意味着具有相同ID 的移动指令要么全部带有圆角区, 要么全部带有停止点。如果在各自的程序任务中同步执行的移动指令中, 一个带有圆角区而另一个带有停止点, 则会发生错误。同步执行的移动指令可以有不同大小的圆角区 (例如, 一个使用 *z10*, 另一个使用 *z50*)。
- `int syncMoveOff (const std::string &syncident)`
syncMoveOff 用于结束同步运动模式。
- `int syncMoveUndo ()`
syncMoveUndo 用于关闭同步运动, 即使不是所有其他任务程序都执行了 *syncMoveUndo* 指令。
- `int waitSyncTasks (const std::string &syncident, const TaskSet &taskset)`
waitSyncTasks 用于在程序中的特定点同步多个任务程序。
- `bool isSyncMoveOn ()`
isSyncMoveOn 用于判断机械单元组是否处于同步运动模式。
- `int syncMoveSuspend ()`
暂停同步运动模式。
- `int syncMoveResume ()`
恢复同步运动模式。
- `int frameAdd (const std::string &name, const std::vector< double > &pose, const std::string &ref_name)`
添加一个名为 *name* 的坐标系, 其初始位姿为 *pose*, 位姿以 *ref_frame* 坐标系表达。此命令仅向世界模型添加一个坐标系, 并不会将其附加到 *ref_frame* 坐标系。如需将新添加的坐标系附加到 *ref_frame*, 请使用 *frameAttach()*。
- `int frameAttach (const std::string &child, const std::string &parent)`
将子坐标系附加到父世界模型对象。附加时会设置父子之间的相对变换, 使得子坐标系在世界中不会移动。
- `int frameDeleteAll ()`
删除所有已添加到世界模型的坐标系。
- `int frameDelete (const std::string &name)`
删除指定名称的坐标系。
- `int frameMove (const std::string &name, const std::vector< double > &pose, const std::string &ref_name)`
更改名为 *name* 的坐标系的位置, 将其移动到由 *pose* 指定的新位置, *pose* 以 *ref_name* 坐标系表达。

- `std::vector< double > frameGetPose` (const std::string &name, const std::string &rel_frame, const std::string &ref_frame)
 获取名为 *name* 的坐标系相对于 *rel_frame* 坐标系的位姿，并以 *ref_frame* 坐标系表达。如果未提供 *ref_frame*，则返回 *name* 坐标系相对于 *rel_frame* 坐标系的位姿，并以 *rel_frame* 坐标系表达。
- `std::vector< double > frameConvertPose` (const std::vector< double > &pose, const std::string &from_frame, const std::string &to_frame)
 将位姿从 *from_frame* 坐标系转换到 *to_frame* 坐标系。
- `bool frameExist` (const std::string &name)
 查询指定名称的坐标系是否存在。
- `std::string frameGetParent` (const std::string &name)
 获取名为 *name* 的坐标系在世界模型中的父坐标系名称。
- `std::vector< std::string > frameGetChildren` (const std::string &name)
 返回指定父对象的直接子对象坐标系名称列表。父子关系由世界模型的附加关系定义。如果用于 *Motion←Plus*，子对象也可以是轴组或轴。
- `int axisGroupAdd` (const std::string &name, const std::vector< double > &pose, const std::string &ref_frame)
 向世界模型添加一个新的轴组，名称为 *name*。轴组基座放置在 *ref_frame* 坐标系下的 *pose* 位置。
- `int axisGroupDelete` (const std::string &name)
 删除具有给定名称的轴组。
- `int axisGroupAddAxis` (const std::string &group_name, const std::string &name, const std::string &parent, const std::vector< double > &pose)
 向名为 *group_name* 的轴组添加一个名为 *name* 的外部轴。该轴在 *parent* 坐标系下的 *pose* 位置附加，*pose* 表示轴位置为 0 时的位姿。轴的类型、最大速度、最大加速度、位置限制和索引分别由 *type*、*v_limit*、*a_limit*、*q_limits* 和 *axis_index* 定义。*pose* 参数通常通过外部轴调试标定流程获得。如果该轴组正被其他函数控制，或附加关系形成闭环，则操作会失败。
- `int axisGroupUpdateAxis` (const std::string &name, const std::vector< double > &pose)
 更新指定名称的轴的相关属性。*pose* 参数通常通过外部轴调试标定流程获得。如果该轴所属的轴组正被其他命令控制，则操作会失败。如果该轴组中任何已附加的轴处于激活和使能状态，则操作会失败。
- `int axisGroupGetAxisIndex` (const std::string &name)
 返回指定轴名称在 *RTDE* 目标位置 and 实际位置数组中的索引。
- `std::string axisGroupGetAxisName` (int index)
 返回指定轴索引对应的轴名称。
- `std::vector< double > axisGroupGetTargetPositions` (const std::string &group_name)
 返回指定轴组的当前目标位置。如果未指定 *group_name*，则返回所有外部轴的目标位置。
- `std::vector< double > axisGroupGetActualPositions` (const std::string &group_name)
 返回指定轴组的当前实际位置。如果未指定 *group_name*，则返回所有外部轴的实际位置。
- `int axisGroupOffsetPositions` (const std::string &group_name, const std::vector< double > &offset)
 通过给定的 *offset*，将轴组 *group_name* 的目标位置和实际位置整体偏移。
- `int axisGroupMoveJoint` (const std::string &group_name, const std::vector< double > &q, double a, double v)
 以梯形速度曲线将名为 *group_name* 的轴组移动到新的位置 *q*。参数 *a* 指定本次运动的最大加速度占各轴加速度极限的百分比。参数 *v* 指定本次运动的最大速度占各轴速度极限的百分比。
- `int axisGroupSpeedJoint` (const std::string &group_name, const std::vector< double > &qd, double a, double t)
 以指定的加速度因子 *a*，将名为 *group_name* 的轴组加速到目标速度 *qd*。该函数会运行 *t* 秒。

Protected 属性

- `void * d_`

11.25.1 详细描述

在文件 `sync_move.h` 第 42 行定义。

11.25.2 构造及析构函数说明

SyncMove()

```
arcs::common_interface::SyncMove::SyncMove ()
```

~SyncMove()

```
virtual arcs::common_interface::SyncMove::~~SyncMove () [virtual]
```

11.25.3 类成员变量说明

d_

```
void* arcs::common_interface::SyncMove::d_ [protected]
```

在文件 [sync_move.h](#) 第 1008 行定义.

该类的文档由以下文件生成:

- [include/aubo/sync_move.h](#)

11.26 arcs::common_interface::SystemInfo 类参考

```
#include <system_info.h>
```

Public 成员函数

- [SystemInfo \(\)](#)
- virtual [~SystemInfo \(\)](#)
- int [getControlSoftwareVersionCode \(\)](#)
获取控制器软件版本号
- std::string [getControlSoftwareFullVersion \(\)](#)
获取完整控制器软件版本号
- int [getInterfaceVersionCode \(\)](#)
获取接口版本号
- std::string [getControlSoftwareBuildDate \(\)](#)
获取控制器软件构建时间
- std::string [getControlSoftwareVersionHash \(\)](#)
获取控制器软件 *git* 版本
- uint64_t [getControlSystemTime \(\)](#)
获取系统时间 (软件启动时间 *ns* 纳秒)

Protected 属性

- void * [d_](#)

11.26.1 详细描述

在文件 [system_info.h](#) 第 21 行定义.

11.26.2 构造及析构函数说明

SystemInfo()

```
arcs::common_interface::SystemInfo::SystemInfo ()
```

~SystemInfo()

```
virtual arcs::common_interface::SystemInfo::~~SystemInfo () [virtual]
```

11.26.3 类成员变量说明

d_

```
void* arcs::common_interface::SystemInfo::d_ [protected]
```

在文件 [system_info.h](#) 第 346 行定义.

该类的文档由以下文件生成:

- [include/aubo/system_info.h](#)

11.27 arcs::common_interface::Trace 类参考

```
#include <trace.h>
```

Public 成员函数

- [Trace](#) ()
- virtual [~Trace](#) ()
- int [alarm](#) ([TraceLevel](#) level, int code, const std::vector< std::string > &args={})
向 *aubo_control* 日志注入告警信息 *code* 定义参考 [error_stack](#)
- int [textmsg](#) (const std::string &msg)
打印文本信息到日志中
- int [notify](#) (const std::string &msg)
通知上位机
- int [popup](#) ([TraceLevel](#) level, const std::string &title, const std::string &msg, int mode)
向连接的 *RTDE* 客户端发送弹窗请求
- [RobotMsgVector](#) [peek](#) (size_t num, uint64_t last_time=0)
peek 最新的 *AlarmInfo*(上次一获取之后)

Protected 属性

- void * [d_](#)

11.27.1 详细描述

在文件 [trace.h](#) 第 22 行定义.

11.27.2 构造及析构函数说明

[Trace](#)()

```
arcs::common_interface::Trace::Trace ()
```

[~Trace](#)()

```
virtual arcs::common_interface::Trace::~~Trace () [virtual]
```

11.27.3 成员函数说明

[notify](#)()

```
int arcs::common_interface::Trace::notify (  
    const std::string & msg)
```

通知上位机

11.27.4 类成员变量说明

d_

```
void* arcs::common_interface::Trace::d_ [protected]
```

在文件 [trace.h](#) 第 229 行定义.

该类的文档由以下文件生成:

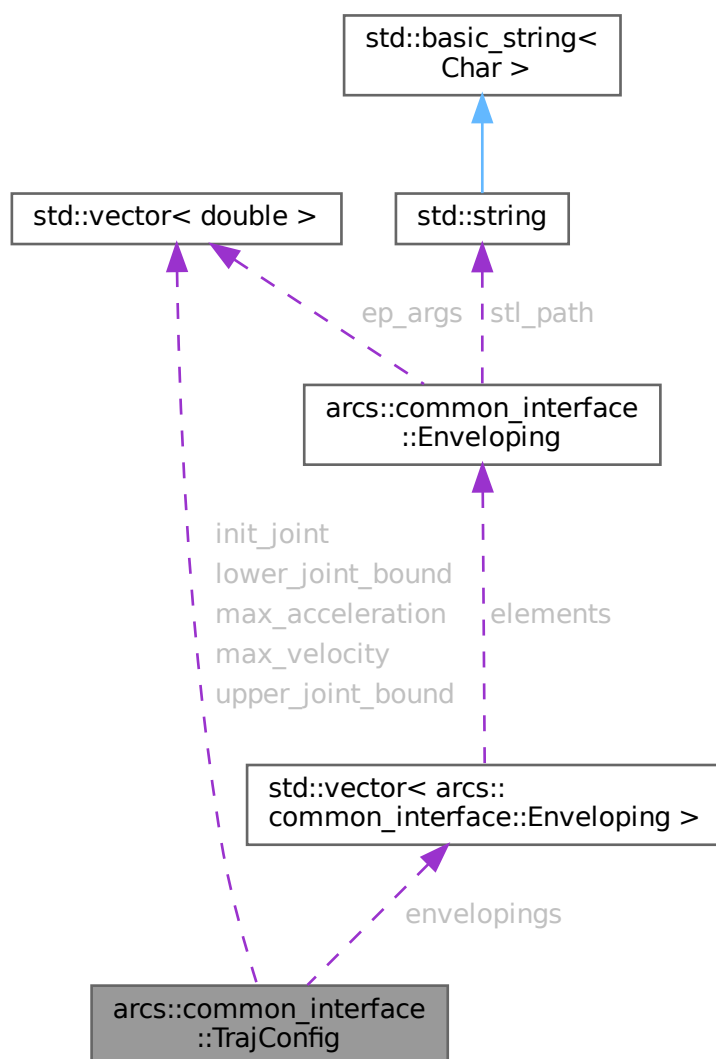
- [include/aubo/trace.h](#)

11.28 arcs::common_interface::TrajConfig 结构体参考

用于负载辨识的轨迹配置

```
#include <type_def.h>
```

arcs::common_interface::TrajConfig 的协作图:



Public 属性

- `std::vector< Enveloping > envelopings`
- `PayloadIdentifyMoveAxis move_axis`
- `std::vector< double > init_joint`
- `std::vector< double > upper_joint_bound`
- `std::vector< double > lower_joint_bound`
- `std::vector< double > max_velocity`
- `std::vector< double > max_acceleration`

11.28.1 详细描述

用于负载辨识的轨迹配置
在文件 [type_def.h](#) 第 795 行定义.

11.28.2 类成员变量说明

`envelopings`

`std::vector<Enveloping> arcs::common_interface::TrajConfig::envelopings`
在文件 [type_def.h](#) 第 797 行定义.

`init_joint`

`std::vector<double> arcs::common_interface::TrajConfig::init_joint`
在文件 [type_def.h](#) 第 799 行定义.

`lower_joint_bound`

`std::vector<double> arcs::common_interface::TrajConfig::lower_joint_bound`
在文件 [type_def.h](#) 第 801 行定义.

`max_acceleration`

`std::vector<double> arcs::common_interface::TrajConfig::max_acceleration`
在文件 [type_def.h](#) 第 803 行定义.

`max_velocity`

`std::vector<double> arcs::common_interface::TrajConfig::max_velocity`
在文件 [type_def.h](#) 第 802 行定义.

`move_axis`

`PayloadIdentifyMoveAxis arcs::common_interface::TrajConfig::move_axis`
在文件 [type_def.h](#) 第 798 行定义.

`upper_joint_bound`

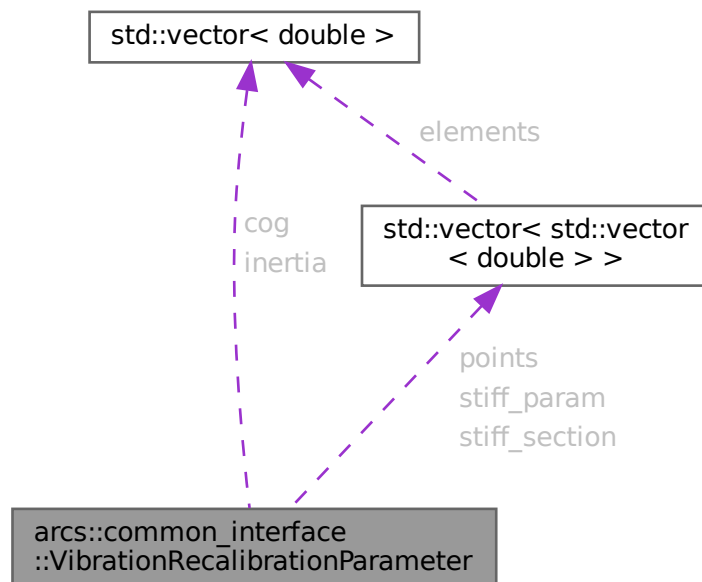
`std::vector<double> arcs::common_interface::TrajConfig::upper_joint_bound`
在文件 [type_def.h](#) 第 800 行定义.
该结构体的文档由以下文件生成:

- `include/aubo/type_def.h`

11.29 arcs::common_interface::VibrationRecalibrationParameter 结构体参考

```
#include <type_def.h>
```

arcs::common_interface::VibrationRecalibrationParameter 的协作图:



Public 属性

- double `mass`
- `std::vector< double >` `cog`
- `std::vector< double >` `inertia`
- `std::vector< std::vector< double > >` `points`
- `std::vector< std::vector< double > >` `stiff_section`
- `std::vector< std::vector< double > >` `stiff_param`

11.29.1 详细描述

在文件 `type_def.h` 第 214 行定义.

11.29.2 类成员变量说明

cog

```
std::vector<double> arcs::common_interface::VibrationRecalibrationParameter::cog
```

在文件 `type_def.h` 第 216 行定义.

inertia

```
std::vector<double> arcs::common_interface::VibrationRecalibrationParameter::inertia
```

在文件 `type_def.h` 第 217 行定义.

mass

```
double arcs::common_interface::VibrationRecalibrationParameter::mass
```

在文件 [type_def.h](#) 第 215 行定义.

points

```
std::vector<std::vector<double> > arcs::common_interface::VibrationRecalibrationParameter::points
```

在文件 [type_def.h](#) 第 218 行定义.

stiff_param

```
std::vector<std::vector<double> > arcs::common_interface::VibrationRecalibrationParameter::stiff_param
```

在文件 [type_def.h](#) 第 220 行定义.

stiff_section

```
std::vector<std::vector<double> > arcs::common_interface::VibrationRecalibrationParameter::stiff_section
```

在文件 [type_def.h](#) 第 219 行定义.

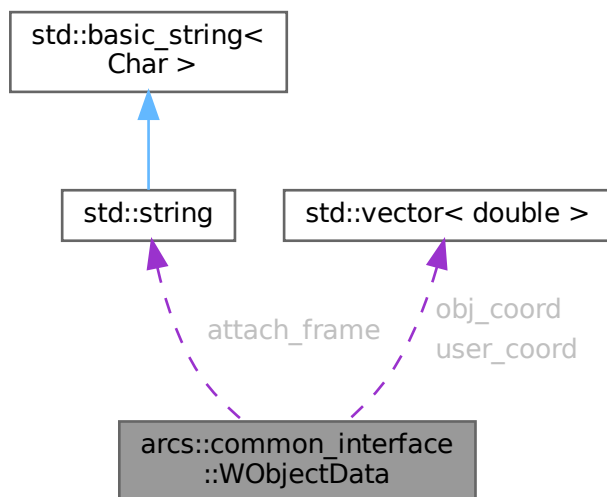
该结构体的文档由以下文件生成:

- [include/aubo/type_def.h](#)

11.30 arcs::common_interface::WObjectData 结构体参考

```
#include <type_def.h>
```

arcs::common_interface::WObjectData 的协作图:

**Public 属性**

- bool [remote_tool](#) { false }
是否为外部工具
- std::string [attach_frame](#) { "" }
- std::vector< double > [user_coord](#) { std::vector<double>(6, 0) }

用户坐标系如果 *robhold* 为 *false*, 那 *uframe* 的数值是基于 *world* 否则, *uframe* 的数值是基于 *flange*

- `std::vector< double > obj_coord { std::vector<double>(6, 0) }`

工件坐标系, 基于 *uframe*

11.30.1 详细描述

在文件 [type_def.h](#) 第 187 行定义.

11.30.2 类成员变量说明

`attach_frame`

```
std::string arcs::common_interface::WObjectData::attach_frame { "" }
```

工件坐标系耦合的

在文件 [type_def.h](#) 第 194 行定义.

`obj_coord`

```
std::vector<double> arcs::common_interface::WObjectData::obj_coord { std::vector<double>(6, 0) }
```

工件坐标系, 基于 *uframe*

在文件 [type_def.h](#) 第 205 行定义.

`remote_tool`

```
bool arcs::common_interface::WObjectData::remote_tool { false }
```

是否为外部工具

在文件 [type_def.h](#) 第 190 行定义.

`user_coord`

```
std::vector<double> arcs::common_interface::WObjectData::user_coord { std::vector<double>(6, 0) }
```

用户坐标系如果 *robhold* 为 *false*, 那 *uframe* 的数值是基于 *world* 否则, *uframe* 的数值是基于 *flange*

在文件 [type_def.h](#) 第 201 行定义.

该结构体的文档由以下文件生成:

- `include/aubo/type_def.h`

Chapter 12

文件说明

12.1 doc/sdk_overview.md 文件参考

12.2 include/aubo/aubo_api.h 文件参考

机器人及外部轴等控制API 接口，如获取机器人列表、获取系统信息等等

```
#include <aubo/system_info.h>
#include <aubo/runtime_machine.h>
#include <aubo/register_control.h>
#include <aubo/robot_interface.h>
#include <aubo/global_config.h>
#include <aubo/math.h>
#include <aubo/socket.h>
#include <aubo/serial.h>
#include <aubo/axis_interface.h>
#include <aubo/gripper_interface.h>
aubo_api.h 的引用 (Include) 关系图:
```



类

- class [arcs::common_interface::AuboApi](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::common_interface](#)

类型定义

- using [arcs::common_interface::AuboApiPtr](#) = std::shared_ptr<[AuboApi](#)>

12.2.1 详细描述

机器人及外部轴等控制API 接口，如获取机器人列表、获取系统信息等等
在文件 [aubo_api.h](#) 中定义.

12.3 aubo_api.h

[浏览该文件的文档.](#)

```

00001 /** @file aubo_api.h
00002  * @brief \-chinese 机器人及外部轴等控制 API 接口，如获取机器人列表、获取系统信息等等
00003  * @brief \-english API for controlling the robot and external axis
00004  */
00005 #ifndef AUBO_SDK_AUBO_API_INTERFACE_H
00006 #define AUBO_SDK_AUBO_API_INTERFACE_H
00007
00008 #include <aubo/system_info.h>
00009 #include <aubo/runtime_machine.h>
00010 #include <aubo/register_control.h>
00011 #include <aubo/robot_interface.h>
00012 #include <aubo/global_config.h>
00013 #include <aubo/math.h>
00014 #include <aubo/socket.h>
00015 #include <aubo/serial.h>
00016 #include <aubo/axis_interface.h>
00017 #include <aubo/gripper_interface.h>
00018
00019 namespace arcs {
00020 namespace common_interface {
00021
00022 /**
00023  * @defgroup AuboApi AuboApi (主入口)
00024  * @ingroup AuboApi
00025  * AuboApi
00026  */
00027 class ARCS_ABI_EXPORT AuboApi
00028 {
00029 public:
00030     AuboApi();
00031     virtual ~AuboApi();
00032
00033     /**
00034      * @ingroup AuboApi
00035      * @ref Math
00036      * \chinese
00037      * 获取纯数学相关接口
00038      *
00039      * @return MathPtr 对象的指针
00040      *
00041      * @par Python 函数原型
00042      * getMath(self: pyaubo_sdk.AuboApi) -> pyaubo_sdk.Math
00043      *
00044      * @par C++ 示例
00045      * @code
00046      * auto rpc_cli = std::make_shared<RpcClient>();
00047      * MathPtr ptr = rpc_cli->getMath();
00048      * @endcode
00049      * \endchinese
00050      *
00051      * \english
00052      * Get pure mathematic related API
00053      *
00054      * @return Shared pointer to a Math object
00055      *
00056      * @par Python function prototype
00057      * getMath(self: pyaubo_sdk.AuboApi) -> pyaubo_sdk.Math
00058      *
00059      * @par C++ example
00060      * @code
00061      * auto rpc_cli = std::make_shared<RpcClient>();
00062      * MathPtr ptr = rpc_cli->getMath();
00063      * @endcode
00064      * \endenglish
00065      */
00066     MathPtr getMath();
00067
00068     /**
00069      * @ingroup AuboApi
00070      * @ref SystemInfo
00071      * \chinese
00072      * 获取系统信息
00073      *
00074      * @return SystemInfoPtr 对象的指针
00075      *
00076      * @par Python 函数原型
00077      * getSystemInfo(self: pyaubo_sdk.AuboApi) -> pyaubo_sdk.SystemInfo
00078      *
00079      * @par C++ 示例
00080      * @code
00081      * auto rpc_cli = std::make_shared<RpcClient>();
00082      * SystemInfoPtr ptr = rpc_cli->getSystemInfo();
00083      * @endcode

```

```

00084     * \endchinese
00085     *
00086     * \english
00087     * Get system info
00088     *
00089     * @return Shared pointer to SystemInfo object
00090     *
00091     * @par Python function prototype
00092     * getSystemInfo(self: pyaubo_sdk.AuboApi) -> pyaubo_sdk.SystemInfo
00093     *
00094     * @par C++ example
00095     * @code
00096     * auto rpc_cli = std::make_shared<RpcClient>();
00097     * SystemInfoPtr ptr = rpc_cli->getSystemInfo();
00098     * @endcode
00099     * \endenglish
00100     */
00101     SystemInfoPtr getSystemInfo();
00102
00103     /**
00104     * @ingroup AuboApi
00105     * @ref RuntimeMachine
00106     * \chinese
00107     * 获取运行时接口
00108     *
00109     * @return RuntimeMachinePtr 对象的指针
00110     *
00111     * @par Python 函数原型
00112     * getRuntimeMachine(self: pyaubo_sdk.AuboApi) -> pyaubo_sdk.RuntimeMachine
00113     *
00114     * @par C++ 示例
00115     * @code
00116     * auto rpc_cli = std::make_shared<RpcClient>();
00117     * RuntimeMachinePtr ptr = rpc_cli->getRuntimeMachine();
00118     * @endcode
00119     * \endchinese
00120     *
00121     * \english
00122     * Get runtime api
00123     *
00124     * @return Shared pointer to RuntimeMachine object
00125     * Python function prototype
00126     * getRuntimeMachine(self: pyaubo_sdk.AuboApi) -> pyaubo_sdk.RuntimeMachine
00127     *
00128     * @par C++ example
00129     * @code
00130     * auto rpc_cli = std::make_shared<RpcClient>();
00131     * RuntimeMachinePtr ptr = rpc_cli->getRuntimeMachine();
00132     * @endcode
00133     * \endenglish
00134     */
00135     RuntimeMachinePtr getRuntimeMachine();
00136
00137     /**
00138     * @ingroup AuboApi
00139     * @ref RegisterControl
00140     * \chinese
00141     * 对外寄存器接口
00142     *
00143     * @return RegisterControlPtr 对象的指针
00144     *
00145     * @par Python 函数原型
00146     * getRegisterControl(self: pyaubo_sdk.AuboApi) ->
00147     * pyaubo_sdk.RegisterControl
00148     *
00149     * @par C++ 示例
00150     * @code
00151     * auto rpc_cli = std::make_shared<RpcClient>();
00152     * RegisterControlPtr ptr = rpc_cli->getRegisterControl();
00153     * @endcode
00154     * \endchinese
00155     *
00156     * \english
00157     * External registers api
00158     *
00159     * @return Shared pointer to RegisterControl object
00160     *
00161     * @par Python function prototype
00162     * getRegisterControl(self: pyaubo_sdk.AuboApi) ->
00163     * pyaubo_sdk.RegisterControl
00164     *
00165     * @par C++ example
00166     * @code
00167     * auto rpc_cli = std::make_shared<RpcClient>();
00168     * RegisterControlPtr ptr = rpc_cli->getRegisterControl();
00169     * @endcode
00170     * \endenglish

```

```

00171     */
00172 RegisterControlPtr getRegisterControl();
00173
00174 /**
00175  * @ingroup AuboApi
00176  * \chinese
00177  * 获取机器人列表
00178  *
00179  * @return 机器人列表
00180  *
00181  * @par Python 函数原型
00182  * getRobotNames(self: pyaubo_sdk.AuboApi) -> List[str]
00183  *
00184  * @par C++ 示例
00185  * @code
00186  * auto rpc_cli = std::make_shared<RpcClient>();
00187  * auto robot_name = rpc_cli->getRobotNames().front();
00188  * @endcode
00189  *
00190  * @par JSON-RPC 请求示例
00191  * {"jsonrpc": "2.0", "method": "getRobotNames", "params": [], "id": 1}
00192  *
00193  * @par JSON-RPC 响应示例
00194  * {"id": 1, "jsonrpc": "2.0", "result": ["rob1"]}
00195  * \endchinese
00196  *
00197  * \english
00198  * Get robot list
00199  *
00200  * @return robot list
00201  *
00202  * @par Python function prototype
00203  * getRobotNames(self: pyaubo_sdk.AuboApi) -> List[str]
00204  *
00205  * @par C++ example
00206  * @code
00207  * auto rpc_cli = std::make_shared<RpcClient>();
00208  * auto robot_name = rpc_cli->getRobotNames().front();
00209  * @endcode
00210  *
00211  * @par JSON-RPC request example
00212  * {"jsonrpc": "2.0", "method": "getRobotNames", "params": [], "id": 1}
00213  *
00214  * @par JSON-RPC response example
00215  * {"id": 1, "jsonrpc": "2.0", "result": ["rob1"]}
00216  * \endenglish
00217  */
00218 std::vector<std::string> getRobotNames();
00219
00220 /**
00221  * @ingroup AuboApi
00222  * @ref RobotInterface
00223  * \chinese
00224  * 根据名字获取 RobotInterfacePtr 接口
00225  *
00226  * @param name 机器人名字
00227  * @return RobotInterfacePtr 对象的指针
00228  *
00229  * @par Python 函数原型
00230  * getRobotInterface(self: pyaubo_sdk.AuboApi, arg0: str) ->
00231  * pyaubo_sdk.RobotInterface
00232  *
00233  * @par C++ 示例
00234  * @code
00235  * auto rpc_cli = std::make_shared<RpcClient>();
00236  * auto robot_name = rpc_cli->getRobotNames().front();
00237  * RobotInterfacePtr ptr = rpc_cli->getRobotInterface(robot_name);
00238  * @endcode
00239  * \endchinese
00240  *
00241  * \english
00242  * Get RobotInterfacePtr based on name
00243  *
00244  * @param name Robot name
00245  * @return Shared pointer to a RobotInterface object
00246  *
00247  * @par Python function prototype
00248  * getRobotInterface(self: pyaubo_sdk.AuboApi, arg0: str) ->
00249  * pyaubo_sdk.RobotInterface
00250  *
00251  * @par C++ example
00252  * @code
00253  * auto rpc_cli = std::make_shared<RpcClient>();
00254  * auto robot_name = rpc_cli->getRobotNames().front();
00255  * RobotInterfacePtr ptr = rpc_cli->getRobotInterface(robot_name);
00256  * @endcode
00257  * \endenglish

```

```

00258     */
00259 RobotInterfacePtr getRobotInterface(const std::string &name);
00260
00261 /**
00262  * @ingroup AuboApi
00263  * \-chinese 获取外部轴列表 \-english Get external axis list
00264  *
00265  * @return
00266  */
00267 std::vector<std::string> getAxisNames();
00268
00269 /**
00270  * @ingroup AuboApi
00271  * @ref AxisInterface
00272  * \-chinese
00273  * 获取外部轴接口
00274  *
00275  * @param name
00276  * @return
00277  * \endchinese
00278  *
00279  * \english
00280  * Get external axis interface
00281  *
00282  * @param name
00283  * @return
00284  * \endenglish
00285  */
00286 AxisInterfacePtr getAxisInterface(const std::string &name);
00287
00288 /**
00289  * @ingroup AuboApi
00290  * @ref Socket
00291  * \-chinese
00292  * 获取 socket
00293  * @return SocketPtr 对象的指针
00294  *
00295  * @par Python 函数原型
00296  * getSocket(self: pyaubo_sdk.AuboApi) -> arcs::common_interface::Socket
00297  * @endcode
00298  *
00299  * @par C++ 示例
00300  * @code
00301  * auto rpc_cli = std::make_shared<RpcClient>();
00302  * SocketPtr ptr = rpc_cli->getSocket();
00303  * @endcode
00304  * \endchinese
00305  *
00306  * \english
00307  * Get socket
00308  * @return Shared pointer to a socket object
00309  *
00310  * @par Python function prototype
00311  * getSocket(self: pyaubo_sdk.AuboApi) -> arcs::common_interface::Socket
00312  * @endcode
00313  *
00314  * @par C++ example
00315  * @code
00316  * auto rpc_cli = std::make_shared<RpcClient>();
00317  * SocketPtr ptr = rpc_cli->getSocket();
00318  * @endcode
00319  * \endenglish
00320  */
00321 SocketPtr getSocket();
00322
00323 /**
00324  * @ingroup AuboApi
00325  * @ref Serial
00326  * \-chinese
00327  * 获取 Serial 串口
00328  * @return SerialPtr 对象的指针
00329  *
00330  * @par Python 函数原型
00331  * getSerial(self: pyaubo_sdk.AuboApi) -> arcs::common_interface::Serial
00332  *
00333  * @par C++ 示例
00334  * @code
00335  * auto rpc_cli = std::make_shared<RpcClient>();
00336  * SerialPtr ptr = rpc_cli->getSerial();
00337  * @endcode
00338  * \endchinese
00339  *
00340  * \english
00341  * @return Shared pointer to Serial object
00342  *
00343  * @par Python function prototype
00344  * getSerial(self: pyaubo_sdk.AuboApi) -> arcs::common_interface::Serial

```

```

00345      *
00346      * @par C++ example
00347      * @code
00348      * auto rpc_cli = std::make_shared<RpcClient>();
00349      * SerialPtr ptr = rpc_cli->getSerial();
00350      * @endcode
00351      * \endenglish
00352      */
00353      SerialPtr getSerial();
00354
00355      /**
00356      * @ingroup AuboApi
00357      * @ref SyncMove
00358      * \-chinese 获取同步运动接口 \-english Get synchronous move interface
00359      *
00360      * \-chinese @return SyncMovePtr 对象的指针
00361      * \-english @return Shared pointer to SyncMove object
00362      */
00363      SyncMovePtr getSyncMove(const std::string &name);
00364
00365      /**
00366      * @ingroup AuboApi
00367      * @ref Trace
00368      * \-chinese 获取告警信息接口
00369      * \-english Get alert interface
00370      *
00371      * \-chinese @return TracePtr 对象的指针
00372      * \-english @return Shared pointer of trace object
00373      */
00374      TracePtr getTrace(const std::string &name);
00375
00376      /**
00377      * @ingroup AuboApi
00378      * @ref GripperInterface
00379      * \-chinese 获取通用夹爪接口
00380      * \-english Get gripper interface
00381      *
00382      * \-chinese @return GripperInterfacePtr 对象的指针
00383      * \-english @return Shared pointer of gripper object
00384      */
00385      GripperInterfacePtr getGripperInterface();
00386
00387      protected:
00388          void *d_{ nullptr };
00389      };
00390      using AuboApiPtr = std::shared_ptr<AuboApi>;
00391
00392      } // namespace common_interface
00393      } // namespace arcs
00394
00395      #endif // AUBO_SDK_AUBO_API_H

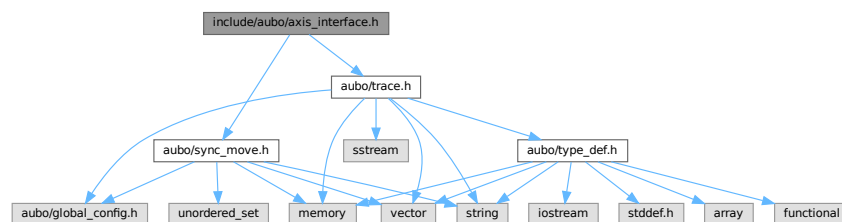
```

12.4 include/aubo/axis_interface.h 文件参考

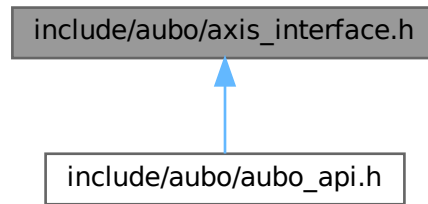
```

#include <aubo/sync_move.h>
#include <aubo/trace.h>
axis_interface.h 的引用 (Include) 关系图:

```



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::common_interface::AxisInterface`

命名空间

- namespace `arcs`
- namespace `arcs::common_interface`

类型定义

- using `arcs::common_interface::AxisInterfacePtr` = `std::shared_ptr<AxisInterface>`

12.5 axis_interface.h

[浏览该文件的文档.](#)

```

00001 /** @file axes.h
00002  * @brief 外部轴接口
00003  */
00004 #ifndef AUBO_SDK_AXIS_INTERFACE_H
00005 #define AUBO_SDK_AXIS_INTERFACE_H
00006
00007 #include <aubo/sync_move.h>
00008 #include <aubo/trace.h>
00009
00010 namespace arcs {
00011 namespace common_interface {
00012
00013 /**
00014  * @defgroup AxisInterface AxisInterface (外部轴)
00015  * \-chinese 外部轴 API 接口
00016  * \-english External axis API interface
00017  */
00018 class ARCS_ABI_EXPORT AxisInterface
00019 {
00020 public:
00021     AxisInterface();
00022     virtual ~AxisInterface();
00023
00024     /**
00025      * @ingroup AxisInterface
00026      * \-chinese 通电
00027      * \-english Power on
00028      * @return
00029      */
00030     int poweronExtAxis();
00031
00032     /**
00033      * @ingroup AxisInterface
00034      * \-chinese 断电
00035      * \-english Power off
00036      * @return
00037      */
00038     int poweroffExtAxis();
  
```

```

00039
00040 /**
00041  * @ingroup AxisInterface
00042  * \-chinese 使能
00043  * \-english Enable
00044  * @return
00045  */
00046 int enableExtAxis();
00047
00048 /**
00049  * @ingroup AxisInterface
00050  * \-chinese 设置外部轴的安装位姿（相对于世界坐标系）
00051  * \-chinese @param pose
00052  * \-english Set mounting pose of external axis (wrt world frame)
00053  * \-english @param pose
00054
00055  * @return
00056  */
00057 int setExtAxisMountingPose(const std::vector<double> &pose);
00058
00059 /**
00060  * @ingroup AxisInterface
00061  * \-chinese 运动到指定点，旋转或者平移
00062  *
00063  * @param pos
00064  * @param v
00065  * @param a
00066  * @param duration
00067  * @return
00068  * \endchinese
00069  *
00070  * \english move to pos, rotation or linear
00071  *
00072  * @param pos
00073  * @param v
00074  * @param a
00075  * @param duration
00076  * @return
00077  * \endenglish
00078  */
00079 int moveExtJoint(double pos, double v, double a, double duration);
00080
00081 /**
00082  * @ingroup AxisInterface
00083  * \-chinese
00084  * 制定目标运动速度
00085  *
00086  * @param v
00087  * @param a
00088  * @param duration
00089  * @return
00090  * \endchinese
00091  *
00092  * \english
00093  * Set target speed, acceleration and duration
00094  * @param v
00095  * @param a
00096  * @param duration
00097  * @return
00098  * \endenglish
00099  */
00100 int speedExtJoint(double v, double a, double duration);
00101
00102 /**
00103  * @ingroup AxisInterface
00104  * \-chinese
00105  * 停止外部轴运动
00106  *
00107  * @param a
00108  * @return
00109  * \endchinese
00110  *
00111  * \english
00112  * stop ext joint
00113  * @param a
00114  * @return
00115  * \endenglish
00116  */
00117 int stopExtJoint(double a);
00118
00119 /**
00120  * @ingroup AxisInterface
00121  * \-chinese 获取外部轴的类型 0 代表是旋转 1 代表平移
00122  * \-english Get external axis type: 0 for rotation, 1 for linear
00123  * @return
00124  */
00125 int getExtAxisType();

```



```

00126
00127 /**
00128  * @ingroup AxisInterface
00129  * \~chinese 设置外部轴的类型
00130  * @param type 0 代表是旋转 1 代表平移
00131  *
00132  * \~english Set external axis type
00133  * @param type: 0 for rotation, 1 for linear
00134  * @return
00135  */
00136 int setExtAxisType(int type);
00137
00138 /**
00139  * @ingroup AxisInterface
00140  * \~chinese
00141  * 获取当前外部轴的状态
00142  *
00143  * @return 当前外部轴的状态
00144  * \endchinese
00145  *
00146  * \english
00147  * Get external axis status
00148  *
00149  * @return Current external axis status
00150  * \endenglish
00151  */
00152 AxisModeType getAxisModeType();
00153
00154 /**
00155  * @ingroup AxisInterface
00156  * \~chinese
00157  * 获取外部轴安装位姿
00158  *
00159  * @return 外部轴安装位姿
00160  * \endchinese
00161  *
00162  * \english
00163  * Get external axis mounting pose
00164  *
00165  * @return External axis pose
00166  * \endenglish
00167  */
00168 std::vector<double> getExtAxisMountingPose();
00169
00170 /**
00171  * @ingroup AxisInterface
00172  * \~chinese
00173  * 获取相对于安装坐标系的位姿，外部轴可能为变位机或者导轨
00174  *
00175  * @return 相对于安装坐标系的位姿
00176  * \endchinese
00177  *
00178  * \english
00179  * Get pose wrt mounting coordinate system, axis can be positioner or linear
00180  * rail
00181  *
00182  * @return Pose wrt mounting coordinate system
00183  * \endenglish
00184  */
00185 std::vector<double> getExtAxisPose();
00186
00187 /**
00188  * @ingroup AxisInterface
00189  * \~chinese
00190  * 获取外部轴位置
00191  *
00192  * @return 外部轴位置
00193  * \endchinese
00194  *
00195  * \english
00196  * Get external axis position
00197  *
00198  * @return External axis position
00199  * \endenglish
00200  */
00201 double getExtAxisPosition();
00202
00203 /**
00204  * @ingroup AxisInterface
00205  * \~chinese
00206  * 获取外部轴运行速度
00207  *
00208  * @return 外部轴运行速度
00209  * \endchinese
00210  *
00211  * \english
00212  * Get external axis speed

```

```

00213      *
00214      * @return External axis speed
00215      * \endenglish
00216      */
00217 double getExtAxisVelocity();
00218
00219 /**
00220  * @ingroup AxisInterface
00221  * \chinese
00222  * 设置外部轴运行速度
00223  *
00224  * @return
00225  * \endchinese
00226  *
00227  * \english
00228  * Set external axis speed
00229  *
00230  * @return
00231  * \endenglish
00232  */
00233 int setExtAxisVelocity(double velocity);
00234
00235 /**
00236  * @ingroup AxisInterface
00237  * \chinese
00238  * 获取外部轴运行加速度
00239  *
00240  * @return 外部轴运行加速度
00241  * \endchinese
00242  *
00243  * \english
00244  * Get external axis acceleration
00245  *
00246  * @return External axis acceleration
00247  * \endenglish
00248  */
00249 double getExtAxisAcceleration();
00250
00251 /**
00252  * @ingroup AxisInterface
00253  * \chinese
00254  * 设置外部轴运行加速度
00255  *
00256  * @param acc
00257  * @return
00258  * \endchinese
00259  *
00260  * \english
00261  * Set external axis acceleration
00262  *
00263  * @param acc
00264  * @return
00265  * \endenglish
00266  */
00267 int setExtAxisAcceleration(double acc);
00268
00269 /**
00270  * @ingroup AxisInterface
00271  * \chinese
00272  * 获取外部轴电流
00273  *
00274  * @return 外部轴电流
00275  * \endchinese
00276  *
00277  * \english
00278  * Get external axis current
00279  *
00280  * @return External axis current
00281  * \endenglish
00282  */
00283 double getExtAxisCurrent();
00284
00285 /**
00286  * @ingroup AxisInterface
00287  * \chinese
00288  * 获取外部轴温度
00289  *
00290  * @return 外部轴温度
00291  * \endchinese
00292  *
00293  * \english
00294  * Get external axis temperature
00295  *
00296  * @return External axis temperature
00297  * \endenglish
00298  */
00299 double getExtAxisTemperature();

```

```

00300
00301 /**
00302  * @ingroup AxisInterface
00303  * \chinese
00304  * 获取外部轴电压
00305  *
00306  * @return 外部轴电压
00307  * \endchinese
00308  *
00309  * \english
00310  * Get external axis voltage
00311  *
00312  * @return External axis voltage
00313  * \endenglish
00314  */
00315 double getExtAxisBusVoltage();
00316
00317 /**
00318  * @ingroup AxisInterface
00319  * \chinese
00320  * 获取外部轴电流
00321  *
00322  * @return 外部轴电流
00323  * \endchinese
00324  *
00325  * \english
00326  * Get external axis current
00327  *
00328  * @return external axis current
00329  * \endenglish
00330  */
00331 double getExtAxisBusCurrent();
00332
00333 /**
00334  * @ingroup AxisInterface
00335  * \chinese
00336  * 获取外部轴最大位置（物理极限）
00337  *
00338  * @return 外部轴最大位置
00339  * \endchinese
00340  *
00341  * \english
00342  * Get external axis max position
00343  *
00344  * @return External axis max position
00345  * \endenglish
00346  */
00347 double getExtAxisMaxPosition();
00348
00349 /**
00350  * @ingroup AxisInterface
00351  * \chinese
00352  * 设置外部轴最大位置（软件/算法限制）
00353  *
00354  * @param q
00355  * @return
00356  * \endchinese
00357  *
00358  * \english
00359  * Set external axis max position
00360  *
00361  * @param q
00362  * @return
00363  * \endenglish
00364  */
00365 int setExtAxisMaxPositionLimit(double q);
00366
00367 /**
00368  * @ingroup AxisInterface
00369  * \chinese
00370  * 获取外部轴最大位置（软件/算法限制）
00371  *
00372  * @return 外部轴最大位置
00373  * \endchinese
00374  *
00375  * \english
00376  * Get external axis max position
00377  *
00378  * @return External axis max position
00379  * \endenglish
00380  */
00381 double getExtAxisMaxPositionLimit();
00382
00383 /**
00384  * @ingroup AxisInterface
00385  * \chinese
00386  * 获取外部轴最小位置（物理极限）

```

```

00387      *
00388      * @return 外部轴最小位置
00389      * \endchinese
00390      *
00391      * \english
00392      * Get external axis min position
00393      *
00394      * @return External axis min position
00395      * \endenglish
00396      */
00397 double getExtMinPosition();
00398
00399 /**
00400  * @ingroup AxisInterface
00401  * \chinese
00402  * 设置外部轴最小位置（软件/算法限制）
00403  *
00404  * @param q
00405  * @return
00406  * \endchinese
00407  *
00408  * \english
00409  * Set external axis min position
00410  *
00411  * @param q
00412  * @return
00413  * \endenglish
00414  */
00415 int setExtMinPositionLimit(double q);
00416
00417 /**
00418  * @ingroup AxisInterface
00419  * \chinese
00420  * 获取外部轴最小位置（软件/算法限制）
00421  *
00422  * @return 外部轴最小位置
00423  * \endchinese
00424  *
00425  * \english
00426  * Get external axis min position
00427  *
00428  * @return External axis min position
00429  * \endenglish
00430  */
00431 double getExtMinPositionLimit();
00432
00433 /**
00434  * @ingroup AxisInterface
00435  * \chinese
00436  * 获取外部轴最大速度（物理极限）
00437  *
00438  * @return 外部轴最大速度
00439  * \endchinese
00440  *
00441  * \english
00442  * Get external axis max speed
00443  *
00444  * @return External axis max speed
00445  * \endenglish
00446  */
00447 double getExtAxisMaxVelocity();
00448
00449 /**
00450  * @ingroup AxisInterface
00451  * \chinese
00452  * 设置外部轴最大速度（软件/算法限制）
00453  *
00454  * @param v
00455  * @return
00456  * \endchinese
00457  *
00458  * \english
00459  * Set external axis max speed
00460  *
00461  * @param v
00462  * @return
00463  * \endenglish
00464  */
00465 int setExtAxisMaxVelocityLimit(double v);
00466
00467 /**
00468  * @ingroup AxisInterface
00469  * \chinese
00470  * 获取外部轴最大速度（软件/算法限制）
00471  *
00472  * @return 外部轴最大速度
00473  * \endchinese

```

```

00474      *
00475      * \english
00476      * Get external axis max speed
00477      *
00478      * @return External axis max speed
00479      * \endenglish
00480      */
00481 double getExtAxisMaxVelocityLimit();
00482
00483 /**
00484  * @ingroup AxisInterface
00485  * \chinese
00486  * 获取外部轴最大加速度（物理极限）
00487  *
00488  * @return 外部轴最大加速度
00489  * \endchinese
00490  *
00491  * \english
00492  * Get external axis max acceleration
00493  *
00494  * @return External axis max acceleration
00495  * \endenglish
00496  */
00497 double getExtAxisMaxAcceleration();
00498
00499 /**
00500  * @ingroup AxisInterface
00501  * \chinese
00502  * 设置外部轴最大加速度（软件/算法限制）
00503  *
00504  * @param acc
00505  * @return
00506  * \endchinese
00507  *
00508  * \english
00509  * Set external axis max acceleration
00510  *
00511  * @param acc
00512  * @return
00513  * \endenglish
00514  */
00515 int setExtAxisMaxAccelerationLimit(double acc);
00516
00517 /**
00518  * @ingroup AxisInterface
00519  * \chinese
00520  * 获取外部轴最大加速度（软件/算法限制）
00521  *
00522  * @return 外部轴最大加速度
00523  * \endchinese
00524  *
00525  * \english
00526  * Get external axis max acceleration
00527  *
00528  * @return External axis max acceleration
00529  * \endenglish
00530  */
00531 double getExtAxisMaxAccelerationLimit();
00532
00533 /**
00534  * @ingroup AxisInterface
00535  * \chinese
00536  * 跟踪另一个外部轴的运动（禁止运动过程中使用）
00537  *
00538  * @param target_name 目标的外部轴名字
00539  * @param phase 相位差
00540  * @param err 跟踪运行的最大误差
00541  * @return
00542  * \endchinese
00543  *
00544  * \english
00545  * Follow motion of another external axis (not to be used during motion)
00546  *
00547  * @param target_name name of target axis
00548  * @param phase phase difference
00549  * @param err max error when following motion
00550  * @return
00551  * \endenglish
00552  */
00553 int followAnotherAxis(const std::string &target_name, double phase,
00554                      double err);
00555
00556 /**
00557  * @ingroup AxisInterface
00558  * \chinese @brief stopFollowAnotherAxis(禁止运动过程中使用)
00559  * \english @brief stopFollowAnotherAxis(not to be used during motion)
00560  * @return

```

```

00561     */
00562     int stopFollowAnotherAxis();
00563
00564     /**
00565      * @ingroup AxisInterface
00566      * \-chinese @brief 获取外部轴错误码(获取外部轴错误码)
00567      * \-english @brief 获取外部轴错误码(Get raw external axis error code)
00568      * @return
00569      */
00570     int getErrorCode();
00571
00572     /**
00573      * @ingroup AxisInterface
00574      * \-chinese
00575      * 重置外部轴错误
00576      *
00577      * @return
00578      * \endchinese
00579      *
00580      * \english
00581      * Reset axis error
00582      *
00583      * @return
00584      * \endenglish
00585      */
00586     int clearAxisError();
00587
00588     /**
00589      * @ingroup AxisInterface
00590      * \-chinese
00591      * 将当前外部轴位置设置为零位
00592      *
00593      * @return
00594      * \endchinese
00595      *
00596      * \english
00597      * Set current external axis position as zero.
00598      *
00599      * @return
00600      * \endenglish
00601      */
00602     int setExtAxisZero();
00603
00604     /**
00605      * @ingroup AxisInterface
00606      * \-chinese
00607      * 设置外部轴减速比
00608      *
00609      * @return
00610      * \endchinese
00611      *
00612      * \english
00613      * Set axis reduction ratio
00614      *
00615      * @return
00616      * \endenglish
00617      */
00618     int setReductionRatio(double ratio);
00619
00620     /**
00621      * @ingroup AxisInterface
00622      * \-chinese
00623      * 获取外部轴减速比
00624      *
00625      * @return
00626      * \endchinese
00627      *
00628      * \english
00629      * Get axis reduction ratio
00630      *
00631      * @return
00632      * \endenglish
00633      */
00634     double getReductionRatio();
00635
00636 protected:
00637     void *d_;
00638 };
00639 using AxisInterfacePtr = std::shared_ptr<AxisInterface>;
00640
00641 } // namespace common_interface
00642 } // namespace arcs
00643
00644 #endif // AUBO_SDK_AXIS_INTERFACE_H

```

12.6 include/aubo/error_stack/error_stack.h 文件参考

汇总错误码

```
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include <string>
#include <sstream>
#include <iomanip>
#include <aubo/global_config.h>
#include <aubo/error_stack/hal_error.h>
#include <aubo/error_stack/rtm_error.h>
#include <aubo/error_stack/system_error.h>
```



命名空间

- namespace `arcs`
- namespace `arcs::error` `stack`

宏定义

- #define _PH1_ ”{}”
- #define _PH2_ ”{}”
- #define _PH3_ ”{}”
- #define _PH4_ ”{}”
- #define AUBO_SDK_HAL_ERROR_H
- #define JOINT_ERRORS
- #define JOINT_EX_ERRORS
- #define EXT_AXIS_ERRORS
- #define TOOL_ERRORS
- #define EX_TOOL_ERRORS
- #define PEDSTRAL_ERRORS
- #define EX_PEDSTRAL_ERRORS
- #define SAFETY_INTERFACE_BOARD_ERRORS
- #define HARDWARE_INTERFACE_ERRORS
- #define HAL_ERRORS
- #define AUBO_SDK_RTM_ERROR_H
- #define RTM_ERRORS
- #define AUBO_SDK_SYSTEM_ERROR_H
- #define SYSTEM_ERRORS
- #define ARCS_ERROR_CODES
- #define _D(n, v, s, r)
- #define _D(n, v, s, r)
- #define _D(n, v, s, r)
- #define _D(n, v, s, r)
- #define _D(n, v, s, r)
- #define _D(n, v, s, r)

枚举

- enum `arcs::error_stack::ErrorCodes` { `arcs::error_stack::ARCS_ERROR_CODES` }

函数

- constexpr int ARCS_ABI_EXPORT `arcs::error_stack::codeCompose` (int aa, int bb, int cccc)
- constexpr int ARCS_ABI_EXPORT `arcs::error_stack::mod` (int x)
- int `arcs::error_stack::str2ErrorCode` (const char *err_code_name)
- const char * `arcs::error_stack::errorCode2Str` (int err_code)
- std::ostream & `arcs::error_stack::dump` (std::ostream &os)

12.6.1 详细描述

汇总错误码

在文件 `error_stack.h` 中定义.

12.6.2 宏定义说明

__D [1/6]

```
#define _D(
    n,
    v,
    s,
    r)
```

值:

```
os << std::setw(20) << #n << "\t" << v << "\t" << s << "\t" << r \
<< std::endl;
```

在文件 `error_stack.h` 第 58 行定义.

__D [2/6]

```
#define _D(
    n,
    v,
    s,
    r)
```

值:

```
if (err_code == v) \
    index = n##_INDEX;
```

在文件 `error_stack.h` 第 58 行定义.

__D [3/6]

```
#define _D(
    n,
    v,
    s,
    r)
```

值:

```
n##_INDEX,
```

在文件 `error_stack.h` 第 58 行定义.

__D [4/6]

```
#define _D(
    n,
    v,
    s,
    r)
```


值:

s,

在文件 [error_stack.h](#) 第 58 行定义.

__D [5/6]

```
#define _D(
    n,
    v,
    s,
    r)
```

值:

```
if (strcmp(#n, err_code_name) == 0) \
    return v;
```

在文件 [error_stack.h](#) 第 58 行定义.

__D [6/6]

```
#define _D(
    n,
    v,
    s,
    r)
```

值:

n = (int)v,

在文件 [error_stack.h](#) 第 58 行定义.

__PH1__

```
#define _PH1_ "{}"
```

在文件 [error_stack.h](#) 第 18 行定义.

__PH2__

```
#define _PH2_ "{}"
```

在文件 [error_stack.h](#) 第 19 行定义.

__PH3__

```
#define _PH3_ "{}"
```

在文件 [error_stack.h](#) 第 20 行定义.

__PH4__

```
#define _PH4_ "{}"
```

在文件 [error_stack.h](#) 第 21 行定义.

ARCS_ERROR_CODES

```
#define ARCS_ERROR_CODES
```

值:

```
SYSTEM_ERRORS          \|
JOINT_ERRORS           \|
JOINT_EX_ERRORS        \|
EXT_AXIS_ERRORS        \|
SAFETY_INTERFACE_BOARD_ERRORS \|
RTM_ERRORS             \|
TOOL_ERRORS            \|
EX_TOOL_ERRORS         \|
PEDSTRAL_ERRORS        \|
EX_PEDSTRAL_ERRORS     \|
HARDWARE_INTERFACE_ERRORS \|
_D(ARCS_MAX_ERROR_CODE, -1, "Max error code", "suggest...")
```

在文件 [error_stack.h](#) 第 41 行定义.

AUBO_SDK_HAL_ERROR_H

`#define AUBO_SDK_HAL_ERROR_H`
在文件 `error_stack.h` 第 6 行定义.

AUBO_SDK_RTM_ERROR_H

`#define AUBO_SDK_RTM_ERROR_H`
在文件 `error_stack.h` 第 6 行定义.

AUBO_SDK_SYSTEM_ERROR_H

`#define AUBO_SDK_SYSTEM_ERROR_H`
在文件 `error_stack.h` 第 6 行定义.

EX_PEDSTRAL_ERRORS

`#define EX_PEDSTRAL_ERRORS`

值:

```
_D(EX_BASE_EC_LOW_VOLTAGE, 50101, "pedstral error: low voltage", "check power supply voltage and battery status") \
_D(EX_BASE_EC_OVER_TEMPERATURE_RES, 50102, "pedstral error: resistor over temperature", "check braking resistor
  temperature and cooling condition") \
_D(EX_BASE_EC_OVER_TARGET_BRAKE_OPEN_VOLT, 50103, "pedstral error: input voltage close to or exceeds regenerative brake
  activation voltage", "check input power voltage and regenerative braking configuration") \
_D(EX_BASE_EC_RES_BREAKAGE, 50104, "pedstral error: brake resistor breakage", "check whether the brake resistor is
  disconnected or damaged") \
_D(EX_BASE_EC_IMU_CALIBRATE, 50105, "pedstral error: IMU calibration required", "perform IMU calibration according to
  maintenance procedure") \
_D(EX_BASE_EC_TEMP_SENSOR_SHORT, 50106, "pedstral error: temperature sensor short circuit", "check temperature sensor
  wiring and solder joints for short circuit") \
_D(EX_BASE_EC_TEMP_SENSOR_BREAK, 50107, "pedstral error: temperature sensor open circuit", "check temperature sensor
  connection and cable continuity") \
_D(EX_BASE_EC_96V_INSTANT_OVER_VOLTAGE, 50108, "pedstral error: 96V instantaneous over voltage after regenerative
  braking", "check regenerative braking behavior and power bus voltage") \
_D(EX_BASE_EC_UNKNOWN, 50200, "pedstral error: unknown error", "check controller logs and hardware status")
```

EX_TOOL_ERRORS

`#define EX_TOOL_ERRORS`

值:

```
_D(EX_TOOL_EC_LOW_VOLTAGE, 40101, "tool error: low voltage", "check tool power supply voltage") \
_D(EX_TOOL_EC_FORCESENSOR_COMM, 40102, "tool error: external force sensor communication error", "check external force
  sensor communication connection") \
_D(EX_TOOL_485_SENDFULL_COMM, 40103, "tool error: 485 transparent transmission buffer full", "check 485 communication load
  and transmission frequency") \
_D(EX_TOOL_FORCESENSOR_FILTER_ZERO, 40104, "tool error: force sensor filter parameter is zero", "check force sensor filter
  parameter configuration") \
_D(EX_TOOL_EC_UNKNOWN, 40200, "tool error: unknown error", "check controller logs and hardware status")
```

EXT_AXIS_ERRORS

`#define EXT_AXIS_ERRORS`

HAL_ERRORS

`#define HAL_ERRORS`

值:

```
JOINT_ERRORS //
JOINT_EX_ERRORS //
EXT_AXIS_ERRORS //
SAFETY_INTERFACE_BOARD_ERRORS //
TOOL_ERRORS //
EX_TOOL_ERRORS //
PEDSTRAL_ERRORS //
EX_PEDSTRAL_ERRORS //
HARDWARE_INTERFACE_ERRORS
```

HARDWARE_INTERFACE_ERRORS

`#define HARDWARE_INTERFACE_ERRORS`

JOINT_ERRORS

```
#define JOINT_ERRORS
```

JOINT_EX_ERRORS

```
#define JOINT_EX_ERRORS
```

PEDSTRAL_ERRORS

```
#define PEDSTRAL_ERRORS
```

值:

```
_D(PKG_LOST, 50001, "Lost package from pedestal", "suggest...") \
_D(PEDSTRAL_OFFLINE, 50002, "pedestal error: pedestal may be offline", "(a) Check pedestal's hardware. (b) Check pedestal's id.") \
_D(PEDSTRAL_ERR_BOOTLOADER, 50003, "pedestal error: The pedestal is in bootloader mode. Retry firmware update. ", "suggest...")
```

RTM_ERRORS

```
#define RTM_ERRORS
```

SAFETY_INTERFACE_BOARD_ERRORS

```
#define SAFETY_INTERFACE_BOARD_ERRORS
```

值:

```
_D(IFB_ERR_ROBOTTYPE, 20001, "Robot error type!", "suggest...") \
_D(IFB_ERR_ADXL_SENS, 20002, "Base Acceleration sensor error!", "suggest...") \
_D(IFB_ERR_EN_LINE, 20003, "Encoder line error!", "suggest...") \
_D(IFB_ERR_ENTER_HDG_MODE, 20004, "Robot enter handguide mode!", "suggest...") \
_D(IFB_ERR_EXIT_HDG_MODE, 20005, "Robot exit handguide mode!", "suggest...") \
_D(IFB_ERR_MAC_DATA_BREAK, 20006, "MAC data break!", "suggest...") \
_D(IFB_ERR_DRV_FIRMWARE_VERSION, 20007, "Motor driver firmware version error!", "suggest...") \
_D(INIT_ERR_EN_DRV, 20008, "Motor driver enable failed!", "suggest...") \
_D(INIT_ERR_EN_AUTO_BACK, 20009, "Motor driver enable auto back failed!", "suggest...") \
_D(INIT_ERR_EN_CUR_LOOP, 20010, "Motor driver enable current loop failed!", "suggest...") \
_D(INIT_ERR_SET_TAG_CUR, 20011, "Motor driver set target current failed!", "suggest...") \
_D(INIT_ERR_RELEASE_BRAKE, 20012, "Motor driver release brake failed!", "suggest...") \
_D(INIT_ERR_EN_POS_LOOP, 20013, "Motor driver enable position loop failed!", "suggest...") \
_D(INIT_ERR_SET_MAX_ACC, 20014, "Motor set max accelerate failed!", "suggest...") \
_D(SAFETY_ERR_PROTECTION_STOP_TIMEOUT, 20015, "Protective stop timeout!", "suggest...") \
_D(SAFETY_ERR_REDUCED_MODE_TIMEOUT, 20016, "Reduced mode timeout!", "suggest...") \
_D(SYS_ERR_MCU_COM, 20017, "Robot system error: mcu communication error!", "suggest...") \
_D(SYS_ERR_RS485_COM, 20018, "Robot system error: RS485 communication error!", "suggest...") \
_D(IFB_ERR_DISCONNECTED, 20019, "Interface board may be disconnected. Please check connection between IPC and Interface board.", "suggest...") \
_D(IFB_ERR_PAYLOAD_ERROR, 20020, "Payload error.", "suggest...") \
_D(IFB_OFFLINE, 20021, "ifaceboard error: ifaceboard may be offline", "(a) Check ifaceboard's hardware. (b) Check ifaceboard's id.") \
_D(IFB_ERR_BOOTLOADER, 20022, "ifaceboard error: The ifaceboard is in bootloader mode. Retry firmware update. ", "suggest...") \
_D(IFB_SLAVE_OFFLINE, 20023, "interface slave board error: interface slave board may be offline", "(a) Check interface slave board's hardware. (b) Check interface slave board's id.") \
_D(IFB_SLAVE_ERR_BOOTLOADER, 20024, "interface slave board error: The interface slave board is in bootloader mode. Retry firmware update. ", "suggest...") \
_D(IFB_TOOL_ERR_ADXL_SENS, 20025, "Tool Acceleration sensor error!", "suggest...") \
_D(HANDLE_OFFLINE, 20026, "handle error: handle may be offline", "(a) Check handle's hardware. (b) Check handle's id.") \
_D(HANDLE_ERR_BOOTLOADER, 20027, "handle error: The handle is in bootloader mode. Retry firmware update. ", "suggest...") \
_D(IFB_POWERLOSS_OFFLINE, 20028, "interface powerloss board error: interface powerloss board may be offline", "(a) Check interface powerloss board's hardware. (b) Check interface powerloss board's id.") \
_D(IFB_POWERLOSS_ERR_BOOTLOADER, 20029, "interface powerloss board error: The interface powerloss board is in bootloader mode. Retry firmware update. ", "suggest...") \
_D(HANDLE_COMM_ERROR, 20030, "handle error: handle comm error", "(a) Check handle's hardware. (b) Check handle's id.")
```

SYSTEM_ERRORS

```
#define SYSTEM_ERRORS
```

TOOL_ERRORS

```
#define TOOL_ERRORS
```

值:

```
_D(TOOL_FLASH_VERIFY_FAILED, 40001, "Flash write verify failed", "suggest...") \
_D(TOOL_PROGRAM_CRC_FAILED, 40002, "Program flash checksum failed during bootloading", "suggest...") \
```

```

_D(TOOL_PROGRAM_CRC_FAILED2, 40003, "Program flash checksum failed at runtime", "suggest...") \
_D(TOOL_ID_UNDEFINED, 40004, "Tool ID is undefined", "suggest...") \
_D(TOOL_ILLEGAL_BL_CMD, 40005, "Illegal bootloader command", "suggest...") \
_D(TOOL_FW_WRONG, 40006, "Wrong firmware at the joint", "suggest...") \
_D(TOOL_HW_INVALID, 40007, "Invalid hardware revision", "suggest...") \
_D(TOOL_SHORT_CURCUIT_H, 40011, "Short circuit detected on Digital Output: " _PH1_ " high side", "suggest...") \
_D(TOOL_SHORT_CURCUIT_L, 40012, "Short circuit detected on Digital Output: " _PH1_ " low side", "suggest...") \
_D(TOOL_AVERAGE_CURR_HIGH, 40013, "10 second Average tool IO Current of " _PH1_ " A is outside of the allowed range.",
"suggest...") \
_D(TOOL_POWER_PIN_OVER_CURR, 40014, "Current of " _PH1_ " A on the POWER pin is outside of the allowed range.",
"suggest...") \
_D(TOOL_DOUT_PIN_OVER_CURR, 40015, "Current of " _PH1_ " A on the Digital Output pins is outside of the allowed range.",
"suggest...") \
_D(TOOL_GROUND_PIN_OVER_CURR, 40016, "Current of " _PH1_ " A on the ground pin is outside of the allowed range.",
"suggest...") \
_D(TOOL_RX_FRAMING, 40021, "RX framing error", "suggest...") \
_D(TOOL_RX_PARITY, 40022, "RX Parity error", "suggest...") \
_D(TOOL_48V_LOW, 40031, "48V input is too low", "suggest...") \
_D(TOOL_48V_HIGH, 40032, "48V input is too high", "suggest...") \
_D(TOOL_ERR_OFFLINE, 40033, "tool error: tool may be offline", "(a) Check tool's hardware. (b) Check joint's id.") \
_D(TOOL_ERR_BOOTLOADER, 40034, "tool error: The tool is in bootloader mode. Retry firmware update.", "suggest...")

```

12.7 error_stack.h

浏览该文件的文档.

```

00001 /** @file error_stack.h
00002  * @brief 汇总错误码
00003  */
00004 #ifndef AUBO_SDK_ERROR_STACK_H
00005 #define AUBO_SDK_ERROR_STACK_H
00006
00007 #include <stdio.h>
00008 #include <stdint.h>
00009 #include <string.h>
00010 #include <string>
00011 #include <sstream>
00012 #include <iomanip>
00013
00014 #include <aubo/global_config.h>
00015
00016 // 格式化占位符, 默认是 fmt 的格式
00017 #ifndef _PH1_
00018 #define _PH1_ "{}"
00019 #define _PH2_ "{}"
00020 #define _PH3_ "{}"
00021 #define _PH4_ "{}"
00022 #endif
00023
00024 namespace arcs {
00025 namespace error_stack {
00026
00027 constexpr int ARCS_ABI_EXPORT codeCompose(int aa, int bb, int cccc)
00028 {
00029     return (int)((aa * 1000000) + (bb * 10000) + cccc);
00030 }
00031
00032 constexpr int ARCS_ABI_EXPORT mod(int x)
00033 {
00034     return (x % 1000000);
00035 }
00036
00037 #include <aubo/error_stack/hal_error.h>
00038 #include <aubo/error_stack/rtm_error.h>
00039 #include <aubo/error_stack/system_error.h>
00040
00041 #define ARCS_ERROR_CODES
00042     SYSTEM_ERRORS
00043     JOINT_ERRORS
00044     JOINT_EX_ERRORS
00045     EXT_AXIS_ERRORS
00046     SAFETY_INTERFACE_BOARD_ERRORS
00047     RTM_ERRORS
00048     TOOL_ERRORS
00049     EX_TOOL_ERRORS
00050     PEDSTRAL_ERRORS
00051     EX_PEDSTRAL_ERRORS
00052     HARDWARE_INTERFACE_ERRORS
00053     _D(ARCS_MAX_ERROR_CODE, -1, "Max error code", "suggest...")
00054
00055 // 错误代码枚举
00056 enum ErrorCodes
00057 {
00058     #define _D(n, v, s, r) n = (int)v,
00059     ARCS_ERROR_CODES

```

```

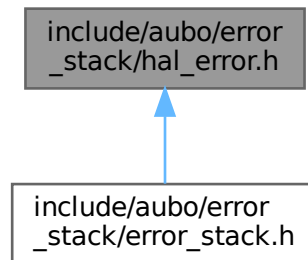
00060 #undef _D
00061 };
00062
00063 inline int str2ErrorCode(const char *err_code_name)
00064 {
00065     #define _D(n, v, s, r) \
00066         if (strcmp(#n, err_code_name) == 0) \
00067             return v;
00068     ARCS_ERROR_CODES
00069     #undef _D
00070     return ARCS_MAX_ERROR_CODE;
00071 }
00072
00073 inline const char *errorCode2Str(int err_code)
00074 {
00075     static const char *errcode_str[] = {
00076     #define _D(n, v, s, r) s,
00077     ARCS_ERROR_CODES
00078     #undef _D
00079     };
00080     enum arcs_index
00081     {
00082     #define _D(n, v, s, r) n##_INDEX,
00083     ARCS_ERROR_CODES
00084     #undef _D
00085     };
00086     int index = -1;
00087
00088     #define _D(n, v, s, r) \
00089         if (err_code == v) \
00090             index = n##_INDEX;
00091     ARCS_ERROR_CODES
00092     #undef _D
00093     if (index == -1) {
00094         index = ARCS_MAX_ERROR_CODE_INDEX;
00095     }
00096     return errcode_str[(unsigned)index];
00097 }
00098
00099 inline std::ostream &dump(std::ostream &os)
00100 {
00101     #define _D(n, v, s, r) \
00102         os << std::setw(20) << #n << "\t" << v << "\t" << s << "\t" << r \
00103         << std::endl;
00104     ARCS_ERROR_CODES
00105     #undef _D
00106     return os;
00107 }
00108
00109 // namespace error_stack
00110 // namespace arcs
00111 #endif // AUBO_SDK_ERROR_STACK_H

```

12.8 include/aubo/error_stack/hal_error.h 文件参考

定义硬件抽象层的错误码

此图展示该文件被哪些文件直接或间接地引用了:



宏定义

- `#define JOINT_ERRORS`
- `#define JOINT_EX_ERRORS`
- `#define EXT_AXIS_ERRORS`
- `#define TOOL_ERRORS`
- `#define EX_TOOL_ERRORS`
- `#define PEDSTRAL_ERRORS`
- `#define EX_PEDSTRAL_ERRORS`
- `#define SAFETY_INTERFACE_BOARD_ERRORS`
- `#define HARDWARE_INTERFACE_ERRORS`
- `#define HAL_ERRORS`

12.8.1 详细描述

定义硬件抽象层的错误码
在文件 `hal_error.h` 中定义.

12.8.2 宏定义说明

EX_PEDSTRAL_ERRORS

`#define EX_PEDSTRAL_ERRORS`

值:

```

_D(EX_BASE_EC_LOW_VOLTAGE, 50101, "pedstral error: low voltage", "check power supply voltage and battery status") \
_D(EX_BASE_EC_OVER_TEMPERATURE_RES, 50102, "pedstral error: resistor over temperature", "check braking resistor \
temperature and cooling condition") \
_D(EX_BASE_EC_OVER_TARGET_BRAKE_OPEN_VOLT, 50103, "pedstral error: input voltage close to or exceeds regenerative brake \
activation voltage", "check input power voltage and regenerative braking configuration") \
_D(EX_BASE_EC_RES_BREAKAGE, 50104, "pedstral error: brake resistor breakage", "check whether the brake resistor is \
disconnected or damaged") \
_D(EX_BASE_EC_IMU_CALIBRATE, 50105, "pedstral error: IMU calibration required", "perform IMU calibration according to \
maintenance procedure") \
_D(EX_BASE_EC_TEMP_SENSOR_SHORT, 50106, "pedstral error: temperature sensor short circuit", "check temperature sensor \
wiring and solder joints for short circuit") \
_D(EX_BASE_EC_TEMP_SENSOR_BREAK, 50107, "pedstral error: temperature sensor open circuit", "check temperature sensor \
connection and cable continuity") \
_D(EX_BASE_EC_96V_INSTANT_OVER_VOLTAGE, 50108, "pedstral error: 96V instantaneous over voltage after regenerative \
braking", "check regenerative braking behavior and power bus voltage") \
_D(EX_BASE_EC_UNKNOWN, 50200, "pedstral error: unknown error", "check controller logs and hardware status")
  
```

在文件 `error_stack.h` 第 258 行定义.

EX_TOOL_ERRORS

```
#define EX_TOOL_ERRORS
```

值:

```

_D(EX_TOOL_EC_LOW_VOLTAGE, 40101, "tool error: low voltage", "check tool power supply voltage") \
_D(EX_TOOL_EC_FORCESENSOR_COMM, 40102, "tool error: external force sensor communication error", "check external force
  sensor communication connection") \
_D(EX_TOOL_485_SENDFULL_COMM, 40103, "tool error: 485 transparent transmission buffer full", "check 485 communication load
  and transmission frequency") \
_D(EX_TOOL_FORCESENSOR_FILTER_ZERO, 40104, "tool error: force sensor filter parameter is zero", "check force sensor filter
  parameter configuration") \
_D(EX_TOOL_EC_UNKNOWN, 40200, "tool error: unknown error", "check controller logs and hardware status")

```

在文件 [error_stack.h](#) 第 246 行定义。

EXT_AXIS_ERRORS

```
#define EXT_AXIS_ERRORS
```

在文件 [error_stack.h](#) 第 157 行定义。

HAL_ERRORS

```
#define HAL_ERRORS
```

值:

```

JOINT_ERRORS           //
JOINT_EX_ERRORS        //
EXT_AXIS_ERRORS         //
SAFETY_INTERFACE_BOARD_ERRORS //
TOOL_ERRORS            //
EX_TOOL_ERRORS          //
PEDSTRAL_ERRORS         //
EX_PEDSTRAL_ERRORS      //
HARDWARE_INTERFACE_ERRORS

```

在文件 [error_stack.h](#) 第 339 行定义。

HARDWARE_INTERFACE_ERRORS

```
#define HARDWARE_INTERFACE_ERRORS
```

在文件 [error_stack.h](#) 第 301 行定义。

JOINT_ERRORS

```
#define JOINT_ERRORS
```

在文件 [error_stack.h](#) 第 20 行定义。

JOINT_EX_ERRORS

```
#define JOINT_EX_ERRORS
```

在文件 [error_stack.h](#) 第 67 行定义。

PEDSTRAL_ERRORS

```
#define PEDSTRAL_ERRORS
```

值:

```

_D(PKG_LOST, 50001, "Lost package from pedestal", "suggest...") \
_D(PEDSTRAL_OFFLINE, 50002, "pedestal error: pedestal may be offline", "(a) Check pedestal's hardware. (b) Check
  pedestal's id.") \
_D(PEDSTRAL_ERR_BOOTLOADER, 50003, "pedestal error: The pedestal is in bootloader mode. Retry firmware update. ",
  "suggest...")

```

在文件 [error_stack.h](#) 第 253 行定义。

SAFETY_INTERFACE_BOARD_ERRORS

```
#define SAFETY_INTERFACE_BOARD_ERRORS
```

值:

```

_D(IFB_ERR_ROBOTTYPE, 20001, "Robot error type!", "suggest...") \
_D(IFB_ERR_ADXL_SENS, 20002, "Base Acceleration sensor error!", "suggest...") \
_D(IFB_ERR_EN_LINE, 20003, "Encoder line error!", "suggest...") \
_D(IFB_ERR_ENTER_HDG_MODE, 20004, "Robot enter handguide mode!", "suggest...") \
_D(IFB_ERR_EXIT_HDG_MODE, 20005, "Robot exit handguide mode!", "suggest...") \

```

```

_D(IFB_ERR_MAC_DATA_BREAK, 20006, "MAC data break!", "suggest...") \
_D(IFB_ERR_DRV_FIRMWARE_VERSION, 20007, "Motor driver firmware version error!", "suggest...") \
_D(INIT_ERR_EN_DRV, 20008, "Motor driver enable failed!", "suggest...") \
_D(INIT_ERR_EN_AUTO_BACK, 20009, "Motor driver enable auto back failed!", "suggest...") \
_D(INIT_ERR_EN_CUR_LOOP, 20010, "Motor driver enable current loop failed!", "suggest...") \
_D(INIT_ERR_SET_TAG_CUR, 20011, "Motor driver set target current failed!", "suggest...") \
_D(INIT_ERR_RELEASE_BRAKE, 20012, "Motor driver release brake failed!", "suggest...") \
_D(INIT_ERR_EN_POS_LOOP, 20013, "Motor driver enable position loop failed!", "suggest...") \
_D(INIT_ERR_SET_MAX_ACC, 20014, "Motor set max accelerate failed!", "suggest...") \
_D(SAFETY_ERR_PROTECTION_STOP_TIMEOUT, 20015, "Protective stop timeout!", "suggest...") \
_D(SAFETY_ERR_REDUCED_MODE_TIMEOUT, 20016, "Reduced mode timeout!", "suggest...") \
_D(SYS_ERR_MCU_COM, 20017, "Robot system error: mcu communication error!", "suggest...") \
_D(SYS_ERR_RS485_COM, 20018, "Robot system error: RS485 communication error!", "suggest...") \
_D(IFB_ERR_DISCONNECTED, 20019, "Interface board may be disconnected. Please check connection between IPC and Interface
board.", "suggest...") \
_D(IFB_ERR_PAYLOAD_ERROR, 20020, "Payload error.", "suggest...") \
_D(IFB_OFFLINE, 20021, "ifaceboard error: ifaceboard may be offline", "(a) Check ifaceboard's hardware. (b) Check
ifaceboard's id.") \
_D(IFB_ERR_BOOTLOADER, 20022, "ifaceboard error: The ifaceboard is in bootloader mode. Retry firmware update. ",
"suggest...") \
_D(IFB_SLAVE_OFFLINE, 20023, "interface slave board error: interface slave board may be offline", "(a) Check interface
slave board's hardware. (b) Check interface slave board's id.") \
_D(IFB_SLAVE_ERR_BOOTLOADER, 20024, "interface slave board error: The interface slave board is in bootloader mode. Retry
firmware update. ", "suggest...") \
_D(IFB_TOOL_ERR_ADXL_SENS, 20025, "Tool Acceleration sensor error!", "suggest...") \
_D(HANDLE_OFFLINE, 20026, "handle error: handle may be offline", "(a) Check handle's hardware. (b) Check handle's id.") \
_D(HANDLE_ERR_BOOTLOADER, 20027, "handle error: The handle is in bootloader mode. Retry firmware update. ", "suggest...") \
_D(IFB_POWERLOSS_OFFLINE, 20028, "interface powerloss board error: interface powerloss board may be offline", "(a) Check
interface powerloss board's hardware. (b) Check interface powerloss board's id.") \
_D(IFB_POWERLOSS_ERR_BOOTLOADER, 20029, "interface powerloss board error: The interface powerloss board is in bootloader
mode. Retry firmware update. ", "suggest...") \
_D(HANDLE_COMM_ERROR, 20030, "handle error: handle comm error", "(a) Check handle's hardware. (b) Check handle's id.")

```

在文件 [error_stack.h](#) 第 269 行定义。

TOOL_ERRORS

```
#define TOOL_ERRORS
```

值:

```

_D(TOOL_FLASH_VERIFY_FAILED, 40001, "Flash write verify failed", "suggest...") \
_D(TOOL_PROGRAM_CRC_FAILED, 40002, "Program flash checksum failed during bootloading", "suggest...") \
_D(TOOL_PROGRAM_CRC_FAILED2, 40003, "Program flash checksum failed at runtime", "suggest...") \
_D(TOOL_ID_UNDEFINED, 40004, "Tool ID is undefined", "suggest...") \
_D(TOOL_ILLEGAL_BL_CMD, 40005, "Illegal bootloader command", "suggest...") \
_D(TOOL_FW_WRONG, 40006, "Wrong firmware at the joint", "suggest...") \
_D(TOOL_HW_INVALID, 40007, "Invalid hardware revision", "suggest...") \
_D(TOOL_SHORT_CURCUI_T_H, 40011, "Short circuit detected on Digital Output: " _PH1_ " high side", "suggest...") \
_D(TOOL_SHORT_CURCUI_T_L, 40012, "Short circuit detected on Digital Output: " _PH1_ " low side", "suggest...") \
_D(TOOL_AVERAGE_CURR_HIGH, 40013, "10 second Average tool I0 Current of " _PH1_ " A is outside of the allowed range.",
"suggest...") \
_D(TOOL_POWER_PIN_OVER_CURR, 40014, "Current of " _PH1_ " A on the POWER pin is outside of the allowed range.",
"suggest...") \
_D(TOOL_DOUT_PIN_OVER_CURR, 40015, "Current of " _PH1_ " A on the Digital Output pins is outside of the allowed range.",
"suggest...") \
_D(TOOL_GROUND_PIN_OVER_CURR, 40016, "Current of " _PH1_ " A on the ground pin is outside of the allowed range.",
"suggest...") \
_D(TOOL_RX_FRAMING, 40021, "RX framing error", "suggest...") \
_D(TOOL_RX_PARITY, 40022, "RX Parity error", "suggest...") \
_D(TOOL_48V_LOW, 40031, "48V input is too low", "suggest...") \
_D(TOOL_48V_HIGH, 40032, "48V input is too high", "suggest...") \
_D(TOOL_ERR_OFFLINE, 40033, "tool error: tool may be offline", "(a) Check tool's hardware. (b) Check joint's id.") \
_D(TOOL_ERR_BOOTLOADER, 40034, "tool error: The tool is in bootloader mode. Retry firmware update. ", "suggest...")

```

在文件 [error_stack.h](#) 第 225 行定义。

12.9 hal_error.h

[浏览该文件的文档.](#)

```

00001 /** @file hal_error.h
00002  * @brief 定义硬件抽象层的错误码
00003  */
00004 #ifndef AUBO_SDK_HAL_ERROR_H
00005 #define AUBO_SDK_HAL_ERROR_H
00006
00007 // 缩写说明
00008 // JNT: joint
00009 // PDL: pedstral
00010 // TP: teach pendant
00011 // COMM: communication
00012 // ENC: encoder
00013 // CURR: current

```



```

00014 // POS: position
00015 // PKG: package
00016 // PROG: program
00017
00018 // clang-format off
00019 #define JOINT_ERRORS \
00020     _D(JOINT_ERR_OVER_CURRENT, 10001, "joint" _PH1_ " error: over current", "(a) Check for short circuit. (b) Do a
Complete rebooting sequence. (c) If this happens more than two times in a row, replace joint") \
00021     _D(JOINT_ERR_OVER_VOLTAGE, 10002, "joint" _PH1_ " error: over voltage", "(a) Do a Complete rebooting sequence.
(b) Check 48 V Power supply, current distributor, energy eater and Control Board for issues") \
00022     _D(JOINT_ERR_LOW_VOLTAGE, 10003, "joint" _PH1_ " error: low voltage", "(a) Do a Complete rebooting sequence.
(b) Check for short circuit in robot arm. (c) Check 48 V Power supply, current distributor, energy eater and Control
Board for issues") \
00023     _D(JOINT_ERR_OVER_TEMP, 10004, "joint" _PH1_ " error: over temperature", "(a) Check robot' s environment and
make sure the robot is operating within recommended limits. (b) Do a Complete rebooting sequence") \
00024     _D(JOINT_ERR_HALL, 10005, "joint" _PH1_ " error: hall", "suggest...") \
00025     _D(JOINT_ERR_ENCODER, 10006, "joint" _PH1_ " error: encoder", "Check encoder connections") \
00026     _D(JOINT_ERR_ABS_ENCODER, 10007, "joint" _PH1_ " error: abs encoder", "suggest...") \
00027     _D(JOINT_ERR_Q_CURRENT, 10008, "joint" _PH1_ " error: detect current", "suggest...") \
00028     _D(JOINT_ERR_ENC_POLL, 10009, "joint" _PH1_ " error: encoder pollution", "suggest...") \
00029     _D(JOINT_ERR_ENC_Z_SIGNAL, 10010, "joint" _PH1_ " error: encoder z signal", "suggest...") \
00030     _D(JOINT_ERR_ENC_CAL, 10011, "joint" _PH1_ " error: encoder calibrate", "suggest...") \
00031     _D(JOINT_ERR_IMU_SENS, 10012, "joint" _PH1_ " error: IMU sensor", "suggest...") \
00032     _D(JOINT_ERR_TEMP_SENS, 10013, "joint" _PH1_ " error: TEMP sensor", "suggest...") \
00033     _D(JOINT_ERR_CAN_BUS, 10014, "joint" _PH1_ " error: CAN bus error", "suggest...") \
00034     _D(JOINT_ERR_SYS_CUR, 10015, "joint" _PH1_ " error: system current error", "suggest...") \
00035     _D(JOINT_ERR_SYS_POS, 10016, "joint" _PH1_ " error: system position error", "suggest...") \
00036     _D(JOINT_ERR_OVER_SP, 10017, "joint" _PH1_ " error: over speed", "suggest...") \
00037     _D(JOINT_ERR_OVER_ACC, 10018, "joint" _PH1_ " error: over accelerate", "suggest...") \
00038     _D(JOINT_ERR_TRACE, 10019, "joint" _PH1_ " error: trace accuracy", "suggest...") \
00039     _D(JOINT_ERR_TAG_POS_OVER, 10020, "joint" _PH1_ " error: target position out of range", "suggest...") \
00040     _D(JOINT_ERR_TAG_SP_OVER, 10021, "joint" _PH1_ " error: target speed out of range", "suggest...") \
00041     _D(JOINT_ERR_COLLISION, 10022, "joint" _PH1_ " error: collision", "suggest...") \
00042     _D(JOINT_ERR_COMMON, 10023, "joint" _PH1_ " error: unknown error. Check communication with joint.", "suggest...")
\
00043     _D(JOINT_ERR_SWITCH_SERVO_MODE, 10024, "joint" _PH1_ " error: switch servo mode timeout.", "suggest...") \
00044     _D(JOINT_ERR_MOTOR_STUCK, 10025, "joint" _PH1_ " error: motor stucked.", "suggest...") \
00045     _D(JOINT_ERR_REDUCER_OVER_TEMP, 10026, "joint" _PH1_ " error: reducer over temperature", "(a) Check robot' s
environment and make sure the robot is operating within recommended limits. (b) Do a Complete rebooting sequence") \
00046     _D(JOINT_ERR_REDUCER_NTC, 10027, "joint" _PH1_ " error: reducer TEMP sensor failure", "(a) Check robot' s
environment and make sure the robot is operating within recommended limits. (b) Do a Complete rebooting sequence") \
00047     _D(JOINT_ERR_ABS_MULTITURN, 10028, "joint" _PH1_ " error: absolute encoder multiturn error", "(a) Check robot' s
environment and make sure the robot is operating within recommended limits. (b) Do a Complete rebooting sequence") \
00048     _D(JOINT_ERR_ADC_ZERO_OFFSET, 10029, "joint" _PH1_ " error: ADC zero offset failure", "(a) Check robot' s
environment and make sure the robot is operating within recommended limits. (b) Do a Complete rebooting sequence") \
00049     _D(JOINT_ERR_SHORT_CIRCUIT, 10030, "joint" _PH1_ " error: short circuit", "(a) Check robot' s environment and
make sure the robot is operating within recommended limits. (b) Do a Complete rebooting sequence") \
00050     _D(JOINT_ERR_PHASE_LOST, 10031, "joint" _PH1_ " error: motor phase lost", "(a) Check robot' s environment and
make sure the robot is operating within recommended limits. (b) Do a Complete rebooting sequence") \
00051     _D(JOINT_ERR_BRAKE, 10032, "joint" _PH1_ " error: brake failure", "(a) Check robot' s environment and make sure
the robot is operating within recommended limits. (b) Do a Complete rebooting sequence") \
00052     _D(JOINT_ERR_FIRMWARE_UPDATE, 10033, "joint" _PH1_ " error: firmware update failure", "(a) Check robot' s
environment and make sure the robot is operating within recommended limits. (b) Do a Complete rebooting sequence") \
00053     _D(JOINT_ERR_BATTERY_LOW, 10034, "joint" _PH1_ " error: battery low", "(a) Check robot' s environment and make
sure the robot is operating within recommended limits. (b) Do a Complete rebooting sequence") \
00054     _D(JOINT_ERR_PHASE_ALIGN, 10035, "joint" _PH1_ " error: phase align", "(a) Check robot' s environment and make
sure the robot is operating within recommended limits. (b) Do a Complete rebooting sequence") \
00055     _D(JOINT_ERR_CAN_HW_FAULT, 10036, "joint" _PH1_ " error: CAN bus hw fault", "(a) Check robot' s environment and
make sure the robot is operating within recommended limits. (b) Do a Complete rebooting sequence") \
00056     _D(JOINT_ERR_POS_DISCONTINUOUS, 10037, "joint" _PH1_ " error: target position discontinuous", "(a) Check robot'
s environment and make sure the robot is operating within recommended limits. (b) Do a Complete rebooting sequence")
\
00057     _D(JOINT_ERR_POS_INIT, 10038, "joint" _PH1_ " error: position initialization failure", "(a) Check robot' s
environment and make sure the robot is operating within recommended limits. (b) Do a Complete rebooting sequence") \
00058     _D(JOINT_ERR_TORQUE_SENSOR, 10039, "joint" _PH1_ " error: torque sensor failure", "(a) Check robot' s
environment and make sure the robot is operating within recommended limits. (b) Do a Complete rebooting sequence") \
00059     _D(JOINT_ERR_OFFLINE, 10040, "joint" _PH1_ " error: joint may be offline", "(a) Check joint's hardware. (b)
Check joint's id.") \
00060     _D(JOINT_ERR_BOOTLOADER, 10041, "joint" _PH1_ " error: The joint is in bootloader mode. Retry firmware update.
", "suggest...") \
00061     _D(JOINT_ERR_SLAVE_OFFLINE, 10042, "slave joint" _PH1_ " error: slave joint may be offline", "(a) Check slave
joint's hardware. (b) Check slave joint's id.") \
00062     _D(JOINT_ERR_SLAVE_BOOTLOADER, 10043, "slave joint" _PH1_ " error: The slave joint is in bootloader mode. Retry
firmware update. ", "suggest...") \
00063     _D(JOINT_ERR_ETHERCAT_BUS, 10044, "joint" _PH1_ " error: ETHERCAT bus error", "suggest...")
00064
00065 // Joint extended error codes (ex_err_code), starting from 10401
00066 #define JOINT_EX_ERRORS \
00067     _D(EX_JOINT_EC_SHORT_CIRCUIT, 10401, "joint" _PH1_ " error: short circuit protection", "suggest...") \
00068     _D(EX_JOINT_EC_SHORT_CURRENT, 10402, "joint" _PH1_ " error: over current", "suggest...") \
00069     _D(EX_JOINT_EC_PHASEA_CURRENT, 10403, "joint" _PH1_ " error: phase A over current", "suggest...") \
00070     _D(EX_JOINT_EC_PHASEB_CURRENT, 10404, "joint" _PH1_ " error: phase B over current", "suggest...") \
00071     _D(EX_JOINT_EC_PHASEC_CURRENT, 10405, "joint" _PH1_ " error: phase C over current", "suggest...") \
00072     _D(EX_JOINT_EC_PHASE_CURRENT, 10406, "joint" _PH1_ " error: phase over current", "suggest...") \
00073     _D(EX_JOINT_EC_MOTOR_PHASE_LOSE, 10407, "joint" _PH1_ " error: motor phase loss", "suggest...") \
00074     _D(EX_JOINT_EC_BUS_OVER_VOLTAGE, 10408, "joint" _PH1_ " error: bus over voltage", "suggest...") \
00075     _D(EX_JOINT_EC_BUS_LOW_VOLTAGE, 10409, "joint" _PH1_ " error: bus under voltage", "suggest...") \

```

```

00076 _D(EX_JOINT_EC_OVERLOAD, 10410, "joint" _PH1_ " error: overload", "suggest...") \
00077 _D(EX_JOINT_EC_IPM_OVER_TEMP, 10411, "joint" _PH1_ " error: IPM over temperature", "suggest...") \
00078 _D(EX_JOINT_EC_REDUCER_OVER_TEMP, 10412, "joint" _PH1_ " error: reducer over temperature", "suggest...") \
00079 _D(EX_JOINT_EC_ADC_ZERO_OFFSET, 10413, "joint" _PH1_ " error: ADC zero offset", "suggest...") \
00080 _D(EX_JOINT_EC_REDUCER_NTC, 10414, "joint" _PH1_ " error: reducer NTC fault", "suggest...") \
00081 _D(EX_JOINT_EC_IPM_NTC, 10415, "joint" _PH1_ " error: IPM NTC fault", "suggest...") \
00082 _D(EX_JOINT_EC_TORQUE_SENSOR, 10416, "joint" _PH1_ " error: torque sensor fault", "suggest...") \
00083 _D(EX_JOINT_EC_TORQUE_SENSOR_COMM, 10417, "joint" _PH1_ " error: torque sensor communication fault",
"suggest...") \
00084 _D(EX_JOINT_EC_MOTOR_ABS_ENC_COMM, 10418, "joint" _PH1_ " error: motor-side absolute encoder communication
fault", "suggest...") \
00085 _D(EX_JOINT_EC_REDUCER_ABS_ENC_COMM, 10419, "joint" _PH1_ " error: reducer-side absolute encoder communication
fault", "suggest...") \
00086 _D(EX_JOINT_EC_REDUCER_ABS_ENC_DATA, 10420, "joint" _PH1_ " error: reducer-side absolute encoder data channel
disabled warning", "suggest...") \
00087 _D(EX_JOINT_EC_REDUCER_ABS_ENC_CMD, 10421, "joint" _PH1_ " error: reducer-side absolute encoder command invalid
warning", "suggest...") \
00088 _D(EX_JOINT_EC_REDUCER_ABS_ENC_ERR, 10422, "joint" _PH1_ " error: reducer-side absolute encoder fault",
"suggest...") \
00089 _D(EX_JOINT_EC_REDUCER_ABS_ENC_WARNING, 10423, "joint" _PH1_ " error: reducer-side absolute encoder warning",
"suggest...") \
00090 _D(EX_JOINT_EC_BRAKE, 10424, "joint" _PH1_ " error: brake fault", "suggest...") \
00091 _D(EX_JOINT_EC_COMM_HWL, 10425, "joint" _PH1_ " error: communication hardware layer error", "suggest...") \
00092 _D(EX_JOINT_EC_FIRMWARE_UPDATE, 10426, "joint" _PH1_ " error: firmware update failed", "suggest...") \
00093 _D(EX_JOINT_EC_FLASH_OP, 10427, "joint" _PH1_ " error: flash operation failed", "suggest...") \
00094 _D(EX_JOINT_EC_MU_SAVE, 10428, "joint" _PH1_ " error: multi-turn data error", "suggest...") \
00095 _D(EX_JOINT_EC_DEMADATA_LOST, 10429, "joint" _PH1_ " error: calibration zero point data lost", "suggest...") \
00096 _D(EX_JOINT_EC_PARAMETER, 10430, "joint" _PH1_ " error: parameter error", "suggest...") \
00097 _D(EX_JOINT_EC_UVW_LOGIC, 10431, "joint" _PH1_ " error: hall signal fault", "suggest...") \
00098 _D(EX_JOINT_EC_UVW_ABZ, 10432, "joint" _PH1_ " error: incremental encoder fault", "suggest...") \
00099 _D(EX_JOINT_EC_ENC_Z_LOST, 10433, "joint" _PH1_ " error: encoder Z signal lost", "suggest...") \
00100 _D(EX_JOINT_EC_ENC_POLLUTE, 10434, "joint" _PH1_ " error: encoder pollution", "suggest...") \
00101 _D(EX_JOINT_EC_ENC_CALI, 10435, "joint" _PH1_ " error: encoder calibration failed", "suggest...") \
00102 _D(EX_JOINT_EC_MT_ABS_DATA, 10436, "joint" _PH1_ " error: multi-turn absolute data error", "suggest...") \
00103 _D(EX_JOINT_EC_ENC_TYPE_INFO, 10437, "joint" _PH1_ " error: encoder type identified and saved", "suggest...") \
00104 _D(EX_JOINT_EC_ENC_TYPE_ERROR, 10438, "joint" _PH1_ " error: encoder type error", "suggest...") \
00105 _D(EX_JOINT_EC_ENC_VERIFY, 10439, "joint" _PH1_ " error: encoder verification failed", "suggest...") \
00106 _D(EX_JOINT_EC_DUAL_ENC_ERROR, 10440, "joint" _PH1_ " error: dual encoder deviation too large", "suggest...") \
00107 _D(EX_JOINT_EC_DUAL_ENC_EANGLE, 10441, "joint" _PH1_ " error: dual encoder electrical angle deviation too
large", "suggest...") \
00108 _D(EX_JOINT_EC_OBJECT_DICT_ERROR, 10442, "joint" _PH1_ " error: object dictionary data error", "suggest...") \
00109 _D(EX_JOINT_EC_MOTOR_STALL, 10443, "joint" _PH1_ " error: motor stall protection", "suggest...") \
00110 _D(EX_JOINT_EC_ABS_ENC_LOW_VOLT, 10444, "joint" _PH1_ " error: absolute encoder low voltage", "suggest...") \
00111 _D(EX_JOINT_EC_MT_BATTERY_LOW, 10445, "joint" _PH1_ " error: multi-turn battery low voltage", "suggest...") \
00112 _D(EX_JOINT_EC_POS_CMD, 10446, "joint" _PH1_ " error: position command discontinuous", "suggest...") \
00113 _D(EX_JOINT_EC_POS_OVER_LIMIT, 10447, "joint" _PH1_ " error: position over limit", "suggest...") \
00114 _D(EX_JOINT_EC_COMM_PROTO, 10448, "joint" _PH1_ " error: communication protocol layer error", "suggest...") \
00115 _D(EX_JOINT_EC_GRAVITY_PARA_WARNING, 10449, "joint" _PH1_ " error: gravity compensation parameter invalid",
"suggest...") \
00116 _D(EX_JOINT_EC_GRAVITY_COMPENSATE_ERROR, 10450, "joint" _PH1_ " error: gravity compensation value sudden
change", "suggest...") \
00117 _D(EX_JOINT_EC_LOW_RIGIDITY, 10451, "joint" _PH1_ " error: collision soft float abnormal", "suggest...") \
00118 _D(EX_JOINT_EC_POS_CMD_WARNING, 10452, "joint" _PH1_ " error: position command unchanged during motion",
"suggest...") \
00119 _D(EX_JOINT_EC_SLAVE_COMM, 10453, "joint" _PH1_ " error: master-slave MCU communication fault", "suggest...") \
00120 _D(EX_JOINT_EC_POS_ERR, 10454, "joint" _PH1_ " error: position following error too large", "suggest...") \
00121 _D(EX_JOINT_EC_DUAL_POS_ERR, 10455, "joint" _PH1_ " error: dual servo position sync error too large",
"suggest...") \
00122 _D(EX_JOINT_EC_DUAL_COMM_ERR, 10456, "joint" _PH1_ " error: dual servo communication", "suggest...") \
00123 _D(EX_JOINT_EC_CAN_BUSOFF, 10457, "joint" _PH1_ " error: CAN bus-off warning", "suggest...") \
00124 _D(EX_JOINT_EC_SYNC_SNAKE, 10458, "joint" _PH1_ " error: sync frame jitter warning", "suggest...") \
00125 _D(EX_JOINT_EC_SYNC_DISCON, 10459, "joint" _PH1_ " error: sync frame discontinuous warning", "suggest...") \
00126 _D(EX_JOINT_EC_RPDO_LOST, 10460, "joint" _PH1_ " error: RPDO lost warning", "suggest...") \
00127 _D(EX_JOINT_EC_RPDO_MANY, 10461, "joint" _PH1_ " error: multiple RPDO in one sync cycle warning", "suggest...")
\
00128 _D(EX_JOINT_EC_GRAVITY_LOST, 10462, "joint" _PH1_ " error: gravity compensation value lost", "suggest...") \
00129 _D(EX_JOINT_EC_MAININT_TIME_WARN, 10463, "joint" _PH1_ " error: servo main interrupt runtime warning",
"suggest...") \
00130 _D(EX_JOINT_EC_MAININT_TIME_ERROR, 10464, "joint" _PH1_ " error: servo main interrupt runtime error",
"suggest...") \
00131 _D(EX_JOINT_EC_SPDINT_TIME_WARN, 10465, "joint" _PH1_ " error: servo speed loop interrupt runtime warning",
"suggest...") \
00132 _D(EX_JOINT_EC_SPDINT_TIME_ERROR, 10466, "joint" _PH1_ " error: servo speed loop interrupt runtime error",
"suggest...") \
00133 _D(EX_JOINT_EC_POS_CMD_JUMP_WARNING, 10467, "joint" _PH1_ " error: position command sudden jump during motion",
"suggest...") \
00134 _D(EX_JOINT_EC_SYNC_TIMCOARSE, 10468, "joint" _PH1_ " error: clock sync compensation status", "suggest...") \
00135 _D(EX_JOINT_EC_ENC_Z_CNTS_ERR, 10469, "joint" _PH1_ " error: incremental encoder Z signal or CNTS abnormal",
"suggest...") \
00136 _D(EX_JOINT_EC_SLAVE_OVER_CURRENT, 10470, "joint" _PH1_ " error: slave MCU detected motor phase current over
safe threshold", "suggest...") \
00137 _D(EX_JOINT_EC_SLAVE_OVER_VOLTAGE, 10471, "joint" _PH1_ " error: slave MCU detected DC bus voltage over upper
threshold", "suggest...") \
00138 _D(EX_JOINT_EC_SLAVE_UNDER_VOLTAGE, 10472, "joint" _PH1_ " error: slave MCU detected DC bus voltage below lower
threshold", "suggest...") \
00139 _D(EX_JOINT_EC_SLAVE_POS_ERR, 10473, "joint" _PH1_ " error: slave MCU detected master-slave motor position
deviation out of range", "suggest...") \

```

```

00140     _D(EX_JOINT_EC_SLAVE_SPEED_ERR, 10474, "joint" _PH1_ " error: slave MCU detected master-slave motor speed
deviation out of range", "suggest...") \
00141     _D(EX_JOINT_EC_SLAVE_TRACE_ERR, 10475, "joint" _PH1_ " error: slave MCU detected position following error out of
control range", "suggest...") \
00142     _D(EX_JOINT_EC_SLAVE_ABS_ERR, 10476, "joint" _PH1_ " error: slave MCU detected absolute encoder feedback
abnormal", "suggest...") \
00143     _D(EX_JOINT_EC_SLAVE_ADC_ZERO_OFFSET, 10477, "joint" _PH1_ " error: slave MCU detected current sampling zero
offset out of calibration range", "suggest...") \
00144     _D(EX_JOINT_EC_SLAVE_ENC_POLLUTE, 10478, "joint" _PH1_ " error: slave MCU detected encoder signal quality
degradation", "suggest...") \
00145     _D(EX_JOINT_EC_SLAVE_ENC_Z_LOST, 10479, "joint" _PH1_ " error: slave MCU detected encoder Z reference signal
lost", "suggest...") \
00146     _D(EX_JOINT_EC_SLAVE_COMM_OVER_TM, 10480, "joint" _PH1_ " error: slave MCU detected master-slave communication
timeout", "suggest...") \
00147     _D(EX_JOINT_EC_SLAVE_TRQ_ERR, 10481, "joint" _PH1_ " error: slave MCU detected master-slave motor torque
deviation out of range", "suggest...") \
00148     _D(EX_JOINT_EC_BRAKE_TYPE_ERR, 10482, "joint" _PH1_ " error: brake type config mismatch with hardware",
"suggest...") \
00149     _D(EX_JOINT_EC_PHASE_ALIGN, 10483, "joint" _PH1_ " error: phase alignment failed", "suggest...") \
00150     _D(EX_JOINT_EC_POS_OVER_LIMIT_WARNING, 10484, "joint" _PH1_ " error: position over limit", "suggest...") \
00151     _D(EX_JOINT_EC_PHASE_ALIGN_WARNING, 10485, "joint" _PH1_ " error: phase alignment warning", "suggest...") \
00152     _D(EX_JOINT_EC_TASK_STACK_SHORTAGE, 10486, "joint" _PH1_ " error: task stack insufficient", "suggest...") \
00153     _D(EX_JOINT_EC_TASK_STACK_OVERFLOW, 10487, "joint" _PH1_ " error: task stack overflow", "suggest...") \
00154     _D(EX_JOINT_EC_UNKNOWN, 10800, "joint" _PH1_ " error: unknown error", "suggest...") \
00155
00156 #define EXT_AXIS_ERRORS
\
00157     _D(EXT_AXIS_ERR_COMMON, 11001, "ext axis" _PH1_ " error: common", "Check communication with ext
axis drive.") \
00158     _D(EXT_AXIS_ERR_OVER_CURRENT, 11002, "ext axis" _PH1_ " error: over current", "Check wiring/short
circuit; reboot; if repeated replace drive/motor.") \
00159     _D(EXT_AXIS_ERR_OVER_VOLTAGE, 11003, "ext axis" _PH1_ " error: over voltage", "Check DC supply, regen,
energy eater; reboot.") \
00160     _D(EXT_AXIS_ERR_LOW_VOLTAGE, 11004, "ext axis" _PH1_ " error: low voltage", "Check DC supply and
cabling; reboot.") \
00161     _D(EXT_AXIS_ERR_OVER_TEMP, 11005, "ext axis" _PH1_ " error: over temperature", "Check environment/
cooling; reboot.") \
00162     _D(EXT_AXIS_ERR_HALL, 11006, "ext axis" _PH1_ " error: hall fault", "Check hall sensor and motor
cabling.") \
00163     _D(EXT_AXIS_ERR_ENCODER, 11007, "ext axis" _PH1_ " error: encoder fault", "Check encoder connection/
cable/noise.") \
00164     _D(EXT_AXIS_ERR_ABS_ENCODER, 11008, "ext axis" _PH1_ " error: absolute encoder fault", "Check abs
encoder power/cable; reboot.") \
00165     _D(EXT_AXIS_ERR_CUR_CALIB, 11009, "ext axis" _PH1_ " error: current calibration fault", "Reboot; check
current sensing circuit.") \
00166     _D(EXT_AXIS_ERR_Q_CURRENT, 11010, "ext axis" _PH1_ " error: current detect fault", "Reboot; check
current sensing circuit.") \
00167     _D(EXT_AXIS_ERR_ENC_POLL, 11011, "ext axis" _PH1_ " error: encoder pollution", "Check encoder
contamination/noise; improve shielding.") \
00168     _D(EXT_AXIS_ERR_ENC_Z_SIGNAL, 11012, "ext axis" _PH1_ " error: encoder Z signal fault", "Check encoder Z
channel and wiring.") \
00169     _D(EXT_AXIS_ERR_ENC_CAL, 11013, "ext axis" _PH1_ " error: encoder calibrate invalid", "Redo
calibration; check encoder.") \
00170     _D(EXT_AXIS_ERR_IMU, 11014, "ext axis" _PH1_ " error: IMU fault", "Check IMU sensor and
connection.") \
00171     _D(EXT_AXIS_ERR_TEMP_SENSOR, 11015, "ext axis" _PH1_ " error: temperature sensor fault", "Check temp
sensor wiring; reboot.") \
00172     _D(EXT_AXIS_ERR_ECABUS, 11016, "ext axis" _PH1_ " error: EtherCAT bus error", "Check EtherCAT
cabling/topology/sync; reboot master/drive.") \
00173     _D(EXT_AXIS_ERR_ECABUS_CONFIG, 11017, "ext axis" _PH1_ " error: EtherCAT config/ESI/SM/PDO fault", "Check
ESI, Mailbox/SM/PDO mapping, vendor/product/revision match.") \
00174     _D(EXT_AXIS_ERR_ECABUS_SYNC, 11018, "ext axis" _PH1_ " error: EtherCAT sync/frame/period fault", "Check
DC sync, cycle time, frame loss; verify NIC/IRQ affinity.") \
00175     _D(EXT_AXIS_ERR_SYS_CUR, 11019, "ext axis" _PH1_ " error: system current fault", "Check current loop
and load; reboot.") \
00176     _D(EXT_AXIS_ERR_SYS_POS, 11020, "ext axis" _PH1_ " error: position out of range", "Check encoder/
scale/limits; reboot.") \
00177     _D(EXT_AXIS_ERR_OVER_SPEED, 11021, "ext axis" _PH1_ " error: over speed", "Check command limits and
tuning parameters.") \
00178     _D(EXT_AXIS_ERR_OVER_ACC, 11022, "ext axis" _PH1_ " error: over acceleration", "Reduce acceleration/
jerk; check tuning.") \
00179     _D(EXT_AXIS_ERR_FOLLOW_ERROR, 11023, "ext axis" _PH1_ " error: following error", "Check gains, load,
saturation; verify feedback.") \
00180     _D(EXT_AXIS_ERR_TAG_POS_OVER, 11024, "ext axis" _PH1_ " error: target position out of range", "Check
target limits and homing.") \
00181     _D(EXT_AXIS_ERR_TAG_SPEED_OVER, 11025, "ext axis" _PH1_ " error: target speed out of range", "Clamp speed;
check profile settings.") \
00182     _D(EXT_AXIS_ERR_TAG_CURRENT_OVER, 11026, "ext axis" _PH1_ " error: target current out of range", "Clamp
current/torque; check load.") \
00183     _D(EXT_AXIS_ERR_COLLISION, 11027, "ext axis" _PH1_ " error: collision", "Remove obstruction; check
torque/force limits.") \
00184     _D(EXT_AXIS_ERR_ADC_ZERO_OFFSET, 11028, "ext axis" _PH1_ " error: ADC zero offset", "Reboot; check ADC/
current sensor offset.") \
00185     _D(EXT_AXIS_ERR_IPM_NTC, 11029, "ext axis" _PH1_ " error: IPM NTC fault", "Check power module
temperature sensing.") \
00186     _D(EXT_AXIS_ERR_SHORT_CIRCUIT, 11030, "ext axis" _PH1_ " error: short circuit", "Check motor phase wiring;
insulation test.") \

```

```

00187     _D(EXT_AXIS_ERR_MOTOR_STALL,
00188         reduce accel; reboot.") \
00189     _D(EXT_AXIS_ERR_ABS_MULTITURN,
00189         encoder battery/params; reboot.") \
00190     _D(EXT_AXIS_ERR_PHASE_LOST,
00190         connector; measure continuity.") \
00191     _D(EXT_AXIS_ERR_BRAKE,
00191         verify brake release.") \
00192     _D(EXT_AXIS_ERR_REDUCER_OVER_TEMP,
00192         temperature/cooling.") \
00193     _D(EXT_AXIS_ERR_REDUCER_NTC,
00193         temperature sensor.") \
00194     _D(EXT_AXIS_ERR_FIRMWARE_UPDATE,
00194         check power stability.") \
00195     _D(EXT_AXIS_ERR_FLASH_OP,
00195         persistent replace drive.") \
00196     _D(EXT_AXIS_ERR_EXT_ABS_ENC,
00196         external abs encoder link/power.") \
00197     _D(EXT_AXIS_ERR_DRIVE_FAULT,
00197         reboot; replace if repeated.") \
00198     _D(EXT_AXIS_ERR_OVERLOAD,
00198         and tuning.") \
00199     _D(EXT_AXIS_ERR_HARDWARE_LIMIT,
00199         limit; check limit switch.") \
00200     _D(EXT_AXIS_ERR_SERVO_MODE_TIMEOUT,
00200         transition and comm; reboot.") \
00201     _D(EXT_AXIS_ERR_UVW_ABZ,
00201         signals wiring.") \
00202     _D(EXT_AXIS_ERR_BATTERY_LOW,
00202         reboot.") \
00203     _D(EXT_AXIS_ERR_PHASE_ALIGN,
00203         check motor params.") \
00204     _D(EXT_AXIS_ERR_POS_DISCONTINUOUS,
00204         trajectory generation and limits.") \
00205     _D(EXT_AXIS_ERR_POS_INIT,
00205         encoder init/homing procedure.") \
00206     _D(EXT_AXIS_ERR_TORQUE_SENSOR,
00206         wiring/calibration.") \
00207     _D(EXT_AXIS_ERR_ABS_ENC_LOW_VOLT,
00207         supply voltage and cable.") \
00208     _D(EXT_AXIS_ERR_OFFLINE,
00208         axis id; check EtherCAT state.") \
00209     _D(EXT_AXIS_ERR_BOOTLOADER,
00209         update.") \
00210     _D(EXT_AXIS_ERR_SLAVE_OFFLINE,
00210         hardware and id.") \
00211     _D(EXT_AXIS_ERR_SLAVE_BOOTLOADER,
00211         firmware update.") \
00212     _D(EXT_AXIS_ERR_EEPROM,
00212         parameter storage; power cycle.") \
00213     _D(EXT_AXIS_ERR_PARAM_CONFIG,
00213         set; restore defaults if needed.") \
00214     _D(EXT_AXIS_ERR_STO,
00214         safety chain; reset safety.") \
00215     _D(EXT_AXIS_ERR_ENCRYPT_CHIP,
00215         chip/keys/firmware compatibility.") \
00216     _D(EXT_AXIS_ERR_BRAKE_RES_OVERLOAD,
00216         resistor/regen circuit; duty cycle.") \
00217     _D(EXT_AXIS_ERR_POWER_LINE_OPEN,
00217         cable continuity/connector.") \
00218     _D(EXT_AXIS_ERR_HOMING,
00218         procedure; retry.") \
00219     _D(EXT_AXIS_ERR_TUNING_FAIL,
00219         resonance; check mechanics.") \
00220     _D(EXT_AXIS_ERR_INERTIA_ID_FAIL,
00220         redo inertia ID; adjust conditions.") \
00221     _D(EXT_AXIS_ERR_FLYAWAY,
00221         polarity/scale; inspect drive params.") \
00222     _D(EXT_AXIS_ERR_SPEED_PULSE_OVER,
00222         feedback and scaling; reduce speed.") \
00223     _D(EXT_AXIS_ERR_CTRL_LOOP,
00223         sampling/current loop/comm timeout; reboot.")
00224 #define TOOL_ERRORS \
00225     _D(TOOL_FLASH_VERIFY_FAILED, 40001, "Flash write verify failed", "suggest...") \
00226     _D(TOOL_PROGRAM_CRC_FAILED, 40002, "Program flash checksum failed during bootloading", "suggest...") \
00227     _D(TOOL_PROGRAM_CRC_FAILED2, 40003, "Program flash checksum failed at runtime", "suggest...") \
00228     _D(TOOL_ID_UNDEFINED, 40004, "Tool ID is undefined", "suggest...") \
00229     _D(TOOL_ILLEGAL_BL_CMD, 40005, "Illegal bootloader command", "suggest...") \
00230     _D(TOOL_FW_WRONG, 40006, "Wrong firmware at the joint", "suggest...") \
00231     _D(TOOL_HW_INVALID, 40007, "Invalid hardware revision", "suggest...") \
00232     _D(TOOL_SHORT_CURCUI_T_H, 40011, "Short circuit detected on Digital Output: " _PH1_ " high side", "suggest...") \
00233     _D(TOOL_SHORT_CURCUI_T_L, 40012, "Short circuit detected on Digital Output: " _PH1_ " low side", "suggest...") \
00234     _D(TOOL_AVERAGE_CURR_HIGH, 40013, "10 second Average tool IO Current of " _PH1_ " A is outside of the allowed range.", "suggest...") \
00235     _D(TOOL_POWER_PIN_OVER_CURR, 40014, "Current of " _PH1_ " A on the POWER pin is outside of the allowed range.", "suggest...") \

```



```

00236     _D(TOOL_DOUT_PIN_OVER_CURR, 40015, "Current of " _PH1_ " A on the Digital Output pins is outside of the allowed
range.", "suggest...") \
00237     _D(TOOL_GROUND_PIN_OVER_CURR, 40016, "Current of " _PH1_ " A on the ground pin is outside of the allowed
range.", "suggest...") \
00238     _D(TOOL_RX_FRAMING, 40021, "RX framing error", "suggest...") \
00239     _D(TOOL_RX_PARITY, 40022, "RX Parity error", "suggest...") \
00240     _D(TOOL_48V_LOW, 40031, "48V input is too low", "suggest...") \
00241     _D(TOOL_48V_HIGH, 40032, "48V input is too high", "suggest...") \
00242     _D(TOOL_ERR_OFFLINE, 40033, "tool error: tool may be offline", "(a) Check tool's hardware. (b) Check joint's
id.") \
00243     _D(TOOL_ERR_BOOTLOADER, 40034, "tool error: The tool is in bootloader mode. Retry firmware update. ",
"suggest...")
00244
00245 #define EX_TOOL_ERRORS \
00246     _D(EX_TOOL_EC_LOW_VOLTAGE, 40101, "tool error: low voltage", "check tool power supply voltage") \
00247     _D(EX_TOOL_EC_FORCESENSOR_COMM, 40102, "tool error: external force sensor communication error", "check external
force sensor communication connection") \
00248     _D(EX_TOOL_485_SENDFULL_COMM, 40103, "tool error: 485 transparent transmission buffer full", "check 485
communication load and transmission frequency") \
00249     _D(EX_TOOL_FORCESENSOR_FILTER_ZERO, 40104, "tool error: force sensor filter parameter is zero", "check force
sensor filter parameter configuration") \
00250     _D(EX_TOOL_EC_UNKNOWN, 40200, "tool error: unknown error", "check controller logs and hardware status")
00251
00252 #define PEDSTRAL_ERRORS \
00253     _D(PKG_LOST, 50001, "Lost package from pedestal", "suggest...") \
00254     _D(PEDSTRAL_OFFLINE, 50002, "pedestal error: pedestal may be offline", "(a) Check pedestal's hardware. (b)
Check pedestal's id.") \
00255     _D(PEDSTRAL_ERR_BOOTLOADER, 50003, "pedestal error: The pedestal is in bootloader mode. Retry firmware update.
", "suggest...")
00256
00257 #define EX_PEDSTRAL_ERRORS \
00258     _D(EX_BASE_EC_LOW_VOLTAGE, 50101, "pedstral error: low voltage", "check power supply voltage and battery
status") \
00259     _D(EX_BASE_EC_OVER_TEMPERATURE_RES, 50102, "pedstral error: resistor over temperature", "check braking resistor
temperature and cooling condition") \
00260     _D(EX_BASE_EC_OVER_TARGET_BRAKE_OPEN_VOLT, 50103, "pedstral error: input voltage close to or exceeds
regenerative brake activation voltage", "check input power voltage and regenerative braking configuration") \
00261     _D(EX_BASE_EC_RES_BREAKAGE, 50104, "pedstral error: brake resistor breakage", "check whether the brake resistor
is disconnected or damaged") \
00262     _D(EX_BASE_EC_IMU_CALIBRATE, 50105, "pedstral error: IMU calibration required", "perform IMU calibration
according to maintenance procedure") \
00263     _D(EX_BASE_EC_TEMP_SENSOR_SHORT, 50106, "pedstral error: temperature sensor short circuit", "check temperature
sensor wiring and solder joints for short circuit") \
00264     _D(EX_BASE_EC_TEMP_SENSOR_BREAK, 50107, "pedstral error: temperature sensor open circuit", "check temperature
sensor connection and cable continuity") \
00265     _D(EX_BASE_EC_96V_INSTANT_OVER_VOLTAGE, 50108, "pedstral error: 96V instantaneous over voltage after
regenerative braking", "check regenerative braking behavior and power bus voltage") \
00266     _D(EX_BASE_EC_UNKNOWN, 50200, "pedstral error: unknown error", "check controller logs and hardware status")
00267
00268 #define SAFETY_INTERFACE_BOARD_ERRORS \
00269     _D(IFB_ERR_ROBOTTYPE, 20001, "Robot error type!", "suggest...") \
00270     _D(IFB_ERR_ADXL_SENS, 20002, "Base Acceleration sensor error!", "suggest...") \
00271     _D(IFB_ERR_EN_LINE, 20003, "Encoder line error!", "suggest...") \
00272     _D(IFB_ERR_ENTER_HDG_MODE, 20004, "Robot enter handguide mode!", "suggest...") \
00273     _D(IFB_ERR_EXIT_HDG_MODE, 20005, "Robot exit handguide mode!", "suggest...") \
00274     _D(IFB_ERR_MAC_DATA_BREAK, 20006, "MAC data break!", "suggest...") \
00275     _D(IFB_ERR_DRV_FIRMWARE_VERSION, 20007, "Motor driver firmware version error!", "suggest...") \
00276     _D(INIT_ERR_EN_DRV, 20008, "Motor driver enable failed!", "suggest...") \
00277     _D(INIT_ERR_EN_AUTO_BACK, 20009, "Motor driver enable auto back failed!", "suggest...") \
00278     _D(INIT_ERR_EN_CUR_LOOP, 20010, "Motor driver enable current loop failed!", "suggest...") \
00279     _D(INIT_ERR_SET_TAG_CUR, 20011, "Motor driver set target current failed!", "suggest...") \
00280     _D(INIT_ERR_RELEASE_BRAKE, 20012, "Motor driver release brake failed!", "suggest...") \
00281     _D(INIT_ERR_EN_POS_LOOP, 20013, "Motor driver enable position loop failed!", "suggest...") \
00282     _D(INIT_ERR_SET_MAX_ACC, 20014, "Motor set max accelerate failed!", "suggest...") \
00283     _D(SAFETY_ERR_PROTECTION_STOP_TIMEOUT, 20015, "Protective stop timeout!", "suggest...") \
00284     _D(SAFETY_ERR_REDUCED_MODE_TIMEOUT, 20016, "Reduced mode timeout!", "suggest...") \
00285     _D(SYS_ERR_MCU_COM, 20017, "Robot system error: mcu communication error!", "suggest...") \
00286     _D(SYS_ERR_RS485_COM, 20018, "Robot system error: RS485 communication error!", "suggest...") \
00287     _D(IFB_ERR_DISCONNECTED, 20019, "Interface board may be disconnected. Please check connection between IPC and
Interface board.", "suggest...") \
00288     _D(IFB_ERR_PAYLOAD_ERROR, 20020, "Payload error.", "suggest...") \
00289     _D(IFB_OFFLINE, 20021, "ifaceboard error: ifaceboard may be offline", "(a) Check ifaceboard's hardware. (b)
Check ifaceboard's id.") \
00290     _D(IFB_ERR_BOOTLOADER, 20022, "ifaceboard error: The ifaceboard is in bootloader mode. Retry firmware update.
", "suggest...") \
00291     _D(IFB_SLAVE_OFFLINE, 20023, "interface slave board error: interface slave board may be offline", "(a) Check
interface slave board's hardware. (b) Check interface slave board's id.") \
00292     _D(IFB_SLAVE_ERR_BOOTLOADER, 20024, "interface slave board error: The interface slave board is in bootloader
mode. Retry firmware update. ", "suggest...") \
00293     _D(IFB_TOOL_ERR_ADXL_SENS, 20025, "Tool Acceleration sensor error!", "suggest...") \
00294     _D(HANDLE_OFFLINE, 20026, "handle error: handle may be offline", "(a) Check handle's hardware. (b) Check
handle's id.") \
00295     _D(HANDLE_ERR_BOOTLOADER, 20027, "handle error: The handle is in bootloader mode. Retry firmware update. ",
"suggest...") \
00296     _D(IFB_POWERLOSS_OFFLINE, 20028, "interface powerloss board error: interface powerloss board may be offline",
"(a) Check interface powerloss board's hardware. (b) Check interface powerloss board's id.") \
00297     _D(IFB_POWERLOSS_ERR_BOOTLOADER, 20029, "interface powerloss board error: The interface powerloss board is in

```

```

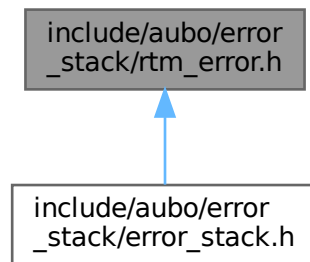
bootloader mode. Retry firmware update. ", "suggest...") \
00298     _D(HANDLE_COMM_ERROR, 20030, "handle error: handle comm error", "(a) Check handle's hardware. (b) Check
handle's id.")
00299
00300 #define HARDWARE_INTERFACE_ERRORS \
00301     _D(HW_SCB_SETUP_FAILED, 60001, "Setup of Interface Board failed", "suggest...") \
00302     _D(HW_PKG_CNT_DISAGREE, 60002, "Packet counter disagreements", "suggest...") \
00303     _D(HW_SCB_DISCONNECT, 60003, "Connection to Interface Board lost", "suggest...") \
00304     _D(HW_SCB_PKG_LOST, 60004, "Package lost from Interface Board", "suggest...") \
00305     _D(HW_SCB_CONN_INIT_FAILED, 60005, "Ethernet connection initialization with Interface Board failed",
"suggest...") \
00306     _D(HW_LOST_JOINT_PKG, 60006, "Lost package from joint " _PH1_ "", "suggest...") \
00307     _D(HW_LOST_TOOL_PKG, 60007, "Lost package from tool", "suggest...") \
00308     _D(HW_JOINT_PKG_CNT_DISAGREE, 60008, "Packet counter disagreement in packet from joint " _PH1_ "", "suggest...")
\
00309     _D(HW_TOOL_PKG_CNT_DISAGREE, 60009, "Packet counter disagreement in packet from tool", "suggest...") \
00310     _D(HW_JOINTS_FAULT, 60011, "" _PH1_ " joint entered the Fault State", "suggest...") \
00311     _D(HW_JOINTS_VIOLATION, 60012, "" _PH1_ " joint entered the Violation State", "suggest...") \
00312     _D(HW_TP_FAULT, 60013, "Teach Pendant entered the Fault State", "suggest...") \
00313     _D(HW_TP_VIOLATION, 60014, "Teach Pendant entered the Violation State", "suggest...") \
00314     _D(HW_JOINT_MV_TOO_FAR, 60021, "" _PH1_ " joint moved too far before robot entered RUNNING State", "suggest...")
\
00315     _D(HW_JOINT_STOP_NOT_FAST, 60022, "Joint Not stopping fast enough", "suggest...") \
00316     _D(HW_JOINT_MV_LIMIT, 60023, "Joint moved more than allowable limit", "suggest...") \
00317     _D(HW_FT_SENSOR_DATA_INVALID, 60024, "Force-Torque Sensor data invalid", "suggest...") \
00318     _D(HW_NO_FT_SENSOR, 60025, "Force-Torque sensor is expected, but it cannot be detected", "suggest...") \
00319     _D(HW_FT_SENSOR_NOT_CALIB, 60026, "Force-Torque sensor is detected but not calibrated", "suggest...") \
00320     _D(HW_RELEASE_BRAKE_FAILED, 60030, "Robot was not able to brake release, see log for details", "suggest...") \
00321     _D(HW_OVERCURR_SHUTDOWN, 60040, "Overcurrent shutdown", "suggest...") \
00322     _D(HW_ENERGY_SURPLUS, 60050, "Energy surplus shutdown", "suggest...") \
00323     _D(HW_IDLE_POWER_HIGH, 60060, "Idle power consumption to high", "suggest...") \
00324     _D(HW_ENTER_COLLISION_TIMEOUT, 60071, "Enter collision stop procedure timeout", "suggest...") \
00325     _D(HW_POWERON_TIMEOUT, 60072, "Poweron robot timeout", "suggest...") \
00326     _D(HW_NO_NIC_FOUND, 60073, "No network cards found.", "suggest...") \
00327     _D(HW_IFB_NOT_FOUND, 60074, "No Interface Board found.", "suggest...") \
00328     _D(HW_IFB_BOOTLOAD, 60075, "The Interface Board is in bootloader mode. Update firmware firstly.", "suggest...")
\
00329     _D(HW_TOOL_NOT_FOUND, 60076, "No Tool Board found.", "suggest...") \
00330     _D(HW_BASE_NOT_FOUND, 60077, "No Base Board found.", "suggest...") \
00331     _D(HW_BRINGUP_TIMEOUT, 60078, "Poweron robot timeout", "suggest...") \
00332     _D(HW_COLLISION_RECOVERY_FAILED, 60079, "Collision recovery failed", "suggest...") \
00333     _D(HW_TP_ENABLED, 60080, "Teach pendant enabled status changed to " _PH1_ "", "suggest...")
00334
00335 // clang-format on
00336
00337 // 定义硬件抽象层的错误代码
00338 #define HAL_ERRORS \
00339     JOINT_ERRORS \
00340     JOINT_EX_ERRORS \
00341     EXT_AXIS_ERRORS \
00342     SAFETY_INTERFACE_BOARD_ERRORS \
00343     TOOL_ERRORS \
00344     EX_TOOL_ERRORS \
00345     PEDSTRAL_ERRORS \
00346     EX_PEDSTRAL_ERRORS \
00347     HARDWARE_INTERFACE_ERRORS
00348
00349 #endif // AUBO_SDK_JOINT_ERROR_H

```

12.10 include/aubo/error_stack/rtm_error.h 文件参考

运行时错误码

此图展示该文件被哪些文件直接或间接地引用了:



宏定义

- `#define RTM_ERRORS`

12.10.1 详细描述

运行时错误码

在文件 `rtm_error.h` 中定义.

12.10.2 宏定义说明

RTM_ERRORS

`#define RTM_ERRORS`

在文件 `error_stack.h` 第 10 行定义.

12.11 rtm_error.h

浏览该文件的文档.

```

00001 /** @file rtm_error.h
00002  * @brief 运行时错误码
00003  */
00004 #ifndef AUBO_SDK_RTM_ERROR_H
00005 #define AUBO_SDK_RTM_ERROR_H
00006
00007 // clang-format off
00008
00009 #define RTM_ERRORS \
00010     _D(ROBOT_BE_PULLING, 30001, "Something is pulling the robot.", "Please check TCP configuration,payload and
00011     mounting settings") \
00012     _D(PSTOP_ELLOW_POS, 30002, "Protective Stop: Elbow position close to safety plane limits.", "Please move robot
00013     Elbow joint away from the safety plane") \
00014     _D(PSTOP_STOP_TIME, 30003, "Protective Stop: Exceeding user safety settings for stopping time.", "(a) Check
00015     speeds and accelerations in the program (b) Check usage of TCP,payload and CoG correctly (c) Check external
00016     equipmentactivation if correctly set") \
00017     _D(PSTOP_STOP_DISTANCE, 30004, "Protective Stop: Exceeding user safety settings for stopping distance.", "(a)
00018     Check speeds and accelerations in the program (b) Check usage of TCP,payload and CoG correctly (c) Check external
00019     equipmentactivation if correctly set") \
00020     _D(PSTOP_CLAMP, 30005, "Protective Stop: Danger of clamping between the Robot' s lower arm and tool.", "(a) Check
00021     speeds and accelerations in the program (b) Check usage of TCP,payload and CoG correctly (c) Check external
00022     equipmentactivation if correctly set") \
00023     _D(PSTOP_POS_LIMIT, 30006, "Protective Stop: Position close to joint limits", "suggest...") \
00024     _D(PSTOP_ORI_LIMIT, 30007, "Protective Stop: Tool orientation close to limits", "suggest...") \
00025     _D(PSTOP_PLANE_LIMIT, 30008, "Protective Stop: Position close to safety plane limits", "suggest...") \
00026     _D(PSTOP_POS_DEVIATE, 30009, "Protective Stop: Position deviates from path", "Check payload, center of gravity
00027     and acceleration settings.") \
00028     _D(JOINT_CHK_PAYLOAD, 30010, "Joint " _PH1_ ": Check payload, center of gravity and acceleration settings. Log
00029     screen may contain additional information.", "suggest...") \
00030     _D(PSTOP_SINGULARITY, 30011, "Protective Stop: Position in singularity.", "Please use MoveJ or change the
00031     motion") \
  
```

```

00021 _D(PSTOP_CANNOT_MAINTAIN, 30012, "Protective Stop: Robot cannot maintain its position, check if payload is
correct", "suggest...") \
00022 _D(PSTOP_WRONG_PAYLOAD, 30013, "Protective Stop: Wrong payload or mounting detected, or something is pushing the
robot when entering Freedrive mode","Verify that the TCP configuration and mounting in the used installation is
correct") \
00023 _D(PSTOP_JOINT_COLLISION, 30014, "Protective Stop: Collision detected by joint " _PH1_, "Make sure no objects
are in the path of the robot and resume the program") \
00024 _D(PSTOP_POS_DISAGREE, 30015, "Protective stop: The robot was powered off last time due to a joint position
disagreement.", " (a) Verify that the robot position in the 3D graphics matches the real robot, to ensure that the
encoders function before releasing the brakes. Stand back and monitor the robot performing its first program cycle
as expected. (b) If the position is not correct, the robot must be repaired. In this case, click Power Off Robot.
(c) If the position is correct, please tick the check box below the 3D graphics and click Robot Position Verified")
\
00025 _D(TARGET_JOINT_SPEED_EXCEED, 30016, "Target joint speed exceed limits", "suggest...") \
00026 _D(TARGET_POS_SUDDEN_CHG, 30017, "Sudden change in target position", "suggest...") \
00027 _D(SUDDEN_STOP, 30018, "Sudden stop.", " To abort a motion, use \"stopj\" or \"stopl\" script commands to
generate a smooth deceleration before using \"wait\". Avoid aborting motions between waypoints with blend" ) \
00028 _D(ROBOT_STOP_ABNORMAL, 30019, "Robot has not stopped in the allowed reaction and braking time", "suggest...") \
00029 _D(PROG_INVALID_SETP, 30020, "Robot program resulted in invalid setpoint.", "Please review waypoints in the
program") \
00030 _D(BLEND_INVALID_SETP, 30021, "Blending failed and resulted in an invalid setpoint.", "Try changing the blend
radius or contact technical support") \
00031 _D(APPROACH_SINGULARITY, 30022, "Robot approaching singularity - Acceleration threshold failed.", "Review
waypoints in the program, try using MoveJ instead of MoveL in the position close to singularity") \
00032 _D(TSPEED_UNMATCH_POS, 30023, "Target speed does not match target position", "suggest...") \
00033 _D(INCONSIS_TPOS_SPD, 30024, "Inconsistency between target position and speed", "suggest...") \
00034 _D(JOINT_TSPD_UNMATCH_POS, 30025, "Target joint speed does not match target joint position change - Joint "
_PH1_ " ", "suggest...") \
00035 _D(FIELDBUS_INPUT_DISCONN, 30026, "Fieldbus input disconnected.", "Please check fieldbus connections (RTDE,
ModBus, EtherNet/IP and Profinet) or disable the fieldbus in the installation. Check RTDE watchdog feature. Check if
a URCap is using this feature.") \
00036 _D(OPMODE_CHANGED, 30027, "Operational mode changed: " _PH1_ " ", "suggest...") \
00037 _D(NO_KIN_CALIB, 30028, "No Kinematic Calibration found (calibration.conf file is either corrupt or
missing).", "A new kinematics calibration may be needed if the robot needs to improve its kinematics, otherwise,
ignore this message") \
00038 _D(KIN_CALIB_UNMATCH_JOINT, 30029, "Kinematic Calibration for the robot does not match the joint(s).", "If
moving a program from a different robot to this one, rekinematic calibrate the second robot to improve kinematics,
otherwise ignore this message.") \
00039 _D(KIN_CALIB_UNMATCH_ROBOT, 30030, "Kinematic Calibration does not match the robot.", "Please check if the serial
number of the robot arm matches the Control Box") \
00040 _D(JOINT_OFFSET_CHANGED, 30031, "Large movement of the robot detected while it was powered off. The joints were
moved while it was powered off, or the encoders do not function", "suggest...") \
00041 _D(OFFSET_CHANGE_HIGH, 30032, "Change in offset is too high", "suggest...") \
00042 _D(JOINT_SPEED_LIMIT, 30033, "Close to joint speed safety limit.", "Review program speed and acceleration") \
00043 _D(TOOL_SPEED_LIMIT, 30034, "Close to tool speed safety limit.", "Review program speed and acceleration") \
00044 _D(MOMENTUM_LIMIT, 30035, "Close to momentum safety limit.", "Review program speed and acceleration") \
00045 _D(ROBOT_MV_STOP, 30036, "Robot is moving when in Stop Mode", "suggest...") \
00046 _D(HAND_PROTECTION, 30037, "Hand protection: Tool is too close to the lower arm: " _PH1_ " meter.", "(a) Check
wrist position. (b) Verify mounting (c) Do a Complete rebooting sequence (d) Update software (e) Contact your local
AUBO Robots service provider for assistance") \
00047 _D(WRONG_SAFETYMODE, 30038, "Wrong safety mode: " _PH1_, "suggest...") \
00048 _D(SAFETYMODE_CHANGED, 30039, "Safety mode changed: " _PH1_, "suggest...") \
00049 _D(JOINT_ACC_LIMIT, 30040, "Close to joint acceleration safety limit", "suggest...") \
00050 _D(TOOL_ACC_LIMIT, 30041, "Close to tool acceleration safety limit", "suggest...") \
00051 _D(JOINT_TEMPERATURE_LIMIT, 30042, "Joint " _PH1_ " temperature too high(>" _PH2_ "°C)", "suggest...") \
00052 _D(CONTROL_BOX_TEMPERATURE_LIMIT, 30043, "Control box temperature too high(>" _PH1_ "°C)", "suggest...") \
00053 _D(ROBOT_EMERGENCY_STOP, 30044, "Robot emergency stop", "suggest...") \
00054 _D(ROBOTMODE_CHANGED, 30045, "Robot mode changed: " _PH1_, "suggest...") \
00055 _D(ROBOTMODE_ERROR, 30046, "Wrong robot mode: " _PH1_, "suggest...") \
00056 _D(POSE_OUT_OF_REACH, 30047, "Target pose [ " _PH1_ " ] out of reach", "suggest...") \
00057 _D(TP_PLAN_FAILED, 30048, "Trajectory plan FAILED." , "suggest...") \
00058 _D(START_FORCE_FAILED, 30049, "Start force control failed, because force sensor does not exist." , "suggest...")
\
00059 _D(OVER_SAFE_PLANE_LIMIT, 30050, _PH1_ " axis exceeds the safety plane limit (Move_type:" _PH2_ " id:" _PH3_
").", "Please move the robot to the safety plane range.") \
00060 _D(POWERON_FAIL_VIOLATION, 30051, "Failed to power on because the robot safety mode is in violation",
"suggest...") \
00061 _D(POWERON_FAIL_SYSTEMEMERGENCYSTOP, 30052, "Failed to power on because the robot safety mode is in system
emergency stop", "suggest...") \
00062 _D(POWERON_FAIL_ROBOTEMERGENCYSTOP, 30053, "Failed to power on because the robot safety mode is in robot
emergency stop", "Pop up the red emergency stop button on the teach pendant when the robot is in a safe range of
motion") \
00063 _D(POWERON_FAIL_FAULT, 30054, "Failed to power on because the robot safety mode is in fault", "suggest...") \
00064 _D(STARTUP_FAIL_VIOLATION, 30055, "Failed to startup because the robot safety mode is in violation",
"suggest...") \
00065 _D(STARTUP_FAIL_SYSTEMEMERGENCYSTOP, 30056, "Failed to startup because the robot safety mode is in system
emergency stop", "suggest...") \
00066 _D(STARTUP_FAIL_ROBOTEMERGENCYSTOP, 30057, "Failed to startup because the robot safety mode is in robot
emergency stop", "Pop up the red emergency stop button on the teach pendant when the robot is in a safe range of
motion") \
00067 _D(STARTUP_FAIL_FAULT, 30058, "Failed to startup because the robot safety mode is in fault", "suggest...") \
00068 _D(BACKDRIVE_FAIL_VIOLATION, 30059, "Failed to backdrive because the robot safety mode is in violation",
"suggest...") \
00069 _D(BACKDRIVE_FAIL_SYSTEMEMERGENCYSTOP, 30060, "Failed to backdrive because the robot safety mode is in system
emergency stop", "suggest...") \
00070 _D(BACKDRIVE_FAIL_ROBOTEMERGENCYSTOP, 30061, "Failed to backdrive because the robot safety mode is in robot
emergency stop", "Pop up the red emergency stop button on the teach pendant when the robot is in a safe range of

```



```

motion") \
00071     _D(BACKDRIVE_FAIL_FAULT, 30062, "Failed to backdrive because the robot safety mode is in fault", "suggest...") \
00072     _D(SETSIM_FAIL_VIOLATION, 30063, "Switch sim mode failed because the robot safety mode is in violation",
"suggest...") \
00073     _D(SETSIM_FAIL_SYSTEMEMERGENCYSTOP, 30064, "Switch sim mode failed because the robot safety mode is in system
emergency stop", "suggest...") \
00074     _D(SETSIM_FAIL_ROBOTEMERGENCYSTOP, 30065, "Switch sim mode failed because the robot safety mode is in robot
emergency stop", "Pop up the red emergency stop button on the teach pendant when the robot is in a safe range of
motion") \
00075     _D(SETSIM_FAIL_FAULT, 30066, "Switch sim mode failed because the robot safety mode is in fault", "suggest...") \
00076     _D(FREEDRIVE_FAIL_VIOLATION, 30067, "Enable handguide mode failed because the robot safety mode is in
violation", "suggest...") \
00077     _D(FREEDRIVE_FAIL_SYSTEMEMERGENCYSTOP, 30068, "Enable handguide mode failed because the robot safety mode is in
system emergency stop", "suggest...") \
00078     _D(FREEDRIVE_FAIL_ROBOTEMERGENCYSTOP, 30069, "Enable handguide mode failed because the robot safety mode is in
robot emergency stop", "Pop up the red emergency stop button on the teach pendant when the robot is in a safe range
of motion") \
00079     _D(FREEDRIVE_FAIL_FAULT, 30070, "Enable handguide mode failed because the robot safety mode is in fault",
"suggest...") \
00080     _D(UPFIRMWARE_FAIL_VIOLATION, 30071, "Firmware update failed because the robot safety mode is in violation",
"suggest...") \
00081     _D(UPFIRMWARE_FAIL_SYSTEMEMERGENCYSTOP, 30072, "Firmware update failed because the robot safety mode is in
system emergency stop", "suggest...") \
00082     _D(UPFIRMWARE_FAIL_ROBOTEMERGENCYSTOP, 30073, "Firmware update failed because the robot safety mode is in robot
emergency stop", "Pop up the red emergency stop button on the teach pendant when the robot is in a safe range of
motion") \
00083     _D(UPFIRMWARE_FAIL_FAULT, 30074, "Firmware update failed because the robot safety mode is in fault",
"suggest...") \
00084     _D(SETPERSOSTENT_FAIL_VIOLATION, 30075, "Set persistent parameter failed because the robot safety mode is in
violation", "suggest...") \
00085     _D(SETPERSOSTENT_FAIL_SYSTEMEMERGENCYSTOP, 30076, "Set persistent parameter failed because the robot safety mode
is in system emergency stop", "suggest...") \
00086     _D(SETPERSOSTENT_FAIL_ROBOTEMERGENCYSTOP, 30077, "Set persistent parameter failed because the robot safety mode
is in robot emergency stop", "Pop up the red emergency stop button on the teach pendant when the robot is in a safe
range of motion") \
00087     _D(SETPERSOSTENT_FAIL_FAULT, 30078, "Set persistent parameter failed because the robot safety mode is in fault",
"suggest...") \
00088     _D(SETPERSOSTENT_FAIL_PARAM_ERR, 30079, "Set persistent parameter failed", "(a) Check the parameter format,
whether all are floating point numbers") \
00089     _D(ROBOT_CABLE_DISCONN, 30080, "Robot cable not connected", "(a) Make sure the cable between Control Box and
Robot Arm is correctly connected and it has no damage. (b) Check for loose connections (c) Do a Complete rebooting
sequence (d) Update software (e) Contact your local AUBO Robots service provider for assistance. Contact your local
AUBO Robots service provider for assistance.") \
00090     _D(TP_TOO_SHORT, 30081, "The generated trajectory is ignored because it is too short", "(a) Please check if the
added waypoints are coincident (b) If it is an arc movement, please check whether the three points are collinear") \
00091     _D(INV_KIN_FAIL, 30082, "Inverse kinematics solution failed. The target pose may be in a singular position or
exceed the joint limits", "(a) Change the target pose and try moving again") \
00092     _D(FREEDRIVE_ENABLED, 30083, "Freedrive status changed to " _PH1_ "", "suggest...") \
00093     _D(TP_INV_FAIL_REFERENCE_JOINT_OUT_OF_LIMIT, 30084, "Inverse kinematics solution failed. Reference angle ["
_PH1_ "] exceeds joint limit [" _PH2_ "].", "suggest...") \
00094     _D(TP_INV_FAIL_NO_SOLUTION, 30085, "Inverse kinematics solution failed. The reference angle [" _PH1_ "] and the
target angle [" _PH2_ "] are used as parameters. there is no solution in the calculation of the inverse solution
process.", "suggest...") \
00095     _D(SERVO_FAIL_VIOLATION, 30086, "Switch servo mode failed because the robot safety mode is in violation",
"suggest...") \
00096     _D(SERVO_FAIL_SYSTEMEMERGENCYSTOP, 30087, "Switch servo mode failed because the robot safety mode is in system
emergency stop", "suggest...") \
00097     _D(SERVO_FAIL_ROBOTEMERGENCYSTOP, 30088, "Switch servo mode failed because the robot safety mode is in robot
emergency stop", "Pop up the red emergency stop button on the teach pendant when the robot is in a safe range of
motion") \
00098     _D(SERVO_FAIL_FAULT, 30089, "Switch servo mode failed because the robot safety mode is in fault", "suggest...")
\
00099     _D(FREEDRIVE_FAIL_NO_RUNNING, 30090, "Enable handguide mode failed because the robot mode type is " _PH1_ "(not
running)", "suggest...") \
00100     _D(RUNTIME_MACHINE_ERROR, 30091, "The state of the running machine is " _PH1_ ", not " _PH2_ ". " _PH3_ "
function execution failed because the state is wrong.", "suggest...") \
00101     _D(RESUME_FAR_PAUSE_PT, 30092, "Cannot resume from joint position [" _PH1_ "].\nToo far away from paused point
[" _PH2_ "].", "suggest...") \
00102     _D(PAYLOAD_LIGHTER_ERROR, 30093, "The payload setting is too small!", "suggest...") \
00103     _D(PAYLOAD_OVERLOAD_ERROR, 30094, "The payload setting is too large!", "suggest...") \
00104     _D(PAUSE_FAIL_NOT_POSITION_PLAN_MODE, 30095, "This motion does not support the pause function. The motion is
stopping.", "suggest...") \
00105     _D(TP_PLAN_FAILED_CIRCULAR_WAYPOINTS_COINCIDE, 30096, "The planning failed because the three waypoints of the
arc were determined to coincide.", "Check the circular waypoints to make sure they are different.") \
00106     _D(SERVO_WRONG_SAFETYMODE, 30097, "Switch servo mode failed because the robot safety mode is in " _PH1_ ".",
"Check the circular waypoints to make sure they are different.") \
00107     _D(SET_PERSTPARAM_WRONG_SAFETYMODE, 30098, "Set persistent parameter failed because the robot safety mode is in
" _PH1_ ", "suggest...") \
00108     _D(SET_KINPARAM_WRONG_SAFETYMODE, 30099, "Set Kinematics Compensate parameters failed because the robot safety
mode is in " _PH1_ ", "suggest...") \
00109     _D(SET_ROBOT_ZERO_WRONG_SAFETYMODE, 30100, "Set current joint angles to zero failed because the robot safety
mode is in " _PH1_ ", "suggest...") \
00110     _D(UPFIRMWARE_WRONG_SAFETYMODE, 30101, "Firmware update failed because the robot safety mode is in " _PH1_ ,
"suggest...") \
00111     _D(POWERON_WRONG_SAFETYMODE, 30102, "Failed to power on because the robot safety mode is in " _PH1_ ,
"suggest...") \
00112     _D(STARTUP_WRONG_SAFETYMODE, 30103, "Failed to startup because the robot safety mode is in " _PH1_ ,

```

```

"suggest...") \
00113 _D(BACKDRIVE_WRONG_SAFETYMODE, 30104, "Failed to backdrive because the robot safety mode is in system emergency
stop", "suggest...") \
00114 _D(SETSIM_WRONG_SAFETYMODE, 30105, "Switch sim mode failed because the robot safety mode is in violation",
"suggest...") \
00115 _D(FREEDRIVE_WRONG_SAFETYMODE, 30106, "Enable handguide mode failed because the robot safety mode is in wrong
safety mode: " _PH1_, "suggest...") \
00116 _D(TP_PLAN_FAILED_JOINT_JUMP_BIGGER, 30107, "Inverse kinematics solution failed. The target point and the
current point are in different robot configuration spaces.", "Add a few more points between the target point and the
current point.") \
00117 _D(RUN_PROGRAM_FAILED, 30108, "Run program " _PH1_ " failed.", "suggset...") \
00118 _D(FREEDRIVE_FAIL_WRONG_RTMSTATE, 30109, "Unable to enter the HandGuide mode as the robot is not currently in a
stopped or paused state.", "suggset...") \
00119 _D(SAFEGUARDSTOP_CONFIGURABLE_INPUT, 30110, "Configurable safety input is triggered.", "suggset...") \
00120 _D(SAFEGUARDSTOP_3PE, 30111, "3PE is triggered.", "suggset...") \
00121 _D(SAFEGUARDSTOP_SI, 30112, "SIO/SI1 is triggered.", "suggset...") \
00122 _D(ROBOT_TYPE_CHANGED, 30200, "Robot type changed to " _PH1_ "", and robot subtype changed to " _PH2_ "",
"suggest...") \
00123 _D(LINKMODE_CHANGED, 30201, "Link mode changed to " _PH1_ "", "suggest...") \
00124 _D(ROBOT_SELF_COLLISION, 30301, "Detect risk of robot self collision", "suggest...") \
00125 _D(CONSTANT_INVALID, 30302, "Joint torque constants are invalid. HandGuide will be disabled, and the collision
protection may be triggered by mistake.", "suggest...") \
00126 _D(GRAVITY_INVALID, 30303, "Abnormal value of gravity acceleration sensor. HandGuide will be disabled, and the
collision protection may be triggered by mistake.", "suggest...") \
00127 _D(DYNAMICS_INVALID, 30304, "Robot dynamics parameters are invalid. HandGuide will be disabled, and the
collision protection may be triggered by mistake.", "suggest...") \
00128 _D(FRICTION_INVALID, 30305, "Joint friction parameters are invalid. HandGuide will be disabled, and the
collision protection may be triggered by mistake.", "suggest...") \
00129 _D(HANDGUIDE_UNDER_DEVELOP, 30306, "Robot type of " _PH1_ " function under development. HandGuide will be
disabled, and the collision protection may be triggered by mistake.", "suggest...") \
00130 _D(SLOW_DOWN_INFO, 30307, "Slow down level changed to " _PH1_ "(" _PH2_ "%)", "suggest...") \
00131 _D(WRONG_JOINT_DESIGNED_LIMIT, 30308, "Joint designed ranges exceeds ranges read from hardware interface.",
"suggest...") \
00132 _D(FREEDRIVE_IN_SIMULATION, 30309, "Enable handguide mode failed because the robot is in simulation mode.",
"suggest...") \
00133 _D(ROBOT_STOPPING_TIMEOUT, 30310, "Robot stopping timeout.", "suggest...") \
00134 _D(PSTOP_INCORRECT_FORCE_OFFSET, 30311, "Protective Stop: Sudden change in force control target position. Force
sensor offset may be incorrect or force sensor fault.", "suggest...") \
00135 _D(WRONG_JOINT_SAFETY_LIMIT, 30312, "Joint safety ranges exceeds designed ranges.", "suggest...") \
00136 _D(PSTOP_TCP_PLANE_VIOLATION, 30401, "Protective Stop: TCP position close to safety plane limits.",
"suggest...") \
00137 _D(PSTOP_ELBOW_PLANE_VIOLATION, 30402, "Protective Stop: elbow position close to safety plane limits.",
"suggest...") \
00138 _D(PSTOP_JOINT_TORQUE_VIOLATION, 30403, "Protective Stop: joint" _PH1_ " exceeds torque limit.", "suggest...") \
00139 _D(PSTOP_JOINT_POSITION_VIOLATION, 30404, "Protective Stop: joint" _PH1_ " exceeds position limit.",
"suggest...") \
00140 _D(PSTOP_JOINT_SPEED_VIOLATION, 30405, "Protective Stop: joint" _PH1_ " exceeds speed limit.", "suggest...") \
00141 _D(PSTOP_TCP_SPEED_VIOLATION, 30406, "Protective Stop: TCP speed close to safety limits.", "suggest...") \
00142 _D(PSTOP_ELBOW_SPEED_VIOLATION, 30407, "Protective Stop: elbow speed close to safety limits.", "suggest...") \
00143 _D(PSTOP_TCP_FORCE_VIOLATION, 30408, "Protective Stop: TCP force close to safety limits.", "suggest...") \
00144 _D(PSTOP_ELBOW_TORQUE_VIOLATION, 30409, "Protective Stop: elbow torque close to safety limits.", "suggest...") \
00145 _D(PSTOP_POWER_VIOLATION, 30410, "Protective Stop: robot power close to safety limits.", "suggest...") \
00146 _D(PSTOP_MOMENTUM_VIOLATION, 30411, "Protective Stop: robot momentum close to safety limits.", "suggest...") \
00147 _D(PSTOP_TCP_CUBE_VIOLATION, 30412, "Protective Stop: TCP position close to safety cube.", "suggest...") \
00148 _D(PSTOP_ELBOW_CUBE_VIOLATION, 30413, "Protective Stop: TCP position close to safety cube.", "suggest...") \
00149 _D(REDUCE_ELBOW_PLANE_TRIGGER, 30414, "Reduce mode: elbow close to safety plane triggers reduction mode.",
"suggest...") \
00150 _D(REDUCE_TCP_PLANE_TRIGGER, 30415, "Reduce mode: TCP close to safety plane triggers reduction mode.",
"suggest...") \
00151 _D(PSTOP_MOVE_OUT_RANGE, 30416, "Joint " _PH1_ " has exceeded the limit, please do not continue to move out of
the range", "suggest...") \
00152 _D(RESUME_PAUSE_FAILED, 30417, "Resume Failed: Safety mode type is " _PH1_ "", "suggest...") \
00153 _D(FIRMWARE_UPDATE_FAIL_EMERGENCYSTOP, 30418, "Failed to firmware update because the robot safety mode is in "
_PPH1_, "Release emergency stop when the robot is in a safe range of motion") \
00154 _D(TOOL_SENSOR_CHANGED, 30419, "Tool sensor type changed to " _PH1_ "", "suggest...") \
00155 _D(TOOL_SENSOR_REMOVED, 30420, "Tool sensor is removed.", "suggest...") \
00156 _D(CAL_TARGET_CURRENT_ERR, 30421, "The calculation of the target current failed. Please try again later.",
"suggest...") \
00157 _D(CONVEYOR_MODE_CHANGED, 30422, "Conveyor" _PH1_ ": track mode changed to " _PH2_ ", track item id is " _PH3_,
"suggest...") \
00158 _D(CONVEYOR_ENQUEUE, 30423, "Conveyor" _PH1_ ": the queue has been changed, item" _PH2_ " is enqueue",
"suggest...") \
00159 _D(CONVEYOR_DEQUEUE_FINISH, 30424, "Conveyor" _PH1_ ": the queue has been changed, item" _PH2_ " dequeue due to
track finished", "suggest...") \
00160 _D(CONVEYOR_DEQUEUE_STARTWINDOW, 30425, "Conveyor" _PH1_ ": the queue has been changed, item" _PH2_ " dequeue
due to exceeds startwindow", "suggest...") \
00161 _D(CONVEYOR_DEQUEUE_LIMIT, 30426, "Conveyor" _PH1_ ": the queue has been changed, item" _PH2_ " dequeue due to
exceed limit area", "suggest...") \
00162 _D(CONVEYOR_DEQUEUE_CLEAR, 30427, "Conveyor" _PH1_ ": item queue is cleared", "suggest...") \
00163 _D(CONVEYOR_NEXT_TRACK, 30428, "Conveyor" _PH1_ ": item" _PH2_ " inside the start window that can be tracked ",
"suggest...") \
00164 _D(CONVEYOR_EXCEED_LIMIT, 30429, "Conveyor" _PH1_ ": item" _PH2_ " exceeds the limit area during tracking",
"suggest...") \
00165 _D(WRONG_POWER_SAFETY_LIMIT, 30430, "Robot power safety value exceeds designed value.", "suggest...") \
00166 _D(WRONG_POWER_DESIGNED_LIMIT, 30431, "Power designed value exceeds value read from hardware interface.",
"suggest...") \
00167 _D(TOOL_SENSOR_STATUS_CHANGED, 30432, "Tool sensor status changed to " _PH1_, "suggest...") \

```

```

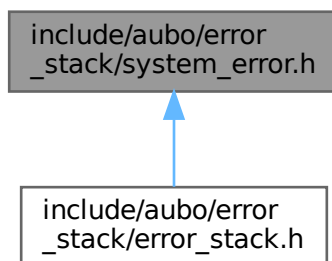
00168     _D(COLLISION_THRESHOLD_INVALID, 30433, "Robot collision threshold parameters are invalid.Please reidentify the
threshold or modify the configuration to ensure that it does not cause accidental collisions.", "suggest...") \
00169     _D(GRIPPER_DISCONNECT, 30434, "The gripper " _PH1_ " is disconnected.", "suggest...") \
00170     _D(GRIPPER_UNKNOWN_FAULT, 30435, "There is an unknown fault with the gripper " _PH1_ , "suggest...") \
00171     _D(GRIPPER_CURRENT_ANOMALY_FAULT, 30436, "There is an abnormal current fault with the gripper " _PH1_ ,
"suggest...") \
00172     _D(GRIPPER_VOLTAGE_ANOMALY_FAULT, 30437, "There is an abnormal voltage fault with the gripper " _PH1_ ,
"suggest...") \
00173     _D(GRIPPER_OVER_TEMPERATURE_FAULT, 30438, "There is an over-temperature fault with the gripper " _PH1_ ,
"suggest...") \
00174     _D(GRIPPER_INTERNAL_FAULT, 30439, "There is an internal fault with the gripper " _PH1_ , "suggest...") \
00175     _D(GRIPPER_COMMUNICATION_FAULT, 30440, "There is an communication fault with the gripper " _PH1_ , "suggest...")
\
00176     _D(GRIPPER_CONTROL_COMMAND_FAULT, 30441, "There is an control command fault with the gripper " _PH1_ ,
"suggest...") \
00177     _D(GRIPPER_ENABLE_FAULT, 30442, "There is an enable fault with the gripper " _PH1_ , "suggest...") \
00178     _D(WRIST_SINGULARITY_RISK, 30443, "Wrist singularity detected. Linear motion may cause excessive joint
speed.Adjust robot posture to avoid J5 near 0° or 180°, or use joint motion instead of linear motion.", "suggest...")
\
00179     _D(PSTOP_PATH_OFFSET_OVER_LIMIT, 30444, "Protective Stop: the offset of the robot has exceeded the limit.",
"suggest...") \
00180     _D(EXT_AXIS_SET_PARAM_FAILED_BUSY, 30445, "Set external axis parameter " _PH1_ " failed because the robot mode
is in " _PH2_ ". Please power off the robot before changing this parameter.", "Power off the robot and try again.")
\
00181     _D(WRONG_JOINT_POS_LIMIT, 30446, "Joint limit max pos less than min pos.", "suggest...") \
00182     _D(WRONG_JOINT_VEL_LIMIT, 30447, "Joint limit max vel is invalid.", "suggest...")
00183 // clang-format on
00184
00185 #endif // AUBO_SDK_RTM_ERROR_H

```

12.12 include/aubo/error_stack/system_error.h 文件参考

系统错误码

此图展示该文件被哪些文件直接或间接地引用了:



宏定义

- #define [SYSTEM_ERRORS](#)

12.12.1 详细描述

系统错误码

在文件 [system_error.h](#) 中定义.

12.12.2 宏定义说明

SYSTEM_ERRORS

#define SYSTEM_ERRORS

在文件 [error_stack.h](#) 第 8 行定义.

12.13 system_error.h

浏览该文件的文档.

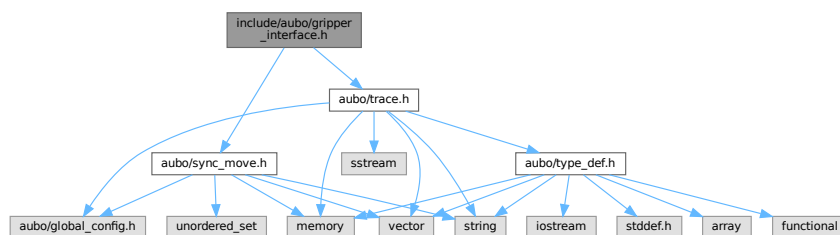
```
00001 /** @file system_error.h
00002  * @brief 系统错误码
00003  */
00004 #ifndef AUBO_SDK_SYSTEM_ERROR_H
00005 #define AUBO_SDK_SYSTEM_ERROR_H
00006
00007 #define SYSTEM_ERRORS
00008     _D(DEBUG, 0, "Debug message " _PH1_, "suggest...")
00009     _D(POPUP, 1, "Popup title: " _PH1_ " , msg: " _PH2_ " , mode: " _PH3_ ,
00010         "suggest...")
00011     _D(POPUP_DISMISS, 2, _PH1_, "suggest...")
00012     _D(SYSTEM_HALT, 3, _PH1_, "suggest...")
00013     _D(INV_ARGUMENTS, 4, "Invalid arguments.", "suggest...")
00014     _D(USER_NOTIFY, 5, _PH1_, "suggest...")
00015     _D(POPUP_DISMISS_BY_ID, 6, _PH1_, "suggest...")
00016     _D(MODBUS_SIGNAL_CREATED, 10, "Modbus signal " _PH1_ " created.",
00017         "suggest...")
00018     _D(MODBUS_SIGNAL_REMOVED, 11, "Modbus signal " _PH1_ " removed.",
00019         "suggest...")
00020     _D(MODBUS_SIGNAL_VALUE_CHANGED, 12,
00021         "Modbus signal " _PH1_ " value changed to " _PH2_ , "suggest...")
00022     _D(RUNTIME_CONTEXT, 13,
00023         "tid: " _PH1_ " lineno: " _PH2_ " index: " _PH3_ " comment: " _PH4_ ,
00024         "suggest...")
00025     _D(INTERP_CONTEXT, 14,
00026         "tid: " _PH1_ " lineno: " _PH2_ " index: " _PH3_ " comment: " _PH4_ ,
00027         "suggest...")
00028     _D(PROGRAM_LOADED, 15, "program loaded: " _PH1_ , "suggest...")
00029     _D(TASK_DELETED, 16, "tid: " _PH1_ , " was deleted")
00030     _D(MODBUS_SLAVE_BIT, 20, "Modbus slave address: " _PH1_ " value " _PH2_ ,
00031         "suggest...")
00032     _D(MODBUS_SLAVE_REG, 21, "Modbus slave address: " _PH1_ " value " _PH2_ ,
00033         "suggest...")
00034     _D(PNIO_SLAVE_SLOT_VALUE, 30,
00035         "PNIO slot: " _PH1_ " subslot " _PH2_ " index " _PH3_ " value " _PH4_ ,
00036         "suggest...")
00037     _D(PNIO_CONNECT_STATUS, 31, "PNIO connection status changed to " _PH1_ ,
00038         "suggest...")
00039     _D(PNIO_DEVICE_NAME, 32, "PNIO device name changed to " _PH1_ ,
00040         "suggest...")
00041     _D(PNIO_IP, 33, "PNIO ip " _PH1_ " mask " _PH2_ " gateway " _PH3_ ,
00042         "suggest...")
00043     _D(ICM_SERVER_STATUS, 40, " ICM server status changed to " _PH1_ ,
00044         "suggest...")
00045     _D(EIP_SLAVE_VALUE, 50,
00046         "EIP slave: trans_type " _PH1_ " index " _PH2_ " value " _PH3_ ,
00047         "suggest...")
00048     _D(EIP_SLAVE_CONNECT_STATUS, 51,
00049         "EIP slave connection status changed to " _PH1_ , "suggest...")
00050     _D(LOG_PROGRAM_SUCCESS, 100,
00051         "[" _PH1_ "] Load program " _PH2_ " successful", "suggest...")
00052     _D(LOG_PROGRAM_FAILED, 101,
00053         "[" _PH1_ "] Load program " _PH2_ " failed, file not found",
00054         "suggest...")
00055     _D(LOG_PROGRAM_FAILED2, 102,
00056         "[" _PH1_ "] Load program " _PH2_
00057         " failed, configuration file (.ins) does not match",
00058         "suggest...")
00059
00060 #endif // AUBO_SDK_SYSTEM_ERROR_H
```

12.14 include/aubo/gripper_interface.h 文件参考

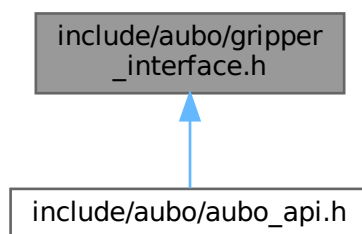
通用夹爪接口

```
#include <aubo/sync_move.h>
#include <aubo/trace.h>
```

gripper_interface.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::common_interface::GripperInterface](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::common_interface](#)

类型定义

- using [arcs::common_interface::GripperInterfacePtr](#) = std::shared_ptr<[GripperInterface](#)>

12.14.1 详细描述

通用夹爪接口

在文件 [gripper_interface.h](#) 中定义.

12.15 gripper_interface.h

[浏览该文件的文档.](#)

```

00001 /** @file gripper_interface.h
00002  * @brief 通用夹爪接口
00003  */
00004 #ifndef AUBO_SDK_GRIPPER_INTERFACE_H
00005 #define AUBO_SDK_GRIPPER_INTERFACE_H
00006
00007 #include <aubo/sync_move.h>
00008 #include <aubo/trace.h>
00009

```

```

00010 namespace arcs {
00011 namespace common_interface {
00012
00013 /**
00014  * @defgroup GripperInterface GripperInterface (夹爪)
00015  * \-chinese 通用夹爪 API 接口
00016  */
00017 class ARCS_ABI_EXPORT GripperInterface
00018 {
00019 public:
00020     GripperInterface();
00021     virtual ~GripperInterface();
00022
00023     /**
00024      * @ingroup GripperInterface
00025      * \chinese
00026      * 获取支持的所有夹爪
00027      *
00028      * @return
00029      * \endchinese
00030      */
00031     std::vector<std::string> gripperGetSupportedModels();
00032
00033     /**
00034      * @ingroup GripperInterface
00035      * \chinese
00036      * 扫描该型号下的所有设备
00037      *
00038      * @param model 夹爪品牌型号
00039      * @param device_name 设备名
00040      * @return 成功返回设备列表和 0, 失败返回错误码
00041      * \endchinese
00042      */
00043     ResultWithErrno2 gripperScanDevices(const std::string &model,
00044                                         const std::string &device_name);
00045
00046     /**
00047      * @ingroup GripperInterface
00048      * \chinese
00049      * 获取已添加的夹爪
00050      * @return
00051      */
00052     std::vector<std::string> gripperGetNames();
00053
00054     /**
00055      * @ingroup GripperInterface
00056      * \chinese
00057      * 添加夹爪
00058      *
00059      * @param name 用户自定义名字, 唯一
00060      * @param model 夹爪品牌及型号
00061      * @return 成功返回 0, 失败返回错误码
00062      * \endchinese
00063      */
00064     int gripperAdd(const std::string &name, const std::string &model);
00065
00066     /**
00067      * @ingroup GripperInterface
00068      * \chinese
00069      * 删除夹爪
00070      *
00071      * @param name 用户自定义名字
00072      * @return 成功返回 0, 失败返回错误码
00073      * \endchinese
00074      */
00075     int gripperDelete(const std::string &name);
00076
00077     /**
00078      * @ingroup GripperInterface
00079      * \chinese
00080      * 修改夹爪名
00081      *
00082      * @param name
00083      * @param new_name 新名字
00084      * @return 成功返回 0, 失败返回错误码
00085      * \endchinese
00086      */
00087     int gripperRename(const std::string &name, const std::string &new_name);
00088
00089     /**
00090      * @ingroup GripperInterface
00091      * \chinese
00092      * 夹爪连接
00093      *
00094      * @param name
00095      * @param device_name 设备名
00096      * @return 成功返回 0, 失败返回错误码

```

```

00097     * \endchinese
00098     */
00099 int gripperConnect(const std::string &name, const std::string &device_name);
00100
00101 /**
00102  * @ingroup GripperInterface
00103  * \chinese
00104  * 夹爪断开连接
00105  *
00106  * @param name
00107  * @return 成功返回 0, 失败返回错误码
00108  * \endchinese
00109  */
00110 int gripperDisconnect(const std::string &name);
00111
00112 /**
00113  * @ingroup GripperInterface
00114  * \chinese
00115  * 夹爪是否连接
00116  *
00117  * @param name
00118  * @return 已连接返回 true, 未连接返回 false
00119  * \endchinese
00120  */
00121 bool gripperIsConnected(const std::string &name);
00122
00123 /**
00124  * @ingroup GripperInterface
00125  * \chinese
00126  * 设置工作模式
00127  *
00128  * @param name
00129  * @param work_mode 工作模式
00130  * @return 成功返回 0, 失败返回错误码
00131  * \endchinese
00132  */
00133 int gripperSetWorkMode(const std::string &name, int work_mode);
00134
00135 /**
00136  * @ingroup GripperInterface
00137  * \chinese
00138  * 获取工作模式
00139  *
00140  * @param name
00141  * @return 返回工作模式
00142  * \endchinese
00143  */
00144 int gripperGetWorkMode(const std::string &name);
00145
00146 /**
00147  * @ingroup GripperInterface
00148  * \chinese
00149  * 设置夹爪安装偏移
00150  *
00151  * @param name
00152  * @param isVisibled 是否显示
00153  * @param pose 末端 (不包括手指) 相对法兰的位姿
00154  * @return 成功返回 true, 失败返回 false
00155  * \endchinese
00156  */
00157 int gripperSetMountPose(const std::string &name,
00158                        const std::vector<double> &pose,
00159                        bool enable_collision);
00160
00161 /**
00162  * @brief 获取夹爪安装偏移
00163  * @param name
00164  * @return 返回夹爪安装偏移
00165  */
00166 std::vector<double> gripperGetMountPose(const std::string &name);
00167
00168 /**
00169  * @ingroup GripperInterface
00170  * \chinese
00171  * 使能夹爪
00172  *
00173  * @param name
00174  * @return
00175  * \endchinese
00176  */
00177 int gripperEnable(const std::string &name, bool enable);
00178
00179 /**
00180  * @ingroup GripperInterface
00181  * \chinese
00182  * 夹爪是否使能
00183  *

```



```

00184     * @param name
00185     * @return
00186     * \endchinese
00187     */
00188     bool gripperIsEnabled(const std::string &name);
00189
00190     /**
00191     * @ingroup GripperInterface
00192     * \chinese
00193     * 设置运动参数
00194     *
00195     * @param name
00196     * @param position 位置, 单位: m/Pa
00197     * @param velocity_percent 速度, 单位: %, 范围: [0, 1]
00198     * @param force 夹持力, 单位: N
00199     * @param angle 旋转角度, 单位: rad
00200     * @param r_velocity_percent 旋转速度, 单位: %, 范围: [0, 1]
00201     * @param torque_percent 旋转扭矩, 单位: %, 范围: [0, 1]
00202     *
00203     * @return
00204     * \endchinese
00205     */
00206     int gripperSetPosition(const std::string &name, const double position);
00207     int gripperSetVelocity(const std::string &name,
00208                           const double velocity_percent);
00209     int gripperSetForce(const std::string &name, const double force);
00210     int gripperSetAngle(const std::string &name, const double angle);
00211     int gripperSetRVelocity(const std::string &name,
00212                             const double r_velocity_percent);
00213     int gripperSetTorque(const std::string &name, const double torque_percent);
00214
00215     /**
00216     * @ingroup GripperInterface
00217     * \chinese
00218     * 开始运动
00219     *
00220     * @param name
00221     * @return
00222     * \endchinese
00223     */
00224     int gripperMove(const std::string &name);
00225
00226     /**
00227     * @ingroup GripperInterface
00228     * \chinese
00229     * 停止运动
00230     *
00231     * @param name
00232     * @return
00233     * \endchinese
00234     */
00235     int gripperStop(const std::string &name);
00236
00237     /**
00238     * @ingroup GripperInterface
00239     * \chinese
00240     * 获取夹爪状态
00241     *
00242     * @param name
00243     * @return
00244     *
00245     * @note 位置, 速度, 夹持力, 旋转角度, 旋转速度, 旋转扭矩的单位与 Set 接口一致
00246     *
00247     * \endchinese
00248     */
00249     std::string gripperGetHardwareVersion(const std::string &name);
00250     std::string gripperGetSoftwareVersion(const std::string &name);
00251     double gripperGetPosition(const std::string &name);
00252     double gripperGetVelocity(const std::string &name);
00253     double gripperGetForce(const std::string &name);
00254     double gripperGetAngle(const std::string &name);
00255     double gripperGetRVelocity(const std::string &name);
00256     double gripperGetTorque(const std::string &name);
00257     bool gripperGetObjectDetection(const std::string &name);
00258     bool gripperGetMotionState(const std::string &name);
00259     double gripperGetVoltage(const std::string &name);
00260     double gripperGetTemperature(const std::string &name);
00261
00262     /**
00263     * @ingroup GripperInterface
00264     * \chinese
00265     * 重置 modbus 从站 ID
00266     *
00267     * @param name
00268     * @param slave_id 从站 Id
00269     * @return
00270     * \endchinese

```



```

00271     */
00272     int gripperResetSlaveId(const std::string &name, const int slave_id);
00273
00274     /**
00275     * @ingroup GripperInterface
00276     * \chinese
00277     * 获取夹爪状态
00278     * @param name
00279     * @return 状态
00280     * \endchinese
00281     */
00282     int gripperGetStatusCode(const std::string &name);
00283
00284 protected:
00285     void *d_;
00286 };
00287 using GripperInterfacePtr = std::shared_ptr<GripperInterface>;
00288
00289 } // namespace common_interface
00290 } // namespace arcs
00291
00292 #endif // AUBO_SDK_GRIPPER_INTERFACE_H

```

12.16 include/aubo/math.h 文件参考

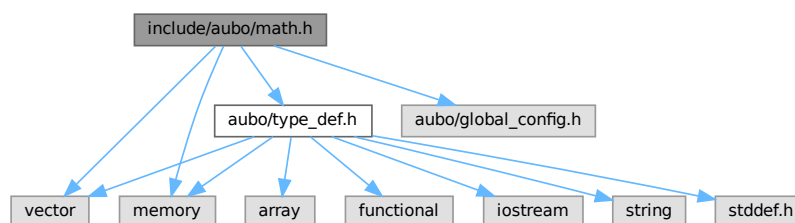
数学方法接口，如欧拉角与四元数转换、位姿的加减运算

```

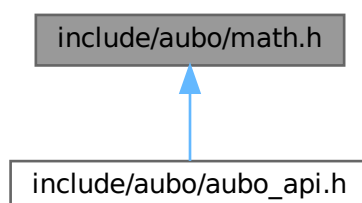
#include <vector>
#include <memory>
#include <aubo/type_def.h>
#include <aubo/global_config.h>

```

math.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::common_interface::Math](#)

命名空间

- namespace `arcs`
- namespace `arcs::common_interface`

类型定义

- using `arcs::common_interface::MathPtr` = `std::shared_ptr<Math>`

12.16.1 详细描述

数学方法接口，如欧拉角与四元数转换、位姿的加减运算
在文件 `math.h` 中定义。

12.17 math.h

[浏览该文件的文档.](#)

```
00001 /** @file math.h
00002 * \-chinese @brief 数学方法接口，如欧拉角与四元数转换、位姿的加减运算
00003 * \-english @brief Mathematic operation interface, such as euler to quaternion conversion, addition/subtraction of
00004 * poses
00005 */
00006 #ifndef AUBO_SDK_MATH_INTERFACE_H
00007 #define AUBO_SDK_MATH_INTERFACE_H
00008 #include <vector>
00009 #include <memory>
00010 #include <aubo/type_def.h>
00011 #include <aubo/global_config.h>
00012 namespace arcs {
00013 namespace common_interface {
00014 /**
00015 * @defgroup Math Math (数学工具)
00016 * \-chinese 数学工具 API 接口
00017 */
00018 class ARCS_ABI_EXPORT Math
00019 {
00020 public:
00021     Math();
00022     virtual ~Math();
00023 /**
00024 * @ingroup Math
00025 * \english
00026 * Pose addition
00027 *
00028 * Both arguments contain three position parameters (x, y, z) jointly called
00029 * P, and three rotation parameters (R_x, R_y, R_z) jointly called R. This
00030 * function calculates the result x_3 as the addition of the given poses as
00031 * follows:
00032 *
00033 * p_3.P = p_1.P + p_2.P
00034 *
00035 * p_3.R = p_1.R * p_2.R
00036 *
00037 * @param p1 Tool pose 1
00038 * @param p2 Tool pose 2
00039 * @return sum of position parts and product of rotation parts (pose)
00040 *
00041 * @par Python interface prototype
00042 * poseAdd(self: pyaubo_sdk.Math, arg0: List[float], arg1: List[float]) ->
00043 * List[float]
00044 *
00045 * @par Lua interface prototype
00046 * poseAdd(p1: table, p2: table) -> table
00047 *
00048 * @par Lua example
00049 * pose_add = poseAdd({0.2, 0.5, 0.1, 1.57, 0, 0},{0.2, 0.5, 0.6, 1.57, 0, 0})
00050 *
00051 * @par JSON-RPC request example
00052 * {"jsonrpc":"2.0","method":"Math.poseAdd","params":[[0.2, 0.5, 0.1, 1.57,
00053 * 0, 0],[0.2, 0.5, 0.6, 1.57, 0, 0]],"id":1}
00054 *
00055 * @par JSON-RPC response example
00056 * {"id":1,"jsonrpc":"2.0","result":[0.4,1.0,0.7,3.14,-0.0,0.0]}
00057 * \endenglish
00058 */
00059 }
```

```

00062      *
00063      * \chinese
00064      * 位姿相加。
00065      * 两个参数都包含三个位置参数 (x、y、z)，统称为 P，
00066      * 以及三个旋转参数 (R_x、R_y、R_z)，统称为 R。
00067      * 此函数根据以下方式计算结果 p_3，即给定位姿的相加：
00068      * p_3.P = p_1.P + p_2.P，
00069      * p_3.R = p_1.R * p_2.R
00070      *
00071      * @param p1 工具位姿 1 (pose)
00072      * @param p2 工具位姿 2 (pose)
00073      * @return 位置部分之和和旋转部分之积 (pose)
00074      *
00075      * @par Python 函数原型
00076      * poseAdd(self: pyaubo_sdk.Math, arg0: List[float], arg1: List[float]) ->
00077      * List[float]
00078      *
00079      * @par Lua 函数原型
00080      * poseAdd(p1: table, p2: table) -> table
00081      *
00082      * @par Lua 示例
00083      * pose_add = poseAdd({0.2, 0.5, 0.1, 1.57, 0, 0},{0.2, 0.5, 0.6, 1.57, 0, 0})
00084      *
00085      * @par JSON-RPC 请求示例
00086      * {"jsonrpc": "2.0", "method": "Math.poseAdd", "params": [[0.2, 0.5, 0.1, 1.57,
00087      * 0, 0],[0.2, 0.5, 0.6, 1.57, 0, 0]], "id": 1}
00088      *
00089      * @par JSON-RPC 响应示例
00090      * {"id": 1, "jsonrpc": "2.0", "result": [0.4, 1.0, 0.7, 3.14, -0.0, 0.0]}
00091      * \endchinese
00092      */
00093      std::vector<double> poseAdd(const std::vector<double> &p1,
00094                                const std::vector<double> &p2);
00095
00096      /**
00097      * @ingroup Math
00098      * \chinese
00099      * 位姿相减
00100      *
00101      * 两个参数都包含三个位置参数 (x、y、z)，统称为 P，
00102      * 以及三个旋转参数 (R_x、R_y、R_z)，统称为 R。
00103      * 此函数根据以下方式计算结果 p_3，即给定位姿的相加：
00104      * p_3.P = p_1.P - p_2.P，
00105      * p_3.R = p_1.R * p_2.R.inverse
00106      *
00107      * @param p1 工具位姿 1
00108      * @param p2 工具位姿 2
00109      * @return 位姿相减计算结果
00110      *
00111      * @par Python 函数原型
00112      * poseSub(self: pyaubo_sdk.Math, arg0: List[float], arg1: List[float]) ->
00113      * List[float]
00114      *
00115      * @par Lua 函数原型
00116      * poseSub(p1: table, p2: table) -> table
00117      *
00118      * @par Lua 示例
00119      * pose_sub = poseSub({0.2, 0.5, 0.1, 1.57, 0, 0},{0.2, 0.5, 0.6, 1.57, 0, 0})
00120      *
00121      * @par JSON-RPC 请求示例
00122      * {"jsonrpc": "2.0", "method": "Math.poseSub", "params": [[0.2, 0.5, 0.1, 1.57,
00123      * 0, 0],[0.2, 0.5, 0.6, 1.57, 0, 0]], "id": 1}
00124      *
00125      * @par JSON-RPC 响应示例
00126      * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, -0.5, 0.0, -0.0, 0.0]}
00127      * \endchinese
00128      *
00129      * \english
00130      * Pose subtraction
00131      *
00132      * Both arguments contain three position parameters (x, y, z) jointly called
00133      * P, and three rotation parameters (R_x, R_y, R_z) jointly called R. This
00134      * function calculates the result x_3 as the addition of the given poses as
00135      * follows:
00136      *
00137      * p_3.P = p_1.P - p_2.P,
00138      *
00139      * p_3.R = p_1.R * p_2.R.inverse
00140      *
00141      * @param p1 tool pose 1
00142      * @param p2 tool pose 2
00143      * @return difference between two poses
00144      *
00145      * @par Python interface prototype
00146      * poseSub(self: pyaubo_sdk.Math, arg0: List[float], arg1: List[float]) ->
00147      * List[float]
00148      *

```

```

00149      * @par Lua interface prototype
00150      * poseSub(p1: table, p2: table) -> table
00151      *
00152      * @par Lua example
00153      * pose_sub = poseSub({0.2, 0.5, 0.1, 1.57, 0, 0},{0.2, 0.5, 0.6, 1.57, 0, 0})
00154      *
00155      * @par JSON-RPC request example
00156      * {"jsonrpc":"2.0","method":"Math.poseSub","params":[[0.2, 0.5, 0.1, 1.57,
00157      * 0, 0],[0.2, 0.5, 0.6, 1.57, 0, 0]],"id":1}
00158      *
00159      * @par JSON-RPC response example
00160      * {"id":1,"jsonrpc":"2.0","result":[0.0,0.0,-0.5,0.0,-0.0,0.0]}
00161      * \endenglish
00162      */
00163      std::vector<double> poseSub(const std::vector<double> &p1,
00164                                const std::vector<double> &p2);
00165
00166      /**
00167      * @ingroup Math
00168      * \chinese
00169      * 计算线性插值
00170      *
00171      * @param p1 起点的 TCP 位姿
00172      * @param p2 终点的 TCP 位姿
00173      * @param alpha 系数,
00174      * 当 0<alpha<1, 返回 p1 和 p2 两点直线的之间靠近 p1 端且占总路径比例为 alpha 的点;
00175      * 例如当 alpha=0.3, 返回的是靠近 p1 那端, 总路径的百分之 30 的点;
00176      * 当 alpha>1, 返回 p2;
00177      * 当 alpha<0, 返回 p1;
00178      * @return 插值计算结果
00179      *
00180      * @par Python 函数原型
00181      * interpolatePose(self: pyaubo_sdk.Math, arg0: List[float], arg1:
00182      * List[float], arg2: float) -> List[float]
00183      *
00184      * @par Lua 函数原型
00185      * interpolatePose(p1: table, p2: table, alpha: number) -> table
00186      *
00187      * @par Lua 示例
00188      * pose_interpolate = interpolatePose({0.2, 0.2, 0.4, 0, 0, 0},{0.2, 0.2, 0.6, 0, 0, 0},0.5)
00189      *
00190      * @par JSON-RPC 请求示例
00191      * {"jsonrpc":"2.0","method":"Math.interpolatePose","params":[[0.2, 0.2,
00192      * 0.4, 0, 0, 0],[0.2, 0.2, 0.6, 0, 0, 0],0.5],"id":1}
00193      *
00194      * @par JSON-RPC 响应示例
00195      * {"id":1,"jsonrpc":"2.0","result":[0.2,0.2,0.5,0.0,-0.0,0.0]}
00196      * \endchinese
00197      *
00198      * \english
00199      * Calculate linear interpolation
00200      *
00201      * @param p1 starting TCP pose
00202      * @param p2 ending TCP pose
00203      * @param alpha coefficient;
00204      * When 0<alpha<1, return a point between p1 & p2 that is closer to p1, at alpha percentage of the path;
00205      * For example when alpha=0.3,point returned is closer to p1, at 30% of the total distance;
00206      * When alpha>1, return p2;
00207      * When alpha<0,return p1;
00208      * @return interpolation result
00209      *
00210      * @par Python interface prototype
00211      * interpolatePose(self: pyaubo_sdk.Math, arg0: List[float], arg1:
00212      * List[float], arg2: float) -> List[float]
00213      *
00214      * @par Lua interface prototype
00215      * interpolatePose(p1: table, p2: table, alpha: number) -> table
00216      *
00217      * @par Lua example
00218      * pose_interpolate = interpolatePose({0.2, 0.2, 0.4, 0, 0, 0},{0.2, 0.2, 0.6, 0, 0, 0},0.5)
00219      *
00220      * @par JSON-RPC request example
00221      * {"jsonrpc":"2.0","method":"Math.interpolatePose","params":[[0.2, 0.2,
00222      * 0.4, 0, 0, 0],[0.2, 0.2, 0.6, 0, 0, 0],0.5],"id":1}
00223      *
00224      * @par JSON-RPC response example
00225      * {"id":1,"jsonrpc":"2.0","result":[0.2,0.2,0.5,0.0,-0.0,0.0]}
00226      * \endenglish
00227      *
00228      */
00229      std::vector<double> interpolatePose(const std::vector<double> &p1,
00230                                         const std::vector<double> &p2,
00231                                         double alpha);
00232
00233      /**
00234      * @ingroup Math
00235      * \chinese
00236      * 位姿变换

```

```

00236 *
00237 * 第一个参数 p_from 用于转换第二个参数 p_from_to, 并返回结果。
00238 * 这意味着结果是从 p_from 的坐标系开始,
00239 * 然后在该坐标系中移动 p_from_to 后的位姿。
00240 *
00241 * 这个函数可以从两个不同的角度来看。
00242 * 一种是函数将 p_from_to 根据 p_from 的参数进行转换, 即平移和旋转。
00243 * 另一种是函数被用于获取结果姿态, 先对 p_from 进行移动, 然后再对 p_from_to
00244 * 进行移动。如果将姿态视为转换矩阵, 它看起来像是:
00245 *
00246 *  $T_{world \rightarrow to} = T_{world \rightarrow from} * T_{from \rightarrow to}$ ,
00247 *  $T_x \rightarrow to = T_x \rightarrow from * T_{from \rightarrow to}$ 
00248 *
00249 * 这两个方程描述了姿态变换的基本原理, 根据给定的起始姿态和相对于起始姿态的姿态变化, 可以计算出目标姿态。
00250 *
00251 * 举个例子, 已知 B 相对于 A 的位姿、C 相对于 B 的位姿, 求 C 相对于 A 的位姿。
00252 * 第一个参数是 B 相对于 A 的位姿, 第二个参数是 C 相对于 B 的位姿,
00253 * 返回值是 C 相对于 A 的位姿。
00254 *
00255 * @param pose_from 起始位姿 (空间向量)
00256 * @param pose_from_to 相对于起始位姿的姿态变化 (空间向量)
00257 * @return 结果位姿 (空间向量)
00258 *
00259 * @par Python 函数原型
00260 * poseTrans(self: pyaubo_sdk.Math, arg0: List[float], arg1: List[float]) ->
00261 * List[float]
00262 *
00263 * @par Lua 函数原型
00264 * poseTrans(pose_from: table, pose_from_to: table) -> table
00265 *
00266 * @par Lua 示例
00267 * pose_trans = poseTrans({0.2, 0.5, 0.1, 1.57, 0, 0},{0.2, 0.5, 0.6, 1.57, 0, 0})
00268 *
00269 * @par JSON-RPC 请求示例
00270 * {"jsonrpc":"2.0","method":"Math.poseTrans","params":[[0.2, 0.5,
00271 * 0.1, 1.57, 0, 0],[0.2, 0.5, 0.6, 1.57, 0, 0]], "id":1}
00272 *
00273 * @par JSON-RPC 响应示例
00274 * {"id":1,"jsonrpc":"2.0","result":[0.4,-0.09960164640373415,0.6004776374923573,3.14,-0.0,0.0]}
00275 * \endchinese
00276 *
00277 * \english
00278 * Pose transformation
00279 *
00280 * The first argument, p_from, is used to transform the second argument,
00281 * p_from_to, and the result is then returned. This means that the result is
00282 * the resulting pose, when starting at the coordinate system of p_from, and
00283 * then in that coordinate system moving p_from_to.
00284 *
00285 * This function can be seen in two different views. Either the function
00286 * transforms, that is translates and rotates, p_from_to by the parameters
00287 * of p_from. Or the function is used to get the resulting pose, when first
00288 * making a move of p_from and then from there, a move of p_from_to. If the
00289 * poses were regarded as transformation matrices, it would look like:
00290 *
00291 *  $T_{world \rightarrow to} = T_{world \rightarrow from} * T_{from \rightarrow to}$ ,
00292 *  $T_x \rightarrow to = T_x \rightarrow from * T_{from \rightarrow to}$ 
00293 *
00294 *
00295 * These two equations describes the foundations for pose transformation.
00296 * Based on a starting pose and the pose transformation relative to the starting pose, we can get the target
pose.
00297 *
00298 * For example, we know pose of B relative to A, pose of C relative to B, find pose of C relative to A.
00299 * param 1 is pose of B relative to A, param 2 is pose of C relative to B,
00300 * the return value is the pose of C relative to A
00301 *
00302 * @param pose_from starting pose (vector in 3D space)
00303 * @param pose_from_to pose transformation relative to starting pose (vector in 3D space)
00304 * @return final pose (vector in 3D space)
00305 *
00306 * @par Python interface prototype
00307 * poseTrans(self: pyaubo_sdk.Math, arg0: List[float], arg1: List[float]) ->
00308 * List[float]
00309 *
00310 * @par Lua interface prototype
00311 * poseTrans(pose_from: table, pose_from_to: table) -> table
00312 *
00313 * @par Lua example
00314 * pose_trans = poseTrans({0.2, 0.5, 0.1, 1.57, 0, 0},{0.2, 0.5, 0.6, 1.57, 0, 0})
00315 *
00316 * @par JSON-RPC request example
00317 * {"jsonrpc":"2.0","method":"Math.poseTrans","params":[[0.2, 0.5,
00318 * 0.1, 1.57, 0, 0],[0.2, 0.5, 0.6, 1.57, 0, 0]], "id":1}
00319 *
00320 * @par JSON-RPC response example
00321 * {"id":1,"jsonrpc":"2.0","result":[0.4,-0.09960164640373415,0.6004776374923573,3.14,-0.0,0.0]}

```

```

00322     * \endenglish
00323     */
00324     std::vector<double> poseTrans(const std::vector<double> &pose_from,
00325                                 const std::vector<double> &pose_from_to);
00326
00327 /**
00328  * \english
00329  * Pose inverse transformation
00330  *
00331  * Given pose of C relative to A, pose of C relative to B, find pose of B relative to A.
00332  * param 1 is pose of C relative to A, param 2 is pose of C relative to B,
00333  * the return value is the pose of B relative to A
00334  *
00335  * @param pose_from starting pose
00336  * @param pose_to_from pose transformation relative to final pose
00337  * @return resulting pose
00338  *
00339  * @par Python interface prototype
00340  * poseTransInv(self: pyaubo_sdk.Math, arg0: List[float], arg1: List[float])
00341  * -> List[float]
00342  *
00343  * @par Lua interface prototype
00344  * poseTransInv(pose_from: table, pose_to_from: table) -> table
00345  *
00346  * @par Lua example
00347  * pose_trans_inv = poseTransInv({0.4, -0.0996016, 0.600478, 3.14, 0, 0},{0.2, 0.5, 0.6, 1.57, 0, 0})
00348  *
00349  * @par JSON-RPC request example
00350  * {"jsonrpc":"2.0","method":"Math.poseTransInv","params":[[0.4, -0.0996016,
00351  * 0.600478, 3.14, 0, 0],[0.2, 0.5, 0.6, 1.57, 0, 0]],"id":1}
00352  *
00353  * @par JSON-RPC response example
00354  * {"id":1,"jsonrpc":"2.0","result":[0.2,0.5000000464037341,0.10000036250764266,1.57,-0.0,0.0]}
00355  * \endenglish
00356  *
00357  * @ingroup Math
00358  * \chinese
00359  * 姿态逆变换
00360  *
00361  * 已知 C 相对于 A 的位姿、C 相对于 B 的位姿，求 B 相对于 A 的位姿。
00362  * 第一个参数是 C 相对于 A 的位姿，第二个参数是 C 相对于 B 的位姿，
00363  * 返回值是 B 相对于 A 的位姿。
00364  *
00365  * @param pose_from 起始位姿
00366  * @param pose_to_from 相对于结果位姿的姿态变化
00367  * @return 结果位姿
00368  *
00369  * @par Python 函数原型
00370  * poseTransInv(self: pyaubo_sdk.Math, arg0: List[float], arg1: List[float])
00371  * -> List[float]
00372  *
00373  * @par Lua 函数原型
00374  * poseTransInv(pose_from: table, pose_to_from: table) -> table
00375  *
00376  * @par Lua 示例
00377  * pose_trans_inv = poseTransInv({0.4, -0.0996016, 0.600478, 3.14, 0, 0},{0.2, 0.5, 0.6, 1.57, 0, 0})
00378  *
00379  * @par JSON-RPC 请求示例
00380  * {"jsonrpc":"2.0","method":"Math.poseTransInv","params":[[0.4, -0.0996016,
00381  * 0.600478, 3.14, 0, 0],[0.2, 0.5, 0.6, 1.57, 0, 0]],"id":1}
00382  *
00383  * @par JSON-RPC 响应示例
00384  * {"id":1,"jsonrpc":"2.0","result":[0.2,0.5000000464037341,0.10000036250764266,1.57,-0.0,0.0]}
00385  * \endchinese
00386  */
00387     std::vector<double> poseTransInv(const std::vector<double> &pose_from,
00388                                     const std::vector<double> &pose_to_from);
00389
00390 /**
00391  * @ingroup Math
00392  * \chinese
00393  * 获取位姿的逆
00394  *
00395  * @param pose 工具位姿 (空间向量)
00396  * @return 工具位姿的逆转换 (空间向量)
00397  *
00398  * @par Python 函数原型
00399  * poseInverse(self: pyaubo_sdk.Math, arg0: List[float]) -> List[float]
00400  *
00401  * @par Lua 函数原型
00402  * poseInverse(pose: table) -> table
00403  *
00404  * @par Lua 示例
00405  * pose_inverse = poseInverse({0.2, 0.5, 0.1, 1.57, 0, 3.14})
00406  *
00407  * @par JSON-RPC 请求示例
00408  * {"jsonrpc":"2.0","method":"Math.poseInverse","params":[[0.2, 0.5,

```

```

00409     * 0.1, 1.57, 0, 3.14]], "id": 1}
00410     *
00411     * @par JSON-RPC 响应示例
00412     * {"id": 1, "jsonrpc": "2.0", "result": [0.19920341988726448, -0.09960155178838484, -0.5003973704832628,
00413     * 1.569999989900404, -0.0015926530848129354, -3.1415913853161266]}
00414     *
00415     * \endchinese
00416     *
00417     * \english
00418     * Get the inverse of a pose
00419     *
00420     * @param pose tool pose (spatial vector)
00421     * @return inverse tool pose transformation (spatial vector)
00422     *
00423     * @par Python interface prototype
00424     * poseInverse(self: pyaubo_sdk.Math, arg0: List[float]) -> List[float]
00425     *
00426     * @par Lua interface prototype
00427     * poseInverse(pose: table) -> table
00428     *
00429     * @par Lua example
00430     * pose_inverse = poseInverse({0.2, 0.5, 0.1, 1.57, 0, 3.14})
00431     *
00432     * @par JSON-RPC request example
00433     * {"jsonrpc": "2.0", "method": "Math.poseInverse", "params": [[0.2, 0.5,
00434     * 0.1, 1.57, 0, 3.14]], "id": 1}
00435     *
00436     * @par JSON-RPC response example
00437     * {"id": 1, "jsonrpc": "2.0", "result": [0.19920341988726448, -0.09960155178838484, -0.5003973704832628,
00438     * 1.569999989900404, -0.0015926530848129354, -3.1415913853161266]}
00439     * \endenglish
00440     *
00441     */
00442 std::vector<double> poseInverse(const std::vector<double> &pose);
00443
00444 /**
00445  * @ingroup Math
00446  * \chinese
00447  * 计算两个位姿的位置距离
00448  *
00449  * @param p1 位姿 1
00450  * @param p2 位姿 2
00451  * @return 两个位姿的位置距离
00452  *
00453  * @par Lua 函数原型
00454  * poseDistance(p1: table, p2: table) -> number
00455  *
00456  * @par Lua 示例
00457  * pose_distance = poseDistance({0.1, 0.3, 0.1, 0.3142, 0.0, 1.571}, {0.2, 0.5, 0.6, 0, -0.172, 0.0})
00458  *
00459  * @par JSON-RPC 请求示例
00460  * {"jsonrpc": "2.0", "method": "Math.poseDistance", "params": [[0.1, 0.3, 0.1,
00461  * 0.3142, 0.0, 1.571], [0.2, 0.5, 0.6, 0, -0.172, 0.0]], "id": 1}
00462  *
00463  * @par JSON-RPC 响应示例
00464  * {"id": 1, "jsonrpc": "2.0", "result": 0.5477225575051661}
00465  * \endchinese
00466  *
00467  * \english
00468  * Calculate distance between two poses
00469  *
00470  * @param p1 pose 1
00471  * @param p2 pose 2
00472  * @return distance between the poses
00473  *
00474  * @par Lua function prototype
00475  * poseDistance(p1: table, p2: table) -> number
00476  *
00477  * @par Lua example
00478  * pose_distance = poseDistance({0.1, 0.3, 0.1, 0.3142, 0.0, 1.571}, {0.2, 0.5, 0.6, 0, -0.172, 0.0})
00479  *
00480  * @par JSON-RPC request example
00481  * {"jsonrpc": "2.0", "method": "Math.poseDistance", "params": [[0.1, 0.3, 0.1,
00482  * 0.3142, 0.0, 1.571], [0.2, 0.5, 0.6, 0, -0.172, 0.0]], "id": 1}
00483  *
00484  * @par JSON-RPC response example
00485  * {"id": 1, "jsonrpc": "2.0", "result": 0.5477225575051661}
00486  * \endenglish
00487  *
00488  */
00489 double poseDistance(const std::vector<double> &p1,
00490     const std::vector<double> &p2);
00491
00492 /**
00493  * @ingroup Math
00494  * \chinese
00495  * 计算两个位姿的轴角距离

```

```

00496      *
00497      * @param p1 位姿 1
00498      * @param p2 位姿 2
00499      * @return 轴角距离
00500      *
00501      * @par Lua 函数原型
00502      * poseAngleDistance(p1: table, p2: table) -> number
00503      *
00504      * @par Lua 示例
00505      * pose_angle_distance = poseAngleDistance({0.1, 0.3, 0.1, 0.3142, 0.0, 1.571},{0.2, 0.5, 0.6, 0, -0.172, 0.0})
00506      *
00507      * \endchinese
00508      *
00509      * \english
00510      * Calculate axis-angle difference between two poses
00511      *
00512      * @param p1 pose 1
00513      * @param p2 pose 2
00514      * @return axis angle difference
00515      *
00516      * @par Lua 函数原型
00517      * poseAngleDistance(p1: table, p2: table) -> number
00518      *
00519      * @par Lua 示例
00520      * pose_angle_distance = poseAngleDistance({0.1, 0.3, 0.1, 0.3142, 0.0, 1.571},{0.2, 0.5, 0.6, 0, -0.172, 0.0})
00521      *
00522      * \endenglish
00523      */
00524 double poseAngleDistance(const std::vector<double> &p1,
00525
00526                          const std::vector<double> &p2);
00527
00528 /**
00529  * @ingroup Math
00530  * \chinese
00531  * 判断两个位姿是否相等
00532  *
00533  * @param p1 位姿 1
00534  * @param p2 位姿 2
00535  * @param eps 误差
00536  * @return 相等返回 true, 反之返回 false
00537  *
00538  * @par Lua 函数原型
00539  * poseEqual(p1: table, p2: table, eps: number) -> boolean
00540  *
00541  * @par Lua 示例
00542  * pose_equal = poseEqual({0.1, 0.3, 0.1, 0.3142, 0.0, 1.571},{0.1, 0.3, 0.1, 0.3142, 0.0, 1.5711},0.01)
00543  *
00544  * @par JSON-RPC 请求示例
00545  * {"jsonrpc":"2.0","method":"Math.poseEqual","params":[[0.1, 0.3, 0.1,
00546  * 0.3142, 0.0, 1.571],[0.1, 0.3, 0.1, 0.3142, 0.0, 1.5711],0.01]], "id":1}
00547  *
00548  * @par JSON-RPC 响应示例
00549  * {"id":1,"jsonrpc":"2.0","result":0.0}
00550  * \endchinese
00551  *
00552  * \english
00553  * Determine if two poses are equivalent
00554  *
00555  * @param p1 pose 1
00556  * @param p2 pose 2
00557  * @param eps error margin
00558  * @return true or false
00559  *
00560  * @par Lua 函数原型
00561  * poseEqual(p1: table, p2: table, eps: number) -> boolean
00562  *
00563  * @par Lua 示例
00564  * pose_equal = poseEqual({0.1, 0.3, 0.1, 0.3142, 0.0, 1.571},{0.1, 0.3, 0.1, 0.3142, 0.0, 1.5711},0.01)
00565  *
00566  * @par JSON-RPC request example
00567  * {"jsonrpc":"2.0","method":"Math.poseEqual","params":[[0.1, 0.3, 0.1,
00568  * 0.3142, 0.0, 1.571],[0.1, 0.3, 0.1, 0.3142, 0.0, 1.5711],0.01]], "id":1}
00569  *
00570  * @par JSON-RPC response example
00571  * {"id":1,"jsonrpc":"2.0","result":0.0}
00572  * \endenglish
00573  */
00574 bool poseEqual(const std::vector<double> &p1, const std::vector<double> &p2,
00575               double eps = 5e-5);
00576
00577 /**
00578  * @ingroup Math
00579  * \chinese
00580  * @param F_b_a_old
00581  * @param V_in_a
00582  * @param type

```



```

00583     * @return
00584     *
00585     * @par Python 函数原型
00586     * transferRefFrame(self: pyaubo_sdk.Math, arg0: List[float], arg1:
00587     * List[float[3]], arg2: int) -> List[float]
00588     *
00589     * @par Lua 函数原型
00590     * transferRefFrame(F_b_a_old: table, V_in_a: table, type: number) -> table
00591     * \endchinese
00592     *
00593     * \english
00594     * @param F_b_a_old
00595     * @param V_in_a
00596     * @param type
00597     * @return
00598     *
00599     * @par Python interface prototype
00600     * transferRefFrame(self: pyaubo_sdk.Math, arg0: List[float], arg1:
00601     * List[float[3]], arg2: int) -> List[float]
00602     *
00603     * @par Lua interface prototype
00604     * transferRefFrame(F_b_a_old: table, V_in_a: table, type: number) -> table
00605     * \endenglish
00606     */
00607 std::vector<double> transferRefFrame(const std::vector<double> &F_b_a_old,
00608                                     const Vector3d &V_in_a, int type);
00609
00610 /**
00611  * @ingroup Math
00612  * \chinese
00613  * 姿态旋转
00614  *
00615  * @param pose
00616  * @param rotv
00617  * @return
00618  *
00619  * @par Python 函数原型
00620  * poseRotation(self: pyaubo_sdk.Math, arg0: List[float], arg1: List[float])
00621  * -> List[float]
00622  *
00623  * @par Lua 函数原型
00624  * poseRotation(pose: table, rotv: table) -> table
00625  * \endchinese
00626  *
00627  * \english
00628  * Pose rotation
00629  *
00630  * @param pose
00631  * @param rotv
00632  * @return
00633  *
00634  * @par Python interface prototype
00635  * poseRotation(self: pyaubo_sdk.Math, arg0: List[float], arg1: List[float])
00636  * -> List[float]
00637  *
00638  * @par Lua interface prototype
00639  * poseRotation(pose: table, rotv: table) -> table
00640  * \endenglish
00641  */
00642 std::vector<double> poseRotation(const std::vector<double> &pose,
00643                                 const std::vector<double> &rotv);
00644
00645 /**
00646  * @ingroup Math
00647  * \chinese
00648  * 欧拉角转四元数
00649  *
00650  * @param rpy 欧拉角
00651  * @return 四元数
00652  *
00653  * @par Python 函数原型
00654  * rpyToQuaternion(self: pyaubo_sdk.Math, arg0: List[float]) -> List[float]
00655  *
00656  * @par Lua 函数原型
00657  * rpyToQuaternion(rpy: table) -> table
00658  *
00659  * @par Lua 示例
00660  * quaternion = rpyToQuaternion({0.611, 0.785, 0.960})
00661  *
00662  * @par JSON-RPC 请求示例
00663  * {"jsonrpc": "2.0", "method": "Math.rpyToQuaternion", "params": [[0.611, 0.785,
00664  * 0.960]], "id": 1}
00665  *
00666  * @par JSON-RPC 响应示例
00667  * {"id": 1, "jsonrpc": "2.0", "result":
00668  * [0.834721517970497, 0.07804256900772265, 0.4518931575790371, 0.3048637712043723]}
00669  * \endchinese

```

```

00669      *
00670      * \english
00671      * Euler angles to quaternions
00672      *
00673      * @param rpy euler angles
00674      * @return quaternions
00675      *
00676      * @par Python interface prototype
00677      * rpyToQuaternion(self: pyaubo_sdk.Math, arg0: List[float]) -> List[float]
00678      *
00679      * @par Lua interface prototype
00680      * rpyToQuaternion(rpy: table) -> table
00681      *
00682      * @par Lua example
00683      * quaternion = rpyToQuaternion({0.611, 0.785, 0.960})
00684      *
00685      * @par JSON-RPC request example
00686      * {"jsonrpc": "2.0", "method": "Math.rpyToQuaternion", "params": [[0.611, 0.785,
00687      * 0.960]], "id": 1}
00688      *
00689      * @par JSON-RPC response example
00690      * {"id": 1, "jsonrpc": "2.0", "result":
[0.834721517970497, 0.07804256900772265, 0.4518931575790371, 0.3048637712043723]}
00691      * \endenglish
00692      */
00693      std::vector<double> rpyToQuaternion(const std::vector<double> &rpy);
00694
00695      /**
00696      * @ingroup Math
00697      * \chinese
00698      * 四元数转欧拉角
00699      *
00700      * @param quat 四元数
00701      * @return 欧拉角
00702      *
00703      * @par Python 函数原型
00704      * quaternionToRpy(self: pyaubo_sdk.Math, arg0: List[float]) -> List[float]
00705      *
00706      * @par Lua 函数原型
00707      * quaternionToRpy(quat: table) -> table
00708      *
00709      * @par Lua 示例
00710      * rpy = quaternionToRpy({0.834722, 0.0780426, 0.451893, 0.304864})
00711      *
00712      * @par JSON-RPC 请求示例
00713      * {"jsonrpc": "2.0", "method": "Math.quaternionToRpy", "params": [[0.834722,
00714      * 0.0780426, 0.451893, 0.304864]], "id": 1}
00715      *
00716      * @par JSON-RPC 响应示例
00717      * {"id": 1, "jsonrpc": "2.0", "result": [0.6110000520523781, 0.7849996877683915, 0.960000543982093]}
00718      * \endchinese
00719      *
00720      * \english
00721      * Quaternions to euler angles
00722      *
00723      * @param quat quaternions
00724      * @return euler angles
00725      *
00726      * @par Python interface prototype
00727      * quaternionToRpy(self: pyaubo_sdk.Math, arg0: List[float]) -> List[float]
00728      *
00729      * @par Lua interface prototype
00730      * quaternionToRpy(quat: table) -> table
00731      *
00732      * @par Lua example
00733      * rpy = quaternionToRpy({0.834722, 0.0780426, 0.451893, 0.304864})
00734      *
00735      * @par JSON-RPC request example
00736      * {"jsonrpc": "2.0", "method": "Math.quaternionToRpy", "params": [[0.834722,
00737      * 0.0780426, 0.451893, 0.304864]], "id": 1}
00738      *
00739      * @par JSON-RPC response example
00740      * {"id": 1, "jsonrpc": "2.0", "result": [0.6110000520523781, 0.7849996877683915, 0.960000543982093]}
00741      * \endenglish
00742      */
00743      std::vector<double> quaternionToRpy(const std::vector<double> &quat);
00744
00745      /**
00746      * @ingroup Math
00747      * \chinese
00748      * 四点法标定 TCP 偏移
00749      *
00750      * 找一个尖点，将机械臂工具末端点绕着尖点示教四个位置，姿态差别要大。
00751      * 设置完毕后即可计算出来结果。
00752      *
00753      * @param poses 四个点的位姿集合
00754      * @return TCP 标定结果和标定结果是否有效

```

```

00755      *
00756      * @par Python 函数原型
00757      * tcpOffsetIdentify(self: pyaubo_sdk.Math, arg0: List[List[float]]) ->
00758      * Tuple[List[float], int]
00759      *
00760      * @par Lua 函数原型
00761      * tcpOffsetIdentify(poses: table) -> table
00762      *
00763      * @par Lua 示例
00764      * p1 = {0.48659,0.10456,0.24824,-3.135,0.004,1.569} \n
00765      * p2 = {0.38610,0.09096,0.22432,2.552,-0.437,1.008} \n
00766      * p3 = {0.38610,0.05349,0.16791,-3.021,-0.981,0.517} \n
00767      * p4 = {0.49675,0.06417,0.21408,-2.438,0.458,0.651} \n
00768      * p = {p1,p2,p3,p4} \n
00769      * tcp_result = tcpOffsetIdentify(p)
00770      *
00771      * \endchinese
00772      *
00773      * \english
00774      * Four point method calibration for TCP offset
00775      *
00776      * About a sharp point, move the robot's tcp in four different poses. Difference between each pose should be
drastic.
00777      * Result can be obtained based on these four poses
00778      *
00779      * @param poses combination of four different poses
00780      * @return TCP calibration result and whether successfull
00781      *
00782      * @par Python interface prototype
00783      * tcpOffsetIdentify(self: pyaubo_sdk.Math, arg0: List[List[float]]) ->
00784      * Tuple[List[float], int]
00785      *
00786      * @par Lua interface prototype
00787      * tcpOffsetIdentify(poses: table) -> table
00788      *
00789      * @par Lua example
00790      * p1 = {0.48659,0.10456,0.24824,-3.135,0.004,1.569} \n
00791      * p2 = {0.38610,0.09096,0.22432,2.552,-0.437,1.008} \n
00792      * p3 = {0.38610,0.05349,0.16791,-3.021,-0.981,0.517} \n
00793      * p4 = {0.49675,0.06417,0.21408,-2.438,0.458,0.651} \n
00794      * p = {p1,p2,p3,p4} \n
00795      * tcp_result = tcpOffsetIdentify(p)
00796      *
00797      * \endenglish
00798      */
00799      ResultWithErrno tcpOffsetIdentify(
00800          const std::vector<std::vector<double>> &poses);
00801
00802      /**
00803      * @ingroup Math
00804      * \chinese
00805      * 三点法标定坐标系
00806      *
00807      * @param poses 三个点的位姿集合
00808      * @param type 类型:\n
00809      *      0 - oxy 原点 x 轴正方向 xy 平面 (y 轴正方向) \n
00810      *      1 - oxz 原点 x 轴正方向 xz 平面 (z 轴正方向) \n
00811      *      2 - oyz 原点 y 轴正方向 yz 平面 (z 轴正方向) \n
00812      *      3 - oyx 原点 y 轴正方向 yx 平面 (x 轴正方向) \n
00813      *      4 - ozx 原点 z 轴正方向 zx 平面 (x 轴正方向) \n
00814      *      5 - ozy 原点 z 轴正方向 zy 平面 (y 轴正方向) \n
00815      * @return 坐标系标定结果和标定结果是否有效
00816      *
00817      * @par Lua 函数原型
00818      * calibrateCoordinate(poses: table, type: int) -> table, number
00819      *
00820      * @par Lua 示例
00821      * p1 = {0.55462,0.06219,0.37175,-3.142,0.0,1.580} \n
00822      * p2 = {0.63746,0.11805,0.37175,-3.142,0.0,1.580} \n
00823      * p3 = {0.40441,0.28489,0.37174,-3.142,0.0,1.580} \n
00824      * p = {p1,p2,p3} \n
00825      * coord_pose, cal_result = calibrateCoordinate(p,0)
00826      *
00827      * @par JSON-RPC 请求示例
00828      * {"jsonrpc":"2.0","method":"Math.calibrateCoordinate","params": [[[0.55462,0.06219,0.37175,-3.142,0.0,1.580],
00829      * [0.63746,0.11805,0.37175,-3.142,0.0,1.580],[0.40441,0.28489,0.37174,-3.142,0.0,1.580]],0],"id":1}
00830      *
00831      * @par JSON-RPC 响应示例
00832      * {"id":1,"jsonrpc":"2.0","result":
[[0.55462,0.06219,0.37175,-3.722688983883945e-05,-1.6940658945086007e-21,0.5932768162455785],0]}
00833      * \endchinese
00834      *
00835      * \english
00836      * Calibrate coordinate system with 3 points
00837      *
00838      * @param poses set of 3 poses
00839      * @param type type:\n

```

```

00840      *      0 - oxy origin, +x axis, xy plane (+y direction) \n
00841      *      1 - oxz origin, +x axis, xz plane (+z direction) \n
00842      *      2 - oyz origin, +y axis, yz plane (+z direction) \n
00843      *      3 - oyx origin, +y axis, yx plane (+x direction) \n
00844      *      4 - ozx origin, +z axis, zx plane (+x direction) \n
00845      *      5 - ozy origin, +z axis, zy plane (+y direction) \n
00846      * @return Coordinate system calibration result and whether the calibration result is valid
00847      *
00848      * @par Lua function prototype
00849      * calibrateCoordinate(poses: table, type: int) -> table, number
00850      *
00851      * @par Lua example
00852      * p1 = {0.55462,0.06219,0.37175,-3.142,0.0,1.580} \n
00853      * p2 = {0.63746,0.11805,0.37175,-3.142,0.0,1.580} \n
00854      * p3 = {0.40441,0.28489,0.37174,-3.142,0.0,1.580} \n
00855      * p = {p1,p2,p3} \n
00856      * coord_pose, cal_result = calibrateCoordinate(p,0)
00857      *
00858      * @par JSON-RPC request example
00859      * {"jsonrpc":"2.0","method":"Math.calibrateCoordinate","params": [[[0.55462,0.06219,0.37175,-3.142,0.0,1.580],
00860      * [0.63746,0.11805,0.37175,-3.142,0.0,1.580], [0.40441,0.28489,0.37174,-3.142,0.0,1.580]],0], "id":1}
00861      *
00862      * @par JSON-RPC response example
00863      * {"id":1,"jsonrpc":"2.0","result":
[[0.55462,0.06219,0.37175,-3.722688983883945e-05,-1.6940658945086007e-21,0.5932768162455785],0]}
00864      * \endenglish
00865      */
00866      ResultWithErrno calibrateCoordinate(
00867          const std::vector<std::vector<double>> &poses, int type);
00868
00869      /**
00870      * @ingroup Math
00871      * \chinese
00872      * 根据圆弧的三个点，计算出拟合成的圆的另一半圆弧的中间点位置
00873      *
00874      * @param p1 圆弧的起始点
00875      * @param p2 圆弧的中间点
00876      * @param p3 圆弧的结束点
00877      * @param mode 当 mode 等于 1 的时候，表示需要对姿态进行圆弧规划；
00878      * 当 mode 等于 0 的时候，表示不需要对姿态进行圆弧规划
00879      *
00880      * @return 拟合成的圆的另一半圆弧的中间点位置和计算结果是否有效
00881      *
00882      * @par Lua 函数原型
00883      * calculateCircleFourthPoint(p1: table, p2: table, p3: table, mode: int)
00884      *
00885      * @par Lua 示例
00886      * center_pose, cal_result =
calculateCircleFourthPoint({0.5488696249770836,-0.1214996547187204,0.2631931199112321,-3.14159198038469,-3.673205103150083e-06,1.5707963
00887      *
00888      * {0.5488696249770835,-0.1214996547187207,0.3599720701808493,-3.14159198038469,-3.6732051029273e-06,1.570796326792423},
00889      *
00890      * {0.5488696249770836,-0.0389996547187214,0.3599720701808496,-3.141591980384691,-3.673205102557476e-06,1.570796326792422},1)
00891      *
00892      * @par JSON-RPC 请求示例
00893      * {"jsonrpc":"2.0","method":"Math.calculateCircleFourthPoint","params":
[[0.5488696249770836,-0.1214996547187204,0.2631931199112321,-3.14159198038469,-3.673205103150083e-06,1.570796326792424],
00894      *
00895      * [0.5488696249770835,-0.1214996547187207,0.3599720701808493,-3.14159198038469,-3.6732051029273e-06,1.570796326792423],
00896      * [0.5488696249770836,-0.0389996547187214,0.3599720701808496,-3.141591980384691,-3.673205102557476e-06,
00897      * 1.570796326792422],1], "id":1}
00898      *
00899      * @par JSON-RPC 响应示例
00900      * {"id":1,"jsonrpc":"2.0","result":
[[0.5488696249770837,-0.031860179583911546,0.27033259504604207,-3.1415919803846903,-3.67320510285378e-06,1.570796326792423],1]}
00901      * \endchinese
00902      * Based on three points on an arc, calculate the position of the midpoint of the other half of the fitted
circle's arc
00903      *
00904      * @param p1 start point of the arc
00905      * @param p2 middle point of the arc
00906      * @param p3 end point of the arc
00907      * @param mode when mode = 1, need to plan for orientation around arc;
00908      * when mode = 0, do not need to plan for orientation around arc.
00909      * @return position of the midpoint of the other half of the fitted circle's arc and whether the result is
valid.
00910      *
00911      *
00912      * @par Lua 函数原型
00913      * calculateCircleFourthPoint(p1: table, p2: table, p3: table, mode: int)
00914      *
00915      * @par Lua 示例
00916      * center_pose, cal_result =
calculateCircleFourthPoint({0.5488696249770836,-0.1214996547187204,0.2631931199112321,-3.14159198038469,-3.673205103150083e-06,1.5707963

```

```

00917     *
00918     {0.5488696249770835,-0.1214996547187207,0.3599720701808493,-3.14159198038469,-3.6732051029273e-06,1.570796326792423},
00919     *
00920     {0.5488696249770836,-0.0389996547187214,0.3599720701808496,-3.141591980384691,-3.673205102557476e-06,1.570796326792422},1)
00921     *
00922     * @par JSON-RPC request example
00923     * {"jsonrpc":"2.0","method":"Math.calculateCircleFourthPoint","params":
00924     * [[0.5488696249770836,-0.1214996547187204,0.2631931199112321,-3.14159198038469,-3.673205103150083e-06,1.570796326792424],
00925     * [0.5488696249770835,-0.1214996547187207,0.3599720701808493,-3.14159198038469,-3.6732051029273e-06,1.570796326792423],
00926     * [0.5488696249770836,-0.0389996547187214,0.3599720701808496,-3.141591980384691,-3.673205102557476e-06,
00927     * 1.570796326792422],1], "id":1}
00928     *
00929     * @par JSON-RPC response example
00930     * {"id":1,"jsonrpc":"2.0","result":
00931     * [[0.5488696249770837,-0.031860179583911546,0.27033259504604207,-3.1415919803846903,-3.67320510285378e-06,1.570796326792423],1]}
00932     *
00933     */
00934     ResultWithErrno calculateCircleFourthPoint(const std::vector<double> &p1,
00935     const std::vector<double> &p2,
00936     const std::vector<double> &p3,
00937     int mode);
00938 /**
00939     * @ingroup Math
00940     * \chinese
00941     * @brief forceTrans:
00942     * 变换力和力矩的参考坐标系 force_in_b = pose_a_in_b * force_in_a
00943     * @param pose_a_in_b: a 坐标系在 b 坐标系的位姿
00944     * @param force_in_a: 力和力矩在 a 坐标系的描述
00945     * @return force_in_b, 力和力矩在 b 坐标系的描述
00946     * \endchinese
00947     *
00948     * \english
00949     * @brief forceTrans:
00950     * Transform the reference frame of force and torque: force_in_b = pose_a_in_b * force_in_a
00951     * @param pose_a_in_b: pose of frame a in frame b
00952     * @param force_in_a: force and torque described in frame a
00953     * @return Force_in_b, force and torque described in frame b
00954     * \endenglish
00955     */
00956     std::vector<double> forceTrans(const std::vector<double> &pose_a_in_b,
00957     const std::vector<double> &force_in_a);
00958 /**
00959     * @ingroup Math
00960     * \chinese
00961     * @brief 通过距离计算工具坐标系下的位姿增量
00962     * @param distances: N 个距离, N >= 3
00963     * @param position: 距离参考轨迹的保持高度
00964     * @param radius: 传感器中心距离末端 tcp 的等效半径
00965     * @param track_scale: 跟踪比例, 设置范围 (0, 1], 1 表示跟踪更快
00966     * @return 基于工具坐标系的位姿增量
00967     * \endchinese
00968     *
00969     * \english
00970     * @brief Calculate pose increment in tool coordinate system based on sensor data
00971     * @param distances: N distances, N >= 3
00972     * @param position: reference height to maintain from the trajectory
00973     * @param radius: effective radius from sensor center to tool TCP
00974     * @param track_scale: tracking ratio, range (0, 1], 1 means faster tracking
00975     * @return Pose increment in tool coordinate system
00976     * \endenglish
00977     */
00978     std::vector<double> getDeltaPoseBySensorDistance(
00979     const std::vector<double> &distances, double position, double radius,
00980     double track_scale);
00981 /**
00982     * @ingroup Math
00983     * \chinese
00984     * @brief changeFTFrame: 变换力和力矩的参考坐标系
00985     * @param pose_a_in_b: a 坐标系在 b 坐标系的位姿
00986     * @param ft_in_a: 作用在 a 点的力和力矩在 a 坐标系的描述
00987     * @return ft_in_b, 作用在 b 点的力和力矩在 b 坐标系的描述
00988     * \endchinese
00989     *
00990     * \english
00991     * @brief changeFTFrame: Transform the reference frame of force and torque
00992     * @param pose_a_in_b: pose of frame a in frame b
00993     * @param ft_in_a: force and torque applied at point a, described in frame a
00994     * @return ft_in_b, force and torque applied at point b, described in frame b
00995     * \endenglish
00996     */
00997     std::vector<double> deltaPoseTrans(const std::vector<double> &pose_a_in_b,
00998     const std::vector<double> &ft_in_a);
00999

```

```

00999  /**
01000  * @ingroup Math
01001  * \chinese
01002  * @brief addDeltaPose: 计算以给定速度变换单位时间后的位姿
01003  * @param pose_a_in_b: 当前时刻 a 相对于 b 的位姿
01004  * @param v_in_b: 当前时刻 a 坐标系的速度在 b 的描述
01005  * @return pose_in_b, 单位时间后的位姿在 b 的描述
01006  * \endchinese
01007  *
01008  * \english
01009  * @brief addDeltaPose: Calculate the pose after unit time given a velocity
01010  * @param pose_a_in_b: current pose of a relative to b
01011  * @param v_in_b: velocity of frame a described in frame b at current time
01012  * @return pose_in_b, pose after unit time described in frame b
01013  * \endenglish
01014  */
01015  std::vector<double> deltaPoseAdd(const std::vector<double> &pose_a_in_b,
01016                                const std::vector<double> &v_in_b);
01017
01018  /**
01019  * @ingroup Math
01020  * \chinese
01021  * @brief changePoseWithXYRef: 修改 pose_tar 的 x y 轴方向, 尽量与 pose_ref 一致,
01022  * @param pose_tar: 需要修改的目标位姿
01023  * @param pose_ref: 参考位姿
01024  * @return 修改后的位姿, 采用 pose_tar 的 xyz 坐标和 z 轴方向
01025  * \endchinese
01026  *
01027  * \english
01028  * @brief changePoseWithXYRef: Modify the XY axis direction of pose_tar to be as consistent as possible with
pose_ref
01029  * @param pose_tar: target pose to be modified
01030  * @param pose_ref: reference pose
01031  * @return Modified pose, using the xyz coordinates and z axis direction of pose_tar
01032  * \endenglish
01033  */
01034  std::vector<double> changePoseWithXYRef(
01035      const std::vector<double> &pose_tar,
01036      const std::vector<double> &pose_ref);
01037
01038  /**
01039  * @ingroup Math
01040  * \chinese
01041  * @brief homMatrixToPose: 由齐次变换矩阵得到位姿
01042  * @param homMatrix: 4*4 齐次变换矩阵, 输入元素采用横向排列
01043  * @return 对应的位姿
01044  * \endchinese
01045  *
01046  * \english
01047  * @brief homMatrixToPose: Get pose from homogeneous transformation matrix
01048  * @param homMatrix: 4x4 homogeneous transformation matrix, input elements are arranged row-wise
01049  * @return corresponding pose
01050  * \endenglish
01051  */
01052  std::vector<double> homMatrixToPose(const std::vector<double> &homMatrix);
01053
01054  /**
01055  * @ingroup Math
01056  * \chinese
01057  * @brief poseToHomMatrix: 位姿变换得到齐次变换矩阵
01058  * @param pose: 输入的位姿
01059  * @return 输出的齐次变换矩阵, 元素横向排列
01060  * \endchinese
01061  *
01062  * \english
01063  * @brief poseToHomMatrix: Get homogeneous transformation matrix from pose
01064  * @param pose: input pose
01065  * @return output homogeneous transformation matrix, elements arranged row-wise
01066  * \endenglish
01067  */
01068  std::vector<double> poseToHomMatrix(const std::vector<double> &pose);
01069
01070 protected:
01071     void *d_;
01072 };
01073 using MathPtr = std::shared_ptr<Math>;
01074
01075 } // namespace common_interface
01076 } // namespace arcs
01077 #endif

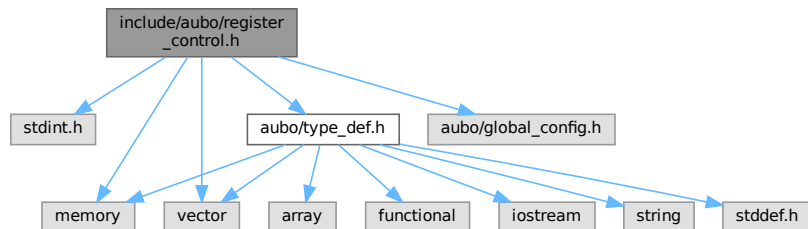
```

