北京汉王智通

停车场一体机SDK接口

**使 用 说 明 书**

目录

[1 概述 3](#_Toc381186745)

[2 基本定义 3](#_Toc381186746)

[2.1 SDK输出文件定义 3](#_Toc381186747)

[2.2 SDK要求 3](#_Toc381186748)

[2.3 公共数据定义 3](#_Toc381186749)

[2.3.1 宏定义 3](#_Toc381186750)

[2.3.2 重定义 3](#_Toc381186751)

[2.3.3 枚举定义 3](#_Toc381186752)

[2.3.4 结构体定义 3](#_Toc381186753)

[2.4 开发接口定义 3](#_Toc381186754)

[2.4.1 回调函数定义 3](#_Toc381186755)

[2.4.2 是否记录开发包日志 3](#_Toc381186756)

[2.4.3 获取反馈代码描述 3](#_Toc381186757)

[2.4.4 初始化 3](#_Toc381186758)

[2.4.5 释放资源 3](#_Toc381186759)

[2.4.6 扫描设备 3](#_Toc381186760)

[2.4.7 网络参数设置 3](#_Toc381186761)

[2.4.8 连接设备 3](#_Toc381186762)

[2.4.9 断开连接 3](#_Toc381186763)

[2.4.10 非阻塞强制抓拍 3](#_Toc381186764)

[2.4.11 阻塞强制抓拍 3](#_Toc381186765)

[2.4.12 检测状态 3](#_Toc381186766)

[2.4.13 校时 3](#_Toc381186767)

[2.4.14 是否启用NTP校时 3](#_Toc381186768)

[2.4.15 版本信息获取 3](#_Toc381186769)

[2.4.16 设备参数控制（内部停车场收费软件使用） 3](#_Toc381186770)

[2.4.17 获取设备快门值（研发测试用） 3](#_Toc381186771)

[2.4.18 设置设备快门值（内部测试用） 3](#_Toc381186772)

[2.4.19 设置工作模式（研发测试用） 3](#_Toc381186773)

[2.5 接口调用流程 3](#_Toc381186774)

[2.6 注意事项 3](#_Toc381186775)

**变更记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **版本号** | **修改点说明** | **变更人** | **存放位置** | **审批人** | **审批日期** |
| Ver 1.0 | 创建文档 | 褚晓玲 |  |  |  |
| Ver 1.1 | 1. 增加NTP是否校时使能接口 2. 增加接口函数调用流程 3. 增加自动化测试工具使用的接口说明， 4. 增加实时JPG流使能接口 5. 回调中增加用户数据输出 6. 各个数据输出结构体，增加设备句柄 | 褚晓玲 |  |  |  |

修改点说明的内容有如下几种：创建、修改（+修改说明）、删除（+删除说明）

# 概述

停车场一体机接口主要包括以下文件：

LPRSDK.dll、LPRSDK.lib、LPR\_Interface.h、LPR\_CommonDefine.h

LPRSDK.dll：动态库链接库文件

LPRSDK.lib：静态库lib文件

LPR\_Interface.h：接口头文件

LPR\_CommonDefine.h：数据类型定义文件

采用非C、C++语言调用时，以下定义接口声明和数据类型定义文件需要在其他平台中，重新编写。本开发包仅提供 C、C++、C#三类语言的接口声明及类型定义描述文件，其中C、C++，见开发包中“LPR\_CommonDefine.h”文件，C#见开发包中“LPRSDK.cs”文件。

以下定义描述和“LPR\_CommonDefine.h”不一致的，以“LPR\_CommonDefine.h”文件的定义为准。

开发包提供内容如下：

|  |  |  |
| --- | --- | --- |
| 描述 | 包含内容 | 说明 |
| 接口文件 | LPR\_CommonDefine.h、  LPR\_Interface.h 、LPRSDK.dll、  LPRSDK.lib | 接口相关文件及定义 |
| VC测试示例 | VC6.0测试工程，采用静态调用模式 |  |
| C#测试用例 | VS2008开发 | 其中包含接口定义和类型定义文件可供参考 |

