**云台开发手册**

# 初始化

## 创建ptz客户端对象

### 函数

* /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 函数声明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\* 函数名称: New \*\* 功能说明: 创建ptz客户端对象，返回指针对象。 \*\* 入参 : 无 \*\* 出参 : \*PtzClient \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/// New .func New() (ptr \*PtzClient) {}

## 获取ptz客户端实例

### 函数

* /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 函数声明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\* 函数名称: GetIns \*\* 功能说明: 获取ptz客户端实例 \*\* 入参 : 无 \*\* 出参 : \*PtzClient \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/// GetIns .func GetIns() \*PtzClient {}

## ptz客户端初始化

### 函数

* /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 函数声明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\* 函数名称: Init \*\* 功能说明: ptz客户端初始化，初始化指定串口，实例赋值，设置默认配置，注册配置回调函数。 \*\* 入参 : 无 \*\* 出参 : bool \*\* 返回 : true:成功; false:不成功 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/// Init .func (client \*PtzClient) Init() bool {}

## ptz客户端开始

### 函数

* /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 函数声明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\* 函数名称: Start \*\* 功能说明: ptz客户端开始，获取当前配置，设置串口属性。 \*\* 入参 : 无 \*\* 出参 : bool \*\* 返回 : true:成功; false:不成功 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///Start .func (client \*PtzClient) Start() bool {}

## ptz客户端停止

### 函数

* /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 函数声明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\* 函数名称: Stop \*\* 功能说明: ptz客户端停止，销毁串口。 \*\* 入参 : 无 \*\* 出参 : 无 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///Stop .func (cli \*PtzClient) Stop() {}

## 配置回调函数

### 函数

* /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 函数声明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\* 函数名称: OnApplyConfig \*\* 功能说明: 配置回调函数：协议判断，设置串口属性。 \*\* 入参 : (val interface{}) \*\* 出参 : (bool, int32) \*\* 返回 : 成功：true，0; 失败：true，1-需要重启系统,2-将自动重启系统,4-验证失败,8-回调执行失败 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///OnApplyConfig 配置回调函数func (client \*PtzClient) OnApplyConfig(val interface{}) (bool, int32) {}

## 485串口发送命令

### 函数

* /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 函数声明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\* 函数名称: SendCommand \*\* 功能说明: 485串口发送命令，写入数据。 \*\* 入参 : (input \*dsd.InputParam)，结构体指针 \*\* 出参 : bool \*\* 返回 : true:成功;false：失败 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/// SendPTZCommand 发送指令func (client \*PtzClient) SendCommand(input \*dsd.InputParam) bool {}

# 串口

## 串口类型

### 函数

* const ( TypeNone = iota // 无, 防止创建时未传递类型 TypeRs232 // 通用串口 TypeRs485 // 通用485 TypeRs422 // 通用422 TypePtz // 云台控制485 TypeMax // 最大类型数)

## 串口属性

### 函数

* //CommAttribute 串口属性type CommAttribute struct { BodeRate uint32 `description:"波特率"` DataBits uint8 `description:"数据位数"` Parity uint8 `description:"奇偶校验选项,0无,1奇校验,2偶校验,3标志校验,4空校验"` StopBits uint8 `description:"停止位,0:1,1:1.5,2:2"`}

## 初始化指定串口

### 函数

* /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 函数声明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\* 函数名称: InitComm \*\* 功能说明: 初始化指定串口 \*\* 入参 : (commtype uint8)，串口类型 \*\* 出参 : bool \*\* 返回 : true:成功;false：失败 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///InitComm 初始化指定串口func InitComm(commtype uint8) bool {}

## 设置串口属性

### 函数

* /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 函数声明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\* 函数名称: SetCommAttr \*\* 功能说明: 设置串口属性 \*\* 入参 : 1.commtype:串口类型 \*\* : 2.CommAttribute:串口属性 \*\* 出参 : bool \*\* 返回 : true:成功;false：失败 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///SetCommAttr 设置串口属性func SetCommAttr(commtype uint8, attr CommAttribute) bool {}

