红外热像仪  
网络SDK编程指南  
V1.4.0.1209

2022-11-1

# YoseenSDK简介

YoseenSDK是基于热像仪私有网络通信协议开发的，提供热像仪发现、热像仪参数配置和系统维护、温度或视频数据预览、单帧或多帧数据保存、数据后分析算法等接口。

|  |  |
| --- | --- |
| **文件** | **说明** |
| SDK  ├── YoseenSDK  │   ├── YoseenAlg\_Mea.h  │   ├── YoseenAlg\_Strech.h  │   ├── YoseenDeviceTypes.h  │   ├── YoseenFileTypes.h  │   ├── YoseenSDK.h  │   ├── x64  │   │   ├── YoseenFfmpeg.dll  │   │   ├── YoseenSDK.dll  │   │   ├── YoseenSDK.json  │   │   └── YoseenSDK.lib  │   └── x86  │   ├── YoseenFfmpeg.dll  │   ├── YoseenSDK.dll  │   ├── YoseenSDK.json  │   └── YoseenSDK.lib  └── yc\_type.h | SDK头文件+库文件，支持多平台。  请保持SDK目录的内部结构。  修改根目录SDK的名称，可能需要调整编译中的头文件和库文件依赖。  YoseenSDK.json用于配置日志目录、H264编解码。支持none=软编解码，qsv=INTEL集成显卡硬编解码。 |
| YoseenDemoVS  ├── YoseenDemo  ├── YoseenDemoConsole  ├── YoseenDemoCustomDraw  ├── YoseenDemoWinForm  ├── YoseenDemoWpf  └── YoseenSDKCS | Win32、控制台、WinForm、WPF示例  32位构建目录\build-Debug32  64位构建目录\build-Debug64  YoseenSDKCS是C#封装库 |
| YoseenDemoQt  ├── YoseenDemoQt.pro  ├── main.cpp  ├── mainwindow.cpp  ├── mainwindow.h  ├── mainwindow.ui  ├── tempcanvas.cpp  ├── tempcanvas.h  ├── yoseencamera.cpp  └── yoseencamera.h | QT示例 |
| YoseenDemoJava  ├── YoseenDemoJava.iml  ├── lib  │   └── jna.jar  └── src  ├── Test  │   ├── Test\*Template.java  │   ├── TestMeasure.java  │   ├── TestYoseenDevice.java  │   ├── TestYoseenDevice\_PTZ.java  │   └── TestYoseenFile.java  └── Yoseen  ├── EError.java  ├── YoseenData.java  ├── YoseenSDK.java  └── YoseenUtil.java | Java示例，基于JNA，IDEA工程  采用build-Debug64，作为运行目录 |
|  |  |

**SDK版本更新**

**Version 1.3.0.1128**

\*结构优化统一

**Version 1.0.0.0901**

\*[添加测温模板接口](#_测温模板)

**Version 1.0.0.0109**

\*添加文件回放接口[YoseenPlayback](#_文件回放)

\*优化图像算法

**Version 1.0.0.0901**

\*添加暂停预览[Yoseen\_PausePreview](#_暂停预览)

\*优化DDE算法

**Version 1.0.0.0619**

**Version 1.0.0.0529**

**Version 1.0.0.0**

# 函数调用顺序

## SDK基本调用流程

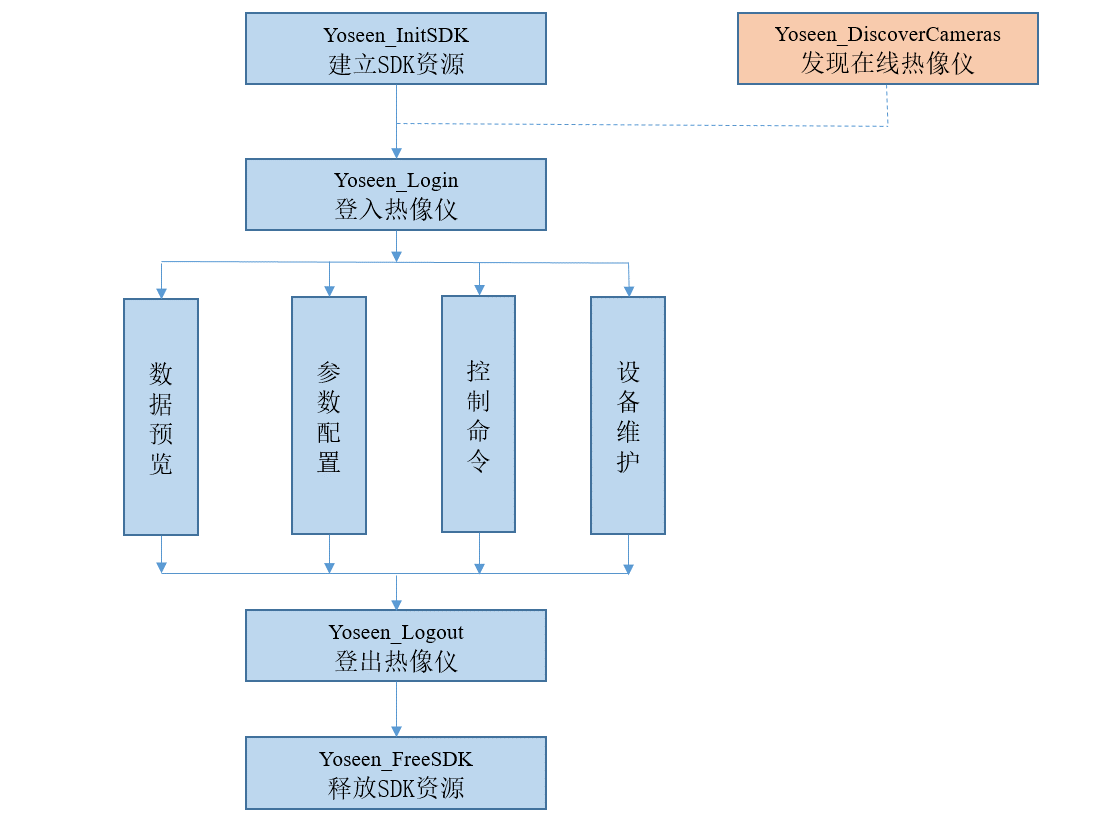


图2.1 基本调用流程

蓝色框建立SDK资源、登入热像仪、登出热像仪、释放SDK资源是必不可少的4个流程。橙色框发现在线热像仪是可选流程。

* 建立SDK资源Yoseen\_InitSDK：初始化网络、日志、显示、ffmpeg等。
* 登入热像仪Yoseen\_Login: 登入之后等到用户句柄，才可以调用各类需要用户句柄的函数。 SDK支持32个用户句柄，只验证热像仪IP地址；热像仪不对登入用户做限制。
* 登出热像仪Yoseen\_Logout：登出之后释放用户句柄。
* 释放SDK资源Yoseen\_FreeSDK：反初始化网络、日志、显示、ffmpeg等。
* 数据预览：

