

SR Series SDK Programming Guide

V1.0

SR Series SDK Programming Guide

Version	Date	Author	Approved By	Remark
V1.0	Oct. 13 th , 2018			New

© 2018 Shenzhen Dianyong Technology Co., Ltd. All rights reserved.

Shenzhen Dianyong Technology Co., Ltd. CONFIDENTIAL: This document contains proprietary information of Shenzhen Dianyong Technology Co., Ltd. and is not to be disclosed or used without the prior written permission of Shenzhen Dianyong Technology Co., Ltd.

Due to update and improvement of Shenzhen Dianyong Technology Co., Ltd. products and technologies, information in this document is subjected to change without notice.

Contents

1.1	Overview	4
2	Function Call Sequence	5
2.1	SDK Basic Call Process	5
2.1.1	Data Preview	6
2.1.2	Parameter Configuration	6
2.1.3	Control Commander	6
2.1.4	Equipment maintenance	6
2.2	Data Preview Call Flow	7
2.3	Parameter Configuration or Control Command Flow	7
3	Function Call Example	9
3.1	YoseenDemo project description	9
3.2	Demo Interface	9
4	Function Introduction	15
4.1	SDK Initialization	15
4.1.1	Create SDK resources	15
4.1.2	Release SDK resources	15
4.2	User Registration	15
4.2.1	Login to the camera	15
4.2.2	Log out of the camera	16
4.3	Data preview	16
4.3.1	Start preview	16
4.3.2	Stop preview	17
4.3.3	Pause preview	17
4.3.4	Start saving	17
4.3.5	Stop saving	18
4.3.6	Set preview image algorithm information	18
4.3.7	Set preview temperature measurement object information	18
4.3.8	Get the feature information of the current frame in the temperature stream or video	19
4.3.9	Save a single frame to a file	19
4.3.10	Save a single frame to memory	19
4.4	Discover camera	20
4.5	Parameter configuration	20
4.5.1	Get basic information about the camera	20
4.5.2	Set the basic information of the camera	21
4.5.3	Get camera network information	22
4.5.4	Set up the camera network information	22
4.5.5	Obtain the shutter calibration information	23
4.5.6	Set the shutter calibration information	23
4.5.7	Get analog video information	24
4.5.8	Set up analog video information	25
4.5.9	Get temperature measurement object information	26
4.5.10	Set temperature measurement object information	26
4.5.11	Get OSD information	27

4.5.12	Set OSD information	27
4.5.13	Get temperature correction information	28
4.5.14	Set temperature correction information	28
4.5.15	Get GPIO information.....	29
4.5.16	Set GPIO information	29
4.5.17	Get serial information	29
4.5.18	Set serial port information	29
4.6	Control commands	30
4.6.1	Send control information	30
4.6.2	Send control information X	30
4.7	Equipment maintenance.....	31
4.7.1	Upload local files to the camera	31
4.7.2	Download the camera file to the local.....	31
4.8	Temperature convert to bmp file algorithm	31
4.8.1	Establish temperature convert to bmp file algorithm	31
4.8.2	Release temperature convert bmp file algorithm	32
4.8.3	Temperature convert BGRA bitmap	32
4.8.4	Get algorithm parameters.....	32
4.8.5	Set algorithm parameters	32
4.9	Obtain temperature measurement results	33
4.10	Assign palette data.....	33
4.11	File parsing	34
4.11.1	open a file	34
4.11.2	Close file	34
4.11.3	Save document	34
5	File Format Description.....	35
5.1	Temperature stream single frame data.....	35
5.2	Video stream single frame data.....	36
5.3	Single frame temperature data png and jpg file formats	37
5.4	Multi-frame temperature data stream file format.....	38
6	Error code description	40

1.1 Overview

The SDK is based on the customized infrared thermal imager's network communication protocol and provides interfaces for infrared thermal imager discovery, configuration and system maintenance, temperature or video data preview, single or multi-frame data storage, and post-data analysis algorithms.

No.	Windows Platform	
1	YoseenSDK.dll	SDK dynamic link library
2	YoseenFfmpeg.dll	Ffmpeg codec, demultiplexing library
3	pthreadGC2.dll	Pthread library
4	YoseenSDKCS.dll	C# package library
5	YoseenSDK.h	Network interface header file
6	YoseenBasicTypes.h	Basic data type header file
7	YoseenTypes.h	Data type header file
8	YoseenSDK.lib	Network interface library file
9	YoseenAlg.h	Infrared algorithm header file
10	YoseenFile.h	File parsing header file

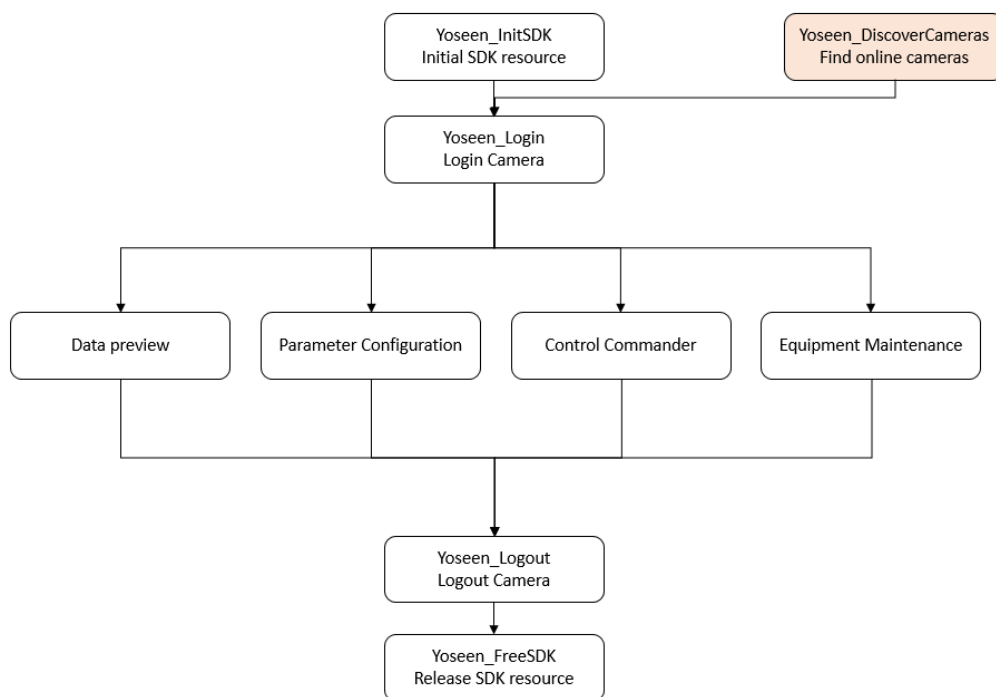
No.	Linux Platform	
1	libYoseenSDK.so	SDK dynamic link library
2	libYoseenFfmpeg.so	Ffmpeg codec, demultiplexing library
3	YoseenSDK.h	Network interface header file
4	YoseenBasicTypes.h	Basic data type header file
5	YoseenTypes.h	Data type header file
6	YoseenAlg.h	Infrared algorithm header file
7	YoseenFile.h	File parsing header file

The Linux version does not support the preview window handle (does not affect the preview data), and the rest is exactly the same as the Windows version. The Linux x64 is default version.

2 Function Call Sequence

2.1 SDK Basic Call Process

Basic call process is shown below,



Above is the mandatory processes to building SDK resources, logging in to the camera, logging out of the camera, and releasing SDK resources. The orange box to find that the online camera is an optional process.