12.18 include/aubo/register_control.h 文件参考

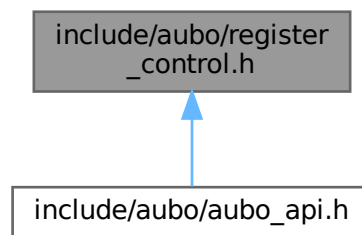
寄存器操作接口, 用于三个模块之间的数据交换功能

```
#include <stdint.h>
#include <memory>
#include <vector>
#include <aubo/type_def.h>
#include <aubo/global_config.h>
```

register_control.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::common_interface::RegisterControl`

命名空间

- namespace `arcs`
- namespace `arcs::common_interface`

类型定义

- using `arcs::common_interface::RegisterControlPtr` = `std::shared_ptr<RegisterControl>`

枚举

- enum `ModbusErrorNum` {
`MB_ERR_NOT_INIT` = -1 , `MB_ERR_DISCONNECTED` = -2 , `MB_ERR_ILLEGAL_FUNCTION`
= 1 , `MB_ERR_ILLEGAL_DATA_ADDRESS` = 2 ,
`MB_ERR_ILLEGAL_DATA_VALUE` = 3 , `MB_ERR_SLAVE_DEVICE_FAILURE` = 4 , `MB_ERR_ACKNO`
= 5 , `MB_ERR_SLAVE_DEVICE_BUSY` = 6 }

12.18.1 详细描述

寄存器操作接口，用于三个模块之间的数据交换功能
在文件 [register_control.h](#) 中定义。

12.18.2 枚举类型说明

ModbusErrorNum

```
enum ModbusErrorNum
```

枚举值

MB_ERR_NOT_INIT	MODBUS unit not initialized
MB_ERR_DISCONNECTED	MODBUS unit disconnected
MB_ERR_ILLEGAL_FUNCTION	The function code received in the query is not an allowable action for the server (or slave).
MB_ERR_ILLEGAL_DATA_ADDRESS	The function code received in the query is not an allowable action for the server (or slave), check that the entered signal address corresponds to the setup of the remote MODBUS server.
MB_ERR_ILLEGAL_DATA_VALUE	A value contained in the query data field is not an allowable value for server (or slave), check that the entered signal value is valid for the specified address on the remote MODBUS server.
MB_ERR_SLAVE_DEVICE_FAILURE	An unrecoverable error occurred while the server (or slave) was attempting to perform the requested action.
MB_ERR_ACKNOWLEDGE	Specialized use in conjunction with programming commands sent to the remote MODBUS unit.
MB_ERR_SLAVE_DEVICE_BUSY	Specialized use in conjunction with programming commands sent to the remote MODBUS unit, the slave (server) is not able to respond now

在文件 [register_control.h](#) 第 14 行定义。

12.19 register_control.h

[浏览该文件的文档.](#)

```

00001 /** @file register_control.h
00002  * @brief 寄存器操作接口，用于三个模块之间的数据交换功能
00003  */
00004 #ifndef AUBO_SDK_REGISTER_CONTROL_INTERFACE_H
00005 #define AUBO_SDK_REGISTER_CONTROL_INTERFACE_H
00006
00007 #include <stdint.h>
00008 #include <memory>
00009 #include <vector>
00010
00011 #include <aubo/type_def.h>
00012 #include <aubo/global_config.h>
00013
00014 enum ModbusErrorNum
00015 {
00016     /** MODBUS unit not initialized
00017     */
00018     MB_ERR_NOT_INIT = -1,
00019
00020     /** MODBUS unit disconnected
00021     */
00022     MB_ERR_DISCONNECTED = -2,
00023

```



```

00024  /** The function code received in the query is not an allowable action for
00025  * the server (or slave).
00026  */
00027  MB_ERR_ILLEGAL_FUNCTION = 1,
00028
00029  /** The function code received in the query is not an allowable action for
00030  * the server (or slave), check that the entered signal address corresponds
00031  * to the setup of the remote MODBUS server.
00032  */
00033  MB_ERR_ILLEGAL_DATA_ADDRESS = 2,
00034
00035  /** A value contained in the query data field is not an allowable value for
00036  * server (or slave), check that the entered signal value is valid for the
00037  * specified address on the remote MODBUS server.
00038  */
00039  MB_ERR_ILLEGAL_DATA_VALUE = 3,
00040
00041  /** An unrecoverable error occurred while the server (or slave) was
00042  * attempting to perform the requested action.
00043  */
00044  MB_ERR_SLAVE_DEVICE_FAILURE = 4,
00045
00046  /** Specialized use in conjunction with programming commands sent to the
00047  * remote MODBUS unit.
00048  */
00049  MB_ERR_ACKNOWLEDGE = 5,
00050
00051  /** Specialized use in conjunction with programming commands sent to the
00052  * remote MODBUS unit, the slave (server) is not able to respond now
00053  */
00054  MB_ERR_SLAVE_DEVICE_BUSY = 6,
00055 };
00056
00057 namespace arcs {
00058 namespace common_interface {
00059
00060 /**
00061  * @defgroup RegisterControl RegisterControl (寄存器操作)
00062  * \-chinese 通用寄存器 \-english General Registers
00063  */
00064 class ARCS_ABI_EXPORT RegisterControl
00065 {
00066 public:
00067     RegisterControl();
00068     virtual ~RegisterControl();
00069
00070 /**
00071  * @ingroup RegisterControl
00072  * \chinese
00073  *
00074  * 从一个输入寄存器中读取布尔值，也可以通过现场总线进行访问。
00075  * 注意，它使用自己的内存空间。
00076  *
00077  * @param address 寄存器的地址 (0:127)
00078  * @return 寄存器中保存的布尔值 (true、false)
00079  *
00080  * @note 布尔输入寄存器的较低范围 [0:63] 保留供 FieldBus/PLC 接口使用。
00081  * 较高范围 [64:127] 无法通过 FieldBus/PLC 接口访问，因为它保留供外部 RTDE 客户端使用。
00082  *
00083  * @par Python 函数原型
00084  * getBoolInput(self: pyaubo_sdk.RegisterControl, arg0: int) -> bool
00085  *
00086  * @par Lua 函数原型
00087  * getBoolInput(address: number) -> boolean
00088  *
00089  * @par Lua 示例
00090  * BoolInput_0 = getBoolInput(0)
00091  *
00092  * @par JSON-RPC 请求示例
00093  * {"jsonrpc": "2.0", "method": "RegisterControl.getBoolInput", "params": [0], "id": 1}
00094  *
00095  * @par JSON-RPC 响应示例
00096  * {"id": 1, "jsonrpc": "2.0", "result": false}
00097  * \endchinese
00098  *
00099  * \english
00100  * Reads the boolean from one of the input registers, which can also be
00101  * accessed by a Field bus. Note, uses its own memory space.
00102  *
00103  * @param address Address of the register (0:127)
00104  * @return Boolean value held by the register (true, false)
00105  *
00106  * @note The lower range of the boolean input registers [0:63] is reserved
00107  * for FieldBus/PLC interface usage. The upper range [64:127] cannot be
00108  * accessed by FieldBus/PLC interfaces, since it is reserved for external
00109  * RTDE clients.
00110  *

```

```

00111     * @par Python interface prototype
00112     * getBoolInput(self: pyaubo_sdk.RegisterControl, arg0: int) -> bool
00113     *
00114     * @par Lua interface prototype
00115     * getBoolInput(address: number) -> boolean
00116     *
00117     * @par Lua example
00118     * BoolInput_0 = getBoolInput(0)
00119     *
00120     * @par JSON-RPC request example
00121     * {"jsonrpc": "2.0", "method": "RegisterControl.getBoolInput", "params": [0], "id": 1}
00122     *
00123     * @par JSON-RPC response example
00124     * {"id": 1, "jsonrpc": "2.0", "result": false}
00125     * \endenglish
00126     */
00127     bool getBoolInput(uint32_t address);
00128
00129     /**
00130     * @ingroup RegisterControl
00131     * \chinese
00132     *
00133     * @param address
00134     * @param value
00135     * @return
00136     *
00137     * @note 只有在实现 RTDE/Modbus Slave/PLC 服务端时使用
00138     *
00139     * @par Python 函数原型
00140     * setBoolInput(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: bool) ->
00141     * int
00142     *
00143     * @par Lua 函数原型
00144     * setBoolInput(address: number, value: boolean) -> nil
00145     *
00146     * @par Lua 示例
00147     * setBoolInput(0)
00148     *
00149     * @par JSON-RPC 请求示例
00150     * {"jsonrpc": "2.0", "method": "RegisterControl.setBoolInput", "params": [0, true], "id": 1}
00151     *
00152     * @par JSON-RPC 响应示例
00153     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00154     *
00155     * \endchinese
00156     * \english
00157     *
00158     * @param address Address of the register (0:127)
00159     * @param value Boolean value to set (true or false)
00160     * @return Returns 0 on success, or an error code
00161     *
00162     * @note Only used when implementing RTDE/Modbus Slave/PLC server
00163     *
00164     * @par Python interface prototype
00165     * setBoolInput(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: bool) ->
00166     * int
00167     *
00168     * @par Lua interface prototype
00169     * setBoolInput(address: number, value: boolean) -> nil
00170     *
00171     * @par Lua example
00172     * setBoolInput(0)
00173     *
00174     * @par JSON-RPC request example
00175     * {"jsonrpc": "2.0", "method": "RegisterControl.setBoolInput", "params": [0, true], "id": 1}
00176     *
00177     * @par JSON-RPC response example
00178     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00179     *
00180     * \endenglish
00181     */
00182     int setBoolInput(uint32_t address, bool value);
00183
00184     /**
00185     * @ingroup RegisterControl
00186     * \chinese
00187     *
00188     * 从一个输入寄存器中读取整数值，也可以通过现场总线进行访问。
00189     * 注意，它使用自己的内存空间。
00190     *
00191     * @param address 寄存器的地址 (0:47)
00192     * @return 寄存器中保存的整数值 [-2,147,483,648 : 2,147,483,647]
00193     *
00194     * @note 整数输入寄存器的较低范围 [0:23] 保留供 FieldBus/PLC 接口使用。
00195     * 较高范围 [24:47] 无法通过 FieldBus/PLC 接口访问，因为它保留供外部 RTDE 客户端使用。
00196     *
00197     * @par Python 函数原型

```

```

00198     * getInt32Input(self: pyaubo_sdk.RegisterControl, arg0: int) -> int
00199     *
00200     * @par Lua 函数原型
00201     * getInt32Input(address: number) -> number
00202     *
00203     * @par Lua 示例
00204     * Int32Input_0 = getInt32Input(0)
00205     *
00206     * @par JSON-RPC 请求示例
00207     * {"jsonrpc": "2.0", "method": "RegisterControl.getInt32Input", "params": [0], "id": 1}
00208     *
00209     * @par JSON-RPC 响应示例
00210     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00211     * \endchinese
00212     *
00213     * \english
00214     * Reads the integer from one of the input registers, which can also be
00215     * accessed by a FieldBus. Note, uses it's own memory space.
00216     *
00217     * @param address Address of the register (0:47)
00218     * @return The value held by the register [-2,147,483,648 : 2,147,483,647]
00219     *
00220     * @note The lower range of the integer input registers [0:23] is reserved
00221     * for FieldBus/PLC interface usage. The upper range [24:47] cannot be
00222     * accessed by FieldBus/PLC interfaces, since it is reserved for external
00223     * RTDE clients.
00224     *
00225     * @par Python interface prototype
00226     * getInt32Input(self: pyaubo_sdk.RegisterControl, arg0: int) -> int
00227     *
00228     * @par Lua interface prototype
00229     * getInt32Input(address: number) -> number
00230     *
00231     * @par Lua example
00232     * Int32Input_0 = getInt32Input(0)
00233     *
00234     * @par JSON-RPC request example
00235     * {"jsonrpc": "2.0", "method": "RegisterControl.getInt32Input", "params": [0], "id": 1}
00236     *
00237     * @par JSON-RPC response example
00238     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00239     * \endchinese
00240     */
00241     int getInt32Input(uint32_t address);
00242
00243     /**
00244     * @ingroup RegisterControl
00245     * @ingroup RegisterControl
00246     * \chinese
00247     *
00248     * @param address 寄存器的地址 (0:47)
00249     * @param value 要设置的整数值
00250     * @return 返回 0 表示成功, 其他为错误码
00251     *
00252     * @note 只有在实现 RTDE/Modbus Slave/PLC 服务端时使用
00253     *
00254     * @par Python 函数原型
00255     * setInt32Input(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: int) ->
00256     * int
00257     *
00258     * @par Lua 函数原型
00259     * setInt32Input(address: number, value: number) -> nil
00260     *
00261     * @par Lua 示例
00262     * setInt32Input(0)
00263     *
00264     * @par JSON-RPC 请求示例
00265     * {"jsonrpc": "2.0", "method": "RegisterControl.setInt32Input", "params": [0,33], "id": 1}
00266     *
00267     * @par JSON-RPC 响应示例
00268     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00269     *
00270     * \endchinese
00271     * \english
00272     *
00273     * @param address Address of the register (0:47)
00274     * @param value Integer value to set
00275     * @return Returns 0 on success, or an error code
00276     *
00277     * @note Only used when implementing RTDE/Modbus Slave/PLC server
00278     *
00279     * @par Python interface prototype
00280     * setInt32Input(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: int) ->
00281     * int
00282     *
00283     * @par Lua interface prototype
00284     * setInt32Input(address: number, value: number) -> nil

```

```

00285     *
00286     * @par Lua example
00287     * setInt32Input(0)
00288     *
00289     * @par JSON-RPC request example
00290     * {"jsonrpc":"2.0","method":"RegisterControl.setInt32Input","params":[0,33],"id":1}
00291     *
00292     * @par JSON-RPC response example
00293     * {"id":1,"jsonrpc":"2.0","result":0}
00294     *
00295     * \endenglish
00296     */
00297 int setInt32Input(uint32_t address, int value);
00298
00299 /**
00300     * @ingroup RegisterControl
00301     * \chinese
00302     * Reads the float from one of the input registers, which can also be
00303     * accessed by a Field bus. Note, uses it' s own memory space.
00304     *
00305     * 从一个输入寄存器中读取浮点数, 也可以通过现场总线进行访问。
00306     * 注意, 它使用自己的内存空间。
00307     *
00308     * @param address Address of the register (0:47)
00309     * 寄存器地址 (0:47)
00310     * @return The value held by the register (float)
00311     * 寄存器中保存的浮点数值
00312     *
00313     * @note The lower range of the float input registers [0:23] is reserved
00314     * for FieldBus/PLC interface usage. The upper range [24:47] cannot be
00315     * accessed by FieldBus/PLC interfaces, since it is reserved for external
00316     * RTDE clients.
00317     * 浮点数输入寄存器的较低范围 [0:23] 保留供现场总线/PLC 接口使用。
00318     * 较高范围 [24:47] 不能通过现场总线/PLC 接口访问, 因为它们是为外部 RTDE 客户端保留的。
00319     *
00320     * @par Python 函数原型
00321     * getFloatInput(self: pyaubo_sdk.RegisterControl, arg0: int) -> float
00322     *
00323     * @par Lua 函数原型
00324     * getFloatInput(address: number) -> number
00325     *
00326     * @par Lua 示例
00327     * FloatInput_0 = getFloatInput(0)
00328     *
00329     * @par JSON-RPC 请求示例
00330     * {"jsonrpc":"2.0","method":"RegisterControl.getFloatInput","params":[0],"id":1}
00331     *
00332     * @par JSON-RPC 响应示例
00333     * {"id":1,"jsonrpc":"2.0","result":0.0}
00334     *
00335     * \endchinese
00336     *
00337     * \english
00338     * Reads the float from one of the input registers, which can also be
00339     * accessed by a Field bus. Note, uses it' s own memory space.
00340     *
00341     * @param address Address of the register (0:47)
00342     * @return The value held by the register (float)
00343     *
00344     * @note The lower range of the float input registers [0:23] is reserved
00345     * for FieldBus/PLC interface usage. The upper range [24:47] cannot be
00346     * accessed by FieldBus/PLC interfaces, since it is reserved for external
00347     * RTDE clients.
00348     *
00349     * @par Python interface prototype
00350     * getFloatInput(self: pyaubo_sdk.RegisterControl, arg0: int) -> float
00351     *
00352     * @par Lua interface prototype
00353     * getFloatInput(address: number) -> number
00354     *
00355     * @par Lua example
00356     * FloatInput_0 = getFloatInput(0)
00357     *
00358     * @par JSON-RPC request example
00359     * {"jsonrpc":"2.0","method":"RegisterControl.getFloatInput","params":[0],"id":1}
00360     *
00361     * @par JSON-RPC response example
00362     * {"id":1,"jsonrpc":"2.0","result":0.0}
00363     */
00364 float getFloatInput(uint32_t address);
00365
00366 /**
00367     * @ingroup RegisterControl
00368     * \chinese
00369     *
00370     * @param address 寄存器的地址 (0:47)
00371     * @param value 要设置的浮点数值

```

```

00372      * @return 返回 0 表示成功, 其他为错误码
00373      *
00374      * @note 只有在实现 RTDE/Modbus Slave/PLC 服务端时使用
00375      *
00376      * @par Python 函数原型
00377      * setFloatInput(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: float)
00378      * -> int
00379      *
00380      * @par Lua 函数原型
00381      * setFloatInput(address: number, value: number) -> nil
00382      *
00383      * @par Lua 示例
00384      * setFloatInput(0, 3.3)
00385      *
00386      * @par JSON-RPC 请求示例
00387      * {"jsonrpc": "2.0", "method": "RegisterControl.setFloatInput", "params": [0, 3.3], "id": 1}
00388      *
00389      * @par JSON-RPC 响应示例
00390      * {"id": 1, "jsonrpc": "2.0", "result": 0}
00391      *
00392      * \endchinese
00393      * \english
00394      *
00395      * @param address Address of the register (0:47)
00396      * @param value Float value to set
00397      * @return Returns 0 on success, or an error code
00398      *
00399      * @note Only used when implementing RTDE/Modbus Slave/PLC server
00400      *
00401      * @par Python interface prototype
00402      * setFloatInput(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: float)
00403      * -> int
00404      *
00405      * @par Lua interface prototype
00406      * setFloatInput(address: number, value: number) -> nil
00407      *
00408      * @par Lua example
00409      * setFloatInput(0, 3.3)
00410      *
00411      * @par JSON-RPC request example
00412      * {"jsonrpc": "2.0", "method": "RegisterControl.setFloatInput", "params": [0, 3.3], "id": 1}
00413      *
00414      * @par JSON-RPC response example
00415      * {"id": 1, "jsonrpc": "2.0", "result": 0}
00416      *
00417      * \endenglish
00418      */
00419 int setFloatInput(uint32_t address, float value);
00420
00421 /**
00422  * @ingroup RegisterControl
00423  * \chinese
00424  *
00425  * 从一个输入寄存器中读取双精度浮点数, 也可以通过现场总线进行访问。
00426  * 注意, 它使用自己的内存空间。
00427  *
00428  * @param address 寄存器的地址 (0:47)
00429  * @return 寄存器中保存的双精度浮点数值
00430  *
00431  * @par Python 函数原型
00432  * getDoubleInput(self: pyaubo_sdk.RegisterControl, arg0: int) -> float
00433  *
00434  * @par Lua 函数原型
00435  * getDoubleInput(address: number) -> number
00436  *
00437  * @par Lua 示例
00438  * DoubleInput_0 = getDoubleInput(0)
00439  *
00440  * @par JSON-RPC 请求示例
00441  * {"jsonrpc": "2.0", "method": "RegisterControl.getDoubleInput", "params": [0], "id": 1}
00442  *
00443  * @par JSON-RPC 响应示例
00444  * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
00445  *
00446  * \endchinese
00447  * \english
00448  *
00449  * Reads the double value from one of the input registers, which can also be
00450  * accessed by a FieldBus. Note, uses its own memory space.
00451  *
00452  * @param address Address of the register (0:47)
00453  * @return The double value held by the register
00454  *
00455  * @par Python interface prototype
00456  * getDoubleInput(self: pyaubo_sdk.RegisterControl, arg0: int) -> float
00457  *
00458  * @par Lua interface prototype

```

```

00459     * getDoubleInput(address: number) -> number
00460     *
00461     * @par Lua example
00462     * DoubleInput_0 = getDoubleInput(0)
00463     *
00464     * @par JSON-RPC request example
00465     * {"jsonrpc": "2.0", "method": "RegisterControl.getDoubleInput", "params": [0], "id": 1}
00466     *
00467     * @par JSON-RPC response example
00468     * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
00469     *
00470     * \endenglish
00471     */
00472 double getDoubleInput(uint32_t address);
00473
00474 /**
00475     * \english
00476     * @param address
00477     * @param value
00478     * @return
00479     *
00480     * @note Only used when implementing RTDE/Modbus Slave/PLC server
00481     *
00482     * @par Python interface prototype
00483     * setDoubleInput(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: float)
00484     * -> int
00485     *
00486     * @par Lua interface prototype
00487     * setDoubleInput(address: number, value: number) -> nil
00488     *
00489     * @par Lua example
00490     * setDoubleInput(0, 3.3)
00491     *
00492     * @par JSON-RPC request example
00493     * {"jsonrpc": "2.0", "method": "RegisterControl.setDoubleInput", "params": [0, 6.6], "id": 1}
00494     *
00495     * @par JSON-RPC response example
00496     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00497     * \endenglish
00498     *
00499     * @ingroup RegisterControl
00500     * \chinese
00501     * @param address
00502     * @param value
00503     * @return
00504     *
00505     * @note 只有在实现 RTDE/Modbus Slave/PLC 服务端时使用
00506     *
00507     * @par Python 函数原型
00508     * setDoubleInput(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: float)
00509     * -> int
00510     *
00511     * @par Lua 函数原型
00512     * setDoubleInput(address: number, value: number) -> nil
00513     *
00514     * @par Lua 示例
00515     * setDoubleInput(0, 3.3)
00516     *
00517     * @par JSON-RPC 请求示例
00518     * {"jsonrpc": "2.0", "method": "RegisterControl.setDoubleInput", "params": [0, 6.6], "id": 1}
00519     *
00520     * @par JSON-RPC 响应示例
00521     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00522     * \endchinese
00523     */
00524 int setDoubleInput(uint32_t address, double value);
00525
00526 /**
00527     * @ingroup RegisterControl
00528     * \chinese
00529     * 从一个输出寄存器中读取布尔值，也可以通过现场总线进行访问。
00530     * 注意，它使用自己的内存空间。
00531     *
00532     * @param address 寄存器地址 (0:127)
00533     * @return 寄存器中保存的布尔值 (true, false)
00534     *
00535     * @note 布尔输出寄存器的较低范围 [0:63] 保留供现场总线/PLC 接口使用。
00536     * 较高范围 [64:127] 不能通过现场总线/PLC 接口访问，因为它们是为外部 RTDE 客户端保留的。
00537     *
00538     * @par Python 函数原型
00539     * getBoolOutput(self: pyaubo_sdk.RegisterControl, arg0: int) -> bool
00540     *
00541     * @par Lua 函数原型
00542     * getBoolOutput(address: number) -> boolean
00543     *
00544     * @par Lua 示例
00545     * BoolOutput_0 = getBoolOutput(0)

```

```

00546      *
00547      * @par JSON-RPC 请求示例
00548      * {"jsonrpc":"2.0","method":"RegisterControl.getBoolOutput","params":[0],"id":1}
00549      *
00550      * @par JSON-RPC 响应示例
00551      * {"id":1,"jsonrpc":"2.0","result":false}
00552      * \endchinese
00553      *
00554      * \english
00555      * Reads the boolean from one of the output registers, which can also be
00556      * accessed by a Field bus.
00557      * Note, uses its own memory space.
00558      *
00559      * @param address Address of the register (0:127)
00560      * @return The boolean value held by the register (true, false)
00561      *
00562      * @note The lower range of the boolean output registers [0:63] is reserved
00563      * for FieldBus/PLC interface usage. The upper range [64:127] cannot be
00564      * accessed by FieldBus/PLC interfaces, since it is reserved for external
00565      * RTDE clients.
00566      *
00567      * @par Python interface prototype
00568      * getBoolOutput(self: pyaubo_sdk.RegisterControl, arg0: int) -> bool
00569      *
00570      * @par Lua interface prototype
00571      * getBoolOutput(address: number) -> boolean
00572      *
00573      * @par Lua example
00574      * BoolOutput_0 = getBoolOutput(0)
00575      *
00576      * @par JSON-RPC request example
00577      * {"jsonrpc":"2.0","method":"RegisterControl.getBoolOutput","params":[0],"id":1}
00578      *
00579      * @par JSON-RPC response example
00580      * {"id":1,"jsonrpc":"2.0","result":false}
00581      * \endenglish
00582      */
00583      bool getBoolOutput(uint32_t address);
00584
00585      /**
00586      * @ingroup RegisterControl
00587      * \chinese
00588      *
00589      * @param address 寄存器地址 (0:127)
00590      * @param value 要设置的布尔值 (true 或 false)
00591      * @return 返回 0 表示成功, 其他为错误码
00592      *
00593      * @par Python 函数原型
00594      * setBoolOutput(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: bool) ->
00595      * int
00596      *
00597      * @par Lua 函数原型
00598      * setBoolOutput(address: number, value: boolean) -> nil
00599      *
00600      * @par Lua 示例
00601      * setBoolOutput(0)
00602      *
00603      * @par JSON-RPC 请求示例
00604      * {"jsonrpc":"2.0","method":"RegisterControl.setBoolOutput","params":[0,false],"id":1}
00605      *
00606      * @par JSON-RPC 响应示例
00607      * {"id":1,"jsonrpc":"2.0","result":0}
00608      *
00609      * \endchinese
00610      * \english
00611      *
00612      * @param address Address of the register (0:127)
00613      * @param value Boolean value to set (true or false)
00614      * @return Returns 0 on success, or an error code
00615      *
00616      * @par Python interface prototype
00617      * setBoolOutput(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: bool) ->
00618      * int
00619      *
00620      * @par Lua interface prototype
00621      * setBoolOutput(address: number, value: boolean) -> nil
00622      *
00623      * @par Lua example
00624      * setBoolOutput(0)
00625      *
00626      * @par JSON-RPC request example
00627      * {"jsonrpc":"2.0","method":"RegisterControl.setBoolOutput","params":[0,false],"id":1}
00628      *
00629      * @par JSON-RPC response example
00630      * {"id":1,"jsonrpc":"2.0","result":0}
00631      *
00632      * \endenglish

```



```

00633     */
00634     int setBoolOutput(uint32_t address, bool value);
00635
00636     /**
00637     * @ingroup RegisterControl
00638     * \chinese
00639     * 从一个输出寄存器中读取整数值，也可以通过现场总线进行访问。
00640     * 注意，它使用自己的内存空间。
00641     *
00642     * @param address 寄存器地址 (0:47)
00643     * @return 寄存器中保存的整数值 (-2,147,483,648 : 2,147,483,647)
00644     *
00645     * @note 整数输出寄存器的较低范围 [0:23] 保留供现场总线/PLC 接口使用。
00646     * 较高范围 [24:47] 不能通过现场总线/PLC 接口访问，因为它们是为外部 RTDE 客户端保留的。
00647     *
00648     * @par Python 函数原型
00649     * getInt32Output(self: pyaubo_sdk.RegisterControl, arg0: int) -> int
00650     *
00651     * @par Lua 函数原型
00652     * getInt32Output(address: number) -> number
00653     *
00654     * @par Lua 示例
00655     * Int32Output_0 = getInt32Output(0)
00656     *
00657     * @par JSON-RPC 请求示例
00658     * {"jsonrpc": "2.0", "method": "RegisterControl.getInt32Output", "params": [0], "id": 1}
00659     *
00660     * @par JSON-RPC 响应示例
00661     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00662     * \endchinese
00663     *
00664     * \english
00665     * Reads the integer from one of the output registers, which can also be
00666     * accessed by a FieldBus. Note, uses its own memory space.
00667     *
00668     * @param address Address of the register (0:47)
00669     * @return The int value held by the register [-2,147,483,648 :
00670     * 2,147,483,647]
00671     *
00672     * @note The lower range of the integer output registers [0:23] is reserved
00673     * for FieldBus/PLC interface usage. The upper range [24:47] cannot be
00674     * accessed by FieldBus/PLC interfaces, since it is reserved for external
00675     * RTDE clients.
00676     *
00677     * @par Python interface prototype
00678     * getInt32Output(self: pyaubo_sdk.RegisterControl, arg0: int) -> int
00679     *
00680     * @par Lua interface prototype
00681     * getInt32Output(address: number) -> number
00682     *
00683     * @par Lua example
00684     * Int32Output_0 = getInt32Output(0)
00685     *
00686     * @par JSON-RPC request example
00687     * {"jsonrpc": "2.0", "method": "RegisterControl.getInt32Output", "params": [0], "id": 1}
00688     *
00689     * @par JSON-RPC response example
00690     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00691     * \endenglish
00692     */
00693     int getInt32Output(uint32_t address);
00694
00695     /**
00696     * @ingroup RegisterControl
00697     * \chinese
00698     *
00699     * @param address 寄存器地址 (0:47)
00700     * @param value 要设置的整数值
00701     * @return 返回 0 表示成功，其他为错误码
00702     *
00703     * @par Python 函数原型
00704     * setInt32Output(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: int) ->
00705     * int
00706     *
00707     * @par Lua 函数原型
00708     * setInt32Output(address: number, value: number) -> nil
00709     *
00710     * @par Lua 示例
00711     * setInt32Output(0, 100)
00712     *
00713     * @par JSON-RPC 请求示例
00714     * {"jsonrpc": "2.0", "method": "RegisterControl.setInt32Output", "params": [0,100], "id": 1}
00715     *
00716     * @par JSON-RPC 响应示例
00717     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00718     *
00719     * \endchinese

```



```

00720     * \english
00721     *
00722     * @param address Address of the register (0:47)
00723     * @param value Integer value to set
00724     * @return Returns 0 on success, or an error code
00725     *
00726     * @par Python interface prototype
00727     * setInt32Output(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: int) ->
00728     * int
00729     *
00730     * @par Lua interface prototype
00731     * setInt32Output(address: number, value: number) -> nil
00732     *
00733     * @par Lua example
00734     * setInt32Output(0, 100)
00735     *
00736     * @par JSON-RPC request example
00737     * {"jsonrpc": "2.0", "method": "RegisterControl.setInt32Output", "params": [0, 100], "id": 1}
00738     *
00739     * @par JSON-RPC response example
00740     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00741     *
00742     * \endenglish
00743     */
00744     int setInt32Output(uint32_t address, int value);
00745
00746     /**
00747     * @ingroup RegisterControl
00748     * \chinese
00749     * 从一个输出寄存器中读取浮点数，也可以通过现场总线进行访问。
00750     * 注意，它使用自己的内存空间。
00751     *
00752     * @param address 寄存器地址 (0:47)
00753     * @return 寄存器中保存的浮点数值 (float)
00754     *
00755     * @note 浮点数输出寄存器的较低范围 [0:23] 保留供现场总线/PLC 接口使用。
00756     * 较高范围 [24:47] 不能通过现场总线/PLC 接口访问，因为它们是为外部 RTDE 客户端保留的。
00757     *
00758     * @par Python 函数原型
00759     * getFloatOutput(self: pyaubo_sdk.RegisterControl, arg0: int) -> float
00760     *
00761     * @par Lua 函数原型
00762     * getFloatOutput(address: number) -> number
00763     *
00764     * @par Lua 示例
00765     * FloatOutput_0 = getFloatOutput(0)
00766     *
00767     * @par JSON-RPC 请求示例
00768     * {"jsonrpc": "2.0", "method": "RegisterControl.getFloatOutput", "params": [0], "id": 1}
00769     *
00770     * @par JSON-RPC 响应示例
00771     * {"id": 1, "jsonrpc": "2.0", "result": 3.3}
00772     * \endchinese
00773     *
00774     * \english
00775     * Reads the float from one of the output registers, which can also be
00776     * accessed by a FieldBus. Note, uses its own memory space.
00777     *
00778     * @param address Address of the register (0:47)
00779     * @return The value held by the register (float)
00780     *
00781     * @note The lower range of the float output registers [0:23] is reserved
00782     * for FieldBus/PLC interface usage. The upper range [24:47] cannot be
00783     * accessed by FieldBus/PLC interfaces, since it is reserved for external
00784     * RTDE clients.
00785     *
00786     * @par Python interface prototype
00787     * getFloatOutput(self: pyaubo_sdk.RegisterControl, arg0: int) -> float
00788     *
00789     * @par Lua interface prototype
00790     * getFloatOutput(address: number) -> number
00791     *
00792     * @par Lua example
00793     * FloatOutput_0 = getFloatOutput(0)
00794     *
00795     * @par JSON-RPC request example
00796     * {"jsonrpc": "2.0", "method": "RegisterControl.getFloatOutput", "params": [0], "id": 1}
00797     *
00798     * @par JSON-RPC response example
00799     * {"id": 1, "jsonrpc": "2.0", "result": 3.3}
00800     * \endenglish
00801     */
00802     float getFloatOutput(uint32_t address);
00803
00804     /**
00805     * @ingroup RegisterControl
00806     * \chinese

```

```

00807      *
00808      * @param address 寄存器地址 (0:47)
00809      * @param value 要设置的浮点数值
00810      * @return 返回 0 表示成功, 其他为错误码
00811      *
00812      * @par Python 函数原型
00813      * setFloatOutput(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: float)
00814      * -> int
00815      *
00816      * @par Lua 函数原型
00817      * setFloatOutput(address: number, value: number) -> nil
00818      *
00819      * @par Lua 示例
00820      * setFloatOutput(0,5.5)
00821      *
00822      * @par JSON-RPC 请求示例
00823      * {"jsonrpc":"2.0","method":"RegisterControl.setFloatOutput","params":[0,5.5],"id":1}
00824      *
00825      * @par JSON-RPC 响应示例
00826      * {"id":1,"jsonrpc":"2.0","result":0}
00827      *
00828      * \endchinese
00829      * \english
00830      *
00831      * @param address Address of the register (0:47)
00832      * @param value Float value to set
00833      * @return Returns 0 on success, or an error code
00834      *
00835      * @par Python interface prototype
00836      * setFloatOutput(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: float)
00837      * -> int
00838      *
00839      * @par Lua interface prototype
00840      * setFloatOutput(address: number, value: number) -> nil
00841      *
00842      * @par Lua example
00843      * setFloatOutput(0,5.5)
00844      *
00845      * @par JSON-RPC request example
00846      * {"jsonrpc":"2.0","method":"RegisterControl.setFloatOutput","params":[0,5.5],"id":1}
00847      *
00848      * @par JSON-RPC response example
00849      * {"id":1,"jsonrpc":"2.0","result":0}
00850      *
00851      * \endenglish
00852      */
00853      int setFloatOutput(uint32_t address, float value);
00854
00855      /**
00856      * @ingroup RegisterControl
00857      * \chinese
00858      *
00859      * 从一个输出寄存器中读取双精度浮点数。
00860      *
00861      * @param address 寄存器地址
00862      * @return 寄存器中保存的双精度浮点数值
00863      *
00864      * @par Python 函数原型
00865      * getDoubleOutput(self: pyaubo_sdk.RegisterControl, arg0: int) -> float
00866      *
00867      * @par Lua 函数原型
00868      * getDoubleOutput(address: number) -> number
00869      *
00870      * @par Lua 示例
00871      * DoubleOutput_0 = getDoubleOutput(0)
00872      *
00873      * @par JSON-RPC 请求示例
00874      * {"jsonrpc":"2.0","method":"RegisterControl.getDoubleOutput","params":[0],"id":1}
00875      *
00876      * @par JSON-RPC 响应示例
00877      * {"id":1,"jsonrpc":"2.0","result":0.0}
00878      *
00879      * \endchinese
00880      * \english
00881      *
00882      * Reads the double value from one of the output registers.
00883      *
00884      * @param address Address of the register
00885      * @return The double value held by the register
00886      *
00887      * @par Python interface prototype
00888      * getDoubleOutput(self: pyaubo_sdk.RegisterControl, arg0: int) -> float
00889      *
00890      * @par Lua interface prototype
00891      * getDoubleOutput(address: number) -> number
00892      *
00893      * @par Lua example

```

```

00894     * DoubleOutput_0 = getDoubleOutput(0)
00895     *
00896     * @par JSON-RPC request example
00897     * {"jsonrpc": "2.0", "method": "RegisterControl.getDoubleOutput", "params": [0], "id": 1}
00898     *
00899     * @par JSON-RPC response example
00900     * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
00901     *
00902     * \endenglish
00903     */
00904 double getDoubleOutput(uint32_t address);
00905
00906 /**
00907  * @ingroup RegisterControl
00908  * \chinese
00909  *
00910  * @param address 寄存器地址
00911  * @param value 要设置的双精度浮点数值
00912  * @return 返回 0 表示成功, 其他为错误码
00913  *
00914  * @par Python 函数原型
00915  * setDoubleOutput(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: float)
00916  * -> int
00917  *
00918  * @par Lua 函数原型
00919  * setDoubleOutput(address: number, value: number) -> nil
00920  *
00921  * @par Lua 示例
00922  * setDoubleOutput(0,4.4)
00923  *
00924  * @par JSON-RPC 请求示例
00925  * {"jsonrpc": "2.0", "method": "RegisterControl.setDoubleOutput", "params": [0,4.4], "id": 1}
00926  *
00927  * @par JSON-RPC 响应示例
00928  * {"id": 1, "jsonrpc": "2.0", "result": 0}
00929  *
00930  * \endchinese
00931  * \english
00932  *
00933  * @param address Address of the register
00934  * @param value Double value to set
00935  * @return Returns 0 on success, or an error code
00936  *
00937  * @par Python interface prototype
00938  * setDoubleOutput(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: float)
00939  * -> int
00940  *
00941  * @par Lua interface prototype
00942  * setDoubleOutput(address: number, value: number) -> nil
00943  *
00944  * @par Lua example
00945  * setDoubleOutput(0,4.4)
00946  *
00947  * @par JSON-RPC request example
00948  * {"jsonrpc": "2.0", "method": "RegisterControl.setDoubleOutput", "params": [0,4.4], "id": 1}
00949  *
00950  * @par JSON-RPC response example
00951  * {"id": 1, "jsonrpc": "2.0", "result": 0}
00952  *
00953  * \endenglish
00954  */
00955 int setDoubleOutput(uint32_t address, double value);
00956
00957 /**
00958  * @ingroup RegisterControl
00959  * \chinese
00960  * 用于 Modbus Slave
00961  *
00962  * @param address
00963  * @return
00964  *
00965  * @par Python 函数原型
00966  * getInt16Register(self: pyaubo_sdk.RegisterControl, arg0: int) -> int
00967  *
00968  * @par Lua 函数原型
00969  * getInt16Register(address: number) -> number
00970  *
00971  * @par Lua 示例
00972  * Int16Register_0 = getInt16Register(0)
00973  *
00974  * @par JSON-RPC 请求示例
00975  * {"jsonrpc": "2.0", "method": "RegisterControl.getInt16Register", "params": [0], "id": 1}
00976  *
00977  * @par JSON-RPC 响应示例
00978  * {"id": 1, "jsonrpc": "2.0", "result": 0}
00979  *
00980  * \endchinese

```

```

00981     * \english
00982     * Used for Modbus Slave
00983     *
00984     * @param address
00985     * @return
00986     *
00987     * @par Python interface prototype
00988     * getInt16Register(self: pyaubo_sdk.RegisterControl, arg0: int) -> int
00989     *
00990     * @par Lua interface prototype
00991     * getInt16Register(address: number) -> number
00992     *
00993     * @par Lua example
00994     * Int16Register_0 = getInt16Register(0)
00995     *
00996     * @par JSON-RPC request example
00997     * {"jsonrpc": "2.0", "method": "RegisterControl.getInt16Register", "params": [0], "id": 1}
00998     *
00999     * @par JSON-RPC response example
01000     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01001     *
01002     * \endenglish
01003     */
01004     int16_t getInt16Register(uint32_t address);
01005
01006     /**
01007     * @ingroup RegisterControl
01008     * \chinese
01009     *
01010     * @param address 寄存器地址
01011     * @param value 要设置的值
01012     * @return 返回 0 表示成功, 其他为错误码
01013     *
01014     * @par Python 函数原型
01015     * setInt16Register(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: int)
01016     * -> int
01017     *
01018     * @par Lua 函数原型
01019     * setInt16Register(address: number, value: number) -> nil
01020     *
01021     * @par Lua 示例
01022     * setInt16Register(0,4.4)
01023     *
01024     * @par JSON-RPC 请求示例
01025     * {"jsonrpc": "2.0", "method": "RegisterControl.setInt16Register", "params": [0,0], "id": 1}
01026     *
01027     * @par JSON-RPC 响应示例
01028     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01029     *
01030     * \endchinese
01031     * \english
01032     *
01033     * @param address Register address
01034     * @param value Value to set
01035     * @return Returns 0 on success, otherwise error code
01036     *
01037     * @par Python interface prototype
01038     * setInt16Register(self: pyaubo_sdk.RegisterControl, arg0: int, arg1: int)
01039     * -> int
01040     *
01041     * @par Lua interface prototype
01042     * setInt16Register(address: number, value: number) -> nil
01043     *
01044     * @par Lua example
01045     * setInt16Register(0,4.4)
01046     *
01047     * @par JSON-RPC request example
01048     * {"jsonrpc": "2.0", "method": "RegisterControl.setInt16Register", "params": [0,0], "id": 1}
01049     *
01050     * @par JSON-RPC response example
01051     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01052     *
01053     * \endenglish
01054     */
01055     int setInt16Register(uint32_t address, int16_t value);
01056
01057     /**
01058     * @ingroup RegisterControl
01059     * \chinese
01060     * 获取 Int16 寄存器的某个 bit 的状态
01061     *
01062     * @param address: Int16 寄存器地址
01063     * @param bit_offset: 寄存器中的 bit 偏移, 0 ~ 15
01064     * @return 指定 bit 的状态, true 表示 1, false 表示 0 或参数无效
01065     *
01066     * @par Python 函数原型
01067     * getInt16RegisterBit(self: pyaubo_sdk.RegisterControl, address: int,

```

```

01068     * bit_offset: int) -> bool
01069     *
01070     * @par Lua 函数原型
01071     * getInt16RegisterBit(address: number, bit_offset: number) -> boolean
01072     *
01073     * @par JSON-RPC 请求示例
01074     * {"jsonrpc": "2.0", "method": "RegisterControl.getInt16RegisterBit", "params": [1,
01075     * 5], "id": 1}
01076     *
01077     * @par JSON-RPC 响应示例
01078     * {"id": 1, "jsonrpc": "2.0", "result": true}
01079     *
01080     * \endchinese
01081     * \english
01082     * Get the status of a specific bit in an Int16 register
01083     *
01084     * @param address Int16 register address
01085     * @param bit_offset Bit offset in the register, 0 ~ 15
01086     * @return Status of the specified bit, true for 1, false for 0 or invalid
01087     * parameters
01088     *
01089     * @par Python interface prototype
01090     * getInt16RegisterBit(self: pyaubo_sdk.RegisterControl, address: int,
01091     * bit_offset: int) -> bool
01092     *
01093     * @par Lua interface prototype
01094     * getInt16RegisterBit(address: number, bit_offset: number) -> boolean
01095     *
01096     * @par JSON-RPC request example
01097     * {"jsonrpc": "2.0", "method": "RegisterControl.getInt16RegisterBit", "params": [1,
01098     * 5], "id": 1}
01099     *
01100     * @par JSON-RPC response example
01101     * {"id": 1, "jsonrpc": "2.0", "result": true}
01102     *
01103     * \endenglish
01104     */
01105 bool getInt16RegisterBit(uint32_t address, uint8_t bit_offset);
01106
01107 /**
01108     * @ingroup RegisterControl
01109     * \chinese
01110     * 设置 Int16 寄存器的某个 bit 的状态
01111     *
01112     * @param address: Int16 寄存器地址
01113     * @param bit_offset: 寄存器中的 bit 偏移, 0 ~ 15
01114     * @param value: 要设置的值, true 表示置 1, false 表示清 0
01115     * @return 成功返回 0, 失败返回错误码
01116     *
01117     * @par Python 函数原型
01118     * setInt16RegisterBit(self: pyaubo_sdk.RegisterControl, address: int,
01119     * bit_offset: int, value: bool) -> int
01120     *
01121     * @par Lua 函数原型
01122     * setInt16RegisterBit(address: number, bit_offset: number, value: boolean)
01123     * -> number
01124     *
01125     * @par JSON-RPC 请求示例
01126     * {"jsonrpc": "2.0", "method": "RegisterControl.setInt16RegisterBit", "params": [1,
01127     * 5, true], "id": 1}
01128     *
01129     * @par JSON-RPC 响应示例
01130     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01131     *
01132     * \endchinese
01133     * \english
01134     * Set the status of a specific bit in an Int16 register
01135     *
01136     * @param address Int16 register address
01137     * @param bit_offset Bit offset in the register, 0 ~ 15
01138     * @param value Value to set, true to set to 1, false to clear to 0
01139     * @return Returns 0 on success, error code on failure
01140     *
01141     * @par Python interface prototype
01142     * setInt16RegisterBit(self: pyaubo_sdk.RegisterControl, address: int,
01143     * bit_offset: int, value: bool) -> int
01144     *
01145     * @par Lua interface prototype
01146     * setInt16RegisterBit(address: number, bit_offset: number, value: boolean)
01147     * -> number
01148     *
01149     * @par JSON-RPC request example
01150     * {"jsonrpc": "2.0", "method": "RegisterControl.setInt16RegisterBit", "params": [1,
01151     * 5, true], "id": 1}
01152     *
01153     * @par JSON-RPC response example
01154     * {"id": 1, "jsonrpc": "2.0", "result": 0}

```

```

01155     *
01156     * \endenglish
01157     */
01158     int setInt16RegisterBit(uint32_t address, uint8_t bit_offset, bool value);
01159
01160     /**
01161     * @ingroup RegisterControl
01162     * \chinese
01163     * 具名变量是否存在
01164     *
01165     * @param key 变量名
01166     * @return
01167     *
01168     * @par Lua 函数原型
01169     * hasNamedVariable(key: string) -> boolean
01170     *
01171     * @par Lua 示例
01172     * NamedVariable = hasNamedVariable("custom")
01173     *
01174     * @par JSON-RPC 请求示例
01175     * {"jsonrpc": "2.0", "method": "RegisterControl.hasNamedVariable", "params": ["custom"], "id": 1}
01176     *
01177     * @par JSON-RPC 响应示例
01178     * {"id": 1, "jsonrpc": "2.0", "result": false}
01179     *
01180     * \endchinese
01181     * \english
01182     * Whether the named variable exists
01183     *
01184     * @param key Variable name
01185     * @return
01186     *
01187     * @par Lua interface prototype
01188     * hasNamedVariable(key: string) -> boolean
01189     *
01190     * @par Lua example
01191     * NamedVariable = hasNamedVariable("custom")
01192     *
01193     * @par JSON-RPC request example
01194     * {"jsonrpc": "2.0", "method": "RegisterControl.hasNamedVariable", "params": ["custom"], "id": 1}
01195     *
01196     * @par JSON-RPC response example
01197     * {"id": 1, "jsonrpc": "2.0", "result": false}
01198     *
01199     * \endenglish
01200     */
01201     bool hasNamedVariable(const std::string &key);
01202
01203     /**
01204     * @ingroup RegisterControl
01205     * \chinese
01206     * 获取具名变量的类型
01207     *
01208     * @param key
01209     * @return
01210     *
01211     * @par Lua 函数原型
01212     * getNamedVariableType(key: string) -> string
01213     *
01214     * @par Lua 示例
01215     * NamedVariableType = getNamedVariableType("custom")
01216     *
01217     * @par JSON-RPC 请求示例
01218     * {"jsonrpc": "2.0", "method": "RegisterControl.getNamedVariableType", "params": ["custom"], "id": 1}
01219     *
01220     * @par JSON-RPC 响应示例
01221     * {"id": 1, "jsonrpc": "2.0", "result": "NONE"}
01222     *
01223     * \endchinese
01224     * \english
01225     * Get the type of a named variable
01226     *
01227     * @param key
01228     * @return
01229     *
01230     * @par Lua interface prototype
01231     * getNamedVariableType(key: string) -> string
01232     *
01233     * @par Lua example
01234     * NamedVariableType = getNamedVariableType("custom")
01235     *
01236     * @par JSON-RPC request example
01237     * {"jsonrpc": "2.0", "method": "RegisterControl.getNamedVariableType", "params": ["custom"], "id": 1}
01238     *
01239     * @par JSON-RPC response example
01240     * {"id": 1, "jsonrpc": "2.0", "result": "NONE"}
01241     *

```

```

01242     * \endenglish
01243     */
01244     std::string getNamedVariableType(const std::string &key);
01245
01246 /**
01247  * @ingroup RegisterControl
01248  * \chinese
01249  * 具名变量是否更新
01250  *
01251  * @param key
01252  * @param since
01253  * @return
01254  *
01255  * @par Python 函数原型
01256  * variableUpdated(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: int)
01257  * -> bool
01258  *
01259  * @par Lua 函数原型
01260  * variableUpdated(key: string, since: number) -> boolean
01261  *
01262  * @par Lua 示例
01263  * Variable_Updated = variableUpdated("custom" , 0)
01264  *
01265  * \endchinese
01266  *
01267  * \english
01268  * Whether the named variable has been updated
01269  *
01270  * @param key
01271  * @param since
01272  * @return
01273  *
01274  * @par Python interface prototype
01275  * variableUpdated(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: int)
01276  * -> bool
01277  *
01278  * @par Lua interface prototype
01279  * variableUpdated(key: string, since: number) -> boolean
01280  *
01281  * @par Lua example
01282  * Variable_Updated = variableUpdated("custom" , 0)
01283  *
01284  * \endchinese
01285  */
01286     bool variableUpdated(const std::string &key, uint64_t since);
01287
01288 /**
01289  * @ingroup RegisterControl
01290  * \chinese
01291  * 获取变量值
01292  *
01293  * @param key
01294  * @param default_value
01295  * @return
01296  *
01297  * @par Python 函数原型
01298  * getBool(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: bool) -> bool
01299  *
01300  * @par Lua 函数原型
01301  * getBool(key: string, default_value: boolean) -> boolean
01302  *
01303  * @par Lua 示例
01304  * Bool_var = getBool("custom",false)
01305  *
01306  * @par JSON-RPC 请求示例
01307  * {"jsonrpc":"2.0","method":"RegisterControl.getBool","params":["custom",false],"id":1}
01308  *
01309  * @par JSON-RPC 响应示例
01310  * {"id":1,"jsonrpc":"2.0","result":true}
01311  *
01312  * \endchinese
01313  * \english
01314  * Get variable value
01315  *
01316  * @param key
01317  * @param default_value
01318  * @return
01319  *
01320  * @par Python interface prototype
01321  * getBool(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: bool) -> bool
01322  *
01323  * @par Lua interface prototype
01324  * getBool(key: string, default_value: boolean) -> boolean
01325  *
01326  * @par Lua example
01327  * Bool_var = getBool("custom",false)
01328  *

```

```

01329     * @par JSON-RPC request example
01330     * {"jsonrpc": "2.0", "method": "RegisterControl.getBool", "params": ["custom", false], "id": 1}
01331     *
01332     * @par JSON-RPC response example
01333     * {"id": 1, "jsonrpc": "2.0", "result": true}
01334     *
01335     * \endenglish
01336     */
01337     bool getBool(const std::string &key, bool default_value);
01338
01339     /**
01340     * @ingroup RegisterControl
01341     * \chinese
01342     * 设置/更新变量值
01343     *
01344     * @param key
01345     * @param value
01346     * @return
01347     *
01348     * @par Python 函数原型
01349     * setBool(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: bool) -> int
01350     *
01351     * @par Lua 函数原型
01352     * setBool(key: string, value: boolean) -> nil
01353     *
01354     * @par Lua 示例
01355     * setBool("custom", true)
01356     *
01357     * @par JSON-RPC 请求示例
01358     * {"jsonrpc": "2.0", "method": "RegisterControl.setBool", "params": ["custom", true], "id": 1}
01359     *
01360     * @par JSON-RPC 响应示例
01361     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01362     *
01363     * \endchinese
01364     * \english
01365     * Set or update the variable value
01366     *
01367     * @param key
01368     * @param value
01369     * @return Returns 0 on success, otherwise error code
01370     *
01371     * @par Python interface prototype
01372     * setBool(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: bool) -> int
01373     *
01374     * @par Lua interface prototype
01375     * setBool(key: string, value: boolean) -> nil
01376     *
01377     * @par Lua example
01378     * setBool("custom", true)
01379     *
01380     * @par JSON-RPC request example
01381     * {"jsonrpc": "2.0", "method": "RegisterControl.setBool", "params": ["custom", true], "id": 1}
01382     *
01383     * @par JSON-RPC response example
01384     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01385     *
01386     * \endenglish
01387     */
01388     int setBool(const std::string &key, bool value);
01389
01390     /**
01391     * @ingroup RegisterControl
01392     * \chinese
01393     * 获取变量值
01394     *
01395     * @param key
01396     * @param default_value
01397     * @return
01398     *
01399     * @par Python 函数原型
01400     * getVecChar(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: List[str])
01401     * -> List[str]
01402     *
01403     * @par Lua 函数原型
01404     * getVecChar(key: string, default_value: table) -> table
01405     *
01406     * @par Lua 示例
01407     * VecChar = getVecChar("custom", {})
01408     *
01409     * @par JSON-RPC 请求示例
01410     * {"jsonrpc": "2.0", "method": "RegisterControl.getVecChar", "params": ["custom", []], "id": 1}
01411     *
01412     * @par JSON-RPC 响应示例
01413     * {"id": 1, "jsonrpc": "2.0", "result": [0, 1, 0]}
01414     *
01415     * \endchinese

```



```

01416     * \english
01417     * Get variable value
01418     *
01419     * @param key
01420     * @param default_value
01421     * @return
01422     *
01423     * @par Python interface prototype
01424     * getVecChar(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: List[str])
01425     * -> List[str]
01426     *
01427     * @par Lua interface prototype
01428     * getVecChar(key: string, default_value: table) -> table
01429     *
01430     * @par Lua example
01431     * VecChar = getVecChar("custom",{})
01432     *
01433     * @par JSON-RPC request example
01434     * {"jsonrpc":"2.0","method":"RegisterControl.getVecChar","params":["custom",[]],"id":1}
01435     *
01436     * @par JSON-RPC response example
01437     * {"id":1,"jsonrpc":"2.0","result":[0,1,0]}
01438     *
01439     * \endenglish
01440     */
01441 std::vector<char> getVecChar(const std::string &key,
01442                             const std::vector<char> &default_value);
01443
01444 /**
01445  * @ingroup RegisterControl
01446  * \chinese
01447  * 设置/更新变量值
01448  *
01449  * @param key
01450  * @param value
01451  * @return
01452  *
01453  * @par Python 函数原型
01454  * setVecChar(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: List[str])
01455  * -> int
01456  *
01457  * @par Lua 函数原型
01458  * setVecChar(key: string, value: table) -> nil
01459  *
01460  * @par Lua 示例
01461  * setVecChar("custom",{0,1,0})
01462  *
01463  * @par JSON-RPC 请求示例
01464  * {"jsonrpc":"2.0","method":"RegisterControl.setVecChar","params":["custom",[0,1,0]],"id":1}
01465  *
01466  * @par JSON-RPC 响应示例
01467  * {"id":1,"jsonrpc":"2.0","result":0}
01468  *
01469  * \endchinese
01470  * \english
01471  * Set or update the variable value
01472  *
01473  * @param key
01474  * @param value
01475  * @return Returns 0 on success, otherwise error code
01476  *
01477  * @par Python interface prototype
01478  * setVecChar(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: List[str])
01479  * -> int
01480  *
01481  * @par Lua interface prototype
01482  * setVecChar(key: string, value: table) -> nil
01483  *
01484  * @par Lua example
01485  * setVecChar("custom",{0,1,0})
01486  *
01487  * @par JSON-RPC request example
01488  * {"jsonrpc":"2.0","method":"RegisterControl.setVecChar","params":["custom",[0,1,0]],"id":1}
01489  *
01490  * @par JSON-RPC response example
01491  * {"id":1,"jsonrpc":"2.0","result":0}
01492  *
01493  * \endenglish
01494  */
01495 int setVecChar(const std::string &key, const std::vector<char> &value);
01496
01497 /**
01498  * @ingroup RegisterControl
01499  * \chinese
01500  * 获取变量值
01501  *
01502  * @param key

```

```

01503     * @param default_value
01504     * @return
01505     *
01506     * @par Python 函数原型
01507     * getInt32(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: int) -> int
01508     *
01509     * @par Lua 函数原型
01510     * getInt32(key: string, default_value: number) -> number
01511     *
01512     * @par Lua 示例
01513     * Int32 = getInt32("custom",0)
01514     *
01515     * @par JSON-RPC 请求示例
01516     * {"jsonrpc":"2.0","method":"RegisterControl.getInt32","params":["custom",0],"id":1}
01517     *
01518     * @par JSON-RPC 响应示例
01519     * {"id":1,"jsonrpc":"2.0","result":6}
01520     *
01521     * \endchinese
01522     * \english
01523     * Get variable value
01524     *
01525     * @param key
01526     * @param default_value
01527     * @return
01528     *
01529     * @par Python interface prototype
01530     * getInt32(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: int) -> int
01531     *
01532     * @par Lua interface prototype
01533     * getInt32(key: string, default_value: number) -> number
01534     *
01535     * @par Lua example
01536     * Int32 = getInt32("custom",0)
01537     *
01538     * @par JSON-RPC request example
01539     * {"jsonrpc":"2.0","method":"RegisterControl.getInt32","params":["custom",0],"id":1}
01540     *
01541     * @par JSON-RPC response example
01542     * {"id":1,"jsonrpc":"2.0","result":6}
01543     *
01544     * \endenglish
01545     */
01546     int getInt32(const std::string &key, int default_value);
01547
01548     /**
01549     * @ingroup RegisterControl
01550     * \chinese
01551     * 设置/更新变量值
01552     *
01553     * @param key
01554     * @param value
01555     * @return
01556     *
01557     * @par Python 函数原型
01558     * setInt32(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: int) -> int
01559     *
01560     * @par Lua 函数原型
01561     * setInt32(key: string, value: number) -> nil
01562     *
01563     * @par Lua 示例
01564     * setInt32("custom",6)
01565     *
01566     * @par JSON-RPC 请求示例
01567     * {"jsonrpc":"2.0","method":"RegisterControl.setInt32","params":["custom",6],"id":1}
01568     *
01569     * @par JSON-RPC 响应示例
01570     * {"id":1,"jsonrpc":"2.0","result":0}
01571     *
01572     * \endchinese
01573     * \english
01574     * Set or update the variable value
01575     *
01576     * @param key
01577     * @param value
01578     * @return
01579     *
01580     * @par Python interface prototype
01581     * setInt32(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: int) -> int
01582     *
01583     * @par Lua interface prototype
01584     * setInt32(key: string, value: number) -> nil
01585     *
01586     * @par Lua example
01587     * setInt32("custom",6)
01588     *
01589     * @par JSON-RPC request example

```

```

01590     * {"jsonrpc": "2.0", "method": "RegisterControl.setInt32", "params": ["custom", 6], "id": 1}
01591     *
01592     * @par JSON-RPC response example
01593     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01594     *
01595     * \endenglish
01596     */
01597 int setInt32(const std::string &key, int value);
01598
01599 /**
01600  * @ingroup RegisterControl
01601  * \chinese
01602  * 获取变量值
01603  *
01604  * @param key
01605  * @param default_value
01606  * @return
01607  *
01608  * @par Python 函数原型
01609  * getVecInt32(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: List[int])
01610  * -> List[int]
01611  *
01612  * @par Lua 函数原型
01613  * getVecInt32(key: string, default_value: table) -> table
01614  *
01615  * @par Lua 示例
01616  * VecInt32 = getVecInt32("custom", {})
01617  *
01618  * @par JSON-RPC 请求示例
01619  * {"jsonrpc": "2.0", "method": "RegisterControl.getVecInt32", "params": ["custom", []], "id": 1}
01620  *
01621  * @par JSON-RPC 响应示例
01622  * {"id": 1, "jsonrpc": "2.0", "result": [1, 2, 3, 4]}
01623  *
01624  * \endchinese
01625  * \english
01626  * Get variable value
01627  *
01628  * @param key
01629  * @param default_value
01630  * @return
01631  *
01632  * @par Python interface prototype
01633  * getVecInt32(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: List[int])
01634  * -> List[int]
01635  *
01636  * @par Lua interface prototype
01637  * getVecInt32(key: string, default_value: table) -> table
01638  *
01639  * @par Lua example
01640  * VecInt32 = getVecInt32("custom", {})
01641  *
01642  * @par JSON-RPC request example
01643  * {"jsonrpc": "2.0", "method": "RegisterControl.getVecInt32", "params": ["custom", []], "id": 1}
01644  *
01645  * @par JSON-RPC response example
01646  * {"id": 1, "jsonrpc": "2.0", "result": [1, 2, 3, 4]}
01647  *
01648  * \endenglish
01649  */
01650 std::vector<int32_t> getVecInt32(const std::string &key,
01651                                const std::vector<int32_t> &default_value);
01652
01653 /**
01654  * @ingroup RegisterControl
01655  * \chinese
01656  * 设置/更新变量值
01657  *
01658  * @param key
01659  * @param value
01660  * @return
01661  *
01662  * @par Python 函数原型
01663  * setVecInt32(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: List[int])
01664  * -> int
01665  *
01666  * @par Lua 函数原型
01667  * setVecInt32(key: string, value: table) -> nil
01668  *
01669  * @par Lua 示例
01670  * setVecInt32("custom", {1, 2, 3, 4})
01671  *
01672  * @par JSON-RPC 请求示例
01673  * {"jsonrpc": "2.0", "method": "RegisterControl.setVecInt32", "params": ["custom", [1, 2, 3, 4]], "id": 1}
01674  *
01675  * @par JSON-RPC 响应示例
01676  * {"id": 1, "jsonrpc": "2.0", "result": 0}

```

```

01677      *
01678      * \endchinese
01679      * \english
01680      * Set or update the variable value
01681      *
01682      * @param key
01683      * @param value
01684      * @return
01685      *
01686      * @par Python interface prototype
01687      * setVecInt32(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: List[int])
01688      * -> int
01689      *
01690      * @par Lua interface prototype
01691      * setVecInt32(key: string, value: table) -> nil
01692      *
01693      * @par Lua example
01694      * setVecInt32("custom",{1,2,3,4})
01695      *
01696      * @par JSON-RPC request example
01697      * {"jsonrpc":"2.0","method":"RegisterControl.setVecInt32","params":["custom",[1,2,3,4]],"id":1}
01698      *
01699      * @par JSON-RPC response example
01700      * {"id":1,"jsonrpc":"2.0","result":0}
01701      *
01702      * \endenglish
01703      */
01704 int setVecInt32(const std::string &key, const std::vector<int32_t> &value);
01705
01706 /**
01707  * @ingroup RegisterControl
01708  * \chinese
01709  * 获取变量值
01710  *
01711  * @param key
01712  * @param default_value
01713  * @return
01714  *
01715  * @par Python 函数原型
01716  * getFloat(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: float) ->
01717  * float
01718  *
01719  * @par Lua 函数原型
01720  * getFloat(key: string, default_value: number) -> number
01721  *
01722  * @par Lua 示例
01723  * var_Float = getFloat("custom",0.0)
01724  *
01725  * @par JSON-RPC 请求示例
01726  * {"jsonrpc":"2.0","method":"RegisterControl.getFloat","params":["custom",0.0],"id":1}
01727  *
01728  * @par JSON-RPC 响应示例
01729  * {"id":1,"jsonrpc":"2.0","result":4.400000095367432}
01730  *
01731  * \endchinese
01732  * \english
01733  * Get variable value
01734  *
01735  * @param key
01736  * @param default_value
01737  * @return
01738  *
01739  * @par Python interface prototype
01740  * getFloat(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: float) ->
01741  * float
01742  *
01743  * @par Lua interface prototype
01744  * getFloat(key: string, default_value: number) -> number
01745  *
01746  * @par Lua example
01747  * var_Float = getFloat("custom",0.0)
01748  *
01749  * @par JSON-RPC request example
01750  * {"jsonrpc":"2.0","method":"RegisterControl.getFloat","params":["custom",0.0],"id":1}
01751  *
01752  * @par JSON-RPC response example
01753  * {"id":1,"jsonrpc":"2.0","result":4.400000095367432}
01754  *
01755  * \endenglish
01756  */
01757 float getFloat(const std::string &key, float default_value);
01758
01759 /**
01760  * @ingroup RegisterControl
01761  * \chinese
01762  * 设置/更新变量值
01763  *

```

```

01764     * @param key
01765     * @param value
01766     * @return
01767     *
01768     * @par Python 函数原型
01769     * setFloat(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: float) -> int
01770     *
01771     * @par Lua 函数原型
01772     * setFloat(key: string, value: number) -> nil
01773     *
01774     * @par Lua 示例
01775     * setFloat("custom",4.4)
01776     *
01777     * @par JSON-RPC 请求示例
01778     * {"jsonrpc":"2.0","method":"RegisterControl.setFloat","params":["custom",4.4],"id":1}
01779     *
01780     * @par JSON-RPC 响应示例
01781     * {"id":1,"jsonrpc":"2.0","result":0}
01782     *
01783     * \endchinese
01784     * \english
01785     * Set or update the variable value
01786     *
01787     * @param key
01788     * @param value
01789     * @return
01790     *
01791     * @par Python interface prototype
01792     * setFloat(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: float) -> int
01793     *
01794     * @par Lua interface prototype
01795     * setFloat(key: string, value: number) -> nil
01796     *
01797     * @par Lua example
01798     * setFloat("custom",4.4)
01799     *
01800     * @par JSON-RPC request example
01801     * {"jsonrpc":"2.0","method":"RegisterControl.setFloat","params":["custom",4.4],"id":1}
01802     *
01803     * @par JSON-RPC response example
01804     * {"id":1,"jsonrpc":"2.0","result":0}
01805     *
01806     * \endenglish
01807     */
01808 int setFloat(const std::string &key, float value);
01809
01810 /**
01811  * @ingroup RegisterControl
01812  * \chinese
01813  * 获取变量值
01814  *
01815  * @param key
01816  * @param default_value
01817  * @return
01818  *
01819  * @par Python 函数原型
01820  * getVecFloat(self: pyaubo_sdk.RegisterControl, arg0: str, arg1:
01821  * List[float]) -> List[float]
01822  *
01823  * @par Lua 函数原型
01824  * getVecFloat(key: string, default_value: table) -> table
01825  *
01826  * @par Lua 示例
01827  * VecFloat = getVecFloat("custom",{})
01828  *
01829  * @par JSON-RPC 请求示例
01830  * {"jsonrpc":"2.0","method":"RegisterControl.getVecFloat","params":["custom",[]],"id":1}
01831  *
01832  * @par JSON-RPC 响应示例
01833  * {"id":1,"jsonrpc":"2.0","result":[0.0,0.10000000149011612,3.299999952316284]}
01834  *
01835  * \endchinese
01836  * \english
01837  * Get variable value
01838  *
01839  * @param key
01840  * @param default_value
01841  * @return
01842  *
01843  * @par Python interface prototype
01844  * getVecFloat(self: pyaubo_sdk.RegisterControl, arg0: str, arg1:
01845  * List[float]) -> List[float]
01846  *
01847  * @par Lua interface prototype
01848  * getVecFloat(key: string, default_value: table) -> table
01849  *
01850  * @par Lua example

```

```

01851     * VecFloat = getVecFloat("custom",{})
01852     *
01853     * @par JSON-RPC request example
01854     * {"jsonrpc":"2.0","method":"RegisterControl.getVecFloat","params":["custom",[]],"id":1}
01855     *
01856     * @par JSON-RPC response example
01857     * {"id":1,"jsonrpc":"2.0","result":[0.0,0.10000000149011612,3.299999952316284]}
01858     *
01859     * \endenglish
01860     */
01861 std::vector<float> getVecFloat(const std::string &key,
01862                               const std::vector<float> &default_value);
01863
01864 /**
01865  * @ingroup RegisterControl
01866  * \chinese
01867  * 设置/更新变量值
01868  *
01869  * @param key
01870  * @param value
01871  * @return
01872  *
01873  * @par Python 函数原型
01874  * setVecFloat(self: pyaubo_sdk.RegisterControl, arg0: str, arg1:
01875  * List[float]) -> int
01876  *
01877  * @par Lua 函数原型
01878  * setVecFloat(key: string, value: table) -> nil
01879  *
01880  * @par Lua 示例
01881  * setVecFloat("custom", {0.0,0.1,3.3})
01882  *
01883  * @par JSON-RPC 请求示例
01884  * {"jsonrpc":"2.0","method":"RegisterControl.setVecFloat","params":["custom",[0.0,0.1,3.3]],"id":1}
01885  *
01886  * @par JSON-RPC 响应示例
01887  * {"id":1,"jsonrpc":"2.0","result":0}
01888  *
01889  * \endchinese
01890  * \english
01891  * Set or update the variable value
01892  *
01893  * @param key
01894  * @param value
01895  * @return
01896  *
01897  * @par Python interface prototype
01898  * setVecFloat(self: pyaubo_sdk.RegisterControl, arg0: str, arg1:
01899  * List[float]) -> int
01900  *
01901  * @par Lua interface prototype
01902  * setVecFloat(key: string, value: table) -> nil
01903  *
01904  * @par Lua example
01905  * setVecFloat("custom", {0.0,0.1,3.3})
01906  *
01907  * @par JSON-RPC request example
01908  * {"jsonrpc":"2.0","method":"RegisterControl.setVecFloat","params":["custom",[0.0,0.1,3.3]],"id":1}
01909  *
01910  * @par JSON-RPC response example
01911  * {"id":1,"jsonrpc":"2.0","result":0}
01912  *
01913  * \endenglish
01914  */
01915 int setVecFloat(const std::string &key, const std::vector<float> &value);
01916
01917 /**
01918  * @ingroup RegisterControl
01919  * \chinese
01920  * 获取变量值
01921  *
01922  * @param key
01923  * @param default_value
01924  * @return
01925  *
01926  * @par Python 函数原型
01927  * getDouble(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: float) ->
01928  * float
01929  *
01930  * @par Lua 函数原型
01931  * getDouble(key: string, default_value: number) -> number
01932  *
01933  * @par Lua 示例
01934  * var_Double = getDouble("custom",0.0)
01935  *
01936  * @par JSON-RPC 请求示例
01937  * {"jsonrpc":"2.0","method":"RegisterControl.getDouble","params":["custom",0.0],"id":1}

```

```

01938      *
01939      * @par JSON-RPC 响应示例
01940      * {"id":1,"jsonrpc":"2.0","result":0.0}
01941      *
01942      * \endchinese
01943      * \english
01944      * Get variable value
01945      *
01946      * @param key
01947      * @param default_value
01948      * @return
01949      *
01950      * @par Python interface prototype
01951      * getDouble(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: float) ->
01952      * float
01953      *
01954      * @par Lua interface prototype
01955      * getDouble(key: string, default_value: number) -> number
01956      *
01957      * @par Lua example
01958      * var_Double = getDouble("custom",0.0)
01959      *
01960      * @par JSON-RPC request example
01961      * {"jsonrpc":"2.0","method":"RegisterControl.getDouble","params":["custom",0.0],"id":1}
01962      *
01963      * @par JSON-RPC response example
01964      * {"id":1,"jsonrpc":"2.0","result":0.0}
01965      *
01966      * \endenglish
01967      */
01968 double getDouble(const std::string &key, double default_value);
01969
01970 /**
01971  * @ingroup RegisterControl
01972  * \chinese
01973  * 设置/更新变量值
01974  *
01975  * @param key
01976  * @param value
01977  * @return
01978  *
01979  * @par Python 函数原型
01980  * setDouble(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: float) ->
01981  * int
01982  *
01983  * @par Lua 函数原型
01984  * setDouble(key: string, value: number) -> nil
01985  *
01986  * @par Lua 示例
01987  * setDouble("custom",6.6)
01988  *
01989  * @par JSON-RPC 请求示例
01990  * {"jsonrpc":"2.0","method":"RegisterControl.setDouble","params":["custom",6.6],"id":1}
01991  *
01992  * @par JSON-RPC 响应示例
01993  * {"id":1,"jsonrpc":"2.0","result":0}
01994  *
01995  * \endchinese
01996  * \english
01997  * Set or update the variable value
01998  *
01999  * @param key
02000  * @param value
02001  * @return
02002  *
02003  * @par Python interface prototype
02004  * setDouble(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: float) ->
02005  * int
02006  *
02007  * @par Lua interface prototype
02008  * setDouble(key: string, value: number) -> nil
02009  *
02010  * @par Lua example
02011  * setDouble("custom",6.6)
02012  *
02013  * @par JSON-RPC request example
02014  * {"jsonrpc":"2.0","method":"RegisterControl.setDouble","params":["custom",6.6],"id":1}
02015  *
02016  * @par JSON-RPC response example
02017  * {"id":1,"jsonrpc":"2.0","result":0}
02018  *
02019  * \endenglish
02020  */
02021 int setDouble(const std::string &key, double value);
02022
02023 /**
02024  * @ingroup RegisterControl

```

```

02025     * \chinese
02026     * 获取变量值
02027     *
02028     * @param key
02029     * @param default_value
02030     * @return
02031     *
02032     * @par Python 函数原型
02033     * getVecDouble(self: pyaubo_sdk.RegisterControl, arg0: str, arg1:
02034     * List[float]) -> List[float]
02035     *
02036     * @par Lua 函数原型
02037     * getVecDouble(key: string, default_value: table) -> table
02038     *
02039     * @par Lua 示例
02040     * VecDouble = getVecDouble("custom",{})
02041     *
02042     * @par JSON-RPC 请求示例
02043     * {"jsonrpc":"2.0","method":"RegisterControl.getVecDouble","params":["custom",[]],"id":1}
02044     *
02045     * @par JSON-RPC 响应示例
02046     * {"id":1,"jsonrpc":"2.0","result":[0.1,0.2,0.3]}
02047     *
02048     * \endchinese
02049     * \english
02050     * Get variable value
02051     *
02052     * @param key
02053     * @param default_value
02054     * @return
02055     *
02056     * @par Python interface prototype
02057     * getVecDouble(self: pyaubo_sdk.RegisterControl, arg0: str, arg1:
02058     * List[float]) -> List[float]
02059     *
02060     * @par Lua interface prototype
02061     * getVecDouble(key: string, default_value: table) -> table
02062     *
02063     * @par Lua example
02064     * VecDouble = getVecDouble("custom",{})
02065     *
02066     * @par JSON-RPC request example
02067     * {"jsonrpc":"2.0","method":"RegisterControl.getVecDouble","params":["custom",[]],"id":1}
02068     *
02069     * @par JSON-RPC response example
02070     * {"id":1,"jsonrpc":"2.0","result":[0.1,0.2,0.3]}
02071     *
02072     * \endenglish
02073     */
02074     std::vector<double> getVecDouble(const std::string &key,
02075                                     const std::vector<double> &default_value);
02076
02077     /**
02078     * @ingroup RegisterControl
02079     * \chinese
02080     * 设置/更新变量值
02081     *
02082     * @param key
02083     * @param value
02084     * @return
02085     *
02086     * @par Python 函数原型
02087     * setVecDouble(self: pyaubo_sdk.RegisterControl, arg0: str, arg1:
02088     * List[float]) -> int
02089     *
02090     * @par Lua 函数原型
02091     * setVecDouble(key: string, value: table) -> nil
02092     *
02093     * @par Lua 示例
02094     * setVecDouble("custom",{0.1,0.2,0.3})
02095     *
02096     * @par JSON-RPC 请求示例
02097     * {"jsonrpc":"2.0","method":"RegisterControl.setVecDouble","params":["custom",[0.1,0.2,0.3]],"id":1}
02098     *
02099     * @par JSON-RPC 响应示例
02100     * {"id":1,"jsonrpc":"2.0","result":0}
02101     *
02102     * \endchinese
02103     * \english
02104     * Set or update the variable value
02105     *
02106     * @param key
02107     * @param value
02108     * @return
02109     *
02110     * @par Python interface prototype
02111     * setVecDouble(self: pyaubo_sdk.RegisterControl, arg0: str, arg1:

```



```

02112     * List[float]) -> int
02113     *
02114     * @par Lua interface prototype
02115     * setVecDouble(key: string, value: table) -> nil
02116     *
02117     * @par Lua example
02118     * setVecDouble("custom",{0.1,0.2,0.3})
02119     *
02120     * @par JSON-RPC request example
02121     * {"jsonrpc":"2.0","method":"RegisterControl.setVecDouble","params":["custom",[0.1,0.2,0.3]],"id":1}
02122     *
02123     * @par JSON-RPC response example
02124     * {"id":1,"jsonrpc":"2.0","result":0}
02125     *
02126     * \endenglish
02127     */
02128     int setVecDouble(const std::string &key, const std::vector<double> &value);
02129
02130     /**
02131     * @ingroup RegisterControl
02132     * \chinese
02133     * 获取变量值
02134     *
02135     * @param key
02136     * @param default_value
02137     * @return
02138     *
02139     * @par Python 函数原型
02140     * getString(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: str) -> str
02141     *
02142     * @par Lua 函数原型
02143     * getString(key: string, default_value: string) -> string
02144     *
02145     * @par Lua 示例
02146     * var_String = getString("custom","")
02147     *
02148     * @par JSON-RPC 请求示例
02149     * {"jsonrpc":"2.0","method":"RegisterControl.getString","params":["custom",""],"id":1}
02150     *
02151     * @par JSON-RPC 响应示例
02152     * {"id":1,"jsonrpc":"2.0","result":"test"}
02153     *
02154     * \endchinese
02155     * \english
02156     * Get variable value
02157     *
02158     * @param key
02159     * @param default_value
02160     * @return
02161     *
02162     * @par Python interface prototype
02163     * getString(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: str) -> str
02164     *
02165     * @par Lua interface prototype
02166     * getString(key: string, default_value: string) -> string
02167     *
02168     * @par Lua example
02169     * var_String = getString("custom","")
02170     *
02171     * @par JSON-RPC request example
02172     * {"jsonrpc":"2.0","method":"RegisterControl.getString","params":["custom",""],"id":1}
02173     *
02174     * @par JSON-RPC response example
02175     * {"id":1,"jsonrpc":"2.0","result":"test"}
02176     *
02177     * \endenglish
02178     */
02179     std::string getString(const std::string &key,
02180                          const std::string &default_value);
02181
02182     /**
02183     * @ingroup RegisterControl
02184     * \chinese
02185     * 设置/更新变量值
02186     *
02187     * @param key
02188     * @param value
02189     * @return
02190     *
02191     * @par Python 函数原型
02192     * setString(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: str) -> int
02193     *
02194     * @par Lua 函数原型
02195     * setString(key: string, value: string) -> nil
02196     *
02197     * @par Lua 示例
02198     * setString("custom","test")

```

```

02199      *
02200      * @par JSON-RPC 请求示例
02201      * {"jsonrpc": "2.0", "method": "RegisterControl.setString", "params": ["custom", "test"], "id": 1}
02202      *
02203      * @par JSON-RPC 响应示例
02204      * {"id": 1, "jsonrpc": "2.0", "result": 0}
02205      *
02206      * \endchinese
02207      * \english
02208      * Set or update the variable value
02209      *
02210      * @param key
02211      * @param value
02212      * @return Returns 0 on success, otherwise error code
02213      *
02214      * @par Python interface prototype
02215      * setString(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: str) -> int
02216      *
02217      * @par Lua interface prototype
02218      * setString(key: string, value: string) -> nil
02219      *
02220      * @par Lua example
02221      * setString("custom", "test")
02222      *
02223      * @par JSON-RPC request example
02224      * {"jsonrpc": "2.0", "method": "RegisterControl.setString", "params": ["custom", "test"], "id": 1}
02225      *
02226      * @par JSON-RPC response example
02227      * {"id": 1, "jsonrpc": "2.0", "result": 0}
02228      *
02229      * \endenglish
02230      */
02231 int setString(const std::string &key, const std::string &value);
02232
02233 /**
02234  * @ingroup RegisterControl
02235  * \chinese
02236  * 清除变量
02237  *
02238  * @param key
02239  * @return
02240  *
02241  * @par Python 函数原型
02242  * clearNamedVariable(self: pyaubo_sdk.RegisterControl, arg0: str) -> int
02243  *
02244  * @par Lua 函数原型
02245  * clearNamedVariable(key: string) -> nil
02246  *
02247  * @par Lua 示例
02248  * clearNamedVariable("custom")
02249  *
02250  * @par JSON-RPC 请求示例
02251  * {"jsonrpc": "2.0", "method": "RegisterControl.clearNamedVariable", "params": ["custom"], "id": 1}
02252  *
02253  * @par JSON-RPC 响应示例
02254  * {"id": 1, "jsonrpc": "2.0", "result": 1}
02255  *
02256  * \endchinese
02257  * \english
02258  * Clear variable
02259  *
02260  * @param key
02261  * @return
02262  *
02263  * @par Python interface prototype
02264  * clearNamedVariable(self: pyaubo_sdk.RegisterControl, arg0: str) -> int
02265  *
02266  * @par Lua interface prototype
02267  * clearNamedVariable(key: string) -> nil
02268  *
02269  * @par Lua example
02270  * clearNamedVariable("custom")
02271  *
02272  * @par JSON-RPC request example
02273  * {"jsonrpc": "2.0", "method": "RegisterControl.clearNamedVariable", "params": ["custom"], "id": 1}
02274  *
02275  * @par JSON-RPC response example
02276  * {"id": 1, "jsonrpc": "2.0", "result": 1}
02277  *
02278  * \endenglish
02279  */
02280 int clearNamedVariable(const std::string &key);
02281
02282 /**
02283  * @ingroup RegisterControl
02284  * \chinese
02285  * 设置看门狗

```

```

02286 *
02287 * 看门狗被触发之后控制器会执行对应的动作，并自动删除看门狗
02288 *
02289 * @param key
02290 * @param timeout 超时时间，单位秒 (s)，超时时间最小为 0.1s
02291 * @param action
02292 * NONE (0): 无动作
02293 * PAUSE(1): 暂停运行时
02294 * STOP (2): 停止运行时/停止机器人运动
02295 * PROTECTIVE_STOP (3): 触发防护停止
02296 * @return
02297 * \endchinese
02298 * \english
02299 * Set the watchdog
02300 *
02301 * After the watchdog is triggered, the controller will perform the
02302 * corresponding action and automatically delete the watchdog.
02303 *
02304 * @param key
02305 * @param timeout Timeout in seconds (s), minimum timeout is 0.1s
02306 * @param action
02307 * NONE (0): No action
02308 * PAUSE(1): Pause runtime
02309 * STOP (2): Stop runtime/stop robot motion
02310 * PROTECTIVE_STOP (3): Trigger protective stop
02311 * @return
02312 * \endenglish
02313 */
02314 int setWatchDog(const std::string &key, double timeout, int action);
02315
02316 /**
02317 * @ingroup RegisterControl
02318 * \chinese
02319 * 获取看门狗动作
02320 *
02321 * @param key
02322 * @return
02323 * \endchinese
02324 * \english
02325 * Get the watchdog action
02326 *
02327 * @param key
02328 * @return
02329 * \endenglish
02330 */
02331 int getWatchDogAction(const std::string &key);
02332
02333 /**
02334 * @ingroup RegisterControl
02335 * \chinese
02336 * 获取看门狗超时时间
02337 *
02338 * @param key
02339 * @return
02340 * \endchinese
02341 * \english
02342 * Get the watchdog timeout value
02343 *
02344 * @param key
02345 * @return
02346 * \endenglish
02347 */
02348 int getWatchDogTimeout(const std::string &key);
02349
02350 /**
02351 * @ingroup RegisterControl
02352 * \chinese
02353 * 添加一个新的 Modbus 信号以供控制器监视。不需要返回响应。
02354 *
02355 * @param device_info 设备信息
02356 * 设备信息是 RTU 格式，例如: "serial_port,baud,parity,data_bit,stop_bit"
02357 * (1)serial_port 参数指定串口的名称，例如，在 Linux 上为"/dev/ttyS0" 或"/dev/ttyUSB0"，在 Windows 上为"\\.COM10"
02358 * (2)baud 参数指定通信的波特率，例如 9600、19200、57600、115200 等
02359 * (3)parity 参数指定奇偶校验方式，N 表示无校验，E 表示偶校验，O 表示奇校验
02360 * (4)data_bit 参数指定数据位数，允许的值为 5、6、7 和 8
02361 * (5)stop_bit 参数指定停止位数，允许的值为 1 和 2
02362 *
02363 * 设备信息是 TCP 格式，例如: "ip address,port"
02364 * (1)ip address 参数指定服务器的 IP 地址
02365 * (2)port 参数指定服务器监听的端口号
02366 * @param slave_number 通常不使用，设置为 255 即可，但可以在 0 到 255 之间自由选择
02367 * @param signal_address
02368 * 指定新信号应该反映的线圈或寄存器的地址。请参考 Modbus 单元的配置以获取此信息。
02369 * @param signal_type 指定要添加的信号类型。0 = 数字输入，1 = 数字输出，2 =
02370 * 寄存器输入，3 = 寄存器输出。
02371 * @param signal_name
02372 * 唯一标识信号的名词。如果提供的字符串与已添加的信号相等，则新信号将替换旧信号。字符串的长度不能超过 20 个字符。

```

```

02373     * @param sequential_mode
02374     * 设置为 True 会强制 Modbus 客户端在发送下一个请求之前等待响应。某些 fieldbus 单元需要此模式。可选参数。
02375     * @return
02376     *
02377     * @par Python 函数原型
02378     * modbusAddSignal(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: int,
02379     * arg2: int, arg3: int, arg4: str, arg5: bool) -> int
02380     *
02381     * @par Lua 函数原型
02382     * modbusAddSignal(device_info: string, slave_number: number,
02383     * signal_address: number, signal_type: number, signal_name: string,
02384     * sequential_mode: boolean) -> nil
02385     *
02386     * @par Lua 示例
02387     * modbusAddSignal("/dev/ttyRobotTool,115200,N,8,1", 1, signal_address:
02388     * number, 264, "Modbus_0", false)
02389     *
02390     * @par JSON-RPC 请求示例
02391     * {"jsonrpc": "2.0", "method": "RegisterControl.modbusAddSignal", "params": ["/dev/
ttyRobotTool,115200,N,8,1",1,264,3,"Modbus_0",false], "id": 1}
02392     *
02393     * @par JSON-RPC 响应示例
02394     * {"id": 1, "jsonrpc": "2.0", "result": 0}
02395     * \endchinese
02396     * \english
02397     * Adds a new modbus signal for the controller to supervise. Expects no
02398     * response.
02399     *
02400     * @param device_info is rtu format.
02401     * eg, "serial_port, baud, parity, data_bit, stop_bit"
02402     * (1) The serial_port argument specifies the name of the serial port eg. On
02403     * Linux, "/dev/ttyS0" or "/dev/ttyUSB0". On Windows, "\\.\COM10". (2) The
02404     * baud argument specifies the baud rate of the communication, eg. 9600,
02405     * 19200, 57600, 115200, etc. (3) parity: N for none, E for even, O for odd.
02406     * (4) data_bit: The data_bits argument specifies the number of bits of data,
02407     * the allowed values are 5, 6, 7 and 8. (5) stop_bit: The stop_bits argument
02408     * specifies the bits of stop, the allowed values are 1 and 2.
02409     *
02410     * device_info is tcp format. eg, "ip address, port"
02411     * (1) The ip address parameter specifies the ip address of the server
02412     * (2) The port parameter specifies the port number that the server is
02413     * listening on.
02414     * @param slave_number An integer normally not used and set to 255, but is a
02415     * free choice between 0 and 255.
02416     * @param signal_address An integer specifying the address of the either the
02417     * coil or the register that this new signal should reflect. Consult the
02418     * configuration of the modbus unit for this information.
02419     * @param signal_type An integer specifying the type of signal to add. 0 =
02420     * digital input, 1 = digital output, 2 = register input and 3 = register
02421     * output.
02422     * @param signal_name A string uniquely identifying the signal. If a string
02423     * is supplied which is equal to an already added signal, the new signal
02424     * will replace the old one. The length of the string cannot exceed 20
02425     * characters.
02426     * @param sequential_mode Setting to True forces the modbus client to wait
02427     * for a response before sending the next request. This mode is required by
02428     * some fieldbus units (Optional).
02429     * @return
02430     *
02431     * @par Python interface prototype
02432     * modbusAddSignal(self: pyaubo_sdk.RegisterControl, arg0: str, arg1: int,
02433     * arg2: int, arg3: int, arg4: str, arg5: bool) -> int
02434     *
02435     * @par Lua interface prototype
02436     * modbusAddSignal(device_info: string, slave_number: number,
02437     * signal_address: number, signal_type: number, signal_name: string,
02438     * sequential_mode: boolean) -> nil
02439     *
02440     * @par Lua example
02441     * modbusAddSignal("/dev/ttyRobotTool,115200,N,8,1", 1, signal_address:
02442     * number, 264, "Modbus_0", false)
02443     *
02444     * @par JSON-RPC request example
02445     * {"jsonrpc": "2.0", "method": "RegisterControl.modbusAddSignal", "params": ["/dev/
ttyRobotTool,115200,N,8,1",1,264,3,"Modbus_0",false], "id": 1}
02446     *
02447     * @par JSON-RPC response example
02448     * {"id": 1, "jsonrpc": "2.0", "result": 0}
02449     * \endenglish
02450     */
02451 int modbusAddSignal(const std::string &device_info, int slave_number,
02452                    int signal_address, int signal_type,
02453                    const std::string &signal_name, bool sequential_mode);
02454
02455 /**
02456  * @ingroup RegisterControl
02457  * \chinese

```

```

02458 * 为指定 Modbus 寄存器信号创建固定读取分组, 优化连续地址的批量读取效率
02459 *
02460 * 功能: 根据传入的起始信号名定位到对应的 16 位 Modbus 寄存器地址,
02461 * 并基于该起始地址和连续数量创建 "不可扩展的固定分组", 该分组会被独立管理,
02462 * 不会与其他普通分组合并或扩展, 适用于需要稳定、原子化读取的连续寄存器地址段;
02463 * 固定分组创建后, 对该段地址的读取操作会严格按照分组范围执行, 避免因动态分组扩展导致的读取范围变化。
02464 *
02465 * @param signal_name
02466 * 起始信号名 (作为固定分组的基准信号, 需为已通过 modbusAddSignal 创建的 16 位 Modbus 寄存器信号,
02467 * 支持保持寄存器/输入寄存器类型, 不支持数字量信号)
02468 * @param count 固定分组包含的连续信号 (寄存器) 数量 (大于等于 1):
02469 * - 值为 1:
02470 * 仅包含起始信号对应的单个寄存器地址 (取消包含该信号的固定分组);
02471 * - 值 >1:
02472 * 从起始信号地址开始的连续 count 个寄存器地址, 需确保地址连续且不出设备支持范围;
02473 *
02474 * @return 成功返回 0, 失败返回错误码。
02475 *
02476 * @note
02477 * 1. 该接口仅针对 16 位 Modbus 寄存器信号 (signal_type 为 2/3) 生效;
02478 * 2. 单个固定分组的最大连续数量受设备限制 (默认最大 30 个);
02479 * 3. 重复为同一信号创建固定分组会覆盖原有配置 (以最后一次的 count 值为准)。
02480 *
02481 * @par Python 接口原型
02482 * addFixedModbusGroup(self: pyaubo_sdk.RegisterControl, arg0: str, arg1:
02483 * int) -> int
02484 *
02485 * @par Lua 接口原型
02486 * addFixedModbusGroup(signal_name: string, count: number) -> nil
02487 *
02488 * @par Lua 示例
02489 * -- 为起始信号"Modbus_0" 创建包含 10 个连续寄存器的固定分组
02490 * addFixedModbusGroup("Modbus_0", 10)
02491 *
02492 * @par JSON-RPC 请求示例
02493 * {"jsonrpc": "2.0", "method": "RegisterControl.addFixedModbusGroup", "params": ["Modbus_0", 10], "id": 1}
02494 *
02495 * @par JSON-RPC 响应示例
02496 * {"id": 1, "jsonrpc": "2.0", "result": 0}
02497 * \endchinese
02498 * \english
02499 * Create a fixed read group for the specified Modbus register signal to
02500 * optimize the batch reading efficiency of continuous addresses
02501 *
02502 * Function: Locate the address of the corresponding 16-bit Modbus register
02503 * according to the input start signal name, and create a "non-extensible
02504 * fixed group" based on the start address and continuous count. This group
02505 * will be managed independently, and will not be merged or expanded with
02506 * other normal groups. It is suitable for continuous register address
02507 * segments that require stable and atomic reading; After the fixed group is
02508 * created, the read operation for this address segment will be strictly
02509 * executed according to the group range, avoiding read range changes caused
02510 * by dynamic group expansion.
02511 *
02512 * @param signal_name
02513 * Start signal name (used as the reference signal for the fixed group, must
02514 * be a 16-bit Modbus register signal created via modbusAddSignal,
02515 * supporting holding register/input register types, not supporting digital
02516 * signals)
02517 * @param count Number of continuous signals (registers) contained in the
02518 * fixed group (greater than or equal to 1):
02519 * - Value = 1: Only include the single register address
02520 * corresponding to the start signal (cancel the fixed group containing this
02521 * signal);
02522 * - Value > 1: Continuous count register addresses starting
02523 * from the start signal address, ensure the addresses are continuous and do
02524 * not exceed the device support range;
02525 *
02526 * @return Returns 0 on success, returns an error code on failure.
02527 *
02528 * @note
02529 * 1. This interface only takes effect for 16-bit Modbus register signals
02530 * (signal_type is 2/3);
02531 * 2. The maximum continuous count of a single fixed group is limited by the
02532 * device (default maximum 30);
02533 * 3. Repeated creation of a fixed group for the same start signal will
02534 * overwrite the original configuration (the last count value shall
02535 * prevail).
02536 *
02537 * @par Python interface prototype
02538 * addFixedModbusGroup(self: pyaubo_sdk.RegisterControl, arg0: str, arg1:
02539 * int) -> int
02540 *
02541 * @par Lua interface prototype
02542 * addFixedModbusGroup(signal_name: string, count: number) -> nil
02543 *
02544 * @par Lua example

```

```

02545     * -- Create a fixed group containing 10 consecutive registers for the start
02546     * signal "Modbus_0" addFixedModbusGroup("Modbus_0", 10)
02547     *
02548     * @par JSON-RPC request example
02549     * {"jsonrpc": "2.0", "method": "RegisterControl.addFixedModbusGroup", "params": ["Modbus_0", 10], "id": 1}
02550     *
02551     * @par JSON-RPC response example
02552     * {"id": 1, "jsonrpc": "2.0", "result": 0}
02553     * \endenglish
02554     */
02555     int addFixedModbusGroup(const std::string &signal_name, int count);
02556
02557     /**
02558     * @ingroup RegisterControl
02559     * \chinese
02560     * 删除指定名称的信号。
02561     *
02562     * @param signal_name 要删除的信号的名称
02563     * @return
02564     *
02565     * @par Python 函数原型
02566     * modbusDeleteSignal(self: pyaubo_sdk.RegisterControl, arg0: str) -> int
02567     *
02568     * @par Lua 函数原型
02569     * modbusDeleteSignal(signal_name: string) -> nil
02570     *
02571     * @par Lua 示例
02572     * modbusDeleteSignal("Modbus_1")
02573     *
02574     * @par JSON-RPC 请求示例
02575     * {"jsonrpc": "2.0", "method": "RegisterControl.modbusDeleteSignal", "params": ["Modbus_1"], "id": 1}
02576     *
02577     * @par JSON-RPC 响应示例
02578     * {"id": 1, "jsonrpc": "2.0", "result": 0}
02579     * \endchinese
02580     * \english
02581     * Deletes the signal identified by the supplied signal name.
02582     *
02583     * @param signal_name A string equal to the name of the signal that should
02584     * be deleted.
02585     * @return
02586     *
02587     * @par Python interface prototype
02588     * modbusDeleteSignal(self: pyaubo_sdk.RegisterControl, arg0: str) -> int
02589     *
02590     * @par Lua interface prototype
02591     * modbusDeleteSignal(signal_name: string) -> nil
02592     *
02593     * @par Lua example
02594     * modbusDeleteSignal("Modbus_1")
02595     *
02596     * @par JSON-RPC request example
02597     * {"jsonrpc": "2.0", "method": "RegisterControl.modbusDeleteSignal", "params": ["Modbus_1"], "id": 1}
02598     *
02599     * @par JSON-RPC response example
02600     * {"id": 1, "jsonrpc": "2.0", "result": 0}
02601     * \endenglish
02602     */
02603     int modbusDeleteSignal(const std::string &signal_name);
02604
02605     /**
02606     * @ingroup RegisterControl
02607     * \chinese
02608     * 删除所有 modbus 信号
02609     *
02610     * @return
02611     *
02612     * @par JSON-RPC 请求示例
02613     * {"jsonrpc": "2.0", "method": "RegisterControl.modbusDeleteAllSignals", "params": [], "id": 1}
02614     *
02615     * @par JSON-RPC 响应示例
02616     * {"id": 1, "jsonrpc": "2.0", "result": 0}
02617     *
02618     * \endchinese
02619     * \english
02620     * Delete all modbus signals
02621     *
02622     * @return
02623     *
02624     * @par JSON-RPC request example
02625     * {"jsonrpc": "2.0", "method": "RegisterControl.modbusDeleteAllSignals", "params": [], "id": 1}
02626     *
02627     * @par JSON-RPC response example
02628     * {"id": 1, "jsonrpc": "2.0", "result": 0}
02629     *
02630     * \endenglish
02631     */

```

```

02632     int modbusDeleteAllSignals();
02633
02634     /**
02635      * @ingroup RegisterControl
02636      * \chinese
02637      * 读取特定信号的当前值。
02638      *
02639      * @param signal_name 要获取值的信号的名称
02640      * @return 对于数字信号: 1 或 0。
02641      * 对于寄存器信号: 表示为整数的寄存器值。如果值为-1, 则表示该信号不存在。
02642      *
02643      * @par Python 函数原型
02644      * modbusGetSignalStatus(self: pyaubo_sdk.RegisterControl, arg0: str) -> int
02645      *
02646      * @par Lua 函数原型
02647      * modbusGetSignalStatus(signal_name: string) -> number
02648      *
02649      * @par Lua 示例
02650      * var0 = modbusGetSignalStatus("Modbus_0")
02651      *
02652      * @par JSON-RPC 请求示例
02653      * {"jsonrpc": "2.0", "method": "RegisterControl.modbusGetSignalStatus", "params": ["Modbus_0"], "id": 1}
02654      *
02655      * @par JSON-RPC 响应示例
02656      * {"id": 1, "jsonrpc": "2.0", "result": 1}
02657      * \endchinese
02658      * \english
02659      * Reads the current value of a specific signal.
02660      *
02661      * @param signal_name A string equal to the name of the signal for which the
02662      * value should be gotten.
02663      * @return An integer or a boolean. For digital signals: 1 or 0. For
02664      * register signals: The register value expressed as an integer. If the
02665      * value is -1, it means the signal does not exist.
02666      *
02667      * @par Python interface prototype
02668      * modbusGetSignalStatus(self: pyaubo_sdk.RegisterControl, arg0: str) -> int
02669      *
02670      * @par Lua interface prototype
02671      * modbusGetSignalStatus(signal_name: string) -> number
02672      *
02673      * @par Lua example
02674      * var0 = modbusGetSignalStatus("Modbus_0")
02675      *
02676      * @par JSON-RPC request example
02677      * {"jsonrpc": "2.0", "method": "RegisterControl.modbusGetSignalStatus", "params": ["Modbus_0"], "id": 1}
02678      *
02679      * @par JSON-RPC response example
02680      * {"id": 1, "jsonrpc": "2.0", "result": 1}
02681      * \endenglish
02682      */
02683     int modbusGetSignalStatus(const std::string &signal_name);
02684
02685     /**
02686      * @ingroup RegisterControl
02687      * \chinese
02688      * 获取所有信号的名字集合
02689      *
02690      * @return 所有信号的名字集合
02691      *
02692      * @par JSON-RPC 请求示例
02693      * {"jsonrpc": "2.0", "method": "RegisterControl.modbusGetSignalNames", "params": [], "id": 1}
02694      *
02695      * @par JSON-RPC 响应示例
02696      * {"id": 1, "jsonrpc": "2.0", "result": ["Modbus_0"]}
02697      *
02698      * \endchinese
02699      * \english
02700      * Get the collection of all signal names
02701      *
02702      * @return Collection of all signal names
02703      *
02704      * @par JSON-RPC request example
02705      * {"jsonrpc": "2.0", "method": "RegisterControl.modbusGetSignalNames", "params": [], "id": 1}
02706      *
02707      * @par JSON-RPC response example
02708      * {"id": 1, "jsonrpc": "2.0", "result": ["Modbus_0"]}
02709      *
02710      * \endenglish
02711      */
02712     std::vector<std::string> modbusGetSignalNames();
02713
02714     /**
02715      * @ingroup RegisterControl
02716      * \chinese
02717      * 获取所有信号的类型集合
02718      *

```



```

02719     * @return 所有信号的类型集合
02720     *
02721     * @par JSON-RPC 请求示例
02722     * {"jsonrpc": "2.0", "method": "RegisterControl.modbusGetSignalTypes", "params": [], "id": 1}
02723     *
02724     * @par JSON-RPC 响应示例
02725     * {"id": 1, "jsonrpc": "2.0", "result": [1, 0, 2, 3]}
02726     *
02727     * \endchinese
02728     * \english
02729     * Get the collection of all signal types
02730     *
02731     * @return Collection of all signal types
02732     *
02733     * @par JSON-RPC request example
02734     * {"jsonrpc": "2.0", "method": "RegisterControl.modbusGetSignalTypes", "params": [], "id": 1}
02735     *
02736     * @par JSON-RPC response example
02737     * {"id": 1, "jsonrpc": "2.0", "result": [1, 0, 2, 3]}
02738     *
02739     * \endenglish
02740     */
02741     std::vector<int> modbusGetSignalTypes();
02742
02743     /**
02744     * @ingroup RegisterControl
02745     * \chinese
02746     * 获取所有信号的数值集合
02747     *
02748     * @return 所有信号的数值集合
02749     *
02750     * @par JSON-RPC 请求示例
02751     * {"jsonrpc": "2.0", "method": "RegisterControl.modbusGetSignalValues", "params": [], "id": 1}
02752     *
02753     * @par JSON-RPC 响应示例
02754     * {"id": 1, "jsonrpc": "2.0", "result": [1, 1, 88, 33]}
02755     *
02756     * \endchinese
02757     * \english
02758     * Get the collection of all signal values
02759     *
02760     * @return Collection of all signal values
02761     *
02762     * @par JSON-RPC request example
02763     * {"jsonrpc": "2.0", "method": "RegisterControl.modbusGetSignalValues", "params": [], "id": 1}
02764     *
02765     * @par JSON-RPC response example
02766     * {"id": 1, "jsonrpc": "2.0", "result": [1, 1, 88, 33]}
02767     *
02768     * \endenglish
02769     */
02770     std::vector<int> modbusGetSignalValues();
02771
02772     /**
02773     * @ingroup RegisterControl
02774     * \chinese
02775     * 获取所有信号的请求是否有错误 (0: 无错误, 其他: 有错误) 集合
02776     *
02777     * @return ModbusErrorNum
02778     *
02779     * @par JSON-RPC 请求示例
02780     * {"jsonrpc": "2.0", "method": "RegisterControl.modbusGetSignalErrors", "params": [], "id": 1}
02781     *
02782     * @par JSON-RPC 响应示例
02783     * {"id": 1, "jsonrpc": "2.0", "result": [6, 6, 6, 6]}
02784     *
02785     * \endchinese
02786     * \english
02787     * Get the error status of all signal requests (0: no error, others: error)
02788     * as a collection
02789     *
02790     * @return ModbusErrorNum
02791     *
02792     * @par JSON-RPC request example
02793     * {"jsonrpc": "2.0", "method": "RegisterControl.modbusGetSignalErrors", "params": [], "id": 1}
02794     *
02795     * @par JSON-RPC response example
02796     * {"id": 1, "jsonrpc": "2.0", "result": [6, 6, 6, 6]}
02797     *
02798     * \endenglish
02799     */
02800     std::vector<int> modbusGetSignalErrors();
02801
02802     /**
02803     * @ingroup RegisterControl
02804     * \chinese
02805     * 将用户指定的命令发送到指定 IP 地址上的 Modbus 单元。

```



```

02806 * 由于不会接收到响应, 因此不能用于请求数据。
02807 * 用户负责提供对所提供的功能码有意义的数据。
02808 * 内置函数负责构建 Modbus 帧, 因此用户不需要关心命令的长度。
02809 *
02810 * @param device_info 设备信息
02811 * 设备信息是 RTU 格式, 例如: "serial_port,baud,parity,data_bit,stop_bit"
02812 * (1)serial_port 参数指定串口的名称, 例如, 在 Linux 上为"/dev/ttyS0" 或"/dev/ttyUSB0", 在 Windows 上为"\\.\\COM10"
02813 * (2)baud 参数指定通信的波特率, 例如 9600、19200、57600、115200 等
02814 * (3)parity 参数指定奇偶校验方式, N 表示无校验, E 表示偶校验, O 表示奇校验
02815 * (4)data_bit 参数指定数据位数, 允许的值为 5、6、7 和 8
02816 * (5)stop_bit 参数指定停止位数, 允许的值为 1 和 2
02817 *
02818 * 设备信息是 TCP 格式, 例如: "ip address,port"
02819 * (1)ip address 参数指定服务器的 IP 地址
02820 * (2)port 参数指定服务器监听的端口号
02821 * @param slave_number 指定用于自定义命令的从站号
02822 * @param function_code 指定自定义命令的功能码
02823 *
02824 * Modbus 功能码
02825 * MODBUS_FC_READ_COILS 0x01
02826 * MODBUS_FC_READ_DISCRETE_INPUTS 0x02
02827 * MODBUS_FC_READ_HOLDING_REGISTERS 0x03
02828 * MODBUS_FC_READ_INPUT_REGISTERS 0x04
02829 * MODBUS_FC_WRITE_SINGLE_COIL 0x05
02830 * MODBUS_FC_WRITE_SINGLE_REGISTER 0x06
02831 * MODBUS_FC_READ_EXCEPTION_STATUS 0x07
02832 * MODBUS_FC_WRITE_MULTIPLE_COILS 0x0F
02833 * MODBUS_FC_WRITE_MULTIPLE_REGISTERS 0x10
02834 * MODBUS_FC_REPORT_SLAVE_ID 0x11
02835 * MODBUS_FC_MASK_WRITE_REGISTER 0x16
02836 * MODBUS_FC_WRITE_AND_READ_REGISTERS 0x17
02837 *
02838 * @param data 必须是有效的字节值 (0-255)
02839 * @return
02840 *
02841 * @par Python 函数原型
02842 * modbusSendCustomCommand(self: pyaubo_sdk.RegisterControl, arg0: str,
02843 * arg1: int, arg2: int, arg3: List[int]) -> int
02844 *
02845 * @par Lua 函数原型
02846 * modbusSendCustomCommand(device_info: string, slave_number: number,
02847 * function_code: number, data: table) -> nil
02848 *
02849 * @par Lua 示例
02850 * modbusSendCustomCommand("/dev/ttyRobotTool,115200,N,8,1", 1, 10,
02851 * {1,2,0,2,4,0,0,0}) -> nil
02852 *
02853 * @par JSON-RPC 请求示例
02854 * {"jsonrpc":"2.0","method":"RegisterControl.modbusSendCustomCommand","params":["/dev/
ttyRobotTool,115200,N,8,1",1,10,[1,2,0,2,4,0,0,0]],"id":1}
02855 *
02856 * @par JSON-RPC 响应示例
02857 * {"id":1,"jsonrpc":"2.0","result":0}
02858 * \endchinese
02859 * \english
02860 * Sends a command specified by the user to the modbus unit located on the
02861 * specified IP address. Cannot be used to request data, since the response
02862 * will not be received. The user is responsible for supplying data which
02863 * is meaningful to the supplied function code. The builtin function takes
02864 * care of constructing the modbus frame, so the user should not be
02865 * concerned with the length of the command.
02866 *
02867 * @param device_info is rtu format.
02868 * eg,"serial_port,baud,parity,data_bit,stop_bit"
02869 * (1)The serial_port argument specifies the name of the serial port eg. On
02870 * Linux ,"/dev/ttyS0" or "/dev/ttyUSB0". On Windows, "\.\\COM10".
02871 * (2)The baud argument specifies the baud rate of the communication, eg.
02872 * 9600, 19200, 57600, 115200, etc.
02873 * (3)parity:N for none,E for even,0 for odd.
02874 * (4)data_bit:The data_bits argument specifies the number of bits of data,
02875 * the allowed values are 5, 6, 7 and 8.
02876 * (5)stop_bit:The stop_bits argument
02877 * specifies the bits of stop, the allowed values are 1 and 2.
02878 *
02879 * device_info is tcp format.eg,"ip address,port"
02880 * (1)The ip address parameter specifies the ip address of the server
02881 * (2)The port parameter specifies the port number that the server is
02882 * listening on.
02883 * @param slave_number An integer specifying the slave number to use for
02884 * the custom command.
02885 * @param function_code An integer specifying the function code for the
02886 * custom command.
02887 *
02888 * Modbus function codes
02889 * MODBUS_FC_READ_COILS 0x01
02890 * MODBUS_FC_READ_DISCRETE_INPUTS 0x02
02891 * MODBUS_FC_READ_HOLDING_REGISTERS 0x03

```

```

02892 * MODBUS_FC_READ_INPUT_REGISTERS      0x04
02893 * MODBUS_FC_WRITE_SINGLE_COIL          0x05
02894 * MODBUS_FC_WRITE_SINGLE_REGISTER      0x06
02895 * MODBUS_FC_READ_EXCEPTION_STATUS      0x07
02896 * MODBUS_FC_WRITE_MULTIPLE_COILS      0x0F
02897 * MODBUS_FC_WRITE_MULTIPLE_REGISTERS   0x10
02898 * MODBUS_FC_REPORT_SLAVE_ID            0x11
02899 * MODBUS_FC_MASK_WRITE_REGISTER        0x16
02900 * MODBUS_FC_WRITE_AND_READ_REGISTERS   0x17
02901 *
02902 * @param data An array of integers in which each entry must be a valid
02903 * byte (0-255) value.
02904 * @return
02905 *
02906 * @par Python interface prototype
02907 * modbusSendCustomCommand(self: pyaubo_sdk.RegisterControl, arg0: str,
02908 * arg1: int, arg2: int, arg3: List[int]) -> int
02909 *
02910 * @par Lua interface prototype
02911 * modbusSendCustomCommand(device_info: string, slave_number: number,
02912 * function_code: number, data: table) -> nil
02913 *
02914 * @par Lua example
02915 * modbusSendCustomCommand("/dev/ttyRobotTool,115200,N,8,1", 1, 10,
02916 * {1,2,0,2,4,0,0,0}) -> nil
02917 *
02918 * @par JSON-RPC request example
02919 * {"jsonrpc": "2.0", "method": "RegisterControl.modbusSendCustomCommand", "params": ["/dev/
ttyRobotTool,115200,N,8,1",1,10,[1,2,0,2,4,0,0,0]], "id": 1}
02920 *
02921 * @par JSON-RPC response example
02922 * {"id": 1, "jsonrpc": "2.0", "result": 0}
02923 * \endenglish
02924 */
02925 int modbusSendCustomCommand(const std::string &device_info,
02926                             int slave_number, int function_code,
02927                             const std::vector<uint8_t> &data);
02928
02929 /**
02930 * @ingroup RegisterControl
02931 * \chinese
02932 * 将选择的数字输入信号设置为“default”或“freedrive”
02933 *
02934 * @param robot_name 连接的机器人名称
02935 * @param signal_name 先前被添加的数字输入信号
02936 * @param action 操作类型。操作可以是“default”或“freedrive”
02937 * @return
02938 *
02939 * @par Python 函数原型
02940 * modbusSetDigitalInputAction(self: pyaubo_sdk.RegisterControl, arg0: str,
02941 * arg1: str, arg2: int)
02942 *
02943 * @par Lua 函数原型
02944 * modbusSetDigitalInputAction(robot_name: string, signal_name: string,
02945 * action: number) -> nil
02946 *
02947 * @par Lua 示例
02948 * modbusSetDigitalInputAction("rob1", "Modbus_0", "Handguide")
02949 *
02950 * @par JSON-RPC 请求示例
02951 * {"jsonrpc": "2.0", "method": "RegisterControl.modbusSetDigitalInputAction", "params":
["rob1", "Modbus_0", "Handguide"], "id": 1}
02952 *
02953 * @par JSON-RPC 响应示例
02954 * {"id": 1, "jsonrpc": "2.0", "result": 0}
02955 * \endchinese
02956 * \english
02957 * Sets the selected digital input signal to either a "default" or
02958 * "freedrive" action.
02959 *
02960 * @param robot_name A string identifying a robot name that connected robot
02961 * @param signal_name A string identifying a digital input signal that was
02962 * previously added.
02963 * @param action The type of action. The action can either be "default" or
02964 * "freedrive". (string)
02965 * @return
02966 *
02967 * @par Python interface prototype
02968 * modbusSetDigitalInputAction(self: pyaubo_sdk.RegisterControl, arg0: str,
02969 * arg1: str, arg2: int)
02970 *
02971 * @par Lua interface prototype
02972 * modbusSetDigitalInputAction(robot_name: string, signal_name: string,
02973 * action: number) -> nil
02974 *
02975 * @par Lua example
02976 * modbusSetDigitalInputAction("rob1", "Modbus_0", "Handguide")

```

```

02977     *
02978     * @par JSON-RPC request example
02979     * {"jsonrpc": "2.0", "method": "RegisterControl.modbusSetDigitalInputAction", "params":
["rob1", "Modbus_0", "Handguide"], "id": 1}
02980     *
02981     * @par JSON-RPC response example
02982     * {"id": 1, "jsonrpc": "2.0", "result": 0}
02983     * \endenglish
02984     */
02985     int modbusSetDigitalInputAction(const std::string &robot_name,
02986                                   const std::string &signal_name,
02987                                   StandardInputAction action);
02988
02989     /**
02990     * @ingroup RegisterControl
02991     * \chinese
02992     * 设置 Modbus 信号输出动作
02993     *
02994     * @param robot_name
02995     * @param signal_name
02996     * @param runstate
02997     * @return
02998     *
02999     * @par JSON-RPC 请求示例
03000     * {"jsonrpc": "2.0", "method": "RegisterControl.modbusSetOutputRunstate", "params":
["rob1", "Modbus_0", "None"], "id": 1}
03001     *
03002     * @par JSON-RPC 响应示例
03003     * {"id": 1, "jsonrpc": "2.0", "result": 0}
03004     *
03005     * \endchinese
03006     * \english
03007     * Set Modbus signal output action
03008     *
03009     * @param robot_name
03010     * @param signal_name
03011     * @param runstate
03012     * @return
03013     *
03014     * @par JSON-RPC request example
03015     * {"jsonrpc": "2.0", "method": "RegisterControl.modbusSetOutputRunstate", "params":
["rob1", "Modbus_0", "None"], "id": 1}
03016     *
03017     * @par JSON-RPC response example
03018     * {"id": 1, "jsonrpc": "2.0", "result": 0}
03019     *
03020     * \endenglish
03021     */
03022     int modbusSetOutputRunstate(const std::string &robot_name,
03023                                const std::string &signal_name,
03024                                StandardOutputRunState runstate);
03025
03026     /**
03027     * @ingroup RegisterControl
03028     * \chinese
03029     * 设置 Modbus 信号输出动作，并支持基于路点/平面/障碍物的条件阈值判断。
03030     *
03031     * 详细说明：当需要根据机器人是否到达某个路点或与某个平面/障碍物的距离来触发 Modbus 信号时，
03032     * 可通过本接口指定条件对象及阈值。在条件满足时会将指定的 Modbus 信号设置为对应的运行状态。
03033     *
03034     * @param robot_name 目标机器人的名称，用于标识要监听/触发条件的机器人实例
03035     * @param signal_name
03036     * 要设置的 Modbus 信号名称（需已通过 modbusAddSignal 添加）
03037     * @param runstate 输出动作状态
03038     * @param object_name 条件对象的名称（需为系统中已定义的唯一标识）
03039     * @param threshold 判断阈值：
03040     * - 对于路点：表示角度阈值，单位为弧度（rad）
03041     * - 对于平面/障碍物：表示距离阈值，单位为米（m）
03042     *
03043     * @return 返回操作结果码：
03044     * - 0：操作成功
03045     * - 非 0：操作失败，返回具体错误码（参见系统错误码定义）
03046     *
03047     * @par Python 接口原型
03048     * modbusSetOutputRunstate1(self: pyaubo_sdk.RegisterControl, arg0: str,
03049     * arg1: str, arg2: str, arg3: str, arg4: float) -> int
03050     *
03051     * @par Lua 接口原型
03052     * modbusSetOutputRunstate1(robot_name: string, signal_name: string,
03053     * runstate: string, object_name: string, threshold: number) -> nil
03054     *
03055     * @par JSON-RPC 请求示例
03056     * {"jsonrpc": "2.0", "method": "RegisterControl.modbusSetOutputRunstate1", "params":
["rob1", "Modbus_0", "WaypointArrived", "P1", 3], "id": 1}
03057     *
03058     * @par JSON-RPC 响应示例
03059     * {"id": 1, "jsonrpc": "2.0", "result": 0}

```

```

03060      *
03061      * @note 使用说明:
03062      * 1. 请确保 object_name 与 object_type 对应且已在系统中定义;
03063      * 2. threshold 的单位依据 object_type 不同而不同,
03064      * 务必使用正确单位以避免触发误判;
03065      * \endchinese
03066      * \english
03067      * Set Modbus signal output action with optional condition checking based on
03068      * waypoint, plane or obstacle thresholds.
03069      *
03070      * Detailed description: Use this API to control a Modbus output signal when
03071      * a specified robot condition is met (for example, waypoint arrival or
03072      * distance to a plane/obstacle). Specify an object type, name and a
03073      * threshold value to enable conditional triggering. When the condition is
03074      * satisfied, the given Modbus signal will be set to the requested
03075      * runstate.
03076      *
03077      * @param robot_name The target robot name identifying which robot's
03078      *                    conditions to monitor.
03079      * @param signal_name The Modbus signal name to set (must be previously
03080      *                    added via modbusAddSignal).
03081      * @param runstate The desired output action.
03082      * @param object_name Unique identifier of the condition object
03083      * @param threshold Condition threshold: angle in radians for waypoints, or
03084      *                    distance in meters for planes/obstacles
03085      *
03086      * @return Returns 0 on success, non-zero error code on failure.
03087      *
03088      * @par JSON-RPC request example
03089      * {"jsonrpc":"2.0","method":"RegisterControl.modbusSetOutputRunstate1","params":
["rob1","Modbus_0","WaypointArrived","P1",3],"id":1}
03090      *
03091      * @par JSON-RPC response example
03092      * {"id":1,"jsonrpc":"2.0","result":0}
03093      * \endenglish
03094      */
03095      int modbusSetOutputRunstate1(const std::string &robot_name,
03096                                  const std::string &signal_name,
03097                                  StandardOutputRunState runstate,
03098                                  const std::string &object_name,
03099                                  double threshold);
03100
03101      /**
03102      * @ingroup RegisterControl
03103      * \chinese
03104      * 将指定名称的输出寄存器信号设置为给定的值
03105      *
03106      * @param signal_name 提前被添加的输出寄存器信号
03107      * @param value 必须有效的整数, 范围是 0-65535
03108      * @return
03109      *
03110      * @par Python 函数原型
03111      * modbusSetOutputSignal(self: pyaubo_sdk.RegisterControl, arg0: str, arg1:
03112      * int) -> int
03113      *
03114      * @par Lua 函数原型
03115      * modbusSetOutputSignal(signal_name: string, value: number) -> nil
03116      *
03117      * @par Lua 示例
03118      * modbusSetOutputSignal("Modbus_0",0)
03119      *
03120      * @par JSON-RPC 请求示例
03121      * {"jsonrpc":"2.0","method":"RegisterControl.modbusSetOutputSignal","params":["Modbus_0",0],"id":1}
03122      *
03123      * @par JSON-RPC 响应示例
03124      * {"id":1,"jsonrpc":"2.0","result":0}
03125      * \endchinese
03126      * \english
03127      * Sets the output register signal identified by the given name to the given
03128      * value.
03129      *
03130      * @param signal_name A string identifying an output register signal that in
03131      * advance has been added.
03132      * @param value An integer which must be a valid word (0-65535)
03133      * @return
03134      *
03135      * @par Python interface prototype
03136      * modbusSetOutputSignal(self: pyaubo_sdk.RegisterControl, arg0: str, arg1:
03137      * int) -> int
03138      *
03139      * @par Lua interface prototype
03140      * modbusSetOutputSignal(signal_name: string, value: number) -> nil
03141      *
03142      * @par Lua example
03143      * modbusSetOutputSignal("Modbus_0",0)
03144      *
03145      * @par JSON-RPC request example

```

```

03146     * {"jsonrpc": "2.0", "method": "RegisterControl.modbusSetOutputSignal", "params": ["Modbus_0", 0], "id": 1}
03147     *
03148     * @par JSON-RPC response example
03149     * {"id": 1, "jsonrpc": "2.0", "result": 0}
03150     * \endenglish
03151     */
03152 int modbusSetOutputSignal(const std::string &signal_name, uint16_t value);
03153
03154 /**
03155  * @ingroup RegisterControl
03156  * \chinese
03157  * 将从指定名称开始的若干个连续输出寄存器信号设置为给定的值
03158  *
03159  * @param signal_name 提前被添加的输出寄存器信号的起始名称
03160  * @param values 整数数组, 每个元素范围是 0-65535, 对应连续的多个信号
03161  * @return
03162  *
03163  * @par Python 函数原型
03164  * modbusSetOutputSignal1(self: pyaubo_sdk.RegisterControl, arg0: str, arg1:
03165  * list[int]) -> int
03166  *
03167  * @par Lua 函数原型
03168  * modbusSetOutputSignal1(signal_name: string, value: table) -> nil
03169  *
03170  * @par Lua 示例
03171  * modbusSetOutputSignal1("Modbus_0", {0, 1, 2})
03172  *
03173  * @par JSON-RPC 请求示例
03174  * {"jsonrpc": "2.0", "method": "RegisterControl.modbusSetOutputSignal1", "params": ["Modbus_0", [0, 1, 2]], "id": 1}
03175  *
03176  * @par JSON-RPC 响应示例
03177  * {"id": 1, "jsonrpc": "2.0", "result": 0}
03178  * \endchinese
03179  * \english
03180  * Sets multiple consecutive output register signals starting from the given
03181  * name to the given values.
03182  *
03183  * @param signal_name A string identifying the starting output register
03184  * signal that in advance has been added.
03185  * @param values An array of integers where each element must be a valid
03186  * word (0-65535), corresponding to multiple consecutive signals
03187  * @return
03188  *
03189  * @par Python interface prototype
03190  * modbusSetOutputSignal1(self: pyaubo_sdk.RegisterControl, arg0: str, arg1:
03191  * list[int]) -> int
03192  *
03193  * @par Lua interface prototype
03194  * modbusSetOutputSignal1(signal_name: string, value: table) -> nil
03195  *
03196  * @par Lua example
03197  * modbusSetOutputSignal1("Modbus_0", {0, 1, 2})
03198  *
03199  * @par JSON-RPC request example
03200  * {"jsonrpc": "2.0", "method": "RegisterControl.modbusSetOutputSignal1", "params": ["Modbus_0", [0, 1, 2]], "id": 1}
03201  *
03202  * @par JSON-RPC response example
03203  * {"id": 1, "jsonrpc": "2.0", "result": 0}
03204  * \endenglish
03205  */
03206 int modbusSetOutputSignal1(const std::string &signal_name,
03207                           const std::vector<uint16_t> &values);
03208
03209 /**
03210  * \chinese
03211  * 将指定名称的输出寄存器信号设置为给定的值, 并且带超时判断
03212  *
03213  * @param signal_name 提前被添加的输出寄存器信号
03214  * @param value 必须是有效的整数, 范围是 0-65535
03215  * @param timeout 超时时间, 单位秒
03216  * @return
03217  *
03218  * @par Python 函数原型
03219  * modbusSetOutputSignalWithTimeout(self: pyaubo_sdk.RegisterControl, arg0:
03220  * str, arg1: int, arg2: double) -> int
03221  *
03222  * @par Lua 函数原型
03223  * modbusSetOutputSignalWithTimeout(signal_name: string, value: number,
03224  * timeout: number) -> nil
03225  *
03226  * @par Lua 示例
03227  * modbusSetOutputSignalWithTimeout("Modbus_0", 0, 0.5)
03228  *
03229  * @par JSON-RPC 请求示例
03230  * {"jsonrpc": "2.0", "method": "RegisterControl.modbusSetOutputSignalWithTimeout", "params":
03231  * ["Modbus_0", 0, 0.5], "id": 1}
03231  *

```

```

03232     * @par JSON-RPC 响应示例
03233     * {"id":1,"jsonrpc":"2.0","result":0}
03234     * \endchinese
03235     * \english
03236     * Sets the output register signal identified by the given name to the
03237     * given, with timeout value.
03238     *
03239     * @param signal_name A string identifying an output register signal that in
03240     * advance has been added.
03241     * @param value An integer which must be a valid word (0-65535)
03242     * @param timeout seconds
03243     * @return
03244     *
03245     * @par Python interface prototype
03246     * modbusSetOutputSignalWithTimeout(self: pyaubo_sdk.RegisterControl, arg0:
03247     * str, arg1: int, arg2: timeout) -> int
03248     *
03249     * @par Lua interface prototype
03250     * modbusSetOutputSignalWithTimeout(signal_name: string, value: number,
03251     * timeout: number) -> nil
03252     *
03253     * @par Lua example
03254     * modbusSetOutputSignalWithTimeout("Modbus_0",0,0.5)
03255     *
03256     * @par JSON-RPC request example
03257     * {"jsonrpc":"2.0","method":"RegisterControl.modbusSetOutputSignalWithTimeout","params":
03258     * [{"Modbus_0",0,0.5},"id":1]}
03259     * @par JSON-RPC response example
03260     * {"id":1,"jsonrpc":"2.0","result":0}
03261     * \endenglish
03262     */
03263 int modbusSetOutputSignalWithTimeout(const std::string &signal_name,
03264                                     uint16_t value, double timeout);
03265
03266 /**
03267     * @ingroup RegisterControl
03268     * \chinese
03269     * 设置 modbus 信号输出脉冲（仅支持线圈输出类型）
03270     *
03271     * @param signal_name: 提前被添加的输出寄存器信号
03272     * @param value: 必须有效的整数，范围是 0-65535
03273     * @param duration: 信号持续时间，单位为秒
03274     * @return
03275     *
03276     * @par Python 函数原型
03277     * modbusSetOutputSignalPulse(self: pyaubo_sdk.RegisterControl, arg0: str,
03278     * arg1: int, arg2 double) -> int
03279     *
03280     * @par Lua 函数原型
03281     * modbusSetOutputSignalPulse(signal_name: string, value: number, duration:
03282     * number) -> nil
03283     *
03284     * @par Lua 示例
03285     * modbusSetOutputSignalPulse("Modbus_0",1,0.5)
03286     *
03287     * @par JSON-RPC 请求示例
03288     * {"jsonrpc":"2.0","method":"RegisterControl.modbusSetOutputSignalPulse","params":["Modbus_0",1,0.5],"id":1}
03289     *
03290     * @par JSON-RPC 响应示例
03291     * {"id":1,"jsonrpc":"2.0","result":0}
03292     *
03293     * \endchinese
03294     * \english
03295     * Set modbus signal output pulse (only supports coil output type)
03296     *
03297     * @param signal_name: A string identifying an output register signal that
03298     * has been added in advance.
03299     * @param value: An integer which must be a valid word (0-65535)
03300     * @param duration: Duration of the signal, in seconds
03301     * @return
03302     *
03303     * @par Python interface prototype
03304     * modbusSetOutputSignalPulse(self: pyaubo_sdk.RegisterControl, arg0: str,
03305     * arg1: int, arg2: double) -> int
03306     *
03307     * @par Lua interface prototype
03308     * modbusSetOutputSignalPulse(signal_name: string, value: number, duration:
03309     * number) -> nil
03310     *
03311     * @par Lua example
03312     * modbusSetOutputSignalPulse("Modbus_0",1,0.5)
03313     *
03314     * @par JSON-RPC request example
03315     * {"jsonrpc":"2.0","method":"RegisterControl.modbusSetOutputSignalPulse","params":["Modbus_0",1,0.5],"id":1}
03316     *
03317     * @par JSON-RPC response example

```



```

03318     * {"id":1,"jsonrpc":"2.0","result":0}
03319     *
03320     * \endenglish
03321     */
03322 int modbusSetOutputSignalPulse(const std::string &signal_name,
03323                               uint16_t value, double duration);
03324
03325 /**
03326  * @ingroup RegisterControl
03327  * \chinese
03328  * 设置机器人向 Modbus 控制器发送请求的频率，用于读取或写入信号值
03329  *
03330  * @param signal_name 提前被添加的输出数字信号
03331  * @param update_frequency 更新频率（以赫兹为单位），范围是 0-125
03332  * @return
03333  *
03334  * @par Python 函数原型
03335  * modbusSetSignalUpdateFrequency(self: pyaubo_sdk.RegisterControl, arg0:
03336  * str, arg1: int) -> int
03337  *
03338  * @par Lua 函数原型
03339  * modbusSetSignalUpdateFrequency(signal_name: string, update_frequency:
03340  * number) -> nil
03341  *
03342  * @par Lua 示例
03343  * modbusSetSignalUpdateFrequency("Modbus_0",1)
03344  *
03345  * @par JSON-RPC 请求示例
03346  * {"jsonrpc":"2.0","method":"RegisterControl.modbusSetSignalUpdateFrequency","params":["Modbus_0",1],"id":1}
03347  *
03348  * @par JSON-RPC 响应示例
03349  * {"id":1,"jsonrpc":"2.0","result":0}
03350  * \endchinese
03351  * \english
03352  * Sets the frequency with which the robot will send requests to the Modbus
03353  * controller to either read or write the signal value.
03354  *
03355  * @param signal_name A string identifying an output digital signal that
03356  * in advance has been added.
03357  * @param update_frequency An integer in the range 0-125 specifying the
03358  * update frequency in Hz.
03359  * @return
03360  *
03361  * @par Python interface prototype
03362  * modbusSetSignalUpdateFrequency(self: pyaubo_sdk.RegisterControl, arg0:
03363  * str, arg1: int) -> int
03364  *
03365  * @par Lua interface prototype
03366  * modbusSetSignalUpdateFrequency(signal_name: string, update_frequency:
03367  * number) -> nil
03368  *
03369  * @par Lua example
03370  * modbusSetSignalUpdateFrequency("Modbus_0",1)
03371  *
03372  * @par JSON-RPC request example
03373  * {"jsonrpc":"2.0","method":"RegisterControl.modbusSetSignalUpdateFrequency","params":["Modbus_0",1],"id":1}
03374  *
03375  * @par JSON-RPC response example
03376  * {"id":1,"jsonrpc":"2.0","result":0}
03377  * \endenglish
03378  */
03379 int modbusSetSignalUpdateFrequency(const std::string &signal_name,
03380                                   int update_frequency);
03381
03382 /**
03383  * @ingroup RegisterControl
03384  * \chinese
03385  * 获取指定 modbus 信号索引，从 0 开始，不能存在则返回-1
03386  *
03387  * @param signal_name
03388  * @return
03389  *
03390  * @par JSON-RPC 请求示例
03391  * {"jsonrpc":"2.0","method":"RegisterControl.modbusGetSignalIndex","params":["Modbus_0"],"id":1}
03392  *
03393  * @par JSON-RPC 响应示例
03394  * {"id":1,"jsonrpc":"2.0","result":0}
03395  *
03396  * \endchinese
03397  * \english
03398  * Get the index of the specified modbus signal, starting from 0. Returns -1
03399  * if it does not exist.
03400  *
03401  * @param signal_name
03402  * @return
03403  *
03404  * @par JSON-RPC request example

```

```

03405     * {"jsonrpc": "2.0", "method": "RegisterControl.modbusGetSignalIndex", "params": ["Modbus_0"], "id": 1}
03406     *
03407     * @par JSON-RPC response example
03408     * {"id": 1, "jsonrpc": "2.0", "result": 0}
03409     *
03410     * \endenglish
03411     */
03412 int modbusGetSignalIndex(const std::string &signal_name);
03413
03414 /**
03415     * @ingroup RegisterControl
03416     * \chinese
03417     * 获取指定 modbus 信号的错误状态
03418     *
03419     * @param signal_name
03420     * @return 返回错误代码 ModbusErrorNum
03421     *
03422     * @par JSON-RPC 请求示例
03423     * {"jsonrpc": "2.0", "method": "RegisterControl.modbusGetSignalError", "params": ["Modbus_0"], "id": 1}
03424     *
03425     * @par JSON-RPC 响应示例
03426     * {"id": 1, "jsonrpc": "2.0", "result": 6}
03427     * \endchinese
03428     * \english
03429     * Get the error status of the specified modbus signal
03430     *
03431     * @param signal_name
03432     * @return Returns error code ModbusErrorNum
03433     *
03434     * @par JSON-RPC request example
03435     * {"jsonrpc": "2.0", "method": "RegisterControl.modbusGetSignalError", "params": ["Modbus_0"], "id": 1}
03436     *
03437     * @par JSON-RPC response example
03438     * {"id": 1, "jsonrpc": "2.0", "result": 6}
03439     * \endenglish
03440     */
03441 int modbusGetSignalError(const std::string &signal_name);
03442
03443 /**
03444     * @ingroup RegisterControl
03445     * \chinese
03446     * 获取指定 modbus 设备的连接状态
03447     *
03448     * @param device_name
03449     * 设备名是 TCP 格式, "ip:port", 例如: "127.0.0.1:502" \n
03450     * 设备名是 RTU 格式, "serial_port", 例如: "/dev/ttyUSB0" \n
03451     *
03452     * @return
03453     * 0: 表示设备处于连接状态
03454     * -1: 表示设备不存在
03455     * -2: 表示设备处于断开状态
03456     *
03457     * @par JSON-RPC 请求示例
03458     * {"jsonrpc": "2.0", "method": "RegisterControl.getModbusDeviceStatus", "params": ["172.16.26.248:502"], "id": 1}
03459     *
03460     * @par JSON-RPC 响应示例
03461     * {"id": 1, "jsonrpc": "2.0", "result": 0}
03462     *
03463     * \endchinese
03464     * \english
03465     * Get the connection status of the specified modbus device
03466     *
03467     * @param device_name
03468     * Device name in TCP format: "ip:port", e.g. "127.0.0.1:502" \n
03469     * Device name in RTU format: "serial_port", e.g. "/dev/ttyUSB0" \n
03470     *
03471     * @return
03472     * 0: Device is connected
03473     * -1: Device does not exist
03474     * -2: Device is disconnected
03475     *
03476     * @par JSON-RPC request example
03477     * {"jsonrpc": "2.0", "method": "RegisterControl.getModbusDeviceStatus", "params": ["172.16.26.248:502"], "id": 1}
03478     *
03479     * @par JSON-RPC response example
03480     * {"id": 1, "jsonrpc": "2.0", "result": 0}
03481     *
03482     * \endenglish
03483     */
03484 int getModbusDeviceStatus(const std::string &device_name);
03485
03486 /**
03487     * @ingroup RegisterControl
03488     * \chinese
03489     * 将某个 modbus 寄存器信号作为编码器
03490     *
03491     * @param encoder_id 不能为 0

```



```

03492     * @param signal_name modbus 信号名字, 必须为寄存器类型
03493     * @return
03494     * \endchinese
03495     * \english
03496     * Use a modbus register signal as an encoder
03497     *
03498     * @param encoder_id Must not be 0
03499     * @param signal_name Name of the modbus signal, must be of register type
03500     * @return
03501     * \endenglish
03502     */
03503     int addModbusEncoder(int encoder_id, int range_id,
03504                         const std::string &signal_name);
03505
03506     /**
03507     * @ingroup RegisterControl
03508     * \chinese
03509     * 添加 Int32 寄存器的虚拟编码器
03510     *
03511     * @param encoder_id 编码器 ID
03512     * @param range_id 范围 ID
03513     * @param key 变量名
03514     * @return
03515     * \endchinese
03516     * \english
03517     * Add a virtual encoder for an Int32 register
03518     *
03519     * @param encoder_id Encoder ID
03520     * @param range_id Range ID
03521     * @param key Variable name
03522     * @return
03523     * \endenglish
03524     */
03525     int addInt32RegEncoder(int encoder_id, int range_id,
03526                          const std::string &key);
03527
03528     /**
03529     * @ingroup RegisterControl
03530     * \chinese
03531     * 删除虚拟编码器
03532     *
03533     * @param encoder_id
03534     * @return
03535     * \endchinese
03536     * \english
03537     * Delete virtual encoder
03538     *
03539     * @param encoder_id
03540     * @return
03541     * \endenglish
03542     */
03543     int deleteVirtualEncoder(int encoder_id);
03544
03545     protected:
03546     void *d_;
03547 };
03548 using RegisterControlPtr = std::shared_ptr<RegisterControl>;
03549
03550 } // namespace common_interface
03551 } // namespace arcs
03552
03553 #endif // AUBO_SDK_REGISTER_CONTROL_INTERFACE_H

```

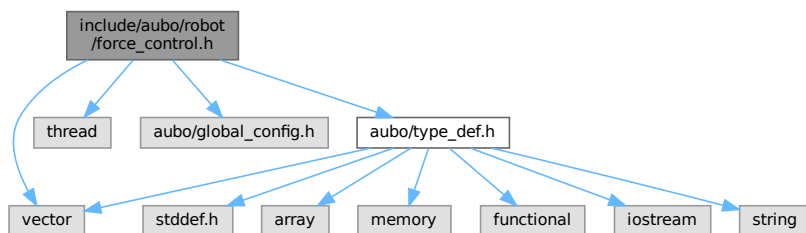
12.20 include/aubo/robot/force_control.h 文件参考

```

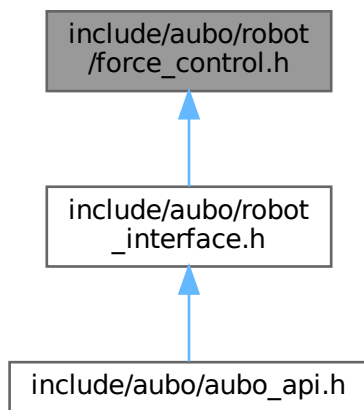
#include <vector>
#include <thread>
#include <aubo/global_config.h>
#include <aubo/type_def.h>

```

force_control.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::common_interface::ForceControl`

命名空间

- namespace `arcs`
- namespace `arcs::common_interface`

类型定义

- using `arcs::common_interface::ForceControlPtr = std::shared_ptr<ForceControl>`

12.20.1 详细描述

力控接口

力控的限制当机器人处于力控模式时，以下功能不可用：

- 碰撞检测（选项 613-1）
- SoftMove（选项 885-1）
- 跟踪功能，如输送带跟踪（选项 606-1）、光学跟踪（6601）和焊接引导（815-2）
- 传感器同步或模拟同步

- 世界区域 (选项 608-1)
- 独立轴 (选项 610-1)
- 路径偏移 (选项 612-1)
- 弧焊选项
- PickMaster 选项
- 关节软伺服 (指令 SoftAct)
- 当机器人以 MultiMove 协调模式 (选项 604-1) 运行时, 不能激活力控。
- 如果力控与 SafeMove (选项 810-2) 或电子位置开关 (选项 810-1) 一起使用, 必须使用操作安全范围功能。请参阅这些选项的相关手册。
- RAPID 指令如 FCAct、FCDeact、FCConditionWaitWhile 和 FCRefStop 只能从运动任务的普通级别调用。

应用: 抛光、打磨、清洁

FC Pressure

设置轨迹坐标系的 z 方向为力控轴, spring 设置为 0

在还没接触前设置输出力为 0, spring 设置为固定值 (根据 vel 确定)

离开接触面: 设置输出力为 0, spring 设置为固定值

活塞装配

Forward clutch hub

设置力控终止模式

基于末端力传感器的拖动示教

spring = 0; force_ref = 0; 参考轨迹点任意

在文件 [force_control.h](#) 中定义。

12.21 force_control.h

[浏览该文件的文档。](#)

```
00001 /**
00002  * \chinese
00003  * @file force_control.h
00004  * @brief 力控接口
00005  *
00006  * 力控的限制
00007  * 当机器人处于力控模式时, 以下功能不可用:
00008  *
00009  * • 碰撞检测 (选项 613-1) \n
00010  * • SoftMove (选项 885-1) \n
00011  * • 跟踪功能, 如输送带跟踪 (选项 606-1)、光学跟踪 (6601) 和焊接引导 (815-2) \n
00012  * • 传感器同步或模拟同步 \n
00013  * • 世界区域 (选项 608-1) \n
00014  * • 独立轴 (选项 610-1) \n
00015  * • 路径偏移 (选项 612-1) \n
00016  * • 弧焊选项 \n
00017  * • PickMaster 选项 \n
00018  * • 关节软伺服 (指令 SoftAct) \n
00019  * • 当机器人以 MultiMove 协调模式 (选项 604-1) 运行时, 不能激活力控。 \n
00020  * • 如果力控与 SafeMove (选项 810-2) 或电子位置开关 (选项 810-1) 一起使用, 必须使用操作安全范围功能。请参阅这些选项的相关手册。 \n
00021  * • RAPID 指令如 FCAct、FCDeact、FCConditionWaitWhile 和 FCRefStop 只能从运动任务的普通级别调用。
00022  *
00023  * 应用: 抛光、打磨、清洁 \n
00024  * FC Pressure\n
00025  * 设置轨迹坐标系的 z 方向为力控轴, spring 设置为 0\n
00026  * 在还没接触前设置输出力为 0, spring 设置为固定值 (根据 vel 确定) \n
00027  * 离开接触面: 设置输出力为 0, spring 设置为固定值 \n
00028  *
00029  * 活塞装配 \n
00030  * Forward clutch hub\n
00031  * 设置力控终止模式
00032  *
00033  * 基于末端力传感器的拖动示教 \n
00034  * spring = 0; force_ref = 0; 参考轨迹点任意
00035  * \endchinese
00036  * \english
00037  * @file force_control.h
00038  * @brief Force control interface
00039  *
00040  * Force control limitations
00041  * When the robot is force controlled, the following functionality is not accessible:
00042  *
00043  * • Collision Detection (option 613-1) \n
00044  * • SoftMove (option 885-1) \n
00045  * • Tracking functionality like Conveyor Tracking (option 606-1), Optical Tracking (6601) and Weld Guide (815-2) \n
00046  * • Sensor Synchronization or Analog Synchronization \n
```

```

00047 * • World Zones (option 608-1) \n
00048 * • Independent Axes (option 610-1) \n
00049 * • Path Offset (option 612-1) \n
00050 * • Arc options \n
00051 * • PickMaster options \n
00052 * • Joint soft servo (instruction SoftAct) \n
00053 * • Force Control cannot be activated when the robot is running in MultiMove Coordinated mode (option 604-1). \n
00054 * • If Force Control is used together with SafeMove (option 810-2) or Electronic Position Switches (option 810-1),
the function Operational Safety Range must be used. See the respective manual for these options. \n
00055 * • RAPID instructions such as FCAct, FCDeact, FCConditionWaitWhile and FCRefStop can only be called from normal
level in a motion task.
00056 *
00057 * Applications: polishing, grinding, cleaning \n
00058 * FC Pressure \n
00059 * Set the z direction of the trajectory coordinate system as the force control axis, set spring to 0 \n
00060 * Before contact, set output force to 0, spring to a fixed value (determined by vel) \n
00061 * Leaving the contact surface: set output force to 0, spring to a fixed value \n
00062 *
00063 * Piston assembly \n
00064 * Forward clutch hub \n
00065 * Set force control termination mode
00066 *
00067 * Drag teaching based on end force sensor \n
00068 * spring = 0; force_ref = 0; reference trajectory point arbitrary
00069 * \endenglish
00070 */
00071 #ifndef AUBO_SDK_FORCE_CONTROL_INTERFACE_H
00072 #define AUBO_SDK_FORCE_CONTROL_INTERFACE_H
00073
00074 #include <vector>
00075 #include <thread>
00076
00077 #include <aubo/global_config.h>
00078 #include <aubo/type_def.h>
00079
00080 namespace arcs {
00081 namespace common_interface {
00082
00083 /**
00084 * @defgroup ForceControl ForceControl(力控模块)
00085 * @ingroup RobotInterface
00086 * \chinese
00087
00088 * 力控接口抽象类
00089 * \endchinese
00090 * \english
00091 * Abstract class for force control interface
00092 * \endenglish
00093 */
00094 class ARCS_ABI_EXPORT ForceControl
00095 {
00096 public:
00097     ForceControl();
00098     virtual ~ForceControl();
00099
00100     /**
00101      * @ingroup ForceControl
00102      * \chinese
00103      * 使能力控。
00104      * fcEnable 被用于使能力控。在力控被使能的同时,
00105      * fcEnable 用于定义力控的坐标系, 并调整力和力矩的阻尼。
00106      * 如果在 fcEnable 中未指定坐标系,
00107      * 则会创建一个默认的力控制坐标系, 其方向与工作对象坐标系相同。
00108      * 所有力控制监管功能都被 fcEnable 激活。
00109      *
00110      * @return 成功返回 0; 失败返回错误码
00111      * AUBO_BUSY
00112      * AUBO_BAD_STATE
00113      * -AUBO_BAD_STATE
00114      *
00115      * @throws arcs::common_interface::AuboException
00116      *
00117      * @par Python 函数原型
00118      * fcEnable(self: pyaubo_sdk.ForceControl) -> int
00119      *
00120      * @par Lua 函数原型
00121      * fcEnable() -> nil
00122      *
00123      * @par Lua 示例
00124      * fcEnable()
00125      *
00126      * @par JSON-RPC 请求示例
00127      * {"jsonrpc": "2.0", "method": "rob1.ForceControl.fcEnable", "params": [], "id": 1}
00128      *
00129      * @par JSON-RPC 响应示例
00130      * {"id": 1, "jsonrpc": "2.0", "result": 0}
00131      *

```

```

00132     * \endchinese
00133     *
00134     * \english
00135     * Start force control
00136     *
00137     * fcEnable is used to enable Force Control. At the same time as Force
00138     * Control is enabled, fcEnable is used to define the coordinate system
00139     * for Force Control, and tune the force and torque damping. If a coordinate
00140     * system is not specified in fcEnable a default force control coordinate
00141     * system is created with the same orientation as the work object coordinate
00142     * system. All Force Control supervisions are activated by fcEnable.
00143     *
00144     *
00145     * @return Return 0 if succeeded; return error code if failed
00146     * AUBO_BUSY
00147     * AUBO_BAD_STATE
00148     * -AUBO_BAD_STATE
00149     *
00150     * @throws arcs::common_interface::AuboException
00151     *
00152     * @par Python Function Prototype
00153     * fcEnable(self: pyaubo_sdk.ForceControl) -> int
00154     *
00155     * @par Lua Function Prototype
00156     * fcEnable() -> nil
00157     *
00158     * @par Lua example
00159     * fcEnable()
00160     *
00161     * @par JSON-RPC Request example
00162     * {"jsonrpc": "2.0", "method": "rob1.ForceControl.fcEnable", "params": [], "id": 1}
00163     *
00164     * @par JSON-RPC Response example
00165     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00166     * \endenglish
00167     */
00168
00169 int fcEnable();
00170
00171 /**
00172  * @ingroup ForceControl
00173  * \chinese
00174  * 失能力控。
00175  * fcDisable 被用于失能力控。在成功失能力控之后，机器人将回到位置控制模式。
00176  *
00177  * @return 成功返回 0；失败返回错误码
00178  * AUBO_BUSY
00179  * AUBO_BAD_STATE
00180  * -AUBO_BAD_STATE
00181  *
00182  * @throws arcs::common_interface::AuboException
00183  *
00184  * @par Python 函数原型
00185  * fcDisable(self: pyaubo_sdk.ForceControl) -> int
00186  *
00187  * @par Lua 函数原型
00188  * fcDisable() -> nil
00189  *
00190  * @par Lua 示例
00191  * fcDisable()
00192  *
00193  * @par JSON-RPC 请求示例
00194  * {"jsonrpc": "2.0", "method": "rob1.ForceControl.fcDisable", "params": [], "id": 1}
00195  *
00196  * @par JSON-RPC 响应示例
00197  * {"id": 1, "jsonrpc": "2.0", "result": 0}
00198  * \endchinese
00199  *
00200  * \english
00201  * End force control
00202  *
00203  * fcDisable is used to disable Force Control. After a successful
00204  * deactivation the robot is back in position control.
00205  *
00206  * @return Return 0 if succeeded; return error code if failed
00207  * AUBO_BUSY
00208  * AUBO_BAD_STATE
00209  * -AUBO_BAD_STATE
00210  *
00211  * @throws arcs::common_interface::AuboException
00212  *
00213  * @par Python Function Prototype
00214  * fcDisable(self: pyaubo_sdk.ForceControl) -> int
00215  *
00216  * @par Lua Function Prototype
00217  * fcDisable() -> nil
00218  *

```

```

00219     * @par Lua example
00220     * fcDisable()
00221     *
00222     * @par JSON-RPC Request example
00223     * {"jsonrpc": "2.0", "method": "rob1.ForceControl.fcDisable", "params": [], "id": 1}
00224     *
00225     * @par JSON-RPC Response example
00226     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00227     * \endenglish
00228     */
00229
00230 int fcDisable();
00231
00232 /**
00233     * @ingroup ForceControl
00234     * \chinese
00235     * 判断力控是否被使能
00236     *
00237     * @return 使能返回 true, 失能返回 false
00238     *
00239     * @throws arcs::common_interface::AuboException
00240     *
00241     * @par Python 函数原型
00242     * isEnabled(self: pyaubo_sdk.ForceControl) -> bool
00243     *
00244     * @par Lua 函数原型
00245     * isEnabled() -> boolean
00246     *
00247     * @par Lua 示例
00248     * Fc_status = isEnabled()
00249     *
00250     * @par JSON-RPC 请求示例
00251     * {"jsonrpc": "2.0", "method": "rob1.ForceControl.isEnabled", "params": [], "id": 1}
00252     *
00253     * @par JSON-RPC 响应示例
00254     * {"id": 1, "jsonrpc": "2.0", "result": false}
00255     * \endchinese
00256     *
00257     * \english
00258     * Check if force control is enabled
00259     *
00260     * @return Returns true if enabled, false if disabled
00261     *
00262     * @throws arcs::common_interface::AuboException
00263     *
00264     * @par Python Function Prototype
00265     * isEnabled(self: pyaubo_sdk.ForceControl) -> bool
00266     *
00267     * @par Lua Function Prototype
00268     * isEnabled() -> boolean
00269     *
00270     * @par Lua example
00271     * Fc_status = isEnabled()
00272     *
00273     * @par JSON-RPC Request example
00274     * {"jsonrpc": "2.0", "method": "rob1.ForceControl.isEnabled", "params": [], "id": 1}
00275     *
00276     * @par JSON-RPC Response example
00277     * {"id": 1, "jsonrpc": "2.0", "result": false}
00278     * \endenglish
00279     */
00280 bool isEnabled();
00281
00282 /**
00283     * @ingroup ForceControl
00284     * \chinese
00285     * 设置力控参考（目标）值
00286     *
00287     * @param feature: 参考几何特征，用于生成力控参考坐标系
00288     * @param compliance: 柔性轴（方向）选择
00289     * @param wrench: 目标力/力矩
00290     * @param limits: 速度限制
00291     * @param type: 力控参考坐标系类型
00292     *
00293     * 使用说明：
00294     * 1. 基坐标系：
00295     *     feature = {0,0,0,0,0,0}
00296     *     type = TaskFrameType::NONE
00297     *
00298     * 2. 法兰坐标系：
00299     *     feature = {0,0,0,0,0,0}
00300     *     type = TaskFrameType::TOOL_FORCE
00301     *
00302     * 3. TCP 坐标系：
00303     *     feature = tcp_offset
00304     *     type = TaskFrameType::TOOL_FORCE
00305     *

```

```

00306 * 4. 用户坐标系 (FRAME_FORCE):
00307 *   type = TaskFrameType::FRAME_FORCE
00308 *   feature 设为用户定义的参考坐标, 例如 getTcpPose() 表示以当前 TCP 坐标作为力控坐标系。
00309 *
00310 * @return 成功返回 0; 失败返回错误码
00311 * AUBO_BUSY
00312 * AUBO_BAD_STATE
00313 * -AUBO_INVL_ARGUMENT
00314 * -AUBO_BAD_STATE
00315 *
00316 * @throws arcs::common_interface::AuboException
00317 *
00318 * @par Python 函数原型
00319 * setTargetForce(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
00320 * List[bool], arg2: List[float], arg3: List[float], arg4:
00321 * arcs::common_interface::TaskFrameType) -> int
00322 *
00323 * @par Lua 函数原型
00324 * setTargetForce(feature: table, compliance: table, wrench: table, limits:
00325 * table, type: number) -> nil
00326 *
00327 * @par Lua 示例
00328 * setTargetForce({0,0,0,0,0,0},{true,false,false,false,false,false},{10,0,0,0,0,0},{0,0,0,0,0,0},4)
00329 *
00330 * \endchinese
00331 * \english
00332 * Set force control reference (target) value
00333 *
00334 * @param feature: Reference geometric feature for generating force control reference frame
00335 * @param compliance: Compliance axis (direction) selection
00336 * @param wrench: Target force/torque
00337 * @param limits: Velocity limits
00338 * @param type: Force control reference frame type
00339 *
00340 * Usage Examples:
00341 * 1. Base Coordinate Frame:
00342 *   feature = {0,0,0,0,0,0}
00343 *   type = TaskFrameType::NONE
00344 *
00345 * 2. Flange Coordinate Frame:
00346 *   feature = {0,0,0,0,0,0}
00347 *   type = TaskFrameType::TOOL_FORCE
00348 *
00349 * 3. TCP Coordinate Frame:
00350 *   feature = tcp_offset
00351 *   type = TaskFrameType::TOOL_FORCE
00352 *
00353 * 4. User Coordinate Frame (FRAME_FORCE):
00354 *   type = TaskFrameType::FRAME_FORCE
00355 *   feature should be the user-defined reference frame,
00356 *   for example, getTcpPose() means setting the force control frame to current TCP pose.
00357 *
00358 * @return Return 0 if succeeded; return error code if failed
00359 * AUBO_BUSY
00360 * AUBO_BAD_STATE
00361 * -AUBO_INVL_ARGUMENT
00362 * -AUBO_BAD_STATE
00363 *
00364 * @throws arcs::common_interface::AuboException
00365 *
00366 * @par Python Function Prototype
00367 * setTargetForce(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
00368 * List[bool], arg2: List[float], arg3: List[float], arg4:
00369 * arcs::common_interface::TaskFrameType) -> int
00370 *
00371 * @par Lua Function Prototype
00372 * setTargetForce(feature: table, compliance: table, wrench: table, limits:
00373 * table, type: number) -> nil
00374 *
00375 * @par Lua example
00376 * setTargetForce({0,0,0,0,0,0},{true,false,false,false,false,false},{10,0,0,0,0,0},{0,0,0,0,0,0},4)
00377 *
00378 * \endenglish
00379 */
00380 int setTargetForce(const std::vector<double> &feature,
00381                   const std::vector<bool> &compliance,
00382                   const std::vector<double> &wrench,
00383                   const std::vector<double> &limits,
00384                   TaskFrameType type = TaskFrameType::FRAME_FORCE);
00385
00386 /**
00387 * @ingroup ForceControl
00388 * \chinese
00389 * 设置力控动力学模型
00390 *
00391 * @param env_stiff: 环境刚度, 表示为接触轴方向上的工件刚度, 取值范围 [0, 1], 默认为 0
00392 * @param damp_scale: 表征阻尼水平的参数, 取值范围 [0, 1], 默认为 0.5

```

```

00393     * @param stiff_scale: 表征软硬程度的参数, 取值范围 [0, 1], 默认为 0.5
00394     *
00395     * @return 成功返回 0; 失败返回错误码
00396     * AUBO_BUSY
00397     * AUBO_BAD_STATE
00398     * -AUBO_INVL_ARGUMENT
00399     * -AUBO_BAD_STATE
00400     *
00401     * @throws arcs::common_interface::AuboException
00402     *
00403     * @par Python 函数原型
00404     * setDynamicModel1(self: pyaubo_sdk.ForceControl, arg0: List[float],
00405     * arg1: List[float], arg2: List[float]) -> int
00406     *
00407     * @par Lua 函数原型
00408     * setDynamicModel1(env_stiff: table, damp_scale: table, stiff_scale:
00409     * table) -> nil
00410     *
00411     * @par Lua 示例
00412     * setDynamicModel1({0,0,0,0,0,0},{0.5,0.5,0.5,0.5,0.5,0.5},{0.5,0.5,0.5,0.5,0.5,0.5})
00413     *
00414     * \endchinese
00415     * \english
00416     * Set force control dynamics model
00417     *
00418     * @param env_stiff: Environment stiffness, representing the workpiece stiffness in the contact axis direction,
00419     range [0, 1], default 0
00420     * @param damp_scale: Parameter representing damping level, range [0, 1], default 0.5
00421     * @param stiff_scale: Parameter representing stiffness level, range [0, 1], default 0.5
00422     *
00423     * @return Return 0 if succeeded; return error code if failed
00424     * AUBO_BUSY
00425     * AUBO_BAD_STATE
00426     * -AUBO_INVL_ARGUMENT
00427     * -AUBO_BAD_STATE
00428     *
00429     * @throws arcs::common_interface::AuboException
00430     *
00431     * @par Python Function Prototype
00432     * setDynamicModel1(self: pyaubo_sdk.ForceControl, arg0: List[float],
00433     * arg1: List[float], arg2: List[float]) -> int
00434     *
00435     * @par Lua Function Prototype
00436     * setDynamicModel1(env_stiff: table, damp_scale: table, stiff_scale:
00437     * table) -> nil
00438     *
00439     * @par Lua example
00440     * setDynamicModel1({0,0,0,0,0,0},{0.5,0.5,0.5,0.5,0.5,0.5},{0.5,0.5,0.5,0.5,0.5,0.5})
00441     *
00442     * \endenglish
00443     */
00444     int setDynamicModel1(const std::vector<double> &env_stiff,
00445                         const std::vector<double> &damp_scale,
00446                         const std::vector<double> &stiff_scale);
00447
00448     /**
00449     * @ingroup ForceControl
00450     * \chinese
00451     * 计算力控动力学模型
00452     *
00453     * @param env_stiff: 环境刚度, 表示为接触轴方向上的工件刚度, 取值范围 [0, 1], 默认为 0
00454     * @param damp_scale: 表征阻尼水平的参数, 取值范围 [0, 1], 默认为 0.5
00455     * @param stiff_scale: 表征软硬程度的参数, 取值范围 [0, 1], 默认为 0.5
00456     *
00457     * @return 力控动力学模型 MDK
00458     * AUBO_BUSY
00459     * AUBO_BAD_STATE
00460     * -AUBO_INVL_ARGUMENT
00461     * -AUBO_BAD_STATE
00462     *
00463     * @throws arcs::common_interface::AuboException
00464     *
00465     * @par Python 函数原型
00466     * fcCalDynamicModel(self: pyaubo_sdk.ForceControl, arg0: List[float],
00467     * arg1: List[float], arg2: List[float]) -> Tuple[List[float], List[float], List[float]]
00468     *
00469     * @par Lua 函数原型
00470     * fcCalDynamicModel(env_stiff: table, damp_scale: table, stiff_scale: table) -> table
00471     *
00472     * @par Lua 示例
00473     * fc_table = fcCalDynamicModel({0,0,0,0,0,0},{0.5,0.5,0.5,0.5,0.5,0.5},{0.5,0.5,0.5,0.5,0.5,0.5})
00474     *
00475     * \endchinese
00476     * \english
00477     * Calculate force control dynamics model
00478     *
00479     * @param env_stiff: Environment stiffness, representing the workpiece stiffness in the contact axis direction,

```



```

range [0, 1], default 0
00479 * @param damp_scale: Parameter representing damping level, range [0, 1], default 0.5
00480 * @param stiff_scale: Parameter representing stiffness level, range [0, 1], default 0.5
00481 *
00482 * @return Force control dynamics model MDK
00483 * AUBO_BUSY
00484 * AUBO_BAD_STATE
00485 * -AUBO_INVL_ARGUMENT
00486 * -AUBO_BAD_STATE
00487 *
00488 * @throws arcs::common_interface::AuboException
00489 *
00490 * @par Python Function Prototype
00491 * fcCalDynamicModel(self: pyaubo_sdk.ForceControl, arg0: List[float],
00492 * arg1: List[float], arg2: List[float]) -> Tuple[List[float], List[float], List[float]]
00493 *
00494 * @par Lua Function Prototype
00495 * fcCalDynamicModel(env_stiff: table, damp_scale: table, stiff_scale: table) -> table
00496 *
00497 * @par Lua example
00498 * fc_table = fcCalDynamicModel({0,0,0,0,0,0},{0.5,0.5,0.5,0.5,0.5,0.5},{0.5,0.5,0.5,0.5,0.5,0.5})
00499 *
00500 * \endenglish
00501 */
00502 DynamicsModel fcCalDynamicModel(const std::vector<double> &env_stiff,
00503                                 const std::vector<double> &damp_scale,
00504                                 const std::vector<double> &stiff_scale);
00505
00506 /**
00507 * @ingroup ForceControl
00508 * \chinese
00509 * 设置力控搜孔场景下的动力学模型
00510 *
00511 * @param damp_scale: 表征阻尼水平的参数, 取值范围 [0, 1], 默认为 0.5
00512 * @param stiff_scale: 表征软硬程度的参数, 取值范围 [0, 1], 默认为 0.5
00513 *
00514 * @return 成功返回 0; 失败返回错误码
00515 * AUBO_BUSY
00516 * AUBO_BAD_STATE
00517 * -AUBO_INVL_ARGUMENT
00518 * -AUBO_BAD_STATE
00519 *
00520 * @throws arcs::common_interface::AuboException
00521 *
00522 * @par Python 函数原型
00523 * setDynamicModelSearch(self: pyaubo_sdk.ForceControl, arg0: List[float],
00524 * arg1: List[float]) -> int
00525 *
00526 * @par Lua 函数原型
00527 * setDynamicModelSearch(damp_scale: table, stiff_scale: table) -> nil
00528 *
00529 * @par Lua 示例
00530 * setDynamicModelSearch({0.5,0.5,0.5,0.5,0.5,0.5},{0.5,0.5,0.5,0.5,0.5,0.5})
00531 *
00532 * \endchinese
00533 * \english
00534 * Set force control dynamics model for hole searching scenario
00535 *
00536 * @param damp_scale: Parameter representing damping level, range [0, 1], default 0.5
00537 * @param stiff_scale: Parameter representing stiffness level, range [0, 1], default 0.5
00538 *
00539 * @return Return 0 if succeeded; return error code if failed
00540 * AUBO_BUSY
00541 * AUBO_BAD_STATE
00542 * -AUBO_INVL_ARGUMENT
00543 * -AUBO_BAD_STATE
00544 *
00545 * @throws arcs::common_interface::AuboException
00546 *
00547 * @par Python Function Prototype
00548 * setDynamicModelSearch(self: pyaubo_sdk.ForceControl, arg0: List[float],
00549 * arg1: List[float]) -> int
00550 *
00551 * @par Lua Function Prototype
00552 * setDynamicModelSearch(damp_scale: table, stiff_scale: table) -> nil
00553 *
00554 * @par Lua example
00555 * setDynamicModelSearch({0.5,0.5,0.5,0.5,0.5,0.5},{0.5,0.5,0.5,0.5,0.5,0.5})
00556 *
00557 * \endenglish
00558 */
00559 int setDynamicModelSearch(const std::vector<double> &damp_scale,
00560                          const std::vector<double> &stiff_scale);
00561
00562 /**
00563 * @ingroup ForceControl
00564 * \chinese

```

```

00565 * 设置力控插/拔孔场景下的动力学模型
00566 *
00567 * @param damp_scale: 表征阻尼水平的参数, 取值范围 [0, 1], 默认为 0.5
00568 * @param stiff_scale: 表征软硬程度的参数, 取值范围 [0, 1], 默认为 0.5
00569 *
00570 * @return 成功返回 0; 失败返回错误码
00571 * AUBO_BUSY
00572 * AUBO_BAD_STATE
00573 * -AUBO_INVL_ARGUMENT
00574 * -AUBO_BAD_STATE
00575 *
00576 * @throws arcs::common_interface::AuboException
00577 *
00578 * @par Python 函数原型
00579 * setDynamicModelInsert(self: pyaubo_sdk.ForceControl, arg0: List[float],
00580 * arg1: List[float]) -> int
00581 *
00582 * @par Lua 函数原型
00583 * setDynamicModelInsert(damp_scale: table, stiff_scale: table) -> nil
00584 *
00585 * @par Lua 示例
00586 * setDynamicModelInsert({0.5,0.5,0.5,0.5,0.5,0.5},{0.5,0.5,0.5,0.5,0.5,0.5})
00587 *
00588 * \endchinese
00589 * \english
00590 * Set force control dynamics model for insertion/extraction scenario
00591 *
00592 * @param damp_scale: Parameter representing damping level, range [0, 1], default 0.5
00593 * @param stiff_scale: Parameter representing stiffness level, range [0, 1], default 0.5
00594 *
00595 * @return Return 0 if succeeded; return error code if failed
00596 * AUBO_BUSY
00597 * AUBO_BAD_STATE
00598 * -AUBO_INVL_ARGUMENT
00599 * -AUBO_BAD_STATE
00600 *
00601 * @throws arcs::common_interface::AuboException
00602 *
00603 * @par Python Function Prototype
00604 * setDynamicModelInsert(self: pyaubo_sdk.ForceControl, arg0: List[float],
00605 * arg1: List[float]) -> int
00606 *
00607 * @par Lua Function Prototype
00608 * setDynamicModelInsert(damp_scale: table, stiff_scale: table) -> nil
00609 *
00610 * @par Lua example
00611 * setDynamicModelInsert({0.5,0.5,0.5,0.5,0.5,0.5},{0.5,0.5,0.5,0.5,0.5,0.5})
00612 *
00613 * \endenglish
00614 */
00615 int setDynamicModelInsert(const std::vector<double> &damp_scale,
00616                          const std::vector<double> &stiff_scale);
00617 /**
00618 * @ingroup ForceControl
00619 * \chinese
00620 * 设置力控接触场景下的动力学模型
00621 *
00622 * @param env_stiff: 表征环境刚度的参数, 取值范围 [0, 1], 默认为 0
00623 * @param damp_scale: 表征阻尼水平的参数, 取值范围 [0, 1], 默认为 0.5
00624 * @param stiff_scale: 表征软硬程度的参数, 取值范围 [0, 1], 默认为 0.5
00625 *
00626 * @return 成功返回 0; 失败返回错误码
00627 * AUBO_BUSY
00628 * AUBO_BAD_STATE
00629 * -AUBO_INVL_ARGUMENT
00630 * -AUBO_BAD_STATE
00631 *
00632 * @throws arcs::common_interface::AuboException
00633 *
00634 * @par Python 函数原型
00635 * setDynamicModelContact(self: pyaubo_sdk.ForceControl, arg0: List[float],
00636 * arg1: List[float], arg2: List[float]) -> int
00637 *
00638 * @par Lua 函数原型
00639 * setDynamicModelContact(env_stiff: table, damp_scale: table, stiff_scale:
00640 * table) -> nil
00641 *
00642 * @par Lua 示例
00643 * setDynamicModelContact({0,0,0,0,0,0},{0.5,0.5,0.5,0.5,0.5,0.5},{0.5,0.5,0.5,0.5,0.5,0.5})
00644 *
00645 * \endchinese
00646 * \english
00647 * Set force control dynamics model for contact scenario
00648 *
00649 * @param env_stiff: Parameter representing environment stiffness, range [0, 1], default 0
00650 * @param damp_scale: Parameter representing damping level, range [0, 1], default 0.5
00651 * @param stiff_scale: Parameter representing stiffness level, range [0, 1], default 0.5

```

```

00652      *
00653      * @return Return 0 if succeeded; return error code if failed
00654      * AUBO_BUSY
00655      * AUBO_BAD_STATE
00656      * -AUBO_INVL_ARGUMENT
00657      * -AUBO_BAD_STATE
00658      *
00659      * @throws arcs::common_interface::AuboException
00660      *
00661      * @par Python Function Prototype
00662      * setDynamicModelContact(self: pyaubo_sdk.ForceControl, arg0: List[float],
00663      * arg1: List[float], arg2: List[float]) -> int
00664      *
00665      * @par Lua Function Prototype
00666      * setDynamicModelContact(env_stiff: table, damp_scale: table, stiff_scale:
00667      * table) -> nil
00668      *
00669      * @par Lua example
00670      * setDynamicModelContact({0,0,0,0,0,0},{0.5,0.5,0.5,0.5,0.5,0.5},{0.5,0.5,0.5,0.5,0.5,0.5})
00671      *
00672      * \endenglish
00673      */
00674      int setDynamicModelContact(const std::vector<double> &env_stiff,
00675      const std::vector<double> &damp_scale,
00676      const std::vector<double> &stiff_scale);
00677
00678      /**
00679      * @ingroup ForceControl
00680      * \chinese
00681      * 设置力控动力学模型
00682      *
00683      * @param m 质量参数
00684      * @param d 阻尼参数
00685      * @param k 刚度参数
00686      *
00687      * 参数单位说明:
00688      * - 质量:
00689      *   - 笛卡尔空间: 单位为 kg, 长度为工作空间维度 (一般为 6), 顺序为 [x, y, z, Rx, Ry, Rz]
00690      *   - 关节空间: 单位为 kg·m², 长度为机器人自由度 (一般为 6), 顺序为 [J1, J2, J3, J4, J5, J6]
00691      * - 阻尼:
00692      *   - 笛卡尔空间: 单位为 N·s/m, 顺序为 [x, y, z, Rx, Ry, Rz]
00693      *   - 关节空间: 单位为 N·m·s/rad, 顺序为 [J1, J2, J3, J4, J5, J6]
00694      * - 刚度:
00695      *   - 笛卡尔空间: 单位为 N/m, 顺序为 [x, y, z, Rx, Ry, Rz]
00696      *   - 关节空间: 单位为 N·m/rad, 顺序为 [J1, J2, J3, J4, J5, J6]
00697      *
00698      * @return 成功返回 0; 失败返回错误码
00699      * AUBO_BUSY
00700      * AUBO_BAD_STATE
00701      * -AUBO_INVL_ARGUMENT
00702      * -AUBO_BAD_STATE
00703      *
00704      * @throws arcs::common_interface::AuboException
00705      *
00706      * @par Python 函数原型
00707      * setDynamicModel(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
00708      * List[float], arg2: List[float]) -> int
00709      *
00710      * @par Lua 函数原型
00711      * setDynamicModel(m: table, d: table, k: table) -> nil
00712      *
00713      * @par Lua 示例
00714      * setDynamicModel({20.0, 20.0, 20.0, 10.0, 10.0, 10.0},{2000, 0, 0, 0, 0, 0},{0, 0, 0, 0, 0, 0})
00715      *
00716      * \endchinese
00717      * \english
00718      * Set force control dynamics model
00719      *
00720      * @param m Mass parameters
00721      * @param d Damping parameters
00722      * @param k Stiffness parameters
00723      *
00724      * Parameter unit description:
00725      * - Mass:
00726      *   - Cartesian space: unit is kg, length is 6, order is [x, y, z, Rx, Ry, Rz]
00727      *   - Joint space: unit is kg·m², length is 6, order is [J1, J2, J3, J4, J5, J6]
00728      * - Damping:
00729      *   - Cartesian space: unit is N·s/m, order is [x, y, z, Rx, Ry, Rz]
00730      *   - Joint space: unit is N·m·s/rad, order is [J1, J2, J3, J4, J5, J6]
00731      * - Stiffness:
00732      *   - Cartesian space: unit is N/m, order is [x, y, z, Rx, Ry, Rz]
00733      *   - Joint space: unit is N·m/rad, order is [J1, J2, J3, J4, J5, J6]
00734      *
00735      * @return Return 0 if succeeded; return error code if failed
00736      * AUBO_BUSY
00737      * AUBO_BAD_STATE
00738      * -AUBO_INVL_ARGUMENT

```

```

00739     * -AUBO_BAD_STATE
00740     *
00741     * @throws arcs::common_interface::AuboException
00742     *
00743     * @par Python Function Prototype
00744     * setDynamicModel(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
00745     * List[float], arg2: List[float]) -> int
00746     *
00747     * @par Lua Function Prototype
00748     * setDynamicModel(m: table, d: table, k: table) -> nil
00749     *
00750     * @par Lua example
00751     * setDynamicModel({20.0, 20.0, 20.0, 10.0, 10.0, 10.0},{2000, 0, 0, 0, 0, 0},{0, 0, 0, 0, 0, 0})
00752     *
00753     * \endenglish
00754     */
00755 int setDynamicModel(const std::vector<double> &m,
00756                   const std::vector<double> &d,
00757                   const std::vector<double> &k);
00758
00759 /**
00760  * @ingroup ForceControl
00761  * \chinese
00762  * 设置力控阈值
00763  *
00764  * @param thresholds: 力控阈值
00765  *
00766  * @return 成功返回 0; 失败返回错误码
00767  * AUBO_BUSY
00768  * AUBO_BAD_STATE
00769  * -AUBO_INVL_ARGUMENT
00770  * -AUBO_BAD_STATE
00771  *
00772  * @throws arcs::common_interface::AuboException
00773  *
00774  * @par Python 函数原型
00775  * fcSetSensorThresholds(self: pyaubo_sdk.ForceControl, arg0:
00776  * List[float]) -> int
00777  *
00778  * @par Lua 函数原型
00779  * fcSetSensorThresholds(thresholds: table) -> nil
00780  *
00781  * @par Lua 示例
00782  * fcSetSensorThresholds({1.0,1.0,1.0,0.5,0.5,0.5})
00783  *
00784  * \endch
00785  * \endchinese
00786  * \english
00787  * Set force control thresholds
00788  *
00789  * @param thresholds: Force control thresholds
00790  *
00791  * @return Return 0 if succeeded; return error code if failed
00792  * AUBO_BUSY
00793  * AUBO_BAD_STATE
00794  * -AUBO_INVL_ARGUMENT
00795  * -AUBO_BAD_STATE
00796  *
00797  * @throws arcs::common_interface::AuboException
00798  *
00799  * @par Python Function Prototype
00800  * fcSetSensorThresholds(self: pyaubo_sdk.ForceControl, arg0:
00801  * List[float]) -> int
00802  *
00803  * @par Lua Function Prototype
00804  * fcSetSensorThresholds(thresholds: table) -> nil
00805  *
00806  * @par Lua example
00807  * fcSetSensorThresholds({1.0,1.0,1.0,0.5,0.5,0.5})
00808  *
00809  * \endenglish
00810  */
00811 int fcSetSensorThresholds(const std::vector<double> &thresholds);
00812
00813 /**
00814  * @ingroup ForceControl
00815  * \chinese
00816  * 设置力控最大受力限制
00817  *
00818  * @param limits: 力限制
00819  *
00820  * @return 成功返回 0; 失败返回错误码
00821  * AUBO_BUSY
00822  * AUBO_BAD_STATE
00823  * -AUBO_INVL_ARGUMENT
00824  * -AUBO_BAD_STATE
00825  *

```

```

00826     * @throws arcs::common_interface::AuboException
00827     *
00828     * @par Python 函数原型
00829     * fcSetSensorLimits(self: pyaubo_sdk.ForceControl, arg0:
00830     * List[float]) -> int
00831     *
00832     * @par Lua 函数原型
00833     * fcSetSensorLimits(limits: table) -> nil
00834     *
00835     * @par Lua 示例
00836     * fcSetSensorLimits({200.0,200.0,200.0,50.0,50.0,50.0})
00837     *
00838     * \endchinese
00839     * \english
00840     * Set force control maximum force limits
00841     *
00842     * @param limits: Force limits
00843     *
00844     * @return Return 0 if succeeded; return error code if failed
00845     * AUBO_BUSY
00846     * AUBO_BAD_STATE
00847     * -AUBO_INVL_ARGUMENT
00848     * -AUBO_BAD_STATE
00849     *
00850     * @throws arcs::common_interface::AuboException
00851     *
00852     * @par Python Function Prototype
00853     * fcSetSensorLimits(self: pyaubo_sdk.ForceControl, arg0:
00854     * List[float]) -> int
00855     *
00856     * @par Lua Function Prototype
00857     * fcSetSensorLimits(limits: table) -> nil
00858     *
00859     * @par Lua example
00860     * fcSetSensorLimits({200.0,200.0,200.0,50.0,50.0,50.0})
00861     *
00862     * \endenglish
00863     */
00864 int fcSetSensorLimits(const std::vector<double> &limits);
00865
00866 /**
00867  * @ingroup ForceControl
00868  * \chinese
00869  * 获取力控阈值
00870  *
00871  * @return 力控最小力阈值
00872  *
00873  * @throws arcs::common_interface::AuboException
00874  *
00875  * @par Python 函数原型
00876  * getFcSensorThresholds(self: pyaubo_sdk.ForceControl) -> list
00877  *
00878  * @par Lua 函数原型
00879  * getFcSensorThresholds() -> table
00880  *
00881  * @par Lua 函数示例
00882  * Fc_Thresholds = getFcSensorThresholds()
00883  *
00884  * \endchinese
00885  * \english
00886  * Get force control thresholds
00887  *
00888  * @return Minimum force control thresholds
00889  *
00890  * @throws arcs::common_interface::AuboException
00891  *
00892  * @par Python Function Prototype
00893  * getFcSensorThresholds(self: pyaubo_sdk.ForceControl) -> list
00894  *
00895  * @par Lua Function Prototype
00896  * getFcSensorThresholds() -> table
00897  *
00898  * @par Lua example
00899  * Fc_Thresholds = getFcSensorThresholds()
00900  *
00901  * \endenglish
00902  */
00903 std::vector<double> getFcSensorThresholds();
00904
00905 /**
00906  * @ingroup ForceControl
00907  * \chinese
00908  * 获取最大力限制
00909  *
00910  * @return 力控最大力限制
00911  *
00912  * @throws arcs::common_interface::AuboException

```

```

00913     *
00914     * @par Python 函数原型
00915     * getFcSensorLimits(self: pyaubo_sdk.ForceControl) -> list
00916     *
00917     * @par Lua 函数原型
00918     * getFcSensorLimits() -> table
00919     *
00920     * @par Lua 示例
00921     * Fc_Limits = getFcSensorLimits()
00922     *
00923     * \endchinese
00924     * \english
00925     * Get maximum force limits
00926     *
00927     * @return Force control maximum force limits
00928     *
00929     * @throws arcs::common_interface::AuboException
00930     *
00931     * @par Python Function Prototype
00932     * getFcSensorLimits(self: pyaubo_sdk.ForceControl) -> list
00933     *
00934     * @par Lua Function Prototype
00935     * getFcSensorLimits() -> table
00936     *
00937     * @par Lua example
00938     * Fc_Limits = getFcSensorLimits()
00939     *
00940     * \endenglish
00941     */
00942 std::vector<double> getFcSensorLimits();
00943
00944 /**
00945  * @ingroup ForceControl
00946  * \chinese
00947  * 获取力控动力学模型
00948  *
00949  * @return 力控动力学模型
00950  *
00951  * @throws arcs::common_interface::AuboException
00952  *
00953  * @par Python 函数原型
00954  * getDynamicModel(self: pyaubo_sdk.ForceControl) -> Tuple[List[float],
00955  * List[float], List[float]]
00956  *
00957  * @par Lua 函数原型
00958  * getDynamicModel() -> table
00959  *
00960  * @par Lua 示例
00961  * Fc_Model = getDynamicModel()
00962  *
00963  * @par JSON-RPC 请求示例
00964  * {"jsonrpc": "2.0", "method": "rob1.ForceControl.getDynamicModel", "params": [], "id": 1}
00965  *
00966  * @par JSON-RPC 响应示例
00967  * {"id": 1, "jsonrpc": "2.0", "result": [[], [20.0, 20.0, 20.0, 5.0, 5.0, 5.0], []]}
00968  * \endchinese
00969  * \english
00970  * Get force control dynamics model
00971  *
00972  * @return Force control dynamics model
00973  *
00974  * @throws arcs::common_interface::AuboException
00975  *
00976  * @par Python Function Prototype
00977  * getDynamicModel(self: pyaubo_sdk.ForceControl) -> Tuple[List[float],
00978  * List[float], List[float]]
00979  *
00980  * @par Lua Function Prototype
00981  * getDynamicModel() -> table
00982  *
00983  * @par Lua example
00984  * Fc_Model = getDynamicModel()
00985  *
00986  * @par JSON-RPC Request example
00987  * {"jsonrpc": "2.0", "method": "rob1.ForceControl.getDynamicModel", "params": [], "id": 1}
00988  *
00989  * @par JSON-RPC Response example
00990  * {"id": 1, "jsonrpc": "2.0", "result": [[], [20.0, 20.0, 20.0, 5.0, 5.0, 5.0], []]}
00991  * \endenglish
00992  */
00993 DynamicsModel getDynamicModel();
00994
00995 /**
00996  * @ingroup ForceControl
00997  * \chinese
00998  * 设置力控终止条件: 力, 当测量的力在设置的范围之内, 力控算法将保持运行, 直到设置的条件不满足, 力控将退出
00999  *

```

```

01000 * The condition is lateractivated by calling the instruction
01001 * FCCondWaitWhile, which will wait and hold the program execution while the
01002 * specified condition is true. This allows the reference force, torque and
01003 * movement to continue until the force is outside the specified limits.
01004 *
01005 * A force condition is set up by defining minimum and maximum limits for
01006 * the force in the directions of the force control coordinate system. Once
01007 * activated with FCCondWaitWhile, the program execution will continue to
01008 * wait while the measured force is within its specified limits.
01009 *
01010 * It is possible to specify that the condition is fulfilled when the force
01011 * is outside the specified limits instead. This is done by using the switch
01012 * argument Outside. The condition on force is specified in the force
01013 * control coordinate system. This coordinate system is setup by the user in
01014 * the instruction FCAct.
01015 *
01016 * @param min 各方向最小的力/力矩
01017 * @param max 各方向最大的力/力矩
01018 * @param outside false 在设置条件的范围之内有效
01019 *                true 在设置条件的范围之外有效
01020 * @param timeout
01021 * 时间限制, 单位 s(秒), 从开始力控到达该时间时, 不管是否满足力控终止条件, 都会终止力控
01022 *
01023 * @return 成功返回 0; 失败返回错误码
01024 * AUBO_BUSY
01025 * AUBO_BAD_STATE
01026 * -AUBO_INVL_ARGUMENT
01027 * -AUBO_BAD_STATE
01028 *
01029 * @throws arcs::common_interface::AuboException
01030 *
01031 * @par Python 函数原型
01032 * setCondForce(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
01033 * List[float], arg2: bool, arg3: float) -> int
01034 *
01035 * @par Lua 函数原型
01036 * setCondForce(min: table, max: table, outside: boolean, timeout: number)
01037 * -> nil
01038 * @par Lua 示例
01039 * setCondForce({0.1,0.1,0.1,0.1,0.1,0.1},{50,50,50,5,5,5},{true,true,true,true,true,true},10)
01040 *
01041 * \endchinese
01042 * \english
01043 * Set force control termination condition: Force. When the measured force is within
01044 * the specified range, the force control algorithm will continue to run until the set
01045 * condition is not met, at which point force control will exit.
01046 *
01047 * The condition is lateractivated by calling the instruction
01048 * FCCondWaitWhile, which will wait and hold the program execution while the
01049 * specified condition is true. This allows the reference force, torque and
01050 * movement to continue until the force is outside the specified limits.
01051 *
01052 * A force condition is set up by defining minimum and maximum limits for
01053 * the force in the directions of the force control coordinate system. Once
01054 * activated with FCCondWaitWhile, the program execution will continue to
01055 * wait while the measured force is within its specified limits.
01056 *
01057 * It is possible to specify that the condition is fulfilled when the force
01058 * is outside the specified limits instead. This is done by using the switch
01059 * argument Outside. The condition on force is specified in the force
01060 * control coordinate system. This coordinate system is setup by the user in
01061 * the instruction FCAct.
01062 *
01063 * @param min Minimum force/torque in each direction
01064 * @param max Maximum force/torque in each direction
01065 * @param outside false: valid within the specified range
01066 *                true:  valid outside the specified range
01067 * @param timeout
01068 * Time limit in seconds; when this time is reached from the start of force control,
01069 * force control will terminate regardless of whether the end condition is met
01070 *
01071 * @return Return 0 if succeeded; return error code if failed
01072 * AUBO_BUSY
01073 * AUBO_BAD_STATE
01074 * -AUBO_INVL_ARGUMENT
01075 * -AUBO_BAD_STATE
01076 *
01077 * @throws arcs::common_interface::AuboException
01078 *
01079 * @par Python Function Prototype
01080 * setCondForce(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
01081 * List[float], arg2: bool, arg3: float) -> int
01082 *
01083 * @par Lua Function Prototype
01084 * setCondForce(min: table, max: table, outside: boolean, timeout: number)
01085 * -> nil
01086 * @par Lua example

```



```

01087     * setCondForce({0.1,0.1,0.1,0.1,0.1,0.1},{50,50,50,5,5,5},{true,true,true,true,true,true},10)
01088     *
01089     * \endenglish
01090     */
01091 int setCondForce(const std::vector<double> &min,
01092                 const std::vector<double> &max, bool outside,
01093                 double timeout);
01094
01095 /**
01096  * @ingroup ForceControl
01097  * \chinese
01098  * 设置力控终止条件: 姿态, 当测量的姿态在设置的范围之内, 力控算法将保持运行, 直到设置的条件不满足, 力控将退出.
01099  *
01100  * setCondOrient is used to set up an end condition for the tool orientation.
01101  * The condition is later activated by calling the instruction
01102  * FCondWaitWhile, which will wait and hold the program execution while the
01103  * specified condition is true. This allows the reference force, torque and
01104  * movement to continue until the orientation is outside the specified
01105  * limits.
01106  *
01107  * An orientation condition is set up by defining a maximum angle and a
01108  * maximum rotation from a reference orientation. The reference orientation
01109  * is either defined by the current z direction of the tool, or by
01110  * specifying an orientation in relation to the z direction of the work
01111  * object.
01112  *
01113  * Once activated, the tool orientation must be within the limits (or
01114  * outside, if the argument Outside is used).
01115  *
01116  * @param frame
01117  * @param max_angle
01118  * @param max_rot
01119  * @param outside
01120  * @param timeout
01121  *
01122  * @return 成功返回 0; 失败返回错误码
01123  * AUBO_BUSY
01124  * AUBO_BAD_STATE
01125  * -AUBO_INVL_ARGUMENT
01126  * -AUBO_BAD_STATE
01127  *
01128  * @throws arcs::common_interface::AuboException
01129  *
01130  * @par Python 函数原型
01131  * setCondOrient(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
01132  * float, arg2: float, arg3: bool, arg4: float) -> int
01133  *
01134  * @par Lua 函数原型
01135  * setCondOrient(frame: table, max_angle: number, max_rot: number, outside:
01136  * boolean, timeout: number) -> nil
01137  *
01138  * @par Lua 示例
01139  * setCondOrient({0.1, 0.3, 0.1, 0.3142, 0.0, 1.571},30.0,20.0,true,10.0)
01140  *
01141  * \endchinese
01142  *
01143  * \english
01144  * setCondOrient is used to set up an end condition for the tool orientation.
01145  * The condition will wait and hold the program execution while the
01146  * specified condition is true. This allows the reference force, torque and
01147  * movement to continue until the orientation is outside the specified
01148  * limits.
01149  *
01150  * An orientation condition is set up by defining a maximum angle and a
01151  * maximum rotation from a reference orientation. The reference orientation
01152  * is either defined by the current z direction of the tool, or by
01153  * specifying an orientation in relation to the z direction of the work
01154  * object.
01155  *
01156  * Once activated, the tool orientation must be within the limits (or
01157  * outside, if the argument Outside is used).
01158  *
01159  * @param frame
01160  * @param max_angle
01161  * @param max_rot
01162  * @param outside
01163  * @param timeout
01164  *
01165  * @return Return 0 if succeeded; return error code if failed
01166  * AUBO_BUSY
01167  * AUBO_BAD_STATE
01168  * -AUBO_INVL_ARGUMENT
01169  * -AUBO_BAD_STATE
01170  *
01171  * @throws arcs::common_interface::AuboException
01172  *
01173  * @par Python Function Prototype

```



```

01174     * setCondOrient(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
01175     * float, arg2: float, arg3: bool, arg4: float) -> int
01176     *
01177     * @par Lua Function Prototype
01178     * setCondOrient(frame: table, max_angle: number, max_rot: number, outside:
01179     * boolean, timeout: number) -> nil
01180     *
01181     * @par Lua example
01182     * setCondOrient({0.1, 0.3, 0.1, 0.3142, 0.0, 1.571},30.0,20.0,true,10.0)
01183     *
01184     * \endenglish
01185     */
01186 int setCondOrient(const std::vector<double> &frame, double max_angle,
01187                  double max_rot, bool outside, double timeout);
01188
01189 /**
01190  * @ingroup ForceControl
01191  * \chinese
01192  * 指定力控有效平面, x-y 平面, z 方向有效
01193  *
01194  * @param plane={A,B,C,D}
01195  *     平面表示方法  $Ax + By + Cz + D = 0$ 
01196  *     其中,  $n = (A, B, C)$  是平面的法向量,
01197  *      $D$  是将平面平移到坐标原点所需距离 (所以  $D=0$  时, 平面过原点)
01198  * @param timeout
01199  *
01200  * @return 成功返回 0; 失败返回错误码
01201  * AUBO_BUSY
01202  * AUBO_BAD_STATE
01203  * -AUBO_INVL_ARGUMENT
01204  * -AUBO_BAD_STATE
01205  *
01206  * @throws arcs::common_interface::AuboException
01207  *
01208  * @par Python 函数原型
01209  * setCondPlane(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
01210  * float) -> int
01211  *
01212  * @par Lua 函数原型
01213  * setCondPlane(plane: table, timeout: number) -> nil
01214  *
01215  * @par Lua 示例
01216  * setCondPlane({0.1, 0.3, 0.1, 0.3},10.0)
01217  *
01218  * \endchinese
01219  * \english
01220  * Specify a valid force control plane, x-y plane, z direction is valid
01221  *
01222  * @param plane={A,B,C,D}
01223  *     Plane equation:  $Ax + By + Cz + D = 0$ 
01224  *     where  $n = (A, B, C)$  is the normal vector of the plane,
01225  *      $D$  is the distance required to move the plane to the origin (so  $D=0$  means the plane passes through the
01226  * origin)
01227  * @param timeout
01228  *
01229  * @return Return 0 if succeeded; return error code if failed
01230  * AUBO_BUSY
01231  * AUBO_BAD_STATE
01232  * -AUBO_INVL_ARGUMENT
01233  * -AUBO_BAD_STATE
01234  *
01235  * @throws arcs::common_interface::AuboException
01236  *
01237  * @par Python Function Prototype
01238  * setCondPlane(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
01239  * float) -> int
01240  *
01241  * @par Lua Function Prototype
01242  * setCondPlane(plane: table, timeout: number) -> nil
01243  *
01244  * @par Lua example
01245  * setCondPlane({0.1, 0.3, 0.1, 0.3},10.0)
01246  *
01247  * \endenglish
01248  */
01249 int setCondPlane(const std::vector<double> &plane, double timeout);
01250
01251 /**
01252  * @ingroup ForceControl
01253  * \chinese
01254  * 指定力控有效圆柱体, 提供中心轴和圆柱半径, 可以指定圆柱内部还是外部
01255  *
01256  * @param axis
01257  * @param radius
01258  * @param outside
01259  * @param timeout
01260  *

```

```

01260      * @return 成功返回 0; 失败返回错误码
01261      * AUBO_BUSY
01262      * AUBO_BAD_STATE
01263      * -AUBO_INVL_ARGUMENT
01264      * -AUBO_BAD_STATE
01265      *
01266      * @throws arcs::common_interface::AuboException
01267      *
01268      * @par Python 函数原型
01269      * setCondCylinder(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
01270      * float, arg2: bool, arg3: float) -> int
01271      *
01272      * @par Lua 函数原型
01273      * setCondCylinder(axis: table, radius: number, outside: boolean, timeout:
01274      * number) -> nil
01275      *
01276      * @par Lua 示例
01277      * setCondCylinder({0,1,0},10.0,true,5,0)
01278      *
01279      * \endchinese
01280      * \english
01281      * Specify a valid force control cylinder by providing the central axis and cylinder radius, and specify whether
the inside or outside of the cylinder is valid.
01282      *
01283      * @param axis
01284      * @param radius
01285      * @param outside
01286      * @param timeout
01287      *
01288      * @return Return 0 if succeeded; return error code if failed
01289      * AUBO_BUSY
01290      * AUBO_BAD_STATE
01291      * -AUBO_INVL_ARGUMENT
01292      * -AUBO_BAD_STATE
01293      *
01294      * @throws arcs::common_interface::AuboException
01295      *
01296      * @par Python Function Prototype
01297      * setCondCylinder(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
01298      * float, arg2: bool, arg3: float) -> int
01299      *
01300      * @par Lua Function Prototype
01301      * setCondCylinder(axis: table, radius: number, outside: boolean, timeout:
01302      * number) -> nil
01303      *
01304      * @par Lua example
01305      * setCondCylinder({0,1,0},10.0,true,5,0)
01306      *
01307      * \endenglish
01308      */
01309 int setCondCylinder(const std::vector<double> &axis, double radius,
01310                    bool outside, double timeout);
01311
01312 /**
01313  * @ingroup ForceControl
01314  * \chinese
01315  * 指定力控有效球体, 提供球心和半径, 可以指定球体内部还是外部
01316  *
01317  * @param center
01318  * @param radius
01319  * @param outside
01320  * @param timeout
01321  *
01322  * @return 成功返回 0; 失败返回错误码
01323  * AUBO_BUSY
01324  * AUBO_BAD_STATE
01325  * -AUBO_INVL_ARGUMENT
01326  * -AUBO_BAD_STATE
01327  *
01328  * @throws arcs::common_interface::AuboException
01329  *
01330  * @par Python 函数原型
01331  * setCondSphere(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
01332  * float, arg2: bool, arg3: float) -> int
01333  *
01334  * @par Lua 函数原型
01335  * setCondSphere(center: table, radius: number, outside: boolean, timeout:
01336  * number) -> nil
01337  *
01338  * @par Lua 示例
01339  * setCondSphere({0.2, 0.5, 0.1, 1.57, 0, 0},10.0,true,5,0)
01340  *
01341  * \endchinese
01342  * \english
01343  * Specify a valid force control sphere by providing the center and radius, and specify whether the inside or
outside of the sphere is valid.
01344  *

```

```

01345     * @param center
01346     * @param radius
01347     * @param outside
01348     * @param timeout
01349     *
01350     * @return Return 0 if succeeded; return error code if failed
01351     * AUBO_BUSY
01352     * AUBO_BAD_STATE
01353     * -AUBO_INVL_ARGUMENT
01354     * -AUBO_BAD_STATE
01355     *
01356     * @throws arcs::common_interface::AuboException
01357     *
01358     * @par Python Function Prototype
01359     * setCondSphere(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
01360     * float, arg2: bool, arg3: float) -> int
01361     *
01362     * @par Lua Function Prototype
01363     * setCondSphere(center: table, radius: number, outside: boolean, timeout:
01364     * number) -> nil
01365     *
01366     * @par Lua example
01367     * setCondSphere({0.2, 0.5, 0.1, 1.57, 0, 0},10.0,true,5,0)
01368     *
01369     * \endenglish
01370     */
01371 int setCondSphere(const std::vector<double> &center, double radius,
01372                  bool outside, double timeout);
01373
01374 /**
01375  * @ingroup ForceControl
01376  * \chinese
01377  * 设置 TCP 速度的终止条件。该条件可通过调用 FCCondWaitWhile 指令激活，
01378  * 在指定条件为真时，程序将等待并保持执行。这样可以使参考力、力矩和运动继续，
01379  * 直到速度超出指定范围。
01380  *
01381  * 通过定义 TCP 在工作对象所有方向上的最小和最大速度限制来设置 TCP 速度条件。
01382  * 一旦通过 FCCondWaitWhile 激活，程序将在测量速度处于指定范围内时继续等待。
01383  *
01384  * 也可以指定当速度超出指定范围时条件成立，通过使用 outside 参数实现。
01385  * TCP 速度条件在工作对象坐标系中指定。
01386  *
01387  * @param min 最小速度
01388  * @param max 最大速度
01389  * @param outside 是否在范围外有效
01390  * @param timeout 超时时间
01391  *
01392  * @return 成功返回 0；失败返回错误码
01393  * AUBO_BUSY
01394  * AUBO_BAD_STATE
01395  * -AUBO_INVL_ARGUMENT
01396  * -AUBO_BAD_STATE
01397  *
01398  * @throws arcs::common_interface::AuboException
01399  *
01400  * @par Python 函数原型
01401  * setCondTcpSpeed(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
01402  * List[float], arg2: bool, arg3: float) -> int
01403  *
01404  * @par Lua 函数原型
01405  * setCondTcpSpeed(min: table, max: table, outside: boolean, timeout:
01406  * number) -> nil
01407  *
01408  * @par Lua 示例
01409  * setCondTcpSpeed({0.2, 0.2, 0.2, 0.2, 0.2, 0.2},{1.2, 1.2, 1.2, 1.2, 1.2, 1.2},true,5,0)
01410  *
01411  * \endchinese
01412  * \english
01413  * setCondTcpSpeed is used to setup an end condition for the TCP speed. The
01414  * condition is later activated by calling the instruction FCCondWaitWhile,
01415  * which will wait and hold the program execution while the specified
01416  * condition is true. This allows the reference force, torque and movement
01417  * to continue until the speed is outside the specified limits.
01418  *
01419  * A TCP speed condition is set up by defining minimum and maximum limits
01420  * for the TCP speed in all directions of the work object. Once activated
01421  * with FCCondWaitWhile, the program execution will continue to wait while
01422  * the measured speed is within its specified limits.
01423  *
01424  * It is possible to specify that the condition is fulfilled when the speed
01425  * is outside the specified limits instead. This is done by using the
01426  * switch argument Outside. The condition on TCP speed is specified in the
01427  * work object coordinate system.
01428  *
01429  * @param min
01430  * @param max
01431  * @param outside

```

```

01432     * @param timeout
01433     *
01434     * @return Return 0 if succeeded; return error code if failed
01435     * AUBO_BUSY
01436     * AUBO_BAD_STATE
01437     * -AUBO_INVL_ARGUMENT
01438     * -AUBO_BAD_STATE
01439     *
01440     * @throws arcs::common_interface::AuboException
01441     *
01442     * @par Python Function Prototype
01443     * setCondTcpSpeed(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
01444     * List[float], arg2: bool, arg3: float) -> int
01445     *
01446     * @par Lua Function Prototype
01447     * setCondTcpSpeed(min: table, max: table, outside: boolean, timeout:
01448     * number) -> nil
01449     *
01450     * @par Lua example
01451     * setCondTcpSpeed({0.2, 0.2, 0.2, 0.2, 0.2, 0.2},{1.2, 1.2, 1.2, 1.2, 1.2, 1.2},true,5,0)
01452     *
01453     * \endenglish
01454     */
01455 int setCondTcpSpeed(const std::vector<double> &min,
01456                   const std::vector<double> &max, bool outside,
01457                   double timeout);
01458
01459 /**
01460  * @ingroup ForceControl
01461  * \chinese
01462  * 力控终止条件-距离
01463  *
01464  * @param distance 距离
01465  * @param timeout 超时时间
01466  *
01467  * @return 成功返回 0; 失败返回错误码
01468  * AUBO_BUSY
01469  * AUBO_BAD_STATE
01470  * -AUBO_INVL_ARGUMENT
01471  * -AUBO_BAD_STATE
01472  *
01473  * @throws arcs::common_interface::AuboException
01474  *
01475  * @par Lua 函数原型
01476  * setCondDistance(distance: number, timeout: number)
01477  *
01478  * @par Lua 示例
01479  * setCondDistance(0.2, 10.0)
01480  *
01481  * \endchinese
01482  * \english
01483  * Force control termination condition - distance
01484  *
01485  * @param distance Distance
01486  * @param timeout Timeout
01487  *
01488  * @return Return 0 if succeeded; return error code if failed
01489  * AUBO_BUSY
01490  * AUBO_BAD_STATE
01491  * -AUBO_INVL_ARGUMENT
01492  * -AUBO_BAD_STATE
01493  *
01494  * @throws arcs::common_interface::AuboException
01495  *
01496  * @par Lua Function Prototype
01497  * setCondDistance(distance: number, timeout: number)
01498  *
01499  * @par Lua example
01500  * setCondDistance(0.2, 10.0)
01501  *
01502  * \endenglish
01503  */
01504 int setCondDistance(double distance, double timeout);
01505
01506 /**
01507  * @ingroup ForceControl
01508  * \chinese
01509  * 高级力控终止条件
01510  *
01511  * @param type 类型
01512  * @param args 参数
01513  * @param timeout 超时时间
01514  *
01515  * @return 成功返回 0; 失败返回错误码
01516  * AUBO_BUSY
01517  * AUBO_BAD_STATE
01518  * -AUBO_INVL_ARGUMENT

```

```

01519     * -AUBO_BAD_STATE
01520     *
01521     * @throws arcs::common_interface::AuboException
01522     *
01523     * @par Lua 函数原型
01524     * setCondAdvanced(type: number, args: table, timeout: number)
01525     *
01526     * @par Lua 示例
01527     * setCondAdvanced(1, {0,1,0,0,0,0},10)
01528     *
01529     * \endchinese
01530     * \english
01531     * Advanced force control termination condition
01532     *
01533     * @param type Type
01534     * @param args Arguments
01535     * @param timeout Timeout
01536     *
01537     * @return Return 0 if succeeded; return error code if failed
01538     * AUBO_BUSY
01539     * AUBO_BAD_STATE
01540     * -AUBO_INVL_ARGUMENT
01541     * -AUBO_BAD_STATE
01542     *
01543     * @throws arcs::common_interface::AuboException
01544     *
01545     * @par Lua Function Prototype
01546     * setCondAdvanced(type: number, args: table, timeout: number)
01547     *
01548     * @par Lua example
01549     * setCondAdvanced(1, {0,1,0,0,0,0},10)
01550     *
01551     * \endenglish
01552     */
01553 int setCondAdvanced(const std::string &type,
01554                   const std::vector<double> &args, double timeout);
01555
01556 /**
01557  * @ingroup ForceControl
01558  * \chinese
01559  * 激活力控终止条件
01560  *
01561  * @return 成功返回 0; 失败返回错误码
01562  * AUBO_BUSY
01563  * AUBO_BAD_STATE
01564  * -AUBO_BAD_STATE
01565  *
01566  * @throws arcs::common_interface::AuboException
01567  *
01568  * @par Python 函数原型
01569  * setCondActive(self: pyaubo_sdk.ForceControl) -> int
01570  *
01571  * @par Lua 函数原型
01572  * setCondActive() -> nil
01573  *
01574  * @par Lua 示例
01575  * setCondActive()
01576  *
01577  * @par JSON-RPC 请求示例
01578  * {"jsonrpc": "2.0", "method": "rob1.ForceControl.setCondActive", "params": [], "id": 1}
01579  *
01580  * @par JSON-RPC 响应示例
01581  * {"id": 1, "jsonrpc": "2.0", "result": 0}
01582  * \endchinese
01583  * \english
01584  * Activate force control termination condition
01585  *
01586  * @return Return 0 if succeeded; return error code if failed
01587  * AUBO_BUSY
01588  * AUBO_BAD_STATE
01589  * -AUBO_BAD_STATE
01590  *
01591  * @throws arcs::common_interface::AuboException
01592  *
01593  * @par Python Function Prototype
01594  * setCondActive(self: pyaubo_sdk.ForceControl) -> int
01595  *
01596  * @par Lua Function Prototype
01597  * setCondActive() -> nil
01598  *
01599  * @par Lua example
01600  * setCondActive()
01601  *
01602  * @par JSON-RPC Request example
01603  * {"jsonrpc": "2.0", "method": "rob1.ForceControl.setCondActive", "params": [], "id": 1}
01604  *
01605  * @par JSON-RPC Response example

```

```

01606     * {"id":1,"jsonrpc":"2.0","result":0}
01607     * \endenglish
01608     */
01609     int setCondActive();
01610
01611     /**
01612     * @ingroup ForceControl
01613     * \chinese
01614     * 力控终止条件是否已经满足
01615     *
01616     * @return
01617     *
01618     * @throws arcs::common_interface::AuboException
01619     *
01620     * @par Python 函数原型
01621     * isCondFullfiled(self: pyaubo_sdk.ForceControl) -> bool
01622     *
01623     * @par Lua 函数原型
01624     * isCondFullfiled() -> boolean
01625     *
01626     * @par Lua 示例
01627     * status = isCondFullfiled()
01628     *
01629     * @par JSON-RPC 请求示例
01630     * {"jsonrpc":"2.0","method":"rob1.ForceControl.isCondFullfiled","params":[],"id":1}
01631     *
01632     * @par JSON-RPC 响应示例
01633     * {"id":1,"jsonrpc":"2.0","result":false}
01634     * \endchinese
01635     * \english
01636     * Check if the force control termination condition has been fulfilled
01637     *
01638     * @return
01639     *
01640     * @throws arcs::common_interface::AuboException
01641     *
01642     * @par Python Function Prototype
01643     * isCondFullfiled(self: pyaubo_sdk.ForceControl) -> bool
01644     *
01645     * @par Lua Function Prototype
01646     * isCondFullfiled() -> boolean
01647     *
01648     * @par Lua example
01649     * status = isCondFullfiled()
01650     *
01651     * @par JSON-RPC Request example
01652     * {"jsonrpc":"2.0","method":"rob1.ForceControl.isCondFullfiled","params":[],"id":1}
01653     *
01654     * @par JSON-RPC Response example
01655     * {"id":1,"jsonrpc":"2.0","result":false}
01656     * \endenglish
01657     */
01658     bool isCondFullfiled();
01659
01660     /**
01661     * @ingroup ForceControl
01662     * \chinese
01663     * setSupvForce 用于在力控中设置力监督。监督在通过 FCAct 指令激活力控时被激活。
01664     *
01665     * 力监督通过在力控坐标系的各个方向上定义最小和最大力限制来设置。
01666     * 一旦激活，如果力超出允许的范围，监督将停止执行。力监督在力控坐标系中指定。
01667     * 该坐标系由用户通过 FCAct 指令设置。
01668     *
01669     * @param min 最小力限制
01670     * @param max 最大力限制
01671     *
01672     * @return 成功返回 0；失败返回错误码
01673     * AUBO_BUSY
01674     * AUBO_BAD_STATE
01675     * -AUBO_INVL_ARGUMENT
01676     * -AUBO_BAD_STATE
01677     *
01678     * @throws arcs::common_interface::AuboException
01679     *
01680     * @par Python 函数原型
01681     * setSupvForce(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
01682     * List[float]) -> int
01683     *
01684     * @par Lua 函数原型
01685     * setSupvForce(min: table, max: table) -> nil
01686     *
01687     * @par Lua 示例
01688     * setSupvForce({0.1 ,0.1 ,0.1 ,0.1 ,0.1 ,0.1}, {10.0 ,10.0 ,10.0 ,10.0 ,10.0 ,10.0})
01689     *
01690     * \endchinese
01691     * \english
01692     * setSupvForce is used to set up force supervision in Force Control. The

```

```

01693     * supervision is activated when Force Control is activated with the
01694     * instruction FCAct.
01695     *
01696     * The force supervision is set up by defining minimum and maximum limits
01697     * for the force in the directions of the force control coordinate system.
01698     * Once activated, the supervision will stop the execution if the force is
01699     * outside the allowed values. The force supervision is specified in the
01700     * force control coordinate system. This coordinate system is setup by the
01701     * user with the instruction FCAct.
01702     *
01703     * @param min
01704     * @param max
01705     *
01706     * @return Return 0 if succeeded; return error code if failed
01707     * AUBO_BUSY
01708     * AUBO_BAD_STATE
01709     * -AUBO_INVL_ARGUMENT
01710     * -AUBO_BAD_STATE
01711     *
01712     * @throws arcs::common_interface::AuboException
01713     *
01714     * @par Python Function Prototype
01715     * setSupvForce(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
01716     * List[float]) -> int
01717     *
01718     * @par Lua Function Prototype
01719     * setSupvForce(min: table, max: table) -> nil
01720     *
01721     * @par Lua example
01722     * setSupvForce({0.1 ,0.1 ,0.1 ,0.1 ,0.1 ,0.1}, {10.0 ,10.0 ,10.0 ,10.0 ,10.0 ,10.0})
01723     *
01724     * \endenglish
01725     */
01726 int setSupvForce(const std::vector<double> &min,
01727                 const std::vector<double> &max);
01728
01729 /**
01730  * @ingroup ForceControl
01731  * \chinese
01732  * setSupvOrient 用于设置工具姿态的监督条件。
01733  * 当通过 FCAct 指令激活力控时，监督条件被激活。
01734  *
01735  * 姿态监督通过定义相对于参考姿态的最大角度和最大旋转来设置。
01736  * 参考姿态可以由工具当前的 z 方向定义，也可以通过指定相对于工作对象 z 方向的姿态来定义。
01737  *
01738  * 一旦激活，工具姿态必须在限制范围内，否则监督将停止执行。
01739  *
01740  * @param frame
01741  * @param max_angle
01742  * @param max_rot
01743  * @param outside
01744  *
01745  * @return 成功返回 0；失败返回错误码
01746  * AUBO_BUSY
01747  * AUBO_BAD_STATE
01748  * -AUBO_INVL_ARGUMENT
01749  * -AUBO_BAD_STATE
01750  *
01751  * @throws arcs::common_interface::AuboException
01752  *
01753  * @par Python 函数原型
01754  * setSupvOrient(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
01755  * float, arg2: float, arg3: bool) -> int
01756  *
01757  * @par Lua 函数原型
01758  * setSupvOrient(frame: table, max_angle: number, max_rot: number,
01759  * outside: boolean) -> nil
01760  * \endchinese
01761  * \english
01762  * setSupvOrient is used to set up an supervision for the tool orientation.
01763  * The supervision is activated when Force Control is activated with the
01764  * instruction FCAct.
01765  *
01766  * An orientation supervision is set up by defining a maximum angle and a
01767  * maximum rotation from a reference orientation. The reference orientation
01768  * is either defined by the current z direction of the tool, or by
01769  * specifying an orientation in relation to the z direction of the work
01770  * object.
01771  *
01772  * Once activated, the tool orientation must be within the limits otherwise
01773  * the supervision will stop the execution.
01774  *
01775  * @param frame
01776  * @param max_angle
01777  * @param max_rot
01778  * @param outside
01779  *

```

```

01780     * @return Return 0 if succeeded; return error code if failed
01781     * AUBO_BUSY
01782     * AUBO_BAD_STATE
01783     * -AUBO_INVL_ARGUMENT
01784     * -AUBO_BAD_STATE
01785     *
01786     * @throws arcs::common_interface::AuboException
01787     *
01788     * @par Python Function Prototype
01789     * setSupvOrient(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
01790     * float, arg2: float, arg3: bool) -> int
01791     *
01792     * @par Lua Function Prototype
01793     * setSupvOrient(frame: table, max_angle: number, max_rot: number,
01794     * outside: boolean) -> nil
01795     * \endenglish
01796     */
01797     int setSupvOrient(const std::vector<double> &frame, double max_angle,
01798                     double max_rot, bool outside);
01799
01800 /**
01801  * @ingroup ForceControl
01802  * \chinese
01803  * setSupvPosBox 用于在力控中设置位置监督。监督在通过 FCAct 指令激活力控时被激活。
01804  * 位置监督通过为 TCP 定义空间体积来设置。一旦激活，如果 TCP 超出该体积，监督将停止执行。
01805  *
01806  * @param frame
01807  * @param box
01808  *
01809  * @return 成功返回 0；失败返回错误码
01810  * AUBO_BUSY
01811  * AUBO_BAD_STATE
01812  * -AUBO_INVL_ARGUMENT
01813  * -AUBO_BAD_STATE
01814  *
01815  * @throws arcs::common_interface::AuboException
01816  *
01817  * @par Python 函数原型
01818  * setSupvPosBox(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
01819  * List[float[6]]) -> int
01820  *
01821  * @par Lua 函数原型
01822  * setSupvPosBox(frame: table, box: table) -> nil
01823  * \endchinese
01824  * \english
01825  * setSupvPosBox is used to set up position supervision in Force Control.
01826  * Supervision is activated when Force Control is activated with the
01827  * instruction FCAct. Position supervision is set up by defining a volume in
01828  * space for the TCP. Once activated, the supervision will stop the
01829  * execution if the TCP is outside this volume.
01830  *
01831  * @param frame
01832  * @param box
01833  *
01834  * @return Return 0 if succeeded; return error code if failed
01835  * AUBO_BUSY
01836  * AUBO_BAD_STATE
01837  * -AUBO_INVL_ARGUMENT
01838  * -AUBO_BAD_STATE
01839  *
01840  * @throws arcs::common_interface::AuboException
01841  *
01842  * @par Python Function Prototype
01843  * setSupvPosBox(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
01844  * List[float[6]]) -> int
01845  *
01846  * @par Lua Function Prototype
01847  * setSupvPosBox(frame: table, box: table) -> nil
01848  * \endenglish
01849  */
01850     int setSupvPosBox(const std::vector<double> &frame, const Box &box);
01851
01852 /**
01853  * @ingroup ForceControl
01854  * \chinese
01855  *
01856  * @param frame
01857  * @param cylinder
01858  *
01859  * @return 成功返回 0；失败返回错误码
01860  * AUBO_BUSY
01861  * AUBO_BAD_STATE
01862  * -AUBO_INVL_ARGUMENT
01863  * -AUBO_BAD_STATE
01864  *
01865  * @throws arcs::common_interface::AuboException
01866  *

```



```

01867      * @par Python 函数原型
01868      * setSupvPosCylinder(self: pyaubo_sdk.ForceControl, arg0: List[float],
01869      * arg1: List[float[5]]) -> int
01870      *
01871      * @par Lua 函数原型
01872      * setSupvPosCylinder(frame: table, cylinder: table) -> nil
01873      *
01874      * \endchinese
01875      * \english
01876      *
01877      * @param frame
01878      * @param cylinder
01879      *
01880      * @return Return 0 if succeeded; return error code if failed
01881      * AUBO_BUSY
01882      * AUBO_BAD_STATE
01883      * -AUBO_INVL_ARGUMENT
01884      * -AUBO_BAD_STATE
01885      *
01886      * @throws arcs::common_interface::AuboException
01887      *
01888      * @par Python Function Prototype
01889      * setSupvPosCylinder(self: pyaubo_sdk.ForceControl, arg0: List[float],
01890      * arg1: List[float[5]]) -> int
01891      *
01892      * @par Lua Function Prototype
01893      * setSupvPosCylinder(frame: table, cylinder: table) -> nil
01894      *
01895      * \endenglish
01896      */
01897  int setSupvPosCylinder(const std::vector<double> &frame,
01898                        const Cylinder &cylinder);
01899
01900  /**
01901   * @ingroup ForceControl
01902   * \chinese
01903   *
01904   * @param frame
01905   * @param sphere
01906   *
01907   * @return 成功返回 0; 失败返回错误码
01908   * AUBO_BUSY
01909   * AUBO_BAD_STATE
01910   * -AUBO_INVL_ARGUMENT
01911   * -AUBO_BAD_STATE
01912   *
01913   * @throws arcs::common_interface::AuboException
01914   *
01915   * @par Python 函数原型
01916   * setSupvPosSphere(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
01917   * List[float[3]]) -> int
01918   *
01919   * @par Lua 函数原型
01920   * setSupvPosSphere(frame: table, sphere: table) -> nil
01921   *
01922   * \endchinese
01923   * \english
01924   *
01925   * @param frame
01926   * @param sphere
01927   *
01928   * @return Return 0 if succeeded; return error code if failed
01929   * AUBO_BUSY
01930   * AUBO_BAD_STATE
01931   * -AUBO_INVL_ARGUMENT
01932   * -AUBO_BAD_STATE
01933   *
01934   * @throws arcs::common_interface::AuboException
01935   *
01936   * @par Python Function Prototype
01937   * setSupvPosSphere(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1:
01938   * List[float[3]]) -> int
01939   *
01940   * @par Lua Function Prototype
01941   * setSupvPosSphere(frame: table, sphere: table) -> nil
01942   *
01943   * \endenglish
01944   */
01945  int setSupvPosSphere(const std::vector<double> &frame,
01946                      const Sphere &sphere);
01947
01948  /**
01949   * @ingroup ForceControl
01950   * \chinese
01951   * setSupvReoriSpeed 用于在力控中设置重新定向速度监督。监督在通过 FCAct 指令激活力控时被激活。
01952   *
01953   * 重新定向速度监督通过定义工作对象坐标系轴周围重新定向速度的最小和最大限制来设置。

```

```

01954      * 一旦激活，如果重新定向速度的值过高，监督将停止执行。
01955      *
01956      * 有两种速度监督：FCSupvReoriSpeed 和 FCSupvTCPSPeed，后者在第 199 页的 FCSupvTCPSPeed 部分有描述。
01957      * 可能需要两种监督，因为：
01958      * - 机器人轴可以在 TCP 静止时高速旋转。
01959      * - 当 TCP 距离旋转轴较远时，轴的微小旋转可能导致 TCP 的高速运动。
01960      *
01961      * @param speed_limit 速度限制
01962      * @param outside 是否在范围外有效
01963      * @param timeout 超时时间
01964      *
01965      * @return 成功返回 0；失败返回错误码
01966      * AUBO_BUSY
01967      * AUBO_BAD_STATE
01968      * -AUBO_INVL_ARGUMENT
01969      * -AUBO_BAD_STATE
01970      *
01971      * @throws arcs::common_interface::AuboException
01972      *
01973      * @par Python 函数原型
01974      * setSupvReoriSpeed(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: bool, arg2: float) -> int
01975      *
01976      * @par Lua 函数原型
01977      * setSupvReoriSpeed(speed_limit: table, outside: boolean, timeout: number) -> nil
01978      * \endchinese
01979      * \english
01980      * setSupvReoriSpeed is used to set up reorientation speed supervision in Force Control. The supervision is
activated when Force Control is activated with the instruction FCAct.
01981      *
01982      * The reorientation speed supervision is set up by defining minimum and maximum limits for the reorientation
speed around the axis of the work object coordinate system. Once activated, the supervision will stop the execution
if the values of the reorientation speed are too high.
01983      *
01984      * There are two speed supervisions: FCSupvReoriSpeed and FCSupvTCPSPeed, which is described in section
FCSupvTCPSPeed on page 199.
01985      * Both supervisions may be required because:
01986      * - A robot axis can rotate with high speed while the TCP is stationary.
01987      * - The TCP can be far from the rotating axis and a small axis rotation may result in a high speed movement of
the TCP.
01988      *
01989      * @param speed_limit
01990      * @param outside
01991      * @param timeout
01992      *
01993      * @return Return 0 if succeeded; return error code if failed
01994      * AUBO_BUSY
01995      * AUBO_BAD_STATE
01996      * -AUBO_INVL_ARGUMENT
01997      * -AUBO_BAD_STATE
01998      *
01999      * @throws arcs::common_interface::AuboException
02000      *
02001      * @par Python Function Prototype
02002      * setSupvReoriSpeed(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: bool, arg2: float) -> int
02003      *
02004      * @par Lua Function Prototype
02005      * setSupvReoriSpeed(speed_limit: table, outside: boolean, timeout: number) -> nil
02006      * \endenglish
02007      */
02008      int setSupvReoriSpeed(const std::vector<double> &speed_limit, bool outside,
double timeout);
02009
02010
02011      /**
02012      * @ingroup ForceControl
02013      * \chinese
02014      * setSupvTcpspeed 用于在力控中设置 TCP 速度监督。监督在通过 FCAct 指令激活力控时被激活。
02015      * TCP 速度监督通过定义工作对象坐标系各方向上的最小和最大速度限制来设置。
02016      * 一旦激活，如果检测到过高的 TCP 速度值，监督将停止执行。
02017      *
02018      * 有两种速度监督：FCSupvTCPSPeed 和 FCSupvReoriSpeed，后者在第 197 页的 FCSupvReoriSpeed 部分有描述。
02019      *
02020      * 可能需要两种监督，因为：
02021      * - 机器人轴可以在 TCP 静止时高速旋转。
02022      * - 当 TCP 距离旋转轴较远时，轴的微小旋转可能导致 TCP 的高速运动。
02023      *
02024      * @param speed_limit 速度限制
02025      * @param outside 是否在范围外有效
02026      * @param timeout 超时时间
02027      *
02028      * @return 成功返回 0；失败返回错误码
02029      * AUBO_BUSY
02030      * AUBO_BAD_STATE
02031      * -AUBO_INVL_ARGUMENT
02032      * -AUBO_BAD_STATE
02033      *
02034      * @throws arcs::common_interface::AuboException
02035      *

```

```

02036 * @par Python 函数原型
02037 * setSupvTcpSpeed(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: bool, arg2: float) -> int
02038 *
02039 * @par Lua 函数原型
02040 * setSupvTcpSpeed(speed_limit: table, outside: boolean, timeout: number) -> nil
02041 * \endchinese
02042 * \english
02043 * setSupvTcpSpeed is used to set up TCP speed supervision in Force Control.
02044 * The supervision is activated when Force Control is activated with the
02045 * instruction FCAct. The TCP speed supervision is set up by defining
02046 * minimum and maximum limits for the TCP speed in the directions of the
02047 * work object coordinate system. Once activated, the supervision will stop
02048 * the execution if too high TCP speed values are detected.
02049 *
02050 * There are two speed supervisions: FCSupvTCPSpeed and FCSupvReoriSpeed,
02051 * which is described in section FCSupvReoriSpeed on page 197.
02052 *
02053 * Both supervisions may be required because:
02054 * - A robot axis can rotate with high speed while the TCP is stationary.
02055 * - The TCP can be far from the rotating axis and a small axis rotation may
02056 *   result in a high speed movement of the TCP.
02057 *
02058 * @param speed_limit
02059 * @param outside
02060 * @param timeout
02061 *
02062 * @return Return 0 if succeeded; return error code if failed
02063 * AUBO_BUSY
02064 * AUBO_BAD_STATE
02065 * -AUBO_INVL_ARGUMENT
02066 * -AUBO_BAD_STATE
02067 *
02068 * @throws arcs::common_interface::AuboException
02069 *
02070 * @par Python Function Prototype
02071 * setSupvTcpSpeed(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: bool, arg2: float) -> int
02072 *
02073 * @par Lua Function Prototype
02074 * setSupvTcpSpeed(speed_limit: table, outside: boolean, timeout: number) -> nil
02075 * \endenglish
02076 */
02077 int setSupvTcpSpeed(const std::vector<double> &speed_limit, bool outside,
02078                    double timeout);
02079
02080 // 设置低通滤波器
02081 // --- force frame filter: 过滤测量到的力/力矩
02082 // +++ force loop filter: 力控输出参考速度的滤波器
02083
02084 /**
02085 * @ingroup ForceControl
02086 * \chinese
02087 * 设置低通滤波器
02088 *
02089 * --- force frame filter: 过滤测量到的力/力矩
02090 * +++ force loop filter: 力控输出参考速度的滤波器
02091 *
02092 * @param cutoff_freq 截止频率
02093 *
02094 * @return 成功返回 0; 失败返回错误码
02095 * AUBO_BUSY
02096 * AUBO_BAD_STATE
02097 * -AUBO_INVL_ARGUMENT
02098 * -AUBO_BAD_STATE
02099 *
02100 * @throws arcs::common_interface::AuboException
02101 *
02102 * @par Python 函数原型
02103 * setLpFilter(self: pyaubo_sdk.ForceControl, arg0: List[float]) -> int
02104 *
02105 * @par Lua 函数原型
02106 * setLpFilter(cutoff_freq: table) -> nil
02107 * \endchinese
02108 * \english
02109 * Set low-pass filter
02110 *
02111 * --- force frame filter: filter measured force/torque
02112 * +++ force loop filter: filter for force control output reference speed
02113 *
02114 * FCSetLPFilterTune is used to change the response of force loop according to
02115 * the description in Damping and LP-filter on page 103.
02116 *
02117 * @param cutoff_freq Cutoff frequency
02118 *
02119 * @return Return 0 if succeeded; return error code if failed
02120 * AUBO_BUSY
02121 * AUBO_BAD_STATE
02122 * -AUBO_INVL_ARGUMENT

```

```

02123     * -AUBO_BAD_STATE
02124     *
02125     * @throws arcs::common_interface::AuboException
02126     *
02127     * @par Python Function Prototype
02128     * setLpFilter(self: pyaubo_sdk.ForceControl, arg0: List[float]) -> int
02129     *
02130     * @par Lua Function Prototype
02131     * setLpFilter(cutoff_freq: table) -> nil
02132     * \endenglish
02133     */
02134     int setLpFilter(const std::vector<double> &cutoff_freq);
02135
02136     /**
02137     * @ingroup ForceControl
02138     * \chinese
02139     * 重置低通滤波器
02140     *
02141     * @return 成功返回 0; 失败返回错误码
02142     * AUBO_BUSY
02143     * AUBO_BAD_STATE
02144     * -AUBO_BAD_STATE
02145     *
02146     * @throws arcs::common_interface::AuboException
02147     *
02148     * @par Python 函数原型
02149     * resetLpFilter(self: pyaubo_sdk.ForceControl) -> int
02150     *
02151     * @par Lua 函数原型
02152     * resetLpFilter() -> nil
02153     *
02154     * @par JSON-RPC 请求示例
02155     * {"jsonrpc": "2.0", "method": "rob1.ForceControl.resetLpFilter", "params": [], "id": 1}
02156     *
02157     * @par JSON-RPC 响应示例
02158     * {"id": 1, "jsonrpc": "2.0", "result": 0}
02159     * \endchinese
02160     * \english
02161     * Reset low-pass filter
02162     *
02163     * @return Return 0 if succeeded; return error code if failed
02164     * AUBO_BUSY
02165     * AUBO_BAD_STATE
02166     * -AUBO_BAD_STATE
02167     *
02168     * @throws arcs::common_interface::AuboException
02169     *
02170     * @par Python Function Prototype
02171     * resetLpFilter(self: pyaubo_sdk.ForceControl) -> int
02172     *
02173     * @par Lua Function Prototype
02174     * resetLpFilter() -> nil
02175     *
02176     * @par JSON-RPC Request example
02177     * {"jsonrpc": "2.0", "method": "rob1.ForceControl.resetLpFilter", "params": [], "id": 1}
02178     *
02179     * @par JSON-RPC Response example
02180     * {"id": 1, "jsonrpc": "2.0", "result": 0}
02181     * \endenglish
02182     */
02183     int resetLpFilter();
02184
02185     /**
02186     * @ingroup ForceControl
02187     * \chinese
02188     * speedChangeEnable 用于激活 FC SpeedChange 功能, 并设置期望的参考力和恢复行为。
02189     * 当 FC SpeedChange 功能被激活时, 机器人速度会根据测量信号与参考值的接近程度自动降低或提高。
02190     *
02191     * @param ref_force 参考力
02192     *
02193     * @return 成功返回 0; 失败返回错误码
02194     * AUBO_BUSY
02195     * AUBO_BAD_STATE
02196     * -AUBO_INVL_ARGUMENT
02197     * -AUBO_BAD_STATE
02198     *
02199     * @throws arcs::common_interface::AuboException
02200     *
02201     * @par Python 函数原型
02202     * speedChangeEnable(self: pyaubo_sdk.ForceControl, arg0: float) -> int
02203     *
02204     * @par Lua 函数原型
02205     * speedChangeEnable(ref_force: number) -> nil
02206     * \endchinese
02207     * \english
02208     * The speedChangeEnable is used to activate FC SpeedChange function with desired
02209     * reference and recover behavior. When FC SpeedChange function is active,

```

```

02210     * the robot speed will be reduced/increased in order to keep the measured
02211     * signal close to the reference.
02212     *
02213     * @param ref_force
02214     *
02215     * @return Return 0 if succeeded; return error code if failed
02216     * AUBO_BUSY
02217     * AUBO_BAD_STATE
02218     * -AUBO_INVL_ARGUMENT
02219     * -AUBO_BAD_STATE
02220     *
02221     * @throws arcs::common_interface::AuboException
02222     *
02223     * @par Python Function Prototype
02224     * speedChangeEnable(self: pyaubo_sdk.ForceControl, arg0: float) -> int
02225     *
02226     * @par Lua Function Prototype
02227     * speedChangeEnable(ref_force: number) -> nil
02228     * \endenglish
02229     */
02230 int speedChangeEnable(double ref_force);
02231
02232 /**
02233  * @ingroup ForceControl
02234  * \chinese
02235  * 停用 FC SpeedChange 功能。
02236  *
02237  * @return 成功返回 0；失败返回错误码
02238  * AUBO_BUSY
02239  * AUBO_BAD_STATE
02240  * -AUBO_BAD_STATE
02241  *
02242  * @throws arcs::common_interface::AuboException
02243  *
02244  * @par Python 函数原型
02245  * speedChangeDisable(self: pyaubo_sdk.ForceControl) -> int
02246  *
02247  * @par Lua 函数原型
02248  * speedChangeDisable() -> nil
02249  *
02250  * @par JSON-RPC 请求示例
02251  * {"jsonrpc": "2.0", "method": "rob1.ForceControl.speedChangeDisable", "params": [], "id": 1}
02252  *
02253  * @par JSON-RPC 响应示例
02254  * {"id": 1, "jsonrpc": "2.0", "result": 0}
02255  * \endchinese
02256  * \english
02257  * Deactivate FC SpeedChange function.
02258  *
02259  * @return Return 0 if succeeded; return error code if failed
02260  * AUBO_BUSY
02261  * AUBO_BAD_STATE
02262  * -AUBO_BAD_STATE
02263  *
02264  * @throws arcs::common_interface::AuboException
02265  *
02266  * @par Python Function Prototype
02267  * speedChangeDisable(self: pyaubo_sdk.ForceControl) -> int
02268  *
02269  * @par Lua Function Prototype
02270  * speedChangeDisable() -> nil
02271  *
02272  * @par JSON-RPC Request example
02273  * {"jsonrpc": "2.0", "method": "rob1.ForceControl.speedChangeDisable", "params": [], "id": 1}
02274  *
02275  * @par JSON-RPC Response example
02276  * {"id": 1, "jsonrpc": "2.0", "result": 0}
02277  * \endenglish
02278  */
02279 int speedChangeDisable();
02280
02281 /**
02282  * @ingroup ForceControl
02283  * \chinese
02284  * speedChangeTune 用于将 FC SpeedChange 系统参数设置为新值。
02285  *
02286  * @param speed_levels 速度级别
02287  * @param speed_ratio_min 最小速度比例
02288  *
02289  * @return 成功返回 0；失败返回错误码
02290  * AUBO_BUSY
02291  * AUBO_BAD_STATE
02292  * -AUBO_INVL_ARGUMENT
02293  * -AUBO_BAD_STATE
02294  *
02295  * @throws arcs::common_interface::AuboException
02296  *

```

```

02297 * @par Python 函数原型
02298 * speedChangeTune(self: pyaubo_sdk.ForceControl, arg0: int, arg1: float) -> int
02299 *
02300 * @par Lua 函数原型
02301 * speedChangeTune(speed_levels: number, speed_ratio_min: number) -> nil
02302 * \endchinese
02303 * \english
02304 * speedChangeTune is used to set FC SpeedChange system parameters to a new value.
02305 *
02306 * @param speed_levels
02307 * @param speed_ratio_min
02308 *
02309 * @return Return 0 if succeeded; return error code if failed
02310 * AUBO_BUSY
02311 * AUBO_BAD_STATE
02312 * -AUBO_INVL_ARGUMENT
02313 * -AUBO_BAD_STATE
02314 *
02315 * @throws arcs::common_interface::AuboException
02316 *
02317 * @par Python Function Prototype
02318 * speedChangeTune(self: pyaubo_sdk.ForceControl, arg0: int, arg1: float) -> int
02319 *
02320 * @par Lua Function Prototype
02321 * speedChangeTune(speed_levels: number, speed_ratio_min: number) -> nil
02322 * \endenglish
02323 */
02324 int speedChangeTune(int speed_levels, double speed_ratio_min);
02325
02326 /* Defines how many Newtons are required to make the robot move 1 m/s. The
02327    higher the value, the less responsive the robot gets.
02328    The damping can be tuned (as a percentage of the system parameter values)
02329    by the RAPID instruction FCAct. */
02330 // 设置阻尼系数，阻尼的系统参数需要通过配置文件设置
02331 // [damping_fx, damping_fy, damping_fz, damping_tx, damping_ty, damping_tz]
02332 // A value between min and 10,000,000 Ns/m.
02333 // A value between minimum and 10,000,000 Nms/rad.
02334
02335 /**
02336  * @ingroup ForceControl
02337  * \chinese
02338  * setDamping 用于在力控坐标系中调整阻尼。可调参数包括扭矩 x 方向到扭矩 z 方向的阻尼（见第 255 页）以及力 x 方向到力 z
    方向的阻尼（见第 254 页）。
02339  *
02340  * 阻尼可以通过配置文件或 FCAct 指令设置。不同之处在于本指令可在力控激活时使用。FCSetDampingTune 调整的是 FCAct 指令
    设置的实际值，而不是配置文件中的值。
02341  *
02342  * @param damping 阻尼参数
02343  * @param ramp_time 斜坡时间
02344  *
02345  * @return 成功返回 0；失败返回错误码
02346  * AUBO_BUSY
02347  * AUBO_BAD_STATE
02348  * -AUBO_INVL_ARGUMENT
02349  * -AUBO_BAD_STATE
02350  *
02351  * @throws arcs::common_interface::AuboException
02352  *
02353  * @par Python 函数原型
02354  * setDamping(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: float) -> int
02355  *
02356  * @par Lua 函数原型
02357  * setDamping(damping: table, ramp_time: number) -> nil
02358  * \endchinese
02359  * \english
02360  * setDamping is used to tune the damping in the force control coordinate systems. The parameters tuned are
    those described in Damping in Torque x Direction - Damping in Torque z Direction on page 255 and Damping in Force x
    Direction - Damping in Force z Direction on page 254.
02361  *
02362  * Damping can be set in the configuration file or by the instruction FCAct. The difference is that this
    instruction can be used when force control is active. FCSetDampingTune tunes the actual values set by the
    instruction FCAct, not the value in the configuration file.
02363  *
02364  * @param damping
02365  * @param ramp_time
02366  *
02367  * @return Return 0 if succeeded; return error code if failed
02368  * AUBO_BUSY
02369  * AUBO_BAD_STATE
02370  * -AUBO_INVL_ARGUMENT
02371  * -AUBO_BAD_STATE
02372  *
02373  * @throws arcs::common_interface::AuboException
02374  *
02375  * @par Python Function Prototype
02376  * setDamping(self: pyaubo_sdk.ForceControl, arg0: List[float], arg1: float) -> int
02377  *

```

```

02378     * @par Lua Function Prototype
02379     * setDamping(damping: table, ramp_time: number) -> nil
02380     * \endenglish
02381     */
02382 int setDamping(const std::vector<double> &damping, double ramp_time);
02383
02384 /**
02385  * @ingroup ForceControl
02386  * \chinese
02387  * 重置阻尼参数
02388  *
02389  * @return 成功返回 0；失败返回错误码
02390  * AUBO_BUSY
02391  * AUBO_BAD_STATE
02392  * -AUBO_BAD_STATE
02393  *
02394  * @throws arcs::common_interface::AuboException
02395  *
02396  * @par Python 函数原型
02397  * resetDamping(self: pyaubo_sdk.ForceControl) -> int
02398  *
02399  * @par Lua 函数原型
02400  * resetDamping() -> nil
02401  *
02402  * @par JSON-RPC 请求示例
02403  * {"jsonrpc": "2.0", "method": "rob1.ForceControl.resetDamping", "params": [], "id": 1}
02404  *
02405  * @par JSON-RPC 响应示例
02406  * {"id": 1, "jsonrpc": "2.0", "result": 0}
02407  * \endchinese
02408  * \english
02409  * Reset damping parameters
02410  *
02411  * @return Return 0 if succeeded; return error code if failed
02412  * AUBO_BUSY
02413  * AUBO_BAD_STATE
02414  * -AUBO_BAD_STATE
02415  *
02416  * @throws arcs::common_interface::AuboException
02417  *
02418  * @par Python Function Prototype
02419  * resetDamping(self: pyaubo_sdk.ForceControl) -> int
02420  *
02421  * @par Lua Function Prototype
02422  * resetDamping() -> nil
02423  *
02424  * @par JSON-RPC Request example
02425  * {"jsonrpc": "2.0", "method": "rob1.ForceControl.resetDamping", "params": [], "id": 1}
02426  *
02427  * @par JSON-RPC Response example
02428  * {"id": 1, "jsonrpc": "2.0", "result": 0}
02429  * \endenglish
02430  */
02431 int resetDamping();
02432
02433 /**
02434  * @ingroup ForceControl
02435  * \chinese
02436  * 启用软浮动功能。
02437  *
02438  * @return 成功返回 0；失败返回错误码
02439  * AUBO_BUSY
02440  * AUBO_BAD_STATE
02441  * -AUBO_BAD_STATE
02442  *
02443  * @throws arcs::common_interface::AuboException
02444  *
02445  * @par JSON-RPC 请求示例
02446  * {"jsonrpc": "2.0", "method": "rob1.ForceControl.softFloatEnable", "params": [], "id": 1}
02447  *
02448  * @par JSON-RPC 响应示例
02449  * {"id": 1, "jsonrpc": "2.0", "result": 0}
02450  * \endchinese
02451  * \english
02452  * Enable soft float function.
02453  *
02454  * @return Return 0 if succeeded; return error code if failed
02455  * AUBO_BUSY
02456  * AUBO_BAD_STATE
02457  * -AUBO_BAD_STATE
02458  *
02459  * @throws arcs::common_interface::AuboException
02460  *
02461  * @par JSON-RPC Request example
02462  * {"jsonrpc": "2.0", "method": "rob1.ForceControl.softFloatEnable", "params": [], "id": 1}
02463  *
02464  * @par JSON-RPC Response example

```



```

02465     * {"id":1,"jsonrpc":"2.0","result":0}
02466     * \endenglish
02467     */
02468     int softFloatEnable();
02469
02470     /**
02471     * @ingroup ForceControl
02472     * \chinese
02473     * 停用软浮动功能。
02474     *
02475     * @return 成功返回 0；失败返回错误码
02476     * AUBO_BUSY
02477     * AUBO_BAD_STATE
02478     * -AUBO_BAD_STATE
02479     *
02480     * @throws arcs::common_interface::AuboException
02481     *
02482     * @par JSON-RPC 请求示例
02483     * {"jsonrpc":"2.0","method":"robl.ForceControl.softFloatDisable","params":[],"id":1}
02484     *
02485     * @par JSON-RPC 响应示例
02486     * {"id":1,"jsonrpc":"2.0","result":-1}
02487     * \endchinese
02488     * \english
02489     * Disable soft float function.
02490     *
02491     * @return Return 0 if succeeded; return error code if failed
02492     * AUBO_BUSY
02493     * AUBO_BAD_STATE
02494     * -AUBO_BAD_STATE
02495     *
02496     * @throws arcs::common_interface::AuboException
02497     *
02498     * @par JSON-RPC Request example
02499     * {"jsonrpc":"2.0","method":"robl.ForceControl.softFloatDisable","params":[],"id":1}
02500     *
02501     * @par JSON-RPC Response example
02502     * {"id":1,"jsonrpc":"2.0","result":-1}
02503     * \endenglish
02504     */
02505     int softFloatDisable();
02506
02507     /**
02508     * @ingroup ForceControl
02509     * \chinese
02510     * 返回是否开启了软浮动
02511     *
02512     * @return
02513     *
02514     * @throws arcs::common_interface::AuboException
02515     *
02516     * @par JSON-RPC 请求示例
02517     * {"jsonrpc":"2.0","method":"robl.ForceControl.isSoftFloatEnabled","params":[],"id":1}
02518     *
02519     * @par JSON-RPC 响应示例
02520     * {"id":1,"jsonrpc":"2.0","result":false}
02521     * \endchinese
02522     * \english
02523     * Returns whether soft float is enabled
02524     *
02525     * @return
02526     *
02527     * @throws arcs::common_interface::AuboException
02528     *
02529     * @par JSON-RPC Request example
02530     * {"jsonrpc":"2.0","method":"robl.ForceControl.isSoftFloatEnabled","params":[],"id":1}
02531     *
02532     * @par JSON-RPC Response example
02533     * {"id":1,"jsonrpc":"2.0","result":false}
02534     * \endenglish
02535     */
02536     bool isSoftFloatEnabled();
02537
02538     /**
02539     * @ingroup ForceControl
02540     * \chinese
02541     * 设置软浮动参数
02542     *
02543     * @param joint_softfloat 是否在关节空间启用软浮动
02544     * @param select 选择哪些自由度启用软浮动
02545     * @param stiff_percent 刚度百分比
02546     * @param stiff_damp_ratio 刚度阻尼比
02547     * @param force_threshold 力阈值
02548     * @param force_limit 力限制
02549     * @return 返回 0 表示成功，其他为错误码
02550     * \endchinese
02551     * \english

```



```

02552     * Set soft float parameters
02553     *
02554     * @param joint_softfloat Whether to enable soft float in joint space
02555     * @param select Select which degrees of freedom to enable soft float
02556     * @param stiff_percent Stiffness percentage
02557     * @param stiff_damp_ratio Stiffness damping ratio
02558     * @param force_threshold Force threshold
02559     * @param force_limit Force limit
02560     * @return Return 0 if succeeded, otherwise error code
02561     * \endenglish
02562     */
02563     int setSoftFloatParams(bool joint_space, const std::vector<bool> &select,
02564                           const std::vector<double> &stiff_percent,
02565                           const std::vector<double> &stiff_damp_ratio,
02566                           const std::vector<double> &force_threshold,
02567                           const std::vector<double> &force_limit);
02568
02569     /**
02570     * @ingroup ForceControl
02571     * \chinese
02572     * 检测工具和外部物体的接触
02573     *
02574     * @param direction
02575     * 预期的接触方向, 如果所有的元素为 0, 表示检测所有方向的接触
02576     * @return
02577     * 返回从当前点回退到碰撞开始点的周期步数, 如果返回值为 0, 表示没有接触
02578     * \endchinese
02579     * \english
02580     * Detect contact between the tool and external objects
02581     *
02582     * @param direction
02583     * Expected contact direction. If all elements are 0, detect contact in all directions.
02584     * @return
02585     * Returns the number of cycle steps back from the current point to the collision start point. If the return
02586     * value is 0, no contact is detected.
02587     * \endenglish
02588     */
02589     int toolContact(const std::vector<bool> &direction);
02590
02591     /**
02592     * @ingroup ForceControl
02593     * \chinese
02594     * @brief 获取历史关节角度
02595     *
02596     * 根据给定的周期步数, 从关节状态历史中回退指定数量的周期, 获取当时的关节角度数据。
02597     *
02598     * @param steps
02599     * 需要回退的周期数 (单位: 控制周期数), 值越大表示获取越早的历史数据
02600     * @return std::vector<double>
02601     * 对应时间点的各关节角度 (单位: 弧度)
02602     * \endchinese
02603     * \english
02604     * @brief Get historical joint positions
02605     *
02606     * According to the given number of cycle steps, go back the specified number of cycles from the joint state
02607     * history to obtain the joint position data at that time.
02608     *
02609     * @param steps
02610     * Number of cycles to go back (unit: control cycles), the larger the value, the earlier the historical data is
02611     * obtained
02612     * @return std::vector<double>
02613     * Joint positions (unit: radians) at the corresponding time point
02614     * \endenglish
02615     */
02616     std::vector<double> getActualJointPositionsHistory(int steps);
02617
02618     protected:
02619     void *d_{ nullptr };
02620 };
02621 using ForceControlPtr = std::shared_ptr<ForceControl>;
02622 } // namespace common_interface
02623 } // namespace arcs
02624
02625 #endif // AUBO_SDK_FORCE_CONTROL_INTERFACE_H

```

12.22 include/aubo/robot/io_control.h 文件参考

IO 控制接口

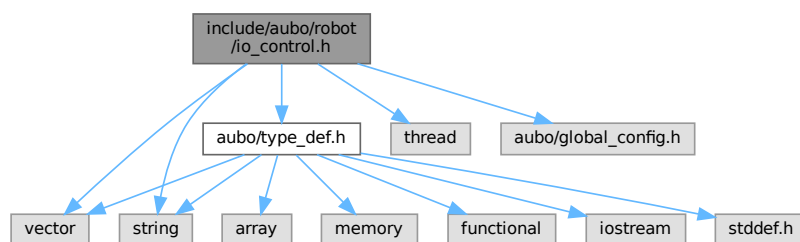
```

#include <vector>
#include <thread>
#include <string>
#include <aubo/global_config.h>

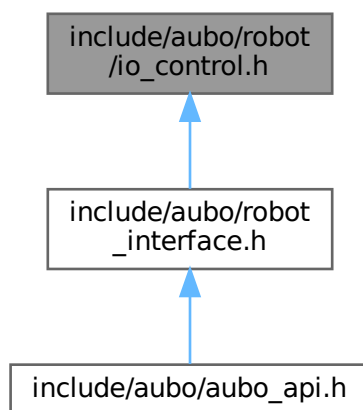
```

```
#include <aubo/type_def.h>
```

io_control.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::common_interface::IoControl`

命名空间

- namespace `arcs`
- namespace `arcs::common_interface`

类型定义

- using `arcs::common_interface::IoControlPtr` = `std::shared_ptr<IoControl>`

12.22.1 详细描述

IO 控制接口

在文件 `io_control.h` 中定义.