注意：

该开发包仅提供Windows系统调用。

# 基本定义

## SDK输出文件定义

* WINDOWS操作系统

SDK包的输出形式以动态链接库方式提供用户进行二次开发，其中动态连接库的输出文件包括：头文件（.h）、动态库（.dll）、静态库（.lib）。

动态连接库为标准WIN32动态库，动态库中的函数为标准WINAPI类型函数。

## SDK要求

## 公共数据定义

公共数据包括接口中用到的结构体类型、枚举类型、宏等，以下是相关数据的定义。

### 宏定义

|  |  |  |
| --- | --- | --- |
| 宏定义 | 值 | 含义 |
| INVALID\_HANLE | 0 | 无效句柄 |
| IP\_MAX\_LEN | 16 | IP地址字符串最大长度 |
| MACADDR\_LEN | 6 | MAC地址 |
| VERSION\_LEN | 255 | 版本长度 |
| MAX\_PLATE\_LEN | 16 | 车牌号码长度 |
| MAX\_TIME\_LEN | 8 | 时间长度 |
| AUTHORINFO\_LEN | 100 | 授权相关信息长度 |
| SERIALNO\_LEN | 20 | 序列号长度 |
| FEEKMSG\_MAX\_LEN | 100 | 反馈描述信息长度 |
| CMD\_READ | 0 | 读指令 |
| CMD\_WRITE | 1 | 写指令 |
| ENABLE\_FUNCTION | 1 | 某项功能使能 |
| DISABLE\_FUNCTION | 0 | 某项功能不使能 |

### 重定义

typedef void \* PDEVICEHANDLE;

### 枚举定义

#### 函数反馈值定义

typedef enum

{

RESULT\_OK = 0, // 执行成功

INIT\_SUCCESS = 1, // 已经初始化

NO\_INIT = 2, // 未初始化

PARAMS\_ERROR = 3, // 参数不正确

SOCKET\_NULL = 4, // SOCKET句柄为NULL

THREAD\_FAIL = 5, // 线程失败

CREATE\_FAIL = 6, // 生成命令失败

SEND\_FAIL = 7, // 发送命令失败

NO\_FIND\_DEVICE = 8, // 没有发现设备

DEVICE\_OPENED = 9, // 设备已经打开

DEVICE\_DISCONNECT = 10, // 和设备的连接断开

CONTORL\_NOCONNECTION = 11, // 控制端口未连接

NO\_RECEIVED\_DATA = 12, // 阻塞触发没有收到数据

OTHER\_ERROR = 100, // 其他错误

TIME\_OUT = 209 // 函数执行超时

} FEEKBACK\_TYPE;

#### 设备状态

typedef enum

{

CONNECT\_SUCCESS = 0, // 连接成功

CREATE\_ERROR = 1, // 生成句柄错误

CONNECT\_ERROR = 2, // 连接错误

ABNORMALNET\_ERROR = 3 // 网络异常错误

} DEVICE\_STATUS;

#### 车牌颜色定义

typedef enum

{

NON\_PLATE = 0, // 未知车牌颜色

BLUE\_COLOR, // 蓝色

WHITE\_COLOR, // 白色

BLACK\_COLOR, // 黑色

YELLOW\_COLOR // 黄色

} PLATE\_COLOR;

#### 触发类型

typedef enum

{

UNKNOWN\_TRIGGER\_MODE = 0, // 未知

VIDEO\_TRIGGER\_MODE = 1, // 视频

LINE\_TRIGGER\_MODE = 2, // 线圈

SOFT\_TRIGGER\_MODE // 模拟

} DEVICE\_TRIGGERMODE;

#### 用户类型定义

typedef enum

{

NORMAL\_USER = 0, // 正常用户

UNKNOWN\_USER = 1 // 未授权用户

} USER\_TYPE;

#### 指令主类定义

typedef enum

{

SYSTEM\_CMD = 0x00, // 系统指令

LIGHT\_CMD = 0x20, // 调光指令

OSD\_CMD = 0x30, // OSD指令

EPLD\_CMD = 0x60, // 单片机指令

LED\_CMD = 0x70, // 补光指令

CAMERA\_CMD = 0x71, // 相机控制指令

MOVE\_CMD = 0x80, // 动作指令

CAPTURE\_DATA = 0xFF // 抓拍数据

} MAIN\_CMD;