温度流预览，传输的是全像面温度数据，带宽占用大，支持1台热像仪对1个客户端的传输。SDK在预览回调函数中提供全像面温度数据、位图数据供用户处理。

视频流预览，传输的是H264编码的图像数据和特征温度数据，带宽占用小，支持1台热像仪对16个客户端传输。SDK在预览回调函数中提供特征温度数据、位图数据供用户处理。

* 参数配置：

获取和设置热像仪的基本信息、网络信息、挡板校零信息、模拟视频信息、测温修正信息等。

* 控制命令：

手动挡板校零、自动对焦、获取和设置热像仪当前时间、获取和设置热像仪测温档位等。

* 设备维护：

上传热像仪程序更新包、下载热像仪日志文件等。

## 数据预览调用流程

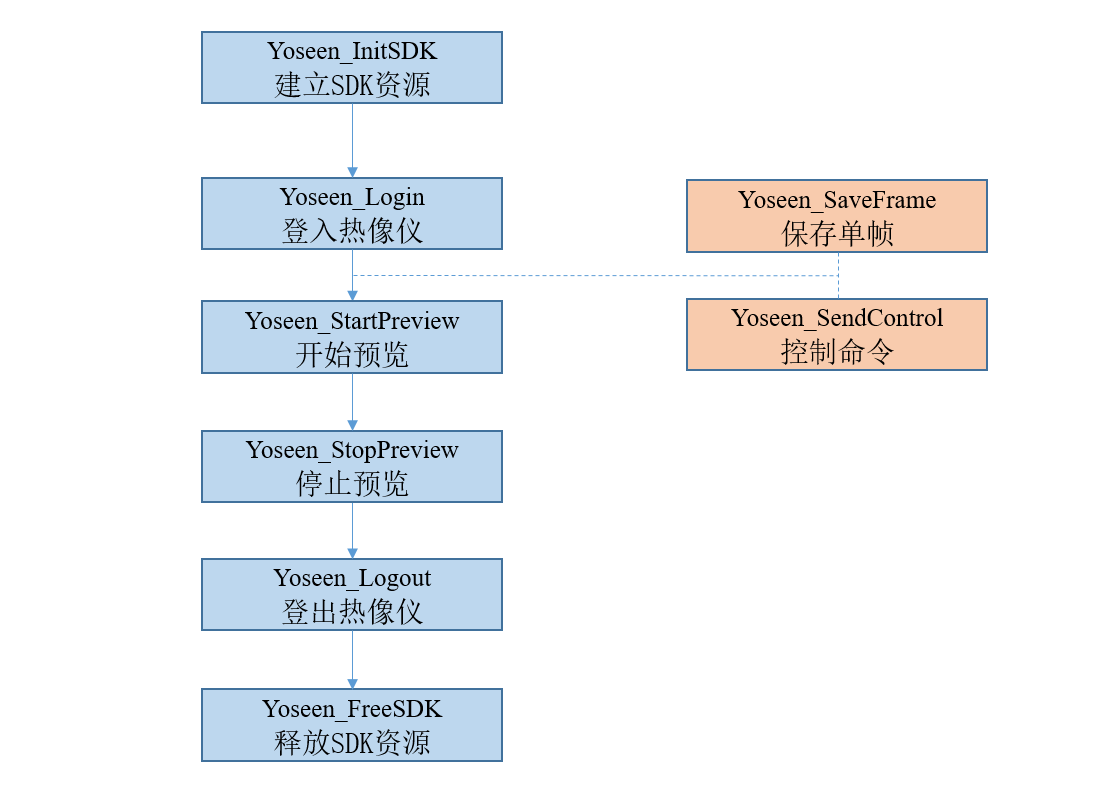


图2.2 数据预览调用流程图

蓝色框是必选流程，橙色框是可选流程。用户可关联窗口，实现画面显示；可设定预览回调，处理每帧数据。配置热像仪、发送控制命令、保存单帧jpg，只需要用户句柄就能使用，不管预览是否开启。

