控制服务说明

修改历史

Version	Contributor	Date	Change Log
v1.0	彭钉	17/12/22	1.相关参数增加说明文档 2.增加红外接口API文档
v1.1	彭钉	17/1/10	1.增加清除舵机保护位接口
v1.2	彭钉	18/02/02	1.新增设置基准角度接口

概述

控制服务是ROSE中一个服务应用，安装控制服务相应要安装ROSE中Master应用，master应用的安装参看《master用户指导》。目前，在alpha系列机器人内包含一块控制板，机器人核心板通过串口连接控制板，核心板上的android应用使用一组约定的串口协议与控制板通讯，通过控制板控制舵机。控制服务的主要功能在于定义统一的控制板访问接口，包括：舵机控制，控制板升级，红外等其他杂项设置; 另外，为了让机器人执行一组复杂的动作，预置动作配置文件即ubx动作文件于机器人内，控制服务具有解析播放ubx文件功能,并对外提供接口。本文档简要说明这些接口及sdk集成。

舵机控制接口

```
//舵机信息protobuf
message Motor {
```

```
    int32 id= 1 ;// id 编号
    int32 upperLimitAngle = 2;//舵机转角上限，单位弧度
    int32 lowerLimitAngle = 3;//舵机转角下限，单位弧度
    int32 servoSener = 4; //舵机传感器类型0-电位器1-磁编码
    // 舵机电机类型  0-DC铁芯电机1- DC空心杯电机 2- BLDC 3-PMSM
    int32 servoMotor = 5;
    // 舵机MCU类型 0-MEGA8 1-ATSAMD10D14  2-ATSAMC21E15A
    int32 servoMcu = 6;
    int32 servoDriver = 7; // 舵机驱动芯片型号0-8838 1-6515 2-4959
    int32 servoMode = 8; // 舵机型号 单位KG
    int32 rigidity = 9; //刚性值
    int32 protectMode = 10; //保护模式
    int32 protectLevel = 11;//保护等级
    bool protectable = 12;//保护使能
    int32 offsetAngle = 13;//偏移角度
    string softwareVersion = 14;//软件版本
    string hardwareVersion = 15;//硬件版本
    int32 servoVolt = 16; //舵机工作电压= ServoVolt*3.7
    int32 resetAngle = 17;//复位角度
    int32 maxSpeed = 18;//最大转动速度
    int32 minSPeed = 19;//最小转动速度
}

//舵机运动参数
message MotorArgs {
    int32 id = 1;//舵机ID
    int32 angle = 2;//舵机新角度
    int32 runMode = 3;//运动模式:0:表示匀速，1：表示变速
    int32 runTime = 4;//运动时间
    bool interrupted = 5;//舵机运动时是否可接收下一条指令
}
// 辅助结构
message MotorAngle{
    int32 id = 1;//舵机ID
    int32 angle = 2;//舵机新角度
}
message ServoStatus {
```

```
    int32 id = 1;
    int32 status = 2;//取值如下：
    //0:舵机响应函数操作成功,
    //1:舵机未响应函数操作或响应超时;
    //2:舵机响应数据出错
    //3:舵机群组结构体读写函数未初始化或失败
    //4:舵机温度低保护
    //5:舵机过温保护
    //6:舵机低压保护
    //7:舵机过压保护
    //8:舵机过流保护
    //9:舵机力矩保护
    //10:舵机熔丝位错保护
    //11:舵机堵转保护
    //12:舵机硬件出错
    //13:升级失败
}



/**
 * 获取机器人舵机信息列表
 *
 * @param listener ResponseListener with List of Motion.Motor
 */
public void getMotorList(@NonNull final ResponseListener<List<Motion.Motor>> listener) ;

/**
 * 获取机器人舵机信息列表
 *
 * @return List of Motion.Motor 机器人上舵机信息列表
 */
public @Nullable List<Motion.Motor> getMotorList() ;

/**
 * 读取机器人舵机信息
 *
 * @param id 舵机id
```

```
     * @return Motion.Motor
     */
    public @Nullable Motion.Motor getMotor(int id) ;

    /**
     * 获取机器人上舵机个数
     *
     * @return int
     */
    public int getMotorSum() ;
    /**
     * 移动单个舵机到指定的绝对角度
     *
     * @param motorId 舵机id
     * @param angle 角度
     * @param duration 运行时长
     * @param listener ResponseListener
     */
    public void moveToAbsoluteAngle(@IntRange(from = 1, to = 14) final int motorId, final int angle,
        @IntRange(from = 0, to = 5000) final int duration,
        @Nullable ResponseListener<Void> listener) ;

    /**
     * 移动单个舵机到指定的绝对角度
     *
     * @param motorId 舵机id
     * @param angle 角度
     * @param duration 运行时长
     * @param priority 优先级
     * @param listener ResponseListener
     */
    public void moveToAbsoluteAngle(@IntRange(from = 1, to = 14) final int motorId, final int angle,
        @IntRange(from = 0, to = 5000) final int duration, Priority priority,
        @Nullable final ResponseListener<Void> listener) ;

    /**
     * 移动单个舵机到指定的绝对角度
```

```
     *
     * @param motorId 舵机id
     * @param angle 角度
     * @param duration 运行时长
     * @param runMode 运行模式
