



ASICamera Software Development Kit

**Revision: V1, 2
2016.9.8**

All material in this publication is subject to change without notice and is
copyright Zhen Wang Optical company.



1 Introduction	4
2 Definition of enum-type and struct	4
2.1 enum Control_TYPE	4
2.2 enum IMG_TYPE	4
2.3 enum GuideDirections	5
2.4 enum BayerPattern	5
2.5 enum EXPOSURE_STATUS	5
2.6 typedef struct _ASIID	5
3 Function declaration	5
3.1 getNumberOfConnectedCameras	5
3.2 getCameraModel	6
3.3 openCamera	6
3.4 initCamera	6
3.5 closeCamera	6
3.6 isColorCam	6
3.7 getPixelSize	6
3.8 getColorBayer	6
3.9 EnableDarkSubtract	6
3.10 DisableDarkSubtract	6
3.11 isAvailable	6
3.12 isAutoSupported	7
3.13 getValue	7
3.14 getMin	7
3.15 getMax	7
3.16 setValue	7
3.17 setAutoPara	7
3.18 getAutoPara	7
3.19 getMaxWidth	8
3.20 getMaxHeight	8
3.21 getWidth	8
3.22 getHeight	8
3.23 getStartX	8
3.24 getStartY	8
3.25 getSensorTemp	8
3.26 getDroppedFrames	8
3.27 SetMisc	8
3.28 GetMisc	8
3.29 isBinSupported	9
3.30 isImgTypeSupported	9
3.31 getBin	9
3.32 setStartPos	9
3.33 setImageFormat	9
3.34 getImgType	9



3.35 startCapture	9
3.36 stopCapture	10
3.37 getImageData	10
3.38 pulseGuide	10
3.39 startExposure	10
3.40 getExpStatus	10
3.41 getImageAfterExp	10
3.42 stopExposure	11
3.43 isUSB3Host	11
3.44 isCoolerCam	11
3.45 GetID	11
3.46 SetID	11
3.47 isUSB3Camera	11
3.48 getElectronsPerADU	11
4 Suggested call sequence	11
4.1 Initialization	11
4.2 Get and set control value	11
4.3 Capture image	12
4.4 Close camera	12



1 Introduction

This SDK is used to operate ASI serial cameras, can be used by C, C++, C# and other develop tools, is suit for Windows, Linux, OSX operating system of x86 and x64.

Header file: ASICamera.h

Under Windows the import library and dynamic library: ASICamera.lib、ASICamera.dll

Under Linux the dynamic library and static library: ASICamera.so、ASICamera.a

Under OSX the dynamic library and static library: ASICamera.dylib、ASICamera.a

Installation method:

Under Windows, extract the downloaded zip file to any directory, and add DLL's path to system environment variables, sometimes logout and re-login is required

2 Definition of enum-type and struct

2.1 enum Control_TYPE

```
{
    CONTROL_GAIN=0, //gain
    CONTROL_EXPOSURE, // exposure time(us)
    CONTROL_GAMMA, // gamma
    CONTROL_WB_R, // red component of white balance
    CONTROL_WB_B, // blue component of white balance
    CONTROL_BRIGHTNESS, // offset
    CONTROL_BANDWIDTHOVERLOAD, // USB band width
    CONTROL_OVERCLOCK, // over clock
    CONTROL_TEMPERATURE, // sensor temperature, 10 times the actual temperature
    CONTROL_HARDWAREBIN, // hardware bin
    CONTROL_HIGHSPEED, //high speed mode
    CONTROL_COOLERPOWERPERC, // cooler power percent(only cool camera)
    CONTROL_TARGETTEMP, // sensor's target temperature(only cool camera), don't multiply
    by 10
    CONTROL_COOLER_ON, // open cooler(only cool camera)
    CONTROL_MONO_BIN, //lead to less grid at software bin mode for color camera
    CONTROL_FAN_ON
};
    Camera control type
```

2.2 enum IMG_TYPE

```
{
    IMG_RAW8=0, //1 byte every pixel
    IMG_RGB24, // Each pixel consists of RGB, 3 bytes totally (color cameras only)
```



```
IMG_RAW16,// 2 byte every pixel
IMG_Y8,// mono mode, 1 byte every pixel (color cameras only)
};
    Image type
```

2.3 enum GuideDirections

```
{
    guideNorth=0,
    guideSouth,
    guideEast,
    guideWest
};
    Moving direction when guiding
```

2.4 enum BayerPattern

```
{
    BayerRG=0,
    BayerBG,
    BayerGR,
    BayerGB
};
    Bayer filter type
```