12.23 io_control.h

[浏览该文件的文档.](#)

```

00001 /** @file io_control.h
00002  * @brief IO 控制接口
00003  */
00004 #ifndef AUBO_SDK_IO_CONTROL_INTERFACE_H
00005 #define AUBO_SDK_IO_CONTROL_INTERFACE_H
00006
00007 #include <vector>
00008 #include <thread>
00009 #include <string>
00010
00011 #include <aubo/global_config.h>
00012 #include <aubo/type_def.h>
00013
00014 namespace arcs {
00015     namespace common_interface {
00016
00017         /**
00018          * @defgroup IoControl IoControl (IO 输入输出控制)
00019          * @ingroup RobotInterface
00020          * \chinese
00021          * IoControl 类提供了一系列的接口对机器人标配的一些数字、模拟 IO 进行配置，输出状态设置、读取
00022          *
00023          * 1. 获取各种 IO 的数量 \n
00024          * 2. 配置 IO 的输入输出功能 \n
00025          * 3. 可配置 IO 的配置 \n
00026          * 4. 模拟 IO 的输入输出范围设置、读取
00027          *
00028          * 标准数字输入输出：控制柜 IO 面板上的标准 IO \n
00029          * 工具端数字输入输出：通过工具末端航插暴露的数字 IO \n
00030          * 可配置输入输出：可以配置为安全 IO 或者普通数字 IO \n
00031          * \endchinese
00032          * \english
00033          * The IoControl class provides a series of interfaces for configuring and
00034          * reading the robot's standard digital and analog IO, as well as setting output
00035          * states.
00036          *
00037          * 1. Get the number of various IOs \n
00038          * 2. Configure IO input/output functions \n
00039          * 3. Configuration of configurable IOs \n
00040          * 4. Set and read the input/output range of analog IOs
00041          *
00042          * Standard digital input/output: Standard IOs on the control cabinet IO panel
00043          * \n Tool digital input/output: Digital IOs exposed via the tool-end connector
00044          * \n Configurable input/output: Can be configured as safety IO or general
00045          * digital IO \n \endenglish
00046          */
00047         class ARCS_ABI_EXPORT IoControl
00048         {
00049         public:
00050             IoControl();
00051             virtual ~IoControl();
00052
00053             /**
00054              * @ingroup IoControl
00055              * \chinese
00056              * 获取标准数字输入数量
00057              *
00058              * @return 标准数字输入数量
00059              *
00060              * @throws arcs::common_interface::AuboException
00061              *
00062              * @par Python 函数原型
00063              * getStandardDigitalInputNum(self: pyaubo_sdk.IoControl) -> int
00064              *
00065              * @par Lua 函数原型
00066              * getStandardDigitalInputNum() -> number
00067              *
00068              * @par Lua 示例
00069              * num = getStandardDigitalInputNum()
00070              *
00071              * @par JSON-RPC 请求示例
00072              * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardDigitalInputNum", "params": [], "id": 1}
00073              *
00074              * @par JSON-RPC 响应示例
00075              * {"id": 1, "jsonrpc": "2.0", "result": 16}
00076              *
00077              * \endchinese
00078              * \english
00079              * Get the number of standard digital inputs.
00080              *
00081              * @return Number of standard digital inputs.
00082              *
00083              * @throws arcs::common_interface::AuboException

```

```

00084      *
00085      * @par Python function prototype
00086      * getStandardDigitalInputNum(self: pyaubo_sdk.IoControl) -> int
00087      *
00088      * @par Lua function prototype
00089      * getStandardDigitalInputNum() -> number
00090      *
00091      * @par Lua example
00092      * num = getStandardDigitalInputNum()
00093      *
00094      * @par JSON-RPC request example
00095      * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardDigitalInputNum", "params": [], "id": 1}
00096      *
00097      * @par JSON-RPC response example
00098      * {"id": 1, "jsonrpc": "2.0", "result": 16}
00099      *
00100      * \endenglish
00101      */
00102  int getStandardDigitalInputNum();
00103
00104  /**
00105   * @ingroup IoControl
00106   * \chinese
00107   * 获取工具端数字 IO 数量（包括数字输入和数字输出）
00108   *
00109   * @return 工具端数字 IO 数量（包括数字输入和数字输出）
00110   *
00111   * @throws arcs::common_interface::AuboException
00112   *
00113   * @par Python 函数原型
00114   * getToolDigitalInputNum(self: pyaubo_sdk.IoControl) -> int
00115   *
00116   * @par Lua 函数原型
00117   * getToolDigitalInputNum() -> number
00118   *
00119   * @par Lua 示例
00120   * num = getStandardDigitalInputNum()
00121   *
00122   * @par JSON-RPC 请求示例
00123   * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolDigitalInputNum", "params": [], "id": 1}
00124   *
00125   * @par JSON-RPC 响应示例
00126   * {"id": 1, "jsonrpc": "2.0", "result": 4}
00127   *
00128   * \endchinese
00129   * \english
00130   * Get the number of tool digital IOs (including digital inputs and
00131   * outputs).
00132   *
00133   * @return Number of tool digital IOs (including digital inputs and
00134   * outputs).
00135   *
00136   * @throws arcs::common_interface::AuboException
00137   *
00138   * @par Python function prototype
00139   * getToolDigitalInputNum(self: pyaubo_sdk.IoControl) -> int
00140   *
00141   * @par Lua function prototype
00142   * getToolDigitalInputNum() -> number
00143   *
00144   * @par Lua example
00145   * num = getStandardDigitalInputNum()
00146   *
00147   * @par JSON-RPC request example
00148   * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolDigitalInputNum", "params": [], "id": 1}
00149   *
00150   * @par JSON-RPC response example
00151   * {"id": 1, "jsonrpc": "2.0", "result": 4}
00152   *
00153   * \endenglish
00154   */
00155  int getToolDigitalInputNum();
00156
00157  /**
00158   * @ingroup IoControl
00159   * \chinese
00160   * 获取可配置数字输入数量
00161   *
00162   * @return 可配置数字输入数量
00163   *
00164   * @throws arcs::common_interface::AuboException
00165   *
00166   * @par Python 函数原型
00167   * getConfigurableDigitalInputNum(self: pyaubo_sdk.IoControl) -> int
00168   *
00169   * @par Lua 函数原型
00170   * getConfigurableDigitalInputNum() -> number

```

```

00171      *
00172      * @par Lua 示例
00173      * num = getConfigurableDigitalInputNum()
00174      *
00175      * @par JSON-RPC 请求示例
00176      * {"jsonrpc": "2.0", "method": "rob1.IoControl.getConfigurableDigitalInputNum", "params": [], "id": 1}
00177      *
00178      * @par JSON-RPC 响应示例
00179      * {"id": 1, "jsonrpc": "2.0", "result": 16}
00180      *
00181      * \endchinese
00182      * \english
00183      * Get the number of configurable digital inputs.
00184      *
00185      * @return Number of configurable digital inputs.
00186      *
00187      * @throws arcs::common_interface::AuboException
00188      *
00189      * @par Python function prototype
00190      * getConfigurableDigitalInputNum(self: pyaubo_sdk.IoControl) -> int
00191      *
00192      * @par Lua function prototype
00193      * getConfigurableDigitalInputNum() -> number
00194      *
00195      * @par Lua example
00196      * num = getConfigurableDigitalInputNum()
00197      *
00198      * @par JSON-RPC request example
00199      * {"jsonrpc": "2.0", "method": "rob1.IoControl.getConfigurableDigitalInputNum", "params": [], "id": 1}
00200      *
00201      * @par JSON-RPC response example
00202      * {"id": 1, "jsonrpc": "2.0", "result": 16}
00203      *
00204      * \endenglish
00205      */
00206 int getConfigurableDigitalInputNum();
00207
00208 /**
00209  * @ingroup IoControl
00210  * \chinese
00211  * 获取标准数字输出数量
00212  *
00213  * @return 标准数字输出数量
00214  *
00215  * @throws arcs::common_interface::AuboException
00216  *
00217  * @par Python 函数原型
00218  * getStandardDigitalOutputNum(self: pyaubo_sdk.IoControl) -> int
00219  *
00220  * @par Lua 函数原型
00221  * getStandardDigitalOutputNum() -> number
00222  *
00223  * @par Lua 示例
00224  * num = getStandardDigitalOutputNum()
00225  *
00226  * @par JSON-RPC 请求示例
00227  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardDigitalOutputNum", "params": [], "id": 1}
00228  *
00229  * @par JSON-RPC 响应示例
00230  * {"id": 1, "jsonrpc": "2.0", "result": 8}
00231  *
00232  * \endchinese
00233  * \english
00234  * Get the number of standard digital outputs.
00235  *
00236  * @return Number of standard digital outputs.
00237  *
00238  * @throws arcs::common_interface::AuboException
00239  *
00240  * @par Python function prototype
00241  * getStandardDigitalOutputNum(self: pyaubo_sdk.IoControl) -> int
00242  *
00243  * @par Lua function prototype
00244  * getStandardDigitalOutputNum() -> number
00245  *
00246  * @par Lua example
00247  * num = getStandardDigitalOutputNum()
00248  *
00249  * @par JSON-RPC request example
00250  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardDigitalOutputNum", "params": [], "id": 1}
00251  *
00252  * @par JSON-RPC response example
00253  * {"id": 1, "jsonrpc": "2.0", "result": 8}
00254  *
00255  * \endenglish
00256  */
00257 int getStandardDigitalOutputNum();

```

```

00258
00259 /**
00260  * @ingroup IoControl
00261  * \chinese
00262  * 获取工具端数字 IO 数量 (包括数字输入和数字输出)
00263  *
00264  * @return 工具端数字 IO 数量 (包括数字输入和数字输出)
00265  *
00266  * @throws arcs::common_interface::AuboException
00267  *
00268  * @par Python 函数原型
00269  * getToolDigitalOutputNum(self: pyaubo_sdk.IoControl) -> int
00270  *
00271  * @par Lua 函数原型
00272  * getToolDigitalOutputNum() -> number
00273  *
00274  * @par Lua 示例
00275  * num = getToolDigitalOutputNum()
00276  *
00277  * @par JSON-RPC 请求示例
00278  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolDigitalOutputNum", "params": [], "id": 1}
00279  *
00280  * @par JSON-RPC 响应示例
00281  * {"id": 1, "jsonrpc": "2.0", "result": 4}
00282  *
00283  * \endchinese
00284  * \english
00285  * Get the number of tool digital IOs (including digital inputs and
00286  * outputs).
00287  *
00288  * @return Number of tool digital IOs (including digital inputs and
00289  * outputs).
00290  *
00291  * @throws arcs::common_interface::AuboException
00292  *
00293  * @par Python function prototype
00294  * getToolDigitalOutputNum(self: pyaubo_sdk.IoControl) -> int
00295  *
00296  * @par Lua function prototype
00297  * getToolDigitalOutputNum() -> number
00298  *
00299  * @par Lua example
00300  * num = getToolDigitalOutputNum()
00301  *
00302  * @par JSON-RPC request example
00303  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolDigitalOutputNum", "params": [], "id": 1}
00304  *
00305  * @par JSON-RPC response example
00306  * {"id": 1, "jsonrpc": "2.0", "result": 4}
00307  *
00308  * \endenglish
00309  */
00310 int getToolDigitalOutputNum();
00311
00312 /**
00313  * @ingroup IoControl
00314  * \chinese
00315  * 设置指定的工具端数字 IO 为输入或输出
00316  *
00317  * 工具端数字 IO 比较特殊, IO 可以配置为输入或者输出
00318  *
00319  * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
00320  * 例如, 0 表示第一个管脚。
00321  * @param input: 表示指定 IO 是否为输入。
00322  * input 为 true 时, 设置指定 IO 为输入, 否则为输出
00323  * @return 成功返回 0; 失败返回错误码
00324  * AUBO_BUSY
00325  * AUBO_BAD_STATE
00326  * -AUBO_INVL_ARGUMENT
00327  * -AUBO_BAD_STATE
00328  *
00329  * @throws arcs::common_interface::AuboException
00330  *
00331  * @par Python 函数原型
00332  * setToolIoInput(self: pyaubo_sdk.IoControl, arg0: int, arg1: bool) -> int
00333  *
00334  * @par Lua 函数原型
00335  * setToolIoInput(index: number, input: boolean) -> nil
00336  *
00337  * @par Lua 示例
00338  * setToolIoInput(0, true)
00339  *
00340  * @par JSON-RPC 请求示例
00341  * {"jsonrpc": "2.0", "method": "rob1.IoControl.setToolIoInput", "params": [0, true], "id": 1}
00342  *
00343  * @par JSON-RPC 响应示例
00344  * {"id": 1, "jsonrpc": "2.0", "result": 0}

```

```

00345      *
00346      * \endchinese
00347      * \english
00348      * Set the specified tool digital IO as input or output.
00349      *
00350      * Tool digital IOs are special and can be configured as input or output.
00351      *
00352      * @param index: Indicates the IO pin, starting from 0.
00353      * For example, 0 means the first pin.
00354      * @param input: Indicates whether the specified IO is input.
00355      * If input is true, set the IO as input; otherwise, set as output.
00356      * @return Returns 0 on success; error code on failure.
00357      * AUBO_BUSY
00358      * AUBO_BAD_STATE
00359      * -AUBO_INVL_ARGUMENT
00360      * -AUBO_BAD_STATE
00361      *
00362      * @throws arcs::common_interface::AuboException
00363      *
00364      * @par Python function prototype
00365      * setToolIoInput(self: pyaubo_sdk.IoControl, arg0: int, arg1: bool) -> int
00366      *
00367      * @par Lua function prototype
00368      * setToolIoInput(index: number, input: boolean) -> nil
00369      *
00370      * @par Lua example
00371      * setToolIoInput(0,true)
00372      *
00373      * @par JSON-RPC request example
00374      * {"jsonrpc":"2.0","method":"rob1.IoControl.setToolIoInput","params":[0,true],"id":1}
00375      *
00376      * @par JSON-RPC response example
00377      * {"id":1,"jsonrpc":"2.0","result":0}
00378      *
00379      * \endenglish
00380      */
00381 int setToolIoInput(int index, bool input);
00382
00383 /**
00384  * @ingroup IoControl
00385  * \chinese
00386  * 判断指定的工具端数字 IO 类型是否为输入
00387  *
00388  * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
00389  * 例如, 0 表示第一个管脚。
00390  * @return 当指定的 IO 为输入时返回 true, 否则为 false
00391  *
00392  * @throws arcs::common_interface::AuboException
00393  *
00394  * @par Python 函数原型
00395  * isToolIoInput(self: pyaubo_sdk.IoControl, arg0: int) -> bool
00396  *
00397  * @par Lua 函数原型
00398  * isToolIoInput(index: number) -> boolean
00399  *
00400  * @par Lua 示例
00401  * status = isToolIoInput(0)
00402  *
00403  * @par JSON-RPC 请求示例
00404  * {"jsonrpc":"2.0","method":"rob1.IoControl.isToolIoInput","params":[0],"id":1}
00405  *
00406  * @par JSON-RPC 响应示例
00407  * {"id":1,"jsonrpc":"2.0","result":true}
00408  *
00409  * \endchinese
00410  * \english
00411  * Determine whether the specified tool digital IO is configured as input.
00412  *
00413  * @param index: Indicates the IO pin, starting from 0.
00414  * For example, 0 means the first pin.
00415  * @return Returns true if the specified IO is input, otherwise false.
00416  *
00417  * @throws arcs::common_interface::AuboException
00418  *
00419  * @par Python function prototype
00420  * isToolIoInput(self: pyaubo_sdk.IoControl, arg0: int) -> bool
00421  *
00422  * @par Lua function prototype
00423  * isToolIoInput(index: number) -> boolean
00424  *
00425  * @par Lua example
00426  * status = isToolIoInput(0)
00427  *
00428  * @par JSON-RPC request example
00429  * {"jsonrpc":"2.0","method":"rob1.IoControl.isToolIoInput","params":[0],"id":1}
00430  *
00431  * @par JSON-RPC response example

```

```

00432     * {"id":1,"jsonrpc":"2.0","result":true}
00433     *
00434     * \endenglish
00435     */
00436 bool isToolIoInput(int index);
00437
00438 /**
00439  * @ingroup IoControl
00440  * \chinese
00441  * 获取可配置数字输出数量
00442  *
00443  * @return 可配置数字输出数量
00444  *
00445  * @throws arcs::common_interface::AuboException
00446  *
00447  * @par Python 函数原型
00448  * getConfigurableDigitalOutputNum(self: pyaubo_sdk.IoControl) -> int
00449  *
00450  * @par Lua 函数原型
00451  * getConfigurableDigitalOutputNum() -> number
00452  *
00453  * @par Lua 示例
00454  * num = getConfigurableDigitalOutputNum()
00455  *
00456  * @par JSON-RPC 请求示例
00457  * {"jsonrpc":"2.0","method":"rob1.IoControl.getConfigurableDigitalOutputNum","params":[],"id":1}
00458  *
00459  * @par JSON-RPC 响应示例
00460  * {"id":1,"jsonrpc":"2.0","result":16}
00461  *
00462  * \endchinese
00463  * \english
00464  * Get the number of configurable digital outputs.
00465  *
00466  * @return Number of configurable digital outputs.
00467  *
00468  * @throws arcs::common_interface::AuboException
00469  *
00470  * @par Python function prototype
00471  * getConfigurableDigitalOutputNum(self: pyaubo_sdk.IoControl) -> int
00472  *
00473  * @par Lua function prototype
00474  * getConfigurableDigitalOutputNum() -> number
00475  *
00476  * @par Lua example
00477  * num = getConfigurableDigitalOutputNum()
00478  *
00479  * @par JSON-RPC request example
00480  * {"jsonrpc":"2.0","method":"rob1.IoControl.getConfigurableDigitalOutputNum","params":[],"id":1}
00481  *
00482  * @par JSON-RPC response example
00483  * {"id":1,"jsonrpc":"2.0","result":16}
00484  *
00485  * \endenglish
00486  */
00487 int getConfigurableDigitalOutputNum();
00488
00489 /**
00490  * @ingroup IoControl
00491  * \chinese
00492  * 获取标准模拟输入数量
00493  *
00494  * @return 标准模拟输入数量
00495  *
00496  * @throws arcs::common_interface::AuboException
00497  *
00498  * @par Python 函数原型
00499  * getStandardAnalogInputNum(self: pyaubo_sdk.IoControl) -> int
00500  *
00501  * @par Lua 函数原型
00502  * getStandardAnalogInputNum() -> number
00503  *
00504  * @par Lua 示例
00505  * num = getStandardAnalogInputNum()
00506  *
00507  * @par JSON-RPC 请求示例
00508  * {"jsonrpc":"2.0","method":"rob1.IoControl.getStandardAnalogInputNum","params":[],"id":1}
00509  *
00510  * @par JSON-RPC 响应示例
00511  * {"id":1,"jsonrpc":"2.0","result":2}
00512  *
00513  * \endchinese
00514  * \english
00515  * Get the number of standard analog inputs.
00516  *
00517  * @return Number of standard analog inputs.
00518  *

```



```

00519     * @throws arcs::common_interface::AuboException
00520     *
00521     * @par Python function prototype
00522     * getStandardAnalogInputNum(self: pyaubo_sdk.IoControl) -> int
00523     *
00524     * @par Lua function prototype
00525     * getStandardAnalogInputNum() -> number
00526     *
00527     * @par Lua example
00528     * num = getStandardAnalogInputNum()
00529     *
00530     * @par JSON-RPC request example
00531     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardAnalogInputNum", "params": [], "id": 1}
00532     *
00533     * @par JSON-RPC response example
00534     * {"id": 1, "jsonrpc": "2.0", "result": 2}
00535     *
00536     * \endenglish
00537     */
00538 int getStandardAnalogInputNum();
00539
00540 /**
00541  * @ingroup IoControl
00542  * \chinese
00543  * 获取工具端模拟输入数量
00544  *
00545  * @return 工具端模拟输入数量
00546  *
00547  * @throws arcs::common_interface::AuboException
00548  *
00549  * @par Python 函数原型
00550  * getToolAnalogInputNum(self: pyaubo_sdk.IoControl) -> int
00551  *
00552  * @par Lua 函数原型
00553  * getToolAnalogInputNum() -> number
00554  *
00555  * @par Lua 示例
00556  * num = getToolAnalogInputNum()
00557  *
00558  * @par JSON-RPC 请求示例
00559  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolAnalogInputNum", "params": [], "id": 1}
00560  *
00561  * @par JSON-RPC 响应示例
00562  * {"id": 1, "jsonrpc": "2.0", "result": 2}
00563  *
00564  * \endchinese
00565  * \english
00566  * Get the number of tool analog inputs.
00567  *
00568  * @return Number of tool analog inputs.
00569  *
00570  * @throws arcs::common_interface::AuboException
00571  *
00572  * @par Python function prototype
00573  * getToolAnalogInputNum(self: pyaubo_sdk.IoControl) -> int
00574  *
00575  * @par Lua function prototype
00576  * getToolAnalogInputNum() -> number
00577  *
00578  * @par Lua example
00579  * num = getToolAnalogInputNum()
00580  *
00581  * @par JSON-RPC request example
00582  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolAnalogInputNum", "params": [], "id": 1}
00583  *
00584  * @par JSON-RPC response example
00585  * {"id": 1, "jsonrpc": "2.0", "result": 2}
00586  *
00587  * \endenglish
00588  */
00589 int getToolAnalogInputNum();
00590
00591 /**
00592  * @ingroup IoControl
00593  * \chinese
00594  * 获取标准模拟输出数量
00595  *
00596  * @return 标准模拟输出数量
00597  *
00598  * @throws arcs::common_interface::AuboException
00599  *
00600  * @par Python 函数原型
00601  * getStandardAnalogOutputNum(self: pyaubo_sdk.IoControl) -> int
00602  *
00603  * @par Lua 函数原型
00604  * getStandardAnalogOutputNum() -> number
00605  *

```

```

00606     * @par Lua 示例
00607     * num = getStandardAnalogOutputNum()
00608     *
00609     * @par JSON-RPC 请求示例
00610     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardAnalogOutputNum", "params": [], "id": 1}
00611     *
00612     * @par JSON-RPC 响应示例
00613     * {"id": 1, "jsonrpc": "2.0", "result": 2}
00614     *
00615     * \endchinese
00616     * \english
00617     * Get the number of standard analog outputs.
00618     *
00619     * @return Number of standard analog outputs.
00620     *
00621     * @throws arcs::common_interface::AuboException
00622     *
00623     * @par Python function prototype
00624     * getStandardAnalogOutputNum(self: pyaubo_sdk.IoControl) -> int
00625     *
00626     * @par Lua function prototype
00627     * getStandardAnalogOutputNum() -> number
00628     *
00629     * @par Lua example
00630     * num = getStandardAnalogOutputNum()
00631     *
00632     * @par JSON-RPC request example
00633     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardAnalogOutputNum", "params": [], "id": 1}
00634     *
00635     * @par JSON-RPC response example
00636     * {"id": 1, "jsonrpc": "2.0", "result": 2}
00637     *
00638     * \endenglish
00639     */
00640 int getStandardAnalogOutputNum();
00641
00642 /**
00643     * @ingroup IoControl
00644     * \chinese
00645     * 获取工具端模拟输出数量
00646     *
00647     * @return 工具端模拟输出数量
00648     *
00649     * @throws arcs::common_interface::AuboException
00650     *
00651     * @par Python 函数原型
00652     * getToolAnalogOutputNum(self: pyaubo_sdk.IoControl) -> int
00653     *
00654     * @par Lua 函数原型
00655     * getToolAnalogOutputNum() -> number
00656     *
00657     * @par Lua 示例
00658     * num = getToolAnalogOutputNum()
00659     *
00660     * @par JSON-RPC 请求示例
00661     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolAnalogOutputNum", "params": [], "id": 1}
00662     *
00663     * @par JSON-RPC 响应示例
00664     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00665     *
00666     * \endchinese
00667     * \english
00668     * Get the number of tool analog outputs.
00669     *
00670     * @return Number of tool analog outputs.
00671     *
00672     * @throws arcs::common_interface::AuboException
00673     *
00674     * @par Python function prototype
00675     * getToolAnalogOutputNum(self: pyaubo_sdk.IoControl) -> int
00676     *
00677     * @par Lua function prototype
00678     * getToolAnalogOutputNum() -> number
00679     *
00680     * @par Lua example
00681     * num = getToolAnalogOutputNum()
00682     *
00683     * @par JSON-RPC request example
00684     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolAnalogOutputNum", "params": [], "id": 1}
00685     *
00686     * @par JSON-RPC response example
00687     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00688     *
00689     * \endenglish
00690     */
00691 int getToolAnalogOutputNum();
00692

```

```

00693  /**
00694   * @ingroup IoControl
00695   * \chinese
00696   * 设置所有数字输入动作为无触发
00697   *
00698   * @note
00699   * 当输入动作为无触发时，用户设置数字输入值为高电平，不会触发机器人发生动作
00700   *
00701   * @return 成功返回 0；失败返回错误码
00702   * AUBO_BUSY
00703   * AUBO_BAD_STATE
00704   * -AUBO_BAD_STATE
00705   *
00706   * @throws arcs::common_interface::AuboException
00707   *
00708   * @par Python 函数原型
00709   * setDigitalInputActionDefault(self: pyaubo_sdk.IoControl) -> int
00710   *
00711   * @par Lua 函数原型
00712   * setDigitalInputActionDefault() -> nil
00713   *
00714   * @par Lua 示例
00715   * setDigitalInputActionDefault()
00716   *
00717   * @par JSON-RPC 请求示例
00718   * {"jsonrpc":"2.0","method":"robl.IoControl.setDigitalInputActionDefault","params":[],"id":1}
00719   *
00720   * @par JSON-RPC 响应示例
00721   * {"id":1,"jsonrpc":"2.0","result":0}
00722   *
00723   * \endchinese
00724   * \english
00725   * Set all digital input actions to no trigger.
00726   *
00727   * @note
00728   * When the input action is set to no trigger, setting the digital input
00729   * value to high will not trigger any robot action.
00730   *
00731   * @return Returns 0 on success; error code on failure.
00732   * AUBO_BUSY
00733   * AUBO_BAD_STATE
00734   * -AUBO_BAD_STATE
00735   *
00736   * @throws arcs::common_interface::AuboException
00737   *
00738   * @par Python function prototype
00739   * setDigitalInputActionDefault(self: pyaubo_sdk.IoControl) -> int
00740   *
00741   * @par Lua function prototype
00742   * setDigitalInputActionDefault() -> nil
00743   *
00744   * @par Lua example
00745   * setDigitalInputActionDefault()
00746   *
00747   * @par JSON-RPC request example
00748   * {"jsonrpc":"2.0","method":"robl.IoControl.setDigitalInputActionDefault","params":[],"id":1}
00749   *
00750   * @par JSON-RPC response example
00751   * {"id":1,"jsonrpc":"2.0","result":0}
00752   *
00753   * \endenglish
00754   */
00755  int setDigitalInputActionDefault();
00756
00757  /**
00758   * @ingroup IoControl
00759   * \chinese
00760   * 设置标准数字输入触发动作
00761   *
00762   * @note
00763   * 当给输入设置为无触发动作（StandardInputAction::Default）时，
00764   * 用户设置数字输入值为高电平，不会触发机器人发生动作。\\n
00765   * 当给输入设置触发动作时，用户设置数字输入值为高电平，会触发机器人执行相应的动作。\\n
00766   * 例如，当设置 DIO 的触发动作作为拖动示教（StandardInputAction::Handguide）时，
00767   * 用户设置 DIO 为高电平，机器人会进入拖动示教。
00768   * 设置 DIO 为低电平，机器人会退出拖动示教。
00769   *
00770   * @param index: 表示 IO 口的管脚，管脚编号从 0 开始。
00771   * 例如，0 表示第一个管脚。
00772   * @param action: 触发动作
00773   *
00774   * @return 成功返回 0；失败返回错误码
00775   * AUBO_REQUEST_IGNORE
00776   * AUBO_BUSY
00777   * AUBO_BAD_STATE
00778   * -AUBO_INVL_ARGUMENT
00779   * -AUBO_BAD_STATE

```

```

00780      *
00781      * @throws arcs::common_interface::AuboException
00782      *
00783      * @par Python 函数原型
00784      * setStandardDigitalInputAction(self: pyaubo_sdk.IoControl, arg0: int,
00785      * arg1: arcs::common_interface::StandardInputAction) -> int
00786      *
00787      * @par Lua 函数原型
00788      * setStandardDigitalInputAction(index: number, action: number) -> nil
00789      *
00790      * @par Lua 示例
00791      * setStandardDigitalInputAction(0,1)
00792      *
00793      * @par JSON-RPC 请求示例
00794      * {"jsonrpc": "2.0", "method": "rob1.IoControl.setStandardDigitalInputAction", "params": [0, "Handguide"], "id": 1}
00795      *
00796      * @par JSON-RPC 响应示例
00797      * {"id": 1, "jsonrpc": "2.0", "result": 0}
00798      *
00799      * \endchinese
00800      * \english
00801      * Set the trigger action for standard digital input.
00802      *
00803      * @note
00804      * When the input is set to no trigger action
00805      * (StandardInputAction::Default), setting the digital input value to high
00806      * will not trigger any robot action.\n When a trigger action is set,
00807      * setting the digital input value to high will trigger the corresponding
00808      * robot action.\n For example, if DIO is set to trigger Handguide
00809      * (StandardInputAction::Handguide), setting DIO to high will enable
00810      * hand-guiding mode. Setting DIO to low will exit hand-guiding mode.
00811      *
00812      * @param index: Indicates the IO pin, starting from 0.
00813      * For example, 0 means the first pin.
00814      * @param action: Trigger action
00815      *
00816      * @return Returns 0 on success; error code on failure.
00817      * AUBO_REQUEST_IGNORE
00818      * AUBO_BUSY
00819      * AUBO_BAD_STATE
00820      * -AUBO_INVL_ARGUMENT
00821      * -AUBO_BAD_STATE
00822      *
00823      * @throws arcs::common_interface::AuboException
00824      *
00825      * @par Python function prototype
00826      * setStandardDigitalInputAction(self: pyaubo_sdk.IoControl, arg0: int,
00827      * arg1: arcs::common_interface::StandardInputAction) -> int
00828      *
00829      * @par Lua function prototype
00830      * setStandardDigitalInputAction(index: number, action: number) -> nil
00831      *
00832      * @par Lua example
00833      * setStandardDigitalInputAction(0,1)
00834      *
00835      * @par JSON-RPC request example
00836      * {"jsonrpc": "2.0", "method": "rob1.IoControl.setStandardDigitalInputAction", "params": [0, "Handguide"], "id": 1}
00837      *
00838      * @par JSON-RPC response example
00839      * {"id": 1, "jsonrpc": "2.0", "result": 0}
00840      *
00841      * \endenglish
00842      */
00843      int setStandardDigitalInputAction(int index, StandardInputAction action);
00844
00845      /**
00846      * @ingroup IoControl
00847      * \chinese
00848      * 设置工具数字输入触发动作
00849      *
00850      * @note
00851      * 当给输入设置为无触发动作 (StandardInputAction::Default) 时,
00852      * 用户设置工具数字输入值为高电平, 不会触发机器人发生动作。 \n
00853      * 当给输入设置触发动作时, 用户设置工具数字输入值为高电平, 会触发机器人执行相应的动作。 \n
00854      * 例如, 当设置 TOOL_IO[0] 的类型为输入而且触发动作为拖动示教 (StandardInputAction::Handguide) 时,
00855      * 用户设置 TOOL_IO[0] 为高电平, 机器人会进入拖动示教。
00856      * 设置 TOOL_IO[0] 为低电平, 机器人会退出拖动示教。
00857      *
00858      * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
00859      * 例如, 0 表示第一个管脚。
00860      * @param action: 触发动作
00861      *
00862      * @return 成功返回 0; 失败返回错误码
00863      * AUBO_REQUEST_IGNORE
00864      * AUBO_BUSY
00865      * AUBO_BAD_STATE
00866      * -AUBO_INVL_ARGUMENT

```

```

00867     * -AUBO_BAD_STATE
00868     *
00869     * @throws arcs::common_interface::AuboException
00870     *
00871     * @par Python 函数原型
00872     * setToolDigitalInputAction(self: pyaubo_sdk.IoControl, arg0: int, arg1:
00873     * arcs::common_interface::StandardInputAction) -> int
00874     *
00875     * @par Lua 函数原型
00876     * setToolDigitalInputAction(index: number, action: number) -> nil
00877     *
00878     * @par Lua 示例
00879     * setToolDigitalInputAction(0,1)
00880     *
00881     * @par JSON-RPC 请求示例
00882     * {"jsonrpc":"2.0","method":"robl.IoControl.setToolDigitalInputAction","params":[0,"Handguide"],"id":1}
00883     *
00884     * @par JSON-RPC 响应示例
00885     * {"id":1,"jsonrpc":"2.0","result":0}
00886     *
00887     * \endchinese
00888     * \english
00889     * Set the trigger action for tool digital input.
00890     *
00891     * @note
00892     * When the input is set to no trigger action
00893     * (StandardInputAction::Default), setting the tool digital input value to
00894     * high will not trigger any robot action.\n When a trigger action is set,
00895     * setting the tool digital input value to high will trigger the
00896     * corresponding robot action.\n For example, if TOOL_IO[0] is set as input
00897     * and its trigger action is Handguide (StandardInputAction::Handguide),
00898     * setting TOOL_IO[0] to high will enable hand-guiding mode.
00899     * Setting TOOL_IO[0] to low will exit hand-guiding mode.
00900     *
00901     * @param index: Indicates the IO pin, starting from 0.
00902     * For example, 0 means the first pin.
00903     * @param action: Trigger action
00904     *
00905     * @return Returns 0 on success; error code on failure.
00906     * AUBO_REQUEST_IGNORE
00907     * AUBO_BUSY
00908     * AUBO_BAD_STATE
00909     * -AUBO_INVL_ARGUMENT
00910     * -AUBO_BAD_STATE
00911     *
00912     * @throws arcs::common_interface::AuboException
00913     *
00914     * @par Python function prototype
00915     * setToolDigitalInputAction(self: pyaubo_sdk.IoControl, arg0: int, arg1:
00916     * arcs::common_interface::StandardInputAction) -> int
00917     *
00918     * @par Lua function prototype
00919     * setToolDigitalInputAction(index: number, action: number) -> nil
00920     *
00921     * @par Lua example
00922     * setToolDigitalInputAction(0,1)
00923     *
00924     * @par JSON-RPC request example
00925     * {"jsonrpc":"2.0","method":"robl.IoControl.setToolDigitalInputAction","params":[0,"Handguide"],"id":1}
00926     *
00927     * @par JSON-RPC response example
00928     * {"id":1,"jsonrpc":"2.0","result":0}
00929     *
00930     * \endenglish
00931     */
00932 int setToolDigitalInputAction(int index, StandardInputAction action);
00933
00934 /**
00935  * @ingroup IoControl
00936  * \chinese
00937  * 设置可配置数字输入触发动作
00938  *
00939  * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
00940  * 例如, 0 表示第一个管脚。
00941  * @param action: 触发动作
00942  *
00943  * @return 成功返回 0; 失败返回错误码
00944  * AUBO_REQUEST_IGNORE
00945  * AUBO_BUSY
00946  * AUBO_BAD_STATE
00947  * -AUBO_INVL_ARGUMENT
00948  * -AUBO_BAD_STATE
00949  *
00950  * @throws arcs::common_interface::AuboException
00951  *
00952  * @note 需要将可配置输入的安全输入动作设置为
00953  * SafetyInputAction::Unassigned 时这个函数的配置才会生效

```

```

00954 *
00955 * @par Python 函数原型
00956 * setConfigurableDigitalInputAction(self: pyaubo_sdk.IoControl, arg0: int,
00957 * arg1: arcs::common_interface::StandardInputAction) -> int
00958 *
00959 * @par Lua 函数原型
00960 * setConfigurableDigitalInputAction(index: number, action: number) -> nil
00961 *
00962 * @par Lua 示例
00963 * setConfigurableDigitalInputAction(0,1)
00964 *
00965 * @par JSON-RPC 请求示例
00966 * {"jsonrpc": "2.0", "method": "rob1.IoControl.setConfigurableDigitalInputAction", "params": [0, "Handguide"], "id": 1}
00967 *
00968 * @par JSON-RPC 响应示例
00969 * {"id": 1, "jsonrpc": "2.0", "result": 0}
00970 *
00971 * \endchinese
00972 * \english
00973 * Set the trigger action for configurable digital input.
00974 *
00975 * @param index: Indicates the IO pin, starting from 0.
00976 * For example, 0 means the first pin.
00977 * @param action: Trigger action
00978 *
00979 * @return Returns 0 on success; error code on failure.
00980 * AUBO_REQUEST_IGNORE
00981 * AUBO_BUSY
00982 * AUBO_BAD_STATE
00983 * -AUBO_INVL_ARGUMENT
00984 * -AUBO_BAD_STATE
00985 *
00986 * @throws arcs::common_interface::AuboException
00987 *
00988 * @note This function takes effect only when the safety input action of the
00989 * configurable input is set to SafetyInputAction::Unassigned.
00990 *
00991 * @par Python function prototype
00992 * setConfigurableDigitalInputAction(self: pyaubo_sdk.IoControl, arg0: int,
00993 * arg1: arcs::common_interface::StandardInputAction) -> int
00994 *
00995 * @par Lua function prototype
00996 * setConfigurableDigitalInputAction(index: number, action: number) -> nil
00997 *
00998 * @par Lua example
00999 * setConfigurableDigitalInputAction(0,1)
01000 *
01001 * @par JSON-RPC request example
01002 * {"jsonrpc": "2.0", "method": "rob1.IoControl.setConfigurableDigitalInputAction", "params": [0, "Handguide"], "id": 1}
01003 *
01004 * @par JSON-RPC response example
01005 * {"id": 1, "jsonrpc": "2.0", "result": 0}
01006 *
01007 * \endenglish
01008 */
01009 int setConfigurableDigitalInputAction(int index,
01010                                     StandardInputAction action);
01011
01012 /**
01013 * @ingroup IoControl
01014 * \chinese
01015 * 获取标准数字输入触发动作
01016 *
01017 * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
01018 * 例如, 0 表示第一个管脚。
01019 * @return 标准数字输入触发动作
01020 *
01021 * @par Python 函数原型
01022 * getStandardDigitalInputAction(self: pyaubo_sdk.IoControl, arg0: int) ->
01023 * arcs::common_interface::StandardInputAction
01024 *
01025 * @par Lua 函数原型
01026 * getStandardDigitalInputAction(index: number) -> number
01027 *
01028 * @par Lua 示例
01029 * num = getStandardDigitalInputAction(0)
01030 *
01031 * @par JSON-RPC 请求示例
01032 * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardDigitalInputAction", "params": [0], "id": 1}
01033 *
01034 * @par JSON-RPC 响应示例
01035 * {"id": 1, "jsonrpc": "2.0", "result": "Default"}
01036 *
01037 * \endchinese
01038 * \english
01039 * Get the trigger action for standard digital input.
01040 *

```

```

01041     * @param index: Indicates the IO pin, starting from 0.
01042     * For example, 0 means the first pin.
01043     * @return Standard digital input trigger action
01044     *
01045     * @par Python function prototype
01046     * getStandardDigitalInputAction(self: pyaubo_sdk.IoControl, arg0: int) ->
01047     * arcs::common_interface::StandardInputAction
01048     *
01049     * @par Lua function prototype
01050     * getStandardDigitalInputAction(index: number) -> number
01051     *
01052     * @par Lua example
01053     * num = getStandardDigitalInputAction(0)
01054     *
01055     * @par JSON-RPC request example
01056     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardDigitalInputAction", "params": [0], "id": 1}
01057     *
01058     * @par JSON-RPC response example
01059     * {"id": 1, "jsonrpc": "2.0", "result": "Default"}
01060     *
01061     * \endenglish
01062     */
01063     StandardInputAction getStandardDigitalInputAction(int index);
01064
01065     /**
01066     * @ingroup IoControl
01067     * \chinese
01068     * 获取工具端数字输入触发动作
01069     *
01070     * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
01071     * 例如, 0 表示第一个管脚。
01072     * @return 工具端数字输入触发动作
01073     *
01074     * @par Python 函数原型
01075     * getToolDigitalInputAction(self: pyaubo_sdk.IoControl, arg0: int) ->
01076     * arcs::common_interface::StandardInputAction
01077     *
01078     * @par Lua 函数原型
01079     * getToolDigitalInputAction(index: number) -> number
01080     *
01081     * @par Lua 示例
01082     * getToolDigitalInputAction(0)
01083     *
01084     * @par JSON-RPC 请求示例
01085     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolDigitalInputAction", "params": [0], "id": 1}
01086     *
01087     * @par JSON-RPC 响应示例
01088     * {"id": 1, "jsonrpc": "2.0", "result": "Default"}
01089     *
01090     * \endchinese
01091     * \english
01092     * Get the trigger action for tool digital input.
01093     *
01094     * @param index: Indicates the IO pin, starting from 0.
01095     * For example, 0 means the first pin.
01096     * @return Tool digital input trigger action
01097     *
01098     * @par Python function prototype
01099     * getToolDigitalInputAction(self: pyaubo_sdk.IoControl, arg0: int) ->
01100     * arcs::common_interface::StandardInputAction
01101     *
01102     * @par Lua function prototype
01103     * getToolDigitalInputAction(index: number) -> number
01104     *
01105     * @par Lua example
01106     * getToolDigitalInputAction(0)
01107     *
01108     * @par JSON-RPC request example
01109     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolDigitalInputAction", "params": [0], "id": 1}
01110     *
01111     * @par JSON-RPC response example
01112     * {"id": 1, "jsonrpc": "2.0", "result": "Default"}
01113     *
01114     * \endenglish
01115     */
01116     StandardInputAction getToolDigitalInputAction(int index);
01117
01118     /**
01119     * @ingroup IoControl
01120     * \chinese
01121     * 获取可配置数字输入的输入触发动作
01122     *
01123     * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
01124     * 例如, 0 表示第一个管脚。
01125     * @return 返回输入触发动作
01126     *
01127     * @par Python 函数原型

```



```

01128     * getConfigurableDigitalInputAction(self: pyaubo_sdk.IoControl, arg0: int)
01129     * -> arcs::common_interface::StandardInputAction
01130     *
01131     * @par Lua 函数原型
01132     * getConfigurableDigitalInputAction(index: number) -> number
01133     *
01134     * @par Lua 示例
01135     * num = getConfigurableDigitalInputAction(0)
01136     *
01137     * @par JSON-RPC 请求示例
01138     * {"jsonrpc": "2.0", "method": "robl.IoControl.getConfigurableDigitalInputAction", "params": [0], "id": 1}
01139     *
01140     * @par JSON-RPC 响应示例
01141     * {"id": 1, "jsonrpc": "2.0", "result": "Default"}
01142     *
01143     * \endchinese
01144     * \english
01145     * Get the trigger action for configurable digital input.
01146     *
01147     * @param index: Indicates the IO pin, starting from 0.
01148     * For example, 0 means the first pin.
01149     * @return Returns the input trigger action.
01150     *
01151     * @par Python function prototype
01152     * getConfigurableDigitalInputAction(self: pyaubo_sdk.IoControl, arg0: int)
01153     * -> arcs::common_interface::StandardInputAction
01154     *
01155     * @par Lua function prototype
01156     * getConfigurableDigitalInputAction(index: number) -> number
01157     *
01158     * @par Lua example
01159     * num = getConfigurableDigitalInputAction(0)
01160     *
01161     * @par JSON-RPC request example
01162     * {"jsonrpc": "2.0", "method": "robl.IoControl.getConfigurableDigitalInputAction", "params": [0], "id": 1}
01163     *
01164     * @par JSON-RPC response example
01165     * {"id": 1, "jsonrpc": "2.0", "result": "Default"}
01166     *
01167     * \endenglish
01168     */
01169 StandardInputAction getConfigurableDigitalInputAction(int index);
01170
01171 /**
01172     * @ingroup IoControl
01173     * \chinese
01174     * 设置所有数字输出状态选择为无
01175     *
01176     * @note
01177     * 当给输出状态设置为无 (StandardOutputRunState::None) 时,
01178     * 用户可以设置数字输出值。\\n
01179     * 当给输出设置状态时, 用户不可设置数字输出值, 控制器会自动设置数字输出值。\\n
01180     * 例如, 当设置 D00 的输出状态为高电平指示正在拖动示教 (StandardOutputRunState::Handguiding) 时,
01181     * 机器人进入拖动示教, D00 会自动变为高电平。
01182     * 机器人退出拖动示教, D00 会自动变为低电平。
01183     *
01184     * @return 成功返回 0; 失败返回错误码
01185     * AUBO_BUSY
01186     * AUBO_BAD_STATE
01187     * -AUBO_BAD_STATE
01188     *
01189     * @throws arcs::common_interface::AuboException
01190     *
01191     * @par Python 函数原型
01192     * setDigitalOutputRunstateDefault(self: pyaubo_sdk.IoControl) -> int
01193     *
01194     * @par Lua 函数原型
01195     * setDigitalOutputRunstateDefault() -> nil
01196     *
01197     * @par Lua 示例
01198     * setDigitalOutputRunstateDefault()
01199     *
01200     * @par JSON-RPC 请求示例
01201     * {"jsonrpc": "2.0", "method": "robl.IoControl.setDigitalOutputRunstateDefault", "params": [], "id": 1}
01202     *
01203     * @par JSON-RPC 响应示例
01204     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01205     *
01206     * \endchinese
01207     * \english
01208     * Set all digital output runstates to None.
01209     *
01210     * @note
01211     * When the output runstate is set to None (StandardOutputRunState::None),
01212     * users can set the digital output value.\\n
01213     * When the output runstate is set, users cannot set the digital output
01214     * value, and the controller will set it automatically.\\n For example, when

```



```

01215     * D00's output runstate is set to indicate hand-guiding
01216     * (StandardOutputRunState::Handguiding), the robot enters hand-guiding mode
01217     * and D00 will automatically become high. When the robot exits
01218     * hand-guiding, D00 will automatically become low.
01219     *
01220     * @return Returns 0 on success; error code on failure.
01221     * AUBO_BUSY
01222     * AUBO_BAD_STATE
01223     * -AUBO_BAD_STATE
01224     *
01225     * @throws arcs::common_interface::AuboException
01226     *
01227     * @par Python function prototype
01228     * setDigitalOutputRunstateDefault(self: pyaubo_sdk.IoControl) -> int
01229     *
01230     * @par Lua function prototype
01231     * setDigitalOutputRunstateDefault() -> nil
01232     *
01233     * @par Lua example
01234     * setDigitalOutputRunstateDefault()
01235     *
01236     * @par JSON-RPC request example
01237     * {"jsonrpc": "2.0", "method": "rob1.IoControl.setDigitalOutputRunstateDefault", "params": [], "id": 1}
01238     *
01239     * @par JSON-RPC response example
01240     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01241     *
01242     * \endenglish
01243     */
01244 int setDigitalOutputRunstateDefault();
01245
01246 /**
01247     * @ingroup IoControl
01248     * \chinese
01249     * 设置标准数字输出状态选择
01250     *
01251     * @note
01252     * 当给输出状态设置为无 (StandardOutputRunState::None) 时,
01253     * 用户可以设置数字输出值。 \n
01254     * 当给输出设置状态时, 用户不可设置数字输出值, 控制器会自动设置数字输出值。 \n
01255     * 例如, 当设置 D00 的输出状态为高电平指示正在拖动示教 (StandardOutputRunState::Handguiding) 时,
01256     * 机器人进入拖动示教, D00 会自动变为高电平。
01257     * 机器人退出拖动示教, D00 会自动变为低电平。
01258     *
01259     * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
01260     * 例如, 0 表示第一个管脚。
01261     * @param runstate: 输出状态选择
01262     *
01263     * @return 成功返回 0; 失败返回错误码
01264     * AUBO_REQUEST_IGNORE
01265     * AUBO_BUSY
01266     * AUBO_BAD_STATE
01267     * -AUBO_INVL_ARGUMENT
01268     * -AUBO_BAD_STATE
01269     *
01270     * @throws arcs::common_interface::AuboException
01271     *
01272     * @par Python 函数原型
01273     * setStandardDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int,
01274     * arg1: arcs::common_interface::StandardOutputRunState) -> int
01275     *
01276     * @par Lua 函数原型
01277     * setStandardDigitalOutputRunstate(index: number, runstate: number) -> nil
01278     *
01279     * @par Lua 示例
01280     * setStandardDigitalOutputRunstate(0,1)
01281     *
01282     * @par JSON-RPC 请求示例
01283     * {"jsonrpc": "2.0", "method": "rob1.IoControl.setStandardDigitalOutputRunstate", "params": [0, "PowerOn"], "id": 1}
01284     *
01285     * @par JSON-RPC 响应示例
01286     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01287     *
01288     * \endchinese
01289     * \english
01290     * Set the runstate for standard digital output.
01291     *
01292     * @note
01293     * When the output runstate is set to None (StandardOutputRunState::None),
01294     * users can set the digital output value. \n
01295     * When the output runstate is set, users cannot set the digital output
01296     * value, and the controller will set it automatically. \n For example, when
01297     * D00's output runstate is set to indicate hand-guiding
01298     * (StandardOutputRunState::Handguiding), the robot enters hand-guiding mode
01299     * and D00 will automatically become high. When the robot exits
01300     * hand-guiding, D00 will automatically become low.
01301     *

```

```

01302     * @param index: Indicates the IO pin, starting from 0.
01303     * For example, 0 means the first pin.
01304     * @param runstate: Output runstate selection
01305     *
01306     * @return Returns 0 on success; error code on failure.
01307     * AUBO_REQUEST_IGNORE
01308     * AUBO_BUSY
01309     * AUBO_BAD_STATE
01310     * -AUBO_INVL_ARGUMENT
01311     * -AUBO_BAD_STATE
01312     *
01313     * @throws arcs::common_interface::AuboException
01314     *
01315     * @par Python function prototype
01316     * setStandardDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int,
01317     * arg1: arcs::common_interface::StandardOutputRunState) -> int
01318     *
01319     * @par Lua function prototype
01320     * setStandardDigitalOutputRunstate(index: number, runstate: number) -> nil
01321     *
01322     * @par Lua example
01323     * setStandardDigitalOutputRunstate(0,1)
01324     *
01325     * @par JSON-RPC request example
01326     * {"jsonrpc": "2.0", "method": "robo.IoControl.setStandardDigitalOutputRunstate", "params": [0, "PowerOn"], "id": 1}
01327     *
01328     * @par JSON-RPC response example
01329     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01330     *
01331     * \endenglish
01332     */
01333 int setStandardDigitalOutputRunstate(int index,
01334                                     StandardOutputRunState runstate);
01335
01336 /**
01337  * @ingroup IoControl
01338  * \chinese
01339  * 设置工具端数字输出状态选择
01340  *
01341  * @note
01342  * 当给输出状态设置为无 (StandardOutputRunState::None) 时,
01343  * 用户可以设置数字输出值。 \n
01344  * 当给输出设置状态时, 用户不可设置数字输出值, 控制器会自动设置数字输出值。 \n
01345  * 例如, 当设置 TOOL_IO[0] 类型为输出且输出状态为高电平指示正在拖动示教 (StandardOutputRunState::Handguiding) 时,
01346  * 机器人进入拖动示教, TOOL_IO[0] 会自动变为高电平。
01347  * 机器人退出拖动示教, TOOL_IO[0] 会自动变为低电平。
01348  *
01349  * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
01350  * 例如, 0 表示第一个管脚。
01351  * @param runstate: 输出状态选择
01352  *
01353  * @return 成功返回 0; 失败返回错误码
01354  * AUBO_REQUEST_IGNORE
01355  * AUBO_BUSY
01356  * AUBO_BAD_STATE
01357  * -AUBO_INVL_ARGUMENT
01358  * -AUBO_BAD_STATE
01359  *
01360  * @throws arcs::common_interface::AuboException
01361  *
01362  * @par Python 函数原型
01363  * setToolDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int, arg1:
01364  * arcs::common_interface::StandardOutputRunState) -> int
01365  *
01366  * @par Lua 函数原型
01367  * setToolDigitalOutputRunstate(index: number, runstate: number) -> nil
01368  *
01369  * @par Lua 示例
01370  * setToolDigitalOutputRunstate(0,1)
01371  *
01372  * @par JSON-RPC 请求示例
01373  * {"jsonrpc": "2.0", "method": "robo.IoControl.setToolDigitalOutputRunstate", "params": [0, "None"], "id": 1}
01374  *
01375  * @par JSON-RPC 响应示例
01376  * {"id": 1, "jsonrpc": "2.0", "result": 0}
01377  *
01378  * \endchinese
01379  * \english
01380  * Set the runstate for tool digital output.
01381  *
01382  * @note
01383  * When the output runstate is set to None (StandardOutputRunState::None),
01384  * users can set the digital output value. \n
01385  * When the output runstate is set, users cannot set the digital output
01386  * value, and the controller will set it automatically. \n For example, when
01387  * TOOL_IO[0] is set as output and its runstate is set to indicate
01388  * hand-guiding (StandardOutputRunState::Handguiding), the robot enters

```

```

01389     * hand-guiding mode and TOOL_IO[0] will automatically become high. When the
01390     * robot exits hand-guiding, TOOL_IO[0] will automatically become low.
01391     *
01392     * @param index: Indicates the IO pin, starting from 0.
01393     * For example, 0 means the first pin.
01394     * @param runstate: Output runstate selection
01395     *
01396     * @return Returns 0 on success; error code on failure.
01397     * AUBO_REQUEST_IGNORE
01398     * AUBO_BUSY
01399     * AUBO_BAD_STATE
01400     * -AUBO_INVL_ARGUMENT
01401     * -AUBO_BAD_STATE
01402     *
01403     * @throws arcs::common_interface::AuboException
01404     *
01405     * @par Python function prototype
01406     * setToolDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int, arg1:
01407     * arcs::common_interface::StandardOutputRunState) -> int
01408     *
01409     * @par Lua function prototype
01410     * setToolDigitalOutputRunstate(index: number, runstate: number) -> nil
01411     *
01412     * @par Lua example
01413     * setToolDigitalOutputRunstate(0,1)
01414     *
01415     * @par JSON-RPC request example
01416     * {"jsonrpc": "2.0", "method": "rob1.IoControl.setToolDigitalOutputRunstate", "params": [0, "None"], "id": 1}
01417     *
01418     * @par JSON-RPC response example
01419     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01420     *
01421     * \endenglish
01422     */
01423 int setToolDigitalOutputRunstate(int index,
01424                                 StandardOutputRunState runstate);
01425
01426 /**
01427  * @ingroup IoControl
01428  * \chinese
01429  * 设置可配置数字输出状态选择
01430  *
01431  * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
01432  * 例如, 0 表示第一个管脚。
01433  * @param runstate: 输出状态选择
01434  *
01435  * @return 成功返回 0; 失败返回错误码
01436  * AUBO_REQUEST_IGNORE
01437  * AUBO_BUSY
01438  * AUBO_BAD_STATE
01439  * -AUBO_INVL_ARGUMENT
01440  * -AUBO_BAD_STATE
01441  *
01442  * @throws arcs::common_interface::AuboException
01443  *
01444  * @par Python 函数原型
01445  * setConfigurableDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0:
01446  * int, arg1: arcs::common_interface::StandardOutputRunState) -> int
01447  *
01448  * @par Lua 函数原型
01449  * setConfigurableDigitalOutputRunstate(index: number, runstate: number) ->
01450  * nil
01451  *
01452  * @par Lua 示例
01453  * setConfigurableDigitalOutputRunstate(0,1)
01454  *
01455  * @par JSON-RPC 请求示例
01456  * {"jsonrpc": "2.0", "method": "rob1.IoControl.setConfigurableDigitalOutputRunstate", "params": [0, "None"], "id": 1}
01457  *
01458  * @par JSON-RPC 响应示例
01459  * {"id": 1, "jsonrpc": "2.0", "result": 0}
01460  *
01461  * \endchinese
01462  * \english
01463  * Set the runstate for configurable digital output.
01464  *
01465  * @param index: Indicates the IO pin, starting from 0.
01466  * For example, 0 means the first pin.
01467  * @param runstate: Output runstate selection
01468  *
01469  * @return Returns 0 on success; error code on failure.
01470  * AUBO_REQUEST_IGNORE
01471  * AUBO_BUSY
01472  * AUBO_BAD_STATE
01473  * -AUBO_INVL_ARGUMENT
01474  * -AUBO_BAD_STATE
01475  *

```

```

01476     * @throws arcs::common_interface::AuboException
01477     *
01478     * @par Python function prototype
01479     * setConfigurableDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0:
01480     * int, arg1: arcs::common_interface::StandardOutputRunState) -> int
01481     *
01482     * @par Lua function prototype
01483     * setConfigurableDigitalOutputRunstate(index: number, runstate: number) ->
01484     * nil
01485     *
01486     * @par Lua example
01487     * setConfigurableDigitalOutputRunstate(0,1)
01488     *
01489     * @par JSON-RPC request example
01490     * {"jsonrpc": "2.0", "method": "robo.IoControl.setConfigurableDigitalOutputRunstate", "params": [0, "None"], "id": 1}
01491     *
01492     * @par JSON-RPC response example
01493     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01494     *
01495     * \endenglish
01496     */
01497 int setConfigurableDigitalOutputRunstate(int index,
01498                                         StandardOutputRunState runstate);
01499
01500 /**
01501  * @ingroup IoControl
01502  * \chinese
01503  * 设置数字输出的状态选择，并支持基于路点/平面/障碍物的条件阈值判断。
01504  *
01505  * 详细说明：当需要根据机器人是否到达某个路点或与某个平面/障碍物的距离来触发数字输出时，
01506  * 可通过本接口指定条件对象及阈值。在条件满足时会将指定的数字输出设置为对应的运行状态。
01507  *
01508  * @param io_type
01509  * 数字输出类型 (0= 标准数字输出、1= 工具数字输出、2= 可配置数字输出)，用整数表示
01510  * @param index 输出口的索引，管脚编号从 0 开始
01511  * @param runstate 输出动作状态
01512  * @param object_name 条件对象的名称
01513  * @param threshold
01514  * 判断阈值：对路点为角度阈值，单位为弧度 (rad)，对平面/障碍物为距离阈值，单位为米 (m)
01515  *
01516  * @return 返回操作结果码：0 表示成功，非 0 表示失败
01517  *
01518  * @par Python 函数原型
01519  * setDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int, arg1:
01520  * int, arg2: arcs::common_interface::StandardOutputRunState, arg3: str="",
01521  * arg4: float=0.0) -> int
01522  *
01523  * @par Lua 函数原型
01524  * setDigitalOutputRunstate(io_type: number, index: number, runstate:
01525  * number, object_name: string="", threshold: number=0.0) -> nil
01526  *
01527  * @par JSON-RPC 请求示例
01528  * {"jsonrpc": "2.0", "method": "robo.IoControl.setDigitalOutputRunstate", "params":
01529  * [1,0,"WaypointArrived","P1",3],"id":1}
01530  *
01531  * @par JSON-RPC 响应示例
01532  * {"id":1,"jsonrpc":"2.0","result":0}
01533  * \endchinese
01534  * \english
01535  * Set digital output runstate with optional condition checking based on
01536  * waypoint, plane or obstacle thresholds.
01537  *
01538  * Use this API to control a digital output when a specified condition is
01539  * met (for example, waypoint arrival or distance to a plane/obstacle).
01540  * Specify an object type, name and a threshold value to enable conditional
01541  * triggering. When the condition is satisfied, the given digital output
01542  * will be set to the requested runstate.
01543  *
01544  * @param io_type Numeric IO type (e.g. 0=standard, 1=tool, 2=configurable)
01545  * @param index IO pin index starting from 0
01546  * @param runstate The desired output action
01547  * @param object_name Unique identifier of the condition object (optional)
01548  * @param threshold Condition threshold: angle in radians for waypoints,
01549  * or distance in meters for planes/obstacles (optional)
01550  *
01551  * @return Returns 0 on success, non-zero error code on failure.
01552  * \endenglish
01553  */
01554 int setDigitalOutputRunstate(int io_type, int index,
01555                             StandardOutputRunState runstate,
01556                             const std::string &object_name,
01557                             double threshold);
01558
01559 /**
01560  * @ingroup IoControl
01561  * \chinese
01562  * 获取标准数字输出状态选择

```

```

01562      *
01563      * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
01564      * 例如, 0 表示第一个管脚。
01565      * @return 输出状态选择
01566      *
01567      * @throws arcs::common_interface::AuboException
01568      *
01569      * @par Python 函数原型
01570      * getStandardDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int)
01571      * -> arcs::common_interface::StandardOutputRunState
01572      *
01573      * @par Lua 函数原型
01574      * getStandardDigitalOutputRunstate(index: number) -> number
01575      *
01576      * @par Lua 示例
01577      * num = getStandardDigitalOutputRunstate(0)
01578      *
01579      * @par JSON-RPC 请求示例
01580      * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardDigitalOutputRunstate", "params": [0], "id": 1}
01581      *
01582      * @par JSON-RPC 响应示例
01583      * {"id": 1, "jsonrpc": "2.0", "result": "None"}
01584      *
01585      * \endchinese
01586      * \english
01587      * Get the runstate for standard digital output.
01588      *
01589      * @param index: Indicates the IO pin, starting from 0.
01590      * For example, 0 means the first pin.
01591      * @return Output runstate selection
01592      *
01593      * @throws arcs::common_interface::AuboException
01594      *
01595      * @par Python function prototype
01596      * getStandardDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int)
01597      * -> arcs::common_interface::StandardOutputRunState
01598      *
01599      * @par Lua function prototype
01600      * getStandardDigitalOutputRunstate(index: number) -> number
01601      *
01602      * @par Lua example
01603      * num = getStandardDigitalOutputRunstate(0)
01604      *
01605      * @par JSON-RPC request example
01606      * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardDigitalOutputRunstate", "params": [0], "id": 1}
01607      *
01608      * @par JSON-RPC response example
01609      * {"id": 1, "jsonrpc": "2.0", "result": "None"}
01610      *
01611      * \endenglish
01612      */
01613 StandardOutputRunState getStandardDigitalOutputRunstate(int index);
01614
01615 /**
01616  * @ingroup IoControl
01617  * \chinese
01618  * 获取工具端数字输出状态选择
01619  *
01620  * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
01621  * 例如, 0 表示第一个管脚。
01622  * @return 输出状态选择
01623  *
01624  * @throws arcs::common_interface::AuboException
01625  *
01626  * @par Python 函数原型
01627  * getToolDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int) ->
01628  * arcs::common_interface::StandardOutputRunState
01629  *
01630  * @par Lua 函数原型
01631  * getToolDigitalOutputRunstate(index: number) -> number
01632  *
01633  * @par Lua 示例
01634  * num = getToolDigitalOutputRunstate(0)
01635  *
01636  * @par JSON-RPC 请求示例
01637  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolDigitalOutputRunstate", "params": [0], "id": 1}
01638  *
01639  * @par JSON-RPC 响应示例
01640  * {"id": 1, "jsonrpc": "2.0", "result": "None"}
01641  *
01642  * \endchinese
01643  * \english
01644  * Get the runstate for tool digital output.
01645  *
01646  * @param index: Indicates the IO pin, starting from 0.
01647  * For example, 0 means the first pin.
01648  * @return Output runstate selection

```

```

01649     *
01650     * @throws arcs::common_interface::AuboException
01651     *
01652     * @par Python function prototype
01653     * getToolDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int) ->
01654     * arcs::common_interface::StandardOutputRunState
01655     *
01656     * @par Lua function prototype
01657     * getToolDigitalOutputRunstate(index: number) -> number
01658     *
01659     * @par Lua example
01660     * num = getToolDigitalOutputRunstate(0)
01661     *
01662     * @par JSON-RPC request example
01663     * {"jsonrpc": "2.0", "method": "robl.IoControl.getToolDigitalOutputRunstate", "params": [0], "id": 1}
01664     *
01665     * @par JSON-RPC response example
01666     * {"id": 1, "jsonrpc": "2.0", "result": "None"}
01667     *
01668     * \endenglish
01669     */
01670 StandardOutputRunState getToolDigitalOutputRunstate(int index);
01671
01672 /**
01673     * @ingroup IoControl
01674     * \chinese
01675     * 获取可配置数字输出状态选择
01676     *
01677     * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
01678     * 例如, 0 表示第一个管脚。
01679     * @return 输出状态选择
01680     *
01681     * @throws arcs::common_interface::AuboException
01682     *
01683     * @par Python 函数原型
01684     * getConfigurableDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0:
01685     * int)
01686     * -> arcs::common_interface::StandardOutputRunState
01687     *
01688     * @par Lua 函数原型
01689     * getConfigurableDigitalOutputRunstate(index: number) -> number
01690     *
01691     * @par Lua 示例
01692     * num = getConfigurableDigitalOutputRunstate(0)
01693     *
01694     * @par JSON-RPC 请求示例
01695     * {"jsonrpc": "2.0", "method": "robl.IoControl.getConfigurableDigitalOutputRunstate", "params": [0], "id": 1}
01696     *
01697     * @par JSON-RPC 响应示例
01698     * {"id": 1, "jsonrpc": "2.0", "result": "None"}
01699     *
01700     * \endchinese
01701     * \english
01702     * Get the runstate for configurable digital output.
01703     *
01704     * @param index: Indicates the IO pin, starting from 0.
01705     * For example, 0 means the first pin.
01706     * @return Output runstate selection
01707     *
01708     * @throws arcs::common_interface::AuboException
01709     *
01710     * @par Python function prototype
01711     * getConfigurableDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0:
01712     * int)
01713     * -> arcs::common_interface::StandardOutputRunState
01714     *
01715     * @par Lua function prototype
01716     * getConfigurableDigitalOutputRunstate(index: number) -> number
01717     *
01718     * @par Lua example
01719     * num = getConfigurableDigitalOutputRunstate(0)
01720     *
01721     * @par JSON-RPC request example
01722     * {"jsonrpc": "2.0", "method": "robl.IoControl.getConfigurableDigitalOutputRunstate", "params": [0], "id": 1}
01723     *
01724     * @par JSON-RPC response example
01725     * {"id": 1, "jsonrpc": "2.0", "result": "None"}
01726     *
01727     * \endenglish
01728     */
01729 StandardOutputRunState getConfigurableDigitalOutputRunstate(int index);
01730
01731 /**
01732     * @ingroup IoControl
01733     * \chinese
01734     * 获取指定类型数字输出状态选择。
01735     *

```

```

01736 * 用于查询不同类型的数字输出（标准数字输出、工具端数字输出、可配置数字输出）的
01737 * 当前运行状态选择（例如 WaypointArrived 等），
01738 * 一般跟 setDigitalOutputRunstate 一起使用。
01739 *
01740 * @param io_type
01741 * 数字输出类型（0= 标准数字输出、1= 工具数字输出、2= 可配置数字输出）
01742 * @param index 输出索引，管脚编号从 0 开始
01743 * @return 返回对应的 `StandardOutputRunState` 枚举值
01744 *
01745 * @throws arcs::common_interface::AuboException
01746 *
01747 * @par Python 函数原型
01748 * getDigitalOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int, arg1:
01749 * int) -> arcs::common_interface::StandardOutputRunState
01750 *
01751 * @par Lua 函数原型
01752 * getDigitalOutputRunstate(io_type: number, index: number) -> number
01753 *
01754 * @par JSON-RPC 请求示例
01755 * {"jsonrpc": "2.0", "method": "robl.IoControl.getDigitalOutputRunstate", "params": [1, 0], "id": 1}
01756 *
01757 * @par JSON-RPC 响应示例
01758 * {"id": 1, "jsonrpc": "2.0", "result": "None"}
01759 * \endchinese
01760 * \english
01761 * Get the current runstate of a digital output of the specified type.
01762 *
01763 * Use this API to query the runstate selection (for example None,
01764 * Handguiding, WaypointArrived) of different digital output types
01765 * (standard, tool, configurable).
01766 *
01767 * @param io_type Numeric IO type (0=standard, 1=tool, 2=configurable)
01768 * @param index IO pin index starting from 0
01769 * @return Returns the corresponding `StandardOutputRunState` enum value
01770 *
01771 * @throws arcs::common_interface::AuboException
01772 * \endenglish
01773 */
01774 StandardOutputRunState getDigitalOutputRunstate(int io_type, int index);
01775 /**
01776 * @ingroup IoControl
01777 * \chinese
01778 * 新增一条 DIO Cross Connection 逻辑规则。
01779 *
01780 *
01781 * 支持 digital/bool 类 DIO 信号和虚拟 DO 信号。规则添加成功后立即生效；
01782 * DI 只能作为 actor 被读取，不能作为 result 写入。actor_inverts 表示各
01783 * actor 读取原始 bool 值后是否取反参与运算，不改变 IO 的硬件方向。
01784 *
01785 * result_type 和 actor_types 使用同一套数字类型。作为 result 时表示输出侧，
01786 * 作为 actor 时表示输入侧或可读的虚拟 DO：
01787 * - 0：标准数字 IO，result 表示标准 DO，actor 表示标准 DI。
01788 * - 1：工具数字 IO，result 表示工具 DO，actor 表示工具 DI。
01789 * - 2：可配置数字 IO，result 表示可配置 DO，actor 表示可配置 DI。
01790 * - 3：虚拟 DO，不占用真实 DO 口，可作为级联中间结果；作为 result 时写入
01791 * 虚拟 DO，作为 actor 时读取虚拟 DO。
01792 *
01793 * operator_type 由控制器实现解释，建议统一为 0=FWD、1=NOT、2=AND、
01794 * 3=OR。FWD/NOT 必须恰好 1 个 actor，AND/OR 必须为 2-5 个 actor。
01795 * actor_types、actor_pins、actor_inverts 长度必须一致。
01796 *
01797 * 限制：
01798 * - Cross Connection 规则总数最大 300。
01799 * - 规则 id 最大 32 个字符，且不能重复。
01800 * - 虚拟 DO 级联深度最大 20。
01801 * - 不允许形成闭环依赖。
01802 * - 不允许多个规则写同一个 result，包括真实 DO 和虚拟 DO。
01803 * - 虚拟 DO actor 必须由前序规则生成，读取未定义虚拟 DO 应返回失败。
01804 * - 虚拟 DO pin/index 只要求非负，不占用真实 DO 口。
01805 * - 已被 Cross Connection 写入的真实 DO 不应再配置 runstate；后续手动
01806 * set/pulse 该 DO 时应拒绝并提示该 DO 已被对应 Cross Connection 占用。
01807 *
01808 * @param id 规则唯一 ID
01809 * @param operator_type 逻辑运算类型
01810 * @param result_type 结果信号类型，必须是可写输出类类型或虚拟 DO
01811 * @param result_pin 结果信号 pin/index；虚拟 DO 使用虚拟信号 index
01812 * @param actor_types actor 信号类型列表，可读取 DI 或虚拟 DO
01813 * @param actor_pins actor 信号 pin/index 列表；虚拟 DO 使用虚拟信号 index
01814 * @param actor_inverts actor 极性列表
01815 * @return 成功返回 0；失败返回错误码
01816 *
01817 * @throws arcs::common_interface::AuboException
01818 *
01819 * @par Python 函数原型
01820 * addCrossConnection(self: pyaubo_sdk.IoControl, arg0: str, arg1: int,
01821 * arg2: int, arg3: int, arg4: List[int], arg5: List[int],
01822 * arg6: List[bool]) -> int

```



```

01823      *
01824      * @par Lua 函数原型
01825      * addCrossConnection(id: string, operator_type: number,
01826      * result_type: number, result_pin: number, actor_types: table,
01827      * actor_pins: table, actor_inverts: table) -> number
01828      *
01829      * @par Lua 示例
01830      * addCrossConnection("start_permit", 2, 0, 0, {0, 0}, {3, 4},
01831      * {false, true})
01832      *
01833      * @par JSON-RPC 请求示例
01834      * {"jsonrpc": "2.0", "method": "rob1.IoControl.addCrossConnection", "params": ["start_permit", 2, 0, 0, [0, 0], [3, 4],
01835      * [false, true]], "id": 1}
01836      *
01837      * @par JSON-RPC 虚拟 DO 级联示例
01838      * 先生成中间信号 VirtualDO[0] = DI[0] AND NOT DI[1], 再输出
01839      * DO[0] = VirtualDO[0] OR DI[2]:
01840      * {"jsonrpc": "2.0", "method": "rob1.IoControl.clearCrossConnections", "params": [], "id": 10}
01841      * {"jsonrpc": "2.0", "method": "rob1.IoControl.addCrossConnection", "params": ["v0_di0_and_not_di1", 2, 3, 0, [0, 0],
01842      * [0, 1], [false, true]], "id": 11}
01843      * {"jsonrpc": "2.0", "method": "rob1.IoControl.addCrossConnection", "params": ["v0_or_di2_to_do0", 3, 0, 0, [3, 0], [0, 2],
01844      * [false, false]], "id": 12}
01845      *
01846      * @par JSON-RPC 响应示例
01847      * {"id": 1, "jsonrpc": "2.0", "result": 0}
01848      *
01849      * \endchinese
01850      * \english
01851      * Add a DIO Cross Connection logic rule.
01852      *
01853      * Supports digital/bool DIO signals and virtual DO signals. The rule takes
01854      * effect immediately after it is added successfully. DI signals may only be
01855      * read as actors and cannot be written as results. actor_inverts tells the
01856      * controller whether to invert each actor's raw bool value before
01857      * evaluating the rule; it does not change the hardware IO direction.
01858      *
01859      * result_type and actor_types use the same numeric type set. As a result,
01860      * the value denotes the output side; as an actor, it denotes the input side
01861      * or a readable virtual DO:
01862      * - 0: Standard digital IO. result means standard DO, actor means standard
01863      *   DI.
01864      * - 1: Tool digital IO. result means tool DO, actor means tool DI.
01865      * - 2: Configurable digital IO. result means configurable DO, actor means
01866      *   configurable DI.
01867      * - 3: Virtual DO. It does not occupy a real DO pin and can be used as an
01868      *   intermediate result for cascaded rules. As a result it writes the
01869      *   virtual DO; as an actor it reads the virtual DO.
01870      *
01871      * operator_type is interpreted by the controller implementation; the
01872      * recommended mapping is 0=FWD, 1=NOT, 2=AND, 3=OR. FWD/NOT require exactly
01873      * one actor, and AND/OR require 2 to 5 actors. actor_types, actor_pins, and
01874      * actor_inverts must have the same length.
01875      *
01876      * Limits:
01877      * - Up to 300 Cross Connection rules are allowed.
01878      * - A rule id can contain up to 32 characters and must be unique.
01879      * - Virtual DO cascade depth is limited to 20.
01880      * - Cyclic dependencies are not allowed.
01881      * - Multiple rules cannot write the same result, including real DO and
01882      *   virtual DO results.
01883      * - A virtual DO actor must be produced by an earlier rule; reading an
01884      *   undefined virtual DO should fail.
01885      * - A virtual DO pin/index only needs to be non-negative and does not
01886      *   occupy a real DO pin.
01887      * - A real DO written by Cross Connection should not also have a runstate
01888      *   configured. Later manual set/pulse calls to that DO should be rejected
01889      *   with a message that names the owning Cross Connection.
01890      *
01891      * @param id Unique rule ID
01892      * @param operator_type Logic operator type
01893      * @param result_type Result signal type; must be writable output type or
01894      *   virtual DO
01895      * @param result_pin Result signal pin/index; virtual DO uses virtual signal
01896      *   index
01897      * @param actor_types Actor signal type list; may read DI or virtual DO
01898      * @param actor_pins Actor signal pin/index list; virtual DO uses virtual
01899      *   signal index
01900      * @param actor_inverts Actor polarity list
01901      * @return Returns 0 on success, non-zero error code on failure.
01902      *
01903      * @throws arcs::common_interface::AuboException
01904      *
01905      * @par Python function prototype
01906      * addCrossConnection(self: pyaubo_sdk.IoControl, arg0: str, arg1: int,
01907      * arg2: int, arg3: int, arg4: List[int], arg5: List[int],
01908      * arg6: List[bool]) -> int
01909      *
01910      *

```



```

01907      * @par Lua function prototype
01908      * addCrossConnection(id: string, operator_type: number,
01909      * result_type: number, result_pin: number, actor_types: table,
01910      * actor_pins: table, actor_inverts: table) -> number
01911      *
01912      * @par Lua example
01913      * addCrossConnection("start_permit", 2, 0, 0, {0, 0}, {3, 4},
01914      * {false, true})
01915      *
01916      * @par JSON-RPC request example
01917      * {"jsonrpc": "2.0", "method": "rob1.IoControl.addCrossConnection", "params": ["start_permit", 2, 0, 0, [0, 0], [3, 4],
01918      * [false, true]], "id": 1}
01919      *
01920      * @par JSON-RPC virtual DO cascade example
01921      * First generate the intermediate signal VirtualDO[0] = DI[0] AND NOT
01922      * DI[1], then drive DO[0] = VirtualDO[0] OR DI[2]:
01923      * {"jsonrpc": "2.0", "method": "rob1.IoControl.clearCrossConnections", "params": [], "id": 10}
01924      * {"jsonrpc": "2.0", "method": "rob1.IoControl.addCrossConnection", "params": ["v0_di0_and_not_di1", 2, 3, 0, [0, 0],
01925      * [0, 1], [false, true]], "id": 11}
01926      * {"jsonrpc": "2.0", "method": "rob1.IoControl.addCrossConnection", "params": ["v0_or_di2_to_do0", 3, 0, 0, [3, 0], [0, 2],
01927      * [false, false]], "id": 12}
01928      *
01929      * @par JSON-RPC response example
01930      * {"id": 1, "jsonrpc": "2.0", "result": 0}
01931      *
01932      * \endenglish
01933      */
01934      int addCrossConnection(const std::string &id, int operator_type,
01935      int result_type, int result_pin,
01936      const std::vector<int> &actor_types,
01937      const std::vector<int> &actor_pins,
01938      const std::vector<bool> &actor_inverts);
01939
01940      /**
01941      * @ingroup IoControl
01942      * \chinese
01943      * 删除一条 DIO Cross Connection 逻辑规则。
01944      *
01945      * @param id 规则唯一 ID
01946      * @return 成功返回 0；失败返回错误码
01947      *
01948      * @throws arcs::common_interface::AuboException
01949      *
01950      * @par Python 函数原型
01951      * deleteCrossConnection(self: pyaubo_sdk.IoControl, arg0: str) -> int
01952      *
01953      * @par Lua 函数原型
01954      * deleteCrossConnection(id: string) -> number
01955      *
01956      * @par JSON-RPC 请求示例
01957      * {"jsonrpc": "2.0", "method": "rob1.IoControl.deleteCrossConnection", "params": ["start_permit"], "id": 1}
01958      *
01959      * @par JSON-RPC 响应示例
01960      * {"id": 1, "jsonrpc": "2.0", "result": 0}
01961      *
01962      * \endchinese
01963      * \english
01964      * Delete a DIO Cross Connection logic rule.
01965      *
01966      * @param id Unique rule ID
01967      * @return Returns 0 on success, non-zero error code on failure.
01968      *
01969      * @throws arcs::common_interface::AuboException
01970      *
01971      * @par Python function prototype
01972      * deleteCrossConnection(self: pyaubo_sdk.IoControl, arg0: str) -> int
01973      *
01974      * @par Lua function prototype
01975      * deleteCrossConnection(id: string) -> number
01976      *
01977      * @par JSON-RPC request example
01978      * {"jsonrpc": "2.0", "method": "rob1.IoControl.deleteCrossConnection", "params": ["start_permit"], "id": 1}
01979      *
01980      * @par JSON-RPC response example
01981      * {"id": 1, "jsonrpc": "2.0", "result": 0}
01982      *
01983      * \endenglish
01984      */
01985      int deleteCrossConnection(const std::string &id);
01986
01987      /**
01988      * @ingroup IoControl
01989      * \chinese
01990      * 清空所有 DIO Cross Connection 逻辑规则。
01991      *
01992      * 该接口只清空规则，不清空或重置真实数字输出值。
01993      *
01994      * \endchinese
01995      * \english
01996      * Clear all DIO Cross Connection logic rules.
01997      *
01998      * This interface only clears rules, it does not clear or reset the real digital output value.
01999      *
02000      * \endenglish
02001      */

```

```

01991     * @return 成功返回 0; 失败返回错误码
01992     *
01993     * @throws arcs::common_interface::AuboException
01994     *
01995     * @par Python 函数原型
01996     * clearCrossConnections(self: pyaubo_sdk.IoControl) -> int
01997     *
01998     * @par Lua 函数原型
01999     * clearCrossConnections() -> number
02000     *
02001     * @par JSON-RPC 请求示例
02002     * {"jsonrpc": "2.0", "method": "rob1.IoControl.clearCrossConnections", "params": [], "id": 1}
02003     *
02004     * @par JSON-RPC 响应示例
02005     * {"id": 1, "jsonrpc": "2.0", "result": 0}
02006     *
02007     * \endchinese
02008     * \english
02009     * Clear all DIO Cross Connection logic rules.
02010     *
02011     * This only clears rules. It does not clear or reset real digital output
02012     * values.
02013     *
02014     * @return Returns 0 on success, non-zero error code on failure.
02015     *
02016     * @throws arcs::common_interface::AuboException
02017     *
02018     * @par Python function prototype
02019     * clearCrossConnections(self: pyaubo_sdk.IoControl) -> int
02020     *
02021     * @par Lua function prototype
02022     * clearCrossConnections() -> number
02023     *
02024     * @par JSON-RPC request example
02025     * {"jsonrpc": "2.0", "method": "rob1.IoControl.clearCrossConnections", "params": [], "id": 1}
02026     *
02027     * @par JSON-RPC response example
02028     * {"id": 1, "jsonrpc": "2.0", "result": 0}
02029     *
02030     * \endenglish
02031     */
02032 int clearCrossConnections();
02033
02034 /**
02035  * @ingroup IoControl
02036  * \chinese
02037  * 设置标准模拟输出状态选择
02038  *
02039  * @note
02040  * 当给输出状态设置为无 (StandardOutputRunState::None) 时,
02041  * 用户可以设置模拟输出值。 \n
02042  * 当给输出设置状态时, 用户不可设置模拟输出值, 控制器会自动设置模拟输出值。 \n
02043  * 例如, 当设置 A00 的输出状态为高电平指示正在拖动示教 (StandardOutputRunState::Handguiding) 时,
02044  * 机器人进入拖动示教, A00 的值会自动变为最大值。
02045  * 机器人退出拖动示教, A00 的值会自动变为 0。
02046  *
02047  * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
02048  * 例如, 0 表示第一个管脚。
02049  * @param runstate: 输出状态选择
02050  * @return 成功返回 0; 失败返回错误码
02051  * AUBO_REQUEST_IGNORE
02052  * AUBO_BUSY
02053  * AUBO_BAD_STATE
02054  * -AUBO_INVL_ARGUMENT
02055  * -AUBO_BAD_STATE
02056  *
02057  * @throws arcs::common_interface::AuboException
02058  *
02059  * @par Python 函数原型
02060  * setStandardAnalogOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int,
02061  * arg1: arcs::common_interface::StandardOutputRunState) -> int
02062  *
02063  * @par Lua 函数原型
02064  * setStandardAnalogOutputRunstate(index: number, runstate: number) -> nil
02065  *
02066  * @par Lua 示例
02067  * setStandardAnalogOutputRunstate(0,6)
02068  *
02069  * @par JSON-RPC 请求示例
02070  * {"jsonrpc": "2.0", "method": "rob1.IoControl.setStandardAnalogOutputRunstate", "params": [0, "None"], "id": 1}
02071  *
02072  * @par JSON-RPC 响应示例
02073  * {"id": 1, "jsonrpc": "2.0", "result": 0}
02074  *
02075  * \endchinese
02076  * \english
02077  * Set the runstate for standard analog output.

```

```

02078      *
02079      * @note
02080      * When the output runstate is set to None (StandardOutputRunState::None),
02081      * users can set the analog output value.\n
02082      * When the output runstate is set, users cannot set the analog output
02083      * value, and the controller will set it automatically.\n For example, when
02084      * A00's output runstate is set to indicate hand-guiding
02085      * (StandardOutputRunState::Handguiding), the robot enters hand-guiding mode
02086      * and A00 will automatically become the maximum value. When the robot exits
02087      * hand-guiding, A00 will automatically become 0.
02088      *
02089      * @param index: Indicates the IO pin, starting from 0.
02090      * For example, 0 means the first pin.
02091      * @param runstate: Output runstate selection
02092      * @return Returns 0 on success; error code on failure.
02093      * AUBO_REQUEST_IGNORE
02094      * AUBO_BUSY
02095      * AUBO_BAD_STATE
02096      * -AUBO_INVL_ARGUMENT
02097      * -AUBO_BAD_STATE
02098      *
02099      * @throws arcs::common_interface::AuboException
02100      *
02101      * @par Python function prototype
02102      * setStandardAnalogOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int,
02103      * arg1: arcs::common_interface::StandardOutputRunState) -> int
02104      *
02105      * @par Lua function prototype
02106      * setStandardAnalogOutputRunstate(index: number, runstate: number) -> nil
02107      *
02108      * @par Lua example
02109      * setStandardAnalogOutputRunstate(0,6)
02110      *
02111      * @par JSON-RPC request example
02112      * {"jsonrpc":"2.0","method":"rob1.IoControl.setStandardAnalogOutputRunstate","params":[0,"None"],"id":1}
02113      *
02114      * @par JSON-RPC response example
02115      * {"id":1,"jsonrpc":"2.0","result":0}
02116      *
02117      * \endenglish
02118      */
02119 int setStandardAnalogOutputRunstate(int index,
02120                                     StandardOutputRunState runstate);
02121
02122 /**
02123  * @ingroup IoControl
02124  * \chinese
02125  * 设置工具端模拟输出状态选择
02126  *
02127  * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
02128  * 例如, 0 表示第一个管脚。
02129  * @param runstate: 输出状态选择
02130  * @return 成功返回 0; 失败返回错误码
02131  * AUBO_REQUEST_IGNORE
02132  * AUBO_BUSY
02133  * AUBO_BAD_STATE
02134  * -AUBO_INVL_ARGUMENT
02135  * -AUBO_BAD_STATE
02136  *
02137  * @throws arcs::common_interface::AuboException
02138  *
02139  * @par Python 函数原型
02140  * setToolAnalogOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int, arg1:
02141  * arcs::common_interface::StandardOutputRunState) -> int
02142  *
02143  * @par Lua 函数原型
02144  * setToolAnalogOutputRunstate(index: number, runstate: number) -> nil
02145  *
02146  * @par Lua 示例
02147  * setToolAnalogOutputRunstate(0,1)
02148  *
02149  * @par JSON-RPC 请求示例
02150  * {"jsonrpc":"2.0","method":"rob1.IoControl.setToolAnalogOutputRunstate","params":[0,"None"],"id":1}
02151  *
02152  * @par JSON-RPC 响应示例
02153  * {"id":1,"jsonrpc":"2.0","result":0}
02154  *
02155  * \endchinese
02156  * \english
02157  * Set the runstate for tool analog output.
02158  *
02159  * @param index: Indicates the IO pin, starting from 0.
02160  * For example, 0 means the first pin.
02161  * @param runstate: Output runstate selection
02162  * @return Returns 0 on success; error code on failure.
02163  * AUBO_REQUEST_IGNORE
02164  * AUBO_BUSY

```

```

02165     * AUBO_BAD_STATE
02166     * -AUBO_INVL_ARGUMENT
02167     * -AUBO_BAD_STATE
02168     *
02169     * @throws arcs::common_interface::AuboException
02170     *
02171     * @par Python function prototype
02172     * setToolAnalogOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int, arg1:
02173     * arcs::common_interface::StandardOutputRunState) -> int
02174     *
02175     * @par Lua function prototype
02176     * setToolAnalogOutputRunstate(index: number, runstate: number) -> nil
02177     *
02178     * @par Lua example
02179     * setToolAnalogOutputRunstate(0,1)
02180     *
02181     * @par JSON-RPC request example
02182     * {"jsonrpc":"2.0","method":"rob1.IoControl.setToolAnalogOutputRunstate","params":[0,"None"],"id":1}
02183     *
02184     * @par JSON-RPC response example
02185     * {"id":1,"jsonrpc":"2.0","result":0}
02186     *
02187     * \endenglish
02188     */
02189 int setToolAnalogOutputRunstate(int index, StandardOutputRunState runstate);
02190
02191 /**
02192     * @ingroup IoControl
02193     * \chinese
02194     * 获取标准模拟输出状态选择
02195     *
02196     * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
02197     * 例如, 0 表示第一个管脚。
02198     * @return 标准模拟输出状态选择
02199     *
02200     * @throws arcs::common_interface::AuboException
02201     *
02202     * @par Python 函数原型
02203     * getStandardAnalogOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int) ->
02204     * arcs::common_interface::StandardOutputRunState
02205     *
02206     * @par Lua 函数原型
02207     * getStandardAnalogOutputRunstate(index: number) -> number
02208     *
02209     * @par Lua 示例
02210     * num = getStandardAnalogOutputRunstate(0)
02211     *
02212     * @par JSON-RPC 请求示例
02213     * {"jsonrpc":"2.0","method":"rob1.IoControl.getStandardAnalogOutputRunstate","params":[0],"id":1}
02214     *
02215     * @par JSON-RPC 响应示例
02216     * {"id":1,"jsonrpc":"2.0","result":"None"}
02217     *
02218     * \endchinese
02219     * \english
02220     * Get the runstate for standard analog output.
02221     *
02222     * @param index: Indicates the IO pin, starting from 0.
02223     * For example, 0 means the first pin.
02224     * @return Standard analog output runstate selection
02225     *
02226     * @throws arcs::common_interface::AuboException
02227     *
02228     * @par Python function prototype
02229     * getStandardAnalogOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int) ->
02230     * arcs::common_interface::StandardOutputRunState
02231     *
02232     * @par Lua function prototype
02233     * getStandardAnalogOutputRunstate(index: number) -> number
02234     *
02235     * @par Lua example
02236     * num = getStandardAnalogOutputRunstate(0)
02237     *
02238     * @par JSON-RPC request example
02239     * {"jsonrpc":"2.0","method":"rob1.IoControl.getStandardAnalogOutputRunstate","params":[0],"id":1}
02240     *
02241     * @par JSON-RPC response example
02242     * {"id":1,"jsonrpc":"2.0","result":"None"}
02243     *
02244     * \endenglish
02245     */
02246 StandardOutputRunState getStandardAnalogOutputRunstate(int index);
02247
02248 /**
02249     * @ingroup IoControl
02250     * \chinese
02251     * 获取工具端模拟输出状态选择

```

```

02252      *
02253      * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
02254      * 例如, 0 表示第一个管脚。
02255      * @return 工具端模拟输出状态选择
02256      *
02257      * @throws arcs::common_interface::AuboException
02258      *
02259      * @par Python 函数原型
02260      * getToolAnalogOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int) ->
02261      * arcs::common_interface::StandardOutputRunState
02262      *
02263      * @par Lua 函数原型
02264      * getToolAnalogOutputRunstate(index: number) -> number
02265      *
02266      * @par Lua 示例
02267      * num = getToolAnalogOutputRunstate(0)
02268      *
02269      * @par JSON-RPC 请求示例
02270      * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolAnalogOutputRunstate", "params": [0], "id": 1}
02271      *
02272      * @par JSON-RPC 响应示例
02273      * {"id": 1, "jsonrpc": "2.0", "result": "None"}
02274      *
02275      * \endchinese
02276      * \english
02277      * Get the runstate for tool analog output.
02278      *
02279      * @param index: Indicates the IO pin, starting from 0.
02280      * For example, 0 means the first pin.
02281      * @return Tool analog output runstate selection
02282      *
02283      * @throws arcs::common_interface::AuboException
02284      *
02285      * @par Python function prototype
02286      * getToolAnalogOutputRunstate(self: pyaubo_sdk.IoControl, arg0: int) ->
02287      * arcs::common_interface::StandardOutputRunState
02288      *
02289      * @par Lua function prototype
02290      * getToolAnalogOutputRunstate(index: number) -> number
02291      *
02292      * @par Lua example
02293      * num = getToolAnalogOutputRunstate(0)
02294      *
02295      * @par JSON-RPC request example
02296      * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolAnalogOutputRunstate", "params": [0], "id": 1}
02297      *
02298      * @par JSON-RPC response example
02299      * {"id": 1, "jsonrpc": "2.0", "result": "None"}
02300      *
02301      * \endenglish
02302      */
02303      StandardOutputRunState getToolAnalogOutputRunstate(int index);
02304
02305      /**
02306      * @ingroup IoControl
02307      * \chinese
02308      * 设置所有数字输出急停后状态为默认 (不做改变)
02309      *
02310      * @return 成功返回 0; 失败返回错误码
02311      * AUBO_BUSY
02312      * AUBO_BAD_STATE
02313      * -AUBO_BAD_STATE
02314      *
02315      * @throws arcs::common_interface::AuboException
02316      *
02317      * @par Python 函数原型
02318      * setDigitalOutputAfterEStopDefault(self: pyaubo_sdk.IoControl) -> int
02319      *
02320      * @par Lua 函数原型
02321      * setDigitalOutputAfterEStopDefault() -> nil
02322      *
02323      * @par Lua 示例
02324      * setDigitalOutputAfterEStopDefault()
02325      *
02326      * @par JSON-RPC 请求示例
02327      * {"jsonrpc": "2.0", "method": "rob1.IoControl.setDigitalOutputAfterEStopDefault", "params": [], "id": 1}
02328      *
02329      * @par JSON-RPC 响应示例
02330      * {"id": 1, "jsonrpc": "2.0", "result": 0}
02331      *
02332      * \endchinese
02333      * \english
02334      * Set all digital output states after emergency stop to default(no change)
02335      *
02336      * @return Returns 0 on success; error code on failure.
02337      * AUBO_BUSY
02338      * AUBO_BAD_STATE

```

```

02339     * -AUBO_BAD_STATE
02340     *
02341     * @throws arcs::common_interface::AuboException
02342     *
02343     * @par Python function prototype
02344     * setDigitalOutputAfterEStopDefault(self: pyaubo_sdk.IoControl) -> int
02345     *
02346     * @par Lua function prototype
02347     * setDigitalOutputAfterEStopDefault() -> nil
02348     *
02349     * @par Lua example
02350     * setDigitalOutputAfterEStopDefault()
02351     *
02352     * @par JSON-RPC request example
02353     * {"jsonrpc": "2.0", "method": "rob1.IoControl.setDigitalOutputAfterEStopDefault", "params": [], "id": 1}
02354     *
02355     * @par JSON-RPC response example
02356     * {"id": 1, "jsonrpc": "2.0", "result": 0}
02357     *
02358     * \endenglish
02359     */
02360 int setDigitalOutputAfterEStopDefault();
02361
02362 /**
02363  * @ingroup IoControl
02364  * \chinese
02365  * 设置所有模拟输出急停后状态为默认（不做改变）
02366  *
02367  * @return 成功返回 0；失败返回错误码
02368  * AUBO_BUSY
02369  * AUBO_BAD_STATE
02370  * -AUBO_BAD_STATE
02371  *
02372  * @throws arcs::common_interface::AuboException
02373  *
02374  * @par Python 函数原型
02375  * setAnalogOutputAfterEStopDefault(self: pyaubo_sdk.IoControl) -> int
02376  *
02377  * @par Lua 函数原型
02378  * setAnalogOutputAfterEStopDefault() -> nil
02379  *
02380  * @par Lua 示例
02381  * setAnalogOutputAfterEStopDefault()
02382  *
02383  * @par JSON-RPC 请求示例
02384  * {"jsonrpc": "2.0", "method": "rob1.IoControl.setAnalogOutputAfterEStopDefault", "params": [], "id": 1}
02385  *
02386  * @par JSON-RPC 响应示例
02387  * {"id": 1, "jsonrpc": "2.0", "result": 0}
02388  *
02389  * \endchinese
02390  * \english
02391  * Set all analog output states after emergency stop to default(no change)
02392  *
02393  * @return Returns 0 on success; error code on failure.
02394  * AUBO_BUSY
02395  * AUBO_BAD_STATE
02396  * -AUBO_BAD_STATE
02397  *
02398  * @throws arcs::common_interface::AuboException
02399  *
02400  * @par Python function prototype
02401  * setAnalogOutputAfterEStopDefault(self: pyaubo_sdk.IoControl) -> int
02402  *
02403  * @par Lua function prototype
02404  * setAnalogOutputAfterEStopDefault() -> nil
02405  *
02406  * @par Lua example
02407  * setAnalogOutputAfterEStopDefault()
02408  *
02409  * @par JSON-RPC request example
02410  * {"jsonrpc": "2.0", "method": "rob1.IoControl.setAnalogOutputAfterEStopDefault", "params": [], "id": 1}
02411  *
02412  * @par JSON-RPC response example
02413  * {"id": 1, "jsonrpc": "2.0", "result": 0}
02414  *
02415  * \endenglish
02416  */
02417 int setAnalogOutputAfterEStopDefault();
02418
02419 /**
02420  * @ingroup IoControl
02421  * \chinese
02422  * 设置标准数字输出急停后的输出值
02423  *
02424  * @param index: 表示 IO 口的管脚,
02425  * @param value: 输出值

```

```

02426     * @return 成功返回 0; 失败返回错误码
02427     * AUBO_REQUEST_IGNORE
02428     * AUBO_BUSY
02429     * AUBO_BAD_STATE
02430     * -AUBO_INVL_ARGUMENT
02431     * -AUBO_BAD_STATE
02432     *
02433     * @throws arcs::common_interface::AuboException
02434     *
02435     * @par Python 函数原型
02436     * setStandardDigitalOutputAfterEStop(self: pyaubo_sdk.IoControl, arg0: int,
02437     * arg1: bool) -> int
02438     *
02439     * @par Lua 函数原型
02440     * setStandardDigitalOutputAfterEStop(index: number, value: boolean) -> nil
02441     *
02442     * @par Lua 示例
02443     * setStandardDigitalOutputAfterEStop(0,true)
02444     *
02445     * @par JSON-RPC 请求示例
02446     * {"jsonrpc":"2.0","method":"rob1.IoControl.setStandardDigitalOutputAfterEStop","params":[0,true],"id":1}
02447     *
02448     * @par JSON-RPC 响应示例
02449     * {"id":1,"jsonrpc":"2.0","result":0}
02450     *
02451     * \endchinese
02452     * \english
02453     * Set the value of a standard digital output after emergency stop
02454     *
02455     * @param index: Indicates the IO pin.
02456     * @param value: Output value.
02457     * @return Returns 0 on success; error code on failure.
02458     * AUBO_REQUEST_IGNORE
02459     * AUBO_BUSY
02460     * AUBO_BAD_STATE
02461     * -AUBO_INVL_ARGUMENT
02462     * -AUBO_BAD_STATE
02463     *
02464     * @throws arcs::common_interface::AuboException
02465     *
02466     * @par Python function prototype
02467     * setStandardDigitalOutputAfterEStop(self: pyaubo_sdk.IoControl, arg0: int,
02468     * arg1: bool) -> int
02469     *
02470     * @par Lua function prototype
02471     * setStandardDigitalOutputAfterEStop(index: number, value: boolean) -> nil
02472     *
02473     * @par Lua example
02474     * setStandardDigitalOutputAfterEStop(0,true)
02475     *
02476     * @par JSON-RPC request example
02477     * {"jsonrpc":"2.0","method":"rob1.IoControl.setStandardDigitalOutputAfterEStop","params":[0,true],"id":1}
02478     *
02479     * @par JSON-RPC response example
02480     * {"id":1,"jsonrpc":"2.0","result":0}
02481     *
02482     * \endenglish
02483     */
02484     int setStandardDigitalOutputAfterEStop(int index, bool value);
02485
02486     /**
02487     * @ingroup IoControl
02488     * \chinese
02489     * 设置可配置数字输出急停后的输出值
02490     *
02491     * @param index: 表示 IO 口的管脚,
02492     * @param value: 输出值
02493     * @return 成功返回 0; 失败返回错误码
02494     * AUBO_REQUEST_IGNORE
02495     * AUBO_BUSY
02496     * AUBO_BAD_STATE
02497     * -AUBO_INVL_ARGUMENT
02498     * -AUBO_BAD_STATE
02499     *
02500     * @throws arcs::common_interface::AuboException
02501     *
02502     * @par Python 函数原型
02503     * setConfigurableDigitalOutputAfterEStop(self: pyaubo_sdk.IoControl, arg0:
02504     * int, arg1: bool) -> int
02505     *
02506     * @par Lua 函数原型
02507     * setConfigurableDigitalOutputAfterEStop(index: number, value: boolean) ->
02508     * nil
02509     *
02510     * @par Lua 示例
02511     * setConfigurableDigitalOutputAfterEStop(0,true)
02512     *

```



```

02513     * @par JSON-RPC 请求示例
02514     * {"jsonrpc": "2.0", "method": "rob1.IoControl.setConfigurableDigitalOutputAfterEStop", "params": [0, true], "id": 1}
02515     *
02516     * @par JSON-RPC 响应示例
02517     * {"id": 1, "jsonrpc": "2.0", "result": 0}
02518     *
02519     * \endchinese
02520     * \english
02521     * Set the value of a configurable digital output after emergency stop
02522     *
02523     * @param index: Indicates the IO pin.
02524     * @param value: Output value.
02525     * @return Returns 0 on success; error code on failure.
02526     * AUBO_REQUEST_IGNORE
02527     * AUBO_BUSY
02528     * AUBO_BAD_STATE
02529     * -AUBO_INVL_ARGUMENT
02530     * -AUBO_BAD_STATE
02531     *
02532     * @throws arcs::common_interface::AuboException
02533     *
02534     * @par Python function prototype
02535     * setConfigurableDigitalOutputAfterEStop(self: pyaubo_sdk.IoControl, arg0:
02536     * int, arg1: bool) -> int
02537     *
02538     * @par Lua function prototype
02539     * setConfigurableDigitalOutputAfterEStop(index: number, value: boolean) ->
02540     * nil
02541     *
02542     * @par Lua example
02543     * setConfigurableDigitalOutputAfterEStop(0, true)
02544     *
02545     * @par JSON-RPC request example
02546     * {"jsonrpc": "2.0", "method": "rob1.IoControl.setConfigurableDigitalOutputAfterEStop", "params": [0, true], "id": 1}
02547     *
02548     * @par JSON-RPC response example
02549     * {"id": 1, "jsonrpc": "2.0", "result": 0}
02550     *
02551     * \endenglish
02552     */
02553 int setConfigurableDigitalOutputAfterEStop(int index, bool value);
02554
02555 /**
02556  * @ingroup IoControl
02557  * \chinese
02558  * 设置标准模拟输出急停后的输出值
02559  *
02560  * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
02561  * 例如, 0 表示第一个管脚。
02562  * @param value: 模拟输出值
02563  *
02564  * @return 成功返回 0; 失败返回错误码
02565  * AUBO_REQUEST_IGNORE
02566  * AUBO_BUSY
02567  * AUBO_BAD_STATE
02568  * -AUBO_INVL_ARGUMENT
02569  * -AUBO_BAD_STATE
02570  *
02571  * @throws arcs::common_interface::AuboException
02572  *
02573  * @par Python 函数原型
02574  * setStandardAnalogOutputAfterEStop(self: pyaubo_sdk.IoControl, arg0: int,
02575  * arg1: float) -> int
02576  *
02577  * @par Lua 函数原型
02578  * setStandardAnalogOutputAfterEStop(index: number, value: number) -> nil
02579  *
02580  * @par Lua 示例
02581  * setStandardAnalogOutputAfterEStop(0, 5.4)
02582  *
02583  * \endchinese
02584  * \english
02585  * Set the value of standard analog output after emergency stop
02586  *
02587  * @param index: Indicates the IO pin, starting from 0.
02588  * For example, 0 means the first pin.
02589  * @param value: Output value.
02590  *
02591  * @return Returns 0 on success; error code on failure.
02592  * AUBO_REQUEST_IGNORE
02593  * AUBO_BUSY
02594  * AUBO_BAD_STATE
02595  * -AUBO_INVL_ARGUMENT
02596  * -AUBO_BAD_STATE
02597  *
02598  * @throws arcs::common_interface::AuboException
02599  *

```



```

02600     * @par Python function prototype
02601     * setStandardAnalogOutputAfterEStop(self: pyaubo_sdk.IoControl, arg0: int,
02602     * arg1: float) -> int
02603     *
02604     * @par Lua function prototype
02605     * setStandardAnalogOutputAfterEStop(index: number, value: number) -> nil
02606     *
02607     * @par Lua example
02608     * setStandardAnalogOutputAfterEStop(0,5.4)
02609     *
02610     * \endenglish
02611     */
02612 int setStandardAnalogOutputAfterEStop(int index, double value);
02613
02614 /**
02615  * @ingroup IoControl
02616  * \chinese
02617  * 设置标准模拟输入的范围
02618  *
02619  * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
02620  * 例如, 0 表示第一个管脚。
02621  * @param domain: 输入的范围
02622  *
02623  * @return 成功返回 0; 失败返回错误码
02624  * AUBO_REQUEST_IGNORE
02625  * AUBO_BUSY
02626  * AUBO_BAD_STATE
02627  * -AUBO_INVL_ARGUMENT
02628  * -AUBO_BAD_STATE
02629  *
02630  * @throws arcs::common_interface::AuboException
02631  *
02632  * @par Python 函数原型
02633  * setStandardAnalogInputDomain(self: pyaubo_sdk.IoControl, arg0: int, arg1:
02634  * int) -> int
02635  *
02636  * @par Lua 函数原型
02637  * setStandardAnalogInputDomain(index: number, domain: number) -> nil
02638  *
02639  * @par Lua 示例
02640  * setStandardAnalogInputDomain(0,1)
02641  *
02642  * @par JSON-RPC 请求示例
02643  * {"jsonrpc": "2.0", "method": "robl.IoControl.setStandardAnalogInputDomain", "params": [0,8], "id": 1}
02644  *
02645  * @par JSON-RPC 响应示例
02646  * {"id": 1, "jsonrpc": "2.0", "result": 0}
02647  *
02648  * \endchinese
02649  * \english
02650  * Set the range of standard analog input.
02651  *
02652  * @param index: Indicates the IO pin, starting from 0.
02653  * For example, 0 means the first pin.
02654  * @param domain: Input range
02655  *
02656  * @return Returns 0 on success; error code on failure.
02657  * AUBO_REQUEST_IGNORE
02658  * AUBO_BUSY
02659  * AUBO_BAD_STATE
02660  * -AUBO_INVL_ARGUMENT
02661  * -AUBO_BAD_STATE
02662  *
02663  * @throws arcs::common_interface::AuboException
02664  *
02665  * @par Python function prototype
02666  * setStandardAnalogInputDomain(self: pyaubo_sdk.IoControl, arg0: int, arg1:
02667  * int) -> int
02668  *
02669  * @par Lua function prototype
02670  * setStandardAnalogInputDomain(index: number, domain: number) -> nil
02671  *
02672  * @par Lua example
02673  * setStandardAnalogInputDomain(0,1)
02674  *
02675  * @par JSON-RPC request example
02676  * {"jsonrpc": "2.0", "method": "robl.IoControl.setStandardAnalogInputDomain", "params": [0,8], "id": 1}
02677  *
02678  * @par JSON-RPC response example
02679  * {"id": 1, "jsonrpc": "2.0", "result": 0}
02680  *
02681  * \endenglish
02682  */
02683 int setStandardAnalogInputDomain(int index, int domain);
02684
02685 /**
02686  * @ingroup IoControl

```

```

02687 * \chinese
02688 * 设置工具端模拟输入的范围
02689 *
02690 * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
02691 * 例如, 0 表示第一个管脚。
02692 * @param domain: 输入的范围
02693 * @return 成功返回 0; 失败返回错误码
02694 * AUBO_REQUEST_IGNORE
02695 * AUBO_BUSY
02696 * AUBO_BAD_STATE
02697 * -AUBO_INVL_ARGUMENT
02698 * -AUBO_BAD_STATE
02699 *
02700 * @throws arcs::common_interface::AuboException
02701 *
02702 * @par Python 函数原型
02703 * setToolAnalogInputDomain(self: pyaubo_sdk.IoControl, arg0: int, arg1:
02704 * int) -> int
02705 *
02706 * @par Lua 函数原型
02707 * setToolAnalogInputDomain(index: number, domain: number) -> nil
02708 *
02709 * @par Lua 示例
02710 * setToolAnalogInputDomain(0,1)
02711 *
02712 * @par JSON-RPC 请求示例
02713 * {"jsonrpc": "2.0", "method": "robo.IoControl.setToolAnalogInputDomain", "params": [0,8], "id": 1}
02714 *
02715 * @par JSON-RPC 响应示例
02716 * {"id": 1, "jsonrpc": "2.0", "result": 0}
02717 *
02718 * \endchinese
02719 * \english
02720 * Set the range of tool analog input.
02721 *
02722 * @param index: Indicates the IO pin, starting from 0.
02723 * For example, 0 means the first pin.
02724 * @param domain: Input range
02725 * @return Returns 0 on success; error code on failure.
02726 * AUBO_REQUEST_IGNORE
02727 * AUBO_BUSY
02728 * AUBO_BAD_STATE
02729 * -AUBO_INVL_ARGUMENT
02730 * -AUBO_BAD_STATE
02731 *
02732 * @throws arcs::common_interface::AuboException
02733 *
02734 * @par Python function prototype
02735 * setToolAnalogInputDomain(self: pyaubo_sdk.IoControl, arg0: int, arg1:
02736 * int) -> int
02737 *
02738 * @par Lua function prototype
02739 * setToolAnalogInputDomain(index: number, domain: number) -> nil
02740 *
02741 * @par Lua example
02742 * setToolAnalogInputDomain(0,1)
02743 *
02744 * @par JSON-RPC request example
02745 * {"jsonrpc": "2.0", "method": "robo.IoControl.setToolAnalogInputDomain", "params": [0,8], "id": 1}
02746 *
02747 * @par JSON-RPC response example
02748 * {"id": 1, "jsonrpc": "2.0", "result": 0}
02749 *
02750 * \endenglish
02751 */
02752 int setToolAnalogInputDomain(int index, int domain);
02753
02754 /**
02755 * @ingroup IoControl
02756 * \chinese
02757 * 获取标准模式输入范围
02758 *
02759 * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
02760 * 例如, 0 表示第一个管脚。
02761 * @return 标准模式输入范围
02762 *
02763 * @throws arcs::common_interface::AuboException
02764 *
02765 * @par Python 函数原型
02766 * getStandardAnalogInputDomain(self: pyaubo_sdk.IoControl, arg0: int) ->
02767 * int
02768 *
02769 * @par Lua 函数原型
02770 * getStandardAnalogInputDomain(index: number) -> number
02771 *
02772 * @par Lua 示例
02773 * num = getStandardAnalogInputDomain(0)

```

```

02774      *
02775      * @par JSON-RPC 请求示例
02776      * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardAnalogInputDomain", "params": [0], "id": 1}
02777      *
02778      * @par JSON-RPC 响应示例
02779      * {"id": 1, "jsonrpc": "2.0", "result": 0}
02780      *
02781      * \endchinese
02782      * \english
02783      * Get the domain of standard analog input.
02784      *
02785      * @param index: Indicates the IO pin, starting from 0.
02786      * For example, 0 means the first pin.
02787      * @return Standard analog input domain
02788      *
02789      * @throws arcs::common_interface::AuboException
02790      *
02791      * @par Python function prototype
02792      * getStandardAnalogInputDomain(self: pyaubo_sdk.IoControl, arg0: int) ->
02793      * int
02794      *
02795      * @par Lua function prototype
02796      * getStandardAnalogInputDomain(index: number) -> number
02797      *
02798      * @par Lua example
02799      * num = getStandardAnalogInputDomain(0)
02800      *
02801      * @par JSON-RPC request example
02802      * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardAnalogInputDomain", "params": [0], "id": 1}
02803      *
02804      * @par JSON-RPC response example
02805      * {"id": 1, "jsonrpc": "2.0", "result": 0}
02806      *
02807      * \endenglish
02808      */
02809 int getStandardAnalogInputDomain(int index);
02810
02811 /**
02812  * @ingroup IoControl
02813  * \chinese
02814  * 获取工具端模式输入范围
02815  *
02816  * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
02817  * 例如, 0 表示第一个管脚。
02818  * @return 工具端模式输入范围
02819  *
02820  * @throws arcs::common_interface::AuboException
02821  *
02822  * @par Python 函数原型
02823  * getToolAnalogInputDomain(self: pyaubo_sdk.IoControl, arg0: int) -> int
02824  *
02825  * @par Lua 函数原型
02826  * getToolAnalogInputDomain(index: number) -> number
02827  *
02828  * @par Lua 示例
02829  * num = getToolAnalogInputDomain(0)
02830  *
02831  * @par JSON-RPC 请求示例
02832  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolAnalogInputDomain", "params": [0], "id": 1}
02833  *
02834  * @par JSON-RPC 响应示例
02835  * {"id": 1, "jsonrpc": "2.0", "result": 10}
02836  *
02837  * \endchinese
02838  * \english
02839  * Get the domain of tool analog input.
02840  *
02841  * @param index: Indicates the IO pin, starting from 0.
02842  * For example, 0 means the first pin.
02843  * @return Tool analog input domain
02844  *
02845  * @throws arcs::common_interface::AuboException
02846  *
02847  * @par Python function prototype
02848  * getToolAnalogInputDomain(self: pyaubo_sdk.IoControl, arg0: int) -> int
02849  *
02850  * @par Lua function prototype
02851  * getToolAnalogInputDomain(index: number) -> number
02852  *
02853  * @par Lua example
02854  * num = getToolAnalogInputDomain(0)
02855  *
02856  * @par JSON-RPC request example
02857  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolAnalogInputDomain", "params": [0], "id": 1}
02858  *
02859  * @par JSON-RPC response example
02860  * {"id": 1, "jsonrpc": "2.0", "result": 10}

```

```

02861     *
02862     * \endenglish
02863     */
02864 int getToolAnalogInputDomain(int index);
02865
02866 /**
02867  * @ingroup IoControl
02868  * \chinese
02869  * 设置标准模拟输出的范围
02870  *
02871  * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
02872  * 例如, 0 表示第一个管脚。
02873  * @param domain: 输出的范围
02874  *
02875  * @return 成功返回 0; 失败返回错误码
02876  * AUBO_REQUEST_IGNORE
02877  * AUBO_BUSY
02878  * AUBO_BAD_STATE
02879  * -AUBO_INVL_ARGUMENT
02880  * -AUBO_BAD_STATE
02881  *
02882  * @throws arcs::common_interface::AuboException
02883  *
02884  * @par Python 函数原型
02885  * setStandardAnalogOutputDomain(self: pyaubo_sdk.IoControl, arg0: int,
02886  * arg1: int) -> int
02887  *
02888  * @par Lua 函数原型
02889  * setStandardAnalogOutputDomain(index: number, domain: number) -> nil
02890  *
02891  * @par Lua 示例
02892  * setStandardAnalogOutputDomain(0,1)
02893  *
02894  * @par JSON-RPC 请求示例
02895  * {"jsonrpc": "2.0", "method": "robo.IoControl.setStandardAnalogOutputDomain", "params": [0,8], "id": 1}
02896  *
02897  * @par JSON-RPC 响应示例
02898  * {"id": 1, "jsonrpc": "2.0", "result": 0}
02899  *
02900  * \endchinese
02901  * \english
02902  * Set the range of standard analog output.
02903  *
02904  * @param index: Indicates the IO pin, starting from 0.
02905  * For example, 0 means the first pin.
02906  * @param domain: Output range
02907  *
02908  * @return Returns 0 on success; error code on failure.
02909  * AUBO_REQUEST_IGNORE
02910  * AUBO_BUSY
02911  * AUBO_BAD_STATE
02912  * -AUBO_INVL_ARGUMENT
02913  * -AUBO_BAD_STATE
02914  *
02915  * @throws arcs::common_interface::AuboException
02916  *
02917  * @par Python function prototype
02918  * setStandardAnalogOutputDomain(self: pyaubo_sdk.IoControl, arg0: int,
02919  * arg1: int) -> int
02920  *
02921  * @par Lua function prototype
02922  * setStandardAnalogOutputDomain(index: number, domain: number) -> nil
02923  *
02924  * @par Lua example
02925  * setStandardAnalogOutputDomain(0,1)
02926  *
02927  * @par JSON-RPC request example
02928  * {"jsonrpc": "2.0", "method": "robo.IoControl.setStandardAnalogOutputDomain", "params": [0,8], "id": 1}
02929  *
02930  * @par JSON-RPC response example
02931  * {"id": 1, "jsonrpc": "2.0", "result": 0}
02932  *
02933  * \endenglish
02934  */
02935 int setStandardAnalogOutputDomain(int index, int domain);
02936
02937 /**
02938  * @ingroup IoControl
02939  * \chinese
02940  * 设置工具端模拟输出范围
02941  *
02942  * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
02943  * 例如, 0 表示第一个管脚。
02944  * @param domain: 输出的范围
02945  *
02946  * @return 成功返回 0; 失败返回错误码
02947  * AUBO_REQUEST_IGNORE

```

```

02948 * AUBO_BUSY
02949 * AUBO_BAD_STATE
02950 * -AUBO_INVL_ARGUMENT
02951 * -AUBO_BAD_STATE
02952 *
02953 * @throws arcs::common_interface::AuboException
02954 *
02955 * @par Python 函数原型
02956 * setToolAnalogOutputDomain(self: pyaubo_sdk.IoControl, arg0: int, arg1:
02957 * int) -> int
02958 *
02959 * @par Lua 函数原型
02960 * setToolAnalogOutputDomain(index: number, domain: number) -> nil
02961 *
02962 * @par Lua 示例
02963 * setToolAnalogOutputDomain(0,1)
02964 *
02965 * \endchinese
02966 * \english
02967 * Set the range of tool analog output.
02968 *
02969 * @param index: Indicates the IO pin, starting from 0.
02970 * For example, 0 means the first pin.
02971 * @param domain: Output range
02972 *
02973 * @return Returns 0 on success; error code on failure.
02974 * AUBO_REQUEST_IGNORE
02975 * AUBO_BUSY
02976 * AUBO_BAD_STATE
02977 * -AUBO_INVL_ARGUMENT
02978 * -AUBO_BAD_STATE
02979 *
02980 * @throws arcs::common_interface::AuboException
02981 *
02982 * @par Python function prototype
02983 * setToolAnalogOutputDomain(self: pyaubo_sdk.IoControl, arg0: int, arg1:
02984 * int) -> int
02985 *
02986 * @par Lua function prototype
02987 * setToolAnalogOutputDomain(index: number, domain: number) -> nil
02988 *
02989 * @par Lua example
02990 * setToolAnalogOutputDomain(0,1)
02991 *
02992 * \endenglish
02993 */
02994 int setToolAnalogOutputDomain(int index, int domain);
02995
02996 /**
02997 * @ingroup IoControl
02998 * \chinese
02999 * 获取标准模拟输出范围
03000 *
03001 * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
03002 * 例如, 0 表示第一个管脚。
03003 * @return 标准模拟输出范围
03004 *
03005 * @throws arcs::common_interface::AuboException
03006 *
03007 * @par Python 函数原型
03008 * getStandardAnalogOutputDomain(self: pyaubo_sdk.IoControl, arg0: int) ->
03009 * int
03010 *
03011 * @par Lua 函数原型
03012 * getStandardAnalogOutputDomain(index: number) -> number
03013 *
03014 * @par Lua 示例
03015 * num = getStandardAnalogOutputDomain(0)
03016 *
03017 * @par JSON-RPC 请求示例
03018 * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardAnalogOutputDomain", "params": [0], "id": 1}
03019 *
03020 * @par JSON-RPC 响应示例
03021 * {"id": 1, "jsonrpc": "2.0", "result": 0}
03022 *
03023 * \endchinese
03024 * \english
03025 * Get the domain of standard analog output.
03026 *
03027 * @param index: Indicates the IO pin, starting from 0.
03028 * For example, 0 means the first pin.
03029 * @return Standard analog output domain
03030 *
03031 * @throws arcs::common_interface::AuboException
03032 *
03033 * @par Python function prototype
03034 * getStandardAnalogOutputDomain(self: pyaubo_sdk.IoControl, arg0: int) ->

```

```

03035     * int
03036     *
03037     * @par Lua function prototype
03038     * getStandardAnalogOutputDomain(index: number) -> number
03039     *
03040     * @par Lua example
03041     * num = getStandardAnalogOutputDomain(0)
03042     *
03043     * @par JSON-RPC request example
03044     * {"jsonrpc": "2.0", "method": "robo.IoControl.getStandardAnalogOutputDomain", "params": [0], "id": 1}
03045     *
03046     * @par JSON-RPC response example
03047     * {"id": 1, "jsonrpc": "2.0", "result": 0}
03048     *
03049     * \endenglish
03050     */
03051 int getStandardAnalogOutputDomain(int index);
03052
03053 /**
03054  * @ingroup IoControl
03055  * \chinese
03056  * 获取工具端模拟输出范围
03057  *
03058  * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
03059  * 例如, 0 表示第一个管脚。
03060  * @return 工具端模拟输出范围
03061  *
03062  * @throws arcs::common_interface::AuboException
03063  *
03064  * @par Python 函数原型
03065  * getToolAnalogOutputDomain(self: pyaubo_sdk.IoControl, arg0: int) -> int
03066  *
03067  * @par Lua 函数原型
03068  * getToolAnalogOutputDomain(index: number) -> number
03069  *
03070  * @par Lua 示例
03071  * num = getToolAnalogOutputDomain(0)
03072  *
03073  * @par JSON-RPC 请求示例
03074  * {"jsonrpc": "2.0", "method": "robo.IoControl.getToolAnalogOutputDomain", "params": [0], "id": 1}
03075  *
03076  * @par JSON-RPC 响应示例
03077  * {"id": 1, "jsonrpc": "2.0", "result": 0}
03078  *
03079  * \endchinese
03080  * \english
03081  * Get the domain of tool analog output.
03082  *
03083  * @param index: Indicates the IO pin, starting from 0.
03084  * For example, 0 means the first pin.
03085  * @return Tool analog output domain
03086  *
03087  * @throws arcs::common_interface::AuboException
03088  *
03089  * @par Python function prototype
03090  * getToolAnalogOutputDomain(self: pyaubo_sdk.IoControl, arg0: int) -> int
03091  *
03092  * @par Lua function prototype
03093  * getToolAnalogOutputDomain(index: number) -> number
03094  *
03095  * @par Lua example
03096  * num = getToolAnalogOutputDomain(0)
03097  *
03098  * @par JSON-RPC request example
03099  * {"jsonrpc": "2.0", "method": "robo.IoControl.getToolAnalogOutputDomain", "params": [0], "id": 1}
03100  *
03101  * @par JSON-RPC response example
03102  * {"id": 1, "jsonrpc": "2.0", "result": 0}
03103  *
03104  * \endenglish
03105  */
03106 int getToolAnalogOutputDomain(int index);
03107
03108 /**
03109  * @ingroup IoControl
03110  * \chinese
03111  * 设置工具端电源电压值 (单位 V)
03112  *
03113  * @param domain: 工具端电源电压值, 可选三个档位, 分别为 0、12 和 24。 \n
03114  * 0 表示 0V, 12 表示 12V, 24 表示 24V。
03115  *
03116  * @return 成功返回 0; 失败返回错误码
03117  * AUBO_REQUEST_IGNORE
03118  * AUBO_BUSY
03119  * AUBO_BAD_STATE
03120  * -AUBO_INVL_ARGUMENT
03121  * -AUBO_BAD_STATE

```

```

03122     *
03123     * @throws arcs::common_interface::AuboException
03124     *
03125     * @par Python 函数原型
03126     * setToolVoltageOutputDomain(self: pyaubo_sdk.IoControl, arg0: int) -> int
03127     *
03128     * @par Lua 函数原型
03129     * setToolVoltageOutputDomain(domain: number) -> nil
03130     *
03131     * @par Lua 示例
03132     * setToolVoltageOutputDomain(24)
03133     *
03134     * @par JSON-RPC 请求示例
03135     * {"jsonrpc": "2.0", "method": "rob1.IoControl.setToolVoltageOutputDomain", "params": [24], "id": 1}
03136     *
03137     * @par JSON-RPC 响应示例
03138     * {"id": 1, "jsonrpc": "2.0", "result": 0}
03139     *
03140     * \endchinese
03141     * \english
03142     * Set the tool voltage output value (unit: V)
03143     *
03144     * @param domain: Tool voltage output value, can be 0, 12, or 24.\n
03145     *   0 means 0V, 12 means 12V, 24 means 24V.
03146     *
03147     * @return Returns 0 on success; error code on failure.
03148     *   AUBO_REQUEST_IGNORE
03149     *   AUBO_BUSY
03150     *   AUBO_BAD_STATE
03151     *   -AUBO_INVL_ARGUMENT
03152     *   -AUBO_BAD_STATE
03153     *
03154     * @throws arcs::common_interface::AuboException
03155     *
03156     * @par Python function prototype
03157     * setToolVoltageOutputDomain(self: pyaubo_sdk.IoControl, arg0: int) -> int
03158     *
03159     * @par Lua function prototype
03160     * setToolVoltageOutputDomain(domain: number) -> nil
03161     *
03162     * @par Lua example
03163     * setToolVoltageOutputDomain(24)
03164     *
03165     * @par JSON-RPC request example
03166     * {"jsonrpc": "2.0", "method": "rob1.IoControl.setToolVoltageOutputDomain", "params": [24], "id": 1}
03167     *
03168     * @par JSON-RPC response example
03169     * {"id": 1, "jsonrpc": "2.0", "result": 0}
03170     *
03171     * \endenglish
03172     */
03173 int setToolVoltageOutputDomain(int domain);
03174
03175 /**
03176  * @ingroup IoControl
03177  * \chinese
03178  * 获取工具端电源电压值 (单位 V)
03179  *
03180  * @return 工具端电源电压值 (单位 V)
03181  *
03182  * @throws arcs::common_interface::AuboException
03183  *
03184  * @par Python 函数原型
03185  * getToolVoltageOutputDomain(self: pyaubo_sdk.IoControl) -> int
03186  *
03187  * @par Lua 函数原型
03188  * getToolVoltageOutputDomain() -> number
03189  *
03190  * @par Lua 示例
03191  * num = getToolVoltageOutputDomain()
03192  *
03193  * @par JSON-RPC 请求示例
03194  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolVoltageOutputDomain", "params": [], "id": 1}
03195  *
03196  * @par JSON-RPC 响应示例
03197  * {"id": 1, "jsonrpc": "2.0", "result": 0}
03198  *
03199  * \endchinese
03200  * \english
03201  * Get the tool voltage output value (unit: V)
03202  *
03203  * @return Tool voltage output value (unit: V)
03204  *
03205  * @throws arcs::common_interface::AuboException
03206  *
03207  * @par Python function prototype
03208  * getToolVoltageOutputDomain(self: pyaubo_sdk.IoControl) -> int

```

```

03209     *
03210     * @par Lua function prototype
03211     * getToolVoltageOutputDomain() -> number
03212     *
03213     * @par Lua example
03214     * num = getToolVoltageOutputDomain()
03215     *
03216     * @par JSON-RPC request example
03217     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolVoltageOutputDomain", "params": [], "id": 1}
03218     *
03219     * @par JSON-RPC response example
03220     * {"id": 1, "jsonrpc": "2.0", "result": 0}
03221     *
03222     * \endenglish
03223     */
03224 int getToolVoltageOutputDomain();
03225
03226 /**
03227  * @ingroup IoControl
03228  * \chinese
03229  * 设置标准数字输出值
03230  *
03231  * @param index: 表示 IO 口的管脚,
03232  * @param value: 输出值
03233  * @return 成功返回 0; 失败返回错误码
03234  * AUBO_REQUEST_IGNORE
03235  * AUBO_BUSY
03236  * AUBO_BAD_STATE
03237  * -AUBO_INVL_ARGUMENT
03238  * -AUBO_BAD_STATE
03239  *
03240  * @throws arcs::common_interface::AuboException
03241  *
03242  * @par Python 函数原型
03243  * setStandardDigitalOutput(self: pyaubo_sdk.IoControl, arg0: int, arg1:
03244  * bool) -> int
03245  *
03246  * @par Lua 函数原型
03247  * setStandardDigitalOutput(index: number, value: boolean) -> nil
03248  *
03249  * @par Lua 示例
03250  * setStandardDigitalOutput(0,true)
03251  *
03252  * @par JSON-RPC 请求示例
03253  * {"jsonrpc": "2.0", "method": "rob1.IoControl.setStandardDigitalOutput", "params": [0,true], "id": 1}
03254  *
03255  * @par JSON-RPC 响应示例
03256  * {"id": 1, "jsonrpc": "2.0", "result": 0}
03257  *
03258  * \endchinese
03259  * \english
03260  * Set the value of a standard digital output.
03261  *
03262  * @param index: Indicates the IO pin.
03263  * @param value: Output value.
03264  * @return Returns 0 on success; error code on failure.
03265  * AUBO_REQUEST_IGNORE
03266  * AUBO_BUSY
03267  * AUBO_BAD_STATE
03268  * -AUBO_INVL_ARGUMENT
03269  * -AUBO_BAD_STATE
03270  *
03271  * @throws arcs::common_interface::AuboException
03272  *
03273  * @par Python function prototype
03274  * setStandardDigitalOutput(self: pyaubo_sdk.IoControl, arg0: int, arg1:
03275  * bool) -> int
03276  *
03277  * @par Lua function prototype
03278  * setStandardDigitalOutput(index: number, value: boolean) -> nil
03279  *
03280  * @par Lua example
03281  * setStandardDigitalOutput(0,true)
03282  *
03283  * @par JSON-RPC request example
03284  * {"jsonrpc": "2.0", "method": "rob1.IoControl.setStandardDigitalOutput", "params": [0,true], "id": 1}
03285  *
03286  * @par JSON-RPC response example
03287  * {"id": 1, "jsonrpc": "2.0", "result": 0}
03288  *
03289  * \endenglish
03290  */
03291 int setStandardDigitalOutput(int index, bool value);
03292
03293 /**
03294  * @ingroup IoControl
03295  * \chinese

```



```

03296     * 注册组数字输出
03297     *
03298     * 注册一个按名称访问的数字输出组。
03299     * `indexes[0]` 对应 bit0, `indexes[1]` 对应 bit1, 以此类推。
03300     *
03301     * @param name: 组名称。
03302     * @param io_type: IO 类型, 0 表示 Standard, 1 表示 Tool, 2 表示 Configurable。
03303     * @param indexes: 组成员对应的数字输出索引列表。
03304     *
03305     * @return 成功返回 0; 失败返回错误码
03306     *
03307     * @throws arcs::common_interface::AuboException
03308     *
03309     * @par Python 函数原型
03310     * addGroupOutput(self: pyaubo_sdk.IoControl, arg0: str, arg1: int, arg2:
03311     * List[int]) -> int
03312     *
03313     * @par Lua 函数原型
03314     * addGroupOutput(name: string, io_type: number, indexes: table) -> nil
03315     *
03316     * @par Lua 示例
03317     * addGroupOutput("go_state", 0, {4,5,6,7})
03318     *
03319     * @par JSON-RPC 请求示例
03320     * {"jsonrpc": "2.0", "method": "rob1.IoControl.addGroupOutput", "params": ["go_state", 0, [4,5,6,7]], "id": 1}
03321     *
03322     * @par JSON-RPC 响应示例
03323     * {"id": 1, "jsonrpc": "2.0", "result": 0}
03324     *
03325     * \endchinese
03326     * \english
03327     * Register a grouped digital output.
03328     *
03329     * Registers a named digital output group. `indexes[0]` maps to bit0,
03330     * `indexes[1]` maps to bit1, and so on.
03331     *
03332     * @param name: Group name.
03333     * @param io_type: IO type. 0 means Standard, 1 means Tool, and 2 means
03334     * Configurable.
03335     * @param indexes: Index list of digital outputs in the group.
03336     *
03337     * @return Returns 0 on success; error code on failure.
03338     *
03339     * @throws arcs::common_interface::AuboException
03340     *
03341     * @par Python function prototype
03342     * addGroupOutput(self: pyaubo_sdk.IoControl, arg0: str, arg1: int, arg2:
03343     * List[int]) -> int
03344     *
03345     * @par Lua function prototype
03346     * addGroupOutput(name: string, io_type: number, indexes: table) -> nil
03347     *
03348     * @par Lua example
03349     * addGroupOutput("go_state", 0, {4,5,6,7})
03350     *
03351     * @par JSON-RPC request example
03352     * {"jsonrpc": "2.0", "method": "rob1.IoControl.addGroupOutput", "params": ["go_state", 0, [4,5,6,7]], "id": 1}
03353     *
03354     * @par JSON-RPC response example
03355     * {"id": 1, "jsonrpc": "2.0", "result": 0}
03356     *
03357     * \endenglish
03358     */
03359 int addGroupOutput(const std::string &name, int io_type,
03360                   const std::vector<int> &indexes);
03361
03362 /**
03363     * @ingroup IoControl
03364     * \chinese
03365     * 删除组数字输出
03366     *
03367     * @param name: 组名称。
03368     *
03369     * @return 成功返回 0; 失败返回错误码
03370     *
03371     * @throws arcs::common_interface::AuboException
03372     *
03373     * @par Python 函数原型
03374     * deleteGroupOutput(self: pyaubo_sdk.IoControl, arg0: str) -> int
03375     *
03376     * @par Lua 函数原型
03377     * deleteGroupOutput(name: string) -> nil
03378     *
03379     * @par JSON-RPC 请求示例
03380     * {"jsonrpc": "2.0", "method": "rob1.IoControl.deleteGroupOutput", "params": ["go_state"], "id": 1}
03381     *
03382     * @par JSON-RPC 响应示例

```

```

03383     * {"id":1,"jsonrpc":"2.0","result":0}
03384     *
03385     * \endchinese
03386     * \english
03387     * Delete a grouped digital output.
03388     *
03389     * @param name: Group name.
03390     *
03391     * @return Returns 0 on success; error code on failure.
03392     *
03393     * @throws arcs::common_interface::AuboException
03394     *
03395     * @par Python function prototype
03396     * deleteGroupOutput(self: pyaubo_sdk.IoControl, arg0: str) -> int
03397     *
03398     * @par Lua function prototype
03399     * deleteGroupOutput(name: string) -> nil
03400     *
03401     * @par JSON-RPC request example
03402     * {"jsonrpc":"2.0","method":"rob1.IoControl.deleteGroupOutput","params":["go_state"],"id":1}
03403     *
03404     * @par JSON-RPC response example
03405     * {"id":1,"jsonrpc":"2.0","result":0}
03406     *
03407     * \endenglish
03408     */
03409 int deleteGroupOutput(const std::string &name);
03410
03411 /**
03412  * @ingroup IoControl
03413  * \chinese
03414  * 设置组数字输出值
03415  *
03416  * 按名称查找组数字输出后, 将 `value` 按位拆分写入对应组成员。
03417  *
03418  * @param name: 组名称。
03419  * @param value: 需要写入组数字输出的整数值。
03420  *
03421  * @return 成功返回 0; 失败返回错误码
03422  * AUBO_REQUEST_IGNORE
03423  * AUBO_BUSY
03424  * AUBO_BAD_STATE
03425  * -AUBO_INVL_ARGUMENT
03426  * -AUBO_BAD_STATE
03427  *
03428  * @throws arcs::common_interface::AuboException
03429  *
03430  * @par Python 函数原型
03431  * setGroupOutput(self: pyaubo_sdk.IoControl, arg0: str, arg1: int) -> int
03432  *
03433  * @par Lua 函数原型
03434  * setGroupOutput(name: string, value: number) -> nil
03435  *
03436  * @par Lua 示例
03437  * setGroupOutput("go_state", 13)
03438  *
03439  * @par JSON-RPC 请求示例
03440  * {"jsonrpc":"2.0","method":"rob1.IoControl.setGroupOutput","params":["go_state",13],"id":1}
03441  *
03442  * @par JSON-RPC 响应示例
03443  * {"id":1,"jsonrpc":"2.0","result":0}
03444  *
03445  * \endchinese
03446  * \english
03447  * Set the grouped digital output value.
03448  *
03449  * Looks up the named grouped digital output and writes `value` to the
03450  * group's members bit by bit.
03451  *
03452  * @param name: Group name.
03453  * @param value: Integer value to write to the grouped digital output.
03454  *
03455  * @return Returns 0 on success; error code on failure.
03456  * AUBO_REQUEST_IGNORE
03457  * AUBO_BUSY
03458  * AUBO_BAD_STATE
03459  * -AUBO_INVL_ARGUMENT
03460  * -AUBO_BAD_STATE
03461  *
03462  * @throws arcs::common_interface::AuboException
03463  *
03464  * @par Python function prototype
03465  * setGroupOutput(self: pyaubo_sdk.IoControl, arg0: str, arg1: int) -> int
03466  *
03467  * @par Lua function prototype
03468  * setGroupOutput(name: string, value: number) -> nil
03469  *

```

```

03470     * @par Lua example
03471     * setGroupOutput("go_state", 13)
03472     *
03473     * @par JSON-RPC request example
03474     * {"jsonrpc": "2.0", "method": "rob1.IoControl.setGroupOutput", "params": ["go_state", 13], "id": 1}
03475     *
03476     * @par JSON-RPC response example
03477     * {"id": 1, "jsonrpc": "2.0", "result": 0}
03478     *
03479     * \endenglish
03480     */
03481 int setGroupOutput(const std::string &name, uint32_t value);
03482
03483 /**
03484     * @ingroup IoControl
03485     * \chinese
03486     * 设置数字输出脉冲
03487     *
03488     * @param index
03489     * @param value
03490     * @param duration
03491     *
03492     * @return 成功返回 0; 失败返回错误码
03493     * AUBO_REQUEST_IGNORE
03494     * AUBO_BUSY
03495     * AUBO_BAD_STATE
03496     * -AUBO_INVL_ARGUMENT
03497     * -AUBO_BAD_STATE
03498     *
03499     * @throws arcs::common_interface::AuboException
03500     *
03501     * @par Python 函数原型
03502     * setStandardDigitalOutputPulse(self: pyaubo_sdk.IoControl, arg0: int,
03503     * arg1: bool, arg2: float) -> int
03504     *
03505     * @par Lua 函数原型
03506     * setStandardDigitalOutputPulse(index: number, value: boolean, duration:
03507     * number) -> nil
03508     *
03509     * @par Lua 示例
03510     * setStandardDigitalOutputPulse(0,true,2.5)
03511     *
03512     * @par JSON-RPC 请求示例
03513     * {"jsonrpc": "2.0", "method": "rob1.IoControl.setStandardDigitalOutputPulse", "params": [0,true,0.5], "id": 1}
03514     *
03515     * @par JSON-RPC 响应示例
03516     * {"id": 1, "jsonrpc": "2.0", "result": 0}
03517     *
03518     * \endchinese
03519     * \english
03520     * Set digital output pulse.
03521     *
03522     * @param index
03523     * @param value
03524     * @param duration
03525     *
03526     * @return Returns 0 on success; error code on failure.
03527     * AUBO_REQUEST_IGNORE
03528     * AUBO_BUSY
03529     * AUBO_BAD_STATE
03530     * -AUBO_INVL_ARGUMENT
03531     * -AUBO_BAD_STATE
03532     *
03533     * @throws arcs::common_interface::AuboException
03534     *
03535     * @par Python function prototype
03536     * setStandardDigitalOutputPulse(self: pyaubo_sdk.IoControl, arg0: int,
03537     * arg1: bool, arg2: float) -> int
03538     *
03539     * @par Lua function prototype
03540     * setStandardDigitalOutputPulse(index: number, value: boolean, duration:
03541     * number) -> nil
03542     *
03543     * @par Lua example
03544     * setStandardDigitalOutputPulse(0,true,2.5)
03545     *
03546     * @par JSON-RPC request example
03547     * {"jsonrpc": "2.0", "method": "rob1.IoControl.setStandardDigitalOutputPulse", "params": [0,true,0.5], "id": 1}
03548     *
03549     * @par JSON-RPC response example
03550     * {"id": 1, "jsonrpc": "2.0", "result": 0}
03551     *
03552     * \endenglish
03553     */
03554 int setStandardDigitalOutputPulse(int index, bool value, double duration);
03555
03556 /**

```

```

03557     * @ingroup IoControl
03558     * \chinese
03559     * 设置工具端数字输出值
03560     *
03561     * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
03562     * 例如, 0 表示第一个管脚。
03563     * @param value: 数字输出值
03564     *
03565     * @return 成功返回 0; 失败返回错误码
03566     * AUBO_REQUEST_IGNORE
03567     * AUBO_BUSY
03568     * AUBO_BAD_STATE
03569     * -AUBO_INVL_ARGUMENT
03570     * -AUBO_BAD_STATE
03571     *
03572     * @throws arcs::common_interface::AuboException
03573     *
03574     * @par Python 函数原型
03575     * setToolDigitalOutput(self: pyaubo_sdk.IoControl, arg0: int, arg1: bool)
03576     * -> int
03577     *
03578     * @par Lua 函数原型
03579     * setToolDigitalOutput(index: number, value: boolean) -> nil
03580     *
03581     * @par Lua 示例
03582     * setToolDigitalOutput(0,true)
03583     *
03584     * @par JSON-RPC 请求示例
03585     * {"jsonrpc":"2.0","method":"robo.IoControl.setToolDigitalOutput","params":[0,true],"id":1}
03586     *
03587     * @par JSON-RPC 响应示例
03588     * {"id":1,"jsonrpc":"2.0","result":0}
03589     *
03590     * \endchinese
03591     * \english
03592     * Set the value of tool digital output.
03593     *
03594     * @param index: Indicates the IO pin, starting from 0.
03595     * For example, 0 means the first pin.
03596     * @param value: Output value.
03597     *
03598     * @return Returns 0 on success; error code on failure.
03599     * AUBO_REQUEST_IGNORE
03600     * AUBO_BUSY
03601     * AUBO_BAD_STATE
03602     * -AUBO_INVL_ARGUMENT
03603     * -AUBO_BAD_STATE
03604     *
03605     * @throws arcs::common_interface::AuboException
03606     *
03607     * @par Python function prototype
03608     * setToolDigitalOutput(self: pyaubo_sdk.IoControl, arg0: int, arg1: bool)
03609     * -> int
03610     *
03611     * @par Lua function prototype
03612     * setToolDigitalOutput(index: number, value: boolean) -> nil
03613     *
03614     * @par Lua example
03615     * setToolDigitalOutput(0,true)
03616     *
03617     * @par JSON-RPC request example
03618     * {"jsonrpc":"2.0","method":"robo.IoControl.setToolDigitalOutput","params":[0,true],"id":1}
03619     *
03620     * @par JSON-RPC response example
03621     * {"id":1,"jsonrpc":"2.0","result":0}
03622     *
03623     * \endenglish
03624     */
03625 int setToolDigitalOutput(int index, bool value);
03626
03627 /**
03628     * @ingroup IoControl
03629     * \chinese
03630     * 设置工具端数字输出脉冲
03631     *
03632     * @param index
03633     * @param value
03634     * @param duration
03635     *
03636     * @return 成功返回 0; 失败返回错误码
03637     * AUBO_REQUEST_IGNORE
03638     * AUBO_BUSY
03639     * AUBO_BAD_STATE
03640     * -AUBO_INVL_ARGUMENT
03641     * -AUBO_BAD_STATE
03642     *
03643     * @throws arcs::common_interface::AuboException

```

```

03644 *
03645 * @par Python 函数原型
03646 * setToolDigitalOutputPulse(self: pyaubo_sdk.IoControl, arg0: int, arg1:
03647 * bool, arg2: float) -> int
03648 *
03649 * @par Lua 函数原型
03650 * setToolDigitalOutputPulse(index: number, value: boolean, duration:
03651 * number) -> nil
03652 *
03653 * @par Lua 示例
03654 * setToolDigitalOutputPulse(0,true,2)
03655 *
03656 * @par JSON-RPC 请求示例
03657 * {"jsonrpc":"2.0","method":"rob1.IoControl.setToolDigitalOutputPulse","params":[0,true,0.5],"id":1}
03658 *
03659 * @par JSON-RPC 响应示例
03660 * {"id":1,"jsonrpc":"2.0","result":0}
03661 *
03662 * \endchinese
03663 * \english
03664 * Set tool digital output pulse.
03665 *
03666 * @param index
03667 * @param value
03668 * @param duration
03669 *
03670 * @return Returns 0 on success; error code on failure.
03671 * AUBO_REQUEST_IGNORE
03672 * AUBO_BUSY
03673 * AUBO_BAD_STATE
03674 * -AUBO_INVL_ARGUMENT
03675 * -AUBO_BAD_STATE
03676 *
03677 * @throws arcs::common_interface::AuboException
03678 *
03679 * @par Python function prototype
03680 * setToolDigitalOutputPulse(self: pyaubo_sdk.IoControl, arg0: int, arg1:
03681 * bool, arg2: float) -> int
03682 *
03683 * @par Lua function prototype
03684 * setToolDigitalOutputPulse(index: number, value: boolean, duration:
03685 * number) -> nil
03686 *
03687 * @par Lua example
03688 * setToolDigitalOutputPulse(0,true,2)
03689 *
03690 * @par JSON-RPC request example
03691 * {"jsonrpc":"2.0","method":"rob1.IoControl.setToolDigitalOutputPulse","params":[0,true,0.5],"id":1}
03692 *
03693 * @par JSON-RPC response example
03694 * {"id":1,"jsonrpc":"2.0","result":0}
03695 *
03696 * \endenglish
03697 */
03698 int setToolDigitalOutputPulse(int index, bool value, double duration);
03699
03700 /**
03701 * @ingroup IoControl
03702 * \chinese
03703 * 设置可配置数字输出值
03704 *
03705 * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
03706 * 例如, 0 表示第一个管脚。
03707 * @param value: 数字输出值
03708 *
03709 * @return 成功返回 0; 失败返回错误码
03710 * AUBO_REQUEST_IGNORE
03711 * AUBO_BUSY
03712 * AUBO_BAD_STATE
03713 * -AUBO_INVL_ARGUMENT
03714 * -AUBO_BAD_STATE
03715 *
03716 * @throws arcs::common_interface::AuboException
03717 *
03718 * @par Python 函数原型
03719 * setConfigurableDigitalOutput(self: pyaubo_sdk.IoControl, arg0: int, arg1:
03720 * bool) -> int
03721 *
03722 * @par Lua 函数原型
03723 * setConfigurableDigitalOutput(index: number, value: boolean) -> nil
03724 *
03725 * @par Lua 示例
03726 * setConfigurableDigitalOutput(0,true)
03727 *
03728 * @par JSON-RPC 请求示例
03729 * {"jsonrpc":"2.0","method":"rob1.IoControl.setConfigurableDigitalOutput","params":[0,true],"id":1}
03730 *

```

```

03731     * @par JSON-RPC 响应示例
03732     * {"id":1,"jsonrpc":"2.0","result":0}
03733     *
03734     * \endchinese
03735     * \english
03736     * Set the value of configurable digital output.
03737     *
03738     * @param index: Indicates the IO pin, starting from 0.
03739     * For example, 0 means the first pin.
03740     * @param value: Output value.
03741     *
03742     * @return Returns 0 on success; error code on failure.
03743     * AUBO_REQUEST_IGNORE
03744     * AUBO_BUSY
03745     * AUBO_BAD_STATE
03746     * -AUBO_INVL_ARGUMENT
03747     * -AUBO_BAD_STATE
03748     *
03749     * @throws arcs::common_interface::AuboException
03750     *
03751     * @par Python function prototype
03752     * setConfigurableDigitalOutput(self: pyaubo_sdk.IoControl, arg0: int, arg1:
03753     * bool) -> int
03754     *
03755     * @par Lua function prototype
03756     * setConfigurableDigitalOutput(index: number, value: boolean) -> nil
03757     *
03758     * @par Lua example
03759     * setConfigurableDigitalOutput(0,true)
03760     *
03761     * @par JSON-RPC request example
03762     * {"jsonrpc":"2.0","method":"rob1.IoControl.setConfigurableDigitalOutput","params":[0,true],"id":1}
03763     *
03764     * @par JSON-RPC response example
03765     * {"id":1,"jsonrpc":"2.0","result":0}
03766     *
03767     * \endenglish
03768     */
03769 int setConfigurableDigitalOutput(int index, bool value);
03770
03771 /**
03772  * @ingroup IoControl
03773  * \chinese
03774  * 设置可配置数字输出脉冲
03775  *
03776  * @param index
03777  * @param value
03778  * @param duration
03779  *
03780  * @return 成功返回 0; 失败返回错误码
03781  * AUBO_REQUEST_IGNORE
03782  * AUBO_BUSY
03783  * AUBO_BAD_STATE
03784  * -AUBO_INVL_ARGUMENT
03785  * -AUBO_BAD_STATE
03786  *
03787  * @throws arcs::common_interface::AuboException
03788  *
03789  * @par Python 函数原型
03790  * setConfigurableDigitalOutputPulse(self: pyaubo_sdk.IoControl, arg0: int,
03791  * arg1: bool, arg2: float) -> int
03792  *
03793  * @par Lua 函数原型
03794  * setConfigurableDigitalOutputPulse(index: number, value: boolean,
03795  * duration: number) -> nil
03796  *
03797  * @par Lua 示例
03798  * setConfigurableDigitalOutputPulse(0,true,2.3)
03799  *
03800  * @par JSON-RPC 请求示例
03801  * {"jsonrpc":"2.0","method":"rob1.IoControl.setConfigurableDigitalOutputPulse","params":[0,true,0.5],"id":1}
03802  *
03803  * @par JSON-RPC 响应示例
03804  * {"id":1,"jsonrpc":"2.0","result":0}
03805  *
03806  * \endchinese
03807  * \english
03808  * Set configurable digital output pulse.
03809  *
03810  * @param index
03811  * @param value
03812  * @param duration
03813  *
03814  * @return Returns 0 on success; error code on failure.
03815  * AUBO_REQUEST_IGNORE
03816  * AUBO_BUSY
03817  * AUBO_BAD_STATE

```

```

03818     * -AUBO_INVL_ARGUMENT
03819     * -AUBO_BAD_STATE
03820     *
03821     * @throws arcs::common_interface::AuboException
03822     *
03823     * @par Python function prototype
03824     * setConfigurableDigitalOutputPulse(self: pyaubo_sdk.IoControl, arg0: int,
03825     * arg1: bool, arg2: float) -> int
03826     *
03827     * @par Lua function prototype
03828     * setConfigurableDigitalOutputPulse(index: number, value: boolean,
03829     * duration: number) -> nil
03830     *
03831     * @par Lua example
03832     * setConfigurableDigitalOutputPulse(0,true,2.3)
03833     *
03834     * @par JSON-RPC request example
03835     * {"jsonrpc":"2.0","method":"rob1.IoControl.setConfigurableDigitalOutputPulse","params":[0,true,0.5],"id":1}
03836     *
03837     * @par JSON-RPC response example
03838     * {"id":1,"jsonrpc":"2.0","result":0}
03839     *
03840     * \endenglish
03841     */
03842 int setConfigurableDigitalOutputPulse(int index, bool value,
03843                                     double duration);
03844
03845 /**
03846  * @ingroup IoControl
03847  * \chinese
03848  * 设置标准模拟输出值
03849  *
03850  * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
03851  * 例如, 0 表示第一个管脚。
03852  * @param value: 模拟输出值
03853  *
03854  * @return 成功返回 0; 失败返回错误码
03855  * AUBO_REQUEST_IGNORE
03856  * AUBO_BUSY
03857  * AUBO_BAD_STATE
03858  * -AUBO_INVL_ARGUMENT
03859  * -AUBO_BAD_STATE
03860  *
03861  * @throws arcs::common_interface::AuboException
03862  *
03863  * @par Python 函数原型
03864  * setStandardAnalogOutput(self: pyaubo_sdk.IoControl, arg0: int, arg1:
03865  * float) -> int
03866  *
03867  * @par Lua 函数原型
03868  * setStandardAnalogOutput(index: number, value: number) -> nil
03869  *
03870  * @par Lua 示例
03871  * setStandardAnalogOutput(0,5.4)
03872  *
03873  * \endchinese
03874  * \english
03875  * Set the value of standard analog output.
03876  *
03877  * @param index: Indicates the IO pin, starting from 0.
03878  * For example, 0 means the first pin.
03879  * @param value: Output value.
03880  *
03881  * @return Returns 0 on success; error code on failure.
03882  * AUBO_REQUEST_IGNORE
03883  * AUBO_BUSY
03884  * AUBO_BAD_STATE
03885  * -AUBO_INVL_ARGUMENT
03886  * -AUBO_BAD_STATE
03887  *
03888  * @throws arcs::common_interface::AuboException
03889  *
03890  * @par Python function prototype
03891  * setStandardAnalogOutput(self: pyaubo_sdk.IoControl, arg0: int, arg1:
03892  * float) -> int
03893  *
03894  * @par Lua function prototype
03895  * setStandardAnalogOutput(index: number, value: number) -> nil
03896  *
03897  * @par Lua example
03898  * setStandardAnalogOutput(0,5.4)
03899  *
03900  * \endenglish
03901  */
03902 int setStandardAnalogOutput(int index, double value);
03903
03904 /**

```

```

03905 * @ingroup IoControl
03906 * \chinese
03907 * 设置工具端模拟输出值
03908 *
03909 * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
03910 * 例如, 0 表示第一个管脚。
03911 * @param value: 模拟输出
03912 *
03913 * @return 成功返回 0; 失败返回错误码
03914 * AUBO_REQUEST_IGNORE
03915 * AUBO_BUSY
03916 * AUBO_BAD_STATE
03917 * -AUBO_INVL_ARGUMENT
03918 * -AUBO_BAD_STATE
03919 *
03920 * @throws arcs::common_interface::AuboException
03921 *
03922 * @par Python 函数原型
03923 * setToolAnalogOutput(self: pyaubo_sdk.IoControl, arg0: int, arg1: float)
03924 * -> int
03925 *
03926 * @par Lua 函数原型
03927 * setToolAnalogOutput(index: number, value: number) -> nil
03928 *
03929 * @par Lua 示例
03930 * setToolAnalogOutput(0,1.2)
03931 *
03932 * @par JSON-RPC 请求示例
03933 * {"jsonrpc":"2.0","method":"robo.IoControl.setToolAnalogOutput","params":[0,0.5],"id":1}
03934 *
03935 * @par JSON-RPC 响应示例
03936 * {"id":1,"jsonrpc":"2.0","result":13}
03937 *
03938 * \endchinese
03939 * \english
03940 * Set the value of tool analog output.
03941 *
03942 * @param index: Indicates the IO pin, starting from 0.
03943 * For example, 0 means the first pin.
03944 * @param value: Output value.
03945 *
03946 * @return Returns 0 on success; error code on failure.
03947 * AUBO_REQUEST_IGNORE
03948 * AUBO_BUSY
03949 * AUBO_BAD_STATE
03950 * -AUBO_INVL_ARGUMENT
03951 * -AUBO_BAD_STATE
03952 *
03953 * @throws arcs::common_interface::AuboException
03954 *
03955 * @par Python function prototype
03956 * setToolAnalogOutput(self: pyaubo_sdk.IoControl, arg0: int, arg1: float)
03957 * -> int
03958 *
03959 * @par Lua function prototype
03960 * setToolAnalogOutput(index: number, value: number) -> nil
03961 *
03962 * @par Lua example
03963 * setToolAnalogOutput(0,1.2)
03964 *
03965 * @par JSON-RPC request example
03966 * {"jsonrpc":"2.0","method":"robo.IoControl.setToolAnalogOutput","params":[0,0.5],"id":1}
03967 *
03968 * @par JSON-RPC response example
03969 * {"id":1,"jsonrpc":"2.0","result":13}
03970 *
03971 * \endenglish
03972 */
03973 int setToolAnalogOutput(int index, double value);
03974
03975 /**
03976 * @ingroup IoControl
03977 * \chinese
03978 * 获取标准数字输入值
03979 *
03980 * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
03981 * 例如, 0 表示第一个管脚。
03982 *
03983 * @return 高电平返回 true; 低电平返回 false
03984 *
03985 * @throws arcs::common_interface::AuboException
03986 *
03987 * @par Python 函数原型
03988 * getStandardDigitalInput(self: pyaubo_sdk.IoControl, arg0: int) -> bool
03989 *
03990 * @par Lua 函数原型
03991 * getStandardDigitalInput(index: number) -> boolean

```



```

03992      *
03993      * @par Lua 示例
03994      * status = getStandardDigitalInput(0)
03995      *
03996      * @par JSON-RPC 请求示例
03997      * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardDigitalInput", "params": [0], "id": 1}
03998      *
03999      * @par JSON-RPC 响应示例
04000      * {"id": 1, "jsonrpc": "2.0", "result": false}
04001      *
04002      * \endchinese
04003      * \english
04004      * Get the value of a standard digital input.
04005      *
04006      * @param index: Indicates the IO pin, starting from 0.
04007      * For example, 0 means the first pin.
04008      *
04009      * @return Returns true for high level; false for low level.
04010      *
04011      * @throws arcs::common_interface::AuboException
04012      *
04013      * @par Python function prototype
04014      * getStandardDigitalInput(self: pyaubo_sdk.IoControl, arg0: int) -> bool
04015      *
04016      * @par Lua function prototype
04017      * getStandardDigitalInput(index: number) -> boolean
04018      *
04019      * @par Lua example
04020      * status = getStandardDigitalInput(0)
04021      *
04022      * @par JSON-RPC request example
04023      * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardDigitalInput", "params": [0], "id": 1}
04024      *
04025      * @par JSON-RPC response example
04026      * {"id": 1, "jsonrpc": "2.0", "result": false}
04027      *
04028      * \endenglish
04029      */
04030 bool getStandardDigitalInput(int index);
04031
04032 /**
04033  * @ingroup IoControl
04034  * \chinese
04035  * 获取所有的标准数字输入值
04036  *
04037  * @return 所有的标准数字输入值 \n
04038  * 例如, 当返回值是 2863267846 时, 换成 2 进制后是 10101010101010000000000000110。
04039  * 后 16 位就是所有的标准数字输入状态值,
04040  * 最后一位表示 DI00 的输入状态值, 倒数第二位表示 DI01 的输入状态值, 以此类推。 \n
04041  * 1 表示高电平状态, 0 表示低电平状态
04042  *
04043  * @throws arcs::common_interface::AuboException
04044  *
04045  * @par Python 函数原型
04046  * getStandardDigitalInputs(self: pyaubo_sdk.IoControl) -> int
04047  *
04048  * @par Lua 函数原型
04049  * getStandardDigitalInputs() -> number
04050  *
04051  * @par Lua 示例
04052  * num = getStandardDigitalInputs()
04053  *
04054  * @par JSON-RPC 请求示例
04055  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardDigitalInputs", "params": [], "id": 1}
04056  *
04057  * @par JSON-RPC 响应示例
04058  * {"id": 1, "jsonrpc": "2.0", "result": 0}
04059  *
04060  * \endchinese
04061  * \english
04062  * Get all standard digital input values.
04063  *
04064  * @return All standard digital input values. \n
04065  * For example, if the return value is 2863267846, its binary representation
04066  * is 101010101010101000000000000000110. The lower 16 bits represent the
04067  * status of all standard digital inputs, the least significant bit
04068  * indicates the input status of DI00, the second least significant bit
04069  * indicates DI01, and so on. \n 1 means high level, 0 means low level.
04070  *
04071  * @throws arcs::common_interface::AuboException
04072  *
04073  * @par Python function prototype
04074  * getStandardDigitalInputs(self: pyaubo_sdk.IoControl) -> int
04075  *
04076  * @par Lua function prototype
04077  * getStandardDigitalInputs() -> number
04078  *

```

```

04079     * @par Lua example
04080     * num = getStandardDigitalInputs()
04081     *
04082     * @par JSON-RPC request example
04083     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardDigitalInputs", "params": [], "id": 1}
04084     *
04085     * @par JSON-RPC response example
04086     * {"id": 1, "jsonrpc": "2.0", "result": 0}
04087     *
04088     * \endenglish
04089     */
04090 uint32_t getStandardDigitalInputs();
04091
04092 /**
04093     * @ingroup IoControl
04094     * \chinese
04095     * 注册组数字输入
04096     *
04097     * 注册一个按名称访问的数字输入组。
04098     * `indexes[0]` 对应 bit0, `indexes[1]` 对应 bit1, 以此类推。
04099     *
04100     * @param name: 组名称。
04101     * @param io_type: IO 类型, 0 表示 Standard, 1 表示 Tool, 2 表示 Configurable。
04102     * @param indexes: 组成员对应的数字输入索引列表。
04103     *
04104     * @return 成功返回 0; 失败返回错误码
04105     *
04106     * @throws arcs::common_interface::AuboException
04107     *
04108     * @par Python 函数原型
04109     * addGroupInput(self: pyaubo_sdk.IoControl, arg0: str, arg1: int, arg2:
04110     * List[int]) -> int
04111     *
04112     * @par Lua 函数原型
04113     * addGroupInput(name: string, io_type: number, indexes: table) -> nil
04114     *
04115     * @par Lua 示例
04116     * addGroupInput("gi_state", 0, {0,1,2,3})
04117     *
04118     * @par JSON-RPC 请求示例
04119     * {"jsonrpc": "2.0", "method": "rob1.IoControl.addGroupInput", "params": ["gi_state", 0, [0,1,2,3]], "id": 1}
04120     *
04121     * @par JSON-RPC 响应示例
04122     * {"id": 1, "jsonrpc": "2.0", "result": 0}
04123     *
04124     * \endchinese
04125     * \english
04126     * Register a grouped digital input.
04127     *
04128     * Registers a named digital input group. `indexes[0]` maps to bit0,
04129     * `indexes[1]` maps to bit1, and so on.
04130     *
04131     * @param name: Group name.
04132     * @param io_type: IO type. 0 means Standard, 1 means Tool, and 2 means
04133     * Configurable.
04134     * @param indexes: Index list of digital inputs in the group.
04135     *
04136     * @return Returns 0 on success; error code on failure.
04137     *
04138     * @throws arcs::common_interface::AuboException
04139     *
04140     * @par Python function prototype
04141     * addGroupInput(self: pyaubo_sdk.IoControl, arg0: str, arg1: int, arg2:
04142     * List[int]) -> int
04143     *
04144     * @par Lua function prototype
04145     * addGroupInput(name: string, io_type: number, indexes: table) -> nil
04146     *
04147     * @par Lua example
04148     * addGroupInput("gi_state", 0, {0,1,2,3})
04149     *
04150     * @par JSON-RPC request example
04151     * {"jsonrpc": "2.0", "method": "rob1.IoControl.addGroupInput", "params": ["gi_state", 0, [0,1,2,3]], "id": 1}
04152     *
04153     * @par JSON-RPC response example
04154     * {"id": 1, "jsonrpc": "2.0", "result": 0}
04155     *
04156     * \endenglish
04157     */
04158 int addGroupInput(const std::string &name, int io_type,
04159                  const std::vector<int> &indexes);
04160
04161 /**
04162     * @ingroup IoControl
04163     * \chinese
04164     * 删除组数字输入
04165     *

```

```

04166     * @param name: 组名称。
04167     *
04168     * @return 成功返回 0；失败返回错误码
04169     *
04170     * @throws arcs::common_interface::AuboException
04171     *
04172     * @par Python 函数原型
04173     * deleteGroupInput(self: pyaubo_sdk.IoControl, arg0: str) -> int
04174     *
04175     * @par Lua 函数原型
04176     * deleteGroupInput(name: string) -> nil
04177     *
04178     * @par JSON-RPC 请求示例
04179     * {"jsonrpc": "2.0", "method": "rob1.IoControl.deleteGroupInput", "params": ["gi_state"], "id": 1}
04180     *
04181     * @par JSON-RPC 响应示例
04182     * {"id": 1, "jsonrpc": "2.0", "result": 0}
04183     *
04184     * \endchinese
04185     * \english
04186     * Delete a grouped digital input.
04187     *
04188     * @param name: Group name.
04189     *
04190     * @return Returns 0 on success; error code on failure.
04191     *
04192     * @throws arcs::common_interface::AuboException
04193     *
04194     * @par Python function prototype
04195     * deleteGroupInput(self: pyaubo_sdk.IoControl, arg0: str) -> int
04196     *
04197     * @par Lua function prototype
04198     * deleteGroupInput(name: string) -> nil
04199     *
04200     * @par JSON-RPC request example
04201     * {"jsonrpc": "2.0", "method": "rob1.IoControl.deleteGroupInput", "params": ["gi_state"], "id": 1}
04202     *
04203     * @par JSON-RPC response example
04204     * {"id": 1, "jsonrpc": "2.0", "result": 0}
04205     *
04206     * \endenglish
04207     */
04208 int deleteGroupInput(const std::string &name);
04209
04210 /**
04211  * @ingroup IoControl
04212  * \chinese
04213  * 获取组数字输入值
04214  *
04215  * 按名称查找组数字输入后，读取对应组成员并按位拼装返回整数。
04216  *
04217  * @param name: 组名称。
04218  *
04219  * @return 返回组数字输入的组合值。
04220  *
04221  * @throws arcs::common_interface::AuboException
04222  *
04223  * @par Python 函数原型
04224  * getGroupInput(self: pyaubo_sdk.IoControl, arg0: str) -> int
04225  *
04226  * @par Lua 函数原型
04227  * getGroupInput(name: string) -> number
04228  *
04229  * @par Lua 示例
04230  * value = getGroupInput("gi_state")
04231  *
04232  * @par JSON-RPC 请求示例
04233  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getGroupInput", "params": ["gi_state"], "id": 1}
04234  *
04235  * @par JSON-RPC 响应示例
04236  * {"id": 1, "jsonrpc": "2.0", "result": 13}
04237  *
04238  * \endchinese
04239  * \english
04240  * Get the grouped digital input value.
04241  *
04242  * Looks up the named grouped digital input, reads the group's members, and
04243  * combines them into an integer value.
04244  *
04245  * @param name: Group name.
04246  *
04247  * @return Returns the combined grouped digital input value.
04248  *
04249  * @throws arcs::common_interface::AuboException
04250  *
04251  * @par Python function prototype
04252  * getGroupInput(self: pyaubo_sdk.IoControl, arg0: str) -> int

```

```

04253     *
04254     * @par Lua function prototype
04255     * getGroupInput(name: string) -> number
04256     *
04257     * @par Lua example
04258     * value = getGroupInput("gi_state")
04259     *
04260     * @par JSON-RPC request example
04261     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getGroupInput", "params": ["gi_state"], "id": 1}
04262     *
04263     * @par JSON-RPC response example
04264     * {"id": 1, "jsonrpc": "2.0", "result": 13}
04265     *
04266     * \endenglish
04267     */
04268 uint32_t getGroupInput(const std::string &name);
04269
04270 /**
04271  * @ingroup IoControl
04272  * \chinese
04273  * 获取工具端数字输入值
04274  *
04275  * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
04276  * 例如, 0 表示第一个管脚。
04277  *
04278  * @return 高电平返回 true; 低电平返回 false
04279  *
04280  * @throws arcs::common_interface::AuboException
04281  *
04282  * @par Python 函数原型
04283  * getToolDigitalInput(self: pyaubo_sdk.IoControl, arg0: int) -> bool
04284  *
04285  * @par Lua 函数原型
04286  * getToolDigitalInput(index: number) -> boolean
04287  *
04288  * @par Lua 示例
04289  * status = getToolDigitalInput(0)
04290  *
04291  * @par JSON-RPC 请求示例
04292  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolDigitalInput", "params": [0], "id": 1}
04293  *
04294  * @par JSON-RPC 响应示例
04295  * {"id": 1, "jsonrpc": "2.0", "result": false}
04296  *
04297  * \endchinese
04298  * \english
04299  * Get the value of tool digital input.
04300  *
04301  * @param index: Indicates the IO pin, starting from 0.
04302  * For example, 0 means the first pin.
04303  *
04304  * @return Returns true for high level; false for low level.
04305  *
04306  * @throws arcs::common_interface::AuboException
04307  *
04308  * @par Python function prototype
04309  * getToolDigitalInput(self: pyaubo_sdk.IoControl, arg0: int) -> bool
04310  *
04311  * @par Lua function prototype
04312  * getToolDigitalInput(index: number) -> boolean
04313  *
04314  * @par Lua example
04315  * status = getToolDigitalInput(0)
04316  *
04317  * @par JSON-RPC request example
04318  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolDigitalInput", "params": [0], "id": 1}
04319  *
04320  * @par JSON-RPC response example
04321  * {"id": 1, "jsonrpc": "2.0", "result": false}
04322  *
04323  * \endenglish
04324  */
04325 bool getToolDigitalInput(int index);
04326
04327 /**
04328  * @ingroup IoControl
04329  * \chinese
04330  * 获取所有的工具端数字输入值
04331  *
04332  * @return 返回所有的工具端数字输入值
04333  *
04334  * @throws arcs::common_interface::AuboException
04335  *
04336  * @par Python 函数原型
04337  * getToolDigitalInputs(self: pyaubo_sdk.IoControl) -> int
04338  *
04339  * @par Lua 函数原型

```

```

04340     * getToolDigitalInputs() -> number
04341     *
04342     * @par Lua 示例
04343     * num = getToolDigitalInputs()
04344     *
04345     * @par JSON-RPC 请求示例
04346     * {"jsonrpc":"2.0","method":"rob1.IoControl.getToolDigitalInputs","params":[],"id":1}
04347     *
04348     * @par JSON-RPC 响应示例
04349     * {"id":1,"jsonrpc":"2.0","result":0}
04350     *
04351     * \endchinese
04352     * \english
04353     * Get all tool digital input values.
04354     *
04355     * @return Returns all tool digital input values.
04356     *
04357     * @throws arcs::common_interface::AuboException
04358     *
04359     * @par Python function prototype
04360     * getToolDigitalInputs(self: pyaubo_sdk.IoControl) -> int
04361     *
04362     * @par Lua function prototype
04363     * getToolDigitalInputs() -> number
04364     *
04365     * @par Lua example
04366     * num = getToolDigitalInputs()
04367     *
04368     * @par JSON-RPC request example
04369     * {"jsonrpc":"2.0","method":"rob1.IoControl.getToolDigitalInputs","params":[],"id":1}
04370     *
04371     * @par JSON-RPC response example
04372     * {"id":1,"jsonrpc":"2.0","result":0}
04373     *
04374     * \endenglish
04375     */
04376 uint32_t getToolDigitalInputs();
04377
04378 /**
04379  * @ingroup IoControl
04380  * \chinese
04381  * 获取可配置数字输入值
04382  *
04383  * @note 可用于获取安全 IO 的输入值
04384  *
04385  * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
04386  * 例如, 0 表示第一个管脚。
04387  *
04388  * @return 高电平返回 true; 低电平返回 false
04389  *
04390  * @throws arcs::common_interface::AuboException
04391  *
04392  * @par Python 函数原型
04393  * getConfigurableDigitalInput(self: pyaubo_sdk.IoControl, arg0: int) ->
04394  * bool
04395  *
04396  * @par Lua 函数原型
04397  * getConfigurableDigitalInput(index: number) -> boolean
04398  *
04399  * @par Lua 示例
04400  * status = getConfigurableDigitalInput(0)
04401  *
04402  * @par JSON-RPC 请求示例
04403  * {"jsonrpc":"2.0","method":"rob1.IoControl.getConfigurableDigitalInput","params":[0],"id":1}
04404  *
04405  * @par JSON-RPC 响应示例
04406  * {"id":1,"jsonrpc":"2.0","result":false}
04407  *
04408  * \endchinese
04409  * \english
04410  * Get the value of configurable digital input.
04411  *
04412  * @note Can be used to get the value of safety IO.
04413  *
04414  * @param index: Indicates the IO pin, starting from 0.
04415  * For example, 0 means the first pin.
04416  *
04417  * @return Returns true for high level; false for low level.
04418  *
04419  * @throws arcs::common_interface::AuboException
04420  *
04421  * @par Python function prototype
04422  * getConfigurableDigitalInput(self: pyaubo_sdk.IoControl, arg0: int) ->
04423  * bool
04424  *
04425  * @par Lua function prototype
04426  * getConfigurableDigitalInput(index: number) -> boolean

```

```

04427     *
04428     * @par Lua example
04429     * status = getConfigurableDigitalInput(0)
04430     *
04431     * @par JSON-RPC request example
04432     * {"jsonrpc":"2.0","method":"rob1.IoControl.getConfigurableDigitalInput","params":[0],"id":1}
04433     *
04434     * @par JSON-RPC response example
04435     * {"id":1,"jsonrpc":"2.0","result":false}
04436     *
04437     * \endenglish
04438     */
04439 bool getConfigurableDigitalInput(int index);
04440
04441 /**
04442  * @ingroup IoControl
04443  * \chinese
04444  * 获取所有的可配置数字输入值
04445  *
04446  * @note 可用于获取安全 IO 的输入值
04447  *
04448  * @return 所有的可配置数字输入值 \n
04449  * 例如, 当返回值是 2863267846 时, 换成 2 进制后是 10101010101010000000000000110。
04450  * 后 16 位就是所有的输入状态值,
04451  * 最后一位表示管脚 0 的输入状态值, 倒数第二位表示管脚 1 的输入状态值, 以此类推。 \n
04452  * 1 表示高电平状态, 0 表示低电平状态
04453  *
04454  * @throws arcs::common_interface::AuboException
04455  *
04456  * @par Python 函数原型
04457  * getConfigurableDigitalInputs(self: pyaubo_sdk.IoControl) -> int
04458  *
04459  * @par Lua 函数原型
04460  * getConfigurableDigitalInputs() -> number
04461  *
04462  * @par Lua 示例
04463  * num = getConfigurableDigitalInputs()
04464  *
04465  * @par JSON-RPC 请求示例
04466  * {"jsonrpc":"2.0","method":"rob1.IoControl.getConfigurableDigitalInputs","params":[],"id":1}
04467  *
04468  * @par JSON-RPC 响应示例
04469  * {"id":1,"jsonrpc":"2.0","result":0}
04470  *
04471  * \endchinese
04472  * \english
04473  * Get all configurable digital input values.
04474  *
04475  * @note Can be used to get the value of safety IO.
04476  *
04477  * @return All configurable digital input values.\n
04478  * For example, if the return value is 2863267846, its binary representation
04479  * is 101010101010101000000000000000110. The lower 16 bits represent the
04480  * status of all input pins, the least significant bit indicates the input
04481  * status of pin 0, the second least significant bit indicates pin 1, and so
04482  * on.\n 1 means high level, 0 means low level.
04483  *
04484  * @throws arcs::common_interface::AuboException
04485  *
04486  * @par Python function prototype
04487  * getConfigurableDigitalInputs(self: pyaubo_sdk.IoControl) -> int
04488  *
04489  * @par Lua function prototype
04490  * getConfigurableDigitalInputs() -> number
04491  *
04492  * @par Lua example
04493  * num = getConfigurableDigitalInputs()
04494  *
04495  * @par JSON-RPC request example
04496  * {"jsonrpc":"2.0","method":"rob1.IoControl.getConfigurableDigitalInputs","params":[],"id":1}
04497  *
04498  * @par JSON-RPC response example
04499  * {"id":1,"jsonrpc":"2.0","result":0}
04500  *
04501  * \endenglish
04502  */
04503 uint32_t getConfigurableDigitalInputs();
04504
04505 /**
04506  * @ingroup IoControl
04507  * \chinese
04508  * 获取标准数字输出值
04509  *
04510  * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
04511  * 例如, 0 表示第一个管脚。
04512  *
04513  * @return 高电平返回 true; 低电平返回 false

```

```

04514      *
04515      * @throws arcs::common_interface::AuboException
04516      *
04517      * @par Python 函数原型
04518      * getStandardDigitalOutput(self: pyaubo_sdk.IoControl, arg0: int) -> bool
04519      *
04520      * @par Lua 函数原型
04521      * getStandardDigitalOutput(index: number) -> boolean
04522      *
04523      * @par Lua 示例
04524      * status = getStandardDigitalOutput(0)
04525      *
04526      * @par JSON-RPC 请求示例
04527      * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardDigitalOutput", "params": [0], "id": 1}
04528      *
04529      * @par JSON-RPC 响应示例
04530      * {"id": 1, "jsonrpc": "2.0", "result": true}
04531      *
04532      * \endchinese
04533      * \english
04534      * Get the value of a standard digital output.
04535      *
04536      * @param index: Indicates the IO pin, starting from 0.
04537      * For example, 0 means the first pin.
04538      *
04539      * @return Returns true for high level; false for low level.
04540      *
04541      * @throws arcs::common_interface::AuboException
04542      *
04543      * @par Python function prototype
04544      * getStandardDigitalOutput(self: pyaubo_sdk.IoControl, arg0: int) -> bool
04545      *
04546      * @par Lua function prototype
04547      * getStandardDigitalOutput(index: number) -> boolean
04548      *
04549      * @par Lua example
04550      * status = getStandardDigitalOutput(0)
04551      *
04552      * @par JSON-RPC request example
04553      * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardDigitalOutput", "params": [0], "id": 1}
04554      *
04555      * @par JSON-RPC response example
04556      * {"id": 1, "jsonrpc": "2.0", "result": true}
04557      *
04558      * \endenglish
04559      */
04560 bool getStandardDigitalOutput(int index);
04561
04562 /**
04563  * @ingroup IoControl
04564  * \chinese
04565  * 获取所有的标准数字输出值
04566  *
04567  * @return 所有的标准数字输出值 \n
04568  * 例如, 当返回值是 2863267846 时, 换成 2 进制后是 10101010101010000000000000110。
04569  * 后 16 位就是所有的标准数字输出状态值,
04570  * 最后一位表示 DIO0 的输出状态值, 倒数第二位表示 DIO1 的输出状态值, 以此类推。 \n
04571  * 1 表示高电平状态, 0 表示低电平状态。
04572  *
04573  * @throws arcs::common_interface::AuboException
04574  *
04575  * @par Python 函数原型
04576  * getStandardDigitalOutputs(self: pyaubo_sdk.IoControl) -> int
04577  *
04578  * @par Lua 函数原型
04579  * getStandardDigitalOutputs() -> number
04580  *
04581  * @par Lua 示例
04582  * num = getStandardDigitalOutputs()
04583  *
04584  * @par JSON-RPC 请求示例
04585  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardDigitalOutputs", "params": [], "id": 1}
04586  *
04587  * @par JSON-RPC 响应示例
04588  * {"id": 1, "jsonrpc": "2.0", "result": 69}
04589  *
04590  * \endchinese
04591  * \english
04592  * Get all standard digital output values.
04593  *
04594  * @return All standard digital output values. \n
04595  * For example, if the return value is 2863267846, its binary representation
04596  * is 1010101010101000000000000000110. The lower 16 bits represent the
04597  * status of all standard digital outputs, the least significant bit
04598  * indicates the output status of D000, the second least significant bit
04599  * indicates D001, and so on. \n 1 means high level, 0 means low level.
04600  *

```

```

04601     * @throws arcs::common_interface::AuboException
04602     *
04603     * @par Python function prototype
04604     * getStandardDigitalOutputs(self: pyaubo_sdk.IoControl) -> int
04605     *
04606     * @par Lua function prototype
04607     * getStandardDigitalOutputs() -> number
04608     *
04609     * @par Lua example
04610     * num = getStandardDigitalOutputs()
04611     *
04612     * @par JSON-RPC request example
04613     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardDigitalOutputs", "params": [], "id": 1}
04614     *
04615     * @par JSON-RPC response example
04616     * {"id": 1, "jsonrpc": "2.0", "result": 69}
04617     *
04618     * \endenglish
04619     */
04620 uint32_t getStandardDigitalOutputs();
04621
04622 /**
04623  * @ingroup IoControl
04624  * \chinese
04625  * 获取组数字输出值
04626  *
04627  * 按名称查找组数字输出后，读取对应组成员并按位拼装返回整数。
04628  *
04629  * @param name: 组名称。
04630  *
04631  * @return 返回组数字输出的组合值。
04632  *
04633  * @throws arcs::common_interface::AuboException
04634  *
04635  * @par Python 函数原型
04636  * getGroupOutput(self: pyaubo_sdk.IoControl, arg0: str) -> int
04637  *
04638  * @par Lua 函数原型
04639  * getGroupOutput(name: string) -> number
04640  *
04641  * @par Lua 示例
04642  * value = getGroupOutput("go_state")
04643  *
04644  * @par JSON-RPC 请求示例
04645  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getGroupOutput", "params": ["go_state"], "id": 1}
04646  *
04647  * @par JSON-RPC 响应示例
04648  * {"id": 1, "jsonrpc": "2.0", "result": 13}
04649  *
04650  * \endchinese
04651  * \english
04652  * Get the grouped digital output value.
04653  *
04654  * Looks up the named grouped digital output, reads the group's members,
04655  * and combines them into an integer value.
04656  *
04657  * @param name: Group name.
04658  *
04659  * @return Returns the combined grouped digital output value.
04660  *
04661  * @throws arcs::common_interface::AuboException
04662  *
04663  * @par Python function prototype
04664  * getGroupOutput(self: pyaubo_sdk.IoControl, arg0: str) -> int
04665  *
04666  * @par Lua function prototype
04667  * getGroupOutput(name: string) -> number
04668  *
04669  * @par Lua example
04670  * value = getGroupOutput("go_state")
04671  *
04672  * @par JSON-RPC request example
04673  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getGroupOutput", "params": ["go_state"], "id": 1}
04674  *
04675  * @par JSON-RPC response example
04676  * {"id": 1, "jsonrpc": "2.0", "result": 13}
04677  *
04678  * \endenglish
04679  */
04680 uint32_t getGroupOutput(const std::string &name);
04681
04682 /**
04683  * @ingroup IoControl
04684  * \chinese
04685  * 获取工具端数字输出值
04686  *
04687  * @param index: 表示 IO 口的管脚，管脚编号从 0 开始。

```



```

04688     * 例如, 0 表示第一个管脚。
04689     *
04690     * @return 高电平返回 true; 低电平返回 false
04691     *
04692     * @throws arcs::common_interface::AuboException
04693     *
04694     * @par Python 函数原型
04695     * getToolDigitalOutput(self: pyaubo_sdk.IoControl, arg0: int) -> bool
04696     *
04697     * @par Lua 函数原型
04698     * getToolDigitalOutput(index: number) -> boolean
04699     *
04700     * @par Lua 示例
04701     * status = getToolDigitalOutput(0)
04702     *
04703     * @par JSON-RPC 请求示例
04704     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolDigitalOutput", "params": [0], "id": 1}
04705     *
04706     * @par JSON-RPC 响应示例
04707     * {"id": 1, "jsonrpc": "2.0", "result": false}
04708     *
04709     * \endchinese
04710     * \english
04711     * Get the value of tool digital output.
04712     *
04713     * @param index: Indicates the IO pin, starting from 0.
04714     * For example, 0 means the first pin.
04715     *
04716     * @return Returns true for high level; false for low level.
04717     *
04718     * @throws arcs::common_interface::AuboException
04719     *
04720     * @par Python function prototype
04721     * getToolDigitalOutput(self: pyaubo_sdk.IoControl, arg0: int) -> bool
04722     *
04723     * @par Lua function prototype
04724     * getToolDigitalOutput(index: number) -> boolean
04725     *
04726     * @par Lua example
04727     * status = getToolDigitalOutput(0)
04728     *
04729     * @par JSON-RPC request example
04730     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolDigitalOutput", "params": [0], "id": 1}
04731     *
04732     * @par JSON-RPC response example
04733     * {"id": 1, "jsonrpc": "2.0", "result": false}
04734     *
04735     * \endenglish
04736     */
04737 bool getToolDigitalOutput(int index);
04738
04739 /**
04740     * @ingroup IoControl
04741     * \chinese
04742     * 获取所有的工具端数字输出值
04743     *
04744     * @return 所有的工具端数字输出值
04745     *
04746     * @throws arcs::common_interface::AuboException
04747     *
04748     * @par Python 函数原型
04749     * getToolDigitalOutputs(self: pyaubo_sdk.IoControl) -> int
04750     *
04751     * @par Lua 函数原型
04752     * getToolDigitalOutputs() -> number
04753     *
04754     * @par Lua 示例
04755     * num = getToolDigitalOutputs()
04756     *
04757     * @par JSON-RPC 请求示例
04758     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolDigitalOutputs", "params": [], "id": 1}
04759     *
04760     * @par JSON-RPC 响应示例
04761     * {"id": 1, "jsonrpc": "2.0", "result": 9}
04762     *
04763     * \endchinese
04764     * \english
04765     * Get all tool digital output values.
04766     *
04767     * @return All tool digital output values.
04768     *
04769     * @throws arcs::common_interface::AuboException
04770     *
04771     * @par Python function prototype
04772     * getToolDigitalOutputs(self: pyaubo_sdk.IoControl) -> int
04773     *
04774     * @par Lua function prototype

```

```

04775     * getToolDigitalOutputs() -> number
04776     *
04777     * @par Lua example
04778     * num = getToolDigitalOutputs()
04779     *
04780     * @par JSON-RPC request example
04781     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolDigitalOutputs", "params": [], "id": 1}
04782     *
04783     * @par JSON-RPC response example
04784     * {"id": 1, "jsonrpc": "2.0", "result": 9}
04785     *
04786     * \endenglish
04787     */
04788     uint32_t getToolDigitalOutputs();
04789
04790     /**
04791     * @ingroup IoControl
04792     * \chinese
04793     * 获取可配值数字输出值
04794     *
04795     * @note 可用于获取安全 IO 的输出值
04796     *
04797     * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
04798     * 例如, 0 表示第一个管脚。
04799     *
04800     * @return 高电平返回 true; 低电平返回 false
04801     *
04802     * @throws arcs::common_interface::AuboException
04803     *
04804     * @par Python 函数原型
04805     * getConfigurableDigitalOutput(self: pyaubo_sdk.IoControl, arg0: int) ->
04806     * bool
04807     *
04808     * @par Lua 函数原型
04809     * getConfigurableDigitalOutput(index: number) -> boolean
04810     *
04811     * @par Lua 示例
04812     * status = getConfigurableDigitalOutput(0)
04813     *
04814     * @par JSON-RPC 请求示例
04815     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getConfigurableDigitalOutput", "params": [0], "id": 1}
04816     *
04817     * @par JSON-RPC 响应示例
04818     * {"id": 1, "jsonrpc": "2.0", "result": true}
04819     *
04820     * \endchinese
04821     * \english
04822     * Get the value of configurable digital output.
04823     *
04824     * @note Can be used to get the value of safety IO.
04825     *
04826     * @param index: Indicates the IO pin, starting from 0.
04827     * For example, 0 means the first pin.
04828     *
04829     * @return Returns true for high level; false for low level.
04830     *
04831     * @throws arcs::common_interface::AuboException
04832     *
04833     * @par Python function prototype
04834     * getConfigurableDigitalOutput(self: pyaubo_sdk.IoControl, arg0: int) ->
04835     * bool
04836     *
04837     * @par Lua function prototype
04838     * getConfigurableDigitalOutput(index: number) -> boolean
04839     *
04840     * @par Lua example
04841     * status = getConfigurableDigitalOutput(0)
04842     *
04843     * @par JSON-RPC request example
04844     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getConfigurableDigitalOutput", "params": [0], "id": 1}
04845     *
04846     * @par JSON-RPC response example
04847     * {"id": 1, "jsonrpc": "2.0", "result": true}
04848     *
04849     * \endenglish
04850     */
04851     bool getConfigurableDigitalOutput(int index);
04852
04853     /**
04854     * @ingroup IoControl
04855     * \chinese
04856     * 获取所有的可配值数字输出值
04857     *
04858     * @note 可用于获取安全 IO 的输出值
04859     *
04860     * @return 所有的可配值数字输出 \n
04861     * 例如, 当返回值是 2863267846 时, 换成 2 进制后是 10101010101010000000000000110。

```

```

04862 * 后 16 位就是所有的输出值，
04863 * 最后一位表示管脚 0 的输出状态值，倒数第二位表示管脚 1 的输出状态值，以此类推。\\n
04864 * 1 表示高电平状态，0 表示低电平状态。
04865 *
04866 * @throws arcs::common_interface::AuboException
04867 *
04868 * @par Python 函数原型
04869 * getConfigurableDigitalOutputs(self: pyaubo_sdk.IoControl) -> int
04870 *
04871 * @par Lua 函数原型
04872 * getConfigurableDigitalOutputs() -> number
04873 *
04874 * @par Lua 示例
04875 * num = getConfigurableDigitalOutputs()
04876 *
04877 * @par JSON-RPC 请求示例
04878 * {"jsonrpc": "2.0", "method": "rob1.IoControl.getConfigurableDigitalOutputs", "params": [], "id": 1}
04879 *
04880 * @par JSON-RPC 响应示例
04881 * {"id": 1, "jsonrpc": "2.0", "result": 1}
04882 *
04883 * \\endchinese
04884 * \\english
04885 * Get all configurable digital output values.
04886 *
04887 * @note Can be used to get the value of safety IO.
04888 *
04889 * @return All configurable digital output values.\\n
04890 * For example, if the return value is 2863267846, its binary representation
04891 * is 10101010101010100000000000000110. The lower 16 bits represent the
04892 * status of all output pins, the least significant bit indicates the output
04893 * status of pin 0, the second least significant bit indicates pin 1, and so
04894 * on.\\n 1 means high level, 0 means low level.
04895 *
04896 * @throws arcs::common_interface::AuboException
04897 *
04898 * @par Python function prototype
04899 * getConfigurableDigitalOutputs(self: pyaubo_sdk.IoControl) -> int
04900 *
04901 * @par Lua function prototype
04902 * getConfigurableDigitalOutputs() -> number
04903 *
04904 * @par Lua example
04905 * num = getConfigurableDigitalOutputs()
04906 *
04907 * @par JSON-RPC request example
04908 * {"jsonrpc": "2.0", "method": "rob1.IoControl.getConfigurableDigitalOutputs", "params": [], "id": 1}
04909 *
04910 * @par JSON-RPC response example
04911 * {"id": 1, "jsonrpc": "2.0", "result": 1}
04912 *
04913 * \\endenglish
04914 */
04915 uint32_t getConfigurableDigitalOutputs();
04916
04917 /**
04918 * @ingroup IoControl
04919 * \\chinese
04920 * 获取标准模拟输入值
04921 *
04922 * @param index: 表示 IO 口的管脚，管脚编号从 0 开始。
04923 * 例如，0 表示第一个管脚。
04924 *
04925 * @return 标准模拟输入值
04926 *
04927 * @throws arcs::common_interface::AuboException
04928 *
04929 * @par Python 函数原型
04930 * getStandardAnalogInput(self: pyaubo_sdk.IoControl, arg0: int) -> float
04931 *
04932 * @par Lua 函数原型
04933 * getStandardAnalogInput(index: number) -> number
04934 *
04935 * @par Lua 示例
04936 * getStandardAnalogInput(0)
04937 *
04938 * @par JSON-RPC 请求示例
04939 * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardAnalogInput", "params": [0], "id": 1}
04940 *
04941 * @par JSON-RPC 响应示例
04942 * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
04943 *
04944 * \\endchinese
04945 * \\english
04946 * Get the value of standard analog input.
04947 *
04948 * @param index: Indicates the IO pin, starting from 0.

```

```

04949     * For example, 0 means the first pin.
04950     *
04951     * @return Standard analog input value.
04952     *
04953     * @throws arcs::common_interface::AuboException
04954     *
04955     * @par Python function prototype
04956     * getStandardAnalogInput(self: pyaubo_sdk.IoControl, arg0: int) -> float
04957     *
04958     * @par Lua function prototype
04959     * getStandardAnalogInput(index: number) -> number
04960     *
04961     * @par Lua example
04962     * getStandardAnalogInput(0)
04963     *
04964     * @par JSON-RPC request example
04965     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardAnalogInput", "params": [0], "id": 1}
04966     *
04967     * @par JSON-RPC response example
04968     * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
04969     *
04970     * \endenglish
04971     */
04972 double getStandardAnalogInput(int index);
04973
04974 /**
04975  * @ingroup IoControl
04976  * \chinese
04977  * 获取工具端模拟输入值
04978  *
04979  * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
04980  * 例如, 0 表示第一个管脚。
04981  *
04982  * @return 工具端模拟输入值
04983  *
04984  * @throws arcs::common_interface::AuboException
04985  *
04986  * @par Python 函数原型
04987  * getToolAnalogInput(self: pyaubo_sdk.IoControl, arg0: int) -> float
04988  *
04989  * @par Lua 函数原型
04990  * getToolAnalogInput(index: number) -> number
04991  *
04992  * @par Lua 示例
04993  * num = getToolAnalogInput(0)
04994  *
04995  * @par JSON-RPC 请求示例
04996  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolAnalogInput", "params": [0], "id": 1}
04997  *
04998  * @par JSON-RPC 响应示例
04999  * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
05000  *
05001  * \endchinese
05002  * \english
05003  * Get the value of tool analog input.
05004  *
05005  * @param index: Indicates the IO pin, starting from 0.
05006  * For example, 0 means the first pin.
05007  *
05008  * @return Tool analog input value.
05009  *
05010  * @throws arcs::common_interface::AuboException
05011  *
05012  * @par Python function prototype
05013  * getToolAnalogInput(self: pyaubo_sdk.IoControl, arg0: int) -> float
05014  *
05015  * @par Lua function prototype
05016  * getToolAnalogInput(index: number) -> number
05017  *
05018  * @par Lua example
05019  * num = getToolAnalogInput(0)
05020  *
05021  * @par JSON-RPC request example
05022  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolAnalogInput", "params": [0], "id": 1}
05023  *
05024  * @par JSON-RPC response example
05025  * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
05026  *
05027  * \endenglish
05028  */
05029 double getToolAnalogInput(int index);
05030
05031 /**
05032  * @ingroup IoControl
05033  * \chinese
05034  * 获取标准模拟输出值
05035  *

```

```

05036     * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
05037     * 例如, 0 表示第一个管脚。
05038     *
05039     * @return 标准模拟输出值
05040     *
05041     * @throws arcs::common_interface::AuboException
05042     *
05043     * @par Python 函数原型
05044     * getStandardAnalogOutput(self: pyaubo_sdk.IoControl, arg0: int) -> float
05045     *
05046     * @par Lua 函数原型
05047     * getStandardAnalogOutput(index: number) -> number
05048     *
05049     * @par Lua 示例
05050     * num = getStandardAnalogOutput(0)
05051     *
05052     * @par JSON-RPC 请求示例
05053     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardAnalogOutput", "params": [0], "id": 1}
05054     *
05055     * @par JSON-RPC 响应示例
05056     * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
05057     *
05058     * \endchinese
05059     * \english
05060     * Get the value of standard analog output.
05061     *
05062     * @param index: Indicates the IO pin, starting from 0.
05063     * For example, 0 means the first pin.
05064     *
05065     * @return Standard analog output value.
05066     *
05067     * @throws arcs::common_interface::AuboException
05068     *
05069     * @par Python function prototype
05070     * getStandardAnalogOutput(self: pyaubo_sdk.IoControl, arg0: int) -> float
05071     *
05072     * @par Lua function prototype
05073     * getStandardAnalogOutput(index: number) -> number
05074     *
05075     * @par Lua example
05076     * num = getStandardAnalogOutput(0)
05077     *
05078     * @par JSON-RPC request example
05079     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStandardAnalogOutput", "params": [0], "id": 1}
05080     *
05081     * @par JSON-RPC response example
05082     * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
05083     *
05084     * \endenglish
05085     */
05086 double getStandardAnalogOutput(int index);
05087
05088 /**
05089     * @ingroup IoControl
05090     * \chinese
05091     * 获取工具端模拟输出值
05092     *
05093     * @param index: 表示 IO 口的管脚, 管脚编号从 0 开始。
05094     * 例如, 0 表示第一个管脚。
05095     *
05096     * @return 工具端模拟输出值
05097     *
05098     * @throws arcs::common_interface::AuboException
05099     *
05100     * @par Python 函数原型
05101     * getToolAnalogOutput(self: pyaubo_sdk.IoControl, arg0: int) -> float
05102     *
05103     * @par Lua 函数原型
05104     * getToolAnalogOutput(index: number) -> number
05105     *
05106     * @par Lua 示例
05107     * num = getToolAnalogOutput(0)
05108     *
05109     * @par JSON-RPC 请求示例
05110     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolAnalogOutput", "params": [0], "id": 1}
05111     *
05112     * @par JSON-RPC 响应示例
05113     * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
05114     *
05115     * \endchinese
05116     * \english
05117     * Get the value of tool analog output.
05118     *
05119     * @param index: Indicates the IO pin, starting from 0.
05120     * For example, 0 means the first pin.
05121     *
05122     * @return Tool analog output value.