#### 系统指令次指令定义

typedef enum

{

HEART\_CMD = 0x00, // 心跳

VERSION\_INFO = 0x04, // 版本信息

IMAGE\_QUALITY = 0x05, // 图像压缩比

DEVICE\_RESTART = 0x0C, // 是否重启

RESTART\_TIME = 0x0D, // 重启时间

OCR\_PARAMS = 0x16, // 识别参数

SERIRAL\_NO = 0x18, // 设备序列号

TIME\_SPACE = 0x19, // 车辆间隔时间

ALG\_PARAMS = 0x1A, // 算法参数

DEVICE\_ID = 0x1B, // 设备编号

DEVICE\_CODE = 0x1C, // 机器码

AUTHOR\_INFO = 0x1D, // 授权码

AUTHOR\_VALIDTIME = 0x23, // 授权截止时间

WORK\_MODE = 0x1E, // 工作模式，正常，曝光测试模式

VERIFYTIME\_MODE = 0x1F, // 校时方式SDK 1 NTP

NTP\_SERVERIP = 0x20 // NTP服务IP

} SYSTEM\_SUBCMD;

#### 系统指令中对象定义

##### 版本对象

typedef enum

{

MAIN\_VERSION = 0x00, // 主版本

DSP\_SUBVERSION = 0x01, // 次版本

MCU\_VERSION = 0x02, // MCU版本

ALG\_VERSION = 0x03, // 识别算法版本

JPG\_VERSION = 0x04, // JPG版本信息

EEPROM\_VERSION = 0x05 // EEPROM版本号

} VERSION\_OBJECT;

##### 图像压缩比

typedef enum

{

BIG\_IMAGE\_QUALITY = 0x00, // 大图

CIF\_IMAGE\_QUALITY = 0x01, // CIF图

PLATE\_IMAGE\_QUALITY = 0x02 // 车牌图

} IMAGE\_QUALITY\_OBJECT;

#### 调光指令次指令定义

typedef enum

{

LIGHT\_MODE = 0x00, // 调光模式

AUTOLIGHT\_ENABLE = 0x01, // 自动调光使能

SYSTEM\_TIME= 0x02, // 系统时间

LIGHT\_PARAMS = 0x03, // 调光参数表，分四套

IMAGE\_BRIGHTNESS = 0x07, // 图片亮度

CAMERA\_CURSTATUS = 0x0A, // 相机当前状态

CAMERA\_PARAMS = 0x0B, // 摄像头参数

CUR\_LIGHTBASE = 0x0D, // 当前调光基准

LIGHT\_VALIDNUM = 0x0E, // 调光有效次数

DAY\_EXPOSURE\_COMPENSATION = 0x0F, // 白天曝光补偿

NIGHT\_EXPOSURE\_COMPENSATION = 0x10, // 夜间曝光补偿

REF\_GAIN = 0x11, // 参考增益

REF\_BRIGHTNESS = 0x12 // 参考亮度

} LIGHT\_SUBCMD;

#### 调光指令中对象及模式定义

##### 调光模式值

typedef enum

{

AUTO\_LIGHT = 0x00, // 自动调光

MIXED\_LIGHT = 0x01 // 混合调光

} LIGHT\_MODE\_VALUE;

##### 相机当前状态对象

typedef enum

{

SHUTTER\_VALUE = 0x00, // 快门

APARTURE\_VALUE = 0x01, // 光圈

GAIN\_VALUE = 0x02, // 增益

VIEW\_VALUE = 0x03, // 视野

F\_VALUE = 0x04, // 焦距

BRIGHT\_NUM = 0x05, // 调亮

DARK\_NUM = 0x06, // 调暗

} CAMERA\_CURSTATUS\_PARAMS;

注意：

其中快门、光圈、增益三项也可用于调光参数中的对象应用。

#### OSD叠加指令次指令定义

typedef enum

{

OSD\_ENABLE = 0x0C, // 使能控制

OSD\_INFO = 0x0D // 信息内容

} OSD\_SUBCMD;

