



## 睿尔曼机械臂接口函数说明 (MATLAB) V1.7



---

睿尔曼智能科技（北京）有限公司



## 文件修订记录：

版本号	时间	备注
V1.0	2023-09-11	拟制
V1.1	2023-11-27	文档勘误 修复 MATLAB API 读线圈接口 coils_data 为输出参数 完善注释
V1.2	2023-12-27	新增电子围栏相关接口 增加 modbus-TCP 主站和 modbus-RTU 从站接口 新增查询示教参考坐标系接口 新增自碰撞安全检测相关接口 新增读取软件信息接口 新增关节驱动器转速、加速度以及最大最小限位设置查询接口 优化开始复合模式拖动示教、movej、movei、movec、movej_p、一键设置关节限位等接口 新增查询夹爪信息接口 新增设置、查询机械臂仿真/真实模式
V1.3	2024-3-30	适配 GEN72、ECO63 机械臂 运动接口优化，适配控制器 1.5.0 软件版本 修复灵巧手相关接口阻塞模式不生效问题 修复夹爪夹取松开速度低时执行成功返回 7 的问题，新增超时时间设置 修改 SensorType 枚举项名称 B、ZF、SF 为 _B、_ZF、_SF 优化电子围栏与虚拟墙数据管理接口 新增工作坐标系和工具坐标系的更新接口 新增工具坐标系包络参数相关接口 新增虚拟墙相关接口 新增在线编程文件信息修改接口



		新增 IO 控制默认运行在线编程文件接口 新增全局点位相关接口
V1.4	2024-5-30	新增样条曲线运动 Moves 新增读多个输入寄存器接口 Read_Multiple_Input_Registers 发送在线编程文件新增保存到控制器 id 参数
V1.5	2024-7-3	适配 GEN72 型号机械臂
V1.6	2024-8-5	修复使用控制器 1.6.0 时无法使用 UDP 的问题 修改运动接口交融半径
V1.7	2024-8-19	修复交融半径调用失败 新增计算平移、旋转运动位姿算法接口 添加交融半径使用注意事项 适配控制器 1.6.0 主动上报接口自定义项配置、获取及上报信息获取



# 目录

1. 简介 .....	18
2. 功能介绍 .....	18
3. 使用说明 .....	19
4. 数据类型说明 .....	22
4.1. 控制器错误类型 .....	22
4.2. 关节错误类型 .....	23
4.3. API 错误类型 .....	24
4.4. 结构化数据类定义 .....	25
4.4.1. 位姿结构体 Pose--clib.rm_service.Pose .....	25
4.4.2. 坐标系结构体 FRAME--clib.rm_service.FRAME .....	26
4.4.3. 关节状态结构体 JOINT_STATE--clib.rm_service.JOINT_STATE .....	27
4.4.4. 实时推送机械臂状态结构体 RobotStatus--clib.rm_service.RobotStatus..	28
5. 接口库函数说明 .....	30
5.1. 连接相关函数 .....	30
5.1.1. API 初始化 Service_RM_API_Init .....	30
5.1.2. API 反初始化 Service_RM_API_UnInit .....	30
5.1.3. 查询 API 版本信息 Service_API_Version .....	30
5.1.4. 连接机械臂 Service_Arm_Socket_Start .....	31
5.1.5. 查询连接状态 Service_Arm_Socket_State .....	31
5.1.6. 关闭连接 Service_Arm_Socket_Close .....	31



5.2. 关节配置函数 .....	32
5.2.1. 设置关节最大速度 Service_Set_Joint_Speed .....	33
5.2.2. 设置关节最大加速度 Service_Set_Joint_Acc .....	33
5.2.3. 设置关节最小限位 Service_Set_Joint_Min_Pos .....	34
5.2.4. 设置关节最大限位 Service_Set_Joint_Max_Pos .....	35
5.2.5. 设置关节最大速度（驱动器） Service_Set_Joint_Drive_Speed .....	35
5.2.6. 设置关节最大加速度（驱动器） Service_Set_Joint_Drive_Acc .....	36
5.2.7. 设置关节最小限位（驱动器） Service_Set_Joint_Drive_Min_Pos .....	37
5.2.8. 设置关节最大限位（驱动器） Service_Set_Joint_Drive_Max_Pos .....	37
5.2.9. 设置关节使能 Service_Set_Joint_EN_State .....	38
5.2.10. 设置关节零位 Service_Set_Joint_Zero_Pos .....	38
5.2.11. 清除关节错误代码 Service_Set_Joint_Err_Clear .....	39
5.2.12. 自动设置限位 Service_Auto_Set_Joint_Limit .....	39
5.3. 关节参数查询函数 .....	40
5.3.1. 查询关节最大速度 Service_Get_Joint_Speed .....	40
5.3.2. 查询关节最大加速度 Service_Get_Joint_Acc .....	40
5.3.3. 获取关节最小限位 Service_Get_Joint_Min_Pos .....	41
5.3.4. 获取关节最大限位 Service_Get_Joint_Max_Pos .....	41
5.3.5. 查询关节最大速度（驱动器） Service_Get_Joint_Drive_Speed .....	42
5.3.6. 查询关节最大加速度（驱动器） Service_Get_Joint_Acc .....	42
5.3.7. 获取关节最小限位（驱动器） Service_Get_Joint_Drive_Min_Pos .....	42
5.3.8. 获取关节最大限位（驱动器） Service_Get_Joint_Drive_Max_Pos .....	43



5.3.9. 获取关节使能状态 Service_Get_Joint_EN_State .....	43
5.3.10. 获取关节错误代码 Service_Get_Joint_Err_Flag .....	44
5.3.11. 查询关节软件版本号 Service_Get_Joint_Software_Version .....	44
5.4. 机械臂末端运动参数配置 .....	46
5.4.1. 设置末端最大线速度 Service_Set_Arm_Line_Speed .....	46
5.4.2. 设置末端最大线加速度 Service_Set_Arm_Line_Acc .....	47
5.4.3. 设置末端最大角速度 Service_Set_Arm_Angular_Speed .....	47
5.4.4. 设置末端最大角加速度 Service_Set_Arm_Angular_Acc .....	48
5.4.5. 获取末端最大线速度 Service_Get_Arm_Line_Speed .....	49
5.4.6. 获取最大末端线加速度 Service_Get_Arm_Line_Acc .....	49
5.4.7. 获取末端最大角速度 Service_Get_Arm_Angular_Speed .....	49
5.4.8. 获取末端最大角加速度 Service_Get_Arm_Angular_Acc .....	50
5.4.9. 设置机械臂末端参数为初始值 Service_Set_Arm_Tip_Init .....	50
5.4.10. 设置碰撞等级 Service_Set_Collision_Stage .....	51
5.4.11. 查询碰撞等级 Service_Get_Collision_Stage .....	51
5.4.12. 设置关节零位补偿角度 Service_Set_Joint_Zero_Offset .....	52
5.5. 机械臂末端接口板 .....	54
5.5.1. 查询末端接口板软件版本号 Service_Get_Tool_Software_Version .....	54
5.6. 工具坐标系设置 .....	54
5.6.1. 标定点位 Service_Auto_Set_Tool_Frame .....	54
5.6.2. 生成工具坐标系 Service_Generate_Auto_Tool_Frame .....	55
5.6.3. 手动设置工具坐标系 Service_Manual_Set_Tool_Frame .....	56



5.6.4. 切换当前工具坐标系 Service_Change_Tool_Frame .....	57
5.6.5. 删除指定工具坐标系 Service_Delete_Tool_Frame .....	58
5.6.6. 修改指定工具坐标系 Service_Update_Tool_Frame .....	58
5.6.7. 设置工具坐标系的包络参数 Service_Set_Tool_Envelope .....	59
5.6.8. 获取工具坐标系的包络参数 Service_Get_Tool_Envelope .....	60
5.7. 工具坐标系查询 .....	61
5.7.1. 获取当前工具坐标系 Service_Get_Current_Tool_Frame .....	61
5.7.2. 获取指定工具坐标系 Service_Get_Given_Tool_Frame .....	61
5.7.3. 获取所有工具坐标系名称 Service_Get_All_Tool_Frame .....	62
5.8. 工作坐标系设置 .....	64
5.8.1. 自动设置工作坐标系 Service_Auto_Set_Work_Frame .....	64
5.8.2. 手动设置工作坐标系 Service_Manual_Set_Work_Frame .....	65
5.8.3. 切换当前工作坐标系 Service_Change_Work_Frame .....	66
5.8.4. 删除指定工作坐标系 Service_Delete_Work_Frame .....	66
5.8.5. 修改指定工作坐标系 Service_Update_Work_Frame .....	67
5.9. 工作坐标系查询 .....	68
5.9.1. 获取当前工作坐标系 Service_Get_Current_Work_Frame .....	68
5.9.2. 获取指定工作坐标系 Service_Get_Given_Work_Frame .....	68
5.9.3. 获取所有工作坐标系名称 Service_Get_All_Work_Frame .....	69
5.10. 机械臂状态查询 .....	69
5.10.1. 获取机械臂当前状态 Service_Get_Current_Arm_State .....	69
5.10.2. 获取关节温度 Service_Get_Joint_Temperature .....	70



5.10.3. 获取关节电流 Service_Get_Joint_Current .....	71
5.10.4. 获取关节电压 Service_Get_Joint_Voltage .....	71
5.10.5. 获取关节当前角度 Service_Get_Joint_Degree .....	72
5.10.6. 获取所有状态 Service_Get_Arm_All_State .....	72
5.10.7. 获取轨迹规划计数 Service_Get_Arm_Plan_Num .....	72
5.11. 机械臂初始位姿 .....	74
5.11.1. 设置初始位姿角度 Service_Set_Arm_Init_Pose .....	74
5.11.2. 获取初始位姿角度 Service_Get_Arm_Init_Pose .....	75
5.11.3. 设置安装角度 Service_Set_Install_Pose .....	75
5.11.4. 查询安装角度 Service_Get_Install_Pose .....	76
5.12. 机械臂运动规划 .....	77
5.12.1. 关节空间运动 Service_Movej_Cmd .....	78
5.12.2. 笛卡尔空间直线运动 Service_MoveL_Cmd .....	79
5.12.3. 笛卡尔空间圆弧运动 Service_Movec_Cmd .....	80
5.12.4. 关节角度 CANFD 透传 Service_Movej_CANFD .....	82
5.12.5. 位姿 CANFD 透传 Service_Movep_CANFD .....	84
5.12.6. 计算环绕运动位姿 Service_MoveRotate_Cmd .....	84
5.12.7. 沿工具端位姿移动 Service_MoveCartesianTool_Cmd .....	85
5.12.8. 快速急停 Service_Move_Stop_Cmd .....	87
5.12.9. 暂停当前规划 Service_Move_Pause_Cmd .....	87
5.12.10. 继续当前轨迹 Service_Move_Continue_Cmd .....	88
5.12.11. 清除当前轨迹 Service_Clear_Current_Trajectory .....	88





5.12.12. 清除所有轨迹 Service_Clear_All_Trajectory .....	89
5.12.13. 关节空间运动 Service_Movej_P_Cmd .....	89
5.12.14. 样条曲线运动 Service_Moves_Cmd .....	90
5.13. 机械臂示教 .....	91
5.13.1. 关节示教 Service_Joint_Teach_Cmd .....	91
5.13.2. 位置示教 Service_Pos_Teach_Cmd .....	92
5.13.3. 姿态示教 Service_Ort_Teach_Cmd .....	93
5.13.4. 示教停止 Service_Teach_Stop_Cmd .....	94
5.13.5. 切换示教运动坐标系 Service_Set_Teach_Frame .....	94
5.13.6. 获取示教运动坐标系 Service_Get_Teach_Frame .....	95
5.14. 机械臂步进 .....	95
5.14.1. 关节步进 Service_Joint_Step_Cmd .....	95
5.14.2. 位置步进 Service_Pos_Step_Cmd .....	96
5.14.3. 姿态步进 Service_Ort_Step_Cmd .....	97
5.15. 控制器配置 .....	98
5.15.1. 获取控制器状态 Service_Get_Controller_State .....	98
5.15.2. 设置 WiFi AP 模式设置 Service_Set_WiFi_AP_Data .....	99
5.15.3. 设置 WiFi STA 模式设置 Service_Set_WiFi_STA_Data .....	99
5.15.4. 设置 UART_USB 接口波特率 Service_Set_USB_Data .....	100
5.15.5. 设置 RS485 配置 Service_Set_RS485 .....	100
5.15.6. 设置机械臂电源 Service_Set_Arm_Power .....	101
5.15.7. 获取机械臂电源 Service_Get_Arm_Power_State .....	102



5.15.8. 读取机械臂软件版本 Service_Get_Arm_Software_Version .....	102
5.15.9. 获取控制器的累计运行时间 Service_Get_System_Runtime .....	103
5.15.10. 清空控制器累计运行时间 Service_Clear_System_Runtime .....	103
5.15.11. 获取关节累计转动角度 Service_Get_Joint_Odom .....	104
5.15.12. 清除关节累计转动角度 Service_Clear_Joint_Odom .....	104
5.15.13. 配置高速网口 Service_Set_High_Speed_Eth .....	105
5.15.14. 设置高速网口网络配置 Service_Set_High_Ethernet--基础系列 .....	105
5.15.15. 获取高速网口网络配置 Service_Get_High_Ethernet--基础系列 .....	106
5.15.16. 保存参数 Service_Save_Device_Info_All--基础系列 .....	107
5.15.17. 配置有线网卡 IP 地址 Service_Set_NetIP--I 系列 .....	107
5.15.18. 查询有线网卡网络信息 Service_Get_Wired_Net--I 系列 .....	107
5.15.19. 查询无线网卡网络信息 Service_Get_Wifi_Net--I 系列 .....	108
5.15.20. 恢复网络出厂设置 Service_Set_Net_Default--I 系列 .....	109
5.15.21. 清除系统错误代码 Service_Clear_System_Err .....	110
5.15.22. 读取机械臂软件信息 Service_Get_Arm_Software_Info .....	110
5.15.23. 设置机械臂模式(仿真/真实)Service_Set_Arm_Run_Mode .....	111
5.15.24. 获取机械臂模式(仿真/真实)Service_Get_Arm_Run_Mode .....	111
5.16. IO 配置 .....	111
5.16.1. 设置数字 IO 模式 Service_Set_IO_Mode--I 系列 .....	112
5.16.2. 设置数字 IO 输出状态 Service_Set_DO_State .....	113
5.16.3. 查询指定 IO 状态 Service_Get_IO_State .....	113
5.16.4. 查询数字 IO 输出状态 Service_Get_DO_State--基础系列 .....	114



5.16.5. 查询数字 IO 输入状态 Service_Get_DI_State--基础系列 .....	115
5.16.6. 设置模拟 IO 输出状态 Service_Set_AO_State--基础系列 .....	115
5.16.7. 查询模拟 IO 输出状态 Service_Get_AO_State--基础系列 .....	116
5.16.8. 查询数字 IO 输入状态 Service_Get_AI_State--基础系列 .....	116
5.16.9. 查询所有 IO 输入状态 Service_Get_IO_Input .....	117
5.16.10. 查询所有 IO 的输出状态 Service_Get_IO_Output .....	117
5.16.11. 设置电源输出 Service_Set_Voltage--I 系列 .....	118
5.16.12. 获取电源输出 Service_Get_Voltage--I 系列 .....	118
5.17. 末端工具 IO 配置 .....	119
5.17.1. 设置工具端数字 IO 输出状态 Service_Set_Tool_DO_State .....	119
5.17.2. 设置工具端数字 IO 模式 Service_Set_Tool_IO_Mode .....	120
5.17.3. 查询工具端数字 IO 状态 Service_Get_Tool_IO_State .....	121
5.17.4. 设置工具端电源输出 Service_Set_Tool_Voltage .....	121
5.17.5. 获取工具端电源输出 Service_Get_Tool_Voltage .....	122
5.18. 末端手爪控制（选配） .....	122
5.18.1. 配置手爪的开口度 Service_Set_Gripper_Route .....	122
5.18.2. 设置夹爪松开到最大位置 Service_Set_Gripper_Release .....	123
5.18.3. 设置夹爪夹取 Service_Set_Gripper_Pick .....	124
5.18.4. 设置夹爪持续夹取 Service_Set_Gripper_Pick_On .....	124
5.18.5. 设置夹爪到指定开口位置 Service_Set_Gripper_Position .....	125
5.18.6. 获取夹爪状态 Service_Get_Gripper_State .....	126
5.19. 拖动示教及轨迹复现 .....	126



5.19.1. 进入拖动示教模式 Service_Start_Drag_Teach .....	127
5.19.2. 退出拖动示教模式 Service_Stop_Drag_Teach .....	127
5.19.3. 拖动示教轨迹复现 Service_Run_Drag_Trajectory .....	128
5.19.4. 拖动示教轨迹复现暂停 Service_Pause_Drag_Trajectory .....	128
5.19.5. 拖动示教轨迹复现继续 Service_Continue_Drag_Trajectory .....	129
5.19.6. 拖动示教轨迹复现停止 Service_Stop_Drag_Trajectory .....	129
5.19.7. 运动到轨迹起点 Service_Drag_Trajectory-Origin .....	130
5.19.8. 复合模式拖动示教 Service_Start_Multi_Drag_Teach .....	130
5.19.9. 保存拖动示教轨迹 Service_Save_Trajectory .....	131
5.19.10. 设置力位混合控制 Service_Set_Force_Postion .....	131
5.19.11. 结束力位混合控制 Service_Service_Stop_Force_Postion .....	132
5.20. 末端六维力传感器的使用（选配） .....	134
5.20.1. 获取六维力数据 Service_Get_Force_Data .....	135
5.20.2. 清空六维力数据 Service_Clear_Force_Data .....	135
5.20.3. 设置六维力重心参数 Service_Set_Force_Sensor .....	136
5.20.4. 手动标定六维力数据 Service_Manual_Set_Force .....	136
5.20.5. 退出标定流程 Service_Stop_Set_Force_Sensor .....	137
5.21. 末端五指灵巧手控制（选配） .....	139
5.21.1. 设置灵巧手手势序号 Service_Set_Hand_Posture .....	139
5.21.2. 设置灵巧手动作序列序号 Service_Set_Hand_Seq .....	140
5.21.3. 设置灵巧手角度 Service_Set_Hand_Angle .....	140
5.21.4. 设置灵巧手各关节速度 Service_Set_Hand_Speed .....	141



5.21.5. 设置灵巧手各关节力阈值 Service_Set_Hand_Force .....	141
5.22. 末端传感器—维力（选配） .....	142
5.22.1. 查询一维力数据 Service_Get_Fz .....	142
5.22.2. 清空一维力数据 Service_Clear_Fz .....	143
5.22.3. 自动标定末端一维力数据 Service_Auto_Set_Fz .....	143
5.22.4. 手动标定末端一维力数据 Service_Manual_Set_Fz .....	144
5.23. Modbus 配置 .....	145
5.23.1. 设置通讯端口 Modbus RTU 模式 Service_Set_Modbus_Mode .....	145
5.23.2. 关闭通讯端口 Modbus RTU 模式 Service_Close_Modbus_Mode .....	146
5.23.3. 配置连接 ModbusTCP 从站 Service_Set_Modbustcp_Mode--I 系列 .....	147
5.23.4. 配置关闭 ModbusTCP 从站 Service_Close_Modbustcp_Mode--I 系列 .....	147
5.23.5. 读线圈 Service_Get_Read_Coils .....	148
5.23.6. 读离散输入量 Service_Get_Read_Input_Status .....	149
5.23.7. 读保持寄存器 Service_Get_Read_Holding_Registers .....	150
5.23.8. 读输入寄存器 Service_Get_Read_Input_Registers .....	150
5.23.9. 写单圈数据 Service_Write_Single_Coil .....	151
5.23.10. 写单个寄存器 Service_Write_Single_Register .....	152
5.23.11. 写多个寄存器 Service_Write_Registers .....	153
5.23.12. 写多圈数据 Service_Write_Coils .....	154
5.23.13. 读多圈数据 Service_Get_Read_Multiple_Coils .....	155
5.23.14. 读多个保持寄存器 Service_Read_Multiple_Holding_Registers .....	156
5.23.15. 读多个输入寄存器 Service_Read_Multiple_Input_Registers .....	157



5.24. 升降机构 .....	158
5.24.1. 升降机构速度开环控制 Service_Set_Lift_Speed .....	158
5.24.2. 设置升降机构高度 Service_Set_Lift_Height .....	159
5.24.3. 获取升降机构状态 Service_Get_Lift_State .....	159
5.25. 透传力位混合控制补偿 .....	160
5.25.1. 开启透传力位混合控制补偿模式 Service_Start_Force_Position_Move .....	160
5.25.2. 力位混合控制补偿透传模式（关节角度） Service_Force_Position_Move_Joint .....	161
5.25.3. 力位混合控制补偿透传模式（位姿） Service_Force_Position_Move_Pose .....	162
5.25.4. 关闭透传力位混合控制补偿模式 Service_Stop_Force_Position_Move .....	163
5.26. 算法工具接口 .....	164
5.26.1. 初始化算法依赖数据 Service_Algo_Init_Sys_Data .....	164
5.26.2. 设置算法的安装角度 Service_Algo_Set_Angle .....	165
5.26.3. 获取算法的安装角度 Service_Algo_Get_Angle .....	165
5.26.4. 设置算法工作坐标系 Service_Algo_Set_WorkFrame .....	165
5.26.5. 获取当前工作坐标系 Service_Algo_Get_Curr_WorkFrame .....	166
5.26.6. 设置算法工具坐标系和负载 Service_Algo_Set_ToolFrame_Params .....	166
5.26.7. 获取算法当前工具坐标系 Service_Algo_Get_Curr_ToolFrame .....	166
5.26.8. 正解 Service_Algo_Forward_Kinematics .....	167
5.26.9. 逆解 Service_Algo_inverse_Kinematics .....	167
5.26.10. 计算平移、旋转运动位姿 Service_Algo_PoseMove .....	168
5.26.11. 计算环绕运动位姿 Service_Algo_RotateMove .....	168



5.26.12. 末端位姿转成工具位姿 Service_Algo_end2tool .....	169
5.26.13. 工具位姿转末端位姿 Service_Algo_tool2end .....	169
5.26.14. 四元数转欧拉角 Service_Algo_quaternion2euler .....	170
5.26.15. 欧拉角转四元数 Service_Algo_euler2quaternion .....	170
5.26.16. 欧拉角转旋转矩阵 Service_Algo_euler2matrix .....	170
5.26.17. 位姿转旋转矩阵 Service_Algo_pos2matrix .....	171
5.26.18. 旋转矩阵转位姿 Service_Algo_matrix2pos .....	171
5.26.19. 基坐标系转工作坐标系 Service_Algo_Base2WorkFrame .....	172
5.26.20. 工作坐标系转基坐标系 Service_Algo_WorkFrame_To_Base .....	172
5.26.21. 计算沿工具坐标系运动位姿 Service_Algo_Cartesian_Tool .....	173
5.26.22. 设置算法关节最大限位 Service_Set_Algo_Joint_Max_Limit .....	173
5.26.23. 获取算法关节最大限位 Service_Algo_Get_Joint_Max_Limit .....	173
5.26.24. 设置算法关节最小限位 Service_Algo_Set_Joint_Min_Limit .....	174
5.26.25. 获取算法关节最小限位 Service_Algo_Get_Joint_Min_Limit .....	174
5.26.26. 设置算法关节最大速度 Service_Algo_Set_Joint_Max_Speed .....	174
5.26.27. 获取算法关节最大速度 Service_Algo_Get_Joint_Max_Speed .....	174
5.26.28. 设置算法关节最大加速度 Service_Algo_Set_Joint_Max_Acc .....	175
5.26.29. 获取算法关节最大加速度 Service_Algo_Get_Joint_Max_Acc .....	175
5.27. 在线编程 .....	177
5.27.1. 文件下发 Service_Send_TrajectoryFile .....	177
5.27.2. 轨迹规划中改变速度比例系数 Service_Set_Plan_Speed .....	177
5.27.3. 文件树弹窗提醒 Service_Popup .....	178



5.28. 机械臂状态主动上报 .....	179
5.28.1. 设置主动上报配置 Service_Set_Realtime_Push .....	180
5.28.2. 获取主动上报配置 Service_Get_Realtime_Push .....	180
5.28.3. 机械臂状态主动上报 Service_Realtime_Arm_Joint_State .....	181
5.29. 通用扩展关节 .....	182
5.29.1. 关节速度环控制 Service_Expand_Set_Speed .....	182
5.29.2. 关节位置环控制 Service_Expand_Set_Pos .....	183
5.29.3. 扩展关节状态获取 Service_Expand_Get_State .....	184
5.30. 在线编程存储列表（I 系列） .....	185
5.30.1. 查询在线编程程序列表 Service_Get_Program_Trajectory_List .....	185
5.30.2. 查询当前在线编程文件的运行状态 Service_Get_Program_Run_State .....	186
5.30.3. 运行指定编号在线编程 Service_Set_Program_ID_Start .....	187
5.30.4. 删除指定 ID 的轨迹 Service_Delete_Program_Trajectory .....	188
5.30.5. 修改指定编号轨迹的信息 Service_Update_Program_Trajectory .....	188
5.30.6. 设置 IO 默认运行的在线编程文件编号 Service_Set_Default_Run_Program .....	189
5.30.7. 获取 IO 默认运行的在线编程文件编号 Service_Get_Default_Run_Program .....	189
5.31. 全局路点（I 系列） .....	191
5.31.1. 新增全局路点 Service_Add_Global_Waypoint .....	191
5.31.2. 更新全局路点 Service_Update_Global_Waypoint .....	193
5.31.3. 删除全局路点 Service_Delete_Global_Waypoint .....	194





5.31.4. 查询多个全局路点 Service_Get_Global_Point_List .....	194
5.31.5. 查询指定全局路点 Service_Given_Global_Waypoint .....	195
5.32. 电子围栏和虚拟墙（I 系列） .....	196
5.32.1. 新增几何模型参数 Service_Add_Electronic_Fence_Config .....	197
5.32.2. 更新几何模型参数 Service_Update_Electronic_Fence_Config .....	197
5.32.3. 删除几何模型参数 Service_Delete_Electronic_Fence_Config .....	197
5.32.4. 查询所有几何模型名称 Service_Get_Electronic_Fence_List_Names ..	198
5.32.5. 查询指定几何模型参数 Service_Given_Electronic_Fence_Config .....	198
5.32.6. 查询所有几何模型信息 Service_Get_Electronic_Fence_List_Info .....	199
5.32.7. 设置电子围栏使能状态 Service_Set_Electronic_Fence_Enable .....	200
5.32.8. 获取电子围栏使能状态 Service_Get_Electronic_Fence_Enable .....	200
5.32.9. 设置当前电子围栏参数 Service_Set_Electronic_Fence_Config .....	201
5.32.10. 获取当前电子围栏参数 Service_Get_Electronic_Fence_Config .....	201
5.32.11. 设置虚拟墙使能状态 Service_Set_Virtual_Wall_Enable .....	202
5.32.12. 获取虚拟墙使能状态 Service_Get_Virtual_Wall_Enable .....	203
5.32.13. 设置当前虚拟墙参数 Service_Set_Virtual_Wall_Config .....	203
5.32.14. 获取当前虚拟墙参数 Service_Get_Virtual_Wall_Config .....	204
5.33. 自碰撞安全检测（I 系列） .....	204
5.33.1. 设置自碰撞安全检测使能状态 Service_Set_Self_Collision_Enable .....	204
5.33.2. 获取自碰撞安全检测使能状态 Service_Get_Self_Collision_Enable .....	205



