6. 接口函数详细说明

6.1 执行器接口

类名	功能描述	主要方法
AsioExecutor	异步 IO 执行器，管理后台工作线程处理通信和回调	`create()`，`get_io_context()`

- **异步执行器创建** `create`

字段	说明
函数名	`create`
函数原型	`static std::unique_ptr<AsioExecutor> create(uint16_t thread_num) noexcept;`
功能概述	工厂方法创建异步执行器实例，指定工作线程数量并初始化 IO 上下文
参数	`thread_num`：工作线程数量
返回值	`std::unique_ptr<AsioExecutor>`，指向异步执行器实例的智能指针
备注	会尝试配置实时调度和内存锁定以提升性能

代码示例：

```
auto executor = AsioExecutor::create(8);
```

- **获取 IO 上下文接口** `get_io_context`

字段	说明
函数名	`get_io_context`

字段	说明
函数原型	`virtual ExecutorPtr get_io_context() const noexcept = 0;`
功能概述	获取异步执行器管理的 IO 上下文共享指针
参数	无
返回值	`ExecutorPtr`，IO 上下文的共享指针
备注	用于传递给其他组件进行异步 IO 操作

代码示例：

```
auto io_context = executor->get_io_context();
```

6.2 机械臂控制接口

类名	功能描述	主要方法
Arm<6>	6 轴机械臂控制类，集成电机控制功能	`create()`，`init()`，`enable()`，`disable()`，`pvt()`，`csv()`，`mit()`，`state()`
Arm<7>	7 轴机械臂控制类，集成电机和末端执行器控制	`create()`，`init()`，`enable()`，`disable()`，`pvt()`，`csv()`，`mit()`，`state()`

- **机械臂类创建** `create`

字段	说明
函数名	`create`
函数原型	`template <MotorType m1, MotorType m2, MotorType m3, MotorType m4, MotorType m5, MotorType m6, EEFType eef_type, MotorType eef_motor> static std::unique_ptr<Arm<dof>> create() noexcept;`

字段	说明
功能概述	工厂方法创建 Arm 实例，通过指定各关节电机类型和末端执行器类型来创建机械臂控制对象
参数	`m1-m6`：6 个关节的电机类型 (OD/DM/ODM/EC) `eef_type`：末端执行器类型 (E2/G2/NA) `eef_motor`：末端执行器电机类型
返回值	`std::unique_ptr<Arm<dof>>`，指向 Arm 实例的智能指针
备注	使用模板参数指定电机配置，支持不同硬件配置的机械臂

代码示例：

```
auto arm = Arm<6>::create<MotorType::OD, MotorType::OD, MotorType::OD,
                          MotorType::DM, MotorType::DM, MotorType::DM,
                          EEFType::NA, MotorType::NA>();
```

- **机械臂初始化接口** `init`

字段	说明
函数名	`init`
函数原型	`virtual bool init(ExecutorPtr io_context, const std::string& interface, uint16_t spin_freq) noexcept = 0;`
功能概述	初始化机械臂，建立与硬件的通信连接
参数	`io_context`：异步 IO 执行器指针 `interface`：通信接口名称 ("can0") `spin_freq`：控制频率 (Hz)
返回值	`bool`，初始化成功返回 true，失败返回 false
备注	必须在其他操作前调用，控制频率影响控制精度和响应速度

代码示例：

```
auto executor = AsioExecutor::create(8);
auto io_context = executor->get_io_context();
bool success = arm->init(io_context, "can0", 250);
```

- **机械臂资源释放接口** `uninit`

字段	说明
函数名	`uninit`
函数原型	`virtual bool uninit() noexcept = 0;`
功能概述	反初始化机械臂，释放通信资源
参数	无
返回值	`bool`，反初始化成功返回 true，失败返回 false
备注	程序结束前应调用此方法释放资源

代码示例：

```
arm->disable();
arm->uninit();
```

- **机械臂连接检测接口** `ping`

字段	说明
函数名	`ping`
函数原型	`virtual bool ping() const noexcept = 0;`
功能概述	检测机械臂各组件连接状态，依次 ping 所有电机和末端执行器
参数	无
返回值	`bool`，所有组件连接正常返回 true，否则返回 false
备注	用于诊断硬件连接问题