     * @param priority 优先级
     * @param listener ResponseListener
     */
    public void moveToAbsoluteAngle(@IntRange(from = 1, to = 14) final int motorId, final int angle,
        @IntRange(from = 0, to = 5000) final int duration, final RunMode runMode, Priority priority,
        @Nullable final ResponseListener<Void> listener);

    /**
     * 移动一组舵机到指定的绝对角度
     *
     * @param motorArgs 舵机参数
     * @param listener ResponseListener
     */
    public void moveToAbsoluteAngle(@NonNull List<Motion.MotorArg> motorArgs,
        @Nullable ResponseListener<Void> listener);

    /**
     * 移动一组舵机到指定的绝对角度
     *
     * @param motorArgs 舵机参数
     * @param listener 回调接口
     * @param priority 优先级
     */
    public void moveToAbsoluteAngle(@NonNull final List<Motion.MotorArg> motorArgs, Priority priority,
        @Nullable final ResponseListener<Void> listener);

    /**
     * 读取单个舵机当前角度
     *
     * @param motorId 舵机ID
     * @param adcump 是否掉电读取
     * @param listener 回调接口
```

```
     */
    public void readAbsoluteAngle(final int motorId, boolean adcump,
        @NonNull final ResponseListener<Integer> listener) ;

    /**
     * 读取一组舵机当前角度
     *
     * @param motorId 舵机id
     * @param adcump 是否掉电读取
     * @return int 大于0,舵机角度，等于-1,出错
     */
    public int readAbsoluteAngle(final int motorId, boolean adcump);

    /**
     * 读取一组舵机当前角度
     *
     * @param motorId 舵机id
     * @return int 大于0,舵机角度，等于-1,出错
     */
    public int readAbsoluteAngle(final int motorId) ;

    /**
     * 读取一组舵机的当前角度
     *
     * @param motorIds 舵机id
     * @param adcump 是否掉电读取
     * @param listener 回调接口
     */
    public void readAbsoluteAngle(@NonNull List<Integer> motorIds, boolean adcump,
        @NonNull final ResponseListener<List<Motion.MotorAngle>> listener);

    /**
     * 读取一组舵机的当前角度
     *
     * @param motorIds 一组舵机ID
     * @param adcump 是否掉电读取
     * @return List of Motion.MotorAngle
```

```
     */
    public @Nullable List<Motion.MotorAngle> readAbsoluteAngle(@NonNull List<Integer> motorIds,
        boolean adcump) ;

    /**
     * 读取一组舵机的当前角度
     *
     * @param motorIds 一组舵机ID
     * @return List of Motion.MotorAngle
     */
    public @Nullable List<Motion.MotorAngle> readAbsoluteAngle(@NonNull List<Integer> motorIds) ;

    /**
     * 读取一组舵机的当前角度
     *
     * @param motorIds 一组舵机ID
     * @param adcump 是否掉电读取
     * @return List of Motion.MotorAngle
     */
    public @Nullable List<Motion.MotorAngle> readAbsoluteAngle(boolean adcump, int... motorIds);

    /**
     * 不掉电回读取一组舵机角度
     *
     * @param motorIds 一组舵机ID
     * @return List of Motion.MotorAngle
     */
    public @Nullable List<Motion.MotorAngle> readAbsoluteAngle(int... motorIds) ;

    /**
     * 读取舵机调整角度
     *
     * @param motorId 舵机id
     * @param listener 回调接口
     */
    public void readOffsetAngle(final int motorId,
        @NonNull final ResponseListener<Integer> listener) ;
```

```
/**
 * 读取舵机调整角度，正负都有可能
 *
 * @param motorId 舵机ID
 * @return int Integer.MAX_VALUE：读取失败
 */
public int readOffsetAngle(final int motorId) ;

/**
 * 读取舵机调整角度
 *
 * @param motorIds 一组舵机id
 * @return List of Motion.MotorAngle
 */
public @Nullable List<Motion.MotorAngle> readOffsetAngle(int... motorIds);

/**
 * 读取舵机调整角度
 *
 * @param motorIds 一组舵机id
 * @return List of Motion.MotorAngle
 */
public @Nullable List<Motion.MotorAngle> readOffsetAngle(@NonNull List<Integer> motorIds) ;