## 参数配置或控制命令流程

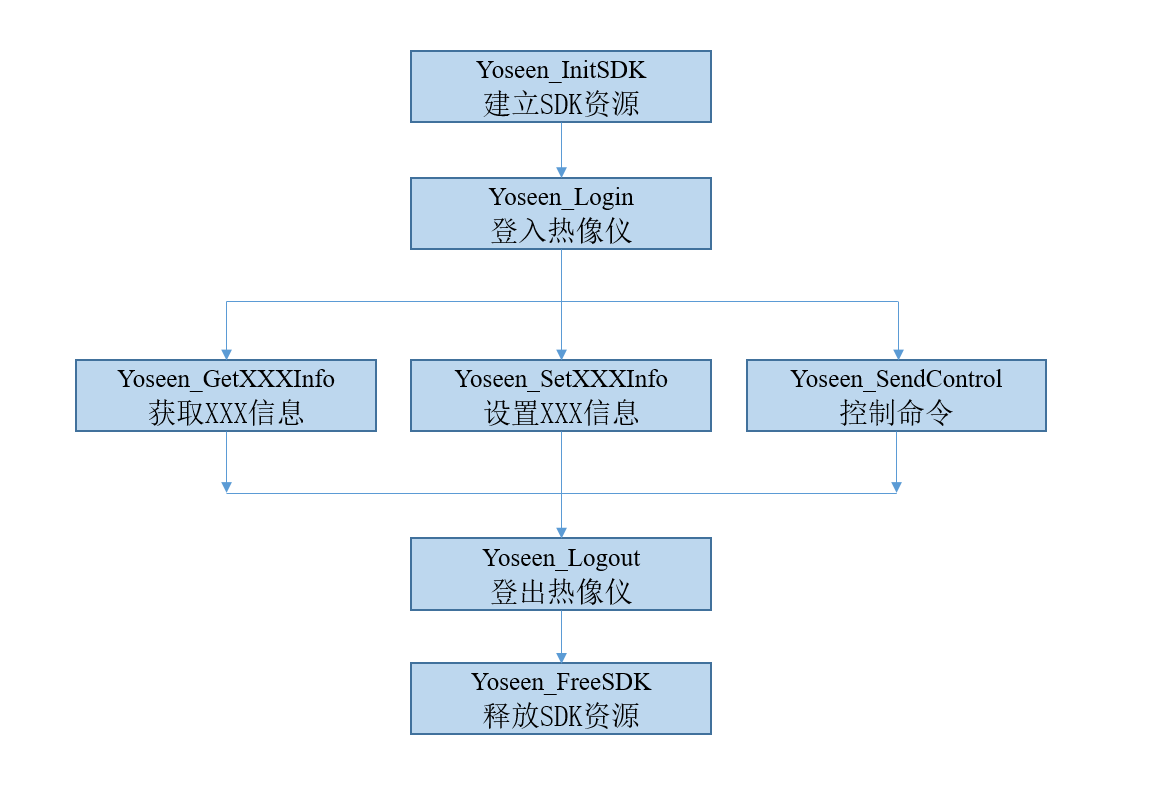


图2.3 参数配置或控制命令流程

# 函数调用实例

## YoseenDemo工程说明

YoseenDemo是使用VS2019开发的，分Win32、WinForm、Wpf三个工程。三个工程共用生成目录，32位生成目录为build-Debug32，64位生成目录为build-Debug64。界面逻辑和SDK调用一致。Win32工程代码说明如下：