## 1.简介

本文档为 MATLAB 版本二次开发接口文档。

支持平台：Windows 64 位

Matlab 版本说明：此 Matlab 接口库使用 Matlab R2023a 生成，本文档中的示例皆在 Matlab R2023a 中测试运行。

## 2.功能介绍

MATLAB API 通过调用 C++ 版本的睿尔曼机械臂接口函数库实现，支持在 Windows 平台使用。

接口函数将用户指令封装成标准的 JSON 格式下发给机械臂，并解析机械臂回传的数据提供给用户。

接口函数基于 TCP/IP 协议编写，其中：



机械臂默认 IP 地址：192.168.1.18，端口号：8080

无论是 WIFI 模式还是以太网口模式，机械臂均以该 IP 和端口号对外进行 socket 通信，机械臂为 Server 模式，用户为 Client 模式。



### 3. 使用说明

- 接口函数包内包括以下文件：

 RM_Service.dll	2023/10/31 14:42	应用程序扩展	554 KB
 rm_serviceInterface.dll	2023/10/31 18:30	应用程序扩展	5,291 KB

rm\_serviceInterface.dll：Matlab API 接口实现。

RM\_Service.dll：Matlab API 库依赖的 C++库。

- 以下为 Matlab API 的使用步骤：

步骤一：将所依赖的 c++库 RM\_Service.dll 与 rm\_serviceInterface.dll 放在同一路径下，如下图。或者，将依赖库添加到系统路径。

名称	修改日期	类型	大小
 RM_Service.dll	2023/9/14 15:59	应用程序扩展	554 KB
 rm_serviceInterface.dll	2023/9/14 15:59	应用程序扩展	5,853 KB

步骤二：调用 addpath 包含接口文件的文件夹。

例如：接口文件位于“H:\Desktop\MATLAB API”，则调用以下命令包含接口文件。

```
addpath('H:\Desktop\MATLAB API')
```

步骤三：设置库的执行模式。

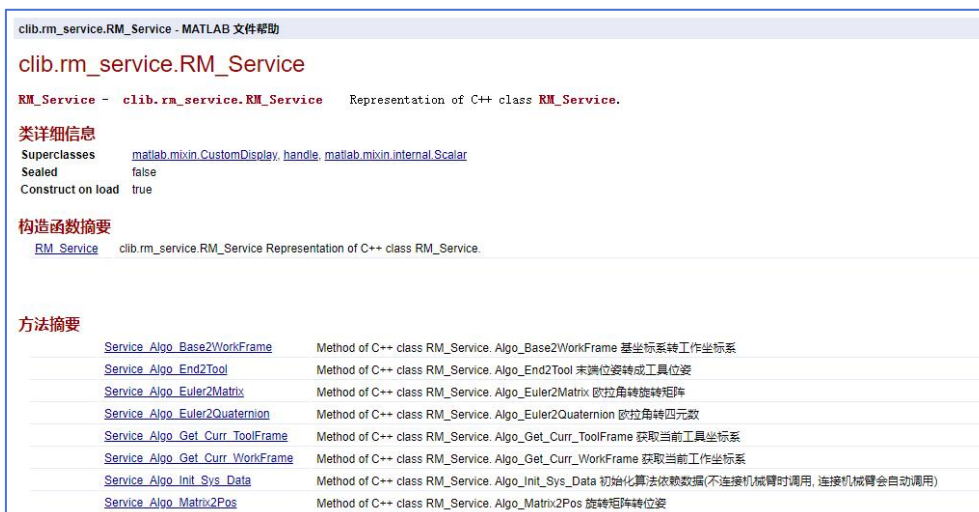
执行模式，指是在与 MATLAB 相同的进程中还是在外部进程中加载 C++ 接口，指定为 inprocess 或 outofprocess。

```
clibConfiguration('rm_service','ExecutionMode','outofprocess')
```

步骤四：使用 help 或者 doc 函数可获取库中函数的帮助提示。

例如：运行以下命令查看 RM\_Service 类中的方法。

```
doc clib.rm_service.RM_Service
```



运行以下命令查看 Service\_RM\_API\_Init 的帮助。

```
help clib.rm_service.RM_Service.Service_RM_API_Init
```

```
>> help clib.rm_service.RM_Service.Service_RM_API_Init
Service_RM_API_Init - Service_RM_API_Init Method of C++ class RM_Service.
    Service_RM_API_Init          API初始化

This content is from the external library documentation.

RetVal = clib.rm_service.RM_Service/Service_RM_API_Init(obj,devMode,pCallback)
Input Arguments
    obj          只读 clib.rm_service.RM_Service

    devMode      int32
    dev_mode     目标设备型号ARM_65/ARM_75/ARM_63_1/ARM_63_2/ARM_ECO65

    pCallback    clib.rm_service.RM_Callback
    pCallback    用于接收透传接口回调函数，不需要可以传入NULL
    RM_APISHARED_EXPORT

Output Arguments
    RetVal       int32

See also: clib.rm\_service.RM\_Callback
```

步骤五：使用 MATLAB clib 包加载库，并创建“RM\_Service”类的实例，控制机械臂。

例如创建一个名为“robot”的对象，该对象属于 RM\_Service 类

```
robot = clib.rm_service.RM_Service()
```

步骤六：连接机械臂，用户根据需要调用接口函数。

```
% API 初始化
```



```
RetVal = robot.Service_RM_API_Init(65, clib.type.nullptr);  
% 连接机械臂  
IP = '192.168.1.18'  
revtime = 200  
handle = robot.Service_Arm_Socket_Start(IP, 8080, revtime);  
  
% API 版本  
ver = robot.Service_API_Version()  
  
% 关节运动  
fjoint = [0,0,0,0,90,0,0]  
robot.Service_Movej_Cmd(handle, fjoint, 50, 0, true)
```

步骤七：API 反初始化，并断开与机械臂的连接。

```
% API 反初始化  
RetVal = robot.Service_RM_API_UnInit()  
% 断开连接  
robot.Service_Arm_Socket_Close(handle)
```

步骤八：卸载库。

```
% 卸载库  
unload(clibConfiguration('rm_service'));
```



## 4. 数据类型说明

### 4.1. 控制器错误类型

序号	错误代码（16 进制）	错误内容
1	0x0000	系统正常
2	0x1001	关节通信异常
3	0x1002	目标角度超过限位
4	0x1003	该处不可达，为奇异点
5	0x1004	实时内核通信错误
6	0x1005	关节通信总线错误
7	0x1006	规划层内核错误
8	0x1007	关节超速
9	0x1008	末端接口板无法连接
10	0x1009	超速度限制
11	0x100A	超加速度限制
12	0x100B	关节抱闸未打开
13	0x100C	拖动示教时超速
14	0x100D	机械臂发生碰撞
15	0x100E	无该工作坐标系
16	0x100F	无该工具坐标系
17	0x1010	关节发生掉使能错误
18	0x5001	预留



19	0x5002	预留
20	0x5003	控制器过温
21	0x5004	预留
22	0x5005	控制器过流
23	0x5006	控制器欠流
24	0x5007	控制器过压
25	0x5008	控制器欠压
26	0x5009	实时内核通讯错误

## 4.2. 关节错误类型

序号	错误代码（16 进制）	错误内容
1	0x0000	关节正常
2	0x0001	FOC 错误
3	0x0002	过压
4	0x0004	欠压
5	0x0008	过温
6	0x0010	启动失败
7	0x0020	编码器错误
8	0x0040	过流
9	0x0080	软件错误
10	0x0100	温度传感器错误
11	0x0200	位置超限错误



12	0x0400	关节 ID 非法
13	0x0800	位置跟踪错误
14	0x1000	电流检测错误
15	0x2000	抱闸打开失败
16	0x4000	位置指令阶跃警告
17	0x8000	多圈关节丢圈数
18	0xF000	通信丢帧

### 4.3. API 错误类型

Set 类的接口返回错误码 Retval, 返回 0 为成功, 其他值可查询下表。Get 类的接口的返回值形式为[RetVal, data], RetVal 为错误码, 返回 0 成功, 返回其他值可查询下表; data 为获取到的数据。

序号	错误代码 (16 进制)	错误内容
1	0x0000	系统运行正常
2	0x0001	消息请求返回 FALSE
3	0x0002	机械臂未初始化或输入型号非法
4	0x0003	非法超时时间
5	0x0004	Socket 初始化失败
6	0x0005	Socket 连接失败
7	0x0006	Socket 发送失败
8	0x0007	Socket 通讯超时
9	0x0008	未知错误





10	0x0009	数据不完整
11	0x000A	数组长度错误
12	0x000B	数据类型错误
13	0x000C	型号错误
14	0x000D	缺少回调函数
15	0x000E	机械臂异常停止
16	0x000F	轨迹文件名称过长
17	0x0010	轨迹文件校验失败
18	0x0011	轨迹文件读取失败
19	0x0012	控制器忙，请稍后再试
20	0x0013	非法输入
21	0x0014	数据队列已满
22	0x0015	计算失败
23	0x0016	文件打开失败
24	0x0017	力控标定手动停止
25	0x0018	没有可保存轨迹
26	0x0019	UDP 监听接口运行报错

## 4.4. 结构化数据类定义

结构化数据的类定义列举了一些常用结构体，其他结构体和数据类型可查看

rm\_define.h 和 robot\_define.h，或者通过 doc 命令查看文档帮助。

### 4.4.1. 位姿结构体 Pose--clib.rm\_service.Pose



Pose - 属性:

position: [1 × 1 clib.rm\_service.Pos]

quaternion: [1 × 1 clib.rm\_service.Quat]

euler: [1 × 1 clib.rm\_service.Euler]

### 结构体成员

position

位置坐标，单位：m

quaternion

四元数

euler

姿态角，单位：rad

#### 4.4.2. 坐标系结构体 FRAME--clib.rm\_service.FRAME

FRAME - 属性:

frame\_name: [1 × 1 clib.rm\_service.FRAME\_NAME]

pose: [1 × 1 clib.rm\_service.Pose]

payload: 0

x: 0

y: 0

z: 0

### 结构体成员

frame\_name

坐标系名称



Pose

坐标系位姿

payload

坐标系末端负载重量 单位 g

x,y,z

坐标系末端负载位置，单位：m

#### 4.4.3. 关节状态结构体 JOINT\_STATE--clib.rm\_service.JOINT\_STATE

JOINT\_STATE - 属性:

temperature: [1 × 1 clib.array.rm\_service.Float]

voltage: [1 × 1 clib.array.rm\_service.Float]

current: [1 × 1 clib.array.rm\_service.Float]

en\_state: [1 × 1 clib.array.rm\_service.UnsignedChar]

err\_flag: [1 × 1 clib.array.rm\_service.UnsignedShort]

sys\_err: 0

#### 结构体成员

joint

关节角度，每个关节角度，float 类型，单位：°

temperature

关节温度，float 类型，单位：摄氏度

voltage

关节电压，float 类型，单位：V

current



关节电流，float 类型，单位：mA

en\_state

使能状态

err\_flag

关节错误代码，unsigned int 类型

sys\_err

机械臂系统错误代码，unsigned int 类型

#### 4.4.4. 实时推送机械臂状态结构体 RobotStatus--clib.rm\_service.RobotStatus

RobotStatus - 属性:

errCode: 0

arm\_ip: [0×0 string]

arm\_err: 0

joint\_status: [1×1 clib.rm\_service.JointStatus]

force\_sensor: [1×1 clib.rm\_service.ForceData]

sys\_err: 0

waypoint: [1×1 clib.rm\_service.Pose]

#### 结构体成员

errCode

API 解析错误码

arm\_ip

推送消息的机械臂 IP

arm\_err



机械臂错误码

joint\_status

当前关节状态

force\_sensor

力数据

sys\_err

机械臂系统错误代码

waypoint

路点信息



## 5. 接口库函数说明

### 5.1. 连接相关函数

该部分函数用来控制 socket 连接的打开与关闭。

#### 5.1.1. API 初始化 Service\_RM\_API\_Init

Service_RM_API_Init(devMode,pCallback)	
描述	该函数用于初始化 API。
输入	<p>(1) devMode                      int32</p> <p>目标设备型号 RM65、RML63 、ECO65、RM75、GEN72 分别对应整数 65、631、632、651、75、72。若传入型号非法， 则默认机械臂为六轴。</p> <p>(2) pCallback                      clib.rm_service.RM_Callback</p> <p>用于基础系列机械臂接收透传接口回调函数，I 系列或者不 需要则传入 clib.type.nullptr</p>

#### 5.1.2. API 反初始化 Service\_RM\_API\_UnInit

Service_RM_API_UnInit()	
描述	该函数用于 API 反初始化。

#### 5.1.3. 查询 API 版本信息 Service\_API\_Version

RetVal = Service_API_Version(obj)	
描述	该函数用于查询 API 当前版本。
返回值	<p>(1) RetVal                      string</p> <p>API 版本号</p>



#### 5.1.4. 连接机械臂 Service\_Arm\_Socket\_Start

```
ArmSocket = Service_Arm_Socket_Start(arm_ip, arm_port, recv_timeout)
```

描述	该函数用于连接机械臂。
参数	<p>(1) arm_ip            string</p> <p>用于传入要连接机械臂的 IP 地址，机械臂 IP 地址默认为 192.168.1.18。</p> <p>(2) arm_port        int32</p> <p>用于传入要连接机械臂的端口，机械臂端口默认为 8080。</p> <p>(3) recv_timeout   int32</p> <p>Socket 连接超时时间，单位：ms</p>
返回值	<p>(1) ArmSocket       uint64</p> <p>成功返回：Socket 句柄；失败返回：API 错误码，查询 API 错误类型。</p>

#### 5.1.5. 查询连接状态 Service\_Arm\_Socket\_State

```
Arm_Socket_State(ArmSocket)
```

描述	用于查询连接状态。
参数	<p>(1) ArmSocket       uint64</p> <p>Socket 句柄。</p>
返回值	<p>(1) RetVal           int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.1.6. 关闭连接 Service\_Arm\_Socket\_Close

```
Service_Arm_Socket_Close(ArmSocket)
```



描述	该函数用于关闭与机械臂的 socket 连接。
参数	(1) ArmSocket      uint64 Socket 句柄。

代码示例：

```
% 创建 RM_Service 对象

robot = clib.rm_service.RM_Service();

% API 版本

ver = robot.Service_API_Version()

% 连接机械臂

IP = '192.168.1.18';

revtime = 200;

RetVal = robot.Service_RM_API_Init(65, clib.type.nullptr);

handle = robot.Service_Arm_Socket_Start(IP, 8080, revtime);

% 连接状态

RetVal = robot.Service_Arm_Socket_State(handle);

% 断开连接

robot.Service_RM_API_Uninit()

robot.Service_Arm_Socket_Close(handle)
```

## 5.2. 关节配置函数

睿尔曼机械臂在出厂前所有参数都已经配置到最佳状态，一般不建议用户修改关节的底层参数。若用户确需修改，首先应使机械臂处于非使能状态，然后再





发送修改参数指令，参数设置成功后，发送关节恢复使能指令。需要注意的是，关节恢复使能时，用户需要保证关节处于静止状态，以免上使能过程中关节发生报错。关节正常上使能后，用户方可控制关节运动。

### 5.2.1. 设置关节最大速度 Service\_Set\_Joint\_Speed

RetVal = Service_Set_Joint_Speed(ArmSocket, joint_num, speed, block)	
描述	该函数用于设置关节最大速度，单位：°/s。建议使用默认最大速度，如需更改，设置的关节最大加速度与最大速度的比值需要 $\geq 1.5$ ，否则可能出现运动异常。
参数	<div>(1) ArmSocket          uint64 Socket 句柄。</div> <div>(2) joint_num          uint8 关节序号，1~7</div> <div>(3) speed              single 关节转速，单位：°/s</div> <div>(4) block              logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</div>
返回值	<div>(1) RetVal            int32</div> <div>成功返回：0；失败返回：错误码，查询 API 错误类型。</div>

### 5.2.2. 设置关节最大加速度 Service\_Set\_Joint\_Acc

RetVal = Service_Set_Joint_Acc(ArmSocket, joint_num, acc, block)	
描述	该函数用于设置关节最大加速度，单位：°/s <sup>2</sup> 。建议使用默认关节最



	大加速度，如需更改，设置的关节最大加速度与最大速度的比值需要 $\geq 1.5$ ，否则可能出现运动异常。
参数	<p>(1) ArmSocket      uint64 Socket 句柄。</p> <p>(2) joint_num      uint8 关节序号，1~7</p> <p>(3) acc              single 关节转速，单位：<math>^{\circ}/s^2</math></p> <p>(4) block           logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal           int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

### 5.2.3. 设置关节最小限位 Service\_Set\_Joint\_Min\_Pos

RetVal = Service\_Set\_Joint\_Min\_Pos(ArmSocket, joint\_num, joint, block)

描述	该函数用于设置关节所能到达的最小限位，单位： $^{\circ}$ 。
参数	<p>(1) ArmSocket      uint64 socket 句柄</p> <p>(2) joint_num      uint8 关节序号，1~7</p> <p>(3) joint            single 关节最小位置，单位：<math>^{\circ}</math></p>



	<p>(4) block                      logical</p> <p>false-非阻塞，发送后立即返回；true-阻塞，等待控制器返回设置成功指令</p>
返回值	<p>(1) RetVal                      int32</p> <p>0-成功，失败返回:错误码,查询 API 错误类型.</p>

#### 5.2.4. 设置关节最大限位 Service\_Set\_Joint\_Max\_Pos

RetVal = Service_Set_Joint_Max_Pos(ArmSocket, joint_num, joint, block)	
描述	该函数用于设置关节所能达到的最大限位，单位：°
参数	<p>(1) ArmSocket                      uint64</p> <p>socket 句柄</p> <p>(2) joint_num                      uint8</p> <p>关节序号，1~7</p> <p>(3) joint                      single</p> <p>关节最大位置，单位：°</p> <p>(4) block                      logical</p> <p>false-非阻塞，发送后立即返回；true-阻塞，等待控制器返回设置成功指令</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.2.5. 设置关节最大速度（驱动器） Service\_Set\_Joint\_Drive\_Speed

RetVal = Service_Set_Joint_Drive_Speed(ArmSocket, joint_num, speed)	
描述	该函数用于设置关节（驱动器）最大速度，单位：°/s。建议使用默



	认最大速度，如需更改，设置的关节最大加速度与最大速度的比值需要 $\geq 1.5$ ，否则可能出现运动异常。
参数	<div>(1) ArmSocket      uint64 Socket 句柄。</div> <div>(2) joint_num      uint8 关节序号，1~7</div> <div>(3) speed          single 关节转速，单位：<math>^\circ/\text{s}</math></div>
返回值	<div>(1) RetVal          int32</div> <div>成功返回：0；失败返回：错误码，查询 API 错误类型。</div>

#### 5.2.6. 设置关节最大加速度（驱动器）Service\_Set\_Joint\_Drive\_Acc

RetVal = Service_Set_Joint_Drive_Acc(ArmSocket, joint_num, acc)	
描述	该函数用于设置关节（驱动器）最大加速度，单位： $^\circ/\text{s}^2$ 。建议使用默认关节最大加速度，如需更改，设置的关节最大加速度与最大速度的比值需要 $\geq 1.5$ ，否则可能出现运动异常。
参数	<div>(1) ArmSocket      uint64 Socket 句柄。</div> <div>(2) joint_num      uint8 关节序号，1~7</div> <div>(3) acc            single 关节转速，单位：<math>^\circ/\text{s}^2</math></div>
返回值	<div>(1) RetVal          int32</div>



成功返回：0；失败返回：错误码，查询 API 错误类型。

#### 5.2.7. 设置关节最小限位（驱动器）Service\_Set\_Joint\_Drive\_Min\_Pos

RetVal = Service\_Set\_Joint\_Drive\_Min\_Pos(ArmSocket, joint\_num, joint)

描述	该函数用于设置关节所能到达的最小限位，单位：°。
参数	<p>(1) ArmSocket          uint64 socket 句柄</p> <p>(2) joint_num          uint8 关节序号，1~7</p> <p>(3) joint                  single 关节最小位置，单位：°</p>
返回值	<p>(1) RetVal              int32 0-成功，失败返回:错误码,查询 API 错误类型.</p>

#### 5.2.8. 设置关节最大限位（驱动器）Service\_Set\_Joint\_Drive\_Max\_Pos

RetVal = Service\_Set\_Joint\_Drive\_Max\_Pos(ArmSocket, joint\_num, joint)

描述	该函数用于设置关节（驱动器）所能达到的最大限位，单位：°
参数	<p>(1) ArmSocket          uint64 socket 句柄</p> <p>(2) joint_num          uint8 关节序号，1~7</p> <p>(3) joint                  single 关节最大位置，单位：°</p>
返回值	<p>(1) RetVal              int32</p>



成功返回：0；失败返回：错误码，查询 API 错误类型。

#### 5.2.9. 设置关节使能 Service\_Set\_Joint\_EN\_State

```
RetVal = Service_Set_Joint_EN_State(ArmSocket, joint_num, state, block)
```

描述	该函数用于设置关节（驱动器）使能状态。
参数	<p>(1) ArmSocket          uint64 socket 句柄</p> <p>(2) joint_num          uint8 关节序号，1~7</p> <p>(3) state              logical true-上使能，false-掉使能</p> <p>(4) block              logical false-非阻塞，发送后立即返回；true-阻塞，等待控制器返回设置成功指令</p>
返回值	<p>(1) RetVal            int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.2.10. 设置关节零位 Service\_Set\_Joint\_Zero\_Pos

```
RetVal = Service_Set_Joint_Zero_Pos(ArmSocket, joint_num, block)
```

描述	该函数用于将当前位置设置为关节零位。
参数	<p>(1) ArmSocket          uint64 socket 句柄</p> <p>(2) joint_num          uint8 关节序号，1~7</p>



	<p>(3) block                      logical</p> <p>false-非阻塞，发送后立即返回；true-阻塞，等待控制器返回设置成功指令</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.2.11. 清除关节错误代码 Service\_Set\_Joint\_Err\_Clear

RetVal = Service_Set_Joint_Err_Clear(ArmSocket, joint_num, block)	
描述	该函数用于清除关节错误代码。
参数	<p>(1) ArmSocket                      uint64</p> <p>socket 句柄</p> <p>(2) joint_num                      uint8</p> <p>关节序号，1~7</p> <p>(3) block                      logical</p> <p>false-非阻塞，发送后立即返回；true-阻塞，等待控制器返回设置成功指令</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.2.12. 自动设置限位 Service\_Auto\_Set\_Joint\_Limit

RetVal = Service_Auto_Set_Joint_Limit(ArmSocket, joint_num, block)	
描述	该函数用于清除关节错误代码。
参数	<p>(1) ArmSocket                      uint64</p> <p>socket 句柄</p>



	<p>(2) joint_num          uint8</p> <p>关节序号, 1~7</p> <p>(3) block              logical</p> <p>false-非阻塞, 发送后立即返回; true-阻塞, 等待控制器返回设置成功指令</p>
返回值	<p>(1) RetVal            int32</p> <p>成功返回: 0; 失败返回: 错误码, 查询 API 错误类型。</p>

### 5.3. 关节参数查询函数

#### 5.3.1. 查询关节最大速度 Service\_Get\_Joint\_Speed

[RetVal, speed] = Service_Get_Joint_Speed(ArmSocket)	
描述	该函数用于查询关节最大速度。
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p>
返回值	<p>(1) RetVal            int32</p> <p>0-成功, 失败返回:错误码,查询 API 错误类型.</p> <p>(2) speed              7 element vector single</p> <p>speed 关节 1~7 转速数组, 单位: °/s</p>

#### 5.3.2. 查询关节最大加速度 Service\_Get\_Joint\_Acc

[RetVal,acc] = Service_Get_Joint_Acc(ArmSocket)	
描述	该函数用于查询关节最大加速度。
参数	<p>(1) ArmSocket          uint64</p>





	Socket 句柄
返回值	<p>(1) RetVal            int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) acc                7 element vector single</p> <p>存放关节 1~7 加速度数组，单位：<math>^{\circ}/s^2</math></p>

#### 5.3.3. 获取关节最小限位 Service\_Get\_Joint\_Min\_Pos

[RetVal,min_joint] = Service_Get_Joint_Min_Pos(ArmSocket)	
描述	该函数用于获取关节最小限位。
参数	<p>(1) ArmSocket        uint64</p> <p>Socket 句柄</p>
返回值	<p>(1) RetVal            int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) min_joint        7 element vector single</p> <p>存放关节 1~7 最小限位数组，单位：<math>^{\circ}</math></p>

#### 5.3.4. 获取关节最大限位 Service\_Get\_Joint\_Max\_Pos

[RetVal,max_joint] = Service_Get_Joint_Max_Pos(ArmSocket)	
描述	该函数用于获取关节最大限位。
参数	<p>(1) ArmSocket        uint64</p> <p>Socket 句柄</p>
返回值	<p>(1) RetVal            int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) max_joint        7 element vector single</p>



存放关节 1~7 最大限位数组，单位：°

### 5.3.5. 查询关节最大速度（驱动器）Service\_Get\_Joint\_Drive\_Speed

[RetVal, speed] = Service_Get_Joint_Drive_Speed(ArmSocket)	
描述	该函数用于查询关节（驱动器）最大速度。
参数	(1) ArmSocket          uint64 Socket 句柄
返回值	(1) RetVal              int32 0-成功，失败返回:错误码,查询 API 错误类型. (2) speed                7 element vector single speed 关节 1~7 转速数组，单位：°/s

### 5.3.6. 查询关节最大加速度（驱动器）Service\_Get\_Joint\_Acc

[RetVal,acc] = Service_Get_Joint_Drive_Acc(ArmSocket)	
描述	该函数用于查询关节（驱动器）最大加速度。
参数	(2) ArmSocket          uint64 Socket 句柄
返回值	(3) RetVal              int32 成功返回：0；失败返回：错误码，查询 API 错误类型。 (4) acc                 7 element vector single 存放关节 1~7 加速度数组，单位：°/s <sup>2</sup>

### 5.3.7. 获取关节最小限位（驱动器）Service\_Get\_Joint\_Drive\_Min\_Pos

[RetVal,min_joint] = Service_Get_Joint_Drive_Min_Pos(ArmSocket)	
描述	该函数用于获取关节（驱动器）最小限位。



参数	(1) ArmSocket          uint64  Socket 句柄
返回值	(1) RetVal                  int32  成功返回：0；失败返回：错误码，查询 API 错误类型。  (2) min_joint              7 element vector single  存放关节 1~7 最小限位数组，单位：°

#### 5.3.8. 获取关节最大限位（驱动器）Service\_Get\_Joint\_Drive\_Max\_Pos

[RetVal,max_joint] = Service_Get_Joint_Drive_Max_Pos(ArmSocket)	
描述	该函数用于获取关节（驱动器）最大限位。
参数	(1) ArmSocket          uint64  Socket 句柄
返回值	(1) RetVal                  int32  成功返回：0；失败返回：错误码，查询 API 错误类型。  (2) max_joint              7 element vector single  存放关节 1~7 最大限位数组，单位：°