## 销毁串口

### 函数

* /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 函数声明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\* 函数名称: DestroyComm \*\* 功能说明: 销毁串口 \*\* 入参 : commtype uint8:串口类型 \*\* 出参 : bool \*\* 返回 : true:成功;false：失败 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/// DestroyComm 销毁串口func DestroyComm(commtype uint8) bool {}

## 读取数据

### 函数

* /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 函数声明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\* 函数名称: ReadComm \*\* 功能说明: 读取数据 \*\* 入参 : 1.commtype:串口类型 \*\* : 2.len:命令长度 \*\* 出参 : 1.[]byte:返回的字节数组 \*\* : 2.bool:返回布尔值 \*\* 返回 : true:成功;false：失败 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/// ReadComm 读取数据func ReadComm(commtype uint8, len int) ([]byte, bool) {}

## 写入数据

### 函数

* /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 函数声明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\* 函数名称: WriteComm \*\* 功能说明: 写入数据 \*\* 入参 : 1.commtype:串口类型 \*\* : 2.data:数组字节 \*\* 出参 : bool \*\* 返回 : true:成功;false：失败 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/// WriteComm 写入数据func WriteComm(commtype uint8, data []byte) bool {}

## 清串口设备缓存

### 函数

* /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 函数声明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\* 函数名称: FlushComm \*\* 功能说明: 清串口设备缓存 \*\* 入参 : 1.commtype:串口类型 \*\* : 2.queueSec:清空队列，0:清空输入队列, 1: 清空输出队列, 2:清空输入输出队列。 \*\* 出参 : bool \*\* 返回 : true:成功;false：失败 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/// FlushComm 清串口设备缓存func FlushComm(commtype uint8, queueSec int) bool {}

# 云台控制

## 云台转动

### 函数

* // @Title 云台转动// @Summary 云台转动// @Description 云台转动// @Param method query int true "转动方向: 0上,1下,2左,3右,4左上,5右上,6左下,7右下"// @Param speed query uint8 true "转动速度"// @Success 200// @Failure 400000 操作失败// @router /turn [put]func (c \*Controller) Turn() {}

### PelcoD命令集合

* var ( PelcoMap = make(map[int][2]byte) //PelcoDMap pelco-d中部分指令集合 PelcoDMap = map[int][2]byte{ TurnUp: {0x00, 0x08}, TurnDown: {0x00, 0x10}, TurnLeft: {0x00, 0x04}, TurnRight: {0x00, 0x02}, TurnUpLeft: {0x00, 0x0c}, TurnUpRight: {0x00, 0x0a}, TurnDownLeft: {0x00, 0x14}, TurnDownRight: {0x00, 0x12},)

### PelcoP命令集合

* var (   //PelcoPMap pelco-p中部分指令集合 PelcoPMap = map[int][2]byte{ TurnUp: {0x00, 0x08}, TurnDown: {0x00, 0x10}, TurnLeft: {0x00, 0x04}, TurnRight: {0x00, 0x02}, TurnUpLeft: {0x00, 0x0c}, TurnUpRight: {0x00, 0x0a}, TurnDownLeft: {0x00, 0x14}, TurnDownRight: {0x00, 0x12},)

## 操作云台

### 函数

* // @Title 操作云台// @Summary 操作云台// @Description 改变云台光圈大小,焦距长短,聚焦远近或停止// @Param method query int true "执行方法: 8焦距变短,9焦距变长,10聚焦变近,11聚焦变远,12光圈变小,13光圈变大"// @Success 200// @Failure 400000 操作失败// @router /operation [put]func (c \*Controller) Operate() {}

### PelcoD命令集合

* FocalShorter //变倍短---8 FocalLonger //变倍长 FocusCloser //聚焦短 FocusFarther //聚焦长 ApertureSmaller //光圈小 ApertureLarger //光圈大---13

### PelcoP命令集合

* FocalShorter: {0x00, 0x20}, FocalLonger: {0x00, 0x40}, FocusCloser: {0x00, 0x80}, FocusFarther: {0x01, 0x00}, ApertureSmaller: {0x02, 0x00}, ApertureLarger: {0x04, 0x00},