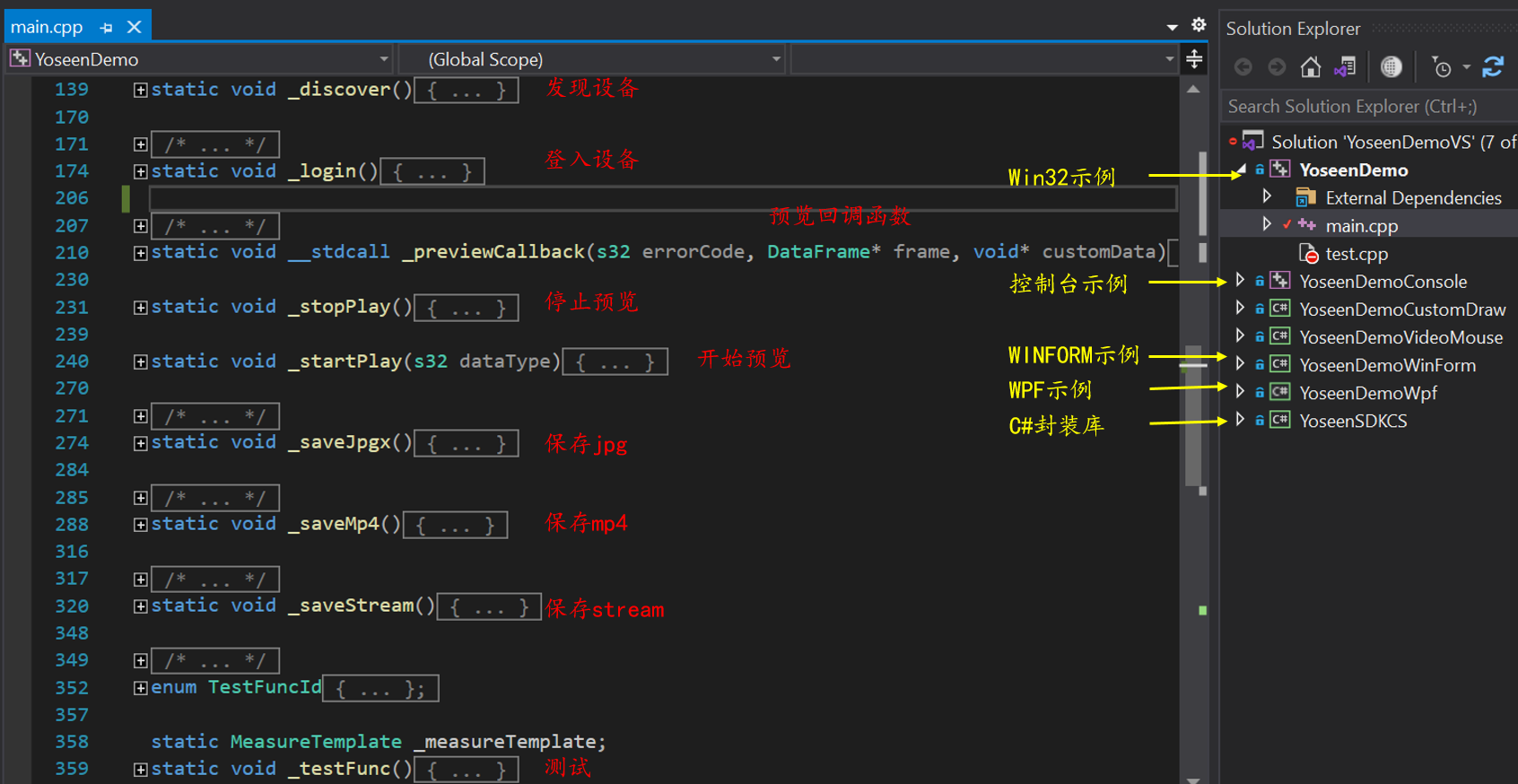


图3.1 YoseenDemo工程说明

## YoseenDemo界面说明

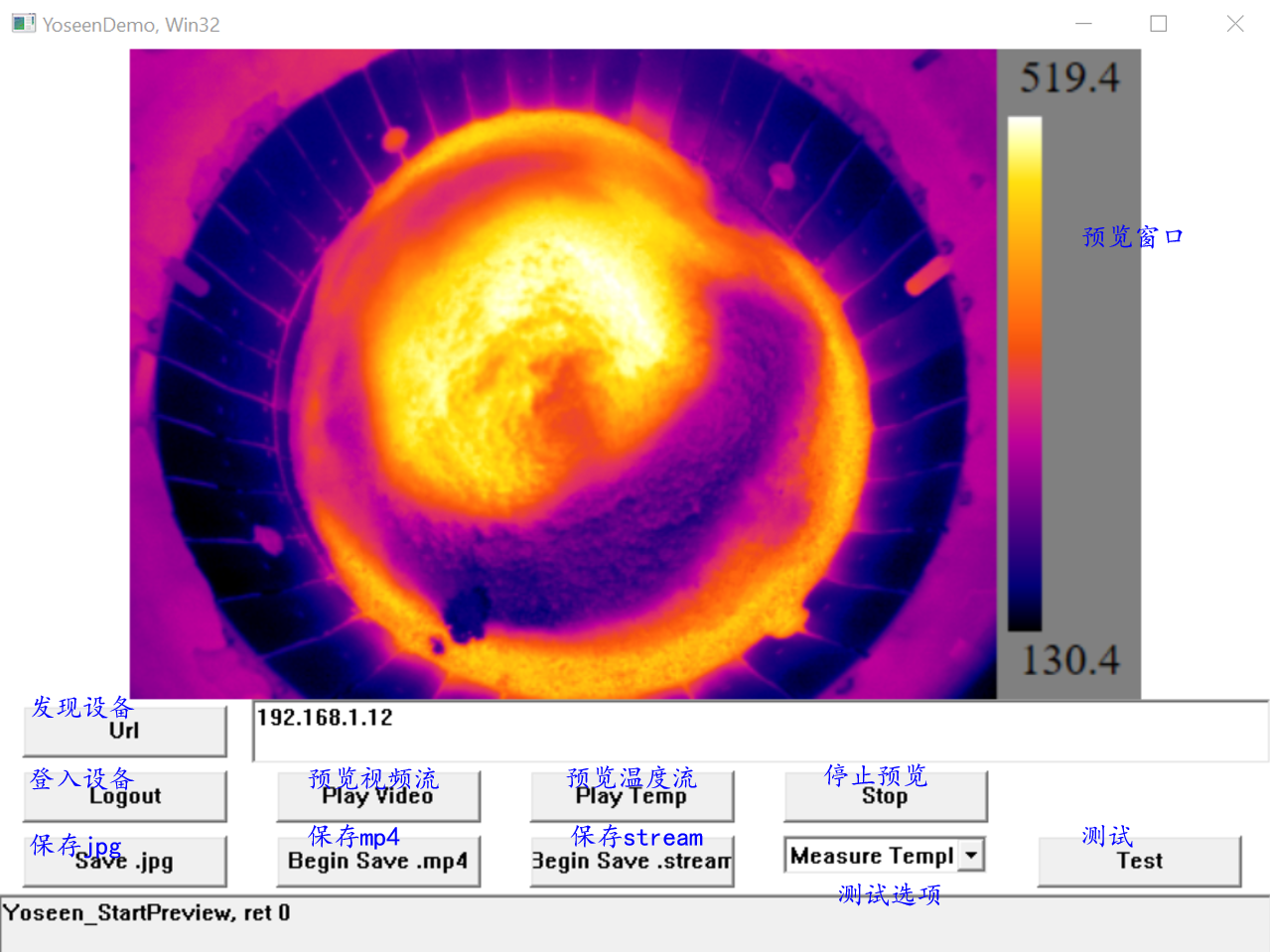


图3.2 YoseenDemo界面说明

# 函数说明

## SDK初始化

### 建立SDK资源

函 数：s32 Yoseen\_InitSDK()

参 数：无

返回值：0 成功, x失败

说 明：开始使用SDK前调用。

### 释放SDK资源

函 数：void Yoseen\_FreeSDK()

参 数：无

返回值：0 成功, x失败

说 明：结束使用SDK后调用。

## 用户注册

### 登入热像仪

函 数：s32 Yoseen\_Login(const YoseenLoginInfo\* loginInfo, CameraBasicInfo\* cameraBasicInfo)

参 数：loginInfo 登入信息

cameraBasicInfo 热像仪基本信息

返回值：>=0用户句柄, <0 错误码

说 明：成功登入会填充热像仪基本信息。目前只验证热像仪地址，不验证用户名和密码；热像仪不限制登入的用户。

### 登出热像仪

函 数：s32 Yoseen\_Logout(s32 userHandle)

参 数：无

返回值：0 成功, x失败

说 明：无

## 数据预览

### 开始预览

函 数：s32 Yoseen\_StartPreview(s32 userHandle, YoseenPreviewInfo\* previewInfo)

参 数：userHandle 用户句柄

previewInfo 预览信息

返回值：>=0 预览句柄, <0 错误码

说 明：一台热像仪能同时支持 "1对16的视频流预览" 和 "1对1的温度流" 预览。

视频流预览, 后端接收到的是H264编码的图像数据和特征温度数据, 带宽要求低.

温度流预览, 后端接收到的是全帧温度数据, 带宽要求高. 用户可设置预览信息中窗口为空，只取数据。预览信息如下：

typedef struct \_YoseenPreviewInfo{

s32 DataType; ///< xxxdatatype\_video 视频流, xxxdatatype\_temp 温度流

void\* Hwnd; ///< 显示窗口句柄, 可以为空

YoseenPreviewCallback CustomCallback; ///< 预览回调, 可以为空

void\* CustomData; ///< 用户数据

/\*

用户不用设置, 预览成功后, 用户据此调整显示窗口宽高比

\*/

u16 OutputWidth; ///< 显示画面宽度

u16 OutputHeight; ///< 显示画面高度

}YoseenPreviewInfo;

### 停止预览

函 数：s32 Yoseen\_StopPreview(s32 previewHandle)

参 数：previewHandle 预览句柄

返回值：0 成功, x失败

说 明：

### 暂停预览

函 数：s32 Yoseen\_PausePreview(s32 previewHandle, s32 pause)

参 数：previewHandle 预览句柄

pause 0-非暂停, 1-暂停

返回值：0 成功, x失败

说 明：暂停预览，数据传输不暂停, 预览回调函数不触发，窗口画面不更新.

### 开始保存

函 数：s32 Yoseen\_BeginSave(s32 previewHandle, const char\* fn, s32 fileType)

参 数：previewHandle 预览句柄

fn 文件名

filetype 文件类型（多帧温度流xxxmediafile\_stream，视频xxxmediafile\_mp4）

返回值：0 成功, x失败

说 明：开始保存, 温度流预览支持保存.stream和.mp4文件, 视频流预览支持保存.mp4文件.

### 停止保存

函 数：s32 Yoseen\_EndSave(s32 previewHandle, s32 fileType)

参 数：previewHandle 预览句柄

fileType 文件类型

返回值：0 成功, x失败

说 明：

### 设置预览图像算法信息

函 数：s32 s32 Yoseen\_PreviewSetImage(s32 previewHandle, const strech\_control\* strechControl, s32 paletteType)

参 数：previewHandle 预览句柄

stretchControl 图像算法参数

paletteType 调色板类型

返回值：0 成功, x失败

说 明：设置温度流预览, 温度转位图算法参数；视频流预览使用Yoseen\_SetTvoutInfo.

### 设置预览测温对象信息

函 数：s32 Yoseen\_PreviewSetMeasure(s32 previewHandle, const MeasureInfo\* measureInfo)

参 数：previewHandle 预览句柄

measureInfo 测温对象信息

返回值：0 成功, x失败

说 明：此接口不再支持。

### 保存单帧到文件

函 数：s32 Yoseen\_SaveFrame(s32 userHandle, const char\* fn, s32 withTemp)

参 数：userHandle 用户句柄

fn 文件名

withTemp 是否带温度，0不带，1带

返回值：0 成功, x失败

说 明： 保存单帧温度到文件(jpg扩展格式), 不需要预览的支持. 热像仪支持同一时刻处理16个保存请求,降低了在同一时刻请求过多时, 后面请求失败的几率.