#### 叠加使能控制对象定义

typedef enum

{

OSD\_TIME\_ENABLE = 0x00, // 时间叠加使能

OSD\_INFO\_ENABLE = 0x03, // 自定义信息叠加使能

OSD\_PLATE\_ENABLE = 0x04 // 识别结果叠加使能

} OSD\_ENABLE\_OBJECT;

#### 单片机指令次指令定义

typedef enum

{

LINE\_POLARITY = 0x01, // 线圈触发极性

CAMERA\_MENU = 0x0A, // 相机菜单

VIEW\_F = 0x0B // 视野焦距

} EPLD\_SUBCMD;

#### 视野焦距对象定义

typedef enum

{

VIEW\_OBJECT = 0x00, // 视野

F\_OBJECT = 0x01 // 焦距

} VIEW\_F\_OBJECT;

#### 补光指令次指令定义

typedef enum

{

ENABLE\_LED = 0x00, // 使能

OPEN\_CLOSE\_BASE = 0x01, // 开关灯基准

LED\_BRIGHTNESS = 0x02, // LED亮度

LED\_STATUS = 0x03, // LED状态

LED\_DELAY = 0x04 // LED延迟

} LED\_SUBCMD;

#### 补光指令中开关灯基准对象定义

typedef enum

{

OPEN\_BASE = 0x00, // 开灯基准

CLOSE\_BASE = 0x01 // 关灯基准

} LED\_STATUS\_OBJECT;

#### 相机控制指令次指令定义

typedef enum

{

POINT\_EXPOSURE = 0x06, // 点曝光

CAMERA\_PHASE = 0x07 // 相机相位

} CAMERA\_CONTROL\_CMD;

### 结构体定义

#### 设备网络参数

typedef struct tagNetParams

{

unsigned char ucDeviceIP[IP\_MAX\_LEN]; // 设备IP

unsigned char ucMaskIP[IP\_MAX\_LEN]; // 子网掩码

unsigned char ucGateIP[IP\_MAX\_LEN]; // 网关IP

unsigned char ucMac[MACADDR\_LEN]; // MAC地址

} NetParams, \*PNetParams;

#### 设备基本信息

typedef struct tagDeviceInfo

{

NetParams DeviceNetInfo; // 设备网络信息参数

unsigned char ucMAC[MACADDR\_LEN]; // MAC地址

unsigned char ucDeviceVersion[VERSION\_LEN]; // 设备版本信息

unsigned char ucConnectNum; // 用户连接数

} DeviceInfo, \*PDeviceInfo;

#### 设备状态

typedef struct tagDeviceStatus

{

PDEVICEHANDLE pHandle; // 设备句柄

unsigned char ucDeviceIP[IP\_MAX\_LEN]; // 设备IP

DEVICE\_STATUS status; // 设备状态

} DeviceStatus, \*PDeviceStatus;

#### 输出内容控制

typedef struct tagOutputContent

{

bool bOutputBigImage; // 是否输出大图

bool bOutputCIFImage; // 是否输出CIF图

bool bOutputPlateImage; // 是否输出车牌图

} OutputContent, \*POutputContent;