/**
 * 读取舵机调整角度
 *
 * @param motorIds 一组舵机id
 * @param listener 回调接口
 */
public void readOffsetAngle(@NonNull List<Integer> motorIds,
    @NonNull final ResponseListener<List<Motion.MotorAngle>> listener);

/**
 * 设置单个舵机校准角度
 *
```

```
     * @param motorId 舵机id
     * @param offsetAngle 校准角度
     * @return boolean 是否成功
     */
    public boolean setOffsetAngle(final int motorId, final int offsetAngle);

    /**
     * 设置一组舵机校准角度
     *
     * @param motorAngles 一组舵机
     * @return boolean 是否成功
     */
    public boolean setOffsetAngle(Motion.MotorAngle... motorAngles) ;

    /**
     * 设置一组舵机校准角度
     *
     * @param motorAngles 一组舵机
     * @return boolean 是否成功
     */
    public List<Motion.ServoStatus> setOffsetAngle(@NonNull List<Motion.MotorAngle> motorAngles);

    /**
     * 设置单个舵机校准角度
     *
     * @param motorId 舵机id
     * @param adjustAngle 校准角度
     * @param listener 回调接口
     */
    public void setOffsetAngle(final int motorId, final int adjustAngle,
        @Nullable final ResponseListener<List<Motion.ServoStatus>> listener);

    /**
     * 设置一组舵机校准角度
     *
     * @param motorAngles 一组舵机角度
     * @param listener 回调接口
```

```
     */
    public void setOffsetAngle(@NonNull List<Motion.MotorAngle> motorAngles,
        @Nullable final ResponseListener<List<Motion.ServoStatus>> listener) ;

    /**
     * 设置单个舵机基准角度
     *
     * @param motorId 舵机id
     * @param BasicAngle 基准角度
     * @param servoSener 舵机类型
     * @return boolean 是否成功
     */
    public boolean setBasicAngle(final int motorId, final int BasicAngle,
        final ServoSener servoSener) ;

    /**
     * 设置一组舵机基准角度
     *
     * @param motorAngles 一组舵机
     * @return List of Motion.ServoStatus
     */
    public @Nullable List<Motion.ServoStatus> setBasicAngle(Motion.MotorAngle... motorAngles);

    /**
     * 设置一组舵机基准角度
     *
     * @param motorAngles 一组舵机
     * @return List of Motion.ServoStatus
     */
    public @Nullable List<Motion.ServoStatus> setBasicAngle(
        @NonNull List<Motion.MotorAngle> motorAngles);

    /**
     * 设置单个舵机基准角度
     *
     * @param motorId 舵机id
     * @param basicAngle 基准角度
```

```
     * @param servoSener 舵机类型：ServoSener
     * @param listener 回调接口
     */
    public void setBasicAngle(final int motorId, final int basicAngle, final ServoSener servoSener,
        @Nullable final ResponseListener<Boolean> listener) ;

    /**
     * 设置一组舵机基准角度
     *
     * @param motorAngles 一组舵机角度
     * @param listener 回调接口
     */
    public void setBasicAngle(@NonNull List<Motion.MotorAngle> motorAngles,
        @Nullable final ResponseListener<List<Motion.ServoStatus>> listener);

    /**
     * 锁紧舵机
     *
     * @param listener listener or lock motor
     */
    public void lockAllMotor(@Nullable ResponseListener<Boolean> listener);

    /**
     * 锁紧舵机： 如果舵机在运动状态，舵机立即锁住，如果舵机在掉电状态，舵机上电紧锁
     *
     * @param motorId 单个舵机id
     * @param listener 回调接口
     */
    public void lockMotor(final int motorId, @Nullable final ResponseListener<Boolean> listener);

    /**
     * 锁紧舵机
     *
     * @param motorIds 一组舵机id
     * @param listener 回调接口
     * @param priority 优先级
     */
```

```
public void lockMotor(@NonNull final List<Integer> motorIds, Priority priority,
    @Nullable final ResponseListener<Boolean> listener);

/**
 * 所有舵机掉电
 *
 * @param listener listener of unlock motor
 */