### 保存单帧到内存

函 数：s32 Yoseen\_SaveFrameToMem(s32 userHandle, TempFrameFile\* frameFile)

参 数：userHandle 用户句柄

frameFile 文件数据

返回值：0 成功, x失败

说 明：保存单帧温度到内存, 用户提供frameFile, SDK填充frameFile的各个数据指针.

frameFile的各个数据指针指向的都是内部数据缓冲区, 对同一userHandle的保存请求会更新此内部数据缓冲区的内容.

## 发现热像仪

函 数：DiscoverCameraResp2\* Yoseen\_DiscoverCameras(s32 discoverFlags)

参 数：discoverFlags 发现方式，0x01广播, 0x02组播, 0x04 广播2

返回值：成功返回链表，失败返回空

说 明：在不知道热像仪IP的时候，执行发现热像仪，获取热像仪IP。使用IP登入热像仪，就可以对热像仪进行预览数据、配置参数、发送控制命令等操作。建议使用Yoseen\_SetCameraNetworkInfo配置热像仪使用固定IP。请使用Yoseen\_DiscoverCamerasFree释放链表。

## 参数配置

### 获取热像仪基本信息

函 数：s32 Yoseen\_GetCameraBasicInfo(s32 userHandle, CameraBasicInfo\* cameraBasicInfo)

参 数：userHandle 用户句柄

cameraBasicInfo 热像仪基本信息

返回值：0成功，x失败

说 明：热像仪基本信息如下

typedef struct \_CameraBasicInfo{

char CameraId[32]; ///< 序列号

char CameraName[32]; ///< 名称

char CameraType[16]; ///< 类型

char FPAId[32]; ///< 探测器序列号

char HardwareId[32]; ///< 硬件版本

char SoftwareId[32]; ///< 软件版本

u16 DataWidth; ///< 数据宽度

u16 DataHeight; ///< 数据高度

/\*\*数据帧率

网络温度数据帧率=DataFps/(DataRatio\*SendRatio)

网络视频数据帧率=DataFps/DataRatio

模拟视频帧率=DataFps/DataRatio

\*/

u16 DataFps;

u16 DataRatio; ///< 数据率, 设备每隔多少帧取一帧数据

u16 SendRatio; ///< 发送率, 设备每隔多少帧发一帧数据

u8 DataTransform; ///< 数据变换类型

u8 reserved[29];

}CameraBasicInfo;

### 设置热像仪基本信息

函 数：s32 Yoseen\_SetCameraBasicInfo(s32 userHandle, const CameraBasicInfo\* cameraBasicInfo)

参 数：userHandle 用户句柄

cameraBasicInfo 热像仪基本信息

返回值：0成功，x失败

说 明：

### 获取热像仪网络信息

函 数：s32 Yoseen\_GetCameraNetworkInfo(s32 userHandle, CameraNetworkInfo\* cameraNetworkInfo)

参 数：userHandle 用户句柄

cameraNetworkInfo 热像仪网络信息

返回值：0成功，x失败

说 明：热像仪网络信息如下

typedef struct \_CameraNetworkInfo{

bool UseStaticIp; ///< 是否使用静态IP

u8 pad;

u16 MulticastPort; ///< 组播端口, 未使用

u32 StaticIp; ///< 静态IP

u32 SubnetMask; ///< 子网掩码

u32 Gateway; ///< 网关

u32 MulticastIp; ///< 组播IP, 未使用

u8 MacAddr[6]; ///< MAC地址

u8 pad2[2];

u32 Dns; ///< 主DNS

u32 Dns2; ///< 次DNS

u8 reserved[16];

}CameraNetworkInfo;

### 设置热像仪网络信息

函 数：s32 Yoseen\_SetCameraNetworkInfo(s32 userHandle, const CameraNetworkInfo\* cameraNetworkInfo)

参 数：userHandle 用户句柄

cameraNetworkInfo 热像仪网络信息

返回值：0成功

说 明：设置热像仪网络信息, 成功设置后, 热像仪IP可能发生变化, 请重新发现热像仪。若配置成使用静态IP, 但是忘记IP, 请使用热像仪的GPIO复位热像仪。

### 获取挡板校零信息

函 数：s32 Yoseen\_GetFFCInfo(s32 userHandle, FFCInfo\* ffcInfo)

参 数：userHandle 用户句柄

ffcInfo 挡板校零信息

返回值：0成功，x失败

说 明：挡板校零信息如下

typedef struct \_FFCInfo{

u8 frames\_skipped\_after\_close; ///< 挡板关闭后忽略帧数

u8 frames\_accumlated\_when\_closed; ///< 挡板关闭后累积帧数

u8 frames\_skipped\_after\_open; ///< 挡板打开后忽略帧数

u8 pad;

u16 time\_interval; ///< 时间触发间隔, 单位帧数

u16 temp\_interval; ///< 温度触发间隔, 单位mK

s8 af\_rebound; ///< 自动对焦回弹, 单位帧数

u8 reserved[7];

}FFCInfo;

### 设置挡板校零信息

函 数：s32 Yoseen\_SetFFCInfo(s32 userHandle, const FFCInfo\* ffcInfo)

参 数：userHandle 用户句柄

ffcInfo 挡板校零信息

返回值：0成功，x失败

说 明：设置挡板校零信息, 设置成功后, 会触发一次挡板校零.