- Create SDK resource **Yoseen_InitSDK**: Initialize network, log, display, ffmpeg, etc.
- Login to the camera **Yoseen_Login**: After logging in and waiting for the user handle, you can call various functions that require a user handle. The SDK supports 32 user handles and only verifies the camera IP address; the camera does not restrict the login user.
- Log out of the camera **Yoseen_Logout**: Release the user handle after logging out.
- Release the SDK resource **Yoseen_FreeSDK**: Reverse initialization network, log, display, ffmpeg, etc.

2.1.1 Data Preview

This is a temperature stream preview that transmits full frame temperature data and consumes a large bandwidth, supporting the transmission of one camera to one client. The SDK provides full frame temperature data and bitmap data for the user to process in the preview callback function.

The video stream preview transmits H.264 encoded video data and characteristic temperature data, and the bandwidth is low, and supports one camera to transmit to 16 clients. The SDK provides feature temperature data and bitmap data in the preview callback function for the user to process.

2.1.2 Parameter Configuration

Obtain and set basic information of the camera, network information, shutter calibration information, analog video information, temperature correction information, etc.

2.1.3 Control Commander

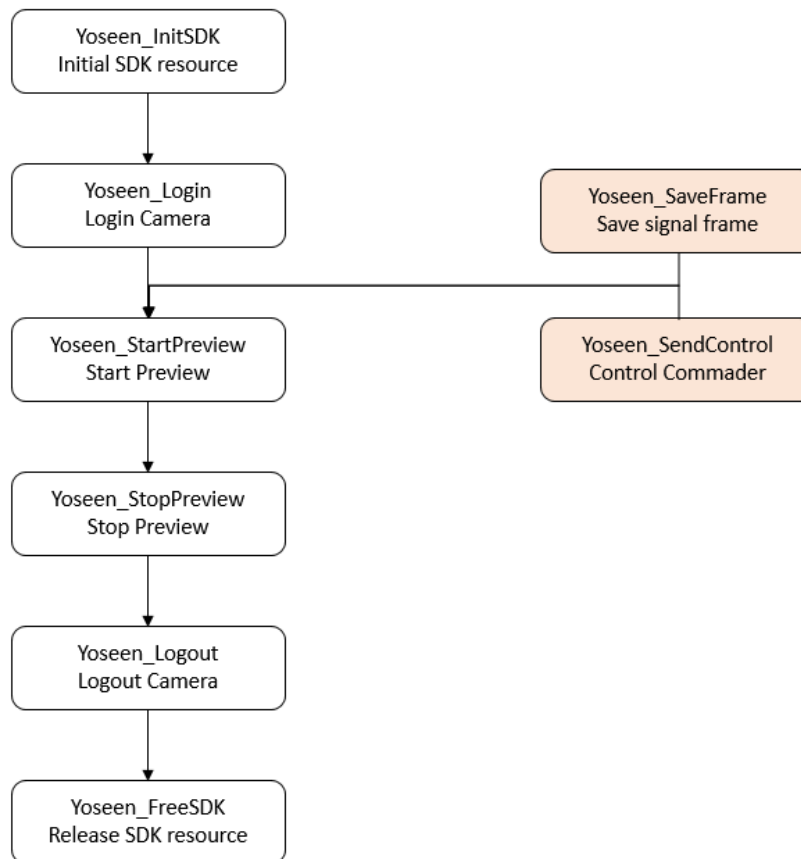
Manual shutter calibration, auto focus, get and set the current time of the camera, get and set the camera temperature gear position.

2.1.4 Equipment maintenance

Upload the camera update package, download the camera log file etc.

2.2 Data Preview Call Flow

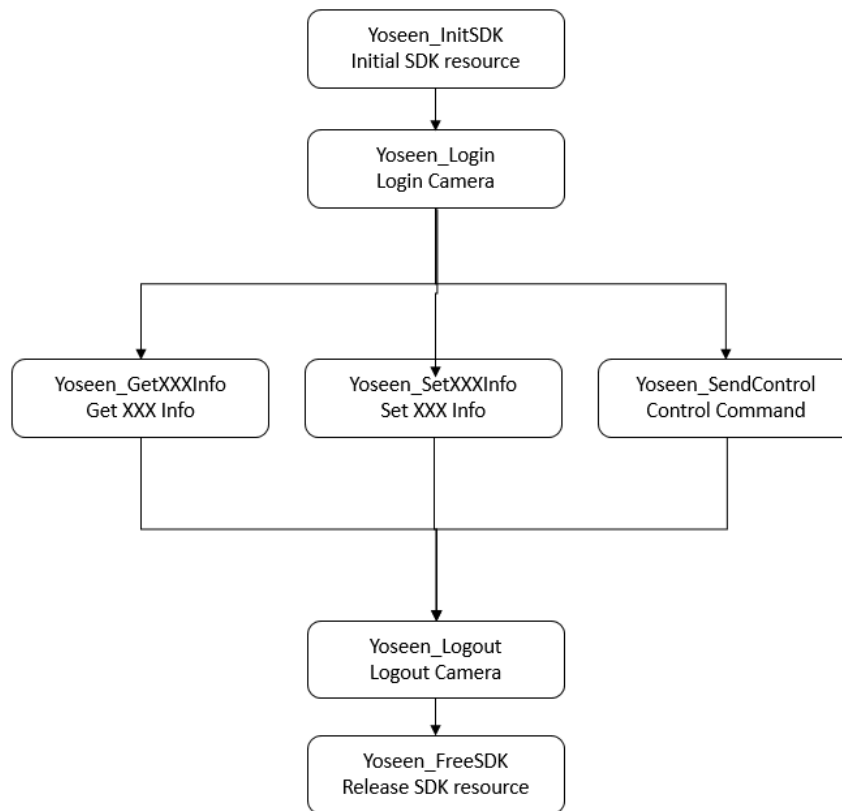
Below is data preview call flow process,



Above is a data preview process and the orange box is an optional process. The user can associate the window to display the screen; the preview callback can be set to process each frame of data. Configure the camera, send control commands, save a single frame jpg, and only need the user handle to use, regardless of whether the preview is turned on.

2.3 Parameter Configuration or Control Command Flow

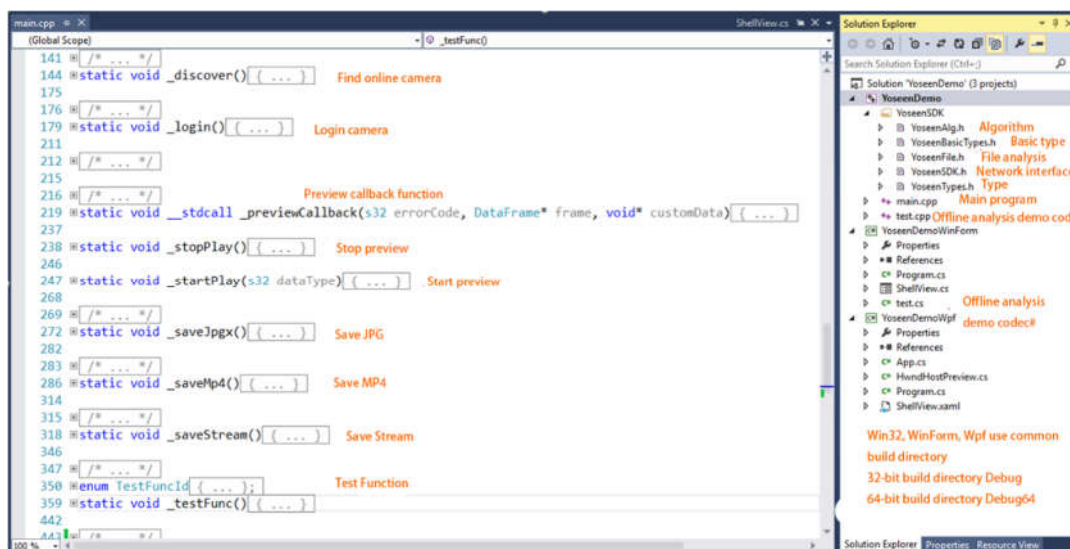
Below figure shows parameter configuration or control command flow