## 辅助功能

### 函数

* // @Title 辅助命令操作// @Summary 辅助命令操作// @Description 辅助命令操作// @Param id path uint8 true "辅助命令编号"// @Param method query string true "执行方法: 18设置辅助命令,19清空辅助命令"// @Success 200// @Failure 400000 操作失败// @router /auxiliary [put]func (c \*Controller) AuxiliaryOpt() {

### PelcoD命令集合

* SetAuxiliary: {0x00, 0x09}, ClearAuxiliary: {0x00, 0x0b},

## 回到水平零点

### 函数

* // @Title 回到水平零点// @Summary 回到水平零点// @Description 回到水平零点，水平零位检测 FF 01 00 07 00 22 sum// @Success 200// @Failure 400000 操作失败// @router /zero [put]func (c \*Controller) GoToZeroPan() {}

## 快速定位（3D定位）

### 函数

* // @Title 快速定位（3D定位）// @Summary 快速定位（3D定位）// @Description 快速定位（3D定位），自动聚焦和光圈// @Param panPosition query float32 true "水平坐标"// @Param tiltPosition query float32 true "垂直坐标"// @Success 200// @Failure 400000 操作失败// @router /moveDirectly [post]func (c \*Controller) MoveDirectly() {}

### 结构体

* // 快速定位（3D定位）type RapidPositioning struct { PanPosition float32 `description:"水平坐标"` TiltPosition float32 `description:"垂直坐标"` Zoom int8 `description:"变倍固定值，自动聚焦和光圈"`}

# 预置点

## 添加预置点

### 函数

* // @Title 增加预设点// @Summary 增加预设点// @Description 增加预设点// @Param id path uint8 true "预设点id"// @Param name query string true "预设点名称"// @Success 200// @Failure 400000 操作失败// @router /point/id/:id [post]func (c \*Controller) AddPoint() {}

## 修改预置点

### 函数

* // @Title 修改预设点// @Summary 修改预设点// @Description 修改预设点// @Param id query uint8 true "预设点id"// @Param name query string true "预设点新名称"// @Success 200// @Failure 400000 操作失败// @router /point/id/:id [put]func (c \*Controller) ModifyPoint() {}

## 获取所有预置点

### 函数

* // @Title 获取所有预设点// @Summary 获取所有预设点// @Description 获取所有预设点// @Success 200 {object} []conf.PresetPoint// @router /points [get]func (c \*Controller) GetPoint() {}

## 删除预置点

### 函数

* // @Title 删除预设点// @Summary 删除预设点// @Description 删除预设点// @Param id path []uint8 true "预设点id"// @Success 200// @Failure 400000 操作失败// @router /point [delete]func (c \*Controller) DeletePoint() {}

## 转至预置点

### 函数

* // @Title 转至预设点// @Summary 转至预设点// @Description 转至预设点// @Param id path uint8 true "预设点id"// @Success 200// @Failure 400000 操作失败// @router /point/id/:id/turn [put]func (c \*Controller) TurnToPoint() {}

## 清空预置点

### 函数

* // @Title 删除全部预设点// @Summary 删除全部预设点// @Description 删除全部预设点// @Param id path []uint8 true "全部预设点id"// @Success 200// @Failure 400000 操作失败// @router /deleteAllPoint [delete]func (c \*Controller) DeleteAllPoint(){}

## 结构体

### type PresetPoints []PresetPointtype PresetPoint struct { ID uint8 `description:"预设点id"` Name string `description:"预设点名称"`}

# 巡航组

## 添加巡航组和预置点（保存）

### 结构体

* // 3.巡航组预置点type CruisePreset struct { ID uint `description:"ID"` Name string `description:"名字"` Enable bool `description:"使能"` Preset conf.PresetPoint `description:"关联预置点"` ResidenceTime uint `description:"停留时间（s）"`}

### 函数

* // @Title 添加巡航组和预置点（保存）// @Summary 添加巡航组和预置点（保存）// @Description 添加巡航组和预置点（保存），接收前端传过来的JSON// @Param cruisePreset body dsd.CruisePreset true "水平坐标"// @Success 200// @Failure 400000 操作失败// @router /addCruisePreset [post]func (c \*Controller) AddCruisePreset() {}