#### 5.3.9. 获取关节使能状态 Service\_Get\_Joint\_EN\_State

[RetVal,state] = Service_Get_Joint_EN_State(ArmSocket)	
描述	该函数用于获取关节使能状态。
参数	(1) ArmSocket          uint64  Socket 句柄
返回值	(1) RetVal                  int32  成功返回：0；失败返回：错误码，查询 API 错误类型。



	(2) state                      7 element vector uint8  关节 1~7 使能状态数组, 1-使能状态, 0-掉使能状态
--	---

#### 5.3.10. 获取关节错误代码 Service\_Get\_Joint\_Err\_Flag

[RetVal,state,bstate] = Service_Get_Joint_Err_Flag(ArmSocket)	
描述	该函数用于获取关节错误代码。
参数	(1) ArmSocket                  uint64  Socket 句柄
返回值	(1) RetVal                      int32  0-成功, 失败返回:错误码,查询 API 错误类型.  (2) state                        7 element vector uint16  存放关节错误码 (请参考 api 文档中的关节错误码)  (3) bstate                       7 element vector uint16  关节抱闸状态(1 代表抱闸未打开, 0 代表抱闸已打开)

#### 5.3.11. 查询关节软件版本号 Service\_Get\_Joint\_Software\_Version

[RetVal,version] = Service_Get_Joint_Software_Version(ArmSocket)	
描述	该函数用于查询关节软件版本号。
参数	(1) ArmSocket                  uint64  Socket 句柄
返回值	(1) RetVal                      int32  0-成功, 失败返回:错误码,查询 API 错误类型.  (2) version                      7 element vector single  获取到的关节软件版本号



## 代码示例：

```
% 创建 RM_Service 对象

robot = clib.rm_service.RM_Service();

% API 版本

ver = robot.Service_API_Version()

% 连接机械臂

IP = '192.168.1.18';

revtime = 200;

RetVal = robot.Service_RM_API_Init(65, clib.type.nullptr);

handle = robot.Service_Arm_Socket_Start(IP, 8080, revtime);

% 连接状态

RetVal = robot.Service_Arm_Socket_State(handle);

% 设置 1 关节速度为 180 ° /s

RetVal = robot.Service_Set_Joint_Speed(handle,1,180,true);

% 设置 1 关节上使能

[~] = robot.Service_Set_Joint_EN_State(handle, 1, true, true);

pause(1)

% 获取关节速度

[~, speed] = robot.Service_Get_Joint_Speed(handle)

% 获取关节错误代码

[~, sta, bsta] = robot.Service_Get_Joint_Err_Flag(handle)

% 获取关节软件版本
```



```
[~, jver] = robot.Service_Get_Joint_Software_Version(handle);  
  
dec2hex(jver)  
  
% 断开连接  
  
robot.Service_RM_API_Uninit()  
  
robot.Service_Arm_Socket_Close(handle)
```

## 5.4. 机械臂末端运动参数配置

### 5.4.1. 设置末端最大线速度 Service\_Set\_Arm\_Line\_Speed

RetVal = Service_Set_Arm_Line_Speed(ArmSocket,speed,block)	
描述	该函数用于设置机械臂末端最大线速度。建议使用默认最大线速度，如需更改,设置的机械臂末端最大线加速度与最大线速度的比值需要 $\geq 3$ ，否则可能出现运动异常。
参数	<div>(1) ArmSocket          uint64 socket 句柄</div> <div>(2) speed                single 末端最大线速度，单位 m/s</div> <div>(3) block                logical false-非阻塞，发送后立即返回；true-阻塞，等待控制器返回设置成功指令</div>
返回值	<div>(1) RetVal                int32 0-成功，失败返回:错误码,查询 API 错误类型.</div>



#### 5.4.2. 设置末端最大线加速度 Service\_Set\_Arm\_Line\_Acc

RetVal = Service_Set_Arm_Line_Acc(ArmSocket,acc,block)	
描述	该函数用于设置机械臂末端最大线加速度。建议使用默认最大线加速度，如需更改，设置的机械臂末端最大线加速度与最大线速度的比值需要 $\geq 3$ ，否则可能出现运动异常。
参数	<p>(1) ArmSocket          uint64 socket 句柄</p> <p>(2) acc                  single 末端最大线加速度，单位 <math>m/s^2</math></p> <p>(3) block                logical false-非阻塞，发送后立即返回；true-阻塞，等待控制器返回设置成功指令</p>
返回值	<p>(1) RetVal              int32 0-成功，失败返回:错误码,查询 API 错误类型.</p>

#### 5.4.3. 设置末端最大角速度 Service\_Set\_Arm\_Angular\_Speed

RetVal = Service_Set_Arm_Angular_Speed(ArmSocket,speed,block)	
描述	该函数用于设置机械臂末端最大角速度。建议使用默认最大角速度，如需更改，设置的机械臂末端最大角加速度与最大角速度的比值需要 $\geq 3$ ，否则可能出现运动异常。



参数	<p>(1) ArmSocket      uint64</p> <p>socket 句柄</p> <p>(2) speed            single</p> <p>末端最大角速度, 单位 <math>m/s^2</math></p> <p>(3) block            logical</p> <p>false-非阻塞, 发送后立即返回; true-阻塞, 等待控制器返回设置成功指令</p>
返回值	<p>(1) RetVal           int32</p> <p>0-成功, 失败返回:错误码,查询 API 错误类型.</p>

#### 5.4.4. 设置末端最大角加速度 Service\_Set\_Arm\_Angular\_Acc

RetVal = Service\_Set\_Arm\_Angular\_Acc(ArmSocket,acc,block)

描述	该函数用于设置机械臂末端最大角加速度。建议使用默认最大角加速度, 如需更改, 设置的机械臂末端最大角加速度与最大角速度的比值需要 $\geq 3$ , 否则可能出现运动异常。
参数	<p>(1) ArmSocket      uint64</p> <p>socket 句柄</p> <p>(2) acc              single</p> <p>末端最大角加速度, 单位 <math>m/s^2</math></p> <p>(3) block            logical</p> <p>false-非阻塞, 发送后立即返回; true-阻塞, 等待控制器返回设置成功指令</p>
返回值	<p>(1) RetVal           int32</p>





0-成功, 失败返回:错误码,查询 API 错误类型.

#### 5.4.5. 获取末端最大线速度 Service\_Get\_Arm\_Line\_Speed

[RetVal,speed] = Service_Get_Arm_Line_Speed(ArmSocket)	
描述	该函数用于获取机械臂末端最大线速度。
参数	(1) ArmSocket          uint64 socket 句柄
返回值	(1) RetVal              int32 0-成功, 失败返回:错误码,查询 API 错误类型. (2) speed                single speed 末端最大线速度, 单位 m/s

#### 5.4.6. 获取最大末端线加速度 Service\_Get\_Arm\_Line\_Acc

[RetVal,acc] = Service_Get_Arm_Line_Acc(ArmSocket)	
描述	该函数用于获取机械臂末端最大线加速度。
参数	(1) ArmSocket          uint64 socket 句柄
返回值	(1) RetVal              int32 0-成功, 失败返回:错误码,查询 API 错误类型. (2) acc                  single acc 末端最大线加速度, 单位 m/s^2

#### 5.4.7. 获取末端最大角速度 Service\_Get\_Arm\_Angular\_Speed

[RetVal,speed] = Service_Get_Arm_Angular_Speed(ArmSocket)	
描述	该函数用于获取机械臂末端最大角速度。



参数	(1) ArmSocket          uint64  socket 句柄
返回值	(1) RetVal              int32  0-成功, 失败返回:错误码,查询 API 错误类型.  (2) speed              single  末端最大角加速度, 单位 rad/s

#### 5.4.8. 获取末端最大角加速度 Service\_Get\_Arm\_Angular\_Acc

```
[RetVal, acc] = Service_Get_Arm_Angular_Acc(ArmSocket)
```

描述	该函数用于获取机械臂末端最大角加速度。
参数	(1) ArmSocket          uint64  socket 句柄
返回值	(1) RetVal              int32  0-成功, 失败返回:错误码,查询 API 错误类型.  (2) acc                  single  acc 末端最大角加速度, 单位 rad/s^2

#### 5.4.9. 设置机械臂末端参数为初始值 Service\_Set\_Arm\_Tip\_Init

```
RetVal = Service_Set_Arm_Tip_Init(ArmSocket,block)
```

描述	该函数用于设置机械臂末端参数为初始值。
参数	(1) ArmSocket          uint64  socket 句柄  (2) block              logical  false-非阻塞, 发送后立即返回; true-阻塞, 等待控制器返



	回设置成功指令
返回值	(1) RetVal                  int32  0-成功, 失败返回:错误码,查询 API 错误类型.

#### 5.4.10. 设置碰撞等级 Service\_Set\_Collision\_Stage

RetVal = Service_Set_Collision_Stage(ArmSocket,stage,block)	
描述	该函数用于设置机械臂动力学碰撞等级, 等级 0~8, 等级越高, 碰撞检测越灵敏, 同时也更容易发生误碰撞检测。机械臂上电后默认碰撞等级为 0, 即不检测碰撞。
参数	(1) ArmSocket                  uint64  socket 句柄  (2) stage                          int32  等级: 0~8, 0-无碰撞, 8-碰撞最灵敏  (3) block                          logical  false-非阻塞, 发送后立即返回; true-阻塞, 等待控制器返回设置成功指令
返回值	(1) RetVal                  int32  成功返回: 0; 失败返回: 错误码, 查询 API 错误类型。

#### 5.4.11. 查询碰撞等级 Service\_Get\_Collision\_Stage

[RetVal,stage] = Service_Get_Collision_Stage(ArmSocket)	
描述	该函数用于查询机械臂动力学碰撞等级, 等级 0~8, 数值越高, 碰撞检测越灵敏, 同时也更易发生误碰撞检测。机械臂上电后默认碰撞等级为 0, 即不检测碰撞。



参数	(1) ArmSocket          uint64  Socket 句柄
返回值	(1) RetVal              int32  成功返回：0；失败返回：错误码，查询 API 错误类型。  (2) stage                int32  碰撞等级，等级：0~8

#### 5.4.12. 设置关节零位补偿角度 Service\_Set\_Joint\_Zero\_Offset

RetVal = Service\_Set\_Joint\_Zero\_Offset(ArmSocket,offset,block)

描述	该函数用于设置机械臂各关节零位补偿角度，一般在对机械臂零位进行标定后调用该函数。
参数	(1) ArmSocket          uint64  Socket 句柄  (2) offset                7 element vector single  关节 1~7 零位补偿角度数组，角度单位：度  (3) block                logical  false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。
返回值	(1) RetVal              int32  成功返回：0；失败返回：错误码，查询 API 错误类型。
备注	该指令用户不可自行使用，必须配合测量设备进行绝对精度补偿时方可使用，否则会导致机械臂参数错误！

代码示例：



```
% 创建 RM_Service 对象

robot = clib.rm_service.RM_Service();

% API 版本

ver = robot.Service_API_Version()

% 连接机械臂

IP = '192.168.1.18';

revtime = 200;

RetVal = robot.Service_RM_API_Init(65, clib.type.nullptr);

handle = robot.Service_Arm_Socket_Start(IP, 8080, revtime);

% 连接状态

RetVal = robot.Service_Arm_Socket_State(handle);

% 设置 1 关节速度为 180 ° /s

RetVal = robot.Service_Set_Joint_Speed(handle,1,180,true);

% 设置 1 关节上使能

[~] = robot.Service_Set_Joint_EN_State(handle, 1, true, true);

pause(1)

% 获取关节速度

[~, speed] = robot.Service_Get_Joint_Speed(handle)

% 获取关节错误代码

[~, sta, bsta] = robot.Service_Get_Joint_Err_Flag(handle)

% 获取关节软件版本

[~, jver] = robot.Service_Get_Joint_Software_Version(handle);
```



```
dec2hex(jver)

% 断开连接

robot.Service_RM_API_UnInit()

robot.Service_Arm_Socket_Close(handle)
```

## 5.5. 机械臂末端接口板

### 5.5.1. 查询末端接口板软件版本号 Service\_Get\_Tool\_Software\_Version

[RetVal, version] = Service_Get_Tool_Software_Version(ArmSocket)	
描述	该函数用于查询末端接口板软件版本号
参数	(1) ArmSocket      uint64 Socket 句柄
返回值	(1) RetVal          int32 成功返回：0；失败返回：错误码，查询 API 错误类型。 (2) version          int32 末端接口板软件版本号

## 5.6. 工具坐标系设置

### 5.6.1. 标定点位 Service\_Auto\_Set\_Tool\_Frame

RetVal = Service_Auto_Set_Tool_Frame(ArmSocket, point_num, block)	
描述	该函数用于六点法自动设置工具坐标系（标记点位），机械臂控制器最多只能存储 10 个工具坐标系信息，在建立新的工具坐标系之前，请确认工具坐标系数量没有超过限制，否则建立新工具坐标



	系无法成功。
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) point_num          uint8</p> <p>1~6 代表 6 个标定点。</p> <p>(3) block                  logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                  int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>
备注	机械臂控制器最多只能存储 10 个工具信息，在建立新的工具之前，请确认工具数量没有超过限制，否则建立新工具无法成功。

#### 5.6.2. 生成工具坐标系 Service\_Generate\_Auto\_Tool\_Frame

RetVal = Service_Generate_Auto_Tool_Frame(ArmSocket, name, payload, x, y, z, block)	
描述	该函数用于六点法自动设置工具坐标系(生成坐标系)，机械臂控制器最多只能存储 10 个工具坐标系信息，在建立新的工具坐标系之前，请确认工具坐标系数量没有超过限制，否则建立新工具坐标系无法成功。
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) name                  string</p>



	<p>工具坐标系名称，不能超过十个字节。</p> <p>(3) payload                  single</p> <p>新工具坐标系执行末端负载重量    单位 kg</p> <p>(4) x,y,z                      single</p> <p>新工具坐标系执行末端负载位置    位置 x, y, z    单位 mm</p> <p>(5) block                      logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                  int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>
备注	控制器只能存储十个工具坐标系，超过十个控制器不予响应，请在标定前查询已有工具坐标系。

### 5.6.3. 手动设置工具坐标系 Service\_Manual\_Set\_Tool\_Frame

RetVal = Service_Manual_Set_Tool_Frame(ArmSocket, name, pose, payload, x, y, z, block)	
描述	该函数用于手动设置工具坐标系。
参数	<p>(1) ArmSocket                  uint64</p> <p>Socket 句柄</p> <p>(2) name                      string</p> <p>工具坐标系名称，不能超过十个字节。</p> <p>(3) pose                      clib.rm_service.Pose</p> <p>新工具坐标系执行末端相对于机械臂法兰中心的位姿</p>





	<p>(4) payload                  single</p> <p>新工具坐标系执行末端负载重量    单位 kg</p> <p>(5) x,y,z                      single</p> <p>新工具坐标系执行末端负载位置    位置 x, y, z    单位 mm</p> <p>(6) block                      logical</p> <p>false-非阻塞, 发送后立即返回;    true-阻塞, 等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                  int32</p> <p>成功返回: 0; 失败返回: 错误码, 查询 API 错误类型。</p>
备注	控制器只能存储十个工具坐标系, 超过十个控制器不予响应, 请在标定前查询已有工具。

#### 5.6.4. 切换当前工具坐标系 Service\_Change\_Tool\_Frame

RetVal = Service\_Change\_Tool\_Frame(ArmSocket, name, block)

描述	该函数用于切换当前工具坐标系
参数	<p>(1) ArmSocket                  uint64</p> <p>Socket 句柄</p> <p>(2) name                      string</p> <p>目标工具坐标系名称</p> <p>(3) block                      logical</p> <p>false-非阻塞, 发送后立即返回;    true-阻塞, 等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                  int32</p>



成功返回：0；失败返回：错误码，查询 API 错误类型。

#### 5.6.5. 删除指定工具坐标系 Service\_Delete\_Tool\_Frame

```
RetVal = Service_Delete_Tool_Frame(ArmSocket, name, block)
```

描述	该函数用于删除指定工具坐标系
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) name                string</p> <p>要删除的工具坐标系名称</p> <p>(3) block               logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal              int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>
备注	删除坐标系后，机械臂将切换到机械臂法兰末端工具坐标系。

#### 5.6.6. 修改指定工具坐标系 Service\_Update\_Tool\_Frame

```
RetVal = Service_Update_Tool_Frame(ArmSocket, name, pose, payload, x, y, z)
```

描述	该函数用于修改指定工具坐标系。
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) name                string</p> <p>要修改的工具坐标系名称，不能超过十个字节。</p>



	<p>(3) pose                      clib.rm_service.Pose</p> <p>更新执行末端相对于机械臂法兰中心的位姿</p> <p>(4) payload                  single</p> <p>更新工具坐标系执行末端负载重量    单位 kg</p> <p>(5) x,y,z                      single</p> <p>更新工具坐标系执行末端负载位置    位置 x, y, z    单位 mm</p>
返回值	<p>(1) RetVal                  int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.6.7. 设置工具坐标系的包络参数 Service\_Set\_Tool\_Envelope

```
RetVal = Service_Set_Tool_Envelope( ArmSocket, toolName, count, envelopeNames, envelope)
```

描述	该函数用于设置工具坐标系的包络参数
参数	<p>(1) ArmSocket              uint64</p> <p>Socket 句柄</p> <p>(2) toolName              只读 string</p> <p>指定工具坐标系名称</p> <p>(3) count                  int32</p> <p>设置第几个包络球，count&lt;=当前存在的包络参数数量+1，即可修改当前存在的包络参数，或者新增一个包络参数，每个工具最多支持 5 个包络球，该参数设置 0 即清空所有包络</p> <p>(4) envelopeNames    只读 string</p> <p>包络球名称</p>



	<p>(5) envelope          clib.rm_service.ToolEnvelope</p> <p>包络参数</p>
返回值	<p>(1) RetVal          int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>
示例	<pre>% 设置 “test_1” 工具坐标系第一个包络参数为 “envelope1”  envelope1 = clib.rm_service.ToolEnvelope  envelope1.radius=0.1;  envelope1.x=0.1;  envelope1.y=0.1;  envelope1.z=0.1;  robot.Service_Set_Tool_Envelope(handle, "test_1", 1,"envelope1", envelope1)</pre>

#### 5.6.8. 获取工具坐标系的包络参数 Service\_Get\_Tool\_Envelope

RetVal = Service_Get_Tool_Envelope(ArmSocket,name,list)	
描述	该函数用于获取工具坐标系的包络参数
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) name          string</p> <p>控制器中已存在的工具坐标系名称</p> <p>(3) list          clib.rm_service.ToolEnvelopeList</p> <p>包络参数列表，每个工具最多支持 5 个包络球，可以没有包络</p>



返回值	(1) RetVal            int32  成功返回：0；失败返回：错误码，查询 API 错误类型。
	(2) list                clib.rm_service.ToolEnvelopeList  包络参数列表，每个工具最多支持 5 个包络球，可以没有包络

## 5.7. 工具坐标系查询

### 5.7.1. 获取当前工具坐标系 Service\_Get\_Current\_Tool\_Frame

[RetVal, tool] = Service_Get_Current_Tool_Frame(ArmSocket,tool)	
描述	该函数用于获取当前工具坐标系
参数	(1) ArmSocket          uint64  Socket 句柄
	(2) tool                clib.rm_service.FRAME  返回的工具坐标系参数
返回值	(1) RetVal            int32  成功返回：0；失败返回：错误码，查询 API 错误类型。
	(2) tool                clib.rm_service.FRAME  返回的工具坐标系参数

### 5.7.2. 获取指定工具坐标系 Service\_Get\_Given\_Tool\_Frame

[RetVal, tool] = Service_Get_Given_Tool_Frame(ArmSocket, name,tool)	
描述	该函数用于获取指定工具坐标系
参数	(1) ArmSocket          uint64



	<p>Socket 句柄</p> <p>(2) name                      string</p> <p>指定的工具坐标系名称</p> <p>(3) Tool                      clib.rm_service.FRAME</p> <p>返回的工具坐标系参数</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) Tool                      clib.rm_service.FRAME</p> <p>返回的工具坐标系参数</p>

### 5.7.3. 获取所有工具坐标系名称 Service\_Get\_All\_Tool\_Frame

[RetVal, names, len]= Service_Get_All_Tool_Frame(ArmSocket,names)	
描述	该函数用于获取所有工具坐标系名称
参数	<p>(1) ArmSocket                  uint64</p> <p>Socket 句柄</p> <p>(2) names                      clib.rm_service.FRAMENAME</p> <p>返回的工具坐标系名称数组</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) names                      clib.rm_service.FRAMENAME</p> <p>返回的工具坐标系名称数组</p> <p>(3) len                          int32</p> <p>返回工具坐标系数量。</p>



## 代码示例：

```
% 创建 RM_Service 对象

robot = clib.rm_service.RM_Service();

% API 版本

ver = robot.Service_API_Version()

% 连接机械臂

IP = '192.168.1.18';

revtime = 200;

RetVal = robot.Service_RM_API_Init(65, clib.type.nullptr);

handle = robot.Service_Arm_Socket_Start(IP, 8080, revtime);

% 连接状态

RetVal = robot.Service_Arm_Socket_State(handle);

% 手动设置工具坐标系

pose = clib.rm_service.Pose;

[~] = robot.Service_Manual_Set_Tool_Frame(handle, "mat", pose, 1, 2, 3, 4, true );

% 切换当前工具坐标系

[~] = robot.Service_Change_Tool_Frame(handle, "mat", true);

% 获取当前工具坐标系

tool = clib.rm_service.FRAME;

[~, frame1] = robot.Service_Get_Current_Tool_Frame(handle, tool);

char(frame1.frame_name.name.int32)

% 获取指定工具坐标系
```



```
tool = clib.rm_service.FRAME;

[~, mat] = robot.Service_Get_Given_Tool_Frame(handle, "mat", tool);

name=char(mat.frame_name.name.int32)

% 删除坐标系

[~] = robot.Service_Delete_Tool_Frame(handle, "mat", true);

% 获取全部工具坐标系

names = clib.rm_service.FRAMENAME();

[ret, names, len] = robot.Service_Get_All_Tool_Frame(handle, names);

for i = 1:len

char(names.name(i).name.int32)

end

% 断开连接

robot.Service_RM_API_UnInit()

robot.Service_Arm_Socket_Close(handle)
```

## 5.8. 工作坐标系设置

### 5.8.1. 自动设置工作坐标系 Service\_Auto\_Set\_Work\_Frame

```
RetVal = Service_Auto_Set_Work_Frame(ArmSocket, name, point_num, block)
```

描述	该函数用于三点法自动设置工作坐标系。	
参数	(1) ArmSocket	uint64
	Socket 句柄	





	<p>(2) name            string</p> <p>工作坐标系名称，不能超过十个字节。</p> <p>(3) point_num       uint8</p> <p>1~3 代表 3 个标定点，依次为原点、X 轴上任意点、Y 轴上任意点，4 代表生成坐标系。</p> <p>(4) block            logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal           int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>
备注	机械臂控制器最多只能存储 10 个工作坐标系信息，在建立新的工作坐标系之前，请确认工作坐标系数量没有超过限制，否则建立新工作坐标系无法成功。

#### 5.8.2. 手动设置工作坐标系 Service\_Manual\_Set\_Work\_Frame

RetVal = Service_Manual_Set_Work_Frame(ArmSocket, name, pose, block)	
描述	该函数用于手动设置工作坐标系。
参数	<p>(1) ArmSocket       uint64</p> <p>Socket 句柄</p> <p>(2) name            string</p> <p>工作坐标系名称，不能超过十个字节。</p> <p>(3) pose             clib.rm_service.Pose</p> <p>新工作坐标系相对于基坐标系的位姿。</p>



	<p>(4) block                      logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>
备注	控制器只能存储十个工作坐标系，超过十个控制器不予响应，请在标定前查询已有工作坐标系。

#### 5.8.3. 切换当前工作坐标系 Service\_Change\_Work\_Frame

RetVal = Service\_Change\_Work\_Frame(ArmSocket, name, block)

描述	该函数用于切换当前工作坐标系
参数	<p>(1) ArmSocket                      uint64</p> <p>Socket 句柄</p> <p>(2) name                      single</p> <p>目标工作坐标系名称</p> <p>(3) block                      logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.8.4. 删除指定工作坐标系 Service\_Delete\_Work\_Frame

RetVal = Service\_Delete\_Work\_Frame(ArmSocket, name, block)

描述	该函数用于删除指定工作坐标系。
----	-----------------



参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) Name                string</p> <p>要删除的工作坐标系名称</p> <p>(3) block                logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal              int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>
备注	删除坐标系后，机械臂将切换到机械臂基坐标系

#### 5.8.5. 修改指定工作坐标系 Service\_Update\_Work\_Frame

RetVal = Service\_Update\_Work\_Frame(ArmSocket, name, pose)

描述	该函数用于修改指定工作坐标系。
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) name                string</p> <p>工作坐标系名称，不能超过十个字节。</p> <p>(3) pose                clib.rm_service.Pose</p> <p>新工作坐标系相对于基坐标系的位姿。</p>
返回值	<p>(2) RetVal              int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>



## 5.9. 工作坐标系查询

### 5.9.1. 获取当前工作坐标系 Service\_Get\_Current\_Work\_Frame

[RetVal, frame] = Service_Get_Current_Work_Frame(ArmSocket, frame)	
描述	该函数用于获取当前工作坐标系。
参数	<div>(1) ArmSocket          uint64 Socket 句柄</div> <div>(2) frame              clib.rm_service.FRAME 返回的工作坐标系位姿参数</div>
返回值	<div>(1) RetVal            int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</div> <div>(2) frame            clib.rm_service.FRAME 返回的工作坐标系位姿参数</div>

### 5.9.2. 获取指定工作坐标系 Service\_Get\_Given\_Work\_Frame

[RetVal, pose] = Service_Get_Given_Work_Frame(ArmSocket, name, pose)	
描述	该函数用于获取指定工作坐标系
参数	<div>(1) ArmSocket          uint64 Socket 句柄</div> <div>(2) name              string 指定的工作坐标系名称</div> <div>(3) pose              clib.rm_service.Pose 返回的工作坐标系位姿参数</div>



返回值	(1) RetVal                      int32  成功返回：0；失败返回：错误码，查询 API 错误类型。
	(2) pose                          clib.rm_service.Pose  返回的工作坐标系位姿参数

### 5.9.3. 获取所有工作坐标系名称 Service\_Get\_All\_Work\_Frame

```
[RetVal, names, len] = Service_Get_All_Work_Frame(ArmSocket,names)
```

描述	该函数用于获取所有工作坐标系名称。
参数	(1) ArmSocket                      uint64  Socket 句柄
	(2) names                          clib.rm_service.FRAMENAME  返回的工作坐标系的名称数组
返回值	(1) RetVal                          int32  成功返回：0；失败返回：错误码，查询 API 错误类型。
	(2) names                          clib.rm_service.FRAMENAME  返回的工作坐标系的名称数组
返回值	(3) len                              int32  返回的工作坐标系的数量。

## 5.10. 机械臂状态查询

### 5.10.1. 获取机械臂当前状态 Service\_Get\_Current\_Arm\_State

```
[RetVal, joint, pose, Arm_Err, Sys_Err] =
```



Service\_Get\_Current\_Arm\_State(ArmSocket, pose)