### 获取模拟视频信息

函 数：s32 Yoseen\_GetTvoutInfo(s32 userHandle, TvoutInfo\* tvoutInfo)

参 数：userHandle 用户句柄

tvoutInfo 模拟视频信息

返回值：0成功，x失败

说 明：模拟视频信息如下

typedef struct \_TvoutInfo{

bool EnableTvout; ///< 是否开启模拟视频

u8 PaletteType; ///< 调色板类型

u8 pad;

u8 Contrast; ///< 对比度

u8 Brightness; ///< 亮度

u8 Zoom; ///< 数字变焦

u16 XuiDisplayFlags; ///< 显示标志位, 参见XuiDisplayFlags

float Gain; ///< 增益

u32 H264\_Bitrate; ///< 流媒体码率

u16 H264\_GopSize; ///< 流媒体图像组大小

u8 reserved[22];

}TvoutInfo;

enum XuiDisplayFlags{

XDF\_Palette = 0x0001, ///< 显示调色板

XDF\_TrackHigh = 0x0002, ///< 显示高温追踪

XDF\_TrackLow = 0x0004, ///< 显示低温追踪

XDF\_NoGlobalMax = 0x0008, ///< 不显示全局最高温

XDF\_NoGlobalMin = 0x0010, ///< 不显示全局最低温

};

### 设置模拟视频信息

函 数：s32 Yoseen\_SetTvoutInfo(s32 userHandle, const TvoutInfo\* tvoutInfo)

参 数：userHandle 用户句柄

tvoutInfo 模拟视频信息

返回值：0成功，x失败

说 明：设置视频流画面图像参数、流媒体编码参数。为保证画面上文字清晰，分辨率低于640x480的图像放大到640x480去显示，文字自动切换黑白颜色。流媒体全码率是图像宽度\*图像高度\*帧率\*8，流媒体码率上限为全码率的1/10，下限为全码率的1/50；流媒体图像组大小上限为帧率的5倍，下限为帧率的1/5。

### 获取测温修正信息

函 数：s32 Yoseen\_GetFixInfo(s32 userHandle, FixInfo\* fixInfo)

参 数：userHandle 用户句柄

fixInfo 测温修正信息

返回值：0成功，x失败

说 明：测温修正信息如下

typedef struct \_FixInfo{

float AtmosphericTemperature; ///< 气温

float RelativeHumidity; ///< 相对湿度

float Visibility; ///< 能见度

float RainfallIntensity; ///< 降雨强度

float SnowfallIntensity; ///< 降雪强度

float TargetDistance; ///< 目标距离

float GlobalEmissivity; ///< 全局发射率

float InfraredWindowTrans; ///< 红外窗口透过率

float TempOffset; ///< 温度漂移

bool EnableAtmFix; ///< 是否开大气修正

u8 reserved[35];

}FixInfo;

### 设置测温修正信息

函 数：s32 Yoseen\_SetFixInfo(s32 userHandle, const FixInfo\* fixInfo)

参 数：userHandle 用户句柄

fixInfo 测温修正信息

返回值：0成功，x失败

说 明：热像仪支持测温修正，修正全像面每点温度。

### 获取GPIO信息

函 数：s32 Yoseen\_GetGpioInfo(s32 userHandle, GpioInfo\* gpioInfo)

参 数：userHandle 用户句柄

gpioInfo GPIO信息

返回值：0成功，x失败

### 设置GPIO信息

函 数：s32 Yoseen\_SetGpioInfo(s32 userHandle, const GpioInfo\* gpioInfo)

参 数：userHandle 用户句柄

gpioInfo GPIO信息

返回值：0成功，x失败

### 获取串口信息

函 数：s32 Yoseen\_GetSerialPortInfo(s32 userHandle, SerialPortInfo\* serialPortInfo)

参 数：userHandle 用户句柄

serialPortInfo 串口信息

返回值：0成功，x失败

### 设置串口信息

函 数：s32 Yoseen\_SetSerialPortInfo(s32 userHandle, SerialPortInfo\* serialPortInfo)

参 数：userHandle 用户句柄

serialPortInfo 串口信息

返回值：0成功，x失败

### 获取双光混合输出信息

函 数：s32 Yoseen\_GetCameraOutputInfo(s32 userHandle, CameraOutputInfo\* cameraOutputInfo)

参 数：userHandle 用户句柄

cameraOutputInfo 双光混合信息

返回值：0成功，x失败

说 明：获取卡片机红外图像效果、可见光的裁剪、混合的方式等。

### 设置双光混合输出信息

函 数：s32 Yoseen\_SetCameraOutputInfo(s32 userHandle, const CameraOutputInfo\* cameraOutputInfo)

参 数：userHandle 用户句柄

cameraOutputInfo 双光混合信息

返回值：0成功，x失败

说 明：设置卡片机红外图像效果、可见光的裁剪、混合的方式等。

## 控制命令

### 发送控制信息

函 数：s32 Yoseen\_SendControl(s32 userHandle, const Ctl\* ctl)

参 数：userHandle 用户句柄

ctl 控制信息

返回值：0成功，x失败

说 明：用于实现自动对焦、手动挡板校零、恢复出厂配置、重启等。控制信息如下

typedef struct \_Ctl{

u16 Type; ///< 类型

union {

u8 reserved[8]; ///< 联合体大小

u8 DataType; ///< 温度数据类型

u8 ShutterState; ///< 0挡板开, 1挡板关

s16 FocusDelta; ///< 电动镜头电机转动时间, 单位5ms, 正数-远焦, 负数-近焦

xxxfocusrect FocusRect; ///< 自动对焦区域

u8 FocusType; ///< 电动镜头电机转动方式, 0 停止, 1 远焦转动, 2 近焦转动

xxxgpioalarm Alarm; ///< GPIO报警

u8 DisableFFC; ///< 1禁用, 0启用

}Data;

}Ctl;

### 发送控制信息X

函 数：s32 Yoseen\_SendControlX(s32 userHandle, const CtlX\* ctlx)

参 数：userHandle 用户句柄

ctlx 控制信息X

返回值：0成功，x失败

说 明：用于实现获取和设置热像仪时间、获取和设置热像仪测温档位、云台控制（获取和设置角度、预置位、手动）等。

### 发送串口数据

函 数：s32 Yoseen\_WriteSerialPortData(s32 userHandle, const xxxserialportdata\* spd)

参 数：userHandle 用户句柄

spd 串口数据

返回值：0成功，x失败

说 明：通过网络发送数据到设备，设备再发送到串口，用以实现云台控制。

## 设备维护

### 上传本地文件到热像仪

函 数：s32 Yoseen\_UploadFile(s32 userHandle, const char\* fn, s32 fileType)

参 数：userHandle 用户句柄

fn 文件名称

fileType 文件类型xxxcamerafile\_bin，xxxcamerafile\_bad

返回值：0成功，x失败

说 明：用于热像仪程序更新、坏点补偿等

### 下载热像仪文件到本地

函 数：s32 Yoseen\_DownloadFile(s32 userHandle, const char\* fn, s32 fileType)

参 数：userHandle 用户句柄

fn 文件名称

fileType 文件类型xxxcamerafile\_log

返回值：0成功，x失败

说 明：用于热像仪日志分析等

## 分配调色板数据

函 数：xxxpalettedata\* Yoseen\_AllocPaletteData(s32& count);

参 数：count 调色板数量

返回值：成功返回调色板数组，失败返回NULL

说 明：构造一份调色板数据，释放请使用Yoseen\_FreePaletteData

## 文件回放

### 建立回放上下文

函 数：YoseenPlaybackContext\* YoseenPlayback\_Create()