2.5 enum EXPOSURE_STATUS

```
{
    EXP_IDLE = 0, // idle, ready to start exposure
    EXP_WORKING, // exposing
    EXP_SUCCESS, // exposure successfully, image can be read out
    EXP_FAILED, // exposure fail, require restart exposure
};
    Used under snap mode to describe exposure status
```

2.6 typedef struct _ASIID

```
{
    unsigned char id[8];
}ASIID;
    ID to be write into camera flash, 8 bytes totally
```

3 Function declaration

3.1 getNumberOfConnectedCameras

Syntax: int getNumberOfConnectedCameras();



Usage: get the count of connected cameras

3.2 getCameraModel

Syntax: char* getCameraModel(int camIndex);

Usage: get camera name(0 is the first one)

3.3 openCamera

Syntax: bool openCamera(int camIndex);

Usage: open a camera

3.4 initCamera

Syntax: bool initCamera();

Usage: initialize the camera, called after openCamera()

3.5 closeCamera

Syntax: void closeCamera();

Usage: close camera

3.6 isColorCam

Syntax: bool isColorCam();

Usage: whether camera is color

3.7 getPixelSize

Syntax: double getPixelSize();

Usage: get pixel size(um)

3.8 getColorBayer

Syntax: BayerPattern getColorBayer();

Usage: get Bayer filter type

3.9 EnableDarkSubtract

Syntax: bool EnableDarkSubtract(char *BMPPath);

Usage: enable dark subtract function

Description:

char *BMPPath: path of dark field image(.bmp)

return : success or not

Notes: dark field image is get by camera's direct show driver, located in capture application's menu "video capture filter"->"ROI and others" table

3.10 DisableDarkSubtract

Syntax: void DisableDarkSubtract();

Usage: disable dark subtract function

3.11 isAvailable



Syntax: bool isAvailable(Control_TYPE control) ;

Usage: Whether the control type is supported

3.12 isAutoSupported

Syntax: bool isAutoSupported(Control_TYPE control) ;

Usage: Whether the control type supports auto adjust

3.13 getValue

Syntax: int getValue(Control_TYPE control, bool *pbAuto) ;

Usage: get value of control

Description:

Control_TYPE control: control type

bool *pbAuto: is auto adjust?

Return: value

3.14 getMin

Syntax: int getMin(Control_TYPE control) ;

Usage: get the minimum value of the control type

3.15 getMax

Syntax: int getMax(Control_TYPE control) ;

Usage: get the maximum value of the control value

3.16 setValue

Syntax: void setValue(Control_TYPE control, int value, bool autose);

Usage: set value of the control value

Description:

Control_TYPE control: control type

int value: value

bool autose: is auto adjust?

Notes: when set to auto adjust (autose = true), IValue should be current value

3.17 setAutoPara

Syntax: void setAutoPara(int iMaxGain, int iMaxExp, int iDestBrightness);

Usage: set parameters of auto adjusting

Description:

int iMaxGain: maximum gain value

int iMaxExp: maximum value of exposure time, unit is second

int iDestBrightness: target brightness

3.18 getAutoPara

Syntax: void getAutoPara(int *pMaxGain, int *pMaxExp, int *pDestBrightness);

Usage: get parameters of auto adjusting

Description:



int *pMaxGain: maximum gain value
int *pMaxExp: maximum value of exposure time, unit is second
int *pDestBrightness: target brightness

3.19 getMaxWidth

Syntax: int getMaxWidth();

Usage: get maximum image width

3.20 getMaxHeight

Syntax: int getMaxHeight();

Usage: get maximum image height

3.21 getWidth

Syntax: int getWidth();

Usage: get current image width

3.22 getHeight

Syntax: int getHeight();

Usage: get current image height

3.23 getStartX

Syntax: int getStartX();

Usage: get start position of x-axis when ROI

3.24 getStartY

Syntax: int getStartY();

Usage: get start position of y-axis when ROI

3.25 getSensorTemp

Syntax: float getSensorTemp();

Usage: get sensor temperature, accurate to one decimal place

3.26 getDroppedFrames

Syntax: unsigned long getDroppedFrames();

Usage: get dropped frames' count during video capture

3.27 SetMisc

Syntax: void SetMisc(bool bFlipRow, bool bFlipColumn);

Usage: flip image

Description:

bool bFlipRow: horizontal flip

bool bFlipColumn: vertical flip

3.28 GetMisc



Syntax: void GetMisc(bool * pbFlipRow, bool * pbFlipColumn);