描述	该函数用于获取机械臂当前状态
参数	<div>(1) ArmSocket          uint64 Socket 句柄</div> <div>(2) pose                  clib.rm_service.Pose 机械臂当前位姿</div>
返回值	<div>(1) RetVal              int32 成功返回：0。失败返回:错误码,查询 API 错误类型。</div> <div>(2) joint                7 element vector single 关节 1~7 角度数组</div> <div>(3) pose                  clib.rm_service.Pose 机械臂当前位姿</div> <div>(4) Arm_Err            uint16 机械臂运行错误代码</div> <div>(5) Sys_Err            uint16 控制器错误代码</div>

#### 5.10.2. 获取关节温度 Service\_Get\_Joint\_Temperature

[RetVal, temperature] = Service\_Get\_Joint\_Temperature(ArmSocket)

描述	该函数用于获取关节当前温度。
参数	<div>(1) ArmSocket          uint64 Socket 句柄</div>
返回值	<div>(1) RetVal              int32</div>



	成功返回：0；失败返回：错误码，查询 API 错误类型。  (2) temperature      7 element vector single  关节 1~7 温度数组
--	---

#### 5.10.3. 获取关节电流 Service\_Get\_Joint\_Current

[RetVal, current] = Service_Get_Joint_Current(ArmSocket)	
描述	该函数用于获取关节当前电流。
参数	(1) ArmSocket      uint64  Socket 句柄
返回值	(1) RetVal      int32  成功返回：0；失败返回：错误码，查询 API 错误类型。  (2) current      7 element vector single  关节 1~7 电流数组

#### 5.10.4. 获取关节电压 Service\_Get\_Joint\_Voltage

[RetVal, voltage] = Service_Get_Joint_Voltage(ArmSocket)	
描述	该函数用于获取关节当前电压。
参数	(1) ArmSocket      uint64  Socket 句柄
返回值	(1) RetVal      int32  成功返回：0；失败返回：错误码，查询 API 错误类型。  (2) voltage      7 element vector single  关节 1~7 电压数组



#### 5.10.5. 获取关节当前角度 Service\_Get\_Joint\_Degree

[RetVal, joint] = Service_Get_Joint_Degree (ArmSocket)	
描述	该函数用于获取机械臂各关节的当前角度
参数	(1) ArmSocket          uint64 Socket 句柄
返回值	(1) RetVal              int32 成功返回：0；失败返回：错误码，查询 API 错误类型。  (2) Joint                7 element vector single 关节 1~7 当前角度数组；

#### 5.10.6. 获取所有状态 Service\_Get\_Arm\_All\_State

[RetVal, joint_state] = Service_Get_Arm_All_State(ArmSocket, joint_state)	
描述	该函数用于获取机械臂所有状态
参数	(1) ArmSocket          uint64 Socket 句柄  (2) joint_state          clib.rm_service.JOINT_STATE 机械臂所有状态；
返回值	(1) RetVal              int32 成功返回：0；失败返回：错误码，查询 API 错误类型。  (2) joint_state          clib.rm_service.JOINT_STATE 机械臂所有状态

#### 5.10.7. 获取轨迹规划计数 Service\_Get\_Arm\_Plan\_Num

RetVal = Service_Get_Arm_Plan_Num(ArmSocket, plan)	
--	--





描述	该函数用于获取机械臂轨迹规划计数
参数	(1) ArmSocket          uint64 Socket 句柄
返回值	(1) RetVal              int32 成功返回：0；失败返回：错误码，查询 API 错误类型。  (2) plan                int32 查询到的轨迹规划计数

代码示例：

```
% 创建 RM_Service 对象

robot = clib.rm_service.RM_Service();

% API 版本

ver = robot.Service_API_Version()

% 连接机械臂

IP = '192.168.1.18';

revtime = 200;

RetVal = robot.Service_RM_API_Init(65, clib.type.nullptr);

handle = robot.Service_Arm_Socket_Start(IP, 8080, revtime);

% 连接状态

RetVal = robot.Service_Arm_Socket_State(handle);

% 获取机械臂状态

pose = clib.rm_service.Pose;

[ret, joint, pose, arm_err, sys_err] = robot.Service_Get_Current_Arm_State(handle,
```



```
pose)

% 获取所有状态

joint_state = clib.rm_service.JOINT_STATE;

[~, jointstate] = robot.Service_Get_Arm_All_State(handle, joint_state);

jointstate.temperature.int32    % 温度

jointstate.en_state.int32      % 使能状态

jointstate.err_flag.int32      % 关节错误代码

% 断开连接

robot.Service_RM_API_UnInit()

robot.Service_Arm_Socket_Close(handle)
```

## 5.11. 机械臂初始位姿

### 5.11.1. 设置初始位姿角度 Service\_Set\_Arm\_Init\_Pose

RetVal = Service\_Set\_Arm\_Init\_Pose(ArmSocket, target, block)

描述	该函数用于设置机械臂的初始位姿角度。
参数	<div><div>(1) ArmSocket      uint64</div><div>Socket 句柄</div><div>(2) Target          7 element vector single</div><div>机械臂初始位置关节角度数组</div><div>(3) block            logical</div><div>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</div></div>



返回值	(1) RetVal            int32  成功返回：0；失败返回：错误码，查询 API 错误类型。
-----	---

#### 5.11.2. 获取初始位姿角度 Service\_Get\_Arm\_Init\_Pose

RetVal = Service_Get_Arm_Init_Pose(ArmSocket, joint)	
描述	该函数用于获取机械臂初始位姿角度。
参数	(1) ArmSocket            uint64  Socket 句柄  (2) Joint                      7 element vector single  机械臂初始位姿关节角度数组
返回值	(1) RetVal            int32  成功返回：0；失败返回：错误码，查询 API 错误类型。

#### 5.11.3. 设置安装角度 Service\_Set\_Install\_Pose

RetVal = Service_Set_Install_Pose(ArmSocket, x, y, z, block)	
描述	该函数用于设置机械臂安装方式。
参数	(1) ArmSocket            uint64  Socket 句柄  (2) x                      single  旋转角，单位 °  (3) y                      single  俯仰角，单位 °  (4) z                      single  方位角，单位 °



	<p>(5) block                      logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.11.4. 查询安装角度 Service\_Get\_Install\_Pose

RetVal = Service_Get_Install_Pose(ArmSocket, fx, fy, fz)	
描述	该函数用于查询机械臂安装角度。
参数	<p>(1) ArmSocket                      uint64</p> <p>Socket 句柄</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) fx                      single</p> <p>旋转角，单位 °</p> <p>(3) fy                      single</p> <p>俯仰角，单位 °</p> <p>(4) fz                      single</p> <p>方位角，单位 °</p>

代码示例：

```
% 创建 RM_Service 对象

robot = clib.rm_service.RM_Service();

% API 版本
```



```
ver = robot.Service_API_Version()

% 连接机械臂

IP = '192.168.1.18';

revtime = 200;

RetVal = robot.Service_RM_API_Init(65, clib.type.nullptr);

handle = robot.Service_Arm_Socket_Start(IP, 8080, revtime);

% 连接状态

RetVal = robot.Service_Arm_Socket_State(handle);

% 设置初始位姿

initpose = [0,0,0,90,0,0]

[~] = robot.Service_Set_Arm_Init_Pose(handle, initpose, true)

% 获取初始位姿

[~, fpose] = robot.Service_Get_Arm_Init_Pose(handle)

% 设置安装角度

[~] = robot.Service_Set_Install_Pose(handle, 90, 90, 90, true)

% 查询安装角度

[~,x,y,z] = robot.Service_Get_Install_Pose(handle)

% 断开连接

robot.Service_RM_API_UnInit()

robot.Service_Arm_Socket_Close(handle)
```

## 5.12. 机械臂运动规划



### 5.12.1. 关节空间运动 Service\_Movej\_Cmd

RetVal = Service\_Movej\_Cmd(ArmSocket, joint, v, r, trajectory\_connect, block)

描述	该函数用于关节空间运动。
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) joint                7 element vector single 目标关节 1~7 角度数组</p> <p>(3) v                    uint8 速度百分比系数，1~100。</p> <p>(4) r                    single 交融半径百分比系数，0~100。</p> <p>(5) trajectory_connect    int32 代表是否和下一条运动一起规划，0 代表立即规划，1 代表和下一条轨迹一起规划，当为 1 时，轨迹不会立即执行</p> <p>(6) block                logical false-非阻塞，发送后立即返回； true-阻塞，等待机械臂到达位置或者规划失败。</p>
返回值	<p>(1) RetVal                int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p>
注意	trajectory_connect 参数为 1 交融半径才生效，如果为 0 则交融半径不生效



### 5.12.2. 笛卡尔空间直线运动 Service\_MoveCmd

```
RetVal = Service_MoveCmd(ArmSocket, pose, v, r, trajectory_connect, block)
```

描述	该函数用于笛卡尔空间直线运动。
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) pose                clib.rm_service.Pose 目标位姿，位置单位：米，姿态单位：弧度</p> <p>(3) v                    uint8 速度百分比系数，1~100</p> <p>(4) r                    single 交融半径百分比系数，0~100。</p> <p>(5) trajectory_connect   int32 代表是否和下一条运动一起规划，0 代表立即规划，1 代表和下一条轨迹一起规划，当为 1 时，轨迹不会立即执行</p> <p>(6) block                logical false-非阻塞，发送后立即返回； true-阻塞，等待机械臂到达位置或者规划失败。</p>
返回值	<p>(1) RetVal                int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p>
注意	trajectory_connect 参数为 1 交融半径才生效，如果为 0 则交融半径不生效



### 5.12.3. 笛卡尔空间圆弧运动 Service\_Movec\_Cmd

```
RetVal = Service_Movec_Cmd(ArmSocket, pose_via, pose_to, v, r, loop, trajectory_connect, block)
```

描述	该函数用于笛卡尔空间圆弧运动
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) pose_vai          clib.rm_service.Pose 中间点位姿，位置单位：米，姿态单位：弧度</p> <p>(3) pose_to          clib.rm_service.Pose 终点位姿，位置单位：米，姿态单位：弧度</p> <p>(4) v                  uint8 速度比例 1~100，即规划速度和加速度占机械臂末端最大角速度和角加速度的百分比</p> <p>(5) r                  single 交融半径百分比系数，0~100。</p> <p>(6) loop              uint8 规划圈数，目前默认 0。</p> <p>(7) trajectory_connect    int32 代表是否和下一条运动一起规划，0 代表立即规划，1 代表和下一条轨迹一起规划，当为 1 时，轨迹不会立即执行</p> <p>(8) block              logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器返</p>





	回设置成功指令。
返回值	(1) RetVal                      int32 成功返回：0；失败返回：错误码，查询 API 错误类型。
注意	trajectory_connect 参数为 1 交融半径才生效，如果为 0 则交融半径不生效

代码示例：

```
% 创建 RM_Service 对象

robot = clib.rm_service.RM_Service();

% API 版本

ver = robot.Service_API_Version()

% 连接机械臂

IP = '192.168.1.18';

revtime = 200;

RetVal = robot.Service_RM_API_Init(65, clib.type.nullptr);

handle = robot.Service_Arm_Socket_Start(IP, 8080, revtime);

% 连接状态

RetVal = robot.Service_Arm_Socket_State(handle);

% 关节空间运动

fjoint = [20,20,70,30,90,120,0];

Ret = robot.Service_Movej_Cmd(handle ,fjoint,20,0,0,true);

pause(1);

% 获取当前 pose
```



```
mpose = clib.rm_service.Pose;

[ret, joint, mpose, arm_err, sys_err] =

robot.Service_Get_Current_Arm_State(handle, mpose)

% 笛卡尔空间直线运动

mpose.position.x = mpose.position.x+0.10;

ret = robot.Service_MoveL_Cmd(handle,mpose,20,0,0,true);

% 笛卡尔空间圆弧运动

Ret = robot.Service_Movej_Cmd(handle ,fjoint,20,0,0,true);

[RetVal1,joint,mpose,Arm_Err,Sys_Err] =

robot.Service_Get_Current_Arm_State(handle,pose);

pause(1);

[RetVal2,joint1,npose,Arm_Err1,Sys_Err1] =

robot.Service_Get_Current_Arm_State(handle,pose);

mpose.position.x = mpose.position.x+0.05;

npose.position.y = mpose.position.y+0.05;

Ret = robot.Service_Movec_Cmd(ArmSocket,mpose,npose,20,0,0,0,true);

% 断开连接

robot.Service_RM_API_UnInit()

robot.Service_Arm_Socket_Close(handle)
```

#### 5.12.4. 关节角度 CANFD 透传 Service\_Movej\_CANFD

```
RetVal = Service_Movej_CANFD(ArmSocket, joint, follow, expand)
```



描述	该函数用于角度不经规划，直接通过 CANFD 透传给机械臂，使用透传接口时，请勿使用其他运动接口。
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) joint                7 element vector single</p> <p>关节 1~7 目标角度数组</p> <p>(3) follow              logical</p> <p>是否高跟随,true--高跟随，false--低跟随</p> <p>(4) expand              single</p> <p>扩展关节目标位置，单位°。如果存在通用扩展轴，并需要进行透传，可使用该参数进行透传发送。不需要时传入 0 即可</p>
返回值	<p>(1) RetVal              int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>
备注	<p>透传周期越快，控制效果越好，越平顺。基础系列 WIFI 和网口模式透传周期最快 20ms，USB 和 RS48 模式透传周期最快 10ms。高速网口的透传周期最快也可到 10ms，不过在使用该高速网口前，需要使用指令打开配置。另外 I 系列有线网口周期最快可达 5ms。</p> <p>用户使用该函数时请做好轨迹规划，轨迹规划的平滑成都决定了机械臂的运行状态，帧与帧之间关节的角度差不能超过 10°，并保证关节规划的速度不超过 180°/s，否则关节不会响应。</p> <p>由于该模式直接下发给机械臂，不经控制器规划，因此只要控制器运行正常并且目标角度在可达范围内，机械臂立即返回成功指令，</p>



此时机械臂可能仍在运行；若有错误，立即返回失败指令。

#### 5.12.5. 位姿 CANFD 透传 Service\_Movep\_CANFD

RetVal = Service\_Movep\_CANFD(ArmSocket, pose, follow)

**描述** 该函数用于位姿不经规划，直接通过 CANFD 透传给机械臂，使用透传接口是，请勿使用其他运动接口。

**参数**

(1) ArmSocket          uint64  
Socket 句柄

(2) pose                  clib.rm\_service.Pose  
位姿 (优先采用四元数表达)

(3) follow                logical  
true-高跟随，false-低跟随。若使用高跟随，透传周期要求不超过 10ms

**返回值**

(1) RetVal                int32  
成功返回：0；失败返回：错误码，查询 API 错误类型。

#### 5.12.6. 计算环绕运动位姿 Service\_MoveRotate\_Cmd

RetVal = Service\_MoveRotate\_Cmd(ArmSocket, rotateAxis, rotateAngle, choose\_axis, v, r, trajectory\_connect, block)

**描述** 该函数用于计算环绕运动位姿并按照结果运动。

**参数**

(1) ArmSocket          uint64  
Socket 句柄

(2) rotateAxis          int32  
旋转轴： 1：x 轴， 2：y 轴， 3：z 轴



	<p>(3) rotateAngle      single</p> <p>旋转角度： 旋转角度， 单位(度)</p> <p>(4) choose_axis      clib.rn_service.Pose</p> <p>指定计算时使用的坐标系</p> <p>(5) v                  uint8</p> <p>速度</p> <p>(6) r                  single</p> <p>轨迹交融半径，暂不支持交融，目前默认 0</p> <p>(7) trajectory_connect      int32</p> <p>代表是否和下一条运动一起规划，0 代表立即规划，1 代表和下一条轨迹一起规划，当为 1 时，轨迹不会立即执行</p> <p>(8) block              logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal              int32</p> <p>成功返回： 0；失败返回： 错误码， 查询 API 错误类型。</p>

#### 5.12.7. 沿工具端位姿移动 Service\_MoveCartesianTool\_Cmd

RetVal = Service\_MoveCartesianTool\_Cmd(ArmSocket, movelengthx, movelengthy, movelengthz, m\_dev, v, r, trajectory\_connect, block)

描述	该函数用于沿工具端位姿运动。
参数	<p>(1) ArmSocket              uint64</p> <p>Socket 句柄</p>



	<p>(2) Joint_Cur            7 element vector single</p> <p>当前关节角度</p> <p>(3) movelengthx        single</p> <p>沿 X 轴移动长度，米为单位</p> <p>(4) Movelengthy        single</p> <p>沿 Y 轴移动长度，米为单位</p> <p>(5) movelengthz        single</p> <p>沿 Z 轴移动长度，米为单位</p> <p>(6) m_dev                int32</p> <p>机械臂型号</p> <p>(7) v                    uint8</p> <p>速度</p> <p>(8) r                    single</p> <p>轨迹交融半径，暂不支持交融，目前默认 0</p> <p>(9) trajectory_connect    int32</p> <p>代表是否和下一条运动一起规划，0 代表立即规划，1 代表和下一条轨迹一起规划，当为 1 时，轨迹不会立即执行</p> <p>(10) block                logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                int32</p>



成功返回：0；失败返回：错误码，查询 API 错误类型。

#### 5.12.8. 快速急停 Service\_Move\_Stop\_Cmd

RetVal = Service\_Move\_Stop\_Cmd(ArmSocket, block)

描述	该函数用于突发状况，机械臂以最快速度急停，轨迹不可恢复。
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) block                logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.12.9. 暂停当前规划 Service\_Move\_Pause\_Cmd

RetVal = Service\_Move\_Pause\_Cmd(ArmSocket, block)

描述	该函数用于轨迹暂停，暂停在规划轨迹上，轨迹可恢复。
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) block                logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>



#### 5.12.10. 继续当前轨迹 Service\_Move\_Continue\_Cmd

RetVal = Service_Move_Continue_Cmd(ArmSocket, block)	
描述	该函数用于轨迹暂停后，继续当前轨迹运动
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) block                logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.12.11. 清除当前轨迹 Service\_Clear\_Current\_Trajectory

RetVal = Service_Clear_Current_Trajectory(ArmSocket, block)	
描述	该函数用于清除当前轨迹，必须在暂停后使用，否则机械臂会发生意外！！！！
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) block                logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p>





#### 5.12.12. 清除所有轨迹 Service\_Clear\_All\_Trajectory

RetVal = Service_Clear_All_Trajectory(ArmSocket, block)	
描述	该函数用于清除所有轨迹，必须在暂停后使用，否则机械臂会发生意外！
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) block                logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.12.13. 关节空间运动 Service\_Movej\_P\_Cmd

RetVal = Service_Movej_P_Cmd(ArmSocket, pose, v, r, trajectory_connect, block)	
描述	该函数用于关节空间运动到目标位姿
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) pose                clib.rm_service.Pose 目标位姿，位置单位：米，姿态单位：弧度。</p> <p>注意：该目标位姿必须是机械臂当前工具坐标系相对于当前工作坐标系的位，用户在使用该指令前务必确保，否则目标位姿会出错！！！！</p>



	<p>(3) v                      uint8</p> <p>速度百分比系数，1~100</p> <p>(4) r                      single</p> <p>轨迹交融半径，目前默认 0。</p> <p>(5) trajectory_connect      int32</p> <p>代表是否和下一条运动一起规划，0 代表立即规划，1 代表和下一条轨迹一起规划，当为 1 时，轨迹不会立即执行</p> <p>(6) block                  logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                  int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>
注意	该运动暂不支持轨迹交融。

#### 5.12.14. 样条曲线运动 Service\_Moves\_Cmd

RetVal = Service_Moves_Cmd(ArmSocket, pose, v, r, trajectory_connect, block)	
描述	该函数用于样条曲线运动到目标位姿
参数	<p>(1) ArmSocket              uint64</p> <p>Socket 句柄</p> <p>(2) pose                      clib.rm_service.Pose</p> <p>目标位姿，位置单位：米，姿态单位：弧度。</p> <p>(3) v                          uint8</p>



	<p>速度百分比系数，1~100</p> <p>(4) r                      single</p> <p>轨迹交融半径，目前默认 0。</p> <p>(5) trajectory_connect      int32</p> <p>代表是否和下一条运动一起规划，0 代表立即规划，1 代表和下一条轨迹一起规划，当为 1 时，轨迹不会立即执行，样条曲线运动需至少连续下发三个点位，否则运动轨迹为直线</p> <p>(6) block                      logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>
注意	该运动暂不支持轨迹交融。

## 5.13. 机械臂示教

### 5.13.1. 关节示教 Service\_Joint\_Teach\_Cmd

RetVal = Service_Joint_Teach_Cmd(ArmSocket, num, direction, v, block)	
描述	该函数用于关节示教，关节从当前位置开始按照指定方向转动，接收到停止指令或者到达关节限位后停止。
参数	<p>(1) ArmSocket                      uint64</p> <p>Socket 句柄</p> <p>(2) num                              uint8</p>



	<p>示教关节的序号，1~7</p> <p>(3) direction                  uint8</p> <p>示教方向，0-负方向，1-正方向</p> <p>(4) v                                  uint8</p> <p>速度比例 1~100，即规划速度和加速度占关节最大线转速和加速度的百分比</p> <p>(5) block                          logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                          int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

### 5.13.2. 位置示教 Service\_Pos\_Teach\_Cmd

RetVal = Service\_Pos\_Teach\_Cmd(ArmSocket, type, direction, v, block)

描述	<p>该函数用于当前坐标系下（默认为当前工作坐标系下，调用 5.13.5 切换示教运动坐标系 Set_Teach_Frame 可切换为工具坐标系），笛卡尔空间位置示教。机械臂在当前工作坐标系下，按照指定坐标轴方向开始直线运动，接收到停止指令或者该处无逆解时停止。</p>
参数	<p>(1) ArmSocket                  uint64</p> <p>Socket 句柄</p> <p>(2) type                                  clib.rm_service.POS_TEACH_MODES</p> <p>示教类型</p> <p>(3) direction                          uint8</p>



	<p>示教方向，0-负方向，1-正方向</p> <p>(4) v                      uint8</p> <p>速度百分比系数，1~100</p> <p>(5) block                      logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

### 5.13.3. 姿态示教 Service\_Ort\_Teach\_Cmd

RetVal = Service\_Ort\_Teach\_Cmd(ArmSocket, type, direction, v, block)

描述	<p>该函数用于当前坐标系下（默认为当前工作坐标系下，调用 5.13.5 切换示教运动坐标系 Set_Teach_Frame 可切换为工具坐标系），笛卡尔空间末端姿态示教。机械臂在当前工作坐标系下，绕指定坐标轴旋转，接收到停止指令或者该处无逆解时停止。</p>
参数	<p>(1) ArmSocket                      uint64</p> <p>Socket 句柄</p> <p>(2) type                              clib.rm_service.ORT_TEACH_MODES</p> <p>示教类型</p> <p>(3) direction                      uint8</p> <p>示教方向，0-负方向，1-正方向</p> <p>(4) v                                  uint8</p> <p>速度比例 1~100，即规划速度和加速度占机械臂末端最大角</p>



	速度和角加速度的百分比  (5) block                      logical  false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。
返回值	(1) RetVal                      int32  成功返回：0；失败返回：错误码，查询 API 错误类型。

#### 5.13.4. 示教停止 Service\_Teach\_Stop\_Cmd

RetVal = Service_Teach_Stop_Cmd(ArmSocket, block)	
描述	该函数用于示教停止。
参数	(1) ArmSocket                      uint64  Socket 句柄  (2) block                      logical  false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。
返回值	(1) RetVal                      int32  成功返回：0；失败返回：错误码，查询 API 错误类型。

#### 5.13.5. 切换示教运动坐标系 Service\_Set\_Teach\_Frame

RetVal = Service_Set_Teach_Frame(ArmSocket, type, block)	
描述	该函数用于切换示教运动坐标系。
参数	(1) ArmSocket                      uint64  Socket 句柄  (2) type                      int32



	<p>0: 基座标运动, 1: 工具坐标系运动</p> <p>(3) block                      logical</p> <p>false-非阻塞, 发送后立即返回; true-阻塞, 等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回: 0; 失败返回: 错误码, 查询 API 错误类型。</p>

#### 5.13.6. 获取示教运动坐标系 Service\_Get\_Teach\_Frame

[RetVal, type] = Service_Get_Teach_Frame(ArmSocket)	
描述	该函数用于切换示教运动坐标系。
参数	<p>(1) ArmSocket                      uint64</p> <p>Socket 句柄</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回: 0; 失败返回: 错误码, 查询 API 错误类型。</p> <p>(2) type                      int32</p> <p>0: 基座标运动, 1: 工具坐标系运动</p>

### 5.14. 机械臂步进

#### 5.14.1. 关节步进 Service\_Joint\_Step\_Cmd

RetVal = Service_Joint_Step_Cmd(ArmSocket, num, step, v, block)	
描述	该函数用于关节步进。关节在当前位置下步进指定角度。
参数	<p>(1) ArmSocket                      uint64</p> <p>Socket 句柄</p>



	<p>(2) num                  uint8</p> <p>关节序号，1~7</p> <p>(3) step                  single</p> <p>步进的角度</p> <p>(4) v                      uint8</p> <p>速度比例 1~100，即规划速度和加速度占指定关节最大关节转速和关节加速度的百分比</p> <p>(5) block                  logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                  int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.14.2. 位置步进 Service\_Pos\_Step\_Cmd

RetVal = Service_Pos_Step_Cmd(ArmSocket, type, step, v, block)	
描述	该函数用于当前坐标系下（默认为当前工作坐标系下，调用 5.13.5 切换示教运动坐标系 Set_Teach_Frame 可切换为工具坐标系），位置步进。机械臂末端在当前工作坐标系下，朝指定坐标轴方向步进指定距离，到达位置返回成功指令，规划错误返回失败指令。
参数	<p>(1) ArmSocket                  uint64</p> <p>Socket 句柄</p> <p>(2) type                      clib.rm_service.POS_TEACH_MODES</p> <p>示教类型</p>





	<p>(3) step                      single</p> <p>    步进的距离，单位 m，精确到 0.001mm</p> <p>(4) v                          uint8</p> <p>    速度百分比系数，1~100</p> <p>(5) block                      logical</p> <p>    false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                      int32</p> <p>    成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.14.3. 姿态步进 Service\_Ort\_Step\_Cmd

RetVal = Service\_Ort\_Step\_Cmd(ArmSocket, type, step, v, block)

描述	<p>该函数用于当前坐标系下（默认为当前工作坐标系下，调用 5.13.5 切换示教运动坐标系 Set_Teach_Frame 可切换为工具坐标系），姿态步进。机械臂末端在当前工作坐标系下，绕指定坐标轴方向步进指定弧度，到达位置返回成功指令，规划错误返回失败指令。</p>
参数	<p>(1) ArmSocket                      uint64</p> <p>    Socket 句柄</p> <p>(2) type                          clib.rm_service.ORT_TEACH_MODES</p> <p>    示教类型</p> <p>(3) step                          single</p> <p>    步进的弧度，单位 rad，精确到 0.001rad</p> <p>(4) v                              uint8</p>



	<p>速度比例 1~100，即规划速度和加速度占机械臂末端最大角速度和角加速度的百分比</p> <p>(5) block                      logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