## 删除巡航组

### 函数

* // @Title 删除巡航组和预置点// @Summary 删除巡航组和预置点// @Description 删除巡航组和预置点// @Param id path uint8 true "删除巡航组id"// @Success 200// @Failure 400000 操作失败// @router /deleteCruise [delete]func (c \*Controller) DeleteCruise() {}

## 修改巡航名称

### 函数

* // @Title 修改巡航组和预置点// @Summary 修改巡航组和预置点// @Description 修改巡航组和预置点// @Param id path uint8 true "修改巡航组id"// @Param name path string true "修改巡航组名称"// @Success 200// @Failure 400000 操作失败// @router /modifyCruise [put]func (c \*Controller) ModifyCruise() {}

## 查询巡航组和预置点（刷新）

### 函数

* // @Title 查询巡航组和预置点// @Summary 查询巡航组和预置点// @Description 查询巡航组和预置点// @Success 200// @Failure 400000 操作失败// @router /getCruise [get]func (c \*Controller) GetCruise() {}

## 开始

### 函数

* // @Title 开启自动巡航// @Summary 开启自动巡航// @Description 开启自动巡航// @Success 200// @router /startTour [post]func (c \*Controller) StartTour() {}

## 停止

### 函数

* // @Title 关闭自动巡航// @Summary 关闭自动巡航// @Description 关闭自动巡航// @Success 200// @router /stopTour [put]func (c \*Controller) StopTour() {}

# 线性扫描

## 设置左右边界

### 结构体

* // 4.线性扫描type LinearSweep struct { ID uint `description:"ID"` Speed uint `description:"线扫速度"` ResidenceTimeLeft uint `description:"停留时间（左）"` ResidenceTimeRight uint `description:"停留时间（右）"` LeftMargin float64 `description:"设置左边界"` RightMargin float64 `description:"设置右边界"`}

### 函数

* // @Title 添加线性扫描// @Summary 添加线性扫描// @Description 添加线性扫描，接收前端传过来的JSON，先设置左右边界，根据使能判断是否线扫开始// @Param cruise body dsd.LinearSweep true "水平坐标"// @Success 200// @Failure 400000 操作失败// @router /addLineSweep [post]func (c \*Controller) AddLineSweep() {}

## 线扫开始

### 函数

* // @Title 线扫开始// @Summary 线扫开始// @Description 线扫开始// @Param id path uint8 true "线扫开始id"// @Success 200// @Failure 400000 操作失败// @router /startlineSweep [post]func (c \*Controller) StartLineSweep() {}

## 线扫停止

### 函数

* // @Title 线扫停止// @Summary 线扫停止// @Description 线扫停止// @Param id path uint8 true "线扫停止id"// @Success 200// @Failure 400000 操作失败// @router /stopLineSweep [post]func (c \*Controller) StopLineSweep() {}

# 巡迹

## 巡迹开始记录

### 结构体

* // 5.巡迹type Track struct { Enable bool `description:"使能"` ID uint8 `description:"ID"`}

### 函数

* // @Title 巡迹开始记录// @Summary 巡迹开始记录// @Description 巡迹开始记录// @Param id path uint8 true "巡迹开始记录id"// @Success 200// @Failure 400000 操作失败// @router /startPatrolRecord [post]func (c \*Controller) StartPatrolRecord() {}

## 巡迹停止记录

### 函数

* // @Title 巡迹停止记录// @Summary 巡迹停止记录// @Description 巡迹停止记录// @Param id path uint8 true "巡迹停止记录id"// @Success 200// @Failure 400000 操作失败// @router /stopPatrolRecord [post]func (c \*Controller) StopPatrolRecord() {}

## 开始巡迹

### 函数

* // @Title 巡迹开始// @Summary 巡迹开始// @Description 巡迹开始// @Param id path uint8 true "巡迹开始id"// @Success 200// @Failure 400000 操作失败// @router /startPatrol [post]func (c \*Controller) StartPatrol() {}

## 停止巡迹

### 函数

* // @Title 巡迹停止// @Summary 巡迹停止// @Description 巡迹停止// @Param id path uint8 true "巡迹停止id"// @Success 200// @Failure 400000 操作失败// @router /stopPatrol [post]func (c \*Controller) StopPatrol() {}