#### 抓拍数据

typedef struct tagVehicleData

{

PDEVICEHANDLE pDeviceHandle; // 设备句柄

unsigned short usSerialNO; // 序列号

unsigned char ucDeviceIP[IP\_MAX\_LEN]; // 设备IP

unsigned int ui32DeviceID; // 设备ID

USER\_TYPE ui32UserType; // 表示用户类型:0 正常用户，未授权用户，其他值保留

unsigned int ui32Brightness; // 亮度值

DEVICE\_TRIGGERMODE triggermode; // 触发类型:1 视频2 线圈3 模拟0 未知，其他值保留

unsigned char ai8UserDefineInfo[VERSION\_LEN]; // 用户自定义信息

unsigned char ucTime[MAX\_TIME\_LEN]; // 时间，年（减掉后的年份）、月、日、时、分、秒、毫秒（占两个字节）

unsigned char ucPlate[MAX\_PLATE\_LEN]; // 车牌号码

PLATE\_COLOR PlateColor; // 车牌颜色

unsigned char \*pucBigImage; // 抓拍大图数据

unsigned int uiBigImageLen; // 抓拍大图数据长度

unsigned short usBigImageWidth; // 抓拍大图宽度

unsigned short usBigImageHeight; // 抓拍大图高度

unsigned char \*pucCIFImage; // 抓拍CIF图数据

unsigned int uiCIFImageLen; // 抓拍CIF图数据长度

unsigned short usCIFImageWidth; // 抓拍CIF图宽度

unsigned short usCIFImageHeight; // 抓拍CIF图高度

unsigned char \*pucPlateImage; // 抓拍车牌图数据

unsigned int uiPlateImageLen; // 抓拍车牌图数据长度

unsigned short usPlateImageWidth; // 抓拍车牌图宽度

unsigned short usPlateImageHeight; // 抓拍车牌图高度

} VehicleData, \*PVehicleData;

#### 实时JPG流（暂不提供）

typedef struct tagJPGData

{

PDEVICEHANDLE pDeviceHandle; // 设备句柄

unsigned char ucDeviceIP[IP\_MAX\_LEN]; // 设备IP

unsigned char \*pucImage; // 一帧实时JPG流数据

unsigned int uiImageLen; // 一帧实时JPG流数据长度

unsigned short usImageWidth; // 一帧实时JPG流宽度

unsigned short usImageHeight; // 一帧实时JPG流高度

unsigned short usLeft; // 车牌位置坐标点，左

unsigned short usRight; // 车牌位置坐标点，上

unsigned short usTop; // 车牌位置坐标点，右

unsigned short usBottom; // 车牌位置坐标点，下

} JPGData, \*PJPGData;

#### 版本信息

typedef struct tagVersionInfo

{

unsigned char ucMainVer[VERSION\_LEN]; // 主版本

unsigned char ucDSPVer[VERSION\_LEN]; // 次版本

unsigned char ucMCUVer[VERSION\_LEN]; // MCU版本

unsigned char ucOCRVer[VERSION\_LEN]; // 识别算法版本

unsigned char ucEPROMVer[VERSION\_LEN]; // EEPROM版本号

} VersionInfo, \*PVersionInfo;

#### 控制参数结构体

typedef struct tagParamsInfo

{

unsigned char ucMainCmd; // 主指令

unsigned char ucSubCmd; // 次指令

bool bHasObject; // 是否有对象号

unsigned char ucObject; // 对象号

unsigned char ucCmdType; // 命令类型，读或者写

unsigned short usCmdLen; // 命令长度

unsigned char ucCmd[CMD\_LEN]; // 命令内容

} ParamsInfo, \*PParamsInfo;

#### 算法参数相关定义

// 主行方向

typedef enum

{

LEFT\_TOP = 0, // 左上行驶

LEFT\_BOTTOM, // 左下行驶

RIGHT\_TOP, // 右上行驶

RIGHT\_BOTTOM, // 右上行驶

UNDECIDED // 不确定

} MAINDIRECT;

// 触发方式

typedef enum

{

LINE\_TRIGGER = 0, // 线圈触发

VIDEO\_TRIGGER = 2 // 视频触发

} TRIGGER\_MODE;

// 抓拍识别

typedef struct tagAlgParams

{

TRIGGER\_MODE TriggerMode; // 线圈触发或视频触发

unsigned short usLeft; // 检测区域左边界

unsigned short usTop; // 检测区域上边界

unsigned short usRight; // 检测区域右边界

unsigned short usBottom; // 检测区域下边界

MAINDIRECT MainDirect; // 主行方向：左上，左下，右上，右下

bool bGetOppEn; // 是否抓拍逆行

} AlgParams, \*PAlgParams;

## 开发接口定义

### 回调函数定义

#### 设备信息回调

typedef void(CALLBACK \*DeviceInfoCallback)(void \*pUserData, DeviceInfo \*pDeviceInfo);

参数说明：

pUserData：用户设置回调函数传入的用户数据

pDeviceInfo：见DeviceInfo结构体定义

#### 设备状态回调

typedef void(CALLBACK \*DeviceStatusCallback) (void \*pUserData, DeviceStatus \*pStatus);