代码示例：

```
if (!arm->ping()) {
    std::cout << "Hardware connection failed" << std::endl;
}
```

- **机械臂使能接口** `enable`

字段	说明
函数名	`enable`
函数原型	`virtual bool enable() const noexcept = 0;`
功能概述	使能机械臂，依次使能所有电机和末端执行器
参数	无
返回值	`bool`，所有组件使能成功返回 true，否则返回 false
备注	使能后机械臂可以接收控制命令

代码示例：

```
if (arm->enable()) {
    std::cout << "Arm enabled successfully" << std::endl;
}
```

- **机械臂失能接口** `disable`

字段	说明
函数名	`disable`
函数原型	`virtual bool disable() const noexcept = 0;`
功能概述	失能机械臂，依次失能所有电机和末端执行器
参数	无
返回值	`bool`，所有组件失能成功返回 true，否则返回 false

字段	说明
备注	失能后机械臂停止运动，进入安全状态

代码示例：

```
arm->disable();
```

- **机械臂回零接口** `set_zero`

字段	说明
函数名	`set_zero`
函数原型	`virtual bool set_zero() const noexcept = 0;`
功能概述	设置机械臂零位参考点，依次为所有电机和末端执行器设置零位
参数	无
返回值	`bool`，所有组件零位设置成功返回 true，否则返回 false
备注	用于校准机械臂位置参考

代码示例：

```
if (arm->set_zero()) {
    std::cout << "Zero position set successfully" << std::endl;
}
```

- **机械臂错误复位接口**

字段	说明
函数名	`reset_error`
函数原型	`virtual bool reset_error() const noexcept = 0;`
功能概述	复位机械臂各组件错误状态

字段	说明
参数	无
返回值	`bool`，所有组件错误复位成功返回 true，否则返回 false
备注	用于清除硬件错误状态

代码示例：

```
if (arm->reset_error()) {
    std::cout << "Errors reset successfully" << std::endl;
}
```

- **速度控制接口**

字段	说明
函数名	`csv`
函数原型	`virtual bool csv(const std::array<double, dof>& vel) const noexcept = 0;`
功能概述	发送循环同步速度控制帧，控制各关节目标速度
参数	`vel`：各关节目标速度数组 (rad/s)
返回值	`bool`，控制命令发送成功返回 true，否则返回 false
备注	需要持续调用以维持连续控制，适用于速度控制场景

代码示例：

```
std::array<double, 6> velocities = {0.1, 0.0, 0.0, 0.0, 0.0, 0.0};
arm->csv(velocities);
```

- **`pvt` 位置控制接口**

字段	说明

字段	说明
函数名	`pvt`
函数原型	`virtual bool pvt(const std::array<double, dof>& pos, const std::array<double, dof>& max_vel = DEFAULT_MAX_VEL, const std::array<double, dof>& max_eff = DEFAULT_MAX_EFF) const noexcept = 0;`
功能概述	发送位置-速度-时间控制帧，控制各关节目标位置，带速度和力矩限制
参数	`pos`：各关节目标位置数组 (rad) `max_vel`：最大速度限制 (rad/s) `max_eff`：最大力矩限制
返回值	`bool`，控制命令发送成功返回 true，否则返回 false
备注	适用于轨迹跟踪和路径规划，支持速度和力矩限制
备注	适用于轨迹跟踪和路径规划，支持速度和力矩限制

代码示例：

```
std::array<double, 6> positions = {0.5, 0.0, 0.0, 0.0, 0.0, 0.0};
arm->pvt(positions);
```

- **MIT 控制接口**

字段	说明
函数名	`mit`
函数原型	`virtual bool mit(const std::array<double, dof>& pos, const std::array<double, dof>& vel, const std::array<double, dof>& eff, const std::array<double, dof>& mit_kp, const std::array<double, dof>& mit_kd) const noexcept = 0;`
功能概述	发送 MIT 控制帧，实现位置-速度-力矩复合控制，支持前馈力矩和 PID 参数调节
参数	`pos`：目标位置数组 (rad) `vel`：目标速度数组 (rad/s)