3 Function Call Example

3.1 YoseenDemo project description

YoseenDemo is developed using VS2013 and is divided into Win32, WinForm and Wpf projects. The three projects share the generated directory, the 32-bit generated directory is Debug, and the 64-bit generated directory is Debug64. The interface logic is exactly the same as the SDK call. The Win32 project code is described as follows,



The screenshot displays the Visual Studio 2013 IDE. The main window shows the `main.cpp` file with the following code:

```
141 // ...
144 static void _discover() { ... } Find online camera
175
176 // ...
179 static void _login() { ... } Login camera
211
212 // ...
215
216 // ...
219 static void _stdcall_previewCallback(s32 errorCode, DataFrame* frame, void* customData) { ... }
237
238 static void _stopPlay() { ... } Stop preview
246
247 static void _startPlay(s32 dataType) { ... } Start preview
268
269 // ...
272 static void _saveJpgx() { ... } Save JPG
282
283 // ...
286 static void _saveMp4() { ... } Save MP4
314
315 // ...
318 static void _saveStream() { ... } Save Stream
346
347 // ...
350 enum TestFuncId { ... };
359 static void _testFunc() { ... }
442
443 // ...
```

The Solution Explorer on the right shows the project structure for 'YoseenDemo' (3 projects):

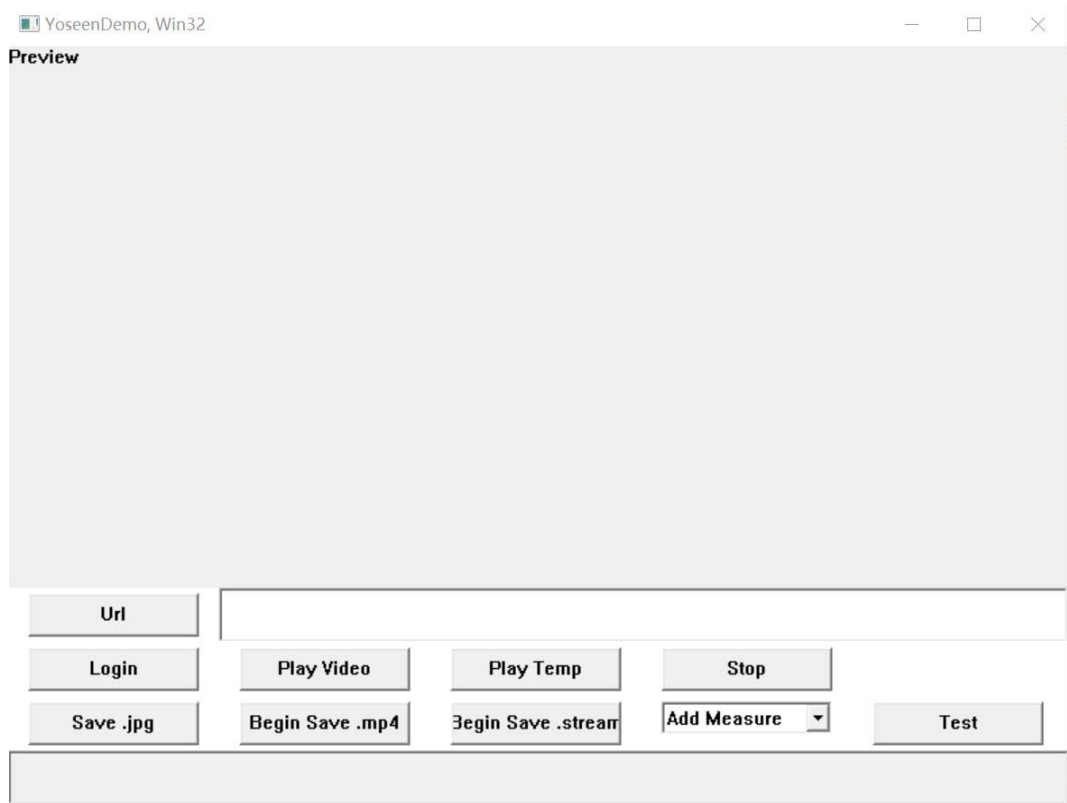
- YoseenSDK
 - YoseenAlg.h Algorithm
 - YoseenBasicTypes.h Basic type
 - YoseenFile.h File analysis
 - YoseenSDK.h Network interface
 - YoseenTypes.h Type
- main.cpp Main program
- test.cpp Offline analysis demo code
- YoseenDemoWinForm
 - Properties
 - References
 - Program.cs
 - ShellView.cs
 - test.cs
- YoseenDemoWpf
 - Properties
 - References
 - App.cs
 - HwndHostPreview.cs
 - Program.cs
 - ShellView.xaml

Additional information at the bottom of the Solution Explorer:

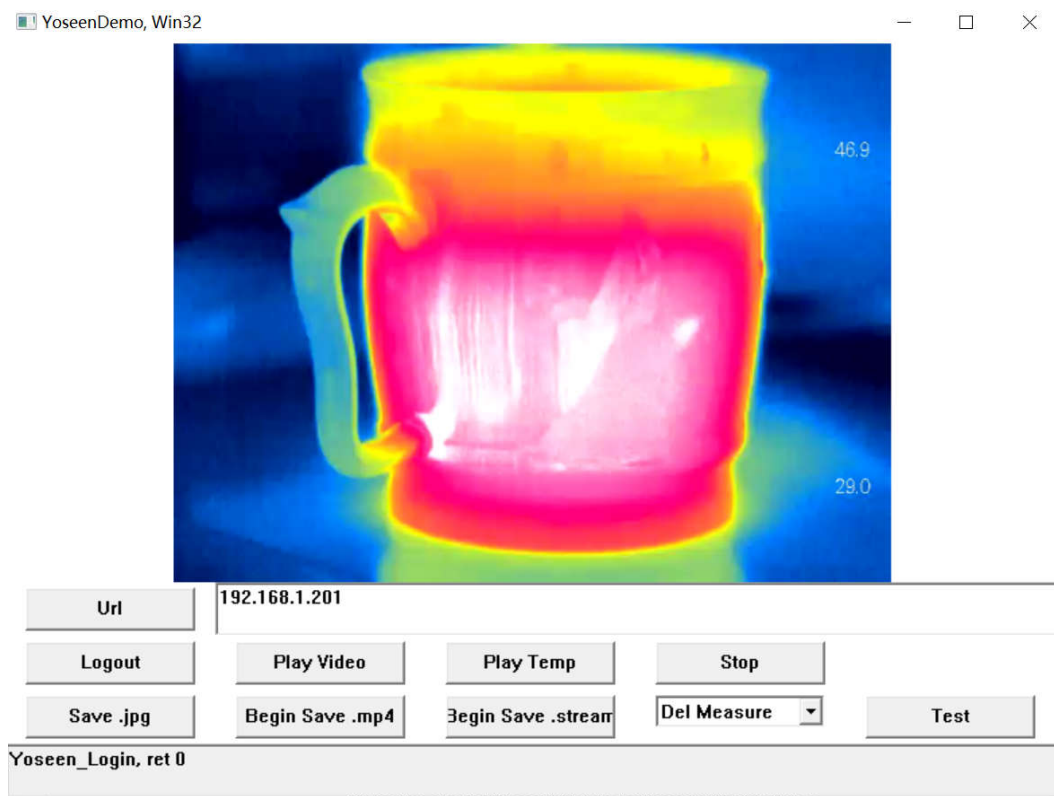
- Win32, WinForm, Wpf use common build directory
- 32-bit build directory Debug
- 64-bit build directory Debug64

