设备状态服务接口

功能	C++	ROS
获取常规状态信息	✓	
获取基本状态	✓	✓
获取控制模式	✓	✓
获取当前场景（步态）	✓	✓
获取上桩状态	✓	
获取关节状态信息	✓	

1. 获取常规状态信息

1.1. C++ 接口

获取机器人常规状态信息，绝大部分本体信息均能在该结构体。

`rcClientInterfaceGetCommonStatus` 简介

```
/**
 * @brief 获取被控机器人常规状态信息(机器人状态反馈的主要接口)。
 *
 * 该函数用于获取机器人的常规状态信息。
 *
 * @param com_state 指向保存机器人状态信息的结构体指针。
 * @return bool 返回 `true` 表示获取成功，返回 `false` 表示获取失败。
 */
bool rcClientInterfaceGetCommonStatus(common::ROBOT_COMMON_STATUS *com_state);
```

结构体定义

common::ROBOT_COMMON_STATUS C++

```cpp
typedef struct ROBOT_COMMON_STATUS_{
    int network; //机器人当前网络状态
    BODY_POSTURE_STATE belie_or_stand; //机器人处于站立趴下状态
    bool emergency; //是否处于紧急状态
    bool avoidance; //自动避障状态
    long int heartbeat; //心跳
    float cur_speed; //当前速度
    float cur_height; //当前高度
    MC_QUATERNIONS position; //机器人世界坐标系下的位置，xyz与yaw角
    float max_height; //机器人最大高度
    float max_speed; //机器人最大速度
    SCENE_SWITCH_STATE cur_scene_switch_state; //切换场景是否成功
    SCENE_TYPE cur_scene; //当前处于的场景
    DRIVER_ENABLE_STATE driver_enable_state;
    BODY_ODOMETRY odometry; //机器人里程计信息
    float tip_position_wrt_hip[LEG_NUMBER*3];//足端相对髋关节坐标系位置
    float vf_planned_tip_position_wrt_base[LEG_NUMBER * 3];
    float touch_detection[LEG_NUMBER];//触地检测，范围从0到1.0，等于1.0为绝对触地，后期会变为概率，但目前还是只有1或者0两种可能
    float target_body_orientation[3]; //raw pitch yaw

    //错误代码
    int error_code;
    bool has_error;
    unsigned int error_wheel[CODE_WHEEL_NUM];
    unsigned int warn_wheel[CODE_WHEEL_NUM];
    unsigned int note_wheel[CODE_WHEEL_NUM];
    unsigned int joint_error_wheel[JOINT_CODE_WHEEL_NUM];
    unsigned int joint_warn_wheel[JOINT_CODE_WHEEL_NUM];
    unsigned int joint_note_wheel[JOINT_CODE_WHEEL_NUM];
    ROBOT_CHARGE_STATUS charge_status;
    bool salute_state;
    NAV_OR_JOY_MODE joy_mode;
    EQA_STATE eqa_state;

    bool guardian_switch; //表示当前停障功能的开关的状态
    bool visual_foothold_switch; //表示当前视觉落脚点功能开关的状态
    GUARDIAN_VELOCITY_DECAY_RATIO guardian_velocity_decay_ratio; //用于显示前左右三个方位的障碍

    LIE_DOWN_ADJUST_ lie_down_adjust; //用于判断趴下前是否正在做足端位置调整
    float speed_bar_on_controller; //机器人遥控器上当前选中的线速度
    CALIBRATE_POSE_STATE calibrate_pos_state;
    int self_test_code;
    bool encoder_calibrate_state;
```

```
    bool nav_pos_state;//当前导航定位位置是否正常
    bool nav_init_pos;//是否初始化导航定位
    TERRAIN_SEG_DETECTION_RESULTS terrain_seg_result;
    bool uwb_state;
    STAIRS_HEIGH_STATE stairs_state;
    bool weight_load_switch;
    float cur_weight_load;
    bool exceeding_weight_load;
}ROBOT_COMMON_STATUS;
```

使用案例

```
common::ROBOT_COMMON_STATUS com_state;
bool success = rcClientInterfaceGetCommonStatus(&com_state);
if (success) {
    std::cout << "Common Status Retrieved Successfully." << std::endl;
    // 输出或处理 com_state 数据
} else {
    std::cout << "Failed to Retrieve Common Status." << std::endl;
}
```

注意事项

- `com_state` 必须指向有效的 `common::ROBOT_COMMON_STATUS` 结构体。

2. 获取基本状态

2.1. C++ 接口

参考1.1 `rcClientInterfaceGetCommonStatus`

使用案例

```
common::ROBOT_COMMON_STATUS com_state;
bool success = rcClientInterfaceGetCommonStatus(&com_state);
if (success) {
    std::cout << "Common Status Retrieved Successfully." << std::endl;
    auto statue = com_state.belie_or_stand
    // 输出或处理 com_state 数据
} else {
    std::cout << "Failed to Retrieve Common Status." << std::endl;
}
```