```

```

05123     *
05124     * @throws arcs::common_interface::AuboException
05125     *
05126     * @par Python function prototype
05127     * getToolAnalogOutput(self: pyaubo_sdk.IoControl, arg0: int) -> float
05128     *
05129     * @par Lua function prototype
05130     * getToolAnalogOutput(index: number) -> number
05131     *
05132     * @par Lua example
05133     * num = getToolAnalogOutput(0)
05134     *
05135     * @par JSON-RPC request example
05136     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getToolAnalogOutput", "params": [0], "id": 1}
05137     *
05138     * @par JSON-RPC response example
05139     * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
05140     *
05141     * \endenglish
05142     */
05143 double getToolAnalogOutput(int index);
05144
05145 /**
05146  * @ingroup IoControl
05147  * \chinese
05148  * 获取联动输入数量
05149  *
05150  * @return 联动输入数量
05151  *
05152  * @throws arcs::common_interface::AuboException
05153  *
05154  * @par Python 函数原型
05155  * getStaticLinkInputNum(self: pyaubo_sdk.IoControl) -> int
05156  *
05157  * @par Lua 函数原型
05158  * getStaticLinkInputNum() -> number
05159  *
05160  * @par Lua 示例
05161  * num = getStaticLinkInputNum()
05162  *
05163  * @par JSON-RPC 请求示例
05164  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStaticLinkInputNum", "params": [], "id": 1}
05165  *
05166  * @par JSON-RPC 响应示例
05167  * {"id": 1, "jsonrpc": "2.0", "result": 8}
05168  *
05169  * \endchinese
05170  * \english
05171  * Get the number of static link inputs.
05172  *
05173  * @return Number of static link inputs.
05174  *
05175  * @throws arcs::common_interface::AuboException
05176  *
05177  * @par Python function prototype
05178  * getStaticLinkInputNum(self: pyaubo_sdk.IoControl) -> int
05179  *
05180  * @par Lua function prototype
05181  * getStaticLinkInputNum() -> number
05182  *
05183  * @par Lua example
05184  * num = getStaticLinkInputNum()
05185  *
05186  * @par JSON-RPC request example
05187  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStaticLinkInputNum", "params": [], "id": 1}
05188  *
05189  * @par JSON-RPC response example
05190  * {"id": 1, "jsonrpc": "2.0", "result": 8}
05191  *
05192  * \endenglish
05193  */
05194 int getStaticLinkInputNum();
05195
05196 /**
05197  * @ingroup IoControl
05198  * \chinese
05199  * 获取联动输出数量
05200  *
05201  * @return 联动输出数量
05202  *
05203  * @throws arcs::common_interface::AuboException
05204  *
05205  * @par Python 函数原型
05206  * getStaticLinkOutputNum(self: pyaubo_sdk.IoControl) -> int
05207  *
05208  * @par Lua 函数原型
05209  * getStaticLinkOutputNum() -> number

```

```

05210      *
05211      * @par Lua 示例
05212      * num = getStaticLinkOutputNum()
05213      *
05214      * @par JSON-RPC 请求示例
05215      * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStaticLinkOutputNum", "params": [], "id": 1}
05216      *
05217      * @par JSON-RPC 响应示例
05218      * {"id": 1, "jsonrpc": "2.0", "result": 0}
05219      *
05220      * \endchinese
05221      * \english
05222      * Get the number of static link outputs.
05223      *
05224      * @return Number of static link outputs.
05225      *
05226      * @throws arcs::common_interface::AuboException
05227      *
05228      * @par Python function prototype
05229      * getStaticLinkOutputNum(self: pyaubo_sdk.IoControl) -> int
05230      *
05231      * @par Lua function prototype
05232      * getStaticLinkOutputNum() -> number
05233      *
05234      * @par Lua example
05235      * num = getStaticLinkOutputNum()
05236      *
05237      * @par JSON-RPC request example
05238      * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStaticLinkOutputNum", "params": [], "id": 1}
05239      *
05240      * @par JSON-RPC response example
05241      * {"id": 1, "jsonrpc": "2.0", "result": 0}
05242      *
05243      * \endenglish
05244      */
05245 int getStaticLinkOutputNum();
05246
05247 /**
05248  * @ingroup IoControl
05249  * \chinese
05250  * 获取所有的联动输入值
05251  *
05252  * @return 所有的联动输入值 \n
05253  * 例如, 当返回值是 2863267846 时, 换成 2 进制后是 10101010101010000000000000110。
05254  * 后 16 位就是所有的联动输入状态值,
05255  * 最后一位表示管脚 0 的输入状态值, 倒数第二位表示管脚 1 的输入状态值, 以此类推。 \n
05256  * 1 表示高电平状态, 0 表示低电平状态。
05257  *
05258  * @throws arcs::common_interface::AuboException
05259  *
05260  * @par Python 函数原型
05261  * getStaticLinkInputs(self: pyaubo_sdk.IoControl) -> int
05262  *
05263  * @par Lua 函数原型
05264  * getStaticLinkInputs() -> number
05265  *
05266  * @par Lua 示例
05267  * num = getStaticLinkInputs()
05268  *
05269  * @par JSON-RPC 请求示例
05270  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStaticLinkInputs", "params": [], "id": 1}
05271  *
05272  * @par JSON-RPC 响应示例
05273  * {"id": 1, "jsonrpc": "2.0", "result": 0}
05274  *
05275  * \endchinese
05276  * \english
05277  * Get all static link input values.
05278  *
05279  * @return All static link input values. \n
05280  * For example, if the return value is 2863267846, its binary representation
05281  * is 101010101010101000000000000000110. The lower 16 bits represent the
05282  * status of all static link inputs, the least significant bit indicates the
05283  * input status of pin 0, the second least significant bit indicates pin 1,
05284  * and so on. \n 1 means high level, 0 means low level.
05285  *
05286  * @throws arcs::common_interface::AuboException
05287  *
05288  * @par Python function prototype
05289  * getStaticLinkInputs(self: pyaubo_sdk.IoControl) -> int
05290  *
05291  * @par Lua function prototype
05292  * getStaticLinkInputs() -> number
05293  *
05294  * @par Lua example
05295  * num = getStaticLinkInputs()
05296  *

```

```

05297     * @par JSON-RPC request example
05298     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStaticLinkInputs", "params": [], "id": 1}
05299     *
05300     * @par JSON-RPC response example
05301     * {"id": 1, "jsonrpc": "2.0", "result": 0}
05302     *
05303     * \endenglish
05304     */
05305 uint32_t getStaticLinkInputs();
05306
05307 /**
05308     * @ingroup IoControl
05309     * \chinese
05310     * 获取所有的联动输出值
05311     *
05312     * @return 返回所有的联动输出值 \n
05313     * 例如, 当返回值是 2863267846 时, 换成 2 进制后是 10101010101010000000000000110。
05314     * 后 16 位就是所有的联动输出状态值,
05315     * 最后一位表示管脚 0 的输出状态值, 倒数第二位表示管脚 1 的输出状态值, 以此类推。 \n
05316     * 1 表示高电平状态, 0 表示低电平状态。
05317     *
05318     * @throws arcs::common_interface::AuboException
05319     *
05320     * @par Python 函数原型
05321     * getStaticLinkOutputs(self: pyaubo_sdk.IoControl) -> int
05322     *
05323     * @par Lua 函数原型
05324     * getStaticLinkOutputs() -> number
05325     *
05326     * @par Lua 示例
05327     * num = getStaticLinkOutputs()
05328     *
05329     * @par JSON-RPC 请求示例
05330     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStaticLinkOutputs", "params": [], "id": 1}
05331     *
05332     * @par JSON-RPC 响应示例
05333     * {"id": 1, "jsonrpc": "2.0", "result": 0}
05334     *
05335     * \endchinese
05336     * \english
05337     * Get all static link output values.
05338     *
05339     * @return Returns all static link output values. \n
05340     * For example, if the return value is 2863267846, its binary representation
05341     * is 101010101010101000000000000000110. The lower 16 bits represent the
05342     * status of all static link outputs, the least significant bit indicates
05343     * the output status of pin 0, the second least significant bit indicates
05344     * pin 1, and so on. \n 1 means high level, 0 means low level.
05345     *
05346     * @throws arcs::common_interface::AuboException
05347     *
05348     * @par Python function prototype
05349     * getStaticLinkOutputs(self: pyaubo_sdk.IoControl) -> int
05350     *
05351     * @par Lua function prototype
05352     * getStaticLinkOutputs() -> number
05353     *
05354     * @par Lua example
05355     * num = getStaticLinkOutputs()
05356     *
05357     * @par JSON-RPC request example
05358     * {"jsonrpc": "2.0", "method": "rob1.IoControl.getStaticLinkOutputs", "params": [], "id": 1}
05359     *
05360     * @par JSON-RPC response example
05361     * {"id": 1, "jsonrpc": "2.0", "result": 0}
05362     *
05363     * \endenglish
05364     */
05365 uint32_t getStaticLinkOutputs();
05366
05367 /**
05368     * @ingroup IoControl
05369     * \chinese
05370     * 机器人是否配置了编码器
05371     * 集成编码器的编号为 0
05372     *
05373     * @return 机器人配置编码器返回 true, 反之返回 false
05374     *
05375     * @throws arcs::common_interface::AuboException
05376     *
05377     * @par JSON-RPC 请求示例
05378     * {"jsonrpc": "2.0", "method": "rob1.IoControl.hasEncoderSensor", "params": [], "id": 1}
05379     *
05380     * @par JSON-RPC 响应示例
05381     * {"id": 1, "jsonrpc": "2.0", "result": true}
05382     *
05383     * \endchinese

```



```

05384     * \english
05385     * Whether the robot is equipped with an encoder.
05386     * The integrated encoder number is 0.
05387     *
05388     * @return Returns true if the robot is equipped with an encoder, otherwise
05389     * false.
05390     *
05391     * @throws arcs::common_interface::AuboException
05392     *
05393     * @par JSON-RPC request example
05394     * {"jsonrpc":"2.0","method":"rob1.IoControl.hasEncoderSensor","params":[],"id":1}
05395     *
05396     * @par JSON-RPC response example
05397     * {"id":1,"jsonrpc":"2.0","result":true}
05398     *
05399     * \endenglish
05400     */
05401 bool hasEncoderSensor();
05402 /**
05403     * @ingroup IoControl
05404     * \chinese
05405     * 设置集成编码器的解码方式
05406     *
05407     * @param type
05408     * 0-禁用编码器
05409     * 1-AB 正交
05410     * 2-AB 正交 +Z
05411     * 3-AB 差分正交
05412     * 4-AB 差分正交 +Z 差分
05413     *
05414     * @param range_id
05415     * 0 表示 32 位有符号编码器, 范围为 [-2147483648, 2147483647]
05416     * 1 表示 8 位无符号编码器, 范围为 [0, 255]
05417     * 2 表示 16 位无符号编码器, 范围为 [0, 65535]
05418     * 3 表示 24 位无符号编码器, 范围为 [0, 16777215]
05419     * 4 表示 32 位无符号编码器, 范围为 [0, 4294967295]
05420     *
05421     * @return 成功返回 0; 失败返回错误码
05422     * AUBO_NO_ACCESS
05423     * AUBO_BUSY
05424     * AUBO_BAD_STATE
05425     * -AUBO_INVL_ARGUMENT
05426     * -AUBO_BAD_STATE
05427     *
05428     * @throws arcs::common_interface::AuboException
05429     * \endchinese
05430     * \english
05431     * Set the decoding method of the integrated encoder.
05432     *
05433     * @param type
05434     * 0 - Disable encoder
05435     * 1 - AB quadrature
05436     * 2 - AB quadrature + Z
05437     * 3 - AB differential quadrature
05438     * 4 - AB differential quadrature + Z differential
05439     *
05440     * @param range_id
05441     * 0 is a 32-bit signed encoder, range [-2147483648, 2147483647]
05442     * 1 is an 8-bit unsigned encoder, range [0, 255]
05443     * 2 is a 16-bit unsigned encoder, range [0, 65535]
05444     * 3 is a 24-bit unsigned encoder, range [0, 16777215]
05445     * 4 is a 32-bit unsigned encoder, range [0, 4294967295]
05446     *
05447     * @return Returns 0 on success; error code on failure.
05448     * AUBO_NO_ACCESS
05449     * AUBO_BUSY
05450     * AUBO_BAD_STATE
05451     * -AUBO_INVL_ARGUMENT
05452     * -AUBO_BAD_STATE
05453     *
05454     * @throws arcs::common_interface::AuboException
05455     * \endenglish
05456     */
05457 int setEncDecoderType(int type, int range_id);
05458 /**
05459     * @ingroup IoControl
05460     * \chinese
05461     * 设置集成编码器脉冲数
05462     *
05463     * @param tick 脉冲数
05464     *
05465     * @return 成功返回 0; 失败返回错误码
05466     * AUBO_NO_ACCESS
05467     * AUBO_BUSY
05468     * AUBO_BAD_STATE
05469     * -AUBO_INVL_ARGUMENT
05470     */

```

```

05471     * -AUBO_BAD_STATE
05472     *
05473     * @throws arcs::common_interface::AuboException
05474     * \endchinese
05475     * \english
05476     * Set the tick count of the integrated encoder.
05477     *
05478     * @param tick Tick count
05479     *
05480     * @return Returns 0 on success; error code on failure.
05481     * AUBO_NO_ACCESS
05482     * AUBO_BUSY
05483     * AUBO_BAD_STATE
05484     * -AUBO_INVL_ARGUMENT
05485     * -AUBO_BAD_STATE
05486     *
05487     * @throws arcs::common_interface::AuboException
05488     * \endenglish
05489     */
05490 int setEncTickCount(int tick);
05491
05492 /**
05493  * @ingroup IoControl
05494  * \chinese
05495  * 获取编码器的解码方式
05496  *
05497  * @return 成功返回 0；失败返回错误码
05498  * AUBO_NO_ACCESS
05499  * AUBO_BUSY
05500  * AUBO_BAD_STATE
05501  * -AUBO_BAD_STATE
05502  *
05503  * @throws arcs::common_interface::AuboException
05504  *
05505  * @par JSON-RPC 请求示例
05506  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getEncDecoderType", "params": [], "id": 1}
05507  *
05508  * @par JSON-RPC 响应示例
05509  * {"id": 1, "jsonrpc": "2.0", "result": 0}
05510  *
05511  * \endchinese
05512  * \english
05513  * Get the decoder type of the encoder.
05514  *
05515  * @return Returns 0 on success; error code on failure.
05516  * AUBO_NO_ACCESS
05517  * AUBO_BUSY
05518  * AUBO_BAD_STATE
05519  * -AUBO_BAD_STATE
05520  *
05521  * @throws arcs::common_interface::AuboException
05522  *
05523  * @par JSON-RPC request example
05524  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getEncDecoderType", "params": [], "id": 1}
05525  *
05526  * @par JSON-RPC response example
05527  * {"id": 1, "jsonrpc": "2.0", "result": 0}
05528  *
05529  * \endenglish
05530  */
05531 int getEncDecoderType();
05532
05533 /**
05534  * @ingroup IoControl
05535  * \chinese
05536  * 获取脉冲数
05537  *
05538  * @return 成功返回 0；失败返回错误码
05539  * AUBO_NO_ACCESS
05540  * AUBO_BUSY
05541  * AUBO_BAD_STATE
05542  * -AUBO_BAD_STATE
05543  *
05544  * @throws arcs::common_interface::AuboException
05545  *
05546  * @par JSON-RPC 请求示例
05547  * {"jsonrpc": "2.0", "method": "rob1.IoControl.getEncTickCount", "params": [], "id": 1}
05548  *
05549  * @par JSON-RPC 响应示例
05550  * {"id": 1, "jsonrpc": "2.0", "result": 0}
05551  *
05552  * \endchinese
05553  * \english
05554  * Get the tick count
05555  *
05556  * @return Returns 0 on success; error code on failure.
05557  * AUBO_NO_ACCESS

```

```

05558     * AUBO_BUSY
05559     * AUBO_BAD_STATE
05560     * -AUBO_BAD_STATE
05561     *
05562     * @throws arcs::common_interface::AuboException
05563     *
05564     * @par JSON-RPC request example
05565     * {"jsonrpc":"2.0","method":"rob1.IoControl.getEncTickCount","params":[],"id":1}
05566     *
05567     * @par JSON-RPC response example
05568     * {"id":1,"jsonrpc":"2.0","result":0}
05569     *
05570     * \endenglish
05571     */
05572 int getEncTickCount();
05573
05574 /**
05575  * @ingroup IoControl
05576  * \chinese
05577  * 防止在计数超出范围时计数错误
05578  *
05579  * @param delta_count
05580  *
05581  * @return 成功返回 0；失败返回错误码
05582  * AUBO_NO_ACCESS
05583  * AUBO_BUSY
05584  * AUBO_BAD_STATE
05585  * -AUBO_BAD_STATE
05586  *
05587  * @throws arcs::common_interface::AuboException
05588  * \endchinese
05589  * \english
05590  * Prevent counting errors when the count exceeds the range
05591  *
05592  * @param delta_count
05593  *
05594  * @return Returns 0 on success; error code on failure.
05595  * AUBO_NO_ACCESS
05596  * AUBO_BUSY
05597  * AUBO_BAD_STATE
05598  * -AUBO_BAD_STATE
05599  *
05600  * @throws arcs::common_interface::AuboException
05601  * \endenglish
05602  */
05603 int unwindEncDeltaTickCount(int delta_count);
05604
05605 /**
05606  * @ingroup IoControl
05607  * \chinese
05608  * 获取末端按钮状态
05609  *
05610  * @return 按下返回 true；否则返回 false
05611  *
05612  * @throws arcs::common_interface::AuboException
05613  *
05614  * @par Python 函数原型
05615  * getToolButtonStatus() -> bool
05616  *
05617  * @par Lua 函数原型
05618  * getToolButtonStatus() -> boolean
05619  *
05620  * @par Lua 示例
05621  * status = getToolButtonStatus()
05622  *
05623  * @par JSON-RPC 请求示例
05624  * {"jsonrpc":"2.0","method":"rob1.IoControl.getToolButtonStatus","params":[],"id":1}
05625  *
05626  * @par JSON-RPC 响应示例
05627  * {"id":1,"jsonrpc":"2.0","result":false}
05628  *
05629  * \endchinese
05630  * \english
05631  * Get the status of the tool button.
05632  *
05633  * @return Returns true if pressed; otherwise false.
05634  *
05635  * @throws arcs::common_interface::AuboException
05636  *
05637  * @par Python function prototype
05638  * getToolButtonStatus() -> bool
05639  *
05640  * @par Lua function prototype
05641  * getToolButtonStatus() -> boolean
05642  *
05643  * @par Lua example
05644  * status = getToolButtonStatus()

```

```

05645     *
05646     * @par JSON-RPC request example
05647     * {"jsonrpc":"2.0","method":"rob1.IoControl.getToolButtonStatus","params":[],"id":1}
05648     *
05649     * @par JSON-RPC response example
05650     * {"id":1,"jsonrpc":"2.0","result":false}
05651     *
05652     * \endenglish
05653     */
05654 bool getToolButtonStatus();
05655
05656 /**
05657  * @ingroup IoControl
05658  * \chinese
05659  * 获取手柄按键状态
05660  *
05661  * @note 获取手柄按键状态
05662  *
05663  * @return 所有的手柄按键输入值 \n
05664  * 例如, 当返回值是 2863267846 时, 换成 2 进制后是 10101010101010000000000000110。
05665  * 后 16 位就是所有的输入状态值,
05666  * 最后一位表示管脚 0 的输入状态值, 倒数第二位表示管脚 1 的输入状态值, 以此类推。 \n
05667  * 1 表示高电平状态, 0 表示低电平状态
05668  *
05669  * @throws arcs::common_interface::AuboException
05670  *
05671  * @par Python 函数原型
05672  * getHandleIoStatus(self: pyaubo_sdk.IoControl) -> int
05673  *
05674  * @par Lua 函数原型
05675  * getHandleIoStatus() -> number
05676  *
05677  * @par Lua 示例
05678  * num = getHandleIoStatus()
05679  *
05680  * @par JSON-RPC 请求示例
05681  * {"jsonrpc":"2.0","method":"rob1.IoControl.getHandleIoStatus","params":[],"id":1}
05682  *
05683  * @par JSON-RPC 响应示例
05684  * {"id":1,"jsonrpc":"2.0","result":0}
05685  *
05686  * \endchinese
05687  * \english
05688  * Get the status of handle buttons.
05689  *
05690  * @note Get the status of handle buttons.
05691  *
05692  * @return All handle button input values. \n
05693  * For example, if the return value is 2863267846, its binary representation
05694  * is 1010101010101000000000000000110. The lower 16 bits represent the
05695  * status of all input pins, the least significant bit indicates the input
05696  * status of pin 0, the second least significant bit indicates pin 1, and so
05697  * on. \n 1 means high level, 0 means low level.
05698  *
05699  * @throws arcs::common_interface::AuboException
05700  *
05701  * @par Python function prototype
05702  * getHandleIoStatus(self: pyaubo_sdk.IoControl) -> int
05703  *
05704  * @par Lua function prototype
05705  * getHandleIoStatus() -> number
05706  *
05707  * @par Lua example
05708  * num = getHandleIoStatus()
05709  *
05710  * @par JSON-RPC request example
05711  * {"jsonrpc":"2.0","method":"rob1.IoControl.getHandleIoStatus","params":[],"id":1}
05712  *
05713  * @par JSON-RPC response example
05714  * {"id":1,"jsonrpc":"2.0","result":0}
05715  *
05716  * \endenglish
05717  */
05718 uint32_t getHandleIoStatus();
05719
05720 /**
05721  * @ingroup IoControl
05722  * \chinese
05723  * 获取手柄类型
05724  *
05725  * @return type
05726  *
05727  * @throws arcs::common_interface::AuboException
05728  *
05729  * @par Python 函数原型
05730  * getHandleType() -> int
05731  *

```

```

05732      * @par Lua 函数原型
05733      * getHandleType() -> int
05734      *
05735      * @par Lua 示例
05736      * int_num = getHandleType()
05737      *
05738      * @par JSON-RPC 请求示例
05739      * {"jsonrpc": "2.0", "method": "rob1.IoControl.getHandleType", "params": [], "id": 1}
05740      *
05741      * @par JSON-RPC 响应示例
05742      * {"id": 1, "jsonrpc": "2.0", "result": 0}
05743      *
05744      * \endchinese
05745      * \english
05746      * Get the handle type.
05747      *
05748      * @return type
05749      *
05750      * @throws arcs::common_interface::AuboException
05751      *
05752      * @par Python function prototype
05753      * getHandleType() -> int
05754      *
05755      * @par Lua function prototype
05756      * getHandleType() -> int
05757      *
05758      * @par Lua example
05759      * int_num = getHandleType()
05760      *
05761      * @par JSON-RPC request example
05762      * {"jsonrpc": "2.0", "method": "rob1.IoControl.getHandleType", "params": [], "id": 1}
05763      *
05764      * @par JSON-RPC response example
05765      * {"id": 1, "jsonrpc": "2.0", "result": 0}
05766      *
05767      * \endenglish
05768      */
05769      int getHandleType();
05770
05771 protected:
05772     void *d_;
05773 };
05774 using IoControlPtr = std::shared_ptr<IoControl>;
05775 } // namespace common_interface
05776 } // namespace arcs
05777
05778 #endif // AUBO_SDK_IO_CONTROL_INTERFACE_H

```

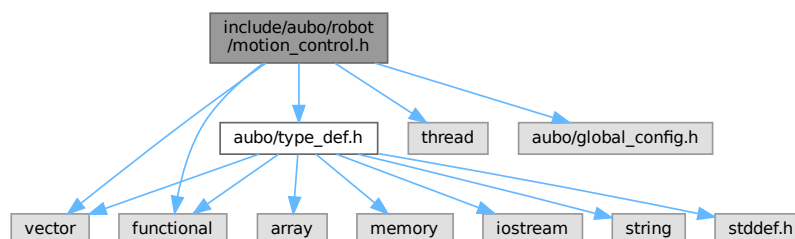
12.24 include/aubo/robot/motion_control.h 文件参考

运动控制接口

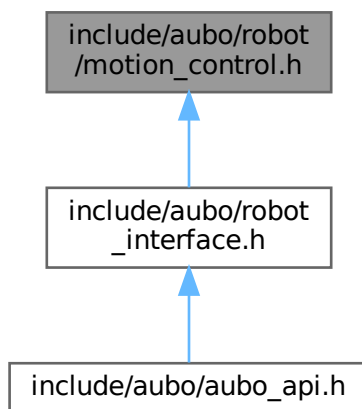
```

#include <vector>
#include <functional>
#include <thread>
#include <aubo/global_config.h>
#include <aubo/type_def.h>
motion_control.h 的引用 (Include) 关系图:

```



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::common_interface::MotionControl](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::common_interface](#)

类型定义

- using [arcs::common_interface::MotionControlPtr](#) = std::shared_ptr<[MotionControl](#)>

枚举

- enum [arcs::common_interface::PathBufferType](#) {
[arcs::common_interface::PathBuffer_TOPPRA](#) = 1 , [arcs::common_interface::PathBuffer_CubicSpline](#)
[arcs::common_interface::PathBuffer_JointSpline](#) = 3 , [arcs::common_interface::PathBuffer_JointSplineC](#)
= 4 ,
[arcs::common_interface::PathBuffer_JointBSpline](#) = 5 , [arcs::common_interface::PathBuffer_JointBSplineC](#)
= 6 }
pathBuffer 类型

12.24.1 详细描述

运动控制接口

在文件 [motion_control.h](#) 中定义.

12.25 motion__control.h

[浏览该文件的文档.](#)

```

00001 /** @file motion_control.h
00002  * \-chinese @brief 运动控制接口
00003  * \-english @brief Motion control interface
00004  */
00005 #ifndef AUBO_SDK_MOTION_CONTROL_INTERFACE_H
00006 #define AUBO_SDK_MOTION_CONTROL_INTERFACE_H
  
```

```

00007
00008 #include <vector>
00009 #include <functional>
00010 #include <thread>
00011
00012 #include <aubo/global_config.h>
00013 #include <aubo/type_def.h>
00014
00015 namespace arcs {
00016 namespace common_interface {
00017
00018 /**
00019  * \chinese
00020  * pathBuffer 类型
00021  *
00022  * 机器人运动被编程为位姿到位姿的运动，即从当前位置移动到新位置。两点之间的路径由机器人自动计算。
00023  * \endchinese
00024  * \english
00025  *
00026  * pathBuffer Type
00027  *
00028  * The robot movements are programmed as pose-to-pose movements, that is move
00029  * from the current position to a new position. The path between these two
00030  * positions is then automatically calculated by the robot.
00031  * \endenglish
00032  */
00033
00034 enum PathBufferType
00035 {
00036     PathBuffer_TOPPPRA = 1, ///< \chinese 1: toppra 时间最优路径规划 \-english
00037                               ///< 1: toppra time optimal path planning
00038     PathBuffer_CubicSpline =
00039         2, ///< \chinese 2: cubic_spline(录制的轨迹) \-english 2: cubic_spline
00040            ///< (recorded trajectory)
00041
00042     ///< \chinese 3: 关节 B 样条插值, 最少三个点 \-english 3: Joint B-spline
00043     ///< interpolation, at least three points
00044     ///< \chinese 废弃, 建议用 5 替代, 现在实际是关节空间 CUBIC_SPLINE \-english
00045     ///< Deprecated, use 5 instead, currently it is joint space CUBIC_SPLINE
00046     PathBuffer_JointSpline = 3,
00047
00048     ///< \chinese 4: 关节 B 样条插值, 最少三个点, 但是传入的是笛卡尔空间位姿
00049     ///< \-english 4: Joint B-spline interpolation, at least three points, but
00050     ///< the input is Cartesian space pose 废弃, 建议用 6 替代, 现在实际是关节空间
00051     ///< CUBIC_SPLINE
00052     PathBuffer_JointSplineC = 4,
00053
00054     ///< \chinese 5: 关节 B 样条插值, 最少三个点 \-english 5: Joint B-spline
00055     ///< interpolation, at least three points
00056     PathBuffer_JointBSpline = 5,
00057
00058     ///< \chinese 6: 关节 B 样条插值, 最少三个点, 但是传入的是笛卡尔空间位姿
00059     ///< \-english 6: Joint B-spline interpolation, at least three points, but
00060     ///< the input is Cartesian space pose
00061     PathBuffer_JointBSplineC = 6,
00062 };
00063
00064 /**
00065  * @defgroup MotionControl MotionControl (运动规划与控制)
00066  * @ingroup RobotInterface
00067  * MotionControl
00068  */
00069 class ARCS_ABI_EXPORT MotionControl
00070 {
00071 public:
00072     MotionControl();
00073     virtual ~MotionControl();
00074
00075     /**
00076      * @ingroup MotionControl
00077      * \chinese
00078      * 获取等效半径, 单位 m
00079      * moveLine/moveCircle 时, 末端姿态旋转的角度等效到末端位置移动
00080      * 可以通过 setEqradius 设置, 默认为 1
00081      *
00082      * @return 返回等效半径
00083      *
00084      * @throws arcs::common_interface::AuboException
00085      *
00086      * @par Python 函数原型
00087      * getEqradius(self: pyaubo_sdk.MotionControl) -> float
00088      *
00089      * @par Lua 函数原型
00090      * getEqradius() -> number
00091      *
00092      * @par Lua 示例
00093      * num = getEqradius()

```

```

00094      *
00095      * @par JSON-RPC 请求示例
00096      * {"jsonrpc": "2.0", "method": "robo1.MotionControl.getEqradius", "params": [], "id": 1}
00097      *
00098      * @par JSON-RPC 响应示例
00099      * {"id": 1, "jsonrpc": "2.0", "result": 1.0}
00100      * \endchinese
00101      * \english
00102      * Get the equivalent radius, in meters.
00103      * When using moveLine/moveCircle, the rotation angle of the end effector's
00104      * pose is converted to an equivalent end position movement. Can be set via
00105      * setEqradius, default is 1.
00106      *
00107      * @return Returns the equivalent radius.
00108      *
00109      * @throws arcs::common_interface::AuboException
00110      *
00111      * @par Python function prototype
00112      * getEqradius(self: pyaubo_sdk.MotionControl) -> float
00113      *
00114      * @par Lua function prototype
00115      * getEqradius() -> number
00116      *
00117      * @par Lua example
00118      * num = getEqradius()
00119      *
00120      * @par JSON-RPC request example
00121      * {"jsonrpc": "2.0", "method": "robo1.MotionControl.getEqradius", "params": [], "id": 1}
00122      *
00123      * @par JSON-RPC response example
00124      * {"id": 1, "jsonrpc": "2.0", "result": 1.0}
00125      * \endenglish
00126      */
00127 double getEqradius();
00128
00129 /**
00130  * @ingroup MotionControl
00131  * \chinese
00132  * 设置等效半径, 单位 m
00133  * moveLine/moveCircle 时, 末端姿态旋转的角度等效到末端位置移动, 数值越大, 姿态旋转速度越快
00134  *
00135  * @param eqradius 0 表示只规划移动, 姿态旋转跟随移动
00136  *
00137  * @return 成功返回 0; 失败返回错误码
00138  * AUBO_BAD_STATE
00139  * AUBO_BUSY
00140  * -AUBO_INVL_ARGUMENT
00141  * -AUBO_BAD_STATE
00142  *
00143  * @throws arcs::common_interface::AuboException
00144  *
00145  * @par JSON-RPC 请求示例
00146  * {"jsonrpc": "2.0", "method": "robo1.MotionControl.setEqradius", "params": [0.8], "id": 1}
00147  *
00148  * @par JSON-RPC 响应示例
00149  * {"id": 1, "jsonrpc": "2.0", "result": 0}
00150  *
00151  * @par Python 函数原型
00152  * setEqradius(self: pyaubo_sdk.MotionControl, arg0: float) -> int
00153  *
00154  * @par Lua 函数原型
00155  * setEqradius(eqradius: number) -> number
00156  *
00157  * @par Lua 示例
00158  * num = setEqradius(1)
00159  *
00160  * \endchinese
00161  * \english
00162  * Set the equivalent radius, in meters.
00163  * When using moveLine/moveCircle, the rotation angle of the end effector's
00164  * pose is converted to an equivalent end position movement. The larger the
00165  * value, the faster the posture rotation speed.
00166  *
00167  * @param eqradius 0 means only plan the movement, and the posture rotation
00168  * follows the movement.
00169  *
00170  * @return Returns 0 on success; otherwise returns an error code:
00171  * AUBO_BAD_STATE
00172  * AUBO_BUSY
00173  * -AUBO_INVL_ARGUMENT
00174  * -AUBO_BAD_STATE
00175  *
00176  * @throws arcs::common_interface::AuboException
00177  *
00178  * @par JSON-RPC request example
00179  * {"jsonrpc": "2.0", "method": "robo1.MotionControl.setEqradius", "params": [0.8], "id": 1}
00180  *

```



```

00181     * @par JSON-RPC response example
00182     * {"id":1,"jsonrpc":"2.0","result":0}
00183     *
00184     * @par Python function prototype
00185     * setEqradius(self: pyaubo_sdk.MotionControl, arg0: float) -> int
00186     *
00187     * @par Lua function prototype
00188     * setEqradius(eqradius: number) -> number
00189     *
00190     * @par Lua example
00191     * num = setEqradius(1)
00192     *
00193     * \endenglish
00194     */
00195 int setEqradius(double eqradius);
00196
00197 /**
00198  * @ingroup MotionControl
00199  * \chinese
00200  * 动态调整机器人运行速度和加速度比例 (0., 1.]
00201  *
00202  * @param fraction 机器人运行速度和加速度比例
00203  * @return 成功返回 0; 失败返回错误码
00204  * AUBO_INVL_ARGUMENT
00205  * AUBO_BUSY
00206  * AUBO_BAD_STATE
00207  * -AUBO_INVL_ARGUMENT
00208  * -AUBO_BAD_STATE
00209  *
00210  * @throws arcs::common_interface::AuboException
00211  *
00212  * @par Python 函数原型
00213  * setSpeedFraction(self: pyaubo_sdk.MotionControl, arg0: float) -> int
00214  *
00215  * @par Lua 函数原型
00216  * setSpeedFraction(fraction: number) -> nil
00217  *
00218  * @par Lua 示例
00219  * setSpeedFraction(0.5)
00220  *
00221  * @par JSON-RPC 请求示例
00222  * {"jsonrpc":"2.0","method":"rob1.MotionControl.setSpeedFraction","params":[0.8],"id":1}
00223  *
00224  * @par JSON-RPC 响应示例
00225  * {"id":1,"jsonrpc":"2.0","result":0}
00226  * \endchinese
00227  * \english
00228  * Dynamically adjust the robot's speed and acceleration ratio (0., 1.]
00229  *
00230  * @param fraction The ratio of robot speed and acceleration
00231  * @return Returns 0 on success; otherwise returns an error code:
00232  * AUBO_INVL_ARGUMENT
00233  * AUBO_BUSY
00234  * AUBO_BAD_STATE
00235  * -AUBO_INVL_ARGUMENT
00236  * -AUBO_BAD_STATE
00237  *
00238  * @throws arcs::common_interface::AuboException
00239  *
00240  * @par Python function prototype
00241  * setSpeedFraction(self: pyaubo_sdk.MotionControl, arg0: float) -> int
00242  *
00243  * @par Lua function prototype
00244  * setSpeedFraction(fraction: number) -> nil
00245  *
00246  * @par Lua example
00247  * setSpeedFraction(0.5)
00248  *
00249  * @par JSON-RPC request example
00250  * {"jsonrpc":"2.0","method":"rob1.MotionControl.setSpeedFraction","params":[0.8],"id":1}
00251  *
00252  * @par JSON-RPC response example
00253  * {"id":1,"jsonrpc":"2.0","result":0}
00254  * \endenglish
00255  */
00256 int setSpeedFraction(double fraction);
00257
00258 /**
00259  * @ingroup MotionControl
00260  * \chinese
00261  * 获取速度和加速度比例, 默认为 1
00262  * 可以通过 setSpeedFraction 接口设置
00263  *
00264  * @return 返回速度和加速度比例
00265  *
00266  * @throws arcs::common_interface::AuboException
00267  *

```

```

00268     * @par Python 函数原型
00269     * getSpeedFraction(self: pyaubo_sdk.MotionControl) -> float
00270     *
00271     * @par Lua 函数原型
00272     * getSpeedFraction() -> number
00273     *
00274     * @par Lua 示例
00275     * num = getSpeedFraction()
00276     *
00277     * @par JSON-RPC 请求示例
00278     * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getSpeedFraction", "params": [], "id": 1}
00279     *
00280     * @par JSON-RPC 响应示例
00281     * {"id": 1, "jsonrpc": "2.0", "result": 1.0}
00282     * \endchinese
00283     * \english
00284     * Get the speed and acceleration ratio, default is 1.
00285     * Can be set via setSpeedFraction interface.
00286     *
00287     * @return Returns the speed and acceleration ratio.
00288     *
00289     * @throws arcs::common_interface::AuboException
00290     *
00291     * @par Python function prototype
00292     * getSpeedFraction(self: pyaubo_sdk.MotionControl) -> float
00293     *
00294     * @par Lua function prototype
00295     * getSpeedFraction() -> number
00296     *
00297     * @par Lua example
00298     * num = getSpeedFraction()
00299     *
00300     * @par JSON-RPC request example
00301     * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getSpeedFraction", "params": [], "id": 1}
00302     *
00303     * @par JSON-RPC response example
00304     * {"id": 1, "jsonrpc": "2.0", "result": 1.0}
00305     * \endenglish
00306     */
00307 double getSpeedFraction();
00308
00309 /**
00310  * @ingroup MotionControl
00311  * \chinese
00312  * 速度比例设置临界区，使能之后速度比例被强制设定为 1.，
00313  * 失能之后恢复之前的速度比例
00314  *
00315  * @param enable
00316  * @return 成功返回 0；失败返回错误码
00317  * AUBO_BAD_STATE
00318  * AUBO_BUSY
00319  * -AUBO_BAD_STATE
00320  *
00321  * @throws arcs::common_interface::AuboException
00322  *
00323  * @par Lua 函数原型
00324  * speedFractionCritical() -> int
00325  *
00326  * @par Lua 示例
00327  * speedFractionCritical(true)
00328  *
00329  * @par JSON-RPC 请求示例
00330  * {"jsonrpc": "2.0", "method": "rob1.MotionControl.speedFractionCritical", "params": [true], "id": 1}
00331  *
00332  * @par JSON-RPC 响应示例
00333  * {"id": 1, "jsonrpc": "2.0", "result": 0}
00334  * \endchinese
00335  * \english
00336  * Speed fraction critical section. When enabled, the speed fraction is
00337  * forced to 1. When disabled, the previous speed fraction is restored.
00338  *
00339  * @param enable
00340  * @return Returns 0 on success; otherwise returns an error code:
00341  * AUBO_BAD_STATE
00342  * AUBO_BUSY
00343  * -AUBO_BAD_STATE
00344  *
00345  * @throws arcs::common_interface::AuboException
00346  *
00347  * @par Lua function prototype
00348  * speedFractionCritical() -> int
00349  *
00350  * @par Lua example
00351  * speedFractionCritical(true)
00352  *
00353  * @par JSON-RPC request example
00354  * {"jsonrpc": "2.0", "method": "rob1.MotionControl.speedFractionCritical", "params": [true], "id": 1}

```

```

00355      *
00356      * @par JSON-RPC response example
00357      * {"id":1,"jsonrpc":"2.0","result":0}
00358      * \endenglish
00359      */
00360      int speedFractionCritical(bool enable);
00361
00362      /**
00363      * @ingroup MotionControl
00364      * \chinese
00365      * 速度比例设置临界区，使能之后速度比例被强制设定为当前生效的比例，
00366      * 失能之后恢复之前的速度比例
00367      *
00368      * @param enable
00369      * @return 成功返回 0；失败返回错误码
00370      * AUBO_BAD_STATE
00371      * AUBO_BUSY
00372      * -AUBO_BAD_STATE
00373      *
00374      * @throws arcs::common_interface::AuboException
00375      *
00376      * @par Lua 函数原型
00377      * speedFractionCritical1() -> int
00378      *
00379      * @par Lua 示例
00380      * speedFractionCritical1(true)
00381      *
00382      * @par JSON-RPC 请求示例
00383      * {"jsonrpc":"2.0","method":"rob1.MotionControl.speedFractionCritical1","params":[true],"id":1}
00384      *
00385      * @par JSON-RPC 响应示例
00386      * {"id":1,"jsonrpc":"2.0","result":0}
00387      * \endchinese
00388      * \english
00389      * Speed fraction critical section. When enabled, the speed fraction is
00390      * forced to the currently effective speed fraction. When disabled, the
00391      * previous speed fraction is restored.
00392      *
00393      * @param enable
00394      * @return Returns 0 on success; otherwise returns an error code:
00395      * AUBO_BAD_STATE
00396      * AUBO_BUSY
00397      * -AUBO_BAD_STATE
00398      *
00399      * @throws arcs::common_interface::AuboException
00400      *
00401      * @par Lua function prototype
00402      * speedFractionCritical1() -> int
00403      *
00404      * @par Lua example
00405      * speedFractionCritical1(true)
00406      *
00407      * @par JSON-RPC request example
00408      * {"jsonrpc":"2.0","method":"rob1.MotionControl.speedFractionCritical1","params":[true],"id":1}
00409      *
00410      * @par JSON-RPC response example
00411      * {"id":1,"jsonrpc":"2.0","result":0}
00412      * \endenglish
00413      */
00414      int speedFractionCritical1(bool enable);
00415
00416      /**
00417      * @ingroup MotionControl
00418      * \chinese
00419      * 是否处于速度比例设置临界区
00420      *
00421      * @return 处于速度比例设置临界区返回 true；反之返回 false
00422      *
00423      * @throws arcs::common_interface::AuboException
00424      *
00425      * @par Lua 函数原型
00426      * isSpeedFractionCritical() -> bool
00427      *
00428      * @par Lua 示例
00429      * status = isSpeedFractionCritical()
00430      *
00431      * @par JSON-RPC 请求示例
00432      * {"jsonrpc":"2.0","method":"rob1.MotionControl.isSpeedFractionCritical","params":[],"id":1}
00433      *
00434      * @par JSON-RPC 响应示例
00435      * {"id":1,"jsonrpc":"2.0","result":true}
00436      * \endchinese
00437      * \english
00438      * Whether it is in the speed fraction critical section
00439      *
00440      * @return Returns true if in the speed fraction critical section; otherwise
00441      * returns false

```

```

00442      *
00443      * @throws arcs::common_interface::AuboException
00444      *
00445      * @par Lua function prototype
00446      * isSpeedFractionCritical() -> bool
00447      *
00448      * @par Lua example
00449      * status = isSpeedFractionCritical()
00450      *
00451      * @par JSON-RPC request example
00452      * {"jsonrpc": "2.0", "method": "rob1.MotionControl.isSpeedFractionCritical", "params": [], "id": 1}
00453      *
00454      * @par JSON-RPC response example
00455      * {"id": 1, "jsonrpc": "2.0", "result": true}
00456      * \endenglish
00457      */
00458 bool isSpeedFractionCritical();
00459
00460 /**
00461  * @ingroup MotionControl
00462  * \chinese
00463  * 是否处交融区
00464  *
00465  * @return 处交融区返回 true; 反之返回 false
00466  *
00467  * @throws arcs::common_interface::AuboException
00468  *
00469  * @par Lua 函数原型
00470  * isBlending() -> bool
00471  *
00472  * @par Lua 示例
00473  * status = isBlending()
00474  *
00475  * @par JSON-RPC 请求示例
00476  * {"jsonrpc": "2.0", "method": "rob1.MotionControl.isBlending", "params": [], "id": 1}
00477  *
00478  * @par JSON-RPC 响应示例
00479  * {"id": 1, "jsonrpc": "2.0", "result": false}
00480  * \endchinese
00481  * \english
00482  * Whether it is in the blending area
00483  *
00484  * @return Returns true if in the blending area; otherwise returns false
00485  *
00486  * @throws arcs::common_interface::AuboException
00487  *
00488  * @par Lua function prototype
00489  * isBlending() -> bool
00490  *
00491  * @par Lua example
00492  * status = isBlending()
00493  *
00494  * @par JSON-RPC request example
00495  * {"jsonrpc": "2.0", "method": "rob1.MotionControl.isBlending", "params": [], "id": 1}
00496  *
00497  * @par JSON-RPC response example
00498  * {"id": 1, "jsonrpc": "2.0", "result": false}
00499  * \endenglish
00500  */
00501 bool isBlending();
00502
00503 /**
00504  * @ingroup MotionControl
00505  * \chinese
00506  * 设置偏移的最大速度和最大加速度
00507  * 仅对 pathOffsetSet 中 type=1 有效
00508  * @param v 最大速度
00509  * @param a 最大加速度
00510  * @return 成功返回 0; 失败返回错误码
00511  * AUBO_BUSY
00512  * AUBO_BAD_STATE
00513  * -AUBO_INVL_ARGUMENT
00514  * -AUBO_BAD_STATE
00515  *
00516  * @throws arcs::common_interface::AuboException
00517  *
00518  * @par Python 函数原型
00519  * pathOffsetLimits(self: pyaubo_sdk.MotionControl, arg0: float, arg1:
00520  * float) -> int
00521  *
00522  * @par Lua 函数原型
00523  * pathOffsetLimits(v: number, a: number) -> nil
00524  *
00525  * @par Lua 示例
00526  * pathOffsetLimits(1.5, 2.5)
00527  *
00528  * \endchinese

```

```

00529     * \english
00530     * Set the maximum speed and maximum acceleration for offset.
00531     * Only valid when type=1 in pathOffsetSet.
00532     * @param v Maximum speed
00533     * @param a Maximum acceleration
00534     * @return Returns 0 on success; otherwise returns an error code:
00535     * AUBO_BUSY
00536     * AUBO_BAD_STATE
00537     * -AUBO_INVL_ARGUMENT
00538     * -AUBO_BAD_STATE
00539     *
00540     * @throws arcs::common_interface::AuboException
00541     *
00542     * @par Python function prototype
00543     * pathOffsetLimits(self: pyaubo_sdk.MotionControl, arg0: float, arg1:
00544     * float) -> int
00545     *
00546     * @par Lua function prototype
00547     * pathOffsetLimits(v: number, a: number) -> nil
00548     *
00549     * @par Lua example
00550     * pathOffsetLimits(1.5,2.5)
00551     *
00552     * \endenglish
00553     */
00554     int pathOffsetLimits(double v, double a);
00555
00556     /**
00557     * @ingroup MotionControl
00558     * \chinese
00559     * 设置偏移的参考坐标系
00560     * 仅对 pathOffsetSet 中 type=1 有效
00561     * @param ref_coord 参考坐标系 0-基坐标系 1-TCP
00562     *
00563     * @throws arcs::common_interface::AuboException
00564     *
00565     * @par Python 函数原型
00566     * pathOffsetCoordinate(self: pyaubo_sdk.MotionControl, arg0: int) -> float
00567     *
00568     * @par Lua 函数原型
00569     * pathOffsetCoordinate(ref_coord: number) -> number
00570     *
00571     * @par Lua 示例
00572     * num = pathOffsetCoordinate(0)
00573     *
00574     * @par JSON-RPC 请求示例
00575     * {"jsonrpc": "2.0", "method": "robl.MotionControl.pathOffsetCoordinate", "params": [0], "id": 1}
00576     *
00577     * @par JSON-RPC 响应示例
00578     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00579     * \endchinese
00580     * \english
00581     * Set the reference coordinate system for offset.
00582     * Only valid when type=1 in pathOffsetSet.
00583     * @param ref_coord Reference coordinate system: 0-base coordinate, 1-TCP
00584     *
00585     * @throws arcs::common_interface::AuboException
00586     *
00587     * @par Python function prototype
00588     * pathOffsetCoordinate(self: pyaubo_sdk.MotionControl, arg0: int) -> float
00589     *
00590     * @par Lua function prototype
00591     * pathOffsetCoordinate(ref_coord: number) -> number
00592     *
00593     * @par Lua example
00594     * num = pathOffsetCoordinate(0)
00595     *
00596     * @par JSON-RPC request example
00597     * {"jsonrpc": "2.0", "method": "robl.MotionControl.pathOffsetCoordinate", "params": [0], "id": 1}
00598     *
00599     * @par JSON-RPC response example
00600     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00601     * \endenglish
00602     */
00603     int pathOffsetCoordinate(int ref_coord);
00604
00605     /**
00606     * @ingroup MotionControl
00607     * \chinese
00608     * 路径偏移使能
00609     *
00610     * @return 成功返回 0; 失败返回错误码
00611     * AUBO_BUSY
00612     * AUBO_BAD_STATE
00613     * -AUBO_BAD_STATE
00614     *
00615     * @throws arcs::common_interface::AuboException

```

```

00616 *
00617 * @par Python 函数原型
00618 * pathOffsetEnable(self: pyaubo_sdk.MotionControl) -> int
00619 *
00620 * @par Lua 函数原型
00621 * pathOffsetEnable() -> number
00622 *
00623 * @par Lua 示例
00624 * num = pathOffsetEnable()
00625 *
00626 * @par JSON-RPC 请求示例
00627 * {"jsonrpc": "2.0", "method": "rob1.MotionControl.pathOffsetEnable", "params": [], "id": 1}
00628 *
00629 * @par JSON-RPC 响应示例
00630 * {"id": 1, "jsonrpc": "2.0", "result": 0}
00631 * \endchinese
00632 * \english
00633 * Enable path offset
00634 *
00635 * @return Returns 0 on success; otherwise returns an error code:
00636 * AUBO_BUSY
00637 * AUBO_BAD_STATE
00638 * -AUBO_BAD_STATE
00639 *
00640 * @throws arcs::common_interface::AuboException
00641 *
00642 * @par Python function prototype
00643 * pathOffsetEnable(self: pyaubo_sdk.MotionControl) -> int
00644 *
00645 * @par Lua function prototype
00646 * pathOffsetEnable() -> number
00647 *
00648 * @par Lua example
00649 * num = pathOffsetEnable()
00650 *
00651 * @par JSON-RPC request example
00652 * {"jsonrpc": "2.0", "method": "rob1.MotionControl.pathOffsetEnable", "params": [], "id": 1}
00653 *
00654 * @par JSON-RPC response example
00655 * {"id": 1, "jsonrpc": "2.0", "result": 0}
00656 * \endenglish
00657 */
00658 int pathOffsetEnable();
00659
00660 /**
00661 * @ingroup MotionControl
00662 * \chinese
00663 * 设置路径偏移
00664 *
00665 * @param offset 在各方向的位姿偏移
00666 * @param type 运动类型 0-位置规划 1-速度规划
00667 * @return 成功返回 0; 失败返回错误码
00668 * AUBO_BAD_STATE
00669 * AUBO_BUSY
00670 * -AUBO_INVL_ARGUMENT
00671 * -AUBO_BAD_STATE
00672 *
00673 * @throws arcs::common_interface::AuboException
00674 *
00675 * @par Python 函数原型
00676 * pathOffsetSet(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
00677 * int) -> int
00678 *
00679 * @par Lua 函数原型
00680 * pathOffsetSet(offset: table, type: number) -> nil
00681 *
00682 * @par Lua 示例
00683 * pathOffsetSet({ 0, 0, 0.1, 0, 0, 0 }, 0)
00684 *
00685 * @par JSON-RPC 请求示例
00686 * {"jsonrpc": "2.0", "method": "rob1.MotionControl.pathOffsetSet", "params": [[0,0,0.01,0,0,0],0], "id": 1}
00687 *
00688 * @par JSON-RPC 响应示例
00689 * {"id": 1, "jsonrpc": "2.0", "result": 0}
00690 * \endchinese
00691 * \english
00692 * Set path offset
00693 *
00694 * @param offset Pose offset in each direction
00695 * @param type Motion type 0-position planning 1-velocity planning
00696 * @return Returns 0 on success; otherwise returns error code
00697 * AUBO_BAD_STATE
00698 * AUBO_BUSY
00699 * -AUBO_INVL_ARGUMENT
00700 * -AUBO_BAD_STATE
00701 *
00702 * @throws arcs::common_interface::AuboException

```

```

00703     *
00704     * @par Python function prototype
00705     * pathOffsetSet(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
00706     * int) -> int
00707     *
00708     * @par Lua function prototype
00709     * pathOffsetSet(offset: table, type: number) -> nil
00710     *
00711     * @par Lua example
00712     * pathOffsetSet({ 0, 0, 0.1, 0, 0, 0 }, 0)
00713     *
00714     * @par JSON-RPC request example
00715     * {"jsonrpc": "2.0", "method": "robl.MotionControl.pathOffsetSet", "params": [[0,0,0.01,0,0,0],0], "id": 1}
00716     *
00717     * @par JSON-RPC response example
00718     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00719     * \endenglish
00720     */
00721 int pathOffsetSet(const std::vector<double> &offset, int type = 0);
00722
00723 /**
00724     * @ingroup MotionControl
00725     * \chinese
00726     * 路径偏移失能
00727     *
00728     * @return 成功返回 0; 失败返回错误码
00729     * AUBO_BAD_STATE
00730     * AUBO_BUSY
00731     * -AUBO_BAD_STATE
00732     *
00733     * @throws arcs::common_interface::AuboException
00734     *
00735     * @par Python 函数原型
00736     * pathOffsetDisable(self: pyaubo_sdk.MotionControl) -> int
00737     *
00738     * @par Lua 函数原型
00739     * pathOffsetDisable() -> nil
00740     *
00741     * @par Lua 示例
00742     * pathOffsetDisable()
00743     *
00744     * @par JSON-RPC 请求示例
00745     * {"jsonrpc": "2.0", "method": "robl.MotionControl.pathOffsetDisable", "params": [], "id": 1}
00746     *
00747     * @par JSON-RPC 响应示例
00748     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00749     * \endchinese
00750     * \english
00751     * Disable path offset
00752     *
00753     * @return Returns 0 on success; otherwise returns an error code:
00754     * AUBO_BAD_STATE
00755     * AUBO_BUSY
00756     * -AUBO_BAD_STATE
00757     *
00758     * @throws arcs::common_interface::AuboException
00759     *
00760     * @par Python function prototype
00761     * pathOffsetDisable(self: pyaubo_sdk.MotionControl) -> int
00762     *
00763     * @par Lua function prototype
00764     * pathOffsetDisable() -> nil
00765     *
00766     * @par Lua example
00767     * pathOffsetDisable()
00768     *
00769     * @par JSON-RPC request example
00770     * {"jsonrpc": "2.0", "method": "robl.MotionControl.pathOffsetDisable", "params": [], "id": 1}
00771     *
00772     * @par JSON-RPC response example
00773     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00774     * \endenglish
00775     */
00776 int pathOffsetDisable();
00777
00778 /**
00779     * @ingroup MotionControl
00780     * \chinese
00781     * @brief 监控轨迹偏移范围
00782     * @param min: 沿坐标轴负方向最大偏移量
00783     * @param max: 沿坐标轴正方向最大偏移量
00784     * @param strategy: 达到最大偏移量后监控策略
00785     * 0-禁用监控
00786     * 1-饱和限制, 即维持最大姿态
00787     * 2-保护停止
00788     * @return
00789     *

```

```

00790     * @throws arcs::common_interface::AuboException
00791     *
00792     * @par Python 函数原型
00793     * pathOffsetSupv(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
00794     * List[float], arg2: int) -> int
00795     *
00796     * @par Lua 函数原型
00797     * pathOffsetSupv(min: table, max: table, strategy: number) -> number
00798     *
00799     * @par Lua 示例
00800     * num = pathOffsetSupv({0,0,-0.2,0,0,0},{0,0,0.5,0,0,0},0)
00801     *
00802     * \endchinese
00803     * \english
00804     * @brief Monitor trajectory offset range
00805     * @param min: Maximum offset in the negative direction of the coordinate
00806     * axis
00807     * @param max: Maximum offset in the positive direction of the coordinate
00808     * axis
00809     * @param strategy: Monitoring strategy after reaching the maximum offset
00810     * 0 - Disable monitoring
00811     * 1 - Saturation limit, i.e., maintain maximum pose
00812     * 2 - Protective stop
00813     * @return
00814     *
00815     * @throws arcs::common_interface::AuboException
00816     *
00817     * @par Python function prototype
00818     * pathOffsetSupv(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
00819     * List[float], arg2: int) -> int
00820     *
00821     * @par Lua function prototype
00822     * pathOffsetSupv(min: table, max: table, strategy: number) -> number
00823     *
00824     * @par Lua example
00825     * num = pathOffsetSupv({0,0,-0.2,0,0,0},{0,0,0.5,0,0,0},0)
00826     *
00827     * \endenglish
00828     */
00829 int pathOffsetSupv(const std::vector<double> &min,
00830                   const std::vector<double> &max, int strategy);
00831
00832 /**
00833  * @ingroup MotionControl
00834  * \chinese
00835  * 关节偏移使能
00836  *
00837  * @return 成功返回 0; 失败返回错误码
00838  * AUBO_BAD_STATE
00839  * AUBO_BUSY
00840  * -AUBO_BAD_STATE
00841  *
00842  * @throws arcs::common_interface::AuboException
00843  *
00844  * @par Python 函数原型
00845  * jointOffsetEnable(self: pyaubo_sdk.MotionControl) -> int
00846  *
00847  * @par Lua 函数原型
00848  * jointOffsetEnable() -> nil
00849  *
00850  * @par Lua 示例
00851  * jointOffsetEnable()
00852  *
00853  * @par JSON-RPC 请求示例
00854  * {"jsonrpc": "2.0", "method": "robl.MotionControl.jointOffsetEnable", "params": [], "id": 1}
00855  *
00856  * @par JSON-RPC 响应示例
00857  * {"id": 1, "jsonrpc": "2.0", "result": 0}
00858  * \endchinese
00859  * \english
00860  * Enable joint offset
00861  *
00862  * @return Returns 0 on success; otherwise returns an error code:
00863  * AUBO_BAD_STATE
00864  * AUBO_BUSY
00865  * -AUBO_BAD_STATE
00866  *
00867  * @throws arcs::common_interface::AuboException
00868  *
00869  * @par Python function prototype
00870  * jointOffsetEnable(self: pyaubo_sdk.MotionControl) -> int
00871  *
00872  * @par Lua function prototype
00873  * jointOffsetEnable() -> nil
00874  *
00875  * @par Lua example
00876  * jointOffsetEnable()

```



```

00877      *
00878      * @par JSON-RPC request example
00879      * {"jsonrpc": "2.0", "method": "rob1.MotionControl.jointOffsetEnable", "params": [], "id": 1}
00880      *
00881      * @par JSON-RPC response example
00882      * {"id": 1, "jsonrpc": "2.0", "result": 0}
00883      * \endenglish
00884      */
00885      int jointOffsetEnable();
00886
00887      /**
00888      * @ingroup MotionControl
00889      * \chinese
00890      * 设置关节偏移
00891      *
00892      * @param offset 在各关节的位姿偏移
00893      * @param type
00894      * @return 成功返回 0; 失败返回错误码
00895      * AUBO_BAD_STATE
00896      * AUBO_BUSY
00897      * -AUBO_INVL_ARGUMENT
00898      * -AUBO_BAD_STATE
00899      *
00900      * @throws arcs::common_interface::AuboException
00901      *
00902      * @par Python 函数原型
00903      * jointOffsetSet(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
00904      * int) -> int
00905      *
00906      * @par Lua 函数原型
00907      * jointOffsetSet(offset: table, type: number) -> nil
00908      *
00909      * @par Lua 示例
00910      * jointOffsetSet({0.1,0,0,0,0,0},1)
00911      *
00912      * @par JSON-RPC 请求示例
00913      * {"jsonrpc": "2.0", "method": "rob1.MotionControl.jointOffsetSet", "params": [[0.1,0,0,0,0],1], "id": 1}
00914      *
00915      * @par JSON-RPC 响应示例
00916      * {"id": 1, "jsonrpc": "2.0", "result": 0}
00917      * \endchinese
00918      * \english
00919      * Set joint offset
00920      *
00921      * @param offset Pose offset for each joint
00922      * @param type
00923      * @return Returns 0 on success; otherwise returns error code
00924      * AUBO_BAD_STATE
00925      * AUBO_BUSY
00926      * -AUBO_INVL_ARGUMENT
00927      * -AUBO_BAD_STATE
00928      *
00929      * @throws arcs::common_interface::AuboException
00930      *
00931      * @par Python function prototype
00932      * jointOffsetSet(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
00933      * int) -> int
00934      *
00935      * @par Lua function prototype
00936      * jointOffsetSet(offset: table, type: number) -> nil
00937      *
00938      * @par Lua example
00939      * jointOffsetSet({0.1,0,0,0,0,0},1)
00940      *
00941      * @par JSON-RPC request example
00942      * {"jsonrpc": "2.0", "method": "rob1.MotionControl.jointOffsetSet", "params": [[0.1,0,0,0,0],1], "id": 1}
00943      *
00944      * @par JSON-RPC response example
00945      * {"id": 1, "jsonrpc": "2.0", "result": 0}
00946      * \endenglish
00947      */
00948      int jointOffsetSet(const std::vector<double> &offset, int type = 1);
00949
00950      /**
00951      * @ingroup MotionControl
00952      * \chinese
00953      * 关节偏移失能
00954      *
00955      * @return 成功返回 0; 失败返回错误码
00956      * AUBO_BAD_STATE
00957      * AUBO_BUSY
00958      * -AUBO_BAD_STATE
00959      *
00960      * @throws arcs::common_interface::AuboException
00961      *
00962      * @par Python 函数原型
00963      * jointOffsetDisable(self: pyaubo_sdk.MotionControl) -> int

```

```

00964 *
00965 * @par Lua 函数原型
00966 * jointOffsetDisable() -> nil
00967 *
00968 * @par Lua 示例
00969 * jointOffsetDisable()
00970 *
00971 * @par JSON-RPC 请求示例
00972 * {"jsonrpc":"2.0","method":"rob1.MotionControl.jointOffsetDisable","params":[],"id":1}
00973 *
00974 * @par JSON-RPC 响应示例
00975 * {"id":1,"jsonrpc":"2.0","result":0}
00976 * \endchinese
00977 * \english
00978 * Disable joint offset
00979 *
00980 * @return Returns 0 on success; otherwise returns an error code:
00981 * AUBO_BAD_STATE
00982 * AUBO_BUSY
00983 * -AUBO_BAD_STATE
00984 *
00985 * @throws arcs::common_interface::AuboException
00986 *
00987 * @par Python function prototype
00988 * jointOffsetDisable(self: pyaubo_sdk.MotionControl) -> int
00989 *
00990 * @par Lua function prototype
00991 * jointOffsetDisable() -> nil
00992 *
00993 * @par Lua example
00994 * jointOffsetDisable()
00995 *
00996 * @par JSON-RPC request example
00997 * {"jsonrpc":"2.0","method":"rob1.MotionControl.jointOffsetDisable","params":[],"id":1}
00998 *
00999 * @par JSON-RPC response example
01000 * {"id":1,"jsonrpc":"2.0","result":0}
01001 * \endenglish
01002 */
01003 int jointOffsetDisable();
01004
01005 /**
01006 * @ingroup MotionControl
01007 * \chinese
01008 * 获取已经入队的指令段 (INST) 数量, 运动指令包括
01009 * moveJoint/moveLine/moveCircle 等运动指令以及 setPayload 等配置指令
01010 *
01011 * 指令一般会在接口宏定义里面用 _INST 指示, 比如 moveJoint
01012 * _INST(MotionControl, 5, moveJoint, q, a, v, blend_radius, duration)
01013 *
01014 * @return 已经入队的指令段数量
01015 *
01016 * @throws arcs::common_interface::AuboException
01017 *
01018 * @par Python 函数原型
01019 * getQueueSize(self: pyaubo_sdk.MotionControl) -> int
01020 *
01021 * @par Lua 函数原型
01022 * getQueueSize() -> number
01023 *
01024 * @par Lua 示例
01025 * num = getQueueSize()
01026 *
01027 * @par JSON-RPC 请求示例
01028 * {"jsonrpc":"2.0","method":"rob1.MotionControl.getQueueSize","params":[],"id":1}
01029 *
01030 * @par JSON-RPC 响应示例
01031 * {"id":1,"jsonrpc":"2.0","result":0}
01032 *
01033 * \endchinese
01034 * \english
01035 * Get the number of enqueued instruction segments (INST), including motion
01036 * instructions such as moveJoint/moveLine/moveCircle and configuration
01037 * instructions such as setPayload.
01038 *
01039 * Instructions are generally indicated with _INST in macro definitions, for
01040 * example: _INST(MotionControl, 5, moveJoint, q, a, v, blend_radius,
01041 * duration)
01042 *
01043 * @return The number of enqueued instruction segments.
01044 *
01045 * @throws arcs::common_interface::AuboException
01046 *
01047 * @par Python function prototype
01048 * getQueueSize(self: pyaubo_sdk.MotionControl) -> int
01049 *
01050 * @par Lua function prototype

```

```

01051     * getQueueSize() -> number
01052     *
01053     * @par Lua example
01054     * num = getQueueSize()
01055     *
01056     * @par JSON-RPC request example
01057     * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getQueueSize", "params": [], "id": 1}
01058     *
01059     * @par JSON-RPC response example
01060     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01061     *
01062     * \endenglish
01063     */
01064 int getQueueSize();
01065
01066 /**
01067  * @ingroup MotionControl
01068  * \chinese
01069  * 获取已经入队的运动规划插补点数量
01070  *
01071  * @return 已经入队的运动规划插补点数量
01072  *
01073  * @throws arcs::common_interface::AuboException
01074  *
01075  * @par Python 函数原型
01076  * getTrajectoryQueueSize(self: pyaubo_sdk.MotionControl) -> int
01077  *
01078  * @par Lua 函数原型
01079  * getTrajectoryQueueSize() -> number
01080  *
01081  * @par Lua 示例
01082  * num = getTrajectoryQueueSize()
01083  *
01084  * @par JSON-RPC 请求示例
01085  * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getTrajectoryQueueSize", "params": [], "id": 1}
01086  *
01087  * @par JSON-RPC 响应示例
01088  * {"id": 1, "jsonrpc": "2.0", "result": 0}
01089  * \endchinese
01090  * \english
01091  * Get the number of enqueued motion planning interpolation points
01092  *
01093  * @return The number of enqueued motion planning interpolation points
01094  *
01095  * @throws arcs::common_interface::AuboException
01096  *
01097  * @par Python function prototype
01098  * getTrajectoryQueueSize(self: pyaubo_sdk.MotionControl) -> int
01099  *
01100  * @par Lua function prototype
01101  * getTrajectoryQueueSize() -> number
01102  *
01103  * @par Lua example
01104  * num = getTrajectoryQueueSize()
01105  *
01106  * @par JSON-RPC request example
01107  * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getTrajectoryQueueSize", "params": [], "id": 1}
01108  *
01109  * @par JSON-RPC response example
01110  * {"id": 1, "jsonrpc": "2.0", "result": 0}
01111  * \endenglish
01112  */
01113 int getTrajectoryQueueSize();
01114
01115 /**
01116  * @ingroup MotionControl
01117  * \chinese
01118  * 获取当前正在插补的运动指令段的 ID
01119  *
01120  * @return 当前正在插补的运动指令段的 ID
01121  * @retval -1 表示轨迹队列为空 \n
01122  * 像 movePathBuffer 运动中的 buffer 或者规划器 (moveJoint 和 moveLine 等) 里的队列都属于轨迹队列
01123  *
01124  * @throws arcs::common_interface::AuboException
01125  *
01126  * @par Python 函数原型
01127  * getExecId(self: pyaubo_sdk.MotionControl) -> int
01128  *
01129  * @par Lua 函数原型
01130  * getExecId() -> number
01131  *
01132  * @par Lua 示例
01133  * num = getExecId()
01134  *
01135  * @par JSON-RPC 请求示例
01136  * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getExecId", "params": [], "id": 1}
01137  *

```

```

01138     * @par JSON-RPC 响应示例
01139     * {"id":1,"jsonrpc":"2.0","result":-1}
01140     *
01141     * \endchinese
01142     * \english
01143     * Get the ID of the currently interpolating motion instruction segment.
01144     *
01145     * @return The ID of the currently interpolating motion instruction segment.
01146     * @retval -1 Indicates the trajectory queue is empty. \n
01147     * Both the buffer in movePathBuffer motion and the queue in the planner
01148     * (such as moveJoint and moveLine) belong to the trajectory queue.
01149     *
01150     * @throws arcs::common_interface::AuboException
01151     *
01152     * @par Python function prototype
01153     * getExecId(self: pyaubo_sdk.MotionControl) -> int
01154     *
01155     * @par Lua function prototype
01156     * getExecId() -> number
01157     *
01158     * @par Lua example
01159     * num = getExecId()
01160     *
01161     * @par JSON-RPC request example
01162     * {"jsonrpc":"2.0","method":"rob1.MotionControl.getExecId","params":[],"id":1}
01163     *
01164     * @par JSON-RPC response example
01165     * {"id":1,"jsonrpc":"2.0","result":-1}
01166     *
01167     * \endenglish
01168     */
01169     int getExecId();
01170
01171     /**
01172     * @ingroup MotionControl
01173     * \chinese
01174     * 获取指定 ID 的运动指令段的预期执行时间
01175     *
01176     * @param id 运动指令段 ID
01177     * @return 返回预期执行时间
01178     *
01179     * @throws arcs::common_interface::AuboException
01180     *
01181     * @par Python 函数原型
01182     * getDuration(self: pyaubo_sdk.MotionControl, arg0: int) -> float
01183     *
01184     * @par Lua 函数原型
01185     * getDuration(id: number) -> number
01186     *
01187     * @par Lua 示例
01188     * num = getDuration(16)
01189     *
01190     * @par JSON-RPC 请求示例
01191     * {"jsonrpc":"2.0","method":"rob1.MotionControl.getDuration","params":[16],"id":1}
01192     *
01193     * @par JSON-RPC 响应示例
01194     * {"id":1,"jsonrpc":"2.0","result":0.0}
01195     *
01196     * \endchinese
01197     * \english
01198     * Get the expected execution duration of the motion segment with the
01199     * specified ID.
01200     *
01201     * @param id Motion segment ID
01202     * @return Returns the expected execution duration
01203     *
01204     * @throws arcs::common_interface::AuboException
01205     *
01206     * @par Python function prototype
01207     * getDuration(self: pyaubo_sdk.MotionControl, arg0: int) -> float
01208     *
01209     * @par Lua function prototype
01210     * getDuration(id: number) -> number
01211     *
01212     * @par Lua example
01213     * num = getDuration(16)
01214     *
01215     * @par JSON-RPC request example
01216     * {"jsonrpc":"2.0","method":"rob1.MotionControl.getDuration","params":[16],"id":1}
01217     *
01218     * @par JSON-RPC response example
01219     * {"id":1,"jsonrpc":"2.0","result":0.0}
01220     *
01221     * \endenglish
01222     */
01223     double getDuration(int id);
01224

```

```

01225  /**
01226   * @ingroup MotionControl
01227   * \chinese
01228   * 获取指定 ID 的运动指令段的剩余执行时间
01229   *
01230   * @param id 运动指令段 ID
01231   * @return 返回剩余执行时间
01232   *
01233   * @throws arcs::common_interface::AuboException
01234   *
01235   * @par Python 函数原型
01236   * getMotionLeftTime(self: pyaubo_sdk.MotionControl, arg0: int) -> float
01237   *
01238   * @par Lua 函数原型
01239   * getMotionLeftTime(id: number) -> number
01240   *
01241   * @par Lua 示例
01242   * num = getMotionLeftTime(16)
01243   *
01244   * @par JSON-RPC 请求示例
01245   * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getMotionLeftTime", "params": [16], "id": 1}
01246   *
01247   * @par JSON-RPC 响应示例
01248   * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
01249   * \endchinese
01250   * \english
01251   * Get the remaining execution time of the motion segment with the specified
01252   * ID.
01253   *
01254   * @param id Motion segment ID
01255   * @return Returns the remaining execution time
01256   *
01257   * @throws arcs::common_interface::AuboException
01258   *
01259   * @par Python function prototype
01260   * getMotionLeftTime(self: pyaubo_sdk.MotionControl, arg0: int) -> float
01261   *
01262   * @par Lua function prototype
01263   * getMotionLeftTime(id: number) -> number
01264   *
01265   * @par Lua example
01266   * num = getMotionLeftTime(16)
01267   *
01268   * @par JSON-RPC request example
01269   * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getMotionLeftTime", "params": [16], "id": 1}
01270   *
01271   * @par JSON-RPC response example
01272   * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
01273   * \endenglish
01274   */
01275 double getMotionLeftTime(int id);
01276
01277  /**
01278   * @ingroup MotionControl
01279   * \chinese
01280   * StopMove 用于临时停止机器人和外部轴的运动以及相关工艺进程。如果调用
01281   * StartMove 指令，则运动和工艺进程将恢复。
01282   *
01283   * 该指令可用于中断处理程序中，在发生中断时临时停止机器人。
01284   *
01285   * @param quick true: 以最快速度在路径上停止机器人。未指定 quick
01286   * 参数时，机器人将在路径上停止，但制动距离较长（与普通程序停止相同）。
01287   *
01288   * @return 成功返回 0；失败返回错误码
01289   * AUBO_BAD_STATE
01290   * AUBO_BUSY
01291   * -AUBO_BAD_STATE
01292   *
01293   * @throws arcs::common_interface::AuboException
01294   *
01295   * @par Lua 函数原型
01296   * StopMove(quick: bool, all_tasks: bool) -> number
01297   *
01298   * @par Lua 示例
01299   * num = StopMove(true, true)
01300   *
01301   * \endchinese
01302   * \english
01303   * StopMove is used to stop robot and external axes movements and any
01304   * belonging process temporarily. If the instruction StartMove is given then
01305   * the movement and process resumes.
01306   *
01307   * This instruction can, for example, be used in a trap routine to stop the
01308   * robot temporarily when an interrupt occurs.
01309   *
01310   * @param quick true: Stops the robot on the path as fast as possible.
01311   * Without the optional parameter \Quick, the robot stops on the path, but

```

```

01312     * the braking distance is longer (same as for normal Program Stop).
01313     *
01314     * @return Returns 0 on success; otherwise returns an error code:
01315     * AUBO_BAD_STATE
01316     * AUBO_BUSY
01317     * -AUBO_BAD_STATE
01318     *
01319     * @throws arcs::common_interface::AuboException
01320     *
01321     * @par Lua function prototype
01322     * StopMove(quick: bool,all_tasks: bool) -> number
01323     *
01324     * @par Lua example
01325     * num = StopMove(true,true)
01326     *
01327     * \endenglish
01328     */
01329 int stopMove(bool quick, bool all_tasks);
01330
01331 /**
01332  * @ingroup MotionControl
01333  * \chinese
01334  * StartMove 用于在以下情况下恢复机器人、外部轴的运动以及相关工艺进程:
01335  * • 通过 StopMove 指令停止后。
01336  * • 执行 StorePath ... RestoPath 序列后。
01337  * • 发生异步运动错误 (如 ERR_PATH_STOP) 或特定工艺错误并在 ERROR
01338  * 处理器中处理后。
01339  *
01340  * @return 成功返回 0; 失败返回错误码
01341  * AUBO_BAD_STATE
01342  * AUBO_BUSY
01343  * -AUBO_BAD_STATE
01344  *
01345  * @throws arcs::common_interface::AuboException
01346  *
01347  * @par Lua 函数原型
01348  * startMove() -> number
01349  *
01350  * @par Lua 示例
01351  * num = startMove()
01352  *
01353  * @par JSON-RPC 请求示例
01354  * {"jsonrpc":"2.0","method":"rob1.MotionControl.startMove","params":[],"id":1}
01355  *
01356  * @par JSON-RPC 响应示例
01357  * {"id":1,"jsonrpc":"2.0","result":0}
01358  * \endchinese
01359  * \english
01360  * StartMove is used to resume robot, external axes movement and belonging
01361  * process after the movement has been stopped
01362  *
01363  * • by the instruction StopMove.
01364  * • after execution of StorePath ... RestoPath sequence.
01365  * • after asynchronously raised movements errors, such as ERR_PATH_STOP or
01366  * specific process error after handling in the ERROR handler.
01367  *
01368  * @return Returns 0 on success; otherwise returns an error code:
01369  * AUBO_BAD_STATE
01370  * AUBO_BUSY
01371  * -AUBO_BAD_STATE
01372  *
01373  * @throws arcs::common_interface::AuboException
01374  *
01375  * @par Lua function prototype
01376  * startMove() -> number
01377  *
01378  * @par Lua example
01379  * num = startMove()
01380  *
01381  * @par JSON-RPC request example
01382  * {"jsonrpc":"2.0","method":"rob1.MotionControl.startMove","params":[],"id":1}
01383  *
01384  * @par JSON-RPC response example
01385  * {"id":1,"jsonrpc":"2.0","result":0}
01386  * \endenglish
01387  */
01388 int startMove();
01389
01390 /**
01391  * @ingroup MotionControl
01392  * \chinese
01393  * storePath
01394  *
01395  * @param keep_sync
01396  * @return 成功返回 0; 失败返回错误码
01397  * AUBO_BAD_STATE
01398  * AUBO_BUSY

```

```

01399      * -AUBO_BAD_STATE
01400      *
01401      * @throws arcs::common_interface::AuboException
01402      *
01403      * @par Lua 函数原型
01404      * storePath(keep_sync: bool) -> number
01405      *
01406      * @par Lua 示例
01407      * num = storePath()
01408      *
01409      * \endchinese
01410      * \english
01411      * storePath
01412      *
01413      * @param keep_sync
01414      * @return Returns 0 on success; otherwise returns an error code:
01415      * AUBO_BAD_STATE
01416      * AUBO_BUSY
01417      * -AUBO_BAD_STATE
01418      *
01419      * @throws arcs::common_interface::AuboException
01420      *
01421      * @par Lua function prototype
01422      * storePath(keep_sync: bool) -> number
01423      *
01424      * @par Lua example
01425      * num = storePath()
01426      *
01427      * \endenglish
01428      */
01429 int storePath(bool keep_sync);
01430
01431 /**
01432  * @ingroup MotionControl
01433  * \chinese
01434  * ClearPath (清除路径) 清除当前运动路径层级（基础层级或 StorePath
01435  * 层级）上的所有运动路径。
01436  *
01437  * @return 成功返回 0；失败返回错误码
01438  * AUBO_BAD_STATE
01439  * AUBO_BUSY
01440  * -AUBO_BAD_STATE
01441  *
01442  * @throws arcs::common_interface::AuboException
01443  *
01444  * @par Lua 函数原型
01445  * clearPath() -> number
01446  *
01447  * @par Lua 示例
01448  * num = clearPath()
01449  *
01450  * @par JSON-RPC 请求示例
01451  * {"jsonrpc": "2.0", "method": "robl.MotionControl.clearPath", "params": [], "id": 1}
01452  *
01453  * @par JSON-RPC 响应示例
01454  * {"id": 1, "jsonrpc": "2.0", "result": 0}
01455  *
01456  * \endchinese
01457  * \english
01458  * ClearPath clears the whole motion path on the current motion path level
01459  * (base level or StorePath level).
01460  *
01461  * @return Returns 0 on success; otherwise returns an error code:
01462  * AUBO_BAD_STATE
01463  * AUBO_BUSY
01464  * -AUBO_BAD_STATE
01465  *
01466  * @throws arcs::common_interface::AuboException
01467  *
01468  * @par Lua function prototype
01469  * clearPath() -> number
01470  *
01471  * @par Lua example
01472  * num = clearPath()
01473  *
01474  * @par JSON-RPC request example
01475  * {"jsonrpc": "2.0", "method": "robl.MotionControl.clearPath", "params": [], "id": 1}
01476  *
01477  * @par JSON-RPC response example
01478  * {"id": 1, "jsonrpc": "2.0", "result": 0}
01479  *
01480  * \endenglish
01481  */
01482 int clearPath();
01483
01484 /**
01485  * @ingroup MotionControl

```

```

01486     * \chinese
01487     * restoPath
01488     *
01489     * @return 成功返回 0; 失败返回错误码
01490     * AUBO_BAD_STATE
01491     * AUBO_BUSY
01492     * -AUBO_BAD_STATE
01493     *
01494     * @throws arcs::common_interface::AuboException
01495     *
01496     * @par Lua 函数原型
01497     * restoPath() -> number
01498     *
01499     * @par Lua 示例
01500     * num = restoPath()
01501     *
01502     * @par JSON-RPC 请求示例
01503     * {"jsonrpc": "2.0", "method": "rob1.MotionControl.restoPath", "params": [], "id": 1}
01504     *
01505     * @par JSON-RPC 响应示例
01506     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01507     * \endchinese
01508     * \english
01509     * restoPath
01510     *
01511     * @return Returns 0 on success; otherwise returns an error code:
01512     * AUBO_BAD_STATE
01513     * AUBO_BUSY
01514     * -AUBO_BAD_STATE
01515     *
01516     * @throws arcs::common_interface::AuboException
01517     *
01518     * @par Lua function prototype
01519     * restoPath() -> number
01520     *
01521     * @par Lua example
01522     * num restoPath()
01523     *
01524     * @par JSON-RPC request example
01525     * {"jsonrpc": "2.0", "method": "rob1.MotionControl.restoPath", "params": [], "id": 1}
01526     *
01527     * @par JSON-RPC response example
01528     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01529     * \endenglish
01530     */
01531 int restoPath();
01532
01533 /**
01534     * @ingroup MotionControl
01535     * \chinese
01536     * 获取当前运动指令段的执行进度
01537     *
01538     * @return 返回执行进度
01539     *
01540     * @throws arcs::common_interface::AuboException
01541     *
01542     * @par Python 函数原型
01543     * getProgress(self: pyaubo_sdk.MotionControl) -> float
01544     *
01545     * @par Lua 函数原型
01546     * getProgress() -> number
01547     *
01548     * @par Lua 示例
01549     * num = getProgress()
01550     *
01551     * @par JSON-RPC 请求示例
01552     * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getProgress", "params": [], "id": 1}
01553     *
01554     * @par JSON-RPC 响应示例
01555     * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
01556     *
01557     * \endchinese
01558     * \english
01559     * Get the execution progress of the current motion instruction segment.
01560     *
01561     * @return Returns the execution progress.
01562     *
01563     * @throws arcs::common_interface::AuboException
01564     *
01565     * @par Python function prototype
01566     * getProgress(self: pyaubo_sdk.MotionControl) -> float
01567     *
01568     * @par Lua function prototype
01569     * getProgress() -> number
01570     *
01571     * @par Lua example
01572     * num = getProgress()

```



```

01573      *
01574      * @par JSON-RPC request example
01575      * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getProgress", "params": [], "id": 1}
01576      *
01577      * @par JSON-RPC response example
01578      * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
01579      *
01580      * \endenglish
01581      */
01582 double getProgress();
01583
01584 /**
01585  * @ingroup MotionControl
01586  * \chinese
01587  * 当工件安装在另外一台机器人的末端或者外部轴上时, 指定其名字和安装位置
01588  *
01589  * @note 暂未实现
01590  *
01591  * @param module_name 控制模块名字
01592  * @param mounting_pose 抓取的相对位置,
01593  * 如果是机器人, 则相对于机器人末端中心点 (非 TCP 点)
01594  * @return 成功返回 0; 失败返回错误码
01595  * AUBO_BAD_STATE
01596  * AUBO_BUSY
01597  * -AUBO_INVL_ARGUMENT
01598  * -AUBO_BAD_STATE
01599  *
01600  * @throws arcs::common_interface::AuboException
01601  *
01602  * @par Python 函数原型
01603  * setWorkObjectHold(self: pyaubo_sdk.MotionControl, arg0: str, arg1:
01604  * List[float]) -> int
01605  *
01606  * @par Lua 函数原型
01607  * setWorkObjectHold(module_name: string, mounting_pose: table) -> nil
01608  *
01609  * @par Lua 示例
01610  * setWorkObjectHold("object", {0.2, 0.1, -0.4, 3.14, 0, -1.57})
01611  *
01612  * \endchinese
01613  * \english
01614  * Specify the name and mounting position when the workpiece is installed on
01615  * the end of another robot or external axis.
01616  *
01617  * @note Not implemented yet
01618  *
01619  * @param module_name Name of the control module
01620  * @param mounting_pose Relative position of the grasp,
01621  * if it is a robot, it is relative to the robot's end center point (not the
01622  * TCP point)
01623  * @return Returns 0 on success; otherwise returns an error code:
01624  * AUBO_BAD_STATE
01625  * AUBO_BUSY
01626  * -AUBO_INVL_ARGUMENT
01627  * -AUBO_BAD_STATE
01628  *
01629  * @throws arcs::common_interface::AuboException
01630  *
01631  * @par Python function prototype
01632  * setWorkObjectHold(self: pyaubo_sdk.MotionControl, arg0: str, arg1:
01633  * List[float]) -> int
01634  *
01635  * @par Lua function prototype
01636  * setWorkObjectHold(module_name: string, mounting_pose: table) -> nil
01637  *
01638  * @par Lua example
01639  * setWorkObjectHold("object", {0.2, 0.1, -0.4, 3.14, 0, -1.57})
01640  *
01641  * \endenglish
01642  */
01643 int setWorkObjectHold(const std::string &module_name,
01644                      const std::vector<double> &mounting_pose);
01645
01646 /**
01647  * @ingroup MotionControl
01648  * \chinese
01649  * 获取工件安装信息
01650  *
01651  * @note 暂未实现
01652  *
01653  * @return 返回一个包含控制模块名字和安装位姿的元组
01654  *
01655  * @throws arcs::common_interface::AuboException
01656  *
01657  * @par Python 函数原型
01658  * getWorkObjectHold(self: pyaubo_sdk.MotionControl) -> Tuple[str,
01659  * List[float]]

```

```

01660      *
01661      * @par Lua 函数原型
01662      * getWorkObjectHold() -> table
01663      *
01664      * @par Lua 示例
01665      * Object_table = getWorkObjectHold()
01666      *
01667      * @par JSON-RPC 请求示例
01668      * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getWorkObjectHold", "params": [], "id": 1}
01669      *
01670      * @par JSON-RPC 响应示例
01671      * {"id": 1, "jsonrpc": "2.0", "result": ["", []]}
01672      * \endchinese
01673      * \english
01674      * getWorkObjectHold
01675      *
01676      * @note Not implemented yet
01677      *
01678      * @return Returns a tuple containing the control module name and mounting
01679      * pose
01680      *
01681      * @throws arcs::common_interface::AuboException
01682      *
01683      * @par Python function prototype
01684      * getWorkObjectHold(self: pyaubo_sdk.MotionControl) -> Tuple[str,
01685      * List[float]]
01686      *
01687      * @par Lua function prototype
01688      * getWorkObjectHold() -> table
01689      *
01690      * @par Lua example
01691      * Object_table = getWorkObjectHold()
01692      *
01693      * @par JSON-RPC request example
01694      * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getWorkObjectHold", "params": [], "id": 1}
01695      *
01696      * @par JSON-RPC response example
01697      * {"id": 1, "jsonrpc": "2.0", "result": ["", []]}
01698      * \endenglish
01699      */
01700      std::tuple<std::string, std::vector<double>> getWorkObjectHold();
01701
01702      /**
01703      * @ingroup MotionControl
01704      * \chinese
01705      * @note 获取暂停点关节位置
01706      *
01707      * 常用于运行工程时发生碰撞后继续运动过程中（先通过 resumeMoveJoint 或 resumeMoveLine 运动到暂停位置，再恢复工程）
01708      *
01709      * @return 暂停关节位置
01710      *
01711      * @throws arcs::common_interface::AuboException
01712      *
01713      * @par Python 函数原型
01714      * getPauseJointPositions(self: pyaubo_sdk.MotionControl) -> List[float]
01715      *
01716      * @par Lua 函数原型
01717      * getPauseJointPositions() -> table
01718      *
01719      * @par Lua 示例
01720      * JointPositions = getPauseJointPositions()
01721      *
01722      * @par JSON-RPC 请求示例
01723      * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getPauseJointPositions", "params": [], "id": 1}
01724      *
01725      * @par JSON-RPC 响应示例
01726      * {"id": 1, "jsonrpc": "2.0", "result": [8.2321e-13, -0.200999, 1.33999, 0.334999, 1.206, -6.39383e-12]}
01727      * \endchinese
01728      * \english
01729      * @note Get the joint positions at the pause point.
01730      *
01731      * Commonly used during process recovery after a collision occurs (first
01732      * move to the pause position using resumeMoveJoint or resumeMoveLine, then
01733      * resume the process).
01734      *
01735      * @return Pause joint positions
01736      *
01737      * @throws arcs::common_interface::AuboException
01738      *
01739      * @par Python function prototype
01740      * getPauseJointPositions(self: pyaubo_sdk.MotionControl) -> List[float]
01741      *
01742      * @par Lua function prototype
01743      * getPauseJointPositions() -> table
01744      *
01745      * @par Lua example
01746      * JointPositions = getPauseJointPositions()

```

```

01747      *
01748      * @par JSON-RPC request example
01749      * {"jsonrpc": "2.0", "method": "robl.MotionControl.getPauseJointPositions", "params": [], "id": 1}
01750      *
01751      * @par JSON-RPC response example
01752      * {"id": 1, "jsonrpc": "2.0", "result": [8.2321e-13, -0.200999, 1.33999, 0.334999, 1.206, -6.39383e-12]}
01753      * \endenglish
01754      */
01755      std::vector<double> getPauseJointPositions();
01756
01757      /**
01758      * @ingroup MotionControl
01759      * \chinese
01760      * 设置继续运动参数
01761      *
01762      * @param q 继续运动起始位置
01763      * @param move_type 0: 关节空间 1: 笛卡尔空间
01764      * @param blend_radius 交融半径
01765      * @param qdmax 关节运动最大速度 (6 维度数据)
01766      * @param qddmax 关节运动最大加速度 (6 维度数据)
01767      * @param vmax 直线运动最大线速度, 角速度 (2 维度数据)
01768      * @param amax 直线运动最大线加速度, 角加速度 (2 维度数据)
01769      * @return 成功返回 0; 失败返回错误码
01770      * AUBO_BAD_STATE
01771      * AUBO_BUSY
01772      * -AUBO_BAD_STATE
01773      *
01774      * @throws arcs::common_interface::AuboException
01775      *
01776      * @par Python 函数原型
01777      * setResumeStartPoint(self: pyaubo_sdk.MotionControl,
01778      * arg0: List[float], arg1: int, arg2: float, arg3: List[float], arg4:
01779      * List[float], arg5: float, arg6: float) -> int
01780      *
01781      * @par Lua 函数原型
01782      * setResumeStartPoint(q : table, move_type: number, blend_radius:
01783      * number, qdmax: table, qddmax:
01784      * table, vmax: number, amax: number) -> nilr
01785      *
01786      * @par Lua 示例
01787      * setResumeStartPoint({0,0,0,0,0,0},1,1,{0,0,0,0,0,0},{0,0,0,0,0,0},{1,1},{1,1})
01788      *
01789      * @par JSON-RPC 请求示例
01790      * {"jsonrpc": "2.0", "method": "robl.MotionControl.setResumeStartPoint", "params": [[0,0,0,0,0,0],1,1,[0,0,0,0,0,0],
01791      * [0,0,0,0,0,0],1,1],"id":1}
01792      *
01793      * @par JSON-RPC 响应示例
01794      * {"id":1,"jsonrpc":"2.0","result":0}
01795      * \endchinese
01796      * \english
01797      * Set resume motion parameters
01798      *
01799      * @param q Resume start position
01800      * @param move_type 0: Joint space, 1: Cartesian space
01801      * @param blend_radius Blend radius
01802      * @param qdmax Maximum joint speed (6 elements)
01803      * @param qddmax Maximum joint acceleration (6 elements)
01804      * @param vmax Maximum linear and angular speed for linear motion (2
01805      * elements)
01806      * @param amax Maximum linear and angular acceleration for linear motion (2
01807      * elements)
01808      * @return Returns 0 on success; otherwise returns error code
01809      * AUBO_BAD_STATE
01810      * AUBO_BUSY
01811      * -AUBO_BAD_STATE
01812      *
01813      * @throws arcs::common_interface::AuboException
01814      *
01815      * @par Python function prototype
01816      * setResumeStartPoint(self: pyaubo_sdk.MotionControl,
01817      * arg0: List[float], arg1: int, arg2: float, arg3: List[float], arg4:
01818      * List[float], arg5: float, arg6: float) -> int
01819      *
01820      * @par Lua function prototype
01821      * setResumeStartPoint(q: table, move_type: number, blend_radius:
01822      * number, qdmax: table, qddmax:
01823      * table, vmax: number, amax: number) -> nil
01824      *
01825      * @par Lua example
01826      * setResumeStartPoint({0,0,0,0,0,0},1,1,{0,0,0,0,0,0},{0,0,0,0,0,0},{1,1},{1,1})
01827      *
01828      * @par JSON-RPC request example
01829      * {"jsonrpc": "2.0", "method": "robl.MotionControl.setResumeStartPoint", "params": [[0,0,0,0,0,0],1,1,[0,0,0,0,0,0],
01830      * [0,0,0,0,0,0],1,1],"id":1}
01831      *
01832      * @par JSON-RPC response example
01833      * {"id":1,"jsonrpc":"2.0","result":0}

```

```

01832     * \endenglish
01833     */
01834 int setResumeStartPoint(const std::vector<double> &q, int move_type,
01835                        double blend_radius,
01836                        const std::vector<double> &qdmax,
01837                        const std::vector<double> &qddmax,
01838                        const std::vector<double> &vmax,
01839                        const std::vector<double> &amax);
01840 /**
01841  * @ingroup MotionControl
01842  * \chinese
01843  * 获取继续运动模式
01844  *
01845  * @return 0: 继续运动起始点为暂停点 1: 继续运动起始点为指定点
01846  *
01847  * @throws arcs::common_interface::AuboException
01848  *
01849  * @par Python 函数原型
01850  * getResumeMode(self: pyaubo_sdk.MotionControl) -> int
01851  *
01852  * @par Lua 函数原型
01853  * getResumeMode() -> int
01854  *
01855  * @par Lua 示例
01856  * num = getResumeMode()
01857  *
01858  * @par JSON-RPC 请求示例
01859  * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getResumeMode", "params": [], "id": 1}
01860  *
01861  * @par JSON-RPC 响应示例
01862  * {"id": 1, "jsonrpc": "2.0", "result": false}
01863  *
01864  * \endchinese
01865  * \english
01866  * Get resume motion mode
01867  *
01868  * @return 0: Resume start point is the pause point; 1: Resume start point
01869  * is the specified point
01870  *
01871  * @throws arcs::common_interface::AuboException
01872  *
01873  * @par Python function prototype
01874  * getResumeMode(self: pyaubo_sdk.MotionControl) -> int
01875  *
01876  * @par Lua function prototype
01877  * getResumeMode() -> int
01878  *
01879  * @par Lua example
01880  * num = getResumeMode()
01881  *
01882  * @par JSON-RPC request example
01883  * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getResumeMode", "params": [], "id": 1}
01884  *
01885  * @par JSON-RPC response example
01886  * {"id": 1, "jsonrpc": "2.0", "result": false}
01887  *
01888  * \endenglish
01889  */
01890 int getResumeMode();
01891
01892 /**
01893  * @ingroup MotionControl
01894  * \chinese
01895  * 设置伺服模式
01896  * 使用 setServoModeSelect 替代
01897  *
01898  * @param enable 是否使能
01899  * @return 成功返回 0; 失败返回错误码
01900  * AUBO_BAD_STATE
01901  * AUBO_BUSY
01902  * -AUBO_BAD_STATE
01903  *
01904  * @throws arcs::common_interface::AuboException
01905  *
01906  * @par Python 函数原型
01907  * setServoMode(self: pyaubo_sdk.MotionControl, arg0: bool) -> int
01908  *
01909  * @par Lua 函数原型
01910  * setServoMode(enable: boolean) -> nil
01911  *
01912  * @par Lua 示例
01913  * setServoMode(true)
01914  *
01915  * @par JSON-RPC 请求示例
01916  * {"jsonrpc": "2.0", "method": "rob1.MotionControl.setServoMode", "params": [true], "id": 1}
01917  *
01918  * @par JSON-RPC 响应示例

```

```

01919     * {"id":1,"jsonrpc":"2.0","result":0}
01920     * \endchinese
01921     * \english
01922     * Set servo mode
01923     * Use setServoModeSelect instead
01924     *
01925     * @param enable Whether to enable
01926     * @return Returns 0 on success; otherwise returns an error code:
01927     * AUBO_BAD_STATE
01928     * AUBO_BUSY
01929     * -AUBO_BAD_STATE
01930     *
01931     * @throws arcs::common_interface::AuboException
01932     *
01933     * @par Python function prototype
01934     * setServoMode(self: pyaubo_sdk.MotionControl, arg0: bool) -> int
01935     *
01936     * @par Lua function prototype
01937     * setServoMode(enable: boolean) -> nil
01938     *
01939     * @par Lua example
01940     * setServoMode(true)
01941     *
01942     * @par JSON-RPC request example
01943     * {"jsonrpc":"2.0","method":"robl.MotionControl.setServoMode","params":[true],"id":1}
01944     *
01945     * @par JSON-RPC response example
01946     * {"id":1,"jsonrpc":"2.0","result":0}
01947     * \endenglish
01948     */
01949 ARCS_DEPRECATED int setServoMode(bool enable);
01950
01951 /**
01952  * @ingroup MotionControl
01953  * \chinese
01954  * 判断伺服模式是否使能
01955  * 使用 getServoModeSelect 替代
01956  *
01957  * @return 已使能返回 true, 反之则返回 false
01958  *
01959  * @throws arcs::common_interface::AuboException
01960  *
01961  * @par Python 函数原型
01962  * isServoModeEnabled(self: pyaubo_sdk.MotionControl) -> bool
01963  *
01964  * @par Lua 函数原型
01965  * isServoModeEnabled() -> boolean
01966  *
01967  * @par Lua 示例
01968  * Servo_status = isServoModeEnabled()
01969  *
01970  * @par JSON-RPC 请求示例
01971  * {"jsonrpc":"2.0","method":"robl.MotionControl.isServoModeEnabled","params":[],"id":1}
01972  *
01973  * @par JSON-RPC 响应示例
01974  * {"id":1,"jsonrpc":"2.0","result":false}
01975  *
01976  * \endchinese
01977  * \english
01978  * Determine whether the servo mode is enabled.
01979  * Use getServoModeSelect instead.
01980  *
01981  * @return Returns true if enabled, otherwise returns false.
01982  *
01983  * @throws arcs::common_interface::AuboException
01984  *
01985  * @par Python function prototype
01986  * isServoModeEnabled(self: pyaubo_sdk.MotionControl) -> bool
01987  *
01988  * @par Lua function prototype
01989  * isServoModeEnabled() -> boolean
01990  *
01991  * @par Lua example
01992  * Servo_status = isServoModeEnabled()
01993  *
01994  * @par JSON-RPC request example
01995  * {"jsonrpc":"2.0","method":"robl.MotionControl.isServoModeEnabled","params":[],"id":1}
01996  *
01997  * @par JSON-RPC response example
01998  * {"id":1,"jsonrpc":"2.0","result":false}
01999  *
02000  * \endenglish
02001  */
02002 ARCS_DEPRECATED bool isServoModeEnabled();
02003
02004 /**
02005  * @ingroup MotionControl

```

```

02006 * \chinese
02007 * 设置伺服运动模式
02008 *
02009 * @param mode
02010 * 0-退出伺服模式
02011 * 1-(截断式) 规划伺服模式
02012 * 2-透传模式 (直接下发)
02013 * 3-透传模式 (缓存)
02014 * 4-1ms 透传模式 (缓存)
02015 * 5-规划伺服模式
02016 * 6-(截断式) 规划伺服模式, 可以叠加力控
02017 * 7-规划伺服模式, 可以叠加力控
02018 * 伺服模式 1 添加路点后, 会实时调整目标点和规划路线 (当前的目标点被更新后, 不能保证达到之前设定的目标点)
02019 * 伺服模式 5 添加路点后, 能保证经过所有目标点
02020 * @return
02021 *
02022 * @par Lua 函数原型
02023 * setServoModeSelect(0) -> nil
02024 *
02025 * @par Lua 示例
02026 * setServoModeSelect(0)
02027 *
02028 * @par JSON-RPC 请求示例
02029 * {"jsonrpc":"2.0","method":"rob1.MotionControl.setServoModeSelect","params":[0],"id":1}
02030 *
02031 * @par JSON-RPC 响应示例
02032 * {"id":1,"jsonrpc":"2.0","result":0}
02033 *
02034 * \endchinese
02035 * \english
02036 * Set servo motion mode
02037 *
02038 * @param mode
02039 * 0 - Exit servo mode
02040 * 1 - Planning servo mode
02041 * 2 - Transparent mode (direct send)
02042 * 3 - Transparent mode (buffered)
02043 * @return
02044 *
02045 * @par Lua function prototype
02046 * setServoModeSelect(0) -> nil
02047 *
02048 * @par Lua example
02049 * setServoModeSelect(0)
02050 *
02051 * @par JSON-RPC request example
02052 * {"jsonrpc":"2.0","method":"rob1.MotionControl.setServoModeSelect","params":[0],"id":1}
02053 *
02054 * @par JSON-RPC response example
02055 * {"id":1,"jsonrpc":"2.0","result":0}
02056 *
02057 * \endenglish
02058 */
02059 int setServoModeSelect(int mode);
02060
02061 /**
02062 * @ingroup MotionControl
02063 * \chinese
02064 * 获取伺服运动模式
02065 *
02066 * @return
02067 *
02068 * @par Lua 函数原型
02069 * getServoModeSelect() -> number
02070 *
02071 * @par Lua 示例
02072 * num = getServoModeSelect()
02073 *
02074 * @par JSON-RPC 请求示例
02075 * {"jsonrpc":"2.0","method":"rob1.MotionControl.getServoModeSelect","params":[],"id":1}
02076 *
02077 * @par JSON-RPC 响应示例
02078 * {"id":1,"jsonrpc":"2.0","result":0}
02079 *
02080 * \endchinese
02081 * \english
02082 * Get the servo motion mode
02083 *
02084 * @return
02085 *
02086 * @par Lua function prototype
02087 * getServoModeSelect() -> number
02088 *
02089 * @par Lua example
02090 * num = getServoModeSelect()
02091 *
02092 * @par JSON-RPC request example

```

```

02093     * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getServoModeSelect", "params": [], "id": 1}
02094     *
02095     * @par JSON-RPC response example
02096     * {"id": 1, "jsonrpc": "2.0", "result": 0}
02097     *
02098     * \endenglish
02099     */
02100 int getServoModeSelect();
02101
02102 /**
02103  * @ingroup MotionControl
02104  * \chinese
02105  * 关节空间伺服
02106  *
02107  * 目前可用参数只有 q 和 t;
02108  * @param q 关节角, 单位 rad,
02109  * @param a 加速度, 单位 rad/s^2,
02110  * @param v 速度, 单位 rad/s,
02111  * @param t 运行时间, 单位 s \n
02112  * t 值越大, 机器臂运动越慢, 反之, 运动越快;
02113  * 该参数最优值为连续调用 servoJoint 接口的间隔时间。
02114  * @param lookahead_time 前瞻时间, 单位 s \n
02115  * 指定机器臂开始减速前要运动的时长, 用前瞻时间来平滑轨迹 [0.03, 0.2],
02116  * 当 lookahead_time 小于一个控制周期时, 越小则超调量越大,
02117  * 该参数最优值为一个控制周期。
02118  * @param gain 比例增益
02119  * 跟踪目标位置的比例增益 [100, 200],
02120  * 用于控制运动的平滑性和精度,
02121  * 比例增益越大, 到达目标位置的时间越长, 超调量越小。
02122  *
02123  * @retval 0 成功
02124  * @retval AUBO_BAD_STATE(1) 当前安全模式处于非
02125  * Normal、ReducedMode、Recovery 状态
02126  * @retval AUBO_QUEUE_FULL(2) 规划队列已满
02127  * @retval AUBO_BUSY(3) 上一条指令正在执行中
02128  * @retval -AUBO_BAD_STATE(-1)
02129  * 可能的原因包括但不限于: 线程已分离、线程被终止、task_id
02130  * 未找到, 或者当前机器人模式非 Running
02131  * @retval -AUBO_TIMEOUT(-4) 调用接口超时
02132  * @retval -AUBO_INVL_ARGUMENT(-5) 轨迹位置超限、速度超限或相邻点角度差超限 (±15°)
02133  * @retval -AUBO_REQUEST_IGNORE(-13) 当前处于非 servo 模式
02134  * @retval -AUBO_POS_BOUND_ERR(-18) 关节位置超出机器人最大限制
02135  * @retval -AUBO_TRAJ_SELF_COLLISION(-22) 轨迹存在自碰撞风险
02136  * @retval -AUBO_ALGORITHM_PLAN_FAILED(-30) 轨迹规划失败
02137  *
02138  * @throws arcs::common_interface::AuboException
02139  *
02140  * @par Python 函数原型
02141  * servoJoint(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
02142  * float, arg2: float, arg3: float, arg4: float, arg5: float) -> int
02143  *
02144  * @par Lua 函数原型
02145  * servoJoint(q: table, a: number, v: number, t: number, lookahead_time:
02146  * number, gain: number) -> nil
02147  *
02148  * @par Lua 示例
02149  * servoJoint({0,0,0,0,0,0},0,0,10,0,0)
02150  *
02151  * @par JSON-RPC 请求示例
02152  * {"jsonrpc": "2.0", "method": "rob1.MotionControl.servoJoint", "params": [[0,0,0,0,0,0],0,0,10,0,0], "id": 1}
02153  *
02154  * @par JSON-RPC 响应示例
02155  * {"id": 1, "jsonrpc": "2.0", "result": -13}
02156  *
02157  * \endchinese
02158  * \english
02159  * Joint space servo
02160  *
02161  * Currently only q and t parameters are available;
02162  * @param q Joint angles, unit: rad
02163  * @param a Acceleration, unit: rad/s^2
02164  * @param v Velocity, unit: rad/s
02165  * @param t Execution time, unit: s \n
02166  * The larger the t value, the slower the robot moves, and vice versa;
02167  * The optimal value for this parameter is the interval time for continuous
02168  * calls to the servoJoint interface.
02169  * @param lookahead_time Lookahead time, unit: s \n
02170  * Specifies the duration to move before the robot starts to decelerate,
02171  * used to smooth the trajectory [0.03, 0.2]. When lookahead_time is less
02172  * than one control cycle, the smaller it is, the greater the overshoot. The
02173  * optimal value for this parameter is one control cycle.
02174  * @param gain Proportional gain
02175  * Proportional gain for tracking the target position [100, 200],
02176  * used to control the smoothness and accuracy of the motion.
02177  * The larger the proportional gain, the longer it takes to reach the target
02178  * position, and the smaller the overshoot.
02179  *

```



```

02180 * @retval 0 Success
02181 * @retval AUBO_BAD_STATE(1) The current safety mode is not Normal,
02182 * ReducedMode, or Recovery
02183 * @retval AUBO_QUEUE_FULL(2) Planning queue is full
02184 * @retval AUBO_BUSY(3) The previous instruction is being executed
02185 * @retval -AUBO_BAD_STATE(-1)
02186 * Possible reasons include but are not limited to: thread has been
02187 * detached, thread terminated, task_id not found, or the current robot mode
02188 * is not Running
02189 * @retval -AUBO_TIMEOUT(-4) Interface call timeout
02190 * @retval -AUBO_INVL_ARGUMENT(-5) Trajectory position out of range, velocity
02191 * out of range, or adjacent points angle difference exceeded( $\pm 15^\circ$ )
02192 * @retval -AUBO_REQUEST_IGNORE(-13) Not in servo mode
02193 * @retval -AUBO_POS_BOUND_ERR(-18) Joint position exceeds robot maximum limits
02194 * @retval -AUBO_TRAJ_SELF_COLLISION(-22) Trajectory has self-collision risk
02195 * @retval -AUBO_ALGORITHM_PLAN_FAILED(-30) Trajectory planning failed
02196 *
02197 * @throws arcs::common_interface::AuboException
02198 *
02199 * @par Python function prototype
02200 * servoJoint(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
02201 * float, arg2: float, arg3: float, arg4: float, arg5: float) -> int
02202 *
02203 * @par Lua function prototype
02204 * servoJoint(q: table, a: number, v: number, t: number, lookahead_time:
02205 * number, gain: number) -> nil
02206 *
02207 * @par Lua example
02208 * servoJoint({0,0,0,0,0,0},0,0,10,0,0)
02209 *
02210 * @par JSON-RPC request example
02211 * {"jsonrpc":"2.0","method":"robl.MotionControl.servoJoint","params":[[0,0,0,0,0,0],0,0,10,0,0],"id":1}
02212 *
02213 * @par JSON-RPC response example
02214 * {"id":1,"jsonrpc":"2.0","result":-13}
02215 *
02216 * \endenglish
02217 */
02218 int servoJoint(const std::vector<double> &q, double a, double v, double t,
02219 double lookahead_time, double gain);
02220
02221 /**
02222 * @ingroup MotionControl
02223 * \chinese
02224 * 笛卡尔空间伺服
02225 *
02226 * 目前可用参数只有 pose 和 t;
02227 * @param pose 位姿, 单位 m,
02228 * @param a 加速度, 单位  $m/s^2$ ,
02229 * @param v 速度, 单位  $m/s$ ,
02230 * @param t 运行时间, 单位 s \n
02231 * t 值越大, 机器臂运动越慢, 反之, 运动越快;
02232 * 该参数最优值为连续调用 servoCartesian 接口的间隔时间。
02233 * @param lookahead_time 前瞻时间, 单位 s \n
02234 * 指定机器臂开始减速前要运动的时长, 用前瞻时间来平滑轨迹 [0.03, 0.2],
02235 * 当 lookahead_time 小于一个控制周期时, 越小则超调量越大,
02236 * 该参数最优值为一个控制周期。
02237 * @param gain 比例增益
02238 * 跟踪目标位置的比例增益 [100, 200],
02239 * 用于控制运动的顺滑性和精度,
02240 * 比例增益越大, 到达目标位置的时间越长, 超调量越小。
02241 *
02242 * @retval 0 成功
02243 * @retval AUBO_BAD_STATE(1) 当前安全模式处于非
02244 * Normal、ReducedMode、Recovery 状态
02245 * @retval AUBO_QUEUE_FULL(2) 规划队列已满
02246 * @retval AUBO_BUSY(3) 上一条指令正在执行中
02247 * @retval -AUBO_BAD_STATE(-1)
02248 * 可能的原因包括但不限于: 线程已分离、线程被终止、task_id
02249 * 未找到, 或者当前机器人模式非 Running
02250 * @retval -AUBO_TIMEOUT(-4) 调用接口超时
02251 * @retval -AUBO_INVL_ARGUMENT(-5) 轨迹位置超限或速度超限
02252 * @retval -AUBO_REQUEST_IGNORE(-13) 当前处于非 servo 模式
02253 * @retval -AUBO_IK_NO_CONVERGE(-23) 逆解计算不收敛, 计算出错;
02254 * @retval -AUBO_IK_OUT_OF_RANGE(-24) 逆解计算超出机器人最大限制;
02255 * @retval AUBO_IK_CONFIG_DISMATCH(-25) 逆解输入配置存在错误;
02256 * @retval -AUBO_IK_JACOBIAN_FAILED(-26) 逆解雅可比矩阵计算失败;
02257 * @retval -AUBO_IK_NO_SOLU(-27) 目标点存在解析解, 但均不满足选解条件;
02258 * @retval -AUBO_IK_UNKOWN_ERROR(-28) 逆解返回未知类型错误;
02259 *
02260 * @throws arcs::common_interface::AuboException
02261 *
02262 * @par Python 函数原型
02263 * servoCartesian(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
02264 * float, arg2: float, arg3: float, arg4: float, arg5: float) -> int
02265 *
02266 * @par Lua 函数原型

```



```

02267 * servoCartesian(pose: table, a: number, v: number, t: number,
02268 * lookahead_time: number, gain: number) -> nil
02269 *
02270 * @par Lua 示例
02271 * servoCartesian({0.58712,-0.15775,0.48703,2.76,0.344,1.432},0,0,10,0,0)
02272 *
02273 * @par JSON-RPC 请求示例
02274 * {"jsonrpc":"2.0","method":"rob1.MotionControl.servoCartesian","params":
[[0.58712,-0.15775,0.48703,2.76,0.344,1.432],0,0,10,0,0],"id":1}
02275 *
02276 * @par JSON-RPC 响应示例
02277 * {"id":1,"jsonrpc":"2.0","result":-13}
02278 * \endchinese
02279 * \english
02280 * Cartesian space servo
02281 *
02282 * Currently, only pose and t parameters are available;
02283 * @param pose Pose, unit: m
02284 * @param a Acceleration, unit: m/s^2
02285 * @param v Velocity, unit: m/s
02286 * @param t Execution time, unit: s \n
02287 * The larger the t value, the slower the robot moves, and vice versa;
02288 * The optimal value for this parameter is the interval time for continuous
02289 * calls to the servoCartesian interface.
02290 * @param lookahead_time Lookahead time, unit: s \n
02291 * Specifies the duration to move before the robot starts to decelerate,
02292 * used to smooth the trajectory [0.03, 0.2]. When lookahead_time is less
02293 * than one control cycle, the smaller it is, the greater the overshoot. The
02294 * optimal value for this parameter is one control cycle.
02295 * @param gain Proportional gain
02296 * Proportional gain for tracking the target position [100, 200],
02297 * used to control the smoothness and accuracy of the motion,
02298 * The larger the proportional gain, the longer it takes to reach the target
02299 * position, and the smaller the overshoot.
02300 *
02301 * @retval 0 Success
02302 * @retval AUBO_BAD_STATE(1) The current safety mode is not Normal,
02303 * ReducedMode, or Recovery
02304 * @retval AUBO_QUEUE_FULL(2) Planning queue is full
02305 * @retval AUBO_BUSY(3) The previous instruction is being executed
02306 * @retval -AUBO_BAD_STATE(-1)
02307 * Possible reasons include but are not limited to: thread has been
02308 * detached, thread terminated, task_id not found, or the current robot mode
02309 * is not Running
02310 * @retval -AUBO_TIMEOUT(-4) Interface call timeout
02311 * @retval -AUBO_INVL_ARGUMENT(-5) Trajectory position or velocity out of
02312 * range
02313 * @retval -AUBO_REQUEST_IGNORE(-13) Not in servo mode
02314 * @retval -AUBO_IK_NO_CONVERGE(-23) Inverse kinematics calculation does not
02315 * converge, calculation error;
02316 * @retval -AUBO_IK_OUT_OF_RANGE(-24) Inverse kinematics calculation exceeds
02317 * robot maximum limits;
02318 * @retval AUBO_IK_CONFIG_DISMATH(-25) Inverse kinematics input
02319 * configuration error;
02320 * @retval -AUBO_IK_JACOBIAN_FAILED(-26) Inverse kinematics Jacobian
02321 * calculation failed;
02322 * @retval -AUBO_IK_NO_SOLU(-27) Analytical solution exists for the target
02323 * point, but none meet the selection criteria;
02324 * @retval -AUBO_IK_UNKNOWN_ERROR(-28) Inverse kinematics returned an unknown
02325 * error type;
02326 *
02327 * @throws arcs::common_interface::AuboException
02328 *
02329 * @par Python function prototype
02330 * servoCartesian(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
02331 * float, arg2: float, arg3: float, arg4: float, arg5: float) -> int
02332 *
02333 * @par Lua function prototype
02334 * servoCartesian(pose: table, a: number, v: number, t: number,
02335 * lookahead_time: number, gain: number) -> nil
02336 *
02337 * @par Lua example
02338 * servoCartesian({0.58712,-0.15775,0.48703,2.76,0.344,1.432},0,0,10,0,0)
02339 *
02340 * @par JSON-RPC request example
02341 * {"jsonrpc":"2.0","method":"rob1.MotionControl.servoCartesian","params":
[[0.58712,-0.15775,0.48703,2.76,0.344,1.432],0,0,10,0,0],"id":1}
02342 *
02343 * @par JSON-RPC response example
02344 * {"id":1,"jsonrpc":"2.0","result":-13}
02345 * \endenglish
02346 */
02347 int servoCartesian(const std::vector<double> &pose, double a, double v,
02348 double t, double lookahead_time, double gain);
02349
02350 /**
02351 * @ingroup MotionControl

```

```

02352     * \chinese
02353     * 伺服运动（带外部轴），用于执行离线轨迹、透传用户规划轨迹等
02354     *
02355     * @param q
02356     * @param extq
02357     * @param t
02358     * @param smooth_scale
02359     * @param delay_sacle
02360     * @return
02361     * \endchinese
02362     * \english
02363     * Servo motion (with external axes), used for executing offline
02364     * trajectories, pass-through user planned trajectories, etc.
02365     *
02366     * @param q
02367     * @param extq
02368     * @param t
02369     * @param smooth_scale
02370     * @param delay_sacle
02371     * @return
02372     * \endenglish
02373     */
02374     int servoJointWithAxes(const std::vector<double> &q,
02375                           const std::vector<double> &extq, double a, double v,
02376                           double t, double lookahead_time, double gain);
02377
02378     int servoJointWithAxisGroup(const std::vector<double> &q, double a,
02379                                double v, double t, double lookahead_time,
02380                                double gain, const std::string &group_name,
02381                                const std::vector<double> &extq);
02382
02383     /**
02384     * @ingroup MotionControl
02385     * \chinese
02386     * 伺服运动（带外部轴），用于执行离线轨迹、透传用户规划轨迹等
02387     * 与 servoJointWithAxes 区别在于接收笛卡尔空间位姿而不是关节角度
02388     * （由软件内部直接做逆解）
02389     *
02390     * @param pose
02391     * @param extq
02392     * @param t
02393     * @param smooth_scale
02394     * @param delay_sacle
02395     * @return
02396     * \endchinese
02397     * \english
02398     * Servo motion (with external axes), used for executing offline
02399     * trajectories, pass-through user planned trajectories, etc. The difference
02400     * from servoJointWithAxes is that it receives Cartesian poses instead of
02401     * joint angles (inverse kinematics is performed internally by the
02402     * software).
02403     *
02404     * @param pose
02405     * @param extq
02406     * @param t
02407     * @param smooth_scale
02408     * @param delay_sacle
02409     * @return
02410     * \endenglish
02411     */
02412     int servoCartesianWithAxes(const std::vector<double> &pose,
02413                                const std::vector<double> &extq, double a,
02414                                double v, double t, double lookahead_time,
02415                                double gain);
02416
02417     int servoCartesianWithAxisGroup(const std::vector<double> &pose, double a,
02418                                     double v, double t, double lookahead_time,
02419                                     double gain, const std::string &group_name,
02420                                     const std::vector<double> &extq);
02421
02422     /**
02423     * @ingroup MotionControl
02424     * \chinese
02425     * 跟踪运动，用于执行离线轨迹、透传用户规划轨迹等
02426     *
02427     * @param q
02428     * @param smooth_scale
02429     * @param delay_sacle
02430     * @return
02431     *
02432     * @par Lua 函数原型
02433     * trackJoint(q: table,t: number,smooth_scale: number,delay_sacle: number)
02434     * -> nil
02435     *
02436     * @par Lua 示例
02437     * trackJoint({0,0,0,0,0,0},0.01,0.5,1)
02438     *

```

```

02439      * @par JSON-RPC 请求示例
02440      * {"jsonrpc": "2.0", "method": "rob1.MotionControl.trackJoint", "params": [[0,0,0,0,0,0],0.01,0.5,1], "id": 1}
02441      *
02442      * @par JSON-RPC 响应示例
02443      * {"id": 1, "jsonrpc": "2.0", "result": 0}
02444      *
02445      * \endchinese
02446      * \english
02447      * Tracking motion, used for executing offline trajectories or passing
02448      * through user-planned trajectories, etc.
02449      *
02450      * @param q
02451      * @param smooth_scale
02452      * @param delay_sacle
02453      * @return
02454      *
02455      * @par Lua function prototype
02456      * trackJoint(q: table, t: number, smooth_scale: number, delay_sacle: number)
02457      * -> nil
02458      *
02459      * @par Lua example
02460      * trackJoint({0,0,0,0,0,0},0.01,0.5,1)
02461      *
02462      * @par JSON-RPC request example
02463      * {"jsonrpc": "2.0", "method": "rob1.MotionControl.trackJoint", "params": [[0,0,0,0,0,0],0.01,0.5,1], "id": 1}
02464      *
02465      * @par JSON-RPC response example
02466      * {"id": 1, "jsonrpc": "2.0", "result": 0}
02467      *
02468      * \endenglish
02469      */
02470      int trackJoint(const std::vector<double> &q, double t, double smooth_scale,
02471                    double delay_sacle);
02472
02473      /**
02474      * @ingroup MotionControl
02475      * \chinese
02476      * 跟踪运动，用于执行离线轨迹、透传用户规划轨迹等
02477      * 与 trackJoint 区别在于接收笛卡尔空间位姿而不是关节角度
02478      * （由软件内部直接做逆解）
02479      *
02480      * @param pose
02481      * @param t
02482      * @param smooth_scale
02483      * @param delay_sacle
02484      * @return
02485      *
02486      * @par Lua 函数原型
02487      * trackCartesian(pose: table, t: number, smooth_scale: number, delay_sacle:
02488      * number) -> nil
02489      *
02490      * @par Lua 示例
02491      * trackCartesian({0.58712,-0.15775,0.48703,2.76,0.344,1.432},0.01,0.5,1)
02492      *
02493      * @par JSON-RPC 请求示例
02494      * {"jsonrpc": "2.0", "method": "rob1.MotionControl.trackCartesian", "params":
02495      * [[0.58712,-0.15775,0.48703,2.76,0.344,1.432],0.01,0.5,1], "id": 1}
02496      *
02497      * @par JSON-RPC 响应示例
02498      * {"id": 1, "jsonrpc": "2.0", "result": 0}
02499      * \endchinese
02500      * \english
02501      * Tracking motion, used for executing offline trajectories or passing
02502      * through user-planned trajectories, etc. The difference from trackJoint is
02503      * that it receives Cartesian poses instead of joint angles (inverse
02504      * kinematics is performed internally by the software).
02505      *
02506      * @param pose
02507      * @param t
02508      * @param smooth_scale
02509      * @param delay_sacle
02510      * @return
02511      *
02512      * @par Lua function prototype
02513      * trackCartesian(pose: table, t: number, smooth_scale: number, delay_sacle:
02514      * number) -> nil
02515      *
02516      * @par Lua example
02517      * trackCartesian({0.58712,-0.15775,0.48703,2.76,0.344,1.432},0.01,0.5,1)
02518      *
02519      * @par JSON-RPC request example
02520      * {"jsonrpc": "2.0", "method": "rob1.MotionControl.trackCartesian", "params":
02521      * [[0.58712,-0.15775,0.48703,2.76,0.344,1.432],0.01,0.5,1], "id": 1}
02522      *
02523      * @par JSON-RPC response example
02524      * {"id": 1, "jsonrpc": "2.0", "result": 0}
02525      * \endenglish

```

```

02524     */
02525 int trackCartesian(const std::vector<double> &pose, double t,
02526                   double smooth_scale, double delay_sacle);
02527
02528 /**
02529  * @ingroup MotionControl
02530  * \chinese
02531  * 关节空间跟随
02532  *
02533  * @note 暂未实现
02534  *
02535  * @throws arcs::common_interface::AuboException
02536  *
02537  * @par Python 函数原型
02538  * followJoint(self: pyaubo_sdk.MotionControl, arg0: List[float]) -> int
02539  *
02540  * @par Lua 函数原型
02541  * followJoint(q: table) -> nil
02542  *
02543  * @par Lua 示例
02544  * followJoint({0,0,0,0,0,0})
02545  *
02546  * \endchinese
02547  * \english
02548  * Joint space following
02549  *
02550  * @note Not implemented yet
02551  *
02552  * @throws arcs::common_interface::AuboException
02553  *
02554  * @par Python function prototype
02555  * followJoint(self: pyaubo_sdk.MotionControl, arg0: List[float]) -> int
02556  *
02557  * @par Lua function prototype
02558  * followJoint(q: table) -> nil
02559  *
02560  * @par Lua example
02561  * followJoint({0,0,0,0,0,0})
02562  *
02563  * \endenglish
02564  */
02565 int followJoint(const std::vector<double> &q);
02566
02567 /**
02568  * @ingroup MotionControl
02569  * \chinese
02570  * 笛卡尔空间跟随
02571  *
02572  * @note 暂未实现
02573  *
02574  * @throws arcs::common_interface::AuboException
02575  *
02576  * @par Python 函数原型
02577  * followLine(self: pyaubo_sdk.MotionControl, arg0: List[float]) -> int
02578  *
02579  * @par Lua 函数原型
02580  * followLine(pose: table) -> nil
02581  *
02582  * @par Lua 示例
02583  * followLine({0.58712,-0.15775,0.48703,2.76,0.344,1.432})
02584  *
02585  * \endchinese
02586  * \english
02587  * Cartesian space following
02588  *
02589  * @note Not implemented yet
02590  *
02591  * @throws arcs::common_interface::AuboException
02592  *
02593  * @par Python function prototype
02594  * followLine(self: pyaubo_sdk.MotionControl, arg0: List[float]) -> int
02595  *
02596  * @par Lua function prototype
02597  * followLine(pose: table) -> nil
02598  *
02599  * @par Lua example
02600  * followLine({0.58712,-0.15775,0.48703,2.76,0.344,1.432})
02601  *
02602  * \endenglish
02603  */
02604 int followLine(const std::vector<double> &pose);
02605
02606 /**
02607  * @ingroup MotionControl
02608  * \chinese
02609  * 关节空间速度跟随
02610  *

```

```

02611 * 当机械臂还没达到目标速度的时候，给一个新的目标速度，机械臂会立刻达到新的目标速度
02612 *
02613 * @param qd 目标关节速度，单位 rad/s
02614 * @param a 主轴的加速度，单位 rad/s^2
02615 * @param t 函数返回所需要的时间，单位 s \n
02616 * 如果 t = 0，当达到目标速度的时候，函数将返回；
02617 * 反之，则经过 t 时间后，函数返回，不管是否达到目标速度。 \n
02618 * 如果没有达到目标速度，会减速到零。
02619 * 如果达到了目标速度就是按照目标速度匀速运动。
02620 * @retval 0 成功
02621 * @retval AUBO_BAD_STATE(1) 当前安全模式处于非
02622 * Normal、ReducedMode、Recovery 状态
02623 * @retval AUBO_BUSY(3) 上一条指令正在执行中
02624 * @retval -AUBO_BAD_STATE(-1)
02625 * 可能的原因包括但不限于：线程已分离、线程被终止、task_id
02626 * 未找到，或者当前机器人模式非 Running
02627 * @retval -AUBO_TIMEOUT(-4) 调用接口超时
02628 * @retval -AUBO_INVL_ARGUMENT(-5) 参数数组 qd 的长度小于当前机器臂的自由度
02629 *
02630 * @throws arcs::common_interface::AuboException
02631 *
02632 * @par Python 函数原型
02633 * speedJoint(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
02634 * float, arg2: float) -> int
02635 *
02636 * @par Lua 函数原型
02637 * speedJoint(qd: table, a: number, t: number) -> nil
02638 *
02639 * @par Lua 示例
02640 * speedJoint({0.2,0,0,0,0,0}, 1.5,10)
02641 *
02642 * @par JSON-RPC 请求示例
02643 * {"jsonrpc":"2.0","method":"rob1.MotionControl.speedJoint","params":[[0.2,0,0,0,0,0],1.5,100],"id":1}
02644 *
02645 * @par JSON-RPC 响应示例
02646 * {"id":1,"jsonrpc":"2.0","result":0}
02647 *
02648 * \endchinese
02649 * \english
02650 * Joint space velocity following
02651 *
02652 * When the robot arm has not yet reached the target velocity, giving a new
02653 * target velocity will cause the robot arm to immediately reach the new
02654 * target velocity.
02655 *
02656 * @param qd Target joint velocity, unit: rad/s
02657 * @param a Main axis acceleration, unit: rad/s^2
02658 * @param t Time required for the function to return, unit: s \n
02659 * If t = 0, the function returns when the target velocity is reached;
02660 * otherwise, the function returns after t seconds, regardless of whether
02661 * the target velocity is reached. \n If the target velocity is not reached,
02662 * it will decelerate to zero. If the target velocity is reached, it will
02663 * move at a constant speed.
02664 * @retval 0 Success
02665 * @retval AUBO_BAD_STATE(1) The current safety mode is not Normal,
02666 * ReducedMode, or Recovery
02667 * @retval AUBO_BUSY(3) The previous instruction is being executed
02668 * @retval -AUBO_BAD_STATE(-1)
02669 * Possible reasons include but are not limited to: thread has been
02670 * detached, thread terminated, task_id not found, or the current robot mode
02671 * is not Running
02672 * @retval -AUBO_TIMEOUT(-4) Interface call timeout
02673 * @retval -AUBO_INVL_ARGUMENT(-5) The length of the qd array is less than
02674 * the degrees of freedom of the current robot arm
02675 *
02676 * @throws arcs::common_interface::AuboException
02677 *
02678 * @par Python function prototype
02679 * speedJoint(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
02680 * float, arg2: float) -> int
02681 *
02682 * @par Lua function prototype
02683 * speedJoint(qd: table, a: number, t: number) -> nil
02684 *
02685 * @par Lua example
02686 * speedJoint({0.2,0,0,0,0,0}, 1.5,10)
02687 *
02688 * @par JSON-RPC request example
02689 * {"jsonrpc":"2.0","method":"rob1.MotionControl.speedJoint","params":[[0.2,0,0,0,0,0],1.5,100],"id":1}
02690 *
02691 * @par JSON-RPC response example
02692 * {"id":1,"jsonrpc":"2.0","result":0}
02693 *
02694 * \endenglish
02695 */
02696 int speedJoint(const std::vector<double> &qd, double a, double t);
02697

```

```

02698 /**
02699  * @ingroup MotionControl
02700  * \chinese
02701  * 关节空间速度跟随（机械臂运行工程时发生碰撞，通过此接口移动到安全位置）
02702  *
02703  * 当机械臂还没达到目标速度的时候，给一个新的目标速度，机械臂会立刻达到新的目标速度
02704  *
02705  * @param qd 目标关节速度，单位 rad/s
02706  * @param a 主轴的加速度，单位 rad/s^2
02707  * @param t 函数返回所需要的时间，单位 s
02708  * 如果 t = 0，当达到目标速度的时候，函数将返回；
02709  * 反之，则经过 t 时间后，函数返回，不管是否达到目标速度。
02710  * 如果没有达到目标速度，会减速到零。
02711  * 如果达到了目标速度就是按照目标速度匀速运动。
02712  * @return 成功返回 0；失败返回错误码
02713  * AUBO_BUSY
02714  * -AUBO_INVL_ARGUMENT
02715  * -AUBO_BAD_STATE
02716  *
02717  * @throws arcs::common_interface::AuboException
02718  *
02719  * @par Python 函数原型
02720  * resumeSpeedJoint(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
02721  * float, arg2: float) -> int
02722  *
02723  * @par Lua 函数原型
02724  * resumeSpeedJoint(q: table, a: number, t: number) -> nil
02725  *
02726  * @par Lua 示例
02727  * resumeSpeedJoint({0.2,0,0,0,0,0},1.5,10)
02728  *
02729  * @par JSON-RPC 请求示例
02730  * {"jsonrpc":"2.0","method":"rob1.MotionControl.resumeSpeedJoint","params":[[0.2,0,0,0,0,0],1.5,100],"id":1}
02731  *
02732  * @par JSON-RPC 响应示例
02733  * {"id":1,"jsonrpc":"2.0","result":-1}
02734  * \endchinese
02735  * \english
02736  * Joint space velocity following (used to move to a safe position after a
02737  * collision during process execution)
02738  *
02739  * When the robot arm has not yet reached the target velocity, giving a new
02740  * target velocity will cause the robot arm to immediately reach the new
02741  * target velocity.
02742  *
02743  * @param qd Target joint velocity, unit: rad/s
02744  * @param a Main axis acceleration, unit: rad/s^2
02745  * @param t Time required for the function to return, unit: s
02746  * If t = 0, the function returns when the target velocity is reached;
02747  * otherwise, the function returns after t seconds, regardless of whether
02748  * the target velocity is reached. If the target velocity is not reached, it
02749  * will decelerate to zero. If the target velocity is reached, it will move
02750  * at a constant speed.
02751  * @return Returns 0 on success; otherwise returns error code
02752  * AUBO_BUSY
02753  * -AUBO_INVL_ARGUMENT
02754  * -AUBO_BAD_STATE
02755  *
02756  * @throws arcs::common_interface::AuboException
02757  *
02758  * @par Python function prototype
02759  * resumeSpeedJoint(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
02760  * float, arg2: float) -> int
02761  *
02762  * @par Lua function prototype
02763  * resumeSpeedJoint(q: table, a: number, t: number) -> nil
02764  *
02765  * @par Lua example
02766  * resumeSpeedJoint({0.2,0,0,0,0,0},1.5,10)
02767  *
02768  * @par JSON-RPC request example
02769  * {"jsonrpc":"2.0","method":"rob1.MotionControl.resumeSpeedJoint","params":[[0.2,0,0,0,0,0],1.5,100],"id":1}
02770  *
02771  * @par JSON-RPC response example
02772  * {"id":1,"jsonrpc":"2.0","result":-1}
02773  * \endenglish
02774  */
02775 int resumeSpeedJoint(const std::vector<double> &qd, double a, double t);
02776
02777 /**
02778  * @ingroup MotionControl
02779  * \chinese
02780  * 笛卡尔空间速度跟随
02781  *
02782  * 当机械臂还没达到目标速度的时候，给一个新的目标速度，机械臂会立刻达到新的目标速度
02783  *
02784  * @param xd 工具速度，单位 m/s

```

```

02785 * @param a 工具位置加速度, 单位 m/s^2
02786 * @param t 函数返回所需要的时间, 单位 s \n
02787 * 如果 t = 0, 当达到目标速度的时候, 函数将返回;
02788 * 反之, 则经过 t 时间后, 函数返回, 不管是否达到目标速度。
02789 * 如果没有达到目标速度, 会减速到零。
02790 * 如果达到了目标速度就是按照目标速度匀速运动。
02791 * @retval 0 成功
02792 * @retval AUBO_BAD_STATE(1) 当前安全模式处于非
02793 * Normal、ReducedMode、Recovery 状态
02794 * @retval AUBO_BUSY(3) 上一条指令正在执行中
02795 * @retval -AUBO_BAD_STATE(-1)
02796 * 可能的原因包括但不限于: 线程已分离、线程被终止、task_id
02797 * 未找到, 或者当前机器人模式非 Running
02798 * @retval -AUBO_TIMEOUT(-4) 调用接口超时
02799 *
02800 * @throws arcs::common_interface::AuboException
02801 *
02802 * @par Python 函数原型
02803 * speedLine(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: float,
02804 * arg2: float) -> int
02805 *
02806 * @par Lua 函数原型
02807 * speedLine(pose: table, a: number, t: number) -> nil
02808 *
02809 * @par Lua 示例
02810 * speedLine({0.25,0,0,0,0,0},1.2,100)
02811 *
02812 * @par JSON-RPC 请求示例
02813 * {"jsonrpc":"2.0","method":"rob1.MotionControl.speedLine","params":[[0.25,0,0,0,0,0],1.2,100],"id":1}
02814 *
02815 * @par JSON-RPC 响应示例
02816 * {"id":1,"jsonrpc":"2.0","result":0}
02817 *
02818 * \endchinese
02819 * \english
02820 * Cartesian space velocity following
02821 *
02822 * When the robot arm has not yet reached the target velocity, giving a new
02823 * target velocity will cause the robot arm to immediately reach the new
02824 * target velocity.
02825 *
02826 * @param xd Tool velocity, unit: m/s
02827 * @param a Tool position acceleration, unit: m/s^2
02828 * @param t Time required for the function to return, unit: s \n
02829 * If t = 0, the function returns when the target velocity is reached;
02830 * otherwise, the function returns after t seconds, regardless of whether
02831 * the target velocity is reached. If the target velocity is not reached, it
02832 * will decelerate to zero. If the target velocity is reached, it will move
02833 * at a constant speed.
02834 * @retval 0 Success
02835 * @retval AUBO_BAD_STATE(1) The current safety mode is not Normal,
02836 * ReducedMode, or Recovery
02837 * @retval AUBO_BUSY(3) The previous instruction is being executed
02838 * @retval -AUBO_BAD_STATE(-1)
02839 * Possible reasons include but are not limited to: thread has been
02840 * detached, thread terminated, task_id not found, or the current robot mode
02841 * is not Running
02842 * @retval -AUBO_TIMEOUT(-4) Interface call timeout
02843 *
02844 * @throws arcs::common_interface::AuboException
02845 *
02846 * @par Python function prototype
02847 * speedLine(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: float,
02848 * arg2: float) -> int
02849 *
02850 * @par Lua function prototype
02851 * speedLine(pose: table, a: number, t: number) -> nil
02852 *
02853 * @par Lua example
02854 * speedLine({0.25,0,0,0,0,0},1.2,100)
02855 *
02856 * @par JSON-RPC request example
02857 * {"jsonrpc":"2.0","method":"rob1.MotionControl.speedLine","params":[[0.25,0,0,0,0,0],1.2,100],"id":1}
02858 *
02859 * @par JSON-RPC response example
02860 * {"id":1,"jsonrpc":"2.0","result":0}
02861 *
02862 * \endenglish
02863 */
02864 int speedLine(const std::vector<double> &xd, double a, double t);
02865
02866 /**
02867 * @ingroup MotionControl
02868 * \chinese
02869 * 笛卡尔空间速度跟随 (机械臂运行工程时发生碰撞, 通过此接口移动到安全位置)
02870 *
02871 * 当机械臂还没达到目标速度的时候, 给一个新的目标速度, 机械臂会立刻达到新的目标速度

```



```

02872 *
02873 * @param xd 工具速度, 单位 m/s
02874 * @param a 工具位置加速度, 单位 m/s^2
02875 * @param t 函数返回所需要的时间, 单位 s \n
02876 * 如果 t = 0, 当达到目标速度的时候, 函数将返回;
02877 * 反之, 则经过 t 时间后, 函数返回, 不管是否达到目标速度。
02878 * 如果没有达到目标速度, 会减速到零。
02879 * 如果达到了目标速度就是按照目标速度匀速运动。
02880 * @return 成功返回 0; 失败返回错误码
02881 * -AUBO_BAD_STATE
02882 *
02883 * @throws arcs::common_interface::AuboException
02884 *
02885 * @par Python 函数原型
02886 * resumeSpeedLine(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
02887 * float, arg2: float) -> int
02888 *
02889 * @par Lua 函数原型
02890 * resumeSpeedLine(pose: table, a: number, t: number) -> nil
02891 *
02892 * @par Lua 示例
02893 * resumeSpeedLine({0.25,0,0,0,0,0},1.2,100)
02894 *
02895 * @par JSON-RPC 请求示例
02896 * {"jsonrpc":"2.0","method":"robl.MotionControl.resumeSpeedLine","params":[[0.25,0,0,0,0,0],1.2,100],"id":1}
02897 *
02898 * @par JSON-RPC 响应示例
02899 * {"id":1,"jsonrpc":"2.0","result":-1}
02900 * \endchinese
02901 * \english
02902 * Cartesian space velocity following (used to move to a safe position after
02903 * a collision during process execution)
02904 *
02905 * When the robot arm has not yet reached the target velocity, giving a new
02906 * target velocity will cause the robot arm to immediately reach the new
02907 * target velocity.
02908 *
02909 * @param xd Tool velocity, unit: m/s
02910 * @param a Tool position acceleration, unit: m/s^2
02911 * @param t Time required for the function to return, unit: s \n
02912 * If t = 0, the function returns when the target velocity is reached;
02913 * otherwise, the function returns after t seconds, regardless of whether
02914 * the target velocity is reached. If the target velocity is not reached, it
02915 * will decelerate to zero. If the target velocity is reached, it will move
02916 * at a constant speed.
02917 * @return Returns 0 on success; otherwise returns error code
02918 * -AUBO_BAD_STATE
02919 *
02920 * @throws arcs::common_interface::AuboException
02921 *
02922 * @par Python function prototype
02923 * resumeSpeedLine(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
02924 * float, arg2: float) -> int
02925 *
02926 * @par Lua function prototype
02927 * resumeSpeedLine(pose: table, a: number, t: number) -> nil
02928 *
02929 * @par Lua example
02930 * resumeSpeedLine({0.25,0,0,0,0,0},1.2,100)
02931 *
02932 * @par JSON-RPC request example
02933 * {"jsonrpc":"2.0","method":"robl.MotionControl.resumeSpeedLine","params":[[0.25,0,0,0,0,0],1.2,100],"id":1}
02934 *
02935 * @par JSON-RPC response example
02936 * {"id":1,"jsonrpc":"2.0","result":-1}
02937 * \endenglish
02938 */
02939 int resumeSpeedLine(const std::vector<double> &xd, double a, double t);
02940
02941 /**
02942 * @ingroup MotionControl
02943 * \chinese
02944 * 在关节空间做样条插值
02945 *
02946 * @param q 关节角度, 如果传入参数维度为 0, 表示样条运动结束
02947 * @param a 加速度, 单位 rad/s^2,
02948 * 最大值可通过 RobotConfig 类中的接口 getJointMaxAccelerations() 来获取
02949 * @param v 速度, 单位 rad/s,
02950 * 最大值可通过 RobotConfig 类中的接口 getJointMaxSpeeds() 来获取
02951 * @param duration
02952 * @return 成功返回 0; 失败返回错误码
02953 * AUBO_BUSY
02954 * AUBO_BAD_STATE
02955 * -AUBO_BAD_STATE
02956 *
02957 * @throws arcs::common_interface::AuboException
02958 *

```



```

02959 * @par Python 函数原型
02960 * moveSpline(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
02961 * float, arg2: float, arg3: float) -> int
02962 *
02963 * @par Lua 函数原型
02964 * moveSpline(q: table, a: number, v: number, duration: number) -> nil
02965 *
02966 * @par Lua 示例
02967 * moveSpline({-2.05177,-0.400292, 1.19625, 0.0285152, 1.57033,
02968 * -2.28774},1.3,1.0,0)
02969 *
02970 * @par JSON-RPC 请求示例
02971 * {"jsonrpc":"2.0","method":"robl.MotionControl.moveSpline","params":[[0,0,0,0,0,0],1,1,0],"id":1}
02972 *
02973 * @par JSON-RPC 响应示例
02974 * {"id":1,"jsonrpc":"2.0","result":0}
02975 * \endchinese
02976 * \english
02977 * Perform spline interpolation in joint space
02978 *
02979 * @param q Joint angles. If the input vector is of size 0, it indicates the
02980 * end of the spline motion.
02981 * @param a Acceleration, unit: rad/s2. The maximum value can be obtained
02982 * via RobotConfig::getJointMaxAccelerations().
02983 * @param v Velocity, unit: rad/s. The maximum value can be obtained via
02984 * RobotConfig::getJointMaxSpeeds().
02985 * @param duration
02986 * @return Returns 0 on success; otherwise returns an error code:
02987 * AUBO_BUSY
02988 * AUBO_BAD_STATE
02989 * -AUBO_BAD_STATE
02990 *
02991 * @throws arcs::common_interface::AuboException
02992 *
02993 * @par Python function prototype
02994 * moveSpline(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
02995 * float, arg2: float, arg3: float) -> int
02996 *
02997 * @par Lua function prototype
02998 * moveSpline(q: table, a: number, v: number, duration: number) -> nil
02999 *
03000 * @par Lua example
03001 * moveSpline({-2.05177,-0.400292, 1.19625, 0.0285152, 1.57033,
03002 * -2.28774},1.3,1.0,0)
03003 *
03004 * @par JSON-RPC request example
03005 * {"jsonrpc":"2.0","method":"robl.MotionControl.moveSpline","params":[[0,0,0,0,0,0],1,1,0],"id":1}
03006 *
03007 * @par JSON-RPC response example
03008 * {"id":1,"jsonrpc":"2.0","result":0}
03009 * \endenglish
03010 */
03011 int moveSpline(const std::vector<double> &q, double a, double v,
03012               double duration);
03013
03014 /**
03015 * @ingroup MotionControl
03016 * \chinese
03017 * 添加关节运动
03018 *
03019 * @param q 关节角, 单位 rad
03020 * @param a 加速度, 单位 rad/s2,
03021 * 最大值可通过 RobotConfig 类中的接口 getJointMaxAccelerations() 来获取
03022 * @param v 速度, 单位 rad/s,
03023 * 最大值可通过 RobotConfig 类中的接口 getJointMaxSpeeds() 来获取
03024 * @param blend_radius 交融半径, 单位 m
03025 * @param duration 运行时间, 单位 s \n
03026 * 通常当给定了速度和加速度, 便能够确定轨迹的运行时间。
03027 * 如果想延长轨迹的运行时间, 便要设置 duration 这个参数。
03028 * duration 可以延长轨迹运动的时间, 但是不能缩短轨迹时间。 \n
03029 * 当 duration = 0 的时候,
03030 * 表示不指定运行时间, 即没有明确完成运动的时间, 将根据速度与加速度计算运行时间。
03031 * 如果 duration 不等于 0, a 和 v 的值将被忽略。
03032 * @retval 0 成功
03033 * @retval AUBO_BAD_STATE(1) 当前安全模式处于非
03034 * Normal、ReducedMode、Recovery 状态
03035 * @retval AUBO_QUEUE_FULL(2) 规划队列已满
03036 * @retval AUBO_BUSY(3) 上一条指令正在执行中
03037 * @retval -AUBO_BAD_STATE(-1)
03038 * 可能的原因包括但不限于: 线程已分离、线程被终止、task_id
03039 * 未找到, 或者当前机器人模式非 Running
03040 * @retval -AUBO_TIMEOUT(-4) 调用接口超时
03041 * @retval -AUBO_INVL_ARGUMENT(-5) 参数数组 q 的长度小于当前机器臂的自由度
03042 * @retval AUBO_REQUEST_IGNORE(13) 参数数组 q 的长度太短且无需在该点停留
03043 *
03044 * @throws arcs::common_interface::AuboException
03045 *

```

```

03046 * @par Python 函数原型
03047 * moveJoint(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: float,
03048 * arg2: float, arg3: float, arg4: float) -> int
03049 *
03050 * @par Lua 函数原型
03051 * moveJoint(q: table, a: number, v: number, blend_radius: number, duration:
03052 * number) -> nil
03053 *
03054 * @par Lua 示例
03055 * moveJoint({-2.05177,-0.400292, 1.19625, 0.0285152, 1.57033,
03056 * -2.28774},0.3,0.3,0,0)
03057 *
03058 * @par JSON-RPC 请求示例
03059 * {"jsonrpc":"2.0","method":"rob1.MotionControl.moveJoint","params":[[-2.05177,
03060 * -0.400292, 1.19625, 0.0285152, 1.57033, -2.28774],0.3,0.3,0,0],"id":1}
03061 *
03062 * @par JSON-RPC 响应示例
03063 * {"id":1,"jsonrpc":"2.0","result":0}
03064 *
03065 * \endchinese
03066 * \english
03067 * Add joint motion
03068 *
03069 * @param q Joint angles, unit: rad
03070 * @param a Acceleration, unit: rad/s^2,
03071 * The maximum value can be obtained via
03072 * RobotConfig::getJointMaxAccelerations()
03073 * @param v Velocity, unit: rad/s,
03074 * The maximum value can be obtained via RobotConfig::getJointMaxSpeeds()
03075 * @param blend_radius Blend radius, unit: m
03076 * @param duration Execution time, unit: s \n
03077 * Normally, when speed and acceleration are given, the trajectory duration
03078 * can be determined. If you want to extend the trajectory duration, set the
03079 * duration parameter. Duration can extend the trajectory time, but cannot
03080 * shorten it.\n When duration = 0, it means the execution time is not
03081 * specified, and the time will be calculated based on speed and
03082 * acceleration. If duration is not 0, a and v values will be ignored.
03083 * @retval 0 Success
03084 * @retval AUBO_BAD_STATE(1) The current safety mode is not Normal,
03085 * ReducedMode, or Recovery
03086 * @retval AUBO_QUEUE_FULL(2) Planning queue is full
03087 * @retval AUBO_BUSY(3) The previous instruction is being executed
03088 * @retval -AUBO_BAD_STATE(-1)
03089 * Possible reasons include but are not limited to: thread has been
03090 * detached, thread terminated, task_id not found, or the current robot mode
03091 * is not Running
03092 * @retval -AUBO_TIMEOUT(-4) Interface call timeout
03093 * @retval -AUBO_INVL_ARGUMENT(-5) The length of q is less than the degrees
03094 * of freedom of the current robot arm
03095 * @retval AUBO_REQUEST_IGNORE(13) The length of q is too short and there is
03096 * no need to stop at that point
03097 *
03098 * @throws arcs::common_interface::AuboException
03099 *
03100 * @par Python function prototype
03101 * moveJoint(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: float,
03102 * arg2: float, arg3: float, arg4: float) -> int
03103 *
03104 * @par Lua function prototype
03105 * moveJoint(q: table, a: number, v: number, blend_radius: number, duration:
03106 * number) -> nil
03107 *
03108 * @par Lua example
03109 * moveJoint({-2.05177,-0.400292, 1.19625, 0.0285152, 1.57033,
03110 * -2.28774},0.3,0.3,0,0)
03111 *
03112 * @par JSON-RPC request example
03113 * {"jsonrpc":"2.0","method":"rob1.MotionControl.moveJoint","params":[[-2.05177,
03114 * -0.400292, 1.19625, 0.0285152, 1.57033, -2.28774],0.3,0.3,0,0],"id":1}
03115 *
03116 * @par JSON-RPC response example
03117 * {"id":1,"jsonrpc":"2.0","result":0}
03118 *
03119 * \endenglish
03120 */
03121 int moveJoint(const std::vector<double> &q, double a, double v,
03122 double blend_radius, double duration);
03123
03124 /**
03125 * @ingroup MotionControl
03126 * \chinese
03127 * 机器人与外部轴同步运动
03128 *
03129 * @param group_name
03130 * @param q
03131 * @param a
03132 * @param v

```

```

03133     * @param blend_radius
03134     * @param duration
03135     * @return
03136     * \endchinese
03137     * \english
03138     * Synchronous motion of robot and external axes
03139     *
03140     * @param group_name
03141     * @param q
03142     * @param a
03143     * @param v
03144     * @param blend_radius
03145     * @param duration
03146     * @return
03147     * \endenglish
03148     */
03149 int moveJointWithAxisGroup(const std::vector<double> &q, double a, double v,
03150                           double blend_radius, double duration,
03151                           const std::string &group_name,
03152                           const std::vector<double> &extq);
03153
03154 /**
03155  * @ingroup MotionControl
03156  * \chinese
03157  * 通过关节运动移动到暂停点的位置
03158  *
03159  * @param q 关节角, 单位 rad
03160  * @param a 加速度, 单位 rad/s^2
03161  * @param v 速度, 单位 rad/s
03162  * @param duration 运行时间, 单位 s \n
03163  * 通常当给定了速度和加速度, 便能够确定轨迹的运行时间。
03164  * 如果想延长轨迹的运行时间, 便要设置 duration 这个参数。
03165  * duration 可以延长轨迹运动的时间, 但是不能缩短轨迹时间。 \n
03166  * 当 duration = 0 的时候,
03167  * 表示不指定运行时间, 即没有明确完成运动的时间, 将根据速度与加速度计算运行时间。
03168  * 如果 duration 不等于 0, a 和 v 的值将被忽略。
03169  * @return 成功返回 0; 失败返回错误码
03170  * -AUBO_INVL_ARGUMENT
03171  * -AUBO_BAD_STATE
03172  *
03173  * @throws arcs::common_interface::AuboException
03174  *
03175  * @par Python 函数原型
03176  * resumeMoveJoint(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
03177  * float, arg2: float, arg3: float) -> int
03178  *
03179  * @par Lua 函数原型
03180  * resumeMoveJoint(q: table, a: number, v: number, duration: number) -> nil
03181  *
03182  * @par Lua 示例
03183  * resumeMoveJoint({-2.05177,-0.400292, 1.19625, 0.0285152, 1.57033,
03184  * -2.28774},1.3,1.0,0)
03185  *
03186  * @par JSON-RPC 请求示例
03187  * {"jsonrpc":"2.0","method":"rob1.MotionControl.resumeMoveJoint","params":[[-2.05177,
03188  * -0.400292, 1.19625, 0.0285152, 1.57033, -2.28774],0.3,0.3,0],"id":1}
03189  *
03190  * @par JSON-RPC 响应示例
03191  * {"id":1,"jsonrpc":"2.0","result":-1}
03192  * \endchinese
03193  * \english
03194  * Move to the pause point using joint motion.
03195  *
03196  * @param q Joint angles, unit: rad
03197  * @param a Acceleration, unit: rad/s^2
03198  * @param v Velocity, unit: rad/s
03199  * @param duration Execution time, unit: s \n
03200  * Normally, when speed and acceleration are given, the trajectory duration
03201  * can be determined. If you want to extend the trajectory duration, set the
03202  * duration parameter. Duration can extend the trajectory time, but cannot
03203  * shorten it. \n When duration = 0, it means the execution time is not
03204  * specified, and the time will be calculated based on speed and
03205  * acceleration. If duration is not 0, a and v values will be ignored.
03206  * @return Returns 0 on success; otherwise returns error code
03207  * -AUBO_INVL_ARGUMENT
03208  * -AUBO_BAD_STATE
03209  *
03210  * @throws arcs::common_interface::AuboException
03211  *
03212  * @par Python function prototype
03213  * resumeMoveJoint(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
03214  * float, arg2: float, arg3: float) -> int
03215  *
03216  * @par Lua function prototype
03217  * resumeMoveJoint(q: table, a: number, v: number, duration: number) -> nil
03218  *
03219  * @par Lua example

```

```

03220 * resumeMoveJoint({-2.05177,-0.400292, 1.19625, 0.0285152, 1.57033,
03221 * -2.28774},1.3,1.0,0)
03222 *
03223 * @par JSON-RPC request example
03224 * {"jsonrpc":"2.0","method":"rob1.MotionControl.resumeMoveJoint","params":[[-2.05177,
03225 * -0.400292, 1.19625, 0.0285152, 1.57033, -2.28774],0.3,0.3,0],"id":1}
03226 *
03227 * @par JSON-RPC response example
03228 * {"id":1,"jsonrpc":"2.0","result":-1}
03229 * \endenglish
03230 */
03231 int resumeMoveJoint(const std::vector<double> &q, double a, double v,
03232 double duration);
03233
03234 /**
03235 * @ingroup MotionControl
03236 * \chinese
03237 * 添加直线运动
03238 *
03239 * @param pose 目标位姿
03240 * @param a 加速度 (如果位置变化小于 1mm, 姿态变化大于 1e-4
03241 * rad, 此加速度会被作为角加速度, 单位 rad/s^2. 否则为线加速度, 单位 m/s^2)
03242 * 最大值可通过 RobotConfig 类中的接口 getTcpMaxAccelerations() 来获取
03243 * @param v 速度 (如果位置变化小于 1mm, 姿态变化大于 1e-4
03244 * rad, 此速度会被作为角速度, 单位 rad/s. 否则为线速度, 单位 m/s)
03245 * 最大值可通过 RobotConfig 类中的接口 getTcpMaxSpeeds() 来获取
03246 * @param blend_radius 交融半径, 单位 m, 值介于 0.001 和 1 之间
03247 * @param duration 运行时间, 单位 s \n
03248 * 通常当给定了速度和加速度, 便能够确定轨迹的运行时间。
03249 * 如果想延长轨迹的运行时间, 便要设置 duration 这个参数。
03250 * duration 可以延长轨迹运动的时间, 但是不能缩短轨迹时间。 \n
03251 * 当 duration = 0 的时候,
03252 * 表示不指定运行时间, 即没有明确完成运动的时间, 将根据速度与加速度计算运行时间。
03253 * 如果 duration 不等于 0, a 和 v 的值将被忽略。
03254 * @retval 0 成功
03255 * @retval AUBO_BAD_STATE(1) 当前安全模式处于非
03256 * Normal、ReducedMode、Recovery 状态
03257 * @retval AUBO_QUEUE_FULL(2) 轨迹队列已满
03258 * @retval AUBO_BUSY(3) 上一条指令正在执行中
03259 * @retval -AUBO_BAD_STATE(-1)
03260 * 可能的原因包括但不限于: 线程已分离、线程被终止、task_id
03261 * 未找到, 或者当前机器人模式非 Running
03262 * @retval -AUBO_TIMEOUT(-4) 调用接口超时
03263 * @retval -AUBO_INVL_ARGUMENT(-5) 参数数组 pose
03264 * 的长度小于当前机器臂的自由度
03265 * @retval AUBO_REQUEST_IGNORE(13) 参数数组 pose
03266 * 路径长度太短且无需在该点停留
03267 *
03268 * @throws arcs::common_interface::AuboException
03269 *
03270 * @par Python 函数原型
03271 * moveLine(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: float,
03272 * arg2: float, arg3: float, arg4: float) -> int
03273 *
03274 * @par Lua 函数原型
03275 * moveLine(pose: table, a: number, v: number, blend_radius: number,
03276 * duration: number) -> nil
03277 *
03278 * @par Lua 示例
03279 * moveLine({0.58815,0.0532,0.62391,2.46,0.479,1.619},1.3,1.0,0)
03280 *
03281 * @par JSON-RPC 请求示例
03282 * {"jsonrpc":"2.0","method":"rob1.MotionControl.moveLine","params":
[[0.58815,0.0532,0.62391,2.46,0.479,1.619],1.2,0.25,0,0],"id":1}
03283 *
03284 * @par JSON-RPC 响应示例
03285 * {"id":1,"jsonrpc":"2.0","result":0}
03286 *
03287 * \endchinese
03288 * \english
03289 * Add linear motion
03290 *
03291 * @param pose Target pose
03292 * @param a Acceleration (if the position change is less than 1mm and the
03293 * posture change is greater than 1e-4 rad, this acceleration is treated as
03294 * angular acceleration in rad/s^2; otherwise, as linear acceleration in
03295 * m/s^2). The maximum value can be obtained via
03296 * RobotConfig::getTcpMaxAccelerations().
03297 * @param v Velocity (if the position change is less than 1mm and the
03298 * posture change is greater than 1e-4 rad, this velocity is treated as
03299 * angular velocity in rad/s; otherwise, as linear velocity in m/s). The
03300 * maximum value can be obtained via RobotConfig::getTcpMaxSpeeds().
03301 * @param blend_radius Blend radius, unit: m, value between 0.001 and 1
03302 * @param duration Execution time, unit: s \n
03303 * Normally, when speed and acceleration are given, the trajectory duration
03304 * can be determined. If you want to extend the trajectory duration, set the
03305 * duration parameter. Duration can extend the trajectory time, but cannot

```

```

03306     * shorten it.\n When duration = 0, it means the execution time is not
03307     * specified, and the time will be calculated based on speed and
03308     * acceleration. If duration is not 0, a and v values will be ignored.
03309     * @retval 0 Success
03310     * @retval AUBO_BAD_STATE(1) The current safety mode is not Normal,
03311     * ReducedMode, or Recovery
03312     * @retval AUBO_QUEUE_FULL(2) Trajectory queue is full
03313     * @retval AUBO_BUSY(3) The previous instruction is being executed
03314     * @retval -AUBO_BAD_STATE(-1) Possible reasons include but are not limited
03315     * to: thread has been detached, thread terminated, task_id not found, or
03316     * the current robot mode is not Running
03317     * @retval -AUBO_TIMEOUT(-4) Interface call timeout
03318     * @retval -AUBO_INVL_ARGUMENT(-5) The length of the pose array is less than
03319     * the degrees of freedom of the current robot arm
03320     * @retval AUBO_REQUEST_IGNORE(13) The length of the pose array is too short
03321     * and there is no need to stop at that point
03322     *
03323     * @throws arcs::common_interface::AuboException
03324     *
03325     * @par Python function prototype
03326     * moveLine(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1: float,
03327     * arg2: float, arg3: float, arg4: float) -> int
03328     *
03329     * @par Lua function prototype
03330     * moveLine(pose: table, a: number, v: number, blend_radius: number,
03331     * duration: number) -> nil
03332     *
03333     * @par Lua example
03334     * moveLine({0.58815,0.0532,0.62391,2.46,0.479,1.619},1.3,1.0,0)
03335     *
03336     * @par JSON-RPC request example
03337     * {"jsonrpc":"2.0","method":"robl.MotionControl.moveLine","params":
[[0.58815,0.0532,0.62391,2.46,0.479,1.619],1.2,0.25,0,0],"id":1}
03338     *
03339     * @par JSON-RPC response example
03340     * {"id":1,"jsonrpc":"2.0","result":0}
03341     *
03342     * \endenglish
03343     */
03344 int moveLine(const std::vector<double> &pose, double a, double v,
03345             double blend_radius, double duration);
03346
03347 /**
03348  * @ingroup MotionControl
03349  * \chinese
03350  * 直线运动与外部轴同步运动
03351  *
03352  * @param group_name 外部轴组名称
03353  * @param pose 目标位姿
03354  * @param a 加速度
03355  * @param v 速度
03356  * @param blend_radius 交融半径
03357  * @param duration 运行时间
03358  * @return
03359  * \endchinese
03360  * \english
03361  * Linear motion synchronized with external axes
03362  *
03363  * @param group_name Name of the external axis group
03364  * @param pose Target pose
03365  * @param a Acceleration
03366  * @param v Velocity
03367  * @param blend_radius Blend radius
03368  * @param duration Execution time
03369  * @return
03370  * \endenglish
03371  */
03372 int moveLineWithAxisGroup(const std::vector<double> &pose, double a,
03373                          double v, double blend_radius, double duration,
03374                          const std::string &group_name,
03375                          const std::vector<double> &extq);
03376
03377 /**
03378  * @ingroup MotionControl
03379  * \chinese
03380  * 添加工艺运动
03381  *
03382  * @param pose
03383  * @param a
03384  * @param v
03385  * @param blend_radius
03386  * @return 成功返回 0; 失败返回错误码
03387  * AUBO_BAD_STATE
03388  * AUBO_BUSY
03389  * -AUBO_INVL_ARGUMENT
03390  * -AUBO_BAD_STATE
03391  *

```

```

03392     * @throws arcs::common_interface::AuboException
03393     *
03394     * @par Lua 函数原型
03395     * moveProcess(pose: table, a: number, v: number, blend_radius: number,
03396     * duration: number) -> nil
03397     *
03398     * @par Lua 示例
03399     * moveProcess({0.58815,0.0532,0.62391,2.46,0.479,1.619},1.2,0.25,0)
03400     *
03401     * @par JSON-RPC 请求示例
03402     * {"jsonrpc":"2.0","method":"rob1.MotionControl.moveProcess","params":
[[0.58815,0.0532,0.62391,2.46,0.479,1.619],1.2,0.25,0],"id":1}
03403     *
03404     * @par JSON-RPC 响应示例
03405     * {"id":1,"jsonrpc":"2.0","result":0}
03406     * \endchinese
03407     * \english
03408     * Add process motion
03409     *
03410     * @param pose
03411     * @param a
03412     * @param v
03413     * @param blend_radius
03414     * @return Returns 0 on success; otherwise returns an error code:
03415     * AUBO_BAD_STATE
03416     * AUBO_BUSY
03417     * -AUBO_INVL_ARGUMENT
03418     * -AUBO_BAD_STATE
03419     *
03420     * @throws arcs::common_interface::AuboException
03421     *
03422     * @par Lua function prototype
03423     * moveProcess(pose: table, a: number, v: number, blend_radius: number,
03424     * duration: number) -> nil
03425     *
03426     * @par Lua example
03427     * moveProcess({0.58815,0.0532,0.62391,2.46,0.479,1.619},1.2,0.25,0)
03428     *
03429     * @par JSON-RPC request example
03430     * {"jsonrpc":"2.0","method":"rob1.MotionControl.moveProcess","params":
[[0.58815,0.0532,0.62391,2.46,0.479,1.619],1.2,0.25,0],"id":1}
03431     *
03432     * @par JSON-RPC response example
03433     * {"id":1,"jsonrpc":"2.0","result":0}
03434     * \endenglish
03435     */
03436     int moveProcess(const std::vector<double> &pose, double a, double v,
03437                   double blend_radius);
03438
03439     /**
03440     * @ingroup MotionControl
03441     * \chinese
03442     * 通过直线运动移动到暂停点的位置
03443     *
03444     * @param pose 目标位姿
03445     * @param a 加速度, 单位 m/s^2
03446     * @param v 速度, 单位 m/s
03447     * @param duration 运行时间, 单位 s \n
03448     * 通常当给定了速度和加速度, 便能够确定轨迹的运行时间。
03449     * 如果想延长轨迹的运行时间, 便要设置 duration 这个参数。
03450     * duration 可以延长轨迹运动的时间, 但是不能缩短轨迹时间。 \n
03451     * 当 duration = 0 的时候,
03452     * 表示不指定运行时间, 即没有明确完成运动的时间, 将根据速度与加速度计算运行时间。
03453     * 如果 duration 不等于 0, a 和 v 的值将被忽略。
03454     * @return 成功返回 0; 失败返回错误码
03455     * -AUBO_INVL_ARGUMENT
03456     * -AUBO_BAD_STATE
03457     *
03458     * @throws arcs::common_interface::AuboException
03459     *
03460     * @par Python 函数原型
03461     * resumeMoveLine(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
03462     * float, arg2: float, arg3: float) -> int
03463     *
03464     * @par Lua 函数原型
03465     * resumeMoveLine(pose: table, a: number, v: number,duration: number) -> nil
03466     *
03467     * @par Lua 示例
03468     * resumeMoveLine({0.58815,0.0532,0.62391,2.46,0.479,1.619},1.2,0.25,0)
03469     *
03470     * @par JSON-RPC 请求示例
03471     * {"jsonrpc":"2.0","method":"rob1.MotionControl.resumeMoveLine","params":
[[0.58815,0.0532,0.62391,2.46,0.479,1.619],1.2,0.25,0],"id":1}
03472     *
03473     * @par JSON-RPC 响应示例
03474     * {"id":1,"jsonrpc":"2.0","result":0}
03475     *

```



```

03476 * \endchinese
03477 * \english
03478 * Move to the pause point using linear motion.
03479 *
03480 * @param pose Target pose
03481 * @param a Acceleration, unit: m/s^2
03482 * @param v Velocity, unit: m/s
03483 * @param duration Execution time, unit: s \n
03484 * Normally, when speed and acceleration are given, the trajectory duration
03485 * can be determined. If you want to extend the trajectory duration, set the
03486 * duration parameter. Duration can extend the trajectory time, but cannot
03487 * shorten it.\n When duration = 0, it means the execution time is not
03488 * specified, and the time will be calculated based on speed and
03489 * acceleration. If duration is not 0, a and v values will be ignored.
03490 * @return Returns 0 on success; otherwise returns error code
03491 * -AUBO_INVL_ARGUMENT
03492 * -AUBO_BAD_STATE
03493 *
03494 * @throws arcs::common_interface::AuboException
03495 *
03496 * @par Python function prototype
03497 * resumeMoveLine(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
03498 * float, arg2: float, arg3: float) -> int
03499 *
03500 * @par Lua function prototype
03501 * resumeMoveLine(pose: table, a: number, v: number, duration: number) ->
03502 * nil
03503 *
03504 * @par Lua example
03505 * resumeMoveLine({0.58815,0.0532,0.62391,2.46,0.479,1.619},1.2,0.25,0)
03506 *
03507 * @par JSON-RPC request example
03508 * {"jsonrpc":"2.0","method":"rob1.MotionControl.resumeMoveLine","params":
[[0.58815,0.0532,0.62391,2.46,0.479,1.619],1.2,0.25,0],"id":1}
03509 *
03510 * @par JSON-RPC response example
03511 * {"id":1,"jsonrpc":"2.0","result":0}
03512 *
03513 * \endenglish
03514 */
03515 int resumeMoveLine(const std::vector<double> &pose, double a, double v,
03516 double duration);
03517
03518 /**
03519 * @ingroup MotionControl
03520 * \chinese
03521 * 添加圆弧运动
03522 *
03523 * @todo 可以由多段圆弧组成圆周运动
03524 *
03525 * @param via_pose 圆弧运动途中点的位姿
03526 * @param end_pose 圆弧运动结束点的位姿
03527 * @param a 加速度（如果 via_pose 与上一个路点位置变化小于 1mm，姿态变化大于 1e-4
03528 * rad，此加速度会被作为角加速度，单位 rad/s^2。否则为线加速度，单位 m/s^2）
03529 * @param v 速度（如果 via_pose 与上一个路点位置变化小于 1mm，姿态变化大于 1e-4
03530 * rad，此速度会被作为角速度，单位 rad / s。否则为线速度，单位 m / s）
03531 * @param blend_radius 交融半径，单位: m
03532 * @param duration 运行时间，单位: s \n
03533 * 通常当给定了速度和加速度，便能够确定轨迹的运行时间。
03534 * 如果想延长轨迹的运行时间，便要设置 duration 这个参数。
03535 * duration 可以延长轨迹运动的时间，但是不能缩短轨迹时间。 \n
03536 * 当 duration = 0 的时候，
03537 * 表示不指定运行时间，即没有明确完成运动的时间，将根据速度与加速度计算运行时间。
03538 * 如果 duration 不等于 0，a 和 v 的值将被忽略。
03539 * @return 成功返回 0；失败返回错误码
03540 * AUBO_BAD_STATE
03541 * AUBO_BUSY
03542 * -AUBO_INVL_ARGUMENT
03543 * -AUBO_BAD_STATE
03544 *
03545 * @throws arcs::common_interface::AuboException
03546 *
03547 * @par Python 函数原型
03548 * moveCircle(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
03549 * List[float], arg2: float, arg3: float, arg4: float, arg5: float) -> int
03550 *
03551 * @par Lua 函数原型
03552 * moveCircle(via_pose: table, end_pose: table, a: number, v: number,
03553 * blend_radius: number, duration: number) -> nil
03554 *
03555 * @par Lua 示例
03556 * moveCircle({0.440164,-0.00249391,0.398658,2.45651,0,1.5708},
{0.440164,0.166256,0.297599,2.45651,0,1.5708},1.2,0.25,0)
03557 *
03558 * @par JSON-RPC 请求示例
03559 * {"jsonrpc":"2.0","method":"rob1.MotionControl.moveCircle","params":
[[0.440164,-0.00249391,0.398658,2.45651,0,1.5708],

```

```

03560 [0.440164,0.166256,0.297599,2.45651,0,1.5708],1.2,0.25,0,0],"id":1}
03561 *
03562 * @par JSON-RPC 响应示例
03563 * {"id":1,"jsonrpc":"2.0","result":0}
03564 * \endchinese
03565 * \english
03566 * Add circular arc motion
03567 *
03568 * @todo Can be composed of multiple arcs to form a circular motion
03569 *
03570 * @param via_pose The pose of the via point during the arc motion
03571 * @param end_pose The pose of the end point of the arc motion
03572 * @param a Acceleration (if the position change between via_pose and the
03573 * previous waypoint is less than 1mm and the posture change is greater than
03574 * 1e-4 rad, this acceleration is treated as angular acceleration in
03575 * rad/s2; otherwise, as linear acceleration in m/s2)
03576 * @param v Velocity (if the position change between via_pose and the
03577 * previous waypoint is less than 1mm and the posture change is greater than
03578 * 1e-4 rad, this velocity is treated as angular velocity in rad/s;
03579 * otherwise, as linear velocity in m/s)
03580 * @param blend_radius Blend radius, unit: m
03581 * @param duration Execution time, unit: s \n
03582 * Normally, when speed and acceleration are given, the trajectory duration
03583 * can be determined. If you want to extend the trajectory duration, set the
03584 * duration parameter. Duration can extend the trajectory time, but cannot
03585 * shorten it.\n When duration = 0, it means the execution time is not
03586 * specified, and the time will be calculated based on speed and
03587 * acceleration. If duration is not 0, a and v values will be ignored.
03588 * @return Returns 0 on success; otherwise returns error code
03589 * AUBO_BAD_STATE
03590 * AUBO_BUSY
03591 * -AUBO_INVL_ARGUMENT
03592 * -AUBO_BAD_STATE
03593 *
03594 * @throws arcs::common_interface::AuboException
03595 *
03596 * @par Python function prototype
03597 * moveCircle(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
03598 * List[float], arg2: float, arg3: float, arg4: float, arg5: float) -> int
03599 *
03600 * @par Lua function prototype
03601 * moveCircle(via_pose: table, end_pose: table, a: number, v: number,
03602 * blend_radius: number, duration: number) -> nil
03603 *
03604 * @par Lua example
03605 * moveCircle({0.440164,-0.00249391,0.398658,2.45651,0,1.5708},
03606 * {0.440164,0.166256,0.297599,2.45651,0,1.5708},1.2,0.25,0)
03607 *
03608 * @par JSON-RPC request example
03609 * {"jsonrpc":"2.0","method":"robl.MotionControl.moveCircle","params":
03610 * [[0.440164,-0.00249391,0.398658,2.45651,0,1.5708],
03611 * [0.440164,0.166256,0.297599,2.45651,0,1.5708],1.2,0.25,0,0],"id":1}
03612 */
03613 int moveCircle(const std::vector<double> &via_pose,
03614               const std::vector<double> &end_pose, double a, double v,
03615               double blend_radius, double duration);
03616
03617 /**
03618 * @ingroup MotionControl
03619 * \chinese
03620 * moveCircle 与外部轴同步运动
03621 *
03622 * @param group_name 外部轴组名称
03623 * @param via_pose 圆弧运动途中点的位姿
03624 * @param end_pose 圆弧运动结束点的位姿
03625 * @param a 加速度
03626 * @param v 速度
03627 * @param blend_radius 交融半径
03628 * @param duration 运行时间
03629 * @return
03630 * \endchinese
03631 * \english
03632 * moveCircle with external axes synchronized motion
03633 *
03634 * @param group_name Name of the external axis group
03635 * @param via_pose The pose of the via point during the arc motion
03636 * @param end_pose The pose of the end point of the arc motion
03637 * @param a Acceleration
03638 * @param v Velocity
03639 * @param blend_radius Blend radius
03640 * @param duration Execution time
03641 * @return
03642 * \endenglish

```



```

03643     */
03644 int moveCircleWithAxisGroup(const std::vector<double> &via_pose,
03645                             const std::vector<double> &end_pose, double a,
03646                             double v, double blend_radius, double duration,
03647                             const std::string &group_name,
03648                             const std::vector<double> &extq);
03649
03650 /**
03651  * @ingroup MotionControl
03652  * \chinese
03653  * 设置圆弧路径模式
03654  *
03655  * @param mode 模式 \n
03656  * 0: 工具姿态相对于圆弧路径点坐标系保持不变 \n
03657  * 1: 姿态线性变化, 绕着空间定轴转动, 从起始点姿态变化到目标点姿态 \n
03658  * 2: 从起点姿态开始经过中间点姿态, 变化到目标点姿态
03659  * @return 成功返回 0; 失败返回错误码
03660  * AUBO_BAD_STATE
03661  * AUBO_BUSY
03662  * -AUBO_INVL_ARGUMENT
03663  * -AUBO_BAD_STATE
03664  *
03665  * @throws arcs::common_interface::AuboException
03666  *
03667  * @par Python 函数原型
03668  * setCirclePathMode(self: pyaubo_sdk.MotionControl, arg0: int) -> int
03669  *
03670  * @par Lua 函数原型
03671  * setCirclePathMode(mode: number) -> nil
03672  *
03673  * @par Lua 示例
03674  * setCirclePathMode(1)
03675  *
03676  * @par JSON-RPC 请求示例
03677  * {"jsonrpc": "2.0", "method": "robl.MotionControl.setCirclePathMode", "params": [0], "id": 1}
03678  *
03679  * @par JSON-RPC 响应示例
03680  * {"id": 1, "jsonrpc": "2.0", "result": 0}
03681  *
03682  * \endchinese
03683  * \english
03684  * Set circle path mode
03685  *
03686  * @param mode Mode \n
03687  * 0: Tool orientation remains unchanged relative to the arc path point
03688  * coordinate system \n 1: Orientation changes linearly, rotating around a
03689  * fixed axis in space, from the start orientation to the target orientation
03690  * \n 2: Orientation changes from the start orientation, through the via
03691  * point orientation, to the target orientation
03692  * @return Returns 0 on success; otherwise returns an error code
03693  * AUBO_BAD_STATE
03694  * AUBO_BUSY
03695  * -AUBO_INVL_ARGUMENT
03696  * -AUBO_BAD_STATE
03697  *
03698  * @throws arcs::common_interface::AuboException
03699  *
03700  * @par Python function prototype
03701  * setCirclePathMode(self: pyaubo_sdk.MotionControl, arg0: int) -> int
03702  *
03703  * @par Lua function prototype
03704  * setCirclePathMode(mode: number) -> nil
03705  *
03706  * @par Lua example
03707  * setCirclePathMode(1)
03708  *
03709  * @par JSON-RPC request example
03710  * {"jsonrpc": "2.0", "method": "robl.MotionControl.setCirclePathMode", "params": [0], "id": 1}
03711  *
03712  * @par JSON-RPC response example
03713  * {"id": 1, "jsonrpc": "2.0", "result": 0}
03714  *
03715  * \endenglish
03716  */
03717 int setCirclePathMode(int mode);
03718
03719 /**
03720  * @ingroup MotionControl
03721  * \chinese
03722  * 高级圆弧或者圆周运动
03723  *
03724  * @param param 圆周运动参数
03725  * @return 成功返回 0; 失败返回错误码
03726  * AUBO_BAD_STATE
03727  * AUBO_BUSY
03728  * -AUBO_INVL_ARGUMENT
03729  * -AUBO_BAD_STATE

```

```

03730      *
03731      * @throws arcs::common_interface::AuboException
03732      *
03733      * @par Python 函数原型
03734      * moveCircle2(self: pyaubo_sdk.MotionControl, arg0:
03735      * arcs::common_interface::CircleParameters) -> int
03736      *
03737      * @par Lua 函数原型
03738      * moveCircle2(param: table) -> nil
03739      *
03740      * @par Lua 示例
03741      * moveCircle2({0.440164,-0.00249391,0.398658,2.45651,0,1.570},
03742      {0.440164,0.166256,0.297599,2.45651,0,1.5708},1.2,0.25,0,0,0,0,0)
03743      *
03744      * @par JSON-RPC 请求示例
03745      * {"jsonrpc":"2.0","method":"robl.MotionControl.moveCircle2","params":[{"pose_via":
03746      [0.440164,-0.00249391,0.398658,2.45651,0,1.570},
03747      * "pose_to":
03748      [0.440164,0.166256,0.297599,2.45651,0,1.5708],"a":1.2,"v":0.25,"blend_radius":0,"duration":0,"helix":0,
03749      * "spiral":0,"direction":0,"loop_times":0}], "id":1}
03750      *
03751      * @par JSON-RPC 响应示例
03752      * {"id":1,"jsonrpc":"2.0","result":0}
03753      * \endchinese
03754      * \english
03755      * Advanced arc or circular motion
03756      *
03757      * @param param Circular motion parameters
03758      * @return Returns 0 on success; otherwise returns an error code:
03759      * AUBO_BAD_STATE
03760      * AUBO_BUSY
03761      * -AUBO_INVL_ARGUMENT
03762      * -AUBO_BAD_STATE
03763      *
03764      * @throws arcs::common_interface::AuboException
03765      *
03766      * @par Python function prototype
03767      * moveCircle2(self: pyaubo_sdk.MotionControl, arg0:
03768      * arcs::common_interface::CircleParameters) -> int
03769      *
03770      * @par Lua function prototype
03771      * moveCircle2(param: table) -> nil
03772      *
03773      * @par Lua example
03774      * moveCircle2({0.440164,-0.00249391,0.398658,2.45651,0,1.570},
03775      {0.440164,0.166256,0.297599,2.45651,0,1.5708},1.2,0.25,0,0,0,0,0)
03776      *
03777      * @par JSON-RPC request example
03778      * {"jsonrpc":"2.0","method":"robl.MotionControl.moveCircle2","params":[{"pose_via":
03779      [0.440164,-0.00249391,0.398658,2.45651,0,1.570},
03780      * "pose_to":
03781      [0.440164,0.166256,0.297599,2.45651,0,1.5708],"a":1.2,"v":0.25,"blend_radius":0,"duration":0,"helix":0,
03782      * "spiral":0,"direction":0,"loop_times":0}], "id":1}
03783      *
03784      * @par JSON-RPC response example
03785      * {"id":1,"jsonrpc":"2.0","result":0}
03786      * \endenglish
03787      */
03788      int moveCircle2(const CircleParameters &param);
03789      /**
03790      * @ingroup MotionControl
03791      * \chinese
03792      * 新建一个路径点缓存
03793      *
03794      * @param name 指定路径的名字
03795      * @param type 路径的类型 \n
03796      * 1: toppra 时间最优路径规划 \n
03797      * 2: cubic_spline(录制的轨迹) \n
03798      * 3: 关节 B 样条插值, 最少三个点
03799      * @param size 缓存区大小
03800      * @return 新建成功返回 AUBO_OK(0)
03801      *
03802      * @throws arcs::common_interface::AuboException
03803      *
03804      * @par Python 函数原型
03805      * pathBufferAlloc(self: pyaubo_sdk.MotionControl, arg0: str, arg1: int,
03806      * arg2: int) -> int
03807      *
03808      * @par Lua 函数原型
03809      * pathBufferAlloc(name: string, type: number, size: number) -> nil
03810      *
03811      * @par Lua 示例
03812      * pathBufferAlloc("traje_name",5,100)
03813      *
03814      * @par JSON-RPC 请求示例
03815      * {"jsonrpc":"2.0","method":"robl.MotionControl.pathBufferAlloc","params":["rec",2,3],"id":1}

```

```

03811      *
03812      * @par JSON-RPC 响应示例
03813      * {"id":1,"jsonrpc":"2.0","result":0}
03814      *
03815      * \endchinese
03816      * \english
03817      * Create a new path point buffer
03818      *
03819      * @param name Name of the path
03820      * @param type Type of the path \n
03821      * 1: toppra time-optimal path planning \n
03822      * 2: cubic_spline (recorded trajectory) \n
03823      * 3: Joint B-spline interpolation, at least three points
03824      * @param size Buffer size
03825      * @return Returns AUBO_OK(0) on success
03826      *
03827      * @throws arcs::common_interface::AuboException
03828      *
03829      * @par Python function prototype
03830      * pathBufferAlloc(self: pyaubo_sdk.MotionControl, arg0: str, arg1: int,
03831      * arg2: int) -> int
03832      *
03833      * @par Lua function prototype
03834      * pathBufferAlloc(name: string, type: number, size: number) -> nil
03835      *
03836      * @par Lua exmaple
03837      * pathBufferAlloc("traje_name",5,100)
03838      *
03839      * @par JSON-RPC request example
03840      * {"jsonrpc":"2.0","method":"rob1.MotionControl.pathBufferAlloc","params":["rec",2,3],"id":1}
03841      *
03842      * @par JSON-RPC response example
03843      * {"id":1,"jsonrpc":"2.0","result":0}
03844      *
03845      * \endenglish
03846      */
03847      int pathBufferAlloc(const std::string &name, int type, int size);
03848
03849      /**
03850      * @ingroup MotionControl
03851      * \chinese
03852      * 向路径缓存添加路点
03853      *
03854      * @param name 路径缓存的名字
03855      * @param waypoints 路点
03856      * @return 成功返回 0；失败返回错误码
03857      * -AUBO_INVL_ARGUMENT
03858      * -AUBO_BAD_STATE
03859      *
03860      * @throws arcs::common_interface::AuboException
03861      *
03862      * @par Python 函数原型
03863      * pathBufferAppend(self: pyaubo_sdk.MotionControl, arg0: str, arg1:
03864      * List[List[float]]) -> int
03865      *
03866      * @par Lua 函数原型
03867      * pathBufferAppend(name: string, waypoints: table) -> nil
03868      *
03869      * @par Lua 示例
03870      * pathBufferAppend("traje_name",{{-0.000000,0.000009,-0.000001,0.000002,0.000000},
03871      {-0.000001,0.000010,-0.000002,0.000000,0.000003,0.000002}})
03872      *
03873      * @par JSON-RPC 请求示例
03874      * {"jsonrpc":"2.0","method":"rob1.MotionControl.pathBufferAppend","params":["rec",
03875      [[{-0.000000,0.000009,-0.000001,0.000002,0.000000,0.000000},
03876      [-0.000001,0.000010,-0.000002,0.000000,0.000003,0.000002]]],"id":1}
03877      *
03878      * @par JSON-RPC 响应示例
03879      * {"id":1,"jsonrpc":"2.0","result":0}
03880      *
03881      * \endchinese
03882      * \english
03883      * Add waypoints to the path buffer
03884      *
03885      * @param name Name of the path buffer
03886      * @param waypoints Waypoints
03887      * @return Returns 0 on success; otherwise returns an error code
03888      * -AUBO_INVL_ARGUMENT
03889      * -AUBO_BAD_STATE
03890      *
03891      * @throws arcs::common_interface::AuboException
03892      *
03893      * @par Python function prototype
03894      * pathBufferAppend(self: pyaubo_sdk.MotionControl, arg0: str, arg1:
03895      * List[List[float]]) -> int
03896      *
03897      * @par Lua function prototype
03898      * pathBufferAppend(name: string, waypoints: table) -> nil

```

```

03895     *
03896     * @par Lua example
03897     * pathBufferAppend("traje_name",{-0.000000,0.000009,-0.000001,0.000002,0.000002,0.000000},
    {-0.000001,0.000010,-0.000002,0.000000,0.000003,0.000002})
03898     *
03899     * @par JSON-RPC request example
03900     * {"jsonrpc":"2.0","method":"rob1.MotionControl.pathBufferAppend","params":["rec",
    [[-0.000000,0.000009,-0.000001,0.000002,0.000002,0.000000],
    [-0.000001,0.000010,-0.000002,0.000000,0.000003,0.000002]]],"id":1}
03901     *
03902     * @par JSON-RPC response example
03903     * {"id":1,"jsonrpc":"2.0","result":0}
03904     * \endenglish
03905     */
03906     int pathBufferAppend(const std::string &name,
    const std::vector<std::vector<double>> &waypoints);
03907
03908 /**
03909  * @ingroup MotionControl
03910  * \chinese
03911  * 计算、优化等耗时操作，传入的参数相同时不会重新计算
03912  *
03913  * @param name 通过 pathBufferAlloc 新建的路径点缓存的名字
03914  * @param a 关节加速度限制，需要和机器人自由度大小相同，单位 rad/s^2
03915  * @param v 关节速度限制，需要和机器人自由度大小相同，单位 rad/s
03916  * @param t 时间 \n
03917  * pathBufferAlloc 这个接口分配内存的时候要指定类型，
03918  * 根据 pathBufferAlloc 这个接口的类型:\n
03919  * 类型为 1 时，表示运动持续时间 \n
03920  * 类型为 2 时，表示采样时间间隔 \n
03921  * 类型为 3 时:t 表示运动持续时间 \n
03922  * 若 t=0， 则由使用软件内部默认的时间（推荐使用）\n
03923  * 若 t!=0， t 需要设置为 路径点数 * 相邻点时间间隔 (points *
03924  * interval,interval >= 0.01)
03925  * @return 成功返回 0；失败返回错误码
03926  * -AUBO_INVL_ARGUMENT
03927  * -AUBO_BAD_STATE
03928  *
03929  * @throws arcs::common_interface::AuboException
03930  *
03931  * @par Python 函数原型
03932  * pathBufferEval(self: pyaubo_sdk.MotionControl, arg0: str, arg1:
03933  * List[float], arg2: List[float], arg3: float) -> int
03934  *
03935  * @par Lua 函数原型
03936  * pathBufferEval(name: string, a: table, v: table, t: number) -> nil
03937  *
03938  * @par Lua 示例
03939  * pathBufferEval("traje_name",{1,1,1,1,1,1},{1,1,1,1,1,1},0.02)
03940  *
03941  * @par JSON-RPC 请求示例
03942  * {"jsonrpc":"2.0","method":"rob1.MotionControl.pathBufferEval","params":["rec",[1,1,1,1,1,1],
    [1,1,1,1,1,1],0.02],"id":1}
03943
03944     *
03945     * @par JSON-RPC 响应示例
03946     * {"id":1,"jsonrpc":"2.0","result":0}
03947     * \endchinese
03948     * \english
03949     * Perform computation and optimization (time-consuming operations). If the
03950     * input parameters are the same, the calculation will not be repeated.
03951     *
03952     * @param name Name of the path point buffer created by pathBufferAlloc
03953     * @param a Joint acceleration limits, must match the robot DOF, unit:
03954     * rad/s^2
03955     * @param v Joint velocity limits, must match the robot DOF, unit: rad/s
03956     * @param t Time \n
03957     * When allocating memory with pathBufferAlloc, the type must be specified.
03958     * According to the type in pathBufferAlloc:\n
03959     * Type 1: t means motion duration\n
03960     * Type 2: t means sampling interval\n
03961     * Type 3: t means motion duration\n
03962     * If t=0, the internal default time is used (recommended)\n
03963     * If t!=0, t should be set to number_of_points *
03964     * interval_between_points (interval >= 0.01)
03965     * @return Returns 0 on success; otherwise returns an error code
03966     * -AUBO_INVL_ARGUMENT
03967     * -AUBO_BAD_STATE
03968     *
03969     * @throws arcs::common_interface::AuboException
03970     *
03971     * @par Python function prototype
03972     * pathBufferEval(self: pyaubo_sdk.MotionControl, arg0: str, arg1:
03973     * List[float], arg2: List[float], arg3: float) -> int
03974     *
03975     * @par Lua function prototype
03976     * pathBufferEval(name: string, a: table, v: table, t: number) -> nil
03977     *

```

```

03978      * @par Lua example
03979      * pathBufferEval("traje_name",{1,1,1,1,1},{1,1,1,1,1},0.02)
03980      *
03981      * @par JSON-RPC request example
03982      * {"jsonrpc":"2.0","method":"rob1.MotionControl.pathBufferEval","params":["rec",[1,1,1,1,1],
[1,1,1,1,1],0.02],"id":1}
03983      *
03984      * @par JSON-RPC response example
03985      * {"id":1,"jsonrpc":"2.0","result":0}
03986      * \endenglish
03987      */
03988      int pathBufferEval(const std::string &name, const std::vector<double> &a,
03989                        const std::vector<double> &v, double t);
03990
03991      /**
03992      * @ingroup MotionControl
03993      * \chinese
03994      * 指定名字的 buffer 是否有效
03995      *
03996      * buffer 需要满足三个条件有效: \n
03997      * 1、buffer 存在, 已经分配过内存 \n
03998      * 2、传进 buffer 的点要大于等于 buffer 的大小 \n
03999      * 3、要执行一次 pathBufferEval
04000      *
04001      * @param name buffer 的名字
04002      * @return 有效返回 true; 无效返回 false
04003      *
04004      * @throws arcs::common_interface::AuboException
04005      *
04006      * @par Python 函数原型
04007      * pathBufferValid(self: pyaubo_sdk.MotionControl, arg0: str) -> bool
04008      *
04009      * @par Lua 函数原型
04010      * pathBufferValid(name: string) -> boolean
04011      *
04012      * @par Lua 示例
04013      * buffer_status = pathBufferValid("traje_name")
04014      *
04015      * @par JSON-RPC 请求示例
04016      * {"jsonrpc":"2.0","method":"rob1.MotionControl.pathBufferValid","params":["rec"],"id":1}
04017      *
04018      * @par JSON-RPC 响应示例
04019      * {"id":1,"jsonrpc":"2.0","result":false}
04020      * \endchinese
04021      * \english
04022      * Whether the buffer with the specified name is valid
04023      *
04024      * The buffer must meet three conditions to be valid: \n
04025      * 1. The buffer exists and memory has been allocated \n
04026      * 2. The number of points passed into the buffer must be greater than or
04027      * equal to the buffer size \n
04028      * 3. pathBufferEval must be executed once
04029      *
04030      * @param name Name of the buffer
04031      * @return Returns true if valid; false otherwise
04032      *
04033      * @throws arcs::common_interface::AuboException
04034      *
04035      * @par Python function prototype
04036      * pathBufferValid(self: pyaubo_sdk.MotionControl, arg0: str) -> bool
04037      *
04038      * @par Lua function prototype
04039      * pathBufferValid(name: string) -> boolean
04040      *
04041      * @par Lua example
04042      * buffer_status = pathBufferValid("traje_name")
04043      *
04044      * @par JSON-RPC request example
04045      * {"jsonrpc":"2.0","method":"rob1.MotionControl.pathBufferValid","params":["rec"],"id":1}
04046      *
04047      * @par JSON-RPC response example
04048      * {"id":1,"jsonrpc":"2.0","result":false}
04049      * \endenglish
04050      */
04051      bool pathBufferValid(const std::string &name);
04052
04053      /**
04054      * @ingroup MotionControl
04055      * \chinese
04056      * 释放路径缓存
04057      *
04058      * @param name 缓存路径的名字
04059      * @return 成功返回 0; 失败返回错误码
04060      * -AUBO_INVL_ARGUMENT
04061      * -AUBO_BAD_STATE
04062      *
04063      * @throws arcs::common_interface::AuboException

```

```

04064 *
04065 * @par Python 函数原型
04066 * pathBufferFree(self: pyaubo_sdk.MotionControl, arg0: str) -> int
04067 *
04068 * @par Lua 函数原型
04069 * pathBufferFree(name: string) -> nil
04070 *
04071 * @par Lua 示例
04072 * pathBufferFree("traje_name")
04073 *
04074 * @par JSON-RPC 请求示例
04075 * {"jsonrpc": "2.0", "method": "rob1.MotionControl.pathBufferFree", "params": ["rec"], "id": 1}
04076 *
04077 * @par JSON-RPC 响应示例
04078 * {"id": 1, "jsonrpc": "2.0", "result": -5}
04079 *
04080 * \endchinese
04081 * \english
04082 * Release path buffer
04083 *
04084 * @param name Name of the path buffer
04085 * @return Returns 0 on success; otherwise returns an error code
04086 * -AUBO_INVL_ARGUMENT
04087 * -AUBO_BAD_STATE
04088 *
04089 * @throws arcs::common_interface::AuboException
04090 *
04091 * @par Python function prototype
04092 * pathBufferFree(self: pyaubo_sdk.MotionControl, arg0: str) -> int
04093 *
04094 * @par Lua function prototype
04095 * pathBufferFree(name: string) -> nil
04096 *
04097 * @par Lua example
04098 * pathBufferFree("traje_name")
04099 *
04100 * @par JSON-RPC request example
04101 * {"jsonrpc": "2.0", "method": "rob1.MotionControl.pathBufferFree", "params": ["rec"], "id": 1}
04102 *
04103 * @par JSON-RPC response example
04104 * {"id": 1, "jsonrpc": "2.0", "result": -5}
04105 *
04106 * \endenglish
04107 */
04108 int pathBufferFree(const std::string &name);
04109
04110 /**
04111 * @ingroup MotionControl
04112 * \chinese
04113 * 关节空间路径滤波器
04114 *
04115 * @brief pathBufferFilter
04116 *
04117 * @param name 缓存路径的名称
04118 * @param order 滤波器阶数 (一般取 2)
04119 * @param fd 截止频率, 越小越光滑, 但是延迟会大 (一般取 3-20)
04120 * @param fs 离散数据的采样频率 (一般取 20-500)
04121 *
04122 * @return 成功返回 0
04123 *
04124 * @throws arcs::common_interface::AuboException
04125 *
04126 * @par Python 函数原型
04127 * pathBufferFilter(self: pyaubo_sdk.MotionControl, arg0: str, arg1: int,
04128 * arg2: float, arg3: float) -> int
04129 *
04130 * @par Lua 函数原型
04131 * pathBufferFilter(name: string, order: number, fd: number, fs: number) ->
04132 * nil
04133 *
04134 * @par Lua 示例
04135 * pathBufferFilter("traje_name", 2, 5, 125)
04136 *
04137 * \endchinese
04138 * \english
04139 * Joint space path filter
04140 *
04141 * @brief pathBufferFilter
04142 *
04143 * @param name Name of the path buffer
04144 * @param order Filter order (usually 2)
04145 * @param fd Cutoff frequency, the smaller the smoother but with more delay
04146 * (usually 3-20)
04147 * @param fs Sampling frequency of discrete data (usually 20-500)
04148 *
04149 * @return Returns 0 on success
04150 *

```

```

04151     * @throws arcs::common_interface::AuboException
04152     *
04153     * @par Python function prototype
04154     * pathBufferFilter(self: pyaubo_sdk.MotionControl, arg0: str, arg1: int,
04155     * arg2: float, arg3: float) -> int
04156     *
04157     * @par Lua function prototype
04158     * pathBufferFilter(name: string, order: number, fd: number, fs:number) ->
04159     * nil
04160     *
04161     * @par Lua example
04162     * pathBufferFilter("traje_name",2,5,125)
04163     *
04164     * \endenglish
04165     */
04166 int pathBufferFilter(const std::string &name, int order, double fd,
04167                     double fs);
04168
04169 /**
04170  * @ingroup MotionControl
04171  * \chinese
04172  * 列出所有缓存路径的名字
04173  *
04174  * @return 返回所有缓存路径的名字
04175  *
04176  * @throws arcs::common_interface::AuboException
04177  *
04178  * @par Python 函数原型
04179  * pathBufferList(self: pyaubo_sdk.MotionControl) -> List[str]
04180  *
04181  * @par Lua 函数原型
04182  * pathBufferList() -> table
04183  *
04184  * @par Lua 示例
04185  * Buffer_table = pathBufferList()
04186  *
04187  * @par JSON-RPC 请求示例
04188  * {"jsonrpc":"2.0","method":"rob1.MotionControl.pathBufferList","params":[],"id":1}
04189  *
04190  * @par JSON-RPC 响应示例
04191  * {"id":1,"jsonrpc":"2.0","result":[]}
04192  *
04193  * \endchinese
04194  * \english
04195  * List all cached path names
04196  *
04197  * @return Returns all cached path names
04198  *
04199  * @throws arcs::common_interface::AuboException
04200  *
04201  * @par Python function prototype
04202  * pathBufferList(self: pyaubo_sdk.MotionControl) -> List[str]
04203  *
04204  * @par Lua function prototype
04205  * pathBufferList() -> table
04206  *
04207  * @par Lua example
04208  * Buffer_table = pathBufferList()
04209  *
04210  * @par JSON-RPC request example
04211  * {"jsonrpc":"2.0","method":"rob1.MotionControl.pathBufferList","params":[],"id":1}
04212  *
04213  * @par JSON-RPC response example
04214  * {"id":1,"jsonrpc":"2.0","result":[]}
04215  *
04216  * \endenglish
04217  */
04218 std::vector<std::string> pathBufferList();
04219
04220 /**
04221  * @ingroup MotionControl
04222  * \chinese
04223  * 执行缓存的路径
04224  *
04225  * @param name 缓存路径的名字
04226  * @return 成功返回 0；失败返回错误码
04227  * -AUBO_INVL_ARGUMENT
04228  * -AUBO_BAD_STATE
04229  *
04230  * @throws arcs::common_interface::AuboException
04231  *
04232  * @par Python 函数原型
04233  * movePathBuffer(self: pyaubo_sdk.MotionControl, arg0: str) -> int
04234  *
04235  * @par Lua 函数原型
04236  * movePathBuffer(name: string) -> nil
04237  *

```



```

04238     * @par Lua 示例
04239     * movePathBuffer("traje_name")
04240     *
04241     * @par JSON-RPC 请求示例
04242     * {"jsonrpc": "2.0", "method": "rob1.MotionControl.movePathBuffer", "params": ["rec"], "id": 1}
04243     *
04244     * @par JSON-RPC 响应示例
04245     * {"id": 1, "jsonrpc": "2.0", "result": 0}
04246     * \endchinese
04247     * \english
04248     * Execute the cached path
04249     *
04250     * @param name Name of the cached path
04251     * @return Returns 0 on success; otherwise returns an error code
04252     * -AUBO_INVL_ARGUMENT
04253     * -AUBO_BAD_STATE
04254     *
04255     * @throws arcs::common_interface::AuboException
04256     *
04257     * @par Python function prototype
04258     * movePathBuffer(self: pyaubo_sdk.MotionControl, arg0: str) -> int
04259     *
04260     * @par Lua function prototype
04261     * movePathBuffer(name: string) -> nil
04262     *
04263     * @par Lua example
04264     * movePathBuffer("traje_name")
04265     *
04266     * @par JSON-RPC request example
04267     * {"jsonrpc": "2.0", "method": "rob1.MotionControl.movePathBuffer", "params": ["rec"], "id": 1}
04268     *
04269     * @par JSON-RPC response example
04270     * {"id": 1, "jsonrpc": "2.0", "result": 0}
04271     * \endenglish
04272     */
04273 int movePathBuffer(const std::string &name);
04274
04275 /**
04276     * @ingroup MotionControl
04277     * \chinese
04278     * 相贯线接口
04279     *
04280     * @param pose 由三个示教位姿组成（首先需要运动到起始点，起始点的要求：过主圆柱
04281     * 柱体轴心且与子圆柱体轴线平行的平面与子圆柱体在底部的交点）
04282     * p1: 过子圆柱体轴线且与大圆柱体轴线平行的平面，与小圆柱体在左侧顶部的交点
04283     * p2: 过子圆柱体轴线且与大圆柱体轴线平行的平面，与大圆柱体在左侧底部的交点
04284     * p3: 过子圆柱体轴线且与大圆柱体轴线平行的平面，与大圆柱体在右侧底部的交点
04285     * @param v 速度
04286     * @param a 加速度
04287     * @param main_pipe_radius 主圆柱体半径
04288     * @param sub_pipe_radius 子圆柱体半径
04289     * @param normal_distance 两圆柱体轴线距离
04290     * @param normal_alpha 两圆柱体轴线的夹角
04291     *
04292     * @return 成功返回 0；失败返回错误码
04293     * AUBO_BUSY
04294     * AUBO_BAD_STATE
04295     * -AUBO_INVL_ARGUMENT
04296     * -AUBO_BAD_STATE
04297     *
04298     * @throws arcs::common_interface::AuboException
04299     *
04300     * @par Python 函数原型
04301     * moveIntersection(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
04302     * float, arg2: float, arg3: float, arg4: float, arg5: float, arg6: float)
04303     * -> int
04304     *
04305     * @par Lua 函数原型
04306     * moveIntersection(poses: table, a: number, v: number,
04307     * main_pipe_radius: number, sub_pipe_radius: number, normal_distance:
04308     * number, normal_alpha: number) -> nil \endchinese \english Intersection
04309     * interface
04310     *
04311     * @param poses Consists of three taught poses (first, move to the starting
04312     * point. The starting point requirement: the intersection of the plane
04313     * passing through the main cylinder axis and parallel to the sub-cylinder
04314     * axis with the sub-cylinder at the bottom) p1: Intersection of the plane
04315     * passing through the sub-cylinder axis and parallel to the main cylinder
04316     * axis with the sub-cylinder at the left top p2: Intersection of the plane
04317     * passing through the sub-cylinder axis and parallel to the main cylinder
04318     * axis with the main cylinder at the left bottom p3: Intersection of the
04319     * plane passing through the sub-cylinder axis and parallel to the main
04320     * cylinder axis with the main cylinder at the right bottom
04321     * @param v Velocity
04322     * @param a Acceleration
04323     * @param main_pipe_radius Main cylinder radius
04324     * @param sub_pipe_radius Sub cylinder radius

```



```

04325     * @param normal_distance Distance between the two cylinder axes
04326     * @param normal_alpha Angle between the two cylinder axes
04327     *
04328     * @return Returns 0 on success; otherwise returns an error code
04329     * AUBO_BUSY
04330     * AUBO_BAD_STATE
04331     * -AUBO_INVL_ARGUMENT
04332     * -AUBO_BAD_STATE
04333     *
04334     * @throws arcs::common_interface::AuboException
04335     *
04336     * @par Python function prototype
04337     * moveIntersection(self: pyaubo_sdk.MotionControl, arg0: List[float], arg1:
04338     * float, arg2: float, arg3: float, arg4: float, arg5: float, arg6: float)
04339     * -> int
04340     *
04341     * @par Lua function prototype
04342     * moveIntersection(poses: table, a: number, v: number,
04343     * main_pipe_radius: number, sub_pipe_radius: number, normal_distance:
04344     * number, normal_alpha: number) -> nil \endenglish
04345     */
04346 int moveIntersection(const std::vector<std::vector<double>> &poses,
04347                     double a, double v, double main_pipe_radius,
04348                     double sub_pipe_radius, double normal_distance,
04349                     double normal_alpha);
04350 /**
04351     * @ingroup MotionControl
04352     * \chinese
04353     * 关节空间停止运动
04354     *
04355     * @param acc 关节加速度, 单位: rad/s^2
04356     * @return 成功返回 0; 失败返回错误码
04357     * AUBO_BUSY
04358     * AUBO_BAD_STATE
04359     * -AUBO_INVL_ARGUMENT
04360     * -AUBO_BAD_STATE
04361     *
04362     * @throws arcs::common_interface::AuboException
04363     *
04364     * @par Python 函数原型
04365     * stopJoint(self: pyaubo_sdk.MotionControl, arg0: float) -> int
04366     *
04367     * @par Lua 函数原型
04368     * stopJoint(acc: number) -> nil
04369     *
04370     * @par Lua 示例
04371     * stopJoint(2)
04372     *
04373     * @par JSON-RPC 请求示例
04374     * {"jsonrpc": "2.0", "method": "robl.MotionControl.stopJoint", "params": [31], "id": 1}
04375     *
04376     * @par JSON-RPC 响应示例
04377     * {"id": 1, "jsonrpc": "2.0", "result": 0}
04378     *
04379     * \endchinese
04380     * \english
04381     * Stop motion in joint space
04382     *
04383     * @param acc Joint acceleration, unit: rad/s^2
04384     * @return Returns 0 on success; otherwise returns an error code
04385     * AUBO_BUSY
04386     * AUBO_BAD_STATE
04387     * -AUBO_INVL_ARGUMENT
04388     * -AUBO_BAD_STATE
04389     *
04390     * @throws arcs::common_interface::AuboException
04391     *
04392     * @par Python function prototype
04393     * stopJoint(self: pyaubo_sdk.MotionControl, arg0: float) -> int
04394     *
04395     * @par Lua function prototype
04396     * stopJoint(acc: number) -> nil
04397     *
04398     * @par Lua example
04399     * stopJoint(2)
04400     *
04401     * @par JSON-RPC request example
04402     * {"jsonrpc": "2.0", "method": "robl.MotionControl.stopJoint", "params": [31], "id": 1}
04403     *
04404     * @par JSON-RPC response example
04405     * {"id": 1, "jsonrpc": "2.0", "result": 0}
04406     *
04407     * \endenglish
04408     */
04409 int stopJoint(double acc);
04410
04411 /**

```

```

04412     * @ingroup MotionControl
04413     * \chinese
04414     * 关节空间停止运动（机械臂运行工程时发生碰撞，通过 resumeSpeedJoint 接口移动到安全位置后需要停止时调用此接口）
04415     *
04416     * @param acc 关节加速度，单位：rad/s^2
04417     * @return 成功返回 0；失败返回错误码
04418     * AUBO_BUSY
04419     * AUBO_BAD_STATE
04420     * -AUBO_INVL_ARGUMENT
04421     * -AUBO_BAD_STATE
04422     *
04423     * @throws arcs::common_interface::AuboException
04424     *
04425     * @par Python 函数原型
04426     * resumeStopJoint(self: pyaubo_sdk.MotionControl, arg0: float) -> int
04427     *
04428     * @par Lua 函数原型
04429     * resumeStopJoint(acc: number) -> nil
04430     *
04431     * @par Lua 示例
04432     * resumeStopJoint(31)
04433     *
04434     * @par JSON-RPC 请求示例
04435     * {"jsonrpc": "2.0", "method": "rob1.MotionControl.resumeStopJoint", "params": [31], "id": 1}
04436     *
04437     * @par JSON-RPC 响应示例
04438     * {"id": 1, "jsonrpc": "2.0", "result": -1}
04439     *
04440     * \endchinese
04441     * \english
04442     * Stop motion in joint space (used after moving to a safe position via
04443     * resumeSpeedJoint following a collision during process execution)
04444     *
04445     * @param acc Joint acceleration, unit: rad/s^2
04446     * @return Returns 0 on success; otherwise returns an error code
04447     * AUBO_BUSY
04448     * AUBO_BAD_STATE
04449     * -AUBO_INVL_ARGUMENT
04450     * -AUBO_BAD_STATE
04451     *
04452     * @throws arcs::common_interface::AuboException
04453     *
04454     * @par Python function prototype
04455     * resumeStopJoint(self: pyaubo_sdk.MotionControl, arg0: float) -> int
04456     *
04457     * @par Lua function prototype
04458     * resumeStopJoint(acc: number) -> nil
04459     *
04460     * @par Lua example
04461     * resumeStopJoint(31)
04462     *
04463     * @par JSON-RPC request example
04464     * {"jsonrpc": "2.0", "method": "rob1.MotionControl.resumeStopJoint", "params": [31], "id": 1}
04465     *
04466     * @par JSON-RPC response example
04467     * {"id": 1, "jsonrpc": "2.0", "result": -1}
04468     *
04469     * \endenglish
04470     */
04471 int resumeStopJoint(double acc);
04472
04473 /**
04474     * @ingroup MotionControl
04475     * \chinese
04476     * 停止 moveLine/moveCircle 等在笛卡尔空间的运动
04477     *
04478     * @param acc 工具加速度，单位：m/s^2
04479     * @param acc_rot
04480     * @return 成功返回 0；失败返回错误码
04481     * AUBO_BUSY
04482     * AUBO_BAD_STATE
04483     * -AUBO_INVL_ARGUMENT
04484     * -AUBO_BAD_STATE
04485     *
04486     * @throws arcs::common_interface::AuboException
04487     *
04488     * @par Python 函数原型
04489     * stopLine(self: pyaubo_sdk.MotionControl, arg0: float, arg1: float) -> int
04490     *
04491     * @par Lua 函数原型
04492     * stopLine(acc: number, acc_rot: number) -> nil
04493     *
04494     * @par Lua 示例
04495     * stopLine(10,10)
04496     *
04497     * @par JSON-RPC 请求示例
04498     * {"jsonrpc": "2.0", "method": "rob1.MotionControl.stopLine", "params": [10,10], "id": 1}

```

```

04499      *
04500      * @par JSON-RPC 响应示例
04501      * {"id":1,"jsonrpc":"2.0","result":0}
04502      *
04503      * \endchinese
04504      * \english
04505      * Stop motions in Cartesian space such as moveLine/moveCircle.
04506      *
04507      * @param acc Tool acceleration, unit: m/s^2
04508      * @param acc_rot
04509      * @return Returns 0 on success; otherwise returns an error code
04510      * AUBO_BUSY
04511      * AUBO_BAD_STATE
04512      * -AUBO_INVL_ARGUMENT
04513      * -AUBO_BAD_STATE
04514      *
04515      * @throws arcs::common_interface::AuboException
04516      *
04517      * @par Python function prototype
04518      * stopLine(self: pyaubo_sdk.MotionControl, arg0: float, arg1: float) -> int
04519      *
04520      * @par Lua function prototype
04521      * stopLine(acc: number, acc_rot: number) -> nil
04522      *
04523      * @par Lua example
04524      * stopLine(10,10)
04525      *
04526      * @par JSON-RPC request example
04527      * {"jsonrpc":"2.0","method":"rob1.MotionControl.stopLine","params":[10,10],"id":1}
04528      *
04529      * @par JSON-RPC response example
04530      * {"id":1,"jsonrpc":"2.0","result":0}
04531      *
04532      * \endenglish
04533      */
04534      int stopLine(double acc, double acc_rot);
04535
04536      /**
04537      * @ingroup MotionControl
04538      * \chinese
04539      * 笛卡尔空间停止运动（机械臂运行工程时发生碰撞，通过 resumeSpeedLine 接口移动到安全位置后需要停止时调用此接口）
04540      *
04541      * @param acc 位置加速度，单位：m/s^2
04542      * @param acc_rot 姿态加速度，单位：rad/s^2
04543      * @return 成功返回 0；失败返回错误码
04544      * AUBO_BUSY
04545      * AUBO_BAD_STATE
04546      * -AUBO_INVL_ARGUMENT
04547      * -AUBO_BAD_STATE
04548      *
04549      * @throws arcs::common_interface::AuboException
04550      *
04551      * @par Python 函数原型
04552      * resumeStopLine(self: pyaubo_sdk.MotionControl, arg0: float, arg1: float)
04553      * -> int
04554      *
04555      * @par Lua 函数原型
04556      * resumeStopLine(acc: number, acc_rot: number) -> nil
04557      *
04558      * @par Lua 示例
04559      * resumeStopLine(10,10)
04560      *
04561      * @par JSON-RPC 请求示例
04562      * {"jsonrpc":"2.0","method":"rob1.MotionControl.resumeStopLine","params":[10,10],"id":1}
04563      *
04564      * @par JSON-RPC 响应示例
04565      * {"id":1,"jsonrpc":"2.0","result":-1}
04566      *
04567      * \endchinese
04568      * \english
04569      * Stop motion in Cartesian space (used after moving to a safe position via
04570      * resumeSpeedLine following a collision during process execution)
04571      *
04572      * @param acc Position acceleration, unit: m/s^2
04573      * @param acc_rot Orientation acceleration, unit: rad/s^2
04574      * @return Returns 0 on success; otherwise returns an error code
04575      * AUBO_BUSY
04576      * AUBO_BAD_STATE
04577      * -AUBO_INVL_ARGUMENT
04578      * -AUBO_BAD_STATE
04579      *
04580      * @throws arcs::common_interface::AuboException
04581      *
04582      * @par Python function prototype
04583      * resumeStopLine(self: pyaubo_sdk.MotionControl, arg0: float, arg1: float)
04584      * -> int
04585      *

```

```

04586 * @par Lua function prototype
04587 * resumeStopLine(acc: number, acc_rot: number) -> nil
04588 *
04589 * @par Lua example
04590 * resumeStopLine(10,10)
04591 *
04592 * @par JSON-RPC request example
04593 * {"jsonrpc": "2.0", "method": "rob1.MotionControl.resumeStopLine", "params": [10,10], "id": 1}
04594 *
04595 * @par JSON-RPC response example
04596 * {"id": 1, "jsonrpc": "2.0", "result": -1}
04597 *
04598 * \endenglish
04599 */
04600 int resumeStopLine(double acc, double acc_rot);
04601
04602 /**
04603 * @ingroup MotionControl
04604 * \chinese
04605 *
04606 * \warning 调用 weaveStart 前, 需提前开启 RuntimeMachine 规划器 start。
04607 *
04608 * 开始摆动: 在 weaveStart 与 weaveEnd 之间的
04609 * moveLine/moveCircle/moveProcess 根据 params 执行摆动。
04610 *
04611 * @param params Json 字符串。
04612 *
04613 * @par 自 v0.29 起的字段
04614 * - "type": <string> —波形类型: "SINE" |
04615 * "SPIRAL" | "TRIANGLE" | "TRAPEZOIDAL" (区分大小写)。
04616 * - "step": <number> —
04617 * 相邻摆动采样的弧长步长, 单位 m。
04618 * - "amplitude": [<number>, <number>] —摆弧幅度左右侧, 单位 m。
04619 * - "hold_distance": [<number>, <number>] —
04620 * 在幅度峰值处沿路径的停留距离, 单位 m。
04621 * - "hold_time": [<number>, <number>] —
04622 * 在幅度峰值处的停留时间, 单位 s。
04623 * - "angle": <number> —
04624 * 叠加方向与法平面的夹角, 单位 rad。
04625 * - "direction": <integer> —
04626 * 起始摆动方向: 0= 上方, 1= 下方。
04627 * - "movep_uniform_vel": <bool> — moveProcess
04628 * 匀速规划, 布尔量。
04629 *
04630 * @par 自 v0.31 起
04631 * - 与 v0.29 相同。
04632 *
04633 * @par 自 v0.32 起新增/变更
04634 * - "angle": [<number>, <number>] —运动叠加方向与法平面的夹角, 单位 rad。
04635 * - "frequency": <number> —摆动频率, 单位 Hz (范围 [0.1, 5])。
04636 * - "ori_weave_range": [<number>, <number>, <number>] —姿态摆动范围
04637 * [x, y, z], 单位 rad; 常与阿基米德螺旋用于力控搜孔。
04638 * - "ori_weave_frequency": [<number>, <number>, <number>] —姿态摆动频率
04639 * [x, y, z], 单位 m-1 (按路径弧长计; 范围 [0, 1])。
04640 * - "adjust_cycle_num": <integer> —
04641 * 动态调节频率/幅值的过渡段数量。
04642 * - "azimuth": <number> —摆焊波形方位角, 单位 rad。
04643 * - "ridge_height": <number> —中心上升高度, 单位 m。
04644 *
04645 * \warning 可使用关系 **frequency = vel / step** 与外部速度 vel 关联 (vel
04646 * 单位 m * s-1)。
04647 * \warning **hold_time** 仅在 **SINE** 下可用, 且可左右不对称。
04648 * \warning "hold_distance"/"hold_time" 互斥, 仅一个生效
04649 *
04650 * \note
04651 * - 数组约定: "amplitude"/"hold_distance"/"hold_time" 为
04652 * [left, right]; "ori_weave_*" 为姿态 [x, y, z]。
04653 * - 所有角度均为弧度 (rad)。
04654 *
04655 * @return 成功返回 0; 失败返回错误码
04656 * - AUBO_BUSY
04657 * - AUBO_BAD_STATE
04658 * - AUBO_INVL_ARGUMENT
04659 * - AUBO_BAD_STATE
04660 *
04661 * @throws arcs::common_interface::AuboException
04662 *
04663 * @par 示例 v0.29/v0.31
04664 * \code{.json}
04665 * params = {
04666 *   "type": "SINE",
04667 *   "step": 0.005,
04668 *   "amplitude": [0.01, 0.01],
04669 *   "hold_distance": [0.001, 0.001],
04670 *   "hold_time": [0, 0],
04671 *   "angle": 0,
04672 *   "direction": 0,

```

```

04673     *      "movep_uniform_vel": false
04674     * }
04675     * \endcode
04676     *
04677     * @par 示例 v0.32
04678     * \code{.json}
04679     * params = {
04680     *     "type" : "SINE", // "SINE" "SPIRAL" "TRIANGLE" "SAWTOOTH" "CRESCENT"
04681     *     "step" : 0.0, // 可用 frequency = vel / step (vel: m·s-1)
04682     *     "frequency" : 0.0, // [0.1, 5] Hz
04683     *     "amplitude" : [0.01, 0.01], // m
04684     *     "hold_distance" : [0.001, 0.001], // m
04685     *     "hold_time" : [0, 0], // s (仅 SINE)
04686     *     "angle" : [0, 0], // rad
04687     *     "direction" : 0,
04688     *     "ori_weave_range" : [0.001, 0.001, 0.001], // rad
04689     *     "ori_weave_frequency" : [0.001, 0.001, 0.001], // m-1
04690     *     "adjust_cycle_num": 0,
04691     *     "azimuth" : 0, // rad
04692     *     "ridge_height" : 0 // m
04693     * }
04694     * \endcode
04695     *
04696     * @par Python 函数原型
04697     * \code{.py}
04698     * weaveStart(self: pyaubo_sdk.MotionControl, arg0: str) -> int
04699     * \endcode
04700     *
04701     * @par Python 示例
04702     * \code{.py}
04703     * robot_name = rpc_cli.getRobotNames()[0]
04704     * robot_interface = rpc_cli.getRobotInterface(robot_name)
04705     * robot_interface.getMotionControl().weaveStart(params)
04706     * \endcode
04707     *
04708     * @par Lua 函数原型
04709     * \code{.lua}
04710     * weaveStart(params: string) -> nil
04711     * \endcode
04712     *
04713     * @par Lua 示例 (v0.29/0.31)
04714     * \code{.lua}
04715     * weaveStart("{\"type\":\"SINE\",\"step\":0.005,\"amplitude\":[0.01,0.01],\"hold_distance\":
04716     * [0.001,0.001],\"hold_time\":[0,0],\"angle\":0,\"direction\":0,\"movep_uniform_vel\":false}")
04717     * \endcode
04718     *
04719     * @par Lua 示例 (v0.32)
04720     * \code{.lua}
04721     * weaveStart("{\"type\":\"SINE\",\"step\":0.0,\"frequency\":2.0,\"amplitude\":[0.01,0.01],\"hold_distance\":
04722     * [0.001,0.001],\"hold_time\":[0,0],\"angle\":0,\"direction\":0,\"ori_weave_range\":
04723     * [0.001,0.001,0.001],\"ori_weave_frequency\":
04724     * [0.001,0.001,0.001],\"adjust_cycle_num\":0,\"azimuth\":0,\"ridge_height\":0}")
04725     * \endcode
04726     *
04727     * @par JSON-RPC 请求示例 (v0.29/0.31)
04728     * \code{.json}
04729     * {"jsonrpc":"2.0","method":"rob1.MotionControl.weaveStart","params":
04730     * [{"type":"SINE","step":0.005,"amplitude":[0.01,0.01],\"hold_distance\":[0.001,0.001],\"hold_time\":
04731     * [0,0],\"angle\":0,\"direction\":0,\"movep_uniform_vel\":false}],\"id\":1}
04732     * \endcode
04733     *
04734     * @par JSON-RPC 请求示例 (v0.32)
04735     * \code{.json}
04736     * {"jsonrpc":"2.0","method":"rob1.MotionControl.weaveStart","params":
04737     * [{"type":"SINE","step":0.0,\"frequency\":2.0,\"amplitude\":[0.01,0.01],\"hold_distance\":
04738     * [0.001,0.001],\"hold_time\":[0,0],\"angle\":0,\"direction\":0,\"ori_weave_range\":
04739     * [0.001,0.001,0.001],\"ori_weave_frequency\":
04740     * [0.001,0.001,0.001],\"adjust_cycle_num\":0,\"azimuth\":0,\"ridge_height\":0}],\"id\":1}
04741     * \endcode
04742     *
04743     * @par JSON-RPC 响应示例
04744     * \code{.json}
04745     * {"id":1,"jsonrpc":"2.0","result":0}
04746     * \endcode
04747     * \endchinese
04748     *
04749     * \english
04750     * Start weaving: between weaveStart and weaveEnd,
04751     * moveLine/moveCircle/moveProcess follows params.
04752     *
04753     * \warning Start RuntimeMachine before calling weaveStart.
04754     *
04755     * @param params Json string.
04756     *
04757     * @par Since v0.29
04758     * - "type": <string> — Waveform: "SINE" | "SPIRAL"
04759     * | "TRIANGLE" | "TRAPEZOIDAL" (case-sensitive).