## 5.15. 控制器配置

### 5.15.1. 获取控制器状态 Service\_Get\_Controller\_State

[RetVal,voltage,current,temperature,sys_err] = Service_Get_Controller_State(ArmSocket)	
描述	该函数用于获取控制器状态。
参数	<p>(1) ArmSocket                      uint64</p> <p>Socket 句柄</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) voltage                      single</p> <p>返回的电压</p> <p>(3) current                      single</p> <p>返回的电流</p> <p>(4) temperature                      single</p>



	返回的温度
(5) sys_err	uint16
	控制器运行错误代码

#### 5.15.2. 设置 WiFi AP 模式设置 Service\_Set\_WiFi\_AP\_Data

RetVal = Service_Set_WiFi_AP_Data(ArmSocket, wifi_name, password)	
描述	该函数用于控制器 WiFi AP 模式设置，非阻塞模式，机械臂收到后更改参数，蜂鸣器响后代表更改成功，控制器重启，以 WIFI AP 模式通信。
参数	<p>(1) ArmSocket      uint64 Socket 句柄</p> <p>(2) wifi_name      string 控制器 wifi 名称</p> <p>(3) password      string wifi 密码</p>
返回值	<p>(1) RetVal      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.15.3. 设置 WiFi STA 模式设置 Service\_Set\_WiFi\_STA\_Data

RetVal = Service_Set_WiFi_STA_Data(ArmSocket, router_name, password)	
描述	该函数用于控制器 WiFi STA 模式设置，非阻塞模式，机械臂收到后更改参数，蜂鸣器响后代表更改成功，控制器重启，以 WIFI STA 模式通信。
参数	(1) ArmSocket      uint64



	<p>Socket 句柄</p> <p>(2) router_name      string</p> <p>路由器名称</p> <p>(3) password          string</p> <p>路由器 Wifi 密码</p>
返回值	<p>(1) RetVal            int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.15.4. 设置 UART\_USB 接口波特率 Service\_Set\_USB\_Data

RetVal = Service\_Set\_USB\_Data(ArmSocket, baudrate)

描述	<p>该函数用于控制器 UART_USB 接口波特率设置非阻塞模式，机械臂收到后更改参数，然后立即通过 UART-USB 接口与外界通信。</p> <p>该指令下发后控制器会记录当前波特率，断电重启后仍会使用该波特率对外通信。</p>
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) baudrate           int32</p> <p>波特率：9600，19200，38400，115200 和 460800，若用户设置其他数据，控制器会默认按照 460800 处理。</p>
返回值	<p>(1) RetVal            int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.15.5. 设置 RS485 配置 Service\_Set\_RS485

RetVal = Service\_Set\_RS485(ArmSocket, baudrate)



描述	<p>该函数用于控制器设置 RS485 配置。</p> <p>该指令下发后，若 Modbus 模式为打开状态，则会自动关闭，同时控制器会记录当前波特率，断电重启后仍会使用该波特率对外通信。</p>
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) baudrate          int32</p> <p>波特率：9600，19200，38400，115200 和 460800，若用户设置其他数据，控制器会默认按照 460800 处理。</p>
返回值	<p>(1) RetVal          int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.15.6. 设置机械臂电源 Service\_Set\_Arm\_Power

RetVal = Service_Set_Arm_Power (ArmSocket, cmd, block)	
描述	该函数用于设置机械臂电源。
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) cmd          logical</p> <p>true-上电，      false-断电</p> <p>(3) block          logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal          int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>



### 5.15.7. 获取机械臂电源 Service\_Get\_Arm\_Power\_State

RetVal = Service_Get_Arm_Power_State (ArmSocket, power)	
描述	该函数用于获取机械臂电源
参数	(1) ArmSocket          uint64 Socket 句柄
返回值	(1) RetVal              int32 成功返回：0；失败返回：错误码，查询 API 错误类型。  (2) power              int32 获取到的机械臂电源状态：1-上电，0-断电

### 5.15.8. 读取机械臂软件版本 Service\_Get\_Arm\_Software\_Version

[RetVal, plan_version, ctrl_version, kernal1, kernal2, product_version] = Service_Get_Arm_Software_Version(ArmSocket)	
描述	该函数用于读取机械臂软件版本
参数	(1) ArmSocket          uint64 Socket 句柄
返回值	(1) RetVal              int32 成功返回：0；失败返回：错误码，查询 API 错误类型。  (2) plan_version        string 读取到的用户接口内核版本号  (3) ctrl_version        string 实时内核版本号  (4) kernal1             string



	实时内核子核心 1 版本号 (5) kernal2                      string 实时内核子核心 2 版本号 (6) product_version              string 机械臂型号，仅 I 系列机械臂支持[-I]
--	---

#### 5.15.9. 获取控制器的累计运行时间 Service\_Get\_System\_Runtime

[RetVal, day, hour, min, sec] = Service\_Get\_System\_Runtime (ArmSocket)

描述	读取控制器的累计运行时间。
参数	(1) ArmSocket              uint64 Socket 句柄
返回值	(1) RetVal                  int32 成功返回：0；失败返回：错误码，查询 API 错误类型。 (2) day                      int32 天 (3) hour                      int32 小时 (4) min                      int32 分 (5) sec                      int32 秒

#### 5.15.10. 清空控制器累计运行时间 Service\_Clear\_System\_Runtime

RetVal = Service\_Clear\_System\_Runtime(ArmSocket, block)



描述	该函数用于清空控制器累计运行时间。
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) block                logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal              int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.15.11. 获取关节累计转动角度 Service\_Get\_Joint\_Odom

[RetVal, odom] = Service_Get_Joint_Odom(ArmSocket, odom)	
描述	该函数用于读取关节的累计转动角度。
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p>
返回值	<p>(1) RetVal              int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) odom                7 element vector single 各关节累计的转动角度值</p>

#### 5.15.12. 清除关节累计转动角度 Service\_Clear\_Joint\_Odom

RetVal = Service_Clear_Joint_Odom(ArmSocket, block)	
描述	该函数用于清空关节累计转动角度
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p>





	<p>(2) block                      logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.15.13. 配置高速网口 Service\_Set\_High\_Speed\_Eth

RetVal = Service_Set_High_Speed_Eth (ArmSocket, num, block)	
描述	该函数用于配置高速网口。
参数	<p>(1) ArmSocket                      uint64</p> <p>Socket 句柄</p> <p>(2) num                              uint8</p> <p>0-关闭 1-开启</p> <p>(3) block                              logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                              int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.15.14. 设置高速网口网络配置 Service\_Set\_High\_Ethernet--基础系列

RetVal = Service_Set_High_Ethernet(ArmSocket, ip, mask, gateway)	
描述	该函数用于设置高速网口网络配置[配置通讯内容]。
参数	<p>(1) ArmSocket                              uint64</p> <p>Socket 句柄</p>



	<div>(2) ip string</div> <div>网络地址</div> <div>(3) mask string</div> <div>子网掩码</div> <div>(4) gateway string</div> <div>网关</div>
返回值	<div>(1) RetVal int32</div> <div>成功返回：0；失败返回：错误码，查询 API 错误类型。</div>

#### 5.15.15. 获取高速网口网络配置 Service\_Get\_High\_Ethernet--基础系列

[RetVal, ip, mask, gateway, mac] = Service_Get_High_Ethernet(ArmSocket)	
描述	该函数用于获取高速网口网络配置[配置通讯内容]
参数	<div>(1) ArmSocket uint64</div> <div>Socket 句柄</div>
返回值	<div>(1) RetVal int32</div> <div>成功返回：0；失败返回：错误码，查询 API 错误类型。</div> <div>(2) ip string</div> <div>网络地址</div> <div>(3) mask string</div> <div>子网掩码</div> <div>(4) gateway string</div> <div>网关</div>



	(5) mac                      string  MAC 地址
--	---

#### 5.15.16. 保存参数 Service\_Save\_Device\_Info\_All--基础系列

RetVal = Service_Save_Device_Info_All(ArmSocket)	
描述	该函数用于保存所有参数
参数	(1) ArmSocket              uint64  Socket 句柄
返回值	(1) RetVal                  int32  成功返回：0；失败返回：错误码，查询 API 错误类型。

#### 5.15.17. 配置有线网卡 IP 地址 Service\_Set\_NetIP--I 系列

RetVal = Service_Set_NetIP(ArmSocket, ip)	
描述	该函数用于配置有线网卡 IP 地址[-I]
参数	(1) ArmSocket              uint64  Socket 句柄  (2) ip                          string  网络地址
返回值	(1) RetVal                  int32  成功返回：0；失败返回：错误码，查询 API 错误类型。

#### 5.15.18. 查询有线网卡网络信息 Service\_Get\_Wired\_Net--I 系列

[RetVal, ip, mask, mac] = Service_Get_Wired_Net(ArmSocket)	
描述	该函数用于查询有线网卡网络信息[-I]
参数	(1) ArmSocket              uint64



	Socket 句柄
返回值	<p>(1) RetVal            int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) ip                string</p> <p>网络地址</p> <p>(3) mask             string</p> <p>子网掩码</p> <p>(4) mac              string</p> <p>MAC 地址</p>

#### 5.15.19. 查询无线网卡网络信息 Service\_Get\_Wifi\_Net--I 系列

[RetVal, network] = Service_Get_Wifi_Net(ArmSocket)	
描述	该函数用于查询无线网卡网络信息[-I]
参数	<p>(1) ArmSocket        uint64</p> <p>Socket 句柄</p>
返回值	<p>(1) RetVal            int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) network           clib.rm_service.WiFi_Info</p> <p>无线网卡网络信息结构体</p>

代码示例：

```
% 创建 RM_Service 对象

robot = clib.rm_service.RM_Service();

% API 版本
```



```
ver = robot.Service_API_Version()

% 连接机械臂

IP = '192.168.1.18';

revtime = 200;

RetVal = robot.Service_RM_API_Init(65, clib.type.nullptr);

handle = robot.Service_Arm_Socket_Start(IP, 8080, revtime);

% 连接状态

RetVal = robot.Service_Arm_Socket_State(handle);

% 获取无线网卡网络信息

network = clib.rm_service.WiFi_Info;

[~, net] = robot.Service_Get_Wifi_Net(handle, network);

char(net.ip.int32)

% 断开连接

robot.Service_RM_API_Uninit()

robot.Service_Arm_Socket_Close(handle)
```

#### 5.15.20. 恢复网络出厂设置 Service\_Set\_Net\_Default--I 系列

RetVal = Service_Set_Net_Default(ArmSocket)	
描述	该函数用于恢复网络出厂设置[-I]
参数	(1) ArmSocket      uint64 Socket 句柄
返回值	(1) RetVal      int32 成功返回：0；失败返回：错误码，查询 API 错误类型。



#### 5.15.21. 清除系统错误代码 Service\_Clear\_System\_Err

RetVal = Service_Clear_System_Err(ArmSocket, block)	
描述	该函数用于清除系统错误
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) block                logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器 返回设置成功指令。</p>
返回值	<p>(1) RetVal              int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.15.22. 读取机械臂软件信息 Service\_Get\_Arm\_Software\_Info

[RetVal, software_info] = Service_Get_Arm_Software_Info(ArmSocket, software_info)	
描述	该函数用于读取机械臂软件信息
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) software_info      clib.rm_service.ArmSoftwareInfo 机械臂软件信息</p>
返回值	<p>(1) RetVal              int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) software_info      clib.rm_service.ArmSoftwareInfo 机械臂软件信息。</p>



### 5.15.23. 设置机械臂模式(仿真/真实)Service\_Set\_Arm\_Run\_Mode

[RetVal] = Service_Set_Arm_Run_Mode(ArmSocket, mode)	
描述	该函数用于设置机械臂模式(仿真/真实)
参数	(1) ArmSocket          uint64 Socket 句柄  (2) mode                  int32 模式 0:仿真 1:真实
返回值	(1) RetVal                  int32  成功返回：0；失败返回：错误码，查询 API 错误类型。

### 5.15.24. 获取机械臂模式(仿真/真实)Service\_Get\_Arm\_Run\_Mode

[RetVal, mode] = Service_Get_Arm_Run_Mode(ArmSocket)	
描述	该函数用于获取机械臂模式(仿真/真实)
参数	(1) ArmSocket          uint64 Socket 句柄
返回值	(1) RetVal                  int32 成功返回：0；失败返回：错误码，查询 API 错误类型。  (2) mode                  int32 模式 0:仿真 1:真实

## 5.16. IO 配置

机械臂具有 IO 端口，基础系列数量和分类如下所示：

数字输出：DO	4 路，可配置为 0~12V
---------	----------------



数字输入：DI	3 路，可配置为 0~12V
模拟输出：AO	4 路，输出电压 0~10V
模拟输入：AI	4 路，输入电压 0~10V

I 系列数量和分类如下所示：

数字 IO：DO/DI 复用	4 路，可配置为 0~24V
-------------------	----------------

#### 5.16.1. 设置数字 IO 模式 Service\_Set\_IO\_Mode--I 系列

RetVal = Service_Set_IO_Mode(ArmSocket, io_num, io_mode)	
描述	该函数用于设置数字 IO 模式[-I]。
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) io_num            uint8</p> <p>IO 端口号，范围：1~4</p> <p>(3) io_mode           uint8</p> <p>模式，0-通用输入模式、1-通用输出模式、2-输入开始功能复用模式、3-输入暂停功能复用模式、4-输入继续功能复用模式、5-输入急停功能复用模式、6-输入进入电流环拖动复用模式、7-输入进入力只动位置拖动模式（六维力版本可配置）、8-输入进入力只动姿态拖动模式（六维力版本可配置）、9-输入进入力位姿结合拖动复用模式（六维力版本可配置）、10-输入外部轴最大软限位复用模式（外部轴模式可配置）、11-输入外部轴最小软限位复用模式（外部轴模式可配置）。</p>
返回值	(1) RetVal            int32





成功返回：0；失败返回：错误码，查询 API 错误类型。

### 5.16.2. 设置数字 IO 输出状态 Service\_Set\_DO\_State

```
RetVal = Service_Set_DO_State(ArmSocket,io_num,state,block)
```

描述	该函数用于配置指定 IO 输出状态。
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) io_num              uint8 指定 IO 通道号，范围 1~4</p> <p>(3) state                logical IO 状态，true-高，      false-低</p> <p>(4) block                logical False-非阻塞，发送后立即返回； True-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal              int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

### 5.16.3. 查询指定 IO 状态 Service\_Get\_IO\_State

```
[RetVal, state, mode] = Service_Get_IO_State(ArmSocket, num)
```

描述	该函数用于查询指定 IO 状态。
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) num                 logical 指定数字 IO 通道号，范围 1~4</p>



返回值	(1) RetVal                  int32  成功返回：0；失败返回：错误码，查询 API 错误类型。
	(2) state                    uint8  输出参数 true-高，      false-低
	(3) mode                    uint8  模式，0-通用输入模式，1-通用输出模式、2-输入开始功能 复用模式、3-输入暂停功能复用模式、4-输入继续功能复用模式、5- 输入急停功能复用模式、6-输入进入电流环拖动复用模式、7-输入进 入力只动位置拖动模式（六维力版本可配置）、8-输入进入力只动姿 态拖动模式（六维力版本可配置）、9-输入进入力位姿结合拖动复用 模式（六维力版本可配置）、10-输入外部轴最大软限位复用模式（外 部轴模式可配置）、11-输入外部轴最小软限位复用模式（外部轴模 式可配置）。

#### 5.16.4. 查询数字 IO 输出状态 Service\_Get\_DO\_State--基础系列

[RetVal, state] = Service_Get_DO_State(ArmSocket, num)	
描述	该函数用于获取数字 IO 输出状态。
参数	(1) ArmSocket                  uint64  Socket 句柄
	(2) num                        uint8  指定 IO 通道号，范围 1~4
返回值	(1) RetVal                    int32  成功返回：0；失败返回：错误码，查询 API 错误类型。



	<p>(2) state                  uint8</p> <p>指定数字 IO 通道返回的状态，1-输出高，0-输出低。</p>
--	---

#### 5.16.5. 查询数字 IO 输入状态 Service\_Get\_DI\_State--基础系列

[RetVal, state] = Service_Get_DI_State(ArmSocket, num)	
描述	该函数用于获取数字 IO 输入状态。
参数	<p>(1) ArmSocket                  uint64</p> <p>Socket 句柄</p> <p>(2) num                          uint8</p> <p>指定 IO 通道号，范围 1~3</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) state                          uint8</p> <p>指定数字 IO 通道返回的状态，1-输入高，0-输入低。</p>

#### 5.16.6. 设置模拟 IO 输出状态 Service\_Set\_AO\_State--基础系列

RetVal = Service_Set_AO_State(ArmSocket, num, voltage, block)	
描述	该函数用于设置模拟 IO 输出状态。
参数	<p>(1) ArmSocket                  uint64</p> <p>Socket 句柄</p> <p>(2) num                          uint8</p> <p>指定通道号，1~4</p> <p>(3) voltage                      single</p> <p>IO 输出电压，分辨率 0.001V，范围：0~10000，代表输出</p>



	电压 0v~10v  (4) block                      logical  False-非阻塞，发送后立即返回； True-阻塞，等待机械臂到达位置或者规划失败。
返回值	(1) RetVal                      int32  成功返回：0；失败返回：错误码，查询 API 错误类型。

#### 5.16.7. 查询模拟 IO 输出状态 Service\_Get\_AO\_State--基础系列

[RetVal, voltage] = Service_Get_AO_State(ArmSocket, num)	
描述	该函数用于获取模拟 IO 输出状态。
参数	(1) ArmSocket                      uint64  Socket 句柄  (2) num                      uint8  指定 IO 通道号，范围 1~4
返回值	(1) RetVal                      int32  成功返回：0；失败返回：错误码，查询 API 错误类型。  (2) voltage                      uint8  IO 输出电压，分辨率 0.001V，范围：0~10000，代表输出电压 0v~10v

#### 5.16.8. 查询数字 IO 输入状态 Service\_Get\_AI\_State--基础系列

[RetVal, voltage] = Service_Get_AI_State(ArmSocket, num)	
描述	该函数用于获取模拟 IO 输入状态。
参数	(1) ArmSocket                      uint64



	<p>Socket 句柄</p> <p>(2) num                  uint8</p> <p>指定 IO 通道号，范围 1~4</p>
返回值	<p>(1) RetVal                  int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) voltage                  uint8</p> <p>IO 输入电压，分辨率 0.001V，范围：0~10000，代表输出电压 0v~10v</p>

#### 5.16.9. 查询所有 IO 输入状态 Service\_Get\_IO\_Input

[RetVal, DI_state, AI_voltage] = Service_Get_IO_Input(ArmSocket)	
描述	该函数用于查询所有 IO 输入状态。
参数	<p>(1) ArmSocket                  uint64</p> <p>Socket 句柄</p>
返回值	<p>(1) RetVal                  int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) DI_state                  4 element vector int32</p> <p>数字 IO 输入通道 1~4 状态数组地址，1-高，0-低</p> <p>(3) AI_voltage                  4 element vector single</p> <p>模拟 IO 输入通道 1~4 输入电压数组</p>

#### 5.16.10. 查询所有 IO 的输出状态 Service\_Get\_IO\_Output

[RetVal, DO_state, AO_voltage] = Service_Get_IO_Output(ArmSocket)	
描述	该函数用于查询所有 IO 输出状态。



参数	(1) ArmSocket          uint64  Socket 句柄
返回值	(1) RetVal              int32  成功返回：0；失败返回：错误码，查询 API 错误类型。  (2) DO_state            4 element vector int32  数字 IO 输出通道 1~4 状态数组地址，1-高，0-低  (3) AO_voltage          4 element vector single  模拟 IO 输出通道 1~4 输出电压数组

#### 5.16.11. 设置电源输出 Service\_Set\_Voltage--I 系列

RetVal = Service\_Set\_Voltage(ArmSocket, voltage\_type, start\_enable)

描述	该函数用于设置控制器端电源输出。
参数	(1) ArmSocket          uint64  Socket 句柄  (2) voltage_type        uint8  电源输出类型，范围：0~3(0-0V，2-12V，3-24V)  (3) start_enable        logical  true-开机启动时即输出此配置电压，false-取消开启启动配置电压
返回值	(1) RetVal              int32  成功返回：0；失败返回：错误码，查询 API 错误类型。

#### 5.16.12. 获取电源输出 Service\_Get\_Voltage--I 系列

[RetVal, voltage\_type] = Service\_Get\_Voltage(ArmSocket)



描述	该函数用于获取控制器端电源输出。
参数	(1) ArmSocket      uint64 Socket 句柄
返回值	(1) RetVal      int32 成功返回：0；失败返回：错误码，查询 API 错误类型。  (2) voltage_type    uint8 电源输出类型，范围：0~3(0-0V，2-12V，3-24V)

## 5.17. 末端工具 IO 配置

机械臂末端工具端具有 IO 端口，数量和分类如下所示：

电源输出	1 路，可配置为 0V/5V/12V/24V
数字 IO	2 路，输入输出可配置 输入：参考电平 12V~24V 输出：5~24V，与输出电压一致
通讯接口	1 路，可配置为 RS485

### 5.17.1. 设置工具端数字 IO 输出状态 Service\_Set\_Tool\_DO\_State

RetVal = Service_Set_Tool_DO_State(ArmSocket, num, state, block)	
描述	该函数用于配置工具端指定数字 IO 输出状态。
参数	(1) ArmSocket      uint64 Socket 句柄  (2) num      uint8 指定数字 IO 输出通道号，范围 1~2



	<p>(3) state                      logical</p> <p>    输入参数 true-高，      false-低</p> <p>(4) block                      logical</p> <p>    false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                      int32</p> <p>    成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.17.2. 设置工具端数字 IO 模式 Service\_Set\_Tool\_IO\_Mode

RetVal = Service_Set_Tool_IO_Mode(ArmSocket, num, state, block)	
描述	该函数用于设置数字输入输出 IO 模式。
参数	<p>(1) ArmSocket                      uint64</p> <p>    Socket 句柄</p> <p>(2) num                      uint8</p> <p>    指定数字输入通道号，范围 1~2</p> <p>(3) state                      logical</p> <p>    模式，false-输入状态，true-输出状态</p> <p>(4) block                      logical</p> <p>    false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                      int32</p> <p>    成功返回：0；失败返回：错误码，查询 API 错误类型。</p>





### 5.17.3. 查询工具端数字 IO 状态 Service\_Get\_Tool\_IO\_State

[RetVal, IO_Mode, IO_state] = Service_Get_Tool_IO_State(ArmSocket)	
描述	该函数用于查询工具端数字 IO 状态。
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p>
返回值	<p>(1) RetVal              int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) IO_Mode            2 element vector single</p> <p>指定数字 IO 通道模式(范围 1~2)，0-输入模式，1-输出模式</p> <p>(3) IO_state            2 element vector single</p> <p>指定数字 IO 通道当前输入状态(范围 1~2)，1-高电平，0-低电平</p>

### 5.17.4. 设置工具端电源输出 Service\_Set\_Tool\_Voltage

RetVal = Service_Set_Tool_Voltage(ArmSocket, type, block)	
描述	该函数用于设置工具端电源输出。
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) type                uint8</p> <p>电源输出类型，0-0V，1-5V，2-12V，3-24V</p> <p>(3) block               logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>



返回值	(1) RetVal            int32  成功返回：0；失败返回：错误码，查询 API 错误类型。
-----	---

#### 5.17.5. 获取工具端电源输出 Service\_Get\_Tool\_Voltage

[RetVal, voltage] = Service_Get_Tool_Voltage(ArmSocket)	
描述	该函数用于获取工具端电源输出。
参数	(1) ArmSocket            uint64  Socket 句柄
返回值	(1) RetVal            int32  成功返回：0；失败返回：错误码，查询 API 错误类型。  (2) voltage            uint8  读取回来的电源输出类型：0-0V，1-5V，2-12V，3-24V

### 5.18. 末端手爪控制（选配）

睿尔曼机械臂末端配备了因时机器人公司的 EG2-4C2 手爪，为了便于用户操作手爪，机械臂控制器对用户开放了手爪的 API 函数（手爪控制 API 与末端 modbus 功能互斥）。

#### 5.18.1. 配置手爪的开口度 Service\_Set\_Gripper\_Route

RetVal = Service_Set_Gripper_Route(ArmSocket, min_limit, max_limit, block)	
描述	该函数用于配置手爪的开口度。
参数	(1) ArmSocket            uint64  Socket 句柄  (2) min_limit            int32



	<p>手爪开口最小值，范围：0~1000，无单位量纲</p> <p>(3) Max_limit          int32</p> <p>手爪开口最大值，范围：0~1000，无单位量纲</p> <p>(4) block                  logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                  int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.18.2. 设置夹爪松开到最大位置 Service\_Set\_Gripper\_Release

RetVal = Service_Set_Gripper_Release (ArmSocket, speed, block, timeout)	
描述	该函数用于控制手爪以指定速度张开到最大开口处
参数	<p>(1) ArmSocket                  uint64</p> <p>Socket 句柄</p> <p>(2) speed                          int32</p> <p>手爪松开速度 ，范围 1~1000，无单位量纲</p> <p>(3) block                          logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p> <p>(4) timeout                          int32</p> <p>设置返回超时时间，阻塞模式生效，单位：秒</p>
返回值	<p>(1) RetVal                          int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>



### 5.18.3. 设置夹爪夹取 Service\_Set\_Gripper\_Pick

RetVal = Service_Set_Gripper_Pick(ArmSocket, speed, force, block, timeout)	
描述	该函数用于控制手爪以设定的速度去夹取，当手爪所受力矩大于设定的力矩阈值时，停止运动。
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) speed                int32 手爪夹取速度，范围：1~1000，无单位量纲</p> <p>(3) force                int32 手爪夹取力矩阈值，范围：50~1000，无单位量纲</p> <p>(4) block                logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p> <p>(5) timeout              int32 设置返回超时时间，阻塞模式生效，单位：秒</p>
返回值	<p>(1) RetVal                int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

### 5.18.4. 设置夹爪持续夹取 Service\_Set\_Gripper\_Pick\_On

RetVal = Service_Set_Gripper_Pick_On(ArmSocket, speed, force, block, timeout)	
描述	该函数用于控制手爪以设定的速度去持续夹取，当手爪所受力矩大于设定的力矩阈值时，停止运动。之后当手爪所受力矩小于设定力矩后，



	手爪继续持续夹取，直到再次手爪所受力矩大于设定的力矩阈值时，停止运动。
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) speed                int32 手爪夹取速度 ，范围：1~1000，无单位量纲</p> <p>(3) force                int32 手爪夹取力矩阈值，范围 ：50~1000，无单位量纲</p> <p>(4) block                logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p> <p>(5) timeout              int32 设置返回超时时间，阻塞模式生效，单位：秒</p>
返回值	<p>(1) RetVal                int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.18.5. 设置夹爪到指定开口位置 Service\_Set\_Gripper\_Position

RetVal = Service_Set_Gripper_Position (ArmSocket, position, block,timeout)	
描述	该函数用于控制手爪到达指定开口度位置。
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) position            int32 手爪指定开口度，范围 ：1~1000，无单位量纲</p>



	<p>(3) block                      logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p> <p>(4) timeout                      int32</p> <p>设置返回超时时间，阻塞模式生效，单位：秒</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.18.6. 获取夹爪状态 Service\_Get\_Gripper\_State

[RetVal,      gripper_state]      =      Service_Get_Gripper_State(ArmSocket, gripper_state)	
描述	该函数用于获取夹爪状态。
参数	<p>(1) ArmSocket                      uint64</p> <p>Socket 句柄</p> <p>(2) gripper_state      clib.rm_service.GripperState</p> <p>夹爪状态</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) gripper_state      clib.rm_service.GripperState</p> <p>夹爪状态</p>
备注	此接口需升级夹爪最新固件方可使用。