状态码定义

```
typedef enum BODY_POSTURE_STATE_{
    NULL_POSTURE = 0,
    STARTMOVING = 1,
    MOVING = 2,
    LIEDOWN = 3,
    STANDUP = 4,
    CHARGING=5  //在桩上  可能是充电，也可能已经充满了
}BODY_POSTURE_STATE;
```

2.2. ROS2 接口

该接口用于发布机器人的基本状态信息。

Topic 名称	Topic 类型	角色
`/robot_state`	`std_msgs::msg::UInt32`	发布方

消息结构

`std_msgs::msg::UInt32` 消息类型包含以下字段：

- `0`：lie down
- `1`: standing up
- `2`：stand up
- `3`：enable force control
- `4`：stepping
- `5`：sitting down
- `6`：fall down

示例消息

```
std_msgs::msg::UInt32 state_msg;
state_msg.data = 2; // standup
```

测试方法

```
ros2 topic echo /robot_state
```

3. 获取控制模式

获取机器人控制模式信息：处于导航模式或者是手动模式。

3.1. C++ 接口

参考 【设备状态服务接口】-【获取常规状态信息】。

```
common::ROBOT_COMMON_STATUS com_state;
bool success = rcClientInterfaceGetCommonStatus(&com_state);
auto state = com_state.joy_mode
```

状态码如下：

```
typedef enum NAV_OR_JOY_MODE_{
    nall_control = 0,
    nav_control = 1,
    joy_control = 2
}NAV_OR_JOY_MODE;
```

3.2. ROS2 接口

该接口用于发布机器人当前的控制模式。

Topic 名称	Topic 类型	角色
`/control_mode`	`std_msgs::msg::UInt8`	发布方

消息结构

`std_msgs::msg::UInt8` 消息类型包含以下字段：

- `data`：表示机器人的控制模式，即状态码。
- **状态码描述：**
  - `0`：手持遥控
  - `1`：自动（导航）

示例消息

```
std_msgs::msg::UInt8 control_mode_msg;
control_mode_msg.data = 1; // 自动（导航）
```

测试方法

```
ros2 topic echo /control_mode
```

4. 获取机器人当前场景（步态）

4.1. C++ 接口

参考 【设备状态服务接口】-【获取常规状态信息】。

```
common::ROBOT_COMMON_STATUS com_state;
bool success = rcClientInterfaceGetCommonStatus(&com_state);
auto state = com_state.cur_scene
```

状态码如下：

```
typedef enum SCENE_TYPE_{
    NULL_SCENE = 0,     // 无
    LIE_DOWN = 1,       // 趴下
    WALKING = 2,        // 行走
    STAIRS = 3,         // 楼梯
    CHARGE = 4,         // 充电
    PERCEIVE_STAIRS = 5,//
    SNOW = 6,           // 雪地
    SLIPPY = 7,         // 滑坡
    STONE = 8           // 石子路
}SCENE_TYPE;
```

4.2. ROS2 接口

该接口用于发布机器人的当前步态。

Topic 名称	Topic 类型	角色
`/gait`	`std_msgs::msg::UInt8`	发布方

消息结构

`std_msgs::msg::UInt8` 消息类型包含以下字段：

- `data`：表示机器人的当前步态。步态码如下：

- `0`：正常步态
- `1`：楼梯步态
- `2`：斜坡步态

示例消息

```
std_msgs::msg::UInt8 gait_msg;
gait_msg.data = 2; // 行走
```

测试方法

```
ros2 topic echo /gait
```

5. 获取上桩状态

5.1. C++ 接口

参考 【设备状态服务接口】-【获取常规状态信息】。

```
common::ROBOT_COMMON_STATUS com_state;
bool success = rcClientInterfaceGetCommonStatus(&com_state);
auto state = com_state.charge_status.charge_switch_state
```

状态码如下：

```
typedef enum CHARGE_SWITCH_STATE_{
    NULL_CHARGE_SWITCH = 0,
    EXIT_SUCCEEDED = 1,
    EXITING_TILE = 2,
    EXIT_FAILED=3,
    ENTER_SUCCEEDED = 4,
    ENTERING_TILE = 5,
    ENTER_FAILED=6   //尝试上桩超过三次后进入该状态
}CHARGE_SWITCH_STATE;
```

6. 获取关节状态信息

6.1. C++ 接口

`rcClientInterfaceGetJointsStatus` 简介

```
/**
 * @brief 获取机器人关节上本体状态信息。
 *
 * 该函数用于获取机器人的关节状态信息。
 *
 * @param state 指向保存机器人关节状态信息的结构体指针。
 * @return bool 返回 `true` 表示获取成功，返回 `false` 表示获取失败。
 */
bool rcClientInterfaceGetJointsStatus(common::ROBOT_JOINTS_STATUS *state);
```

使用案例

```
common::ROBOT_JOINTS_STATUS state;
bool success = rcClientInterfaceGetJointsStatus(&state);
if (success) {
    std::cout << "Joints Status Retrieved Successfully." << std::endl;
    // 输出或处理 state 数据
} else {
    std::cout << "Failed to Retrieve Joints Status." << std::endl;
}
```

注意事项

- `state` 必须指向有效的 `common::ROBOT_JOINTS_STATUS` 结构体。