```

```

04750 * - "step": <number>                - Arc-length sampling step,
04751 * m.
04752 * - "amplitude": [<number>,<number>] - Left/right weave amplitude,
04753 * m.
04754 * - "hold_distance": [<number>,<number>] - Dwell distance at peak
04755 * amplitude along path, m.
04756 * - "hold_time": [<number>,<number>] - Dwell time at peak
04757 * amplitude, s.
04758 * - "angle": <number>                - Angle to normal plane, rad.
04759 * - "direction": <integer>           - Initial direction: 0=above
04760 * path, 1=below path.
04761 * - "movep_uniform_vel": <bool>      - Uniform-velocity planning
04762 * (boolean value).
04763 *
04764 * @par Since v0.31
04765 * - Same as v0.29.
04766 *
04767 * @par New/changed in v0.32
04768 * - "frequency": <number>            - Weave frequency, Hz (range
04769 * [0.1, 5]).
04770 * - "ori_weave_range": [<number>,<number>,<number>] - Attitude weave
04771 * range [x,y,z], rad; often used with Archimedean spiral for force-guided
04772 * hole finding.
04773 * - "ori_weave_frequency": [<number>,<number>,<number>] - Attitude weave
04774 * frequency [x,y,z], m-1 (per unit arc; range [0, 1]).
04775 * - "adjust_cycle_num": <integer>    - Number of transition
04776 * cycles.
04777 * - "azimuth": <number>              - Waveform azimuth, rad.
04778 * - "ridge_height": <number>        - Center raise height, m.
04779 *
04780 * \warning **hold_time** is supported for **SINE** only and may be
04781 * asymmetric. \warning You may relate **frequency = vel / step** when an
04782 * external path speed vel (m·s-1) is available.
04783 *
04784 * \note
04785 * - Two-element arrays are [left,right]; three-element arrays are attitude
04786 * [x,y,z].
04787 * - All angles are in radians (rad).
04788 *
04789 * @return Returns 0 on success; otherwise returns an error code:
04790 * - AUBO_BUSY
04791 * - AUBO_BAD_STATE
04792 * - AUBO_INVL_ARGUMENT
04793 * - AUBO_BAD_STATE
04794 *
04795 * @throws arcs::common_interface::AuboException
04796 *
04797 * @par Example v0.29/v0.31
04798 * \code{.json}
04799 * params = {
04800 *     "type": "SINE",
04801 *     "step": 0.005,
04802 *     "amplitude": [0.01, 0.01],
04803 *     "hold_distance": [0.001, 0.001],
04804 *     "hold_time": [0, 0],
04805 *     "angle": 0,
04806 *     "direction": 0,
04807 *     "movep_uniform_vel": false
04808 * }
04809 * \endcode
04810 *
04811 * @par Example v0.32
04812 * \code{.json}
04813 * params = {
04814 *     "type" : "SINE", // "SINE" "SPIRAL" "TRIANGLE" "TRAPEZOIDAL"
04815 *     "step" : 0.0,    // frequency = vel / step (vel: m·s-1)
04816 *     "frequency" : 0.0, // [0.1, 5] Hz
04817 *     "amplitude" : [0.01, 0.01], // m
04818 *     "hold_distance" : [0.001, 0.001], // m
04819 *     "hold_time" : [0, 0], // s (SINE only)
04820 *     "angle" : 0, // rad
04821 *     "direction" : 0,
04822 *     "ori_weave_range" : [0.001, 0.001, 0.001], // rad
04823 *     "ori_weave_frequency": [0.001, 0.001, 0.001], // m-1
04824 *     "adjust_cycle_num": 0,
04825 *     "azimuth" : 0, // rad
04826 *     "ridge_height" : 0 // m
04827 * }
04828 * \endcode
04829 *
04830 * @par Python function prototype
04831 * \code{.py}
04832 * weaveStart(self: pyaubo_sdk.MotionControl, arg0: str) -> int
04833 * \endcode
04834 *
04835 * @par Python example
04836 * \code{.py}

```

```

04837     * robot_name = rpc_cli.getRobotNames()[0]
04838     * robot_interface = rpc_cli.getRobotInterface(robot_name)
04839     * robot_interface.getMotionControl().weaveStart(params)
04840     * \endcode
04841     *
04842     * @par Lua function prototype
04843     * \code{.lua}
04844     * weaveStart(params: string) -> nil
04845     * \endcode
04846     *
04847     * @par Lua example (v0.29/0.31)
04848     * \code{.lua}
04849     * weaveStart("{ \"type\": \"SINE\", \"step\": 0.005, \"amplitude\": [0.01, 0.01], \"hold_distance\":
[0.001, 0.001], \"hold_time\": [0, 0], \"angle\": 0, \"direction\": 0, \"movep_uniform_vel\": false}")
04850     * \endcode
04851     *
04852     * @par Lua example (v0.32)
04853     * \code{.lua}
04854     * weaveStart("{ \"type\": \"SINE\", \"step\": 0.0, \"frequency\": 2.0, \"amplitude\": [0.01, 0.01], \"hold_distance\":
[0.001, 0.001], \"hold_time\": [0, 0], \"angle\": 0, \"direction\": 0, \"ori_weave_range\":
[0.001, 0.001, 0.001], \"ori_weave_frequency\":
[0.001, 0.001, 0.001], \"adjust_cycle_num\": 0, \"azimuth\": 0, \"ridge_height\": 0}")
04855     * \endcode
04856     *
04857     * @par JSON-RPC request example (v0.29/0.31)
04858     * \code{.json}
04859     * {"jsonrpc": "2.0", "method": "rob1.MotionControl.weaveStart", "params":
["{ \"type\": \"SINE\", \"step\": 0.005, \"amplitude\": [0.01, 0.01], \"hold_distance\": [0.001, 0.001], \"hold_time\":
[0, 0], \"angle\": 0, \"direction\": 0, \"movep_uniform_vel\": false}"], "id": 1}
04860     * \endcode
04861     *
04862     * @par JSON-RPC request example (v0.32)
04863     * \code{.json}
04864     * {"jsonrpc": "2.0", "method": "rob1.MotionControl.weaveStart", "params":
["{ \"type\": \"SINE\", \"step\": 0.0, \"frequency\": 2.0, \"amplitude\": [0.01, 0.01], \"hold_distance\":
[0.001, 0.001], \"hold_time\": [0, 0], \"angle\": 0, \"direction\": 0, \"ori_weave_range\":
[0.001, 0.001, 0.001], \"ori_weave_frequency\":
[0.001, 0.001, 0.001], \"adjust_cycle_num\": 0, \"azimuth\": 0, \"ridge_height\": 0}"], "id": 1}
04865     * \endcode
04866     *
04867     * @par JSON-RPC response example
04868     * \code{.json}
04869     * {"id": 1, "jsonrpc": "2.0", "result": 0}
04870     * \endcode
04871     * \endenglish
04872     */
04873     int weaveStart(const std::string &params);
04874
04875     /**
04876     * @ingroup MotionControl
04877     * \chinese
04878     * 结束摆动
04879     *
04880     * @return 成功返回 0；失败返回错误码
04881     * AUBO_BUSY
04882     * AUBO_BAD_STATE
04883     * -AUBO_BAD_STATE
04884     *
04885     * @throws arcs::common_interface::AuboException
04886     *
04887     * @par JSON-RPC 请求示例
04888     * {"jsonrpc": "2.0", "method": "rob1.MotionControl.weaveEnd", "params": [], "id": 1}
04889     *
04890     * @par JSON-RPC 响应示例
04891     * {"id": 1, "jsonrpc": "2.0", "result": 0}
04892     *
04893     * \endchinese
04894     * \english
04895     * End weaving
04896     *
04897     * @return Returns 0 on success; otherwise returns an error code
04898     * AUBO_BUSY
04899     * AUBO_BAD_STATE
04900     * -AUBO_BAD_STATE
04901     *
04902     * @throws arcs::common_interface::AuboException
04903     *
04904     * @par JSON-RPC request example
04905     * {"jsonrpc": "2.0", "method": "rob1.MotionControl.weaveEnd", "params": [], "id": 1}
04906     *
04907     * @par JSON-RPC response example
04908     * {"id": 1, "jsonrpc": "2.0", "result": 0}
04909     *
04910     * \endenglish
04911     */
04912     int weaveEnd();
04913

```



```

04914 /**
04915  * @ingroup MotionControl
04916  * \chinese
04917  * 设置未来路径上点的采样时间间隔
04918  *
04919  * @param sample_time 采样时间间隔 单位: m/s^2
04920  * @return 成功返回 0; 失败返回错误码
04921  * AUBO_BUSY
04922  * AUBO_BAD_STATE
04923  * -AUBO_INVL_ARGUMENT
04924  * -AUBO_BAD_STATE
04925  *
04926  * @throws arcs::common_interface::AuboException
04927  * \endchinese
04928  * \english
04929  * Set the sampling interval for points on the future path
04930  *
04931  * @param sample_time Sampling interval, unit: m/s^2
04932  * @return Returns 0 on success; otherwise returns an error code
04933  * AUBO_BUSY
04934  * AUBO_BAD_STATE
04935  * -AUBO_INVL_ARGUMENT
04936  * -AUBO_BAD_STATE
04937  *
04938  * @throws arcs::common_interface::AuboException
04939  * \endenglish
04940  */
04941 int setFuturePointSamplePeriod(double sample_time);
04942
04943 /**
04944  * @ingroup MotionControl
04945  * \chinese
04946  * 获取未来路径上的轨迹点
04947  *
04948  * @return 路点 (100ms * 10)
04949  *
04950  * @throws arcs::common_interface::AuboException
04951  *
04952  * @par JSON-RPC 请求示例
04953  * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getFuturePathPointsJoint", "params": [], "id": 1}
04954  *
04955  * @par JSON-RPC 响应示例
04956  * {"id": 1, "jsonrpc": "2.0", "result": []}
04957  *
04958  * \endchinese
04959  * \english
04960  * Get trajectory points on the future path
04961  *
04962  * @return Waypoints (100ms * 10)
04963  *
04964  * @throws arcs::common_interface::AuboException
04965  *
04966  * @par JSON-RPC request example
04967  * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getFuturePathPointsJoint", "params": [], "id": 1}
04968  *
04969  * @par JSON-RPC response example
04970  * {"id": 1, "jsonrpc": "2.0", "result": []}
04971  *
04972  * \endenglish
04973  */
04974 std::vector<std::vector<double>> getFuturePathPointsJoint();
04975
04976 /**
04977  * @ingroup MotionControl
04978  * \chinese
04979  * 设置传送带编码器参数
04980  *
04981  * @param encoder_id 预留
04982  * @param tick_per_meter
04983  * 线性传送带 ==> 一米的脉冲值
04984  * 环形传送带 ==> 一圈的脉冲值
04985  * @return
04986  *
04987  * @throws arcs::common_interface::AuboException
04988  *
04989  * @par Python 函数原型
04990  * setConveyorTrackEncoder(self: pyaubo_sdk.MotionControl) -> int
04991  *
04992  * @par Lua 函数原型
04993  * setConveyorTrackEncoder(encoder_id: bumber, tick_per_meter: number) -> int
04994  *
04995  * @par Lua 示例
04996  * num = setConveyorTrackEncoder(1, 40000)
04997  *
04998  * @par JSON-RPC 请求示例
04999  * {"jsonrpc": "2.0", "method": "rob1.MotionControl.setConveyorTrackEncoder", "params": [1, 40000], "id": 1}
05000  *

```



```

05001     * @par JSON-RPC 响应示例
05002     * {"id":1,"jsonrpc":"2.0","result":0}
05003     *
05004     * \endchinese
05005     * \english
05006     * Set conveyor encoder parameters
05007     *
05008     * @param encoder_id Reserved
05009     * @param tick_per_meter
05010     * Linear conveyor: pulses per meter
05011     * Circular conveyor: pulses per revolution
05012     * @return
05013     *
05014     * @throws arcs::common_interface::AuboException
05015     *
05016     * @par Python function prototype
05017     * setConveyorTrackEncoder(self: pyaubo_sdk.MotionControl) -> int
05018     *
05019     * @par Lua function prototype
05020     * setConveyorTrackEncoder(encoder_id: bumber,tick_per_meter: number) -> int
05021     *
05022     * @par Lua example
05023     * num = setConveyorTrackEncoder(1,40000)
05024     *
05025     * @par JSON-RPC request example
05026     * {"jsonrpc":"2.0","method":"rob1.MotionControl.setConveyorTrackEncoder","params":[1,40000],"id":1}
05027     *
05028     * @par JSON-RPC response example
05029     * {"id":1,"jsonrpc":"2.0","result":0}
05030     *
05031     * \endenglish
05032     */
05033 int setConveyorTrackEncoder(int encoder_id, int tick_per_meter);
05034 /**
05035     * @ingroup MotionControl
05036     * \chinese
05037     * 圆形传送带跟随
05038     *
05039     * @note 暂未实现
05040     *
05041     * @param encoder_id
05042     * 0-集成传感器
05043     * @param rotate_tool
05044     * @return
05045     *
05046     * @throws arcs::common_interface::AuboException
05047     * \endchinese
05048     * \english
05049     * Circular conveyor tracking
05050     *
05051     * @note Not implemented yet
05052     *
05053     * @param encoder_id
05054     * 0 - integrated sensor
05055     * @param rotate_tool
05056     * @return
05057     *
05058     * @throws arcs::common_interface::AuboException
05059     * \endenglish
05060     */
05061 int conveyorTrackCircle(int encoder_id, const std::vector<double> &center,
05062                        bool rotate_tool);
05063 /**
05064     * @ingroup MotionControl
05065     * \chinese
05066     * 线性传送带跟随
05067     *
05068     *
05069     * @param encoder_id 预留
05070     * @param direction 传送带相对机器人基坐标系的移动方向
05071     * @return
05072     *
05073     * @throws arcs::common_interface::AuboException
05074     *
05075     * @par Python 函数原型
05076     * conveyorTrackLine(self: pyaubo_sdk.MotionControl) -> int
05077     *
05078     * @par Lua 函数原型
05079     * conveyorTrackLine -> int
05080     *
05081     * @par JSON-RPC 请求示例
05082     * {"jsonrpc":"2.0","method":"rob1.MotionControl.conveyorTrackLine","params":[1,
05083     * [1.0,0.0,0.0,0.0,0.0,0.0]],"id":1}
05084     *
05085     * @par JSON-RPC 响应示例
05086     * {"id":1,"jsonrpc":"2.0","result":0}
05087     *

```

```

05087     * \endchinese
05088     * \english
05089     * Linear conveyor tracking
05090     *
05091     * @param encoder_id Reserved
05092     * @param direction The movement direction of the conveyor relative to the
05093     * robot base coordinate system
05094     * @return
05095     *
05096     * @throws arcs::common_interface::AuboException
05097     *
05098     * @par Python function prototype
05099     * conveyorTrackLine(self: pyaubo_sdk.MotionControl) -> int
05100     *
05101     * @par Lua function prototype
05102     * conveyorTrackLine -> int
05103     *
05104     * @par JSON-RPC request example
05105     * {"jsonrpc": "2.0", "method": "robl.MotionControl.conveyorTrackLine", "params": [1,
05106     * [1.0, 0.0, 0.0, 0.0, 0.0, 0.0], "id": 1]}
05107     *
05108     * @par JSON-RPC response example
05109     * {"id": 1, "jsonrpc": "2.0", "result": 0}
05110     * \endenglish
05111     */
05112 int conveyorTrackLine(int encoder_id, const std::vector<double> &direction);
05113 /**
05114     * @ingroup MotionControl
05115     * \chinese
05116     * 终止传送带跟随
05117     *
05118     * @param encoder_id 预留
05119     * @param a 预留
05120     * @return
05121     *
05122     * @throws arcs::common_interface::AuboException
05123     *
05124     * @par Python 函数原型
05125     * conveyorTrackStop(self: pyaubo_sdk.MotionControl) -> int
05126     *
05127     * @par Lua 函数原型
05128     * conveyorTrackStop -> int
05129     *
05130     * @par JSON-RPC 请求示例
05131     * {"jsonrpc": "2.0", "method": "robl.MotionControl.conveyorTrackStop", "params": [0, 1.0], "id": 1}
05132     *
05133     * @par JSON-RPC 响应示例
05134     * {"id": 1, "jsonrpc": "2.0", "result": 0}
05135     * \endchinese
05136     * \english
05137     * Stop conveyor tracking
05138     *
05139     * @param encoder_id Reserved
05140     * @param a Reserved
05141     * @return
05142     *
05143     * @throws arcs::common_interface::AuboException
05144     *
05145     * @par Python function prototype
05146     * conveyorTrackStop(self: pyaubo_sdk.MotionControl) -> int
05147     *
05148     * @par Lua function prototype
05149     * conveyorTrackStop -> int
05150     *
05151     * @par JSON-RPC request example
05152     * {"jsonrpc": "2.0", "method": "robl.MotionControl.conveyorTrackStop", "params": [0, 1.0], "id": 1}
05153     *
05154     * @par JSON-RPC response example
05155     * {"id": 1, "jsonrpc": "2.0", "result": 0}
05156     * \endenglish
05157     */
05158
05159 int conveyorTrackStop(int encoder_id, double a);
05160 /**
05161     * @ingroup MotionControl
05162     * \chinese
05163     * 切换传送带追踪物品
05164     * 如果当前物品正处于跟踪状态，则将该物品出队，不再跟踪，返回 true
05165     * 如果没有物品正处于跟踪状态，返回 false
05166     *
05167     * @return
05168     *
05169     * @throws arcs::common_interface::AuboException
05170     *
05171     * @param encoder_id 预留

```

```

05173 *
05174 * @par Python 函数原型
05175 * conveyorTrackSwitch(self: pyaubo_sdk.MotionControl) -> bool
05176 *
05177 * @par Lua 函数原型
05178 * conveyorTrackSwitch() -> boolean
05179 *
05180 * @par JSON-RPC 请求示例
05181 * {"jsonrpc": "2.0", "method": "rob1.MotionControl.conveyorTrackSwitch", "params": [0], "id": 1}
05182 *
05183 * @par JSON-RPC 响应示例
05184 * {"id": 1, "jsonrpc": "2.0", "result": true}
05185 *
05186 * \endchinese
05187 * \english
05188 * Switch conveyor tracking item.
05189 * If the current item is being tracked, it will be dequeued and no longer
05190 * tracked, returning true. If no item is currently being tracked, returns
05191 * false.
05192 *
05193 * @return
05194 *
05195 * @throws arcs::common_interface::AuboException
05196 *
05197 * @param encoder_id Reserved
05198 *
05199 * @par Python function prototype
05200 * conveyorTrackSwitch(self: pyaubo_sdk.MotionControl) -> bool
05201 *
05202 * @par Lua function prototype
05203 * conveyorTrackSwitch() -> boolean
05204 *
05205 * @par JSON-RPC request example
05206 * {"jsonrpc": "2.0", "method": "rob1.MotionControl.conveyorTrackSwitch", "params": [0], "id": 1}
05207 *
05208 * @par JSON-RPC response example
05209 * {"id": 1, "jsonrpc": "2.0", "result": true}
05210 *
05211 * \endenglish
05212 */
05213 bool conveyorTrackSwitch(int encoder_id);
05214
05215 /**
05216 * @ingroup MotionControl
05217 * \chinese
05218 * 传送带上是否有物品可以跟踪
05219 * @param encoder_id 预留
05220 * @return
05221 * 如果队列第一个物品为跟踪状态, 返回 true
05222 * 如果队列第一个物品不为跟踪状态, 则对队列中其余物品进行是否在启动窗口内的判断
05223 * 超出启动窗口则出队, 直到有物品处于启动窗口内, 使其变为跟踪状态返回 true,
05224 * 队列中没有物品在启动窗口内返回 false
05225 *
05226 * @throws arcs::common_interface::AuboException
05227 *
05228 * @par Python 函数原型
05229 * hasItemOnConveyorToTrack(self: pyaubo_sdk.MotionControl) -> bool
05230 *
05231 * @par Lua 函数原型
05232 * hasItemOnConveyorToTrack() -> boolean
05233 *
05234 * @par JSON-RPC 请求示例
05235 * {"jsonrpc": "2.0", "method": "rob1.MotionControl.hasItemOnConveyorToTrack", "params": [0], "id": 1}
05236 *
05237 * @par JSON-RPC 响应示例
05238 * {"id": 1, "jsonrpc": "2.0", "result": {true}}
05239 * \endchinese
05240 * \english
05241 * Whether there is an item on the conveyor that can be tracked
05242 * @param encoder_id Reserved
05243 * @return
05244 * If the first item in the queue is in tracking state, returns true.
05245 * If the first item is not in tracking state, checks the rest of the queue
05246 * for items within the start window. Items outside the start window are
05247 * dequeued until an item is found within the window, which is then set to
05248 * tracking state and returns true. If no item in the queue is within the
05249 * start window, returns false.
05250 *
05251 * @throws arcs::common_interface::AuboException
05252 *
05253 * @par Python function prototype
05254 * hasItemOnConveyorToTrack(self: pyaubo_sdk.MotionControl) -> bool
05255 *
05256 * @par Lua function prototype
05257 * hasItemOnConveyorToTrack() -> boolean
05258 *
05259 * @par JSON-RPC request example

```

```

05260     * {"jsonrpc": "2.0", "method": "robl.MotionControl.hasItemOnConveyorToTrack", "params": [0], "id": 1}
05261     *
05262     * @par JSON-RPC response example
05263     * {"id": 1, "jsonrpc": "2.0", "result": {true}}
05264     * \endenglish
05265     */
05266 bool hasItemOnConveyorToTrack(int encoder_id);
05267
05268 /**
05269  * @ingroup MotionControl
05270  * \chinese
05271  * 增加传送带队列
05272  *
05273  * @param encoder_id 预留
05274  * @param item_id 物品 ID
05275  * @param offset 当前物品点位相对于模板物品点位的偏移值
05276  * @return
05277  *
05278  * @throws arcs::common_interface::AuboException
05279  *
05280  * @par Python 函数原型
05281  * conveyorTrackCreatItem(self: pyaubo_sdk.MotionControl, arg0: int,
05282  * arg1: int, arg2: List[float]) -> int
05283  *
05284  * @par Lua 函数原型
05285  * conveyorTrackCreatItem(encoder_id: number, item_id: number, offset:
05286  * table) -> number
05287  *
05288  * @par JSON-RPC 请求示例
05289  * {"jsonrpc": "2.0", "method": "robl.MotionControl.conveyorTrackCreatItem", "params": [0, 2,
05290  * {0.0, 0.0, 0.0, 0.0, 0.0, 0.0}], "id": 1}
05291  *
05292  * @par JSON-RPC 响应示例
05293  * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
05294  * \endchinese
05295  * \english
05296  * Add an item to the conveyor queue
05297  *
05298  * @param encoder_id Reserved
05299  * @param item_id Item ID
05300  * @param offset Offset of the current item position relative to the
05301  * template item position
05302  * @return
05303  *
05304  * @throws arcs::common_interface::AuboException
05305  *
05306  * @par Python function prototype
05307  * conveyorTrackCreatItem(self: pyaubo_sdk.MotionControl, arg0: int,
05308  * arg1: int, arg2: List[float]) -> int
05309  *
05310  * @par Lua function prototype
05311  * conveyorTrackCreatItem(encoder_id: number, item_id: number, offset:
05312  * table) -> number
05313  *
05314  * @par JSON-RPC request example
05315  * {"jsonrpc": "2.0", "method": "robl.MotionControl.conveyorTrackCreatItem", "params": [0, 2,
05316  * {0.0, 0.0, 0.0, 0.0, 0.0, 0.0}], "id": 1}
05317  *
05318  * @par JSON-RPC response example
05319  * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
05320  * \endenglish
05321  */
05322 int conveyorTrackCreatItem(int encoder_id, int item_id,
05323                             const std::vector<double> &offset);
05324
05325 /**
05326  * @ingroup MotionControl
05327  * \chinese
05328  * 设置传送带跟踪的补偿值
05329  *
05330  * @param encoder_id 预留
05331  * @param comp 传送带补偿值
05332  * @return
05333  *
05334  * @throws arcs::common_interface::AuboException
05335  *
05336  * @par Python 函数原型
05337  * setConveyorTrackCompensate(self: pyaubo_sdk.MotionControl, arg0: int,
05338  * arg1: float) -> int
05339  *
05340  * @par Lua 函数原型
05341  * setConveyorTrackCompensate(comp: number) -> number
05342  *
05343  * @par JSON-RPC 请求示例
05344  * {"jsonrpc": "2.0", "method": "robl.MotionControl.setConveyorTrackCompensate", "params": [0,
05345  * 0.1], "id": 1}
05346  *
05347  * @par JSON-RPC 响应示例

```

```

05347     * {"id":1,"jsonrpc":"2.0","result":0}
05348     * \endchinese
05349     * \english
05350     * Set the compensation value for conveyor tracking
05351     *
05352     * @param encoder_id Reserved
05353     * @param comp Conveyor compensation value
05354     * @return
05355     *
05356     * @throws arcs::common_interface::AuboException
05357     *
05358     * @par Python function prototype
05359     * setConveyorTrackCompensate(self: pyaubo_sdk.MotionControl, arg0: int,
05360     * arg1: float) -> int
05361     *
05362     * @par Lua function prototype
05363     * setConveyorTrackCompensate(comp: number) -> number
05364     *
05365     * @par JSON-RPC request example
05366     * {"jsonrpc":"2.0","method":"rob1.MotionControl.setConveyorTrackCompensate","params":[0,
05367     * 0.1],"id":1}
05368     *
05369     * @par JSON-RPC response example
05370     * {"id":1,"jsonrpc":"2.0","result":0}
05371     * \endenglish
05372     */
05373 int setConveyorTrackCompensate(int encoder_id, double comp);
05374
05375 /**
05376     * @ingroup MotionControl
05377     * \chinese
05378     * 判断传送带与机械臂之间是否达到相对静止
05379     *
05380     * @param encoder_id 预留
05381     * @return
05382     *
05383     * @throws arcs::common_interface::AuboException
05384     *
05385     * @par Python 函数原型
05386     * isConveyorTrackSync(self: pyaubo_sdk.MotionControl, arg0: int) -> bool
05387     *
05388     * @par Lua 函数原型
05389     * isConveyorTrackSync(encoder_id: number) -> bool
05390     *
05391     * @par JSON-RPC 请求示例
05392     * {"jsonrpc":"2.0","method":"rob1.MotionControl.isConveyorTrackSync","params":[0],"id":1}
05393     *
05394     * @par JSON-RPC 响应示例
05395     * {"id":1,"jsonrpc":"2.0","result":false}
05396     *
05397     * \endchinese
05398     * \english
05399     * Determine whether the conveyor and the robot arm have reached relative
05400     * rest
05401     *
05402     * @param encoder_id Reserved
05403     * @return
05404     *
05405     * @throws arcs::common_interface::AuboException
05406     *
05407     * @par Python function prototype
05408     * isConveyorTrackSync(self: pyaubo_sdk.MotionControl, arg0: int) -> bool
05409     *
05410     * @par Lua function prototype
05411     * isConveyorTrackSync(encoder_id: number) -> bool
05412     *
05413     * @par JSON-RPC request example
05414     * {"jsonrpc":"2.0","method":"rob1.MotionControl.isConveyorTrackSync","params":[0],"id":1}
05415     *
05416     * @par JSON-RPC response example
05417     * {"id":1,"jsonrpc":"2.0","result":false}
05418     *
05419     * \endenglish
05420     */
05421 bool isConveyorTrackSync(int encoder_id);
05422
05423 /**
05424     * @ingroup MotionControl
05425     * \chinese
05426     * 设置传送带跟踪的最大距离限制
05427     *
05428     * @param encoder_id 预留
05429     * @param limit
05430     * 传送带跟踪的最大距离限制, 单位: 米
05431     * @return
05432     *
05433     * @throws arcs::common_interface::AuboException

```

```

05434     *
05435     * @par Python 函数原型
05436     * setConveyorTrackLimit(self: pyaubo_sdk.MotionControl, arg0: int, arg1:
05437     * double) -> int
05438     *
05439     * @par Lua 函数原型
05440     * setConveyorTrackLimit(encoder_id: number, limit: number) -> int
05441     *
05442     * @par JSON-RPC 请求示例
05443     * {"jsonrpc": "2.0", "method": "robl.MotionControl.setConveyorTrackLimit", "params": [0,
05444     * 1.5], "id": 1}
05445     *
05446     * @par JSON-RPC 响应示例
05447     * {"id": 1, "jsonrpc": "2.0", "result": 0}
05448     * \endchinese
05449     * \english
05450     * Set the maximum distance limit for conveyor tracking
05451     *
05452     * @param encoder_id Reserved
05453     * @param limit
05454     * Maximum distance limit for conveyor tracking, unit: meter
05455     * @return
05456     *
05457     * @throws arcs::common_interface::AuboException
05458     *
05459     * @par Python function prototype
05460     * setConveyorTrackLimit(self: pyaubo_sdk.MotionControl, arg0: int, arg1:
05461     * double) -> int
05462     *
05463     * @par Lua function prototype
05464     * setConveyorTrackLimit(encoder_id: number, limit: number) -> int
05465     *
05466     * @par JSON-RPC request example
05467     * {"jsonrpc": "2.0", "method": "robl.MotionControl.setConveyorTrackLimit", "params": [0,
05468     * 1.5], "id": 1}
05469     *
05470     * @par JSON-RPC response example
05471     * {"id": 1, "jsonrpc": "2.0", "result": 0}
05472     * \endenglish
05473     */
05474 int setConveyorTrackLimit(int encoder_id, double limit);
05475
05476 /**
05477     * @ingroup MotionControl
05478     * \chinese
05479     * 设置传送带跟踪的启动窗口
05480     *
05481     * @param encoder_id 预留
05482     * @param min_window
05483     * 启动窗口的起始位置, 单位: 米
05484     * @param max_window
05485     * 启动窗口的结束位置, 单位: 米
05486     * @return
05487     *
05488     * @throws arcs::common_interface::AuboException
05489     *
05490     * @par Python 函数原型
05491     * setConveyorTrackStartWindow(self: pyaubo_sdk.MotionControl, arg0: int,
05492     * arg1: double, arg2: double) -> int
05493     *
05494     * @par Lua 函数原型
05495     * setConveyorTrackStartWindow(encoder_id: number, min_window: number,
05496     * max_window: number) -> int
05497     *
05498     * @par JSON-RPC 请求示例
05499     * {"jsonrpc": "2.0", "method": "robl.MotionControl.setConveyorTrackStartWindow", "params": [0,
05500     * 0.2, 1.0], "id": 1}
05501     *
05502     * @par JSON-RPC 响应示例
05503     * {"id": 1, "jsonrpc": "2.0", "result": 0}
05504     *
05505     * \endchinese
05506     * \english
05507     * Set the start window for conveyor tracking
05508     *
05509     * @param encoder_id Reserved
05510     * @param min_window
05511     * Start position of the window, unit: meter
05512     * @param max_window
05513     * End position of the window, unit: meter
05514     * @return
05515     *
05516     * @throws arcs::common_interface::AuboException
05517     *
05518     * @par Python function prototype
05519     * setConveyorTrackStartWindow(self: pyaubo_sdk.MotionControl, arg0: int,
05520     * arg1: double, arg2: double) -> int

```

```

05521      *
05522      * @par Lua function prototype
05523      * setConveyorTrackStartWindow(encoder_id: number, min_window: number,
05524      * max_window: number) -> int
05525      *
05526      * @par JSON-RPC request example
05527      * {"jsonrpc": "2.0", "method": "robl.MotionControl.setConveyorTrackStartWindow", "params": [0,
05528      * 0.2, 1.0], "id": 1}
05529      *
05530      * @par JSON-RPC response example
05531      * {"id": 1, "jsonrpc": "2.0", "result": 0}
05532      *
05533      * \endenglish
05534      */
05535      int setConveyorTrackStartWindow(int encoder_id, double window_min,
05536                                     double window_max);
05537
05538      /**
05539      * @ingroup MotionControl
05540      * \chinese
05541      * 设置传送带示教位置到同步开关之间的距离
05542      *
05543      * @param encoder_id 预留
05544      * @param offset
05545      * 传送带示教位置到同步开关之间的距离，单位：米
05546      * @return
05547      *
05548      * @throws arcs::common_interface::AuboException
05549      *
05550      * @par Python 函数原型
05551      * setConveyorTrackSensorOffset(self: pyaubo_sdk.MotionControl, arg0: int,
05552      * arg1: double) -> int
05553      *
05554      * @par Lua 函数原型
05555      * setConveyorTrackSensorOffset(encoder_id: number, offset: number) -> int
05556      *
05557      * @par JSON-RPC 请求示例
05558      * {"jsonrpc": "2.0", "method": "robl.MotionControl.setConveyorTrackSensorOffset", "params": [0,
05559      * 0.2], "id": 1}
05560      *
05561      * @par JSON-RPC 响应示例
05562      * {"id": 1, "jsonrpc": "2.0", "result": 0}
05563      *
05564      * \endchinese
05565      * \english
05566      * Set the distance from the conveyor teaching position to the sync switch
05567      *
05568      * @param encoder_id Reserved
05569      * @param offset
05570      * Distance from the conveyor teaching position to the sync switch, unit:
05571      * meter
05572      * @return
05573      *
05574      * @throws arcs::common_interface::AuboException
05575      *
05576      * @par Python function prototype
05577      * setConveyorTrackSensorOffset(self: pyaubo_sdk.MotionControl, arg0: int,
05578      * arg1: double) -> int
05579      *
05580      * @par Lua function prototype
05581      * setConveyorTrackSensorOffset(encoder_id: number, offset: number) -> int
05582      *
05583      * @par JSON-RPC request example
05584      * {"jsonrpc": "2.0", "method": "robl.MotionControl.setConveyorTrackSensorOffset", "params": [0,
05585      * 0.2], "id": 1}
05586      *
05587      * @par JSON-RPC response example
05588      * {"id": 1, "jsonrpc": "2.0", "result": 0}
05589      *
05590      * \endenglish
05591      */
05592      int setConveyorTrackSensorOffset(int encoder_id, double offset);
05593
05594      /**
05595      * @ingroup MotionControl
05596      * \chinese
05597      * 设置传送带同步分离，用于过滤掉同步开关中不需要的信号
05598      *
05599      * @param encoder_id 预留
05600      * @param distance
05601      * 从出现一个同步信号后到把一个新的同步信号接受为一个有效对象前走的最短距离，单位：米
05602      * @param time
05603      * 从出现一个同步信号后到把一个新的同步信号接受为一个有效对象前走的最短时间，单位：秒
05604      *
05605      * distance 和 time 设置数值大于 0 即为生效
05606      * 当 distance 与 time 同时设置时，优先生效 distance
05607      * @return

```



```

05608     *
05609     * @throws arcs::common_interface::AuboException
05610     *
05611     * @par Python 函数原型
05612     * setConveyorTrackSyncSeparation(self: pyaubo_sdk.MotionControl, arg0: int,
05613     * arg1: double, arg2: double) -> int
05614     *
05615     * @par Lua 函数原型
05616     * setConveyorTrackSyncSeparation(encoder_id: number, distance: number,
05617     * time: number) -> int
05618     *
05619     * @par JSON-RPC 请求示例
05620     * {"jsonrpc": "2.0", "method": "robl.MotionControl.setConveyorTrackSyncSeparation", "params": [0,
05621     * 0.05, 0.2], "id": 1}
05622     *
05623     * @par JSON-RPC 响应示例
05624     * {"id": 1, "jsonrpc": "2.0", "result": 0}
05625     * \endchinese
05626     * \english
05627     * Set conveyor sync separation, used to filter out unwanted signals from
05628     * the sync switch.
05629     *
05630     * @param encoder_id Reserved
05631     * @param distance
05632     * The minimum distance to travel after a sync signal appears before
05633     * accepting a new sync signal as a valid object, unit: meter
05634     * @param time
05635     * The minimum time to elapse after a sync signal appears before accepting a
05636     * new sync signal as a valid object, unit: second
05637     *
05638     * If distance and time are both set greater than 0, the setting takes
05639     * effect. If both distance and time are set, distance takes precedence.
05640     * @return
05641     *
05642     * @throws arcs::common_interface::AuboException
05643     *
05644     * @par Python function prototype
05645     * setConveyorTrackSyncSeparation(self: pyaubo_sdk.MotionControl, arg0: int,
05646     * arg1: double, arg2: double) -> int
05647     *
05648     * @par Lua function prototype
05649     * setConveyorTrackSyncSeparation(encoder_id: number, distance: number,
05650     * time: number) -> int
05651     *
05652     * @par JSON-RPC request example
05653     * {"jsonrpc": "2.0", "method": "robl.MotionControl.setConveyorTrackSyncSeparation", "params": [0,
05654     * 0.05, 0.2], "id": 1}
05655     *
05656     * @par JSON-RPC response example
05657     * {"id": 1, "jsonrpc": "2.0", "result": 0}
05658     * \endenglish
05659     */
05660     int setConveyorTrackSyncSeparation(int encoder_id, double distance,
05661                                       double time);
05662
05663     /**
05664     * @ingroup MotionControl
05665     * \chinese
05666     * 设置传送带运行速度阈值，用于过滤掉编码器突变值导致的位置突变
05667     *
05668     * @param encoder_id 预留
05669     * @param threshold
05670     * 传送带运行时期望容许的最大速度（默认为 0.2），单位：米/秒
05671     *
05672     *
05673     * @return
05674     *
05675     * @throws arcs::common_interface::AuboException
05676     *
05677     * @par Python 函数原型
05678     * setConveyorTrackVelThreshold(self: pyaubo_sdk.MotionControl, arg0: int,
05679     * arg1: double) -> int
05680     *
05681     * @par Lua 函数原型
05682     * setConveyorTrackVelThreshold(encoder_id: number, threshold: number) ->
05683     * int
05684     *
05685     * @par JSON-RPC 请求示例
05686     * {"jsonrpc": "2.0", "method": "robl.MotionControl.setConveyorTrackVelThreshold", "params": [0,
05687     * 0.2], "id": 1}
05688     *
05689     * @par JSON-RPC 响应示例
05690     * {"id": 1, "jsonrpc": "2.0", "result": 0}
05691     * \endchinese
05692     * \english
05693     * Set the conveyor velocity threshold used to suppress position jumps
05694     * caused by abnormal encoder spikes.

```



```

05695      *
05696      * @param encoder_id Reserved
05697      * @param threshold
05698      * The maximum allowable conveyor speed during operation, unit:
05699      * meter/second. Default: 0.2 m/s.
05700      *
05701      * @return
05702      * 0 on success; otherwise a non-zero error code.
05703      *
05704      * @throws arcs::common_interface::AuboException
05705      *
05706      * @par Python function prototype
05707      * setConveyorTrackVelThreshold(self: pyaubo_sdk.MotionControl, arg0: int,
05708      * arg1: double) -> int
05709      *
05710      * @par Lua function prototype
05711      * setConveyorTrackVelThreshold(encoder_id: number, threshold: number) ->
05712      * int
05713      *
05714      * @par JSON-RPC request example
05715      * {"jsonrpc": "2.0", "method": "rob1.MotionControl.setConveyorTrackVelThreshold", "params": [0,
05716      * 0.2], "id": 1}
05717      *
05718      * @par JSON-RPC response example
05719      * {"id": 1, "jsonrpc": "2.0", "result": 0}
05720      * \endenglish
05721      */
05722 int setConveyorTrackVelThreshold(int encoder_id, double threshold);
05723
05724 /**
05725  * \chinese
05726  * 传送带上工件的移动距离是否超过最大限值
05727  *
05728  * @param encoder_id 预留
05729  * @return
05730  * true : 移动距离超过了最大限值
05731  * false : 移动距离没有超过了最大限值
05732  *
05733  * @throws arcs::common_interface::AuboException
05734  *
05735  * @par Python 函数原型
05736  * isConveyorTrackExceed(self: pyaubo_sdk.MotionControl, arg0: int) -> bool
05737  *
05738  * @par Lua 函数原型
05739  * isConveyorTrackExceed(encoder_id: number) -> bool
05740  *
05741  * @par JSON-RPC 请求示例
05742  * {"jsonrpc": "2.0", "method": "rob1.MotionControl.isConveyorTrackExceed", "params": [0], "id": 1}
05743  *
05744  * @par JSON-RPC 响应示例
05745  * {"id": 1, "jsonrpc": "2.0", "result": false}
05746  * \endchinese
05747  * \english
05748  * Whether the workpiece on the conveyor has moved beyond the maximum limit
05749  *
05750  * @param encoder_id Reserved
05751  * @return
05752  * true : The movement distance exceeds the maximum limit
05753  * false : The movement distance does not exceed the maximum limit
05754  *
05755  * @throws arcs::common_interface::AuboException
05756  *
05757  * @par Python function prototype
05758  * isConveyorTrackExceed(self: pyaubo_sdk.MotionControl, arg0: int) -> bool
05759  *
05760  * @par Lua function prototype
05761  * isConveyorTrackExceed(encoder_id: number) -> bool
05762  *
05763  * @par JSON-RPC request example
05764  * {"jsonrpc": "2.0", "method": "rob1.MotionControl.isConveyorTrackExceed", "params": [0], "id": 1}
05765  *
05766  * @par JSON-RPC response example
05767  * {"id": 1, "jsonrpc": "2.0", "result": false}
05768  * \endenglish
05769  */
05770 bool isConveyorTrackExceed(int encoder_id);
05771
05772 /**
05773  * @ingroup MotionControl
05774  * \chinese
05775  * 清空传动带队列中的所有对象
05776  *
05777  * @param encoder_id 预留
05778  * @return
05779  *
05780  * @throws arcs::common_interface::AuboException
05781  *

```

```

05782 * @par Python 函数原型
05783 * conveyorTrackClearItems(self: pyaubo_sdk.MotionControl, arg0: int) -> int
05784 *
05785 * @par Lua 函数原型
05786 * conveyorTrackClearItems(encoder_id: number) -> int
05787 *
05788 * @par JSON-RPC 请求示例
05789 * {"jsonrpc": "2.0", "method": "robl.MotionControl.conveyorTrackClearItems", "params": [0], "id": 1}
05790 *
05791 * @par JSON-RPC 响应示例
05792 * {"id": 1, "jsonrpc": "2.0", "result": 0}
05793 *
05794 * \endchinese
05795 * \english
05796 * Clear all items in the conveyor queue
05797 *
05798 * @param encoder_id Reserved
05799 * @return
05800 *
05801 * @throws arcs::common_interface::AuboException
05802 *
05803 * @par Python function prototype
05804 * conveyorTrackClearItems(self: pyaubo_sdk.MotionControl, arg0: int) -> int
05805 *
05806 * @par Lua function prototype
05807 * conveyorTrackClearItems(encoder_id: number) -> int
05808 *
05809 * @par JSON-RPC request example
05810 * {"jsonrpc": "2.0", "method": "robl.MotionControl.conveyorTrackClearItems", "params": [0], "id": 1}
05811 *
05812 * @par JSON-RPC response example
05813 * {"id": 1, "jsonrpc": "2.0", "result": 0}
05814 *
05815 * \endenglish
05816 */
05817 int conveyorTrackClearItems(int encoder_id);
05818
05819 /**
05820 * @ingroup MotionControl
05821 * \chinese
05822 * 获取传送带队列的编码器值
05823 *
05824 * @param encoder_id 预留
05825 * @return
05826 *
05827 * @throws arcs::common_interface::AuboException
05828 *
05829 * @par Python 函数原型
05830 * getConveyorTrackQueue(self: pyaubo_sdk.MotionControl, arg0: int) ->
05831 * List[int]
05832 * @par Lua 函数原型
05833 * getConveyorTrackQueue(encoder_id: number) -> table
05834 *
05835 * @par JSON-RPC 请求示例
05836 * {"jsonrpc": "2.0", "method": "robl.MotionControl.getConveyorTrackQueue", "params": [0], "id": 1}
05837 *
05838 * @par JSON-RPC 响应示例
05839 * {"id": 1, "jsonrpc": "2.0", "result": {[-500, -200, 150, -50]}}
05840 * \endchinese
05841 * \english
05842 * Get encoder values of the conveyor queue
05843 *
05844 * @param encoder_id Reserved
05845 * @return
05846 *
05847 * @throws arcs::common_interface::AuboException
05848 *
05849 * @par Python function prototype
05850 * getConveyorTrackQueue(self: pyaubo_sdk.MotionControl, arg0: int) ->
05851 * List[int]
05852 * @par Lua function prototype
05853 * getConveyorTrackQueue(encoder_id: number) -> table
05854 *
05855 * @par JSON-RPC request example
05856 * {"jsonrpc": "2.0", "method": "robl.MotionControl.getConveyorTrackQueue", "params": [0], "id": 1}
05857 *
05858 * @par JSON-RPC response example
05859 * {"id": 1, "jsonrpc": "2.0", "result": {[-500, -200, 150, -50]}}
05860 * \endenglish
05861 */
05862 std::vector<int> getConveyorTrackQueue(int encoder_id);
05863 /**
05864 * \chinese
05865 * 获取下一个跟踪的传送带物品的 id
05866 *
05867 * @param encoder_id 预留
05868 * @return

```

```

05869     * 返回物品 id, 没有 next item 返回 -1
05870     *
05871     * @throws arcs::common_interface::AuboException
05872     *
05873     * @par Python 函数原型
05874     * getConveyorTrackNextItem(self: pyaubo_sdk.MotionControl, arg0: int) ->
05875     * int
05876     * @par Lua 函数原型
05877     * getConveyorTrackNextItem(encoder_id: number) -> int
05878     *
05879     * @par JSON-RPC 请求示例
05880     * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getConveyorTrackNextItem", "params": [0], "id": 1}
05881     *
05882     * @par JSON-RPC 响应示例
05883     * {"id": 1, "jsonrpc": "2.0", "result": {10}}
05884     * \endchinese
05885     * \english
05886     * Get the ID of the next item on the conveyor to be tracked
05887     *
05888     * @param encoder_id Reserved
05889     * @return
05890     * Returns the item ID; returns -1 if there is no next item
05891     *
05892     * @throws arcs::common_interface::AuboException
05893     *
05894     * @par Python function prototype
05895     * getConveyorTrackNextItem(self: pyaubo_sdk.MotionControl, arg0: int) ->
05896     * int
05897     * @par Lua function prototype
05898     * getConveyorTrackNextItem(encoder_id: number) -> int
05899     *
05900     * @par JSON-RPC Request Example
05901     * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getConveyorTrackNextItem", "params": [0], "id": 1}
05902     *
05903     * @par JSON-RPC Response Example
05904     * {"id": 1, "jsonrpc": "2.0", "result": {10}}
05905     * \endenglish
05906     */
05907
05908 int getConveyorTrackNextItem(int encoder_id);
05909
05910 /**
05911  * @ingroup MotionControl
05912  * \chinese
05913  * 螺旋线运动
05914  *
05915  * @param param 封装的参数
05916  * @param blend_radius
05917  * @param v
05918  * @param a
05919  * @param t
05920  * @return 成功返回 0; 失败返回错误码
05921  * AUBO_BUSY
05922  * AUBO_BAD_STATE
05923  * -AUBO_INVL_ARGUMENT
05924  * -AUBO_BAD_STATE
05925  *
05926  * @throws arcs::common_interface::AuboException
05927  *
05928  * @par JSON-RPC 请求示例
05929  * {"jsonrpc": "2.0", "method": "rob1.MotionControl.moveSpiral",
05930  * "params": [{"spiral": 0.005, "helix": 0.005, "angle": 18.84, "plane": 0, "frame": [0, 0, 0, 0, 0]}, 0, 0.25, 1.2, 0], "id": 1}
05931  *
05932  * @par JSON-RPC 响应示例
05933  * {"id": 1, "jsonrpc": "2.0", "result": 0}
05934  * \endchinese
05935  * \english
05936  * Spiral motion
05937  *
05938  * @param param Encapsulated parameters
05939  * @param blend_radius
05940  * @param v
05941  * @param a
05942  * @param t
05943  * @return Returns 0 on success; otherwise returns an error code
05944  * AUBO_BUSY
05945  * AUBO_BAD_STATE
05946  * -AUBO_INVL_ARGUMENT
05947  * -AUBO_BAD_STATE
05948  *
05949  * @throws arcs::common_interface::AuboException
05950  *
05951  * @par JSON-RPC request example
05952  * {"jsonrpc": "2.0", "method": "rob1.MotionControl.moveSpiral",
05953  * "params": [{"spiral": 0.005, "helix": 0.005, "angle": 18.84, "plane": 0, "frame": [0, 0, 0, 0, 0]}, 0, 0.25, 1.2, 0], "id": 1}
05954  *
05955  * @par JSON-RPC response example

```

```

05956     * {"id":1,"jsonrpc":"2.0","result":0}
05957     * \endenglish
05958     */
05959 int moveSpiral(const SpiralParameters &param, double blend_radius, double v,
05960               double a, double t);
05961
05962 /**
05963  * @ingroup MotionControl
05964  * \chinese
05965  * 获取前瞻段数
05966  *
05967  * @return 返回前瞻段数
05968  *
05969  * @throws arcs::common_interface::AuboException
05970  *
05971  * @par Python 函数原型
05972  * getLookAheadSize(self: pyaubo_sdk.MotionControl) -> int
05973  *
05974  * @par Lua 函数原型
05975  * getLookAheadSize() -> number
05976  *
05977  * @par JSON-RPC 请求示例
05978  * {"jsonrpc":"2.0","method":"rob1.MotionControl.getLookAheadSize","params":[],"id":1}
05979  *
05980  * @par JSON-RPC 响应示例
05981  * {"id":1,"jsonrpc":"2.0","result":1}
05982  *
05983  * \endchinese
05984  * \english
05985  * Get look-ahead segment size
05986  *
05987  * @return Returns the look-ahead segment size
05988  *
05989  * @throws arcs::common_interface::AuboException
05990  *
05991  * @par Python function prototype
05992  * getLookAheadSize(self: pyaubo_sdk.MotionControl) -> int
05993  *
05994  * @par Lua function prototype
05995  * getLookAheadSize() -> number
05996  *
05997  * @par JSON-RPC request example
05998  * {"jsonrpc":"2.0","method":"rob1.MotionControl.getLookAheadSize","params":[],"id":1}
05999  *
06000  * @par JSON-RPC response example
06001  * {"id":1,"jsonrpc":"2.0","result":1}
06002  *
06003  * \endenglish
06004  */
06005 int getLookAheadSize();
06006
06007 /**
06008  * @ingroup MotionControl
06009  * \chinese
06010  * 设置前瞻段数
06011  * 1. 对于有较高速度平稳性要求的任务，如数控加工，涂胶，焊接等匀速需求，较长的前瞻段数可以提供更优的速度规划，产生的运
    动会更加平滑；
06012  * 2. 对于快速响应的抓取类任务，更倾向于较短的前瞻段数，以提高反应速度，但可能因为进给的路径不够及时导致速度波动很大。
06013  *
06014  * @return 成功返回 0
06015  *
06016  * @throws arcs::common_interface::AuboException
06017  *
06018  * @par Python 函数原型
06019  * setLookAheadSize(self: pyaubo_sdk.MotionControl, arg0: int) -> int
06020  *
06021  * @par Lua 函数原型
06022  * setLookAheadSize(eqradius: number) -> number
06023  *
06024  * @par JSON-RPC 请求示例
06025  * {"jsonrpc":"2.0","method":"rob1.MotionControl.setLookAheadSize","params":[1],"id":1}
06026  *
06027  * @par JSON-RPC 响应示例
06028  * {"id":1,"jsonrpc":"2.0","result":0}
06029  * \endchinese
06030  * \english
06031  * Set look-ahead segment size
06032  * 1. For tasks requiring high speed smoothness, such as CNC machining,
06033  * gluing, welding, etc., a longer look-ahead segment size can provide
06034  * better speed planning and smoother motion.
06035  * 2. For fast-response tasks such as picking, a shorter look-ahead segment
06036  * size is preferred to improve response speed, but may cause large speed
06037  * fluctuations due to insufficiently timely path feeding.
06038  *
06039  * @return Returns 0 on success
06040  *
06041  * @throws arcs::common_interface::AuboException

```

```

06042     *
06043     * @par Python function prototype
06044     * setLookAheadSize(self: pyaubo_sdk.MotionControl, arg0: int) -> int
06045     *
06046     * @par Lua function prototype
06047     * setLookAheadSize(eqradius: number) -> number
06048     *
06049     * @par JSON-RPC request example
06050     * {"jsonrpc": "2.0", "method": "rob1.MotionControl.setLookAheadSize", "params": [1], "id": 1}
06051     *
06052     * @par JSON-RPC response example
06053     * {"id": 1, "jsonrpc": "2.0", "result": 0}
06054     * \endenglish
06055     */
06056 int setLookAheadSize(int size);
06057
06058 /**
06059  * @ingroup MotionControl
06060  * \chinese
06061  * 更新摆动过程中的频率和振幅
06062  *
06063  * @param params Json 字符串用于定义摆动参数
06064  * {
06065  *   "frequency": <num>,
06066  *   "amplitude": {<num>, <num>}
06067  * }
06068  *
06069  * @return 成功返回 0
06070  *
06071  * @throws arcs::common_interface::AuboException
06072  * \endchinese
06073  * \english
06074  * Update frequency and amplitude during weaving process
06075  *
06076  * @param params Json string used to define weaving parameters
06077  * {
06078  *   "frequency": <num>,
06079  *   "amplitude": {<num>, <num>}
06080  * }
06081  *
06082  * @return Returns 0 on success
06083  *
06084  * @throws arcs::common_interface::AuboException
06085  * \endenglish
06086  */
06087 int weaveUpdateParameters(const std::string &params);
06088
06089 /**
06090  * @ingroup MotionControl
06091  * \chinese
06092  * 设置关节电流环刚度系数
06093  *
06094  * @param stiffness 各个关节刚度系数，百分比。[0 -> 1]，值越大，表现为越硬。
06095  * @return
06096  *
06097  * @throws arcs::common_interface::AuboException
06098  *
06099  * @par Python 函数原型
06100  * enableJointSoftServo(self: pyaubo_sdk.RobotConfig, arg0: List[float]) ->
06101  * int
06102  *
06103  * @par Lua 函数原型
06104  * enableJointSoftServo(stiffness: table) -> int
06105  *
06106  * @par JSON-RPC 请求示例
06107  * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.enableJointSoftServo", "params":
06108  * [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]], "id": 1}
06109  *
06110  * @par JSON-RPC 响应示例
06111  * {"id": 1, "jsonrpc": "2.0", "result": 0}
06112  * \endchinese
06113  * \english
06114  * Set joint current loop stiffness coefficient
06115  *
06116  * @param stiffness Stiffness coefficient for each joint, as a percentage [0
06117  * -> 1]. The larger the value, the stiffer the joint.
06118  * @return
06119  *
06120  * @throws arcs::common_interface::AuboException
06121  *
06122  * @par Python function prototype
06123  * enableJointSoftServo(self: pyaubo_sdk.RobotConfig, arg0: List[float]) ->
06124  * int
06125  *
06126  * @par Lua function prototype
06127  * enableJointSoftServo(stiffness: table) -> int
06128  *

```

```

06128     * @par JSON-RPC request example
06129     * {"jsonrpc":"2.0","method":"rob1.RobotConfig.enableJointSoftServo","params":
[[0.0,0.0,0.0,0.0,0.0,0.0,0.0]],"id":1}
06130     *
06131     * @par JSON-RPC response example
06132     * {"id":1,"jsonrpc":"2.0","result":0}
06133     * \endenglish
06134     */
06135     int enableJointSoftServo(const std::vector<double> &stiffness);
06136
06137     /**
06138     * @ingroup MotionControl
06139     * \chinese
06140     * 关闭关节电流环刚度系数
06141     *
06142     * @return 返回 0 表示成功，其他为错误码
06143     *
06144     * @throws arcs::common_interface::AuboException
06145     *
06146     * @par Python 函数原型
06147     * disableJointSoftServo(self: pyaubo_sdk.MotionControl) -> int
06148     *
06149     * @par Lua 函数原型
06150     * disableJointSoftServo() -> number
06151     *
06152     * @par JSON-RPC 请求示例
06153     * {"jsonrpc":"2.0","method":"rob1.MotionControl.disableJointSoftServo","params":[],"id":1}
06154     *
06155     * @par JSON-RPC 响应示例
06156     * {"id":1,"jsonrpc":"2.0","result":1}
06157     * \endchinese
06158     * \english
06159     * Disable joint current loop stiffness coefficient
06160     *
06161     * @return Returns 0 on success, otherwise error code
06162     *
06163     * @throws arcs::common_interface::AuboException
06164     *
06165     * @par Python function prototype
06166     * disableJointSoftServo(self: pyaubo_sdk.MotionControl) -> int
06167     *
06168     * @par Lua function prototype
06169     * disableJointSoftServo() -> number
06170     *
06171     * @par JSON-RPC request example
06172     * {"jsonrpc":"2.0","method":"rob1.MotionControl.disableJointSoftServo","params":[],"id":1}
06173     *
06174     * @par JSON-RPC response example
06175     * {"id":1,"jsonrpc":"2.0","result":1}
06176     * \endenglish
06177     */
06178     int disableJointSoftServo();
06179
06180     /**
06181     * @ingroup MotionControl
06182     * \chinese
06183     * 判断关节电流环刚度系数是否使能
06184     *
06185     * @return 已使能返回 true，反之则返回 false
06186     *
06187     * @throws arcs::common_interface::AuboException
06188     *
06189     * @par Python 函数原型
06190     * isJointSoftServoEnabled(self: pyaubo_sdk.MotionControl) -> bool
06191     *
06192     * @par Lua 函数原型
06193     * isJointSoftServoEnabled() -> boolean
06194     *
06195     * @par JSON-RPC 请求示例
06196     * {"jsonrpc":"2.0","method":"rob1.MotionControl.isJointSoftServoEnabled","params":[],"id":1}
06197     *
06198     * @par JSON-RPC 响应示例
06199     * {"id":1,"jsonrpc":"2.0","result":1}
06200     * \endchinese
06201     * \english
06202     * Determine whether the joint current loop stiffness coefficient is
06203     * enabled.
06204     *
06205     * @return Returns true if enabled, otherwise returns false.
06206     *
06207     * @throws arcs::common_interface::AuboException
06208     *
06209     * @par Python function prototype
06210     * isJointSoftServoEnabled(self: pyaubo_sdk.MotionControl) -> bool
06211     *
06212     * @par Lua function prototype
06213     * isJointSoftServoEnabled() -> boolean

```

```

06214      *
06215      * @par JSON-RPC request example
06216      * {"jsonrpc": "2.0", "method": "rob1.MotionControl.isJointSoftServoEnabled", "params": [], "id": 1}
06217      *
06218      * @par JSON-RPC response example
06219      * {"id": 1, "jsonrpc": "2.0", "result": 1}
06220      * \endenglish
06221      */
06222      bool isJointSoftServoEnabled();
06223
06224      /**
06225      * @ingroup MotionControl
06226      * \chinese
06227      * 打开振动抑制
06228      *
06229      * @param omega 振动抑制频率，长度一般为笛卡尔维度或关节自由度。为 0 时关闭抑制
06230      * \n
06231      * @param zeta 振动抑制阻尼比，长度一般为笛卡尔维度或关节自由度。为 0 时关闭抑制
06232      * \n
06233      * @param level
06234      * 振动抑制等级（1-3），等级越高抑制效果越明显响应速度相应变慢。等级 0 关闭振动抑制
06235      * \n
06236      * @retval 0 成功
06237      * @retval AUBO_BAD_STATE(1) 当前安全模式处于非
06238      * Normal、ReducedMode、Recovery 状态
06239      * @retval AUBO_QUEUE_FULL(2) 规划队列已满
06240      * @retval AUBO_BUSY(3) 上一条指令正在执行中
06241      * @retval -AUBO_BAD_STATE(-1)
06242      * 可能的原因包括但不限于：线程已分离、线程被终止、task_id
06243      * 未找到，或者当前机器人模式非 Running
06244      * @retval -AUBO_TIMEOUT(-4) 调用接口超时
06245      *
06246      * @throws arcs::common_interface::AuboException
06247      *
06248      * @par Python 函数原型
06249      * enableVibrationSuppress(self: pyaubo_sdk.MotionControl, arg0:
06250      * list[float], arg1: list[float], arg2: int) -> int
06251      *
06252      * @par Lua 函数原型
06253      * enableVibrationSuppress(omega: table, zeta: table, level: number) -> nil
06254      *
06255      * @par JSON-RPC 请求示例
06256      * {"jsonrpc": "2.0", "method": "rob1.MotionControl.enableVibrationSuppress", "params": [[0.5, 0.5, 0.5, 0.5, 0.5, 0.5],
06257      * [0.5, 0.5, 0.5, 0.5, 0.5, 0.5], 2], "id": 1}
06258      *
06259      * @par JSON-RPC 响应示例
06260      * {"id": 1, "jsonrpc": "2.0", "result": 0}
06261      * \endchinese
06262      */
06263      int enableVibrationSuppress(const std::vector<double> &omega,
06264      const std::vector<double> &zeta, int level);
06265
06266      /**
06267      * @ingroup MotionControl
06268      * \chinese
06269      * 关闭振动抑制
06270      *
06271      * @return 已使能返回 true，反之则返回 false
06272      *
06273      * @throws arcs::common_interface::AuboException
06274      *
06275      * @par Python 函数原型
06276      * disbleVibrationSuppress(self: pyaubo_sdk.MotionControl) -> int
06277      *
06278      * @par Lua 函数原型
06279      * disbleVibrationSuppress() -> number
06280      *
06281      * @par JSON-RPC 请求示例
06282      * {"jsonrpc": "2.0", "method": "rob1.MotionControl.disbleVibrationSuppress", "params": [], "id": 1}
06283      *
06284      * @par JSON-RPC 响应示例
06285      * {"id": 1, "jsonrpc": "2.0", "result": 1}
06286      * \endchinese
06287      */
06288      int disbleVibrationSuppress();
06289
06290      /**
06291      * @ingroup MotionControl
06292      * \chinese
06293      * 设置时间最优算法 默认关闭
06294      *
06295      * @param enable
06296      *
06297      * @throws arcs::common_interface::AuboException
06298      *
06299      * @par Lua 函数原型
06300      * setTimeOptimalEnable() -> bool

```



```

06300      *
06301      * @par Lua 示例
06302      * setTimeOptimalEnable(true)
06303      *
06304      * @par JSON-RPC 请求示例
06305      * {"jsonrpc": "2.0", "method": "robl.MotionControl.setTimeOptimalEnable", "params": [true], "id": 1}
06306      *
06307      * @par JSON-RPC 响应示例
06308      * {"id": 1, "jsonrpc": "2.0", "result": 0}
06309      * \endchinese
06310      * \english
06311      *
06312      * @param enable
06313      *
06314      * @throws arcs::common_interface::AuboException
06315      *
06316      * @par Lua function prototype
06317      * setTimeOptimalEnable() -> number
06318      *
06319      * @par Lua example
06320      * setTimeOptimalEnable(true)
06321      *
06322      * @par JSON-RPC request example
06323      * {"jsonrpc": "2.0", "method": "robl.MotionControl.setTimeOptimalEnable", "params": [true], "id": 1}
06324      *
06325      * @par JSON-RPC response example
06326      * {"id": 1, "jsonrpc": "2.0", "result": 0}
06327      * \endenglish
06328      */
06329 int setTimeOptimalEnable(bool enable);
06330
06331 /**
06332  * @ingroup MotionControl
06333  * \chinese
06334  * 获取时间最优算法状态:
06335  *   true - 开启
06336  *   false - 关闭
06337  *
06338  * @return 返回时间最优算法状态
06339  *
06340  * @throws arcs::common_interface::AuboException
06341  *
06342  * @par Python 函数原型
06343  * isTimeOptimalEnabled(self: pyaubo_sdk.MotionControl) -> float
06344  *
06345  * @par Lua 函数原型
06346  * isTimeOptimalEnabled() -> number
06347  *
06348  * @par Lua 示例
06349  * num = isTimeOptimalEnabled()
06350  *
06351  * @par JSON-RPC 请求示例
06352  * {"jsonrpc": "2.0", "method": "robl.MotionControl.isTimeOptimalEnabled", "params": [], "id": 1}
06353  *
06354  * @par JSON-RPC 响应示例
06355  * {"id": 1, "jsonrpc": "2.0", "result": 1.0}
06356  * \endchinese
06357  * \english
06358  * Get the time optimal algorithm state:
06359  *   true - enable
06360  *   false - disable
06361  *
06362  * @return Returns the time optimal algorithm state
06363  *
06364  * @throws arcs::common_interface::AuboException
06365  *
06366  * @par Python function prototype
06367  * isTimeOptimalEnabled(self: pyaubo_sdk.MotionControl) -> bool
06368  *
06369  * @par Lua function prototype
06370  * isTimeOptimalEnabled() -> boolean
06371  *
06372  * @par Lua example
06373  * flag = isTimeOptimalEnabled()
06374  *
06375  * @par JSON-RPC request example
06376  * {"jsonrpc": "2.0", "method": "robl.MotionControl.isTimeOptimalEnabled", "params": [], "id": 1}
06377  *
06378  * @par JSON-RPC response example
06379  * {"id": 1, "jsonrpc": "2.0", "result": true}
06380  * \endenglish
06381  */
06382 bool isTimeOptimalEnabled();
06383
06384 /**
06385  * @ingroup MotionControl
06386  * \chinese

```



```

06387 * 获取是否支持时间最优算法:
06388 *   true - 支持
06389 *   false - 不支持
06390 *
06391 * @return 返回是否支持时间最优算法
06392 *
06393 * @throws arcs::common_interface::AuboException
06394 *
06395 * @par Python 函数原型
06396 * isTimeOptimalEnabled(self: pyaubo_sdk.MotionControl) -> bool
06397 *
06398 * @par Lua 函数原型
06399 * isSupportedTimeOptimal() -> boolean
06400 *
06401 * @par Lua 示例
06402 * num = isSupportedTimeOptimal()
06403 *
06404 * @par JSON-RPC 请求示例
06405 * {"jsonrpc":"2.0","method":"robl.MotionControl.isSupportedTimeOptimal","params":[],"id":1}
06406 *
06407 * @par JSON-RPC 响应示例
06408 * {"id":1,"jsonrpc":"2.0","result":1.0}
06409 * \endchinese
06410 * \english
06411 * Check whether the time optimal algorithm is supported
06412 *   true - supported
06413 *   false - not supported
06414 *
06415 * @return Return whether the time-optimal algorithm is supported
06416 *
06417 * @throws arcs::common_interface::AuboException
06418 *
06419 * @par Python function prototype
06420 * isSupportedTimeOptimal(self: pyaubo_sdk.MotionControl) -> bool
06421 *
06422 * @par Lua function prototype
06423 * isSupportedTimeOptimal() -> boolean
06424 *
06425 * @par Lua example
06426 * flag = isSupportedTimeOptimal()
06427 *
06428 * @par JSON-RPC request example
06429 * {"jsonrpc":"2.0","method":"robl.MotionControl.isSupportedTimeOptimal","params":[],"id":1}
06430 *
06431 * @par JSON-RPC response example
06432 * {"id":1,"jsonrpc":"2.0","result":true}
06433 * \endenglish
06434 */
06435 bool isSupportedTimeOptimal();
06436
06437 /**
06438 * @ingroup MotionControl
06439 * \chinese
06440 * 设置 TCP 最大线速度
06441 *
06442 * @param v TCP 最大线速度值, 单位为 m/s (米/秒)
06443 *
06444 * @throws arcs::common_interface::AuboException
06445 *
06446 * @par Lua 函数原型
06447 * setTcpMaxLinearVelocity(number) -> number
06448 *
06449 * @par Lua 示例
06450 * setTcpMaxLinearVelocity(0.5) -- 设置 TCP 最大线速度为 0.5 米/秒
06451 *
06452 * @par Python 函数原型
06453 * setTcpMaxLinearVelocity(v: float) -> int
06454 *
06455 * @par Python 示例
06456 * setTcpMaxLinearVelocity(0.5) -- 设置 TCP 最大线速度为 0.5 米/秒
06457 *
06458 * @par JSON-RPC 请求示例
06459 * {"jsonrpc":"2.0","method":"robl.MotionControl.setTcpMaxLinearVelocity","params":[0.5],"id":1}
06460 *
06461 * @par JSON-RPC 响应示例
06462 * {"id":1,"jsonrpc":"2.0","result":0}
06463 * \endchinese
06464 * \english
06465 * Set TCP maximum linear velocity
06466 *
06467 * @param v TCP maximum linear velocity value, unit is m/s
06468 *
06469 * @throws arcs::common_interface::AuboException
06470 *
06471 * @par Lua function prototype
06472 * setTcpMaxLinearVelocity(number) -> number
06473 *

```

```

06474 * @par Lua example
06475 * setTcpMaxLinearVelocity(0.5) -- Set TCP maximum linear velocity to 0.5
06476 * meters per second
06477 *
06478 * @par Python function prototype
06479 * setTcpMaxLinearVelocity(v: float) -> int
06480 *
06481 * @par Python example
06482 * setTcpMaxLinearVelocity(0.5) -- Set TCP maximum linear velocity to 0.5
06483 *
06484 * @par JSON-RPC request example
06485 * {"jsonrpc": "2.0", "method": "rob1.MotionControl.setTcpMaxLinearVelocity", "params": [0.5], "id": 1}
06486 *
06487 * @par JSON-RPC response example
06488 * {"id": 1, "jsonrpc": "2.0", "result": 0}
06489 * \endenglish
06490 */
06491 int setTcpMaxLinearVelocity(double v);
06492
06493 /**
06494 * @ingroup MotionControl
06495 * \chinese
06496 * 获取 TCP 最大线速度
06497 *
06498 * @return 当前 TCP 最大线速度值, 单位为 m/s (米/秒)
06499 *
06500 * @throws arcs::common_interface::AuboException
06501 *
06502 * @par Lua 函数原型
06503 * getTcpMaxLinearVelocity() -> number
06504 *
06505 * @par Lua 示例
06506 * local v = getTcpMaxLinearVelocity() -- 获取 TCP 最大线速度
06507 *
06508 * @par Python 函数原型
06509 * getTcpMaxLinearVelocity() -> float
06510 *
06511 * @par Python 示例
06512 * v = getTcpMaxLinearVelocity() # 获取 TCP 最大线速度
06513 *
06514 * @par JSON-RPC 请求示例
06515 * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getTcpMaxLinearVelocity", "params": [], "id": 1}
06516 *
06517 * @par JSON-RPC 响应示例
06518 * {"id": 1, "jsonrpc": "2.0", "result": 0.5}
06519 * \endchinese
06520 * \english
06521 * Get TCP maximum linear velocity
06522 *
06523 * @return Current TCP maximum linear velocity value, unit is m/s
06524 *
06525 * @throws arcs::common_interface::AuboException
06526 *
06527 * @par Lua function prototype
06528 * getTcpMaxLinearVelocity() -> number
06529 *
06530 * @par Lua example
06531 * local v = getTcpMaxLinearVelocity() -- Get TCP maximum linear velocity
06532 *
06533 * @par Python function prototype
06534 * getTcpMaxLinearVelocity() -> float
06535 *
06536 * @par Python example
06537 * v = getTcpMaxLinearVelocity() # Get TCP maximum linear velocity
06538 *
06539 * @par JSON-RPC request example
06540 * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getTcpMaxLinearVelocity", "params": [], "id": 1}
06541 *
06542 * @par JSON-RPC response example
06543 * {"id": 1, "jsonrpc": "2.0", "result": 0.5}
06544 * \endenglish
06545 */
06546 double getTcpMaxLinearVelocity();
06547
06548 /**
06549 * @ingroup MotionControl
06550 * \chinese
06551 * 重置 TCP 最大线速度
06552 *
06553 * 清除通过 setTcpMaxLinearVelocity 接口设置的 TCP 最大线速度覆盖值
06554 *
06555 * @return 成功返回 0; 失败返回错误码
06556 *
06557 * @throws arcs::common_interface::AuboException
06558 *
06559 * @par Lua 函数原型
06560 * resetTcpMaxLinearVelocity() -> number

```

```

06561      *
06562      * @par Lua 示例
06563      * resetTcpMaxLinearVelocity() -- 重置 TCP 最大线速度为安全配置中的值
06564      *
06565      * @par Python 函数原型
06566      * resetTcpMaxLinearVelocity() -> int
06567      *
06568      * @par Python 示例
06569      * resetTcpMaxLinearVelocity() -- 重置 TCP 最大线速度为安全配置中的值
06570      *
06571      * @par JSON-RPC 请求示例
06572      * {"jsonrpc":"2.0","method":"rob1.MotionControl.resetTcpMaxLinearVelocity","params":[],"id":1}
06573      *
06574      * @par JSON-RPC 响应示例
06575      * {"id":1,"jsonrpc":"2.0","result":0}
06576      * \endchinese
06577      *
06578      * \english
06579      * Reset TCP maximum linear velocity
06580      *
06581      * Clear the TCP maximum linear velocity override set by
06582      * setTcpMaxLinearVelocity
06583      *
06584      * @return Returns 0 on success; error code on failure.
06585      *
06586      * @throws arcs::common_interface::AuboException
06587      *
06588      * @par Lua function prototype
06589      * resetTcpMaxLinearVelocity() -> number
06590      *
06591      * @par Lua example
06592      * resetTcpMaxLinearVelocity() -- Reset TCP maximum linear velocity to
06593      * the value in the safety configuration
06594      *
06595      * @par Python function prototype
06596      * resetTcpMaxLinearVelocity() -> int
06597      *
06598      * @par Python example
06599      * resetTcpMaxLinearVelocity() -- Reset TCP maximum linear velocity
06600      *
06601      * @par JSON-RPC request example
06602      * {"jsonrpc":"2.0","method":"rob1.MotionControl.resetTcpMaxLinearVelocity","params":[],"id":1}
06603      *
06604      * @par JSON-RPC response example
06605      * {"id":1,"jsonrpc":"2.0","result":0}
06606      * \endenglish
06607      */
06608 int resetTcpMaxLinearVelocity();
06609
06610 /**
06611  * @ingroup MotionControl
06612  * \chinese
06613  * 设置轨迹终止点（以当前轨迹段为界）
06614  *
06615  * 显式指定以当前正在执行的轨迹段为边界，终止后续轨迹段的交融（blending）。
06616  * 当前轨迹段将完整执行完毕，但不会与后续轨迹进行速度或位置上的平滑过渡，
06617  * 后续轨迹段将被视为新的独立轨迹起点。
06618  *
06619  * 该接口不会触发急停，也不会中断当前轨迹段的执行，仅影响轨迹段之间的交融行为。
06620  *
06621  * @return 成功返回 0；失败返回错误码
06622  *
06623  * @throws arcs::common_interface::AuboException
06624  *
06625  * @par Lua 函数原型
06626  * setEndPath() -> number
06627  *
06628  * @par Lua 示例
06629  * setEndPath() -- 以当前轨迹段为界，终止后续轨迹交融
06630  *
06631  * @par Python 函数原型
06632  * setEndPath() -> int
06633  *
06634  * @par Python 示例
06635  * setEndPath() # 以当前轨迹段为界，终止后续轨迹交融
06636  *
06637  * @par JSON-RPC 请求示例
06638  * {"jsonrpc":"2.0","method":"rob1.MotionControl.setEndPath","params":[],"id":1}
06639  *
06640  * @par JSON-RPC 响应示例
06641  * {"id":1,"jsonrpc":"2.0","result":0}
06642  * \endchinese
06643  *
06644  * \english
06645  * Set end path (terminate blending at current trajectory segment)
06646  *
06647  * Explicitly specifies the currently executing trajectory segment as the

```

```

06648 * boundary at which trajectory blending is terminated. The current
06649 * trajectory segment will be executed completely, but no velocity or
06650 * position blending will be applied with subsequent trajectory segments.
06651 *
06652 * This interface does not trigger an emergency stop and does not interrupt
06653 * the execution of the current trajectory segment. It only affects the
06654 * blending behavior between trajectory segments.
06655 *
06656 * @return Returns 0 on success; error code on failure.
06657 *
06658 * @throws arcs::common_interface::AuboException
06659 *
06660 * @par Lua function prototype
06661 * setEndPath() -> number
06662 *
06663 * @par Lua example
06664 * setEndPath() -- Terminate trajectory blending at the current segment
06665 *
06666 * @par Python function prototype
06667 * setEndPath() -> int
06668 *
06669 * @par Python example
06670 * setEndPath() # Terminate trajectory blending at the current segment
06671 *
06672 * @par JSON-RPC request example
06673 * {"jsonrpc":"2.0","method":"rob1.MotionControl.setEndPath","params":[],"id":1}
06674 *
06675 * @par JSON-RPC response example
06676 * {"id":1,"jsonrpc":"2.0","result":0}
06677 * \endenglish
06678 */
06679 int setEndPath();
06680
06681 /**
06682 * @ingroup MotionControl
06683 * \chinese
06684 * 获取当前摆焊轨迹点方向
06685 *
06686 * 获取机器人当前执行轨迹点在摆焊（Weave）运动中的方向状态，
06687 * 用于指示该点位于摆焊坐标系 Y 轴的正半轴或负半轴区域。
06688 *
06689 * 当机器人正在执行摆焊轨迹时，该接口返回当前摆焊点所属的摆动方向，
06690 * 可用于判断摆焊运动处于正向摆动阶段或反向摆动阶段。
06691 *
06692 * 当机器人未执行摆焊轨迹时，该接口返回 0。
06693 *
06694 * 该接口为状态查询接口，不会影响当前轨迹执行，也不会改变机器人运动状态。
06695 *
06696 * @return 返回当前摆焊方向，对应 WeaveDirection 枚举值：
06697 * - 0 : NONE 当前未执行摆焊轨迹
06698 * - 1 : POSITIVE 当前位于摆焊坐标系 Y 轴正半轴区域
06699 * - 2 : NEGATIVE 当前位于摆焊坐标系 Y 轴负半轴区域
06700 *
06701 * @throws arcs::common_interface::AuboException
06702 *
06703 * @par Lua 函数原型
06704 * getWeaveDirection() -> number
06705 *
06706 * @par Lua 示例
06707 * local dir = getWeaveDirection()
06708 * if dir == 1 then
06709 *     -- 正方向摆动
06710 * elseif dir == 2 then
06711 *     -- 负方向摆动
06712 * end
06713 *
06714 * @par Python 函数原型
06715 * getWeaveDirection() -> int
06716 *
06717 * @par Python 示例
06718 * dir = getWeaveDirection()
06719 * if dir == 1:
06720 *     # positive direction
06721 *     pass
06722 * elif dir == 2:
06723 *     # negative direction
06724 *     pass
06725 *
06726 * @par JSON-RPC 请求示例
06727 * {"jsonrpc":"2.0","method":"rob1.MotionControl.getWeaveDirection","params":[],"id":1}
06728 *
06729 * @par JSON-RPC 响应示例
06730 * {"id":1,"jsonrpc":"2.0","result":1}
06731 *
06732 * \endchinese
06733 *
06734 * \english

```

```

06735      * Get current weave trajectory direction
06736      *
06737      * Retrieves the directional state of the current trajectory point during
06738      * weave motion, indicating whether the point is located on the positive or
06739      * negative side of the weave coordinate system Y-axis.
06740      *
06741      * When no weave trajectory is active, this interface returns 0.
06742      *
06743      * This is a state query interface and does not affect trajectory execution
06744      * or robot motion.
06745      *
06746      * @return Returns current weave direction corresponding to WeaveDirection
06747      * enum:
06748      * - 0 : NONE      No active weave trajectory
06749      * - 1 : POSITIVE  Positive Y-axis side
06750      * - 2 : NEGATIVE  Negative Y-axis side
06751      *
06752      * @throws arcs::common_interface::AuboException
06753      *
06754      * @par Lua function prototype
06755      * getWeaveDirection() -> number
06756      *
06757      * @par Lua example
06758      * local dir = getWeaveDirection()
06759      *
06760      * @par Python function prototype
06761      * getWeaveDirection() -> int
06762      *
06763      * @par Python example
06764      * dir = getWeaveDirection()
06765      *
06766      * @par JSON-RPC request example
06767      * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getWeaveDirection", "params": [], "id": 1}
06768      *
06769      * @par JSON-RPC response example
06770      * {"id": 1, "jsonrpc": "2.0", "result": 1}
06771      *
06772      * \endenglish
06773      */
06774      int getWeaveDirection();
06775
06776  protected:
06777      void *d_;
06778  };
06779  using MotionControlPtr = std::shared_ptr<MotionControl>;
06780  } // namespace common_interface
06781  } // namespace arcs
06782
06783  #endif // AUBO_SDK_MOTION_CONTROL_INTERFACE_H

```

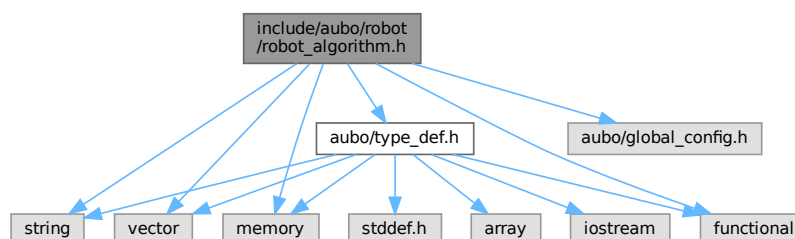
12.26 include/aubo/robot/robot_algorithm.h 文件参考

机器人算法相关的对外接口

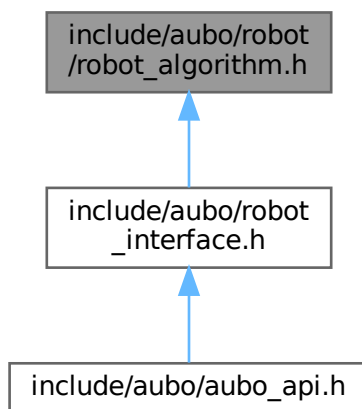
```

#include <string>
#include <vector>
#include <memory>
#include <functional>
#include <aubo/global_config.h>
#include <aubo/type_def.h>
robot_algorithm.h 的引用 (Include) 关系图:

```



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::common_interface::RobotAlgorithm](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::common_interface](#)

类型定义

- using [arcs::common_interface::RobotAlgorithmPtr](#) = std::shared_ptr<[RobotAlgorithm](#)>

12.26.1 详细描述

机器人算法相关的对外接口
在文件 [robot_algorithm.h](#) 中定义.

12.27 robot__algorithm.h

[浏览该文件的文档.](#)

```

00001 /** @file robot_algorithm.h
00002  * @brief 机器人算法相关的对外接口
00003  */
00004 #ifndef AUBO_SDK_ROBOT_ALGORITHM_INTERFACE_H
00005 #define AUBO_SDK_ROBOT_ALGORITHM_INTERFACE_H
00006
00007 #include <string>
00008 #include <vector>
00009 #include <memory>
00010 #include <functional>
00011
00012 #include <aubo/global_config.h>
00013 #include <aubo/type_def.h>
00014
00015 namespace arcs {
00016 namespace common_interface {
00017
00018 // clang-format off
00019
00020 /**
00021  * @defgroup RobotAlgorithm RobotAlgorithm (机器人算法工具)
  
```

```

00022 * @ingroup RobotInterface
00023 * \chinese
00024 * 机器人算法相关的对外接口
00025 * \endchinese
00026 * \english
00027 * Interfaces related to robot algorithms
00028 * \endenglish
00029 */
00030 class ARCS_ABI_EXPORT RobotAlgorithm
00031 {
00032 public:
00033     RobotAlgorithm();
00034     virtual ~RobotAlgorithm();
00035
00036     /**
00037     * @ingroup RobotAlgorithm
00038     * \chinese
00039     * 力传感器标定算法（三点标定法）
00040     *
00041     * @param force 力数据
00042     * @param q 关节角度
00043     * @return 标定结果
00044     *
00045     * @throws arcs::common_interface::AuboException
00046     *
00047     * @par Python 函数原型
00048     * calibrateTcpForceSensor(self: pyaubo_sdk.RobotAlgorithm, arg0:
00049     * List[List[float]], arg1: List[List[float]]) -> Tuple[List[float],
00050     * List[float], float, List[float])
00051     *
00052     * @par Lua 函数原型
00053     * calibrateTcpForceSensor(force: table, q: table) -> table
00054     *
00055     * @par Lua 示例
00056     * cal_table = calibrateTcpForceSensor({10.0,10.0,10.0,-1.2,-1.2,-1.2}, {3.083,1.227,1.098,0.670,-1.870,-0.397})
00057     *
00058     * \endchinese
00059     * \english
00060     * Force sensor calibration algorithm (three-point calibration method)
00061     *
00062     * @param force Force data
00063     * @param q Joint angles
00064     * @return Calibration result
00065     *
00066     * @throws arcs::common_interface::AuboException
00067     *
00068     * @par Python function prototype
00069     * calibrateTcpForceSensor(self: pyaubo_sdk.RobotAlgorithm, arg0:
00070     * List[List[float]], arg1: List[List[float]]) -> Tuple[List[float],
00071     * List[float], float, List[float])
00072     *
00073     * @par Lua function prototype
00074     * calibrateTcpForceSensor(force: table, q: table) -> table
00075     *
00076     * @par Lua example
00077     * cal_table = calibrateTcpForceSensor({10.0,10.0,10.0,-1.2,-1.2,-1.2}, {3.083,1.227,1.098,0.670,-1.870,-0.397})
00078     *
00079     * \endenglish
00080     */
00081     ForceSensorCalibResult calibrateTcpForceSensor(
00082         const std::vector<std::vector<double>> &forces,
00083         const std::vector<std::vector<double>> &poses);
00084
00085     /**
00086     * @ingroup RobotAlgorithm
00087     * \chinese
00088     * 力传感器标定算法（三点标定法）
00089     * @param forces
00090     * @param poses
00091     * @return force_offset, com, mass, angle error
00092     *
00093     * @throws arcs::common_interface::AuboException
00094     * \endchinese
00095     * \english
00096     * Force sensor calibration algorithm (three-point calibration method)
00097     * @param forces
00098     * @param poses
00099     * @return force_offset, com, mass, angle error
00100     *
00101     * @throws arcs::common_interface::AuboException
00102     * \endenglish
00103     */
00104     ForceSensorCalibResultWithError calibrateTcpForceSensor2(
00105         const std::vector<std::vector<double>> &forces,
00106         const std::vector<std::vector<double>> &poses);
00107
00108     /**

```

```

00109     * @ingroup RobotAlgorithm
00110     * \chinese
00111     * 力传感器偏置标定算法
00112     *
00113     * @param force 力数据
00114     * @param poses 位姿
00115     * @param mass 质量, 单位: kg
00116     * @param cog 重心, 单位: m, 形式为 (CoGx, CoGy, CoGz)
00117     * @return 标定结果
00118     *
00119     * @throws arcs::common_interface::AuboException
00120     * \endchinese
00121     * \english
00122     * Force sensor offset calibration algorithm
00123     *
00124     * @param forces Force data
00125     * @param poses
00126     * @param m Mass, unit: kg
00127     * @param cog Center of gravity, unit: m, format (CoGx, CoGy, CoGz)
00128     * @return Calibration result
00129     *
00130     * @throws arcs::common_interface::AuboException
00131     * \endenglish
00132     */
00133 ResultWithErrno calibrateTcpForceSensor3(
00134     const std::vector<std::vector<double>> &forces,
00135     const std::vector<std::vector<double>> &poses, const double &mass,
00136     const std::vector<double>&cog);
00137
00138 /**
00139     * @ingroup RobotAlgorithm
00140     * \chinese
00141     * 基于电流的负载辨识算法接口
00142     *
00143     * 需要采集空载时运行激励轨迹的位置、速度、电流以及带负载时运行激励轨迹的位置、速度、电流
00144     *
00145     * @param data_file_no_payload
00146     * 空载时运行激励轨迹各关节数据的文件路径 (.csv 格式), 共 18 列, 依次为 6 个关节位置、6 个关节速度、6 个关节电流
00147     * @param data_file_with_payload
00148     * 带负载运行激励轨迹各关节数据的文件路径 (.csv 格式), 共 18 列, 依次为 6 个关节位置、6 个关节速度、6 个关节电流
00149     * @return 辨识的结果
00150     *
00151     * @throws arcs::common_interface::AuboException
00152     *
00153     * @par Python 函数原型
00154     * payloadIdentify(self: pyaubo_sdk.RobotAlgorithm, arg0: List[List[float]],
00155     * arg1: List[List[float]]) -> Tuple[List[float], List[float], float,
00156     * List[float]]
00157     *
00158     * @par Lua 函数原型
00159     * payloadIdentify(data_with_payload: table, data_with_payload: table) ->
00160     * table
00161     * \endchinese
00162     * \english
00163     * Payload identification algorithm interface based on current
00164     *
00165     * Requires collecting position, velocity, and current data when running the excitation trajectory without load,
    as well as with load.
00166     *
00167     * @param data_file_no_payload
00168     * File path of joint data when running the excitation trajectory without load (.csv format), 18 columns in
    total: 6 joint positions, 6 joint velocities, 6 joint currents
00169     * @param data_file_with_payload
00170     * File path of joint data when running the excitation trajectory with load (.csv format), 18 columns in total:
    6 joint positions, 6 joint velocities, 6 joint currents
00171     * @return Identification result
00172     *
00173     * @throws arcs::common_interface::AuboException
00174     *
00175     * @par Python function prototype
00176     * payloadIdentify(self: pyaubo_sdk.RobotAlgorithm, arg0: List[List[float]],
00177     * arg1: List[List[float]]) -> Tuple[List[float], List[float], float,
00178     * List[float]]
00179     *
00180     * @par Lua function prototype
00181     * payloadIdentify(data_with_payload: table, data_with_payload: table) ->
00182     * table
00183     * \endenglish
00184     */
00185 int payloadIdentify(const std::string &data_file_no_payload,
00186     const std::string &data_file_with_payload);
00187
00188 /**
00189     * @ingroup RobotAlgorithm
00190     * \chinese
00191     * 新版基于电流的负载辨识算法接口
00192     *

```



```

00193     * 需要采集带载时运行最少三个点的位置、速度、加速度、电流、温度、末端传感器数据、底座数据
00194     *
00195     * @param data
00196     * 带负载的各关节数据的文件路径 (.csv 格式), 共 42 列, 末端传感器数据、底座数据默认为 0
00197     * @return 辨识的结果
00198     * \endchinese
00199     * \english
00200     * New version of payload identification algorithm interface based on current
00201     *
00202     * Requires collecting at least three points of position, velocity, acceleration, current, temperature, end
sensor data, and base data when running with load
00203     *
00204     * @param data
00205     * File path of joint data with load (.csv format), 42 columns in total, end sensor data and base data default
to 0
00206     * @return Identification result
00207     * \endenglish
00208     */
00209     int payloadIdentify1(const std::string &file_name);
00210
00211     /**
00212     * @ingroup RobotAlgorithm
00213     * \chinese
00214     * 负载辨识是否计算完成
00215     * @return 完成返回 0; 正在进行中返回 1; 计算失败返回 <0;
00216     *
00217     * @throws arcs::common_interface::AuboException
00218     *
00219     * @par JSON-RPC 请求示例
00220     * {"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.payloadCalculateFinished", "params": [], "id": 1}
00221     *
00222     * @par JSON-RPC 响应示例
00223     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00224     *
00225     * \endchinese
00226     * \english
00227     * Whether payload identification calculation is finished
00228     * @return 0 if finished; 1 if in progress; <0 if failed;
00229     *
00230     * @throws arcs::common_interface::AuboException
00231     *
00232     * @par JSON-RPC request example
00233     * {"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.payloadCalculateFinished", "params": [], "id": 1}
00234     *
00235     * @par JSON-RPC response example
00236     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00237     *
00238     * \endenglish
00239     */
00240     int payloadCalculateFinished();
00241
00242     /**
00243     * @ingroup RobotAlgorithm
00244     * \chinese
00245     * 获取负载辨识结果
00246     * @return
00247     *
00248     * @throws arcs::common_interface::AuboException
00249     *
00250     * @par JSON-RPC 请求示例
00251     * {"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.getPayloadIdentifyResult", "params": [], "id": 1}
00252     *
00253     * @par JSON-RPC 响应示例
00254     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, [], [], []]}
00255     *
00256     * \endchinese
00257     * \english
00258     * Get the result of payload identification
00259     * @return
00260     *
00261     * @throws arcs::common_interface::AuboException
00262     *
00263     * @par JSON-RPC request example
00264     * {"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.getPayloadIdentifyResult", "params": [], "id": 1}
00265     *
00266     * @par JSON-RPC response example
00267     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, [], [], []]}
00268     *
00269     * \endenglish
00270     */
00271     Payload getPayloadIdentifyResult();
00272
00273     /**
00274     * @ingroup RobotAlgorithm
00275     * \chinese
00276     * 关节摩擦力模型辨识算法接口
00277     *

```

```

00278 * @param q 关节角度
00279 * @param qd 关节速度
00280 * @param qdd 关节加速度
00281 * @param temp 关节温度
00282 * @return 是否辨识成功
00283 *
00284 * @throws arcs::common_interface::AuboException
00285 *
00286 * @par Python 函数原型
00287 * frictionModelIdentify(self: pyaubo_sdk.RobotAlgorithm, arg0:
00288 * List[List[float]], arg1: List[List[float]], arg2: List[List[float]],
00289 * arg3: List[List[float]]) -> bool
00290 *
00291 * @par Lua 函数原型
00292 * frictionModelIdentify(q: table, qd: table, qdd: table, temp: table) ->
00293 * boolean
00294 *
00295 * @par Lua 示例
00296 * Identify_result = frictionModelIdentify({3.083,1.227,1.098,0.670,-1.870,-0.397},
00297 * {10.0,10.0,10.0,10.0,10.0,10.0},{20.0,20.0,20.0,20.0,20.0,20.0},{30.0,30.0,30.0,30.0,30.0,30.0})
00298 *
00299 * \endchinese
00300 * \english
00301 * Joint friction model identification algorithm interface
00302 *
00303 * @param q Joint positions
00304 * @param qd Joint velocities
00305 * @param qdd Joint accelerations
00306 * @param temp Joint temperatures
00307 * @return Whether identification succeeded
00308 *
00309 * @throws arcs::common_interface::AuboException
00310 *
00311 * @par Python function prototype
00312 * frictionModelIdentify(self: pyaubo_sdk.RobotAlgorithm, arg0:
00313 * List[List[float]], arg1: List[List[float]], arg2: List[List[float]],
00314 * arg3: List[List[float]]) -> bool
00315 *
00316 * @par Lua function prototype
00317 * frictionModelIdentify(q: table, qd: table, qdd: table, temp: table) ->
00318 * boolean
00319 *
00320 * @par Lua example
00321 * Identify_result = frictionModelIdentify({3.083,1.227,1.098,0.670,-1.870,-0.397},
00322 * {10.0,10.0,10.0,10.0,10.0,10.0},{20.0,20.0,20.0,20.0,20.0,20.0},{30.0,30.0,30.0,30.0,30.0,30.0})
00323 *
00324 * \endenglish
00325 */
00326 bool frictionModelIdentify(const std::vector<std::vector<double>> &q,
00327                             const std::vector<std::vector<double>> &qd,
00328                             const std::vector<std::vector<double>> &qdd,
00329                             const std::vector<std::vector<double>> &temp);
00330
00331 /**
00332 * @ingroup RobotAlgorithm
00333 * \chinese
00334 * 工件坐标系标定算法接口（需要在调用之前正确的设置机器人的 TCP 偏移）
00335 * 输入多组关节角度和标定类型，输出工件坐标系位姿（相对于机器人基坐标系）
00336 *
00337 * @param q 关节角度
00338 * @param type 标定类型
00339 * @return 计算结果（工件坐标系位姿）以及错误代码
00340 *
00341 * @throws arcs::common_interface::AuboException
00342 *
00343 * @par Python 函数原型
00344 * calibWorkpieceCoordinatePara(self: pyaubo_sdk.RobotAlgorithm, arg0:
00345 * List[List[float]], arg1: int) -> Tuple[List[float], int]
00346 *
00347 * @par Lua 函数原型
00348 * calibWorkpieceCoordinatePara(q: table, type: number) -> table, number
00349 *
00350 * @par Lua 示例
00351 * coord_pose,coord_num = calibWorkpieceCoordinatePara({3.083,1.227,1.098,0.670,-1.870,-0.397},1)
00352 *
00353 * \endchinese
00354 * \english
00355 * Workpiece coordinate calibration algorithm interface (requires correct TCP offset set before calling)
00356 * Input multiple sets of joint angles and calibration type, output workpiece coordinate pose (relative to robot
base)
00357 *
00358 * @param q Joint angles
00359 * @param type Calibration type
00360 * @return Calculation result (workpiece coordinate pose) and error code
00361 *
00362 * @throws arcs::common_interface::AuboException
00363 *

```

```

00364      * @par Python function prototype
00365      * calibWorkpieceCoordinatePara(self: pyaubo_sdk.RobotAlgorithm, arg0:
00366      * List[List[float]], arg1: int) -> Tuple[List[float], int]
00367      *
00368      * @par Lua function prototype
00369      * calibWorkpieceCoordinatePara(q: table, type: number) -> table, number
00370      *
00371      * @par Lua example
00372      * coord_pose, coord_num = calibWorkpieceCoordinatePara({3.083,1.227,1.098,0.670,-1.870,-0.397},1)
00373      *
00374      * \endenglish
00375      */
00376 ResultWithErrno calibWorkpieceCoordinatePara(
00377     const std::vector<std::vector<double>> &q, int type);
00378
00379 /**
00380  * @ingroup RobotAlgorithm
00381  * \chinese
00382  * 动力学正解
00383  *
00384  * @param q 关节角
00385  * @param torqs
00386  * @return 计算结果以及错误代码
00387  *
00388  * @throws arcs::common_interface::AuboException
00389  *
00390  * @par Python 函数原型
00391  * forwardDynamics(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float], arg1:
00392  * List[float]) -> Tuple[List[float], int]
00393  *
00394  * @par Lua 函数原型
00395  * forwardDynamics(q: table, torqs: table) -> table, number
00396  *
00397  * @par Lua 示例
00398  * Dynamics, fk_result = forwardDynamics({3.083,1.227,1.098,0.670,-1.870,-0.397},{0.0,0.0,0.0,0.0,0.0,0.0})
00399  *
00400  * \endchinese
00401  * \english
00402  * Forward dynamics
00403  *
00404  * @param q Joint angles
00405  * @param torqs
00406  * @return Calculation result and error code
00407  *
00408  * @throws arcs::common_interface::AuboException
00409  *
00410  * @par Python function prototype
00411  * forwardDynamics(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float], arg1:
00412  * List[float]) -> Tuple[List[float], int]
00413  *
00414  * @par Lua function prototype
00415  * forwardDynamics(q: table, torqs: table) -> table, number
00416  *
00417  * @par Lua example
00418  * Dynamics, fk_result = forwardDynamics({3.083,1.227,1.098,0.670,-1.870,-0.397},{0.0,0.0,0.0,0.0,0.0,0.0})
00419  *
00420  * \endenglish
00421  */
00422 ResultWithErrno forwardDynamics(const std::vector<double> &q,
00423     const std::vector<double> &torqs);
00424
00425 /**
00426  * @ingroup RobotAlgorithm
00427  * \chinese
00428  * 动力学正解，基于给定的 TCP 偏移
00429  *
00430  * @param q 关节角
00431  * @param torqs
00432  * @param tcp_offset TCP 偏移
00433  * @return 计算结果以及错误代码，同 forwardDynamics
00434  *
00435  * @throws arcs::common_interface::AuboException
00436  *
00437  * @par Python 函数原型
00438  * forwardDynamics1(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float], arg1:
00439  * List[float], arg2: List[float]) -> Tuple[List[float], int]
00440  *
00441  * @par Lua 函数原型
00442  * forwardDynamics1(q: table, torqs: table, tcp_offset: table) -> table, number
00443  *
00444  * @par Lua 示例
00445  * Dynamics, fk_result = forwardDynamics1({3.083,1.227,1.098,0.670,-1.870,-0.397},{0.0,0.0,0.0,0.0,0.0,0.0},
00446  {0.0,0.13201,0.03879,0,0,0})
00447  *
00448  * \endchinese
00449  * \english
00449  * Forward dynamics based on the given TCP offset

```

```

00450      *
00451      * @param q Joint angles
00452      * @param torqs
00453      * @param tcp_offset TCP offset
00454      * @return Calculation result and error code, same as forwardDynamics
00455      *
00456      * @throws arcs::common_interface::AuboException
00457      *
00458      * @par Python function prototype
00459      * forwardDynamics(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float], arg1:
00460      * List[float], arg2: List[float]) -> Tuple[List[float], int]
00461      *
00462      * @par Lua function prototype
00463      * forwardDynamics1(q: table, torqs: table, tcp_offset: table) -> table, number
00464      *
00465      * @par Lua example
00466      * Dynamics, fk_result = forwardDynamics1({3.083,1.227,1.098,0.670,-1.870,-0.397},{0.0,0.0,0.0,0.0,0.0,0.0},
00467      {0.0,0.13201,0.03879,0,0,0})
00468      * \endenglish
00469      */
00470      ResultWithErrno forwardDynamics1(const std::vector<double> &q,
00471      const std::vector<double> &torqs,
00472      const std::vector<double> &tcp_offset);
00473
00474      /**
00475      * @ingroup RobotAlgorithm
00476      * \chinese
00477      * 运动学正解，基于激活的 TCP 偏移（最近的通过 setTcpOffset 设置的参数）
00478      * 输入关节角度，输出 TCP 位姿
00479      *
00480      * @param q 关节角
00481      * @return TCP 位姿和正解结果是否有效
00482      * 返回值的第一个参数为正解结果，第二个为正解错误码，错误码返回值列表如下
00483      * 0 - 成功
00484      * -1 - 机械臂状态不对（未初始化完成，可尝试再次调用）
00485      * -5 - 输入的关节角无效（维度错误）
00486      * @throws arcs::common_interface::AuboException
00487      *
00488      * @par Python 函数原型
00489      * forwardKinematics(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float]) ->
00490      * Tuple[List[float], int]
00491      *
00492      * @par Lua 函数原型
00493      * forwardKinematics(q: table) -> table, number
00494      *
00495      * @par Lua 示例
00496      * pose, fk_result = forwardKinematics({3.083,1.227,1.098,0.670,-1.870,-0.397})
00497      *
00498      * @par JSON-RPC 请求示例
00499      * {"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.forwardKinematics", "params":
00500      [[3.083688522170976,1.2273215976885394,1.098072739631141,0.6705738810610149,-1.870715392248607,-0.39708546603119627]], "id": 1}
00501      *
00502      * @par JSON-RPC 响应示例
00503      * {"id": 1, "jsonrpc": "2.0", "result":
00504      [[0.7137448715395925,0.08416057568819092,0.6707994191515292,2.4599818776908724,0.4789772388601265,1.6189630435878408],0]}
00505      * \endchinese
00506      * \english
00507      * Forward kinematics, based on the activated TCP offset (the most recently set via setTcpOffset)
00508      * Input joint angles, output TCP pose
00509      *
00510      * @param q Joint angles
00511      * @return TCP pose and whether the result is valid
00512      * The first parameter of the return value is the forward kinematics result, the second is the error code. Error
00513      codes are as follows:
00514      * 0 - Success
00515      * -1 - The status of the robot is incorrect (not initialized yet, you can try calling it again)
00516      * -5 - The input joint angles is invalid (dimension error)
00517      * @throws arcs::common_interface::AuboException
00518      *
00519      * @par Python function prototype
00520      * forwardKinematics(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float]) ->
00521      * Tuple[List[float], int]
00522      *
00523      * @par Lua function prototype
00524      * forwardKinematics(q: table) -> table, number
00525      *
00526      * @par Lua example
00527      * pose, fk_result = forwardKinematics({3.083,1.227,1.098,0.670,-1.870,-0.397})
00528      *
00529      * @par JSON-RPC request example
00530      * {"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.forwardKinematics", "params":
00531      [[3.083688522170976,1.2273215976885394,1.098072739631141,0.6705738810610149,-1.870715392248607,-0.39708546603119627]], "id": 1}
00532      *
00533      * @par JSON-RPC response example
00534      * {"id": 1, "jsonrpc": "2.0", "result":
00535      [[0.7137448715395925,0.08416057568819092,0.6707994191515292,2.4599818776908724,0.4789772388601265,1.6189630435878408],0]}

```

```

00531     * \endenglish
00532     */
00533     ResultWithErrno forwardKinematics(const std::vector<double> &q);
00534
00535     /**
00536     * @ingroup RobotAlgorithm
00537     * \chinese
00538     * 运动学正解
00539     * 输入关节角度, 输出 TCP 位姿
00540     * @param q 关节角
00541     * @param tcp_offset tcp 偏移
00542     * @return TCP 位姿和正解结果是否有效
00543     * 返回值的第一个参数为正解结果, 第二个为正解错误码, 错误码返回值列表如下
00544     * 0 - 成功
00545     * -1 - 机械臂状态不对 (未初始化完成, 可尝试再次调用)
00546     * -5 - 输入的关节角或 tcp 偏移无效 (维度错误)
00547     * @throws arcs::common_interface::AuboException
00548     *
00549     * @par Python 函数原型
00550     * forwardKinematics(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float],
00551     * arg1: List[float]) -> Tuple[List[float], int]
00552     *
00553     * @par Lua 函数原型
00554     * forwardKinematics1(q: table, tcp_offset: table) -> table, number
00555     *
00556     * @par Lua 示例
00557     * pose, fk_result = forwardKinematics1({3.083,1.227,1.098,0.670,-1.870,-0.397},{0.0,0.13201,0.03879,0,0,0})
00558     *
00559     * @par JSON-RPC 请求示例
00560     * {"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.forwardKinematics1","params":
[[3.083688522170976,1.2273215976885394,1.098072739631141,0.6705738810610149,-1.870715392248607,-0.39708546603119627],
[0.0,
00561     * 0.13201,0.03879,0,0,0]], "id":1}
00562     *
00563     * @par JSON-RPC 响应示例
00564     * {"id":1,"jsonrpc":"2.0","result":
[[0.7137636726659518,0.0837705432006433,0.6710022027216355,2.459981877690872,0.4789772388601267,1.6189630435878408],0]}
00565     *
00566     * @since 0.24.1
00567     * \endchinese
00568     * \english
00569     * Forward kinematics
00570     * Input joint angles, output TCP pose
00571     *
00572     * @param q Joint angles
00573     * @param tcp_offset TCP offset
00574     * @return TCP pose and whether the result is valid
00575     * The first parameter of the return value is the forward kinematics result, the second is the error code. Error
codes are as follows:
00576     * 0 - Success
00577     * -1 - The status of the robot is incorrect (not initialized yet, you can try calling it again)
00578     * -5 - The input joint angles or tcp offset is invalid (dimension error)
00579     * @throws arcs::common_interface::AuboException
00580     *
00581     * @par Python function prototype
00582     * forwardKinematics(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float],
00583     * arg1: List[float]) -> Tuple[List[float], int]
00584     *
00585     * @par Lua function prototype
00586     * forwardKinematics1(q: table, tcp_offset: table) -> table, number
00587     *
00588     * @par Lua example
00589     * pose, fk_result = forwardKinematics1({3.083,1.227,1.098,0.670,-1.870,-0.397},{0.0,0.13201,0.03879,0,0,0})
00590     *
00591     * @par JSON-RPC request example
00592     * {"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.forwardKinematics1","params":
[[3.083688522170976,1.2273215976885394,1.098072739631141,0.6705738810610149,-1.870715392248607,-0.39708546603119627],
[0.0,
00593     * 0.13201,0.03879,0,0,0]], "id":1}
00594     *
00595     * @par JSON-RPC response example
00596     * {"id":1,"jsonrpc":"2.0","result":
[[0.7137636726659518,0.0837705432006433,0.6710022027216355,2.459981877690872,0.4789772388601267,1.6189630435878408],0]}
00597     *
00598     * @since 0.24.1
00599     * \endenglish
00600     */
00601     ResultWithErrno forwardKinematics1(const std::vector<double> &q,
00602     const std::vector<double> &tcp_offset);
00603
00604     /**
00605     * @ingroup RobotAlgorithm
00606     * \chinese
00607     * 运动学正解 (忽略 TCP 偏移值)
00608     *
00609     * @param q 关节角
00610     * @return 法兰盘中心位姿和正解结果是否有效

```

```

00611     * 返回值的第一个参数为正解结果，第二个为正解错误码，错误码返回值列表如下
00612     * 0 - 成功
00613     * -1 - 机械臂状态不对（未初始化完成，可尝试再次调用）
00614     * -5 - 输入的关节角无效（维度错误）
00615     *
00616     * @throws arcs::common_interface::AuboException
00617     *
00618     * @par Lua 函数原型
00619     * forwardToolKinematics(q: table) -> table, number
00620     *
00621     * @par Lua 示例
00622     * pose, fk_result = forwardToolKinematics({3.083,1.227,1.098,0.670,-1.870,-0.397})
00623     *
00624     * @par JSON-RPC 请求示例
00625     * {"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.forwardToolKinematics","params":
[[3.083688522170976,1.2273215976885394,1.098072739631141,0.6705738810610149,-1.870715392248607,-0.39708546603119627]], "id":1}
00626     *
00627     * @par JSON-RPC 响应示例
00628     * {"id":1,"jsonrpc":"2.0","result":
[[0.5881351149440136,0.05323734739426938,0.623922550656701,2.4599818776908724,0.4789772388601265,1.6189630435878408],0]}
00629     * \endchinese
00630     * \english
00631     * Forward kinematics (ignoring TCP offset)
00632     *
00633     * @param q Joint angles
00634     * @return Flange center pose and whether the result is valid
00635     * The first parameter of the return value is the forward kinematics result, the second is the error code. Error
codes are as follows:
00636     * 0 - Success
00637     * -1 - The status of the robot is incorrect (not initialized yet, you can try calling it again)
00638     * -5 - The input joint angles is invalid (dimension error)
00639     *
00640     * @par Lua function prototype
00641     * forwardToolKinematics(q: table) -> table, number
00642     *
00643     * @par Lua example
00644     * pose, fk_result = forwardToolKinematics({3.083,1.227,1.098,0.670,-1.870,-0.397})
00645     *
00646     * @throws arcs::common_interface::AuboException
00647     *
00648     * @par JSON-RPC request example
00649     * {"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.forwardToolKinematics","params":
[[3.083688522170976,1.2273215976885394,1.098072739631141,0.6705738810610149,-1.870715392248607,-0.39708546603119627]], "id":1}
00650     *
00651     * @par JSON-RPC response example
00652     * {"id":1,"jsonrpc":"2.0","result":
[[0.5881351149440136,0.05323734739426938,0.623922550656701,2.4599818776908724,0.4789772388601265,1.6189630435878408],0]}
00653     * \endenglish
00654     */
00655     ResultWithErrno forwardToolKinematics(const std::vector<double> &q);
00656
00657     /**
00658     * @ingroup RobotAlgorithm
00659     * \chinese
00660     * 运动学正解，基于激活的 TCP 偏移（最近的通过 setTcpOffset 设置的参数）
00661     * 输入关节角度，输出各连杆位姿
00662     *
00663     * @param q 关节角
00664     * @return 各个连杆位姿和正解结果是否有效
00665     * 返回值的第一个参数为正解结果，第二个为正解错误码，错误码返回值列表如下
00666     * 0 - 成功
00667     * -1 - 机械臂状态不对（未初始化完成，可尝试再次调用）
00668     * -5 - 输入的关节角无效（维度错误）
00669     * @throws arcs::common_interface::AuboException
00670     *
00671     * @par Python 函数原型
00672     * forwardKinematicsAll(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float]) ->
00673     * Tuple[List[List[float]], int]
00674     *
00675     * @par Lua 函数原型
00676     * forwardKinematicsAll(q: table) -> table, number
00677     *
00678     * @par Lua 示例
00679     * poses, fk_result = forwardKinematicsAll({3.083,1.227,1.098,0.670,-1.870,-0.397})
00680     *
00681     * @par JSON-RPC 请求示例
00682     * {"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.forwardKinematicsAll","params":{"q":
[-7.32945e-11,-0.261799,1.74533,0.436332,1.5708,-2.14136e-10]], "id":1}
00683     *
00684     * @par JSON-RPC 响应示例
00685     * {"id":1,"jsonrpc":"2.0","result":[[[0.0010004,0.0,0.122,0.0,0.0,3.141592653589793],
[0.0010003999910823934,-0.12166795404953117,0.12205392567412496,1.5690838552993878,-1.3089969602251492,0.0016541204887245322],
00687     * [0.10659809449330016,-0.12166124765791932,0.5161518260534821,-1.5718540206872664,0.43633111081725523,-0.002370419930605744],
00688     * [0.4482255342315581,-0.1226390142692,0.3568490758201224,-0.000788980482890453,1.5665204619271216,-1.5719618592940798],

```



```

00689     *
00690     * [0.5519603828181741,-0.12282266347465487,0.35639753697117343,1.5707938201843654,0.0042721909279596635,1.569044569428874],
00691     * [0.5513243939877184,-0.12401224959696328,0.2615193425222568,-3.1349118101994096,0.004272190926529066,1.569044569643007],
00692     * [0.7494883076798231,-0.025647479212994567,-0.04023458911333461,-3.1349118101994096,0.004272190926529066,1.569044569643007]],0]}
00692     * \endchinese
00693     * \english
00694     * Forward kinematics, based on the activated TCP offset (the most recently set via setTcpOffset)
00695     * Input joint angles, output links poses
00696     *
00697     * @param q Joint angles
00698     * @return links poses and whether the result is valid
00699     * The first parameter of the return value is the forward kinematics result, the second is the error code. Error
codes are as follows:
00700     * 0 - Success
00701     * -1 - The status of the robot is incorrect (not initialized yet, you can try calling it again)
00702     * -5 - The input joint angles is invalid (dimension error)
00703     * @throws arcs::common_interface::AuboException
00704     *
00705     * @par Python function prototype
00706     * forwardKinematicsAll(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float]) ->
00707     * Tuple[List[List[float]], int]
00708     *
00709     * @par Lua function prototype
00710     * forwardKinematicsAll(q: table) -> table, number
00711     *
00712     * @par Lua example
00713     * pose, fk_result = forwardKinematicsAll({3.083,1.227,1.098,0.670,-1.870,-0.397})
00714     *
00715     * @par JSON-RPC request example
00716     * {"jsonrpc":"2.0","method":"robl.RobotAlgorithm.forwardKinematicsAll","params":{"q":
[-7.32945e-11,-0.261799,1.74533,0.436332,1.5708,-2.14136e-10]},"id":1}
00717     *
00718     * @par JSON-RPC response example
00719     * {"id":1,"jsonrpc":"2.0","result":[[[0.0010004,0.0,0.122,0.0,0.0,3.141592653589793],
00720     *
00721     * [0.0010003999910823934,-0.12166795404953117,0.12205392567412496,1.5690838552993878,-1.3089969602251492,0.0016541204887245322],
00722     * [0.10659809449330016,-0.12166124765791932,0.5161518260534821,-1.5718540206872664,0.43633111081725523,-0.002370419930605744],
00723     * [0.4482255342315581,-0.1226390142692,0.3568490758201224,-0.000788980482890453,1.5665204619271216,-1.5719618592940798],
00724     * [0.5519603828181741,-0.12282266347465487,0.35639753697117343,1.5707938201843654,0.0042721909279596635,1.569044569428874],
00725     * [0.5513243939877184,-0.12401224959696328,0.2615193425222568,-3.1349118101994096,0.004272190926529066,1.569044569643007],
00726     * [0.7494883076798231,-0.025647479212994567,-0.04023458911333461,-3.1349118101994096,0.004272190926529066,1.569044569643007]],0]}
00726     * \endenglish
00727     */
00728     ResultWithErrno1 forwardKinematicsAll(const std::vector<double> &q);
00729
00730     /**
00731     * @ingroup RobotAlgorithm
00732     * \chinese
00733     * 运动学逆解
00734     * 输入 TCP 位姿和参考关节角度, 输出关节角度
00735     *
00736     * @param qnear 参考关节角
00737     * @param pose TCP 位姿
00738     * @return 关节角和逆解结果是否有效
00739     * 返回值的第一个参数为逆解结果, 第二个为逆解错误码, 错误码返回列表如下
00740     * 0 - 成功
00741     * -1 - 机械臂状态不对 (未初始化完成, 可尝试再次调用)
00742     * -5 - 输入的参考关节角或 TCP 位姿无效 (维度错误)
00743     * -23 - 逆解计算不收敛, 计算出错
00744     * -24 - 逆解计算超出机器人最大限制
00745     * -25 - 逆解输入配置存在错误
00746     * -26 - 逆解雅可比矩阵计算失败
00747     * -27 - 目标点存在解析解, 但均不满足选解条件
00748     * -28 - 逆解返回未知类型错误
00749     * 若错误码非 0, 则返回值的第一个参数为输入参考关节角 qnear
00750     *
00751     * @throws arcs::common_interface::AuboException
00752     *
00753     * @par Python 函数原型
00754     * inverseKinematics(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float],
00755     * arg1: List[float]) -> Tuple[List[float], int]
00756     *
00757     * @par Lua 函数原型
00758     * inverseKinematics(qnear: table, pose: table) -> table, int
00759     *
00760     * @par Lua 示例
00761     * joint,ik_result = inverseKinematics({0,0,0,0,0,0},{0.81665,-0.20419,0.43873,-3.135,0.004,1.569})
00762     *
00763     * @par JSON-RPC 请求示例
00764     * {"jsonrpc":"2.0","method":"robl.RobotAlgorithm.inverseKinematics","params":[[0,0,0,0,0,0],

```

```

00765     [0.71374,0.08417,0.6708,2.46,0.479,1.619]],"id":1}
00766     * @par JSON-RPC 响应示例
00767     * {"id":1,"jsonrpc":"2.0","result":
00768     * [[3.083688522170976,1.2273215976885394,1.098072739631141,0.6705738810610149,-1.870715392248607,-0.39708546603119627],0]}
00769     * \endchinese
00770     * \english
00771     * Inverse kinematics
00772     * Input TCP pose and reference joint angles, output joint angles
00773     * @param qnear Reference joint angles
00774     * @param pose TCP pose
00775     * @return Joint angles and whether the inverse kinematics result is valid
00776     * The first parameter of the return value is the inverse kinematics result, the second is the error code. Error
codes are as follows:
00777     * 0 - Success
00778     * -1 - The status of the robot is incorrect (not initialized yet, you can try calling it again)
00779     * -5 - The input reference joint angles or tcp pose is invalid (dimension error)
00780     * -23 - Inverse kinematics calculation does not converge, calculation error
00781     * -24 - Inverse kinematics calculation exceeds robot limits
00782     * -25 - Inverse kinematics input configuration error
00783     * -26 - Inverse kinematics Jacobian calculation failed
00784     * -27 - Analytical solution exists but none satisfy the selection criteria
00785     * -28 - Unknown error in inverse kinematics
00786     * If the error code is not 0, the first parameter of the return value is the input reference joint angles qnear
00787     *
00788     * @throws arcs::common_interface::AuboException
00789     *
00790     * @par Python function prototype
00791     * inverseKinematics(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float],
00792     * arg1: List[float]) -> Tuple[List[float], int]
00793     *
00794     * @par Lua function prototype
00795     * inverseKinematics(qnear: table, pose: table) -> table, int
00796     *
00797     * @par Lua example
00798     * joint,ik_result = inverseKinematics({0,0,0,0,0,0},{0.81665,-0.20419,0.43873,-3.135,0.004,1.569})
00799     *
00800     * @par JSON-RPC request example
00801     * {"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.inverseKinematics","params":[[0,0,0,0,0,0],
00802     * [0.71374,0.08417,0.6708,2.46,0.479,1.619]],"id":1}
00803     *
00804     * @par JSON-RPC response example
00805     * {"id":1,"jsonrpc":"2.0","result":
00806     * [[3.083688522170976,1.2273215976885394,1.098072739631141,0.6705738810610149,-1.870715392248607,-0.39708546603119627],0]}
00807     * \endenglish
00808     */
00809     ResultWithErrno inverseKinematics(const std::vector<double> &qnear,
00810     const std::vector<double> &pose);
00811     /**
00812     * @ingroup RobotAlgorithm
00813     * \chinese
00814     * 运动学逆解
00815     * 输入 TCP 位姿和参考关节角度, 输出关节角度
00816     *
00817     * @param qnear 参考关节角
00818     * @param pose TCP 位姿
00819     * @param tcp_offset TCP 偏移
00820     * @return 关节角和逆解结果是否有效, 同 inverseKinematics
00821     * @throws arcs::common_interface::AuboException
00822     *
00823     * @par Python 函数原型
00824     * inverseKinematics1(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float],
00825     * arg1: List[float], arg2: List[float]) -> Tuple[List[float], int]
00826     *
00827     * @par Lua 函数原型
00828     * inverseKinematics1(qnear: table, pose: table, tcp_offset: table) -> table, int
00829     *
00830     * @par Lua 示例
00831     * joint,ik_result = inverseKinematics1({0,0,0,0,0,0},{0.81665,-0.20419,0.43873,-3.135,0.004,1.569},
00832     {0.04,-0.035,0.1,0,0,0})
00833     *
00834     * @par JSON-RPC 请求示例
00835     * {"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.inverseKinematics1","params":[[0,0,0,0,0,0],
00836     * [0.71374,0.08417,0.6708,2.46,0.479,1.619],[0.0,
00837     * 0.13201,0.03879,0,0,0]],"id":1}
00838     *
00839     * @par JSON-RPC 响应示例
00840     * {"id":1,"jsonrpc":"2.0","result":
00841     * [[3.084454549595208,1.2278265883747776,1.0986586440159576,0.6708221281915528,-1.8712459848518375,-0.3965111476861782],0]}
00842     * \endchinese
00843     * \english
00844     * Inverse kinematics
00845     * Input TCP pose and reference joint angles, output joint angles
00846     *
00847     * @param qnear Reference joint angles

```



```

00844     * @param pose TCP pose
00845     * @param tcp_offset TCP offset
00846     * @return Joint angles and whether the result is valid, same as inverseKinematics
00847     * @throws arcs::common_interface::AuboException
00848     *
00849     * @par Python function prototype
00850     * inverseKinematics1(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float],
00851     * arg1: List[float], arg2: List[float]) -> Tuple[List[float], int]
00852     *
00853     * @par Lua function prototype
00854     * inverseKinematics1(qnear: table, pose: table, tcp_offset: table) -> table, int
00855     *
00856     * @par Lua example
00857     * joint,ik_result = inverseKinematics1({0,0,0,0,0},{0.81665,-0.20419,0.43873,-3.135,0.004,1.569},
{0.04,-0.035,0.1,0,0,0})
00858     *
00859     * @par JSON-RPC request example
00860     * {"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.inverseKinematics1","params":[[0,0,0,0,0],
[0.71374,0.08417,0.6708,2.46,0.479,1.619],[0,0,
0.13201,0.03879,0,0,0]],"id":1}
00861     *
00862     *
00863     * @par JSON-RPC response example
00864     * {"id":1,"jsonrpc":"2.0","result":
[[3.084454549595208,1.2278265883747776,1.0986586440159576,0.6708221281915528,-1.8712459848518375,-0.3965111476861782],0]}
00865     * \endenglish
00866     */
00867     ResultWithErrno inverseKinematics1(const std::vector<double> &qnear,
00868     const std::vector<double> &pose,
00869     const std::vector<double> &tcp_offset);
00870
00871     /**
00872     * @ingroup RobotAlgorithm
00873     * \chinese
00874     * 同构型最小关节变化逆解
00875     * 按照 qnear 的机械臂构型计算目标 TCP 位姿逆解,并在同构型逆解中选择相对 qnear 关节变化最小的一组
00876     * 基于激活的 TCP 偏移 (最近的通过 setTcpOffset 设置的参数)
00877     *
00878     * @param qnear 参考关节角
00879     * @param pose TCP 位姿,基于激活的 TCP 偏移 (最近的通过 setTcpOffset 设置的参数)
00880     * @return 关节角和逆解结果是否有效
00881     * 返回值的第一个参数为逆解结果,第二个为逆解错误码,错误码返回列表如下
00882     * 0 - 成功
00883     * -1 - 机械臂状态不对 (未初始化完成,可尝试再次调用)
00884     * -5 - 输入的参数关节角或 TCP 位姿无效 (维度错误)
00885     * -23 - 逆解计算不收敛,计算出错
00886     * -24 - 逆解计算超出机器人最大限制
00887     * -25 - 逆解输入配置存在错误
00888     * -26 - 逆解雅可比矩阵计算失败
00889     * -27 - 目标点存在解析解,但均不满足同构型最小关节变化逆解条件
00890     * -28 - 逆解返回未知类型错误
00891     * 若错误码非 0,则返回值的第一个参数为输入参考关节角 qnear
00892     * @throws arcs::common_interface::AuboException
00893     *
00894     * @par Python 函数原型
00895     * inverseKinematics2(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float],
00896     * arg1: List[float]) -> Tuple[List[float], int]
00897     *
00898     * @par Lua 函数原型
00899     * inverseKinematics2(qnear: table, pose: table) -> table, int
00900     *
00901     * @par Lua 示例
00902     * joint,ik_result = inverseKinematics2({0,0,0,0,0},{0.81665,-0.20419,0.43873,-3.135,0.004,1.569})
00903     *
00904     * @par JSON-RPC 请求示例
00905     * {"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.inverseKinematics2","params":[[0,0,0,0,0],
[0.71374,0.08417,0.6708,2.46,0.479,1.619]],"id":1}
00906     *
00907     * @par JSON-RPC 响应示例
00908     * {"id":1,"jsonrpc":"2.0","result":
[[3.083688522170976,1.2273215976885394,1.098072739631141,0.6705738810610149,-1.870715392248607,-0.39708546603119627],0]}
00909     * \endchinese
00910     * \english
00911     * Inverse kinematics with the same robot configuration and minimum joint change
00912     * Calculate the inverse kinematics solution for the target TCP pose according to the robot configuration of
qnear, and select the solution with the minimum joint change relative to qnear among solutions with the same
configuration.
00913     * This interface is based on the activated TCP offset, which is the latest parameter set by setTcpOffset.
00914     *
00915     * @param qnear Reference joint angles
00916     * @param pose TCP pose, based on the activated TCP offset, which is the latest parameter set by setTcpOffset
00917     * @return Joint angles and whether the result is valid
00918     * The first parameter of the return value is the inverse kinematics result, the second is the error code. Error
codes are as follows:
00919     * 0 - Success
00920     * -1 - The status of the robot is incorrect (not initialized yet, you can try calling it again)
00921     * -5 - The input reference joint angles or tcp pose is invalid (dimension error)
00922     * -23 - Inverse kinematics calculation does not converge, calculation error

```

```

00923     * -24 - Inverse kinematics calculation exceeds robot limits
00924     * -25 - Inverse kinematics input configuration error
00925     * -26 - Inverse kinematics Jacobian calculation failed
00926     * -27 - Analytical solution exists but none satisfy the same-configuration minimum-joint-change selection
criteria
00927     * -28 - Unknown error in inverse kinematics
00928     * If the error code is not 0, the first parameter of the return value is the input reference joint angles qnear
00929     * @throws arcs::common_interface::AuboException
00930     *
00931     * @par Python function prototype
00932     * inverseKinematics2(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float],
00933     * arg1: List[float]) -> Tuple[List[float], int]
00934     *
00935     * @par Lua function prototype
00936     * inverseKinematics2(qnear: table, pose: table) -> table, int
00937     *
00938     * @par Lua example
00939     * joint,ik_result = inverseKinematics2({0,0,0,0,0},{0.81665,-0.20419,0.43873,-3.135,0.004,1.569})
00940     *
00941     * @par JSON-RPC request example
00942     * {"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.inverseKinematics2","params":[[0,0,0,0,0],
[0.71374,0.08417,0.6708,2.46,0.479,1.619]],"id":1}
00943     *
00944     * @par JSON-RPC response example
00945     * {"id":1,"jsonrpc":"2.0","result":
[[3.083688522170976,1.2273215976885394,1.098072739631141,0.6705738810610149,-1.870715392248607,-0.39708546603119627],0]}
00946     * \endenglish
00947     */
00948     ResultWithErrno inverseKinematics2(const std::vector<double> &qnear,
00949                                       const std::vector<double> &pose);
00950
00951     /**
00952     * @ingroup RobotAlgorithm
00953     * \chinese
00954     * 求出所有的逆解，基于激活的 TCP 偏移
00955     *
00956     * @param pose TCP 位姿
00957     * @return 关节角和逆解结果是否有效
00958     * 返回的错误码同 inverseKinematics
00959     *
00960     * @throws arcs::common_interface::AuboException
00961     *
00962     * @par JSON-RPC 请求示例
00963     * {"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.inverseKinematicsAll","params":
[[0.71374,0.08417,0.6708,2.46,0.479,1.619]],"id":1}
00964     *
00965     * @par JSON-RPC 响应示例
00966     * {"id":1,"jsonrpc":"2.0","result":
[[[3.083688522170976,1.2273215976885394,1.098072739631141,0.6705738810610149,-1.870715392248607,-0.39708546603119627],
00967     *
[3.081056801097411,0.17985038037652645,-1.0991717292664145,-0.4806460200109001,-1.869182975312333,-0.402066016835411],
00968     *
[0.4090095277807992,-0.1623365054641728,1.081775890307679,0.26993250263224805,0.9738255833642309,0.000572556627720845],
00969     *
[0.4116449425067969,-1.1931664523907126,-1.0822709833775688,-0.8665964106161371,0.9732141569888207,0.006484919654891586]],0]}
00970     * \endchinese
00971     * \english
00972     * Solve all inverse kinematics solutions based on the activated TCP offset
00973     *
00974     * @param pose TCP pose
00975     * @return Joint angles and whether the result is valid
00976     * The returned error code is the same as inverseKinematics
00977     *
00978     * @throws arcs::common_interface::AuboException
00979     *
00980     * @par JSON-RPC request example
00981     * {"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.inverseKinematicsAll","params":
[[0.71374,0.08417,0.6708,2.46,0.479,1.619]],"id":1}
00982     *
00983     * @par JSON-RPC response example
00984     * {"id":1,"jsonrpc":"2.0","result":
[[[3.083688522170976,1.2273215976885394,1.098072739631141,0.6705738810610149,-1.870715392248607,-0.39708546603119627],
00985     *
[3.081056801097411,0.17985038037652645,-1.0991717292664145,-0.4806460200109001,-1.869182975312333,-0.402066016835411],
00986     *
[0.4090095277807992,-0.1623365054641728,1.081775890307679,0.26993250263224805,0.9738255833642309,0.000572556627720845],
00987     *
[0.4116449425067969,-1.1931664523907126,-1.0822709833775688,-0.8665964106161371,0.9732141569888207,0.006484919654891586]],0]}
00988     * \endenglish
00989     */
00990     ResultWithErrno1 inverseKinematicsAll(const std::vector<double> &pose);
00991
00992     /**
00993     * @ingroup RobotAlgorithm
00994     * \chinese
00995     * 求出所有的逆解，基于提供的 TCP 偏移
00996     *

```

```

00997     * @param pose TCP 位姿
00998     * @param tcp_offset TCP 偏移
00999     * @return 关节角和逆解结果是否有效, 同 inverseKinematicsAll
01000     * 返回的错误码同 inverseKinematics
01001     *
01002     * @throws arcs::common_interface::AuboException
01003     *
01004     * @par JSON-RPC 请求示例
01005     * {"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.inverseKinematicsAll1", "params":
[[0.71374, 0.08417, 0.6708, 2.46, 0.479, 1.619], [0.0,
01006     * 0.13201, 0.03879, 0, 0, 0]], "id": 1}
01007     *
01008     * @par JSON-RPC 响应示例
01009     * {"id": 1, "jsonrpc": "2.0", "result":
[[[3.084454549595208, 1.2278265883747776, 1.0986586440159576, 0.6708221281915528, -1.8712459848518375, -0.3965111476861782],
01010     *
[3.0818224058231602, 0.17980369843203092, -1.0997576631122077, -0.48102131527371267, -1.8697135490338517, -0.40149459722060593],
01011     *
[0.40972960018231047, -0.16226026285489026, 1.0823403816496, 0.2700204411869427, 0.9734251963887868, 0.0012903686498106507],
01012     *
[0.41236549588802296, -1.193621392918341, -1.0828346680836718, -0.8671097369314354, 0.972815367289568, 0.007206851371073478]], 0]], 0}
01013     * \endchinese
01014     * \english
01015     * Solve all inverse kinematics solutions based on the provided TCP offset
01016     *
01017     * @param pose TCP pose
01018     * @param tcp_offset TCP offset
01019     * @return Joint angles and whether the result is valid, same as inverseKinematicsAll
01020     * The returned error code is the same as inverseKinematics
01021     *
01022     * @throws arcs::common_interface::AuboException
01023     *
01024     * @par JSON-RPC request example
01025     * {"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.inverseKinematicsAll1", "params":
[[0.71374, 0.08417, 0.6708, 2.46, 0.479, 1.619], [0.0,
01026     * 0.13201, 0.03879, 0, 0, 0]], "id": 1}
01027     *
01028     * @par JSON-RPC response example
01029     * {"id": 1, "jsonrpc": "2.0", "result":
[[[3.084454549595208, 1.2278265883747776, 1.0986586440159576, 0.6708221281915528, -1.8712459848518375, -0.3965111476861782],
01030     *
[3.0818224058231602, 0.17980369843203092, -1.0997576631122077, -0.48102131527371267, -1.8697135490338517, -0.40149459722060593],
01031     *
[0.40972960018231047, -0.16226026285489026, 1.0823403816496, 0.2700204411869427, 0.9734251963887868, 0.0012903686498106507],
01032     *
[0.41236549588802296, -1.193621392918341, -1.0828346680836718, -0.8671097369314354, 0.972815367289568, 0.007206851371073478]], 0]], 0}
01033     * \endenglish
01034     */
01035     ResultWithErrno1 inverseKinematicsAll1(
01036         const std::vector<double> &pose, const std::vector<double> &tcp_offset);
01037
01038     /**
01039     * @ingroup RobotAlgorithm
01040     * \chinese
01041     * 运动学逆解 (忽略 TCP 偏移值)
01042     *
01043     * @param qnear 参考关节角
01044     * @param pose 法兰盘中心的位姿
01045     * @return 关节角和逆解结果是否有效
01046     * 返回值的第一个参数为逆解结果, 第二个为逆解错误码, 错误码返回列表如下
01047     * 0 - 成功
01048     * -1 - 机械臂状态不对 (未初始化完成, 可尝试再次调用)
01049     * -5 - 输入的参考关节角或位姿无效 (维度错误)
01050     *
01051     * @throws arcs::common_interface::AuboException
01052     *
01053     * @par Lua 函数原型
01054     * inverseToolKinematics(qnear: table, pose: table) -> table, int
01055     *
01056     * @par Lua 示例
01057     * joint, ik_result = inverseToolKinematics({0,0,0,0,0,0},{0.58815,0.0532,0.62391,2.46,0.479,1.619})
01058     *
01059     * @par JSON-RPC 请求示例
01060     * {"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.inverseToolKinematics", "params": [[0,0,0,0,0,0],
[[0.58815, 0.0532, 0.62391, 2.46, 0.479, 1.619]], "id": 1}
01061     *
01062     * @par JSON-RPC 响应示例
01063     * {"id": 1, "jsonrpc": "2.0", "result":
[[[3.083609363838651, 1.22736129158332, 1.098095443698268, 0.6705395395487186, -1.8706605026855632, -0.39714507002376465], 0]], 0}
01064     * \endchinese
01065     * \english
01066     * Inverse kinematics (ignoring TCP offset)
01067     *
01068     * @param qnear Reference joint angles
01069     * @param pose Flange center pose
01070     * @return Joint angles and whether the result is valid
01071     * The first parameter of the return value is the inverse kinematics result, the second is the error code. Error

```

```

codes are as follows:
01072     * 0 - Success
01073     * -1 - The status of the robot is incorrect (not initialized yet, you can try calling it again)
01074     * -5 - The input reference joint angles or pose is invalid (dimension error)
01075     * @throws arcs::common_interface::AuboException
01076     *
01077     * @par Lua function prototype
01078     * inverseToolKinematics(qnear: table, pose: table) -> table, int
01079     *
01080     * @par Lua example
01081     * joint, ik_result = inverseToolKinematics({0,0,0,0,0,0},{0.58815,0.0532,0.62391,2.46,0.479,1.619})
01082     *
01083     * @par JSON-RPC request example
01084     * {"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.inverseToolKinematics","params":[[0,0,0,0,0,0],
01085     * [0.58815,0.0532,0.62391,2.46,0.479,1.619]],"id":1}
01086     *
01087     * @par JSON-RPC response example
01088     * {"id":1,"jsonrpc":"2.0","result":
01089     * [[3.083609363838651,1.22736129158332,1.098095443698268,0.6705395395487186,-1.8706605026855632,-0.39714507002376465],0]}
01090     * \endenglish
01091     */
01092     ResultWithErrno inverseToolKinematics(const std::vector<double> &qnear,
01093     const std::vector<double> &pose);
01094
01095 /**
01096  * @ingroup RobotAlgorithm
01097  * \chinese
01098  * 运动学逆解 (忽略 TCP 偏移值)
01099  *
01100  * @param qnear 参考关节角
01101  * @param pose 法兰盘中心的位置
01102  * @return 关节角和逆解结果是否有效
01103  *
01104  * @throws arcs::common_interface::AuboException
01105  *
01106  * @par JSON-RPC 请求示例
01107  * {"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.inverseToolKinematicsAll","params":
01108  * [[0.58815,0.0532,0.62391,2.46,0.479,1.619]],"id":1}
01109  *
01110  * @par JSON-RPC 响应示例
01111  * {"id":1,"jsonrpc":"2.0","result":
01112  * [[[3.083609363838651,1.22736129158332,1.098095443698268,0.6705395395487186,-1.8706605026855632,-0.39714507002376465],
01113  * [3.0809781797426523,0.17987122696706134,-1.0991932793263717,-0.4807053707530958,-1.8691282890274434,-0.40212516672751814],
01114  * [0.40892195618737215,-0.16235398607358653,1.081812753177426,0.27003586475871766,0.9738744130114284,0.00048462518316674287],
01115  * [0.41155633414333076,-1.1932173012004512,-1.082306542045813,-0.8665312056504818,0.9732632365861417,0.0063958311601771175]],0]}
01116  * \endchinese
01117  * \english
01118  * Inverse kinematics (ignoring TCP offset)
01119  *
01120  * @param qnear Reference joint angles
01121  * @param pose Flange center pose
01122  * @return Joint angles and whether the result is valid
01123  *
01124  * @throws arcs::common_interface::AuboException
01125  *
01126  * @par JSON-RPC request example
01127  * {"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.inverseToolKinematicsAll","params":
01128  * [[0.58815,0.0532,0.62391,2.46,0.479,1.619]],"id":1}
01129  *
01130  * @par JSON-RPC response example
01131  * {"id":1,"jsonrpc":"2.0","result":
01132  * [[[3.083609363838651,1.22736129158332,1.098095443698268,0.6705395395487186,-1.8706605026855632,-0.39714507002376465],
01133  * [3.0809781797426523,0.17987122696706134,-1.0991932793263717,-0.4807053707530958,-1.8691282890274434,-0.40212516672751814],
01134  * [0.40892195618737215,-0.16235398607358653,1.081812753177426,0.27003586475871766,0.9738744130114284,0.00048462518316674287],
01135  * [0.41155633414333076,-1.1932173012004512,-1.082306542045813,-0.8665312056504818,0.9732632365861417,0.0063958311601771175]],0]}
01136  * \endenglish
01137  */
01138     ResultWithErrno1 inverseToolKinematicsAll(const std::vector<double> &pose);
01139
01140 /**
01141  * @ingroup RobotAlgorithm
01142  * \chinese
01143  * 根据输入的关节角计算并返回对应的机械臂构型
01144  *
01145  * 机械臂构型由三个维度的状态组合而成, 各维度定义如下:
01146  * - 肩部方向: LEFT(肩部朝左) / RIGHT(肩部朝右)
01147  * - 肘部方向: UP(肘部朝上) / DOWN(肘部朝下)
01148  * - 腕部状态: FLIP(腕部翻转) / NOFLIP(腕部不翻转)
01149  *
01150  * 三个维度两两组合共形成 8 种基础构型 (L/R + U/D + F/N), 该接口会根据输入的关节角解析出当前对应的构型类型,
01151  * 并返回其枚举值对应的整数形式及错误码 (注: 接口返回的是组合构型值, 如 LUF 对应 0, 而非单独的 LEFT/UP/FLIP 枚举值)。

```

```

01146 *
01147 * @param q 输入的关节角数组, 6 轴机械臂为 6 个元素, 单位: 弧度 (rad)
01148 * @return 机械臂构型结果及错误码 (类型为 ResultWithErrno3, 即 std::tuple<int, int>):
01149 *     - 第一个 int: 机械臂构型枚举 (RobotConfiguration) 对应的整数值, 取值范围及含义:
01150 *         -1 (NONE)    - 无效构型
01151 *         0 (LUF)      - LEFT+UP+FLIP (左肩、肘上、腕翻转)
01152 *         1 (LUN)      - LEFT+UP+NOFLIP (左肩、肘上、腕不翻转)
01153 *         2 (LDF)      - LEFT+DOWN+FLIP (左肩、肘下、腕翻转)
01154 *         3 (LDN)      - LEFT+DOWN+NOFLIP (左肩、肘下、腕不翻转)
01155 *         4 (RUF)      - RIGHT+UP+FLIP (右肩、肘上、腕翻转)
01156 *         5 (RUN)      - RIGHT+UP+NOFLIP (右肩、肘上、腕不翻转)
01157 *         6 (RDF)      - RIGHT+DOWN+FLIP (右肩、肘下、腕翻转)
01158 *         7 (RDN)      - RIGHT+DOWN+NOFLIP (右肩、肘下、腕不翻转)
01159 *     - 第二个 int: 错误码, 错误码含义如下:
01160 *         0 - 成功: 构型计算完成, 返回有效构型值
01161 *         -1 - 机械臂状态异常: 未初始化完成, 可尝试重新初始化后调用
01162 *         -5 - 输入参数无效: 关节角数组维度错误 (非 6 个元素) 或数值超出合理范围
01163 *
01164 * @throws arcs::common_interface::AuboException 输入参数非法 (如空数组、元素数量错误) 时抛出
01165 *
01166 * @par Python 函数原型
01167 * getRobotConfiguration(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float]) -> Tuple[int, int]
01168 *
01169 * @par Lua 函数原型
01170 * getRobotConfiguration(q: table) -> number, number
01171 *
01172 * @par Lua 示例
01173 * -- 输入 6 轴关节角 (单位: rad), 获取构型及错误码
01174 * local joint_angles = {3.083,1.227,1.098,0.670,-1.870,-0.397}
01175 * local config_val, err_code = getRobotConfiguration(joint_angles)
01176 * if err_code == 0 then
01177 *     print(" 机械臂构型值: ", config_val) -- 示例输出: 4 (对应 RUF, 右肩、肘上、腕翻转)
01178 * else
01179 *     print(" 获取构型失败, 错误码: ", err_code)
01180 * end
01181 *
01182 * @par JSON-RPC 请求示例
01183 * {"jsonrpc":"2.0","method":"robl.RobotAlgorithm.getRobotConfiguration","params":
[[3.083688522170976,1.2273215976885394,1.098072739631141,0.6705738810610149,-1.870715392248607,-0.39708546603119627]],"id":1}
01184 *
01185 * @par JSON-RPC 响应示例
01186 * {"id":1,"jsonrpc":"2.0","result":[4,0]}
01187 * \endchinese
01188 * \english
01189 * Calculate and return the corresponding robot configuration based on input joint angles
01190 *
01191 * The robot configuration consists of three dimensions of states, with the following definitions for each
dimension:
01192 * - Shoulder direction: LEFT (Shoulder to the left) / RIGHT (Shoulder to the right)
01193 * - Elbow direction: UP (Elbow up) / DOWN (Elbow down)
01194 * - Wrist state: FLIP (Wrist flipped) / NOFLIP (Wrist not flipped)
01195 *
01196 * The combination of the three dimensions forms 8 basic configurations (L/R + U/D + F/N). This interface parses
the current
01197 * configuration type from the input joint angles and returns its integer value corresponding to the
RobotConfiguration enumeration
01198 * and an error code (Note: The interface returns combined configuration values, e.g., LUF corresponds to 0, not
the individual
01199 * enumeration values of LEFT/UP/FLIP).
01200 *
01201 * @param q Input joint angle array, 6 elements for 6-axis robot, unit: radians (rad)
01202 * @return Robot configuration result and error code (type: ResultWithErrno3, i.e., std::tuple<int, int>):
01203 *     - First int: Integer value corresponding to the RobotConfiguration enumeration, value range and
meanings:
01204 *         -1 (NONE)    - Invalid configuration
01205 *         0 (LUF)      - LEFT+UP+FLIP (Left shoulder, elbow up, wrist flipped)
01206 *         1 (LUN)      - LEFT+UP+NOFLIP (Left shoulder, elbow up, wrist not flipped)
01207 *         2 (LDF)      - LEFT+DOWN+FLIP (Left shoulder, elbow down, wrist flipped)
01208 *         3 (LDN)      - LEFT+DOWN+NOFLIP (Left shoulder, elbow down, wrist not flipped)
01209 *         4 (RUF)      - RIGHT+UP+FLIP (Right shoulder, elbow up, wrist flipped)
01210 *         5 (RUN)      - RIGHT+UP+NOFLIP (Right shoulder, elbow up, wrist not flipped)
01211 *         6 (RDF)      - RIGHT+DOWN+FLIP (Right shoulder, elbow down, wrist flipped)
01212 *         7 (RDN)      - RIGHT+DOWN+NOFLIP (Right shoulder, elbow down, wrist not flipped)
01213 *     - Second int: Error code with the following meanings:
01214 *         0 - Success: Configuration calculation completed, valid configuration value returned
01215 *         -1 - Abnormal robot state: Not initialized completely, try reinitializing before calling
01216 *         -5 - Invalid input parameters: Incorrect dimension of joint angle array (not 6 elements) or values
out of reasonable range
01217 *
01218 * @throws arcs::common_interface::AuboException Thrown when input parameters are illegal (e.g., empty array,
wrong number of elements)
01219 *
01220 * @par Python function prototype
01221 * getRobotConfiguration(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float]) -> Tuple[int, int]
01222 *
01223 * @par Lua function prototype
01224 * getRobotConfiguration(q: table) -> number, number

```

```

01225      *
01226      * @par Lua example
01227      * -- Input 6-axis joint angles (unit: rad) to get configuration and error code
01228      * local joint_angles = {3.083,1.227,1.098,0.670,-1.870,-0.397}
01229      * local config_val, err_code = getRobotConfiguration(joint_angles)
01230      * if err_code == 0 then
01231      *     print("Robot configuration value: ", config_val) -- Example output: 4 (corresponding to RUF, Right
shoulder, elbow up, wrist flipped)
01232      * else
01233      *     print("Failed to get configuration, error code: ", err_code)
01234      * end
01235      *
01236      * @par JSON-RPC request example
01237      * {"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.getRobotConfiguration","params":
[[3.083688522170976,1.2273215976885394,1.098072739631141,0.6705738810610149,-1.870715392248607,-0.39708546603119627]], "id":1}
01238      *
01239      * @par JSON-RPC response example
01240      * {"id":1,"jsonrpc":"2.0","result":[4,0]}
01241      * \endenglish
01242      */
01243      ResultWithErrno3 getRobotConfiguration(const std::vector<double>& q);
01244
01245      /**
01246      * \chinese
01247      * 求解 movej 之间的轨迹点
01248      *
01249      * @param q1 movej 的起点
01250      * @param r1 在 q1 处的交融半径
01251      * @param q2 movej 的终点
01252      * @param r2 在 q2 处的交融半径
01253      * @param d 采样距离
01254      * @return q1-q2 之间笛卡尔空间离散轨迹点 (x,y,z,rx,ry,rz) 集合
01255      *
01256      * @throws arcs::common_interface::AuboException
01257      *
01258      * @par Python 函数原型
01259      * pathMovej(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float], arg1:
01260      * float, arg2: List[float], arg3: float, arg4: float) -> List[List[float]]
01261      *
01262      * @par Lua 函数原型
01263      * pathMovej(q1: table, r1: number, q2: table, r2: number, d: number) ->
01264      * table, number
01265      *
01266      * @par Lua 示例
01267      * path , num = pathMovej({0.0,-0.2618,1.7453,0.4364,1.5711,0.0},0.25,
{0.3234,-0.5405,1.5403,0.5881,1.2962,0.7435},0.03,0.2)
01268      *
01269      * \endchinese
01270      * \english
01271      * Solve the trajectory points between movej
01272      *
01273      * @param q1 Start point of movej
01274      * @param r1 Blend radius at q1
01275      * @param q2 End point of movej
01276      * @param r2 Blend radius at q2
01277      * @param d Sampling distance
01278      * @return Discrete trajectory points (x, y, z, rx, ry, rz) between q1 and q2 in Cartesian space
01279      *
01280      * @throws arcs::common_interface::AuboException
01281      *
01282      * @par Python function prototype
01283      * pathMovej(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float], arg1:
01284      * float, arg2: List[float], arg3: float, arg4: float) -> List[List[float]]
01285      *
01286      * @par Lua function prototype
01287      * pathMovej(q1: table, r1: number, q2: table, r2: number, d: number) ->
01288      * table, number
01289      *
01290      * @par Lua example
01291      * path , num = pathMovej({0.0,-0.2618,1.7453,0.4364,1.5711,0.0},0.25,
{0.3234,-0.5405,1.5403,0.5881,1.2962,0.7435},0.03,0.2)
01292      *
01293      * \endenglish
01294      */
01295      std::vector<std::vector<double>> pathMovej(const std::vector<double> &q1,
01296      double r1,
01297      const std::vector<double> &q2,
01298      double r2, double d);
01299      /**
01300      * @ingroup RobotAlgorithm
01301      * \chinese
01302      * 计算机机械臂末端的雅克比矩阵
01303      *
01304      * @param q 关节角
01305      * @param base_or_end 参考坐标系为基坐标系 (或者末端坐标系)
01306      * true: 在 base 下描述
01307      * false: 在 末端坐标系 下描述

```



```

01308 * @return 雅克比矩阵是否有效
01309 * 返回值的第一个参数为该构型下对应的雅克比矩阵，第二个为逆解错误码
01310 * 此接口的错误码返回值在 0.28.1-rc.21 0.29.0-alpha.25 版本之后做了修改。
01311 * 此前逆解错误时返回 30082，修改后错误码返回列表如下
01312 * 0 - 成功
01313 * -23 - 逆解计算不收敛，计算出错
01314 * -24 - 逆解计算超出机器人最大限制
01315 * -25 - 逆解输入配置存在错误
01316 * -26 - 逆解雅可比矩阵计算失败
01317 * -27 - 目标点存在解析解，但均不满足选解条件
01318 * -28 - 逆解返回未知类型错误
01319 * 若错误码非 0，则返回值的第一个参数为输入参考关节角 qnear
01320 *
01321 * @throws arcs::common_interface::AuboException
01322 *
01323 * @par Python 函数原型
01324 * calJacobian(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float],
01325 * arg1: bool) -> Tuple[List[float], int]
01326 *
01327 * @par Lua 函数原型
01328 * calJacobian(q: table, base_or_end: boolean) -> table
01329 *
01330 * @par Lua 示例
01331 * calJ_result = calJacobian({0.58815,0.0532,0.62391,2.46,0.479,1.619},true)
01332 *
01333 * @par JSON-RPC 请求示例
01334 * {"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.calcJacobian","params":
[[0.58815,0.0532,0.62391,2.46,0.479,1.619],true],"id":1}
01335 *
01336 * @par JSON-RPC 响应示例
01337 * {"id":1,"jsonrpc":"2.0","result":
[[0.20822779551242535,-0.5409416184208162,0.2019786999613013,0.061264982268770196,-0.026269884327316487,
0.10131708699859962,0.26388933410019777,-0.36074292664199115,0.1346954733416397,0.04085636647597124,-0.07244204452918337,0.0708466286633
0.0,0.10401808481666497,-0.12571344758923886,-0.07741290545882097,0.18818543519232858,0.04628646442706299,0.0,0.5548228314607867,
-0.5548228314607868,0.5548228314607868,-0.7901273140338193,0.37230961532208007,0.0,-0.8319685244586092,0.8319685244586091,-0.83196852445
-0.5269197820578843,-0.8184088260676008,1.0,3.749399456654644e-33,-6.512048180336603e-18,1.0956823467534067e-16,-0.31313634553301894,
0.43771285536682175],0]}
01342 * \endchinese
01343 * \english
01344 * Calculate the Jacobian matrix at the robot end-effector
01345 *
01346 * @param q Joint angles
01347 * @param base_or_end Reference frame: base (or end-effector)
01348 * true: described in base frame
01349 * false: described in end-effector frame
01350 * @return Whether the Jacobian matrix is valid
01351 * The first parameter of the return value is the Jacobian matrix for this configuration, the second is the
error code.
01352 * The error code list was updated after versions 0.28.1-rc.21 and 0.29.0-alpha.25.
01353 * Previously, inverse kinematics errors returned 30082. The updated error codes are:
01354 * 0 - Success
01355 * -23 - Inverse kinematics calculation does not converge, calculation error
01356 * -24 - Inverse kinematics calculation exceeds robot limits
01357 * -25 - Inverse kinematics input configuration error
01358 * -26 - Inverse kinematics Jacobian calculation failed
01359 * -27 - Analytical solution exists but none satisfy the selection criteria
01360 * -28 - Unknown error in inverse kinematics
01361 * If the error code is not 0, the first parameter of the return value is the input reference joint angles qnear
01362 *
01363 * @throws arcs::common_interface::AuboException
01364 *
01365 *
01366 * @par Python function prototype
01367 * calJacobian(self: pyaubo_sdk.RobotAlgorithm, arg0: List[float],
01368 * arg1: bool) -> Tuple[List[float], int]
01369 *
01370 * @par Lua function prototype
01371 * calJacobian(q: table, base_or_end: boolean) -> table
01372 *
01373 * @par Lua example
01374 * calJ_result = calJacobian({0.58815,0.0532,0.62391,2.46,0.479,1.619},true)
01375 *
01376 * @par JSON-RPC request example
01377 * {"jsonrpc":"2.0","method":"rob1.RobotAlgorithm.calcJacobian","params":
[[0.58815,0.0532,0.62391,2.46,0.479,1.619],true],"id":1}
01378 *
01379 * @par JSON-RPC response example
01380 * {"id":1,"jsonrpc":"2.0","result":
[[0.20822779551242535,-0.5409416184208162,0.2019786999613013,0.061264982268770196,-0.026269884327316487,
0.10131708699859962,0.26388933410019777,-0.36074292664199115,0.1346954733416397,0.04085636647597124,-0.07244204452918337,0.0708466286633
0.0,0.10401808481666497,-0.12571344758923886,-0.07741290545882097,0.18818543519232858,0.04628646442706299,0.0,0.5548228314607867,
01381 *
01382 *
01383 *

```



```

01467     * @param name 轨迹名字
01468     * @param traj_conf 各关节轨迹的限制条件
01469     * traj_conf.move_axis: 运动的轴
01470     * 由于实际用户现场可能不希望在负载辨识时控制机械臂多关节大幅度运动, 故最好选用
    traj_conf.move_axis=LoadIdentifyMoveAxis::Joint_4_6;
01471     * traj_conf.init_joint:
01472     * 运动初始关节角, 为了避免关节 5 接近零位时的奇异问题, 应设置 traj_conf.init_joint[4] 的绝对值不小于 0.3(rad), 接近
    1.57(rad) 为宜。其余关节的关节角可任意设置
01473     * traj_conf.lower_joint_bound, traj_conf.upper_joint_bound:
01474     * 关节角上下限, 维度应与 config.move_axis 维度保持一致, 推荐设置 upper_joint_bound 为 2, lower_joint_bound 为-2
01475     * config.max_velocity, config.max_acceleration:
01476     * 关节角速度角加速度限制, 维度应与 config.move_axis 维度保持一致, 出于安全和驱动器跟随性能的考虑, 推荐设置
    max_velocity=3,max_acceleration=5
01477     *
01478     * @return 成功返回 0; 失败返回错误码
01479     * AUBO_BUSY
01480     * AUBO_BAD_STATE
01481     * -AUBO_INVL_ARGUMENT
01482     * -AUBO_BAD_STATE
01483     *
01484     * @throws arcs::common_interface::AuboException
01485     * \endchinese
01486     * \english
01487     * Generate excitation trajectory for payload identification
01488     * This interface internally calls pathBufferAppend
01489     * The offline trajectory is stored in the buffer, and can be executed later via movePathBuffer
01490     * @param name Trajectory name
01491     * @param traj_conf Joint trajectory constraints
01492     * traj_conf.move_axis: Moving axes
01493     * Since users may not want large multi-joint movements during payload identification, it is recommended to use
    traj_conf.move_axis=LoadIdentifyMoveAxis::Joint_4_6;
01494     * traj_conf.init_joint:
01495     * Initial joint angles. To avoid singularity issues near joint 5 zero position, set
    abs(traj_conf.init_joint[4]) >= 0.3(rad), preferably close to 1.57(rad). Other joints can be set arbitrarily.
01496     * traj_conf.lower_joint_bound, traj_conf.upper_joint_bound:
01497     * Joint angle limits, dimensions should match config.move_axis. Recommended: upper_joint_bound=2,
    lower_joint_bound=-2
01498     * config.max_velocity, config.max_acceleration:
01499     * Joint velocity and acceleration limits, dimensions should match config.move_axis. For safety and driver
    performance, recommended: max_velocity=3, max_acceleration=5
01500     *
01501     * @return Returns 0 on success; error code on failure
01502     * AUBO_BUSY
01503     * AUBO_BAD_STATE
01504     * -AUBO_INVL_ARGUMENT
01505     * -AUBO_BAD_STATE
01506     *
01507     * @throws arcs::common_interface::AuboException
01508     * \endenglish
01509     */
01510 int generatePayloadIdentifyTraj(const std::string &name,
01511                                const TrajConfig &traj_conf);
01512 /**
01513  * @ingroup RobotAlgorithm
01514  * \chinese
01515  * 负载辨识轨迹是否生成完成
01516  *
01517  * @return 完成返回 0; 正在进行中返回 1; 计算失败返回 <0;
01518  *
01519  * @throws arcs::common_interface::AuboException
01520  *
01521  * @par JSON-RPC 请求示例
01522  * {"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.payloadIdentifyTrajGenFinished", "params": [], "id": 1}
01523  *
01524  * @par JSON-RPC 响应示例
01525  * {"id": 1, "jsonrpc": "2.0", "result": 0}
01526  *
01527  * \endchinese
01528  * \english
01529  * Whether payload identification trajectory generation is finished
01530  *
01531  * @return 0 if finished; 1 if in progress; <0 if failed;
01532  *
01533  * @throws arcs::common_interface::AuboException
01534  *
01535  * @par JSON-RPC request example
01536  * {"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.payloadIdentifyTrajGenFinished", "params": [], "id": 1}
01537  *
01538  * @par JSON-RPC response example
01539  * {"id": 1, "jsonrpc": "2.0", "result": 0}
01540  *
01541  * \endenglish
01542  */
01543 int payloadIdentifyTrajGenFinished();
01544
01545 /**
01546  * @ingroup RobotAlgorithm

```

```

01547     * \chinese
01548     * 求解 moveS 的轨迹点
01549     *
01550     * @brief pathMoveS
01551     * @param qs 样条轨迹生成点集合
01552     * @param d 采样距离
01553     * @return
01554     *
01555     * @throws arcs::common_interface::AuboException
01556     * \endchinese
01557     * \english
01558     * Solve the trajectory points for moveS
01559     *
01560     * @brief pathMoveS
01561     * @param qs Spline trajectory generation point set
01562     * @param d Sampling distance
01563     * @return
01564     *
01565     * @throws arcs::common_interface::AuboException
01566     * \endenglish
01567     */
01568     std::vector<std::vector<double>> pathMoveS(
01569         const std::vector<std::vector<double>> &qs, double d);
01570
01571     /**
01572     * @ingroup RobotAlgorithm
01573     * \chinese
01574     * 振动抑制参数辨识算法接口
01575     *
01576     * @param q 当前关节角度
01577     * @param qd 当前关节速度
01578     * @param target_q 目标关节角度
01579     * @param target_qd 关节速度
01580     * @param target_qdd 关节加速度
01581     * @param tool_offset 工具 TCP 信息
01582     * @param omega 振动频率
01583     * @param zeta 振动阻尼比
01584     * @return 振动抑制参数和是否辨识成功
01585     *
01586     * @throws arcs::common_interface::AuboException
01587     *
01588     * @par Python 函数原型
01589     * calibVibrationParams(self: pyaubo_sdk.RobotAlgorithm, arg0:
01590     * List[List[float]], arg1: List[List[float]], arg2: List[List[float]], arg3: List[List[float]],
01591     * arg4: List[List[float]], arg5: List[float]) -> list[list[float]],int
01592     *
01593     * @par Lua 函数原型
01594     * calibVibrationParams(q: table,qd: table, target_q: table, target_qd: table,
01595     * target_qdd: table, tool_offset: table, omega: table, zeta: table) -> table,number
01596     * \endchinese
01597     */
01598     ResultWithErrno1 calibVibrationParams(const std::vector<std::vector<double>> &q,
01599         const std::vector<std::vector<double>> &qd,
01600         const std::vector<std::vector<double>> &target_q,
01601         const std::vector<std::vector<double>> &target_qd,
01602         const std::vector<std::vector<double>> &target_qdd,
01603         const std::vector<double> &tool_offset);
01604
01605     /**
01606     * @ingroup RobotAlgorithm
01607     * \chinese
01608     * 振动抑制参数辨识算法接口 1
01609     *
01610     * @param record_cache_name 目标缓存名称
01611     * @param tool_offset 工具 TCP 信息
01612     * @return 振动抑制参数和是否辨识成功
01613     *
01614     * @throws arcs::common_interface::AuboException
01615     *
01616     * @par Python 函数原型
01617     * calibVibrationParams(self: pyaubo_sdk.RobotAlgorithm, arg0:
01618     * string, arg1: List[float]) -> list[list[float]],int
01619     *
01620     * @par Lua 函数原型
01621     * calibVibrationParams(record_cache_name: string, tool_offset: table) -> table,number
01622     * \endchinese
01623     */
01624     ResultWithErrno1 calibVibrationParams1(const std::string &record_cache_name, const std::vector<double>
&tool_offset);
01625
01626     /**
01627     * @ingroup RobotAlgorithm
01628     * \chinese
01629     * 判断是否需要重新辨识振动参数
01630     *
01631     * @param param1 参考参数 (如上一次辨识结果)
01632     * @param param2 当前参数 (如新测量结果)

```

```

01633 * @param threshold 变化阈值 (0~1), 超过则需重新辨识
01634 * @return >0 需要重新辨识, =0 不需要, <0 出错
01635 *
01636 * @par Lua 函数原型
01637 * needVibrationRecalib(param1: table, param2: table, threshold: number) -> number
01638 *
01639 * @par JSON-RPC 示例
01640 * {"jsonrpc": "2.0", "method": "VibrationController.needVibrationRecalib",
01641 *  "params": [param1_obj, param2_obj, 0.1], "id": 1}
01642 * \endchinese
01643 */
01644 int needVibrationRecalib(const VibrationRecalibrationParameter &param1,
01645                         const VibrationRecalibrationParameter &param2,
01646                         double threshold);
01647
01648 /**
01649 * @ingroup RobotAlgorithm
01650 * \chinese
01651 * 添加立方体碰撞对象
01652 *
01653 * 该接口用于向当前机器人算法场景中添加一个命名的立方体碰撞对象组。
01654 *
01655 * @param name 碰撞对象唯一标识
01656 * @param link_name 碰撞对象挂载的参考 link 名称。常用名称包括:
01657 * `world`, `base_link`, `shoulder_Link`, `upperArm_Link`,
01658 * `foreArm_Link`, `wrist1_Link`, `wrist2_Link`, `wrist3_Link`,
01659 * `end_effector`
01660 * @param sizes 立方体尺寸列表。每个元素格式为
01661 * `[length, width, height]`, 单位: m
01662 * @param poses 碰撞对象位姿列表。每个元素表示相对参考 link 的位姿,
01663 * 格式为 `[x, y, z, rx, ry, rz]`
01664 * @return 0 表示成功, 非 0 表示失败
01665 *
01666 * @par Lua 函数原型
01667 * addCollisionBox(name, link_name, sizes, poses) -> integer
01668 *
01669 * @par Lua 示例
01670 * -- 添加挂载到 world 的环境碰撞体
01671 * addCollisionBox("env_box", "world",
01672 *                {{0.4, 0.4, 0.2}},
01673 *                {{0.6, 0.0, 0.1, 0.0, 0.0, 0.0}})
01674 *
01675 * -- 添加挂载到 end_effector 的末端碰撞体
01676 * addCollisionBox("tool_box", "end_effector",
01677 *                {{0.1, 0.08, 0.06}},
01678 *                {{0.0, 0.0, 0.05, 0.0, 0.0, 0.0}})
01679 *
01680 * @par Python 函数原型
01681 * addCollisionBox(name: str, link_name: str, sizes: list[list[float]],
01682 * poses: list[list[float]]) -> int
01683 *
01684 * @par Python 示例
01685 * # 添加挂载到 world 的环境碰撞体
01686 * robot_algorithm.addCollisionBox(
01687 *     "env_box", "world",
01688 *     [[0.4, 0.4, 0.2]],
01689 *     [[0.6, 0.0, 0.1, 0.0, 0.0, 0.0]]
01690 * )
01691 *
01692 * # 添加挂载到 end_effector 的末端碰撞体
01693 * robot_algorithm.addCollisionBox(
01694 *     "tool_box", "end_effector",
01695 *     [[0.1, 0.08, 0.06]],
01696 *     [[0.0, 0.0, 0.05, 0.0, 0.0, 0.0]]
01697 * )
01698 *
01699 * @par JSON-RPC 请求示例
01700 * {"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.addCollisionBox",
01701 *  "params": ["env_box", "world", [[0.4, 0.4, 0.2]],
01702 *  [[0.6, 0.0, 0.1, 0.0, 0.0, 0.0]]], "id": 1}
01703 *
01704 * @par JSON-RPC 响应示例
01705 * {"id": 1, "jsonrpc": "2.0", "result": 0}
01706 * \endchinese
01707 *
01708 * \english
01709 * Add a box collision object
01710 *
01711 * This interface adds a named group of box collision objects into the
01712 * current robot algorithm scene.
01713 *
01714 * @param name Unique identifier of the collision object
01715 * @param link_name Name of the reference link the collision object is
01716 * attached to. Common names include: `world`, `base_link`,
01717 * `shoulder_Link`, `upperArm_Link`, `foreArm_Link`, `wrist1_Link`,
01718 * `wrist2_Link`, `wrist3_Link`, and `end_effector`
01719 * @param sizes List of box sizes. Each element is in the format

```

```

01720 * `[length, width, height]`, unit: m
01721 * @param poses List of collision object poses. Each element is a pose
01722 * relative to the reference link in the format `[x, y, z, rx, ry, rz]`
01723 * @return 0 on success, non-zero on failure
01724 *
01725 * @par Lua function prototype
01726 * addCollisionBox(name, link_name, sizes, poses) -> integer
01727 *
01728 * @par Lua example
01729 * -- Add an environment collision box attached to world
01730 * addCollisionBox("env_box", "world",
01731 *               {{0.4, 0.4, 0.2}},
01732 *               {{0.6, 0.0, 0.1, 0.0, 0.0, 0.0}})
01733 *
01734 * -- Add a tool collision box attached to end_effector
01735 * addCollisionBox("tool_box", "end_effector",
01736 *               {{0.1, 0.08, 0.06}},
01737 *               {{0.0, 0.0, 0.05, 0.0, 0.0, 0.0}})
01738 *
01739 * @par Python function prototype
01740 * addCollisionBox(name: str, link_name: str, sizes: list[list[float]],
01741 * poses: list[list[float]]) -> int
01742 *
01743 * @par Python example
01744 * # Add an environment collision box attached to world
01745 * robot_algorithm.addCollisionBox(
01746 *     "env_box", "world",
01747 *     [[0.4, 0.4, 0.2]],
01748 *     [[0.6, 0.0, 0.1, 0.0, 0.0, 0.0]]
01749 * )
01750 *
01751 * # Add a tool collision box attached to end_effector
01752 * robot_algorithm.addCollisionBox(
01753 *     "tool_box", "end_effector",
01754 *     [[0.1, 0.08, 0.06]],
01755 *     [[0.0, 0.0, 0.05, 0.0, 0.0, 0.0]]
01756 * )
01757 *
01758 * @par JSON-RPC request example
01759 * {"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.addCollisionBox",
01760 *  "params": ["env_box", "world", [[0.4, 0.4, 0.2]],
01761 *  "[[0.6, 0.0, 0.1, 0.0, 0.0, 0.0]]", "id": 1}
01762 *
01763 * @par JSON-RPC response example
01764 * {"id": 1, "jsonrpc": "2.0", "result": 0}
01765 * \endenglish
01766 */
01767 int addCollisionBox(const std::string &name,
01768                   const std::string &link_name,
01769                   const std::vector<std::vector<double>> &sizes,
01770                   const std::vector<std::vector<double>> &poses);
01771
01772 /**
01773 * @ingroup RobotAlgorithm
01774 * \chinese
01775 * 删除指定的碰撞对象
01776 *
01777 * @param name 需要删除的碰撞对象名称
01778 * @return 0 表示成功, 非 0 表示失败
01779 *
01780 * @par Lua 函数原型
01781 * removeCollisionObject(name) -> integer
01782 *
01783 * @par Lua 示例
01784 * removeCollisionObject("tool_box")
01785 *
01786 * @par Python 函数原型
01787 * removeCollisionObject(name: str) -> int
01788 *
01789 * @par Python 示例
01790 * robot_algorithm.removeCollisionObject("tool_box")
01791 *
01792 * @par JSON-RPC 请求示例
01793 * {"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.removeCollisionObject",
01794 *  "params": ["tool_box"], "id": 1}
01795 *
01796 * @par JSON-RPC 响应示例
01797 * {"id": 1, "jsonrpc": "2.0", "result": 0}
01798 * \endchinese
01799 *
01800 * \english
01801 * Remove a collision object by name
01802 *
01803 * @param name Name of the collision object to remove
01804 * @return 0 on success, non-zero on failure
01805 *
01806 * @par Lua function prototype

```

```

01807     * removeCollisionObject(name) -> integer
01808     *
01809     * @par Lua example
01810     * removeCollisionObject("tool_box")
01811     *
01812     * @par Python function prototype
01813     * removeCollisionObject(name: str) -> int
01814     *
01815     * @par Python example
01816     * robot_algorithm.removeCollisionObject("tool_box")
01817     *
01818     * @par JSON-RPC request example
01819     * {"jsonrpc": "2.0", "method": "robl.RobotAlgorithm.removeCollisionObject",
01820     *   "params": ["tool_box"], "id": 1}
01821     *
01822     * @par JSON-RPC response example
01823     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01824     * \endenglish
01825     */
01826 int removeCollisionObject(const std::string &name);
01827
01828 /**
01829  * @ingroup RobotAlgorithm
01830  * \chinese
01831  * 验证机器人运动路径从起点到终点的可达性，不启用外部碰撞检测。
01832  *
01833  * 该接口保留用于兼容旧版 SDK、Lua、JSON-RPC 和 ROS 服务调用。
01834  *
01835  * @param type 路径类型标识，用于指定起点和终点的数据类型：
01836  *   0 - 起点：关节角，终点：关节角
01837  *   1 - 起点：位姿，终点：位姿
01838  * @param start 路径起点数据
01839  * @param r1 起点交融半径，单位：m（米）
01840  * @param end 路径终点数据
01841  * @param r2 终点交融半径，单位：m（米）
01842  * @param d 路径采样间隔，单位：m（米），间隔越小验证精度越高，但耗时越长
01843  * @return 路径可达性结果码
01844  *
01845  * \endchinese
01846  *
01847  * \english
01848  * Validate the reachability of the robot's motion path from start point to
01849  * end point without external collision checks.
01850  *
01851  * This interface is kept for compatibility with existing SDK, Lua,
01852  * JSON-RPC, and ROS service calls.
01853  *
01854  * @param type Path type identifier, used to specify the data types of
01855  * start and end:
01856  *   0 - Start: Joint angles, End: Joint angles
01857  *   1 - Start: Pose, End: Pose
01858  * @param start Start point data of the path
01859  * @param r1 Start point blending radius, unit: m (meters)
01860  * @param end End point data of the path
01861  * @param r2 End point blending radius, unit: m (meters)
01862  * @param d Path sampling interval, unit: m (meters). A smaller interval
01863  * improves validation accuracy but takes more time.
01864  * @return Path reachability result code
01865  *
01866  * \endenglish
01867  */
01868 int validatePath(int type, const std::vector<double> &start, double r1,
01869                 const std::vector<double> &end, double r2, double d);
01870
01871 /**
01872  * @ingroup RobotAlgorithm
01873  * \chinese
01874  * 验证机器人运动路径从起点到终点的可达性，可选择是否启用外部碰撞检测。
01875  *
01876  * 该接口通过采样的方式验证指定路径是否存在超限、自碰撞、外部碰撞、
01877  * 奇异等不可达情况，支持关节角 <-> 关节角、位姿 <-> 位姿两种路径类型验证。
01878  *
01879  * @param type 路径类型标识，用于指定起点和终点的数据类型：
01880  *   0 - 起点：关节角，终点：关节角
01881  *   1 - 起点：位姿，终点：位姿
01882  * @param start 路径起点数据
01883  * @param r1 起点交融半径，单位：m（米）
01884  * @param end 路径终点数据
01885  * @param r2 终点交融半径，单位：m（米）
01886  * @param d 路径采样间隔，单位：m（米），间隔越小验证精度越高，但耗时越长
01887  * @param check_external_collision 是否启用外部碰撞检测：
01888  *   false - 保持原有路径可达性与自碰撞校验
01889  *   true - 在原有校验基础上增加与外部对象的碰撞检测
01890  * @return 路径可达性结果码：
01891  *   0 - 可达：路径无异常，可正常运动
01892  *   -18 - 路径中存在关节超限/笛卡尔空间超限
01893  *   -21 - 轨迹生成失败

```

```

01894      *      -22 - 路径中机器人本体发生自碰撞
01895      *      -24 - 路径中经过机器人奇异位形
01896      *      -27 - 目标点有解但超出关节限位
01897      *
01898      * @throws arcs::common_interface::AuboException
01899      *
01900      * @par Lua 函数原型
01901      * validatePath1(type, start, r1, end, r2, d, check_external_collision) -> number
01902      *
01903      * @par Lua 示例
01904      * local result =
01905      *     validatePath1(0, start_joint, 0.01, end_joint, 0.01, 0.05, true)
01906      *
01907      * @par Python 函数原型
01908      * validatePath1(type: int, start: list[float], r1: float, end: list[float],
01909      * r2: float, d: float, check_external_collision: bool) -> int
01910      *
01911      * @par Python 示例
01912      * result = validatePath1(1, start_joint, 0.01, end_pose, 0.01, 0.05, True)
01913      *
01914      * @par JSON-RPC 请求示例
01915      * {"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.validatePath1", "params": [0,
01916      * [0.0, 0.0, 1.57, 0.0, 1.57, 0.0], 0.01, [0.52, 0.0, 1.57, 0.0, 1.57, 0.0], 0.01,
01917      * 0.05, true], "id": 1}
01918      *
01919      * @par JSON-RPC 响应示例
01920      * {"id": 1, "jsonrpc": "2.0", "result": 0}
01921      * \endchinese
01922      *
01923      * \english
01924      * Validate the reachability of the robot's motion path from start point to
01925      * end point, with optional external collision checks.
01926      *
01927      * This interface verifies whether the specified path has unreachable
01928      * conditions such as out-of-limit, self-collision, collision with external
01929      * objects, singularity, etc., by sampling. It supports two path type
01930      * validations: Joint-Joint and Pose-Pose.
01931      *
01932      * @param type Path type identifier, used to specify the data types of
01933      * start and end:
01934      *     0 - Start: Joint angles, End: Joint angles
01935      *     1 - Start: Pose, End: Pose
01936      * @param start Start point data of the path
01937      * @param r1 Start point blending radius, unit: m (meters)
01938      * @param end End point data of the path
01939      * @param r2 End point blending radius, unit: m (meters)
01940      * @param d Path sampling interval, unit: m (meters). A smaller interval
01941      * improves validation accuracy but takes more time.
01942      * @param check_external_collision Whether to enable external collision
01943      * checks:
01944      *     false - Keep the original reachability and self-collision checks
01945      *     true  - Add collision checks against external objects
01946      *
01947      * @return Path reachability result code:
01948      *     0 - Reachable: Path is normal, movement is possible
01949      *    -18 - Joint/Cartesian space limits exceeded in the path
01950      *    -21 - Trajectory generation failed
01951      *    -22 - Self-collision of the robot body in the path
01952      *    -24 - Path passing through robot singular configuration
01953      *    -27 - The target point has a solution but exceeds joint limits
01954      *
01955      * @throws arcs::common_interface::AuboException Thrown when parameters are
01956      * invalid.
01957      *
01958      * @par Lua function prototype
01959      * validatePath1(type, start, r1, end, r2, d, check_external_collision) -> integer
01960      *
01961      * @par Lua example
01962      * result = validatePath1(0, start_joint, 0.01, end_joint, 0.01, 0.05, true)
01963      *
01964      * @par Python function prototype
01965      * validatePath1(type: int, start: list[float], r1: float, end: list[float],
01966      * r2: float, d: float, check_external_collision: bool) -> int
01967      *
01968      * @par Python example
01969      * result = validatePath1(1, start_joint, 0.01, end_pose, 0.01, 0.005, True)
01970      *
01971      * @par JSON-RPC request example
01972      * {"jsonrpc": "2.0", "method": "rob1.RobotAlgorithm.validatePath1", "params": [0,
01973      * [0.0, 0.0, 1.57, 0.0, 1.57, 0.0], 0.01, [0.52, 0.0, 1.57, 0.0, 1.57, 0.0], 0.01,
01974      * 0.005, true], "id": 1}
01975      *
01976      * @par JSON-RPC response example {"id": 1, "jsonrpc": "2.0", "result": 0}
01977      * \endenglish
01978      */
01979      int validatePath1(int type, const std::vector<double> &start, double r1,
01980                      const std::vector<double> &end, double r2, double d,

```

```

01981             bool check_external_collision);
01982
01983 protected:
01984     void *d_;
01985 };
01986 using RobotAlgorithmPtr = std::shared_ptr<RobotAlgorithm>;
01987
01988 } // namespace common_interface
01989 } // namespace arcs
01990
01991 #endif // AUBO_SDK_ROBOT_ALGORITHM_INTERFACE_H

```

12.28 include/aubo/robot/robot_config.h 文件参考

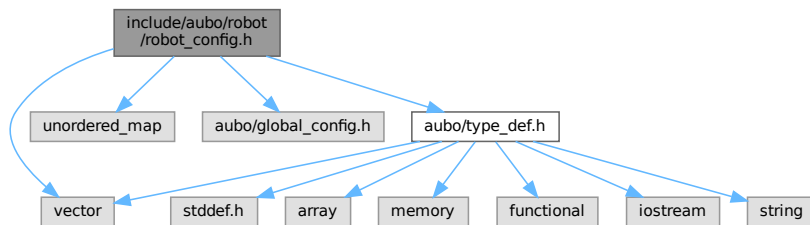
获取机器人配置接口，如获取DH 参数、碰撞等级、安装位姿等等

```

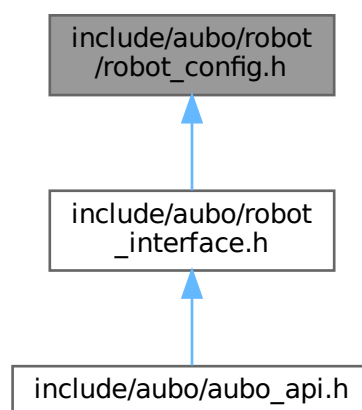
#include <vector>
#include <unordered_map>
#include <aubo/global_config.h>
#include <aubo/type_def.h>

```

robot_config.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::common_interface::RobotConfig`

命名空间

- namespace `arcs`
- namespace `arcs::common_interface`

类型定义

- using `arcs::common_interface::RobotConfigPtr` = `std::shared_ptr<RobotConfig>`

12.28.1 详细描述

获取机器人配置接口，如获取DH 参数、碰撞等级、安装位姿等等在文件 `robot_config.h` 中定义.

12.29 robot_config.h

[浏览该文件的文档.](#)

```
00001 /** @file robot_config.h
00002  * @brief 获取机器人配置接口，如获取 DH 参数、碰撞等级、安装位姿等等
00003  */
00004 #ifndef AUBO_SDK_ROBOT_CONFIG_H
00005 #define AUBO_SDK_ROBOT_CONFIG_H
00006
00007 #include <vector>
00008 #include <unordered_map>
00009
00010 #include <aubo/global_config.h>
00011 #include <aubo/type_def.h>
00012
00013 namespace arcs {
00014 namespace common_interface {
00015
00016 /**
00017  * @defgroup RobotConfig RobotConfig (机器人配置管理)
00018  * @ingroup RobotInterface
00019  * RobotConfig
00020  */
00021 class ARCS_ABI_EXPORT RobotConfig
00022 {
00023 public:
00024     RobotConfig();
00025     virtual ~RobotConfig();
00026
00027     /**
00028      * @ingroup RobotConfig
00029      * \chinese
00030      * 获取机器人的名字
00031      *
00032      * @return 返回机器人的名字
00033      *
00034      * @throws arcs::common_interface::AuboException
00035      *
00036      * @par Python 函数原型
00037      * getName(self: pyaubo_sdk.RobotConfig) -> str
00038      *
00039      * @par Lua 函数原型
00040      * getName() -> string
00041      *
00042      * @par Lua 示例
00043      * robot_name = getName()
00044      *
00045      * @par JSON-RPC 请求示例
00046      * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getName", "params": [], "id": 1}
00047      *
00048      * @par JSON-RPC 响应示例
00049      * {"id": 1, "jsonrpc": "2.0", "result": "rob1"}
00050      * \endchinese
00051      * \english
00052      * Get the robot's name.
00053      *
00054      * @return The robot's name.
00055      *
00056      * @throws arcs::common_interface::AuboException
00057      *
00058      * @par Python function prototype
00059      * getName(self: pyaubo_sdk.RobotConfig) -> str
00060      *
00061      * @par Lua function prototype
00062      * getName() -> string
```



```

00063      *
00064      * @par Lua example
00065      * robot_name = getName()
00066      *
00067      * @par JSON-RPC request example
00068      * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getName", "params": [], "id": 1}
00069      *
00070      * @par JSON-RPC response example
00071      * {"id": 1, "jsonrpc": "2.0", "result": "rob1"}
00072      * \endenglish
00073      */
00074      std::string getName();
00075
00076      /**
00077      * @ingroup RobotConfig
00078      * \chinese
00079      * 获取机器人的自由度（从硬件抽象层读取）
00080      *
00081      * @return 返回机器人的自由度
00082      *
00083      * @throws arcs::common_interface::AuboException
00084      *
00085      * @par Python 函数原型
00086      * getDof(self: pyaubo_sdk.RobotConfig) -> int
00087      *
00088      * @par Lua 函数原型
00089      * getDof() -> number
00090      *
00091      * @par Lua 示例
00092      * robot_dof = getDof()
00093      *
00094      * @par JSON-RPC 请求示例
00095      * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getDof", "params": [], "id": 1}
00096      *
00097      * @par JSON-RPC 响应示例
00098      * {"id": 1, "jsonrpc": "2.0", "result": 6}
00099      * \endchinese
00100      * \english
00101      * Get the robot's degrees of freedom (from the hardware abstraction layer).
00102      *
00103      * @return The robot's degrees of freedom.
00104      *
00105      * @throws arcs::common_interface::AuboException
00106      *
00107      * @par Python function prototype
00108      * getDof(self: pyaubo_sdk.RobotConfig) -> int
00109      *
00110      * @par Lua function prototype
00111      * getDof() -> number
00112      *
00113      * @par Lua example
00114      * robot_dof = getDof()
00115      *
00116      * @par JSON-RPC request example
00117      * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getDof", "params": [], "id": 1}
00118      *
00119      * @par JSON-RPC response example
00120      * {"id": 1, "jsonrpc": "2.0", "result": 6}
00121      * \endenglish
00122      */
00123      int getDof();
00124
00125      /**
00126      * @ingroup RobotConfig
00127      * \chinese
00128      * 获取机器人的伺服控制周期（从硬件抽象层读取）
00129      *
00130      * @return 机器人的伺服控制周期
00131      *
00132      * @throws arcs::common_interface::AuboException
00133      *
00134      * @par Python 函数原型
00135      * getCycletime(self: pyaubo_sdk.RobotConfig) -> float
00136      *
00137      * @par Lua 函数原型
00138      * getCycletime() -> number
00139      *
00140      * @par Lua 示例
00141      * robot_Cycletime = getCycletime()
00142      *
00143      * @par JSON-RPC 请求示例
00144      * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getCycletime", "params": [], "id": 1}
00145      *
00146      * @par JSON-RPC 响应示例
00147      * {"id": 1, "jsonrpc": "2.0", "result": 0.005}
00148      * \endchinese
00149      * \english

```

```

00150     * Get the robot's servo control cycle time (from the hardware abstraction
00151     * layer).
00152     *
00153     * @return The robot's servo control cycle time.
00154     *
00155     * @throws arcs::common_interface::AuboException
00156     *
00157     * @par Python function prototype
00158     * getCycleTime(self: pyaubo_sdk.RobotConfig) -> float
00159     *
00160     * @par Lua function prototype
00161     * getCycleTime() -> number
00162     *
00163     * @par Lua example
00164     * robot_CycleTime = getCycleTime()
00165     *
00166     * @par JSON-RPC request example
00167     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getCycleTime", "params": [], "id": 1}
00168     *
00169     * @par JSON-RPC response example
00170     * {"id": 1, "jsonrpc": "2.0", "result": 0.005}
00171     * \endenglish
00172     */
00173 double getCycleTime();
00174
00175 /**
00176  * @ingroup RobotConfig
00177  * \chinese
00178  * 预设缓速模式下的速度缩减比例
00179  *
00180  * @param level 缓速等级 1, 2
00181  * @param fraction
00182  *
00183  * @return 成功返回 0; 失败返回错误码
00184  * AUBO_BUSY
00185  * AUBO_BAD_STATE
00186  * AUBO_INVL_ARGUMENT
00187  * -AUBO_BAD_STATE
00188  *
00189  * @throws arcs::common_interface::AuboException
00190  *
00191  * @par JSON-RPC 请求示例
00192  * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.setSlowDownFraction", "params": [1, 0.8], "id": 1}
00193  *
00194  * @par JSON-RPC 响应示例
00195  * {"id": 1, "jsonrpc": "2.0", "result": 0}
00196  * \endchinese
00197  * \english
00198  * Set the speed reduction ratio for the preset slow-down mode.
00199  *
00200  * @param level Slow-down level 1, 2
00201  * @param fraction
00202  *
00203  * @return Returns 0 on success; error code on failure
00204  * AUBO_BUSY
00205  * AUBO_BAD_STATE
00206  * AUBO_INVL_ARGUMENT
00207  * -AUBO_BAD_STATE
00208  *
00209  * @throws arcs::common_interface::AuboException
00210  *
00211  * @par JSON-RPC request example
00212  * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.setSlowDownFraction", "params": [1, 0.8], "id": 1}
00213  *
00214  * @par JSON-RPC response example
00215  * {"id": 1, "jsonrpc": "2.0", "result": 0}
00216  * \endenglish
00217  */
00218 int setSlowDownFraction(int level, double fraction);
00219
00220 /**
00221  * @ingroup RobotConfig
00222  * \chinese
00223  * 获取预设的缓速模式下的速度缩减比例
00224  *
00225  * @param level 缓速等级 1, 2
00226  * @return 返回预设的缓速模式下的速度缩减比例
00227  *
00228  * @throws arcs::common_interface::AuboException
00229  *
00230  * @par JSON-RPC 请求示例
00231  * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getSlowDownFraction", "params": [1], "id": 1}
00232  *
00233  * @par JSON-RPC 响应示例
00234  * {"id": 1, "jsonrpc": "2.0", "result": 0.5}
00235  * \endchinese
00236  * \english

```

```

00237     * Get the speed reduction ratio for the preset slow-down mode.
00238     *
00239     * @param level Slow-down level 1, 2
00240     * @return The speed reduction ratio for the preset slow-down mode.
00241     *
00242     * @throws arcs::common_interface::AuboException
00243     *
00244     * @par JSON-RPC request example
00245     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getSlowDownFraction", "params": [1], "id": 1}
00246     *
00247     * @par JSON-RPC response example
00248     * {"id": 1, "jsonrpc": "2.0", "result": 0.5}
00249     * \endenglish
00250     */
00251 double getSlowDownFraction(int level);
00252
00253 /**
00254     * @ingroup RobotConfig
00255     * \chinese
00256     * 获取默认的工具端加速度, 单位 m/s^2
00257     *
00258     * @return 默认的工具端加速度
00259     *
00260     * @throws arcs::common_interface::AuboException
00261     *
00262     * @par Python 函数原型
00263     * getDefaultToolAcc(self: pyaubo_sdk.RobotConfig) -> float
00264     *
00265     * @par Lua 函数原型
00266     * getDefaultToolAcc() -> number
00267     *
00268     * @par Lua 示例
00269     * DefaultToolAcc = getDefaultToolAcc()
00270     *
00271     * @par JSON-RPC 请求示例
00272     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getDefaultToolAcc", "params": [], "id": 1}
00273     *
00274     * @par JSON-RPC 响应示例
00275     * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
00276     * \endchinese
00277     * \english
00278     * Get the default tool acceleration, in m/s^2.
00279     *
00280     * @return The default tool acceleration.
00281     *
00282     * @throws arcs::common_interface::AuboException
00283     *
00284     * @par Python function prototype
00285     * getDefaultToolAcc(self: pyaubo_sdk.RobotConfig) -> float
00286     *
00287     * @par Lua function prototype
00288     * getDefaultToolAcc() -> number
00289     *
00290     * @par Lua example
00291     * DefaultToolAcc = getDefaultToolAcc()
00292     *
00293     * @par JSON-RPC request example
00294     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getDefaultToolAcc", "params": [], "id": 1}
00295     *
00296     * @par JSON-RPC response example
00297     * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
00298     * \endenglish
00299     */
00300 double getDefaultToolAcc();
00301
00302 /**
00303     * @ingroup RobotConfig
00304     * \chinese
00305     * 获取默认的工具端速度, 单位 m/s
00306     *
00307     * @return 默认的工具端速度
00308     *
00309     * @throws arcs::common_interface::AuboException
00310     *
00311     * @par Python 函数原型
00312     * getDefaultToolSpeed(self: pyaubo_sdk.RobotConfig) -> float
00313     *
00314     * @par Lua 函数原型
00315     * getDefaultToolSpeed() -> number
00316     *
00317     * @par Lua 示例
00318     * DefaultToolSpeed = getDefaultToolSpeed()
00319     *
00320     * @par JSON-RPC 请求示例
00321     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getDefaultToolSpeed", "params": [], "id": 1}
00322     *
00323     * @par JSON-RPC 响应示例

```

```

00324     * {"id":1,"jsonrpc":"2.0","result":0.0}
00325     * \endchinese
00326     * \english
00327     * Get the default tool speed, in m/s.
00328     *
00329     * @return The default tool speed.
00330     *
00331     * @throws arcs::common_interface::AuboException
00332     *
00333     * @par Python function prototype
00334     * getDefaultToolSpeed(self: pyaubo_sdk.RobotConfig) -> float
00335     *
00336     * @par Lua function prototype
00337     * getDefaultToolSpeed() -> number
00338     *
00339     * @par Lua example
00340     * DefaultToolSpeed = getDefaultToolSpeed()
00341     *
00342     * @par JSON-RPC request example
00343     * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getDefaultToolSpeed","params":[],"id":1}
00344     *
00345     * @par JSON-RPC response example
00346     * {"id":1,"jsonrpc":"2.0","result":0.0}
00347     * \endenglish
00348     */
00349 double getDefaultToolSpeed();
00350
00351 /**
00352     * @ingroup RobotConfig
00353     * \chinese
00354     * 获取默认关节加速度, 单位 rad/s^2
00355     *
00356     * @return 默认关节加速度
00357     *
00358     * @throws arcs::common_interface::AuboException
00359     *
00360     * @par Python 函数原型
00361     * getDefaultJointAcc(self: pyaubo_sdk.RobotConfig) -> float
00362     *
00363     * @par Lua 函数原型
00364     * getDefaultJointAcc() -> number
00365     *
00366     * @par Lua 示例
00367     * DefaultJointAcc = getDefaultJointAcc()
00368     *
00369     * @par JSON-RPC 请求示例
00370     * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getDefaultJointAcc","params":[],"id":1}
00371     *
00372     * @par JSON-RPC 响应示例
00373     * {"id":1,"jsonrpc":"2.0","result":0.0}
00374     * \endchinese
00375     * \english
00376     * Get the default joint acceleration, in rad/s^2.
00377     *
00378     * @return The default joint acceleration.
00379     *
00380     * @throws arcs::common_interface::AuboException
00381     *
00382     * @par Python function prototype
00383     * getDefaultJointAcc(self: pyaubo_sdk.RobotConfig) -> float
00384     *
00385     * @par Lua function prototype
00386     * getDefaultJointAcc() -> number
00387     *
00388     * @par Lua example
00389     * DefaultJointAcc = getDefaultJointAcc()
00390     *
00391     * @par JSON-RPC request example
00392     * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getDefaultJointAcc","params":[],"id":1}
00393     *
00394     * @par JSON-RPC response example
00395     * {"id":1,"jsonrpc":"2.0","result":0.0}
00396     * \endenglish
00397     */
00398 double getDefaultJointAcc();
00399
00400 /**
00401     * @ingroup RobotConfig
00402     * \chinese
00403     * 获取默认关节速度, 单位 rad/s
00404     *
00405     * @return 默认关节速度
00406     *
00407     * @throws arcs::common_interface::AuboException
00408     *
00409     * @par Python 函数原型
00410     * getDefaultJointSpeed(self: pyaubo_sdk.RobotConfig) -> float

```

```

00411     *
00412     * @par Lua 函数原型
00413     * getDefaultJointSpeed() -> number
00414     *
00415     * @par Lua 示例
00416     * DefaultJointSpeed = getDefaultJointSpeed()
00417     *
00418     * @par JSON-RPC 请求示例
00419     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getDefaultJointSpeed", "params": [], "id": 1}
00420     *
00421     * @par JSON-RPC 响应示例
00422     * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
00423     * \endchinese
00424     * \english
00425     * Get the default joint speed, in rad/s.
00426     *
00427     * @return The default joint speed.
00428     *
00429     * @throws arcs::common_interface::AuboException
00430     *
00431     * @par Python function prototype
00432     * getDefaultJointSpeed(self: pyaubo_sdk.RobotConfig) -> float
00433     *
00434     * @par Lua function prototype
00435     * getDefaultJointSpeed() -> number
00436     *
00437     * @par Lua example
00438     * DefaultJointSpeed = getDefaultJointSpeed()
00439     *
00440     * @par JSON-RPC request example
00441     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getDefaultJointSpeed", "params": [], "id": 1}
00442     *
00443     * @par JSON-RPC response example
00444     * {"id": 1, "jsonrpc": "2.0", "result": 0.0}
00445     * \endenglish
00446     */
00447 double getDefaultJointSpeed();
00448
00449 /**
00450  * @ingroup RobotConfig
00451  * \chinese
00452  * 获取机器人类型代码
00453  *
00454  * @return 机器人类型代码
00455  *
00456  * @throws arcs::common_interface::AuboException
00457  *
00458  * @par Python 函数原型
00459  * getRobotType(self: pyaubo_sdk.RobotConfig) -> str
00460  *
00461  * @par Lua 函数原型
00462  * getRobotType() -> string
00463  *
00464  * @par Lua 示例
00465  * RobotType = getRobotType()
00466  *
00467  * @par JSON-RPC 请求示例
00468  * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getRobotType", "params": [], "id": 1}
00469  *
00470  * @par JSON-RPC 响应示例
00471  * {"id": 1, "jsonrpc": "2.0", "result": "aubo_i5H"}
00472  * \endchinese
00473  * \english
00474  * Get the robot type code.
00475  *
00476  * @return The robot type code.
00477  *
00478  * @throws arcs::common_interface::AuboException
00479  *
00480  * @par Python function prototype
00481  * getRobotType(self: pyaubo_sdk.RobotConfig) -> str
00482  *
00483  * @par Lua function prototype
00484  * getRobotType() -> string
00485  *
00486  * @par Lua example
00487  * RobotType = getRobotType()
00488  *
00489  * @par JSON-RPC request example
00490  * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getRobotType", "params": [], "id": 1}
00491  *
00492  * @par JSON-RPC response example
00493  * {"id": 1, "jsonrpc": "2.0", "result": "aubo_i5H"}
00494  * \endenglish
00495  */
00496 std::string getRobotType();
00497

```

```

00498  /**
00499  * @ingroup RobotConfig
00500  * \chinese
00501  * 获取机器人子类型代码
00502  *
00503  * @return 机器人子类型代码
00504  *
00505  * @throws arcs::common_interface::AuboException
00506  *
00507  * @par Python 函数原型
00508  * getRobotSubType(self: pyaubo_sdk.RobotConfig) -> str
00509  *
00510  * @par Lua 函数原型
00511  * getRobotSubType() -> string
00512  *
00513  * @par Lua 示例
00514  * RobotSubType = getRobotSubType()
00515  *
00516  * @par JSON-RPC 请求示例
00517  * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getRobotSubType","params":[],"id":1}
00518  *
00519  * @par JSON-RPC 响应示例
00520  * {"id":1,"jsonrpc":"2.0","result":"B0"}
00521  * \endchinese
00522  * \english
00523  * Get the robot sub-type code.
00524  *
00525  * @return The robot sub-type code.
00526  *
00527  * @throws arcs::common_interface::AuboException
00528  *
00529  * @par Python function prototype
00530  * getRobotSubType(self: pyaubo_sdk.RobotConfig) -> str
00531  *
00532  * @par Lua function prototype
00533  * getRobotSubType() -> string
00534  *
00535  * @par Lua example
00536  * RobotSubType = getRobotSubType()
00537  *
00538  * @par JSON-RPC request example
00539  * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getRobotSubType","params":[],"id":1}
00540  *
00541  * @par JSON-RPC response example
00542  * {"id":1,"jsonrpc":"2.0","result":"B0"}
00543  * \endenglish
00544  */
00545 std::string getRobotSubType();
00546
00547  /**
00548  * @ingroup RobotConfig
00549  * \chinese
00550  * 获取控制柜类型代码
00551  *
00552  * @return 控制柜类型代码
00553  *
00554  * @throws arcs::common_interface::AuboException
00555  *
00556  * @par Python 函数原型
00557  * getControlBoxType(self: pyaubo_sdk.RobotConfig) -> str
00558  *
00559  * @par Lua 函数原型
00560  * getControlBoxType() -> string
00561  *
00562  * @par Lua 示例
00563  * ControlBoxType = getControlBoxType()
00564  *
00565  * @par JSON-RPC 请求示例
00566  * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getControlBoxType","params":[],"id":1}
00567  *
00568  * @par JSON-RPC 响应示例
00569  * {"id":1,"jsonrpc":"2.0","result":"cb_ISStation"}
00570  * \endchinese
00571  * \english
00572  * Get the control box type code.
00573  *
00574  * @return The control box type code.
00575  *
00576  * @throws arcs::common_interface::AuboException
00577  *
00578  * @par Python function prototype
00579  * getControlBoxType(self: pyaubo_sdk.RobotConfig) -> str
00580  *
00581  * @par Lua function prototype
00582  * getControlBoxType() -> string
00583  *
00584  * @par Lua example

```

```

00585     * ControlBoxType = getControlBoxType()
00586     *
00587     * @par JSON-RPC request example
00588     * {"jsonrpc": "2.0", "method": "robl.RobotConfig.getControlBoxType", "params": [], "id": 1}
00589     *
00590     * @par JSON-RPC response example
00591     * {"id": 1, "jsonrpc": "2.0", "result": "cb_ISStation"}
00592     * \endenglish
00593     */
00594     std::string getControlBoxType();
00595
00596     /**
00597     * @ingroup RobotConfig
00598     * \chinese
00599     * 设置安装位姿（机器人的基坐标系相对于世界坐标系） world->base
00600     *
00601     * 一般在多机器人系统中使用，默认为 [0,0,0,0,0,0]
00602     *
00603     * @param pose 安装位姿
00604     * @return 成功返回 0；失败返回错误码
00605     * AUBO_BUSY
00606     * AUBO_BAD_STATE
00607     * -AUBO_INVL_ARGUMENT
00608     * -AUBO_BAD_STATE
00609     *
00610     * @throws arcs::common_interface::AuboException
00611     *
00612     * @par Python 函数原型
00613     * setMountingPose(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
00614     *
00615     * @par Lua 函数原型
00616     * setMountingPose(pose: table) -> nil
00617     *
00618     * @par Lua 示例
00619     * setMountingPose({0.0,0.0,0.0,0.0,0.0,0.0})
00620     *
00621     * @par JSON-RPC 请求示例
00622     * {"jsonrpc": "2.0", "method": "robl.RobotConfig.setMountingPose", "params": [[0.0,0.0,0.0,0.0,0.0,0.0]], "id": 1}
00623     *
00624     * @par JSON-RPC 响应示例
00625     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00626     * \endchinese
00627     * \english
00628     * Set the mounting pose (robot base coordinate system relative to world
00629     * coordinate system) world->base
00630     *
00631     * Typically used in multi-robot systems, default is [0,0,0,0,0,0]
00632     *
00633     * @param pose Mounting pose
00634     * @return Returns 0 on success; error code on failure
00635     * AUBO_BUSY
00636     * AUBO_BAD_STATE
00637     * -AUBO_INVL_ARGUMENT
00638     * -AUBO_BAD_STATE
00639     *
00640     * @throws arcs::common_interface::AuboException
00641     *
00642     * @par Python function prototype
00643     * setMountingPose(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
00644     *
00645     * @par Lua function prototype
00646     * setMountingPose(pose: table) -> nil
00647     *
00648     * @par Lua example
00649     * setMountingPose({0.0,0.0,0.0,0.0,0.0,0.0})
00650     *
00651     * @par JSON-RPC request example
00652     * {"jsonrpc": "2.0", "method": "robl.RobotConfig.setMountingPose", "params": [[0.0,0.0,0.0,0.0,0.0,0.0]], "id": 1}
00653     *
00654     * @par JSON-RPC response example
00655     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00656     * \endenglish
00657     */
00658     int setMountingPose(const std::vector<double> &pose);
00659
00660     /**
00661     * @ingroup RobotConfig
00662     * \chinese
00663     * 获取安装位姿（机器人的基坐标系相对于世界坐标系）
00664     *
00665     * @return 安装位姿
00666     *
00667     * @throws arcs::common_interface::AuboException
00668     *
00669     * @par Python 函数原型
00670     * getMountingPose(self: pyaubo_sdk.RobotConfig) -> List[float]
00671     *

```

```

00672     * @par Lua 函数原型
00673     * getMountingPose() -> table
00674     *
00675     * @par Lua 示例
00676     * MountingPose = getMountingPose()
00677     *
00678     * @par JSON-RPC 请求示例
00679     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getMountingPose", "params": [], "id": 1}
00680     *
00681     * @par JSON-RPC 响应示例
00682     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
00683     * \endchinese
00684     * \english
00685     * Get the mounting pose (robot base coordinate system relative to world
00686     * coordinate system)
00687     *
00688     * @return Mounting pose
00689     *
00690     * @throws arcs::common_interface::AuboException
00691     *
00692     * @par Python function prototype
00693     * getMountingPose(self: pyaubo_sdk.RobotConfig) -> List[float]
00694     *
00695     * @par Lua function prototype
00696     * getMountingPose() -> table
00697     *
00698     * @par Lua example
00699     * MountingPose = getMountingPose()
00700     *
00701     * @par JSON-RPC request example
00702     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getMountingPose", "params": [], "id": 1}
00703     *
00704     * @par JSON-RPC response example
00705     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
00706     * \endenglish
00707     */
00708 std::vector<double> getMountingPose();
00709
00710 /**
00711  * @ingroup RobotConfig
00712  * \chinese
00713  * 将机器人绑定到一个坐标系，如果这个坐标系是运动的，那机器人也会跟着运动
00714  * 应用于地轨或者龙门
00715  * 这个函数调用的时候 frame 和 ROBOTBASE 的相对关系就固定了
00716  * \endchinese
00717  * \english
00718  * Attach the robot base to a coordinate frame. If this frame moves, the
00719  * robot will move accordingly. Used for applications like ground rails or
00720  * gantries. When this function is called, the relationship between the
00721  * frame and ROBOTBASE is fixed. \endenglish
00722  */
00723 int attachRobotBaseTo(const std::string &frame);
00724 std::string getRobotBaseParent();
00725
00726 /**
00727  * @ingroup RobotConfig
00728  * \chinese
00729  * 设置工件数据，编程点位都是基于工件坐标系
00730  * \endchinese
00731  * \english
00732  * Set work object data. All programmed points are based on the work object
00733  * coordinate system. \endenglish
00734  */
00735
00736 int setWorkObjectData(const WObjectData &wobj);
00737
00738 /**
00739  * @ingroup RobotConfig
00740  * \chinese
00741  * 设置碰撞灵敏度等级
00742  * 数值越大越灵敏
00743  *
00744  * @param level 碰撞灵敏度等级
00745  * 0: 关闭碰撞检测功能
00746  * 1-9: 碰撞灵敏等级
00747  * @return 成功返回 0; 失败返回错误码
00748  * AUBO_BUSY
00749  * AUBO_BAD_STATE
00750  * -AUBO_INVL_ARGUMENT
00751  * -AUBO_BAD_STATE
00752  *
00753  * @throws arcs::common_interface::AuboException
00754  *
00755  * @par Python 函数原型
00756  * setCollisionLevel(self: pyaubo_sdk.RobotConfig, arg0: int) -> int
00757  *
00758  * @par Lua 函数原型

```



```

00759     * setCollisionLevel(level: number) -> nil
00760     *
00761     * @par Lua 示例
00762     * setCollisionLevel(6)
00763     *
00764     * @par JSON-RPC 请求示例
00765     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.setCollisionLevel", "params": [6], "id": 1}
00766     *
00767     * @par JSON-RPC 响应示例
00768     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00769     * \endchinese
00770     * \english
00771     * Set the collision sensitivity level.
00772     * The higher the value, the more sensitive.
00773     *
00774     * @param level Collision sensitivity level
00775     * 0: Disable collision detection
00776     * 1-9: Collision sensitivity levels
00777     * @return Returns 0 on success; error code on failure
00778     * AUBO_BUSY
00779     * AUBO_BAD_STATE
00780     * -AUBO_INVL_ARGUMENT
00781     * -AUBO_BAD_STATE
00782     *
00783     * @throws arcs::common_interface::AuboException
00784     *
00785     * @par Python function prototype
00786     * setCollisionLevel(self: pyaubo_sdk.RobotConfig, arg0: int) -> int
00787     *
00788     * @par Lua function prototype
00789     * setCollisionLevel(level: number) -> nil
00790     *
00791     * @par Lua example
00792     * setCollisionLevel(6)
00793     *
00794     * @par JSON-RPC request example
00795     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.setCollisionLevel", "params": [6], "id": 1}
00796     *
00797     * @par JSON-RPC response example
00798     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00799     * \endenglish
00800     */
00801 int setCollisionLevel(int level);
00802
00803 /**
00804  * @ingroup RobotConfig
00805  * \chinese
00806  * 获取碰撞灵敏度等级
00807  *
00808  * @return 碰撞灵敏度等级
00809  *
00810  * @throws arcs::common_interface::AuboException
00811  *
00812  * @par Python 函数原型
00813  * getCollisionLevel(self: pyaubo_sdk.RobotConfig) -> int
00814  *
00815  * @par Lua 函数原型
00816  * getCollisionLevel() -> number
00817  *
00818  * @par Lua 示例
00819  * CollisionLevel = getCollisionLevel()
00820  *
00821  * @par JSON-RPC 请求示例
00822  * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getCollisionLevel", "params": [], "id": 1}
00823  *
00824  * @par JSON-RPC 响应示例
00825  * {"id": 1, "jsonrpc": "2.0", "result": 6}
00826  * \endchinese
00827  * \english
00828  * Get the collision sensitivity level.
00829  *
00830  * @return Collision sensitivity level.
00831  *
00832  * @throws arcs::common_interface::AuboException
00833  *
00834  * @par Python function prototype
00835  * getCollisionLevel(self: pyaubo_sdk.RobotConfig) -> int
00836  *
00837  * @par Lua function prototype
00838  * getCollisionLevel() -> number
00839  *
00840  * @par Lua example
00841  * CollisionLevel = getCollisionLevel()
00842  *
00843  * @par JSON-RPC request example
00844  * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getCollisionLevel", "params": [], "id": 1}
00845  *

```

```

00846     * @par JSON-RPC response example
00847     * {"id":1,"jsonrpc":"2.0","result":6}
00848     * \endenglish
00849     */
00850 int getCollisionLevel();
00851
00852 /**
00853  * @ingroup RobotConfig
00854  * \chinese
00855  * 设置碰撞停止类型
00856  *
00857  * @param type 类型 \n
00858  * 0: 碰撞后浮动, 即碰撞之后进入拖动示教模式 \n
00859  * 1: 碰撞后静止 \n
00860  * 2: 碰撞后抱闸
00861  *
00862  * @return 成功返回 0; 失败返回错误码
00863  * AUBO_BUSY
00864  * AUBO_BAD_STATE
00865  * -AUBO_INVL_ARGUMENT
00866  * -AUBO_BAD_STATE
00867  *
00868  * @throws arcs::common_interface::AuboException
00869  *
00870  * @par Python 函数原型
00871  * setCollisionStopType(self: pyaubo_sdk.RobotConfig, arg0: int) -> int
00872  *
00873  * @par Lua 函数原型
00874  * setCollisionStopType(type: number) -> nil
00875  *
00876  * @par Lua 示例
00877  * setCollisionStopType(1)
00878  *
00879  * @par JSON-RPC 请求示例
00880  * {"jsonrpc":"2.0","method":"robl.RobotConfig.setCollisionStopType","params":[1],"id":1}
00881  *
00882  * @par JSON-RPC 响应示例
00883  * {"id":1,"jsonrpc":"2.0","result":0}
00884  * \endchinese
00885  * \english
00886  * Set the collision stop type.
00887  *
00888  * @param type Type \n
00889  * 0: Floating after collision, i.e., enters freedrive mode after collision
00890  * \n 1: Stop after collision \n 2: Brake after collision
00891  *
00892  * @return Returns 0 on success; error code on failure
00893  * AUBO_BUSY
00894  * AUBO_BAD_STATE
00895  * -AUBO_INVL_ARGUMENT
00896  * -AUBO_BAD_STATE
00897  *
00898  * @throws arcs::common_interface::AuboException
00899  *
00900  * @par Python function prototype
00901  * setCollisionStopType(self: pyaubo_sdk.RobotConfig, arg0: int) -> int
00902  *
00903  * @par Lua function prototype
00904  * setCollisionStopType(type: number) -> nil
00905  *
00906  * @par Lua example
00907  * setCollisionStopType(1)
00908  *
00909  * @par JSON-RPC request example
00910  * {"jsonrpc":"2.0","method":"robl.RobotConfig.setCollisionStopType","params":[1],"id":1}
00911  *
00912  * @par JSON-RPC response example
00913  * {"id":1,"jsonrpc":"2.0","result":0}
00914  * \endenglish
00915  */
00916 int setCollisionStopType(int type);
00917
00918 /**
00919  * @ingroup RobotConfig
00920  * \chinese
00921  * 获取碰撞停止类型
00922  *
00923  * @return 返回碰撞停止类型 \n
00924  * 0: 碰撞后浮动, 即碰撞之后进入拖动示教模式 \n
00925  * 1: 碰撞后静止 \n
00926  * 2: 碰撞后抱闸
00927  *
00928  * @throws arcs::common_interface::AuboException
00929  *
00930  * @par Python 函数原型
00931  * getCollisionStopType(self: pyaubo_sdk.RobotConfig) -> int
00932  *

```

```

00933     * @par Lua 函数原型
00934     * getCollisionStopType() -> number
00935     *
00936     * @par Lua 示例
00937     * CollisionStopType = getCollisionStopType()
00938     *
00939     * @par JSON-RPC 请求示例
00940     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getCollisionStopType", "params": [], "id": 1}
00941     *
00942     * @par JSON-RPC 响应示例
00943     * {"id": 1, "jsonrpc": "2.0", "result": 1}
00944     *
00945     * \endchinese
00946     * \english
00947     * Get the collision stop type.
00948     *
00949     * @return The collision stop type. \n
00950     * 0: Floating after collision, i.e., enters freedrive mode after collision
00951     * \n 1: Stop after collision \n 2: Brake after collision
00952     *
00953     * @throws arcs::common_interface::AuboException
00954     *
00955     * @par Python function prototype
00956     * getCollisionStopType(self: pyaubo_sdk.RobotConfig) -> int
00957     *
00958     * @par Lua function prototype
00959     * getCollisionStopType() -> number
00960     *
00961     * @par Lua example
00962     * CollisionStopType = getCollisionStopType()
00963     *
00964     * @par JSON-RPC request example
00965     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getCollisionStopType", "params": [], "id": 1}
00966     *
00967     * @par JSON-RPC response example
00968     * {"id": 1, "jsonrpc": "2.0", "result": 1}
00969     *
00970     * \endenglish
00971     */
00972 int getCollisionStopType();
00973
00974 /**
00975  * @ingroup RobotConfig
00976  * \chinese
00977  * 设置机器人的原点位置
00978  *
00979  * @param positions 关节角度
00980  * @return 成功返回 0；失败返回错误码
00981  * AUBO_BUSY
00982  * AUBO_BAD_STATE
00983  * -AUBO_INVL_ARGUMENT
00984  * -AUBO_BAD_STATE
00985  *
00986  * @throws arcs::common_interface::AuboException
00987  *
00988  * @par JSON-RPC 请求示例
00989  * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.setHomePosition", "params":
[[0.0, -0.2617993877991494, 1.74532925199433, 0.4363323129985824, 1.570796326794897, 0.0]], "id": 1}
00990  *
00991  * @par JSON-RPC 响应示例
00992  * {"id": 1, "jsonrpc": "2.0", "result": 0}
00993  * \endchinese
00994  * \english
00995  * Set the robot's home position.
00996  *
00997  * @param positions Joint angles
00998  * @return Returns 0 on success; error code on failure
00999  * AUBO_BUSY
01000  * AUBO_BAD_STATE
01001  * -AUBO_INVL_ARGUMENT
01002  * -AUBO_BAD_STATE
01003  *
01004  * @throws arcs::common_interface::AuboException
01005  *
01006  * @par JSON-RPC request example
01007  * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.setHomePosition", "params":
[[0.0, -0.2617993877991494, 1.74532925199433, 0.4363323129985824, 1.570796326794897, 0.0]], "id": 1}
01008  *
01009  * @par JSON-RPC response example
01010  * {"id": 1, "jsonrpc": "2.0", "result": 0}
01011  * \endenglish
01012  */
01013 int setHomePosition(const std::vector<double> &positions);
01014
01015 /**
01016  * @ingroup RobotConfig
01017  * \chinese

```

```

01018     * 获取机器人的原点位置
01019     *
01020     * @return
01021     *
01022     * @throws arcs::common_interface::AuboException
01023     *
01024     * @par JSON-RPC 请求示例
01025     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getHomePosition", "params": [], "id": 1}
01026     *
01027     * @par JSON-RPC 响应示例
01028     * {"id": 1, "jsonrpc": "2.0", "result":
[0.0, -0.2617993877991494, 1.74532925199433, 0.4363323129985824, 1.570796326794897, 0.0]}
01029     * \endchinese
01030     * \english
01031     * Get the robot's home position.
01032     *
01033     * @return
01034     *
01035     * @throws arcs::common_interface::AuboException
01036     *
01037     * @par JSON-RPC request example
01038     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getHomePosition", "params": [], "id": 1}
01039     *
01040     * @par JSON-RPC response example
01041     * {"id": 1, "jsonrpc": "2.0", "result":
[0.0, -0.2617993877991494, 1.74532925199433, 0.4363323129985824, 1.570796326794897, 0.0]}
01042     * \endenglish
01043     */
01044     std::vector<double> getHomePosition();
01045
01046     /**
01047     * @ingroup RobotConfig
01048     * \chinese
01049     * 设置拖动阻尼
01050     *
01051     * @param damp 阻尼
01052     *
01053     * @return 成功返回 0；失败返回错误码
01054     * AUBO_BUSY
01055     * AUBO_BAD_STATE
01056     * -AUBO_INVL_ARGUMENT
01057     * -AUBO_BAD_STATE
01058     *
01059     * @throws arcs::common_interface::AuboException
01060     *
01061     * @par Python 函数原型
01062     * setFreedriveDamp(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
01063     *
01064     * @par Lua 函数原型
01065     * setFreedriveDamp(damp: table) -> nil
01066     *
01067     * @par Lua 示例
01068     * setFreedriveDamp({0.5,0.5,0.5,0.5,0.5,0.5})
01069     *
01070     * @par JSON-RPC 请求示例
01071     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.setFreedriveDamp", "params": [[0.5,0.5,0.5,0.5,0.5,0.5]], "id": 1}
01072     *
01073     * @par JSON-RPC 响应示例
01074     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01075     *
01076     * \endchinese
01077     * \english
01078     * Set freedrive damping.
01079     *
01080     * @param damp Damping values.
01081     *
01082     * @return Returns 0 on success; error code on failure
01083     * AUBO_BUSY
01084     * AUBO_BAD_STATE
01085     * -AUBO_INVL_ARGUMENT
01086     * -AUBO_BAD_STATE
01087     *
01088     * @throws arcs::common_interface::AuboException
01089     *
01090     * @par Python function prototype
01091     * setFreedriveDamp(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
01092     *
01093     * @par Lua function prototype
01094     * setFreedriveDamp(damp: table) -> nil
01095     *
01096     * @par Lua example
01097     * setFreedriveDamp({0.5,0.5,0.5,0.5,0.5,0.5})
01098     *
01099     * @par JSON-RPC request example
01100     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.setFreedriveDamp", "params": [[0.5,0.5,0.5,0.5,0.5,0.5]], "id": 1}
01101     *
01102     * @par JSON-RPC response example

```

```

01103     * {"id":1,"jsonrpc":"2.0","result":0}
01104     *
01105     * \endenglish
01106     */
01107 int setFreedriveDamp(const std::vector<double> &damp);
01108
01109 /**
01110  * @ingroup RobotConfig
01111  * \chinese
01112  * 获取拖动阻尼
01113  *
01114  * @return 拖动阻尼
01115  *
01116  * @throws arcs::common_interface::AuboException
01117  *
01118  * @par Python 函数原型
01119  * getFreedriveDamp(self: pyaubo_sdk.RobotConfig) -> List[float]
01120  *
01121  * @par Lua 函数原型
01122  * getFreedriveDamp() -> table
01123  *
01124  * @par Lua 示例
01125  * FreedriveDamp = getFreedriveDamp()
01126  *
01127  * @par JSON-RPC 请求示例
01128  * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getFreedriveDamp","params":[],"id":1}
01129  *
01130  * @par JSON-RPC 响应示例
01131  * {"id":1,"jsonrpc":"2.0","result":[0.5,0.5,0.5,0.5,0.5,0.5]}
01132  * \endchinese
01133  * \english
01134  * Get freedrive damping.
01135  *
01136  * @return Freedrive damping values.
01137  *
01138  * @throws arcs::common_interface::AuboException
01139  *
01140  * @par Python function prototype
01141  * getFreedriveDamp(self: pyaubo_sdk.RobotConfig) -> List[float]
01142  *
01143  * @par Lua function prototype
01144  * getFreedriveDamp() -> table
01145  *
01146  * @par Lua example
01147  * FreedriveDamp = getFreedriveDamp()
01148  *
01149  * @par JSON-RPC request example
01150  * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getFreedriveDamp","params":[],"id":1}
01151  *
01152  * @par JSON-RPC response example
01153  * {"id":1,"jsonrpc":"2.0","result":[0.5,0.5,0.5,0.5,0.5,0.5]}
01154  * \endenglish
01155  */
01156 std::vector<double> getFreedriveDamp();
01157
01158 /**
01159  * @ingroup RobotConfig
01160  * \chinese
01161  * 设置混合拖动阻尼
01162  *
01163  * @param damp 阻尼
01164  *
01165  * @return 成功返回 0; 失败返回错误码
01166  * AUBO_BUSY
01167  * AUBO_BAD_STATE
01168  * -AUBO_INVL_ARGUMENT
01169  * -AUBO_BAD_STATE
01170  *
01171  * @throws arcs::common_interface::AuboException
01172  *
01173  * @par Python 函数原型
01174  * setHandguidDamp(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
01175  *
01176  * @par Lua 函数原型
01177  * setHandguidDamp(damp: table) -> number
01178  *
01179  * @par Lua 示例
01180  * setHandguidDamp({0.5,0.5,0.5,0.5,0.5,0.5})
01181  *
01182  * @par JSON-RPC 请求示例
01183  * {"jsonrpc":"2.0","method":"rob1.RobotConfig.setHandguidDamp","params":[[0.5,0.5,0.5,0.5,0.5,0.5]],"id":1}
01184  *
01185  * @par JSON-RPC 响应示例
01186  * {"id":1,"jsonrpc":"2.0","result":0}
01187  * \endchinese
01188  * \english
01189  * Set handguiding damping.