public void unlockAllMotor(@Nullable ResponseListener<Boolean> listener) ;

/**
 * 舵机松弛
 *
 * @param motorId 舵机id
 * @param listener ResponseListener
 */
public void unlockMotor(final int motorId, final @Nullable ResponseListener<Boolean> listener) ;

/**
 * 舵机松弛：如果舵机在运动状态，舵机会先停止运动，再掉电；如果舵机在停止紧锁状态，舵机掉电
 *
 * @param motorIds 一组舵机id
 * @param priority 优先级
 * @param listener ResponseListener
 */
public void unlockMotor(@NonNull final List<Integer> motorIds, Priority priority,
    @Nullable final ResponseListener<Boolean> listener) ;

/**
 * 单个舵机是否工作正常
 *
 * @param motorId 舵机id
 * @return boolean 正常工作
 */
public boolean checkServoStatus(final int motorId) ;

/**
```

```
     * 检查一组舵机是否正常
     *
     * @param motorIds 一组舵机id
     * @return List of Motion.ServoStatus
     */
    public @Nullable List<Motion.ServoStatus> checkServoStatus(@NonNull List<Integer> motorIds) ;

    /**
     * 检查一组舵机是否正常
     *
     * @param motorIds 一组舵机id
     * @return List of Motion.ServoStatus
     */
    public @Nullable List<Motion.ServoStatus> checkServoStatus(int... motorIds) ;

    /**
     * 清除舵机保护位
     *
     * @param motorIds 一组舵机id
     */
    public boolean clearProtectFlag(@NonNull List<Integer> motorIds) ;

    /**
     * 清除舵机保护位
     *
     * @param motorIds 一组舵机id
     */
    public boolean clearProtectFlag(int... motorIds);

    /**
     * 舵机复位
     *
     * @param priority priority
     * @param listener 是否成功
     */
    public void reset(Priority priority, @Nullable final ResponseListener<Boolean> listener);
```

```
/**
 * 关闭舵机电源
 */
public void powerOff() ;

/**
 * 打开舵机电源
 *
 * @param listener listener of call
 */
public void powerOn(@Nullable final ResponseListener<Boolean> listener) ;

/**
 * 打开舵机电源, 同步调用, 不要在主线程中使用
 */
public void powerOn();
/**
 * 舵机复位
 *
 * @param listener 是否成功
 */
public void reset(@Nullable final ResponseListener<Boolean> listener) ;

/**
 * motor 硬件异常 订阅
 *
 * @param receiver 监听
 */
public void subscribeMotorErrorEvent(MotorErrorReceiver receiver) ;

/**
 * motor 状态 订阅
 *
 * @param receiver 监听
 */
public void subscribeMotorStatusEvent(MotorStatusReceiver receiver) ;
```

```
/**
 * 取消订阅 motor 硬件异常
 *
 * @param receiver 监听
 */
public void unsubscribeMotorErrorEvent(MotorErrorReceiver receiver) ;

/**
 * 取消订阅 motor 状态
 *
 * @param receiver 监听
 */
public void unsubscribeMotorStatusEvent(MotorStatusReceiver receiver) ;
```

升级接口

- 说明： 升级接口分两类，控制板升级和舵机升级，如果两者都需升级，先升胸板，再升舵机。

```
/**
* 开始胸板升级
*
* @param filesize 总文件大小
* @param mode     0：Boot, 1:App --控制板分两个程序，Boot是引导程序;App是实现一些功能的程序
* @param listener 回调
*/
public void beginCtrlUpgrade(@IntRange(from = 0, to = 1) int mode, int filesize, @NonNull  ResponseListener<Void> listener);

/**
* 写入胸板升级数据包
*
* @param pageNum  当前页号
* @param pageData 分页数据
```

```
     * @param listener 回调
     */
    public void writeCtrlUpgradeData(int pageNum, @NonNull byte[] pageData, @NonNull  ResponseListener<Void> listener);

    /**
     * 结束胸板升级
     *
     * @param md5      校验
     * @param listener 回调接口
     */
    public void endCtrlUpgrade(@NonNull byte[] md5, @NonNull  ResponseListener<Void> listener);

    /**
     * 开始舵机升级
     *
     * @param ids      批量升级的舵机ID
     * @param fileSize 文件大小
     * @param listener 回调接口
     */
    public void beginMotorUpgrade(@NonNull List<Integer> ids, int fileSize, @NonNull  ResponseListener<Void> listener);

    /**
     * 写入舵机升级包
     *