字段	说明
	`eff` : 前馈力矩数组 `mit_kp` : 比例增益数组 `mit_kd` : 微分增益数组
返回值	`bool` , 控制命令发送成功返回 true，否则返回 false
备注	适用于精确力控制和阻抗控制, 支持 PID 参数调节

代码示例：

```
std::array<double, 6> pos = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
std::array<double, 6> vel = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
std::array<double, 6> eff = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
std::array<double, 6> kp = {100.0, 100.0, 100.0, 100.0, 100.0, 100.0};
std::array<double, 6> kd = {10.0, 10.0, 10.0, 10.0, 10.0, 10.0};
arm->mit(pos, vel, eff, kp, kd);
```

- **参数获取接口** `get_param`

字段	说明
函数名	`get_param`
函数原型	`virtual bool get_param(std::string_view name) const noexcept = 0;`
功能概述	发送参数获取请求, 获取机械臂参数值
参数	`name` : 参数名称 ("arm.control_mode" 或 "motor.")
返回值	`bool` , 请求发送成功返回 true，否则返回 false
备注	仅发送请求, 不等待响应, 需通过 params() 方法获取实际值

代码示例：

```
arm->get_param("arm.control_mode");
auto params = arm->params();
```

- **参数设置接口** `set_param`

字段	说明
函数名	`set_param`
函数原型	`virtual bool set_param(std::string_view name, ParamValue value) const noexcept = 0;`
功能概述	发送参数设置请求，设置机械臂参数值
参数	`name`：参数名称 `value`：参数值
返回值	`bool`，请求发送成功返回 true，否则返回 false
备注	设置后响应帧包含已设置的参数，无需再次调用 get_param

代码示例：

```
arm->set_param("arm.control_mode", MotorControlMode::MIT);
```

- **参数持久化接口** `persist_param`

字段	说明
函数名	`persist_param`
函数原型	`virtual bool persist_param(std::string_view name, ParamValue value) const noexcept = 0;`
功能概述	发送参数持久化请求，将参数值保存到硬件
参数	`name`：参数名称 `value`：参数值
返回值	`bool`，请求发送成功返回 true，否则返回 false
备注	持久化后参数在重启后仍然有效

代码示例：

```
arm->persist_param("arm.control_mode", MotorControlMode::MIT);
```

- **状态获取接口** `state`

字段	说明
函数名	`state`
函数原型	`virtual ArmState<dof> state() const noexcept = 0;`
功能概述	获取机械臂当前状态，包括位置、速度、力矩等信息
参数	无
返回值	`ArmState<dof>`，包含 is_valid、pos、vel、eff 字段的状态结构体
备注	线程安全，可多线程调用，返回状态快照

代码示例：

```
auto state = arm->state();
if (state.is_valid) {
    std::cout << "Position: " << state.pos[0] << std::endl;
}
```

- **参数值获取接口** `params`

字段	说明
函数名	`params`
函数原型	`virtual ArmParams params() const noexcept = 0;`
功能概述	获取机械臂当前参数，返回参数键值对映射
参数	无
返回值	`ArmParams`，参数名称到参数值的映射
备注	线程安全，可多线程调用，返回参数快照

代码示例：

```
auto params = arm->params();
auto control_mode = params["arm.control_mode"];
```

6.3 末端执行器接口

类名	功能描述	主要方法
EEF1	末端执行器控制类，支持 E2 和 G2 夹爪	`create()`，`init()`，`enable()`，`disable()`，`pvt()`，`state()`

- **末端执行器工厂方法**

字段	说明
函数名	`create`
函数原型	`template <EEFType eef_type, MotorType motor_type> static std::unique_ptr<EEF> create() noexcept;`
功能概述	工厂方法创建末端执行器实例，通过指定末端执行器类型和电机类型来创建控制对象
参数	`eef_type`：末端执行器类型 (G2/E2/NA) `motor_type`：驱动电机类型 (DM/ODM/OD)
返回值	`std::unique_ptr<EEF>`，指向末端执行器实例的智能指针
备注	支持 G2+DM、G2+ODM 等组合，E2+OD 组合待支持