```

```

01190      *
01191      * @param damp damping coef.
01192      *
01193      * @return return 0 for success; return rror code for failure
01194      * AUBO_BUSY
01195      * AUBO_BAD_STATE
01196      * -AUBO_INVL_ARGUMENT
01197      * -AUBO_BAD_STATE
01198      *
01199      * @throws arcs::common_interface::AuboException
01200      *
01201      * @par Python function prototype
01202      * setHandguidDamp(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
01203      *
01204      * @par Lua function prototype
01205      * setHandguidDamp(damp: table) -> number
01206      *
01207      * @par Lua example
01208      * setHandguidDamp({0.5,0.5,0.5,0.5,0.5,0.5})
01209      *
01210      * @par JSON-RPC request example
01211      * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.setHandguidDamp", "params": [[0.5,0.5,0.5,0.5,0.5,0.5]], "id": 1}
01212      *
01213      * @par JSON-RPC response example
01214      * {"id": 1, "jsonrpc": "2.0", "result": 0}
01215      * \endenglish
01216      */
01217 int setHandguidDamp(const std::vector<double> &damp);
01218
01219 /**
01220  * @ingroup RobotConfig
01221  * \chinese
01222  * 获取混合拖动阻尼
01223  *
01224  * @return 拖动阻尼
01225  *
01226  * @throws arcs::common_interface::AuboException
01227  *
01228  * @par Python 函数原型
01229  * getHandguidDamp(self: pyaubo_sdk.RobotConfig) -> List[float]
01230  *
01231  * @par Lua 函数原型
01232  * getHandguidDamp() -> table
01233  *
01234  * @par Lua 示例
01235  * HandguidDamp = getHandguidDamp()
01236  *
01237  * @par JSON-RPC 请求示例
01238  * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getHandguidDamp", "params": [], "id": 1}
01239  *
01240  * @par JSON-RPC 响应示例
01241  * {"id": 1, "jsonrpc": "2.0", "result": [0.5,0.5,0.5,0.5,0.5,0.5]}
01242  * \endchinese
01243  * \english
01244  * Get handguiding damping.
01245  *
01246  * @return handguiding damping
01247  *
01248  * @throws arcs::common_interface::AuboException
01249  *
01250  * @par Python function prototype
01251  * getHandguidDamp(self: pyaubo_sdk.RobotConfig) -> List[float]
01252  *
01253  * @par Lua fuunction prototype
01254  * getHandguidDamp() -> table
01255  *
01256  * @par Lua example
01257  * HandguidDamp = getHandguidDamp()
01258  *
01259  * @par JSON-RPC request example
01260  * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getHandguidDamp", "params": [], "id": 1}
01261  *
01262  * @par JSON-RPC response example
01263  * {"id": 1, "jsonrpc": "2.0", "result": [0.5,0.5,0.5,0.5,0.5,0.5]}
01264  * \endenglish
01265  */
01266 std::vector<double> getHandguidDamp();
01267
01268 /**
01269  * @ingroup RobotConfig
01270  * \chinese
01271  * 获取机器人 DH 参数
01272  * alpha a d theta beta
01273  *
01274  * @param real 读取真实参数（理论值 + 补偿值）或者理论参数
01275  * @return 返回机器人 DH 参数
01276  *

```

```

01277     * @throws arcs::common_interface::AuboException
01278     *
01279     * @par Python 函数原型
01280     * getKinematicsParam(self: pyaubo_sdk.RobotConfig, arg0: bool) -> Dict[str,
01281     * List[float]]
01282     *
01283     * @par Lua 函数原型
01284     * getKinematicsParam(real: boolean) -> table
01285     *
01286     * @par Lua 示例
01287     * KinematicsParam = getKinematicsParam(true)
01288     *
01289     * @par JSON-RPC 请求示例
01290     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getKinematicsParam", "params": [true], "id": 1}
01291     *
01292     * @par JSON-RPC 响应示例
01293     * {"id": 1, "jsonrpc": "2.0", "result": {"a":
01294 [0.0, -0.0001255999959539622, 0.4086348000024445, 0.3757601999930339, -0.00018950000230688602, 3.7000001611886546e-05],
01295     * "alpha":
01296 [0.0, -1.5701173967707458, 3.1440308735524347, 3.14650750358636, -1.5703093767832055, 1.5751177669179182], "beta":
01297 [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]},
01298     * "d": [0.122, 0.12154769999941345, -4.769999941345304e-05, 4.769999941345304e-05, 0.10287890000385232, 0.116],
01299     * "theta": [3.141592653589793, -1.5707963267948966, 0.0, -1.5707963267948966, 0.0, 0.0]}}
01300     * \endchinese
01301     * \english
01302     * Get the robot's DH parameters: alpha, a, d, theta, beta.
01303     *
01304     * @param real Read real parameters (theoretical + compensation) or
01305     * theoretical parameters.
01306     * @return Returns the robot's DH parameters.
01307     *
01308     * @throws arcs::common_interface::AuboException
01309     *
01310     * @par Python function prototype
01311     * getKinematicsParam(self: pyaubo_sdk.RobotConfig, arg0: bool) -> Dict[str,
01312     * List[float]]
01313     *
01314     * @par Lua function prototype
01315     * getKinematicsParam(real: boolean) -> table
01316     *
01317     * @par Lua example
01318     * KinematicsParam = getKinematicsParam(true)
01319     *
01320     * @par JSON-RPC request example
01321     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getKinematicsParam", "params": [true], "id": 1}
01322     *
01323     * @par JSON-RPC response example
01324     * {"id": 1, "jsonrpc": "2.0", "result": {"a":
01325 [0.0, -0.0001255999959539622, 0.4086348000024445, 0.3757601999930339, -0.00018950000230688602, 3.7000001611886546e-05],
01326     * "alpha":
01327 [0.0, -1.5701173967707458, 3.1440308735524347, 3.14650750358636, -1.5703093767832055, 1.5751177669179182], "beta":
01328 [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]},
01329     * "d": [0.122, 0.12154769999941345, -4.769999941345304e-05, 4.769999941345304e-05, 0.10287890000385232, 0.116],
01330     * "theta": [3.141592653589793, -1.5707963267948966, 0.0, -1.5707963267948966, 0.0, 0.0]}}
01331     * \endenglish
01332     */
01327 std::unordered_map<std::string, std::vector<double>> getKinematicsParam(
01328     bool real);
01329
01330 /**
01331     * @ingroup RobotConfig
01332     * \chinese
01333     * 获取指定温度下的 DH 参数补偿值:alpha a d theta beta
01334     *
01335     * @param ref_temperature 参考温度 ℃, 默认 20℃
01336     * @return 返回 DH 参数补偿值
01337     *
01338     * @throws arcs::common_interface::AuboException
01339     *
01340     * @par Python 函数原型
01341     * getKinematicsCompensate(self: pyaubo_sdk.RobotConfig, arg0: float) ->
01342     * Dict[str, List[float]]
01343     *
01344     * @par Lua 函数原型
01345     * getKinematicsCompensate(ref_temperature: number) -> table
01346     *
01347     * @par Lua 示例
01348     * KinematicsCompensate = getKinematicsCompensate(20)
01349     *
01350     * @par JSON-RPC 请求示例
01351     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getKinematicsCompensate", "params": [20], "id": 1}
01352     *
01353     * @par JSON-RPC 响应示例
01354     * {"id": 1, "jsonrpc": "2.0", "result": {"a":
01355 [0.0, -0.0001255999959539622, 0.0006348000024445355, -0.0002398000069661066, -0.00018950000230688602, 3.7000001611886546e-05],
01356     * "alpha":
01357 [0.0, 0.000678930024150759, 0.002438219962641597, 0.0049148499965667725, 0.00048695001169107854, 0.004321440123021603],

```

```

01356     * "beta": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0], "d":
[0.0, 4.769999941345304e-05, -4.769999941345304e-05, 4.769999941345304e-05, 0.0003789000038523227, 0.0],
01357     * "theta": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}}
01358     * \endchinese
01359     * \english
01360     * Get DH parameter compensation values (alpha, a, d, theta, beta) at the
01361     * specified temperature.
01362     *
01363     * @param ref_temperature Reference temperature in °C, default is 20°C
01364     * @return Returns DH parameter compensation values.
01365     *
01366     * @throws arcs::common_interface::AuboException
01367     *
01368     * @par Python function prototype
01369     * getKinematicsCompensate(self: pyaubo_sdk.RobotConfig, arg0: float) ->
01370     * Dict[str, List[float]]
01371     *
01372     * @par Lua function prototype
01373     * getKinematicsCompensate(ref_temperature: number) -> table
01374     *
01375     * @par Lua example
01376     * KinematicsCompensate = getKinematicsCompensate(20)
01377     *
01378     * @par JSON-RPC request example
01379     * {"jsonrpc": "2.0", "method": "robl.RobotConfig.getKinematicsCompensate", "params": [20], "id": 1}
01380     *
01381     * @par JSON-RPC response example
01382     * {"id": 1, "jsonrpc": "2.0", "result": {"a":
[0.0, -0.000125599959539622, 0.0006348000024445355, -0.0002398000069661066, -0.00018950000230688602, 3.7000001611886546e-05],
01383     * "alpha":
[0.0, 0.000678930024150759, 0.002438219962641597, 0.0049148499965667725, 0.00048695001169107854, 0.004321440123021603],
01384     * "beta": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0], "d":
[0.0, 4.769999941345304e-05, -4.769999941345304e-05, 4.769999941345304e-05, 0.0003789000038523227, 0.0],
01385     * "theta": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}}
01386     * \endenglish
01387     */
01388     std::unordered_map<std::string, std::vector<double>>
01389     getKinematicsCompensate(double ref_temperature);
01390
01391     /**
01392     * @ingroup RobotConfig
01393     * \chinese
01394     * 设置标准 DH 补偿到机器人
01395     *
01396     * @param param
01397     *
01398     * @return 成功返回 0；失败返回错误码
01399     * AUBO_BUSY
01400     * AUBO_BAD_STATE
01401     * -AUBO_INVL_ARGUMENT
01402     * -AUBO_BAD_STATE
01403     *
01404     * @throws arcs::common_interface::AuboException
01405     * \endchinese
01406     * \english
01407     * Set standard DH compensation to the robot.
01408     *
01409     * @param param
01410     *
01411     * @return Returns 0 on success; error code on failure
01412     * AUBO_BUSY
01413     * AUBO_BAD_STATE
01414     * -AUBO_INVL_ARGUMENT
01415     * -AUBO_BAD_STATE
01416     *
01417     * @throws arcs::common_interface::AuboException
01418     * \endenglish
01419     */
01420     int setKinematicsCompensate(
01421         const std::unordered_map<std::string, std::vector<double>> &param);
01422
01423     /**
01424     * @ingroup RobotConfig
01425     * \chinese
01426     * 设置需要保存到接口板底座的参数
01427     *
01428     * @param param 补偿数据
01429     *
01430     * @return 成功返回 0；失败返回错误码
01431     * AUBO_BUSY
01432     * AUBO_BAD_STATE
01433     * -AUBO_INVL_ARGUMENT
01434     * -AUBO_BAD_STATE
01435     *
01436     * @throws arcs::common_interface::AuboException
01437     *
01438     * @par Python 函数原型

```



```

01439     * setPersistentParameters(self: pyaubo_sdk.RobotConfig, arg0: str) -> int
01440     *
01441     * @par Lua 函数原型
01442     * setPersistentParameters(param: string) -> nil
01443     *
01444     * \endchinese
01445     * \english
01446     * Set parameters to be saved to the interface board base.
01447     *
01448     * @param param Compensation data.
01449     *
01450     * @return Returns 0 on success; error code on failure.
01451     * AUBO_BUSY
01452     * AUBO_BAD_STATE
01453     * -AUBO_INVL_ARGUMENT
01454     * -AUBO_BAD_STATE
01455     *
01456     * @throws arcs::common_interface::AuboException
01457     *
01458     * @par Python function prototype
01459     * setPersistentParameters(self: pyaubo_sdk.RobotConfig, arg0: str) -> int
01460     *
01461     * @par Lua function prototype
01462     * setPersistentParameters(param: string) -> nil
01463     *
01464     * \endenglish
01465     */
01466 int setPersistentParameters(const std::string &param);
01467
01468 /**
01469     * @ingroup RobotConfig
01470     * \chinese
01471     * 设置硬件抽象层自定义参数
01472     * 目的是为了做不同硬件之间的兼容
01473     *
01474     * @param param 自定义参数
01475     *
01476     * @return 成功返回 0; 失败返回错误码
01477     * AUBO_BUSY
01478     * AUBO_BAD_STATE
01479     * -AUBO_INVL_ARGUMENT
01480     * -AUBO_BAD_STATE
01481     *
01482     * @throws arcs::common_interface::AuboException
01483     * \endchinese
01484     * \english
01485     * Set custom parameters for the hardware abstraction layer.
01486     * The purpose is to ensure compatibility between different hardware.
01487     *
01488     * @param param Custom parameters.
01489     *
01490     * @return Returns 0 on success; error code on failure.
01491     * AUBO_BUSY
01492     * AUBO_BAD_STATE
01493     * -AUBO_INVL_ARGUMENT
01494     * -AUBO_BAD_STATE
01495     *
01496     * @throws arcs::common_interface::AuboException
01497     * \endenglish
01498     */
01499 int setHardwareCustomParameters(const std::string &param);
01500
01501 /**
01502     * @ingroup RobotConfig
01503     * \chinese
01504     * 获取硬件抽象层自定义参数
01505     *
01506     * @param
01507     * @return 返回硬件抽象层自定义的参数
01508     *
01509     * @throws arcs::common_interface::AuboException
01510     * \endchinese
01511     * \english
01512     * Get custom parameters from the hardware abstraction layer.
01513     *
01514     * @param
01515     * @return Returns custom parameters from the hardware abstraction layer.
01516     *
01517     * @throws arcs::common_interface::AuboException
01518     * \endenglish
01519     */
01520 std::string getHardwareCustomParameters(const std::string &param);
01521
01522 /**
01523     * @ingroup RobotConfig
01524     * \chinese
01525     * 设置机器人关节零位

```

```

01526     *
01527     * @return 成功返回 0；失败返回错误码
01528     * AUBO_BUSY
01529     * AUBO_BAD_STATE
01530     * -AUBO_BAD_STATE
01531     *
01532     * @throws arcs::common_interface::AuboException
01533     *
01534     * @par Python 函数原型
01535     * setRobotZero(self: pyaubo_sdk.RobotConfig) -> int
01536     *
01537     * @par Lua 函数原型
01538     * setRobotZero() -> nil
01539     *
01540     * @par Lua 示例
01541     * setRobotZero()
01542     *
01543     * @par JSON-RPC 请求示例
01544     * {"jsonrpc": "2.0", "method": "robl.RobotConfig.setRobotZero", "params": [], "id": 1}
01545     *
01546     * @par JSON-RPC 响应示例
01547     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01548     * \endchinese
01549     * \english
01550     * Set the robot joint zero position.
01551     *
01552     * @return Returns 0 on success; error code on failure
01553     * AUBO_BUSY
01554     * AUBO_BAD_STATE
01555     * -AUBO_BAD_STATE
01556     *
01557     * @throws arcs::common_interface::AuboException
01558     *
01559     * @par Python function prototype
01560     * setRobotZero(self: pyaubo_sdk.RobotConfig) -> int
01561     *
01562     * @par Lua function prototype
01563     * setRobotZero() -> nil
01564     *
01565     * @par Lua example
01566     * setRobotZero()
01567     *
01568     * @par JSON-RPC request example
01569     * {"jsonrpc": "2.0", "method": "robl.RobotConfig.setRobotZero", "params": [], "id": 1}
01570     *
01571     * @par JSON-RPC response example
01572     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01573     * \endenglish
01574     */
01575     int setRobotZero();
01576
01577     /**
01578     * @ingroup RobotConfig
01579     * \chinese
01580     * 获取可用的末端力矩传感器的名字
01581     *
01582     * @return 返回可用的末端力矩传感器的名字
01583     *
01584     * @throws arcs::common_interface::AuboException
01585     *
01586     * @par Python 函数原型
01587     * getTcpForceSensorNames(self: pyaubo_sdk.RobotConfig) -> List[str]
01588     *
01589     * @par Lua 函数原型
01590     * getTcpForceSensorNames() -> table
01591     *
01592     * @par Lua 示例
01593     * TcpForceSensorNames = getTcpForceSensorNames()
01594     *
01595     * @par JSON-RPC 请求示例
01596     * {"jsonrpc": "2.0", "method": "robl.RobotConfig.getTcpForceSensorNames", "params": [], "id": 1}
01597     *
01598     * @par JSON-RPC 响应示例
01599     * {"id": 1, "jsonrpc": "2.0", "result": []}
01600     *
01601     * \endchinese
01602     * \english
01603     * Get the names of available TCP force sensors.
01604     *
01605     * @return Returns the names of available TCP force sensors.
01606     *
01607     * @throws arcs::common_interface::AuboException
01608     *
01609     * @par Python function prototype
01610     * getTcpForceSensorNames(self: pyaubo_sdk.RobotConfig) -> List[str]
01611     *
01612     * @par Lua function prototype

```

```

01613     * getTcpForceSensorNames() -> table
01614     *
01615     * @par Lua example
01616     * TcpForceSensorNames = getTcpForceSensorNames()
01617     *
01618     * @par JSON-RPC request example
01619     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getTcpForceSensorNames", "params": [], "id": 1}
01620     *
01621     * @par JSON-RPC response example
01622     * {"id": 1, "jsonrpc": "2.0", "result": []}
01623     *
01624     * \endenglish
01625     */
01626     std::vector<std::string> getTcpForceSensorNames();
01627
01628     /**
01629     * @ingroup RobotConfig
01630     * \chinese
01631     * 设置末端力矩传感器
01632     * 如果存在内置的末端力矩传感器，默认将使用内置的力矩传感器
01633     *
01634     * @param name 末端力矩传感器的名字
01635     *
01636     * @return 成功返回 0；失败返回错误码
01637     * AUBO_BUSY
01638     * AUBO_BAD_STATE
01639     * -AUBO_INVL_ARGUMENT
01640     * -AUBO_BAD_STATE
01641     *
01642     * @throws arcs::common_interface::AuboException
01643     *
01644     * @par Python 函数原型
01645     * selectTcpForceSensor(self: pyaubo_sdk.RobotConfig, arg0: str) -> int
01646     *
01647     * @par Lua 函数原型
01648     * selectTcpForceSensor(name: string) -> nil
01649     * \endchinese
01650     * \english
01651     * Set the TCP force sensor.
01652     * If there is a built-in TCP force sensor, the built-in sensor will be used
01653     * by default.
01654     *
01655     * @param name Name of the TCP force sensor.
01656     *
01657     * @return Returns 0 on success; error code on failure.
01658     * AUBO_BUSY
01659     * AUBO_BAD_STATE
01660     * -AUBO_INVL_ARGUMENT
01661     * -AUBO_BAD_STATE
01662     *
01663     * @throws arcs::common_interface::AuboException
01664     *
01665     * @par Python function prototype
01666     * selectTcpForceSensor(self: pyaubo_sdk.RobotConfig, arg0: str) -> int
01667     *
01668     * @par Lua function prototype
01669     * selectTcpForceSensor(name: string) -> nil
01670     * \endenglish
01671     */
01672     int selectTcpForceSensor(const std::string &name);
01673
01674     /**
01675     * @ingroup RobotConfig
01676     * \chinese
01677     * 设置传感器安装位姿
01678     *
01679     * @param sensor_pose 传感器安装位姿
01680     *
01681     * @return 成功返回 0；失败返回错误码
01682     * AUBO_BUSY
01683     * AUBO_BAD_STATE
01684     * -AUBO_INVL_ARGUMENT
01685     * -AUBO_BAD_STATE
01686     *
01687     * @throws arcs::common_interface::AuboException
01688     * \endchinese
01689     * \english
01690     * Set the sensor mounting pose.
01691     *
01692     * @param sensor_pose Sensor mounting pose
01693     *
01694     * @return Returns 0 on success; error code on failure
01695     * AUBO_BUSY
01696     * AUBO_BAD_STATE
01697     * -AUBO_INVL_ARGUMENT
01698     * -AUBO_BAD_STATE
01699     *

```

```

01700     * @throws arcs::common_interface::AuboException
01701     * \endenglish
01702     */
01703     int setTcpForceSensorPose(const std::vector<double> &sensor_pose);
01704
01705     /**
01706     * @ingroup RobotConfig
01707     * \chinese
01708     * 获取传感器安装位姿
01709     *
01710     * @return 传感器安装位姿
01711     *
01712     * @throws arcs::common_interface::AuboException
01713     *
01714     * @par JSON-RPC 请求示例
01715     * {"jsonrpc": "2.0", "method": "robl.RobotConfig.getTcpForceSensorPose", "params": [], "id": 1}
01716     *
01717     * @par JSON-RPC 响应示例
01718     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
01719     * \endchinese
01720     * \english
01721     * Get the sensor mounting pose.
01722     *
01723     * @return Sensor mounting pose.
01724     *
01725     * @throws arcs::common_interface::AuboException
01726     *
01727     * @par JSON-RPC request example
01728     * {"jsonrpc": "2.0", "method": "robl.RobotConfig.getTcpForceSensorPose", "params": [], "id": 1}
01729     *
01730     * @par JSON-RPC response example
01731     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
01732     * \endenglish
01733     */
01734     std::vector<double> getTcpForceSensorPose();
01735
01736     /**
01737     * @ingroup RobotConfig
01738     * \chinese
01739     * 是否安装了末端力矩传感器
01740     *
01741     * @return 安装返回 true; 没有安装返回 false
01742     *
01743     * @throws arcs::common_interface::AuboException
01744     *
01745     * @par Python 函数原型
01746     * hasTcpForceSensor(self: pyaubo_sdk.RobotConfig) -> bool
01747     *
01748     * @par Lua 函数原型
01749     * hasTcpForceSensor() -> boolean
01750     *
01751     * @par Lua 示例
01752     * hasTcpForceSensor = hasTcpForceSensor()
01753     *
01754     * @par JSON-RPC 请求示例
01755     * {"jsonrpc": "2.0", "method": "robl.RobotConfig.hasTcpForceSensor", "params": [], "id": 1}
01756     *
01757     * @par JSON-RPC 响应示例
01758     * {"id": 1, "jsonrpc": "2.0", "result": true}
01759     *
01760     * \endchinese
01761     * \english
01762     * Whether a TCP force sensor is installed.
01763     *
01764     * @return Returns true if installed; false otherwise.
01765     *
01766     * @throws arcs::common_interface::AuboException
01767     *
01768     * @par Python function prototype
01769     * hasTcpForceSensor(self: pyaubo_sdk.RobotConfig) -> bool
01770     *
01771     * @par Lua function prototype
01772     * hasTcpForceSensor() -> boolean
01773     *
01774     * @par Lua example
01775     * hasTcpForceSensor = hasTcpForceSensor()
01776     *
01777     * @par JSON-RPC request example
01778     * {"jsonrpc": "2.0", "method": "robl.RobotConfig.hasTcpForceSensor", "params": [], "id": 1}
01779     *
01780     * @par JSON-RPC response example
01781     * {"id": 1, "jsonrpc": "2.0", "result": true}
01782     *
01783     * \endenglish
01784     */
01785     bool hasTcpForceSensor();
01786

```

```

01787 /**
01788  * @ingroup RobotConfig
01789  * \chinese
01790  * 设置末端力矩偏移
01791  *
01792  * @param force_offset 末端力矩偏移
01793  * @return 成功返回 0; 失败返回错误码
01794  *
01795  * @throws arcs::common_interface::AuboException
01796  *
01797  * @par Python 函数原型
01798  * setTcpForceOffset(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
01799  *
01800  * @par Lua 函数原型
01801  * setTcpForceOffset(force_offset: table) -> nil
01802  *
01803  * @par Lua 示例
01804  * setTcpForceOffset({0.0,0.0,0.0,0.0,0.0,0.0})
01805  *
01806  * \endchinese
01807  * \english
01808  * Set the TCP force/torque offset.
01809  *
01810  * @param force_offset TCP force/torque offset
01811  * @return Returns 0 on success; error code on failure
01812  *
01813  * @throws arcs::common_interface::AuboException
01814  *
01815  * @par Python function prototype
01816  * setTcpForceOffset(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
01817  *
01818  * @par Lua function prototype
01819  * setTcpForceOffset(force_offset: table) -> nil
01820  *
01821  * @par Lua example
01822  * setTcpForceOffset({0.0,0.0,0.0,0.0,0.0,0.0})
01823  *
01824  * \endenglish
01825  */
01826 int setTcpForceOffset(const std::vector<double> &force_offset);
01827
01828 /**
01829  * @ingroup RobotConfig
01830  * \chinese
01831  * 获取末端力矩偏移
01832  *
01833  * @return 返回末端力矩偏移
01834  *
01835  * @throws arcs::common_interface::AuboException
01836  *
01837  * @par Python 函数原型
01838  * getTcpForceOffset(self: pyaubo_sdk.RobotConfig) -> List[float]
01839  *
01840  * @par Lua 函数原型
01841  * getTcpForceOffset() -> table
01842  *
01843  * @par Lua 示例
01844  * TcpForceOffset = getTcpForceOffset()
01845  *
01846  * @par JSON-RPC 请求示例
01847  * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getTcpForceOffset", "params": [], "id": 1}
01848  *
01849  * @par JSON-RPC 响应示例
01850  * {"id": 1, "jsonrpc": "2.0", "result": [0.0,0.0,0.0,0.0,0.0,0.0]}
01851  * \endchinese
01852  * \english
01853  * Get the TCP force/torque offset.
01854  *
01855  * @return Returns the TCP force/torque offset.
01856  *
01857  * @throws arcs::common_interface::AuboException
01858  *
01859  * @par Python function prototype
01860  * getTcpForceOffset(self: pyaubo_sdk.RobotConfig) -> List[float]
01861  *
01862  * @par Lua function prototype
01863  * getTcpForceOffset() -> table
01864  *
01865  * @par Lua example
01866  * TcpForceOffset = getTcpForceOffset()
01867  *
01868  * @par JSON-RPC request example
01869  * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getTcpForceOffset", "params": [], "id": 1}
01870  *
01871  * @par JSON-RPC response example
01872  * {"id": 1, "jsonrpc": "2.0", "result": [0.0,0.0,0.0,0.0,0.0,0.0]}
01873  * \endenglish

```

```

01874     */
01875     std::vector<double> getTcpForceOffset();
01876
01877     /**
01878     * @ingroup RobotConfig
01879     * \chinese
01880     * 获取可用的底座力矩传感器的名字
01881     *
01882     * @return 返回可用的底座力矩传感器的名字
01883     *
01884     * @throws arcs::common_interface::AuboException
01885     *
01886     * @par Python 函数原型
01887     * getBaseForceSensorNames(self: pyaubo_sdk.RobotConfig) -> List[str]
01888     *
01889     * @par Lua 函数原型
01890     * getBaseForceSensorNames() -> table
01891     *
01892     * @par Lua 示例
01893     * BaseForceSensorNames = getBaseForceSensorNames()
01894     *
01895     * @par JSON-RPC 请求示例
01896     * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getBaseForceSensorNames","params":[],"id":1}
01897     *
01898     * @par JSON-RPC 响应示例
01899     * {"id":1,"jsonrpc":"2.0","result":[]}
01900     *
01901     * \endchinese
01902     * \english
01903     * Get the names of available base force sensors.
01904     *
01905     * @return Returns the names of available base force sensors.
01906     *
01907     * @throws arcs::common_interface::AuboException
01908     *
01909     * @par Python function prototype
01910     * getBaseForceSensorNames(self: pyaubo_sdk.RobotConfig) -> List[str]
01911     *
01912     * @par Lua function prototype
01913     * getBaseForceSensorNames() -> table
01914     *
01915     * @par Lua example
01916     * BaseForceSensorNames = getBaseForceSensorNames()
01917     *
01918     * @par JSON-RPC request example
01919     * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getBaseForceSensorNames","params":[],"id":1}
01920     *
01921     * @par JSON-RPC response example
01922     * {"id":1,"jsonrpc":"2.0","result":[]}
01923     *
01924     * \endenglish
01925     */
01926     std::vector<std::string> getBaseForceSensorNames();
01927
01928     /**
01929     * @ingroup RobotConfig
01930     * \chinese
01931     * 设置底座力矩传感器
01932     * 如果存在内置的底座力矩传感器，默认将使用内置的力矩传感器
01933     *
01934     * @param name 底座力矩传感器的名字
01935     *
01936     * @return 成功返回 0；失败返回错误码
01937     * AUBO_BUSY
01938     * AUBO_BAD_STATE
01939     * -AUBO_INVL_ARGUMENT
01940     * -AUBO_BAD_STATE
01941     *
01942     * @throws arcs::common_interface::AuboException
01943     *
01944     * @par Python 函数原型
01945     * selectBaseForceSensor(self: pyaubo_sdk.RobotConfig, arg0: str) -> int
01946     *
01947     * @par Lua 函数原型
01948     * selectBaseForceSensor(name: string) -> nil
01949     * \endchinese
01950     * \english
01951     * Set the base force sensor.
01952     * If there is a built-in base force sensor, the built-in sensor will be
01953     * used by default.
01954     *
01955     * @param name Name of the base force sensor.
01956     *
01957     * @return Returns 0 on success; error code on failure.
01958     * AUBO_BUSY
01959     * AUBO_BAD_STATE
01960     * -AUBO_INVL_ARGUMENT

```

```

01961     * -AUBO_BAD_STATE
01962     *
01963     * @throws arcs::common_interface::AuboException
01964     *
01965     * @par Python function prototype
01966     * selectBaseForceSensor(self: pyaubo_sdk.RobotConfig, arg0: str) -> int
01967     *
01968     * @par Lua function prototype
01969     * selectBaseForceSensor(name: string) -> nil
01970     * \endenglish
01971     */
01972 int selectBaseForceSensor(const std::string &name);
01973
01974 /**
01975     * @ingroup RobotConfig
01976     * \chinese
01977     * 是否安装了底座力矩传感器
01978     *
01979     * @return 安装返回 true; 没有安装返回 false
01980     *
01981     * @throws arcs::common_interface::AuboException
01982     *
01983     * @par Python 函数原型
01984     * hasBaseForceSensor(self: pyaubo_sdk.RobotConfig) -> bool
01985     *
01986     * @par Lua 函数原型
01987     * hasBaseForceSensor() -> boolean
01988     *
01989     * @par Lua 示例
01990     * hasBaseForceSensor = hasBaseForceSensor()
01991     *
01992     * @par JSON-RPC 请求示例
01993     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.hasBaseForceSensor", "params": [], "id": 1}
01994     *
01995     * @par JSON-RPC 响应示例
01996     * {"id": 1, "jsonrpc": "2.0", "result": false}
01997     *
01998     * \endchinese
01999     * \english
02000     * Whether a base force sensor is installed.
02001     *
02002     * @return Returns true if installed; false otherwise.
02003     *
02004     * @throws arcs::common_interface::AuboException
02005     *
02006     * @par Python function prototype
02007     * hasBaseForceSensor(self: pyaubo_sdk.RobotConfig) -> bool
02008     *
02009     * @par Lua function prototype
02010     * hasBaseForceSensor() -> boolean
02011     *
02012     * @par Lua example
02013     * hasBaseForceSensor = hasBaseForceSensor()
02014     *
02015     * @par JSON-RPC request example
02016     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.hasBaseForceSensor", "params": [], "id": 1}
02017     *
02018     * @par JSON-RPC response example
02019     * {"id": 1, "jsonrpc": "2.0", "result": false}
02020     *
02021     * \endenglish
02022     */
02023 bool hasBaseForceSensor();
02024
02025 /**
02026     * @ingroup RobotConfig
02027     * \chinese
02028     * 设置底座力矩偏移
02029     *
02030     * @param force_offset 底座力矩偏移
02031     *
02032     * @return 成功返回 0; 失败返回错误码
02033     * AUBO_BUSY
02034     * AUBO_BAD_STATE
02035     * -AUBO_INVL_ARGUMENT
02036     * -AUBO_BAD_STATE
02037     *
02038     * @throws arcs::common_interface::AuboException
02039     *
02040     * @par Python 函数原型
02041     * setBaseForceOffset(self: pyaubo_sdk.RobotConfig, arg0: List[float]) ->
02042     * int
02043     *
02044     * @par Lua 函数原型
02045     * setBaseForceOffset(force_offset: table) -> nil
02046     *
02047     * @par Lua 示例

```

```

02048     * setBaseForceOffset({0.0,0.0,0.0,0.0,0.0,0.0})
02049     *
02050     * \endchinese
02051     * \english
02052     * Set the base force/torque offset.
02053     *
02054     * @param force_offset Base force/torque offset
02055     *
02056     * @return Returns 0 on success; error code on failure
02057     * AUBO_BUSY
02058     * AUBO_BAD_STATE
02059     * -AUBO_INVL_ARGUMENT
02060     * -AUBO_BAD_STATE
02061     *
02062     * @throws arcs::common_interface::AuboException
02063     *
02064     * @par Python function prototype
02065     * setBaseForceOffset(self: pyaubo_sdk.RobotConfig, arg0: List[float]) ->
02066     * int
02067     *
02068     * @par Lua function prototype
02069     * setBaseForceOffset(force_offset: table) -> nil
02070     *
02071     * @par Lua 示例
02072     * setBaseForceOffset({0.0,0.0,0.0,0.0,0.0,0.0})
02073     *
02074     * \endenglish
02075     */
02076 int setBaseForceOffset(const std::vector<double> &force_offset);
02077
02078 /**
02079  * @ingroup RobotConfig
02080  * \chinese
02081  * 获取底座力矩偏移
02082  *
02083  * @return 返回底座力矩偏移
02084  *
02085  * @throws arcs::common_interface::AuboException
02086  *
02087  * @par Python 函数原型
02088  * getBaseForceOffset(self: pyaubo_sdk.RobotConfig) -> List[float]
02089  *
02090  * @par Lua 函数原型
02091  * getBaseForceOffset() -> table
02092  *
02093  * @par Lua 示例
02094  * BaseForceOffset = getBaseForceOffset()
02095  *
02096  * @par JSON-RPC 请求示例
02097  * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getBaseForceOffset", "params": [], "id": 1}
02098  *
02099  * @par JSON-RPC 响应示例
02100  * {"id": 1, "jsonrpc": "2.0", "result": []}
02101  *
02102  * \endchinese
02103  * \english
02104  * Get the base force/torque offset.
02105  *
02106  * @return Returns the base force/torque offset.
02107  *
02108  * @throws arcs::common_interface::AuboException
02109  *
02110  * @par Python function prototype
02111  * getBaseForceOffset(self: pyaubo_sdk.RobotConfig) -> List[float]
02112  *
02113  * @par Lua function prototype
02114  * getBaseForceOffset() -> table
02115  *
02116  * @par Lua example
02117  * BaseForceOffset = getBaseForceOffset()
02118  *
02119  * @par JSON-RPC request example
02120  * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getBaseForceOffset", "params": [], "id": 1}
02121  *
02122  * @par JSON-RPC response example
02123  * {"id": 1, "jsonrpc": "2.0", "result": []}
02124  *
02125  * \endenglish
02126  */
02127 std::vector<double> getBaseForceOffset();
02128
02129 /**
02130  * @ingroup RobotConfig
02131  * \chinese
02132  * 获取安全参数校验码 CRC32
02133  *
02134  * @return 返回安全参数校验码

```



```

02135     *
02136     * @throws arcs::common_interface::AuboException
02137     *
02138     * @par Python 函数原型
02139     * getSafetyParametersCheckSum(self: pyaubo_sdk.RobotConfig) -> int
02140     *
02141     * @par Lua 函数原型
02142     * getSafetyParametersCheckSum() -> number
02143     *
02144     * @par Lua 示例
02145     * SafetyParametersCheckSum = getSafetyParametersCheckSum()
02146     *
02147     * @par JSON-RPC 请求示例
02148     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getSafetyParametersCheckSum", "params": [], "id": 1}
02149     *
02150     * @par JSON-RPC 响应示例
02151     * {"id": 1, "jsonrpc": "2.0", "result": 2033397241}
02152     *
02153     * \endchinese
02154     * \english
02155     * Get the safety parameter checksum CRC32.
02156     *
02157     * @return Returns the safety parameter checksum.
02158     *
02159     * @throws arcs::common_interface::AuboException
02160     *
02161     * @par Python function prototype
02162     * getSafetyParametersCheckSum(self: pyaubo_sdk.RobotConfig) -> int
02163     *
02164     * @par Lua function prototype
02165     * getSafetyParametersCheckSum() -> number
02166     *
02167     * @par Lua 示例
02168     * SafetyParametersCheckSum = getSafetyParametersCheckSum()
02169     *
02170     * @par JSON-RPC request example
02171     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getSafetyParametersCheckSum", "params": [], "id": 1}
02172     *
02173     * @par JSON-RPC response example
02174     * {"id": 1, "jsonrpc": "2.0", "result": 2033397241}
02175     *
02176     * \endenglish
02177     */
02178     uint32_t getSafetyParametersCheckSum();
02179
02180     /**
02181     * @ingroup RobotConfig
02182     * \chinese
02183     * 发起确认安全配置参数请求:
02184     * 将安全配置参数写入到安全接口板 flash 或文件
02185     *
02186     * @param parameters 安全配置参数
02187     *
02188     * @return 成功返回 0; 失败返回错误码
02189     * AUBO_BUSY
02190     * AUBO_BAD_STATE
02191     * -AUBO_INVL_ARGUMENT
02192     * -AUBO_BAD_STATE
02193     *
02194     * @throws arcs::common_interface::AuboException
02195     *
02196     * @par Python 函数原型
02197     * confirmSafetyParameters(self: pyaubo_sdk.RobotConfig, arg0:
02198     * arcs::common_interface::RobotSafetyParameterRange) -> int
02199     *
02200     * @par Lua 函数原型
02201     *
02202     * \endchinese
02203     * \english
02204     * Initiate a request to confirm safety configuration parameters:
02205     * Write safety configuration parameters to the safety interface board flash
02206     * or file.
02207     *
02208     * @param parameters Safety configuration parameters.
02209     *
02210     * @return Returns 0 on success; error code on failure.
02211     * AUBO_BUSY
02212     * AUBO_BAD_STATE
02213     * -AUBO_INVL_ARGUMENT
02214     * -AUBO_BAD_STATE
02215     *
02216     * @throws arcs::common_interface::AuboException
02217     *
02218     * @par Python function prototype
02219     * confirmSafetyParameters(self: pyaubo_sdk.RobotConfig, arg0:
02220     * arcs::common_interface::RobotSafetyParameterRange) -> int
02221     *

```

```

02222     * @par Lua function prototype
02223     *
02224     * \endenglish
02225     */
02226 int confirmSafetyParameters(const RobotSafetyParameterRange &parameters);
02227
02228 /**
02229  * @ingroup RobotConfig
02230  * \chinese
02231  * 计算安全参数的 CRC32 校验值
02232  *
02233  * @return crc32
02234  *
02235  * @throws arcs::common_interface::AuboException
02236  * \endchinese
02237  * \english
02238  * Calculate the CRC32 checksum of safety parameters.
02239  *
02240  * @return crc32
02241  *
02242  * @throws arcs::common_interface::AuboException
02243  * \endenglish
02244  */
02245 uint32_t calcSafetyParametersCheckSum(
02246     const RobotSafetyParameterRange &parameters);
02247
02248 /**
02249  * @ingroup RobotConfig
02250  * \chinese
02251  * 获取关节最大位置 (物理极限)
02252  *
02253  * @return 返回关节最大位置
02254  *
02255  * @throws arcs::common_interface::AuboException
02256  *
02257  * @par Python 函数原型
02258  * getJointMaxPositions(self: pyaubo_sdk.RobotConfig) -> List[float]
02259  *
02260  * @par Lua 函数原型
02261  * getJointMaxPositions() -> table
02262  *
02263  * @par Lua 示例
02264  * JointMaxPositions = getJointMaxPositions()
02265  *
02266  * @par JSON-RPC 请求示例
02267  * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getJointMaxPositions", "params": [], "id": 1}
02268  *
02269  * @par JSON-RPC 响应示例
02270  * {"id": 1, "jsonrpc": "2.0", "result":
[6.283185307179586, 6.283185307179586, 6.283185307179586, 6.283185307179586, 6.283185307179586, 6.283185307179586]}
02271  * \endchinese
02272  * \english
02273  * Get the joint maximum positions (physical limits).
02274  *
02275  * @return Returns the joint maximum positions.
02276  *
02277  * @throws arcs::common_interface::AuboException
02278  *
02279  * @par Python function prototype
02280  * getJointMaxPositions(self: pyaubo_sdk.RobotConfig) -> List[float]
02281  *
02282  * @par Lua function prototype
02283  * getJointMaxPositions() -> table
02284  *
02285  * @par Lua example
02286  * JointMaxPositions = getJointMaxPositions()
02287  *
02288  * @par JSON-RPC request example
02289  * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getJointMaxPositions", "params": [], "id": 1}
02290  *
02291  * @par JSON-RPC response example
02292  * {"id": 1, "jsonrpc": "2.0", "result":
[6.283185307179586, 6.283185307179586, 6.283185307179586, 6.283185307179586, 6.283185307179586, 6.283185307179586]}
02293  * \endenglish
02294  */
02295 std::vector<double> getJointMaxPositions();
02296
02297 /**
02298  * @ingroup RobotConfig
02299  * \chinese
02300  * 获取关节最小位置 (物理极限)
02301  *
02302  * @return 返回关节最小位置
02303  *
02304  * @throws arcs::common_interface::AuboException
02305  *
02306  * @par Python 函数原型

```

```

02307     * getJointMinPositions(self: pyaubo_sdk.RobotConfig) -> List[float]
02308     *
02309     * @par Lua 函数原型
02310     * getJointMinPositions() -> table
02311     *
02312     * @par Lua 示例
02313     * JointMinPositions = getJointMinPositions()
02314     *
02315     * @par JSON-RPC 请求示例
02316     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getJointMinPositions", "params": [], "id": 1}
02317     *
02318     * @par JSON-RPC 响应示例
02319     * {"id": 1, "jsonrpc": "2.0", "result":
[-6.283185307179586, -6.283185307179586, -6.283185307179586, -6.283185307179586, -6.283185307179586, -6.283185307179586]}
02320     * \endchinese
02321     * \english
02322     * Get the joint minimum positions (physical limits).
02323     *
02324     * @return Returns the joint minimum positions.
02325     *
02326     * @throws arcs::common_interface::AuboException
02327     *
02328     * @par Python function prototype
02329     * getJointMinPositions(self: pyaubo_sdk.RobotConfig) -> List[float]
02330     *
02331     * @par Lua function prototype
02332     * getJointMinPositions() -> table
02333     *
02334     * @par Lua example
02335     * JointMinPositions = getJointMinPositions()
02336     *
02337     * @par JSON-RPC request example
02338     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getJointMinPositions", "params": [], "id": 1}
02339     *
02340     * @par JSON-RPC response example
02341     * {"id": 1, "jsonrpc": "2.0", "result":
[-6.283185307179586, -6.283185307179586, -6.283185307179586, -6.283185307179586, -6.283185307179586, -6.283185307179586]}
02342     * \endenglish
02343     */
02344     std::vector<double> getJointMinPositions();
02345
02346     /**
02347     * @ingroup RobotConfig
02348     * \chinese
02349     * 获取关节最大速度（物理极限）
02350     *
02351     * @return 返回关节最大速度
02352     *
02353     * @throws arcs::common_interface::AuboException
02354     *
02355     * @par Python 函数原型
02356     * getJointMaxSpeeds(self: pyaubo_sdk.RobotConfig) -> List[float]
02357     *
02358     * @par Lua 函数原型
02359     * getJointMaxSpeeds() -> table
02360     *
02361     * @par Lua 示例
02362     * JointMaxSpeeds = getJointMaxSpeeds()
02363     *
02364     * @par JSON-RPC 请求示例
02365     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getJointMaxSpeeds", "params": [], "id": 1}
02366     *
02367     * @par JSON-RPC 响应示例
02368     * {"id": 1, "jsonrpc": "2.0", "result":
[3.892084231947355, 3.892084231947355, 3.892084231947355, 3.892084231947355, 3.1066860685499065, 3.1066860685499065]}
02369     * \endchinese
02370     * \english
02371     * Get the joint maximum speeds (physical limits).
02372     *
02373     * @return Returns the joint maximum speeds.
02374     *
02375     * @throws arcs::common_interface::AuboException
02376     *
02377     * @par Python function prototype
02378     * getJointMaxSpeeds(self: pyaubo_sdk.RobotConfig) -> List[float]
02379     *
02380     * @par Lua function prototype
02381     * getJointMaxSpeeds() -> table
02382     *
02383     * @par Lua example
02384     * JointMaxSpeeds = getJointMaxSpeeds()
02385     *
02386     * @par JSON-RPC request example
02387     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getJointMaxSpeeds", "params": [], "id": 1}
02388     *
02389     * @par JSON-RPC response example
02390     * {"id": 1, "jsonrpc": "2.0", "result":

```

```

[3.892084231947355,3.892084231947355,3.892084231947355,3.1066860685499065,3.1066860685499065,3.1066860685499065]}
02391     * \endenglish
02392     */
02393     std::vector<double> getJointMaxSpeeds();
02394
02395     /**
02396     * @ingroup RobotConfig
02397     * \chinese
02398     * 获取关节最大加速度（物理极限）
02399     *
02400     * @return 返回关节最大加速度
02401     *
02402     * @throws arcs::common_interface::AuboException
02403     *
02404     * @par Python 函数原型
02405     * getJointMaxAccelerations(self: pyaubo_sdk.RobotConfig) -> List[float]
02406     *
02407     * @par Lua 函数原型
02408     * getJointMaxAccelerations() -> table
02409     *
02410     * @par Lua 示例
02411     * JointMaxAccelerations = getJointMaxAccelerations()
02412     *
02413     * @par JSON-RPC 请求示例
02414     * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getJointMaxAccelerations","params":[],"id":1}
02415     *
02416     * @par JSON-RPC 响应示例
02417     * {"id":1,"jsonrpc":"2.0","result":
[31.104877758314785,31.104877758314785,31.104877758314785,20.73625684294463,20.73625684294463,20.73625684294463]}
02418     * \endchinese
02419     * \english
02420     * Get the joint maximum accelerations (physical limits).
02421     *
02422     * @return Returns the joint maximum accelerations.
02423     *
02424     * @throws arcs::common_interface::AuboException
02425     *
02426     * @par Python function prototype
02427     * getJointMaxAccelerations(self: pyaubo_sdk.RobotConfig) -> List[float]
02428     *
02429     * @par Lua function prototype
02430     * getJointMaxAccelerations() -> table
02431     *
02432     * @par Lua example
02433     * JointMaxAccelerations = getJointMaxAccelerations()
02434     *
02435     * @par JSON-RPC request example
02436     * {"id":1,"jsonrpc":"2.0","method":"rob1.RobotConfig.getJointMaxAccelerations","params":[],"id":1}
02437     *
02438     * @par JSON-RPC response example
02439     * {"id":1,"jsonrpc":"2.0","result":
[31.104877758314785,31.104877758314785,31.104877758314785,20.73625684294463,20.73625684294463,20.73625684294463]}
02440     * \endenglish
02441     */
02442     std::vector<double> getJointMaxAccelerations();
02443
02444     /**
02445     * @ingroup RobotConfig
02446     * \chinese
02447     * 获取 TCP 最大速度（物理极限）
02448     *
02449     * @return 返回 TCP 最大速度
02450     *
02451     * @throws arcs::common_interface::AuboException
02452     *
02453     * @par Python 函数原型
02454     * getTcpMaxSpeeds(self: pyaubo_sdk.RobotConfig) -> List[float]
02455     *
02456     * @par Lua 函数原型
02457     * getTcpMaxSpeeds() -> table
02458     *
02459     * @par Lua 示例
02460     * TcpMaxSpeeds = getTcpMaxSpeeds()
02461     *
02462     * @par JSON-RPC 请求示例
02463     * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getTcpMaxSpeeds","params":[],"id":1}
02464     *
02465     * @par JSON-RPC 响应示例
02466     * {"id":1,"jsonrpc":"2.0","result":[2.0,5.0]}
02467     * \endchinese
02468     * \english
02469     * Get the TCP maximum speed (physical limits).
02470     *
02471     * @return Returns the TCP maximum speed.
02472     *
02473     * @throws arcs::common_interface::AuboException
02474     *

```

```

02475     * @par Python function prototype
02476     * getTcpMaxSpeeds(self: pyaubo_sdk.RobotConfig) -> List[float]
02477     *
02478     * @par Lua function prototype
02479     * getTcpMaxSpeeds() -> table
02480     *
02481     * @par Lua example
02482     * TcpMaxSpeeds = getTcpMaxSpeeds()
02483     *
02484     * @par JSON-RPC request example
02485     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getTcpMaxSpeeds", "params": [], "id": 1}
02486     *
02487     * @par JSON-RPC response example
02488     * {"id": 1, "jsonrpc": "2.0", "result": [2.0, 5.0]}
02489     * \endenglish
02490     */
02491     std::vector<double> getTcpMaxSpeeds();
02492
02493     /**
02494     * @ingroup RobotConfig
02495     * \chinese
02496     * 获取 TCP 最大加速度 (物理极限)
02497     *
02498     * @return 返回 TCP 最大加速度
02499     *
02500     * @throws arcs::common_interface::AuboException
02501     *
02502     * @par Python 函数原型
02503     * getTcpMaxAccelerations(self: pyaubo_sdk.RobotConfig) -> List[float]
02504     *
02505     * @par Lua 函数原型
02506     * getTcpMaxAccelerations() -> table
02507     *
02508     * @par Lua 示例
02509     * TcpMaxAccelerations = getTcpMaxAccelerations()
02510     *
02511     * @par JSON-RPC 请求示例
02512     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getTcpMaxAccelerations", "params": [], "id": 1}
02513     *
02514     * @par JSON-RPC 响应示例
02515     * {"id": 1, "jsonrpc": "2.0", "result": [10.0, 10.0]}
02516     *
02517     * \endchinese
02518     * \english
02519     * Get the TCP maximum acceleration (physical limits).
02520     *
02521     * @return Returns the TCP maximum acceleration.
02522     *
02523     * @throws arcs::common_interface::AuboException
02524     *
02525     * @par Python function prototype
02526     * getTcpMaxAccelerations(self: pyaubo_sdk.RobotConfig) -> List[float]
02527     *
02528     * @par Lua function prototype
02529     * getTcpMaxAccelerations() -> table
02530     *
02531     * @par Lua example
02532     * TcpMaxAccelerations = getTcpMaxAccelerations()
02533     *
02534     * @par JSON-RPC request example
02535     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getTcpMaxAccelerations", "params": [], "id": 1}
02536     *
02537     * @par JSON-RPC response example
02538     * {"id": 1, "jsonrpc": "2.0", "result": [10.0, 10.0]}
02539     * \endenglish
02540     */
02541     std::vector<double> getTcpMaxAccelerations();
02542
02543     /**
02544     * @ingroup RobotConfig
02545     * \chinese
02546     * 设置机器人安装姿态
02547     *
02548     * @param gravity 安装姿态
02549     *
02550     * @return 成功返回 0; 失败返回错误码
02551     * AUBO_BUSY
02552     * AUBO_BAD_STATE
02553     * -AUBO_INVL_ARGUMENT
02554     * -AUBO_BAD_STATE
02555     *
02556     * @throws arcs::common_interface::AuboException
02557     *
02558     * @par Python 函数原型
02559     * setGravity(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
02560     *
02561     * @par Lua 函数原型

```

```

02562     * setGravity(gravity: table) -> nil
02563     *
02564     * @par Lua 示例
02565     * setGravity({0.0,0.0,-9.87654321})
02566     *
02567     * @par JSON-RPC 请求示例
02568     * {"jsonrpc":"2.0","method":"rob1.RobotConfig.setGravity","params":[[0.0,0.0,-9.87654321]],"id":1}
02569     *
02570     * @par JSON-RPC 响应示例
02571     * {"id":1,"jsonrpc":"2.0","result":0}
02572     * \endchinese
02573     * \english
02574     * Set the robot installation attitude (gravity vector).
02575     *
02576     * @param gravity Installation attitude (gravity vector)
02577     *
02578     * @return Returns 0 on success; error code on failure
02579     * AUBO_BUSY
02580     * AUBO_BAD_STATE
02581     * -AUBO_INVL_ARGUMENT
02582     * -AUBO_BAD_STATE
02583     *
02584     * @throws arcs::common_interface::AuboException
02585     *
02586     * @par Python function prototype
02587     * setGravity(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
02588     *
02589     * @par Lua function prototype
02590     * setGravity(gravity: table) -> nil
02591     *
02592     * @par Lua example
02593     * setGravity({0.0,0.0,-9.87654321})
02594     *
02595     * @par JSON-RPC request example
02596     * {"jsonrpc":"2.0","method":"rob1.RobotConfig.setGravity","params":[[0.0,0.0,-9.87654321]],"id":1}
02597     *
02598     * @par JSON-RPC response example
02599     * {"id":1,"jsonrpc":"2.0","result":0}
02600     * \endenglish
02601     */
02602     int setGravity(const std::vector<double> &gravity);
02603
02604     /**
02605     * @ingroup RobotConfig
02606     * \chinese
02607     * 获取机器人的安装姿态
02608     *
02609     * 如果机器人底座安装了姿态传感器，则从传感器读取数据，否则按照用户设置
02610     *
02611     * @return 返回安装姿态
02612     *
02613     * @throws arcs::common_interface::AuboException
02614     *
02615     * @par Python 函数原型
02616     * getGravity(self: pyaubo_sdk.RobotConfig) -> List[float]
02617     *
02618     * @par Lua 函数原型
02619     * getGravity() -> table
02620     *
02621     * @par Lua 示例
02622     * Gravity = getGravity()
02623     *
02624     * @par JSON-RPC 请求示例
02625     * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getGravity","params":[],"id":1}
02626     *
02627     * @par JSON-RPC 响应示例
02628     * {"id":1,"jsonrpc":"2.0","result":[0.0,0.0,-9.87654321]}
02629     *
02630     * \endchinese
02631     * \english
02632     * Get the robot's installation attitude (gravity vector).
02633     *
02634     * If the robot base is equipped with an attitude sensor, data is read from
02635     * the sensor; otherwise, the user setting is used.
02636     *
02637     * @return Returns the installation attitude.
02638     *
02639     * @throws arcs::common_interface::AuboException
02640     *
02641     * @par Python function prototype
02642     * getGravity(self: pyaubo_sdk.RobotConfig) -> List[float]
02643     *
02644     * @par Lua function prototype
02645     * getGravity() -> table
02646     *
02647     * @par Lua example
02648     * Gravity = getGravity()

```

```

02649      *
02650      * @par JSON-RPC request example
02651      * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getGravity", "params": [], "id": 1}
02652      *
02653      * @par JSON-RPC response example
02654      * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, -9.87654321]}
02655      *
02656      * \endenglish
02657      */
02658      std::vector<double> getGravity();
02659
02660      /**
02661      * @ingroup RobotConfig
02662      * \chinese
02663      * 设置当前的 TCP 偏移
02664      *
02665      * TCP 偏移表示形式为 (x,y,z,rx,ry,rz)。
02666      * 其中 x、y、z 是工具中心点 (TCP) 在基坐标系下相对于法兰盘中心的位置偏移, 单位是 m。
02667      * rx、ry、rz 是工具中心点 (TCP) 在基坐标系下相对于法兰盘中心的姿态偏移, 是 ZYX 欧拉角, 单位是 rad。
02668      *
02669      * @param offset 当前的 TCP 偏移, 形式为 (x,y,z,rx,ry,rz)
02670      *
02671      * @return 成功返回 0; 失败返回错误码
02672      * AUBO_BUSY
02673      * AUBO_BAD_STATE
02674      * -AUBO_INVL_ARGUMENT
02675      * -AUBO_BAD_STATE
02676      *
02677      * @throws arcs::common_interface::AuboException
02678      *
02679      * @par Python 函数原型
02680      * setTcpOffset(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
02681      *
02682      * @par Lua 函数原型
02683      * setTcpOffset(offset: table) -> nil
02684      *
02685      * @par Lua 示例
02686      * setTcpOffset({0.0,0.0,0.0,0.0,0.0,0.0})
02687      *
02688      * @par JSON-RPC 请求示例
02689      * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.setTcpOffset", "params": [[0.0,0.0,0.0,0.0,0.0,0.0]], "id": 1}
02690      *
02691      * @par JSON-RPC 响应示例
02692      * {"id": 1, "jsonrpc": "2.0", "result": 0}
02693      * \endchinese
02694      * \english
02695      * Set the current TCP offset.
02696      *
02697      * The TCP offset is represented as (x, y, z, rx, ry, rz).
02698      * x, y, z are the position offsets of the Tool Center Point (TCP) relative
02699      * to the flange center in the base coordinate system, in meters. rx, ry, rz
02700      * are the orientation offsets of the TCP relative to the flange center in
02701      * the base coordinate system, as ZYX Euler angles in radians.
02702      *
02703      * @param offset The current TCP offset, in the form (x, y, z, rx, ry, rz)
02704      *
02705      * @return Returns 0 on success; error code on failure
02706      * AUBO_BUSY
02707      * AUBO_BAD_STATE
02708      * -AUBO_INVL_ARGUMENT
02709      * -AUBO_BAD_STATE
02710      *
02711      * @throws arcs::common_interface::AuboException
02712      *
02713      * @par Python function prototype
02714      * setTcpOffset(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> int
02715      *
02716      * @par Lua function prototype
02717      * setTcpOffset(offset: table) -> nil
02718      *
02719      * @par Lua example
02720      * setTcpOffset({0.0,0.0,0.0,0.0,0.0,0.0})
02721      *
02722      * @par JSON-RPC request example
02723      * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.setTcpOffset", "params": [[0.0,0.0,0.0,0.0,0.0,0.0]], "id": 1}
02724      *
02725      * @par JSON-RPC response example
02726      * {"id": 1, "jsonrpc": "2.0", "result": 0}
02727      * \endenglish
02728      */
02729      int setTcpOffset(const std::vector<double> &offset);
02730
02731      /**
02732      * @ingroup RobotConfig
02733      * \chinese
02734      * 获取当前的 TCP 偏移
02735      *

```

```

02736 * TCP 偏移表示形式为 (x,y,z,rx,ry,rz)。
02737 * 其中 x、y、z 是工具中心点 (TCP) 在基坐标系下相对于法兰盘中心的位置偏移, 单位是 m。
02738 * rx、ry、rz 是工具中心点 (TCP) 在基坐标系下相对于法兰盘中心的姿态偏移, 是 ZYX 欧拉角, 单位是 rad。
02739 *
02740 * @return 当前的 TCP 偏移, 形式为 (x,y,z,rx,ry,rz)
02741 *
02742 * @throws arcs::common_interface::AuboException
02743 *
02744 * @par Python 函数原型
02745 * getTcpOffset(self: pyaubo_sdk.RobotConfig) -> List[float]
02746 *
02747 * @par Lua 函数原型
02748 * getTcpOffset() -> table
02749 *
02750 * @par Lua 示例
02751 * TcpOffset = getTcpOffset()
02752 *
02753 * @par JSON-RPC 请求示例
02754 * {"jsonrpc": "2.0", "method": "robl.RobotConfig.getTcpOffset", "params": [], "id": 1}
02755 *
02756 * @par JSON-RPC 响应示例
02757 * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
02758 * \endchinese
02759 * \english
02760 * Get the current TCP offset.
02761 *
02762 * The TCP offset is represented as (x, y, z, rx, ry, rz).
02763 * x, y, z are the position offsets of the Tool Center Point (TCP) relative
02764 * to the flange center in the base coordinate system, in meters. rx, ry, rz
02765 * are the orientation offsets of the TCP relative to the flange center in
02766 * the base coordinate system, as ZYX Euler angles in radians.
02767 *
02768 * @return The current TCP offset, in the form (x, y, z, rx, ry, rz)
02769 *
02770 * @throws arcs::common_interface::AuboException
02771 *
02772 * @par Python function prototype
02773 * getTcpOffset(self: pyaubo_sdk.RobotConfig) -> List[float]
02774 *
02775 * @par Lua function prototype
02776 * getTcpOffset() -> table
02777 *
02778 * @par Lua example
02779 * TcpOffset = getTcpOffset()
02780 *
02781 * @par JSON-RPC request example
02782 * {"jsonrpc": "2.0", "method": "robl.RobotConfig.getTcpOffset", "params": [], "id": 1}
02783 *
02784 * @par JSON-RPC response example
02785 * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
02786 * \endenglish
02787 */
02788 std::vector<double> getTcpOffset();
02789
02790 /**
02791 * @ingroup RobotConfig
02792 * \chinese
02793 * 设置工具端质量、质心及惯量
02794 *
02795 * @param m 工具端质量
02796 * @param com 质心
02797 * @param inertial 惯量
02798 *
02799 * @return 成功返回 0; 失败返回错误码
02800 * AUBO_BUSY
02801 * AUBO_BAD_STATE
02802 * -AUBO_INVL_ARGUMENT
02803 * -AUBO_BAD_STATE
02804 *
02805 * @throws arcs::common_interface::AuboException
02806 *
02807 * @par Python 函数原型
02808 * setToolInertial(self: pyaubo_sdk.RobotConfig, arg0: float, arg1:
02809 * List[float], arg2: List[float]) -> int
02810 *
02811 * @par Lua 函数原型
02812 * setToolInertial(m: number, com: table, inertial: table) -> nil
02813 *
02814 * @par Lua 示例
02815 * setToolInertial(3, {0,0,0}, {0,0,0}, {0,0,0,0,0,0,0,0})
02816 *
02817 * \endchinese
02818 * \english
02819 * Set tool mass, center of mass, and inertia.
02820 *
02821 * @param m Tool mass
02822 * @param com Center of mass

```



```

02823     * @param inertial Inertia
02824     *
02825     * @return Returns 0 on success; error code on failure
02826     * AUBO_BUSY
02827     * AUBO_BAD_STATE
02828     * -AUBO_INVL_ARGUMENT
02829     * -AUBO_BAD_STATE
02830     *
02831     * @throws arcs::common_interface::AuboException
02832     *
02833     * @par Python function prototype
02834     * setToolInertial(self: pyaubo_sdk.RobotConfig, arg0: float, arg1:
02835     * List[float], arg2: List[float]) -> int
02836     *
02837     * @par Lua function prototype
02838     * setToolInertial(m: number, com: table, inertial: table) -> nil
02839     *
02840     * @par Lua example
02841     * setToolInertial(3, {0,0,0}, {0,0,0}, {0,0,0,0,0,0,0,0})
02842     *
02843     * \endenglish
02844     */
02845 int setToolInertial(double m, const std::vector<double> &com,
02846                   const std::vector<double> &inertial);
02847
02848 /**
02849  * @ingroup RobotConfig
02850  * \chinese
02851  * 设置有效负载
02852  *
02853  * @param m 质量, 单位: kg
02854  * @param cog 重心, 单位: m, 形式为 (CoGx, CoGy, CoGz)
02855  * @param aom 力矩轴的方向, 单位: rad, 形式为 (rx, ry, rz)
02856  * @param inertia 惯量, 单位: kg*m^2, 形式为 (Ixx, Iyy, Izz, Ixy, Ixz,
02857  * Iyz) 或 (Ixx, Ixy, Ixz, Iyx, Iyy, Iyz, Izx, Izy, Izz)
02858  *
02859  * @return 成功返回 0; 失败返回错误码
02860  * AUBO_BUSY
02861  * AUBO_BAD_STATE
02862  * -AUBO_INVL_ARGUMENT
02863  * -AUBO_BAD_STATE
02864  *
02865  * @throws arcs::common_interface::AuboException
02866  *
02867  * @par Python 函数原型
02868  * setPayload(self: pyaubo_sdk.RobotConfig, arg0: float, arg1: List[float],
02869  * arg2: List[float], arg3: List[float]) -> int
02870  *
02871  * @par Lua 函数原型
02872  * setPayload(m: number, cog: table, aom: table, inertia: table) -> nil
02873  *
02874  * @par Lua 示例
02875  * setPayload(3, {0,0,0}, {0,0,0}, {0,0,0,0,0,0,0,0})
02876  *
02877  * @par JSON-RPC 请求示例
02878  * {"jsonrpc": "2.0", "method": "robl.RobotConfig.setPayload", "params": [3, [0,0,0], [0,0,0],
02879  * [0,0,0,0,0,0,0,0]], "id": 1}
02880  *
02881  * @par JSON-RPC 响应示例
02882  * {"id": 1, "jsonrpc": "2.0", "result": 0}
02883  * \endchinese
02884  * \english
02885  * Set the payload.
02886  *
02887  * @param m Mass, unit: kg
02888  * @param cog Center of gravity, unit: m, format (CoGx, CoGy, CoGz)
02889  * @param aom Axis of moment, unit: rad, format (rx, ry, rz)
02890  * @param inertia Inertia, unit: kg*m^2, format (Ixx, Iyy, Izz, Ixy, Ixz,
02891  * Iyz) or (Ixx, Ixy, Ixz, Iyx, Iyy, Iyz, Izx, Izy, Izz)
02892  *
02893  * @return Returns 0 on success; error code on failure
02894  * AUBO_BUSY
02895  * AUBO_BAD_STATE
02896  * -AUBO_INVL_ARGUMENT
02897  * -AUBO_BAD_STATE
02898  *
02899  * @throws arcs::common_interface::AuboException
02900  *
02901  * @par Python function prototype
02902  * setPayload(self: pyaubo_sdk.RobotConfig, arg0: float, arg1: List[float],
02903  * arg2: List[float], arg3: List[float]) -> int
02904  *
02905  * @par Lua function prototype
02906  * setPayload(m: number, cog: table, aom: table, inertia: table) -> nil
02907  *
02908  * @par Lua example
02909  * setPayload(3, {0,0,0}, {0,0,0}, {0,0,0,0,0,0,0,0})

```

```

02909      *
02910      * @par JSON-RPC request example
02911      * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.setPayload", "params": [3, [0,0,0], [0,0,0],
[0,0,0,0,0,0,0,0,0]], "id": 1}
02912      *
02913      * @par JSON-RPC response example
02914      * {"id": 1, "jsonrpc": "2.0", "result": 0}
02915      * \endenglish
02916      */
02917      int setPayload(double m, const std::vector<double> &cog,
02918                    const std::vector<double> &aom,
02919                    const std::vector<double> &inertia);
02920
02921      /**
02922      * @ingroup RobotConfig
02923      * \chinese
02924      * 获取有效负载
02925      *
02926      * @return 有效负载.
02927      * 第一个元素表示质量, 单位: kg;
02928      * 第二个元素表示重心, 单位: m, 形式为 (CoGx, CoGy, CoGz);
02929      * 第三个元素表示力矩轴的方向, 单位: rad, 形式为 (rx, ry, rz);
02930      * 第四个元素表示惯量, 单位: kg*m^2, 形式为 (Ixx, Iyy, Izz, Ixy, Ixz, Iyz)
02931      *
02932      * @throws arcs::common_interface::AuboException
02933      *
02934      * @par Python 函数原型
02935      * getPayload(self: pyaubo_sdk.RobotConfig) -> Tuple[float, List[float],
02936      * List[float], List[float]]
02937      *
02938      * @par Lua 函数原型
02939      * getPayload() -> number, table, table, table
02940      *
02941      * @par Lua 示例
02942      * m, cog, aom, inertia = getPayload()
02943      *
02944      * @par JSON-RPC 请求示例
02945      * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getPayload", "params": [], "id": 1}
02946      *
02947      * @par JSON-RPC 响应示例
02948      * {"id": 1, "jsonrpc": "2.0", "result": [3.0, [0.0,0.0,0.0], [0.0,0.0,0.0], [0.0,0.0,0.0,0.0,0.0,0.0]]}
02949      * \endchinese
02950      * \english
02951      * Get the payload.
02952      *
02953      * @return The payload.
02954      * The first element is the mass in kg;
02955      * The second element is the center of gravity in meters, format (CoGx,
02956      * CoGy, CoGz); The third element is the axis of moment in radians, format
02957      * (rx, ry, rz); The fourth element is the inertia in kg*m^2, format (Ixx,
02958      * Iyy, Izz, Ixy, Ixz, Iyz)
02959      *
02960      * @throws arcs::common_interface::AuboException
02961      *
02962      * @par Python function prototype
02963      * getPayload(self: pyaubo_sdk.RobotConfig) -> Tuple[float, List[float],
02964      * List[float], List[float]]
02965      *
02966      * @par Lua function prototype
02967      * getPayload() -> number, table, table, table
02968      *
02969      * @par Lua example
02970      * m, cog, aom, inertia = getPayload()
02971      *
02972      * @par JSON-RPC request example
02973      * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getPayload", "params": [], "id": 1}
02974      *
02975      * @par JSON-RPC response example
02976      * {"id": 1, "jsonrpc": "2.0", "result": [3.0, [0.0,0.0,0.0], [0.0,0.0,0.0], [0.0,0.0,0.0,0.0,0.0,0.0]]}
02977      * \endenglish
02978      */
02979      Payload getPayload();
02980
02981      /**
02982      * @ingroup RobotConfig
02983      * \chinese
02984      * 末端位姿是否在安全范围之内
02985      *
02986      * @param pose 末端位姿
02987      * @return 在安全范围内返回 true; 反之返回 false
02988      *
02989      * @throws arcs::common_interface::AuboException
02990      *
02991      * @par Python 函数原型
02992      * toolSpaceInRange(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> bool
02993      *
02994      * @par Lua 函数原型

```

```

02995     * toolSpaceInRange(pose: table) -> boolean
02996     *
02997     * @par Lua 示例
02998     * SpaceInRange = toolSpaceInRange({0.58712,-0.15775, 0.48703, 2.76,
02999     * 0.344, 1.432})
03000     *
03001     * @par JSON-RPC 请求示例
03002     * {"jsonrpc":"2.0","method":"rob1.RobotConfig.toolSpaceInRange","params":[[0.58712,
03003     * -0.15775, 0.48703, 2.76, 0.344, 1.432]],"id":1}
03004     *
03005     * @par JSON-RPC 响应示例
03006     * {"id":1,"jsonrpc":"2.0","result":true}
03007     * \endchinese
03008     * \english
03009     * Whether the end-effector pose is within the safety range.
03010     *
03011     * @param pose End-effector pose
03012     * @return Returns true if within the safety range; otherwise returns false
03013     *
03014     * @throws arcs::common_interface::AuboException
03015     *
03016     * @par Python function prototype
03017     * toolSpaceInRange(self: pyaubo_sdk.RobotConfig, arg0: List[float]) -> bool
03018     *
03019     * @par Lua function prototype
03020     * toolSpaceInRange(pose: table) -> boolean
03021     *
03022     * @par Lua example
03023     * SpaceInRange = toolSpaceInRange({0.58712,-0.15775, 0.48703, 2.76,
03024     * 0.344, 1.432})
03025     *
03026     * @par JSON-RPC request example
03027     * {"jsonrpc":"2.0","method":"rob1.RobotConfig.toolSpaceInRange","params":[[0.58712,
03028     * -0.15775, 0.48703, 2.76, 0.344, 1.432]],"id":1}
03029     *
03030     * @par JSON-RPC response example
03031     * {"id":1,"jsonrpc":"2.0","result":true}
03032     * \endenglish
03033     */
03034 bool toolSpaceInRange(const std::vector<double> &pose);
03035
03036 /**
03037  * @ingroup RobotConfig
03038  * \chinese
03039  * 发起固件升级请求，控制器软件将进入固件升级模式
03040  *
03041  * @param fw 固件升级路径。该路径的格式为：固件安装包路径 \# 升级节点列表。
03042  * 其中固件安装包路径和升级节点列表以井号 (#) 分隔，升级节点以逗号 (,) 分隔。
03043  * 如果节点名称后带有!，则表示强制（不带版本校验）升级该节点；
03044  * 反之，则表示带校验版本本地升级节点，
03045  * 即在升级该节点前，会先判断当前版本和目标版本是否相同，如果相同就不升级该节点。 \n
03046  * 可以根据实际需求灵活设置需要升级的节点。 \n
03047  * 例如，
03048  * /tmp/firmware_update-1.0.42-rc.5+2347b0d.firm\#master_mcu!,slace_mcu!,
03049  * base!,tool!,joint1!,joint2!,joint3!,joint4!,joint5!,joint6!
03050  * 表示强制升级接口板主板、接口板从板、基座、工具和 6 个关节 (joint1 至 joint6)。 \n
03051  * all 表示所有的节点，例如
03052  * /tmp/firm_XXX.firm\#all 表示带校验版本本地升级全部节点，
03053  * /tmp/firm_XXX.firm\#all! 表示强制升级全部节点
03054  *
03055  * @return 指令下发成功返回 0；失败返回错误码。 \n
03056  * -AUBO_BAD_STATE: 运行时 (RuntimeMachine) 的当前状态不是 Stopped，
03057  * 固件升级请求被拒绝。 AUBO_BAD_STATE 的值是 1。 \n
03058  * -AUBO_TIMEOUT: 超时。 AUBO_TIMEOUT 的值是 4。 \n
03059  *
03060  * @throws arcs::common_interface::AuboException
03061  *
03062  * @par Python 函数原型
03063  * firmwareUpdate(self: pyaubo_sdk.RobotConfig, arg0: str) -> int
03064  *
03065  * @par Lua 函数原型
03066  * firmwareUpdate(fw: string) -> nil
03067  *
03068  * @par Lua 示例
03069  * firmwareUpdate("/tmp/firmware_update-1.0.42-rc.12+3e33eac.firm#master_mcu")
03070  *
03071  * @par JSON-RPC 请求示例
03072  * {"jsonrpc":"2.0","method":"rob1.RobotConfig.firmwareUpdate","params":["/tmp/
firmware_update-1.0.42-rc.12+3e33eac.firm#master_mcu"],"id":1}
03073  *
03074  * @par JSON-RPC 响应示例
03075  * {"id":1,"jsonrpc":"2.0","result":0}
03076  * \endchinese
03077  * \english
03078  * Initiate a firmware upgrade request. The controller software will enter
03079  * firmware upgrade mode.
03080  *

```

```

03081 * @param fw Firmware upgrade path. The format is: firmware package
03082 * path/#upgrade node list. The firmware package path and node list are
03083 * separated by a hash (#), and nodes are separated by commas (,). If a node
03084 * name ends with !, it means to force upgrade (without version check) that
03085 * node; otherwise, the node will be upgraded only if the current version
03086 * differs from the target version. You can flexibly specify which nodes to
03087 * upgrade as needed. For example,
03088 * /tmp/
firmware_update-1.0.42-rc.5+2347b0d.firm#master_mcu!,slace_mcu!,base!,tool!,joint1!,joint2!,joint3!,joint4!,joint5!,joint6!
03089 * means to force upgrade the interface board main, interface board slave,
03090 * base, tool, and joints 1-6. "all" means all nodes, e.g.
03091 * /tmp/firm_XXX.firm#all means upgrade all nodes with version check,
03092 * /tmp/firm_XXX.firm#all! means force upgrade all nodes.
03093 *
03094 * @return Returns 0 if the command is sent successfully; otherwise, returns
03095 * an error code. -AUBO_BAD_STATE: The current state of the RuntimeMachine
03096 * is not Stopped, so the firmware upgrade request is rejected.
03097 * AUBO_BAD_STATE = 1. -AUBO_TIMEOUT: Timeout. AUBO_TIMEOUT = 4.
03098 *
03099 * @throws arcs::common_interface::AuboException
03100 *
03101 * @par Python function prototype
03102 * firmwareUpdate(self: pyaubo_sdk.RobotConfig, arg0: str) -> int
03103 *
03104 * @par Lua function prototype
03105 * firmwareUpdate(fw: string) -> nil
03106 *
03107 * @par Lua example
03108 * firmwareUpdate("/tmp/firmware_update-1.0.42-rc.12+3e33eac.firm#master_mcu")
03109 *
03110 * @par JSON-RPC request example
03111 * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.firmwareUpdate", "params": ["/tmp/
firmware_update-1.0.42-rc.12+3e33eac.firm#master_mcu"], "id": 1}
03112 *
03113 * @par JSON-RPC response example
03114 * {"id": 1, "jsonrpc": "2.0", "result": 0}
03115 * \endenglish
03116 */
03117 int firmwareUpdate(const std::string &fw);
03118
03119 /**
03120 * @ingroup RobotConfig
03121 * \chinese
03122 * 获取当前的固件升级的进程
03123 *
03124 * @return 当前的固件升级进程。 \n
03125 * 第一个元素表示步骤名称。如果是 failed, 则表示固件升级失败 \n
03126 * 第二个元素表示升级的进度 (0~1), 完成之后, 返回 ("", 1)
03127 *
03128 * @throws arcs::common_interface::AuboException
03129 *
03130 * @par Python 函数原型
03131 * getFirmwareUpdateProcess(self: pyaubo_sdk.RobotConfig) -> Tuple[str,
03132 * float]
03133 *
03134 * @par Lua 函数原型
03135 * getFirmwareUpdateProcess() -> table
03136 *
03137 * @par Lua 示例
03138 * step,progress = getFirmwareUpdateProcess()
03139 *
03140 * @par JSON-RPC 请求示例
03141 * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getFirmwareUpdateProcess", "params": [], "id": 1}
03142 *
03143 * @par JSON-RPC 响应示例
03144 * {"id": 1, "jsonrpc": "2.0", "result": ["", 0.0]}
03145 *
03146 * \endchinese
03147 * \english
03148 * Get the current firmware upgrade process.
03149 *
03150 * @return The current firmware upgrade process. \n
03151 * The first element is the step name. If it is "failed", the firmware
03152 * upgrade failed. \n The second element is the upgrade progress (0~1). When
03153 * finished, returns ("", 1)
03154 *
03155 * @throws arcs::common_interface::AuboException
03156 *
03157 * @par Python function prototype
03158 * getFirmwareUpdateProcess(self: pyaubo_sdk.RobotConfig) -> Tuple[str,
03159 * float]
03160 *
03161 * @par Lua function prototype
03162 * getFirmwareUpdateProcess() -> table
03163 *
03164 * @par Lua example
03165 * step,progress = getFirmwareUpdateProcess()

```

```

03166      *
03167      * @par JSON-RPC request example
03168      * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getFirmwareUpdateProcess","params":[],"id":1}
03169      *
03170      * @par JSON-RPC response example
03171      * {"id":1,"jsonrpc":"2.0","result":[" ",0.0]}
03172      *
03173      * \endenglish
03174      */
03175      std::tuple<std::string, double> getFirmwareUpdateProcess();
03176
03177      /**
03178      * @ingroup RobotConfig
03179      * \chinese
03180      * 获取关节最大位置（当前正在使用的限制值）
03181      *
03182      * @return 返回关节最大位置（当前正在使用的限制值）
03183      *
03184      * @throws arcs::common_interface::AuboException
03185      *
03186      * @par Python 函数原型
03187      * getLimitJointMaxPositions(self: pyaubo_sdk.RobotConfig) -> List[float]
03188      *
03189      * @par Lua 函数原型
03190      * getLimitJointMaxPositions() -> table
03191      *
03192      * @par Lua 示例
03193      * LimitJointMaxPositions = getLimitJointMaxPositions()
03194      *
03195      * @par JSON-RPC 请求示例
03196      * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getLimitJointMaxPositions","params":[],"id":1}
03197      *
03198      * @par JSON-RPC 响应示例
03199      * {"id":1,"jsonrpc":"2.0","result":
03200      * [6.2831854820251465,6.2831854820251465,6.2831854820251465,6.2831854820251465,6.2831854820251465,6.2831854820251465]}
03201      * \endchinese
03202      * \english
03203      * Get the joint maximum positions (currently used limit values).
03204      *
03205      * @return Returns the joint maximum positions (currently used limit
03206      * values).
03207      *
03208      * @throws arcs::common_interface::AuboException
03209      *
03210      * @par Python function prototype
03211      * getLimitJointMaxPositions(self: pyaubo_sdk.RobotConfig) -> List[float]
03212      *
03213      * @par Lua function prototype
03214      * getLimitJointMaxPositions() -> table
03215      *
03216      * @par Lua example
03217      * LimitJointMaxPositions = getLimitJointMaxPositions()
03218      *
03219      * @par JSON-RPC request example
03220      * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getLimitJointMaxPositions","params":[],"id":1}
03221      *
03222      * @par JSON-RPC response example
03223      * {"id":1,"jsonrpc":"2.0","result":
03224      * [6.2831854820251465,6.2831854820251465,6.2831854820251465,6.2831854820251465,6.2831854820251465,6.2831854820251465]}
03225      * \endenglish
03226      */
03227      std::vector<double> getLimitJointMaxPositions();
03228
03229      /**
03230      * @ingroup RobotConfig
03231      * \chinese
03232      * 获取关节最小位置（当前正在使用的限制值）
03233      *
03234      * @return 返回关节最小位置（当前正在使用的限制值）
03235      *
03236      * @throws arcs::common_interface::AuboException
03237      *
03238      * @par Python 函数原型
03239      * getLimitJointMinPositions(self: pyaubo_sdk.RobotConfig) -> List[float]
03240      *
03241      * @par Lua 函数原型
03242      * getLimitJointMinPositions() -> table
03243      *
03244      * @par Lua 示例
03245      * LimitJointMinPositions = getLimitJointMinPositions()
03246      *
03247      * @par JSON-RPC 请求示例
03248      * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getLimitJointMinPositions","params":[],"id":1}
03249      *
03250      * @par JSON-RPC 响应示例
03251      * {"id":1,"jsonrpc":"2.0","result":
03252      * [-6.2831854820251465,-6.2831854820251465,-6.2831854820251465,-6.2831854820251465,-6.2831854820251465,-6.2831854820251465]}

```

```

03250     * \endchinese
03251     * \english
03252     * Get the joint minimum positions (currently used limit values).
03253     *
03254     * @return Returns the joint minimum positions (currently used limit
03255     * values).
03256     *
03257     * @throws arcs::common_interface::AuboException
03258     *
03259     * @par Python function prototype
03260     * getLimitJointMinPositions(self: pyaubo_sdk.RobotConfig) -> List[float]
03261     *
03262     * @par Lua function prototype
03263     * getLimitJointMinPositions() -> table
03264     *
03265     * @par Lua example
03266     * LimitJointMinPositions = getLimitJointMinPositions()
03267     *
03268     * @par JSON-RPC request example
03269     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getLimitJointMinPositions", "params": [], "id": 1}
03270     *
03271     * @par JSON-RPC response example
03272     * {"id": 1, "jsonrpc": "2.0", "result":
[-6.2831854820251465, -6.2831854820251465, -6.2831854820251465, -6.2831854820251465, -6.2831854820251465, -6.2831854820251465]}
03273     * \endenglish
03274     */
03275     std::vector<double> getLimitJointMinPositions();
03276
03277     /**
03278     * @ingroup RobotConfig
03279     * \chinese
03280     * 获取关节最大速度（当前正在使用的限制值）
03281     *
03282     * @return 返回关节最大速度（当前正在使用的限制值）
03283     *
03284     * @throws arcs::common_interface::AuboException
03285     *
03286     * @par Python 函数原型
03287     * getLimitJointMaxSpeeds(self: pyaubo_sdk.RobotConfig) -> List[float]
03288     *
03289     * @par Lua 函数原型
03290     * getLimitJointMaxSpeeds() -> table
03291     *
03292     * @par Lua 示例
03293     * LimitJointMaxSpeeds = getLimitJointMaxSpeeds()
03294     *
03295     * @par JSON-RPC 请求示例
03296     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getLimitJointMaxSpeeds", "params": [], "id": 1}
03297     *
03298     * @par JSON-RPC 响应示例
03299     * {"id": 1, "jsonrpc": "2.0", "result":
[3.8920841217041016, 3.8920841217041016, 3.8920841217041016, 3.1066861152648926, 3.1066861152648926, 3.1066861152648926]}
03300     * \endchinese
03301     * \english
03302     * Get the joint maximum speeds (currently used limit values).
03303     *
03304     * @return Returns the joint maximum speeds (currently used limit values).
03305     *
03306     * @throws arcs::common_interface::AuboException
03307     *
03308     * @par Python function prototype
03309     * getLimitJointMaxSpeeds(self: pyaubo_sdk.RobotConfig) -> List[float]
03310     *
03311     * @par Lua function prototype
03312     * getLimitJointMaxSpeeds() -> table
03313     *
03314     * @par Lua example
03315     * LimitJointMaxSpeeds = getLimitJointMaxSpeeds()
03316     *
03317     * @par JSON-RPC request example
03318     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.getLimitJointMaxSpeeds", "params": [], "id": 1}
03319     *
03320     * @par JSON-RPC response example
03321     * {"id": 1, "jsonrpc": "2.0", "result":
[3.8920841217041016, 3.8920841217041016, 3.8920841217041016, 3.1066861152648926, 3.1066861152648926, 3.1066861152648926]}
03322     * \endenglish
03323     */
03324     std::vector<double> getLimitJointMaxSpeeds();
03325
03326     /**
03327     * @ingroup RobotConfig
03328     * \chinese
03329     * 获取关节最大加速度（当前正在使用的限制值）
03330     *
03331     * @return 返回关节最大加速度（当前正在使用的限制值）
03332     *
03333     * @throws arcs::common_interface::AuboException

```

```

03334 *
03335 * @par Python 函数原型
03336 * getLimitJointMaxAccelerations(self: pyaubo_sdk.RobotConfig) ->
03337 * List[float]
03338 *
03339 * @par Lua 函数原型
03340 * getLimitJointMaxAccelerations() -> table
03341 *
03342 * @par Lua 示例
03343 * LimitJointMaxAccelerations = getLimitJointMaxAccelerations()
03344 *
03345 * @par JSON-RPC 请求示例
03346 * {"jsonrpc": "2.0", "method": "robl.RobotConfig.getLimitJointMaxAccelerations", "params": [], "id": 1}
03347 *
03348 * @par JSON-RPC 响应示例
03349 * {"id": 1, "jsonrpc": "2.0", "result":
[31.104877758314785, 31.104877758314785, 31.104877758314785, 20.73625684294463, 20.73625684294463, 20.73625684294463]}
03350 * \endchinese
03351 * \english
03352 * Get the joint maximum accelerations (currently used limit values).
03353 *
03354 * @return Returns the joint maximum accelerations (currently used limit
03355 * values).
03356 *
03357 * @throws arcs::common_interface::AuboException
03358 *
03359 * @par Python function prototype
03360 * getLimitJointMaxAccelerations(self: pyaubo_sdk.RobotConfig) ->
03361 * List[float]
03362 *
03363 * @par Lua function prototype
03364 * getLimitJointMaxAccelerations() -> table
03365 *
03366 * @par Lua example
03367 * LimitJointMaxAccelerations = getLimitJointMaxAccelerations()
03368 *
03369 * @par JSON-RPC request example
03370 * {"jsonrpc": "2.0", "method": "robl.RobotConfig.getLimitJointMaxAccelerations", "params": [], "id": 1}
03371 *
03372 * @par JSON-RPC response example
03373 * {"id": 1, "jsonrpc": "2.0", "result":
[31.104877758314785, 31.104877758314785, 31.104877758314785, 20.73625684294463, 20.73625684294463, 20.73625684294463]}
03374 * \endenglish
03375 */
03376 std::vector<double> getLimitJointMaxAccelerations();
03377
03378 /**
03379 * @ingroup RobotConfig
03380 * \chinese
03381 * 获取 TCP 最大速度（当前正在使用的限制值）
03382 *
03383 * @return 返回 TCP 最大速度（当前正在使用的限制值）
03384 *
03385 * @throws arcs::common_interface::AuboException
03386 *
03387 * @par Python 函数原型
03388 * getLimitTcpMaxSpeed(self: pyaubo_sdk.RobotConfig) -> List[float]
03389 *
03390 * @par Lua 函数原型
03391 * getLimitTcpMaxSpeed() -> number
03392 *
03393 * @par Lua 示例
03394 * LimitTcpMaxSpeed = getLimitTcpMaxSpeed()
03395 *
03396 * @par JSON-RPC 请求示例
03397 * {"jsonrpc": "2.0", "method": "robl.RobotConfig.getLimitTcpMaxSpeed", "params": [], "id": 1}
03398 *
03399 * @par JSON-RPC 响应示例
03400 * {"id": 1, "jsonrpc": "2.0", "result": 2.0}
03401 * \endchinese
03402 * \english
03403 * Get the TCP maximum speed (currently used limit value).
03404 *
03405 * @return Returns the TCP maximum speed (currently used limit value).
03406 *
03407 * @throws arcs::common_interface::AuboException
03408 *
03409 * @par Python function prototype
03410 * getLimitTcpMaxSpeed(self: pyaubo_sdk.RobotConfig) -> List[float]
03411 *
03412 * @par Lua function prototype
03413 * getLimitTcpMaxSpeed() -> number
03414 *
03415 * @par Lua example
03416 * LimitTcpMaxSpeed = getLimitTcpMaxSpeed()
03417 *
03418 * @par JSON-RPC request example

```



```

03419     * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getLimitTcpMaxSpeed","params":[],"id":1}
03420     *
03421     * @par JSON-RPC response example
03422     * {"id":1,"jsonrpc":"2.0","result":2.0}
03423     * \endenglish
03424     */
03425 double getLimitTcpMaxSpeed();
03426
03427 /**
03428  * @ingroup RobotConfig
03429  * \chinese
03430  * 获取当前安全停止的类型
03431  *
03432  * @return 返回当前安全停止的类型
03433  *
03434  * @par JSON-RPC 请求示例
03435  * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getSafeguardStopType","params":[],"id":1}
03436  *
03437  * @par JSON-RPC 响应示例
03438  * {"id":1,"jsonrpc":"2.0","result":"None"}
03439  * \endchinese
03440  * \english
03441  * Get the current safeguard stop type.
03442  *
03443  * @return Returns the current safeguard stop type.
03444  *
03445  * @par JSON-RPC request example
03446  * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getSafeguardStopType","params":[],"id":1}
03447  *
03448  * @par JSON-RPC response example
03449  * {"id":1,"jsonrpc":"2.0","result":"None"}
03450  * \endenglish
03451  */
03452 SafeguardStopType getSafeguardStopType();
03453
03454 /**
03455  * @ingroup RobotConfig
03456  * \chinese
03457  * 按位获取完整的安全停止触发源
03458  *
03459  * @return 返回所有安全停止触发源
03460  *
03461  * 安全停止的原因:
03462  * 手动模式下可配置安全 IO 触发的安全停止 - 1<<0
03463  * 自动模式下可配置安全 IO 触发的安全停止 - 1<<1
03464  * 控制柜 SI 输入触发的安全停止 - 1<<2
03465  * 示教器三态开关触发的安全停止 - 1<<3
03466  * 自动切手动触发的安全停止 - 1<<4
03467  *
03468  * @par JSON-RPC 请求示例
03469  * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getSafeguardStopSource","params":[],"id":1}
03470  *
03471  * @par JSON-RPC 响应示例
03472  * {"id":1,"jsonrpc":"2.0","result":0}
03473  * \endchinese
03474  * \english
03475  * Get the complete safeguard stop source as a bitmask.
03476  *
03477  * @return Returns all safeguard stop sources.
03478  *
03479  * Reasons for safeguard stop:
03480  * Safeguard stop triggered by configurable safety IO in manual mode - 1<<0
03481  * Safeguard stop triggered by configurable safety IO in automatic mode -
03482  * 1<<1 Safeguard stop triggered by control box SI input - 1<<2 Safeguard
03483  * stop triggered by teach pendant three-position switch - 1<<3 Safeguard
03484  * stop triggered by switching from auto to manual - 1<<4
03485  *
03486  * @par JSON-RPC request example
03487  * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getSafeguardStopSource","params":[],"id":1}
03488  *
03489  * @par JSON-RPC response example
03490  * {"id":1,"jsonrpc":"2.0","result":0}
03491  * \endenglish
03492  */
03493 int getSafeguardStopSource();
03494
03495 /**
03496  * @ingroup RobotConfig
03497  * \chinese
03498  * 按位获取完整的机器人紧急停止触发源
03499  *
03500  * @return 返回所有机器人紧急停止触发源
03501  *
03502  * 紧急停止的原因:
03503  * 控制柜紧急停止按钮触发 - 1<<0
03504  * 示教器紧急停止按钮触发 - 1<<1
03505  * 手柄紧急停止按钮触发 - 1<<2

```



```

03506 * 固定 IO 紧急停止触发 - 1<<3
03507 *
03508 * @par JSON-RPC 请求示例
03509 * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getRobotEmergencyStopSource","params":[],"id":1}
03510 *
03511 * @par JSON-RPC 响应示例
03512 * {"id":1,"jsonrpc":"2.0","result":0}
03513 * \endchinese
03514 * \english
03515 * Get the complete robot emergency stop source as a bitmask.
03516 *
03517 * @return Returns all robot emergency stop sources.
03518 *
03519 * Reasons for emergency stop:
03520 * Emergency stop triggered by control box button - 1<<0
03521 * Emergency stop triggered by teach pendant button - 1<<1
03522 * Emergency stop triggered by handle button - 1<<2
03523 * Emergency stop triggered by fixed IO - 1<<3
03524 *
03525 * @par JSON-RPC request example
03526 * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getRobotEmergencyStopSource","params":[],"id":1}
03527 *
03528 * @par JSON-RPC response example
03529 * {"id":1,"jsonrpc":"2.0","result":0}
03530 * \endenglish
03531 */
03532 int getRobotEmergencyStopSource();
03533
03534 /**
03535 * @ingroup RobotConfig
03536 * \chinese
03537 * 获取工具端力矩传感器的名字
03538 *
03539 * @return 当前工具端力矩传感器名字
03540 *
03541 * @throws arcs::common_interface::AuboException
03542 *
03543 * @par Python 函数原型
03544 * getSelectedTcpForceSensorName(self: pyaubo_sdk.RobotConfig) -> str
03545 *
03546 * @par Lua 函数原型
03547 * getSelectedTcpForceSensorName() -> str
03548 *
03549 * @par Lua 示例
03550 * TcpForceSensorName = getSelectedTcpForceSensorName()
03551 *
03552 * @par JSON-RPC 请求示例
03553 * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getSelectedTcpForceSensorName","params":[],"id":1}
03554 * \endchinese
03555 * \english
03556 * Get the name of the selected TCP force sensor.
03557 *
03558 * @return The name of the currently selected TCP force sensor.
03559 *
03560 * @throws arcs::common_interface::AuboException
03561 *
03562 * @par Python function prototype
03563 * getSelectedTcpForceSensorName(self: pyaubo_sdk.RobotConfig) -> str
03564 *
03565 * @par Lua function prototype
03566 * getSelectedTcpForceSensorName() -> str
03567 *
03568 * @par Lua example
03569 * TcpForceSensorName = getSelectedTcpForceSensorName()
03570 *
03571 * @par JSON-RPC request example
03572 * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getSelectedTcpForceSensorName","params":[],"id":1}
03573 * \endenglish
03574 */
03575 std::string getSelectedTcpForceSensorName();
03576
03577 /**
03578 * @ingroup RobotConfig
03579 * \chinese
03580 * 添加焊枪动力学参数
03581 *
03582 * @param m 质量, 单位: kg
03583 * @param cog 质心, 单位: m, 形式为 (CoGx, CoGy, CoGz)
03584 * @param inertia 惯性张量, 单位: kg*m^2, 形式为 (Ixx, Iyy, Izz, Ixy,
03585 * Ixz, Iyz) 或 (Ixx, Ixy, Ixz, Iyx, Iyy, Iyz, Izx, Izy, Izz)
03586 * @return 成功返回 0; 失败返回错误码
03587 *
03588 * @throws arcs::common_interface::AuboException
03589 *
03590 * @par Python 函数原型
03591 * attachWeldingGun(self: pyaubo_sdk.RobotConfig, arg0: float, arg1:
03592 * List[float], arg2: List[float]) -> int

```

```

03593     *
03594     * @par Lua 函数原型
03595     * attachWeldingGun(m: number, cog: table, inertia: table) -> nil
03596     *
03597     * @par Lua 示例
03598     * attachWeldingGun(3, {0,0,0}, {0,0,0,0,0,0,0,0})
03599     *
03600     * @par JSON-RPC 请求示例
03601     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.attachWeldingGun", "params": [3, [0,0,0],
[0,0,0,0,0,0,0,0,0]], "id": 1}
03602     *
03603     * @par JSON-RPC 响应示例
03604     * {"id": 1, "jsonrpc": "2.0", "result": 0}
03605     * \endchinese
03606     */
03607 int attachWeldingGun(double m, const std::vector<double> &cog,
03608                     const std::vector<double> &inertia);
03609 /**
03610     * @ingroup RobotConfig
03611     * \chinese
03612     * 设置碰撞阈值
03613     *
03614     * @param threshold 碰撞阈值
03615     * @return 成功返回 0；失败返回错误码
03616     *
03617     * @throws arcs::common_interface::AuboException
03618     *
03619     * @par Python 函数原型
03620     * setCollisionThreshold(self: pyaubo_sdk.RobotConfig, arg0: List[float])
03621     * -> int
03622     *
03623     * @par Lua 函数原型
03624     * setCollisionThreshold(threshold: table) -> nil
03625     *
03626     * @par Lua 示例
03627     * setCollisionThreshold({99.0,
03628     * 114.0,
03629     * 103.0, 56.0, 51.0, 60.0,54.6111111, 66.22222222, 59.66666667, 28.94444444,
03630     * 27.05555556, 30.11111111,19.1, 28.0, 25.0, 7.3, 7.9, 6.2})
03631     *
03632     * @par JSON-RPC 请求示例
03633     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.setCollisionThreshold", "params": [[99.0,
03634     * 114.0,
03635     * 103.0, 56.0, 51.0, 60.0,54.6111111, 66.22222222, 59.66666667, 28.94444444,
03636     * 27.05555556, 30.11111111,19.1, 28.0, 25.0, 7.3, 7.9, 6.2]], "id": 1}
03637     *
03638     * @par JSON-RPC 响应示例
03639     * {"id": 1, "jsonrpc": "2.0", "result": 0}
03640     * \endchinese
03641     * \english
03642     * Enable optimized collision force.
03643     *
03644     * @param threshold
03645     * @return Returns 0 on success; error code on failure.
03646     *
03647     * @par JSON-RPC request example
03648     * {"jsonrpc": "2.0", "method": "rob1.RobotConfig.setCollisionThreshold", "params": [[99.0,
03649     * 114.0,
03650     * 103.0, 56.0, 51.0, 60.0,54.6111111, 66.22222222, 59.66666667, 28.94444444,
03651     * 27.05555556, 30.11111111,19.1, 28.0, 25.0, 7.3, 7.9, 6.2]], "id": 1}
03652     *
03653     * @par JSON-RPC response example
03654     * {"id": 1, "jsonrpc": "2.0", "result": "None"}
03655     * \endenglish
03656     */
03657 int setCollisionThreshold(const std::vector<double> &threshold);
03658
03659 /**
03660     * @ingroup RobotConfig
03661     * \chinese
03662     * 获取碰撞阈值
03663     *
03664     * @return 碰撞阈值
03665     *
03666     * @throws arcs::common_interface::AuboException
03667     *
03668     * @par Python 函数原型
03669     * getCollisionThreshold(self: pyaubo_sdk.RobotConfig) -> List[float]
03670     *
03671     * @par Lua 函数原型
03672     * getCollisionThreshold() -> table
03673     *
03674     * @par Lua 示例
03675     * CollisionThreshold = getCollisionThreshold()
03676     *
03677     * @par JSON-RPC 请求示例
03678     * {"jsonrpc": "2.0", "method": "rob1.MotionControl.getCollisionThreshold", "params": [99.0,

```

```

03679     * 114.0, 103.0, 56.0, 51.0, 60.0],"id":1}
03680     *
03681     * @par JSON-RPC 响应示例
03682     * {"id":1,"jsonrpc":"2.0","result":1}
03683     * \endchinese
03684     * \english
03685     * Get collision threshold.
03686     *
03687     * @return collision threshold.
03688     *
03689     * @throws arcs::common_interface::AuboException
03690     *
03691     * @par Python function prototype
03692     * getCollisionThreshold(self: pyaubo_sdk.RobotConfig) -> List[float]
03693     *
03694     * @par Lua function prototype
03695     * getCollisionThreshold() -> table
03696     *
03697     * @par Lua example
03698     * CollisionThreshold = getCollisionThreshold()
03699     *
03700     * @par JSON-RPC request example
03701     * {"jsonrpc":"2.0","method":"rob1.RobotConfig.getCollisionThreshold","params":[[99.0,
03702     * 114.0,
03703     * 103.0, 56.0, 51.0, 60.0,54.6111111, 66.22222222, 59.66666667, 28.94444444,
03704     * 27.05555556, 30.11111111,19.1, 28.0, 25.0, 7.3, 7.9, 6.2]],"id":1}
03705     *
03706     * @par JSON-RPC response example
03707     * {"id":1,"jsonrpc":"2.0","result":"None"}
03708     * \endenglish
03709     */
03710     std::vector<double> getCollisionThreshold();
03711
03712     int enableAxisGroup(const std::string &group_name);
03713     int disableAxisGroup();
03714
03715     /**
03716     * @ingroup RobotConfig
03717     * \chinese
03718     * 使能末端碰撞检测
03719     *
03720     * @param enable 使能碰撞阈值
03721     * @return 成功返回 0; 失败返回错误码
03722     *
03723     * @throws arcs::common_interface::AuboException
03724     *
03725     * @par Python 函数原型
03726     * enableEndCollisionCheck(self: pyaubo_sdk.RobotConfig, arg0: bool)
03727     * -> int
03728     *
03729     * @par Lua 函数原型
03730     * enableEndCollisionCheck(threshold: table) -> nil
03731     *
03732     * @par Lua 示例
03733     * enableEndCollisionCheck(true)
03734     *
03735     * @par JSON-RPC 请求示例
03736     * {"jsonrpc":"2.0","method":"rob1.RobotConfig.enableEndCollisionCheck","params":[true],"id":1}
03737     *
03738     * @par JSON-RPC 响应示例
03739     * {"id":1,"jsonrpc":"2.0","result":0}
03740     * \endchinese
03741     * \english
03742     * Set end collision threshold.
03743     *
03744     * @param threshold
03745     * @return Returns 0 on success; error code on failure.
03746     *
03747     * @par JSON-RPC request example
03748     * {"jsonrpc":"2.0","method":"rob1.RobotConfig.enableEndCollisionCheck","params":[true],"id":1}
03749     *
03750     * @par JSON-RPC response example
03751     * {"id":1,"jsonrpc":"2.0","result":"None"}
03752     * \endenglish
03753     */
03754     int enableEndCollisionCheck(bool enable);
03755
03756     /**
03757     * @ingroup RobotConfig
03758     * \chinese
03759     * 获取末端碰撞检测是否使能
03760     *
03761     * @return 是否使能末端碰撞检测
03762     *
03763     * @throws arcs::common_interface::AuboException
03764     *
03765     * @par Python 函数原型

```

```

03766     * isEndCollisionCheckEnabled(self: pyaubo_sdk.RobotConfig) -> List[float]
03767     *
03768     * @par Lua 函数原型
03769     * isEndCollisionCheckEnabled() -> table
03770     *
03771     * @par Lua 示例
03772     * IsEndCollisionCheckEnabled = isEndCollisionCheckEnabled()
03773     *
03774     * @par JSON-RPC 请求示例
03775     * {"jsonrpc": "2.0", "method": "robl.MotionControl.isEndCollisionCheckEnabled", "params": [], "id": 1}
03776     *
03777     * @par JSON-RPC 响应示例
03778     * {"id": 1, "jsonrpc": "2.0", "result": true}
03779     * \endchinese
03780     * \english
03781     * Get collision check is enabled.
03782     *
03783     * @return end collision threshold.
03784     *
03785     * @throws arcs::common_interface::AuboException
03786     *
03787     * @par Python function prototype
03788     * isEndCollisionCheckEnabled(self: pyaubo_sdk.RobotConfig) -> bool
03789     *
03790     * @par Lua function prototype
03791     * isEndCollisionCheckEnabled() -> bool
03792     *
03793     * @par Lua example
03794     * IsEndCollisionCheckEnabled = isEndCollisionCheckEnabled()
03795     *
03796     * @par JSON-RPC request example
03797     * {"jsonrpc": "2.0", "method": "robl.RobotConfig.isEndCollisionCheckEnabled", "params": [], "id": 1}
03798     *
03799     * @par JSON-RPC response example
03800     * {"id": 1, "jsonrpc": "2.0", "result": true}
03801     * \endenglish
03802     */
03803     bool isEndCollisionCheckEnabled();
03804
03805 protected:
03806     void *d_;
03807 };
03808 using RobotConfigPtr = std::shared_ptr<RobotConfig>;
03809
03810 } // namespace common_interface
03811 } // namespace arcs
03812 #endif // AUBO_SDK_ROBOT_CONFIG_H

```

12.30 include/aubo/robot/robot_manage.h 文件参考

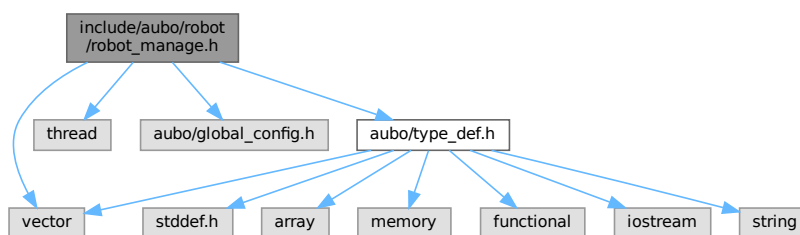
机器人管理接口，如上电、启动、拖动示教模式等

```

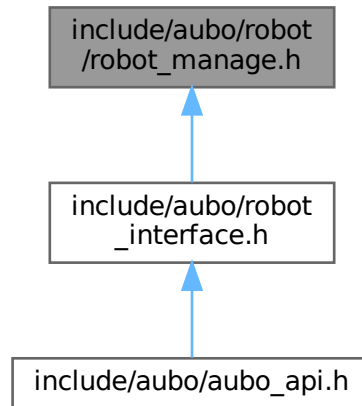
#include <vector>
#include <thread>
#include <aubo/global_config.h>
#include <aubo/type_def.h>

```

robot_manage.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::common_interface::RobotManage](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::common_interface](#)

类型定义

- using [arcs::common_interface::RobotManagePtr](#) = std::shared_ptr<[RobotManage](#)>

12.30.1 详细描述

机器人管理接口，如上电、启动、拖动示教模式等在文件 [robot_manage.h](#) 中定义.

12.31 robot__manage.h

[浏览该文件的文档.](#)

```

00001 /** @file robot_manage.h
00002  * @brief 机器人管理接口，如上电、启动、拖动示教模式等
00003  */
00004 #ifndef AUBO_SDK_ROBOT_CONTROL_INTERFACE_H
00005 #define AUBO_SDK_ROBOT_CONTROL_INTERFACE_H
00006
00007 #include <vector>
00008 #include <thread>
00009
00010 #include <aubo/global_config.h>
00011 #include <aubo/type_def.h>
00012
00013 namespace arcs {
00014 namespace common_interface {
00015
00016 /**
00017  * @defgroup RobotManage RobotManage (机器人生命周期管理)
00018  * @ingroup RobotInterface
00019  * RobotManage
00020  */
00021 class ARCS_ABI_EXPORT RobotManage
  
```

```

00022 {
00023 public:
00024     RobotManage();
00025     virtual ~RobotManage();
00026
00027     /**
00028      * @ingroup RobotManage
00029      * \chinese
00030      * 发起机器人上电请求
00031      *
00032      * @return 成功返回 0; 失败返回错误码
00033      * AUBO_BAD_STATE
00034      *
00035      * @throws arcs::common_interface::AuboException
00036      *
00037      * @par Python 函数原型
00038      * poweron(self: pyaubo_sdk.RobotManage) -> int
00039      *
00040      * @par Lua 函数原型
00041      * poweron() -> number
00042      *
00043      * @par Lua 示例
00044      * num = poweron()
00045      *
00046      * @par JSON-RPC 请求示例
00047      * {"jsonrpc": "2.0", "method": "rob1.RobotManage.poweron", "params": [], "id": 1}
00048      *
00049      * @par JSON-RPC 响应示例
00050      * {"id": 1, "jsonrpc": "2.0", "result": 0}
00051      *
00052      * @par C++ 实例
00053      * @code
00054      * auto rpc_cli = std::make_shared<RpcClient>();
00055      * auto robot_name = rpc_cli->getRobotNames().front();
00056      * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->poweron();
00057      * @endcode
00058      *
00059      * \endchinese
00060      *
00061      * \english
00062      * Initiate robot power-on request
00063      *
00064      * @return Returns 0 on success; error code on failure
00065      * AUBO_BAD_STATE
00066      *
00067      * @throws arcs::common_interface::AuboException
00068      *
00069      * @par Python function prototype
00070      * poweron(self: pyaubo_sdk.RobotManage) -> int
00071      *
00072      * @par Lua function prototype
00073      * poweron() -> number
00074      *
00075      * @par Lua example
00076      * num = poweron()
00077      *
00078      * @par JSON-RPC request example
00079      * {"jsonrpc": "2.0", "method": "rob1.RobotManage.poweron", "params": [], "id": 1}
00080      *
00081      * @par JSON-RPC response example
00082      * {"id": 1, "jsonrpc": "2.0", "result": 0}
00083      *
00084      * @par C++ example
00085      * @code
00086      * auto rpc_cli = std::make_shared<RpcClient>();
00087      * auto robot_name = rpc_cli->getRobotNames().front();
00088      * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->poweron();
00089      * @endcode
00090      * \endenglish
00091      */
00092     int poweron();
00093
00094     /**
00095      * @ingroup RobotManage
00096      * \chinese
00097      * 发起机器人启动请求
00098      *
00099      * @return 成功返回 0; 失败返回错误码
00100      * AUBO_BAD_STATE
00101      *
00102      * @throws arcs::common_interface::AuboException
00103      *
00104      * @par Python 函数原型
00105      * startup(self: pyaubo_sdk.RobotManage) -> int
00106      *
00107      * @par Lua 函数原型
00108      * startup() -> number

```

```

00109      *
00110      * @par Lua 示例
00111      * num = startup()
00112      *
00113      * @par JSON-RPC 请求示例
00114      * {"jsonrpc":"2.0","method":"rob1.RobotManage.startup","params":[],"id":1}
00115      *
00116      * @par JSON-RPC 响应示例
00117      * {"id":1,"jsonrpc":"2.0","result":0}
00118      *
00119      * @par C++ 示例
00120      * @code
00121      * auto rpc_cli = std::make_shared<RpcClient>();
00122      * auto robot_name = rpc_cli->getRobotNames().front();
00123      * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->startup();
00124      * @endcode
00125      * \endchinese
00126      *
00127      * \english
00128      * Initiate robot startup request
00129      *
00130      * @return Returns 0 on success; error code on failure
00131      * AUBO_BAD_STATE
00132      *
00133      * @throws arcs::common_interface::AuboException
00134      *
00135      * @par Python function prototype
00136      * startup(self: pyaubo_sdk.RobotManage) -> int
00137      *
00138      * @par Lua function prototype
00139      * startup() -> number
00140      *
00141      * @par Lua example
00142      * num = startup()
00143      *
00144      * @par JSON-RPC request example
00145      * {"jsonrpc":"2.0","method":"rob1.RobotManage.startup","params":[],"id":1}
00146      *
00147      * @par JSON-RPC response example
00148      * {"id":1,"jsonrpc":"2.0","result":0}
00149      *
00150      * @par C++ example
00151      * @code
00152      * auto rpc_cli = std::make_shared<RpcClient>();
00153      * auto robot_name = rpc_cli->getRobotNames().front();
00154      * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->startup();
00155      * @endcode
00156      * \endenglish
00157      */
00158      int startup();
00159
00160      /**
00161      * @ingroup RobotManage
00162      * \chinese
00163      * 发起机器人松开刹车请求
00164      *
00165      * @return 成功返回 0; 失败返回错误码
00166      * AUBO_BAD_STATE
00167      *
00168      * @throws arcs::common_interface::AuboException
00169      *
00170      * @par Python 函数原型
00171      * releaseRobotBrake(self: pyaubo_sdk.RobotManage) -> int
00172      *
00173      * @par Lua 函数原型
00174      * releaseRobotBrake() -> number
00175      *
00176      * @par Lua 示例
00177      * num = releaseRobotBrake()
00178      *
00179      * @par JSON-RPC 请求示例
00180      * {"jsonrpc":"2.0","method":"rob1.RobotManage.releaseRobotBrake","params":[],"id":1}
00181      *
00182      * @par JSON-RPC 响应示例
00183      * {"id":1,"jsonrpc":"2.0","result":0}
00184      *
00185      * @par C++ 示例
00186      * @code
00187      * auto rpc_cli = std::make_shared<RpcClient>();
00188      * auto robot_name = rpc_cli->getRobotNames().front();
00189      * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->releaseRobotBrake();
00190      * @endcode
00191      * \endchinese
00192      *
00193      * \english
00194      * Initiate robot brake release request
00195      *

```

```

00196     * @return Returns 0 on success; error code on failure
00197     * AUBO_BAD_STATE
00198     *
00199     * @throws arcs::common_interface::AuboException
00200     *
00201     * @par Python function prototype
00202     * releaseRobotBrake(self: pyaubo_sdk.RobotManage) -> int
00203     *
00204     * @par Lua function prototype
00205     * releaseRobotBrake() -> number
00206     *
00207     * @par Lua example
00208     * num = releaseRobotBrake()
00209     *
00210     * @par JSON-RPC request example
00211     * {"jsonrpc": "2.0", "method": "rob1.RobotManage.releaseRobotBrake", "params": [], "id": 1}
00212     *
00213     * @par JSON-RPC response example
00214     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00215     *
00216     * @par C++ example
00217     * @code
00218     * auto rpc_cli = std::make_shared<RpcClient>();
00219     * auto robot_name = rpc_cli->getRobotNames().front();
00220     * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->releaseRobotBrake();
00221     * @endcode
00222     * \endenglish
00223     */
00224     int releaseRobotBrake();
00225
00226     /**
00227     * @ingroup RobotManage
00228     * \chinese
00229     * 发起机器人刹车请求
00230     *
00231     * @return 成功返回 0; 失败返回错误码
00232     * AUBO_BAD_STATE
00233     *
00234     * @throws arcs::common_interface::AuboException
00235     *
00236     * @par Python 函数原型
00237     * lockRobotBrake(self: pyaubo_sdk.RobotManage) -> int
00238     *
00239     * @par Lua 函数原型
00240     * lockRobotBrake() -> number
00241     *
00242     * @par Lua 示例
00243     * num = lockRobotBrake()
00244     *
00245     * @par JSON-RPC 请求示例
00246     * {"jsonrpc": "2.0", "method": "rob1.RobotManage.lockRobotBrake", "params": [], "id": 1}
00247     *
00248     * @par JSON-RPC 响应示例
00249     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00250     *
00251     * @par C++ 示例
00252     * @code
00253     * auto rpc_cli = std::make_shared<RpcClient>();
00254     * auto robot_name = rpc_cli->getRobotNames().front();
00255     * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->lockRobotBrake();
00256     * @endcode
00257     * \endchinese
00258     *
00259     * \english
00260     * Initiate robot brake lock request
00261     *
00262     * @return Returns 0 on success; error code on failure
00263     * AUBO_BAD_STATE
00264     *
00265     * @throws arcs::common_interface::AuboException
00266     *
00267     * @par Python function prototype
00268     * lockRobotBrake(self: pyaubo_sdk.RobotManage) -> int
00269     *
00270     * @par Lua function prototype
00271     * lockRobotBrake() -> number
00272     *
00273     * @par Lua example
00274     * num = lockRobotBrake()
00275     *
00276     * @par JSON-RPC request example
00277     * {"jsonrpc": "2.0", "method": "rob1.RobotManage.lockRobotBrake", "params": [], "id": 1}
00278     *
00279     * @par JSON-RPC response example
00280     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00281     *
00282     * @par C++ example

```



```

00283     * @code
00284     * auto rpc_cli = std::make_shared<RpcClient>();
00285     * auto robot_name = rpc_cli->getRobotNames().front();
00286     * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->lockRobotBrake();
00287     * @endcode
00288     * \endenglish
00289     */
00290 int lockRobotBrake();
00291
00292 /**
00293  * @ingroup RobotManage
00294  * \chinese
00295  * 发起机器人断电请求
00296  *
00297  * @return 成功返回 0; 失败返回错误码
00298  * AUBO_BAD_STATE
00299  *
00300  * @throws arcs::common_interface::AuboException
00301  *
00302  * @par Python 函数原型
00303  * poweroff(self: pyaubo_sdk.RobotManage) -> int
00304  *
00305  * @par Lua 函数原型
00306  * poweroff() -> number
00307  *
00308  * @par Lua 示例
00309  * num = poweroff()
00310  *
00311  * @par JSON-RPC 请求示例
00312  * {"jsonrpc": "2.0", "method": "rob1.RobotManage.poweroff", "params": [], "id": 1}
00313  *
00314  * @par JSON-RPC 响应示例
00315  * {"id": 1, "jsonrpc": "2.0", "result": 0}
00316  *
00317  * @par C++ 示例
00318  * @code
00319  * auto rpc_cli = std::make_shared<RpcClient>();
00320  * auto robot_name = rpc_cli->getRobotNames().front();
00321  * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->poweroff();
00322  * @endcode
00323  * \endchinese
00324  *
00325  * \english
00326  * Initiate robot power-off request
00327  *
00328  * @return Returns 0 on success; error code on failure
00329  * AUBO_BAD_STATE
00330  *
00331  * @throws arcs::common_interface::AuboException
00332  *
00333  * @par Python function prototype
00334  * poweroff(self: pyaubo_sdk.RobotManage) -> int
00335  *
00336  * @par Lua function prototype
00337  * poweroff() -> number
00338  *
00339  * @par Lua example
00340  * num = poweroff()
00341  *
00342  * @par JSON-RPC request example
00343  * {"jsonrpc": "2.0", "method": "rob1.RobotManage.poweroff", "params": [], "id": 1}
00344  *
00345  * @par JSON-RPC response example
00346  * {"id": 1, "jsonrpc": "2.0", "result": 0}
00347  *
00348  * @par C++ example
00349  * @code
00350  * auto rpc_cli = std::make_shared<RpcClient>();
00351  * auto robot_name = rpc_cli->getRobotNames().front();
00352  * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->poweroff();
00353  * @endcode
00354  * \endenglish
00355  */
00356 int poweroff();
00357
00358 /**
00359  * @ingroup RobotManage
00360  * \chinese
00361  * 发起机器人反向驱动请求
00362  *
00363  * @param enable
00364  *
00365  * @return 成功返回 0; 失败返回错误码
00366  * AUBO_BUSY
00367  * AUBO_BAD_STATE
00368  * -AUBO_INVL_ARGUMENT
00369  * -AUBO_BAD_STATE

```

```

00370     *
00371     * @throws arcs::common_interface::AuboException
00372     *
00373     * @par Python 函数原型
00374     * backdrive(self: pyaubo_sdk.RobotManage, arg0: bool) -> int
00375     *
00376     * @par Lua 函数原型
00377     * backdrive(enable: boolean) -> number
00378     *
00379     * @par Lua 示例
00380     * num = backdrive(false)
00381     *
00382     * @par JSON-RPC 请求示例
00383     * {"jsonrpc": "2.0", "method": "rob1.RobotManage.backdrive", "params": [false], "id": 1}
00384     *
00385     * @par JSON-RPC 响应示例
00386     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00387     *
00388     * @par C++ 示例
00389     * @code
00390     * auto rpc_cli = std::make_shared<RpcClient>();
00391     * auto robot_name = rpc_cli->getRobotNames().front();
00392     * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->backdrive(true);
00393     * @endcode
00394     * \endchinese
00395     *
00396     * \english
00397     * Initiate robot backdrive request
00398     *
00399     * @param enable
00400     *
00401     * @return Returns 0 on success; error code on failure
00402     * AUBO_BUSY
00403     * AUBO_BAD_STATE
00404     * -AUBO_INVL_ARGUMENT
00405     * -AUBO_BAD_STATE
00406     *
00407     * @throws arcs::common_interface::AuboException
00408     *
00409     * @par Python function prototype
00410     * backdrive(self: pyaubo_sdk.RobotManage, arg0: bool) -> int
00411     *
00412     * @par Lua function prototype
00413     * backdrive(enable: boolean) -> number
00414     *
00415     * @par Lua example
00416     * num = backdrive(false)
00417     *
00418     * @par JSON-RPC request example
00419     * {"jsonrpc": "2.0", "method": "rob1.RobotManage.backdrive", "params": [false], "id": 1}
00420     *
00421     * @par JSON-RPC response example
00422     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00423     *
00424     * @par C++ example
00425     * @code
00426     * auto rpc_cli = std::make_shared<RpcClient>();
00427     * auto robot_name = rpc_cli->getRobotNames().front();
00428     * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->backdrive(true);
00429     * @endcode
00430     * \endenglish
00431     */
00432     int backdrive(bool enable);
00433
00434     /**
00435     * @ingroup RobotManage
00436     * \chinese
00437     * 发起机器人自由驱动请求
00438     * 接口在软件版本 0.31.x 后已废弃, 使用 handguideMode 接口替换
00439     * handguideMode({1,1,1,1,1}, {0,0,0,0,0,0})
00440     *
00441     * @param enable
00442     *
00443     * @return 成功返回 0; 失败返回错误码
00444     * AUBO_BUSY
00445     * AUBO_BAD_STATE
00446     * -AUBO_INVL_ARGUMENT
00447     * -AUBO_BAD_STATE
00448     *
00449     * @throws arcs::common_interface::AuboException
00450     *
00451     * @par Python 函数原型
00452     * freedrive(self: pyaubo_sdk.RobotManage, arg0: bool) -> int
00453     *
00454     * @par Lua 函数原型
00455     * freedrive(enable: boolean) -> number
00456     *

```

```

00457     * @par Lua 示例
00458     * num = freedrive(false)
00459     *
00460     * @par JSON-RPC 请求示例
00461     * {"jsonrpc": "2.0", "method": "rob1.RobotManage.freedrive", "params": [true], "id": 1}
00462     *
00463     * @par JSON-RPC 响应示例
00464     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00465     *
00466     * @par C++ 示例
00467     * @code
00468     * auto rpc_cli = std::make_shared<RpcClient>();
00469     * auto robot_name = rpc_cli->getRobotNames().front();
00470     * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->freedrive(true);
00471     * @endcode
00472     * \endchinese
00473     *
00474     * \english
00475     * Initiate robot freedrive request
00476     * This interface is deprecated after software version 0.31.x, use
00477     * handguideMode instead: handguideMode({1,1,1,1,1}, {0,0,0,0,0,0})
00478     *
00479     * @param enable
00480     *
00481     * @return Returns 0 on success; error code on failure
00482     * AUBO_BUSY
00483     * AUBO_BAD_STATE
00484     * -AUBO_INVL_ARGUMENT
00485     * -AUBO_BAD_STATE
00486     *
00487     * @throws arcs::common_interface::AuboException
00488     *
00489     * @par Python function prototype
00490     * freedrive(self: pyaubo_sdk.RobotManage, arg0: bool) -> int
00491     *
00492     * @par Lua function prototype
00493     * freedrive(enable: boolean) -> number
00494     *
00495     * @par Lua example
00496     * num = freedrive(false)
00497     *
00498     * @par JSON-RPC request example
00499     * {"jsonrpc": "2.0", "method": "rob1.RobotManage.freedrive", "params": [true], "id": 1}
00500     *
00501     * @par JSON-RPC response example
00502     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00503     *
00504     * @par C++ example
00505     * @code
00506     * auto rpc_cli = std::make_shared<RpcClient>();
00507     * auto robot_name = rpc_cli->getRobotNames().front();
00508     * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->freedrive(true);
00509     * @endcode
00510     * \endenglish
00511     */
00512     int freedrive(bool enable);
00513
00514     /**
00515     * @ingroup RobotManage
00516     * \chinese
00517     * 设置拖动示教参数
00518     *
00519     * @param freeAxes 可以拖动的轴 0-不能拖动 1-可以拖动
00520     * @param feature 如果维度为 0, 代表基于 TCP 坐标系拖动
00521     *
00522     * @return 成功返回 0; 失败返回错误码
00523     * AUBO_BUSY
00524     * AUBO_BAD_STATE
00525     * -AUBO_INVL_ARGUMENT
00526     * -AUBO_BAD_STATE
00527     *
00528     * @throws arcs::common_interface::AuboException
00529     * \endchinese
00530     *
00531     * \english
00532     * Advanced hand-guiding mode
00533     *
00534     * @param freeAxes Axes that can be moved: 0-cannot move, 1-can move
00535     * @param feature If the dimension is 0, it means hand-guiding based on
00536     * the TCP coordinate system
00537     *
00538     * @return Returns 0 on success; error code on failure
00539     * AUBO_BUSY
00540     * AUBO_BAD_STATE
00541     * -AUBO_INVL_ARGUMENT
00542     * -AUBO_BAD_STATE
00543     *

```

```

00544     * @throws arcs::common_interface::AuboException
00545     * \endenglish
00546     */
00547 int setHandguideParams(const std::vector<int> &freeAxes,
00548                       const std::vector<double> &feature);
00549
00550 /**
00551  * @ingroup RobotManage
00552  * \chinese
00553  * 获取拖动轴
00554  * \endchinese
00555  *
00556  * \english
00557  * Get Axes that can be moved
00558  * \endenglish
00559  */
00560 std::vector<int> getHandguideFreeAxes();
00561
00562 /**
00563  * @ingroup RobotManage
00564  * \chinese
00565  * 获取拖动参考坐标系
00566  * \endchinese
00567  *
00568  * \english
00569  * Get the drag reference coordinate system
00570  * \endenglish
00571  */
00572 std::vector<double> getHandguideFeature();
00573
00574 /**
00575  * @ingroup RobotManage
00576  * \chinese
00577  * 高阶拖动示教
00578  *
00579  * @param freeAxes 可以拖动的轴 0-不能拖动 1-可以拖动
00580  * @param feature 如果维度为 0，代表基于 TCP 坐标系拖动
00581  *
00582  * @return 成功返回 0；失败返回错误码
00583  * AUBO_BUSY
00584  * AUBO_BAD_STATE
00585  * -AUBO_INVL_ARGUMENT
00586  * -AUBO_BAD_STATE
00587  *
00588  * @throws arcs::common_interface::AuboException
00589  * \endchinese
00590  *
00591  * \english
00592  * Advanced hand-guiding mode
00593  *
00594  * @param freeAxes Axes that can be moved: 0-cannot move, 1-can move
00595  * @param feature If the dimension is 0, it means hand-guiding based on the
00596  * TCP coordinate system
00597  *
00598  * @return Returns 0 on success; error code on failure
00599  * AUBO_BUSY
00600  * AUBO_BAD_STATE
00601  * -AUBO_INVL_ARGUMENT
00602  * -AUBO_BAD_STATE
00603  *
00604  * @throws arcs::common_interface::AuboException
00605  * \endenglish
00606  */
00607 int handguideMode(const std::vector<int> &freeAxes,
00608                  const std::vector<double> &feature);
00609
00610 /**
00611  * @ingroup RobotManage
00612  * \chinese
00613  * 退出拖动示教
00614  *
00615  * @note 需 0.31.x 及以后软件版本
00616  *
00617  * @return 成功返回 0；失败返回错误码
00618  * AUBO_BUSY
00619  * AUBO_BAD_STATE
00620  * -AUBO_BAD_STATE
00621  *
00622  * @throws arcs::common_interface::AuboException
00623  * \endchinese
00624  *
00625  * \english
00626  * Exit hand-guiding mode
00627  *
00628  * @note Requires software version 0.31.x or later
00629  *
00630  * @return Returns 0 on success; error code on failure

```

```

00631      * AUBO_BUSY
00632      * AUBO_BAD_STATE
00633      * -AUBO_BAD_STATE
00634      *
00635      * @throws arcs::common_interface::AuboException
00636      * \endenglish
00637      */
00638  int  exitHandguideMode();
00639
00640  /**
00641   * @ingroup RobotManage
00642   * \chinese
00643   * 获取拖动示教器的状态（是否处于奇异空间）
00644   *
00645   * @note 暂未实现
00646   *
00647   * @return
00648   * • 0 - 正常操作.
00649   * • 1 - 接近奇异空间.
00650   * • 2 - 极其接近奇异点，将产生较大的拖动阻尼.
00651   *
00652   * @throws arcs::common_interface::AuboException
00653   * \endchinese
00654   *
00655   * \english
00656   * Get the status of the hand-guiding device (whether it is in a singular
00657   * space)
00658   *
00659   * @note Not implemented yet
00660   *
00661   * @return
00662   * • 0 - Normal operation.
00663   * • 1 - Approaching singular space.
00664   * • 2 - Extremely close to a singularity, large hand-guiding damping will
00665   * occur.
00666   *
00667   * @throws arcs::common_interface::AuboException
00668   * \endenglish
00669   */
00670  int  getHandguideStatus();
00671
00672  /**
00673   * @ingroup RobotManage
00674   * \chinese
00675   * 获取拖动示教器触发源
00676   *
00677   * @note 暂未实现
00678   *
00679   * @return
00680   *
00681   * @throws arcs::common_interface::AuboException
00682   * \endchinese
00683   *
00684   * \english
00685   * Get the trigger source of the hand-guiding device
00686   *
00687   * @note Not implemented yet
00688   *
00689   * @return
00690   *
00691   * @throws arcs::common_interface::AuboException
00692   * \endenglish
00693   */
00694  int  getHandguideTrigger();
00695
00696  /**
00697   * @ingroup RobotManage
00698   * \chinese
00699   * 获取拖动示教使能状态
00700   *
00701   * @return 使能返回 true; 失能返回 false
00702   *
00703   * @throws arcs::common_interface::AuboException
00704   *
00705   * @par Lua 函数原型
00706   * isHandguideEnabled() -> boolean
00707   *
00708   * @par Lua 示例
00709   * Handguide = isHandguideEnabled()
00710   *
00711   * @par JSON-RPC 请求示例
00712   * {"jsonrpc": "2.0", "method": "rob1.RobotManage.isHandguideEnabled", "params": [], "id": 1}
00713   *
00714   * @par JSON-RPC 响应示例
00715   * {"id": 1, "jsonrpc": "2.0", "result": false}
00716   *
00717   * \endchinese

```

```

00718     * \english
00719     * Get the hand-guiding enable status
00720     *
00721     * @return Returns true if enabled; false if disabled
00722     *
00723     * @throws arcs::common_interface::AuboException
00724     *
00725     * @par Lua function prototype
00726     * isHandguideEnabled() -> boolean
00727     *
00728     * @par Lua example
00729     * Handguide = isHandguideEnabled()
00730     *
00731     * @par JSON-RPC request example
00732     * {"jsonrpc": "2.0", "method": "rob1.RobotManage.isHandguideEnabled", "params": [], "id": 1}
00733     *
00734     * @par JSON-RPC response example
00735     * {"id": 1, "jsonrpc": "2.0", "result": false}
00736     *
00737     * \endenglish
00738     */
00739 bool isHandguideEnabled();
00740
00741 /**
00742     * @ingroup RobotManage
00743     * \chinese
00744     * 发起机器人进入/退出仿真模式请求
00745     *
00746     * @param enable
00747     *
00748     * @return 成功返回 0; 失败返回错误码
00749     * AUBO_BUSY
00750     * AUBO_BAD_STATE
00751     * -AUBO_INVL_ARGUMENT
00752     * -AUBO_BAD_STATE
00753     *
00754     * @throws arcs::common_interface::AuboException
00755     *
00756     * @par Python 函数原型
00757     * setSim(self: pyaubo_sdk.RobotManage, arg0: bool) -> int
00758     *
00759     * @par Lua 函数原型
00760     * setSim(enable: boolean) -> number
00761     *
00762     * @par Lua 示例
00763     * num = setSim(true)
00764     *
00765     * @par JSON-RPC 请求示例
00766     * {"jsonrpc": "2.0", "method": "rob1.RobotManage.setSim", "params": [true], "id": 1}
00767     *
00768     * @par JSON-RPC 响应示例
00769     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00770     *
00771     * @par C++ 示例
00772     * @code
00773     * auto rpc_cli = std::make_shared<RpcClient>();
00774     * auto robot_name = rpc_cli->getRobotNames().front();
00775     * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->setSim(true);
00776     * @endcode
00777     * \endchinese
00778     *
00779     * \english
00780     * Initiate robot enter/exit simulation mode request
00781     *
00782     * @param enable
00783     *
00784     * @return Returns 0 on success; error code on failure
00785     * AUBO_BUSY
00786     * AUBO_BAD_STATE
00787     * -AUBO_INVL_ARGUMENT
00788     * -AUBO_BAD_STATE
00789     *
00790     * @throws arcs::common_interface::AuboException
00791     *
00792     * @par Python function prototype
00793     * setSim(self: pyaubo_sdk.RobotManage, arg0: bool) -> int
00794     *
00795     * @par Lua function prototype
00796     * setSim(enable: boolean) -> number
00797     *
00798     * @par Lua example
00799     * num = setSim(true)
00800     *
00801     * @par JSON-RPC request example
00802     * {"jsonrpc": "2.0", "method": "rob1.RobotManage.setSim", "params": [true], "id": 1}
00803     *
00804     * @par JSON-RPC response example

```

```

00805     * {"id":1,"jsonrpc":"2.0","result":0}
00806     *
00807     * @par C++ example
00808     * @code
00809     * auto rpc_cli = std::make_shared<RpcClient>();
00810     * auto robot_name = rpc_cli->getRobotNames().front();
00811     * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->setSim(true);
00812     * @endcode
00813     * \endenglish
00814     */
00815 int setSim(bool enable);
00816
00817 /**
00818  * @ingroup RobotManage
00819  * \chinese
00820  * 设置机器人操作模式
00821  *
00822  * @param mode 操作模式
00823  *
00824  * @return 成功返回 0; 失败返回错误码
00825  * -AUBO_BAD_STATE
00826  *
00827  * @throws arcs::common_interface::AuboException
00828  *
00829  * @par Python 函数原型
00830  * setOperationalMode(self: pyaubo_sdk.RobotManage, arg0:
00831  * arcs::common_interface::OperationalModeType) -> int
00832  *
00833  * @par Lua 函数原型
00834  * setOperationalMode(mode: number) -> number
00835  *
00836  * @par Lua 示例
00837  * num = setOperationalMode("Manual")
00838  *
00839  * @par JSON-RPC 请求示例
00840  * {"jsonrpc":"2.0","method":"rob1.RobotManage.setOperationalMode","params":["Manual"],"id":1}
00841  *
00842  * @par JSON-RPC 响应示例
00843  * {"id":1,"jsonrpc":"2.0","result":0}
00844  *
00845  * @par C++ 示例
00846  * @code
00847  * auto rpc_cli = std::make_shared<RpcClient>();
00848  * auto robot_name = rpc_cli->getRobotNames().front();
00849  * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->setOperationalMode(OperationalModeType::Automatic);
00850  * @endcode
00851  * \endchinese
00852  * \english
00853  * Set the robot operational mode
00854  *
00855  * @param mode Operational mode
00856  *
00857  * @return Returns 0 on success; error code on failure
00858  * -AUBO_BAD_STATE
00859  *
00860  * @throws arcs::common_interface::AuboException
00861  *
00862  * @par Python function prototype
00863  * setOperationalMode(self: pyaubo_sdk.RobotManage, arg0:
00864  * arcs::common_interface::OperationalModeType) -> int
00865  *
00866  * @par Lua function prototype
00867  * setOperationalMode(mode: number) -> number
00868  *
00869  * @par Lua example
00870  * num = setOperationalMode("Manual")
00871  *
00872  * @par JSON-RPC request example
00873  * {"jsonrpc":"2.0","method":"rob1.RobotManage.setOperationalMode","params":["Manual"],"id":1}
00874  *
00875  * @par JSON-RPC response example
00876  * {"id":1,"jsonrpc":"2.0","result":0}
00877  *
00878  * @par C++ example
00879  * @code
00880  * auto rpc_cli = std::make_shared<RpcClient>();
00881  * auto robot_name = rpc_cli->getRobotNames().front();
00882  * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->setOperationalMode(OperationalModeType::Automatic);
00883  * @endcode
00884  * \endenglish
00885  */
00886 int setOperationalMode(OperationalModeType mode);
00887
00888 /**
00889  * @ingroup RobotManage
00890  * \chinese
00891  * 获取机器人操作模式

```

```

00892     *
00893     * @return 机器人操作模式
00894     *
00895     * @throws arcs::common_interface::AuboException
00896     *
00897     * @par Python 函数原型
00898     * getOperationalMode(self: pyaubo_sdk.RobotManage) ->
00899     * arcs::common_interface::OperationalModeType
00900     *
00901     * @par Lua 函数原型
00902     * getOperationalMode() -> number
00903     *
00904     * @par Lua 示例
00905     * num = getOperationalMode()
00906     *
00907     * @par JSON-RPC 请求示例
00908     * {"jsonrpc": "2.0", "method": "rob1.RobotManage.getOperationalMode", "params": [], "id": 1}
00909     *
00910     * @par JSON-RPC 响应示例
00911     * {"id": 1, "jsonrpc": "2.0", "result": "Manual"}
00912     *
00913     * @par C++ 示例
00914     * @code
00915     * auto rpc_cli = std::make_shared<RpcClient>();
00916     * auto robot_name = rpc_cli->getRobotNames().front();
00917     * OperationalModeType mode =
00918     *   rpc_cli->getRobotInterface(robot_name)->getRobotManage()->getOperationalMode();
00919     * @endcode
00920     * \endchinese
00921     * \english
00922     * Get the robot operational mode
00923     *
00924     * @return Robot operational mode
00925     *
00926     * @throws arcs::common_interface::AuboException
00927     *
00928     * @par Python function prototype
00929     * getOperationalMode(self: pyaubo_sdk.RobotManage) ->
00930     * arcs::common_interface::OperationalModeType
00931     *
00932     * @par Lua function prototype
00933     * getOperationalMode() -> number
00934     *
00935     * @par Lua example
00936     * num = getOperationalMode()
00937     *
00938     * @par JSON-RPC request example
00939     * {"jsonrpc": "2.0", "method": "rob1.RobotManage.getOperationalMode", "params": [], "id": 1}
00940     *
00941     * @par JSON-RPC response example
00942     * {"id": 1, "jsonrpc": "2.0", "result": "Manual"}
00943     *
00944     * @par C++ example
00945     * @code
00946     * auto rpc_cli = std::make_shared<RpcClient>();
00947     * auto robot_name = rpc_cli->getRobotNames().front();
00948     * OperationalModeType mode =
00949     *   rpc_cli->getRobotInterface(robot_name)->getRobotManage()->getOperationalMode();
00950     * @endcode
00951     * \endenglish
00952     */
00953 OperationalModeType getOperationalMode();
00954
00955 /**
00956  * @ingroup RobotManage
00957  * \chinese
00958  * 获取控制模式
00959  *
00960  * @return 控制模式
00961  *
00962  * @throws arcs::common_interface::AuboException
00963  *
00964  * @par Python 函数原型
00965  * getRobotControlMode(self: pyaubo_sdk.RobotManage) ->
00966  * arcs::common_interface::RobotControlModeType
00967  *
00968  * @par Lua 函数原型
00969  * getRobotControlMode() -> number
00970  *
00971  * @par Lua 示例
00972  * num = getRobotControlMode()
00973  *
00974  * @par JSON-RPC 请求示例
00975  * {"jsonrpc": "2.0", "method": "rob1.RobotManage.getRobotControlMode", "params": [], "id": 1}
00976  *
00977  * @par JSON-RPC 响应示例
00978  * {"id": 1, "jsonrpc": "2.0", "result": "Position"}

```



```

00979      *
00980      * @par C++ 示例
00981      * @code
00982      * auto rpc_cli = std::make_shared<RpcClient>();
00983      * auto robot_name = rpc_cli->getRobotNames().front();
00984      * RobotControlModeType mode =
00985      * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->getRobotControlMode();
00986      * @endcode
00987      * \endchinese
00988      * \english
00989      * Get the control mode
00990      *
00991      * @return Control mode
00992      *
00993      * @throws arcs::common_interface::AuboException
00994      *
00995      * @par Python function prototype
00996      * getRobotControlMode(self: pyaubo_sdk.RobotManage) ->
00997      * arcs::common_interface::RobotControlModeType
00998      *
00999      * @par Lua function prototype
01000      * getRobotControlMode() -> number
01001      *
01002      * @par Lua example
01003      * num = getRobotControlMode()
01004      *
01005      * @par JSON-RPC request example
01006      * {"jsonrpc": "2.0", "method": "rob1.RobotManage.getRobotControlMode", "params": [], "id": 1}
01007      *
01008      * @par JSON-RPC response example
01009      * {"id": 1, "jsonrpc": "2.0", "result": "Position"}
01010      *
01011      * @par C++ example
01012      * @code
01013      * auto rpc_cli = std::make_shared<RpcClient>();
01014      * auto robot_name = rpc_cli->getRobotNames().front();
01015      * RobotControlModeType mode =
01016      * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->getRobotControlMode();
01017      * @endcode
01018      * \endenglish
01019      */
01020 RobotControlModeType getRobotControlMode();
01021
01022 /**
01023  * @ingroup RobotManage
01024  * \chinese
01025  * 是否使能了拖动示教模式
01026  *
01027  * @return 使能返回 true; 反之返回 false
01028  *
01029  * @throws arcs::common_interface::AuboException
01030  *
01031  * @par Python 函数原型
01032  * isFreedriveEnabled(self: pyaubo_sdk.RobotManage) -> bool
01033  *
01034  * @par Lua 函数原型
01035  * isFreedriveEnabled() -> boolean
01036  *
01037  * @par Lua 示例
01038  * FreedriveEnabled = isFreedriveEnabled()
01039  *
01040  * @par JSON-RPC 请求示例
01041  * {"jsonrpc": "2.0", "method": "rob1.RobotManage.isFreedriveEnabled", "params": [], "id": 1}
01042  *
01043  * @par JSON-RPC 响应示例
01044  * {"id": 1, "jsonrpc": "2.0", "result": false}
01045  *
01046  * @par C++ 示例
01047  * @code
01048  * auto rpc_cli = std::make_shared<RpcClient>();
01049  * auto robot_name = rpc_cli->getRobotNames().front();
01050  * bool isEnabled =
01051  * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->isFreedriveEnabled();
01052  * @endcode
01053  * \endchinese
01054  * \english
01055  * Whether the freedrive mode is enabled
01056  *
01057  * @return Returns true if enabled; otherwise returns false
01058  *
01059  * @throws arcs::common_interface::AuboException
01060  *
01061  * @par Python function prototype
01062  * isFreedriveEnabled(self: pyaubo_sdk.RobotManage) -> bool
01063  *
01064  * @par Lua function prototype
01065  * isFreedriveEnabled() -> boolean

```

```

01066     *
01067     * @par Lua example
01068     * FreedriveEnabled = isFreedriveEnabled()
01069     *
01070     * @par JSON-RPC request example
01071     * {"jsonrpc": "2.0", "method": "robl.RobotManage.isFreedriveEnabled", "params": [], "id": 1}
01072     *
01073     * @par JSON-RPC response example
01074     * {"id": 1, "jsonrpc": "2.0", "result": false}
01075     *
01076     * @par C++ example
01077     * @code
01078     * auto rpc_cli = std::make_shared<RpcClient>();
01079     * auto robot_name = rpc_cli->getRobotNames().front();
01080     * bool isEnabled =
01081     *   rpc_cli->getRobotInterface(robot_name)->getRobotManage()->isFreedriveEnabled();
01082     * @endcode
01083     * \endenglish
01084     */
01085     bool isFreedriveEnabled();
01086
01087     /**
01088     * @ingroup RobotManage
01089     * \chinese
01090     * 是否使能了反向驱动模式
01091     *
01092     * @return 使能返回 true; 反之返回 false
01093     *
01094     * @throws arcs::common_interface::AuboException
01095     *
01096     * @par Python 函数原型
01097     * isBackdriveEnabled(self: pyaubo_sdk.RobotManage) -> bool
01098     *
01099     * @par Lua 函数原型
01100     * isBackdriveEnabled() -> boolean
01101     *
01102     * @par Lua 示例
01103     * BackdriveEnabled = isBackdriveEnabled()
01104     *
01105     * @par JSON-RPC 请求示例
01106     * {"jsonrpc": "2.0", "method": "robl.RobotManage.isBackdriveEnabled", "params": [], "id": 1}
01107     *
01108     * @par JSON-RPC 响应示例
01109     * {"id": 1, "jsonrpc": "2.0", "result": false}
01110     *
01111     * @par C++ 示例
01112     * @code
01113     * auto rpc_cli = std::make_shared<RpcClient>();
01114     * auto robot_name = rpc_cli->getRobotNames().front();
01115     * bool isEnabled =
01116     *   rpc_cli->getRobotInterface(robot_name)->getRobotManage()->isBackdriveEnabled();
01117     * @endcode
01118     * \endchinese
01119     * \english
01120     * Whether the backdrive mode is enabled
01121     *
01122     * @return Returns true if enabled; otherwise returns false
01123     *
01124     * @throws arcs::common_interface::AuboException
01125     *
01126     * @par Python function prototype
01127     * isBackdriveEnabled(self: pyaubo_sdk.RobotManage) -> bool
01128     *
01129     * @par Lua function prototype
01130     * isBackdriveEnabled() -> boolean
01131     *
01132     * @par Lua example
01133     * BackdriveEnabled = isBackdriveEnabled()
01134     *
01135     * @par JSON-RPC request example
01136     * {"jsonrpc": "2.0", "method": "robl.RobotManage.isBackdriveEnabled", "params": [], "id": 1}
01137     *
01138     * @par JSON-RPC response example
01139     * {"id": 1, "jsonrpc": "2.0", "result": false}
01140     *
01141     * @par C++ example
01142     * @code
01143     * auto rpc_cli = std::make_shared<RpcClient>();
01144     * auto robot_name = rpc_cli->getRobotNames().front();
01145     * bool isEnabled =
01146     *   rpc_cli->getRobotInterface(robot_name)->getRobotManage()->isBackdriveEnabled();
01147     * @endcode
01148     * \endenglish
01149     */
01150     bool isBackdriveEnabled();
01151
01152     /**

```

```

01153     * @ingroup RobotManage
01154     * \chinese
01155     * 是否使能了仿真模式
01156     *
01157     * @return 使能返回 true; 反之返回 false
01158     *
01159     * @throws arcs::common_interface::AuboException
01160     *
01161     * @par Python 函数原型
01162     * isSimulationEnabled(self: pyaubo_sdk.RobotManage) -> bool
01163     *
01164     * @par Lua 函数原型
01165     * isSimulationEnabled() -> boolean
01166     *
01167     * @par Lua 示例
01168     * SimulationEnabled = isSimulationEnabled()
01169     *
01170     * @par JSON-RPC 请求示例
01171     * {"jsonrpc": "2.0", "method": "robl.RobotManage.isSimulationEnabled", "params": [], "id": 1}
01172     *
01173     * @par JSON-RPC 响应示例
01174     * {"id": 1, "jsonrpc": "2.0", "result": true}
01175     *
01176     * @par C++ 示例
01177     * @code
01178     * auto rpc_cli = std::make_shared<RpcClient>();
01179     * auto robot_name = rpc_cli->getRobotNames().front();
01180     * bool isEnabled =
01181     *   rpc_cli->getRobotInterface(robot_name)->getRobotManage()->isSimulationEnabled();
01182     * @endcode
01183     * \endchinese
01184     * \english
01185     * Whether the simulation mode is enabled
01186     *
01187     * @return Returns true if enabled; otherwise returns false
01188     *
01189     * @throws arcs::common_interface::AuboException
01190     *
01191     * @par Python function prototype
01192     * isSimulationEnabled(self: pyaubo_sdk.RobotManage) -> bool
01193     *
01194     * @par Lua function prototype
01195     * isSimulationEnabled() -> boolean
01196     *
01197     * @par Lua example
01198     * SimulationEnabled = isSimulationEnabled()
01199     *
01200     * @par JSON-RPC request example
01201     * {"jsonrpc": "2.0", "method": "robl.RobotManage.isSimulationEnabled", "params": [], "id": 1}
01202     *
01203     * @par JSON-RPC response example
01204     * {"id": 1, "jsonrpc": "2.0", "result": true}
01205     *
01206     * @par C++ example
01207     * @code
01208     * auto rpc_cli = std::make_shared<RpcClient>();
01209     * auto robot_name = rpc_cli->getRobotNames().front();
01210     * bool isEnabled =
01211     *   rpc_cli->getRobotInterface(robot_name)->getRobotManage()->isSimulationEnabled();
01212     * @endcode
01213     * \endenglish
01214     */
01215     bool isSimulationEnabled();
01216
01217     /**
01218     * @ingroup RobotManage
01219     * \chinese
01220     * 清除防护停机, 包括碰撞停机
01221     *
01222     * @return 成功返回 0; 失败返回错误码
01223     * AUBO_BUSY
01224     * AUBO_BAD_STATE
01225     * -AUBO_BAD_STATE
01226     *
01227     * @throws arcs::common_interface::AuboException
01228     *
01229     * @par Python 函数原型
01230     * setUnlockProtectiveStop(self: pyaubo_sdk.RobotManage) -> int
01231     *
01232     * @par Lua 函数原型
01233     * setUnlockProtectiveStop() -> number
01234     *
01235     * @par Lua 示例
01236     * num = setUnlockProtectiveStop()
01237     *
01238     * @par JSON-RPC 请求示例
01239     * {"jsonrpc": "2.0", "method": "robl.RobotManage.setUnlockProtectiveStop", "params": [], "id": 1}

```

```

01240      *
01241      * @par JSON-RPC 响应示例
01242      * {"id":1,"jsonrpc":"2.0","result":0}
01243      *
01244      * @par C++ 示例
01245      * @code
01246      * auto rpc_cli = std::make_shared<RpcClient>();
01247      * auto robot_name = rpc_cli->getRobotNames().front();
01248      * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->setUnlockProtectiveStop();
01249      * @endcode
01250      * \endchinese
01251      * \english
01252      * Clear protective stop, including collision stop
01253      *
01254      * @return Returns 0 on success; error code on failure
01255      * AUBO_BUSY
01256      * AUBO_BAD_STATE
01257      * -AUBO_BAD_STATE
01258      *
01259      * @throws arcs::common_interface::AuboException
01260      *
01261      * @par Python function prototype
01262      * setUnlockProtectiveStop(self: pyaubo_sdk.RobotManage) -> int
01263      *
01264      * @par Lua function prototype
01265      * setUnlockProtectiveStop() -> number
01266      *
01267      * @par Lua example
01268      * num = setUnlockProtectiveStop()
01269      *
01270      * @par JSON-RPC request example
01271      * {"jsonrpc":"2.0","method":"rob1.RobotManage.setUnlockProtectiveStop","params":[],"id":1}
01272      *
01273      * @par JSON-RPC response example
01274      * {"id":1,"jsonrpc":"2.0","result":0}
01275      *
01276      * @par C++ example
01277      * @code
01278      * auto rpc_cli = std::make_shared<RpcClient>();
01279      * auto robot_name = rpc_cli->getRobotNames().front();
01280      * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->setUnlockProtectiveStop();
01281      * @endcode
01282      * \endenglish
01283      */
01284 int setUnlockProtectiveStop();
01285
01286 /**
01287  * @ingroup RobotManage
01288  * \chinese
01289  * 重置安全接口板，一般在机器人断电之后需要重置时调用，比如机器人急停、故障等之后
01290  *
01291  * @return 成功返回 0；失败返回错误码
01292  * AUBO_BUSY
01293  * AUBO_BAD_STATE
01294  * -AUBO_BAD_STATE
01295  *
01296  * @throws arcs::common_interface::AuboException
01297  *
01298  * @par Python 函数原型
01299  * restartInterfaceBoard(self: pyaubo_sdk.RobotManage) -> int
01300  *
01301  * @par Lua 函数原型
01302  * restartInterfaceBoard() -> number
01303  *
01304  * @par Lua 示例
01305  * num = restartInterfaceBoard()
01306  *
01307  * @par JSON-RPC 请求示例
01308  * {"jsonrpc":"2.0","method":"rob1.RobotManage.restartInterfaceBoard","params":[],"id":1}
01309  *
01310  * @par JSON-RPC 响应示例
01311  * {"id":1,"jsonrpc":"2.0","result":0}
01312  *
01313  * @par C++ 示例
01314  * @code
01315  * auto rpc_cli = std::make_shared<RpcClient>();
01316  * auto robot_name = rpc_cli->getRobotNames().front();
01317  * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->restartInterfaceBoard();
01318  * @endcode
01319  * \endchinese
01320  * \english
01321  * Reset the safety interface board, usually called after the robot is
01322  * powered off and needs to be reset, such as after emergency stop or fault.
01323  *
01324  * @return Returns 0 on success; error code on failure
01325  * AUBO_BUSY
01326  * AUBO_BAD_STATE

```

```

01327     * -AUBO_BAD_STATE
01328     *
01329     * @throws arcs::common_interface::AuboException
01330     *
01331     * @par Python function prototype
01332     * restartInterfaceBoard(self: pyaubo_sdk.RobotManage) -> int
01333     *
01334     * @par Lua function prototype
01335     * restartInterfaceBoard() -> number
01336     *
01337     * @par Lua example
01338     * num = restartInterfaceBoard()
01339     *
01340     * @par JSON-RPC request example
01341     * {"jsonrpc": "2.0", "method": "rob1.RobotManage.restartInterfaceBoard", "params": [], "id": 1}
01342     *
01343     * @par JSON-RPC response example
01344     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01345     *
01346     * @par C++ example
01347     * @code
01348     * auto rpc_cli = std::make_shared<RpcClient>();
01349     * auto robot_name = rpc_cli->getRobotNames().front();
01350     * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->restartInterfaceBoard();
01351     * @endcode
01352     * \endenglish
01353     */
01354 int restartInterfaceBoard();
01355
01356 /**
01357  * @ingroup RobotManage
01358  * \chinese
01359  * 释放并清空指定内存缓存的记录数据
01360  *
01361  * @param name
01362  * 缓存名称
01363  *
01364  * @return 成功返回 0；失败返回错误码 (<0)
01365  *
01366  * @throws arcs::common_interface::AuboException
01367  *
01368  * @par Lua 函数原型
01369  * recordCacheFree(name: string) -> nil
01370  *
01371  * @par Lua 示例
01372  * recordCacheFree("rec")
01373  *
01374  * \endchinese
01375  * \english
01376  * Free and clear recorded data of the specified memory cache
01377  *
01378  * @param name
01379  * Cache name
01380  *
01381  * @return Returns 0 on success; negative error code on failure
01382  *
01383  * @throws arcs::common_interface::AuboException
01384  *
01385  * @par Lua function prototype
01386  * recordCacheFree(name: string) -> nil
01387  *
01388  * @par Lua example
01389  * recordCacheFree("rec")
01390  *
01391  * \endenglish
01392  */
01393 int recordCacheFree(const std::string &name);
01394
01395 /**
01396  * @ingroup RobotManage
01397  * \chinese
01398  * 开始实时轨迹的内存缓存记录（不落盘）
01399  *
01400  * @param name
01401  * 缓存名称（用于索引内存中的记录数据）。空字符串将返回参数错误。
01402  *
01403  * @return 成功返回 0；失败返回错误码 (<0)
01404  *
01405  * @throws arcs::common_interface::AuboException
01406  *
01407  * @par Lua 函数原型
01408  * startRecordCache(name: string) -> nil
01409  *
01410  * @par Lua 示例
01411  * startRecordCache("rec")
01412  *
01413  * \endchinese

```

```

01414      * \english
01415      * Start real-time trajectory recording to memory cache (no file output)
01416      *
01417      * @param name
01418      * Cache name used to index recorded data in memory. Empty string returns
01419      * invalid argument.
01420      *
01421      * @return Returns 0 on success; negative error code on failure
01422      *
01423      * @throws arcs::common_interface::AuboException
01424      *
01425      * @par Lua function prototype
01426      * startRecordCache(name: string) -> nil
01427      *
01428      * @par Lua example
01429      * startRecordCache("rec")
01430      *
01431      * \endenglish
01432      */
01433      int startRecordCache(const std::string &name);
01434
01435      /**
01436      * @ingroup RobotManage
01437      * \chinese
01438      * 停止当前实时轨迹内存缓存记录
01439      *
01440      * @return 成功返回 0；失败返回错误码 (<0)
01441      *
01442      * @throws arcs::common_interface::AuboException
01443      *
01444      * @par Lua 函数原型
01445      * stopRecordCache() -> nil
01446      *
01447      * @par Lua 示例
01448      * stopRecordCache()
01449      *
01450      * \endchinese
01451      * \english
01452      * Stop current real-time trajectory recording to memory cache
01453      *
01454      * @return Returns 0 on success; negative error code on failure
01455      *
01456      * @throws arcs::common_interface::AuboException
01457      *
01458      * @par Lua function prototype
01459      * stopRecordCache() -> nil
01460      *
01461      * @par Lua example
01462      * stopRecordCache()
01463      *
01464      * \endenglish
01465      */
01466      int stopRecordCache();
01467
01468      /**
01469      * @ingroup RobotManage
01470      * \chinese
01471      * 暂停/恢复当前实时轨迹内存缓存记录
01472      *
01473      * @param pause
01474      * true: 暂停缓存记录; false: 恢复缓存记录
01475      *
01476      * @return 成功返回 0；失败返回错误码 (<0)
01477      *
01478      * @throws arcs::common_interface::AuboException
01479      *
01480      * @par Lua 函数原型
01481      * pauseRecordCache(pause: boolean) -> nil
01482      *
01483      * @par Lua 示例
01484      * pauseRecordCache(true)  -- 暂停
01485      * pauseRecordCache(false) -- 恢复
01486      *
01487      * \endchinese
01488      * \english
01489      * Pause or resume current real-time trajectory recording to memory cache
01490      *
01491      * @param pause
01492      * true to pause cache recording; false to resume cache recording
01493      *
01494      * @return Returns 0 on success; negative error code on failure
01495      *
01496      * @throws arcs::common_interface::AuboException
01497      *
01498      * @par Lua function prototype
01499      * pauseRecordCache(pause: boolean) -> nil
01500      *

```

```

01501     * @par Lua example
01502     * pauseRecordCache(true)  -- pause
01503     * pauseRecordCache(false) -- resume
01504     *
01505     * \endenglish
01506     */
01507 int pauseRecordCache(bool pause);
01508
01509 /**
01510  * @ingroup RobotManage
01511  * \chinese
01512  * 获取指定内存缓存的记录数据
01513  *
01514  * @param name
01515  * 缓存名称
01516  *
01517  * @param frames
01518  * 获取帧数上限:
01519  * - frames=0: 获取全部已记录帧 (1 帧 =1 周期)
01520  * - frames>0: 最多获取最近 frames 帧
01521  *
01522  * @return 成功返回获取到的帧数 (>=0); 失败返回错误码 (<0)
01523  *
01524  * @throws arcs::common_interface::AuboException
01525  *
01526  * @par Lua 函数原型
01527  * getRecordCache(name: string, frames: number = 0) -> number
01528  *
01529  * @par Lua 示例
01530  * n = getRecordCache("rec")          -- 获取全部, 返回帧数
01531  * n = getRecordCache("rec", 2000)    -- 获取最近 2000 帧, 返回帧数
01532  *
01533  * \endchinese
01534  * \english
01535  * Get recorded data from the specified memory cache
01536  *
01537  * @param name
01538  * Cache name
01539  *
01540  * @param frames
01541  * Frame limit:
01542  * - frames=0: get all recorded frames (1 frame = 1 control cycle)
01543  * - frames>0: get up to the latest 'frames' frames
01544  *
01545  * @return Returns the number of frames fetched (>=0) on success;
01546  *         negative error code on failure
01547  *
01548  * @throws arcs::common_interface::AuboException
01549  *
01550  * @par Lua function prototype
01551  * getRecordCache(name: string, frames: number = 0) -> number
01552  *
01553  * @par Lua example
01554  * n = getRecordCache("rec")          -- all frames, returns count
01555  * n = getRecordCache("rec", 2000)    -- latest 2000 frames, returns count
01556  *
01557  * \endenglish
01558  */
01559 int getRecordCache(const std::string &name, size_t frames = 0);
01560
01561 /**
01562  * @ingroup RobotManage
01563  * \chinese
01564  * 开始实时轨迹的记录
01565  *
01566  * @param file_name
01567  *
01568  * @return 成功返回 0; 失败返回错误码
01569  * AUBO_BUSY
01570  * AUBO_BAD_STATE
01571  * -AUBO_INVL_ARGUMENT
01572  * -AUBO_BAD_STATE
01573  *
01574  * @throws arcs::common_interface::AuboException
01575  *
01576  * @par Lua 函数原型
01577  * startRecord(file_name: string) -> nil
01578  *
01579  * @par Lua 示例
01580  * startRecord("traje.csv")
01581  *
01582  * \endchinese
01583  * \english
01584  * Start real-time trajectory recording
01585  *
01586  * @param file_name
01587  *

```

```

01588     * @return Returns 0 on success; error code on failure
01589     * AUBO_BUSY
01590     * AUBO_BAD_STATE
01591     * -AUBO_INVL_ARGUMENT
01592     * -AUBO_BAD_STATE
01593     *
01594     * @throws arcs::common_interface::AuboException
01595     *
01596     * @par Lua function prototype
01597     * startRecord(fiel_name: string) -> nil
01598     *
01599     * @par Lua example
01600     * startRecord("traje.csv")
01601     *
01602     * \endenglish
01603     */
01604     int startRecord(const std::string &file_name);
01605
01606     /**
01607     * @ingroup RobotManage
01608     * \chinese
01609     * 停止实时记录
01610     *
01611     * @return 成功返回 0；失败返回错误码
01612     * AUBO_BUSY
01613     * AUBO_BAD_STATE
01614     * -AUBO_BAD_STATE
01615     *
01616     * @throws arcs::common_interface::AuboException
01617     *
01618     * @par Lua 函数原型
01619     * stopRecord() -> nil
01620     *
01621     * @par Lua 示例
01622     * stopRecord()
01623     *
01624     * @par JSON-RPC 请求示例
01625     * {"jsonrpc":"2.0","method":"rob1.RobotManage.stopRecord","params":[],"id":1}
01626     *
01627     * @par JSON-RPC 响应示例
01628     * {"id":1,"jsonrpc":"2.0","result":0}
01629     *
01630     * \endchinese
01631     * \english
01632     * Stop real-time recording
01633     *
01634     * @return Returns 0 on success; error code on failure
01635     * AUBO_BUSY
01636     * AUBO_BAD_STATE
01637     * -AUBO_BAD_STATE
01638     *
01639     * @throws arcs::common_interface::AuboException
01640     *
01641     * @par Lua function prototype
01642     * stopRecord() -> nil
01643     *
01644     * @par Lua example
01645     * stopRecord()
01646     *
01647     * @par JSON-RPC request example
01648     * {"jsonrpc":"2.0","method":"rob1.RobotManage.stopRecord","params":[],"id":1}
01649     *
01650     * @par JSON-RPC response example
01651     * {"id":1,"jsonrpc":"2.0","result":0}
01652     *
01653     * \endenglish
01654     */
01655     int stopRecord();
01656
01657     /**
01658     * @ingroup RobotManage
01659     * \chinese
01660     * 暂停实时记录
01661     *
01662     * @param pause
01663     *
01664     * @return 成功返回 0；失败返回错误码
01665     * AUBO_BUSY
01666     * AUBO_BAD_STATE
01667     * -AUBO_BAD_STATE
01668     *
01669     * @throws arcs::common_interface::AuboException
01670     *
01671     * @par JSON-RPC 请求示例
01672     * {"jsonrpc":"2.0","method":"rob1.RobotManage.pauseRecord","params":[true],"id":1}
01673     *
01674     * @par JSON-RPC 响应示例

```



```

01675     * {"id":1,"jsonrpc":"2.0","result":0}
01676     *
01677     * \endchinese
01678     * \english
01679     * Pause real-time recording
01680     *
01681     * @param pause
01682     *
01683     * @return Returns 0 on success; error code on failure
01684     * AUBO_BUSY
01685     * AUBO_BAD_STATE
01686     * -AUBO_BAD_STATE
01687     *
01688     * @throws arcs::common_interface::AuboException
01689     *
01690     * @par JSON-RPC request example
01691     * {"jsonrpc":"2.0","method":"rob1.RobotManage.pauseRecord","params":[true],"id":1}
01692     *
01693     * @par JSON-RPC response example
01694     * {"id":1,"jsonrpc":"2.0","result":0}
01695     *
01696     * \endenglish
01697     */
01698 int pauseRecord(bool pause);
01699
01700 /**
01701  * @ingroup RobotManage
01702  * \chinese
01703  * 发起机器人进入/退出联动模式请求，
01704  * 只有操作模式为自动或者无时，才能使能联动模式
01705  *
01706  * @param enable
01707  *
01708  * @return 成功返回 0；失败返回错误码
01709  * AUBO_BUSY
01710  * AUBO_REQUEST_IGNORE
01711  * -AUBO_BAD_STATE
01712  *
01713  * @throws arcs::common_interface::AuboException
01714  *
01715  * @par Python 函数原型
01716  * setLinkModeEnable(self: pyaubo_sdk.RobotManage, arg0: bool) -> int
01717  *
01718  * @par Lua 函数原型
01719  * setLinkModeEnable(enable: boolean) -> number
01720  *
01721  * @par Lua 示例
01722  * num = setLinkModeEnable(true)
01723  *
01724  * @par JSON-RPC 请求示例
01725  * {"jsonrpc":"2.0","method":"rob1.RobotManage.setLinkModeEnable","params":[true],"id":1}
01726  *
01727  * @par JSON-RPC 响应示例
01728  * {"id":1,"jsonrpc":"2.0","result":0}
01729  *
01730  * @par C++ 示例
01731  * @code
01732  * auto rpc_cli = std::make_shared<RpcClient>();
01733  * auto robot_name = rpc_cli->getRobotNames().front();
01734  * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->setLinkModeEnable(true);
01735  * @endcode
01736  * \endchinese
01737  * \english
01738  * Initiate robot enter/exit link mode request.
01739  * Link mode can only be enabled when the operation mode is Automatic or
01740  * None.
01741  *
01742  * @param enable
01743  *
01744  * @return Returns 0 on success; error code on failure
01745  * AUBO_BUSY
01746  * AUBO_REQUEST_IGNORE
01747  * -AUBO_BAD_STATE
01748  *
01749  * @throws arcs::common_interface::AuboException
01750  *
01751  * @par Python function prototype
01752  * setLinkModeEnable(self: pyaubo_sdk.RobotManage, arg0: bool) -> int
01753  *
01754  * @par Lua function prototype
01755  * setLinkModeEnable(enable: boolean) -> number
01756  *
01757  * @par Lua example
01758  * num = setLinkModeEnable(true)
01759  *
01760  * @par JSON-RPC request example
01761  * {"jsonrpc":"2.0","method":"rob1.RobotManage.setLinkModeEnable","params":[true],"id":1}

```

```

01762     *
01763     * @par JSON-RPC response example
01764     * {"id":1,"jsonrpc":"2.0","result":0}
01765     *
01766     * @par C++ example
01767     * @code
01768     * auto rpc_cli = std::make_shared<RpcClient>();
01769     * auto robot_name = rpc_cli->getRobotNames().front();
01770     * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->setLinkModeEnable(true);
01771     * @endcode
01772     * \endenglish
01773     */
01774 int setLinkModeEnable(bool enable);
01775
01776 /**
01777  * @ingroup RobotManage
01778  * \chinese
01779  * 是否使能了联动模式，联动模式下用户可以通过外部 IO 控制机器人（用户可以对 IO 的功能进行配置）
01780  *
01781  * @return 使能返回 true；反之返回 false
01782  *
01783  * @throws arcs::common_interface::AuboException
01784  *
01785  * @par Python 函数原型
01786  * isLinkModeEnabled(self: pyaubo_sdk.RobotManage) -> bool
01787  *
01788  * @par Lua 函数原型
01789  * isLinkModeEnabled() -> boolean
01790  *
01791  * @par Lua 示例
01792  * LinkModeEnabled = isLinkModeEnabled()
01793  *
01794  * @par JSON-RPC 请求示例
01795  * {"jsonrpc":"2.0","method":"robl.RobotManage.isLinkModeEnabled","params":[],"id":1}
01796  *
01797  * @par JSON-RPC 响应示例
01798  * {"id":1,"jsonrpc":"2.0","result":false}
01799  *
01800  * @par C++ 示例
01801  * @code
01802  * auto rpc_cli = std::make_shared<RpcClient>();
01803  * auto robot_name = rpc_cli->getRobotNames().front();
01804  * bool isEnabled =
01805  * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->isLinkModeEnabled();
01806  * @endcode
01807  * \endchinese
01808  * \english
01809  * Whether the link mode is enabled. In link mode, users can control the
01810  * robot via external IO (users can configure the IO functions).
01811  *
01812  * @return Returns true if enabled; otherwise returns false
01813  *
01814  * @throws arcs::common_interface::AuboException
01815  *
01816  * @par Python function prototype
01817  * isLinkModeEnabled(self: pyaubo_sdk.RobotManage) -> bool
01818  *
01819  * @par Lua function prototype
01820  * isLinkModeEnabled() -> boolean
01821  *
01822  * @par Lua example
01823  * LinkModeEnabled = isLinkModeEnabled()
01824  *
01825  * @par JSON-RPC request example
01826  * {"jsonrpc":"2.0","method":"robl.RobotManage.isLinkModeEnabled","params":[],"id":1}
01827  *
01828  * @par JSON-RPC response example
01829  * {"id":1,"jsonrpc":"2.0","result":false}
01830  *
01831  * @par C++ example
01832  * @code
01833  * auto rpc_cli = std::make_shared<RpcClient>();
01834  * auto robot_name = rpc_cli->getRobotNames().front();
01835  * bool isEnabled =
01836  * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->isLinkModeEnabled();
01837  * @endcode
01838  * \endenglish
01839  */
01840 bool isLinkModeEnabled();
01841
01842 /**
01843  * @ingroup RobotManage
01844  * \chinese
01845  * 手动触发生成诊断文件
01846  *
01847  * @return 指令下发成功返回 0；失败返回错误码。 \n
01848  * -AUBO_BAD_STATE: 运行时（RuntimeMachine）的当前状态不是 Stopped，

```

```

01849      * 固件升级请求被拒绝。AUBO_BAD_STATE 的值是 1。 \n
01850      * -AUBO_TIMEOUT: 超时。AUBO_TIMEOUT 的值是 4。 \n
01851      *
01852      * @throws arcs::common_interface::AuboException
01853      *
01854      * @par Python 函数原型
01855      * generateDiagnoseFile(self: pyaubo_sdk.RobotManage, arg0: str) -> int
01856      *
01857      * @par Lua 函数原型
01858      * generateDiagnoseFile(reason: string) -> nil
01859      *
01860      * @par Lua 示例
01861      * generateDiagnoseFile("reason")
01862      *
01863      * @par JSON-RPC 请求示例
01864      * {"jsonrpc": "2.0", "method": "robl.RobotManage.generateDiagnoseFile", "params": ["reason"], "id": 1}
01865      *
01866      * @par JSON-RPC 响应示例
01867      * {"id": 1, "jsonrpc": "2.0", "result": false}
01868      *
01869      * @par C++ 示例
01870      * @code
01871      * auto rpc_cli = std::make_shared<RpcClient>();
01872      * auto robot_name = rpc_cli->getRobotNames().front();
01873      * bool isEnabled =
01874      * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->generateDiagnoseFile("reason");
01875      * @endcode
01876      * \endchinese
01877      * \english
01878      * Manually trigger the generation of a diagnostic file
01879      *
01880      * @return Returns 0 if the command is successfully issued; error code on
01881      * failure. \n -AUBO_BAD_STATE: The current state of the RuntimeMachine is
01882      * not Stopped, the firmware upgrade request is rejected. The value of
01883      * AUBO_BAD_STATE is 1. \n -AUBO_TIMEOUT: Timeout. The value of AUBO_TIMEOUT
01884      * is 4. \n
01885      *
01886      * @throws arcs::common_interface::AuboException
01887      *
01888      * @par Python function prototype
01889      * generateDiagnoseFile(self: pyaubo_sdk.RobotManage, arg0: str) -> int
01890      *
01891      * @par Lua function prototype
01892      * generateDiagnoseFile(reason: string) -> nil
01893      *
01894      * @par Lua example
01895      * generateDiagnoseFile("reason")
01896      *
01897      * @par JSON-RPC request example
01898      * {"jsonrpc": "2.0", "method": "robl.RobotManage.generateDiagnoseFile", "params": ["reason"], "id": 1}
01899      *
01900      * @par JSON-RPC response example
01901      * {"id": 1, "jsonrpc": "2.0", "result": false}
01902      *
01903      * @par C++ example
01904      * @code
01905      * auto rpc_cli = std::make_shared<RpcClient>();
01906      * auto robot_name = rpc_cli->getRobotNames().front();
01907      * bool isEnabled =
01908      * rpc_cli->getRobotInterface(robot_name)->getRobotManage()->generateDiagnoseFile("reason");
01909      * @endcode
01910      * \endenglish
01911      */
01912      int generateDiagnoseFile(const std::string &reason);
01913
01914 protected:
01915     void *d_;
01916 };
01917 using RobotManagePtr = std::shared_ptr<RobotManage>;
01918 } // namespace common_interface
01919 } // namespace arcs
01920
01921 #endif // AUBO_SDK_ROBOT_CONTROL_INTERFACE_H

```

12.32 include/aubo/robot/robot_state.h 文件参考

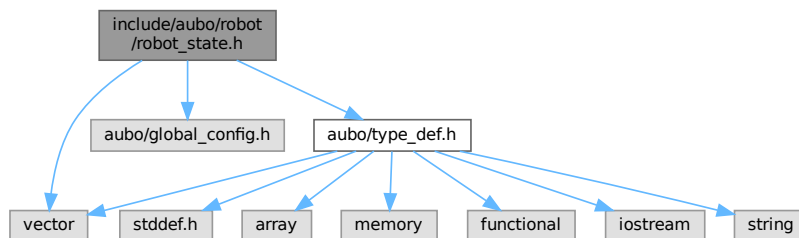
获取机器人状态接口，如关节速度、关节角度、固件/硬件版本

```

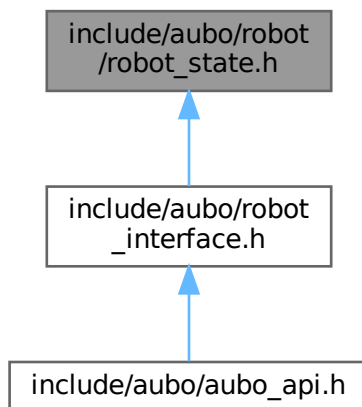
#include <vector>
#include <aubo/global_config.h>
#include <aubo/type_def.h>

```

robot_state.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::common_interface::RobotState`

命名空间

- namespace `arcs`
- namespace `arcs::common_interface`

类型定义

- using `arcs::common_interface::RobotStatePtr = std::shared_ptr<RobotState>`

12.32.1 详细描述

获取机器人状态接口, 如关节速度、关节角度、固件/硬件版本在文件 `robot_state.h` 中定义.

12.33 robot_state.h

[浏览该文件的文档.](#)

```

00001 /** @file robot_state.h
00002  * @brief 获取机器人状态接口，如关节速度、关节角度、固件/硬件版本
00003  */
00004 #ifndef AUBO_SDK_ROBOT_STATE_INTERFACE_H
00005 #define AUBO_SDK_ROBOT_STATE_INTERFACE_H
00006
00007 #include <vector>
00008
00009 #include <aubo/global_config.h>
00010 #include <aubo/type_def.h>
00011
00012 namespace arcs {
00013 namespace common_interface {
00014
00015 /**
00016  * @defgroup RobotState RobotState (机器人状态查询)
00017  * @ingroup RobotInterface
00018  * RobotState
00019  */
00020 class ARCS_ABI_EXPORT RobotState
00021 {
00022 public:
00023     RobotState();
00024     virtual ~RobotState();
00025
00026     /**
00027      * @ingroup RobotState
00028      * \chinese
00029      * 获取机器人的模式状态
00030      *
00031      * @return 机器人的模式状态
00032      *
00033      * @throws arcs::common_interface::AuboException
00034      *
00035      * @par Python 函数原型
00036      * getRobotModeType(self: pyaubo_sdk.RobotState) ->
00037      * arcs::common_interface::RobotModeType
00038      *
00039      * @par Lua 函数原型
00040      * getRobotModeType() -> number
00041      *
00042      * @par Lua 示例
00043      * RobotModeType = getRobotModeType()
00044      *
00045      * @par JSON-RPC 请求示例
00046      * {"jsonrpc": "2.0", "method": "robl.RobotState.getRobotModeType", "params": [], "id": 1}
00047      *
00048      * @par JSON-RPC 响应示例
00049      * {"id": 1, "jsonrpc": "2.0", "result": "Running"}
00050      *
00051      * \endchinese
00052      * \english
00053      * Get the robot mode state
00054      *
00055      * @return The robot's mode state
00056      *
00057      * @throws arcs::common_interface::AuboException
00058      *
00059      * @par Python function prototype
00060      * getRobotModeType(self: pyaubo_sdk.RobotState) ->
00061      * arcs::common_interface::RobotModeType
00062      *
00063      * @par Lua function prototype
00064      * getRobotModeType() -> number
00065      *
00066      * @par Lua example
00067      * RobotModeType = getRobotModeType()
00068      *
00069      * @par JSON-RPC request example
00070      * {"jsonrpc": "2.0", "method": "robl.RobotState.getRobotModeType", "params": [], "id": 1}
00071      *
00072      * @par JSON-RPC response example
00073      * {"id": 1, "jsonrpc": "2.0", "result": "Running"}
00074      * \endenglish
00075      */
00076
00077     RobotModeType getRobotModeType();
00078
00079     /**
00080      * @ingroup RobotState
00081      * \chinese
00082      * 获取安全模式
00083      *

```

```

00084     * @return 安全模式
00085     *
00086     * @throws arcs::common_interface::AuboException
00087     *
00088     * @par Python 函数原型
00089     * getSafetyModeType(self: pyaubo_sdk.RobotState) ->
00090     * arcs::common_interface::SafetyModeType
00091     *
00092     * @par Lua 函数原型
00093     * getSafetyModeType() -> number
00094     *
00095     * @par Lua 示例
00096     * SafetyModeType = getSafetyModeType()
00097     *
00098     * @par JSON-RPC 请求示例
00099     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getSafetyModeType", "params": [], "id": 1}
00100     *
00101     * @par JSON-RPC 响应示例
00102     * {"id": 1, "jsonrpc": "2.0", "result": "Normal"}
00103     * \endchinese
00104     * \english
00105     * Get the safety mode
00106     *
00107     * @return The safety mode
00108     *
00109     * @throws arcs::common_interface::AuboException
00110     *
00111     * @par Python function prototype
00112     * getSafetyModeType(self: pyaubo_sdk.RobotState) ->
00113     * arcs::common_interface::SafetyModeType
00114     *
00115     * @par Lua function prototype
00116     * getSafetyModeType() -> number
00117     *
00118     * @par Lua example
00119     * SafetyModeType = getSafetyModeType()
00120     *
00121     * @par JSON-RPC request example
00122     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getSafetyModeType", "params": [], "id": 1}
00123     *
00124     * @par JSON-RPC response example
00125     * {"id": 1, "jsonrpc": "2.0", "result": "Normal"}
00126     * \endenglish
00127     */
00128 SafetyModeType getSafetyModeType();
00129
00130 /**
00131     * @ingroup RobotState
00132     * \chinese
00133     * 获取机器人通电状态
00134     *
00135     * @return 机器人通电返回 true; 反之返回 false
00136     *
00137     * @throws arcs::common_interface::AuboException
00138     *
00139     * @par Lua 函数原型
00140     * isPowerOn() -> boolean
00141     *
00142     * @par Lua 示例
00143     * PowerOn = isPowerOn()
00144     *
00145     * @par JSON-RPC 请求示例
00146     * {"jsonrpc": "2.0", "method": "rob1.RobotState.isPowerOn", "params": [], "id": 1}
00147     *
00148     * @par JSON-RPC 响应示例
00149     * {"id": 1, "jsonrpc": "2.0", "result": true}
00150     * \endchinese
00151     * \english
00152     * Get the robot power-on state
00153     *
00154     * @return Returns true if the robot is powered on; otherwise returns false
00155     *
00156     * @throws arcs::common_interface::AuboException
00157     *
00158     * @par Lua function prototype
00159     * isPowerOn() -> boolean
00160     *
00161     * @par Lua example
00162     * PowerOn = isPowerOn()
00163     *
00164     * @par JSON-RPC request example
00165     * {"jsonrpc": "2.0", "method": "rob1.RobotState.isPowerOn", "params": [], "id": 1}
00166     *
00167     * @par JSON-RPC response example
00168     * {"id": 1, "jsonrpc": "2.0", "result": true}
00169     * \endenglish
00170     */

```

```

00171     bool isPowerOn();
00172
00173     /**
00174      * @ingroup RobotState
00175      * \chinese
00176      * 机器人是否已经停止下来
00177      *
00178      * @return 停止返回 true; 反之返回 false
00179      *
00180      * @throws arcs::common_interface::AuboException
00181      *
00182      * @par Python 函数原型
00183      * isSteady(self: pyaubo_sdk.RobotState) -> bool
00184      *
00185      * @par Lua 函数原型
00186      * isSteady() -> boolean
00187      *
00188      * @par Lua 示例
00189      * Steady = isSteady()
00190      *
00191      * @par JSON-RPC 请求示例
00192      * {"jsonrpc": "2.0", "method": "rob1.RobotState.isSteady", "params": [], "id": 1}
00193      *
00194      * @par JSON-RPC 响应示例
00195      * {"id": 1, "jsonrpc": "2.0", "result": true}
00196      * \endchinese
00197      * \english
00198      * Whether the robot has stopped
00199      *
00200      * @return Returns true if stopped; otherwise returns false
00201      *
00202      * @throws arcs::common_interface::AuboException
00203      *
00204      * @par Python function prototype
00205      * isSteady(self: pyaubo_sdk.RobotState) -> bool
00206      *
00207      * @par Lua function prototype
00208      * isSteady() -> boolean
00209      *
00210      * @par Lua exaple
00211      * Steady = isSteady()
00212      *
00213      * @par JSON-RPC request example
00214      * {"jsonrpc": "2.0", "method": "rob1.RobotState.isSteady", "params": [], "id": 1}
00215      *
00216      * @par JSON-RPC response example
00217      * {"id": 1, "jsonrpc": "2.0", "result": true}
00218      * \endenglish
00219      */
00220     bool isSteady();
00221
00222     /**
00223      * @ingroup RobotState
00224      * \chinese
00225      * 机器人是否发生了碰撞
00226      *
00227      * @return 发生碰撞返回 true; 反之返回 false
00228      *
00229      * @throws arcs::common_interface::AuboException
00230      *
00231      * @par Lua 函数原型
00232      * isCollisionOccurred() -> boolean
00233      *
00234      * @par Lua 示例
00235      * CollisionOccurred = isCollisionOccurred()
00236      *
00237      * @par JSON-RPC 请求示例
00238      * {"jsonrpc": "2.0", "method": "rob1.RobotState.isCollisionOccurred", "params": [], "id": 1}
00239      *
00240      * @par JSON-RPC 响应示例
00241      * {"id": 1, "jsonrpc": "2.0", "result": false}
00242      * \endchinese
00243      * \english
00244      * Whether a collision has occurred
00245      *
00246      * @return Returns true if a collision occurred; otherwise returns false
00247      *
00248      * @throws arcs::common_interface::AuboException
00249      *
00250      * @par Lua function prototype
00251      * isCollisionOccurred() -> boolean
00252      *
00253      * @par Lua example
00254      * CollisionOccurred = isCollisionOccurred()
00255      *
00256      * @par JSON-RPC request example
00257      * {"jsonrpc": "2.0", "method": "rob1.RobotState.isCollisionOccurred", "params": [], "id": 1}

```

```

00258     *
00259     * @par JSON-RPC response example
00260     * {"id":1,"jsonrpc":"2.0","result":false}
00261     * \endenglish
00262     */
00263     bool isCollisionOccurred();
00264
00265     /**
00266     * @ingroup RobotState
00267     * \chinese
00268     * 机器人是否已经在安全限制之内
00269     *
00270     * @return 在安全限制之内返回 true; 反之返回 false
00271     *
00272     * @throws arcs::common_interface::AuboException
00273     *
00274     * @par Python 函数原型
00275     * isWithinSafetyLimits(self: pyaubo_sdk.RobotState) -> bool
00276     *
00277     * @par Lua 函数原型
00278     * isWithinSafetyLimits() -> boolean
00279     *
00280     * @par Lua 示例
00281     * WithinSafetyLimits = isWithinSafetyLimits()
00282     *
00283     * @par JSON-RPC 请求示例
00284     * {"jsonrpc":"2.0","method":"rob1.RobotState.isWithinSafetyLimits","params":[],"id":1}
00285     *
00286     * @par JSON-RPC 响应示例
00287     * {"id":1,"jsonrpc":"2.0","result":true}
00288     * \endchinese
00289     * \english
00290     * Whether the robot is within safety limits
00291     *
00292     * @return Returns true if within safety limits; otherwise returns false
00293     *
00294     * @throws arcs::common_interface::AuboException
00295     *
00296     * @par Python function prototype
00297     * isWithinSafetyLimits(self: pyaubo_sdk.RobotState) -> bool
00298     *
00299     * @par Lua function prototype
00300     * isWithinSafetyLimits() -> boolean
00301     *
00302     * @par Lua example
00303     * WithinSafetyLimits = isWithinSafetyLimits()
00304     *
00305     * @par JSON-RPC request example
00306     * {"jsonrpc":"2.0","method":"rob1.RobotState.isWithinSafetyLimits","params":[],"id":1}
00307     *
00308     * @par JSON-RPC response example
00309     * {"id":1,"jsonrpc":"2.0","result":true}
00310     * \endenglish
00311     */
00312     bool isWithinSafetyLimits();
00313
00314     /**
00315     * @ingroup RobotState
00316     * \chinese
00317     * 获取当前的 TCP 位姿, 其 TCP 偏移可以通过 getActualTcpOffset 获取
00318     *
00319     * 位姿表示形式为 (x,y,z,rx,ry,rz)。
00320     * 其中 x、y、z 是工具中心点 (TCP) 在基坐标系下的位置, 单位是 m。
00321     * rx、ry、rz 是工具中心点 (TCP) 在基坐标系下的姿态, 是 ZYX 欧拉角, 单位是 rad。
00322     *
00323     * @return TCP 的位姿, 形式为 (x,y,z,rx,ry,rz)
00324     *
00325     * @throws arcs::common_interface::AuboException
00326     *
00327     * @par Python 函数原型
00328     * getTcpPose(self: pyaubo_sdk.RobotState) -> List[float]
00329     *
00330     * @par Lua 函数原型
00331     * getTcpPose() -> table
00332     *
00333     * @par Lua 示例
00334     * TcpPose = getTcpPose()
00335     *
00336     * @par JSON-RPC 请求示例
00337     * {"jsonrpc":"2.0","method":"rob1.RobotState.getTcpPose","params":[],"id":1}
00338     *
00339     * @par JSON-RPC 响应示例
00340     * {"id":1,"jsonrpc":"2.0","result":
00341     * [0.41777839846910425,-0.132500000000000012,0.20928451364415995,3.1415792312578987,0.0,1.5707963267948963]}
00342     * \endchinese
00343     * \english
00344     * Get the current TCP pose. The TCP offset can be obtained by

```



```

00344     * getActualTcpOffset.
00345     *
00346     * The pose is represented as (x, y, z, rx, ry, rz).
00347     * x, y, z are the position of the Tool Center Point (TCP) in the base
00348     * coordinate system, in meters. rx, ry, rz are the orientation of the TCP
00349     * in the base coordinate system, as ZYX Euler angles in radians.
00350     *
00351     * @return The TCP pose, in the form (x, y, z, rx, ry, rz)
00352     *
00353     * @throws arcs::common_interface::AuboException
00354     *
00355     * @par Python function prototype
00356     * getTcpPose(self: pyaubo_sdk.RobotState) -> List[float]
00357     *
00358     * @par Lua function prototype
00359     * getTcpPose() -> table
00360     *
00361     * @par Lua example
00362     * TcpPose = getTcpPose()
00363     *
00364     * @par JSON-RPC request example
00365     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getTcpPose", "params": [], "id": 1}
00366     *
00367     * @par JSON-RPC response example
00368     * {"id": 1, "jsonrpc": "2.0", "result":
[0.41777839846910425, -0.132500000000000012, 0.20928451364415995, 3.1415792312578987, 0.0, 1.5707963267948963]}
00369     * \endenglish
00370     */
00371     std::vector<double> getTcpPose();
00372
00373     /**
00374     * @ingroup RobotState
00375     * \chinese
00376     * 获取当前的 TCP 偏移, 也就是 getTcpPose 返回的 pose 用到的 TCP 偏移
00377     *
00378     * @return 当前的 TCP 偏移
00379     *
00380     * @par Lua 函数原型
00381     * getActualTcpOffset() -> table
00382     *
00383     * @par Lua 示例
00384     * ActualTcpOffset = getActualTcpOffset()
00385     *
00386     * \endchinese
00387     * \english
00388     * Get the current TCP offset, which is the TCP offset used by the pose
00389     * returned from getTcpPose
00390     *
00391     * @return The current TCP offset
00392     *
00393     * @par Lua function prototype
00394     * getActualTcpOffset() -> table
00395     *
00396     * @par Lua example
00397     * ActualTcpOffset = getActualTcpOffset()
00398     *
00399     * \endenglish
00400     */
00401     std::vector<double> getActualTcpOffset();
00402
00403     /**
00404     * @ingroup RobotState
00405     * \chinese
00406     * 获取下一个目标路点
00407     * 注意与 getTcpTargetPose 的区别, 此处定义存在歧义, 命名需要优化
00408     *
00409     * 位姿表示形式为 (x,y,z,rx,ry,rz)。
00410     * 其中 x、y、z 是工具中心点 (TCP) 在基坐标系下的目标位置, 单位是 m。
00411     * rx、ry、rz 是工具中心点 (TCP) 在基坐标系下的目标姿态, 是 ZYX 欧拉角, 单位是 rad。
00412     *
00413     * @return 当前目标位姿, 形式为 (x,y,z,rx,ry,rz)
00414     *
00415     * @throws arcs::common_interface::AuboException
00416     *
00417     * @par Python 函数原型
00418     * getTargetTcpPose(self: pyaubo_sdk.RobotState) -> List[float]
00419     *
00420     * @par Lua 函数原型
00421     * getTargetTcpPose() -> table
00422     *
00423     * @par Lua 示例
00424     * TargetTcpPose = getTargetTcpPose()
00425     *
00426     * @par JSON-RPC 请求示例
00427     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getTargetTcpPose", "params": [], "id": 1}
00428     *
00429     * @par JSON-RPC 响应示例

```

```

00430     * {"id":1,"jsonrpc":"2.0","result":
[0.4173932217619493,-0.13250000000000012,0.43296496133045825,3.141577313781914,0.0,1.5707963267948963]}
00431     * \endchinese
00432     * \english
00433     * Get the next target waypoint.
00434     * Note the difference from getTcpTargetPose; the definition here is
00435     * ambiguous and the naming needs improvement.
00436     *
00437     * The pose is represented as (x, y, z, rx, ry, rz).
00438     * x, y, z are the target position of the Tool Center Point (TCP) in the
00439     * base coordinate system, in meters. rx, ry, rz are the target orientation
00440     * of the TCP in the base coordinate system, as ZYX Euler angles in radians.
00441     *
00442     * @return The current target pose, in the form (x, y, z, rx, ry, rz)
00443     *
00444     * @throws arcs::common_interface::AuboException
00445     *
00446     * @par Python function prototype
00447     * getTargetTcpPose(self: pyaubo_sdk.RobotState) -> List[float]
00448     *
00449     * @par Lua function prototype
00450     * getTargetTcpPose() -> table
00451     *
00452     * @par Lua example
00453     * TargetTcpPose = getTargetTcpPose()
00454     *
00455     * @par JSON-RPC request example
00456     * {"jsonrpc":"2.0","method":"rob1.RobotState.getTargetTcpPose","params":[],"id":1}
00457     *
00458     * @par JSON-RPC response example
00459     * {"id":1,"jsonrpc":"2.0","result":
[0.4173932217619493,-0.13250000000000012,0.43296496133045825,3.141577313781914,0.0,1.5707963267948963]}
00460     * \endenglish
00461     */
00462     std::vector<double> getTargetTcpPose();
00463
00464     /**
00465     * @ingroup RobotState
00466     * \chinese
00467     * 获取工具端的位姿（不带 TCP 偏移）
00468     *
00469     * 位姿表示形式为 (x,y,z,rx,ry,rz)。
00470     * 其中 x、y、z 是法兰盘中心在基坐标系下的目标位置，单位是 m。
00471     * rx、ry、rz 是法兰盘中心在基坐标系下的目标姿态，是 ZYX 欧拉角，单位是 rad。
00472     *
00473     * @return 工具端的位姿，形式为 (x,y,z,rx,ry,rz)
00474     *
00475     * @throws arcs::common_interface::AuboException
00476     *
00477     * @par Python 函数原型
00478     * getToolPose(self: pyaubo_sdk.RobotState) -> List[float]
00479     *
00480     * @par Lua 函数原型
00481     * getToolPose() -> table
00482     *
00483     * @par Lua 示例
00484     * ToolPose = getToolPose()
00485     *
00486     * @par JSON-RPC 请求示例
00487     * {"jsonrpc":"2.0","method":"rob1.RobotState.getToolPose","params":[],"id":1}
00488     *
00489     * @par JSON-RPC 响应示例
00490     * {"id":1,"jsonrpc":"2.0","result":
[0.4177820858878617,-0.13250000000000012,0.20928410288421018,3.141579231257899,0.0,1.5707963267948963]}
00491     * \endchinese
00492     * \english
00493     * Get the tool pose (without TCP offset)
00494     *
00495     * The pose is represented as (x, y, z, rx, ry, rz).
00496     * x, y, z are the target position of the flange center in the base
00497     * coordinate system, in meters. rx, ry, rz are the target orientation of
00498     * the flange center in the base coordinate system, as ZYX Euler angles in
00499     * radians.
00500     *
00501     * @return The tool pose, in the form (x, y, z, rx, ry, rz)
00502     *
00503     * @throws arcs::common_interface::AuboException
00504     *
00505     * @par Python function prototype
00506     * getToolPose(self: pyaubo_sdk.RobotState) -> List[float]
00507     *
00508     * @par Lua function prototype
00509     * getToolPose() -> table
00510     *
00511     * @par Lua example
00512     * ToolPose = getToolPose()
00513     *

```

```

00514     * @par JSON-RPC request example
00515     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getToolPose", "params": [], "id": 1}
00516     *
00517     * @par JSON-RPC response example
00518     * {"id": 1, "jsonrpc": "2.0", "result":
[0.41777820858878617, -0.132500000000000012, 0.20928410288421018, 3.141579231257899, 0.0, 1.5707963267948963]}
00519     * \endenglish
00520     */
00521     std::vector<double> getToolPose();
00522
00523 /**
00524     * @ingroup RobotState
00525     * \chinese
00526     * 获取 TCP 速度
00527     *
00528     * @return TCP 速度
00529     *
00530     * @throws arcs::common_interface::AuboException
00531     *
00532     * @par Python 函数原型
00533     * getTcpSpeed(self: pyaubo_sdk.RobotState) -> List[float]
00534     *
00535     * @par Lua 函数原型
00536     * getTcpSpeed() -> table
00537     *
00538     * @par Lua 示例
00539     * TcpSpeed = getTcpSpeed()
00540     *
00541     * @par JSON-RPC 请求示例
00542     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getTcpSpeed", "params": [], "id": 1}
00543     *
00544     * @par JSON-RPC 响应示例
00545     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
00546     * \endchinese
00547     * \english
00548     * Get the TCP speed
00549     *
00550     * @return TCP speed
00551     *
00552     * @throws arcs::common_interface::AuboException
00553     *
00554     * @par Python function prototype
00555     * getTcpSpeed(self: pyaubo_sdk.RobotState) -> List[float]
00556     *
00557     * @par Lua function prototype
00558     * getTcpSpeed() -> table
00559     *
00560     * @par Lua example
00561     * TcpSpeed = getTcpSpeed()
00562     *
00563     * @par JSON-RPC request example
00564     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getTcpSpeed", "params": [], "id": 1}
00565     *
00566     * @par JSON-RPC response example
00567     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
00568     * \endenglish
00569     */
00570     std::vector<double> getTcpSpeed();
00571
00572 /**
00573     * @ingroup RobotState
00574     * \chinese
00575     * 获取 TCP 的力/力矩
00576     *
00577     * @return TCP 的力/力矩
00578     *
00579     * @par Python 函数原型
00580     * getTcpForce(self: pyaubo_sdk.RobotState) -> List[float]
00581     *
00582     * @par Lua 函数原型
00583     * getTcpForce() -> table
00584     *
00585     * @par Lua 示例
00586     * TcpForce = getTcpForce()
00587     *
00588     * @par JSON-RPC 请求示例
00589     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getTcpForce", "params": [], "id": 1}
00590     *
00591     * @par JSON-RPC 响应示例
00592     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
00593     * \endchinese
00594     * \english
00595     * Get the TCP force/torque
00596     *
00597     * @return TCP force/torque
00598     *
00599     * @par Python function prototype

```

```

00600     * getTcpForce(self: pyaubo_sdk.RobotState) -> List[float]
00601     *
00602     * @par Lua function prototype
00603     * getTcpForce() -> table
00604     *
00605     * @par Lua example
00606     * TcpForce = getTcpForce()
00607     *
00608     * @par JSON-RPC request example
00609     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getTcpForce", "params": [], "id": 1}
00610     *
00611     * @par JSON-RPC response example
00612     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
00613     * \endenglish
00614     */
00615     std::vector<double> getTcpForce();
00616
00617     /**
00618     * @ingroup RobotState
00619     * \chinese
00620     * 获取肘部的位置
00621     *
00622     * @return 肘部的位置
00623     *
00624     * @throws arcs::common_interface::AuboException
00625     *
00626     * @par Python 函数原型
00627     * getElbowPosistion(self: pyaubo_sdk.RobotState) -> List[float]
00628     *
00629     * @par Lua 函数原型
00630     * getElbowPosistion() -> table
00631     *
00632     * @par Lua 示例
00633     * ElbowPosistion = getElbowPosistion()
00634     *
00635     * @par JSON-RPC 请求示例
00636     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getElbowPosistion", "params": [], "id": 1}
00637     *
00638     * @par JSON-RPC 响应示例
00639     * {"id": 1, "jsonrpc": "2.0", "result":
00640     * [0.07355755887512408, -0.1325, 0.43200874126125227, -1.5707963267948968, 0.433006344376404, 0.0]}
00641     * \endchinese
00642     * \english
00643     * Get the position of the elbow
00644     *
00645     * @return The position of the elbow
00646     *
00647     * @throws arcs::common_interface::AuboException
00648     *
00649     * @par Python function prototype
00650     * getElbowPosistion(self: pyaubo_sdk.RobotState) -> List[float]
00651     *
00652     * @par Lua function prototype
00653     * getElbowPosistion() -> table
00654     *
00655     * @par Lua example
00656     * ElbowPosistion = getElbowPosistion()
00657     *
00658     * @par JSON-RPC request example
00659     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getElbowPosistion", "params": [], "id": 1}
00660     *
00661     * @par JSON-RPC response example
00662     * {"id": 1, "jsonrpc": "2.0", "result":
00663     * [0.07355755887512408, -0.1325, 0.43200874126125227, -1.5707963267948968, 0.433006344376404, 0.0]}
00664     * \endenglish
00665     */
00666     std::vector<double> getElbowPosistion();
00667
00668     /**
00669     * @ingroup RobotState
00670     * \chinese
00671     * 获取肘部速度
00672     *
00673     * @return 肘部速度
00674     *
00675     * @throws arcs::common_interface::AuboException
00676     *
00677     * @par Python 函数原型
00678     * getElbowVelocity(self: pyaubo_sdk.RobotState) -> List[float]
00679     *
00680     * @par Lua 函数原型
00681     * getElbowVelocity() -> table
00682     *
00683     * @par Lua 示例
00684     * ElbowVelocity = getElbowVelocity()
00685     *
00686     * @par JSON-RPC 请求示例

```

```

00685     * {"jsonrpc": "2.0", "method": "robl.RobotState.getElbowVelocity", "params": [], "id": 1}
00686     *
00687     * @par JSON-RPC 响应示例
00688     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
00689     * \endchinese
00690     * \english
00691     * Get the elbow velocity
00692     *
00693     * @return Elbow velocity
00694     *
00695     * @throws arcs::common_interface::AuboException
00696     *
00697     * @par Python function prototype
00698     * getElbowVelocity(self: pyaubo_sdk.RobotState) -> List[float]
00699     *
00700     * @par Lua function prototype
00701     * getElbowVelocity() -> table
00702     *
00703     * @par Lua example
00704     * ElbowVelocity = getElbowVelocity()
00705     *
00706     * @par JSON-RPC request example
00707     * {"jsonrpc": "2.0", "method": "robl.RobotState.getElbowVelocity", "params": [], "id": 1}
00708     *
00709     * @par JSON-RPC response example
00710     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
00711     * \endenglish
00712     */
00713     std::vector<double> getElbowVelocity();
00714
00715     /**
00716     * @ingroup RobotState
00717     * \chinese
00718     * 获取基座力/力矩
00719     *
00720     * @return 基座力/力矩
00721     *
00722     * @throws arcs::common_interface::AuboException
00723     *
00724     * @par Python 函数原型
00725     * getBaseForce(self: pyaubo_sdk.RobotState) -> List[float]
00726     *
00727     * @par Lua 函数原型
00728     * getBaseForce() -> table
00729     *
00730     * @par Lua 示例
00731     * BaseForce = getBaseForce()
00732     *
00733     * @par JSON-RPC 请求示例
00734     * {"jsonrpc": "2.0", "method": "robl.RobotState.getBaseForce", "params": [], "id": 1}
00735     *
00736     * @par JSON-RPC 响应示例
00737     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
00738     * \endchinese
00739     * \english
00740     * Get the base force/torque
00741     *
00742     * @return Base force/torque
00743     *
00744     * @throws arcs::common_interface::AuboException
00745     *
00746     * @par Python function prototype
00747     * getBaseForce(self: pyaubo_sdk.RobotState) -> List[float]
00748     *
00749     * @par Lua function prototype
00750     * getBaseForce() -> table
00751     *
00752     * @par Lua example
00753     * BaseForce = getBaseForce()
00754     *
00755     * @par JSON-RPC request example
00756     * {"jsonrpc": "2.0", "method": "robl.RobotState.getBaseForce", "params": [], "id": 1}
00757     *
00758     * @par JSON-RPC response example
00759     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
00760     * \endenglish
00761     */
00762     std::vector<double> getBaseForce();
00763
00764     /**
00765     * @ingroup RobotState
00766     * \chinese
00767     * 获取上一次发送的 TCP 目标位姿
00768     *
00769     * @return TCP 目标位姿
00770     *
00771     * @throws arcs::common_interface::AuboException

```

```

00772      *
00773      * @par Python 函数原型
00774      * getTcpTargetPose(self: pyaubo_sdk.RobotState) -> List[float]
00775      *
00776      * @par Lua 函数原型
00777      * getTcpTargetPose() -> table
00778      *
00779      * @par Lua 示例
00780      * TcpTargetPose = getTcpTargetPose()
00781      *
00782      * @par JSON-RPC 请求示例
00783      * {"jsonrpc": "2.0", "method": "rob1.RobotState.getTcpTargetPose", "params": [], "id": 1}
00784      *
00785      * @par JSON-RPC 响应示例
00786      * {"id": 1, "jsonrpc": "2.0", "result":
[0.41777829240862013, -0.132500000000000012, 0.2092832117232601, 3.1415812372223217, 0.0, 1.5707963267948963]}
00787      * \endchinese
00788      * \english
00789      * Get the last sent TCP target pose
00790      *
00791      * @return TCP target pose
00792      *
00793      * @throws arcs::common_interface::AuboException
00794      *
00795      * @par Python function prototype
00796      * getTcpTargetPose(self: pyaubo_sdk.RobotState) -> List[float]
00797      *
00798      * @par Lua function prototype
00799      * getTcpTargetPose() -> table
00800      *
00801      * @par Lua example
00802      * TcpTargetPose = getTcpTargetPose()
00803      *
00804      * @par JSON-RPC request example
00805      * {"jsonrpc": "2.0", "method": "rob1.RobotState.getTcpTargetPose", "params": [], "id": 1}
00806      *
00807      * @par JSON-RPC response example
00808      * {"id": 1, "jsonrpc": "2.0", "result":
[0.41777829240862013, -0.132500000000000012, 0.2092832117232601, 3.1415812372223217, 0.0, 1.5707963267948963]}
00809      * \endenglish
00810      */
00811      std::vector<double> getTcpTargetPose();
00812
00813      /**
00814      * @ingroup RobotState
00815      * \chinese
00816      * 获取 TCP 目标速度
00817      *
00818      * @return TCP 目标速度
00819      *
00820      * @throws arcs::common_interface::AuboException
00821      *
00822      * @par Python 函数原型
00823      * getTcpTargetSpeed(self: pyaubo_sdk.RobotState) -> List[float]
00824      *
00825      * @par Lua 函数原型
00826      * getTcpTargetSpeed() -> table
00827      *
00828      * @par Lua 示例
00829      * TcpTargetSpeed = getTcpTargetSpeed()
00830      *
00831      * @par JSON-RPC 请求示例
00832      * {"jsonrpc": "2.0", "method": "rob1.RobotState.getTcpTargetSpeed", "params": [], "id": 1}
00833      *
00834      * @par JSON-RPC 响应示例
00835      * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
00836      * \endchinese
00837      * \english
00838      * Get the TCP target speed
00839      *
00840      * @return TCP target speed
00841      *
00842      * @throws arcs::common_interface::AuboException
00843      *
00844      * @par Python function prototype
00845      * getTcpTargetSpeed(self: pyaubo_sdk.RobotState) -> List[float]
00846      *
00847      * @par Lua function prototype
00848      * getTcpTargetSpeed() -> table
00849      *
00850      * @par Lua example
00851      * TcpTargetSpeed = getTcpTargetSpeed()
00852      *
00853      * @par JSON-RPC request example
00854      * {"jsonrpc": "2.0", "method": "rob1.RobotState.getTcpTargetSpeed", "params": [], "id": 1}
00855      *
00856      * @par JSON-RPC response example

```

```

00857     * {"id":1,"jsonrpc":"2.0","result":[0.0,0.0,0.0,0.0,0.0,0.0]}
00858     * \endenglish
00859     */
00860 std::vector<double> getTcpTargetSpeed();
00861
00862 /**
00863  * @ingroup RobotState
00864  * \chinese
00865  * 获取 TCP 目标力/力矩
00866  *
00867  * @return TCP 目标力/力矩
00868  *
00869  * @throws arcs::common_interface::AuboException
00870  *
00871  * @par Python 函数原型
00872  * getTcpTargetForce(self: pyaubo_sdk.RobotState) -> List[float]
00873  *
00874  * @par Lua 函数原型
00875  * getTcpTargetForce() -> table
00876  *
00877  * @par Lua 示例
00878  * TcpTargetForce = getTcpTargetForce()
00879  *
00880  * @par JSON-RPC 请求示例
00881  * {"jsonrpc":"2.0","method":"rob1.RobotState.getTcpTargetForce","params":[],"id":1}
00882  *
00883  * @par JSON-RPC 响应示例
00884  * {"id":1,"jsonrpc":"2.0","result":[0.0,0.0,0.0,0.0,0.0,0.0]}
00885  * \endchinese
00886  * \english
00887  * Get the TCP target force/torque
00888  *
00889  * @return TCP target force/torque
00890  *
00891  * @throws arcs::common_interface::AuboException
00892  *
00893  * @par Python function prototype
00894  * getTcpTargetForce(self: pyaubo_sdk.RobotState) -> List[float]
00895  *
00896  * @par Lua function prototype
00897  * getTcpTargetForce() -> table
00898  *
00899  * @par Lua example
00900  * TcpTargetForce = getTcpTargetForce()
00901  *
00902  * @par JSON-RPC request example
00903  * {"jsonrpc":"2.0","method":"rob1.RobotState.getTcpTargetForce","params":[],"id":1}
00904  *
00905  * @par JSON-RPC response example
00906  * {"id":1,"jsonrpc":"2.0","result":[0.0,0.0,0.0,0.0,0.0,0.0]}
00907  * \endenglish
00908  */
00909 std::vector<double> getTcpTargetForce();
00910
00911 /**
00912  * @ingroup RobotState
00913  * \chinese
00914  * 获取机械臂关节标志
00915  *
00916  * @return 机械臂关节标志
00917  *
00918  * @throws arcs::common_interface::AuboException
00919  *
00920  * @par Python 函数原型
00921  * getJointState(self: pyaubo_sdk.RobotState) ->
00922  * List[arcs::common_interface::JointStateType]
00923  *
00924  * @par Lua 函数原型
00925  * getJointState() -> table
00926  *
00927  * @par Lua 示例
00928  * JointState = getJointState()
00929  *
00930  * @par JSON-RPC 请求示例
00931  * {"jsonrpc":"2.0","method":"rob1.RobotState.getJointState","params":[],"id":1}
00932  *
00933  * @par JSON-RPC 响应示例
00934  * {"id":1,"jsonrpc":"2.0","result":["Running","Running","Running","Running","Running","Running"]}
00935  * \endchinese
00936  * \english
00937  * Get the joint state of the manipulator
00938  *
00939  * @return Joint state of the manipulator
00940  *
00941  * @throws arcs::common_interface::AuboException
00942  *
00943  * @par Python function prototype

```

```

00944     * getJointState(self: pyaubo_sdk.RobotState) ->
00945     * List[arcs::common_interface::JointStateType]
00946     *
00947     * @par Lua function prototype
00948     * getJointState() -> table
00949     *
00950     * @par Lua example
00951     * JointState = getJointState()
00952     *
00953     * @par JSON-RPC request example
00954     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointState", "params": [], "id": 1}
00955     *
00956     * @par JSON-RPC response example
00957     * {"id": 1, "jsonrpc": "2.0", "result": ["Running", "Running", "Running", "Running", "Running", "Running"]}
00958     * \endenglish
00959     */
00960     std::vector<JointStateType> getJointState();
00961
00962     /**
00963     * @ingroup RobotState
00964     * \chinese
00965     * 获取关节的伺服状态
00966     *
00967     * @return 关节的伺服状态
00968     *
00969     * @throws arcs::common_interface::AuboException
00970     *
00971     * @par Python 函数原型
00972     * getJointServoMode(self: pyaubo_sdk.RobotState) ->
00973     * List[arcs::common_interface::JointServoModeType]
00974     *
00975     * @par Lua 函数原型
00976     * getJointServoMode() -> table
00977     *
00978     * @par Lua 示例
00979     * JointServoMode = getJointServoMode()
00980     *
00981     * @par JSON-RPC 请求示例
00982     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointServoMode", "params": [], "id": 1}
00983     *
00984     * @par JSON-RPC 响应示例
00985     * {"id": 1, "jsonrpc": "2.0", "result": ["Position", "Position", "Position", "Position", "Position", "Position"]}
00986     * \endchinese
00987     * \english
00988     * Get the servo state of the joints
00989     *
00990     * @return The servo state of the joints
00991     *
00992     * @throws arcs::common_interface::AuboException
00993     *
00994     * @par Python function prototype
00995     * getJointServoMode(self: pyaubo_sdk.RobotState) ->
00996     * List[arcs::common_interface::JointServoModeType]
00997     *
00998     * @par Lua function prototype
00999     * getJointServoMode() -> table
01000     *
01001     * @par Lua example
01002     * JointServoMode = getJointServoMode()
01003     *
01004     * @par JSON-RPC request example
01005     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointServoMode", "params": [], "id": 1}
01006     *
01007     * @par JSON-RPC response example
01008     * {"id": 1, "jsonrpc": "2.0", "result": ["Position", "Position", "Position", "Position", "Position", "Position"]}
01009     * \endenglish
01010     */
01011     std::vector<JointServoModeType> getJointServoMode();
01012
01013     /**
01014     * @ingroup RobotState
01015     * \chinese
01016     * 获取机械臂关节角度
01017     *
01018     * @return 机械臂关节角度
01019     *
01020     * @throws arcs::common_interface::AuboException
01021     *
01022     * @par Python 函数原型
01023     * getJointPositions(self: pyaubo_sdk.RobotState) -> List[float]
01024     *
01025     * @par Lua 函数原型
01026     * getJointPositions() -> table
01027     *
01028     * @par Lua 示例
01029     * JointPositions = getJointPositions()
01030     *

```



```

01031      * @par JSON-RPC 请求示例
01032      * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointPositions", "params": [], "id": 1}
01033      *
01034      * @par JSON-RPC 响应示例
01035      * {"id": 1, "jsonrpc": "2.0", "result":
[0.0, -0.26199241371495835, 1.7418102574563423, 0.4330197667082982, 1.5707963267948966, 0.0]}
01036      * \endchinese
01037      * \english
01038      * Get the joint positions of the manipulator
01039      *
01040      * @return Joint positions of the manipulator
01041      *
01042      * @throws arcs::common_interface::AuboException
01043      *
01044      * @par Python function prototype
01045      * getJointPositions(self: pyaubo_sdk.RobotState) -> List[float]
01046      *
01047      * @par Lua function prototype
01048      * getJointPositions() -> table
01049      *
01050      * @par Lua example
01051      * JointPositions = getJointPositions()
01052      *
01053      * @par JSON-RPC request example
01054      * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointPositions", "params": [], "id": 1}
01055      *
01056      * @par JSON-RPC response example
01057      * {"id": 1, "jsonrpc": "2.0", "result":
[0.0, -0.26199241371495835, 1.7418102574563423, 0.4330197667082982, 1.5707963267948966, 0.0]}
01058      * \endenglish
01059      */
01060      std::vector<double> getJointPositions();
01061
01062      /**
01063      * @ingroup RobotState
01064      * \chinese
01065      * 获取机械臂历史关节角度
01066      *
01067      * @param steps 步数
01068      * @return 机械臂历史关节角度
01069      * \endchinese
01070      * \english
01071      * Get the historical joint positions of the manipulator
01072      *
01073      * @param steps Number of steps
01074      * @return Historical joint positions of the manipulator
01075      * \endenglish
01076      */
01077      std::vector<double> getJointPositionsHistory(int steps);
01078
01079      /**
01080      * @ingroup RobotState
01081      * \chinese
01082      * 获取机械臂关节速度
01083      *
01084      * @return 机械臂关节速度
01085      *
01086      * @throws arcs::common_interface::AuboException
01087      *
01088      * @par Python 函数原型
01089      * getJointSpeeds(self: pyaubo_sdk.RobotState) -> List[float]
01090      *
01091      * @par Lua 函数原型
01092      * getJointSpeeds() -> table
01093      *
01094      * @par Lua 示例
01095      * JointSpeeds = getJointSpeeds()
01096      *
01097      * @par JSON-RPC 请求示例
01098      * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointSpeeds", "params": [], "id": 1}
01099      *
01100      * @par JSON-RPC 响应示例
01101      * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
01102      * \endchinese
01103      * \english
01104      * Get the joint speeds of the manipulator
01105      *
01106      * @return Joint speeds of the manipulator
01107      *
01108      * @throws arcs::common_interface::AuboException
01109      *
01110      * @par Python function prototype
01111      * getJointSpeeds(self: pyaubo_sdk.RobotState) -> List[float]
01112      *
01113      * @par Lua function prototype
01114      * getJointSpeeds() -> table
01115      *

```

```

01116     * @par Lua example
01117     * JointSpeeds = getJointSpeeds()
01118     *
01119     * @par JSON-RPC request example
01120     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointSpeeds", "params": [], "id": 1}
01121     *
01122     * @par JSON-RPC response example
01123     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
01124     * \endenglish
01125     */
01126     std::vector<double> getJointSpeeds();
01127
01128     /**
01129     * @ingroup RobotState
01130     * \chinese
01131     * 获取机械臂关节加速度
01132     *
01133     * @return 机械臂关节加速度
01134     *
01135     * @throws arcs::common_interface::AuboException
01136     *
01137     * @par Python 函数原型
01138     * getJointAccelerations(self: pyaubo_sdk.RobotState) -> List[float]
01139     *
01140     * @par Lua 函数原型
01141     * getJointAccelerations() -> table
01142     *
01143     * @par Lua 示例
01144     * JointAccelerations = getJointAccelerations()
01145     *
01146     * @par JSON-RPC 请求示例
01147     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointAccelerations", "params": [], "id": 1}
01148     *
01149     * @par JSON-RPC 响应示例
01150     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
01151     * \endchinese
01152     * \english
01153     * Get the joint accelerations of the manipulator
01154     *
01155     * @return Joint accelerations of the manipulator
01156     *
01157     * @throws arcs::common_interface::AuboException
01158     *
01159     * @par Python function prototype
01160     * getJointAccelerations(self: pyaubo_sdk.RobotState) -> List[float]
01161     *
01162     * @par Lua function prototype
01163     * getJointAccelerations() -> table
01164     *
01165     * @par Lua example
01166     * JointAccelerations = getJointAccelerations()
01167     *
01168     * @par JSON-RPC request example
01169     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointAccelerations", "params": [], "id": 1}
01170     *
01171     * @par JSON-RPC response example
01172     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
01173     * \endenglish
01174     */
01175     std::vector<double> getJointAccelerations();
01176
01177     /**
01178     * @ingroup RobotState
01179     * \chinese
01180     * 获取机械臂关节力矩
01181     *
01182     * @return 机械臂关节力矩
01183     *
01184     * @throws arcs::common_interface::AuboException
01185     *
01186     * @par Python 函数原型
01187     * getJointTorqueSensors(self: pyaubo_sdk.RobotState) -> List[float]
01188     *
01189     * @par Lua 函数原型
01190     * getJointTorqueSensors() -> table
01191     *
01192     * @par Lua 示例
01193     * JointTorqueSensors = getJointTorqueSensors()
01194     *
01195     * @par JSON-RPC 请求示例
01196     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointTorqueSensors", "params": [], "id": 1}
01197     *
01198     * @par JSON-RPC 响应示例
01199     * {"id": 1, "jsonrpc": "2.0", "result":
01200     [0.0, 6275.367736816406, -7704.2816162109375, 3586.9766235351563, 503.0364990234375, 1506.0882568359375]}
01201     * \endchinese
01202     * \english

```

```

01202     * Get the joint torques of the manipulator
01203     *
01204     * @return Joint torques of the manipulator
01205     *
01206     * @throws arcs::common_interface::AuboException
01207     *
01208     * @par Python function prototype
01209     * getJointTorqueSensors(self: pyaubo_sdk.RobotState) -> List[float]
01210     *
01211     * @par Lua function prototype
01212     * getJointTorqueSensors() -> table
01213     *
01214     * @par Lua example
01215     * JointTorqueSensors = getJointTorqueSensors()
01216     *
01217     * @par JSON-RPC request example
01218     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointTorqueSensors", "params": [], "id": 1}
01219     *
01220     * @par JSON-RPC response example
01221     * {"id": 1, "jsonrpc": "2.0", "result":
[0.0, 6275.367736816406, -7704.2816162109375, 3586.9766235351563, 503.0364990234375, 1506.0882568359375]}
01222     * \endenglish
01223     */
01224     std::vector<double> getJointTorqueSensors();
01225
01226     /**
01227     * @ingroup RobotState
01228     * \chinese
01229     * 获取机械臂关节接触力矩（外力矩）
01230     *
01231     * @return 机械臂关节接触力矩
01232     *
01233     * @throws arcs::common_interface::AuboException
01234     *
01235     * @par Python 函数原型
01236     * getJointContactTorques(self: pyaubo_sdk.RobotState) -> List[float]
01237     *
01238     * @par Lua 函数原型
01239     * getJointContactTorques() -> table
01240     *
01241     * @par Lua 示例
01242     * JointContactTorques = getJointContactTorques()
01243     *
01244     * @par JSON-RPC 请求示例
01245     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointContactTorques", "params": [], "id": 1}
01246     *
01247     * @par JSON-RPC 响应示例
01248     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
01249     * \endchinese
01250     * \english
01251     * Get the joint contact torques (external torques) of the manipulator
01252     *
01253     * @return Joint contact torques of the manipulator
01254     *
01255     * @throws arcs::common_interface::AuboException
01256     *
01257     * @par Python function prototype
01258     * getJointContactTorques(self: pyaubo_sdk.RobotState) -> List[float]
01259     *
01260     * @par Lua function prototype
01261     * getJointContactTorques() -> table
01262     *
01263     * @par Lua example
01264     * JointContactTorques = getJointContactTorques()
01265     *
01266     * @par JSON-RPC request example
01267     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointContactTorques", "params": [], "id": 1}
01268     *
01269     * @par JSON-RPC response example
01270     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
01271     * \endenglish
01272     */
01273     std::vector<double> getJointContactTorques();
01274
01275     /**
01276     * @ingroup RobotState
01277     * \chinese
01278     * 获取机械臂关节重力矩
01279     *
01280     * @return 机械臂关节重力矩，包含 6 个关节的力矩值（单位：Nm）
01281     *
01282     * @throws arcs::common_interface::AuboException
01283     *
01284     * @par Python 函数原型
01285     * getJointGravityTorques(self: pyaubo_sdk.RobotState) -> List[float]
01286     *
01287     * @par Lua 函数原型

```

```

01288     * getJointGravityTorques() -> table
01289     *
01290     * @par Lua 示例
01291     * local jointGravityTorques = getJointGravityTorques()
01292     *
01293     * @par JSON-RPC 请求示例
01294     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointGravityTorques", "params": [], "id": 1}
01295     *
01296     * @par JSON-RPC 响应示例
01297     * {"id": 1, "jsonrpc": "2.0", "result": [1.23, -2.34, 3.45, -4.56, 0.78, -0.12]}
01298     * \endchinese
01299     * \english
01300     * Get the joint gravity torque of the manipulator
01301     *
01302     * @return Joint gravity torque of the manipulator, containing torque values
01303     * for 6 joints (unit: Nm)
01304     *
01305     * @throws arcs::common_interface::AuboException
01306     *
01307     * @par Python function prototype
01308     * getJointGravityTorques(self: pyaubo_sdk.RobotState) -> List[float]
01309     *
01310     * @par Lua function prototype
01311     * getJointGravityTorques() -> table
01312     *
01313     * @par Lua example
01314     * local jointGravityTorques = getJointGravityTorques()
01315     *
01316     * @par JSON-RPC request example
01317     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointGravityTorques", "params": [], "id": 1}
01318     *
01319     * @par JSON-RPC response example
01320     * {"id": 1, "jsonrpc": "2.0", "result": [1.23, -2.34, 3.45, -4.56, 0.78, -0.12]}
01321     * \endenglish
01322     */
01323 std::vector<double> getJointGravityTorques();
01324
01325 /**
01326     * \chinese
01327     * 获取底座力传感器读数
01328     *
01329     * @return 底座力传感器读数
01330     *
01331     * @throws arcs::common_interface::AuboException
01332     *
01333     * @par Python 函数原型
01334     * getBaseForceSensor(self: pyaubo_sdk.RobotState) -> List[float]
01335     *
01336     * @par Lua 函数原型
01337     * getBaseForceSensor() -> table
01338     *
01339     * @par Lua 示例
01340     * BaseForceSensor = getBaseForceSensor()
01341     *
01342     * @par JSON-RPC 请求示例
01343     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getBaseForceSensor", "params": [], "id": 1}
01344     *
01345     * @par JSON-RPC 响应示例
01346     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
01347     * \endchinese
01348     * \english
01349     * Get the base force sensor readings
01350     *
01351     * @return Base force sensor readings
01352     *
01353     * @throws arcs::common_interface::AuboException
01354     *
01355     * @par Python function prototype
01356     * getBaseForceSensor(self: pyaubo_sdk.RobotState) -> List[float]
01357     *
01358     * @par Lua function prototype
01359     * getBaseForceSensor() -> table
01360     *
01361     * @par Lua example
01362     * BaseForceSensor = getBaseForceSensor()
01363     *
01364     * @par JSON-RPC request example
01365     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getBaseForceSensor", "params": [], "id": 1}
01366     *
01367     * @par JSON-RPC response example
01368     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
01369     * \endenglish
01370     */
01371 std::vector<double> getBaseForceSensor();
01372
01373 /**
01374     * @ingroup RobotState

```

```

01375 * \chinese
01376 * 获取 TCP 力传感器读数
01377 *
01378 * @return TCP 力传感器读数
01379 *
01380 * @throws arcs::common_interface::AuboException
01381 *
01382 * @par Python 函数原型
01383 * getTcpForceSensors(self: pyaubo_sdk.RobotState) -> List[float]
01384 *
01385 * @par Lua 函数原型
01386 * getTcpForceSensors() -> table
01387 *
01388 * @par Lua 示例
01389 * TcpForceSensors = getTcpForceSensors()
01390 *
01391 * @par JSON-RPC 请求示例
01392 * {"jsonrpc": "2.0", "method": "rob1.RobotState.getTcpForceSensors", "params": [], "id": 1}
01393 *
01394 * @par JSON-RPC 响应示例
01395 * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
01396 * \endchinese
01397 * \english
01398 * Get the TCP force sensor readings
01399 *
01400 * @return TCP force sensor readings
01401 *
01402 * @throws arcs::common_interface::AuboException
01403 *
01404 * @par Python function prototype
01405 * getTcpForceSensors(self: pyaubo_sdk.RobotState) -> List[float]
01406 *
01407 * @par Lua function prototype
01408 * getTcpForceSensors() -> table
01409 *
01410 * @par Lua example
01411 * TcpForceSensors = getTcpForceSensors()
01412 *
01413 * @par JSON-RPC request example
01414 * {"jsonrpc": "2.0", "method": "rob1.RobotState.getTcpForceSensors", "params": [], "id": 1}
01415 *
01416 * @par JSON-RPC response example
01417 * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
01418 * \endenglish
01419 */
01420 std::vector<double> getTcpForceSensors();
01421
01422 /**
01423 * @ingroup RobotState
01424 * \chinese
01425 * 获取机械臂关节电流
01426 *
01427 * @return 机械臂关节电流
01428 *
01429 * @throws arcs::common_interface::AuboException
01430 *
01431 * @par Python 函数原型
01432 * getJointCurrents(self: pyaubo_sdk.RobotState) -> List[float]
01433 *
01434 * @par Lua 函数原型
01435 * getJointCurrents() -> table
01436 *
01437 * @par Lua 示例
01438 * JointCurrents = getJointCurrents()
01439 *
01440 * @par JSON-RPC 请求示例
01441 * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointCurrents", "params": [], "id": 1}
01442 *
01443 * @par JSON-RPC 响应示例
01444 * {"id": 1, "jsonrpc": "2.0", "result":
[0.0, 1.25885009765625, -1.5289306640625, 0.71868896484375, 0.1007080078125, 0.3021240234375]}
01445 * \endchinese
01446 * \english
01447 * Get the joint currents of the manipulator
01448 *
01449 * @return Joint currents of the manipulator
01450 *
01451 * @throws arcs::common_interface::AuboException
01452 *
01453 * @par Python function prototype
01454 * getJointCurrents(self: pyaubo_sdk.RobotState) -> List[float]
01455 *
01456 * @par Lua function prototype
01457 * getJointCurrents() -> table
01458 *
01459 * @par Lua example
01460 * JointCurrents = getJointCurrents()

```

```

01461      *
01462      * @par JSON-RPC request example
01463      * {"jsonrpc": "2.0", "method": "robl.RobotState.getJointCurrents", "params": [], "id": 1}
01464      *
01465      * @par JSON-RPC response example
01466      * {"id": 1, "jsonrpc": "2.0", "result":
[0.0, 1.25885009765625, -1.5289306640625, 0.71868896484375, 0.1007080078125, 0.3021240234375]}
01467      * \endenglish
01468      */
01469      std::vector<double> getJointCurrents();
01470
01471      /**
01472      * @ingroup RobotState
01473      * \chinese
01474      * 获取机械臂关节电压
01475      *
01476      * @return 机械臂关节电压
01477      *
01478      * @throws arcs::common_interface::AuboException
01479      *
01480      * @par Python 函数原型
01481      * getJointVoltages(self: pyaubo_sdk.RobotState) -> List[float]
01482      *
01483      * @par Lua 函数原型
01484      * getJointVoltages() -> table
01485      *
01486      * @par Lua 示例
01487      * JointVoltages = getJointVoltages()
01488      *
01489      * @par JSON-RPC 请求示例
01490      * {"jsonrpc": "2.0", "method": "robl.RobotState.getJointVoltages", "params": [], "id": 1}
01491      *
01492      * @par JSON-RPC 响应示例
01493      * {"id": 1, "jsonrpc": "2.0", "result": [2.0, 2.5, 3.0, 2.0, 2.5, 2.0]}
01494      * \endchinese
01495      * \english
01496      * Get the joint voltages of the manipulator
01497      *
01498      * @return Joint voltages of the manipulator
01499      *
01500      * @throws arcs::common_interface::AuboException
01501      *
01502      * @par Python function prototype
01503      * getJointVoltages(self: pyaubo_sdk.RobotState) -> List[float]
01504      *
01505      * @par Lua function prototype
01506      * getJointVoltages() -> table
01507      *
01508      * @par Lua example
01509      * JointVoltages = getJointVoltages()
01510      *
01511      * @par JSON-RPC request example
01512      * {"jsonrpc": "2.0", "method": "robl.RobotState.getJointVoltages", "params": [], "id": 1}
01513      *
01514      * @par JSON-RPC response example
01515      * {"id": 1, "jsonrpc": "2.0", "result": [2.0, 2.5, 3.0, 2.0, 2.5, 2.0]}
01516      * \endenglish
01517      */
01518      std::vector<double> getJointVoltages();
01519
01520      /**
01521      * @ingroup RobotState
01522      * \chinese
01523      * 获取机械臂关节温度
01524      *
01525      * @return 机械臂关节温度
01526      *
01527      * @throws arcs::common_interface::AuboException
01528      *
01529      * @par Python 函数原型
01530      * getJointTemperatures(self: pyaubo_sdk.RobotState) -> List[float]
01531      *
01532      * @par Lua 函数原型
01533      * getJointTemperatures() -> table
01534      *
01535      * @par Lua 示例
01536      * JointTemperatures = getJointTemperatures()
01537      *
01538      * @par JSON-RPC 请求示例
01539      * {"jsonrpc": "2.0", "method": "robl.RobotState.getJointTemperatures", "params": [], "id": 1}
01540      *
01541      * @par JSON-RPC 响应示例
01542      * {"id": 1, "jsonrpc": "2.0", "result": [38.0, 38.0, 38.0, 39.0, 38.0, 39.0]}
01543      * \endchinese
01544      * \english
01545      * Get the joint temperatures of the manipulator
01546      *

```

```

01547     * @return Joint temperatures of the manipulator
01548     *
01549     * @throws arcs::common_interface::AuboException
01550     *
01551     * @par Python function prototype
01552     * getJointTemperatures(self: pyaubo_sdk.RobotState) -> List[float]
01553     *
01554     * @par Lua function prototype
01555     * getJointTemperatures() -> table
01556     *
01557     * @par Lua example
01558     * JointTemperatures = getJointTemperatures()
01559     *
01560     * @par JSON-RPC request example
01561     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointTemperatures", "params": [], "id": 1}
01562     *
01563     * @par JSON-RPC response example
01564     * {"id": 1, "jsonrpc": "2.0", "result": [38.0, 38.0, 38.0, 39.0, 38.0, 39.0]}
01565     * \endenglish
01566     */
01567     std::vector<double> getJointTemperatures();
01568
01569     /**
01570     * @ingroup RobotState
01571     * \chinese
01572     * 获取关节全球唯一 ID
01573     *
01574     * @return 关节全球唯一 ID
01575     *
01576     * @throws arcs::common_interface::AuboException
01577     *
01578     * @par Python 函数原型
01579     * getJointUniqueIds(self: pyaubo_sdk.RobotState) -> List[str]
01580     *
01581     * @par Lua 函数原型
01582     * getJointUniqueIds() -> table
01583     *
01584     * @par Lua 示例
01585     * JointUniqueIds = getJointUniqueIds()
01586     *
01587     * @par JSON-RPC 请求示例
01588     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointUniqueIds", "params": [], "id": 1}
01589     *
01590     * @par JSON-RPC 响应示例
01591     * {"id": 1, "jsonrpc": "2.0", "result":
["00800020ffffffff31405153", "00800020ffffffff3e3f5153", "00800020ffffffff414b5153", "00800020ffffffff31065153", "00800020ffffffff41535153",
01592     * \endchinese
01593     * \english
01594     * Get the globally unique IDs of the joints
01595     *
01596     * @return Globally unique IDs of the joints
01597     *
01598     * @throws arcs::common_interface::AuboException
01599     *
01600     * @par Python function prototype
01601     * getJointUniqueIds(self: pyaubo_sdk.RobotState) -> List[str]
01602     *
01603     * @par Lua function prototype
01604     * getJointUniqueIds() -> table
01605     *
01606     * @par Lua example
01607     * JointUniqueIds = getJointUniqueIds()
01608     *
01609     * @par JSON-RPC request example
01610     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointUniqueIds", "params": [], "id": 1}
01611     *
01612     * @par JSON-RPC response example
01613     * {"id": 1, "jsonrpc": "2.0", "result":
["00800020ffffffff31405153", "00800020ffffffff3e3f5153", "00800020ffffffff414b5153", "00800020ffffffff31065153", "00800020ffffffff41535153",
01614     * \endenglish
01615     */
01616     std::vector<std::string> getJointUniqueIds();
01617
01618     /**
01619     * @ingroup RobotState
01620     * \chinese
01621     * 获取关节固件版本
01622     *
01623     * @return 关节固件版本
01624     *
01625     * @throws arcs::common_interface::AuboException
01626     *
01627     * @par Python 函数原型
01628     * getJointFirmwareVersions(self: pyaubo_sdk.RobotState) -> List[int]
01629     *
01630     * @par Lua 函数原型
01631     * getJointFirmwareVersions() -> table

```

```

01632     *
01633     * @par Lua 示例
01634     * JointFirmwareVersions = getJointFirmwareVersions()
01635     *
01636     * @par JSON-RPC 请求示例
01637     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointFirmwareVersions", "params": [], "id": 1}
01638     *
01639     * @par JSON-RPC 响应示例
01640     * {"id": 1, "jsonrpc": "2.0", "result": [1000010, 1000010, 1000010, 1000010, 1000010, 1000010]}
01641     * \endchinese
01642     * \english
01643     * Get the joint firmware versions
01644     *
01645     * @return Joint firmware versions
01646     *
01647     * @throws arcs::common_interface::AuboException
01648     *
01649     * @par Python function prototype
01650     * getJointFirmwareVersions(self: pyaubo_sdk.RobotState) -> List[int]
01651     *
01652     * @par Lua function prototype
01653     * getJointFirmwareVersions() -> table
01654     *
01655     * @par Lua example
01656     * JointFirmwareVersions = getJointFirmwareVersions()
01657     *
01658     * @par JSON-RPC request example
01659     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointFirmwareVersions", "params": [], "id": 1}
01660     *
01661     * @par JSON-RPC response example
01662     * {"id": 1, "jsonrpc": "2.0", "result": [1000010, 1000010, 1000010, 1000010, 1000010, 1000010]}
01663     * \endenglish
01664     */
01665 std::vector<int> getJointFirmwareVersions();
01666
01667 /**
01668     * @ingroup RobotState
01669     * \chinese
01670     * 获取关节硬件版本
01671     *
01672     * @return 关节硬件版本
01673     *
01674     * @throws arcs::common_interface::AuboException
01675     *
01676     * @par Python 函数原型
01677     * getJointHardwareVersions(self: pyaubo_sdk.RobotState) -> List[int]
01678     *
01679     * @par Lua 函数原型
01680     * getJointHardwareVersions() -> table
01681     *
01682     * @par Lua 示例
01683     * JointHardwareVersions = getJointHardwareVersions()
01684     *
01685     * @par JSON-RPC 请求示例
01686     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointHardwareVersions", "params": [], "id": 1}
01687     *
01688     * @par JSON-RPC 响应示例
01689     * {"id": 1, "jsonrpc": "2.0", "result": [1000000, 1000000, 1004000, 1004000, 1004000, 1004000]}
01690     * \endchinese
01691     * \english
01692     * Get the joint hardware versions
01693     *
01694     * @return Joint hardware versions
01695     *
01696     * @throws arcs::common_interface::AuboException
01697     *
01698     * @par Python function prototype
01699     * getJointHardwareVersions(self: pyaubo_sdk.RobotState) -> List[int]
01700     *
01701     * @par Lua function prototype
01702     * getJointHardwareVersions() -> table
01703     *
01704     * @par Lua example
01705     * JointHardwareVersions = getJointHardwareVersions()
01706     *
01707     * @par JSON-RPC request example
01708     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointHardwareVersions", "params": [], "id": 1}
01709     *
01710     * @par JSON-RPC response example
01711     * {"id": 1, "jsonrpc": "2.0", "result": [1000000, 1000000, 1004000, 1004000, 1004000, 1004000]}
01712     * \endenglish
01713     */
01714 std::vector<int> getJointHardwareVersions();
01715
01716 /**
01717     * @ingroup RobotState
01718     * \chinese

```



```

01719 * 获取 MasterBoard 全球唯一 ID
01720 *
01721 * @return MasterBoard 全球唯一 ID
01722 *
01723 * @throws arcs::common_interface::AuboException
01724 *
01725 * @par Python 函数原型
01726 * getMasterBoardUniqueId(self: pyaubo_sdk.RobotState) -> str
01727 *
01728 * @par Lua 函数原型
01729 * getMasterBoardUniqueId() -> string
01730 *
01731 * @par Lua 示例
01732 * MasterBoardUniqueId = getMasterBoardUniqueId()
01733 *
01734 * @par JSON-RPC 请求示例
01735 * {"jsonrpc": "2.0", "method": "rob1.RobotState.getMasterBoardUniqueId", "params": [], "id": 1}
01736 *
01737 * @par JSON-RPC 响应示例
01738 * {"id": 1, "jsonrpc": "2.0", "result": "001e0044510f343037323637"}
01739 * \endchinese
01740 * \english
01741 * Get the globally unique ID of the MasterBoard
01742 *
01743 * @return Globally unique ID of the MasterBoard
01744 *
01745 * @throws arcs::common_interface::AuboException
01746 *
01747 * @par Python function prototype
01748 * getMasterBoardUniqueId(self: pyaubo_sdk.RobotState) -> str
01749 *
01750 * @par Lua function prototype
01751 * getMasterBoardUniqueId() -> string
01752 *
01753 * @par Lua example
01754 * MasterBoardUniqueId = getMasterBoardUniqueId()
01755 *
01756 * @par JSON-RPC request example
01757 * {"jsonrpc": "2.0", "method": "rob1.RobotState.getMasterBoardUniqueId", "params": [], "id": 1}
01758 *
01759 * @par JSON-RPC response example
01760 * {"id": 1, "jsonrpc": "2.0", "result": "001e0044510f343037323637"}
01761 * \endenglish
01762 */
01763 std::string getMasterBoardUniqueId();
01764
01765 /**
01766 * @ingroup RobotState
01767 * \chinese
01768 * 获取 MasterBoard 固件版本
01769 *
01770 * @return MasterBoard 固件版本
01771 *
01772 * @throws arcs::common_interface::AuboException
01773 *
01774 * @par Python 函数原型
01775 * getMasterBoardFirmwareVersion(self: pyaubo_sdk.RobotState) -> int
01776 *
01777 * @par Lua 函数原型
01778 * getMasterBoardFirmwareVersion() -> number
01779 *
01780 * @par Lua 示例
01781 * MasterBoardFirmwareVersion = getMasterBoardFirmwareVersion()
01782 *
01783 * @par JSON-RPC 请求示例
01784 * {"jsonrpc": "2.0", "method": "rob1.RobotState.getMasterBoardFirmwareVersion", "params": [], "id": 1}
01785 *
01786 * @par JSON-RPC 响应示例
01787 * {"id": 1, "jsonrpc": "2.0", "result": 1000004}
01788 * \endchinese
01789 * \english
01790 * Get the MasterBoard firmware version
01791 *
01792 * @return MasterBoard firmware version
01793 *
01794 * @throws arcs::common_interface::AuboException
01795 *
01796 * @par Python function prototype
01797 * getMasterBoardFirmwareVersion(self: pyaubo_sdk.RobotState) -> int
01798 *
01799 * @par Lua function prototype
01800 * getMasterBoardFirmwareVersion() -> number
01801 *
01802 * @par Lua example
01803 * MasterBoardFirmwareVersion = getMasterBoardFirmwareVersion()
01804 *
01805 * @par JSON-RPC request example

```

```

01806     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getMasterBoardFirmwareVersion", "params": [], "id": 1}
01807     *
01808     * @par JSON-RPC response example
01809     * {"id": 1, "jsonrpc": "2.0", "result": 1000004}
01810     * \endenglish
01811     */
01812     int getMasterBoardFirmwareVersion();
01813
01814     /**
01815     * @ingroup RobotState
01816     * \chinese
01817     * 获取 MasterBoard 硬件版本
01818     *
01819     * @return MasterBoard 硬件版本
01820     *
01821     * @throws arcs::common_interface::AuboException
01822     *
01823     * @par Python 函数原型
01824     * getMasterBoardHardwareVersion(self: pyaubo_sdk.RobotState) -> int
01825     *
01826     * @par Lua 函数原型
01827     * getMasterBoardHardwareVersion() -> number
01828     *
01829     * @par Lua 示例
01830     * MasterBoardHardwareVersion = getMasterBoardHardwareVersion()
01831     *
01832     * @par JSON-RPC 请求示例
01833     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getMasterBoardHardwareVersion", "params": [], "id": 1}
01834     *
01835     * @par JSON-RPC 响应示例
01836     * {"id": 1, "jsonrpc": "2.0", "result": 1000000}
01837     * \endchinese
01838     * \english
01839     * Get the MasterBoard hardware version
01840     *
01841     * @return MasterBoard hardware version
01842     *
01843     * @throws arcs::common_interface::AuboException
01844     *
01845     * @par Python function prototype
01846     * getMasterBoardHardwareVersion(self: pyaubo_sdk.RobotState) -> int
01847     *
01848     * @par Lua function prototype
01849     * getMasterBoardHardwareVersion() -> number
01850     *
01851     * @par Lua example
01852     * MasterBoardHardwareVersion = getMasterBoardHardwareVersion()
01853     *
01854     * @par JSON-RPC request example
01855     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getMasterBoardHardwareVersion", "params": [], "id": 1}
01856     *
01857     * @par JSON-RPC response example
01858     * {"id": 1, "jsonrpc": "2.0", "result": 1000000}
01859     * \endenglish
01860     */
01861     int getMasterBoardHardwareVersion();
01862
01863     /**
01864     * @ingroup RobotState
01865     * \chinese
01866     * 获取 SlaveBoard 全球唯一 ID
01867     *
01868     * @return SlaveBoard 全球唯一 ID
01869     *
01870     * @throws arcs::common_interface::AuboException
01871     *
01872     * @par Python 函数原型
01873     * getSlaveBoardUniqueId(self: pyaubo_sdk.RobotState) -> str
01874     *
01875     * @par Lua 函数原型
01876     * getSlaveBoardUniqueId() -> string
01877     *
01878     * @par Lua 示例
01879     * SlaveBoardUniqueId = getSlaveBoardUniqueId()
01880     *
01881     * @par JSON-RPC 请求示例
01882     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getSlaveBoardUniqueId", "params": [], "id": 1}
01883     *
01884     * @par JSON-RPC 响应示例
01885     * {"id": 1, "jsonrpc": "2.0", "result": "73657263008000000000000000000000"}
01886     * \endchinese
01887     * \english
01888     * Get the globally unique ID of the SlaveBoard
01889     *
01890     * @return Globally unique ID of the SlaveBoard
01891     *
01892     * @throws arcs::common_interface::AuboException