3.2 Demo Interface

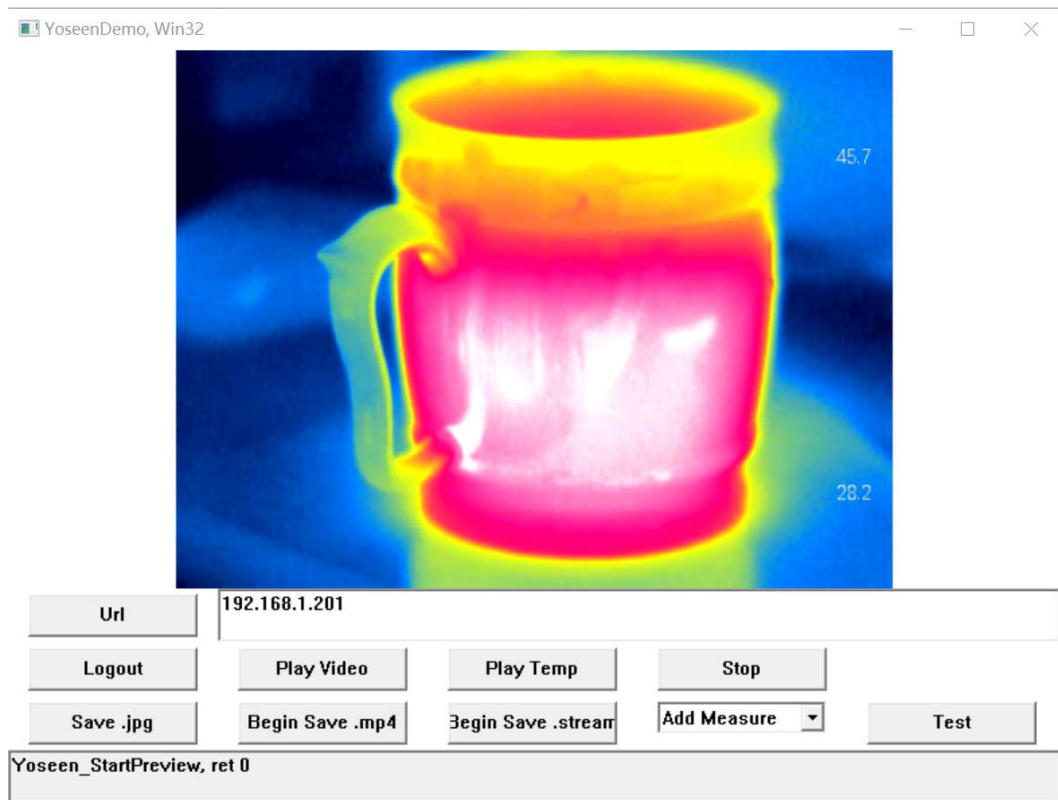
Below figure is demo interface,



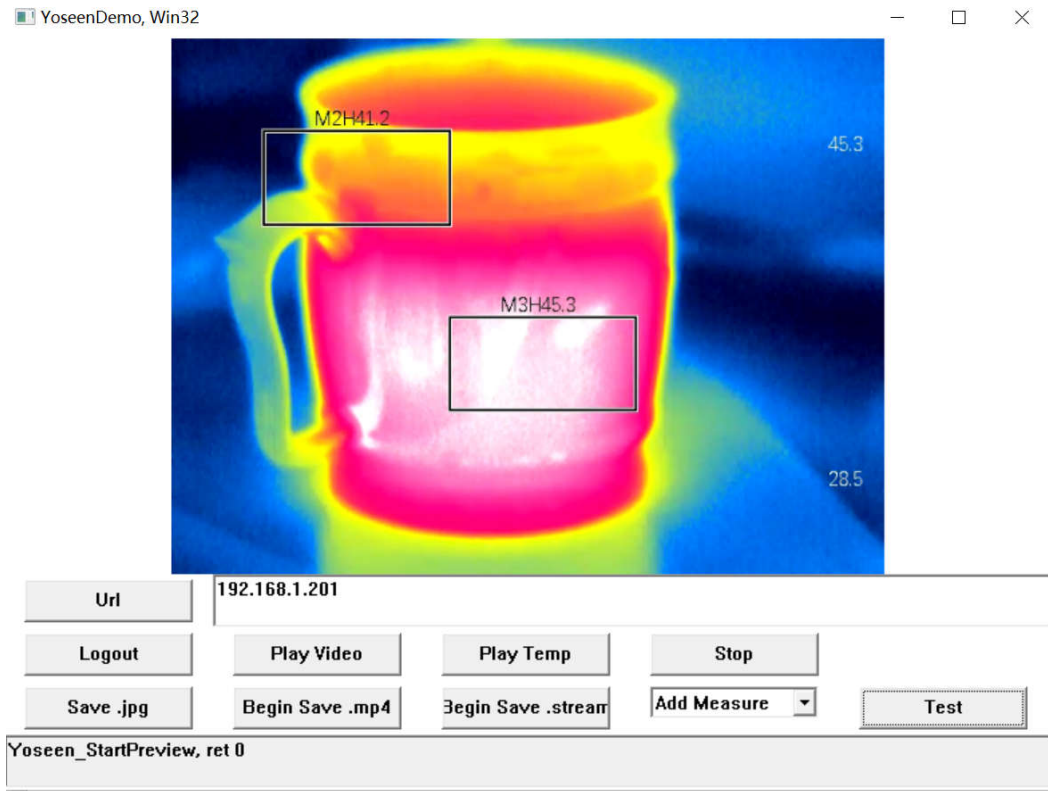
- To configure computer IP address with “192.168.1.xxx”, xxx is not the same as the camera IP address.
- Input camera IP address in the above “**Url**” input box, then press login to access the camera.



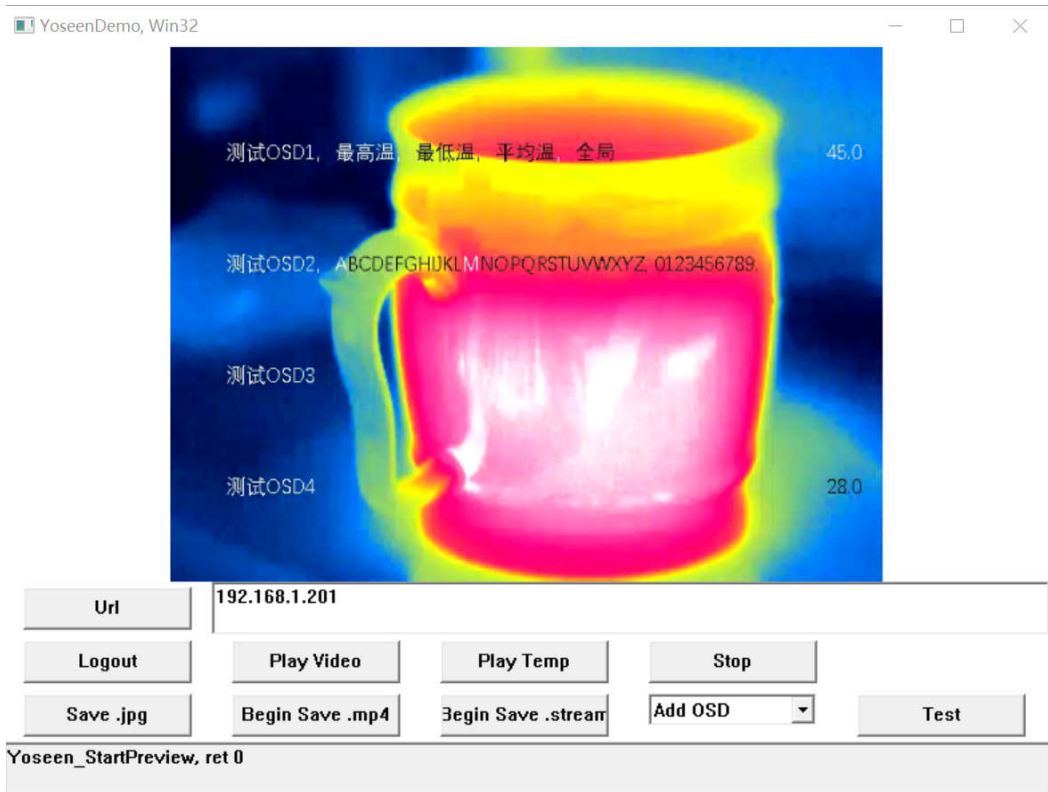
- **Play video** button shows video stream
- **Play temp** button shows video and temperature combination video.