#### 5.19. 拖动示教及轨迹复现



睿尔曼机械臂采用关节电流环实现拖动示教，拖动示教及轨迹复现的配置函

数如下所示。

#### 5.19.1. 进入拖动示教模式 Service\_Start\_Drag\_Teach

RetVal = Service_Start_Drag_Teach (ArmSocket, block)	
描述	该函数用于控制机械臂进入拖动示教模式
参数	<div>(1) ArmSocket      uint64 Socket 句柄</div> <div>(2) block          logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</div>
返回值	<div>(1) RetVal          int32</div> <div>成功返回：0；失败返回：错误码，查询 API 错误类型。</div>

#### 5.19.2. 退出拖动示教模式 Service\_Stop\_Drag\_Teach

RetVal = Service_Stop_Drag_Teach (ArmSocket, block)	
描述	该函数用于控制机械臂退出拖动示教模式
参数	<div>(1) ArmSocket      uint64 Socket 句柄</div> <div>(2) block          logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</div>
返回值	<div>(1) RetVal          int32</div> <div>成功返回：0；失败返回：错误码，查询 API 错误类型。</div>



### 5.19.3. 拖动示教轨迹复现 Service\_Run\_Drag\_Trajectory

RetVal = Service_Run_Drag_Trajectory (ArmSocket, block)	
描述	该函数用于控制机械臂复现拖动示教的轨迹，必须在拖动示教结束后才能使用，同时保证机械臂位于拖动示教的起点位置。若当前位置没有位于轨迹复现起点，请先调用 120 函数，否则会返回报错信息。
参数	<div>(1) ArmSocket          uint64 Socket 句柄</div> <div>(2) block              logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</div>
返回值	<div>(1) RetVal            int32</div> <div>成功返回：0；失败返回：错误码，查询 API 错误类型。</div>

### 5.19.4. 拖动示教轨迹复现暂停 Service\_Pause\_Drag\_Trajectory

RetVal = Service_Pause_Drag_Trajectory (ArmSocket, block)	
描述	该函数用于控制机械臂在轨迹复现过程中的暂停。
参数	<div>(1) ArmSocket          uint64 Socket 句柄</div> <div>(2) block              logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</div>
返回值	<div>(1) RetVal            int32</div> <div>成功返回：0；失败返回：错误码，查询 API 错误类型。</div>





#### 5.19.5. 拖动示教轨迹复现继续 Service\_Continue\_Drag\_Trajectory

RetVal = Service_Continue_Drag_Trajectory (ArmSocket, block)	
描述	该函数用于控制机械臂在轨迹复现过程中暂停之后的继续，轨迹继续时，必须保证机械臂位于暂停时的位置，否则会报错，用户只能从开始位置重新复现轨迹。
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) block                logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.19.6. 拖动示教轨迹复现停止 Service\_Stop\_Drag\_Trajectory

RetVal = Service_Stop_Drag_Trajectory (ArmSocket, block)	
描述	该函数用于控制机械臂在轨迹复现过程中停止，停止后，不可继续。若要再次轨迹复现，只能从第一个轨迹点开始。
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) block                logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                int32</p>



成功返回：0；失败返回：错误码，查询 API 错误类型。

#### 5.19.7. 运动到轨迹起点 Service\_Drag\_Trajectory-Origin

RetVal = Service_Drag_Trajectory-Origin(ArmSocket, block)	
描述	轨迹复现前，必须控制机械臂运动到轨迹起点，如果设置正确，机械臂将以 20%的速度运动到轨迹起点。
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) block                logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.19.8. 复合模式拖动示教 Service\_Start\_Multi\_Drag\_Teach

RetVal        =        Service_Start_Multi_Drag_Teach(ArmSocket,        mode, singular_wall,block)	
描述	该函数用于复合模式拖动示。
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) mode                int32 拖动示教模式 0-电流环模式，1-使用末端六维力，只动位置，2-使用末端六维力 ，只动姿态，3-使用末端六维力，位置和姿态同时动</p>



	<p>(3) singular_wall</p> <p>仅在六维力模式拖动示教中生效，用于指定是否开启拖动奇异墙，0 表示关闭拖动奇异墙，1 表示开启拖动奇异墙</p> <p>(4) block                      logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.19.9. 保存拖动示教轨迹 Service\_Save\_Trajectory

[RetVal, num] = Service_Save_Trajectory(ArmSocket, filename)	
描述	该函数用于保存拖动示教轨迹。
参数	<p>(1) ArmSocket                      uint64</p> <p>Socket 句柄</p> <p>(2) filename                      string</p> <p>轨迹要保存路径及名称，例: c:/rm_test.txt</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) num                      int32</p> <p>轨迹点数</p>

#### 5.19.10. 设置力位混合控制 Service\_Set\_Force\_Postion

RetVal = Service_Set_Force_Postion(ArmSocket, sensor, mode, direction, N, block)
--



描述	该函数用于设置力位混合控制。在笛卡尔空间轨迹规划时，使用该功能可保证机械臂末端接触力恒定，使用时力的方向与机械臂运动方向不能在同一个方向。开启力位混合控制，执行笛卡尔空间运动，接收到运动完成反馈后，需要等待 2S 后继续下发下一条运动指令。
参数	<p>(1) ArmSocket          uint64 socket 句柄</p> <p>(2) sensor              int32 0-一维力；1-六维力</p> <p>(3) mode                int32 0-基坐标系力控；1-工具坐标系力控</p> <p>(4) direction           int32 力控方向；0-沿 X 轴；1-沿 Y 轴；2-沿 Z 轴；3-沿 RX 姿态方向；4-沿 RY 姿态方向；5-沿 RZ 姿态方向</p> <p>(5) N                    int32 力的大小，单位 N，精确到 0.1N</p> <p>(6) block                logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal              int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.19.11. 结束力位混合控制 Service\_Service\_Stop\_Force\_Postion

RetVal = Service\_Stop\_Force\_Postion (ArmSocket, block)



描述	该函数用于结束力位混合控制.
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) block                logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

代码示例：

```
% 创建 RM_Service 对象

robot = clib.rm_service.RM_Service();

% API 版本

ver = robot.Service_API_Version()

% 连接机械臂

IP = '192.168.1.18';

revtime = 200;

RetVal = robot.Service_RM_API_Init(65, clib.type.nullptr);

handle = robot.Service_Arm_Socket_Start(IP, 8080, revtime);

% 连接状态

RetVal = robot.Service_Arm_Socket_State(handle);

% 关节运动到起点

fjoint = [0,-20,-70,0,-90,0,0];
```



```
robot.Service_Movej_Cmd(handle, fjoint, 50, 0, true);

[ret, joint, pose, arm_err, sys_err] = robot.Service_Get_Current_Arm_State(handle)

[ret, joint, pose1, arm_err, sys_err]= robot.Service_Get_Current_Arm_State(handle)

% 设置力位混合控制

[RetVal] = robot.Service_Set_Force_Postion(handle,1,0,2,2,true);

pause(1);

% 笛卡尔空间运动

pose.position.x = pose.position.x + 0.05    % x 轴移动 0.05m

pose1.position.y = pose1.position.y + 0.05    % y 轴移动 0.05m

for i = 1:5

robot.Service_MoveI_Cmd(handle,pose,10,0,true)

pause(2)

robot.Service_MoveI_Cmd(handle,pose1,10,0,true)

pause(2)

end

% 结束力位混合控制

robot.Service_Stop_Force_Postion(handle,true)

% 断开连接

robot.Service_RM_API_UnInit()

robot.Service_Arm_Socket_Close(handle)
```

## 5.20. 末端六维力传感器的使用（选配）



睿尔曼 RM-65F 机械臂末端配备集成式六维力传感器，无需外部走线，用

户可直接通过 API 对六维力进行操作，获取六维力数据。

#### 5.20.1. 获取六维力数据 Service\_Get\_Force\_Data

[RetVal, force, zero_force, work_zero, tool_zero] = Service_Get_Force_Data(ArmSocket)	
描述	该函数用于获取当前六维力传感器得到的力和力矩信息，若要周期获取力数据，周期不能小于 50ms。
参数	(1) ArmSocket          uint64 Socket 句柄
返回值	(1) RetVal              int32 成功返回：0；失败返回：错误码，查询 API 错误类型。  (2) force                6 element vector single 返回的力和力矩数组地址，数组 6 个元素，依次为 Fx，Fy，Fz，Mx，My，Mz。其中，力的单位为 N；力矩单位为 Nm。  (3) zero_force          6 element vector single 传感器坐标系下系统受到的外力数据  (4) work_zero            6 element vector single 当前工作坐标系下系统受到的外力数据  (5) tool_zero            6 element vector single 当前工具坐标系下系统受到的外力数据

#### 5.20.2. 清空六维力数据 Service\_Clear\_Force\_Data

RetVal = Service_Clear_Force_Data(ArmSocket, block)	
描述	该函数用具清空六维力数据，即后续获得的所有数据都是基于当前数



	据的偏移量。。
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) block                  logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                  int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.20.3. 设置六维力重心参数 Service\_Set\_Force\_Sensor

RetVal = Service_Set_Force_Sensor (ArmSocket)	
描述	<p>设置六维力重心参数，六维力重新安装后，必须重新计算六维力所收到的初始力和重心。分别在不同姿态下，获取六维力的数据，用于计算重心位置。该指令下发后，机械臂以 20%的速度运动到各标定点，该过程不可中断，中断后必须重新标定。</p> <p><b>重要说明：</b>必须保证在机械臂静止状态下标定。</p>
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p>
返回值	<p>(1) RetVal                  int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.20.4. 手动标定六维力数据 Service\_Manual\_Set\_Force

RetVal = Service_Manual_Set_Force (ArmSocket, type, joint)	
描述	该手动标定流程，适用于空间狭窄工作区域，以防自动标定过程中机





	机械臂发生碰撞，用户手动标定六维力时，需要选择四个点位的数据，连续调用函数四次，机械臂开始自动沿用户设置的目标运动，并在此过程中计算六维力重心
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) type                  int32 点位，依次调用四次发送 1~4；</p> <p>(3) joint                  7 element vector single 关节角度</p>
返回值	<p>(1) RetVal                  int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.20.5. 退出标定流程 Service\_Stop\_Set\_Force\_Sensor

RetVal = Service_Stop_Set_Force_Sensor (ArmSocket, block)	
描述	在标定六/一维力过程中，如果发生意外，发送该指令，停止机械臂运动，退出标定流程。
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) block                  logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                  int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>



## 代码示例：

```
% 创建 RM_Service 对象

robot = clib.rm_service.RM_Service();

% API 版本

ver = robot.Service_API_Version()

% 连接机械臂

IP = '192.168.1.18';

revtime = 200;

RetVal = robot.Service_RM_API_Init(65, clib.type.nullptr);

handle = robot.Service_Arm_Socket_Start(IP, 8080, revtime);

% 连接状态

RetVal = robot.Service_Arm_Socket_State(handle);

% 六维力重心标定

joint1 = [0,0,0,0,0,90,0];

joint2 = [0,0,0,0,0,-90,0];

joint3 = [0,0,0,0,0,0,-90];

joint4 = [0,0,0,0,0,0,90];

[RetVal1] = robot.Service_Manual_Set_Force(handle,1,joint1);

pause(0.05);

[RetVal2] = robot.Service_Manual_Set_Force(handle,2,joint2);

pause(0.05);

[RetVal3] = robot.Service_Manual_Set_Force(handle,3,joint3);
```



```
pause(0.05);

[RetVal4] = robot.Service_Manual_Set_Force(handle,4,joint4);

pause(0.05);

% 断开连接

robot.Service_RM_API_UnInit()

robot.Service_Arm_Socket_Close(handle)
```

## 5.21. 末端五指灵巧手控制（选配）

睿尔曼 RM-65 机械臂末端配备了五指灵巧手，可通过 API 对灵巧手进行设置。

### 5.21.1. 设置灵巧手手势序号 Service\_Set\_Hand\_Posture

RetVal = Service_Set_Hand_Posture (ArmSocket, posture_num, block)		
描述	设置灵巧手手势序号，设置成功后，灵巧手按照预先保存在 Flash 中的手势运动。	
参数	<div><div>(1) ArmSocket</div><div>uint64</div><div>Socket 句柄</div></div> <div><div>(2) posture_num</div><div>int32</div><div>预先保存在灵巧手内的手势序号，范围：1~40</div></div> <div><div>(3) block</div><div>logical</div><div>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</div></div>	
返回值	(1) RetVal	int32



成功返回：0；失败返回：错误码，查询 API 错误类型。

### 5.21.2. 设置灵巧手动作序列序号 Service\_Set\_Hand\_Seq

RetVal = Service\_Set\_Hand\_Seq (ArmSocket, seq\_num, block)

**描述** 设置灵巧手动作序列序号，设置成功后，灵巧手按照预先保存在 Flash 中的动作序列运动。

**参数**

(1) ArmSocket          uint64  
Socket 句柄

(2) seq\_num          int32  
预先保存在灵巧手内的动作序列序号，范围：1~40

(3) block          logical  
false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。

**返回值**

(1) RetVal          int32  
成功返回：0；失败返回：错误码，查询 API 错误类型。

### 5.21.3. 设置灵巧手角度 Service\_Set\_Hand\_Angle

RetVal = Service\_Set\_Hand\_Angle(ArmSocket,angle, block)

**描述** 设置灵巧手角度，灵巧手有 6 个自由度，从 1~6 分别为小拇指，无名指，中指，食指，大拇指弯曲，大拇指旋转。

**参数**

(1) ArmSocket          uint64  
Socket 句柄

(2) angle          6 element vector int32  
手指角度数组，6 个元素分别代表 6 个自由度的角度。范围：



	0~1000。另外，-1 代表该自由度不执行任何操作，保持当前状态  (3) block                      logical  false-非阻塞，发送后立即返回； true-阻塞，等待控制器 返回设置成功指令。
返回值	(1) RetVal                      int32  成功返回：0；失败返回：错误码，查询 API 错误类型。

#### 5.21.4. 设置灵巧手各关节速度 Service\_Set\_Hand\_Speed

RetVal = Service_Set_Hand_Speed (ArmSocket, speed, block)	
描述	设置灵巧手各关节速度
参数	(1) ArmSocket                      uint64  Socket 句柄  (2) speed                      int32  灵巧手各关节速度设置，范围：1~1000  (3) block                      logical  false-非阻塞，发送后立即返回； true-阻塞，等待控制器 返回设置成功指令。
返回值	(1) RetVal                      int32  成功返回：0；失败返回：错误码，查询 API 错误类型。

#### 5.21.5. 设置灵巧手各关节力阈值 Service\_Set\_Hand\_Force

RetVal = Service_Set_Hand_Force (ArmSocket, force, block)	
描述	设置灵巧手各关节力阈值。
参数	(1) ArmSocket                      uint64



	<p>Socket 句柄</p> <p>(2) force                      int32</p> <p>灵巧手各关节力阈值设置，范围：1~1000，代表各关节的力矩阈值（四指握力 0~10N，拇指握力 0~15N）。</p> <p>(3) block                      logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

## 5.22. 末端传感器-一维力（选配）

睿尔曼机械臂末端接口板集成了一维力传感器，可获取 Z 方向的力，量程 200N，准度 0.5%FS。

### 5.22.1. 查询一维力数据 Service\_Get\_Fz

[RetVal, fz, work_zero, tool_zero] = Service_Get_Fz(ArmSocket)	
描述	该函数用于查询末端一维力数据。
参数	<p>(1) ArmSocket                      uint64</p> <p>Socket 句柄</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0。失败返回:错误码,查询 API 错误类型。</p> <p>(2) Fz                              single</p> <p>反馈的一维力原始数据 单位：N</p>



	<p>(3) zero_force      single</p> <p>传感器坐标系下系统受到的外力数据</p> <p>(4) work_zero      single</p> <p>当前工作坐标系下系统受到的外力数据</p> <p>(5) tool_zero      single</p> <p>当前工具坐标系下系统受到的外力数据</p>
备注	第一帧指令下发后，开始更新一维力数据，此时返回的数据有滞后性；请从第二帧的数据开始使用。若周期查询 Fz 数据，频率不能高于 40Hz。

#### 5.22.2. 清空一维力数据 Service\_Clear\_Fz

```
RetVal = Service_Clear_Fz(ArmSocket, block)
```

描述	该函数用于清零末端一维力数据。清空一维力数据后，后续所有获取到的数据都是基于当前的偏置。
参数	<p>(1) ArmSocket      uint64</p> <p>Socket 句柄</p> <p>(2) block      logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.22.3. 自动标定末端一维力数据 Service\_Auto\_Set\_Fz

```
RetVal = Service_Auto_Set_Fz(ArmSocket)
```



描述	该函数用于自动标定末端一维力数据。一维力重新安装后，必须重新计算一维力所受到的初始力和重心。分别在不同姿态下，获取一维力的数据，用于计算重心位置，该步骤对于基于一维力的力位混合控制操作具有重要意义
参数	(1) ArmSocket          uint64 Socket 句柄
返回值	(1) RetVal              int32 成功返回：0；失败返回：错误码，查询 API 错误类型。

#### 5.22.4. 手动标定末端一维力数据 Service\_Manual\_Set\_Fz

RetVal = Service\_Manual\_Set\_Fz(ArmSocket, joint1, joint2)

描述	该函数用于手动标定末端一维力数据。一维力重新安装后，必须重新计算一维力所受到的初始力和重心。该手动标定流程，适用于空间狭窄工作区域，以防自动标定过程中机械臂发生碰撞，用户可以手动选取 2 个位姿下发，当下发完后，机械臂开始自动沿用户设置的目标运动，并在此过程中计算一维力重心
参数	(1) ArmSocket          uint64 Socket 句柄  (2) joint1              7 element vector single 点位 1 关节角度  (3) joint2              7 element vector single 点位 2 关节角度
返回值	(1) RetVal              int32





成功返回：0；失败返回：错误码，查询 API 错误类型。

## 5.23. Modbus 配置

睿尔曼机械臂在控制器的航插和末端接口板航插处，各有 1 路 RS485 通讯接口，这两个 RS485 端口可通过接口配置为标准的 ModbusRTU 模式。然后通过接口对端口连接的外设进行读写操作。

注意：控制器的 RS485 接口在未配置为 Modbus RTU 模式的情况下，可用于用户对机械臂进行控制，这两种模式不可兼容。若要恢复机械臂控制模式，必须将该端口的 Modbus RTU 模式关闭。Modbus RTU 模式关闭后，系统会自动切换回机械臂控制模式，波特率 460800BPS，停止位 1，数据位 8，无检验。

同时，I 系列控制器支持 modbus-TCP 主站配置，可配置使用 modbus-TCP 主站，用于连接外部设备的 modbus-TCP 从站。

### 5.23.1. 设置通讯端口 Modbus RTU 模式 Service\_Set\_Modbus\_Mode

RetVal = Service\_Set\_Modbus\_Mode (ArmSocket, port, baudrate, timeout, block)

**描述** 该函数用于配置通讯端口 Modbus RTU 模式。机械臂启动后，要对通讯端口进行任何操作，必须先启动该指令，否则会返回报错信息。另外，机械臂会对用户的配置方式进行保存，机械臂重启后会自动恢复到用户断电之前配置的模式。

**参数**

(1) ArmSocket      uint64  
Socket 句柄

(2) port            int32



	<p>通讯端口，0-控制器 RS485 端口为 RTU 主站，1-末端接口板 RS485 接口为 RTU 主站，2-控制器 RS485 端口为 RTU 从站</p> <p>(3) baudrate          int32</p> <p>波特率，支持 9600，115200，460800 三种常见波特率</p> <p>(4) timeout          int32</p> <p>超时时间，单位百毫秒。对 Modbus 设备所有的读写指令，在规定的超时时间内未返回响应数据，则返回超时报错提醒。超时时间若设置为 0，则机械臂按 1 进行配置。</p> <p>(5) block          logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal          int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

### 5.23.2. 关闭通讯端口 Modbus RTU 模式 Service\_Close\_Modbus\_Mode

RetVal = Service\_Close\_Modbus\_Mode (ArmSocket, port , block)

描述	该函数用于关闭通讯端口 Modbus RTU 模式。
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) port          int32</p> <p>通讯端口，0-控制器 RS485 端口为 RTU 主站，1-末端接口板 RS485 接口为 RTU 主站，2-控制器 RS485 端口为 RTU 从站</p> <p>(3) block          logical</p>



	false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。
返回值	(1) RetVal            int32  成功返回：0；失败返回：错误码，查询 API 错误类型。

#### 5.23.3. 配置连接 ModbusTCP 从站 Service\_Set\_Modbustcp\_Mode--I 系列

RetVal = Service_Set_Modbustcp_Mode (ArmSocket, ip,port,timeout)	
描述	该函数用于配置连接 ModbusTCP 从站。
参数	(1) ArmSocket            uint64  Socket 句柄  (2) ip                    string  从机 IP 地址  (3) port                  int32  端口号  (4) timeout              int32  超时时间，单位秒。
返回值	(1) RetVal            int32  成功返回：0；失败返回：错误码，查询 API 错误类型。

#### 5.23.4. 配置关闭 ModbusTCP 从站 Service\_Close\_Modbustcp\_Mode--I 系列

RetVal = Service_Close_Modbustcp_Mode (ArmSocket)	
描述	该函数用于配置关闭 ModbusTCP 从站。
参数	(1) ArmSocket            uint64  Socket 句柄



返回值	(1) RetVal            int32  成功返回：0；失败返回：错误码，查询 API 错误类型。
-----	---

#### 5.23.5. 读线圈 Service\_Get\_Read\_Coils

[RetVal,coils_data] = Service_Get_Read_Coils(ArmSocket, port, address, num, device)	
描述	该函数用于读线圈。
参数	<p>(1) ArmSocket            uint64 Socket 句柄</p> <p>(2) port                  int32 通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口，3-控制器 ModbusTCP 设备</p> <p>(3) address              int32 线圈起始地址</p> <p>(4) num                   int32 要读的线圈的数量，该指令最多一次性支持读 8 个线圈数据，即返回的数据不会超过一个字节</p> <p>(5) device                int32 外设设备地址</p>
返回值	<p>(1) RetVal                int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) coils_data            int32 返回线圈状态</p>



### 5.23.6. 读离散输入量 Service\_Get\_Read\_Input\_Status

[RetVal,coils_data] = Service_Get_Read_Multiple_Coils(ArmSocket,port,address,num,device,len)	
描述	读离散输入量。
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) port                  int32 通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口，3-控制器 ModbusTCP 设备</p> <p>(3) address              int32 数据起始地址</p> <p>(4) num                   int32 要读的数据的数量,该指令最多一次性支持读 8 个离散量数据，即返回的数据不会超过一个字节</p> <p>(5) device                int32 外设设备地址</p> <p>(6) len                   int32 返回 coils_data 数组长度</p>
返回值	<p>(1) RetVal                int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) coils_data          vector int32 返回离散量</p>



### 5.23.7. 读保持寄存器 Service\_Get\_Read\_Holding\_Registers

[RetVal, coils_data] = Service_Get_Read_Holding_Registers(ArmSocket, port, address, device)	
描述	该函数用于读保持寄存器。该函数每次只能读 1 个寄存器，即 2 个字节的数据，不可一次性读取多个寄存器数据。
参数	<p>(1) ArmSocket      uint64 Socket 句柄</p> <p>(2) port            int32 通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口，3-控制器 ModbusTCP 设备</p> <p>(3) address        int32 数据起始地址</p> <p>(4) device          int32 外设设备地址</p>
返回值	<p>(1) RetVal          int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) coils_data      int32 返回寄存器数据</p>

### 5.23.8. 读输入寄存器 Service\_Get\_Read\_Input\_Registers

[RetVal, coils_data] = Service_Get_Read_Input_Registers(ArmSocket, port, address, device)	
描述	该函数用于读输入寄存器。



参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) port                  int32</p> <p>通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口，3-控制器 ModbusTCP 设备</p> <p>(3) address              int32</p> <p>数据起始地址</p> <p>(4) device                int32</p> <p>外设设备地址</p>
返回值	<p>(1) RetVal                int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) coils_data            int32</p> <p>返回寄存器数据</p>

#### 5.23.9. 写单圈数据 Service\_Write\_Single\_Coil

RetVal = Service\_Write\_Single\_Coil(ArmSocket, port,address,data,device, block)

描述	该函数用于写单圈数据。
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) port                  int32</p> <p>通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口，3-控制器 ModbusTCP 设备</p>



	<p>(3) address            int32</p> <p>线圈起始地址</p> <p>(4) data                int16</p> <p>要写入线圈的数据</p> <p>(5) device             int32</p> <p>外设设备地址</p> <p>(6) block              logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal             int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.23.10. 写单个寄存器 Service\_Write\_Single\_Register

RetVal = Service_Write_Single_Register(ArmSocket, port,address,data,device, block)	
描述	该函数用于写单个寄存器。
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) port                int32</p> <p>通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口，3-控制器 ModbusTCP 设备</p> <p>(3) address            int32</p> <p>寄存器起始地址。</p>





	<p>(4) data                      int16</p> <p>要写入寄存器的数据。</p> <p>(5) device                      int32</p> <p>外设设备地址</p> <p>(6) block                      logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.23.11. 写多个寄存器 Service\_Write\_Registers

RetVal = Service\_Write\_Registers(ArmSocket, port,address,num,single\_data, device, block)

描述	该函数用于写多个寄存器。
参数	<p>(1) ArmSocket                      uint64</p> <p>Socket 句柄</p> <p>(2) port                      int32</p> <p>通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口，3-控制器 ModbusTCP 设备</p> <p>(3) address                      int32</p> <p>寄存器起始地址</p> <p>(4) num                      int32</p> <p>写寄存器个数，寄存器每次写的数量不超过 10 个</p>



	<p>(5) single_data      vector uint8</p> <p>要写入寄存器的数据数组，类型：byte</p> <p>(6) device            int32</p> <p>外设备地址</p> <p>(7) block            logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal            int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.23.12. 写多圈数据 Service\_Write\_Coils

RetVal        =        Service_Write_Coils(ArmSocket,        port,address,num, coils_data,device, block)	
描述	该函数用于写多圈数据。
参数	<p>(1) ArmSocket        uint64</p> <p>Socket 句柄</p> <p>(2) port            int32</p> <p>通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口，3-控制器 ModbusTCP 设备</p> <p>(3) address        int32</p> <p>线圈起始地址。</p> <p>(4) num            int32</p> <p>写线圈个数，每次写的数量不超过 160 个</p>



	<p>(5) coils_data      vector uint8</p> <p>要写入线圈的数据数组，类型：byte。若线圈个数不大于 8，则写入的数据为 1 个字节；否则，则为多个数据的数组。</p> <p>(6) device            int32</p> <p>外设设备地址</p> <p>(7) block            logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal            int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

#### 5.23.13. 读多圈数据 Service\_Get\_Read\_Multiple\_Coils

[RetVal, coils\_data] = Service\_Get\_Read\_Multiple\_Coils(ArmSocket, port, address, num, device, len)

描述	该函数用于读多圈数据。
参数	<p>(1) ArmSocket      uint64</p> <p>Socket 句柄</p> <p>(2) port            int32</p> <p>通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口，3-控制器 ModbusTCP 设备</p> <p>(3) address        int32</p> <p>线圈起始地址</p> <p>(4) num            int32</p>