# 水平旋转

## 开始旋转

### 函数

* // @Title 水平旋转开始// @Summary 水平旋转开始// @Description 水平旋转开始// @Param speed path uint8 true “速度"// @Param direction path uint8 true “方向：0 顺时针，1 逆时针"// @Success 200// @Failure 400000 操作失败// @router /startMove [post]func (c \*Controller) StartMove() {}

## 停止旋转

### 函数

* // @Title 水平旋转停止// @Summary 水平旋转停止// @Description 水平旋转停止，调用FF 0B 00 00 00 00 0B// @Success 200// @Failure 400000 操作失败// @router /stopMove [post]func (c \*Controller) StopMove() {}

# 空闲动作

## 确定

### 结构体

* // 预置点、巡航组、线性扫描、巡迹动作type Action struct { Preset conf.PresetPoint `description:"关联预置点"` Cruise CruisePreset `description:"关联巡航组"` LineScan LinearSweep `description:"关联线性扫描"` PatrolScan Patrol `description:"关联巡迹"`}// 7.空闲动作type IdleMotion struct { Enable bool `description:"使能"` Action PTZAction`description:"空闲动作"` FreeTime uint8 `description:"空闲时间 1~60分钟"`}

### 函数

* // @Title 添加空闲动作// @Summary 添加空闲动作// @Description 添加空闲动作,接收前端传过来的JSON// @Param cruise body dsd.IdleMotion true "空闲动作JSON"// @Success 200// @Failure 400000 操作失败// @router /idleMotion [post]func (c \*Controller) IdleMotion() {}

## 刷新

### 函数

* // @Title 刷新空闲动作// @Summary 刷新空闲动作// @Description 刷新空闲动作,给前端返回JSON，前端做判断// @Success 200// @Failure 400000 操作失败// @router /refreshIdleMotion [get]func (c \*Controller) RefreshIdleMotion() {}

# 开机动作

## 添加开机动作

### 结构体

* // 预置点、巡航组、线性扫描、巡迹动作type Action struct { Preset conf.PresetPoint `description:"关联预置点"` Cruise CruisePreset `description:"关联巡航组"` LineScan LinearSweep `description:"关联线性扫描"` PatrolScan Patrol `description:"关联巡迹"`}// 8.开机动作type BootMotion struct { Enable bool `description:"使能"` Action PTZAction`description:"关联预置点、巡航组、线性扫描、巡迹动作"` Auto string `description:"关联开机自动动作"`}

### 函数

* // @Title 添加开机动作// @Summary 添加空闲动作// @Description 添加开机动作,接收前端传过来的JSON// @Param cruise body dsd.BootMotion true "开机动作JSON"// @Success 200// @Failure 400000 操作失败// @router /bootMotion [post]func (c \*Controller) BootMotion() {}

## 刷新开机动作

### 函数

* // @Title 刷新开机动作// @Summary 刷新开机动作// @Description 刷新开机动作,给前端返回JSON，前端做判断,前端选项是否与配置文件的相同，不同的恢复到配置的状态// @Success 200// @Failure 400000 操作失败// @router /refreshbootMotion [get]func (c \*Controller) RefreshBootMotion() {}

# 区域扫描

## 开始区域扫描

### 结构体

* // 9.区域扫描type RegionScan struct { Enable bool `description:"使能"` UpMargin float64 `description:"设置上边界"` DownMargin float64 `description:"设置下边界"` LeftMargin float64 `description:"设置左边界"` RightMargin float64 `description:"设置右边界"` RegionPreset []conf.PresetPoint}

### 函数

* // @Title 开始区域扫描// @Summary 开始区域扫描// @Description 开始区域扫描,接收前端传过来的JSON,根据使能判断。“弓”字扫描，8个预置点组合成巡航组；烟雾及火点检测，火点屏蔽区域，触发火点居中及联动放大。// @Param cruise body dsd.RegionScan true "开机动作JSON"// @Success 200// @Failure 400000 操作失败// @router /startRegionScan [post]func (c \*Controller) StartRegionScan() {}