- Choose **Add Measure** then press **Test** button will show the measure frame.
- Choose **Del Measure** then press **Test** button will delete the measure frame.



- Choose **Add OSD** then press **Test** button will show the OSD information.
- Choose **Del OSD** then press **Test** button will delete the OSD information.



4 Function Introduction

4.1 SDK Initialization

4.1.1 Create SDK resources

Function: s32 Yoseen_InitSDK()

Parameters: None

Return value: YET_None success, x failure

Description: Call before starting to use the SDK.

4.1.2 Release SDK resources

Function: void Yoseen_FreeSDK()

Parameters: None

Return value: YET_None success, x failure

Description: Called after ending the use of the SDK.

4.2 User Registration

4.2.1 Login to the camera

Function: s32 Yoseen_Login(const YoseenLoginInfo* loginInfo, CameraBasicInfo* cameraBasicInfo)

Parameter: loginInfo login information

cameraBasicInfo Camera Basic Information

Return value: >=0 user handle, <0 error code

Note: Successful login will fill in the basic information of the camera. Currently only the camera address is verified, the username and password are not verified; the camera does not restrict the user who logs in.

4.2.2 Log out of the camera

Function: s32 Yoseen_Logout(s32 userHandle)

Parameters: None

Return value: YET_None success, x failure

Description: None

4.3 Data preview

4.3.1 Start preview

Function: s32 Yoseen_StartPreview(s32 userHandle, YoseenPreviewInfo* previewInfo)

Parameter:	userHandle	user handle
	previewInfo	preview information

Return value: ≥ 0 preview handle, < 0 error code

Description: An infrared thermal camera can simultaneously support "1 to 16 video stream preview" and "1 to 1 temperature stream" preview. Video stream preview, the back end receives H.264 encoded image data and feature temperature data, it requires low bandwidth. For the temperature stream preview, the back end receives full frame temperature data, bandwidth requirements are high. Users can set the do not preview information window, only take data. The preview information is as follows:

```
typedef struct _YoseenPreviewInfo{
    s32 DataType;                ///< xxxdatatype_video video data,
    xxxdatatype_temp temperature data

    void* Hwnd;                  ///< display window handle, can be
    empty

    YoseenPreviewCallback CustomCallback; ///< preview callback, can be empty

    void* CustomData;            ///< use data

    /*
    The user does not need to set, after the preview is successful, the user adjusts
    the aspect ratio of the display window accordingly.
    */

    u16 OutputWidth;             ///< display width

    u16 OutputHeight;           ///< display height
}YoseenPreviewInfo;
```

4.3.2 Stop preview

Function: s32 Yoseen_StopPreview(s32 previewHandle)

Parameter: previewHandle preview handle

Return value: YET_None success, x failure

4.3.3 Pause preview

Function: s32 Yoseen_PausePreview(s32 previewHandle, s32 pause)

Parameters: previewHandle Preview handle pause 0-non-pause, 1-pause

Return value: YET_None success, x failure

Description: Pause preview, data transfer is not paused, preview callback function does not trigger, window screen is not updated.

4.3.4 Start saving

Function: s32 Yoseen_BeginSave(s32 previewHandle, const char* fn, s32 fileType)

Parameters: previewHandle Preview handle]
 fn File name
 filetype File type (multiframe temperature stream
 xxxmediafile_stream, video xxxmediafile_mp4)

Return value: YET_None success, x failure

Description: Start saving, temperature stream preview supports saving .stream and .mp4 files, video stream preview supports saving .mp4 files.

4.3.5 Stop saving

Function: s32 Yoseen_EndSave(s32 previewHandle, s32 fileType)

Parameter: previewHandle preview handle
 fileType file type

Return value: YET_None success, x failure

Description:

4.3.6 Set preview image algorithm information

Function: s32 s32 Yoseen_PreviewSetImage(s32 previewHandle, const stretch_control* stretchControl, s32 paletteType)

Parameters: previewHandle preview handle
 stretchControl image algorithm parameters
 paletteType palette type

Return value: YET_None success, x failure

Description: Set temperature flow preview, temperature index map algorithm parameters; video stream preview using Yosen_SetTvoutInfo.

4.3.7 Set preview temperature measurement object information

Function: s32 Yoseen_PreviewSetMeasure(s32 previewHandle, const MeasureInfo* measureInfo)

Parameters: previewHandle preview handle

measureInfo temperature object information

Return value: YET_None success, x failure

Description: Set temperature flow preview, temperature measurement object parameters; video stream preview using Yosen_SetMeasureInfo.

4.3.8 Get the feature information of the current frame in the temperature stream or video

Function: s32 Yoseen_PreviewGetResult(s32 previewHandle, const H264FrameHeader* result)

Parameters: previewHandle preview handle

result characteristic temperature information

Return value: YET_None success, x failure

Description: Acquire asynchronously the characteristic temperature information of the current frame in the temperature stream or video stream (global high and low temperature, center temperature, local high and low temperature). It is recommended to synchronously access the characteristic temperature information of the current frame in the preview callback function.

4.3.9 Save a single frame to a file

Function: s32 Yoseen_SaveFrame(s32 userHandle, const char* fn, s32 withTemp)

Parameter: userHandle User handle

fn File name withTemp Whether with temperature, 0 no, 1
yes

Return value: YET_None success, x failure

Description: Save single frame temperature to file (jpg extended format), no need for preview support. The camera supports 16 save requests at the same time, reducing the chance of subsequent requests failing at too many requests at the same time.

4.3.10 Save a single frame to memory

Function: s32 Yoseen_SaveFrameToMem(s32 userHandle, TempFrameFile* frameFile)

Parameter: userHandle user handle

frameFile file data Return value: YET_None Success, x failed

Description: Save the single frame temperature to the memory, the user provides the frameFile, the SDK fills the data pointer of the frameFile. The data pointer of the frameFile points to the internal data buffer, and the save request of the same userHandle will update the content of the internal data buffer.

4.4 Discover camera

Function: DiscoverCameraResp2* Yosen_DiscoverCameras(s32 discoverFlags)

Parameters: discoverFlags discovery mode, 0x01 broadcast, 0x02 multicast, 0x04 broadcast 2

Return value: return the linked list successfully, fail to return empty

Note: When you don't know the IP of the camera, you can find the camera and get the IP of the camera. By using the IP login camera, you can preview the camera, configure parameters, and send control commands to the camera. It is recommended to use the Yosen_SetCameraNetworkInfo to configure the camera to use a fixed IP. Please use the Yosen_DiscoverCamerasFree release list.