	<p>8 &lt; num &lt;= 120 要读的线圈的数量，该指令最多一次性支持读 120 个线圈数据，即 15 个 byte</p> <p>(5) device                      int32</p> <p>      外设设备地址</p> <p>(6) len                         int32</p> <p>      返回 coils_data 数组长度</p>
返回值	<p>(1) RetVal                      int32</p> <p>      成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) coils_data                 vector&lt;int32&gt;</p> <p>      返回线圈状态</p>

#### 5.23.14. 读多个保持寄存器 Service\_Read\_Multiple\_Holding\_Registers

[RetVal, coils\_data] = Service\_Read\_Multiple\_Holding\_Registers(ArmSocket, port, address, num, device, len)

描述	该函数用于读多个保持寄存器。
参数	<p>(1) ArmSocket                  uint64</p> <p>      Socket 句柄</p> <p>(2) port                        int32</p> <p>      通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口，3-控制器 ModbusTCP 设备</p> <p>(3) address                    int32</p> <p>      寄存器起始地址</p> <p>(4) num                        int32</p>



	<p><math>2 &lt; \text{num} &lt; 13</math> 要读的寄存器的数量，该指令最多一次性支持读 12 个寄存器数据，即 24 个</p> <p>(5) device                  int32 外设设备地址</p> <p>(6) len                      int32 返回 coils_data 数组长度</p>
返回值	<p>(1) RetVal                  int32 成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(2) coils_data 返回寄存器数据</p>

#### 5.23.15. 读多个输入寄存器 Service\_Read\_Multiple\_Input\_Registers

[RetVal, coils\_data] = Service\_Read\_Multiple\_Input\_Registers(ArmSocket, port, address, num, device, len)

描述	该函数用于读多个输入寄存器。
参数	<p>(1) ArmSocket              uint64 Socket 句柄</p> <p>(2) port                    int32 通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口，3-控制器 ModbusTCP 设备</p> <p>(3) address                int32 寄存器起始地址</p> <p>(4) num                    int32</p>



	<p><math>2 &lt; \text{num} &lt; 13</math> 要读的寄存器的数量，该指令最多一次性支持读 12 个寄存器数据，即 24 个</p> <p>(5) device                      int32</p> <p>      外设设备地址</p> <p>(6) len                          int32</p> <p>      返回 coils_data 数组长度</p>
返回值	<p>(3) RetVal                      int32</p> <p>      成功返回：0；失败返回：错误码，查询 API 错误类型。</p> <p>(4) coils_data</p> <p>      返回寄存器数据</p>

## 5.24. 升降机构

睿尔曼机械臂可集成自主研发升降机构。

### 5.24.1. 升降机构速度开环控制 Service\_Set\_Lift\_Speed

RetVal = Service_Set_Lift_Speed (ArmSocket, speed)	
描述	设置移动平台运动速度。
参数	<p>(1) ArmSocket                  uint64</p> <p>      Socket 句柄</p> <p>(2) speed                      int32</p> <p>      升降机速度百分比，-100 ~100</p> <p>      speed&lt;0：升降机构向下运动</p> <p>      speed&gt;0：升降机构向上运动</p>



	Speed=0: 升降机构停止运动
返回值	(1) RetVal            int32  成功返回：0；失败返回：错误码，查询 API 错误类型。

#### 5.24.2. 设置升降机构高度 Service\_Set\_Lift\_Height

RetVal = Service_Set_Lift_Height(ArmSocket, height,speed, block)	
描述	该函数用于设置升降机构高度。
参数	(1) ArmSocket            uint64  Socket 句柄  (2) height                int32  目标高度，单位 mm，范围：0~2600  (3) speed                 int32  升降机速度百分比，1~100  (4) block                 logical  false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。
返回值	(1) RetVal                int32  成功返回：0；失败返回：错误码，查询 API 错误类型。

#### 5.24.3. 获取升降机构状态 Service\_Get\_Lift\_State

[RetVal, height,current,err_flag,mode] = Service_Get_Lift_State(ArmSocket)	
描述	该函数用于获取升降机构状态。
参数	(1) ArmSocket            uint64  Socket 句柄



返回值	(1) RetVal            int32  成功返回：0；失败返回：错误码，查询 API 错误类型。
	(2) height            int32  当前升降机构高度，单位：mm，精度：1mm，范围：0~2300
	(3) current           int32  当前升降驱动电流，单位：mA，精度：1mA
	(4) err                int32  升降驱动错误代码，错误代码类型参考关节错误代码
	(5) mode              int32  当前升降状态，0-空闲，1-正方向速度运动，2-正方向位置运动，3-负方向速度运动，4-负方向位置运动

## 5.25. 透传力位混合控制补偿

针对睿尔曼带一维力和六维力版本的机械臂，用户除了可直接使用示教器调用底层的力位混合控制模块外，还可以将自定义的轨迹以周期性透传的形式结合底层的力位混合控制算法进行补偿。

在进行力的操作之前，如果未进行力数据标定，可使用清空一维力、六维力数据接口对零位进行标定。

### 5.25.1. 开启透传力位混合控制补偿模式 Service\_Start\_Force\_Position\_Move

RetVal = Service_Start_Force_Position_Move (ArmSocket, block)	
描述	开启透传力位混合控制补偿模式。
参数	(1) ArmSocket            uint64





	<p>Socket 句柄</p> <p>(2) block                      logical</p> <p>false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>

### 5.25.2. 力 位 混 合 控 制 补 偿 透 传 模 式 （ 关 节 角 度 ）

#### Service\_Force\_Position\_Move\_Joint

RetVal                      =                      Service_Force_Position_Move_Joint(ArmSocket, joint,sensor,mode,dir,force, follow)	
描述	该函数用于力位混合控制补偿透传模式（关节角度）。
参数	<p>(1) ArmSocket                      uint64</p> <p>Socket 句柄</p> <p>(2) joint                      7 element vector 只读 single</p> <p>目标关节角度</p> <p>(3) sensor                      uint8</p> <p>所使用传感器类型，0-一维力，1-六维力</p> <p>(4) mode                      uint8</p> <p>模式，0-沿基坐标系，1-沿工具端坐标系</p> <p>(5) dir                      int32</p> <p>力控方向，0~5 分别代表 X/Y/Z/Rx/Ry/Rz，其中一维力类型时默认方向为 Z 方向</p>



	<p>(6) force                      single</p> <p>力的大小 单位 N</p> <p>(7) follow                      logical</p> <p>表示驱动器的运动跟随效果，true 为高跟随，false 为低跟随</p>
返回值	<p>(1) RetVal                      int32</p> <p>成功返回：0；失败返回：错误码，查询 API 错误类型。</p>
备注	<p>备注 1：该功能只适用于一维力传感器和六维力传感器机械臂版本</p> <p>备注 2：透传周期越快，力位混合控制效果越好。基础系列 WIFI 和网口模式透传周期最快 20ms，USB 和 RS485 模式透传周期最快 10ms。高速网口的透传周期最快也可到 10ms，不过在使用该高速网口前，需要使用指令打开配置。另外 I 系列有线网口周期最快可达 5ms</p>

### 5.25.3. 力 位 混 合 控 制 补 偿 透 传 模 式 （ 位 姿 ）

Service\_Force\_Position\_Move\_Pose

RetVal                      =                      Service_Force_Position_Move_Pose(ArmSocket, pose,sensor,mode,dir,force, follow)	
描述	该函数用于力位混合控制补偿透传模式（位姿）。
参数	<p>(1) ArmSocket                      uint64</p> <p>Socket 句柄</p> <p>(2) pose                      clib.rm_service.Pose</p> <p>当前坐标系下目标位姿，位姿中包括姿态欧拉角和姿态四元数，四元数合理情况下，优先使用姿态四元数</p>



	<p>(3) sensor                  uint8</p> <p>    所使用传感器类型，0-一维力，1-六维力</p> <p>(4) mode                    uint8</p> <p>    模式，0-沿基坐标系，1-沿工具端坐标系</p> <p>(5) dir                      int32</p> <p>    力控方向，0~5 分别代表 X/Y/Z/Rx/Ry/Rz，其中一维力类型 时默认方向为 Z 方向</p> <p>(6) force                   single</p> <p>    力的大小 单位 0.1N</p> <p>(7) follow                  logical</p> <p>    是否高跟随</p>
返回值	<p>(1) RetVal                  int32</p> <p>    成功返回：0；失败返回：错误码，查询 API 错误类型。</p>
备注	<p>1、该功能只适用于一维力传感器和六维力传感器机械臂版本</p> <p>2、透传周期越快，力位混合控制效果越好。基础系列 WIFI 和网口模式透传周期最快 20ms，USB 和 RS485 模式透传周期最快 10ms。高速网口的透传周期最快也可到 10ms，不过在使用该高速网口前，需要使用指令打开配置。另外 I 系列有线网口周期最快可达 5ms。</p> <p>3、透传开始的起点务必为机械臂当前位姿，否则可能会力控补偿失败或机械臂无法运动</p>

#### 5.25.4. 关闭透传力位混合控制补偿模式 Service\_Stop\_Force\_Position\_Move

```
RetVal = Service_Stop_Force_Position_Move(ArmSocket, block)
```



描述	该函数用于关闭透传力位混合控制补偿模式。
参数	<div>(1) ArmSocket            uint64 Socket 句柄</div> <div>(2) block                logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器返回设置成功指令。</div>
返回值	<div>(1) RetVal                int32</div> <div>成功返回：0；失败返回：错误码，查询 API 错误类型。</div>

## 5.26. 算法工具接口

针对睿尔曼机械臂，提供正解、逆解等工具接口。

算法接口可单独使用，也可连接机械臂使用。连接机械臂使用时，为保证算法所用的参数是机械臂当前的数据，需调用获取工作、工具坐标系，获取安装角度等接口同步信息，否则算法将使用 API 与机械臂创建连接时机械臂的坐标系、安装角度等数据计算。

### 5.26.1. 初始化算法依赖数据 Service\_Algo\_Init\_Sys\_Data

Service_Algo_Init_Sys_Data(Mode, bType)	
描述	初始化算法依赖数据(不连接机械臂时调用， 连接机械臂会自动调用)。
参数	<div>(1) dMode                clib.rm_service.RobotType 机械臂型号</div> <div>(2) bType                clib.rm_service.SensorType</div>



传感器型号

### 5.26.2. 设置算法的安装角度 Service\_Algo\_Set\_Angle

Service\_Algo\_Set\_Angle(x, y, z)

描述	设置算法的安装角度参数（不连接机械臂时调用，连接机械臂时调用 Set_Install_Pose 设置安装方式参数会自动调用该接口）。
----	--

参数	(1) x                      single X 轴安装角度，单位度。
	(2) y                      single Y 轴安装角度，单位度。
	(3) z                      single Z 轴安装角度，单位度。

### 5.26.3. 获取算法的安装角度 Service\_Algo\_Get\_Angle

[x,y,z] = Service\_Algo\_Get\_Angle()

描述	获取算法的安装角度参数。
----	--------------

返回值	(1) x                      single X 轴安装角度，单位度。
	(2) y                      single Y 轴安装角度，单位度。
	(3) z                      single Z 轴安装角度，单位度。

### 5.26.4. 设置算法工作坐标系 Service\_Algo\_Set\_WorkFrame

Service\_Algo\_Set\_WorkFrame(coord\_work)



描述	设置算法工作坐标系（不连接机械臂时调用，连接机械臂时调用 Change_Work_Frame 切换工作坐标系会自动调用该接口）。
参数	(1) coord_work      clib.rm_service.FRAME 工作坐标系数据
备注	该接口调用时无须设置坐标系名称

#### 5.26.5. 获取当前工作坐标系 Service\_Algo\_Get\_Curr\_WorkFrame

[coord_work] = Service_Algo_Get_Curr_WorkFrame(coord_work)	
描述	设置算法工作坐标系。
返回值	(1) coord_work      clib.rm_service.FRAME 工作坐标系数据

#### 5.26.6. 设置算法工具坐标系和负载 Service\_Algo\_Set\_ToolFrame\_Params

Service_Algo_Set_ToolFrame(coord_tool)	
描述	设置算法工具坐标系和负载（不连接机械臂时调用，连接机械臂时调用 Change_Tool_Frame 切换工具坐标系会自动调用该接口）。
参数	(1) coord_tool      clib.rm_service.FRAME 工具坐标系数据

#### 5.26.7. 获取算法当前工具坐标系 Service\_Algo\_Get\_Curr\_ToolFrame

coord_tool = Service_Algo_Get_Curr_ToolFrame()	
描述	设置算法工具坐标系和负载。
返回值	(1) coord_tool      clib.rm_service.FRAME 坐标系数据
备注	该接口调用时无须设置坐标系名称



#### 5.26.8. 正解 Service\_Algo\_Forward\_Kinematics

pose = Service_Algo_Forward_Kinematics(joint)	
描述	用于睿尔曼机械臂正解计算。
参数	(1) joint                      7 element vector single 关节 1 到关节 7 角度，单位度。
返回值	(1) pose                      clib.rm_service.Pose 目标位姿

#### 5.26.9. 逆解 Service\_Algo\_inverse\_Kinematics

[RetVal, q_out] = Service_Algo_Inverse_Kinematics(q_in,q_pose, flag)	
描述	用于睿尔曼机械臂逆解计算。
参数	(1) q_in                      7 element vector single 上一时刻关节角，单位度。  (2) q_pose                      clib.rm_service.Pose 目标位姿。  (3) flag                      uint8 姿态参数类别：0-四元数；1-欧拉角
返回值	(1) RetVal                      int32  SYS_NORMAL：计算正常，CALCULATION_FAILED：计算失败；  (2) q_out                      7 element vector single 输出的关节角度，单位度。



#### 5.26.10. 计算平移、旋转运动位姿 Service\_Algo\_PoseMove

```
pose = Service_Algo_PoseMove(poseCurrent, deltaPosAndRot, frameMode)
```

描述	用于计算 Pos 和 Rot 沿某坐标系有一定的位移和旋转角度后，所得到的位姿数据。
参数	<p>(1) poseCurrent      6 element vector single 当前位姿，输入 position 和 euler。</p> <p>(2) deltaPosAndRot      6 element vector single 沿轴位移和绕轴旋转数组 (dx, dy, dz, rotx, roty, rotz) ， 位置移动单位：m，旋转单位：度</p> <p>(3) frameMode      int32 旋转角度：旋转角度，单位（度）</p>
返回值	<p>(1) pose              clib.rm_service.Pose 经平移、旋转后的位姿</p>

#### 5.26.11. 计算环绕运动位姿 Service\_Algo\_RotateMove

```
pose = Service_Algo_RotateMove(curr_joint, rotate_axis, rotate_angle, choose_axis)
```

描述	用于计算环绕运动位姿。
参数	<p>(1) curr_joint      7 element vector single 当前关节角度，单位度。</p> <p>(2) rotate_axis      int32 旋转轴：1：x 轴， 2：y 轴， 3：z 轴</p> <p>(3) rotate_angle      single</p>





	旋转角度：旋转角度，单位（度）  (4) choose_axis      clib.rm_service.Pose  指定计算时使用的坐标系
返回值	(1) pose              clib.rm_service.Pose  计算位姿结果

#### 5.26.12. 末端位姿转成工具位姿 Service\_Algo\_end2tool

pose = Service_Algo_End2Tool(eu_end)	
描述	末端位姿转成工具位姿。即为：机械臂末端在基坐标系下的位姿，转化成工具坐标系末端在工作坐标系下的位姿
参数	(1) eu_end              clib.rm_service.Pose  基于世界坐标系和默认工具坐标系的末端位姿
返回值	(2) pose              clib.rm_service.Pose  基于工作坐标系和工具坐标系的末端位姿

#### 5.26.13. 工具位姿转末端位姿 Service\_Algo\_tool2end

pose = Service_Algo_Tool2End(eu_tool)	
描述	工具位姿转末端位姿，即为：工具坐标系末端在工作坐标系下的位姿，转换成机械臂末端在基坐标系下的位姿。
参数	(1) eu_tool              clib.rm_service.Pose  基于工作坐标系和工具坐标系的末端位姿
返回值	(1) pose              clib.rm_service.Pose  基于世界坐标系和默认工具坐标系的末端位姿



#### 5.26.14. 四元数转欧拉角 Service\_Algo\_quaternion2euler

eu = Service_Algo_Quaternion2Euler(qua)	
描述	四元数转欧拉角。
参数	(1) qua                      clib.rm_service.Quat 四元数
返回值	(1) eu                      clib.rm_service.Euler 欧拉角

#### 5.26.15. 欧拉角转四元数 Service\_Algo\_euler2quaternion

qua = Service_Algo_Euler2Quaternion(eu)	
描述	欧拉角转四元数。
参数	(1) eu                      clib.rm_service.Euler 欧拉角
返回值	(1) qua                      clib.rm_service.Quat 四元数

#### 5.26.16. 欧拉角转旋转矩阵 Service\_Algo\_euler2matrix

[matrix,data] = Service_Algo_Euler2Matrix(_rot3)	
描述	欧拉角转旋转矩阵。
参数	(1) _rot3                      clib.rm_service.Euler 欧拉角
返回值	(1) matrix                      clib.rm_service.Matrix 旋转矩阵行数列数 irow, iline



	(2) data 4x4 single 旋转矩阵
--	--------------------------------

#### 5.26.17. 位姿转旋转矩阵 Service\_Algo\_pos2matrix

[matrix,data] = Service_Algo_Pos2Matrix(_point)	
描述	位姿转旋转矩阵。
参数	(1) _point clib.rm_service.Pose 位姿
返回值	(1) matrix clib.rm_service.Matrix 旋转矩阵 行数列数 irow, iline (2) data 4x4 single 旋转矩阵

#### 5.26.18. 旋转矩阵转位姿 Service\_Algo\_matrix2pos

pose = Service_Algo_Matrix2Pos(matrix,data)	
描述	旋转矩阵转位姿。
参数	(1) matrix clib.rm_service.Matrix 旋转矩阵的行数列数 irow, iline (2) data 4x4 single 旋转矩阵
返回值	(1) pose clib.rm_service.Pose 位姿



#### 5.26.19. 基坐标系转工作坐标系 Service\_Algo\_Base2WorkFrame

pose = Service_Algo_Base2WorkFrame(matrix, data, state)	
描述	基坐标系转工作坐标系。
参数	<p>(1) matrix                  clib.rm_service.Matrix 矩阵的行数列数 irow, iline</p> <p>(2) data                      4x4 single 工作坐标系在基坐标系下的矩阵</p> <p>(3) state                    clib.rm_service.Pose 工具端坐标在基坐标系下位姿</p>
返回值	<p>(1) pose                    clib.rm_service.Pose 基坐标系在工作坐标系下的位姿</p>

#### 5.26.20. 工作坐标系转基坐标系 Service\_Algo\_WorkFrame\_To\_Base

pose = Service_Algo_WorkFrame2Base(matrix, data, state)	
描述	工作坐标系转基坐标系。
参数	<p>(1) matrix                  clib.rm_service.Matrix 矩阵的行数列数 irow, iline</p> <p>(2) data                      4x4 single 工作坐标系在基坐标系下的矩阵</p> <p>(3) state                    clib.rm_service.Pose 工具端坐标在工作坐标系下位姿</p>
返回值	<p>(1) pose                    clib.rm_service.Pose 工作坐标系在基坐标系下的位姿</p>



#### 5.26.21. 计算沿工具坐标系运动位姿 Service\_Algo\_Cartesian\_Tool

```
pose = Service_Algo_Cartesian_Tool(curr_joint, move_lengthx, move_lengthy, move_lengthz)
```

<b>描述</b>	计算沿工具坐标系运动位姿。
<b>参数</b>	<p>(1) curr_joint      7 element vector single 当前关节角度，单位度</p> <p>(2) move_lengthx    single 沿 X 轴移动长度，米为单位</p> <p>(3) move_lengthy    single 沿 Y 轴移动长度，米为单位</p> <p>(4) move_lengthz    single 沿 Z 轴移动长度，米为单位</p>
<b>返回值</b>	<p>(1) pose            clib.rm_service.Pose 工作坐标系下的位姿</p>

#### 5.26.22. 设置算法关节最大限位 Service\_Set\_Algo\_Joint\_Max\_Limit

```
Service_Algo_Set_Joint_Max_Limit(joint_limit)
```

<b>描述</b>	设置算法关节最大限位。
<b>参数</b>	<p>(1) Joint_limit      7 element vector single 关节的最大限位数组</p>

#### 5.26.23. 获取算法关节最大限位 Service\_Algo\_Get\_Joint\_Max\_Limit

```
joint_limit = Service_Algo_Get_Joint_Max_Limit()
```

<b>描述</b>	设置算法关节最大限位。
-----------	-------------



返回值	(1) Joint_limit     7 element vector single 关节的最大限位数组
-----	--

#### 5.26.24. 设置算法关节最小限位 Service\_Algo\_Set\_Joint\_Min\_Limit

Service_Algo_Set_Joint_Min_Limit(joint_limit)	
描述	设置算法关节最大限位。
参数	(1) Joint_limit     7 element vector single 关节的最小限位数组

#### 5.26.25. 获取算法关节最小限位 Service\_Algo\_Get\_Joint\_Min\_Limit

joint_limit = Service_Algo_Get_Joint_Min_Limit( )	
描述	设置算法关节最大限位。
返回值	(1) Joint_limit     7 element vector single 关节的最小限位数组

#### 5.26.26. 设置算法关节最大速度 Service\_Algo\_Set\_Joint\_Max\_Speed

Service_Algo_Set_Joint_Max_Speed(joint_slim_max)	
描述	设置算法关节最大速度。
参数	(1) Joint_slim_max     7 element vector single 关节的最大速度数组

#### 5.26.27. 获取算法关节最大速度 Service\_Algo\_Get\_Joint\_Max\_Speed

joint_slim_max = Service_Algo_Get_Joint_Max_Speed()	
描述	设置算法关节最大速度。
返回值	(1) Joint_slim_max     7 element vector single



关节的最大速度数组

#### 5.26.28. 设置算法关节最大加速度 Service\_Algo\_Set\_Joint\_Max\_Acc

Service\_Algo\_Set\_Joint\_Max\_Acc(joint\_alim\_max)

**描述** 设置算法关节最大加速度。

**参数** (1) Joint\_alim\_max 7 element vector single  
关节的最大加速度数组

#### 5.26.29. 获取算法关节最大加速度 Service\_Algo\_Get\_Joint\_Max\_Acc

joint\_alim\_max = Service\_Algo\_Get\_Joint\_Max\_Acc( )

**描述** 设置算法关节最大加速度。

**返回值** (1) Joint\_alim\_max 7 element vector single  
返回的关节的最大加速度数组

代码示例：

```
% 创建 RM_Service 对象

robot = clib.rm_service.RM_Service();

% API 版本

ver = robot.Service_API_Version()

% 不连接机械臂使用算法接口，初始化算法依赖

rtype = clib.rm_service.RobotType.RM65;

stype = clib.rm_service.SensorType.B;

robot.Service_Algo_Init_Sys_Data(rtype, stype);

% 设置工作坐标系
```



```
frame = clib.rm_service.FRAME();

frame.pose.position.x = 0;

frame.pose.position.y = 0;

frame.pose.position.z = 0.483;

frame.pose.euler.rx = 0;

frame.pose.euler.ry = 0;

frame.pose.euler.rz = 0.535;

frame.x = 1;

frame.y = 2;

frame.z = 3;

robot.Service_Algo_Set_WorkFrame(frame);

% 获取坐标系

wframe = robot.Service_Algo_Get_Curr_WorkFrame(frame);

wframe.pose.position

% 正解计算

fjoint = [20,20,0,70,30,90,0];

retpose = robot.Service_Algo_Forward_Kinematics(fjoint);

retpose.position

retpose.euler

% 逆解计算

retpose.position.x = retpose.position.x + 0.001;

[ret, q_out] = robot.Service_Algo_Inverse_Kinematics(fjoint, retpose, 1)
```





## 5.27. 在线编程

### 5.27.1. 文件下发 Service\_Send\_TrajectoryFile

[RetVal,err_line] = Service_Send_TrajectoryFile(ArmSocket, send_param)	
描述	在线编程文件下发。
参数	<p>(1) ArmSocket          uint64 socket 句柄</p> <p>(2) send_param      clib.rm_service.Send_Project_Params 文件下发参数</p>
返回值	<p>(1) RetVal            int32 成功返回：0。失败返回：错误码，查询 API 错误类型</p> <p>(2) err_line          int32 有问题的工程行数</p>

### 5.27.2. 轨迹规划中改变速度比例系数 Service\_Set\_Plan\_Speed

RetVal = Service_Set_Plan_Speed(ArmSocket, speed, block)	
描述	该函数用于轨迹规划中改变速度比例系数。
参数	<p>(1) ArmSocket          uint64 socket 句柄</p> <p>(2) speed              int32 当前进度条的速度数据</p> <p>(3) block              logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器</p>



	返回设置成功指令。
返回值	(1) RetVal            int32  成功返回：0。失败返回：错误码，查询 API 错误类型

### 5.27.3. 文件树弹窗提醒 Service\_Popup

RetVal = Service\_Popup(ArmSocket, content, block)

描述	文件树弹窗提醒。本指令是控制器发送给示教器，返回值：是示教器发送给控制器。
参数	(1) ArmSocket            uint64  socket 句柄  (2) content  轨迹文件完整路径 例： c:/rm_file.txt  (3) block                logical  false-非阻塞，发送后立即返回； true-阻塞，等待控制器 返回设置成功指令。
返回值	(1) RetVal            int32  成功返回：0。失败返回：错误码，查询 API 错误类型



代码示例：

```
% 创建 RM_Service 对象

robot = clib.rm_service.RM_Service();

% API 版本

ver = robot.Service_API_Version()

% 连接机械臂

IP = '192.168.1.18';

revtime = 200;

RetVal = robot.Service_RM_API_Init(65, clib.type.nullptr);

handle = robot.Service_Arm_Socket_Start(IP, 8080, revtime);

% 连接状态

RetVal = robot.Service_Arm_Socket_State(handle);

% 文件下发，以 50%的速度运行

file_name = 'C:/Users/dell/Desktop/file/API/test.txt';

file_name_len = strlen(file_name);

[RetVal,err_line] =

robot.Service_Send_TrajectoryFile(ArmSocket,file_name,file_name_len,50,1,0);

% 断开连接

robot.Service_RM_API_Uninit()

robot.Service_Arm_Socket_Close(handle)
```

## 5.28. 机械臂状态主动上报



### 5.28.1. 设置主动上报配置 Service\_Set\_Realtime\_Push

RetVal = Service_Set_Realtime_Push(ArmSocket,config)	
描述	该函数用于设置主动上报接口配置。
参数	(1) ArmSocket      uint64 socket 句柄  (2) config          clib.rm_service.Realtime_Push_Config 主动上报接口配置
返回值	(1) RetVal          int32  成功返回：0。失败返回：错误码，查询 API 错误类型

### 5.28.2. 获取主动上报配置 Service\_Get\_Realtime\_Push

[RetVal,config] = Service_Get_Realtime_Push(ArmSocket,config)	
描述	该函数用于获取主动上报接口配置。
参数	(1) ArmSocket      uint64 socket 句柄  (2) config          clib.rm_service.Realtime_Push_Config 主动上报接口配置结构体
返回值	(1) RetVal          int32  成功返回：0。失败返回：错误码，查询 API 错误类型  (2) config          clib.rm_service.Realtime_Push_Config 主动上报接口配置结构体