代码示例：

```
auto eef = EEF<1>::create<EEFType::G2, MotorType::DM>();
```

- **末端执行器初始化接口**

字段	说明

字段	说明
函数名	`init`
函数原型	`virtual bool init(ExecutorPtr io_context, const std::string& interface, uint16_t spin_freq) noexcept = 0;`
功能概述	初始化末端执行器，建立与硬件的通信连接
参数	`io_context`：异步 IO 执行器指针 `interface`：通信接口名称 (　"can0") `spin_freq`：控制频率 (Hz)
返回值	`bool`，初始化成功返回 true，失败返回 false
备注	必须在其他操作前调用，控制频率影响控制精度和响应速度

代码示例：

```
bool success = eef->init(io_context, "can0", 250);
```

- **末端执行器资源释放接口** `uninit`

字段	说明
函数名	`uninit`
函数原型	`virtual bool uninit() const noexcept = 0;`
功能概述	反初始化末端执行器，释放通信资源
参数	无
返回值	`bool`，反初始化成功返回 true，失败返回 false
备注	程序结束前应调用此方法释放资源

代码示例：

```
eef->disable();
eef->uninit();
```

- **末端执行器连接检测接口** `ping`

字段	说明
函数名	`ping`
函数原型	`virtual bool ping() const noexcept = 0;`
功能概述	检测末端执行器连接状态
参数	无
返回值	`bool`，所有组件连接正常返回 true，否则返回 false
备注	用于诊断硬件连接问题

代码示例：

```
if (!eef->ping()) {
    std::cout << "connection failed" << std::endl;
}
```

- **末端执行器使能接口** `enable`

字段	说明
函数名	`enable`
函数原型	`virtual bool enable() const noexcept = 0;`
功能概述	使能末端执行器
参数	无
返回值	`bool`，所有组件使能成功返回 true，否则返回 false
备注	使能后末端执行器可以接收控制命令

代码示例：

```
if (eef->enable()) {
    std::cout << "eef enabled successfully" << std::endl;
}
```

- **末端执行器失能接口** `disable`

字段	说明
函数名	`disable`
函数原型	`virtual bool disable() const noexcept = 0;`
功能概述	失能末端执行器
参数	无
返回值	`bool`，所有组件使能成功返回 true，否则返回 false
备注	失能后末端执行器停止接收控制命令

代码示例：

```
arm->disable();
```

- **末端执行器回零接口** `set_zero`

字段	说明
函数名	`set_zero`
函数原型	`virtual bool set_zero() const noexcept = 0;`
功能概述	设置末端执行器零位参考点，为末端执行器设置零位
参数	无
返回值	`bool`，所有组件零位设置成功返回 true，否则返回 false
备注	用于校准末端执行器位置参考

- **末端执行器错误复位接口**

字段	说明
函数名	`reset_error`
函数原型	`virtual bool reset_error() const noexcept = 0;`
功能概述	复位末端执行器各组件错误状态
参数	无
返回值	`bool`，所有组件错误复位成功返回 true，否则返回 false
备注	用于清除硬件错误状态

- **末端执行器 PVT 控制接口**

字段	说明
函数名	`pvt`
函数原型	`virtual bool pvt(const EEFCommand<dof>& cmd) const noexcept = 0;`
功能概述	发送 PVT 控制命令到末端执行器，依次发送命令到各电机
参数	`cmd`：末端执行器控制命令结构体，包含位置、速度、力矩等信息
返回值	`bool`，控制命令发送成功返回 true，否则返回 false
备注	适用于轨迹跟踪和路径规划

代码示例：

```
EEFCommand<1> cmd;
cmd.pos[0] = 0.05;
eef->pvt(cmd);
```

- **末端执行器 MIT 控制接口**

字段	说明
函数名	`mit`
函数原型	`vvirtual bool mit(const EEFCommand<dof>& cmd) const noexcept = 0;`
功能概述	发送 MIT 控制帧，实现位置-速度-力矩复合控制，支持前馈力矩和 PID 参数调节
参数	`pos`：目标位置数组 (rad) `vel`：目标速度数组 (rad/s) `eff`：前馈力矩数组 `mit_kp`：比例增益数组 `mit_kd`：微分增益数组
返回值	`bool`，控制命令发送成功返回 true，否则返回 false
备注	适用于精确力控制和阻抗控制，支持 PID 参数调节