返回值： 成功返回回放上下文，失败返回NULL

说 明：回放上下文可打开单帧温度 （bmp\jpg\png）和多帧温度（stream）文件

### 释放回放上下文

函 数： void YoseenPlayback\_Free(YoseenPlaybackContext\*\* pp)

参 数：pp 回放上下文

说 明：关闭打开的文件，释放有关资源

### 打开文件

函 数：s32 YoseenPlayback\_OpenFile(YoseenPlaybackContext\* ctx, const char\* fileName, s32 fileType)

参 数：ctx 回放上下文

fileName 文件名

fileType 文件类型

返回值：0成功，x失败

说 明： 打开文件，支持单帧温度（bmp\jpg\png）和多帧温度（stream）

### 关闭文件

函 数：void YoseenPlayback\_CloseFile(YoseenPlaybackContext\* ctx)

参 数：ctx 回放上下文

说 明：关闭文件

### 保存文件

函 数：s32 YoseenPlayback\_SaveFile(YoseenPlaybackContext\* ctx, TempFrameFile\* tff, const char\* fileName)

参 数：ctx 回放上下文

tff 文件封面

fileType 文件类型

返回值：0成功，x失败

说 明：只支持保存单帧文件，封面数据用户自行生成。

### 读取封面数据

函 数：s32 YoseenPlayback\_ReadCover(YoseenPlaybackContext\* ctx, TempFrameFile\* tff)

参 数：ctx 回放上下文

tff 封面数据

返回值：0成功，x失败

说 明：封面数据指的是文件的图像封面和参数信息。

### 读取温度数据

函 数：s32 YoseenPlayback\_ReadFrame(YoseenPlaybackContext\* ctx, s32 frameIndex, DataFrame\* dataFrame);

参 数：ctx 回放上下文

frameIndex 帧索引，仅对流文件有效

dataFrame 温度帧

返回值：0成功，x失败

说 明：读取温度帧，读取触发位图数据重新生成，

### 获取图像参数

函 数：void YoseenPlayback\_GetImage(YoseenPlaybackContext\* ctx, strech\_control\* sc, s32\* paletteType)

参 数：ctx 回放上下文

sc 图像算法参数

paletteType 调色板

说 明： 获取当前图像算法和调色板参数。

### 设置图像参数

函 数：void YoseenPlayback\_SetImage(YoseenPlaybackContext\* ctx, strech\_control\* sc, s32\* paletteType)

参 数：ctx 回放上下文

sc 图像算法参数

paletteType 调色板

说 明： 设置当前图像算法和调色板参数；设置的参数在下次读取温度数据，生成图像时，生效。

## 测温模板

测温模板是由1个全局测温对象、8个全局屏蔽区、16个局部测温对象组成。**视频流数据**或**混合流数据**中H264帧头包含1个全局测温对象和16个局部测温对象的测量结果。**温度流数据**可使用SDK后端的测温模板接口，提取测量结果。

### 上传本地数据到设备

函 数：s32 Yoseen\_UploadMem(s32 userHandle, s32 memType, void\* memData, s32 memSize)

参 数：userHandle 用户句柄

memType 内存数据类型, 0测温模板

memData 内存数据

memSize 内存数据大小

返回值：0成功，x失败

说 明： 上传本地数据到设备，用于设置测温模板。

### 下载设备数据到本地

函 数：s32 Yoseen\_DownloadMem(s32 userHandle, s32 memType, void\* memData, s32& memSize)

参 数：userHandle 用户句柄

memType 内存数据类型, 0测温模板

memData 内存数据

memSize 内存数据大小

返回值：0成功，x失败

说 明： 下载设备数据到本地，用于获取测温模板。

### 获取测量结果

函 数：s32 mtGetResult(DataFrameHeader\* dfh, s16\* dfd, xxxmeasure\* measure, xxxresult\* result)

参 数：dfh 温度数据帧头

dfd 温度数据区

measure 测温对象,支持点、折线、多边形、线、矩形、椭圆

result 测量结果

说 明： 获取测量结果，无需测温模板上下文，对于多边形无优化。

### 建立测温模板上下文

函 数：MTContext\* mtCreate(s32 width, s32 height)

参 数：width 数据宽度

height 数据高度

返回值：NULL失败，x模板上下文

说 明： 建立测温模板上下文，对于屏蔽区和多边形有优化。

### 释放测温模板上下文

函 数：void mtFree(MTContext\*\* pp)

参 数：pp 测温模板上下文

说 明： 释放测温模板上下文。

### 重载测温模板

函 数：s32 mtReload(MTContext\* context, MeasureTemplate\* templ)

参 数：context 测温模板上下文

templ 测温模板

返回值：0成功，x失败

说 明： 重载测温模板，用于修改测温模板配置。

### 获取测温模板的测量结果

函 数：s32 mtCheck(MTContext\* context, DataFrameHeader\* dfh, s16\* dfd, MeasureTemplateResult\* result)

参 数：context 测温模板上下文

dfh 温度数据帧头

dfd 温度数据区

result 测量结果

返回值：0成功，x失败

说 明： 获取测量结果，对于屏蔽区和多边形有优化。

# 数据格式说明

## 温度流单帧数据

用户在预览回调中得到的温度数据帧dataFrame的各部分数据如下：

dataFrame->Head，温度数据帧头，128个字节；

dataFrame->Temp，温度数据区，pixels个short数，左上角为原点；

dataFrame->Bmp，位图数据区, pixels个bgra数，在后端由温度数据根据算法生成的位图。

温度数据帧头定义如下

typedef struct \_DataFrameHeader{

u16 Width; ///< 宽度

u16 Height; ///< 高度

u32 ComSize; ///< 压缩大小

u8 DataType; ///< 数据类型

u8 ComType; ///< 压缩类型

u16 Index; ///< 帧索引

/\*\*

全帧温度以16位有符号整数数组表示, 温度浮点值=温度整数值/Slope+Offset

\*/

u16 Slope;

s16 Offset;

s32 FPATemp; ///< 探测器温度, 内部使用

s32 ShellTemp; ///< 等效机壳温度, 内部使用

u8 pad;

u8 GpioInput0; ///< GPIO输入0

u8 GpioInput1; ///< GPIO输入1

u8 pad2[5];

s64 Timestamp; ///< 时间戳, 单位100ns

u8 reserved[88];

}DataFrameHeader;