参数说明：

pUserData：用户设置回调函数传入的用户数据

pStatus：见DeviceStatus结构体定义

#### 抓拍数据回调

typedef void(CALLBACK \*VehicleDataCallback)(void \*pUserData, VehicleData \*pData);

参数说明：

pUserData：用户设置回调函数传入的用户数据

pData：见VehicleData结构体定义

#### 实时JPG流回调

typedef void(CALLBACK \*JPGStreamCallBack)(void \*pUserData, JPGData \*pJPGData);

参数说明：

pUserData：用户设置回调函数传入的用户数据

pJPGData：见JPGData结构体定义

### 是否记录开发包日志

函数实体：unsigned int WINAPI LPR\_IsWriteLog(bool bWriteLog);

函数名称: LPR\_IsWriteLog

函数说明：是否记录动态库日志。日志文件自动生成在调用程序所在目录下，自动建立LPRLog文件夹，然后日志名称以时间命名，如：“20110101.log”

参数说明：

bWriteLog：true记录日志，false不记录日志

返回值：

RESULT\_OK表示成功，否则返回错误代码

### 获取反馈代码描述

函数实体：unsigned int WINAPI LPR\_GetErrorMsg(FEEKBACK\_TYPE FeekCode, char \*pchErrMsg);

函数名称: LPR\_GetErrorMsg

函数说明：获取反馈值对应的文字描述

参数说明：

FeekCode：反馈码，即接口函数返回值

pchErrMsg：反馈代码对应的文字描述，分配空间大小为FEEKMSG\_MAX\_LEN

返回值：

RESULT\_OK表示成功，否则返回错误代码

### 初始化

函数实体：unsigned int WINAPI LPR\_Init(HWND hWnd, void \*pUserData, DeviceInfoCallback fucDeviceInfo, DeviceStatusCallback fucStatus, VehicleDataCallback fucVehicleData, JPGStreamCallBack fucJPGStream);

函数名称: LPR\_Init

函数说明：设置回调函数

参数说明：

hWnd：如此值为NULL，则回调函数在动态库所建立线程中响应；反之，在该句柄所在线程响应

pUserData：用户数据，在回调函数响应时传出

fucDeviceInfo：设备信息回调函数，见DeviceInfoCallback函数定义

fucStatus：设备状态回到函数，见DeviceStatusCallback函数定义

fucVehicleData：抓拍数据回调函数，见VehicleDataCallback函数定义

fucJPGStream：实时JPG流回调函数，见JPGStreamCallBack函数定义

返回值：

RESULT\_OK表示成功，否则返回错误代码

### 释放资源

函数实体：unsigned int WINAPI LPR\_Quit();

函数名称: LPR\_Quit

函数说明：释放动态库所有资源

返回值：

RESULT\_OK表示成功，否则返回错误代码

### 扫描设备

函数实体：unsigned int WINAPI LPR\_ScanDevice();

函数名称：HWTC\_ ScanDevice

函数说明：通过UDP命令，查找同一网段IP对应的设备

返回值：

RESULT\_OK表示成功，否则返回错误代码

备注：

执行该函数后，如有扫描到设备，会立即响应设备信息回调函数。

### 网络参数设置

函数实体：unsigned int WINAPI LPR\_SetNetParams(PDEVICEHANDLE pHandle, NetParams \*pOldParams, NetParams \*pNewParams);

函数名称: LPR \_ConnectCamera

函数说明：网络参数设置

参数说明：

pHandle：如果是UDP方式设置，则该参数是0；反之TCP时，指通过LPR \_ConnectCamera获取

pOldParams：设备原始网络参数，UDP设置时，需要

pNewParams：设备新设置的参数

返回值：

返回值是RESULT\_OK表示成功，否则返回错误代码

### 连接设备

函数实体：unsigned int WINAPI LPR\_ConnectCamera(char \*pchDeviceIP, PDEVICEHANDLE pHandle);