     * @param motorId  舵机id
     * @param islast   是否最后一个包
     * @param pageNum  分页号
     * @param pageData 分页数据
     * @param listener 回调
     */
    public void writeMotorUpgradeData(int motorId, boolean islast, int pageNum, @NonNull byte[] pageData, @NonNull  ResponseListener<Void> list
```

设置接口

```
/**
 * 读取机器人序列号
 *
 * @param listener ResponseListener 回调接口
 */
public void readRobotSid(@NonNull final ResponseListener<String> listener) ;

/**
 * 读取机器人序列号, 默认从Provider读取
 *
 * @return String 序列号
 */
public String readRobotSid() ;

/**
 * 写入机器人序列号
 *
 * @param sid String 序列号
 * @param listener ResponseListener 回调接口
 */
public void writeRobotSid(@NonNull String sid, @Nullable final ResponseListener<Void> listener) ;

/**
 * 写入机器人序列号
 *
 * @param sid String 序列号
 * @return boolean 是否成功
 */
public boolean writeRobotSid(@NonNull String sid);

/**
 * 读取胸板软件版本
 *
 * @param listener ResponseListener 回调接口
 */
public void readCtrlVersion(@NonNull final ResponseListener<String> listener) ;
```

```
/**
 * 读取胸板软件版本
 *
 * @return String 胸板软件版本
 */
public String readCtrlVersion() ;

/**
 * 读取App胸板软件版本
 *
 * @return String 胸板App软件版本
 */
public String readAppVersion();

/**
 * 读取OTA升级固件版本
 *
 * @param listener ResponseListener 回调接口
 */
public void readFirmwareVersion(@NonNull final ResponseListener<String> listener) ;

/**
 * 读取OTA升级固件版本
 *
 * @return String 读取OTA升级固件版本
 */
public String readFirmwareVersion() ;

/**
 * 设置OTA升级固件版本
 *
 * @param version firmware version
 * @param listener ResponseListener 回调接口
 */
public void setFirmwareVersion(@NonNull String version,
    @NonNull final ResponseListener<Boolean> listener);
```

```
/**
 * 设置OTA升级固件版本
 *
 * @param version ota version
 * @return boolean
 */
public boolean setFirmwareVersion(@NonNull String version) ;

/**
 * 关机胸口板系统
 */
public void shutdown() {
  sys.call("/shutdown", new ResponseCallback() ;

/**
 * 启动胸口板系统: 启动胸口板至少需要有600毫秒的时间
 *
 * @param listener {@link ResponseListener}
 */
public void startup(@NonNull final ResponseListener<Void> listener) ;

/**
 * 判断胸板系统是否启动
 *
 * @return boolean
 */
public boolean isStarted() ;

/**
 * subsystem error 订阅
 *
 * @param receiver 监听
 */
public void subscribeSubsystemErrorEvent(SubsystemErrorReceiver receiver) ;
/**
 * 取消订阅 subsystem error
```

```
     *
     * @param receiver 监听
     */
    public void unsubscribeSubsystemErrorEvent(SubsystemErrorReceiver receiver) ;
```

动作接口

```
//动作文件信息
message Action {
    string id = 1; //动作id
    string cnName = 2; //中文名
    string enName = 3; //英文名
    string desc = 4; 描述
    int32 time = 5; //时长
    int32 type = 6;
    bool unsafeAction = 7; // 高危动作
}
```

```
/**
     * 异步获取系统内置的动作文件列表
     *
     * @param listener ResponseListener
     */
    public void getActionList(@NonNull final ResponseListener<List<Motion.Action>> listener);

    /**
     * 获取系统内置的支持的动作文件列表
     *
     * @return list of Motion.Action
     */
    public @Nullable List<Motion.Action> getActionList();
```

```
/**
 * 获取机器人上内置到sdcard/customize/actions的自定义动作文件列表
 *
 * @return list of {@link Motion.Action}
 */
public List<Motion.Action> getCustomizeActionList();

/**
 * 获取系统内置的某个动作信息
 *
 * @param actionId action id
 * @return {@link Motion.Action}
 */
public @Nullable Motion.Action getActionInfo(String actionId);

/**
 * 获取/sdcard/sdcard/customize目录下的某个动作信息
 *
 * @param actionId action id
 * @return {@link Motion.Action}
 */