## 停止区域扫描

### 函数

* // @Title 停止区域扫描// @Summary 停止区域扫描// @Description 停止区域扫描,接收前端传过来的JSON// @Param cruise body dsd.RegionScan true "开机动作JSON"// @Success 200// @Failure 400000 操作失败// @router /stopRegionScan [post]func (c \*Controller) StopRegionScan() {}

# 云台限位

## 垂直限位

### 函数

* // @Title 水平限位// @Summary 水平限位// @Description 水平限位,接收前端传过来的JSON// @Param cruise body dsd.PanLimits true "水平限位JSON"// @Success 200// @Failure 400000 操作失败// @router /panLimit [post]func (c \*Controller) PanLimit() {}

## 水平限高

### 函数

* // @Title 垂直限位// @Summary 垂直限位// @Description 垂直限位,接收前端传过来的JSON// @Param cruise body dsd.TiltLimits true "垂直限位JSON"// @Success 200// @Failure 400000 操作失败// @router /tiltLimit [post]func (c \*Controller) TiltLimit() {}

## X轴原点

### 函数

* // @Title X轴原点// @Summary X轴原点// @Description X轴原点// @Success 200// @Failure 400000 操作失败// @router /xOriginPoint [post]func (c \*Controller) XOriginPoint() { // FF 0B 00 4B 00 00 56}

# 定时任务

## 添加定时任务

### 结构体

* // 11.定时任务type PTZCrontab struct { Enable bool `description:"使能"` ID uint8 `description:"ID"` PTZCrontabAction Action AutoReset uint8 `description:""` Schedule types.WeekSchedule `description:"布防计划"`}//Linkage 联动动作type Linkage struct { Record types.LinkChannel `description:"联动录像通道"` Snap types.LinkChannel `description:"联动抓图通道"` AlarmOut types.LinkChannel `description:"联动I/O通道"` PTZ PTZAction `description:"联动云台操作"` SMTP bool `description:"联动发送邮件"` FTP bool `description:"联动上传FTP"` Buzzer bool `description:"联动蜂鸣器"` VoicePrompt types.LinkVoice `description:"联动语音提示"` Flash types.LinkLight `description:"联动白光灯闪光"`}

### 结构体

* // @Title 添加定时任务// @Summary 添加定时任务// @Description 添加定时任务,接收前端传过来的JSON// @Param cruise body dsd.PTZCrontab true "定时任务JSON"// @Success 200// @Failure 400000 操作失败// @router /addPTZCronTab [post]func (c \*Controller) AddPTZCronTab() {}

## 刷新定时任务

### 函数

* // @Title 刷新定时任务// @Summary 刷新定时任务// @Description 刷新定时任务,给前端返回JSON，前端做判断,前端选项是否与配置文件的相同，不同的恢复到配置的状态// @Success 200// @Failure 400000 操作失败// @router /refreshPTZCronTab [get]func (c \*Controller) RefreshPTZCronTab() {}

## 清空任务

### 函数

* // @Title 删除定时任务// @Summary 删除定时任务// @Description 删除定时任务// @Param id query uint8 true "定时任务id"// @Success 200// @Failure 400000 操作失败// @router /deletePTZCronTab [delete]func (c \*Controller) DeletePTZCronTab() {}

# 云台重启、恢复默认

## 云台重启

### 函数

* // @Title 云台重启// @Summary 云台重启// @Description 云台重启// @Success 200// @Failure 400000 操作失败// @router /ptzReboot [post]func (c \*Controller) PTZReboot() {// FF 0B 00 0F 00 00 1A}

## 恢复默认

### 函数

* // @Title 恢复默认// @Summary 恢复默认// @Description 恢复默认，清除所有云台配置，恢复到默认配置。// @Success 200// @Failure 400000 操作失败// @router /ptzDefaultConfig [post]func (c \*Controller) PTZDefaultConfig() { }

# 精确定位

## 函数

### // @Title 精确定位// @Summary 精确定位// @Description 精确定位// @Param pan query uint8 true "水平角度(0~3600) "// @Param tilt query uint8 true "垂直角度(-900~900) "// @Param zoom query uint8 true "变倍(1~128) "// @Success 200// @Failure 400000 操作失败// @router /ptzPosition [post]func (c \*Controller) PTZPosition() {}