- **末端执行器参数获取接口** `get_param`

字段	说明
函数名	`get_param`
函数原型	`virtual bool get_param(std::string_view name) const noexcept = 0;`
功能概述	发送参数获取请求，获取末端执行器参数值
参数	`name`：参数名称 ("arm.control_mode" 或 "motor.")
返回值	`bool`，请求发送成功返回 true，否则返回 false
备注	仅发送请求，不等待响应，需通过 params() 方法获取实际值

- **末端执行器参数设置接口** `set_param`

字段	说明
函数名	`set_param`

字段	说明
函数原型	`virtual bool set_param(std::string_view name, ParamValue value) const noexcept = 0;`
功能概述	发送参数设置请求，设置机械臂参数值
参数	`name`：参数名称 `value`：参数值
返回值	`bool`，请求发送成功返回 true，否则返回 false
备注	设置后响应帧包含已设置的参数，无需再次调用 get_param

- **末端执行器参数持久化接口** `persist_param`

字段	说明
函数名	`persist_param`
函数原型	`virtual bool persist_param(std::string_view name, ParamValue value) const noexcept = 0;`
功能概述	发送参数持久化请求，将参数值保存到硬件
参数	`name`：参数名称 `value`：参数值
返回值	`bool`，请求发送成功返回 true，否则返回 false
备注	持久化后参数在重启后仍然有效

- **末端执行器状态获取接口**

字段	说明
函数名	`state`
函数原型	`virtual EEFState<dof> state() const noexcept = 0;`

字段	说明
功能概述	获取末端执行器当前状态，包括位置、速度、力矩等信息
参数	无
返回值	`EEFState<dof>`，包含 is_valid、pos、vel、eff 字段的状态结构体
备注	线程安全，可多线程调用，返回状态快照

代码示例：

```
auto state = eef->state();
if (state.is_valid) {
    std::cout << "Position: " << state.pos[0] << std::endl;
}
```

- **末端执行器参数值获取接口** `params`

字段	说明
函数名	`params`
函数原型	`virtual EEFParams params() const noexcept = 0;`
功能概述	获取末端执行器当前参数，返回参数键值对映射
参数	无
返回值	`EEFParams`，参数名称到参数值的映射
备注	线程安全，可多线程调用，返回参数快照

6.4 灵巧手

6.4.1 灵巧手接口

- **DexterousHand 灵巧手工厂方法** `create`

字段	说明
函数名	`create`
函数原型	`static std::unique_ptr<DexterousHand> create(uint8_t id, DexterousHandTypes type) noexcept;`
功能概述	工厂方法创建灵巧手实例，通过指定设备 ID 和设备类型来创建控制对象
参数	id: 设备 ID
type: 类型	
返回值	`std::unique_ptr<DexterousHand>`，指向灵巧手实例的智能指针
备注	创建失败时返回 nullptr

代码示例：

```
auto hand = DexterousHand::create(1, DexterousHandTypes::BRAINCO_REV02);
```

- **DexterousHand 灵巧手初始化接口** `init`

字段	说明
函数名	`init`
函数原型	`virtual bool init(ExecutorPtr io_context, const std::string& interface, uint16_t spin_freq) = 0;`
功能概述	初始化灵巧手，建立与硬件的通信连接
参数	`io_context`：异步 IO 执行器指针 `interface`：通信接口名称 ("can0") `spin_freq`：控制循环频率
返回值	`bool`，初始化成功返回 `true`，失败返回 `false`
备注	必须在其他操作前调用

代码示例：

```
bool success = hand->init(get_io_context, "can0", 250);
```

- **DexterousHand 灵巧手状态更新接口** `update_state`

字段	说明
函数名	`update_state`
函数原型	`virtual bool update_state() noexcept = 0;`
功能概述	更新灵巧手当前状态，包括位置、速度、力矩等信息，状态的更新可能会有延迟。
参数	无
返回值	`bool`，初始化成功返回 `true`，失败返回 `false`
备注	需要在获取状态前调用

代码示例：

```
hand->update_state();
```

- **DexterousHand 灵巧手状态获取接口** `state`

字段	说明
函数名	`state`
函数原型	`virtual HandState state() const noexcept = 0;`
功能概述	获取灵巧手当前状态，包括位置、速度、力矩等信息
参数	无
返回值	`HandState`，包含灵巧手状态信息的结构体
备注	线程不安全，避免多线程调用，返回状态快照