4.5 Parameter configuration

4.5.1 Get basic information about the camera

Function: s32 Yosen_GetCameraBasicInfo(s32 userHandle, CameraBasicInfo* cameraBasicInfo)

Parameter: userHandle User handle
 cameraBasicInfo Camera basic information

Return value: YET_None succeeds, x fails

Description: The basic information of the camera is as follows

```
typedef struct _CameraBasicInfo{  
    char CameraId[32];           ///< Serial number  
    char CameraName[32];        ///< name  
    char CameraType[16];        ///< type  
    char FPAId[32];             ///< FPGA serial number  
    char HardwareId[32];        ///< hardware version  
    char SoftwareId[32];        ///< software version  
    u16 DataWidth;              ///< data width  
    u16 DataHeight;             ///< data height  
  
    /**data frame rate  
    Network temperature data frame rate = DataFps / (DataRatio * SendRatio)  
    Network video data frame rate = DataFps / DataRatio  
    Analog video frame rate = DataFps / DataRatio  
    */  
    u16 DataFps;  
    u16 DataRatio;              ///< Send rate, the device takes one  
    frame of data how many frames.  
    u16 SendRatio;              ///< Data rate, the device sends a frame  
    of data how many frames  
    u8 DataTransform;           ///< Data transformation type  
    u8 reserved[29];  
}CameraBasicInfo;
```

4.5.2 Set the basic information of the camera

Function: s32 Yoseen_SetCameraBasicInfo(s32 userHandle, const CameraBasicInfo* cameraBasicInfo)

Parameter:	userHandle	User handle
	cameraBasicInfo	Camera basic information

Return value: YET_None succeeds, x fails

Description:

4.5.3 Get camera network information

Function: s32 Yoseen_GetCameraNetworkInfo(s32 userHandle, CameraNetworkInfo* cameraNetworkInfo)

Parameter: userHandle User handle
cameraNetworkInfo Camera network information

Return value: YET_None succeeds, x fails

Description: The camera network information is as follows

```
typedef struct _CameraNetworkInfo{  
    bool UseStaticIp;           ///< Whether use static IP address  
    u8 pad;  
    u16 MulticastPort;         ///< multicast port, not use  
    u32 StaticIp;              ///< static IP address  
    u32 SubnetMask;            ///< sub net  
    u32 Gateway;               ///< gateway  
    u32 MulticastIp;           ///< multicast IP address, not use  
    u8 MacAddr[6];            ///< MAC address  
    u8 pad2[2];  
    u32 Dns;                    ///< main DNS  
    u32 Dns2;                   ///< diversity DNS  
    u8 reserved[16];  
}CameraNetworkInfo;
```

4.5.4 Set up the camera network information

Function: s32 Yoseen_SetCameraNetworkInfo(s32 userHandle, const CameraNetworkInfo* cameraNetworkInfo)

Parameter: userHandle User handle
cameraNetworkInfo Camera network information

Return value: YET_None succeeded

Note: Set the camera network information. After successful setting, the camera IP may change, please rediscover the camera. If you are configuring to use a static IP, but forget the IP, use the GPIO of the camera to reset the camera.

4.5.5 Obtain the shutter calibration information

Function: s32 Yoseen_GetFFCInfo(s32 userHandle, FFCInfo* ffcInfo)

Parameter: userHandle User handle
 ffcInfo shutter calibration information

Return value: YET_None succeeds, x fails

Description: The shutter calibration information is as follows,

```
Typedef struct _FFCInfo{  
  
    u8 frames_skipped_after_close;            ///< Ignore frames after the  
    shutter is closed  
  
    u8 frames_accumlated_when_closed;        ///< Cumulative frames after the  
    shutter is closed  
  
    u8 frames_skipped_after_open;            ///< Ignore frames after the  
    shutter is opened  
  
    u8 pad;  
  
    u16 time_interval;                        ///< Time trigger interval, unit number of  
    frames  
  
    u16 temp_interval;                        ///< Temperature trigger interval, unit mK  
  
    s8 af_rebound;                            ///< AF rebound, unit number of frames  
  
    u8 reserved[7];  
  
}FFCInfo;
```

4.5.6 Set the shutter calibration information

Function: s32 Yoseen_SetFFCInfo(s32 userHandle, const FFCInfo* ffcInfo)

Parameters: userHandle user handle
 ffcInfo shutter calibration information

Return value: YET_None success, x failure

Description: Set the shutter calibration information. After the setting is successful, it will trigger the shutter calibration.

4.5.7 **Get analog video information**

Function: s32 Yoseen_GetTvoutInfo(s32 userHandle, TvoutInfo* tvoutInfo)

Parameters: userHandle user handle
 tvoutInfo analog video information

Return value: YET_None success, x failure

Description: The analog video information is as follows

```
typedef struct _TvoutInfo{
    bool EnableTvout;           ///< whether to enable analog video
    u8 PaletteType;            ///< color palette type
    u8 pad;
    u8 Contrast;               ///< contrast
    u8 Brightness;            ///< brightness
    u8 Zoom;                   ///< digital zoom
    u16 XuiDisplayFlags;       ///< display flag, for the detail please refer
    to XuiDisplayFlags
    float Gain;                ///< gain
    u32 H264_Bitrate;          ///< streaming code rate
    u16 H264_GopSize;          ///< streaming image group size
    u8 reserved[22];
}TvoutInfo;

enum XuiDisplayFlags{
    XDF_Palette = 0x0001,      ///< display color palette
    XDF_TrackHigh = 0x0002,    ///< shows highest temperature tracking
    XDF_TrackLow = 0x0004,     ///< shows lowest temperature tracking
    XDF_NoGlobalMax = 0x0008,  ///< does not show the highest global
    temperature
    XDF_NoGlobalMin = 0x0010,  ///< does not show the lowest global
    temperature
};
```

4.5.8 Set up analog video information

Function: s32 Yoseen_SetTvoutInfo(s32 userHandle, const TvoutInfo* tvoutInfo)

Parameters:	userHandle	user handle
	tvoutInfo	analog video information

Return value: YET_None success, x failure

Description: Set the video stream image parameters and streaming code parameters. To ensure clear text on the screen, images with a resolution lower than 640x480 are enlarged to 640x480 for display, and the text automatically switches between black and white. The full media rate of the streaming media = image width * image height * frame rate * 8, the upper limit of the streaming media rate is 1/10 of the full code rate, the lower limit is 1/50 of the full code rate; the upper limit of the streaming media image group size is the frame rate * 5, the lower limit is 1/5 of the frame rate.

4.5.9 Get temperature measurement object information

Function: s32 Yoseen_GetMeasureInfo(s32 userHandle, MeasureInfo* measureInfo)

Parameters: userHandle user handle
 measureInfo temperature measurement object information

Return value: YET_None success, x failure

Description: Temperature measurement object information is as follows

```
typedef struct _MeasureInfo{  
    s32 MOC;                                ///< Number of temperature measurement  
    objects  
    xxxmeasure_object MOS[8];            ///< Temperature measurement  
    object array  
}MeasureInfo;
```

4.5.10 Set temperature measurement object information

Function: s32 Yoseen_SetMeasureInfo(s32 userHandle, const MeasureInfo* measureInfo)

Parameters: userHandle user handle
 measureInfo temperature measurement object information

Return value: YET_None success, x failure

Description: Set the temperature measurement object parameters of the streaming media video.