```

```

01893     *
01894     * @par Python function prototype
01895     * getSlaveBoardUniqueId(self: pyaubo_sdk.RobotState) -> str
01896     *
01897     * @par Lua function prototype
01898     * getSlaveBoardUniqueId() -> string
01899     *
01900     * @par Lua example
01901     * SlaveBoardUniqueId = getSlaveBoardUniqueId()
01902     *
01903     * @par JSON-RPC request example
01904     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getSlaveBoardUniqueId", "params": [], "id": 1}
01905     *
01906     * @par JSON-RPC response example
01907     * {"id": 1, "jsonrpc": "2.0", "result": "736572630080000000000000"}
01908     * \endenglish
01909     */
01910     std::string getSlaveBoardUniqueId();
01911
01912     /**
01913     * @ingroup RobotState
01914     * \chinese
01915     * 获取 SlaveBoard 固件版本
01916     *
01917     * @return SlaveBoard 固件版本
01918     *
01919     * @throws arcs::common_interface::AuboException
01920     *
01921     * @par Python 函数原型
01922     * getSlaveBoardFirmwareVersion(self: pyaubo_sdk.RobotState) -> int
01923     *
01924     * @par Lua 函数原型
01925     * getSlaveBoardFirmwareVersion() -> number
01926     *
01927     * @par Lua 示例
01928     * SlaveBoardFirmwareVersion = getSlaveBoardFirmwareVersion()
01929     *
01930     * @par JSON-RPC 请求示例
01931     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getSlaveBoardFirmwareVersion", "params": [], "id": 1}
01932     *
01933     * @par JSON-RPC 响应示例
01934     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01935     * \endchinese
01936     * \english
01937     * Get the SlaveBoard firmware version
01938     *
01939     * @return SlaveBoard firmware version
01940     *
01941     * @throws arcs::common_interface::AuboException
01942     *
01943     * @par Python function prototype
01944     * getSlaveBoardFirmwareVersion(self: pyaubo_sdk.RobotState) -> int
01945     *
01946     * @par Lua function prototype
01947     * getSlaveBoardFirmwareVersion() -> number
01948     *
01949     * @par Lua example
01950     * SlaveBoardFirmwareVersion = getSlaveBoardFirmwareVersion()
01951     *
01952     * @par JSON-RPC request example
01953     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getSlaveBoardFirmwareVersion", "params": [], "id": 1}
01954     *
01955     * @par JSON-RPC response example
01956     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01957     * \endenglish
01958     */
01959     int getSlaveBoardFirmwareVersion();
01960
01961     /**
01962     * @ingroup RobotState
01963     * \chinese
01964     * 获取 SlaveBoard 硬件版本
01965     *
01966     * @return SlaveBoard 硬件版本
01967     *
01968     * @throws arcs::common_interface::AuboException
01969     *
01970     * @par Python 函数原型
01971     * getSlaveBoardHardwareVersion(self: pyaubo_sdk.RobotState) -> int
01972     *
01973     * @par Lua 函数原型
01974     * getSlaveBoardHardwareVersion() -> number
01975     *
01976     * @par Lua 示例
01977     * SlaveBoardHardwareVersion = getSlaveBoardHardwareVersion()
01978     *
01979     * @par JSON-RPC 请求示例

```

```

01980     * {"jsonrpc": "2.0", "method": "robl.RobotState.getSlaveBoardHardwareVersion", "params": [], "id": 1}
01981     *
01982     * @par JSON-RPC 响应示例
01983     * {"id": 1, "jsonrpc": "2.0", "result": 6030098}
01984     * \endchinese
01985     * \english
01986     * Get the SlaveBoard hardware version
01987     *
01988     * @return SlaveBoard hardware version
01989     *
01990     * @throws arcs::common_interface::AuboException
01991     *
01992     * @par Python function prototype
01993     * getSlaveBoardHardwareVersion(self: pyaubo_sdk.RobotState) -> int
01994     *
01995     * @par Lua function prototype
01996     * getSlaveBoardHardwareVersion() -> number
01997     *
01998     * @par Lua example
01999     * SlaveBoardHardwareVersion = getSlaveBoardHardwareVersion()
02000     *
02001     * @par JSON-RPC request example
02002     * {"jsonrpc": "2.0", "method": "robl.RobotState.getSlaveBoardHardwareVersion", "params": [], "id": 1}
02003     *
02004     * @par JSON-RPC response example
02005     * {"id": 1, "jsonrpc": "2.0", "result": 6030098}
02006     * \endenglish
02007     */
02008 int getSlaveBoardHardwareVersion();
02009
02010 /**
02011  * @ingroup RobotState
02012  * \chinese
02013  * 获取 Handle 全球唯一 ID
02014  *
02015  * @return Handle 全球唯一 ID
02016  *
02017  * @throws arcs::common_interface::AuboException
02018  *
02019  * @par Python 函数原型
02020  * getHandleUniqueId(self: pyaubo_sdk.RobotState) -> str
02021  *
02022  * @par Lua 函数原型
02023  * getHandleUniqueId() -> string
02024  *
02025  * @par Lua 示例
02026  * HandleUniqueId = getHandleUniqueId()
02027  *
02028  * @par JSON-RPC 请求示例
02029  * {"jsonrpc": "2.0", "method": "robl.RobotState.getHandleUniqueId", "params": [], "id": 1}
02030  *
02031  * @par JSON-RPC 响应示例
02032  * {"id": 1, "jsonrpc": "2.0", "result": "73657263008000000000000000"}
02033  * \endchinese
02034  * \english
02035  * Get the globally unique ID of the Handle
02036  *
02037  * @return Globally unique ID of the Handle
02038  *
02039  * @throws arcs::common_interface::AuboException
02040  *
02041  * @par Python function prototype
02042  * getHandleUniqueId(self: pyaubo_sdk.RobotState) -> str
02043  *
02044  * @par Lua function prototype
02045  * getHandleUniqueId() -> string
02046  *
02047  * @par Lua example
02048  * HandleUniqueId = getHandleUniqueId()
02049  *
02050  * @par JSON-RPC request example
02051  * {"jsonrpc": "2.0", "method": "robl.RobotState.getHandleUniqueId", "params": [], "id": 1}
02052  *
02053  * @par JSON-RPC response example
02054  * {"id": 1, "jsonrpc": "2.0", "result": "73657263008000000000000000"}
02055  * \endenglish
02056  */
02057 std::string getHandleUniqueId();
02058
02059 /**
02060  * @ingroup RobotState
02061  * \chinese
02062  * 获取 Handle 固件版本
02063  *
02064  * @return Handle 固件版本
02065  *
02066  * @throws arcs::common_interface::AuboException

```

```

02067      *
02068      * @par Python 函数原型
02069      * getHandleFirmwareVersion(self: pyaubo_sdk.RobotState) -> int
02070      *
02071      * @par Lua 函数原型
02072      * getHandleFirmwareVersion() -> number
02073      *
02074      * @par Lua 示例
02075      * HandleFirmwareVersion = getHandleFirmwareVersion()
02076      *
02077      * @par JSON-RPC 请求示例
02078      * {"jsonrpc": "2.0", "method": "rob1.RobotState.getHandleFirmwareVersion", "params": [], "id": 1}
02079      *
02080      * @par JSON-RPC 响应示例
02081      * {"id": 1, "jsonrpc": "2.0", "result": 0}
02082      * \endchinese
02083      * \english
02084      * Get the Handle firmware version
02085      *
02086      * @return Handle firmware version
02087      *
02088      * @throws arcs::common_interface::AuboException
02089      *
02090      * @par Python function prototype
02091      * getHandleFirmwareVersion(self: pyaubo_sdk.RobotState) -> int
02092      *
02093      * @par Lua function prototype
02094      * getHandleFirmwareVersion() -> number
02095      *
02096      * @par Lua example
02097      * HandleFirmwareVersion = getHandleFirmwareVersion()
02098      *
02099      * @par JSON-RPC request example
02100      * {"jsonrpc": "2.0", "method": "rob1.RobotState.getHandleFirmwareVersion", "params": [], "id": 1}
02101      *
02102      * @par JSON-RPC response example
02103      * {"id": 1, "jsonrpc": "2.0", "result": 0}
02104      * \endenglish
02105      */
02106  int getHandleFirmwareVersion();
02107
02108  /**
02109   * @ingroup RobotState
02110   * \chinese
02111   * 获取 Handle 硬件版本
02112   *
02113   * @return Handle 硬件版本
02114   *
02115   * @throws arcs::common_interface::AuboException
02116   *
02117   * @par Python 函数原型
02118   * getHandleHardwareVersion(self: pyaubo_sdk.RobotState) -> int
02119   *
02120   * @par Lua 函数原型
02121   * getHandleHardwareVersion() -> number
02122   *
02123   * @par Lua 示例
02124   * HandleHardwareVersion = getHandleHardwareVersion()
02125   *
02126   * @par JSON-RPC 请求示例
02127   * {"jsonrpc": "2.0", "method": "rob1.RobotState.getHandleHardwareVersion", "params": [], "id": 1}
02128   *
02129   * @par JSON-RPC 响应示例
02130   * {"id": 1, "jsonrpc": "2.0", "result": 6030098}
02131   * \endchinese
02132   * \english
02133   * Get the Handle hardware version
02134   *
02135   * @return Handle hardware version
02136   *
02137   * @throws arcs::common_interface::AuboException
02138   *
02139   * @par Python function prototype
02140   * getHandleHardwareVersion(self: pyaubo_sdk.RobotState) -> int
02141   *
02142   * @par Lua function prototype
02143   * getHandleHardwareVersion() -> number
02144   *
02145   * @par Lua example
02146   * HandleHardwareVersion = getHandleHardwareVersion()
02147   *
02148   * @par JSON-RPC request example
02149   * {"jsonrpc": "2.0", "method": "rob1.RobotState.getHandleHardwareVersion", "params": [], "id": 1}
02150   *
02151   * @par JSON-RPC response example
02152   * {"id": 1, "jsonrpc": "2.0", "result": 6030098}
02153   * \endenglish

```

```

02154     */
02155     int getHandleHardwareVersion();
02156
02157     /**
02158     * @ingroup RobotState
02159     * \chinese
02160     * 获取 Handle 状态
02161     *
02162     * @return Handle 状态
02163     *
02164     * @throws arcs::common_interface::AuboException
02165     *
02166     * @par Python 函数原型
02167     * getHandleState(self: pyaubo_sdk.RobotState) -> int
02168     *
02169     * @par Lua 函数原型
02170     * getHandleState() -> number
02171     *
02172     * @par Lua 示例
02173     * HandleState = getHandleState()
02174     *
02175     * @par JSON-RPC 请求示例
02176     * {"jsonrpc":"2.0","method":"rob1.RobotState.getHandleState","params":[],"id":1}
02177     *
02178     * @par JSON-RPC 响应示例
02179     * {"id":1,"jsonrpc":"2.0","result":6030098}
02180     * \endchinese
02181     * \english
02182     * Get the Handle state
02183     *
02184     * @return Handle state
02185     *
02186     * @throws arcs::common_interface::AuboException
02187     *
02188     * @par Python function prototype
02189     * getHandleState(self: pyaubo_sdk.RobotState) -> int
02190     *
02191     * @par Lua function prototype
02192     * getHandleState() -> number
02193     *
02194     * @par Lua example
02195     * HandleState = getHandleState()
02196     *
02197     * @par JSON-RPC request example
02198     * {"jsonrpc":"2.0","method":"rob1.RobotState.getHandleState","params":[],"id":1}
02199     *
02200     * @par JSON-RPC response example
02201     * {"id":1,"jsonrpc":"2.0","result":6030098}
02202     * \endenglish
02203     */
02204     HandleStateType getHandleState();
02205
02206     /**
02207     * @ingroup RobotState
02208     * \chinese
02209     * 获取 Handle 类型
02210     *
02211     * @return Handle 类型
02212     *
02213     * @throws arcs::common_interface::AuboException
02214     *
02215     * @par Python 函数原型
02216     * getHandleMode(self: pyaubo_sdk.RobotState) -> int
02217     *
02218     * @par Lua 函数原型
02219     * getHandleMode() -> number
02220     *
02221     * @par Lua 示例
02222     * HandleMode = getHandleMode()
02223     *
02224     * @par JSON-RPC 请求示例
02225     * {"jsonrpc":"2.0","method":"rob1.RobotState.getHandleMode","params":[],"id":1}
02226     *
02227     * @par JSON-RPC 响应示例
02228     * {"id":1,"jsonrpc":"2.0","result":6030098}
02229     * \endchinese
02230     * \english
02231     * Get the Handle mode
02232     *
02233     * @return Handle mode
02234     *
02235     * @throws arcs::common_interface::AuboException
02236     *
02237     * @par Python function prototype
02238     * getHandleMode(self: pyaubo_sdk.RobotState) -> int
02239     *
02240     * @par Lua function prototype

```

```

02241     * getHandleMode() -> number
02242     *
02243     * @par Lua example
02244     * HandleMode = getHandleMode()
02245     *
02246     * @par JSON-RPC request example
02247     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getHandleMode", "params": [], "id": 1}
02248     *
02249     * @par JSON-RPC response example
02250     * {"id": 1, "jsonrpc": "2.0", "result": 1}
02251     * \endenglish
02252     */
02253 HandleModeType getHandleMode();
02254
02255 /**
02256  * @ingroup RobotState
02257  * \chinese
02258  * 获取 PowerlossBoard 全球唯一 ID
02259  *
02260  * @return PowerlossBoard 全球唯一 ID
02261  *
02262  * @throws arcs::common_interface::AuboException
02263  *
02264  * @par Python 函数原型
02265  * getPowerlossBoardUniqueId(self: pyaubo_sdk.RobotState) -> str
02266  *
02267  * @par Lua 函数原型
02268  * getPowerlossBoardUniqueId() -> string
02269  *
02270  * @par Lua 示例
02271  * PowerlossBoardUniqueId = getPowerlossBoardUniqueId()
02272  *
02273  * @par JSON-RPC 请求示例
02274  * {"jsonrpc": "2.0", "method": "rob1.RobotState.getPowerlossBoardUniqueId", "params": [], "id": 1}
02275  *
02276  * @par JSON-RPC 响应示例
02277  * {"id": 1, "jsonrpc": "2.0", "result": "73657263008000000000000000"}
02278  * \endchinese
02279  * \english
02280  * Get the globally unique ID of the PowerlossBoard
02281  *
02282  * @return Globally unique ID of the PowerlossBoard
02283  *
02284  * @throws arcs::common_interface::AuboException
02285  *
02286  * @par Python function prototype
02287  * getPowerlossBoardUniqueId(self: pyaubo_sdk.RobotState) -> str
02288  *
02289  * @par Lua function prototype
02290  * getPowerlossBoardUniqueId() -> string
02291  *
02292  * @par Lua example
02293  * PowerlossBoardUniqueId = getPowerlossBoardUniqueId()
02294  *
02295  * @par JSON-RPC request example
02296  * {"jsonrpc": "2.0", "method": "rob1.RobotState.getPowerlossBoardUniqueId", "params": [], "id": 1}
02297  *
02298  * @par JSON-RPC response example
02299  * {"id": 1, "jsonrpc": "2.0", "result": "73657263008000000000000000"}
02300  * \endenglish
02301  */
02302 std::string getPowerlossBoardUniqueId();
02303
02304 /**
02305  * @ingroup RobotState
02306  * \chinese
02307  * 获取 PowerlossBoard 固件版本
02308  *
02309  * @return PowerlossBoard 固件版本
02310  *
02311  * @throws arcs::common_interface::AuboException
02312  *
02313  * @par Python 函数原型
02314  * getPowerlossBoardFirmwareVersion(self: pyaubo_sdk.RobotState) -> int
02315  *
02316  * @par Lua 函数原型
02317  * getPowerlossBoardFirmwareVersion() -> number
02318  *
02319  * @par Lua 示例
02320  * PowerlossBoardFirmwareVersion = getPowerlossBoardFirmwareVersion()
02321  *
02322  * @par JSON-RPC 请求示例
02323  * {"jsonrpc": "2.0", "method": "rob1.RobotState.getPowerlossBoardFirmwareVersion", "params": [], "id": 1}
02324  *
02325  * @par JSON-RPC 响应示例
02326  * {"id": 1, "jsonrpc": "2.0", "result": 0}
02327  * \endchinese

```

```

02328     * \english
02329     * Get the PowerlossBoard firmware version
02330     *
02331     * @return PowerlossBoard firmware version
02332     *
02333     * @throws arcs::common_interface::AuboException
02334     *
02335     * @par Python function prototype
02336     * getPowerlossBoardFirmwareVersion(self: pyaubo_sdk.RobotState) -> int
02337     *
02338     * @par Lua function prototype
02339     * getPowerlossBoardFirmwareVersion() -> number
02340     *
02341     * @par Lua example
02342     * PowerlossBoardFirmwareVersion = getPowerlossBoardFirmwareVersion()
02343     *
02344     * @par JSON-RPC request example
02345     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getPowerlossBoardFirmwareVersion", "params": [], "id": 1}
02346     *
02347     * @par JSON-RPC response example
02348     * {"id": 1, "jsonrpc": "2.0", "result": 0}
02349     * \endenglish
02350     */
02351 int getPowerlossBoardFirmwareVersion();
02352
02353 /**
02354     * @ingroup RobotState
02355     * \chinese
02356     * 获取 PowerlossBoard 硬件版本
02357     *
02358     * @return PowerlossBoard 硬件版本
02359     *
02360     * @throws arcs::common_interface::AuboException
02361     *
02362     * @par Python 函数原型
02363     * getPowerlossBoardHardwareVersion(self: pyaubo_sdk.RobotState) -> int
02364     *
02365     * @par Lua 函数原型
02366     * getPowerlossBoardHardwareVersion() -> number
02367     *
02368     * @par Lua 示例
02369     * PowerlossBoardHardwareVersion = getPowerlossBoardHardwareVersion()
02370     *
02371     * @par JSON-RPC 请求示例
02372     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getPowerlossBoardHardwareVersion", "params": [], "id": 1}
02373     *
02374     * @par JSON-RPC 响应示例
02375     * {"id": 1, "jsonrpc": "2.0", "result": 6030098}
02376     * \endchinese
02377     * \english
02378     * Get the PowerlossBoard hardware version
02379     *
02380     * @return PowerlossBoard hardware version
02381     *
02382     * @throws arcs::common_interface::AuboException
02383     *
02384     * @par Python function prototype
02385     * getPowerlossBoardHardwareVersion(self: pyaubo_sdk.RobotState) -> int
02386     *
02387     * @par Lua function prototype
02388     * getPowerlossBoardHardwareVersion() -> number
02389     *
02390     * @par Lua example
02391     * PowerlossBoardHardwareVersion = getPowerlossBoardHardwareVersion()
02392     *
02393     * @par JSON-RPC request example
02394     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getPowerlossBoardHardwareVersion", "params": [], "id": 1}
02395     *
02396     * @par JSON-RPC response example
02397     * {"id": 1, "jsonrpc": "2.0", "result": 6030098}
02398     * \endenglish
02399     */
02400 int getPowerlossBoardHardwareVersion();
02401
02402 /**
02403     * @ingroup RobotState
02404     * \chinese
02405     * 获取工具端全球唯一 ID
02406     *
02407     * @return 工具端全球唯一 ID
02408     *
02409     * @throws arcs::common_interface::AuboException
02410     *
02411     * @par Python 函数原型
02412     * getToolUniqueId(self: pyaubo_sdk.RobotState) -> str
02413     *
02414     * @par Lua 函数原型

```



```

02415     * getToolUniqueId() -> string
02416     *
02417     * @par Lua 示例
02418     * ToolUniqueId = getToolUniqueId()
02419     *
02420     * @par JSON-RPC 请求示例
02421     * {"jsonrpc": "2.0", "method": "robl.RobotState.getToolUniqueId", "params": [], "id": 1}
02422     *
02423     * @par JSON-RPC 响应示例
02424     * {"id": 1, "jsonrpc": "2.0", "result": "397d4e5331541252314d3042"}
02425     * \endchinese
02426     * \english
02427     * Get the globally unique ID of the tool
02428     *
02429     * @return Globally unique ID of the tool
02430     *
02431     * @throws arcs::common_interface::AuboException
02432     *
02433     * @par Python function prototype
02434     * getToolUniqueId(self: pyaubo_sdk.RobotState) -> str
02435     *
02436     * @par Lua function prototype
02437     * getToolUniqueId() -> string
02438     *
02439     * @par Lua example
02440     * ToolUniqueId = getToolUniqueId()
02441     *
02442     * @par JSON-RPC request example
02443     * {"jsonrpc": "2.0", "method": "robl.RobotState.getToolUniqueId", "params": [], "id": 1}
02444     *
02445     * @par JSON-RPC response example
02446     * {"id": 1, "jsonrpc": "2.0", "result": "397d4e5331541252314d3042"}
02447     * \endenglish
02448     */
02449     std::string getToolUniqueId();
02450
02451     /**
02452     * @ingroup RobotState
02453     * \chinese
02454     * 获取工具端固件版本
02455     *
02456     * @return 工具端固件版本
02457     *
02458     * @throws arcs::common_interface::AuboException
02459     *
02460     * @par Python 函数原型
02461     * getToolFirmwareVersion(self: pyaubo_sdk.RobotState) -> int
02462     *
02463     * @par Lua 函数原型
02464     * getToolFirmwareVersion() -> number
02465     *
02466     * @par Lua 示例
02467     * ToolFirmwareVersion = getToolFirmwareVersion()
02468     *
02469     * @par JSON-RPC 请求示例
02470     * {"jsonrpc": "2.0", "method": "robl.RobotState.getToolFirmwareVersion", "params": [], "id": 1}
02471     *
02472     * @par JSON-RPC 响应示例
02473     * {"id": 1, "jsonrpc": "2.0", "result": 1001003}
02474     * \endchinese
02475     * \english
02476     * Get the tool firmware version
02477     *
02478     * @return Tool firmware version
02479     *
02480     * @throws arcs::common_interface::AuboException
02481     *
02482     * @par Python function prototype
02483     * getToolFirmwareVersion(self: pyaubo_sdk.RobotState) -> int
02484     *
02485     * @par Lua function prototype
02486     * getToolFirmwareVersion() -> number
02487     *
02488     * @par Lua example
02489     * ToolFirmwareVersion = getToolFirmwareVersion()
02490     *
02491     * @par JSON-RPC request example
02492     * {"jsonrpc": "2.0", "method": "robl.RobotState.getToolFirmwareVersion", "params": [], "id": 1}
02493     *
02494     * @par JSON-RPC response example
02495     * {"id": 1, "jsonrpc": "2.0", "result": 1001003}
02496     * \endenglish
02497     */
02498     int getToolFirmwareVersion();
02499
02500     /**
02501     * @ingroup RobotState

```

```

02502     * \chinese
02503     * 获取工具端硬件版本
02504     *
02505     * @return 工具端硬件版本
02506     *
02507     * @throws arcs::common_interface::AuboException
02508     *
02509     * @par Python 函数原型
02510     * getToolHardwareVersion(self: pyaubo_sdk.RobotState) -> int
02511     *
02512     * @par Lua 函数原型
02513     * getToolHardwareVersion() -> number
02514     *
02515     * @par Lua 示例
02516     * ToolHardwareVersion = getToolHardwareVersion()
02517     *
02518     * @par JSON-RPC 请求示例
02519     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getToolHardwareVersion", "params": [], "id": 1}
02520     *
02521     * @par JSON-RPC 响应示例
02522     * {"id": 1, "jsonrpc": "2.0", "result": 1000000}
02523     * \endchinese
02524     * \english
02525     * Get the tool hardware version
02526     *
02527     * @return Tool hardware version
02528     *
02529     * @throws arcs::common_interface::AuboException
02530     *
02531     * @par Python function prototype
02532     * getToolHardwareVersion(self: pyaubo_sdk.RobotState) -> int
02533     *
02534     * @par Lua function prototype
02535     * getToolHardwareVersion() -> number
02536     *
02537     * @par Lua example
02538     * ToolHardwareVersion = getToolHardwareVersion()
02539     *
02540     * @par JSON-RPC request example
02541     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getToolHardwareVersion", "params": [], "id": 1}
02542     *
02543     * @par JSON-RPC response example
02544     * {"id": 1, "jsonrpc": "2.0", "result": 1000000}
02545     * \endenglish
02546     */
02547 int getToolHardwareVersion();
02548
02549 /**
02550     * @ingroup RobotState
02551     * \chinese
02552     * 获取末端通信模式
02553     *
02554     * @return 末端通信模式
02555     * 0: 表示无串口
02556     * 1: 表示只有串口
02557     * 2: 表示带力传感器和串口
02558     *
02559     * @throws arcs::common_interface::AuboException
02560     *
02561     * @par Python 函数原型
02562     * getToolCommMode(self: pyaubo_sdk.RobotState) -> int
02563     *
02564     * @par Lua 函数原型
02565     * getToolCommMode() -> number
02566     *
02567     * @par Lua 示例
02568     * ToolCommMode = getToolCommMode()
02569     *
02570     * @par JSON-RPC 请求示例
02571     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getToolCommMode", "params": [], "id": 1}
02572     *
02573     * @par JSON-RPC 响应示例
02574     * {"id": 1, "jsonrpc": "2.0", "result": 1}
02575     * \endchinese
02576     * \english
02577     * Get the tool communication mode
02578     *
02579     * @return Tool communication mode
02580     * 0: No serial port
02581     * 1: Serial port only
02582     * 2: With force sensor and serial port
02583     *
02584     * @throws arcs::common_interface::AuboException
02585     *
02586     * @par Python function prototype
02587     * getToolCommMode(self: pyaubo_sdk.RobotState) -> int
02588     *

```

```

02589     * @par Lua function prototype
02590     * getToolCommMode() -> number
02591     *
02592     * @par Lua example
02593     * ToolCommMode = getToolCommMode()
02594     *
02595     * @par JSON-RPC request example
02596     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getToolCommMode", "params": [], "id": 1}
02597     *
02598     * @par JSON-RPC response example
02599     * {"id": 1, "jsonrpc": "2.0", "result": 1}
02600     * \endenglish
02601     */
02602     int getToolCommMode();
02603
02604     /**
02605     * @ingroup RobotState
02606     * \chinese
02607     * 获取底座全球唯一 ID
02608     *
02609     * @return 底座全球唯一 ID
02610     *
02611     * @throws arcs::common_interface::AuboException
02612     *
02613     * @par Python 函数原型
02614     * getPedestalUniqueId(self: pyaubo_sdk.RobotState) -> str
02615     *
02616     * @par Lua 函数原型
02617     * getPedestalUniqueId() -> string
02618     *
02619     * @par Lua 示例
02620     * PedestalUniqueId = getPedestalUniqueId()
02621     *
02622     * @par JSON-RPC 请求示例
02623     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getPedestalUniqueId", "params": [], "id": 1}
02624     *
02625     * @par JSON-RPC 响应示例
02626     * {"id": 1, "jsonrpc": "2.0", "result": "205257533543065248544339"}
02627     * \endchinese
02628     * \english
02629     * Get the globally unique ID of the pedestal
02630     *
02631     * @return Globally unique ID of the pedestal
02632     *
02633     * @throws arcs::common_interface::AuboException
02634     *
02635     * @par Python function prototype
02636     * getPedestalUniqueId(self: pyaubo_sdk.RobotState) -> str
02637     *
02638     * @par Lua function prototype
02639     * getPedestalUniqueId() -> string
02640     *
02641     * @par Lua example
02642     * PedestalUniqueId = getPedestalUniqueId()
02643     *
02644     * @par JSON-RPC request example
02645     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getPedestalUniqueId", "params": [], "id": 1}
02646     *
02647     * @par JSON-RPC response example
02648     * {"id": 1, "jsonrpc": "2.0", "result": "205257533543065248544339"}
02649     * \endenglish
02650     */
02651     std::string getPedestalUniqueId();
02652
02653     /**
02654     * @ingroup RobotState
02655     * \chinese
02656     * 获取底座固件版本
02657     *
02658     * @return 底座固件版本
02659     *
02660     * @throws arcs::common_interface::AuboException
02661     *
02662     * @par Python 函数原型
02663     * getPedestalFirmwareVersion(self: pyaubo_sdk.RobotState) -> int
02664     *
02665     * @par Lua 函数原型
02666     * getPedestalFirmwareVersion() -> number
02667     *
02668     * @par Lua 示例
02669     * PedestalFirmwareVersion = getPedestalFirmwareVersion()
02670     *
02671     * @par JSON-RPC 请求示例
02672     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getPedestalFirmwareVersion", "params": [], "id": 1}
02673     *
02674     * @par JSON-RPC 响应示例
02675     * {"id": 1, "jsonrpc": "2.0", "result": 1000004}

```

```

02676     * \endchinese
02677     * \english
02678     * Get the pedestal firmware version
02679     *
02680     * @return Pedestal firmware version
02681     *
02682     * @throws arcs::common_interface::AuboException
02683     *
02684     * @par Python function prototype
02685     * getPedestalFirmwareVersion(self: pyaubo_sdk.RobotState) -> int
02686     *
02687     * @par Lua function prototype
02688     * getPedestalFirmwareVersion() -> number
02689     *
02690     * @par Lua example
02691     * PedestalFirmwareVersion = getPedestalFirmwareVersion()
02692     *
02693     * @par JSON-RPC request example
02694     * {"jsonrpc": "2.0", "method": "robl.RobotState.getPedestalFirmwareVersion", "params": [], "id": 1}
02695     *
02696     * @par JSON-RPC response example
02697     * {"id": 1, "jsonrpc": "2.0", "result": 1000004}
02698     * \endenglish
02699     */
02700 int getPedestalFirmwareVersion();
02701
02702 /**
02703     * @ingroup RobotState
02704     * \chinese
02705     * 获取底座硬件版本
02706     *
02707     * @return 底座硬件版本
02708     *
02709     * @throws arcs::common_interface::AuboException
02710     *
02711     * @par Python 函数原型
02712     * getPedestalHardwareVersion(self: pyaubo_sdk.RobotState) -> int
02713     *
02714     * @par Lua 函数原型
02715     * getPedestalHardwareVersion() -> number
02716     *
02717     * @par Lua 示例
02718     * PedestalHardwareVersion = getPedestalHardwareVersion()
02719     *
02720     * @par JSON-RPC 请求示例
02721     * {"jsonrpc": "2.0", "method": "robl.RobotState.getPedestalHardwareVersion", "params": [], "id": 1}
02722     *
02723     * @par JSON-RPC 响应示例
02724     * {"id": 1, "jsonrpc": "2.0", "result": 1007000}
02725     * \endchinese
02726     * \english
02727     * Get the pedestal hardware version
02728     *
02729     * @return Pedestal hardware version
02730     *
02731     * @throws arcs::common_interface::AuboException
02732     *
02733     * @par Python function prototype
02734     * getPedestalHardwareVersion(self: pyaubo_sdk.RobotState) -> int
02735     *
02736     * @par Lua function prototype
02737     * getPedestalHardwareVersion() -> number
02738     *
02739     * @par Lua example
02740     * PedestalHardwareVersion = getPedestalHardwareVersion()
02741     *
02742     * @par JSON-RPC request example
02743     * {"jsonrpc": "2.0", "method": "robl.RobotState.getPedestalHardwareVersion", "params": [], "id": 1}
02744     *
02745     * @par JSON-RPC response example
02746     * {"id": 1, "jsonrpc": "2.0", "result": 1007000}
02747     * \endenglish
02748     */
02749 int getPedestalHardwareVersion();
02750
02751 /**
02752     * @ingroup RobotState
02753     * \chinese
02754     * 获取机械臂关节目标位置角度
02755     *
02756     * @return 机械臂关节目标位置角度
02757     *
02758     * @throws arcs::common_interface::AuboException
02759     *
02760     * @par Python 函数原型
02761     * getJointTargetPositions(self: pyaubo_sdk.RobotState) -> List[float]
02762     *

```

```

02763      * @par Lua 函数原型
02764      * getJointTargetPositions() -> table
02765      *
02766      * @par Lua 示例
02767      * JointTargetPositions = getJointTargetPositions()
02768      *
02769      * @par JSON-RPC 请求示例
02770      * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointTargetPositions", "params": [], "id": 1}
02771      *
02772      * @par JSON-RPC 响应示例
02773      * {"id": 1, "jsonrpc": "2.0", "result":
[0.0, -0.2619944355631239, 1.7418124015308052, 0.4330219266665035, 1.5707963267948966, 0.0]}
02774      * \endchinese
02775      * \english
02776      * Get the target joint positions (angles) of the manipulator
02777      *
02778      * @return Target joint positions (angles) of the manipulator
02779      *
02780      * @throws arcs::common_interface::AuboException
02781      *
02782      * @par Python function prototype
02783      * getJointTargetPositions(self: pyaubo_sdk.RobotState) -> List[float]
02784      *
02785      * @par Lua function prototype
02786      * getJointTargetPositions() -> table
02787      *
02788      * @par Lua example
02789      * JointTargetPositions = getJointTargetPositions()
02790      *
02791      * @par JSON-RPC request example
02792      * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointTargetPositions", "params": [], "id": 1}
02793      *
02794      * @par JSON-RPC response example
02795      * {"id": 1, "jsonrpc": "2.0", "result":
[0.0, -0.2619944355631239, 1.7418124015308052, 0.4330219266665035, 1.5707963267948966, 0.0]}
02796      * \endenglish
02797      */
02798      std::vector<double> getJointTargetPositions();
02799
02800      /**
02801      * @ingroup RobotState
02802      * \chinese
02803      * 获取机械臂关节目标速度
02804      *
02805      * @return 机械臂关节目标速度
02806      *
02807      * @throws arcs::common_interface::AuboException
02808      *
02809      * @par Python 函数原型
02810      * getJointTargetSpeeds(self: pyaubo_sdk.RobotState) -> List[float]
02811      *
02812      * @par Lua 函数原型
02813      * getJointTargetSpeeds() -> table
02814      *
02815      * @par Lua 示例
02816      * JointTargetSpeeds = getJointTargetSpeeds()
02817      *
02818      * @par JSON-RPC 请求示例
02819      * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointTargetSpeeds", "params": [], "id": 1}
02820      *
02821      * @par JSON-RPC 响应示例
02822      * {"id": 1, "jsonrpc": "2.0", "result":
[0.0, 0.00024227101509399773, 0.0016521760307419697, 0.0026521060731088397, 0.0, 0.0]}
02823      * \endchinese
02824      * \english
02825      * Get the target joint speeds of the manipulator
02826      *
02827      * @return Target joint speeds of the manipulator
02828      *
02829      * @throws arcs::common_interface::AuboException
02830      *
02831      * @par Python function prototype
02832      * getJointTargetSpeeds(self: pyaubo_sdk.RobotState) -> List[float]
02833      *
02834      * @par Lua function prototype
02835      * getJointTargetSpeeds() -> table
02836      *
02837      * @par Lua example
02838      * JointTargetSpeeds = getJointTargetSpeeds()
02839      *
02840      * @par JSON-RPC request example
02841      * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointTargetSpeeds", "params": [], "id": 1}
02842      *
02843      * @par JSON-RPC response example
02844      * {"id": 1, "jsonrpc": "2.0", "result":
[0.0, 0.00024227101509399773, 0.0016521760307419697, 0.0026521060731088397, 0.0, 0.0]}
02845      * \endenglish

```

```

02846     */
02847     std::vector<double> getJointTargetSpeeds();
02848
02849     /**
02850     * @ingroup RobotState
02851     * \chinese
02852     * 获取机械臂关节目标加速度
02853     *
02854     * @return 机械臂关节目标加速度
02855     *
02856     * @throws arcs::common_interface::AuboException
02857     *
02858     * @par Python 函数原型
02859     * getJointTargetAccelerations(self: pyaubo_sdk.RobotState) -> List[float]
02860     *
02861     * @par Lua 函数原型
02862     * getJointTargetAccelerations() -> table
02863     *
02864     * @par Lua 示例
02865     * JointTargetAccelerations = getJointTargetAccelerations()
02866     *
02867     * @par JSON-RPC 请求示例
02868     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointTargetAccelerations", "params": [], "id": 1}
02869     *
02870     * @par JSON-RPC 响应示例
02871     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, -0.6737932929246071, -12.610253240108449, 0.0, 0.0, 0.0]}
02872     * \endchinese
02873     * \english
02874     * Get the target joint accelerations of the manipulator
02875     *
02876     * @return Target joint accelerations of the manipulator
02877     *
02878     * @throws arcs::common_interface::AuboException
02879     *
02880     * @par Python function prototype
02881     * getJointTargetAccelerations(self: pyaubo_sdk.RobotState) -> List[float]
02882     *
02883     * @par Lua function prototype
02884     * getJointTargetAccelerations() -> table
02885     *
02886     * @par Lua example
02887     * JointTargetAccelerations = getJointTargetAccelerations()
02888     *
02889     * @par JSON-RPC request example
02890     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointTargetAccelerations", "params": [], "id": 1}
02891     *
02892     * @par JSON-RPC response example
02893     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, -0.6737932929246071, -12.610253240108449, 0.0, 0.0, 0.0]}
02894     * \endenglish
02895     */
02896     std::vector<double> getJointTargetAccelerations();
02897
02898     /**
02899     * @ingroup RobotState
02900     * \chinese
02901     * 获取机械臂关节目标力矩
02902     *
02903     * @return 机械臂关节目标力矩
02904     *
02905     * @throws arcs::common_interface::AuboException
02906     *
02907     * @par Python 函数原型
02908     * getJointTargetTorques(self: pyaubo_sdk.RobotState) -> List[float]
02909     *
02910     * @par Lua 函数原型
02911     * getJointTargetTorques() -> table
02912     *
02913     * @par Lua 示例
02914     * JointTargetTorques = getJointTargetTorques()
02915     *
02916     * @par JSON-RPC 请求示例
02917     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointTargetTorques", "params": [], "id": 1}
02918     *
02919     * @par JSON-RPC 响应示例
02920     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
02921     * \endchinese
02922     * \english
02923     * Get the target joint torques of the manipulator
02924     *
02925     * @return Target joint torques of the manipulator
02926     *
02927     * @throws arcs::common_interface::AuboException
02928     *
02929     * @par Python function prototype
02930     * getJointTargetTorques(self: pyaubo_sdk.RobotState) -> List[float]
02931     *
02932     * @par Lua function prototype

```

```

02933     * getJointTargetTorques() -> table
02934     *
02935     * @par Lua example
02936     * JointTargetTorques = getJointTargetTorques()
02937     *
02938     * @par JSON-RPC request example
02939     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointTargetTorques", "params": [], "id": 1}
02940     *
02941     * @par JSON-RPC response example
02942     * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
02943     * \endenglish
02944     */
02945 std::vector<double> getJointTargetTorques();
02946
02947 /**
02948  * @ingroup RobotState
02949  * \chinese
02950  * 获取机械臂关节目标电流
02951  *
02952  * @return 机械臂关节目标电流
02953  *
02954  * @throws arcs::common_interface::AuboException
02955  *
02956  * @par Python 函数原型
02957  * getJointTargetCurrents(self: pyaubo_sdk.RobotState) -> List[float]
02958  *
02959  * @par Lua 函数原型
02960  * getJointTargetCurrents() -> table
02961  *
02962  * @par Lua 示例
02963  * JointTargetCurrents = getJointTargetCurrents()
02964  *
02965  * @par JSON-RPC 请求示例
02966  * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointTargetCurrents", "params": [], "id": 1}
02967  *
02968  * @par JSON-RPC 响应示例
02969  * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
02970  * \endchinese
02971  * \english
02972  * Get the target joint currents of the manipulator
02973  *
02974  * @return Target joint currents of the manipulator
02975  *
02976  * @throws arcs::common_interface::AuboException
02977  *
02978  * @par Python function prototype
02979  * getJointTargetCurrents(self: pyaubo_sdk.RobotState) -> List[float]
02980  *
02981  * @par Lua function prototype
02982  * getJointTargetCurrents() -> table
02983  *
02984  * @par Lua example
02985  * JointTargetCurrents = getJointTargetCurrents()
02986  *
02987  * @par JSON-RPC request example
02988  * {"jsonrpc": "2.0", "method": "rob1.RobotState.getJointTargetCurrents", "params": [], "id": 1}
02989  *
02990  * @par JSON-RPC response example
02991  * {"id": 1, "jsonrpc": "2.0", "result": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
02992  * \endenglish
02993  */
02994 std::vector<double> getJointTargetCurrents();
02995
02996 /**
02997  * @ingroup RobotState
02998  * \chinese
02999  * 获取示教器是否已启用
03000  * 指示教器使能按钮是否处于按下状态
03001  *
03002  * @return 按下示教器使能按钮返回 false; 反之返回 true
03003  *
03004  * @throws arcs::common_interface::AuboException
03005  *
03006  * @par Python 函数原型
03007  * isTeachPendantEnabled(self: pyaubo_sdk.RobotState) -> bool
03008  *
03009  * @par Lua 函数原型
03010  * isTeachPendantEnabled() -> boolean
03011  *
03012  * @par Lua 示例
03013  * TeachPendantEnabled = isTeachPendantEnabled()
03014  *
03015  * @par JSON-RPC 请求示例
03016  * {"jsonrpc": "2.0", "method": "rob1.RobotState.isTeachPendantEnabled", "params": [], "id": 1}
03017  *
03018  * @par JSON-RPC 响应示例
03019  * {"id": 1, "jsonrpc": "2.0", "result": true}

```

```

03020     * \endchinese
03021     * \english
03022     * Get whether the teach pendant is enabled.
03023     * Indicates whether the enable button of teach pendant is in the pressed
03024     * state.
03025     *
03026     * @return Returns false if the enable button of teach pendant is pressed;
03027     * otherwise returns true
03028     *
03029     * @throws arcs::common_interface::AuboException
03030     *
03031     * @par Python function prototype
03032     * isTeachPendantEnabled(self: pyaubo_sdk.RobotState) -> bool
03033     *
03034     * @par Lua function prototype
03035     * isTeachPendantEnabled() -> boolean
03036     *
03037     * @par Lua example
03038     * TeachPendantEnabled = isTeachPendantEnabled()
03039     *
03040     * @par JSON-RPC request example
03041     * {"jsonrpc": "2.0", "method": "rob1.RobotState.isTeachPendantEnabled", "params": [], "id": 1}
03042     *
03043     * @par JSON-RPC response example
03044     * {"id": 1, "jsonrpc": "2.0", "result": true}
03045     * \endenglish
03046     */
03047 bool isTeachPendantEnabled();
03048
03049 /**
03050     * @ingroup RobotState
03051     * \chinese
03052     * 获取机械臂末端是否已启用
03053     *
03054     * @return 无末端返回 false; 有末端返回 true
03055     *
03056     * @throws arcs::common_interface::AuboException
03057     *
03058     * @par Python 函数原型
03059     * isToolFlangeEnabled(self: pyaubo_sdk.RobotState) -> bool
03060     *
03061     * @par Lua 函数原型
03062     * isToolFlangeEnabled() -> boolean
03063     *
03064     * @par Lua 示例
03065     * toolEnabled = isToolFlangeEnabled()
03066     *
03067     * @par JSON-RPC 请求示例
03068     * {"jsonrpc": "2.0", "method": "rob1.RobotState.isToolFlangeEnabled", "params": [], "id": 1}
03069     *
03070     * @par JSON-RPC 响应示例
03071     * {"id": 1, "jsonrpc": "2.0", "result": true}
03072     * \endchinese
03073     * \english
03074     * Get whether the tool flange is enabled.
03075     *
03076     * @return Returns false if the tool flange is disabled; otherwise returns
03077     * true
03078     *
03079     * @throws arcs::common_interface::AuboException
03080     *
03081     * @par Python function prototype
03082     * isToolFlangeEnabled(self: pyaubo_sdk.RobotState) -> bool
03083     *
03084     * @par Lua function prototype
03085     * isToolFlangeEnabled() -> boolean
03086     *
03087     * @par Lua example
03088     * toolEnabled = isToolFlangeEnabled()
03089     *
03090     * @par JSON-RPC request example
03091     * {"jsonrpc": "2.0", "method": "rob1.RobotState.isToolFlangeEnabled", "params": [], "id": 1}
03092     *
03093     * @par JSON-RPC response example
03094     * {"id": 1, "jsonrpc": "2.0", "result": true}
03095     * \endenglish
03096     */
03097 bool isToolFlangeEnabled();
03098
03099 /**
03100     * @ingroup RobotState
03101     * \chinese
03102     * 获取控制柜温度
03103     *
03104     * @return 控制柜温度
03105     *
03106     * @throws arcs::common_interface::AuboException

```



```

03107      *
03108      * @par Python 函数原型
03109      * getControlBoxTemperature(self: pyaubo_sdk.RobotState) -> float
03110      *
03111      * @par Lua 函数原型
03112      * getControlBoxTemperature() -> number
03113      *
03114      * @par Lua 示例
03115      * ControlBoxTemperature = getControlBoxTemperature()
03116      *
03117      * @par JSON-RPC 请求示例
03118      * {"jsonrpc": "2.0", "method": "rob1.RobotState.getControlBoxTemperature", "params": [], "id": 1}
03119      *
03120      * @par JSON-RPC 响应示例
03121      * {"id": 1, "jsonrpc": "2.0", "result": 25.0}
03122      * \endchinese
03123      * \english
03124      * Get the control box temperature
03125      *
03126      * @return Control box temperature
03127      *
03128      * @throws arcs::common_interface::AuboException
03129      *
03130      * @par Python function prototype
03131      * getControlBoxTemperature(self: pyaubo_sdk.RobotState) -> float
03132      *
03133      * @par Lua function prototype
03134      * getControlBoxTemperature() -> number
03135      *
03136      * @par Lua example
03137      * ControlBoxTemperature = getControlBoxTemperature()
03138      *
03139      * @par JSON-RPC request example
03140      * {"jsonrpc": "2.0", "method": "rob1.RobotState.getControlBoxTemperature", "params": [], "id": 1}
03141      *
03142      * @par JSON-RPC response example
03143      * {"id": 1, "jsonrpc": "2.0", "result": 25.0}
03144      * \endenglish
03145      */
03146 double getControlBoxTemperature();
03147
03148 /**
03149  * @ingroup RobotState
03150  * \chinese
03151  * 获取控制柜湿度
03152  *
03153  * @return 控制柜湿度
03154  *
03155  * @throws arcs::common_interface::AuboException
03156  *
03157  * @par Python 函数原型
03158  * getControlBoxHumidity(self: pyaubo_sdk.RobotState) -> float
03159  *
03160  * @par Lua 函数原型
03161  * getControlBoxHumidity() -> number
03162  *
03163  * @par Lua 示例
03164  * ControlBoxHumidity = getControlBoxHumidity()
03165  *
03166  * @par JSON-RPC 请求示例
03167  * {"jsonrpc": "2.0", "method": "rob1.RobotState.getControlBoxHumidity", "params": [], "id": 1}
03168  *
03169  * @par JSON-RPC 响应示例
03170  * {"id": 1, "jsonrpc": "2.0", "result": 20.0}
03171  * \endchinese
03172  * \english
03173  * Get the control box humidity
03174  *
03175  * @return Control box humidity
03176  *
03177  * @throws arcs::common_interface::AuboException
03178  *
03179  * @par Python function prototype
03180  * getControlBoxHumidity(self: pyaubo_sdk.RobotState) -> float
03181  *
03182  * @par Lua function prototype
03183  * getControlBoxHumidity() -> number
03184  *
03185  * @par Lua example
03186  * ControlBoxHumidity = getControlBoxHumidity()
03187  *
03188  * @par JSON-RPC request example
03189  * {"jsonrpc": "2.0", "method": "rob1.RobotState.getControlBoxHumidity", "params": [], "id": 1}
03190  *
03191  * @par JSON-RPC response example
03192  * {"id": 1, "jsonrpc": "2.0", "result": 20.0}
03193  * \endenglish

```

```

03194     */
03195     double getControlBoxHumidity();
03196
03197     /**
03198     * @ingroup RobotState
03199     * \chinese
03200     * 获取母线电压
03201     *
03202     * @return 母线电压
03203     *
03204     * @throws arcs::common_interface::AuboException
03205     *
03206     * @par Python 函数原型
03207     * getMainVoltage(self: pyaubo_sdk.RobotState) -> float
03208     *
03209     * @par Lua 函数原型
03210     * getMainVoltage() -> number
03211     *
03212     * @par Lua 示例
03213     * MainVoltage = getMainVoltage()
03214     *
03215     * @par JSON-RPC 请求示例
03216     * {"jsonrpc":"2.0","method":"rob1.RobotState.getMainVoltage","params":[],"id":1}
03217     *
03218     * @par JSON-RPC 响应示例
03219     * {"id":1,"jsonrpc":"2.0","result":52.75}
03220     * \endchinese
03221     * \english
03222     * Get the main bus voltage
03223     *
03224     * @return Main bus voltage
03225     *
03226     * @throws arcs::common_interface::AuboException
03227     *
03228     * @par Python function prototype
03229     * getMainVoltage(self: pyaubo_sdk.RobotState) -> float
03230     *
03231     * @par Lua function prototype
03232     * getMainVoltage() -> number
03233     *
03234     * @par Lua example
03235     * MainVoltage = getMainVoltage()
03236     *
03237     * @par JSON-RPC request example
03238     * {"jsonrpc":"2.0","method":"rob1.RobotState.getMainVoltage","params":[],"id":1}
03239     *
03240     * @par JSON-RPC response example
03241     * {"id":1,"jsonrpc":"2.0","result":52.75}
03242     * \endenglish
03243     */
03244     double getMainVoltage();
03245
03246     /**
03247     * @ingroup RobotState
03248     * \chinese
03249     * 获取母线电流
03250     *
03251     * @return 母线电流
03252     *
03253     * @throws arcs::common_interface::AuboException
03254     *
03255     * @par Python 函数原型
03256     * getMainCurrent(self: pyaubo_sdk.RobotState) -> float
03257     *
03258     * @par Lua 函数原型
03259     * getMainCurrent() -> number
03260     *
03261     * @par Lua 示例
03262     * MainCurrent = getMainCurrent()
03263     *
03264     * @par JSON-RPC 请求示例
03265     * {"jsonrpc":"2.0","method":"rob1.RobotState.getMainCurrent","params":[],"id":1}
03266     *
03267     * @par JSON-RPC 响应示例
03268     * {"id":1,"jsonrpc":"2.0","result":0.3204345703125}
03269     * \endchinese
03270     * \english
03271     * Get the main bus current
03272     *
03273     * @return Main bus current
03274     *
03275     * @throws arcs::common_interface::AuboException
03276     *
03277     * @par Python function prototype
03278     * getMainCurrent(self: pyaubo_sdk.RobotState) -> float
03279     *
03280     * @par Lua function prototype

```

```

03281     * getMainCurrent() -> number
03282     *
03283     * @par Lua example
03284     * MainCurrent = getMainCurrent()
03285     *
03286     * @par JSON-RPC request example
03287     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getMainCurrent", "params": [], "id": 1}
03288     *
03289     * @par JSON-RPC response example
03290     * {"id": 1, "jsonrpc": "2.0", "result": 0.3204345703125}
03291     * \endenglish
03292     */
03293 double getMainCurrent();
03294
03295 /**
03296  * @ingroup RobotState
03297  * \chinese
03298  * 获取机器人电压
03299  *
03300  * @return 机器人电压
03301  *
03302  * @throws arcs::common_interface::AuboException
03303  *
03304  * @par Python 函数原型
03305  * getRobotVoltage(self: pyaubo_sdk.RobotState) -> float
03306  *
03307  * @par Lua 函数原型
03308  * getRobotVoltage() -> number
03309  *
03310  * @par Lua 示例
03311  * RobotVoltage = getRobotVoltage()
03312  *
03313  * @par JSON-RPC 请求示例
03314  * {"jsonrpc": "2.0", "method": "rob1.RobotState.getRobotVoltage", "params": [], "id": 1}
03315  *
03316  * @par JSON-RPC 响应示例
03317  * {"id": 1, "jsonrpc": "2.0", "result": 52.75}
03318  * \endchinese
03319  * \english
03320  * Get the robot voltage
03321  *
03322  * @return Robot voltage
03323  *
03324  * @throws arcs::common_interface::AuboException
03325  *
03326  * @par Python function prototype
03327  * getRobotVoltage(self: pyaubo_sdk.RobotState) -> float
03328  *
03329  * @par Lua function prototype
03330  * getRobotVoltage() -> number
03331  *
03332  * @par Lua example
03333  * RobotVoltage = getRobotVoltage()
03334  *
03335  * @par JSON-RPC request example
03336  * {"jsonrpc": "2.0", "method": "rob1.RobotState.getRobotVoltage", "params": [], "id": 1}
03337  *
03338  * @par JSON-RPC response example
03339  * {"id": 1, "jsonrpc": "2.0", "result": 52.75}
03340  * \endenglish
03341  */
03342 double getRobotVoltage();
03343
03344 /**
03345  * @ingroup RobotState
03346  * \chinese
03347  * 获取机器人电流
03348  *
03349  * @return 机器人电流
03350  *
03351  * @throws arcs::common_interface::AuboException
03352  *
03353  * @par Python 函数原型
03354  * getRobotCurrent(self: pyaubo_sdk.RobotState) -> float
03355  *
03356  * @par Lua 函数原型
03357  * getRobotCurrent() -> number
03358  *
03359  * @par Lua 示例
03360  * RobotCurrent = getRobotCurrent()
03361  *
03362  * @par JSON-RPC 请求示例
03363  * {"jsonrpc": "2.0", "method": "rob1.RobotState.getRobotCurrent", "params": [], "id": 1}
03364  *
03365  * @par JSON-RPC 响应示例
03366  * {"id": 1, "jsonrpc": "2.0", "result": 0.3204345703125}
03367  * \endchinese

```

```

03368     * \english
03369     * Get the robot current
03370     *
03371     * @return Robot current
03372     *
03373     * @throws arcs::common_interface::AuboException
03374     *
03375     * @par Python function prototype
03376     * getRobotCurrent(self: pyaubo_sdk.RobotState) -> float
03377     *
03378     * @par Lua function prototype
03379     * getRobotCurrent() -> number
03380     *
03381     * @par Lua example
03382     * RobotCurrent = getRobotCurrent()
03383     *
03384     * @par JSON-RPC request example
03385     * {"jsonrpc": "2.0", "method": "rob1.RobotState.getRobotCurrent", "params": [], "id": 1}
03386     *
03387     * @par JSON-RPC response example
03388     * {"id": 1, "jsonrpc": "2.0", "result": 0.3204345703125}
03389     * \endenglish
03390     */
03391 double getRobotCurrent();
03392
03393 /**
03394  * @ingroup RobotState
03395  * \chinese
03396  * 获取底座制动电阻温度
03397  *
03398  * @return 底座制动电阻温度
03399  *
03400  * @throws arcs::common_interface::AuboException
03401  *
03402  * @par Python 函数原型
03403  * getPedestalBrakeResTemp(self: pyaubo_sdk.RobotState) -> float
03404  *
03405  * @par Lua 函数原型
03406  * getPedestalBrakeResTemp() -> number
03407  *
03408  * @par Lua 示例
03409  * PedestalBrakeResTemp = getPedestalBrakeResTemp()
03410  *
03411  * @par JSON-RPC 请求示例
03412  * {"jsonrpc": "2.0", "method": "rob1.RobotState.getPedestalBrakeResTemp", "params": [], "id": 1}
03413  *
03414  * @par JSON-RPC 响应示例
03415  * {"id": 1, "jsonrpc": "2.0", "result": 23.1111}
03416  * \endchinese
03417  * \english
03418  * Get the Pedestal Brake Res Temp
03419  *
03420  * @return Pedestal Brake Res Temp
03421  *
03422  * @throws arcs::common_interface::AuboException
03423  *
03424  * @par Python function prototype
03425  * getPedestalBrakeResTemp(self: pyaubo_sdk.RobotState) -> float
03426  *
03427  * @par Lua function prototype
03428  * getPedestalBrakeResTemp() -> number
03429  *
03430  * @par Lua example
03431  * PedestalBrakeResTemp = getPedestalBrakeResTemp()
03432  *
03433  * @par JSON-RPC request example
03434  * {"jsonrpc": "2.0", "method": "rob1.RobotState.getPedestalBrakeResTemp", "params": [], "id": 1}
03435  *
03436  * @par JSON-RPC response example
03437  * {"id": 1, "jsonrpc": "2.0", "result": 24.235}
03438  * \endenglish
03439  */
03440 double getPedestalBrakeResTemp();
03441
03442 /**
03443  * @ingroup RobotState
03444  * \chinese
03445  * 获取机器人缓速等级
03446  *
03447  * @return 机器人缓速等级
03448  *
03449  * @throws arcs::common_interface::AuboException
03450  *
03451  * @par JSON-RPC 请求示例
03452  * {"jsonrpc": "2.0", "method": "rob1.RobotState.getSlowDownLevel", "params": [], "id": 1}
03453  *
03454  * @par JSON-RPC 响应示例

```

```

03455     * {"id":1,"jsonrpc":"2.0","result":0}
03456     * \endchinese
03457     * \english
03458     * Get the robot slow down level
03459     *
03460     * @return Robot slow down level
03461     *
03462     * @throws arcs::common_interface::AuboException
03463     *
03464     * @par JSON-RPC request example
03465     * {"jsonrpc":"2.0","method":"rob1.RobotState.getSlowDownLevel","params":[],"id":1}
03466     *
03467     * @par JSON-RPC response example
03468     * {"id":1,"jsonrpc":"2.0","result":0}
03469     * \endenglish
03470     */
03471 int getSlowDownLevel();
03472
03473 /**
03474  * @ingroup RobotState
03475  * \chinese
03476  * 获取末端力传感器通信状态
03477  *
03478  * @param name 力传感器名称, 与 selectTcpForceSensor 的参数一致
03479  * @return 通信正常返回 true; 反之返回 false
03480  *
03481  * @throws arcs::common_interface::AuboException
03482  *
03483  * @par JSON-RPC 请求示例
03484  * {"jsonrpc":"2.0","method":"rob1.RobotState.getTcpForceSensorStatus","params":["tool.KWR75A"],"id":1}
03485  *
03486  * @par JSON-RPC 响应示例
03487  * {"id":1,"jsonrpc":"2.0","result":0}
03488  * \endchinese
03489  * \english
03490  * Get the communication status of the tool force sensor
03491  *
03492  * @param name force sensor name, it is consistent with the parameters of
03493  * 'selectTcpForceSensor'
03494  * @return Returns true if communication is normal; otherwise returns false
03495  *
03496  * @throws arcs::common_interface::AuboException
03497  *
03498  * @par JSON-RPC request example
03499  * {"jsonrpc":"2.0","method":"rob1.RobotState.getSlowDownLevel","params":["tool.KWR75A"],"id":1}
03500  *
03501  * @par JSON-RPC response example
03502  * {"id":1,"jsonrpc":"2.0","result":0}
03503  * \endenglish
03504  */
03505 bool getTcpForceSensorStatus(const std::string &name);
03506
03507 protected:
03508     void *d_;
03509 };
03510 using RobotStatePtr = std::shared_ptr<RobotState>;
03511 } // namespace common_interface
03512 } // namespace arcs
03513 #endif // AUBO_SDK_ROBOT_STATE_INTERFACE_H

```

12.34 include/aubo/robot_interface.h 文件参考

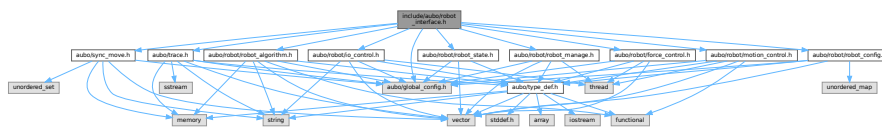
机器人API 接口

```

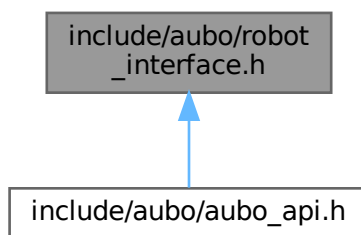
#include <aubo/sync_move.h>
#include <aubo/trace.h>
#include <aubo/robot/motion_control.h>
#include <aubo/robot/force_control.h>
#include <aubo/robot/io_control.h>
#include <aubo/robot/robot_algorithm.h>
#include <aubo/robot/robot_state.h>
#include <aubo/robot/robot_manage.h>
#include <aubo/robot/robot_config.h>
#include <aubo/global_config.h>

```

robot_interface.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::common_interface::RobotInterface](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::common_interface](#)

类型定义

- using [arcs::common_interface::RobotInterfacePtr](#) = [std::shared_ptr<RobotInterface>](#)

12.34.1 详细描述

机器人API 接口

在文件 [robot_interface.h](#) 中定义.

12.35 robot_interface.h

[浏览该文件的文档.](#)

```
00001 /** @file robot_interface.h
00002  * @brief 机器人 API 接口
00003  */
00004 #ifndef AUBO_SDK_ROBOT_INTERFACE_H
00005 #define AUBO_SDK_ROBOT_INTERFACE_H
00006
00007 #include <aubo/sync_move.h>
00008 #include <aubo/trace.h>
00009 #include <aubo/robot/motion_control.h>
00010 #include <aubo/robot/force_control.h>
00011 #include <aubo/robot/io_control.h>
00012 #include <aubo/robot/robot_algorithm.h>
00013 #include <aubo/robot/robot_state.h>
00014 #include <aubo/robot/robot_manage.h>
00015 #include <aubo/robot/robot_config.h>
00016 #include <aubo/global_config.h>
```

```

00017
00018 namespace arcs {
00019 namespace common_interface {
00020
00021 /**
00022  * @defgroup RobotInterface RobotInterface(机器人模块)
00023  * \-chinese 机器人模块 \-english RobotInterface
00024  */
00025 class ARCS_ABI_EXPORT RobotInterface
00026 {
00027 public:
00028     RobotInterface();
00029     virtual ~RobotInterface();
00030
00031     /**
00032      * @ingroup RobotInterface
00033      * @ref RobotConfig
00034      * \chinese
00035      * 获取 RobotConfig 接口
00036      *
00037      * @return RobotConfigPtr 对象的指针
00038      *
00039      * @par Python 函数原型
00040      * getRobotConfig(self: pyaubo_sdk.RobotInterface) ->
00041      * arcs::common_interface::RobotConfig
00042      *
00043      * @par C++ 示例
00044      * @code
00045      * auto rpc_cli = std::make_shared<RpcClient>();
00046      * auto robot_name = rpc_cli->getRobotNames().front();
00047      * RobotConfigPtr ptr =
00048      * rpc_cli->getRobotInterface(robot_name)->getRobotConfig();
00049      * @endcode
00050      * \endchinese
00051      * \english
00052      * Get RobotConfig interface
00053      *
00054      * @return Pointer to RobotConfig object
00055      *
00056      * @par Python function prototype
00057      * getRobotConfig(self: pyaubo_sdk.RobotInterface) ->
00058      * arcs::common_interface::RobotConfig
00059      *
00060      * @par C++ example
00061      * @code
00062      * auto rpc_cli = std::make_shared<RpcClient>();
00063      * auto robot_name = rpc_cli->getRobotNames().front();
00064      * RobotConfigPtr ptr =
00065      * rpc_cli->getRobotInterface(robot_name)->getRobotConfig();
00066      * @endcode
00067      * \endenglish
00068      */
00069
00070     RobotConfigPtr getRobotConfig();
00071
00072     /**
00073      * @ingroup RobotInterface
00074      * @ref MotionControl
00075      * \chinese
00076      * 获取运动规划接口
00077      *
00078      * @return MotionControlPtr 对象的指针
00079      *
00080      * @par Python 函数原型
00081      * getMotionControl(self: pyaubo_sdk.RobotInterface) ->
00082      * arcs::common_interface::MotionControl
00083      *
00084      * @par C++ 示例
00085      * @code
00086      * auto rpc_cli = std::make_shared<RpcClient>();
00087      * auto robot_name = rpc_cli->getRobotNames().front();
00088      * MotionControlPtr ptr =
00089      * rpc_cli->getRobotInterface(robot_name)->getMotionControl();
00090      * @endcode
00091      * \endchinese
00092      * \english
00093      * Get motion planning interface
00094      *
00095      * @return Pointer to MotionControl object
00096      *
00097      * @par Python function prototype
00098      * getMotionControl(self: pyaubo_sdk.RobotInterface) ->
00099      * arcs::common_interface::MotionControl
00100      *
00101      * @par C++ example
00102      * @code
00103      * auto rpc_cli = std::make_shared<RpcClient>();

```

```

00104     * auto robot_name = rpc_cli->getRobotNames().front();
00105     * MotionControlPtr ptr =
00106     * rpc_cli->getRobotInterface(robot_name)->getMotionControl();
00107     * @endcode
00108     * \endenglish
00109     */
00110 MotionControlPtr getMotionControl();
00111
00112 /**
00113  * @ingroup RobotInterface
00114  * @ref ForceControl
00115  * \chinese
00116  * 获取力控接口
00117  *
00118  * @return ForceControlPtr 对象的指针
00119  *
00120  * @par Python 函数原型
00121  * getForceControl(self: pyaubo_sdk.RobotInterface) ->
00122  * arcs::common_interface::ForceControl
00123  *
00124  * @par C++ 示例
00125  * @code
00126  * auto rpc_cli = std::make_shared<RpcClient>();
00127  * auto robot_name = rpc_cli->getRobotNames().front();
00128  * ForceControlPtr ptr =
00129  * rpc_cli->getRobotInterface(robot_name)->getForceControl();
00130  * @endcode
00131  * \endchinese
00132  * \english
00133  * Get force control interface
00134  *
00135  * @return Pointer to ForceControl object
00136  *
00137  * @par Python function prototype
00138  * getForceControl(self: pyaubo_sdk.RobotInterface) ->
00139  * arcs::common_interface::ForceControl
00140  *
00141  * @par C++ example
00142  * @code
00143  * auto rpc_cli = std::make_shared<RpcClient>();
00144  * auto robot_name = rpc_cli->getRobotNames().front();
00145  * ForceControlPtr ptr =
00146  * rpc_cli->getRobotInterface(robot_name)->getForceControl();
00147  * @endcode
00148  * \endenglish
00149  */
00150 ForceControlPtr getForceControl();
00151
00152 /**
00153  * @ingroup RobotInterface
00154  * @ref IoControl
00155  * \chinese
00156  * 获取 IO 控制的接口
00157  *
00158  * @return IoControlPtr 对象的指针
00159  *
00160  * @par Python 函数原型
00161  * getIoControl(self: pyaubo_sdk.RobotInterface) ->
00162  * arcs::common_interface::IoControl
00163  *
00164  * @par C++ 示例
00165  * @code
00166  * auto rpc_cli = std::make_shared<RpcClient>();
00167  * auto robot_name = rpc_cli->getRobotNames().front();
00168  * IoControlPtr ptr =
00169  * rpc_cli->getRobotInterface(robot_name)->getIoControl();
00170  * @endcode
00171  * \endchinese
00172  * \english
00173  * Get IO control interface
00174  *
00175  * @return Pointer to IoControl object
00176  *
00177  * @par Python function prototype
00178  * getIoControl(self: pyaubo_sdk.RobotInterface) ->
00179  * arcs::common_interface::IoControl
00180  *
00181  * @par C++ example
00182  * @code
00183  * auto rpc_cli = std::make_shared<RpcClient>();
00184  * auto robot_name = rpc_cli->getRobotNames().front();
00185  * IoControlPtr ptr =
00186  * rpc_cli->getRobotInterface(robot_name)->getIoControl();
00187  * @endcode
00188  * \endenglish
00189  */
00190 IoControlPtr getIoControl();

```



```

00191
00192 /**
00193  * @ingroup RobotInterface
00194  * @ref SyncMove
00195  * \chinese
00196  * 获取同步运动接口
00197  *
00198  * @return SyncMovePtr 对象的指针
00199  *
00200  * @par Python 函数原型
00201  * getSyncMove(self: pyaubo_sdk.RobotInterface) ->
00202  * arcs::common_interface::SyncMove
00203  *
00204  * @par C++ 示例
00205  * @code
00206  * auto rpc_cli = std::make_shared<RpcClient>();
00207  * auto robot_name = rpc_cli->getRobotNames().front();
00208  * SyncMovePtr ptr = rpc_cli->getRobotInterface(robot_name)->getSyncMove();
00209  * @endcode
00210  * \endchinese
00211  * \english
00212  * Get synchronized motion interface
00213  *
00214  * @return Pointer to SyncMove object
00215  *
00216  * @par Python function prototype
00217  * getSyncMove(self: pyaubo_sdk.RobotInterface) ->
00218  * arcs::common_interface::SyncMove
00219  *
00220  * @par C++ example
00221  * @code
00222  * auto rpc_cli = std::make_shared<RpcClient>();
00223  * auto robot_name = rpc_cli->getRobotNames().front();
00224  * SyncMovePtr ptr = rpc_cli->getRobotInterface(robot_name)->getSyncMove();
00225  * @endcode
00226  * \endenglish
00227  */
00228 SyncMovePtr getSyncMove();
00229
00230 /**
00231  * @ingroup RobotInterface
00232  * @ref RobotAlgorithm
00233  * \chinese
00234  * 获取机器人实用算法接口
00235  *
00236  * @return RobotAlgorithmPtr 对象的指针
00237  *
00238  * @par Python 函数原型
00239  * getRobotAlgorithm(self: pyaubo_sdk.RobotInterface) ->
00240  * arcs::common_interface::RobotAlgorithm
00241  *
00242  * @par C++ 示例
00243  * @code
00244  * auto rpc_cli = std::make_shared<RpcClient>();
00245  * auto robot_name = rpc_cli->getRobotNames().front();
00246  * RobotAlgorithmPtr ptr =
00247  * rpc_cli->getRobotInterface(robot_name)->getRobotAlgorithm();
00248  * @endcode
00249  * \endchinese
00250  * \english
00251  * Get robot utility algorithm interface
00252  *
00253  * @return Pointer to RobotAlgorithm object
00254  *
00255  * @par Python function prototype
00256  * getRobotAlgorithm(self: pyaubo_sdk.RobotInterface) ->
00257  * arcs::common_interface::RobotAlgorithm
00258  *
00259  * @par C++ example
00260  * @code
00261  * auto rpc_cli = std::make_shared<RpcClient>();
00262  * auto robot_name = rpc_cli->getRobotNames().front();
00263  * RobotAlgorithmPtr ptr =
00264  * rpc_cli->getRobotInterface(robot_name)->getRobotAlgorithm();
00265  * @endcode
00266  * \endenglish
00267  */
00268 RobotAlgorithmPtr getRobotAlgorithm();
00269
00270 /**
00271  * @ingroup RobotInterface
00272  * @ref RobotManage
00273  * \chinese
00274  * 获取机器人管理接口（上电、启动、停止等）
00275  *
00276  * @return RobotManagePtr 对象的指针
00277  *

```

```

00278      * @par Python 函数原型
00279      * getRobotManage(self: pyaubo_sdk.RobotInterface) ->
00280      * arcs::common_interface::RobotManage
00281      *
00282      * @par C++ 示例
00283      * @code
00284      * auto rpc_cli = std::make_shared<RpcClient>();
00285      * auto robot_name = rpc_cli->getRobotNames().front();
00286      * RobotManagePtr ptr =
00287      * rpc_cli->getRobotInterface(robot_name)->getRobotManage();
00288      * @endcode
00289      * \endchinese
00290      * \english
00291      * Get robot management interface (power on, start, stop, etc.)
00292      *
00293      * @return Pointer to RobotManage object
00294      *
00295      * @par Python function prototype
00296      * getRobotManage(self: pyaubo_sdk.RobotInterface) ->
00297      * arcs::common_interface::RobotManage
00298      *
00299      * @par C++ example
00300      * @code
00301      * auto rpc_cli = std::make_shared<RpcClient>();
00302      * auto robot_name = rpc_cli->getRobotNames().front();
00303      * RobotManagePtr ptr =
00304      * rpc_cli->getRobotInterface(robot_name)->getRobotManage();
00305      * @endcode
00306      * \endenglish
00307      */
00308 RobotManagePtr getRobotManage();
00309
00310 /**
00311      * @ingroup RobotInterface
00312      * @ref RobotState
00313      * \chinese
00314      * 获取机器人状态接口
00315      *
00316      * @return RobotStatePtr 对象的指针
00317      *
00318      * @par Python 函数原型
00319      * getRobotState(self: pyaubo_sdk.RobotInterface) ->
00320      * arcs::common_interface::RobotState
00321      *
00322      * @par C++ 示例
00323      * @code
00324      * auto rpc_cli = std::make_shared<RpcClient>();
00325      * auto robot_name = rpc_cli->getRobotNames().front();
00326      * RobotStatePtr ptr =
00327      * rpc_cli->getRobotInterface(robot_name)->getRobotState();
00328      * @endcode
00329      * \endchinese
00330      * \english
00331      * Get robot state interface
00332      *
00333      * @return Pointer to RobotState object
00334      *
00335      * @par Python function prototype
00336      * getRobotState(self: pyaubo_sdk.RobotInterface) ->
00337      * arcs::common_interface::RobotState
00338      *
00339      * @par C++ example
00340      * @code
00341      * auto rpc_cli = std::make_shared<RpcClient>();
00342      * auto robot_name = rpc_cli->getRobotNames().front();
00343      * RobotStatePtr ptr =
00344      * rpc_cli->getRobotInterface(robot_name)->getRobotState();
00345      * @endcode
00346      * \endenglish
00347      */
00348 RobotStatePtr getRobotState();
00349
00350 /**
00351      * @ingroup RobotInterface
00352      * @ref Trace
00353      * \chinese
00354      * 获取告警信息接口
00355      *
00356      * @return TracePtr 对象的指针
00357      *
00358      * @par Python 函数原型
00359      * getTrace(self: pyaubo_sdk.RobotInterface) ->
00360      * arcs::common_interface::Trace
00361      *
00362      * @par C++ 示例
00363      * @code
00364      * auto rpc_cli = std::make_shared<RpcClient>();

```

```

00365     * auto robot_name = rpc_cli->getRobotNames().front();
00366     * TracePtr ptr = rpc_cli->getRobotInterface(robot_name)->getTrace();
00367     * @endcode
00368     * \endchinese
00369     * \english
00370     * Get alarm information interface
00371     *
00372     * @return Pointer to Trace object
00373     *
00374     * @par Python function prototype
00375     * getTrace(self: pyaubo_sdk.RobotInterface) ->
00376     * arcs::common_interface::Trace
00377     *
00378     * @par C++ example
00379     * @code
00380     * auto rpc_cli = std::make_shared<RpcClient>();
00381     * auto robot_name = rpc_cli->getRobotNames().front();
00382     * TracePtr ptr = rpc_cli->getRobotInterface(robot_name)->getTrace();
00383     * @endcode
00384     * \endenglish
00385     */
00386     TracePtr getTrace();
00387
00388 protected:
00389     void *d_;
00390 };
00391 using RobotInterfacePtr = std::shared_ptr<RobotInterface>;
00392
00393 } // namespace common_interface
00394 } // namespace arcs
00395
00396 #endif // AUBO_SDK_ROBOT_INTERFACE_H

```

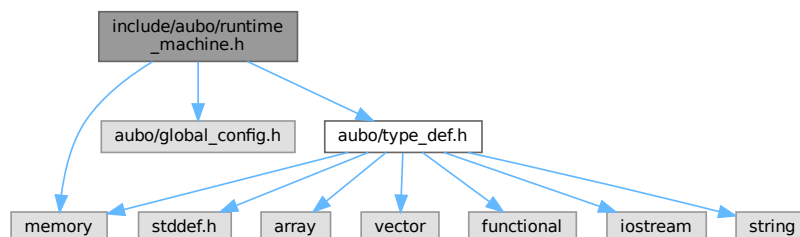
12.36 include/aubo/runtime_machine.h 文件参考

Script interpreter runtime interface, allows pausing the script interpreter and setting/removing break-points.

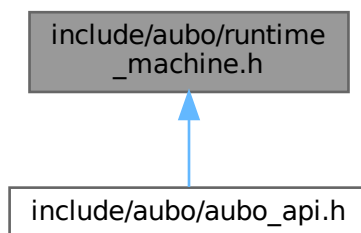
```

#include <memory>
#include <aubo/global_config.h>
#include <aubo/type_def.h>
runtime_machine.h 的引用 (Include) 关系图:

```



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::common_interface::RuntimeMachine](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::common_interface](#)

类型定义

- using [arcs::common_interface::RuntimeMachinePtr](#) = std::shared_ptr<[RuntimeMachine](#)>

12.36.1 详细描述

Script interpreter runtime interface, allows pausing the script interpreter and setting/removing breakpoints.

在文件 [runtime_machine.h](#) 中定义.

12.37 runtime__machine.h

[浏览该文件的文档.](#)

```

00001 /** @file runtime_machine.h
00002  * @brief Script interpreter runtime interface,
00003  * allows pausing the script interpreter and setting/removing breakpoints.
00004  */
00005 #ifndef AUBO_SDK_RUNTIME_MACHINE_INTERFACE_H
00006 #define AUBO_SDK_RUNTIME_MACHINE_INTERFACE_H
00007
00008 #include <memory>
00009 #include <aubo/global_config.h>
00010 #include <aubo/type_def.h>
00011
00012 namespace arcs {
00013 namespace common_interface {
00014
00015 /**
00016  * @defgroup RuntimeMachine RuntimeMachine (运行时管理)
00017  * The RuntimeMachine class
00018  */
00019 class ARCS_ABI_EXPORT RuntimeMachine
00020 {
00021 public:
00022     RuntimeMachine();
00023     virtual ~RuntimeMachine();
00024
00025     /**
00026      * @ingroup RuntimeMachine
00027      * \chinese
00028      * 返回 task_id
  
```

```

00029      *
00030      * @par JSON-RPC 请求示例
00031      * {"jsonrpc": "2.0", "method": "RuntimeMachine.newTask", "params": [false], "id": 1}
00032      *
00033      * @par JSON-RPC 响应示例
00034      * {"id": 1, "jsonrpc": "2.0", "result": 26}
00035      * \endchinese
00036      * \english
00037      * Returns the task_id
00038      *
00039      * @par JSON-RPC Request Example
00040      * {"jsonrpc": "2.0", "method": "RuntimeMachine.newTask", "params": [false], "id": 1}
00041      *
00042      * @par JSON-RPC Response Example
00043      * {"id": 1, "jsonrpc": "2.0", "result": 26}
00044      * \endenglish
00045      */
00046      int newTask(bool daemon = false);
00047
00048      /**
00049      * @ingroup RuntimeMachine
00050      * \chinese
00051      * 删除 task，会终止正在执行的运动
00052      *
00053      * @par JSON-RPC 请求示例
00054      * {"jsonrpc": "2.0", "method": "RuntimeMachine.deleteTask", "params": [26], "id": 1}
00055      *
00056      * @par JSON-RPC 响应示例
00057      * {"id": 1, "jsonrpc": "2.0", "result": 0}
00058      *
00059      * \endchinese
00060      * \english
00061      * Delete a task, which will terminate any ongoing motion.
00062      *
00063      * @par JSON-RPC Request Example
00064      * {"jsonrpc": "2.0", "method": "RuntimeMachine.deleteTask", "params": [26], "id": 1}
00065      *
00066      * @par JSON-RPC Response Example
00067      * {"id": 1, "jsonrpc": "2.0", "result": 0}
00068      *
00069      * \endenglish
00070      */
00071      int deleteTask(int tid);
00072
00073      /**
00074      * @ingroup RuntimeMachine
00075      * \chinese
00076      * 等待 task 自然结束
00077      *
00078      * @param tid
00079      * @return
00080      *
00081      * @par JSON-RPC 请求示例
00082      * {"jsonrpc": "2.0", "method": "RuntimeMachine.detachTask", "params": [26], "id": 1}
00083      *
00084      * @par JSON-RPC 响应示例
00085      * {"id": 1, "jsonrpc": "2.0", "result": 0}
00086      *
00087      * \endchinese
00088      * \english
00089      * Wait for the task to finish naturally
00090      *
00091      * @param tid
00092      * @return
00093      *
00094      * @par JSON-RPC Request Example
00095      * {"jsonrpc": "2.0", "method": "RuntimeMachine.detachTask", "params": [26], "id": 1}
00096      *
00097      * @par JSON-RPC Response Example
00098      * {"id": 1, "jsonrpc": "2.0", "result": 0}
00099      *
00100      * \endenglish
00101      */
00102      int detachTask(int tid);
00103
00104      /**
00105      * @ingroup RuntimeMachine
00106      * \chinese
00107      * 判断任务是否存活
00108      *
00109      * @param tid
00110      * @return
00111      *
00112      * @par JSON-RPC 请求示例
00113      * {"jsonrpc": "2.0", "method": "RuntimeMachine.isTaskAlive", "params": [26], "id": 1}
00114      *
00115      * @par JSON-RPC 响应示例

```

```

00116     * {"id":1,"jsonrpc":"2.0","result":true}
00117     *
00118     * \endchinese
00119     * \english
00120     * Check if the task is alive
00121     *
00122     * @param tid
00123     * @return
00124     *
00125     * @par JSON-RPC Request Example
00126     * {"jsonrpc":"2.0","method":"RuntimeMachine.isTaskAlive","params":[26],"id":1}
00127     *
00128     * @par JSON-RPC Response Example
00129     * {"id":1,"jsonrpc":"2.0","result":true}
00130     *
00131     * \endenglish
00132     */
00133 bool isTaskAlive(int tid);
00134
00135 /**
00136  * @ingroup RuntimeMachine
00137  * \chinese
00138  * 获取任务中缓存的指令的数量
00139  *
00140  * @param tid
00141  * @return
00142  * \endchinese
00143  * \english
00144  * Get the number of cached instructions in the task
00145  *
00146  * @param tid
00147  * @return
00148  * \endenglish
00149  */
00150 int getTaskQueueSize(int tid);
00151
00152 /**
00153  * @ingroup RuntimeMachine
00154  * \chinese
00155  * 切换当前线程，切换之后接下来的指令将被插入切换后的线程中
00156  *
00157  * @param tid
00158  * @return
00159  *
00160  * @par JSON-RPC 请求示例
00161  * {"jsonrpc":"2.0","method":"RuntimeMachine.switchTask","params":[26],"id":1}
00162  *
00163  * @par JSON-RPC 响应示例
00164  * {"id":1,"jsonrpc":"2.0","result":0}
00165  *
00166  * \endchinese
00167  * \english
00168  * Switch the current thread. After switching, subsequent instructions will
00169  * be inserted into the switched thread.
00170  *
00171  * @param tid
00172  * @return
00173  *
00174  * @par JSON-RPC Request Example
00175  * {"jsonrpc":"2.0","method":"RuntimeMachine.switchTask","params":[26],"id":1}
00176  *
00177  * @par JSON-RPC Response Example
00178  * {"id":1,"jsonrpc":"2.0","result":0}
00179  *
00180  * \endenglish
00181  */
00182 int switchTask(int tid);
00183
00184 /**
00185  * @ingroup RuntimeMachine
00186  * \chinese
00187  * 标记记下来的指令的行号和注释
00188  *
00189  * @param lineno
00190  * @param comment
00191  * @return
00192  *
00193  * @par JSON-RPC 请求示例
00194  * {"jsonrpc":"2.0","method":"RuntimeMachine.setLabel","params":[5,"moveJoint"],"id":1}
00195  *
00196  * @par JSON-RPC 响应示例
00197  * {"id":1,"jsonrpc":"2.0","result":0}
00198  *
00199  * \endchinese
00200  * \english
00201  * Mark the line number and comment of the recorded instruction
00202  *

```

```

00203     * @param lineno
00204     * @param comment
00205     * @return
00206     *
00207     * @par JSON-RPC Request Example
00208     * {"jsonrpc": "2.0", "method": "RuntimeMachine.setLabel", "params": [5, "moveJoint"], "id": 1}
00209     *
00210     * @par JSON-RPC Response Example
00211     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00212     *
00213     * \endenglish
00214     */
00215 int setLabel(int lineno, const std::string &comment);
00216
00217 /**
00218     * @ingroup RuntimeMachine
00219     * \chinese
00220     * 向 aubo_control 日志中添加注释
00221     * 使用 setLabel 替换
00222     *
00223     * @param tid 指令的线程 ID
00224     * @param lineno 行号
00225     * @param comment 注释
00226     * @return
00227     *
00228     * @par Python 函数原型
00229     * setPlanContext(self: pyaubo_sdk.RuntimeMachine, arg0: int, arg1: int,
00230     * arg2: str) -> int
00231     *
00232     * @par Lua 函数原型
00233     * setPlanContext(tid: number, lineno: number, comment: string) -> number
00234     *
00235     * @par JSON-RPC 请求示例
00236     * {"jsonrpc": "2.0", "method": "RuntimeMachine.setPlanContext", "params": [26, 3, "moveJoint"], "id": 1}
00237     *
00238     * @par JSON-RPC 响应示例
00239     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00240     *
00241     * \endchinese
00242     * \english
00243     * Add a comment to the aubo_control log
00244     * Use setLabel instead
00245     *
00246     * @param tid Thread ID of the instruction
00247     * @param lineno Line number
00248     * @param comment Comment
00249     * @return
00250     *
00251     * @par Python function prototype
00252     * setPlanContext(self: pyaubo_sdk.RuntimeMachine, arg0: int, arg1: int,
00253     * arg2: str) -> int
00254     *
00255     * @par Lua function prototype
00256     * setPlanContext(tid: number, lineno: number, comment: string) -> number
00257     *
00258     * @par JSON-RPC Request Example
00259     * {"jsonrpc": "2.0", "method": "RuntimeMachine.setPlanContext", "params": [26, 3, "moveJoint"], "id": 1}
00260     *
00261     * @par JSON-RPC Response Example
00262     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00263     *
00264     * \endenglish
00265     */
00266 ARCS_DEPRECATED int setPlanContext(int tid, int lineno,
00267                                     const std::string &comment);
00268
00269 /**
00270     * @ingroup RuntimeMachine
00271     * \chinese
00272     * 空操作
00273     *
00274     * @return
00275     *
00276     * @par JSON-RPC 请求示例
00277     * {"jsonrpc": "2.0", "method": "RuntimeMachine.nop", "params": [], "id": 1}
00278     *
00279     * @par JSON-RPC 响应示例
00280     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00281     *
00282     * \endchinese
00283     * \english
00284     * No operation
00285     *
00286     * @return
00287     *
00288     * @par JSON-RPC Request Example
00289     * {"jsonrpc": "2.0", "method": "RuntimeMachine.nop", "params": [], "id": 1}

```

```

00290     *
00291     * @par JSON-RPC Response Example
00292     * {"id":1,"jsonrpc":"2.0","result":0}
00293     *
00294     * \endenglish
00295     */
00296     int nop();
00297
00298     /**
00299     * @ingroup RuntimeMachine
00300     * \chinese
00301     * 获取耗时的接口 (INST) 执行状态, 如 setPersistentParameters
00302     *
00303     * @return 指令名字, 执行状态
00304     * 执行状态: EXECUTING/FINISHED
00305     *
00306     * @par Python 函数原型
00307     * getExecutionStatus(self: pyaubo_sdk.RuntimeMachine) -> Tuple[str, str,
00308     * int]
00309     *
00310     * @par Lua 函数原型
00311     * getExecutionStatus() -> string, string, number
00312     *
00313     * @par JSON-RPC 请求示例
00314     * {"jsonrpc":"2.0","method":"RuntimeMachine.getExecutionStatus","params":[],"id":1}
00315     *
00316     * @par JSON-RPC 响应示例
00317     * {"id":1,"jsonrpc":"2.0","result":["confirmSafetyParameters","FINISHED"]}
00318     *
00319     * \endchinese
00320     * \english
00321     * Get the execution status of time-consuming interfaces (INST), such as
00322     * setPersistentParameters
00323     *
00324     * @return Instruction name, execution status
00325     * Execution status: EXECUTING/FINISHED
00326     *
00327     * @par Python function prototype
00328     * getExecutionStatus(self: pyaubo_sdk.RuntimeMachine) -> Tuple[str, str,
00329     * int]
00330     *
00331     * @par Lua function prototype
00332     * getExecutionStatus() -> string, string, number
00333     *
00334     * @par JSON-RPC Request Example
00335     * {"jsonrpc":"2.0","method":"RuntimeMachine.getExecutionStatus","params":[],"id":1}
00336     *
00337     * @par JSON-RPC Response Example
00338     * {"id":1,"jsonrpc":"2.0","result":["confirmSafetyParameters","FINISHED"]}
00339     *
00340     * \endenglish
00341     */
00342     std::tuple<std::string, std::string> getExecutionStatus();
00343     std::tuple<std::string, std::string, int> getExecutionStatus1();
00344
00345     /**
00346     * @ingroup RuntimeMachine
00347     * \chinese
00348     * 跳转到指定行号
00349     *
00350     * @param lineno
00351     * @return
00352     *
00353     * @par Python 函数原型
00354     * gotoLine(self: pyaubo_sdk.RuntimeMachine, arg0: int) -> int
00355     *
00356     * @par Lua 函数原型
00357     * gotoLine(lineno: number) -> number
00358     *
00359     * @par JSON-RPC 请求示例
00360     * {"jsonrpc":"2.0","method":"RuntimeMachine.gotoLine","params":[10],"id":1}
00361     *
00362     * @par JSON-RPC 响应示例
00363     * {"id":1,"jsonrpc":"2.0","result":0}
00364     *
00365     * \endchinese
00366     * \english
00367     * Jump to the specified line number
00368     *
00369     * @param lineno
00370     * @return
00371     *
00372     * @par Python function prototype
00373     * gotoLine(self: pyaubo_sdk.RuntimeMachine, arg0: int) -> int
00374     *
00375     * @par Lua function prototype
00376     * gotoLine(lineno: number) -> number

```



```

00377      *
00378      * @par JSON-RPC Request Example
00379      * {"jsonrpc": "2.0", "method": "RuntimeMachine.gotoLine", "params": [10], "id": 1}
00380      *
00381      * @par JSON-RPC Response Example
00382      * {"id": 1, "jsonrpc": "2.0", "result": 0}
00383      *
00384      * \endenglish
00385      */
00386      int gotoLine(int lineno);
00387
00388      /**
00389      * @ingroup RuntimeMachine
00390      * \chinese
00391      * 获取当前运行上下文
00392      *
00393      * @param tid 任务编号
00394      * 如果指定（不是-1），返回对应任务的运行上下文；如果不指定（是-1），返回正在运行的线程的运行上下文
00395      *
00396      * @return
00397      *
00398      * @par Python 函数原型
00399      * getPlanContext(self: pyaubo_sdk.RuntimeMachine) -> Tuple[int, int, str]
00400      *
00401      * @par Lua 函数原型
00402      * getPlanContext() -> number
00403      *
00404      * @par JSON-RPC 请求示例
00405      * {"jsonrpc": "2.0", "method": "RuntimeMachine.getPlanContext", "params": [-1], "id": 1}
00406      *
00407      * @par JSON-RPC 响应示例
00408      * {"id": 1, "jsonrpc": "2.0", "result": [-1, 0, ""]}
00409      *
00410      * \endchinese
00411      * \english
00412      * Get the current runtime context
00413      *
00414      * @param tid Task ID
00415      * If specified (not -1), returns the runtime context of the corresponding
00416      * task; if not specified (is -1), returns the runtime context of the
00417      * currently running thread
00418      *
00419      * @return
00420      *
00421      * @par Python function prototype
00422      * getPlanContext(self: pyaubo_sdk.RuntimeMachine) -> Tuple[int, int, str]
00423      *
00424      * @par Lua function prototype
00425      * getPlanContext() -> number
00426      *
00427      * @par JSON-RPC Request Example
00428      * {"jsonrpc": "2.0", "method": "RuntimeMachine.getPlanContext", "params": [-1], "id": 1}
00429      *
00430      * @par JSON-RPC Response Example
00431      * {"id": 1, "jsonrpc": "2.0", "result": [-1, 0, ""]}
00432      *
00433      * \endenglish
00434      */
00435      std::tuple<int, int, std::string> getPlanContext(int tid = -1);
00436
00437      /**
00438      * @ingroup RuntimeMachine
00439      * \chinese
00440      * 获取提前运行规划器的上下文信息
00441      *
00442      * @param tid 任务编号
00443      * 如果指定（不是-1），返回对应任务运行规划器的上下文信息；如果不指定（是-1），返回正在运行的线程运行规划器的上下文信息
00444      *
00445      * @return
00446      *
00447      * @par JSON-RPC 请求示例
00448      * {"jsonrpc": "2.0", "method": "RuntimeMachine.getAdvancePlanContext", "params": [-1], "id": 1}
00449      *
00450      * @par JSON-RPC 响应示例
00451      * {"id": 1, "jsonrpc": "2.0", "result": [-1, -1, ""]}
00452      *
00453      * \endchinese
00454      * \english
00455      * Get the context information of the advance planner
00456      *
00457      * @param tid Task ID
00458      * If specified (not -1), returns the context information of the advance
00459      * planner for the corresponding task; if not specified (is -1), returns the
00460      * context information of the advance planner for the currently running
00461      * thread
00462      *
00463      * @return

```

```

00464      *
00465      * @par JSON-RPC Request Example
00466      * {"jsonrpc":"2.0","method":"RuntimeMachine.getAdvancePlanContext","params":[-1],"id":1}
00467      *
00468      * @par JSON-RPC Response Example
00469      * {"id":1,"jsonrpc":"2.0","result":[-1,-1,""]}
00470      *
00471      * \endenglish
00472      */
00473      std::tuple<int, int, std::string> getAdvancePlanContext(int tid = -1);
00474
00475      /**
00476      * @ingroup RuntimeMachine
00477      * \chinese
00478      * 获取 AdvanceRun 的程序指针
00479      *
00480      * @return
00481      *
00482      * @par JSON-RPC 请求示例
00483      * {"jsonrpc":"2.0","method":"RuntimeMachine.getAdvancePtr","params":[-1],"id":1}
00484      *
00485      * @par JSON-RPC 响应示例
00486      * {"id":1,"jsonrpc":"2.0","result":-1}
00487      *
00488      * \endchinese
00489      * \english
00490      * Get the program pointer of AdvanceRun
00491      *
00492      * @return
00493      *
00494      * @par JSON-RPC Request Example
00495      * {"jsonrpc":"2.0","method":"RuntimeMachine.getAdvancePtr","params":[-1],"id":1}
00496      *
00497      * @par JSON-RPC Response Example
00498      * {"id":1,"jsonrpc":"2.0","result":-1}
00499      *
00500      * \endenglish
00501      */
00502      int getAdvancePtr(int tid = -1);
00503
00504      /**
00505      * @ingroup RuntimeMachine
00506      * \chinese
00507      * 获取机器人运动的程序指针
00508      *
00509      * @param tid 任务编号
00510      * 如果指定（不是-1），返回对应任务的程序指针；如果不指定（是-1），返回正在运行线程的程序指针
00511      *
00512      * @return
00513      *
00514      * @par JSON-RPC 请求示例
00515      * {"jsonrpc":"2.0","method":"RuntimeMachine.getMainPtr","params":[-1],"id":1}
00516      *
00517      * @par JSON-RPC 响应示例
00518      * {"id":1,"jsonrpc":"2.0","result":-1}
00519      *
00520      * \endchinese
00521      * \english
00522      * Get the program pointer of robot motion
00523      *
00524      * @param tid Task ID
00525      * If specified (not -1), returns the program pointer of the corresponding
00526      * task; if not specified (is -1), returns the program pointer of the
00527      * currently running thread
00528      *
00529      * @return
00530      *
00531      * @par JSON-RPC Request Example
00532      * {"jsonrpc":"2.0","method":"RuntimeMachine.getMainPtr","params":[-1],"id":1}
00533      *
00534      * @par JSON-RPC Response Example
00535      * {"id":1,"jsonrpc":"2.0","result":-1}
00536      *
00537      * \endenglish
00538      */
00539      int getMainPtr(int tid = -1);
00540
00541      /**
00542      * @ingroup RuntimeMachine
00543      * \chinese
00544      * 获取最近解释过的指令指针
00545      *
00546      * @param tid
00547      * @return
00548      *
00549      * @par JSON-RPC 请求示例
00550      * {"jsonrpc":"2.0","method":"RuntimeMachine.getInterpPtr","params":[26],"id":1}

```

```

00551      *
00552      * @par JSON-RPC 响应示例
00553      * {"id":1,"jsonrpc":"2.0","result":-1}
00554      *
00555      * \endchinese
00556      * \english
00557      * Get the pointer of the most recently interpreted instruction
00558      *
00559      * @param tid
00560      * @return
00561      *
00562      * @par JSON-RPC Request Example
00563      * {"jsonrpc":"2.0","method":"RuntimeMachine.getInterpPtr","params":[26],"id":1}
00564      *
00565      * @par JSON-RPC Response Example
00566      * {"id":1,"jsonrpc":"2.0","result":-1}
00567      *
00568      * \endenglish
00569      */
00570      int getInterpPtr(int tid);
00571
00572      /**
00573      * @ingroup RuntimeMachine
00574      * \chinese
00575      * 加载本地工程文件
00576      * Lua 脚本, 只需要给出文件名字, 不需要后缀, 需要从 ${ARCS_WS}/program
00577      * 目录中查找
00578      *
00579      * @param program
00580      * @return
00581      *
00582      * @par JSON-RPC 请求示例
00583      * {"jsonrpc":"2.0","method":"RuntimeMachine.loadProgram","params":["demo"],"id":1}
00584      *
00585      * @par JSON-RPC 响应示例
00586      * {"id":1,"jsonrpc":"2.0","result":0}
00587      *
00588      * \endchinese
00589      * \english
00590      * Load a local project file.
00591      * For Lua scripts, only the file name is required (no extension), and it
00592      * will be searched in the ${ARCS_WS}/program directory.
00593      *
00594      * @param program
00595      * @return
00596      *
00597      * @par JSON-RPC Request Example
00598      * {"jsonrpc":"2.0","method":"RuntimeMachine.loadProgram","params":["demo"],"id":1}
00599      *
00600      * @par JSON-RPC Response Example
00601      * {"id":1,"jsonrpc":"2.0","result":0}
00602      *
00603      * \endenglish
00604      */
00605      int loadProgram(const std::string &program);
00606
00607      /**
00608      * @ingroup RuntimeMachine
00609      * \chinese
00610      * 预加载工程文件
00611      *
00612      * @param index 0-99 工程索引号
00613      * @param program 工程名字
00614      * @return
00615      * \endchinese
00616      * \english
00617      * Preload project file
00618      *
00619      * @param index Project index number (0-99)
00620      * @param program Project name
00621      * @return
00622      * \endenglish
00623      */
00624      int preloadProgram(int index, const std::string &program);
00625
00626      /**
00627      * @ingroup RuntimeMachine
00628      * \chinese
00629      * 获取预加载工程文件名字, 如果没有加载或者超出索引范围则返回空字符串
00630      *
00631      * @param index 0-99 工程索引号
00632      * @return 工程文件名字
00633      * \endchinese
00634      * \english
00635      * Get the name of the preloaded project file. Returns an empty string if
00636      * not loaded or index is out of range.
00637      *

```

```

00638     * @param index Project index number (0-99)
00639     * @return Project file name
00640     * \endenglish
00641     */
00642     std::string getPreloadProgram(int index);
00643
00644     /**
00645     * @ingroup RuntimeMachine
00646     * \chinese
00647     * 清除所有已预加载的工程文件
00648     * 调用此方法将释放所有通过 preloadProgram
00649     * 预加载的工程索引及其关联的工程名称
00650     *
00651     * @return 成功返回 0;
00652     * \endchinese
00653     * \english
00654     * Clear all preloaded project files.
00655     * Calling this method will release all project indices and their associated
00656     * program names that were previously preloaded via preloadProgram.
00657     *
00658     * @return Returns 0 on success;
00659     * \endenglish
00660     */
00661     int clearPreloadPrograms();
00662
00663     /**
00664     * @ingroup RuntimeMachine
00665     * \chinese
00666     * 运行已经加载的工程文件
00667     *
00668     * @return
00669     *
00670     * @par JSON-RPC 请求示例
00671     * {"jsonrpc": "2.0", "method": "RuntimeMachine.runProgram", "params": [], "id": 1}
00672     *
00673     * @par JSON-RPC 响应示例
00674     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00675     *
00676     * \endchinese
00677     * \english
00678     * Run the already loaded project file
00679     *
00680     * @return
00681     *
00682     * @par JSON-RPC Request Example
00683     * {"jsonrpc": "2.0", "method": "RuntimeMachine.runProgram", "params": [], "id": 1}
00684     *
00685     * @par JSON-RPC Response Example
00686     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00687     *
00688     * \endenglish
00689     */
00690     int runProgram();
00691
00692     /**
00693     * @ingroup RuntimeMachine
00694     * \chinese
00695     * 开始运行时
00696     *
00697     * @return
00698     *
00699     * @par Python 函数原型
00700     * start(self: pyaubo_sdk.RuntimeMachine) -> int
00701     *
00702     * @par Lua 函数原型
00703     * start() -> number
00704     *
00705     * @par JSON-RPC 请求示例
00706     * {"jsonrpc": "2.0", "method": "RuntimeMachine.start", "params": [], "id": 1}
00707     *
00708     * @par JSON-RPC 响应示例
00709     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00710     *
00711     * \endchinese
00712     * \english
00713     * Start the runtime
00714     *
00715     * @return
00716     *
00717     * @par Python function prototype
00718     * start(self: pyaubo_sdk.RuntimeMachine) -> int
00719     *
00720     * @par Lua function prototype
00721     * start() -> number
00722     *
00723     * @par JSON-RPC Request Example
00724     * {"jsonrpc": "2.0", "method": "RuntimeMachine.start", "params": [], "id": 1}

```

```

00725      *
00726      * @par JSON-RPC Response Example
00727      * {"id":1,"jsonrpc":"2.0","result":0}
00728      *
00729      * \endenglish
00730      */
00731      int start();
00732
00733      /**
00734      * @ingroup RuntimeMachine
00735      * \chinese
00736      * 停止运行时即脚本运行，无法停止运行时状态为 Stopped 时的机器人运动
00737      *
00738      * 如果考虑停止机器人所有运动，可以调用 RuntimeMachine::abort 接口
00739      *
00740      * @return
00741      *
00742      * @par Python 函数原型
00743      * stop(self: pyaubo_sdk.RuntimeMachine) -> int
00744      *
00745      * @par Lua 函数原型
00746      * stop() -> number
00747      *
00748      * @par JSON-RPC 请求示例
00749      * {"jsonrpc":"2.0","method":"RuntimeMachine.stop","params":[],"id":1}
00750      *
00751      * @par JSON-RPC 响应示例
00752      * {"id":1,"jsonrpc":"2.0","result":0}
00753      *
00754      * \endchinese
00755      * \english
00756      * Stop the runtime, i.e., stop script execution. Cannot stop robot motion
00757      * when the runtime state is Stopped.
00758      *
00759      * If you want to stop all robot motion, use the RuntimeMachine::abort
00760      * interface.
00761      *
00762      * @return
00763      *
00764      * @par Python function prototype
00765      * stop(self: pyaubo_sdk.RuntimeMachine) -> int
00766      *
00767      * @par Lua function prototype
00768      * stop() -> number
00769      *
00770      * @par JSON-RPC Request Example
00771      * {"jsonrpc":"2.0","method":"RuntimeMachine.stop","params":[],"id":1}
00772      *
00773      * @par JSON-RPC Response Example
00774      * {"id":1,"jsonrpc":"2.0","result":0}
00775      *
00776      * \endenglish
00777      */
00778      int stop();
00779
00780      /**
00781      * @ingroup RuntimeMachine
00782      * \chinese
00783      * 终止机器人运行.
00784      *
00785      * 如果只是考虑停止运行时，可以调用 RuntimeMachine::stop 接口
00786      *
00787      * 如果脚本运行时处于 Running 状态，则终止运行时；如果运行时处于 Stopped
00788      * 且机器人正在移动，则停止机器人移动；如果此时力控开启了，则机器人停止力控
00789      *
00790      * @return
00791      *
00792      * @par Python 函数原型
00793      * abort(self: pyaubo_sdk.RuntimeMachine) -> int
00794      *
00795      * @par Lua 函数原型
00796      * abort() -> number
00797      *
00798      * @par JSON-RPC 请求示例
00799      * {"jsonrpc":"2.0","method":"RuntimeMachine.abort","params":[],"id":1}
00800      *
00801      * @par JSON-RPC 响应示例
00802      * {"id":1,"jsonrpc":"2.0","result":0}
00803      *
00804      * \endchinese
00805      * \english
00806      * Abort robot operation.
00807      *
00808      * If you only want to stop the runtime, you can call the
00809      * RuntimeMachine::stop interface.
00810      *
00811      * If the script runtime is in the Running state, aborts the runtime; if the

```

```

00812     * runtime is Stopped and the robot is moving, stops the robot motion; if
00813     * force control is enabled, stops force control.
00814     *
00815     * @return
00816     *
00817     * @par Python function prototype
00818     * abort(self: pyaubo_sdk.RuntimeMachine) -> int
00819     *
00820     * @par Lua function prototype
00821     * abort() -> number
00822     *
00823     * @par JSON-RPC Request Example
00824     * {"jsonrpc": "2.0", "method": "RuntimeMachine.abort", "params": [], "id": 1}
00825     *
00826     * @par JSON-RPC Response Example
00827     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00828     *
00829     * \endenglish
00830     */
00831 int abort();
00832
00833 /**
00834     * @ingroup RuntimeMachine
00835     * \chinese
00836     * 暂停解释器
00837     *
00838     * @return
00839     *
00840     * @par Python 函数原型
00841     * pause(self: pyaubo_sdk.RuntimeMachine) -> int
00842     *
00843     * @par Lua 函数原型
00844     * pause() -> number
00845     *
00846     * @par JSON-RPC 请求示例
00847     * {"jsonrpc": "2.0", "method": "RuntimeMachine.pause", "params": [], "id": 1}
00848     *
00849     * @par JSON-RPC 响应示例
00850     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00851     *
00852     * \endchinese
00853     * \english
00854     * Pause the interpreter
00855     *
00856     * @return
00857     *
00858     * @par Python function prototype
00859     * pause(self: pyaubo_sdk.RuntimeMachine) -> int
00860     *
00861     * @par Lua function prototype
00862     * pause() -> number
00863     *
00864     * @par JSON-RPC Request Example
00865     * {"jsonrpc": "2.0", "method": "RuntimeMachine.pause", "params": [], "id": 1}
00866     *
00867     * @par JSON-RPC Response Example
00868     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00869     *
00870     * \endenglish
00871     */
00872 int pause();
00873
00874 /**
00875     * @ingroup RuntimeMachine
00876     * \chinese
00877     * 单步运行
00878     *
00879     * @return
00880     *
00881     * @par Python 函数原型
00882     * step(self: pyaubo_sdk.RuntimeMachine) -> int
00883     *
00884     * @par Lua 函数原型
00885     * step() -> number
00886     *
00887     * @par JSON-RPC 请求示例
00888     * {"jsonrpc": "2.0", "method": "RuntimeMachine.step", "params": [], "id": 1}
00889     *
00890     * @par JSON-RPC 响应示例
00891     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00892     *
00893     * \endchinese
00894     * \english
00895     * Execute a single step
00896     *
00897     * @return
00898     *

```

```

00899     * @par Python function prototype
00900     * step(self: pyaubo_sdk.RuntimeMachine) -> int
00901     *
00902     * @par Lua function prototype
00903     * step() -> number
00904     *
00905     * @par JSON-RPC Request Example
00906     * {"jsonrpc": "2.0", "method": "RuntimeMachine.step", "params": [], "id": 1}
00907     *
00908     * @par JSON-RPC Response Example
00909     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00910     *
00911     * \endenglish
00912     */
00913     int step();
00914
00915     /**
00916     * @ingroup RuntimeMachine
00917     * \chinese
00918     * 恢复解释器
00919     *
00920     * @return
00921     *
00922     * @par Python 函数原型
00923     * resume(self: pyaubo_sdk.RuntimeMachine) -> int
00924     *
00925     * @par Lua 函数原型
00926     * resume() -> number
00927     *
00928     * @par JSON-RPC 请求示例
00929     * {"jsonrpc": "2.0", "method": "RuntimeMachine.resume", "params": [], "id": 1}
00930     *
00931     * @par JSON-RPC 响应示例
00932     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00933     *
00934     * \endchinese
00935     * \english
00936     * Resume the interpreter
00937     *
00938     * @return
00939     *
00940     * @par Python function prototype
00941     * resume(self: pyaubo_sdk.RuntimeMachine) -> int
00942     *
00943     * @par Lua function prototype
00944     * resume() -> number
00945     *
00946     * @par JSON-RPC Request Example
00947     * {"jsonrpc": "2.0", "method": "RuntimeMachine.resume", "params": [], "id": 1}
00948     *
00949     * @par JSON-RPC Response Example
00950     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00951     *
00952     * \endenglish
00953     */
00954     int resume();
00955
00956     /**
00957     * @ingroup RuntimeMachine
00958     * \chinese
00959     * 恢复解释器（不检查当前点和暂停点距离）
00960     *
00961     * @return
00962     *
00963     * @par Python 函数原型
00964     * arbitraryResume(self: pyaubo_sdk.RuntimeMachine) -> int
00965     *
00966     * @par Lua 函数原型
00967     * arbitraryResume() -> number
00968     *
00969     * @par JSON-RPC 请求示例
00970     * {"jsonrpc": "2.0", "method": "RuntimeMachine.arbitraryResume", "params": [], "id": 1}
00971     *
00972     * @par JSON-RPC 响应示例
00973     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00974     *
00975     * \endchinese
00976     * \english
00977     * Resume the interpreter
00978     *
00979     * @return
00980     *
00981     * @par Python function prototype
00982     * arbitraryResume(self: pyaubo_sdk.RuntimeMachine) -> int
00983     *
00984     * @par Lua function prototype
00985     * arbitraryResume() -> number

```

```

00986      *
00987      * @par JSON-RPC Request Example
00988      * {"jsonrpc": "2.0", "method": "RuntimeMachine.arbitraryResume", "params": [], "id": 1}
00989      *
00990      * @par JSON-RPC Response Example
00991      * {"id": 1, "jsonrpc": "2.0", "result": 0}
00992      *
00993      * \endenglish
00994      */
00995      int arbitraryResume();
00996
00997      /**
00998      * @ingroup RuntimeMachine
00999      * \chinese
01000      * 恢复解释器之前等待恢复前之前的序列完成
01001      *
01002      * @param wait
01003      * @return
01004      *
01005      * @par JSON-RPC 请求示例
01006      * {"jsonrpc": "2.0", "method": "RuntimeMachine.setResumeWait", "params": [true], "id": 1}
01007      *
01008      * @par JSON-RPC 响应示例
01009      * {"id": 1, "jsonrpc": "2.0", "result": 0}
01010      *
01011      * \endchinese
01012      * \english
01013      * Wait for the previous sequence to complete before resuming the
01014      * interpreter
01015      *
01016      * @param wait
01017      * @return
01018      *
01019      * @par JSON-RPC Request Example
01020      * {"jsonrpc": "2.0", "method": "RuntimeMachine.setResumeWait", "params": [true], "id": 1}
01021      *
01022      * @par JSON-RPC Response Example
01023      * {"id": 1, "jsonrpc": "2.0", "result": 0}
01024      *
01025      * \endenglish
01026      */
01027      int setResumeWait(bool wait);
01028
01029      /**
01030      * @ingroup RuntimeMachine
01031      * \chinese
01032      * 进入临界区, abort 命令会被推迟执行, 避免临界区内的指令被打断
01033      *
01034      * @param timeout 单位秒, 范围 0-5 秒, 表示 abort
01035      * 命令最大推迟时间, 超过这个时间自动退出临界区并 abort
01036      * @return
01037      *
01038      * @par Python 函数原型
01039      * enterCritical(self: pyaubo_sdk.RuntimeMachine, arg0: double) -> int
01040      *
01041      * @par Lua 函数原型
01042      * enterCritical(timeout: number) -> number
01043      *
01044      * @par JSON-RPC 请求示例
01045      * {"jsonrpc": "2.0", "method": "RuntimeMachine.enterCritical", "params": [5.0], "id": 1}
01046      *
01047      * @par JSON-RPC 响应示例
01048      * {"id": 1, "jsonrpc": "2.0", "result": 0}
01049      *
01050      * \endchinese
01051      * \english
01052      * The abort command is deferred during critical sections to avoid
01053      * interrupting internal instructions.
01054      *
01055      * @param timeout (seconds, 0-5): max deferral time for abort. Exceeding it
01056      * forces exit from critical section and triggers abort.
01057      * @return
01058      *
01059      * @par Python function prototype
01060      * enterCritical(self: pyaubo_sdk.RuntimeMachine, arg0: double) -> int
01061      *
01062      * @par Lua function prototype
01063      * enterCritical(timeout: number) -> number
01064      *
01065      * @par JSON-RPC Request Example
01066      * {"jsonrpc": "2.0", "method": "RuntimeMachine.enterCritical", "params": [5.0], "id": 1}
01067      *
01068      * @par JSON-RPC Response Example
01069      * {"id": 1, "jsonrpc": "2.0", "result": 0}
01070      *
01071      * \endenglish
01072      */

```



```

01073     int enterCritical(double timeout);
01074
01075     /**
01076      * @ingroup RuntimeMachine
01077      * \chinese
01078      * 退出临界区
01079      *
01080      * @param
01081      * @return
01082      *
01083      * @par Python 函数原型
01084      * exitCritical(self: pyaubo_sdk.RuntimeMachine) -> int
01085      *
01086      * @par Lua 函数原型
01087      * exitCritical() -> number
01088      *
01089      * @par JSON-RPC 请求示例
01090      * {"jsonrpc": "2.0", "method": "RuntimeMachine.exitCritical", "params": [], "id": 1}
01091      *
01092      * @par JSON-RPC 响应示例
01093      * {"id": 1, "jsonrpc": "2.0", "result": 0}
01094      *
01095      * \endchinese
01096      * \english
01097      * Exit the critical section
01098      *
01099      * @param
01100      * @return
01101      *
01102      * @par Python function prototype
01103      * exitCritical(self: pyaubo_sdk.RuntimeMachine) -> int
01104      *
01105      * @par Lua function prototype
01106      * exitCritical() -> number
01107      *
01108      * @par JSON-RPC Request Example
01109      * {"jsonrpc": "2.0", "method": "RuntimeMachine.exitCritical", "params": [], "id": 1}
01110      *
01111      * @par JSON-RPC Response Example
01112      * {"id": 1, "jsonrpc": "2.0", "result": 0}
01113      *
01114      * \endenglish
01115      */
01116     int exitCritical();
01117
01118     /**
01119      * @ingroup RuntimeMachine
01120      * \chinese
01121      * 获取规划器的状态
01122      *
01123      * @return
01124      *
01125      * @par Python 函数原型
01126      * getStatus(self: pyaubo_sdk.RuntimeMachine) ->
01127      * arcs::common_interface::RuntimeState
01128      *
01129      * @par Lua 函数原型
01130      * getStatus() -> number
01131      *
01132      * @par JSON-RPC 请求示例
01133      * {"jsonrpc": "2.0", "method": "RuntimeMachine.getStatus", "params": [], "id": 1}
01134      *
01135      * @par JSON-RPC 响应示例
01136      * {"id": 1, "jsonrpc": "2.0", "result": "Running"}
01137      *
01138      * \endchinese
01139      * \english
01140      * Get the status of the planner
01141      *
01142      * @return
01143      *
01144      * @par Python function prototype
01145      * getStatus(self: pyaubo_sdk.RuntimeMachine) ->
01146      * arcs::common_interface::RuntimeState
01147      *
01148      * @par Lua function prototype
01149      * getStatus() -> number
01150      *
01151      * @par JSON-RPC Request Example
01152      * {"jsonrpc": "2.0", "method": "RuntimeMachine.getStatus", "params": [], "id": 1}
01153      *
01154      * @par JSON-RPC Response Example
01155      * {"id": 1, "jsonrpc": "2.0", "result": "Running"}
01156      *
01157      * \endenglish
01158      */
01159     ARCS_DEPRECATED RuntimeState getStatus();

```

```

01160     RuntimeState getRuntimeState();
01161
01162     /**
01163      * @ingroup RuntimeMachine
01164      * \chinese
01165      * 设置断点
01166      *
01167      * @param lineno
01168      * @return
01169      *
01170      * @par Python 函数原型
01171      * setBreakPoint(self: pyaubo_sdk.RuntimeMachine, arg0: int) -> int
01172      *
01173      * @par Lua 函数原型
01174      * setBreakPoint(lineno: number) -> number
01175      *
01176      * @par JSON-RPC 请求示例
01177      * {"jsonrpc": "2.0", "method": "RuntimeMachine.setBreakPoint", "params": [15], "id": 1}
01178      *
01179      * @par JSON-RPC 响应示例
01180      * {"id": 1, "jsonrpc": "2.0", "result": 0}
01181      *
01182      * \endchinese
01183      * \english
01184      * Set a breakpoint
01185      *
01186      * @param lineno
01187      * @return
01188      *
01189      * @par Python function prototype
01190      * setBreakPoint(self: pyaubo_sdk.RuntimeMachine, arg0: int) -> int
01191      *
01192      * @par Lua function prototype
01193      * setBreakPoint(lineno: number) -> number
01194      *
01195      * @par JSON-RPC Request Example
01196      * {"jsonrpc": "2.0", "method": "RuntimeMachine.setBreakPoint", "params": [15], "id": 1}
01197      *
01198      * @par JSON-RPC Response Example
01199      * {"id": 1, "jsonrpc": "2.0", "result": 0}
01200      *
01201      * \endenglish
01202      */
01203     int setBreakPoint(int lineno);
01204
01205     /**
01206      * @ingroup RuntimeMachine
01207      * \chinese
01208      * 移除断点
01209      *
01210      * @param lineno
01211      * @return
01212      *
01213      * @par Python 函数原型
01214      * removeBreakPoint(self: pyaubo_sdk.RuntimeMachine, arg0: int) -> int
01215      *
01216      * @par Lua 函数原型
01217      * removeBreakPoint(lineno: number) -> number
01218      *
01219      * @par JSON-RPC 请求示例
01220      * {"jsonrpc": "2.0", "method": "RuntimeMachine.removeBreakPoint", "params": [15], "id": 1}
01221      *
01222      * @par JSON-RPC 响应示例
01223      * {"id": 1, "jsonrpc": "2.0", "result": 0}
01224      *
01225      * \endchinese
01226      * \english
01227      * Remove a breakpoint
01228      *
01229      * @param lineno
01230      * @return
01231      *
01232      * @par Python function prototype
01233      * removeBreakPoint(self: pyaubo_sdk.RuntimeMachine, arg0: int) -> int
01234      *
01235      * @par Lua function prototype
01236      * removeBreakPoint(lineno: number) -> number
01237      *
01238      * @par JSON-RPC Request Example
01239      * {"jsonrpc": "2.0", "method": "RuntimeMachine.removeBreakPoint", "params": [15], "id": 1}
01240      *
01241      * @par JSON-RPC Response Example
01242      * {"id": 1, "jsonrpc": "2.0", "result": 0}
01243      *
01244      * \endenglish
01245      */
01246     int removeBreakPoint(int lineno);

```

```

01247
01248 /**
01249  * @ingroup RuntimeMachine
01250  * \chinese
01251  * 清除所有断点
01252  *
01253  * @return
01254  *
01255  * @par Python 函数原型
01256  * clearBreakPoints(self: pyaubo_sdk.RuntimeMachine) -> int
01257  *
01258  * @par Lua 函数原型
01259  * clearBreakPoints() -> number
01260  *
01261  * @par JSON-RPC 请求示例
01262  * {"jsonrpc": "2.0", "method": "RuntimeMachine.clearBreakPoints", "params": [], "id": 1}
01263  *
01264  * @par JSON-RPC 响应示例
01265  * {"id": 1, "jsonrpc": "2.0", "result": 0}
01266  *
01267  * \endchinese
01268  * \english
01269  * Clear all breakpoints
01270  *
01271  * @return
01272  *
01273  * @par Python function prototype
01274  * clearBreakPoints(self: pyaubo_sdk.RuntimeMachine) -> int
01275  *
01276  * @par Lua function prototype
01277  * clearBreakPoints() -> number
01278  *
01279  * @par JSON-RPC Request Example
01280  * {"jsonrpc": "2.0", "method": "RuntimeMachine.clearBreakPoints", "params": [], "id": 1}
01281  *
01282  * @par JSON-RPC Response Example
01283  * {"id": 1, "jsonrpc": "2.0", "result": 0}
01284  *
01285  * \endenglish
01286  */
01287 int clearBreakPoints();
01288
01289 /**
01290  * @ingroup RuntimeMachine
01291  * \chinese
01292  * 定时器开始
01293  *
01294  * @param name
01295  * @return
01296  *
01297  * @par Python 函数原型
01298  * timerStart(self: pyaubo_sdk.RuntimeMachine, arg0: str) -> int
01299  *
01300  * @par Lua 函数原型
01301  * timerStart(name: string) -> nil
01302  *
01303  * @par JSON-RPC 请求示例
01304  * {"jsonrpc": "2.0", "method": "RuntimeMachine.timerStart", "params": ["timer"], "id": 1}
01305  *
01306  * @par JSON-RPC 响应示例
01307  * {"id": 1, "jsonrpc": "2.0", "result": 0}
01308  *
01309  * \endchinese
01310  * \english
01311  * Start the timer
01312  *
01313  * @param name
01314  * @return
01315  *
01316  * @par Python function prototype
01317  * timerStart(self: pyaubo_sdk.RuntimeMachine, arg0: str) -> int
01318  *
01319  * @par Lua function prototype
01320  * timerStart(name: string) -> nil
01321  *
01322  * @par JSON-RPC Request Example
01323  * {"jsonrpc": "2.0", "method": "RuntimeMachine.timerStart", "params": ["timer"], "id": 1}
01324  *
01325  * @par JSON-RPC Response Example
01326  * {"id": 1, "jsonrpc": "2.0", "result": 0}
01327  *
01328  * \endenglish
01329  */
01330 int timerStart(const std::string &name);
01331
01332 /**
01333  * @ingroup RuntimeMachine

```

```

01334     * \chinese
01335     * 定时器结束
01336     *
01337     * @param name
01338     * @return
01339     *
01340     * @par Python 函数原型
01341     * timerStop(self: pyaubo_sdk.RuntimeMachine, arg0: str) -> int
01342     *
01343     * @par Lua 函数原型
01344     * timerStop(name: string) -> nil
01345     *
01346     * @par JSON-RPC 请求示例
01347     * {"jsonrpc": "2.0", "method": "RuntimeMachine.timerStop", "params": ["timer"], "id": 1}
01348     *
01349     * @par JSON-RPC 响应示例
01350     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01351     *
01352     * \endchinese
01353     * \english
01354     * Stop the timer
01355     *
01356     * @param name
01357     * @return
01358     *
01359     * @par Python function prototype
01360     * timerStop(self: pyaubo_sdk.RuntimeMachine, arg0: str) -> int
01361     *
01362     * @par Lua function prototype
01363     * timerStop(name: string) -> nil
01364     *
01365     * @par JSON-RPC Request Example
01366     * {"jsonrpc": "2.0", "method": "RuntimeMachine.timerStop", "params": ["timer"], "id": 1}
01367     *
01368     * @par JSON-RPC Response Example
01369     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01370     *
01371     * \endenglish
01372     */
01373 int timerStop(const std::string &name);
01374
01375 /**
01376  * @ingroup RuntimeMachine
01377  * \chinese
01378  * 定时器重置
01379  *
01380  * @param name
01381  * @return
01382  *
01383  * @par Python 函数原型
01384  * timerReset(self: pyaubo_sdk.RuntimeMachine, arg0: str) -> int
01385  *
01386  * @par Lua 函数原型
01387  * timerReset(name: string) -> nil
01388  *
01389  * @par JSON-RPC 请求示例
01390  * {"jsonrpc": "2.0", "method": "RuntimeMachine.timerReset", "params": ["timer"], "id": 1}
01391  *
01392  * @par JSON-RPC 响应示例
01393  * {"id": 1, "jsonrpc": "2.0", "result": 0}
01394  *
01395  * \endchinese
01396  * \english
01397  * Reset the timer
01398  *
01399  * @param name
01400  * @return
01401  *
01402  * @par Python function prototype
01403  * timerReset(self: pyaubo_sdk.RuntimeMachine, arg0: str) -> int
01404  *
01405  * @par Lua function prototype
01406  * timerReset(name: string) -> nil
01407  *
01408  * @par JSON-RPC Request Example
01409  * {"jsonrpc": "2.0", "method": "RuntimeMachine.timerReset", "params": ["timer"], "id": 1}
01410  *
01411  * @par JSON-RPC Response Example
01412  * {"id": 1, "jsonrpc": "2.0", "result": 0}
01413  *
01414  * \endenglish
01415  */
01416 int timerReset(const std::string &name);
01417
01418 /**
01419  * @ingroup RuntimeMachine
01420  * \chinese

```

```

01421     * 定时器删除
01422     *
01423     * @param name
01424     * @return
01425     *
01426     * @par Python 函数原型
01427     * timerDelete(self: pyaubo_sdk.RuntimeMachine, arg0: str) -> int
01428     *
01429     * @par Lua 函数原型
01430     * timerDelete(name: string) -> nil
01431     *
01432     * @par JSON-RPC 请求示例
01433     * {"jsonrpc": "2.0", "method": "RuntimeMachine.timerDelete", "params": ["timer"], "id": 1}
01434     *
01435     * @par JSON-RPC 响应示例
01436     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01437     *
01438     * \endchinese
01439     * \english
01440     * Delete the timer
01441     *
01442     * @param name
01443     * @return
01444     *
01445     * @par Python function prototype
01446     * timerDelete(self: pyaubo_sdk.RuntimeMachine, arg0: str) -> int
01447     *
01448     * @par Lua function prototype
01449     * timerDelete(name: string) -> nil
01450     *
01451     * @par JSON-RPC Request Example
01452     * {"jsonrpc": "2.0", "method": "RuntimeMachine.timerDelete", "params": ["timer"], "id": 1}
01453     *
01454     * @par JSON-RPC Response Example
01455     * {"id": 1, "jsonrpc": "2.0", "result": 0}
01456     *
01457     * \endenglish
01458     */
01459 int timerDelete(const std::string &name);
01460
01461 /**
01462  * @ingroup RuntimeMachine
01463  * \chinese
01464  * 获取定时器数值
01465  *
01466  * @param name
01467  * @return
01468  *
01469  * @par Python 函数原型
01470  * getTimer(self: pyaubo_sdk.RuntimeMachine, arg0: str) -> float
01471  *
01472  * @par Lua 函数原型
01473  * getTimer(name: string) -> number
01474  *
01475  * @par JSON-RPC 请求示例
01476  * {"jsonrpc": "2.0", "method": "RuntimeMachine.getTimer", "params": ["timer"], "id": 1}
01477  *
01478  * @par JSON-RPC 响应示例
01479  * {"id": 1, "jsonrpc": "2.0", "result": 25.409769612}
01480  *
01481  * \endchinese
01482  * \english
01483  * Get the timer value
01484  *
01485  * @param name
01486  * @return
01487  *
01488  * @par Python function prototype
01489  * getTimer(self: pyaubo_sdk.RuntimeMachine, arg0: str) -> float
01490  *
01491  * @par Lua function prototype
01492  * getTimer(name: string) -> number
01493  *
01494  * @par JSON-RPC Request Example
01495  * {"jsonrpc": "2.0", "method": "RuntimeMachine.getTimer", "params": ["timer"], "id": 1}
01496  *
01497  * @par JSON-RPC Response Example
01498  * {"id": 1, "jsonrpc": "2.0", "result": 25.409769612}
01499  *
01500  * \endenglish
01501  */
01502 double getTimer(const std::string &name);
01503
01504 /**
01505  * @ingroup RuntimeMachine
01506  * \chinese
01507  * 开始配置触发

```

```

01508      *
01509      * @param distance
01510      * @param delay
01511      * @return
01512      *
01513      * @par JSON-RPC 请求示例
01514      * {"jsonrpc":"2.0","method":"RuntimeMachine.triggBegin","params":[],"id":1}
01515      *
01516      * @par JSON-RPC 响应示例
01517      * {"id":1,"jsonrpc":"2.0","result":0}
01518      *
01519      * \endchinese
01520      * \english
01521      * Start configuring trigger
01522      *
01523      * @param distance
01524      * @param delay
01525      * @return
01526      *
01527      * @par JSON-RPC Request Example
01528      * {"jsonrpc":"2.0","method":"RuntimeMachine.triggBegin","params":[],"id":1}
01529      *
01530      * @par JSON-RPC Response Example
01531      * {"id":1,"jsonrpc":"2.0","result":0}
01532      *
01533      * \endenglish
01534      */
01535 int triggBegin(double distance, double delay);
01536
01537 /**
01538  * @ingroup RuntimeMachine
01539  * \chinese
01540  * 终止配置触发
01541  *
01542  * @return
01543  *
01544  * @par JSON-RPC 请求示例
01545  * {"jsonrpc":"2.0","method":"RuntimeMachine.triggEnd","params":[],"id":1}
01546  *
01547  * @par JSON-RPC 响应示例
01548  * {"id":1,"jsonrpc":"2.0","result":0}
01549  *
01550  * \endchinese
01551  * \english
01552  * End configuring trigger
01553  *
01554  * @return
01555  *
01556  * @par JSON-RPC Request Example
01557  * {"jsonrpc":"2.0","method":"RuntimeMachine.triggEnd","params":[],"id":1}
01558  *
01559  * @par JSON-RPC Response Example
01560  * {"id":1,"jsonrpc":"2.0","result":0}
01561  *
01562  * \endenglish
01563  */
01564 int triggEnd();
01565
01566 /**
01567  * @ingroup RuntimeMachine
01568  * \chinese
01569  * 返回自动分配的中断号
01570  *
01571  * @param distance
01572  * @param delay
01573  * @param intnum
01574  * @return
01575  * \endchinese
01576  * \english
01577  * Returns the automatically assigned interrupt number
01578  *
01579  * @param distance
01580  * @param delay
01581  * @param intnum
01582  * @return
01583  * \endenglish
01584  */
01585 int triggInterrupt(double distance, double delay);
01586
01587 /**
01588  * @ingroup RuntimeMachine
01589  * \chinese
01590  * 获取所有的中断号列表
01591  *
01592  * @return
01593  *
01594  * @par JSON-RPC 请求示例

```

```

01595     * {"jsonrpc":"2.0","method":"RuntimeMachine.getTriggInterrupts","params":[],"id":1}
01596     *
01597     * @par JSON-RPC 响应示例
01598     * {"id":1,"jsonrpc":"2.0","result":[]}
01599     *
01600     * \endchinese
01601     * \english
01602     * Get the list of all interrupt numbers
01603     *
01604     * @return
01605     *
01606     * @par JSON-RPC Request Example
01607     * {"jsonrpc":"2.0","method":"RuntimeMachine.getTriggInterrupts","params":[],"id":1}
01608     *
01609     * @par JSON-RPC Response Example
01610     * {"id":1,"jsonrpc":"2.0","result":[]}
01611     *
01612     * \endenglish
01613     */
01614     std::vector<int> getTriggInterrupts();
01615
01616 protected:
01617     void *d_;
01618 };
01619
01620 using RuntimeMachinePtr = std::shared_ptr<RuntimeMachine>;
01621
01622 } // namespace common_interface
01623 } // namespace arcs
01624 #endif // AUBO_SDK_RUNTIME_MACHINE_H

```

12.38 include/aubo/serial.h 文件参考

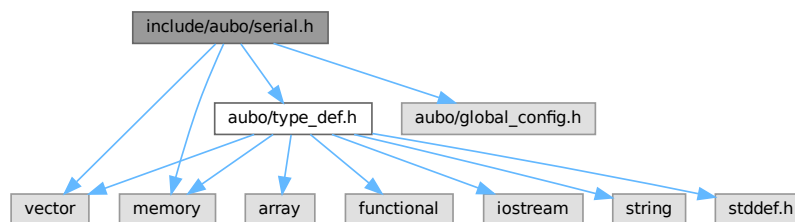
串口通信

```

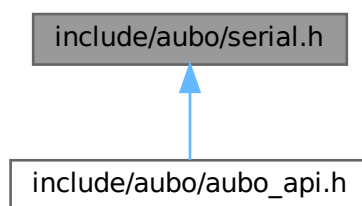
#include <vector>
#include <memory>
#include <aubo/type_def.h>
#include <aubo/global_config.h>

```

serial.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::common_interface::Serial](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::common_interface](#)

类型定义

- using [arcs::common_interface::SerialPtr](#) = std::shared_ptr<[Serial](#)>

12.38.1 详细描述

串口通信

在文件 [serial.h](#) 中定义.

12.39 serial.h

[浏览该文件的文档.](#)

```

00001 /** @file serial.h
00002  * @brief 串口通信
00003  */
00004 #ifndef AUBO_SDK_SERIAL_INTERFACE_H
00005 #define AUBO_SDK_SERIAL_INTERFACE_H
00006
00007 #include <vector>
00008 #include <memory>
00009
00010 #include <aubo/type_def.h>
00011 #include <aubo/global_config.h>
00012
00013 namespace arcs {
00014 namespace common_interface {
00015
00016 /**
00017  * @defgroup Serial Serial (串口通信)
00018  * \-chinese 串口通信
00019  * \-english Serial control
00020  */
00021 class ARCS_ABI_EXPORT Serial
00022 {
00023 public:
00024     Serial();
00025     virtual ~Serial();
00026
00027     /**
00028      * @ingroup Serial
00029      * \-chinese
00030      * 打开 TCP/IP 以太网通信串口
00031      *
  
```



```

00032     * @param device 设备名
00033     * @param baud 波特率
00034     * @param stop_bits 停止位
00035     * @param even 校验位
00036     * @param serial_name 串口名称
00037     * @return 返回值
00038     *
00039     * @par Python 函数原型
00040     * serialOpen(self: pyaubo_sdk.Serial, arg0: str, arg1: int, arg2: float,
00041     * arg3: int, arg4: str) -> int
00042     *
00043     * @par Lua 函数原型
00044     * serialOpen(device: string, baud: number, stop_bits: number, even: number,
00045     * serial_name: string) -> nil
00046     * \endchinese
00047     * \english
00048     * Open TCP/IP ethernet communication serial
00049     *
00050     * @param device
00051     * @param baud
00052     * @param stop_bits
00053     * @param even
00054     * @param serial_name
00055     * @return
00056     *
00057     * @par Python function prototype
00058     * serialOpen(self: pyaubo_sdk.Serial, arg0: str, arg1: int, arg2: float,
00059     * arg3: int, arg4: str) -> int
00060     *
00061     * @par Lua function prototype
00062     * serialOpen(device: string, baud: number, stop_bits: number, even: number,
00063     * serial_name: string) -> nil
00064     * \endenglish
00065     */
00066
00067 int serialOpen(const std::string &device, int baud, float stop_bits,
00068               int even, const std::string &serial_name = "serial_0");
00069
00070 /**
00071     * @ingroup Serial
00072     * \chinese
00073     * 关闭 TCP/IP 串口通信
00074     * 关闭与服务器的串口连接。
00075     *
00076     * @param serial_name 串口名称
00077     * @return 返回值
00078     *
00079     * @par Python 函数原型
00080     * serialClose(self: pyaubo_sdk.Serial, arg0: str) -> int
00081     *
00082     * @par Lua 函数原型
00083     * serialClose(serial_name: string) -> nil
00084     * \endchinese
00085     * \english
00086     * Close TCP/IP serial communication
00087     * Close down the serial connection to the server.
00088     *
00089     * @param serial_name
00090     * @return
00091     *
00092     * @par Python function prototype
00093     * serialClose(self: pyaubo_sdk.Serial, arg0: str) -> int
00094     *
00095     * @par Lua function prototype
00096     * serialClose(serial_name: string) -> nil
00097     * \endenglish
00098     */
00099 int serialClose(const std::string &serial_name = "serial_0");
00100
00101 /**
00102     * @ingroup Serial
00103     * \chinese
00104     * 从串口读取指定数量的字节。字节为网络字节序。一次最多可读取 30 个值。
00105     *
00106     * @param variable 变量
00107     * @param serial_name 串口名称
00108     * @return 返回值
00109     *
00110     * @par Python 函数原型
00111     * serialReadByte(self: pyaubo_sdk.Serial, arg0: str, arg1: str) -> int
00112     *
00113     * @par Lua 函数原型
00114     * serialReadByte(variable: string, serial_name: string) -> number
00115     * \endchinese
00116     * \english
00117     * Reads a number of bytes from the serial. Bytes are in network byte
00118     * order. A maximum of 30 values can be read in one command.

```

```

00119     *
00120     * @param variable
00121     * @param serial_name
00122     * @return
00123     *
00124     * @par Python function prototype
00125     * serialReadByte(self: pyaubo_sdk.Serial, arg0: str, arg1: str) -> int
00126     *
00127     * @par Lua function prototype
00128     * serialReadByte(variable: string, serial_name: string) -> number
00129     * \endenglish
00130     */
00131 int serialReadByte(const std::string &variable,
00132                   const std::string &serial_name = "serial_0");
00133
00134 /**
00135  * @ingroup Serial
00136  * \chinese
00137  * 从串口读取指定数量的字节。字节为网络字节序。一次最多可读取 30 个值。
00138  * 返回读取到的数字列表 (int 列表, 长度 =number+1)。
00139  *
00140  * @param number 读取的字节数
00141  * @param variable 变量
00142  * @param serial_name 串口名称
00143  * @return 返回值
00144  *
00145  * @par Python 函数原型
00146  * serialReadByteList(self: pyaubo_sdk.Serial, arg0: int, arg1: str, arg2: str) -> int
00147  *
00148  * @par Lua 函数原型
00149  * serialReadByteList(number: number, variable: string, serial_name: string) -> number
00150  * \endchinese
00151  * \english
00152  * Reads a number of bytes from the serial. Bytes are in network byte
00153  * order. A maximum of 30 values can be read in one command.
00154  * A list of numbers read (list of ints, length=number+1)
00155  *
00156  * @param number Number of bytes to read
00157  * @param variable
00158  * @param serial_name Serial port name
00159  * @return Return value
00160  *
00161  * @par Python function prototype
00162  * serialReadByteList(self: pyaubo_sdk.Serial, arg0: int, arg1: str, arg2: str) -> int
00163  *
00164  * @par Lua function prototype
00165  * serialReadByteList(number: number, variable: string, serial_name: string) -> number
00166  * \endenglish
00167  */
00168 int serialReadByteList(int number, const std::string &variable,
00169                       const std::string &serial_name = "serial_0");
00170
00171 /**
00172  * @ingroup Serial
00173  * \chinese
00174  * 从串口读取所有数据，并将数据作为字符串返回。
00175  * 字节为网络字节序。
00176  *
00177  * 可选参数 "prefix" 和 "suffix" 用于指定从串口提取的内容。
00178  * "prefix" 指定提取子串 (消息) 的起始位置。直到 "prefix" 结束的数据会被忽略并从串口移除。
00179  * "suffix" 指定提取子串 (消息) 的结束位置。串口中 "suffix" 之后的剩余数据会被保留。
00180  * 例如，如果串口服务器发送字符串 "noise>hello<", 控制器可以通过设置 prefix=">" 和 suffix="<" 来接收 "hello"。
00181  * 通过使用 "prefix" 和 "suffix", 还可以一次向控制器发送多条字符串，因为 "suffix" 定义了消息的结束位置。
00182  * 例如发送 ">hello<>world<"
00183  *
00184  * @param variable 变量
00185  * @param serial_name 串口名称
00186  * @param prefix 前缀
00187  * @param suffix 后缀
00188  * @param interpret_escape 是否解释转义字符
00189  * @return 返回值
00190  *
00191  * @par Python 函数原型
00192  * serialReadString(self: pyaubo_sdk.Serial, arg0: str, arg1: str, arg2: str, arg3: str, arg4: bool) -> int
00193  *
00194  * @par Lua 函数原型
00195  * serialReadString(variable: string, serial_name: string, prefix: string, suffix: string, interpret_escape:
00196  * boolean) -> number
00197  * \endchinese
00198  * \english
00199  * Reads all data from the serial and returns the data as a string.
00200  * Bytes are in network byte order.
00201  *
00202  * The optional parameters "prefix" and "suffix", can be used to express
00203  * what is extracted from the serial. The "prefix" specifies the start
00204  * of the substring (message) extracted from the serial. The data up to
00205  * the end of the "prefix" will be ignored and removed from the serial.

```

```

00205     * The "suffix" specifies the end of the substring (message) extracted
00206     * from the serial. Any remaining data on the serial, after the "suffix",
00207     * will be preserved. E.g. if the serial server sends a string
00208     * "noise>hello<", the controller can receive the "hello" by calling this
00209     * script function with the prefix=">" and suffix="<". By using the
00210     * "prefix" and "suffix" it is also possible send multiple string to the
00211     * controller at once, because the suffix defines where the message ends.
00212     * E.g. sending ">hello<>world<"
00213     *
00214     * @param variable
00215     * @param serial_name
00216     * @param prefix
00217     * @param suffix
00218     * @param interpret_escape
00219     * @return
00220     *
00221     * @par Python function prototype
00222     * serialReadString(self: pyaubo_sdk.Serial, arg0: str, arg1: str, arg2: str, arg3: str, arg4: bool) -> int
00223     *
00224     * @par Lua function prototype
00225     * serialReadString(variable: string, serial_name: string, prefix: string, suffix: string, interpret_escape:
boolean) -> number
00226     * \endenglish
00227     */
00228     int serialReadString(const std::string &variable,
00229                         const std::string &serial_name = "serial_0",
00230                         const std::string &prefix = "",
00231                         const std::string &suffix = "",
00232                         bool interpret_escape = false);
00233
00234     /**
00235     * @ingroup Serial
00236     * \chinese
00237     * 发送一个字节到服务器
00238     * 通过串口发送字节 <value>。不期望有响应。可用于发送特殊的 ASCII 字符；10 为换行符，2 为文本开始，3 为文本结束。
00239     *
00240     * @param value 字节值
00241     * @param serial_name 串口名称
00242     * @return 返回值
00243     *
00244     * @par Python 函数原型
00245     * serialSendByte(self: pyaubo_sdk.Serial, arg0: str, arg1: str) -> int
00246     *
00247     * @par Lua 函数原型
00248     * serialSendByte(value: string, serial_name: string) -> nil
00249     * \endchinese
00250     * \english
00251     * Sends a byte to the server
00252     * Sends the byte <value> through the serial. Expects no response. Can
00253     * be used to send special ASCII characters; 10 is newline, 2 is start of
00254     * text, 3 is end of text.
00255     *
00256     * @param value
00257     * @param serial_name
00258     * @return
00259     *
00260     * @par Python function prototype
00261     * serialSendByte(self: pyaubo_sdk.Serial, arg0: str, arg1: str) -> int
00262     *
00263     * @par Lua function prototype
00264     * serialSendByte(value: string, serial_name: string) -> nil
00265     * \endenglish
00266     */
00267     int serialSendByte(char value, const std::string &serial_name = "serial_0");
00268
00269     /**
00270     * @ingroup Serial
00271     * \chinese
00272     * 发送一个整数 (int32_t) 到服务器
00273     * 通过串口发送整数 <value>。以网络字节序发送。不期望有响应。
00274     *
00275     * @param value 整数值
00276     * @param serial_name 串口名称
00277     * @return 返回值
00278     *
00279     * @par Python 函数原型
00280     * serialSendInt(self: pyaubo_sdk.Serial, arg0: int, arg1: str) -> int
00281     *
00282     * @par Lua 函数原型
00283     * serialSendInt(value: number, serial_name: string) -> nil
00284     * \endchinese
00285     * \english
00286     * Sends an int (int32_t) to the server
00287     * Sends the int <value> through the serial. Send in network byte order.
00288     * Expects no response.
00289     *
00290     * @param value

```

```

00291     * @param serial_name
00292     * @return
00293     *
00294     * @par Python function prototype
00295     * serialSendInt(self: pyaubo_sdk.Serial, arg0: int, arg1: str) -> int
00296     *
00297     * @par Lua function prototype
00298     * serialSendInt(value: number, serial_name: string) -> nil
00299     * \endenglish
00300     */
00301 int serialSendInt(int value, const std::string &serial_name = "serial_0");
00302
00303 /**
00304     * @ingroup Serial
00305     * \chinese
00306     * 发送带有换行符的字符串到服务器
00307     * 以 ASCII 编码通过串口发送字符串 <str>, 并在末尾添加换行符。不期望有响应。
00308     *
00309     * @param str 字符串
00310     * @param serial_name 串口名称
00311     * @return 返回值
00312     *
00313     * @par Python 函数原型
00314     * serialSendLine(self: pyaubo_sdk.Serial, arg0: str, arg1: str) -> int
00315     *
00316     * @par Lua 函数原型
00317     * serialSendLine(str: string, serial_name: string) -> nil
00318     * \endchinese
00319     * \english
00320     * Sends a string with a newline character to the server
00321     * Sends the string <str> through the serial in ASCII coding, appending a newline at the end. Expects no
response.
00322     *
00323     * @param str
00324     * @param serial_name
00325     * @return
00326     *
00327     * @par Python function prototype
00328     * serialSendLine(self: pyaubo_sdk.Serial, arg0: str, arg1: str) -> int
00329     *
00330     * @par Lua function prototype
00331     * serialSendLine(str: string, serial_name: string) -> nil
00332     * \endenglish
00333     */
00334 int serialSendLine(const std::string &str,
00335                   const std::string &serial_name = "serial_0");
00336
00337 /**
00338     * @ingroup Serial
00339     * \chinese
00340     * 发送字符串到服务器
00341     * 以 ASCII 编码通过串口发送字符串 <str>。不期望有响应。
00342     *
00343     * @param str 字符串
00344     * @param serial_name 串口名称
00345     * @return 返回值
00346     *
00347     * @par Python 函数原型
00348     * serialSendString(self: pyaubo_sdk.Serial, arg0: str, arg1: str) -> int
00349     *
00350     * @par Lua 函数原型
00351     * serialSendString(str: string, serial_name: string) -> nil
00352     * \endchinese
00353     * \english
00354     * Sends a string to the server
00355     * Sends the string <str> through the serial in ASCII coding. Expects no
00356     * response.
00357     *
00358     * @param str
00359     * @param serial_name
00360     * @return
00361     *
00362     * @par Python function prototype
00363     * serialSendString(self: pyaubo_sdk.Serial, arg0: str, arg1: str) -> int
00364     *
00365     * @par Lua function prototype
00366     * serialSendString(str: string, serial_name: string) -> nil
00367     * \endenglish
00368     */
00369 int serialSendString(const std::string &str,
00370                    const std::string &serial_name = "serial_0");
00371
00372 /**
00373     * @ingroup Serial
00374     * \chinese
00375     *
00376     * @param is_check 是否校验

```

```

00377      * @param str 字符串数组
00378      * @param serial_name 串口名称
00379      * @return 返回值
00380      *
00381      * @par Python 函数原型
00382      * serialSendAllString(self: pyaubo_sdk.Serial, arg0: bool, arg1: List[str], arg2: str) -> int
00383      *
00384      * @par Lua 函数原型
00385      * serialSendAllString(is_check: boolean, str: table, serial_name: string) -> nil
00386      * \endchinese
00387      * \english
00388      *
00389      * @param is_check Whether to check
00390      * @param str Array of strings
00391      * @param serial_name Serial port name
00392      * @return Return value
00393      *
00394      * @par Python function prototype
00395      * serialSendAllString(self: pyaubo_sdk.Serial, arg0: bool, arg1: List[str], arg2: str) -> int
00396      *
00397      * @par Lua function prototype
00398      * serialSendAllString(is_check: boolean, str: table, serial_name: string) -> nil
00399      * \endenglish
00400      */
00401      int serialSendAllString(bool is_check, const std::vector<char> &str,
00402                             const std::string &serial_name = "serial_0");
00403
00404 protected:
00405     void *d_;
00406 };
00407 using SerialPtr = std::shared_ptr<Serial>;
00408
00409 } // namespace common_interface
00410 } // namespace arcs
00411 #endif

```

12.40 include/aubo/socket.h 文件参考

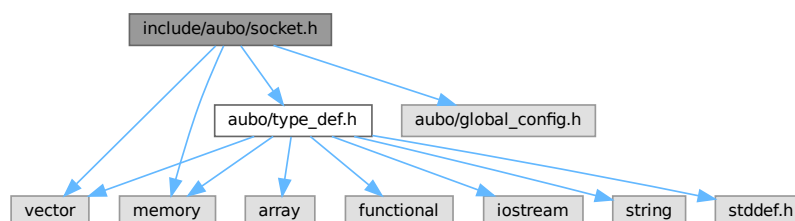
socket 通信

```

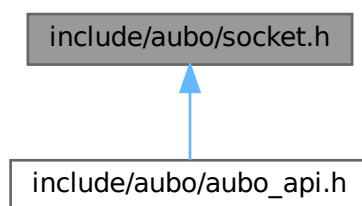
#include <vector>
#include <memory>
#include <aubo/type_def.h>
#include <aubo/global_config.h>

```

socket.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::common_interface::Socket](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::common_interface](#)

类型定义

- using [arcs::common_interface::SocketPtr](#) = std::shared_ptr<[Socket](#)>

12.40.1 详细描述

socket 通信

在文件 [socket.h](#) 中定义.

12.41 socket.h

[浏览该文件的文档.](#)

```

00001 /** @file socket.h
00002  * @brief socket 通信
00003  */
00004 #ifndef AUBO_SDK_SOCKET_INTERFACE_H
00005 #define AUBO_SDK_SOCKET_INTERFACE_H
00006
00007 #include <vector>
00008 #include <memory>
00009
00010 #include <aubo/type_def.h>
00011 #include <aubo/global_config.h>
00012
00013 namespace arcs {
00014 namespace common_interface {
00015
00016
00017 /**
00018  * @defgroup Socket Socket (socket 网络通信)
00019  * \~chinese 串口通信
00020  * \~english Socket control
00021  */
00022 class ARCS_ABI_EXPORT Socket
00023 {
00024 public:
00025     Socket();
00026     virtual ~Socket();
00027
00028     /**
00029      * @ingroup Socket
00030      * \english
00031      * Open TCP/IP ethernet communication socket
  
```

```

00032      *
00033      * Instruction
00034      *
00035      * @param address
00036      * @param port
00037      * @param socket_name
00038      * @return
00039      *
00040      * @par Python function prototype
00041      * socketOpen(self: pyaubo_sdk.Socket, arg0: str, arg1: int, arg2: str) -> int
00042      *
00043      * @par Lua function prototype
00044      * socketOpen(address: string, port: number, socket_name: string) -> nil
00045      *
00046      * @par JSON-RPC request example
00047      * {"jsonrpc": "2.0", "method": "Socket.socketOpen", "params": ["172.16.26.248", 8000, "socket_0"], "id": 1}
00048      *
00049      * @par JSON-RPC response example
00050      * {"id": 1, "jsonrpc": "2.0", "result": 0}
00051      * \endenglish
00052      * \chinese
00053      * 打开 TCP/IP 以太网通信 socket
00054      *
00055      * 指令
00056      *
00057      * @param address 地址
00058      * @param port 端口
00059      * @param socket_name 套接字名称
00060      * @return 返回值
00061      *
00062      * @par Python 函数原型
00063      * socketOpen(self: pyaubo_sdk.Socket, arg0: str, arg1: int, arg2: str) -> int
00064      *
00065      * @par Lua 函数原型
00066      * socketOpen(address: string, port: number, socket_name: string) -> nil
00067      *
00068      * @par JSON-RPC 请求示例
00069      * {"jsonrpc": "2.0", "method": "Socket.socketOpen", "params": ["172.16.26.248", 8000, "socket_0"], "id": 1}
00070      *
00071      * @par JSON-RPC 响应示例
00072      * {"id": 1, "jsonrpc": "2.0", "result": 0}
00073      * \endchinese
00074      */
00075      int socketOpen(const std::string &address, int port,
00076                    const std::string &socket_name = "socket_0");
00077
00078      /**
00079      * @ingroup Socket
00080      * \english
00081      * Closes TCP/IP socket communication
00082      * Closes down the socket connection to the server.
00083      *
00084      * Instruction
00085      *
00086      * @param socket_name
00087      * @return
00088      *
00089      * @par Python function prototype
00090      * socketClose(self: pyaubo_sdk.Socket, arg0: str) -> int
00091      *
00092      * @par Lua function prototype
00093      * socketClose(socket_name: string) -> nil
00094      *
00095      * @par JSON-RPC request example
00096      * {"jsonrpc": "2.0", "method": "Socket.socketClose", "params": ["socket_0"], "id": 1}
00097      *
00098      * @par JSON-RPC response example
00099      * {"id": 1, "jsonrpc": "2.0", "result": 0}
00100      * \endenglish
00101      * \chinese
00102      * 关闭 TCP/IP socket 通信
00103      * 关闭与服务器的 socket 连接。
00104      *
00105      * 指令
00106      *
00107      * @param socket_name 套接字名称
00108      * @return 返回值
00109      *
00110      * @par Python 函数原型
00111      * socketClose(self: pyaubo_sdk.Socket, arg0: str) -> int
00112      *
00113      * @par Lua 函数原型
00114      * socketClose(socket_name: string) -> nil
00115      *
00116      * @par JSON-RPC 请求示例
00117      * {"jsonrpc": "2.0", "method": "Socket.socketClose", "params": ["socket_0"], "id": 1}
00118      *

```

```

00119     * @par JSON-RPC 响应示例
00120     * {"id":1,"jsonrpc":"2.0","result":0}
00121     * \endchinese
00122     */
00123 int socketClose(const std::string &socket_name = "socket_0");
00124
00125 /**
00126  * @ingroup Socket
00127  * \english
00128  * Reads a number of ascii formatted floats from the socket. A maximum
00129  * of 30 values can be read in one command.
00130  * A list of numbers read (list of floats, length=number+1)
00131  *
00132  * Result will be stored in a register named reg_key. Use getFloatVec
00133  * to retrieve data
00134  *
00135  * @param number
00136  * @param variable
00137  * @param socket_name
00138  * @return
00139  *
00140  * @par Python function prototype
00141  * socketReadAsciiFloat(self: pyaubo_sdk.Socket, arg0: int, arg1: str, arg2:
00142  * str) -> int
00143  *
00144  * @par Lua function prototype
00145  * socketReadAsciiFloat(number: number, variable: string, socket_name:
00146  * string) -> number
00147  * \endenglish
00148  * \chinese
00149  * 从 socket 读取指定数量的 ASCII 格式浮点数。一次最多可读取 30 个值。
00150  * 读取到的数字列表（浮点数列表，长度 =number+1）
00151  *
00152  * 结果将存储在名为 reg_key 的寄存器中。使用 getFloatVec 获取数据
00153  *
00154  * @param number 数量
00155  * @param variable 变量名
00156  * @param socket_name 套接字名称
00157  * @return 返回值
00158  *
00159  * @par Python 函数原型
00160  * socketReadAsciiFloat(self: pyaubo_sdk.Socket, arg0: int, arg1: str, arg2:
00161  * str) -> int
00162  *
00163  * @par Lua 函数原型
00164  * socketReadAsciiFloat(number: number, variable: string, socket_name:
00165  * string) -> number
00166  * \endchinese
00167  */
00168 int socketReadAsciiFloat(int number, const std::string &variable,
00169                          const std::string &socket_name = "socket_0");
00170
00171 /**
00172  * @ingroup Socket
00173  * \english
00174  * Reads a number of 32 bit integers from the socket. Bytes are in
00175  * network byte order. A maximum of 30 values can be read in one
00176  * command.
00177  * A list of numbers read (list of ints, length=number+1)
00178  *
00179  * Instruction
00180  *
00181  * std::vector<int>
00182  *
00183  * @param number
00184  * @param variable
00185  * @param socket_name
00186  * @return
00187  *
00188  * @par Python function prototype
00189  * socketReadBinaryInteger(self: pyaubo_sdk.Socket, arg0: int, arg1: str,
00190  * arg2: str) -> int
00191  *
00192  * @par Lua function prototype
00193  * socketReadBinaryInteger(number: number, variable: string, socket_name:
00194  * string) -> number
00195  * \endenglish
00196  * \chinese
00197  * 从 socket 读取指定数量的 32 位整数。字节为网络字节序。一次最多可读取 30 个值。
00198  * 读取到的数字列表（整数列表，长度 =number+1）
00199  *
00200  * 指令
00201  *
00202  * std::vector<int>
00203  *
00204  * @param number 数量
00205  * @param variable 变量名

```



```

00206     * @param socket_name 套接字名称
00207     * @return 返回值
00208     *
00209     * @par Python 函数原型
00210     * socketReadBinaryInteger(self: pyaubo_sdk.Socket, arg0: int, arg1: str,
00211     * arg2: str) -> int
00212     *
00213     * @par Lua 函数原型
00214     * socketReadBinaryInteger(number: number, variable: string, socket_name:
00215     * string) -> number
00216     * \endchinese
00217     */
00218 int socketReadBinaryInteger(int number, const std::string &variable,
00219                             const std::string &socket_name = "socket_0");
00220
00221 /**
00222  * @ingroup Socket
00223  * \english
00224  * Reads a number of bytes from the socket. Bytes are in network byte
00225  * order. A maximum of 30 values can be read in one command.
00226  * A list of numbers read (list of ints, length=number+1)
00227  *
00228  * Instruction
00229  *
00230  * std::vector<char>
00231  *
00232  * @param number
00233  * @param variable
00234  * @param socket_name
00235  * @return
00236  *
00237  * @par Python function prototype
00238  * socketReadByteList(self: pyaubo_sdk.Socket, arg0: int, arg1: str, arg2:
00239  * str) -> int
00240  *
00241  * @par Lua function prototype
00242  * socketReadByteList(number: number, variable: string, socket_name: string)
00243  * -> number
00244  * \endenglish
00245  * \chinese
00246  * 从 socket 读取指定数量的字节。字节为网络字节序。一次最多可读取 30 个值。
00247  * 读取到的数字列表（整数列表，长度 =number+1）
00248  *
00249  * 指令
00250  *
00251  * std::vector<char>
00252  *
00253  * @param number 数量
00254  * @param variable 变量名
00255  * @param socket_name 套接字名称
00256  * @return 返回值
00257  *
00258  * @par Python 函数原型
00259  * socketReadByteList(self: pyaubo_sdk.Socket, arg0: int, arg1: str, arg2:
00260  * str) -> int
00261  *
00262  * @par Lua 函数原型
00263  * socketReadByteList(number: number, variable: string, socket_name: string)
00264  * -> number
00265  * \endchinese
00266  */
00267 int socketReadByteList(int number, const std::string &variable,
00268                        const std::string &socket_name = "socket_0");
00269
00270 /**
00271  * @ingroup Socket
00272  * \english
00273  * Reads all data from the socket and returns the data as a string.
00274  * Bytes are in network byte order.
00275  *
00276  * The optional parameters "prefix" and "suffix", can be used to express
00277  * what is extracted from the socket. The "prefix" specifies the start
00278  * of the substring (message) extracted from the socket. The data up to
00279  * the end of the "prefix" will be ignored and removed from the socket.
00280  * The "suffix" specifies the end of the substring (message) extracted
00281  * from the socket. Any remaining data on the socket, after the "suffix",
00282  * will be preserved. E.g. if the socket server sends a string
00283  * "noise>hello<", the controller can receive the "hello" by calling this
00284  * script function with the prefix=">" and suffix="<". By using the
00285  * "prefix" and "suffix" it is also possible send multiple string to the
00286  * controller at once, because the suffix defines where the message ends.
00287  * E.g. sending ">hello<>world<"
00288  *
00289  * Instruction
00290  *
00291  * std::string
00292  *

```

```

00293     * @param variable
00294     * @param socket_name
00295     * @param prefix
00296     * @param suffix
00297     * @param interpret_escape
00298     * @return
00299     *
00300     * @par Python function prototype
00301     * socketReadString(self: pyaubo_sdk.Socket, arg0: str, arg1: str, arg2:
00302     * str, arg3: str, arg4: bool) -> int
00303     *
00304     * @par Lua function prototype
00305     * socketReadString(variable: string, socket_name: string, prefix: string,
00306     * suffix: string, interpret_escape: boolean) -> number
00307     *
00308     * @par JSON-RPC request example
00309     * {"jsonrpc": "2.0", "method": "Socket.socketReadString", "params": ["camera", "socket_0", "", "", false], "id": 1}
00310     *
00311     * @par JSON-RPC response example
00312     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00313     * \endenglish
00314     * \chinese
00315     * 从 socket 读取所有数据并将其作为字符串返回。
00316     * 字节为网络字节序。
00317     *
00318     * 可选参数"prefix" 和"suffix" 可用于指定从 socket 中提取的内容。
00319     * "prefix" 指定提取子字符串（消息）的起始位置。直到"prefix" 结尾的数据将被忽略并从 socket 中移除。
00320     * "suffix" 指定提取子字符串（消息）的结束位置。"suffix" 之后的任何剩余数据将保留在 socket 中。
00321     * 例如，如果 socket 服务器发送字符串"noise>hello<", 控制器可以通过调用此脚本函数并设置 prefix=">" 和 suffix="<" 来
接收"hello"。
00322     * 通过使用"prefix" 和"suffix", 还可以一次向控制器发送多条字符串，因为"suffix" 定义了消息的结束位置。例如发
送">hello<>world<"
00323     *
00324     * 指令
00325     *
00326     * std::string
00327     *
00328     * @param variable 变量名
00329     * @param socket_name 套接字名称
00330     * @param prefix 前缀
00331     * @param suffix 后缀
00332     * @param interpret_escape 是否解释转义字符
00333     * @return 返回值
00334     *
00335     * @par Python 函数原型
00336     * socketReadString(self: pyaubo_sdk.Socket, arg0: str, arg1: str, arg2:
00337     * str, arg3: str, arg4: bool) -> int
00338     *
00339     * @par Lua 函数原型
00340     * socketReadString(variable: string, socket_name: string, prefix: string,
00341     * suffix: string, interpret_escape: boolean) -> number
00342     *
00343     * @par JSON-RPC 请求示例
00344     * {"jsonrpc": "2.0", "method": "Socket.socketReadString", "params": ["camera", "socket_0", "", "", false], "id": 1}
00345     *
00346     * @par JSON-RPC 响应示例
00347     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00348     * \endchinese
00349     */
00350 int socketReadString(const std::string &variable,
00351                     const std::string &socket_name = "socket_0",
00352                     const std::string &prefix = "",
00353                     const std::string &suffix = "",
00354                     bool interpret_escape = false);
00355
00356 /**
00357  * @ingroup Socket
00358  * \english
00359  * Reads all data from the socket and returns the data as a vector of chars.
00360  *
00361  * Instruction
00362  * std::vector<char>
00363  *
00364  * @param variable
00365  * @param socket_name
00366  * @return
00367  *
00368  * @par Python function prototype
00369  * socketReadAllString(self: pyaubo_sdk.Socket, arg0: str, arg1: str) -> int
00370  *
00371  * @par Lua function prototype
00372  * socketReadAllString(variable: string, socket_name: string) -> number
00373  *
00374  * @par JSON-RPC request example
00375  * {"jsonrpc": "2.0", "method": "Socket.socketReadAllString", "params": ["camera", "socket_0"], "id": 1}
00376  *
00377  * @par JSON-RPC response example

```

```

00378     * {"id":1,"jsonrpc":"2.0","result":0}
00379     * \endenglish
00380     * \chinese
00381     * 从 socket 读取所有数据并将其作为 char 向量返回。
00382     *
00383     * 指令
00384     * std::vector<char>
00385     *
00386     * @param variable 变量名
00387     * @param socket_name 套接字名称
00388     * @return 返回值
00389     *
00390     * @par Python 函数原型
00391     * socketReadAllString(self: pyaubo_sdk.Socket, arg0: str, arg1: str) -> int
00392     *
00393     * @par Lua 函数原型
00394     * socketReadAllString(variable: string, socket_name: string) -> number
00395     *
00396     * @par JSON-RPC 请求示例
00397     * {"jsonrpc":"2.0","method":"Socket.socketReadAllString","params":["camera","socket_0"],"id":1}
00398     *
00399     * @par JSON-RPC 响应示例
00400     * {"id":1,"jsonrpc":"2.0","result":0}
00401     * \endchinese
00402     */
00403 int socketReadAllString(const std::string &variable,
00404                        const std::string &socket_name = "socket_0");
00405
00406 /**
00407  * @ingroup Socket
00408  * \english
00409  * Sends a byte to the server
00410  * Sends the byte <value> through the socket. Expects no response. Can
00411  * be used to send special ASCII characters; 10 is newline, 2 is start of
00412  * text, 3 is end of text.
00413  *
00414  * Instruction
00415  *
00416  * @param value
00417  * @param socket_name
00418  * @return
00419  *
00420  * @par Python function prototype
00421  * socketSendByte(self: pyaubo_sdk.Socket, arg0: str, arg1: str) -> int
00422  *
00423  * @par Lua function prototype
00424  * socketSendByte(value: string, socket_name: string) -> nil
00425  *
00426  * \endenglish
00427  * \chinese
00428  * 发送一个字节到服务器
00429  * 通过 socket 发送字节 <value>, 不期望响应。可用于发送特殊 ASCII 字符; 10 为换行符, 2 为文本开始, 3 为文本结束。
00430  *
00431  * 指令
00432  *
00433  * @param value 字节值
00434  * @param socket_name 套接字名称
00435  * @return 返回值
00436  *
00437  * @par Python 函数原型
00438  * socketSendByte(self: pyaubo_sdk.Socket, arg0: str, arg1: str) -> int
00439  *
00440  * @par Lua 函数原型
00441  * socketSendByte(value: string, socket_name: string) -> nil
00442  *
00443  * \endchinese
00444  */
00445 int socketSendByte(char value, const std::string &socket_name = "socket_0");
00446
00447 /**
00448  * @ingroup Socket
00449  * \english
00450  * Sends an int (int32_t) to the server
00451  * Sends the int <value> through the socket. Send in network byte order.
00452  * Expects no response
00453  *
00454  * Instruction
00455  *
00456  * @param value
00457  * @param socket_name
00458  * @return
00459  *
00460  * @par Python function prototype
00461  * socketSendInt(self: pyaubo_sdk.Socket, arg0: int, arg1: str) -> int
00462  *
00463  * @par Lua function prototype
00464  * socketSendInt(value: number, socket_name: string) -> nil

```

```

00465      *
00466      * \endenglish
00467      * \chinese
00468      * 发送一个 int (int32_t) 到服务器
00469      * 通过 socket 发送 int <value>, 以网络字节序发送。不期望响应。
00470      *
00471      * 指令
00472      *
00473      * @param value 整数值
00474      * @param socket_name 套接字名称
00475      * @return 返回值
00476      *
00477      * @par Python 函数原型
00478      * socketSendInt(self: pyaubo_sdk.Socket, arg0: int, arg1: str) -> int
00479      *
00480      * @par Lua 函数原型
00481      * socketSendInt(value: number, socket_name: string) -> nil
00482      *
00483      * \endchinese
00484      */
00485 int socketSendInt(int value, const std::string &socket_name = "socket_0");
00486
00487 /**
00488  * @ingroup Socket
00489  * \english
00490  * Sends a string with a newline character to the server
00491  * Sends the string <str> through the socket in ASCII coding. Expects no
00492  * response.
00493  *
00494  * Instruction
00495  *
00496  * @param str
00497  * @param socket_name
00498  * @return
00499  *
00500  * @par Python function prototype
00501  * socketSendLine(self: pyaubo_sdk.Socket, arg0: str, arg1: str) -> int
00502  *
00503  * @par Lua function prototype
00504  * socketSendLine(str: string, socket_name: string) -> nil
00505  *
00506  * @par JSON-RPC request example
00507  * {"jsonrpc": "2.0", "method": "Socket.socketSendLine", "params": ["abcd", "socket_0"], "id": 1}
00508  *
00509  * @par JSON-RPC response example
00510  * {"id": 1, "jsonrpc": "2.0", "result": 0}
00511  * \endenglish
00512  * \chinese
00513  * 发送带有换行符的字符串到服务器。
00514  * 通过 socket 以 ASCII 编码发送字符串 <str>, 不期望响应。
00515  *
00516  * 指令
00517  *
00518  * @param str 字符串
00519  * @param socket_name 套接字名称
00520  * @return 返回值
00521  *
00522  * @par Python 函数原型
00523  * socketSendLine(self: pyaubo_sdk.Socket, arg0: str, arg1: str) -> int
00524  *
00525  * @par Lua 函数原型
00526  * socketSendLine(str: string, socket_name: string) -> nil
00527  *
00528  * @par JSON-RPC 请求示例
00529  * {"jsonrpc": "2.0", "method": "Socket.socketSendLine", "params": ["abcd", "socket_0"], "id": 1}
00530  *
00531  * @par JSON-RPC 响应示例
00532  * {"id": 1, "jsonrpc": "2.0", "result": 0}
00533  * \endchinese
00534  */
00535 int socketSendLine(const std::string &str,
00536                   const std::string &socket_name = "socket_0");
00537
00538 /**
00539  * @ingroup Socket
00540  * \english
00541  * Sends a string to the server
00542  * Sends the string <str> through the socket in ASCII coding. Expects no
00543  * response.
00544  *
00545  * Instruction
00546  *
00547  * @param str
00548  * @param socket_name
00549  * @return
00550  *
00551  * @par Python function prototype

```

```

00552     * socketSendString(self: pyaubo_sdk.Socket, arg0: str, arg1: str) -> int
00553     *
00554     * @par Lua function prototype
00555     * socketSendString(str: string, socket_name: string) -> nil
00556     *
00557     * @par JSON-RPC request example
00558     * {"jsonrpc": "2.0", "method": "Socket.socketSendString", "params": ["abcd", "socket_0"], "id": 1}
00559     *
00560     * @par JSON-RPC response example
00561     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00562     * \endenglish
00563     * \chinese
00564     * 发送字符串到服务器
00565     * 通过 socket 以 ASCII 编码发送字符串 <str>, 不期望响应。
00566     *
00567     * 指令
00568     *
00569     * @param str 字符串
00570     * @param socket_name 套接字名称
00571     * @return 返回值
00572     *
00573     * @par Python 函数原型
00574     * socketSendString(self: pyaubo_sdk.Socket, arg0: str, arg1: str) -> int
00575     *
00576     * @par Lua 函数原型
00577     * socketSendString(str: string, socket_name: string) -> nil
00578     *
00579     * @par JSON-RPC 请求示例
00580     * {"jsonrpc": "2.0", "method": "Socket.socketSendString", "params": ["abcd", "socket_0"], "id": 1}
00581     *
00582     * @par JSON-RPC 响应示例
00583     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00584     * \endchinese
00585     */
00586 int socketSendString(const std::string &str,
00587                     const std::string &socket_name = "socket_0");
00588
00589 /**
00590  * @ingroup Socket
00591  * \english
00592  * Sends all data in the given vector of chars to the server.
00593  *
00594  * @param is_check Whether to check the sending status
00595  * @param str The data to send as a vector of chars
00596  * @param socket_name The name of the socket
00597  * @return Status code
00598  *
00599  * @par Python function prototype
00600  * socketSendAllString(self: pyaubo_sdk.Socket, arg0: bool, arg1: List[str], arg2: str) -> int
00601  *
00602  * @par Lua function prototype
00603  * socketSendAllString(is_check: boolean, str: table, socket_name: string) -> nil
00604  * \endenglish
00605  * \chinese
00606  * 发送给定 char 向量中的所有数据到服务器。
00607  *
00608  * @param is_check 是否检查发送状态
00609  * @param str 要发送的数据, char 向量
00610  * @param socket_name 套接字名称
00611  * @return 状态码
00612  *
00613  * @par Python 函数原型
00614  * socketSendAllString(self: pyaubo_sdk.Socket, arg0: bool, arg1: List[str], arg2: str) -> int
00615  *
00616  * @par Lua 函数原型
00617  * socketSendAllString(is_check: boolean, str: table, socket_name: string) -> nil
00618  * \endchinese
00619  */
00620 int socketSendAllString(bool is_check, const std::vector<char> &str,
00621                        const std::string &socket_name = "socket_0");
00622
00623 /**
00624  * @ingroup Socket
00625  * \-chinese 检测 socket 连接是否成功 \-english Check if the socket is connected
00626  * @brief socketHasConnected
00627  * @param socket_name
00628  *
00629  * @return
00630  *
00631  * @par Python 函数原型
00632  * socketHasConnected(self: pyaubo_sdk.Socket, arg0: str) -> bool
00633  *
00634  * @par Lua 函数原型
00635  * socketHasConnected(socket_name: string) -> boolean
00636  */
00637 bool socketHasConnected(const std::string &socket_name = "socket_0");
00638

```

```

00639 protected:
00640     void *d_;
00641 };
00642 using SocketPtr = std::shared_ptr<Socket>;
00643
00644 } // namespace common_interface
00645 } // namespace arcs
00646 #endif

```

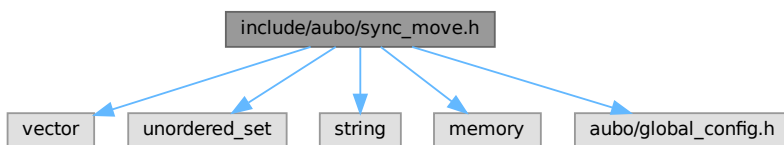
12.42 include/aubo/sync_move.h 文件参考

同步运行

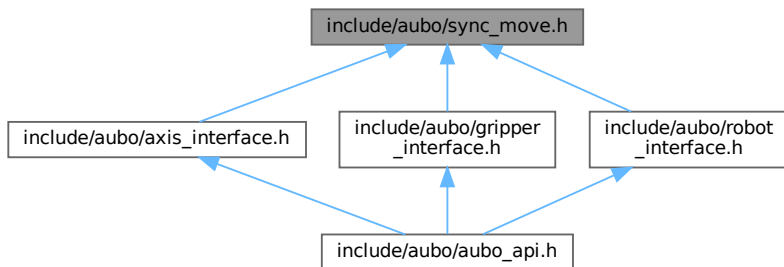
```

#include <vector>
#include <unordered_set>
#include <string>
#include <memory>
#include <aubo/global_config.h>
sync_move.h 的引用 (Include) 关系图:

```



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::common_interface::SyncMove](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::common_interface](#)

类型定义

- typedef `std::unordered_set< std::string >` [arcs::common_interface::TaskSet](#)
- using [arcs::common_interface::SyncMovePtr](#) = `std::shared_ptr<SyncMove>`

12.42.1 详细描述

同步运行

1. Independent movements If the different task programs, and their robots, work independently, no synchronization or coordination is needed. Each task program is then written as if it was the program for a single robot system.
2. Semi coordinated movements Several robots can work with the same work object, without synchronized movements, as long as the work object is not moving. A positioner can move the work object when the robots are not coordinated to it, and the robots can be coordinated to the work object when it is not moving. Switching between moving the object and coordinating the robots is called semi coordinated movements.
3. Coordinated synchronized movements Several robots can work with the same moving work object. The positioner or robot that holds the work object and the robots that work with the work object must have synchronized movements. This means that the RAPID task programs, that handle one mechanical unit each, execute their move instructions simultaneously.

在文件 [sync_move.h](#) 中定义.

12.43 sync_move.h

[浏览该文件的文档.](#)

```
00001 /** @file sync_move.h
00002  * @brief 同步运行
00003  *
00004  * 1. Independent movements
00005  * If the different task programs, and their robots, work independently, no
00006  * synchronization or coordination is needed. Each task program is then
00007  * written as if it was the program for a single robot system.
00008  *
00009  * 2. Semi coordinated movements
00010  * Several robots can work with the same work object, without synchronized
00011  * movements, as long as the work object is not moving.
00012  * A positioner can move the work object when the robots are not coordinated
00013  * to it, and the robots can be coordinated to the work object when it is not
00014  * moving. Switching between moving the object and coordinating the robots is
00015  * called semi coordinated movements.
00016  *
00017  * 3. Coordinated synchronized movements
00018  * Several robots can work with the same moving work object.
00019  * The positioner or robot that holds the work object and the robots that work
00020  * with the work object must have synchronized movements. This means that the
00021  * RAPID task programs, that handle one mechanical unit each, execute their
00022  * move instructions simultaneously.
00023  */
00024 #ifndef AUBO_SDK_SYNC_MOVE_INTERFACE_H
00025 #define AUBO_SDK_SYNC_MOVE_INTERFACE_H
00026
00027 #include <vector>
00028 #include <unordered_set>
00029 #include <string>
00030 #include <memory>
00031 #include <aubo/global_config.h>
00032
00033 namespace arcs {
00034 namespace common_interface {
00035
00036 typedef std::unordered_set<std::string> TaskSet;
00037
00038 /**
00039  * @defgroup SyncMove SyncMove (同步运动控制和轴组管理)
00040  * \-chinese 同步运动控制和轴组管理 API 接口
00041  */
00042 class ARCS_ABI_EXPORT SyncMove
00043 {
00044 public:
00045     SyncMove();
00046     virtual ~SyncMove();
00047
00048     /**
00049      * @ingroup SyncMove
00050      * \-chinese
00051      * syncMoveOn 用于启动同步运动模式。
00052      *
00053      * syncMoveOn 指令会等待其他任务程序。当所有任务程序都到达 syncMoveOn 时，
```

```

00054      * 它们将继续以同步运动模式执行。不同任务程序中的移动指令将同时执行，
00055      * 直到执行 syncMoveOff 指令为止。在 syncMoveOn 指令之前必须编程一个停止点。
00056      *
00057      * @param syncident
00058      * @param taskset
00059      * @return
00060      *
00061      * @par Python 函数原型
00062      * syncMoveOn(self: pyaubo_sdk.SyncMove, arg0: str, arg1: Set[str]) -> int
00063      *
00064      * @par Lua 函数原型
00065      * syncMoveOn(syncident: string, taskset: table) -> nil
00066      * @endcoe
00067      * \endchinese
00068      * \english
00069      * syncMoveOn is used to start synchronized movement mode.
00070      *
00071      * A syncMoveOn instruction will wait for the other task programs. When
00072      * all task programs have reached the syncMoveOn, they will continue
00073      * their execution in synchronized movement mode. The move instructions
00074      * in the different task programs are executed simultaneously, until the
00075      * instruction syncMoveOff is executed.
00076      * A stop point must be programmed before the syncMoveOn instruction.
00077      *
00078      * @param syncident
00079      * @param taskset
00080      * @return
00081      *
00082      * @par Python function prototype
00083      * syncMoveOn(self: pyaubo_sdk.SyncMove, arg0: str, arg1: Set[str]) -> int
00084      *
00085      * @par Lua function prototype
00086      * syncMoveOn(syncident: string, taskset: table) -> nil
00087      * @endcoe
00088      * \endenglish
00089      */
00090      int syncMoveOn(const std::string &syncident, const TaskSet &taskset);
00091
00092      /**
00093      * @ingroup SyncMove
00094      * \chinese
00095      * 设置同步路径段的 ID
00096      * 在同步运动模式下，所有同时执行的移动指令必须全部编程为圆角区（corner zones）或全部为停止点（stop points）。
00097      * 这意味着具有相同 ID 的移动指令要么全部带有圆角区，要么全部带有停止点。
00098      * 如果在各自的程序任务中同步执行的移动指令中，一个带有圆角区而另一个带有停止点，则会发生错误。
00099      * 同步执行的移动指令可以有不同大小的圆角区（例如，一个使用 z10，另一个使用 z50）。
00100      *
00101      * @param id
00102      * @return
00103      *
00104      * @par Python 函数原型
00105      * syncMoveSegment(self: pyaubo_sdk.SyncMove, arg0: int) -> bool
00106      *
00107      * @par Lua 函数原型
00108      * syncMoveSegment(id: number) -> boolean
00109      * @endcoe
00110      * \endchinese
00111      * \english
00112      * Set the ID for the synchronized path segment.
00113      * In synchronized movements mode, all or none of the simultaneous move instructions must be programmed with
00114      * corner zones.
00115      * This means that move instructions with the same ID must either all have corner zones, or all have stop
00116      * points.
00117      * If a move instruction with a corner zone and a move instruction with a stop point are synchronously executed
00118      * in their respective task program, an error will occur.
00119      * Synchronously executed move instructions can have corner zones of different sizes (e.g. one uses z10 and one
00120      * uses z50).
00121      *
00122      * @param id
00123      * @return
00124      *
00125      * @par Python prototype
00126      * syncMoveSegment(self: pyaubo_sdk.SyncMove, arg0: int) -> bool
00127      *
00128      * @par Lua prototype
00129      * syncMoveSegment(id: number) -> boolean
00130      * @endcoe
00131      * \endenglish
00132      */
00133      bool syncMoveSegment(int id);
00134
00135      /**
00136      * @ingroup SyncMove
00137      * \chinese
00138      * syncMoveOff 用于结束同步运动模式。
00139      *
00140      * syncMoveOff 指令会等待其他任务程序。当所有任务程序都到达 syncMoveOff 时，

```



```

00137      * 它们将继续以非同步模式执行。在 syncMoveOff 指令之前必须编程一个停止点。
00138      *
00139      * @param syncident
00140      * @return
00141      *
00142      * @par Python 函数原型
00143      * syncMoveOff(self: pyaubo_sdk.SyncMove, arg0: str) -> int
00144      *
00145      * @par Lua 函数原型
00146      * syncMoveOff(syncident: string) -> nil
00147      * @endcode
00148      * \endchinese
00149      * \english
00150      * syncMoveOff is used to end synchronized movement mode.
00151      *
00152      * A syncMoveOff instruction will wait for the other task programs. When
00153      * all task programs have reached the syncMoveOff, they will continue
00154      * their execution in unsynchronized mode.
00155      * A stop point must be programmed before the syncMoveOff instruction.
00156      *
00157      * @param syncident
00158      * @return
00159      *
00160      * @par Python prototype
00161      * syncMoveOff(self: pyaubo_sdk.SyncMove, arg0: str) -> int
00162      *
00163      * @par Lua prototype
00164      * syncMoveOff(syncident: string) -> nil
00165      * @endcode
00166      * \endenglish
00167      */
00168      int syncMoveOff(const std::string &syncident);
00169
00170      /**
00171      * @ingroup SyncMove
00172      * \chinese
00173      * syncMoveUndo 用于关闭同步运动，即使不是所有其他任务程序都执行了 syncMoveUndo 指令。
00174      *
00175      * syncMoveUndo 主要用于 UNDO 处理程序。当程序指针从过程移动时，syncMoveUndo 用于关闭同步。
00176      *
00177      * @return
00178      *
00179      * @par Python 函数原型
00180      * syncMoveUndo(self: pyaubo_sdk.SyncMove) -> int
00181      *
00182      * @par Lua 函数原型
00183      * syncMoveUndo() -> nil
00184      * @endcode
00185      * \endchinese
00186      * \english
00187      * syncMoveUndo is used to turn off synchronized movements, even if not
00188      * all the other task programs execute the syncMoveUndo instruction.
00189      *
00190      * syncMoveUndo is intended for UNDO handlers. When the program
00191      * pointer is moved from the procedure, syncMoveUndo is used to turn off
00192      * the synchronization.
00193      *
00194      * @return
00195      *
00196      * @par Python prototype
00197      * syncMoveUndo(self: pyaubo_sdk.SyncMove) -> int
00198      *
00199      * @par Lua prototype
00200      * syncMoveUndo() -> nil
00201      * @endcode
00202      * \endenglish
00203      */
00204      int syncMoveUndo();
00205
00206      /**
00207      * @ingroup SyncMove
00208      * \chinese
00209      * waitSyncTasks 用于在程序中的特定点同步多个任务程序。
00210      *
00211      * waitSyncTasks 指令会等待其他任务程序。当所有任务程序都到达 waitSyncTasks 指令时，
00212      * 它们将继续执行。
00213      *
00214      * @param syncident
00215      * @param taskset
00216      * @return
00217      *
00218      * @par Python 函数原型
00219      * waitSyncTasks(self: pyaubo_sdk.SyncMove, arg0: str, arg1: Set[str]) -> int
00220      *
00221      * @par Lua 函数原型
00222      * waitSyncTasks(syncident: string, taskset: table) -> nil
00223      * @endcode

```

```

00224     * \endchinese
00225     * \english
00226     * waitSyncTasks is used to synchronize several task programs at a specific
00227     * point in the program.
00228     *
00229     * A waitSyncTasks instruction will wait for the other task programs. When all
00230     * task programs have reached the waitSyncTasks instruction, they will continue
00231     * their execution.
00232     *
00233     * @param syncident
00234     * @param taskset
00235     * @return
00236     *
00237     * @par Python prototype
00238     * waitSyncTasks(self: pyaubo_sdk.SyncMove, arg0: str, arg1: Set[str]) -> int
00239     *
00240     * @par Lua prototype
00241     * waitSyncTasks(syncident: string, taskset: table) -> nil
00242     * @endcode
00243     * \endenglish
00244     */
00245 int waitSyncTasks(const std::string &syncident, const TaskSet &taskset);
00246
00247 /**
00248     * @ingroup SyncMove
00249     * \chinese
00250     * isSyncMoveOn 用于判断机械单元组是否处于同步运动模式。
00251     *
00252     * 不控制任何机械单元的任务可以通过该函数判断参数“使用机械单元组”中定义的机械单元是否处于同步运动模式。
00253     *
00254     * @return
00255     *
00256     * @par Python 函数原型
00257     * isSyncMoveOn(self: pyaubo_sdk.SyncMove) -> bool
00258     *
00259     * @par Lua 函数原型
00260     * isSyncMoveOn() -> boolean
00261     * \endchinese
00262     * \english
00263     * isSyncMoveOn is used to tell if the mechanical unit group is in synchronized movement mode.
00264     *
00265     * A task that does not control any mechanical unit can find out if the mechanical units defined in the
    parameter Use Mechanical Unit Group are in synchronized movement mode.
00266     *
00267     * @return
00268     *
00269     * @par Python prototype
00270     * isSyncMoveOn(self: pyaubo_sdk.SyncMove) -> bool
00271     *
00272     * @par Lua prototype
00273     * isSyncMoveOn() -> boolean
00274     * \endenglish
00275     */
00276 bool isSyncMoveOn();
00277
00278 /**
00279     * @ingroup SyncMove
00280     * \chinese
00281     * 暂停同步运动模式。
00282     *
00283     * @return
00284     *
00285     * @par Python 函数原型
00286     * syncMoveSuspend(self: pyaubo_sdk.SyncMove) -> int
00287     *
00288     * @par Lua 函数原型
00289     * syncMoveSuspend() -> nil
00290     * \endchinese
00291     * \english
00292     * Suspend synchronized movement mode.
00293     *
00294     * @return
00295     *
00296     * @par Python prototype
00297     * syncMoveSuspend(self: pyaubo_sdk.SyncMove) -> int
00298     *
00299     * @par Lua prototype
00300     * syncMoveSuspend() -> nil
00301     * \endenglish
00302     */
00303 int syncMoveSuspend();
00304
00305 /**
00306     * @ingroup SyncMove
00307     * \chinese
00308     * 恢复同步运动模式。
00309     *

```

```

00310     * @return
00311     *
00312     * @par Python 函数原型
00313     * syncMoveResume(self: pyaubo_sdk.SyncMove) -> int
00314     *
00315     * @par Lua 函数原型
00316     * syncMoveResume() -> nil
00317     * \endchinese
00318     * \english
00319     * Resume synchronized movement mode.
00320     *
00321     * @return
00322     *
00323     * @par Python prototype
00324     * syncMoveResume(self: pyaubo_sdk.SyncMove) -> int
00325     *
00326     * @par Lua prototype
00327     * syncMoveResume() -> nil
00328     * \endenglish
00329     */
00330 int syncMoveResume();
00331
00332 /**
00333  * @ingroup SyncMove
00334  * \chinese
00335  * 添加一个名为 name 的坐标系，其初始位姿为 pose，位姿以 ref_frame 坐标系表达。
00336  * 此命令仅向世界模型添加一个坐标系，并不会将其附加到 ref_frame 坐标系。
00337  * 如需将新添加的坐标系附加到 ref_frame，请使用 frameAttach()。
00338  *
00339  * @param name: 要添加的坐标系名称。名称不能与任何已存在的世界模型对象（坐标系、轴或轴组）重复，否则会抛出异常。
00340  * @param pose: 新对象的初始位姿。
00341  * @param ref_frame: 位姿所表达的参考坐标系对象名称。若未指定，默认使用机器人“base”坐标系。
00342  *
00343  * @return
00344  * \endchinese
00345  * \english
00346  * Add a frame with the name "name" initialized at the specified pose
00347  * expressed in the ref_frame coordinate frame. This command only adds a
00348  * frame to the world, it does not attach it to the ref_frame coordinate
00349  * frame. Use frameAttach() to attach the newly added frame to ref_frame if
00350  * desired.
00351  *
00352  * @param name: name of the frame to be added. The name must not be the same
00353  * as any existing world model object (frame, axis, or axis group),
00354  * otherwise an exception is thrown
00355  * @param pose: initial pose of the new object
00356  * @param ref_frame: name of the world model object whose coordinate frame
00357  * the pose is expressed in. If nothing is provided here, the default is the
00358  * robot “base” frame.
00359  *
00360  * @return
00361  * \endenglish
00362  */
00363 int frameAdd(const std::string &name, const std::vector<double> &pose,
00364             const std::string &ref_name);
00365
00366 /**
00367  * @ingroup SyncMove
00368  * \chinese
00369  * 将子坐标系附加到父世界模型对象。附加时会设置父子之间的相对变换，使得子坐标系在世界中不会移动。
00370  *
00371  * 子坐标系不能为“world”、“flange”、“tcp”，也不能与父坐标系同名。
00372  *
00373  * 如果子或父不是已存在的坐标系，或导致形成闭环，则操作会失败。
00374  *
00375  * 如果用于 MotionPlus，parent 参数可以是外部轴或轴组的名称。
00376  *
00377  * @param child: 要附加的子坐标系名称，不能为“world”、“flange”或“tcp”。
00378  * @param parent: 子坐标系将要附加到的父对象名称。
00379  *
00380  * @return
00381  * \endchinese
00382  * \english
00383  * Attaches the child frame to the parent world model object. The relative
00384  * transform between the parent and child will be set such that the child
00385  * does not move in the world when the attachment occurs.
00386  *
00387  * The child cannot be “world”, “flange”, “tcp”, or the same as parent.
00388  *
00389  * This will fail if child or parent is not an existing frame, or this makes
00390  * the attachments form a closed chain.
00391  *
00392  * If being used with the MotionPlus, the parent argument can be the name of
00393  * an external axis or axis group.
00394  *
00395  * @param child: name of the frame to be attached. The name must not be
00396  * “world”, “flange”, or “tcp”.

```

```

00397     * @param parent: name of the object that the child frame will be attached
00398     * to.
00399     *
00400     * @return
00401     * \endenglish
00402     */
00403 int frameAttach(const std::string &child, const std::string &parent);
00404
00405 /**
00406  * @ingroup SyncMove
00407  * \chinese
00408  * 删除所有已添加到世界模型的坐标系。
00409  *
00410  * “world”、“base”、“flange” 和 “tcp” 坐标系不能被删除。
00411  *
00412  * 任何附加到被删除坐标系的坐标系将会被附加到 “world” 坐标系，并设置新的偏移，使得被分离的坐标系在世界中不会移动。
00413  *
00414  * @return
00415  * \endchinese
00416  * \english
00417  * Delete all frames that have been added to the world model.
00418  *
00419  * The “world”, “base”, “flange”, and “tcp” frames cannot be deleted.
00420  *
00421  * Any frames that are attached to the deleted frames will be attached to
00422  * the “world” frame with new frame offsets set such that the detached
00423  * frames do not move in the world.
00424  *
00425  * @return
00426  * \endenglish
00427  */
00428 int frameDeleteAll();
00429
00430 /**
00431  * @ingroup SyncMove
00432  * \chinese
00433  * 删除指定名称的坐标系。
00434  *
00435  * “world”、“base”、“flange” 和 “tcp” 坐标系不能被删除。
00436  *
00437  * 任何附加到被删除坐标系的坐标系将会被附加到 “world” 坐标系，并设置新的偏移，使得被分离的坐标系在世界中不会移动。
00438  *
00439  * 如果指定的坐标系不存在，则操作会失败。
00440  *
00441  * @param name: 要删除的坐标系名称
00442  *
00443  * @return
00444  * \endchinese
00445  * \english
00446  * Delete the frame with name from the world model.
00447  *
00448  * The “world”, “base”, “flange”, and “tcp” frames cannot be deleted.
00449  *
00450  * Any frames that are attached to the deleted frame will be attached to the
00451  * “world” frame with new frame offsets set such that the detached frame
00452  * does not move in the world.
00453  *
00454  * This command will fail if the frame does not exist.
00455  *
00456  * @param name: name of the frame to be deleted
00457  *
00458  * @return
00459  * \endenglish
00460  */
00461 int frameDelete(const std::string &name);
00462
00463 /**
00464  * @ingroup SyncMove
00465  * \chinese
00466  * 更改名为 name 的坐标系的位置，将其移动到由 pose 指定的新位置，pose 以 ref_name 坐标系表达。
00467  *
00468  * 如果 name 为 “world”、“flange”、“tcp”，或该坐标系不存在，则操作会失败。注意：如需移动 “tcp” 坐标系，请使用
00469  * set_tcp() 命令。
00470  *
00471  * 如果用于 MotionPlus，ref_name 参数可以是外部轴或轴组的名称。
00472  *
00473  * @param name: 要移动的坐标系名称
00474  * @param pose: 新的位置
00475  * @param ref_name: pose 所表达的参考坐标系，默认值为机器人的 “base” 坐标系。
00476  *
00477  * @return
00478  * \endchinese
00479  * \english
00480  * Changes the placement of the coordinate frame named name to the new
00481  * placement given by pose that is defined in the ref_name coordinate frame.
00482  *
00483  * This will fail if name is “world”, “flange”, “tcp”, or if the frame does

```

```

00483     * not exist. Note: to move the “tcp” frame, use the set_tcp() command
00484     * instead.
00485     *
00486     * If being used with the MotionPlus, the ref_name argument can be the name
00487     * of an external axis or axis group.
00488     *
00489     * @param name: the name of the frame to move
00490     * @param pose: the new placement
00491     * @param ref_name: the coordinate frame that pose is expressed in. The
00492     * default value is the robot's “base” frame.
00493     *
00494     * @return
00495     * \endenglish
00496     */
00497 int frameMove(const std::string &name, const std::vector<double> &pose,
00498               const std::string &ref_name);
00499
00500 /**
00501  * @ingroup SyncMove
00502  * \chinese
00503  * 获取名为 name 的坐标系相对于 rel_frame 坐标系的位姿, 并以 ref_frame 坐标系表达。
00504  * 如果未提供 ref_frame, 则返回 name 坐标系相对于 rel_frame 坐标系的位姿, 并以 rel_frame 坐标系表达。
00505  *
00506  * 如果任一参数不是已存在的坐标系, 则操作会失败。
00507  *
00508  * 如果用于 MotionPlus, 所有三个参数也可以是外部轴或轴组的名称。
00509  *
00510  * @param name: 要查询的坐标系名称。
00511  * @param rel_frame: “相对坐标系”, 用于计算相对位姿的坐标系。
00512  * @param ref_frame: “参考坐标系”, 用于表达结果相对位姿的坐标系。如果未提供, 则默认为 rel_frame。
00513  *
00514  * @return 以 ref_frame 坐标系表达的 name 坐标系的位姿。
00515  * \endchinese
00516  * \english
00517  * Get the pose of the name frame relative to the rel_frame frame but
00518  * expressed in the coordinates of the ref_frame frame. If ref_frame is not
00519  * provided, then this returns the pose of the name frame relative to and
00520  * expressed in the same frame as rel_frame.
00521  *
00522  * This will fail if any arguments are not an existing frame.
00523  *
00524  * If being used with MotionPlus, all three arguments can also be the names
00525  * of external axes or axis groups.
00526  *
00527  * @param name: name of the frame to query.
00528  * @param rel_frame: short for “relative frame” is the frame where the pose
00529  * is computed relative to
00530  * @param ref_frame: short for “reference frame” is the frame to express the
00531  * coordinates of resulting relative pose in. If this is not provided, then
00532  * it will default to match the value of rel_frame.
00533  *
00534  * @return The pose of the frame expressed in the ref_frame coordinates.
00535  * \endenglish
00536  */
00537 std::vector<double> frameGetPose(const std::string &name,
00538                                  const std::string &rel_frame,
00539                                  const std::string &ref_frame);
00540
00541 /**
00542  * @ingroup SyncMove
00543  * \chinese
00544  * 将位姿从 from_frame 坐标系转换到 to_frame 坐标系。
00545  *
00546  * 如果任一坐标系参数不是已存在的坐标系, 则操作会失败。
00547  *
00548  * 如果用于 MotionPlus, 所有三个参数也可以是外部轴或轴组的名称。
00549  *
00550  * @param pose: 要转换的位姿
00551  * @param from_frame: 原始坐标系的参考坐标系名称
00552  * @param to_frame: 新坐标系的参考坐标系名称
00553  *
00554  * @return 以 to_frame 坐标系表达的 pose 值。
00555  * \endchinese
00556  * \english
00557  * Convert pose from from_frame to to_frame.
00558  *
00559  * This will fail if either coordinate system argument is not an existing
00560  * frame.
00561  *
00562  * If being used with MotionPlus, all three arguments can also be the names
00563  * of external axes or axis groups.
00564  *
00565  * @param pose: pose to be converted
00566  * @param from_frame: name of reference frame at origin of old coordinate
00567  * system
00568  * @param to_frame: name of reference frame at origin of new coordinate
00569  * system

```

```

00570      *
00571      * @return Value of pose expressed in the coordinates of to_frame.
00572      * \endenglish
00573      */
00574      std::vector<double> frameConvertPose(const std::vector<double> &pose,
00575                                          const std::string &from_frame,
00576                                          const std::string &to_frame);
00577
00578      /**
00579      * @ingroup SyncMove
00580      * \chinese
00581      * 查询指定名称的坐标系是否存在。
00582      *
00583      * @param name: 要查询的坐标系名称。
00584      *
00585      * @return 如果存在该名称的坐标系则返回 true, 否则返回 false。
00586      * \endchinese
00587      * \english
00588      * Queries for the existence of a frame by the given name.
00589      *
00590      * @param name: name of the frame to be queried.
00591      *
00592      * @return Returns true if there is a frame by the given name, false if not.
00593      * \endenglish
00594      */
00595      bool frameExist(const std::string &name);
00596
00597      /**
00598      * @ingroup SyncMove
00599      * \chinese
00600      * 获取名为 name 的坐标系在世界模型中的父坐标系名称。
00601      *
00602      * 如果该坐标系没有附加到其他坐标系, 则其父坐标系为 "world"。
00603      *
00604      * @param name: 要查询的坐标系名称
00605      *
00606      * @return 父坐标系的名称, 字符串类型
00607      * \endchinese
00608      * \english
00609      * Get the parent of the frame named name in the world model.
00610      *
00611      * If the frame is not attached to another frame, then "world" is the
00612      * parent.
00613      *
00614      * @param name: the frame being queried
00615      *
00616      * @return name of the parent as a string
00617      * \endenglish
00618      */
00619      std::string frameGetParent(const std::string &name);
00620
00621      /**
00622      * @ingroup SyncMove
00623      * \chinese
00624      * 返回指定父对象的直接子对象坐标名称列表。父子关系由世界模型的附加关系定义。
00625      * 如果用于 MotionPlus, 子对象也可以是轴组或轴。
00626      *
00627      * @param name: 父对象的名称。
00628      *
00629      * @return 直接子对象坐标名称列表
00630      * \endchinese
00631      * \english
00632      * Returns a list of immediate child object frame names. Parent-child
00633      * relationships are defined by world model attachments. If being used with
00634      * MotionPlus, the child objects may also be an axis group or an axis.
00635      *
00636      * @param name: the name of the parent object.
00637      *
00638      * @return a list of immediate child object frame names
00639      * \endenglish
00640      */
00641      std::vector<std::string> frameGetChildren(const std::string &name);
00642
00643      /**
00644      * @ingroup SyncMove
00645      * \chinese
00646      * 向世界模型添加一个新的轴组, 名称为 name。轴组基座放置在 ref_frame 坐标系下的 pose 位置。
00647      *
00648      * 轴组只能附加到世界坐标系。
00649      *
00650      * 每个轴组的基座都附加有一个坐标系, 可以通过轴组名称作为参数传递给其他世界模型函数。
00651      *
00652      * 世界模型最多可添加 6 个轴组。
00653      *
00654      * @param name: 要添加的轴组名称, 不能为空字符串。世界模型对象 (如坐标系、轴组、轴等) 的名称必须唯一。
00655      * @param pose: 轴组基座在参考坐标系下的位姿。
00656      * @param ref_frame (可选): pose 所在的参考坐标系名称, 可以是任何带有坐标系的世界模型实体 (如坐标系、轴组、轴等)。默认

```

```

    值 "base" 表示机器人基座坐标系。
00657     *
00658     * @return
00659     * \endchinese
00660     * \english
00661     * Adds a new axis group with the given name to the world model. It is
00662     * placed at the given pose in the reference coordinate frame defined by
00663     * ref_frame.
00664     *
00665     * An axis group can only be attached to the world coordinate frame.
00666     *
00667     * Each axis group has a coordinate frame attached to its base, which can be
00668     * used as an argument to other world model functions by referring the name
00669     * of the group.
00670     *
00671     * At most 6 axis groups can be added to the world model.
00672     *
00673     * @param name: (string) Name of the axis group to add. The name cannot be
00674     * an empty string. Names used by world model objects (e.g., frame, axis
00675     * group, axis, etc.) must be unique.
00676     *
00677     * @param pose: (pose) Pose of the axis group's base, in the reference
00678     * coordinate frame.
00679     *
00680     * @param ref_frame (optional): (string) Name of the reference coordinate
00681     * frame that pose is defined in. This can be any world model entity with a
00682     * coordinate system (e.g., frame, axis group, axis, etc.). The default
00683     * value "base" refers to the robot's base frame.
00684     *
00685     * @return
00686     * \endenglish
00687     */
00688 int axisGroupAdd(const std::string &name, const std::vector<double> &pose,
00689                 const std::string &ref_frame);
00690
00691 /**
00692  * @ingroup SyncMove
00693  * \chinese
00694  * 删除具有给定名称的轴组。
00695  *
00696  * 所有附加的轴也会被禁用（如果处于活动状态）并删除。
00697  *
00698  * 如果该轴组正被其他函数控制，则操作会失败。
00699  *
00700  * @param name: 要删除的轴组名称。该名称的轴组必须存在。
00701  *
00702  * @return
00703  * \endchinese
00704  * \english
00705  * Deletes the axis group with the given name from the world model.
00706  *
00707  * All attached axes are also disabled (if live) and deleted.
00708  *
00709  * This function will fail, if this axis group is under control by another function.
00710  *
00711  * @param name: (string) Name of the axis group to delete. Axis group with
00712  * such name must exist.
00713  *
00714  * @return
00715  * \endenglish
00716  */
00717 int axisGroupDelete(const std::string &name);
00718
00719 /**
00720  * @ingroup SyncMove
00721  * \chinese
00722  * 向名为 group_name 的轴组添加一个名为 name 的外部轴。该轴在 parent 坐标系下的 pose 位置附加，pose 表示轴位置为 0 时
    的位姿。
00723  * 轴的类型、最大速度、最大加速度、位置限制和索引分别由 type、v_limit、a_limit、q_limits 和 axis_index 定义。
00724  * pose 参数通常通过外部轴调试标定流程获得。
00725  * 如果该轴组正被其他函数控制，或附加关系形成闭环，则操作会失败。
00726  *
00727  * @param group_name: 要添加轴的轴组名称，需已通过 axis_group_add() 创建且存在。
00728  * @param name: 新轴的名称，不能为空且需唯一。
00729  * @param parent: 父轴名称，若为空或与 group_name 相同，则附加到轴组基座。父轴需已存在于该轴组。
00730  * @param pose: 轴在父坐标系下的零位姿。type 为 0（旋转轴）时，z 轴为旋转轴；type 为 1（直线轴）时，z 轴为移动方向。
00731  * @param type: 轴类型，0 表示旋转轴，1 表示直线轴。
00732  * @param v_limit: 最大速度。
00733  * @param a_limit: 最大加速度。
00734  * @param q_limits: 位置限制。
00735  * @param axis_index: 轴索引。
00736  * \endchinese
00737  * \english
00738  * Adds an external axis with the given name to the axis group named group_name.
00739  * The axis is attached at the given pose in the reference coordinate frame defined by parent when its axis
    position is 0.
00740  * The type, max velocity, max acceleration, position limits, and index of this axis are defined by type,

```



```

v_limit, a_limit, q_limits, and axis_index, respectively.
00741 * The pose parameter is typically obtained from a calibration process when the external axis is commissioned.
00742 * This function will fail if this axis group is under control of another function, or if the kinematic chain
    created by the attachment forms a closed chain.
00743 *
00744 * @param group_name: Name of the axis group this new axis is added to. The axis group would have been created
    using axis_group_add(). Axis group with such name must exist.
00745 * @param name: Name of the new axis. The name cannot be an empty string. Names used by world model objects
    (e.g., frame, axis group, axis, etc.) must be unique.
00746 * @param parent: Name of the parent axis. If it's empty or the same as group_name, the new axis will be
    attached to the base of the axis group. Axis with such name must exist in the axis group.
00747 * @param pose: The zero-position pose, in the parent coordinate frame, this axis will be placed and attached
    to. This is the pose the axis will be (relative to its parent) when its axis position is 0. If type is 0 (rotary),
    then the z axis of the frame corresponds to the axis of rotation. If type is 1 (linear), then the z axis is the axis
    of translation.
00748 * @param type: Axis type, 0 for rotary, 1 for linear.
00749 * @param v_limit: Maximum velocity.
00750 * @param a_limit: Maximum acceleration.
00751 * @param q_limits: Position limits.
00752 * @param axis_index: Axis index.
00753 * \endenglish
00754 */
00755 int axisGroupAddAxis(const std::string &group_name, const std::string &name,
00756                    const std::string &parent,
00757                    const std::vector<double> &pose);
00758
00759 /**
00760 * @ingroup SyncMove
00761 * \chinese
00762 * 更新指定名称的轴的相关属性。pose 参数通常通过外部轴调试标定流程获得。
00763 * 如果该轴所属的轴组正被其他命令控制，则操作会失败。
00764 * 如果该轴组中任何已附加的轴处于激活和使能状态，则操作会失败。
00765 *
00766 * @param name: 要更新的轴的名称，需已存在。
00767 * @param pose (可选): 轴在父轴（或轴组）坐标系下的零位姿。即轴位置为 0 时的位姿。
00768 * \endchinese
00769 * \english
00770 * Updates the corresponding properties of axis with name. The pose
00771 * parameter is typically obtained from a calibration process when the
00772 * external axis is commissioned. See here for a guide on a basic routine
00773 * for calibrating a single rotary axis.
00774 *
00775 * This function will fail, if the axis group the axis attached to is
00776 * already being controlled by another command.
00777 * This function will fail, if any attached axis of the axis group is live
00778 * and enabled.
00779 *
00780 * @param name: (string) Name of the axis to update. Axis with such name
00781 * must exist.
00782 *
00783 * @param pose (optional): (pose) New zero-position pose, in the coordinate
00784 * frame of the parent axis (or axis group), of the axis. This is the pose
00785 * of the axis when its axis position is 0.
00786 * \endenglish
00787 */
00788 int axisGroupUpdateAxis(const std::string &name,
00789                        const std::vector<double> &pose);
00790
00791 /**
00792 * @ingroup SyncMove
00793 * \chinese
00794 * 返回指定轴名称在 RTDE 目标位置 and 实际位置数组中的索引。
00795 *
00796 * @param axis_name: (string) 要查询的轴名称。该名称的轴必须存在。
00797 *
00798 * @return integer: 该轴在 RTDE 目标位置 and 实际位置数组中的索引。
00799 * \endchinese
00800 * \english
00801 * Returns the index of the axis with given axis_name in the RTDE target
00802 * positions and actual positions arrays.
00803 *
00804 * @param axis_name: (string) Name of the axis in query.
00805 * Axis with such name must exist.
00806 *
00807 * @return integer: Index of the axis in the RTDE target positions and
00808 * actual positions arrays.
00809 * \endenglish
00810 */
00811 int axisGroupGetAxisIndex(const std::string &name);
00812
00813 /**
00814 * @ingroup SyncMove
00815 * \chinese
00816 * 返回指定轴索引对应的轴名称。
00817 *
00818 * @param axis_index: (整数) 要查询的轴索引。该索引的轴必须存在。
00819 *

```



```

00820      * @return 字符串: 轴的名称。
00821      * \endchinese
00822      * \english
00823      * Returns the name of the axis with the given axis_index.
00824      *
00825      * @param axis_index: (integer) Index of the axis in query.
00826      * Axis with such index must exist.
00827      *
00828      * @return string: Name of the axis.
00829      * \endenglish
00830      */
00831      std::string axisGroupGetAxisName(int index);
00832
00833      /**
00834      * @ingroup SyncMove
00835      * \chinese
00836      * 返回指定轴组的当前目标位置。
00837      * 如果未指定 group_name, 则返回所有外部轴的目标位置。
00838      *
00839      * 如果外部轴总线被禁用, 则该函数会失败。
00840      *
00841      * @param group_name (可选): (string) 要查询的轴组名称。该名称的轴组必须真实存在。
00842      *
00843      * @return Double[]: 涉及轴的目标位置, 顺序为其外部轴索引顺序。
00844      * \endchinese
00845      * \english
00846      * Returns the current target positions of the axis group with group_name.
00847      * If group_name is not provided, the target positions of all external axes
00848      * will be returned.
00849      *
00850      * This function will fail, if the external axis bus is disabled.
00851      *
00852      * @param group_name (optional): (string) Name of the axis group in query.
00853      * Axis group with such name must REALLY exist.
00854      *
00855      * @return Double[]: Target positions of the involved axes, in the order of
00856      * their external axis indices.
00857      * \endenglish
00858      */
00859      std::vector<double> axisGroupGetTargetPositions(
00860          const std::string &group_name);
00861
00862      /**
00863      * @ingroup SyncMove
00864      * \chinese
00865      * 返回指定轴组的当前实际位置。
00866      * 如果未指定 group_name, 则返回所有外部轴的实际位置。
00867      *
00868      * 如果外部轴总线被禁用, 则该函数会失败。
00869      *
00870      * @param group_name (可选): (string) 要查询的轴组名称。该名称的轴组必须真实存在。
00871      *
00872      * @return Double[]: 涉及轴的实际位置, 顺序为其外部轴索引顺序。
00873      * \endchinese
00874      * \english
00875      * Returns the current actual positions of the axis group with group_name.
00876      * If group_name is not provided, the actual positions of all external axes
00877      * will be returned.
00878      *
00879      * This function will fail, if the external axis bus is disabled.
00880      *
00881      * @param group_name (optional): (string) Name of the axis group in query.
00882      * Axis group with such name must exist.
00883      *
00884      * @return Double[]: Actual positions of the involved axes, in the order of
00885      * their external axis indices.
00886      * \endenglish
00887      */
00888      std::vector<double> axisGroupGetActualPositions(
00889          const std::string &group_name);
00890
00891      /**
00892      * @ingroup SyncMove
00893      * \chinese
00894      * 通过给定的 offset, 将轴组 group_name 的目标位置和实际位置整体偏移。
00895      *
00896      * 这是一个仅在控制器内部进行的软件偏移, 不会影响外部轴驱动器。该偏移也会应用于通过 RTDE 发布的任何目标和实际位置流。
00897      *
00898      * @param group_name: (string) 要应用偏移的轴组名称。该名称的轴组必须存在。
00899      *
00900      * @param offset: (float[]) 目标和实际位置需要整体偏移的量。offset 的大小必须与该轴组所包含的轴数量一致。
00901      *
00902      * @return
00903      * \endchinese
00904      * \english
00905      * Shifts the target and actual positions of the axis group group_name by
00906      * the given offset.

```

```

00907      *
00908      * This is a software shift that happens in the controller only, it does not
00909      * affect external axis drives. The shift is also applied to any streamed
00910      * target and actual positions published on RTDE.
00911      *
00912      * @param group_name: (string) Name of the axis group to apply the offset
00913      * positions to. Axis group with such name must exist.
00914      *
00915      * @param offset: (float[]) Offsets that the target and actual positions
00916      * should be shifted by. The size of offset must match the number of axes
00917      * attached to the given group.
00918      *
00919      * @return
00920      * \endenglish
00921      */
00922  int axisGroupOffsetPositions(const std::string &group_name,
00923                             const std::vector<double> &offset);
00924
00925  /**
00926   * @ingroup SyncMove
00927   * \chinese
00928   * 以梯形速度曲线将名为 group_name 的轴组移动到新的位置 q。
00929   * 参数 a 指定本次运动的最大加速度占各轴加速度极限的百分比。
00930   * 参数 v 指定本次运动的最大速度占各轴速度极限的百分比。
00931   *
00932   * 实际的加速度和速度由最受限制的轴决定，以确保所有轴在加速、匀速和减速阶段同时完成。
00933   *
00934   * @param group_name: (string) 要移动的轴组名称。该名称的轴组必须存在。
00935   * @param q: (float[]) 目标位置，旋转轴为弧度，直线轴为米。如果目标超出位置极限，则会被限制在最近的极限值。涉及的轴按
其索引递增排序。q 的大小必须与该轴组包含的轴数量一致。
00936   * @param a: (float) 本次运动的最大加速度因子，取值范围 (0,1]，表示占加速度极限的百分比。
00937   * @param v: (float) 本次运动的最大速度因子，取值范围 (0,1]，表示占速度极限的百分比。
00938   *
00939   * 返回值: 无
00940   * \endchinese
00941   * \english
00942   * Moves the axes of axis group named group_name to new positions q, using a
00943   * trapezoidal velocity profile. Factor a specifying the percentage of the
00944   * max profile accelerations out of the acceleration limits of each axes.
00945   * Factor v specifying the percentage of the max profile velocities out of
00946   * the velocity limits of each axes.
00947   *
00948   * The actual accelerations and velocities are determined by the most
00949   * constraining axis, so that all the axes complete the acceleration,
00950   * cruise, and deceleration phases at the same time.
00951   *
00952   * @param group_name: (string) Name of the axis group to move.
00953   * Axis group with such name must exist.
00954   *
00955   * @param q: (float[]) Target positions in rad (rotary) or in m (linear). If
00956   * the target exceeds the position limits, then it is set to the nearest
00957   * limit. The involved axes are ordered increasingly by their axis indices.
00958   * The size of q must match the number of axes attached to the given group.
00959   *
00960   * @param a: (float) Factor specifying the max accelerations of this move
00961   * out of the acceleration limits. a must be in range of (0,1].
00962   *
00963   * @param v: (float) Factor specifying the max velocities of this move out
00964   * of the velocity limits. v must be in range of (0,1].
00965   *
00966   * Return: n/a
00967   * \endenglish
00968   */
00969  int axisGroupMoveJoint(const std::string &group_name,
00970                        const std::vector<double> &q, double a, double v);
00971
00972  /**
00973   * @ingroup SyncMove
00974   * \chinese
00975   * 以指定的加速度因子 a，将名为 group_name 的轴组加速到目标速度 qd。该函数会运行 t 秒。
00976   * @param group_name: (string) 要控制的外部轴组名称，必须已存在。
00977   * @param qd: (float[]) 轴组各轴的目标速度。如果目标速度超过速度极限，则会被限制在极限值。涉及的轴按其索引递增排序。qd
的大小必须与该轴组包含的轴数量一致。
00978   * @param a: (float) 本次运动的最大加速度因子，取值范围 (0,1]，表示占加速度极限的百分比。
00979   * @param t (可选): (float) 函数运行的持续时间 (秒)。若 t < 0，则函数将在目标速度达到时返回；若 t ≥ 0，则函数将在该持
续时间后返回，无论实际速度是否达到目标值。
00980   * \endchinese
00981   * \english
00982   * Accelerates the axes of axis group named group_name up to the target
00983   * velocities qd. Factor a specifying the percentage of the max
00984   * accelerations out of the acceleration limits of each axes. The function
00985   * will run for a period of t seconds.
00986   *
00987   * @param group_name: (string) Name of the external axis group to control.
00988   * Axis group with such name must exist.
00989   *
00990   * @param qd: (float[]) Target velocities for the axes in the axis group. If

```

```

00991      * the target exceeds the velocity limits, then it is set to the limit. The
00992      * involved axes are ordered increasingly by their axis indices. The size of
00993      * qd must match the number of axes attached to the given group.
00994      *
00995      * @param a: (float) Factor specifying the max accelerations of this move
00996      * out of the acceleration limits. a must be in range of (0,1].
00997      *
00998      * @param t (optional): (float) Duration in seconds before the function
00999      * returns. If t < 0, then the function will return when the target
01000      * velocities are reached. if t ≥ 0, then the function will return after
01001      * this duration, regardless of what the achieved axes velocities are.
01002      * \endenglish
01003      */
01004      int axisGroupSpeedJoint(const std::string &group_name,
01005                             const std::vector<double> &qd, double a, double t);
01006
01007      protected:
01008          void *d_;
01009      };
01010
01011      using SyncMovePtr = std::shared_ptr<SyncMove>;
01012  } // namespace common_interface
01013  } // namespace arcs
01014  #endif // AUBO_SDK_SYNC_MOVE_INTERFACE_H

```

12.44 include/aubo/system_info.h 文件参考

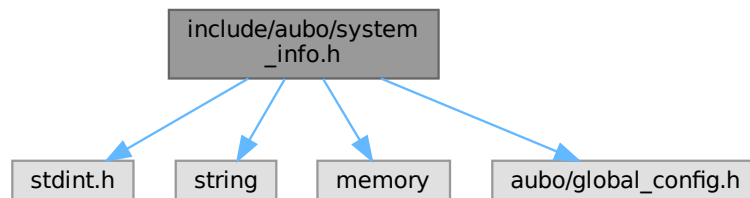
获取系统信息接口，如接口板的版本号、示教器软件的版本号

```

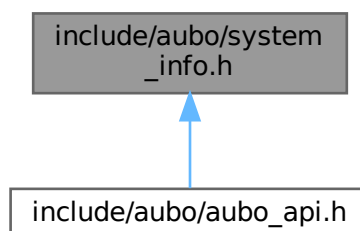
#include <stdint.h>
#include <string>
#include <memory>
#include <aubo/global_config.h>

```

system_info.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::common_interface::SystemInfo`

命名空间

- namespace `arcs`
- namespace `arcs::common_interface`

类型定义

- using `arcs::common_interface::SystemInfoPtr` = `std::shared_ptr<SystemInfo>`

12.44.1 详细描述

获取系统信息接口，如接口板的版本号、示教器软件的版本号
在文件 `system_info.h` 中定义。

12.45 system_info.h

[浏览该文件的文档.](#)

```
00001 /** @file system_info.h
00002  * @brief 获取系统信息接口，如接口板的版本号、示教器软件的版本号
00003  */
00004 #ifndef AUBO_SDK_SYSTEM_INFO_INTERFACE_H
00005 #define AUBO_SDK_SYSTEM_INFO_INTERFACE_H
00006
00007 #include <stdint.h>
00008 #include <string>
00009 #include <memory>
00010
00011 #include <aubo/global_config.h>
00012
00013 namespace arcs {
00014 namespace common_interface {
00015
00016 /**
00017  * @defgroup SystemInfo SystemInfo (系统信息)
00018  * \-chinese 系统信息
00019  * \-english SystemInfo
00020  */
00021 class ARCS_ABI_EXPORT SystemInfo
00022 {
00023 public:
00024     SystemInfo();
00025     virtual ~SystemInfo();
00026
00027     /**
00028      * @ingroup SystemInfo
00029      * \-chinese
00030      * 获取控制器软件版本号
00031      *
00032      * @return 返回控制器软件版本号
00033      *
00034      * @par Python 函数原型
00035      * getControlSoftwareVersionCode(self: pyaubo_sdk.SystemInfo) -> int
00036      *
00037      * @par Lua 函数原型
00038      * getControlSoftwareVersionCode() -> number
00039      *
00040      * @par C++ 示例
00041      * @code
00042      * int control_version =
00043      * rpc_cli->getSystemInfo()->getControlSoftwareVersionCode();
00044      * @endcode
00045      *
00046      * @par JSON-RPC 请求示例
00047      * {"jsonrpc": "2.0", "method": "SystemInfo.getControlSoftwareVersionCode", "params": [], "id": 1}
00048      *
00049      * @par JSON-RPC 响应示例
00050      * {"id": 1, "jsonrpc": "2.0", "result": 28003}
00051      *
00052      * \-endchinese
00053      * \-english
00054      * Get the controller software version code
00055      *
00056      * @return Returns the controller software version code
00057      */
```

```

00058     * @par Python prototype
00059     * getControlSoftwareVersionCode(self: pyaubo_sdk.SystemInfo) -> int
00060     *
00061     * @par Lua prototype
00062     * getControlSoftwareVersionCode() -> number
00063     *
00064     * @par C++ example
00065     * @code
00066     * int control_version =
00067     * rpc_cli->getSystemInfo()->getControlSoftwareVersionCode();
00068     * @endcode
00069     *
00070     * @par JSON-RPC request example
00071     * {"jsonrpc": "2.0", "method": "SystemInfo.getControlSoftwareVersionCode", "params": [], "id": 1}
00072     *
00073     * @par JSON-RPC response example
00074     * {"id": 1, "jsonrpc": "2.0", "result": 28003}
00075     *
00076     * \endenglish
00077     */
00078 int getControlSoftwareVersionCode();
00079
00080 /**
00081  * @ingroup SystemInfo
00082  * \chinese
00083  * 获取完整控制器软件版本号
00084  *
00085  * @return 返回完整控制器软件版本号
00086  *
00087  * @par Python 函数原型
00088  * getControlSoftwareFullVersion(self: pyaubo_sdk.SystemInfo) -> str
00089  *
00090  * @par Lua 函数原型
00091  * getControlSoftwareFullVersion() -> string
00092  *
00093  * @par C++ 示例
00094  * @code
00095  * std::string control_version =
00096  * rpc_cli->getSystemInfo()->getControlSoftwareFullVersion();
00097  * @endcode
00098  *
00099  * @par JSON-RPC 请求示例
00100  * {"jsonrpc": "2.0", "method": "SystemInfo.getControlSoftwareFullVersion", "params": [], "id": 1}
00101  *
00102  * @par JSON-RPC 响应示例
00103  * {"id": 1, "jsonrpc": "2.0", "result": "0.31.0-alpha.16+20alc76"}
00104  *
00105  * \endchinese
00106  * \english
00107  * Get the full controller software version
00108  *
00109  * @return Returns the full controller software version
00110  *
00111  * @par Python prototype
00112  * getControlSoftwareFullVersion(self: pyaubo_sdk.SystemInfo) -> str
00113  *
00114  * @par Lua prototype
00115  * getControlSoftwareFullVersion() -> string
00116  *
00117  * @par C++ example
00118  * @code
00119  * std::string control_version =
00120  * rpc_cli->getSystemInfo()->getControlSoftwareFullVersion();
00121  * @endcode
00122  *
00123  * @par JSON-RPC request example
00124  * {"jsonrpc": "2.0", "method": "SystemInfo.getControlSoftwareFullVersion", "params": [], "id": 1}
00125  *
00126  * @par JSON-RPC response example
00127  * {"id": 1, "jsonrpc": "2.0", "result": "0.31.0-alpha.16+20alc76"}
00128  *
00129  * \endenglish
00130  */
00131 std::string getControlSoftwareFullVersion();
00132
00133 /**
00134  * @ingroup SystemInfo
00135  * \chinese
00136  * 获取接口版本号
00137  *
00138  * @return 返回接口版本号
00139  *
00140  * @par Python 函数原型
00141  * getInterfaceVersionCode(self: pyaubo_sdk.SystemInfo) -> int
00142  *
00143  * @par Lua 函数原型
00144  * getInterfaceVersionCode() -> number

```

```

00145      *
00146      * @par C++ 示例
00147      * @code
00148      * int interface_version =
00149      * rpc_cli->getSystemInfo()->getInterfaceVersionCode();
00150      * @endcode
00151      *
00152      * @par JSON-RPC 请求示例
00153      * {"jsonrpc": "2.0", "method": "SystemInfo.getInterfaceVersionCode", "params": [], "id": 1}
00154      *
00155      * @par JSON-RPC 响应示例
00156      * {"id": 1, "jsonrpc": "2.0", "result": 22003}
00157      *
00158      * \endchinese
00159      * \english
00160      * Get the interface version code
00161      *
00162      * @return Returns the interface version code
00163      *
00164      * @par Python prototype
00165      * getInterfaceVersionCode(self: pyaubo_sdk.SystemInfo) -> int
00166      *
00167      * @par Lua prototype
00168      * getInterfaceVersionCode() -> number
00169      *
00170      * @par C++ example
00171      * @code
00172      * int interface_version =
00173      * rpc_cli->getSystemInfo()->getInterfaceVersionCode();
00174      * @endcode
00175      *
00176      * @par JSON-RPC request example
00177      * {"jsonrpc": "2.0", "method": "SystemInfo.getInterfaceVersionCode", "params": [], "id": 1}
00178      *
00179      * @par JSON-RPC response example
00180      * {"id": 1, "jsonrpc": "2.0", "result": 22003}
00181      *
00182      * \endenglish
00183      */
00184      int getInterfaceVersionCode();
00185
00186      /**
00187      * @ingroup SystemInfo
00188      * \chinese
00189      * 获取控制器软件构建时间
00190      *
00191      * @return 返回控制器软件构建时间
00192      *
00193      * @par Python 函数原型
00194      * getControlSoftwareBuildDate(self: pyaubo_sdk.SystemInfo) -> str
00195      *
00196      * @par Lua 函数原型
00197      * getControlSoftwareBuildDate() -> string
00198      *
00199      * @par C++ 示例
00200      * @code
00201      * std::string build_date =
00202      * rpc_cli->getSystemInfo()->getControlSoftwareBuildDate();
00203      * @endcode
00204      *
00205      * @par JSON-RPC 请求示例
00206      * {"jsonrpc": "2.0", "method": "SystemInfo.getControlSoftwareBuildDate", "params": [], "id": 1}
00207      *
00208      * @par JSON-RPC 响应示例
00209      * {"id": 1, "jsonrpc": "2.0", "result": "2024-3-5 07:03:20"}
00210      *
00211      * \endchinese
00212      * \english
00213      * Get the controller software build date
00214      *
00215      * @return Returns the controller software build date
00216      *
00217      * @par Python prototype
00218      * getControlSoftwareBuildDate(self: pyaubo_sdk.SystemInfo) -> str
00219      *
00220      * @par Lua prototype
00221      * getControlSoftwareBuildDate() -> string
00222      *
00223      * @par C++ example
00224      * @code
00225      * std::string build_date =
00226      * rpc_cli->getSystemInfo()->getControlSoftwareBuildDate();
00227      * @endcode
00228      *
00229      * @par JSON-RPC request example
00230      * {"jsonrpc": "2.0", "method": "SystemInfo.getControlSoftwareBuildDate", "params": [], "id": 1}
00231      *

```

```

00232     * @par JSON-RPC response example
00233     * {"id":1,"jsonrpc":"2.0","result":"2024-3-5 07:03:20"}
00234     *
00235     * \endenglish
00236     */
00237     std::string getControlSoftwareBuildDate();
00238
00239     /**
00240     * @ingroup SystemInfo
00241     * \chinese
00242     * 获取控制器软件 git 版本
00243     *
00244     * @return 返回控制器软件 git 版本
00245     *
00246     * @par Python 函数原型
00247     * getControlSoftwareVersionHash(self: pyaubo_sdk.SystemInfo) -> str
00248     *
00249     * @par Lua 函数原型
00250     * getControlSoftwareVersionHash() -> string
00251     *
00252     * @par C++ 示例
00253     * @code
00254     * std::string git_version =
00255     * rpc_cli->getSystemInfo()->getControlSoftwareVersionHash();
00256     * @endcode
00257     *
00258     * @par JSON-RPC 请求示例
00259     * {"jsonrpc":"2.0","method":"SystemInfo.getControlSoftwareVersionHash","params":[],"id":1}
00260     *
00261     * @par JSON-RPC 响应示例
00262     * {"id":1,"jsonrpc":"2.0","result":"fa4f64a"}
00263     *
00264     * \endchinese
00265     * \english
00266     * Get the controller software git version
00267     *
00268     * @return Returns the controller software git version
00269     *
00270     * @par Python prototype
00271     * getControlSoftwareVersionHash(self: pyaubo_sdk.SystemInfo) -> str
00272     *
00273     * @par Lua prototype
00274     * getControlSoftwareVersionHash() -> string
00275     *
00276     * @par C++ example
00277     * @code
00278     * std::string git_version =
00279     * rpc_cli->getSystemInfo()->getControlSoftwareVersionHash();
00280     * @endcode
00281     *
00282     * @par JSON-RPC request example
00283     * {"jsonrpc":"2.0","method":"SystemInfo.getControlSoftwareVersionHash","params":[],"id":1}
00284     *
00285     * @par JSON-RPC response example
00286     * {"id":1,"jsonrpc":"2.0","result":"fa4f64a"}
00287     *
00288     * \endenglish
00289     */
00290     std::string getControlSoftwareVersionHash();
00291
00292     /**
00293     * @ingroup SystemInfo
00294     * \chinese
00295     * 获取系统时间（软件启动时间 ns 纳秒）
00296     *
00297     * @return 返回系统时间（软件启动时间 ns 纳秒）
00298     *
00299     * @par Python 函数原型
00300     * getControlSystemTime(self: pyaubo_sdk.SystemInfo) -> int
00301     *
00302     * @par Lua 函数原型
00303     * getControlSystemTime() -> number
00304     *
00305     * @par C++ 示例
00306     * @code
00307     * std::string system_time =
00308     * rpc_cli->getSystemInfo()->getControlSystemTime();
00309     * @endcode
00310     *
00311     * @par JSON-RPC 请求示例
00312     * {"jsonrpc":"2.0","method":"SystemInfo.getControlSystemTime","params":[],"id":1}
00313     *
00314     * @par JSON-RPC 响应示例
00315     * {"id":1,"jsonrpc":"2.0","result":9287799079682}
00316     *
00317     * \endchinese
00318     * \english

```

```

00319     * Get the system time (software start time in nanoseconds)
00320     *
00321     * @return Returns the system time (software start time in nanoseconds)
00322     *
00323     * @par Python prototype
00324     * getControlSystemTime(self: pyaubo_sdk.SystemInfo) -> int
00325     *
00326     * @par Lua prototype
00327     * getControlSystemTime() -> number
00328     *
00329     * @par C++ example
00330     * @code
00331     * std::string system_time =
00332     * rpc_cli->getSystemInfo()->getControlSystemTime();
00333     * @endcode
00334     *
00335     * @par JSON-RPC request example
00336     * {"jsonrpc": "2.0", "method": "SystemInfo.getControlSystemTime", "params": [], "id": 1}
00337     *
00338     * @par JSON-RPC response example
00339     * {"id": 1, "jsonrpc": "2.0", "result": 9287799079682}
00340     *
00341     * \endenglish
00342     */
00343     uint64_t getControlSystemTime();
00344
00345 protected:
00346     void *d_;
00347 };
00348
00349 using SystemInfoPtr = std::shared_ptr<SystemInfo>;
00350
00351 } // namespace common_interface
00352 } // namespace arcs
00353 #endif

```

12.46 include/aubo/trace.h 文件参考

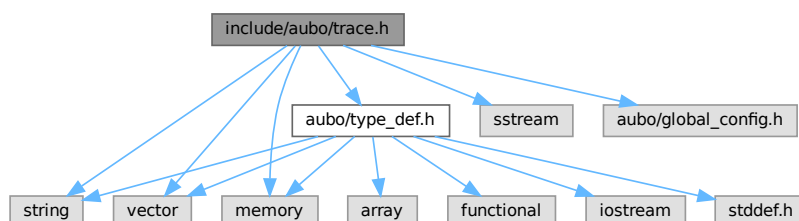
向控制器日志系统注入日志方面的接口

```

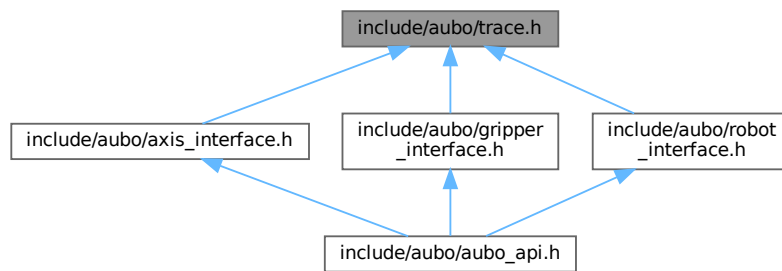
#include <string>
#include <vector>
#include <memory>
#include <sstream>
#include <aubo/global_config.h>
#include <aubo/type_def.h>

```

trace.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::common_interface::Trace`

命名空间

- namespace `arcs`
- namespace `arcs::common_interface`

类型定义

- using `arcs::common_interface::TracePtr = std::shared_ptr<Trace>`

12.46.1 详细描述

向控制器日志系统注入日志方面的接口
在文件 `trace.h` 中定义.