## 视频流或双光混合流单帧数据

用户在预览回调中得到的视频数据帧dataFrame的各部分数据如下：

dataFrame->H264，H264数据帧头，1024个字节，含全局、中心、1到8个局部测温结果等;

dataFrame->Bmp，位图数据区, pixels个bgra数，在后端由ffmpeg解码生成的位图。

H264数据帧头V2定义如下（适用于前端固件版本大于20190901）：

typedef struct \_H264FrameHeaderV2 {

s32 Size; //帧大小

u16 Width; //数据宽度

u16 Height; //数据高度

float FPATemp; //探测器温度

s32 LocalResultFlags; //局部测温结果标志

xxxresult GlobalResult; //全局测温结果

xxxresult LocalResultArray[16]; //局部测温结果

u8 Reserved[668];

}H264FrameHeaderV2;

H264数据帧头定义如下（适用于前端固件版本小于20190901）：

typedef struct \_H264FrameHeader{

s32 Size; //帧大小

u16 Width; //数据宽度

u16 Height; //数据高度

float FPATemp; //探测器温度

H264\_MeasureResult GlobalResult; //全局测温结果

H264\_MeasureResult CenterResult; //中心测温结果

H264\_MeasureResult LocalResults[8]; //局部测温结果

u8 Reserved[772];

}H264FrameHeader;

## 单帧温度数据jpg文件格式

我们通过在标准jpg文件尾部，添加私有数据块teMp，实现单帧温度和图像的混存。用户可从文件尾部读取png温度头，得到数据宽高，进一步得到整个私有数据块teMp。私有数据块teMp二进制布局如下:

|  |  |
| --- | --- |
| 私有数据块teMp二进制布局 | |
| 内容 | 大小（字节） |
| 数据块长度 | 4 |
| 数据块类型 | 4，teMp |
| 帧文件头 | 128 |
| 保留区 | 16\*1024 |
| 温度数据帧头 | 128 |
| 温度数据区 | Pixels\*2 |
| png温度头 | 32 |
| 数据块校验码 | 4 |

struct png\_temp\_header{

u16 width; ///< 宽度

u16 height; ///< 高度

s32 version; ///< 版本

u8 reserved[24];

};

struct png\_temp\_chunk{

s32 length; ///< 长度

s32 type; ///< 类型

frame\_file\_header ffh; ///< 帧文件头

u8 reserved[PNG\_TEMP\_CHUNK\_RESERVED\_SIZE]; ///< 保留

DataFrameHeader dfh; ///< 温度数据帧头

};

## 多帧温度数据stream文件格式

我们通过在标准bmp文件尾部，添加多帧温度数据，实现多帧温度的保存。用户可以从stream文件头部读取bmp文件头，得到数据宽高，进一步得到每帧温度数据。温度流文件二进制布局如下：

|  |  |
| --- | --- |
| 温度流文件二进制布局 | |
| 内容 | 大小（字节） |
| bmp文件头 | 64 |
| bmp数据区 | Pixels\*4 |
| 保留区，测温对象 | 1024 |
| 流文件头 | 128 |
| 第0帧帧头 | 128 |
| 第0帧数据区 | Pixels\*2 |
| 第1帧帧头 | 128 |
| 第1帧数据区 | Pixels\*2 |
| 重复第x帧 |  |

typedef struct \_stream\_file\_header{

char camera\_id[32]; ///< 热像仪序列号

s64 captured\_time; ///< 拍摄时间

u16 width; ///< 数据宽度

u16 height; ///< 数据高度

FixInfo fix\_info; ///< 修正信息

u16 fps\_num; ///< 帧率分子

u16 fps\_den; ///< 帧率分母

u32 frame\_count; ///< 帧数

u8 palette; ///< 调色板

u8 pad;

s16 record\_ratio; ///< 录制率, >0 每隔多少帧保存一帧; <0 每隔多少秒保存一帧

}stream\_file\_header;

# 错误码说明

|  |  |  |
| --- | --- | --- |
| **错误名** | **错误值** | **说明** |
| **Eerror\_None** | 0 | 正常 |
| **EError\_InternalError** | -1 | 内部错误 |
| **EError\_NotSupported** | -2 | 不支持 |
| **EError\_NotImplemented** | -3 | 未实现 |
| **EError\_SDKUninited** | -4 | SDK未初始化 |
| **EError\_NoMemory** | -5 | 无内存 |
| **EError\_NoHandle** | -6 | 无句柄 |
| **EError\_InvalidState** | -7 | 无效状态 |
| **EError\_InvalidArgs** | -8 | 无效参数 |
| **EError\_InvalidResp** | -9 | 无效响应 |
| **EError\_InvalidHandle** | -10 | 无效句柄 |
| **EError\_InvalidData** | -11 | 无效数据 |
| **EError\_PreviewCreateBuffer** | -12 | 预览打开缓冲失败 |
| **EError\_PreviewOpenInput** | -13 | 预览打开输入失败 |
| **EError\_PreviewOpenDisplay** | -14 | 预览打开显示失败 |
| **EError\_PreviewRecoverBegin** | -17 | 预览自动恢复开始 |
| **EError\_PreviewRecoverEnd** | -18 | 预览自动恢复结束 |
|  | | |
| **EError\_SocketOpen** | -100 | Socket打开失败 |
| **EError\_SocketConn** | -99 | Socket连接失败 |
| **EError\_SocketSend** | -98 | Socket发送失败 |
| **EError\_SocketRecv** | -97 | Socket接收失败 |
| **EError\_SocketConfig** | -96 | Socket配置失败 |
| **EError\_SocketData** | -95 | Socket数据有错 |
|  | | |
| **EError\_FileOpen** | -200 | 文件打开失败 |
| **EError\_FileClose** | -199 | 文件关闭失败 |
| **EError\_FileRead** | -198 | 文件读失败 |
| **EError\_FileWrite** | -197 | 文件写失败 |
| **EError\_FileEOF** | -196 | 文件尾 |
| **EError\_FileType** | -195 | 文件类型有误 |
| **EError\_FileData** | -194 | 文件数据有误 |
|  | | |
| **EError\_FfmpegEncode** | -300 | Ffmpeg编码失败 |
| **EError\_FfmpegDecode** | -299 | ffmpeg 解码失败 |
| **EError\_FfmpegMuxOpen** | -298 | ffmpeg mux打开失败 |
| **EError\_FfmpegMuxWrite** | -297 | ffmpeg mux写失败 |
| **EError\_FfmpegMuxClose** | -296 | ffmpeg mux关闭失败 |
| **EError\_FfmpegDemuxOpen** | -295 | ffmpeg demux打开失败 |
| **EError\_FfmpegDemuxRead** | -294 | ffmpeg demux读失败 |
| **EError\_FfmpegDemuxClose** | -293 | ffmpeg demux打开失败 |