4.5.11 Get OSD information

Function: s32 Yoseen_GetCameraOSDInfo(s32 userHandle, CameraOSDInfo* osdInfo)

Parameters: userHandle user handle
 osdInfo OSD information

Return value: YET_None success, x failure

Description: Get the streaming screen OSD information.

```
typedef struct _CameraOSDInfo{  
    u16 X1;                                ///< Coordinate X1  
    u16 Y1;                                ///< Coordinate Y1  
    char Text1[64];                        ///< Text 1, utf-8 coding  
    u16 X2;                                ///< Coordinate X2  
    u16 Y2;                                ///< Coordinate Y2  
    char Text2[64];                        ///< text2  
    u16 X3;                                ///< Coordinate X3  
    u16 Y3;                                ///< Coordinate Y3  
    char Text3[64];                        ///< text3  
    u16 X4;                                ///< Coordinate X4  
    u16 Y4;                                ///< Coordinate Y4  
    char Text4[64];                        ///< text 4  
}CameraOSDInfo;
```

4.5.12 Set OSD information

Function: s32 Yoseen_SetCameraOSDInfo(s32 userHandle, const CameraOSDInfo* osdInfo)

Parameters: userHandle user handle
 osdInfo OSD information

Return value: YET_None success, x failure

Description: Set the streaming screen OSD information.

4.5.13 Get temperature correction information

Function: s32 Yoseen_GetFixInfo(s32 userHandle, FixInfo* fixInfo)

Parameters: userHandle user handle
 fixInfo temperature correction information

Return value: YET_None success, x failure

Description: Temperature correction information is as follows,

```
typedef struct _FixInfo{  
    float AtmosphericTemperature;       ///< temperature  
    float RelativeHumidity;            ///< relative humidity  
    float Visibility;                  ///< visibility  
    float RainfallIntensity;          ///< rainfall intensity  
    float SnowfallIntensity;          ///< snowfall intensity  
    float TargetDistance;             ///< target distance  
    float GlobalEmissivity;           ///< global emissivity  
    float InfraredWindowTrans;        ///< IR window transmittance  
    float TempOffset;                 ///< temperature offset  
    bool EnableAtmFix;                ///< Whether to enable the  
    atmosphere correction  
    u8 reserved[35];  
}FixInfo;
```

4.5.14 Set temperature correction information

Function: s32 Yoseen_SetFixInfo(s32 userHandle, const FixInfo* fixInfo)

Parameters: userHandle user handle
 fixInfo temperature correction information

Return value: YET_None success, x failure

Description: The camera supports temperature correction and corrects the temperature of the full image.

4.5.15 Get GPIO information

Function: s32 Yoseen_GetGpioInfo(s32 userHandle, GpioInfo* gpioInfo)

Parameters:	userHandle	user handle
	gpioInfo	GPIO information

Return value: YET_None success, x failure

4.5.16 Set GPIO information

Function: s32 Yoseen_GetGpioInfo(s32 userHandle, const GpioInfo* gpioInfo)

Parameters:	userHandle	user handle
	gpioInfo	GPIO information

Return value: YET_None success, x failure

4.5.17 Get serial information

Function: s32 Yoseen_GetSerialPortInfo(s32 userHandle, SerialPortInfo* serialPortInfo)

Parameters:	userHandle	user handle
	serialPortInfo	serial information

Return value: YET_None success, x failure

4.5.18 Set serial port information

Function: s32 Yoseen_GetSerialPortInfo(s32 userHandle, SerialPortInfo* serialPortInfo)

Parameters:	userHandle	user handle
	serialPortInfo	serial information

Return value: YET_None success, x failure

4.6 Control commands

4.6.1 Send control information

Function: s32 Yoseen_SendControl(s32 userHandle, const Ctl* ctl)

Parameters: userHandle user handle
 Ctl control information

Return value: YET_None success, x failure

Description: auto focus, manual shutter calibration, restore factory configuration, etc. Control information is as follows,

```
typedef struct _Ctl{  
    u16 Type; ///< type  
    union {  
        u8 reserved[8];            ///< union size  
        u8 DataType;              ///< temperature data type  
        u8 ShutterState;         ///< 0 shutter open, 1 shutter close  
        s16 FocusDelta;          ///< Motorized lens motor rotation time, unit  
        is 5ms, positive number - far focus, negative number - close focus  
        xxxfocusrect FocusRect;     ///< autofocus area  
        u8 FocusType;             ///< Motorized lens motor rotation mode, 0  
        stop, 1 far focus rotation, 2 near focus rotation  
        xxxgpioalarm Alarm;        ///< GPIO alarm  
        u8 DisableFFC;            ///< 1 disable, 0 enable  
    }Data;  
}Ctl;
```

4.6.2 Send control information X

Function: s32 Yoseen_SendControlX(s32 userHandle, const CtlX* ctlx)

Parameters:	userHandle	user handle
	Ctrlx	control information X

Return value: YET_None success, x failure

Description: Used to realize the acquisition and setting of the camera time, acquisition and setting of the camera temperature measurement gear position.

4.7 Equipment maintenance

4.7.1 Upload local files to the camera

Function: s32 Yoseen_UploadFile(s32 userHandle, const char* fn, s32 fileType)

Parameters:	userHandle	user handle
	Fn	file name
	fileType	file type xxxcamerfile_bin, xxxcamerfile_bad

Return value: YET_None success, x failure

Description: For thermal imager program update, dead pixel compensation, etc.

4.7.2 Download the camera file to the local

Function: s32 Yoseen_DownloadFile(s32 userHandle, const char* fn, s32 fileType)

Parameters:	userHandle	user handle
	Fn	file name
	fileType	file type xxxcamerfile_log

Return value: YET_None success, x failure

Description: For thermal imager log analysis, etc.

4.8 Temperature convert to bmp file algorithm

4.8.1 Establish temperature convert to bmp file algorithm

Function: stretch_cfg* strCreate()

Return value: success will return algorithm context, fail return NULL.

Description: The temperature convert algorithm provides three types of algorithms: PHE, DDE, and LINEAR. Please use strFree for release.

4.8.2 Release temperature convert bmp file algorithm

Function: void strFree(strech_cfg* cfg)

Parameters: cfg algorithm context

Description: Release algorithm context

4.8.3 Temperature convert BGRA bitmap

Function: void strTemp2Bgra(strech_cfg* cfg, const s16* temp, const DataFrameHeader* Header, bgra* bmp, const bgra* palette)

Parameters: cfg algorithm context

 Temp temperature data area

 Header temperature data header

 Bmp bitmap data area

 Palette palette data area, palette data can be constructed with Yoseen_AllocPaletteData

Description: 16-bit temperature data convert to 32-bit bgra bitmap data

4.8.4 Get algorithm parameters

Function: void strGetCtl(strech_cfg* cfg, stretch_control* control)

Parameters: cfg algorithm context

 Control algorithm parameter

Description: Get algorithm parameters

4.8.5 Set algorithm parameters

Function: int strSetCtl(strech_cfg* cfg, const stretch_control* control)

Parameters: cfg algorithm context
 Control algorithm parameter

Return value: success 0, failure x

Description: can set the algorithm type, various algorithm parameters, contrast, brightness, etc.