### 5.28.3. 机械臂状态主动上报 Service\_Realtime\_Arm\_Joint\_State

[RetVal, RobotStatus] = Service_Get_Realtime_Arm_Joint_State(RobotStatus)	
描述	该函数可获取机械臂主动上报的状态信息。
参数	(1) RobotStatus     clib.rm_service.RobotStatus 机械臂状态信息结构体。
返回值	(1) RetVal             int32 成功返回：0。失败返回：错误码，查询 API 错误类型  (2) RobotStatus     clib.rm_service.RobotStatus 机械臂状态信息结构体。

代码示例：

```
% 创建 RM_Service 对象

robot = clib.rm_service.RM_Service();

% API 版本

ver = robot.Service_API_Version()

% 连接机械臂

IP = '192.168.1.18';

revtime = 200;

RetVal = robot.Service_RM_API_Init(65, clib.type.nullptr);

handle = robot.Service_Arm_Socket_Start(IP, 8080, revtime);

% 连接状态

RetVal = robot.Service_Arm_Socket_State(handle);

% 配置主动上报接口
```



```
config = clib.rm_service.Realtime_Push_Config();

config.port = 8099;

config.enable = true;

config.custom.joint_speed = 1;

robot.Service_Set_Realtime_Push(handle, config)

% 查询主动上报接口配置

after_config = clib.rm_service.Realtime_Push_Config();

[RetVal, after_config] = robot.Service_Get_Realtime_Push(handle, after_config)

% 获取机械臂主动上报的状态信息

status = clib.rm_service.RobotStatus;

for i = 1:10

[ret, robotstatus] = robot.Service_Get_Realtime_Arm_Joint_State(status);

robotstatus.joint_status.joint_position.single

pause(0.1)

end

% 断开连接

robot.Service_RM_API_UnInit()

robot.Service_Arm_Socket_Close(handle)
```

## 5.29. 通用扩展关节

### 5.29.1. 关节速度环控制 Service\_Expand\_Set\_Speed

```
RetVal = Service_Expand_Set_Speed(ArmSocket, speed, block);
```



描述	扩展关节速度环控制。
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) speed                int32 -50 表示最大速度的百分之五十反方向运动</p> <p>(3) block                logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器 返回设置成功指令。</p>
返回值	<p>(1) RetVal                int32 成功返回：0。失败返回：错误码，查询 API 错误类型</p>

#### 5.29.2. 关节位置环控制 Service\_Expand\_Set\_Pos

RetVal = Service\_Expand\_Set\_Pos(ArmSocket, pos, speed, block)

描述	该函数用于扩展关节位置环控制。
参数	<p>(1) ArmSocket          uint64 Socket 句柄</p> <p>(2) pos                    int32 升降关节精度 1mm    旋转关节精度 0.001°</p> <p>(3) speed                int32 50 表示最大速度的百分之五十,且速度必须大于 0</p> <p>(4) block                logical false-非阻塞，发送后立即返回； true-阻塞，等待控制器 返回设置成功指令。</p>



返回值	(1) RetVal            int32  成功返回：0。失败返回：错误码，查询 API 错误类型
-----	--

### 5.29.3. 扩展关节状态获取 Service\_Expand\_Get\_State

[RetVal,pos,err_flag,current,mode] = Service_Expand_Get_State(ArmSocket)	
描述	该函数用于获取扩展关节状态。
参数	<p>(1) ArmSocket            uint64 Socket 句柄</p> <p>(2) pos                    int32 当前升降机构高度，单位：mm，精度：1mm，如果是旋转关节则为角度 单位度，精度 0.001°</p> <p>(3) err                    int32 升降驱动错误代码，错误代码类型参考关节错误代码</p> <p>(4) current                int32 当前升降驱动电流，单位：mA，精度：1mA</p> <p>(5) mode                   int32 当前升降状态，0-空闲，1-正方向速度运动，2-正方向位置运动，3-负方向速度运动，4-负方向位置运动</p>
返回值	(1) RetVal            int32  成功返回：0。失败返回：错误码，查询 API 错误类型

代码示例：

```
% 创建 RM_Service 对象  
  
robot = clib.rm_service.RM_Service();
```





```
% API 版本

ver = robot.Service_API_Version()

% 连接机械臂

IP = '192.168.1.18';

revtime = 200;

RetVal = robot.Service_RM_API_Init(65, clib.type.nullptr);

handle = robot.Service_Arm_Socket_Start(IP, 8080, revtime);

% 连接状态

RetVal = robot.Service_Arm_Socket_State(handle);

% 获取扩展关节状态

[ret,pos,err_flag,current,mode] = robot.Service_Expand_Get_State(handle)

% 控制扩展关节以 20%的速度反方向运行

robot.Service_Expand_Set_Speed(handle, -20, false)

% 扩展关节位置环控制

robot.Service_Expand_Set_Pos(handle, 100, 20, true)

% 断开连接

robot.Service_RM_API_UnInit()

robot.Service_Arm_Socket_Close(handle)
```

## 5.30. 在线编程存储列表（I 系列）

### 5.30.1. 查询在线编程程序列表 Service\_Get\_Program\_Trajectory\_List

[RetVal,programlist]	=
----------------------	---



Service\_Get\_Program\_Trajectory\_List(ArmSocket,page\_num,page\_size,vague\_search,programlist)

描述	查询在线编程程序列表。
参数	<p>(1) ArmSocket      uint64 Socket 句柄</p> <p>(2) page_num      int32 页码（全部查询时此参数传 0）</p> <p>(3) page_size      int32 每页大小（全部查询时此参数传 0）</p> <p>(4) vague_search    string 模糊搜索（传递此参数可进行模糊查询）</p> <p>(5) programlist    clib.rm_service.ProgramTrajectoryData 符合条件的程序列表</p>
返回值	<p>(1) RetVal          int32 0-成功，失败返回：错误码，查询 API 错误类型。</p> <p>(2) programlist    clib.rm_service.ProgramTrajectoryData 符合条件的程序列表</p>

#### 5.30.2. 查询当前在线编程文件的运行状态 Service\_Get\_Program\_Run\_State

[RetVal,state] = Service\_Get\_Program\_Run\_State(ArmSocket,state)

描述	该函数用于查询当前在线编程文件的运行状态。
参数	<p>(1) ArmSocket      uint64 Socket 句柄</p>



	<p>(2) state                      clib.rm_service.ProgramRunState</p> <p>在线编程运行状态结构体</p>
返回值	<p>(1) RetVal                      int32</p> <p>0-成功，失败返回：错误码，查询 API 错误类型.</p> <p>(2) state                      clib.rm_service.ProgramRunState</p> <p>在线编程运行状态结构体</p>

### 5.30.3. 运行指定编号在线编程 Service\_Set\_Program\_ID\_Start

RetVal = Service\_Set\_Program\_ID\_Start(ArmSocket,id,speed,block)

描述	该函数用于运行指定编号在线编程。
参数	<p>(1) ArmSocket                  uint64</p> <p>Socket 句柄</p> <p>(2) id                          int32</p> <p>运行指定的 ID，1-100，存在轨迹可运行</p> <p>(3) speed                      int32</p> <p>1-100，需要运行轨迹的速度，可设置此参数为 0，按照存储的速度运行</p> <p>(4) block                      logical</p> <p>0-非阻塞，开始运行后返回；1-阻塞，等待在线编程程序运行结束返回</p>
返回值	<p>(1) RetVal                      int32</p> <p>0-成功，失败返回：错误码，查询 API 错误类型.</p>



#### 5.30.4. 删除指定 ID 的轨迹 Service\_Delete\_Program\_Trajectory

RetVal = Service_Delete_Program_Trajectory(ArmSocket,id)	
描述	该函数用于删除指定 ID 的轨迹。
参数	<div>(1) ArmSocket      uint64 Socket 句柄</div> <div>(2) id 指定需要删除的轨迹编号</div>
返回值	<div>(1) RetVal      int32</div> <div>0-成功，失败返回：错误码，查询 API 错误类型.</div>

#### 5.30.5. 修改指定编号轨迹的信息 Service\_Update\_Program\_Trajectory

[RetVal,state] = Service_Update_Program_Trajectory(ArmSocket,id,speed,name)	
描述	该函数用于修改指定编号轨迹的信息。
参数	<div>(1) ArmSocket      uint64 Socket 句柄</div> <div>(2) id 指定在线编程轨迹编号</div> <div>(3) speed 更新后的规划速度比例 1-100</div> <div>(4) name 更新后的文件名称（最大 10 个字节）</div>



返回值	(1) RetVal                  int32  0-成功, 失败返回: 错误码, 查询 API 错误类型.
-----	--

#### 5.30.6. 设置 IO 默认运行的在线编程文件编号

##### Service\_Set\_Default\_Run\_Program

RetVal = Service_Set_Default_Run_Program(ArmSocket,id)	
描述	该函数用于设置 IO 默认运行的在线编程文件编号。
参数	(1) ArmSocket                  uint64  Socket 句柄  (2) id                          int32  设置 IO 默认运行的在线编程文件编号, 支持 0-100, 0 代表取消设置
返回值	(1) RetVal                  int32  0-成功, 失败返回: 错误码, 查询 API 错误类型.

#### 5.30.7. 获取 IO 默认运行的在线编程文件编号

##### Service\_Get\_Default\_Run\_Program

[RetVal,id] = Service_Get_Default_Run_Program(ArmSocket)	
描述	该函数用于删除指定 ID 的轨迹。
参数	(1) ArmSocket                  uint64  Socket 句柄
返回值	(1) RetVal                  int32  0-成功, 失败返回: 错误码, 查询 API 错误类型.  (2) id



	IO 默认运行的在线编程文件编号，支持 0-100，0 代表无默认
--	-----------------------------------

代码示例：

```
% 创建 RM_Service 对象

robot = clib.rm_service.RM_Service();

% API 版本

ver = robot.Service_API_Version()

% 连接机械臂

IP = '192.168.1.18';

revtime = 200;

RetVal = robot.Service_RM_API_Init(65, clib.type.nullptr);

handle = robot.Service_Arm_Socket_Start(IP, 8080, revtime);

% 连接状态

RetVal = robot.Service_Arm_Socket_State(handle);

% 获取在线编程程序列表

program = clib.rm_service.ProgramTrajectoryData;

[ret, list] = robot.Service_Get_Program_Trajectory_List(handle, 1, 10, "", program);

if ret == 0

for i = 1:list.page_size

char(list.list(i).trajectory_name.int32)

end

end
```



```
% 运行 ID 为 1 的在线编程

[RetVal] = robot.Service_Set_Program_ID_Start(handle,1,20,false);

% 获取程序运行状态

[ret, state, id, plan_num, loop_num, loop_cont] =

robot.Service_Get_Program_Run_State(handle, 0);

while state ~= 0

    pause(1)

    [ret, state, id, plan_num, loop_num, loop_cont] =

    robot.Service_Get_Program_Run_State(handle, 0);

end

% 删除 ID 为 7 的在线编程

[RetVal] = robot.Service_Delete_Program_Trajectory(handle,7,true);

% 断开连接

robot.Service_RM_API_Uninit()

robot.Service_Arm_Socket_Close(handle)
```

## 5.31. 全局路点（I 系列）

### 5.31.1. 新增全局路点 Service\_Add\_Global\_Waypoint

[RetVal] = Service_Add_Global_Waypoint(ArmSocket,point_name,wframe_name,tframe _name,joint,pose)	
描述	新增全局路点。



参数	<p>(1) ArmSocket      uint64 Socket 句柄</p> <p>(2) point_name      string 全局路点名称</p> <p>(3) wframe_name      string 全局路点工作坐标系名称</p> <p>(4) tframe_name      string 全局路点工具坐标系名称</p> <p>(5) joint              7 element vector single 全局路点关节角度</p> <p>(6) pose              clib.rm_service.Pose 全局路点位姿</p>
返回值	<p>(1) RetVal            int32</p> <p>0-成功，失败返回：错误码，查询 API 错误类型.</p>
示例	<pre>pose = clib.rm_service.Pose pose.position.x = 0; pose.position.y = 0; pose.position.z = 0.8505; pose.euler.rx = 0; pose.euler.ry = 0; pose.euler.rz = 3.141; robot.Service_Add_Global_Waypoint(handle, "p3", "World",</pre>





	"Arm_Tip", [0,20,0,0,0,0,0], pose)
--	------------------------------------

### 5.31.2. 更新全局路点 Service\_Update\_Global\_Waypoint

[RetVal] = Service_Update_Global_Waypoint(ArmSocket,point_name,wframe_name,tframe_name,joint,pose)	
描述	更新全局路点。
参数	<p>(1) ArmSocket      uint64 Socket 句柄</p> <p>(2) point_name      string 全局路点名称</p> <p>(3) wframe_name      string 全局路点工作坐标系名称</p> <p>(4) tframe_name      string 全局路点工具坐标系名称</p> <p>(5) joint              7 element vector single 全局路点关节角度</p> <p>(6) pose              clib.rm_service.Pose 全局路点位姿</p>
返回值	<p>(1) RetVal              int32</p> <p>0-成功，失败返回：错误码，查询 API 错误类型.</p>
示例	<pre>pose = clib.rm_service.Pose pose.position.x = 0;</pre>



	<pre>pose.position.y = 0;  pose.position.z = 0.8505;  pose.euler.rx = 0;  pose.euler.ry = 0;  pose.euler.rz = 3.141;  robot.Service_Add_Global_Waypoint(handle,    "p3",    "World", "Arm_Tip", [0,20,0,0,0,0,0], pose)</pre>
--	---

### 5.31.3. 删除全局路点 Service\_Delete\_Global\_Waypoint

RetVal = Service_Delete_Global_Waypoint(ArmSocket,name)	
描述	该函数用于运行指定编号在线编程。
参数	<div>(1) ArmSocket          uint64 Socket 句柄</div> <div>(2) name                string 全局路点名称</div>
返回值	<div>(1) RetVal              int32</div> <div>0-成功，失败返回：错误码，查询 API 错误类型.</div>

### 5.31.4. 查询多个全局路点 Service\_Get\_Global\_Point\_List

<pre>[RetVal,point_list]  =Service_Get_Global_Point_List(ArmSocket,page_num,page_size,vague_sea rch,point_list)</pre>	
描述	该函数用于查询多个全局路点。
参数	<div>(1) ArmSocket          uint64</div>



	<p>Socket 句柄</p> <p>(2) page_num          int32</p> <p>页码（全部查询时此参数传 NULL）</p> <p>(3) page_size          int32</p> <p>几何模型每页大小（全部查询时此参数传 NULL）</p> <p>(4) vague_search      string</p> <p>模糊搜索（传递此参数可进行模糊查询）</p> <p>(5) point_list          clib.rm_service.WaypointsList</p> <p>全局路点列表查询结构体</p>
返回值	<p>(1) RetVal              int32</p> <p>0-成功，失败返回：错误码，查询 API 错误类型.</p> <p>(2) point_list          clib.rm_service.WaypointsList</p> <p>全局路点列表查询结构体</p>

#### 5.31.5. 查询指定全局路点 Service\_Given\_Global\_Waypoint

[RetVal,point] = Service_Given_Global_Waypoint(ArmSocket,name,point)	
描述	该函数用于查询指定全局路点。
参数	<p>(1) ArmSocket          uint64</p> <p>Socket 句柄</p> <p>(2) name                  只读 string</p> <p>指定全局路点名称</p> <p>(3) point                  clib.rm_service.Waypoint</p> <p>指定的全局路点参数</p>



返回值	(1) RetVal	int32
	0-成功，失败返回：错误码，查询 API 错误类型.	
	(2) point	clib.rm_service.Waypoint
	指定的全局路点参数	

## 5.32. 电子围栏和虚拟墙（I 系列）

I 系列机械臂具备电子围栏与虚拟墙功能，并提供了针对控制器所保存的电子围栏或虚拟墙几何模型参数的操作接口。用户可以通过这些接口，实现对电子围栏或虚拟墙的新增、查询、更新和删除操作，在使用中，可以灵活的使用保存在控制器中的参数配置，需要注意的是，目前控制器支持保存的参数要求不超过 10 个。

电子围栏功能通过精确设置参数，确保机械臂的轨迹规划、示教等运动均在设定的电子围栏范围内进行。当机械臂的运动轨迹可能超出电子围栏的界限时，系统会立即返回相应的错误码，并自动中止运动，从而有效保障机械臂的安全运行。需要注意的是，电子围栏目前仅支持长方体和点面矢量平面这两种形状，并且其仅在仿真模式下生效，为用户提供一个预演轨迹与进行轨迹优化的安全环境。

虚拟墙功能支持在电流环拖动示教与力控拖动示教两种模式下，对拖动范围进行精确限制。在这两种特定的示教模式下，用户可以借助虚拟墙功能，确保机械臂的拖动操作不会超出预设的范围。但请务必注意，虚拟墙功能目前支持长方体和球体两种形状，并仅在上述两种示教模式下有效。在其他操作模式下，此功能将自动失效。因此，请确保在正确的操作模式下使用虚拟墙功能，以充分发挥其限制拖动范围的作用。



### 5.32.1. 新增几何模型参数 Service\_Add\_Electronic\_Fence\_Config

[RetVal] = Service_Add_Electronic_Fence_Config(ArmSocket,config)	
描述	该函数用于新增几何模型参数，最多支持 10 个几何模型。
参数	<div>(1) ArmSocket          uint64 Socket 句柄</div> <div>(2) config          clib.rm_service.ElectronicFenceConfig 几何模型参数</div>
返回值	<div>(1) RetVal          int32</div> <div>0-成功，失败返回：错误码，查询 API 错误类型.</div>

### 5.32.2. 更新几何模型参数 Service\_Update\_Electronic\_Fence\_Config

[RetVal] = Service_Update_Electronic_Fence_Config(ArmSocket,config)	
描述	该函数用于更新几何模型参数，最多支持 10 个几何模型。
参数	<div>(1) ArmSocket          uint64 Socket 句柄</div> <div>(2) config          clib.rm_service.ElectronicFenceConfig 几何模型参数</div>
返回值	<div>(1) RetVal          int32</div> <div>0-成功，失败返回：错误码，查询 API 错误类型.</div>

### 5.32.3. 删除几何模型参数 Service\_Delete\_Electronic\_Fence\_Config

[RetVal] = Service_Delete_Electronic_Fence_Config(ArmSocket,name)	
描述	该函数用于删除几何模型参数，最多支持 10 个几何模型。



参数	(1) ArmSocket          uint64  Socket 句柄  (2) name                  string  指定几何模型名称
返回值	(1) RetVal                  int32  0-成功，失败返回：错误码，查询 API 错误类型.

#### 5.32.4. 查询所有几何模型名称 Service\_Get\_Electronic\_Fence\_List\_Names

[RetVal,names,len] = Service_Get_Electronic_Fence_List_Names(ArmSocket,names)	
描述	该函数用于查询所有几何模型名称，最多支持 10 个几何模型。
参数	(1) ArmSocket          uint64  Socket 句柄  (2) names                  clib.rm_service.NAMES  几何模型名称列表
返回值	(1) RetVal                  int32  0-成功，失败返回：错误码，查询 API 错误类型.  (2) names                  clib.rm_service.NAMES  几何模型名称列表，长度为实际存在几何模型  (3) len                          int32  几何模型名称列表长度

#### 5.32.5. 查询指定几何模型参数 Service\_Given\_Electronic\_Fence\_Config

[RetVal,config]	=
-----------------	---



Service\_Given\_Electronic\_Fence\_Config(ArmSocket,name,config)

描述	该函数用于查询所有几何模型名称，最多支持 10 个几何模型。
参数	<p>(1) ArmSocket      uint64 Socket 句柄</p> <p>(2) name          string 指定几何模型名称</p> <p>(3) config        clib.rm_service.ElectronicFenceConfig 几何模型参数</p>
返回值	<p>(1) RetVal          int32 0-成功，失败返回：错误码，查询 API 错误类型.</p> <p>(2) config        clib.rm_service.ElectronicFenceConfig 几何模型参数</p>

#### 5.32.6. 查询所有几何模型信息 Service\_Get\_Electronic\_Fence\_List\_Info

[RetVal,list,len] = Service\_Get\_Electronic\_Fence\_List\_Info(ArmSocket,list)

描述	该函数用于查询所有几何模型信息，最多支持 10 个几何模型。
参数	<p>(1) ArmSocket      uint64 Socket 句柄</p> <p>(2) list          clib.rm_service.ElectronicFenceConfigList 几何模型信息列表，长度为实际存在几何模型</p>
返回值	<p>(1) RetVal          int32 0-成功，失败返回：错误码，查询 API 错误类型.</p> <p>(2) list          clib.rm_service.ElectronicFenceConfigList</p>



	几何模型信息列表，长度为实际存在几何模型
(3) len	int32
	几何模型信息列表长度

#### 5.32.7. 设置电子围栏使能状态 Service\_Set\_Electronic\_Fence\_Enable

[RetVal] = Service\_Set\_Electronic\_Fence\_Enable(ArmSocket,enable\_state, in\_out\_side, effective\_region)

描述	设置电子围栏使能状态。
参数	<p>(1) ArmSocket      uint64</p> <p>Socket 句柄</p> <p>(2) enable_state    logical</p> <p>true 代表使能，false 代表禁使能</p> <p>(3) in_out_side      int32</p> <p>0-机器人在电子围栏内部，1-机器人在电子围栏外部</p> <p>(4) effective_region   int32</p> <p>0-针对整臂区域生效</p>
返回值	<p>(1) RetVal            int32</p> <p>0-成功，失败返回：错误码，查询 API 错误类型.</p>

#### 5.32.8. 获取电子围栏使能状态 Service\_Get\_Electronic\_Fence\_Enable

[RetVal,enable_state,            in_out_side,            effective_region]            = Service_Get_Electronic_Fence_Enable(ArmSocket)	
描述	获取电子围栏使能状态。





参数	(1) ArmSocket      uint64  Socket 句柄
返回值	(1) RetVal          int32  0-成功, 失败返回: 错误码, 查询 API 错误类型.  (2) enable_state    logical  true 代表使能, false 代表禁使能  (3) in_out_side      int32  0-机器人在电子围栏内部, 1-机器人在电子围栏外部  (4) effective_region   int32  0-针对整臂区域生效

#### 5.32.9. 设置当前电子围栏参数 Service\_Set\_Electronic\_Fence\_Config

[RetVal] = Service_Set_Electronic_Fence_Config(ArmSocket,config)	
描述	设置当前电子围栏参数。
参数	(1) ArmSocket      uint64  Socket 句柄  (2) config          clib.rm_service.ElectronicFenceConfig  当前电子围栏参数 (无需设置电子围栏名称)
返回值	(1) RetVal          int32  0-成功, 失败返回: 错误码, 查询 API 错误类型.

#### 5.32.10. 获取当前电子围栏参数 Service\_Get\_Electronic\_Fence\_Config

[RetVal,config] = Service_Get_Electronic_Fence_Config(ArmSocket,config)	
描述	获取电子围栏使能状态。



参数	(1) ArmSocket          uint64  Socket 句柄  (2) config    clib.rm_service.ElectronicFenceConfig  当前电子围栏参数（返回参数中不包含电子围栏名称）
返回值	(1) RetVal              int32  0-成功，失败返回：错误码，查询 API 错误类型。  (2) config    clib.rm_service.ElectronicFenceConfig  当前电子围栏参数（返回参数中不包含电子围栏名称）

#### 5.32.11. 设置虚拟墙使能状态 Service\_Set\_Virtual\_Wall\_Enable

[RetVal]        =        Service_Set_Virtual_Wall_Enable(ArmSocket,enable_state, in_out_side, effective_region)	
描述	设置虚拟墙使能状态。
参数	(1) ArmSocket          uint64  Socket 句柄  (2) enable_state    logical  true 代表使能，false 代表禁使能  (3) in_out_side        int32  0-机器人在虚拟墙内部  (4) effective_region    int32  1-针对末端生效
返回值	(1) RetVal              int32  0-成功，失败返回：错误码，查询 API 错误类型。



### 5.32.12. 获取虚拟墙使能状态 Service\_Get\_Virtual\_Wall\_Enable

[RetVal,enable_state, in_out_side, effective_region] = Service_Get_Virtual_Wall_Enable(ArmSocket)			
描述	获取虚拟墙使能状态。		
参数	(1) ArmSocket	uint64	Socket 句柄
返回值	(1) RetVal	int32	0-成功，失败返回：错误码，查询 API 错误类型。 (2) enable_state logical true 代表使能，false 代表禁使能 (3) in_out_side int32 0-机器人在虚拟墙内部 (4) effective_region int32 1-针对末端生效

### 5.32.13. 设置当前虚拟墙参数 Service\_Set\_Virtual\_Wall\_Config

[RetVal] = Service_Set_Virtual_Wall_Config(ArmSocket,config)			
描述	设置当前虚拟墙参数。		
参数	(1) ArmSocket	uint64	Socket 句柄 (2) config clib.rm_service.ElectronicFenceConfig 当前虚拟墙参数（无需设置虚拟墙名称）
返回值	(1) RetVal	int32	



0-成功，失败返回：错误码，查询 API 错误类型.

#### 5.32.14. 获取当前虚拟墙参数 Service\_Get\_Virtual\_Wall\_Config

```
[RetVal,config] = Service_Get_Virtual_Wall_Config(ArmSocket,config)
```

<b>描述</b>	获取虚拟墙使能状态。
<b>参数</b>	(1) ArmSocket      uint64 Socket 句柄 (2) config      clib.rm_service.ElectronicFenceConfig 当前虚拟墙参数（返回参数中不包含虚拟墙名称）
<b>返回值</b>	(1) RetVal      int32 0-成功，失败返回：错误码，查询 API 错误类型. (2) config      clib.rm_service.ElectronicFenceConfig 当前虚拟墙参数（返回参数中不包含虚拟墙名称）

### 5.33. 自碰撞安全检测（I 系列）

I 系列机械臂支持自碰撞安全检测，自碰撞安全检测使能状态下，可确保在轨迹规划、示教等运动过程中机械臂的各个部分不会相互碰撞，需要注意的是，以上自碰撞安全检测功能目前只在仿真模式下生效，用于进行预演轨迹与轨迹优化。

#### 5.33.1. 设置自碰撞安全检测使能状态 Service\_Set\_Self\_Collision\_Enable

```
[RetVal] = Service_Set_Self_Collision_Enable(ArmSocket,enable_state)
```

<b>描述</b>	该函数用于设置自碰撞安全检测使能状态。
-----------	---------------------



参数	(1) ArmSocket      uint64  Socket 句柄  (2) enable_state    logical  true 代表使能，false 代表禁使能
返回值	(1) RetVal            int32  0-成功，失败返回：错误码，查询 API 错误类型.

#### 5.33.2. 获取自碰撞安全检测使能状态 Service\_Get\_Self\_Collision\_Enable

[RetVal,enable\_state] = Service\_Get\_Self\_Collision\_Enable(ArmSocket)

描述	该函数用于获取自碰撞安全检测使能状态。
参数	(1) ArmSocket      uint64  Socket 句柄
返回值	(1) RetVal            int32  0-成功，失败返回：错误码，查询 API 错误类型.  (2) enable_state    logical  true 代表使能，false 代表禁使能