Usage: get image flip

Description:

bool * pbFlipRow: horizontal flip

bool * pbFlipColumn: vertical flip

3.29 isBinSupported

Syntax: bool isBinSupported(int binning);

Usage: whether given bin is supported

3.30 isImgTypeSupported

Syntax: bool isImgTypeSupported(IMG_TYPE img_type);

Usage: whether given image type is supported

3.31 getBin

Syntax: int getBin();

Usage: get current bin value

3.32 setStartPos

Syntax: void setStartPos(int startx, int starty);

Usage: set start position of ROI

Description:

int startx: start position of x-axis

int starty: start position of y-axis

Notes: the position is relative to the image after binning. call this function to change ROI area to the origin after ASISetROIFormat, because ASISetROIFormat will change ROI to the center.

3.33 setImageFormat

Syntax: bool setImageFormat(int width, int height, int binning, IMG_TYPE img_type);

Usage: set ROI size and image type

Description:

int width: image width

int height: image height

int binning: bin value

IMG_TYPE img_type: image type

Return: success or not

Notes: make sure iWidth%8=0, iHeight%2=0. For USB2.0 camera ASI120, make sure iWidth*iHeight%1024=0, otherwise setting will be failed.

3.34 getImgType

IMG_TYPE getImgType();

Usage: get image type

3.35 startCapture



Syntax: void startCapture();

Usage: start video capture

3.36 stopCapture

Syntax: void stopCapture();

Usage: stop video capture

3.37 getImageData

Syntax: bool getImageData(unsigned char* buffer, int bufSize, int waitms);

Usage: after startCapture(), call this function to get image continuously

Description:

unsigned char* buffer: pointer to image buffer

int bufSize: size of buffer

int waitms: wait time, unit is ms, -1 means wait forever

Notes:

If read out speed isn't fast enough, the frame will be discard

bufSize Byte length: for RAW8 and Y8, bufSize >= image_width*image_height, for RAW16, bufSize >= image_width*image_height *2, for RGB8, bufSiz >= image_width*image_height *3

suggested waitms value: exposure_time*2

3.38 pulseGuide

Syntax: void pulseGuide(GuideDirections direction, int timems);

Usage: send ST4 guiding pulse, only the camera with ST4 port support

Description:

GuideDirections direction: guiding direction

int timems: duration in ms

3.39 startExposure

Syntax: void startExposure();

Usage: start snap

3.40 getExpStatus

Syntax: enum EXPOSURE_STATUS getExpStatus();

Usage: get snap status

Notes: after snap is started, the status should be checked continuously

3.41 getImageAfterExp

Syntax: bool getImageAfterExp(unsigned char* buffer, int bufSize);

Usage: get image after snap successfully

Description:

unsigned char* buffer: pointer to image buffer

int bufSize: buffer size

Notes: bufSize refer to ASIGetVideoData ()



3.42 stopExposure

Syntax: void stopExposure();

Usage: stop snap

Notes: if exposure status is success after stop exposure, image can still be read out

3.43 isUSB3Host

Syntax: bool isUSB3Host();

Usage: whether camera works under USB3 status

3.44 isCoolerCam

Syntax: bool isCoolerCam();

Usage: whether camera is equipped with cooler

3.45 GetID

Syntax: bool GetID(ASIID *pID);

Usage: get camera id stored in flash, only available for USB3.0 camera

3.46 SetID

Syntax: bool SetID(ASIID ID);

Usage: write camera id to flash, only available for USB3.0 camera

3.47 isUSB3Camera

Syntax: bool isUSB3Camera();

Usage: whether this is USB3.0 camera

3.48 getElectronsPerADU

float getElectronsPerADU();

Usage: get system gain of sensor in e-/ADU

4 Suggested call sequence

4.1 Initialization

Get count of connected cameras-->--> getNumberOfConnectedCameras

Get cameras' name --> getCameraModel

Open camera --> openCamera

Initialize-->initCamera

Set image size and format--> setImageFormat

Set start position when ROI --> setStartPos

4.2 Get and set control value

getValue

setValue

allowed during capture

4.3 Capture image

There are two mode: video and snap mode. Images are captured continuously under video mode, and only single image is captured under snap mode.

- video mode

Start video capture --> startCapture

Stop video capture --> stopCapture

It is suggested that get and save data in single thread:

```
while(1)
{
    getImageData
    ...
}
```

- snap mode

startExposure

while(1)

```
{
    status = getExpStatus
    ...
}
```

Cancel exposure: stopExposure

if(status == EXP_SUCCESS) //get image if snap successfully
 getImageAfterExp

4.4 Close camera

closeCamera/release resource