4.9 Obtain temperature measurement results

Function:

```
void Alg_MeasurePoint(const xxxpoint* point, DataFrameHeader* dfh, s16* Dfd, xxxmeasure_result* result)
```

```
void Alg_MeasureLine(const xxxline* line, DataFrameHeader* dfh, s16* dfd, xxxmeasure_result* result)
```

```
void Alg_MeasureRectangle(const xxxrectangle* rectangle, DataFrameHeader* dfh, s16* dfd, xxxmeasure_result* result)
```

```
void Alg_MeasureEllipse(const xxxrectangle* ellipse, DataFrameHeader* dfh, s16* dfd, xxxmeasure_result* result)
```

Parameters:	point, line, rect, elli	temperature measurement object
	Dfh	data frame header
	Dfd	data area
	Result	temperature measurement result

Description: Obtain temperature measurement result of a temperature measurement object in single frame temperature data

4.10 Assign palette data

Function: xxxpalettedata* Yoseen_AllocPaletteData(s32& count);

Parameters: count number of palettes

Return value: successfully returned palette array, failed to return NULL

Description: Construct a palette data, release please use Yoseen_FreePaletteData

4.11 File parsing

4.11.1 open a file

Function: TempFrameFile* File_OpenFrame(const char* fileName, s32 fileType)

Parameters: fileName file name
 fileType file type

Return value: The file is returned successfully, and NULL is returned.

Description: Support bmp, png, stream, jpg files for temperature image combination, return file pointer contains temperature data of single frame file, cover image data, frame file header, memory pointer of user data area.

4.11.2 Close file

Function: void File_CloseFrame(TempFrameFile** pp)

Parameters: pp file pointer

Description: Releases the memory associated with the file pointer.

4.11.3 Save document

Function: s32 File_SaveFrame(TempFrameFile* frameFile, const char* fileName)

Parameters: frameFile file pointer
 fileName filename

Description: Saves the memory associated with the file pointer to the specified file. The user can modify the file information by modifying various types of memory contents associated with the file pointer.

5 File Format Description

5.1 Temperature stream single frame data

The temperature data frame obtained by the user in the preview callback data of each part of the dataFrame is as follows:

dataFrame->Head, temperature data frame header, 128 bytes;

dataFrame->Temp, temperature data area, pixels short number, upper left corner is the origin;

dataFrame->Bmp, bitmap data area, pixels bgra number, bitmap generated by the temperature data according to the algorithm at the back end.

The temperature data frame header is defined as follows,

```
typedef struct _DataFrameHeader{  
    u16 Width;                ///< width  
    u16 Height;              ///< height  
    u32 ComSize;             ///< compressed size  
    u8 DataType;            ///< data type  
    u8 ComType;              ///< zip type  
    u16 Index;               ///< frame index  
  
    /**  
    Full frame temperature is represented by a 16-bit signed integer array,  
    temperature floating point value = temperature integer value /Slope+Offset  
    */  
    u16 Slope;  
    s16 Offset;  
    s32 FPATemp;            ///< detector temperature, internal use  
    s32 ShellTemp;         ///< equivalent case temperature, internal use  
    u8 pad;  
    u8 GpioInput0;         ///< GPIO input 0  
    u8 GpioInput1;         ///< GPIO input 1  
    u8 pad2[5];  
    s64 Timestamp;        ///< time stamp, unit is 100ns  
    u8 reserved[88];  
}DataFrameHeader;
```

5.2 Video stream single frame data

The video data frame obtained by the user in the preview callback data of each part is as follows:

dataFrame->H264, H264 data frame header, 1024 bytes, including global, center, 1 to 8 local temperature measurement results, etc.;

dataFrame->Bmp, bitmap data area, pixels bgra number, bitmap generated by ffmpeg decoding at the back end.

The H264 data frame header is defined as follows:

```
typedef struct _H264FrameHeader{
    s32 Size;                //frame size
    u16 Width;              //data width
    u16 Height;            //data height
    float FPATemp;         //detector temperature
    H264_MeasureResult GlobalResult; //global temperature measurement result
    H264_MeasureResult CenterResult; //central temperature measurement result
    H264_MeasureResult LocalResults[8]; //partial temperature measurement result
    u8 Reserved[772];
}H264FrameHeader;
```

5.3 Single frame temperature data png and jpg file formats

We add a single frame temperature and image combination by adding a private block teMp at the end of the standard png or jpg file. The user can read the png temperature header from the end of the file to get the data width and height, and further get the entire private data block teMp. The private data block teMp binary layout is as follows:

Private data block teMp binary description	
Content	Size (bytes)
Data block length	4
Block type	4, teMp
Frame file header	128
Reserved area	16*1024
Temperature data frame header	128
Temperature data area	Pixels*2
Png file temperature head	32
Data block check code	4

```

/**
png  temperature head
*/

struct png_temp_header{

    u16 width; ///< width

    u16 height; ///< height

    s32 version; ///< version

    u8 reserved[24];

};

/**
png  temperature expand module
*/

struct png_temp_chunk{

    s32 length; ///< length

    s32 type; ///< type

    frame_file_header ffh; ///< frame file head

    u8 reserved[PNG_TEMP_CHUNK_RESERVED_SIZE]; ///< reserve

    DataFrameHeader dfh; ///< temperature data frame head

};

```

5.4 Multi-frame temperature data stream file format

We save multi-frame temperature by adding multi-frame temperature data at the end of the standard bmp file. The user can read the bmp file header from the head of the stream file to get the data width and height, and further obtain the temperature data of each frame. The binary layout of the temperature stream file is as follows:

Temperature stream file binary layout	
Content	Size (bytes)
Bmp file header	64
Bmp data area	Pixels*4
Reserved area, temperature measuring object	1024

Temperature stream file binary layout	
Stream file header	128
Frame 0 of the frame	128
0th frame data area	Pixels*2
Frame 1 frame header	128
Frame 1 data area	Pixels*2
Repeat the xth frame	

```

typedef struct _stream_file_header{
    char camera_id[32];           ///< Camera serial number
    s64 captured_time;           ///< shooting time
    u16 width;                   ///< data width
    u16 height;                  ///< data height
    FixInfo fix_info;           ///< correction information
    u16 fps_num;                 ///< frame rate numerator
    u16 fps_den;                 ///< frame rate denominator
    u32 frame_count;            ///< frame number
    u8 palette;                  ///< color palette
    u8 pad;
    s16 record_ratio;           ///< recording rate, >0 saves one frame
                                every how many frames; <0 saves one frame every how many seconds
}stream_file_header;

```

6 Error code description

Error Name	Error code	Description
YET_None	0	normal
YET_Undefined	-1	Undefined
YET_NotImplemented	-2	Not implemented
YET_NotSupported	-3	not support
YET_InvalidState	-4	Invalid state
YET_SDKUninited	-10	Uninitialized
YET_InvalidHandle	-11	Invalid handle
YET_NoMemory	-12	No memory available
YET_NoHandle	-13	No handle available
YET_PreviewOpenBuffer	-14	Preview failed to open buffer
YET_PreviewOpenInput	-15	Preview open input failed
YET_PreviewOpenDisplay	-16	Preview open display failed
YET_PreviewRecoverBegin	-17	Preview auto recovery begins
YET_PreviewRecoverEnd	-18	Preview automatic recovery ends
YET_SocketOpen	-100	Socket open failed
YET_SocketConn	-101	Socket connection failed
YET_SocketSend	-102	Socket failed to send
YET_SocketRecv	-103	Socket reception failed
YET_SocketData	-104	Socket data is wrong
YET_FileOpen	-201	File open failed
YET_FileWrite	-202	File write failed
YET_FileRead	-203	File read failed
YET_FileType	-204	Incorrect file type
YET_InvalidPngData	-205	Invalid png data
YET_FfmpegDecode	-301	Ffmpeg decoding failed
YET_FfmpegMuxOpen	-302	Ffmpeg muxer open failed
YET_FfmpegMuxWrite	-303	Ffmpeg muxer write failed
YET_FfmpegMuxClose	-304	Ffmpeg muxer close failed