public @Nullable Motion.Action getCustomizeActionInfo(String actionId);

/**
 * 播放一个系统内置动作, 优先级， 默认0
 *
 * @param action String
 * @param listener ResponseListener
 */
public void playAction(@NonNull final String action,
    @Nullable final ResponseListener<Void> listener);

/**
 * 播放一个系统内置动作
 *
 * @param action String
```

```
     * @param priority 优先级
     * @param listener ResponseListener
     */
    public void playAction(@NonNull final String action, Priority priority,
        @Nullable final ResponseListener<Void> listener);

    /**
     * 播放一个内置到sdcard/customize/actions中的自定义动作
     *
     * @param actionId action Id
     * @param listener ResponseListener
     */
    public void playCustomizeAction(@NonNull final String actionId,
        @Nullable final ResponseListener<Void> listener);

    /**
     * 播放一个内置到/sdcard/customize/actions中的自定义动作
     *
     * @param actionId action Id
     * @param priority priority
     * @param listener ResponseListener
     */
    public void playCustomizeAction(@NonNull final String actionId, Priority priority,
        @Nullable final ResponseListener<Void> listener);

    /**
     * 异步停止当前播放的动作
     *
     * @param listener ResponseListener
     */
    public void stopAction(@Nullable final ResponseListener<Void> listener);

    /**
     * 同步停止当前播放的动作
     *
     * @return boolean
     */
```

```
public boolean stopAction();

/**
 * 停止指定名字的动作
 *
 * @param name action anme
 */
public void stopActionByName(@NonNull String name) ;

/**
 * 异步调用判断机器人是否正在做动作
 *
 * @param listener ResponseListener
 */
public void isPlaying(@NonNull final ResponseListener<Boolean> listener);

/**
 * 同步调用判断机器人是否正在做动作
 *
 * @return boolean
 */
public boolean isPlaying();

/**
 * 判断动作是否高位
 *
 * @param actionId action id
 * @return boolean
 */
public boolean unsafeAction(@NonNull String actionId);

/**
 * 判断正在执行的动作是否存在危险动作，如果没有动作在播放，返回false
 *
 * @return boolean
 */
public boolean unsafeAction() ;
```

```
/**
 * 获取此刻正在执行的动作信息, 如果当前没有执行动作,返回 null
 *
 * @return Motion.Action of list
 */
public @Nullable List<Motion.Action> currentAction() ;

/**
 * action finished 订阅
 *
 * @param receiver 状态监听
 */
public void subscribeEvent(ActionStoppedReceiver receiver);

/**
 * 取消订阅
 *
 * @param receiver 状态监听
 */
public void unsubscribeEvent(EventReceiver receiver) ;

/**
 * 动作播放错误码
 */
public class ActionErrorCode {
    public static final int INTERRUPTED_BY_STOP = 10001;//动作由于stop接口调用中断
    public static final int INTERRUPTED_BY_BROKEN = 10002;//动作由于舵机硬件损坏被中断
    public static final int INTERRUPTED_BY_PROTECTED = 10003;//动作由于舵机保护被中断
    public static final int INTERRUPTED_BY_ACTION = 10004;//动作由于新动作执行而被打断
    public static final int FILE_NOT_FOUNDED = 10005;//文件未找到
    public static final int DATABASE_EXCEPTION = 10006;//数据库异常
    public static final int UBX_PARSE_ERROR = 10008;//动作文件解析失败
    public static final int UBX_PLAY_ERROR = 10009;//动作文件解析失败
    public static final int INTERRUPTED_BY_STOP_SELF = 10010;//动作由于stopActionByName调用中断
```

```
    }
```

红外接口

```
/**
 * 红外校准--
 *
 * @param mode 校准模式, 取值0, 1
 * @return boolean
 */
public boolean adjust(@IntRange(from = 0, to = 1) int mode);

/**
 * 红外探测距离结果: 50-1500毫米，如果-1表示失败
 *
 * @return int
 */
public int distance();

/**
 * 红外电源powerOn
 *
 * @return boolean
 */
public boolean powerOn();

/**
 * 红外电源powerOff
 *
 * @return boolean
 */
public boolean powerOff();
```