函数名称: LPR \_ConnectCamera

函数说明：打开指定IP设备的连接

参数说明：

pchDeviceIP：待连接的设备IP地址

pHandle：执行成功后，返回设备句柄；否则为INVALID\_HANLE

返回值：

返回值是RESULT\_OK表示成功，否则返回错误代码

### 断开连接

函数实体：unsigned int WINAPI LPR \_DisconnectCamera(PDEVICEHANDLE pHandle);

函数名称: LPR \_DisconnectCamera

函数说明：断开指定设备的连接

参数说明：

pHandle：设备句柄，通过LPR \_ConnectCamera获取

返回值：

返回值是RESULT\_OK表示成功，否则返回错误代码

### 非阻塞强制抓拍

函数实体：unsigned int WINAPI LPR \_Capture(PDEVICEHANDLE pHandle);

函数名称：LPR \_ Capture

函数说明：强制抓拍，向设备发送抓拍命令，设备抓拍图像和采集信息，并上传。该函数只保证参数下发成功后立即返回，不考虑设备是否执行该命令

参数说明:

pHandle：相机句柄，由LPR \_ConnectCamera获取

返回值：

返回值是RESULT\_OK表示成功，否则返回错误代码

### 阻塞强制抓拍

函数实体：unsigned int WINAPI LPR\_CaptureEx(PDEVICEHANDLE pHandle, unsigned int nTimeout);

函数名称：LPR \_ CaptureEx

函数说明：强制抓拍，向设备发送抓拍命令，设备抓拍图像和采集信息，并上传。该函数下发命令后，等待设备响应，直到收到设备传出的信息或者超时时间到，才返回。

参数说明:

pHandle：相机句柄，由LPR \_ConnectCamera获取

nTimeout：阻塞超时时间，以ms为单位

返回值：

返回值是RESULT\_OK表示成功，否则返回错误代码

### 检测状态

函数实体：unsigned int WINAPI LPR \_CheckStatus(PDEVICEHANDLE pHandle, DEVICE\_STATUS \*pConnStatus);

函数名称：LPR \_CheckStatus

函数说明：检测状态

参数说明:

pHandle：相机句柄，由LPR \_ConnectCamera获取

pConnStatus：状态标记，见DEVICE \_STATUS枚举

返回值：

返回值是RESULT\_OK表示成功，否则返回错误代码

### 校时

函数实体：unsigned int WINAPI LPR \_AdjustTime(PDEVICEHANDLE pHandle,DWORD64 dwTimeMs = 0);

函数名称：LPR \_AdjustTime

函数说明：给设备校时

参数说明:

pHandle：相机句柄，由LPR \_ConnectCamera获取

dwTimeMs：时间值(1970-1-1 0:0:0以来的秒数)，64位整数，默认为0，当为0时，直接获取当前系统时间，进行校时

返回值：

返回值是RESULT\_OK表示成功，否则返回错误代码

### 是否启用NTP校时

函数实体：unsigned int WINAPI LPR\_NTPEnable(PDEVICEHANDLE pHandle, bool bNTP, char \*pchNTPSeverIP);

函数名称：LPR\_NTPEnable

函数说明：是否启用NTP校时，如果启用NTP校时，则LPR\_AdjustTime函数执行无效

参数说明:

pHandle：相机句柄，由LPR \_ConnectCamera获取

bNTP true：表示启用；false表示不启用

pchNTPSeverIP：如启用，则需要设置NTP服务器的IP地址

返回值：

返回值是RESULT\_OK表示成功，否则返回错误代码

### 版本信息获取

函数实体：unsigned int WINAPI LPR \_GetAllVersion(PDEVICEHANDLE pHandle, VersionInfo \*pVerInfo);

函数名称：LPR \_ GetAllVersion

函数说明：获取设备所有版本信息

参数说明:

pHandle：相机句柄，由LPR \_ConnectCamera获取

pVerInfo：设备详细版本信息，见VersionInfo定义

返回值：

返回值是RESULT\_OK表示成功，否则返回错误代码

## 接口调用流程



## 注意事项

* 支持多设备连接
* 在响应回调函数时，需应用程序及时将数据移走，不能阻塞动态库中其他线程正常的工作流程。