代码示例：

```
auto state = hand->state();
std::cout << "Hand position: " << state.pos[0] << std::endl;
```

- **DexterousHand 灵巧手位置控制接口** `set_pos`

字段	说明
函数名	`set_pos`
函数原型	`virtual bool set_pos(const HandState& cmd) = 0;`
功能概述	控制灵巧手以 cmd 中的参数状况移动到指定位置
参数	`cmd` : HandState 参数结构体
返回值	`bool` ，初始化成功返回 `true` ，失败返回 `false`
备注	线程不安全，避免多线程调用，返回状态快照

- **DexterousHand 灵巧手参数设置接口** `set_param`

字段	说明
函数名	`set_param`
函数原型	`virtual bool set_param(std::string_view name, ParamValue value) noexcept = 0;`
功能概述	设置灵巧手参数
参数	`name` : 参数名称 `value` : 参数值
返回值	`bool` ，初始化成功返回 `true` ，失败返回 `false`
备注	线程不安全，避免多线程调用，返回状态快照

6.4.2 各型号灵巧手默认出厂 id（仅供参考）

`Hand Type`	`Enum Value (Hex)`	`Default Device ID (Hex)`
INS_RH56DFX	0x00	0x01
INS_RH56BFX	0x01	0x01
INS_RH56E2	0x02	0x01
INS_RH56F1	0x03	0x01
BRAINCO_REVO2	0x21	0x7E (Left) / 0x7F (Right)
ROH_LITES001	0x40	0x02
ROH_A002	0x41	0x02
LINKERBOT_O6	0x51	0x28 (Left) / 0x27 (Right)

6.4.3 参数映射表

不同型号的机械手支持的配置参数各不相同。下表总结了当前支持的参数及其访问方式：

手型	参数名	访问方式	类型	描述
BrainCo Revo2	`hand_side`	R	`uint16`	手的方向（0=左手，1=右手）
	`unit_mode`	R/W	`0 / 1`	工作模式
	`id_485`	R/W	`uint16`	RS-485 节点 ID
	`baudrate`	R/W	`uint8`	通信波特率（115200 / 57600 / 19200 / 460800 / 1M / 2M / 5M）
INS RH56BFX	`hand_id`	R	`uint8`	设备 ID

手型	参数名	访问方式	类型	描述
	`baudrate`	R/W	`uint8`	CAN 波特率
	`current`	R	`uint16[6]`	各通道电流
	`pos_act`	R	`uint16[6]`	实际关节位置反馈
	`clear_error`	W	`0 / 1`	清除错误标志
	`force_calibration`	W	`0 / 1`	触发力校准过程
INS RH56DFX	`hand_id`	R	`uint16`	设备 ID
	`baudrate`	R/W	`uint8`	CAN 波特率
	`current`	R	`uint16[6]`	各通道电流
	`pos_act`	R	`uint16[6]`	实际关节位置反馈
	`clear_error`	W	`0 / 1`	清除错误标志
	`force_calibration`	W	`0 / 1`	触发力校准过程
INS RH56F1	`hand_id`	R	`uint16`	设备 ID
	`baudrate`	R/W	`uint32`	CAN 波特率
	`current`	R	`uint16[6]`	各通道电流
	`pos_act`	R	`uint16[6]`	实际关节位置反馈
	`clear_error`	W	`0 / 1`	清除错误标志

手型	参数名	访问方式	类型	描述
	`force_calibration`	W	`0 / 1`	触发力校准过程
INS RH56E2	`hand_id`	R	`uint16`	设备 ID
	`baudrate`	R/W	`uint8`	CAN 波特率
	`current`	R	`uint16[6]`	各通道电流
	`pos_act`	R	`uint16[6]`	实际关节位置反馈
	`clear_error`	W	`0 / 1`	清除错误标志
	`force_calibration`	W	`0 / 1`	触发力校准过程
ROH A002	`hand_id`	R/W	`uint8`	CAN 节点 ID

> ⚠ 对于不支持的参数，接口会忽略请求或返回 `false`。