12.47 trace.h

[浏览该文件的文档.](#)

```

00001 /** @file trace.h
00002  * \-chinese @brief 向控制器日志系统注入日志方面的接口 \-english @brief Interface for injecting logs into the
00003  */
00004 #ifndef AUBO_SDK_TRACE_INTERFACE_H
00005 #define AUBO_SDK_TRACE_INTERFACE_H
00006
00007 #include <string>
00008 #include <vector>
00009 #include <memory>
00010 #include <sstream>
00011
00012 #include <aubo/global_config.h>
00013 #include <aubo/type_def.h>
00014
00015 namespace arcs {
00016 namespace common_interface {
00017
00018 /**
00019  * @defgroup Trace Trace (日志与弹窗)
00020  * \-chinese 提供给控制器扩展程序的日志记录系统 \-english provides a logging system for controller extension programs
00021  */
00022 class ARCS_ABI_EXPORT Trace
00023 {
00024 public:
00025     Trace();
00026     virtual ~Trace();
00027
00028     /**
00029      * @ingroup Trace
00030      * \-chinese 向 aubo_control 日志注入告警信息 \-english Injects alarm information into the aubo_control log
  
```

```

00031      *
00032      * TraceLevel: \n
00033      * 0 - FATAL \n
00034      * 1 - ERROR \n
00035      * 2 - WARNING \n
00036      * 3 - INFO \n
00037      * 4 - DEBUG \n
00038      *
00039      * \-chinese code 定义参考 error_stack \-english Code definitions refer to error_stack
00040      *
00041      * @param level
00042      * @param code
00043      * @param args
00044      * @return
00045      *
00046      * \-chinese @par Python 函数原型 \-english @par Python function prototype
00047      * alarm(self: pyaubo_sdk.Trace, arg0: arcs::common_interface::TraceLevel,
00048      * arg1: int, arg2: List[str]) -> int
00049      *
00050      * \-chinese @par Lua 函数原型 \-english @par Lua function prototype
00051      * alarm(level: number, code: number, args: table) -> nil
00052      *
00053      * \-chinese @par JSON-RPC 请求示例 \-english @par JSON-RPC request example
00054      * {"jsonrpc": "2.0", "method": "rob1.Trace.alarm", "params": ["", 1, ["Error", "Trajectory
00055      * planning failed!", "1"]], "id": 1}
00056      *
00057      * \-chinese @par JSON-RPC 响应示例 \-english @par JSON-RPC response example
00058      * {"id": 1, "jsonrpc": "2.0", "result": 0}
00059      *
00060      */
00061 int alarm(TraceLevel level, int code,
00062          const std::vector<std::string> &args = {});
00063
00064 /**
00065  * @ingroup Trace
00066  * \chinese
00067  * 打印文本信息到日志中
00068  *
00069  * @param msg 文本信息
00070  * @return
00071  *
00072  * @par Python 函数原型
00073  * textmsg(self: pyaubo_sdk.Trace, arg0: str) -> int
00074  *
00075  * @par Lua 函数原型
00076  * textmsg(msg: string) -> nil
00077  *
00078  * @par JSON-RPC 请求示例
00079  * {"jsonrpc": "2.0", "method": "rob1.Trace.textmsg", "params": ["test"], "id": 1}
00080  *
00081  * @par JSON-RPC 响应示例
00082  * {"id": 1, "jsonrpc": "2.0", "result": 0}
00083  * \endchinese
00084  * \english
00085  * print message into log
00086  *
00087  * @param msg message information
00088  * @return
00089  *
00090  * @par Python function prototype
00091  * textmsg(self: pyaubo_sdk.Trace, arg0: str) -> int
00092  *
00093  * @par Lua function prototype
00094  * textmsg(msg: string) -> nil
00095  *
00096  * @par JSON-RPC request example
00097  * {"jsonrpc": "2.0", "method": "rob1.Trace.textmsg", "params": ["test"], "id": 1}
00098  *
00099  * @par JSON-RPC response example
00100  * {"id": 1, "jsonrpc": "2.0", "result": 0}
00101  * \endenglish
00102  */
00103 int textmsg(const std::string &msg);
00104
00105 /**
00106  * \-chinese 通知上位机 \-english Notify the system
00107  *
00108  * @param msg
00109  * @return
00110  */
00111 int notify(const std::string &msg);
00112
00113 /**
00114  * @ingroup Trace
00115  * \chinese
00116  * 向连接的 RTDE 客户端发送弹窗请求
00117  *

```

```

00118     * @param level
00119     * @param title
00120     * @param msg
00121     * @param mode 模式 \n
00122     * 0: 普通模式 \n
00123     * 1: 阻塞模式 \n
00124     * 2: 输入模式 bool \n
00125     * 3: 输入模式 int \n
00126     * 4: 输入模式 double \n
00127     * 5: 输入模式 string \n
00128     * @return
00129     *
00130     * @par Python 函数原型
00131     * popup(self: pyaubo_sdk.Trace, arg0: arcs::common_interface::TraceLevel,
00132     * arg1: str, arg2: str, arg3: int) -> int
00133     *
00134     * @par Lua 函数原型
00135     * popup(level: number, title: string, msg: string, mode: number) -> nil
00136     *
00137     * @par JSON-RPC 请求示例
00138     * {"jsonrpc": "2.0", "method": "rob1.Trace.popup", "params": ["", "Error", "Trajectory
00139     * planning failed!", 1], "id": 1}
00140     *
00141     * @par JSON-RPC 响应示例
00142     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00143     * \endchinese
00144     * \english
00145     * Send a popup request to the connected RTDE client
00146     *
00147     * @param level
00148     * @param title
00149     * @param msg
00150     * @param mode mode \n
00151     * 0: normal mode \n
00152     * 1: blocking mode \n
00153     * 2: input mode bool \n
00154     * 3: input mode int \n
00155     * 4: input mode double \n
00156     * 5: input mode string \n
00157     * @return
00158     *
00159     * @par Python function prototype
00160     * popup(self: pyaubo_sdk.Trace, arg0: arcs::common_interface::TraceLevel,
00161     * arg1: str, arg2: str, arg3: int) -> int
00162     *
00163     * @par Lua function prototype
00164     * popup(level: number, title: string, msg: string, mode: number) -> nil
00165     *
00166     * @par JSON-RPC request example
00167     * {"jsonrpc": "2.0", "method": "rob1.Trace.popup", "params": ["", "Error", "Trajectory
00168     * planning failed!", 1], "id": 1}
00169     *
00170     * @par JSON-RPC response example
00171     * {"id": 1, "jsonrpc": "2.0", "result": 0}
00172     * \endenglish
00173     */
00174 int popup(TraceLevel level, const std::string &title,
00175           const std::string &msg, int mode);
00176
00177 /**
00178     * @ingroup Trace
00179     * \chinese
00180     * peek 最新的 AlarmInfo(上次一获取之后)
00181     *
00182     * last_time 设置为 0 时, 可以获取到所有的 AlarmInfo
00183     *
00184     * @param num
00185     * @param last_time
00186     * @return
00187     *
00188     * @par Python 函数原型
00189     * peek(self: pyaubo_sdk.Trace, arg0: int, arg1: int) ->
00190     * List[arcs::common_interface::RobotMsg]
00191     *
00192     * @par Lua 函数原型
00193     * peek(num: number, last_time: number) -> table
00194     *
00195     * @par JSON-RPC 请求示例
00196     * {"jsonrpc": "2.0", "method": "rob1.Trace.peek", "params": [1, 0], "id": 1}
00197     *
00198     * @par JSON-RPC 响应示例
00199     * [{"id": 1, "jsonrpc": "2.0", "result": [{"args": ["RobotModeType.Running"],
00200     * "code": 30045, "level": "INFO", "source": "rob1", "timestamp": 5102883064300}]}]
00201     * \endchinese
00202     * \english
00203     * peek the latest AlarmInfo (after the last retrieval)
00204     *

```

```

00205     * When last_time is set as 0, retrieve all AlarmInfo
00206     *
00207     * @param num
00208     * @param last_time
00209     * @return
00210     *
00211     * @par Python function prototype
00212     * peek(self: pyaubo_sdk.Trace, arg0: int, arg1: int) ->
00213     * List[arcs::common_interface::RobotMsg]
00214     *
00215     * @par Lua function prototype
00216     * peek(num: number, last_time: number) -> table
00217     *
00218     * @par JSON-RPC request example
00219     * {"jsonrpc": "2.0", "method": "rob1.Trace.peak", "params": [1, 0], "id": 1}
00220     *
00221     * @par JSON-RPC response example
00222     * [{"id": 1, "jsonrpc": "2.0", "result": [{"args": ["RobotModeType.Running"],
00223     * "code": 30045, "level": "INFO", "source": "rob1", "timestamp": 5102883064300}]}]
00224     * \endenglish
00225     */
00226 RobotMsgVector peek(size_t num, uint64_t last_time = 0);
00227
00228 protected:
00229     void *d_;
00230 };
00231
00232 using TracePtr = std::shared_ptr<Trace>;
00233
00234 } // namespace common_interface
00235 } // namespace arcs
00236 #endif

```

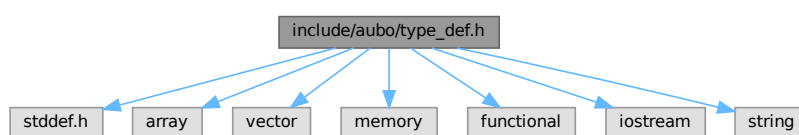
12.48 include/aubo/type_def.h 文件参考

数据类型的定义

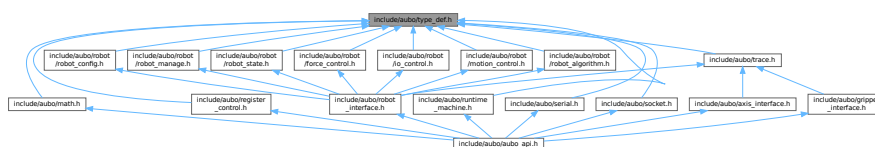
```

#include <stddef.h>
#include <array>
#include <vector>
#include <memory>
#include <functional>
#include <iostream>
#include <string>
type_def.h 的引用 (Include) 关系图:

```



此图展示该文件被哪些文件直接或间接地引用了:



类

- struct arcs::common_interface::RobotSafetyParameterRange

- struct `arcs::common_interface::WObjectData`
- struct `arcs::common_interface::VibrationRecalibrationParameter`
- struct `arcs::common_interface::CircleParameters`
圆周运动参数定义
- struct `arcs::common_interface::SpiralParameters`
- struct `arcs::common_interface::Enveloping`
- struct `arcs::common_interface::TrajConfig`
用于负载辨识的轨迹配置
- struct `arcs::common_interface::RobotMsg`
- struct `arcs::common_interface::GripperStatus`
- struct `arcs::common_interface::RtdeRecipe`
RTDE 菜单
- class `arcs::common_interface::AuboException`
自定义异常类 *AuboException*

命名空间

- namespace `arcs`
- namespace `arcs::common_interface`

宏定义

- #define `CARTESIAN_DOF` 6
Cartesian degree of freedom, 6 for x,y,z,rx,ry,rz
- #define `SAFETY_PARAM_SELECT_NUM` 2
正常 + 缩减
- #define `SAFETY_PLANES_NUM` 8
安全平面的数量
- #define `SAFETY_CUBIC_NUM` 10
安全立方体的数量
- #define `TOOL_CONFIGURATION_NUM` 3
工具配置数量
- #define `ENUM_AuboErrorCodes_DECLARES`
接口函数返回值定义整数为警告，负数为错误，0 为没有错误也没有警告
- #define `ENUM_RuntimeState_DECLARES`
The RuntimeState enum
- #define `ENUM_RobotModeType_DECLARES`
- #define `ENUM_SafetyModeType_DECLARES`
- #define `ENUM_OperationalModeType_DECLARES`
根据ISO 10218-1:2011(E) 5.7 节 *Automatic: In automatic mode, the robot shall execute the task programme and the safeguarding measures shall be functioning.*
- #define `ENUM_RobotControlModeType_DECLARES`
机器人的控制模式，最终的控制对象
- #define `ENUM_JointServoModeType_DECLARES`
- #define `ENUM_JointStateType_DECLARES`
- #define `ENUM_StandardInputAction_DECLARES`
- #define `ENUM_StandardOutputRunState_DECLARES`
- #define `ENUM_SafetyInputAction_DECLARES`
- #define `ENUM_SafetyOutputRunState_DECLARES`
- #define `ENUM_PayloadIdentifyMoveAxis_DECLARES`
- #define `ENUM_EnvelopingShape_DECLARES`
- #define `ENUM_TaskFrameType_DECLARES`
- #define `ENUM_TraceLevel_DECLARES`
- #define `ENUM_AxisModeType_DECLARES`

- `#define ENUM_HandleStateType_DECLARES`
- `#define ENUM_HandleModeType_DECLARES`
- `#define ENUM_SafeguedStopType_DECLARES`
- `#define ENUM_RobotEmergencyStopType_DECLARES`
- `#define ENUM_ITEM(c, n, ...)`
- `#define DECL_TO_STRING_FUNC(ENUM)`
- `#define ENUM_ITEM(c, n, ...)`
- `#define ENUM_ITEM(n, v, s)`
- `#define ENUM_ITEM(n, v, s)`
- `#define ENUM_ITEM(n, v, s)`

类型定义

- `using arcs::common_interface::Vector3d = std::array<double, 3>`
- `using arcs::common_interface::Vector4d = std::array<double, 4>`
- `using arcs::common_interface::Vector3f = std::array<float, 3>`
- `using arcs::common_interface::Vector4f = std::array<float, 4>`
- `using arcs::common_interface::Vector6f = std::array<float, 6>`
- `using arcs::common_interface::ResultWithErrno = std::tuple<std::vector<double>, int>`
- `using arcs::common_interface::ResultWithErrno1 = std::tuple<std::vector<std::vector<double>>, int>`
- `using arcs::common_interface::ResultWithErrno2 = std::tuple<std::vector<std::string>, int>`
- `using arcs::common_interface::ResultWithErrno3 = std::tuple<int, int>`
- `using arcs::common_interface::Payload`
- `using arcs::common_interface::ForceSensorCalibResult`
- `using arcs::common_interface::ForceSensorCalibResultWithError`
- `using arcs::common_interface::DynamicsModel`
- `using arcs::common_interface::Box = std::vector<double>`
- `using arcs::common_interface::Cylinder = std::vector<double>`
- `using arcs::common_interface::Sphere = std::vector<double>`
- `using arcs::common_interface::RobotMsgVector = std::vector<RobotMsg>`
- `using arcs::common_interface::GripperStatusVector = std::vector<GripperStatus>`

枚举

- `enum arcs::common_interface::AuboErrorCodes : int { arcs::common_interface::ENUM_AuboErrorCodes_DECLARES }`
 - `enum class arcs::common_interface::RuntimeState : int { arcs::common_interface::ENUM_RuntimeState_DECLARES }`
 - `enum class arcs::common_interface::RobotModeType : int { arcs::common_interface::ENUM_RobotModeType_DECLARES }`
 - `enum class arcs::common_interface::AxisModeType : int { arcs::common_interface::ENUM_AxisModeType_DECLARES }`
 - `enum class arcs::common_interface::HandleStateType : int { arcs::common_interface::ENUM_HandleStateType_DECLARES }`
 - `enum class arcs::common_interface::HandleModeType : int { arcs::common_interface::ENUM_HandleModeType_DECLARES }`
 - `enum class arcs::common_interface::SafetyModeType : int { arcs::common_interface::ENUM_SafetyModeType_DECLARES }`
 - `enum class arcs::common_interface::OperationalModeType : int { arcs::common_interface::ENUM_OperationalModeType_DECLARES }`
 - `enum class arcs::common_interface::RobotControlModeType : int { arcs::common_interface::ENUM_RobotControlModeType_DECLARES }`
- 安全状态:
- 操作模式
- 机器人控制模式

- enum class arcs::common_interface::JointServoModeType : int { arcs::common_interface::ENUM_JointServoModeType_STOPPED , arcs::common_interface::ENUM_JointServoModeType_RUNNING }
关节伺服模式
- enum class arcs::common_interface::JointStateType : int { arcs::common_interface::ENUM_JointStateType_DECODING , arcs::common_interface::ENUM_JointStateType_STOPPED , arcs::common_interface::ENUM_JointStateType_RUNNING }
关节状态
- enum class arcs::common_interface::StandardOutputRunState : int { arcs::common_interface::ENUM_StandardOutputRunState_STOPPED , arcs::common_interface::ENUM_StandardOutputRunState_RUNNING }
标准输出运行状态
- enum class arcs::common_interface::StandardInputAction : int { arcs::common_interface::ENUM_StandardInputAction_STOPPED , arcs::common_interface::ENUM_StandardInputAction_RUNNING }
The StandardInputAction enum
- enum class arcs::common_interface::SafetyInputAction : int { arcs::common_interface::ENUM_SafetyInputAction_STOPPED , arcs::common_interface::ENUM_SafetyInputAction_RUNNING }
安全输入动作
- enum class arcs::common_interface::SafetyOutputRunState : int { arcs::common_interface::ENUM_SafetyOutputRunState_STOPPED , arcs::common_interface::ENUM_SafetyOutputRunState_RUNNING }
安全输出运行状态
- enum arcs::common_interface::TaskFrameType { arcs::common_interface::ENUM_TaskFrameType_DECLARES , arcs::common_interface::ENUM_TaskFrameType_RUNNING }
任务帧类型
- enum arcs::common_interface::EnvelopingShape : int { arcs::common_interface::ENUM_EnvelopingShape_DECLARES , arcs::common_interface::ENUM_EnvelopingShape_RUNNING }
包络形状
- enum arcs::common_interface::PayloadIdentifyMoveAxis : int { arcs::common_interface::ENUM_PayloadIdentifyMoveAxis_STOPPED , arcs::common_interface::ENUM_PayloadIdentifyMoveAxis_RUNNING }
负载识别移动轴
- enum arcs::common_interface::TraceLevel { arcs::common_interface::ENUM_TraceLevel_DECLARES , arcs::common_interface::ENUM_TraceLevel_RUNNING }
跟踪级别
- enum arcs::common_interface::SafeguardedStopType : int { arcs::common_interface::ENUM_SafeguardedStopType_STOPPED , arcs::common_interface::ENUM_SafeguardedStopType_RUNNING }
安全防护停止类型
- enum arcs::common_interface::RobotEmergencyStopType : int { arcs::common_interface::ENUM_RobotEmergencyStopType_STOPPED , arcs::common_interface::ENUM_RobotEmergencyStopType_RUNNING }
机器人紧急停止类型
- enum class arcs::common_interface::ForceControlState { arcs::common_interface::Stopped , arcs::common_interface::Stropping , arcs::common_interface::Running }
力控制状态
- enum class arcs::common_interface::RefFrameType { arcs::common_interface::None , arcs::common_interface::Tool , arcs::common_interface::Path , arcs::common_interface::Base }
参考帧类型
- enum arcs::common_interface::error_type { arcs::common_interface::parse_error = -32700 , arcs::common_interface::invalid_request = -32600 , arcs::common_interface::method_not_found = -32601 , arcs::common_interface::invalid_params = -32602 , arcs::common_interface::internal_error = -32603 , arcs::common_interface::server_error = -32500 , arcs::common_interface::invalid = -32400 }
异常类型
- enum arcs::common_interface::ExceptionCode { arcs::common_interface::EC_DISCONNECTED = -1 , arcs::common_interface::EC_NOT_LOGINED = -2 , arcs::common_interface::EC_INVALID_SOCKET = -3 , arcs::common_interface::EC_REQUEST_BUSY = -4 , arcs::common_interface::EC_SEND_FAILED = -5 , arcs::common_interface::EC_RECV_TIMEOUT = -6 , arcs::common_interface::EC_RECV_ERROR = -7 , arcs::common_interface::EC_PARSE_ERROR = -8 , arcs::common_interface::EC_INVALID_REQUEST = -9 , arcs::common_interface::EC_METHOD_NOT_FOUND = -10 , arcs::common_interface::EC_INVALID_PARAMS = -11 , arcs::common_interface::EC_INTERNAL_ERROR = -12 , arcs::common_interface::EC_SERVER_ERROR = -13 , arcs::common_interface::EC_INVALID = -14 }
异常码

函数

- `std::ostream & arcs::common_interface::operator<< (std::ostream &os, const RobotSafetyParameterRange &vd)`
- `std::ostream & arcs::common_interface::operator<< (std::ostream &os, WObjectData p)`
- `std::ostream & arcs::common_interface::operator<< (std::ostream &os, const VibrationRecalibrationParameter p)`
- `std::ostream & arcs::common_interface::operator<< (std::ostream &os, CircleParameters p)`
- `std::ostream & arcs::common_interface::operator<< (std::ostream &os, SpiralParameters p)`
- `std::ostream & arcs::common_interface::operator<< (std::ostream &os, Enveloping p)`
- `std::ostream & arcs::common_interface::operator<< (std::ostream &os, TrajConfig p)`
- `const char * arcs::common_interface::returnValue2Str (int retval)`

12.48.1 详细描述

数据类型的定义

在文件 `type_def.h` 中定义.

12.48.2 宏定义说明

CARTESIAN_DOF

```
#define CARTESIAN_DOF 6
```

Cartesian degree of freedom, 6 for x,y,z,rx,ry,rz

在文件 `type_def.h` 第 25 行定义.

DECL_TO_STRING_FUNC

```
#define DECL_TO_STRING_FUNC(  
    ENUM)
```

值:

```
inline std::string toString(ENUM v)
{
    using T = ENUM;
    std::string name = #ENUM ".";
    ENUM_##ENUM##_DECLARES

    return #ENUM ".Unkown";
}
inline std::ostream &operator<<(std::ostream &os, ENUM v)
{
    os << toString(v);
    return os;
}
```

在文件 `type_def.h` 第 690 行定义.

ENUM_AuboErrorCodes_DECLARES

```
#define ENUM_AuboErrorCodes_DECLARES
```

接口函数返回值定义整数为警告, 负数为错误, 0 为没有错误也没有警告

在文件 `type_def.h` 第 232 行定义.

ENUM_AxisModeType_DECLARES

```
#define ENUM_AxisModeType_DECLARES
```

值:

```
ENUM_ITEM(NoController, -1, " 提供给示教器使用的, 如果 aubo_control 进程崩溃则会显示为 NoController") \
ENUM_ITEM(Disconnected, 0, " 未连接") \
ENUM_ITEM(PowerOff, 1, " 断电") \
ENUM_ITEM(BrakeReleasing, 2, " 刹车松开中") \
ENUM_ITEM(Idle, 3, " 空闲") \
ENUM_ITEM(Running, 4, " 运行中") \
ENUM_ITEM(Fault, 5, " 错误状态")
```

在文件 `type_def.h` 第 527 行定义.

ENUM_EnvelopingShape_DECLARES

```
#define ENUM_EnvelopingShape_DECLARES
```

值:

```
ENUM_ITEM(Cube, 1, " 立方体") \
ENUM_ITEM(Column, 2, " 柱状体") \
ENUM_ITEM(Stl, 3, " 以 STL 文件的形式描述负载碰撞集合体")
```

在文件 [type_def.h](#) 第 500 行定义.

ENUM_HandleModeType_DECLARES

```
#define ENUM_HandleModeType_DECLARES
```

值:

```
ENUM_ITEM(None, 0, " 不支持手柄") \
ENUM_ITEM(Default, 1, "S 等系列默认手柄") \
ENUM_ITEM(MultiFuncG1, 2, "G 系列第一代多功能手柄") \
```

在文件 [type_def.h](#) 第 543 行定义.

ENUM_HandleStateType_DECLARES

```
#define ENUM_HandleStateType_DECLARES
```

值:

```
ENUM_ITEM(Disconnected, 0, " 未连接") \
ENUM_ITEM(Init, 1, " 初始化中") \
ENUM_ITEM(Connected, 2, " 已连接") \
ENUM_ITEM(Lock, 3, " 锁定") \
ENUM_ITEM(Fault, 4, " 错误状态") \
```

在文件 [type_def.h](#) 第 536 行定义.

ENUM_ITEM [1/5]

```
#define ENUM_ITEM(
    c,
    n,
    ...)
```

值:

```
if (v == T::c) { \
    return name + #c; \
}
```

在文件 [type_def.h](#) 第 561 行定义.

ENUM_ITEM [2/5]

```
#define ENUM_ITEM(
    c,
    n,
    ...)
```

值:

```
c = n,
```

在文件 [type_def.h](#) 第 561 行定义.

ENUM_ITEM [3/5]

```
#define ENUM_ITEM(
    n,
    v,
    s)
```

值:

```
if (abs(retval) == v) \
    index = n##_INDEX;
```

在文件 [type_def.h](#) 第 561 行定义.

ENUM_ITEM [4/5]

```
#define ENUM_ITEM(
    n,
    v,
    s)
```

值:

n##_INDEX,

在文件 [type_def.h](#) 第 561 行定义.

ENUM_ITEM [5/5]

```
#define ENUM_ITEM(
    n,
    v,
    s)
```

值:

s,

在文件 [type_def.h](#) 第 561 行定义.

ENUM_JointServoModeType_DECLARES

```
#define ENUM_JointServoModeType_DECLARES
```

值:

```
ENUM_ITEM(Unknown, -1, " 未知") \
ENUM_ITEM(Open, 0, " 开环模式 (open loop mode)") \
ENUM_ITEM(Current, 1, " 电流伺服模式 (current servo mode)") \
ENUM_ITEM(Velocity, 2, " 速度伺服模式 (speed servo mode)") \
ENUM_ITEM(Position, 3, " 位置伺服模式 (position servo mode)") \
ENUM_ITEM(Torque, 4, " 力矩伺服模式 (torque servo mode)") \
```

在文件 [type_def.h](#) 第 411 行定义.

ENUM_JointStateType_DECLARES

```
#define ENUM_JointStateType_DECLARES
```

值:

```
ENUM_ITEM(Poweroff, 0, " 节点未连接到接口板或者已经断电 (node not connected to interface board or already powered off)") \
ENUM_ITEM(Idle, 2, " 节点空闲 (node idle)") \
ENUM_ITEM(Fault, 3, " 节点错误, 节点停止伺服运动, 刹车抱死 (node error, node stopped servo move, brake engaged)") \
ENUM_ITEM(Running, 4, " 节点伺服 (node servo)") \
ENUM_ITEM(Bootload, 5, " 节点 bootloader 状态, 暂停一切通讯 (node bootloader state, pause all communication)") \
```

在文件 [type_def.h](#) 第 419 行定义.

ENUM_OperationalModeType_DECLARES

```
#define ENUM_OperationalModeType_DECLARES
```

值:

```
ENUM_ITEM(Disabled, 0, " 禁用模式: 不使用 Operational Mode") \
ENUM_ITEM(Automatic, 1, " 自动模式: 机器人正常工作模式, 运行速度不会被限制 (auto mode: robot normal operation, speed will not be limited)") \
ENUM_ITEM(Manual, 2, " 手动模式: 机器人编程示教模式 (T1), 机器人运行速度将会被限制或者机器人程序校验模式 (T2) (manual mode: robot programming teaching mode (T1), robot running speed will be limited or robot program verification mode (T2))") \
```

根据ISO 10218-1:2011(E) 5.7 节 Automatic: In automatic mode, the robot shall execute the task programme and the safeguarding measures shall be functioning.

Automatic operation shall be prevented if any stop condition is detected. Switching from this mode shall result in a stop.

在文件 [type_def.h](#) 第 393 行定义.

ENUM_PayloadIdentifyMoveAxis_DECLARES

```
#define ENUM_PayloadIdentifyMoveAxis_DECLARES
```

值:

```
ENUM_ITEM(Joint_2_6, 0, " 第 2 和 6 关节运动") \
ENUM_ITEM(Joint_3_6, 1, " 第 3 和 6 关节运动") \
```

```
ENUM_ITEM(Joint_4_6, 2, " 第 4 和 6 关节运动") \
ENUM_ITEM(Joint_4_5_6, 3, " 第 4、5、6 关节运动") \
    在文件 type_def.h 第 494 行定义.
```

ENUM_RobotControlModeType_DECLARES

```
#define ENUM_RobotControlModeType_DECLARES
值:
ENUM_ITEM(Unknown, 0, " 未知的控制模式 (unknown control mode)") \
ENUM_ITEM(Position, 1, " 位置控制 movej (position control)") \
ENUM_ITEM(Speed, 2, " 速度控制 speedj/speedl (speed control)") \
ENUM_ITEM(Servo, 3, " 位置控制 servoj (position control)") \
ENUM_ITEM(Freedrive, 4, " 拖动示教 freedrive_mode") \
ENUM_ITEM(Force, 5, " 末端力控 force_mode") \
ENUM_ITEM(Torque, 6, " 关节力矩控制 (joint torque control)") \
ENUM_ITEM(Collision, 7, " 碰撞模式 (collision mode)") \
    机器人的控制模式, 最终的控制对象
    在文件 type_def.h 第 401 行定义.
```

ENUM_RobotEmergencyStopType_DECLARES

```
#define ENUM_RobotEmergencyStopType_DECLARES
值:
ENUM_ITEM(RobotEmergencyStopNone, 0, " 无紧急停止") \
ENUM_ITEM(RobotEmergencyStopControlBox, 1, " 紧急停止 (控制柜急停)") \
ENUM_ITEM(RobotEmergencyStopTeachPendant, 2, " 紧急停止 (示教器急停)") \
ENUM_ITEM(RobotEmergencyStopHandle, 3, " 紧急停止 (手柄急停)") \
ENUM_ITEM(RobotEmergencyStopEI, 4, " 紧急停止 (固定 IO 急停)")
    在文件 type_def.h 第 554 行定义.
```

ENUM_RobotModeType_DECLARES

```
#define ENUM_RobotModeType_DECLARES
值:
ENUM_ITEM(NoController, -1, " 提供给示教器使用的, 如果 aubo_control 进程崩溃则会显示为 NoController") \
ENUM_ITEM(Disconnected, 0, " 没有连接到机械臂本体 (控制器与接口板断开连接或是 EtherCAT 等总线断开)") \
ENUM_ITEM(ConfirmSafety, 1, " 正在进行安全配置, 断电状态下进行") \
ENUM_ITEM(Booting, 2, " 机械臂本体正在上电初始化") \
ENUM_ITEM(PowerOff, 3, " 机械臂本体处于断电状态") \
ENUM_ITEM(PowerOn, 4, " 机械臂本体上电成功, 刹车暂未松开 (抱死), 关节初始状态未获取") \
ENUM_ITEM(Idle, 5, " 机械臂上电成功, 刹车暂未松开 (抱死), 电机不通电, 关节初始状态获取完成") \
ENUM_ITEM(BrakeReleasing, 6, " 机械臂上电成功, 刹车正在松开") \
ENUM_ITEM(BackDrive, 7, " 反向驱动: 刹车松开, 电机不通电") \
ENUM_ITEM(Running, 8, " 机械臂刹车松开, 运行模式, 控制权由硬件移交给软件") \
ENUM_ITEM(Maintenance, 9, " 维护模式: 包括固件升级、参数写入等") \
ENUM_ITEM(Error, 10, "") \
ENUM_ITEM(PowerOffing, 11, " 机械臂本体处于断电过程中")
    The RobotModeType enum
    硬件强相关
    在文件 type_def.h 第 349 行定义.
```

ENUM_RuntimeState_DECLARES

```
#define ENUM_RuntimeState_DECLARES
值:
ENUM_ITEM(Running, 0, " 正在运行中") \
ENUM_ITEM(Retracting, 1, " 倒退") \
ENUM_ITEM(Pausing, 2, " 暂停中") \
ENUM_ITEM(Paused, 3, " 暂停状态") \
ENUM_ITEM(Stepping, 4, " 单步执行中") \
ENUM_ITEM(Stopping, 5, " 受控停止中 (保持原有轨迹)") \
ENUM_ITEM(Stopped, 6, " 已停止") \
ENUM_ITEM(Aborting, 7, " 停止 (最大速度关节运动停机)")
    The RuntimeState enum
    在文件 type_def.h 第 326 行定义.
```

ENUM_SafeguedStopType_DECLARES

```
#define ENUM_SafeguedStopType_DECLARES
值:
```

```
ENUM_ITEM(None, 0, " 无安全停止") \
ENUM_ITEM(SafeguardedStopIOInput, 1, " 安全停止 (IO 输入)") \
ENUM_ITEM(SafeguardedStop3PE, 2, " 安全停止 (三态开关)") \
ENUM_ITEM(SafeguardedStopOperational, 3, " 安全停止 (操作模式)")
```

在文件 `type_def.h` 第 548 行定义。

ENUM_SafetyInputAction_DECLARES

```
#define ENUM_SafetyInputAction_DECLARES
```

值:

```
ENUM_ITEM(Unassigned, 0, " 安全输入未分配动作") \
ENUM_ITEM(EmergencyStop, 1, " 安全输入触发急停") \
ENUM_ITEM(SafeguardStop, 2, " 安全输入触发防护停止, 边沿触发") \
ENUM_ITEM(SafeguardReset, 3, " 安全输入触发防护重置, 边沿触发") \
ENUM_ITEM(ThreePositionSwitch, 4, "3 档位使能开关") \
ENUM_ITEM(OperationalMode, 5, " 切换自动模式和手动模式") \
ENUM_ITEM(HandGuide, 6, " 拖动示教") \
ENUM_ITEM(ReducedMode, 7, " 安全参数切换 1(缩减模式), 序号越低优先级越高, 三路输出都无效时, 选用第 0 组安全参数") \
ENUM_ITEM(AutomaticModeSafeguardStop, 8, " 自动模式下防护停机输入 (需要配置三档位使能设备)") \
ENUM_ITEM(AutomaticModeSafeguardReset, 9, " 自动模式下上升沿触发防护重置 (需要配置三档位使能设备)")
```

在文件 `type_def.h` 第 471 行定义。

ENUM_SafetyModeType_DECLARES

```
#define ENUM_SafetyModeType_DECLARES
```

值:

```
ENUM_ITEM(Undefined, 0, " 安全状态待定") \
ENUM_ITEM(Normal, 1, " 正常运行模式") \
ENUM_ITEM(ReducedMode, 2, " 缩减运行模式") \
ENUM_ITEM(Recovery, 3, " 启动时如果在安全限制之外, 机器人将进入 recovery 模式") \
ENUM_ITEM(Violation, 4, " 超出安全限制 (根据安全配置, 例如速度超限等)") \
ENUM_ITEM(ProtectiveStop, 5, " 软件触发的停机 (保持轨迹, 不抱闸, 不断电)") \
ENUM_ITEM(SafeguardStop, 6, " IO 触发的防护停机 (不保持轨迹, 抱闸, 不断电)") \
ENUM_ITEM(SystemEmergencyStop, 7, " 系统急停: 急停信号由外部输入 (可配置输入), 不对外输出急停信号") \
ENUM_ITEM(RobotEmergencyStop, 8, " 机器人急停: 控制柜急停输入或者示教器急停按键触发, 对外输出急停信号") \
ENUM_ITEM(Fault, 9, " 机械臂硬件故障或者系统故障")
```

在文件 `type_def.h` 第 364 行定义。

ENUM_SafetyOutputRunState_DECLARES

```
#define ENUM_SafetyOutputRunState_DECLARES
```

值:

```
ENUM_ITEM(Unassigned, 0, " 安全输出未定义") \
ENUM_ITEM(SystemEmergencyStop, 1, " 输出高当有机器人急停输入或者急停按键被按下") \
ENUM_ITEM(NotSystemEmergencyStop, 2, " 输出低当有机器人急停输入或者急停按键被按下") \
ENUM_ITEM(RobotMoving, 3, " 输出高当有关节运动速度超过 0.1rad/s") \
ENUM_ITEM(RobotNotMoving, 4, " 输出高当所有的关节运动速度不超过 0.1rad/s") \
ENUM_ITEM(ReducedMode, 5, " 输出高当机器人处于缩减模式") \
ENUM_ITEM(NotReducedMode, 6, " 输出高当机器人不处于缩减模式") \
ENUM_ITEM(SafeHome, 7, " 输出高当机器人已经处于安全 Home 位姿") \
ENUM_ITEM(RobotNotStopping, 8, " 输出低当机器人正在急停或者安全停止中")
```

在文件 `type_def.h` 第 483 行定义。

ENUM_StandardInputAction_DECLARES

```
#define ENUM_StandardInputAction_DECLARES
```

值:

```
ENUM_ITEM(Default, 0, " 无触发") \
ENUM_ITEM(Handguide, 1, " 拖动示教, 高电平触发") \
ENUM_ITEM(GoHome, 2, " 运动到工程初始位姿, 高电平触发") \
ENUM_ITEM(StartProgram, 3, " 开始工程, 上升沿触发") \
ENUM_ITEM(StopProgram, 4, " 停止工程, 上升沿触发") \
ENUM_ITEM(PauseProgram, 5, " 暂停工程, 上升沿触发") \
ENUM_ITEM(PopupDismiss, 6, " 消除弹窗, 上升沿触发") \
ENUM_ITEM(PowerOn, 7, " 机器人上电/松刹车, 上升沿触发") \
ENUM_ITEM(PowerOff, 8, " 机器人抱死刹车/断电, 上升沿触发") \
ENUM_ITEM(ResumeProgram, 9, " 恢复工程, 上升沿触发") \
ENUM_ITEM(SlowDown1, 10, " 机器人减速触发 1, 高电平触发") \
ENUM_ITEM(SlowDown2, 11, " 机器人减速触发 2, 高电平触发") \
ENUM_ITEM(SafeStop, 12, " 安全停止, 高电平触发") \
ENUM_ITEM(RunningGuard, 13, " 信号, 高电平有效") \
ENUM_ITEM(MoveToFirstPoint, 14, " 运动到工程初始位姿, 高电平触发") \
ENUM_ITEM(xSlowDown1, 15, " 机器人减速触发 1, 低电平触发") \
ENUM_ITEM(xSlowDown2, 16, " 机器人减速触发 2, 低电平触发") \
ENUM_ITEM(ConveyorTrack, 17, " 传送带检测到物品触发, 高电平触发")
```

```
ENUM_ITEM(xConveyorTrack, 18, " 传送带检测到物品触发, 低电平触发") \
ENUM_ITEM(UnlockProtectiveStop, 19, " 解除保护性停止, 上升沿触发") \
ENUM_ITEM(ArbitraryResumeProgram, 20, " 恢复工程, 不检查当前位置和暂停点之间的距离, 上升沿触发")
```

在文件 [type_def.h](#) 第 426 行定义.

ENUM_StandardOutputRunState_DECLARES

```
#define ENUM_StandardOutputRunState_DECLARES
```

值:

```
ENUM_ITEM(None, 0, " 标准输出状态未定义") \
ENUM_ITEM(StopLow, 1, " 低电平指示工程停止") \
ENUM_ITEM(StopHigh, 2, " 高电平指示机器人停止") \
ENUM_ITEM(RunningHigh, 3, " 指示工程正在运行") \
ENUM_ITEM(PausedHigh, 4, " 指示工程已经暂停") \
ENUM_ITEM(AtHome, 5, " 高电平指示机器人正在拖动") \
ENUM_ITEM(Handguiding, 6, " 高电平指示机器人正在拖动") \
ENUM_ITEM(PowerOn, 7, " 高电平指示机器人已经上电") \
ENUM_ITEM(RobotEmergencyStop, 8, " 高电平指示机器人急停按下") \
ENUM_ITEM(SystemEmergencyStop, 9, " 高电平指示外部输入系统急停按下") \
ENUM_ITEM(InternalEmergencyStop, 8, " 高电平指示机器人急停按下") \
ENUM_ITEM(ExternalEmergencyStop, 9, " 高电平指示外部输入系统急停按下") \
ENUM_ITEM(SystemError, 10, " 系统错误, 包括故障、超限、急停、安全停止、防护停止 ") \
ENUM_ITEM(NotSystemError, 11, " 无系统错误, 包括普通模式、缩减模式和恢复模式 ") \
ENUM_ITEM(RobotOperable, 12, " 机器人可操作, 机器人上电且松刹车了 ") \
ENUM_ITEM(OperationalMode, 13, " 高电平指示自动模式, 低电平指示手动模式") \
ENUM_ITEM(SafeguardStop, 14, " 高电平指示处于安全停止状态") \
ENUM_ITEM(ProtectiveStop, 15, " 高电平指示处于防护停止状态") \
ENUM_ITEM(ProgramAborted, 16, " 高电平指示工程意外停止") \
ENUM_ITEM(WaypointArrived, 17, " 高电平指示机器人到达路点")
```

在文件 [type_def.h](#) 第 449 行定义.

ENUM_TaskFrameType_DECLARES

```
#define ENUM_TaskFrameType_DECLARES
```

值:

```
ENUM_ITEM(NONE, 0, "") \
ENUM_ITEM(POINT_FORCE, 1, " 力控坐标系发生变换, 使得力控参考坐标系的 y 轴沿着机器人 TCP 指向力控所选特征的原点, x 和 z 轴取决于所选特征的原始方向" \
    " 力控坐标系发生变换, 使得力控参考坐标系的 y 轴沿着机器人 TCP 指向力控所选特征的原点, x 和 z 轴取决于所选特征的原始方向" \
    " 机器人 TCP 与所选特征的起点之间的距离至少为 10mm" \
    " 优先选择 x 轴, 为所选特征的 x 轴在力控坐标系 Y 轴垂直平面上的投影, 如果所选特征的 x 轴与力控坐标系的 Y 轴平行, " \
    " 通过类似方法确定力控坐标系 Z 轴, Y-X 或者 Y-Z 轴确定之后, 通过右手法则确定剩下的轴") \
ENUM_ITEM(FRAME_FORCE, 2, " 力控坐标系不发生变换 SIMPLE_FORCE") \
ENUM_ITEM(MOTION_FORCE, 3, " 力控坐标系发生变换, 使得力控参考坐标系的 x 轴为机器人 TCP 速度在所选特征 x-y 平面上的投影 y 轴将垂直于机械臂运动, 并在所选特征的 x-y 平面内") \
ENUM_ITEM(TOOL_FORCE, 4, " 以工具末端坐标系作为力控参考坐标系")
```

在文件 [type_def.h](#) 第 505 行定义.

ENUM_TraceLevel_DECLARES

```
#define ENUM_TraceLevel_DECLARES
```

值:

```
ENUM_ITEM(FATAL, 0, "") \
ENUM_ITEM(ERROR, 1, "") \
ENUM_ITEM(WARNING, 2, "") \
ENUM_ITEM(INFO, 3, "") \
ENUM_ITEM(DEBUG, 4, "")
```

在文件 [type_def.h](#) 第 520 行定义.

SAFETY_CUBIC_NUM

```
#define SAFETY_CUBIC_NUM 10
```

安全立方体的数量

在文件 [type_def.h](#) 第 28 行定义.

被这些函数引用 [arcs::common_interface::RobotSafetyParameterRange::RobotSafetyParameterRange\(\)](#).

SAFETY_PARAM_SELECT_NUM

```
#define SAFETY_PARAM_SELECT_NUM 2
```

正常 + 缩减

在文件 `type_def.h` 第 26 行定义.

被这些函数引用 `arcs::common_interface::RobotSafetyParameterRange::RobotSafetyParameterRange()`.

SAFETY_PLANES_NUM

```
#define SAFETY_PLANES_NUM 8
```

安全平面的数量

在文件 `type_def.h` 第 27 行定义.

被这些函数引用 `arcs::common_interface::RobotSafetyParameterRange::RobotSafetyParameterRange()`.

TOOL_CONFIGURATION_NUM

```
#define TOOL_CONFIGURATION_NUM 3
```

工具配置数量

在文件 `type_def.h` 第 29 行定义.

被这些函数引用 `arcs::common_interface::RobotSafetyParameterRange::RobotSafetyParameterRange()`.

12.49 type_def.h

浏览该文件的文档.

```
00001 /** @file type_def.h
00002  * \-chinese @brief 数据类型的定义 \-english @brief enum type definitions
00003  */
00004 #ifndef AUBO_SDK_TYPE_DEF_H
00005 #define AUBO_SDK_TYPE_DEF_H
00006
00007 #include <stddef.h>
00008 #include <array>
00009 #include <vector>
00010 #include <memory>
00011 #include <functional>
00012 #include <iostream>
00013 #include <string>
00014
00015 #ifdef _MSC_VER
00016 #pragma execution_character_set("utf-8")
00017 #endif
00018
00019 // clang-format off
00020
00021 namespace arcs {
00022 namespace common_interface {
00023
00024 /// Cartesian degree of freedom, 6 for x,y,z,rx,ry,rz
00025 #define CARTESIAN_DOF 6
00026 #define SAFETY_PARAM_SELECT_NUM 2 ///< \-chinese 正常 + 缩减 \-english normal + reduced
00027 #define SAFETY_PLANES_NUM 8 ///< \-chinese 安全平面的数量 \-english Number of safety planes
00028 #define SAFETY_CUBIC_NUM 10 ///< \-chinese 安全立方体的数量 \-english Number of safety cubes
00029 #define TOOL_CONFIGURATION_NUM 3 ///< \-chinese 工具配置数量 \-english Number of tool configurations
00030
00031 using Vector3d = std::array<double, 3>;
00032 using Vector4d = std::array<double, 4>;
00033 using Vector3f = std::array<float, 3>;
00034 using Vector4f = std::array<float, 4>;
00035 using Vector6f = std::array<float, 6>;
00036
00037 struct RobotSafetyParameterRange
00038 {
00039     RobotSafetyParameterRange()
00040     {
00041         for (int i = 0; i < SAFETY_PARAM_SELECT_NUM; i++) {
00042             params[i].power = 0.;
00043             params[i].momentum = 0.;
00044             params[i].stop_time = 0.;
00045             params[i].stop_distance = 0.;
00046             params[i].reduced_entry_time = 0.;
00047             params[i].reduced_entry_distance = 0.;
00048             params[i].tcp_speed = 0.;
00049             params[i].elbow_speed = 0.;
00050             params[i].tcp_force = 0.;
00051             params[i].elbow_force = 0.;
00052             std::fill(params[i].qmin.begin(), params[i].qmin.end(), 0.);
00053             std::fill(params[i].qmax.begin(), params[i].qmax.end(), 0.);
00054             std::fill(params[i].qdmx.begin(), params[i].qdmx.end(), 0.);
00055             std::fill(params[i].joint_torque.begin(),
00056                     params[i].joint_torque.end(), 0.);
00057             params[i].tool_orientation.fill(0.);
00058         }
00059     }
00060 }
```

```

00058     params[i].tool_deviation = 0.;
00059     for (int j = 0; j < SAFETY_PLANES_NUM; j++) {
00060         params[i].planes[j].fill(0.);
00061         params[i].restrict_elbow[j] = 0;
00062     }
00063 }
00064 for (int i = 0; i < SAFETY_PLANES_NUM; i++) {
00065     trigger_planes[i].plane.fill(0.);
00066     trigger_planes[i].restrict_elbow = 0;
00067 }
00068 for (int i = 0; i < SAFETY_CUBIC_NUM; i++) {
00069     cubic[i].orig.fill(0.);
00070     cubic[i].size.fill(0.);
00071     cubic[i].restrict_elbow = 0;
00072 }
00073 for (int i = 0; i < TOOL_CONFIGURATION_NUM; i++) {
00074     tools[i].fill(0.);
00075 }
00076
00077 tool_inclination = 0.;
00078 tool_azimuth = 0.;
00079 std::fill(safety_home.begin(), safety_home.end(), 0.);
00080
00081 safety_input_emergency_stop = 0;
00082 safety_input_safeguard_stop = 0;
00083 safety_input_safeguard_reset = 0;
00084 safety_input_auto_safeguard_stop = 0;
00085 safety_input_auto_safeguard_reset = 0;
00086 safety_input_three_position_switch = 0;
00087 safety_input_operational_mode = 0;
00088 safety_input_reduced_mode = 0;
00089 safety_input_handguide = 0;
00090
00091 safety_output_emergency_stop = 0;
00092 safety_output_not_emergency_stop = 0;
00093 safety_output_robot_moving = 0;
00094 safety_output_robot_steady = 0;
00095 safety_output_reduced_mode = 0;
00096 safety_output_not_reduced_mode = 0;
00097 safety_output_safe_home = 0;
00098 safety_output_robot_not_stopping = 0;
00099 safety_output_safetyguard_stop = 0;
00100
00101 tp_3pe_for_handguide = 1;
00102 allow_manual_high_speed = 0;
00103 }
00104
00105 uint32_t crc32{ 0 };
00106
00107 /// \-chinese 最多可以保存 2 套参数, 默认使用第 0 套参数 \-english At most 2
00108 /// sets of parameters can be saved, default is the 0th set
00109 struct
00110 {
00111     float power; ///< \-chinese 关节力矩与关节角速度的乘积之和 \-english sum of joint torques times joint angular
speeds
00112     float momentum; ///< \-chinese 机器人动量限制 \-english robot momentum limit
00113     float stop_time; ///< \-chinese 停机时间 ms \-english stop time in milliseconds
00114     float stop_distance; ///< \-chinese 停机距离 m \-english stop distance in meters
00115     float reduced_entry_time; ///< \-chinese 进入缩减模式的最大时间 \-english maximum time to enter reduced mode
00116     float reduced_entry_distance; ///< \-chinese 进入缩减模式的最大距离 (可由安全平面触发) \-english maximum
distance to enter reduced mode (can be triggered by safety planes)
00117     float tcp_speed;
00118     float elbow_speed;
00119     float tcp_force;
00120     float elbow_force;
00121     std::vector<float> qmin;
00122     std::vector<float> qmax;
00123     std::vector<float> qdmax;
00124     std::vector<float> joint_torque;
00125     Vector3f tool_orientation; ///<
00126     float tool_deviation;
00127     Vector4f planes[SAFETY_PLANES_NUM]; ///< x,y,z,displacement
00128     int restrict_elbow[SAFETY_PLANES_NUM];
00129 } params[SAFETY_PARAM_SELECT_NUM];
00130
00131 /// \-chinese 8 个触发平面 \-english 8 trigger planes
00132 struct
00133 {
00134     Vector4f plane; ///< x,y,z,displacement
00135     int restrict_elbow;
00136 } trigger_planes[SAFETY_PLANES_NUM];
00137
00138 struct
00139 {
00140     Vector6f orig; ///< \-chinese 立方块的原点 (x,y,z,rx,ry,rz) \-english origin of the cubic (x,y,z,rx,ry,rz)
00141     Vector3f size; ///< \-chinese 立方块的尺寸 (x,y,z) \-english size of the cubic (x,y,z)
00142     int restrict_elbow;

```



```

00143     } cubic[SAFETY_CUBIC_NUM]; ///< -chinese 10 个安全空间 \-english 10 safety spaces
00144
00145     ///< -chinese 3 个工具 \-english 3 tools
00146     Vector4f tools[TOOL_CONFIGURATION_NUM]; ///< x,y,z,radius
00147
00148     float tool_inclination{
00149         0.
00150     }; ///< -chinese 倾角 \-english inclination angle
00151     float tool_azimuth{ 0. }; ///< -chinese 方位角 \-english azimuth angle
00152     std::vector<float> safety_home;
00153
00154     ///< -chinese 可配置 IO 的输入输出安全功能配置 \-english Configurable IO input
00155     ///< and output safety functions
00156     uint32_t safety_input_emergency_stop;
00157     uint32_t safety_input_safeguard_stop;
00158     uint32_t safety_input_safeguard_reset;
00159     uint32_t safety_input_auto_safeguard_stop;
00160     uint32_t safety_input_auto_safeguard_reset;
00161     uint32_t safety_input_three_position_switch;
00162     uint32_t safety_input_operational_mode;
00163     uint32_t safety_input_reduced_mode;
00164     uint32_t safety_input_handguide;
00165
00166     uint32_t safety_output_emergency_stop;
00167     uint32_t safety_output_not_emergency_stop;
00168     uint32_t safety_output_robot_moving;
00169     uint32_t safety_output_robot_steady;
00170     uint32_t safety_output_reduced_mode;
00171     uint32_t safety_output_not_reduced_mode;
00172     uint32_t safety_output_safe_home;
00173     uint32_t safety_output_robot_not_stopping;
00174     uint32_t safety_output_safeguard_stop;
00175
00176     int tp_3pe_for_handguide; ///< -chinese 是否将示教器三档位开关作为拖动功能开关 \-english Whether to use the
three-position switch of the teach pendant as a hand guiding function switch
00177     int allow_manual_high_speed; ///< -chinese 手动模式下允许高速运行 \-english Allow high-speed operation in manual
mode
00178 };
00179
00180 inline std::ostream &operator<<(std::ostream &os,
00181                                 const RobotSafetyParameterRange &vd)
00182 {
00183     // os << (int)vd;
00184     return os;
00185 }
00186
00187 struct WObjectData
00188 {
00189     ///< -chinese 是否为外部工具 \-english Whether it is an external tool
00190     bool remote_tool{ false };
00191
00192     ///< -chinese 工件坐标系耦合的 \-english Coupled with the workpiece
00193     ///< coordinate system
00194     std::string attach_frame{ "" };
00195
00196     ///< -english User coordinate system. \-chinese 用户坐标系
00197     ///< -chinese 如果 robhold 为 false, 那 uframe 的数值是基于 world \-english
00198     ///< If robhold is false, the values of uframe are based on world.
00199     ///< -chinese 否则, uframe 的数值是基于 flange \-english Otherwise, the
00200     ///< values of uframe are based on flange.
00201     std::vector<double> user_coord{ std::vector<double>(6, 0) };
00202
00203     ///< -chinese 工件坐标系, 基于 uframe \-english tool coordinate system,
00204     ///< based on uframe
00205     std::vector<double> obj_coord{ std::vector<double>(6, 0) };
00206 };
00207
00208 inline std::ostream &operator<<(std::ostream &os, WObjectData p)
00209 {
00210     return os;
00211 }
00212
00213
00214 struct VibrationRecalibrationParameter{
00215     double mass; // 负载质量
00216     std::vector<double> cog; // 负载质心
00217     std::vector<double> inertia; // 负载惯量
00218     std::vector<std::vector<double>> points; // 路点, 关节角
00219     std::vector<std::vector<double>> stiff_section; // 关节刚度数据
00220     std::vector<std::vector<double>> stiff_param; // 关节刚度数据
00221 };
00222
00223 inline std::ostream &operator<<(std::ostream &os, const VibrationRecalibrationParameter p)
00224 {
00225     return os;
00226 }
00227

```



```

00228 /// \-chinese 接口函数返回值定义 \-english Error codes definition
00229 ///
00230 /// \-chinese 整数为警告, 负数为错误, 0 为没有错误也没有警告 \-english whole
00231 /// number is warning, negative number is error, 0 is no error and no warning
00232 #define ENUM_AuboErrorCodes_DECLARES
00233     ENUM_ITEM(AUBO_OK, 0, "Success")
00234     ENUM_ITEM(AUBO_BAD_STATE, 1, "State error")
00235     ENUM_ITEM(AUBO_QUEUE_FULL, 2, "Planning queue full")
00236     ENUM_ITEM(AUBO_BUSY, 3, "The previous command is executing")
00237     ENUM_ITEM(AUBO_TIMEOUT, 4, "Timeout")
00238     ENUM_ITEM(AUBO_INVL_ARGUMENT, 5, "Invalid parameters")
00239     ENUM_ITEM(AUBO_NOT_IMPLEMENT, 6, "Interface not implemented")
00240     ENUM_ITEM(AUBO_NO_ACCESS, 7, "Cannot access")
00241     ENUM_ITEM(AUBO_CONN_REFUSED, 8, "Connection refused")
00242     ENUM_ITEM(AUBO_CONN_RESET, 9, "Connection is reset")
00243     ENUM_ITEM(AUBO_INPROGRESS, 10, "Execution in progress")
00244     ENUM_ITEM(AUBO_EIO, 11, "Input/Output error")
00245     ENUM_ITEM(AUBO_NOBUFFS, 12, "")
00246     ENUM_ITEM(AUBO_REQUEST_IGNORE, 13, "Request was ignored")
00247     ENUM_ITEM(AUBO_ALGORITHM_PLAN_FAILED, 14,
00248         "Motion planning algorithm error")
00249     ENUM_ITEM(AUBO_VERSION_INCOMPAT, 15, "Interface version unmatch")
00250     ENUM_ITEM(AUBO_DIMENSION_ERR, 16,
00251         "Input parameter dimension is incorrect")
00252     ENUM_ITEM(AUBO_SINGULAR_ERR, 17, "Input configuration may be singular")
00253     ENUM_ITEM(AUBO_POS_BOUND_ERR, 18,
00254         "Input position boundary exceeds the limit range")
00255     ENUM_ITEM(AUBO_INIT_POS_ERR, 19, "Initial position input is unreasonable")
00256     ENUM_ITEM(AUBO_ELP_SETTING_ERR, 20, "Envelope body setting error")
00257     ENUM_ITEM(AUBO_TRAJ_GEN_FAIL, 21, "Trajectory generation failed")
00258     ENUM_ITEM(AUBO_TRAJ_SELF_COLLISION, 22, "Trajectory self collision")
00259     ENUM_ITEM(
00260         AUBO_IK_NO_CONVERGE, 23,
00261         "Inverse kinematics computation did not converge; computation failed")
00262     ENUM_ITEM(AUBO_IK_OUT_OF_RANGE, 24,
00263         "Inverse kinematics result out of robot range")
00264     ENUM_ITEM(AUBO_IK_CONFIG_DISMATCH, 25,
00265         "Inverse kinematics input configuration contains errors")
00266     ENUM_ITEM(AUBO_IK_JACOBIAN_FAILED, 26,
00267         "The calculation of the inverse Jacobian matrix failed")
00268     ENUM_ITEM(AUBO_IK_NO_SOLU, 27,
00269         "The target point has solutions, but it has exceeded the joint "
00270         "limit conditions")
00271     ENUM_ITEM(AUBO_IK_UNKOWN_ERROR, 28, "Inverse kinematics unkown error")
00272     ENUM_ITEM(AUBO_MOVE_IGNORED_SERVOMODE, 29,
00273         "Robot is in servo mode where movement is disabled")
00274     ENUM_ITEM(AUBO_MOVE_INVALID_SPEED_VALUE, 30,
00275         "Movement speed must be a valid positive value (cannot be zero or negative)") \
00276     ENUM_ITEM(AUBO_MOVE_INVALID_ACCELERATION_VALUE, 31,
00277         "Movement acceleration must be a valid positive value (cannot be zero or negative)") \
00278     ENUM_ITEM(AUBO_INVALID_FILE_PATH, 32,
00279         "Please verify the file path. It appears to be invalid.") \
00280     ENUM_ITEM(AUBO_INST_QUEUED, 100, "Instruction pused into queue succeed")
00281     ENUM_ITEM(AUBO_INTERNAL_ERR, 101, "Internal error caused by alg .etc.")
00282     ENUM_ITEM(AUBO_BADSTATE_THREAD_DETACHED, 200,
00283         "Bad state: Operation not allowed on detached thread")
00284     ENUM_ITEM(AUBO_BADSTATE_THREAD_KILLED, 201,
00285         "Bad state: Operation not allowed on killed thread")
00286     ENUM_ITEM(AUBO_BADSTATE_TASK_NOT_FOUND, 202,
00287         "Bad state: Specified task id does not exist in the task queue")
00288     ENUM_ITEM(AUBO_BADSTATE_RTM_NOT_STARTED, 203,
00289         "Bad state: RuntimeMachine has not been started yet")
00290     ENUM_ITEM(AUBO_BADSTATE_RTM_NOT_STOPPED, 204,
00291         "Bad state: RuntimeMachine must be in Stopped state for this operation") \
00292     ENUM_ITEM(AUBO_BADSTATE_RTM_NOT_PAUSED, 205,
00293         "Bad state: RuntimeMachine must be in Paused state for this operation") \
00294     ENUM_ITEM(AUBO_BADSTATE_RTM_ABORTING, 206, "Bad state: RuntimeMachine is in aborting state") \
00295     ENUM_ITEM(AUBO_BADSTATE_PSTOP, 207, "Bad state: Operation blocked by protective stop") \
00296     ENUM_ITEM(AUBO_BADSTATE_ROBOT_ESTOP, 208, "Bad state: Robot emergency stop triggered") \
00297     ENUM_ITEM(AUBO_BADSTATE_SYSTEM_ESTOP, 209, "Bad state: System emergency stop triggered") \
00298     ENUM_ITEM(AUBO_BADSTATE_INVALID_ROBOT_MODE, 210,
00299         "Bad state: Robot is not in required operation mode")
00300     ENUM_ITEM(AUBO_BADSTATE_INVALID_SAFETY_MODE, 211,
00301         "Bad state: Robot is not in required safety mode")
00302     ENUM_ITEM(AUBO_BADSTATE_ROBOT_NOT_RUNNING, 212,
00303         "Bad state: Robot must be in Running mode for this operation")
00304     ENUM_ITEM(AUBO_BADSTATE_ROBOT_NOT_POWERED_OFF, 213,
00305         "Bad state: Robot must be in PowerOff mode for this operation") \
00306     ENUM_ITEM(AUBO_BADSTATE_SERIAL_OPEN_FAILED, 214, "Bad state: Failed to open serial device") \
00307     ENUM_ITEM(AUBO_BADSTATE_SERIAL_NOT_A_TERMINAL, 215,
00308         "Bad state: Specified device is not a terminal")
00309     ENUM_ITEM(AUBO_BADSTATE_SERIAL_CONFIG_FAILED, 216,
00310         "Bad state: Failed to configure serial port parameters")
00311     ENUM_ITEM(AUBO_BADSTATE_KINEMATICS_COMPENSATE_FAILED, 217,
00312         "Bad state: Failed to set kinematics compensation parameters")
00313     ENUM_ITEM(AUBO_BADSTATE_ROBOT_NOT_STEADY, 218, "Bad state: Robot is not in steady state") \
00314     ENUM_ITEM(AUBO_BADSTATE_FREE_DRIVE_ACTIVE, 219, "Bad state: Free-drive mode is active") \

```

```

00315     ENUM_ITEM(AUBO_BADSTATE_FORCE_CTRL_ACTIVE, 220, "Bad state: Force control mode is active") \
00316     ENUM_ITEM(AUBO_BADSTATE_SIMULATION_MODE_ACTIVE, 221, \
00317         "Bad state: Operation not allowed in simulation mode") \
00318     ENUM_ITEM(AUBO_BADSTATE_MB_ERROR, 222, "Bad state: Modbus signal has error") \
00319     ENUM_ITEM(AUBO_BADSTATE_MB_TIMEOUT, 223, "Bad state: Modbus signal timeout") \
00320     ENUM_ITEM(AUBO_ERR_UNKOWN, 99999, "Unkown error occurred.") \
00321
00322 /**
00323  * The RuntimeState enum
00324  *
00325  */
00326 #define ENUM_RuntimeState_DECLARES \
00327     ENUM_ITEM(Running, 0, " 正在运行中") \
00328     ENUM_ITEM(Retracting, 1, " 倒退") \
00329     ENUM_ITEM(Pausing, 2, " 暂停中") \
00330     ENUM_ITEM(Paused, 3, " 暂停状态") \
00331     ENUM_ITEM(Stepping, 4, " 单步执行中") \
00332     ENUM_ITEM(Stopping, 5, " 受控停止中 (保持原有轨迹)") \
00333     ENUM_ITEM(Stopped, 6, " 已停止") \
00334     ENUM_ITEM(Aborting, 7, " 停止 (最大速度关节运动停机)") \
00335
00336 /**
00337  * \chinese
00338  * @brief The RobotModeType enum
00339  *
00340  * 硬件强相关
00341  * \endchinese
00342  *
00343  * \english
00344  * @brief The RobotModeType enum
00345  *
00346  * Hardware related
00347  * \endenglish
00348  */
00349 #define ENUM_RobotModeType_DECLARES \
00350     ENUM_ITEM(NoController, -1, " 提供给示教器使用的, 如果 aubo_control 进程崩溃则会显示为 NoController") \
00351     ENUM_ITEM(Disconnected, 0, " 没有连接到机械臂本体 (控制器与接口板断开连接或是 EtherCAT 等总线断开)") \
00352     ENUM_ITEM(ConfirmSafety, 1, " 正在进行安全配置, 断电状态下进行") \
00353     ENUM_ITEM(Booting, 2, " 机械臂本体正在上电初始化") \
00354     ENUM_ITEM(PowerOff, 3, " 机械臂本体处于断电状态") \
00355     ENUM_ITEM(PowerOn, 4, " 机械臂本体上电成功, 刹车暂未松开 (抱死), 关节初始状态未获取") \
00356     ENUM_ITEM(Idle, 5, " 机械臂上电成功, 刹车暂未松开 (抱死), 电机不通电, 关节初始状态获取完成") \
00357     ENUM_ITEM(BrakeReleasing, 6, " 机械臂上电成功, 刹车正在松开") \
00358     ENUM_ITEM(BackDrive, 7, " 反向驱动: 刹车松开, 电机不通电") \
00359     ENUM_ITEM(Running, 8, " 机械臂刹车松开, 运行模式, 控制权由硬件移交给软件") \
00360     ENUM_ITEM(Maintainance, 9, " 维护模式: 包括固件升级、参数写入等") \
00361     ENUM_ITEM(Error, 10, "") \
00362     ENUM_ITEM(PowerOffing, 11, " 机械臂本体处于断电过程中") \
00363
00364 #define ENUM_SafetyModeType_DECLARES \
00365     ENUM_ITEM(Undefined, 0, " 安全状态待定") \
00366     ENUM_ITEM(Normal, 1, " 正常运行模式") \
00367     ENUM_ITEM(ReducedMode, 2, " 缩减运行模式") \
00368     ENUM_ITEM(Recovery, 3, " 启动时如果在安全限制之外, 机器人将进入 recovery 模式") \
00369     ENUM_ITEM(Violation, 4, " 超出安全限制 (根据安全配置, 例如速度超限等)") \
00370     ENUM_ITEM(ProtectiveStop, 5, " 软件触发的停机 (保持轨迹, 不抱闸, 不断电)") \
00371     ENUM_ITEM(SafeguardStop, 6, " IO 触发的防护停机 (不保持轨迹, 抱闸, 不断电)") \
00372     ENUM_ITEM(SystemEmergencyStop, 7, " 系统急停: 急停信号由外部输入 (可配置输入), 不对外输出急停信号") \
00373     ENUM_ITEM(RobotEmergencyStop, 8, " 机器人急停: 控制柜急停输入或者示教器急停按键触发, 对外输出急停信号") \
00374     ENUM_ITEM(Fault, 9, " 机械臂硬件故障或者系统故障") \
00375 //ValidateJointId
00376
00377 /**
00378  * \chinese
00379  * 根据 ISO 10218-1:2011(E) 5.7 节
00380  * Automatic: In automatic mode, the robot shall execute the task programme and
00381  * the safeguarding measures shall be functioning. Automatic operation shall be
00382  * prevented if any stop condition is detected. Switching from this mode shall
00383  * result in a stop.
00384  * \endchinese
00385  * \english
00386  * Based on ISO 10218-1:2011(E) Section 5.7
00387  * Automatic: In automatic mode, the robot shall execute the task programme and
00388  * the safeguarding measures shall be functioning. Automatic operation shall be
00389  * prevented if any stop condition is detected. Switching from this mode shall
00390  * result in a stop.
00391  * \endenglish
00392  */
00393 #define ENUM_OperationalModeType_DECLARES \
00394     ENUM_ITEM(Disabled, 0, " 禁用模式: 不使用 Operational Mode") \
00395     ENUM_ITEM(Automatic, 1, " 自动模式: 机器人正常工作模式, 运行速度不会被限制 (auto mode: robot normal operation, speed will not be limited)") \
00396     ENUM_ITEM(Manual, 2, " 手动模式: 机器人编程示教模式 (T1), 机器人运行速度将会被限制或者机器人程序校验模式 (T2) (manual mode: robot programming teaching mode (T1), robot running speed will be limited or robot program verification mode (T2)") \
00397
00398 /**

```

```

00399  * \-chinese 机器人的控制模式, 最终的控制对象 \-english Robot control mode, the final control object
00400  */
00401  #define ENUM_RobotControlModeType_DECLARES
00402      ENUM_ITEM(Unknown, 0, " 未知的控制模式 (unknown control mode)")
00403      ENUM_ITEM(Position, 1, " 位置控制 movej (position control)")
00404      ENUM_ITEM(Speed, 2, " 速度控制 speedj/speedl (speed control)")
00405      ENUM_ITEM(Servo, 3, " 位置控制 servoj (position control)")
00406      ENUM_ITEM(Freedrive, 4, " 拖动示教 freedrive_mode")
00407      ENUM_ITEM(Force, 5, " 末端力控 force_mode")
00408      ENUM_ITEM(Torque, 6, " 关节力矩控制 (joint torque control)")
00409      ENUM_ITEM(Collision, 7, " 碰撞模式 (collision mode)")
00410
00411  #define ENUM_JointServoModeType_DECLARES
00412      ENUM_ITEM(Unknown, -1, " 未知")
00413      ENUM_ITEM(Open, 0, " 开环模式 (open loop mode)")
00414      ENUM_ITEM(Current, 1, " 电流伺服模式 (current servo mode)")
00415      ENUM_ITEM(Velocity, 2, " 速度伺服模式 (speed servo mode)")
00416      ENUM_ITEM(Position, 3, " 位置伺服模式 (position servo mode)")
00417      ENUM_ITEM(Torque, 4, " 力矩伺服模式 (torque servo mode)")
00418
00419  #define ENUM_JointStateType_DECLARES
00420      ENUM_ITEM(Poweroff, 0, " 节点未连接到接口板或者已经断电 (node not conected to interface board or already powered
off)")
00421      ENUM_ITEM(Idle, 2, " 节点空闲 (node idle)")
00422      ENUM_ITEM(Fault, 3, " 节点错误, 节点停止伺服运动, 刹车抱死 (node error, node stopped servo move, brake
engaged)")
00423      ENUM_ITEM(Running, 4, " 节点伺服 (node servo)")
00424      ENUM_ITEM(Bootload, 5, " 节点 bootloader 状态, 暂停一切通讯 (node bootloader state, pause all
communication)")
00425
00426  #define ENUM_StandardInputAction_DECLARES
00427      ENUM_ITEM(Default, 0, " 无触发")
00428      ENUM_ITEM(Handguide, 1, " 拖动示教, 高电平触发")
00429      ENUM_ITEM(GoHome, 2, " 运动到工程初始位姿, 高电平触发")
00430      ENUM_ITEM(StartProgram, 3, " 开始工程, 上升沿触发")
00431      ENUM_ITEM(StopProgram, 4, " 停止工程, 上升沿触发")
00432      ENUM_ITEM(PauseProgram, 5, " 暂停工程, 上升沿触发")
00433      ENUM_ITEM(PopupDismiss, 6, " 消除弹窗, 上升沿触发")
00434      ENUM_ITEM(PowerOn, 7, " 机器人上电/松刹车, 上升沿触发")
00435      ENUM_ITEM(PowerOff, 8, " 机器人抱死刹车/断电, 上升沿触发")
00436      ENUM_ITEM(ResumeProgram, 9, " 恢复工程, 上升沿触发")
00437      ENUM_ITEM(SlowDown1, 10, " 机器人减速触发 1, 高电平触发")
00438      ENUM_ITEM(SlowDown2, 11, " 机器人减速触发 2, 高电平触发")
00439      ENUM_ITEM(SafeStop, 12, " 安全停止, 高电平触发")
00440      ENUM_ITEM(RunningGuard, 13, " 信号, 高电平有效")
00441      ENUM_ITEM(MoveToFirstPoint, 14, " 运动到工程初始位姿, 高电平触发")
00442      ENUM_ITEM(xSlowDown1, 15, " 机器人减速触发 1, 低电平触发")
00443      ENUM_ITEM(xSlowDown2, 16, " 机器人减速触发 2, 低电平触发")
00444      ENUM_ITEM(ConveyorTrack, 17, " 传送带检测到物品触发, 高电平触发")
00445      ENUM_ITEM(xConveyorTrack, 18, " 传送带检测到物品触发, 低电平触发")
00446      ENUM_ITEM(UnlockProtectiveStop, 19, " 解除保护性停止, 上升沿触发")
00447      ENUM_ITEM(ArbitraryResumeProgram, 20, " 恢复工程, 不检查当前位置和暂停点之间的距离, 上升沿触发")
00448
00449  #define ENUM_StandardOutputRunState_DECLARES
00450      ENUM_ITEM(None, 0, " 标准输出状态未定义")
00451      ENUM_ITEM(StopLow, 1, " 低电平指示工程停止")
00452      ENUM_ITEM(StopHigh, 2, " 高电平指示机器人停止")
00453      ENUM_ITEM(RunningHigh, 3, " 指示工程正在运行")
00454      ENUM_ITEM(PausedHigh, 4, " 指示工程已经暂停")
00455      ENUM_ITEM(AtHome, 5, " 高电平指示机器人正在拖动")
00456      ENUM_ITEM(Handguiding, 6, " 高电平指示机器人正在拖动")
00457      ENUM_ITEM(PowerOn, 7, " 高电平指示机器人已经上电")
00458      ENUM_ITEM(RobotEmergencyStop, 8, " 高电平指示机器人急停按下")
00459      ENUM_ITEM(SystemEmergencyStop, 9, " 高电平指示外部输入系统急停按下")
00460      ENUM_ITEM(InternalEmergencyStop, 8, " 高电平指示机器人急停按下")
00461      ENUM_ITEM(ExternalEmergencyStop, 9, " 高电平指示外部输入系统急停按下")
00462      ENUM_ITEM(SystemError, 10, " 系统错误, 包括故障、超限、急停、安全停止、防护停止 ")
00463      ENUM_ITEM(NotSystemError, 11, " 无系统错误, 包括普通模式、缩减模式和恢复模式 ")
00464      ENUM_ITEM(RobotOperable, 12, " 机器人可操作, 机器人上电且松刹车了 ")
00465      ENUM_ITEM(OperationalMode, 13, " 高电平指示自动模式, 低电平指示手动模式")
00466      ENUM_ITEM(SafeguardStop, 14, " 高电平指示处于安全停止状态")
00467      ENUM_ITEM(ProtectiveStop, 15, " 高电平指示处于防护停止状态")
00468      ENUM_ITEM(ProgramAborted, 16, " 高电平指示工程意外停止")
00469      ENUM_ITEM(WaypointArrived, 17, " 高电平指示机器人到达路点")
00470
00471  #define ENUM_SafetyInputAction_DECLARES
00472      ENUM_ITEM(Unassigned, 0, " 安全输入未分配动作")
00473      ENUM_ITEM(EmergencyStop, 1, " 安全输入触发急停")
00474      ENUM_ITEM(SafeguardStop, 2, " 安全输入触发防护停止, 边沿触发")
00475      ENUM_ITEM(SafeguardReset, 3, " 安全输入触发防护重置, 边沿触发")
00476      ENUM_ITEM(ThreePositionSwitch, 4, " 3 档位使能开关")
00477      ENUM_ITEM(OperationalMode, 5, " 切换自动模式和手动模式")
00478      ENUM_ITEM(HandGuide, 6, " 拖动示教")
00479      ENUM_ITEM(ReducuedMode, 7, " 安全参数切换 1(缩减模式), 序号越低优先级越高, 三路输出都无效时, 选用第 0 组安全参数")
00480
00481      ENUM_ITEM(AutomaticModeSafeguardStop, 8, " 自动模式下防护停机输入 (需要配置三档位使能设备)")
00481      ENUM_ITEM(AutomaticModeSafeguardReset, 9, " 自动模式下上升沿触发防护重置 (需要配置三档位使能设备)")

```

```

00482
00483 #define ENUM_SafetyOutputRunState_DECLARES \
00484     ENUM_ITEM(Unassigned, 0, " 安全输出未定义") \
00485     ENUM_ITEM(SystemEmergencyStop, 1, " 输出高当有机器人急停输入或者急停按键被按下") \
00486     ENUM_ITEM(NotSystemEmergencyStop, 2, " 输出低当有机器人急停输入或者急停按键被按下") \
00487     ENUM_ITEM(RobotMoving, 3, " 输出高当有关节运动速度超过 0.1rad/s") \
00488     ENUM_ITEM(RobotNotMoving, 4, " 输出高当所有的关节运动速度不超过 0.1rad/s") \
00489     ENUM_ITEM(ReducedMode, 5, " 输出高当机器人处于缩减模式") \
00490     ENUM_ITEM(NotReducedMode, 6, " 输出高当机器人不处于缩减模式") \
00491     ENUM_ITEM(SafeHome, 7, " 输出高当机器人已经处于安全 Home 位姿") \
00492     ENUM_ITEM(RobotNotStopping, 8, " 输出低当机器人正在急停或者安全停止中") \
00493
00494 #define ENUM_PayloadIdentifyMoveAxis_DECLARES \
00495     ENUM_ITEM(Joint_2_6, 0, " 第 2 和 6 关节运动") \
00496     ENUM_ITEM(Joint_3_6, 1, " 第 3 和 6 关节运动") \
00497     ENUM_ITEM(Joint_4_6, 2, " 第 4 和 6 关节运动") \
00498     ENUM_ITEM(Joint_4_5_6, 3, " 第 4、5、6 关节运动") \
00499
00500 #define ENUM_EnvelopingShape_DECLARES \
00501     ENUM_ITEM(Cube, 1, " 立方体") \
00502     ENUM_ITEM(Column, 2, " 柱状体") \
00503     ENUM_ITEM(Stl, 3, " 以 STL 文件的形式描述负载碰撞集合体") \
00504
00505 #define ENUM_TaskFrameType_DECLARES \
00506     ENUM_ITEM(NONE, 0, "") \
00507     ENUM_ITEM(POINT_FORCE, 1, " 力控坐标系发生变换, 使得力控参考坐标系的 y 轴沿着机器人 TCP 指向力控所选特征的原点, x 和 z 轴取决于所选特征的原始方向" \
00508     " 力控坐标系发生变换, 使得力控参考坐标系的 y 轴沿着机器人 TCP 指向力控所选特征的原点, x 和 z 轴取决于所选特征的原始方向" \
00509     " 机器人 TCP 与所选特征的起点之间的距离至少为 10mm" \
00510     " 优先选择 x 轴, 为所选特征的 x 轴在力控坐标系 Y 轴垂直平面上的投影, 如果所选特征的 x 轴与力控坐标系的 Y 轴平行, " \
00511     " 通过类似方法确定力控坐标系 Z 轴, Y-X 或者 Y-Z 轴确定之后, 通过右手法则确定剩下的轴") \
00512     ENUM_ITEM(FRAME_FORCE, 2, " 力控坐标系不发生变换 SIMPLE_FORCE") \
00513     ENUM_ITEM(MOTION_FORCE, 3, " 力控坐标系发生变换, 使得力控参考坐标系的 x 轴为机器人 TCP 速度在所选特征 x-y 平面上的投影 y 轴将垂直于机械臂运动, 并在所选特征的 x-y 平面内") \
00514     ENUM_ITEM(TOOL_FORCE, 4, " 以工具末端坐标系作为力控参考坐标系") \
00515
00516 #ifdef ERROR
00517 #undef ERROR
00518 #endif
00519
00520 #define ENUM_TraceLevel_DECLARES \
00521     ENUM_ITEM(FATAL, 0, "") \
00522     ENUM_ITEM(ERROR, 1, "") \
00523     ENUM_ITEM(WARNING, 2, "") \
00524     ENUM_ITEM(INFO, 3, "") \
00525     ENUM_ITEM(DEBUG, 4, "") \
00526
00527 #define ENUM_AxisModeType_DECLARES \
00528     ENUM_ITEM(NoController, -1, " 提供给示教器使用的, 如果 aubo_control 进程崩溃则会显示为 NoController") \
00529     ENUM_ITEM(Disconnected, 0, " 未连接") \
00530     ENUM_ITEM(PowerOff, 1, " 断电") \
00531     ENUM_ITEM(BrakeReleasing, 2, " 刹车松开中") \
00532     ENUM_ITEM(Idle, 3, " 空闲") \
00533     ENUM_ITEM(Running, 4, " 运行中") \
00534     ENUM_ITEM(Fault, 5, " 错误状态") \
00535
00536 #define ENUM_HandleStateType_DECLARES \
00537     ENUM_ITEM(Disconnected, 0, " 未连接") \
00538     ENUM_ITEM(Init, 1, " 初始化中") \
00539     ENUM_ITEM(Connected, 2, " 已连接") \
00540     ENUM_ITEM(Lock, 3, " 锁定") \
00541     ENUM_ITEM(Fault, 4, " 错误状态") \
00542
00543 #define ENUM_HandleModeType_DECLARES \
00544     ENUM_ITEM(None, 0, " 不支持手柄") \
00545     ENUM_ITEM(Default, 1, "S 等系列默认手柄") \
00546     ENUM_ITEM(MultiFuncG1, 2, "G 系列第一代多功能手柄") \
00547
00548 #define ENUM_SafeguardedStopType_DECLARES \
00549     ENUM_ITEM(None, 0, " 无安全停止") \
00550     ENUM_ITEM(SafeguardedStopIOInput, 1, " 安全停止 (IO 输入)") \
00551     ENUM_ITEM(SafeguardedStop3PE, 2, " 安全停止 (三态开关)") \
00552     ENUM_ITEM(SafeguardedStopOperational, 3, " 安全停止 (操作模式)") \
00553
00554 #define ENUM_RobotEmergencyStopType_DECLARES \
00555     ENUM_ITEM(RobotEmergencyStopNone, 0, " 无紧急停止") \
00556     ENUM_ITEM(RobotEmergencyStopControlBox, 1, " 紧急停止 (控制柜急停)") \
00557     ENUM_ITEM(RobotEmergencyStopTeachPendant, 2, " 紧急停止 (示教器急停)") \
00558     ENUM_ITEM(RobotEmergencyStopHandle, 3, " 紧急停止 (手柄急停)") \
00559     ENUM_ITEM(RobotEmergencyStopEI, 4, " 紧急停止 (固定 IO 急停)") \
00560
00561 #define ENUM_ITEM(c, n, ...) c = n,
00562 enum AuboErrorCodes : int
00563 {
00564     ENUM_AuboErrorCodes_DECLARES

```

```

00565 };
00566
00567 enum class RuntimeState : int
00568 {
00569     ENUM_RuntimeState_DECLARES
00570 };
00571
00572 enum class RobotModeType : int
00573 {
00574     ENUM_RobotModeType_DECLARES
00575 };
00576
00577 enum class AxisModeType : int
00578 {
00579     ENUM_AxisModeType_DECLARES
00580 };
00581
00582 enum class HandleStateType : int
00583 {
00584     ENUM_HandleStateType_DECLARES
00585 };
00586
00587 enum class HandleModeType : int
00588 {
00589     ENUM_HandleModeType_DECLARES
00590 };
00591
00592 /**
00593  * \-chinese 安全状态: \-english Safety Mode
00594  *
00595  */
00596 enum class SafetyModeType : int
00597 {
00598     ENUM_SafetyModeType_DECLARES
00599 };
00600
00601 /**
00602  * \-chinese 操作模式 \-english Operational Mode
00603  */
00604 enum class OperationalModeType : int
00605 {
00606     ENUM_OperationalModeType_DECLARES
00607 };
00608
00609 /**
00610  * \-chinese 机器人控制模式 \-english Robot Control Mode
00611  */
00612 enum class RobotControlModeType : int
00613 {
00614     ENUM_RobotControlModeType_DECLARES
00615 };
00616
00617 /**
00618  * \-chinese 关节伺服模式 \-english Joint Servo Mode
00619  */
00620 enum class JointServoModeType : int
00621 {
00622     ENUM_JointServoModeType_DECLARES
00623 };
00624
00625 /**
00626  * \-chinese 关节状态 \-english Joint State
00627  */
00628 enum class JointStateType : int
00629 {
00630     ENUM_JointStateType_DECLARES
00631 };
00632
00633 /**
00634  * \-chinese 标准输出运行状态 \-english Standard Output Run State
00635  */
00636 enum class StandardOutputRunState : int
00637 {
00638     ENUM_StandardOutputRunState_DECLARES
00639 };
00640
00641 /**
00642  * @brief The StandardInputAction enum
00643  */
00644 enum class StandardInputAction : int
00645 {
00646     ENUM_StandardInputAction_DECLARES
00647 };
00648
00649 enum class SafetyInputAction : int
00650 {
00651     ENUM_SafetyInputAction_DECLARES

```



```

00652 };
00653
00654 enum class SafetyOutputRunState : int
00655 {
00656     ENUM_SafetyOutputRunState_DECLARES
00657 };
00658
00659 enum TaskFrameType
00660 {
00661     ENUM_TaskFrameType_DECLARES
00662 };
00663
00664 enum EnvelopingShape : int
00665 {
00666     ENUM_EnvelopingShape_DECLARES
00667 };
00668
00669 enum PayloadIdentifyMoveAxis : int
00670 {
00671     ENUM_PayloadIdentifyMoveAxis_DECLARES
00672 };
00673
00674 enum TraceLevel
00675 {
00676     ENUM_TraceLevel_DECLARES
00677 };
00678
00679 enum SafeguedStopType : int
00680 {
00681     ENUM_SafeguedStopType_DECLARES
00682 };
00683
00684 enum RobotEmergencyStopType : int
00685 {
00686     ENUM_RobotEmergencyStopType_DECLARES
00687 };
00688 #undef ENUM_ITEM
00689
00690 #define DECL_TO_STRING_FUNC(ENUM)
00691     inline std::string toString(ENUM v)
00692     {
00693         using T = ENUM;
00694         std::string name = #ENUM ".";
00695         ENUM_##ENUM##_DECLARES
00696
00697         return #ENUM ".Unkown";
00698     }
00699     inline std::ostream &operator<<(std::ostream &os, ENUM v)
00700     {
00701         os << toString(v);
00702         return os;
00703     }
00704
00705 #define ENUM_ITEM(c, n, ...) \
00706     if (v == T::c) { \
00707         return name + #c; \
00708     }
00709
00710 DECL_TO_STRING_FUNC(RuntimeState)
00711 DECL_TO_STRING_FUNC(RobotModeType)
00712 DECL_TO_STRING_FUNC(AxisModeType)
00713 DECL_TO_STRING_FUNC(SafetyModeType)
00714 DECL_TO_STRING_FUNC(OperationalModeType)
00715 DECL_TO_STRING_FUNC(RobotControlModeType)
00716 DECL_TO_STRING_FUNC(JointServoModeType)
00717 DECL_TO_STRING_FUNC(JointStateType)
00718 DECL_TO_STRING_FUNC(StandardInputAction)
00719 DECL_TO_STRING_FUNC(StandardOutputRunState)
00720 DECL_TO_STRING_FUNC(SafetyInputAction)
00721 DECL_TO_STRING_FUNC(SafetyOutputRunState)
00722 DECL_TO_STRING_FUNC(TaskFrameType)
00723 DECL_TO_STRING_FUNC(TraceLevel)
00724 DECL_TO_STRING_FUNC(HandleModeType)
00725 DECL_TO_STRING_FUNC(HandleStateType)
00726
00727 #undef ENUM_ITEM
00728
00729 enum class ForceControlState
00730 {
00731     Stopped,
00732     Starting,
00733     Stopping,
00734     Running
00735 };
00736
00737 enum class RefFrameType
00738 {

```

```

00739     None, ///  

00740     Tool, ///  

00741     Path, ///  

00742     Base ///  

00743 };  

00744  

00745 ///  

00746 struct CircleParameters  

00747 {  

00748     std::vector<double> pose_via; ///  

00749     std::vector<double> pose_to; ///  

00750     double a; ///  

00751     double v; ///  

00752     double blend_radius; ///  

00753     double duration; ///  

00754     double helix;  

00755     double spiral;  

00756     double direction;  

00757     int loop_times; ///  

00758 };  

00759  

00760 inline std::ostream &operator<<(std::ostream &os, CircleParameters p)  

00761 {  

00762     return os;  

00763 }  

00764  

00765 struct SpiralParameters  

00766 {  

00767     std::vector<double> frame; ///  

00768     int plane; ///  

00769     double angle; ///  

00770     double spiral; ///  

00771     double helix; ///  

00772 };  

00773  

00774 inline std::ostream &operator<<(std::ostream &os, SpiralParameters p)  

00775 {  

00776     return os;  

00777 }  

00778  

00779 struct Enveloping  

00780 {  

00781     EnvelopingShape shape; ///  

00782     std::vector<double> ep_args; ///  

00783     ///  

00784     ///  

00785     std::string stl_path; ///  

00786 };  

00787  

00788 inline std::ostream &operator<<(std::ostream &os, Enveloping p)  

00789 {  

00790     return os;  

00791 }  

00792  

00793 ///  

00794 ///  

00795 struct TrajConfig  

00796 {  

00797     std::vector<Enveloping> envelopings; ///  

00798     PayloadIdentifyMoveAxis move_axis; ///  

00799     std::vector<double> init_joint; ///  

00800     std::vector<double> upper_joint_bound; ///  

00801     std::vector<double> lower_joint_bound; ///  

00802     std::vector<double> max_velocity; ///  

00803     std::vector<double> max_acceleration; ///  

00804 };  

00805  

00806 inline std::ostream &operator<<(std::ostream &os, TrajConfig p)  

00807 {  

00808     return os;  

00809 }  

00810  

00811 // result with error code  

00812 using ResultWithErrno = std::tuple<std::vector<double>, int>;  

00813 using ResultWithErrno1 = std::tuple<std::vector<std::vector<double>>, int>;

```

```

00814 using ResultWithErrno2 = std::tuple<std::vector<std::string>, int>;
00815 using ResultWithErrno3 = std::tuple<int, int>;
00816
00817 // mass, cog, aom, inertia
00818 using Payload = std::tuple<double, std::vector<double>, std::vector<double>,
00819                             std::vector<double>>;
00820
00821 // force_offset, com, mass, angle
00822 using ForceSensorCalibResult =
00823     std::tuple<std::vector<double>, std::vector<double>, double,
00824               std::vector<double>>;
00825
00826 // force_offset, com, mass, angle error
00827 using ForceSensorCalibResultWithError =
00828     std::tuple<std::vector<double>, std::vector<double>, double,
00829               std::vector<double>, double>;
00830
00831 // \-chinese 动力学模型 m,d,k \-english Dynamics model m, d, k
00832 using DynamicsModel =
00833     std::tuple<std::vector<double>, std::vector<double>, std::vector<double>>;
00834
00835 // double xmin;
00836 // double xmax;
00837 // double ymin;
00838 // double ymax;
00839 // double zmin;
00840 // double zmax;
00841 using Box = std::vector<double>;
00842
00843 // double xcbottom;
00844 // double ycbottom;
00845 // double zcbottom;
00846 // double height;
00847 // double radius;
00848 using Cylinder = std::vector<double>;
00849
00850 // double xc;
00851 // double yc;
00852 // double radius;
00853 using Sphere = std::vector<double>;
00854
00855 struct RobotMsg
00856 {
00857     uint64_t timestamp; ///< \-chinese 时间戳, 即系统时间 \-english Timestamp,
00858                        ///< i.e., system time
00859     TraceLevel level; ///< \-chinese 日志等级 \-english Log level
00860     int code;         ///< \-chinese 错误码 \-english Error code
00861     std::string
00862         source; ///< \-chinese 发送消息的机器人别名 alias \-english Alias of the
00863                ///< robot sending the message
00864                ///< \-chinese 可在 /root/arcs_ws/config/aubo_control.conf
00865                ///< \-english Can be found in
00866                ///< /root/arcs_ws/config/aubo_control.conf
00867                ///< \-chinese 配置文件中查到机器人的 alias \-english The robot's
00868                ///< alias can be found in the configuration file
00869                ///< /root/arcs_ws/config/aubo_control.conf
00870     std::vector<std::string> args; ///< \-chinese 机器人参数 \-english Robot parameters
00871 };
00872 using RobotMsgVector = std::vector<RobotMsg>;
00873
00874 struct GripperStatus
00875 {
00876     std::string name;
00877     bool is_connected;
00878     bool is_enabled;
00879     double position;
00880     double velocity;
00881     double force;
00882     double angle;
00883     double r_velocity;
00884     double torque;
00885     bool object_detection;
00886     bool motion_state;
00887     double voltage;
00888     double temperature;
00889     int status_code;
00890 };
00891 using GripperStatusVector = std::vector<GripperStatus>;
00892
00893 ///< \-chinese RTDE 菜单 \-english RTDE menu
00894 struct RtdeRecipe
00895 {
00896     bool to_server; ///< \-chinese 输入/输出 \-english Input/Output
00897     int channel;    ///< \-chinese 通道 \-english Channel
00898     double frequency; ///< \-chinese 更新频率 \-english Update frequency
00899     int trigger;     ///< \-chinese 触发方式 (该功能暂未实现): 0 - 周期; 1 - 变化
00900                    ///< \-english Trigger method (this feature is not yet

```



```

00901         ///< implemented): 0 - Periodic; 1 - Change
00902     std::vector<std::string> segments; ///< \-chinese 字段列表 \-english segment list
00903 };
00904
00905 ///< \-chinese 异常类型 \-english Error type
00906 enum error_type
00907 {
00908     parse_error = -32700, ///< \-chinese 解析错误 \-english Parse error
00909     invalid_request = -32600, ///< \-chinese 无效请求 \-english Invalid request
00910     method_not_found = -32601, ///< \-chinese 方法未找到 \-english Method not found
00911     invalid_params = -32602, ///< \-chinese 无效参数 \-english Invalid parameters
00912     internal_error = -32603, ///< \-chinese 内部错误 \-english Internal error
00913     server_error = -32500, ///< \-chinese 服务器错误 \-english Server error
00914     invalid = -32400 ///< \-chinese 无效 \-english Invalid
00915 };
00916
00917 ///< \-chinese 异常码 \-english Exception code
00918 enum ExceptionCode
00919 {
00920     EC_DISCONNECTED = -1, ///< \-chinese 断开连接 \-english Disconnected
00921     EC_NOT_LOGINED = -2, ///< \-chinese 未登录 \-english Not logged in
00922     EC_INVALID_SOCKET = -3, ///< \-chinese 无效套接字 \-english Invalid socket
00923     EC_REQUEST_BUSY = -4, ///< \-chinese 请求繁忙 \-english Request busy
00924     EC_SEND_FAILED = -5, ///< \-chinese 发送失败 \-english Send failed
00925     EC_RECV_TIMEOUT = -6, ///< \-chinese 接收超时 \-english Receive timeout
00926     EC_RECV_ERROR = -7, ///< \-chinese 接收错误 \-english Receive error
00927     EC_PARSE_ERROR = -8, ///< \-chinese 解析错误 \-english Parse error
00928     EC_INVALID_REQUEST = -9, ///< \-chinese 无效请求 \-english Invalid request
00929     EC_METHOD_NOT_FOUND = -10, ///< \-chinese 方法未找到 \-english Method not found
00930     EC_INVALID_PARAMS = -11, ///< \-chinese 无效参数 \-english Invalid parameters
00931     EC_INTERNAL_ERROR = -12, ///< \-chinese 内部错误 \-english Internal error
00932     EC_SERVER_ERROR = -13, ///< \-chinese 服务器错误 \-english Server error
00933     EC_INVALID = -14 ///< \-chinese 无效 \-english Invalid
00934 };
00935
00936 ///< \-chinese 自定义异常类 AuboException \-english Custom exception class
00937 ///< AuboException
00938 class AuboException : public std::exception
00939 {
00940 public:
00941     AuboException(int code, const std::string &prefix,
00942                  const std::string &message) noexcept
00943         : code_(code), message_(prefix + "-" + message)
00944     {
00945     }
00946
00947     AuboException(int code, const std::string &message) noexcept
00948         : code_(code), message_(message)
00949     {
00950     }
00951
00952     error_type type() const
00953     {
00954         if (code_ >= -32603 && code_ <= -32600) {
00955             return static_cast<error_type>(code_);
00956         } else if (code_ >= -32099 && code_ <= -32000) {
00957             return server_error;
00958         } else if (code_ == -32700) {
00959             return parse_error;
00960         }
00961         return invalid;
00962     }
00963
00964     int code() const { return code_; }
00965     const char *what() const noexcept override { return message_.c_str(); }
00966
00967 private:
00968     int code_; ///< \-chinese 异常码 \-english Exception code
00969     std::string message_; ///< \-chinese 异常消息 \-chinese Exception message
00970 };
00971
00972 inline const char *returnValue2Str(int retval)
00973 {
00974     static const char *retval_str[] = {
00975 #define ENUM_ITEM(n, v, s) s,
00976     ENUM_AuboErrorCodes_DECLARES
00977 #undef ENUM_ITEM
00978     };
00979
00980     enum arcs_index
00981     {
00982 #define ENUM_ITEM(n, v, s) n##_INDEX,
00983     ENUM_AuboErrorCodes_DECLARES
00984 #undef ENUM_ITEM
00985     };
00986
00987     int index = -1;

```

```
00988
00989 #define ENUM_ITEM(n, v, s) \
00990     if (abs(retval) == v) \
00991         index = n##_INDEX;
00992     ENUM_AuboErrorCodes_DECLARES
00993 #undef ENUM_ITEM
00994
00995     if (index == -1)
00996     {
00997         index = AUBO_ERR_UNKOWN_INDEX;
00998     }
00999
01000     return retval_str[(unsigned)index];
01001 }
01002
01003 } // namespace common_interface
01004 } // namespace arcs
01005 #endif
01006
01007 // clang-format on
01008
01009 #if defined ENABLE_JSON_TYPES
01010 #include "bindings/jsonrpc/json_types.h"
01011 #endif
```

Index

- `_D`
 - `error_stack.h`, 608, 609
 - `_PH1`
 - `error_stack.h`, 609
 - `_PH2`
 - `error_stack.h`, 609
 - `_PH3`
 - `error_stack.h`, 609
 - `_PH4`
 - `error_stack.h`, 609
 - `~AuboApi`
 - `arcs::common_interface::AuboApi`, 518
 - `~AxisInterface`
 - `arcs::common_interface::AxisInterface`, 523
 - `~ForceControl`
 - `arcs::common_interface::ForceControl`, 528
 - `~GripperInterface`
 - `arcs::common_interface::GripperInterface`, 530
 - `~IoControl`
 - `arcs::common_interface::IoControl`, 538
 - `~Math`
 - `arcs::common_interface::Math`, 540
 - `~MotionControl`
 - `arcs::common_interface::MotionControl`, 545
 - `~RegisterControl`
 - `arcs::common_interface::RegisterControl`, 550
 - `~RobotAlgorithm`
 - `arcs::common_interface::RobotAlgorithm`, 554
 - `~RobotConfig`
 - `arcs::common_interface::RobotConfig`, 558
 - `~RobotInterface`
 - `arcs::common_interface::RobotInterface`, 560
 - `~RobotManage`
 - `arcs::common_interface::RobotManage`, 562
 - `~RobotState`
 - `arcs::common_interface::RobotState`, 574
 - `~RuntimeMachine`
 - `arcs::common_interface::RuntimeMachine`, 579
 - `~Serial`
 - `arcs::common_interface::Serial`, 580
 - `~Socket`
 - `arcs::common_interface::Socket`, 581
 - `~SyncMove`
 - `arcs::common_interface::SyncMove`, 585
 - `~SystemInfo`
 - `arcs::common_interface::SystemInfo`, 585
 - `~Trace`
 - `arcs::common_interface::Trace`, 586
- a
 - `arcs::common_interface::CircleParameters`, 524
- abort
 - `RuntimeMachine` (运行时管理), 445
- addCollisionBox
 - `RobotAlgorithm` (机器人算法工具), 302
- addCrossConnection
 - `IoControl` (IO 输入输出控制), 145
- addFixedModbusGroup
 - `RegisterControl` (寄存器操作), 62
- addGroupInput
 - `IoControl` (IO 输入输出控制), 146
- addGroupOutput
 - `IoControl` (IO 输入输出控制), 147
- addInt32RegEncoder
 - `RegisterControl` (寄存器操作), 63
- addModbusEncoder
 - `RegisterControl` (寄存器操作), 64
- alarm
 - `Trace` (日志与弹窗), 494
- allow_manual_high_speed
 - `arcs::common_interface::RobotSafetyParameterRange`, 566
- angle
 - `arcs::common_interface::GripperStatus`, 533
 - `arcs::common_interface::SpiralParameters`, 582
- arbitraryResume
 - `RuntimeMachine` (运行时管理), 445
- arcs, 501
- `arcs::common_interface`, 501
 - `AuboApiPtr`, 504
 - `AuboErrorCodes`, 507
 - `AxisInterfacePtr`, 504
 - `AxisModeType`, 507
 - `Base`, 510
 - `Box`, 504
 - `Cylinder`, 504
 - `DynamicsModel`, 504
 - `EC_DISCONNECTED`, 508
 - `EC_INTERNAL_ERROR`, 508
 - `EC_INVALID_SOCKET`, 508
 - `EC_INVALID`, 508
 - `EC_INVALID_PARAMS`, 508
 - `EC_INVALID_REQUEST`, 508
 - `EC_METHOD_NOT_FOUND`, 508
 - `EC_NOT_LOGINED`, 508
 - `EC_PARSE_ERROR`, 508
 - `EC_RECV_ERROR`, 508
 - `EC_RECV_TIMEOUT`, 508
 - `EC_REQUEST_BUSY`, 508

- EC_SEND_FAILED, 508
- EC_SERVER_ERROR, 508
- ENUM_AuboErrorCodes_DECLARES, 507
- ENUM_AxisModeType_DECLARES, 507
- ENUM_EnvelopingShape_DECLARES, 507
- ENUM_HandleModeType_DECLARES, 509
- ENUM_HandleStateType_DECLARES, 509
- ENUM_JointServoModeType_DECLARES, 509
- ENUM_JointStateType_DECLARES, 509
- ENUM_OperationalModeType_DECLARES, 509
- ENUM_PayloadIdentifyMoveAxis_DECLARES, 510
- ENUM_RobotControlModeType_DECLARES, 511
- ENUM_RobotEmergencyStopType_DECLARES, 511
- ENUM_RobotModeType_DECLARES, 511
- ENUM_RuntimeState_DECLARES, 511
- ENUM_SafeguardedStopType_DECLARES, 511
- ENUM_SafetyInputAction_DECLARES, 512
- ENUM_SafetyModeType_DECLARES, 512
- ENUM_SafetyOutputRunState_DECLARES, 512
- ENUM_StandardInputAction_DECLARES, 512
- ENUM_StandardOutputRunState_DECLARES, 512
- ENUM_TaskFrameType_DECLARES, 513
- ENUM_TraceLevel_DECLARES, 513
- EnvelopingShape, 507
- error_type, 507
- ExceptionCode, 508
- ForceControlPtr, 504
- ForceControlState, 508
- ForceSensorCalibResult, 504
- ForceSensorCalibResultWithError, 504
- GripperInterfacePtr, 504
- GripperStatusVector, 504
- HandleModeType, 509
- HandleStateType, 509
- internal_error, 508
- invalid, 508
- invalid_params, 508
- invalid_request, 508
- IoControlPtr, 504
- JointServoModeType, 509
- JointStateType, 509
- MathPtr, 505
- method_not_found, 508
- MotionControlPtr, 505
- None, 510
- OperationalModeType, 509
- operator<<, 513, 514
- parse_error, 508
- Path, 510
- PathBuffer_CubicSpline, 510
- PathBuffer_JointBSpline, 510
- PathBuffer_JointBSplineC, 510
- PathBuffer_JointSpline, 510
- PathBuffer_JointSplineC, 510
- PathBuffer_TOPPRA, 510
- PathBufferType, 510
- Payload, 505
- PayloadIdentifyMoveAxis, 510
- RefFrameType, 510
- RegisterControlPtr, 505
- ResultWithErrno, 505
- ResultWithErrno1, 505
- ResultWithErrno2, 505
- ResultWithErrno3, 505
- returnValue2Str, 514
- RobotAlgorithmPtr, 505
- RobotConfigPtr, 505
- RobotControlModeType, 510
- RobotEmergencyStopType, 511
- RobotInterfacePtr, 505
- RobotManagePtr, 505
- RobotModeType, 511
- RobotMsgVector, 506
- RobotStatePtr, 506
- Running, 508
- RuntimeMachinePtr, 506
- RuntimeState, 511
- SafeguardedStopType, 511
- SafetyInputAction, 511
- SafetyModeType, 512
- SafetyOutputRunState, 512
- SerialPtr, 506
- server_error, 508
- SocketPtr, 506
- Sphere, 506
- StandardInputAction, 512
- StandardOutputRunState, 512
- Starting, 508
- Stopped, 508
- Stopping, 508
- SyncMovePtr, 506
- SystemInfoPtr, 506
- TaskFrameType, 512
- TaskSet, 506
- Tool, 510
- TraceLevel, 513
- TracePtr, 506
- Vector3d, 506
- Vector3f, 506
- Vector4d, 507
- Vector4f, 507
- Vector6f, 507
- arcs::common_interface::AuboApi, 517
 - ~AuboApi, 518
 - AuboApi, 518
 - d_, 518
- arcs::common_interface::AuboException, 518
 - AuboException, 519, 520
 - code, 520
 - code_, 521

- message_, 521
- type, 520
- what, 520
- arcs::common_interface::AxisInterface, 521
 - ~AxisInterface, 523
 - AxisInterface, 523
 - d_, 523
- arcs::common_interface::CircleParameters, 523
 - a, 524
 - blend_radius, 524
 - direction, 524
 - duration, 524
 - helix, 524
 - loop_times, 524
 - pose_to, 524
 - pose_via, 525
 - spiral, 525
 - v, 525
- arcs::common_interface::Enveloping, 525
 - ep_args, 526
 - shape, 526
 - stl_path, 526
- arcs::common_interface::ForceControl, 526
 - ~ForceControl, 528
 - d_, 529
 - ForceControl, 528
- arcs::common_interface::GripperInterface, 529
 - ~GripperInterface, 530
 - d_, 532
 - gripperGetAngle, 530
 - gripperGetForce, 530
 - gripperGetMotionState, 530
 - gripperGetMountPose, 530
 - gripperGetObjectDetection, 531
 - gripperGetPosition, 531
 - gripperGetRVelocity, 531
 - gripperGetSoftwareVersion, 531
 - gripperGetTemperature, 531
 - gripperGetTorque, 531
 - gripperGetVelocity, 531
 - gripperGetVoltage, 531
 - GripperInterface, 530
 - gripperSetAngle, 531
 - gripperSetForce, 531
 - gripperSetRVelocity, 532
 - gripperSetTorque, 532
 - gripperSetVelocity, 532
- arcs::common_interface::GripperStatus, 532
 - angle, 533
 - force, 533
 - is_connected, 533
 - is_enabled, 533
 - motion_state, 533
 - name, 533
 - object_detection, 533
 - position, 533
 - r_velocity, 534
 - status_code, 534
 - temperature, 534
 - torque, 534
 - velocity, 534
 - voltage, 534
- arcs::common_interface::IoControl, 534
 - ~IoControl, 538
 - d_, 538
 - IoControl, 538
- arcs::common_interface::Math, 538
 - ~Math, 540
 - d_, 540
 - Math, 540
- arcs::common_interface::MotionControl, 540
 - ~MotionControl, 545
 - d_, 547
 - getConveyorTrackNextItem, 545
 - isConveyorTrackExceed, 546
 - MotionControl, 545
 - servoCartesianWithAxisGroup, 547
 - servoJointWithAxisGroup, 547
- arcs::common_interface::RegisterControl, 547
 - ~RegisterControl, 550
 - d_, 551
 - modbusSetOutputSignalWithTimeout, 551
 - RegisterControl, 550
- arcs::common_interface::RobotAlgorithm, 552
 - ~RobotAlgorithm, 554
 - d_, 555
 - pathMovej, 554
 - RobotAlgorithm, 554
- arcs::common_interface::RobotConfig, 555
 - ~RobotConfig, 558
 - d_, 559
 - disableAxisGroup, 559
 - enableAxisGroup, 559
 - getRobotBaseParent, 559
 - RobotConfig, 558
- arcs::common_interface::RobotInterface, 559
 - ~RobotInterface, 560
 - d_, 560
 - RobotInterface, 560
- arcs::common_interface::RobotManage, 560
 - ~RobotManage, 562
 - d_, 562
 - RobotManage, 562
- arcs::common_interface::RobotMsg, 562
 - args, 563
 - code, 563
 - level, 563
 - source, 563
 - timestamp, 563
- arcs::common_interface::RobotSafetyParameterRange, 564
 - allow_manual_high_speed, 566
 - crc32, 566
 - cubic, 566
 - elbow_force, 566
 - elbow_speed, 566

- joint_torque, 566
- momentum, 566
- orig, 566
- params, 566
- plane, 567
- planes, 567
- power, 567
- qdmx, 567
- qmax, 567
- qmin, 567
- reduced_entry_distance, 567
- reduced_entry_time, 567
- restrict_elbow, 567
- RobotSafetyParameterRange, 566
- safety_home, 567
- safety_input_auto_safeguard_reset, 568
- safety_input_auto_safeguard_stop, 568
- safety_input_emergency_stop, 568
- safety_input_handguide, 568
- safety_input_operational_mode, 568
- safety_input_reduced_mode, 568
- safety_input_safeguard_reset, 568
- safety_input_safeguard_stop, 568
- safety_input_three_position_switch, 568
- safety_output_emergency_stop, 568
- safety_output_not_emergency_stop, 569
- safety_output_not_reduced_mode, 569
- safety_output_reduced_mode, 569
- safety_output_robot_moving, 569
- safety_output_robot_not_stopping, 569
- safety_output_robot_steady, 569
- safety_output_safe_home, 569
- safety_output_safeguard_stop, 569
- size, 569
- stop_distance, 569
- stop_time, 570
- tcp_force, 570
- tcp_speed, 570
- tool_azimuth, 570
- tool_deviation, 570
- tool_inclination, 570
- tool_orientation, 570
- tools, 570
- tp_3pe_for_handguide, 570
- trigger_planes, 571
- arcs::common_interface::RobotState, 571
 - ~RobotState, 574
 - d_, 575
 - getBaseForceSensor, 574
 - RobotState, 574
- arcs::common_interface::RtdeRecipe, 575
 - chanel, 576
 - frequency, 576
 - segments, 576
 - to_server, 576
 - trigger, 576
- arcs::common_interface::RuntimeMachine, 576
 - ~RuntimeMachine, 579
 - d_, 579
 - getExecutionStatus1, 579
 - getRuntimeState, 579
 - RuntimeMachine, 579
- arcs::common_interface::Serial, 579
 - ~Serial, 580
 - d_, 580
 - Serial, 580
- arcs::common_interface::Socket, 580
 - ~Socket, 581
 - d_, 581
 - Socket, 581
- arcs::common_interface::SpiralParameters, 582
 - angle, 582
 - frame, 582
 - helix, 582
 - plane, 582
 - spiral, 583
- arcs::common_interface::SyncMove, 583
 - ~SyncMove, 585
 - d_, 585
 - SyncMove, 585
- arcs::common_interface::SystemInfo, 585
 - ~SystemInfo, 585
 - d_, 586
 - SystemInfo, 585
- arcs::common_interface::Trace, 586
 - ~Trace, 586
 - d_, 587
 - notify, 586
 - Trace, 586
- arcs::common_interface::TrajConfig, 587
 - envelopings, 588
 - init_joint, 588
 - lower_joint_bound, 588
 - max_acceleration, 588
 - max_velocity, 588
 - move_axis, 588
 - upper_joint_bound, 588
- arcs::common_interface::VibrationRecalibrationParameter, 589
 - cog, 589
 - inertia, 589
 - mass, 589
 - points, 590
 - stiff_param, 590
 - stiff_section, 590
- arcs::common_interface::WObjectData, 590
 - attach_frame, 591
 - obj_coord, 591
 - remote_tool, 591
 - user_coord, 591
- arcs::error_stack, 514
 - ARCS_ERROR_CODES, 514
 - codeCompose, 514
 - dump, 514
 - errorCode2Str, 515
 - ErrorCodes, 514

- mod, 515
- str2ErrorCode, 515
- ARCS_ERROR_CODES
 - arcs::error_stack, 514
 - error_stack.h, 609
- args
 - arcs::common_interface::RobotMsg, 563
- attach_frame
 - arcs::common_interface::WObjectData, 591
- attachRobotBaseTo
 - RobotConfig (机器人配置管理), 330
- attachWeldingGun
 - RobotConfig (机器人配置管理), 330
- Aubo SDK, 9
- AUBO_SDK_HAL_ERROR_H
 - error_stack.h, 609
- AUBO_SDK_RTM_ERROR_H
 - error_stack.h, 610
- AUBO_SDK_SYSTEM_ERROR_H
 - error_stack.h, 610
- AuboApi
 - arcs::common_interface::AuboApi, 518
- AuboApi (主入口), 25
 - getAxisInterface, 26
 - getAxisNames, 26
 - getGripperInterface, 26
 - getMath, 26
 - getRegisterControl, 26
 - getRobotInterface, 27
 - getRobotNames, 27
 - getRuntimeMachine, 27
 - getSerial, 28
 - getSocket, 28
 - getSyncMove, 28
 - getSystemInfo, 28
 - getTrace, 29
- AuboApiPtr
 - arcs::common_interface, 504
- AuboErrorCodes
 - arcs::common_interface, 507
- AuboException
 - arcs::common_interface::AuboException, 519, 520
- axisGroupAdd
 - SyncMove (同步运动控制和轴组管理), 479
- axisGroupAddAxis
 - SyncMove (同步运动控制和轴组管理), 480
- axisGroupDelete
 - SyncMove (同步运动控制和轴组管理), 480
- axisGroupGetActualPositions
 - SyncMove (同步运动控制和轴组管理), 481
- axisGroupGetAxisIndex
 - SyncMove (同步运动控制和轴组管理), 481
- axisGroupGetAxisName
 - SyncMove (同步运动控制和轴组管理), 481
- axisGroupGetTargetPositions
 - SyncMove (同步运动控制和轴组管理), 482
- axisGroupMoveJoint
 - SyncMove (同步运动控制和轴组管理), 482
- axisGroupOffsetPositions
 - SyncMove (同步运动控制和轴组管理), 482
- axisGroupSpeedJoint
 - SyncMove (同步运动控制和轴组管理), 483
- axisGroupUpdateAxis
 - SyncMove (同步运动控制和轴组管理), 483
- AxisInterface
 - arcs::common_interface::AxisInterface, 523
- AxisInterface (外部轴), 29
 - clearAxisError, 31
 - enableExtAxis, 31
 - followAnotherAxis, 31
 - getAxisModeType, 31
 - getErrorCode, 31
 - getExtAxisAcceleration, 32
 - getExtAxisBusCurrent, 32
 - getExtAxisBusVoltage, 32
 - getExtAxisCurrent, 32
 - getExtAxisMaxAcceleration, 32
 - getExtAxisMaxAccelerationLimit, 32
 - getExtAxisMaxPosition, 32
 - getExtAxisMaxPositionLimit, 33
 - getExtAxisMaxVelocity, 33
 - getExtAxisMaxVelocityLimit, 33
 - getExtAxisMountingPose, 33
 - getExtAxisPose, 33
 - getExtAxisPosition, 33
 - getExtAxisTemperature, 33
 - getExtAxisType, 34
 - getExtAxisVelocity, 34
 - getExtMinPosition, 34
 - getExtMinPositionLimit, 34
 - getReductionRatio, 34
 - moveExtJoint, 34
 - poweroffExtAxis, 35
 - poweronExtAxis, 35
 - setExtAxisAcceleration, 35
 - setExtAxisMaxAccelerationLimit, 35
 - setExtAxisMaxPositionLimit, 35
 - setExtAxisMaxVelocityLimit, 36
 - setExtAxisMountingPose, 36
 - setExtAxisType, 36
 - setExtAxisVelocity, 36
 - setExtAxisZero, 37
 - setExtMinPositionLimit, 37
 - setReductionRatio, 37
 - speedExtJoint, 37
 - stopExtJoint, 38
 - stopFollowAnotherAxis, 38
- AxisInterfacePtr
 - arcs::common_interface, 504
- AxisModeType
 - arcs::common_interface, 507
- backdrive
 - RobotManage (机器人生命周期管理), 378
- Base
 - arcs::common_interface, 510

- blend_radius
 - arcs::common_interface::CircleParameters, 524
- Box
 - arcs::common_interface, 504
- calcJacobian
 - RobotAlgorithm (机器人算法工具), 302
- calcSafetyParametersChecksum
 - RobotConfig (机器人配置管理), 331
- calculateCircleFourthPoint
 - Math (数学工具), 46
- calibrateCoordinate
 - Math (数学工具), 47
- calibrateTcpForceSensor
 - RobotAlgorithm (机器人算法工具), 303
- calibrateTcpForceSensor2
 - RobotAlgorithm (机器人算法工具), 304
- calibrateTcpForceSensor3
 - RobotAlgorithm (机器人算法工具), 305
- calibVibrationParams
 - RobotAlgorithm (机器人算法工具), 305
- calibVibrationParams1
 - RobotAlgorithm (机器人算法工具), 306
- calibWorkpieceCoordinatePara
 - RobotAlgorithm (机器人算法工具), 307
- CARTESIAN_DOF
 - type_def.h, 1072
- chanel
 - arcs::common_interface::RtdeRecipe, 576
- changePoseWithXYRef
 - Math (数学工具), 48
- clearAxisError
 - AxisInterface (外部轴), 31
- clearBreakPoints
 - RuntimeMachine (运行时管理), 446
- clearCrossConnections
 - IoControl (IO 输入输出控制), 148
- clearNamedVariable
 - RegisterControl (寄存器操作), 64
- clearPath
 - MotionControl (运动规划与控制), 217
- clearPreloadPrograms
 - RuntimeMachine (运行时管理), 446
- code
 - arcs::common_interface::AuboException, 520
 - arcs::common_interface::RobotMsg, 563
- code_
 - arcs::common_interface::AuboException, 521
- codeCompose
 - arcs::error_stack, 514
- cog
 - arcs::common_interface::VibrationRecalibrationPara, 589
- confirmSafetyParameters
 - RobotConfig (机器人配置管理), 331
- conveyorTrackCircle
 - MotionControl (运动规划与控制), 218
- conveyorTrackClearItems
 - MotionControl (运动规划与控制), 218
- conveyorTrackCreatItem
 - MotionControl (运动规划与控制), 219
- conveyorTrackLine
 - MotionControl (运动规划与控制), 220
- conveyorTrackStop
 - MotionControl (运动规划与控制), 221
- conveyorTrackSwitch
 - MotionControl (运动规划与控制), 221
- crc32
 - arcs::common_interface::RobotSafetyParameterRange, 566
- cubic
 - arcs::common_interface::RobotSafetyParameterRange, 566
- Cylinder
 - arcs::common_interface, 504
- d_
 - arcs::common_interface::AuboApi, 518
 - arcs::common_interface::AxisInterface, 523
 - arcs::common_interface::ForceControl, 529
 - arcs::common_interface::GripperInterface, 532
 - arcs::common_interface::IoControl, 538
 - arcs::common_interface::Math, 540
 - arcs::common_interface::MotionControl, 547
 - arcs::common_interface::RegisterControl, 551
 - arcs::common_interface::RobotAlgorithm, 555
 - arcs::common_interface::RobotConfig, 559
 - arcs::common_interface::RobotInterface, 560
 - arcs::common_interface::RobotManage, 562
 - arcs::common_interface::RobotState, 575
 - arcs::common_interface::RuntimeMachine, 579
 - arcs::common_interface::Serial, 580
 - arcs::common_interface::Socket, 581
 - arcs::common_interface::SyncMove, 585
 - arcs::common_interface::SystemInfo, 586
 - arcs::common_interface::Trace, 587
- DECL_TO_STRING_FUNC
 - type_def.h, 1072
- deleteCrossConnection
 - IoControl (IO 输入输出控制), 148
- deleteGroupInput
 - IoControl (IO 输入输出控制), 149
- deleteGroupOutput
 - IoControl (IO 输入输出控制), 150
- deleteTask
 - RuntimeMachine (运行时管理), 446
- deleteVirtualEncoder
 - RegisterControl (寄存器操作), 65
- deltaPoseAdd
 - Math (数学工具), 48
- deltaPoseTrans
 - Math (数学工具), 49
- detachTask
 - RuntimeMachine (运行时管理), 447
- direction
 - arcs::common_interface::CircleParameters, 524
- disableAxisGroup
 - arcs::common_interface::RobotConfig, 559

- disableJointSoftServo
 - MotionControl (运动规划与控制), 222
- disbaleVibrationSuppress
 - MotionControl (运动规划与控制), 222
- doc 目录参考, 498
- doc/sdk_overview.md, 593
- dump
 - arcs::error_stack, 514
- duration
 - arcs::common_interface::CircleParameters, 524
- DynamicsModel
 - arcs::common_interface, 504
- EC_DISCONNECTED
 - arcs::common_interface, 508
- EC_INTERNAL_ERROR
 - arcs::common_interface, 508
- EC_INVALID_SOCKET
 - arcs::common_interface, 508
- EC_INVALID
 - arcs::common_interface, 508
- EC_INVALID_PARAMS
 - arcs::common_interface, 508
- EC_INVALID_REQUEST
 - arcs::common_interface, 508
- EC_METHOD_NOT_FOUND
 - arcs::common_interface, 508
- EC_NOT_LOGINED
 - arcs::common_interface, 508
- EC_PARSE_ERROR
 - arcs::common_interface, 508
- EC_RECV_ERROR
 - arcs::common_interface, 508
- EC_RECV_TIMEOUT
 - arcs::common_interface, 508
- EC_REQUEST_BUSY
 - arcs::common_interface, 508
- EC_SEND_FAILED
 - arcs::common_interface, 508
- EC_SERVER_ERROR
 - arcs::common_interface, 508
- elbow_force
 - arcs::common_interface::RobotSafetyParameterRange, 566
- elbow_speed
 - arcs::common_interface::RobotSafetyParameterRange, 566
- enableAxisGroup
 - arcs::common_interface::RobotConfig, 559
- enableEndCollisionCheck
 - RobotConfig (机器人配置管理), 332
- enableExtAxis
 - AxisInterface (外部轴), 31
- enableJointSoftServo
 - MotionControl (运动规划与控制), 223
- enableVibrationSuppress
 - MotionControl (运动规划与控制), 224
- enterCritical
 - RuntimeMachine (运行时管理), 447
- ENUM_AuboErrorCodes_DECLARES
 - arcs::common_interface, 507
 - type_def.h, 1072
- ENUM_AxisModeType_DECLARES
 - arcs::common_interface, 507
 - type_def.h, 1072
- ENUM_EnvelopingShape_DECLARES
 - arcs::common_interface, 507
 - type_def.h, 1072
- ENUM_HandleModeType_DECLARES
 - arcs::common_interface, 509
 - type_def.h, 1073
- ENUM_HandleStateType_DECLARES
 - arcs::common_interface, 509
 - type_def.h, 1073
- ENUM_ITEM
 - type_def.h, 1073, 1074
- ENUM_JointServoModeType_DECLARES
 - arcs::common_interface, 509
 - type_def.h, 1074
- ENUM_JointStateType_DECLARES
 - arcs::common_interface, 509
 - type_def.h, 1074
- ENUM_OperationalModeType_DECLARES
 - arcs::common_interface, 509
 - type_def.h, 1074
- ENUM_PayloadIdentifyMoveAxis_DECLARES
 - arcs::common_interface, 510
 - type_def.h, 1074
- ENUM_RobotControlModeType_DECLARES
 - arcs::common_interface, 511
 - type_def.h, 1075
- ENUM_RobotEmergencyStopType_DECLARES
 - arcs::common_interface, 511
 - type_def.h, 1075
- ENUM_RobotModeType_DECLARES
 - arcs::common_interface, 511
 - type_def.h, 1075
- ENUM_RuntimeState_DECLARES
 - arcs::common_interface, 511
 - type_def.h, 1075
- ENUM_SafeguardedStopType_DECLARES
 - arcs::common_interface, 511
 - type_def.h, 1075
- ENUM_SafetyInputAction_DECLARES
 - arcs::common_interface, 512
 - type_def.h, 1076
- ENUM_SafetyModeType_DECLARES
 - arcs::common_interface, 512
 - type_def.h, 1076
- ENUM_SafetyOutputRunState_DECLARES
 - arcs::common_interface, 512
 - type_def.h, 1076
- ENUM_StandardInputAction_DECLARES
 - arcs::common_interface, 512
 - type_def.h, 1076
- ENUM_StandardOutputRunState_DECLARES
 - arcs::common_interface, 512

- type_def.h, 1077
- ENUM_TaskFrameType_DECLARES
 - arcs::common_interface, 513
 - type_def.h, 1077
- ENUM_TraceLevel_DECLARES
 - arcs::common_interface, 513
 - type_def.h, 1077
- envelopings
 - arcs::common_interface::TrajConfig, 588
- EnvelopingShape
 - arcs::common_interface, 507
- ep_args
 - arcs::common_interface::Enveloping, 526
- error_stack.h
 - _D, 608, 609
 - _PH1_, 609
 - _PH2_, 609
 - _PH3_, 609
 - _PH4_, 609
 - ARCS_ERROR_CODES, 609
 - AUBO_SDK_HAL_ERROR_H, 609
 - AUBO_SDK_RTM_ERROR_H, 610
 - AUBO_SDK_SYSTEM_ERROR_H, 610
 - EX_PEDSTRAL_ERRORS, 610
 - EX_TOOL_ERRORS, 610
 - EXT_AXIS_ERRORS, 610
 - HAL_ERRORS, 610
 - HARDWARE_INTERFACE_ERRORS, 610
 - JOINT_ERRORS, 610
 - JOINT_EX_ERRORS, 611
 - PEDSTRAL_ERRORS, 611
 - RTM_ERRORS, 611
 - SAFETY_INTERFACE_BOARD_ERRORS, 611
 - SYSTEM_ERRORS, 611
 - TOOL_ERRORS, 611
- error_type
 - arcs::common_interface, 507
- errorCode2Str
 - arcs::error_stack, 515
- ErrorCodes
 - arcs::error_stack, 514
- EX_PEDSTRAL_ERRORS
 - error_stack.h, 610
 - hal_error.h, 614
- EX_TOOL_ERRORS
 - error_stack.h, 610
 - hal_error.h, 614
- ExceptionCode
 - arcs::common_interface, 508
- exitCritical
 - RuntimeMachine (运行时管理), 448
- exitHandguideMode
 - RobotManage (机器人生命周期管理), 379
- EXT_AXIS_ERRORS
 - error_stack.h, 610
 - hal_error.h, 615
- fcCalDynamicModel
 - ForceControl(力控模块), 112
- fcDisable
 - ForceControl(力控模块), 112
- fcEnable
 - ForceControl(力控模块), 113
- fcSetSensorLimits
 - ForceControl(力控模块), 113
- fcSetSensorThresholds
 - ForceControl(力控模块), 114
- firmwareUpdate
 - RobotConfig (机器人配置管理), 333
- followAnotherAxis
 - AxisInterface (外部轴), 31
- followJoint
 - MotionControl (运动规划与控制), 225
- followLine
 - MotionControl (运动规划与控制), 225
- force
 - arcs::common_interface::GripperStatus, 533
- ForceControl
 - arcs::common_interface::ForceControl, 528
- ForceControl(力控模块), 109
 - fcCalDynamicModel, 112
 - fcDisable, 112
 - fcEnable, 113
 - fcSetSensorLimits, 113
 - fcSetSensorThresholds, 114
 - getActualJointPositionsHistory, 115
 - getDynamicModel, 115
 - getFcSensorLimits, 116
 - getFcSensorThresholds, 116
 - isCondFullfiled, 116
 - isFcEnabled, 117
 - isSoftFloatEnabled, 118
 - resetDamping, 118
 - resetLpFilter, 119
 - setCondActive, 119
 - setCondAdvanced, 120
 - setCondCylinder, 120
 - setCondDistance, 121
 - setCondForce, 122
 - setCondOrient, 123
 - setCondPlane, 124
 - setCondSphere, 124
 - setCondTcpSpeed, 125
 - setDamping, 126
 - setDynamicModel, 126
 - setDynamicModel1, 127
 - setDynamicModelContact, 128
 - setDynamicModelInsert, 129
 - setDynamicModelSearch, 129
 - setLpFilter, 130
 - setSoftFloatParams, 131
 - setSupvForce, 131
 - setSupvOrient, 132
 - setSupvPosBox, 133
 - setSupvPosCylinder, 133
 - setSupvPosSphere, 134

- setSupvReoriSpeed, 134
- setSupvTcpSpeed, 135
- setTargetForce, 136
- softFloatDisable, 137
- softFloatEnable, 137
- speedChangeDisable, 138
- speedChangeEnable, 138
- speedChangeTune, 139
- toolContact, 139
- ForceControlPtr
 - arcs::common_interface, 504
- ForceControlState
 - arcs::common_interface, 508
- ForceSensorCalibResult
 - arcs::common_interface, 504
- ForceSensorCalibResultWithError
 - arcs::common_interface, 504
- forceTrans
 - Math (数学工具), 49
- forwardDynamics
 - RobotAlgorithm (机器人算法工具), 307
- forwardDynamics1
 - RobotAlgorithm (机器人算法工具), 308
- forwardKinematics
 - RobotAlgorithm (机器人算法工具), 308
- forwardKinematics1
 - RobotAlgorithm (机器人算法工具), 309
- forwardKinematicsAll
 - RobotAlgorithm (机器人算法工具), 310
- forwardToolKinematics
 - RobotAlgorithm (机器人算法工具), 311
- frame
 - arcs::common_interface::SpiralParameters, 582
- frameAdd
 - SyncMove (同步运动控制和轴组管理), 483
- frameAttach
 - SyncMove (同步运动控制和轴组管理), 484
- frameConvertPose
 - SyncMove (同步运动控制和轴组管理), 484
- frameDelete
 - SyncMove (同步运动控制和轴组管理), 484
- frameDeleteAll
 - SyncMove (同步运动控制和轴组管理), 485
- frameExist
 - SyncMove (同步运动控制和轴组管理), 485
- frameGetChildren
 - SyncMove (同步运动控制和轴组管理), 485
- frameGetParent
 - SyncMove (同步运动控制和轴组管理), 486
- frameGetPose
 - SyncMove (同步运动控制和轴组管理), 486
- frameMove
 - SyncMove (同步运动控制和轴组管理), 486
- freedrive
 - RobotManage (机器人生命周期管理), 379
- frequency
 - arcs::common_interface::RtdeRecipe, 576
- frictionModelIdentify
 - RobotAlgorithm (机器人算法工具), 312
- generateDiagnoseFile
 - RobotManage (机器人生命周期管理), 380
- generatePayloadIdentifyTraj
 - RobotAlgorithm (机器人算法工具), 313
- getActualJointPositionsHistory
 - ForceControl(力控模块), 115
- getActualTcpOffset
 - RobotState (机器人状态查询), 402
- getAdvancePlanContext
 - RuntimeMachine (运行时管理), 448
- getAdvancePtr
 - RuntimeMachine (运行时管理), 449
- getAxisInterface
 - AuboApi (主入口), 26
- getAxisModeType
 - AxisInterface (外部轴), 31
- getAxisNames
 - AuboApi (主入口), 26
- getBaseForce
 - RobotState (机器人状态查询), 402
- getBaseForceOffset
 - RobotConfig (机器人配置管理), 334
- getBaseForceSensor
 - arcs::common_interface::RobotState, 574
- getBaseForceSensorNames
 - RobotConfig (机器人配置管理), 334
- getBool
 - RegisterControl (寄存器操作), 65
- getBoolInput
 - RegisterControl (寄存器操作), 66
- getBoolOutput
 - RegisterControl (寄存器操作), 66
- getCollisionLevel
 - RobotConfig (机器人配置管理), 335
- getCollisionStopType
 - RobotConfig (机器人配置管理), 335
- getCollisionThreshold
 - RobotConfig (机器人配置管理), 336
- getConfigurableDigitalInput
 - IoControl (IO 输入输出控制), 150
- getConfigurableDigitalInputAction
 - IoControl (IO 输入输出控制), 151
- getConfigurableDigitalInputNum
 - IoControl (IO 输入输出控制), 152
- getConfigurableDigitalInputs
 - IoControl (IO 输入输出控制), 152
- getConfigurableDigitalOutput
 - IoControl (IO 输入输出控制), 153
- getConfigurableDigitalOutputNum
 - IoControl (IO 输入输出控制), 154
- getConfigurableDigitalOutputRunstate
 - IoControl (IO 输入输出控制), 154
- getConfigurableDigitalOutputs
 - IoControl (IO 输入输出控制), 155
- getControlBoxHumidity
 - RobotState (机器人状态查询), 402
- getControlBoxTemperature

- RobotState (机器人状态查询), 403
- getControlBoxType
 - RobotConfig (机器人配置管理), 337
- getControlSoftwareBuildDate
 - SystemInfo (系统信息), 491
- getControlSoftwareFullVersion
 - SystemInfo (系统信息), 491
- getControlSoftwareVersionCode
 - SystemInfo (系统信息), 491
- getControlSoftwareVersionHash
 - SystemInfo (系统信息), 492
- getControlSystemTime
 - SystemInfo (系统信息), 492
- getConveyorTrackNextItem
 - arcs::common_interface::MotionControl, 545
- getConveyorTrackQueue
 - MotionControl (运动规划与控制), 226
- getCycletime
 - RobotConfig (机器人配置管理), 337
- getDefaultJointAcc
 - RobotConfig (机器人配置管理), 338
- getDefaultJointSpeed
 - RobotConfig (机器人配置管理), 338
- getDefaultToolAcc
 - RobotConfig (机器人配置管理), 339
- getDefaultToolSpeed
 - RobotConfig (机器人配置管理), 340
- getDeltaPoseBySensorDistance
 - Math (数学工具), 50
- getDigitalOutputRunstate
 - IoControl (IO 输入输出控制), 156
- getDof
 - RobotConfig (机器人配置管理), 340
- getDouble
 - RegisterControl (寄存器操作), 67
- getDoubleInput
 - RegisterControl (寄存器操作), 68
- getDoubleOutput
 - RegisterControl (寄存器操作), 68
- getDuration
 - MotionControl (运动规划与控制), 226
- getDynamicModel
 - ForceControl(力控模块), 115
- getElbowPosistion
 - RobotState (机器人状态查询), 403
- getElbowVelocity
 - RobotState (机器人状态查询), 404
- getEncDecoderType
 - IoControl (IO 输入输出控制), 157
- getEncTickCount
 - IoControl (IO 输入输出控制), 157
- getEqradius
 - MotionControl (运动规划与控制), 227
- getErrorCode
 - AxisInterface (外部轴), 31
- getExecId
 - MotionControl (运动规划与控制), 228
- getExecutionStatus
 - RuntimeMachine (运行时管理), 449
- getExecutionStatus1
 - arcs::common_interface::RuntimeMachine, 579
- getExtAxisAcceleration
 - AxisInterface (外部轴), 32
- getExtAxisBusCurrent
 - AxisInterface (外部轴), 32
- getExtAxisBusVoltage
 - AxisInterface (外部轴), 32
- getExtAxisCurrent
 - AxisInterface (外部轴), 32
- getExtAxisMaxAcceleration
 - AxisInterface (外部轴), 32
- getExtAxisMaxAccelerationLimit
 - AxisInterface (外部轴), 32
- getExtAxisMaxPosition
 - AxisInterface (外部轴), 32
- getExtAxisMaxPositionLimit
 - AxisInterface (外部轴), 33
- getExtAxisMaxVelocity
 - AxisInterface (外部轴), 33
- getExtAxisMaxVelocityLimit
 - AxisInterface (外部轴), 33
- getExtAxisMountingPose
 - AxisInterface (外部轴), 33
- getExtAxisPose
 - AxisInterface (外部轴), 33
- getExtAxisPosition
 - AxisInterface (外部轴), 33
- getExtAxisTemperature
 - AxisInterface (外部轴), 33
- getExtAxisType
 - AxisInterface (外部轴), 34
- getExtAxisVelocity
 - AxisInterface (外部轴), 34
- getExtMinPosition
 - AxisInterface (外部轴), 34
- getExtMinPositionLimit
 - AxisInterface (外部轴), 34
- getFcSensorLimits
 - ForceControl(力控模块), 116
- getFcSensorThresholds
 - ForceControl(力控模块), 116
- getFirmwareUpdateProcess
 - RobotConfig (机器人配置管理), 341
- getFloat
 - RegisterControl (寄存器操作), 69
- getFloatInput
 - RegisterControl (寄存器操作), 70
- getFloatOutput
 - RegisterControl (寄存器操作), 70
- getForceControl
 - RobotInterface(机器人模块), 106
- getFreedriveDamp
 - RobotConfig (机器人配置管理), 341
- getFuturePathPointsJoint
 - MotionControl (运动规划与控制), 228
- getGravity

- RobotConfig (机器人配置管理), 342
- getGripperInterface
 - AuboApi (主入口), 26
- getGroupInput
 - IoControl (IO 输入输出控制), 157
- getGroupOutput
 - IoControl (IO 输入输出控制), 158
- getHandguidDamp
 - RobotConfig (机器人配置管理), 343
- getHandguideFeature
 - RobotManage (机器人生命周期管理), 381
- getHandguideFreeAxes
 - RobotManage (机器人生命周期管理), 381
- getHandguideStatus
 - RobotManage (机器人生命周期管理), 381
- getHandguideTrigger
 - RobotManage (机器人生命周期管理), 381
- getHandleFirmwareVersion
 - RobotState (机器人状态查询), 405
- getHandleHardwareVersion
 - RobotState (机器人状态查询), 405
- getHandleIoStatus
 - IoControl (IO 输入输出控制), 159
- getHandleMode
 - RobotState (机器人状态查询), 406
- getHandleState
 - RobotState (机器人状态查询), 406
- getHandleType
 - IoControl (IO 输入输出控制), 160
- getHandleUniqueId
 - RobotState (机器人状态查询), 407
- getHardwareCustomParameters
 - RobotConfig (机器人配置管理), 343
- getHomePosition
 - RobotConfig (机器人配置管理), 344
- getInt16Register
 - RegisterControl (寄存器操作), 71
- getInt16RegisterBit
 - RegisterControl (寄存器操作), 72
- getInt32
 - RegisterControl (寄存器操作), 72
- getInt32Input
 - RegisterControl (寄存器操作), 73
- getInt32Output
 - RegisterControl (寄存器操作), 74
- getInterfaceVersionCode
 - SystemInfo (系统信息), 493
- getInterpPtr
 - RuntimeMachine (运行时管理), 449
- getIoControl
 - RobotInterface(机器人模块), 106
- getJointAccelerations
 - RobotState (机器人状态查询), 408
- getJointContactTorques
 - RobotState (机器人状态查询), 408
- getJointCurrents
 - RobotState (机器人状态查询), 409
- getJointFirmwareVersions
 - RobotState (机器人状态查询), 409
- getJointGravityTorques
 - RobotState (机器人状态查询), 410
- getJointHardwareVersions
 - RobotState (机器人状态查询), 411
- getJointMaxAccelerations
 - RobotConfig (机器人配置管理), 344
- getJointMaxPositions
 - RobotConfig (机器人配置管理), 345
- getJointMaxSpeeds
 - RobotConfig (机器人配置管理), 345
- getJointMinPositions
 - RobotConfig (机器人配置管理), 346
- getJointPositions
 - RobotState (机器人状态查询), 411
- getJointPositionsHistory
 - RobotState (机器人状态查询), 412
- getJointServoMode
 - RobotState (机器人状态查询), 412
- getJointSpeeds
 - RobotState (机器人状态查询), 413
- getJointState
 - RobotState (机器人状态查询), 413
- getJointTargetAccelerations
 - RobotState (机器人状态查询), 414
- getJointTargetCurrents
 - RobotState (机器人状态查询), 414
- getJointTargetPositions
 - RobotState (机器人状态查询), 415
- getJointTargetSpeeds
 - RobotState (机器人状态查询), 416
- getJointTargetTorques
 - RobotState (机器人状态查询), 416
- getJointTemperatures
 - RobotState (机器人状态查询), 417
- getJointTorqueSensors
 - RobotState (机器人状态查询), 417
- getJointUniqueIds
 - RobotState (机器人状态查询), 418
- getJointVoltages
 - RobotState (机器人状态查询), 419
- getKinematicsCompensate
 - RobotConfig (机器人配置管理), 346
- getKinematicsParam
 - RobotConfig (机器人配置管理), 347
- getLimitJointMaxAccelerations
 - RobotConfig (机器人配置管理), 348
- getLimitJointMaxPositions
 - RobotConfig (机器人配置管理), 349
- getLimitJointMaxSpeeds
 - RobotConfig (机器人配置管理), 349
- getLimitJointMinPositions
 - RobotConfig (机器人配置管理), 350
- getLimitTcpMaxSpeed
 - RobotConfig (机器人配置管理), 350
- getLookAheadSize
 - MotionControl (运动规划与控制), 229
- getMainCurrent

- RobotState (机器人状态查询), 419
- getMainPtr
 - RuntimeMachine (运行时管理), 450
- getMainVoltage
 - RobotState (机器人状态查询), 420
- getMasterBoardFirmwareVersion
 - RobotState (机器人状态查询), 420
- getMasterBoardHardwareVersion
 - RobotState (机器人状态查询), 421
- getMasterBoardUniqueId
 - RobotState (机器人状态查询), 422
- getMath
 - AuboApi (主入口), 26
- getModbusDeviceStatus
 - RegisterControl (寄存器操作), 74
- getMotionControl
 - RobotInterface(机器人模块), 106
- getMotionLeftTime
 - MotionControl (运动规划与控制), 229
- getMountingPose
 - RobotConfig (机器人配置管理), 351
- getName
 - RobotConfig (机器人配置管理), 352
- getNamedVariableType
 - RegisterControl (寄存器操作), 75
- getOperationalMode
 - RobotManage (机器人生命周期管理), 382
- getPauseJointPositions
 - MotionControl (运动规划与控制), 230
- getPayload
 - RobotConfig (机器人配置管理), 352
- getPayloadIdentifyResult
 - RobotAlgorithm (机器人算法工具), 313
- getPedestalBrakeResTemp
 - RobotState (机器人状态查询), 422
- getPedestalFirmwareVersion
 - RobotState (机器人状态查询), 423
- getPedestalHardwareVersion
 - RobotState (机器人状态查询), 423
- getPedestalUniqueId
 - RobotState (机器人状态查询), 424
- getPlanContext
 - RuntimeMachine (运行时管理), 450
- getPowerlossBoardFirmwareVersion
 - RobotState (机器人状态查询), 425
- getPowerlossBoardHardwareVersion
 - RobotState (机器人状态查询), 425
- getPowerlossBoardUniqueId
 - RobotState (机器人状态查询), 426
- getPreloadProgram
 - RuntimeMachine (运行时管理), 451
- getProgress
 - MotionControl (运动规划与控制), 231
- getQueueSize
 - MotionControl (运动规划与控制), 231
- getRecordCache
 - RobotManage (机器人生命周期管理), 382
- getReductionRatio
 - AxisInterface (外部轴), 34
- getRegisterControl
 - AuboApi (主入口), 26
- getResumeMode
 - MotionControl (运动规划与控制), 232
- getRobotAlgorithm
 - RobotInterface(机器人模块), 107
- getRobotBaseParent
 - arcs::common_interface::RobotConfig, 559
- getRobotConfig
 - RobotInterface(机器人模块), 107
- getRobotConfiguration
 - RobotAlgorithm (机器人算法工具), 314
- getRobotControlMode
 - RobotManage (机器人生命周期管理), 383
- getRobotCurrent
 - RobotState (机器人状态查询), 426
- getRobotEmergencyStopSource
 - RobotConfig (机器人配置管理), 353
- getRobotInterface
 - AuboApi (主入口), 27
- getRobotManage
 - RobotInterface(机器人模块), 107
- getRobotModeType
 - RobotState (机器人状态查询), 427
- getRobotNames
 - AuboApi (主入口), 27
- getRobotState
 - RobotInterface(机器人模块), 108
- getRobotSubType
 - RobotConfig (机器人配置管理), 353
- getRobotType
 - RobotConfig (机器人配置管理), 354
- getRobotVoltage
 - RobotState (机器人状态查询), 428
- getRuntimeMachine
 - AuboApi (主入口), 27
- getRuntimeState
 - arcs::common_interface::RuntimeMachine, 579
- getSafeguardStopSource
 - RobotConfig (机器人配置管理), 354
- getSafeguardStopType
 - RobotConfig (机器人配置管理), 355
- getSafetyModeType
 - RobotState (机器人状态查询), 428
- getSafetyParametersChecksum
 - RobotConfig (机器人配置管理), 355
- getSelectedTcpForceSensorName
 - RobotConfig (机器人配置管理), 356
- getSerial
 - AuboApi (主入口), 28
- getServoModeSelect
 - MotionControl (运动规划与控制), 233
- getSlaveBoardFirmwareVersion
 - RobotState (机器人状态查询), 429
- getSlaveBoardHardwareVersion
 - RobotState (机器人状态查询), 429
- getSlaveBoardUniqueId

- RobotState (机器人状态查询), 430
- getSlowDownFraction
 - RobotConfig (机器人配置管理), 356
- getSlowDownLevel
 - RobotState (机器人状态查询), 431
- getSocket
 - AuboApi (主入口), 28
- getSpeedFraction
 - MotionControl (运动规划与控制), 233
- getStandardAnalogInput
 - IoControl (IO 输入输出控制), 160
- getStandardAnalogInputDomain
 - IoControl (IO 输入输出控制), 161
- getStandardAnalogInputNum
 - IoControl (IO 输入输出控制), 162
- getStandardAnalogOutput
 - IoControl (IO 输入输出控制), 162
- getStandardAnalogOutputDomain
 - IoControl (IO 输入输出控制), 163
- getStandardAnalogOutputNum
 - IoControl (IO 输入输出控制), 164
- getStandardAnalogOutputRunstate
 - IoControl (IO 输入输出控制), 164
- getStandardDigitalInput
 - IoControl (IO 输入输出控制), 165
- getStandardDigitalInputAction
 - IoControl (IO 输入输出控制), 166
- getStandardDigitalInputNum
 - IoControl (IO 输入输出控制), 166
- getStandardDigitalInputs
 - IoControl (IO 输入输出控制), 167
- getStandardDigitalOutput
 - IoControl (IO 输入输出控制), 168
- getStandardDigitalOutputNum
 - IoControl (IO 输入输出控制), 168
- getStandardDigitalOutputRunstate
 - IoControl (IO 输入输出控制), 169
- getStandardDigitalOutputs
 - IoControl (IO 输入输出控制), 170
- getStaticLinkInputNum
 - IoControl (IO 输入输出控制), 170
- getStaticLinkInputs
 - IoControl (IO 输入输出控制), 171
- getStaticLinkOutputNum
 - IoControl (IO 输入输出控制), 172
- getStaticLinkOutputs
 - IoControl (IO 输入输出控制), 172
- getStatus
 - RuntimeMachine (运行时管理), 451
- getString
 - RegisterControl (寄存器操作), 75
- getSyncMove
 - AuboApi (主入口), 28
 - RobotInterface(机器人模块), 108
- getSystemInfo
 - AuboApi (主入口), 28
- getTargetTcpPose
 - RobotState (机器人状态查询), 431
- getTaskQueueSize
 - RuntimeMachine (运行时管理), 451
- getTcpForce
 - RobotState (机器人状态查询), 432
- getTcpForceOffset
 - RobotConfig (机器人配置管理), 357
- getTcpForceSensorNames
 - RobotConfig (机器人配置管理), 357
- getTcpForceSensorPose
 - RobotConfig (机器人配置管理), 358
- getTcpForceSensors
 - RobotState (机器人状态查询), 432
- getTcpForceSensorStatus
 - RobotState (机器人状态查询), 433
- getTcpMaxAccelerations
 - RobotConfig (机器人配置管理), 358
- getTcpMaxLinearVelocity
 - MotionControl (运动规划与控制), 234
- getTcpMaxSpeeds
 - RobotConfig (机器人配置管理), 359
- getTcpOffset
 - RobotConfig (机器人配置管理), 359
- getTcpPose
 - RobotState (机器人状态查询), 433
- getTcpSpeed
 - RobotState (机器人状态查询), 434
- getTcpTargetForce
 - RobotState (机器人状态查询), 435
- getTcpTargetPose
 - RobotState (机器人状态查询), 435
- getTcpTargetSpeed
 - RobotState (机器人状态查询), 436
- getTimer
 - RuntimeMachine (运行时管理), 452
- getToolAnalogInput
 - IoControl (IO 输入输出控制), 173
- getToolAnalogInputDomain
 - IoControl (IO 输入输出控制), 174
- getToolAnalogInputNum
 - IoControl (IO 输入输出控制), 174
- getToolAnalogOutput
 - IoControl (IO 输入输出控制), 175
- getToolAnalogOutputDomain
 - IoControl (IO 输入输出控制), 176
- getToolAnalogOutputNum
 - IoControl (IO 输入输出控制), 176
- getToolAnalogOutputRunstate
 - IoControl (IO 输入输出控制), 177
- getToolButtonStatus
 - IoControl (IO 输入输出控制), 178
- getToolCommMode
 - RobotState (机器人状态查询), 436
- getToolDigitalInput
 - IoControl (IO 输入输出控制), 178
- getToolDigitalInputAction
 - IoControl (IO 输入输出控制), 179
- getToolDigitalInputNum
 - IoControl (IO 输入输出控制), 180

- getToolDigitalInputs
 - IoControl (IO 输入输出控制), 180
- getToolDigitalOutput
 - IoControl (IO 输入输出控制), 181
- getToolDigitalOutputNum
 - IoControl (IO 输入输出控制), 182
- getToolDigitalOutputRunstate
 - IoControl (IO 输入输出控制), 182
- getToolDigitalOutputs
 - IoControl (IO 输入输出控制), 183
- getToolFirmwareVersion
 - RobotState (机器人状态查询), 437
- getToolHardwareVersion
 - RobotState (机器人状态查询), 438
- getToolPose
 - RobotState (机器人状态查询), 438
- getToolUniqueId
 - RobotState (机器人状态查询), 439
- getToolVoltageOutputDomain
 - IoControl (IO 输入输出控制), 184
- getTrace
 - AuboApi (主入口), 29
 - RobotInterface(机器人模块), 108
- getTrajectoryQueueSize
 - MotionControl (运动规划与控制), 234
- getTriggInterrupts
 - RuntimeMachine (运行时管理), 452
- getVecChar
 - RegisterControl (寄存器操作), 76
- getVecDouble
 - RegisterControl (寄存器操作), 77
- getVecFloat
 - RegisterControl (寄存器操作), 77
- getVecInt32
 - RegisterControl (寄存器操作), 78
- getWatchDogAction
 - RegisterControl (寄存器操作), 79
- getWatchDogTimeout
 - RegisterControl (寄存器操作), 79
- getWeaveDirection
 - MotionControl (运动规划与控制), 235
- getWorkObjectHold
 - MotionControl (运动规划与控制), 236
- gotoLine
 - RuntimeMachine (运行时管理), 452
- gripperAdd
 - GripperInterface (夹爪), 39
- gripperConnect
 - GripperInterface (夹爪), 39
- gripperDelete
 - GripperInterface (夹爪), 40
- gripperDisconnect
 - GripperInterface (夹爪), 40
- gripperEnable
 - GripperInterface (夹爪), 40
- gripperGetAngle
 - arcs::common_interface::GripperInterface, 530
- gripperGetForce
 - arcs::common_interface::GripperInterface, 530
- gripperGetHardwareVersion
 - GripperInterface (夹爪), 41
- gripperGetMotionState
 - arcs::common_interface::GripperInterface, 530
- gripperGetMountPose
 - arcs::common_interface::GripperInterface, 530
- gripperGetNames
 - GripperInterface (夹爪), 41
- gripperGetObjectDetection
 - arcs::common_interface::GripperInterface, 531
- gripperGetPosition
 - arcs::common_interface::GripperInterface, 531
- gripperGetRVelocity
 - arcs::common_interface::GripperInterface, 531
- gripperGetSoftwareVersion
 - arcs::common_interface::GripperInterface, 531
- gripperGetStatusCode
 - GripperInterface (夹爪), 41
- gripperGetSupportedModels
 - GripperInterface (夹爪), 41
- gripperGetTemperature
 - arcs::common_interface::GripperInterface, 531
- gripperGetTorque
 - arcs::common_interface::GripperInterface, 531
- gripperGetVelocity
 - arcs::common_interface::GripperInterface, 531
- gripperGetVoltage
 - arcs::common_interface::GripperInterface, 531
- gripperGetWorkMode
 - GripperInterface (夹爪), 42
- GripperInterface
 - arcs::common_interface::GripperInterface, 530
- GripperInterface (夹爪), 38
 - gripperAdd, 39
 - gripperConnect, 39
 - gripperDelete, 40
 - gripperDisconnect, 40
 - gripperEnable, 40
 - gripperGetHardwareVersion, 41
 - gripperGetNames, 41
 - gripperGetStatusCode, 41
 - gripperGetSupportedModels, 41
 - gripperGetWorkMode, 42
 - gripperIsConnected, 42
 - gripperIsEnabled, 42
 - gripperMove, 42
 - gripperRename, 43
 - gripperResetSlaveId, 43
 - gripperScanDevices, 43
 - gripperSetMountPose, 44
 - gripperSetPosition, 44
 - gripperSetWorkMode, 44
 - gripperStop, 45
- GripperInterfacePtr
 - arcs::common_interface, 504
- gripperIsConnected
 - GripperInterface (夹爪), 42

- gripperIsEnabled
 - GripperInterface (夹爪), 42
- gripperMove
 - GripperInterface (夹爪), 42
- gripperRename
 - GripperInterface (夹爪), 43
- gripperResetSlaveId
 - GripperInterface (夹爪), 43
- gripperScanDevices
 - GripperInterface (夹爪), 43
- gripperSetAngle
 - arcs::common_interface::GripperInterface, 531
- gripperSetForce
 - arcs::common_interface::GripperInterface, 531
- gripperSetMountPose
 - GripperInterface (夹爪), 44
- gripperSetPosition
 - GripperInterface (夹爪), 44
- gripperSetRVelocity
 - arcs::common_interface::GripperInterface, 532
- gripperSetTorque
 - arcs::common_interface::GripperInterface, 532
- gripperSetVelocity
 - arcs::common_interface::GripperInterface, 532
- gripperSetWorkMode
 - GripperInterface (夹爪), 44
- GripperStatusVector
 - arcs::common_interface, 504
- gripperStop
 - GripperInterface (夹爪), 45
- hal_error.h
 - EX_PEDSTRAL_ERRORS, 614
 - EX_TOOL_ERRORS, 614
 - EXT_AXIS_ERRORS, 615
 - HAL_ERRORS, 615
 - HARDWARE_INTERFACE_ERRORS, 615
 - JOINT_ERRORS, 615
 - JOINT_EX_ERRORS, 615
 - PEDSTRAL_ERRORS, 615
 - SAFETY_INTERFACE_BOARD_ERRORS, 615
 - TOOL_ERRORS, 616
- HAL_ERRORS
 - error_stack.h, 610
 - hal_error.h, 615
- handguideMode
 - RobotManage (机器人生命周期管理), 384
- HandleModeType
 - arcs::common_interface, 509
- HandleStateType
 - arcs::common_interface, 509
- HARDWARE_INTERFACE_ERRORS
 - error_stack.h, 610
 - hal_error.h, 615
- hasBaseForceSensor
 - RobotConfig (机器人配置管理), 360
- hasEncoderSensor
 - IoControl (IO 输入输出控制), 184
- hasItemOnConveyorToTrack
 - MotionControl (运动规划与控制), 237
- hasNamedVariable
 - RegisterControl (寄存器操作), 79
- hasTcpForceSensor
 - RobotConfig (机器人配置管理), 361
- helix
 - arcs::common_interface::CircleParameters, 524
 - arcs::common_interface::SpiralParameters, 582
- homMatrixToPose
 - Math (数学工具), 50
- include 目录参考, 499
- include/aubo 目录参考, 497
- include/aubo/aubo_api.h, 593, 594
- include/aubo/axis_interface.h, 598, 599
- include/aubo/error_stack 目录参考, 498
- include/aubo/error_stack/error_stack.h, 607, 612
- include/aubo/error_stack/hal_error.h, 613, 616
- include/aubo/error_stack/rtm_error.h, 622, 623
- include/aubo/error_stack/system_error.h, 627, 628
- include/aubo/gripper_interface.h, 628, 629
- include/aubo/math.h, 633, 634
- include/aubo/register_control.h, 646, 648
- include/aubo/robot 目录参考, 499
- include/aubo/robot/force_control.h, 689, 691
- include/aubo/robot/io_control.h, 721, 723
- include/aubo/robot/motion_control.h, 789, 790
- include/aubo/robot/robot_algorithm.h, 869, 870
- include/aubo/robot/robot_config.h, 895, 896
- include/aubo/robot/robot_manage.h, 940, 941
- include/aubo/robot/robot_state.h, 963, 965
- include/aubo/robot/robot_interface.h, 1005, 1006
- include/aubo/runtime_machine.h, 1011, 1012
- include/aubo/serial.h, 1031, 1032
- include/aubo/socket.h, 1037, 1038
- include/aubo/sync_move.h, 1046, 1047
- include/aubo/system_info.h, 1059, 1060
- include/aubo/trace.h, 1064, 1065
- include/aubo/type_def.h, 1068, 1078
- inertia
 - arcs::common_interface::VibrationRecalibrationParameter, 589
- init_joint
 - arcs::common_interface::TrajConfig, 588
- internal_error
 - arcs::common_interface, 508
- interpolatePose
 - Math (数学工具), 50
- invalid
 - arcs::common_interface, 508
- invalid_params
 - arcs::common_interface, 508
- invalid_request
 - arcs::common_interface, 508
- inverseKinematics
 - RobotAlgorithm (机器人算法工具), 315
- inverseKinematics1
 - RobotAlgorithm (机器人算法工具), 316

- inverseKinematics2
 - RobotAlgorithm (机器人算法工具), 317
- inverseKinematicsAll
 - RobotAlgorithm (机器人算法工具), 318
- inverseKinematicsAll1
 - RobotAlgorithm (机器人算法工具), 318
- inverseToolKinematics
 - RobotAlgorithm (机器人算法工具), 319
- inverseToolKinematicsAll
 - RobotAlgorithm (机器人算法工具), 320
- IoControl
 - arcs::common_interface::IoControl, 538
- IoControl (IO 输入输出控制), 140
 - addCrossConnection, 145
 - addGroupInput, 146
 - addGroupOutput, 147
 - clearCrossConnections, 148
 - deleteCrossConnection, 148
 - deleteGroupInput, 149
 - deleteGroupOutput, 150
 - getConfigurableDigitalInput, 150
 - getConfigurableDigitalInputAction, 151
 - getConfigurableDigitalInputNum, 152
 - getConfigurableDigitalInputs, 152
 - getConfigurableDigitalOutput, 153
 - getConfigurableDigitalOutputNum, 154
 - getConfigurableDigitalOutputRunstate, 154
 - getConfigurableDigitalOutputs, 155
 - getDigitalOutputRunstate, 156
 - getEncDecoderType, 157
 - getEncTickCount, 157
 - getGroupInput, 157
 - getGroupOutput, 158
 - getHandleIoStatus, 159
 - getHandleType, 160
 - getStandardAnalogInput, 160
 - getStandardAnalogInputDomain, 161
 - getStandardAnalogInputNum, 162
 - getStandardAnalogOutput, 162
 - getStandardAnalogOutputDomain, 163
 - getStandardAnalogOutputNum, 164
 - getStandardAnalogOutputRunstate, 164
 - getStandardDigitalInput, 165
 - getStandardDigitalInputAction, 166
 - getStandardDigitalInputNum, 166
 - getStandardDigitalInputs, 167
 - getStandardDigitalOutput, 168
 - getStandardDigitalOutputNum, 168
 - getStandardDigitalOutputRunstate, 169
 - getStandardDigitalOutputs, 170
 - getStaticLinkInputNum, 170
 - getStaticLinkInputs, 171
 - getStaticLinkOutputNum, 172
 - getStaticLinkOutputs, 172
 - getToolAnalogInput, 173
 - getToolAnalogInputDomain, 174
 - getToolAnalogInputNum, 174
 - getToolAnalogOutput, 175
 - getToolAnalogOutputDomain, 176
 - getToolAnalogOutputNum, 176
 - getToolAnalogOutputRunstate, 177
 - getToolButtonStatus, 178
 - getToolDigitalInput, 178
 - getToolDigitalInputAction, 179
 - getToolDigitalInputNum, 180
 - getToolDigitalInputs, 180
 - getToolDigitalOutput, 181
 - getToolDigitalOutputNum, 182
 - getToolDigitalOutputRunstate, 182
 - getToolDigitalOutputs, 183
 - getToolVoltageOutputDomain, 184
 - hasEncoderSensor, 184
 - isToolIoInput, 185
 - setAnalogOutputAfterEStopDefault, 185
 - setConfigurableDigitalInputAction, 186
 - setConfigurableDigitalOutput, 187
 - setConfigurableDigitalOutputAfterEStop, 188
 - setConfigurableDigitalOutputPulse, 189
 - setConfigurableDigitalOutputRunstate, 189
 - setDigitalInputActionDefault, 190
 - setDigitalOutputAfterEStopDefault, 191
 - setDigitalOutputRunstate, 191
 - setDigitalOutputRunstateDefault, 192
 - setEncDecoderType, 193
 - setEncTickCount, 193
 - setGroupOutput, 194
 - setStandardAnalogInputDomain, 195
 - setStandardAnalogOutput, 195
 - setStandardAnalogOutputAfterEStop, 196
 - setStandardAnalogOutputDomain, 197
 - setStandardAnalogOutputRunstate, 198
 - setStandardDigitalInputAction, 199
 - setStandardDigitalOutput, 200
 - setStandardDigitalOutputAfterEStop, 200
 - setStandardDigitalOutputPulse, 201
 - setStandardDigitalOutputRunstate, 202
 - setToolAnalogInputDomain, 203
 - setToolAnalogOutput, 204
 - setToolAnalogOutputDomain, 204
 - setToolAnalogOutputRunstate, 205
 - setToolDigitalInputAction, 206
 - setToolDigitalOutput, 207
 - setToolDigitalOutputPulse, 208
 - setToolDigitalOutputRunstate, 209
 - setToolIoInput, 210
 - setToolVoltageOutputDomain, 210
 - unwindEncDeltaTickCount, 211
- IoControlPtr
 - arcs::common_interface, 504
- is_connected
 - arcs::common_interface::GripperStatus, 533
- is_enabled
 - arcs::common_interface::GripperStatus, 533
- isBackdriveEnabled
- isBackdriveManage (机器人生命周期管理), 384
- isBlending

- MotionControl (运动规划与控制), 237
- isCollisionOccurred
 - RobotState (机器人状态查询), 439
- isCondFullfiled
 - ForceControl(力控模块), 116
- isConveyorTrackExceed
 - arcs::common_interface::MotionControl, 546
- isConveyorTrackSync
 - MotionControl (运动规划与控制), 238
- isEndCollisionCheckEnabled
 - RobotConfig (机器人配置管理), 361
- isFcEnabled
 - ForceControl(力控模块), 117
- isFreedriveEnabled
 - RobotManage (机器人生命周期管理), 385
- isHandguideEnabled
 - RobotManage (机器人生命周期管理), 385
- isJointSoftServoEnabled
 - MotionControl (运动规划与控制), 239
- isLinkModeEnabled
 - RobotManage (机器人生命周期管理), 386
- isPowerOn
 - RobotState (机器人状态查询), 440
- isServoModeEnabled
 - MotionControl (运动规划与控制), 239
- isSimulationEnabled
 - RobotManage (机器人生命周期管理), 387
- isSoftFloatEnabled
 - ForceControl(力控模块), 118
- isSpeedFractionCritical
 - MotionControl (运动规划与控制), 240
- isSteady
 - RobotState (机器人状态查询), 440
- isSupportedTimeOptimal
 - MotionControl (运动规划与控制), 240
- isSyncMoveOn
 - SyncMove (同步运动控制和轴组管理), 487
- isTaskAlive
 - RuntimeMachine (运行时管理), 453
- isTeachPendantEnabled
 - RobotState (机器人状态查询), 441
- isTimeOptimalEnabled
 - MotionControl (运动规划与控制), 241
- isToolFlangeEnabled
 - RobotState (机器人状态查询), 442
- isToolIoInput
 - IoControl (IO 输入输出控制), 185
- isWithinSafetyLimits
 - RobotState (机器人状态查询), 442
- JOINT_ERRORS
 - error_stack.h, 610
 - hal_error.h, 615
- JOINT_EX_ERRORS
 - error_stack.h, 611
 - hal_error.h, 615
- joint_torque
 - arcs::common_interface::RobotSafetyParameterManager, 566
- jointOffsetDisable
 - MotionControl (运动规划与控制), 241
- jointOffsetEnable
 - MotionControl (运动规划与控制), 242
- jointOffsetSet
 - MotionControl (运动规划与控制), 243
- JointServoModeType
 - arcs::common_interface, 509
- JointStateType
 - arcs::common_interface, 509
- level
 - arcs::common_interface::RobotMsg, 563
- loadProgram
 - RuntimeMachine (运行时管理), 453
- lockRobotBrake
 - RobotManage (机器人生命周期管理), 387
- loop_times
 - arcs::common_interface::CircleParameters, 524
- lower_joint_bound
 - arcs::common_interface::TrajConfig, 588
- mass
 - arcs::common_interface::VibrationRecalibrationParameter, 589
- Math
 - arcs::common_interface::Math, 540
- Math (数学工具), 45
 - calculateCircleFourthPoint, 46
 - calibrateCoordinate, 47
 - changePoseWithXYRef, 48
 - deltaPoseAdd, 48
 - deltaPoseTrans, 49
 - forceTrans, 49
 - getDeltaPoseBySensorDistance, 50
 - homMatrixToPose, 50
 - interpolatePose, 50
 - poseAdd, 51
 - poseAngleDistance, 51
 - poseDistance, 52
 - poseEqual, 52
 - poseInverse, 53
 - poseRotation, 53
 - poseSub, 54
 - poseToHomMatrix, 55
 - poseTrans, 55
 - poseTransInv, 56
 - quaternionToRpy, 57
 - rpyToQuaternion, 57
 - tcpOffsetIdentify, 58
 - transferRefFrame, 58
- MathPtr
 - arcs::common_interface, 505
- max_acceleration
 - arcs::common_interface::TrajConfig, 588
- max_velocity
 - arcs::common_interface::TrajConfig, 588
- MB_ERR_ACKNOWLEDGE
 - register_control.h, 648

- MB_ERR_DISCONNECTED
 - register_control.h, 648
- MB_ERR_ILLEGAL_DATA_ADDRESS
 - register_control.h, 648
- MB_ERR_ILLEGAL_DATA_VALUE
 - register_control.h, 648
- MB_ERR_ILLEGAL_FUNCTION
 - register_control.h, 648
- MB_ERR_NOT_INIT
 - register_control.h, 648
- MB_ERR_SLAVE_DEVICE_BUSY
 - register_control.h, 648
- MB_ERR_SLAVE_DEVICE_FAILURE
 - register_control.h, 648
- message_
 - arcs::common_interface::AuboException, 521
- method_not_found
 - arcs::common_interface, 508
- mod
 - arcs::error_stack, 515
- modbusAddSignal
 - RegisterControl (寄存器操作), 80
- modbusDeleteAllSignals
 - RegisterControl (寄存器操作), 81
- modbusDeleteSignal
 - RegisterControl (寄存器操作), 81
- ModbusErrorNum
 - register_control.h, 648
- modbusGetSignalError
 - RegisterControl (寄存器操作), 82
- modbusGetSignalErrors
 - RegisterControl (寄存器操作), 82
- modbusGetSignalIndex
 - RegisterControl (寄存器操作), 83
- modbusGetSignalNames
 - RegisterControl (寄存器操作), 83
- modbusGetSignalStatus
 - RegisterControl (寄存器操作), 83
- modbusGetSignalTypes
 - RegisterControl (寄存器操作), 84
- modbusGetSignalValues
 - RegisterControl (寄存器操作), 84
- modbusSendCustomCommand
 - RegisterControl (寄存器操作), 84
- modbusSetDigitalInputAction
 - RegisterControl (寄存器操作), 86
- modbusSetOutputRunstate
 - RegisterControl (寄存器操作), 86
- modbusSetOutputRunstate1
 - RegisterControl (寄存器操作), 87
- modbusSetOutputSignal
 - RegisterControl (寄存器操作), 88
- modbusSetOutputSignal1
 - RegisterControl (寄存器操作), 89
- modbusSetOutputSignalPulse
 - RegisterControl (寄存器操作), 89
- modbusSetOutputSignalWithTimeout
 - arcs::common_interface::RegisterControl, 551
- modbusSetSignalUpdateFrequency
 - RegisterControl (寄存器操作), 90
- momentum
 - arcs::common_interface::RobotSafetyParameterRange, 566
- motion_state
 - arcs::common_interface::GripperStatus, 533
- MotionControl
 - arcs::common_interface::MotionControl, 545
- MotionControl (运动规划与控制), 212
 - clearPath, 217
 - conveyorTrackCircle, 218
 - conveyorTrackClearItems, 218
 - conveyorTrackCreatItem, 219
 - conveyorTrackLine, 220
 - conveyorTrackStop, 221
 - conveyorTrackSwitch, 221
 - disableJointSoftServo, 222
 - disbaleVibrationSuppress, 222
 - enableJointSoftServo, 223
 - enableVibrationSuppress, 224
 - followJoint, 225
 - followLine, 225
 - getConveyorTrackQueue, 226
 - getDuration, 226
 - getEqradius, 227
 - getExecId, 228
 - getFuturePathPointsJoint, 228
 - getLookAheadSize, 229
 - getMotionLeftTime, 229
 - getPauseJointPositions, 230
 - getProgress, 231
 - getQueueSize, 231
 - getResumeMode, 232
 - getServoModeSelect, 233
 - getSpeedFraction, 233
 - getTcpMaxLinearVelocity, 234
 - getTrajectoryQueueSize, 234
 - getWeaveDirection, 235
 - getWorkObjectHold, 236
 - hasItemOnConveyorToTrack, 237
 - isBlending, 237
 - isConveyorTrackSync, 238
 - isJointSoftServoEnabled, 239
 - isServoModeEnabled, 239
 - isSpeedFractionCritical, 240
 - isSupportedTimeOptimal, 240
 - isTimeOptimalEnabled, 241
 - jointOffsetDisable, 241
 - jointOffsetEnable, 242
 - jointOffsetSet, 243
 - moveCircle, 243
 - moveCircle2, 245
 - moveCircleWithAxisGroup, 245
 - moveIntersection, 246
 - moveJoint, 247
 - moveJointWithAxisGroup, 248
 - moveLine, 249

- moveLineWithAxisGroup, 250
- movePathBuffer, 251
- moveProcess, 251
- moveSpiral, 252
- moveSpline, 253
- pathBufferAlloc, 254
- pathBufferAppend, 255
- pathBufferEval, 255
- pathBufferFilter, 256
- pathBufferFree, 257
- pathBufferList, 258
- pathBufferValid, 258
- pathOffsetCoordinate, 259
- pathOffsetDisable, 260
- pathOffsetEnable, 260
- pathOffsetLimits, 261
- pathOffsetSet, 262
- pathOffsetSupv, 263
- resetTcpMaxLinearVelocity, 263
- restoPath, 264
- resumeMoveJoint, 264
- resumeMoveLine, 265
- resumeSpeedJoint, 266
- resumeSpeedLine, 267
- resumeStopJoint, 268
- resumeStopLine, 269
- servoCartesian, 270
- servoCartesianWithAxes, 271
- servoJoint, 272
- servoJointWithAxes, 273
- setCirclePathMode, 274
- setConveyorTrackCompensate, 274
- setConveyorTrackEncoder, 275
- setConveyorTrackLimit, 276
- setConveyorTrackSensorOffset, 277
- setConveyorTrackStartWindow, 277
- setConveyorTrackSyncSeparation, 278
- setConveyorTrackVelThreshold, 279
- setEndPath, 280
- setEqradius, 281
- setFuturePointSamplePeriod, 281
- setLookAheadSize, 282
- setResumeStartPoint, 282
- setServoMode, 283
- setServoModeSelect, 284
- setSpeedFraction, 285
- setTcpMaxLinearVelocity, 285
- setTimeOptimalEnable, 286
- setWorkObjectHold, 287
- speedFractionCritical, 287
- speedFractionCritical1, 288
- speedJoint, 289
- speedLine, 290
- startMove, 291
- stopJoint, 292
- stopLine, 292
- stopMove, 293
- storePath, 294
- trackCartesian, 294
- trackJoint, 295
- weaveEnd, 296
- weaveStart, 296
- weaveUpdateParameters, 299
- MotionControlPtr
 - arcs::common_interface, 505
- move_axis
 - arcs::common_interface::TrajConfig, 588
- moveCircle
 - MotionControl (运动规划与控制), 243
- moveCircle2
 - MotionControl (运动规划与控制), 245
- moveCircleWithAxisGroup
 - MotionControl (运动规划与控制), 245
- moveExtJoint
 - AxisInterface (外部轴), 34
- moveIntersection
 - MotionControl (运动规划与控制), 246
- moveJoint
 - MotionControl (运动规划与控制), 247
- moveJointWithAxisGroup
 - MotionControl (运动规划与控制), 248
- moveLine
 - MotionControl (运动规划与控制), 249
- moveLineWithAxisGroup
 - MotionControl (运动规划与控制), 250
- movePathBuffer
 - MotionControl (运动规划与控制), 251
- moveProcess
 - MotionControl (运动规划与控制), 251
- moveSpiral
 - MotionControl (运动规划与控制), 252
- moveSpline
 - MotionControl (运动规划与控制), 253
- name
 - arcs::common_interface::GripperStatus, 533
- needVibrationRecalib
 - RobotAlgorithm (机器人算法工具), 320
- newTask
 - RuntimeMachine (运行时管理), 454
- None
 - arcs::common_interface, 510
- nop
 - RuntimeMachine (运行时管理), 454
- notify
 - arcs::common_interface::Trace, 586
- obj_coord
 - arcs::common_interface::WObjectData, 591
- object_detection
 - arcs::common_interface::GripperStatus, 533
- OperationalModeType
 - arcs::common_interface, 509
- operator<<
 - arcs::common_interface, 513, 514
- orig

- arcs::common_interface::RobotSafetyParameterRange
 - RobotManage (机器人生命周期管理), 388
 - pauseRecordCache
 - RobotManage (机器人生命周期管理), 388
 - Payload
 - arcs::common_interface, 505
 - payloadCalculateFinished
 - RobotAlgorithm (机器人算法工具), 322
 - payloadIdentify
 - RobotAlgorithm (机器人算法工具), 323
 - payloadIdentify1
 - RobotAlgorithm (机器人算法工具), 323
 - PayloadIdentifyMoveAxis
 - arcs::common_interface, 510
 - payloadIdentifyTrajGenFinished
 - RobotAlgorithm (机器人算法工具), 324
- PEDSTRAL_ERRORS
 - error_stack.h, 611
 - hal_error.h, 615
- peek
 - Trace (日志与弹窗), 494
- plane
 - arcs::common_interface::RobotSafetyParameterRange, 567
 - arcs::common_interface::SpiralParameters, 582
- planes
 - arcs::common_interface::RobotSafetyParameterRange, 567
- points
 - arcs::common_interface::VibrationRecalibrationParameter, 590
- popup
 - Trace (日志与弹窗), 495
- pose_to
 - arcs::common_interface::CircleParameters, 524
- pose_via
 - arcs::common_interface::CircleParameters, 525
- poseAdd
 - Math (数学工具), 51
- poseAngleDistance
 - Math (数学工具), 51
- poseDistance
 - Math (数学工具), 52
- poseEqual
 - Math (数学工具), 52
- poseInverse
 - Math (数学工具), 53
- poseRotation
 - Math (数学工具), 53
- poseSub
 - Math (数学工具), 54
- poseToHomMatrix
 - Math (数学工具), 55
- poseTrans
 - Math (数学工具), 55
- poseTransInv
 - Math (数学工具), 56
- position
 - arcs::common_interface::GripperStatus, 533
- params
 - arcs::common_interface::RobotSafetyParameterRange, 566
- parse_error
 - arcs::common_interface, 508
- Path
 - arcs::common_interface, 510
- pathBlend3Points
 - RobotAlgorithm (机器人算法工具), 321
- PathBuffer_CubicSpline
 - arcs::common_interface, 510
- PathBuffer_JointBSpline
 - arcs::common_interface, 510
- PathBuffer_JointBSplineC
 - arcs::common_interface, 510
- PathBuffer_JointSpline
 - arcs::common_interface, 510
- PathBuffer_JointSplineC
 - arcs::common_interface, 510
- PathBuffer_TOPPRA
 - arcs::common_interface, 510
- pathBufferAlloc
 - MotionControl (运动规划与控制), 254
- pathBufferAppend
 - MotionControl (运动规划与控制), 255
- pathBufferEval
 - MotionControl (运动规划与控制), 255
- pathBufferFilter
 - MotionControl (运动规划与控制), 256
- pathBufferFree
 - MotionControl (运动规划与控制), 257
- pathBufferList
 - MotionControl (运动规划与控制), 258
- PathBufferType
 - arcs::common_interface, 510
- pathBufferValid
 - MotionControl (运动规划与控制), 258
- pathMovej
 - arcs::common_interface::RobotAlgorithm, 554
- pathMoveS
 - RobotAlgorithm (机器人算法工具), 322
- pathOffsetCoordinate
 - MotionControl (运动规划与控制), 259
- pathOffsetDisable
 - MotionControl (运动规划与控制), 260
- pathOffsetEnable
 - MotionControl (运动规划与控制), 260
- pathOffsetLimits
 - MotionControl (运动规划与控制), 261
- pathOffsetSet
 - MotionControl (运动规划与控制), 262
- pathOffsetSupv
 - MotionControl (运动规划与控制), 263
- pause
 - RuntimeMachine (运行时管理), 454
- pauseRecord

- power
 - arcs::common_interface::RobotSafetyParameterRange, 567
- poweroff
 - RobotManage (机器人生命周期管理), 389
- poweroffExtAxis
 - AxisInterface (外部轴), 35
- poweron
 - RobotManage (机器人生命周期管理), 390
- poweronExtAxis
 - AxisInterface (外部轴), 35
- preloadProgram
 - RuntimeMachine (运行时管理), 455
- qdmx
 - arcs::common_interface::RobotSafetyParameterRange, 567
- qmax
 - arcs::common_interface::RobotSafetyParameterRange, 567
- qmin
 - arcs::common_interface::RobotSafetyParameterRange, 567
- quaternionToRpy
 - Math (数学工具), 57
- r_velocity
 - arcs::common_interface::GripperStatus, 534
- recordCacheFree
 - RobotManage (机器人生命周期管理), 390
- reduced_entry_distance
 - arcs::common_interface::RobotSafetyParameterRange, 567
- reduced_entry_time
 - arcs::common_interface::RobotSafetyParameterRange, 567
- RefFrameType
 - arcs::common_interface, 510
- register_control.h
 - MB_ERR_ACKNOWLEDGE, 648
 - MB_ERR_DISCONNECTED, 648
 - MB_ERR_ILLEGAL_DATA_ADDRESS, 648
 - MB_ERR_ILLEGAL_DATA_VALUE, 648
 - MB_ERR_ILLEGAL_FUNCTION, 648
 - MB_ERR_NOT_INIT, 648
 - MB_ERR_SLAVE_DEVICE_BUSY, 648
 - MB_ERR_SLAVE_DEVICE_FAILURE, 648
 - ModbusErrorNum, 648
- RegisterControl
 - arcs::common_interface::RegisterControl, 550
- RegisterControl (寄存器操作), 59
 - addFixedModbusGroup, 62
 - addInt32RegEncoder, 63
 - addModbusEncoder, 64
 - clearNamedVariable, 64
 - deleteVirtualEncoder, 65
 - getBool, 65
 - getBoolInput, 66
 - getBoolOutput, 66
 - getDouble, 67
 - getDoubleInput, 68
 - getDoubleOutput, 68
 - getFloat, 69
 - getFloatInput, 70
 - getFloatOutput, 70
 - getInt16Register, 71
 - getInt16RegisterBit, 72
 - getInt32, 72
 - getInt32Input, 73
 - getInt32Output, 74
 - getModbusDeviceStatus, 74
 - getNamedVariableType, 75
 - getString, 75
 - getVecChar, 76
 - getVecDouble, 77
 - getVecFloat, 77
 - getVecInt32, 78
 - getWatchDogAction, 79
 - getWatchDogTimeout, 79
 - hasNamedVariable, 79
 - modbusAddSignal, 80
 - modbusDeleteAllSignals, 81
 - modbusDeleteSignal, 81
 - modbusGetSignalError, 82
 - modbusGetSignalErrors, 82
 - modbusGetSignalIndex, 83
 - modbusGetSignalNames, 83
 - modbusGetSignalStatus, 83
 - modbusGetSignalTypes, 84
 - modbusGetSignalValues, 84
 - modbusSendCustomCommand, 84
 - modbusSetDigitalInputAction, 86
 - modbusSetOutputRunstate, 86
 - modbusSetOutputRunstate1, 87
 - modbusSetOutputSignal, 88
 - modbusSetOutputSignal1, 89
 - modbusSetOutputSignalPulse, 89
 - modbusSetSignalUpdateFrequency, 90
 - setBool, 91
 - setBoolInput, 91
 - setBoolOutput, 92
 - setDouble, 93
 - setDoubleInput, 93
 - setDoubleOutput, 94
 - setFloat, 95
 - setFloatInput, 95
 - setFloatOutput, 96
 - setInt16Register, 97
 - setInt16RegisterBit, 97
 - setInt32, 98
 - setInt32Input, 99
 - setInt32Output, 99
 - setString, 100
 - setVecChar, 101
 - setVecDouble, 101
 - setVecFloat, 102
 - setVecInt32, 103

- setWatchDog, 103
 - variableUpdated, 104
- RegisterControlPtr
 - arcs::common_interface, 505
- releaseRobotBrake
 - RobotManage (机器人生命周期管理), 391
- remote_tool
 - arcs::common_interface::WObjectData, 591
- removeBreakPoint
 - RuntimeMachine (运行时管理), 455
- removeCollisionObject
 - RobotAlgorithm (机器人算法工具), 324
- resetDamping
 - ForceControl(力控模块), 118
- resetLpFilter
 - ForceControl(力控模块), 119
- resetTcpMaxLinearVelocity
 - MotionControl (运动规划与控制), 263
- restartInterfaceBoard
 - RobotManage (机器人生命周期管理), 391
- restoPath
 - MotionControl (运动规划与控制), 264
- restrict_elbow
 - arcs::common_interface::RobotSafetyParameterRange, 567
- ResultWithErrno
 - arcs::common_interface, 505
- ResultWithErrno1
 - arcs::common_interface, 505
- ResultWithErrno2
 - arcs::common_interface, 505
- ResultWithErrno3
 - arcs::common_interface, 505
- resume
 - RuntimeMachine (运行时管理), 456
- resumeMoveJoint
 - MotionControl (运动规划与控制), 264
- resumeMoveLine
 - MotionControl (运动规划与控制), 265
- resumeSpeedJoint
 - MotionControl (运动规划与控制), 266
- resumeSpeedLine
 - MotionControl (运动规划与控制), 267
- resumeStopJoint
 - MotionControl (运动规划与控制), 268
- resumeStopLine
 - MotionControl (运动规划与控制), 269
- returnValue2Str
 - arcs::common_interface, 514
- RobotAlgorithm
 - arcs::common_interface::RobotAlgorithm, 554
- RobotAlgorithm (机器人算法工具), 299
 - addCollisionBox, 302
 - calcJacobian, 302
 - calibrateTcpForceSensor, 303
 - calibrateTcpForceSensor2, 304
 - calibrateTcpForceSensor3, 305
 - calibVibrationParams, 305
 - calibVibrationParams1, 306
 - calibWorkpieceCoordinatePara, 307
 - forwardDynamics, 307
 - forwardDynamics1, 308
 - forwardKinematics, 308
 - forwardKinematics1, 309
 - forwardKinematicsAll, 310
 - forwardToolKinematics, 311
 - frictionModelIdentify, 312
 - generatePayloadIdentifyTraj, 313
 - getPayloadIdentifyResult, 313
 - getRobotConfiguration, 314
 - inverseKinematics, 315
 - inverseKinematics1, 316
 - inverseKinematics2, 317
 - inverseKinematicsAll, 318
 - inverseKinematicsAll1, 318
 - inverseToolKinematics, 319
 - inverseToolKinematicsAll, 320
 - needVibrationRecalib, 320
 - pathBlend3Points, 321
 - pathMoveS, 322
 - payloadCalculateFinished, 322
 - payloadIdentify, 323
 - payloadIdentify1, 323
 - payloadIdentifyTrajGenFinished, 324
 - removeCollisionObject, 324
 - validatePath, 325
 - validatePath1, 325
- RobotAlgorithmPtr
 - arcs::common_interface, 505
- RobotConfig
 - arcs::common_interface::RobotConfig, 558
- RobotConfig (机器人配置管理), 327
 - attachRobotBaseTo, 330
 - attachWeldingGun, 330
 - calcSafetyParametersChecksum, 331
 - confirmSafetyParameters, 331
 - enableEndCollisionCheck, 332
 - firmwareUpdate, 333
 - getBaseForceOffset, 334
 - getBaseForceSensorNames, 334
 - getCollisionLevel, 335
 - getCollisionStopType, 335
 - getCollisionThreshold, 336
 - getControlBoxType, 337
 - getCycletime, 337
 - getDefaultJointAcc, 338
 - getDefaultJointSpeed, 338
 - getDefaultToolAcc, 339
 - getDefaultToolSpeed, 340
 - getDof, 340
 - getFirmwareUpdateProcess, 341
 - getFreedriveDamp, 341
 - getGravity, 342
 - getHandguidDamp, 343
 - getHardwareCustomParameters, 343
 - getHomePosition, 344

- getJointMaxAccelerations, 344
- getJointMaxPositions, 345
- getJointMaxSpeeds, 345
- getJointMinPositions, 346
- getKinematicsCompensate, 346
- getKinematicsParam, 347
- getLimitJointMaxAccelerations, 348
- getLimitJointMaxPositions, 349
- getLimitJointMaxSpeeds, 349
- getLimitJointMinPositions, 350
- getLimitTcpMaxSpeed, 350
- getMountingPose, 351
- getName, 352
- getPayload, 352
- getRobotEmergencyStopSource, 353
- getRobotSubType, 353
- getRobotType, 354
- getSafeguardStopSource, 354
- getSafeguardStopType, 355
- getSafetyParametersChecksum, 355
- getSelectedTcpForceSensorName, 356
- getSlowDownFraction, 356
- getTcpForceOffset, 357
- getTcpForceSensorNames, 357
- getTcpForceSensorPose, 358
- getTcpMaxAccelerations, 358
- getTcpMaxSpeeds, 359
- getTcpOffset, 359
- hasBaseForceSensor, 360
- hasTcpForceSensor, 361
- isEndCollisionCheckEnabled, 361
- selectBaseForceSensor, 362
- selectTcpForceSensor, 362
- setBaseForceOffset, 363
- setCollisionLevel, 363
- setCollisionStopType, 364
- setCollisionThreshold, 365
- setFreedriveDamp, 366
- setGravity, 366
- setHandguidDamp, 367
- setHardwareCustomParameters, 368
- setHomePosition, 368
- setKinematicsCompensate, 369
- setMountingPose, 369
- setPayload, 370
- setPersistentParameters, 371
- setRobotZero, 372
- setSlowDownFraction, 372
- setTcpForceOffset, 373
- setTcpForceSensorPose, 373
- setTcpOffset, 374
- setToolInertial, 375
- setWorkObjectData, 375
- toolSpaceInRange, 375
- RobotConfigPtr
 - arcs::common_interface, 505
- RobotControlModeType
 - arcs::common_interface, 510
- RobotEmergencyStopType
 - arcs::common_interface, 511
- RobotInterface
 - arcs::common_interface::RobotInterface, 560
- RobotInterface(机器人模块), 105
 - getForceControl, 106
 - getIoControl, 106
 - getMotionControl, 106
 - getRobotAlgorithm, 107
 - getRobotConfig, 107
 - getRobotManage, 107
 - getRobotState, 108
 - getSyncMove, 108
 - getTrace, 108
- RobotInterfacePtr
 - arcs::common_interface, 505
- RobotManage
 - arcs::common_interface::RobotManage, 562
- RobotManage (机器人生命周期管理), 376
 - backdrive, 378
 - exitHandguideMode, 379
 - freedrive, 379
 - generateDiagnoseFile, 380
 - getHandguideFeature, 381
 - getHandguideFreeAxes, 381
 - getHandguideStatus, 381
 - getHandguideTrigger, 381
 - getOperationalMode, 382
 - getRecordCache, 382
 - getRobotControlMode, 383
 - handguideMode, 384
 - isBackdriveEnabled, 384
 - isFreedriveEnabled, 385
 - isHandguideEnabled, 385
 - isLinkModeEnabled, 386
 - isSimulationEnabled, 387
 - lockRobotBrake, 387
 - pauseRecord, 388
 - pauseRecordCache, 388
 - poweroff, 389
 - poweron, 390
 - recordCacheFree, 390
 - releaseRobotBrake, 391
 - restartInterfaceBoard, 391
 - setHandguideParams, 392
 - setLinkModeEnable, 393
 - setOperationalMode, 393
 - setSim, 394
 - setUnlockProtectiveStop, 395
 - startRecord, 396
 - startRecordCache, 396
 - startup, 397
 - stopRecord, 397
 - stopRecordCache, 398
- RobotManagePtr
 - arcs::common_interface, 505
- RobotModeType
 - arcs::common_interface, 511

- RobotMsgVector
 - arcs::common_interface, 506
- RobotSafetyParameterRange
 - arcs::common_interface::RobotSafetyParameterRange, 566
- RobotState
 - arcs::common_interface::RobotState, 574
- RobotState (机器人状态查询), 398
 - getActualTcpOffset, 402
 - getBaseForce, 402
 - getControlBoxHumidity, 402
 - getControlBoxTemperature, 403
 - getElbowPosistion, 403
 - getElbowVelocity, 404
 - getHandleFirmwareVersion, 405
 - getHandleHardwareVersion, 405
 - getHandleMode, 406
 - getHandleState, 406
 - getHandleUniqueId, 407
 - getJointAccelerations, 408
 - getJointContactTorques, 408
 - getJointCurrents, 409
 - getJointFirmwareVersions, 409
 - getJointGravityTorques, 410
 - getJointHardwareVersions, 411
 - getJointPositions, 411
 - getJointPositionsHistory, 412
 - getJointServoMode, 412
 - getJointSpeeds, 413
 - getJointState, 413
 - getJointTargetAccelerations, 414
 - getJointTargetCurrents, 414
 - getJointTargetPositions, 415
 - getJointTargetSpeeds, 416
 - getJointTargetTorques, 416
 - getJointTemperatures, 417
 - getJointTorqueSensors, 417
 - getJointUniqueIds, 418
 - getJointVoltages, 419
 - getMainCurrent, 419
 - getMainVoltage, 420
 - getMasterBoardFirmwareVersion, 420
 - getMasterBoardHardwareVersion, 421
 - getMasterBoardUniqueId, 422
 - getPedestalBrakeResTemp, 422
 - getPedestalFirmwareVersion, 423
 - getPedestalHardwareVersion, 423
 - getPedestalUniqueId, 424
 - getPowerlossBoardFirmwareVersion, 425
 - getPowerlossBoardHardwareVersion, 425
 - getPowerlossBoardUniqueId, 426
 - getRobotCurrent, 426
 - getRobotModeType, 427
 - getRobotVoltage, 428
 - getSafetyModeType, 428
 - getSlaveBoardFirmwareVersion, 429
 - getSlaveBoardHardwareVersion, 429
 - getSlaveBoardUniqueId, 430
 - getSlowDownLevel, 431
 - getTargetTcpPose, 431
 - getTcpForce, 432
 - getTcpForceSensors, 432
 - getTcpForceSensorStatus, 433
 - getTcpPose, 433
 - getTcpSpeed, 434
 - getTcpTargetForce, 435
 - getTcpTargetPose, 435
 - getTcpTargetSpeed, 436
 - getToolCommMode, 436
 - getToolFirmwareVersion, 437
 - getToolHardwareVersion, 438
 - getToolPose, 438
 - getToolUniqueId, 439
 - isCollisionOccurred, 439
 - isPowerOn, 440
 - isSteady, 440
 - isTeachPendantEnabled, 441
 - isToolFlangeEnabled, 442
 - isWithinSafetyLimits, 442
- RobotStatePtr
 - arcs::common_interface, 506
- rpyToQuaternion
 - Math (数学工具), 57
- rtm_error.h
 - RTM_ERRORS, 623
- RTM_ERRORS
 - error_stack.h, 611
 - rtm_error.h, 623
- Running
 - arcs::common_interface, 508
- runProgram
 - RuntimeMachine (运行时管理), 456
- RuntimeMachine
 - arcs::common_interface::RuntimeMachine, 579
- RuntimeMachine (运行时管理), 443
 - abort, 445
 - arbitraryResume, 445
 - clearBreakPoints, 446
 - clearPreloadPrograms, 446
 - deleteTask, 446
 - detachTask, 447
 - enterCritical, 447
 - exitCritical, 448
 - getAdvancePlanContext, 448
 - getAdvancePtr, 449
 - getExecutionStatus, 449
 - getInterpPtr, 449
 - getMainPtr, 450
 - getPlanContext, 450
 - getPreloadProgram, 451
 - getStatus, 451
 - getTaskQueueSize, 451
 - getTimer, 452
 - getTriggInterrupts, 452
 - gotoLine, 452
 - isTaskAlive, 453

- loadProgram, [453](#)
- newTask, [454](#)
- nop, [454](#)
- pause, [454](#)
- preloadProgram, [455](#)
- removeBreakPoint, [455](#)
- resume, [456](#)
- runProgram, [456](#)
- setBreakPoint, [456](#)
- setLabel, [457](#)
- setPlanContext, [457](#)
- setResumeWait, [458](#)
- start, [458](#)
- step, [459](#)
- stop, [459](#)
- switchTask, [460](#)
- timerDelete, [460](#)
- timerReset, [461](#)
- timerStart, [461](#)
- timerStop, [462](#)
- triggBegin, [462](#)
- triggEnd, [463](#)
- triggInterrupt, [463](#)
- RuntimeMachinePtr
 - arcs::common_interface, [506](#)
- RuntimeState
 - arcs::common_interface, [511](#)
- SafeguardStopType
 - arcs::common_interface, [511](#)
- SAFETY_CUBIC_NUM
 - type_def.h, [1077](#)
- safety_home
 - arcs::common_interface::RobotSafetyParameterRange, [567](#)
- safety_input_auto_safeguard_reset
 - arcs::common_interface::RobotSafetyParameterRange, [568](#)
- safety_input_auto_safeguard_stop
 - arcs::common_interface::RobotSafetyParameterRange, [568](#)
- safety_input_emergency_stop
 - arcs::common_interface::RobotSafetyParameterRange, [568](#)
- safety_input_handguide
 - arcs::common_interface::RobotSafetyParameterRange, [568](#)
- safety_input_operational_mode
 - arcs::common_interface::RobotSafetyParameterRange, [568](#)
- safety_input_reduced_mode
 - arcs::common_interface::RobotSafetyParameterRange, [568](#)
- safety_input_safeguard_reset
 - arcs::common_interface::RobotSafetyParameterRange, [568](#)
- safety_input_safeguard_stop
 - arcs::common_interface::RobotSafetyParameterRange, [568](#)
- safety_input_three_position_switch
 - arcs::common_interface::RobotSafetyParameterRange, [568](#)
- SAFETY_INTERFACE_BOARD_ERRORS
 - error_stack.h, [611](#)
 - hal_error.h, [615](#)
- safety_output_emergency_stop
 - arcs::common_interface::RobotSafetyParameterRange, [568](#)
- safety_output_not_emergency_stop
 - arcs::common_interface::RobotSafetyParameterRange, [569](#)
- safety_output_not_reduced_mode
 - arcs::common_interface::RobotSafetyParameterRange, [569](#)
- safety_output_reduced_mode
 - arcs::common_interface::RobotSafetyParameterRange, [569](#)
- safety_output_robot_moving
 - arcs::common_interface::RobotSafetyParameterRange, [569](#)
- safety_output_robot_not_stopping
 - arcs::common_interface::RobotSafetyParameterRange, [569](#)
- safety_output_robot_steady
 - arcs::common_interface::RobotSafetyParameterRange, [569](#)
- safety_output_safe_home
 - arcs::common_interface::RobotSafetyParameterRange, [569](#)
- safety_output_safetyguard_stop
 - arcs::common_interface::RobotSafetyParameterRange, [569](#)
- SAFETY_PARAM_SELECT_NUM
 - type_def.h, [1077](#)
- SAFETY_PLANES_NUM
 - type_def.h, [1078](#)
- SafetyInputAction
 - arcs::common_interface, [511](#)
- SafetyModeType
 - arcs::common_interface, [512](#)
- SafetyOutputRunState
 - arcs::common_interface, [512](#)
- segments
 - arcs::common_interface::RtdeRecipe, [576](#)
- selectBaseForceSensor
 - RobotConfig (机器人配置管理), [362](#)
- selectTcpForceSensor
 - RobotConfig (机器人配置管理), [362](#)
- Serial
 - arcs::common_interface::Serial, [580](#)
- Serial (串口通信), [463](#)
- serialClose, [464](#)
- serialOpen, [465](#)
- serialReadByte, [465](#)
- serialReadByteList, [466](#)
- serialReadString, [466](#)
- serialSendAllString, [467](#)

- serialSendByte, 467
- serialSendInt, 468
- serialSendLine, 468
- serialSendString, 469
- serialClose
 - Serial (串口通信), 464
- serialOpen
 - Serial (串口通信), 465
- SerialPtr
 - arcs::common_interface, 506
- serialReadByte
 - Serial (串口通信), 465
- serialReadByteList
 - Serial (串口通信), 466
- serialReadString
 - Serial (串口通信), 466
- serialSendAllString
 - Serial (串口通信), 467
- serialSendByte
 - Serial (串口通信), 467
- serialSendInt
 - Serial (串口通信), 468
- serialSendLine
 - Serial (串口通信), 468
- serialSendString
 - Serial (串口通信), 469
- server_error
 - arcs::common_interface, 508
- servoCartesian
 - MotionControl (运动规划与控制), 270
- servoCartesianWithAxes
 - MotionControl (运动规划与控制), 271
- servoCartesianWithAxisGroup
 - arcs::common_interface::MotionControl, 547
- servoJoint
 - MotionControl (运动规划与控制), 272
- servoJointWithAxes
 - MotionControl (运动规划与控制), 273
- servoJointWithAxisGroup
 - arcs::common_interface::MotionControl, 547
- setAnalogOutputAfterEStopDefault
 - IoControl (IO 输入输出控制), 185
- setBaseForceOffset
 - RobotConfig (机器人配置管理), 363
- setBool
 - RegisterControl (寄存器操作), 91
- setBoolInput
 - RegisterControl (寄存器操作), 91
- setBoolOutput
 - RegisterControl (寄存器操作), 92
- setBreakPoint
 - RuntimeMachine (运行时管理), 456
- setCirclePathMode
 - MotionControl (运动规划与控制), 274
- setCollisionLevel
 - RobotConfig (机器人配置管理), 363
- setCollisionStopType
 - RobotConfig (机器人配置管理), 364
- setCollisionThreshold
 - RobotConfig (机器人配置管理), 365
- setCondActive
 - ForceControl(力控模块), 119
- setCondAdvanced
 - ForceControl(力控模块), 120
- setCondCylinder
 - ForceControl(力控模块), 120
- setCondDistance
 - ForceControl(力控模块), 121
- setCondForce
 - ForceControl(力控模块), 122
- setCondOrient
 - ForceControl(力控模块), 123
- setCondPlane
 - ForceControl(力控模块), 124
- setCondSphere
 - ForceControl(力控模块), 124
- setCondTcpSpeed
 - ForceControl(力控模块), 125
- setConfigurableDigitalInputAction
 - IoControl (IO 输入输出控制), 186
- setConfigurableDigitalOutput
 - IoControl (IO 输入输出控制), 187
- setConfigurableDigitalOutputAfterEStop
 - IoControl (IO 输入输出控制), 188
- setConfigurableDigitalOutputPulse
 - IoControl (IO 输入输出控制), 189
- setConfigurableDigitalOutputRunstate
 - IoControl (IO 输入输出控制), 189
- setConveyorTrackCompensate
 - MotionControl (运动规划与控制), 274
- setConveyorTrackEncoder
 - MotionControl (运动规划与控制), 275
- setConveyorTrackLimit
 - MotionControl (运动规划与控制), 276
- setConveyorTrackSensorOffset
 - MotionControl (运动规划与控制), 277
- setConveyorTrackStartWindow
 - MotionControl (运动规划与控制), 277
- setConveyorTrackSyncSeparation
 - MotionControl (运动规划与控制), 278
- setConveyorTrackVelThreshold
 - MotionControl (运动规划与控制), 279
- setDamping
 - ForceControl(力控模块), 126
- setDigitalInputActionDefault
 - IoControl (IO 输入输出控制), 190
- setDigitalOutputAfterEStopDefault
 - IoControl (IO 输入输出控制), 191
- setDigitalOutputRunstate
 - IoControl (IO 输入输出控制), 191
- setDigitalOutputRunstateDefault
 - IoControl (IO 输入输出控制), 192
- setDouble
 - RegisterControl (寄存器操作), 93
- setDoubleInput
 - RegisterControl (寄存器操作), 93

- setDoubleOutput
 - RegisterControl (寄存器操作), 94
- setDynamicModel
 - ForceControl(力控模块), 126
- setDynamicModel1
 - ForceControl(力控模块), 127
- setDynamicModelContact
 - ForceControl(力控模块), 128
- setDynamicModelInsert
 - ForceControl(力控模块), 129
- setDynamicModelSearch
 - ForceControl(力控模块), 129
- setEncDecoderType
 - IoControl (IO 输入输出控制), 193
- setEncTickCount
 - IoControl (IO 输入输出控制), 193
- setEndPath
 - MotionControl (运动规划与控制), 280
- setEqradius
 - MotionControl (运动规划与控制), 281
- setExtAxisAcceleration
 - AxisInterface (外部轴), 35
- setExtAxisMaxAccelerationLimit
 - AxisInterface (外部轴), 35
- setExtAxisMaxPositionLimit
 - AxisInterface (外部轴), 35
- setExtAxisMaxVelocityLimit
 - AxisInterface (外部轴), 36
- setExtAxisMountingPose
 - AxisInterface (外部轴), 36
- setExtAxisType
 - AxisInterface (外部轴), 36
- setExtAxisVelocity
 - AxisInterface (外部轴), 36
- setExtAxisZero
 - AxisInterface (外部轴), 37
- setExtMinPositionLimit
 - AxisInterface (外部轴), 37
- setFloat
 - RegisterControl (寄存器操作), 95
- setFloatInput
 - RegisterControl (寄存器操作), 95
- setFloatOutput
 - RegisterControl (寄存器操作), 96
- setFreedriveDamp
 - RobotConfig (机器人配置管理), 366
- setFuturePointSamplePeriod
 - MotionControl (运动规划与控制), 281
- setGravity
 - RobotConfig (机器人配置管理), 366
- setGroupOutput
 - IoControl (IO 输入输出控制), 194
- setHandguidDamp
 - RobotConfig (机器人配置管理), 367
- setHandguideParams
 - RobotManage (机器人生命周期管理), 392
- setHardwareCustomParameters
 - RobotConfig (机器人配置管理), 368
- setHomePosition
 - RobotConfig (机器人配置管理), 368
- setInt16Register
 - RegisterControl (寄存器操作), 97
- setInt16RegisterBit
 - RegisterControl (寄存器操作), 97
- setInt32
 - RegisterControl (寄存器操作), 98
- setInt32Input
 - RegisterControl (寄存器操作), 99
- setInt32Output
 - RegisterControl (寄存器操作), 99
- setKinematicsCompensate
 - RobotConfig (机器人配置管理), 369
- setLabel
 - RuntimeMachine (运行时管理), 457
- setLinkModeEnable
 - RobotManage (机器人生命周期管理), 393
- setLookAheadSize
 - MotionControl (运动规划与控制), 282
- setLpFilter
 - ForceControl(力控模块), 130
- setMountingPose
 - RobotConfig (机器人配置管理), 369
- setOperationalMode
 - RobotManage (机器人生命周期管理), 393
- setPayload
 - RobotConfig (机器人配置管理), 370
- setPersistentParameters
 - RobotConfig (机器人配置管理), 371
- setPlanContext
 - RuntimeMachine (运行时管理), 457
- setReductionRatio
 - AxisInterface (外部轴), 37
- setResumeStartPoint
 - MotionControl (运动规划与控制), 282
- setResumeWait
 - RuntimeMachine (运行时管理), 458
- setRobotZero
 - RobotConfig (机器人配置管理), 372
- setServoMode
 - MotionControl (运动规划与控制), 283
- setServoModeSelect
 - MotionControl (运动规划与控制), 284
- setSim
 - RobotManage (机器人生命周期管理), 394
- setSlowDownFraction
 - RobotConfig (机器人配置管理), 372
- setSoftFloatParams
 - ForceControl(力控模块), 131
- setSpeedFraction
 - MotionControl (运动规划与控制), 285
- setStandardAnalogInputDomain
 - IoControl (IO 输入输出控制), 195
- setStandardAnalogOutput
 - IoControl (IO 输入输出控制), 195
- setStandardAnalogOutputAfterEStop
 - IoControl (IO 输入输出控制), 196

- setStandardAnalogOutputDomain
 - IoControl (IO 输入输出控制), 197
- setStandardAnalogOutputRunstate
 - IoControl (IO 输入输出控制), 198
- setStandardDigitalInputAction
 - IoControl (IO 输入输出控制), 199
- setStandardDigitalOutput
 - IoControl (IO 输入输出控制), 200
- setStandardDigitalOutputAfterEStop
 - IoControl (IO 输入输出控制), 200
- setStandardDigitalOutputPulse
 - IoControl (IO 输入输出控制), 201
- setStandardDigitalOutputRunstate
 - IoControl (IO 输入输出控制), 202
- setString
 - RegisterControl (寄存器操作), 100
- setSupvForce
 - ForceControl(力控模块), 131
- setSupvOrient
 - ForceControl(力控模块), 132
- setSupvPosBox
 - ForceControl(力控模块), 133
- setSupvPosCylinder
 - ForceControl(力控模块), 133
- setSupvPosSphere
 - ForceControl(力控模块), 134
- setSupvReoriSpeed
 - ForceControl(力控模块), 134
- setSupvTcpSpeed
 - ForceControl(力控模块), 135
- setTargetForce
 - ForceControl(力控模块), 136
- setTcpForceOffset
 - RobotConfig (机器人配置管理), 373
- setTcpForceSensorPose
 - RobotConfig (机器人配置管理), 373
- setTcpMaxLinearVelocity
 - MotionControl (运动规划与控制), 285
- setTcpOffset
 - RobotConfig (机器人配置管理), 374
- setTimeOptimalEnable
 - MotionControl (运动规划与控制), 286
- setToolAnalogInputDomain
 - IoControl (IO 输入输出控制), 203
- setToolAnalogOutput
 - IoControl (IO 输入输出控制), 204
- setToolAnalogOutputDomain
 - IoControl (IO 输入输出控制), 204
- setToolAnalogOutputRunstate
 - IoControl (IO 输入输出控制), 205
- setToolDigitalInputAction
 - IoControl (IO 输入输出控制), 206
- setToolDigitalOutput
 - IoControl (IO 输入输出控制), 207
- setToolDigitalOutputPulse
 - IoControl (IO 输入输出控制), 208
- setToolDigitalOutputRunstate
 - IoControl (IO 输入输出控制), 209
- setToolInertial
 - RobotConfig (机器人配置管理), 375
- setToolIoInput
 - IoControl (IO 输入输出控制), 210
- setToolVoltageOutputDomain
 - IoControl (IO 输入输出控制), 210
- setUnlockProtectiveStop
 - RobotManage (机器人生命周期管理), 395
- setVecChar
 - RegisterControl (寄存器操作), 101
- setVecDouble
 - RegisterControl (寄存器操作), 101
- setVecFloat
 - RegisterControl (寄存器操作), 102
- setVecInt32
 - RegisterControl (寄存器操作), 103
- setWatchDog
 - RegisterControl (寄存器操作), 103
- setWorkObjectData
 - RobotConfig (机器人配置管理), 375
- setWorkObjectHold
 - MotionControl (运动规划与控制), 287
- shape
 - arcs::common_interface::Enveloping, 526
- size
 - arcs::common_interface::RobotSafetyParameterRange, 569
- Socket
 - arcs::common_interface::Socket, 581
- Socket (socket 网络通信), 469
 - socketClose, 470
 - socketHasConnected, 471
 - socketOpen, 471
 - socketReadAllString, 472
 - socketReadAsciiFloat, 472
 - socketReadBinaryInteger, 473
 - socketReadByteList, 473
 - socketReadString, 474
 - socketSendAllString, 475
 - socketSendByte, 475
 - socketSendInt, 476
 - socketSendLine, 476
 - socketSendString, 477
- socketClose
 - Socket (socket 网络通信), 470
- socketHasConnected
 - Socket (socket 网络通信), 471
- socketOpen
 - Socket (socket 网络通信), 471
- SocketPtr
 - arcs::common_interface, 506
- socketReadAllString
 - Socket (socket 网络通信), 472
- socketReadAsciiFloat
 - Socket (socket 网络通信), 472
- socketReadBinaryInteger
 - Socket (socket 网络通信), 473
- socketReadByteList

- Socket (socket 网络通信), 473
- socketReadString
 - Socket (socket 网络通信), 474
- socketSendAllString
 - Socket (socket 网络通信), 475
- socketSendByte
 - Socket (socket 网络通信), 475
- socketSendInt
 - Socket (socket 网络通信), 476
- socketSendLine
 - Socket (socket 网络通信), 476
- socketSendString
 - Socket (socket 网络通信), 477
- softFloatDisable
 - ForceControl(力控模块), 137
- softFloatEnable
 - ForceControl(力控模块), 137
- source
 - arcs::common_interface::RobotMsg, 563
- speedChangeDisable
 - ForceControl(力控模块), 138
- speedChangeEnable
 - ForceControl(力控模块), 138
- speedChangeTune
 - ForceControl(力控模块), 139
- speedExtJoint
 - AxisInterface (外部轴), 37
- speedFractionCritical
 - MotionControl (运动规划与控制), 287
- speedFractionCritical1
 - MotionControl (运动规划与控制), 288
- speedJoint
 - MotionControl (运动规划与控制), 289
- speedLine
 - MotionControl (运动规划与控制), 290
- Sphere
 - arcs::common_interface, 506
- spiral
 - arcs::common_interface::CircleParameters, 525
 - arcs::common_interface::SpiralParameters, 583
- StandardInputAction
 - arcs::common_interface, 512
- StandardOutputRunState
 - arcs::common_interface, 512
- start
 - RuntimeMachine (运行时管理), 458
- Starting
 - arcs::common_interface, 508
- startMove
 - MotionControl (运动规划与控制), 291
- startRecord
 - RobotManage (机器人生命周期管理), 396
- startRecordCache
 - RobotManage (机器人生命周期管理), 396
- startup
 - RobotManage (机器人生命周期管理), 397
- status_code
 - arcs::common_interface::GripperStatus, 534
- step
 - RuntimeMachine (运行时管理), 459
- stiff_param
 - arcs::common_interface::VibrationRecalibrationParameter, 590
- stiff_section
 - arcs::common_interface::VibrationRecalibrationParameter, 590
- stl_path
 - arcs::common_interface::Enveloping, 526
- stop
 - RuntimeMachine (运行时管理), 459
- stop_distance
 - arcs::common_interface::RobotSafetyParameterRange, 569
- stop_time
 - arcs::common_interface::RobotSafetyParameterRange, 570
- stopExtJoint
 - AxisInterface (外部轴), 38
- stopFollowAnotherAxis
 - AxisInterface (外部轴), 38
- stopJoint
 - MotionControl (运动规划与控制), 292
- stopLine
 - MotionControl (运动规划与控制), 292
- stopMove
 - MotionControl (运动规划与控制), 293
- Stopped
 - arcs::common_interface, 508
- stopRecord
 - RobotManage (机器人生命周期管理), 397
- stopRecordCache
 - RobotManage (机器人生命周期管理), 398
- storePath
 - MotionControl (运动规划与控制), 294
- str2ErrorCode
 - arcs::error_stack, 515
- Stropping
 - arcs::common_interface, 508
- switchTask
 - RuntimeMachine (运行时管理), 460
- SyncMove
 - arcs::common_interface::SyncMove, 585
- SyncMove (同步运动控制和轴组管理), 477
 - axisGroupAdd, 479
 - axisGroupAddAxis, 480
 - axisGroupDelete, 480
 - axisGroupGetActualPositions, 481
 - axisGroupGetAxisIndex, 481
 - axisGroupGetAxisName, 481
 - axisGroupGetTargetPositions, 482
 - axisGroupMoveJoint, 482
 - axisGroupOffsetPositions, 482
 - axisGroupSpeedJoint, 483
 - axisGroupUpdateAxis, 483
 - frameAdd, 483
 - frameAttach, 484

- frameConvertPose, 484
- frameDelete, 484
- frameDeleteAll, 485
- frameExist, 485
- frameGetChildren, 485
- frameGetParent, 486
- frameGetPose, 486
- frameMove, 486
- isSyncMoveOn, 487
- syncMoveOff, 487
- syncMoveOn, 488
- syncMoveResume, 488
- syncMoveSegment, 488
- syncMoveSuspend, 489
- syncMoveUndo, 489
- waitSyncTasks, 489
- syncMoveOff
 - SyncMove (同步运动控制和轴组管理), 487
- syncMoveOn
 - SyncMove (同步运动控制和轴组管理), 488
- SyncMovePtr
 - arcs::common_interface, 506
- syncMoveResume
 - SyncMove (同步运动控制和轴组管理), 488
- syncMoveSegment
 - SyncMove (同步运动控制和轴组管理), 488
- syncMoveSuspend
 - SyncMove (同步运动控制和轴组管理), 489
- syncMoveUndo
 - SyncMove (同步运动控制和轴组管理), 489
- system_error.h
 - SYSTEM_ERRORS, 627
- SYSTEM_ERRORS
 - error_stack.h, 611
 - system_error.h, 627
- SystemInfo
 - arcs::common_interface::SystemInfo, 585
- SystemInfo (系统信息), 490
 - getControlSoftwareBuildDate, 491
 - getControlSoftwareFullVersion, 491
 - getControlSoftwareVersionCode, 491
 - getControlSoftwareVersionHash, 492
 - getControlSystemTime, 492
 - getInterfaceVersionCode, 493
- SystemInfoPtr
 - arcs::common_interface, 506
- TaskFrameType
 - arcs::common_interface, 512
- TaskSet
 - arcs::common_interface, 506
- tcp_force
 - arcs::common_interface::RobotSafetyParameterRange, 570
- tcp_speed
 - arcs::common_interface::RobotSafetyParameterRange, 570
- tcpOffsetIdentify
 - Math (数学工具), 58
- temperature
 - arcs::common_interface::GripperStatus, 534
- textmsg
 - Trace (日志与弹窗), 495
- timerDelete
 - RuntimeMachine (运行时管理), 460
- timerReset
 - RuntimeMachine (运行时管理), 461
- timerStart
 - RuntimeMachine (运行时管理), 461
- timerStop
 - RuntimeMachine (运行时管理), 462
- timestamp
 - arcs::common_interface::RobotMsg, 563
- to_server
 - arcs::common_interface::RtdeRecipe, 576
- Tool
 - arcs::common_interface, 510
- tool_azimuth
 - arcs::common_interface::RobotSafetyParameterRange, 570
- TOOL_CONFIGURATION_NUM
 - type_def.h, 1078
- tool_deviation
 - arcs::common_interface::RobotSafetyParameterRange, 570
- TOOL_ERRORS
 - error_stack.h, 611
 - hal_error.h, 616
- tool_inclination
 - arcs::common_interface::RobotSafetyParameterRange, 570
- tool_orientation
 - arcs::common_interface::RobotSafetyParameterRange, 570
- toolContact
 - ForceControl(力控模块), 139
- tools
 - arcs::common_interface::RobotSafetyParameterRange, 570
- toolSpaceInRange
 - RobotConfig (机器人配置管理), 375
- torque
 - arcs::common_interface::GripperStatus, 534
- tp_3pe_for_handguide
 - arcs::common_interface::RobotSafetyParameterRange, 570
- Trace
 - arcs::common_interface::Trace, 586
- Trace (日志与弹窗), 494
 - alarm, 494
 - peek, 494
 - popup, 495
 - textmsg, 495
- TraceLevel
 - arcs::common_interface, 513
- TracePtr
 - arcs::common_interface, 506

- trackCartesian
 - MotionControl (运动规划与控制), 294
- trackJoint
 - MotionControl (运动规划与控制), 295
- transferRefFrame
 - Math (数学工具), 58
- triggBegin
 - RuntimeMachine (运行时管理), 462
- triggEnd
 - RuntimeMachine (运行时管理), 463
- trigger
 - arcs::common_interface::RtdeRecipe, 576
- trigger_planes
 - arcs::common_interface::RobotSafetyParameterRange, 571
- triggInterrupt
 - RuntimeMachine (运行时管理), 463
- type
 - arcs::common_interface::AuboException, 520
- type_def.h
 - CARTESIAN_DOF, 1072
 - DECL_TO_STRING_FUNC, 1072
 - ENUM_AuboErrorCodes_DECLARES, 1072
 - ENUM_AxisModeType_DECLARES, 1072
 - ENUM_EnvelopingShape_DECLARES, 1072
 - ENUM_HandleModeType_DECLARES, 1073
 - ENUM_HandleStateType_DECLARES, 1073
 - ENUM_ITEM, 1073, 1074
 - ENUM_JointServoModeType_DECLARES, 1074
 - ENUM_JointStateType_DECLARES, 1074
 - ENUM_OperationalModeType_DECLARES, 1074
 - ENUM_PayloadIdentifyMoveAxis_DECLARES, 1074
 - ENUM_RobotControlModeType_DECLARES, 1075
 - ENUM_RobotEmergencyStopType_DECLARES, 1075
 - ENUM_RobotModeType_DECLARES, 1075
 - ENUM_RuntimeState_DECLARES, 1075
 - ENUM_SafeguardedStopType_DECLARES, 1075
 - ENUM_SafetyInputAction_DECLARES, 1076
 - ENUM_SafetyModeType_DECLARES, 1076
 - ENUM_SafetyOutputRunState_DECLARES, 1076
 - ENUM_StandardInputAction_DECLARES, 1076
 - ENUM_StandardOutputRunState_DECLARES, 1077
 - ENUM_TaskFrameType_DECLARES, 1077
 - ENUM_TraceLevel_DECLARES, 1077
 - SAFETY_CUBIC_NUM, 1077
 - SAFETY_PARAM_SELECT_NUM, 1077
 - SAFETY_PLANES_NUM, 1078
 - TOOL_CONFIGURATION_NUM, 1078
- unwindEncDeltaTickCount
 - IoControl (IO 输入输出控制), 211
- upper_joint_bound
 - arcs::common_interface::TrajConfig, 588
- user_coord
 - arcs::common_interface::WObjectData, 591
- v
 - arcs::common_interface::CircleParameters, 525
- validatePath
 - RobotAlgorithm (机器人算法工具), 325
- validatePath1
 - RobotAlgorithm (机器人算法工具), 325
- variableUpdated
 - RegisterControl (寄存器操作), 104
- Vector3d
 - arcs::common_interface, 506
- Vector3f
 - arcs::common_interface, 506
- Vector4d
 - arcs::common_interface, 507
- Vector4f
 - arcs::common_interface, 507
- Vector6f
 - arcs::common_interface, 507
- velocity
 - arcs::common_interface::GripperStatus, 534
- voltage
 - arcs::common_interface::GripperStatus, 534
- waitSyncTasks
 - SyncMove (同步运动控制和轴组管理), 489
- weaveEnd
 - MotionControl (运动规划与控制), 296
- weaveStart
 - MotionControl (运动规划与控制), 296
- weaveUpdateParameters
 - MotionControl (运动规划与控制), 299
- what
 - arcs::common_interface::AuboException, 520